# Ubuntu

18.04.3 LTS

## VirtualBox setup:

Previously, passing through the GPU on this laptop causes Gnome not to run. Needs further experimentation if I want to use GPU in Linux.

[https://www.virtualbox.org/manual/ch03.html#settings-screen](https://www.virtualbox.org/manual/ch03.html" \l "settings-screen)

VMSVGA with 3D acceleration: May require sudo ubuntu-drivers autoinstall

Paravirtualization: Stick with default. Probably doesn’t matter.

Enable VT-X/AMD-V

Enable Nested Paging

Max video memory

## Preferred Install Features

VirtualBox Guest Additions

htop

gnome-system-monitor

ubuntu-drivers autoinstall

## Performance

I noticed this thing runs my computer’s fans a lot when running in VirtualBox. Why?

**Specs**

My system has 16 GB RAM, 1.8Ghz 4cores

Dedicated to VirtualBox: 8 GB RAM, 1 CPU (0.5 core)

Let’s see if I can cool it down by giving it more cores. Parallelism might reduce its power load.

First, updating all software.

Also, performance monitoring so I can baseline and compare.

First method: sudo top

* h for help
* R to toggle sort order; x to highlight sort field
* f to change fields or change sorted field
  + Use arrows to select a field; space to turn the field on and off; s to set as the sorted field.
    - Holy cow. Dropbox uses 97 threads. It will definitely benefit from more CPU. Gnome uses 10.
  + No obvious way to sort by hard drive I/O.

Compare: sudo apt-get install htop

* Colors, easier menus, the ability to track HDD usage. Much better. In shell.

Compare: sudo apt-get install gnome-system-monitor

* Very pretty! My favorite for looks and function. However, I see when it is running that it eats about 30% of my CPU by way of gnome-shell and xorg. By contrast, top or htop use almost the same resources as each other.

Takeaway: Everything that redraws frequently in a window uses a lot of resources in Gnome. Why is it so much less efficient than Windows?

**Observed on startup:**

Dropbox and gnome-shell consuming a lot of CPU.

Dropbox reduced.

Gnome-software and packagekitd coming up now.

Now after about 15 seconds, resource usage went down.

So apparently: System may run slowly during a brief period after startup and when it’s behind on a lot of updates. Gnome always eats up 3-10% of one core. Worth it. BUT if you wiggle the mouse around the entire screen, gnome-shell can eat up 70% of the CPU! Insane!

Updated VirtualBox Guest Additions. Changed graphics to pass through access to 3D graphics, and set graphics to VBoxSVGA; still no improvement.

Read some documentation that VMSVGA is preferred.

**Result:**

Performance issues continue. Tolerate it. Maybe it’s a feature of running in VirtualBox, but more likely it’s a weakness of Gnome. It’s in the queue, but probably won’t be fixed until a later version.

There are alternatives to Gnome. Worth playing with another time.

**Conclusion:**

Run the workers of a compute farm without GUI. GUI is fine for a single developer machine.

## Install IntelliJ

Already done – I don’t have notes on this.

Note that you’ll have to cd /opt/idea… and sudo ./idea.sh to run it in a privileged mode where you can download updates. When you do, you may be running in a different context than you normally do when you launch Idea, and will have to make a dummy project just to load the program and get to where you can update.

## Hyper-V Setup:

### Installing:

Easiest way: Hyper-V  Quick Create

Pick your distro.

Start downloading.

### Preferences and Setup:

### Useful Commands:

apt search [name or regex]

### Other Stuff to Install:

sudo apt-get install vim

sudo apt-get install openjdk-11-jdk

sudo apt-get install scala

sudo apt-get install git

### Installing IntelliJ:

Go to the Software application.

Install IntelliJ Idea Community.

### Cloning the GitHub Repository

~> mkdir code

cd code

git clone scalatraining

Go down to the CentOs 8 section on GitHub and set up the user and everything else.

# CentOS 8

## Clean Install

Downloading CentOS 8: **CentOS** vs **CentOS Stream:** From what I gather, you don’t want Stream; it’s a mid-cycle piece of software that matters only if you want to be part of developing the next version of RedHat.

Start with CentOS.org , download the DVD, and verify the hash.

To verify, you will need 7Zip (Windows) or run sha256sum [filename] (Linux).

Make a bootable disk from it. Follow the instructions in that. Or mount it on a VirtualBox. See details of different install paths below.

## Once Installed:

Settings  Devices  Display: Join Displays; move the displays around until they are in an arrangement you like, and Keep Changes.

Go into Settings  Devices  Keyboard, roll all the way to the end, and add a shortcut Ctrl-Alt-T to launch gnome-terminal .

Add a shortcut Ctrl-Shift-Escape for gnome-system-monitor

In Settings  Keyboard, find the setting “Hide all Normal Windows” and set it to Super-D.

Settings  Power, switch blank screen to a longer time if you want. I prefer 15 minutes. Leave “Power Button” on Suspend.

When the computer is kept in a safe location, turn off Settings  Privacy  Automatic Screen Lock.

Update all software:

Purge Trash and Temporary Files: Automatically empty Trash and purge Temporary Files On. Purge after 5 days.

Update everything:

1. sudo yum update
2. sudo yum upgrade
3. reboot

Making it possible to run shell scripts from Nautilus:

1. Open Nautilus.
2. Click on the blue “Files” icon in the upper left hand corner of the Nautilus screen.
3. Select Preferences
4. Go to Behavior
5. Executable Text Files: Choose “Ask what to do”

### Problems with fwupd on Gnome Software Update:

SELinux can maybe block gnome software from checking from getting firmware update messages over dbus. It’s a known bug and may be fixed. To disable SELinux:

1. sestatus  Shows status of SELinux.
2. Disable SELinux temporarily:
   1. sudo setenforce 0
   2. sudo vim /etc/selinux/config
      1. Set the SELINUX to disabled
      2. Reboot
   3. Verify by running sestatus and getenforce
   4. To fix, I assume you set SELINUX to enforcing and reboot?
3. Doesn’t seem to solve the problem. But before putting SELinux back on, more Internet searching.
4. Try checking whether the following directory exists: /var/cache/fwupd ; if not, sudo mkdir /var/cache/fwupd and reboot.
5. No joy.
6. **Not solved yet.** Not a huge deal; use yum for firmware updates. But one day, I’d like to solve this.

## Other Stuff to Install:

Install Dash to Dock:

1. Method one: Search using yum list “\*dash\*” and then install the one you find.
2. Method two: Go to Software  Add-Ons  Shell Extensions and install it.
3. Click “Extension Settings.”

Also in Software  Add-Ons  Shell Extensions:

1. Note that there is no search tool and these are not in any particular order.
2. Turn on Window List.

Install Chrome: Can’t get from software list.

1. Search Google for “Install Chrome.”
2. Select the 64-bit .rpm
3. Open with Software Install.
4. Click “Install.” Enter SU credentials.

Install “Open Terminal Here” capability for Nautilus.

1. yum list “\*nautilus\*”  Look for gnome-terminal-nautilus
2. If not already installed, install it.

Install Scala.

1. sudo yum install scala

Install Yum Utilities:

1. sudo yum install yum-utils

## Bare Metal Setup:

Download Rufus. Just drop it in a folder somewhere where you can run it with Admin permissions; it’s a self-contained file that doesn’t need actual install.

Pick a thumb drive with more than 8 GB – at least 16. Pop that in. It should become the default choice for a destination for Rufus.

For “boot selection,” select the ISO file for CentOS.

Partition scheme “MBR”

Target system: “BIOS or UEFI”

File System: Large FAT32 (Default)

Cluster Size: 32K (Default)

Hit Start. It may ask you to download some more recent components so you can install the most up-to-date Linux. Say “Yes.”

DO NOT INTERRUPT UNTIL COMPLETE.

Pop the USB into the target computer.

Figure out the right key to press – typically F12 on Dell – to get to boot options.

Boot from the thumb drive.

Choose “Install CentOS 8.”

1. English
2. Keyboard: English (US)
3. Language Support: English (United States)
4. Time & Date: U.S., Los Angeles
5. Installation Source: Local Media
6. Software Selection: Server with GUI
   1. Windows File Server
   2. Debugging Tools
   3. File and Storage Server
   4. Network File System Client
   5. Network Servers
   6. Performance Tools
   7. Remote Management for Linux
   8. Virtualization Hypervisor
   9. Virtualization Tools
   10. Container Management
   11. Development Tools
   12. Graphical Administration Tools
   13. Headless Management
   14. RPM Development Tools
   15. Scientific Support
   16. Security Tools
   17. System Tools
7. Installation Destination: BE VERY CAREFUL. Make sure you don’t overwrite your Windows system if that’s not what you intend to do.
   1. ASMT 2115 … : The external USB to SATA adapter.
   2. Reclaim space.
      1. When deleting sectors, verify that they look like what you expected to be there.
      2. In this case, deleting an install of CentOS 7. As expected.
   3. Automatic partitioning.
   4. KDUMP Enabled
   5. Connect to Wi-Fi
   6. Go back to Time: Configure NTP. Use the default CentOS pool.
   7. Warning: Processor has Simultaneous Multithreading (SMT) enabled. Click for details.
      1. Essentially, this is a security vulnerability that might allow one process to break out of its cage and escalate. Turning it off has a performance impact. I choose to tolerate it in this context of an expendable system used for development and big data stuff.
      2. In practice, if becoming serious about the use of a system, turn off SMT. Just leave it on, on expendable “worker” systems.
   8. Security Policy: No content found. Leave it like that until we figure out what to do with this later.
8. Begin installation!
9. Root password and user: Per Returnables. For now, make the user an Admin.
10. Pop the thumb drive.
11. Reboot.
12. Hit F12 and select the new CentOS system.
13. Accept the license.
14. Don’t connect to any online accounts.

## VirtualBox setup:

8 GB RAM

Dynamic Hard Drive, VDI; 128 GB HDD

VMSVGA with 3D acceleration: May require driver installs. *How to do on CentOS 8?*

Paravirtualization: Stick with default. Probably doesn’t matter.

Enable VT-X/AMD-V

Enable Nested Paging

Max video memory

Attach the DVD to the DVD drive of the VM and start it.

See Navigating to a DVD or CD-ROM.

### Install Process:

Choose English.

If it doesn’t work, start over and pick different installed options. There’s no point trying to repair it.

gnome-system-monitor

Found that CPU is significantly less loaded in CentOS than in Ubuntu.

## On VirtualBox:

After install, pick the appropriate VirtualBox Guest Additions version to match your current version of VirtualBox: <http://download.virtualbox.org/virtualbox/>

Insert the ISO and let it autorun. It may fail; pay attention to where it saves the failed logfile and see why! Probably you will need to find a missing library:

cd [location where log file named in error is saved]

vi [filename]

yum find “\*libname\*”

sudo yum install [libraryname]

Then re-run VirtualBox additions:

Cd /run/media/[username]/VBoxXXXXXXXX

sudo ./VBoxLinuxAdditions.run

## Installing DropBox

**Conclusion so far: Give up on Dropbox on CentOS 8.**

Go to the Dropbox download page. The download for Fedora will want to open in Software Installer, but will probably not work. It won’t be obvious why. So in a shell:

(Go to the download folder.),….

sudo rpm -ivh nautilus-dropblahlblahlblah

Found: libgnome >= %(gnome\_version) is needed by nautilus dropblahblah

[So far can’t solve this. Does that %(gnome\_version) refer to an environment variable that’s not set?]

Maybe that just won’t work in Centos 8.

Download the latest installer package: It appears that it was version numbers for a while, but now it’s year-month-day. That’s the latest.

Extract to /tmp

Go into the extracted folder in shell.

./configure; make; make install;

[Doesn’t work.]

## Installing IntelliJ

sudo yum install java-latest-openjdk

* May not exist. In that case, type:
* yum list “\*openjdk\*”
* Pick a version you like.
* sudo yum install (the version you liked)

Go to the Downloads page and download the Community edition. Save file as a TAR.GZ. Unpack to /opt/

* To unpack as admin when using Nautilus, unpack to admin:///opt/  It will ask for admin credentials. It may ask again partway through the operation.

Go to admin:///opt

If there’s a subfolder in the extracted folder, drag that back to /opt/

Rename the folder with all the version numbers to simply “idea.”

Launch by going into /opt/idea/bin/idea.sh , this time NOT as admin. It will ask you about creating links, shortcuts, scripts, etc.; say yes to all.

In optional plugins, be sure to add Scala tools.

That’s it!

go to File  Settings → Keymap and select "Default for GNOME" in the Keymap dropdown list.

### Installing Your Favorite Java

Download the OpenJDK. Extract and copy to /usr/lib/java .

Add your favorite Java to the path:

sudo vim /etc/profile.d/pathmod.sh

Add:

export PATH=/usr/lib/java/[your jdk version]/bin:$PATH

### Getting the right Scala in Your Project

File  Project Settings  Libraries  Click +  Scala SDK  Download  Pick the version you want.

## Using Pluralsight

Launch in Firefox. Videos won’t run. After two tries, click “Still having problems?”

Found an error playing HLS and MP4 video.

Gstreamer not available for Firefox on Linux.

Could possibly enable nux-desktop to get third-party support for various media? Not sure it works with CentOS 8.

Following the instructions on https://www.tecmint.com/enable-nux-dexktop-repository-on-rhel-centos/

It doesn’t add as expected on CentOS 8. In any case, the repo is just the first step; you’d still have to know what things you intended to install out of it.

**Pluralsight works with Chrome.**

## Tips

### Gnome

#### Workspaces

To move a window to another workspace: Ctrl-Alt-PgUp or PgDn.

To switch to another workspace: Ctrl-Alt-Up or Down.

### Nautilus

#### Admin in Nautilus

There’s nothing you can install for this in CentOS. What you do is:

1. Type Ctrl-L to switch from a path to address in the top of Nautilus.
2. Path to admin://[path], e.g., admin:///etc
3. May still not be useful for everything you want.

#### Rename File

F2

### File Operations

#### Deleting Tree

rm -r [folder]

sudo rm -r [folder]

#### Navigating to a DVD or CD-ROM

cd /run/media/[username]

#### Rename File

mv [oldname] [newname]

### System Configuration

#### Environment Variables and PATH

Paths added to PATH are binaries that can be launched by name without specifying a path to them. To make them persist, you might be tempted to edit the /etc/profile file.

sudo vim /etc/profile

However, that file might be changed in system updates, wiping out your customizations. That’s why the header of the file recommends instead going another way:

1. Create a PATH modification script:

sudo vim /etc/profile.d/pathmod.sh

Press “I” for “insert”

Add lines in the format of:

export PATH=$PATH:[new location being added to PATH]

(if adding to the end) or

export PATH=[new location being added to PATH]:$PATH

When you’re done, hit Escape

Type :wq for “write changes and quit.”

### Tools

#### Gedit

Gedit is the graphical editor for Gnome. You can open it from command line with gedit [path to file]

You can’t sudo gedit, but you can gedit admin://[path to file including additional initial slash] . If you don’t want to type the whole path from scratch, you can use gedit admin://$(pwd)/[filename] – it should graphically prompt you for the root password.

You can do code highlighting inside gedit – it should be automatic, but if it’s not, you can choose View  Highlight Mode. For BASH shell scripts, use highlight mode “sh”.

#### Yum Utilities

If you want to know where something got installed, like Java:

* repoquery -list [package name]

# Git

## Connecting to GitHub

### Installing/Building Git

First: Make sure you have all the dependencies.

sudo dnf install dh-autoreconf curl-devel expat-devel gettext-devel \

openssl-devel perl-devel zlib-devel

Won’t work! dh-autoreconf depends on cdbs, which is an Ubuntu thing. Just take whatever version of Git you can get from yum and be happy.

sudo yum install git

### Initial Configuration

The Git username and email are not the same as the GitHub username and email. It’s a username that’s used to identify you when you make commits. You are just identifying yourself. You could conceivably pick anything.

git config --global --add user.name “Joe Pride”

git config --global --add --user.email “jhpride1790@gmail.com”

git config --global --add core.editor “gedit”

git config --edit –global

git config --global -l

### Git Commands

git config  configure the tooling

git init  initialize a local repo

git clone  download a project from a remote repo

git add  prepare a file (to staging)

git commit  commit a file to the repo

git config --global core.excludesfile ~/.gitignore\_global  Sets this gitignore file to apply globally to this computer.

* \*.idea

git rm --cached FILENAME  Untrack a file that’s already been committed to a repo.

Put a .gitignore file in the root of your repo  Apply it to that repo. If you commit it, then it will apply for everyone.

#### Basic Workflow

git pull  To pull down and merge the latest version of the current branch (presumably master).

git add [filename]  To add or update a specific file.

git add .  To add or update everything new or changed.

git commit -m “message"

git push -u origin master

#### Branching

git branch [branch-name]  Create branch locally.

git checkout [branch-name]  Switch to a branch.

git checkout -b [branch-name]  Create branch locally and switch to that branch.

git add .  Add the new changes to the branch.

git commit -m “Commit message.”

git push -u [origin] [branch]

Log into GitHub to create a pull request from the branch to its parent branch.

### Gitting Started with Git Service

cd ~/code

git init DemoApp

cd DemoApp

gedit README.md

git add README.md

git commit

git push remote master

vim intro.txt [:wq]

git add .

git commit

git status

### Gitting Started with GitHub

#### Creating SSH Key:

ssh-keygen -t ecdsa -b 521 -C "JoePride"

Let it pick the default place to save this.

Enter a passphrase and record it somewhere.

Go to that location and open the .pub version of the key. Copy the entire document.

Log into GitHub. Go to Profile  Settings  SSH and GPG Keys.

New SSH Key. Name it for the location where you use it, so you can maybe delete it later when you have to.

Paste the entire body of the key.

ssh -T git@github.com

Enter the passphrase.

Browse to the location of your cloned repository, and then do the following:

git remote set-url origin git@github.com:JoePride/scalatraining.git

(Or whatever the equivalent URL is of the repo you’re looking at.)

Verify with:

git remote -v

Connect one time with:

ssh -T [git@github.com](mailto:git@github.com)

Ignore the error message. You should now be able to do git things without re-entering your password.

#### Creating GitHub Token:

1. Log into GitHub.com in a browser
2. Go to Settings
3. Go to Developer Settings
4. Go to Personal Access Tokens
5. Generate New Token
6. Select Scope:
   1. Generally, I prefer to set the token to never expire. Bad practice, I know, but since I use a different one per machine, it’s easy enough to turn it off when I no longer need it.
   2. Permissions:
      1. repo:
         1. repo:status
         2. repo\_deployment
         3. public\_repo
   3. Click “Generate Token”
   4. Copy the token: You’ll never see it again! Save it somewhere.
   5. One option to save the token for future use, as long as you don’t care about it being stored locally in plain text:
      1. git config credential.helper store
      2. The next time you use your password, this should result in it being stored.
   6. When doing a git push or other command, use your regular username and then paste the token as the password.

#### Linking to GitHub Remote Repository:

git remote -v  Determine whether there is a remote repository for this directory.

git remote add origin [https://github.com/[me]/[repo].git](https://github.com/%5Bme%5D/%5Brepo%5D.git)

git remote add origin git@github.com:JoePride/scalatraining.git

git clone [https://github.com/[me]/[repo].git](https://github.com/%5Bme%5D/%5Brepo%5D.git)

git clone git@github.com:JoePride/scalatraining.git

git push -u origin master

#### Setting Up a Git-Ignore

Inside GitHub, at the appropriate level of your repo, add a file titled .gitignore . Each ignore gets its own line.

Important ignores:

Scala compile files:

* \*.class
* \*.log

IntelliJ Idea files:

* .idea/
* .idea/\*
* \*.iml

IntelliJ Target Files:

* \*/target/\*

Microsoft Office temp files:

* ~$\*.\*
* ~\*.tmp

Make sure to “Pull” to your local machine before you do a git add . ; this makes sure the git-ignore will be followed.

### Setting up Git with IntelliJ

#### Making a Scala Project that Can Compile Different Apps

#### Joining an Existing Scala Project to Version Control

1. Go to the VCS menu  Enable Version Control
2. Select a version control system: Git

#### Starting a Project from an Existing Git Repository

1. File  New  Project from Version Control
2. If the repository has changed, VCS  Git  Pull

# Running Scala

## Preparing an SBT Project

### Adding a JDK:

To download automatically:

1. Click the drop-down on the JDK currently in use and go to the bottom: + Add JDK
2. Choose Download
3. Choose the version you want and the type of source. Note that the latest may not be available in the IntelliJ download lists.

To add manually:

### Creating the Project:

1. Go to File → New Project.
2. Select Scala → SBT Project
3. Name the project
4. Choose a path inside of the GIT repo folders you want to use.
5. Choose the Scala and Java versions.
   1. Note that certain versions of Scala require certain versions of Java. See https://docs.scala-lang.org/overviews/jdk-compatibility/overview.html
   2. Note also that generally the older versions of Java still work with the newer versions of Scala. Generally. It’s probably still best to use the matched versions shown on the compatibility table.
6. Create the project.