Welcome to the *Ubuntu Server Guide*!

Changes, Errors, and Bugs

This is the current edition for Ubuntu 20.04 LTS, Focal Fossa. Ubuntu serverguides for previous LTS versions: 18.04 (PDF), 16.04 (PDF).

If you find any errors or have suggestions for improvements to pages, please use the link at the bottom of each topic titled: "Help improve this document in the forum." This link will take you to the Server Discourse forum for the specific page you are viewing. There you can share your comments or let us know about bugs with each page.

Offline

Download this guide as a PDF

Support

There are a couple of different ways that Ubuntu Server Edition is supported: commercial support and community support. The main commercial support (and development funding) is available from Canonical, Ltd. They supply reasonably- priced support contracts on a per desktop or per server basis. For more information see the Ubuntu Advantage page.

Community support is also provided by dedicated individuals and companies that wish to make Ubuntu the best distribution possible. Support is provided through multiple mailing lists, IRC channels, forums, blogs, wikis, etc. The large amount of information available can be overwhelming, but a good search engine query can usually provide an answer to your questions. See the Ubuntu Support page for more information.

Installation

This chapter provides a quick overview of installing Ubuntu 20.04 Server Edition. For more detailed instructions, please refer to the Ubuntu Installation Guide.

Preparing to Install

This section explains various aspects to consider before starting the installation.

System Requirements

Ubuntu 20.04 Server Edition provides a common, minimalist base for a variety of server applications, such as file/print services, web hosting, email hosting, etc. This edition supports five (5) major architectures: AMD64, ARM, POWER9, LinuxONE and z Systems, and introduces initial support for RISC-V.

The "live server" installer (sometimes called "Ubiquity for Servers" or simply "Subiquity") provides a user-friendly and fast installation experience.

The table below lists the recommended minimum hardware specifications. Depending on your needs, you might manage with less than this, but it is not generally recommended.

Install Type	CPU	RAM	Hard Drive Space	
Server (Standard)	1 gigahertz	512 megabytes	1.5 gigabyte	2.5 gigabytes
live server	1 gigahertz (amd64 only)	1 gigabyte	1.5 gigabyte	n/a
Server (Minimal)	300 megahertz	384 megabytes	1.5 gigabytes	2.5 gigabytes

Server and Desktop Differences

The *Ubuntu Server Edition* and the *Ubuntu Desktop Edition* use the same apt repositories, making it just as easy to install a *server* application on the Desktop Edition as on the Server Edition.

One major difference is that the graphical environment used for the Desktop Edition is not installed for the Server. This includes the graphics server itself, the graphical utilities and applications, and the various user-supporting services needed by desktop users.

The Server Edition's kernel used to be tuned differently from Desktop, but currently, both rely on the same Linux configuration.

Backing Up

- Before installing Ubuntu Server Edition you should make sure all data on the system is backed up.

  If this is not the first time an operating system has been installed on your computer, it is likely you will need to re-partition your disk to make room for Ubuntu.

  Any time you partition your disk, you should be prepared to lose everything on the disk should you make a mistake or something goes wrong during partitioning. The programs used in installation are quite reliable, most have seen years of use, but they also perform destructive actions.

Installing using the live server installer

The basic steps to install Ubuntu Server Edition are the same as those for installing any operating system. Unlike the *Desktop Edition*, the *Server Edition* does not include a graphical installation program. The Live Server installer uses a text-based console interface which runs on the default virtual console. The interface can be entirely driven by the enter, up and down arrow keys (with some occasional typing).

During the installation, you can switch to a different console (by pressing Ctrl-Alt-F<n> or Ctrl-Alt-Right) to get access to a shell, if needed. Up to the point where the installation begins, you can use the "back" button to go back to previous screens and choose different options.

- Download the appropriate ISO file from the Ubuntu Server Download Page.
- Boot the system from media (e.g. USB key) containing the ISO file.
- At the boot prompt you will be asked to select a language.
- From the main boot menu there are some additional options to install Ubuntu Server Edition. You can install a basic Ubuntu Server, check the installation media for defects, check the system's RAM, or boot from first hard disk. The rest of this section will cover the basic Ubuntu Server install.
- After booting into the installer, it will ask you which language to use.
- Next, the installation process begins by asking for your keyboard layout. You can ask the installer to attempt auto-detecting it, or you can select it manually from a list. Later stages of the installation will require you to type ASCII characters, so if the layout you select does not allow that, you will be

prompted for a key combination to switch between a layout that does and the one you select. The default keystroke for this is Alt + Shift.

- Next, the installer offers the choice to install the system as a vanilla Ubuntu server, a MAAS bare-metal cloud rack controller or a MAAS region controller. If you select one of the MAAS options you will be asked for some details.
- The installer configures the network to run DHCP on each network interface. If this is not sufficient to get access to the internet you should configure at least one interface manually. Select an interface to configure it.
- If the Ubuntu archive can only be accessed via a proxy in your environment, it can be entered on the next screen. Leave the field blank if it is not required.
- You can then choose to let the installer use an entire disk or configure the partitioning manually. The first disk you create a partition on will be selected as the boot disk and have an extra partition created on it to contain the bootloader; you can move the boot partition to a different drive with the "Select as boot disk" button.

  Once you move on from this screen, the installation progress will begin. It will not be possible to move back to this or previous screens and any data on the disks you have configured the installer to use will be lost.
- The next screen configures the initial user for the system. You can import SSH keys from Launchpad or Github but a password is still required to be set, as this user will have *root* access through the sudo utility.
- The final screen shows the progress of the installer. Once the installation has completed, you will be prompted to reboot into your newly installed system.

Advanced Installation

Software RAID

Redundant Array of Independent Disks "RAID" is a method of using multiple disks to provide different balances of increasing data reliability and/or increasing input/output performance, depending on the RAID level being used. RAID is implemented in either software (where the operating system knows about both drives and actively maintains both of them) or hardware (where a special controller makes the OS think there's only one drive and maintains the drives 'invisibly').

The RAID software included with current versions of Linux (and Ubuntu) is based on the 'mdadm' driver and works very well, better even than many so-called 'hardware' RAID controllers. This section will guide you through installing Ubuntu Server Edition using two RAID1 partitions on two physical hard drives, one for / and another for *swap*.

RAID Configuration

Follow the installation steps until you get to the *Guided storage configuration* step, then:

Select *Custom storage layout*.

Create the */boot* partition in a local disk. So select one of the devices listed in available devices and *Add GPT Partition*. Next, enter the partition size, then choose the desired *Format* (*ext4*) and */boot* as mount point. And finally, select *Create*.

Now to create the RAID device select *Create software RAID (md)* under *AVAILABLE DEVICES*.

Add the name of the RAID disk (the default is *md0*).

For this example, select *"1 (mirrored)"* in *RAID level*, but if you are using a different setup choose the appropriate type (RAID0 RAID1 RAID5 RAID6 RAID10).

> **Note**
>
> In order to use *RAID5*, *RAID6* and *RAID10* you need more than *two* drives. Using RAID0 or RAID1 only *two* drives are required.

Select the devices that will be used by this RAID device. The real devices can be marked as *active* or *spare*, by default it becomes *active* when is selected.

Select the *Size* of the RAID device.

Select *Create*.

The new RAID device (*md0* if you did not change the default) will show up in the available devices list, with *software RAID 1* type and the chosen size.

Repeat steps above for the other RAID devices.

Partitioning

Select the RAID 1 device created (*md0*) then select *"Add GPT Partition"*.

Next, select the *Size* of the partition. This partition will be the *swap* partition, and a general rule for swap size is twice that of RAM. Enter the partition size, then choose *swap* in Format. And finally, select *Create*.

> **Note**
>
> A swap partition size of twice the available RAM capacity may not always be desirable, especially on systems with large amounts of RAM. Calculating the swap partition size for servers is highly dependent on how the system is going to be used.

For the / partition once again select the RAID 1 device then *"Add GPT Partition"*.

Use the rest of the free space on the device, choose the format (default is *ext4*) and select / as mount point, then *Create*.

Repeat steps above for the other partitions.

Once it is finished select *"Done"*.

The installation process will then continue normally.

Degraded RAID

At some point in the life of the computer a disk failure event may occur. When this happens, using Software RAID, the operating system will place the array into what is known as a *degraded* state.

If the array has become degraded, due to the chance of data corruption, by default Ubuntu Server Edition will boot to *initramfs* after thirty seconds. Once the initramfs has booted there is a fifteen second prompt giving you the option to go ahead and boot the system, or attempt manual recover. Booting to the initramfs prompt may or may not be the desired behavior, especially if the machine is in a remote location. Booting to a degraded array can be configured several ways:

- The dpkg-reconfigure utility can be used to configure the default behavior, and during the process you will be queried about additional settings related to the array. Such as monitoring, email alerts, etc. To reconfigure mdadm enter the following:

  ```
  sudo dpkg-reconfigure mdadm
  ```

- The dpkg−reconfigure mdadm process will change the /etc/initramfs−tools/conf.d/mdadm configuration file. The file has the advantage of being able to pre-configure the system's behavior, and can also be manually edited:

  ```
  BOOT_DEGRADED=true
  ```

  Note

  The configuration file can be overridden by using a Kernel argument.

- Using a Kernel argument will allow the system to boot to a degraded array as well:
  - When the server is booting press Shift to open the Grub menu.
  - Press e to edit your kernel command options.
  - Press the down arrow to highlight the kernel line.
  - Add *"bootdegraded=true"* (without the quotes) to the end of the line.
  - Press Ctrl+x to boot the system.

Once the system has booted you can either repair the array see the next section for details, or copy important data to another machine due to major hardware failure.

RAID Maintenance

The mdadm utility can be used to view the status of an array, add disks to an array, remove disks, etc:

- To view the status of an array, from a terminal prompt enter:

  ```
  sudo mdadm –D /dev/md0
  ```

  The *-D* tells mdadm to display *detailed* information about the /dev/md0 device. Replace /dev/md0 with the appropriate RAID device.

- To view the status of a disk in an array:

  ```
  sudo mdadm –E /dev/sda1
  ```

  The output if very similar to the mdadm −D command, adjust /dev/sda1 for each disk.

- If a disk fails and needs to be removed from an array enter:

  ```
  sudo mdadm —remove /dev/md0 /dev/sda1
  ```

  Change /dev/md0 and /dev/sda1 to the appropriate RAID device and disk.

- Similarly, to add a new disk:

  ```
  sudo mdadm —add /dev/md0 /dev/sda1
  ```

Sometimes a disk can change to a *faulty* state even though there is nothing physically wrong with the drive. It is usually worthwhile to remove the drive from the array then re-add it. This will cause the drive to re-sync with the array. If the drive will not sync with the array, it is a good indication of hardware failure.

The /proc/mdstat file also contains useful information about the system's RAID devices:

```
cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [
    raid10]
md0 : active raid1 sda1[0] sdb1[1]
      10016384 blocks [2/2] [UU]
```

```
unused devices: <none>
```

The following command is great for watching the status of a syncing drive:

```
watch -n1 cat /proc/mdstat
```

Press *Ctrl+c* to stop the watch command.

If you do need to replace a faulty drive, after the drive has been replaced and synced, grub will need to be installed. To install grub on the new drive, enter the following:

```
sudo grub-install /dev/md0
```

Replace /dev/md0 with the appropriate array device name.

Resources

The topic of RAID arrays is a complex one due to the plethora of ways RAID can be configured. Please see the following links for more information:

- Ubuntu Wiki Articles on RAID.
- Software RAID HOWTO
- Managing RAID on Linux

Logical Volume Manager (LVM)

Logical Volume Manger, or *LVM*, allows administrators to create *logical* volumes out of one or multiple physical hard disks. LVM volumes can be created on both software RAID partitions and standard partitions residing on a single disk. Volumes can also be extended, giving greater flexibility to systems as requirements change.

Overview

A side effect of LVM's power and flexibility is a greater degree of complication. Before diving into the LVM installation process, it is best to get familiar with some terms.

- *Physical Volume (PV):* physical hard disk, disk partition or software RAID partition formatted as LVM PV.
- *Volume Group (VG):* is made from one or more physical volumes. A VG can can be extended by adding more PVs. A VG is like a virtual disk drive, from which one or more logical volumes are carved.
- *Logical Volume (LV):* is similar to a partition in a non-LVM system. A LV is formatted with the desired file system (EXT3, XFS, JFS, etc), it is then available for mounting and data storage.

Installation

As an example this section covers installing Ubuntu Server Edition with /srv mounted on a LVM volume. During the initial install only one Physical Volume (PV) will be part of the Volume Group (VG). Another PV will be added after install to demonstrate how a VG can be extended.

There are several installation options for LVM in *Guided storage configuration* step:

- Select *"Use an entire disk"*, *"Set up this disk as an LVM group"*, and *Done*. This option will create a */boot* partition in the local disk and the rest of the disk space is allocated to the LVM group.
- Select *"Use an entire disk"*, *"Set up this disk as an LVM group"*, *"Encrypt the LVM group with LUKS"*, insert the password (and confirm it), and *Done*. The output is the same as described above but the LVM group is encrypted.
- Select *"Custom storage layout"*, and *Done*. At this time the only way to configure a system with both LVM and standard partitions, during installation, is to use this approach. This is the option used in this example.

Follow the installation steps until you get to the *Storage configuration* step, then:

Let's first create a */boot* partition in a local disk. Select the hard disk under AVAILABLE DEVICES, and *Add GPT Parition*. Add the size and format (*ext4*), then select */boot* as mount point. Finally, select *Create*. The */boot* partition will be listed under FILE SYSTEM SUMMARY.

Next, create standard *swap*, and */* partitions with whichever filesystem you prefer following the steps above.

Now the LVM volume group will be created. Select *"Create volume group (LVM)"*. Enter a name for the volume group (default is *vg0*), select the device (LVM physical volume) and the size, and choose *"Create"*. There is an option to encrypt your volume, if you want it encrypted select *"Create encrypted volume"* and enter a password (also confirm it). The brand new LVM group (if the default was not changed it is *vg0*) will be listed as a device in AVAILABLE DEVICES.

To create a LVM logical volume select the created LVM volume group and *"Create Logical Volume"*. Give it a name (default is *lv-0*), let's call it *lv-srv* since this will be used to mount */srv*. Insert the size of the volume, your preferred filesytem format, and select */srv* as mount point. Choose *"Create"*. The LVM logical volume mounted at */srv* will be listed in the FILESYSTEM SUMMARY.

Finally, select *"Done"*. Then confirm the changes and continue with the rest of the installation.

There are some useful utilities to view information about LVM:

- *pvdisplay:* shows information about Physical Volumes.
- *vgdisplay:* shows information about Volume Groups.
- *lvdisplay:* shows information about Logical Volumes.

Extending Volume Groups

Continuing with *srv* as an LVM volume example, this section covers adding a second hard disk, creating a Physical Volume (PV), adding it to the volume group (VG), extending the logical volume srv and finally extending the filesystem. This example assumes a second hard disk has been added to the system. In this example, this hard disk will be named /dev/sdb and we will use the entire disk as a physical volume (you could choose to create partitions and use them as different physical volumes)

> **Warning**
>
> Make sure you don't already have an existing /dev/sdb before issuing the commands below. You could lose some data if you issue those commands on a non-empty disk.

First, create the physical volume, in a terminal execute:

```
sudo pvcreate /dev/sdb
```

Now extend the Volume Group (VG):

```
sudo vgextend vg0 /dev/sdb
```

Use vgdisplay to find out the free physical extents - Free PE / size (the size you can allocate). We will assume a free size of 511 PE (equivalent to 2GB with a PE size of 4MB) and we will use the whole free space available. Use your own PE and/or free space.

The Logical Volume (LV) can now be extended by different methods, we will only see how to use the PE to extend the LV:

```
sudo lvextend /dev/vg0/srv -l +511
```

The *-l* option allows the LV to be extended using PE. The *-L* option allows the LV to be extended using Meg, Gig, Tera, etc bytes.

Even though you are supposed to be able to *expand* an ext3 or ext4 filesystem without unmounting it first, it may be a good practice to unmount it anyway and check the filesystem, so that you don't mess up the day you want to reduce a logical volume (in that case unmounting first is compulsory).

The following commands are for an *EXT3* or *EXT4* filesystem. If you are using another filesystem there may be other utilities available.

```
sudo umount /srv
sudo e2fsck -f /dev/vg0/srv
```

The *-f* option of e2fsck forces checking even if the system seems clean.

Finally, resize the filesystem:

```
sudo resize2fs /dev/vg0/srv
```

Now mount the partition and check its size.

```
mount /dev/vg0/srv /srv && df -h /srv
```

Resources

- See the Ubuntu Wiki LVM Articles.
- See the LVM HOWTO for more information.
- For more information on fdisk see the fdisk man page.

iSCSI

The iSCSI protocol can be used to install Ubuntu on systems with or without hard disks attached, and iBFT can be used to automate iSCSI setup on installation and boot.

Installation on a diskless system

The first steps of a diskless iSCSI installation are identical to the Installation using debian-installer section up to "Hard drive layout".

The installer will display a warning with the following message:

```
No disk drive was detected. If you know the name of the driver needed by your
    disk drive, you can select it from the list.
```

Select the item in the list titled *login to iSCSI targets.*

You will be prompted to enter an IP address to scan for iSCSI targets with a description of the format for the address. Enter the IP address for the location of your iSCSI target and navigate to *<continue>* then hit ENTER

If authentication is required in order to access the iSCSI device, provide the *username* in the next field. Otherwise, leave it blank.

If your system is able to connect to the iSCSI provider, you should see a list of available iSCSI targets where the operating system can be installed. The list should be similar to the following :

```
Select the iSCSI targets you wish to use.

iSCSI targets on 192.168.1.29:3260:

[ ] iqn.2016-03.TrustyS-iscsitarget:storage.sys0

<Go Back>                                    <Continue>
```

Select the iSCSI target that you want to use with the space bar. Use the arrow keys to navigate to the target that you want to select.

Navigate to *<Continue>* and hit ENTER.

If the connection to the iSCSI target is successful, you will be prompted with the *[!!] Partition disks* installation menu. The rest of the procedure is identical to any normal installation on attached disks. Once the installation is completed, you will be asked to reboot.

Installation on a system with disk attached

Again, the iSCSI installation on a normal server with one or many disks attached is identical to the Installation using debian-installer section until we reach the disk partitioning menu. Instead of using any of the Guided selection, we need to perform the following steps :

Navigate to the Manual menu entry

Select the Configure iSCSI Volumes menu entry

Choose the Log into iSCSI targets

You will be prompted to Enter an IP address to scan for iSCSI targets. with a description of the format for the address. Enter the IP address and navigate to *<continue>* then hit ENTER

If authentication is required in order to access the iSCSI device, provide the *username* in the next field or leave it blank.

If your system is able to connect to the iSCSI provider, you should see a list of available iSCSI targets where the operating system can be installed. The list should be similar to the following :

```
Select the iSCSI targets you wish to use.

iSCSI targets on 192.168.1.29:3260:

[ ] iqn.2016-03.TrustyS-iscsitarget:storage.sys0

<Go Back>                                    <Continue>
```

Select the iSCSI target that you want to use with the space bar. Use the arrow keys to navigate to the target that you want to select

Navigate to <Continue> and hit ENTER.

If successful, you will come back to the menu asking you to Log into iSCSI targets. Navigate to Finish and hit ENTER

The newly connected iSCSI disk will appear in the overview section as a device prefixed with SCSI. This is the disk that you should select as your installation disk. Once identified, you can choose any of the partitioning methods.

> **Warning**
>
> Depending on your system configuration, there may be other SCSI disks attached to the system. Be very careful to identify the proper device before proceeding with the installation. Otherwise, irreversible data loss may result from performing an installation on the wrong disk.

Installation with iBFT

In order to setup iSCSI based on the iBFT (iSCSI Boot Firmware Table) on the installation and boot, append these options at the installer prompt (or to the preseed file):

```
disk-detect/ibft/enable=true partman-iscsi/iscsi_auto=true
```

This should probe for iBFT information and configure network interface(s) and iSCSI target(s) accordingly during the installation, and configure system boot (initramfs) to do that too in order to find the root device.

> **Warning** The support for iBFT is available in the debian-installer on netboot images as of 2019-06-20 and (expected) on ISO images for the 18.04.3 point release and later.

Rebooting to an iSCSI target

The procedure is specific to your hardware platform. As an example, here is how to reboot to your iSCSI target using iPXE

```
iPXE> dhcp

Configuring (net0 52:54:00:a4:f2:a9)..... ok

iPXE> sanboot iscsi:192.168.1.29::::iqn.2016-03.TrustyS-iscsitarget:storage.
    sys0
```

If the procedure is successful, you should see the Grub menu appear on the screen.

Package Management

Ubuntu features a comprehensive package management system for installing, upgrading, configuring, and removing software. In addition to providing access to an organized base of over 60,000 software packages for your Ubuntu computer, the package management facilities also feature dependency resolution capabilities and software update checking.

Several tools are available for interacting with Ubuntu's package management system, from simple command-line utilities which may be easily automated by system administrators, to a graphical interface which is easy to use by those new to Ubuntu.

Introduction

Ubuntu's package management system is derived from the same system used by the Debian GNU/Linux distribution. The package files contain all of the necessary files, meta-data, and instructions to implement a particular functionality or software application on your Ubuntu computer.

Debian package files typically have the extension .deb, and usually exist in *repositories* which are collections of packages found online or on physical media, such as CD-ROM discs. Packages are normally in a pre-compiled binary format; thus installation is quick and requires no compiling of software.

Many packages use *dependencies*. Dependencies are additional packages required by the principal package in order to function properly. For example, the speech synthesis package festival depends upon the package alsa−utils, which is a package supplying the ALSA sound library tools needed for audio playback. In order for festival to function, it and all of its dependencies must be installed. The software management tools in Ubuntu will do this automatically.

Apt

The apt command is a powerful command-line tool, which works with Ubuntu's *Advanced Packaging Tool* (APT) performing such functions as installation of new software packages, upgrade of existing software packages, updating of the package list index, and even upgrading the entire Ubuntu system.

Some examples of popular uses for the apt utility:

- **Install a Package**: Installation of packages using the apt tool is quite simple. For example, to install the nmap network scanner, type the following:

  ```
  sudo apt install nmap
  ```

- **Remove a Package**: Removal of a package (or packages) is also straightforward. To remove the package installed in the previous example, type the following:

  ```
  sudo apt remove nmap
  ```

  Tip

  Multiple Packages: You may specify multiple packages to be installed or removed, separated by spaces.

  Notice

  Scripting: While apt is a command-line tool, it is intended to be used interactively, and not to be called from non-interactive scripts. The apt−get command should be used in scripts (perhaps with the −−quiet flag). For basic commands the syntax of the two tools is identical.

  Also, adding the −−purge option to apt remove will remove the package configuration files as well. This may or may not be the desired effect, so use with caution.

- **Update the Package Index**: The APT package index is essentially a database of available packages from the repositories defined in the /etc/apt/sources. list file and in the /etc/apt/sources. list .d directory. To update the local package index with the latest changes made in the repositories, type the following:

  ```
  sudo apt update
  ```

- **Upgrade Packages**: Over time, updated versions of packages currently installed on your computer may become available from the package repositories (for example security updates). To upgrade your system, first, update your package index as outlined above, and then type:

  ```
  sudo apt upgrade
  ```

For information on upgrading to a new Ubuntu release see Upgrading.

Actions of the apt command, such as installation and removal of packages, are logged in the /var/log/dpkg.log log file.

For further information about the use of APT, read the comprehensive APT User's Guide or type:

```
apt help
```

Aptitude

Launching Aptitude with no command-line options will give you a menu-driven, text-based front-end to the *Advanced Packaging Tool* (APT) system. Many of the common package management functions, such as installation, removal, and upgrade, can be performed in Aptitude with single-key commands, which are typically lowercase letters.

Aptitude is best suited for use in a non-graphical terminal environment to ensure proper functioning of the command keys. You may start the menu-driven interface of Aptitude as a normal user by typing the following command at a terminal prompt:

```
sudo aptitude
```

When Aptitude starts, you will see a menu bar at the top of the screen and two panes below the menu bar. The top pane contains package categories, such as *New Packages* and *Not Installed Packages*. The bottom pane contains information related to the packages and package categories.

Using Aptitude for package management is relatively straightforward, and the user interface makes common tasks simple to perform. The following are examples of common package management functions as performed in Aptitude:

- **Install Packages**: To install a package, locate the package via the *Not Installed Packages* package category, by using the keyboard arrow keys and the ENTER key. Highlight the desired package, then press the + key. The package entry should turn *green*, indicating that it has been marked for installation. Now press g to be presented with a summary of package actions. Press g again, and downloading and installation of the package will commence. When finished, press ENTER, to return to the menu.
- **Remove Packages**: To remove a package, locate the package via the *Installed Packages* package category, by using the keyboard arrow keys and the ENTER key. Highlight the desired package you wish to remove, then press the - key. The package entry should turn *pink*, indicating it has been marked for removal. Now press g to be presented with a summary of package actions. Press g again, and removal of the package will commence. When finished, press ENTER, to return to the menu.
- **Update Package Index**: To update the package index, simply press the u key. Updating of the package index will commence.
- **Upgrade Packages**: To upgrade packages, perform the update of the package index as detailed above, and then press the U key to mark all packages with updates. Now press g whereby you'll be presented with a summary of package actions. Press g again, and the download and installation will commence. When finished, press ENTER, to return to the menu.

The first column of the information displayed in the package list in the top pane, when actually viewing packages lists the current state of the package, and uses the following key to describe the state of the package:

- **i**: Installed package
- **c**: Package not installed, but package configuration remains on the system
- **p**: Purged from system

- **v**: Virtual package
- **B**: Broken package
- **u**: Unpacked files, but package not yet configured
- **C**: Half-configured - Configuration failed and requires fix
- **H**: Half-installed - Removal failed and requires a fix

To exit Aptitude, simply press the q key and confirm you wish to exit. Many other functions are available from the Aptitude menu by pressing the F10 key.

Command Line Aptitude

You can also use Aptitude as a command-line tool, similar to apt. To install the nmap package with all necessary dependencies, as in the apt example, you would use the following command:

```
sudo aptitude install nmap
```

To remove the same package, you would use the command:

```
sudo aptitude remove nmap
```

Consult the man pages for more details of command-line options for Aptitude.

dpkg

dpkg is a package manager for *Debian*-based systems. It can install, remove, and build packages, but unlike other package management systems, it cannot automatically download and install packages or their dependencies. **Apt and Aptitude are newer, and layer additional features on top of dpkg.** This section covers using dpkg to manage locally installed packages:

- To list all packages in the system's package database, including all packages, installed and uninstalled, from a terminal prompt type:

  ```
  dpkg -l
  ```

- Depending on the number of packages on your system, this can generate a large amount of output. Pipe the output through grep to see if a specific package is installed:

  ```
  dpkg -l | grep apache2
  ```

  Replace apache2 with any package name, part of a package name, or a regular expression.

- To list the files installed by a package, in this case the ufw package, enter:

  ```
  dpkg -L ufw
  ```

- If you are not sure which package installed a file, dpkg -S may be able to tell you. For example:

  ```
  dpkg -S /etc/host.conf
  base-files: /etc/host.conf
  ```

  The output shows that the /etc/host.conf belongs to the base-files package.

  > **Note**
  >
  > Many files are automatically generated during the package install process, and even though they are on the filesystem, dpkg -S may not know which package they belong to.

- You can install a local .deb file by entering:

  ```
  sudo dpkg -i zip_3.0-4_amd64.deb
  ```

  Change zip_3.0-4_amd64.deb to the actual file name of the local .deb file you wish to install.

- Uninstalling a package can be accomplished by:

  ```
  sudo dpkg -r zip
  ```

  Caution

  Uninstalling packages using dpkg, in most cases, is *NOT* recommended. It is better to use a package manager that handles dependencies to ensure that the system is in a consistent state. For example, using dpkg -r zip will remove the zip package, but any packages that depend on it will still be installed and may no longer function correctly.

For more dpkg options see the man page: man dpkg.

APT Configuration

Configuration of the *Advanced Packaging Tool* (APT) system repositories is stored in the /etc/apt/sources.list file and the /etc/apt/sources.list.d directory. An example of this file is referenced here, along with information on adding or removing repository references from the file.

You may edit the file to enable repositories or disable them. For example, to disable the requirement of inserting the Ubuntu CD-ROM whenever package operations occur, simply comment out the appropriate line for the CD-ROM, which appears at the top of the file:

```
# no more prompting for CD-ROM please
# deb cdrom:[DISTRO_APT_CD_NAME - Release i386 (20111013.1)]/ DISTRO_SHORT_
    CODENAME main restricted
```

Extra Repositories

In addition to the officially supported package repositories available for Ubuntu, there exist additional community-maintained repositories which add thousands more packages for potential installation. Two of the most popular are the *universe* and *multiverse* repositories. These repositories are not officially supported by Ubuntu, but because they are maintained by the community they generally provide packages which are safe for use with your Ubuntu computer.

Note

Packages in the *multiverse* repository often have licensing issues that prevent them from being distributed with a free operating system, and they may be illegal in your locality.

Warning

Be advised that neither the *universe* or *multiverse* repositories contain officially supported packages. In particular, there may not be security updates for these packages.

Many other package sources are available, sometimes even offering only one package, as in the case of package sources provided by the developer of a single application. You should always be very careful and cautious when using non-standard package sources, however. Research the source and packages carefully before performing any installation, as some package sources and their packages could render your system unstable or non-functional in some respects.

By default, the *universe* and *multiverse* repositories are enabled but if you would like to disable them edit /etc/apt/sources.list and comment the following lines:

```
deb http://archive.ubuntu.com/ubuntu DISTRO_SHORT_CODENAME universe multiverse
deb-src http://archive.ubuntu.com/ubuntu DISTRO_SHORT_CODENAME universe
    multiverse

deb http://us.archive.ubuntu.com/ubuntu/ DISTRO_SHORT_CODENAME universe
deb-src http://us.archive.ubuntu.com/ubuntu/ DISTRO_SHORT_CODENAME universe
deb http://us.archive.ubuntu.com/ubuntu/ DISTRO_SHORT_CODENAME-updates
    universe
deb-src http://us.archive.ubuntu.com/ubuntu/ DISTRO_SHORT_CODENAME-updates
    universe

deb http://us.archive.ubuntu.com/ubuntu/ DISTRO_SHORT_CODENAME multiverse
deb-src http://us.archive.ubuntu.com/ubuntu/ DISTRO_SHORT_CODENAME multiverse
deb http://us.archive.ubuntu.com/ubuntu/ DISTRO_SHORT_CODENAME-updates
    multiverse
deb-src http://us.archive.ubuntu.com/ubuntu/ DISTRO_SHORT_CODENAME-updates
    multiverse

deb http://security.ubuntu.com/ubuntu DISTRO_SHORT_CODENAME-security universe
deb-src http://security.ubuntu.com/ubuntu DISTRO_SHORT_CODENAME-security
    universe
deb http://security.ubuntu.com/ubuntu DISTRO_SHORT_CODENAME-security
    multiverse
deb-src http://security.ubuntu.com/ubuntu DISTRO_SHORT_CODENAME-security
    multiverse
```

Automatic Updates

The unattended-upgrades package can be used to automatically install updated packages and can be configured to update all packages or just install security updates. First, install the package by entering the following in a terminal:

```
sudo apt install unattended-upgrades
```

To configure unattended-upgrades, edit /etc/apt/apt.conf.d/50unattended-upgrades and adjust the following to fit your needs:

```
Unattended-Upgrade::Allowed-Origins {
        "${distro_id}:${distro_codename}";
        "${distro_id}:${distro_codename}-security";
//      "${distro_id}:${distro_codename}-updates";
//      "${distro_id}:${distro_codename}-proposed";
//      "${distro_id}:${distro_codename}-backports";
};
```

Certain packages can also be *blacklisted* and therefore will not be automatically updated. To blacklist a package, add it to the list:

```
Unattended-Upgrade::Package-Blacklist {
//      "vim";
//      "libc6";
//      "libc6-dev";
//      "libc6-i686";
};
```

Note

The double "//" serve as comments, so whatever follows "//" will not be evaluated.

To enable automatic updates, edit /etc/apt/apt.conf.d/20auto−upgrades and set the appropriate apt configuration options:

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::AutocleanInterval "7";
APT::Periodic::Unattended-Upgrade "1";
```

The above configuration updates the package list, downloads, and installs available upgrades every day. The local download archive is cleaned every week. On servers upgraded to newer versions of Ubuntu, depending on your responses, the file listed above may not be there. In this case, creating a new file of this name should also work.

Note

You can read more about apt Periodic configuration options in the /etc/cron.daily/apt−compat script header.

The results of unattended-upgrades will be logged to /var/log/unattended−upgrades.

Notifications

Configuring Unattended−Upgrade::Mail in /etc/apt/apt.conf.d/50unattended−upgrades will enable unattended-upgrades to email an administrator detailing any packages that need upgrading or have problems.

Another useful package is apticron. apticron will configure a cron job to email an administrator information about any packages on the system that have updates available, as well as a summary of changes in each package.

To install the apticron package, in a terminal enter:

```
sudo apt install apticron
```

Once the package is installed edit /etc/apticron/apticron.conf, to set the email address and other options:

```
EMAIL="root@example.com"
```

References

Most of the material covered in this chapter is available in man pages, many of which are available online.

- The InstallingSoftware Ubuntu wiki page has more information.
- For more dpkg details see the dpkg man page.
- The APT User's Guide and apt man page contain useful information regarding apt usage.
- See the aptitude user's manual for more aptitude options.
- The Adding Repositories HOWTO (Ubuntu Wiki) page contains more details on adding repositories.

Kernel Crash Dump

Introduction

A Kernel Crash Dump refers to a portion of the contents of volatile memory (RAM) that is copied to disk whenever the execution of the kernel is disrupted. The following events can cause a kernel disruption :

- Kernel Panic
- Non Maskable Interrupts (NMI)
- Machine Check Exceptions (MCE)
- Hardware failure
- Manual intervention

For some of those events (panic, NMI) the kernel will react automatically and trigger the crash dump mechanism through *kexec*. In other situations a manual intervention is required in order to capture the memory. Whenever one of the above events occurs, it is important to find out the root cause in order to prevent it from happening again. The cause can be determined by inspecting the copied memory contents.

Kernel Crash Dump Mechanism

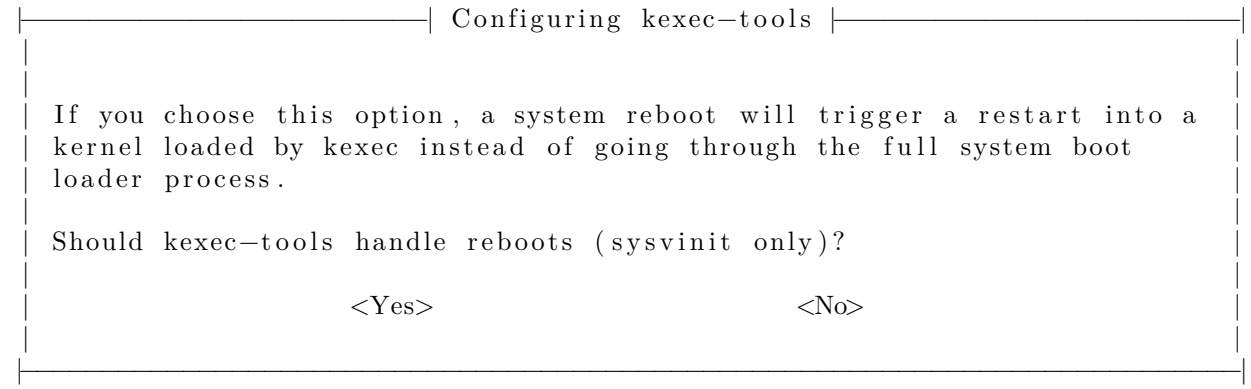
When a kernel panic occurs, the kernel relies on the *kexec* mechanism to quickly reboot a new instance of the kernel in a pre-reserved section of memory that had been allocated when the system booted (see below). This permits the existing memory area to remain untouched in order to safely copy its contents to storage.

Installation

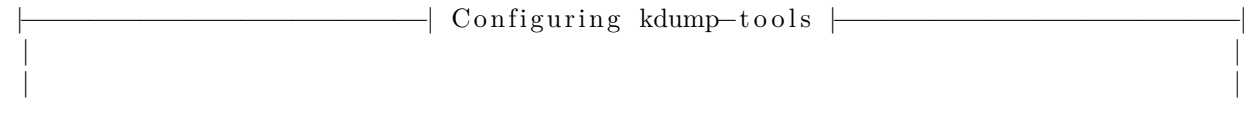
The kernel crash dump utility is installed with the following command:

```
sudo apt install linux-crashdump
```

> **Note**
>
> Starting with 16.04, the kernel crash dump mechanism is enabled by default. During the installation, you will be prompted with the following dialogs.



Select Yes to hook up kexec-tools for all reboots.



```
| If you choose this option, the kdump-tools mechanism will be enabled. A  |
| reboot is still required in order to enable the crashkernel kernel       |
| parameter.                                                               |
|                                                                          |
| Should kdump-tools be enabled be default?                                |
|                                                                          |
|                    <Yes>                        <No>                     |
|                                                                          |
|--------------------------------------------------------------------------|
```

Yes should be selected here as well, to enable kdump−tools.

If you ever need to manually enable the functionality, you can use the dpkg−reconfigure kexec−tools and dpkg−reconfigure kdump−tools commands and answer Yes to the questions. You can also edit /etc/default /kexec and set parameters directly:

```
# Load a kexec kernel (true/false)
LOAD_KEXEC=true
```

As well, edit /etc/default/kdump−tools to enable kdump by including the following line:

```
USE_KDUMP=1
```

If a reboot has not been done since installation of the linux−crashdump package, a reboot will be required in order to activate the crashkernel= boot parameter. Upon reboot, kdump−tools will be enabled and active.

If you enable kdump−tools after a reboot, you will only need to issue the kdump−config load command to activate the kdump mechanism.

You can view the current status of kdump via the command kdump−config show. This will display something like this:

```
DUMP_MODE:          kdump
USE_KDUMP:          1
KDUMP_SYSCTL:       kernel.panic_on_oops=1
KDUMP_COREDIR:      /var/crash
crashkernel addr:
   /var/lib/kdump/vmlinuz
kdump initrd:
   /var/lib/kdump/initrd.img
current state:      ready to kdump
kexec command:
  /sbin/kexec -p --command-line="..." --initrd=...
```

This tells us that we will find core dumps in /var/crash.

Configuration

In addition to local dump, it is now possible to use the remote dump functionality to send the kernel crash dump to a remote server, using either the *SSH* or *NFS* protocols.

Local Kernel Crash Dumps

Local dumps are configured automatically and will remain in use unless a remote protocol is chosen. Many configuration options exist and are thoroughly documented in the /etc/default/kdump−tools file.

Remote Kernel Crash Dumps using the SSH protocol

To enable remote dumps using the *SSH* protocol, the /etc/default/kdump−tools must be modified in the following manner :

```
# ------------------------------------------------------------------------
# Remote dump facilities:
# SSH - username and hostname of the remote server that will receive the dump
#       and dmesg files.
# SSH_KEY - Full path of the ssh private key to be used to login to the remote
#           server. use kdump-config propagate to send the public key to the
#           remote server
# HOSTTAG - Select if hostname of IP address will be used as a prefix to the
#           timestamped directory when sending files to the remote server.
#           'ip' is the default.
SSH="ubuntu@kdump-netcrash"
```

The only mandatory variable to define is SSH. It must contain the username and hostname of the remote server using the format {username}@{remote server}.

SSH_KEY may be used to provide an existing private key to be used. Otherwise, the kdump−config propagate command will create a new keypair. The HOSTTAG variable may be used to use the hostname of the system as a prefix to the remote directory to be created instead of the IP address.

The following example shows how kdump−config propagate is used to create and propagate a new keypair to the remote server :

```
sudo kdump-config propagate
Need to generate a new ssh key...
The authenticity of host 'kdump-netcrash (192.168.1.74)' can't be established.
ECDSA key fingerprint is SHA256:iMp+5Y28qhbd+tevFCWrEXykDd4dI3yN4OVlu3CBBQ4.
Are you sure you want to continue connecting (yes/no)? yes
ubuntu@kdump-netcrash's password:
propagated ssh key /root/.ssh/kdump_id_rsa to server ubuntu@kdump-netcrash
```

The password of the account used on the remote server will be required in order to successfully send the public key to the server

The kdump−config show command can be used to confirm that kdump is correctly configured to use the SSH protocol :

```
kdump-config show
DUMP_MODE:          kdump
USE_KDUMP:          1
KDUMP_SYSCTL:       kernel.panic_on_oops=1
KDUMP_COREDIR:      /var/crash
crashkernel addr: 0x2c000000
   /var/lib/kdump/vmlinuz: symbolic link to /boot/vmlinuz-4.4.0-10-generic
kdump initrd:
   /var/lib/kdump/initrd.img: symbolic link to /var/lib/kdump/initrd.img
      -4.4.0-10-generic
SSH:                ubuntu@kdump-netcrash
SSH_KEY:            /root/.ssh/kdump_id_rsa
HOSTTAG:            ip
current state:      ready to kdump
```

Remote Kernel Crash Dumps using the NFS protocol

To enable remote dumps using the *NFS* protocol, the /etc/default/kdump−tools must be modified in the following manner :

```
# NFS −       Hostname and mount point of the NFS server configured to receive
#             the crash dump. The syntax must be {HOSTNAME}:{MOUNTPOINT}
#             (e.g. remote:/var/crash)
#
NFS="kdump−netcrash:/var/crash"
```

As with the SSH protocol, the HOSTTAG variable can be used to replace the IP address by the hostname as the prefix of the remote directory.

The kdump−config show command can be used to confirm that kdump is correctly configured to use the NFS protocol :

```
kdump−config show
DUMP_MODE:          kdump
USE_KDUMP:          1
KDUMP_SYSCTL:       kernel.panic_on_oops=1
KDUMP_COREDIR:      /var/crash
crashkernel addr: 0x2c000000
   /var/lib/kdump/vmlinuz: symbolic link to /boot/vmlinuz−4.4.0−10−generic
kdump initrd:
   /var/lib/kdump/initrd.img: symbolic link to /var/lib/kdump/initrd.img
      −4.4.0−10−generic
NFS:                kdump−netcrash:/var/crash
HOSTTAG:            hostname
current state:      ready to kdump
```

Verification

To confirm that the kernel dump mechanism is enabled, there are a few things to verify. First, confirm that the *crashkernel* boot parameter is present (note: The following line has been split into two to fit the format of this document:

```
cat /proc/cmdline

BOOT_IMAGE=/vmlinuz−3.2.0−17−server root=/dev/mapper/PreciseS−root ro
 crashkernel=384M−2G:64M,2G−:128M
```

The *crashkernel* parameter has the following syntax:

```
crashkernel=<range1>:<size1>[,<range2>:<size2>,...][@offset]
    range=start−[end] 'start' is inclusive and 'end' is exclusive.
```

So for the crashkernel parameter found in /proc/cmdline we would have :

```
crashkernel=384M−2G:64M,2G−:128M
```

The above value means:

- if the RAM is smaller than 384M, then don't reserve anything (this is the "rescue" case)
- if the RAM size is between 386M and 2G (exclusive), then reserve 64M

- if the RAM size is larger than 2G, then reserve 128M

Second, verify that the kernel has reserved the requested memory area for the kdump kernel by doing:

```
dmesg | grep -i crash

...
[    0.000000] Reserving 64MB of memory at 800MB for crashkernel (System RAM:
    1023MB)
```

Finally, as seen previously, the kdump-config show command displays the current status of the kdump-tools configuration :

```
        kdump-config show
DUMP_MODE:          kdump
USE_KDUMP:          1
KDUMP_SYSCTL:       kernel.panic_on_oops=1
KDUMP_COREDIR:      /var/crash
crashkernel addr: 0x2c000000
   /var/lib/kdump/vmlinuz: symbolic link to /boot/vmlinuz-4.4.0-10-generic
kdump initrd:
       /var/lib/kdump/initrd.img: symbolic link to /var/lib/kdump/initrd.img
           -4.4.0-10-generic
current state:      ready to kdump

kexec command:
       /sbin/kexec -p --command-line="BOOT_IMAGE=/vmlinuz-4.4.0-10-generic root
           =/dev/mapper/VividS--vg-root ro debug break=init console=ttyS0
           ,115200 irqpoll maxcpus=1 nousb systemd.unit=kdump-tools.service" --
           initrd=/var/lib/kdump/initrd.img /var/lib/kdump/vmlinuz
```

Testing the Crash Dump Mechanism

> **Warning**
>
> Testing the Crash Dump Mechanism will cause *a system reboot.* In certain situations, this can cause data loss if the system is under heavy load. If you want to test the mechanism, make sure that the system is idle or under very light load.

Verify that the *SysRQ* mechanism is enabled by looking at the value of the /proc/sys/kernel/sysrq kernel parameter :

```
cat /proc/sys/kernel/sysrq
```

If a value of *0* is returned the dump and then reboot feature is disabled. A value greater than *1* indicates that a sub-set of sysrq features is enabled. See /etc/sysctl.d/10-magic-sysrq.conf for a detailed description of the options and the default value. Enable dump then reboot testing with the following command :

```
sudo sysctl -w kernel.sysrq=1
```

Once this is done, you must become root, as just using sudo will not be sufficient. As the *root* user, you will have to issue the command echo c > /proc/sysrq-trigger. If you are using a network connection, you will lose contact with the system. This is why it is better to do the test while being connected to the system console. This has the advantage of making the kernel dump process visible.

A typical test output should look like the following :

```
sudo -s
[sudo] password for ubuntu:
# echo c > /proc/sysrq-trigger
[   31.659002] SysRq : Trigger a crash
[   31.659749] BUG: unable to handle kernel NULL pointer dereference at
           (null)
[   31.662668] IP: [<ffffffff8139f166>] sysrq_handle_crash+0x16/0x20
[   31.662668] PGD 3bfb9067 PUD 368a7067 PMD 0
[   31.662668] Oops: 0002 [#1] SMP
[   31.662668] CPU 1
....
```

The rest of the output is truncated, but you should see the system rebooting and somewhere in the log, you will see the following line :

```
Begin: Saving vmcore from kernel crash ...
```

Once completed, the system will reboot to its normal operational mode. You will then find the Kernel Crash Dump file, and related subdirectories, in the /var/crash directory :

```
ls /var/crash
201809240744  kexec_cmd  linux-image-4.15.0-34-generic-201809240744.crash
```

If the dump does not work due to OOM (Out Of Memory) error, then try increasing the amount of reserved memory by editing /etc/default/grub.d/kdump-tools.cfg. For example, to reserve 512 megabytes :

```
GRUB_CMDLINE_LINUX_DEFAULT="$GRUB_CMDLINE_LINUX_DEFAULT crashkernel=384M-:512M
    "
```

run sudo update-grub and then reboot afterwards, and then test again.

Resources

Kernel Crash Dump is a vast topic that requires good knowledge of the linux kernel. You can find more information on the topic here :

- Kdump kernel documentation.
- The crash tool
- Analyzing Linux Kernel Crash (Based on Fedora, it still gives a good walkthrough of kernel dump analysis)

Appendix

Reporting Bugs in Ubuntu Server

The Ubuntu Project, and thus Ubuntu Server, uses Launchpad as its bug tracker. In order to file a bug, you will need a Launchpad account. Create one here if necessary.

Reporting Bugs With apport-cli

The preferred way to report a bug is with the apport-cli command. It must be invoked on the machine affected by the bug because it collects information from the system on which it is being run and publishes it to the bug report on Launchpad. Getting that information to Launchpad can, therefore, be a challenge if the system is not running a desktop environment in order to use a browser (common with servers) or if it does not have Internet access. The steps to take in these situations are described below.

> **Note**
>
> The commands apport-cli and ubuntu-bug should give the same results on a CLI server. The latter is actually a symlink to apport-bug which is intelligent enough to know whether a desktop environment is in use and will choose apport-cli if not. Since server systems tend to be CLI-only apport-cli was chosen from the outset in this guide.

Bug reports in Ubuntu need to be filed against a specific software package, so the name of the package (source package or program name/path) affected by the bug needs to be supplied to apport-cli:

```
apport-cli PACKAGENAME
```

Once apport-cli has finished gathering information you will be asked what to do with it. For instance, to report a bug in vim:

```
$ apport-cli vim

*** Collecting problem information

The collected information can be sent to the developers to improve the
application. This might take a few minutes.
...

*** Send problem report to the developers?

After the problem report has been sent, please fill out the form in the
automatically opened web browser.

What would you like to do? Your options are:
  S: Send report (2.8 KB)
  V: View report
  K: Keep report file for sending later or copying to somewhere else
  I: Cancel and ignore future crashes of this program version
  C: Cancel
Please choose (S/V/K/I/C):
```

The first three options are described below:

- **Send:** submits the collected information to Launchpad as part of the process of filing a new bug report. You will be given the opportunity to describe the bug in your own words.

  ```
  *** Uploading problem information

  The collected information is being sent to the bug tracking system.
  This might take a few minutes.
  94%

  *** To continue, you must visit the following URL:
  ```

```
https://bugs.launchpad.net/ubuntu/+source/vim/+filebug/09b2495a-e2ab-11
    e3-879b-68b5996a96c8?

You can launch a browser now, or copy this URL into a browser on another
    computer.


Choices:
  1: Launch a browser now
  C: Cancel
Please choose (1/C):  1
```

The browser that will be used when choosing '1' will be the one known on the system as www-browser via the Debian alternatives system. Examples of text-based browsers to install include links, elinks, lynx, and w3m. You can also manually point an existing browser at the given URL.

- **View:** displays the collected information on the screen for review. This can be a lot of information. Press 'Enter' to scroll by a screenful. Press 'q' to quit and return to the choice menu.
- **Keep:** writes the collected information to disk. The resulting file can be later used to file the bug report, typically after transferring it to another Ubuntu system.

```
What would you like to do? Your options are:
  S: Send report (2.8 KB)
  V: View report
  K: Keep report file for sending later or copying to somewhere else
  I: Cancel and ignore future crashes of this program version
  C: Cancel
Please choose (S/V/K/I/C): k
Problem report file: /tmp/apport.vim.1pg92p02.apport
```

To report the bug, get the file onto an Internet-enabled Ubuntu system and apply apport-cli to it. This will cause the menu to appear immediately (the information is already collected). You should then press 's' to send:

```
apport-cli apport.vim.1pg92p02.apport
```

To directly save a report to disk (without menus) you can do:

```
apport-cli vim --save apport.vim.test.apport
```

Report names should end in *.apport*.

> **Note**
>
> If this Internet-enabled system is non-Ubuntu/Debian, apport-cli is not available so the bug will need to be created manually. An apport report is also not to be included as an attachment to a bug either so it is completely useless in this scenario.

Reporting Application Crashes

The software package that provides the apport-cli utility, apport, can be configured to automatically capture the state of a crashed application. This is enabled by default (in /etc/default/apport).

After an application crashes, if enabled, apport will store a crash report under /var/crash:

```
-rw-r----- 1 peter     whoopsie 150K Jul 24 16:17 _usr_lib_x86_64-linux-
    gnu_libmenu-cache2_libexec_menu-cached.1000.crash
```

Use the apport-cli command without arguments to process any pending crash reports. It will offer to report them one by one.

```
apport-cli

*** Send problem report to the developers?

After the problem report has been sent, please fill out the form in the
automatically opened web browser.

What would you like to do? Your options are:
  S: Send report (153.0 KB)
  V: View report
  K: Keep report file for sending later or copying to somewhere else
  I: Cancel and ignore future crashes of this program version
  C: Cancel
Please choose (S/V/K/I/C): s
```

If you send the report, as was done above, the prompt will be returned immediately and the /var/crash directory will then contain 2 extra files:

```
-rw-r----- 1 peter     whoopsie 150K Jul 24 16:17 _usr_lib_x86_64-linux-
    gnu_libmenu-cache2_libexec_menu-cached.1000.crash
-rw-rw-r-- 1 peter     whoopsie     0 Jul 24 16:37 _usr_lib_x86_64-linux-
    gnu_libmenu-cache2_libexec_menu-cached.1000.upload
-rw------- 1 whoopsie whoopsie     0 Jul 24 16:37 _usr_lib_x86_64-linux-
    gnu_libmenu-cache2_libexec_menu-cached.1000.uploaded
```

Sending in a crash report like this will not immediately result in the creation of a new public bug. The report will be made private on Launchpad, meaning that it will be visible to only a limited set of bug triagers. These triagers will then scan the report for possible private data before creating a public bug.

Resources

- See the Reporting Bugs Ubuntu wiki page.
- Also, the Apport page has some useful information. Though some of it pertains to using a GUI.

Upgrading

There are several ways to upgrade from one Ubuntu release to another. This section gives an overview of the recommended upgrade method.

do-release-upgrade

The recommended way to upgrade a Server Edition installation is to use the do-release-upgrade utility. Part of the *update-manager-core* package, it does not have any graphical dependencies and is installed by default.

Debian based systems can also be upgraded by using apt dist-upgrade. However, using do-release-upgrade is recommended because it has the ability to handle system configuration changes sometimes needed between releases.

To upgrade to a newer release, from a terminal prompt enter:

```
do-release-upgrade
```

It is also possible to use do-release-upgrade to upgrade to a development version of Ubuntu. To accomplish this use the *-d* switch:

```
do-release-upgrade -d
```

Warning

Upgrading to a development release is *not* recommended for production environments.

For further stability of an LTS release, there is a slight change in behaviour if you are currently running an LTS version. LTS systems are only automatically considered for an upgrade to the next LTS via do-release-upgrade with the first point release. So for example 18.04 will only upgrade once 20.04.1 is released. If you want to update before, e.g. on a subset of machines to evaluate the LTS upgrade for your setup the same argument as an upgrade to a dev release has to be used via the *-d* switch.

Device Mapper Multipathing - Introduction

Device Mapper Multipath will be referred here as **multipath** only.

Multipath allows you to configure multiple I/O paths between server nodes and storage arrays into a single device. These I/O paths are physical SAN connections that can include separate cables, switches, and controllers.

Multipathing aggregates the I/O paths, creating a new device that consists of the aggregated paths. This chapter provides an **introduction and a high-level overview of multipath**.

Overview

Multipath can be used to provide:

- *Redundancy* multipath can provide failover in an active/passive configuration. In an active/passive configuration, only half the paths are used at any time for I/O. If any element of an I/O path (the cable, switch, or controller) fails, multipath switches to an alternate path.
- *Improved Performance* Multipath can be configured in active/active mode, where I/O is spread over the paths in a round-robin fashion. In some configurations, multipath can detect loading on the I/O paths and dynamically re-balance the load.

Storage Array Overview

It is a very good idea to consult your **storage vendor** *installation guide* for the recommended multipath configuration variables for your storage model. The default configuration will probably work but will likely need adjustments based on your storage setup.

Multipath Components

| Component | Description | | - | - | | **dm_multipath kernel module** | Reroutes I/O and supports **failover** for paths and path groups. | | **multipath command** | Lists and configures **multipath** devices. Normally started up with /etc/rc.sysinit, it can also be started up by a udev program whenever a block device is added or it can be run by the initramfs file system. | | **multipathd daemon** | Monitors paths; as paths fail and come back, it may initiate path group switches. Provides for interactive

changes to **multipath** devices. This daemon must be restarted for any changes to the /etc/multipath.conf file to take effect. | | **kpartx command** | Creates device mapper devices for the partitions on a device It is necessary to use this command for DOS-based partitions with multipath. The kpartx is provided in its own package, but the **multipath-tools** package depends on it. |

Multipath Setup Overview

multipath includes compiled-in default settings that are suitable for common multipath configurations. Setting up multipath is often a simple procedure. The basic procedure for configuring your system with multipath is as follows:

1. Install the **multipath-tools** and **multipath-tools-boot** packages
2. Create an empty config file called /etc/multipath.conf
3. Edit the **multipath.conf** file to modify default values and save the updated file.
4. Start the multipath daemon
5. Update initial ramdisk

For detailed setup instructions for multipath configuration see DM-Multipath Configuration and DM-Multipath Setup.

Multipath Devices

Without multipath, each path from a server node to a storage controller is treated by the system as a separate device, even when the I/O path connects the same server node to the same storage controller. Multipath provides a way of organizing the I/O paths logically, by creating a single device on top of the underlying paths.

Multipath Device Identifiers

Each multipath device has a World Wide Identifier (WWID), which is guaranteed to be globally unique and unchanging. By default, the name of a multipath device is set to its WWID. Alternately, you can set the **user_friendly_names** option in multipath.conf, which causes multipath to use a **node-unique** alias of the form **mpathn** as the name.

For example, a node with two HBAs attached to a storage controller with two ports via a single unzoned FC switch sees four devices: **/dev/sda**, **/dev/sdb**, **/dev/sdc**, and **/dev/sdd**. Multipath creates a single device with a unique WWID that reroutes I/O to those four underlying devices according to the multipath configuration.

When the user_friendly_names configuration option is set to **yes**, the name of the multipath device is set to **mpathn**. When new devices are brought under the control of multipath, the new devices may be seen in two different places under the **/dev** directory: **/dev/mapper/mpathn** and **/dev/dm-n**.

- The devices in **/dev/mapper** are created early in the boot process. **Use these devices to access the multipathed devices.**
- Any devices of the form **/dev/dm-n** are for **internal use only** and should never be used directly.

You can also set the name of a multipath device to a name of your choosing by using the **alias** option in the **multipaths** section of the multipath configuration file.

> For information on the multipath configuration defaults, including the **user_friendly_names** and **alias** configuration options, see DM-Multipath Configuration.

Consistent Multipath Device Names in a Cluster

When the **user_friendly_names** configuration option is set to yes, the name of the multipath device is unique to a node, but it is not guaranteed to be the same on all nodes using the multipath device. Similarly, if you set the **alias** option for a device in the **multipaths** section of /etc/multipath.conf, the name is not automatically consistent across all nodes in the cluster.

This should not cause any difficulties if you use LVM to create logical devices from the multipath device, but if you require that your multipath device names be consistent in every node it is recommended that you leave the **user_friendly_names** option set to **no** and that you not configure aliases for the devices.

If you configure an alias for a device that you would like to be consistent across the nodes in the cluster, you should ensure that the /etc/multipath.conf file is the same for each node in the cluster by following the same procedure:

1. Configure the aliases for the multipath devices in the in the multipath.conf file on one machine.
2. Disable all of your multipath devices on your other machines by running the following commands:

   ```
   # systemctl stop multipath-tools.service
   # multipath -F
   ```

3. Copy the /etc/multipath.conf file from the first machine to all the other machines in the cluster.
4. Re-enable the multipathd daemon on all the other machines in the cluster by running the following command:

   ```
   # systemctl start multipath-tools.service
   ```

When you add a new device you will need to repeat this process.

Multipath Device attributes

In addition to the **user_friendly_names** and **alias** options, a multipath device has numerous attributes. You can modify these attributes for a specific multipath device by creating an entry for that device in the **multipaths** section of /etc/multipath.conf.

For information on the **multipaths** section of the multipath configuration file, see DM-Multipath Configuration.

Multipath Devices in Logical Volumes

After creating multipath devices, you can use the multipath device names just as you would use a physical device name when creating an LVM physical volume.

For example, if /dev/mapper/mpatha is the name of a multipath device, the following command will mark /dev/mapper/mpatha as a physical volume:

```
# pvcreate /dev/mapper/mpatha
```

You can use the resulting LVM physical device when you create an LVM volume group just as you would use any other LVM physical device.

> **Note**
>
> If you attempt to create an LVM physical volume on a whole device on which you have configured partitions, the pvcreate command will fail.

When you create an LVM logical volume that uses active/passive multipath arrays as the underlying physical devices, you should include filters in the **lvm.conf** to exclude the disks that underlie the multipath devices. This is because if the array automatically changes the active path to the passive path when it receives I/O, multipath will failover and failback whenever LVM scans the passive path if these devices are not filtered.

For active/passive arrays that require a command to make the passive path active, LVM prints a warning message when this occurs. To filter all SCSI devices in the LVM configuration file (lvm.conf), include the following filter in the devices section of the file.

```
filter = [ "r/block/", "r/disk/", "r/sd.*/", "a/.*/" ]
```

After updating /etc/lvm.conf, it's necessary to update the **initrd** so that this file will be copied there, where the filter matters the most, during boot.

Perform:

```
update-initramfs -u -k all
```

> **Note**
>
> Every time either /etc/lvm.conf or /etc/multipath.conf is updated, the initrd should be rebuilt to reflect these changes. This is imperative when blacklists and filters are necessary to maintain a stable storage configuration.

Device Mapper Multipathing - Configuration

> Device Mapper Multipath will be referred here as **multipath** only.
>
> Before moving on with this session it is recommended that you read: 1. Device Mapper Multipathing - Introduction

Multipath is usually able to work out-of-the-box with most common storages. This doesn't mean the default configuration variables should be used in production: they don't treat important parameters your storage might need.

> Consult your **storage manufacturer's install guide** for the Linux Multipath configuration options. It is very common that storage vendors provide the most adequate options for Linux, including minimal kernel and multipath-tools versions required.

Default configuration values for DM-Multipath can be overridden by editing the /etc/multipath.conf file and restarting the multipathd service.

This chapter provides information on parsing and modifying the multipath.conf file and it is split into the following configuration file sections:

- Configuration File Overview
- Configuration File Defaults
- Configuration File Blacklist & Exceptions
- Configuration File Multipath Section
- Configuration File Devices Section

Configuration File Overview

The configuration file contains entries of the form:

```
<section> {
        <attribute> <value>
        ...
        <subsection> {
                <attribute> <value>
                ...
        }
}
```

The following keywords are recognized:

- **defaults** - This section defines default values for attributes which are used whenever no values are given in the appropriate device or multipath sections.
- **blacklist** - This section defines which devices should be excluded from the multipath topology discovery.
- **blacklist_exceptions** - This section defines which devices should be included in the multipath topology discovery, despite being listed in the blacklist section.
- **multipaths** - This section defines the multipath topologies. They are indexed by a World Wide Identifier(WWID).Attributes set in this section take **precedence over all others**.
- **devices** - This section defines the device-specific settings. Devices are identified by vendor, product, and revision.
- **overrides** - This section defines values for attributes that should override the device-specific settings for all devices.

Configuration File Defaults

Currently, the multipath configuration file **ONLY** includes a minor **defaults** section that sets the **user_friendly_names** parameter to **yes**:

```
defaults {
    user_friendly_names yes
}
```

This overwrites the default value of the **user_friendly_names** parameter.

All the multipath attributes that can set in the **defaults** section of the multipath.conf file can be found HERE with an explanation of what they mean. The attributes are:

- verbosity
- polling_interval
- max_polling_interval
- reassign_maps
- multipath_dir
- path_selector
- path_grouping_policy
- uid_attrs
- uid_attribute
- getuid_callout
- prio
- prio_args
- features
- path_checker
- alias_prefix

- failback
- rr_min_io
- rr_min_io_rq
- max_fds
- rr_weight
- no_path_retry
- queue_without_daemon
- checker_timeout
- flush_on_last_del
- user_friendly_names
- fast_io_fail_tmo
- dev_loss_tmo
- bindings_file
- wwids_file
- prkeys_file
- log_checker_err
- reservation_key
- all_tg_pt
- retain_attached_hw_handler
- detect_prio
- detect_checker
- force_sync
- strict_timing
- deferred_remove
- partition_delimiter
- config_dir
- san_path_err_threshold
- san_path_err_forget_rate
- san_path_err_recovery_time
- marginal_path_double_failed_time
- marginal_path_err_sample_time
- marginal_path_err_rate_threshold
- marginal_path_err_recheck_gap_time
- delay_watch_checks
- delay_wait_checks
- marginal_pathgroups
- find_multipaths
- find_multipaths_timeout
- uxsock_timeout
- retrigger_tries
- retrigger_delay
- missing_uev_wait_timeout
- skip_kpartx
- disable_changed_wwids
- remove_retries
- max_sectors_kb
- ghost_delay
- enable_foreign

  Previously the multipath-tools project used to provide a complete configuration file with all the most used options for each of the most used storage devices. Currently you can see all those default options by executing sudo multipath −t. This will dump used configuration file including all the embedded default options.

Configuration File Blacklist & Exceptions

The blacklist section is used to exclude specific devices from the multipath topology. It is most commonly used to exclude local disks, non-multipathed OR non-disk devices.

1. Blacklist by devnode

   The default blacklist consists of the regular expressions "^(ram|zram|raw|loop|fd|md|dm-|sr|scd|st|dcssblk)[0-9]" and "^(td|hd|vd)[a-z]". This causes virtual devices, non-disk devices, and some other device types to be excluded from multipath handling by default.

   ```
   blacklist {
       devnode "^(ram|zram|raw|loop|fd|md|dm-|sr|scd|st|dcssblk)[0-9]"
       devnode "^(td|hd|vd)[a-z]"
       devnode "^cciss!c[0-9]d[0-9]*"
   }
   ```

2. Blacklist by wwid

   Regular expression for the World Wide Identifier of a device to be excluded/included

3. Blacklist by device

   Subsection for the device description. This subsection recognizes the **vendor** and **product** keywords. Both are regular expressions.

   ```
   device {
       vendor "LENOVO"
       product "Universal Xport"
   }
   ```

4. Blacklist by property

   Regular expression for an udev property. All devices that have matching udev properties will be excluded/included. The handling of the property keyword is special, because devices must have at least one whitelisted udev property; otherwise they're treated as blacklisted, and the message "blacklisted, udev property missing" is displayed in the logs.

5. Blacklist by protocol

   The protocol strings that multipath recognizes are scsi:fcp, scsi:spi, scsi:ssa, scsi:sbp, scsi:srp, scsi:iscsi, scsi:sas, scsi:adt, scsi:ata, scsi:unspec, ccw, cciss, nvme, and undef. The protocol that a path is using can be viewed by running multipathd show paths format "%d %P"

6. Blacklist Exceptions

   The blacklist_exceptions section is used to revert the actions of the blacklist section. This allows one to selectively include ("whitelist") devices which would normally be excluded via the blacklist section.

   ```
   blacklist_exceptions {
       property "(SCSI_IDENT_|ID_WWN)"
   }
   ```

   A common usage is to blacklist "everything" using a catch-all regular expression, and create specific blacklist_exceptions entries for those devices that should be handled by multipath-tools.

Configuration File Multipath Section

The multipaths section allows setting attributes of **multipath maps**. The attributes that are set via the **multipaths section** (see list below) take precedence over all other configuration settings, including those from the overrides section.

The only recognized attribute for the multipaths section is the multipath subsection. If there are multiple multipath subsections matching a given WWID, the contents of these sections are merged, and settings from later entries take precedence.

The multipath subsection recognizes the following attributes:

- **wwid** = (Mandatory) World Wide Identifier. Detected multipath maps are matched agains this attribute. Note that, unlike the wwid attribute in the blacklist section, this is not a regular expression or a substring; WWIDs must match exactly inside the multipaths section.
- **alias** = Symbolic name for the multipath map. This takes precedence over a an entry for the same WWID in the bindings_file.

The following attributes are optional; if not set the default values are taken from the overrides, devices, or **defaults section**:

- path_grouping_policy
- path_selector
- prio
- prio_args
- failback
- rr_weight
- no_path_retry
- rr_min_io
- rr_min_io_rq
- flush_on_last_del
- features
- reservation_key
- user_friendly_names
- deferred_remove
- san_path_err_threshold
- san_path_err_forget_rate
- san_path_err_recovery_time
- marginal_path_err_sample_time
- marginal_path_err_rate_threshold
- marginal_path_err_recheck_gap_time
- marginal_path_double_failed_time
- delay_watch_checks
- delay_wait_checks
- skip_kpartx
- max_sectors_kb
- ghost_delay

Example:

```
multipaths {
    multipath {
        wwid                    3600508b4000156d700012000000b0000
        alias                   yellow
        path_grouping_policy    multibus
        path_selector           "round-robin 0"
        failback                manual
        rr_weight               priorities
        no_path_retry           5
    }
    multipath {
        wwid                    1DEC_____321816758474
```

```
        alias                   red
    }
}
```

Configuration File Devices Section

multipath-tools have a built-in device table with reasonable defaults for more than 100 known multipath-capable storage devices.The devices section can be used to override these settings. If there are multiple matches for a given device, the attributes of all matching entries are applied to it. If an attribute is specified in several matching device subsections, later entries take precedence.

The only recognized attribute for the devices section is the device subsection. Devices detected in the system are matched against the device entries using the vendor, product, and revision fields.

The vendor, product, and revision fields that multipath or multipathd detect for devices in a system depend on the device type. For SCSI devices, they correspond to the respective fields of the SCSI INQUIRY page. In general, the command 'multipathd show paths format "%d %s" ' command can be used to see the detected properties for all devices in the system.

The device subsection recognizes the following attributes:

1. **vendor**(Mandatory) Regular expression to match the vendor name.
2. **product**(Mandatory) Regular expression to match the product name.
3. **revision**Regular expression to match the product revision.
4. **product_blacklist**Products with the given vendor matching this string are blacklisted.
5. **alias_prefix**The user_friendly_names prefix to use for this device type, instead of the default "mpath".
6. **hardware_handler**The hardware handler to use for this device type. The following hardware handler are implemented:
    - **1 emc** - (Hardware-dependent) Hardware handler for DGC class arrays as CLARiiON CX/AX and EMC VNX and Unity families.
    - **1 rdac** - (Hardware-dependent) Hardware handler for LSI / Engenio / NetApp RDAC class as NetApp SANtricity E/EF Series, and OEM arrays from IBM DELL SGI STK and SUN.
    - **1 hp_sw** - (Hardware-dependent) Hardware handler for HP/COMPAQ/DEC HSG80 and MSA/HSV arrays with Active/Standby mode exclusively.
    - **1 alua** - (Hardware-dependent) Hardware handler for SCSI-3 ALUA compatible arrays.
    - **1 ana** - (Hardware-dependent) Hardware handler for NVMe ANA compatible arrays.

The following attributes are optional; if not set the default values are taken from the defaults section:

- path_grouping_policy
- uid_attribute
- getuid_callout
- path_selector
- path_checker
- prio
- prio_args
- features
- failback
- rr_weight
- no_path_retry
- rr_min_io
- rr_min_io_rq
- fast_io_fail_tmo
- dev_loss_tmo
- flush_on_last_del

- user_friendly_names
- retain_attached_hw_handler
- detect_prio
- detect_checker
- deferred_remove
- san_path_err_threshold
- san_path_err_forget_rate
- san_path_err_recovery_time
- marginal_path_err_sample_time
- marginal_path_err_rate_threshold
- marginal_path_err_recheck_gap_time
- marginal_path_double_failed_time
- delay_watch_checks
- delay_wait_checks
- skip_kpartx
- max_sectors_kb
- ghost_delay
- all_tg_pt

Example:

```
devices {
    device {
        vendor "3PARdata"
        product "VV"
        path_grouping_policy "group_by_prio"
        hardware_handler "1 alua"
        prio "alua"
        failback "immediate"
        no_path_retry 18
        fast_io_fail_tmo 10
        dev_loss_tmo "infinity"
    }
    device {
        vendor "DEC"
        product "HSG80"
        path_grouping_policy "group_by_prio"
        path_checker "hp_sw"
        hardware_handler "1 hp_sw"
        prio "hp_sw"
        no_path_retry "queue"
    }
}
```

Device Mapper Multipathing - Setup

> Device Mapper Multipath will be referred here as **multipath** only.
>
> Before moving on with this session it is recommended that you read: 1. Device Mapper Multipathing - Introduction 2. Device Mapper Multipathing - Configuration

This section provides step-by-step example procedures for configuring multipath.

It includes the following procedures:

- **Basic setup**
  - Main **defaults** & **devices** attributes.
  - Shows how to **ignore disks** with blacklists
  - Shows how to **rename disks** using WWIDs
- **Configuring active/active paths**

Basic Setup

Before setting up multipath on your system, ensure that your system has been updated and includes the **multipath-tools** package. If boot from SAN is desired, then the **multipath-tools-boot** package is also required.

A very simple **/etc/multipath.conf** file exists, as explained in Device Mapper Multipathing - Configuration session. All the non-declared in multipath.conf attributes are taken from multipath-tools internal database and its internal blacklist.

The internal attributes database can be acquired by doing:

$ sudo multipath -t

Multipath is usually able to work out-of-the-box with most common storages. This **does not** mean the default configuration variables should be **used in production**: they don't treat important parameters your storage might need.

With the **internal attributes**, described above, and the given example bellow, you will likely be able to create your /etc/multipath.conf file by squashing the code blocks bellow. Make sure to read every **defaults** section attribute comments and change it based on your environment needs.

- Example of a **defaults** section:

```
defaults {
    #
    # name    : polling_interval
    # scope   : multipathd
    # desc    : interval between two path checks in seconds. For
    #           properly functioning paths, the interval between checks
    #           will gradually increase to (4 * polling_interval).
    # values  : n > 0
    # default : 5
    #
    polling_interval 10

    #
    # name    : path_selector
    # scope   : multipath & multipathd
    # desc    : the default path selector algorithm to use
    #           these algorithms are offered by the kernel multipath target
    # values  : "round-robin 0"  = Loop through every path in the path group,
    #                              sending the same amount of IO to each.
    #           "queue-length 0" = Send the next bunch of IO down the path
    #                              with the least amount of outstanding IO.
    #           "service-time 0" = Choose the path for the next bunch of IO
    #                              based on the amount of outstanding IO to
    #                              the path and its relative throughput.
    # default : "service-time 0"
```

```
#
path_selector "round-robin 0"

#
# name    : path_grouping_policy
# scope   : multipath & multipathd
# desc    : the default path grouping policy to apply to unspecified
#           multipaths
# values  : failover           = 1 path per priority group
#           multibus           = all valid paths in 1 priority group
#           group_by_serial    = 1 priority group per detected serial
#                                number
#           group_by_prio      = 1 priority group per path priority
#                                value
#           group_by_node_name = 1 priority group per target node name
# default : failover
#
path_grouping_policy multibus

#
# name    : uid_attribute
# scope   : multipath & multipathd
# desc    : the default udev attribute from which the path
#       identifier should be generated.
# default : ID_SERIAL
#
uid_attribute "ID_SERIAL"

#
# name    : getuid_callout
# scope   : multipath & multipathd
# desc    : the default program and args to callout to obtain a unique
#           path identifier. This parameter is deprecated.
#           This parameter is deprecated, superseded by uid_attribute
# default : /lib/udev/scsi_id --whitelisted --device=/dev/%n
#
getuid_callout "/lib/udev/scsi_id --whitelisted --device=/dev/%n"

#
# name    : prio
# scope   : multipath & multipathd
# desc    : the default function to call to obtain a path
#           priority value. The ALUA bits in SPC-3 provide an
#           exploitable prio value for example.
# default : const
#
# prio "alua"

#
# name    : prio_args
# scope   : multipath & multipathd
# desc    : The arguments string passed to the prio function
#           Most prio functions do not need arguments. The
#       datacore prioritizer need one.
```

```
# default : (null)
#
# prio_args "timeout=1000 preferredsds=foo"

#
# name    : features
# scope   : multipath & multipathd
# desc    : The default extra features of multipath devices.
#           Syntax is "num[ feature_0 feature_1 ...]", where 'num' is the
#           number of features in the following (possibly empty) list of
#           features.
# values  : queue_if_no_path = Queue IO if no path is active; consider
#                              using the 'no_path_retry' keyword instead.
#           no_partitions    = Disable automatic partitions generation via
#                              kpartx.
# default : "0"
#
features    "0"
#features   "1 queue_if_no_path"
#features   "1 no_partitions"
#features   "2 queue_if_no_path no_partitions"

#
# name    : path_checker, checker
# scope   : multipath & multipathd
# desc    : the default method used to determine the paths' state
# values  : readsector0|tur|emc_clariion|hp_sw|directio|rdac|cciss_tur
# default : directio
#
path_checker directio

#
# name    : rr_min_io
# scope   : multipath & multipathd
# desc    : the number of IO to route to a path before switching
#           to the next in the same path group for the bio-based
#           multipath implementation. This parameter is used for
#           kernels version up to 2.6.31; newer kernel version
#           use the parameter rr_min_io_rq
# default : 1000
#
rr_min_io 100

#
# name    : rr_min_io_rq
# scope   : multipath & multipathd
# desc    : the number of IO to route to a path before switching
#           to the next in the same path group for the request-based
#           multipath implementation. This parameter is used for
#           kernels versions later than 2.6.31.
# default : 1
#
rr_min_io_rq 1
```

```
#
# name    : flush_on_last_del
# scope   : multipathd
# desc    : If set to "yes", multipathd will disable queueing when the
#           last path to a device has been deleted.
# values  : yes|no
# default : no
#
flush_on_last_del yes

#
# name    : max_fds
# scope   : multipathd
# desc    : Sets the maximum number of open file descriptors for the
#           multipathd process.
# values  : max|n > 0
# default : None
#
max_fds 8192

#
# name    : rr_weight
# scope   : multipath & multipathd
# desc    : if set to priorities the multipath configurator will assign
#           path weights as "path prio * rr_min_io"
# values  : priorities|uniform
# default : uniform
#
rr_weight priorities

#
# name    : failback
# scope   : multipathd
# desc    : tell the daemon to manage path group failback, or not to.
#           0 means immediate failback, values >0 means deffered
#           failback expressed in seconds.
# values  : manual|immediate|n > 0
# default : manual
#
failback immediate

#
# name    : no_path_retry
# scope   : multipath & multipathd
# desc    : tell the number of retries until disable queueing, or
#           "fail" means immediate failure (no queueing),
#           "queue" means never stop queueing
# values  : queue|fail|n (>0)
# default : (null)
#
no_path_retry fail

#
# name    : queue_without_daemon
```

```
# scope   : multipathd
# desc    : If set to "no", multipathd will disable queueing for all
#           devices when it is shut down.
# values  : yes|no
# default : yes
queue_without_daemon no

#
# name    : user_friendly_names
# scope   : multipath & multipathd
# desc    : If set to "yes", using the bindings file
#           /etc/multipath/bindings to assign a persistent and
#           unique alias to the multipath, in the form of mpath<n>.
#           If set to "no" use the WWID as the alias. In either case
#           this be will be overriden by any specific aliases in this
#           file.
# values  : yes|no
# default : no
user_friendly_names yes

#
# name    : mode
# scope   : multipath & multipathd
# desc    : The mode to use for the multipath device nodes, in octal.
# values  : 0000 - 0777
# default : determined by the process
mode 0644

#
# name    : uid
# scope   : multipath & multipathd
# desc    : The user id to use for the multipath device nodes. You
#           may use either the numeric or symbolic uid
# values  : <user_id>
# default : determined by the process
uid 0

#
# name    : gid
# scope   : multipath & multipathd
# desc    : The group id to user for the multipath device nodes. You
#           may use either the numeric or symbolic gid
# values  : <group_id>
# default : determined by the process
gid disk

#
# name    : checker_timeout
# scope   : multipath & multipathd
# desc    : The timeout to use for path checkers and prioritizers
#           that issue scsi commands with an explicit timeout, in
#           seconds.
# values  : n > 0
# default : taken from /sys/block/sd<x>/device/timeout
```

```
checker_timeout 60

#
# name    : fast_io_fail_tmo
# scope   : multipath & multipathd
# desc    : The number of seconds the scsi layer will wait after a
#           problem has been detected on a FC remote port before failing
#           IO to devices on that remote port.
# values  : off | n >= 0 (smaller than dev_loss_tmo)
# default : determined by the OS
fast_io_fail_tmo 5

#
# name    : dev_loss_tmo
# scope   : multipath & multipathd
# desc    : The number of seconds the scsi layer will wait after a
#           problem has been detected on a FC remote port before
#           removing it from the system.
# values  : infinity | n > 0
# default : determined by the OS
dev_loss_tmo 120

#
# name    : bindings_file
# scope   : multipath
# desc    : The location of the bindings file that is used with
#           the user_friendly_names option.
# values  : <full_pathname>
# default : "/var/lib/multipath/bindings"
# bindings_file "/etc/multipath/bindings"

#
# name    : wwids_file
# scope   : multipath
# desc    : The location of the wwids file multipath uses to
#           keep track of the created multipath devices.
# values  : <full_pathname>
# default : "/var/lib/multipath/wwids"
# wwids_file "/etc/multipath/wwids"

#
# name    : reservation_key
# scope   : multipath
# desc    : Service action reservation key used by mpathpersist.
# values  : <key>
# default : (null)
# reservation_key "mpathkey"

#
# name    : force_sync
# scope   : multipathd
# desc    : If set to yes, multipath will run all of the checkers in
#           sync mode, even if the checker has an async mode.
# values  : yes|no
```

```
    # default : no
    force_sync yes

    #
    # name    : config_dir
    # scope   : multipath & multipathd
    # desc    : If not set to an empty string, multipath will search
    #           this directory alphabetically for files ending in ".conf"
    #           and it will read configuration information from these
    #           files, just as if it was in /etc/multipath.conf
    # values  : "" or a fully qualified pathname
    # default : "/etc/multipath/conf.d"

    #
    # name    : delay_watch_checks
    # scope   : multipathd
    # desc    : If set to a value greater than 0, multipathd will watch
    #           paths that have recently become valid for this many
    #           checks.  If they fail again while they are being watched,
    #           when they next become valid, they will not be used until
    #           they have stayed up for delay_wait_checks checks.
    # values  : no|<n> > 0
    # default : no
    delay_watch_checks 12

    #
    # name    : delay_wait_checks
    # scope   : multipathd
    # desc    : If set to a value greater than 0, when a device that has
    #           recently come back online fails again within
    #           delay_watch_checks checks, the next time it comes back
    #           online, it will marked and delayed, and not used until
    #           it has passed delay_wait_checks checks.
    # values  : no|<n> > 0
    # default : no
    delay_wait_checks 12
}
```

- Example of a **multipaths** section. > Note: You can obtain the WWIDs for your LUNs executing: > > $ multipath -ll > > after the service multipath-tools.service has been restarted.

```
multipaths {
    multipath {
        wwid 360000000000000000e00000000030001
        alias yellow
    }
    multipath {
        wwid 360000000000000000e00000000020001
        alias blue
    }
    multipath {
        wwid 360000000000000000e00000000010001
        alias red
    }
    multipath {
```

```
        wwid 360000000000000000e00000000040001
        alias green
    }
    multipath {
        wwid 360000000000000000e00000000050001
        alias purple
    }
}
```

- Small example of a **devices** section:

```
# devices {
#     device {
#         vendor "IBM"
#         product "2107900"
#         path_grouping_policy group_by_serial
#     }
# }
#
```

- Example of a **blacklist** section:

```
# name     : blacklist
# scope    : multipath & multipathd
# desc     : list of device names to discard as not multipath candidates
#
# Devices can be identified by their device node name "devnode",
# their WWID "wwid", or their vender and product strings "device"
# default : fd, hd, md, dm, sr, scd, st, ram, raw, loop, dcssblk
#
# blacklist {
#     wwid 26353900f02796769
#     devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]\*"
#     devnode "^hd[a-z]"
#     devnode "^dcssblk[0-9]\*"
#     device {
#         vendor DEC.\*
#         product MSA[15]00
#     }
# }
```

- Example of a **blacklist exception** section:

```
# name     : blacklist_exceptions
# scope    : multipath & multipathd
# desc     : list of device names to be treated as multipath candidates
#            even if they are on the blacklist.
#
# Note: blacklist exceptions are only valid in the same class.
# It is not possible to blacklist devices using the devnode keyword
# and to exclude some devices of them using the wwid keyword.
# default : -
#
# blacklist_exceptions {
#         devnode "^dasd[c-d]+[0-9]\*"
#         wwid    "IBM.75000000092461.4d00.34"
```

```
#         wwid    "IBM.75000000092461.4d00.35"
#         wwid    "IBM.75000000092461.4d00.36"
# }
```

Device Mapper Multipathing - Usage & Debug

> Device Mapper Multipath will be referred here as **multipath** only.
>
> Before moving on with this session it is recommended that you read: 1. Device Mapper Multipathing - Introduction 2. Device Mapper Multipathing - Configuration 3. Device Mapper Multipathing - Setup

This section provides step-by-step example procedures for configuring multipath.

It includes the following procedures:

- **Resizing Online** Multipath Devices
- **Moving root** File System from a **Single** Path Device to a **Multipath Device**
- The **Multipath Daemon**
- Issues with **queue_if_no_path**
- Multipath **Command Output**
- Multipath **Queries** with multipath Command
- Determining **Device Mapper Entries** with dmsetup Command
- Troubleshooting with the **multipathd interactive console**

Resizing Online Multipath Devices

First find all the paths to the LUN about to be resized:

```
$ sudo multipath -ll
mpathb (360014056eee8ec6e1164fcb959086482) dm-0 LIO-ORG,lun01
size=1.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:1 sde 8:64 active ready running
`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:1 sdf 8:80 active ready running
mpatha (36001405e3c2841430ee4bf3871b1998b) dm-1 LIO-ORG,lun02
size=1.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:2 sdc 8:32 active ready running
`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:2 sdd 8:48 active ready running
```

Now I'll reconfigure **mpathb** (with wwid = 360014056eee8ec6e1164fcb959086482) to have 2GB instead of just 1Gb and check if has changed:

```
$ echo 1 | sudo tee /sys/block/sde/device/rescan
1
$ echo 1 | sudo tee /sys/block/sdf/device/rescan
1
$ sudo multipath -ll
```

```
mpathb (360014056eee8ec6e1164fcb959086482) dm-0 LIO-ORG,lun01
size=1.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:1 sde 8:64 active ready running
`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:1 sdf 8:80 active ready running
mpatha (36001405e3c2841430ee4bf3871b1998b) dm-1 LIO-ORG,lun02
size=1.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:2 sdc 8:32 active ready running
`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:2 sdd 8:48 active ready running
```

Not yet! We still need to re-scan the multipath map:

```
$ sudo multipathd resize map mpathb
ok
```

And **then** we are good:

```
$ sudo multipath -ll
mpathb (360014056eee8ec6e1164fcb959086482) dm-0 LIO-ORG,lun01
size=2.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:1 sde 8:64 active ready running
.`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:1 sdf 8:80 active ready running
mpatha (36001405e3c2841430ee4bf3871b1998b) dm-1 LIO-ORG,lun02
size=1.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:2 sdc 8:32 active ready running
`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:2 sdd 8:48 active ready running
```

Make sure to run resize2fs /dev/mapper/mpathb to resize the filesystem.

Moving root File System from a Single Path Device to a Multipath Device

This is dramatically simplified by the use of UUIDs to identify devices as an intrinsic label. Simply install multipath-tools-boot and reboot. This will rebuild the initial ramdisk and afford multipath the opportunity to build it's paths before the root filesystem is mounted by UUID.

> **Note**
> Whenever multipath.conf is updated, so should the initrd by executing:
> update-initramfs -u -k all
> The reason behind is multipath.conf is copied to the ramdisk and is integral to determining the available devices to map via it's blacklist and devices sections.

The Multipath Daemon

If you find you have trouble implementing a multipath configuration, you should ensure the multipath daemon is running as described in Device Mapper Multipathing - Setup. The **multipathd** daemon must be running in order to use** multipath** devices.

Multipath Command Output

When you create, modify, or list a multipath device, you get a printout of the current device setup. The format is as follows. For each multipath device: ~~~~~~ action_if_any: alias (wwid_if_different_from_alias) dm_device_name_if_known vendor,product size=size features='features' hwhandler='hardware_handler' wp=write_permission_if_known ~~~~~~~~~~~~~~~~~~~~~~~~ For each path group: ~~~~~~ -+- policy='scheduling_policy' prio=prio_if_known status=path_group_status_if_known ~~~~~~~~~~~~~~~~~~~~~~~~~~ For each path: ~~~~~~ − host:channel:id:lun devnode major:minor dm_status_if_known path_status online_status ~~~~~~~~~~~~~~~~~~~~~~~~ For example, the output of a multipath command might appear as follows: ~~~~~~ mpathb (360014056eee8ec6e1164fcb959086482)dm−0 LIO−ORG,lun01 size=2.0G features='0' hwhandler ='1 alua' wp=rw |−+− policy='service−time 0' prio=50 status=active |- 7:0:0:1 sde 8:64 active ready running −+− policy='service−time 0' prio=50 status=enabled- 8:0:0:1 sdf 8:80 active ready running ~~~~~~ If the path is up and ready for I/O, the status of the path is **ready** or **ghost.** If the path is down, the status is **faulty** or **shaky.** The path status is updated periodically by the **multipathd** daemon based on the *polling interval* defined in the /etc/multipath.conf file.

The dm_status is similar to the path status, but from the kernel's point of view. The dm_status has two states: **failed**, which is analogous to **faulty**, and **active**, which covers all other path states. Occasionally, the path state and the dm state of a device will temporary not agree.

The possible values for **online_status** are **running** and **offline**. A status of **offline** means that the SCSI device has been disabled.

Multipath Queries with multipath Command

You can use the -l and -ll options of the multipath command to display the current multipath configuration. The -l option displays multipath topology gathered from information in sysfs and the device mapper. The -ll option displays the information the -l displays in addition to all other available components of the system.

When displaying the multipath configuration, there are three verbosity levels you can specify with the **-v** option of the multipath command. Specifying **-v0** yields **no output**. Specifying**-v1** outputs the **created or updated multipath names only**, which you can then feed to other tools such as kpartx. Specifying **-v2** prints **all detected paths, multipaths, and device maps**.

> **Note** The default **verbosity** level of multipath is **2** and can be globally modified by defining the verbosity attribute in the **defaults** section of multipath.conf.

The following example shows the output of a **multipath -l** command.

```
$ sudo multipath −l
mpathb (360014056eee8ec6e1164fcb959086482) dm−0 LIO−ORG,lun01
size=2.0G features='0' hwhandler='1 alua' wp=rw
|−+− policy='service−time 0' prio=0 status=active
| ‘− 7:0:0:1 sde 8:64 active undef running
‘−+− policy='service−time 0' prio=0 status=enabled
  ‘− 8:0:0:1 sdf 8:80 active undef running
mpatha (36001405e3c2841430ee4bf3871b1998b) dm−1 LIO−ORG,lun02
size=1.0G features='0' hwhandler='1 alua' wp=rw
|−+− policy='service−time 0' prio=0 status=active
| ‘− 7:0:0:2 sdc 8:32 active undef running
‘−+− policy='service−time 0' prio=0 status=enabled
  ‘− 8:0:0:2 sdd 8:48 active undef running
```

Determining Device Mapper Entries with dmsetup Command

You can use the dmsetup command to find out which device mapper entries match the **multipathed** devices. The following command displays all the device mapper devices and their major and minor numbers. The minor numbers determine the name of the dm device. For example, a minor number of 1 corresponds to the multipathd device /dev/dm-1.

```
$ sudo dmsetup ls
mpathb   (253:0)
mpatha   (253:1)

$ ls -lahd /dev/dm*
brw-rw---- 1 root disk 253, 0 Apr 27 14:49 /dev/dm-0
brw-rw---- 1 root disk 253, 1 Apr 27 14:47 /dev/dm-1
```

Troubleshooting with the multipathd interactive console

The **multipathd -k** command is an interactive interface to the **multipathd** daemon. Entering this command brings up an interactive multipath console. After entering this command, you can enter help to get a list of available commands, you can enter a interactive command, or you can enter **CTRL-D** to quit.

The multipathd interactive console can be used to troubleshoot problems you may be having with your system. For example, the following command sequence displays the multipath configuration, including the defaults, before exiting the console. ~~~~~~
$ sudo multipathd -k > show config > CTRL-D ~~~~~~
The following command sequence ensures that multipath has picked up any changes to the multipath.conf, ~~~~~~ $
sudo multipathd -k > reconfigure > CTRL-D ~~~~~~
Use the following command sequence to ensure that the path checker is working properly. ~~~~~~~~~~~~~~~~~~~~
$ sudo multipathd -k > show paths > CTRL-D ~~~~~~
Commands can also be streamed into multipathd using STDIN like so: ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
$ echo 'show config' | sudo multipathd -k ~~~~~~

Networking

Networks consist of two or more devices, such as computer systems, printers, and related equipment which are connected by either physical cabling or wireless links for the purpose of sharing and distributing information among the connected devices.

This section provides general and specific information pertaining to networking, including an overview of network concepts and detailed discussion of popular network protocols.

TCP/IP

The Transmission Control Protocol and Internet Protocol (TCP/IP) is a standard set of protocols developed in the late 1970s by the Defense Advanced Research Projects Agency (DARPA) as a means of communication between different types of computers and computer networks. TCP/IP is the driving force of the Internet, and thus it is the most popular set of network protocols on Earth.

TCP/IP Introduction

The two protocol components of TCP/IP deal with different aspects of computer networking. *Internet Protocol*, the "IP" of TCP/IP is a connectionless protocol which deals only with network packet routing using the *IP Datagram* as the basic unit of networking information. The IP Datagram consists of a header followed by a message. The *Transmission Control Protocol* is the "TCP" of TCP/IP and enables network hosts to establish connections which may be used to exchange data streams. TCP also guarantees that the data between connections is delivered and that it arrives at one network host in the same order as sent from another network host.

TCP/IP Configuration

The TCP/IP protocol configuration consists of several elements which must be set by editing the appropriate configuration files, or deploying solutions such as the Dynamic Host Configuration Protocol (DHCP) server which in turn, can be configured to provide the proper TCP/IP configuration settings to network clients automatically. These configuration values must be set correctly in order to facilitate the proper network operation of your Ubuntu system.

The common configuration elements of TCP/IP and their purposes are as follows:

- **IP address** The IP address is a unique identifying string expressed as four decimal numbers ranging from zero (0) to two-hundred and fifty-five (255), separated by periods, with each of the four numbers representing eight (8) bits of the address for a total length of thirty-two (32) bits for the whole address. This format is called *dotted quad notation*.
- **Netmask** The Subnet Mask (or simply, *netmask*) is a local bit mask, or set of flags which separate the portions of an IP address significant to the network from the bits significant to the *subnetwork*. For example, in a Class C network, the standard netmask is 255.255.255.0 which masks the first three bytes of the IP address and allows the last byte of the IP address to remain available for specifying hosts on the subnetwork.
- **Network Address** The Network Address represents the bytes comprising the network portion of an IP address. For example, the host 12.128.1.2 in a Class A network would use 12.0.0.0 as the network address, where twelve (12) represents the first byte of the IP address, (the network part) and zeroes (0) in all of the remaining three bytes to represent the potential host values. A network host using the private IP address 192.168.1.100 would in turn use a Network Address of 192.168.1.0, which specifies the first three bytes of the Class C 192.168.1 network and a zero (0) for all the possible hosts on the network.
- **Broadcast Address** The Broadcast Address is an IP address which allows network data to be sent simultaneously to all hosts on a given subnetwork rather than specifying a particular host. The standard general broadcast address for IP networks is 255.255.255.255, but this broadcast address cannot be used to send a broadcast message to every host on the Internet because routers block it. A more appropriate broadcast address is set to match a specific subnetwork. For example, on the private Class C IP network, 192.168.1.0, the broadcast address is 192.168.1.255. Broadcast messages are typically produced by network protocols such as the Address Resolution Protocol (ARP) and the Routing Information Protocol (RIP).
- **Gateway Address** A Gateway Address is the IP address through which a particular network, or host on a network, may be reached. If one network host wishes to communicate with another network host, and that host is not located on the same network, then a *gateway* must be used. In many cases, the Gateway Address will be that of a router on the same network, which will in turn pass traffic on to other networks or hosts, such as Internet hosts. The value of the Gateway Address setting must be correct, or your system will not be able to reach any hosts beyond those on the same network.
- **Nameserver Address** Nameserver Addresses represent the IP addresses of Domain Name Service (DNS) systems, which resolve network hostnames into IP addresses. There are three levels of Name-

server Addresses, which may be specified in order of precedence: The *Primary* Nameserver, the *Secondary* Nameserver, and the *Tertiary* Nameserver. In order for your system to be able to resolve network hostnames into their corresponding IP addresses, you must specify valid Nameserver Addresses which you are authorized to use in your system's TCP/IP configuration. In many cases these addresses can and will be provided by your network service provider, but many free and publicly accessible nameservers are available for use, such as the Level3 (Verizon) servers with IP addresses from 4.2.2.1 to 4.2.2.6.

Tip

The IP address, Netmask, Network Address, Broadcast Address, Gateway Address, and Nameserver Addresses are typically specified via the appropriate directives in the file /etc/network/interfaces. For more information, view the system manual page for interfaces, with the following command typed at a terminal prompt:

Access the system manual page for interfaces with the following command:

```
man interfaces
```

IP Routing

IP routing is a means of specifying and discovering paths in a TCP/IP network along which network data may be sent. Routing uses a set of *routing tables* to direct the forwarding of network data packets from their source to the destination, often via many intermediary network nodes known as *routers*. There are two primary forms of IP routing: *Static Routing* and *Dynamic Routing.*

Static routing involves manually adding IP routes to the system's routing table, and this is usually done by manipulating the routing table with the route command. Static routing enjoys many advantages over dynamic routing, such as simplicity of implementation on smaller networks, predictability (the routing table is always computed in advance, and thus the route is precisely the same each time it is used), and low overhead on other routers and network links due to the lack of a dynamic routing protocol. However, static routing does present some disadvantages as well. For example, static routing is limited to small networks and does not scale well. Static routing also fails completely to adapt to network outages and failures along the route due to the fixed nature of the route.

Dynamic routing depends on large networks with multiple possible IP routes from a source to a destination and makes use of special routing protocols, such as the Router Information Protocol (RIP), which handle the automatic adjustments in routing tables that make dynamic routing possible. Dynamic routing has several advantages over static routing, such as superior scalability and the ability to adapt to failures and outages along network routes. Additionally, there is less manual configuration of the routing tables, since routers learn from one another about their existence and available routes. This trait also eliminates the possibility of introducing mistakes in the routing tables via human error. Dynamic routing is not perfect, however, and presents disadvantages such as heightened complexity and additional network overhead from router communications, which does not immediately benefit the end users, but still consumes network bandwidth.

TCP and UDP

TCP is a connection-based protocol, offering error correction and guaranteed delivery of data via what is known as *flow control*. Flow control determines when the flow of a data stream needs to be stopped, and previously sent data packets should to be re-sent due to problems such as *collisions*, for example, thus ensuring complete and accurate delivery of the data. TCP is typically used in the exchange of important information such as database transactions.

The User Datagram Protocol (UDP), on the other hand, is a *connectionless* protocol which seldom deals with the transmission of important data because it lacks flow control or any other method to ensure reliable

delivery of the data. UDP is commonly used in such applications as audio and video streaming, where it is considerably faster than TCP due to the lack of error correction and flow control, and where the loss of a few packets is not generally catastrophic.

ICMP

The Internet Control Messaging Protocol (ICMP) is an extension to the Internet Protocol (IP) as defined in the Request For Comments (RFC) #792 and supports network packets containing control, error, and informational messages. ICMP is used by such network applications as the ping utility, which can determine the availability of a network host or device. Examples of some error messages returned by ICMP which are useful to both network hosts and devices such as routers, include *Destination Unreachable* and *Time Exceeded*.

Daemons

Daemons are special system applications which typically execute continuously in the background and await requests for the functions they provide from other applications. Many daemons are network-centric; that is, a large number of daemons executing in the background on an Ubuntu system may provide network-related functionality. Some examples of such network daemons include the *Hyper Text Transport Protocol Daemon* (httpd), which provides web server functionality; the *Secure SHell Daemon* (sshd), which provides secure remote login shell and file transfer capabilities; and the *Internet Message Access Protocol Daemon* (imapd), which provides E-Mail services.

Resources

- There are man pages for TCP and IP that contain more useful information.
- Also, see the TCP/IP Tutorial and Technical Overview IBM Redbook.
- Another resource is O'Reilly's TCP/IP Network Administration.

Network Configuration

Ubuntu ships with a number of graphical utilities to configure your network devices. This document is geared toward server administrators and will focus on managing your network on the command line.

Ethernet Interfaces

Ethernet interfaces are identified by the system using predictable network interface names. These names can appear as *eno1* or *enp0s25*. However, in some cases an interface may still use the kernel *eth#* style of naming.

Identify Ethernet Interfaces

To quickly identify all available Ethernet interfaces, you can use the ip command as shown below.

```
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    default qlen 1000
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

```
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
   group default qlen 1000
    link/ether 00:16:3e:e2:52:42 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.102.66.200/24 brd 10.102.66.255 scope global dynamic eth0
       valid_lft 3257sec preferred_lft 3257sec
    inet6 fe80::216:3eff:fee2:5242/64 scope link
       valid_lft forever preferred_lft forever
```

Another application that can help identify all network interfaces available to your system is the lshw command. This command provides greater details around the hardware capabilities of specific adapters. In the example below, lshw shows a single Ethernet interface with the logical name of *eth0* along with bus information, driver details and all supported capabilities.

```
sudo lshw -class network
  *-network
       description: Ethernet interface
       product: MT26448 [ConnectX EN 10GigE, PCIe 2.0 5GT/s]
       vendor: Mellanox Technologies
       physical id: 0
       bus info: pci@0004:01:00.0
       logical name: eth4
       version: b0
       serial: e4:1d:2d:67:83:56
       slot: U78CB.001.WZS09KB-P1-C6-T1
       size: 10Gbit/s
       capacity: 10Gbit/s
       width: 64 bits
       clock: 33MHz
       capabilities: pm vpd msix pciexpress bus_master cap_list ethernet
          physical fibre 10000bt-fd
       configuration: autonegotiation=off broadcast=yes driver=mlx4_en
          driverversion=4.0-0 duplex=full firmware=2.9.1326 ip=192.168.1.1
          latency=0 link=yes multicast=yes port=fibre speed=10Gbit/s
       resources: iomemory:24000-23fff irq:481 memory:3fe200000000-3
          fe2000fffff memory:240000000000-240007ffffff
```

Ethernet Interface Logical Names

Interface logical names can also be configured via a netplan configuration. If you would like control which interface receives a particular logical name use the *match* and *set-name* keys. The match key is used to find an adapter based on some criteria like MAC address, driver, etc. Then the set-name key can be used to change the device to the desired logial name.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth_lan0:
      dhcp4: true
      match:
```

```
    macaddress: 00:11:22:33:44:55
set-name: eth_lan0
```

Ethernet Interface Settings

ethtool is a program that displays and changes Ethernet card settings such as auto-negotiation, port speed, duplex mode, and Wake-on-LAN. The following is an example of how to view supported features and configured settings of an Ethernet interface.

```
sudo ethtool eth4
Settings for eth4:
    Supported ports: [ FIBRE ]
    Supported link modes:   10000baseT/Full
    Supported pause frame use: No
    Supports auto-negotiation: No
    Supported FEC modes: Not reported
    Advertised link modes:  10000baseT/Full
    Advertised pause frame use: No
    Advertised auto-negotiation: No
    Advertised FEC modes: Not reported
    Speed: 10000Mb/s
    Duplex: Full
    Port: FIBRE
    PHYAD: 0
    Transceiver: internal
    Auto-negotiation: off
    Supports Wake-on: d
    Wake-on: d
    Current message level: 0x00000014 (20)
                   link ifdown
    Link detected: yes
```

IP Addressing

The following section describes the process of configuring your systems IP address and default gateway needed for communicating on a local area network and the Internet.

Temporary IP Address Assignment

For temporary network configurations, you can use the ip command which is also found on most other GNU/Linux operating systems. The ip command allows you to configure settings which take effect immediately, however they are not persistent and will be lost after a reboot.

To temporarily configure an IP address, you can use the ip command in the following manner. Modify the IP address and subnet mask to match your network requirements.

```
sudo ip addr add 10.102.66.200/24 dev enp0s25
```

The ip can then be used to set the link up or down.

```
ip link set dev enp0s25 up
ip link set dev enp0s25 down
```

To verify the IP address configuration of enp0s25, you can use the ip command in the following manner.

```
ip address show dev enp0s25
10: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    group default qlen 1000
    link/ether 00:16:3e:e2:52:42 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.102.66.200/24 brd 10.102.66.255 scope global dynamic eth0
       valid_lft 2857sec preferred_lft 2857sec
    inet6 fe80::216:3eff:fee2:5242/64 scope link
       valid_lft forever preferred_lft forever6
```

To configure a default gateway, you can use the ip command in the following manner. Modify the default gateway address to match your network requirements.

```
sudo ip route add default via 10.102.66.1
```

To verify your default gateway configuration, you can use the ip command in the following manner.

```
ip route show
default via 10.102.66.1 dev eth0 proto dhcp src 10.102.66.200 metric 100
10.102.66.0/24 dev eth0 proto kernel scope link src 10.102.66.200
10.102.66.1 dev eth0 proto dhcp scope link src 10.102.66.200 metric 100
```

If you require DNS for your temporary network configuration, you can add DNS server IP addresses in the file /etc/resolv.conf. In general, editing /etc/resolv.conf directly is not recommanded, but this is a temporary and non-persistent configuration. The example below shows how to enter two DNS servers to /etc/resolv.conf, which should be changed to servers appropriate for your network. A more lengthy description of the proper persistent way to do DNS client configuration is in a following section.

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

If you no longer need this configuration and wish to purge all IP configuration from an interface, you can use the ip command with the flush option as shown below.

```
ip addr flush eth0
```

> **Note**
>
> Flushing the IP configuration using the ip command does not clear the contents of /etc/resolv.conf. You must remove or modify those entries manually, or re-boot which should also cause /etc/resolv.conf, which is a symlink to /run/systemd/resolve/stub-resolv.conf, to be re-written.

Dynamic IP Address Assignment (DHCP Client)

To configure your server to use DHCP for dynamic address assignment, create a netplan configuration in the file /etc/netplan/99_config.yaml. The example below assumes you are configuring your first Ethernet interface identified as *enp3s0*.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp3s0:
      dhcp4: true
```

The configuration can then be applied using the netplan command.

```
sudo netplan apply
```

Static IP Address Assignment

To configure your system to use static address assignment, create a netplan configuration in the file /etc /netplan/99_config.yaml. The example below assumes you are configuring your first Ethernet interface identified as *eth0*. Change the *addresses*, *gateway4*, and *nameservers* values to meet the requirements of your network.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      addresses:
        - 10.10.10.2/24
      gateway4: 10.10.10.1
      nameservers:
          search: [mydomain, otherdomain]
          addresses: [10.10.10.1, 1.1.1.1]
```

The configuration can then be applied using the netplan command.

```
sudo netplan apply
```

Loopback Interface

The loopback interface is identified by the system as *lo* and has a default IP address of 127.0.0.1. It can be viewed using the ip command.

```
ip address show lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
```

Name Resolution

Name resolution as it relates to IP networking is the process of mapping IP addresses to hostnames, making it easier to identify resources on a network. The following section will explain how to properly configure your system for name resolution using DNS and static hostname records.

DNS Client Configuration

Traditionally, the file /etc/resolv.conf was a static configuration file that rarely needed to be changed or automatically changed via DCHP client hooks. Systemd-resolved handles name server configuration, and it should be interacted with through the systemd−resolve command. Netplan configures systemd-resolved to generate a list of nameservers and domains to put in /etc/resolv.conf, which is a symlink:

```
/etc/resolv.conf -> ../run/systemd/resolve/stub-resolv.conf
```

To configure the resolver, add the IP addresses of the nameservers that are appropriate for your network to the netplan configuration file. You can also add an optional DNS suffix search-lists to match your network domain names. The resulting file might look like the following:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s25:
      addresses:
        - 192.168.0.100/24
      gateway4: 192.168.0.1
      nameservers:
          search: [mydomain, otherdomain]
          addresses: [1.1.1.1, 8.8.8.8, 4.4.4.4]
```

The *search* option can also be used with multiple domain names so that DNS queries will be appended in the order in which they are entered. For example, your network may have multiple sub-domains to search; a parent domain of *example.com*, and two sub-domains, *sales.example.com* and *dev.example.com*.

If you have multiple domains you wish to search, your configuration might look like the following:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s25:
      addresses:
        - 192.168.0.100/24
      gateway4: 192.168.0.1
      nameservers:
          search: [example.com, sales.example.com, dev.example.com]
          addresses: [1.1.1.1, 8.8.8.8, 4.4.4.4]
```

If you try to ping a host with the name of *server1*, your system will automatically query DNS for its Fully Qualified Domain Name (FQDN) in the following order:

1. server1.example.com
2. server1.sales.example.com
3. server1.dev.example.com

If no matches are found, the DNS server will provide a result of *notfound* and the DNS query will fail.

Static Hostnames

Static hostnames are locally defined hostname-to-IP mappings located in the file /etc/hosts. Entries in the hosts file will have precedence over DNS by default. This means that if your system tries to resolve a hostname and it matches an entry in /etc/hosts, it will not attempt to look up the record in DNS. In some configurations, especially when Internet access is not required, servers that communicate with a limited number of resources can be conveniently set to use static hostnames instead of DNS.

The following is an example of a hosts file where a number of local servers have been identified by simple hostnames, aliases and their equivalent Fully Qualified Domain Names (FQDN's).

```
127.0.0.1    localhost
127.0.1.1    ubuntu-server
10.0.0.11    server1 server1.example.com vpn
10.0.0.12    server2 server2.example.com mail
10.0.0.13    server3 server3.example.com www
10.0.0.14    server4 server4.example.com file
```

Note

In the above example, notice that each of the servers have been given aliases in addition to their proper names and FQDN's. *Server1* has been mapped to the name *vpn*, *server2* is referred to as *mail*, *server3* as *www*, and *server4* as *file*.

Name Service Switch Configuration

The order in which your system selects a method of resolving hostnames to IP addresses is controlled by the Name Service Switch (NSS) configuration file /etc/nsswitch.conf. As mentioned in the previous section, typically static hostnames defined in the systems /etc/hosts file have precedence over names resolved from DNS. The following is an example of the line responsible for this order of hostname lookups in the file /etc/nsswitch.conf.

```
hosts:          files mdns4_minimal [NOTFOUND=return] dns mdns4
```

- **files** first tries to resolve static hostnames located in /etc/hosts.
- **mdns4_minimal** attempts to resolve the name using Multicast DNS.
- **[NOTFOUND=return]** means that any response of *notfound* by the preceding *mdns4_minimal* process should be treated as authoritative and that the system should not try to continue hunting for an answer.
- **dns** represents a legacy unicast DNS query.
- **mdns4** represents a Multicast DNS query.

To modify the order of the above mentioned name resolution methods, you can simply change the *hosts:* string to the value of your choosing. For example, if you prefer to use legacy Unicast DNS versus Multicast DNS, you can change the string in /etc/nsswitch.conf as shown below.

```
hosts:          files dns [NOTFOUND=return] mdns4_minimal mdns4
```

Bridging

Bridging multiple interfaces is a more advanced configuration, but is very useful in multiple scenarios. One scenario is setting up a bridge with multiple network interfaces, then using a firewall to filter traffic between two network segments. Another scenario is using bridge on a system with one interface to allow virtual machines direct access to the outside network. The following example covers the latter scenario.

Configure the bridge by editing your netplan configuration found in /etc/netplan/:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp3s0:
      dhcp4: no
  bridges:
    br0:
      dhcp4: yes
      interfaces:
        - enp3s0
```

Note

Enter the appropriate values for your physical interface and network.

Now apply the configuration to enable the bridge:

```
sudo netplan apply
```

The new bridge interface should now be up and running. The brctl provides useful information about the state of the bridge, controls which interfaces are part of the bridge, etc. See man brctl for more information.

Resources

- The Ubuntu Wiki Network page has links to articles covering more advanced network configuration.
- The netplan website has additional examples and documentation.
- The netplan man page has more information on netplan.
- The systemd-resolve man page has details on systemd-resolve command.
- The systemd-resolved man page has more information on systemd-resolved service.
- For more information on *bridging* see the netplan.io examples page and the Linux Foundation's Networking-Bridge page.

Dynamic Host Configuration Protocol (DHCP)

The Dynamic Host Configuration Protocol (DHCP) is a network service that enables host computers to be automatically assigned settings from a server as opposed to manually configuring each network host. Computers configured to be DHCP clients have no control over the settings they receive from the DHCP server, and the configuration is transparent to the computer's user.

The most common settings provided by a DHCP server to DHCP clients include:

- IP address and netmask
- IP address of the default-gateway to use
- IP adresses of the DNS servers to use

However, a DHCP server can also supply configuration properties such as:

- Host Name
- Domain Name
- Time Server
- Print Server

The advantage of using DHCP is that changes to the network, for example a change in the address of the DNS server, need only be changed at the DHCP server, and all network hosts will be reconfigured the next time their DHCP clients poll the DHCP server. As an added advantage, it is also easier to integrate new computers into the network, as there is no need to check for the availability of an IP address. Conflicts in IP address allocation are also reduced.

A DHCP server can provide configuration settings using the following methods:

- Manual allocation (MAC address)
  This method entails using DHCP to identify the unique hardware address of each network card connected to the network and then continually supplying a constant configuration each time the DHCP client makes a request to the DHCP server using that network device. This ensures that a particular address is assigned automatically to that network card, based on it's MAC address.

- Dynamic allocation (address pool)
  In this method, the DHCP server will assign an IP address from a pool of addresses (sometimes also called a range or scope) for a period of time or lease, that is configured on the server or until the client informs the server that it doesn't need the address anymore. This way, the clients will be receiving their configuration properties dynamically and on a "first come, first served" basis. When a DHCP client is no longer on the network for a specified period, the configuration is expired and released back to the address pool for use by other DHCP Clients. This way, an address can be leased or used for a period of time. After this period, the client has to renegociate the lease with the server to maintain use of the address.
- Automatic allocation
  Using this method, the DHCP automatically assigns an IP address permanently to a device, selecting it from a pool of available addresses. Usually DHCP is used to assign a temporary address to a client, but a DHCP server can allow an infinite lease time.

The last two methods can be considered "automatic" because in each case the DHCP server assigns an address with no extra intervention needed. The only difference between them is in how long the IP address is leased, in other words whether a client's address varies over time. The DHCP server Ubuntu makes available is dhcpd (dynamic host configuration protocol daemon), which is easy to install and configure and will be automatically started at system boot.

Installation

At a terminal prompt, enter the following command to install dhcpd:

```
sudo apt install isc-dhcp-server
```

NOTE: dhcpd's messages are being sent to syslog. Look there for diagnostics messages.

Configuration

You will probably need to change the default configuration by editing /etc/dhcp/dhcpd.conf to suit your needs and particular configuration.

Most commonly, what you want to do is assign an IP address randomly. This can be done with settings as follows:

```
# minimal sample /etc/dhcp/dhcpd.conf
default-lease-time 600;
max-lease-time 7200;

subnet 192.168.1.0 netmask 255.255.255.0 {
 range 192.168.1.150 192.168.1.200;
 option routers 192.168.1.254;
 option domain-name-servers 192.168.1.1, 192.168.1.2;
 option domain-name "mydomain.example";
}
```

This will result in the DHCP server giving clients an IP address from the range 192.168.1.150-192.168.1.200. It will lease an IP address for 600 seconds if the client doesn't ask for a specific time frame. Otherwise the maximum (allowed) lease will be 7200 seconds. The server will also "advise" the client to use 192.168.1.254 as the default-gateway and 192.168.1.1 and 192.168.1.2 as its DNS servers.

You also may need to edit /etc/default/isc-dhcp-server to specify the interfaces dhcpd should listen to.

INTERFACESv4="eth4"

After changing the config files you have to restart the dhcpd service:

```
sudo systemctl restart isc-dhcp-server.service
```

References

- The dhcp3-server Ubuntu Wiki page has more information.
- For more /etc/dhcp/dhcpd.conf options see the dhcpd.conf man page.
- ISC dhcp-server

Time Synchronization

NTP is a TCP/IP protocol for synchronizing time over a network. Basically a client requests the current time from a server, and uses it to set its own clock.

Behind this simple description, there is a lot of complexity - there are tiers of NTP servers, with the tier one NTP servers connected to atomic clocks, and tier two and three servers spreading the load of actually handling requests across the Internet. Also the client software is a lot more complex than you might think - it has to factor out communication delays, and adjust the time in a way that does not upset all the other processes that run on the server. But luckily all that complexity is hidden from you!

Ubuntu by default uses *timedatectl / timesyncd* to synchronize time and users can optionally use chrony to serve the Network Time Protocol.

Synchronizing your systems time

Since Ubuntu 16.04 *timedatectl / timesyncd* (which are part of systemd) replace most of *ntpdate / ntp*.

timesyncd is available by default and replaces not only ntpdate, but also the client portion of chrony (or formerly ntpd). So on top of the one-shot action that ntpdate provided on boot and network activation, now timesyncd by default regularly checks and keeps your local time in sync. It also stores time updates locally, so that after reboots monotonically advances if applicable.

If chrony is installed timedatectl steps back to let chrony do the time keeping. That shall ensure that no two time syncing services are fighting. While no more recommended to be used, this still also applies to ntpd being installed to retain any kind of old behavior/config that you had through an upgrade. But it also implies that on an upgrade from a former release ntp/ntpdate might still be installed and therefore renders the new systemd based services disabled.

ntpdate is considered deprecated in favor of timedatectl (or chrony) and thereby no more installed by default. timesyncd will generally do the right thing keeping your time in sync, and chrony will help with more complex cases. But if you had one of a few known special ntpdate use cases, consider the following:

- If you require a one-shot sync use: chronyd -q
- If you require a one-shot time check, without setting the time use: chronyd -Q

Configuring timedatectl and timesyncd

The current status of time and time configuration via timedatectl and timesyncd can be checked with timedatectl status.

```
$ timedatectl status
                      Local time: Fr 2018-02-23 08:47:13 UTC
                  Universal time: Fr 2018-02-23 08:47:13 UTC
                        RTC time: Fr 2018-02-23 08:47:13
                       Time zone: Etc/UTC (UTC, +0000)
       System clock synchronized: yes
systemd-timesyncd.service active: yes
                 RTC in local TZ: no
```

If chrony is running it will automatically switch to:

```
[...]
systemd-timesyncd.service active: no
```

Via timedatectl an admin can control the timezone, how the system clock should relate to the hwclock and if permanent synronization should be enabled or not. See man timedatectl for more details.

timesyncd itself is still a normal service, so you can check its status also more in detail via.

```
$ systemctl status systemd-timesyncd
  systemd-timesyncd.service - Network Time Synchronization
   Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; enabled;
       vendor preset: enabled)
   Active: active (running) since Fri 2018-02-23 08:55:46 UTC; 10s ago
     Docs: man:systemd-timesyncd.service(8)
 Main PID: 3744 (systemd-timesyn)
   Status: "Synchronized to time server 91.189.89.198:123 (ntp.ubuntu.com)."
    Tasks: 2 (limit: 4915)
   CGroup: /system.slice/systemd-timesyncd.service
           |-3744 /lib/systemd/systemd-timesyncd

Feb 23 08:55:46 bionic-test systemd[1]: Starting Network Time Synchronization
   ...
Feb 23 08:55:46 bionic-test systemd[1]: Started Network Time Synchronization.
Feb 23 08:55:46 bionic-test systemd-timesyncd[3744]: Synchronized to time
   server 91.189.89.198:123 (ntp.ubuntu.com).
```

The nameserver to fetch time for timedatectl and timesyncd from can be specified in /etc/systemd/timesyncd .conf and additional config files can be stored in /etc/systemd/timesyncd.conf.d/. The entries for NTP= and FallbackNTP= are space separated lists. See man timesyncd.conf for more.

Serve the Network Time Protocol

If in addition to synchronizing your system you also want to serve NTP information you need an NTP server. There are several options with chrony, ntpd and open-ntp. The recommended solution is chrony.

chrony(d)

The NTP daemon chronyd calculates the drift and offset of your system clock and continuously adjusts it, so there are no large corrections that could lead to inconsistent logs for instance. The cost is a little processing power and memory, but for a modern server this is usually negligible.

Installation

To install chrony, from a terminal prompt enter:

```
sudo apt install chrony
```

This will provide two binaries:

- chronyd - the actual daemon to sync and serve via the NTP protocol
- chronyc - command-line interface for chrony daemon

Chronyd Configuration

Edit /etc/chrony/chrony.conf to add/remove server lines. By default these servers are configured:

```
# Use servers from the NTP Pool Project. Approved by Ubuntu Technical Board
# on 2011-02-08 (LP: #104525). See http://www.pool.ntp.org/join.html for
# more information.
pool 0.ubuntu.pool.ntp.org iburst
pool 1.ubuntu.pool.ntp.org iburst
pool 2.ubuntu.pool.ntp.org iburst
pool 3.ubuntu.pool.ntp.org iburst
```

See man chrony.conf for more details on the configuration options. After changing the any of the config file you have to restart chrony:

```
sudo systemctl restart chrony.service
```

Of the pool 2.ubuntu.pool.ntp.org as well as ntp.ubuntu.com also support ipv6 if needed. If one needs to force ipv6 there also is ipv6.ntp.ubuntu.com which is not configured by default.

View status

Use chronyc to see query the status of the chrony daemon. For example to get an overview of the currently available and selected time sources.

```
chronyc sources
```

```
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^+ gamma.rueckgr.at              2   8   377   135  -1048us[-1048us] +/-   29
    ms
^- 2b.ncomputers.org             2   8   377   204  -1141us[-1124us] +/-   50
    ms
^+ www.kashra.com                2   8   377   139  +3483us[+3483us] +/-   18
    ms
^+ stratum2-4.NTP.TechFak.U>     2   8   377   143  -2090us[-2073us] +/-   19
    ms
^- zepto.mcl.gg                  2   7   377     9   -774us[ -774us] +/-   29
    ms
^- mirrorhost.pw                 2   7   377    78   -660us[ -660us] +/-   53
    ms
^- atto.mcl.gg                   2   7   377     8   -823us[ -823us] +/-   50
    ms
```

```
^- static.140.107.46.78.cli>      2   8   377     9  -1503us[-1503us] +/-   45
   ms
^- 4.53.160.75                    2   8   377   137    -11ms[  -11ms] +/-  117
   ms
^- 37.44.185.42                   3   7   377    10  -3274us[-3274us] +/-   70
   ms
^- bagnikita.com                  2   7   377    74  +3131us[+3131us] +/-   71
   ms
^- europa.ellipse.net             2   8   377   204   -790us[ -773us] +/-   97
   ms
^- tethys.hot-chilli.net          2   8   377   141   -797us[ -797us] +/-   59
   ms
^- 66-232-97-8.static.hvvc.>      2   7   377   206  +1669us[+1686us] +/-  133
   ms
^+ 85.199.214.102                 1   8   377   205   +175us[ +192us] +/-   12
   ms
^* 46-243-26-34.tangos.nl         1   8   377   141   -123us[ -106us] +/-   10
   ms
^- pugot.canonical.com            2   8   377    21    -95us[  -95us] +/-   57
   ms
^- alphyn.canonical.com           2   6   377    23  -1569us[-1569us] +/-   79
   ms
^- golem.canonical.com            2   7   377    92  -1018us[-1018us] +/-   31
   ms
^- chilipepper.canonical.com      2   8   377    21  -1106us[-1106us] +/-   27
   ms
```

```
chronyc sourcestats

210 Number of sources = 20
```

Name/IP Address	NP	NR	Span	Frequency	Freq Skew	Offset	Std Dev
gamma.rueckgr.at	25	15	32m	−0.007	0.142	−878us	106us
2b.ncomputers.org	26	16	35m	−0.132	0.283	−1169us	256us
www.kashra.com	25	15	32m	−0.092	0.259	+3426us	195us
stratum2-4.NTP.TechFak.U>	25	14	32m	−0.018	0.130	−2056us	96us
zepto.mcl.gg	13	11	21m	+0.148	0.196	−683us	66us
mirrorhost.pw	6	5	645	+0.117	0.445	−591us	19us
atto.mcl.gg	21	13	25m	−0.069	0.199	−904us	103us
static.140.107.46.78.cli>	25	18	34m	−0.005	0.094	−1526us	78us
4.53.160.75	25	10	32m	+0.412	0.110	−11ms	84us
37.44.185.42	24	12	30m	−0.983	0.173	−3718us	122us
bagnikita.com	17	7	31m	−0.132	0.217	+3527us	139us
europa.ellipse.net	26	15	35m	+0.038	0.553	−473us	424us
tethys.hot-chilli.net	25	11	32m	−0.094	0.110	−864us	88us
66-232-97-8.static.hvvc.>	20	11	35m	−0.116	0.165	+1561us	109us
85.199.214.102	26	11	35m	−0.054	0.390	+129us	343us
46-243-26-34.tangos.nl	25	16	32m	+0.129	0.297	−307us	198us
pugot.canonical.com	25	14	34m	−0.271	0.176	−143us	135us
alphyn.canonical.com	17	11	1100	−0.087	0.360	−1749us	114us
golem.canonical.com	23	12	30m	+0.057	0.370	−988us	229us
chilipepper.canonical.com	25	18	34m	−0.084	0.224	−1116us	169us

Certain chronyc commands are privileged and can not be run via the network without explicitly allowing

them. See section *Command and monitoring access* in man chrony.conf for more details. A local admin can use sudo as usually as this will grant him access to the local admin socket /var/run/chrony/chronyd.sock.

PPS Support

Chrony supports various PPS types natively. It can use kernel PPS API as well as PTP hardware clock. Most general GPS receivers can be leveraged via GPSD. The latter (and potentially more) can be accessed via *SHM* or via a *socket* (recommended). All of the above can be used to augment chrony with additional high quality time sources for better accuracy, jitter, drift, longer-or-short term accuracy (Usually each kind of clock type is good at one of those, but non-perfect at the others). For more details on configuration see some of the external PPS/GPSD resource listed below.

Note: at the release of 20.04 there was a bug which until fixed you might want to add this content to your /etc/apparmor.d/local/usr.sbin.gpsd.

Example configuration for GPSD to feed Chrony

For the setup you need $ sudo apt install gpsd chrony

But you will want to test/debug your setup and especially the GPS reception, therefore also install $ sudo apt install pps−tools gpsd−clients

GPS devices usually will communicate via serial interfaces, yet the most common type these days are USB GPS devices which have a serial converter behind USB. If you want to use this for PPS please be aware that the majority does not signal PPS via USB. Check the GPSD hardware list for details. The examples below were run with a Navisys GR701-W.

When plugging in such a device (or at boot time) dmesg should report serial connection of some sorts, example:

```
[   52.442199] usb 1−1.1: new full−speed USB device number 3 using xhci_hcd
[   52.546639] usb 1−1.1: New USB device found, idVendor=067b, idProduct=2303,
    bcdDevice= 4.00
[   52.546654] usb 1−1.1: New USB device strings: Mfr=1, Product=2,
   SerialNumber=0
[   52.546665] usb 1−1.1: Product: USB−Serial Controller D
[   52.546675] usb 1−1.1: Manufacturer: Prolific Technology Inc.
[   52.602103] usbcore: registered new interface driver usbserial_generic
[   52.602244] usbserial: USB Serial support registered for generic
[   52.609471] usbcore: registered new interface driver pl2303
[   52.609503] usbserial: USB Serial support registered for pl2303
[   52.609564] pl2303 1−1.1:1.0: pl2303 converter detected
[   52.618366] usb 1−1.1: pl2303 converter now attached to ttyUSB0
```

We see above that it appeared as ttyUSB0. To later on have chrony accept being feeded time information by that we have to set it up in /etc/chrony/chrony.conf (Please replace USB0 to whatever applies to your setup):

```
refclock SHM 0 refid GPS precision 1e−1 offset 0.9999 delay 0.2
refclock SOCK /var/run/chrony.ttyUSB0.sock refid PPS
```

Then restart chrony to make the socket available and waiting. sudo systemctl restart chrony

Next one needs to tell gpsd which device to manager, therefore in /etc/default/gpsd we set: DEVICES="/dev/ttyUSB0"

Furthermore the *default* use-case of gpsd is, well for *gps position tracking*. Therefore it will normally not consume any cpu since it is not running the service but waiting on a *socket* for clients. Furthermore the client will then tell gpsd what it requests and gpsd will only do that. For the use case of gpsd as PPS-providing-daemon you want to set the option to:

- immediately start even without a client connected, this can be set in GPSD_OPTIONS of /etc/default/gpsd:
  - GPSD_OPTIONS="−n"
- enable the service itself and not wait for a client to reach the socket in the future:
  - sudo systemctl enable /lib/systemd/system/gpsd.service

Restarting gpsd will now initialize the PPS from GPS and in dmesg you will see

```
pps_ldisc: PPS line discipline registered
pps pps0: new PPS source usbserial0
pps pps0: source "/dev/ttyUSB0" added
```

In case you have multiple PPS the tool ppsfind might be useful to identify which PPS belongs to which GPS. You could check that with:

```
$ sudo ppsfind /dev/ttyUSB0
pps0: name=usbserial0 path=/dev/ttyUSB0
```

To get any further you need your GPS to get a lock. Tools like cgps or gpsmon need to report a 3D Fix for the data being any good. Then you'd want to check to really have PPS data reported on that with ppstest

```
$ cgps
...
 Status:          3D FIX (7 secs) ...
```

Next one might want to ensure that the pps device really submits PPS data, to do so run ppstest:

```
$ sudo ppstest /dev/pps0
trying PPS source "/dev/pps0"
found PPS source "/dev/pps0"
ok, found 1 source(s), now start fetching data...
source 0 - assert 1588140739.099526246, sequence: 69 - clear
    1588140739.999663721, sequence: 70
source 0 - assert 1588140740.099661485, sequence: 70 - clear
    1588140739.999663721, sequence: 70
source 0 - assert 1588140740.099661485, sequence: 70 - clear
    1588140740.999786664, sequence: 71
source 0 - assert 1588140741.099792447, sequence: 71 - clear
    1588140740.999786664, sequence: 71
```

Ok, gpsd is now running, the GPS reception has found a fix and this is fed into chrony. Now lets check that from the point of view of chrony.

Initially (e.g. before gpsd has started or before it has a lock) those will be new and "untrusted" sources marked with an "?". If your devices stay in the "?" state and won't leave it even after quite some time then gpsd seems not to feed any data to chrony and you'd need to debug why.

```
chronyc> sources
210 Number of sources = 10
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
#? GPS                           0   4     0     -     +0ns[   +0ns] +/-    0
    ns
```

```
#? PPS                              0   4     0     -     +0ns[   +0ns] +/-    0
    ns
```

Over time chrony will classify them as good or bad. In the example case the raw GPS had too much deviation but PPS is good.

```
chronyc> sources
210 Number of sources = 10
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================

#x GPS 0 4 177 24 -876ms[ -876ms] +/- 200ms
#- PPS 0 4 177 21 +916us[ +916us] +/- 63us
^- chilipepper.canonical.com 2 6 37 53 +33us[ +33us] +/- 33ms
```

And finally after a while it used the hardware PPS input as it was better:

```
chronyc> sources
210 Number of sources = 10
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================

#x GPS                              0   4   377    20   -884ms[ -884ms] +/-  200
    ms
#* PPS                              0   4   377    18  +6677ns[  +52us] +/-   58
    us
^- alphyn.canonical.com             2   6   377    20  -1303us[-1258us] +/-  114
    ms
```

The PPS might also be ok but used in a combined way with e.g. the selected server. See man chronyc for more details about how all these combinations will look like.

```
chronyc> sources
210 Number of sources = 11
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================

#? GPS                              0   4     0     -     +0ns[   +0ns] +/-    0
    ns
#+ PPS                              0   4   377    22   +154us[ +154us] +/- 8561
    us
^* chilipepper.canonical.com        2   6   377    50   -353us[ -300us] +/-   44
    ms
```

And if you wonder if your shm based GPS data is any good, you can check for that as well. First of all chrony will not only tell you if it classified it as good or bad. Via sourcestats you can also check the details

```
chronyc> sourcestats
210 Number of sources = 10
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
===============================================================================
GPS                        20   9   302     +1.993     11.501   -868ms  1208us
PPS                         6   3    78     +0.324      5.009  +3365ns    41us
golem.canonical.com        15  10   783     +0.859      0.509   -750us   108us
```

You can also track the raw data that gpsd or other ntpd compliant refclocks are sending via shared memory by using ntpshmmon:

```
$ sudo ntpshmmon -o
ntpshmmon: version 3.20
#            Name              Offset        Clock                    Real                     L
     Prc
sample NTP1            0.000223854   1588265805.000223854   1588265805.000000000 0
     -10
sample NTP0            0.125691783   1588265805.125999851   1588265805.000308068 0
     -20
sample NTP1            0.000349341   1588265806.000349341   1588265806.000000000 0
     -10
sample NTP0            0.130326636   1588265806.130634945   1588265806.000308309 0
     -20
sample NTP1            0.000485216   1588265807.000485216   1588265807.000000000 0
     -10
```

References

- Chrony FAQ
- ntp.org: home of the Network Time Protocol project
- pool.ntp.org: project of virtual cluster of timeservers
- Freedesktop.org info on timedatectl
- Freedesktop.org info on systemd-timesyncd service
- Feeding chrony from GPSD
- See the Ubuntu Time wiki page for more information.

Data Plane Development Kit

The DPDK is a set of libraries and drivers for fast packet processing and runs mostly in Linux userland. It is a set of libraries that provide the so called "Environment Abstraction Layer" (EAL). The EAL hides the details of the environment and provides a standard programming interface. Common use cases are around special solutions for instance network function virtualization and advanced high-throughput network switching. The DPDK uses a run-to-completion model for fast data plane performance and accesses devices via polling to eliminate the latency of interrupt processing at the tradeoff of higher cpu consumption. It was designed to run on any processors. The first supported CPU was Intel x86 and it is now extended to IBM PPC64 and ARM64.

Ubuntu further provides some infrastructure to ease DPDKs usability.

Prerequisites

This package is currently compiled for the lowest possible CPU requirements. Which still requires at least SSE3 and anything activated by -march=corei7 (in gcc) to be supported by the CPU.

The list of upstream DPDK supported network cards can be found at supported NICs. But a lot of those are disabled by default in the upstream Project as they are not yet in a stable state. The subset of network cards that DPDK has enabled in the package as available in Ubuntu 16.04 is:

DPDK has “userspace” drivers for the cards called PMDs. The packages for these follow the pattern of librte−pmd−<type>−<version>. Therefore the example for an intel e1000 in 18.11 would be librte−pmd−e1000−18.11.

The more commonly used, tested and fully supported drivers are installed as dependencies of dpdk. But there are way more in universe that follow the same naming pattern.

Unassigning the default Kernel drivers

Cards have to be unassigned from their kernel driver and instead be assigned to uio_pci_generic of vfio−pci. uio_pci_generic is older and usually getting to work more easily, but also has less features and isolation.

The newer vfio-pci requires that you activate the following kernel parameters to enable iommu.

```
iommu=pt intel_iommu=on
```

Or on AMD

```
amd_iommu=pt
```

On top for vfio-pci you then have to configure and assign the iommu groups accordingly. That is mostly done in Firmware and by HW layout, you can check the group assignment the kernel probed in /sys/kernel/iommu_groups/.

> Note: virtio is special, dpdk can directly work on those devices without vfio_pci/uio_pci_generic. But to avoid issues by kernel and DPDK managing the device you still have to unassign the kernel driver.

Manual configuration and status checks can be done via sysfs or with the tool dpdk_nic_bind

```
dpdk_nic_bind.py --help
```

Usage:

```
dpdk-devbind.py [options] DEVICE1 DEVICE2 ....

where DEVICE1, DEVICE2 etc, are specified via PCI "domain:bus:slot.func"
    syntax
or "bus:slot.func" syntax. For devices bound to Linux kernel drivers, they may
also be referred to by Linux interface name e.g. eth0, eth1, em0, em1, etc.

Options:
--help, --usage:
    Display usage information and quit

-s, --status:
    Print the current status of all known network, crypto, event
    and mempool devices.
    For each device, it displays the PCI domain, bus, slot and function,
    along with a text description of the device. Depending upon whether the
    device is being used by a kernel driver, the igb_uio driver, or no
    driver, other relevant information will be displayed:
    * the Linux interface name e.g. if=eth0
    * the driver being used e.g. drv=igb_uio
    * any suitable drivers not currently using that device
```

```
        e.g. unused=igb_uio
    NOTE: if this flag is passed along with a bind/unbind option, the
    status display will always occur after the other operations have taken
    place.

--status-dev:
    Print the status of given device group. Supported device groups are:
    "net", "crypto", "event", "mempool" and "compress"

-b driver, --bind=driver:
    Select the driver to use or "none" to unbind the device

-u, --unbind:
    Unbind a device (Equivalent to "-b none")

--force:
    By default, network devices which are used by Linux - as indicated by
    having routes in the routing table - cannot be modified. Using the
    --force flag overrides this behavior, allowing active links to be
    forcibly unbound.
    WARNING: This can lead to loss of network connection and should be used
    with caution.

Examples:
---------

To display current device status:
    dpdk-devbind.py --status

To display current network device status:
    dpdk-devbind.py --status-dev net

To bind eth1 from the current driver and move to use igb_uio
    dpdk-devbind.py --bind=igb_uio eth1

To unbind 0000:01:00.0 from using any driver
    dpdk-devbind.py -u 0000:01:00.0

To bind 0000:02:00.0 and 0000:02:00.1 to the ixgbe kernel driver
    dpdk-devbind.py -b ixgbe 02:00.0 02:00.1
```

DPDK Device configuration

The package *dpdk* provides init scripts that ease configuration of device assignment and huge pages. It also makes them persistent across reboots.

The following is an example of the file /etc/dpdk/interfaces configuring two ports of a network card. One with uio_pci_generic and the other one with vfio-pci.

```
# <bus>         Currently only "pci" is supported
# <id>          Device ID on the specified bus
# <driver>      Driver to bind against (vfio-pci or uio_pci_generic)
#
# Be aware that the two DPDK compatible drivers uio_pci_generic and vfio-pci
    are
```

```
# part of linux-image-extra-<VERSION> package.
# This package is not always installed by default - for example in cloud-
    images.
# So please install it in case you run into missing module issues.
#
# <bus> <id>      <driver>
pci 0000:04:00.0 uio_pci_generic
pci 0000:04:00.1 vfio-pci
```

Cards are identified by their PCI-ID. If you are unsure you might use the tool dpdk_nic_bind.py to show the current available devices and the drivers they are assigned to.

```
dpdk_nic_bind.py --status

Network devices using DPDK-compatible driver
============================================
0000:04:00.0 'Ethernet Controller 10-Gigabit X540-AT2' drv=uio_pci_generic
    unused=ixgbe

Network devices using kernel driver
===================================
0000:02:00.0 'NetXtreme BCM5719 Gigabit Ethernet PCIe' if=eth0 drv=tg3 unused=
    uio_pci_generic *Active*
0000:02:00.1 'NetXtreme BCM5719 Gigabit Ethernet PCIe' if=eth1 drv=tg3 unused=
    uio_pci_generic
0000:02:00.2 'NetXtreme BCM5719 Gigabit Ethernet PCIe' if=eth2 drv=tg3 unused=
    uio_pci_generic
0000:02:00.3 'NetXtreme BCM5719 Gigabit Ethernet PCIe' if=eth3 drv=tg3 unused=
    uio_pci_generic
0000:04:00.1 'Ethernet Controller 10-Gigabit X540-AT2' if=eth5 drv=ixgbe
    unused=uio_pci_generic

Other network devices
=====================
<none>
```

DPDK HugePage configuration

DPDK makes heavy use of huge pages to eliminate pressure on the TLB. Therefore hugepages have to be configured in your system.

The *dpdk* package has a config file and scripts that try to ease hugepage configuration for DPDK in the form of /etc/dpdk/dpdk.conf. If you have more consumers of hugepages than just DPDK in your system or very special requirements how your hugepages are going to be set up you likely want to allocate/control them by yourself. If not this can be a great simplification to get DPDK configured for your needs.

Here an example configuring 1024 Hugepages of 2M each and 4 1G pages.

```
NR_2M_PAGES=1024
NR_1G_PAGES=4
```

As shown this supports configuring 2M and the larger 1G hugepages (or a mix of both). It will make sure there are proper hugetlbfs mountpoints for DPDK to find both sizes no matter what your default huge page size is. The config file itself holds more details on certain corner cases and a few hints if you want to allocate hugepages manually via a kernel parameter.

It depends on your needs which size you want - 1G pages are certainly more effective regarding TLB pressure. But there were reports of them fragmenting inside the DPDK memory allocations. Also it can be harder to grab enough free space to set up a certain amount of 1G pages later in the life-cycle of a system.

Compile DPDK Applications

Currently there are not a lot consumers of the DPDK library that are stable and released. OpenVswitch-DPDK being an exception to that (see below) and more are appearing. But in general it might still happen that you might want to compile an app against the library.

You will often find guides that tell you to fetch the DPDK sources, build them to your needs and eventually build your application based on DPDK by setting values RTE_* for the build system. Since Ubuntu provides an already compiled DPDK for you can can skip all that.

DPDK provides a valid pkg-config file to simplify setting the proper variables and options.

```
sudo apt-get install dpdk-dev libdpdk-dev
gcc testdpdkprog.c $(pkg-config --libs --cflags libdpdk) -o testdpdkprog
```

An example of a complex (autoconfigure) user of pkg-config of DPDK including fallbacks to older non pkg-config style can be seen in the OpenVswitch build system.

Depending on what you build it might be a good addition to install all of DPDK build dependencies before the make, which on Ubuntu can be done automatically with.

```
sudo apt-get install build-dep dpdk
```

DPDK in KVM Guests

If you have no access to DPDK supported network cards you can still work with DPDK by using its support for virtio. To do so you have to create guests backed by hugepages (see above).

On top of that there it is required to have at least SSE3. The default CPU model qemu/libvirt uses is only up to SSE2. So you will have to define a model that passed the proper feature flags (or use host-passthrough). An example can be found in following snippet to your virsh xml (or the equivalent virsh interface you use).

```
<cpu mode='host-passthrough'>
```

Also virtio nowadays supports multiqueue which DPDK in turn can exploit for better speed. To modify a normal virtio definition to have multiple queues add the following to your interface definition. This is about enhancing a normal virtio nic to have multiple queues, to later on be consumed e.g. by DPDK in the guest.

```
<driver name="vhost" queues="4"/>
```

Use DPDK

Since DPDK on its own is only (massive) library you most likely might continue to OpenVswitch-DPDK as an example to put it to use.

Resources

- DPDK Documentation
- Release Notes matching the version packages in Ubuntu 16.04

- Linux DPDK User Getting Started
- EAL Command-line Options
- DPDK Api Documentation
- OpenVswitch DPDK installation
- Wikipedias definition of DPDK

OpenVswitch-DPDK

With DPDK being *just* a library it doesn't do a lot on its own, so it depends on emerging projects making use of it. One consumer of the library that already is part of Ubuntu is OpenVswitch with DPDK support in the package openvswitch-switch-dpdk.

Here an example how to install and configure a basic OpenVswitch using DPDK for later use via libvirt/qemu-kvm.

```
sudo apt-get install openvswitch-switch-dpdk
sudo update-alternatives --set ovs-vswitchd /usr/lib/openvswitch-switch-dpdk/
    ovs-vswitchd-dpdk
ovs-vsctl set Open_vSwitch . "other_config:dpdk-init=true"
# run on core 0 only
ovs-vsctl set Open_vSwitch . "other_config:dpdk-lcore-mask=0x1"
# Allocate 2G huge pages (not Numa node aware)
ovs-vsctl set Open_vSwitch . "other_config:dpdk-alloc-mem=2048"
# limit to one whitelisted device
ovs-vsctl set Open_vSwitch . "other_config:dpdk-extra=--pci-whitelist
    =0000:04:00.0"
sudo service openvswitch-switch restart
```

Please remember that you have to assign devices to DPDK compatible drivers see above at Network - DPDK before restarting.

Please note that the section *dpdk-alloc-mem=2048* above is the most basic numa setup for a single socket system. If you have multiple sockets you might want to define how to split your memory among them. More details about these options are outlined in OpenVswitch setup.

Attaching DPDK ports to OpenVswitch

The OpenVswitch you now started supports all port types OpenVswitch usually does, plus DPDK port types. Here an example how to create a bridge and - instead of a normal external port - add an external DPDK port to it. When doing so you can specify the device that will be associated.

```
ovs-vsctl add-br ovsdpdkbr0 -- set bridge ovsdpdkbr0 datapath_type=netdev
ovs-vsctl add-port ovsdpdkbr0 dpdk0 -- set Interface dpdk0 type=dpdk "options
    :dpdk-devargs=${OVSDEV_PCIID}"
```

Further tuning can be applied by setting options:

```
ovs-vsctl set Interface dpdk0 "options:n_rxq=2"
```

OpenVswitch DPDK to KVM Guests

If you are not building some sort of SDN switch or NFV on top of DPDK it is very likely that you want to forward traffic to KVM guests. The good news is, that with the new qemu/libvirt/dpdk/openvswitch versions in Ubuntu this is no more about manually appending commandline string. This chapter covers a basic configuration how to connect a KVM guest to a OpenVswitch-DPDK instance.

The recommended way to get to a KVM guest is using vhost_user_client. This will cause OVS-DPDK to create connect to a socket that qemu created. That way old issues like guest failures on OVS restart are avoided. Here an example how to add such a port to the bridge you created above.

```
ovs-vsctl add-port ovsdpdkbr0 vhost-user-1 -- set Interface vhost-user-1 type=
    dpdkvhostuserclient "options:vhost-server-path=/var/run/vhostuserclient/
    vhost-user-client-1"
```

This will connect to the specified path that has to be created by a guest listening on it.

To let libvirt/kvm consume this socket and create a guest virtio network device for it add a snippet like this to your guest definition as the network definition.

```
<interface type='vhostuser'>
<source type='unix'
path='/var/run/openvswitch/vhost-user-client-1'
mode='server'/>
<model type='virtio'/>
</interface>
```

Tuning Openvswitch-DPDK

DPDK has plenty of options - in combination with Openvswitch-DPDK the two most commonly used are:

```
ovs-vsctl set Open_vSwitch . other_config:n-dpdk-rxqs=2
ovs-vsctl set Open_vSwitch . other_config:pmd-cpu-mask=0x6
```

The first select how many rx queues are to be used for each DPDK interface, while the second controls how many and where to run PMD threads. The example above will utilize two rx queues and run PMD threads on CPU 1 and 2. See the referred links to "EAL Command-line Options" and "OpenVswitch DPDK installation" at the end of this document for more.

As usual with tunings you have to know your system and workload really well - so please verify any tunings with workloads matching your real use case.

Support and Troubleshooting

DPDK is a fast evolving project. In any case of a search for support and further guides it is highly recommended to first check if they apply to the current version.

Check if it is a known issue on:

- DPDK Mailing Lists
- For OpenVswitch-DPDK OpenStack Mailing Lists
- Known issues in DPDK Launchpad Area
- Join the IRC channels #DPDK or #openvswitch on freenode.

Issues are often due to missing small details in the general setup. Later on, these missing details cause problems which can be hard to track down to their root cause. A common case seems to be the "could not open network device dpdk0 (No such device)" issue. This occurs rather late when setting up a port in Open vSwitch with DPDK. But the root cause most of the time is very early in the setup and initialization. Here an example how a proper initialization of a device looks - this can be found in the syslog/journal when starting Open vSwitch with DPDK enabled.

```
ovs-ctl[3560]: EAL: PCI device 0000:04:00.1 on NUMA socket 0
ovs-ctl[3560]: EAL:   probe driver: 8086:1528 rte_ixgbe_pmd
ovs-ctl[3560]: EAL:   PCI memory mapped at 0x7f2140000000
ovs-ctl[3560]: EAL:   PCI memory mapped at 0x7f2140200000
```

If this is missing, either by ignored cards, failed initialization or other reasons, later on there will be no DPDK device to refer to. Unfortunately the logging is spread across syslog/journal and the openvswitch log. To allow some cross checking here an example what can be found in these logs, relative to the entered command.

```
#Note: This log was taken with dpdk 2.2 and openvswitch 2.5 but still looks
    quite similar (a bit extended) these days
Captions:
CMD: that you enter
SYSLOG: (Inlcuding EAL and OVS Messages)
OVS-LOG: (Openvswitch messages)

#PREPARATION
Bind an interface to DPDK UIO drivers, make Hugepages available, enable DPDK
    on OVS

CMD: sudo service openvswitch-switch restart

SYSLOG:
2016-01-22T08:58:31.372Z|00003|daemon_unix(monitor)|INFO|pid 3329 died, killed
     (Terminated), exiting
2016-01-22T08:58:33.377Z|00002|vlog|INFO|opened log file /var/log/openvswitch/
    ovs-vswitchd.log
2016-01-22T08:58:33.381Z|00003|ovs_numa|INFO|Discovered 12 CPU cores on NUMA
    node 0
2016-01-22T08:58:33.381Z|00004|ovs_numa|INFO|Discovered 1 NUMA nodes and 12
    CPU cores
2016-01-22T08:58:33.381Z|00005|reconnect|INFO|unix:/var/run/openvswitch/db.
    sock: connecting...
2016-01-22T08:58:33.383Z|00006|reconnect|INFO|unix:/var/run/openvswitch/db.
    sock: connected
2016-01-22T08:58:33.386Z|00007|bridge|INFO|ovs-vswitchd (Open vSwitch) 2.5.0

OVS-LOG:
systemd[1]: Stopping Open vSwitch...
systemd[1]: Stopped Open vSwitch.
systemd[1]: Stopping Open vSwitch Internal Unit...
ovs-ctl[3541]: * Killing ovs-vswitchd (3329)
ovs-ctl[3541]: * Killing ovsdb-server (3318)
systemd[1]: Stopped Open vSwitch Internal Unit.
systemd[1]: Starting Open vSwitch Internal Unit...
ovs-ctl[3560]: * Starting ovsdb-server
ovs-vsctl: ovs|00001|vsctl|INFO|Called as ovs-vsctl --no-wait -- init -- set
```

```
    Open_vSwitch . db-version=7.12.1
ovs-vsctl: ovs|00001|vsctl|INFO|Called as ovs-vsctl --no-wait set Open_vSwitch
     . ovs-version=2.5.0 "external-ids:system-id=\"e7c5ba80-bb14-45c1-b8eb-628
    f3ad03903\"" "system-type=\"Ubuntu\"" "system-version=\"16.04-xenial\""
ovs-ctl[3560]: * Configuring Open vSwitch system IDs
ovs-ctl[3560]: 2016-01-22T08:58:31Z|00001|dpdk|INFO|No -vhost_sock_dir
    provided - defaulting to /var/run/openvswitch
ovs-vswitchd: ovs|00001|dpdk|INFO|No -vhost_sock_dir provided - defaulting to
    /var/run/openvswitch
ovs-ctl[3560]: EAL: Detected lcore 0 as core 0 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 1 as core 1 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 2 as core 2 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 3 as core 3 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 4 as core 4 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 5 as core 5 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 6 as core 0 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 7 as core 1 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 8 as core 2 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 9 as core 3 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 10 as core 4 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 11 as core 5 on socket 0
ovs-ctl[3560]: EAL: Support maximum 128 logical core(s) by configuration.
ovs-ctl[3560]: EAL: Detected 12 lcore(s)
ovs-ctl[3560]: EAL: VFIO modules not all loaded, skip VFIO support...
ovs-ctl[3560]: EAL: Setting up physically contiguous memory...
ovs-ctl[3560]: EAL: Ask a virtual area of 0x100000000 bytes
ovs-ctl[3560]: EAL: Virtual area found at 0x7f2040000000 (size = 0x100000000)
ovs-ctl[3560]: EAL: Requesting 4 pages of size 1024MB from socket 0
ovs-ctl[3560]: EAL: TSC frequency is ~2397202 KHz
ovs-vswitchd[3592]: EAL: TSC frequency is ~2397202 KHz
ovs-vswitchd[3592]: EAL: Master lcore 0 is ready (tid=fc6cbb00;cpuset=[0])
ovs-vswitchd[3592]: EAL: PCI device 0000:04:00.0 on NUMA socket 0
ovs-vswitchd[3592]: EAL:   probe driver: 8086:1528 rte_ixgbe_pmd
ovs-vswitchd[3592]: EAL:   Not managed by a supported kernel driver, skipped
ovs-vswitchd[3592]: EAL: PCI device 0000:04:00.1 on NUMA socket 0
ovs-vswitchd[3592]: EAL:   probe driver: 8086:1528 rte_ixgbe_pmd
ovs-vswitchd[3592]: EAL:   PCI memory mapped at 0x7f2140000000
ovs-vswitchd[3592]: EAL:   PCI memory mapped at 0x7f2140200000
ovs-ctl[3560]: EAL: Master lcore 0 is ready (tid=fc6cbb00;cpuset=[0])
ovs-ctl[3560]: EAL: PCI device 0000:04:00.0 on NUMA socket 0
ovs-ctl[3560]: EAL:   probe driver: 8086:1528 rte_ixgbe_pmd
ovs-ctl[3560]: EAL:   Not managed by a supported kernel driver, skipped
ovs-ctl[3560]: EAL: PCI device 0000:04:00.1 on NUMA socket 0
ovs-ctl[3560]: EAL:   probe driver: 8086:1528 rte_ixgbe_pmd
ovs-ctl[3560]: EAL:   PCI memory mapped at 0x7f2140000000
ovs-ctl[3560]: EAL:   PCI memory mapped at 0x7f2140200000
ovs-vswitchd[3592]: PMD: eth_ixgbe_dev_init(): MAC: 4, PHY: 3
ovs-vswitchd[3592]: PMD: eth_ixgbe_dev_init(): port 0 vendorID=0x8086 deviceID
    =0x1528
ovs-ctl[3560]: PMD: eth_ixgbe_dev_init(): MAC: 4, PHY: 3
ovs-ctl[3560]: PMD: eth_ixgbe_dev_init(): port 0 vendorID=0x8086 deviceID=0
    x1528
ovs-ctl[3560]: Zone 0: name:<RG_MP_log_history>, phys:0x83fffdec0, len:0x2080,
     virt:0x7f213fffdec0, socket_id:0, flags:0
```

```
ovs-ctl[3560]: Zone 1: name:<MP_log_history>, phys:0x83fd73d40, len:0x28a0c0,
    virt:0x7f213fd73d40, socket_id:0, flags:0
ovs-ctl[3560]: Zone 2: name:<rte_eth_dev_data>, phys:0x83fd43380, len:0x2f700,
     virt:0x7f213fd43380, socket_id:0, flags:0
ovs-ctl[3560]: * Starting ovs-vswitchd
ovs-ctl[3560]: * Enabling remote OVSDB managers
systemd[1]: Started Open vSwitch Internal Unit.
systemd[1]: Starting Open vSwitch...
systemd[1]: Started Open vSwitch.
```

CMD: `sudo ovs-vsctl add-br ovsdpdkbr0 -- set bridge ovsdpdkbr0 datapath_type=netdev`

SYSLOG:

```
2016-01-22T08:58:56.344Z|00008|memory|INFO|37256 kB peak resident set size
    after 24.5 seconds
2016-01-22T08:58:56.346Z|00009|ofproto_dpif|INFO|netdev@ovs-netdev: Datapath
    supports recirculation
2016-01-22T08:58:56.346Z|00010|ofproto_dpif|INFO|netdev@ovs-netdev: MPLS label
     stack length probed as 3
2016-01-22T08:58:56.346Z|00011|ofproto_dpif|INFO|netdev@ovs-netdev: Datapath
    supports unique flow ids
2016-01-22T08:58:56.346Z|00012|ofproto_dpif|INFO|netdev@ovs-netdev: Datapath
    does not support ct_state
2016-01-22T08:58:56.346Z|00013|ofproto_dpif|INFO|netdev@ovs-netdev: Datapath
    does not support ct_zone
2016-01-22T08:58:56.346Z|00014|ofproto_dpif|INFO|netdev@ovs-netdev: Datapath
    does not support ct_mark
2016-01-22T08:58:56.346Z|00015|ofproto_dpif|INFO|netdev@ovs-netdev: Datapath
    does not support ct_label
2016-01-22T08:58:56.360Z|00016|bridge|INFO|bridge ovsdpdkbr0: added interface
    ovsdpdkbr0 on port 65534
2016-01-22T08:58:56.361Z|00017|bridge|INFO|bridge ovsdpdkbr0: using datapath
    ID 00005a4a1ed0a14d
2016-01-22T08:58:56.361Z|00018|connmgr|INFO|ovsdpdkbr0: added service
    controller "punix:/var/run/openvswitch/ovsdpdkbr0.mgmt"
```

OVS-LOG:

```
ovs-vsctl: ovs|00001|vsctl|INFO|Called as ovs-vsctl add-br ovsdpdkbr0 -- set
    bridge ovsdpdkbr0 datapath_type=netdev
systemd-udevd[3607]: Could not generate persistent MAC address for ovs-netdev:
     No such file or directory
kernel: [50165.886554] device ovs-netdev entered promiscuous mode
kernel: [50165.901261] device ovsdpdkbr0 entered promiscuous mode
```

CMD: `sudo ovs-vsctl add-port ovsdpdkbr0 dpdk0 -- set Interface dpdk0 type=dpdk`

SYSLOG:

```
2016-01-22T08:59:06.369Z|00019|memory|INFO|peak resident set size grew 155% in
     last 10.0 seconds, from 37256 kB to 95008 kB
2016-01-22T08:59:06.369Z|00020|memory|INFO|handlers:4 ports:1 revalidators:2
    rules:5
```

```
2016-01-22T08:59:30.989Z|00021|dpdk|INFO|Port 0: 8c:dc:d4:b3:6d:e9
2016-01-22T08:59:31.520Z|00022|dpdk|INFO|Port 0: 8c:dc:d4:b3:6d:e9
2016-01-22T08:59:31.521Z|00023|dpif_netdev|INFO|Created 1 pmd threads on numa
    node 0
2016-01-22T08:59:31.522Z|00001|dpif_netdev(pmd16)|INFO|Core 0 processing port
    'dpdk0'
2016-01-22T08:59:31.522Z|00024|bridge|INFO|bridge ovsdpdkbr0: added interface
    dpdk0 on port 1
2016-01-22T08:59:31.522Z|00025|bridge|INFO|bridge ovsdpdkbr0: using datapath
    ID 00008cdcd4b36de9
2016-01-22T08:59:31.523Z|00002|dpif_netdev(pmd16)|INFO|Core 0 processing port
    'dpdk0'
```

OVS-LOG:

```
ovs-vsctl: ovs|00001|vsctl|INFO|Called as ovs-vsctl add-port ovsdpdkbr0 dpdk0
    -- set Interface dpdk0 type=dpdk
ovs-vswitchd[3595]: PMD: ixgbe_dev_tx_queue_setup(): sw_ring=0x7f211a79ebc0
    hw_ring=0x7f211a7a6c00 dma_addr=0x81a7a6c00
ovs-vswitchd[3595]: PMD: ixgbe_set_tx_function(): Using simple tx code path
ovs-vswitchd[3595]: PMD: ixgbe_set_tx_function(): Vector tx enabled.
ovs-vswitchd[3595]: PMD: ixgbe_dev_rx_queue_setup(): sw_ring=0x7f211a78a6c0
    sw_sc_ring=0x7f211a786580 hw_ring=0x7f211a78e800 dma_addr=0x81a78e800
ovs-vswitchd[3595]: PMD: ixgbe_set_rx_function(): Vector rx enabled, please
    make sure RX burst size no less than 4 (port=0).
ovs-vswitchd[3595]: PMD: ixgbe_dev_tx_queue_setup(): sw_ring=0x7f211a79ebc0
    hw_ring=0x7f211a7a6c00 dma_addr=0x81a7a6c00
...
```

CMD: sudo ovs-vsctl add-port ovsdpdkbr0 vhost-user-1 -- set Interface vhost-user-1 type=dpdkvhostuser

OVS-LOG:

```
2016-01-22T09:00:35.145Z|00026|dpdk|INFO|Socket /var/run/openvswitch/vhost-
    user-1 created for vhost-user port vhost-user-1
2016-01-22T09:00:35.145Z|00003|dpif_netdev(pmd16)|INFO|Core 0 processing port
    'dpdk0'
2016-01-22T09:00:35.145Z|00004|dpif_netdev(pmd16)|INFO|Core 0 processing port
    'vhost-user-1'
2016-01-22T09:00:35.145Z|00027|bridge|INFO|bridge ovsdpdkbr0: added interface
    vhost-user-1 on port 2
```

SYSLOG:

```
ovs-vsctl: ovs|00001|vsctl|INFO|Called as ovs-vsctl add-port ovsdpdkbr0 vhost-
    user-1 -- set Interface vhost-user-1 type=dpdkvhostuser
ovs-vswitchd[3595]: VHOST_CONFIG: socket created, fd:46
ovs-vswitchd[3595]: VHOST_CONFIG: bind to /var/run/openvswitch/vhost-user-1
```

Eventually we can see the poll thread in top

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3595 root      10 -10 4975344 103936   9916 S 100.0  0.3  33:13.56 ovs-
    vswitchd
```

Resources

- DPDK Documentation
- Release Notes matching the version packages in Ubuntu 16.04
- Linux DPDK User Getting Started
- EAL Command-line Options
- DPDK Api Documentation
- OpenVswitch DPDK installation
- Wikipedias definition of DPDK

Security

Security should always be considered when installing, deploying, and using any type of computer system. Although a fresh installation of Ubuntu is relatively safe for immediate use on the Internet, it is important to have a balanced understanding of your system's security posture based on how it will be used after deployment.

This chapter provides an overview of security-related topics as they pertain to Ubuntu DISTRO-REV Server Edition, and outlines simple measures you may use to protect your server and network from any number of potential security threats.

User Management

User management is a critical part of maintaining a secure system. Ineffective user and privilege management often lead many systems into being compromised. Therefore, it is important that you understand how you can protect your server through simple and effective user account management techniques.

Where is root?

Ubuntu developers made a conscientious decision to disable the administrative root account by default in all Ubuntu installations. This does not mean that the root account has been deleted or that it may not be accessed. It merely has been given a password hash which matches no possible value, therefore may not log in directly by itself.

Instead, users are encouraged to make use of a tool by the name of 'sudo' to carry out system administrative duties. Sudo allows an authorized user to temporarily elevate their privileges using their own password instead of having to know the password belonging to the root account. This simple yet effective methodology provides accountability for all user actions, and gives the administrator granular control over which actions a user can perform with said privileges.

- If for some reason you wish to enable the root account, simply give it a password:

  ```
  sudo passwd
  ```

  Sudo will prompt you for your password, and then ask you to supply a new password for root as shown below:

```
[sudo] password for username: (enter your own password)
Enter new UNIX password: (enter a new password for root)
Retype new UNIX password: (repeat new password for root)
passwd: password updated successfully
```

- To disable the root account password, use the following passwd syntax:

  ```
  sudo passwd -l root
  ```

- You should read more on Sudo by reading the man page:

  ```
  man sudo
  ```

By default, the initial user created by the Ubuntu installer is a member of the group sudo which is added to the file /etc/sudoers as an authorized sudo user. If you wish to give any other account full root access through sudo, simply add them to the sudo group.

Adding and Deleting Users

The process for managing local users and groups is straightforward and differs very little from most other GNU/Linux operating systems. Ubuntu and other Debian based distributions encourage the use of the 'adduser' package for account management.

- To add a user account, use the following syntax, and follow the prompts to give the account a password and identifiable characteristics, such as a full name, phone number, etc.

  ```
  sudo adduser username
  ```

- To delete a user account and its primary group, use the following syntax:

  ```
  sudo deluser username
  ```

  Deleting an account does not remove their respective home folder. It is up to you whether or not you wish to delete the folder manually or keep it according to your desired retention policies.

  Remember, any user added later on with the same UID/GID as the previous owner will now have access to this folder if you have not taken the necessary precautions.

  You may want to change these UID/GID values to something more appropriate, such as the root account, and perhaps even relocate the folder to avoid future conflicts:

  ```
  sudo chown -R root:root /home/username/
  sudo mkdir /home/archived_users/
  sudo mv /home/username /home/archived_users/
  ```

- To temporarily lock or unlock a user password, use the following syntax, respectively:

  ```
  sudo passwd -l username
  sudo passwd -u username
  ```

- To add or delete a personalized group, use the following syntax, respectively:

  ```
  sudo addgroup groupname
  sudo delgroup groupname
  ```

- To add a user to a group, use the following syntax:

  ```
  sudo adduser username groupname
  ```

User Profile Security

When a new user is created, the adduser utility creates a brand new home directory named /home/username. The default profile is modeled after the contents found in the directory of /etc/skel, which includes all profile basics.

If your server will be home to multiple users, you should pay close attention to the user home directory permissions to ensure confidentiality. By default, user home directories in Ubuntu are created with world read/execute permissions. This means that all users can browse and access the contents of other users home directories. This may not be suitable for your environment.

- To verify your current user home directory permissions, use the following syntax:

  ```
  ls -ld /home/username
  ```

  The following output shows that the directory /home/username has world-readable permissions:

  ```
  drwxr-xr-x  2 username username  4096 2007-10-02 20:03 username
  ```

- You can remove the world readable-permissions using the following syntax:

  ```
  sudo chmod 0750 /home/username
  ```

  > **Note**
  >
  > Some people tend to use the recursive option (-R) indiscriminately which modifies all child folders and files, but this is not necessary, and may yield other undesirable results. The parent directory alone is sufficient for preventing unauthorized access to anything below the parent.

  A much more efficient approach to the matter would be to modify the adduser global default permissions when creating user home folders. Simply edit the file /etc/adduser.conf and modify the DIR_MODE variable to something appropriate, so that all new home directories will receive the correct permissions.

  ```
  DIR_MODE=0750
  ```

- After correcting the directory permissions using any of the previously mentioned techniques, verify the results using the following syntax:

  ```
  ls -ld /home/username
  ```

  The results below show that world-readable permissions have been removed:

  ```
  drwxr-x---  2 username username  4096 2007-10-02 20:03 username
  ```

Password Policy

A strong password policy is one of the most important aspects of your security posture. Many successful security breaches involve simple brute force and dictionary attacks against weak passwords. If you intend to offer any form of remote access involving your local password system, make sure you adequately address minimum password complexity requirements, maximum password lifetimes, and frequent audits of your authentication systems.

Minimum Password Length

By default, Ubuntu requires a minimum password length of 6 characters, as well as some basic entropy checks. These values are controlled in the file /etc/pam.d/common−password, which is outlined below.

```
password        [success=1 default=ignore]      pam_unix.so obscure sha512
```

If you would like to adjust the minimum length to 8 characters, change the appropriate variable to min=8. The modification is outlined below.

```
password        [success=1 default=ignore]      pam_unix.so obscure sha512
    minlen=8
```

> **Note**
>
> Basic password entropy checks and minimum length rules do not apply to the administrator using sudo level commands to setup a new user.

Password Expiration

When creating user accounts, you should make it a policy to have a minimum and maximum password age forcing users to change their passwords when they expire.

- To easily view the current status of a user account, use the following syntax:

  ```
  sudo chage −l username
  ```

  The output below shows interesting facts about the user account, namely that there are no policies applied:

  ```
  Last password change                                    : Jan 20, 2015
  Password expires                                        : never
  Password inactive                                       : never
  Account expires                                         : never
  Minimum number of days between password change          : 0
  Maximum number of days between password change          : 99999
  Number of days of warning before password expires       : 7
  ```

- To set any of these values, simply use the following syntax, and follow the interactive prompts:

  ```
  sudo chage username
  ```

  The following is also an example of how you can manually change the explicit expiration date (-E) to 01/31/2015, minimum password age (-m) of 5 days, maximum password age (-M) of 90 days, inactivity period (-I) of 30 days after password expiration, and a warning time period (-W) of 14 days before password expiration:

  ```
  sudo chage –E 01/31/2015 –m 5 –M 90 −I 30 –W 14 username
  ```

- To verify changes, use the same syntax as mentioned previously:

  ```
  sudo chage −l username
  ```

  The output below shows the new policies that have been established for the account:

  ```
  Last password change                                    : Jan 20, 2015
  Password expires                                        : Apr 19, 2015
  Password inactive                                       : May 19, 2015
  Account expires                                         : Jan 31, 2015
  ```

```
Minimum number of days between password change          : 5
Maximum number of days between password change          : 90
Number of days of warning before password expires       : 14
```

Other Security Considerations

Many applications use alternate authentication mechanisms that can be easily overlooked by even experienced system administrators. Therefore, it is important to understand and control how users authenticate and gain access to services and applications on your server.

SSH Access by Disabled Users

Simply disabling/locking a user password will not prevent a user from logging into your server remotely if they have previously set up SSH public key authentication. They will still be able to gain shell access to the server, without the need for any password. Remember to check the users home directory for files that will allow for this type of authenticated SSH access, e.g. /home/username/.ssh/authorized_keys.

Remove or rename the directory .ssh/ in the user's home folder to prevent further SSH authentication capabilities.

Be sure to check for any established SSH connections by the disabled user, as it is possible they may have existing inbound or outbound connections. Kill any that are found.

```
who | grep username  (to get the pts/# terminal)
sudo pkill -f pts/#
```

Restrict SSH access to only user accounts that should have it. For example, you may create a group called "sshlogin" and add the group name as the value associated with the AllowGroups variable located in the file /etc/ssh/sshd_config.

```
AllowGroups sshlogin
```

Then add your permitted SSH users to the group "sshlogin", and restart the SSH service.

```
sudo adduser username sshlogin
sudo systemctl restart sshd.service
```

External User Database Authentication

Most enterprise networks require centralized authentication and access controls for all system resources. If you have configured your server to authenticate users against external databases, be sure to disable the user accounts both externally and locally. This way you ensure that local fallback authentication is not possible.

AppArmor

AppArmor is a Linux Security Module implementation of name-based mandatory access controls. AppArmor confines individual programs to a set of listed files and posix 1003.1e draft capabilities.

AppArmor is installed and loaded by default. It uses *profiles* of an application to determine what files and permissions the application requires. Some packages will install their own profiles, and additional profiles can be found in the apparmor-profiles package.

To install the apparmor-profiles package from a terminal prompt:

```
sudo apt install apparmor-profiles
```

AppArmor profiles have two modes of execution:

- Complaining/Learning: profile violations are permitted and logged. Useful for testing and developing new profiles.
- Enforced/Confined: enforces profile policy as well as logging the violation.

Using AppArmor

The optional apparmor-utils package contains command line utilities that you can use to change the AppArmor execution mode, find the status of a profile, create new profiles, etc.

- apparmor_status is used to view the current status of AppArmor profiles.

  ```
  sudo apparmor_status
  ```

- aa-complain places a profile into *complain* mode.

  ```
  sudo aa-complain /path/to/bin
  ```

- aa-enforce places a profile into *enforce* mode.

  ```
  sudo aa-enforce /path/to/bin
  ```

- The /etc/apparmor.d directory is where the AppArmor profiles are located. It can be used to manipulate the *mode* of all profiles.

  Enter the following to place all profiles into complain mode:

  ```
  sudo aa-complain /etc/apparmor.d/*
  ```

  To place all profiles in enforce mode:

  ```
  sudo aa-enforce /etc/apparmor.d/*
  ```

- apparmor_parser is used to load a profile into the kernel. It can also be used to reload a currently loaded profile using the *-r* option after modifying it to have the changes take effect. To reload a profile:

  ```
  sudo apparmor_parser -r /etc/apparmor.d/profile.name
  ```

- systemctl can be used to *reload* all profiles:

  ```
  sudo systemctl reload apparmor.service
  ```

- The /etc/apparmor.d/disable directory can be used along with the apparmor_parser -R option to *disable* a profile.

  ```
  sudo ln -s /etc/apparmor.d/profile.name /etc/apparmor.d/disable/
  sudo apparmor_parser -R /etc/apparmor.d/profile.name
  ```

  To *re-enable* a disabled profile remove the symbolic link to the profile in /etc/apparmor.d/disable/. Then load the profile using the *-a* option.

  ```
  sudo rm /etc/apparmor.d/disable/profile.name
  cat /etc/apparmor.d/profile.name | sudo apparmor_parser -a
  ```

- AppArmor can be disabled, and the kernel module unloaded by entering the following:

```
sudo systemctl stop apparmor.service
sudo systemctl disable apparmor.service
```

- To re-enable AppArmor enter:

  ```
  sudo systemctl enable apparmor.service
  sudo systemctl start apparmor.service
  ```

  Note

  Replace *profile.name* with the name of the profile you want to manipulate. Also, replace /path/to/bin/ with the actual executable file path. For example for the ping command use /bin/ping

Profiles

AppArmor profiles are simple text files located in /etc/apparmor.d/. The files are named after the full path to the executable they profile replacing the "/" with ".". For example /etc/apparmor.d/bin.ping is the AppArmor profile for the /bin/ping command.

There are two main type of rules used in profiles:

- *Path entries:* detail which files an application can access in the file system.
- *Capability entries:* determine what privileges a confined process is allowed to use.

As an example, take a look at /etc/apparmor.d/bin.ping:

```
#include <tunables/global>
/bin/ping flags=(complain) {
  #include <abstractions/base>
  #include <abstractions/consoles>
  #include <abstractions/nameservice>

  capability net_raw,
  capability setuid,
  network inet raw,

  /bin/ping mixr,
  /etc/modules.conf r,
}
```

- *#include <tunables/global>:* include statements from other files. This allows statements pertaining to multiple applications to be placed in a common file.
- */bin/ping flags=(complain):* path to the profiled program, also setting the mode to *complain*.
- *capability net_raw,:* allows the application access to the CAP_NET_RAW Posix.1e capability.
- */bin/ping mixr,:* allows the application read and execute access to the file.

  Note

  After editing a profile file the profile must be reloaded. See above at Using AppArmor for details.

Creating a Profile

- *Design a test plan:* Try to think about how the application should be exercised. The test plan should be divided into small test cases. Each test case should have a small description and list the steps to follow.

Some standard test cases are:

  - Starting the program.
  - Stopping the program.
  - Reloading the program.
  - Testing all the commands supported by the init script.

- *Generate the new profile:* Use aa-genprof to generate a new profile. From a terminal:

  ```
  sudo aa-genprof executable
  ```

  For example:

  ```
  sudo aa-genprof slapd
  ```

- To get your new profile included in the apparmor-profiles package, file a bug in *Launchpad* against the AppArmor package:
  - Include your test plan and test cases.
  - Attach your new profile to the bug.

Updating Profiles

When the program is misbehaving, audit messages are sent to the log files. The program aa-logprof can be used to scan log files for AppArmor audit messages, review them and update the profiles. From a terminal:

```
sudo aa-logprof
```

Further pre-existing Profiles

The packages apport-profiles and apparmor-profiles-extra ship some experimental profiles for AppArmor security policies. Do not expect these profiles to work out-of-the-box, but they can give you a head start when trynig to create a new profile by starting off a base that exists.

These profiles are not considered mature enough to be shipped in enforce mode by default. Therefore they are shipped in complain mode so that users can test them, choose which are desired, and help improve them upstream if needed.

Some even more experimental profiles carried by the package are placed in/usr/share/doc/apparmor-profiles /extras/

Checking and debugging denies

You will see in 'dmesg' and any log that collects kernel messages if you have hit a deny. Right away it is worth to know that this will cover any access that was denied because it was not allowed, but explicit denies will put no message in your logs at all.

Examples might look like:

```
[1521056.552037] audit: type=1400 audit(1571868402.378:24425): apparmor="
    DENIED" operation="open" profile="/usr/sbin/cups-browsed" name="/var/lib/
    libvirt/dnsmasq/" pid=1128 comm="cups-browsed" requested_mask="r"
    denied_mask="r" fsuid=0 ouid=0
```

```
[1482106.651527] audit: type=1400 audit(1571829452.330:24323): apparmor="
    DENIED" operation="sendmsg" profile="snap.lxd.lxc" pid=24115 comm="lxc"
    laddr=10.7.0.69 lport=48796 faddr=10.7.0.231 fport=445 family="inet"
    sock_type="stream" protocol=6 requested_mask="send" denied_mask="send"
```

That follows a generic structure starting with a timestamp, an audit tag and the category apparmor=" DENIED". From the following fields you can derive what was going on and why it was failing.

In the examples above that would be

First example: * operation: open (program tried to open a file) * profile: /usr/sbin/cups−browsed (you'll find /etc/apparmor.d/usr.bin.cups−browsed) * name: /var/lib/ libvirt /dnsmasq (what it wanted to access) * pid/comm: the program that did trigger the access * requested_mask/denied_mask/fsuid/ouid: parameters of that open call

Second example: * operation: sendmsg (program tried send via network) * profile: snap.lxd.lxc (snaps are special, you'll find /var/lib/snapd/apparmor/profiles/snap.lxd.lxc) * pid/comm: the program that did trigger the access * laddr/lport/faddr/fport/family/sock_type/protocol: parameters of that sendmsg call

That way you know in which profile and at what action you have to start if you consider either debugging or adapting the profiles.

Profile customization

Profiles are meant to provide security and thereby can't be all too open. But quite often a very special setup would work with a profile if it wold *just allow this one extra access*. To handle that there are three ways.

- modify the profile itself
  - always works, but has the drawback that profiles are in /etc and considered conffiles. So after modification on a related package update you might get a conffile prompt. Worst case depending on configuration automatic updates might even override it and your custom rule is gone.
- use tunables
  - those provide variables that can be used in templates, for example if you want a custom dir considered as it would be a home directory you could modify /etc/apparmor.d/tunables/home which defines the base path rules use for home directories
  - by design those variables will only influence profiles that use them
- modify a local override
  - to mitigate the drawbacks of above approaches *local includes* got introduced adding the ability to write arbitrary rules that will be used, and not get issues on upgrades that modify the packaged rule.
  - The files can be found in /etc/apparmor.d/local/ and exist for the packages that are known to sometimes need slight tweaks for special setups

References

- See the AppArmor Administration Guide for advanced configuration options.
- For details using AppArmor with other Ubuntu releases see the AppArmor Community Wiki page.
- The OpenSUSE AppArmor page is another introduction to AppArmor.
- (https://wiki.debian.org/AppArmor) is another introduction and basic howto for AppArmor.
- A great place to ask for AppArmor assistance, and get involved with the Ubuntu Server community, is the *#ubuntu-hardened* IRC channel on freenode.

Firewall

Introduction

The Linux kernel includes the *Netfilter* subsystem, which is used to manipulate or decide the fate of network traffic headed into or through your server. All modern Linux firewall solutions use this system for packet filtering.

The kernel's packet filtering system would be of little use to administrators without a userspace interface to manage it. This is the purpose of iptables: When a packet reaches your server, it will be handed off to the Netfilter subsystem for acceptance, manipulation, or rejection based on the rules supplied to it from userspace via iptables. Thus, iptables is all you need to manage your firewall, if you're familiar with it, but many frontends are available to simplify the task.

ufw - Uncomplicated Firewall

The default firewall configuration tool for Ubuntu is ufw. Developed to ease iptables firewall configuration, ufw provides a user-friendly way to create an IPv4 or IPv6 host-based firewall.

ufw by default is initially disabled. From the ufw man page:

"ufw is not intended to provide complete firewall functionality via its command interface, but instead provides an easy way to add or remove simple rules. It is currently mainly used for host-based firewalls."

The following are some examples of how to use ufw:

- First, ufw needs to be enabled. From a terminal prompt enter:

  ```
  sudo ufw enable
  ```

- To open a port (SSH in this example):

  ```
  sudo ufw allow 22
  ```

- Rules can also be added using a *numbered* format:

  ```
  sudo ufw insert 1 allow 80
  ```

- Similarly, to close an opened port:

  ```
  sudo ufw deny 22
  ```

- To remove a rule, use delete followed by the rule:

  ```
  sudo ufw delete deny 22
  ```

- It is also possible to allow access from specific hosts or networks to a port. The following example allows SSH access from host 192.168.0.2 to any IP address on this host:

  ```
  sudo ufw allow proto tcp from 192.168.0.2 to any port 22
  ```

  Replace 192.168.0.2 with 192.168.0.0/24 to allow SSH access from the entire subnet.

- Adding the *–dry-run* option to a *ufw* command will output the resulting rules, but not apply them. For example, the following is what would be applied if opening the HTTP port:

  ```
  sudo ufw —dry-run allow http
  ```

```
*filter
:ufw-user-input - [0:0]
:ufw-user-output - [0:0]
:ufw-user-forward - [0:0]
:ufw-user-limit - [0:0]
:ufw-user-limit-accept - [0:0]
### RULES ###

### tuple ### allow tcp 80 0.0.0.0/0 any 0.0.0.0/0
-A ufw-user-input -p tcp --dport 80 -j ACCEPT

### END RULES ###
-A ufw-user-input -j RETURN
-A ufw-user-output -j RETURN
-A ufw-user-forward -j RETURN
-A ufw-user-limit -m limit --limit 3/minute -j LOG --log-prefix "[UFW
    LIMIT]: "
-A ufw-user-limit -j REJECT
-A ufw-user-limit-accept -j ACCEPT
COMMIT
Rules updated
```

- ufw can be disabled by:

  ```
  sudo ufw disable
  ```

- To see the firewall status, enter:

  ```
  sudo ufw status
  ```

- And for more verbose status information use:

  ```
  sudo ufw status verbose
  ```

- To view the *numbered* format:

  ```
  sudo ufw status numbered
  ```

  Note

  If the port you want to open or close is defined in /etc/services, you can use the port name instead of the number. In the above examples, replace *22* with *ssh*.

This is a quick introduction to using ufw. Please refer to the ufw man page for more information.

ufw Application Integration

Applications that open ports can include an ufw profile, which details the ports needed for the application to function properly. The profiles are kept in /etc/ufw/applications.d, and can be edited if the default ports have been changed.

- To view which applications have installed a profile, enter the following in a terminal:

  ```
  sudo ufw app list
  ```

- Similar to allowing traffic to a port, using an application profile is accomplished by entering:

  ```
  sudo ufw allow Samba
  ```

- An extended syntax is available as well:

  ```
  ufw allow from 192.168.0.0/24 to any app Samba
  ```

  Replace *Samba* and *192.168.0.0/24* with the application profile you are using and the IP range for your network.

  Note

  There is no need to specify the *protocol* for the application, because that information is detailed in the profile. Also, note that the *app* name replaces the *port* number.

- To view details about which ports, protocols, etc., are defined for an application, enter:

  ```
  sudo ufw app info Samba
  ```

Not all applications that require opening a network port come with ufw profiles, but if you have profiled an application and want the file to be included with the package, please file a bug against the package in Launchpad.

```
ubuntu-bug nameofpackage
```

IP Masquerading

The purpose of IP Masquerading is to allow machines with private, non-routable IP addresses on your network to access the Internet through the machine doing the masquerading. Traffic from your private network destined for the Internet must be manipulated for replies to be routable back to the machine that made the request. To do this, the kernel must modify the *source* IP address of each packet so that replies will be routed back to it, rather than to the private IP address that made the request, which is impossible over the Internet. Linux uses *Connection Tracking* (conntrack) to keep track of which connections belong to which machines and reroute each return packet accordingly. Traffic leaving your private network is thus "masqueraded" as having originated from your Ubuntu gateway machine. This process is referred to in Microsoft documentation as Internet Connection Sharing.

ufw Masquerading

IP Masquerading can be achieved using custom ufw rules. This is possible because the current back-end for ufw is iptables-restore with the rules files located in /etc/ufw/*.rules. These files are a great place to add legacy iptables rules used without ufw, and rules that are more network gateway or bridge related.

The rules are split into two different files, rules that should be executed before ufw command line rules, and rules that are executed after ufw command line rules.

- First, packet forwarding needs to be enabled in ufw. Two configuration files will need to be adjusted, in /etc/default/ufw change the *DEFAULT_FORWARD_POLICY* to "ACCEPT":

  ```
  DEFAULT_FORWARD_POLICY="ACCEPT"
  ```

  Then edit /etc/ufw/sysctl.conf and uncomment:

  ```
  net/ipv4/ip_forward=1
  ```

  Similarly, for IPv6 forwarding uncomment:

  ```
  net/ipv6/conf/default/forwarding=1
  ```

- Now add rules to the /etc/ufw/before.rules file. The default rules only configure the *filter* table, and to enable masquerading the *nat* table will need to be configured. Add the following to the top of the file just after the header comments:

```
# nat Table rules
*nat
:POSTROUTING ACCEPT [0:0]

# Forward traffic from eth1 through eth0.
-A POSTROUTING -s 192.168.0.0/24 -o eth0 -j MASQUERADE

# don't delete the 'COMMIT' line or these nat table rules won't be
    processed
COMMIT
```

  The comments are not strictly necessary, but it is considered good practice to document your configuration. Also, when modifying any of the *rules* files in /etc/ufw, make sure these lines are the last line for each table modified:

```
# don't delete the 'COMMIT' line or these rules won't be processed
COMMIT
```

  For each *Table* a corresponding *COMMIT* statement is required. In these examples only the *nat* and *filter* tables are shown, but you can also add rules for the *raw* and *mangle* tables.

  Note

  In the above example replace *eth0*, *eth1*, and *192.168.0.0/24* with the appropriate interfaces and IP range for your network.

- Finally, disable and re-enable ufw to apply the changes:

```
sudo ufw disable && sudo ufw enable
```

IP Masquerading should now be enabled. You can also add any additional FORWARD rules to the /etc/ufw/before.rules. It is recommended that these additional rules be added to the *ufw-before-forward* chain.

iptables Masquerading

iptables can also be used to enable Masquerading.

- Similar to ufw, the first step is to enable IPv4 packet forwarding by editing /etc/sysctl.conf and uncomment the following line:

```
net.ipv4.ip_forward=1
```

  If you wish to enable IPv6 forwarding also uncomment:

```
net.ipv6.conf.default.forwarding=1
```

- Next, execute the sysctl command to enable the new settings in the configuration file:

```
sudo sysctl -p
```

- IP Masquerading can now be accomplished with a single iptables rule, which may differ slightly based on your network configuration:

```
sudo iptables -t nat -A POSTROUTING -s 192.168.0.0/16 -o ppp0 -j
    MASQUERADE
```

The above command assumes that your private address space is 192.168.0.0/16 and that your Internet-facing device is ppp0. The syntax is broken down as follows:

  - -t nat – the rule is to go into the nat table
  - -A POSTROUTING – the rule is to be appended (-A) to the POSTROUTING chain
  - -s 192.168.0.0/16 – the rule applies to traffic originating from the specified address space
  - -o ppp0 – the rule applies to traffic scheduled to be routed through the specified network device
  - -j MASQUERADE – traffic matching this rule is to "jump" (-j) to the MASQUERADE target to be manipulated as described above

- Also, each chain in the filter table (the default table, and where most or all packet filtering occurs) has a default *policy* of ACCEPT, but if you are creating a firewall in addition to a gateway device, you may have set the policies to DROP or REJECT, in which case your masqueraded traffic needs to be allowed through the FORWARD chain for the above rule to work:

```
sudo iptables -A FORWARD -s 192.168.0.0/16 -o ppp0 -j ACCEPT
sudo iptables -A FORWARD -d 192.168.0.0/16 -m state \
--state ESTABLISHED,RELATED -i ppp0 -j ACCEPT
```

  The above commands will allow all connections from your local network to the Internet and all traffic related to those connections to return to the machine that initiated them.

- If you want masquerading to be enabled on reboot, which you probably do, edit /etc/rc.local and add any commands used above. For example add the first command with no filtering:

```
iptables -t nat -A POSTROUTING -s 192.168.0.0/16 -o ppp0 -j MASQUERADE
```

Logs

Firewall logs are essential for recognizing attacks, troubleshooting your firewall rules, and noticing unusual activity on your network. You must include logging rules in your firewall for them to be generated, though, and logging rules must come before any applicable terminating rule (a rule with a target that decides the fate of the packet, such as ACCEPT, DROP, or REJECT).

If you are using ufw, you can turn on logging by entering the following in a terminal:

```
sudo ufw logging on
```

To turn logging off in ufw, simply replace *on* with *off* in the above command.

If using iptables instead of ufw, enter:

```
sudo iptables -A INPUT -m state --state NEW -p tcp --dport 80 \
-j LOG --log-prefix "NEW_HTTP_CONN: "
```

A request on port 80 from the local machine, then, would generate a log in dmesg that looks like this (single line split into 3 to fit this document):

```
[4304885.870000] NEW_HTTP_CONN: IN=lo OUT= MAC
    =00:00:00:00:00:00:00:00:00:00:00:00:08:00
SRC=127.0.0.1 DST=127.0.0.1 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=58288 DF PROTO
    =TCP
SPT=53981 DPT=80 WINDOW=32767 RES=0x00 SYN URGP=0
```

The above log will also appear in /var/log/messages, /var/log/syslog, and /var/log/kern.log. This behavior can be modified by editing /etc/syslog.conf appropriately or by installing and configuring ulogd and using the ULOG target instead of LOG. The ulogd daemon is a userspace server that listens for logging instructions from the kernel specifically for firewalls, and can log to any file you like, or even to a PostgreSQL or MySQL database. Making sense of your firewall logs can be simplified by using a log analyzing tool such as logwatch, fwanalog, fwlogwatch, or lire.

Other Tools

There are many tools available to help you construct a complete firewall without intimate knowledge of iptables. A command-line tool with plain-text configuration files:

- Shorewall is a very powerful solution to help you configure an advanced firewall for any network.

References

- The Ubuntu Firewall wiki page contains information on the development of ufw.
- Also, the ufw manual page contains some very useful information: man ufw.
- See the packet-filtering-HOWTO for more information on using iptables.
- The nat-HOWTO contains further details on masquerading.
- The IPTables HowTo in the Ubuntu wiki is a great resource.

Certificates

One of the most common forms of cryptography today is *public-key* cryptography. Public-key cryptography utilizes a *public key* and a *private key*. The system works by encrypting information using the public key. The information can then only be decrypted using the private key.

A common use for public-key cryptography is encrypting application traffic using a Secure Socket Layer (SSL) or Transport Layer Security (TLS) connection. One example: configuring Apache to provide *HTTPS*, the HTTP protocol over SSL/TLS. This allows a way to encrypt traffic using a protocol that does not itself provide encryption.

A *certificate* is a method used to distribute a *public key* and other information about a server and the organization who is responsible for it. Certificates can be digitally signed by a *Certification Authority*, or CA. A CA is a trusted third party that has confirmed that the information contained in the certificate is accurate.

Types of Certificates

To set up a secure server using public-key cryptography, in most cases, you send your certificate request (including your public key), proof of your company's identity, and payment to a CA. The CA verifies the certificate request and your identity, and then sends back a certificate for your secure server. Alternatively, you can create your own *self-signed* certificate.

> **Note**
>
> Note that self-signed certificates should not be used in most production environments.

Continuing the HTTPS example, a CA-signed certificate provides two important capabilities that a self-signed certificate does not:

- Browsers (usually) automatically recognize the CA signature and allow a secure connection to be made without prompting the user.
- When a CA issues a signed certificate, it is guaranteeing the identity of the organization that is providing the web pages to the browser.

Most of the software supporting SSL/TLS have a list of CAs whose certificates they automatically accept. If a browser encounters a certificate whose authorizing CA is not in the list, the browser asks the user to either accept or decline the connection. Also, other applications may generate an error message when using a self-signed certificate.

The process of getting a certificate from a CA is fairly easy. A quick overview is as follows:

1. Create a private and public encryption key pair.
2. Create a certificate signing request based on the public key. The certificate request contains information about your server and the company hosting it.
3. Send the certificate request, along with documents proving your identity, to a CA. We cannot tell you which certificate authority to choose. Your decision may be based on your past experiences, or on the experiences of your friends or colleagues, or purely on monetary factors.

   Once you have decided upon a CA, you need to follow the instructions they provide on how to obtain a certificate from them.
4. When the CA is satisfied that you are indeed who you claim to be, they send you a digital certificate.
5. Install this certificate on your secure server, and configure the appropriate applications to use the certificate.

Generating a Certificate Signing Request (CSR)

Whether you are getting a certificate from a CA or generating your own self-signed certificate, the first step is to generate a key.

If the certificate will be used by service daemons, such as Apache, Postfix, Dovecot, etc., a key without a passphrase is often appropriate. Not having a passphrase allows the services to start without manual intervention, usually the preferred way to start a daemon.

This section will cover generating a key with a passphrase, and one without. The non-passphrase key will then be used to generate a certificate that can be used with various service daemons.

> **Warning**
>
> Running your secure service without a passphrase is convenient because you will not need to enter the passphrase every time you start your secure service. But it is insecure and a compromise of the key means a compromise of the server as well.

To generate the *keys* for the Certificate Signing Request (CSR) run the following command from a terminal prompt:

```
openssl genrsa -des3 -out server.key 2048

Generating RSA private key, 2048 bit long modulus
.........................++++++
.......++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
```

You can now enter your passphrase. For best security, it should at least contain eight characters. The minimum length when specifying −des3 is four characters. As a best practice it should include numbers and/or punctuation and not be a word in a dictionary. Also remember that your passphrase is case-sensitive.

Re-type the passphrase to verify. Once you have re-typed it correctly, the server key is generated and stored in the server.key file.

Now create the insecure key, the one without a passphrase, and shuffle the key names:

```
openssl rsa -in server.key -out server.key.insecure
mv server.key server.key.secure
mv server.key.insecure server.key
```

The insecure key is now named server.key, and you can use this file to generate the CSR without passphrase.

To create the CSR, run the following command at a terminal prompt:

```
openssl req -new -key server.key -out server.csr
```

It will prompt you enter the passphrase. If you enter the correct passphrase, it will prompt you to enter Company Name, Site Name, Email Id, etc. Once you enter all these details, your CSR will be created and it will be stored in the server.csr file.

You can now submit this CSR file to a CA for processing. The CA will use this CSR file and issue the certificate. On the other hand, you can create self-signed certificate using this CSR.

Creating a Self-Signed Certificate

To create the self-signed certificate, run the following command at a terminal prompt:

```
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

The above command will prompt you to enter the passphrase. Once you enter the correct passphrase, your certificate will be created and it will be stored in the server.crt file.

> **Warning**
>
> If your secure server is to be used in a production environment, you probably need a CA-signed certificate. It is not recommended to use self-signed certificate.

Installing the Certificate

You can install the key file server.key and certificate file server.crt, or the certificate file issued by your CA, by running following commands at a terminal prompt:

```
sudo cp server.crt /etc/ssl/certs
sudo cp server.key /etc/ssl/private
```

Now simply configure any applications, with the ability to use public-key cryptography, to use the *certificate* and *key* files. For example, Apache can provide HTTPS, Dovecot can provide IMAPS and POP3S, etc.

Certification Authority

If the services on your network require more than a few self-signed certificates it may be worth the additional effort to setup your own internal Certification Authority (CA). Using certificates signed by your own CA, allows the various services using the certificates to easily trust other services using certificates issued from the same CA.

First, create the directories to hold the CA certificate and related files:

```
sudo mkdir /etc/ssl/CA
sudo mkdir /etc/ssl/newcerts
```

The CA needs a few additional files to operate, one to keep track of the last serial number used by the CA, each certificate must have a unique serial number, and another file to record which certificates have been issued:

```
sudo sh -c "echo '01' > /etc/ssl/CA/serial"
sudo touch /etc/ssl/CA/index.txt
```

The third file is a CA configuration file. Though not strictly necessary, it is very convenient when issuing multiple certificates. Edit /etc/ssl/openssl.cnf, and in the *[CA_default]* change:

```
dir             = /etc/ssl              # Where everything is kept
database        = $dir/CA/index.txt     # database index file.
certificate     = $dir/certs/cacert.pem # The CA certificate
serial          = $dir/CA/serial        # The current serial number
private_key     = $dir/private/cakey.pem# The private key
```

Next, create the self-signed root certificate:

```
openssl req -new -x509 -extensions v3_ca -keyout cakey.pem -out cacert.pem -
    days 3650
```

You will then be asked to enter the details about the certificate.

Now install the root certificate and key:

```
sudo mv cakey.pem /etc/ssl/private/
sudo mv cacert.pem /etc/ssl/certs/
```

You are now ready to start signing certificates. The first item needed is a Certificate Signing Request (CSR), see Generating a Certificate Signing Request (CSR) for details. Once you have a CSR, enter the following to generate a certificate signed by the CA:

```
sudo openssl ca -in server.csr -config /etc/ssl/openssl.cnf
```

After entering the password for the CA key, you will be prompted to sign the certificate, and again to commit the new certificate. You should then see a somewhat large amount of output related to the certificate creation.

There should now be a new file, /etc/ssl/newcerts/01.pem, containing the same output. Copy and paste everything beginning with the line: *—BEGIN CERTIFICATE—* and continuing through the line: *—END CERTIFICATE—* lines to a file named after the hostname of the server where the certificate will be installed. For example mail.example.com.crt, is a nice descriptive name.

Subsequent certificates will be named 02.pem, 03.pem, etc.

Note

Replace *mail.example.com.crt* with your own descriptive name.

Finally, copy the new certificate to the host that needs it, and configure the appropriate applications to use it. The default location to install certificates is /etc/ssl/certs. This enables multiple services to use the same certificate without overly complicated file permissions.

For applications that can be configured to use a CA certificate, you should also copy the /etc/ssl/certs/cacert.pem file to the /etc/ssl/certs/ directory on each server.

References

- The Wikipedia HTTPS page has more information regarding HTTPS.
- For more information on *OpenSSL* see the OpenSSL Home Page.
- Also, O'Reilly's Network Security with OpenSSL is a good in-depth reference.

Console Security

As with any other security barrier you put in place to protect your server, it is pretty tough to defend against untold damage caused by someone with physical access to your environment, for example, theft of hard drives, power or service disruption, and so on. Therefore, console security should be addressed merely as one component of your overall physical security strategy. A locked "screen door" may deter a casual criminal, or at the very least slow down a determined one, so it is still advisable to perform basic precautions with regard to console security.

The following instructions will help defend your server against issues that could otherwise yield very serious consequences.

Disable Ctrl+Alt+Delete

Anyone that has physical access to the keyboard can simply use the Ctrl+Alt+Delete key combination to reboot the server without having to log on. While someone could simply unplug the power source, you should still prevent the use of this key combination on a production server. This forces an attacker to take more drastic measures to reboot the server, and will prevent accidental reboots at the same time.

To disable the reboot action taken by pressing the Ctrl+Alt+Delete key combination, run the following two commands:

```
sudo systemctl mask ctrl-alt-del.target
sudo systemctl daemon-reload
```

eCryptfs is deprecated

eCryptfs is deprecated and should not be used, instead the LUKS setup as defined by the Ubuntu installer is recommended. That in turn - for a typical remote server setup will need a remote key store as usually no one is there to enter a key on boot.

Virtualization is being adopted in many different environments and situations. If you are a developer, virtualization can provide you with a contained environment where you can safely do almost any sort of development safe from messing up your main working environment. If you are a systems administrator, you can use virtualization to more easily separate your services and move them around based on demand.

The default virtualization technology supported in Ubuntu is KVM. For Intel and AMD hardware KVM requires virtualization extensions. But KVM is also available for IBM Z and LinuxONE, IBM POWER as well as for ARM64. Qemu is part of the KVM experience being the userspace backend for it, but it also can be used for hardware without virtualization extensions by using its TCG mode.

While virtualization is in many ways similar to containers those are different and implemented via other solutions like LXD, systemd-nspawn, containerd and others.

Multipass is the recommended method to create Ubuntu VMs on Ubuntu. It's designed for developers who want a fresh Ubuntu environment with a single command and works on Linux, Windows and macOS.

On Linux it's available as a snap:

```
sudo snap install multipass --beta --classic
```

Usage

Find available images

```
$ multipass find
Image                   Aliases           Version          Description
core                    core16            20190424         Ubuntu Core 16
core18                                    20190213         Ubuntu Core 18
16.04                   xenial            20190628         Ubuntu 16.04 LTS
18.04                   bionic,lts        20190627.1       Ubuntu 18.04 LTS
18.10                   cosmic            20190628         Ubuntu 18.10
19.04                   disco             20190628         Ubuntu 19.04
daily:19.10             devel,eoan        20190623         Ubuntu 19.10
```

Launch a fresh instance of the current Ubuntu LTS

```
$ multipass launch ubuntu
Launching dancing-chipmunk...
Downloading Ubuntu 18.04 LTS.........
Launched: dancing chipmunk
```

Check out the running instances

```
$ multipass list
Name                    State             IPv4             Release
dancing-chipmunk        RUNNING           10.125.174.247   Ubuntu 18.04 LTS
live-naiad              RUNNING           10.125.174.243   Ubuntu 18.04 LTS
snapcraft-asciinema     STOPPED           --               Ubuntu Snapcraft
    builder for Core 18
```

Learn more about the VM instance you just launched

```
$ multipass info dancing-chipmunk
Name:           dancing-chipmunk
State:          RUNNING
IPv4:           10.125.174.247
Release:        Ubuntu 18.04.1 LTS
Image hash:     19e9853d8267 (Ubuntu 18.04 LTS)
Load:           0.97 0.30 0.10
Disk usage:     1.1G out of 4.7G
Memory usage:   85.1M out of 985.4M
```

Connect to a running instance

```
$ multipass shell dancing-chipmunk
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-42-generic x86_64)

...
```

Don't forget to logout (or Ctrl-D) or you may find yourself heading all the way down the Inception levels… ;)

Run commands inside an instance from outside

```
$ multipass exec dancing-chipmunk -- lsb_release -a
No LSB modules are available.
Distributor ID:  Ubuntu
Description:     Ubuntu 18.04.1 LTS
Release:         18.04
Codename:        bionic
```

Stop an instance to save resources

```
$ multipass stop dancing-chipmunk
```

Delete the instance

```
$ multipass delete dancing-chipmunk
```

It will now show up as deleted: $ multipass list Name State IPv4 Release snapcraft-asciinema STOPPED -- Ubuntu Snapcraft builder for Core 18 dancing-chipmunk DELETED -- Not Available

And when you want to completely get rid of it:

```
$ multipass purge
```

Integrate into the rest of your virtualization

You might have other virtualization already based on libvirt either through using the similar older uvtool already or through the common virt-manager.

You might for example want those guests to be on the same bridge to communicate to each other or you need access to the graphical output for some reason.

Fortunately it is possible to integrate this by using the libvirt backend of multipass

```
$ sudo multipass set local.driver=libvirt
```

After that when you start a guest you can also access it via tools like virt-manager or virsh

```
$ multipass launch ubuntu
Launched: engaged-amberjack

$ virsh list
 Id    Name                           State
----------------------------------------------------
 15    engaged-amberjack              running
```

Get help

```
multipass help
multipass help <command>
```

See the multipass documentation for more details.

Qemu

Qemu is a machine emulator that can run operating systems and programs for one machine on a different machine. Mostly it is not used as emulator but as virtualizer in collaboration with KVM kernel components. In that case it utilizes the virtualization technology of the hardware to virtualize guests.

While qemu has a command line interface and a monitor to interact with running guests those is rarely used that way for other means than development purposes. Libvirt provides an abstraction from specific versions and hypervisors and encapsulates some workarounds and best practices.

Running Qemu/KVM

While there are much more user friendly and comfortable ways, using the command below is probably the quickest way to see some called Ubuntu moving on screen is directly running it from the netboot iso.

> **Warning**: this is just for illustration - not generally recommended without verifying the checksums; Multipass and UVTool are much better ways to get actual guests easily.

Run:

> sudo qemu−system−x86_64 −enable−kvm −cdrom http://archive.ubuntu.com/ubuntu/dists/bionic−updates/main/installer−amd64/current/images/netboot/mini.iso

You could download the ISO for faster access at runtime and e.g. add a disk to the same by:

- creating the disk > qemu−img create −f qcow2 disk.qcow 5G
- Using the disk by adding −drive file=disk.qcow,format=qcow2

Those tools can do much more, as you'll find in their respective (long) man pages. There also is a vast assortment of auxiliary tools to make them more consumable for specific use-cases and needs - for example virt-manager for UI driven use through libvirt. But in general - even the tools eventually use that - it comes down to:

> qemu-system-x86_64 options image[s]

So take a look at the man page of qemu, qemu-img and the documentation of qemu and see which options are the right one for your needs.

Graphics

Graphics for qemu/kvm always comes in two pieces.

- A front end - controlled via the −vga argument - which is provided to the guest. Usually one of cirrus, std, qxl, virtio. The default these days is qxl which strikes a good balance between guest compatibility and performance. The guest needs a driver for what is selected, which is the most common reason to switch from the default to either cirrus (e.g. very old Windows versions)
- A back end - controlled via the −display argument - which is what the host uses to actually display the graphical content. That can be an application window via gtk or a vnc.
- In addition one can enable the −spice back-end (can be done in addition to vnc) which can be faster and provides more authentication methods than vnc.
- if you want no graphical output at all you can save some memory and cpu cycles by setting −nographic

If you run with spice or vnc you can use native vnc tools or virtualization focused tools like virt−viewer. More about these in the libvirt section.

All those options above are considered basic usage of graphics. There are advanced options for further needs. Those cases usually differ in their ease-of-use and capability are:

- *Need some 3D acceleration*: −vga virtio with a local display having a GL context −display gtk,gl=on; That will use virgil3d on the host and needs guest drivers for [virt3d] which are common in Linux since Kernels >=4.4 but hard to get by for other cases. While not as fast as the next two options, the big benefit is that it can be used without additional hardware and without a proper IOMMU setup for device passthrough.
- *Need native performance*: use PCI passthrough of additional GPUs in the system. You'll need an IOMMU setup and unbind the cards from the host before you can pass it through like −device vfio−pci,host=05:00.0,bus=1,addr=00.0,multifunction=on,x−vga=on −device vfio−pci,host =05:00.1,bus=1,addr=00.1
- *Need native performance, but multiple guests per card*: Like PCI Passthrough, but using mediated devices to shard a card on the Host into multiple devices and pass those like −display gtk,gl =on −device vfio−pci,sysfsdev=/sys/bus/pci/devices/0000:00:02.0/4dd511f6−ec08−11e8−b839−2 f163ddee3b3,display=on,rombar=0. More at kraxel on vgpu and Ubuntu GPU mdev evaluation. The sharding of the cards is driver specific and therefore will differ per manufacturer like Intel or Nvidia.

Especially the advanced cases can get pretty complex, therefore it is recommended to use qemu through libvirt for those cases. Libvirt will take care of all but the host kernel/bios tasks of such configurations.

Upgrading the machine type

If you are unsure what this is, you might consider this as buying (virtual) Hardware of the same spec but a newer release date. You are encouraged in general and might want to update your machine type of an existing defined guests in particular to:

- to pick up latest security fixes and features
- continue using a guest created on a now unsupported release

In general it is recommended to update machine types when upgrading qemu/kvm to a new major version. But this can likely never be an automated task as this change is guest visible. The guest devices might change in appearance, new features will be announced to the guest and so on. Linux is usually very good at tolerating such changes, but it depends so much on the setup and workload of the guest that this has to be evaluated by the owner/admin of the system. Other operating systems where known to often have severe impacts by changing the hardware. Consider a machine type change similar to replacing all devices and firmware of a physical machine to the latest revision - all considerations that apply there apply to evaluating a machine type upgrade as well.

As usual with major configuration changes it is wise to back up your guest definition and disk state to be able to do a rollback just in case. There is no integrated single command to update the machine type via virsh or similar tools. It is a normal part of your machine definition. And therefore updated the same way as most others.

First shutdown your machine and wait until it has reached that state.

```
virsh shutdown <yourmachine>
# wait
virsh list --inactive
# should now list your machine as "shut off"
```

Then edit the machine definition and find the type in the type tag at the machine attribute.

```
virsh edit <yourmachine>
<type arch='x86_64' machine='pc-i440fx-bionic'>hvm</type>
```

Change this to the value you want. If you need to check what types are available via "-M ?" Note that while providing upstream types as convenience only Ubuntu types are supported. There you can also see what the current default would be. In general it is strongly recommended that you change to newer types if possible to exploit newer features, but also to benefit of bugfixes that only apply to the newer device virtualization.

```
kvm -M ?
# lists machine types, e.g.
pc-i440fx-xenial       Ubuntu 16.04 PC (i440FX + PIIX, 1996) (default)
...
pc-i440fx-bionic       Ubuntu 18.04 PC (i440FX + PIIX, 1996) (default)
...
```

After this you can start your guest again. You can check the current machine type from guest and host depending on your needs.

```
virsh start <yourmachine>
# check from host, via dumping the active xml definition
virsh dumpxml <yourmachine> | xmllint --xpath "string(//domain/os/type/
    @machine)" -
# or from the guest via dmidecode (if supported)
sudo dmidecode | grep Product -A 1
        Product Name: Standard PC (i440FX + PIIX, 1996)
        Version: pc-i440fx-bionic
```

If you keep non-live definitions around - like xml files - remember to update those as well.

> **Note**
>
> This also is documented along some more constraints and considerations at the Ubuntu Wiki

QEMU usage for microvms

QEMU became another use case being used in a container-like style providing an enhanced isolation compared to containers but being focused on initialization speed.

To achieve that several components have been added: - the microvm machine type - alternative simple FW that can boot linux called qboot - qemu build with reduced features matching these use cases called qemu-system-x86-microvm

For example if you happen to already have a stripped down workload that has all it would execute in an initrd you would run it maybe like the following:

$ sudo qemu-system-x86_64 -M ubuntu-q35 -cpu host -m 1024 -enable-kvm -serial mon:stdio -nographic -display curses -append 'console=ttyS0,115200,8n1' -kernel vmlinuz-5.4.0-21 -initrd /boot/initrd.img-5.4.0-21-workload

To run the same with microvm, qboot and the minimized qemu you would do the following

1. run it with with type microvm, so change -M to -M microvm
2. use the qboot bios, add -bios /usr/share/qemu/bios-microvm.bin
3. install the feature-minimized qemu-system package, do $ sudo apt install qemu-system-x86-microvm

An invocation will now look like:

$ sudo qemu-system-x86_64 -M microvm -bios /usr/share/qemu/bios-microvm.bin -cpu host -m 1024 -enable-kvm -serial mon:stdio -nographic -display curses -append 'console=ttyS0,115200,8n1' -kernel vmlinuz-5.4.0-21 -initrd /boot/initrd.img-5.4.0-21-workload

That will have cut down the qemu, bios and virtual-hw initialization time down a lot. You will now - more than you already have before - spend the majority inside the guest which implies that further tuning probably has to go into that kernel and userspace initialization time.

> ** Note ** For now microvm, the qboot bios and other components of this are rather new upstream and not as verified as many other parts of the virtualization stack. Therefore none of the above is the default. Further being the default would also mean many upgraders would regress finding a qemu that doesn't have most features they are used to use. Due to that the qemu-system-x86-microvm package is intentionally a strong opt-in conflicting with the normal qemu-system-x86 package.

libvirt

The libvirt library is used to interface with different virtualization technologies. Before getting started with libvirt it is best to make sure your hardware supports the necessary virtualization extensions for KVM. Enter the following from a terminal prompt:

```
kvm–ok
```

A message will be printed informing you if your CPU *does* or *does not* support hardware virtualization.

> **Note**
>
> On many computers with processors supporting hardware assisted virtualization, it is necessary to activate an option in the BIOS to enable it.

Virtual Networking

There are a few different ways to allow a virtual machine access to the external network. The default virtual network configuration includes *bridging* and *iptables* rules implementing *usermode* networking, which uses the SLIRP protocol. Traffic is NATed through the host interface to the outside network.

To enable external hosts to directly access services on virtual machines a different type of *bridge* than the default needs to be configured. This allows the virtual interfaces to connect to the outside network through the physical interface, making them appear as normal hosts to the rest of the network.

There is a great example how to configure an own bridge and combining it with libvirt so that guests will use it at the netplan.io.

Installation

To install the necessary packages, from a terminal prompt enter:

```
sudo apt install qemu–kvm libvirt –daemon–system
```

After installing libvirt-daemon-system, the user used to manage virtual machines will need to be added to the *libvirt* group. This is done automatically for members of the sudo group, but needs to be done in additon for anyone else that should access system wide libvirt resources. Doing so will grant the user access to the advanced networking options.

In a terminal enter:

```
sudo adduser $USER libvirt
```

Note

If the user chosen is the current user, you will need to log out and back in for the new group membership to take effect.

You are now ready to install a *Guest* operating system. Installing a virtual machine follows the same process as installing the operating system directly on the hardware.

You either need: - a way to automate the installation - a keyboard and monitor will need to be attached to the physical machine. - use cloud images which are meant to self-initialize (see Multipass and UVTool)

In the case of virtual machines a Graphical User Interface (GUI) is analogous to using a physical keyboard and mouse on a real computer. Instead of installing a GUI the virt-viewer or virt-manager application can be used to connect to a virtual machine's console using VNC. See Virtual Machine Manager / Viewer for more information.

Virtual Machine Management

The following section covers the command-line tools around *virsh* that are part of libvirt itself. But there are various options at different levels of complexities and feature-sets, like:

- multipass
- uvt
- virt-* tools
- openstack

virsh

There are several utilities available to manage virtual machines and libvirt. The virsh utility can be used from the command line. Some examples:

- To list running virtual machines:

  ```
  virsh list
  ```

- To start a virtual machine:

  ```
  virsh start <guestname>
  ```

- Similarly, to start a virtual machine at boot:

  ```
  virsh autostart <guestname>
  ```

- Reboot a virtual machine with:

  ```
  virsh reboot <guestname>
  ```

- The *state* of virtual machines can be saved to a file in order to be restored later. The following will save the virtual machine state into a file named according to the date:

  ```
  virsh save <guestname> save-my.state
  ```

  Once saved the virtual machine will no longer be running.

- A saved virtual machine can be restored using:

  ```
  virsh restore save-my.state
  ```

- To shutdown a virtual machine do:

```
virsh shutdown <guestname>
```

- A CDROM device can be mounted in a virtual machine by entering:

  ```
  virsh attach-disk <guestname> /dev/cdrom /media/cdrom
  ```

- To change the definition of a guest virsh exposes the domain via

  ```
  virsh edit <guestname>
  ```

That will allow one to edit the XML representation that defines the guest and when saving it will apply format and integrity checks on these definitions.

Editing the XML directly certainly is the most powerful way, but also the most complex one. Tools like Virtual Machine Manager / Viewer can help unexperienced users to do most of the common tasks.

> If virsh (or other vir* tools) shall connect to something else than the default qemu-kvm/system hipervisor one can find alternatives for the *connect* option in *man virsh* or libvirt doc

system and session scope

Virsh - as well as most other tools to manage virtualization - can be passed connection strings.

$ virsh –connect qemu:///system

There are two options for the connection.

- qemu:///system - connect locally as root to the daemon supervising QEMU and KVM domains
- qemu:///session - connect locally as a normal user to his own set of QEMU and KVM domains

The *default* always was (and still is) qemu:///system as that is the behavior users are used to. But there are a few benefits (and drawbacks) to qemu:///session to consider it.

qemu:///session is per user and can on a multi-user system be used to separate the people. But most importantly things run under the permissions of the user which means no permission struggle on the just donwloaded image in your *$HOME* or the just attached USB-stick. On the other hand it can't access system resources that well, which includes network setup that is known to be hard with qemu:///session. It falls back to slirp networking which is functional but slow and makes it impossible to be reached from other systems.

qemu:///system is different in that it is run by the global system wide libvirt that can arbitrate resources as needed. But you might need to mv and/or chown files to the right places permssions to have them usable.

Applications usually will decide on their primary use-case. Desktop-centric applications often choose qemu :///session while most solutions that involve an administrator anyway continue to default to qemu:///system.

Read more about that in the libvirt FAQ and this blog about the topic.

Migration

There are different types of migration available depending on the versions of libvirt and the hipervisor being used. In general those types are:

- offline migration
- live migration
- postcopy migration

There are various options to those methods, but the entry point for all of them is *virsh migrate*. Read the integrated help for more detail.

```
virsh migrate --help
```

Some useful documentation on constraints and considerations about live migration can be found at the Ubuntu Wiki

Device Passthrough / Hotplug

If instead of the here described hotplugging you want to always pass through a device add the xml content of the device to your static guest xml representation via e.g. virsh edit <guestname>. In that case you don't need to use *attach/detach*. There are different kinds of passthrough. Types available to you depend on your Hardware and software setup.

- USB hotplug/passthrough
- VF hotplug/Passthrough

But both kinds are handled in a very similar way and while there are various way to do it (e.g. also via qemu monitor) driving such a change via libvirt is recommended. That way libvirt can try to manage all sorts of special cases for you and also somewhat masks version differences.

In general when driving hotplug via libvirt you create a xml snippet that describes the device just as you would do in a static guest description. A usb device is usually identified by Vendor/Product id's:

```
<hostdev mode='subsystem' type='usb' managed='yes'>
  <source>
    <vendor id='0x0b6d'/>
    <product id='0x3880'/>
  </source>
</hostdev>
```

Virtual functions are usually assigned via their PCI-ID (domain, bus, slot, function).

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x04' slot='0x10' function='0x0'/>
  </source>
</hostdev>
```

Note

To get the Virtual function in the first place is very device dependent and can therefore not be fully covered here. But in general it involves setting up an iommu, registering via VFIO and sometimes requesting a number of VFs. Here an example on ppc64el to get 4 VFs on a device:

```
$ sudo modprobe vfio-pci
# identify device
$ lspci -n -s 0005:01:01.3
0005:01:01.3 0200: 10df:e228 (rev 10)
# register and request VFs
$ echo 10df e228 | sudo tee /sys/bus/pci/drivers/vfio-pci/new_id
$ echo 4 | sudo tee /sys/bus/pci/devices/0005\:01\:00.0/sriov_numvfs
```

You then attach or detach the device via libvirt by relating the guest with the xml snippet.

```
virsh attach-device <guestname> <device-xml>
# Use the Device int the Guest
virsh detach-device <guestname> <device-xml>
```

Access Qemu Monitor via libvirt

The Qemu Monitor is the way to interact with qemu/KVM while a guest is running. This interface has many and very powerful features for experienced users. When running under libvirt that monitor interface is bound by libvirt itself for management purposes, but a user can run qemu monitor commands via libvirt still. The general syntax is virsh qemu-monitor-command [options] [guest] 'command'

Libvirt covers most use cases needed, but if you every want/need to work around libvirt or want to tweak very special options you can e.g. add a device that way:

```
virsh qemu-monitor-command --hmp focal-test-log 'drive_add 0 if=none,file=/var
    /lib/libvirt/images/test.img,format=raw,id=disk1'
```

But since the monitor is so powerful, you can do a lot especially for debugging purposes like showing the guest registers:

```
virsh qemu-monitor-command --hmp y-ipns 'info registers'
RAX=00ffffc000000000 RBX=ffff8f0f5d5c7e48 RCX=0000000000000000 RDX=
    ffffea00007571c0
RSI=0000000000000000 RDI=ffff8f0fdd5c7e48 RBP=ffff8f0f5d5c7e18 RSP=
    ffff8f0f5d5c7df8
[...]
```

Huge Pages

Using huge pages can help to reduce TLB pressure, page table overhead and speed up some further memory relate actions. Furthermore by default Transparent huge pages are useful, but can be quite some overhead - so if it is clear that using huge pages is preferred making them explicit usually has some gains.

While huge page are admittedly harder to manage, especially later in the lifetime of a system if memory is fragmented they provide a useful boost especially for rather large guests.

Bonus: When using device pass through on very large guests there is an extra benefit of using huge pages as it is faster to do the initial memory clear on vfio dma pin.

Huge page allocation

Huge pages come in different sizes. A *normal* page usually is 4k and huge pages are eithe 2M or 1G, but depending on the architecture other options are possible.

The most simple, yet least reliable way to allocate some huge pages is to just echo a value to sysfs Be sure to re-check if it worked.

```
$ echo 256 | sudo tee /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
$ cat /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
256
```

There one of these sizes is "default huge page size" which will be used in the auto-mounted */dev/hugepages*. Changing the default size requires a reboot and is set via default_hugepagesz

You can check the current default size:

```
$ grep Hugepagesize /proc/meminfo
Hugepagesize:       2048 kB
```

But there can be more than one at the same time one better check:

```
$ tail /sys/kernel/mm/hugepages/hugepages-*/nr_hugepages`
==> /sys/kernel/mm/hugepages/hugepages-1048576kB/nr_hugepages <==
0
==> /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages <==
2
```

And even that could on bigger systems be further split per Numa node.

One can allocate huge pages at boot or runtime, but due to fragmentation there are no guarantees it works later. The kernel documentation lists details on both ways.

Huge pages need to be allocated by the kernel as mentioned above but to be consumable they also have to be mounted. By default *systemd* will make /dev/hugepages available for the default huge page size. Feel free to add more mount points if you need different sized. An overview can be queried with

```
$ hugeadm --list-all-mounts
Mount Point             Options
/dev/hugepages          rw,relatime,pagesize=2M
```

A one-stop info for the overall huge page status of the system can be reported with

```
$ hugeadm --explain
```

Huge page usage in libvirt

With the above in place libvirt can map guest memory to huge pages. In a guest definition add the most simple form of

```
<memoryBacking>
  <hugepages/>
</memoryBacking>
```

That will allocate the huge pages using the default huge page size from a autodetected mountpoint. For more control e.g. how memory is spread over Numa nodes or which page size to use check out the details at the libvirt doc.

Apparmor isolation

By default libvirt will spawn qemu guests using apparmor isolation for enhanced security. The apparmor rules for a guest will consist of multiple elements:

- a static part that all guests share => /etc/apparmor.d/abstractions/libvirt-qemu
- a dynamic part created at guest start time and modified on hotplug/unplug => /etc/apparmor.d/libvirt/libvirt-f9533e35-6b63-45f5-96be-7cccc9696d5e.files

Of the above the former is provided and updated by the libvirt-daemon package and the latter is generated on guest start. None of the two should be manually edited. They will by default cover the vast majority of use cases and work fine. But there are certain cases where users either want to:

- further lock down the guest (e.g. by explicitly denying access that usually would be allowed)
- open up the guest isolation (most of the time this is needed if the setup on the local machine does not follow the commonly used paths)

To do so there are two files to do so. Both are local overrides which allow you to modify them without getting them clobbered or command file prompts on package upgrades.

- /etc/apparmor.d/local/abstractions/libvirt−qemu this will be applied to every guest. Therefore it is rather powerful, but also a rather blunt tool. It is a quite useful place thou to add additional deny rules.
- /etc/apparmor.d/local/usr.lib. libvirt . virt−aa−helper the above mentioned *dynamic part* that is individual per guest is generated by a tool called libvirt . virt−aa−helper. That is under apparmor isolation as well. This is most commonly used if you want to use uncommon paths as it allows to ahve those uncommon paths in the guest XML (see virsh edit) and have those paths rendered to the per-guest dynamic rules.

Resources

- See the KVM home page for more details.
- For more information on libvirt see the libvirt home page
  - xml configuration of domains and storage being the most often used libvirt reference
- Another good resource is the Ubuntu Wiki KVM page.
- For basics how to assign VT-d devices to qemu/KVM, please see the linux-kvm page.

Cloud images and uvtool

Introduction

With Ubuntu being one of the most used operating systems on many cloud platforms, the availability of stable and secure cloud images has become very important. As of 12.04 the utilization of cloud images outside of a cloud infrastructure has been improved. It is now possible to use those images to create a virtual machine without the need of a complete installation.

Creating virtual machines using uvtool

Starting with 14.04 LTS, a tool called uvtool greatly facilitates the task of generating virtual machines (VM) using the cloud images. uvtool provides a simple mechanism to synchronize cloud-images locally and use them to create new VMs in minutes.

Uvtool packages

The following packages and their dependencies will be required in order to use uvtool:

- uvtool
- uvtool-libvirt

To install uvtool, run:

```
$ sudo apt −y install uvtool
```

This will install uvtool's main commands:

- uvt-simplestreams-libvirt
- uvt-kvm

Get the Ubuntu Cloud Image with uvt-simplestreams-libvirt

This is one of the major simplifications that uvtool brings. It is aware of where to find the cloud images so only one command is required to get a new cloud image. For instance, if you want to synchronize all cloud images for the amd64 architecture, the uvtool command would be:

```
$ uvt-simplestreams-libvirt --verbose sync arch=amd64
```

After an amount of time required to download all the images from the Internet, you will have a complete set of cloud images stored locally. To see what has been downloaded use the following command:

```
$ uvt-simplestreams-libvirt query
release=bionic arch=amd64 label=daily (20191107)
release=focal arch=amd64 label=daily (20191029)
...
```

In the case where you want to synchronize only one specific cloud-image, you need to use the release= and arch= filters to identify which image needs to be synchronized.

```
$ uvt-simplestreams-libvirt sync release=DISTRO_SHORT_CODENAME arch=amd64
```

Furthermore you can provide an alternative URL to fetch images from. A common case are the daily images which helps to get the very latest images or if you need access to the not yet released development release of Ubuntu.

```
$ uvt-simplestreams-libvirt sync --source http://cloud-images.ubuntu.com/daily
    [... further options]
```

Create the VM using uvt-kvm

In order to connect to the virtual machine once it has been created, you must have a valid SSH key available for the Ubuntu user. If your environment does not have an SSH key, you can easily create one using the following command:

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
4d:ba:5d:57:c9:49:ef:b5:ab:71:14:56:6e:2b:ad:9b ubuntu@DISTRO_SHORT_CODENAMES
The key's randomart image is:
+--[ RSA 2048]----+
|                 ..|
|                o.=|
|           .     **|
|          +     o+=|
|         S . ...=.|
|          o . .+ .|
|         . .  o o |
|              *   |
|             E    |
+-----------------+
```

To create of a new virtual machine using uvtool, run the following in a terminal:

```
$ uvt-kvm create firsttest
```

This will create a VM named **firsttest** using the current LTS cloud image available locally. If you want to specify a release to be used to create the VM, you need to use the **release=** filter:

```
$ uvt-kvm create secondtest release=DISTRO-SHORT-CODENAME
```

uvt-kvm wait can be used to wait until the creation of the VM has completed:

```
$ uvt-kvm wait secondttest
```

Connect to the running VM

Once the virtual machine creation is completed, you can connect to it using SSH:

```
$ uvt-kvm ssh secondtest
```

You can also connect to your VM using a regular SSH session using the IP address of the VM. The address can be queried using the following command:

```
$ uvt-kvm ip secondtest
192.168.122.199
$ ssh -i ~/.ssh/id_rsa ubuntu@192.168.122.199
[...]
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@secondtest:~$
```

Get the list of running VMs

You can get the list of VMs running on your system with this command:

```
$ uvt-kvm list
secondtest
```

Destroy your VM

Once you are done with your VM, you can destroy it with:

```
$ uvt-kvm destroy secondtest
```

Note: other than libvirts destroy action this will by default also remove the associated virtual storage files.

More uvt-kvm options

The following options can be used to change some of the characteristics of the VM that you are creating:

- --memory : Amount of RAM in megabytes. Default: 512.
- --disk : Size of the OS disk in gigabytes. Default: 8.
- --cpu : Number of CPU cores. Default: 1.

Some other parameters will have an impact on the cloud-init configuration:

- --password password : Allow login to the VM using the Ubuntu account and this provided password.
- --run-script-once script_file : Run script_file as root on the VM the first time it is booted, but never again.
- --packages package_list : Install the comma-separated packages specified in package_list on first boot.

A complete description of all available modifiers is available in the manpage of uvt-kvm.

Resources

If you are interested in learning more, have questions or suggestions, please contact the Ubuntu Server Team at:

- IRC: #ubuntu-server on freenode
- Mailing list: ubuntu-server at lists.ubuntu.com

Introduction

The virt-manager source contains not only virt-manager itself but also a collection of further helpful tools like virt-install, virt-clone and virt-viewer.

Virtual Machine Manager

The virt-manager package contains a graphical utility to manage local and remote virtual machines. To install virt-manager enter:

```
sudo apt install virt-manager
```

Since virt-manager requires a Graphical User Interface (GUI) environment it is recommended to be installed on a workstation or test machine instead of a production server. To connect to the local libvirt service enter:

```
virt-manager
```

You can connect to the libvirt service running on another host by entering the following in a terminal prompt:

```
virt-manager -c qemu+ssh://virtnode1.mydomain.com/system
```

> **Note**
>
> The above example assumes that SSH connectivity between the management system and the target system has already been configured, and uses SSH keys for authentication. SSH *keys* are needed because libvirt sends the password prompt to another process.

virt-manager guest lifecycle

When using virt-manager it is always important to know the context you look at. The main window initially lists only the currently defined guests, you'll see their *name*, *state* and a small chart on *cpu usage*.

virt-manager-gui-start|499x597

On that context there isn't much one can do except start/stop a guest. But by double-clicking on a guest or by clicking the *open* button at the top one can see the guest itself. For a running guest that includes the guests main-console/virtual-screen output.

virt-manager-gui-showoutput|690x386

If you are deeper in the guest config a click in the top left onto "*show the graphical console*" will get you back to this output.

virt-manager guest modification

virt−manager provides a gui assisted way to edit guest definitions which can be handy. To do so the per-guest context view will at the top have "*show virtual hardware details*". Here a user can edit the virtual hardware of the guest which will under the cover alter the guest representation.

virt-manager-gui-edit|690x346

The UI edit is limited to the features known and supported to that GUI feature. Not only does libvirt grow features faster than virt-manager can keep up - adding every feature would also overload the UI to the extend to be unusable. To strike a balance between the two there also is the XML view which can be reached via the "*edit libvirt XML*" button.

virt-manager-gui-XML|690x346

By default this will be read-only and you can see what the UI driven actions have changed, but one can allow RW access in this view in the *preferences*. This is the same content that the virsh edit of the libvirt-client exposes.

Virtual Machine Viewer

The virt-viewer application allows you to connect to a virtual machine's console like virt-manager reduced to the GUI functionality. virt-viewer does require a Graphical User Interface (GUI) to interface with the virtual machine.

To install virt-viewer from a terminal enter:

```
sudo apt install virt−viewer
```

Once a virtual machine is installed and running you can connect to the virtual machine's console by using:

```
virt−viewer <guestname>
```

The UI will be a window representing the virtual screen of the guest, just like virt-manager above but without the extra buttons and features around it.

virt-viewer-gui-showoutput|690x598

Similar to virt-manager, virt-viewer can connect to a remote host using *SSH* with key authentication, as well:

```
virt−viewer −c qemu+ssh://virtnode1.mydomain.com/system <guestname>
```

Be sure to replace *web_devel* with the appropriate virtual machine name.

If configured to use a *bridged* network interface you can also setup SSH access to the virtual machine.

virt-install

virt-install is part of the virtinst package. It can help installing classic ISO based systems and provides a CLI options for the most common options needed to do so. To install it, from a terminal prompt enter:

sudo apt install virtinst

There are several options available when using virt-install. For example:

```
virt-install -n web_devel -r 8192 \
--disk path=/home/doug/vm/web_devel.img,bus=virtio,size=50 \
-c focal-desktop-amd64.iso \
--network network=default,model=virtio \
--video=vmvga --graphics vnc,listen=0.0.0.0 --noautoconsole -v --vcpus=4
```

There are much more arguments that can be found in the man page, explaining those of the example above one by one: * -n web_devel the name of the new virtual machine will be web_devel in this example. * -r 8192 specifies the amount of memory the virtual machine will use in megabytes. * –disk path=/home/doug/vm/web_devel.img,bus=virtio,size=50 indicates the path to the virtual disk which can be a file, partition, or logical volume. In this example a file named web_devel.img in the current users directory, with a size of 50 gigabytes, and using virtio for the disk bus. Depending on the disk path, virt-install my need to be run with elevated privileges. * -c focal-desktop-amd64.iso file to be used as a virtual CDROM. The file can be either an ISO file or the path to the host's CDROM device. * –network provides details related to the VM's network interface. Here the default network is used, and the interface model is configured for virtio. * –video=vmvga the video driver to use. * –graphics vnc,listen=0.0.0.0 exports the guest's virtual console using VNC and on all host interfaces. Typically servers have no GUI, so another GUI based computer on the Local Area Network (LAN) can connect via VNC to complete the installation. * –noautoconsole will not automatically connect to the virtual machine's console. * -v: creates a fully virtualized guest. * –vcpus=4 allocate 4 virtual CPUs.

After launching virt-install you can connect to the virtual machine's console either locally using a GUI (if your server has a GUI), or via a remote VNC client from a GUI-based computer.

virt-clone

The virt-clone application can be used to copy one virtual machine to another. For example:

```
virt-clone --auto-clone --original focal
```

Options used: * –auto-clone: to have virt-clone come up with guest names and disk paths on its own * –original: name of the virtual machine to copy

Also, use -d or –debug option to help troubleshoot problems with virt-clone.

Replace *focal* and with appropriate virtual machine names of your case.

> Warning: please be aware that this is a full clone, therefore any sorts of secrets, keys and for example /etc/machine-id will be shared causing e.g. issues to security and anything that needs to identify the machine like DHCP. You most likely want to edit those afterwards and de-duplicate them as needed.

Resources

- See the KVM home page for more details.
- For more information on libvirt see the libvirt home page
- The Virtual Machine Manager site has more information on virt-manager development.

LXD

LXD (pronounced lex-dee) is the lightervisor, or lightweight container hypervisor. LXC (lex-see) is a program which creates and administers "containers" on a local system. It also provides an API to allow higher level

managers, such as LXD, to administer containers. In a sense, one could compare LXC to QEMU, while comparing LXD to libvirt.

The LXC API deals with a 'container'. The LXD API deals with 'remotes', which serve images and containers. This extends the LXC functionality over the network, and allows concise management of tasks like container migration and container image publishing.

LXD uses LXC under the covers for some container management tasks. However, it keeps its own container configuration information and has its own conventions, so that it is best not to use classic LXC commands by hand with LXD containers. This document will focus on how to configure and administer LXD on Ubuntu systems.

Online Resources

There is excellent documentation for getting started with LXD and an online server allowing you to try out LXD remotely. Stephane Graber also has an excellent blog series on LXD 2.0. Finally, there is great documentation on how to drive lxd using juju.

This document will offer an Ubuntu Server-specific view of LXD, focusing on administration.

Installation

LXD is pre-installed on Ubuntu Server cloud images. On other systems, the lxd package can be installed using:

```
sudo snap install lxd
```

This will install the self-contained LXD snap package.

Kernel preparation

In general, Ubuntu should have all the desired features enabled by default. One exception to this is that in order to enable swap accounting the boot argument swapaccount=1 must be set. This can be done by appending it to the GRUB_CMDLINE_LINUX_DEFAULT=variable in /etc/default/grub, then running 'update-grub' as root and rebooting.

Configuration

In order to use LXD, some basic settings need to be configured first. This is done by running lxd init , which will allow you to choose:

- Directory or ZFS container backend. If you choose ZFS, you can choose which block devices to use, or the size of a file to use as backing store.
- Availability over the network.
- A 'trust password' used by remote clients to vouch for their client certificate.

You must run 'lxd init' as root. 'lxc' commands can be run as any user who is a member of group lxd. If user joe is not a member of group 'lxd', you may run:

```
adduser joe lxd
```

as root to change it. The new membership will take effect on the next login, or after running newgrp lxd from an existing login.

For more information on server, container, profile, and device configuration, please refer to the definitive configuration provided with the source code, which can be found online.

Creating your first container

This section will describe the simplest container tasks.

Creating a container

Every new container is created based on either an image, an existing container, or a container snapshot. At install time, LXD is configured with the following image servers:

- ubuntu: this serves official Ubuntu server cloud image releases.
- ubuntu−daily: this serves official Ubuntu server cloud images of the daily development releases.
- images: this is a default-installed alias for images.linuxcontainers.org. This is serves classical lxc images built using the same images which the LXC 'download' template uses. This includes various distributions and minimal custom-made Ubuntu images. This is not the recommended server for Ubuntu images.

The command to create and start a container is

```
lxc launch remote:image containername
```

Images are identified by their hash, but are also aliased. The ubuntu remote knows many aliases such as 18.04 and bionic. A list of all images available from the Ubuntu Server can be seen using:

```
lxc image list ubuntu:
```

To see more information about a particular image, including all the aliases it is known by, you can use:

```
lxc image info ubuntu:bionic
```

You can generally refer to an Ubuntu image using the release name (bionic) or the release number (18.04). In addition, lts is an alias for the latest supported LTS release. To choose a different architecture, you can specify the desired architecture:

```
lxc image info ubuntu:lts/arm64
```

Now, let's start our first container:

```
lxc launch ubuntu:bionic b1
```

This will download the official current Bionic cloud image for your current architecture, then create a container named b1 using that image, and finally start it. Once the command returns, you can see it using:

```
lxc list
lxc info b1
```

and open a shell in it using:

```
lxc exec b1 -- bash
```

The try-it page mentioned above gives a full synopsis of the commands you can use to administer containers.

Now that the xenial image has been downloaded, it will be kept in sync until no new containers have been created based on it for (by default) 10 days. After that, it will be deleted.

LXD Server Configuration

By default, LXD is socket activated and configured to listen only on a local UNIX socket. While LXD may not be running when you first look at the process listing, any LXC command will start it up. For instance:

```
lxc list
```

This will create your client certificate and contact the LXD server for a list of containers. To make the server accessible over the network you can set the http port using:

```
lxc config set core.https_address :8443
```

This will tell LXD to listen to port 8843 on all addresses.

Authentication

By default, LXD will allow all members of group lxd to talk to it over the UNIX socket. Communication over the network is authorized using server and client certificates.

Before client c1 wishes to use remote r1, r1 must be registered using:

```
lxc remote add r1 r1.example.com:8443
```

The fingerprint of r1's certificate will be shown, to allow the user at c1 to reject a false certificate. The server in turn will verify that c1 may be trusted in one of two ways. The first is to register it in advance from any already-registered client, using:

```
lxc config trust add r1 certfile.crt
```

Now when the client adds r1 as a known remote, it will not need to provide a password as it is already trusted by the server.

The other step is to configure a 'trust password' with r1, either at initial configuration using lxd init, or after the fact using:

```
lxc config set core.trust_password PASSWORD
```

The password can then be provided when the client registers r1 as a known remote.

Backing store

LXD supports several backing stores. The recommended and the default backing store is zfs. If you already have a ZFS pool configured, you can tell LXD to use it during the lxd init procedure, otherwise a file-backed zpool will be created automatically. With ZFS, launching a new container is fast because the filesystem starts as a copy on write clone of the images' filesystem. Note that unless the container is privileged (see below) LXD will need to change ownership of all files before the container can start, however this is fast and change very little of the actual filesystem data.

The other supported backing stores are described in detail in the Storage configuration section of the LXD documentation.

Container configuration

Containers are configured according to a set of profiles, described in the next section, and a set of container-specific configuration. Profiles are applied first, so that container specific configuration can override profile configuration.

Container configuration includes properties like the architecture, limits on resources such as CPU and RAM, security details including apparmor restriction overrides, and devices to apply to the container.

Devices can be of several types, including UNIX character, UNIX block, network interface, or disk. In order to insert a host mount into a container, a 'disk' device type would be used. For instance, to mount /opt in container c1 at /opt, you could use:

```
lxc config device add c1 opt disk source=/opt path=opt
```

See:

```
lxc help config
```

for more information about editing container configurations. You may also use:

```
lxc config edit c1
```

to edit the whole of c1's configuration. Comments at the top of the configuration will show examples of correct syntax to help administrators hit the ground running. If the edited configuration is not valid when the editor is exited, then the editor will be restarted.

Profiles

Profiles are named collections of configurations which may be applied to more than one container. For instance, all containers created with lxc launch, by default, include the default profile, which provides a network interface eth0.

To mask a device which would be inherited from a profile but which should not be in the final container, define a device by the same name but of type 'none':

```
lxc config device add c1 eth1 none
```

Nesting

Containers all share the same host kernel. This means that there is always an inherent trade-off between features exposed to the container and host security from malicious containers. Containers by default are therefore restricted from features needed to nest child containers. In order to run lxc or lxd containers under a lxd container, the security.nesting feature must be set to true:

```
lxc config set container1 security.nesting true
```

Once this is done, container1 will be able to start sub-containers.

In order to run unprivileged (the default in LXD) containers nested under an unprivileged container, you will need to ensure a wide enough UID mapping. Please see the 'UID mapping' section below.

Limits

LXD supports flexible constraints on the resources which containers can consume. The limits come in the following categories:

- CPU: limit cpu available to the container in several ways.
- Disk: configure the priority of I/O requests under load
- RAM: configure memory and swap availability

- Network: configure the network priority under load
- Processes: limit the number of concurrent processes in the container.

For a full list of limits known to LXD, see the configuration documentation.

UID mappings and Privileged containers

By default, LXD creates unprivileged containers. This means that root in the container is a non-root UID on the host. It is privileged against the resources owned by the container, but unprivileged with respect to the host, making root in a container roughly equivalent to an unprivileged user on the host. (The main exception is the increased attack surface exposed through the system call interface)

Briefly, in an unprivileged container, 65536 UIDs are 'shifted' into the container. For instance, UID 0 in the container may be 100000 on the host, UID 1 in the container is 100001, etc, up to 165535. The starting value for UIDs and GIDs, respectively, is determined by the 'root' entry the /etc/subuid and /etc/subgid files. (See the subuid(5) man page.)

It is possible to request a container to run without a UID mapping by setting the security.privileged flag to true:

```
lxc config set c1 security.privileged true
```

Note however that in this case the root user in the container is the root user on the host.

Apparmor

LXD confines containers by default with an apparmor profile which protects containers from each other and the host from containers. For instance this will prevent root in one container from signaling root in another container, even though they have the same uid mapping. It also prevents writing to dangerous, un-namespaced files such as many sysctls and /proc/sysrq−trigger.

If the apparmor policy for a container needs to be modified for a container c1, specific apparmor policy lines can be added in the raw.apparmor configuration key.

Seccomp

All containers are confined by a default seccomp policy. This policy prevents some dangerous actions such as forced umounts, kernel module loading and unloading, kexec, and the open_by_handle_at system call. The seccomp configuration cannot be modified, however a completely different seccomp policy – or none – can be requested using raw.lxc (see below).

Raw LXC configuration

LXD configures containers for the best balance of host safety and container usability. Whenever possible it is highly recommended to use the defaults, and use the LXD configuration keys to request LXD to modify as needed. Sometimes, however, it may be necessary to talk to the underlying lxc driver itself. This can be done by specifying LXC configuration items in the 'raw.lxc' LXD configuration key. These must be valid items as documented in the lxc.container.conf(5) manual page.

Snapshots

Containers can be renamed and live-migrated using the lxc move command:

```
lxc move c1 final-beta
```

They can also be snapshotted:

```
lxc snapshot c1 YYYY-MM-DD
```

Later changes to c1 can then be reverted by restoring the snapshot:

```
lxc restore u1 YYYY-MM-DD
```

New containers can also be created by copying a container or snapshot:

```
lxc copy u1/YYYY-MM-DD testcontainer
```

Publishing images

When a container or container snapshot is ready for consumption by others, it can be published as a new image using;

```
lxc publish u1/YYYY-MM-DD --alias foo-2.0
```

The published image will be private by default, meaning that LXD will not allow clients without a trusted certificate to see them. If the image is safe for public viewing (i.e. contains no private information), then the 'public' flag can be set, either at publish time using

```
lxc publish u1/YYYY-MM-DD --alias foo-2.0 public=true
```

or after the fact using

```
lxc image edit foo-2.0
```

and changing the value of the public field.

Image export and import

Image can be exported as, and imported from, tarballs:

```
lxc image export foo-2.0 foo-2.0.tar.gz
lxc image import foo-2.0.tar.gz --alias foo-2.0 --public
```

Troubleshooting

To view debug information about LXD itself, on a systemd based host use

```
journalctl -u lxd
```

Container logfiles for container c1 may be seen using:

```
lxc info c1 --show-log
```

The configuration file which was used may be found under /var/log/lxd/c1/lxc.conf while apparmor profiles can be found in /var/lib/lxd/security/apparmor/profiles/c1 and seccomp profiles in /var/lib/lxd/security/seccomp/c1.

LXC

Containers are a lightweight virtualization technology. They are more akin to an enhanced chroot than to full virtualization like Qemu or VMware, both because they do not emulate hardware and because containers share the same operating system as the host. Containers are similar to Solaris zones or BSD jails. Linux-vserver and OpenVZ are two pre-existing, independently developed implementations of containers-like functionality for Linux. In fact, containers came about as a result of the work to upstream the vserver and OpenVZ functionality.

There are two user-space implementations of containers, each exploiting the same kernel features. Libvirt allows the use of containers through the LXC driver by connecting to lxc:///. This can be very convenient as it supports the same usage as its other drivers. The other implementation, called simply 'LXC', is not compatible with libvirt, but is more flexible with more userspace tools. It is possible to switch between the two, though there are peculiarities which can cause confusion.

In this document we will mainly describe the lxc package. Use of libvirt-lxc is not generally recommended due to a lack of Apparmor protection for libvirt-lxc containers.

In this document, a container name will be shown as CN, C1, or C2.

Installation

The lxc package can be installed using

```
sudo apt install lxc
```

This will pull in the required and recommended dependencies, as well as set up a network bridge for containers to use. If you wish to use unprivileged containers, you will need to ensure that users have sufficient allocated subuids and subgids, and will likely want to allow users to connect containers to a bridge (see *Basic unprivileged usage* below).

Basic usage

LXC can be used in two distinct ways - privileged, by running the lxc commands as the root user; or unprivileged, by running the lxc commands as a non-root user. (The starting of unprivileged containers by the root user is possible, but not described here.) Unprivileged containers are more limited, for instance being unable to create device nodes or mount block-backed filesystems. However they are less dangerous to the host, as the root UID in the container is mapped to a non-root UID on the host.

Basic privileged usage

To create a privileged container, you can simply do:

```
sudo lxc-create --template download --name u1
```

or, abbreviated

```
sudo lxc-create -t download -n u1
```

This will interactively ask for a container root filesystem type to download – in particular the distribution, release, and architecture. To create the container non-interactively, you can specify these values on the command line:

```
sudo lxc-create -t download -n u1 -- --dist ubuntu --release DISTRO_SHORT_
    CODENAME --arch amd64
```

or

```
sudo lxc-create -t download -n u1 -- -d ubuntu -r DISTRO-SHORT-CODENAME -a
    amd64
```

You can now use lxc-ls to list containers, lxc-info to obtain detailed container information, lxc-start to start and lxc-stop to stop the container. lxc-attach and lxc-console allow you to enter a container, if ssh is not an option. lxc-destroy removes the container, including its rootfs. See the manual pages for more information on each command. An example session might look like:

```
sudo lxc-ls --fancy
sudo lxc-start --name u1 --daemon
sudo lxc-info --name u1
sudo lxc-stop --name u1
sudo lxc-destroy --name u1
```

User namespaces

Unprivileged containers allow users to create and administer containers without having any root privilege. The feature underpinning this is called user namespaces. User namespaces are hierarchical, with privileged tasks in a parent namespace being able to map its ids into child namespaces. By default every task on the host runs in the initial user namespace, where the full range of ids is mapped onto the full range. This can be seen by looking at /proc/self/uid_map and /proc/self/gid_map, which both will show 0 0 4294967295 when read from the initial user namespace. As of Ubuntu 14.04, when new users are created they are by default offered a range of UIDs. The list of assigned ids can be seen in the files /etc/subuid and /etc/subgid See their respective manpages for more information. Subuids and subgids are by convention started at id 100000 to avoid conflicting with system users.

If a user was created on an earlier release, it can be granted a range of ids using usermod, as follows:

```
sudo usermod -v 100000-200000 -w 100000-200000 user1
```

The programs newuidmap and newgidmap are setuid-root programs in the uidmap package, which are used internally by lxc to map subuids and subgids from the host into the unprivileged container. They ensure that the user only maps ids which are authorized by the host configuration.

Basic unprivileged usage

To create unprivileged containers, a few first steps are needed. You will need to create a default container configuration file, specifying your desired id mappings and network setup, as well as configure the host to allow the unprivileged user to hook into the host network. The example below assumes that your mapped user and group id ranges are 100000–165536. Check your actual user and group id ranges and modify the example accordingly:

```
grep $USER /etc/subuid
grep $USER /etc/subgid

mkdir -p ~/.config/lxc
echo "lxc.id_map = u 0 100000 65536" > ~/.config/lxc/default.conf
echo "lxc.id_map = g 0 100000 65536" >> ~/.config/lxc/default.conf
echo "lxc.network.type = veth" >> ~/.config/lxc/default.conf
echo "lxc.network.link = lxcbr0" >> ~/.config/lxc/default.conf
echo "$USER veth lxcbr0 2" | sudo tee -a /etc/lxc/lxc-usernet
```

After this, you can create unprivileged containers the same way as privileged ones, simply without using sudo.

```
lxc-create -t download -n u1 -- -d ubuntu -r DISTRO_SHORT_CODENAME -a amd64
lxc-start -n u1 -d
lxc-attach -n u1
lxc-stop -n u1
lxc-destroy -n u1
```

Nesting

In order to run containers inside containers - referred to as nested containers - two lines must be present in the parent container configuration file:

```
lxc.mount.auto = cgroup
lxc.aa_profile = lxc-container-default-with-nesting
```

The first will cause the cgroup manager socket to be bound into the container, so that lxc inside the container is able to administer cgroups for its nested containers. The second causes the container to run in a looser Apparmor policy which allows the container to do the mounting required for starting containers. Note that this policy, when used with a privileged container, is much less safe than the regular policy or an unprivileged container. See the *Apparmor* section for more information.

Global configuration

The following configuration files are consulted by LXC. For privileged use, they are found under /etc/lxc, while for unprivileged use they are under ~/.config/lxc.

- lxc.conf may optionally specify alternate values for several lxc settings, including the lxcpath, the default configuration, cgroups to use, a cgroup creation pattern, and storage backend settings for lvm and zfs.
- default.conf specifies configuration which every newly created container should contain. This usually contains at least a network section, and, for unprivileged users, an id mapping section
- lxc-usernet.conf specifies how unprivileged users may connect their containers to the host-owned network.

lxc.conf and default.conf are both under /etc/lxc and $HOME/.config/lxc, while lxc-usernet.conf is only host-wide.

By default, containers are located under /var/lib/lxc for the root user.

Networking

By default LXC creates a private network namespace for each container, which includes a layer 2 networking stack. Containers usually connect to the outside world by either having a physical NIC or a veth tunnel endpoint passed into the container. LXC creates a NATed bridge, lxcbr0, at host startup. Containers created using the default configuration will have one veth NIC with the remote end plugged into the lxcbr0 bridge. A NIC can only exist in one namespace at a time, so a physical NIC passed into the container is not usable on the host.

It is possible to create a container without a private network namespace. In this case, the container will have access to the host networking like any other application. Note that this is particularly dangerous if the container is running a distribution with upstart, like Ubuntu, since programs which talk to init, like shutdown, will talk over the abstract Unix domain socket to the host's upstart, and shut down the host.

To give containers on lxcbr0 a persistent ip address based on domain name, you can write entries to /etc/lxc/dnsmasq.conf like:

```
dhcp-host=lxcmail,10.0.3.100
dhcp-host=ttrss,10.0.3.101
```

If it is desirable for the container to be publicly accessible, there are a few ways to go about it. One is to use iptables to forward host ports to the container, for instance

```
iptables -t nat -A PREROUTING -p tcp -i eth0 --dport 587 -j DNAT \
    --to-destination 10.0.3.100:587
```

Then, specify the host's bridge in the container configuration file in place of lxcbr0, for instance

```
lxc.network.type = veth
lxc.network.link = br0
```

Finally, you can ask LXC to use macvlan for the container's NIC. Note that this has limitations and depending on configuration may not allow the container to talk to the host itself. Therefore the other two options are preferred and more commonly used.

There are several ways to determine the ip address for a container. First, you can use lxc-ls --fancy which will print the ip addresses for all running containers, or lxc-info -i -H -n C1 which will print C1's ip address. If dnsmasq is installed on the host, you can also add an entry to /etc/dnsmasq.conf as follows

```
server=/lxc/10.0.3.1
```

after which dnsmasq will resolve C1.lxc locally, so that you can do:

```
ping C1
ssh C1
```

For more information, see the lxc.conf(5) manpage as well as the example network configurations under /usr/share/doc/lxc/examples/.

LXC startup

LXC does not have a long-running daemon. However it does have three upstart jobs.

- /etc/init/lxc-net.conf: is an optional job which only runs if /etc/default/lxc-net specifies USE_LXC_BRIDGE (true by default). It sets up a NATed bridge for containers to use.
- /etc/init/lxc.conf loads the lxc apparmor profiles and optionally starts any autostart containers. The autostart containers will be ignored if LXC_AUTO (true by default) is set to true in /etc/default/lxc. See the lxc-autostart manual page for more information on autostarted containers.
- /etc/init/lxc-instance.conf is used by /etc/init/lxc.conf to autostart a container.

Backing Stores

LXC supports several backing stores for container root filesystems. The default is a simple directory backing store, because it requires no prior host customization, so long as the underlying filesystem is large enough. It also requires no root privilege to create the backing store, so that it is seamless for unprivileged use. The rootfs for a privileged directory backed container is located (by default) under /var/lib/lxc/C1/rootfs, while the rootfs for an unprivileged container is under ~/.local/share/lxc/C1/rootfs. If a custom lxcpath is specified in lxc.system.com, then the container rootfs will be under $lxcpath/C1/rootfs.

A snapshot clone C2 of a directory backed container C1 becomes an overlayfs backed container, with a rootfs called overlayfs:/var/lib/lxc/C1/rootfs:/var/lib/lxc/C2/delta0. Other backing store types include loop, btrfs, LVM and zfs.

A btrfs backed container mostly looks like a directory backed container, with its root filesystem in the same location. However, the root filesystem comprises a subvolume, so that a snapshot clone is created using a subvolume snapshot.

The root filesystem for an LVM backed container can be any separate LV. The default VG name can be specified in lxc.conf. The filesystem type and size are configurable per-container using lxc-create.

The rootfs for a zfs backed container is a separate zfs filesystem, mounted under the traditional /var/lib/lxc/C1/rootfs location. The zfsroot can be specified at lxc-create, and a default can be specified in lxc.system.conf.

More information on creating containers with the various backing stores can be found in the lxc-create manual page.

Templates

Creating a container generally involves creating a root filesystem for the container. lxc−create delegates this work to *templates*, which are generally per-distribution. The lxc templates shipped with lxc can be found under /usr/share/lxc/templates, and include templates to create Ubuntu, Debian, Fedora, Oracle, centos, and gentoo containers among others.

Creating distribution images in most cases requires the ability to create device nodes, often requires tools which are not available in other distributions, and usually is quite time-consuming. Therefore lxc comes with a special *download* template, which downloads pre-built container images from a central lxc server. The most important use case is to allow simple creation of unprivileged containers by non-root users, who could not for instance easily run the debootstrap command.

When running lxc−create, all options which come after – are passed to the template. In the following command, *–name*, *–template* and *–bdev* are passed to lxc−create, while *–release* is passed to the template:

```
lxc-create --template ubuntu --name c1 --bdev loop -- --release DISTRO_SHORT_CODENAME
```

You can obtain help for the options supported by any particular container by passing *–help* and the template name to lxc−create. For instance, for help with the download template,

```
lxc-create --template download --help
```

Autostart

LXC supports marking containers to be started at system boot. Prior to Ubuntu 14.04, this was done using symbolic links under the directory /etc/lxc/auto. Starting with Ubuntu 14.04, it is done through the container configuration files. An entry

```
lxc.start.auto = 1
lxc.start.delay = 5
```

would mean that the container should be started at boot, and the system should wait 5 seconds before starting the next container. LXC also supports ordering and grouping of containers, as well as reboot and shutdown by autostart groups. See the manual pages for lxc-autostart and lxc.container.conf for more information.

Apparmor

LXC ships with a default Apparmor profile intended to protect the host from accidental misuses of privilege inside the container. For instance, the container will not be able to write to /proc/sysrq−trigger or to most /sys files.

The usr.bin.lxc−start profile is entered by running lxc−start. This profile mainly prevents lxc−start from mounting new filesystems outside of the container's root filesystem. Before executing the container's init, LXC requests a switch to the container's profile. By default, this profile is the lxc−container−default policy which is defined in /etc/apparmor.d/lxc/lxc−default. This profile prevents the container from accessing many dangerous paths, and from mounting most filesystems.

Programs in a container cannot be further confined - for instance, MySQL runs under the container profile (protecting the host) but will not be able to enter the MySQL profile (to protect the container).

lxc−execute does not enter an Apparmor profile, but the container it spawns will be confined.

Customizing container policies

If you find that lxc−start is failing due to a legitimate access which is being denied by its Apparmor policy, you can disable the lxc-start profile by doing:

```
sudo apparmor_parser -R /etc/apparmor.d/usr.bin.lxc-start
sudo ln -s /etc/apparmor.d/usr.bin.lxc-start /etc/apparmor.d/disabled/
```

This will make lxc−start run unconfined, but continue to confine the container itself. If you also wish to disable confinement of the container, then in addition to disabling the usr.bin.lxc−start profile, you must add:

```
lxc.aa_profile = unconfined
```

to the container's configuration file.

LXC ships with a few alternate policies for containers. If you wish to run containers inside containers (nesting), then you can use the lxc-container-default-with-nesting profile by adding the following line to the container configuration file

```
lxc.aa_profile = lxc-container-default-with-nesting
```

If you wish to use libvirt inside containers, then you will need to edit that policy (which is defined in /etc/apparmor.d/lxc/lxc−default−with−nesting) by uncommenting the following line:

```
mount fstype=cgroup -> /sys/fs/cgroup/**,
```

and re-load the policy.

Note that the nesting policy with privileged containers is far less safe than the default policy, as it allows containers to re-mount /sys and /proc in nonstandard locations, bypassing apparmor protections. Unprivileged containers do not have this drawback since the container root cannot write to root-owned proc and sys files.

Another profile shipped with lxc allows containers to mount block filesystem types like ext4. This can be useful in some cases like maas provisioning, but is deemed generally unsafe since the superblock handlers in the kernel have not been audited for safe handling of untrusted input.

If you need to run a container in a custom profile, you can create a new profile under /etc/apparmor.d/lxc/. Its name must start with lxc− in order for lxc−start to be allowed to transition to that profile. The lxc−default profile includes the re-usable abstractions file /etc/apparmor.d/abstractions/lxc/container−base.

An easy way to start a new profile therefore is to do the same, then add extra permissions at the bottom of your policy.

After creating the policy, load it using:

```
sudo apparmor_parser -r /etc/apparmor.d/lxc-containers
```

The profile will automatically be loaded after a reboot, because it is sourced by the file /etc/apparmor.d/lxc-containers. Finally, to make container CN use this new lxc-CN-profile, add the following line to its configuration file:

```
lxc.aa_profile = lxc-CN-profile
```

Control Groups

Control groups (cgroups) are a kernel feature providing hierarchical task grouping and per-cgroup resource accounting and limits. They are used in containers to limit block and character device access and to freeze (suspend) containers. They can be further used to limit memory use and block i/o, guarantee minimum cpu shares, and to lock containers to specific cpus.

By default, a privileged container CN will be assigned to a cgroup called /lxc/CN. In the case of name conflicts (which can occur when using custom lxcpaths) a suffix "-n", where n is an integer starting at 0, will be appended to the cgroup name.

By default, a privileged container CN will be assigned to a cgroup called CN under the cgroup of the task which started the container, for instance /usr/1000.user/1.session/CN. The container root will be given group ownership of the directory (but not all files) so that it is allowed to create new child cgroups.

As of Ubuntu 14.04, LXC uses the cgroup manager (cgmanager) to administer cgroups. The cgroup manager receives D-Bus requests over the Unix socket /sys/fs/cgroup/cgmanager/sock. To facilitate safe nested containers, the line

```
lxc.mount.auto = cgroup
```

can be added to the container configuration causing the /sys/fs/cgroup/cgmanager directory to be bind-mounted into the container. The container in turn should start the cgroup management proxy (done by default if the cgmanager package is installed in the container) which will move the /sys/fs/cgroup/cgmanager directory to /sys/fs/cgroup/cgmanager.lower, then start listening for requests to proxy on its own socket /sys/fs/cgroup/cgmanager/sock. The host cgmanager will ensure that nested containers cannot escape their assigned cgroups or make requests for which they are not authorized.

Cloning

For rapid provisioning, you may wish to customize a canonical container according to your needs and then make multiple copies of it. This can be done with the lxc-clone program.

Clones are either snapshots or copies of another container. A copy is a new container copied from the original, and takes as much space on the host as the original. A snapshot exploits the underlying backing store's snapshotting ability to make a copy-on-write container referencing the first. Snapshots can be created from btrfs, LVM, zfs, and directory backed containers. Each backing store has its own peculiarities - for instance, LVM containers which are not thinpool-provisioned cannot support snapshots of snapshots; zfs containers with snapshots cannot be removed until all snapshots are released; LVM containers must be more carefully planned as the underlying filesystem may not support growing; btrfs does not suffer any of these shortcomings, but suffers from reduced fsync performance causing dpkg and apt to be slower.

Snapshots of directory-packed containers are created using the overlay filesystem. For instance, a privileged directory-backed container C1 will have its root filesystem under /var/lib/lxc/C1/rootfs. A snapshot clone of

C1 called C2 will be started with C1's rootfs mounted readonly under /var/lib/lxc/C2/delta0. Importantly, in this case C1 should not be allowed to run or be removed while C2 is running. It is advised instead to consider C1 a *canonical* base container, and to only use its snapshots.

Given an existing container called C1, a copy can be created using:

```
sudo lxc-clone -o C1 -n C2
```

A snapshot can be created using:

```
sudo lxc-clone -s -o C1 -n C2
```

See the lxc-clone manpage for more information.

Snapshots

To more easily support the use of snapshot clones for iterative container development, LXC supports *snapshots*. When working on a container C1, before making a potentially dangerous or hard-to-revert change, you can create a snapshot

```
sudo lxc-snapshot -n C1
```

which is a snapshot-clone called 'snap0' under /var/lib/lxcsnaps or $HOME/.local/share/lxcsnaps. The next snapshot will be called 'snap1', etc. Existing snapshots can be listed using lxc-snapshot -L -n C1, and a snapshot can be restored - erasing the current C1 container - using lxc-snapshot -r snap1 -n C1. After the restore command, the snap1 snapshot continues to exist, and the previous C1 is erased and replaced with the snap1 snapshot.

Snapshots are supported for btrfs, lvm, zfs, and overlayfs containers. If lxc-snapshot is called on a directory-backed container, an error will be logged and the snapshot will be created as a copy-clone. The reason for this is that if the user creates an overlayfs snapshot of a directory-backed container and then makes changes to the directory-backed container, then the original container changes will be partially reflected in the snapshot. If snapshots of a directory backed container C1 are desired, then an overlayfs clone of C1 should be created, C1 should not be touched again, and the overlayfs clone can be edited and snapshotted at will, as such

```
lxc-clone -s -o C1 -n C2
lxc-start -n C2 -d # make some changes
lxc-stop -n C2
lxc-snapshot -n C2
lxc-start -n C2 # etc
```

Ephemeral Containers

While snapshots are useful for longer-term incremental development of images, ephemeral containers utilize snapshots for quick, single-use throwaway containers. Given a base container C1, you can start an ephemeral container using

```
lxc-start-ephemeral -o C1
```

The container begins as a snapshot of C1. Instructions for logging into the container will be printed to the console. After shutdown, the ephemeral container will be destroyed. See the lxc-start-ephemeral manual page for more options.

Lifecycle management hooks

Beginning with Ubuntu 12.10, it is possible to define hooks to be executed at specific points in a container's lifetime:

- Pre-start hooks are run in the host's namespace before the container ttys, consoles, or mounts are up. If any mounts are done in this hook, they should be cleaned up in the post-stop hook.
- Pre-mount hooks are run in the container's namespaces, but before the root filesystem has been mounted. Mounts done in this hook will be automatically cleaned up when the container shuts down.
- Mount hooks are run after the container filesystems have been mounted, but before the container has called pivot_root to change its root filesystem.
- Start hooks are run immediately before executing the container's init. Since these are executed after pivoting into the container's filesystem, the command to be executed must be copied into the container's filesystem.
- Post-stop hooks are executed after the container has been shut down.

If any hook returns an error, the container's run will be aborted. Any *post-stop* hook will still be executed. Any output generated by the script will be logged at the debug priority.

Please see the lxc.container.conf(5) manual page for the configuration file format with which to specify hooks. Some sample hooks are shipped with the lxc package to serve as an example of how to write and use such hooks.

Consoles

Containers have a configurable number of consoles. One always exists on the container's /dev/console. This is shown on the terminal from which you ran lxc−start, unless the *-d* option is specified. The output on /dev/console can be redirected to a file using the *-c console-file* option to lxc−start. The number of extra consoles is specified by the lxc.tty variable, and is usually set to 4. Those consoles are shown on /dev/ttyN (for 1 <= N <= 4). To log into console 3 from the host, use:

```
sudo lxc−console −n container −t 3
```

or if the −t N option is not specified, an unused console will be automatically chosen. To exit the console, use the escape sequence Ctrl−a q. Note that the escape sequence does not work in the console resulting from lxc−start without the −d option.

Each container console is actually a Unix98 pty in the host's (not the guest's) pty mount, bind-mounted over the guest's /dev/ttyN and /dev/console. Therefore, if the guest unmounts those or otherwise tries to access the actual character device 4:N, it will not be serving getty to the LXC consoles. (With the default settings, the container will not be able to access that character device and getty will therefore fail.) This can easily happen when a boot script blindly mounts a new /dev.

Troubleshooting

Logging

If something goes wrong when starting a container, the first step should be to get full logging from LXC:

```
sudo lxc−start −n C1 −l trace −o debug.out
```

This will cause lxc to log at the most verbose level, trace, and to output log information to a file called 'debug.out'. If the file debug.out already exists, the new log information will be appended.

Monitoring container status

Two commands are available to monitor container state changes. lxc−monitor monitors one or more containers for any state changes. It takes a container name as usual with the -*n* option, but in this case the container name can be a posix regular expression to allow monitoring desirable sets of containers. lxc−monitor continues running as it prints container changes. lxc−wait waits for a specific state change and then exits. For instance,

```
sudo lxc-monitor -n cont[0-5]*
```

would print all state changes to any containers matching the listed regular expression, whereas

```
sudo lxc-wait -n cont1 -s 'STOPPED|FROZEN'
```

will wait until container cont1 enters state STOPPED or state FROZEN and then exit.

Attach

As of Ubuntu 14.04, it is possible to attach to a container's namespaces. The simplest case is to simply do

```
sudo lxc-attach -n C1
```

which will start a shell attached to C1's namespaces, or, effectively inside the container. The attach functionality is very flexible, allowing attaching to a subset of the container's namespaces and security context. See the manual page for more information.

Container init verbosity

If LXC completes the container startup, but the container init fails to complete (for instance, no login prompt is shown), it can be useful to request additional verbosity from the init process. For an upstart container, this might be:

```
sudo lxc-start -n C1 /sbin/init loglevel=debug
```

You can also start an entirely different program in place of init, for instance

```
sudo lxc-start -n C1 /bin/bash
sudo lxc-start -n C1 /bin/sleep 100
sudo lxc-start -n C1 /bin/cat /proc/1/status
```

LXC API

Most of the LXC functionality can now be accessed through an API exported by liblxc for which bindings are available in several languages, including Python, lua, ruby, and go.

Below is an example using the python bindings (which are available in the python3-lxc package) which creates and starts a container, then waits until it has been shut down:

```
# sudo python3
Python 3.2.3 (default, Aug 28 2012, 08:26:03)
[GCC 4.7.1 20120814 (prerelease)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import lxc
__main__:1: Warning: The python-lxc API isn't yet stable and may change at any
    point in the future.
```

```
>>> c=lxc.Container("C1")
>>> c.create("ubuntu")
True
>>> c.start()
True
>>> c.wait("STOPPED")
True
```

Security

A namespace maps ids to resources. By not providing a container any id with which to reference a resource, the resource can be protected. This is the basis of some of the security afforded to container users. For instance, IPC namespaces are completely isolated. Other namespaces, however, have various *leaks* which allow privilege to be inappropriately exerted from a container into another container or to the host.

By default, LXC containers are started under a Apparmor policy to restrict some actions. The details of AppArmor integration with lxc are in section *Apparmor*. Unprivileged containers go further by mapping root in the container to an unprivileged host UID. This prevents access to /proc and /sys files representing host resources, as well as any other files owned by root on the host.

Exploitable system calls

It is a core container feature that containers share a kernel with the host. Therefore if the kernel contains any exploitable system calls the container can exploit these as well. Once the container controls the kernel it can fully control any resource known to the host.

In general to run a full distribution container a large number of system calls will be needed. However for application containers it may be possible to reduce the number of available system calls to only a few. Even for system containers running a full distribution security gains may be had, for instance by removing the 32-bit compatibility system calls in a 64-bit container. See the lxc.container.conf manual page for details of how to configure a container to use seccomp. By default, no seccomp policy is loaded.

Resources

- The DeveloperWorks article LXC: Linux container tools was an early introduction to the use of containers.
- The Secure Containers Cookbook demonstrated the use of security modules to make containers more secure.
- The upstream LXC project is hosted at linuxcontainers.org.

Databases

Ubuntu provides two popular database servers. They are:

- MySQL
- PostgreSQL

Both are popular choices among developers, with similar feature sets and performance capabilities. Historically, Postgres tended to be a preferred choice for its attention to standards conformance, features, and extensibility, whereas Mysql may be more preferred for higher performance requirements, however over time each has made good strides catching up with the other. Specialized needs may make one a better option for a certain application, but in general both are good, strong options.

They are available in the main repository and equally supported by Ubuntu. This section explains how to install and configure these database servers.

MySQL

MySQL is a fast, multi-threaded, multi-user, and robust SQL database server. It is intended for mission-critical, heavy-load production systems and mass-deployed software.

Installation

To install MySQL, run the following command from a terminal prompt:

```
sudo apt install mysql-server
```

Once the installation is complete, the MySQL server should be started automatically. You can quickly check its current status via systemd:

```
sudo service mysql status
```

```
 mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:
       enabled)
   Active: active (running) since Tue 2019-10-08 14:37:38 PDT; 2 weeks 5 days
       ago
 Main PID: 2028 (mysqld)
    Tasks: 28 (limit: 4915)
   CGroup: /system.slice/mysql.service
           2028 /usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid

Oct 08 14:37:36 db.example.org systemd[1]: Starting MySQL Community Server...
Oct 08 14:37:38 db.example.org systemd[1]: Started MySQL Community Server.
```

The network status of the MySQL service can also be checked by running the ss command at the terminal prompt:

```
sudo ss -tap | grep mysql
```

When you run this command, you should see something similar to the following:

```
LISTEN     0           151                 127.0.0.1:mysql                0.0.0.0:*
           users:(("mysqld",pid=149190,fd=29))
LISTEN     0           70                        *:33060                      *:*
           users:(("mysqld",pid=149190,fd=32))
```

If the server is not running correctly, you can type the following command to start it:

```
sudo service mysql restart
```

A good starting point for troubleshooting problems is the systemd journal, which can be accessed at the terminal prompt with this command:

```
sudo journalctl -u mysql
```

Configuration

You can edit the files in /etc/mysql/ to configure the basic settings – log file, port number, etc. For example, to configure MySQL to listen for connections from network hosts, in the file /etc/mysql/mysql.conf.d/mysqld .cnf, change the *bind-address* directive to the server's IP address:

```
bind-address            = 192.168.0.5
```

> **Note**
>
> Replace 192.168.0.5 with the appropriate address, which can be determined via ip address show.

After making a configuration change, the MySQL daemon will need to be restarted:

```
sudo systemctl restart mysql.service
```

Database Engines

Whilst the default configuration of MySQL provided by the Ubuntu packages is perfectly functional and performs well there are things you may wish to consider before you proceed.

MySQL is designed to allow data to be stored in different ways. These methods are referred to as either database or storage engines. There are two main engines that you'll be interested in: InnoDB and MyISAM. Storage engines are transparent to the end user. MySQL will handle things differently under the surface, but regardless of which storage engine is in use, you will interact with the database in the same way.

Each engine has its own advantages and disadvantages.

While it is possible, and may be advantageous to mix and match database engines on a table level, doing so reduces the effectiveness of the performance tuning you can do as you'll be splitting the resources between two engines instead of dedicating them to one.

- MyISAM is the older of the two. It can be faster than InnoDB under certain circumstances and favours a read only workload. Some web applications have been tuned around MyISAM (though that's not to imply that they will slow under InnoDB). MyISAM also supports the FULLTEXT data type, which allows very fast searches of large quantities of text data. However MyISAM is only capable of locking an entire table for writing. This means only one process can update a table at a time. As any application that uses the table scales this may prove to be a hindrance. It also lacks journaling, which makes it harder for data to be recovered after a crash. The following link provides some points for consideration about using MyISAM on a production database.
- InnoDB is a more modern database engine, designed to be ACID compliant which guarantees database transactions are processed reliably. Write locking can occur on a row level basis within a table. That means multiple updates can occur on a single table simultaneously. Data caching is also handled in memory within the database engine, allowing caching on a more efficient row level basis rather than file block. To meet ACID compliance all transactions are journaled independently of the main tables. This allows for much more reliable data recovery as data consistency can be checked.

As of MySQL 5.5 InnoDB is the default engine, and is highly recommended over MyISAM unless you have specific need for features unique to the engine.

Advanced configuration

Creating a tuned configuration

There are a number of parameters that can be adjusted within MySQL's configuration files that will allow you to improve the performance of the server over time.

Many of the parameters can be adjusted with the existing database, however some may affect the data layout and thus need more care to apply.

First, if you have existing data, you will need to carry out a mysqldump and reload:

```
mysqldump --all-databases --routines -u root -p > ~/fulldump.sql
```

This will then prompt you for the root password before creating a copy of the data. It is advisable to make sure there are no other users or processes using the database whilst this takes place. Depending on how much data you've got in your database, this may take a while. You won't see anything on the screen during this process.

Once the dump has been completed, shut down MySQL:

```
sudo service mysql stop
```

It's also a good idea to backup the original configuration:

```
sudo rsync -avz /etc/mysql /root/mysql-backup
```

Next, make any desired configuration changes.

Then delete and re-initialise the database space and make sure ownership is correct before restarting MySQL:

```
sudo rm -rf /var/lib/mysql/*
sudo mysqld --initialize
sudo chown -R mysql: /var/lib/mysql
sudo service mysql start
```

The final step is re-importation of your data by piping your SQL commands to the database.

```
cat ~/fulldump.sql | mysql
```

For large data imports, the 'Pipe Viewer' utility can be useful to track import progress. Ignore any ETA times produced by pv, they're based on the average time taken to handle each row of the file, but the speed of inserting can vary wildly from row to row with mysqldumps:

```
sudo apt install pv
pv ~/fulldump.sql | mysql
```

Once that is complete all is good to go!

> **Note**
>
> This is not necessary for all my.cnf changes. Most of the variables you may wish to change to improve performance are adjustable even whilst the server is running. As with anything, make sure to have a good backup copy of config files and data before making changes.

MySQL Tuner

MySQL Tuner connects to a running MySQL instance and offer configuration suggestions to optimize the database for your workload. The longer the server has been running, the better the advice mysqltuner can provide. In a production environment, consider waiting for at least 24 hours before running the tool. You can install mysqltuner from the Ubuntu repositories:

```
sudo apt install mysqltuner
```

Then once its been installed, run:

```
mysqltuner
```

and wait for its final report. The top section provides general information about the database server, and the bottom section provides tuning suggestions to alter in your my.cnf. Most of these can be altered live on the server without restarting; look through the official MySQL documentation (link in Resources section) for the relevant variables to change in production. The following example is part of a report from a production database showing potential benefits from increasing the query cache:

```
———— Recommendations ————————————————————————————————————————————————
General recommendations:
    Run OPTIMIZE TABLE to defragment tables for better performance
    Increase table_cache gradually to avoid file descriptor limits
Variables to adjust:
    key_buffer_size (> 1.4G)
    query_cache_size (> 32M)
    table_cache (> 64)
    innodb_buffer_pool_size (>= 22G)
```

It goes without saying that performance optimization strategies vary from application to application. So for example, what works best for Wordpress might not be the best for Drupal or Joomla. Performance can be dependent on the types of queries, use of indexes, how efficient the database design is and so on. You may find it useful to spend some time searching for database tuning tips based on what applications you're using. Once you've reached the point of diminishing returns from database configuration adjustments, look to the application itself for improvements, or invest in more powerful hardware and/or scaling up the database environment.

Resources

- See the MySQL Home Page for more information.
- Full documentation is available in both online and offline formats from the MySQL Developers portal
- For general SQL information see the O'Reilly books Getting Started with SQL: A Hands-On Approach for Beginners by Thomas Nield as an entry point and SQL in a Nutshell as a quick reference.
- The Apache MySQL PHP Ubuntu Wiki page also has useful information.

PostgreSQL

PostgreSQL is an object-relational database system that has the features of traditional commercial database systems with enhancements to be found in next-generation DBMS systems.

Installation

To install PostgreSQL, run the following command in the command prompt:

```
sudo apt install postgresql
```

The database service is automatically configured with viable defaults, but can be customized based on your specialized needs.

Configuration

PostgreSQL supports multiple client authentication methods. By default, the *IDENT* authentication method is used for postgres and local users. Please refer to the PostgreSQL Administrator's Guide if you would like to configure alternatives like Kerberos.

The following discussion assumes that you wish to enable TCP/IP connections and use the MD5 method for client authentication. PostgreSQL configuration files are stored in the /etc/postgresql/<version>/main directory. For example, if you install PostgreSQL 12, the configuration files are stored in the /etc/postgresql /12/main directory.

> **Tip**
>
> To configure *IDENT* authentication, add entries to the /etc/postgresql/12/main/pg_ident.conf file. There are detailed comments in the file to guide you.

To enable other computers to connect to your PostgreSQL server, edit the file /etc/postgresql/12/main/ postgresql.conf

Locate the line *#listen_addresses = 'localhost'* and change it to:

```
listen_addresses = '*'
```

> **Note**
>
> To allow both IPv4 and IPv6 connections replace 'localhost' with '::'

For details on other parameters, refer to the configuration file or to the PostgreSQL documentation for information on how they can be edited.

Now that we can connect to our PostgreSQL server, the next step is to set a password for the *postgres* user. Run the following command at a terminal prompt to connect to the default PostgreSQL template database:

```
sudo -u postgres psql template1
```

The above command connects to PostgreSQL database *template1* as user *postgres*. Once you connect to the PostgreSQL server, you will be at a SQL prompt. You can run the following SQL command at the psql prompt to configure the password for the user *postgres*.

```
ALTER USER postgres with encrypted password 'your_password';
```

After configuring the password, edit the file /etc/postgresql/12/main/pg_hba.conf to use *MD5* authentication with the *postgres* user:

```
local   all             postgres                                md5
```

Finally, you should restart the PostgreSQL service to initialize the new configuration. From a terminal prompt enter the following to restart PostgreSQL:

```
sudo systemctl restart postgresql.service
```

> **Warning**
>
> The above configuration is not complete by any means. Please refer to the PostgreSQL Administrator's Guide to configure more parameters.

You can test server connections from other machines by using the PostgreSQL client.

```
sudo apt install postgresql-client
psql -h postgres.example.com -U postgres -W
```

> **Note**
>
> Replace the domain name with your actual server domain name.

Backups

PostgreSQL databases should be backed up regularly. Refer to the PostgreSQL Administrator's Guide for different approaches.

Resources

- As mentioned above the PostgreSQL Administrator's Guide is an excellent resource. The guide is also available in the postgresql-doc-12 package. Execute the following in a terminal to install the package:

  ```
  sudo apt install postgresql-doc-12
  ```

  To view the guide enter file :///usr/share/doc/postgresql-doc-12/html/index.html into the address bar of your browser.
- For general SQL information see the O'Reilly books Getting Started with SQL: A Hands-On Approach for Beginners by Thomas Nield as an entry point and SQL in a Nutshell as a quick reference.
- Also, see the PostgreSQL Ubuntu Wiki page for more information.

Active Directory Integration

Accessing a Samba Share

Another, use for Samba is to integrate into an existing Windows network. Once part of an Active Directory domain, Samba can provide file and print services to AD users. For details on how to join a domain, see the SSSD and Active Directory chapter of this guide.

Once part of the Active Directory domain, enter the following command in the terminal prompt:

```
sudo apt install samba cifs-utils smbclient
```

Next, edit /etc/samba/smb.conf changing:

```
workgroup = EXAMPLE
...
security = ads
realm = EXAMPLE.COM
...
idmap backend = lwopen
idmap uid = 50-9999999999
idmap gid = 50-9999999999
```

Restart samba for the new settings to take effect:

```
sudo systemctl restart smbd.service nmbd.service
```

You should now be able to access any Samba shares from a Windows client. However, be sure to give the appropriate AD users or groups access to the share directory. See Securing File and Print Server for more details.

Accessing a Windows Share

Now that the Samba server is part of the Active Directory domain you can access any Windows server shares:

- To mount a Windows file share enter the following in a terminal prompt:

  ```
  mount.cifs //fs01.example.com/share mount_point
  ```

  It is also possible to access shares on computers not part of an AD domain, but a username and password will need to be provided.

- To mount the share during boot place an entry in /etc/fstab, for example:

  ```
  //192.168.0.5/share /mnt/windows cifs auto,username=steve,password=secret,
      rw 0          0
  ```

- Another way to copy files from a Windows server is to use the smbclient utility. To list the files in a Windows share:

  ```
  smbclient //fs01.example.com/share -k -c "ls"
  ```

- To copy a file from the share, enter:

  ```
  smbclient //fs01.example.com/share -k -c "get file.txt"
  ```

  This will copy the file .txt into the current directory.

- And to copy a file to the share:

  ```
  smbclient //fs01.example.com/share -k -c "put /etc/hosts hosts"
  ```

  This will copy the /etc/hosts to //fs01.example.com/share/hosts.

- The *-c* option used above allows you to execute the smbclient command all at once. This is useful for scripting and minor file operations. To enter the *smb: \>* prompt, a FTP like prompt where you can execute normal file and directory commands, simply execute:

  ```
  smbclient //fs01.example.com/share -k
  ```

  Note

  Replace all instances of *fs01.example.com/share*, *//192.168.0.5/share*, *username=steve,password=secret*, and *file.txt* with your server's IP, hostname, share name, file name, and an actual username and password with rights to the share.

Resources

For more smbclient options see the man page: man smbclient, also available online.

The mount.cifs man page is also useful for more detailed information.

The Ubuntu Wiki Samba page.

As a Domain Controller

A Samba server can be configured to appear as a Windows NT4-style domain controller. A major advantage of this configuration is the ability to centralize user and machine credentials. Samba can also use multiple backends to store the user information.

Primary Domain Controller

This section covers configuring Samba as a Primary Domain Controller (PDC) using the default smbpasswd backend.

First, install Samba, and libpam-winbind to sync the user accounts, by entering the following in a terminal prompt:

```
sudo apt install samba libpam-winbind
```

Next, configure Samba by editing /etc/samba/smb.conf. The *security* mode should be set to *user*, and the *workgroup* should relate to your organization:

```
workgroup = EXAMPLE
...
security = user
```

In the commented "Domains" section add or uncomment the following (the last line has been split to fit the format of this document):

```
domain logons = yes
logon path = \\%N\%U\profile
logon drive = H:
logon home = \\%N\%U
logon script = logon.cmd
add machine script = sudo /usr/sbin/useradd -N -g machines -c Machine -d
      /var/lib/samba -s /bin/false %u
```

Note

If you wish to not use *Roaming Profiles* leave the *logon home* and *logon path* options commented.

- *domain logons:* provides the netlogon service causing Samba to act as a domain controller.
- *logon path:* places the user's Windows profile into their home directory. It is also possible to configure a *[profiles]* share placing all profiles under a single directory.
- *logon drive:* specifies the home directory local path.
- *logon home:* specifies the home directory location.
- *logon script:* determines the script to be run locally once a user has logged in. The script needs to be placed in the *[netlogon]* share.
- *add machine script:* a script that will automatically create the *Machine Trust Account* needed for a workstation to join the domain.

  In this example the *machines* group will need to be created using the addgroup utility see ??? for details.

Uncomment the *[homes]* share to allow the *logon home* to be mapped:

```
[homes]
   comment = Home Directories
   browseable = no
   read only = no
   create mask = 0700
   directory mask = 0700
   valid users = %S
```

When configured as a domain controller a *[netlogon]* share needs to be configured. To enable the share, uncomment:

```
[netlogon]
   comment = Network Logon Service
   path = /srv/samba/netlogon
   guest ok = yes
   read only = yes
   share modes = no
```

Note

The original *netlogon* share path is /home/samba/netlogon, but according to the Filesystem Hierarchy Standard (FHS), /srv is the correct location for site-specific data provided by the system.

Now create the netlogon directory, and an empty (for now) logon.cmd script file:

```
sudo mkdir -p /srv/samba/netlogon
sudo touch /srv/samba/netlogon/logon.cmd
```

You can enter any normal Windows logon script commands in logon.cmd to customize the client's environment.

Restart Samba to enable the new domain controller:

```
sudo systemctl restart smbd.service nmbd.service
```

Lastly, there are a few additional commands needed to setup the appropriate rights.

With *root* being disabled by default, in order to join a workstation to the domain, a system group needs to be mapped to the Windows *Domain Admins* group. Using the net utility, from a terminal enter:

```
sudo net groupmap add ntgroup="Domain Admins" unixgroup=sysadmin rid=512 type=
    d
```

Note

Change *sysadmin* to whichever group you prefer. Also, the user used to join the domain needs to be a member of the *sysadmin* group, as well as a member of the system *admin* group. The *admin* group allows sudo use.

If the user does not have Samba credentials yet, you can add them with the smbpasswd utility, change the *sysadmin* username appropriately:

```
sudo smbpasswd -a sysadmin
```

Also, rights need to be explicitly provided to the *Domain Admins* group to allow the *add machine script* (and other admin functions) to work. This is achieved by executing:

```
net rpc rights grant -U sysadmin "EXAMPLE\Domain Admins"
    SeMachineAccountPrivilege \
SePrintOperatorPrivilege SeAddUsersPrivilege SeDiskOperatorPrivilege \
SeRemoteShutdownPrivilege
```

You should now be able to join Windows clients to the Domain in the same manner as joining them to an NT4 domain running on a Windows server.

Backup Domain Controller

With a Primary Domain Controller (PDC) on the network it is best to have a Backup Domain Controller (BDC) as well. This will allow clients to authenticate in case the PDC becomes unavailable.

When configuring Samba as a BDC you need a way to sync account information with the PDC. There are multiple ways of accomplishing this scp, rsync, or by using LDAP as the *passdb backend*.

Using LDAP is the most robust way to sync account information, because both domain controllers can use the same information in real time. However, setting up a LDAP server may be overly complicated for a small number of user and computer accounts. See ??? for details.

First, install samba and libpam-winbind. From a terminal enter:

```
sudo apt install samba libpam-winbind
```

Now, edit /etc/samba/smb.conf and uncomment the following in the *[global]*:

```
workgroup = EXAMPLE
...
security = user
```

In the commented *Domains* uncomment or add:

```
domain logons = yes
domain master = no
```

Make sure a user has rights to read the files in /var/lib/samba. For example, to allow users in the *admin* group to scp the files, enter:

```
sudo chgrp -R admin /var/lib/samba
```

Next, sync the user accounts, using scp to copy the /var/lib/samba directory from the PDC:

```
sudo scp -r username@pdc:/var/lib/samba /var/lib
```

> **Note**
>
> Replace *username* with a valid username and *pdc* with the hostname or IP Address of your actual PDC.

Finally, restart samba:

```
sudo systemctl restart smbd.service nmbd.service
```

You can test that your Backup Domain controller is working by stopping the Samba daemon on the PDC, then trying to login to a Windows client joined to the domain.

Another thing to keep in mind is if you have configured the *logon home* option as a directory on the PDC, and the PDC becomes unavailable, access to the user's *Home* drive will also be unavailable. For this reason it is best to configure the *logon home* to reside on a separate file server from the PDC and BDC.

Resources

- For in depth Samba configurations see the Samba HOWTO Collection
- The guide is also available in printed format.
- O'Reilly's Using Samba is also a good reference.
- Chapter 4 of the Samba HOWTO Collection explains setting up a Primary Domain Controller.
- Chapter 5 of the Samba HOWTO Collection explains setting up a Backup Domain Controller.
- The Ubuntu Wiki Samba page.

File Server

One of the most common ways to network Ubuntu and Windows computers is to configure Samba as a File Server. This section covers setting up a Samba server to share files with Windows clients.

The server will be configured to share files with any client on the network without prompting for a password. If your environment requires stricter Access Controls see Securing File and Print Server.

Installation

The first step is to install the samba package. From a terminal prompt enter:

```
sudo apt install samba
```

That's all there is to it; you are now ready to configure Samba to share files.

Configuration

The main Samba configuration file is located in /etc/samba/smb.conf. The default configuration file has a significant number of comments in order to document various configuration directives.

> **Note**
>
> Not all the available options are included in the default configuration file. See the smb.conf man page or the Samba HOWTO Collection for more details.

First, edit the following key/value pairs in the *[global]* section of /etc/samba/smb.conf:

```
workgroup = EXAMPLE
...
security = user
```

The *security* parameter is farther down in the [global] section, and is commented by default. Also, change *EXAMPLE* to better match your environment.

Create a new section at the bottom of the file, or uncomment one of the examples, for the directory to be shared:

```
[share]
    comment = Ubuntu File Server Share
    path = /srv/samba/share
    browsable = yes
    guest ok = yes
    read only = no
    create mask = 0755
```

- *comment:* a short description of the share. Adjust to fit your needs.
- *path:* the path to the directory to share.

  This example uses /srv/samba/sharename because, according to the *Filesystem Hierarchy Standard (FHS)*, /srv is where site-specific data should be served. Technically Samba shares can be placed anywhere on the filesystem as long as the permissions are correct, but adhering to standards is recommended.
- *browsable:* enables Windows clients to browse the shared directory using Windows Explorer.
- *guest ok:* allows clients to connect to the share without supplying a password.

- *read only:* determines if the share is read only or if write privileges are granted. Write privileges are allowed only when the value is *no*, as is seen in this example. If the value is *yes*, then access to the share is read only.
- *create mask:* determines the permissions new files will have when created.

Now that Samba is configured, the directory needs to be created and the permissions changed. From a terminal enter:

```
sudo mkdir -p /srv/samba/share
sudo chown nobody:nogroup /srv/samba/share/
```

Note

The *-p* switch tells mkdir to create the entire directory tree if it doesn't exist.

Finally, restart the samba services to enable the new configuration:

```
sudo systemctl restart smbd.service nmbd.service
```

Warning

Once again, the above configuration gives all access to any client on the local network. For a more secure configuration see Securing File and Print Server.

From a Windows client you should now be able to browse to the Ubuntu file server and see the shared directory. If your client doesn't show your share automatically, try to access your server by its IP address, e.g. \\192.168.1.1, in a Windows Explorer window. To check that everything is working try creating a directory from Windows.

To create additional shares simply create new *[dir]* sections in /etc/samba/smb.conf, and restart *Samba*. Just make sure that the directory you want to share actually exists and the permissions are correct.

Note

The file share named *"[share]"* and the path /srv/samba/share are just examples. Adjust the share and path names to fit your environment. It is a good idea to name a share after a directory on the file system. Another example would be a share name of *[qa]* with a path of /srv/samba/qa.

Resources

- For in depth Samba configurations see the Samba HOWTO Collection
- The guide is also available in printed format.
- O'Reilly's Using Samba is another good reference.
- The Ubuntu Wiki Samba page.

Samba

Computer networks are often comprised of diverse systems, and while operating a network made up entirely of Ubuntu desktop and server computers would certainly be fun, some network environments must consist of both Ubuntu and Microsoft Windows systems working together in harmony. This section of the UBUNTU SG-TITLE introduces principles and tools used in configuring your Ubuntu Server for sharing network resources with Windows computers.

Introduction

Successfully networking your Ubuntu system with Windows clients involves providing and integrating with services common to Windows environments. Such services assist the sharing of data and information about the computers and users involved in the network, and may be classified under three major categories of functionality:

- **File and Printer Sharing Services**. Using the Server Message Block (SMB) protocol to facilitate the sharing of files, folders, volumes, and the sharing of printers throughout the network.
- **Directory Services**. Sharing vital information about the computers and users of the network with such technologies as the Lightweight Directory Access Protocol (LDAP) and Microsoft Active Directory.
- **Authentication and Access**. Establishing the identity of a computer or user of the network and determining the information the computer or user is authorized to access using such principles and technologies as file permissions, group policies, and the Kerberos authentication service.

Fortunately, your Ubuntu system may provide all such facilities to Windows clients and share network resources among them. One of the principal pieces of software your Ubuntu system includes for Windows networking is the Samba suite of SMB server applications and tools.

This section of the UBUNTU SG-TITLE will introduce some of the common Samba use cases, and how to install and configure the necessary packages. Additional detailed documentation and information on Samba can be found on the Samba website.

Print Server

Another common use of Samba is to configure it to share printers installed, either locally or over the network, on an Ubuntu server. Similar to File Server this section will configure Samba to allow any client on the local network to use the installed printers without prompting for a username and password.

For a more secure configuration see Securing File and Print Server.

Installation

Before installing and configuring Samba it is best to already have a working CUPS installation. See ??? for details.

To install the samba package, from a terminal enter:

```
sudo apt install samba
```

Configuration

After installing samba edit /etc/samba/smb.conf. Change the *workgroup* attribute to what is appropriate for your network, and change *security* to *user*:

```
workgroup = EXAMPLE
...
security = user
```

In the *[printers]* section change the *guest ok* option to *yes*:

```
browsable = yes
guest ok = yes
```

After editing smb.conf restart Samba:

```
sudo systemctl restart smbd.service nmbd.service
```

The default Samba configuration will automatically share any printers installed. Simply install the printer locally on your Windows clients.

Resources

- For in depth Samba configurations see the Samba HOWTO Collection
- The guide is also available in printed format.
- O'Reilly's Using Samba is another good reference.
- Also, see the CUPS Website for more information on configuring CUPS.
- The Ubuntu Wiki Samba page.

Securing File and Print Server

Samba Security Modes

There are two security levels available to the Common Internet Filesystem (CIFS) network protocol *user-level* and *share-level*. Samba's *security mode* implementation allows more flexibility, providing four ways of implementing user-level security and one way to implement share-level:

- *security = user:* requires clients to supply a username and password to connect to shares. Samba user accounts are separate from system accounts, but the libpam-winbind package will sync system users and passwords with the Samba user database.
- *security = domain:* this mode allows the Samba server to appear to Windows clients as a Primary Domain Controller (PDC), Backup Domain Controller (BDC), or a Domain Member Server (DMS). See As a Domain Controller for further information.
- *security = ADS:* allows the Samba server to join an Active Directory domain as a native member. See Active Directory Integration for details.
- *security = server:* this mode is left over from before Samba could become a member server, and due to some security issues should not be used. See the Server Security section of the Samba guide for more details.
- *security = share:* allows clients to connect to shares without supplying a username and password.

The security mode you choose will depend on your environment and what you need the Samba server to accomplish.

Security = User

This section will reconfigure the Samba file and print server, from File Server and Print Server, to require authentication.

First, install the libpam-winbind package which will sync the system users to the Samba user database:

```
sudo apt install libpam-winbind
```

Note

If you chose the *Samba Server* task during installation libpam-winbind is already installed.

Edit /etc/samba/smb.conf, and in the *[share]* section change:

```
guest ok = no
```

Finally, restart Samba for the new settings to take effect:

```
sudo systemctl restart smbd.service nmbd.service
```

Now when connecting to the shared directories or printers you should be prompted for a username and password.

Note

If you choose to map a network drive to the share you can check the "Reconnect at Logon" check box, which will require you to only enter the username and password once, at least until the password changes.

Share Security

There are several options available to increase the security for each individual shared directory. Using the *[share]* example, this section will cover some common options.

Groups

Groups define a collection of computers or users which have a common level of access to particular network resources and offer a level of granularity in controlling access to such resources. For example, if a group *qa* is defined and contains the users *freda*, *danika*, and *rob* and a second group *support* is defined and consists of users *danika*, *jeremy*, and *vincent* then certain network resources configured to allow access by the *qa* group will subsequently enable access by freda, danika, and rob, but not jeremy or vincent. Since the user *danika* belongs to both the *qa* and *support* groups, she will be able to access resources configured for access by both groups, whereas all other users will have only access to resources explicitly allowing the group they are part of.

By default Samba looks for the local system groups defined in /etc/group to determine which users belong to which groups. For more information on adding and removing users from groups see ???.

When defining groups in the Samba configuration file, /etc/samba/smb.conf, the recognized syntax is to preface the group name with an "@" symbol. For example, if you wished to define a group named *sysadmin* in a certain section of the /etc/samba/smb.conf, you would do so by entering the group name as **@sysadmin**.

File Permissions

File Permissions define the explicit rights a computer or user has to a particular directory, file, or set of files. Such permissions may be defined by editing the /etc/samba/smb.conf file and specifying the explicit permissions of a defined file share.

For example, if you have defined a Samba share called *share* and wish to give *read-only* permissions to the group of users known as *qa*, but wanted to allow writing to the share by the group called *sysadmin* and the user named *vincent*, then you could edit the /etc/samba/smb.conf file, and add the following entries under the *[share]* entry:

```
read list = @qa
write list = @sysadmin, vincent
```

Another possible Samba permission is to declare *administrative* permissions to a particular shared resource. Users having administrative permissions may read, write, or modify any information contained in the resource the user has been given explicit administrative permissions to.

For example, if you wanted to give the user *melissa* administrative permissions to the *share* example, you would edit the /etc/samba/smb.conf file, and add the following line under the *[share]* entry:

```
admin users = melissa
```

After editing /etc/samba/smb.conf, restart Samba for the changes to take effect:

```
sudo systemctl restart smbd.service nmbd.service
```

Note

For the *read list* and *write list* to work the Samba security mode must *not* be set to *security = share*

Now that Samba has been configured to limit which groups have access to the shared directory, the filesystem permissions need to be updated.

Traditional Linux file permissions do not map well to Windows NT Access Control Lists (ACLs). Fortunately POSIX ACLs are available on Ubuntu servers providing more fine grained control. For example, to enable ACLs on /srv an EXT3 filesystem, edit /etc/fstab adding the *acl* option:

```
UUID=66bcdd2e-8861-4fb0-b7e4-e61c569fe17d /srv ext3 noatime,relatime,acl 0
1
```

Then remount the partition:

```
sudo mount -v -o remount /srv
```

Note

The above example assumes /srv on a separate partition. If /srv, or wherever you have configured your share path, is part of the / partition a reboot may be required.

To match the Samba configuration above the *sysadmin* group will be given read, write, and execute permissions to /srv/samba/share, the *qa* group will be given read and execute permissions, and the files will be owned by the username *melissa*. Enter the following in a terminal:

```
sudo chown -R melissa /srv/samba/share/
sudo chgrp -R sysadmin /srv/samba/share/
sudo setfacl -R -m g:qa:rx /srv/samba/share/
```

Note

The setfacl command above gives *execute* permissions to all files in the /srv/samba/share directory, which you may or may not want.

Now from a Windows client you should notice the new file permissions are implemented. See the acl and setfacl man pages for more information on POSIX ACLs.

Samba AppArmor Profile

Ubuntu comes with the AppArmor security module, which provides mandatory access controls. The default AppArmor profile for Samba will need to be adapted to your configuration. For more details on using AppArmor see ???.

There are default AppArmor profiles for /usr/sbin/smbd and /usr/sbin/nmbd, the Samba daemon binaries, as part of the apparmor-profiles packages. To install the package, from a terminal prompt enter:

```
sudo apt install apparmor-profiles apparmor-utils
```

Note

This package contains profiles for several other binaries.

By default the profiles for smbd and nmbd are in *complain* mode allowing Samba to work without modifying the profile, and only logging errors. To place the smbd profile into *enforce* mode, and have Samba work as expected, the profile will need to be modified to reflect any directories that are shared.

Edit /etc/apparmor.d/usr.sbin.smbd adding information for *[share]* from the file server example:

```
/srv/samba/share/ r,
/srv/samba/share/** rwkix,
```

Now place the profile into *enforce* and reload it:

```
sudo aa-enforce /usr/sbin/smbd
cat /etc/apparmor.d/usr.sbin.smbd | sudo apparmor_parser -r
```

You should now be able to read, write, and execute files in the shared directory as normal, and the smbd binary will have access to only the configured files and directories. Be sure to add entries for each directory you configure Samba to share. Also, any errors will be logged to /var/log/syslog.

Resources

- For in depth Samba configurations see the Samba HOWTO Collection
- The guide is also available in printed format.
- O'Reilly's Using Samba is also a good reference.
- Chapter 18 of the Samba HOWTO Collection is devoted to security.
- For more information on Samba and ACLs see the Samba ACLs page.
- The Ubuntu Wiki Samba page.

Samba - OpenLDAP Backend

NOTE

This section is flagged as *legacy* because nowadays Samba 4 is best integrated with its own LDAP server in AD mode. Integrating Samba with LDAP as described here covers the NT4 mode, deprecated for many years.

This section covers the integration of Samba with LDAP. The Samba server's role will be that of a "standalone" server and the LDAP directory will provide the authentication layer in addition to containing the user, group, and machine account information that Samba requires in order to function (in any of its 3 possible roles). The pre-requisite is an OpenLDAP server configured with a directory that can accept authentication requests. See Service - LDAP and Service - LDAP with TLS for details on fulfilling this requirement. Once those steps are completed, you will need to decide what specifically you want Samba to do for you and then configure it accordingly.

This guide will assume that the LDAP and Samba services are running on the same server and therefore use SASL EXTERNAL authentication whenever changing something under *cn=config*. If that is not your scenario, you will have to run those ldap commands on the LDAP server.

Software Installation

There are two packages needed when integrating Samba with LDAP: samba and smbldap-tools.

Strictly speaking, the smbldap-tools package isn't needed, but unless you have some other way to manage the various Samba entities (users, groups, computers) in an LDAP context then you should install it.

Install these packages now:

```
sudo apt install samba smbldap-tools
```

LDAP Configuration

We will now configure the LDAP server so that it can accomodate Samba data. We will perform three tasks in this section:

- Import a schema
- Index some entries
- Add objects

Samba schema

In order for OpenLDAP to be used as a backend for Samba, the DIT will need to use attributes that can properly describe Samba data. Such attributes can be obtained by introducing a Samba LDAP schema. Let's do this now.

The schema is found in the now-installed samba package and is already in the ldif format. We can import it with one simple command:

```
sudo ldapadd -Q -Y EXTERNAL -H ldapi:/// -f /usr/share/doc/samba/examples/LDAP
    /samba.ldif
```

To query and view this new schema:

```
sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b cn=schema,cn=config 'cn=*
    samba*'
```

Samba indices

Now that slapd knows about the Samba attributes, we can set up some indices based on them. Indexing entries is a way to improve performance when a client performs a filtered search on the DIT.

Create the file samba_indices.ldif with the following contents:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
replace: olcDbIndex
olcDbIndex: objectClass eq
olcDbIndex: uidNumber,gidNumber eq
olcDbIndex: loginShell eq
olcDbIndex: uid,cn eq,sub
olcDbIndex: memberUid eq,sub
olcDbIndex: member,uniqueMember eq
olcDbIndex: sambaSID eq
```

```
olcDbIndex: sambaPrimaryGroupSID eq
olcDbIndex: sambaGroupType eq
olcDbIndex: sambaSIDList eq
olcDbIndex: sambaDomainName eq
olcDbIndex: default sub,eq
```

Using the ldapmodify utility load the new indices:

```
sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// -f samba_indices.ldif
```

If all went well you should see the new indices using ldapsearch:

```
sudo ldapsearch -Q -LLL -Y EXTERNAL -H \
ldapi:/// -b cn=config olcDatabase={1}mdb olcDbIndex
```

Adding Samba LDAP objects

Next, configure the smbldap-tools package to match your environment. The package comes with a configuration helper script called *smbldap-config*. Before running it, though, you should decide on two important configuration settings in /etc/samba/smb.conf:

- *netbios name*: how this server will be known. The default value is derived from the server's hostname, but truncated at 15 characters.
- *workgroup*: the workgroup name for this server, or, if you later decide to make it a domain controller, this will be the domain.

It's important to make these choices now because *smbldap-config* will use them to generate the config that will be later stored in the LDAP directory. If you run *smbldap-config* now and later change these values in /etc/samba/smb.conf there will be an inconsistency.

Once you are happy with *netbios name* and *workgroup*, proceed to generate the *smbldap-tools* configuration by running the configuration script which will ask you some questions:

```
sudo smbldap-config
```

Some of the more important ones:

- *workgroup name*: has to match what you will configure in /etc/samba/smb.conf later on.
- *ldap suffix*: has to match the ldap suffix you chose when you configured the LDAP server.
- other ldap suffixes: they are all relative to *ldap suffix* above. For example, for *ldap user suffix* you should use *ou=People*, and for computer/machines, use *ou=Computers*.
- *ldap master bind dn* and *bind password*: use the rootDN credentials.

The *smbldap-populate* script will then add the LDAP objects required for Samba. It will ask you for a password for the "domain root" user, which is also the "root" user stored in LDAP:

```
sudo smbldap-populate -g 10000 -u 10000 -r 10000
```

The *-g*, *-u* and *-r* parameters tell smbldap-tools where to start the numeric uid and gid allocation for the LDAP users. You should pick a range start that does not overlap with your local */etc/passwd* users.

You can create a LDIF file containing the new Samba objects by executing sudo smbldap-populate -e samba.ldif. This allows you to look over the changes making sure everything is correct. If it is, rerun the script without the '-e' switch. Alternatively, you can take the LDIF file and import its data per usual.

Your LDAP directory now has the necessary information to authenticate Samba users.

Samba Configuration

To configure Samba to use LDAP, edit its configuration file /etc/samba/smb.conf commenting out the default *passdb backend* parameter and adding some ldap-related ones. Make sure to use the same values you used when running *smbldap-populate*:

```
#   passdb backend = tdbsam
workgroup = EXAMPLE

# LDAP Settings
passdb backend = ldapsam:ldap://ldap01.example.com
ldap suffix = dc=example,dc=com
ldap user suffix = ou=People
ldap group suffix = ou=Groups
ldap machine suffix = ou=Computers
ldap idmap suffix = ou=Idmap
ldap admin dn = cn=admin,dc=example,dc=com
ldap ssl = start tls
ldap passwd sync = yes
```

Change the values to match your environment.

> **Note**
>
> The smb.conf as shipped by the package is quite long and has many configuration examples. An easy way to visualize it without any comments is to run *testparm -s*.

Now inform Samba about the rootDN user's password (the one set during the installation of the slapd package):

```
sudo smbpasswd -W
```

As a final step to have your LDAP users be able to connect to samba and authenticate, we need these users to also show up in the system as "unix" users. Use SSSD for that as detailed in Service - SSSD.

Install sssd-ldap

```
sudo apt install sssd-ldap
```

Configure */etc/sssd/sssd.conf*:

```
[sssd]
config_file_version = 2
domains = example.com

[domain/example.com]
id_provider = ldap
auth_provider = ldap
ldap_uri = ldap://ldap01.example.com
cache_credentials = True
ldap_search_base = dc=example,dc=com
```

Adjust permissions and start the service:

```
sudo chmod 0600 /etc/sssd/sssd.conf
sudo chown root:root /etc/sssd/sssd.conf
sudo systemctl start sssd
```

Restart the Samba services:

```
sudo systemctl restart smbd.service nmbd.service
```

To quickly test the setup, see if getent can list the Samba groups:

```
$ getent group Replicators
Replicators:*:552:
```

Note

The names are case sensitive!

If you have existing LDAP users that you want to include in your new LDAP-backed Samba they will, of course, also need to be given some of the extra Samba specific attributes. The *smbpasswd* utility can do this for you:

```
sudo smbpasswd -a username
```

You will prompted to enter a password. It will be considered as the new password for that user. Making it the same as before is reasonable. Note that this command cannot be used to create a new user from scratch in LDAP (unless you are using *ldapsam:trusted* and *ldapsam:editposix*, not covered in this guide).

To manage user, group, and machine accounts use the utilities provided by the *smbldap-tools* package. Here are some examples:

- To add a new user with a home directory:

  ```
  sudo smbldap-useradd -a -P -m username
  ```

  The *-a* option adds the Samba attributes, and the *-P* option calls the *smbldap-passwd* utility after the user is created allowing you to enter a password for the user. Finally, *-m* creates a local home directory. Test with the getent command:

  ```
  getent passwd username
  ```

- To remove a user:

  ```
  sudo smbldap-userdel username
  ```

  In the above command, use the *-r* option to remove the user's home directory.

- To add a group:

  ```
  sudo smbldap-groupadd -a groupname
  ```

  As for *smbldap-useradd*, the *-a* adds the Samba attributes.

- To make an existing user a member of a group:

  ```
  sudo smbldap-groupmod -m username groupname
  ```

  The *-m* option can add more than one user at a time by listing them in comma-separated format.

- To remove a user from a group:

  ```
  sudo smbldap-groupmod -x username groupname
  ```

- To add a Samba machine account:

  ```
  sudo smbldap-useradd -t 0 -w username
  ```

  Replace *username* with the name of the workstation. The *-t 0* option creates the machine account without a delay, while the *-w* option specifies the user as a machine account.

Resources

- Upstream documentation collection: https://www.samba.org/samba/docs/
- Upstream samba wiki: https://wiki.samba.org/index.php/Main_Page

CUPS - Print Server

The primary mechanism for Ubuntu printing and print services is the **Common UNIX Printing System** (CUPS). This printing system is a freely available, portable printing layer which has become the new standard for printing in most Linux distributions.

CUPS manages print jobs and queues and provides network printing using the standard Internet Printing Protocol (IPP), while offering support for a very large range of printers, from dot-matrix to laser and many in between. CUPS also supports PostScript Printer Description (PPD) and auto-detection of network printers, and features a simple web-based configuration and administration tool.

Installation

To install CUPS on your Ubuntu computer, simply use sudo with the apt command and give the packages to install as the first parameter. A complete CUPS install has many package dependencies, but they may all be specified on the same command line. Enter the following at a terminal prompt to install CUPS:

```
sudo apt install cups
```

Upon authenticating with your user password, the packages should be downloaded and installed without error. Upon the conclusion of installation, the CUPS server will be started automatically.

For troubleshooting purposes, you can access CUPS server errors via the error log file at: /var/log/cups/error_log. If the error log does not show enough information to troubleshoot any problems you encounter, the verbosity of the CUPS log can be increased by changing the **LogLevel** directive in the configuration file (discussed below) to "debug" or even "debug2", which logs everything, from the default of "info". If you make this change, remember to change it back once you've solved your problem, to prevent the log file from becoming overly large.

Configuration

The Common UNIX Printing System server's behavior is configured through the directives contained in the file /etc/cups/cupsd.conf. The CUPS configuration file follows the same syntax as the primary configuration file for the Apache HTTP server, so users familiar with editing Apache's configuration file should feel at ease when editing the CUPS configuration file. Some examples of settings you may wish to change initially will be presented here.

> **Tip**
>
> Prior to editing the configuration file, you should make a copy of the original file and protect it from writing, so you will have the original settings as a reference, and to reuse as necessary.
>
> Copy the /etc/cups/cupsd.conf file and protect it from writing with the following commands, issued at a terminal prompt:

```
sudo cp /etc/cups/cupsd.conf /etc/cups/cupsd.conf.original
sudo chmod a-w /etc/cups/cupsd.conf.original
```

- **ServerAdmin**: To configure the email address of the designated administrator of the CUPS server, simply edit the /etc/cups/cupsd.conf configuration file with your preferred text editor, and add or modify the *ServerAdmin* line accordingly. For example, if you are the Administrator for the CUPS server, and your e-mail address is 'bjoy@somebigco.com', then you would modify the ServerAdmin line to appear as such:

  ```
  ServerAdmin bjoy@somebigco.com
  ```

- **Listen**: By default on Ubuntu, the CUPS server installation listens only on the loopback interface at IP address *127.0.0.1*. In order to instruct the CUPS server to listen on an actual network adapter's IP address, you must specify either a hostname, the IP address, or optionally, an IP address/port pairing via the addition of a Listen directive. For example, if your CUPS server resides on a local network at the IP address *192.168.10.250* and you'd like to make it accessible to the other systems on this subnetwork, you would edit the /etc/cups/cupsd.conf and add a Listen directive, as such:

  ```
  Listen 127.0.0.1:631           # existing loopback Listen
  Listen /var/run/cups/cups.sock # existing socket Listen
  Listen 192.168.10.250:631      # Listen on the LAN interface, Port 631 (
      IPP)
  ```

  In the example above, you may comment out or remove the reference to the Loopback address (127.0.0.1) if you do not wish cupsd to listen on that interface, but would rather have it only listen on the Ethernet interfaces of the Local Area Network (LAN). To enable listening for all network interfaces for which a certain hostname is bound, including the Loopback, you could create a Listen entry for the hostname *socrates* as such:

  ```
  Listen socrates:631  # Listen on all interfaces for the hostname 'socrates
      '
  ```

  or by omitting the Listen directive and using *Port* instead, as in:

  ```
  Port 631  # Listen on port 631 on all interfaces
  ```

For more examples of configuration directives in the CUPS server configuration file, view the associated system manual page by entering the following command at a terminal prompt:

```
man cupsd.conf
```

Note

Whenever you make changes to the /etc/cups/cupsd.conf configuration file, you'll need to restart the CUPS server by typing the following command at a terminal prompt:

```
sudo systemctl restart cups.service
```

Web Interface

Tip

CUPS can be configured and monitored using a web interface, which by default is available at http://localhost:631/admin. The web interface can be used to perform all printer management tasks.

In order to perform administrative tasks via the web interface, you must either have the root account enabled on your server, or authenticate as a user in the *lpadmin* group. For security reasons, CUPS won't authenticate a user that doesn't have a password.

To add a user to the *lpadmin* group, run at the terminal prompt:

```
sudo usermod –aG lpadmin username
```

Further documentation is available in the *Documentation/Help* tab of the web interface.

References

CUPS Website

Debian Open-iSCSI page

Domain Name Service (DNS)

Domain Name Service (DNS) is an Internet service that maps IP addresses and fully qualified domain names (FQDN) to one another. In this way, DNS alleviates the need to remember IP addresses. Computers that run DNS are called *name servers*. Ubuntu ships with BIND (Berkley Internet Naming Daemon), the most common program used for maintaining a name server on Linux.

Installation

At a terminal prompt, enter the following command to install dns:

```
sudo apt install bind9
```

A very useful package for testing and troubleshooting DNS issues is the dnsutils package. Very often these tools will be installed already, but to check and/or install dnsutils enter the following:

```
sudo apt install dnsutils
```

Configuration

There are many ways to configure BIND9. Some of the most common configurations are a caching nameserver, primary server, and secondary server.

- When configured as a caching nameserver BIND9 will find the answer to name queries and remember the answer when the domain is queried again.
- As a primary server, BIND9 reads the data for a zone from a file on its host and is authoritative for that zone.
- As a secondary server, BIND9 gets the zone data from another nameserver that is authoritative for the zone.

Overview

The DNS configuration files are stored in the /etc/bind directory. The primary configuration file is /etc/bind/named.conf, which in the layout provided by the package just includes these files.

- /etc/bind/named.conf.options: global DNS options
- /etc/bind/named.conf.local: for your zones
- /etc/bind/named.conf.default–zones: default zones such as localhost, its reverse, and the root hints

The root nameservers used to be described in the file /etc/bind/db.root. This is now provided instead by the /usr/share/dns/root.hints file shipped with the dns−root−data package, and is referenced in the named.conf.default−zones configuration file above.

It is possible to configure the same server to be a caching name server, primary, and secondary: it all depends on the zones it is serving. A server can be the Start of Authority (SOA) for one zone, while providing secondary service for another zone. All the while providing caching services for hosts on the local LAN.

Caching Nameserver

The default configuration acts as a caching server. Simply uncomment and edit /etc/bind/named.conf. options to set the IP addresses of your ISP's DNS servers:

```
forwarders {
    1.2.3.4;
    5.6.7.8;
};
```

Note

Replace 1.2.3.4 and 5.6.7.8 with the IP Addresses of actual nameservers.

To enable the new configuration, restart the DNS server. From a terminal prompt:

```
sudo systemctl restart bind9.service
```

See dig for information on testing a caching DNS server.

Primary Server

In this section BIND9 will be configured as the Primary server for the domain example.com. Simply replace example.com with your FQDN (Fully Qualified Domain Name).

Forward Zone File

To add a DNS zone to BIND9, turning BIND9 into a Primary server, first edit /etc/bind/named.conf.local:

```
zone "example.com" {
    type master;
    file "/etc/bind/db.example.com";
};
```

Note

If bind will be receiving automatic updates to the file as with DDNS, then use /var/lib/bind /db.example.com rather than /etc/bind/db.example.com both here and in the copy command below.

Now use an existing zone file as a template to create the /etc/bind/db.example.com file:

```
sudo cp /etc/bind/db.local /etc/bind/db.example.com
```

Edit the new zone file /etc/bind/db.example.com and change localhost. to the FQDN of your server, leaving the additional . at the end. Change 127.0.0.1 to the nameserver's IP Address and root.localhost to a valid email address, but with a . instead of the usual @ symbol, again leaving the . at the end. Change the comment to indicate the domain that this file is for.

Create an *A record* for the base domain, example.com. Also, create an *A record* for ns.example.com, the name server in this example:

```
;
; BIND data file for example.com
;
$TTL    604800
@       IN      SOA     example.com. root.example.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL

@       IN      NS      ns.example.com.
@       IN      A       192.168.1.10
@       IN      AAAA    ::1
ns      IN      A       192.168.1.10
```

You must increment the *Serial Number* every time you make changes to the zone file. If you make multiple changes before restarting BIND9, simply increment the Serial once.

Now, you can add DNS records to the bottom of the zone file. See Common Record Types for details.

> **Note**
>
> Many admins like to use the last date edited as the serial of a zone, such as *2020012100* which is *yyyymmddss* (where *ss* is the Serial Number)

Once you have made changes to the zone file BIND9 needs to be restarted for the changes to take effect:

```
sudo systemctl restart bind9.service
```

Reverse Zone File

Now that the zone is setup and resolving names to IP Addresses, a *Reverse zone* needs to be added to allows DNS to resolve an address to a name.

Edit /etc/bind/named.conf.local and add the following:

```
zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192";
};
```

> **Note**
>
> Replace 1.168.192 with the first three octets of whatever network you are using. Also, name the zone file /etc/bind/db.192 appropriately. It should match the first octet of your network.

Now create the /etc/bind/db.192 file:

```
sudo cp /etc/bind/db.127 /etc/bind/db.192
```

Next edit /etc/bind/db.192 changing the same options as /etc/bind/db.example.com:

```
;
; BIND reverse data file for local 192.168.1.XXX net
;
```

```
$TTL    604800
@       IN      SOA     ns.example.com. root.example.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.
10      IN      PTR     ns.example.com.
```

The *Serial Number* in the Reverse zone needs to be incremented on each change as well. For each *A record* you configure in /etc/bind/db.example.com, that is for a different address, you need to create a *PTR record* in /etc/bind/db.192.

After creating the reverse zone file restart BIND9:

```
sudo systemctl restart bind9.service
```

Secondary Server

Once a *Primary Server* has been configured a *Secondary Server* is highly recommended in order to maintain the availability of the domain should the Primary become unavailable.

First, on the Primary server, the zone transfer needs to be allowed. Add the allow−transfer option to the example Forward and Reverse zone definitions in /etc/bind/named.conf.local:

```
zone "example.com" {
    type master;
    file "/etc/bind/db.example.com";
    allow-transfer { 192.168.1.11; };
};

zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192";
    allow-transfer { 192.168.1.11; };
};
```

Note

Replace 192.168.1.11 with the IP Address of your Secondary nameserver.

Restart BIND9 on the Primary server:

```
sudo systemctl restart bind9.service
```

Next, on the Secondary server, install the bind9 package the same way as on the Primary. Then edit the /etc/bind/named.conf.local and add the following declarations for the Forward and Reverse zones:

```
zone "example.com" {
    type slave;
    file "db.example.com";
    masters { 192.168.1.10; };
};

zone "1.168.192.in-addr.arpa" {
```

```
    type slave;
    file "db.192";
    masters { 192.168.1.10; };
};
```

Note

Replace 192.168.1.10 with the IP Address of your Primary nameserver.

Restart BIND9 on the Secondary server:

```
sudo systemctl restart bind9.service
```

In /var/log/syslog you should see something similar to the following (some lines have been split to fit the format of this document):

```
client 192.168.1.10#39448: received notify for zone '1.168.192.in-addr.arpa'
zone 1.168.192.in-addr.arpa/IN: Transfer started.
transfer of '100.18.172.in-addr.arpa/IN' from 192.168.1.10#53:
 connected using 192.168.1.11#37531
zone 1.168.192.in-addr.arpa/IN: transferred serial 5
transfer of '100.18.172.in-addr.arpa/IN' from 192.168.1.10#53:
 Transfer completed: 1 messages,
6 records, 212 bytes, 0.002 secs (106000 bytes/sec)
zone 1.168.192.in-addr.arpa/IN: sending notifies (serial 5)

client 192.168.1.10#20329: received notify for zone 'example.com'
zone example.com/IN: Transfer started.
transfer of 'example.com/IN' from 192.168.1.10#53: connected using
   192.168.1.11#38577
zone example.com/IN: transferred serial 5
transfer of 'example.com/IN' from 192.168.1.10#53: Transfer completed: 1
   messages,
8 records, 225 bytes, 0.002 secs (112500 bytes/sec)
```

Note

Note: A zone is only transferred if the *Serial Number* on the Primary is larger than the one on the Secondary. If you want to have your Primary DNS notifying other Secondary DNS Servers of zone changes, you can add also-notify { ipaddress; }; to /etc/bind/named.conf.local as shown in the example below:

```
zone "example.com" {
    type master;
    file "/etc/bind/db.example.com";
    allow-transfer { 192.168.1.11; };
    also-notify { 192.168.1.11; };
};

zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192";
    allow-transfer { 192.168.1.11; };
    also-notify { 192.168.1.11; };
};
```

Note

The default directory for non-authoritative zone files is /var/cache/bind/. This directory is also configured in AppArmor to allow the named daemon to write to it. For more information on AppArmor see Security - AppArmor.

Troubleshooting

This section covers diagnosing problems with DNS and BIND9 configurations.

Testing

resolv.conf

The first step in testing BIND9 is to add the nameserver's IP Address to a hosts resolver. The Primary nameserver should be configured as well as another host to double check things. Refer to DNS client configuration for details on adding nameserver addresses to your network clients. In the end your nameserver line in /etc/resolv.conf should be pointing at 127.0.0.53 and you should have a search parameter for your domain. Something like this:

```
nameserver   127.0.0.53
search example.com
```

To check which DNS server your local resolver is using, run:

```
systemd-resolve --status
```

Note

You should also add the IP Address of the Secondary nameserver to your client configuration in case the Primary becomes unavailable.

dig

If you installed the dnsutils package you can test your setup using the DNS lookup utility dig:

- After installing BIND9 use dig against the loopback interface to make sure it is listening on port 53. From a terminal prompt:

  ```
  dig -x 127.0.0.1
  ```

  You should see lines similar to the following in the command output:

  ```
  ;; Query time: 1 msec
  ;; SERVER: 192.168.1.10#53(192.168.1.10)
  ```

- If you have configured BIND9 as a *Caching* nameserver "dig" an outside domain to check the query time:

  ```
  dig ubuntu.com
  ```

  Note the query time toward the end of the command output:

  ```
  ;; Query time: 49 msec
  ```

  After a second dig there should be improvement:

  ```
  ;; Query time: 1 msec
  ```

ping

Now to demonstrate how applications make use of DNS to resolve a host name use the ping utility to send an ICMP echo request:

```
ping example.com
```

This tests if the nameserver can resolve the name ns.example.com to an IP Address. The command output should resemble:

```
PING ns.example.com (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=0.800 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=64 time=0.813 ms
```

named-checkzone

A great way to test your zone files is by using the named−checkzone utility installed with the bind9 package. This utility allows you to make sure the configuration is correct before restarting BIND9 and making the changes live.

- To test our example Forward zone file enter the following from a command prompt:

  ```
  named−checkzone example.com /etc/bind/db.example.com
  ```

  If everything is configured correctly you should see output similar to:

  ```
  zone example.com/IN: loaded serial 6
  OK
  ```

- Similarly, to test the Reverse zone file enter the following:

  ```
  named−checkzone 1.168.192.in−addr.arpa /etc/bind/db.192
  ```

  The output should be similar to:

  ```
  zone 1.168.192.in−addr.arpa/IN: loaded serial 3
  OK
  ```

  Note

  The *Serial Number* of your zone file will probably be different.

Quick temporary query logging

With the rndc tool, you can quickly turn query logging on and off, without restarting the service or changing the configuration file.

To turn query logging *on*, run:

```
sudo rndc querylog on
```

Likewise, to turn it off, run:

```
sudo rndc querylog off
```

The logs will be sent to syslog and will show up in /var/log/syslog by default:

```
Jan 20 19:40:50 new-n1 named[816]: received control channel command 'querylog
    on'
Jan 20 19:40:50 new-n1 named[816]: query logging is now on
Jan 20 19:40:57 new-n1 named[816]: client @0x7f48ec101480 192.168.1.10#36139 (
    ubuntu.com): query: ubuntu.com IN A +E(0)K (192.168.1.10)
```

Note

The amount of logs generated by enabling querylog could be huge!

Logging

BIND9 has a wide variety of logging configuration options available, but the two main ones are *channel* and *category*, which configure where logs go, and what information gets logged, respectively.

If no logging options are configured the default configuration is:

```
logging {
     category default { default_syslog; default_debug; };
     category unmatched { null; };
};
```

Let's instead configure BIND9 to send *debug* messages related to DNS queries to a separate file.

We need to configure a *channel* to specify which file to send the messages to, and a *category*. In this example, the *category* will log all queries. Edit /etc/bind/named.conf.local and add the following:

```
logging {
    channel query.log {
        file "/var/log/named/query.log";
        severity debug 3;
    };
    category queries { query.log; };
};
```

Note

The *debug* option can be set from 1 to 3. If a level isn't specified, level 1 is the default.

- Since the *named daemon* runs as the *bind* user the /var/log/named directory must be created and the ownership changed:

  ```
  sudo mkdir /var/log/named
  sudo chown bind:bind /var/log/named
  ```

- Now restart BIND9 for the changes to take effect:

  ```
  sudo systemctl restart bind9.service
  ```

You should see the file /var/log/named/query.log fill with query information. This is a simple example of the BIND9 logging options. For coverage of advanced options see More Information.

References

Common Record Types

This section covers some of the most common DNS record types.

- A record: This record maps an IP Address to a hostname.

```
www     IN    A     192.168.1.12
```

- CNAME record: Used to create an alias to an existing A record. You cannot create a CNAME record pointing to another CNAME record.

```
web     IN    CNAME www
```

- MX record: Used to define where email should be sent to. Must point to an A record, not a CNAME.

```
@       IN    MX 1  mail.example.com.
mail    IN    A     192.168.1.13
```

- NS record: Used to define which servers serve copies of a zone. It must point to an A record, not a CNAME. This is where Primary and Secondary servers are defined.

```
@       IN    NS    ns.example.com.
@       IN    NS    ns2.example.com.
ns      IN    A     192.168.1.10
ns2     IN    A     192.168.1.11
```

More Information

- Upstream BIND9 Documentation
- Bind9.net has links to a large collection of DNS and BIND9 resources.
- DNS and BIND is a popular book now in it's fifth edition. There is now also a DNS and BIND on IPv6 book.
- A great place to ask for BIND9 assistance, and get involved with the Ubuntu Server community, is the *#ubuntu-server* IRC channel on freenode.

FTP Server

File Transfer Protocol (FTP) is a TCP protocol for downloading files between computers. In the past, it has also been used for uploading but, as that method does not use encryption, user credentials as well as data transferred in the clear and are easily intercepted. So if you are here looking for a way to upload and download files securely, see the OpenSSH documentation instead.

FTP works on a client/server model. The server component is called an *FTP daemon*. It continuously listens for FTP requests from remote clients. When a request is received, it manages the login and sets up the connection. For the duration of the session it executes any of commands sent by the FTP client.

Access to an FTP server can be managed in two ways:

- Anonymous
- Authenticated

In the Anonymous mode, remote clients can access the FTP server by using the default user account called "anonymous" or "ftp" and sending an email address as the password. In the Authenticated mode a user must have an account and a password. This latter choice is very insecure and should not be used except in special circumstances. If you are looking to transfer files securely see SFTP in the section on OpenSSH-Server. User access to the FTP server directories and files is dependent on the permissions defined for the account used at login. As a general rule, the FTP daemon will hide the root directory of the FTP server and change it to the FTP Home directory. This hides the rest of the file system from remote sessions.

vsftpd - FTP Server Installation

vsftpd is an FTP daemon available in Ubuntu. It is easy to install, set up, and maintain. To install vsftpd you can run the following command:

```
sudo apt install vsftpd
```

Anonymous FTP Configuration

By default vsftpd is *not* configured to allow anonymous download. If you wish to enable anonymous download edit /etc/vsftpd.conf by changing:

```
anonymous_enable=YES
```

During installation a *ftp* user is created with a home directory of /srv/ftp. This is the default FTP directory.

If you wish to change this location, to /srv/ files /ftp for example, simply create a directory in another location and change the *ftp* user's home directory:

```
sudo mkdir -p /srv/files/ftp
sudo usermod -d /srv/files/ftp ftp
```

After making the change restart vsftpd:

```
sudo systemctl restart vsftpd.service
```

Finally, copy any files and directories you would like to make available through anonymous FTP to /srv/ files /ftp, or /srv/ftp if you wish to use the default.

User Authenticated FTP Configuration

By default vsftpd is configured to authenticate system users and allow them to download files. If you want users to be able to upload files, edit /etc/vsftpd.conf:

```
write_enable=YES
```

Now restart vsftpd:

```
sudo systemctl restart vsftpd.service
```

Now when system users login to FTP they will start in their *home* directories where they can download, upload, create directories, etc.

Similarly, by default, anonymous users are not allowed to upload files to FTP server. To change this setting, you should uncomment the following line, and restart vsftpd:

```
anon_upload_enable=YES
```

> **Warning**
>
> Enabling anonymous FTP upload can be an extreme security risk. It is best to not enable anonymous upload on servers accessed directly from the Internet.

The configuration file consists of many configuration parameters. The information about each parameter is available in the configuration file. Alternatively, you can refer to the man page, man 5 vsftpd.conf for details of each parameter.

Securing FTP

There are options in /etc/vsftpd.conf to help make vsftpd more secure. For example users can be limited to their home directories by uncommenting:

```
chroot_local_user=YES
```

You can also limit a specific list of users to just their home directories:

```
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd.chroot_list
```

After uncommenting the above options, create a /etc/vsftpd.chroot_list containing a list of users one per line. Then restart vsftpd:

```
sudo systemctl restart vsftpd.service
```

Also, the /etc/ftpusers file is a list of users that are *disallowed* FTP access. The default list includes root, daemon, nobody, etc. To disable FTP access for additional users simply add them to the list.

FTP can also be encrypted using *FTPS*. Different from *SFTP*, *FTPS* is FTP over Secure Socket Layer (SSL). *SFTP* is a FTP like session over an encrypted *SSH* connection. A major difference is that users of SFTP need to have a *shell* account on the system, instead of a *nologin* shell. Providing all users with a shell may not be ideal for some environments, such as a shared web host. However, it is possible to restrict such accounts to only SFTP and disable shell interaction.

To configure *FTPS*, edit /etc/vsftpd.conf and at the bottom add:

```
ssl_enable=YES
```

Also, notice the certificate and key related options:

```
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
```

By default these options are set to the certificate and key provided by the ssl-cert package. In a production environment these should be replaced with a certificate and key generated for the specific host. For more information on certificates see Security - Certificates.

Now restart vsftpd, and non-anonymous users will be forced to use *FTPS*:

```
sudo systemctl restart vsftpd.service
```

To allow users with a shell of /usr/sbin/nologin access to FTP, but have no shell access, edit /etc/shells adding the *nologin* shell:

```
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/screen
/usr/sbin/nologin
```

This is necessary because, by default vsftpd uses PAM for authentication, and the /etc/pam.d/vsftpd configuration file contains:

```
auth    required        pam_shells.so
```

The *shells* PAM module restricts access to shells listed in the /etc/shells file.

Most popular FTP clients can be configured to connect using FTPS. The lftp command line FTP client has the ability to use FTPS as well.

References

- See the vsftpd website for more information.

File Servers

If you have more than one computer on a single network. At some point you will probably need to share files between them. In this section we cover installing and configuring FTP, NFS, and CUPS.

Domain Name Service (DNS)

TBD

iSCSI Initiator (or Client)

> Wikipedia iSCSI Definition:
>
> iSCSI an acronym for **Internet Small Computer Systems Interface** , an Internet Protocol (IP)-based storage networking standard for linking data storage facilities. It provides block-level access to storage devices by carrying SCSI commands over a TCP/IP network.
>
> iSCSI is used to facilitate data transfers over intranets and to manage storage over long distances. It can be used to transmit data over local area networks (LANs), wide area networks (WANs), or the Internet and can enable location-independent data storage and retrieval.
>
> The protocol allows clients (called *initiators*) to send SCSI commands (*CDBs*) to storage devices (*targets*) on remote servers. It is a storage area network (SAN) protocol, allowing organizations to consolidate storage into storage arrays while providing clients (such as database and web servers) with the illusion of locally attached SCSI disks.
>
> It mainly competes with Fibre Channel, but unlike traditional Fibre Channel, which usually requires dedicated cabling, iSCSI can be run over long distances using existing network infrastructure.

Ubuntu Server can be configured as both: **iSCSI initiator** and **iSCSI target**. This guide provides commands and configuration options to setup an **iSCSI initiator** (or Client).

Note: It is assumed that ***you already have an iSCSI target on your local network*** *and have the appropriate rights to connect to it. The instructions for setting up a target vary greatly between hardware providers, so consult your vendor documentation to configure your specific iSCSI target.*

Network Interfaces Configuration

Before start configuring iSCSI, make sure to have the network interfaces correctly set and configured in order to have open-iscsi package to behave appropriately, specially during boot time. In Ubuntu 20.04 LTS, the default network configuration tool is netplan.io.

For all the iSCSI examples bellow please consider the following netplan configuration for my iSCSI initiator:

/etc/cloud/cloud.cfg.d/99-disable-network-config.cfg

```
{config: disabled}
```

/etc/netplan/50-cloud-init.yaml

```
network:
    ethernets:
        enp5s0:
            match:
                macaddress: 00:16:3e:af:c4:d6
            set-name: eth0
            dhcp4: true
            dhcp-identifier: mac
        enp6s0:
            match:
                macaddress: 00:16:3e:50:11:9c
            set-name: iscsi01
            dhcp4: true
            dhcp-identifier: mac
            dhcp4-overrides:
              route-metric: 300
        enp7s0:
            match:
                macaddress: 00:16:3e:b3:cc:50
            set-name: iscsi02
            dhcp4: true
            dhcp-identifier: mac
            dhcp4-overrides:
              route-metric: 300
    version: 2
    renderer: networkd
```

With this configuration, the interfaces names change by matching their mac addresses. This makes it easier to manage them in a server containing multiple interfaces.

From this point and beyond, 2 interfaces are going to be mentioned: **iscsi01** and **iscsi02**. This helps to demonstrate how to configure iSCSI in a multipath environment as well (check the Device Mapper Multipath session in this same Server Guide).

> If you have only a single interface for the iSCSI network, make sure to follow the same instructions, but only consider the **iscsi01** interface command line examples.

iSCSI Initiator Install

To configure Ubuntu Server as an iSCSI initiator install the open-iscsi package. In a terminal enter:

```
$ sudo apt install open-iscsi
```

Once the package is installed you will find the following files:

- /etc/iscsi/iscsid.conf
- /etc/iscsi/initiatorname.iscsi

iSCSI Initiator Configuration

Configure the main configuration file like the example bellow:

/etc/iscsi/iscsid.conf

```
### startup settings

## will be controlled by systemd, leave as is
iscsid.startup = /usr/sbin/iscsidnode.startup = manual

### chap settings

# node.session.auth.authmethod = CHAP

## authentication of initiator by target (session)
# node.session.auth.username = username
# node.session.auth.password = password

# discovery.sendtargets.auth.authmethod = CHAP

## authentication of initiator by target (discovery)
# discovery.sendtargets.auth.username = username
# discovery.sendtargets.auth.password = password

### timeouts

## control how much time iscsi takes to propagate an error to the
## upper layer. if using multipath, having 0 here is desirable
## so multipath can handle path errors as quickly as possible
## (and decide to queue or not if missing all paths)
node.session.timeo.replacement_timeout = 0

node.conn[0].timeo.login_timeout = 15
node.conn[0].timeo.logout_timeout = 15

## interval for a NOP-Out request (a ping to the target)
node.conn[0].timeo.noop_out_interval = 5

## and how much time to wait before declaring a timeout
node.conn[0].timeo.noop_out_timeout = 5

## default timeouts for error recovery logics (lu & tgt resets)
node.session.err_timeo.abort_timeout = 15
node.session.err_timeo.lu_reset_timeout = 30
node.session.err_timeo.tgt_reset_timeout = 30

### retry

node.session.initial_login_retry_max = 8
```

```
### session and device queue depth

node.session.cmds_max = 128
node.session.queue_depth = 32

### performance

node.session.xmit_thread_priority = -20
```

and re-start the iSCSI daemon:

```
$ systemctl restart iscsid.service
```

This will set basic things up for the rest of configuration.

The other file mentioned:

> /etc/iscsi/initiatorname.iscsi
>
> ```
> InitiatorName=iqn.1993-08.org.debian:01:60f3517884c3
> ```
>
> contains this node's initiator name and is generated during open-iscsi package installation. If you modify this setting, make sure that you don't have duplicates in the same iSCSI SAN (Storage Area Network).

iSCSI Network Configuration

Before configuring the Logical Units that are going to be accessed by the initiator, it is important to inform the iSCSI service what are the interfaces acting as paths.

A straightforward way to do that is by:

- configuring the following environment variables

```
$ iscsi01_ip=$(ip -4 -o addr show iscsi01 | sed -r 's:.* (([0-9]{1,3}\.)
    {3}[0-9]{1,3})/.*:\1:')
$ iscsi02_ip=$(ip -4 -o addr show iscsi02 | sed -r 's:.* (([0-9]{1,3}\.)
    {3}[0-9]{1,3})/.*:\1:')

$ iscsi01_mac=$(ip -o link show iscsi01 | sed -r 's:.*\s+link/ether (([0-f
    ]{2}(\:|)){6}).*:\1:g')
$ iscsi02_mac=$(ip -o link show iscsi02 | sed -r 's:.*\s+link/ether (([0-f
    ]{2}(\:|)){6}).*:\1:g')
```

- configuring **iscsi01** interface

```
$ sudo iscsiadm -m iface -I iscsi01 --op=new
New interface iscsi01 added
$ sudo iscsiadm -m iface -I iscsi01 --op=update -n iface.hwaddress -v
    $iscsi01_mac
iscsi01 updated.
$ sudo iscsiadm -m iface -I iscsi01 --op=update -n iface.ipaddress -v
    $iscsi01_ip
iscsi01 updated.
```

- configuring **iscsi02** interface

```
$ sudo iscsiadm -m iface -I iscsi02 --op=new
New interface iscsi02 added
$ sudo iscsiadm -m iface -I iscsi02 --op=update -n iface.hwaddress -v
    $iscsi02_mac
iscsi02 updated.
$ sudo iscsiadm -m iface -I iscsi02 --op=update -n iface.ipaddress -v
    $iscsi02_ip
iscsi02 updated.
```

- discovering the **targets**

```
$ sudo iscsiadm -m discovery -I iscsi01 --op=new --op=del --type sendtargets
    --portal storage.iscsi01
10.250.94.99:3260,1 iqn.2003-01.org.linux-iscsi.storage.x8664:sn.2c084c8320ca

$ sudo iscsiadm -m discovery -I iscsi02 --op=new --op=del --type sendtargets
    --portal storage.iscsi02
10.250.93.99:3260,1 iqn.2003-01.org.linux-iscsi.storage.x8664:sn.2c084c8320ca
```

- configuring **automatic login**

```
$ sudo iscsiadm -m node --op=update -n node.conn[0].startup -v automatic
$ sudo iscsiadm -m node --op=update -n node.startup -v automatic
```

- make sure needed **services** are enabled during OS initialization:

```
$ systemctl enable open-iscsi
Synchronizing state of open-iscsi.service with SysV service script with /lib/
    systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable open-iscsi
Created symlink /etc/systemd/system/iscsi.service → /lib/systemd/system/open-
    iscsi.service.
Created symlink /etc/systemd/system/sysinit.target.wants/open-iscsi.service
    → /lib/systemd/system/open-iscsi.service.

$ systemctl enable iscsid
Synchronizing state of iscsid.service with SysV service script with /lib/
    systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable iscsid
Created symlink /etc/systemd/system/sysinit.target.wants/iscsid.service → /
    lib/systemd/system/iscsid.service.
```

- restarting **iscsid** service

```
$ systemctl restart iscsid.service
```

- and, finally, **login in** discovered logical units

```
$ sudo iscsiadm -m node --loginall=automatic
Logging in to [iface: iscsi02, target: iqn.2003-01.org.linux-iscsi.storage.
    x8664:sn.2c084c8320ca, portal: 10.250.93.99,3260] (multiple)
Logging in to [iface: iscsi01, target: iqn.2003-01.org.linux-iscsi.storage.
    x8664:sn.2c084c8320ca, portal: 10.250.94.99,3260] (multiple)
Login to [iface: iscsi02, target: iqn.2003-01.org.linux-iscsi.storage.x8664:sn
    .2c084c8320ca, portal: 10.250.93.99,3260] successful.
Login to [iface: iscsi01, target: iqn.2003-01.org.linux-iscsi.storage.x8664:sn
    .2c084c8320ca, portal: 10.250.94.99,3260] successful.
```

Accessing the Logical Units (or LUNs)

Check dmesg to make sure that the new disks have been detected:

dmesg

```
[  166.840694] scsi 7:0:0:4: Direct-Access     LIO-ORG  TCMU device >
          0002 PQ: 0 ANSI: 5
[  166.840892] scsi 8:0:0:4: Direct-Access     LIO-ORG  TCMU device >
          0002 PQ: 0 ANSI: 5
[  166.841741] sd 7:0:0:4: Attached scsi generic sg2 type 0
[  166.841808] sd 8:0:0:4: Attached scsi generic sg3 type 0
[  166.842278] scsi 7:0:0:3: Direct-Access     LIO-ORG  TCMU device >
          0002 PQ: 0 ANSI: 5
[  166.842571] scsi 8:0:0:3: Direct-Access     LIO-ORG  TCMU device >
          0002 PQ: 0 ANSI: 5
[  166.843482] sd 8:0:0:3: Attached scsi generic sg4 type 0
[  166.843681] sd 7:0:0:3: Attached scsi generic sg5 type 0
[  166.843706] sd 8:0:0:4: [sdd] 2097152 512-byte logical blocks: >
   (1.07 GB/1.00 GiB)
[  166.843884] scsi 8:0:0:2: Direct-Access     LIO-ORG  TCMU device >
          0002 PQ: 0 ANSI: 5
[  166.843971] sd 8:0:0:4: [sdd] Write Protect is off
[  166.843972] sd 8:0:0:4: [sdd] Mode Sense: 2f 00 00 00
[  166.844127] scsi 7:0:0:2: Direct-Access     LIO-ORG  TCMU device >
          0002 PQ: 0 ANSI: 5
[  166.844232] sd 7:0:0:4: [sdc] 2097152 512-byte logical blocks: >
   (1.07 GB/1.00 GiB)
[  166.844421] sd 8:0:0:4: [sdd] Write cache: enabled, read cache: >
   enabled, doesn't support DPO or FUA
[  166.844566] sd 7:0:0:4: [sdc] Write Protect is off
[  166.844568] sd 7:0:0:4: [sdc] Mode Sense: 2f 00 00 00
[  166.844846] sd 8:0:0:2: Attached scsi generic sg6 type 0
[  166.845147] sd 7:0:0:4: [sdc] Write cache: enabled, read cache: >
   enabled, doesn't support DPO or FUA
[  166.845188] sd 8:0:0:4: [sdd] Optimal transfer size 65536 bytes
[  166.845527] sd 7:0:0:2: Attached scsi generic sg7 type 0
[  166.845678] sd 8:0:0:3: [sde] 2097152 512-byte logical blocks: >
   (1.07 GB/1.00 GiB)
[  166.845785] scsi 8:0:0:1: Direct-Access     LIO-ORG  TCMU device >
          0002 PQ: 0 ANSI: 5
[  166.845799] sd 7:0:0:4: [sdc] Optimal transfer size 65536 bytes
[  166.845931] sd 8:0:0:3: [sde] Write Protect is off
[  166.845933] sd 8:0:0:3: [sde] Mode Sense: 2f 00 00 00
[  166.846424] scsi 7:0:0:1: Direct-Access     LIO-ORG  TCMU device >
          0002 PQ: 0 ANSI: 5
[  166.846552] sd 8:0:0:3: [sde] Write cache: enabled, read cache: >
   enabled, doesn't support DPO or FUA
[  166.846708] sd 7:0:0:3: [sdf] 2097152 512-byte logical blocks: >
   (1.07 GB/1.00 GiB)
[  166.847024] sd 8:0:0:1: Attached scsi generic sg8 type 0
[  166.847029] sd 7:0:0:3: [sdf] Write Protect is off
[  166.847031] sd 7:0:0:3: [sdf] Mode Sense: 2f 00 00 00
[  166.847043] sd 8:0:0:3: [sde] Optimal transfer size 65536 bytes
[  166.847133] sd 8:0:0:2: [sdg] 2097152 512-byte logical blocks: >
```

```
    (1.07 GB/1.00 GiB)
[  166.849212] sd 8:0:0:2: [sdg] Write Protect is off
[  166.849214] sd 8:0:0:2: [sdg] Mode Sense: 2f 00 00 00
[  166.849711] sd 7:0:0:3: [sdf] Write cache: enabled, read cache: >
    enabled, doesn't support DPO or FUA
[  166.849718] sd 7:0:0:1: Attached scsi generic sg9 type 0
[  166.849721] sd 7:0:0:2: [sdh] 2097152 512-byte logical blocks: >
    (1.07 GB/1.00 GiB)
[  166.853296] sd 8:0:0:2: [sdg] Write cache: enabled, read cache: >
    enabled, doesn't support DPO or FUA
[  166.853721] sd 8:0:0:2: [sdg] Optimal transfer size 65536 bytes
[  166.853810] sd 7:0:0:2: [sdh] Write Protect is off
[  166.853812] sd 7:0:0:2: [sdh] Mode Sense: 2f 00 00 00
[  166.854026] sd 7:0:0:3: [sdf] Optimal transfer size 65536 bytes
[  166.854431] sd 7:0:0:2: [sdh] Write cache: enabled, read cache: >
    enabled, doesn't support DPO or FUA
[  166.854625] sd 8:0:0:1: [sdi] 2097152 512-byte logical blocks: >
    (1.07 GB/1.00 GiB)
[  166.854898] sd 8:0:0:1: [sdi] Write Protect is off
[  166.854900] sd 8:0:0:1: [sdi] Mode Sense: 2f 00 00 00
[  166.855022] sd 7:0:0:2: [sdh] Optimal transfer size 65536 bytes
[  166.855465] sd 8:0:0:1: [sdi] Write cache: enabled, read cache: >
    enabled, doesn't support DPO or FUA
[  166.855578] sd 7:0:0:1: [sdj] 2097152 512-byte logical blocks: >
    (1.07 GB/1.00 GiB)
[  166.855845] sd 7:0:0:1: [sdj] Write Protect is off
[  166.855847] sd 7:0:0:1: [sdj] Mode Sense: 2f 00 00 00
[  166.855978] sd 8:0:0:1: [sdi] Optimal transfer size 65536 bytes
[  166.856305] sd 7:0:0:1: [sdj] Write cache: enabled, read cache: >
    enabled, doesn't support DPO or FUA
[  166.856701] sd 7:0:0:1: [sdj] Optimal transfer size 65536 bytes
[  166.859624] sd 8:0:0:4: [sdd] Attached SCSI disk
[  166.861304] sd 7:0:0:4: [sdc] Attached SCSI disk
[  166.864409] sd 8:0:0:3: [sde] Attached SCSI disk
[  166.864833] sd 7:0:0:3: [sdf] Attached SCSI disk
[  166.867906] sd 8:0:0:2: [sdg] Attached SCSI disk
[  166.868446] sd 8:0:0:1: [sdi] Attached SCSI disk
[  166.871588] sd 7:0:0:1: [sdj] Attached SCSI disk
[  166.871773] sd 7:0:0:2: [sdh] Attached SCSI disk
```

In the output above you will find **8 x SCSI disks** recognized. The storage server is mapping **4 x LUNs** to this node, AND the node has **2 x PATHs** to each LUN. The OS recognizes each path to each device as 1 SCSI device.

> You will find different output depending on the storage server your node is mapping the LUNs from, and the amount of LUNs being mapped as well.

Although not the objective of this session, let's find the 4 mapped LUNs using multipath-tools.

> You will find further details about multipath in "Device Mapper Multipathing" session of this same guide.

```
$ apt-get install multipath-tools

$ sudo multipath -r
```

```
$ sudo multipath -ll
mpathd (360014051a042fb7c41c4249af9f2cfbc) dm-3 LIO-ORG,TCMU device
size=1.0G features='0' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=1 status=active
| `- 7:0:0:4 sde 8:64  active ready running
`-+- policy='service-time 0' prio=1 status=enabled
  `- 8:0:0:4 sdc 8:32  active ready running
mpathc (360014050d6871110232471d8bcd155a3) dm-2 LIO-ORG,TCMU device
size=1.0G features='0' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=1 status=active
| `- 7:0:0:3 sdf 8:80  active ready running
`-+- policy='service-time 0' prio=1 status=enabled
  `- 8:0:0:3 sdd 8:48  active ready running
mpathb (360014051f65c6cb11b74541b703ce1d4) dm-1 LIO-ORG,TCMU device
size=1.0G features='0' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=1 status=active
| `- 7:0:0:2 sdh 8:112 active ready running
`-+- policy='service-time 0' prio=1 status=enabled
  `- 8:0:0:2 sdg 8:96  active ready running
mpatha (36001405b816e24fcab64fb88332a3fc9) dm-0 LIO-ORG,TCMU device
size=1.0G features='0' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=1 status=active
| `- 7:0:0:1 sdj 8:144 active ready running
`-+- policy='service-time 0' prio=1 status=enabled
  `- 8:0:0:1 sdi 8:128 active ready running
```

Now it is much easier to understand each recognized SCSI device and common paths to same LUNs in the storage server. With the output above one can easily see that:

- **mpatha device** (/dev/mapper/mpatha) is a multipath device for:
  - /dev/sdj
  - /dev/dsi
- **mpathb device** (/dev/mapper/mpathb) is a multipath device for:
  - /dev/sdh
  - /dev/dsg
- **mpathc device** (/dev/mapper/mpathc) is a multipath device for:
  - /dev/sdf
  - /dev/sdd
- **mpathd device** (/dev/mapper/mpathd) is a multipath device for:
  - /dev/sde
  - /dev/sdc

  Do not use this in production without checking appropriate multipath configuration options in the **Device Mapper Multipathing** session. The *default multipath configuration* is less than optimal for regular usage.

Finally, to access the LUN (or remote iSCSI disk) you will:

- If accessing through a single network interface:
  - access it through /dev/sdX where X is a letter given by the OS
- If accessing through multiple network interfaces:
  - configure multipath and access the device through /dev/mapper/X

For everything else, the created devices are block devices and all commands used with local disks should work the same way:

- Creating a partition:

```
$ sudo fdisk /dev/mapper/mpatha

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x92c0322a.

Command (m for help): p
Disk /dev/mapper/mpatha: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 65536 bytes
Disklabel type: dos
Disk identifier: 0x92c0322a

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-2097151, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2097151, default 2097151):

Created a new partition 1 of type 'Linux' and of size 1023 MiB.

Command (m for help): w
The partition table has been altered.
```

- Creating a filesystem:

```
$ sudo mkfs.ext4 /dev/mapper/mpatha-part1
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 261888 4k blocks and 65536 inodes
Filesystem UUID: cdb70b1e-c47c-47fd-9c4a-03db6f038988
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

- Mounting the block device:

```
$ sudo mount /dev/mapper/mpatha-part1 /mnt
```

- Accessing the data:

```
$ ls /mnt
lost+found
```

Make sure to read other important sessions in Ubuntu Server Guide to follow up with concepts explored in this one.

References

1. iscsid
2. iscsi.conf
3. iscsid.conf
4. iscsi.service
5. iscsid.service
6. Open-iSCSI
7. Debian Open-iSCSI

Kerberos

Kerberos is a network authentication system based on the principal of a trusted third party. The other two parties being the user and the service the user wishes to authenticate to. Not all services and applications can use Kerberos, but for those that can, it brings the network environment one step closer to being Single Sign On (SSO).

This section covers installation and configuration of a Kerberos server, and some example client configurations.

Overview

If you are new to Kerberos there are a few terms that are good to understand before setting up a Kerberos server. Most of the terms will relate to things you may be familiar with in other environments:

- *Principal:* any users, computers, and services provided by servers need to be defined as Kerberos Principals.
- *Instances:* are used for service principals and special administrative principals.
- *Realms:* the unique realm of control provided by the Kerberos installation. Think of it as the domain or group your hosts and users belong to. Convention dictates the realm should be in uppercase. By default, ubuntu will use the DNS domain converted to uppercase (EXAMPLE.COM) as the realm.
- *Key Distribution Center:* (KDC) consist of three parts: a database of all principals, the authentication server, and the ticket granting server. For each realm there must be at least one KDC.
- *Ticket Granting Ticket:* issued by the Authentication Server (AS), the Ticket Granting Ticket (TGT) is encrypted in the user's password which is known only to the user and the KDC.
- *Ticket Granting Server:* (TGS) issues service tickets to clients upon request.
- *Tickets:* confirm the identity of the two principals. One principal being a user and the other a service requested by the user. Tickets establish an encryption key used for secure communication during the authenticated session.
- *Keytab Files:* are files extracted from the KDC principal database and contain the encryption key for a service or host.

To put the pieces together, a Realm has at least one KDC, preferably more for redundancy, which contains a database of Principals. When a user principal logs into a workstation that is configured for Kerberos authentication, the KDC issues a Ticket Granting Ticket (TGT). If the user supplied credentials match, the user is authenticated and can then request tickets for Kerberized services from the Ticket Granting Server (TGS). The service tickets allow the user to authenticate to the service without entering another username and password.

Kerberos Server

Installation

For this discussion, we will create a MIT Kerberos domain with the following features (edit them to fit your needs):

- *Realm:* EXAMPLE.COM
- *Primary KDC:* kdc01.example.com (192.168.0.1)
- *Secondary KDC:* kdc02.example.com (192.168.0.2)
- *User principal:* ubuntu
- *Admin principal:* ubuntu/admin

  Note

  It is *strongly* recommended that your network-authenticated users have their uid in a different range (say, starting at 5000) than that of your local users.

Before installing the Kerberos server a properly configured DNS server is needed for your domain. Since the Kerberos Realm by convention matches the domain name, this section uses the *EXAMPLE.COM* domain configured in the section Primary Server of the DNS documentation.

Also, Kerberos is a time sensitive protocol. So if the local system time between a client machine and the server differs by more than five minutes (by default), the workstation will not be able to authenticate. To correct the problem all hosts should have their time synchronized using the same *Network Time Protocol (NTP)* server. Check out the NTP chapter for more details.

The first step in creating a Kerberos Realm is to install the krb5-kdc and krb5-admin-server packages. From a terminal enter:

```
sudo apt install krb5-kdc krb5-admin-server
```

You will be asked at the end of the install to supply the hostname for the Kerberos and Admin servers, which may or may not be the same server, for the realm. Since we are going to create the realm, and thus these servers, type in the full hostname of this server.

Note

By default the realm is created from the KDC's domain name.

Next, create the new realm with the kdb5_newrealm utility:

```
sudo krb5_newrealm
```

It will ask you for a database master password, which is used to encrypt the local database. Chose a secure password: its strength is not verified for you.

Configuration

The questions asked during installation are used to configure the /etc/krb5.conf and /etc/krb5kdc/kdc.conf files. The former is used by the kerberos 5 libraries, and the latter configures the KDC. If you need to adjust the Key Distribution Center (KDC) settings simply edit the file and restart the krb5-kdc daemon. If you need to reconfigure Kerberos from scratch, perhaps to change the realm name, you can do so by typing

```
sudo dpkg-reconfigure krb5-kdc
```

Note

The manpage for krb5.conf is in the krb5−doc package.

Once the KDC is properly running, an admin user – the *admin principal* – is needed. It is recommended to use a different username from your everyday username. Using the kadmin.local utility in a terminal prompt enter:

```
$ sudo kadmin.local
Authenticating as principal root/admin@EXAMPLE.COM with password.
kadmin.local: addprinc ubuntu/admin
WARNING: no policy specified for ubuntu/admin@EXAMPLE.COM; defaulting to no
    policy
Enter password for principal "ubuntu/admin@EXAMPLE.COM":
Re−enter password for principal "ubuntu/admin@EXAMPLE.COM":
Principal "ubuntu/admin@EXAMPLE.COM" created.
kadmin.local: quit
```

In the above example *ubuntu* is the *Principal*, */admin* is an *Instance* of tha principal, and *@EXAMPLE.COM* signifies the realm. The *"every day"* Principal, a.k.a. the *user principal*, would be *ubuntu@EXAMPLE.COM*, and should have only normal user rights.

Note

Replace *EXAMPLE.COM* and *ubuntu* with your Realm and admin username.

Next, the new admin user needs to have the appropriate Access Control List (ACL) permissions. The permissions are configured in the /etc/krb5kdc/kadm5.acl file:

```
ubuntu/admin@EXAMPLE.COM        *
```

This entry grants *ubuntu/admin* the ability to perform any operation on all principals in the realm. You can configure principals with more restrictive privileges, which is convenient if you need an admin principal that junior staff can use in Kerberos clients. Please see the *kadm5.acl* man page for details.

Note

The *extract* privilege is not included in the wildcard privilege; it must be explicitly assigned. his privilege allows the user to extract keys from the database, and must be handled with great care to avoid disclosure of important keys like those of the kadmin/* or krbtgt/* principals. See the *kadm5.acl* man page for details.

Now restart the krb5-admin-server for the new ACL to take affect:

```
sudo systemctl restart krb5−admin−server.service
```

The new user principal can be tested using the kinit utility:

```
$ kinit ubuntu/admin
Password for ubuntu/admin@EXAMPLE.COM:
```

After entering the password, use the klist utility to view information about the Ticket Granting Ticket (TGT):

```
$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: ubuntu/admin@EXAMPLE.COM

Valid starting     Expires            Service principal
04/03/20 19:16:57  04/04/20 05:16:57  krbtgt/EXAMPLE.COM@EXAMPLE.COM
        renew until 04/04/20 19:16:55
```

Where the cache filename krb5cc_1000 is composed of the prefix krb5cc_ and the user id (uid), which in this case is 1000.

kinit will inspect /etc/krb5.conf to find out which KDC to contact, and its address. The KDC can also be found via DNS lookups for special TXT and SRV records. You can add these records to your example.com DNS zone:

```
_kerberos._udp.EXAMPLE.COM.       IN SRV 1   0 88   kdc01.example.com.
_kerberos._tcp.EXAMPLE.COM.       IN SRV 1   0 88   kdc01.example.com.
_kerberos._udp.EXAMPLE.COM.       IN SRV 10  0 88   kdc02.example.com.
_kerberos._tcp.EXAMPLE.COM.       IN SRV 10  0 88   kdc02.example.com.
_kerberos-adm._tcp.EXAMPLE.COM.   IN SRV 1   0 749  kdc01.example.com.
_kpasswd._udp.EXAMPLE.COM.        IN SRV 1   0 464  kdc01.example.com.
```

Note

Replace *EXAMPLE.COM*, *kdc01*, and *kdc02* with your domain name, primary KDC, and secondary KDC.

See the DNS chapter for detailed instructions on setting up DNS.

A very quick and useful way to troubleshoot what kinit is doing is to set the environment variable KRB5_TRACE to a file, or stderr, and it will show extra information. The output is quite verbose, and won't be shown fully here:

```
$ KRB5_TRACE=/dev/stderr kinit ubuntu/admin
[2898] 1585941845.278578: Getting initial credentials for ubuntu/admin@EXAMPLE
    .COM
[2898] 1585941845.278580: Sending unauthenticated request
[2898] 1585941845.278581: Sending request (189 bytes) to EXAMPLE.COM
[2898] 1585941845.278582: Resolving hostname kdc01.example.com
(...)
```

Your new Kerberos Realm is now ready to authenticate clients.

Secondary KDC

Once you have one Key Distribution Center (KDC) on your network, it is good practice to have a Secondary KDC in case the primary becomes unavailable. Also, if you have Kerberos clients that are in different networks (possibly separated by routers using NAT), it is wise to place a secondary KDC in each of those networks.

Note

The native replication mechanism explained here relies on a cronjob, and essentially dumps the DB on the primary and loads it back up on the secondary. You may want to take a look at using the *kldap* backend which can use the OpenLDAP replication mechanism. It is explained further below.

First, install the packages, and when asked for the Kerberos and Admin server names enter the name of the Primary KDC:

```
sudo apt install krb5-kdc krb5-admin-server
```

Once you have the packages installed, create the host principals for both KDCs. From a terminal prompt, enter:

```
kadmin -q "addprinc -randkey host/kdc01.example.com"
kadmin -q "addprinc -randkey host/kdc02.example.com"
```

Note

The kadmin command defaults to using a principal like *username/admin@EXAMPLE.COM*, where *username* is your current shell user. If you need to override that, use −p <principal−you −want>

Make sure the principal you are using has the extra extract−keys privilege in kdc01's /etc/krb5kdc/kadm5.acl file. Something like this:

```
ubuntu/admin@EXAMPLE.COM *e
```

Where "*" means all privileges (except extract−keys), and e means exactly extract−keys.

Extract the *keytab* file:

```
kadmin -q "ktadd -norandkey -k keytab.kdc02 host/kdc02.example.com"
```

There should now be a keytab.kdc02 in the current directory, move the file to /etc/krb5.keytab:

```
sudo mv keytab.kdc02 /etc/krb5.keytab
sudo chown root:root /etc/krb5.keytab
```

Note

If the path to the keytab.kdc02 file is different adjust accordingly.

Also, you can list the principals in a Keytab file, which can be useful when troubleshooting, using the klist utility:

```
sudo klist -k /etc/krb5.keytab
```

The -k option indicates the file is a keytab file.

Next, there needs to be a kpropd.acl file on each KDC that lists all KDCs for the Realm. For example, on both primary and secondary KDC, create /etc/krb5kdc/kpropd.acl:

```
host/kdc01.example.com@EXAMPLE.COM
host/kdc02.example.com@EXAMPLE.COM
```

Create an empty database on the *Secondary KDC*:

```
sudo kdb5_util -s create
```

Now install kpropd daemon, which listens for connections from the kprop utility from the primary kdc:

```
sudo apt install krb5−kpropd
```

The service will be running right after installation.

From a terminal on the *Primary KDC*, create a dump file of the principal database:

```
sudo kdb5_util dump /var/lib/krb5kdc/dump
```

Still on the *Primary KDC*, extract its *keytab* file and copy it to /etc/krb5.keytab:

```
kadmin -q "ktadd -k keytab.kdc01 host/kdc01.example.com"
sudo mv keytab.kdc01 /etc/krb5.keytab
sudo chown root:root /etc/krb5.keytab
```

Note

You can now remove the extract−keys privilege from this principal in kdc01's /etc/krb5kdc/kadm5.acl file

On the *Primary KDC*, run the kprop utility to push the database dump made before to the Secondary KDC:

```
$ sudo kprop -r EXAMPLE.COM -f /var/lib/krb5kdc/dump kdc02.example.com
Database propagation to kdc02.example.com: SUCCEEDED
```

Note the *SUCCEEDED* message, which signals that the propagation worked. If there is an error message check /var/log/syslog on the secondary KDC for more information.

You may also want to create a cron job to periodically update the database on the Secondary KDC. For example, the following will push the database every hour:

```
# m h  dom mon dow   command
0 * * * * root /usr/sbin/kdb5_util dump /var/lib/krb5kdc/dump && /usr/sbin/
    kprop -r EXAMPLE.COM -f /var/lib/krb5kdc/dump kdc02.example.com
```

Back on the *Secondary KDC*, create a *stash* file to hold the Kerberos master key:

```
sudo kdb5_util stash
```

Finally, start the krb5-kdc daemon on the Secondary KDC:

```
sudo systemctl start krb5-kdc.service
```

> **Note**
>
> The *Secondary KDC* does not run an admin server, since it's a read-only copy

From now on, you can specify both KDC servers in /etc/krb5.conf for the EXAMPLE.COM realm, in any host participating in this realm (including kdc01 and kdc02), but remember that there can only be one admin server and that's the one running on kdc01:

```
[realms]
    EXAMPLE.COM = {
            kdc = kdc01.example.com
            kdc = kdc02.example.com
            admin_server = kdc01.example.com
    }
```

The *Secondary KDC* should now be able to issue tickets for the Realm. You can test this by stopping the krb5-kdc daemon on the Primary KDC, then by using kinit to request a ticket. If all goes well you should receive a ticket from the Secondary KDC. Otherwise, check /var/log/syslog and /var/log/auth.log in the Secondary KDC.

Kerberos Linux Client

This section covers configuring a Linux system as a Kerberos client. This will allow access to any kerberized services once a user has successfully logged into the system.

Note that Kerberos alone is not enough for a user to exist in a Linux system. Meaning, we cannot just point the system at a kerberos server and expect all the kerberos principals to be able to *login* on the linux system, simply because these users do not *exist* locally. Kerberos only provides authentication: it doesn't know about user groups, Linux uids and gids, home directories, etc. Normally another network source is used for this information, such as an LDAP or Windows server, and, in the old days, NIS was used for that as well.

Installation

If you have local users matching the principals in a Kerberos realm, and just want to switch the authentication from local to remote using Kerberos, you can follow this section. This is not a very usual scenario, but serves to highlight the separation between user authentication and user information (full name, uid, gid, home directory, groups, etc). If you just want to be able to grab tickets and use them, it's enough to install krb5-user and run kinit.

We are going to use sssd with a trick so that it will fetch the user information from the local system files, instead of a remote source which is the common case.

To install the packages enter the following in a terminal prompt:

```
sudo apt install krb5-user sssd-krb5
```

You will be prompted for the addresses of your KDCs and admin servers. If you have been following this chapter so far, the KDCs will be: kdc01.example.com kdc02.example.com (space separated)

And the admin server will be: kdc01.example.com. Remember that kdc02 is a read-only copy of the primary KDC, so it doesn't run an admin server.

> **Note**
>
> If you have added the appropriate SRV records to DNS, none of those prompts will need answering.

Configuration

If you missed the questions earlier, you can reconfigure the package to fill them in again: sudo dpkg-reconfigure krb5-config.

You can test the kerberos configuration by requesting a ticket using the kinit utility. For example:

```
$ kinit ubuntu/admin@EXAMPLE.COM
Password for ubuntu/admin@EXAMPLE.COM:
```

> **Note**
>
> kinit doesn't need for the principal to exist as a local user in the system. In fact, you can kinit any principal you want. If you don't specify one, then the tool will use the username of whoever is running kinit.

Since we are at it, let's also create a non-admin principal for ubuntu:

```
$ kadmin -q "addprinc ubuntu"
Authenticating as principal ubuntu/admin@EXAMPLE.COM with password.
Password for ubuntu/admin@EXAMPLE.COM:
WARNING: no policy specified for ubuntu@EXAMPLE.COM; defaulting to no policy
Enter password for principal "ubuntu@EXAMPLE.COM":
Re-enter password for principal "ubuntu@EXAMPLE.COM":
Principal "ubuntu@EXAMPLE.COM" created.
```

The only remaining configuration now is for sssd. Create the file /etc/sssd/sssd.conf with the following content:

```
[sssd]
config_file_version = 2
services = pam
domains = example.com
```

```
[pam]

[domain/example.com]
id_provider = proxy
proxy_lib_name = files
auth_provider = krb5
krb5_server = kdc01.example.com,kdc01.example.com
krb5_kpasswd = kdc01.example.com
krb5_realm = EXAMPLE.COM
```

The above configuration will use kerberos for *authentication* (auth_provider), but will use the local system users for user and group information (id_provider).

Adjust the permissions of the config file and start sssd:

```
$ sudo chown root:root /etc/sssd/sssd.conf
$ sudo chmod 0600 /etc/sssd/sssd.conf
$ sudo systemctl start sssd
```

Just by having installed sssd and its dependencies, PAM will already have been configured to use sssd, with a fallback to local user authentication. To try it out, if this is a workstation, simply switch users (in the GUI), or open a login terminal (CTRL-ALT-<number>), or spawn a login shell with sudo login, and try logging in using the name of a kerberos principal. Remember that this user must already exist on the local system:

```
$ sudo login
focal-krb5-client login: ubuntu
Password:
Welcome to Ubuntu Focal Fossa (development branch) (GNU/Linux 5.4.0-21-generic
    x86_64)

(...)

Last login: Thu Apr  9 21:23:50 UTC 2020 from 10.20.20.1 on pts/0
$ klist
Ticket cache: FILE:/tmp/krb5cc_1000_NlfnSX
Default principal: ubuntu@EXAMPLE.COM

Valid starting     Expires            Service principal
04/09/20 21:36:12  04/10/20 07:36:12  krbtgt/EXAMPLE.COM@EXAMPLE.COM
    renew until 04/10/20 21:36:12
```

And you will have a Kerberos ticket already right after login.

Resources

- For more information on MIT's version of Kerberos, see the MIT Kerberos site.
- The Ubuntu Wiki Kerberos page has more details.
- O'Reilly's Kerberos: The Definitive Guide is a great reference when setting up Kerberos.
- Also, feel free to stop by the *#ubuntu-server* and *#kerberos* IRC channels on Freenode if you have Kerberos questions.

Kerberos and LDAP

Kerberos supports a few database backends. The default one is what we have been using so far, called *db2*. The DB Types documentation shows all the options, one of which is LDAP.

There are several reasons why one would want to have the Kerberos principals stored in LDAP as opposed to a local on-disk database. There are also cases when it is not a good idea. Each site has to evaluate the pros and cons. Here are a few: - the OpenLDAP replication is faster and more robust then the native Kerberos one, based on a cron job - setting things up with the LDAP backend isn't exactly trivial and shouldn't be attempted by administrators without prior knowledge of OpenLDAP - as highlighted in LDAP section of DB Types, since krb5kdc is single threaded there may be higher latency in servicing requests when using the OpenLDAP backend - if you already have OpenLDAP setup for other things, like storing users and groups, adding the Kerberos attributes to the same mix might be beneficial and can provide a nice integrated story

This section covers configuring a primary and secondary kerberos server to use OpenLDAP for the principal database. Note that as of version 1.18, the KDC from MIT Kerberos does not support a primary KDC using a read-only consumer (secondary) LDAP server. What we have to consider here is that a Primary KDC is read-write, and it needs a read-write backend. The Secondaries can use both a read-write and read-only backend, because they are expected to be read-only. Therefore there are only some possible layouts we can use:

1. Simple case: Primary KDC connected to primary OpenLDAP, Secondary KDC connected to both Primary and Secondary OpenLDAP
2. Extended simple case: Multiple Primary KDCs connected to one Primary OpenLDAP, and multiple Secondary KDCs connected to Primary and Secondary OpenLDAP
3. OpenLDAP with multi-master replication: multiple primary KDCs connected to all primary OpenLDAP servers

We haven't covered OpenLDAP multi-master replication in this guide, so we will show the first case only. The second scenario is an extension: just add another primary KDC to the mix, talking to the same primary OpenLDAP server.

Configuring OpenLDAP

We are going to install the OpenLDAP server on the same host as the KDC, to simplify the communication between them. In such a setup, we can use the *ldapi:///* transport, which is via an unix socket, and don't need to setup SSL certificates to secure the communication between the Kerberos services and OpenLDAP. Note, however, that SSL is still needed for the OpenLDAP replication. See LDAP with TLS for details.

If you want to use an existing OpenLDAP server that you have somewhere else, that's of course also possible, but keep in mind that you should then use SSL for the communication between the KDC and this OpenLDAP server.

First, the necessary *schema* needs to be loaded on an OpenLDAP server that has network connectivity to the Primary and Secondary KDCs. The rest of this section assumes that you also have LDAP replication configured between at least two servers. For information on setting up OpenLDAP see OpenLDAP Server.

> **Note**
>
> cn=admin,dc=example,dc=com is a default admin user that is created during the installation of the slapd package (the OpenLDAP server). The domain component will change for your server, so adjust accordingly.

- Install the necessary packages (it's assumed that OpenLDAP is already installed):

```
sudo apt install krb5-kdc-ldap krb5-admin-server
```

- Next, extract the kerberos.schema.gz file:

```
sudo cp /usr/share/doc/krb5-kdc-ldap/kerberos.schema.gz /etc/ldap/schema/
sudo gunzip /etc/ldap/schema/kerberos.schema.gz
```

- The *kerberos* schema needs to be added to the *cn=config* tree. This schema file needs to be converted to LDIF format before it can be added. For that we will use a helper tool, called schema2ldif, provided by the package of the same name which is available in the Universe archive:

  ```
  sudo apt install schema2ldif
  ```

- To import the kerberos schema, run:

  ```
  $ sudo ldap-schema-manager -i kerberos.schema
  SASL/EXTERNAL authentication started
  SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
  SASL SSF: 0
  executing 'ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/ldap/schema/kerberos.
      ldif'
  SASL/EXTERNAL authentication started
  SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
  SASL SSF: 0
  adding new entry "cn=kerberos,cn=schema,cn=config"
  ```

- With the new schema loaded, let's index an attribute often used in searches:

  ```
  $ sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// <<EOF
  dn: olcDatabase={1}mdb,cn=config
  add: olcDbIndex
  olcDbIndex: krbPrincipalName eq,pres,sub
  EOF

  modifying entry "olcDatabase={1}mdb,cn=config"
  ```

- Let's create LDAP entries for the Kerberos administrative entities that will contact the OpenLDAP server to perform operations. There are two:

  - ***ldap_kdc_dn***: needs to have read rights on the realm container, principal container and realm sub-trees. If ***disable_last_success*** and ***disable_lockout*** are not set, however, then ***ldap_kdc_dn*** needs write access to the kerberos container just like the admin dn below.
  - ***ldap_kadmind_dn***: needs to have read and write rights on the realm container, principal container and realm sub-trees

  Here is the command to create these entities:

  ```
  $ ldapadd -x -D cn=admin,dc=example,dc=com -W <<EOF
  dn: uid=kdc-service,dc=example,dc=com
  uid: kdc-service
  objectClass: account
  objectClass: simpleSecurityObject
  userPassword: {CRYPT}x
  description: Account used for the Kerberos KDC

  dn: uid=kadmin-service,dc=example,dc=com
  uid: kadmin-service
  objectClass: account
  objectClass: simpleSecurityObject
  userPassword: {CRYPT}x
  description: Account used for the Kerberos Admin server
  ```

```
EOF
Enter LDAP Password:
adding new entry "uid=kdc-service,dc=example,dc=com"

adding new entry "uid=kadmin-service,dc=example,dc=com"
```

Now let's set a password for them. Note that first the tool asks for the password you want for the specified user dn, and then for the password of the ***cn=admin*** dn: $ ldappasswd -x -D cn=admin,dc=example,dc=com -W -S uid=kdc-service,dc=example,dc=com New password: <– password you want for uid-kdc-service Re-enter new password: Enter LDAP Password: <– password for the dn specified with the -D option

Repeat for the uid=kadmin-service dn. These passwords will be needed later.

You can test these with ldapwhoami:

```
$ ldapwhoami -x -D uid=kdc-service,dc=example,dc=com -W
Enter LDAP Password:
dn:uid=kdc-service,dc=example,dc=com
```

- Finally, update the Access Control Lists (ACL). These can be tricky, as it highly depends on what you have defined already. By default, the slapd package configures your database with the following ACLs:

```
olcAccess: {0}to attrs=userPassword by self write by anonymous auth by *
    none
olcAccess: {1}to attrs=shadowLastChange by self write by * read
olcAccess: {2}to * by * read
```

We need to insert new rules before the final to * by * read one, to control access to the Kerberos related entries and attributes:

```
$ sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// <<EOF
dn: olcDatabase={1}mdb,cn=config
add: olcAccess
olcAccess: {2}to attrs=krbPrincipalKey
  by anonymous auth
  by dn.exact="uid=kdc-service,dc=example,dc=com" read
  by dn.exact="uid=kadmin-service,dc=example,dc=com" write
  by self write
  by * none
-
add: olcAccess
olcAccess: {3}to dn.subtree="cn=krbContainer,dc=example,dc=com"
  by dn.exact="uid=kdc-service,dc=example,dc=com" read
  by dn.exact="uid=kadmin-service,dc=example,dc=com" write
  by * none
EOF

modifying entry "olcDatabase={1}mdb,cn=config"
```

This will make the existing {2} rule become {4}. Check with sudo slapcat -b cn=config (the output below was reformatted a bit for clarity):

```
olcAccess: {0}to attrs=userPassword
    by self write
    by anonymous auth
    by * none
```

```
olcAccess: {1}to attrs=shadowLastChange
    by self write
    by * read
olcAccess: {2}to attrs=krbPrincipalKey by anonymous auth
    by dn.exact="uid=kdc-service,dc=example,dc=com" read
    by dn.exact="uid=kadmin-service,dc=example,dc=com" write
    by self write
    by * none
olcAccess: {3}to dn.subtree="cn=krbContainer,dc=example,dc=com"
    by dn.exact="uid=kdc-service,dc=example,dc=com" read
    by dn.exact="uid=kadmin-service,dc=example,dc=com" write
    by * none
olcAccess: {4}to * by * read
```

That's it, your LDAP directory is now ready to serve as a Kerberos principal database.

Primary KDC Configuration (LDAP)

With OpenLDAP configured it is time to configure the KDC. In this example we are doing it in the same OpenLDAP server to take advantage of local unix socket communication.

- Reconfigure the krb5-config package if neededd to get a good starting point with /etc/krb5.conf:

  ```
  sudo dpkg-reconfigure krb5-config
  ```

- Now edit /etc/krb5.conf adding the database_module option to the EXAMPLE.COM realm section:

  ```
  [realms]
          EXAMPLE.COM = {
                  kdc = kdc01.example.com
                  kdc = kdc02.example.com
                  admin_server = kdc01.example.com
                  default_domain = example.com
                  database_module = openldap_ldapconf
          }
  ```

  Then also add these new sections:

  ```
  [dbdefaults]
          ldap_kerberos_container_dn = cn=krbContainer,dc=example,dc=com

  [dbmodules]
          openldap_ldapconf = {
                  db_library = kldap

                  # if either of these is false, then the ldap_kdc_dn needs
                      to
                  # have write access
                  disable_last_success = true
                  disable_lockout  = true

                  # this object needs to have read rights on
                  # the realm container, principal container and realm sub-
                      trees
                  ldap_kdc_dn = "uid=kdc-service,dc=example,dc=com"
  ```

```
                # this object needs to have read and write rights on
                # the realm container, principal container and realm sub-
                    trees
                ldap_kadmind_dn = "uid=kadmin-service,dc=example,dc=com"

                ldap_service_password_file = /etc/krb5kdc/service.keyfile
                ldap_servers = ldapi:///
                ldap_conns_per_server = 5
        }
```

- Next, use the kdb5_ldap_util utility to create the realm:

```
$ sudo kdb5_ldap_util -D cn=admin,dc=example,dc=com create -subtrees dc=
    example,dc=com -r EXAMPLE.COM -s -H ldapi:///
Password for "cn=admin,dc=example,dc=com":
Initializing database for realm 'EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:
Re-enter KDC database master key to verify:
```

- Create a stash of the password used to bind to the LDAP server. Run it once for each *ldap_kdc_dn* and *ldap_kadmin_dn*::

```
sudo kdb5_ldap_util -D cn=admin,dc=example,dc=com stashsrvpw -f /etc/
    krb5kdc/service.keyfile uid=kdc-service,dc=example,dc=com
sudo kdb5_ldap_util -D cn=admin,dc=example,dc=com stashsrvpw -f /etc/
    krb5kdc/service.keyfile uid=kadmin-service,dc=example,dc=com
```

  Note

  The /etc/krb5kdc/service.keyfile file now contains clear text versions of the passwords used by the KDC to contact the LDAP server!

- Create a /etc/krb5kdc/kadm5.acl file for the admin server, if you haven't already:

```
*/admin@EXAMPLE.COM          *
```

- Start the Kerberos KDC and admin server:

```
sudo systemctl start krb5-kdc.service krb5-admin-server.service
```

You can now add Kerberos principals to the LDAP database, and they will be copied to any other LDAP servers configured for replication. To add a principal using the kadmin.local utility enter:

```
$ sudo kadmin.local
Authenticating as principal root/admin@EXAMPLE.COM with password.
kadmin.local:  addprinc ubuntu
WARNING: no policy specified for ubuntu@EXAMPLE.COM; defaulting to no policy
Enter password for principal "ubuntu@EXAMPLE.COM":
Re-enter password for principal "ubuntu@EXAMPLE.COM":
Principal "ubuntu@EXAMPLE.COM" created.
kadmin.local:
```

The above will create an ubuntu *principal* with a *dn* of krbPrincipalName=ubuntu@EXAMPLE.COM,cn =EXAMPLE.COM,cn=krbContainer,dc=example,dc=com. Let's say, however, that you already have an user in your directory, and it's in uid=testuser1,ou=People,dc=example,dc=com, how to add the kerberos attributes to it? You use the -x parameter:

```
$ sudo kadmin.local
Authenticating as principal root/admin@EXAMPLE.COM with password.
kadmin.local:  addprinc -x dn=uid=testuser1,ou=People,dc=example,dc=com
    testuser1
WARNING: no policy specified for testuser1@EXAMPLE.COM; defaulting to no
    policy
Enter password for principal "testuser1@EXAMPLE.COM":
Re-enter password for principal "testuser1@EXAMPLE.COM":
Principal "testuser1@EXAMPLE.COM" created.
```

Since the specified *dn* already exists, kadmin.local will just add the required kerberos attributes to this existing entry. If it didn't exist, it would be created from scratch, with just the kerberos attributes, like what happened with the ubuntu example above, but in the specified location.

Both places are visible for kinit, since, when the realm was created with kdb5_ldap_util, the default value for the search scope and base were taken: subtree, and dc=example,dc=com.

Secondary KDC Configuration (LDAP)

The setup of the secondary KDC (and its OpenLDAP replica) is very similar. Once you have the OpenLDAP replication setup, repeat these steps on the secondary: - install *krb5-kdc-ldap*, *ldap-utils*. Do not install *krb5-admin-server*. - load the kerberos schema using schema2ldif - add the index for *krbPrincipalName* - add the ACLs - configure *krb5.conf* in the same way, initially. If you want and if you configured SSL properly, you can add ldaps://kdc01.example.com to the ldap_servers list after ldapi:///, so that the Secondary KDC can have two LDAP backends at its disposal - **DO NOT** run kdb5_ldap_util. There is no need to create the database since it's being replicated from the Primary - copy over the following files from the Primary KDC and place them in the same location on the Secondary: - /etc/krb5kdc/stash - /etc/krb5kdc/service.keyfile - start the KDC: sudo systemctl start krb5-kdc.service

Resources

- Configuring Kerberos with OpenLDAP back-end
- MIT Kerberos backend types

OpenLDAP Server

The Lightweight Directory Access Protocol, or LDAP, is a protocol for querying and modifying a X.500-based directory service running over TCP/IP. The current LDAP version is LDAPv3, as defined in RFC4510, and the implementation used in Ubuntu is OpenLDAP."

The LDAP protocol accesses directories. A common mistake is to call a directory an *LDAP directory*, or *LDAP database*, but it's really so common, and we all know what we are talking about, that it's ok. Here are some key concepts and terms:

- A directory is a tree of data *entries* that is hierarchical in nature and is called the Directory Information Tree (DIT).
- An entry consists of a set of *attributes*.
- An attribute has a *key* (a name/description) and one or more *values*.
- Every attribute must be defined in at least one *objectClass*.

- Attributes and objectclasses are defined in *schemas* (an objectclass is actually considered as a special kind of attribute).
- Each entry has a unique identifier: its *Distinguished Name* (DN or dn). This, in turn, consists of a *Relative Distinguished Name* (RDN) followed by the parent entry's DN.
- The entry's DN is not an attribute. It is not considered part of the entry itself.

  Note

  The terms *object*, *container*, and *node* have certain connotations but they all essentially mean the same thing as *entry*, the technically correct term.

For example, below we have a single entry consisting of 11 attributes where the following is true:

- DN is "cn=John Doe,dc=example,dc=com"
- RDN is "cn=John Doe"
- parent DN is "dc=example,dc=com"

```
dn: cn=John Doe,dc=example,dc=com
cn: John Doe
givenName: John
sn: Doe
telephoneNumber: +1 888 555 6789
telephoneNumber: +1 888 555 1232
mail: john@example.com
manager: cn=Larry Smith,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
```

The above entry is in *LDIF* format (LDAP Data Interchange Format). Any information that you feed into your DIT must also be in such a format. It is defined in RFC2849.

Such a directory accessed via LDAP is good for anything that involves a large number of access requests to a mostly-read, attribute-based (name:value) backend, and that can benefit from a hierarchical structure. Examples include an address book, company directory, a list of email addresses, and a mail server's configuration.

Installation

The installation of slapd will create a minimal working configuration with a top level entry, and an administrator's DN. In particular, it will create a database instance that you can use to store your data. However, the suffix (or base DN) of this instance will be determined from the domain name of the host. If you want something different, you can change it right after the installation when you still don't have any useful data.

Note

This guide will use a database suffix of *dc=example,dc=com*.

Proceed with the install of the server and the main command line utilities:

```
sudo apt install slapd ldap-utils
```

If you want to change your DIT suffix, now would be a good time, because changing it discards your existing one. To change the suffix, run the following command:

```
sudo dpkg-reconfigure slapd
```

To switch your DIT suffix to *dc=example,dc=com*, for example, so you can follow this guide more closely, answer example.com when asked about the DNS domain name.

Throughout this guide we will issue many commands with the LDAP utilities. To save some typing, we can configure the OpenLDAP libraries with certain defaults in /etc/ldap/ldap.conf:

```
BASE dc=example,dc=com
URI ldap://ldap01.example.com
TLS_CACERT /etc/ssl/certs/ca-certificates.crt
```

> **Note**
>
> Adjust for your server name and directory suffix

Post-install Inspection

The packaging of *slapd* is designed to be configured within the service itself by dedicating a separate DIT for that purpose. This allows one to dynamically configure *slapd* without the need to restart the service or edit config files. This configuration database consists of a collection of text-based LDIF files located under /etc/ldap/slapd.d, but these should never be edited directly. This way of working is known by several names: the *slapd-config* method, the RTC method (Real Time Configuration), or the *cn=config* method. You can still use the traditional flat-file method (*slapd.conf*) but it's not going to be covered in this guide.

Right after installation, you will get two databases, or suffixes: one for your data, based on your host's domain (*dc=example,dc=com*), and one for your configuration, with its root at *cn=config*. To change the data on each we need different credentials and access methods:

- *dc=example,dc=com*: the administrative user for this suffix is *cn=admin,dc=example,dc=com* and its password is the one selected during the installation of the *slapd* package
- *cn=config*: the configuration slapd itself is stored under this suffix. Changes to it can be made by the special DN *gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth*. This is how the local system's root user (uid=0/gid=0) is seen by the directory when using SASL EXTERNAL authentication through the *ldapi:///* transport via the */run/slapd/ldapi* unix socket. Essentially what this means is that only the local root user can update the *cn=config* database. More details later.
- This is what the slapd-config DIT looks like via the LDAP protocol (listing only the *dn*s):

```
$ sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b cn=config dn
dn: cn=config

dn: cn=module{0},cn=config

dn: cn=schema,cn=config

dn: cn={0}core,cn=schema,cn=config

dn: cn={1}cosine,cn=schema,cn=config

dn: cn={2}nis,cn=schema,cn=config

dn: cn={3}inetorgperson,cn=schema,cn=config

dn: olcBackend={0}mdb,cn=config

dn: olcDatabase={-1}frontend,cn=config
```

```
dn: olcDatabase={0}config,cn=config

dn: olcDatabase={1}mdb,cn=config
```

Explanation of entries:

- *cn=config*: global settings
- *cn=module{0},cn=config*: a dynamically loaded module
- *cn=schema,cn=config*: contains hard-coded system-level schema
- *cn={0}core,cn=schema,cn=config*: the hard-coded core schema
- *cn={1}cosine,cn=schema,cn=config*: the cosine schema
- *cn={2}nis,cn=schema,cn=config*: the nis schema
- *cn={3}inetorgperson,cn=schema,cn=config*: the inetorgperson schema
- *olcBackend={0}mdb,cn=config*: the 'mdb' backend storage type
- *olcDatabase={-1}frontend,cn=config*: frontend database, default settings for other databases
- *olcDatabase={0}config,cn=config*: slapd configuration database (cn=config)
- *olcDatabase={1}mdb,cn=config*: your database instance (dc=example,dc=com)

- This is what the *dc=example,dc=com* DIT looks like:

```
$ ldapsearch -x -LLL -H ldap:/// -b dc=example,dc=com dn

dn: dc=example,dc=com

dn: cn=admin,dc=example,dc=com
```

Explanation of entries:

- *dc=example,dc=com*: base of the DIT
- *cn=admin,dc=example,dc=com*: administrator (rootDN) for this DIT (set up during package install)

Notice how we used two different authentication mechanisms:

- *-x*: this is called a *simple bind*, and is essentially a plain text authentication. Since no *binddn* was provided (via *-D*), this became an *anonymous* bind. Without *-x*, the default is to use a *SASL* bind.
- *-Y EXTERNAL*: this is using a *SASL* bind (no *-x* was provided), and further specifying the *EXTERNAL* type. Together with *-H ldapi:///*, this uses a local unix socket connection

In both cases we only got the results that the server ACLs allowed us to see, based on who we are. A very handy tool to verify the authentication is *ldapwhoami*:

```
$ ldapwhoami -x
anonymous

$ ldapwhoami -x -D cn=admin,dc=example,dc=com -W
Enter LDAP Password:
dn:cn=admin,dc=example,dc=com
```

When you use simple bind (*-x*) and specify a *binddn* with *-D* as your authentication dn, the server will look for a *userPassword* attribute in that entry, and use that to verify the credentials. In this particular case above, we used the database *rootDN* entry, i.e., the actual administrator, and that is a special case whose password is set in the configuration when the package is installed.

Note

A simple bind without some sort of transport security mechanism is **clear text**, meaning the credentials are transmitted in the clear. You should add TLS support to your OpenLDAP server as soon as possible.

Here are the SASL EXTERNAL examples:

```
$ ldapwhoami -Y EXTERNAL -H ldapi:/// -Q
dn:gidNumber=1000+uidNumber=1000,cn=peercred,cn=external,cn=auth

$ sudo ldapwhoami -Y EXTERNAL -H ldapi:/// -Q
dn:gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
```

When using SASL EXTERNAL via the *ldapi:///* transport, the *binddn* becomes a combination of the *uid* and *gid* of the connecting user, followed by the suffix *cn=peercred,cn=external,cn=auth*. The server ACLs know about this, and grant the local root user complete write access to *cn=config* via the SASL mechanism.

Modifying/Populating your Database

Let's introduce some content to our database. We will add the following:

- a node called *People* (to store users)
- a node called *Groups* (to store groups)
- a group called *miners*
- a user called *john*

Create the following LDIF file and call it add_content.ldif:

```
dn: ou=People,dc=example,dc=com
objectClass: organizationalUnit
ou: People

dn: ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
ou: Groups

dn: cn=miners,ou=Groups,dc=example,dc=com
objectClass: posixGroup
cn: miners
gidNumber: 5000

dn: uid=john,ou=People,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: john
sn: Doe
givenName: John
cn: John Doe
displayName: John Doe
uidNumber: 10000
gidNumber: 5000
userPassword: {CRYPT}x
gecos: John Doe
```

```
loginShell: /bin/bash
homeDirectory: /home/john
```

Note

It's important that uid and gid values in your directory do not collide with local values. You can use high number ranges, such as starting at 5000 or even higher.

Add the content:

```
$ ldapadd -x -D cn=admin,dc=example,dc=com -W -f add_content.ldif
Enter LDAP Password: ********
adding new entry "ou=People,dc=example,dc=com"

adding new entry "ou=Groups,dc=example,dc=com"

adding new entry "cn=miners,ou=Groups,dc=example,dc=com"

adding new entry "uid=john,ou=People,dc=example,dc=com"
```

We can check that the information has been correctly added with the *ldapsearch* utility. For example, let's search for the *john* entry, and request the *cn* and *gidnumber* attributes:

```
$ ldapsearch -x -LLL -b dc=example,dc=com '(uid=john)' cn gidNumber
dn: uid=john,ou=People,dc=example,dc=com
cn: John Doe
gidNumber: 5000
```

Here we used an LDAP "filter": *(uid=john)*. LDAP filters are very flexible and can become complex. For example, to list the group names of which *john* is a member, we could use the filter:

```
(&(objectClass=posixGroup)(memberUid=john))
```

That is a logical *AND* between two attributes. Filters are very important in LDAP and mastering their syntax will help a long way. They are used for simple queries like this, but can also select what content is to be replicated to a secondary server, or even in complex ACLs. The full specification is defined in RFC 4515.

Notice we set the *userPassword* field for the *john* entry to the cryptic value {CRYPT}x. This essentially is an invalid password, because no hashing will produce just *x*. It's a common pattern when adding a user entry without a default password. To change the password to something valid, you can now use *ldappasswd*:

```
$ ldappasswd -x -D cn=admin,dc=example,dc=com -W -S uid=john,ou=people,dc=
    example,dc=com
New password:
Re-enter new password:
Enter LDAP Password:
```

Note

Remember that simple binds are insecure and you should add TLS support to your server!

Modifying the slapd Configuration Database

The *slapd-config* DIT can also be queried and modified. Here are some common operations.

Adding an index

Use *ldapmodify* to add an "Index" to your *{1}mdb,cn=config* database definition (for *dc=example,dc=com*). Create a file, call it uid_index.ldif, with the following contents:

```
dn: olcDatabase={1}mdb,cn=config
add: olcDbIndex
olcDbIndex: mail eq,sub
```

Then issue the command:

```
$ sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// -f uid_index.ldif

modifying entry "olcDatabase={1}mdb,cn=config"
```

You can confirm the change in this way:

```
$ sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b \
cn=config '(olcDatabase={1}mdb)' olcDbIndex

dn: olcDatabase={1}mdb,cn=config
olcDbIndex: objectClass eq
olcDbIndex: cn,uid eq
olcDbIndex: uidNumber,gidNumber eq
olcDbIndex: member,memberUid eq
olcDbIndex: mail eq,sub
```

Change the *rootDN* password:

First, run *slappasswd* to get the hash for the new password you want:

```
$ slappasswd
New password:
Re-enter new password:
{SSHA}VKrYMxlSKhONGRpC6rnASKNmXG2xHXFo
```

Now prepare a changerootpw.ldif file with this content:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
replace: olcRootPW
olcRootPW: {SSHA}VKrYMxlSKhONGRpC6rnASKNmXG2xHXFo
```

Finally, run the *ldapmodify* command:

```
$ sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// -f changerootpw.ldif
modifying entry "olcDatabase={1}mdb,cn=config"
```

We still have the actual *cn=admin,dc=example,dc=com* dn in the *dc=example,dc=com* database, so let's change it too. Since this is a regular entry in this database suffix, we can use *ldappasswd*:

```
$ ldappasswd -x -D cn=admin,dc=example,dc=com -W -S
New password:
Re-enter new password:
Enter LDAP Password: <-- current password, about to be changed
```

Adding a schema

Schemas can only be added to *cn=config* if they are in LDIF format. If not, they will first have to be converted. You can find unconverted schemas in addition to converted ones in the /etc/ldap/schema directory.

Note

It is not trivial to remove a schema from the slapd-config database. Practice adding schemas on a test system.

In the following example we'll add the password policy (*ppolicy*) schema. This schema exists in both converted (*.ldif*) and native (*.schema*) formats, so we don't have to convert it and can use *ldapadd* directly:

```
$ sudo ldapadd -Q -Y EXTERNAL -H ldapi:/// -f /etc/ldap/schema/ppolicy.ldif
adding new entry "cn=ppolicy,cn=schema,cn=config"
```

If the schema you want to add does not exist in LDIF format, a nice conversion tool that can be used is provided in the schema2ldif package.

Logging

Activity logging for slapd is very useful when implementing an OpenLDAP-based solution yet it must be manually enabled after software installation. Otherwise, only rudimentary messages will appear in the logs. Logging, like any other such configuration, is enabled via the *slapd-config* database.

OpenLDAP comes with multiple logging levels with each one containing the lower one (additive). A good level to try is *stats*. The slapd-config man page has more to say on the different subsystems.

Create the file logging.ldif with the following contents:

```
dn: cn=config
changetype: modify
replace: olcLogLevel
olcLogLevel: stats
```

Implement the change:

```
sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// -f logging.ldif
```

This will produce a significant amount of logging and you will want to throttle back to a less verbose level once your system is in production. While in this verbose mode your host's syslog engine (rsyslog) may have a hard time keeping up and may drop messages:

```
rsyslogd-2177: imuxsock lost 228 messages from pid 2547 due to rate-limiting
```

You may consider a change to rsyslog's configuration. In /etc/rsyslog.conf, put:

```
# Disable rate limiting
# (default is 200 messages in 5 seconds; below we make the 5 become 0)
$SystemLogRateLimitInterval 0
```

And then restart the rsyslog daemon:

```
sudo systemctl restart syslog.service
```

Backup and Restore

Now we have ldap running just the way we want, it is time to ensure we can save all of our work and restore it as needed.

What we need is a way to backup the directory database(s), specifically the configuration backend (*cn=config*) and the DIT (*dc=example,dc=com*). If we are going to backup those databases into, say, /export/backup, we could use *slapcat* as shown in the following script, called /usr/local/bin/ldapbackup:

```
#!/bin/bash

set -e

BACKUP_PATH=/export/backup
SLAPCAT=/usr/sbin/slapcat

nice ${SLAPCAT} -b cn=config > ${BACKUP_PATH}/config.ldif
nice ${SLAPCAT} -b dc=example,dc=com > ${BACKUP_PATH}/example.com.ldif
chown root:root ${BACKUP_PATH}/*
chmod 600 ${BACKUP_PATH}/*.ldif
```

> **Note**
>
> These files are uncompressed text files containing everything in your directory including the tree layout, usernames, and every password. So, you might want to consider making /export/backup an encrypted partition and even having the script encrypt those files as it creates them. Ideally you should do both, but that depends on your security requirements.

Then, it is just a matter of having a cron script to run this program as often as you feel comfortable with. For many, once a day suffices. For others, more often is required. Here is an example of a cron script called /etc/cron.d/ldapbackup that is run every night at 22:45h:

```
MAILTO=backup-emails@domain.com
45 22 * * *   root     /usr/local/bin/ldapbackup
```

Now the files are created, they should be copied to a backup server.

Assuming we did a fresh reinstall of ldap, the restore process could be something like this:

```
#!/bin/bash

set -e

BACKUP_PATH=/export/backup
SLAPADD=/usr/sbin/slapadd

if [ -n "$(ls -l /var/lib/ldap/* 2>/dev/null)" -o -n "$(ls -l /etc/ldap/slapd.
    d/* 2>/dev/null)" ]; then
    echo Run the following to remove the existing db:
    echo sudo systemctl stop slapd.service
    echo sudo rm -rf /etc/ldap/slapd.d/* /var/lib/ldap/*
    exit 1
fi
sudo systemctl stop slapd.service || :
sudo slapadd -F /etc/ldap/slapd.d -b cn=config -l /export/backup/config.ldif
sudo slapadd -F /etc/ldap/slapd.d -b dc=example,dc=com -l /export/backup/
    example.com.ldif
sudo chown -R openldap:openldap /etc/ldap/slapd.d/
sudo chown -R openldap:openldap /var/lib/ldap/
sudo systemctl start slapd.service
```

This is a simplistic backup strategy, of course. It's being shown here as a reference for the basic tooling you can use for backups and restores.

References

- The OpenLDAP administrators guide
- LDAP string representation of search filters

OpenLDAP Replication

The LDAP service becomes increasingly important as more networked systems begin to depend on it. In such an environment, it is standard practice to build redundancy (high availability) into LDAP to prevent havoc should the LDAP server become unresponsive. This is done through *LDAP replication*.

Replication is achieved via the *Syncrepl* engine. This allows changes to be synchronized using a *Consumer - Provider* model. A detailed description this replication mechanism can be found in the OpenLDAP Administrator's Guide and in its defining RFC 4533.

There are two ways to use this replication:

- standard replication: changed entries are sent to the consumer in their entirety. For example, if the *userPassword* attribute of the *uid=john,ou=people,dc=example,dc=com* entry changed, then the whole entry is sent to the consumer
- delta replication: only the actual change is sent, instead of the whole entry

The delta replication sends less data over the network, but is more complex to setup. We will show both in this guide.

> **Important**
>
> You **must** have TLS enabled already. Please consult the LDAP with TLS guide

Provider Configuration - replication user

Both replication strategies will need a replication user and updates to the ACLs and limits regarding this user.

To create the replication user, save the following contents to a file called replicator.ldif:

```
dn: cn=replicator,dc=example,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: replicator
description: Replication user
userPassword: {CRYPT}x
```

Then add it with *ldapadd*:

```
$ ldapadd -x -ZZ -D cn=admin,dc=example,dc=com -W -f replicator.ldif
Enter LDAP Password:
adding new entry "cn=replicator,dc=example,dc=com"
```

Now set a password for it with *ldappasswd*:

```
$ ldappasswd -x -ZZ -D cn=admin,dc=example,dc=com -W -S cn=replicator,dc=
    example,dc=com
New password:
Re-enter new password:
Enter LDAP Password:
```

The next step is to give this replication user the correct privileges:

- read access to the content that we want replicated
- no search limits on this content

For that we need to update the ACLs on the provider. Since ordering matters, first check what the existing ACLs look like on the *dc=example,dc=com* tree:

```
$ sudo ldapsearch -Q -Y EXTERNAL -H ldapi:/// -LLL -b cn=config '(olcSuffix=dc
    =example,dc=com)' olcAccess
dn: olcDatabase={1}mdb,cn=config
olcAccess: {0}to attrs=userPassword by self write by anonymous auth by * none
olcAccess: {1}to attrs=shadowLastChange by self write by * read
olcAccess: {2}to * by * read
```

What we need is to insert a new rule before the first one, and also adjust the limits for the replicator user. Prepare the replicator-acl-limits.ldif file with this content:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcAccess
olcAccess: {0}to *
  by dn.exact="cn=replicator,dc=example,dc=com" read
  by * break
-
add: olcLimits
olcLimits: dn.exact="cn=replicator,dc=example,dc=com"
  time.soft=unlimited time.hard=unlimited
  size.soft=unlimited size.hard=unlimited
```

And add it to the server:

```
$ sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// -f replicator-acl-limits.ldif
modifying entry "olcDatabase={1}mdb,cn=config"
```

Provider Configuration - standard replication

The remaining configuration for the provider using standard replication is to add the *syncprov* overlay on top of the *dc=example,dc=com* database.o

Create a file called provider_simple_sync.ldif with this content:

```
# Add indexes to the frontend db.
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryCSN eq
-
add: olcDbIndex
olcDbIndex: entryUUID eq

#Load the syncprov module.
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: syncprov
```

```
# syncrepl Provider for primary db
dn: olcOverlay=syncprov,olcDatabase={1}mdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov
olcSpCheckpoint: 100 10
olcSpSessionLog: 100
```

Customization warning

The LDIF above has some parameters that you should review before deploying in production on your directory. In particular:

- *olcSpCheckpoint*, *olcSpSessionLog*: please see the slapo-syncprov(5) manpage. In general, *olcSpSessionLog* should be equal to, or preferably larger, than the number of entries in your directory. Also see ITS #8125 for details on an existing bug.

Add the new content:

```
sudo ldapadd -Q -Y EXTERNAL -H ldapi:/// -f provider_simple_sync.ldif
```

The Provider is now configured.

Consumer Configuration - standard replication

Install the software by going through Installation. Make sure schemas and the database suffix are the same, and enable TLS.

Create an LDIF file with the following contents and name it consumer_simple_sync.ldif:

```
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: syncprov

dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryUUID eq
-
add: olcSyncrepl
olcSyncrepl: rid=0
  provider=ldap://ldap01.example.com
  bindmethod=simple
  binddn="cn=replicator,dc=example,dc=com" credentials=<secret>
  searchbase="dc=example,dc=com"
  schemachecking=on
  type=refreshAndPersist retry="60 +"
  starttls=critical tls_reqcert=demand
-
add: olcUpdateRef
olcUpdateRef: ldap://ldap01.example.com
```

Ensure the following attributes have the correct values:

- *provider* (Provider server's hostname – ldap01.example.com in this example – or IP address). It must match what is presented in the provider's SSL certificate.
- *binddn* (the bind DN for the *replicator* user)
- *credentials* (the password you selected for the replicator user)
- *searchbase* (the database suffix you're using, i.e., content that is to be replicated)
- *olcUpdateRef* (Provider server's hostname or IP address, given to clients if they try to write to this consumer)
- *rid* (Replica ID, an unique 3-digit that identifies the replica. Each consumer should have at least one rid)

  Note

  Note that a successful encrypted connection via *START_TLS* is being enforced in this configuration, to avoid sending the credentials in the clear across the network. See LDAP with TLS for details on how to setup OpenLDAP with trusted SSL certificates.

Add the new configuration:

```
sudo ldapadd -Q -Y EXTERNAL -H ldapi:/// -f consumer_simple_sync.ldif
```

You're done. The *dc=example,dc=com* tree should now be synchronizing.

Provider Configuration - delta replication

The remaining provider configuration for delta replication is:

- create a new database called *accesslog*
- add the *syncprov* overlay on top of the *accesslog* and *dc=example,dc=com* databases
- add the *accesslog* overlay on top of the *dc=example,dc=com* database

Add syncprov and accesslog overlays and DBs

Create an LDIF file with the following contents and name it provider_sync.ldif:

```
# Add indexes to the frontend db.
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryCSN eq
-
add: olcDbIndex
olcDbIndex: entryUUID eq

#Load the syncprov and accesslog modules.
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: syncprov
-
add: olcModuleLoad
olcModuleLoad: accesslog

# Accesslog database definitions
```

```
dn: olcDatabase={2}mdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: {2}mdb
olcDbDirectory: /var/lib/ldap/accesslog
olcSuffix: cn=accesslog
olcRootDN: cn=admin,dc=example,dc=com
olcDbIndex: default eq
olcDbIndex: entryCSN,objectClass,reqEnd,reqResult,reqStart
olcAccess: {0}to * by dn.exact="cn=replicator,dc=example,dc=com" read by *
    break
olcLimits: dn.exact="cn=replicator,dc=example,dc=com"
  time.soft=unlimited time.hard=unlimited
  size.soft=unlimited size.hard=unlimited

# Accesslog db syncprov.
dn: olcOverlay=syncprov,olcDatabase={2}mdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov
olcSpNoPresent: TRUE
olcSpReloadHint: TRUE

# syncrepl Provider for primary db
dn: olcOverlay=syncprov,olcDatabase={1}mdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov
olcSpCheckpoint: 100 10
olcSpSessionLog: 100

# accesslog overlay definitions for primary db
dn: olcOverlay=accesslog,olcDatabase={1}mdb,cn=config
objectClass: olcOverlayConfig
objectClass: olcAccessLogConfig
olcOverlay: accesslog
olcAccessLogDB: cn=accesslog
olcAccessLogOps: writes
olcAccessLogSuccess: TRUE
# scan the accesslog DB every day, and purge entries older than 7 days
olcAccessLogPurge: 07+00:00 01+00:00
```

Customization warning

The LDIF above has some parameters that you should review before deploying in production on your directory. In particular:

- *olcSpCheckpoint*, *olcSpSessionLog*: please see the slapo-syncprov(5) manpage. In general, *olcSpSessionLog* should be equal to, or preferably larger, than the number of entries in your directory. Also see ITS #8125 for details on an existing bug.
- *olcAccessLogPurge*: check the slapo-accesslog(5) manpage

Create a directory:

```
sudo -u openldap mkdir /var/lib/ldap/accesslog
```

Add the new content:

```
sudo ldapadd -Q -Y EXTERNAL -H ldapi:/// -f provider_sync.ldif
```

The Provider is now configured.

Consumer Configuration

Install the software by going through Installation. Make sure schemas and the database suffix are the same, and enable TLS.

Create an LDIF file with the following contents and name it consumer_sync.ldif:

```
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: syncprov

dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryUUID eq
-
add: olcSyncrepl
olcSyncrepl: rid=0
  provider=ldap://ldap01.example.com
  bindmethod=simple
  binddn="cn=replicator,dc=example,dc=com" credentials=<secret>
  searchbase="dc=example,dc=com"
  logbase="cn=accesslog"
  logfilter="(&(objectClass=auditWriteObject)(reqResult=0))"
  schemachecking=on
  type=refreshAndPersist retry="60 +"
  syncdata=accesslog
  starttls=critical tls_reqcert=demand
-
add: olcUpdateRef
olcUpdateRef: ldap://ldap01.example.com
```

Ensure the following attributes have the correct values:

- *provider* (Provider server's hostname – ldap01.example.com in this example – or IP address). It must match what is presented in the provider's SSL certificate.
- *binddn* (the bind DN for the *replicator* user)
- *credentials* (the password you selected for the replicator user)
- *searchbase* (the database suffix you're using, i.e., content that is to be replicated)
- *olcUpdateRef* (Provider server's hostname or IP address, given to clients if they try to write to this consumer)
- *rid* (Replica ID, an unique 3-digit that identifies the replica. Each consumer should have at least one rid)

Note

Note that a successful encrypted connection via *START_TLS* is being enforced in this configuration, to avoid sending the credentials in the clear across the network. See LDAP with TLS for details on how to setup OpenLDAP with trusted SSL certificates.

Add the new configuration:

```
sudo ldapadd -Q -Y EXTERNAL -H ldapi:/// -f consumer_sync.ldif
```

You're done. The *dc=example,dc=com* tree should now be synchronizing.

Testing

Once replication starts, you can monitor it by running

```
$ ldapsearch -z1 -LLL -x -s base -b dc=example,dc=com contextCSN
dn: dc=example,dc=com
contextCSN: 20200423222317.722667Z#000000#000#000000
```

on both the provider and the consumer. Once the *contextCSN* value for both match, both trees are in sync. Every time a change is done in the provider, this value will change and so should the one in the consumer(s).

If your connection is slow and/or your ldap database large, it might take a while for the consumer's *contextCSN* match the provider's. But, you will know it is progressing since the consumer's *contextCSN* will be steadly increasing.

If the consumer's *contextCSN* is missing or does not match the provider, you should stop and figure out the issue before continuing. Try checking the slapd entries in */var/log/syslog* in the provider to see if the consumer's authentication requests were successful or its requests to retrieve data return no errors. In particular, verify that you can connect to the provider from the consumer as the replicator binddn using *START_TLS*:

```
ldapwhoami -x -ZZ -D cn=replicator,dc=example,dc=com -W -h ldap01.example.com
```

For our example, you should now see the *john* user in the replicated tree:

```
$ ldapsearch -x -LLL -b dc=example,dc=com -h ldap02.example.com '(uid=john)'
    uid
dn: uid=john,ou=People,dc=example,dc=com
uid: john
```

References

- Replication types, OpenLDAP Administrator's Guide
- LDAP Sync Replication - OpenLDAP Administrator's Guide
- RFC 4533.

LDAP Workstation Authentication

Once you have a working LDAP server, you will need to install libraries on the client that will know how and when to contact it. On Ubuntu, this has been traditionally accomplished by installing the libnss-ldap package, but nowadays you should use SSSD. Please refer to Service - SSSD.

User and Group Management - ldapscripts

Another very common usage case for having an LDAP server is to store unix user and group information in the directory. There are many tools out there, but usually big deployments will have developed their own. Here we will briefly show how to use the ldapscripts package for an easy and quick way to start storing user and group information in OpenLDAP.

Install the package:

```
sudo apt install ldapscripts
```

Then edit the file /etc/ldapscripts/ldapscripts.conf to arrive at something similar to the following:

```
SERVER=ldap://ldap01.example.com
LDAPBINOPTS="-ZZ"
BINDDN='cn=admin,dc=example,dc=com'
BINDPWDFILE="/etc/ldapscripts/ldapscripts.passwd"
SUFFIX='dc=example,dc=com'
GSUFFIX='ou=Groups'
USUFFIX='ou=People'
MSUFFIX='ou=Computers'
```

> **Note**
>
> - Adjust *SERVER* and related *SUFFIX* options to suit your directory structure.
> - Note we are forcing *START_TLS* usage here (*-ZZ* parameter), please refer to LDAP with TLS for details on how to set the server up with TLS support

Store the *cn=admin* password in the /etc/ldapscripts/ldapscripts.passwd file and make sure it's only readable by the *root* local user:

```
$ sudo chmod 400 /etc/ldapscripts/ldapscripts.passwd
```

The scripts are now ready to help manage your directory. Here are some examples of how to use them:

- Create a new user:

  ```
  sudo ldapaddgroup george
  sudo ldapadduser george george
  ```

  This will create a group and user with name *george* and set the user's primary group (gid) to *george*

- Change a user's password:

  ```
  $ sudo ldapsetpasswd george
  Changing password for user uid=george,ou=People,dc=example,dc=com
  New Password:
  Retype New Password:
  Successfully set password for user uid=george,ou=People,dc=example,dc=com
  ```

- Delete a user:

  ```
  sudo ldapdeleteuser george

  > **Note**
  >
  > This won't delete the user's primary group, but will remove the user
    from supplementary ones.
  ```

- Add a group:

```
sudo ldapaddgroup qa
```

- Delete a group:

  ```
  sudo ldapdeletegroup qa
  ```

- Add a user to a group:

  ```
  sudo ldapaddusertogroup george qa
  ```

  You should now see a *memberUid* attribute for the *qa* group with a value of *george*.

- Remove a user from a group:

  ```
  sudo ldapdeleteuserfromgroup george qa
  ```

  The *memberUid* attribute should now be removed from the *qa* group.

- The ldapmodifyuser script allows you to add, remove, or replace a user's attributes. The script uses the same syntax as the ldapmodify utility. For example:

  ```
  sudo ldapmodifyuser george
  # About to modify the following entry :
  dn: uid=george,ou=People,dc=example,dc=com
  objectClass: account
  objectClass: posixAccount
  cn: george
  uid: george
  uidNumber: 10001
  gidNumber: 10001
  homeDirectory: /home/george
  loginShell: /bin/bash
  gecos: george
  description: User account
  userPassword:: e1NTSEF9eXFsTFcyWlhwWkF1eGUybVdFWHZKRzJVMjFTSG9vcHk=

  # Enter your modifications here, end with CTRL-D.
  dn: uid=george,ou=People,dc=example,dc=com
  replace: gecos
  gecos: George Carlin
  ```

  The user's *gecos* should now be "George Carlin".

- A nice feature of ldapscripts is the template system. Templates allow you to customize the attributes of user, group, and machine objects. For example, to enable the *user* template edit /etc/ldapscripts/ldapscripts.conf changing:

  ```
  UTEMPLATE="/etc/ldapscripts/ldapadduser.template"
  ```

  There are *sample* templates in the /usr/share/doc/ldapscripts/examples directory. Copy or rename the ldapadduser.template.sample file to /etc/ldapscripts/ldapadduser.template:

  ```
  sudo cp /usr/share/doc/ldapscripts/examples/ldapadduser.template.sample \
  /etc/ldapscripts/ldapadduser.template
  ```

  Edit the new template to add the desired attributes. The following will create new users with an objectClass of inetOrgPerson:

```
dn: uid=<user>,<usuffix>,<suffix>
objectClass: inetOrgPerson
objectClass: posixAccount
cn: <user>
sn: <ask>
uid: <user>
uidNumber: <uid>
gidNumber: <gid>
homeDirectory: <home>
loginShell: <shell>
gecos: <user>
description: User account
title: Employee
```

Notice the *<ask>* option used for the *sn* attribute. This will make *ldapadduser* prompt you for its value.

There are utilities in the package that were not covered here. This command will output a list:

```
dpkg –L ldapscripts | grep /usr/sbin
```

Resources

- The primary resource is the upstream documentation: www.openldap.org
- Zytrax's LDAP for Rocket Scientists; a less pedantic but comprehensive treatment of LDAP
- A Ubuntu community OpenLDAP wiki page has a collection of notes
- O'Reilly's LDAP System Administration (textbook; 2003)
- Packt's Mastering OpenLDAP (textbook; 2007)

LDAP & TLS

When authenticating to an OpenLDAP server it is best to do so using an encrypted session. This can be accomplished using Transport Layer Security (TLS).

Here, we will be our own *Certificate Authority* and then create and sign our LDAP server certificate as that CA. This guide will use the *certtool* utility to complete these tasks. For simplicity, this is being done on the OpenLDAP server itself, but your real internal CA should be elsewhere.

Install the *gnutls-bin* and *ssl-cert* packages:

```
sudo apt install gnutls-bin ssl-cert
```

Create a private key for the Certificate Authority:

```
sudo certtool --generate-privkey --bits 4096 --outfile /etc/ssl/private/
    mycakey.pem
```

Create the template/file /etc/ssl/ca.info to define the CA:

```
cn = Example Company
ca
cert_signing_key
expiration_days = 3650
```

Create the self-signed CA certificate:

```
sudo certtool --generate-self-signed \
--load-privkey /etc/ssl/private/mycakey.pem \
--template /etc/ssl/ca.info \
--outfile /usr/local/share/ca-certificates/mycacert.crt
```

> **Note**
>
> Yes, the *–outfile* path is correct, we are writing the CA certificate to */usr/local/share/ca-certificates*. This is where *update-ca-certificates* will pick up trusted local CAs from. To pick up CAs from */usr/share/ca-certificates*, a call to dpkg–reconfigure ca–certificates is necessary.

Run update–ca–certificates to add the new CA certificate to the list of trusted CAs. Note the one added CA:

```
$ sudo update-ca-certificates
Updating certificates in /etc/ssl/certs...
1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
```

This also creates a /etc/ssl/certs/mycacert.pem symlink pointing to the real file in /usr/local/share/ca–certificates.

Make a private key for the server:

```
sudo certtool --generate-privkey \
--bits 2048 \
--outfile /etc/ldap/ldap01_slapd_key.pem
```

> **Note**
>
> Replace *ldap01* in the filename with your server's hostname. Naming the certificate and key for the host and service that will be using them will help keep things clear.

Create the /etc/ssl/ldap01.info info file containing:

```
organization = Example Company
cn = ldap01.example.com
tls_www_server
encryption_key
signing_key
expiration_days = 365
```

The above certificate is good for 1 year, and it's valid only for the *ldap01.example.com* hostname. Adjust accordingly.

Create the server's certificate:

```
sudo certtool --generate-certificate \
--load-privkey /etc/ldap/ldap01_slapd_key.pem \
--load-ca-certificate /etc/ssl/certs/mycacert.pem \
--load-ca-privkey /etc/ssl/private/mycakey.pem \
--template /etc/ssl/ldap01.info \
--outfile /etc/ldap/ldap01_slapd_cert.pem
```

Adjust permissions and ownership:

```
sudo chgrp openldap /etc/ldap/ldap01_slapd_key.pem
sudo chmod 0640 /etc/ldap/ldap01_slapd_key.pem
```

Your server is now ready to accept the new TLS configuration.

Create the file certinfo.ldif with the following contents (adjust paths and filenames accordingly):

```
dn: cn=config
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/ssl/certs/mycacert.pem
-
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ldap/ldap01_slapd_cert.pem
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ldap/ldap01_slapd_key.pem
```

Use the ldapmodify command to tell slapd about our TLS work via the slapd-config database:

```
sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f certinfo.ldif
```

Contratry to popular belief, you do not need *ldaps://* in /etc/default/slapd in order to use encryption. You should have just:

```
SLAPD_SERVICES="ldap:/// ldapi:///"
```

> **Note**
>
> LDAP over TLS/SSL (ldaps://) is deprecated in favour of *StartTLS*. The latter refers to an existing LDAP session (listening on TCP port 389) becoming protected by TLS/SSL whereas LDAPS, like HTTPS, is a distinct encrypted-from-the-start protocol that operates over TCP port 636.

Certificate for an OpenLDAP replica

To generate a certificate pair for an OpenLDAP replica (consumer), create a holding directory (which will be used for the eventual transfer) and:

```
mkdir ldap02-ssl
cd ldap02-ssl
certtool --generate-privkey \
--bits 2048 \
--outfile ldap02_slapd_key.pem
```

Create an info file, ldap02.info, for the Consumer server, adjusting its values accordingly:

```
organization = Example Company
cn = ldap02.example.com
tls_www_server
encryption_key
signing_key
expiration_days = 365
```

Create the Consumer's certificate:

```
sudo certtool --generate-certificate \
--load-privkey ldap02_slapd_key.pem \
--load-ca-certificate /etc/ssl/certs/mycacert.pem \
--load-ca-privkey /etc/ssl/private/mycakey.pem \
--template ldap02.info \
--outfile ldap02_slapd_cert.pem
```

Note

We had to use sudo to get access to the CA's private key. This means the generated certificate file is owned by root. You should change that ownership back to your regular user before copying these files over to the Consumer.

Get a copy of the CA certificate:

```
cp /etc/ssl/certs/mycacert.pem .
```

We're done. Now transfer the ldap02-ssl directory to the Consumer. Here we use scp (adjust accordingly):

```
cd ..
scp -r ldap02-ssl user@consumer:
```

On the Consumer side, install the certificate files you just transferred:

```
sudo cp ldap02_slapd_cert.pem ldap02_slapd_key.pem /etc/ldap
sudo chgrp openldap /etc/ldap/ldap02_slapd_key.pem
sudo chmod 0640 /etc/ldap/ldap02_slapd_key.pem
sudo cp mycacert.pem /usr/local/share/ca-certificates/mycacert.crt
sudo update-ca-certificates
```

Create the file certinfo.ldif with the following contents (adjust accordingly regarding paths and filenames, if needed):

```
dn: cn=config
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/ssl/certs/mycacert.pem
-
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ldap/ldap02_slapd_cert.pem
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ldap/ldap02_slapd_key.pem
```

Configure the slapd-config database:

```
sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f certinfo.ldif
```

Test:

```
$ ldapwhoami -x -ZZ -h ldap02.example.com
anonymous
```

Network File System (NFS)

NFS allows a system to share directories and files with others over a network. By using NFS, users and programs can access files on remote systems almost as if they were local files.

Some of the most notable benefits that NFS can provide are:

- Local workstations use less disk space because commonly used data can be stored on a single machine and still remain accessible to others over the network.
- There is no need for users to have separate home directories on every network machine. Home directories could be set up on the NFS server and made available throughout the network.

- Storage devices such as floppy disks, CDROM drives, and USB Thumb drives can be used by other machines on the network. This may reduce the number of removable media drives throughout the network.

Installation

At a terminal prompt enter the following command to install the NFS Server:

```
sudo apt install nfs-kernel-server
```

To start the NFS server, you can run the following command at a terminal prompt:

```
sudo systemctl start nfs-kernel-server.service
```

Configuration

You can configure the directories to be exported by adding them to the /etc/exports file. For example:

```
/srv      *(ro,sync,subtree_check)
/home     *.hostname.com(rw,sync,no_subtree_check)
/scratch  *(rw,async,no_subtree_check,no_root_squash,noexec)
```

Make sure any custom mount points you're adding have been created (/srv and /home will already exist):

```
sudo mkdir /scratch
```

Apply the new config via:

```
sudo exportfs -a
```

You can replace * with one of the hostname formats. Make the hostname declaration as specific as possible so unwanted systems cannot access the NFS mount. Be aware that *.hostname.com will matchfoo.hostname .com but not foo.bar.my-domain.com.

The *sync*/*async* options control whether changes are gauranteed to be committed to stable storage before replying to requests. *async* thus gives a performance benefit but risks data loss or corruption. Even though *sync* is the default, it's worth setting since exportfs will issue a warning if it's left unspecified.

subtree_check and *no_subtree_check* enables or disables a security verification that subdirectories a client attempts to mount for an exported filesystem are ones they're permitted to do so. This verification step has some performance implications for some use cases, such as home directories with frequent file renames. Read-only filesystems are more suitable to enable *subtree_check* on. Like with sync, exportfs will warn if it's left unspecified.

There are a number of optional settings for NFS mounts for tuning performance, tightening security, or providing conveniences. These settings each have their own trade-offs so it is important to use them with care, only as needed for the particular use case. *no_root_squash*, for example, adds a convenience to allow root-owned files to be modified by any client system's root user; in a multi-user environment where executables are allowed on a shared mount point, this could lead to security problems. *noexec* prevents executables from running from the mount point.

NFS Client Configuration

To enable NFS support on a client system, enter the following command at the terminal prompt:

```
sudo apt install nfs-common
```

Use the mount command to mount a shared NFS directory from another machine, by typing a command line similar to the following at a terminal prompt:

```
sudo mkdir /opt/example
sudo mount example.hostname.com:/srv /opt/example
```

> **Warning**
>
> The mount point directory /opt/example must exist. There should be no files or subdirectories in the /opt/example directory, else they will become inaccessible until the nfs filesystem is unmounted.

An alternate way to mount an NFS share from another machine is to add a line to the /etc/fstab file. The line must state the hostname of the NFS server, the directory on the server being exported, and the directory on the local machine where the NFS share is to be mounted.

The general syntax for the line in /etc/fstab file is as follows:

```
example.hostname.com:/srv /opt/example nfs rsize=8192,wsize=8192,timeo=14,intr
```

References

Linux NFS wiki Linux NFS faq

Ubuntu Wiki NFS Howto Ubuntu Wiki NFSv4 Howto

OpenSSH Server

Introduction

Openssh is a powerful collection of tools for the remote control of, and transfer of data between, networked computers. You will also learn about some of the configuration settings possible with the OpenSSH server application and how to change them on your Ubuntu system.

OpenSSH is a freely available version of the Secure Shell (SSH) protocol family of tools for remotely controlling, or transferring files between, computers. Traditional tools used to accomplish these functions, such as telnet or rcp, are insecure and transmit the user's password in cleartext when used. OpenSSH provides a server daemon and client tools to facilitate secure, encrypted remote control and file transfer operations, effectively replacing the legacy tools.

The OpenSSH server component, sshd, listens continuously for client connections from any of the client tools. When a connection request occurs, sshd sets up the correct connection depending on the type of client tool connecting. For example, if the remote computer is connecting with the ssh client application, the OpenSSH server sets up a remote control session after authentication. If a remote user connects to an OpenSSH server with scp, the OpenSSH server daemon initiates a secure copy of files between the server and client after authentication. OpenSSH can use many authentication methods, including plain password, public key, and Kerberos tickets.

Installation

Installation of the OpenSSH client and server applications is simple. To install the OpenSSH client applications on your Ubuntu system, use this command at a terminal prompt:

```
sudo apt install openssh-client
```

To install the OpenSSH server application, and related support files, use this command at a terminal prompt:

```
sudo apt install openssh-server
```

Configuration

You may configure the default behavior of the OpenSSH server application, sshd, by editing the file /etc/ssh/sshd_config. For information about the configuration directives used in this file, you may view the appropriate manual page with the following command, issued at a terminal prompt:

```
man sshd_config
```

There are many directives in the sshd configuration file controlling such things as communication settings, and authentication modes. The following are examples of configuration directives that can be changed by editing the /etc/ssh/sshd_config file.

> **Tip**
>
> Prior to editing the configuration file, you should make a copy of the original file and protect it from writing so you will have the original settings as a reference and to reuse as necessary.
>
> Copy the /etc/ssh/sshd_config file and protect it from writing with the following commands, issued at a terminal prompt:
>
> ```
> sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.original
> sudo chmod a-w /etc/ssh/sshd_config.original
> ```

Furthermore since losing an ssh server might mean losing your way to reach a server, check the configuration after changing it and before restarting the server:

```
sudo sshd -t -f /etc/ssh/sshd_config
```

The following are *examples* of configuration directives you may change:

- To set your OpenSSH to listen on TCP port 2222 instead of the default TCP port 22, change the Port directive as such:

  Port 2222

- To make your OpenSSH server display the contents of the /etc/issue.net file as a pre-login banner, simply add or modify this line in the /etc/ssh/sshd_config file:

  Banner /etc/issue.net

After making changes to the /etc/ssh/sshd_config file, save the file, and restart the sshd server application to effect the changes using the following command at a terminal prompt:

```
sudo systemctl restart sshd.service
```

> **Warning**
>
> Many other configuration directives for sshd are available to change the server application's behavior to fit your needs. Be advised, however, if your only method of access to a server is ssh, and you make a mistake in configuring sshd via the /etc/ssh/sshd_config file, you may find you are locked out of the server upon restarting it. Additionally, if an incorrect configuration directive is supplied, the sshd server may refuse to start, so be extra careful when editing this file on a remote server.

SSH Keys

SSH allow authentication between two hosts without the need of a password. SSH key authentication uses a *private key* and a *public key*.

To generate the keys, from a terminal prompt enter:

```
ssh-keygen -t rsa
```

This will generate the keys using the *RSA Algorithm*. During the process you will be prompted for a password. Simply hit *Enter* when prompted to create the key.

By default the *public* key is saved in the file ~/.ssh/id_rsa.pub, while ~/.ssh/id_rsa is the *private* key. Now copy the id_rsa.pub file to the remote host and append it to ~/.ssh/authorized_keys by entering:

```
ssh-copy-id username@remotehost
```

Finally, double check the permissions on the authorized_keys file, only the authenticated user should have read and write permissions. If the permissions are not correct change them by:

```
chmod 600 .ssh/authorized_keys
```

You should now be able to SSH to the host without being prompted for a password.

Import keys from public keyservers

These days many users have already ssh keys registered with services like launchpad or github. Those can be easily imported with:

```
ssh-import-id <username-on-remote-service>
```

The prefix lp: is implied and means fetching from launchpad, the alternative gh: will make the tool fetch from github instead.

Two factor authentication with U2F/FIDO

OpenSSH 8.2 added support for U2F/FIDO hardware authentication devices. These devices are used to provide an extra layer of security on top of the existing key-based authentication, as the hardware token needs to be present to finish the authentication.

It's very simple to use and setup. The only extra step is generate a new keypair that can be used with the hardware device. For that, there are two key types that can be used: ecdsa-sk and ed25519-sk. The former has broader hardware support, while the latter might need a more recent device.

Once the keypair is generated, it can be used as you would normally use any other type of key in openssh. The only requirement is that in order to use the private key, the U2F device has to be present on the host.

For example, plug the U2F device in and generate a keypair to use with it:

```
$ ssh-keygen -t ecdsa-sk
Generating public/private ecdsa-sk key pair.
You may need to touch your authenticator to authorize key generation. <--
    touch device
Enter file in which to save the key (/home/ubuntu/.ssh/id_ecdsa_sk):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ecdsa_sk
```

```
Your public key has been saved in /home/ubuntu/.ssh/id_ecdsa_sk.pub
The key fingerprint is:
SHA256:V9PQ1MqaU8FODXdHqDiH9Mxb8XK3o5aVYDQLVl9IFRo ubuntu@focal
```

Now just transfer the public part to the server to ~/.ssh/authorized_keys and you are ready to go:

```
$ ssh -i .ssh/id_ecdsa_sk ubuntu@focal.server
Confirm user presence for key ECDSA-SK SHA256:
    V9PQ1MqaU8FODXdHqDiH9Mxb8XK3o5aVYDQLVl9IFRo <-- touch device
Welcome to Ubuntu Focal Fossa (GNU/Linux 5.4.0-21-generic x86_64)
(...)
ubuntu@focal.server:~$
```

References

- Ubuntu Wiki SSH page.
- OpenSSH Website
- OpenSSH 8.2 release notes
- Advanced OpenSSH Wiki Page

VPN

OpenVPN is a Virtual Private Networking (VPN) solution provided in the Ubuntu Repositories. It is flexible, reliable and secure. It belongs to the family of SSL/TLS VPN stacks (different from IPSec VPNs). This chapter will cover installing and configuring OpenVPN to create a VPN.

OpenVPN

If you want more than just pre-shared keys OpenVPN makes it easy to setup and use a Public Key Infrastructure (PKI) to use SSL/TLS certificates for authentication and key exchange between the VPN server and clients. OpenVPN can be used in a routed or bridged VPN mode and can be configured to use either UDP or TCP. The port number can be configured as well, but port 1194 is the official one. And it is only using that single port for all communication. VPN client implementations are available for almost anything including all Linux distributions, OS X, Windows and OpenWRT based WLAN routers.

Server Installation

To install openvpn in a terminal enter:

```
sudo apt install openvpn easy-rsa
```

Public Key Infrastructure Setup

The first step in building an OpenVPN configuration is to establish a PKI (public key infrastructure). The PKI consists of:

- a separate certificate (also known as a public key) and private key for the server and each client, and

- a master Certificate Authority (CA) certificate and key which is used to sign each of the server and client certificates.

OpenVPN supports bidirectional authentication based on certificates, meaning that the client must authenticate the server certificate and the server must authenticate the client certificate before mutual trust is established.

Both server and client will authenticate the other by first verifying that the presented certificate was signed by the master certificate authority (CA), and then by testing information in the now-authenticated certificate header, such as the certificate common name or certificate type (client or server).

Certificate Authority Setup

To setup your own Certificate Authority (CA) and generating certificates and keys for an OpenVPN server and multiple clients first copy the easy−rsa directory to /etc/openvpn. This will ensure that any changes to the scripts will not be lost when the package is updated. From a terminal change to user root and:

```
sudo make−cadir /etc/openvpn/easy−rsa
```

Note: If you want you can - instead of doing so interactively - edit /etc/openvpn/easy−rsa/vars adjusting it to your needs.

Change to the newly created directory /etc/openvpn/easy−rsa and run:

```
./easyrsa init−pki
./easyrsa build−ca
```

Server Keys and Certificates

Next, we will generate a key pair for the server:

```
./easyrsa gen−req myservername nopass
```

Diffie Hellman parameters must be generated for the OpenVPN server. The following will place them in pki/dh.pem.

```
./easyrsa gen−dh
```

And finally a certificate for the server:

```
./easyrsa gen−req myservername nopass
./easyrsa sign−req server myservername
```

All certificates and keys have been generated in subdirectories. Common practice is to copy them to /etc/openvpn/:

```
cp pki/dh.pem pki/ca.crt pki/issued/myservername.crt pki/private/myservername.
    key /etc/openvpn/
```

Client Certificates

The VPN client will also need a certificate to authenticate itself to the server. Usually you create a different certificate for each client.

This can either be done on the server (as the keys and certificates above) and then securely distributed to the client. Or vice versa the client can generate and submit a request that is sent and signed by the server.

To create the certificate, enter the following in a terminal while being user root:

```
./easyrsa gen-req myclient1 nopass
./easyrsa sign-req client myclient1
```

If the first command above was done on a remote system get the .req file to the CA server. There you can then import it via easyrsa import-req /incoming/myclient1.req myclient1. Then you can go on with the second sign-eq command.

In both cases, afterwards copy the following files to the client using a secure method:

- pki/ca.crt
- pki/issued/myclient1.crt

As the client certificates and keys are only required on the client machine, you can remove them from the server.

Simple Server Configuration

Along with your OpenVPN installation you got these sample config files (and many more if you check):

```
root@server:/# ls -l /usr/share/doc/openvpn/examples/sample-config-files/
total 68
-rw-r--r-- 1 root root 3427 2011-07-04 15:09 client.conf
-rw-r--r-- 1 root root 4141 2011-07-04 15:09 server.conf.gz
```

Start with copying and unpacking server.conf.gz to /etc/openvpn/server.conf.

```
sudo cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz /
    etc/openvpn/myserver.conf.gz
sudo gzip -d /etc/openvpn/myserver.conf.gz
```

Edit /etc/openvpn/myserver.conf to make sure the following lines are pointing to the certificates and keys you created in the section above.

```
ca ca.crt
cert myservername.crt
key myservername.key
dh dh2048.pem
```

Complete this set with a ta key in etc/openvpn for tls-auth like:

```
sudo openvpn --genkey --secret ta.key
```

Edit /etc/sysctl.conf and uncomment the following line to enable IP forwarding.

```
#net.ipv4.ip_forward=1
```

Then reload sysctl.

```
sudo sysctl -p /etc/sysctl.conf
```

That is the minimum you have to configure to get a working OpenVPN server. You can use all the default settings in the sample server.conf file. Now start the server.

Be aware that the *"systemctl start openvpn"* is **not** starting your openvpn you just defined. Openvpn uses templatized systemd jobs, openvpn@CONFIGFILENAME. So if for example your configuration file is myserver.conf your service is called openvpn@myserver. You can run all kind of service and systemctl commands like start/stop/enable/disable/preset against a templatized service like openvpn@server.

```
$ sudo systemctl start openvpn@myserver
```

You will find logging and error messages in your via journal. If you started a templatized service openvpn@server you can filter for this particular message source with:

```
sudo journalctl -u openvpn@myserver -xe
```

The same templatized approach works for all of systemctl:

```
$ sudo systemctl status openvpn@myserver
openvpn@myserver.service - OpenVPN connection to myserver
   Loaded: loaded (/lib/systemd/system/openvpn@.service; disabled; vendor
      preset: enabled)
   Active: active (running) since Thu 2019-10-24 10:59:25 UTC; 10s ago
     Docs: man:openvpn(8)
           https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage
           https://community.openvpn.net/openvpn/wiki/HOWTO
 Main PID: 4138 (openvpn)
   Status: "Initialization Sequence Completed"
    Tasks: 1 (limit: 533)
   Memory: 1.0M
   CGroup: /system.slice/system-openvpn.slice/openvpn@myserver.service
           4138 /usr/sbin/openvpn --daemon ovpn-myserver --status /run/openvpn
              /myserver.status 10 --cd /etc/openvpn --script-security 2 --
              config /etc/openvpn/myserver.conf --writepid /run/

Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: /sbin/ip addr add dev
   tun0 local 10.8.0.1 peer 10.8.0.2
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: /sbin/ip route add
   10.8.0.0/24 via 10.8.0.2
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: Could not determine IPv4/
   IPv6 protocol. Using AF_INET
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: Socket Buffers: R
   =[212992->212992] S=[212992->212992]
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: UDPv4 link local (bound):
    [AF_INET][undef]:1194
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: UDPv4 link remote: [
   AF_UNSPEC]
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: MULTI: multi_init called,
    r=256 v=256
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: IFCONFIG POOL: base
   =10.8.0.4 size=62, ipv6=0
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: IFCONFIG POOL LIST
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: Initialization Sequence
   Completed
```

You can enable/disable various openvpn services on one system, but you could also let Ubuntu it for you. There is config for AUTOSTARTin /etc/default/openvpn. Allowed values are “all”, “none” or space separated list of names of the VPNs. If empty, “all” is assumed. The VPN name refers to the VPN configutation file name. i.e. home would be /etc/openvpn/home.conf If you’re running systemd, changing this variable will require running systemctl daemon-reload followed by a *restart* of the openvpn service (if you removed entries you may have to stop those manually). After “systemctl daemon-reload” a restart of the “generic” openvpn will restart all dependent services that the generator in /lib/systemd/system-generators/openvpn-generator created for your conf files when you called daemon-reload.

Now check if OpenVPN created a tun0 interface:

```
root@server:/etc/openvpn# ip addr show dev tun0
```

```
5: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel
    state UNKNOWN group default qlen 100
     link/none
     inet 10.8.0.1 peer 10.8.0.2/32 scope global tun0
        valid_lft forever preferred_lft forever
     inet6 fe80::b5ac:7829:f31e:32c5/64 scope link stable-privacy
        valid_lft forever preferred_lft forever
```

Simple Client Configuration

There are various different OpenVPN client implementations with and without GUIs. You can read more about clients in a later section on VPN Clients. For now we use commandling/service based OpenVPN client for Ubuntu which is part of the very same package as the server. So you have to install the openvpn package again on the client machine:

```
sudo apt install openvpn
```

This time copy the client.conf sample config file to /etc/openvpn/.

```
sudo cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf /etc/
    openvpn/
```

Copy the following client keys and certificate files you created in the section above to e.g. /etc/openvpn/ and edit /etc/openvpn/client.conf to make sure the following lines are pointing to those files. If you have the files in /etc/openvpn/ you can omit the path.

```
ca ca.crt
cert myclient1.crt
key myclient1.key
tls-auth ta.key 1
```

And you have to specify the OpenVPN server name or address. Make sure the keyword client is in the config. That's what enables client mode.

```
client
remote vpnserver.example.com 1194
```

Now start the OpenVPN client with the same templatized mechanism:

```
$ sudo systemctl start openvpn@client
```

You can check status as you did on the server:

```
$ sudo systemctl status openvpn@client
openvpn@client.service - OpenVPN connection to client
   Loaded: loaded (/lib/systemd/system/openvpn@.service; disabled; vendor
      preset: enabled)
   Active: active (running) since Thu 2019-10-24 11:42:35 UTC; 6s ago
     Docs: man:openvpn(8)
           https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage
           https://community.openvpn.net/openvpn/wiki/HOWTO
 Main PID: 3616 (openvpn)
   Status: "Initialization Sequence Completed"
    Tasks: 1 (limit: 533)
   Memory: 1.3M
   CGroup: /system.slice/system-openvpn.slice/openvpn@client.service
```

```
          3616 /usr/sbin/openvpn --daemon ovpn-client --status /run/openvpn/
              client.status 10 --cd /etc/openvpn --script-security 2 --config
               /etc/openvpn/client.conf --writepid /run/openvp

Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: Outgoing Data Channel:
    Cipher 'AES-256-GCM' initialized with 256 bit key
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: Incoming Data Channel:
    Cipher 'AES-256-GCM' initialized with 256 bit key
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: ROUTE_GATEWAY
    192.168.122.1/255.255.255.0 IFACE=ens3 HWADDR=52:54:00:3c:5a:88
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: TUN/TAP device tun0 opened
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: TUN/TAP TX queue length set
     to 100
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: /sbin/ip link set dev tun0
    up mtu 1500
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: /sbin/ip addr add dev tun0
    local 10.8.0.6 peer 10.8.0.5
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: /sbin/ip route add
    10.8.0.1/32 via 10.8.0.5
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: WARNING: this configuration
     may cache passwords in memory -- use the auth-nocache option to prevent
    this
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: Initialization Sequence
    Completed
```

On the server log an incoming connection looks like the following. You can see client name and source address as well as success/failure messages.

```
ovpn-myserver[4818]: 192.168.122.114:55738 TLS: Initial packet from [AF_INET
    ]192.168.122.114:55738, sid=5e943ab8 40ab9fed
ovpn-myserver[4818]: 192.168.122.114:55738 VERIFY OK: depth=1, CN=Easy-RSA CA
ovpn-myserver[4818]: 192.168.122.114:55738 VERIFY OK: depth=0, CN=myclient1
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_VER=2.4.7
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_PLAT=linux
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_PROTO=2
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_NCP=2
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_LZ4=1
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_LZ4v2=1
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_LZO=1
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_COMP_STUB=1
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_COMP_STUBv2=1
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_TCPNL=1
ovpn-myserver[4818]: 192.168.122.114:55738 Control Channel: TLSv1.3, cipher
    TLSv1.3 TLS_AES_256_GCM_SHA384, 2048 bit RSA
ovpn-myserver[4818]: 192.168.122.114:55738 [myclient1] Peer Connection
    Initiated with [AF_INET]192.168.122.114:55738
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 MULTI_sva: pool returned
    IPv4=10.8.0.6, IPv6=(Not enabled)
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 MULTI: Learn: 10.8.0.6 ->
     myclient1/192.168.122.114:55738
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 MULTI: primary virtual IP
     for myclient1/192.168.122.114:55738: 10.8.0.6
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 PUSH: Received control
    message: 'PUSH_REQUEST'
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 SENT CONTROL [myclient1]:
```

```
    'PUSH_REPLY,route 10.8.0.1,topology net30,ping 10,ping-restart 120,
    ifconfig 10.8.0.6 10.8.0.5,peer-id 0,cipher AES-256-GCM' (status=1)
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 Data Channel: using
    negotiated cipher 'AES-256-GCM'
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 Outgoing Data Channel:
    Cipher 'AES-256-GCM' initialized with 256 bit key
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 Incoming Data Channel:
    Cipher 'AES-256-GCM' initialized with 256 bit key
```

And you can check on the client if it created a tun0 interface:

```
$ ip addr show dev tun0
4: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel
    state UNKNOWN group default qlen 100
    link/none
    inet 10.8.0.6 peer 10.8.0.5/32 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::5a94:ae12:8901:5a75/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

Check if you can ping the OpenVPN server:

```
root@client:/etc/openvpn# ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_req=1 ttl=64 time=0.920 ms
```

> **Note**
>
> The OpenVPN server always uses the first usable IP address in the client network and only that IP is pingable. E.g. if you configured a /24 for the client network mask, the .1 address will be used. The P-t-P address you see in the ifconfig output above is usually not answering ping requests.

Check out your routes:

```
$ ip route
default via 192.168.122.1 dev ens3 proto dhcp src 192.168.122.114 metric 100
10.8.0.1 via 10.8.0.5 dev tun0
10.8.0.5 dev tun0 proto kernel scope link src 10.8.0.6
192.168.122.0/24 dev ens3 proto kernel scope link src 192.168.122.114
192.168.122.1 dev ens3 proto dhcp scope link src 192.168.122.114 metric 100
```

First trouble shooting

If the above didn't work for you, check this:

- Check your journal -xe
- Check that you have specified the keyfile names correctly in client and server conf files
- Can the client connect to the server machine? Maybe a firewall is blocking access? Check journal on server.
- Client and server must use same protocol and port, e.g. UDP port 1194, see port and proto config option
- Client and server must use same config regarding compression, see comp-lzo config option
- Client and server must use same config regarding bridged vs routed mode, see server vs server-bridge config option

Advanced configuration

Advanced routed VPN configuration on server

The above is a very simple working VPN. The client can access services on the VPN server machine through an encrypted tunnel. If you want to reach more servers or anything in other networks, push some routes to the clients. E.g. if your company's network can be summarized to the network 192.168.0.0/16, you could push this route to the clients. But you will also have to change the routing for the way back - your servers need to know a route to the VPN client-network.

The example config files that we have been using in this guide are full of all these advanced options in the form of a comment and a disabled configuration line as an example.

> **Note**
>
> Please read the OpenVPN hardening security guide for further security advice.

Advanced bridged VPN configuration on server

OpenVPN can be setup for either a routed or a bridged VPN mode. Sometimes this is also referred to as OSI layer-2 versus layer-3 VPN. In a bridged VPN all layer-2 frames - e.g. all ethernet frames - are sent to the VPN partners and in a routed VPN only layer-3 packets are sent to VPN partners. In bridged mode all traffic including traffic which was traditionally LAN-local like local network broadcasts, DHCP requests, ARP requests etc. are sent to VPN partners whereas in routed mode this would be filtered.

Prepare interface config for bridging on server First, use netplan to configure a bridge device using the desired ethernet device.

```
$ cat /etc/netplan/01-netcfg.yaml
network:
    version: 2
    renderer: networkd
    ethernets:
        enp0s31f6:
            dhcp4: no
    bridges:
        br0:
            interfaces: [enp0s31f6]
            dhcp4: no
            addresses: [10.0.1.100/24]
            gateway4: 10.0.1.1
            nameservers:
                addresses: [10.0.1.1]
```

Static IP addressing is highly suggested. DHCP addressing can also work, but you will still have to encode a static address in the OpenVPN configuration file.

The next step on the server is to configure the ethernet device for promiscuous mode on boot. To do this, ensure the networkd-dispatcher package is installed and create the following configuration script.

```
sudo apt update
sudo apt install networkd-dispatcher
sudo touch /usr/lib/networkd-dispatcher/dormant.d/promisc_bridge
sudo chmod +x /usr/lib/networkd-dispatcher/dormant.d/promisc_bridge
```

Then add the following contents.

```
#!/bin/sh
set -e
if [ "$IFACE" = br0 ]; then
    # no networkd-dispatcher event for 'carrier' on the physical interface
    ip link set enp0s31f6 up promisc on
fi
```

Prepare server config for bridging Edit /etc/openvpn/server.conf to use tap rather than tun and set the server to use the server-bridge directive:

```
;dev tun
dev tap
;server 10.8.0.0 255.255.255.0
server-bridge 10.0.0.4 255.255.255.0 10.0.0.128 10.0.0.254
```

After configuring the server, restart openvpn by entering:

```
sudo systemctl restart openvpn@myserver
```

Prepare client config for bridging The only difference on the client side for bridged mode to what was outlined above is that you need to edit /etc/openvpn/client.conf and set tap mode:

```
dev tap
;dev tun
```

Finally, restart openvpn:

```
sudo systemctl restart openvpn@client
```

You should now be able to connect to the full remote LAN through the VPN.

References

- EasyRSA
- OpenVPN quick start guide
- Snap'ed version of openvpn easy-openvpn
- Debians OpenVPN Guide

Installing a gitolite server

Gitolite provides a traditional source control management server for git, with multiple users and access rights management. gitolite can be installed with the following command:

```
sudo apt install gitolite3
```

Gitolite configuration

Configuration of the gitolite server is a little different that most other servers on Unix-like systems, in that gitolite stores its configuration in a git repository rather than in files in /etc/. The first step to configuring a new installation is therefore to allow access to the configuration repository.

First of all, let's create a user for gitolite to use for the service:

```
sudo adduser --system --shell /bin/bash --group --disabled-password --home /
    home/git git
```

Now we want to let gitolite know about the repository administrator's public SSH key. This assumes that the current user is the repository administrator. If you have not yet configured an SSH key, refer to openssh-keys in this manual.

```
cp ~/.ssh/id_rsa.pub /tmp/$(whoami).pub
```

Let's switch to the git user and import the administrator's key into gitolite.

```
sudo su - git
gl-setup /tmp/*.pub
```

Gitolite will allow you to make initial changes to its configuration file during the setup process. You can now clone and modify the gitolite configuration repository from your administrator user (the user whose public SSH key you imported). Switch back to that user, then clone the configuration repository:

```
exit
git clone git@$IP_ADDRESS:gitolite-admin.git
cd gitolite-admin
```

The gitolite-admin contains two subdirectories, "conf" and "keydir". The configuration files are in the conf dir, and the keydir directory contains the list of user's public SSH keys.

Managing gitolite users and repositories

Adding a new user to gitolite is simple: just obtain their public SSH key and add it to the keydir directory as $DESIRED_USER_NAME.pub. Note that the gitolite usernames don't have to match the system usernames - they are only used in the gitolite configuration file to manage access control. Similarly, users are deleted by deleting their public key files. After each change, do not forget to commit the changes to git, and push the changes back to the server with

```
git commit -a
git push origin master
```

Repositories are managed by editing the conf/ gitolite .conf file. The syntax is space separated, and simply specifies the list of repositories followed by some access rules. The following is a default example

```
repo    gitolite-admin
        RW+     =   admin
        R       =   alice

repo    project1
        RW+     =   alice
        RW      =   bob
        R       =   denise
```

Using your server

Once a user's public key has been imported by the gitolite admin and authorization granted to the user to one or more repositories, the user can access repositories with the following command:

```
git clone git@$SERVER_IP:$PROJECT_NAME.git
```

To add the server as a new remote for an existing git repository:

```
git remote add gitolite git@$SERVER_IP:$PROJECT_NAME.git
```

References

- Gitolite's code repository provides access to source code.
- Gitolite's documentation includes a "fool proof setup" guide and a cookbook with recipes for common tasks.
- Gitolite's maintainer has written a book, *Gitolite Essentials*, for more in-depth information about the software.
- General information about git itself can be found at the Git homepage.

Client software implementations

Linux Network-Manager GUI for OpenVPN

Many Linux distributions including Ubuntu desktop variants come with Network Manager, a nice GUI to configure your network settings. It also can manage your VPN connections. It is the default, but if in doubt make sure you have package network−manager−openvpn installed.

Open the Network Manager GUI, select the VPN tab and then the 'Add' button. Select OpenVPN as the VPN type in the opening requester and press 'Create'. In the next window add the OpenVPN's server name as the 'Gateway', set 'Type' to 'Certificates (TLS)', point 'User Certificate' to your user certificate, 'CA Certificate' to your CA certificate and 'Private Key' to your private key file. Use the advanced button to enable compression (e.g. comp-lzo), dev tap, or other special settings you set on the server. Now try to establish your VPN.

OpenVPN with GUI for Mac OS X

- Tunnelblick is an excellent free, open source implementation of a GUI for OpenVPN for OS X. Download the latest OS X installer from there and install it. It also is recommended by upstream which would have a alternative on their own

Then put your client.ovpn config file together with the certificates and keys in /Users/username/Library/Application Support/Tunnelblick/Configurations/ and lauch Tunnelblick from your Application folder.

Instead of downloading manually, if you have brew set up on MacOS this is as easy as:

```
brew cask install tunnelblick
```

OpenVPN with GUI for Win

First download and install the latest OpenVPN Windows Installer. As of this writing, the management GUI is included with the Windows binary installer.

You need to start the OpenVPN service. Goto Start > Computer > Manage > Services and Applications > Services. Find the OpenVPN service and start it. Set it's startup type to automatic.

When you start the OpenVPN MI GUI the first time you need to run it as an administrator. You have to right click on it and you will see that option.

There is an updated guide by the upstream project for the client on Windows.

References

- See the OpenVPN website for additional information.
- OpenVPN hardening security guide
- Also, Pakt's OpenVPN: Building and Integrating Virtual Private Networks is a good resource.

SSSD

SSSD stands for System Security Services Daemon and it's actually a collection of daemons that handle authentication, authorization, and user and group information from a variety of network sources. At its core it has support for: - Active Directory - LDAP - Kerberos

SSSD provides PAM and NSS modules to integrate these remote sources into your system and allow remote users to login and be recognized as valid users, including group membership. To allow for disconnected operation, SSSD also can also cache this information, so that users can continue to login in the event of a network failure, or other problem of the same sort.

This guide will focus on the most common scenarios where SSSD is deployed.

SSSD and Active Directory

This section describes the use of sssd to authenticate user logins against an Active Directory via using sssd's "ad" provider. At the end, Active Directory users will be able to login on the host using their AD credentials. Group membership will also be maintained.

Prerequisites, Assumptions, and Requirements

- This guide does not explain Active Directory, how it works, how to set one up, or how to maintain it.
- This guide assumes that a working Active Directory domain is already configured and you have access to the credentials to join a machine to that domain.
- The domain controller is acting as an authoritative DNS server for the domain.
- The domain controller is the primary DNS resolver (check with systemd−resolve −−status)
- System time is correct and in sync, maintained via a service like *chrony* or *ntp*
- The domain used in this example is *ad1.example.com* .

Software Installation

Install the following packages:

```
sudo apt install sssd-ad sssd-tools realmd adcli
```

Join the domain

We will use the realm command, from the realmd package, to join the domain and create the sssd configuration.

Let's verify the domain is discoverable via DNS:

```
$ sudo realm -v discover ad1.example.com
 * Resolving: _ldap._tcp.ad1.example.com
 * Performing LDAP DSE lookup on: 10.51.0.5
 * Successfully discovered: ad1.example.com
ad1.example.com
  type: kerberos
  realm-name: AD1.EXAMPLE.COM
  domain-name: ad1.example.com
  configured: no
  server-software: active-directory
  client-software: sssd
  required-package: sssd-tools
  required-package: sssd
  required-package: libnss-sss
  required-package: libpam-sss
  required-package: adcli
  required-package: samba-common-bin
```

This performs several checks and determines the best software stack to use with sssd. sssd can install the missing packages via *packagekit*, but we installed them already previously.

Now let's join the domain:

```
$ sudo realm join ad1.example.com
Password for Administrator:
```

That was quite uneventful. If you want to see what it was doing, pass the -v option:

```
$ sudo realm join -v ad1.example.com
 * Resolving: _ldap._tcp.ad1.example.com
 * Performing LDAP DSE lookup on: 10.51.0.5
 * Successfully discovered: ad1.example.com
Password for Administrator:
 * Unconditionally checking packages
 * Resolving required packages
 * LANG=C /usr/sbin/adcli join --verbose --domain ad1.example.com --domain-
     realm AD1.EXAMPLE.COM --domain-controller 10.51.0.5 --login-type user --
     login-user Administrator --stdin-password
 * Using domain name: ad1.example.com
 * Calculated computer account name from fqdn: AD-CLIENT
 * Using domain realm: ad1.example.com
 * Sending NetLogon ping to domain controller: 10.51.0.5
 * Received NetLogon info from: SERVER1.ad1.example.com
 * Wrote out krb5.conf snippet to /var/cache/realmd/adcli-krb5-hUfTUg/krb5.d/
     adcli-krb5-conf-hv2kzi
 * Authenticated as user: Administrator@AD1.EXAMPLE.COM
 * Looked up short domain name: AD1
 * Looked up domain SID: S-1-5-21-2660147319-831819607-3409034899
 * Using fully qualified name: ad-client.ad1.example.com
 * Using domain name: ad1.example.com
 * Using computer account name: AD-CLIENT
 * Using domain realm: ad1.example.com
 * Calculated computer account name from fqdn: AD-CLIENT
 * Generated 120 character computer password
 * Using keytab: FILE:/etc/krb5.keytab
```

```
* Found computer account for AD-CLIENT$ at: CN=AD-CLIENT,CN=Computers,DC=ad1,
    DC=example,DC=com
* Sending NetLogon ping to domain controller: 10.51.0.5
* Received NetLogon info from: SERVER1.ad1.example.com
* Set computer password
* Retrieved kvno '3' for computer account in directory: CN=AD-CLIENT,CN=
    Computers,DC=ad1,DC=example,DC=com
* Checking RestrictedKrbHost/ad-client.ad1.example.com
*    Added RestrictedKrbHost/ad-client.ad1.example.com
* Checking RestrictedKrbHost/AD-CLIENT
*    Added RestrictedKrbHost/AD-CLIENT
* Checking host/ad-client.ad1.example.com
*    Added host/ad-client.ad1.example.com
* Checking host/AD-CLIENT
*    Added host/AD-CLIENT
* Discovered which keytab salt to use
* Added the entries to the keytab: AD-CLIENT$@AD1.EXAMPLE.COM: FILE:/etc/krb5
    .keytab
* Added the entries to the keytab: host/AD-CLIENT@AD1.EXAMPLE.COM: FILE:/etc/
    krb5.keytab
* Added the entries to the keytab: host/ad-client.ad1.example.com@AD1.EXAMPLE
    .COM: FILE:/etc/krb5.keytab
* Added the entries to the keytab: RestrictedKrbHost/AD-CLIENT@AD1.EXAMPLE.
    COM: FILE:/etc/krb5.keytab
* Added the entries to the keytab: RestrictedKrbHost/ad-client.ad1.example.
    com@AD1.EXAMPLE.COM: FILE:/etc/krb5.keytab
* /usr/sbin/update-rc.d sssd enable
* /usr/sbin/service sssd restart
* Successfully enrolled machine in realm
```

By default, *realm* will use the *Administrator* account of the domain to request the join. If you need to use another account, pass it to the tool with the −U option.

Another popular way of joining a domain is using an *OTP*, or *One Time Password*, token. For that, use the −−one−time−password option.

SSSD Configuration

The *realm* tool already took care of creating an sssd configuration, adding the pam and nss modules, and starting the necessary services.

Let's take a look at /etc/sssd/sssd.conf:

```
[sssd]
domains = ad1.example.com
config_file_version = 2
services = nss, pam

[domain/ad1.example.com]
default_shell = /bin/bash
krb5_store_password_if_offline = True
cache_credentials = True
krb5_realm = AD1.EXAMPLE.COM
realmd_tags = manages-system joined-with-adcli
id_provider = ad
```

```
fallback_homedir = /home/%u@%d
ad_domain = ad1.example.com
use_fully_qualified_names = True
ldap_id_mapping = True
access_provider = ad
```

> **Note**
>
> Something very important to remember is that this file must have permissions *0600* and ownership *root:root*, or else sssd won't start!

Let's highlight a few things from this config:

- *cache_credentials*: this allows logins when the AD server is unreachable
- home directory: it's by default /home/<user>@<domain>. For example, the AD user *john* will have a home directory of */home/john@ad1.example.com*
- *use_fully_qualified_names*: users will be of the form *user@domain*, not just *user*. This should only be changed if you are certain no other domains will ever join the AD forest, via one of the several possible trust relationships

Automatic home directory creation

What the realm tool didn't do for us is setup pam_mkhomedir, so that network users can get a home directory when they login. This remaining step can be done by running the following command:

```
sudo pam-auth-update --enable mkhomedir
```

Checks

You should now be able to fetch information about AD users. In this example, *John Smith* is an AD user:

```
$ getent passwd john@ad1.example.com
john@ad1.example.com:*:1725801106:1725800513:John Smith:/home/john@ad1.example
    .com:/bin/bash
```

Let's see his groups:

```
$ groups john@ad1.example.com
john@ad1.example.com : domain users@ad1.example.com engineering@ad1.example.
    com
```

> **Note**
>
> If you just changed the group membership of a user, it may be a while before sssd notices due to caching.

Finally, how about we try a login:

```
$ sudo login
ad-client login: john@ad1.example.com
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-24-generic x86_64)
...
Creating directory '/home/john@ad1.example.com'.
john@ad1.example.com@ad-client:~$
```

Notice how the home directory was automatically created.

You can also use ssh, but note that the command will look a bit funny because of the multiple *@* signs:

```
$ ssh john@ad1.example.com@10.51.0.11
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-24-generic x86_64)
(...)
Last login: Thu Apr 16 21:22:55 2020
john@ad1.example.com@ad-client:~$
```

> **Note**
>
> In the ssh example, public key authentication was used, so no password was required. Remember that ssh password authentication is by default disabled in /etc/ssh/sshd_config.

Kerberos Tickets

If you install krb5-user, your AD users will also get a kerberos ticket upon logging in:

```
john@ad1.example.com@ad-client:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1725801106_9UxVIz
Default principal: john@AD1.EXAMPLE.COM

Valid starting     Expires            Service principal
04/16/20 21:32:12  04/17/20 07:32:12  krbtgt/AD1.EXAMPLE.COM@AD1.EXAMPLE.COM
    renew until 04/17/20 21:32:12
```

> **Note**
>
> *realm* also configured /etc/krb5.conf for you, so there should be no further configuration prompts when installing krb5-user

Let's test with *smbclient* using kerberos authentication to list he shares of the domain controller:

```
john@ad1.example.com@ad-client:~$ smbclient -k -L server1.ad1.example.com

    Sharename       Type      Comment
    ---------       ----      -------
    ADMIN$          Disk      Remote Admin
    C$              Disk      Default share
    IPC$            IPC       Remote IPC
    NETLOGON        Disk      Logon server share
    SYSVOL          Disk      Logon server share
SMB1 disabled -- no workgroup available
```

Notice how we now have a ticket for the *cifs* service, which was used for the share list above:

```
john@ad1.example.com@ad-client:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1725801106_9UxVIz
Default principal: john@AD1.EXAMPLE.COM

Valid starting     Expires            Service principal
04/16/20 21:32:12  04/17/20 07:32:12  krbtgt/AD1.EXAMPLE.COM@AD1.EXAMPLE.COM
    renew until 04/17/20 21:32:12
04/16/20 21:32:21  04/17/20 07:32:12  cifs/server1.ad1.example.com@AD1.EXAMPLE
    .COM
```

Desktop Ubuntu Authentication

The desktop login only shows local users in the list to pick from, and that's on purpose.

To login with an Active Directory user for the first time, follow these steps:

- click on the "Not listed?" option:

click-not-listed|690x517,50%

- type in the login name followed by the password:

type-in-username|690x517,50%

- the next time you login, the AD user will be listed as if it was a local user:

next-time|690x517,50%

Resources

- GitHub SSSD Project
- Active Directory DNS Zone Entries

SSSD and LDAP

SSSD can also use LDAP for authentication, authorization, and user/group information. In this section we will configure a host to authenticate users from an OpenLDAP directory.

Prerequisites, Assumptions, and Requirements

For this setup, we need:

- an existing OpenLDAP server with SSL enabled and using the RFC2307 schema for users and groups
- a client host where we will install the necessary tools and login as an user from the LDAP server

Software Installation

Install the following packages:

```
sudo apt install sssd-ldap ldap-utils
```

SSSD Configuration

Create the /etc/sssd/sssd.conf configuration file, with permissions *0600* and ownership *root:root*, and this content:

```
[sssd]
config_file_version = 2
domains = example.com

[domain/example.com]
id_provider = ldap
auth_provider = ldap
ldap_uri = ldap://ldap01.example.com
```

```
cache_credentials = True
ldap_search_base = dc=example,dc=com
```

Make sure to start the *sssd* service:

```
sudo systemctl start sssd.service
```

> **Note**
>
> *sssd* will use *START_TLS* by default for authentication requests against the LDAP server (the *auth_provider*), but not for the *id_provider*. If you want to also enable *START_TLS* for the *id_provider*, specify ldap_id_use_start_tls = true.

Automatic home directory creation

To enable automatic home directory creation, run the following command:

```
sudo pam-auth-update --enable mkhomedir
```

Check SSL setup on the client

The client must be able to use *START_TLS* when connecting to the LDAP server, with full certificate checking. This means:

- the client host knows and trusts the CA that signed the LDAP server certificate
- the server certificate was issued for the correct host (ldap01.example.com in this guide)
- the time is correct on all hosts performing the TLS connection
- and, of course, that neither certificate (CA or server's) expired

If using a custom CA, an easy way to have a host trust it is to place it in /usr/local/share/ca-certificates/ with a .crt extension and run sudo update-ca-certificates.

Alternatively, you can edit /etc/ldap/ldap.conf and point TLS_CACERT to the CA public key file.

> **Note**
>
> You may have to restart sssd after these changes: sudo systemctl restart sssd

Once that is all done, check that you can connect to the LDAP server using verified SSL connections:

```
$ ldapwhoami -x -ZZ -h ldap01.example.com
anonymous
```

The -ZZ parameter tells the tool to use *START_TLS*, and that it must not fail. If you have LDAP logging enabled on the server, it will show something like this:

```
slapd[779]: conn=1032 op=0 STARTTLS
slapd[779]: conn=1032 op=0 RESULT oid= err=0 text=
slapd[779]: conn=1032 fd=15 TLS established tls_ssf=256 ssf=256
slapd[779]: conn=1032 op=1 BIND dn="" method=128
slapd[779]: conn=1032 op=1 RESULT tag=97 err=0 text=
slapd[779]: conn=1032 op=2 EXT oid=1.3.6.1.4.1.4203.1.11.3
slapd[779]: conn=1032 op=2 WHOAMI
slapd[779]: conn=1032 op=2 RESULT oid= err=0 text=
```

START_TLS with *err=0* and *TLS established* is what we want to see there, and, of course, the *WHOAMI* extended operation.

Final verification

In this example, the LDAP server has the following user and group entry we are going to use for testing:

```
dn: uid=john,ou=People,dc=example,dc=com
uid: john
objectClass: inetOrgPerson
objectClass: posixAccount
cn: John Smith
sn: Smith
givenName: John
mail: john@example.com
userPassword: johnsecret
uidNumber: 10001
gidNumber: 10001
loginShell: /bin/bash
homeDirectory: /home/john

dn: cn=john,ou=Group,dc=example,dc=com
cn: john
objectClass: posixGroup
gidNumber: 10001
memberUid: john

dn: cn=Engineering,ou=Group,dc=example,dc=com
cn: Engineering
objectClass: posixGroup
gidNumber: 10100
memberUid: john
```

The user *john* should be known to the system:

```
ubuntu@ldap-client:~$ getent passwd john
john:*:10001:10001:John Smith:/home/john:/bin/bash

ubuntu@ldap-client:~$ id john
uid=10001(john) gid=10001(john) groups=10001(john),10100(Engineering)
```

And we should be able to authenticate as *john*:

```
ubuntu@ldap-client:~$ sudo login
ldap-client login: john
Password:
Welcome to Ubuntu Focal Fossa (development branch) (GNU/Linux 5.4.0-24-generic
    x86_64)
(...)
Creating directory '/home/john'.
john@ldap-client:~$
```

SSSD, LDAP and Kerberos

Finally, we can mix it all together in a setup that is very similar to Active Directory in terms of the technologies used: use LDAP for users and groups, and Kerberos for authentication.

Prerequisites, Assumptions, and Requirements

For this setup, we will need:

- an existing OpenLDAP server using the RFC2307 schema for users and groups. SSL support is recommended, but not strictly necessary because authentication in this setup is being done via Kerberos, and not LDAP.
- a Kerberos server. It doesn't have to be using the OpenLDAP backend
- a client host where we will install and configure SSSD

Software Installation

On the client host, install the following packages:

```
sudo apt install sssd-ldap sssd-krb5 ldap-utils krb5-user
```

You may be asked about the default Kerberos realm. For this guide, we are using EXAMPLE.COM.

At this point, you should alreaedy be able to obtain tickets from your Kerberos server, assuming DNS records point at it like explained elsewhere in this guide:

```
$ kinit ubuntu
Password for ubuntu@EXAMPLE.COM:

ubuntu@ldap-krb-client:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: ubuntu@EXAMPLE.COM

Valid starting     Expires            Service principal
04/17/20 19:51:06  04/18/20 05:51:06  krbtgt/EXAMPLE.COM@EXAMPLE.COM
    renew until 04/18/20 19:51:05
```

But we want to be able to login as an LDAP user, authenticated via Kerberos. Let's continue with the configuration.

SSSD Configuration

Create the /etc/sssd/sssd.conf configuration file, with permissions *0600* and ownership *root:root*, and this content:

```
[sssd]
config_file_version = 2
domains = example.com

[domain/example.com]
id_provider = ldap
ldap_uri = ldap://ldap01.example.com
ldap_search_base = dc=example,dc=com
auth_provider = krb5
krb5_server = kdc01.example.com,kdc02.example.com
krb5_kpasswd = kdc01.example.com
krb5_realm = EXAMPLE.COM
cache_credentials = True
```

This example uses two KDCs, which made it necessary to also specify the *krb5_kpasswd* server because the second KDC is a replica and is not running the admin server.

Start the *sssd* service:

```
sudo systemctl start sssd.service
```

Automatic home directory creation

To enable automatic home directory creation, run the following command:

```
sudo pam-auth-update --enable mkhomedir
```

Final verification

In this example, the LDAP server has the following user and group entry we are going to use for testing:

```
dn: uid=john,ou=People,dc=example,dc=com
uid: john
objectClass: inetOrgPerson
objectClass: posixAccount
cn: John Smith
sn: Smith
givenName: John
mail: john@example.com
uidNumber: 10001
gidNumber: 10001
loginShell: /bin/bash
homeDirectory: /home/john

dn: cn=john,ou=Group,dc=example,dc=com
cn: john
objectClass: posixGroup
gidNumber: 10001
memberUid: john

dn: cn=Engineering,ou=Group,dc=example,dc=com
cn: Engineering
objectClass: posixGroup
gidNumber: 10100
memberUid: john
```

Note how the *john* user has no *userPassword* attribute.

The user *john* should be known to the system:

```
ubuntu@ldap-client:~$ getent passwd john
john:*:10001:10001:John Smith:/home/john:/bin/bash

ubuntu@ldap-client:~$ id john
uid=10001(john) gid=10001(john) groups=10001(john),10100(Engineering)
```

Let's try a login as this user:

```
ubuntu@ldap-krb-client:~$ sudo login
ldap-krb-client login: john
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-24-generic x86_64)
(...)
Creating directory '/home/john'.

john@ldap-krb-client:~$ klist
Ticket cache: FILE:/tmp/krb5cc_10001_BOrxWr
Default principal: john@EXAMPLE.COM

Valid starting     Expires            Service principal
04/17/20 20:29:50  04/18/20 06:29:50  krbtgt/EXAMPLE.COM@EXAMPLE.COM
    renew until 04/18/20 20:29:50
john@ldap-krb-client:~$
```

We logged in using the kerberos password, and user/group information from the LDAP server.

SSSD and KDC spoofing

When using SSSD to manage kerberos logins on a Linux host, there is an attack scenario you should be aware of: KDC spoofing.

The objective of the attacker is to login on a workstation that is using Kerberos authentication. Let's say he knows john is a valid user on that machine.

The attacker first deploys a rogue KDC server in the network, and creates the john principal there with a password of his choosing. What he has to do now is to have his rogue KDC respond to the login request from the workstation, before (or instead of) the real KDC. If the workstation isn't authenticating the KDC, it will accept the reply from the rogue server and let john in.

There is a configuration parameter that can be set to protect the workstation from this attack. It will have SSSD authenticate the KDC, and block the login if the KDC cannot be verified. This option is called krb5_validate, and it's false by default.

To enable it, edit /etc/sssd/sssd.conf and add this line to the domain section:

```
[sssd]
config_file_version = 2
domains = example.com

[domain/example.com]
id_provider = ldap
...
krb5_validate = True
```

The second step is to create a host principal on the KDC for this workstation. This is how the KDC's authenticity is verified. It's like a "machine account", with a shared secret that the attacker cannot control and replicate in his rogue KDC..The host principal has the format host/<fqdn>@REALM.

After the host principal is created, its keytab needs to be stored on the workstation. This two step process can be easily done on the workstation itself via kadmin (not kadmin.local) to contact the KDC remotely:

```
$ sudo kadmin -p ubuntu/admin
kadmin:  addprinc -randkey host/ldap-krb-client.example.com@EXAMPLE.COM
WARNING: no policy specified for host/ldap-krb-client.example.com@EXAMPLE.COM;
    defaulting to no policy
```

```
Principal "host/ldap-krb-client.example.com@EXAMPLE.COM" created.

kadmin:  ktadd -k /etc/krb5.keytab host/ldap-krb-client.example.com
Entry for principal host/ldap-krb-client.example.com with kvno 6, encryption
    type aes256-cts-hmac-sha1-96 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/ldap-krb-client.example.com with kvno 6, encryption
    type aes128-cts-hmac-sha1-96 added to keytab WRFILE:/etc/krb5.keytab.
```

Then exit the tool and make sure the permissions on the keytab file are tight:

```
sudo chmod 0600 /etc/krb5.keytab
sudo chown root:root /etc/krb5.keytab
```

You can also do it on the KDC itself using kadmin.local, but you will have to store the keytab temporarily in another file and securely copy it over to the workstation.

Once these steps are complete, you can restart sssd on the workstation and perform the login. If the rogue KDC picks the attempt up and replies, it will fail the host verification. With debugging we can see that happening on the workstation:

```
==> /var/log/sssd/krb5_child.log <==
(Mon Apr 20 19:43:58 2020) [[sssd[krb5_child[2102]]]] [validate_tgt] (0x0020):
    TGT failed verification using key for [host/ldap-krb-client.example.
    com@EXAMPLE.COM].
(Mon Apr 20 19:43:58 2020) [[sssd[krb5_child[2102]]]] [get_and_save_tgt] (0
    x0020): 1741: [-1765328377][Server host/ldap-krb-client.example.
    com@EXAMPLE.COM not found in Kerberos database]
```

And the login is denied. If the real KDC picks it up, however, the host verification succeeds:

```
==> /var/log/sssd/krb5_child.log <==
(Mon Apr 20 19:46:22 2020) [[sssd[krb5_child[2268]]]] [validate_tgt] (0x0400):
    TGT verified using key for [host/ldap-krb-client.example.com@EXAMPLE.COM
    ].
```

And the login is accepted.

Debugging and troubleshooting

Here are some tips to help troubleshoot sssd.

`debug_level`

The debug level of sssd can be changed on-the-fly via `sssctl`, from the sssd-tools package:

```
sudo apt install sssd-tools
sssctl debug-level <new-level>
```

Or change add it to the config file and restart sssd:

```
[sssd]
config_file_version = 2
domains = example.com

[domain/example.com]
debug_level = 6
...
```

Either will yield more logs in /var/log/sssd/*.log and can help identify what is going on. The sssctl approach has the clear advantage of not having to restart the service.

Caching

Caching is useful to speed things up, but it can get in the way big time when troubleshooting. It's useful to be able to remove the cache while chasing down a problem. This can also be done with the sssctl tool from the sssd-tools package.

You can either remove the whole cache:

```
# sssctl cache-remove
Creating backup of local data...
SSSD backup of local data already exists, override? (yes/no) [no] yes
Removing cache files...
SSSD= needs to be running. Start SSSD now? (yes/no) [yes] yes
```

Or just one element:

```
sssctl cache-expire -u john
```

Or expire everything:

```
sssctl cache-expire -E
```

> 429 from API, waiting 45 seconds … ('You've performed this action too many times. Please wait 44 seconds before trying again.') # IRC Server

The Ubuntu repository has many Internet Relay Chat servers. This section explains how to install and configure the original IRC server ircd-irc2.

Installation

To install ircd-irc2, run the following command in the command prompt:

```
sudo apt install ircd-irc2
```

The configuration files are stored in /etc/ircd directory. The documents are available in /usr/share/doc/ircd-irc2 directory.

Configuration

The IRC settings can be done in the configuration file /etc/ircd/ircd.conf. You can set the IRC host name in this file by editing the following line:

```
M: irc.localhost::Debian ircd default configuration::000A
```

Please make sure you add DNS aliases for the IRC host name. For instance, if you set irc.livecipher.com as IRC host name, please make sure irc.livecipher.com is resolvable in your Domain Name Server. The IRC host name should not be same as the host name.

The IRC admin details can be configured by editing the following line:

```
A: Organization, IRC dept.:Daemon <ircd@example.irc.org>:Client Server::IRCnet:
```

You should add specific lines to configure the list of IRC ports to listen on, to configure Operator credentials, to configure client authentication, etc. For details, please refer to the example configuration file /usr/share/doc/ircd-irc2/ircd.conf.example.gz.

The IRC banner to be displayed in the IRC client, when the user connects to the server can be set in /etc/ircd/ircd.motd file.

After making necessary changes to the configuration file, you can restart the IRC server using following command:

```
sudo systemctl restart ircd-irc2.service
```

References

You may also be interested to take a look at other IRC servers available in Ubuntu Repository. It includes, ircd-ircu and ircd-hybrid.

- Refer to IRCD FAQ for more details about the IRC Server.

Dovecot Server

Dovecot is a Mail Delivery Agent, written with security primarily in mind. It supports the major mailbox formats: mbox or Maildir. This section explains how to set it up as an IMAP or POP3 server.

Installation

To install a basic Dovecot server with common POP3 and IMAP functions, run the following command:

```
sudo apt install dovecot-imapd dovecot-pop3d
```

There are various other Dovecot modules including dovecot-sieve (mail filtering), dovecot-solr (full text search), dovecot-antispam (spam filter training), dovecot-ldap (user directory).

Configuration

To configure Dovecot, edit the file /etc/dovecot/dovecot.conf and its included config files in /etc/dovecot/conf.d/. By default all installed protocols will be enabled via an include directive in /etc/dovecot/dovecot.conf.

```
!include_try /usr/share/dovecot/protocols.d/*.protocol
```

IMAPS and POP3S are more secure because they use SSL encryption to connect. A basic self-signed ssl certificate is automatically set up by package ssl-cert and used by Dovecot in /etc/dovecot/conf.d/10-ssl.conf.

By default mbox format is configured, if required you can also use Maildir. More about that can be found in the comments in /etc/dovecot/conf.d//10-mail.conf. Also see the Dovecot web site to learn about further benefits and details.

Make sure to also configure your Mail Transport Agent (MTA) to transfer the incoming mail to the selected type of mailbox.

Once you have configured Dovecot, restart its daemon in order to test your setup:

```
sudo service dovecot restart
```

Try to log in with the commands telnet localhost pop3 (for POP3) or telnet localhost imap2 (for IMAP). You should see something like the following:

```
bhuvan@rainbow:~$ telnet localhost pop3
Trying 127.0.0.1...
Connected to localhost.localdomain.
Escape character is '^]'.
+OK Dovecot ready.
```

Dovecot SSL Configuration

Dovecot is configured to use SSL automatically by default, using the package ssl-cert which provides a self signed certificate.

You can instead generate your own custom certificate for Dovecot using openssh, for example:

```
sudo openssl req -new -x509 -days 1000 -nodes -out "/etc/dovecot/dovecot.pem"
    \
    -keyout "/etc/dovecot/private/dovecot.pem"
```

See certificates-and-security for more details on creating custom certificates.

Then edit /etc/dovecot/conf.d/10-ssl.conf and amend following lines to specify Dovecat use these custom certificates :

```
ssl_cert = </etc/dovecot/private/dovecot.pem
ssl_key = </etc/dovecot/private/dovecot.key
```

You can get the SSL certificate from a Certificate Issuing Authority or you can create self-signed one (see certificates-and-security for details). Once you create the certificate, you will have a key file and a certificate file that you want to make known in the config shown above.

Firewall Configuration for an Email Server

To access your mail server from another computer, you must configure your firewall to allow connections to the server on the necessary ports.

- IMAP - 143
- IMAPS - 993
- POP3 - 110
- POP3S - 995

References

- See the Dovecot website for more information.
- Also, the Dovecot Ubuntu Wiki page has more details.

Exim4

Exim4 is a Message Transfer Agent (MTA) developed at the University of Cambridge for use on Unix systems connected to the Internet. Exim can be installed in place of sendmail, although its configuration is quite different.

Installation

To install exim4, run the following command:

```
sudo apt install exim4
```

Configuration

To configure Exim4, run the following command:

```
sudo dpkg-reconfigure exim4-config
```

This displays a user interface "wizard" for configuring the software. For example, in Exim4 the configuration files are split among multiple files; if you wish to have them in one file you can configure accordingly via this user interface.

All the configurable parameters from the user interface are stored in /etc/exim4/update-exim4.conf.conf file, so to re-configure you can either re-run the wizard or manually edit this file using your favorite editor. Once you are finished, you can run the following command to generate the master configuration file:

```
sudo update-exim4.conf
```

The master configuration file is stored in /var/lib/exim4/config.autogenerated.

> **Warning**
>
> At any time, you should not manually edit the master configuration file, /var/lib/exim4/config.autogenerated, because it is updated automatically every time you run update-exim4.conf, so your changes will risk being accidentally lost during a future update.

The following command will start the Exim4 daemon:

```
sudo service exim4 start
```

SMTP Authentication

Exim4 can be configured to use SMTP-AUTH with TLS and SASL.

First, enter the following into a terminal prompt to create a certificate for use with TLS:

```
sudo /usr/share/doc/exim4-base/examples/exim-gencert
```

Configure Exim4 for TLS by editing /etc/exim4/conf.d/main/03_exim4-config_tlsoptions and adding the following:

```
MAIN_TLS_ENABLE = yes
```

Second, configure Exim4 to use the saslauthd for authentication by editing /etc/exim4/conf.d/auth/30_exim4-config_examples and uncomment the *plain_saslauthd_server* and *login_saslauthd_server* sections:

```
  plain_saslauthd_server:
    driver = plaintext
    public_name = PLAIN
    server_condition = ${if saslauthd{{$auth2}{$auth3}}{1}{0}}
    server_set_id = $auth2
    server_prompts = :
    .ifndef AUTH_SERVER_ALLOW_NOTLS_PASSWORDS
    server_advertise_condition = ${if eq{$tls_cipher}{}{}{*}}
    .endif
#
  login_saslauthd_server:
    driver = plaintext
    public_name = LOGIN
    server_prompts = "Username:: : Password::"
    # don't send system passwords over unencrypted connections
    server_condition = ${if saslauthd{{$auth1}{$auth2}}{1}{0}}
    server_set_id = $auth1
    .ifndef AUTH_SERVER_ALLOW_NOTLS_PASSWORDS
    server_advertise_condition = ${if eq{$tls_cipher}{}{}{*}}
    .endif
```

Additionally, to enable outside mail clients to connect to the new server, a new user needs to be added into exim by using the following commands.

```
sudo /usr/share/doc/exim4-base/examples/exim-adduser
```

Protect the new password files with the following commands:

```
sudo chown root:Debian-exim /etc/exim4/passwd
sudo chmod 640 /etc/exim4/passwd
```

Finally, update the Exim4 configuration and restart the service:

```
sudo update-exim4.conf
sudo systemctl restart exim4.service
```

Configuring SASL

To configure the saslauthd to provide authentication for Exim4, first install the sasl2-bin package by running this command at a terminal prompt:

```
sudo apt install sasl2-bin
```

To configure saslauthd, edit the /etc/default/saslauthd configuration file and set:

```
START=yes
```

Next, to make Exim4 use the saslauthd service, the *Debian-exim* user needs to be part of the *sasl* group:

```
sudo adduser Debian-exim sasl
```

Finally, start the saslauthd service:

```
sudo service saslauthd start
```

Exim4 is now configured with SMTP-AUTH using TLS and SASL authentication.

References

- See exim.org for more information.
- There is also an Exim4 Book available.
- Another resource is the Exim4 Ubuntu Wiki page.
- Further resources to set up mailman3 with exim4

Email Services

The process of getting an email from one person to another over a network or the Internet involves many systems working together. Each of these systems must be correctly configured for the process to work. The sender uses a *Mail User Agent* (MUA), or email client, to send the message through one or more *Mail Transfer Agents* (MTA), the last of which will hand it off to a *Mail Delivery Agent* (MDA) for delivery to the recipient's mailbox, from which it will be retrieved by the recipient's email client, usually via a POP3 or IMAP server.

Mailman

Mailman is an open source program for managing electronic mail discussions and e-newsletter lists. Many open source mailing lists (including all the Ubuntu mailing lists) use Mailman as their mailing list software. It is powerful and easy to install and maintain.

Installation

Mailman provides a web interface for the administrators and users, using an external mail server to send and receive emails. It works perfectly with the following mail servers:

- Postfix
- Exim
- Sendmail
- Qmail

We will see how to install and configure Mailman with, the Apache web server, and either the Postfix or Exim mail server. If you wish to install Mailman with a different mail server, please refer to the references section.

> **Note**
>
> You only need to install one mail server and Postfix is the default Ubuntu Mail Transfer Agent.

Apache2

To install apache2 you refer to ??? for details.

Postfix

For instructions on installing and configuring Postfix refer to Postfix

Exim4

To install Exim4 refer to Exim4.

Once exim4 is installed, the configuration files are stored in the /etc/exim4 directory. In Ubuntu, by default, the exim4 configuration files are split across different files. You can change this behavior by changing the following variable in the /etc/exim4/update−exim4.conf file:

```
dc_use_split_config='true'
```

Mailman

To install Mailman, run following command at a terminal prompt:

```
sudo apt install mailman
```

It copies the installation files in /var/lib/mailman directory. It installs the CGI scripts in /usr/lib/cgi-bin/mailman directory. It creates *list* linux user. It creates the *list* linux group. The mailman process will be owned by this user.

Configuration

This section assumes you have successfully installed mailman, apache2, and postfix or exim4. Now you just need to configure them.

Apache2

An example Apache configuration file comes with Mailman and is placed in /etc/mailman/apache.conf. In order for Apache to use the config file it needs to be copied to /etc/apache2/sites−available:

```
sudo cp /etc/mailman/apache.conf /etc/apache2/sites-available/mailman.conf
```

This will setup a new Apache *VirtualHost* for the Mailman administration site. Now enable the new configuration and restart Apache:

```
sudo a2ensite mailman.conf
sudo systemctl restart apache2.service
```

Mailman uses apache2 to render its CGI scripts. The mailman CGI scripts are installed in the /usr/lib/cgi-bin/mailman directory. So, the mailman url will be http://hostname/cgi-bin/mailman/. You can make changes to the /etc/apache2/sites−available/mailman.conf file if you wish to change this behavior.

Postfix

For Postfix integration, we will associate the domain lists .example.com with the mailing lists. Please replace *lists .example.com* with the domain of your choosing.

You can use the postconf command to add the necessary configuration to /etc/postfix/main.cf:

```
sudo postconf -e 'relay_domains = lists.example.com'
sudo postconf -e 'transport_maps = hash:/etc/postfix/transport'
sudo postconf -e 'mailman_destination_recipient_limit = 1'
```

In /etc/postfix/master.cf double check that you have the following transport:

```
mailman   unix  -       n       n       -       -       pipe
  flags=FR user=list argv=/usr/lib/mailman/bin/postfix-to-mailman.py
  ${nexthop} ${user}
```

It calls the *postfix-to-mailman.py* script when a mail is delivered to a list.

Associate the domain lists .example.com to the Mailman transport with the transport map. Edit the file /etc/postfix/transport:

```
lists.example.com        mailman:
```

Now have Postfix build the transport map by entering the following from a terminal prompt:

```
sudo postmap -v /etc/postfix/transport
```

Then restart Postfix to enable the new configurations:

```
sudo systemctl restart postfix.service
```

Exim4

Once Exim4 is installed, you can start the Exim server using the following command from a terminal prompt:

```
sudo systemctl start exim4.service
```

In order to make mailman work with Exim4, you need to configure Exim4. As mentioned earlier, by default, Exim4 uses multiple configuration files of different types. For details, please refer to the Exim web site. To run mailman, we should add new a configuration file to the following configuration types:

- Main
- Transport
- Router

Exim creates a master configuration file by sorting all these mini configuration files. So, the order of these configuration files is very important.

Main

All the configuration files belonging to the main type are stored in the /etc/exim4/conf.d/main/ directory. You can add the following content to a new file, named 04_exim4-config_mailman:

```
# start
# Home dir for your Mailman installation -- aka Mailman's prefix
# directory.
# On Ubuntu this should be "/var/lib/mailman"
# This is normally the same as ~mailman
MM_HOME=/var/lib/mailman
#
# User and group for Mailman, should match your --with-mail-gid
# switch to Mailman's configure script.  Value is normally "mailman"
MM_UID=list
MM_GID=list
#
# Domains that your lists are in - colon separated list
# you may wish to add these into local_domains as well
```

```
domainlist mm_domains=hostname.com
#
# -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
#
# These values are derived from the ones above and should not need
# editing unless you have munged your mailman installation
#
# The path of the Mailman mail wrapper script
MM_WRAP=MM_HOME/mail/mailman
#
# The path of the list config file (used as a required file when
# verifying list addresses)
MM_LISTCHK=MM_HOME/lists/${lc::$local_part}/config.pck
# end
```

Transport

All the configuration files belonging to transport type are stored in the /etc/exim4/conf.d/transport/ directory. You can add the following content to a new file named 40_exim4-config_mailman:

```
mailman_transport:
  driver = pipe
  command = MM_WRAP \
            '${if def:local_part_suffix \
                  {${sg{$local_part_suffix}{-(\\w+)(\\+.*)?}{\$1}}} \
                  {post}}' \
            $local_part
    current_directory = MM_HOME
    home_directory = MM_HOME
    user = MM_UID
    group = MM_GID
```

Router

All the configuration files belonging to router type are stored in the /etc/exim4/conf.d/router/ directory. You can add the following content in to a new file named 101_exim4-config_mailman:

```
mailman_router:
  driver = accept
  require_files = MM_HOME/lists/$local_part/config.pck
  local_part_suffix_optional
  local_part_suffix = -bounces : -bounces+* : \
                      -confirm+* : -join : -leave : \
                      -owner : -request : -admin
  transport = mailman_transport
```

Warning

The order of main and transport configuration files can be in any order. But, the order of router configuration files must be the same. This particular file must appear before the 200_exim4-config_primary file. These two configuration files contain same type of information. The first file takes the precedence. For more details, please refer to the references section.

Mailman

Once mailman is installed, you can run it using the following command:

```
sudo systemctl start mailman.service
```

Once mailman is installed, you should create the default mailing list. Run the following command to create the mailing list:

```
sudo /usr/sbin/newlist mailman
```

```
Enter the email address of the person running the list: bhuvan at ubuntu.com
Initial mailman password:
To finish creating your mailing list, you must edit your /etc/aliases (or
equivalent) file by adding the following lines, and possibly running the
'newaliases' program:

## mailman mailing list
mailman:              "|/var/lib/mailman/mail/mailman post mailman"
mailman-admin:        "|/var/lib/mailman/mail/mailman admin mailman"
mailman-bounces:      "|/var/lib/mailman/mail/mailman bounces mailman"
mailman-confirm:      "|/var/lib/mailman/mail/mailman confirm mailman"
mailman-join:         "|/var/lib/mailman/mail/mailman join mailman"
mailman-leave:        "|/var/lib/mailman/mail/mailman leave mailman"
mailman-owner:        "|/var/lib/mailman/mail/mailman owner mailman"
mailman-request:      "|/var/lib/mailman/mail/mailman request mailman"
mailman-subscribe:    "|/var/lib/mailman/mail/mailman subscribe mailman"
mailman-unsubscribe:  "|/var/lib/mailman/mail/mailman unsubscribe mailman"

Hit enter to notify mailman owner...

#
```

We have configured either Postfix or Exim4 to recognize all emails from mailman. So, it is not mandatory to make any new entries in /etc/aliases. If you have made any changes to the configuration files, please ensure that you restart those services before continuing to next section.

> **Note**
>
> The Exim4 does not use the above aliases to forward mails to Mailman, as it uses a *discover* approach. To suppress the aliases while creating the list, you can add *MTA=None* line in Mailman configuration file, /etc/mailman/mm_cfg.py.

Administration

We assume you have a default installation. The mailman cgi scripts are still in the /usr/lib/cgi-bin/mailman/ directory. Mailman provides a web based administration facility. To access this page, point your browser to the following url:

http://hostname/cgi-bin/mailman/admin

The default mailing list, *mailman*, will appear in this screen. If you click the mailing list name, it will ask for your authentication password. If you enter the correct password, you will be able to change administrative settings of this mailing list. You can create a new mailing list using the command line utility (/usr/sbin/newlist). Alternatively, you can create a new mailing list using the web interface.

Users

Mailman provides a web based interface for users. To access this page, point your browser to the following url:

http://hostname/cgi-bin/mailman/listinfo

The default mailing list, *mailman*, will appear in this screen. If you click the mailing list name, it will display the subscription form. You can enter your email address, name (optional), and password to subscribe. An email invitation will be sent to you. You can follow the instructions in the email to subscribe.

References

GNU Mailman - Installation Manual

HOWTO - Using Exim 4 and Mailman 2.1 together

Also, see the Mailman Ubuntu Wiki page.

Mail Filtering

One of the largest issues with email today is the problem of Unsolicited Bulk Email (UBE). Also known as SPAM, such messages may also carry viruses and other forms of malware. According to some reports these messages make up the bulk of all email traffic on the Internet.

This section will cover integrating Amavisd-new, Spamassassin, and ClamAV with the Postfix Mail Transport Agent (MTA). Postfix can also check email validity by passing it through external content filters. These filters can sometimes determine if a message is spam without needing to process it with more resource intensive applications. Two common filters are opendkim and python-policyd-spf.

- Amavisd-new is a wrapper program that can call any number of content filtering programs for spam detection, antivirus, etc.
- Spamassassin uses a variety of mechanisms to filter email based on the message content.
- ClamAV is an open source antivirus application.
- opendkim implements a Sendmail Mail Filter (Milter) for the DomainKeys Identified Mail (DKIM) standard.
- python-policyd-spf enables Sender Policy Framework (SPF) checking with Postfix.

This is how the pieces fit together:

- An email message is accepted by Postfix.
- The message is passed through any external filters opendkim and python-policyd-spf in this case.
- Amavisd-new then processes the message.
- ClamAV is used to scan the message. If the message contains a virus Postfix will reject the message.
- Clean messages will then be analyzed by Spamassassin to find out if the message is spam. Spamassassin will then add X-Header lines allowing Amavisd-new to further manipulate the message.

For example, if a message has a Spam score of over fifty the message could be automatically dropped from the queue without the recipient ever having to be bothered. Another, way to handle flagged messages is to deliver them to the Mail User Agent (MUA) allowing the user to deal with the message as they see fit.

Installation

See Postfix for instructions on installing and configuring Postfix.

To install the rest of the applications enter the following from a terminal prompt:

```
sudo apt install amavisd-new spamassassin clamav-daemon
sudo apt install opendkim postfix-policyd-spf-python
```

There are some optional packages that integrate with Spamassassin for better spam detection:

```
sudo apt install pyzor razor
```

Along with the main filtering applications compression utilities are needed to process some email attachments:

```
sudo apt install arj cabextract cpio lha nomarch pax rar unrar unzip zip
```

> **Note**
>
> If some packages are not found, check that the *multiverse* repository is enabled in /etc/apt/sources. list
>
> If you make changes to the file, be sure to run sudo apt update before trying to install again.

Configuration

Now configure everything to work together and filter email.

ClamAV

The default behaviour of ClamAV will fit our needs. For more ClamAV configuration options, check the configuration files in /etc/clamav.

Add the *clamav* user to the *amavis* group in order for Amavisd-new to have the appropriate access to scan files:

```
sudo adduser clamav amavis
sudo adduser amavis clamav
```

Spamassassin

Spamassassin automatically detects optional components and will use them if they are present. This means that there is no need to configure pyzor and razor.

Edit /etc/default/spamassassin to activate the Spamassassin daemon. Change *ENABLED=0* to:

ENABLED=1

Now start the daemon:

```
sudo systemctl start spamassassin.service
```

Amavisd-new

First activate spam and antivirus detection in Amavisd-new by editing /etc/amavis/conf.d/15–content_filter_mode:

```
use strict;

# You can modify this file to re-enable SPAM checking through spamassassin
# and to re-enable antivirus checking.

#
# Default antivirus checking mode
# Uncomment the two lines below to enable it
#

@bypass_virus_checks_maps = (
   \%bypass_virus_checks, \@bypass_virus_checks_acl, \$bypass_virus_checks_re)
      ;


#
# Default SPAM checking mode
# Uncomment the two lines below to enable it
#

@bypass_spam_checks_maps = (
   \%bypass_spam_checks, \@bypass_spam_checks_acl, \$bypass_spam_checks_re);

1;  # insure a defined return
```

Bouncing spam can be a bad idea as the return address is often faked. The default behaviour is to instead discard. This is configured in /etc/amavis/conf.d/20–debian_defaults where *$final_spam_destiny* is set to D_DISCARD rather than D_BOUNCE.

Additionally, you may want to adjust the following options to flag more messages as spam:

```
$sa_tag_level_deflt = -999; # add spam info headers if at, or above that level
$sa_tag2_level_deflt = 6.0; # add 'spam detected' headers at that level
$sa_kill_level_deflt = 21.0; # triggers spam evasive actions
$sa_dsn_cutoff_level = 4; # spam level beyond which a DSN is not sent
```

If the server's *hostname* is different from the domain's MX record you may need to manually set the *$myhostname* option. Also, if the server receives mail for multiple domains the *@local_domains_acl* option will need to be customized. Edit the /etc/amavis/conf.d/50–user file:

```
$myhostname = 'mail.example.com';
@local_domains_acl = ( "example.com", "example.org" );
```

If you want to cover multiple domains you can use the following in the/etc/amavis/conf.d/50–user

```
@local_domains_acl = qw(.);
```

After configuration Amavisd-new needs to be restarted:

```
sudo systemctl restart amavis.service
```

DKIM Whitelist Amavisd-new can be configured to automatically *Whitelist* addresses from domains with valid Domain Keys. There are some pre-configured domains in the /etc/amavis/conf.d/40−policy_banks.

There are multiple ways to configure the Whitelist for a domain:

- *'example.com' => 'WHITELIST',*: will whitelist any address from the "example.com" domain.
- *'.example.com' => 'WHITELIST',*: will whitelist any address from any *subdomains* of "example.com" that have a valid signature.
- *'.example.com/@example.com' => 'WHITELIST',*: will whitelist subdomains of "example.com" that use the signature of *example.com* the parent domain.
- *'./@example.com' => 'WHITELIST',*: adds addresses that have a valid signature from "example.com". This is usually used for discussion groups that sign their messages.

A domain can also have multiple Whitelist configurations. After editing the file, restart amavisd-new:

```
sudo systemctl restart amavis.service
```

> **Note**
>
> In this context, once a domain has been added to the Whitelist the message will not receive any anti-virus or spam filtering. This may or may not be the intended behavior you wish for a domain.

Postfix

For Postfix integration, enter the following from a terminal prompt:

```
sudo postconf -e 'content_filter = smtp-amavis:[127.0.0.1]:10024'
```

Next edit /etc/postfix/master.cf and add the following to the end of the file:

```
smtp-amavis     unix    -       -       -       -       2       smtp
        -o smtp_data_done_timeout=1200
        -o smtp_send_xforward_command=yes
        -o disable_dns_lookups=yes
        -o max_use=20

127.0.0.1:10025 inet    n       -       -       -       -       smtpd
        -o content_filter=
        -o local_recipient_maps=
        -o relay_recipient_maps=
        -o smtpd_restriction_classes=
        -o smtpd_delay_reject=no
        -o smtpd_client_restrictions=permit_mynetworks,reject
        -o smtpd_helo_restrictions=
        -o smtpd_sender_restrictions=
        -o smtpd_recipient_restrictions=permit_mynetworks,reject
        -o smtpd_data_restrictions=reject_unauth_pipelining
        -o smtpd_end_of_data_restrictions=
        -o mynetworks=127.0.0.0/8
        -o smtpd_error_sleep_time=0
        -o smtpd_soft_error_limit=1001
        -o smtpd_hard_error_limit=1000
        -o smtpd_client_connection_count_limit=0
        -o smtpd_client_connection_rate_limit=0
```

```
-o receive_override_options=no_header_body_checks,
   no_unknown_recipient_checks,no_milters
```

Also add the following two lines immediately below the *"pickup"* transport service:

```
-o content_filter=
-o receive_override_options=no_header_body_checks
```

This will prevent messages that are generated to report on spam from being classified as spam.

Now restart Postfix:

```
sudo systemctl restart postfix.service
```

Content filtering with spam and virus detection is now enabled.

Amavisd-new and Spamassassin

When integrating Amavisd-new with Spamassassin, if you choose to disable the bayes filtering by editing /etc/spamassassin/local.cf and use cron to update the nightly rules, the result can cause a situation where a large amount of error messages are sent to the *amavis* user via the amavisd-new cron job.

There are several ways to handle this situation:

- Configure your MDA to filter messages you do not wish to see.
- Change /usr/sbin/amavisd-new-cronjob to check for *use_bayes 0*. For example, edit /usr/sbin/amavisd-new-cronjob and add the following to the top before the *test* statements:

  ```
  egrep -q "^[ \t]*use_bayes[ \t]*0" /etc/spamassassin/local.cf && exit 0
  ```

Testing

First, test that the Amavisd-new SMTP is listening:

```
telnet localhost 10024
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 [127.0.0.1] ESMTP amavisd-new service ready
^]
```

In the Header of messages that go through the content filter you should see:

```
X-Spam-Level:
X-Virus-Scanned: Debian amavisd-new at example.com
X-Spam-Status: No, hits=-2.3 tagged_above=-1000.0 required=5.0 tests=AWL,
   BAYES_00
X-Spam-Level:
```

> **Note**
>
> Your output will vary, but the important thing is that there are *X-Virus-Scanned* and *X-Spam-Status* entries.

Troubleshooting

The best way to figure out why something is going wrong is to check the log files.

- For instructions on Postfix logging see the Troubleshooting section.
- Amavisd-new uses Syslog to send messages to /var/log/mail.log. The amount of detail can be increased by adding the *$log_level* option to /etc/amavis/conf.d/50−user, and setting the value from 1 to 5.

  ```
  $log_level = 2;
  ```

  Note

  When the Amavisd-new log output is increased Spamassassin log output is also increased.
- The ClamAV log level can be increased by editing /etc/clamav/clamd.conf and setting the following option:

  ```
  LogVerbose true
  ```

  By default ClamAV will send log messages to /var/log/clamav/clamav.log.

  Note

  After changing an applications log settings remember to restart the service for the new settings to take affect. Also, once the issue you are troubleshooting is resolved it is a good idea to change the log settings back to normal.

References

For more information on filtering mail see the following links:

- Amavisd-new Documentation
- ClamAV Documentation and ClamAV Wiki
- Spamassassin Wiki
- Pyzor Homepage
- Razor Homepage
- DKIM.org
- Postfix Amavis New

Also, feel free to ask questions in the *#ubuntu-server* IRC channel on freenode.

Postfix

Postfix is the default Mail Transfer Agent (MTA) in Ubuntu. It attempts to be fast and secure, with flexibility in administration. It is compatible with the MTA sendmail. This section will explain installation, including how to configure SMTP for secure communications.

Note

This guide does not cover setting up Postfix *Virtual Domains*, for information on Virtual Domains and other advanced configurations see References.

Installation

To install Postfix run the following command:

```
sudo apt install postfix
```

For now, it is ok to simply accept defaults by pressing return for each question. Some of the configuration options will be investigated in greater detail in the next stage.

> **Deprecation warning:** please note that the mail−stack−delivery metapackage has been deprecated in Focal. The package still exists for compatibility reasons, but won't setup a working email system.

Basic Configuration

There are four things you should decide before starting configuration: - The <Domain> for which you'll accept email (we'll use *mail.example.com* in our example) - The network and class range of your mail server (we'll use *192.168.0.0/24*) - The username (we're using *steve*) - Type of mailbox format (*mbox* is default, we'll use the alternative, *Maildir*)

To configure postfix, run the following command:

```
sudo dpkg−reconfigure postfix
```

The user interface will be displayed. On each screen, select the following values:

- Internet Site
- *mail.example.com*
- *steve*
- *mail.example.com*, localhost.localdomain, localhost
- No
- 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128 *192.168.0.0/24*
- 0
- +
- all

To set the mailbox format, you can either edit the configuration file directly, or use the postconf command. In either case, the configuration parameters will be stored in /etc/postfix/main.cf file. Later if you wish to re-configure a particular parameter, you can either run the command or change it manually in the file.

To configure the mailbox format for **Maildir:**

```
sudo postconf −e 'home_mailbox = Maildir/'
```

> **Note**
>
> This will place new mail in /home/*username*/Maildir so you will need to configure your Mail Delivery Agent (MDA) to use the same path.

SMTP Authentication

SMTP-AUTH allows a client to identify itself through the SASL authentication mechanism, using Transport Layer Security (TLS) to encrypt the authentication process. Once authenticated the SMTP server will allow the client to relay mail.

To configure Postfix for SMTP-AUTH using SASL (Dovecot SASL), run these commands at a terminal prompt:

```
sudo postconf -e 'smtpd_sasl_type = dovecot'
sudo postconf -e 'smtpd_sasl_path = private/auth'
sudo postconf -e 'smtpd_sasl_local_domain ='
sudo postconf -e 'smtpd_sasl_security_options = noanonymous, noplaintext'
sudo postconf -e 'smtpd_sasl_tls_security_options = noanonymous'
sudo postconf -e 'broken_sasl_auth_clients = yes'
sudo postconf -e 'smtpd_sasl_auth_enable = yes'
sudo postconf -e 'smtpd_recipient_restrictions = \
permit_sasl_authenticated, permit_mynetworks, reject_unauth_destination'
```

Note

The *smtpd_sasl_path* config parameter is a path relative to the Postfix queue directory.

There are several SASL mechanism properties worth evaluating to improve the security of your deployment. The options "noanonymous,noplaintext" prevent use of mechanisms that permit anonymous authentication or that transmit credentials unencrypted.

Next, generate or obtain a digital certificate for TLS. See security - certificates in this guide for details about generating digital certificates and setting up your own Certificate Authority (CA).

Note

MUAs connecting to your mail server via TLS will need to recognize the certificate used for TLS. This can either be done using a certificate from a commercial CA or with a self-signed certificate that users manually install/accept. For MTA to MTA TLS certficates are never validated without advance agreement from the affected organizations. For MTA to MTA TLS, unless local policy requires it, there is no reason not to use a self-signed certificate. Refer to security - certificates in this guide for more details.

Once you have a certificate, configure Postfix to provide TLS encryption for both incoming and outgoing mail:

```
sudo postconf -e 'smtp_tls_security_level = may'
sudo postconf -e 'smtpd_tls_security_level = may'
sudo postconf -e 'smtp_tls_note_starttls_offer = yes'
sudo postconf -e 'smtpd_tls_key_file = /etc/ssl/private/server.key'
sudo postconf -e 'smtpd_tls_cert_file = /etc/ssl/certs/server.crt'
sudo postconf -e 'smtpd_tls_loglevel = 1'
sudo postconf -e 'smtpd_tls_received_header = yes'
sudo postconf -e 'myhostname = mail.example.com'
```

If you are using your own *Certificate Authority* to sign the certificate enter:

```
sudo postconf -e 'smtpd_tls_CAfile = /etc/ssl/certs/cacert.pem'
```

Again, for more details about certificates see security - certificates in this guide.

Note

After running all the commands, Postfix is configured for SMTP-AUTH and a self-signed certificate has been created for TLS encryption.

Now, the file /etc/postfix/main.cf should look like this:

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete
# version

smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)
biff = no
```

```
# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

myhostname = server1.example.com
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = server1.example.com, localhost.example.com, localhost
relayhost =
mynetworks = 127.0.0.0/8
mailbox_command = procmail -a "$EXTENSION"
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
smtpd_sasl_local_domain =
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
broken_sasl_auth_clients = yes
smtpd_recipient_restrictions =
permit_sasl_authenticated,permit_mynetworks,reject _unauth_destination
smtpd_tls_auth_only = no
smtp_tls_security_level = may
smtpd_tls_security_level = may
smtp_tls_note_starttls_offer = yes
smtpd_tls_key_file = /etc/ssl/private/smtpd.key
smtpd_tls_cert_file = /etc/ssl/certs/smtpd.crt
smtpd_tls_CAfile = /etc/ssl/certs/cacert.pem
smtpd_tls_loglevel = 1
smtpd_tls_received_header = yes
smtpd_tls_session_cache_timeout = 3600s
tls_random_source = dev:/dev/urandom
```

The postfix initial configuration is complete. Run the following command to restart the postfix daemon:

```
sudo systemctl restart postfix.service
```

Postfix supports SMTP-AUTH as defined in RFC2554. It is based on SASL. However it is still necessary to set up SASL authentication before you can use SMTP-AUTH.

When using ipv6, the mynetworks parameter may need to be modified to allow ipv6 addresses, for example:

```
mynetworks = 127.0.0.0/8, [::1]/128
```

Configuring SASL

Postfix supports two SASL implementations: *Cyrus SASL* and *Dovecot SASL*. To enable Dovecot SASL the dovecot-core package will need to be installed:

```
sudo apt install dovecot-core
```

Next, edit /etc/dovecot/conf.d/10-master.conf and change the following:

```
service auth {
  # auth_socket_path points to this userdb socket by default. It's typically
  # used by dovecot-lda, doveadm, possibly imap process, etc. Its default
  # permissions make it readable only by root, but you may need to relax these
  # permissions. Users that have access to this socket are able to get a list
  # of all usernames and get results of everyone's userdb lookups.
  unix_listener auth-userdb {
    #mode = 0600
    #user =
    #group =
  }

  # Postfix smtp-auth
  unix_listener /var/spool/postfix/private/auth {
    mode = 0660
    user = postfix
    group = postfix
  }
```

To permit use of SMTP-AUTH by Outlook clients, change the following line in the *authentication mechanisms* section of /etc/dovecot/conf.d/10-auth.conf from:

```
auth_mechanisms = plain
```

to this:

```
auth_mechanisms = plain login
```

Once you have Dovecot configured, restart it with:

```
sudo systemctl restart dovecot.service
```

Testing

SMTP-AUTH configuration is complete. Now it is time to test the setup.

To see if SMTP-AUTH and TLS work properly, run the following command:

```
telnet mail.example.com 25
```

After you have established the connection to the Postfix mail server, type:

```
ehlo mail.example.com
```

If you see the following in the output, then everything is working perfectly. Type quit to exit.

```
250-STARTTLS
250-AUTH LOGIN PLAIN
250-AUTH=LOGIN PLAIN
250 8BITMIME
```

Troubleshooting

When problems arise, there are a few common ways to diagnose the cause.

Escaping chroot

The Ubuntu Postfix package will by default install into a *chroot* environment for security reasons. This can add greater complexity when troubleshooting problems.

To turn off the chroot usage, locate the following line in the /etc/postfix/master.cf configuration file:

```
smtp      inet  n       -       -       -       -       smtpd
```

and modify it as follows:

```
smtp      inet  n       -       n       -       -       smtpd
```

You will then need to restart Postfix to use the new configuration. From a terminal prompt enter:

```
sudo service postfix restart
```

SMTPS

If you need secure SMTP, edit /etc/postfix/master.cf and uncomment the following line:

```
smtps     inet  n       -       -       -       -       smtpd
  -o smtpd_tls_wrappermode=yes
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_client_restrictions=permit_sasl_authenticated,reject
  -o milter_macro_daemon_name=ORIGINATING
```

Log Viewing

Postfix sends all log messages to /var/log/mail.log. However, error and warning messages can sometimes get lost in the normal log output so they are also logged to /var/log/mail.err and /var/log/mail.warn respectively.

To see messages entered into the logs in real time you can use the tail -f command:

```
tail -f /var/log/mail.err
```

Increasing Logging Detail

The amount of detail that is recorded in the logs can be increased via the configuration options. For example, to increase *TLS* activity logging set the *smtpd_tls_loglevel* option to a value from 1 to 4.

```
sudo postconf -e 'smtpd_tls_loglevel = 4'
```

Reload the service after any configuration change, to make the new config active:

```
sudo systemctl reload postfix.service
```

Logging mail delivery

If you are having trouble sending or receiving mail from a specific domain you can add the domain to the *debug_peer_list* parameter.

```
sudo postconf -e 'debug_peer_list = problem.domain'
sudo systemctl reload postfix.service
```

Increasing daemon verbosity

You can increase the verbosity of any Postfix daemon process by editing the /etc/postfix/master.cf and adding a *-v* after the entry. For example, edit the *smtp* entry:

```
smtp      unix  -       -       -       -       -        smtp -v
```

Then, reload the service as usual:

```
sudo systemctl reload postfix.service
```

Logging SASL debug info

To increase the amount of information logged when troubleshooting *SASL* issues you can set the following options in /etc/dovecot/conf.d/10-logging.conf

```
auth_debug=yes
auth_debug_passwords=yes
```

Just like Postfix if you change a Dovecot configuration the process will need to be reloaded:

```
sudo systemctl reload dovecot.service
```

> **Note**
>
> Some of the options above can drastically increase the amount of information sent to the log files. Remember to return the log level back to normal after you have corrected the problem. Then reload the appropriate daemon for the new configuration to take affect.

References

Administering a Postfix server can be a very complicated task. At some point you may need to turn to the Ubuntu community for more experienced help.

- The Postfix website documents all available configuration options.
- O'Reilly's Postfix: The Definitive Guide is rather dated but provides deep background information about configuration options.
- The Ubuntu Wiki Postfix page has more information from a Ubuntu context. There is also a Debian Wiki Postfix page that's a bit more up to date; they also have a set of Postfix Tutorials for different Debian versions.
- Info on how to set up mailman3 with postfix

Squid

Squid is a full-featured web proxy cache server application which provides proxy and cache services for Hyper Text Transport Protocol (HTTP), File Transfer Protocol (FTP), and other popular network protocols. Squid can implement caching and proxying of Secure Sockets Layer (SSL) requests and caching of Domain Name Server (DNS) lookups, and perform transparent caching. Squid also supports a wide variety of caching protocols, such as Internet Cache Protocol (ICP), the Hyper Text Caching Protocol (HTCP), the Cache Array Routing Protocol (CARP), and the Web Cache Coordination Protocol (WCCP).

The Squid proxy cache server is an excellent solution to a variety of proxy and caching server needs, and scales from the branch office to enterprise level networks while providing extensive, granular access control mechanisms, and monitoring of critical parameters via the Simple Network Management Protocol (SNMP). When selecting a computer system for use as a dedicated Squid caching proxy server for many users ensure it is configured with a large amount of physical memory as Squid maintains an in-memory cache for increased performance.

Installation

At a terminal prompt, enter the following command to install the Squid server:

```
sudo apt install squid
```

Configuration

Squid is configured by editing the directives contained within the /etc/squid/squid.conf configuration file. The following examples illustrate some of the directives which may be modified to affect the behavior of the Squid server. For more in-depth configuration of Squid, see the References section.

> **Tip**
>
> Prior to editing the configuration file, you should make a copy of the original file and protect it from writing so you will have the original settings as a reference, and to re-use as necessary. Make this copy and protect it from writing using the following commands:
>
> ```
> sudo cp /etc/squid/squid.conf /etc/squid/squid.conf.original
> sudo chmod a-w /etc/squid/squid.conf.original
> ```

- To set your Squid server to listen on TCP port 8888 instead of the default TCP port 3128, change the http_port directive as such:

  ```
  http_port 8888
  ```

- Change the visible_hostname directive in order to give the Squid server a specific hostname. This hostname does not necessarily need to be the computer's hostname. In this example it is set to *weezie*

  ```
  visible_hostname weezie
  ```

- The cache_dir option allows one to configure an on-disk cache, the default option is on-memory cache. The cache_dir directive takes the following arguments:

  ```
  cache_dir <Type> <Directory-Name> <Fs-specific-data> [options]
  ```

  In the config file you can find the default cache_dir directive commented out:

  ```
  # Uncomment and adjust the following to add a disk cache directory.
  #cache_dir ufs /var/spool/squid 100 16 256
  ```

  You can just use the default option but you can also customize your cache directory, basically changing the <Type> of this directory, it can be:

  - ufs: the old well-known Squid storage format that has always been there.
  - aufs: uses the same storage format as ufs, utilizing POSIX-threads to avoid blocking the main Squid process on disk-I/O. This was formerly known in Squid as async-io.
  - diskd: uses the same storage format as ufs, utilizing a separate process to avoid blocking the main Squid process on disk-I/O.
  - rock: is a database-style storage. All cached entries are stored in a "database" file, using fixed-size slots. A single entry occupies one or more slots.

  If you want to use a different directory type please take a look at their different options.

- Using Squid's access control, you may configure use of Internet services proxied by Squid to be available only users with certain Internet Protocol (IP) addresses. For example, we will illustrate access by users of the 192.168.42.0/24 subnetwork only:

  Add the following to the **bottom** of the ACL section of your /etc/squid/squid.conf file:

```
acl fortytwo_network src 192.168.42.0/24
```

Then, add the following to the **top** of the http_access section of your /etc/squid/squid.conf file:

```
http_access allow fortytwo_network
```

- Using the excellent access control features of Squid, you may configure use of Internet services proxied by Squid to be available only during normal business hours. For example, we'll illustrate access by employees of a business which is operating between 9:00AM and 5:00PM, Monday through Friday, and which uses the 10.1.42.0/24 subnetwork:

  Add the following to the **bottom** of the ACL section of your /etc/squid/squid.conf file:

  ```
  acl biz_network src 10.1.42.0/24
  acl biz_hours time M T W T F 9:00-17:00
  ```

  Then, add the following to the **top** of the http_access section of your /etc/squid/squid.conf file:

  ```
  http_access allow biz_network biz_hours
  ```

  Note

  After making changes to the /etc/squid/squid.conf file, save the file and restart the squid server application to effect the changes using the following command entered at a terminal prompt:

  ```
  sudo systemctl restart squid.service
  ```

  Note

  If formerly a customized squid3 was used that set up the spool at /var/log/squid3 to be a mountpoint, but otherwise kept the default configuration the upgrade will fail. The upgrade tries to rename/move files as needed, but it can't do so for an active mountpoint. In that case please either adapt the mountpoint or the config in /etc/squid/squid.conf so that they match.

  The same applies if the **include** config statement was used to pull in more files from the old path at /etc/squid3/. In those cases you should move and adapt your configuration accordingly.

References

Squid Website

Ubuntu Wiki Squid page.

HTTPD - Apache2 Web Server

Apache is the most commonly used Web server on Linux systems. Web servers are used to serve Web pages requested by client computers. Clients typically request and view Web pages using Web browser applications such as Firefox, Opera, Chromium, or Internet Explorer.

Users enter a Uniform Resource Locator (URL) to point to a Web server by means of its Fully Qualified Domain Name (FQDN) and a path to the required resource. For example, to view the home page of the Ubuntu Web site a user will enter only the FQDN:

```
www.ubuntu.com
```

To view the community sub-page, a user will enter the FQDN followed by a path:

```
www.ubuntu.com/community
```

The most common protocol used to transfer Web pages is the Hyper Text Transfer Protocol (HTTP). Protocols such as Hyper Text Transfer Protocol over Secure Sockets Layer (HTTPS), and File Transfer Protocol (FTP), a protocol for uploading and downloading files, are also supported.

Apache Web Servers are often used in combination with the MySQL database engine, the HyperText Preprocessor (PHP) scripting language, and other popular scripting languages such as Python and Perl. This configuration is termed LAMP (Linux, Apache, MySQL and Perl/Python/PHP) and forms a powerful and robust platform for the development and deployment of Web-based applications.

Installation

The Apache2 web server is available in Ubuntu Linux. To install Apache2:

At a terminal prompt enter the following command:

```
sudo apt install apache2
```

Configuration

Apache2 is configured by placing *directives* in plain text configuration files. These *directives* are separated between the following files and directories:

- *apache2.conf:* the main Apache2 configuration file. Contains settings that are *global* to Apache2.
- *httpd.conf:* historically the main Apache2 configuration file, named after the httpd daemon. In other distributions (or older versions of Ubuntu), the file might be present. In Ubuntu, all configuration options have been moved to *apache2.conf* and the below referenced directories, and this file no longer exists.
- *conf-available:* this directory contains available configuration files. All files that were previously in /etc/apache2/conf.d should be moved to /etc/apache2/conf−available.
- *conf-enabled:* holds *symlinks* to the files in /etc/apache2/conf−available. When a configuration file is symlinked, it will be enabled the next time apache2 is restarted.
- *envvars:* file where Apache2 *environment* variables are set.
- *mods-available:* this directory contains configuration files to both load *modules* and configure them. Not all modules will have specific configuration files, however.
- *mods-enabled:* holds *symlinks* to the files in /etc/apache2/mods−available. When a module configuration file is symlinked it will be enabled the next time apache2 is restarted.
- *ports.conf:* houses the directives that determine which TCP ports Apache2 is listening on.
- *sites-available:* this directory has configuration files for Apache2 *Virtual Hosts*. Virtual Hosts allow Apache2 to be configured for multiple sites that have separate configurations.
- *sites-enabled:* like mods-enabled, sites −enabled contains symlinks to the /etc/apache2/sites−available directory. Similarly when a configuration file in sites-available is symlinked, the site configured by it will be active once Apache2 is restarted.
- *magic:* instructions for determining MIME type based on the first few bytes of a file.

In addition, other configuration files may be added using the *Include* directive, and wildcards can be used to include many configuration files. Any directive may be placed in any of these configuration files. Changes to the main configuration files are only recognized by Apache2 when it is started or restarted.

The server also reads a file containing mime document types; the filename is set by the *TypesConfig* directive, typically via /etc/apache2/mods−available/mime.conf, which might also include additions and overrides, and is /etc/mime.types by default.

Basic Settings

This section explains Apache2 server essential configuration parameters. Refer to the Apache2 Documentation for more details.

- Apache2 ships with a virtual-host-friendly default configuration. That is, it is configured with a single default virtual host (using the *VirtualHost* directive) which can be modified or used as-is if you have a single site, or used as a template for additional virtual hosts if you have multiple sites. If left alone, the default virtual host will serve as your default site, or the site users will see if the URL they enter does not match the *ServerName* directive of any of your custom sites. To modify the default virtual host, edit the file /etc/apache2/sites−available/000−default.conf.

  Note

  The directives set for a virtual host only apply to that particular virtual host. If a directive is set server-wide and not defined within the virtual host settings, the default setting is used. For example, you can define a Webmaster email address and not define individual email addresses for each virtual host.

  If you wish to configure a new virtual host or site, copy that file into the same directory with a name you choose. For example:

  ```
  sudo cp /etc/apache2/sites−available/000−default.conf /etc/apache2/sites−
      available/mynewsite.conf
  ```

  Edit the new file to configure the new site using some of the directives described below.

- The *ServerAdmin* directive specifies the email address to be advertised for the server's administrator. The default value is webmaster@localhost. This should be changed to an email address that is delivered to you (if you are the server's administrator). If your website has a problem, Apache2 will display an error message containing this email address to report the problem to. Find this directive in your site's configuration file in /etc/apache2/sites-available.

- The *Listen* directive specifies the port, and optionally the IP address, Apache2 should listen on. If the IP address is not specified, Apache2 will listen on all IP addresses assigned to the machine it runs on. The default value for the Listen directive is 80. Change this to 127.0.0.1:80 to cause Apache2 to listen only on your loopback interface so that it will not be available to the Internet, to (for example) 81 to change the port that it listens on, or leave it as is for normal operation. This directive can be found and changed in its own file, /etc/apache2/ports.conf

- The *ServerName* directive is optional and specifies what FQDN your site should answer to. The default virtual host has no ServerName directive specified, so it will respond to all requests that do not match a ServerName directive in another virtual host. If you have just acquired the domain name mynewsite .com and wish to host it on your Ubuntu server, the value of the ServerName directive in your virtual host configuration file should be mynewsite.com. Add this directive to the new virtual host file you created earlier (/etc/apache2/sites−available/mynewsite.conf).

  You may also want your site to respond to www.mynewsite.com, since many users will assume the www prefix is appropriate. Use the *ServerAlias* directive for this. You may also use wildcards in the ServerAlias directive.

  For example, the following configuration will cause your site to respond to any domain request ending in *.mynewsite.com*.

  ```
  ServerAlias *.mynewsite.com
  ```

- The *DocumentRoot* directive specifies where Apache2 should look for the files that make up the site. The default value is /var/www/html, as specified in /etc/apache2/sites−available/000−default.conf. If desired, change this value in your site's virtual host file, and remember to create that directory if necessary!

Enable the new *VirtualHost* using the a2ensite utility and restart Apache2:

```
sudo a2ensite mynewsite
sudo systemctl restart apache2.service
```

> **Note**
>
> Be sure to replace *mynewsite* with a more descriptive name for the VirtualHost. One method is to name the file after the *ServerName* directive of the VirtualHost.

Similarly, use the a2dissite utility to disable sites. This is can be useful when troubleshooting configuration problems with multiple VirtualHosts:

```
sudo a2dissite mynewsite
sudo systemctl restart apache2.service
```

Default Settings

This section explains configuration of the Apache2 server default settings. For example, if you add a virtual host, the settings you configure for the virtual host take precedence for that virtual host. For a directive not defined within the virtual host settings, the default value is used.

- The *DirectoryIndex* is the default page served by the server when a user requests an index of a directory by specifying a forward slash (/) at the end of the directory name.

  For example, when a user requests the page http://www.example.com/this_directory/, he or she will get either the DirectoryIndex page if it exists, a server-generated directory list if it does not and the Indexes option is specified, or a Permission Denied page if neither is true. The server will try to find one of the files listed in the DirectoryIndex directive and will return the first one it finds. If it does not find any of these files and if *Options Indexes* is set for that directory, the server will generate and return a list, in HTML format, of the subdirectories and files in the directory. The default value, found in /etc/apache2/mods−available/dir.conf is "index.html index.cgi index.pl index.php index.xhtml index.htm". Thus, if Apache2 finds a file in a requested directory matching any of these names, the first will be displayed.

- The *ErrorDocument* directive allows you to specify a file for Apache2 to use for specific error events. For example, if a user requests a resource that does not exist, a 404 error will occur. By default, Apache2 will simply return a HTTP 404 Return code. Read /etc/apache2/conf−available/localized−error−pages.conf for detailed instructions for using ErrorDocument, including locations of example files.

- By default, the server writes the transfer log to the file /var/log/apache2/access.log. You can change this on a per-site basis in your virtual host configuration files with the *CustomLog* directive, or omit it to accept the default, specified in /etc/apache2/conf−available/other−vhosts−access−log.conf. You may also specify the file to which errors are logged, via the *ErrorLog* directive, whose default is /var/log/apache2/error.log. These are kept separate from the transfer logs to aid in troubleshooting problems with your Apache2 server. You may also specify the *LogLevel* (the default value is "warn") and the *LogFormat* (see /etc/apache2/apache2.conf for the default value).

- Some options are specified on a per-directory basis rather than per-server. *Options* is one of these directives. A Directory stanza is enclosed in XML-like tags, like so:

```
<Directory /var/www/html/mynewsite>
...
</Directory>
```

The *Options* directive within a Directory stanza accepts one or more of the following values (among others), separated by spaces:

- **ExecCGI** - Allow execution of CGI scripts. CGI scripts are not executed if this option is not chosen.

  Caution

  Most files should not be executed as CGI scripts. This would be very dangerous. CGI scripts should kept in a directory separate from and outside your DocumentRoot, and only this directory should have the ExecCGI option set. This is the default, and the default location for CGI scripts is /usr/lib/cgi−bin.

- **Includes** - Allow server-side includes. Server-side includes allow an HTML file to *include* other files. See Apache SSI documentation (Ubuntu community) for more information.

- **IncludesNOEXEC** - Allow server-side includes, but disable the *#exec* and *#include* commands in CGI scripts.

- **Indexes** - Display a formatted list of the directory's contents, if no *DirectoryIndex* (such as index.html) exists in the requested directory.

  Caution

  For security reasons, this should usually not be set, and certainly should not be set on your DocumentRoot directory. Enable this option carefully on a per-directory basis only if you are certain you want users to see the entire contents of the directory.

- **Multiview** - Support content-negotiated multiviews; this option is disabled by default for security reasons. See the Apache2 documentation on this option.

- **SymLinksIfOwnerMatch** - Only follow symbolic links if the target file or directory has the same owner as the link.

apache2 Settings

This section explains some basic apache2 daemon configuration settings.

LockFile - The LockFile directive sets the path to the lockfile used when the server is compiled with either USE_FCNTL_SERIALIZED_ACCEPT or USE_FLOCK_SERIALIZED_ACCEPT. It must be stored on the local disk. It should be left to the default value unless the logs directory is located on an NFS share. If this is the case, the default value should be changed to a location on the local disk and to a directory that is readable only by root.

PidFile - The PidFile directive sets the file in which the server records its process ID (pid). This file should only be readable by root. In most cases, it should be left to the default value.

User - The User directive sets the userid used by the server to answer requests. This setting determines the server's access. Any files inaccessible to this user will also be inaccessible to your website's visitors. The default value for User is "www-data".

Warning

Unless you know exactly what you are doing, do not set the User directive to root. Using root as the User will create large security holes for your Web server.

Group - The Group directive is similar to the User directive. Group sets the group under which the server will answer requests. The default group is also "www-data".

Apache2 Modules

Apache2 is a modular server. This implies that only the most basic functionality is included in the core server. Extended features are available through modules which can be loaded into Apache2. By default, a base set of modules is included in the server at compile-time. If the server is compiled to use dynamically loaded modules, then modules can be compiled separately, and added at any time using the LoadModule directive. Otherwise, Apache2 must be recompiled to add or remove modules.

Ubuntu compiles Apache2 to allow the dynamic loading of modules. Configuration directives may be conditionally included on the presence of a particular module by enclosing them in an *<IfModule>* block.

You can install additional Apache2 modules and use them with your Web server. For example, run the following command at a terminal prompt to install the Python 3 WSGI module:

```
sudo apt install libapache2-mod-wsgi-py3
```

The installation will enable the module automatically, but we can disable it with a2dismod:

```
sudo a2dismod wsgi
sudo systemctl restart apache2.service
```

And then use the a2enmod utility to re-enable it:

```
sudo a2enmod wsgi
sudo systemctl restart apache2.service
```

See the /etc/apache2/mods-available directory for additional modules already available on your system.

HTTPS Configuration

The mod_ssl module adds an important feature to the Apache2 server - the ability to encrypt communications. Thus, when your browser is communicating using SSL, the https:// prefix is used at the beginning of the Uniform Resource Locator (URL) in the browser navigation bar.

The mod_ssl module is available in apache2-common package. Execute the following command at a terminal prompt to enable the mod_ssl module:

```
sudo a2enmod ssl
```

There is a default HTTPS configuration file in /etc/apache2/sites-available/default-ssl.conf. In order for Apache2 to provide HTTPS, a *certificate* and *key* file are also needed. The default HTTPS configuration will use a certificate and key generated by the ssl-cert package. They are good for testing, but the auto-generated certificate and key should be replaced by a certificate specific to the site or server. For information on generating a key and obtaining a certificate see Certificates.

To configure Apache2 for HTTPS, enter the following:

```
sudo a2ensite default-ssl
```

> **Note**
>
> The directories /etc/ssl/certs and /etc/ssl/private are the default locations. If you install the certificate and key in another directory make sure to change *SSLCertificateFile* and *SSLCertificateKeyFile* appropriately.

With Apache2 now configured for HTTPS, restart the service to enable the new settings:

```
sudo systemctl restart apache2.service
```

> **Note**
>
> Depending on how you obtained your certificate you may need to enter a passphrase when Apache2 starts.

You can access the secure server pages by typing https://your_hostname/url/ in your browser address bar.

Sharing Write Permission

For more than one user to be able to write to the same directory it will be necessary to grant write permission to a group they share in common. The following example grants shared write permission to /var/www/html to the group "webmasters".

```
sudo chgrp -R webmasters /var/www/html
sudo chmod -R g=rwX /var/www/html/
```

These commands recursively set the group permission on all files and directories in /var/www/html to allow reading, writing and searching of directories. Many admins find this useful for allowing multiple users to edit files in a directory tree.

> **Warning**
>
> The apache2 daemon will run as the www-data user, which has a corresponding www-data group. These *should not* be granted write access to the document root, as this would mean that vulnerabilities in Apache or the applications it is serving would allow attackers to overwrite the served content.

References

- Apache2 Documentation contains in depth information on Apache2 configuration directives. Also, see the apache2-doc package for the official Apache2 docs.
- O'Reilly's Apache Cookbook is a good resource for accomplishing specific Apache2 configurations.
- For Ubuntu specific Apache2 questions, ask in the *#ubuntu-server* IRC channel on freenode.net.

Apache Tomcat

Apache Tomcat is a web container that allows you to serve Java Servlets and JSP (Java Server Pages) web applications.

Ubuntu has supported packages for both Tomcat 6 and 7. Tomcat 6 is the legacy version, and Tomcat 7 is the current version where new features are implemented. Both are considered stable. This guide will focus on Tomcat 7, but most configuration details are valid for both versions.

The Tomcat packages in Ubuntu support two different ways of running Tomcat. You can install them as a classic unique system-wide instance, that will be started at boot time will run as the tomcat7 (or tomcat6) unprivileged user. But you can also deploy private instances that will run with your own user rights, and that you should start and stop by yourself. This second way is particularly useful in a development server context where multiple users need to test on their own private Tomcat instances.

System-wide installation

To install the Tomcat server, you can enter the following command in the terminal prompt:

```
sudo apt install tomcat7
```

This will install a Tomcat server with just a default ROOT webapp that displays a minimal "It works" page by default.

Configuration

Tomcat configuration files can be found in /etc/tomcat7. Only a few common configuration tweaks will be described here, please see Tomcat 7.0 documentation for more.

Changing default ports

By default Tomcat runs a HTTP connector on port 8080 and an AJP connector on port 8009. You might want to change those default ports to avoid conflict with another application on the system. This is done by changing the following lines in /etc/tomcat7/server.xml:

```
<Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
...
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Changing JVM used

By default Tomcat will run preferably with OpenJDK JVMs, then try the Sun JVMs, then try some other JVMs. You can force Tomcat to use a specific JVM by setting JAVA_HOME in /etc/default/tomcat7:

```
JAVA_HOME=/usr/lib/jvm/java-6-sun
```

Declaring users and roles

Usernames, passwords and roles (groups) can be defined centrally in a Servlet container. This is done in the /etc/tomcat7/tomcat-users.xml file:

```
<role rolename="admin"/>
<user username="tomcat" password="s3cret" roles="admin"/>
```

Using Tomcat standard webapps

Tomcat is shipped with webapps that you can install for documentation, administration or demo purposes.

Tomcat documentation

The tomcat7-docs package contains Tomcat documentation, packaged as a webapp that you can access by default at http://yourserver:8080/docs. You can install it by entering the following command in the terminal prompt:

```
sudo apt install tomcat7-docs
```

Tomcat administration webapps

The tomcat7-admin package contains two webapps that can be used to administer the Tomcat server using a web interface. You can install them by entering the following command in the terminal prompt:

```
sudo apt install tomcat7-admin
```

The first one is the *manager* webapp, which you can access by default at http://yourserver:8080/manager/html. It is primarily used to get server status and restart webapps.

> **Note**
>
> Access to the *manager* application is protected by default: you need to define a user with the role "manager-gui" in /etc/tomcat7/tomcat-users.xml before you can access it.

The second one is the *host-manager* webapp, which you can access by default at http://yourserver:8080/host-manager/html. It can be used to create virtual hosts dynamically.

> **Note**
>
> Access to the *host-manager* application is also protected by default: you need to define a user with the role "admin-gui" in /etc/tomcat7/tomcat-users.xml before you can access it.

For security reasons, the tomcat7 user cannot write to the /etc/tomcat7 directory by default. Some features in these admin webapps (application deployment, virtual host creation) need write access to that directory. If you want to use these features execute the following, to give users in the tomcat7 group the necessary rights:

```
sudo chgrp -R tomcat7 /etc/tomcat7
sudo chmod -R g+w /etc/tomcat7
```

Tomcat examples webapps

The tomcat7-examples package contains two webapps that can be used to test or demonstrate Servlets and JSP features, which you can access them by default at http://yourserver:8080/examples. You can install them by entering the following command in the terminal prompt:

```
sudo apt install tomcat7-examples
```

Using private instances

Tomcat is heavily used in development and testing scenarios where using a single system-wide instance doesn't meet the requirements of multiple users on a single system. The Tomcat packages in Ubuntu come with tools to help deploy your own user-oriented instances, allowing every user on a system to run (without root rights) separate private instances while still using the system-installed libraries.

> **Note**
>
> It is possible to run the system-wide instance and the private instances in parallel, as long as they do not use the same TCP ports.

Installing private instance support

You can install everything necessary to run private instances by entering the following command in the terminal prompt:

```
sudo apt install tomcat7-user
```

Creating a private instance

You can create a private instance directory by entering the following command in the terminal prompt:

```
tomcat7-instance-create my-instance
```

This will create a new my-instance directory with all the necessary subdirectories and scripts. You can for example install your common libraries in the lib/ subdirectory and deploy your webapps in the webapps/ subdirectory. No webapps are deployed by default.

Configuring your private instance

You will find the classic Tomcat configuration files for your private instance in the conf/ subdirectory. You should for example certainly edit the conf/server.xml file to change the default ports used by your private Tomcat instance to avoid conflict with other instances that might be running.

Starting/stopping your private instance

You can start your private instance by entering the following command in the terminal prompt (supposing your instance is located in the my-instance directory):

```
my-instance/bin/startup.sh
```

> **Note**
>
> You should check the logs/ subdirectory for any error. If you have a *java.net.BindException: Address already in use<null>:8080* error, it means that the port you're using is already taken and that you should change it.

You can stop your instance by entering the following command in the terminal prompt (supposing your instance is located in the my-instance directory):

```
my-instance/bin/shutdown.sh
```

References

- See the Apache Tomcat website for more information.
- Tomcat: The Definitive Guide is a good resource for building web applications with Tomcat.
- For additional books see the Tomcat Books list page.

Web Servers

The primary function of a web server is to store, process and deliver Web pages to clients. The clients communicate with the server sending HTTP requests. Clients, mostly via Web Browsers, request for a specific resources and the server responds with the content of that resource or an error message. The response is usually a Web page such as HTML documents which may include images, style sheets, scripts, and the content in form of text.

When accessing a Web Server, every HTTP request that is received is responded to with a content and a HTTP status code. HTTP status codes are three-digit codes, and are grouped into five different classes. The class of a status code can be quickly identified by its first digit:

- **1xx** : *Informational* - Request received, continuing process
- **2xx** : *Success* - The action was successfully received, understood, and accepted
- **3xx** : *Redirection* - Further action must be taken in order to complete the request
- **4xx** : *Client Error* - The request contains bad syntax or cannot be fulfilled
- **5xx** : *Server Error* - The server failed to fulfill an apparently valid request

More information about status code check the RFC 2616.

Implementation

Web Servers are heavily used in the deployment of Web sites and in this scenario we can use two different implementations:

- **Static Web Server**: The content of the server's response will be the hosted files "as-is".
- **Dynamic Web Server**: Consist in a Web Server plus an extra software, usually an *application server* and a *database*. For example, to produce the Web pages you see in the Web browser, the application server might fill an HTML template with contents from a database. Due to that we say that the content of the server's response is generated dynamically.

Introduction to High Availability

A definition of High Availability Clusters from Wikipedia:

High Availability Clusters

> **High-availability clusters** (also known as **HA clusters** , **fail-over clusters** or **Metroclusters Active/Active**) are groups of computers that support server applications that can be reliably utilized with a minimum amount of down-time.
>
> They operate by using high availability software to harness redundant computers in groups or clusters that provide continued service when system components fail.
>
> Without clustering, if a server running a particular application crashes, the application will be unavailable until the crashed server is fixed. HA clustering remedies this situation by detecting hardware/software faults, and immediately restarting the application on another system without requiring administrative intervention, a process known as failover.
>
> As part of this process, clustering software may configure the node before starting the application on it. For example, appropriate file systems may need to be imported and mounted, network hardware may have to be configured, and some supporting applications may need to be running as well.
>
> HA clusters are often used for critical databases, file sharing on a network, business applications, and customer services such as electronic commerce websites.

High Availability Cluster Heartbeat

> HA cluster implementations attempt to build redundancy into a cluster to eliminate single points of failure, including multiple network connections and data storage which is redundantly connected via storage area networks.
>
> HA clusters usually use a heartbeat private network connection which is used to monitor the health and status of each node in the cluster. One subtle but serious condition all clustering

software must be able to handle is split-brain, which occurs when all of the private links go down simultaneously, but the cluster nodes are still running.

If that happens, each node in the cluster may mistakenly decide that every other node has gone down and attempt to start services that other nodes are still running. Having duplicate instances of services may cause data corruption on the shared storage.

High Availability Cluster Quorum

HA clusters often also use quorum witness storage (local or cloud) to avoid this scenario. A witness device cannot be shared between two halves of a split cluster, so in the event that all cluster members cannot communicate with each other (e.g., failed heartbeat), if a member cannot access the witness, it cannot become active.

Example

2nodeHAcluster|578x674,75%

Linux High Availability Projects

There are many upstream high availability related projects that are included in Ubuntu Linux. This section will describe the most important ones.

For the latest LTS version, the following packages are considered

Ubuntu HA Core Packages

Packages in this list are supported just like any other package available in **main repository** would be.

Package	URL
libqb	Ubuntu | Upstream
kronosnet	Ubuntu | Upstream
corosync	Ubuntu | Upstream
pacemaker	Ubuntu | Upstream
resource-agents	Ubuntu | Upstream
fence-agents	Ubuntu | Upstream
crmsh	Ubuntu | Upstream
cluster-glue	Ubuntu | Upstream
drbd-utils	Ubuntu | Upstream
dlm	Ubuntu | Upstream
gfs2-utils	Ubuntu | Upstream
keepalived	Ubuntu | Upstream

Ubuntu HA Community Packages

Packages in this list are supported just like any other package available in [**universe**] **repository** would be.

Package	URL

Package	URL
pcs*	Ubuntu | Upstream
csync2	Ubuntu | Upstream
corosync-qdevice	Ubuntu | Upstream
fence-virt	Ubuntu | Upstream
sbd	Ubuntu | Upstream
booth	Ubuntu | Upstream

Note: pcs will likely replace **crmsh** in main repository in future Ubuntu versions.

Ubuntu HA Deprecated Packages

Packages in this list are **only supported by the upstream community** . All bugs opened against these agents will be forwarded to upstream IF makes sense (affected version is close to upstream).

Package	URL
ocfs2-tools	Ubuntu | Upstream

Ubuntu HA Related Packages

Packages in this list aren't necessarily **HA** related packages, but they have a very important role in High Availability Clusters and are supported like any other package provide by the **main** repository.

Package	URL
multipath-tools	Ubuntu | Upstream
open-iscsi	Ubuntu | Upstream
sg3-utils	Ubuntu | Upstream
tgt OR targetcli-fb*	Ubuntu | Upstream
lvm2	Ubuntu | Upstream

Note: **targetcli-fb (Linux LIO)** will likely replace **tgt** in future Ubuntu versions.

Upstream Documentation

The server guide does not have the intent to document every existing option for all the HA related softwares described in this page, but to document recommended scenarios for Ubuntu HA Clusters. You will find more complete documentation upstream at:

- ClusterLabs
  - Clusters From Scratch
  - Managing Pacemaker Clusters
  - Pacemaker Configuration Explained
  - Pacemaker Remote - Scaling HA Clusters
- Other
  - Ubuntu Bionic HA in Shared Disk Environments (Azure)

A very special thanks, and all the credits, to ClusterLabs Project for all that detailed documentation.

Ubuntu HA - Pacemaker Resource Agents Supportability

After discussions among Ubuntu Developers, it was decided that Ubuntu project should focus in splitting all existing Pacemaker Resource Agents into different categories:

- Resource Agents: main
- Resource Agents: [universe]
- Resource Agents: [universe]-community
- Resource Agents: [non-supported]
- Resource Agents: [deprecated]

Note: There is a current plan to split resource agents into different packages so supported agents can be installed independently.

Resource Agents: main

Agents in this list are supported just like any other package available in **main repository** would be.

|

RESOURCE AGENT

|

RESOURCE AGENT DESCRIPTION

| |

SUPPORT AGENTS

|Delay|test resource for introducing delay| |MailTo|sends email to a sysadmin whenever a takeover occurs| |ClusterMon|runs crm_mon to a html page from time to time| |HealthCPU|measures CPU idling and updates #health-cpu attr| |HealthIOWait|measures CPU idling and updates #health-iowait attr| |HealthSMART|measures CPU idling and updates #health-smart attr| ||

SERVICES (OCF, Systemd, SysV)

| |apache|apache web server instance| |dovecot|dovecot IMAP/POP3 server instance| |dhcpd|chrooted ISC dhcp server instance| |mysql|MySQL instance| |mysql-proxy|MySQL proxy instance| |pgsql|pgsql database instance| |named|bind/named server instance| |nfsnotify|nfs sm-notify reboot notifications daemon| |nfsserver|nfs server resource| |exportfs|nfs exports (not the nfs server)| |nginx|Nginx web/proxy server instance| |postfix|postfix mail server instance| |rabbitmq-cluster|cloned rabbitmq cluster instance| |remote|pacemaker remote resource agent| |rsyncd|rsyncd instance| |rsyslog|rsyslogd instance| |slapd|stand-alone LDAP daemon instance| |Squid|squid proxy server instance| |vsftpd|vsftpd server instance| | |

STORAGE

|Raid1|software RAID (MD) devices on shared storage| |iscsi|local iscsi initiator and its conns to targets| |iSCSILogicalUnit|iSCSI logical units| |iSCSITarget|iSCSI target export agent (implementation: tgt / lio)| |LVM|LVM volume as an HA resource| |LVM-activate|LVM activation/deact work for VGs (lvmlockd+LVM-activate OR clvm+LVM-activate)| |Filesystem|filesystem on a shared storage medium| |symlink|symbolic link| |ZFS|ZFS pools import/export| ||

LOCKING & RESERVATIONS

|controld|distributed lock manager for clustered FSs| |clvm|clvmd daemon (cluster logical vol manager)| |lvmlockd|agent manages the lvmlockd daemon.| |mpathpersist|SCSI persistent reservations on mpath devs| |sg_persist|master/slave resource for SCSI3 reservations| ||

NETWORKING

|Route|network routes| |iface-bridge|bridge network interfaces| |iface-vlan|vlan network interfaces| |IPaddr2|virtual IPv4 and IPv6 addresses| |ipsec|ipsec tunnels for VIPs| |IPsrcaddr|preferred source address modification| |IPv6addr|IPv6 aliases| |conntrackd|conntrackd instance| |SendArp|send gratuitous ARP for IP address| |VIPArip|virtual IP address through RIP2| |ifspeed|monitor action runs -> updates CIB with if speed| ||

VIRTUALIZATION

|VirtualDomain|manages virtual domains through libvirt (virtual machine only)| ||

CONTAINERS

|lxc|allows LXC containers to be managed by the cluster|

Resource Agents: [universe]

Agents in this list are supported just like any other package available in **[universe] repository** would be.

RESOURCE AGENT	RESOURCE AGENT DESCRIPTION

||

SUPPORT AGENTS

|anything|generic agent to manage virtually *anything*| |Dummy|testing dummy resource agent (template for RA writers)| |AudibleAlarm|audible beeps at interval| |Stateful|example agent that supports two states| |WinPopup|sends a SMB notification msg (popup) to a host| ||

SERVICES

|asterisk|asterisk PBX| |CTDB|clustered samba (for needed clustered underlying)| |dnsupdate|ip take-over via dynamic dns updates| |fio|fio instance| |galera|galera instance| |garbd|galera arbitrator instance| |jboss|JBoss application server instance| |jira|JIRA server instance| |kamailio|kamailio SIP proxy/registrar instance| |mariadb|MariaDB master/slave instance| |nagios|nagios instance| |ovsmonitor|clone resource to monitor network bonds on diff nodes| |pgagent|pgagent instance| |pound|pound reverse proxy load-bal server instance| |proftpd|proftpd instance| |Pure-FTPd|pure-ftpd instance| |redis|redis server (supports master/slave replicas)| |syslog-ng|syslog-ng instance| |tomcat|tomcat servlet environment instance| |varnish|varnish instance| ||

STORAGE

|AoEtarget|ata over ethernet| ||

NETWORKING

|IPaddr|virtual IPv4 addresses| |ocf:pacemaker:ping|records in CIB number of nodes host can connect to| |portblock|temporarily block/unblock access to tcp/udp ports| ||

OPENSTACK

|openstack-cinder-volume|attach cinder vol to an instance (os-info <->)| |openstack-floating-ip|move a floating IP from an instance to another| ||

VIRTUALIZATION

|Xen|xen unprivileged domains| ||

REGISTRATION (CIB)

|lxd-info|nr of lxd containers running in CIB| |machine-info|records various node attributes in CIB| |NodeUtilization|cpu / host mem / hypervisor mem etc… into CIB| |openstack-info|records attributes of a node into CIB| |SysInfo|records various node attributes into CIB| |SystemHealth|monitors health of system using IPMI| |attribute|sets node attr one way when started and vice-versa|

Resource Agents: [universe]-community

Agents in this list are **only supported by the upstream community**. All bugs opened against these agents will be forwarded to upstream IF makes sense (affected version is close to upstream).

RESOURCE AGENT	RESOURCE AGENT DESCRIPTION

||

SERVICES

|SphinxSearchDaemon|sphix search daemon| |Xinetd|start/stop services managed by xinetd| |zabbixserver|zabbix server instance| ||

STORAGE

|o2cb|oracle cluster filesystem userspace daemon (oracle)| |sfex|excl access to shared storage using SF-EX| ||

VIRTUALIZATION

|aliyun-vpc-move-ip|move ip within a vpc of the aliyum ecs (alibaba)| |awseip|manages aws elastic IP address (aws)| |awsvip|manages aws secondary private ip addresses (aws)| |aws-vpc-move-ip|move ip within a vpc of the aws ec2 (aws)| |aws-vpc-route53|update route53 vpc record for aws ec2 (aws)| |azure-events|monitor for scheduled events for azure vm (azure)| |azure-lb|answers azure load balancer health probe req (azure)| |gcp-vpc-move-ip|floating ip address within a GCP VPC (google)| |ManageVE|openVZ virtual environment (virtuozzo)| |minio|minio server instance| |podman|creates/launches podman containers| |rkt|creates/launches container based on supplied image| ||

CONTAINERS

|docker|docker container resource agent|

Resource Agents: [non-supported]

Agents in this list are NOT supported in Ubuntu and **might be removed in future Ubuntu HA versions.**

RESOURCE AGENT	RESOURCE AGENT DESCRIPTION

||

UNSUPPORTED

|db2|manages IBM DB2 LUW databases (IBM)| |eDir88|Novell eDirectory directory server instance (novell)| |ICP|ICP vortex clustered host drive (intel)| |ids|IBM informix dynamic server (IDS) (IBM)| |SAPDatabase|SAP database (of any type) instance agent (SAP)| |SAPInstance|SAP application server instances agent (SAP)| |ServeRAID|enables/disables shared serveRAID merge groups (IBM)| |ManageRAID|raid devices (/etc/conf.d/HB-ManageRAID)| |oraasm|oracle asm agent / uses ohasd for asm disk grp (oracle)| |oracle|oracle database instance (oracle)| |oralsnr|oracle TNS listener (oracle)| |sybaseASE|sybase ASE failover instance (Sybase)| |vdo-vol|https://bugs.launchpad.net/ubuntu/+bug/1869825| |WAS|websphere application server instance (IBM)| |WAS6|websphere application server instance (IBM)|

Resource Agents: [deprecated]

Agents in this list are NOT supported in Ubuntu OR Upstream (due to being deprecated in favor of other agents) and **might be removed in future Ubuntu HA versions.**

RESOURCE AGENT	RESOURCE AGENT DESCRIPTION

||

DEPRECATED

|Evmsd|clustered evms vol mgmt (evms is not maintained)| |EvmsSCC|clustered evms vol mgmt (evms is not maintained)| |LinuxSCSI|enables/disables scsi devs through kernel scsi hotplug| |scsi2reservation|SCSI-2 reservation agent (depends on scsi_reserve)| |ocf:heartbeat:pingd|monitors connectivity to specific hosts| |ocf:pacemaker:pingd|replaced by pacemaker:ping (this is broken)| |vmware|control vmware server 2.0 virtual machines (2009)|

Ubuntu HA - Pacemaker Fence Agents Supportability

After discussions among Ubuntu Developers, it was decided that Ubuntu project should focus in splitting all existing Pacemaker Fence Agents into different categories:

- Fence Agents: main
- Fence Agents: [universe]
- Fence Agents: [universe]-community
- Fence Agents: [non-supported]
- Fence Agents: [deprecated]

  Note: There is a current plan to split fence agents into different packages so supported agents can be installed independently.

Fence Agents: main

Agents in this list are supported just like any other package available in **main repository** would be.

FENCE AGENT	FENCE AGENT DESCRIPTION	EXTRA DESCRIPTION

||

POWER FENCING AGENTS|| |fence_ipmilan|IPMI controlled machines|| |fence_ilo3|symlink to fence_ipmilan|| |fence_ilo4|symlink to fence_ipmilan|| |fence_ilo5|symlink to fence_ipmilan|| |fence_imm|symlink to fence_ipmilan|| |fence_idrac|symlink to fence_ipmilan|| ||

STORAGE FENCING AGENTS|| |fence_mpath|SCSI-3 persistent reservations to control mpath devs|| |fence_sbd|SBD (shared storage) fencing|| |fence_scsi|SCSI-3 persistent reservations|| ||

VIRTUALIZATION FENCING AGENTS|| |fence_virsh|Libvirt managed virtual machines||

Fence Agents: [universe]

Agents in this list are supported just like any other package available in **[universe] repository** would be.

FENCE AGENT	FENCE AGENT DESCRIPTION	EXTRA DESCRIPTION

||

GENERIC AGENTS|| |fence_dummy|dummy fence agent (for testing)|| |fence_ack_manual|override fencing operations manually|| |fence_kdump|kdump crash recovery service (acks being in kdump)|| |fence_kdump_send|-|acks entered kdump crash recovery service (tool)| ||

POWER FENCING AGENTS|| |fence_amt|Intel AMT|| |fence_intelmodular|Intel Modular device (Intel MFSYS25, MFSYS35)|| |fence_ilo|HP servers with iLO|| |fence_ilo2|-|symlink to fence_ilo| |fence_ilo_ssh|HP iLO devices through ssh|| |fence_ilo3_ssh|-|symlink to fence_ilo_ssh| |fence_ilo4_ssh|-|symlink to fence_ilo_ssh| |fence_ilo5_ssh|-|symlink to fence_ilo_ssh| |fence_ilo_moonshot|HP Moonshot iLO|| |fence_ilo_mp|HP iLO MP|| |fence_hpblade|HP BladeSystem and HP Integrity Superdome X|| |fence_cisco_ucs|Cisco UCS (fence machines)|| |fence_alom|Sun Microsystems ALOM|| |fence_bladecenter|IBM BladeCenter (w/ telnet/ssh support)|| |fence_ipdu|IBM iPDU (network power switch) over SNMP|| |fence_lpar|IBM LPARs (using HMC)|| |fence_ibmblade|IBM BladeCenter over SNMP|| |fence_rsa|IBM RSA II management interface|| |fence_drac|Dell remote access card|| |fence_drac5|Dell remote access card v5 or CMC (DRAC)|| |fence_hds_cb|Hitachi Compute Blade systems|| |fence_rsb|Fujitsu-Siemens RSB management interface|| ||

NETWORK FENCING AGENTS|| |fence_heuristics_ping|uses ping-heuristics to exec other fence in same level|| |fence_ifmib|Any SNMP IF-MIB device (ethernet switches w/ iSCSI)|| ||

STORAGE FENCING AGENTS|| |fence_cisco_mds|Cisco MDS 9000 w/ SNMP enabled|| |fence_sanbox2|QLogic SANBox2 FC switches|| |fence_brocade|Brocade FC switches (disables specified port)|| ||

VIRTUALIZATION FENCING AGENTS|| |fence_vmware|VMware ESX|ESXi virtual machines| |fence_vmware_rest|VMware virtual machines (w/ VMware API)|| |fence_vmware_soap|VMware virtual machines (w/ SOAP API v4.1+)|| |fence_vmware_vcloud|VMware virtual machines on VMware vCloud Director API|| ||

CLOUD FENCING AGENTS|| |fence_aws|AWS (boto3 library)|| |fence_azure_arm|Azure Resource Manager (Azure SDK for Python)|| |fence_compute|OpenStack Nova|| |fence_evacuate|OpenStack Nova|| |fence_openstack|Fence OpenStack's Nova (w/ python-novaclient)|| |fence_gce|Google Cloud Engine (googleapiclient lib)|| ||

CONTAINERS FENCING AGENTS|| |fence_docker|Docker engine containers||

Fence Agents: [universe]-community

Agents in this list are **only supported by the upstream community**. All bugs opened against these agents will be forwarded to upstream IF makes sense (affected version is close to upstream).

FENCE AGENT	FENCE AGENT DESCRIPTION	EXTRA DESCRIPTION

||

POWER FENCING AGENTS|| |fence_apc|APC network power switch|| |fence_apc_snmp|APC network power switch or Tripplite PDU devices|| |fence_tripplite_snmp|-|symlink to fence_tripplite_snmp| |fence_rdc_serial|Serial cable to reset Motherboard switches|| |fence_eaton_snmp|Eaton network power switch (SNMP)|| |fence_emerson|Emerson MPX and MPH2 managed rack PDU|| |fence_eps|ePowerSwitch 8M+|| |fence_netio|Koukaam NETIO-230B PDU (telnet)|| |fence_powerman|Powerman management utility (LLNL Systems)|| |fence_raritan|Raritan Dominion PX (DPXS12-20 PDU)|| |fence_redfish|Out-of-Band controllers supporting Redfish APIs|| |fence_wti|WTI Network Power Switch (NPS)|| ||

VIRTUALIZATION FENCING AGENTS|| |fence_xenapi|Citrix Xen Server over XenAPI|| |fence_vbox|VirtualBox virtual machines|| |fence_pve|Proxmox Virtual Environment|| |fence_zvm|Power fencing on GFS VM in z/VM cluster (z/VM SMAPI)|| |fence_zvmip|z/VM virtual machines (z/VM SMAPI via TCP/IP)|| ||

CLOUD FENCING AGENTS|| |fence_ovh|OVH Cloud Data Centers|| |fence_aliyun|Aliyun||

Fence Agents: [non-supported]

Agents in this list are NOT supported in Ubuntu and **might be removed in future Ubuntu HA versions.**

|FENCE AGENT|

FENCE AGENT DESCRIPTION	EXTRA DESCRIPTION
fence_ironic	OpenStack's Ironic (not intended for production)
fence_rhevm	RHEV-M REST API to fence virtual machines
fence_ldom	Sun Microsystems Logical Domain virtual machines

Fence Agents: [deprecated]

Agents in this list are NOT supported in Ubuntu OR Upstream (due to being deprecated in favor of other agents) and **might be removed in future Ubuntu HA versions.**

|FENCE AGENT|

FENCE AGENT DESCRIPTION	EXTRA DESCRIPTION	
N/A	N/A	N/A

Ubuntu HA - DRBD

Distributed Replicated Block Device (DRBD) mirrors block devices between multiple hosts. The replication is transparent to other applications on the host systems. Any block device hard disks, partitions, RAID devices, logical volumes, etc can be mirrored.

To get started using drbd, first install the necessary packages. From a terminal enter:

```
sudo apt install drbd8-utils
```

> **Note**
>
> If you are using the *virtual kernel* as part of a virtual machine you will need to manually compile the drbd module. It may be easier to install the linux-server package inside the virtual machine.

This section covers setting up a drbd to replicate a separate /srv partition, with an ext3 filesystem between two hosts. The partition size is not particularly relevant, but both partitions need to be the same size.

Configuration

The two hosts in this example will be called *drbd01* and *drbd02*. They will need to have name resolution configured either through DNS or the /etc/hosts file. See ??? for details.

- To configure drbd, on the first host edit /etc/drbd.conf:

```
global { usage-count no; }
common { syncer { rate 100M; } }
resource r0 {
        protocol C;
        startup {
                wfc-timeout  15;
                degr-wfc-timeout 60;
        }
        net {
                cram-hmac-alg sha1;
                shared-secret "secret";
        }
        on drbd01 {
                device /dev/drbd0;
                disk /dev/sdb1;
                address 192.168.0.1:7788;
                meta-disk internal;
        }
        on drbd02 {
                device /dev/drbd0;
                disk /dev/sdb1;
                address 192.168.0.2:7788;
                meta-disk internal;
        }
}
```

> **Note**
>
> There are many other options in /etc/drbd.conf, but for this example their default values are fine.

- Now copy /etc/drbd.conf to the second host:

```
scp /etc/drbd.conf drbd02:~
```

- And, on *drbd02* move the file to /etc:

  ```
  sudo mv drbd.conf /etc/
  ```

- Now using the drbdadm utility initialize the meta data storage. On each server execute:

  ```
  sudo drbdadm create-md r0
  ```

- Next, on both hosts, start the drbd daemon:

  ```
  sudo systemctl start drbd.service
  ```

- On the *drbd01*, or whichever host you wish to be the primary, enter the following:

  ```
  sudo drbdadm -- --overwrite-data-of-peer primary all
  ```

- After executing the above command, the data will start syncing with the secondary host. To watch the progress, on *drbd02* enter the following:

  ```
  watch -n1 cat /proc/drbd
  ```

  To stop watching the output press *Ctrl+c*.

- Finally, add a filesystem to /dev/drbd0 and mount it:

  ```
  sudo mkfs.ext3 /dev/drbd0
  sudo mount /dev/drbd0 /srv
  ```

Testing

To test that the data is actually syncing between the hosts copy some files on the *drbd01*, the primary, to /srv:

```
sudo cp -r /etc/default /srv
```

Next, unmount /srv:

```
sudo umount /srv
```

Demote the *primary* server to the *secondary* role:

```
sudo drbdadm secondary r0
```

Now on the *secondary* server *promote* it to the *primary* role:

```
sudo drbdadm primary r0
```

Lastly, mount the partition:

```
sudo mount /dev/drbd0 /srv
```

Using *ls* you should see /srv/default copied from the former *primary* host *drbd01*.

References

- For more information on DRBD see the DRBD web site.
- The drbd.conf man page contains details on the options not covered in this guide.
- Also, see the drbdadm man page.
- The DRBD Ubuntu Wiki page also has more information.

Byobu

One of the most useful applications for any system administrator is an xterm multiplexor such as screen or tmux. It allows for the execution of multiple shells in one terminal. To make some of the advanced multiplexor features more user-friendly and provide some useful information about the system, the byobu package was created. It acts as a wrapper to these programs. By default Byobu is installed in Ubuntu server and it uses tmux (if installed) but this can be changed by the user.

Invoke it simply with:

```
byobu
```

Now bring up the configuration menu. By default this is done by pressing the *F9* key. This will allow you to:

- Help – Quick Start Guide
- Toggle status notifications
- Change the escape sequence
- Byobu currently does not launch at login (toggle on)

byobu provides a menu which displays the Ubuntu release, processor information, memory information, and the time and date. The effect is similar to a desktop menu.

Using the *"Byobu currently does not launch at login (toggle on)"* option will cause byobu to be executed any time a terminal is opened. Changes made to byobu are on a per user basis, and will not affect other users on the system.

One difference when using byobu is the *scrollback* mode. Press the *F7* key to enter scrollback mode. Scrollback mode allows you to navigate past output using *vi* like commands. Here is a quick list of movement commands:

- *h* - Move the cursor left by one character
- *j* - Move the cursor down by one line
- *k* - Move the cursor up by one line
- *l* - Move the cursor right by one character
- *0* - Move to the beginning of the current line
- *$* - Move to the end of the current line
- *G* - Moves to the specified line (defaults to the end of the buffer)
- */* - Search forward
- *?* - Search backward
- *n* - Moves to the next match, either forward or backward

Resources

- For more information on screen see the screen web site.
- And the Ubuntu Wiki screen page.
- Also, see the byobu project page for more information.

etckeeper

etckeeper allows the contents of /etc to be stored in a Version Control System (VCS) repository. It integrates with APT and automatically commits changes to /etc when packages are installed or upgraded. Placing /etc under version control is considered an industry best practice, and the goal of etckeeper is to make this process as painless as possible.

Install etckeeper by entering the following in a terminal:

```
sudo apt install etckeeper
```

The main configuration file, /etc/etckeeper/etckeeper.conf, is fairly simple. The main option is which VCS to use and by default etckeeper is configured to use git. The repository is automatically initialized (and committed for the first time) during package installation. It is possible to undo this by entering the following command:

```
sudo etckeeper uninit
```

By default, etckeeper will commit uncommitted changes made to /etc daily. This can be disabled using the AVOID_DAILY_AUTOCOMMITS configuration option. It will also automatically commit changes before and after package installation. For a more precise tracking of changes, it is recommended to commit your changes manually, together with a commit message, using:

```
sudo etckeeper commit "Reason for configuration change"
```

The vcs etckeeper command allows to run any subcommand of the VCS that etckeeper is configured to run. t will be run in /etc. For example, in the case of git:

```
sudo etckeeper vcs log /etc/passwd
```

To demonstrate the integration with the package management system (APT), install postfix:

```
sudo apt install postfix
```

When the installation is finished, all the postfix configuration files should be committed to the repository:

```
[master 5a16a0d] committing changes in /etc made by "apt install postfix"
 Author: Your Name <xyz@example.com>
 36 files changed, 2987 insertions(+), 4 deletions(-)
 create mode 100755 init.d/postfix
 create mode 100644 insserv.conf.d/postfix
 create mode 100755 network/if-down.d/postfix
 create mode 100755 network/if-up.d/postfix
 create mode 100644 postfix/dynamicmaps.cf
 create mode 100644 postfix/main.cf
 create mode 100644 postfix/main.cf.proto
 create mode 120000 postfix/makedefs.out
 create mode 100644 postfix/master.cf
 create mode 100644 postfix/master.cf.proto
```

```
create mode 100755 postfix/post-install
create mode 100644 postfix/postfix-files
create mode 100755 postfix/postfix-script
create mode 100755 ppp/ip-down.d/postfix
create mode 100755 ppp/ip-up.d/postfix
create mode 120000 rc0.d/K01postfix
create mode 120000 rc1.d/K01postfix
create mode 120000 rc2.d/S01postfix
create mode 120000 rc3.d/S01postfix
create mode 120000 rc4.d/S01postfix
create mode 120000 rc5.d/S01postfix
create mode 120000 rc6.d/K01postfix
create mode 100755 resolvconf/update-libc.d/postfix
create mode 100644 rsyslog.d/postfix.conf
create mode 120000 systemd/system/multi-user.target.wants/postfix.service
create mode 100644 ufw/applications.d/postfix
```

For an example of how etckeeper tracks manual changes, add new a host to /etc/hosts. Using git you can see which files have been modified:

```
sudo etckeeper vcs status
```

and how:

```
sudo etckeeper vcs diff
```

If you are happy with the changes you can now commit them:

```
sudo etckeeper commit "added new host"
```

Resources

- See the etckeeper site for more details on using etckeeper.
- For documentation on the git VCS tool see the Git website.

Munin

Installation

Before installing Munin on *server01* apache2 will need to be installed. The default configuration is fine for running a munin server. For more information see ???.

First, on *server01* install munin. In a terminal enter:

```
sudo apt install munin
```

Now on *server02* install the munin-node package:

```
sudo apt install munin-node
```

Configuration

On *server01* edit the /etc/munin/munin.conf adding the IP address for *server02*:

```
## First our "normal" host.
[server02]
        address 172.18.100.101
```

Note

Replace *server02* and *172.18.100.101* with the actual hostname and IP address for your server.

Next, configure munin-node on *server02*. Edit /etc/munin/munin−node.conf to allow access by *server01*:

```
allow ^172\.18\.100\.100$
```

Note

Replace *^172\.18\.100\.100$* with IP address for your munin server.

Now restart munin-node on *server02* for the changes to take effect:

```
sudo systemctl restart munin−node.service
```

Finally, in a browser go to *http://server01/munin*, and you should see links to nice graphs displaying information from the standard *munin-plugins* for disk, network, processes, and system.

Note

Since this is a new install it may take some time for the graphs to display anything useful.

Additional Plugins

The munin-plugins-extra package contains performance checks additional services such as DNS, DHCP, Samba, etc. To install the package, from a terminal enter:

```
sudo apt install munin−plugins−extra
```

Be sure to install the package on both the server and node machines.

References

- See the Munin website for more details.
- Specifically the Munin Documentation page includes information on additional plugins, writing plugins, etc.

Nagios

Installation

First, on *server01* install the nagios package. In a terminal enter:

```
sudo apt install nagios3 nagios−nrpe−plugin
```

You will be asked to enter a password for the *nagiosadmin* user. The user's credentials are stored in /etc/nagios3/htpasswd.users. To change the *nagiosadmin* password, or add additional users to the Nagios CGI scripts, use the htpasswd that is part of the apache2-utils package.

For example, to change the password for the *nagiosadmin* user enter:

```
sudo htpasswd /etc/nagios3/htpasswd.users nagiosadmin
```

To add a user:

```
sudo htpasswd /etc/nagios3/htpasswd.users steve
```

Next, on *server02* install the nagios-nrpe-server package. From a terminal on server02 enter:

```
sudo apt install nagios-nrpe-server
```

> **Note**
>
> NRPE allows you to execute local checks on remote hosts. There are other ways of accomplishing this through other Nagios plugins as well as other checks.

Configuration Overview

There are a couple of directories containing Nagios configuration and check files.

- /etc/nagios3: contains configuration files for the operation of the nagios daemon, CGI files, hosts, etc.
- /etc/nagios-plugins: houses configuration files for the service checks.
- /etc/nagios: on the remote host contains the nagios-nrpe-server configuration files.
- /usr/lib/nagios/plugins/: where the check binaries are stored. To see the options of a check use the *-h* option.

  For example: /usr/lib/nagios/plugins/check_dhcp -h

There are a plethora of checks Nagios can be configured to execute for any given host. For this example Nagios will be configured to check disk space, DNS, and a MySQL hostgroup. The DNS check will be on *server02*, and the MySQL hostgroup will include both *server01* and *server02*.

> **Note**
>
> See ??? for details on setting up Apache, ??? for DNS, and ??? for MySQL.

Additionally, there are some terms that once explained will hopefully make understanding Nagios configuration easier:

- *Host*: a server, workstation, network device, etc that is being monitored.
- *Host Group*: a group of similar hosts. For example, you could group all web servers, file server, etc.
- *Service*: the service being monitored on the host. Such as HTTP, DNS, NFS, etc.
- *Service Group*: allows you to group multiple services together. This is useful for grouping multiple HTTP for example.
- *Contact*: person to be notified when an event takes place. Nagios can be configured to send emails, SMS messages, etc.

By default Nagios is configured to check HTTP, disk space, SSH, current users, processes, and load on the *localhost*. Nagios will also ping check the *gateway*.

Large Nagios installations can be quite complex to configure. It is usually best to start small, one or two hosts, get things configured the way you like then expand.

Configuration

- First, create a *host* configuration file for *server02*. Unless otherwise specified, run all these commands on *server01*. In a terminal enter:

```
sudo cp /etc/nagios3/conf.d/localhost_nagios2.cfg \
/etc/nagios3/conf.d/server02.cfg
```

> **Note**
>
> In the above and following command examples, replace *"server01"*, *"server02"* *172.18.100.100*, and *172.18.100.101* with the host names and IP addresses of your servers.

Next, edit /etc/nagios3/conf.d/server02.cfg:

```
define host{
        use                     generic-host   ; Name of host template to
            use
        host_name               server02
        alias                   Server 02
        address                 172.18.100.101
}

# check DNS service.
define service {
        use                             generic-service
        host_name                       server02
        service_description             DNS
        check_command                   check_dns!172.18.100.101
}
```

Restart the nagios daemon to enable the new configuration:

```
sudo systemctl restart nagio3.service
```

- Now add a service definition for the MySQL check by adding the following to /etc/nagios3/conf.d/services_nagios2.cfg:

```
# check MySQL servers.
define service {
        hostgroup_name          mysql-servers
        service_description     MySQL
        check_command           check_mysql_cmdlinecred!nagios!secret!
            $HOSTADDRESS
        use                     generic-service
        notification_interval 0 ; set > 0 if you want to be renotified
}
```

A *mysql-servers* hostgroup now needs to be defined. Edit /etc/nagios3/conf.d/hostgroups_nagios2.cfg adding:

```
# MySQL hostgroup.
define hostgroup {
        hostgroup_name  mysql-servers
                alias           MySQL servers
                members         localhost, server02
        }
```

The Nagios check needs to authenticate to MySQL. To add a *nagios* user to MySQL enter:

```
mysql -u root -p -e "create user nagios identified by 'secret';"
```

Note

The *nagios* user will need to be added all hosts in the *mysql-servers* hostgroup.

Restart nagios to start checking the MySQL servers.

```
sudo systemctl restart nagios3.service
```

- Lastly configure NRPE to check the disk space on *server02*.

  On *server01* add the service check to /etc/nagios3/conf.d/server02.cfg:

```
# NRPE disk check.
define service {
        use                     generic-service
        host_name               server02
        service_description     nrpe-disk
        check_command           check_nrpe_1arg!check_all_disks
            !172.18.100.101
}
```

  Now on *server02* edit /etc/nagios/nrpe.cfg changing:

```
allowed_hosts=172.18.100.100
```

  And below in the command definition area add:

```
command[check_all_disks]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10%
    -e
```

  Finally, restart nagios-nrpe-server:

```
sudo systemctl restart nagios-nrpe-server.service
```

  Also, on *server01* restart nagios:

```
sudo systemctl restart nagios3.service
```

You should now be able to see the host and service checks in the Nagios CGI files. To access them point a browser to http://server01/nagios3. You will then be prompted for the *nagiosadmin* username and password.

References

This section has just scratched the surface of Nagios' features. The nagios-plugins-extra and nagios-snmp-plugins contain many more service checks.

- For more information see Nagios website.
- Specifically the Nagios Online Documentation site.
- There is also a list of books related to Nagios and network monitoring:
- The Nagios Ubuntu Wiki page also has more details.

pam_motd

When logging into an Ubuntu server you may have noticed the informative Message Of The Day (MOTD). This information is obtained and displayed using a couple of packages:

- *landscape-common:* provides the core libraries of landscape-client, which is needed to manage systems with Landscape (proprietary). Yet the package also includes the landscape-sysinfo utility which is responsible for displaying core system data involving cpu, memory, disk space, etc. For instance:

```
System load:  0.0               Processes:           76
Usage of /:   30.2% of 3.11GB   Users logged in:     1
Memory usage: 20%               IP address for eth0: 10.153.107.115
Swap usage:   0%

Graph this data and manage this system at https://landscape.
    canonical.com/
```

  Note

  You can run landscape-sysinfo manually at any time.

- *update-notifier-common:* provides information on available package updates, impending filesystem checks (fsck), and required reboots (e.g.: after a kernel upgrade).

pam_motd executes the scripts in /etc/update−motd.d in order based on the number prepended to the script. The output of the scripts is written to /var/run/motd, keeping the numerical order, then concatenated with /etc/motd.tail.

You can add your own dynamic information to the MOTD. For example, to add local weather information:

- First, install the weather-util package:

```
sudo apt install weather-util
```

- The weather utility uses METAR data from the National Oceanic and Atmospheric Administration and forecasts from the National Weather Service. In order to find local information you will need the 4-character ICAO location indicator. This can be determined by browsing to the National Weather Service site.

  Although the National Weather Service is a United States government agency there are weather stations available world wide. However, local weather information for all locations outside the U.S. may not be available.

- Create /usr/local/bin/local−weather, a simple shell script to use weather with your local ICAO indicator:

```
#!/bin/sh
#
#
# Prints the local weather information for the MOTD.
#
#

# Replace KINT with your local weather station.
# Local stations can be found here: http://www.weather.gov/tg/siteloc.
    shtml

echo
```

```
weather KINT
echo
```

- Make the script executable:

  ```
  sudo chmod 755 /usr/local/bin/local-weather
  ```

- Next, create a symlink to /etc/update-motd.d/98-local-weather:

  ```
  sudo ln -s /usr/local/bin/local-weather /etc/update-motd.d/98-local-
      weather
  ```

- Finally, exit the server and re-login to view the new MOTD.

You should now be greeted with some useful information, and some information about the local weather that may not be quite so useful. Hopefully the local-weather example demonstrates the flexibility of pam_motd.

Resources

- See the update-motd man page for more options available to update-motd.
- The Debian Package of the Day weather article has more details about using the weatherutility.

Puppet

Puppet is a cross platform framework enabling system administrators to perform common tasks using code. The code can do a variety of tasks from installing new software, to checking file permissions, or updating user accounts. Puppet is great not only during the initial installation of a system, but also throughout the system's entire life cycle. In most circumstances puppet will be used in a client/server configuration.

This section will cover installing and configuring Puppet in a client/server configuration. This simple example will demonstrate how to install Apache using Puppet.

Preconfiguration

Prior to configuring puppet you may want to add a DNS *CNAME* record for *puppet.example.com*, where *example.com* is your domain. By default Puppet clients check DNS for puppet.example.com as the puppet server name, or *Puppet Master*. See Domain Name Server for more details.

If you do not wish to use DNS, you can add entries to the server and client /etc/hosts file. For example, in the Puppet server's /etc/hosts file add:

```
127.0.0.1 localhost.localdomain localhost puppet
192.168.1.17 puppetclient.example.com puppetclient
```

On each Puppet client, add an entry for the server:

```
192.168.1.16 puppetmaster.example.com puppetmaster puppet
```

> **Note**
>
> Replace the example IP addresses and domain names above with your actual server and client addresses and domain names.

Installation

To install Puppet, in a terminal on the *server* enter:

```
sudo apt install puppetmaster
```

On the *client* machine, or machines, enter:

```
sudo apt install puppet
```

Configuration

Create a folder path for the apache2 class:

```
sudo mkdir -p /etc/puppet/modules/apache2/manifests
```

Now setup some resources for apache2. Create a file /etc/puppet/modules/apache2/manifests/init.pp containing the following:

```
class apache2 {
  package { 'apache2':
    ensure => installed,
  }

  service { 'apache2':
    ensure  => true,
    enable  => true,
    require => Package['apache2'],
  }
}
```

Next, create a node file /etc/puppet/code/environments/production/manifests/site.pp with:

```
node 'puppetclient.example.com' {
   include apache2
}
```

> **Note**
>
> Replace *puppetclient.example.com* with your actual Puppet client's host name.

The final step for this simple Puppet server is to restart the daemon:

```
sudo systemctl restart puppetmaster.service
```

Now everything is configured on the Puppet server, it is time to configure the client.

First, configure the Puppet agent daemon to start. Edit /etc/default/puppet, changing *START* to yes:

```
START=yes
```

Then start the service:

```
sudo systemctl start puppet.service
```

View the client cert fingerprint

```
sudo puppet agent --fingerprint
```

Back on the Puppet server, view pending certificate signing requests:

```
sudo puppet cert list
```

On the Puppet server, verify the fingerprint of the client and sign puppetclient's cert:

```
sudo puppet cert sign puppetclient.example.com
```

On the Puppet client, run the puppet agent manually in the foreground. This step isn't strictly speaking necessary, but it is the best way to test and debug the puppet service.

```
sudo puppet agent —test
```

Check /var/log/syslog on both hosts for any errors with the configuration. If all goes well the apache2 package and it's dependencies will be installed on the Puppet client.

> **Note**
>
> This example is *very* simple, and does not highlight many of Puppet's features and benefits. For more information see Resources.

Resources

- See the Official Puppet Documentation web site.
- See the Puppet forge, online repository of puppet modules.
- Also see Pro Puppet.

Zentyal

Zentyal is a Linux small business server that can be configured as a gateway, infrastructure manager, unified threat manager, office server, unified communication server or a combination of them. All network services managed by Zentyal are tightly integrated, automating most tasks. This saves time and helps to avoid errors in network configuration and administration. Zentyal is open source, released under the GNU General Public License (GPL) and runs on top of Ubuntu GNU/Linux.

Zentyal consists of a series of packages (usually one for each module) that provide a web interface to configure the different servers or services. The configuration is stored on a key-value Redis database, but users, groups, and domains-related configuration is on OpenLDAP. When you configure any of the available parameters through the web interface, final configuration files are overwritten using the configuration templates provided by the modules. The main advantage of using Zentyal is a unified, graphical user interface to configure all network services and high, out-of-the-box integration between them.

Zentyal publishes one major stable release once a year based on the latest Ubuntu LTS release.

Installation

If you would like to create a new user to access the Zentyal web interface, run:

```
sudo adduser username sudo
```

Add the Zentyal repository to your repository list:

```
sudo add-apt-repository "deb http://archive.zentyal.org/zentyal 3.5 main extra
    "
```

Import the public keys from Zentyal:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 10E239FF
wget -q http://keys.zentyal.org/zentyal-4.2-archive.asc -O- | sudo apt-key add
    -
```

Update your packages and install Zentyal:

```
sudo apt update
sudo apt install zentyal
```

During installation you will be asked to set a root MySQL password and confirm port 443.

First steps

Any system account belonging to the sudo group is allowed to log into the Zentyal web interface. The user created while installing Ubuntu Server will belong to the sudo group by default.

To access the Zentyal web interface, point a browser to https://localhost/ or to the IP address of your remote server. As Zentyal creates its own self-signed SSL certificate, you will have to accept a security exception on your browser. Log in with the same username and password used to log in to your server.

Once logged in you will see an overview of your server. Individual modules, such as Antivirus or Firewall, can be installed by simply clicking them and then clicking Install. Selecting server roles like Gateway or Infrastructure can be used to install multiple modules at once.

Modules can also be installed via the command line:

```
sudo apt install <zentyal-module>
```

See the list of available modules below.

To enable a module, go to the Dashboard, then click Module Status. Click the check box for the module, then Save changes.

To configure any of the features of your installed modules, click the different sections on the left menu. When you make any changes, a red "Save changes" button appears in the upper right corner.

If you need to customize any configuration file or run certain actions (scripts or commands) to configure features not available on Zentyal, place the custom configuration file templates on /etc/zentyal/stubs/<module>/ and the hooks on /etc/zentyal/hooks/<module>.<action>. Read more about stubs and hooks here.

Modules

Zentyal 2.3 is available on Ubuntu DISTRO-REV-SHORT Universe repository. The modules available are:

- zentyal-core & zentyal-common: the core of the Zentyal interface and the common libraries of the framework. Also includes the logs and events modules that give the administrator an interface to view the logs and generate events from them.
- zentyal-network: manages the configuration of the network. From the interfaces (supporting static IP, DHCP, VLAN, bridges or PPPoE), to multiple gateways when having more than one Internet connection, load balancing and advanced routing, static routes or dynamic DNS.
- zentyal-objects & zentyal-services: provide an abstraction level for network addresses (e.g. LAN instead of 192.168.1.0/24) and ports named as services (e.g. HTTP instead of 80/TCP).
- zentyal-firewall: configures the iptables rules to block forbiden connections, NAT and port redirections.

- zentyal-ntp: installs the NTP daemon to keep server on time and allow network clients to synchronize their clocks against the server.
- zentyal-dhcp: configures ISC DHCP server supporting network ranges, static leases and other advanced options like NTP, WINS, dynamic DNS updates and network boot with PXE.
- zentyal-dns: brings ISC Bind9 DNS server into your server for caching local queries as a forwarder or as an authoritative server for the configured domains. Allows to configure A, CNAME, MX, NS, TXT and SRV records.
- zentyal-ca: integrates the management of a Certification Authority within Zentyal so users can use certificates to authenticate against the services, like with OpenVPN.
- zentyal-openvpn: allows to configure multiple VPN servers and clients using OpenVPN with dynamic routing configuration using Quagga.
- zentyal-users: provides an interface to configure and manage users and groups on OpenLDAP. Other services on Zentyal are authenticated against LDAP having a centralized users and groups management. It is also possible to synchronize users, passwords and groups from a Microsoft Active Directory domain.
- zentyal-squid: configures Squid and Dansguardian for speeding up browsing thanks to the caching capabilities and content filtering.
- zentyal-samba: allows Samba configuration and integration with existing LDAP. From the same interface you can define password policies, create shared resources and assign permissions.
- zentyal-printers: integrates CUPS with Samba and allows not only to configure the printers but also give them permissions based on LDAP users and groups.

Not present on Ubuntu Universe repositories, but on Zentyal Team PPA you will find these other modules:

- zentyal-antivirus: integrates ClamAV antivirus with other modules like the proxy, file sharing or mail-filter.
- zentyal-asterisk: configures Asterisk to provide a simple PBX with LDAP based authentication.
- zentyal-bwmonitor: allows to monitor bandwith usage of your LAN clients.
- zentyal-captiveportal: integrates a captive portal with the firewall and LDAP users and groups.
- zentyal-ebackup: allows to make scheduled backups of your server using the popular duplicity backup tool.
- zentyal-ftp: configures a FTP server with LDAP based authentication.
- zentyal-ids: integrates a network intrusion detection system.
- zentyal-ipsec: allows to configure IPsec tunnels using OpenSwan.
- zentyal-jabber: integrates ejabberd XMPP server with LDAP users and groups.
- zentyal-thinclients: a LTSP based thin clients solution.
- zentyal-mail: a full mail stack including Postfix and Dovecot with LDAP backend.
- zentyal-mailfilter: configures amavisd with mail stack to filter spam and attached virus.
- zentyal-monitor: integrates collectd to monitor server performance and running services.
- zentyal-pptp: configures a PPTP VPN server.
- zentyal-radius: integrates FreeRADIUS with LDAP users and groups.
- zentyal-software: simple interface to manage installed Zentyal modules and system updates.
- zentyal-trafficshaping: configures traffic limiting rules to do bandwidth throttling and improve latency.

- zentyal-usercorner: allows users to edit their own LDAP attributes using a web browser.
- zentyal-virt: simple interface to create and manage virtual machines based on libvirt.
- zentyal-webmail: allows to access your mail using the popular Roundcube webmail.
- zentyal-webserver: configures Apache webserver to host different sites on your machine.
- zentyal-zarafa: integrates Zarafa groupware suite with Zentyal mail stack and LDAP.

References

Zentyal Official Documentation page.

Zentyal Community Wiki.

Visit the Zentyal forum for community support, feedback, feature requests, etc.

Monitoring

Overview

The monitoring of essential servers and services is an important part of system administration. Most network services are monitored for performance, availability, or both. This section will cover installation and configuration of Nagios for availability monitoring, and Munin for performance monitoring.

The examples in this section will use two servers with hostnames *server01* and *server02*. *Server01* will be configured with Nagios to monitor services on itself and *server02*. Server01 will also be setup with the munin package to gather information from the network. Using the munin-node package, *server02* will be configured to send information to *server01*.

Hopefully these simple examples will allow you to monitor additional servers and services on your network.

Nagios

Installation

First, on *server01* install the nagios package. In a terminal enter:

```
sudo apt install nagios3 nagios-nrpe-plugin
```

You will be asked to enter a password for the *nagiosadmin* user. The user's credentials are stored in /etc/nagios3/htpasswd.users. To change the *nagiosadmin* password, or add additional users to the Nagios CGI scripts, use the htpasswd that is part of the apache2-utils package.

For example, to change the password for the *nagiosadmin* user enter:

```
sudo htpasswd /etc/nagios3/htpasswd.users nagiosadmin
```

To add a user:

```
sudo htpasswd /etc/nagios3/htpasswd.users steve
```

Next, on *server02* install the nagios-nrpe-server package. From a terminal on server02 enter:

```
sudo apt install nagios-nrpe-server
```

Note

NRPE allows you to execute local checks on remote hosts. There are other ways of accomplishing this through other Nagios plugins as well as other checks.

Configuration Overview

There are a couple of directories containing Nagios configuration and check files.

- /etc/nagios3: contains configuration files for the operation of the nagios daemon, CGI files, hosts, etc.
- /etc/nagios−plugins: houses configuration files for the service checks.
- /etc/nagios: on the remote host contains the nagios-nrpe-server configuration files.
- /usr/lib/nagios/plugins/: where the check binaries are stored. To see the options of a check use the *-h* option.

  For example: /usr/lib/nagios/plugins/check_dhcp −h

There are a plethora of checks Nagios can be configured to execute for any given host. For this example Nagios will be configured to check disk space, DNS, and a MySQL hostgroup. The DNS check will be on *server02*, and the MySQL hostgroup will include both *server01* and *server02*.

Note

See details on setting up Apache, Domain Name Service, and MySQL.

Additionally, there are some terms that once explained will hopefully make understanding Nagios configuration easier:

- *Host*: a server, workstation, network device, etc that is being monitored.
- *Host Group*: a group of similar hosts. For example, you could group all web servers, file server, etc.
- *Service*: the service being monitored on the host. Such as HTTP, DNS, NFS, etc.
- *Service Group*: allows you to group multiple services together. This is useful for grouping multiple HTTP for example.
- *Contact*: person to be notified when an event takes place. Nagios can be configured to send emails, SMS messages, etc.

By default Nagios is configured to check HTTP, disk space, SSH, current users, processes, and load on the *localhost*. Nagios will also ping check the *gateway*.

Large Nagios installations can be quite complex to configure. It is usually best to start small, one or two hosts, get things configured the way you like then expand.

Configuration

- First, create a *host* configuration file for *server02*. Unless otherwise specified, run all these commands on *server01*. In a terminal enter:

```
sudo cp /etc/nagios3/conf.d/localhost_nagios2.cfg \
/etc/nagios3/conf.d/server02.cfg
```

  Note

  In the above and following command examples, replace *"server01"*, *"server02"* *172.18.100.100*, and *172.18.100.101* with the host names and IP addresses of your servers.

Next, edit /etc/nagios3/conf.d/server02.cfg:

```
define host{
        use                     generic-host   ; Name of host template to
            use
        host_name               server02
        alias                   Server 02
        address                 172.18.100.101
}

# check DNS service.
define service {
        use                             generic-service
        host_name                       server02
        service_description             DNS
        check_command                   check_dns!172.18.100.101
}
```

Restart the nagios daemon to enable the new configuration:

```
sudo systemctl restart nagio3.service
```

- Now add a service definition for the MySQL check by adding the following to /etc/nagios3/conf.d/services_nagios2.cfg:

```
# check MySQL servers.
define service {
        hostgroup_name          mysql-servers
        service_description     MySQL
        check_command           check_mysql_cmdlinecred!nagios!secret!
            $HOSTADDRESS
        use                     generic-service
        notification_interval 0 ; set > 0 if you want to be renotified
}
```

A *mysql-servers* hostgroup now needs to be defined. Edit /etc/nagios3/conf.d/hostgroups_nagios2.cfg adding:

```
# MySQL hostgroup.
define hostgroup {
        hostgroup_name  mysql-servers
                alias           MySQL servers
                members         localhost, server02
        }
```

The Nagios check needs to authenticate to MySQL. To add a *nagios* user to MySQL enter:

```
mysql -u root -p -e "create user nagios identified by 'secret';"
```

Note

The *nagios* user will need to be added all hosts in the *mysql-servers* hostgroup.

Restart nagios to start checking the MySQL servers.

```
sudo systemctl restart nagios3.service
```

- Lastly configure NRPE to check the disk space on *server02*.

  On *server01* add the service check to /etc/nagios3/conf.d/server02.cfg:

  ```
  # NRPE disk check.
  define service {
          use                     generic-service
          host_name               server02
          service_description     nrpe-disk
          check_command           check_nrpe_1arg!check_all_disks
              !172.18.100.101
  }
  ```

  Now on *server02* edit /etc/nagios/nrpe.cfg changing:

  ```
  allowed_hosts=172.18.100.100
  ```

  And below in the command definition area add:

  ```
  command[check_all_disks]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10%
      -e
  ```

  Finally, restart nagios-nrpe-server:

  ```
  sudo systemctl restart nagios-nrpe-server.service
  ```

  Also, on *server01* restart nagios:

  ```
  sudo systemctl restart nagios3.service
  ```

You should now be able to see the host and service checks in the Nagios CGI files. To access them point a browser to http://server01/nagios3. You will then be prompted for the *nagiosadmin* username and password.

References

This section has just scratched the surface of Nagios' features. The nagios-plugins-extra and nagios-snmp-plugins contain many more service checks.

- For more information see Nagios website.
- Specifically the Nagios Online Documentation site.
- There is also a list of books related to Nagios and network monitoring:
- The Nagios Ubuntu Wiki page also has more details.

Munin

Installation

Before installing Munin on *server01* apache2 will need to be installed. The default configuration is fine for running a munin server. For more information see setting up Apache.

First, on *server01* install munin. In a terminal enter:

```
sudo apt install munin
```

Now on *server02* install the munin-node package:

```
sudo apt install munin-node
```

Configuration

On *server01* edit the /etc/munin/munin.conf adding the IP address for *server02*:

```
## First our "normal" host.
[server02]
        address 172.18.100.101
```

Note

Replace *server02* and *172.18.100.101* with the actual hostname and IP address for your server.

Next, configure munin-node on *server02*. Edit /etc/munin/munin-node.conf to allow access by *server01*:

```
allow ^172\.18\.100\.100$
```

Note

Replace *^172\.18\.100\.100$* with IP address for your munin server.

Now restart munin-node on *server02* for the changes to take effect:

```
sudo systemctl restart munin-node.service
```

Finally, in a browser go to *http://server01/munin*, and you should see links to nice graphs displaying information from the standard *munin-plugins* for disk, network, processes, and system.

Note

Since this is a new install it may take some time for the graphs to display anything useful.

Additional Plugins

The munin-plugins-extra package contains performance checks additional services such as DNS, DHCP, Samba, etc. To install the package, from a terminal enter:

```
sudo apt install munin-plugins-extra
```

Be sure to install the package on both the server and node machines.

References

- See the Munin website for more details.
- Specifically the Munin Documentation page includes information on additional plugins, writing plugins, etc.

Rsnapshot

Rsnapshot is an rsync-based filesystem snapshot utility. It can take incremental backups of local and remote filesystems for any number of machines. Rsnapshot makes extensive use of hard links, so disk space is only used when absolutely necessary. It leverages the power of rsync to create scheduled, incremental backups.

To install it open a terminal shell and run:

```
sudo apt-get install rsnapshot
```

If you want to backup a remote filesystem the rsnapshot server needs to be able to access the target machine over SSH without password. For more information on how to enable it please see OpenSSH documentation. If the backup target is a local filesystem there is no need to set up OpenSSH.

Configuration

The rsnapshot configuration resides in /etc/rsnapshot.conf. Below you can find some of the options available there.

The root directory where all snapshots will be stored:

```
snapshot_root        /var/cache/rsnapshot/
```

How many old backups you would like to keep. Since rsnapshot uses incremental backups, we can afford to keep older backups for awhile before removing them. You set these up under the BACKUP LEVELS / INTERVALS section. You tell rsnapshot to retain a specific number of backups of each kind of interval.

```
retain  daily   6
retain  weekly    7
retain  monthly   4
```

In this example we will keep 6 snapshots of our daily strategy, 7 snapshots of our weekly strategy, and 4 snapshots of our monthly strategy. These data will guide the rotation made by rsnapshot.

If you are accessing a remote machine over SSH and the port to bind is not the default (port 22), you need to set the following variable with the port number:

```
ssh_args        -p 22222
```

And the most important part, you need to decide on what you would like to backup. If you are backing up locally to the same machine, this is as easy as specifying the directories that you want to save and following it with localhost/ which will be a sub-directory in the snapshot_root that you set up earlier.

```
backup  /home/          localhost/
backup  /etc/           localhost/
backup  /usr/local/     localhost/
```

If you are backing up a remote machine you just need to tell rsnapshot where the server is and which directories you would like to back up.

```
backup root@example.com:/home/ example.com/     +rsync_long_args=--bwlimit=16,
    exclude=core
backup root@example.com:/etc/  example.com/     exclude=mtab,exclude=core
```

As you can see you can see you can pass extra rsync parameters (the + append the parameter to the default list, if you remove the + sign you override it) and also exclude directories.

You can check the comments in /etc/rsnapshot.conf and the rsnapshot man page for more options.

Test Configuration

After modifying the configuration file is a good practice to check if the syntax is ok:

```
sudo rsnapshot configtest
```

You can also test your backup levels with the following command:

sudo rsnapshot -t daily

If you are happy with the output and want to see it in action you can run:

```
sudo rsnapshot daily
```

Scheduling Backups

With rsnapshot working correctly with the current configuration, the only thing left to do is to schedule it to run at certain intervals. We will use cron to make this happen since rsnapshot includes a default cron file in /etc/cron.d/rsnapshot. If you open this file there are some entries commented out as reference.

```
0 4   * * *           root    /usr/bin/rsnapshot daily
0 3   * * 1           root    /usr/bin/rsnapshot weekly
0 2   1 * *           root    /usr/bin/rsnapshot monthly
```

The settings above added to /etc/cron.d/rsnapshot run:

- the **daily snapshot** everyday at 4:00 am
- the **weekly snapshot** every Monday at 3:00 am
- the **monthly snapshot** on the first of every month at 2:00 am

For more information on how to schedule a backup using cron please take a look at Executing with cron section in Backups - Shell Scripts.

References

- Rsnapshot offical web page
- Rsnapshot man page
- Rsync man page

PHP - Scripting Language

PHP is a general-purpose scripting language suited for Web development. PHP scripts can be embedded into HTML. This section explains how to install and configure PHP in an Ubuntu System with Apache2 and MySQL.

This section assumes you have installed and configured Apache2 Web Server and MySQL Database Server. You can refer to the Apache2 and MySQL sections in this document to install and configure Apache2 and MySQL respectively.

Installation

PHP is available in Ubuntu Linux. Unlike Python, which is installed in the base system, PHP must be added.

To install PHP and the Apache PHP module you can enter the following command at a terminal prompt:

```
sudo apt install php libapache2-mod-php
```

You can run PHP scripts at a terminal prompt. To run PHP scripts at a terminal prompt you should install the php-cli package. To install php-cli you can enter the following command:

```
sudo apt install php-cli
```

You can also execute PHP scripts without installing the Apache PHP module. To accomplish this, you should install the php-cgi package via this command:

```
sudo apt install php-cgi
```

To use MySQL with PHP you should install the php-mysql package, like so:

```
sudo apt install php-mysql
```

Similarly, to use PostgreSQL with PHP you should install the php-pgsql package:

```
sudo apt install php-pgsql
```

Configuration

If you have installed the libapache2-mod-php or php-cgi packages, you can run PHP scripts from your web browser. If you have installed the php-cli package, you can run PHP scripts at a terminal prompt.

By default, when libapache2-mod-php is installed, the Apache 2 Web server is configured to run PHP scripts using this module. Please verify if the files /etc/apache2/mods-enabled/php7.*.conf and /etc/apache2/mods-enabled/php7.*.load exist. If they do not exist, you can enable the module using the a2enmod command.

Once you have installed the PHP related packages and enabled the Apache PHP module, you should restart the Apache2 Web server to run PHP scripts, by running the following command:

```
sudo systemctl restart apache2.service
```

Testing

To verify your installation, you can run the following PHP phpinfo script:

```
<?php
  phpinfo();
?>
```

You can save the content in a file phpinfo.php and place it under the DocumentRoot directory of the Apache2 Web server. Pointing your browser to http://hostname/phpinfo.php will display the values of various PHP configuration parameters.

References

- For more in depth information see the php.net documentation.
- There are a plethora of books on PHP. A good book from O'Reilly is Learning PHP, which includes an exploration of PHP 7's enhancements to the language. PHP Cook Book, 3rd Edition is also good, but has not yet been updated for PHP 7.
- Also, see the Apache MySQL PHP Ubuntu Wiki page for more information.

Ruby on Rails

Ruby on Rails is an open source web framework for developing database backed web applications. It is optimized for sustainable productivity of the programmer since it lets the programmer to write code by favouring convention over configuration.

Installation

Before installing Rails you should install Apache (or a preferred web server) and a database service such as MySQL. To install the Apache package, please refer to HTTPD - Apache2 Web Server. For instructions on installing and configuring a database service refer to MySQL documentation for example.

Once you have a web server (e.g. Apache) and a database service (e.g. MySQL) installed and configured, you are ready to install Ruby on Rails package.

To install the Ruby base packages and Ruby on Rails, you can enter the following command in the terminal prompt:

```
sudo apt install rails
```

Configuration

Web Server

Modify the /etc/apache2/sites−available/000−default.conf configuration file to setup your domains.

The first thing to change is the *DocumentRoot* directive:

```
DocumentRoot /path/to/rails/application/public
```

Next, change the <Directory "/path/to/rails/application/public"> directive:

```
<Directory "/path/to/rails/application/public">
        Options Indexes FollowSymLinks MultiViews ExecCGI
        AllowOverride All
        Order allow,deny
        allow from all
        AddHandler cgi-script .cgi
</Directory>
```

You should also enable the mod_rewrite module for Apache. To enable mod_rewrite module, please enter the following command in a terminal prompt:

```
sudo a2enmod rewrite
```

Finally you will need to change the ownership of the /path/to/rails/application/public and /path/to/rails/application/tmp directories to the user used to run the Apache process:

```
sudo chown –R www–data:www–data /path/to/rails/application/public
sudo chown –R www–data:www–data /path/to/rails/application/tmp
```

If you need to compile your application assets run the following command in your application directory:

```
RAILS_ENV=production rake assets:precompile
```

Database

With your database service in place you need to make sure your app database configuration is also correct. For example, if you are using MySQL your config/database.yml should look like this:

```
# Mysql
production:
  adapter: mysql2
  username: user
  password: password
  host: 127.0.0.1
  database: app
```

To finally create your application database and apply its migrations you can run the following commands from your app directory:

```
RAILS_ENV=production rake db:create
RAILS_ENV=production rake db:migrate
```

That's it! Now you have your Server ready for your Ruby on Rails application. You can daemonize your application as you want.

References

- See the Ruby on Rails website for more information.
- Also Agile Development with Rails is a great resource.

Backups

There are many ways to backup an Ubuntu installation. The most important thing about backups is to develop a *backup plan* consisting of what to backup, where to back it up to, and how to restore it.

It is good practice to take backup media off-site in case of a disaster. For backup plans involving physical tape or removable hard drives, the tapes or drives can be manually taken off-site, but in other cases this may not be practical and the archives will need copied over a WAN link to a server in another location.

Archive Rotation

The shell script in Shell Scripts only allows for seven different archives. For a server whose data doesn't change often, this may be enough. If the server has a large amount of data, a more complex rotation scheme should be used.

Rotating NFS Archives

In this section, the shell script will be slightly modified to implement a grandfather-father-son rotation scheme (monthly-weekly-daily):

- The rotation will do a *daily* backup Sunday through Friday.
- On Saturday a *weekly* backup is done giving you four weekly backups a month.

- The *monthly* backup is done on the first of the month rotating two monthly backups based on if the month is odd or even.

Here is the new script:

```
#!/bin/bash
########################################
#
# Backup to NFS mount script with
# grandfather-father-son rotation.
#
########################################

# What to backup.
backup_files="/home /var/spool/mail /etc /root /boot /opt"

# Where to backup to.
dest="/mnt/backup"

# Setup variables for the archive filename.
day=$(date +%A)
hostname=$(hostname -s)

# Find which week of the month 1-4 it is.
day_num=$(date +%-d)
if (( $day_num <= 7 )); then
        week_file="$hostname-week1.tgz"
elif (( $day_num > 7 && $day_num <= 14 )); then
        week_file="$hostname-week2.tgz"
elif (( $day_num > 14 && $day_num <= 21 )); then
        week_file="$hostname-week3.tgz"
elif (( $day_num > 21 && $day_num < 32 )); then
        week_file="$hostname-week4.tgz"
fi

# Find if the Month is odd or even.
month_num=$(date +%m)
month=$(expr $month_num % 2)
if [ $month -eq 0 ]; then
        month_file="$hostname-month2.tgz"
else
        month_file="$hostname-month1.tgz"
fi

# Create archive filename.
if [ $day_num == 1 ]; then
    archive_file=$month_file
elif [ $day != "Saturday" ]; then
        archive_file="$hostname-$day.tgz"
else
    archive_file=$week_file
fi

# Print start status message.
echo "Backing up $backup_files to $dest/$archive_file"
```

```
date
echo

# Backup the files using tar.
tar czf $dest/$archive_file $backup_files

# Print end status message.
echo
echo "Backup finished"
date

# Long listing of files in $dest to check file sizes.
ls -lh $dest/
```

The script can be executed using the same methods as in Executing the Script.

As discussed in the introduction, a copy of the backup archives and/or media can then be transferred off-site.

Tape Drives

A tape drive attached to the server can be used instead of an NFS share. Using a tape drive simplifies archive rotation, and makes taking the media off-site easier as well.

When using a tape drive, the filename portions of the script aren't needed because the data is sent directly to the tape device. Some commands to manipulate the tape are needed. This is accomplished using mt, a magnetic tape control utility part of the cpio package.

Here is the shell script modified to use a tape drive:

```
#!/bin/bash
#######################################
#
# Backup to tape drive script.
#
#######################################

# What to backup.
backup_files="/home /var/spool/mail /etc /root /boot /opt"

# Where to backup to.
dest="/dev/st0"

# Print start status message.
echo "Backing up $backup_files to $dest"
date
echo

# Make sure the tape is rewound.
mt -f $dest rewind

# Backup the files using tar.
tar czf $dest $backup_files

# Rewind and eject the tape.
mt -f $dest rewoffl
```

```
# Print end status message.
echo
echo "Backup finished"
date
```

Note

The default device name for a SCSI tape drive is /dev/st0. Use the appropriate device path for your system.

Restoring from a tape drive is basically the same as restoring from a file. Simply rewind the tape and use the device path instead of a file path. For example to restore the /etc/hosts file to /tmp/etc/hosts:

```
mt -f /dev/st0 rewind
tar -xzf /dev/st0 -C /tmp etc/hosts
```

Bacula

Bacula is a backup program enabling you to backup, restore, and verify data across your network. There are Bacula clients for Linux, Windows, and Mac OS X - making it a cross-platform network wide solution.

Overview

Bacula is made up of several components and services used to manage which files to backup and backup locations:

- Bacula Director: a service that controls all backup, restore, verify, and archive operations.
- Bacula Console: an application allowing communication with the Director. There are three versions of the Console:
  - Text based command line version.
  - Gnome based GTK+ Graphical User Interface (GUI) interface.
  - wxWidgets GUI interface.
- Bacula File: also known as the Bacula Client program. This application is installed on machines to be backed up, and is responsible for the data requested by the Director.
- Bacula Storage: the programs that perform the storage and recovery of data to the physical media.
- Bacula Catalog: is responsible for maintaining the file indexes and volume databases for all files backed up, enabling quick location and restoration of archived files. The Catalog supports three different databases MySQL, PostgreSQL, and SQLite.
- Bacula Monitor: allows the monitoring of the Director, File daemons, and Storage daemons. Currently the Monitor is only available as a GTK+ GUI application.

These services and applications can be run on multiple servers and clients, or they can be installed on one machine if backing up a single disk or volume.

Installation

Note

If using MySQL or PostgreSQL as your database, you should already have the services available. Bacula will not install them for you. For more information take a look at Databases -MySQL and Databases - PostgreSQL.

There are multiple packages containing the different Bacula components. To install Bacula, from a terminal prompt enter:

```
sudo apt install bacula
```

By default installing the bacula package will use a PostgreSQL database for the Catalog. If you want to use SQLite or MySQL, for the Catalog, install bacula−director−sqlite3 or bacula−director−mysql respectively.

During the install process you will be asked to supply a password for the database *owner* of the *bacula* database.

Configuration

Bacula configuration files are formatted based on *resources* comprising of *directives* surrounded by "{}" braces. Each Bacula component has an individual file in the /etc/bacula directory.

The various Bacula components must authorize themselves to each other. This is accomplished using the *password* directive. For example, the *Storage* resource password in the /etc/bacula/bacula−dir.conf file must match the *Director* resource password in /etc/bacula/bacula−sd.conf.

By default the backup job named *BackupClient1* is configured to archive the Bacula Catalog. If you plan on using the server to backup more than one client you should change the name of this job to something more descriptive. To change the name edit /etc/bacula/bacula−dir.conf:

```
#
# Define the main nightly save backup job
#   By default, this job will back up to disk in
Job {
  Name = "BackupServer"
  JobDefs = "DefaultJob"
  Write Bootstrap = "/var/lib/bacula/Client1.bsr"
}
```

Note

The example above changes the job name to *BackupServer* matching the machine's host name. Replace "BackupServer" with your appropriate hostname, or other descriptive name.

The *Console* can be used to query the *Director* about jobs, but to use the Console with a *non-root* user, the user needs to be in the *bacula* group. To add a user to the bacula group enter the following from a terminal:

```
sudo adduser $username bacula
```

Note

Replace *$username* with the actual username. Also, if you are adding the current user to the group you should log out and back in for the new permissions to take effect.

Localhost Backup

This section describes how to backup specified directories on a single host to a local tape drive.

- First, the *Storage* device needs to be configured. Edit /etc/bacula/bacula−sd.conf add:

```
Device {
  Name = "Tape Drive"
  Device Type = tape
  Media Type = DDS−4
  Archive Device = /dev/st0
  Hardware end of medium = No;
  AutomaticMount = yes;                # when device opened, read it
  AlwaysOpen = Yes;
  RemovableMedia = yes;
  RandomAccess = no;
  Alert Command = "sh −c 'tapeinfo −f %c | grep TapeAlert'"
}
```

  The example is for a *DDS-4* tape drive. Adjust the "Media Type" and "Archive Device" to match your hardware.

  You could also uncomment one of the other examples in the file.

- After editing /etc/bacula/bacula−sd.conf the Storage daemon will need to be restarted:

```
sudo systemctl restart bacula−sd.service
```

- Now add a *Storage* resource in /etc/bacula/bacula−dir.conf to use the new Device:

```
# Definition of "Tape Drive" storage device
Storage {
  Name = TapeDrive
  # Do not use "localhost" here
  Address = backupserver                # N.B. Use a fully qualified name
      here
  SDPort = 9103
  Password = "Cv70F6pf1t6pBopT4vQOnigDrR0v3LT3Cgkiyjc"
  Device = "Tape Drive"
  Media Type = tape
}
```

  The *Address* directive needs to be the Fully Qualified Domain Name (FQDN) of the server. Change *backupserver* to the actual host name.

  Also, make sure the *Password* directive matches the password string in /etc/bacula/bacula−sd.conf.

- Create a new *FileSet*, which will determine what directories to backup, by adding:

```
# LocalhostBacup FileSet.
FileSet {
  Name = "LocalhostFiles"
  Include {
    Options {
      signature = MD5
      compression=GZIP
    }
    File = /etc
    File = /home
```

```
  }
}
```

This *FileSet* will backup the /etc and /home directories. The *Options* resource directives configure the FileSet to create an MD5 signature for each file backed up, and to compress the files using GZIP.

- Next, create a new *Schedule* for the backup job:

```
# LocalhostBackup Schedule — Daily.
Schedule {
  Name = "LocalhostDaily"
  Run = Full daily at 00:01
}
```

The job will run every day at 00:01 or 12:01 am. There are many other scheduling options available.

- Finally create the *Job*:

```
# Localhost backup.
Job {
  Name = "LocalhostBackup"
  JobDefs = "DefaultJob"
  Enabled = yes
  Level = Full
  FileSet = "LocalhostFiles"
  Schedule = "LocalhostDaily"
  Storage = TapeDrive
  Write Bootstrap = "/var/lib/bacula/LocalhostBackup.bsr"
}
```

The job will do a *Full* backup every day to the tape drive.

- Each tape used will need to have a *Label*. If the current tape does not have a label Bacula will send an email letting you know. To label a tape using the Console enter the following from a terminal:

```
bconsole
```

- At the Bacula Console prompt enter:

```
label
```

- You will then be prompted for the *Storage* resource:

```
Automatically selected Catalog: MyCatalog
Using Catalog "MyCatalog"
The defined Storage resources are:
     1: File
     2: TapeDrive
Select Storage resource (1-2):2
```

- Enter the new *Volume* name:

```
Enter new Volume name: Sunday
Defined Pools:
     1: Default
     2: Scratch
```

Replace *Sunday* with the desired label.

- Now, select the *Pool*:

```
Select the Pool (1-2): 1
Connecting to Storage daemon TapeDrive at backupserver:9103 ...
Sending label command for Volume "Sunday" Slot 0 ...
```

Congratulations, you have now configured *Bacula* to backup the localhost to an attached tape drive.

Resources

- For more *Bacula* configuration options, refer to Bacula's Documentation.
- The Bacula Home Page contains the latest Bacula news and developments.
- Also, see the Bacula Ubuntu Wiki page.

Shell Scripts

One of the simplest ways to backup a system is using a *shell script*. For example, a script can be used to configure which directories to backup, and pass those directories as arguments to the tar utility, which creates an archive file. The archive file can then be moved or copied to another location. The archive can also be created on a remote file system such as an *NFS* mount.

The tar utility creates one archive file out of many files or directories. tar can also filter the files through compression utilities, thus reducing the size of the archive file.

Simple Shell Script

The following shell script uses tar to create an archive file on a remotely mounted NFS file system. The archive filename is determined using additional command line utilities.

```
#!/bin/bash
#########################################
#
# Backup to NFS mount script.
#
#########################################

# What to backup.
backup_files="/home /var/spool/mail /etc /root /boot /opt"

# Where to backup to.
dest="/mnt/backup"

# Create archive filename.
day=$(date +%A)
hostname=$(hostname -s)
archive_file="$hostname-$day.tgz"

# Print start status message.
echo "Backing up $backup_files to $dest/$archive_file"
```

```
date
echo

# Backup the files using tar.
tar czf $dest/$archive_file $backup_files

# Print end status message.
echo
echo "Backup finished"
date

# Long listing of files in $dest to check file sizes.
ls -lh $dest
```

- *$backup_files:* a variable listing which directories you would like to backup. The list should be customized to fit your needs.
- *$day:* a variable holding the day of the week (Monday, Tuesday, Wednesday, etc). This is used to create an archive file for each day of the week, giving a backup history of seven days. There are other ways to accomplish this including using the date utility.
- *$hostname:* variable containing the *short* hostname of the system. Using the hostname in the archive filename gives you the option of placing daily archive files from multiple systems in the same directory.
- *$archive_file:* the full archive filename.
- *$dest:* destination of the archive file. The directory needs to be created and in this case *mounted* before executing the backup script. See ??? for details of using *NFS*.
- *status messages:* optional messages printed to the console using the echo utility.
- *tar czf dest/archive_file $backup_files:* the tar command used to create the archive file.
  - *c:* creates an archive.
  - *z:* filter the archive through the gzip utility compressing the archive.
  - *f:* output to an archive file. Otherwise the tar output will be sent to STDOUT.
- *ls -lh $dest:* optional statement prints a *-l* long listing in *-h* human readable format of the destination directory. This is useful for a quick file size check of the archive file. This check should not replace testing the archive file.

This is a simple example of a backup shell script; however there are many options that can be included in such a script. See References for links to resources providing more in-depth shell scripting information.

Executing the Script

Executing from a Terminal

The simplest way of executing the above backup script is to copy and paste the contents into a file. backup.sh for example. The file must be made executable:

```
chmod u+x backup.sh
```

Then from a terminal prompt:

```
sudo ./backup.sh
```

This is a great way to test the script to make sure everything works as expected.

Executing with cron

The cron utility can be used to automate the script execution. The cron daemon allows the execution of scripts, or commands, at a specified time and date.

cron is configured through entries in a crontab file. crontab files are separated into fields:

```
# m h dom mon dow   command
```

- *m:* minute the command executes on, between 0 and 59.
- *h:* hour the command executes on, between 0 and 23.
- *dom:* day of month the command executes on.
- *mon:* the month the command executes on, between 1 and 12.
- *dow:* the day of the week the command executes on, between 0 and 7. Sunday may be specified by using 0 or 7, both values are valid.
- *command:* the command to execute.

To add or change entries in a crontab file the crontab -e command should be used. Also, the contents of a crontab file can be viewed using the crontab -l command.

To execute the backup.sh script listed above using cron. Enter the following from a terminal prompt:

```
sudo crontab -e
```

> **Note**
>
> Using sudo with the crontab -e command edits the *root* user's crontab. This is necessary if you are backing up directories only the root user has access to.

Add the following entry to the crontab file:

```
# m h dom mon dow   command
0 0 * * * bash /usr/local/bin/backup.sh
```

The backup.sh script will now be executed every day at 12:00 pm.

> **Note**
>
> The backup.sh script will need to be copied to the /usr/local/bin/ directory in order for this entry to execute properly. The script can reside anywhere on the file system, simply change the script path appropriately.

For more in-depth crontab options see References.

Restoring from the Archive

Once an archive has been created it is important to test the archive. The archive can be tested by listing the files it contains, but the best test is to *restore* a file from the archive.

- To see a listing of the archive contents. From a terminal prompt type:

  ```
  tar -tzvf /mnt/backup/host-Monday.tgz
  ```

- To restore a file from the archive to a different directory enter:

  ```
  tar -xzvf /mnt/backup/host-Monday.tgz -C /tmp etc/hosts
  ```

The *-C* option to tar redirects the extracted files to the specified directory. The above example will extract the /etc/hosts file to /tmp/etc/hosts. tar recreates the directory structure that it contains.

Also, notice the leading *"/"* is left off the path of the file to restore.

- To restore all files in the archive enter the following:

```
cd /
sudo tar -xzvf /mnt/backup/host-Monday.tgz
```

Note

This will overwrite the files currently on the file system.

References

- For more information on shell scripting see the Advanced Bash-Scripting Guide
- The book Teach Yourself Shell Programming in 24 Hours is available online and a great resource for shell scripting.
- The CronHowto Wiki Page contains details on advanced cron options.
- See the GNU tar Manual for more tar options.
- The Wikipedia Backup Rotation Scheme article contains information on other backup rotation schemes.
- The shell script uses tar to create the archive, but there many other command line utilities that can be used. For example:
  - cpio: used to copy files to and from archives.
  - dd: part of the coreutils package. A low level utility that can copy data from one format to another.
  - rsnapshot: a file system snapshot utility used to create copies of an entire file system. Also check the Tools - rsnapshot for some information.
  - rsync: a flexible utility used to create incremental copies of files.

LAMP Applications

Overview

LAMP installations (Linux + Apache + MySQL + PHP/Perl/Python) are a popular setup for Ubuntu servers. There is a plethora of Open Source applications written using the LAMP application stack. Some popular LAMP applications are Wiki's, Content Management Systems, and Management Software such as phpMyAdmin.

One advantage of LAMP is the substantial flexibility for different database, web server, and scripting languages. Popular substitutes for MySQL include PostgreSQL and SQLite. Python, Perl, and Ruby are also frequently used instead of PHP. While Nginx, Cherokee and Lighttpd can replace Apache.

The fastest way to get started is to install LAMP using tasksel. Tasksel is a Debian/Ubuntu tool that installs multiple related packages as a co-ordinated "task" onto your system. To install a LAMP server:

At a terminal prompt enter the following command:

```
sudo tasksel install lamp-server
```

After installing it you'll be able to install most *LAMP* applications in this way:

- Download an archive containing the application source files.
- Unpack the archive, usually in a directory accessible to a web server.
- Depending on where the source was extracted, configure a web server to serve the files.
- Configure the application to connect to the database.
- Run a script, or browse to a page of the application, to install the database needed by the application.
- Once the steps above, or similar steps, are completed you are ready to begin using the application.

A disadvantage of using this approach is that the application files are not placed in the file system in a standard way, which can cause confusion as to where the application is installed. Another larger disadvantage is updating the application. When a new version is released, the same process used to install the application is needed to apply updates.

Fortunately, a number of *LAMP* applications are already packaged for Ubuntu, and are available for installation in the same way as non-LAMP applications. Depending on the application some extra configuration and setup steps may be needed, however.

This section covers how to install some *LAMP* applications.

Moin Moin

MoinMoin is a wiki engine implemented in Python, based on the PikiPiki Wiki engine, and licensed under the GNU GPL.

Installation

To install MoinMoin, run the following command in the command prompt:

```
sudo apt install python-moinmoin
```

You should also install apache2 web server. For installing the apache2 web server, please refer to ??? subsection in ??? section.

Configuration

To configure your first wiki application, please run the following set of commands. Let us assume that you are creating a wiki named *mywiki*:

```
cd /usr/share/moin
sudo mkdir mywiki
sudo cp -R data mywiki
sudo cp -R underlay mywiki
sudo cp server/moin.cgi mywiki
sudo chown -R www-data:www-data mywiki
sudo chmod -R ug+rwX mywiki
sudo chmod -R o-rwx mywiki
```

Now you should configure MoinMoin to find your new wiki *mywiki*. To configure MoinMoin, open /etc/moin /mywiki.py file and change the following line:

```
data_dir = '/org/mywiki/data'
```

to

```
data_dir = '/usr/share/moin/mywiki/data'
```

Also, below the *data_dir* option add the *data_underlay_dir*:

```
data_underlay_dir='/usr/share/moin/mywiki/underlay'
```

Note

If the /etc/moin/mywiki.py file does not exist, you should copy /usr/share/moin/config/wikifarm /mywiki.py file to /etc/moin/mywiki.py file and do the above mentioned change.

Note

If you have named your wiki as *my_wiki_name* you should insert a line "("my_wiki_name", r".*")" in /etc/moin/farmconfig.py file after the line "("mywiki", r".*")".

Once you have configured MoinMoin to find your first wiki application, *mywiki*, you should configure apache2 and make it ready for your wiki.

You should add the following lines in /etc/apache2/sites−available/000−default.conf file inside the "<VirtualHost *>" tag:

```
### moin
   ScriptAlias /mywiki "/usr/share/moin/mywiki/moin.cgi"
   alias /moin_static<version> "/usr/share/moin/htdocs"
  <Directory /usr/share/moin/htdocs>
   Order allow,deny
   allow from all
   </Directory>
### end moin
```

The version in the above example is determined by running:

```
$ moin --version
```

If the output shows version 1.9.7, your second line should be:

```
alias /moin_static197 "/usr/share/moin/htdocs"
```

Once you configure the apache2 web server and make it ready for your wiki application, you should restart it. You can run the following command to restart the apache2 web server:

```
sudo systemctl restart apache2.service
```

Verification

You can verify the Wiki application and see if it works by pointing your web browser to the following URL:

```
http://localhost/mywiki
```

For more details, please refer to the MoinMoin web site.

References

- For more information see the moinmoin Wiki.
- Also, see the Ubuntu Wiki MoinMoin page.

phpMyAdmin

phpMyAdmin is a LAMP application specifically written for administering MySQL servers. Written in PHP, and accessed through a web browser, phpMyAdmin provides a graphical interface for database administration tasks.

Installation

Before installing phpMyAdmin you will need access to a MySQL database either on the same host as that phpMyAdmin is installed on, or on a host accessible over the network. For more information see ???. From a terminal prompt enter:

```
sudo apt install phpmyadmin
```

At the prompt choose which web server to be configured for phpMyAdmin. The rest of this section will use Apache2 for the web server.

In a browser go to *http://servername/phpmyadmin*, replacing *servername* with the server's actual hostname. At the login, page enter *root* for the *username*, or another MySQL user, if you have any setup, and enter the MySQL user's password.

Once logged in you can reset the *root* password if needed, create users, create/destroy databases and tables, etc.

Configuration

The configuration files for phpMyAdmin are located in /etc/phpmyadmin. The main configuration file is /etc /phpmyadmin/config.inc.php. This file contains configuration options that apply globally to phpMyAdmin.

To use phpMyAdmin to administer a MySQL database hosted on another server, adjust the following in /etc/phpmyadmin/config.inc.php:

```
$cfg['Servers'][$i]['host'] = 'db_server';
```

> **Note**
>
> Replace *db_server* with the actual remote database server name or IP address. Also, be sure that the phpMyAdmin host has permissions to access the remote database.

Once configured, log out of phpMyAdmin and back in, and you should be accessing the new server.

The config.header.inc.php and config.footer.inc.php files are used to add a HTML header and footer to phpMyAdmin.

Another important configuration file is /etc/phpmyadmin/apache.conf, this file is symlinked to /etc/apache2/ conf–available/phpmyadmin.conf, and, once enabled, is used to configure Apache2 to serve the phpMyAdmin site. The file contains directives for loading PHP, directory permissions, etc. From a terminal type:

```
sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-available/phpmyadmin.
    conf
sudo a2enconf phpmyadmin.conf
sudo systemctl reload apache2.service
```

For more information on configuring Apache2 see ???.

References

- The phpMyAdmin documentation comes installed with the package and can be accessed from the *phpMyAdmin Documentation* link (a question mark with a box around it) under the phpMyAdmin logo. The official docs can also be access on the phpMyAdmin site.
- Also, Mastering phpMyAdmin is a great resource.
- A third resource is the phpMyAdmin Ubuntu Wiki page.

WordPress

Wordpress is a blog tool, publishing platform and CMS implemented in PHP and licensed under the GNU GPLv2.

Installation

To install WordPress, run the following comand in the command prompt:

```
sudo apt install wordpress
```

You should also install apache2 web server and mysql server. For installing apache2 web server, please refer to ??? sub-section in ??? section. For installing mysql server, please refer to ??? sub-section in ??? section.

Configuration

For configuring your first WordPress application, configure an apache site. Open /etc/apache2/sites–available/wordpress.conf and write the following lines:

```
Alias /blog /usr/share/wordpress
<Directory /usr/share/wordpress>
    Options FollowSymLinks
    AllowOverride Limit Options FileInfo
    DirectoryIndex index.php
    Order allow,deny
    Allow from all
</Directory>
<Directory /usr/share/wordpress/wp–content>
    Options FollowSymLinks
    Order allow,deny
    Allow from all
</Directory>
```

Enable this new WordPress site

```
sudo a2ensite wordpress
```

Once you configure the apache2 web server and make it ready for your WordPress application, you should restart it. You can run the following command to restart the apache2 web server:

```
sudo systemctl restart apache2.service
```

To facilitate multiple WordPress installations, the name of this configuration file is based on the Host header of the HTTP request. This means that you can have a configuration per VirtualHost by simply matching the hostname portion of this configuration with your Apache Virtual Host. e.g. /etc/wordpress/config-10.211.55.50.php, /etc/wordpress/config-hostalias1.php, etc. These instructions assume you can access Apache via the localhost hostname (perhaps by using an ssh tunnel) if not, replace /etc/wordpress/config-localhost.php with /etc/wordpress/config-NAME_OF_YOUR_VIRTUAL_HOST.php.

Once the configuration file is written, it is up to you to choose a convention for username and password to mysql for each WordPress database instance. This documentation shows only one, localhost, example.

Now configure WordPress to use a mysql database. Open /etc/wordpress/config−localhost.php file and write the following lines:

```
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'wordpress');
define('DB_PASSWORD', 'yourpasswordhere');
define('DB_HOST', 'localhost');
define('WP_CONTENT_DIR', '/usr/share/wordpress/wp−content');
?>
```

Now create this mysql database. Open a temporary file with mysql commands wordpress.sql and write the following lines:

```
CREATE DATABASE wordpress;
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER
ON wordpress.*
TO wordpress@localhost
IDENTIFIED BY 'yourpasswordhere';
FLUSH PRIVILEGES;
```

Execute these commands.

```
cat wordpress.sql | sudo mysql −−defaults−extra−file=/etc/mysql/debian.cnf
```

Your new WordPress can now be configured by visiting http://localhost/blog/wp-admin/install.php. (Or http://NAME_OF_YOUR_VIRTUAL_HOST/blog/wp-admin/install.php if your server has no GUI and you are completing WordPress configuration via a web browser running on another computer.) Fill out the Site Title, username, password, and E-mail and click Install WordPress.

Note the generated password (if applicable) and click the login password. Your WordPress is now ready for use.

References

- WordPress.org Codex
- Ubuntu Wiki WordPress

LAMP Applications

Overview

LAMP installations (Linux + Apache + MySQL + PHP/Perl/Python) are a popular setup for Ubuntu servers. There is a plethora of Open Source applications written using the LAMP application stack. Some

popular LAMP applications are Wiki's, Content Management Systems, and Management Software such as phpMyAdmin.

One advantage of LAMP is the substantial flexibility for different database, web server, and scripting languages. Popular substitutes for MySQL include PostgreSQL and SQLite. Python, Perl, and Ruby are also frequently used instead of PHP. While Nginx, Cherokee and Lighttpd can replace Apache.

The fastest way to get started is to install LAMP using tasksel. Tasksel is a Debian/Ubuntu tool that installs multiple related packages as a co-ordinated "task" onto your system. To install a LAMP server:

At a terminal prompt enter the following command:

```
sudo tasksel install lamp-server
```

After installing it you'll be able to install most *LAMP* applications in this way:

- Download an archive containing the application source files.
- Unpack the archive, usually in a directory accessible to a web server.
- Depending on where the source was extracted, configure a web server to serve the files.
- Configure the application to connect to the database.
- Run a script, or browse to a page of the application, to install the database needed by the application.
- Once the steps above, or similar steps, are completed you are ready to begin using the application.

A disadvantage of using this approach is that the application files are not placed in the file system in a standard way, which can cause confusion as to where the application is installed. Another larger disadvantage is updating the application. When a new version is released, the same process used to install the application is needed to apply updates.

Fortunately, a number of *LAMP* applications are already packaged for Ubuntu, and are available for installation in the same way as non-LAMP applications. Depending on the application some extra configuration and setup steps may be needed, however.

This section covers how to install some *LAMP* applications.

phpMyAdmin

phpMyAdmin is a LAMP application specifically written for administering MySQL servers. Written in PHP, and accessed through a web browser, phpMyAdmin provides a graphical interface for database administration tasks.

Installation

Before installing phpMyAdmin you will need access to a MySQL database either on the same host as that phpMyAdmin is installed on, or on a host accessible over the network. For more information see MySQL documentation. From a terminal prompt enter:

```
sudo apt install phpmyadmin
```

At the prompt choose which web server to be configured for phpMyAdmin. The rest of this section will use Apache2 for the web server.

In a browser go to *http://servername/phpmyadmin*, replacing *servername* with the server's actual hostname. At the login, page enter *root* for the *username*, or another MySQL user, if you have any setup, and enter the MySQL user's password.

Once logged in you can reset the *root* password if needed, create users, create/destroy databases and tables, etc.

Configuration

The configuration files for phpMyAdmin are located in /etc/phpmyadmin. The main configuration file is /etc /phpmyadmin/config.inc.php. This file contains configuration options that apply globally to phpMyAdmin.

To use phpMyAdmin to administer a MySQL database hosted on another server, adjust the following in /etc/phpmyadmin/config.inc.php:

```
$cfg['Servers'][$i]['host'] = 'db_server';
```

> **Note**
>
> Replace *db_server* with the actual remote database server name or IP address. Also, be sure that the phpMyAdmin host has permissions to access the remote database.

Once configured, log out of phpMyAdmin and back in, and you should be accessing the new server.

The config.header.inc.php and config.footer.inc.php files in /etc/phpmyadmin directory are used to add a HTML header and footer to phpMyAdmin.

Another important configuration file is /etc/phpmyadmin/apache.conf, this file is symlinked to /etc/apache2/ conf–available/phpmyadmin.conf, and, once enabled, is used to configure Apache2 to serve the phpMyAdmin site. The file contains directives for loading PHP, directory permissions, etc. From a terminal type:

```
sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-available/phpmyadmin.
    conf
sudo a2enconf phpmyadmin.conf
sudo systemctl reload apache2.service
```

For more information on configuring Apache2 see this documentation.

References

- The phpMyAdmin documentation comes installed with the package and can be accessed from the *phpMyAdmin Documentation* link (a question mark with a box around it) under the phpMyAdmin logo. The official docs can also be access on the phpMyAdmin site.
- Another resource is the phpMyAdmin Ubuntu Wiki page.

WordPress

Wordpress is a blog tool, publishing platform and CMS implemented in PHP and licensed under the GNU GPLv2.

Installation

To install WordPress, run the following comand in the command prompt:

```
sudo apt install wordpress
```

You should also install apache2 web server and mysql server. For installing apache2 web server, please refer to Apache2 documentation. For installing mysql server, please refer to MySQL documentation.

Configuration

For configuring your first WordPress application, configure an apache site. Open /etc/apache2/sites–available/wordpress.conf and write the following lines:

```
Alias /blog /usr/share/wordpress
<Directory /usr/share/wordpress>
    Options FollowSymLinks
    AllowOverride Limit Options FileInfo
    DirectoryIndex index.php
    Order allow,deny
    Allow from all
</Directory>
<Directory /usr/share/wordpress/wp–content>
    Options FollowSymLinks
    Order allow,deny
    Allow from all
</Directory>
```

Enable this new WordPress site

```
sudo a2ensite wordpress
```

Once you configure the apache2 web server and make it ready for your WordPress application, you should restart it. You can run the following command to restart the apache2 web server:

```
sudo systemctl reload apache2.service
```

To facilitate multiple WordPress installations, the name of this configuration file is based on the Host header of the HTTP request. This means that you can have a configuration per VirtualHost by simply matching the hostname portion of this configuration with your Apache Virtual Host. e.g. /etc/wordpress/config-10.211.55.50.php, /etc/wordpress/config-hostalias1.php, etc. These instructions assume you can access Apache via the localhost hostname (perhaps by using an ssh tunnel) if not, replace /etc/wordpress/config-localhost.php with /etc/wordpress/config-NAME_OF_YOUR_VIRTUAL_HOST.php.

Once the configuration file is written, it is up to you to choose a convention for username and password to mysql for each WordPress database instance. This documentation shows only one, localhost, example.

Now configure WordPress to use a mysql database. Open /etc/wordpress/config–localhost.php file and write the following lines:

```
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'wordpress');
define('DB_PASSWORD', 'yourpasswordhere');
define('DB_HOST', 'localhost');
define('WP_CONTENT_DIR', '/usr/share/wordpress/wp–content');
?>
```

Now create this mysql database. Open a temporary file with mysql commands wordpress.sql and write the following lines:

```
CREATE DATABASE wordpress;
CREATE USER 'wordpress'@'localhost'
IDENTIFIED BY 'yourpasswordhere';
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER
ON wordpress.*
TO wordpress@localhost;
FLUSH PRIVILEGES;
```

Execute these commands.

```
cat wordpress.sql | sudo mysql --defaults-extra-file=/etc/mysql/debian.cnf
```

Your new WordPress can now be configured by visiting http://localhost/blog/wp-admin/install.php. (Or http://NAME_OF_YOUR_VIRTUAL_HOST/blog/wp-admin/install.php if your server has no GUI and you are completing WordPress configuration via a web browser running on another computer.) Fill out the Site Title, username, password, and E-mail and click Install WordPress.

Note the generated password (if applicable) and click the login password. Your WordPress is now ready for use.

References

- WordPress.org Codex
- Ubuntu Wiki WordPress

Generated on 2020-05-16 03:53:11