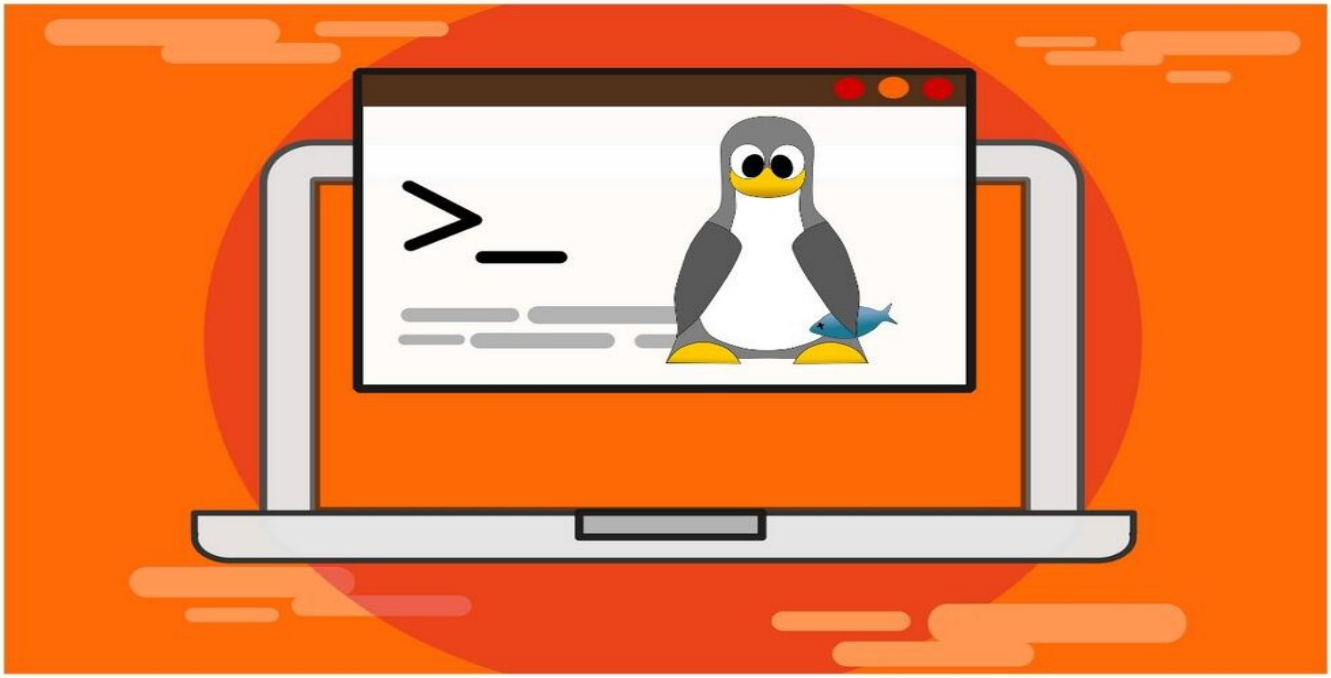
LEARN LINUX

BEGINNER TO EXPERT LEVEL -2020



GO FROM BEGINNER TO ADVANCED WITH LINUX.

GO FROM BEGINN

Contents

Best Linux distros of 2020: for Beginners
- 1. Elementary OS
- 2. Linux Mint
- 4. Ubuntu
- 5.Debian
- Run in the Terminal Commands
  - Shurtcut for terminal

Best Linux Distros of 2020: for Servers and Enterprise
- 4.Red Hat Enterprise Linux
- 5-Oracle OS

Best Linux Distributions for Hacking and Penetration Testing
- 2-BackBox
- 3-CAINE
- Web Browsers in Linux
- Install dual-boot with both Windows and Ubuntu
- Change the Timezone Step by Step
- Working with Filesystem Commands
- Working with Permission
- Working with Linux Networking Commands
- Working with Linux Update-Upgrade Commands
- Working with User Administration
- Working with Disk Usage
- Working with Processes
- Configuring SSH Connection
- Configuring VNC Connection
- Extract .tar.gz Files using Linux Command Line
- Must Know Linux Shortcuts
- Securely Copy Files Using SCP
- Package Managers for Linux
- GUI Tools for Linux System Administrators
- Best Command Line HTTP Clients for Linux
- Best IP Address Management Tools for Linux
- Open Source Log Monitoring and Management Tools
- Backup Utilities for Linux Systems
- Extra:Backup Program for Linux
- Install wget on Linux
- Install and Use dig and nslookup Commands in Linux

Manage Networking with Network Manager in RHEL/CentOS
Set Static IP Address on CentOS/RHEL
File and Disk Encryption Tools for Linux
Cron Jobs for Linux
Run a Command with Time Limit (Timeout) In Linux
Install Apache Tomcat on Debian

Best Linux distros of 2020: for Beginners

Linux is traditionally associated as being an operating system for coders and programmers, but over the years there have been real attempts to make Linux more attractive to general consumers. This is not least due to general consumer dissatisfaction with Windows security issues or even Apple's walled garden.

However, Linux comes in many different forms, known as 'flavors' or 'distros'. This is simply because Linux is so incredibly configurable that different forms tend to be developed for different user base needs or interests.

For example, as mentioned, some have moved toward trying to entice disgruntled Windows users into something more familiar. However, others remain focused on specific environments that may favor programming or scientific applications, or other concerns such as security, resource use, and similar.

Different Linux distros can all work with Linux software and applications, and of course, any cloud-based apps that run through a browser.

1. Elementary OS

Probably the best looking distro in the world.

+Smartly designed and looks great

+Excellent desktop environment

-Not many preinstalled apps

If you’re after a distro that gets you as far away as possible from the image of a nerdy hacker type bashing away at a terminal interface, Elementary OS is what you need. It’s probably the most attractive distro around, with a style similar to that of macOS. This operating system’s superb desktop environment is known as Pantheon, and is based on Gnome.

The latest version of Elementary OS is called Loki, which as well as being that bit prettier and neater than its predecessor Freya, has its own application installer UI called AppCenter. It’s a delightfully simple way to install apps outside the terminal, which is handy as there aren't very many preinstalled.

Elementary OS does, however, come bundled with the Epiphany browser, the Geary email client and a few basic ‘tool’ apps. You may need to add more programs, but this is easy to do using the integrated AppCenter.

2. Linux Mint

A strong option for those new to Linux

+Ideal for those switching from Windows/Mac
+Good media support out of the box
+Impressive amount of customisation options

Linux Mint is a great ‘default’ distro for new Linux users, as it comes with a lot of the software you’ll need when switching from Mac or Windows, such as LibreOffice, the favoured productivity suite of Linux users. It also has better support for proprietary media formats, allowing you to play videos, DVDs and MP3 music files out of the box.

4. Ubuntu

One of the most popular distros for good reasons

+Very accessible for novices

+Security and stability of LTS version

+Lubuntu spin is great for underpowered PCs

Ubuntu is one of the most popular flavors of Linux and along with Mint is strongly recommended for Linux newbies, as it's extremely accessible.

Advertisement

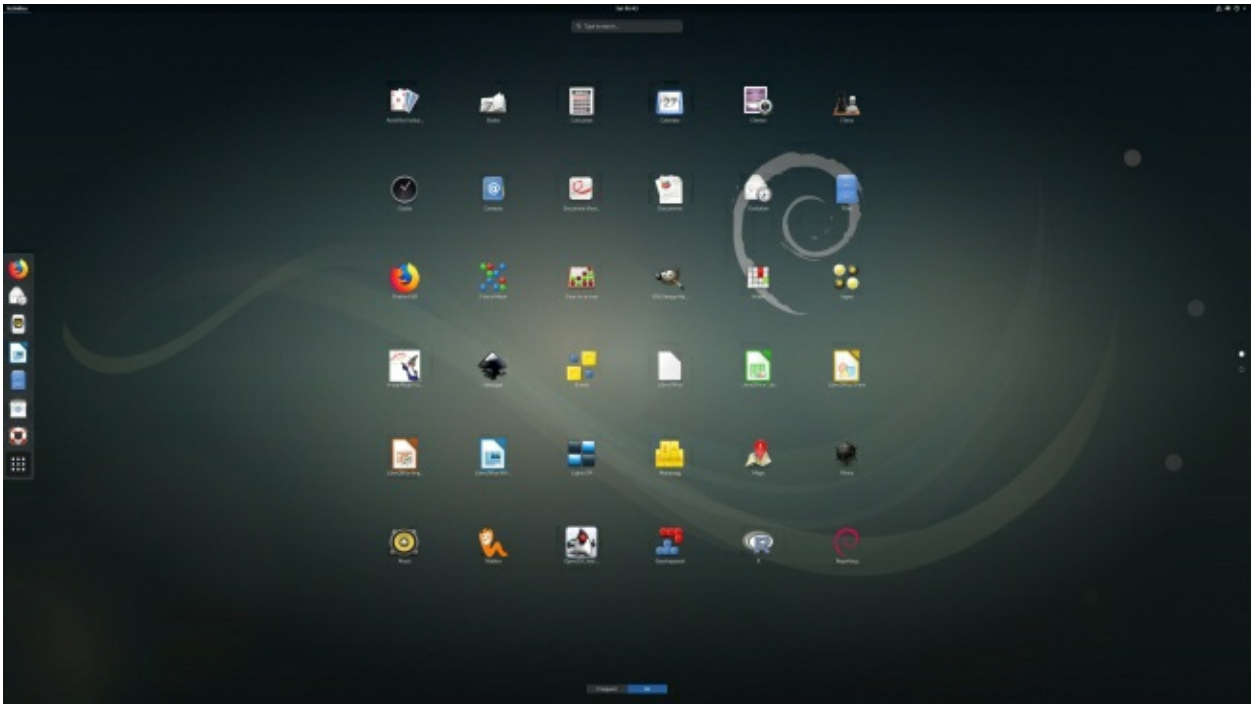
New versions of Ubuntu are released every six months, and every other year the developer Canonical releases an LTS (long term support) version of Ubuntu. These guarantee five years of security and general maintenance updates, so you can carry on using your machine without the hassle of running a full upgrade every few months. Standard releases are supported for one year only.

The current LTS version of Ubuntu uses the Gnome 3 desktop environment, which may be less familiar to Windows and macOS users.

There are variations of Ubuntu which employ different environments such as Lubuntu, which uses a minimal desktop environment based on LXDE and a selection of fast, lightweight applications. This places far less strain on

system resources than the graphic-intensive Unity.

5.Debian



Debian is one of the oldest Linux distributions and is the base for many of the other distributions including Ubuntu and Linux Mint.

It is a community distribution and only ships with free software and free drivers. The Debian repositories offer thousands of applications and there are versions available for a large number of hardware devices.

It isn't the easiest to install and there are various steps you need to go through post installation to get all of your hardware working, though.

Run in the Terminal Commands

An overview of the most important Linux commands

Like most modern operating systems, Linux has two available interfaces for user input. All of the settings that you set via the graphical user interface (GUI) can also be made in the form of **command line directives** via the so-called **shell**.

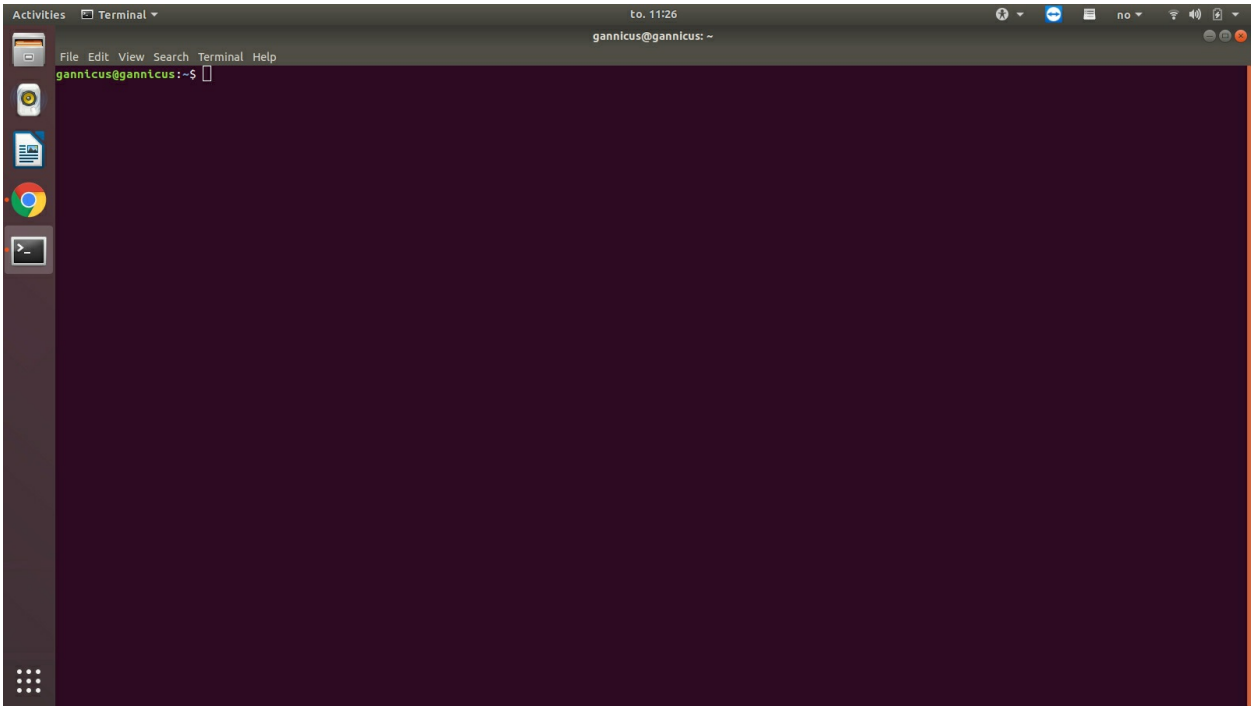
Shell is a program that **functions as an interface between system and user**. It includes a command line interpreter that accepts user input via the

keyboard, evaluates them, starts programs (if necessary), and returns the output in the form of a text entry to the user. In addition, each shell has its own programming language which makes it possible to write shell scripts – for example, to link program calls and facilitate administrative tasks.

Each shell runs in a **terminal**. At the beginning of the computer age, independent devices, or so-called hardcopy terminals (printer or screen plus keyboard), were used. These were replaced on modern computers by **terminal emulators** – programs that provide users with a graphical window for interacting with the shell.

Shurtcut for terminal

As soon as you access the terminal of your operating system, is starts the standard shell (i.e. the *Bourne again shell*, Bash) specified in the settings, and accepts input at the **prompt**.



Best Linux Distros of 2020: for Servers and Enterprise

1- Ubuntu

Built for stability - Ideal for a server

Ubuntu Server brings economic and technical scalability to your datacentre, public or private. Whether you want to deploy an OpenStack cloud, a Kubernetes cluster or a 50,000-node render farm, Ubuntu Server delivers the best value scale-out performance available.

2. CentOS

Offshoot of Enterprise version of Red Hat Linux

Built for stability

Ideal for a server

Not so great for daily desktop usage

CentOS is a community offshoot of the Enterprise version of Red Hat Linux, and its focus is on stability rather than constant updates. Like Red Hat, security and maintenance updates for CentOS are pushed out up to 10 years from the initial release of each version.

CentOS is designed to be super-reliable, which is why it’s a great choice for a server. It's not quite such a good bet for someone looking for a new OS for daily use on their desktop PC or laptop.

3. SUSE

Primarily targeted at devs and sysadmins

-Very polished distro

-Nicely secure

-Can create your own version of the OS

it’s extremely stringent on security protocols.

Advertisement

The operating system is divided into two main distributions: openSUSE Leap and openSUSE Tumbleweed. Leap uses the source code from SUSE Linux Enterprise, which makes it much more stable. New versions are released roughly once a year and are supported for three years, making Leap perfect for business applications.

The OS makes uses of the shiny KDE Plasma desktop, which is perfect if your machine can handle the more advanced graphical features.

One of the more polished Linux distros around, openSUSE is consistently ranked in the top five distros on DistroWatch.com. What’s more, the SUSE Studio Express website allows you to create your own version of openSUSE, complete with tailored preinstalled software packages, desktop and system settings.

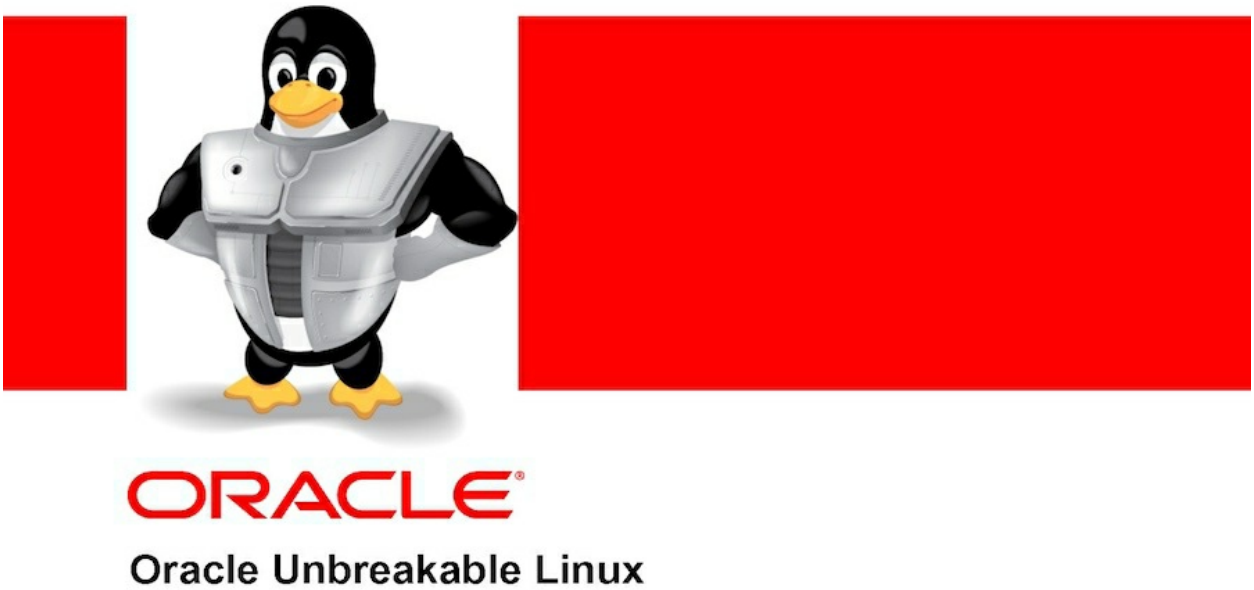
4.Red Hat Enterprise Linux

Red Hat is the market leader of Linux distributions, with an estimated market share of 67%, according to Gartner. RHEL entered many data centers via support deals with hardware and software vendors, an approach emulated by other Linux distributions.

The company employs more developers than other supported Linux vendors. Therefore, Red Hat plays an important role in many open source projects. If your company wants to adopt cutting-edge open source platforms, it makes sense to standardize on Red Hat.

Red Hat offers many products that can be used to provide a complete open source infrastructure, notably Red Hat Enterprise Virtualization and Red Hat Cloud. Red Hat also provides the JBoss middleware platform, which corporate IT teams use to develop applications.

5-Oracle OS



Oracle's Unbreakable Enterprise Kernel started by modifying the open source RHEL software to make a platform for Oracle databases.

Apart from companies that are running Oracle databases, Oracle's distribution hasn't been a huge success. It typically comes into the data center if the database administrator decides which Linux OS to use. Even still, many companies already have a Linux policy and a standard distribution that predates Oracle's open source initiative. This means plenty of Oracle databases still run on SUSE and Red Hat distributions, where it is supported.

Best Linux Distributions for Hacking and Penetration Testing

Best Linux hacking distributions

I have basically listed out various Linux distributions focusing on security. These Linux distros provide various tools that are needed for assessing networking security and other similar tasks. List is in no particular order.

1. Kali Linux

Kali Linux is the most widely known Linux distro for ethical hacking and penetration testing. Kali Linux was developed by Offensive Security taking on the mantle of BackTrack.

Kali Linux is based on Debian. It comes with a large amount of penetration testing tools from various fields of security and forensics. And now it follows the rolling release model, meaning every tool of your collection will always be up-to-date.

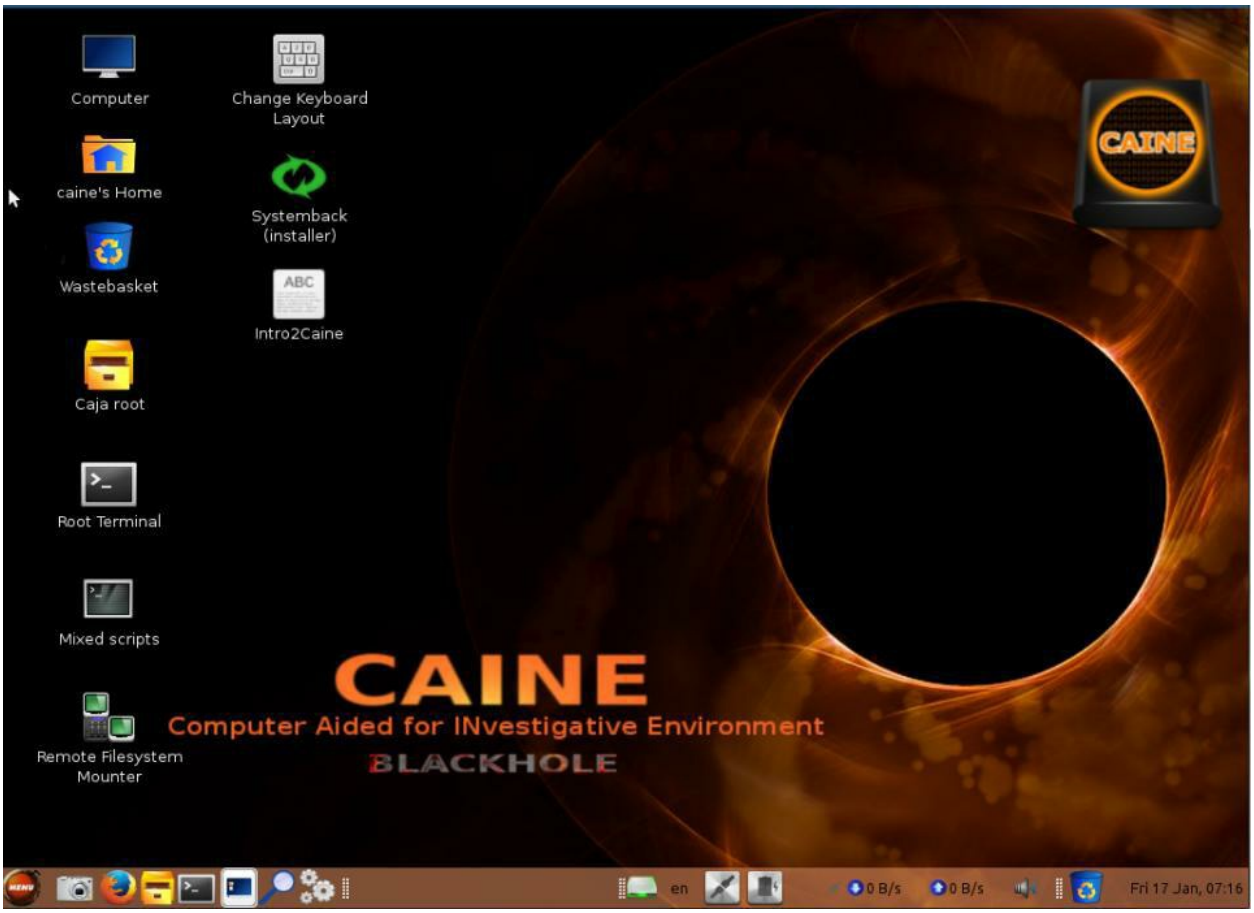
It is the most advanced penetration testing platform out there. It supports a

wide range of devices and hardware platforms. Moreover, Kali Linux provides decent documentation and a large & active community.

2-BackBox

BackBox is a Ubuntu-based distro developed for penetration testing and security assessment purpose. It is one of the best distro in its field.

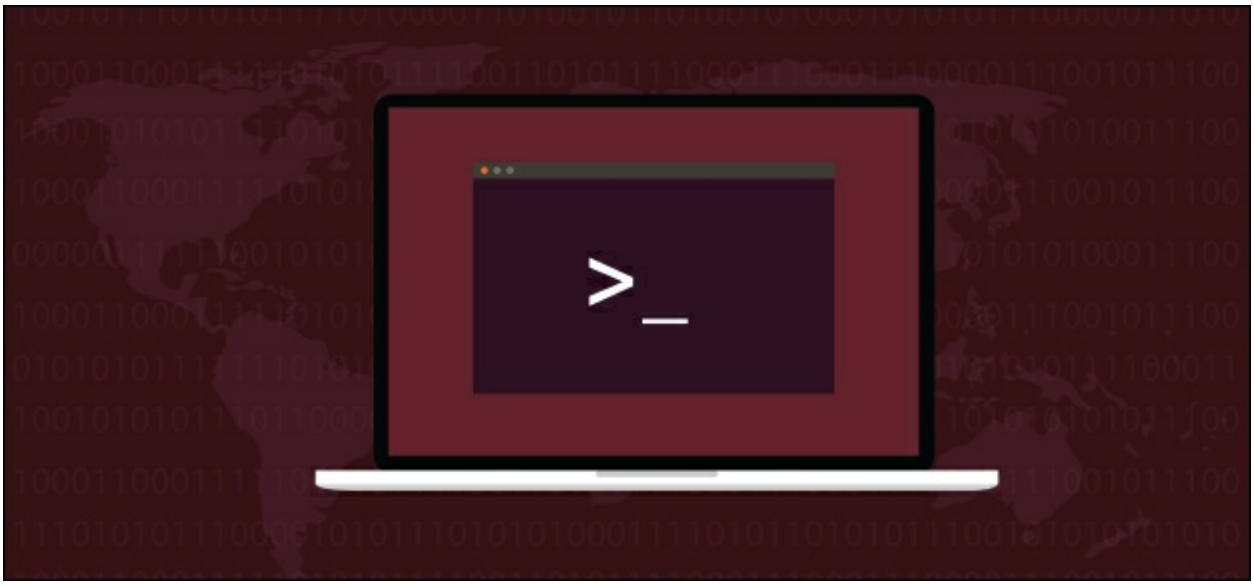
BackBox has its own software repository providing latest stable versions of various system & network analysis toolkits and the best known ethical hacking tools. It is designed with minimalism in mind and uses XFCE desktop environment. It delivers a fast, effective, customizable and complete experience. It also has a very helpful community behind it.



3-CAINE

CAINE stand for Computer Aided Investigative Environment. It is created as a project of Digital Forensics and is completely focused on it.

Web Browsers in Linux



You can find different web browsers for Linux:

- 1. Google Chrome
- 2. Firefox
- 3. Opera
- 4. Vivaldi
- 5. Chromium
- 6. Midori
- 7. Falkon
- 8. Konqueror
- 9. Web (Epiphany)
- 10. Brave

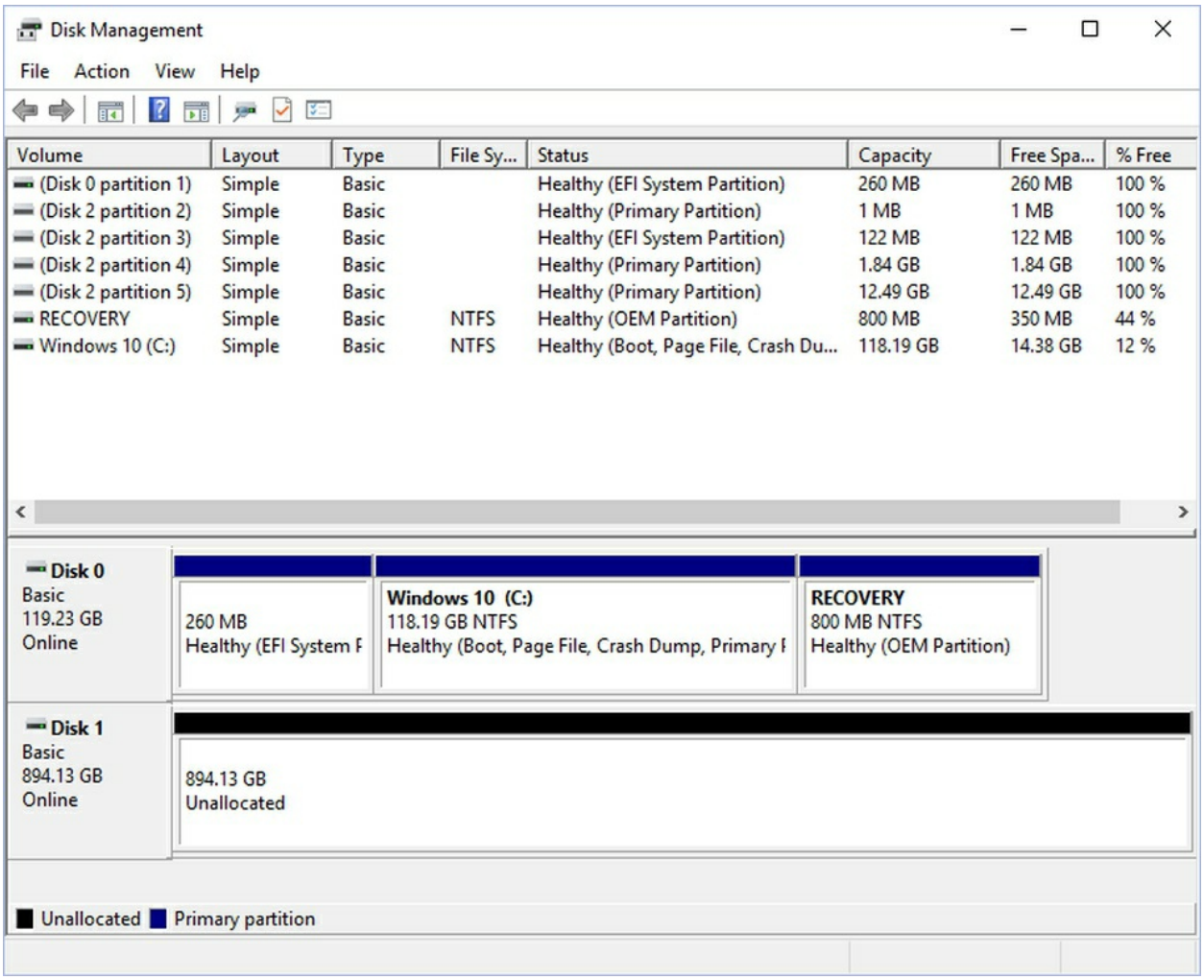
Install dual-boot with both Windows and Ubuntu

PART 1: GET PREPARED

Step 1: Evaluate Your Storage Space

Let's not go into this blind. Put some thought into whether you want to completely wipe out Windows, or dual-boot with both Windows and Ubuntu. The installation you're about to do will give you full control to completely erase your hard drive, or be very specific about partitions and where to put Ubuntu. If you have an extra SSD or hard drive installed and want to dedicate that to Ubuntu, things will be more straightforward. (Don't worry, you'll get to choose Windows or Ubuntu when your system boots up.)

If you're running on a single drive with Windows and are almost out of space, you may want to consider adding that extra drive! Ubuntu doesn't take nearly as much space as Windows, but assuming you enjoy the experience and want to use it regularly, you'll appreciate thinking about this ahead of time.



Windows 10 disk management tool. In my example I've devoted a spare drive to Ubuntu. It's not formatted or partitioned yet.

MICROSOFT

Because of the variables here, I can't give you specific instructions for each one. But to check out your current drive situation, click the Windows key (Start button) and type "disk management." This will show you the number of drives you have, and how they're partitioned.

Step 2: Create A Live USB Version Of Ubuntu

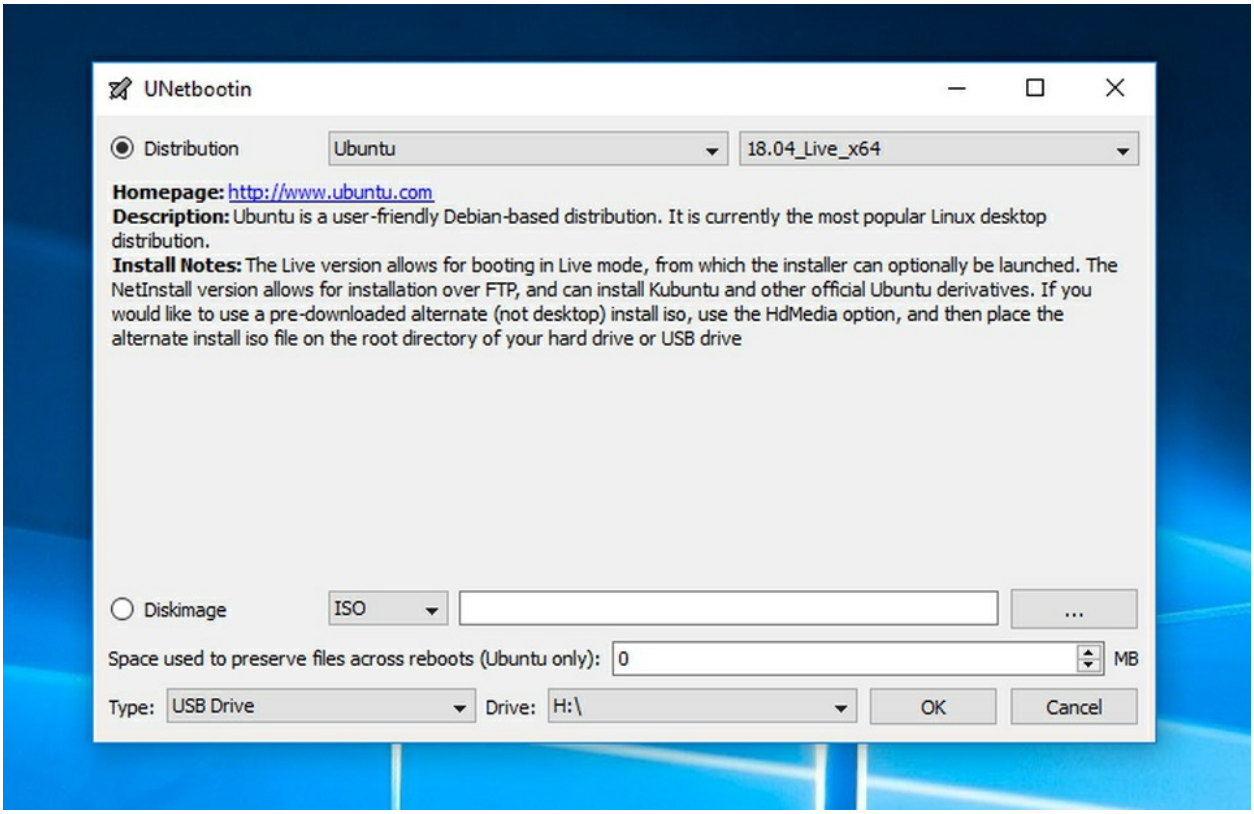
UNetBootin is a lightweight piece of software that works on Windows, Linux and MacOS. It not only creates a bootable USB stick for you, but it also downloads dozens of different Linux distributions automatically.

UNetBootin website

GITHUB

Insert your USB stick of choice (make sure there's nothing on it you want to keep as UNetBootin will format it).

Then, visit unetbootin.github.io/ and download the Windows version. Save it to a location of your choice, or just select "Run" after the download has finished.



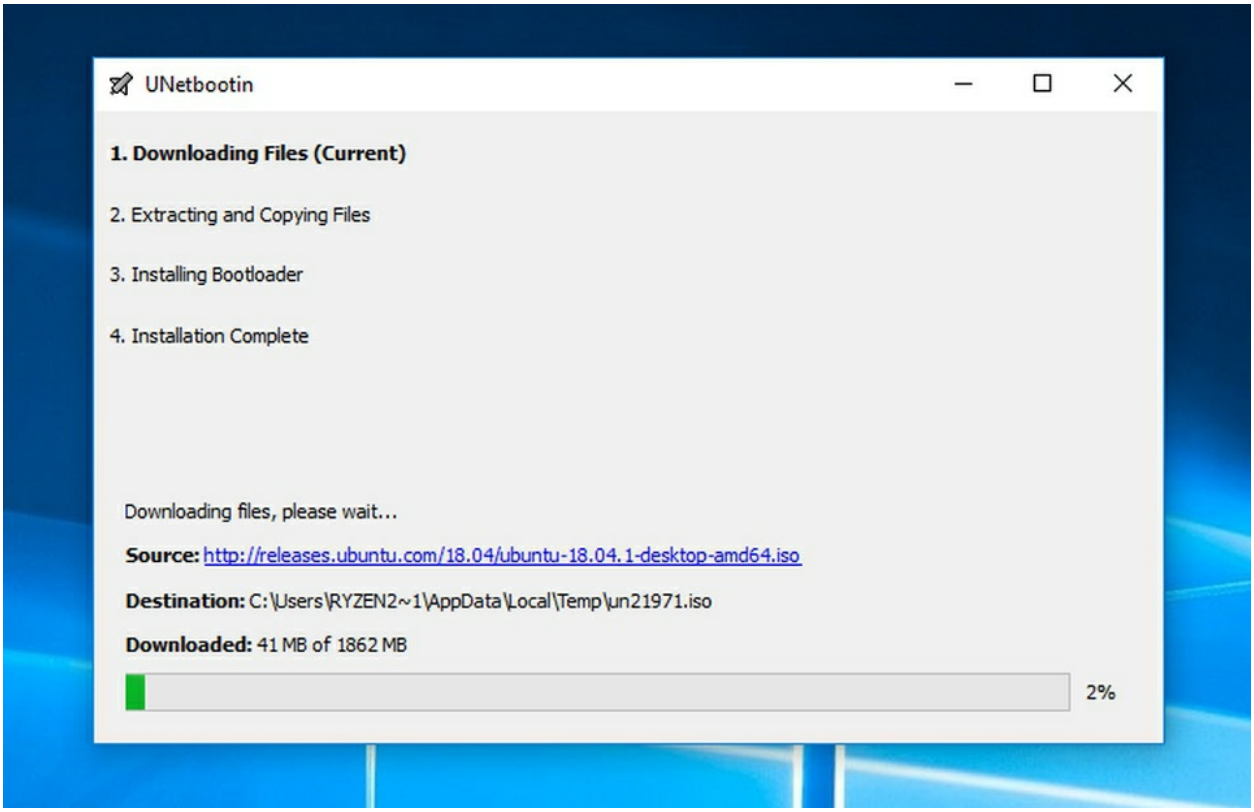
UNetbootin Main Screen

GITHUB

Once it launches, select "Ubuntu" from the left dropdown menu. Then you want to select the version. Choose "Ubuntu 18.04 Live x64". This is the latest

"LTS" release. LTS stands for "Long Term Support" meaning that Canonical, the company behind Ubuntu, will support it with regular maintenance and security updates for five years after its release. The "Live" part means that you can try it right from the USB stick without installing anything. And "x64" means that it's built for modern 64-bit operating systems.

Now, just make sure the right USB stick is selected at the bottom and click OK.



Step 2: Prepare Your PC To Boot From USB

Things may get a little tricky here, but rest assured the answer is out there if you get stuck. You've created a bootable USB stick, but you have to your PC to boot from it first. To do that, you may need to go into your system's BIOS screen, which is accessible only when you first boot your system. (Some PCs may automatically boot to a USB stick if it's inserted and bootable.)

For the majority of systems, you'll probably want to hit the "DEL" key right when your PC boots. Sometimes it may be F12, F11, F10 or F2. If in doubt, consult an online manual for your motherboard.

BIOS Screen for my MSI motherboard

MSI

Every BIOS screen is different, but I'm using an MSI motherboard so I'm showing that example above. Ordinarily you'll need to find a category called "Boot" where you'll see the order your PC looks for devices to boot from. It's probably set to look at a CD/DVD drive first, and the drive where Windows is installed next. Click on the top or first option and change it to USB. This will probably look something like "UEFI USB Hard Disk" and should have the name of the manufacturer next to it.

While you're here, I recommend disabling "Secure Boot." This could save you headaches down the road.

Made it this far? Awesome! Now hit "F10" and select "Save settings and

reboot." (Again the wording may be slightly different on your PC).

Get ready, it's time install Ubuntu!

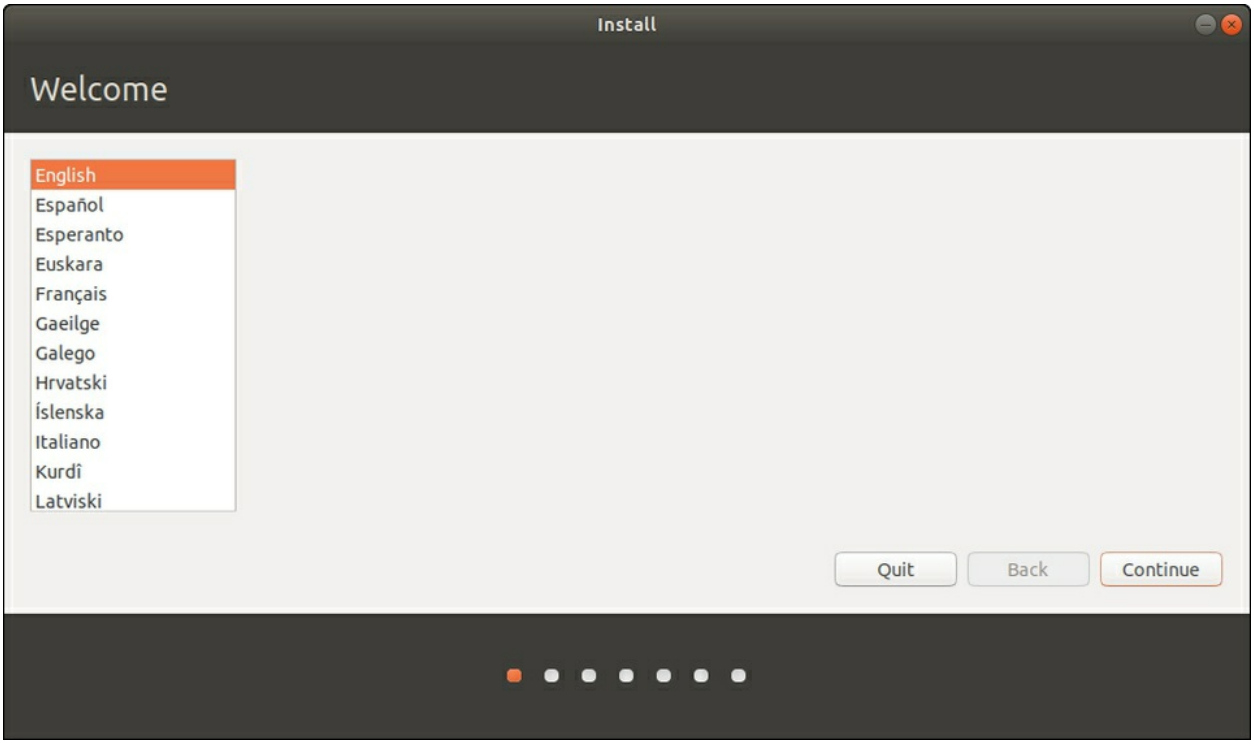
PART 2: UBUNTU TEST DRIVE AND INSTALLATION

You have a bootable Live USB, and your PC should boot from it. When you boot up your system again you should see a text menu with the options to try or install Ubuntu. For now, let's take it for a test drive. It's optional, but it will get you familiar with the layout and user interface, see if your WiFi adapter is detected and check if things like resolution and graphics cards are working properly.

Step 1: Starting The Installation

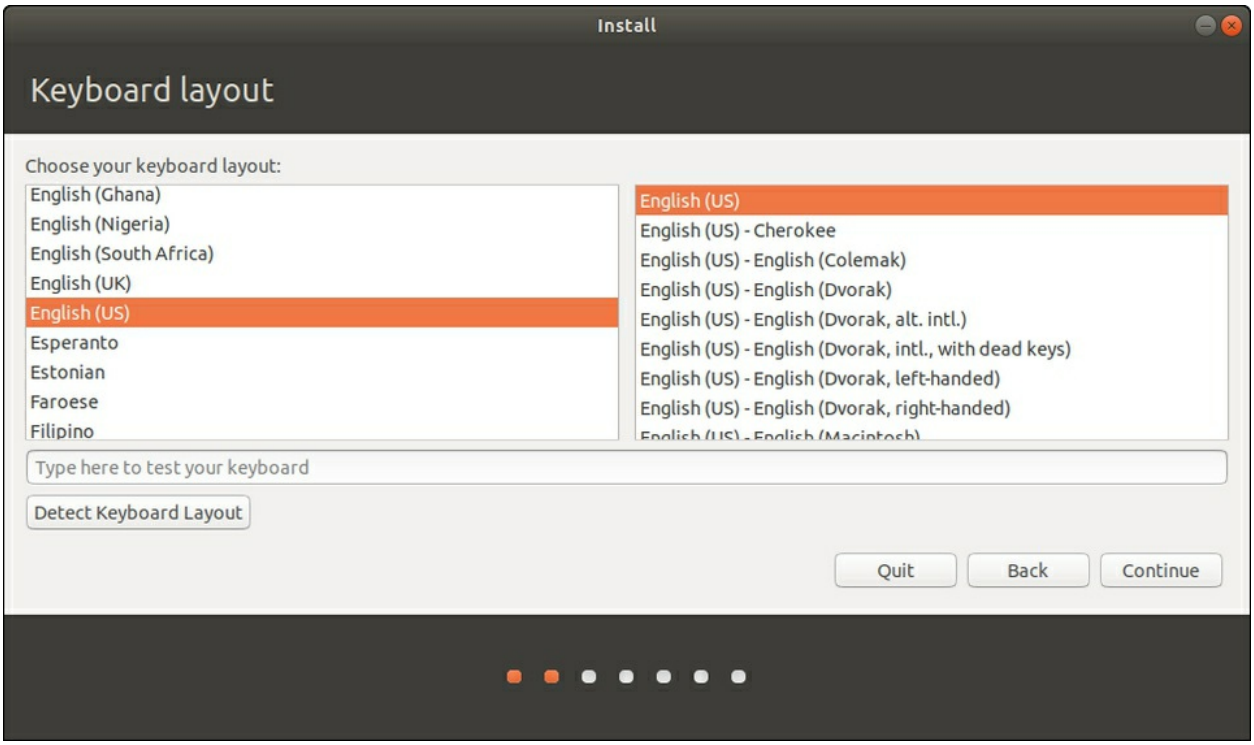
Ready to roll? If you're just booting up, select "Install Ubuntu." If you're taking a test drive, click the top-most icon the dock that says "Install" (it may be a shortcut on the desktop as well).

Installing Linux is so much easier than it used to be, so most of this will be straightforward. Just in case, I've tried to capture and represent each basic step.



Select Language

CANONICAL

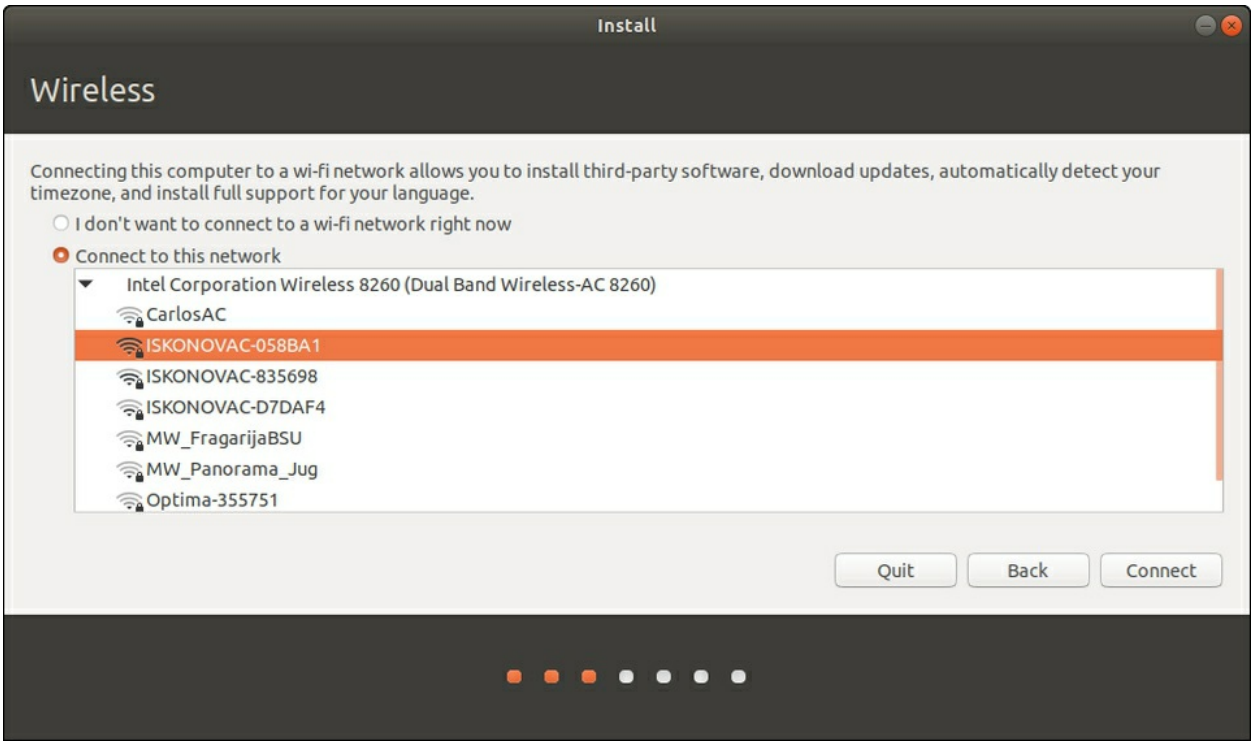


Select keyboard layout (you can add additional layouts at any time)

CANONICAL

In the opening screens you'll choose your language and a keyboard layout, which you can change or add to at any time.

Step 2: Get Connected



Get connected!

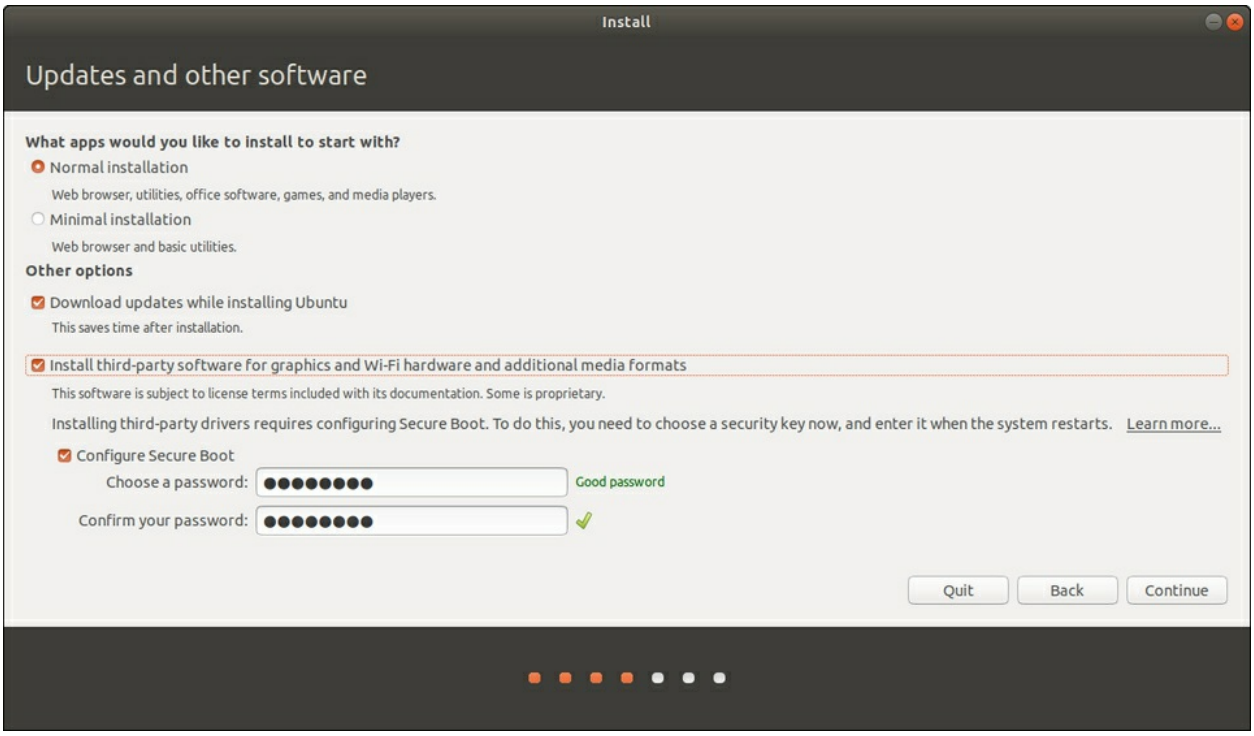
CANONICAL

Then you'll connect to your wired or wireless network. A wired network (ethernet cable) will be automatically detected and initialized, but you'll need to choose your Wireless network name and enter a password. Getting connected now means you can download security and feature updates while Ubuntu is installing.

At its core, Linux supports a large number of wireless adapters and they're normally detected without issue. If yours isn't, you have many options after the installation, including using a Windows driver! It's a bit outside the scope of this guide, though. If you need explore www.AskUbuntu.com.

Step 3: Updates & Other Software

So far so good? Excellent! Now we're going to add a variable or two to the installation process. First, select "Normal" at the top for the fully featured experience. Most users coming from Windows aren't going to want a minimal installation.



Additional installation options

CANONICAL

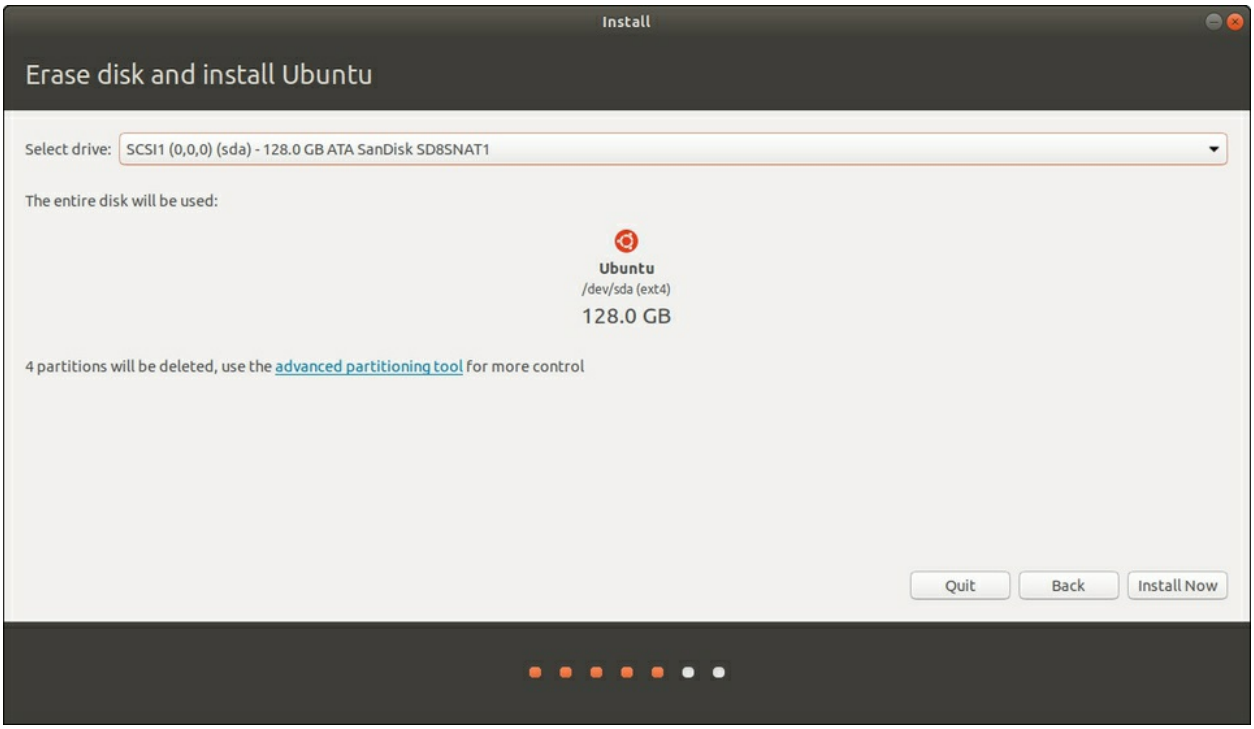
If you have an internet connection check the box to download updates while installing. And definitely check the box that says "install 3rd-party software." This will open up options to activate optional Nvidia or AMD drivers, get you loaded up on media codecs for playing a wider range of music and video formats, and in general supply more hardware support out of the box.

Some PCs will have a Secure Boot option here. If so, Ubuntu will ask you for one-time password that you'll enter here, and then again when your system reboots. Don't worry, the OS will tell you when it's going to happen and instruct you what to press. It'll be during your first post-installation boot. As mentioned earlier, I do recommend disabling Secure Boot altogether in your BIOS.

Step 4: Partition Magic

Now the part that makes people nervous: where to install Ubuntu. Depending on your system's configuration, you'll be presented with a couple core

options. Do you want to install Ubuntu alongside Windows, or erase a disk entirely for Ubuntu? (Remember that you if you choose to dual-boot Windows on a single disk, you can choose which OS to load during boot.)



Want to erase a disk and have Ubuntu do the partitioning for you?

CANONICAL

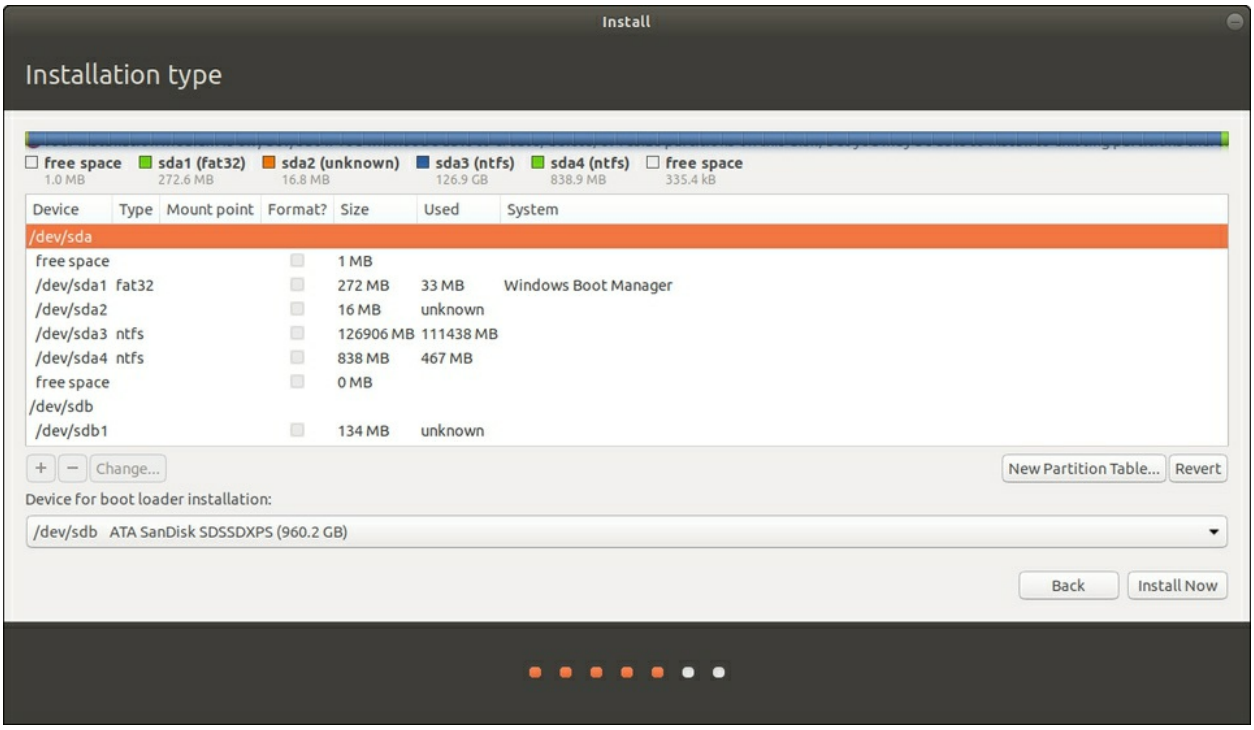
For erasing a disk, you'll get to choose which disk in the following screen. If you have that spare drive installed, just choose that and let Ubuntu do all the heavy lifting and auto-partitioning.

Now, both the above options are equivalent to pressing the easy button. Ubuntu will automatically partition your drive. If you're opting for either of the above, I'll see at Step 5!

"Something Else" is the exact opposite of easy.

"Something Else" means you don't want to install Ubuntu alongside Windows, and you don't want to erase that disk either. It means you have full control over your hard drive(s) here. You can delete your Windows install, resize partitions, erase everything on all disks. So proceed with caution. On the plus side all the changes you make won't be executed until you click Install. You can always go back or start over.

The example I'm going to use is a system with two drives. One has Windows on it, and I want to leave it untouched. The other is blank.

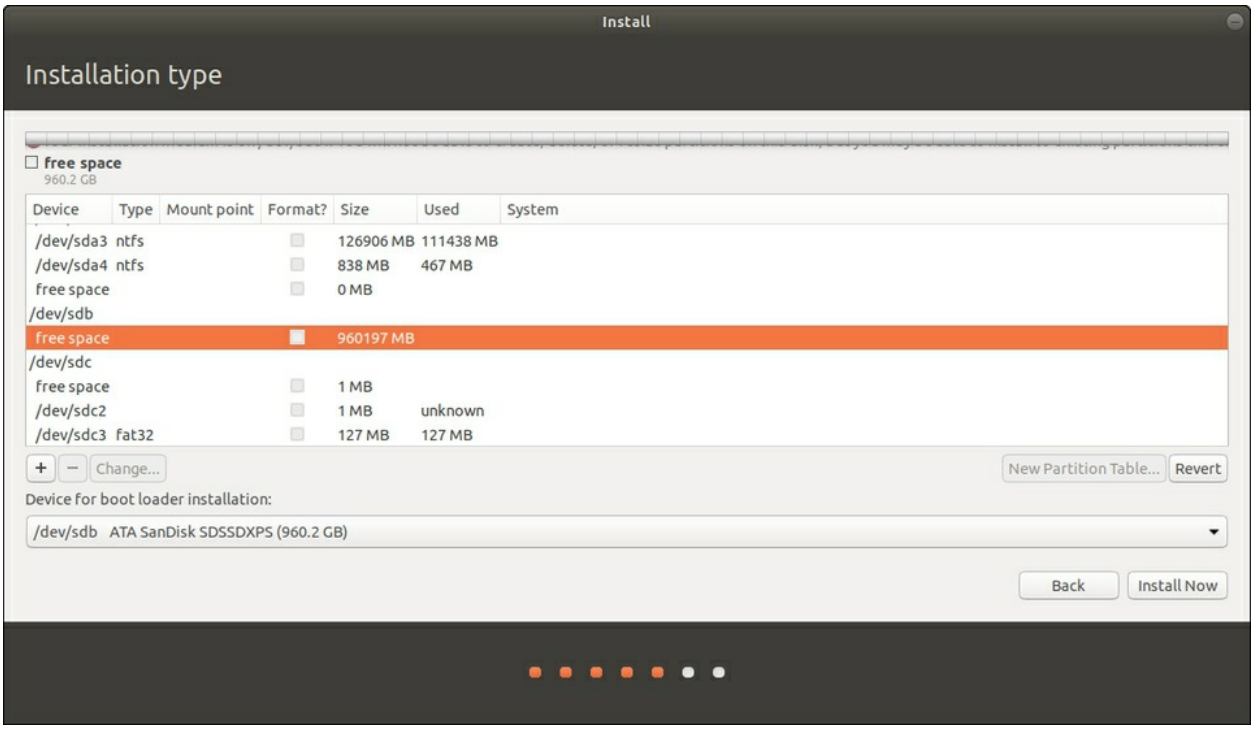


Partition Manager during Ubuntu setup. Be careful here.

CANONICAL

The above example might make your eyes water, and I felt the same way the first few times. Are you SURE you don't want to just erase your disk and avoid all this? Just kidding! So what do all these letters and numbers mean? Time for a. . .

Knowing that, you "Something Else" adventurers will need to add about 4 partitions to your extra drive. I'm going to take you through it step-by-step.

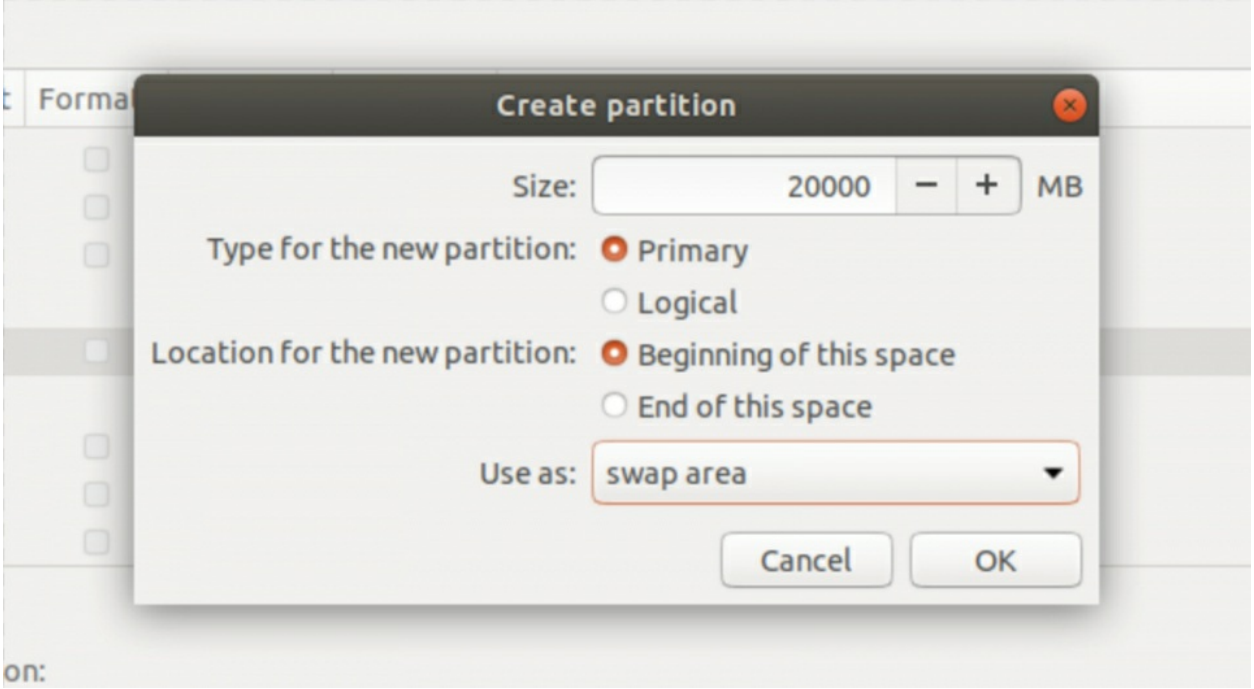


Identify the drive you want to install Ubuntu to

CANONICAL

First, identify the drive you want to install Ubuntu to. I know that it's "sdb" because it lists 1TB of free space and there are no partitions. So, highlight yours (remember it will be the one that looks like "/dev/sdb"and then click "New Partition Table."

Now you will need to create 3 partitions, beginning with a "Swap Area."

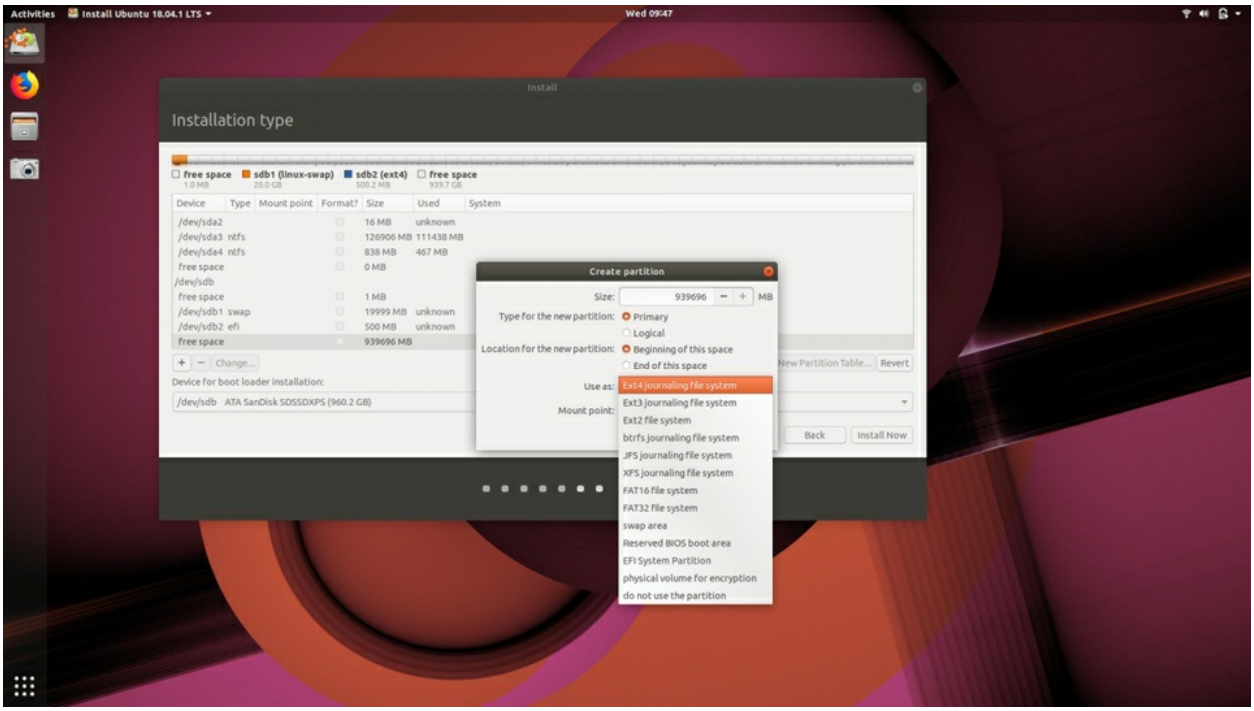


Create a "Swap Area" partition

CANONICAL

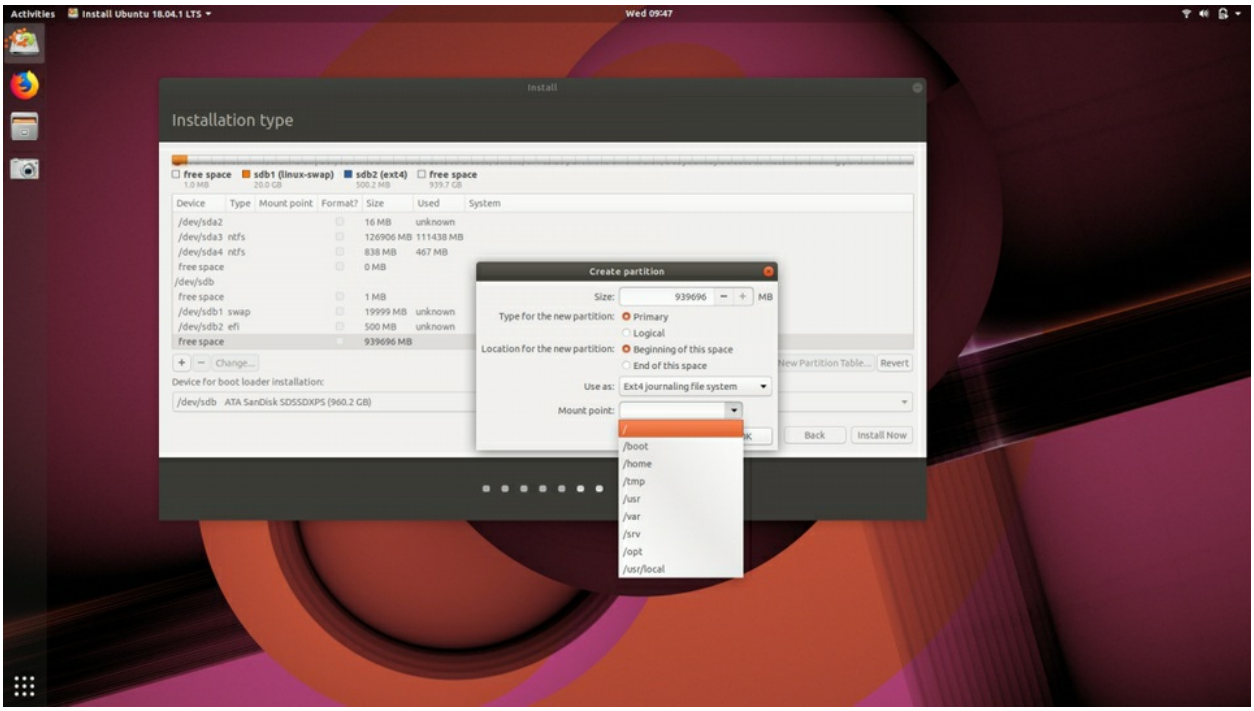
Swap is a small space on the drive that is used like system memory (RAM). As a rule of thumb, you'll want it to be slightly larger than the amount of RAM you have in your PC. I have 16GB of RAM, so I'm creating a 20GB (or 20000MB) swap partition.

Sidebar: You may be required to create an "EFI Partition" as well, and if so Ubuntu will tell you. The EFI partition is where the boot loader is installed and tells your PC which operating systems can be booted. Sometimes you can install it to your Windows drive, and you'll notice there's a drop-down menu to choose where that happens. If you're asked to create one, just follow the same procedure as all other partitions. Make it 250MB.



Create a Root partition with the EXT4 filesystem.

CANONICAL



Then select "/" as the mount point.

CANONICAL

Next let's create our "root" partition using the Ext4 filesystem. (Fun fact: the

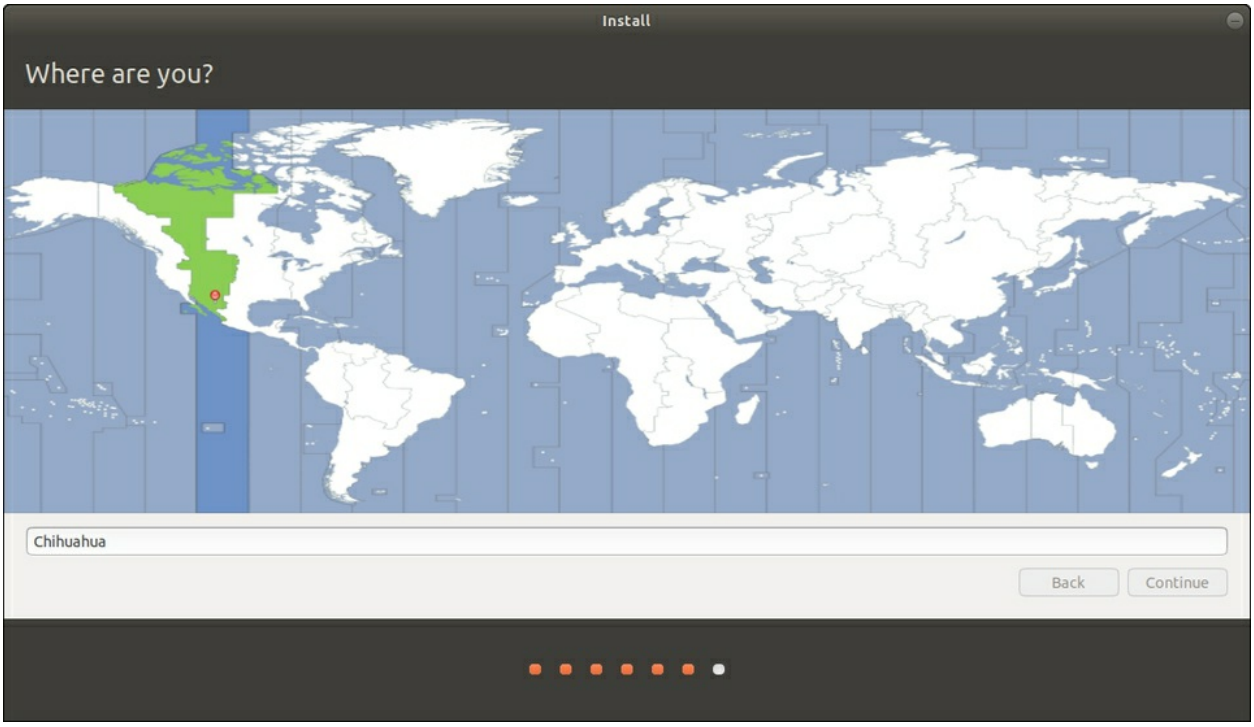
Ext4 filesystem supports volumes up to 1,152,921,504 GB!) The root partition is the top-most directory of the drive and Linux won't work without it.

For its size, you can use the remaining space on your drive which should be automatically inserted. Some will argue that a "/home" partition should also be created (think of /home as the account directory in a Windows installation like "C:/Users/Jason." But Ubuntu will automatically create a "home" folder for your documents, photos, videos and other media and files. It's very easy to back up or transfer to another Ubuntu installation, so I choose to use only the Swap and Root partitions. (As a relative beginner I'm happy to have my mind changed!)

You've told Ubuntu setup how to partition your drive, and when you click "Install" those instructions will get executed. First you'll get a confirmation of the changes. If you're happy with them, make it so!

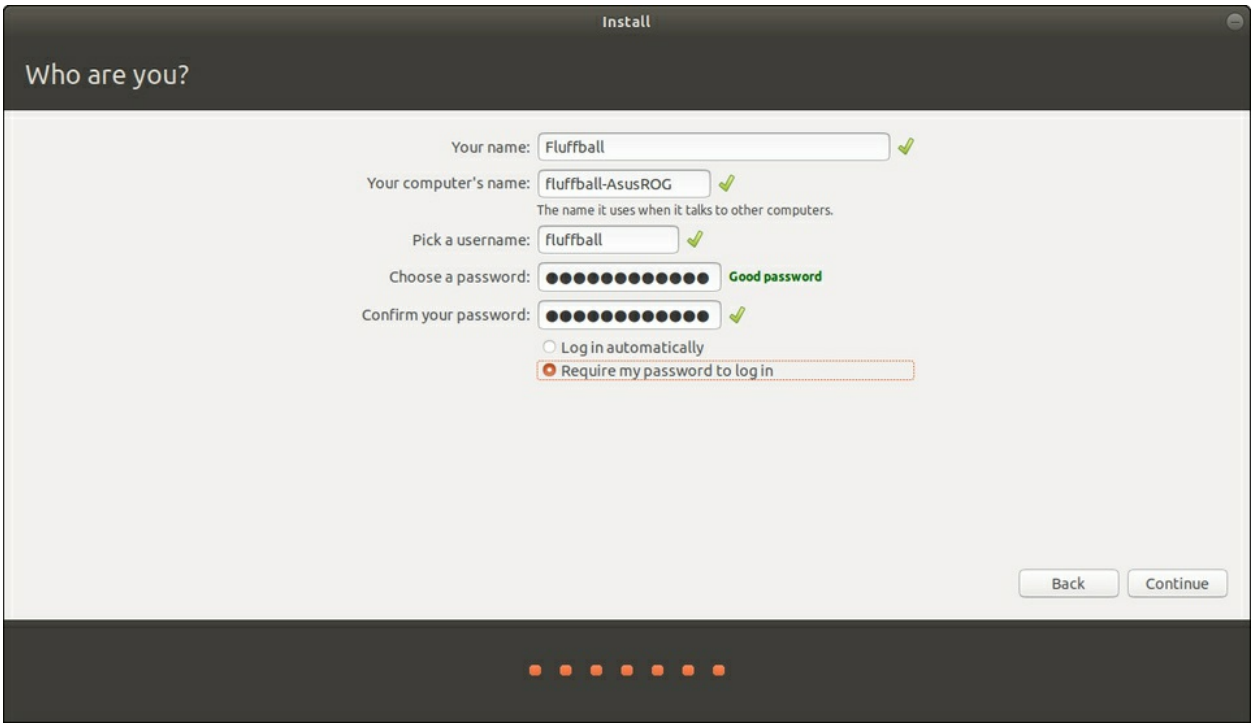
Step 5: Your Region & Account Info

The hard part is over, and I hope it went smoothly for you. All that remains is choosing your region and creating a username and strong password. Because I'm lazy, I usually choose to log in automatically.



Choose your location!

CANONICAL



Create a username and a strong password. Optionally, choose to log in automatically.

CANONICAL

Step 6: Grab A (Small) Cup Of Coffee

Now Ubuntu will start installing and downloading any additional packages if you chose to grab updates. But wait, why a small cup of coffee? Because it's fast. Really fast. On my Dell XPS 13 with a USB 3.0 stick and an NVMe drive, it took less than 4 minutes. Even on older hardware it's really snappy.

Ubuntu installation

CANONICAL

You should be prompted to remove your installation media and reboot. If you had to do the secure boot option, you'll be taken to that screen and your password will be requested.

For me, the installation process was just a first small step in a longer and much more exciting journey. Hopefully yours is headache free, productive, and fun! Explore the Snap store to install popular apps like Spotify, Skype, Discord and Telegram in a single click, or check out the wealth of open source software available.

Ready to start gaming? Check out my beginner's guide to updating your graphics drivers and playing Windows-only games on Steam for Linux!

Open Source Cloud Storage Software for Linux

You will find different Open Source Cloud storage and synchronization softwares here:

- 1. OwnCloud
- 2. Nextcloud
- 3. Seafile
- 4. Pydio
- 5. Syncany
- 6. GlusterFS
- 7. Yandex.Disk
- 8. XigmaNAS

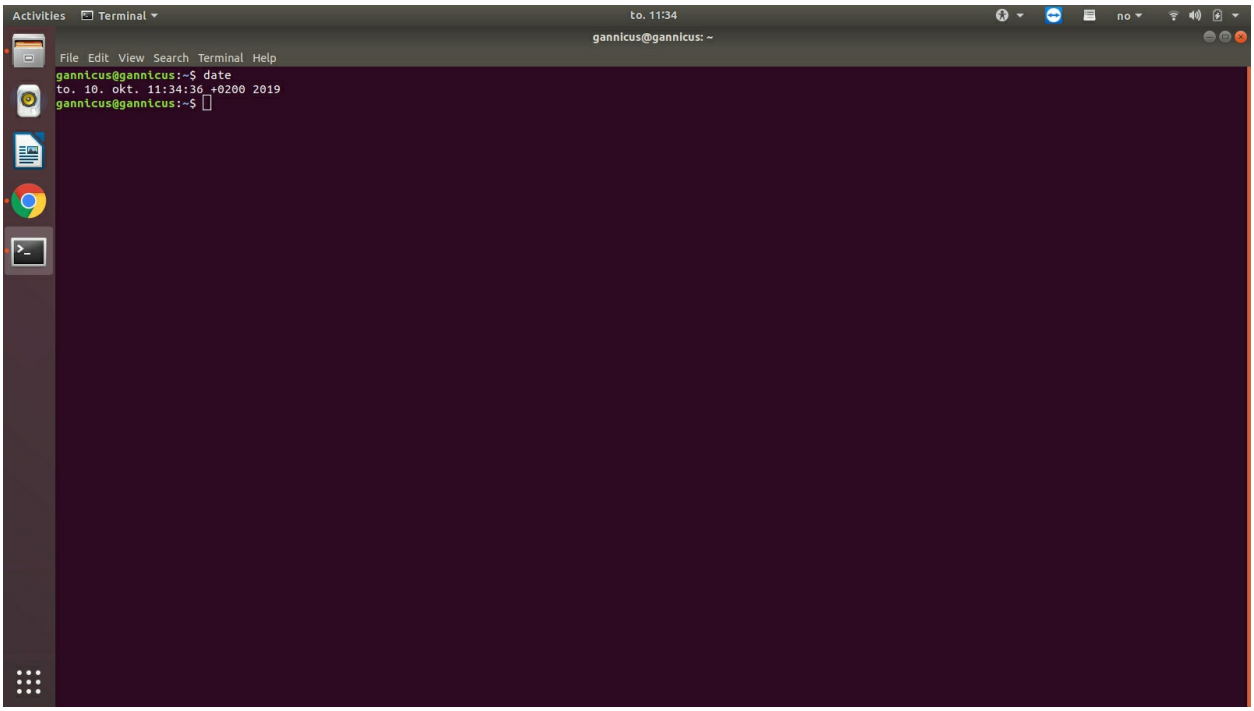
Change the Timezone Step by Step

1. **Open Terminal.**

2-Check your current time zone.

Type date into Terminal and press ↵ Enter. The terminal will display the date in Weekday Month Day Time TimeZone Year format.

- For example, you might see something like **to. 10. okt. 11:34:36 +0200 2019**

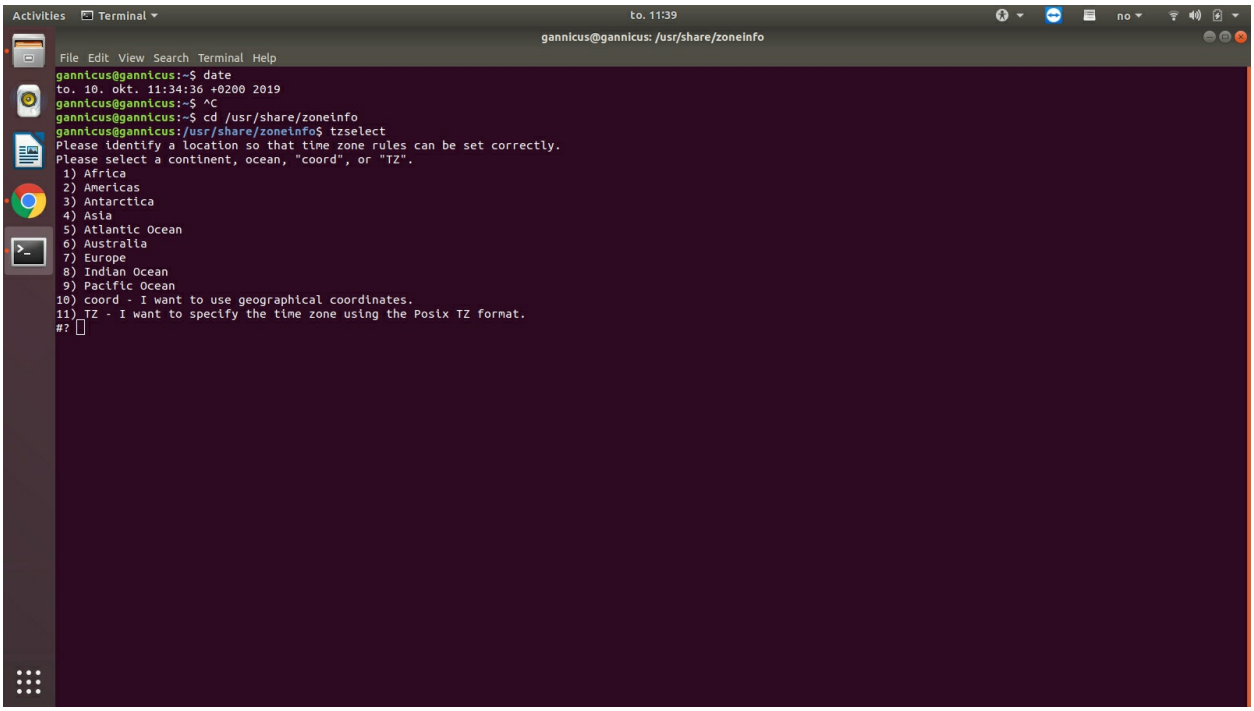


3 -Check available time zones.

Type in **cd /usr/share/zoneinfo** and press ↵ Enter,

then type in tzselect and press ↵ Enter to display a list of locations.

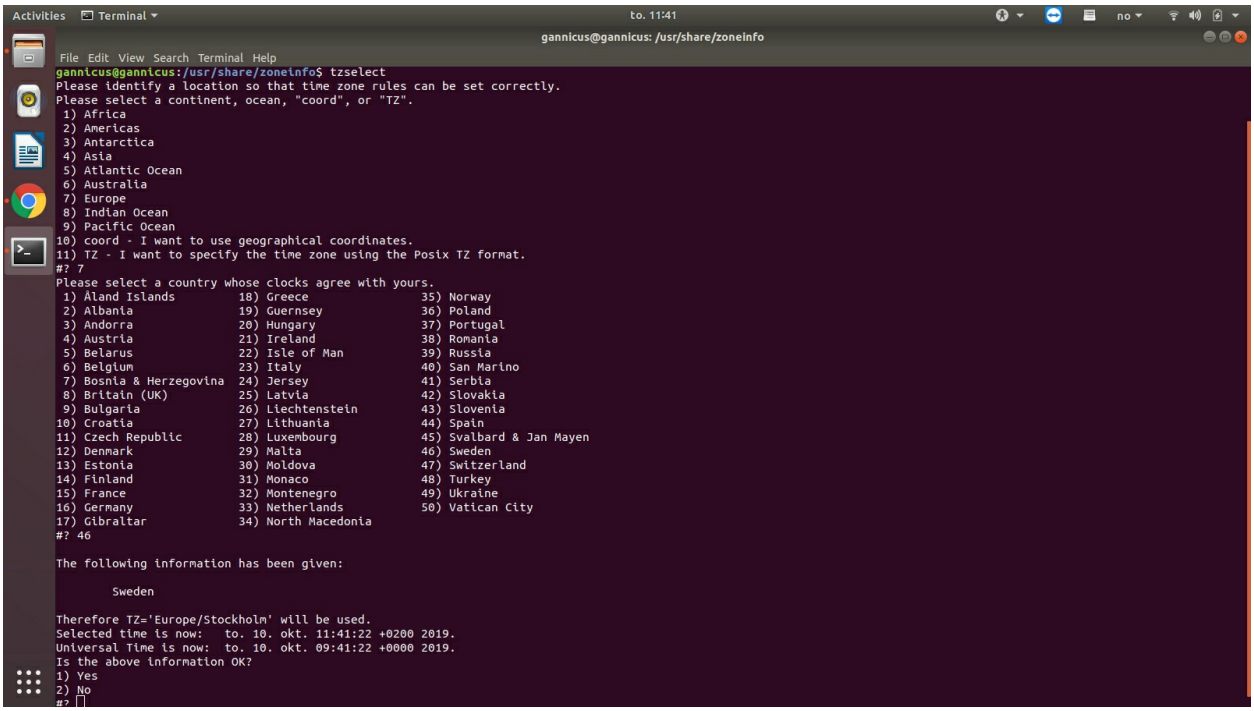
- The /usr/share/zoneinfo directory may vary depending on your Linux distribution.



4-Select a continent or ocean.

Type in a selection number that pertains to your general location, then press ↵ Enter.

5-Select a country. Type in a selection number from the on-screen list and press ↵ Enter.



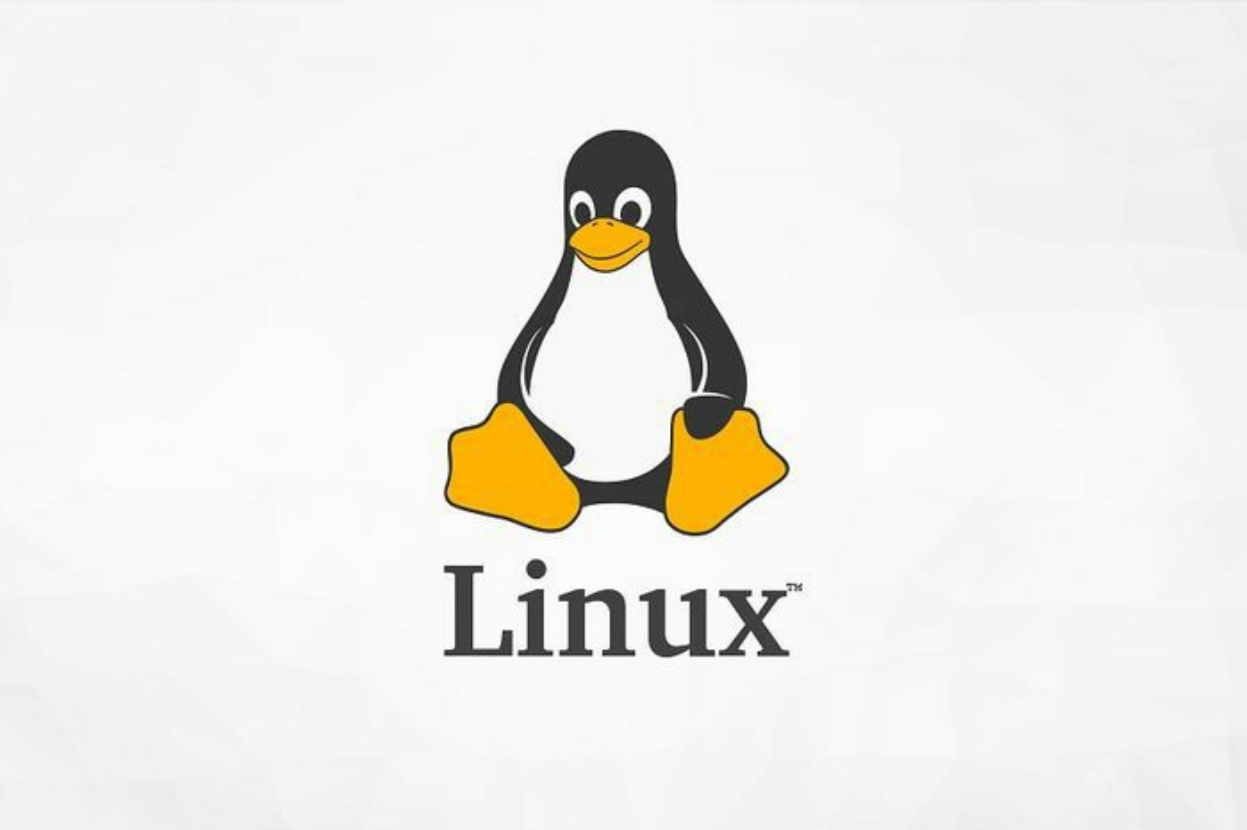
6-Select a time zone and confirm the local time..

Type in a selection number to select the preferred time zone region and press ↵ Enter.

- If your city is not listed in the time zone list, pick a different city from your same time zone.

7-Verify that your time zone has been set. Run the **date** command again and check that the time zone corresponds to the one you just changed to. If you see you new time zone listed, you have successfully changed your computer's time zone.

Working with Filesystem Commands



01 Which Folder Are You In?

When you open a terminal window, the first thing you need to know is where you are in the file system.

Think of this like the "you are here" marker that you find on maps within shopping malls.

To find out which folder you are in, you can use the following command:

```
pwd
```

The results returned by **pwd** may differ depending on whether you are using the shell version of **pwd** or the one installed in your **/usr/bin** directory.

In general, it will print something along the lines of **/home/username**.

02 What Files and Folders Are Under the Current Directory?

Now that you know which folder you are in, you can see which files and folders are under the current directory by using the **ls** command.

```
ls
```

On its own, the **ls** command will list all the files and folders in the directory except for those beginning with a period (.).

To see all the files including hidden files (those starting with a period), you can use the following switch:

```
ls -a
```

Some commands create backups of files which begin with the tilde metacharacter (~).

If you don't want to see the backups when listing the files in a folder, use the following switch:

```
ls -B
```

The most common use of the **ls** command is as follows:

```
ls -lt
```

This provides a long listing sorted by modification time, with the newest first.

Other sort options include by extension, size, and version:

```
ls -lUls -lXls -lv
```

The long listing format gives you the following information:

- Permissions
- Number of inodes for the file (see hard links)
- Owner
- Primary group
- File size
- Last access time

- File/folder/link name

03 How to Navigate to Other Folders

To move around the file system you can use the **cd** command.

The Linux file system is a tree structure. The top of the tree is denoted by a slash (/).

Under the root directory, you will find some or all of the following folders.

- /
- bin
- boot
- cdrom
- dev
- etc
- home
- lib
- lib64
- lost+found
- media
- mnt
- opt
- proc
- root
- run
- sbin
- srv
- sys
- tmp
- var
- usr

The bin folder contains commands that can be run by any user such as the **cd** command, **ls**, **mkdir**, etc.

The sbin contains system binaries.

The usr folder stands for unix system resources and also contains a bin and sbin folder. The /usr/bin folder has an extended set of commands which users can run. Similarly, the /usr/sbin folder contains an extended set of system commands.

The boot folder contains everything required by the boot process.

The cdrom folder is self-explanatory.

The dev folder contains details about all the devices on the system.

The etc folder is generally where all the system configuration files are stored.

The home folder is generally where all the user folders are stored and for the average user, is the only area they should be concerned about.

The lib and lib64 folders contain all the kernel and shared libraries.

The lost+found folder will contain files which no longer have a name which have been found by the fsck command.

The media folder is where mounted media such as USB drives are located.

The mnt folder is also used to mount temporary storage such as USB drives, other file systems, ISO images, etc.

The opt folder is used by some software packages as a place to store the binaries. Other packages use /usr/local.

The proc folder is a system folder used by the kernel. You don't really need to worry about this folder too much.

The root folder is the home directory for the root user.

The run folder is a system folder for storing system runtime information.

The srv folder is where you would keep things like web folders, mysql databases, and subversion repositories, etc.

The sys folder contains a folder structure to provide system information.

The tmp folder is a temporary folder.

The var folder contains a whole wealth of stuff specific to the system, including game data, dynamic libraries, log files, process IDs, messages, and cached application data.

To navigate to a particular folder use the **cd** command as follows:

```
cd /home/username/Documents
```

04 How to Navigate Back to the Home Folder

You can get back to the home folder from anywhere else in the system using the **cd ~** command.

```
cd ~
```

05 How to Create a New Folder

If you want to create a new folder you can use the **mkdir** command:

```
mkdir foldername
```

06 How to Create Files

Linux provides an incredible number of ways for creating new files.

To create an empty file you can use the following **touch** command:

```
touch filename
```

You can also create a file using the **cat** command:

```
cat > filename
```

You can now enter text on the command line and save it to the file using **Ctrl + D**.

A better way of creating files is to use the nano editor. This lets you add lines of text, cut and paste, search and replace text, and save the file in various formats.

07 How to Rename and Move Files Around the File System

The are a number of ways to rename files.

The simplest way to rename a file is to use the **mv** command.

```
mv oldfilename newfilename
```

You can use the mv command to move a file from one folder to another as well.

```
mv /path/of/original/file /path/of/target/folder
```

If you want to rename a lot of files which match a similar pattern you can use the **rename** command.

```
rename expression replacement filename(s)
```

For example:

```
rename "gary" "tom" *
```

This will replace all files in the folder with *gary* in it with *tom*. So a file called *garycv* will become *tomcv*.

Note that the rename command doesn't work on all systems. The mv

command is safer.

08 How to Copy Files

To copy a file using Linux you can use the **cp** command as follows.

```
cp filename filename2
```

The above command will copy *filename1* and call it *filename2*.

You can use the copy command to copy files from one folder to another.

For example:

```
cp /home/username/Documents/userdoc1 /home/username/Documents/UserDocs
```

The above command will copy the file *userdoc1* from */home/username/Documents to /home/username/Documents/UserDocs*

09 How to Delete Files and Folders

You can delete files and folders using the **rm** command:

```
rm filename
```

If you want to remove a folder you need to use the following switch:

```
rm -R foldername
```

The above command removes a folder and its contents including sub-folders.

10 What Are Symbolic Links and Hard Links?

A "symbolic link" is a file that points to another file. A desktop shortcut is basically a symbolic link.

You might, for example, have the following file on your system:

/home/username/document/accounts/useraccounts.doc

Maybe you want to be able to access that document from the home/username folder.

You can create a symbolic link using the following command:

```
ln -s /home/username/documents/accounts/useraccounts.doc /home/username/useraccounts.doc
```

You can edit the *useraccounts.doc* file from both places but when you edit the symbolic link, you are actually editing the file in the

/home/username/documents/accounts folder.

A symbolic link can be created on one filesystem and point to a file on another file system. Essentially, a symbolic link really just creates a file which has a pointer to the other file or folder.

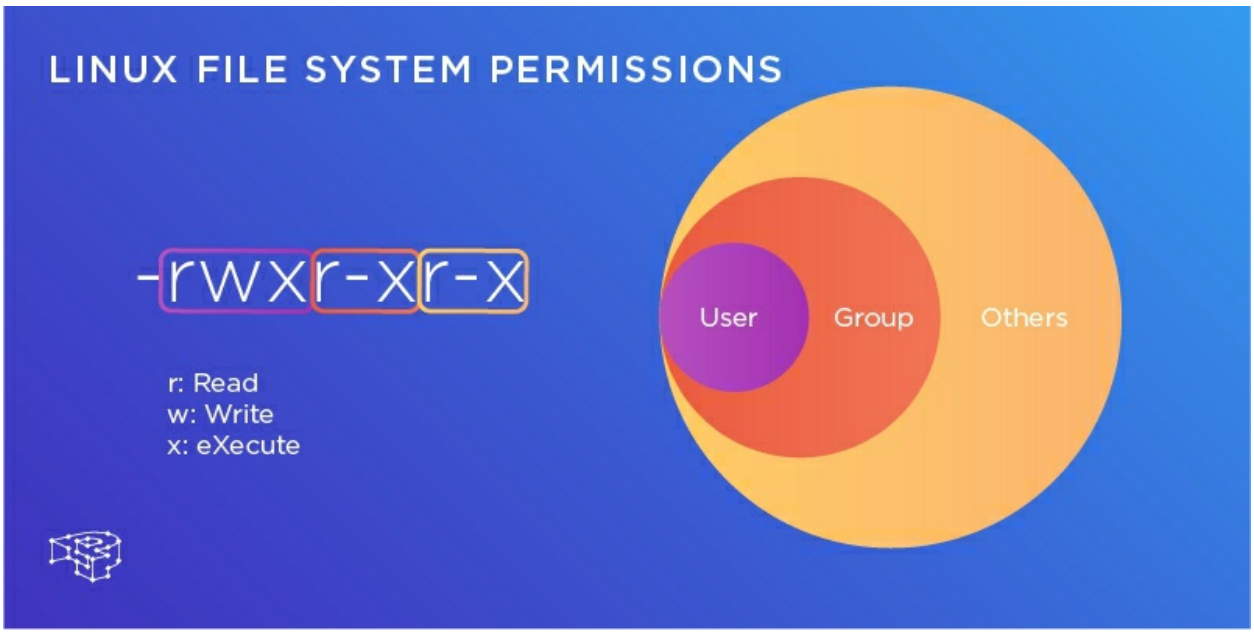
A "hard link," however, creates a direct link between the two files. Essentially they are the same file but with just another name. A hard link provides a good way of categorizing files without taking up further disk space.

You can create a hard link using the following syntax:

```
ln filenamebeinglinked filenametolinkto
```

The syntax is similar to that of a symbolic link but it doesn't use the **-s** switch.

Working with Permission



I mentioned permissions briefly before, when introduced the `ls -al` command.

The weird string you see on each file line, like `drwxr-xr-x`, defines the permissions of the file or folder.

Let's dissect it.

The first letter indicates the type of file:

- `-` means it's a normal file

- `d` means it's a directory
- `l` means it's a link

Then you have 3 sets of values:

- The first set represents the permissions of the **owner** of the file
- The second set represents the permissions of the members of the **group** the file is associated to
- The third set represents the permissions of the **everyone else**

Those sets are composed by 3 values. `rwx` means that specific *persona* has read, write and execution access. Anything that is removed is swapped with a `-`, which lets you form various combinations of values and relative permissions: `rw-`, `r--`, `r-x`, and so on.

You can change the permissions given to a file using the `chmod` command.

`chmod` can be used in 2 ways. The first is using symbolic arguments, the second is using numeric arguments. Let's start with symbols first, which is more intuitive.

You type `chmod` followed by a space, and a letter:

- `a` stands for *all*
- `u` stands for *user*
- `g` stands for *group*
- `o` stands for *others*

Then you type either `+` or `-` to add a permission, or to remove it. Then you enter one or more permissions symbols (`r`, `w`, `x`).

All followed by the file or folder name.

Here are some examples:

```
chmod a+r filename #everyone can now read
chmod a+rw filename #everyone can now read and write
chmod o-rwx filename #others (not the owner, not in the same group of the file) cannot read, write or execute the file
```

You can apply the same permissions to multiple personas by adding multiple letters before the `+` / `-`:

```
chmod og-r filename #other and group can't read any more
```

In case you are editing a folder, you can apply the permissions to every file

contained in that folder using the `-r` (recursive) flag.

Numeric arguments are faster but I find them hard to remember when you are not using them day to day. You use a digit that represents the permissions of the persona. This number value can be a maximum of 7, and it's calculated in this way:

- `1` if has execution permission
- `2` if has write permission
- `4` if has read permission

This gives us 4 combinations:

- `0` no permissions
- `1` can execute
- `2` can write
- `3` can write, execute
- `4` can read
- `5` can read, execute
- `6` can read, write
- `7` can read, write and execute

We use them in pairs of 3, to set the permissions of all the 3 groups altogether:

```
chmod 777 filename
chmod 755 filename
chmod 644 filename
```

Owner and group

You can change the owner of a file using the `chown` command:

```
chown <username> <filename>
```

You can change the group of a file using the `chgrp` command:

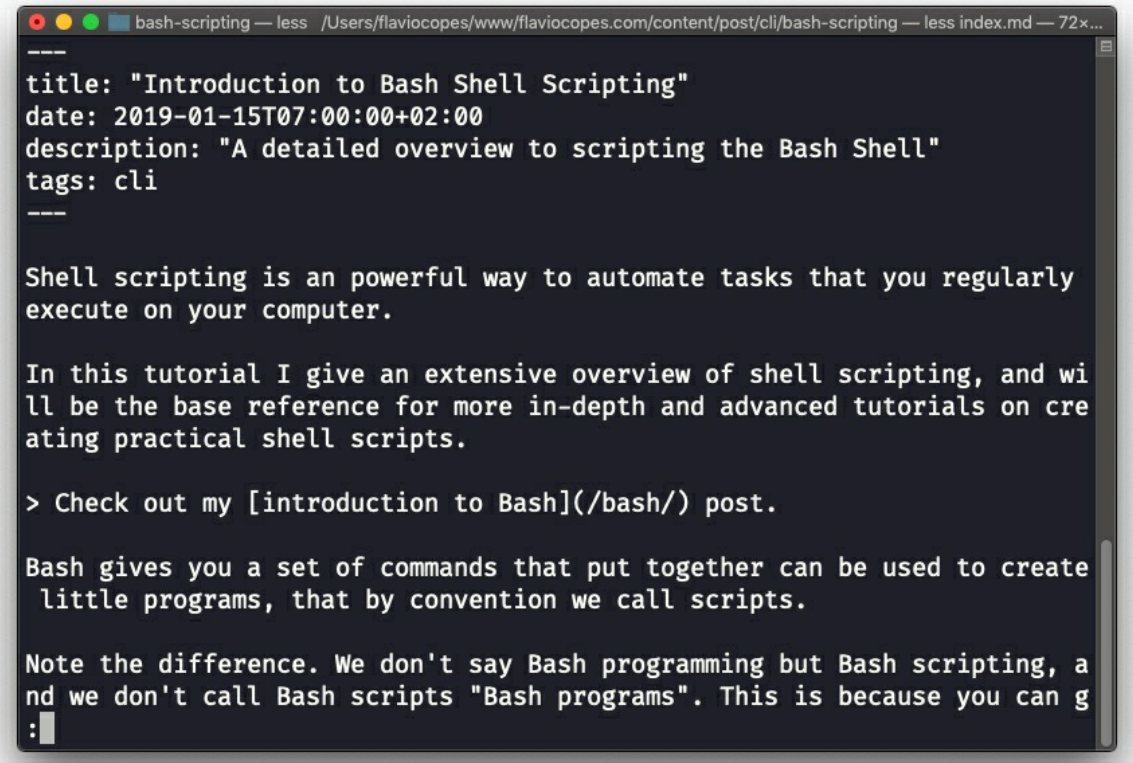
```
chgrp <group> <filename>
```

Managing files content

less

The `less` command is one I use a lot. It shows you the contents stored inside a file, in a nice and interactive UI.

Usage: `less <filename>` .



Once you are inside a `less` session, you can quit by pressing `q`.

You can navigate the file contents using the `up` and `down` keys, or using the `space bar` and `b` to navigate page by page. You can also jump to the end of the file pressing `G` and jump back to the start pressing `g`.

You can search contents inside the file by pressing `/` and typing a word to search. This searches *forward*. You can search backwards using the `?` symbol and typing a word.

This command just visualises the file's content. You can directly open an editor by pressing `v`. It will use the system editor, which in most cases is `vim`.

Pressing the `F` key enters *follow mode*, or *watch mode*. When the file is changed by someone else, like from another program, you get to see the changes *live*. By default this is not happening, and you only see the file version at the time you opened it. You need to press `ctrl-C` to quit this mode. In this case the behaviour is similar to running the `tail -f <filename>` command.

You can open multiple files, and navigate through them using `:n` (to go to the

next file) and `:p` (to go to the previous).

tail

I just mentioned `tail` in the previous command, so let's use this opportunity to explain what it does.

Its best use case in my opinion is when called with the `-f` option. It opens the file at the end, and watches for file changes. Any time there is new content in the file, it is printed in the window. This is great for watching log files, for example:

```
tail -f /var/log/system.log
```

To exit, press `ctrl-C`.

You can print the last 10 lines in a file:

```
tail -n 10 <filename>
```

You can print the whole file content starting from a specific line using `+` before the line number:

```
tail -n +10 <filename>
```

`tail` can do much more and as always my advice is to check `man tail`.

cat

Similar to `tail` in some way, we have `cat`. Except `cat` can also add content to a file, and this makes it super powerful.

In its simplest usage, `cat` prints a file's content to the standard output:

```
cat file
```

You can print the content of multiple files:

```
cat file1 file2
```

and using the output redirection operator `>` you can concatenate the content of multiple files into a new file:

```
cat file1 file2 > file3
```

Using `>>` you can append the content of multiple files into a new file, creating it if it does not exist:

```
cat file1 file2 >> file3
```

When watching source code files it's great to see the line numbers, and you can have `cat` print them using the `-n` option:

```
cat -n file1
```

You can only add a number to non-blank lines using `-b`, or you can also remove all the multiple empty lines using `-s`.

`cat` is often used in combination with the pipe operator `|` to feed a file content

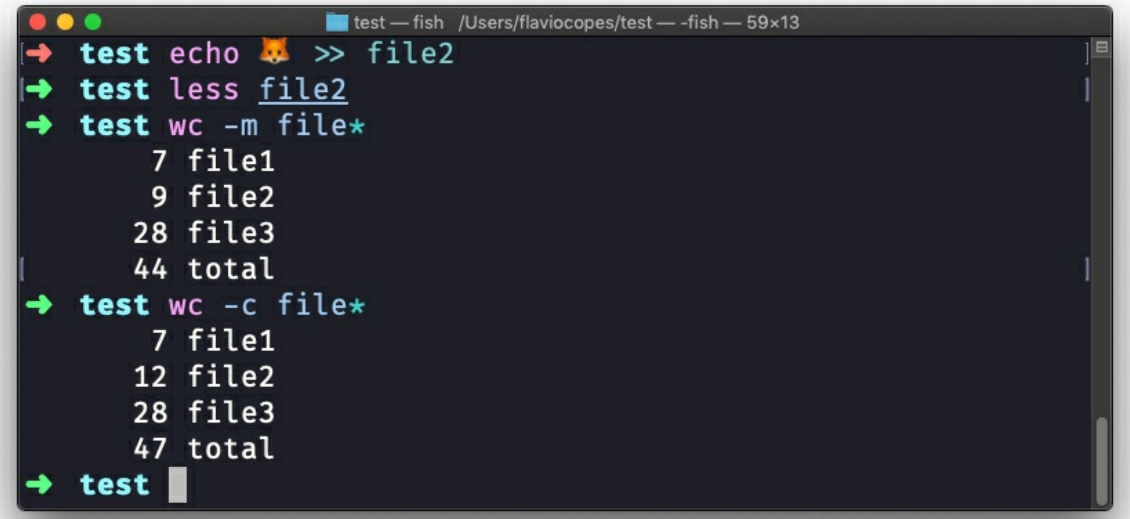
as input to another command: `cat file1 | anothercommand` .

wc

`wc` means *word count*. Here is the most common usage:

- `wc -l file1` count lines
- `wc -w file1` count words
- `wc -c file1` count characters
- `wc -m file1` count characters with multibyte support (i.e. emojis count as 1, not as multiple characters)

You can run `wc` providing multiple files, in which case it will do the calculations on each file separately, and then print a summary:



find

The `find` command can be used to find files or folders matching a particular search pattern. It searches recursively.

Let's learn it by example.

Find all the files under the current tree that have the `.js` extension and print the relative path of each file matching:

```
find . -name '*.js'
```

It's important to use quotes around special characters like `*` to avoid the shell interpreting them.

Find directories under the current tree matching the name "src":

```
find . -type d -name src
```

Use `-type f` to search only files, or `-type l` to only search symbolic links.

`-name` is case sensitive. use `-iname` to perform a case-insensitive search.

You can search under multiple root trees:

```
find folder1 folder2 -name filename.txt
```

Find directories under the current tree matching the name “node_modules” or ‘public’:

```
find . -type d -name node_modules -or -name public
```

You can also exclude a path, using `-not -path` :

```
find . -type d -name '*.md' -not -path 'node_modules/*'
```

You can search files that have more than 100 characters (bytes) in them:

```
find . -type f -size +100c
```

Search files bigger than 100KB but smaller than 1MB:

```
find . -type f -size +100k -size -1M
```

Search files edited more than 3 days ago

```
find . -type f -mtime +3
```

Search files edited in the last 24 hours

```
find . -type f -mtime -1
```

You can delete all the files matching a search by adding the `-delete` option. This deletes all the files edited in the last 24 hours:

```
find . -type f -mtime -1 -delete
```

You can execute a command on each result of the search. In this example we run `cat` to print the file content:

```
find . -type f -exec cat {} \;
```

notice the terminating `\;` . `{}` is filled with the file name at execution time.

Working with Linux Networking Commands

1-Ifconfig

ifconfig utility is used to configure network interface parameters.

Mostly we use this command to check the IP address assigned to the system.

```
[root@localhost ~]# ifconfig -a
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        ether 00:0c:29:c5:a5:61 txqueuelen 1000 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
```

```
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 0 (Local Loopback)
        RX packets 2 bytes 140 (140.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 2 bytes 140 (140.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
[root@localhost ~]#
```

2-traceroute

traceroute print the route packets take to network host.

Destination host or IP is mandatory parameter to use this utility

```
[root@localhost ~]# traceroute eminyardim.com
traceroute to geekflare.com (162.159.243.243), 30 hops max, 60 byte packets
1 172.16.179.2 (172.16.179.2) 0.154 ms 0.074 ms 0.074 ms
2 * * *
3 * * *
```

3-dig

dig (Domain Information Groper) is a flexible tool for interrogating DNS name servers.

It performs DNS lookups and displays the answers that are returned from the name servers.

```
[root@localhost ~]# dig eminyardim.com
; <<>> DiG 9.9.4-RedHat-9.9.4-14.el7 <<>> geekflare.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18699
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; MBZ: 0005 , udp: 4000
;; QUESTION SECTION:
;geekflare.com.                 IN      A
;; ANSWER SECTION:
eminyardim.com.com.             5       IN      A       xxxxxxx
eminyardim.com.com.             5       IN      A       xxxxxx
;; Query time: 6 msec
;; SERVER: 172.16.179.2#53(172.16.179.2)
```

```
;; WHEN: Sun May 01 23:28:19 PDT 2016
;; MSG SIZE rcvd: 74
[root@localhost ~]#
```

4-telnet

telnet connect destination host:port via a telnet protocol if connection establishes means connectivity between two hosts is working fine.

```
[root@localhost ~]# telnet eminyardim.com 443
Trying 162.159.244.243...
Connected to eminyardim.com.
Escape character is '^]'.
```

5-nslookup

nslookup is a program to query Internet domain name servers.

```
[root@localhost ~]# nslookup eminyardim.com
Server:         172.16.179.2
Address:        172.16.179.2#53
Non-authoritative answer:
Name: eminyardim.com
Address: 162.159.243.243
Name: eminyardim.com
Address: 162.159.244.243
[root@localhost ~]#
```

6-netstat

Netstat command allows you a simple way to review each of your network connections and open sockets.

netstat with head output is very helpful while performing web server troubleshooting.

```
[root@localhost ~]# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp        0      0 172.16.179.135:58856   mirror.comp.nus.ed:http TIME_WAIT
tcp        0      0 172.16.179.135:34444   riksun.riken.go.jp:http ESTABLISHED
tcp        0      0 172.16.179.135:37948   mirrors.isu.net.sa:http TIME_WAIT
tcp        0      0 172.16.179.135:53128   ossm.utm.my:http       TIME_WAIT
tcp        0      0 172.16.179.135:59723   103.237.168.15:http     TIME_WAIT
tcp        0      0 172.16.179.135:60244   no-ptr.as20860.net:http TIME_WAIT
```

7-scp

scp allows you to secure copy files to and from another host in the network.

Ex:

```
scp $filename user@targethost:/$path
```

8-w

w prints a summary of the current activity on the system, including what each user is doing, and their processes.

Also list the logged in users and system load average for the past 1, 5, and 15 minutes.

```
[root@localhost ~]# w
23:32:48 up 2:52, 2 users, load average: 0.51, 0.36, 0.19
USER     TTY      LOGIN@   IDLE   JCPU   PCPU WHAT
chandan :0       20:41   ?xdm?   7:07 0.13s gdm-session-worker [pam/gdm-password]
chandan pts/0     20:42   0.00s 0.23s 3.42s /usr/libexec/gnome-terminal-server
[root@localhost ~]#
```

9-nmap

nmap is a one of the powerful commands, which checks the opened port on the server.

Usage example:

```
nmap $server_name
```

10-Enable/Disable Network Interface

You can enable or disable the network interface by using ifup/ifdown commands with ethernet interface parameter.

To enable eth0

```
#ifup eth0
```

To disable eth0

```
#ifdown eth0
```

I hope above Linux Commands help you to gather network information or troubleshoot the networking issue. If you are looking to learn Networking in details, then check out this great online course by SONIC.

Working with Linux Update-Upgrade Commands

How do I update debian based linux(ubuntu,mint,debian etc.) using terminal?

1. Open the terminal application

2. For remote server use the ssh command to login (e.g. ssh user@server-name)
3. Fetch update software list by running **sudo apt-get update** command
4. Update Ubuntu software by running **sudo apt-get upgrade** command
5. Reboot the Ubuntu box if required by running sudo reboot

Let us see all commands in details.

How to sync the package index files from their sources via Internet

Run the command:

```
$ sudo apt-get update
```

OR

```
$ sudo apt update
```

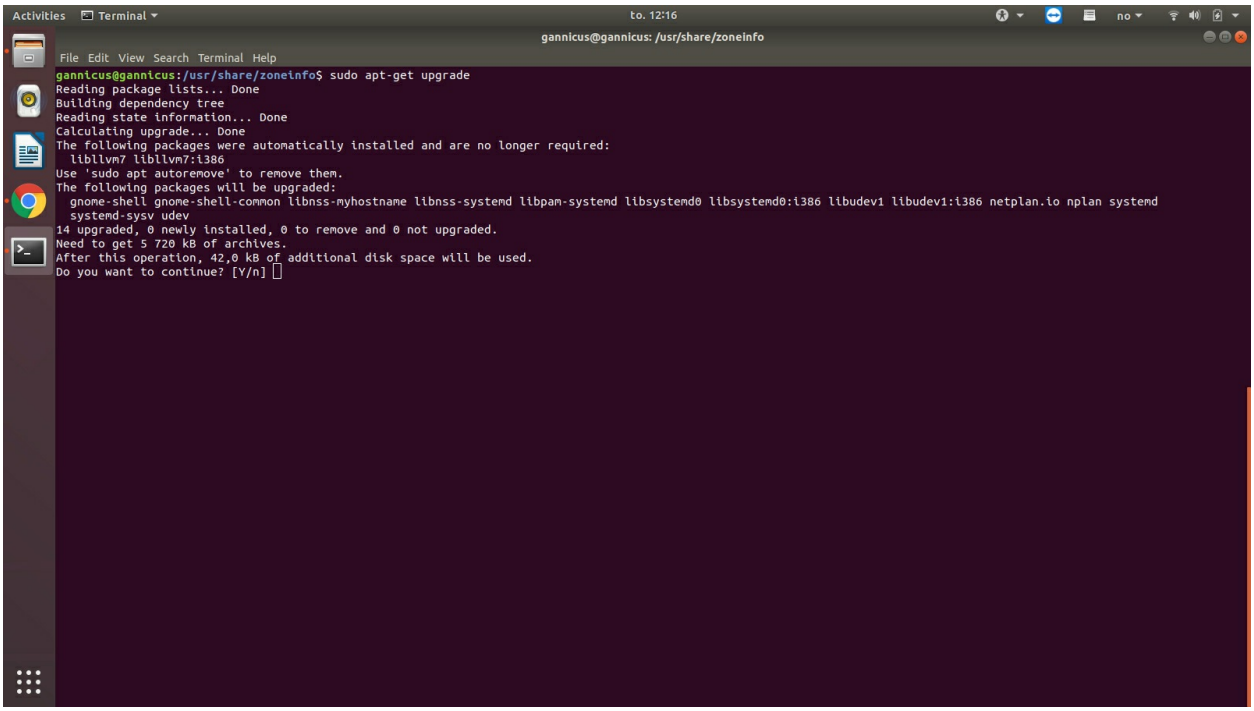
How to install the newest versions of all installed packages on Ubuntu

Execute the command:

```
$ sudo apt-get upgrade
```

OR

```
$ sudo apt upgrade
```



You may need to reboot the Linux server if kernel was updated:
$ sudo reboot

How to update already installed package named nginx

The syntax is:
$ sudo apt install Package Name Here
Alternatively, you can use the following syntax too:
$ sudo apt-get install Package Name Here
Let us upgrade nginx package:
$ sudo apt install nginx

If nginx package is already installed it will try to update to latest version:

Reading package lists... Done
Building dependency tree
Reading state information... Done
nginx is already the newest version (1.10.3-0ubuntu0.16.04.2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

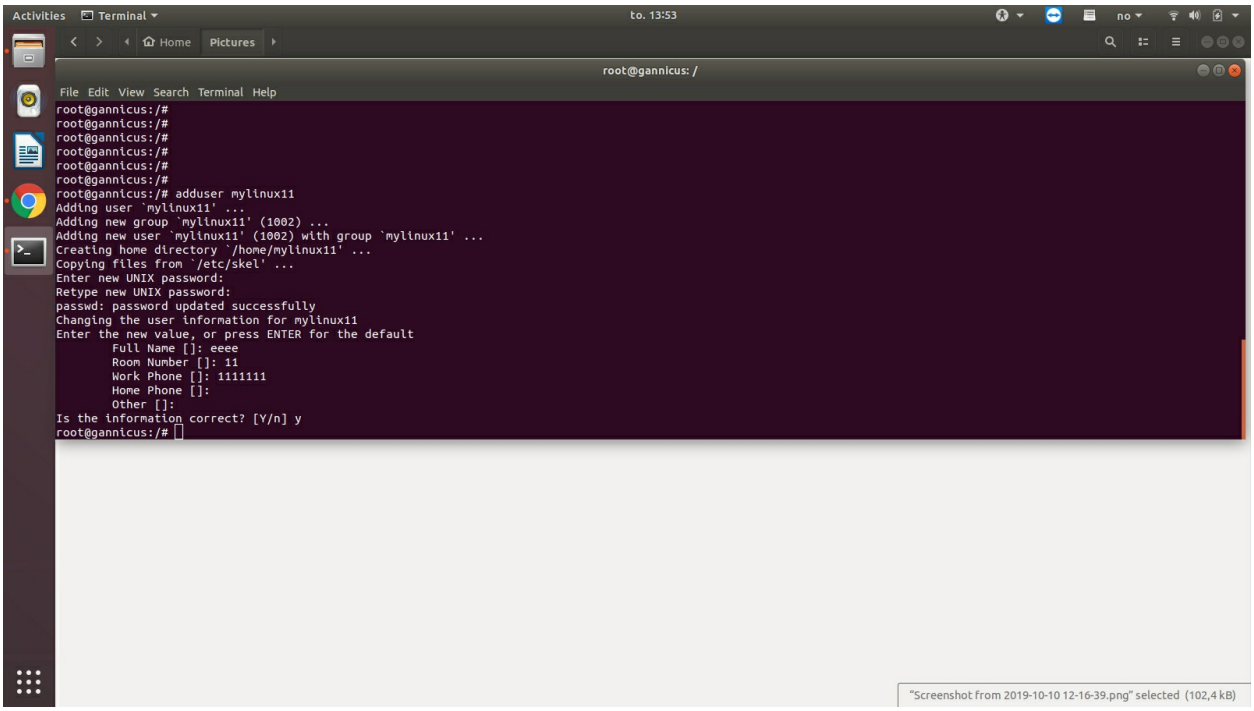
Working with User Administration

Creating a User

In Linux, every user is assigned an individual account which contains all the files, information, and data of the user. You can create multiple users in a Linux operating system. The steps to creating a user are:

Using Terminal

Step 1) Use command sudo adduser



Deleting, disabling account

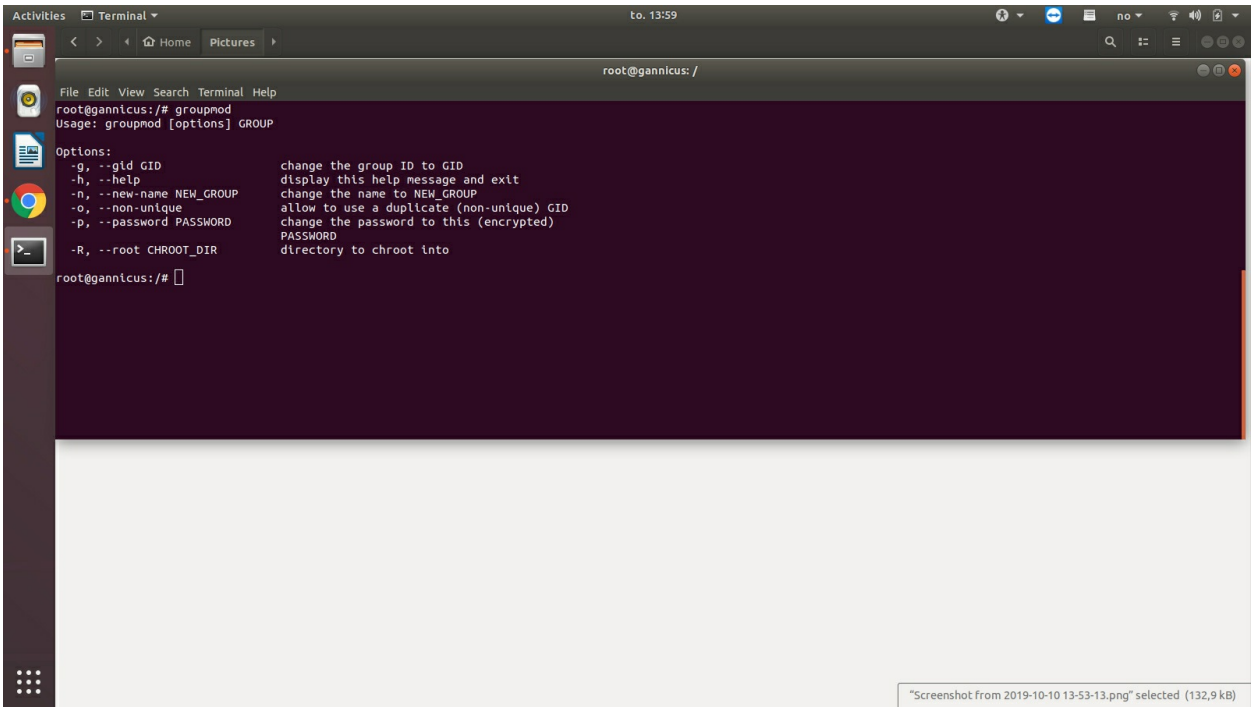
For disabling an account using Terminal, remove the password set on the account.

root@gannicus:/# userdel -r mylinux11

Adding users to the usergroups

You can view the existing groups on your Linux operating system by entering the following command:

groupmod "Press Tab key twice"



Now to add a user to a group, use the following syntax:

sudo usermod -a -G GROUPNAME USERNAME

Linux/Unix user management commands

Here is a list of linux user management commands

sudo adduser username **:Adds a user**

sudo passwd -l 'username' **:Disable a user**

sudo userdel -r 'username' **:Delete a user**

sudo usermod -a -G GROUPNAME USERNAME **:Add user a to a usergroup**

sudo deluser USER GROUPNAME **:Remove user from a user group**

finger **:Gives information on all logged in user**

finger username **:Gives information of a particular user**

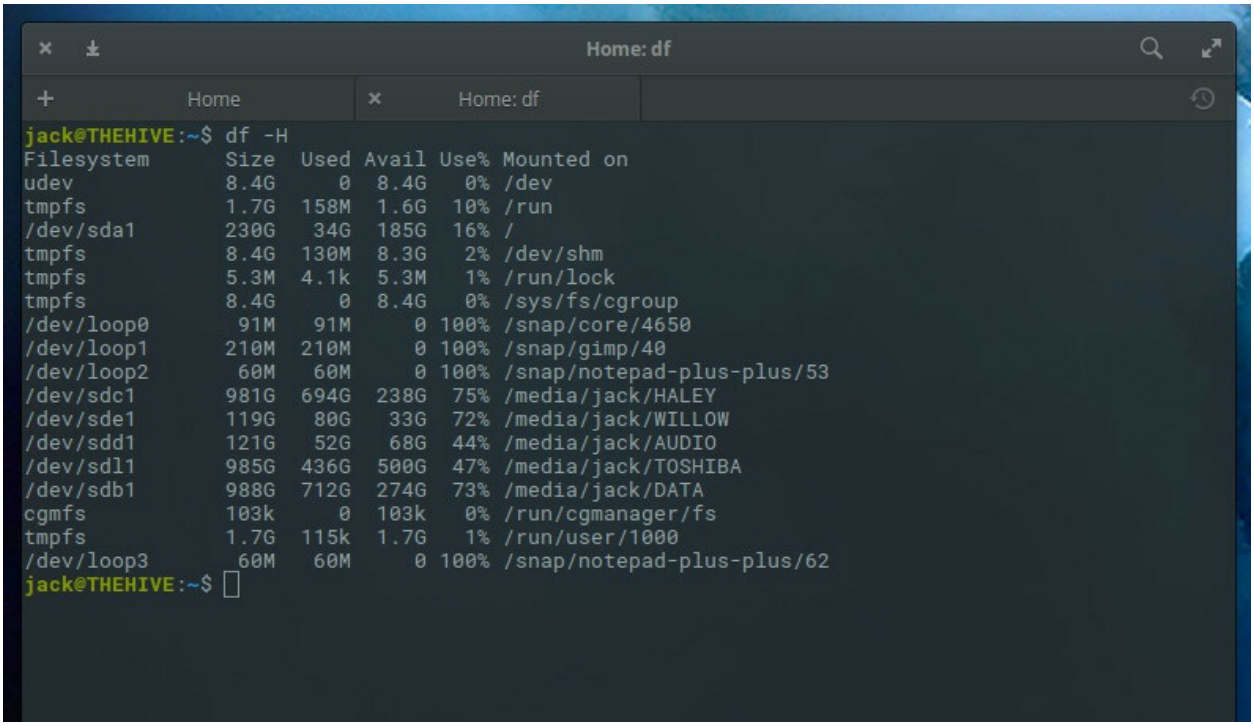
Summary:

- You can use both GUI or Terminal for User Administration
- You can create, disable and remove user accounts.
- You can add/delete a user to a usergroup.

Working with Disk Usage

df

The *df* command is the tool I first used to discover drive space on Linux, way back in the 1990s. It's very simple in both usage and reporting. To this day, df is my go-to command for this task. This command has a few switches but, for basic reporting, you really only need one. That command is *df -H*. The *-H* switch is for human-readable format. The output of *df -H* will report how much space is used, available, percentage used, and the mount point of every disk attached to your system



What if your list of drives is exceedingly long and you just want to view the space used on a single drive? With *df*, that is possible. Let's take a look at how much space has been used up on our primary drive, located at */dev/sda1*. To do that, issue the command:

```
df -H /dev/sda1
```

The output will be limited to that one drive

```
jack@THEHIVE:~$ df -H
Filesystem      Size  Used Avail Use% Mounted on
udev            8.4G     0  8.4G   0% /dev
tmpfs           1.7G  158M  1.6G  10% /run
/dev/sda1       230G   34G  185G  16% /
tmpfs           8.4G  130M  8.3G   2% /dev/shm
tmpfs           5.3M  4.1k  5.3M   1% /run/lock
tmpfs           8.4G     0  8.4G   0% /sys/fs/cgroup
/dev/loop0       91M   91M     0 100% /snap/core/4650
/dev/loop1      210M  210M     0 100% /snap/gimp/40
/dev/loop2       60M   60M     0 100% /snap/notepad-plus-plus/53
/dev/sdc1       981G  694G  238G  75% /media/jack/HALEY
/dev/sde1       119G   80G   33G  72% /media/jack/WILLOW
/dev/sdd1       121G   52G   68G  44% /media/jack/AUDIO
/dev/sdl1       985G  436G  500G  47% /media/jack/TOSHIBA
/dev/sdb1       988G  712G  274G  73% /media/jack/DATA
cgmfs           103k     0  103k   0% /run/cgmanager/fs
tmpfs           1.7G  115k  1.7G   1% /run/user/1000
/dev/loop3       60M   60M     0 100% /snap/notepad-plus-plus/62
jack@THEHIVE:~$ df -H /dev/sda1
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       230G   34G  185G  16% /
jack@THEHIVE:~$
```

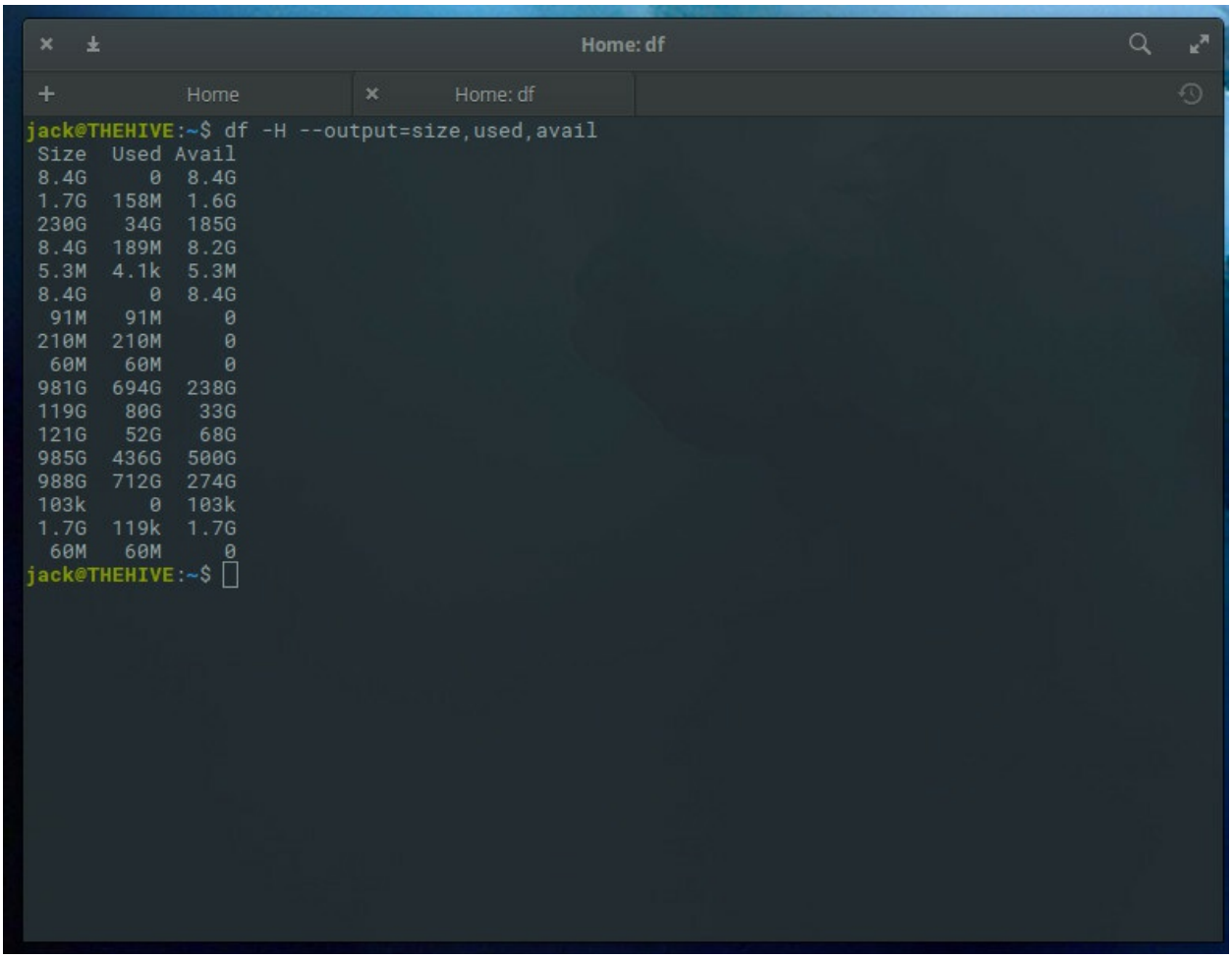
You can also limit the reported fields shown in the *df* output. Available fields are:

- source — the file system source
- size — total number of blocks
- used — spaced used on a drive
- avail — space available on a drive
- pcent — percent of used space, divided by total size
- target — mount point of a drive

Let’s display the output of all our drives, showing only the size, used, and avail (or availability) fields. The command for this would be:

df -H --output=size,used,avail

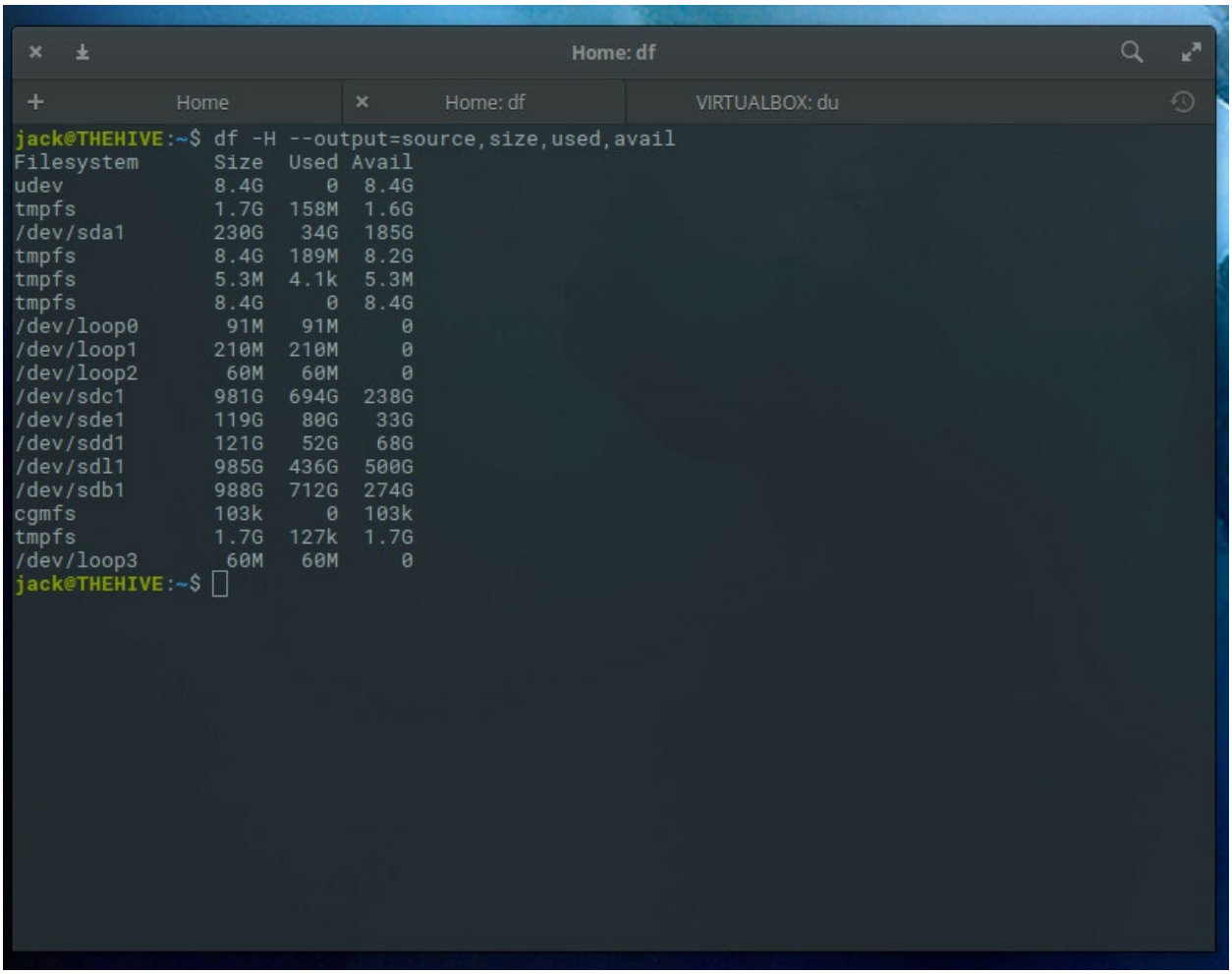
The output of this command is quite easy to read



The only caveat here is that we don’t know the source of the output, so we’d want to include source like so:

df -H --output=source,size,used,avail

Now the output makes more sense



du

Our next command is *du*. As you might expect, that stands for disk usage. The *du* command is quite different to the df command, in that it reports on directories and not drives. Because of this, you'll want to know the names of directories to be checked. Let's say I have a directory containing virtual machine files on my machine. That directory is */media/jack/HALEY/VIRTUALBOX*. If I want to find out how much space is used by that particular directory, I'd issue the command:

du -h /media/lee/HALEY/VIRTUALBOX

The output of the above command will display the size of every file in the directory

Working with Processes

The Linux terminal has a number of useful commands that can display running processes, kill them, and change their priority level. This post lists the classic, traditional commands, as well as some more useful, modern ones.

Many of the commands here perform a single function and can be combined — that's the Unix philosophy of designing programs. Other programs, like htop, provide a friendly interface on top of the commands.

top

The **top** command is the traditional way to view your system's resource usage and see the processes that are taking up the most system resources. Top displays a list of processes, with the ones using the most CPU at the top.

To exit top or htop, use the **Ctrl-C** keyboard shortcut. This keyboard shortcut usually kills the currently running process in the terminal.

htop

The **htop** command is an improved top. It's not installed by default on most Linux distributions — here's the command you'll need to install it on Ubuntu:

sudo apt-get install htop

htop displays the same information with an easier-to-understand layout. It also lets you select processes with the arrow keys and perform actions, such as killing them or changing their priority, with the F keys.

We've covered htop in more detail in the past.

ps

The **ps** command lists running processes. The following command lists all processes running on your system:

ps -A

This may be too many processes to read at one time, so you can pipe the output through the **less** command to scroll through them at your own pace:

ps -A | less

Press **q** to exit when you're done.

You could also pipe the output through **grep** to search for a specific process without using any other commands. The following command would search for the Firefox process:

ps -A | grep firefox

pstree

The **pstree** command is another way of visualizing processes. It displays them in tree format. So, for example, your X server and graphical environment would appear under the display manager that spawned them.

kill

The **kill** command can kill a process, given its process ID. You can get this information from the **ps -A**, **top** or **pgrep** commands.

kill PID

Technically speaking, the kill command can send any signal to a process. You can use **kill -KILL** or **kill -9** instead to kill a stubborn process.

pgrep

Given a search term, **pgrep** returns the process IDs that match it. For example, you could use the following command to find Firefox's PID:

pgrep firefox

You can also combine this command with kill to kill a specific process. Using pkill or killall is simpler, though.

pkill & killall

The **pkill** and **killall** commands can kill a process, given its name. Use either command to kill Firefox:

pkill firefox
killall firefox

We've covered pkill in more depth in the past.

renice

The **renice** command changes the nice value of an already running process. The nice value determines what priority the process runs with. A value of **-19** is very high priority, while a value of **19** is very low priority. A value of **0** is the default priority.

The renice command requires a process's PID. The following command makes a process run with very low priority:

renice 19 *PID*

You can use the **pgrep** trick above with renice, too.

Pgrep firefox

If you're making a process run at a higher priority, you'll require root permissions. On Ubuntu, use **sudo** for that:

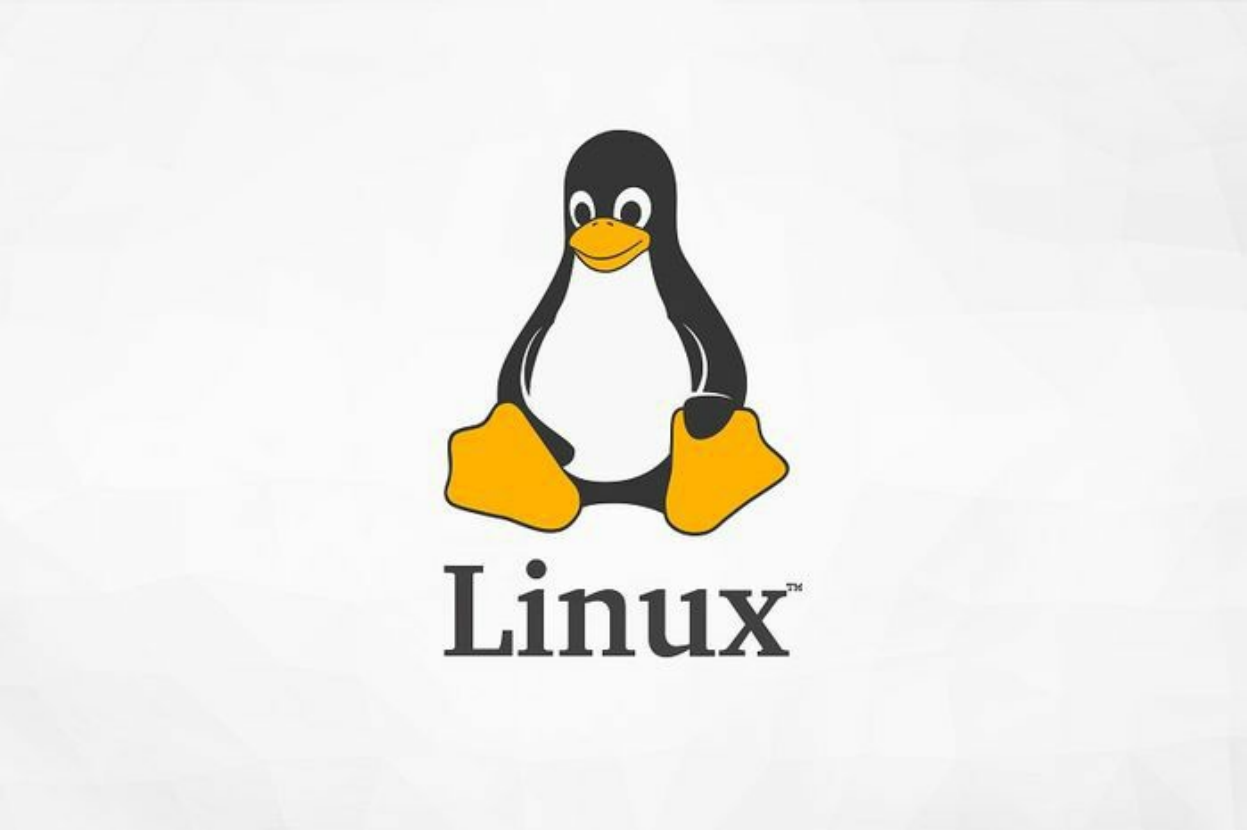
sudo renice -19 #

xkill

The **xkill** command is a way of easily killing graphical programs. Run it and your cursor will turn into an **x** sign. Click a program's window to kill that program. If you don't want to kill a program, you can back out of xkill by right-clicking instead.

You don't have to run this command from a terminal — you can also press Alt-F2, type **xkill** and press Enter to use it from a graphical desktop.

Configuring SSH Connection
and
Connect from another PC



Prerequisites

Before you can **establish a secure remote desktop protocol** with a remote machine, there are a few basic requirements to meet:

- The remote computer must be turned on at all times and have a network connection.
- The client and server applications need to be installed and enabled.
- You need the IP address or the name of the remote machine you want to connect to.
- You need to have the necessary permissions to access the remote computer.
- Firewall settings need to allow the remote connection.

What is SSH?

Secure Shell, sometimes referred to as **Secure Socket Shell**, is a protocol which allows you to connect securely to a remote computer or a server by using a text-based interface.

When a ssh connection , a shell session will be started, and you will be able to manipulate the server by typing commands within the client on your local computer.

System and network administrators use this protocol the most, as well as anyone who needs to manage a computer remotely in a highly secure manner.

How Does SSH Work?

In order to establish an SSH connection, you need two components: a client and the corresponding server-side component. An SSH client is an application you install on the computer which you will use to connect to another computer or a server. The client uses the provided remote host information to initiate the connection and if the credentials are verified, establishes the encrypted connection.

On the server's side, there is a component called an SSH daemon that is constantly listening to a specific TCP/IP port for possible client connection requests. Once a client initiates a connection, the SSH daemon will respond with the software and the protocol versions it supports and the two will exchange their identification data. If the provided credentials are correct, SSH creates a new session for the appropriate environment.

The default SSH protocol version for SSH server and SSH client communication is version 2.

How to Install an OpenSSH Client

Before you proceed with installing an SSH client, make sure it is not already installed. Many Linux distributions already have an SSH client. For Windows machines, you can install PuTTY or any other client of your choice to gain access to a server.

To check if the client is available on your Linux-based system, you will need to:

1. Load an SSH terminal. You can either search for "terminal" or press **CTRL** + **ALT** + **T** on your keyboard.
2. Type in **ssh** and press **Enter** in the terminal.
3. If the client is installed, you will receive a response that looks like this:

```
username@host:~$ ssh

usage: ssh [-1246AaCfGgKkMNnqsTtVvXxYy] [-b bind_address] [-c cipher_spec]
[-D [bind_address:]port] [-E log_file] [-e escape_char]
[-F configfile] [-I pkcs11] [-i identity_file]
[-J [user@]host[:port]] [-L address] [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port] [-
Q query_option] [-R address] [-S ctl_path] [-W host:port] [-w local_tun[:remote_tun]]
[user@]hostname [command]

username@host:~$
```

This means that you are ready to remotely connect to a physical or virtual machine. Otherwise, you will have to install the OpenSSH client:

1. Run the following command to install the OpenSSH client on your computer: sudo apt-get install openssh-client
2. Type in your superuser password when asked.
3. Hit Enter to complete the installation.

You are now able to SSH into any machine with the server-side application on it, provided that you have the necessary privileges to gain access, as well as the hostname or IP address.

How to Install an OpenSSH Server

In order to accept SSH connections, a machine needs to have the server-side part of the SSH software toolkit.

If you first want to check if OpenSSH server is available on the Ubuntu system of the remote computer that needs to accept SSH connections, you can try to connect to the local host:

1. Open the terminal on the server machine. You can either search for “terminal” or press **CTRL + ALT + T** on your keyboard.
2. Type in **ssh localhost** and hit enter.
3. For the systems **without** the SSH server installed the response will look similar to this:

```
username@host:~$ ssh localhost
ssh: connect to host localhost port 22: Connection refused username@host:~$
```

If the above is the case, you will need to install the OpenSSH server. Leave the terminal open and:

1. Run the following command to install the SSH server:

sudo apt-get install openssh-server ii.

1. Type in your **superuser password** when asked.

2. **Enter** and **Y** to allow the installation to continue after the disk space prompt.

The required support files will be installed, and then you can check if the SSH server is running on the machine by typing this command:

```
sudo service ssh status
```

The response in the terminal should look similar to this if the SSH service is now running properly:

```
username@host:-$ sudo service ssh status
• ssh.service - OpenBSD Secure Shell server
Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enab
Active: active (running) since Fr 2018-03-12 10:53:44 CET; 1min 22s ago Process: 1174
ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCES

Main PID: 3165 (sshd)
```

Another way to test if the OpenSSH server is installed properly and will accept connections is to try running the **ssh localhost** command again in your terminal prompt. The response will look similar to this screen when you run the command for the first time:

```
username@host:~$ ssh localhost

The authenticity of host 'localhost (127.0.0.1)' can't be established. ECDSA key fingerprint is
SHA256:9jqmhko9Yo1EQAS1QeNy9xKceHFG5F8W6kp7EX9U3Rs. Are you sure you want to
continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.

username@host:~$
```

Enter **yes** or **y** to continue.

Congratulations! You have set up your server to accept SSH connection requests from a different
computer using an SSH client.

TIP

You can now edit the SSH daemon configuration file, for example, you can change the default port for SSH connections. In the terminal prompt, run this command:

```
sudo nano /etc/ssh/sshd_config
```

The configuration file will open in the editor of your choice. In this case, we used nano.

If you need to install nano, run this command:

```
sudo apt-get install nano
```

Please note that you need to restart SSH service every time you make any changes to the `sshd_config` file by running this command:

```
sudo service ssh restart
```

How to Connect via SSH

Now that you have the OpenSSH client and server installed on every machine you need, you can establish a secure remote connection with your servers. To do so:

1. Open the SSH terminal on your machine and run the following command:

```
ssh your_username@host_ip_address
```

If the username on your local machine matches the one on the server you are trying to connect to, you can just type:

```
ssh host_ip_address
```

And hit **Enter**.

2. Type in your password and hit **Enter**.
3. When you are connecting to a server for the very first time, it will ask you if you want to continue connecting. Just type yes and hit **Enter**.

If the computer you are trying to remotely connect to is on the same network, then it is best to use the private IP address instead of the public IP address. Otherwise, you will have to use the public IP address only. Additionally, make sure that you know the correct TCP port OpenSSH is listening to for connection requests and that the port forwarding settings are correct. The default port is 22 if nobody changed configuration in the sshd_config file. You may also just append the port number after the host IP address.

Here is the example of a connection request using the OpenSSH client. We will specify the port number as well:

```
username@machine:~$ ssh kollli@188.52.53.252 –p7654 phoenixnap@185.52.53.222’s password:

The authenticity of host '185.52.53.222 (185.52.53.222)' can't be established. ECDSA key fingerprint is SHA256:9lyrpzo5Yo1EQAS2QeHy9xKceHFH8F8W6kp7EX2O3Ps. Are you sure you want to continue connecting (yes/no)? yes

username@host:~$
```

You are now able to manage and control a remote machine using your terminal. If you have trouble connecting to a remote server, make sure that:

- The IP address of the remote machine is correct.
- The port SSH daemon is listening to is not blocked by a firewall or forwarded incorrectly.
- Your username and password are correct.
- The SSH software is installed properly.

Configuring VNC Connection

On the computer you want to control

1. Download VNC Server to the computer you want to control as either a DEB or RPM.
2. Install the VNC Server program:
    - Open a Terminal.
    - Change directory to the download location, e.g. `cd ~/Downloads`
    - Run one of the following commands, depending on your version of Linux
      `sudo apt install ./<download-file>` # Ubuntu
      `sudo yum install -y <download-file>` # Red Hat Enterprise Linux (RHEL) and CentOS
3. Download and run the vncsetup-helper script:
    - Open a Terminal window
    - Change directory to the download location, e.g. `cd ~/Downloads`
    - Extract the contents: `tar -xzf vncsetup.tar.gz`

    - Execute the script: `sudo ./vncsetup.sh`

Note:

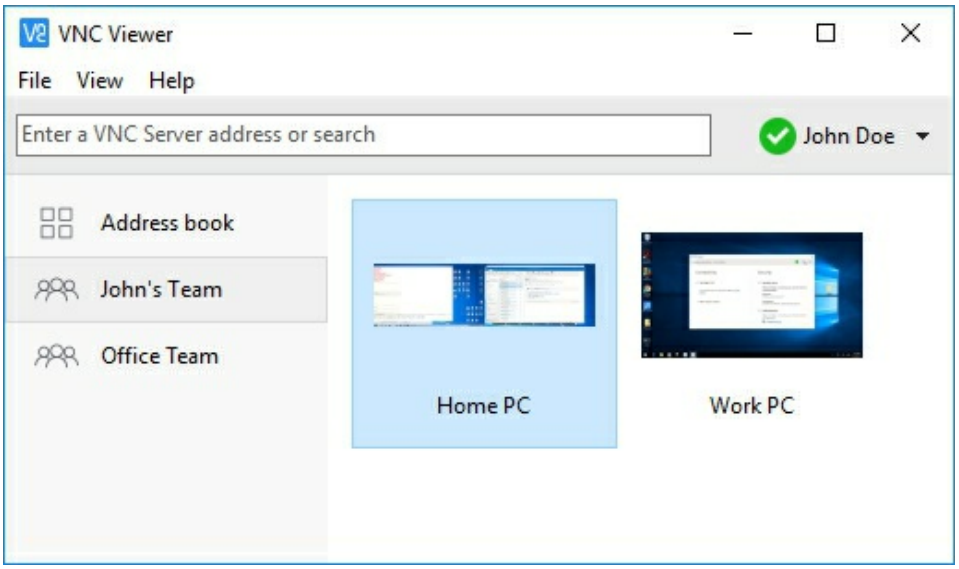
- If you have a Raspberry Pi, VNC Connect is pre-installed and pre-licensed, so all you have to do is follow these instructions to enable it. You don't need to read this page.
- If you have Ubuntu 18.04 LTS, note that Wayland is **not** supported, so for VNC Server in Service Mode you must edit the `/etc/gdm3/custom.conf` file, uncomment `WaylandEnable=false`, and reboot in order to remotely access the login screen.

More advanced instructions and example commands for getting started on Linux can be found here:
https://www.realvnc.com/en/connect/docs/deployment-script.html

By signing in with your RealVNC account, your computer is joined to your team. If you have multiple teams, you will be prompted to select which team to join. You can check by signing in to your RealVNC account online and checking the **Computers** page.

On the device you want to control from

1. Download VNC Viewer.
2. Install the VNC Viewer program:
    - Open a Terminal.
    - Change directory to the download location, e.g. cd ~/Downloads
    - Run one of the following commands, depending on your version of Linux
      sudo apt install ./<download-file> # Ubuntu
      sudo yum install -y <download-file> # Red Hat Enterprise Linux (RHEL) and CentOS
3. Sign in using your RealVNC account credentials. You should see the remote computer appear in your team:



4. Click or tap to connect.

Extract .tar.gz Files using Linux Command Line

Extracting .tar.gz files

1) If your *tar* file is compressed using a *gzip* compressor, use this command to uncompress it.

$ tar xvzf file.tar.gz

Where,

x: This option tells *tar* to extract the files.

v: The “v” stands for “verbose.” This option will list all of the files one by one in the archive.

z: The *z option* is very important and tells the *tar* command to uncompress the file (*gzip*).

f: This options tells *tar* that you are going to give it a file name to work with.

A tarball is a group or archive of files that are bundled together using the tar command and have the .tar file extension.

```
root@mail [~]#
root@mail [~]# tar xvzf  file.tar.gz
test/
test/test.doc
test/test.doc3
test/test.doc1
test/test.doc2
root@mail [~]#
```

2) To uncompress tar.gz file into a different directory, use the command below:

$ tar xvzf file.tar.gz -C /path/to/somedirectory

Where the -C argument is used to specify the path to place the file. By defaults files will be extracted into the current directory. To change the directory, we use -C option.

```
root@mail [~]#
root@mail [~]# tar xvzf file.tar.gz -C test3/
test/
test/test.doc
test/test.doc3
test/test.doc1
test/test.doc2
root@mail [~]#
```

3) To extract test.doc file from the file.tar.gz tarball, use the following command:

$ tar -xvzf file.tar.gz test.doc

```
root@mail [~]#
root@mail [~]# tar -xvzf file.tar.gz test/test.doc
test/test.doc
root@mail [~]#
```

4) To view a detailed table of content by listing all files in archive, you can use the following command:

$ tar -tvf file.tar.gz

```
root@mail [~]#
root@mail [~]# tar -tvf file.tar.gz
drwxr-xr-x root/root         0 2017-08-01 20:12 test/
-rw-r--r-- root/root         0 2017-08-01 20:42 test/test.doc
-rw-r--r-- root/root         0 2017-08-01 20:12 test/test.doc3
-rw-r--r-- root/root         0 2017-08-01 20:12 test/test.doc1
-rw-r--r-- root/root         0 2017-08-01 20:12 test/test.doc2
root@mail [~]#
```

5) To extract the .gz file without tar, you can use the following command:

$ gunzip file.gz

```
root@mail [~]#
root@mail [~]# gunzip file.gz
root@mail [~]#
```

Extracting .tar.bz2 files

This is just about the same as the *gzip* decompression. The major difference is that the *z option* has been replaced by the *j option.*

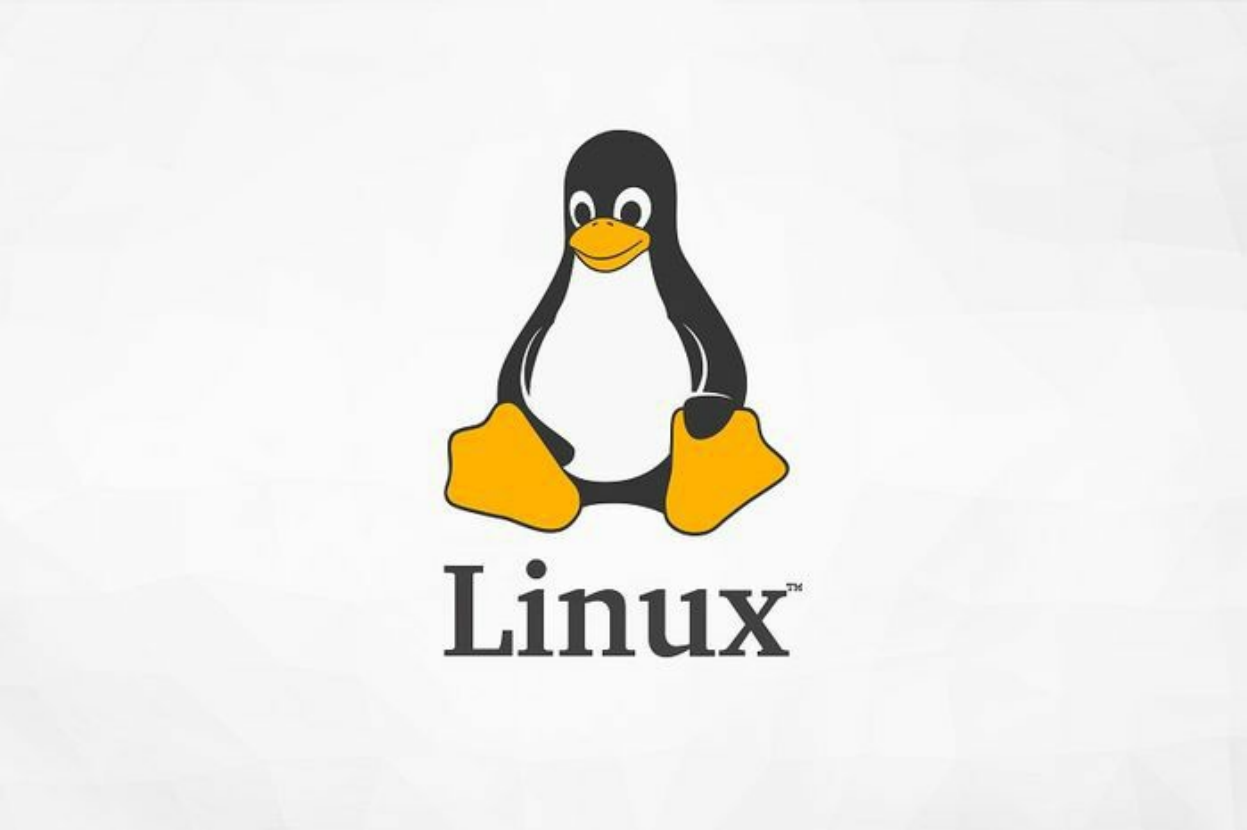
If your *tar* file is compressed using a *bZip2* compressor, use the following command to extract it.

$ tar xvjf file.tar.bz2

Where ‘j’ will decompress a bzip2 file.

```
root@mail [~]#
root@mail [~]# tar xvjf file.tar.bz2
test/
test/test.doc
test/test.doc3
test/test.doc1
test/test.doc2
root@mail [~]#
```

Must Know Linux Shortcuts



1. Tab

This is the Linux shortcut you cannot live without. It will save you so much time in the Linux command line.

Just start typing a command, filename, directory name or even command options and hit the tab key. It will either automatically complete what you were typing or it will show all the possible results for you.

If you could only remember one shortcut, this would be the chosen one.

2. Ctrl + C

These are the keys you should press in order to break out of a command or process on a terminal. This will stop a running program immediately.

If you want to stop using a program running in the foreground, just press this key combination.

3. Ctrl + Z

This shortcut will send a running program in the background. Normally, you can achieve this before running the program using the & option but if you

forgot to do that, use this key combination.

4. Ctrl + D

This keyboard shortcut will log you out of the current terminal. If you are using an SSH connection, it will be closed. If you are using a terminal directly, the application will be closed immediately.

Consider it equivalent to the ‘exit’ command.

5. Ctrl + L

How do you clear your terminal screen? I guess using the clear command.

Instead of writing C-L-E-A-R, you can simply use Ctrl+L to clear the terminal. Handy, isn’t it?

6. Ctrl + A

This shortcut will move the cursor to the beginning of the line.

Suppose you typed a long command or path in the terminal and you want to go to the beginning of it, using the arrow key to move the cursor will take plenty of time. Do note that you cannot use the mouse to move the cursor to the beginning of the line.

This is where Ctrl+A saves the day.

7. Ctrl + E

This shortcut is sort of opposite to Ctrl+A. Ctrl+A sends the cursor to the beginning of the line whereas Ctrl+E moves the cursor to the end of the line.

Note: If you have the Home and End keys on your keyboard, you can also use them. Home is equivalent to Ctrl +A and End is equivalent to Ctrl + E.

8. Ctrl + U

Typed a wrong command? Instead of using the backspace to discard the current command, use Ctrl+U shortcut in the Linux terminal. This shortcut erases everything from the current cursor position to the beginning of the line.

9. Ctrl + K

This one is similar to the Ctrl+U shortcut. The only difference is that instead of the beginning of the line, it erases everything from the current cursor position to the end of the line.

10. Ctrl + W

You just learned about erasing text till the beginning and the end of the line.

But what if you just need to delete a single word? Use the Ctrl+W shortcut.

Using Ctrl+W shortcut, you can erase the word preceding to the cursor position. If the cursor is on a word itself, it will erase all letters from the cursor position to the beginning of the word.

The best way to use it to move the cursor to the next space after the targetted word and then use the Ctrl+W keyboard shortcut.

11. Ctrl + Y

This will paste the erased text that you saw with Ctrl + W, Ctrl + U and Ctrl + K shortcuts. Comes handy in case you erased wrong text or if you need to use the erased text someplace else.

12. Ctrl + P

You can use this shortcut to view the previous command. You can press it repeatedly to keep on going back in the command history. In a lot of terminals, the same can be achieved with PgUp key.

13. Ctrl + N

You can use this shortcut in conjugation with Ctrl+P. Ctrl+N displays the next command. If you are viewing previous commands with Ctrl+P, you can use Ctrl+N to navigate back and forth. Many terminals have this shortcut mapped to the PgDn key.

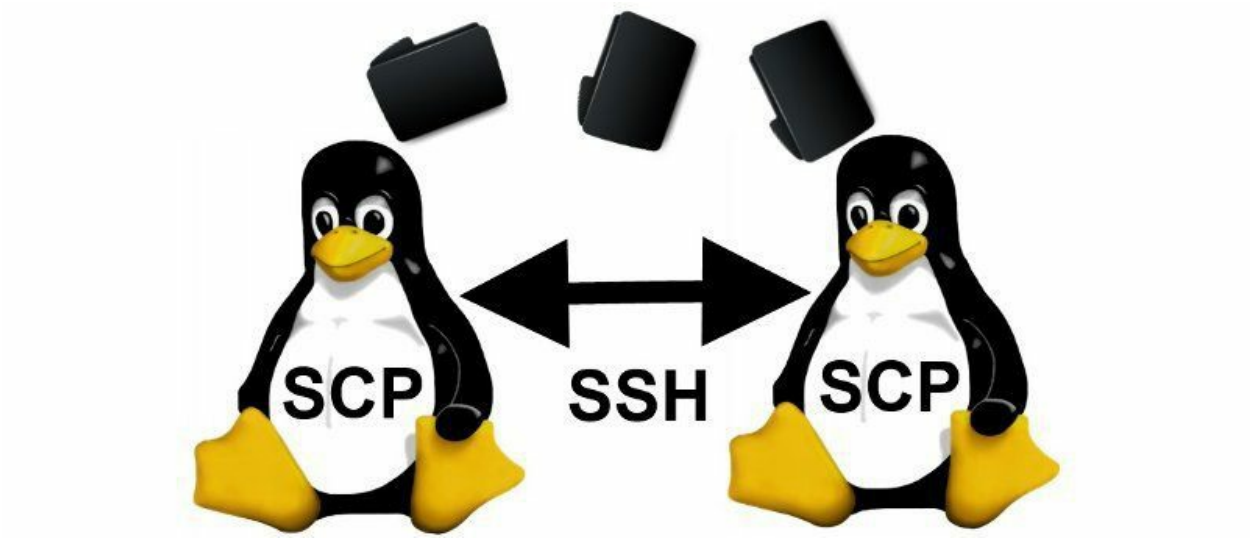
Bonus shortcut: Ctrl + R to search in command history

You typed some command but cannot remember what it was exactly? Meet Ctrl + R.

This keyboard shortcut allows you to perform a search in your command history. Just press Ctrl+R and start typing. It will show the last command that matches the string you typed.

Securely Copy Files Using SCP

This post includes SCP examples. SCP or secure copy allows secure transferring of files between a local host and a remote host or between two remote hosts. It uses the same authentication and security as the Secure Shell (SSH) protocol from which it is based. SCP is loved for it’s simplicity, security and pre-installed availability.



SCP examples

Copy file from a remote host to local host SCP example:

$ scp username@from_host:file.txt /local/directory/

Copy file from local host to a remote host SCP example:

$ scp file.txt username@to_host:/remote/directory/

Copy directory from a remote host to local host SCP example:

$ scp -r username@from_host:/remote/directory/ /local/directory/

Copy directory from local host to a remote hos SCP example:

$ scp -r /local/directory/ username@to_host:/remote/directory/

Copy file from remote host to remote host SCP example:

$ scp username@from_host:/remote/directory/file.txt username@to_host:/remote/directory/

Notes:

— SCP example: scp -r root@122.122.122.122:/var/www/gala/ /home/hydn/backups/test/

— Host can be IP or domain name. Once you click return, you will be prompted for SSH password.

— Although this page covers SCP Linux, the instructions will also work for Mac using “Terminal”. You can also usewinscp accomplish this on a

Windows PC/server.

Package Managers for Linux

Debian and Ubuntu Package Management

The Debian package management system, based on a tool called `dpkg` with the very popular `apt` system, is a powerful, popular, and useful method of package management. In addition to Debian, a number of other prominent distributions of GNU/Linux are derived from the Debian system, most notably the Ubuntu family of distributions.

As a result, these instructions apply to Debian and Ubuntu systems. While Debian and derived systems are not necessarily binary-compatible, `.debs` packaged for Debian are often compatible with Ubuntu (though this is not a supported workflow).

Advanced Packaging Tool (APT)

You may already be familiar with `apt-get`, a command which uses the advanced packaging tool to interact with the operating system's package system. The most relevant and useful commands are (to be run with root privileges):

- `apt-get install package-name(s)` –

  Installs the package(s) specified, along with any dependencies.

- `apt-get remove package-name(s)` –

Removes the package(s) specified, but does not remove dependencies.

- apt-get autoremove - Removes any *orphaned* dependencies, meaning those that remain installed but are no longer required.
- apt-get clean - Removes downloaded package files (.deb) for software that is already installed.
- apt-get purge package-name(s) - Combines the functions of remove and clean for a specific package, as well as configuration files.
- apt-get update - Reads the /etc/apt/sources.list file and updates the system's database of packages available for installation. Run this after changing sources.list .
- apt-get upgrade - Upgrades all packages if there are updates available. Run this after running apt-get update .

While apt-get provides the most often-used functionality, APT provides additional information in the apt-cache command.

- apt-cache search package-name(s) - If you know the name of a piece of software but apt-get install fails or points to the wrong software, this looks for other possible names.
- apt-cache show package-name(s) - Shows dependency information, version numbers and a basic description of the package.
- apt-cache depends package-name(s) - Lists the packages that the specified packages depends upon in a tree. These are the packages that will be installed with the apt-get install command.
- apt-cache rdepends package-name(s) - Outputs a list of packages that depend upon the specified package. This list can often be rather long, so it is best to pipe its output through a command, like less .
- apt-cache pkgnames - Generates a list of the currently installed packages on your system. This list is often rather long, so it is best to pipe its output through a program, like less , or direct the output to a text file.

Aptitude

Aptitude is another front-end interface for APT. In addition to a graphical interface, Aptitude provides a combined command-line interface for most APT functionality. Some notable commands are:

- `aptitude update`, `aptitude install`, `aptitude remove`, `aptitude clean`, or `aptitude purge` - Same as their apt-get counterparts.
- `aptitude search` or `aptitude show`, - Same as their apt-cache counterparts.
- `aptitude download` - Downloads a .deb file for a given package into the current directory.

Aptitude also includes *safe upgrading*, meaning it doesn't remove existing packages, as well as *holding*, which prevents the system from upgrading specific packages.

/etc/apt/sources.list

The file `/etc/apt/sources.list` controls repositories from which APT constructs its database. This file contains lines in the following format:

```
deb location-of-resources distribution component(s)
```

Here are some examples:

```
deb http://mirrors.linode.com/debian/ jessie main contrib
deb http://www.deb-multimedia.org jessie main non-free
```

The first line specifies the Linode mirror for the Debian 8 (code named Jessie) Linux distribution, as well as the main and contributed components. The next line specifies the deb-multimedia.org repository for Jessie, which provides some multimedia packages unavailable in the main repositories for licensing reasons, and its main and non-free components.

In general, one does not want to add new entries to `sources.list` without a lot of scrutiny and diligence, as updating the package cache with additional repositories and running upgrades can sometimes result in the installation of broken packages, unmet dependencies, and system instability. In Debian systems, downgrading is often difficult.

For Debian systems, the repository names can either refer to the distribution code name (e.g., jessie for current-stable, stretch for testing, sid for unstable, wheezy for old-stable) or to a specific branch (e.g., oldstable, stable, testing, unstable).

The component section of the line divides the repository based on how much support the developers of the operating system are able to offer for the contained packages (e.g. main vs. contrib), or if the software is considered “free-software” or simply freely-distributable (e.g., non-free).

The layout of sources.list is a bit different in Ubuntu systems. The lines are in the same format but the names of the distributions and components are different:

- Ubuntu versions have a different naming scheme. Version 14.04 is named “trusty” in sources.list , 15.10 is “wily,” and 16.04 is “xenial.” These names follow an alphabetical pattern.
- Ubuntu components are: “main” and “restricted” for supported free and non-free packages; “universe” and “multiverse” for unsupported free and non-free software.

Using dpkg

Apt-get and apt-cache are merely frontend programs that provide a more usable interface and connections to repositories for the underlying package management tools called dpkg and debconf . These tools are quite powerful, and fully explaining their functionality is beyond the scope of this document. However, a basic understanding of how to use these tools is useful. Some important commands are:

- dpkg -i package-file-name.deb - Installs a .deb file.
- dpkg --list search-pattern - Lists packages currently installed on the system.
- dpkg --configure package-name(s) - Runs a configuration interface to set up a package.
- dpkg-reconfigure package-name(s) - Runs a configuration interface on an already installed package.

Fedora and CentOS Package Management

Fedora and CentOS are closely related distributions, being upstream and downstream (respectively) from Red Hat Enterprise Linux (RHEL). Their main differences stem from how packages are chosen for inclusion in their repositories.

CentOS uses `yum`, *Yellowdog Updater, Modified*, as a front end to interact with system repositories and install dependencies, and also includes a lower-level tool called `rpm`, which allows you to interact with individual packages.

Starting with version 22, Fedora uses the `dnf` package manager instead of YUM to interact with `rpm`. DNF supports many of the same commands as YUM, with some slight changes.

Note: Many operating systems aside from RedHat use `rpm` packages. These include OpenSuSE, AIX, and Mandriva. While it may be possible to install an RPM packaged for one operating system on another, this is not supported or recommended, and the results of this action can vary greatly.

Yellow Dog Updater, Modified (YUM)

The YUM tool was developed for the Yellow Dog Linux system as a

replacement for the Yellow Dog Updater (YUP). RedHat found the YUM tool to be a valuable addition to their systems. Today, YUM is the default package and repository management tool for a number of operating systems.

You can use the following commands to interact with YUM:

- yum install package-name(s) - Installs the specified package(s) along with any required dependencies.
- yum erase package-name(s) - Removes the specified package(s) from your system.
- yum search search-pattern - Searches the list of package names and descriptions for packages that match the search pattern and provides a list of package names, with architectures and a brief description of the package contents. Note that regular expression searches are not permitted.
- yum deplist package-name(s) - Lists all of the libraries and modules that the named package depends on, along with the names of the packages (including versions) that provide those dependencies.
- yum check-update - Refreshes the local cache of the yum database so that dependency information and the latest packages are always up to date.
- yum info package-name(s) - The results of the info command provides the name, description of the package, as well as a link to the upstream home page for the software, release versions and the installed size of the software.
- yum reinstall package-name(s) - Erases and then downloads a new copy of the package file and re-installs the software on your system.
- yum localinstall local-rpm-file - Checks the dependencies of a local .rpm file and then installs it.
- yum update optional-package-name(s) - Downloads and installs all updates including bug fixes, security releases, and upgrades, as provided by the distributors of your operating system. Note that you can specify package names with the update command.
- yum upgrade - Upgrades all packages installed in your system to the latest release.

/etc/yum.config

The file located at /etc/yum.conf provides system-wide configuration options for YUM, as well as information about repositories. Repository information may also be located in files ending in .repo under /etc/yum.repos.d .

The options in the [main] stanza don't need modification, though you may set alternate logging and cache locations for the database by adding the following lines:

/etc/yum.conf

```
1
2
logfile=/var/log/yum.log
cachedir=/var/cache/yum
```

Dandified YUM (DNF)

DNF is the modern extension of the YUM package manager. It retains much of the same command usage and functionality as YUM, with a number of improvements for newer operating systems. DNF was first introduced in Fedora 18 and became the default package manager with the release of Fedora 22.

- dnf install package-name(s) - Installs the specified package(s) along with any required dependencies. dnf install can also accept .rpm files in place of a package name, to install directly from a downloaded

RPM.

-
- dnf remove package-name(s) - Removes the specified package(s) from your system, along with any package(s) that depend upon them.
-
- dnf search search-pattern - Searches the list of package names and descriptions for packages that match the search pattern and provides a list of package names, with architectures and a brief description of the package contents. Note that regular expression searches are not permitted.
-
- dnf provides package-name(s) - Lists all of the libraries and modules that the named package depends on, along with the names of the packages (including versions) that provide those dependencies.
- dnf check-update - Refreshes the local cache of the DNF database so that dependency information and the latest packages are always up to date.
-
- dnf info package-name(s) - Provides the name and description of the package as well as a link to the upstream home page for the software, release versions, and the installed size of the software.
-
- dnf reinstall package-name(s) - Erases and then downloads a new copy of the package file and re-installs the software on your system.
-
- dnf upgrade optional-package-name(s) - Downloads and installs all updates including bug fixes, security releases, and upgrades for a specific package.
- dnf upgrade - With no arguments, upgrade upgrades all packages installed in your system to the latest release.
- dnf config-manager --add-repo example.repo Adds a .repo file as a DNF

repository.

- `dnf config-manager --set-enabled example-repo` Enables a DNF repository.
- `dnf config-manager --set-disabled example-repo` Disables a DNF repository.

/etc/dnf/dnf.conf

The `dnf.conf` file provides global configuration settings for DNF. If DNF `.repo` files are being added manually, instead of with `dnf config-manager`, they should be added to `/etc/yum.repos.d`.

RPM Package Manager (RPM)

YUM and DNF are simply front-ends to a lower-level tool called RPM, similar to `apt-get`'s relationship with `dpkg`. You will likely not need to interact with RPM very often, but there are a few commands that you may find useful.

The following commands should be run as root. The flags are expanded here, but the abbreviated syntax is also included:

- `rpm --install --verbose --hash local-rpm-file-name.rpm` or `rpm -ivh filename.rpm` - Installs an rpm from the file. `rpm` is also capable of installing RPM files from http and ftp sources as well as local files.
- `rpm --erase package-name(s)` or `rpm -e` - Removes the given package. Usually will not complete if `package-name` matches more than one package, but will remove more than one match if used with the `--allmatches` flag.
- `rpm --query --all` or `rpm -qa` - lists the name of all packages currently installed.
- `rpm --query package-name(s)` or `rpm -q` - allows you to confirm whether a given package is installed in your system.
- `rpm --query --info package-name(s)` or `rpm -qi` - displays the information about an installed package.
- `rpm --query --list package-name(s)` or `rpm -ql` - generates a list of files installed by a given package. This is complemented by:
- `rpm --query --file` or `rpm -q qf filename` - checks to see what installed package "owns" a given file.

You can use the following template to define a new stanza for a new repository, replacing the capitalized strings with your own values:

/etc/yum.repos.d/example.repo

```
[REPO-NAME]
name=REPOSITORY-NAME
mirrorlist=HTTP-ACCESSIBLE-MIRROR-LIST
#baseurl=BASE-URL-FOR-REPOSITORY
gpgcheck=BOOLEAN-VALUE-TO-VERIFY-REPOSITORY
gpgkey=FILE-PATH-TO-GPG-KEY
```

The following example is the default configuration for the “Base” repository in CentOS 7:

/etc/yum.repos.d/CentOS-Base.repo

```
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

Slackware Package Management

Packages in Slackware Linux are distributed as compressed tarballs, generally using gzip or lzma compression. These tarballs can be recognized by their suffixes, `.tgz` or `.txz`. This format includes a complete filesystem layout, as well as additional scripts to be run upon installation or removal of the software. Slackware packages do not offer dependency resolution information; this is generally viewed as allowing more flexibility and control.

Packages can also be built using SlackBuilds, shell scripts that compile source or repackage binary distribution packages for easy installation and removal on Slackware. These scripts have been adopted as a community standard by such sites as

Slackware includes `pkgtool` for local package management and `slackpkg` for remote installation of packages from official mirrors. For less interactive tasks, there are `installpkg`, `upgradepkg`, and `removepkg`.

Working With Packages Locally

The `pkgtool` program offers the ability to manage packages on the local system through a text-based menu interface. Each option is self-explanatory, from installing packages from the Current or an Other directory, to Removing installed packages, Viewing a list of files in a package, and running Slackware Setup scripts.

The package installation operations offer a list of packages in a menu, with the ability to choose which packages to install and which to leave alone. The package removal option offers a similar choice, with a list of installed packages. Viewing a package can be useful to determine what is in it. This information includes a description written by the creator of the package along with the expected list of files.

The Setup scripts options do not often apply to Linode, though there is a `netconfig` option that may be helpful to some users.

For those seeking a command-line approach, the `*pkg` commands are fairly straightforward in their use.

- `installpkg package-name(s)` - Installs a package from the current directory, or a pathname you specify. It accepts full filenames, as well as globs such as `*/*.t?z` for all tgz, tbz, tlz, or txz packages in all immediately adjacent directories.
- `upgradepkg package-name(s)` - Upgrades a package from the current directory, or a pathname you specify. If also accepts full filenames and globs like `installpkg`. Note that the `--install-new` flag can be passed to allow `upgradepkg` to operate like `installpkg` on packages that are not currently installed on the system.
- `removepkg package-name(s)` - Removes a package from the system. This command does not require a full filename, but can often operate with only the software name associated with the package.

Working With Packages Remotely

The `slackpkg` program is a recent addition to Slackware that allows official Slackware packages to be installed and upgraded using a remote FTP or

HTTP mirror. Before using Slackpkg, a mirror should be set in /etc/slackpkg/mirrors , and can be added or selected from the available list. Only one mirror can ever be active, and is chosen by uncommenting it (deleting the initial #).

While slackpkg offers a menu-based interface, it can also be run in a console-only mode by setting DIALOG=off in /etc/slackpkg/slackpkg.conf .

- slackpkg check-updates - Checks for new entries to the changelog on the remote mirror. This can be useful in a cron script to notify the system administrator of new patches.
- slackpkg update - Downloads updates to the Slackware changelog and file lists. This check is useful for finding security updates, and must be run before attempting to download updated software.
- slackpkg install-new - Looks for new packages. While rarely useful on a static release, this command should be run before others on machines running the current development release or when upgrading to a new release.
- slackpkg install package-name(s) - Looks for any packages matching the name given, and presents the user with a menu allowing the choice of installation. Note that all commands accepting a package name will also work with Slackware installation series, such as ap, d, or xap.
- slackpkg upgrade-all - Presents the user with a menu listing all packages on the remote mirror that do not match the versions currently installed on your system. While this will generally result in upgrades, outdated software can sometimes be listed as an upgrade, such as when changing to an outdated mirror, or using self-built packages to replace standard Slackware packages. One common area this occurs is using alienBOB's multilib glibc packages on Slackware 64-bit. Upon choosing and confirmation of upgrades, the chosen packages will be downloaded and upgraded.
- slackpkg upgrade package-name(s) - Searches for any packages matching the name given, and presents a menu to allow upgrades.
- slackpkg clean-system - Presents a menu listing all packages on the local system that are not present on upstream mirrors. This includes self-built packages, packages installed from a third-party

source, and packages that were once included in Slackware, but have since been removed.

- slackpkg remove package-name(s) - Attempts to find any installed packages matching the name given, and presents the user with a menu allowing the choice of removal.
- slackpkg reinstall package-name(s) - Reinstalls the given package. This is useful if certain files in that package have been corrupted.
- slackpkg search package-name(s) - Searches for the given package name, and displays matching packages as well as installation status.
- slackpkg file-search filename - Searches installed and remote package descriptions for the given filename, and displays matching packages as well as installation status.
- slackpkg blacklist package-name(s) - Adds the given package to the blacklist located in /etc/slackpkg/blacklist . Blacklisted packages will not be installed, upgraded, or removed by slackpkg.
- slackpkg info package-name(s) - Displays standard information about the given package.

SlackBuilds, sbopkg, and Third-Party Packages

Slackware does not offer as large a selection of official software as some other more community-oriented distributions, so it can include dubious third-party packages. For this reason, the use of third-party software repositories in Slackware is generally discouraged. Third-party package management tools such as slapt-get are also frowned upon, as they have a reputation for disrupting systems.

The Slackware community has produced SlackBuilds.org, offering SlackBuild scripts for a growing volume of third-party software. These scripts are heavily vetted for integrity and proper operation. Dependencies are noted in README files, and all builds are verified to work as advertised in a clean build environment. Local compilation also verifies that packages for your machine will work on your machine.

The sbopkg program, like pkgtool , offers a text-based menu interface. There is also a Sync option, which ensures that you are working with the latest version of all SlackBuilds. SlackBuilds can be browsed or searched for, as can their respective Changelogs. Sbopkg offers an Updates option, which compares

local versions of your packages to remote versions of the SlackBuilds. Unlike `slackpkg`, `sbopkg` will not overwrite a newer package than what's on the server by default. In addition, `sbopkg` offers the ability to manage the order in which SlackBuilds are built using a queue system, and allows the user to make changes to the SlackBuild locally. You can also view which SlackBuilds packages are installed, choose a different repository version to work with, or look for updates to `sbopkg` itself.

Package Management in Arch Linux with Pacman

Arch Linux uses binary packages in a `.tar.xz` format, and also provides a "ports" build system that facilitates building packages.

Arch Linux runs on a *rolling release* schedule, which means packages are added to the main repository when they (and their dependencies) are ready for production. This means that there aren't release versions of Arch, as all systems, once upgraded, are equivalent.

Therefore, administrators of Arch Linux must consider the output of `pacman` carefully before agreeing to upgrade or update any packages.

Pacman

The `pacman` tool is very powerful, but it is also very simple. There are three core commands for basic package management:

- `pacman --query package-name(s)` or `pacman -Q` - Searches the package

database for a package name and version number.

- `pacman --sync package-name(s)` or `pacman -S` - Installs new packages, downloads new content for the database and/or upgrades the system, depending on the options and the named package or packages.
- `pacman --remove package-name(s)` or `pacman -R` - Removes the named package or packages.

Note that the terse flags are all uppercase and case-sensitive. These terse flags are often combined with additional flags for additional functionality. Here are some examples with brief descriptions:

- `pacman -Qi package-name(s)` - Displays information about a given package, including dependency information, the date of the package, a link to the upstream source and other useful information.
- `pacman -Qo filename` - Outputs the version number and name of the package that "owns" a given file.
- `pacman -Qs package-name(s)` - Searches among the installed packages for a given package.
- `pacman -Qu` - Lists out-of-date installed packages that are in need of an update.
- `pacman -Sy` - Triggers a database refresh, and synchronizes the local database with the remote database.
- `pacman -Su` - Triggers a full system update and downloads new packages to upgrade the system. The update and refresh command can (and should) be run together, as in: `pacman -Syu` .
- `pacman -Sc` - Removes uninstalled packages from the cache and attempts to clean up old copies of the repository database.
- `pacman -S --ignore package-name(s)` - Ignores upgrades to a given package or list of packages.
- `pacman -Rs` Removes a package and its dependencies, as long as the dependencies are not needed and were not explicitly installed by the user. This command is the inverse of `pacman -S` .
- `pacman -Ru` Removes packages that are unneeded by any other packages.

Configuration Options

The configuration for `pacman` is defined in the `/etc/pacman.conf` file, while the addresses of the repository mirrors are contained in `/etc/pacman.d/mirrorlist`. The mirror list was created and prioritized during the installation process and you probably will not need to alter this.

The options in the stock `/etc/pacman.conf` are documented in comments, and are beyond the scope of this document. You may access the manual page for this configuration file with the command `man pacman.conf`.

While it is unlikely that you will need to modify the default `pacman.conf` for most installations, you can change default installation and logging directories and specify packages to be held back from upgrades.

If you need to add an additional third-party repository, add a repository stanza:

/etc/pacman.conf

```
[REPOSITORY-NAME]
Server = SERVER-LOCATION
Include = REPOSITORY-LIST
```

The `Server =` and `Include =` lines are both optional, and the order indicates their priority. By default, the testing repository is disabled, which is wise if you're planning to use the system for production work; however, if you need bleeding-edge packages, uncomment those lines.

The Arch Build System (ABS)

The Arch Build System allows users to compile and install software not included in the Arch repository. This brief guide outlines the steps to building a package using the ABS.

NoteAll commands explained here should be run as root unless otherwise specified.

Begin by installing the `abs` framework and the `base-devel` packages:

```
pacman -Sy abs base-devel
```

Edit `/etc/abs.conf` so that the `REPOS` line indicates the proper repositories. Note, repositories prefixed with a bang (!) are disabled. The line might look like:

```
REPOS=(core extra community !testing)
```

To create a local ABS tree in `/var/abs`, run the `abs` command as root. You may now browse `/var/abs`, which contains a representation of the package

collection with folders representing each repository, category, and piece of software.

Arch recommends that you create a build directory at another location, such as `~/abs/` , where actual building will occur.

Begin the build process by copying the files from the ABS tree into your build directory as a non-root user:

```
cp -r /var/abs/REPO/PACKAGE ~/abs
```

Change to the package's directory:

```
cd ~/abs/PACKAGE
```

You have the option of modifying the `PKGBUILD` file. There's a build shell function that you can use to add additional patches to the files if you have modifications to the software or the build process. That shell function generally looks like:

~/abs/PACKAGE/PKGBUILD

```
build() {
  cd $startdir/src/$pkgname-$pkgver.orig

  patch -Np1 -i
  $startdir/src/${pkgname}_${pkgver}-$_patchlevel.diff || return 1

  ./configure --prefix=/usr
  make || return 1
  make install
}
```

To build the package, use the following command as a non-root user:

```
makepkg -s
```

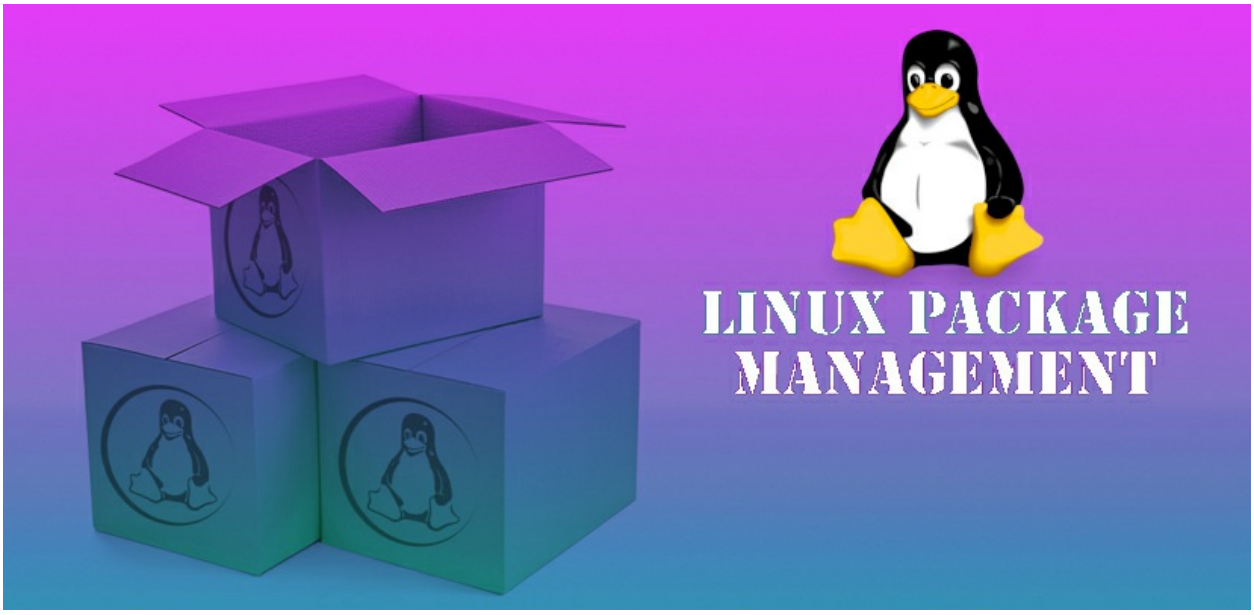
The `makepkg` command creates a package that contains dependency information. As root, issue the following command:

```
pacman -U PACKAGE.pkg.tar.xz
```

Make sure to replace `PACKAGE` with the full package name exactly as it

appears in the file system. Arch will now install the package and any required dependencies.

Because ABS downloads source versions of the PKGBUILD file as it creates the package - sometimes checking out a copy of the source code from the version control system - we don't recommend removing files from the build directory hierarchy.



Gentoo Linux Package Management

Gentoo provides its entire operating system in source format. These source packages, in concert with ebuild scripts, provide a package management system that borrows and builds on many concepts from the BSD's "portage" system.

Like Arch, the Gentoo project produces new versions of Gentoo Linux on a rolling release cycle.

This section addresses common package management tasks and functions using the emerge front end for the portage system. We encourage you to install the "gentoolkit" to provide additional package management tools, such as equery . Install this package with the following command:

```
emerge app-portage/gentoolkit
```

Emerge/Portage Commands

- emerge --sync - Updates the local copy of the portage tree, so that your local system can download and install the latest version of the

software.

- emerge --update --deep world - Checks and updates all packages on the system to the latest version. This should be run regularly to avoid falling behind on a critical update.
- emerge --search keyword or emerge -s keyword - Searches the names of all of the packages for the given keyword. This command accepts a regular expression as the keyword argument.
- emerge --searchdoc keyword or emerge -S keyword - Searches the full description for a given keyword. This command accepts a regular expression as the keyword argument.
- emerge package-name(s) - Installs the specified package or packages.
- emerge -u package-name(s) - Updates the specified package to the latest version. Using the flag -uD also updates dependencies.
- emerge --depclean package-name(s) or emerge -c package-name(s) - Removes the specified package or packages.
- emerge --depclean - Removes packages that are orphaned. This means removal of all packages that weren't explicitly installed and do not depend upon any specific package. We recommend that you run it with the --pretend option before running this command on a production system.
- emerge -evp --deep world - Lists all of the packages currently installed on the system.
- equery depends package-name(s) - Lists all of the packages that depend upon the specified package.
- equery files package-name(s) - Lists all of the files "owned" by a package.
- equery belongs filename - Lists the package that "owns" a particular file.

USE Flag

To discover which USE flags are available for a given package, issue the following command:

equery uses package-name

The equery command depends on the gentoolkit package. This will provide

information about what USE flags are available and which have been installed. To specify additional USE flags:

```
echo "package-name USE-flags" >> /etc/portage/package.use
emerge package-name
```

This will install the specified package with the appropriate options enabled.

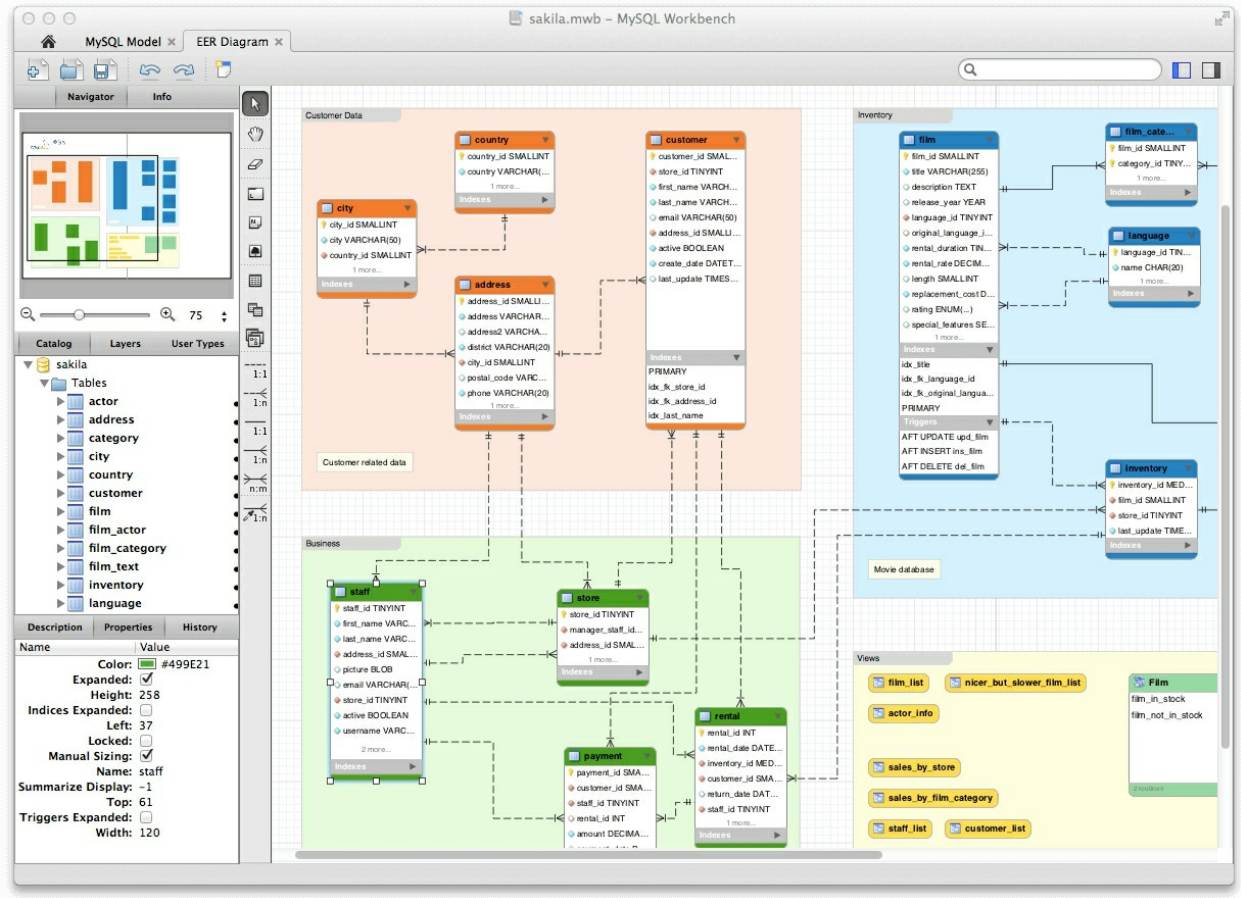
GUI Tools for Linux System Administrators

Here are the 10 best GUI tools for Linux System Administrators.

1. MySQL Workbench

MySQL Workbench is arguably the most popular database administration application across OS platforms. With it, you can design, develop, and manage **MYSQL** databases using a wide variety of tools that allow you to work both locally and remotely.

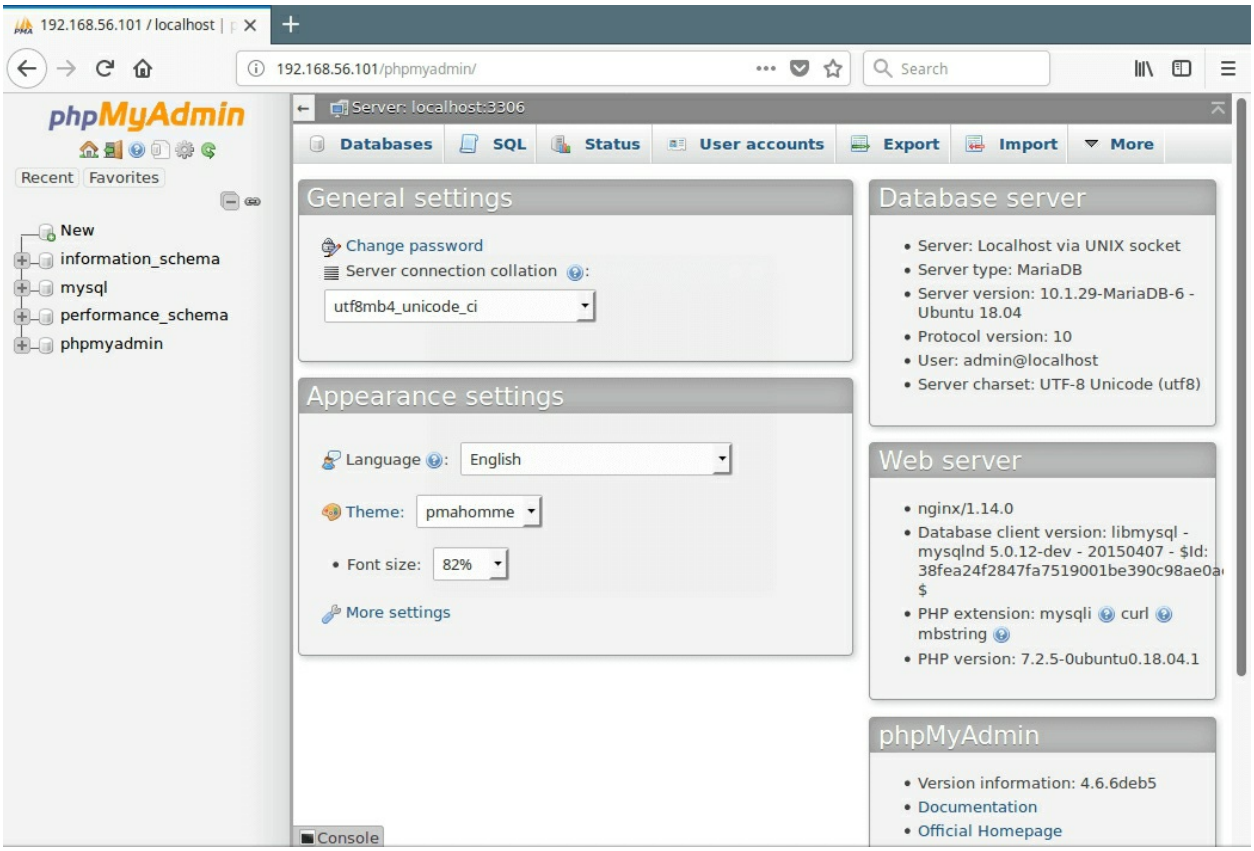
It features the ability to migrate Microsoft Access, Microsoft SQL Server, PostgreSQL, Sybase ASE, and other RDBMS tables, objects, and data to MySQL among other capabilities.



2. phpMyAdmin

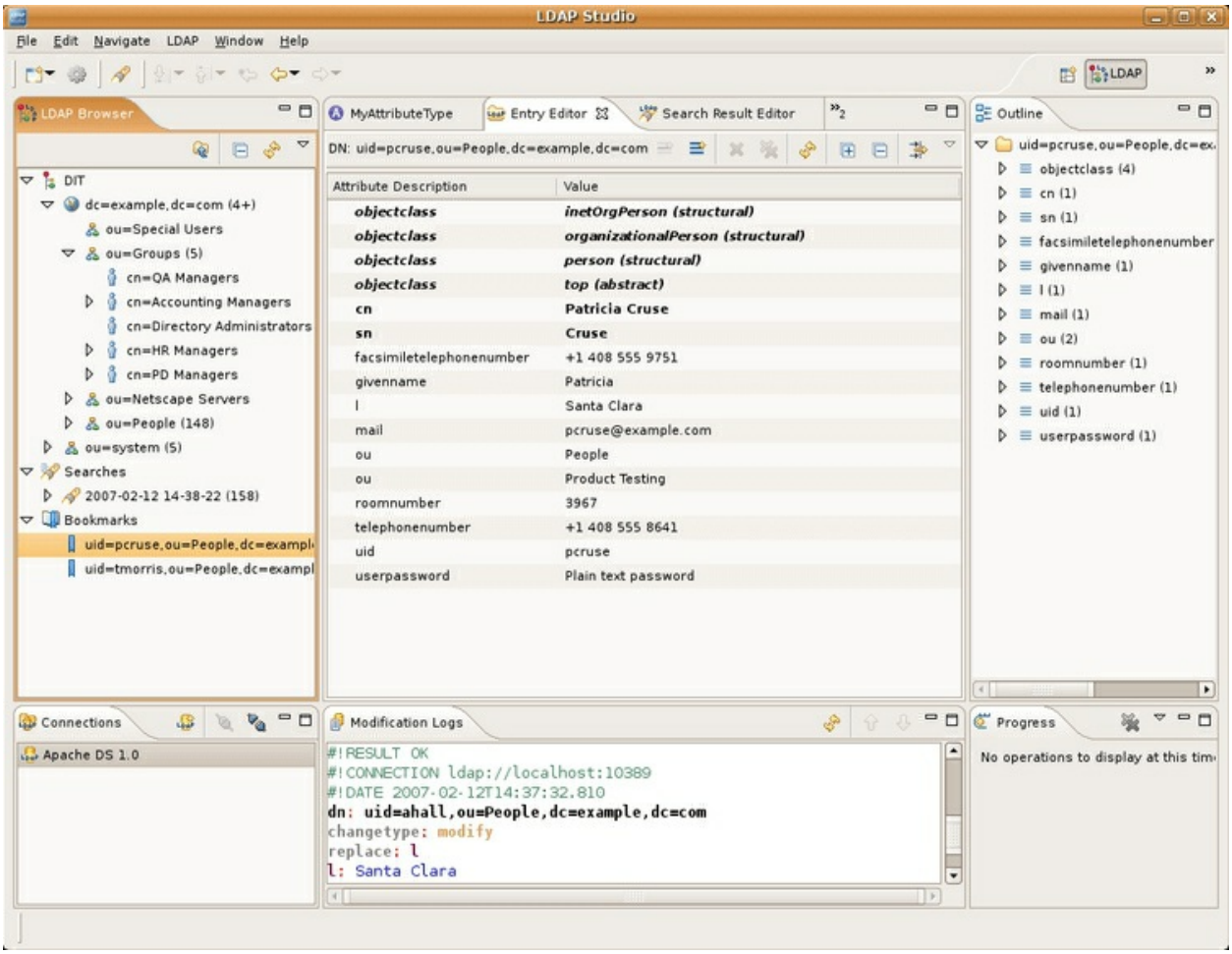
phpMyAdmin is a free and open-source PHP-based web app that allows you to create and manage MySQL databases using a web browser.

It is not as robust as **MySQL Workbench** but can also be used to perform various database administration tasks in a more user-friendly method – one of the reasons why it is a go-to app for students and beginner system admins.



3. Apache Directory

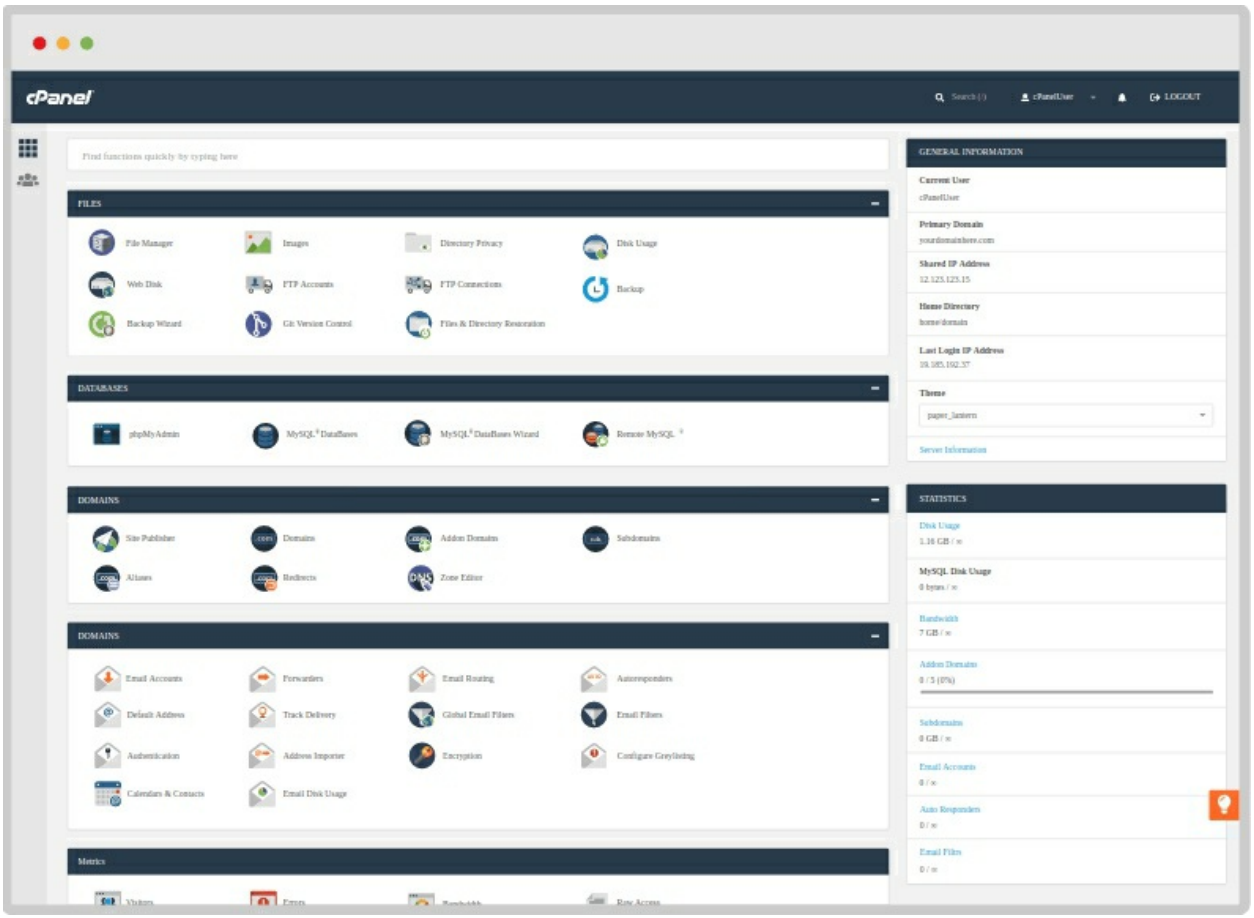
Apache Directory is an **Eclipse RCP** application designed for **ApacheDS** but it can also work as an LDAP browser, LDIF, ApacheDS, and ACI editors, among other functions.



4. cPanel

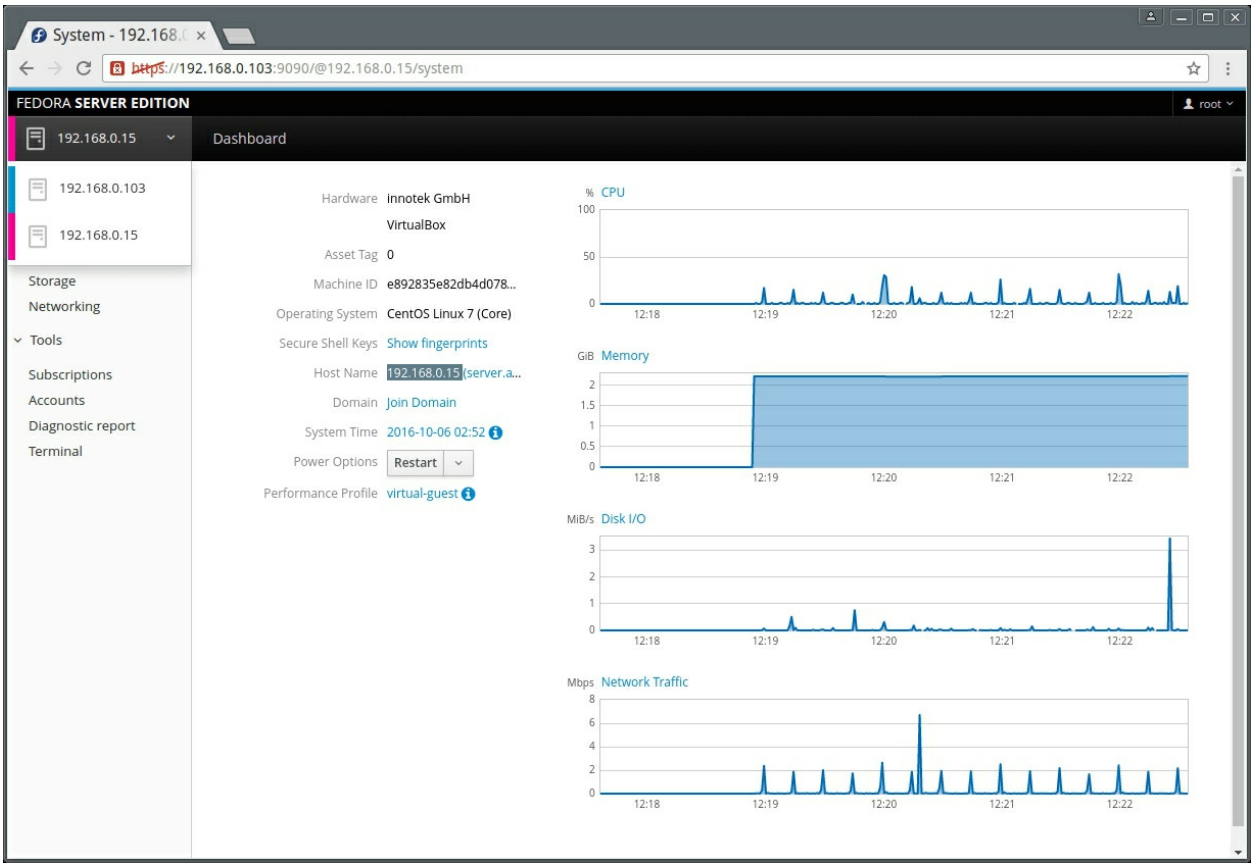
cPanel is arguably the best web-based administration tool ever. With it, you can manage websites, domains, apps and app files, databases, logs, mail, server security, etc.

cPanel is neither free nor open-source but it is worth every penny.



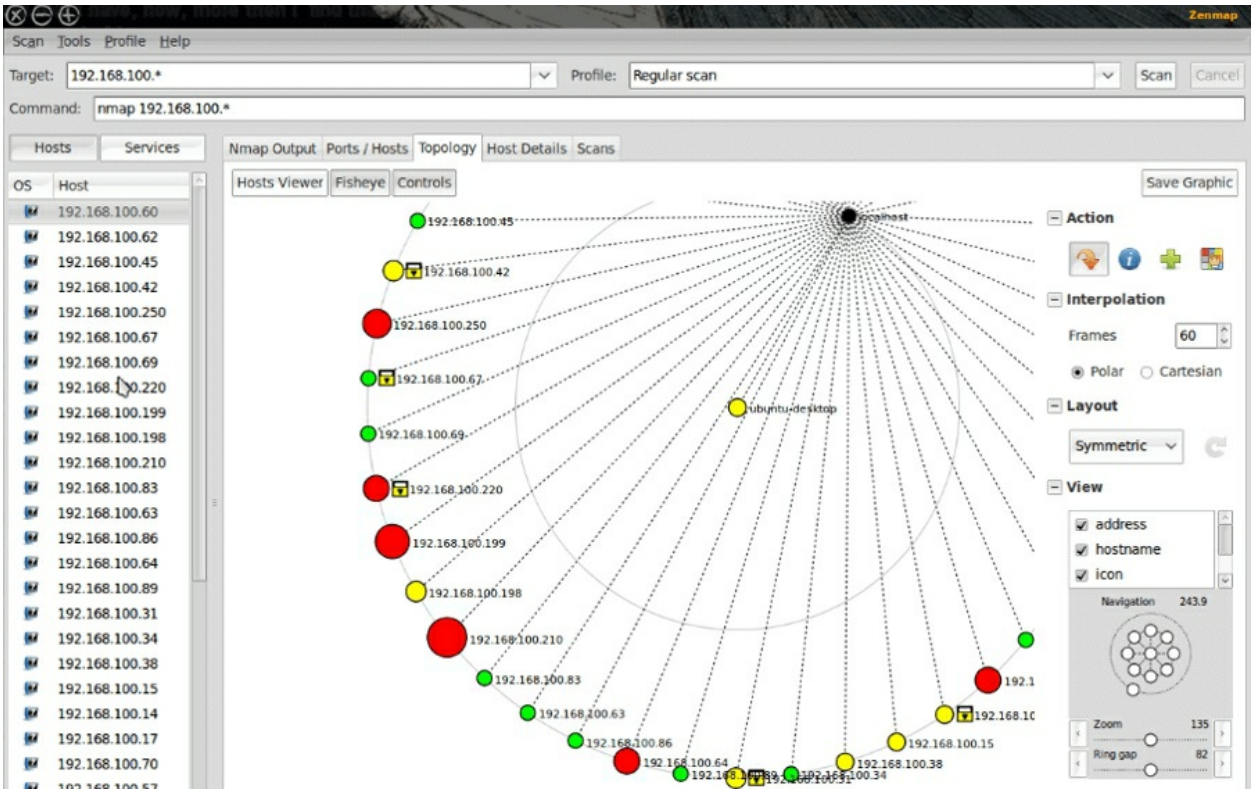
5. Cockpit

Cockpit is an open-source easy-to-use web-based server manager developed by Red Hat to be efficient at monitoring and administering several servers at the same time without any interference.



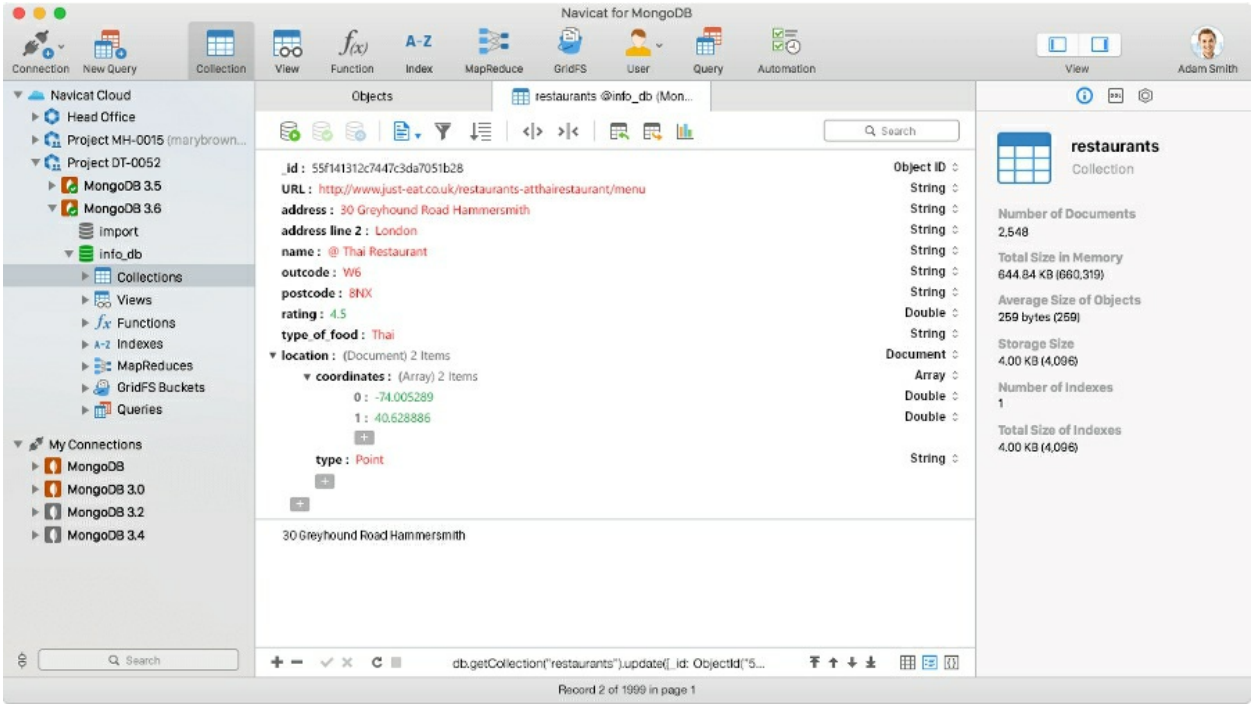
6. Zenmap

Zenmap is a free and open-source whose main purpose is to scan for and troubleshoot network issues. Being the official Nmap Security Scanner GUI, it is designed to be easily used by beginners while still providing advanced tools for experts.



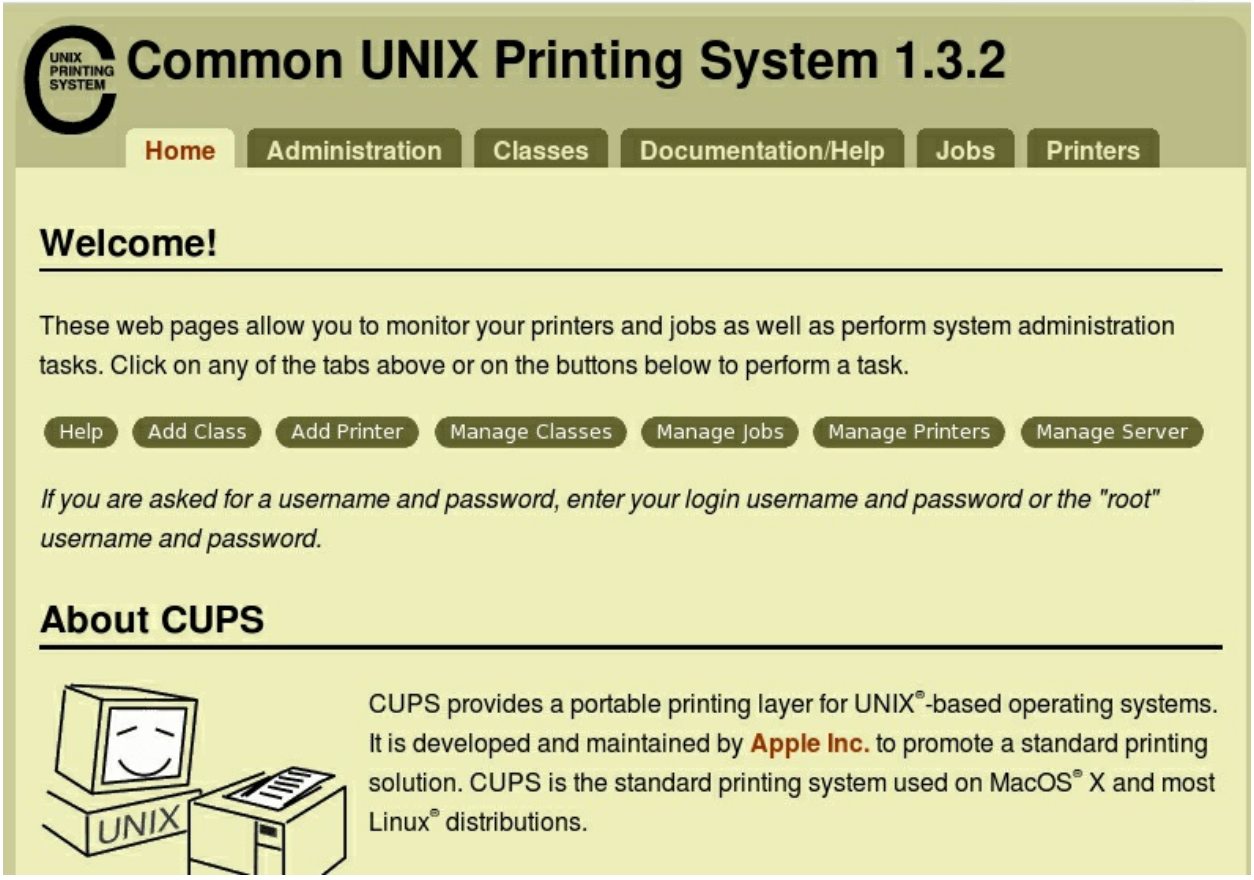
7. YaST

YaST (Yet another Setup Tool*)* can be used to setup entire systems whether they are hardware, networks, system services, and security profiles all from the YaST Control Center. It is the default configuration tool for enterprise-grade SUSE and openSUSE and ships with all SUSE and openSUSE platforms.



8. CUPS

CUPS (Common Unix Printing System) is a printer service built by Apple Inc. for macOS and other UNIX-like OSes. It has a web-based GUI tool with which you can manage printers and printing jobs in both local and network printers using the Internet Printing Protocol (IPP).



9. Shorewall

Shorewall is a free and open-source GUI for creating and managing blacklists, configuring firewalls, gateways, VPNs, and controlling traffic. It takes advantage of the Netfilter (iptables/ipchains) system built into the Linux kernel to provide a greater level of abstraction for describing rules using text files to manage intricate configuration schemes.

```
Terminal - rk@debian: ~/linux-3.12
File  Edit  View  Terminal  Go  Help
#
# Shorewall version 4.0 - Sample Rules File for two-interface configuration.
# Copyright (C) 2006,2007 by the Shorewall Team
#
# This library is free software; you can redistribute it and/or
# modify it under the terms of the GNU Lesser General Public
# License as published by the Free Software Foundation; either
# version 2.1 of the License, or (at your option) any later version.
#
# See the file README.txt for further details.
#------------------------------------------------------------------------------
# For information about entries in this file, type "man shorewall-rules"
###############################################################################
###############################################################################
##
#ACTION         SOURCE          DEST            PROTO   DEST    SOURCE          ORIGINAL
      RATE            USER/   MARK    CONNLIMIT       TIME         HEADERS         SWITCH
#                                                       PORT    PORT(S)         DEST
      LIMIT           GROUP
#SECTION ALL
#SECTION ESTABLISHED
#SECTION RELATED
SECTION NEW

#       Don't allow connection pickup from the net
#
Invalid(DROP)   net             all
#
#       Accept DNS connections from the firewall to the network
#
```

10. Webmin

Webmin is a web-based admin tool with which you can perform virtually all sysadmin tasks on a server including creating user accounts and databases as well as configuring and managing disk quota, PHP, MySQL, and other open source apps. its functionality can also be extended using any of the many 3rd-party modules available online.

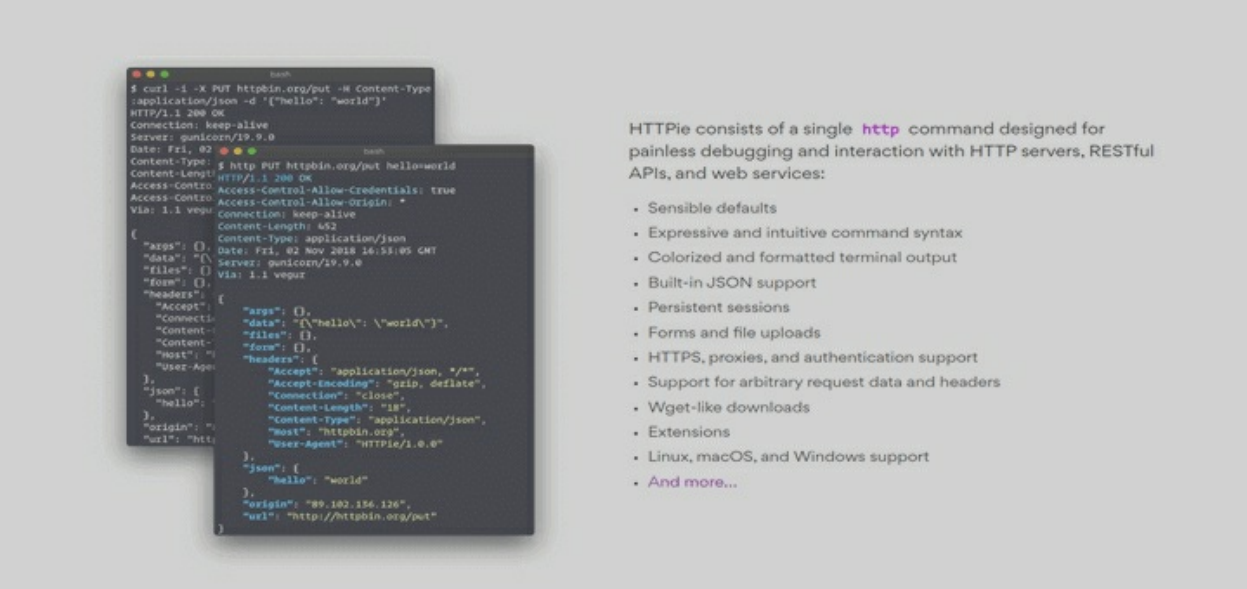
Webmin 1.740 on tecmin
192.168.0.11:10000
Login: root
Webmin
Backup Configuration Files
Change Language and Theme
Webmin Actions Log
Webmin Configuration
Webmin Servers Index
Webmin Users
System
Bootup and Shutdown
Change Passwords
Disk and Network
Filesystems
Filesystem Backup
Log File Rotation
MIME Type Programs
PAM Authentication
Running Processes
Scheduled Cron Jobs
Software Package Updates
Software Packages
System Documentation
System Logs
Users and Groups
Servers
Others
Networking
Hardware
Cluster
Un-used Modules
Search:
Module Config
Webmin Configuration
Webmin 1.740
IP Access Control
Ports and Addresses
Logging
Proxy Servers and Downloads
User Interface
Webmin Modules
Operating System and Environment
Language
Index Page Options
Upgrade Webmin
Authentication
Two-Factor Authentication
Reassign Modules
Edit Categories
Module Titles
Webmin Themes
Trusted Referrers
Anonymous Module Access
File Locking
Mobile Device Options
Blocked Hosts and Users
Background Status Collection
Advanced Options
Debugging Log File
Web Server Options
Webmin Scheduled Functions
Sending Email
SSL Encryption

Best Command Line HTTP Clients for Linux

1. HTTPie

HTTPie is an open source command line HTTP client for interacting with HTTP servers, web services, and RESTful APIs. HTTPie pronounced "**aych-tee-tee-pie**" and its aim is to simplify human interaction with web services via the command line, so most people use it for

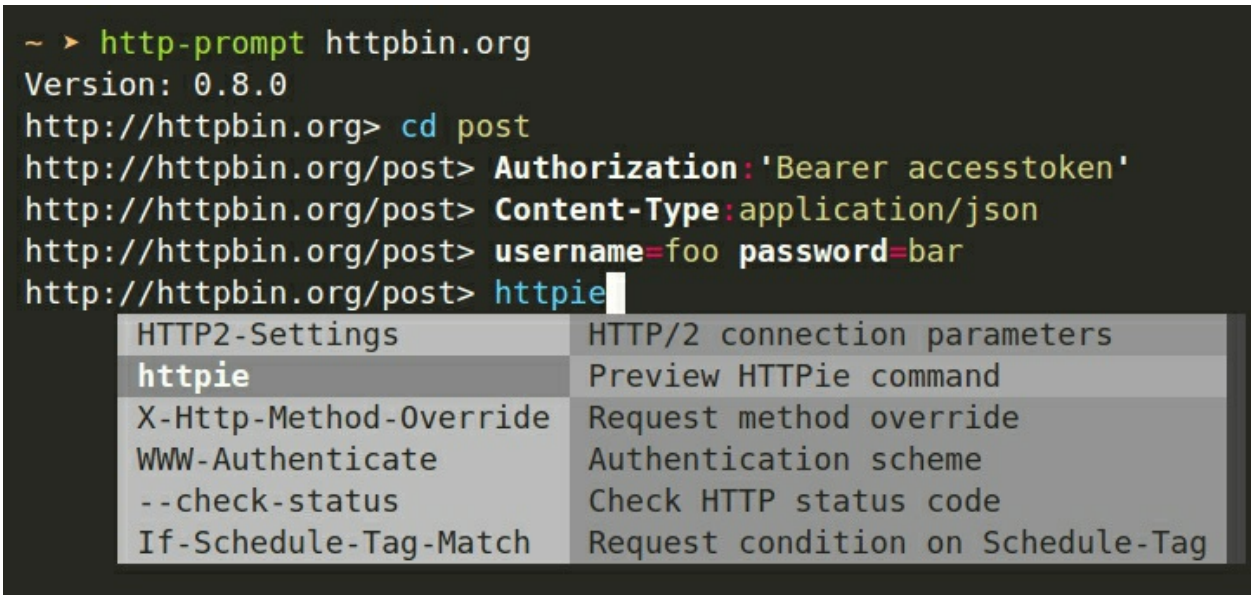
Its other features include a type-based colorized syntax, custom headers, persistent sessions, support for plugins, built-in support for JSON, etc.



HTTPie Linux HTTP Client

2. HTTP Prompt 2

HTTP Prompt is an interactive command-line HTTP client built on prompt_toolkit and **HTTPie** with **20+** themes. Its feature highlights include auto-complete, syntax highlighting, auto cookies, Unix-like pipelines, compatibility with HTTpie, http-prompt persists in-between sessions, and OpenAPI/Swagger integration.



HTTP Prompt – Command Line HTTP Client for Linux

3. Curl

is an open source command line utility and library for transferring files over a network using URL syntax over any of several supported protocols including SCP, SMTPS, HTTPS, IMAP, LDAP, POP3, etc.

Curl is a very popular utility used in not only terminals and scripts to transfer data but also in routers, printers, tablets, mobile phones, set-top boxes, audio equipment, media players, etc. It has a long feature list which includes support for IPv6 and socks5, custom output results after completion, no URL length limit, asynchronous name resolving.

4. Wget

Wget is an open source command-line utility for retrieving content from web servers via HTTP proxies as well as HTTP, HTTPS, and FTP protocols. Its function is recursive downloading which entails that it follows the links in HTML pages and create local versions of remote websites.

Wget boasts several features including the ability to function excellently even when your network connection is slow or unstable, support for HTTP proxies and cookies, resume aborted downloads using REST and RANGE APIs, NLS-based message files for different languages, etc.

5. Aria2

is a lightweight open source command-line download utility with support for HTTP & HTTPS, FTP & SFTP, Metalink, and BitTorrent. Its features include automatic validation for files like **BitTorrent**, parallel file downloads from HTTP(S)/(S)FTP and BitTorrent at the same, Ntrc support, disk caching to reduce disk activity, IPv6 support with Happy Eyeballs, etc.

Best IP Address Management Tools for Linux

IPAM software provides you with an overview of your network, gives you the opportunity to strategically plan your network growth and gives you the ability to provide more reliable service and reduce the number of manual administration tasks.

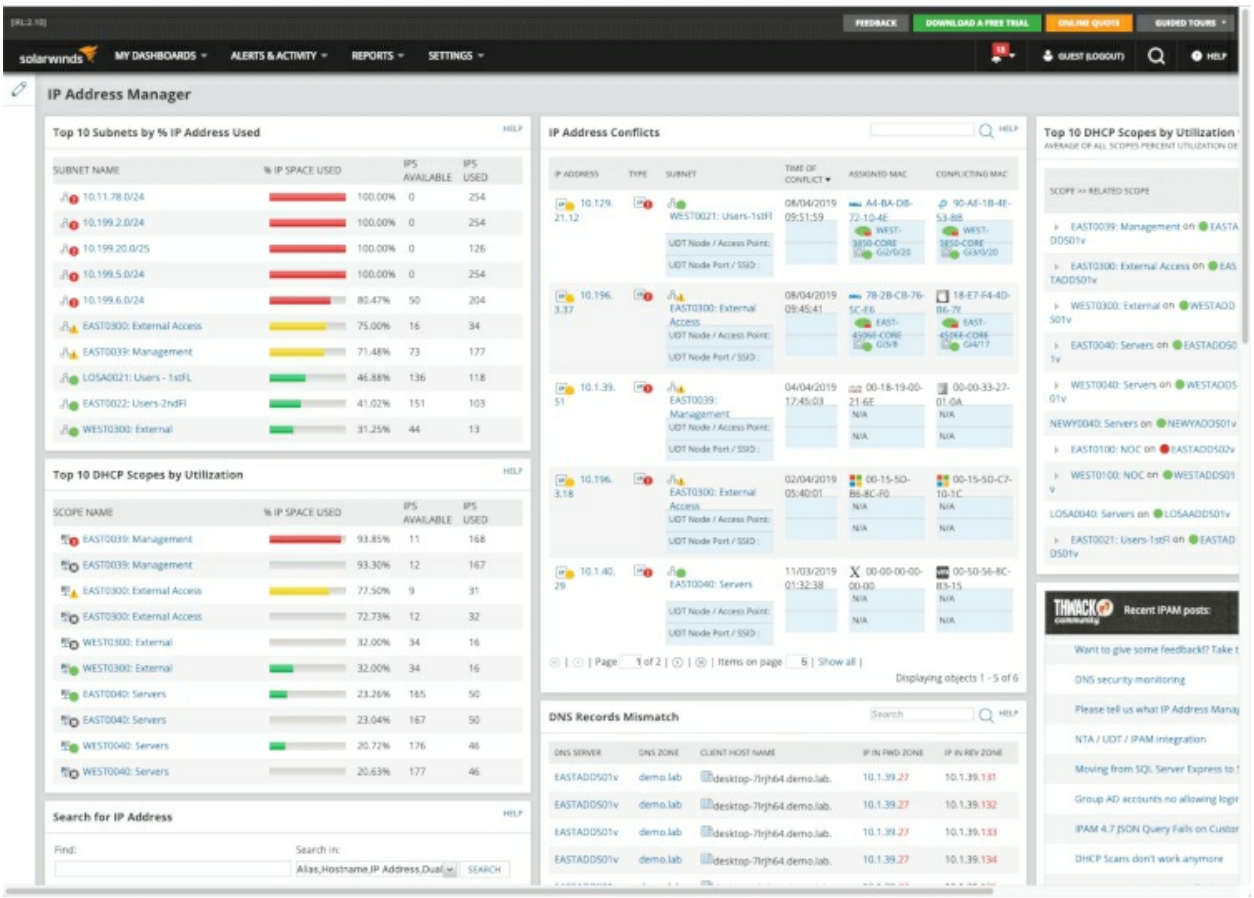
In this article, we are going to review some of the best **IPAM** software that you can use to manage IP addresses.

Solarwinds IPAM

SolarWinds is one of the more well known automated IP address management **IPAM** software in our list, that comes with features such as:

- Automated IP address tracking
- DHCP, DNS IP address management
- Alerting and troubleshooting and reporting
- Multi-vendor support
- Integration with VMWare
- API support for integration with third-party software
- Automation of IP address requests

Solarwinds IPAM features can be easily implemented, its interface is easy to understand and navigate. The dashboard allows you to monitor your entire network from a single place:



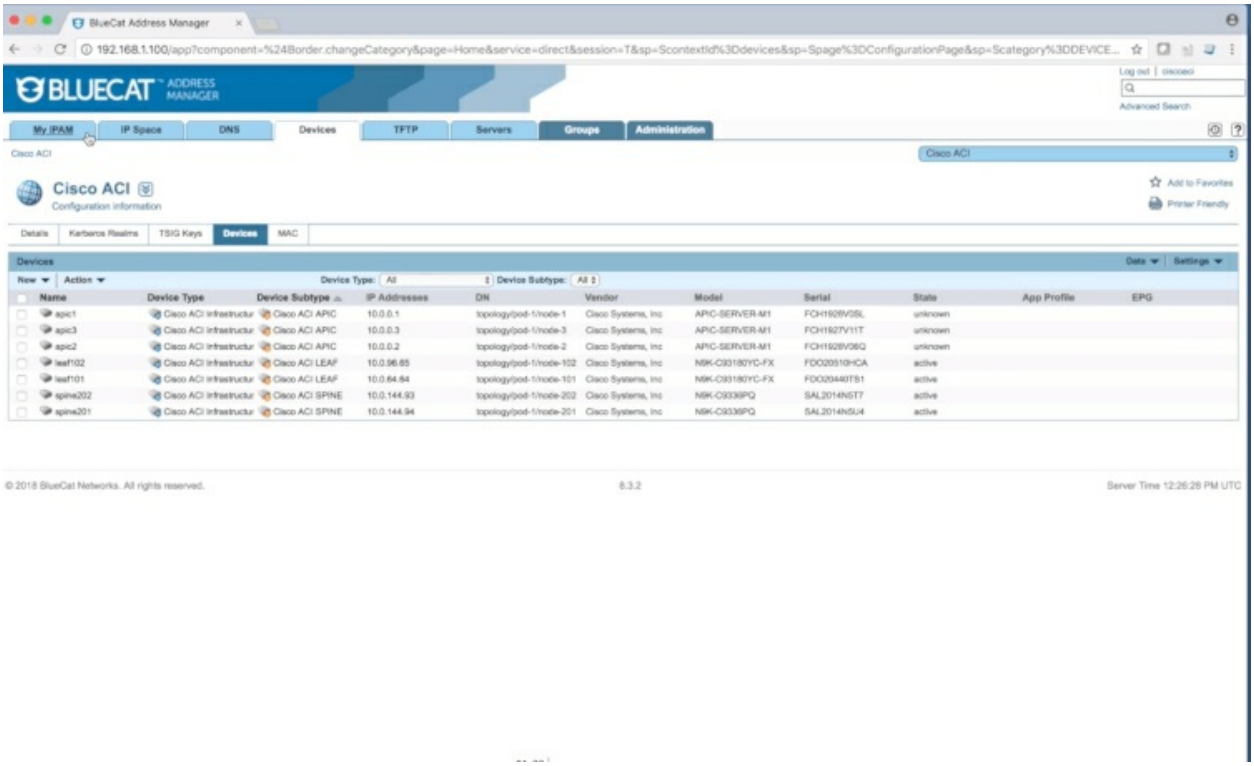
BlueCat Address Manager

Bluecat Address Manager is a powerful tool, that helps you control your complex and dynamic network. You can reduce the manual work and reduce network admin time thanks to its automation features.

BlueCat Address manager gives you:

- Efficient network manager through role-based access control, quick actions and workflow, tracking and auditing.
- Ability to plan and model your network growth through templates and flexible configurations.
- A powerful centralized management interface.
- Integration of IP addresses, DNS and DHCP data.

- Full support of IPv6.
- Network automation through scheduling and on-demand deployments, web services API, automatic network discovery and network reconciliation policies.



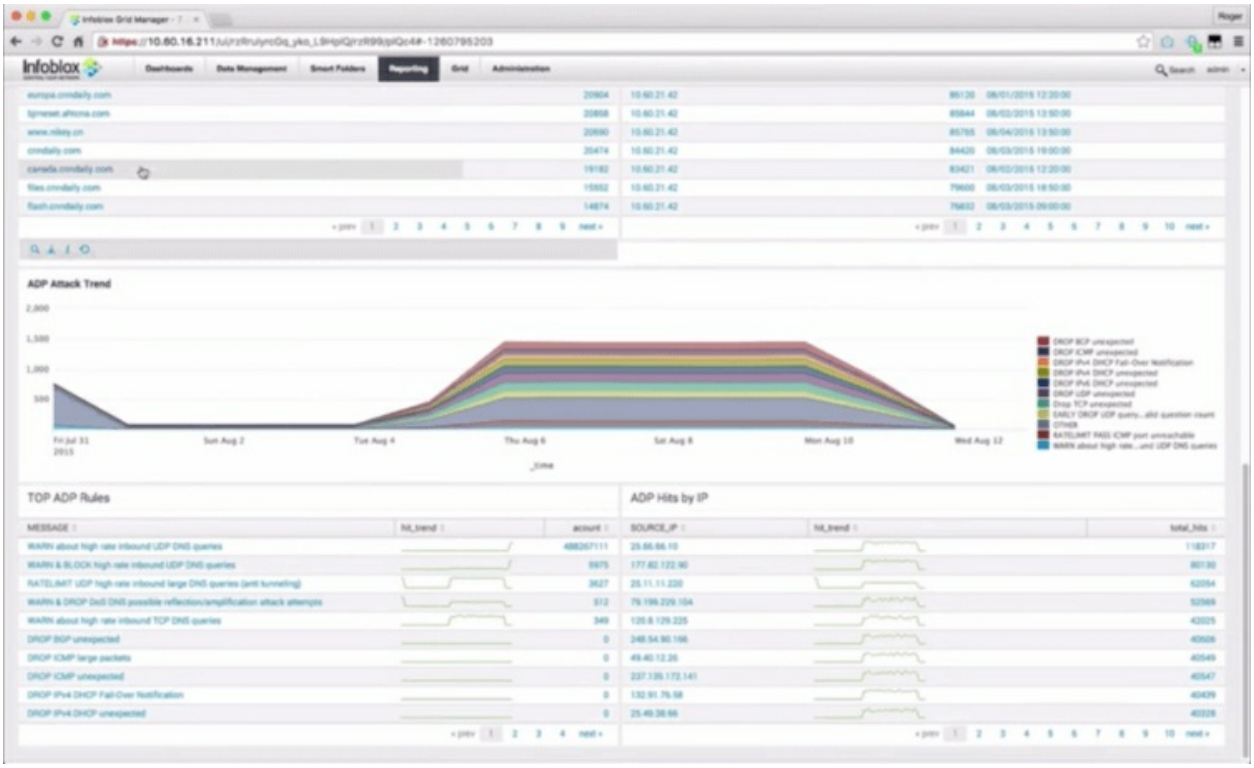
Infoblox

Our next IPAM tool in the list is Infoblox IPAM, which provides enterprise-grade automated network services for hybrid, public and private clouds, and virtualized environments.

Infoblox IPAM gives you:

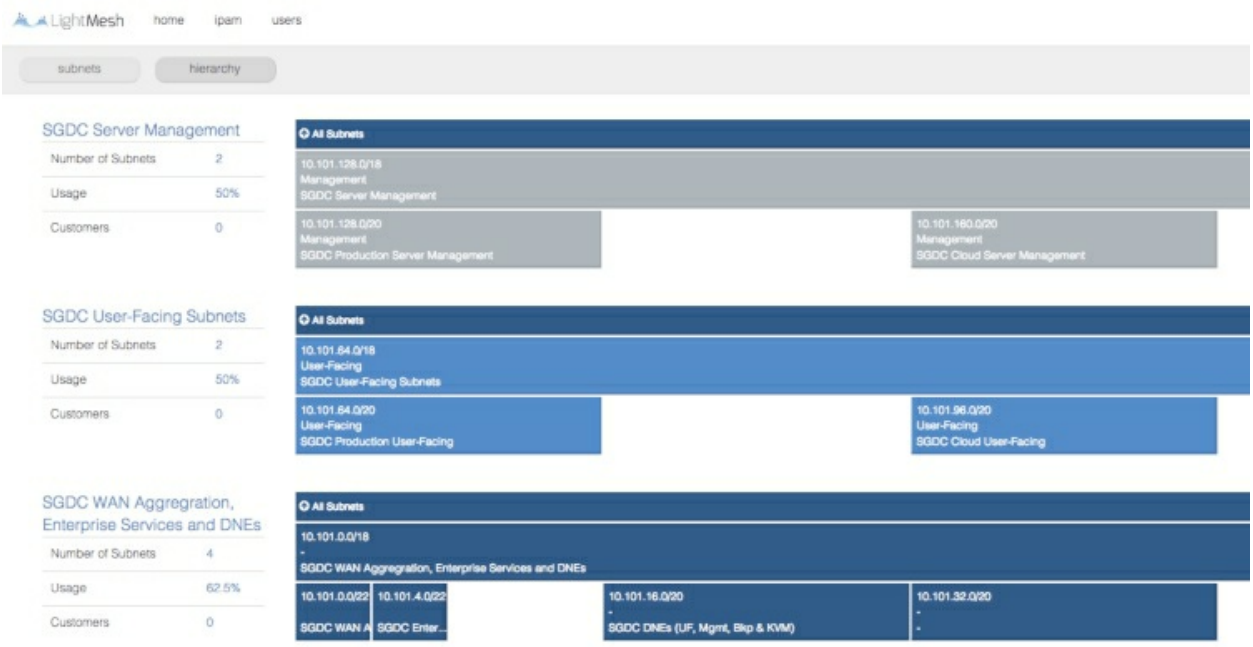
- Increased network agility
- Fewer security risks by automatically detecting and quarantining rogue devices.
- Predictive analysis to avoid address exhaustion and prevent unplanned outages.
- Automatically detect and remediate unmanaged devices.
- DHCP fingerprinting

- Centralized user interface
- Custom reports and alerts
- Customizable templates



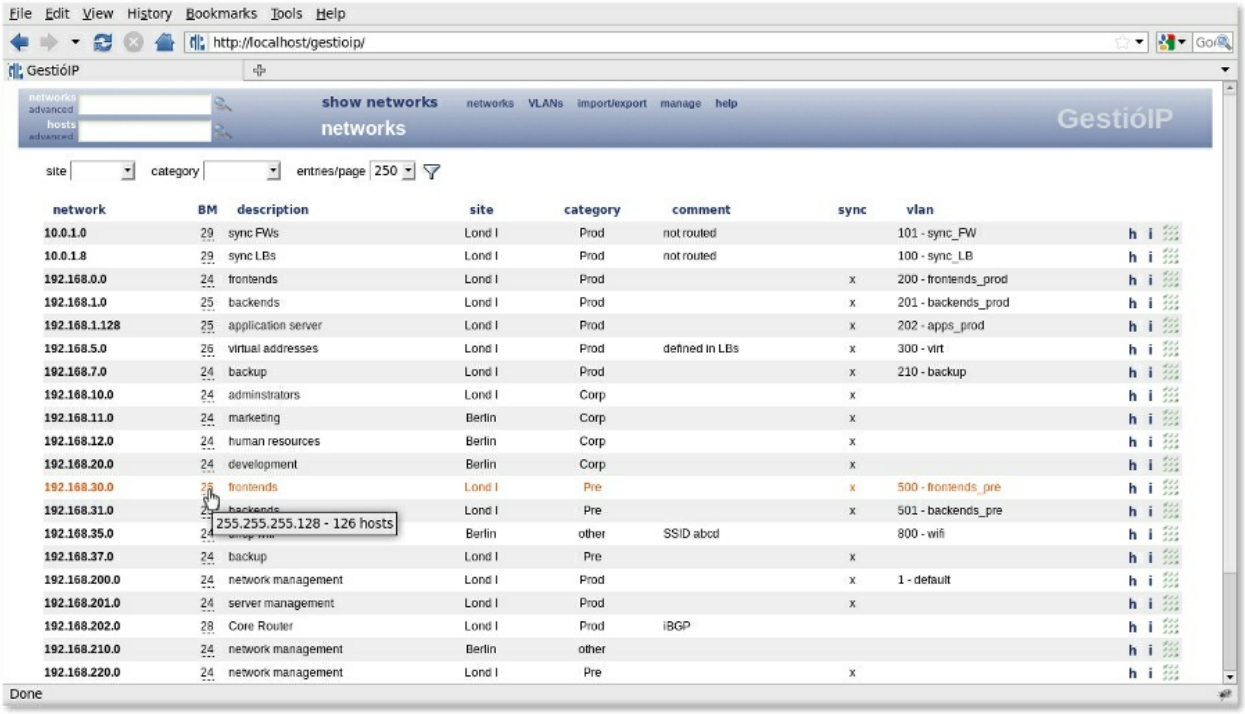
LightMesh IPAM

While LightMesh IPAM provides the same functionality as the same solutions listed earlier, what really makes it stand out next to the others its user interface. It’s very effective and does a great job representing the most important information. It’s a simpler solution for enterprise environments at a relatively cheaper cost – **200$** per month for up to **10000** IP addresses.



GestióIP

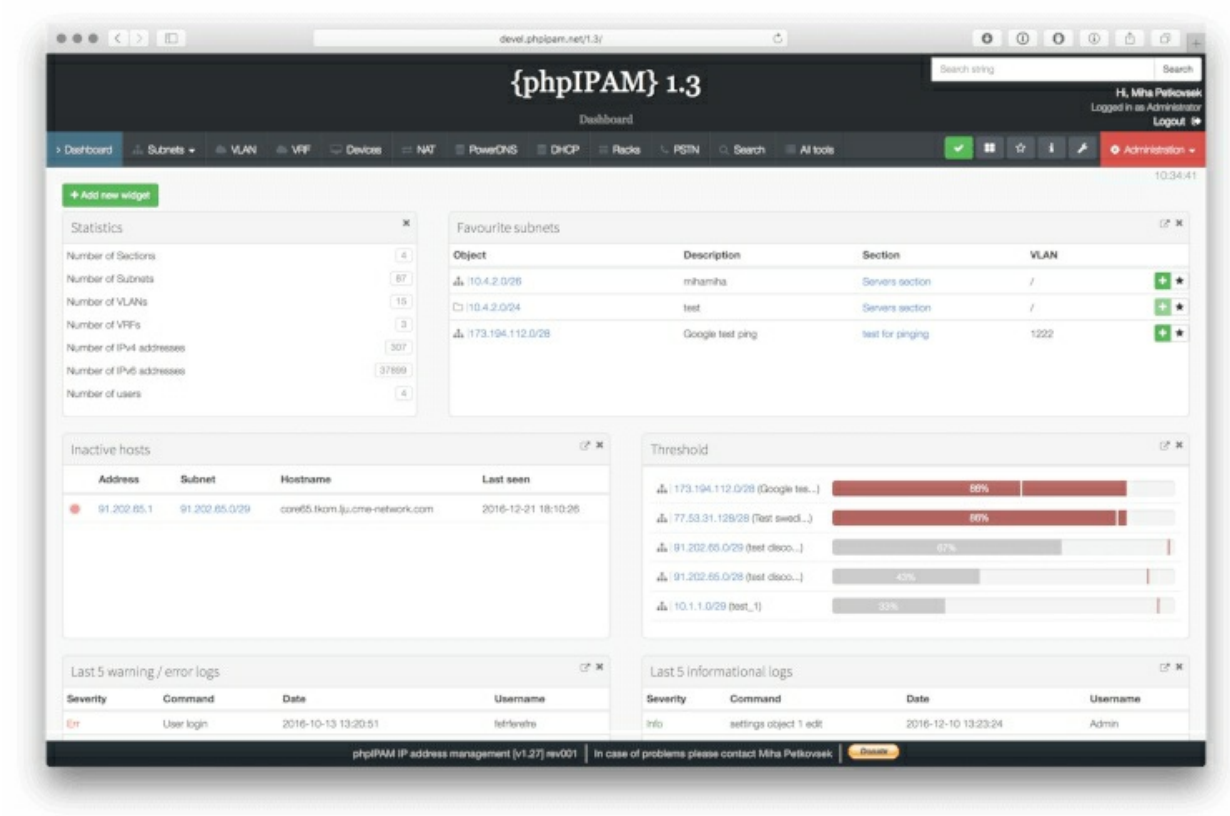
GestióIP is a web-based automated IP address management (IPAM) software comes with powerful features such as network discovery functions, provides search and filter feature for both networks and host, an Internet Search Engine that lets you find the information that network administrators frequently looking for.



GestióIP – Open Source IP Address Management Software

phpIPAM

phpIPAM is an open-source IP address management application, whose main motive is to offer light, modern and easy IP address management. It is based on PHP and uses MySQL database as a backend, it also uses jQuery libraries, Ajax and some HTML5/CSS3 features.



phpIPAM – Open Source IP Address Management Application

Conclusion

This was a short list of IP address management (IPAM) tools to help you keep track of your network.

Open Source Log Monitoring and Management Tools

When an operating system such as Linux is running, there are many events happening and processes that run in the background to enable efficient and reliable use of system resources. These events may happen in system software for example the **init** process or user applications such as Apache, MySQL, FTP and many more.

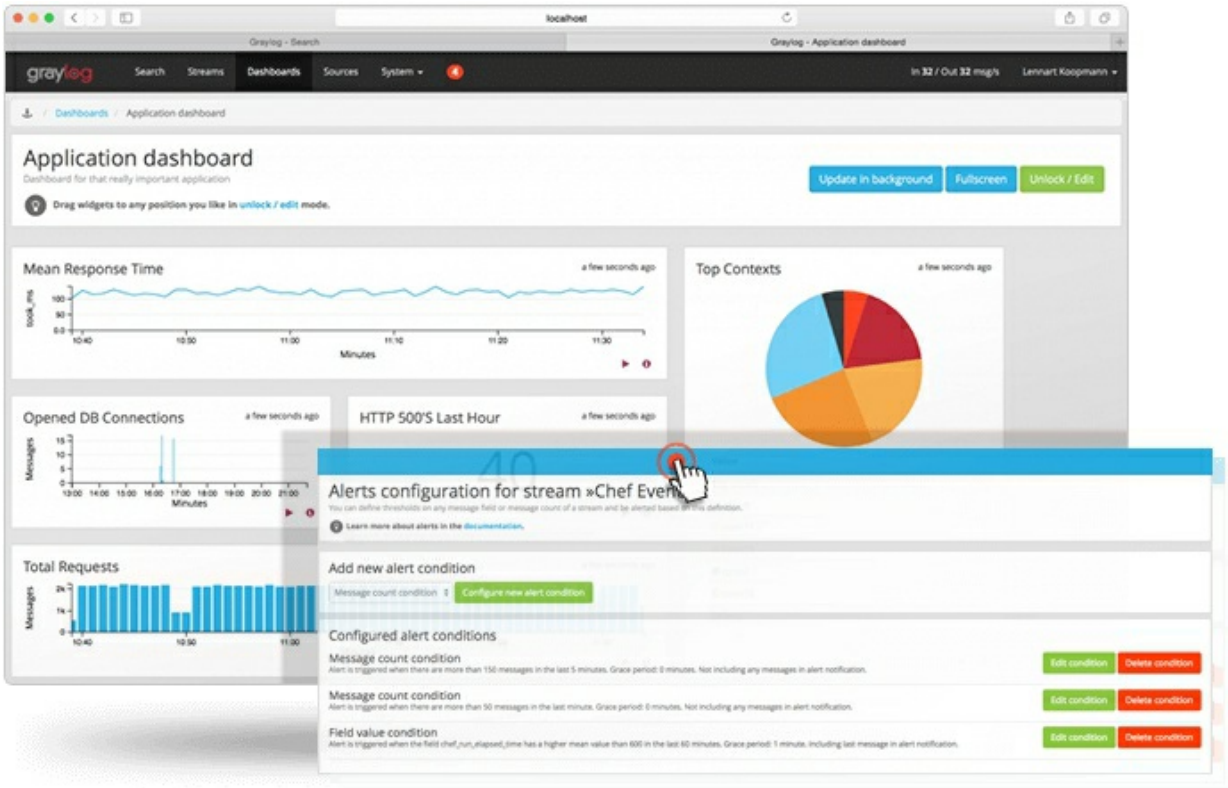
In order to understand the state of the system and different applications and how they are working, System Administrators have to keep reviewing logfiles on daily basis in production environments.

You can imagine having to review logfiles from several system areas and applications, that is where logging systems come in handy. They help to monitor, review, analyzer and even generate reports from different logfiles as configured by a System Administrator.

In this article, we shall look at the top four most used open source logging management systems in Linux today, the standard logging protocol in most if not all distributions today is **syslog**.

1. Graylog 2

This is a fully integrated open source log management system that enables System Administrators to collect, index, and analyze both framed, systematic and disorganized data from just about any available source systems.



This logging system is highly pluggable and enables centralized log management from many systems. It is integrated with external components such as MongoDB for metadata and Elasticsearch used to keep logfiles and enable text search.

Graylog 2 has the following features:

1. Ready for enterprise level production
2. Includes a dashboard and an alerting system
3. Can work on data from any log source
4. Enables real time log processing

5. Enables parsing of unstructured data
6. Extensible and highly customizable
7. Offers an operational data hub

2. Logcheck

Logcheck is an open source log management system that helps System Administrators automatically identify unknown problems and security violations in logfiles. It periodically sends messages about the analysis results to a configured e-mail address.

```
Logcheck(8)                System Manager's Manual                Logcheck(8)

NAME
       logcheck - program to scan system logs for interesting lines

SYNOPSIS
       logcheck [OPTIONS]

DESCRIPTION
       The  logcheck  program helps spot problems and security violations in
       your logfiles automatically and will send the results to you periodi-
       cally  in  an  e-mail.  By default logcheck runs as an hourly cronjob
       just off the hour and after every reboot.

       logcheck supports three level of filtering: "paranoid" is  for  high-
       security  machines  running as few services as possible. Don't use it
       if you can't handle its verbose messages.  "server"  is  the  default
       and  contains rules for many different daemons.  "workstation" is for
       sheltered machines and filters most  of  the  messages.   The  ignore
       rules  work in additive manner. "paranoid" rules are also included at
       level "server". "workstation"  level  includes  both  "paranoid"  and
       "server" rules.

       The  messages  reported  are sorted into three layers, system events,
       security events and attack alerts. The verbosity of system events  is
       controlled  by  which  level you choose, paranoid, server or worksta-
 Manual page logcheck(8) line 1 (press h for help or q to quit)
```

Logcheck is designed as a cronjob on an hourly basis and on every system reboot by default. Three are different levels of logfile filtering are developed in this logging system which include:

1. **Paranoid**: is intended for high-security systems that are running very few services as possible.
2. **Server**: this is the default filtering level for logcheck and its rules are defined for many different system daemons. The rules defined under paranoid level are also included under this level.
3. **Workstation**: it is for sheltered systems and helps to filter most of the messages. It also includes rules defined under paranoid and

server levels.

Logcheck is also capable of sorting messages to be reported into three possible layers which include, security events, system events and system attack alerts. A System Administrator can choose the level of details to which system events are reported depending on the filtering level though this does not affect security events and system attack alerts.

3. Logwatch

Logwatch is a Linux/Unix system logfile analyzer and reporter that can be easily customized and it also allows a System Administrator to add additional plugins, create custom scripts that serve specific logging needs.

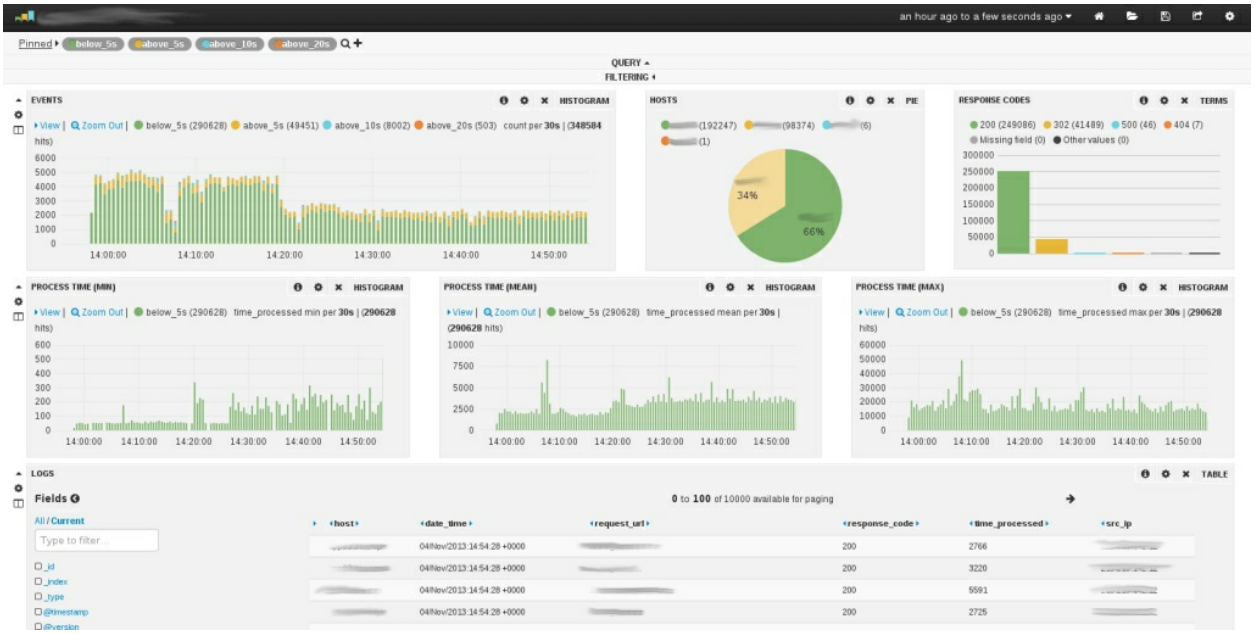
```
LOGWATCH(8)                    User Manuals                    LOGWATCH(8)

NAME
       logwatch - system log analyzer and reporter

SYNOPSIS
       logwatch  [--detail  level  ]  [--logfile log-file-group ] [--service
       service-name ] [--mailto address  ]  [--archives]  [--range  range  ]
       [--debug  level  ]  [--filename  file-name  ]  [--logdir  directory ]
       [--hostlimit hosts ] [--hostname hostname ]  [--html_wrap  number  of
       characters ] [--hostformat host based options ] [--output output-type
       ] [--format report format ] [--encode encoding to use  ]  [--numeric]
       [--version] [--help|--usage]

DESCRIPTION
       Logwatch is a customizable, pluggable log-monitoring system.  It will
       go through your logs for a given period of time and make a report  in
       the  areas  that you wish with the detail that you wish.  Logwatch is
       being used for Linux and many types of UNIX.

OPTIONS
       --detail level
              This is the detail level of the report.  level can be a  posi-
              tive integer, or high, med, low, which correspond to the inte-
              gers 10, 5, and 0, respectively.

 Manual page logwatch(8) line 1 (press h for help or q to quit)
```

What it does is to review system logfiles for a given period to time and then generates a report based on system areas that you wish to collect information from. One feature of this logging system is that it is easy to use for new System Administrator and it also works on most Linux distributions available and many Unix systems.

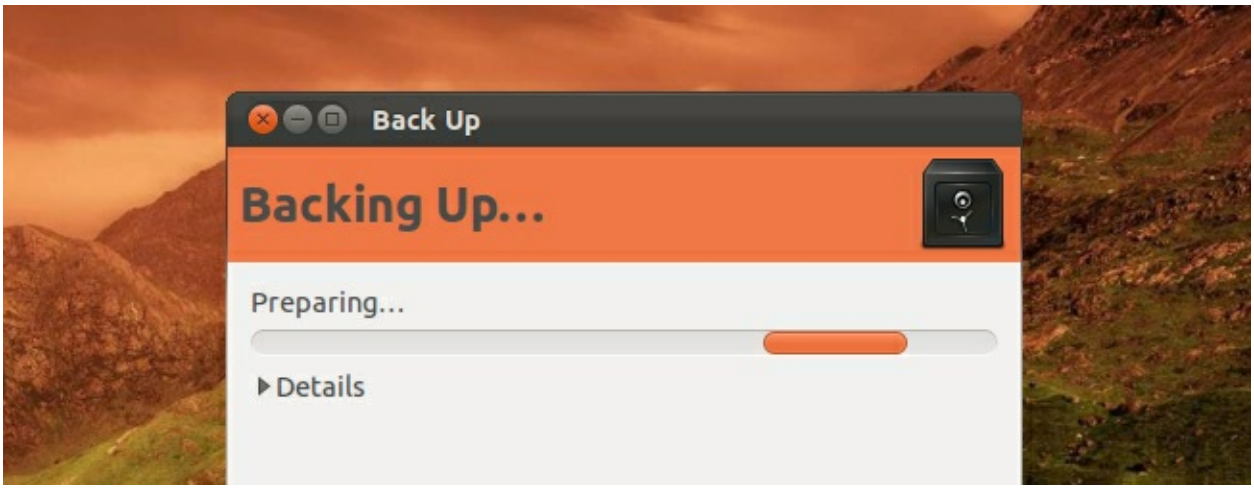
4. Logstash

Logstash is also an open source data collection and logging system available

on Linux, which capable of real-time pipelining, which was originally designed for data collection but its new versions now integrated several other capabilities such as using a wide range of input data formats, filtering and also output plugins and formats.



It can effectively unify data from various log source systems and normalize the data into targets of a System Administrators' choice. Logstash also allows System Administrators to cleanse, compare and standardize all their logging data for distinct advanced analytics and also create visualization use cases as well.

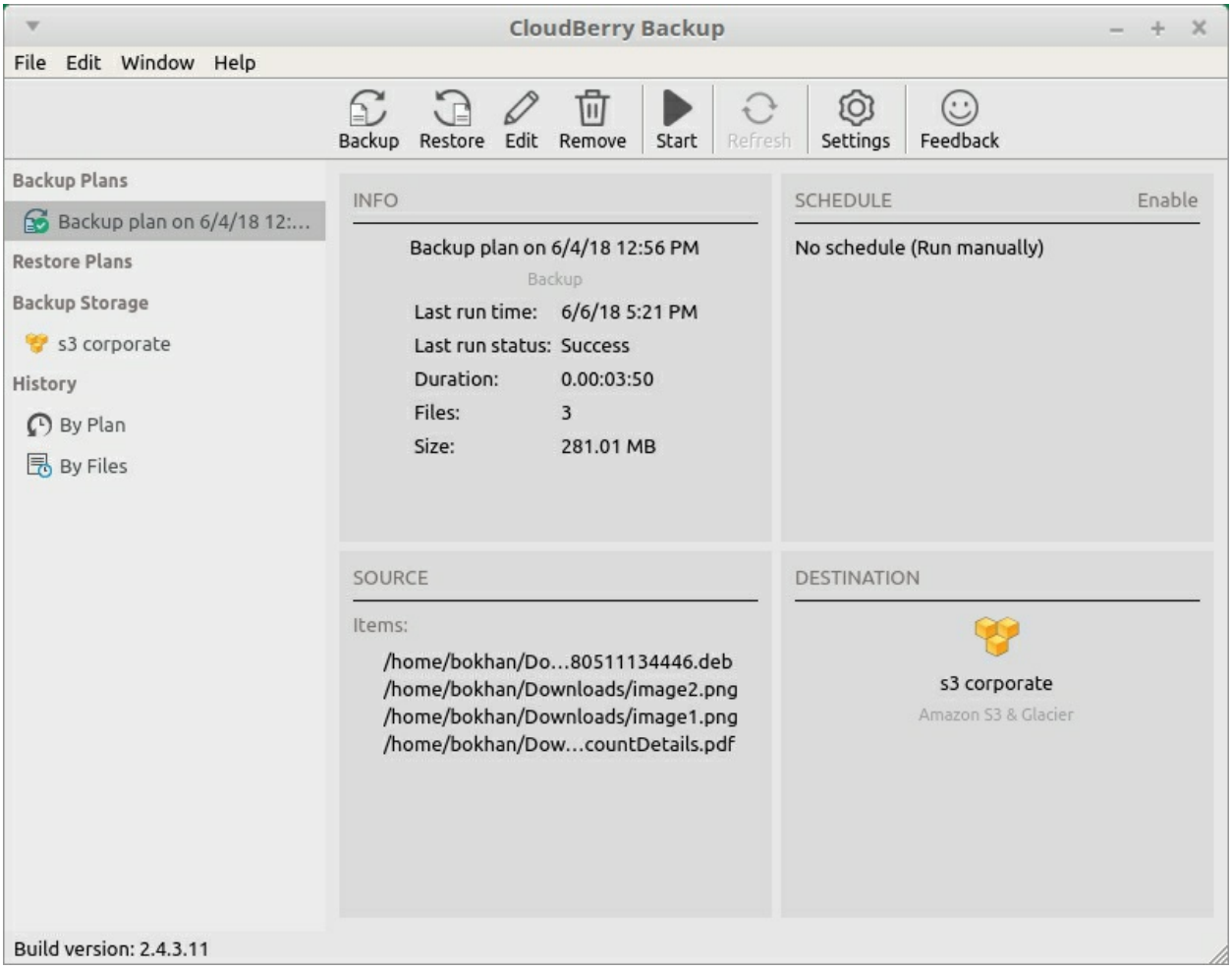


Honorable Mention – CloudBerry Backup

CloudBerry Backup for Linux is a cross-platform cloud backup solution with

advanced backup configuration settings and providing total security of data.

.



With this tool you can backup files and folders to the cloud storage of your choice: it supports more than **20** wide-known cloud storage services. CloudBerry Backup works with Ubuntu, Debian, Suse, Red Hat and other Linux distributions and is also compatible with Windows and Mac OSs.

The primary backup features are:

- Compression
- 256-bit AES Encryption
- Scheduled backup
- Incremental backup
- Command line interface
- Retention policy and more.

1. Rsync

It is a command line backup tool popular among Linux users especially System Administrators. It feature rich including incremental backups, update whole directory tree and file system, both local and remote backups, preserve file permissions, ownership, links and many more.

It also has a graphical user interface called **Grsync** but one advantage with the rsync is that backups can be automated using scripts and cron jobs when used by experienced System Administrators on the command line.

```
rsync(1)                                                          rsync(1)

NAME
       rsync - a fast, versatile, remote (and local) file-copying tool

SYNOPSIS
       Local:  rsync [OPTION...] SRC... [DEST]

       Access via remote shell:
         Pull: rsync [OPTION...] [USER@]HOST:SRC... [DEST]
         Push: rsync [OPTION...] SRC... [USER@]HOST:DEST

       Access via rsync daemon:
         Pull: rsync [OPTION...] [USER@]HOST::SRC... [DEST]
               rsync [OPTION...] rsync://[USER@]HOST[:PORT]/SRC... [DEST]
         Push: rsync [OPTION...] SRC... [USER@]HOST::DEST
               rsync [OPTION...] SRC... rsync://[USER@]HOST[:PORT]/DEST

       Usages with just one SRC arg and no DEST arg will list the source files
       instead of copying.

DESCRIPTION
       Rsync is a fast and extraordinarily versatile file  copying  tool.   It
 Manual page rsync(1) line 1 (press h for help or q to quit)
```

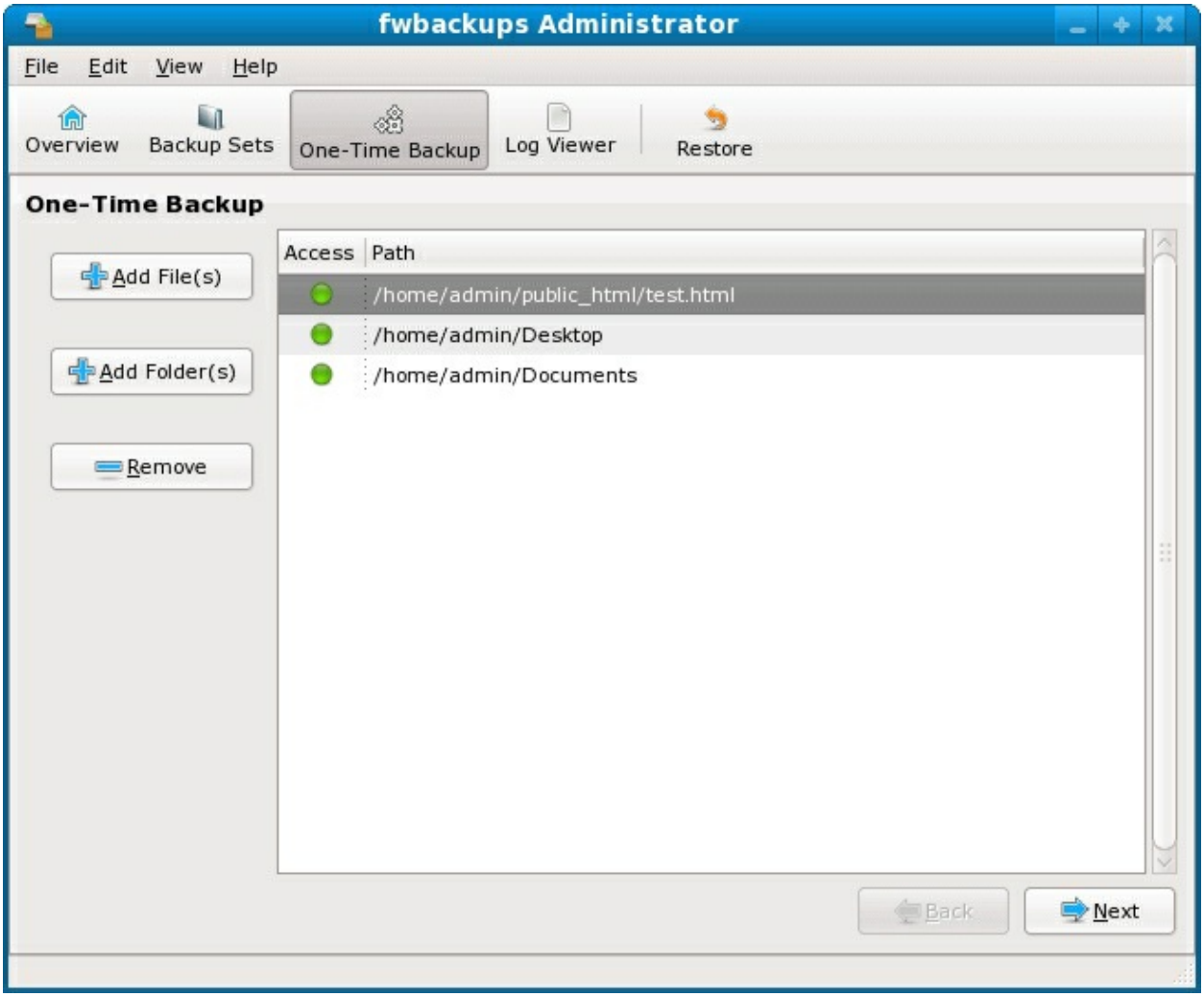
2. Fwbackups

It is free and open source software which is cross platform and feature rich and users can contribute to it’s development or just participate in testing it. It has an intuitive interface that allows users to do backups easily.

It has features such as:

1. Simple interface
2. Flexibility in backup configuration
3. Remote backups
4. Backup entire file system

5. Exclude files and directories plus many more



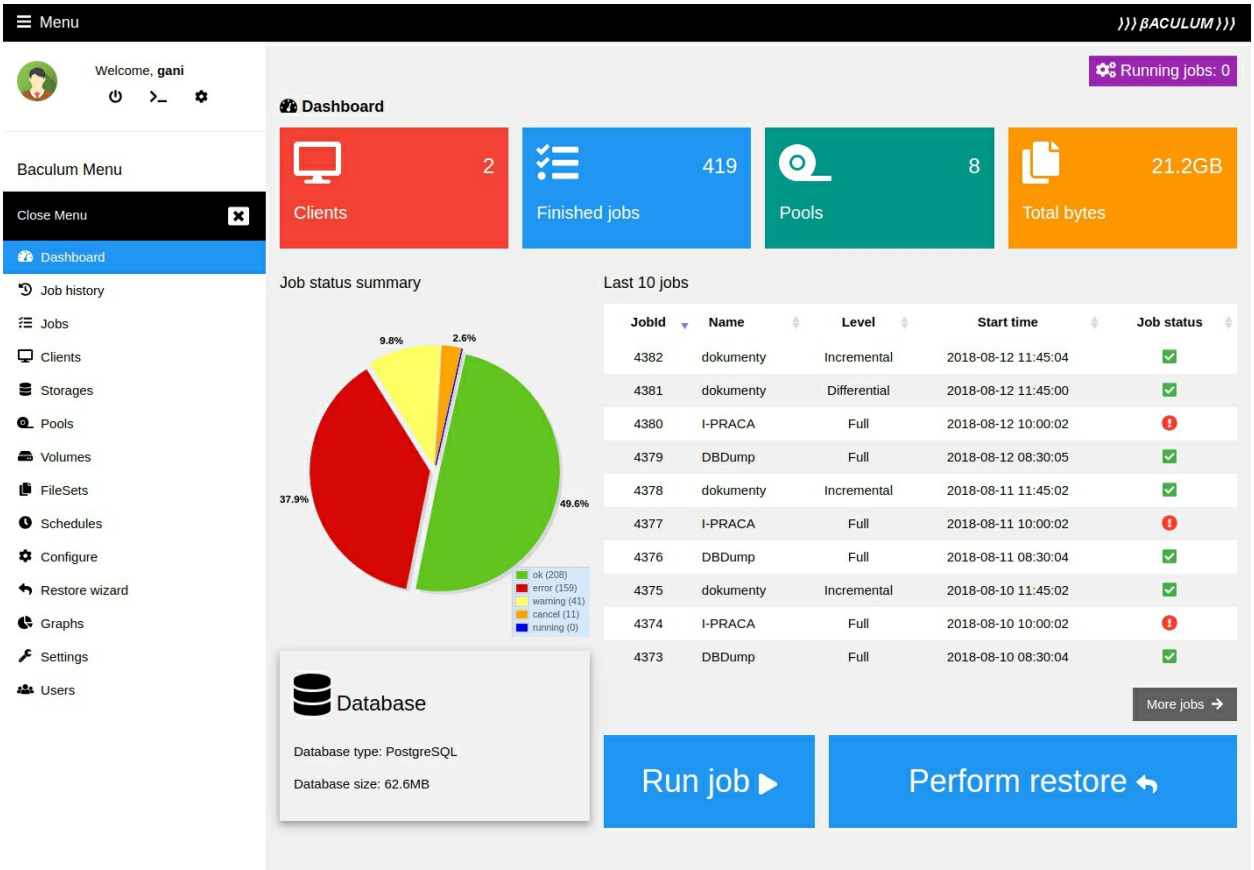
3. Bacula

It is open source data backup, recovery and verification software that is designed to be enterprise ready with certain complexities, though these complexities actually define it's powerful features such as backup configurations, remote backups plus many more .

It is network based and is made up of the following programs:

1. **a director** : program that supervises all operations of Bacula.
2. **a console** : program that allows a user to communicate with the Bacula director above.
3. **a file**: program that is installed on the machine to be backed up.

4. **storage**: program that is used to read and write to your storage space.
5. **catalog**: program responsible for the databases used.
6. **Monitor**: program that keeps track of all events happening in different parts of Bacula.



4. Backupninja

It is powerful backup tool that allows users to design backup activity configuration files that can be drooped in **/etc/backup.d/** directory. It helps to perform secure, remote and also incremental backups over a network.

It has got the following features:

1. Easy to read ini style configuration files.
2. Use scripts to handle new types of backups on your system.
3. Schedule backups

4. Users can choose when status report emails are mailed to them.
5. Easily create backup action configuration file with console-based wizard (ninjahelper).
6. Works with Linux-Vservers.

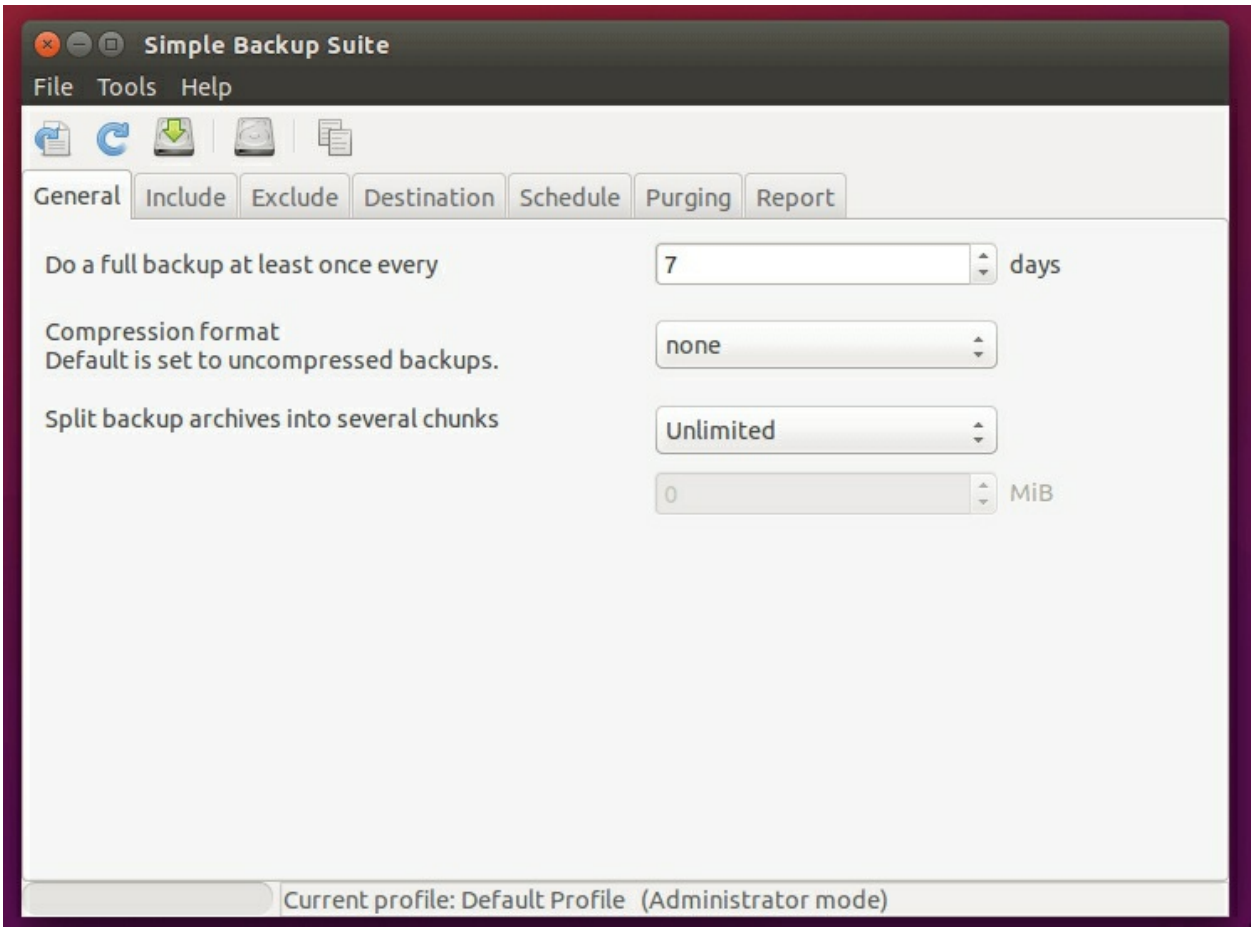
```
BACKUPNINJA(1)                  backupninja package                BACKUPNINJA(1)

NAME
       BACKUPNINJA - A lightweight, extensible meta-backup system
       "a silent flower blossom death strike to lost data."

SYNOPSIS
       backupninja  [  -h ] [ -d ] [ -n ] [ -t ] [ -f filename ] [ --run file-
       name ]

DESCRIPTION
       Backupninja allows you to coordinate system backups by dropping  a  few
       simple configuration files into /etc/backup.d/. Most programs you might
       use for making backups don't have their own configuration file  format.
       Backupninja provides a centralized way to configure and coordinate many
       different backup utilities.

FEATURES
       - easy to read ini style configuration files.

       - you can drop in scripts to handle new types of backups.

       - backup actions can be scheduled.
 Manual page backupninja(1) line 1 (press h for help or q to quit)
```

5. Simple Backup Suite (sbackup)

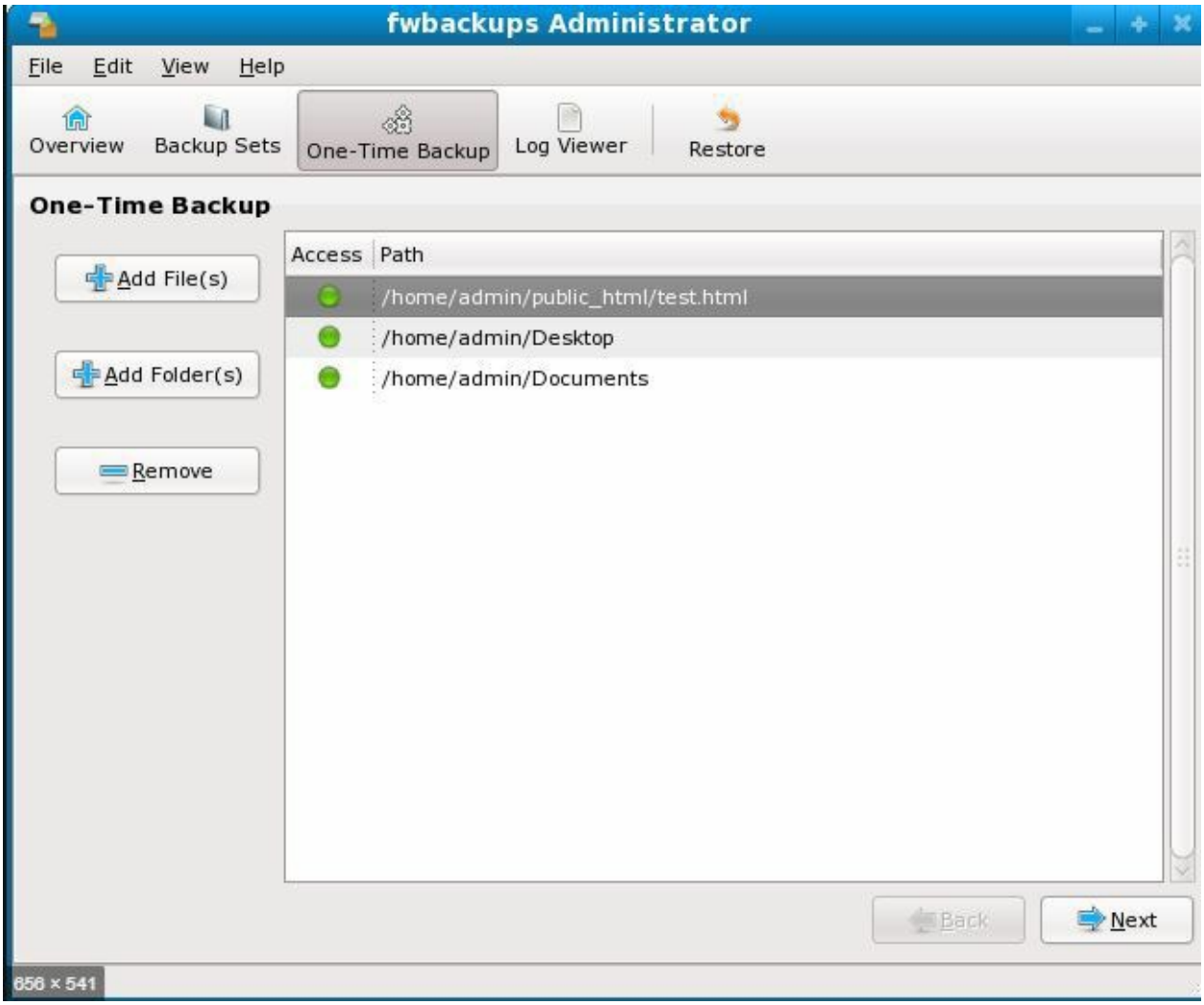
It is backup solution for Gnome desktop where users can access all configuration via Gnome interface. Users can use regex to specify file and directory paths during the backup process.

It has the following features:

1. Creates compressed and uncompressed backups.
2. Supports multiple backup profiles.
3. Allows logging, email notifications.
4. Scheduled backups and manual backups.
5. Split uncompressed backups into several chunks.
6. Supports local and remote backups.


Simple Backup Suite
File
Tools
Help
General
Include
Exclude
Destination
Schedule
Purging
Report
Do a full backup at least once every
7
days
Compression format
Default is set to uncompressed backups.
none
Split backup archives into several chunks
Unlimited
0
MiB
Current profile: Default Profile (Administrator mode)

Extra:Backup Program for Linux



fwbackups – Features

fwbackups offers a rich interface that is both powerful and easy to use with the following features:

- **Simple interface**: Creating new backups or restoring from a previous backup is an easy task.
- **Flexible backup configuration**: Select between a number of backup formats and modes, which includes the archive format and clone copy mode for recovering data from corrupted or damaged disks.
- **Backup your files to any computer**: It can backup files to a remote backup server or a connected media like USB device, making it perfect for all users.
- **Backup the entire computer**: Create archives images of your

entire system so that your files are safe.

- **Scheduled and one-time backups**: Select to run a backup once (on demand) or periodically so that you never have to panic about losing your data again.
- **Faster backups**: Create your backup quicker by taking only the changes from the last backup with the incremental backup modes.
- **Exclude files or folders**: Don't waste disk space on your system by backing up files you don't require.

Install fwbackups on Linux Systems

fwbackups is not included in most Linux distribution's repositories, so the only way to install **fwbackups** using the source tarball as explained below.

Install fwbackups on Debian and Ubuntu/Mint Linux

First, install these following dependencies on your system.

```
$ sudo apt-get install gettext autotools-dev intltool python-crypto python-paramiko python-gtk2 python-glade2 python-notify cron
```

Then download fwbackups home directory using wget command install it from source using the following commands.

```
$ wget http://downloads.diffingo.com/fwbackups/fwbackups-1.43.7.tar.bz2
$ tar xfj fwbackups-1.43.7.tar.bz2
$ cd fwbackups-1.43.7/
$ ./configure --prefix=/usr
$ make && sudo make install
```

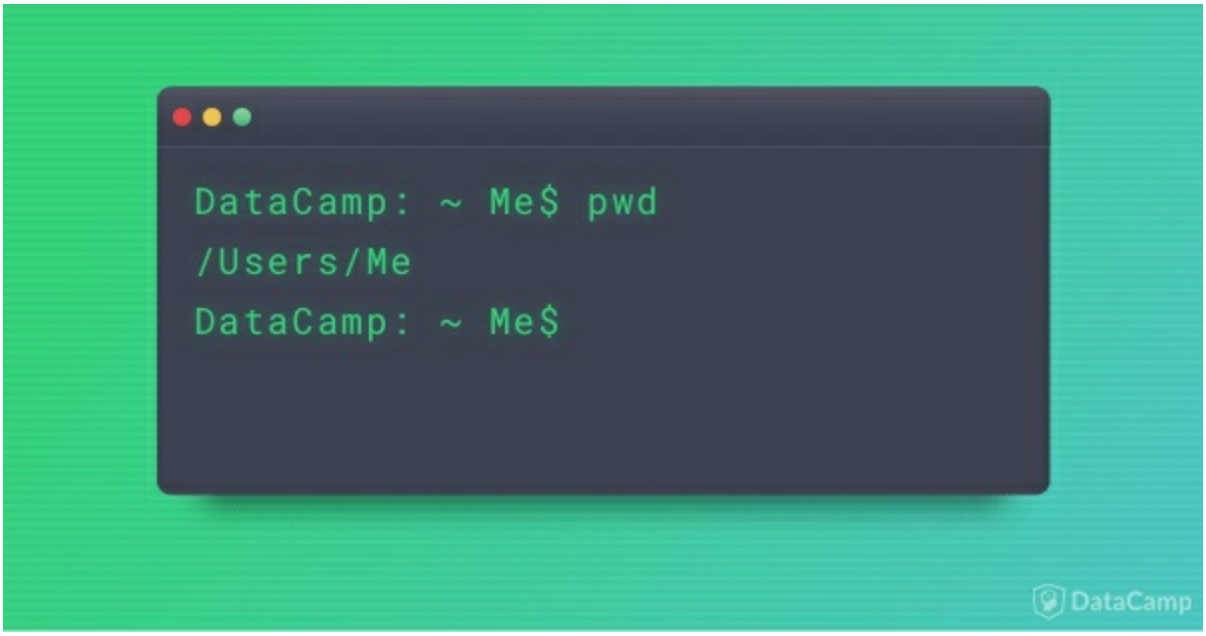
Install fwbackups on CentOS and RHEL

```
$ sudo yum install gettext autotools-dev intltool python-crypto python-paramiko python-gtk2 python-glade2 python-notify cron
```

Next, download fwbackups and install it from source using the following commands.

```
$ wget http://downloads.diffingo.com/fwbackups/fwbackups-1.43.7.tar.bz2
$ tar xfj fwbackups-1.43.7.tar.bz2
$ cd fwbackups-1.43.7/
$ ./configure --prefix=/usr
$ make && sudo make install
```

Install wget on Linux



***install wget on RHEL, CentOS and Fedora**
yum install wget

To confirm the installation, run the command.
rpm -qa | grep wget

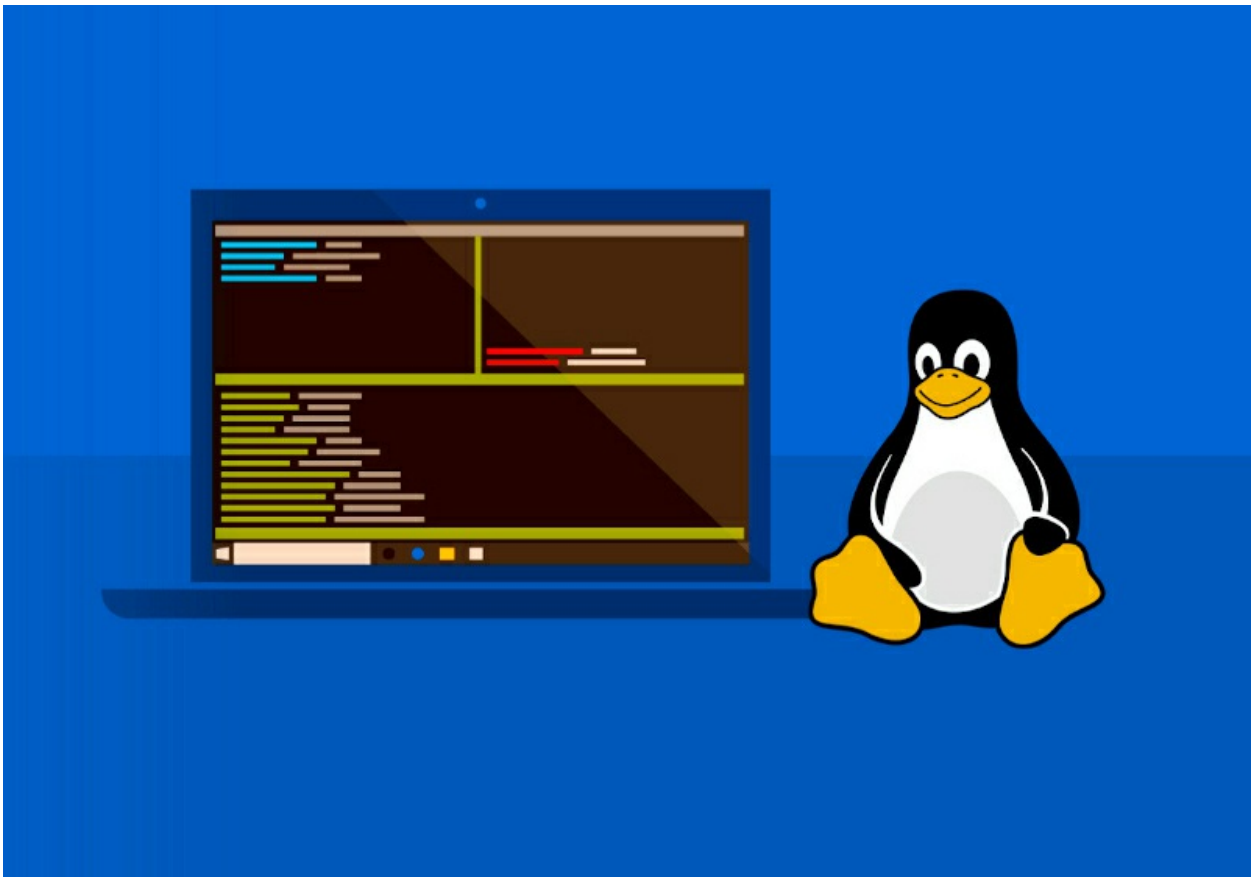
***Install wget on OpenSUSE**
On **OpenSUSE**, install **wget** by running.
zypper install wget

To confirm the installation run.
zypper se wget

***Install wget on ArchLinux**
On **ArchLinux**, install **wget** by running the command.
pacman -Sy wget

To check if **wget** is installed and print more information about the tool run.
pacman -Qi wget

Install and Use dig and nslookup Commands in Linux



Dig, short for **Domain Information Gopher**, is a DNS lookup utility used for probing DNS servers and troubleshooting problems associated with DNS servers

Nslookup is used for handling DNS lookups and displays crucial information such as **MX** records, and the IP address associated with a domain name.

Installing dig & nslookup in CentOS/RHEL

```
# dnf install bind-utils
```

Verify the version

```
# dig -v
```

Installing dig & nslookup on Debian / Ubuntu

apt install dnsutils

Installing dig & nslookup on ArchLinux

pacman -Sy dnsutils

HOW TO USE DIG :

dig google.com

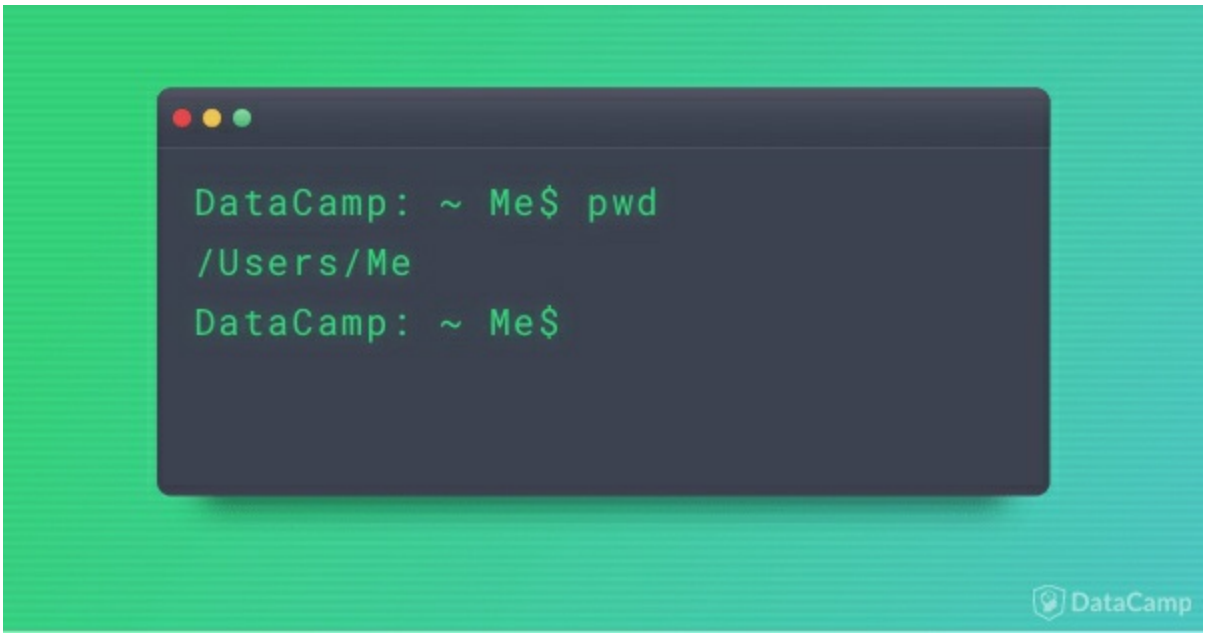
dig google.com MX +short

HOW TO USE NSLOOKUP

nslookup google.com

Manage Networking with Network Manager in RHEL/CentOS

Installing NetworkManager on CentOS/RHEL




```
# dnf install NetworkManager
```

The global configuration file for **NetworkManager** is located at **/etc/NetworkManager/NetworkManager.conf** and additional configuration files can be found in **/etc/NetworkManager/**.

Checking the status of NetworkManager

A minimal installation of **CentOS/RHEL 8** should have **NetworkManager** started and enabled to automatically start at boot time, by default. You can use the following commands to check if NetworkManager is active, enabled, and print runtime status information of the NetworkManager.

```
# systemctl is-active NetworkManager
# systemctl is-enabled NetworkManager
# systemctl status NetworkManager
```

Starting NetworkManager

If the **NetworkManager** is not running, you can start it by simply running.

```
# systemctl start NetworkManager
```

Stoping NetworkManager

To stop or deactivate the **NetworkManager** for one reason or the other, issue

the following command.

```
# systemctl stop NetworkManager
```

Restarting NetworkManager

If you have made any changes to interface configurations files or NetworkManager daemon's configuration (typically located under the **/etc/NetworkManager/** directory), you can restart (stop and then start) the NetworkManager to apply the changes as shown.

```
# systemctl restart NetworkManager
```

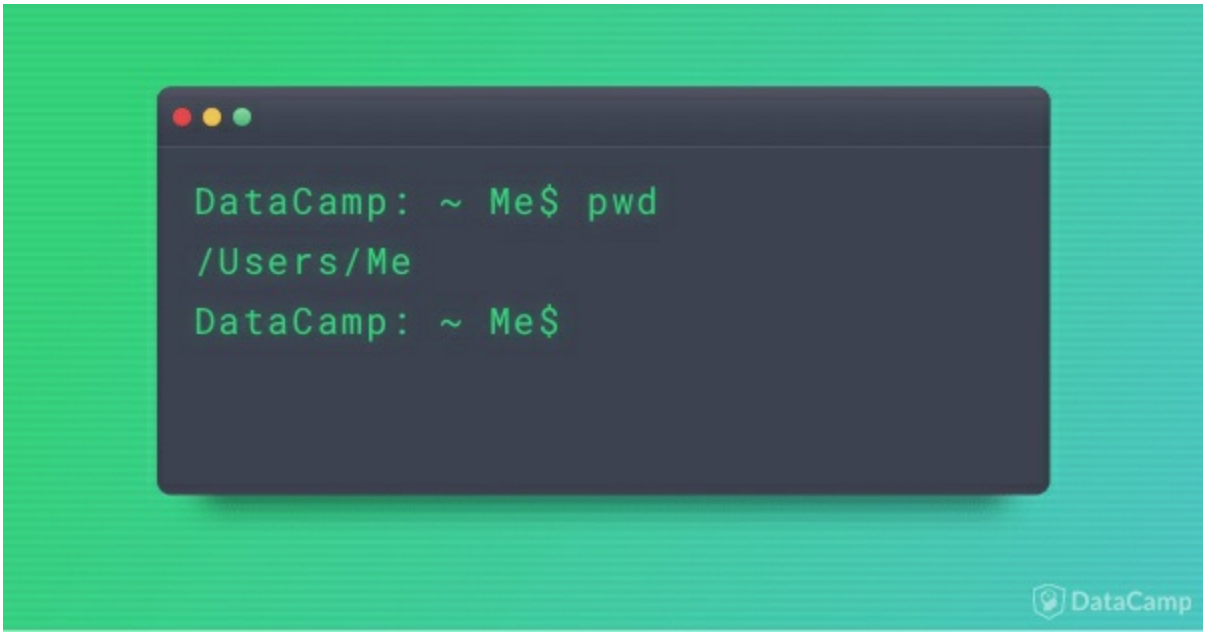
Reloading NetworkManager

To reload the **NetworkManager** daemon's configuration (but not the unit configuration file of systemd) without restarting the service, run the following command.

```
# systemctl reload NetworkManager
```

Set Static IP Address on CentOS/RHEL

Set Static IP Address on CentOS/RHEL



Network interface-specific configuration files are located in the **/etc/sysconfig/network-scripts/** directory. You can edit any of these files, for example, to set a static IP address for your CentOS/RHEL 8 server.

```
# vi /etc/sysconfig/network-scripts/ifcfg-enp0s3
```

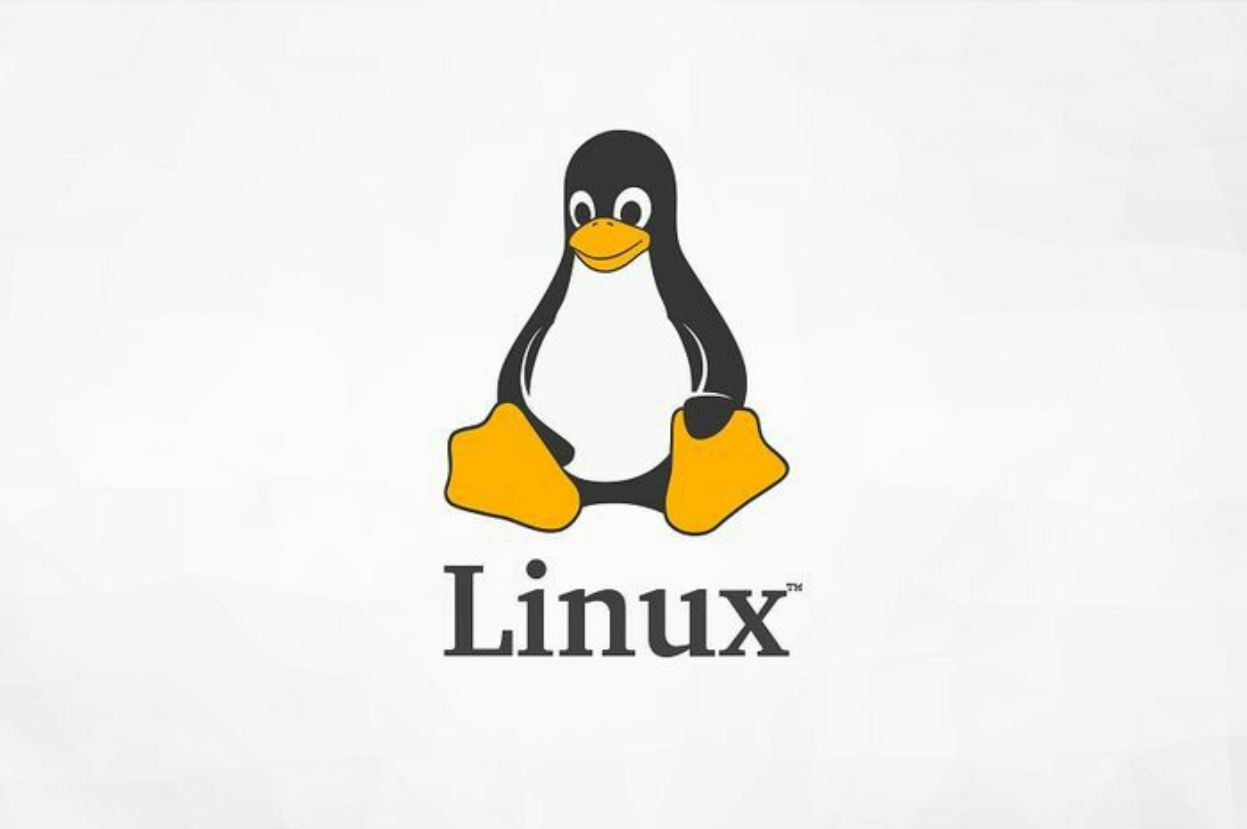
Here is a sample configuration for setting a static IP address.

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp0s3
UUID=e81c46b7-441a-4a63-b695-75d8fe633511
DEVICE=enp0s3
ONBOOT=yes
IPADDR=192.168.0.111
PREFIX=24
GATEWAY=192.168.0.1
DNS1=8.8.8.8
PEERDNS=no
```

After saving the changes, you need to reload all connection profiles or restart the **NetworkManager** for the new changes to apply.

```
# nmcli connection reload
OR
# systemctl restart NetworkManager
```

Starting or Stopping Network Services/Scripts Based on Network Connectivity



NetworkManager has a useful option that allows users to execute services (such as **NFS**, **SMB**, etc.) or simple scripts based on network connectivity.

For example, if you want to automatically mount a remote directory locally with sshfs, mount SMB shares, or mount NFS shares after switching between networks. You may want such network services to be executed not until NetworkManager is up and running (all connections are active).

This feature is provided by the **NetworkManager-dispatcher** service (which must be started and enabled to start automatically at system boot). Once the service is running, you can add your scripts to the **/etc/NetworkManager/dispatcher.d** directory.

All scripts must be executable and writable, and owned by root, for example:

```
# chown root:root /etc/NetworkManager/dispatcher.d/10-nfs-mount.sh
# chmod 755 /etc/NetworkManager/dispatcher.d/10-nfs-mount.sh
```

Important: The dispatcher scripts are will be executed in alphabetical order at connection time, and in reverse alphabetical order at disconnect times.

Using Legacy Network Scripts

As we mentioned earlier, network scripts are deprecated in **CentOS/RHEL 8** and do not come installed by default. If you still want to use the n**etwork scripts**, you need to install the **network-scripts** package.

```
# yum install network-scripts
```

Once installed, this package provides a new version of the **ifup** and **ifdown** scripts which call **NetworkManager** via the **nmcli** tool we have looked at above. Note that **NetworkManager** should be running for you to run these scripts.

For more information, see the **systemctl** and **NetworkManager** man pages.

```
# man systemctl
# man NetworkManager
```

File and Disk Encryption Tools for Linux

1. Tomb

Tomb is a free and open source tool for easily encrypting and backing up files on GNU/Linux systems. It consists of a simple shell script that implements standard GNU tools alongside cryptsetup and LUKS (the Linux kernel's cryptographic API).

Tomb aims to improve safety by adopting a handful of well-tested standards and implementations, applying good practices for key storage, and a minimalist design consisting of concise readable code.

2. Cryptmount

Cryptmount is an open source utility created for GNU/Linux Operating Systems to enable users to mount encrypted files without root privileges.

It works using the newer **devmapper** mechanism which offers several advantages including improved functionality in the kernel, support for encrypted swap partitions for superusers, support for **crypto-swap** at system boot, storing multiple encrypted filesystems in a single disk, etc.

3. CryFS

CryFS is a free and open source cloud-based encryption tool for safely

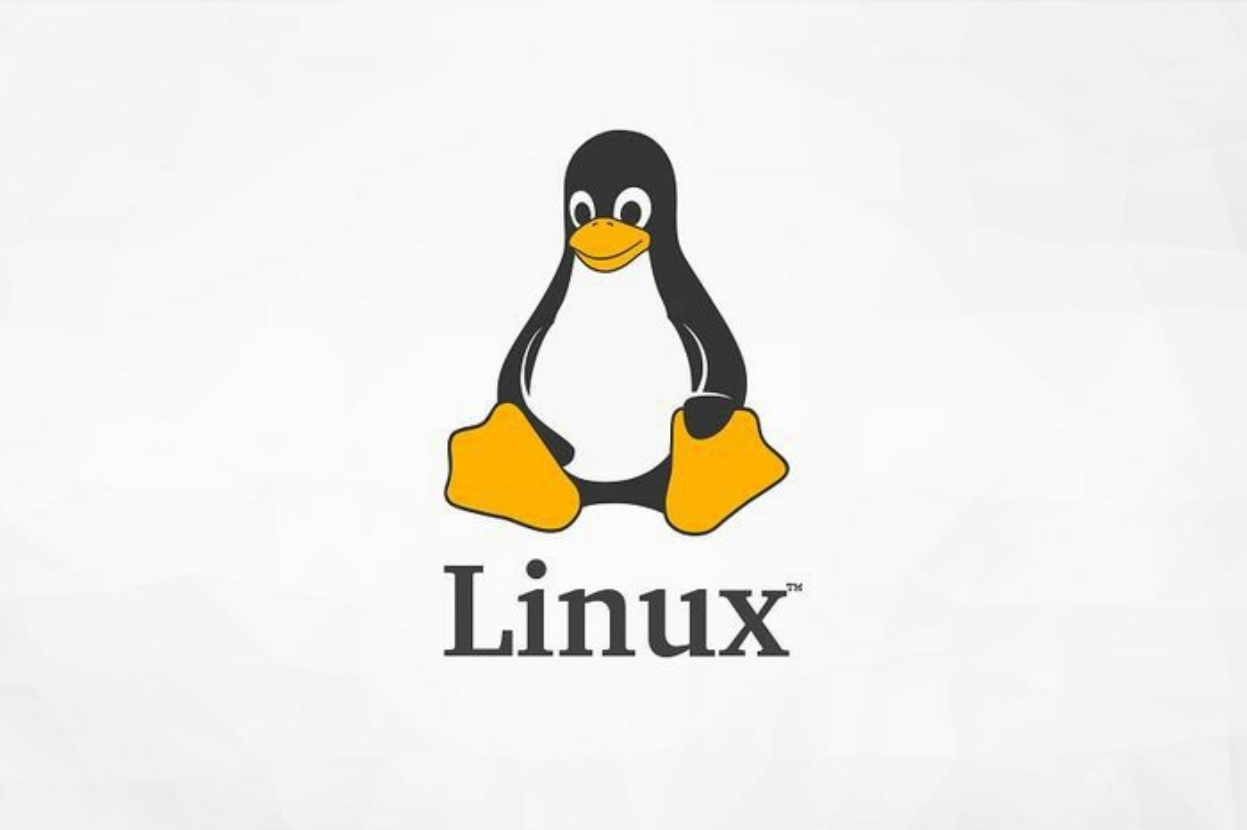
storing files anywhere. It is easy to set up, runs in the background, and works nicely with any popular cloud service not excluding **Dropbox**, **OneDrive**, and **iCloud**.

CryFS ensures that no data, including directory structure, metadata, and file content, leave your computer in an unencrypted format.

4. GnuPG

GnuPG, often referred to as **GPG**, stands for **GNU Privacy Guard** and it is a free and open source collection of cryptographic tools created as a replacement for Symantec's PGP cryptographic software suite.

Cron Jobs for Linux



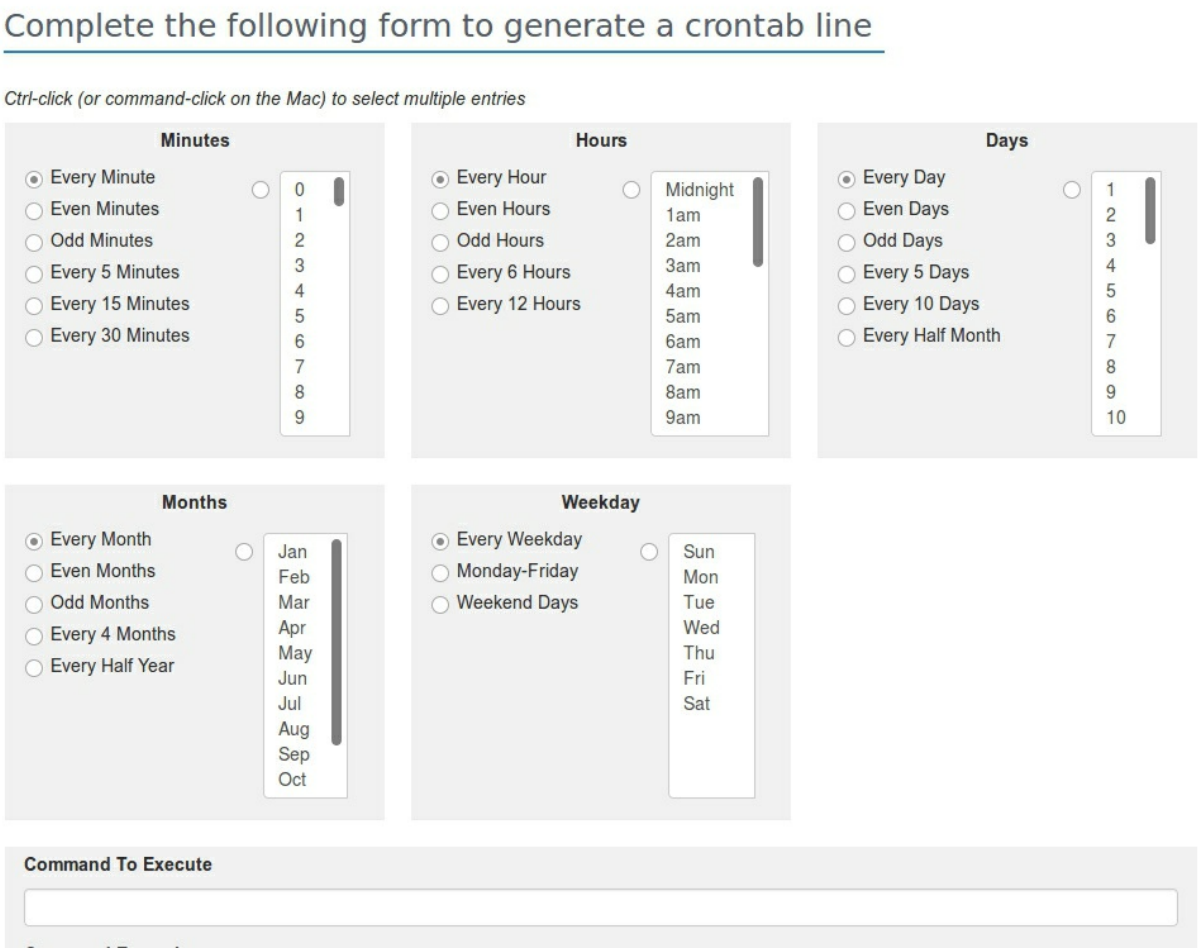
As a Linux system administrator, you can perform time-based scheduling of jobs/tasks using online cron job services or Cron, a powerful utility available in Unix/Linux systems.

In Linux, **cron** runs as a daemon and can be used to schedule tasks such as commands or shell scripts to perform various kinds of backups, system updates and much more, that run periodically and automatically in the background at specific times, dates, or intervals.

Scheduling a **cronjob** with the correct syntax can be confusing sometimes, wrong expressions can cause cronjobs to fail or not even run at all. In this article, we will list 6 useful online (web based) utilities for creating and testing cronjob scheduling syntax in Linux.

1. Crontab Generator

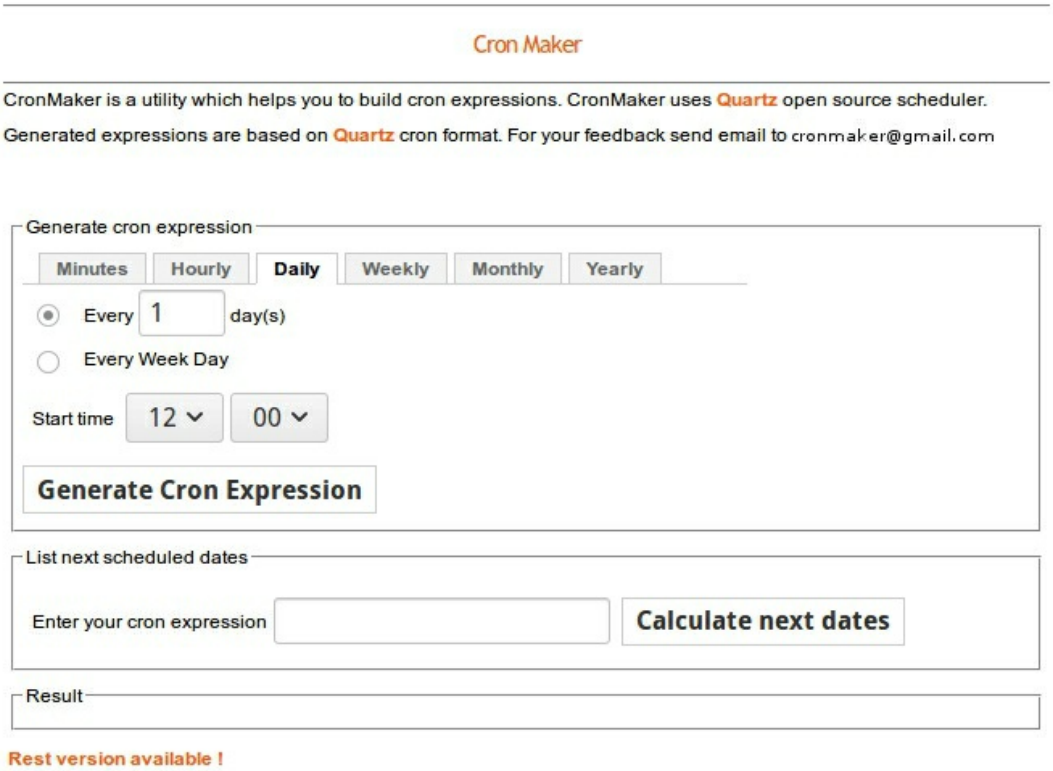
Crontab Generator is a useful online utility for generating a crontab entry to help schedule a job. It offers a simple, descriptive generator that can help you to produce a crontab syntax that you can copy and paste to your crontab file.



Crontab Generator

2. Cron Maker

Cron Maker is a web based utility which helps you to build cron expressions; it employs the Quartz open source library and all expressions are based on Quartz cron format. It also enables you to view next scheduled dates (simply enter a cronjob expression and calculate the next dates).



Cron Maker

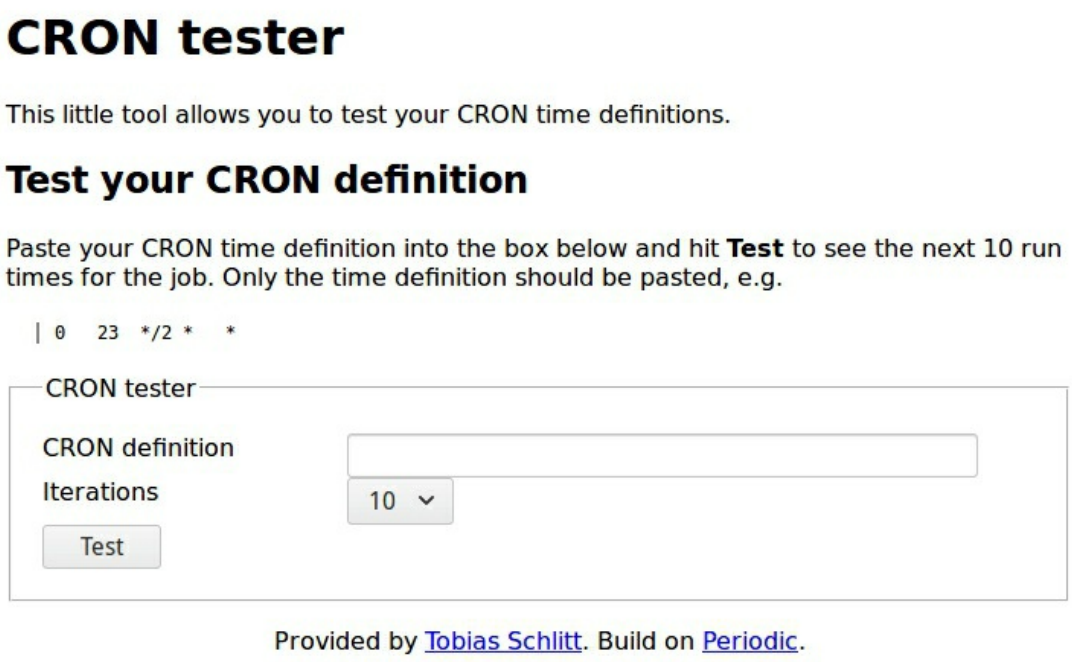
3. Crontab GUI

Crontab GUI is a great and the original online crontab editor. It works well (fully optimized) on mobile devices (you can generate cron syntax on your smart phone or tablet PC’s web browser).

Crontab GUI Tool

4. CRON Tester

CRON Tester is a useful cron tester that allows you to test your cron time definitions. All you need to do is copy and paste your cron syntax in the cron definition field, then choose the number of iterations and click on "**Test**" to see the various dates on which it will run.



Cron Tester

5. Crontab Guru

Crontab Guru is a simple online cron schedule expression editor. In addition, it provides a useful means of monitoring your cronjob. All you need to do is copy a command snippet provided and append at the end of the crontab definition. In case your cron job fails or doesn’t even start, you will receive an alert email.

6. Easycron

Easycron is a great web based cron scheduler for corntab.com cron editor. You can create a cron job by specifying a "**URL to call**", set when it should be executed, specify a cron expression or add it manually from a descriptive form. Importantly, you can optionally use basic HTTP authentication for a small layer of security.

Cron Jobs Plan API Support

Cron Job Scheduler

Add a cron job below. We will trigger it periodically.

URL & Time | Identifier | Method & Headers | Timeout | Notification (Email) | Notification (Webhook)

URL to call http://

HTTP basic auth User Password Optional

When to execute

- (•) Every Choose an interval
- ( ) Specify with cron expression Help
- ( ) Select manually

Create Cron Job

Developer?

Help your users easily set up cron jobs and earn commission »

Run a Command with Time Limit (Timeout) In Linux

Our goal is to run a command named “ping www.google.com” with a time limit of 30 seconds:

1. Open the terminal application
2. Run ping command have it abort after 30 seconds: **timeout 30s ping www.google.com**
3. One can combine sleep and other shell commands: **ping www.google.com & sleep 30s; kill $!**

Linux time limit command

The syntax is as follows for timeout command:

```
timeout DURATION COMMAND
timeout DURATION COMMAND arg1 arg2
timeout 1m ping google.com
timeout 30s tracepath www.cyberciti.biz
timeout [options] DURATION COMMAND arg1 arg2
```

DURATION is a floating point number with an optional suffix as follows:

- **s** for seconds (the default).
- **m** for minutes.
- **h** for hours.
- **d** for days.

How to run ping command for 60 seconds and abort it

Try:

```
timeout 60s ping www.google.com
```

Install Apache Tomcat on Debian

Apache Tomcat is a free, mature, robust, and popular web application server software that is used to serve Java-based applications. It is an open-source.

First Step, update the system software package :

$ sudo apt update

After that install the **Java Development Kit** package:

$ sudo apt install default-jdk

```
The following NEW packages will be installed:
  default-jdk default-jdk-headless firebird3.0-common firebird3.0-common-doc
  firebird3.0-server-core firebird3.0-utils fonts-liberation2 gnome-settings-daemon-common
  gnome-tweaks gstreamer1.0-gtk3 javascript-common libboost-iostreams1.67.0 libboost-locale1.67.0
  libcamel-1.2-62 libcue2 libdazzle-1.0-0 libebook-1.2-19 libedata-cal-1.2-29
  libedataserver-1.2-23 libedataserverui-1.2-2 libepubgen-0.1-1 libfbclient2 libgpgmepp6
  libib-util libice-dev libimagequant0 libnumbertext-1.0-0 libnumbertext-data liborcus-0.14-0
  libpthread-stubs0-dev libqxp-0.0-0 libreoffice-help-common libreoffice-sdbc-firebird
  libreoffice-style-colibre libreoffice-style-elementary libsm-dev libstaroffice-0.0-0 libtommath1
  libtracker-control-2.0-0 libtracker-miner-2.0-0 libx11-dev libxau-dev libxcb1-dev libxdmcp-dev
  libxmlsec1 libxmlsec1-nss libxt-dev libzmf-0.0-0 node-normalize.css openjdk-11-jdk
  openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless orca policykit-1-gnome
  python3-distutils python3-lib2to3 x11proto-core-dev x11proto-dev xorg-sgml-doctools xtrans-dev
The following packages will be upgraded:
  default-jre default-jre-headless evolution evolution-common evolution-data-server
  evolution-data-server-common evolution-plugins gir1.2-gdm-1.0 gir1.2-gnomedesktop-3.0
  gir1.2-lokdocview-0.1 gir1.2-rb-3.0 gnome-calendar gnome-contacts gnome-control-center
  gnome-control-center-data gnome-disk-utility gnome-orca gnome-session-bin gnome-session-common
  gnome-settings-daemon gnome-shell-common gnome-tweak-tool hplip-data libebackend-1.2-10
  libebook-contacts-1.2-2 libecal-1.2-19 libedata-book-1.2-25 libevolution libgdm1
  liblibreofficekitgtk libpython3-stdlib libreoffice libreoffice-avmedia-backend-gstreamer
  libreoffice-base libreoffice-base-core libreoffice-base-drivers libreoffice-calc
  libreoffice-common libreoffice-core libreoffice-draw libreoffice-evolution libreoffice-gnome
  libreoffice-gtk3 libreoffice-help-en-us libreoffice-impress libreoffice-java-common
  libreoffice-math libreoffice-ogltrans libreoffice-pdfimport libreoffice-report-builder
  libreoffice-report-builder-bin libreoffice-script-provider-bsh libreoffice-script-provider-js
  libreoffice-script-provider-python libreoffice-sdbc-hsqldb libreoffice-sdbc-postgresql
  libreoffice-style-tango libreoffice-writer libreofficekit-data librhythmbox-core10 libsane-hpaio
  libtotem0 printer-driver-hpcups python3 python3-apt python3-brlapi python3-cairo python3-cups
  python3-dbus python3-gi python3-gi-cairo python3-markupsafe python3-minimal python3-pil
  python3-pycurl python3-renderpm python3-reportlab-accel python3-smbc python3-uno rhythmbox
  rhythmbox-data rhythmbox-plugin-cdrecorder rhythmbox-plugins totem totem-common tracker
  tracker-extract tracker-miner-fs uno-libs3 ure
90 upgraded, 61 newly installed, 14 to remove and 132 not upgraded.
Need to get 26.9 MB/419 MB of archives.
After this operation, 461 MB of additional disk space will be used.
Do you want to continue? [Y/n] _
```

Install Java on Debian

Once the **Java** installation is complete, check the version of **Java** :

$ java -version

Step 2: Install Tomcat in Debian

We will create a new **tomcat** group and user to run the **Tomcat** service under **/opt/tomcat** directory (Tomcat installation).

```
$ sudo mkdir /opt/tomcat
$ sudo groupadd tomcat
$ sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
```

Now download the tarball and extract the archive to **/opt/tomcat** directory.

```
$ curl -O http://www-eu.apache.org/dist/tomcat/tomcat-9/v9.0.30/bin/apache-tomcat-9.0.30.tar.gz
$ sudo tar xzvf apache-tomcat-9*tar.gz -C /opt/tomcat --strip-components=1
```

Now, assign the permissions to **tomcat** user to have access to the Tomcat installation **/opt/tomcat** directory.

```
$ cd /opt/tomcat
$ sudo chgrp -R tomcat /opt/tomcat
$ sudo chmod -R g+r conf
$ sudo chmod g+x conf
$ sudo chown -R tomcat webapps/ work/ temp/ logs/
```

Step 3: Create Tomcat systemd Service

We will create a new **systemd** service file to manage and run **Tomcat** as a service under **systemd**.

```
sudo update-java-alternatives -l
```

we can create the **systemd** service file called **apache-tomcat.service** in the **/etc/systemd/system** directory by running.

```
$ sudo nano /etc/systemd/system/apache-tomcat.service
```

You can copy -paste the contents into your **apache-omcat.service** file.

/etc/systemd/system/tomcat.service

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking

Environment=JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
```

```
Environment='JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target
```

Now, reload the **systemd** to apply the new changes, so that it knows about **apache-tomcat.service** file.

```
$ sudo systemctl daemon-reload
```

STEP 3: Start Tomcat Service

$ sudo systemctl start tomcat / **start tomcat service**

$ systemctl status tomcat / **check status**

$ systemctl enable tomcat **/ enable tomcat**