

Ruby on Rails Setup using VirtualBox

Contents

Install VirtualBox	1
Install Ubuntu.....	2
Install Chrome	10
Configure the Task Bar	12
Install Git with SSH	14
Install Git	14
Configure Git	15
Create a new repository on GitHub	17
Configure SSH.....	19
Installing Node and NPM on Ubuntu	23
Install cURL.....	23
Install NPM.....	23
Install Node	24
Installing the Heroku CLI	25
Install Atom	26
Installing Ruby and Ruby on Rails	27
Installing RVM on Ubuntu	27
Install Gems.....	28
Install PostgreSQL.....	30

Install VirtualBox

To setup our environment and create our virtual machine we will first need to install free software called VirtualBox

1. <https://www.virtualbox.org/>
2. Download the latest Version

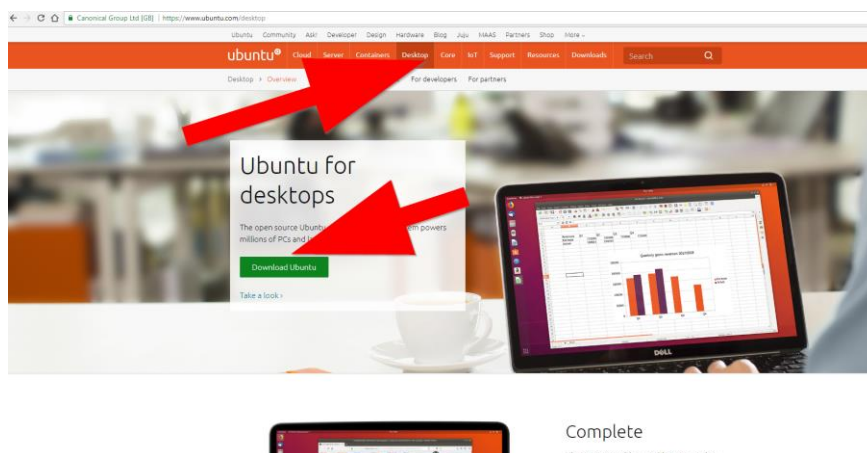


1. Once Downloaded, double click on the file and click “Next”
2. Click on “Next” for each step and click on “Install” and “Finish” to finish the installation

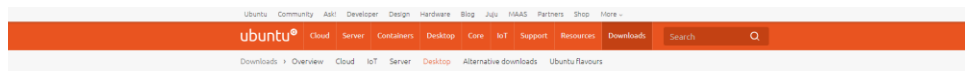
Install Ubuntu

The operating system I will be setting up is Unbuntu which is free to download and install. If you prefer there are other Linux flavours which you could use. As we have setup VirtualBox you could create multiple environments with different operating systems to find the one you prefer.

1. <https://www.ubuntu.com>
2. Select the desktop menu item
3. Download Ubuntu



4. Download the LTS Version. LTS stands for Long Term Support and is the latest stable version



Download Ubuntu Desktop

Ubuntu 18.04 LTS

Download the latest LTS version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years, until April 2023, of free security and maintenance updates, guaranteed.

[Ubuntu 18.04 LTS release notes](#)

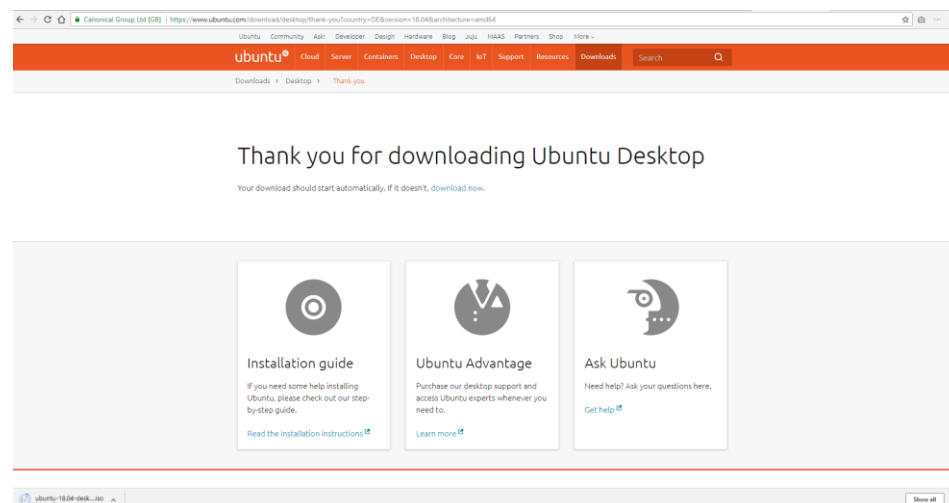
Recommended system requirements:

- 2 GHz dual core processor or better
- 2 GB system memory
- 25 GB of free hard drive space
- Either a DVD drive or a USB port for the installer media
- Internet access is helpful

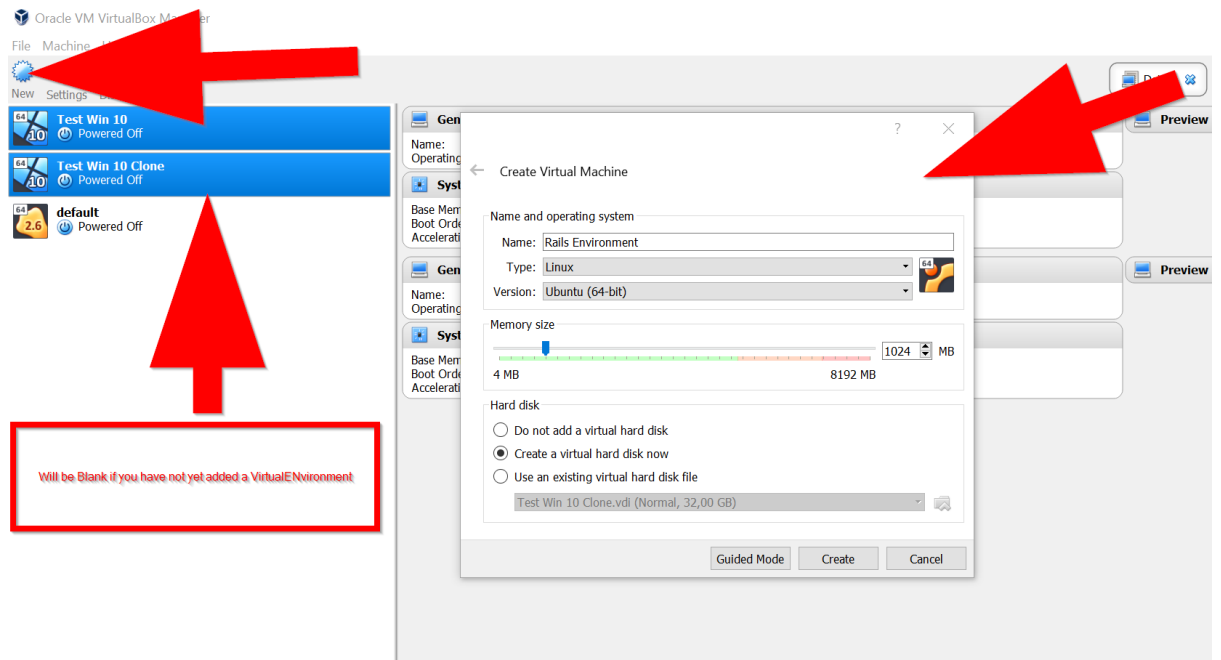
[Download](#)

For other versions of Ubuntu Desktop including torrents, the network installer, a list of local mirrors, and past releases see our [alternative downloads](#).

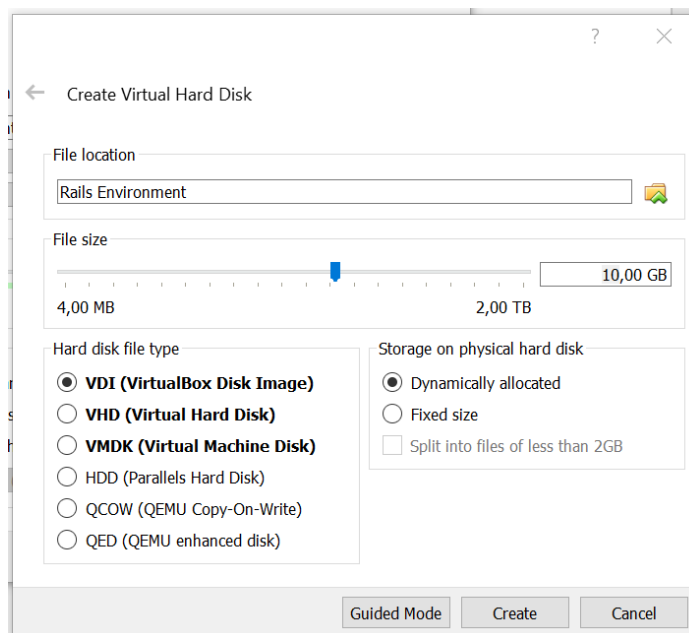
5. When clicking on Download you will be redirected to the “Help shape the future of Ubuntu Page” where they will ask for a donation. Click on “Not now, take me to the download” or the Pay with PayPal button to make a donation.
6. The installation file will start downloading:



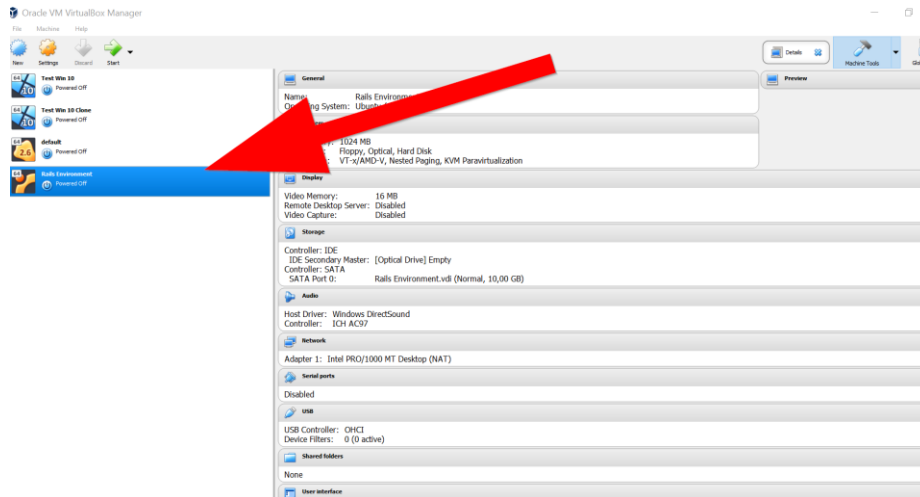
7. After the file has downloaded, open VirtualBox and New Environment:
 - a. If you have not yet added any Virtual Environments, the list will be empty
 - b. Click on the New Button to add a new Environment.
 - c. In the popup window, Give the Environment a name: e.g.: Rails Environment
 - d. Type: Linux
 - e. Version: Ubuntu(64-bit)
 - f. Leave the other settings as default. Depending on your PC i.e. processor, hard drive and memory you would need to adjust the respective options accordingly. Make sure that the settings are sufficient to run the Virtual Machine and that it does not slow down your local PC. These settings can always be changed at a later stage.
 - g. Click on Create to create the new Virtual Machine



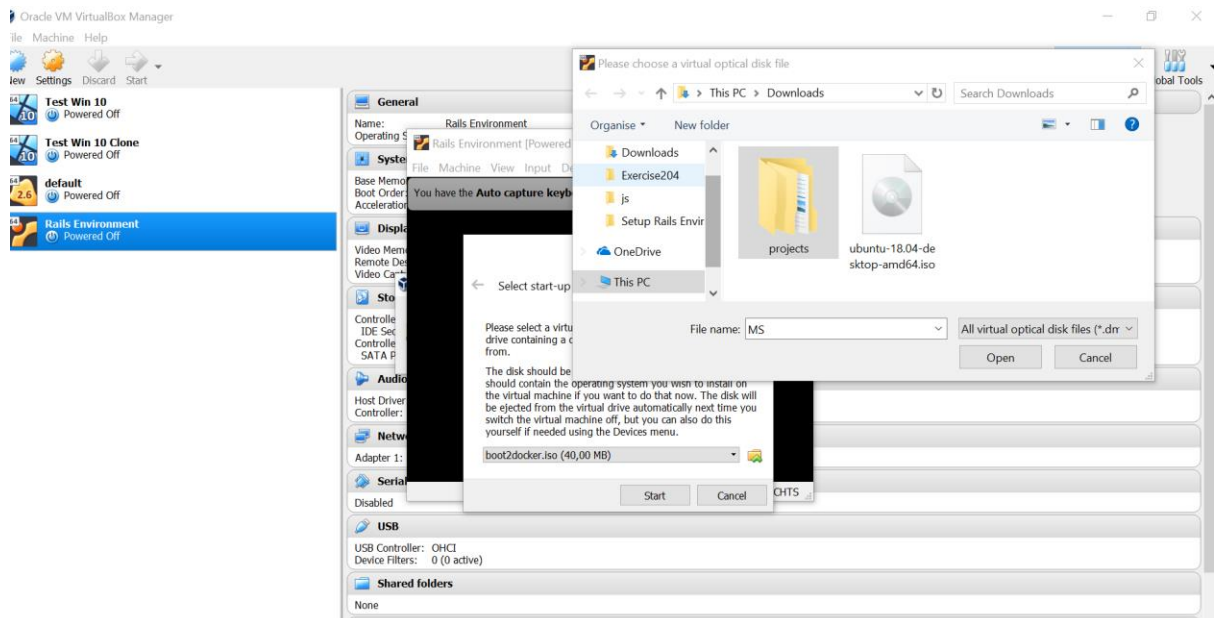
8. A Popup screen will appear, leave the Default settings but if you want to change the file location or size, this can be changed to your requirements:
9. Click Create to create the Virtual Environment



10. Double click on the VM Titled Rails Environment

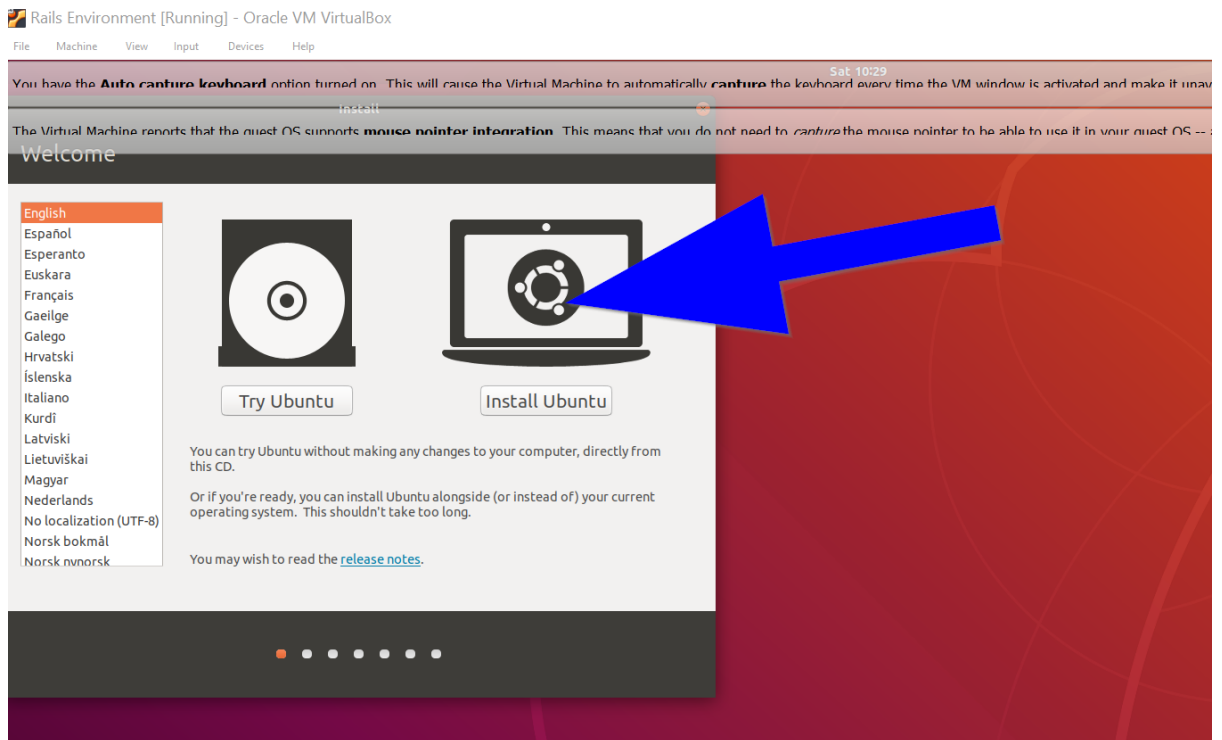


11. Because we have no operating system installed we will get a popup asking us to select a start up disk. From this screen select the location of the Ubuntu file we downloaded eg: ubuntu-18.04-desktop-amd64.iso

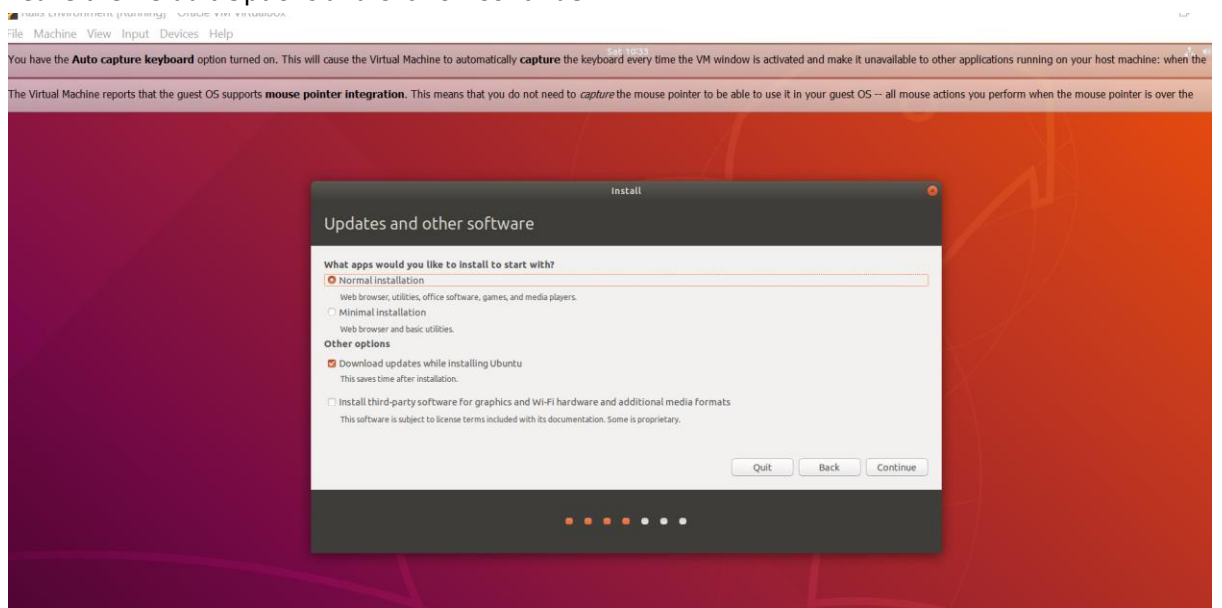


12. Select the File and click on Start to start the installation of Ubuntu

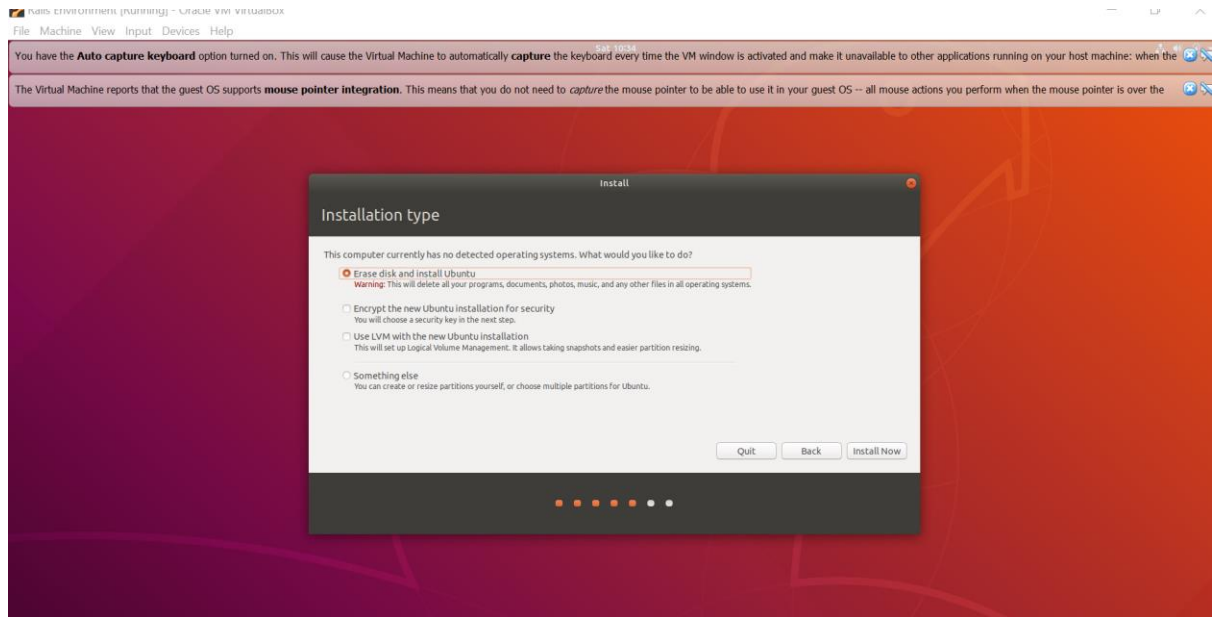
13. When the is available click on Install Ubuntu.



14. Select the keyboard layout and select Continue
15. Leave the Default Options and Click on Continue

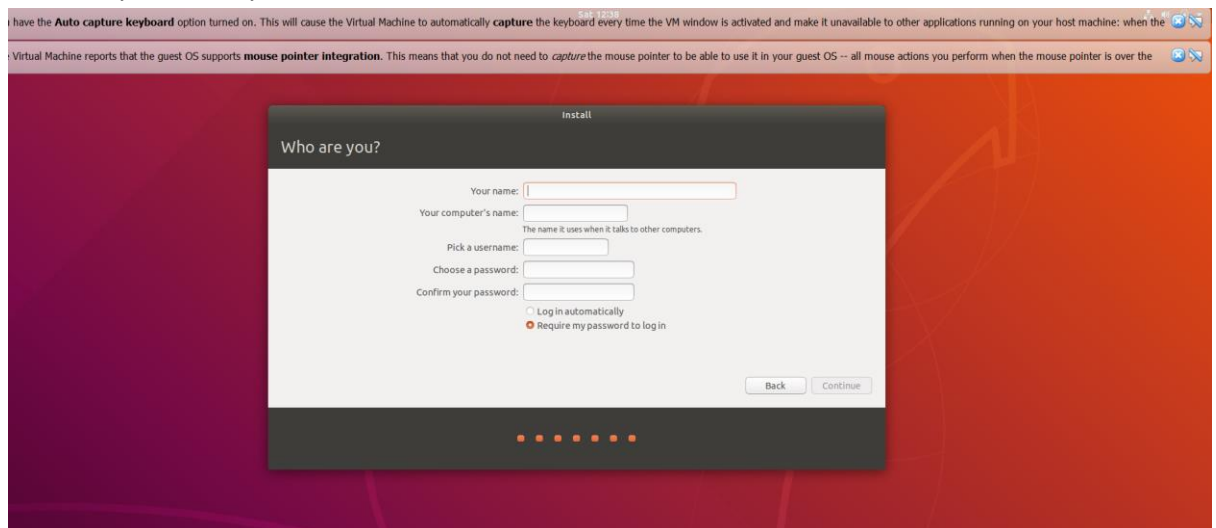


16. When asked for the installation type leave the default settings. The Erase Disk and Install option is not your local disc that will be erased, it is the virtual disk

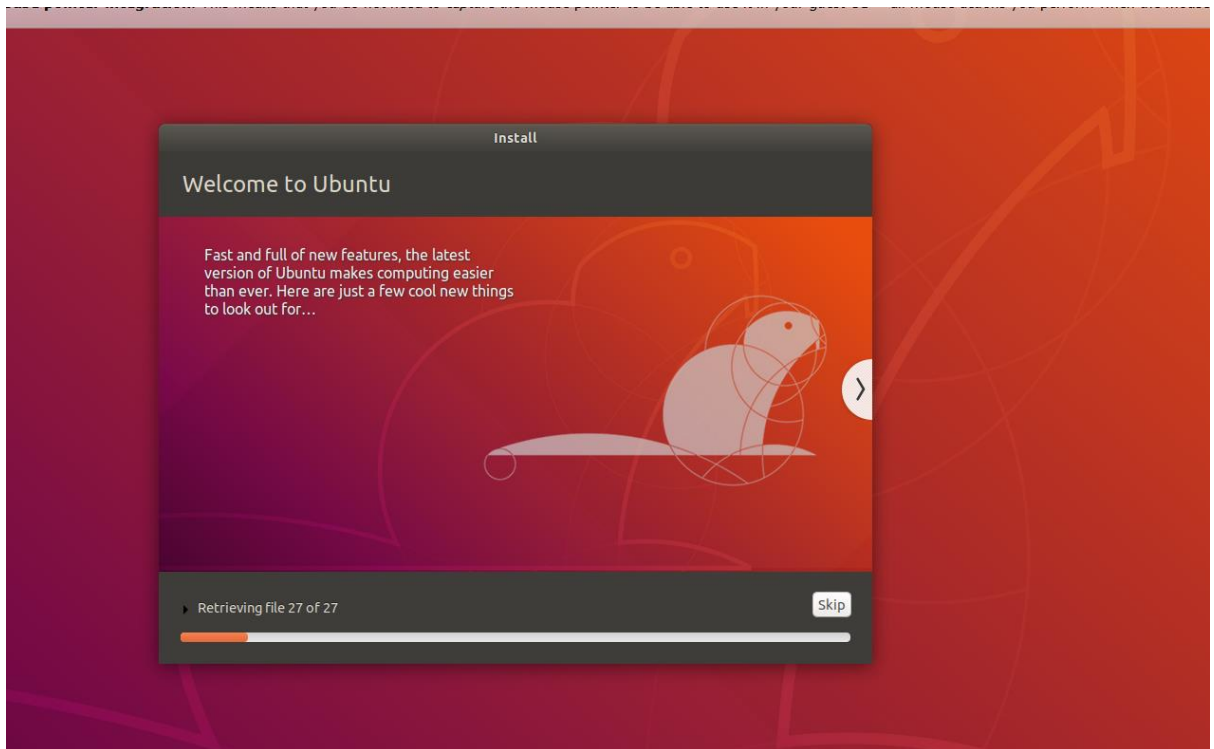


17. Write the Changes to Disk – Select Continue and Select your Region

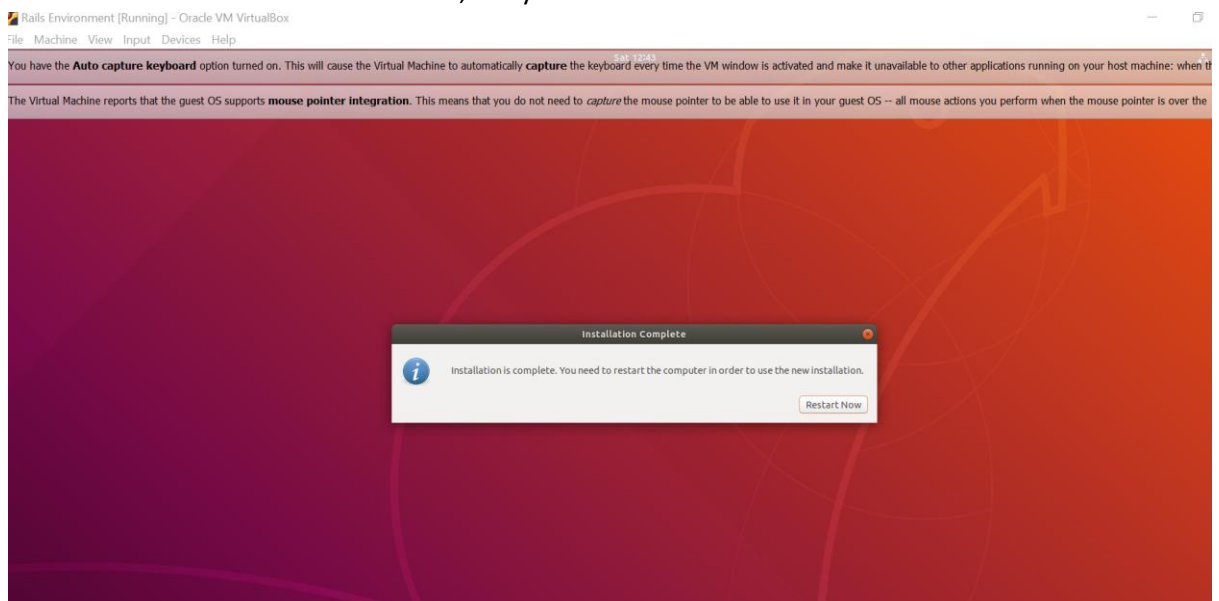
18. Who are you: Enter your details and click Continue:



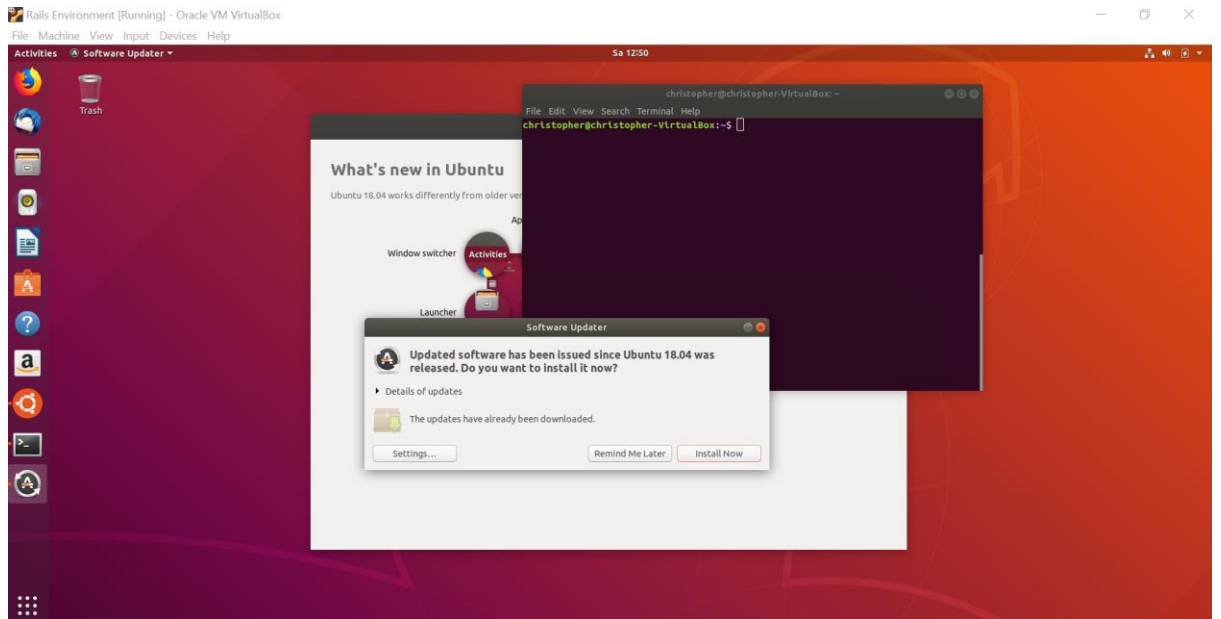
19. This will start the installation of the Operating System



20. When the installation is completed, you will get a message asking you to Reboot the System, this is a reboot of the Virtual Machine, not your PC. Click on Restart Now



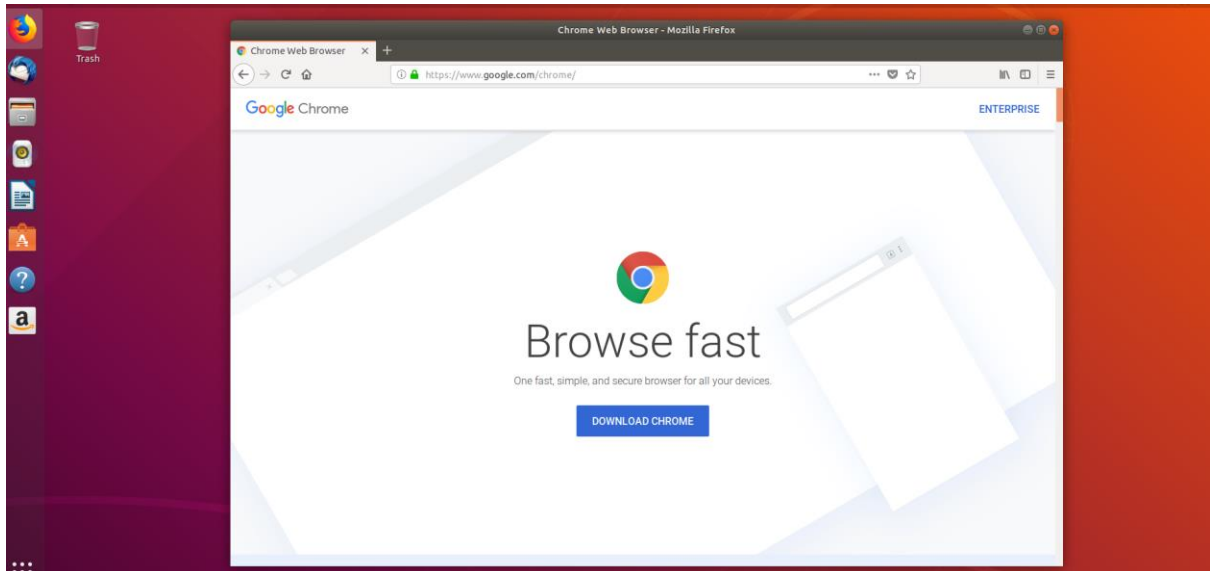
21. After the System has rebooted, enter your password to log in. Once logged in explore the desktop and open the terminal to see if your keyboard is working correctly.
22. If you get a message that there are software updates, you can click on Install Now to install them



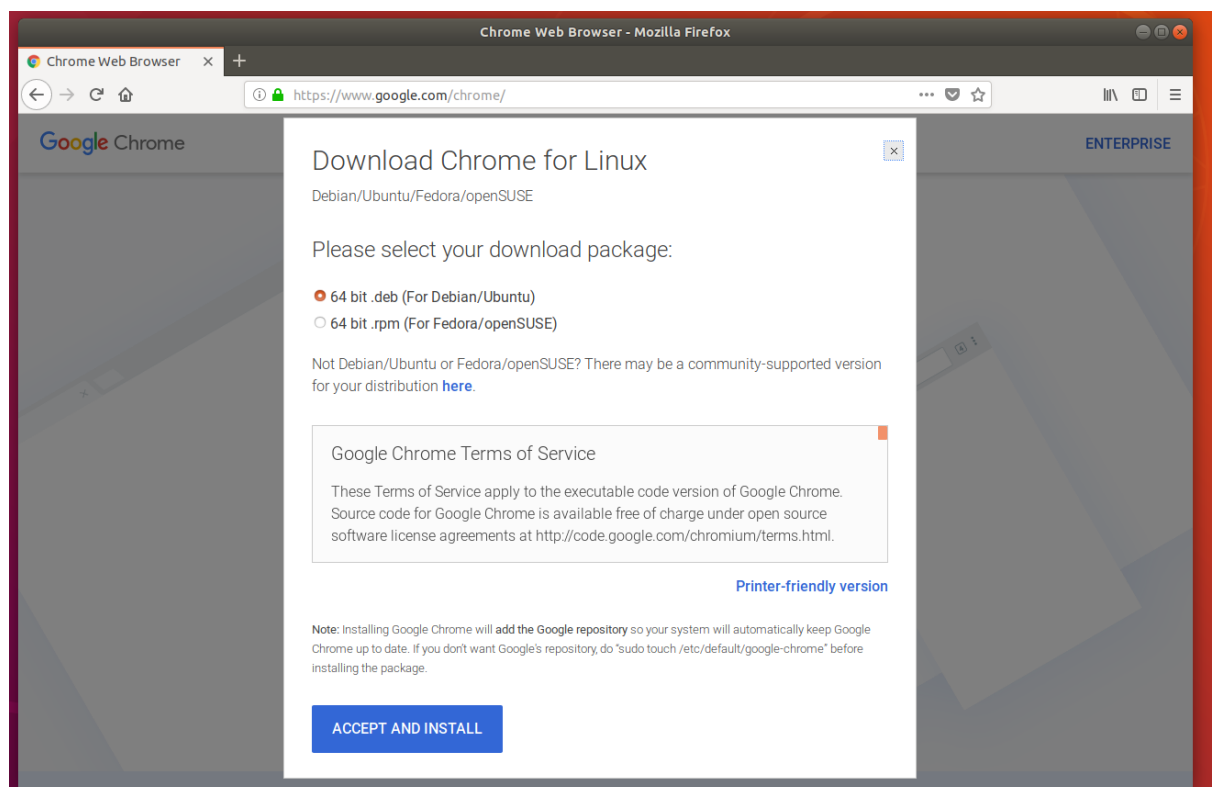
Install Chrome

Firefox should be installed as default but if you prefer Chrome you can install it as follows:

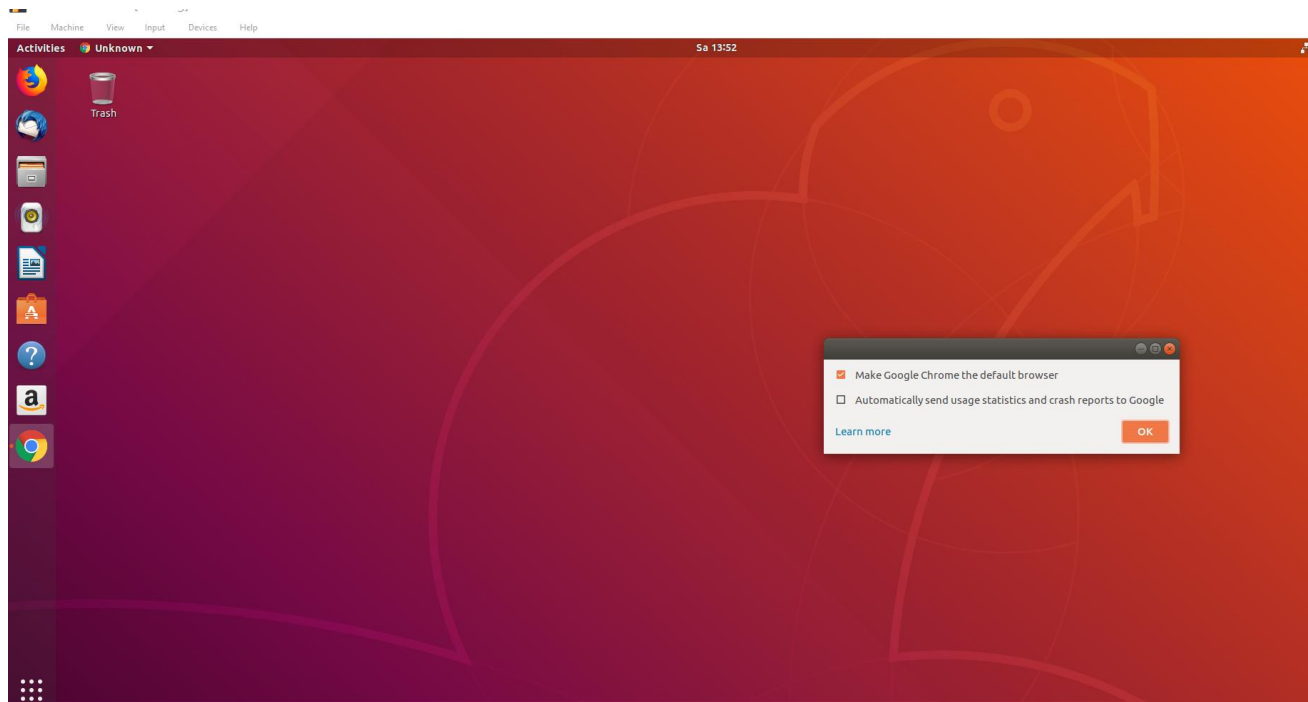
1. Open Firefox and navigate to www.google.com/chrome
2. Click on Download Chrome



3. Select the 64 bit .deb (For Debian/Ubuntu) version and click on Accept and Install



4. Follow the prompts and add your password when requested
5. Open Google Chrome and Set it as the default browser.

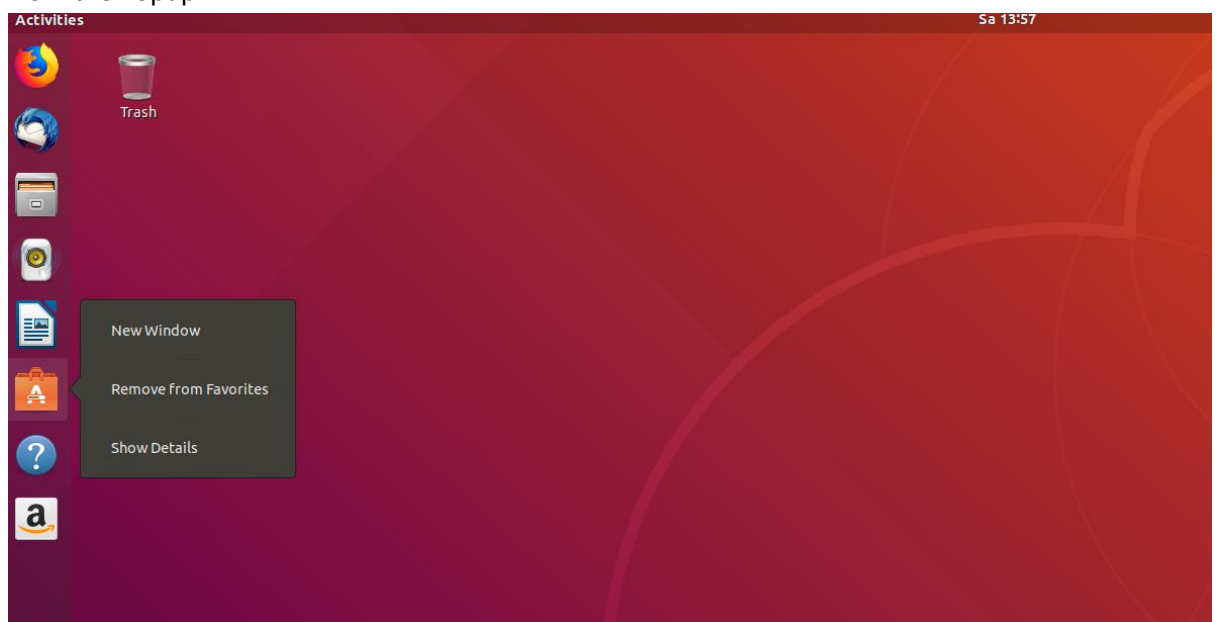


Configure the Task Bar

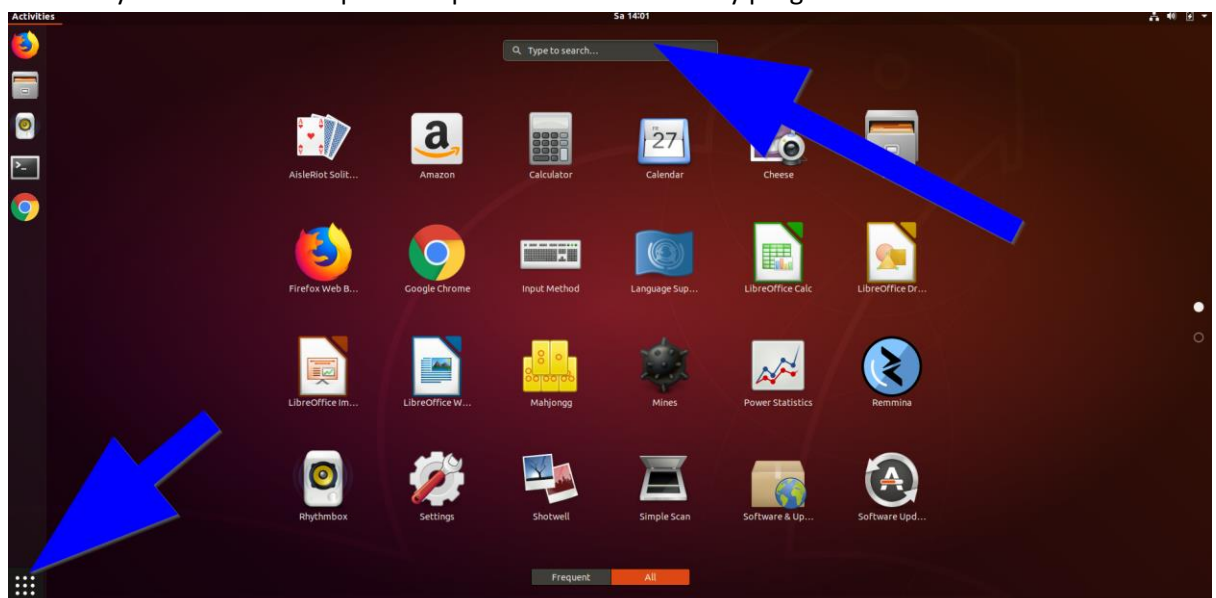
*If your VM is slow shut the system down and change the system settings such as Memory and CPU in VirtualBox. Do not assign too much memory to the VM or else it will slow down your PC but this will all depend on your PC specification.

To make it easier to navigate our system we can edit the programs that are listed on our Task Bar. As we will be using Chrome and the Terminal more than other programs I will add them to the Task Bar.

1. Select unwanted icons from the Task Bar and right click, and select Remove from Favourites from the Popup



2. To add new programs to the Task Bar, select the windows icon or press the windows key on your keyboard. Enter the program name in the search field, e.g. Terminal and right click to add it to your favourites. Repeat this process for all necessary programs



terminal



Terminal

New Terminal

Add to Favorites

Show Details



Ubuntu Software
13 more

Terminator Multiple terminals in one window

MATE Terminal Use the command line

Terminal icon GNOME Shell Extension

Terminal As Root Button GNOME Shell Extension

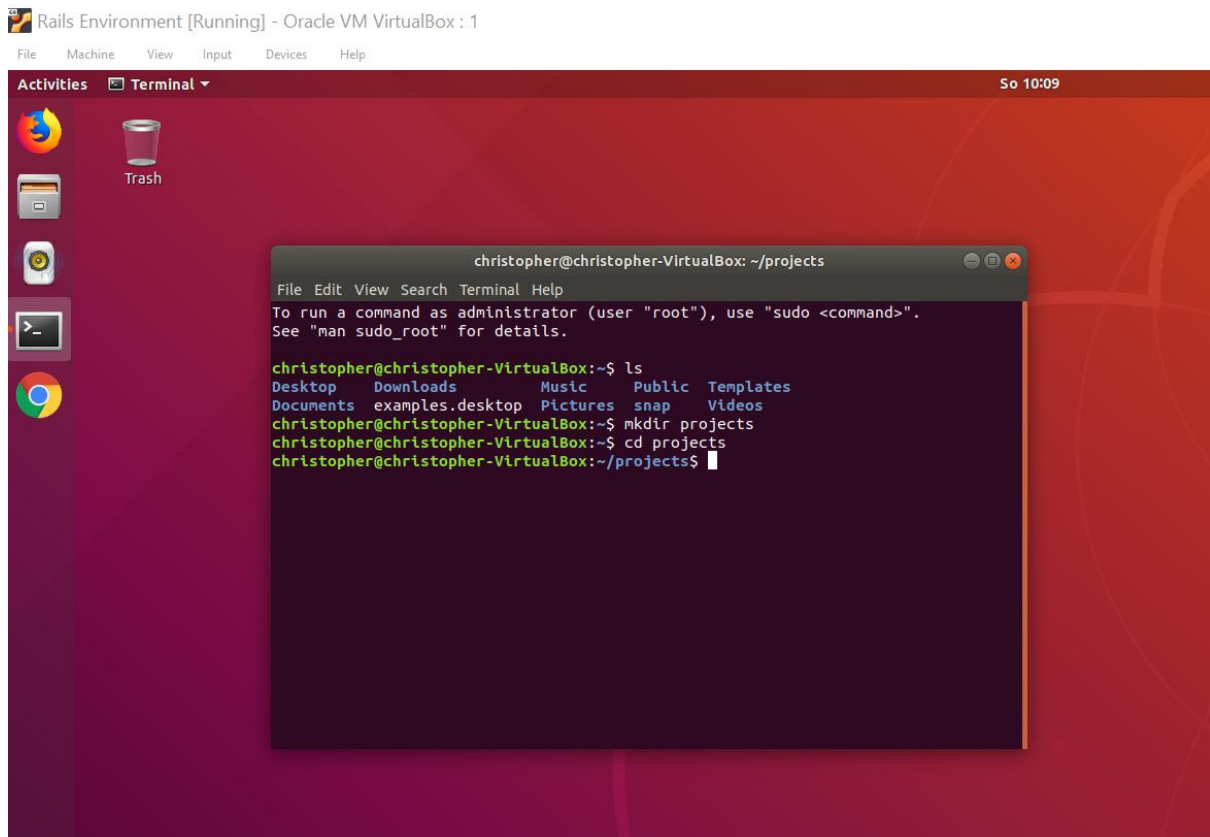
Remove <Alt>F2, autocompletion in Overview... [and other cool things] GNOME Shell Extension

Install Git with SSH

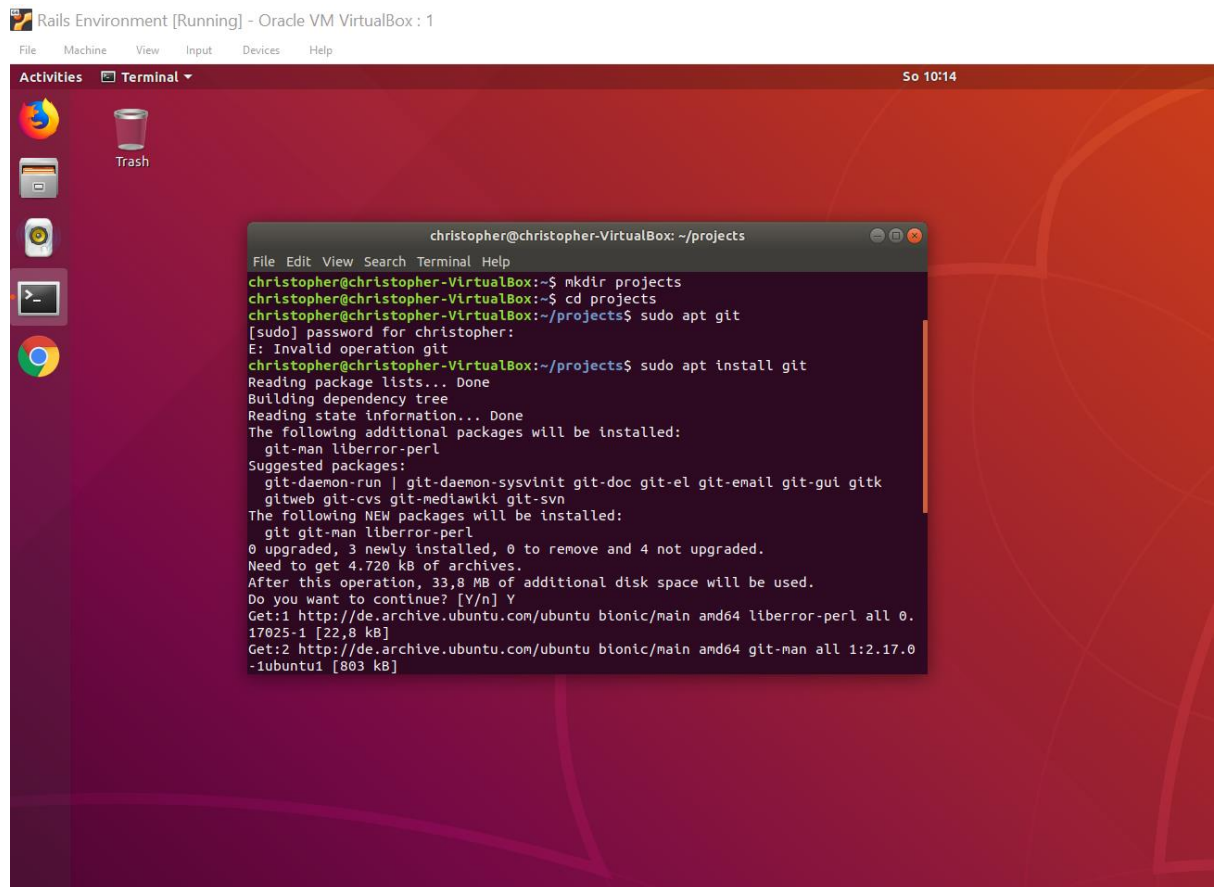
In this section we will install git, and create a test project which we will publish to GitHub using HTTPS or SSH. To complete these steps you will need to register with GitHub

Install Git

1. Open Terminal
2. Create a New Directory for our rails projects called projects
 - a. `mkdir rails`
3. change directory
 - a. `cd projects`



4. to install git type `sudo apt install git` and enter your password when asked



Configure Git

git will now be installed and can be tested by creating a test directory in our projects folder

1. `mkdir test`
2. `cd test`
3. `git init`
4. create a new file called README.MD
 - a. `touch README.ME`
5. list directory settings
 - a. `ls`
6. check the status of your git repository
 - a. `git status`

```

christopher@christopher-VirtualBox: ~/projects/test
File Edit View Search Terminal Help
Setting up liberror-perl (0.17025-1) ...
Processing triggers for man-db (2.8.3-2) ...
Setting up git (1:2.17.0-1ubuntu1) ...
christopher@christopher-VirtualBox:~/projects$ git status
fatal: not a git repository (or any of the parent directories): .git
christopher@christopher-VirtualBox:~/projects$ mkdir test
christopher@christopher-VirtualBox:~/projects$ cd test
christopher@christopher-VirtualBox:~/projects/test$ git init
Initialized empty Git repository in /home/christopher/projects/test/.git/
christopher@christopher-VirtualBox:~/projects/test$ touch README.MD
christopher@christopher-VirtualBox:~/projects/test$ ls
README.MD
christopher@christopher-VirtualBox:~/projects/test$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        README.MD

nothing added to commit but untracked files present (use "git add" to track)
christopher@christopher-VirtualBox:~/projects/test$

```

- b. git status shows us that README.md is not being tracked so we need to the project by typing:
 - i. git add .
- c. if you type git status it will now say : No commits yet
- d. to commit the file type:
 - i. git commit -m "Initial Commit"
- e. The message Initial Commit could be any message you would like to add
- f. Because we have not associated our Github account with Git we Should get a message saying : Please tell me who you are

```

christopher@christopher-VirtualBox: ~/projects/test
File Edit View Search Terminal Help
christopher@christopher-VirtualBox:~/projects/test$ git commit -m "Initial Commit"
*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

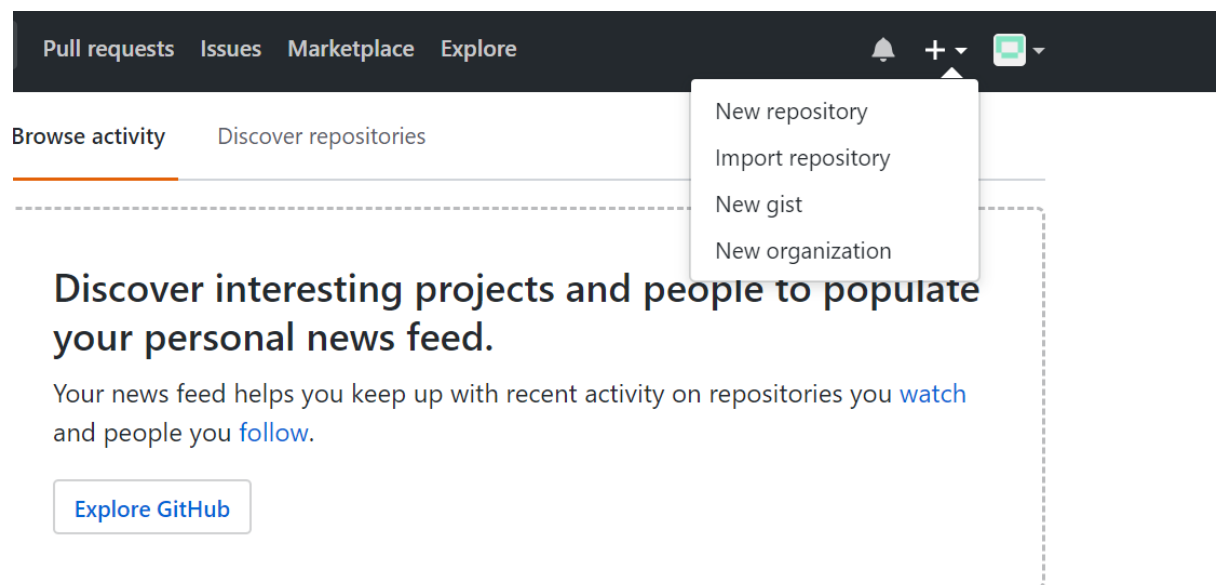
fatal: unable to auto-detect email address (got 'christopher@christopher-VirtualBox.(none)')
christopher@christopher-VirtualBox:~/projects/test$

```


7. To associate your Github account type:
 - a. Git config --global user.email "[name@address.com](#)"
 - b. Git config --global user.name "Name Surname"
8. If you type git commit -m "Initial Commit" your file will be committed and can be confirmed by typing git status
9. If you do not want to type the full sentence out again, press the up arrow to toggle through the previous commands

Create a new repository on GitHub

1. Navigate to <https://github.com/> and click on New repository button in the top left corner



2. Enter the required information:
 - a. Repository Name: test **Give it the same name as our project, it can be different but I prefer to give it the same name.**
 - b. Description: Give it a description or leave it blank
 - c. Click on Create Repository

https://github.com/new

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: wallandall / Repository name: test

Description (optional):

Public (selected) | Private

Initialize this repository with a README

Add .gitignore: None | Add a license: None

Create repository

- This will create the repository as well as provide you with the commands to push our local files to GitHub

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/wallandall/test.git

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/wallandall/test.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/wallandall/test.git
git push -u origin master
```

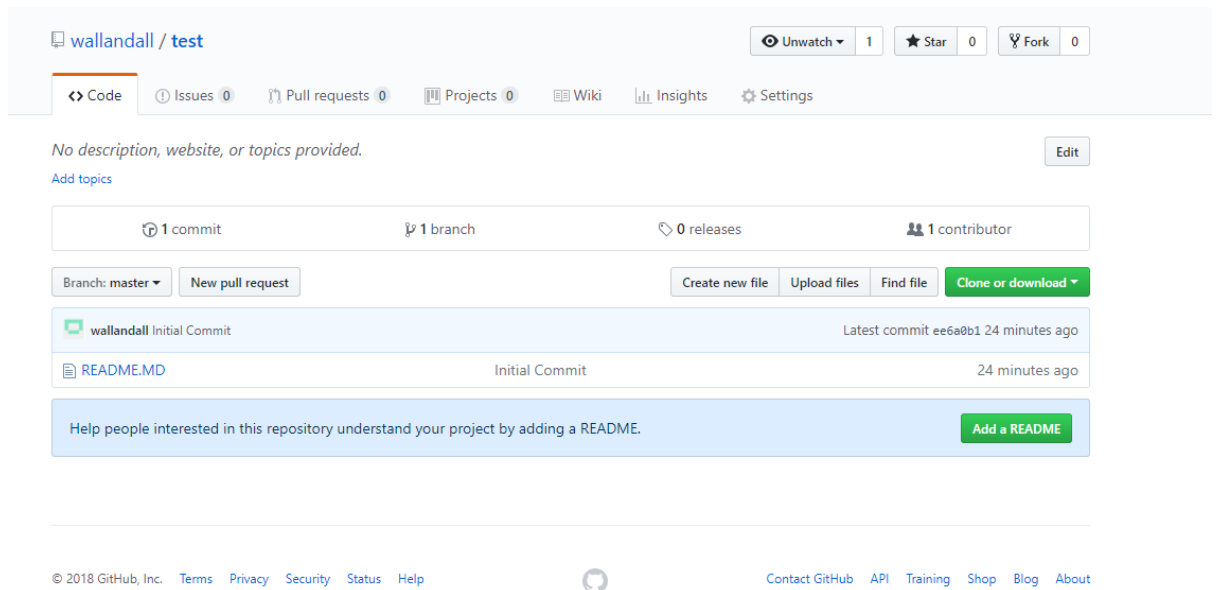
...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

ProTip! Use the URL for this page when adding GitHub as a remote.

- To push our files to GitHub type:
 - `git remote add origin https://github.com/wallandall/test.git`
 - `git push -u origin master`
 - *Make sure the path corresponds to your repository**
- When pushing your files you will be asked for your username and password
- Once you have pushed your files you can go back to GitHub and refresh the page to view the files we have just pushed.

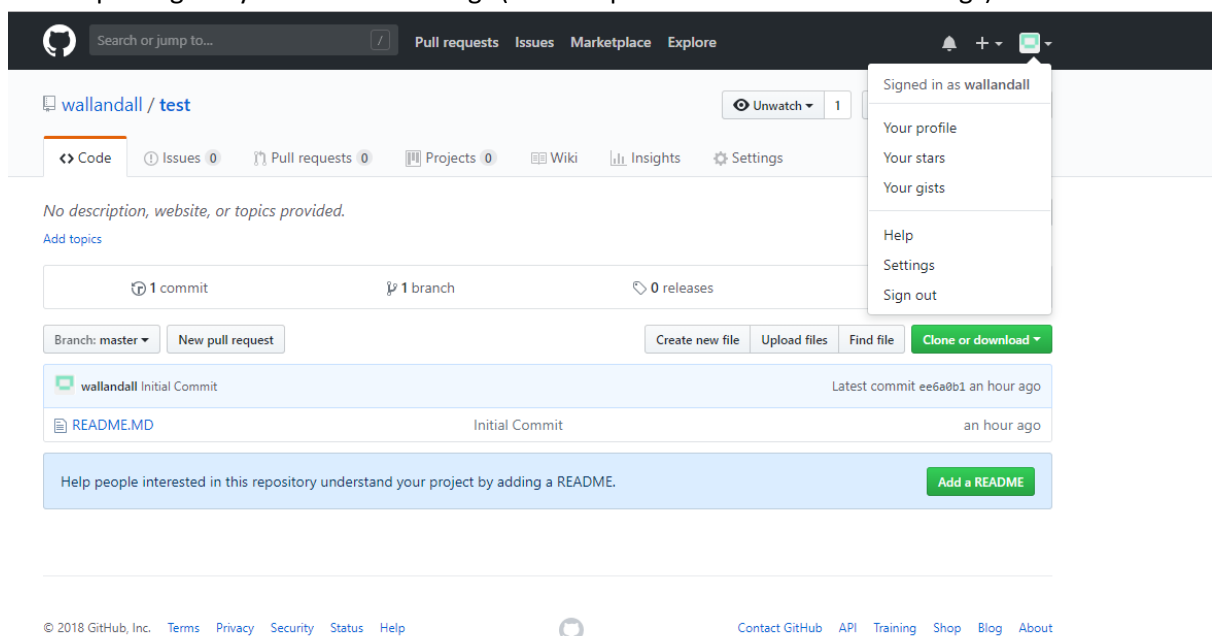


7. If you type git status you will see all your files are up to date.

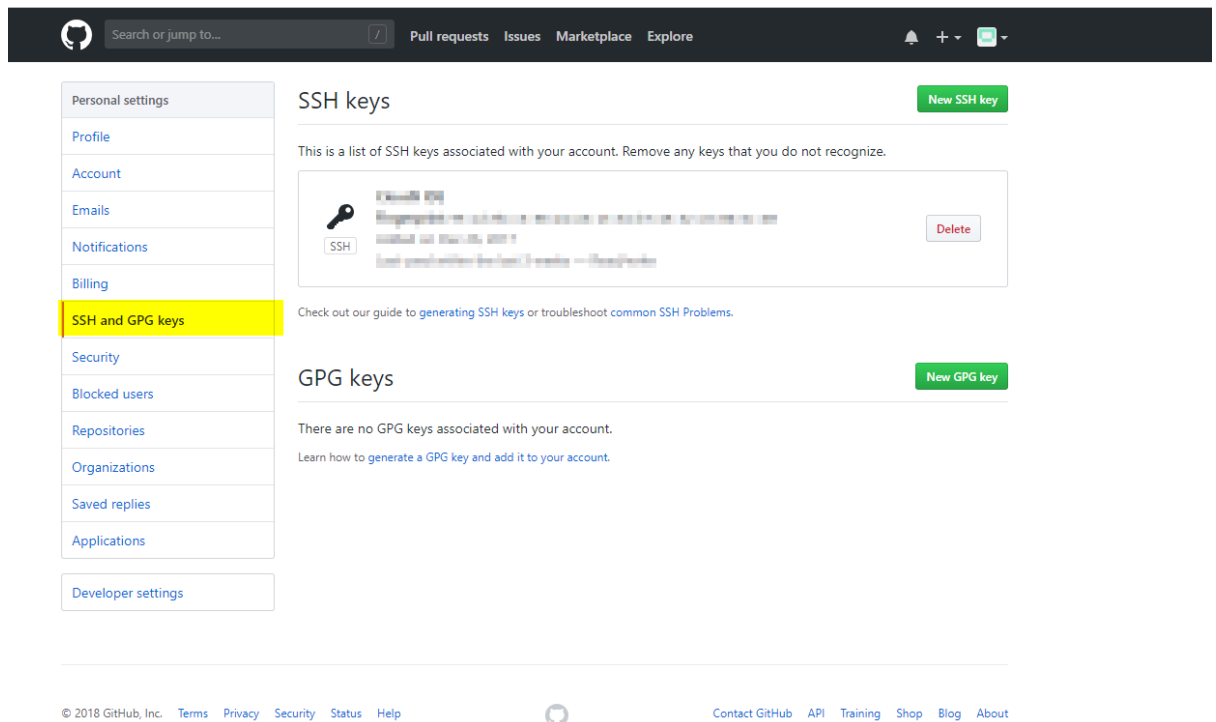
Configure SSH

We have created a test directory, initialised git and pushed our files to GitHub but we pushed our files over HTTPS. This is worked perfectly but is not as secure as SSH, if you would prefer to use SSH you can use the following steps or continue to use HTTPS

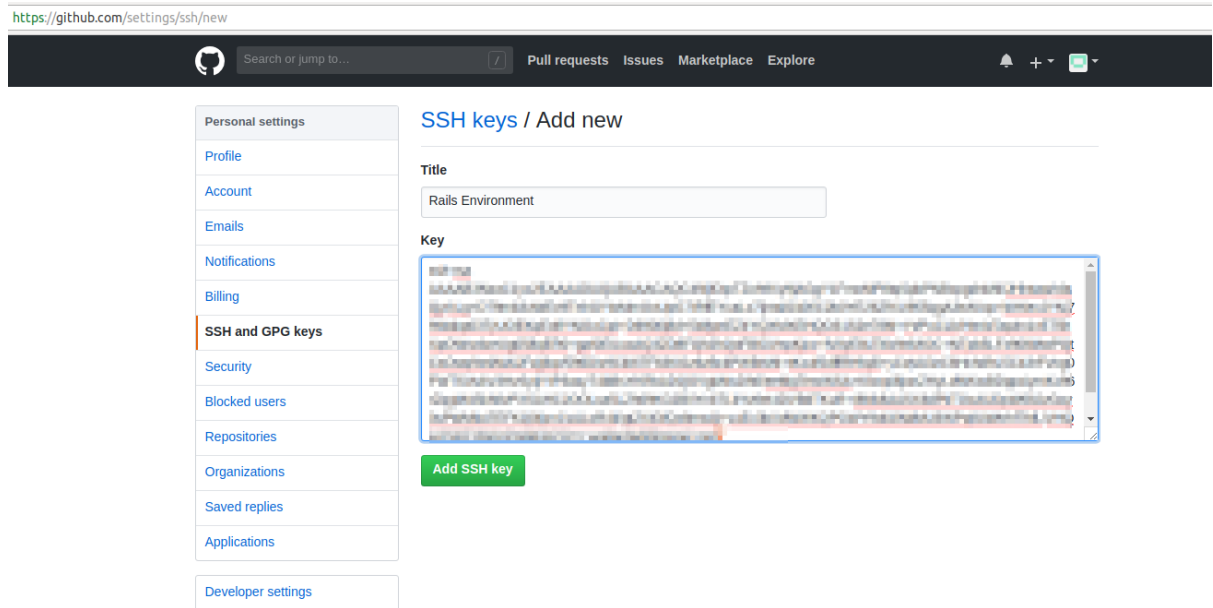
1. To setup SSH go to your account settings (in the top left corner and select settings)



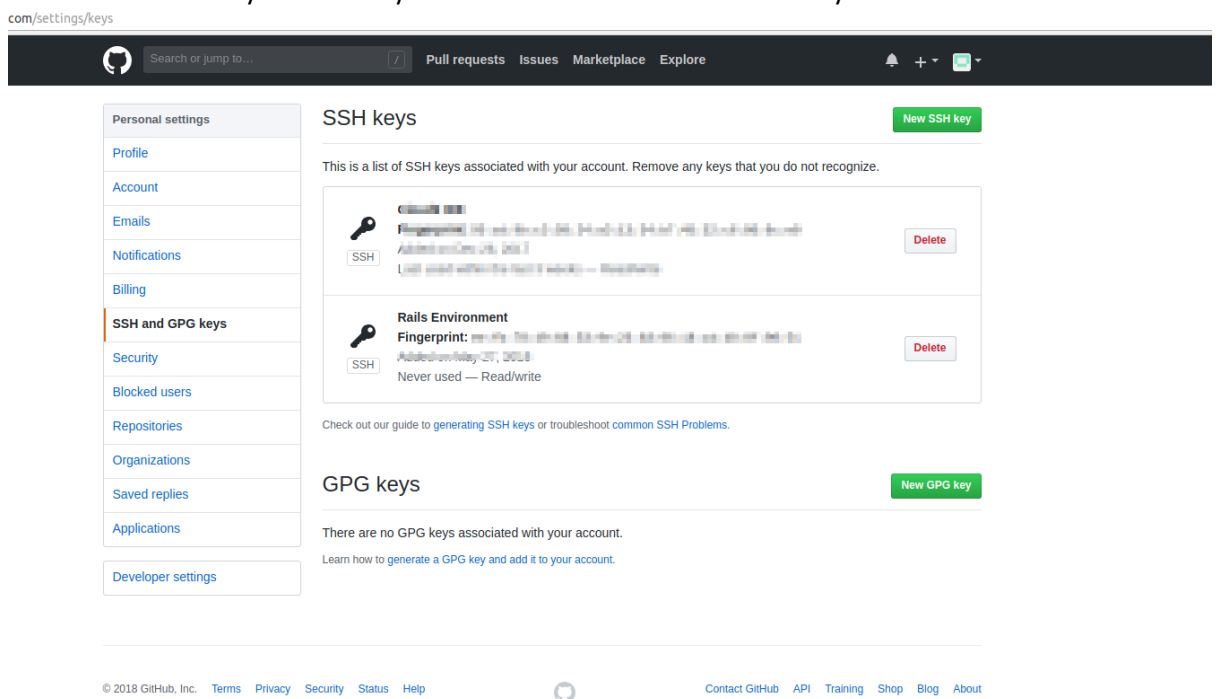
2. Under setting click on SSH and GPG key



3. To generate new SSH Key, run the below command in the Terminal on your Virtual Machine
 - a. `ssh-keygen -t rsa -b 4096 -C "your_email@example.com"`
 - b. Make sure you enter the email address you used when you registered
 - c. Press enter for both the file and passphrase
4. Add your SSH Key to the ssh-agent by typing:
 - a. `eval $(ssh-agent -s)`
 - b. `ssh-add ~/.ssh/id_rsa`
5. Add the key to your GitHub account by typing:
 - a. `sudo apt-get install xclip`
 - b. `xclip -sel clip < ~/.ssh/id_rsa.pub`
6. Go back to the GitHub settings page and click on New SSH Key
7. Give the Key a title
8. Paste the key we generated into the Key Field
 - a. ***the key was copied to our clipboard when we ran `xclip -sel clip < ~/.ssh/id_rsa.pub` so you need this step needs to be done from your Virtual Environment.**
9. Click on Add SSH Key



10. You will notice that your new Key has been added to SSH and GPG Keys



11. We can now push our code to GitHub using SSH

12. Notice the URL will be different :

- a. `git remote add origin github.com/wallandall/test.git`
- b. `git push -u origin master`

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Insights](#) [Settings](#)

Quick setup — if you've done this kind of thing before

Set up in Desktop

 or

HTTPS

SSH

https://github.com/wallandall/test.git

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/wallandall/test.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/wallandall/test.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

ProTip! Use the URL for this page when adding GitHub as a remote.

Because we have setup SSH, we do not need to add a password when pushing our code to GitHub. Because of this it is important not to share your SSH Keys

Installing Node and NPM on Ubuntu

To securely install Node we will be using NPM (Node Package Manager)

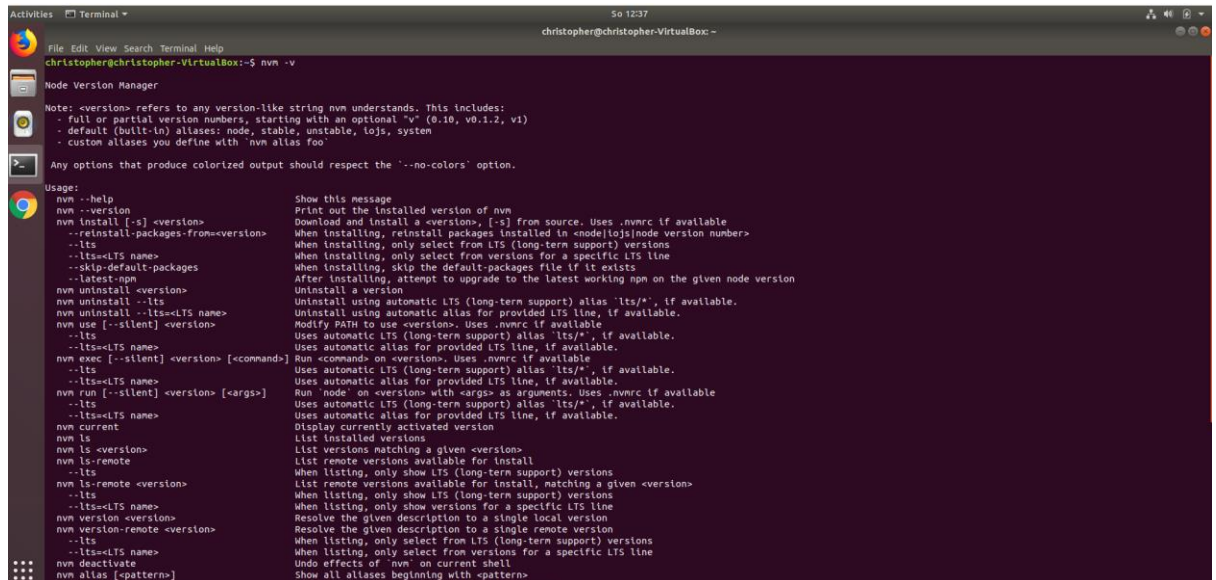
Install cURL

cURL is a command line tool used for transferring files. We will need to install it to get NVM

1. To install cURL type:
 - a. `sudo apt-get install curl`
 - b. Enter Password when prompted

Install NPM

1. `curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.11/install.sh | bash`
2. To check if nvm was installed correctly type:
 - a. `Nvm -v`
 - b. If you get an error close the terminal and reopen it



```
christopher@christopher-VirtualBox:~$ nvm -v
Node Version Manager

Note: <version> refers to any version-like string nvm understands. This includes:
- full or partial version numbers, starting with an optional "v" (0.10, v0.1.2, v1)
- default (built-in) aliases: node, stable, unstable, iojs, system
- custom aliases you define with 'nvm alias foo'

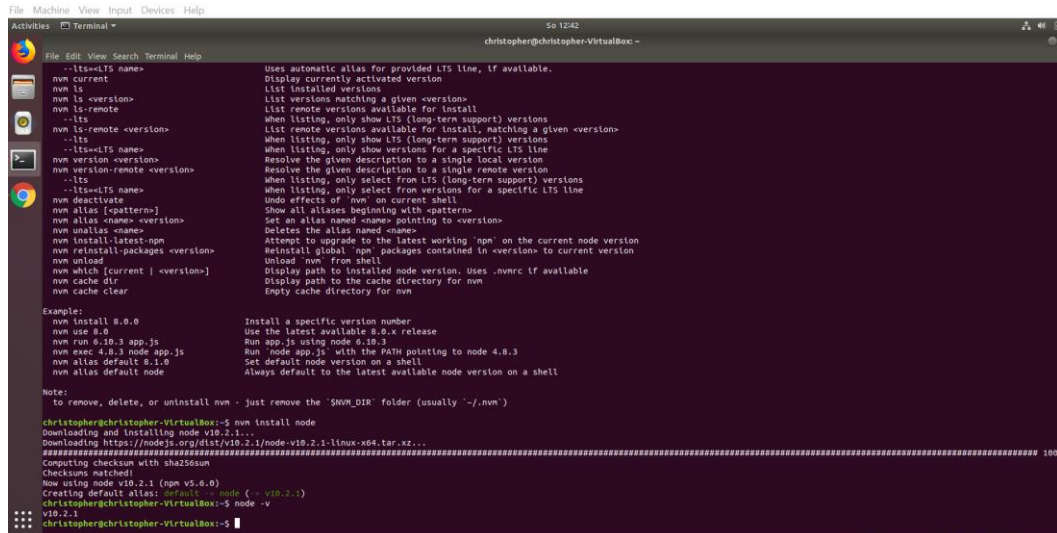
Any options that produce colored output should respect the '--no-colors' option.

Usage:
nvm --help                Show this message
nvm --version             Print out the installed version of nvm
nvm install [-s] <version> Download and install a <version>, [-s] from source. Uses .nvmrc if available
                          When installing, reinstall packages installed in <node>[iojs]node version number>
                          When installing, only select from LTS (long-term support) versions
                          When installing, only select from versions for a specific LTS line
                          When installing, skip the default-packages file if it exists
                          After installing, attempt to upgrade to the latest working npm on the given node version
nvm reinstall-packages <version> Reinstall packages installed in <node>[iojs]node version number>
nvm --lts                 When installing, only select from LTS (long-term support) versions
nvm --lts <LTS name>      When installing, only select from versions for a specific LTS line
nvm --skip-default-packages When installing, skip the default-packages file if it exists
nvm --latest-npm          After installing, attempt to upgrade to the latest working npm on the given node version
nvm uninstall <version>  Uninstall a version
nvm uninstall --lts       Uninstall using automatic LTS (long-term support) alias 'lts/*', if available.
nvm uninstall --lts <LTS name> Uninstall using automatic alias for provided LTS line, if available.
nvm use [--silent] <version> Modify PATH to use <version>. Uses .nvmrc if available.
nvm use --lts <LTS name>    Uses automatic LTS (long-term support) alias 'lts/*', if available.
nvm use --lts <LTS name>    Uses automatic alias for provided LTS line, if available.
nvm exec [--silent] <version> [<command>] Run <command> on <version>. Uses .nvmrc if available.
nvm exec --lts <LTS name>   Uses automatic LTS (long-term support) alias 'lts/*', if available.
nvm run [--silent] <version> [<args>]     Run 'node' on <version> with <args> as arguments. Uses .nvmrc if available.
nvm run --lts <LTS name>     Uses automatic LTS (long-term support) alias 'lts/*', if available.
nvm run --lts <LTS name>     Uses automatic alias for provided LTS line, if available.
nvm current                Display currently activated version
nvm ls                    List installed versions
nvm ls <version>          List versions matching a given <version>
nvm ls-remote             List remote versions available for install
nvm ls-remote <version>   When listing, only show LTS (long-term support) versions
nvm ls-remote <version>   List remote versions available for install, matching a given <version>
nvm --lts                 When listing, only show LTS (long-term support) versions
nvm --lts <LTS name>      When listing, only show versions for a specific LTS line
nvm version <version>     Resolve the given description to a single local version
nvm version-remote <version> Resolve the given description to a single remote version
nvm --lts                 When listing, only select from LTS (long-term support) versions
nvm --lts <LTS name>      When listing, only select from versions for a specific LTS line
nvm deactivate            Undo effects of 'nvm' on current shell
nvm alias [<pattern>]     Show all aliases beginning with <pattern>
```

Install Node

To install Node type:

1. nvm install node
2. once installed type node -v to check the version



```
File Edit View Search Terminal Help
christopher@christopher-VirtualBox: ~
--lts=<LTS name>          Uses automatic alias for provided LTS line, if available.
nvm current               Display currently activated version
nvm ls                   List installed versions
nvm ls <version>          List versions matching a given <version>
nvm ls-remote             List remote versions available for install
--lts                     When listing, only show LTS (long-term support) versions
nvm ls-remote <version>  List remote versions available for install, matching a given <version>
--lts                     When listing, only show LTS (long-term support) versions
--lts=<LTS name>          When listing, only show versions for a specific LTS line
nvm version <version>    Resolve the given description to a single local version
nvm version-remote <version> Resolve the given description to a single remote version
--lts                     When listing, only select from LTS (long-term support) versions
--lts=<LTS name>          When listing, only select from versions for a specific LTS line
nvm deactivate           Undo effects of 'nvm' on current shell
nvm alias [<pattern>]    Show all aliases beginning with <pattern>
nvm alias <name> <version> Set an alias named <name> pointing to <version>
nvm unalias <name>       Deletes the alias named <name>
nvm install-latest-npm    Attempt to upgrade to the latest working 'npm' on the current node version
nvm reinstall-packages <version> Reinstall global 'npm' packages contained in <version> to current version
nvm unload               Unload 'nvm' from shell
nvm which [<current | <version>] Display path to installed node version. Uses .nvmrc if available
nvm cache dir             Display path to the cache directory for nvm
nvm cache clear           Empty cache directory for nvm

Example:
nvm install 8.0.0         Install a specific version number
nvm use 8.0               Use the latest available 8.0.x release
nvm run 6.10.3 app.js     Run app.js using node 6.10.3
nvm exec 4.8.3 node app.js Run 'node app.js' with the PATH pointing to node 4.8.3
nvm alias default 8.1.0   Set default node version on a shell
nvm alias default node     Always default to the latest available node version on a shell

Note:
to remove, delete, or uninstall nvm - just remove the '.nvm' folder (usually '~/.nvm')

christopher@christopher-VirtualBox:~$ nvm install node
Downloading and installing node v10.2.1:..
Downloading https://nodejs.org/dist/v10.2.1/node-v10.2.1-linux-x64.tar.xz...
##### 100%
Computing checksum with sha256sum
Checksums matched!
Now using node v10.2.1 (npm v5.6.0)
Creating default alias: default -> node (.. v10.2.1)
christopher@christopher-VirtualBox:~$ node -v
v10.2.1
christopher@christopher-VirtualBox:~$
```


Installing the Heroku CLI

To deploy application to Heroku, you need to install the Heroku CLI. To use Heroku, you will need to create an account (heroku.com)

For information on the CLI, you can go to the following link:

<https://devcenter.heroku.com/articles/heroku-cli>

1. `curl https://cli-assets.heroku.com/install-ubuntu.sh | sh`
2. To check that you can connect to Heroku, type the following:
 - a. `heroku login`
 - b. `heroku logout`

An alternative to Heroku could be to setup your own Server and although it requires additional effort it can be helpful and cost less when working with larger projects. Two useful links to setting up such an environment are:

- <https://gorails.com/deploy/ubuntu/16.04>
- <https://gist.github.com/jrochkind/2161449>

Install Atom

There are a number of different text editors that can be used for Rails development but I have been using Atom for some time and find that it has a number of plugins that can be helpful. I would recommend testing different editors as they all have their advantages, maybe you find one you prefer. <https://www.ruby-lang.org/en> recommends the following IDE's :

- Linux and cross-platform tools:
 - Aptana Studio
 - Emacs with Ruby mode and Rsense
 - Geany
 - gedit
 - Vim with vim-ruby plugin and Rsense
 - RubyMine
 - SciTe
 - NetBeans
 - Sublime Text
 - Atom
- On Windows:
 - Notepad++
 - E-TextEditor
 - Ruby In Steel
 - Atom
- On Mac OS X:
 - TextMate
 - TextWrangler
 - Dash (documentation browser)
 - Atom

To install Atom on Ubuntu run the following commands from the terminal:

1. `curl -sL https://packagecloud.io/AtomEditor/atom/gpgkey | sudo apt-key add -`
2. `sudo sh -c 'echo "deb [arch=amd64] https://packagecloud.io/AtomEditor/atom/any/ any main" > /etc/apt/sources.list.d/atom.list'`
3. `sudo apt-get update`
4. `sudo apt-get install atom`
5. Add atom to the Task Bar as described previously.

Installing Ruby and Ruby on Rails

There are different ways of installing ruby, I will be using RVM to install and manage my ruby versions, to get more information on the differences, check out the documentation on the Ruby Website:

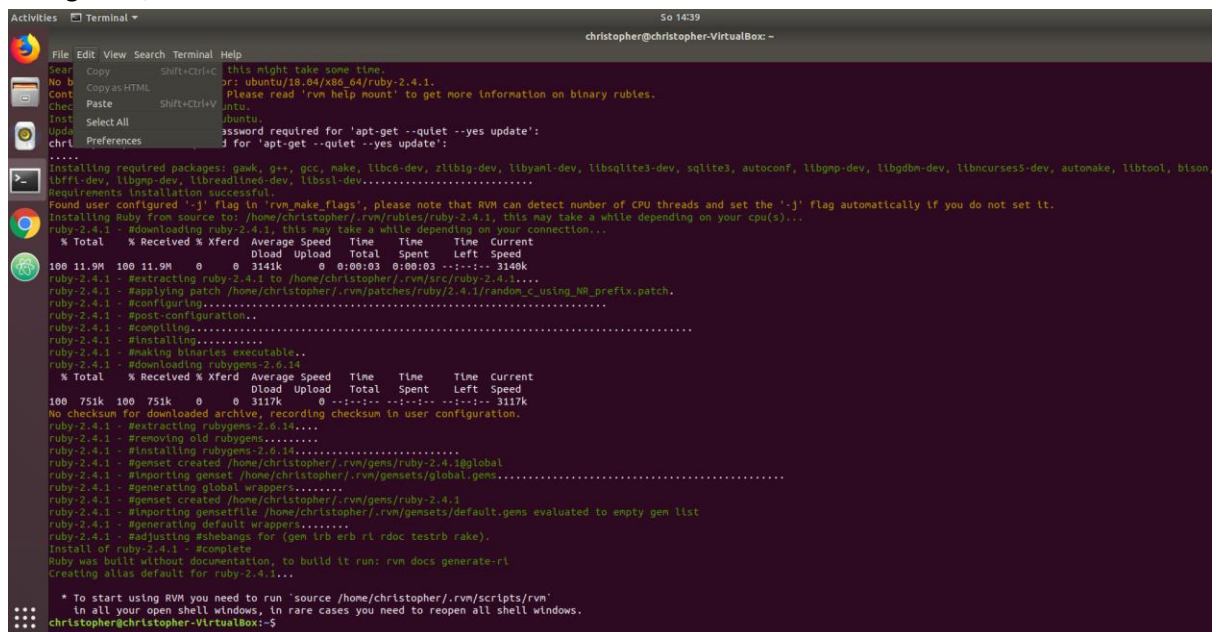
<https://www.ruby-lang.org/en/documentation/installation/>

Installing RVM on Ubuntu

To get more details on the installation go to www.rvm.io/rvm/install

Run the following commands from your terminal:

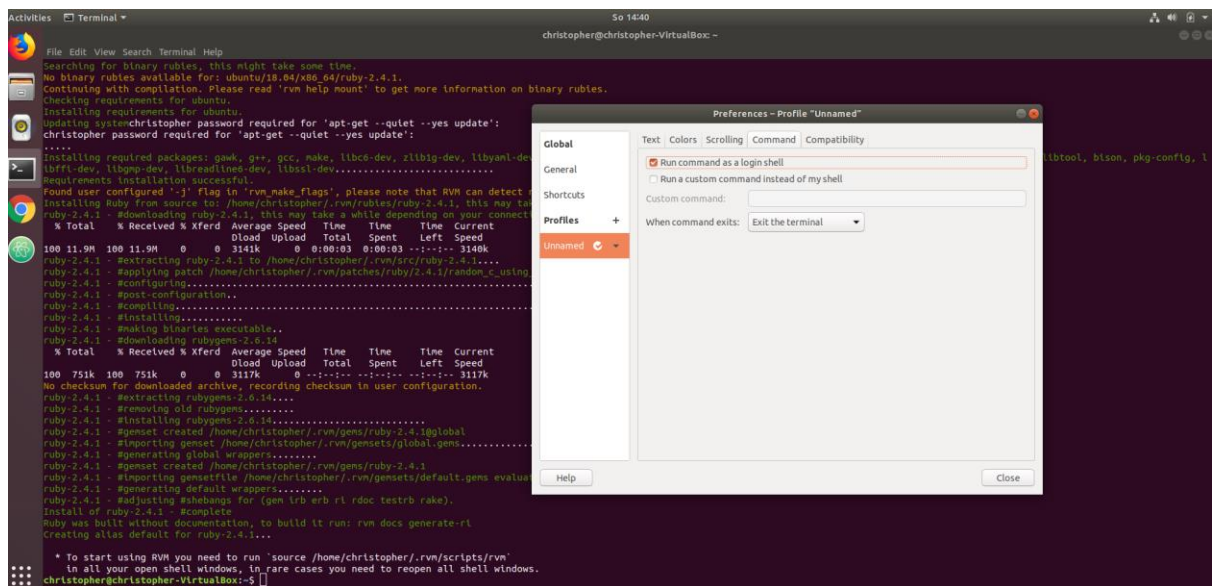
1. `gpg --keyserver hkp://keys.gnupg.net --recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3 7D2BAF1CF37B13E2069D6956105BD0E739499BDB`
2. `\curl -sSL https://get.rvm.io | bash -s stable --ruby`
3. Close the terminal and reopen it
4. The default terminal does not give you permissions to run all the programs by default. To change this, click on Edit -> Preferences from the Terminal Menu Bar



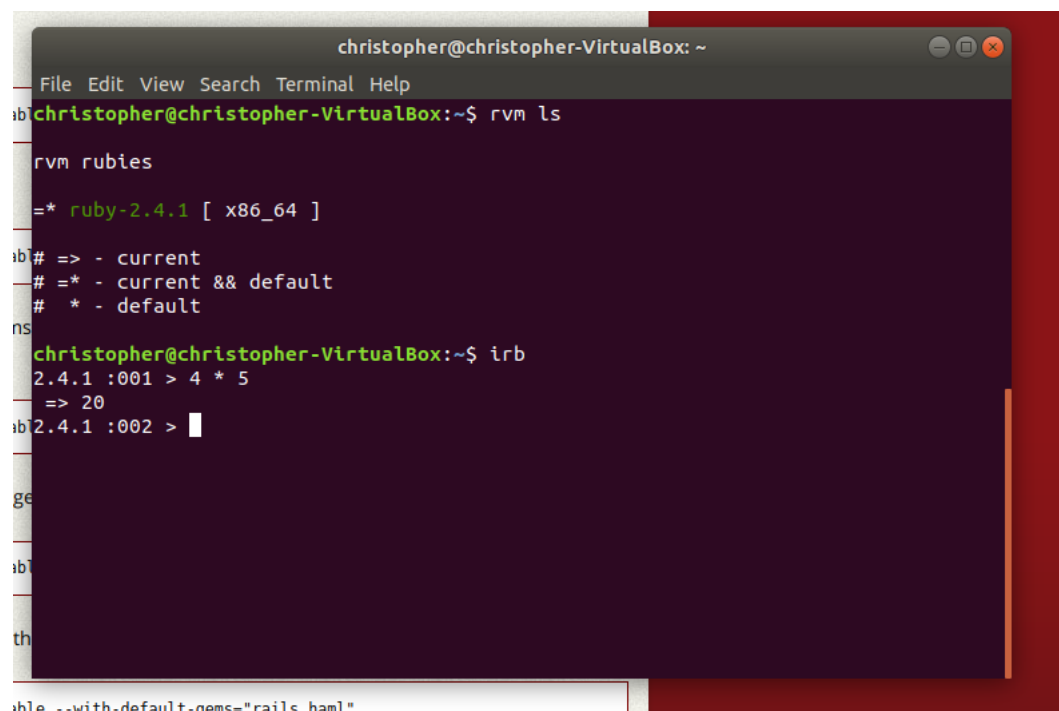
```
Activities Terminal 50 14:39 christopher@christopher-VirtualBox: ~
File Edit View Search Terminal Help
Search Copy Shift+Ctrl+C this might take some time
No B Copy as HTML Ctrl+Shift+C Please read 'rvm help mount' to get more information on binary rubies.
Check Paste Shift+Ctrl+V untu.
Install Select All Shift+Ctrl+A ubuntu.
Update Update Ctrl+U ssword required for 'apt-get --quiet --yes update':
Ctrl Preferences f for 'apt-get --quiet --yes update':
.....
Installing required packages: gawk, g++, gcc, make, libcc-dev, zlib1g-dev, libyaml-dev, libsqlite3-dev, sqlite3, autoconf, libgmp-dev, libgdbm-dev, libncurses-dev, automake, libtool, bison
libffi-dev, libgnp-dev, libreadline-dev, libssl-dev.....
Requirements installation successful.
Found user configured '-j' flag in 'rvm_make_flags', please note that RVM can detect number of CPU threads and set the '-j' flag automatically if you do not set it.
Installing Ruby from source to: /home/christopher/.rvm/rubies/ruby-2.4.1, this may take a while depending on your cpu(s)...
ruby-2.4.1 - #downloading ruby-2.4.1, this may take a while depending on your connection...
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 11.9M 100 11.9M 0 0 3141k 0 0:00:03 0:00:03 --:--:-- 3140k
ruby-2.4.1 - #extracting ruby-2.4.1 to /home/christopher/.rvm/src/ruby-2.4.1...
ruby-2.4.1 - #applying patch /home/christopher/.rvm/patches/ruby/2.4.1/random_c_using_NO_prefix.patch.
ruby-2.4.1 - #configuring.....
ruby-2.4.1 - #post-configuration..
ruby-2.4.1 - #compiling.....
ruby-2.4.1 - #installing.....
ruby-2.4.1 - #making binaries executable..
ruby-2.4.1 - #downloading rubygems-2.6.14
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 751k 100 751k 0 0 3117k 0 0:01:10 0:01:10 --:--:-- 3117k
No checksum for downloaded archive, recording checksum in user configuration.
ruby-2.4.1 - #extracting rubygems-2.6.14...
ruby-2.4.1 - #removing old rubygems.....
ruby-2.4.1 - #installing rubygems-2.6.14.....
ruby-2.4.1 - #gemset created /home/christopher/.rvm/gems/ruby-2.4.1@global
ruby-2.4.1 - #importing gemset /home/christopher/.rvm/gemsets/global.gems.....
ruby-2.4.1 - #generating global wrappers.....
ruby-2.4.1 - #gemset created /home/christopher/.rvm/gems/ruby-2.4.1
ruby-2.4.1 - #importing gemsetfile /home/christopher/.rvm/gemsets/default.gems evaluated to empty gem list
ruby-2.4.1 - #generating default wrappers.....
ruby-2.4.1 - #adjusting #shebangs for (gem irb erb ri rdoc testrb rake).
Install of ruby-2.4.1 - #complete
ruby was built without documentation, to build it run: rvm docs generate-ri
Creating alias default for ruby-2.4.1...

* To start using RVM you need to run 'source /home/christopher/.rvm/scripts/rvm'
in all your open shell windows, in rare cases you need to reopen all shell windows.
christopher@christopher-VirtualBox:~$
```

5. Click on the Command tab and tick the Run command as login shell, tick box



6. Close and reopen the terminal
7. Type `rvm list` to list the versions of ruby that are installed
8. Type `irb` enter `irb`



Install Gems

Ruby Gems are ruby programs and libraries that packaged in a way that makes it easy to install and distribute. <https://rubygems.org> provides a platform where you can host and download Gems. In order to setup our system we will need to install a number of gems.

1. `gem install bundler`
 - a. bundler allows us to install gems eg: `bundler install devise`

- b. You can read up on bundler from the following link:

<https://rubygems.org/gems/bundler>

2. gem install rails

- a. To find out more information on rails :

<https://rubygems.org/gems/rails>

- b. Type: rails -v to ensure it was installed correctly

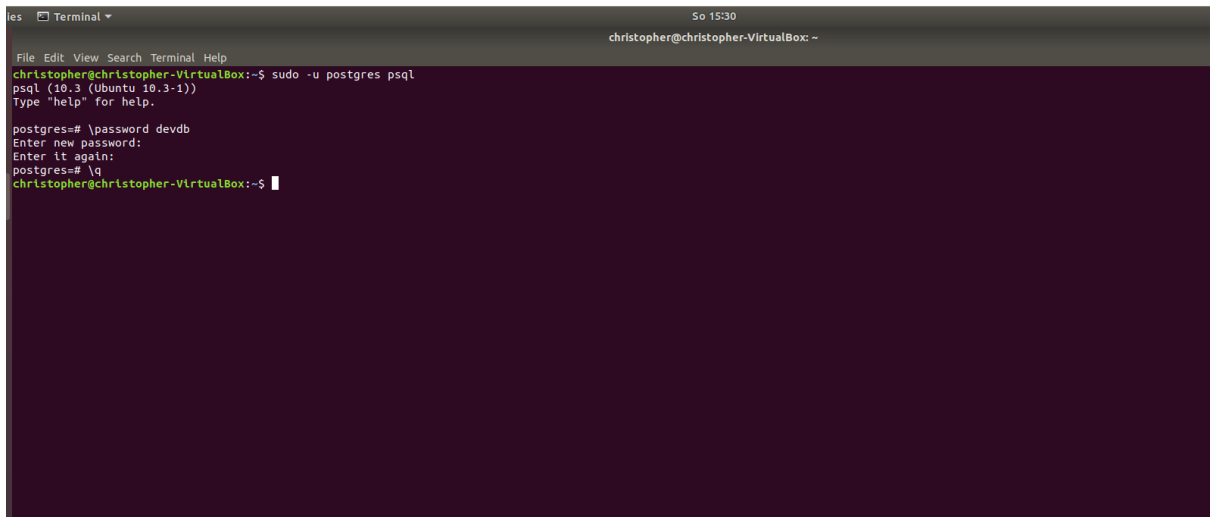
```
File Edit View Search Terminal Help
christopher@christopher-VirtualBox: ~
Installing ri documentation for arel-9.0.0
Parsing documentation for activerecord-5.2.0
Installing ri documentation for activerecord-5.2.0
Parsing documentation for globalid-0.4.1
Installing ri documentation for globalid-0.4.1
Parsing documentation for activejob-5.2.0
Installing ri documentation for activejob-5.2.0
Parsing documentation for mini_mime-1.0.0
Installing ri documentation for mini_mime-1.0.0
Parsing documentation for mail-2.7.0
Installing ri documentation for mail-2.7.0
Parsing documentation for actionmailer-5.2.0
Installing ri documentation for actionmailer-5.2.0
Parsing documentation for nokl-2.3.1
Installing ri documentation for nokl-2.3.1
Parsing documentation for websocket-extensions-0.1.3
Installing ri documentation for websocket-extensions-0.1.3
Parsing documentation for websocket-driver-0.7.0
Installing ri documentation for websocket-driver-0.7.0
Parsing documentation for actioncable-5.2.0
Installing ri documentation for actioncable-5.2.0
Parsing documentation for minenagc-0.3.2
Installing ri documentation for minenagc-0.3.2
Parsing documentation for marcel-0.3.2
Installing ri documentation for marcel-0.3.2
Parsing documentation for activestorage-5.2.0
Installing ri documentation for activestorage-5.2.0
Parsing documentation for thor-0.20.0
Installing ri documentation for thor-0.20.0
Parsing documentation for method_source-0.9.0
Installing ri documentation for method_source-0.9.0
Parsing documentation for railties-5.2.0
Installing ri documentation for railties-5.2.0
Parsing documentation for sprockets-3.7.1
Installing ri documentation for sprockets-3.7.1
Parsing documentation for sprockets-rails-3.2.1
Installing ri documentation for sprockets-rails-3.2.1
Parsing documentation for rails-5.2.0
Installing ri documentation for rails-5.2.0
Done installing documentation for concurrent-ruby, i18n, thread_safe, tzinfo, activesupport, rack, rack-test, mini_portile2, nokogiri, crass, loofah, rails-html-sanitizer, rails-dom-testing, builder, erub
i, actionview, actionpack, activemodel, arel, activerecord, globalid, activejob, mini_mime, mail, actionmailer, nokl, websocket-extensions, websocket-driver, actioncable, minenagc, marcel, activestorage
, thor, method_source, railties, sprockets, sprockets-rails, rails after 62 seconds
38 gems installed
christopher@christopher-VirtualBox:~$ rails -v
rails 5.2.0
christopher@christopher-VirtualBox:~$
```

Install PostgreSQL

Ruby on Rails comes with sqlite3 however, if you plan on developing productive applications it is best practice to use the same Database on your dev and productive environments. As we will be deploying our applications to Heroku I will be installing PostgreSQL but if are deploying to your own server or a server with MySQL installed you should choose the database that is the same as your productive environment.

To install PostgreSQL run the following commands from your terminal:

1. `sudo apt-install postgresql postgresql-contrib libpq-dev`
2. create the postgres user account
 - a. `sudo -u postgres createuser -s devdb`
3. To access the postgres database type:
 - a. `sudo -u postgres psql`
4. to set a password for the devdb user we need to type the below command from the postgres terminal:
 - a. `\password devdb`
 - b. When you press enter it will prompt you for a password
 - c. Type `\q` to exit the prompt

A terminal window titled 'Terminal' with a dark background. The prompt is 'christopher@christopher-VirtualBox: ~'. The user enters 'sudo -u postgres psql', which opens a postgres prompt. The user enters '\password devdb', followed by 'Enter new password:', 'Enter it again:', and '\q'. The prompt returns to 'christopher@christopher-VirtualBox: ~'.

```
les  Terminal  So 15:30 christopher@christopher-VirtualBox: ~
christopher@christopher-VirtualBox:~$ sudo -u postgres psql
psql (10.3 (Ubuntu 10.3-1))
Type "help" for help.

postgres=# \password devdb
Enter new password:
Enter it again:
postgres=# \q
christopher@christopher-VirtualBox:~$
```