DevOps Exercise Book

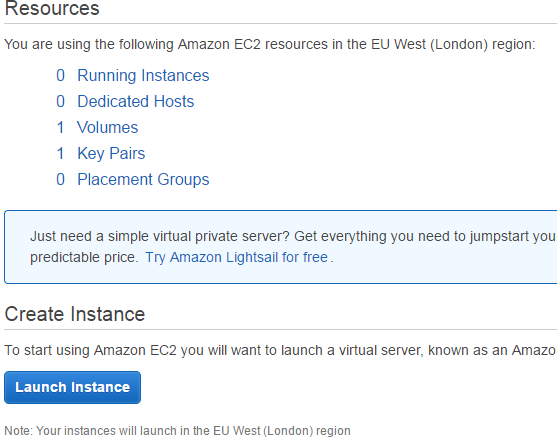
# Task1: Your first Virtual Machine

Using AWS EC2 Ubuntu box

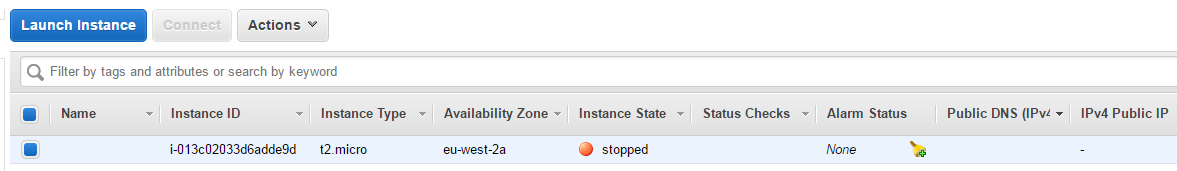


Create instance of Ubuntu box according to needs.

Launch instance from Running Instances

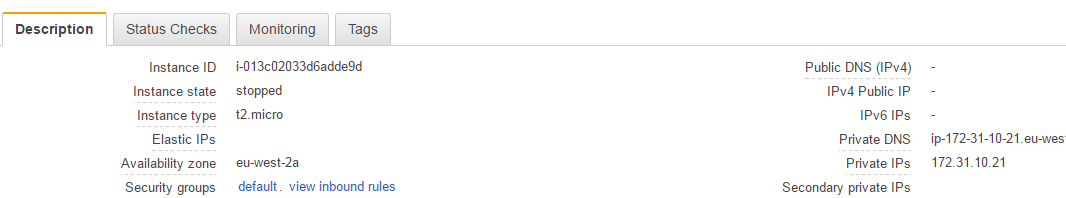


Following instance is present.

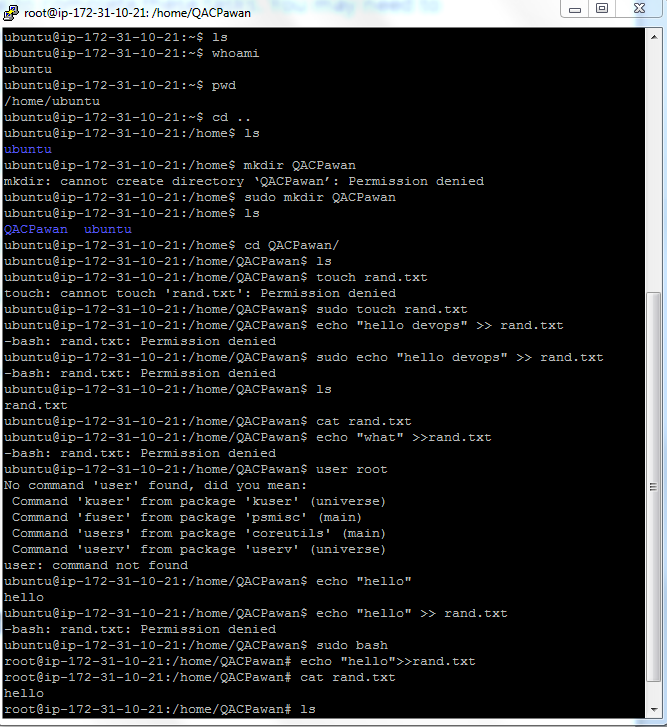


SSH To

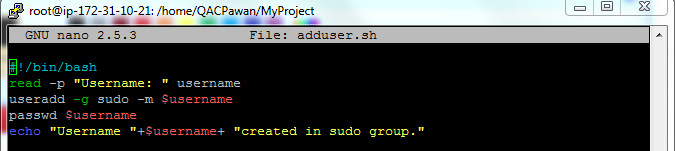
username@publicdns; ubuntu@...

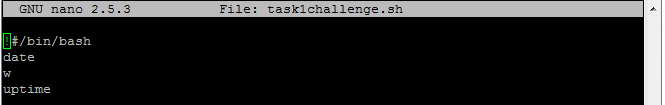


# Task 2: Terminal Exploration



## Task 3: Creating a script file



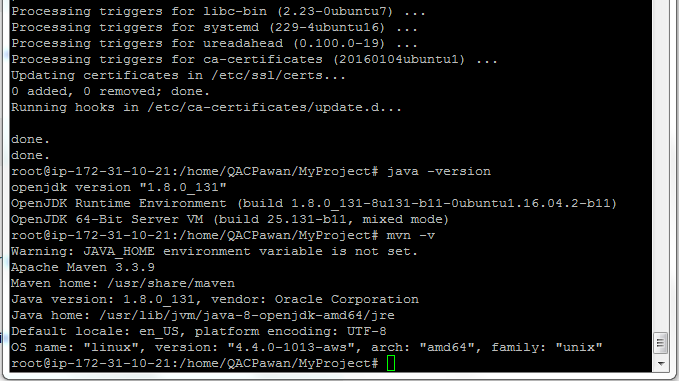


# Task 4: Configuring the Linux environment

**apt-get update**

**apt-get install maven**

(Maven installed itself and java as well)



# Task 5: Using Iptables

Start iptables:

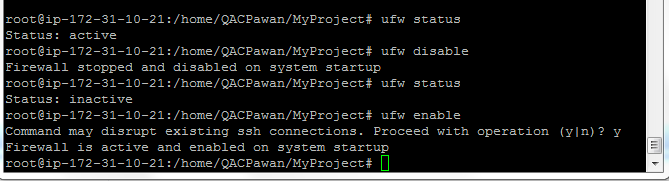
**sudo ufw enable**

Stop iptables:

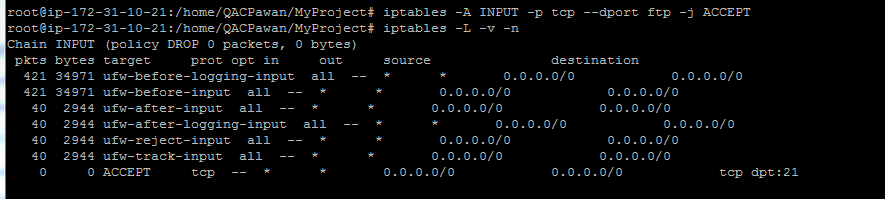
**sudo ufw disable**

Show status:

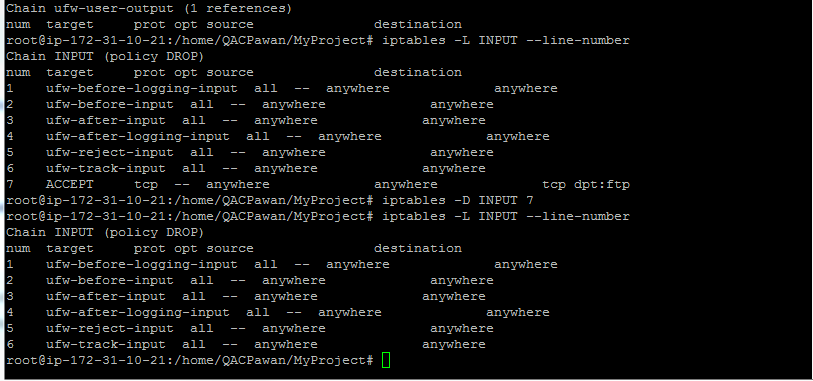
**sudo ufw status**



Allowing ftp at port 21 rule:



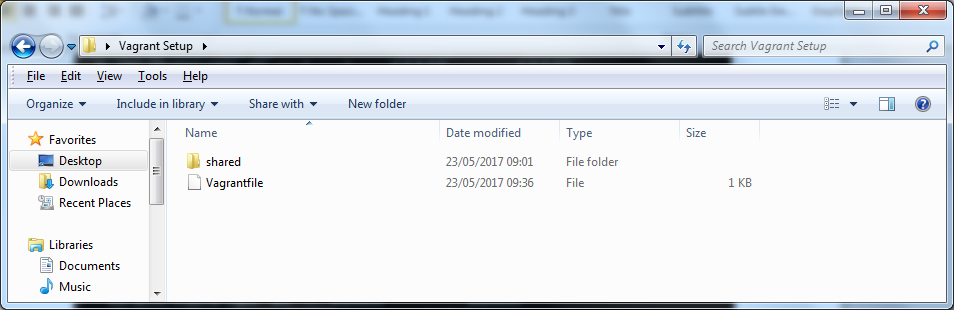
View rule by input chain and by line number, delete by input chain+linenumber



# Task 6: Vagrant scripting

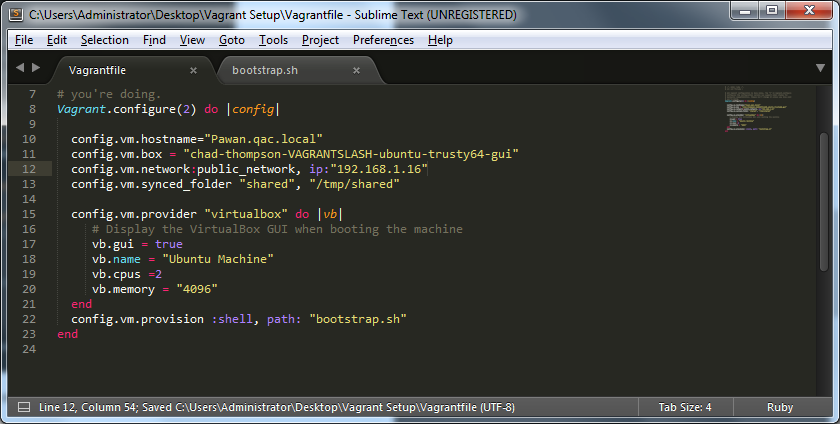
Create directory “Vagrant Setup” - in here open Git bash and use the command, *vagrant init*, to initialise a new vagrant repository. This should add a Vagrantfile to the directory

Create a shared folder to use with VM.

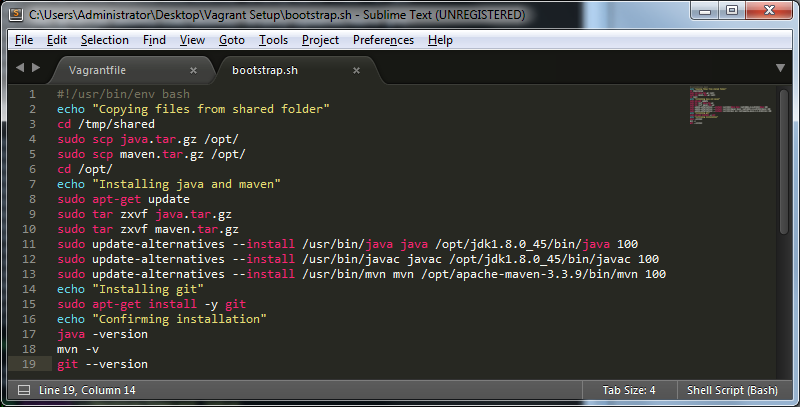


## For single VM installation

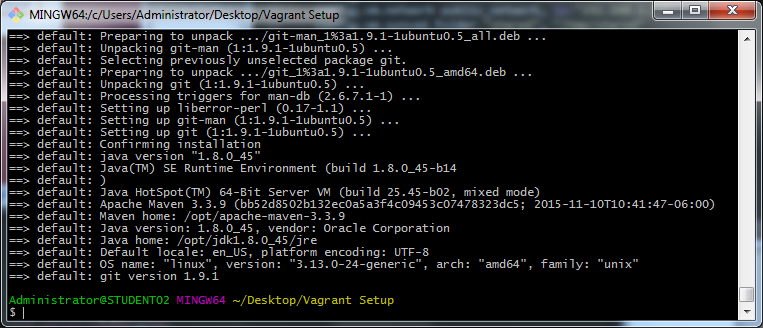
Edited vagrantfile to setup the VM with required specification.



Bash script used to install JAVA, Maven and Git on VM.



Use “vagrant up” on the folder with vagrantfile to create VM and run bash script.



## For multiple VM installation

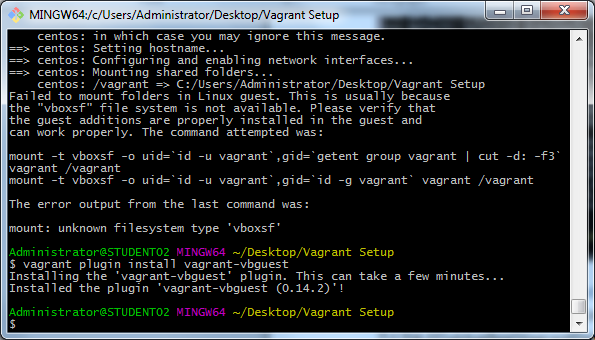
For CentOS installation, edit C:\Users\Administrator\.vagrant.d\boxes\centos-VAGRANTSLASH-7\1704.01\virtualbox\Vagrantfile

**config.vm.synced\_folder ".", "/vagrant", type: "rsync"**

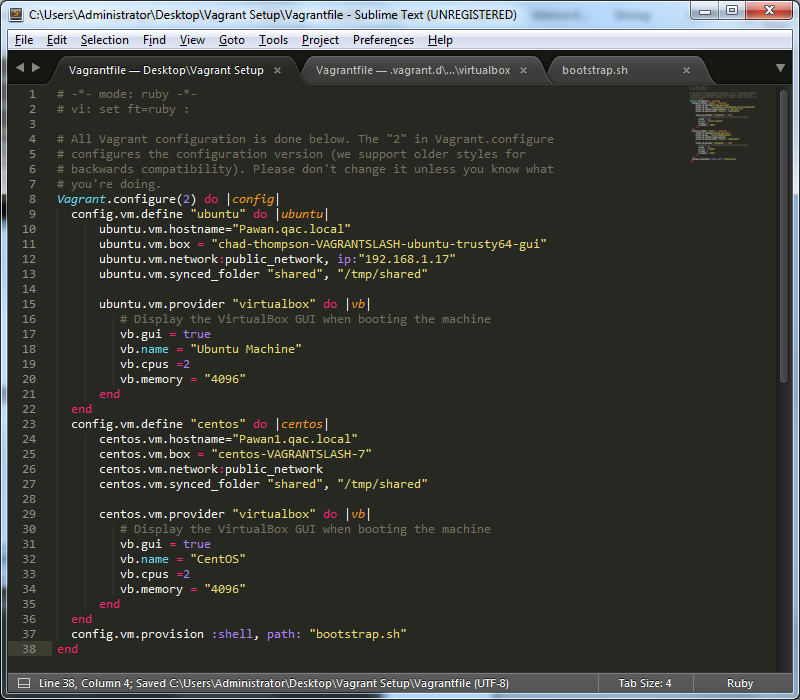
**config.vm.synced\_folder ".", "/vagrant", type: "virtualbox"**

Rsync isn’t installed.

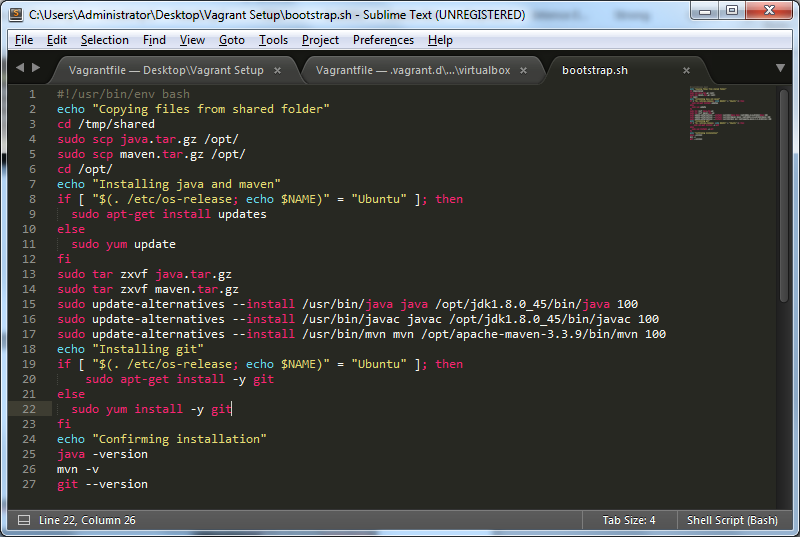
Run **vagrant plugin install vagrant-vbguest** on git bash as the shared folder encounters problem with centOS due to mismatch of GuestAdditions between centOS and virtualbox.



Vagrantfile for multiple VM

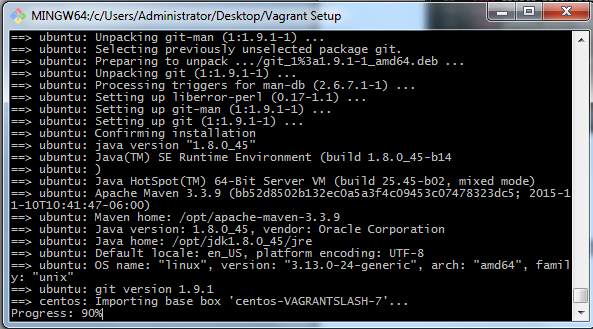


Script file for multiple VM

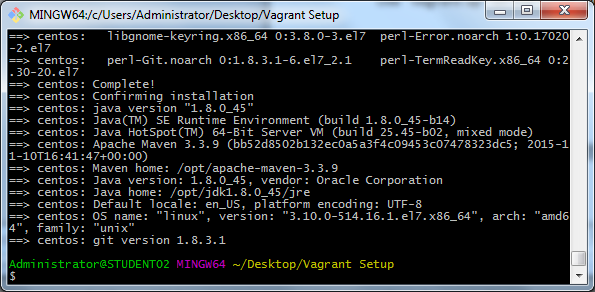


Use “vagrant up” on the folder with vagrantfile to create VM and run bash script.

Ubuntu installed with JAVA, MAVEN, and GIT.

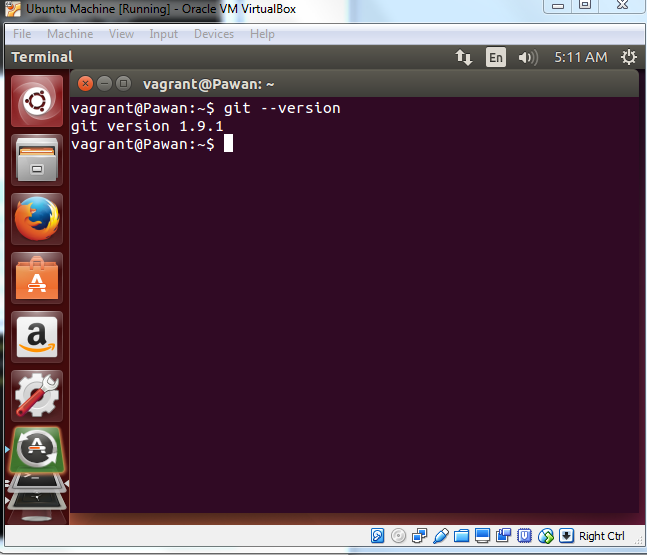


CentOS installed with JAVA, MAVEN, and GIT.

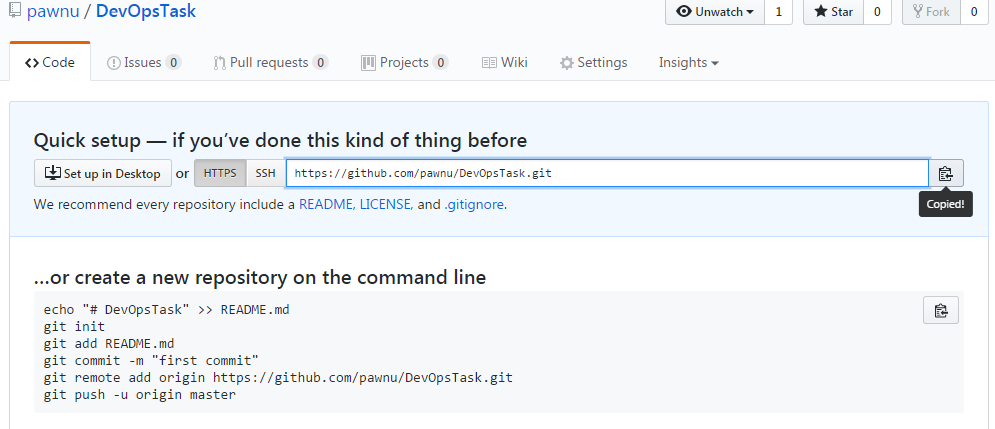


# Task 7: Repository Management

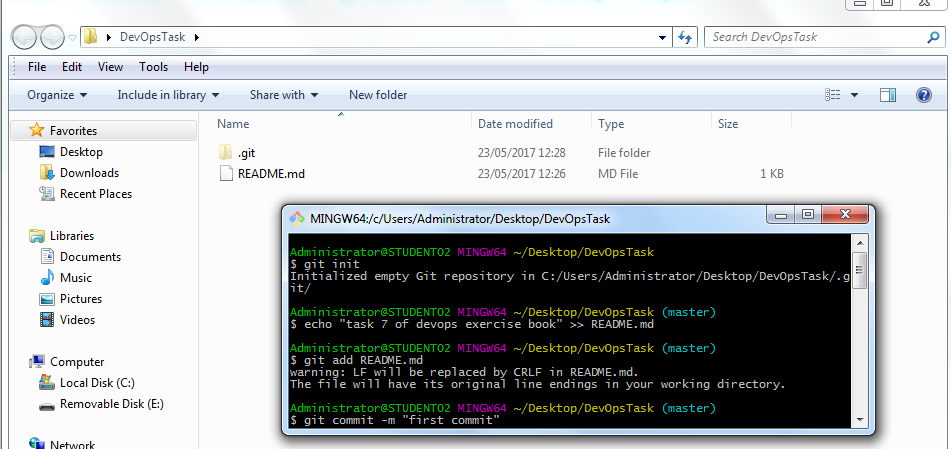
Confirmed Git is installed in VM



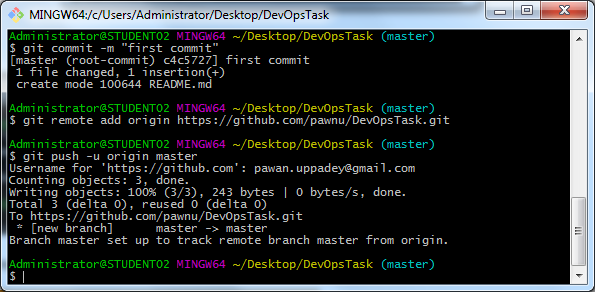
On Windows host machine, create a repository on github.



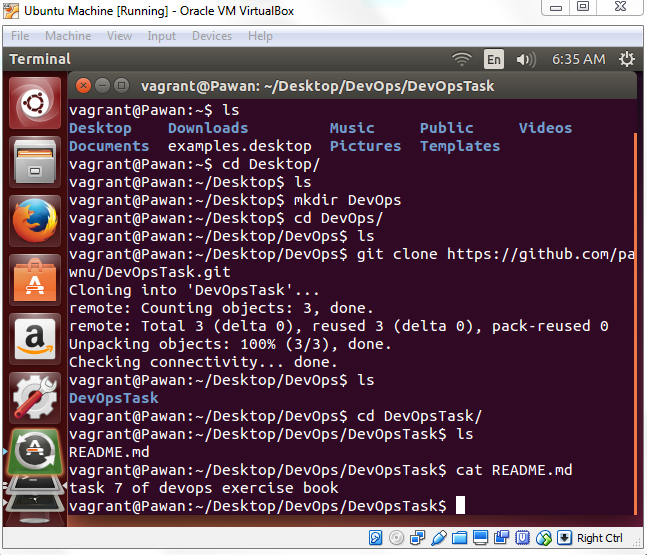
On host windows machine, create a directory for git and use “git init” command on that folder. Create a file, add and commit it.



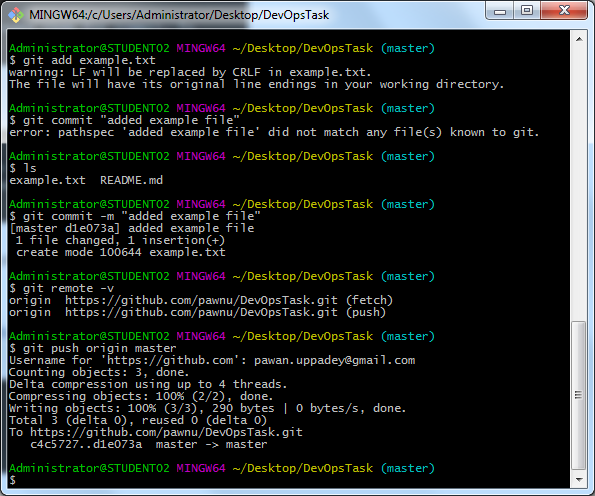
Add the commit to your repository on github with HTTPS link and push the changes.



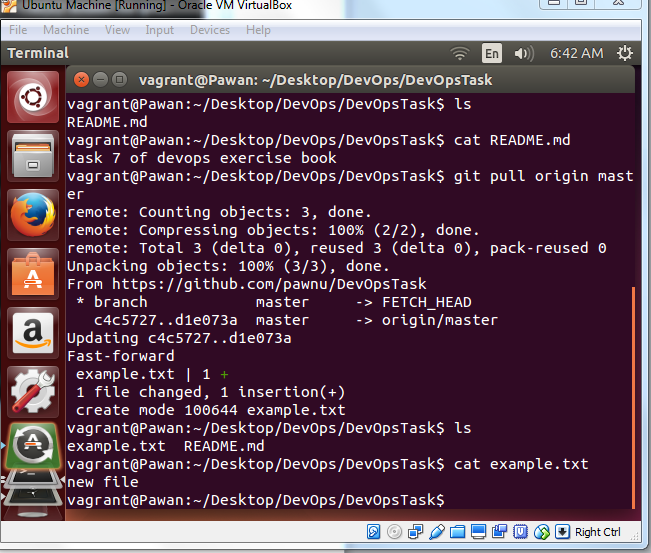
On Ubuntu guest machine, clone and confirm the file created earlier is present.



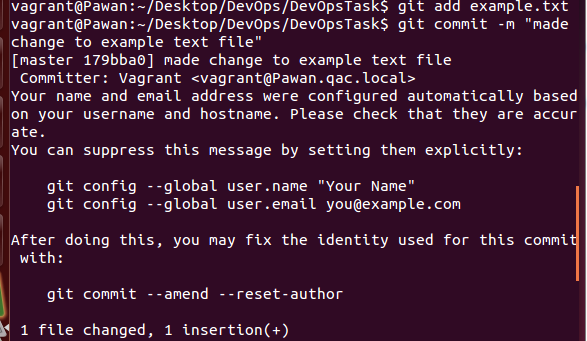
Create a new file on Windows host machine and push change to repository.

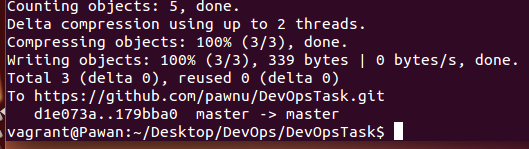


Confirm the file present on Ubuntu guest machine.

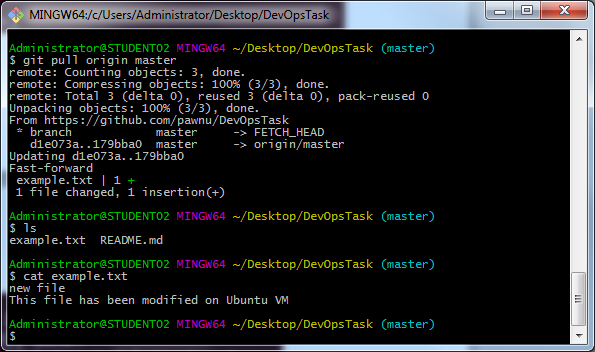


Make change to a file on Ubuntu and push to repository





Confirm changes on Windows host machine side



# Task 8: Tooling

## Task 1a – Setting up Jenkins

sudo scp jenkins\_2.1\_all.deb /home/vagrant/Desktop/

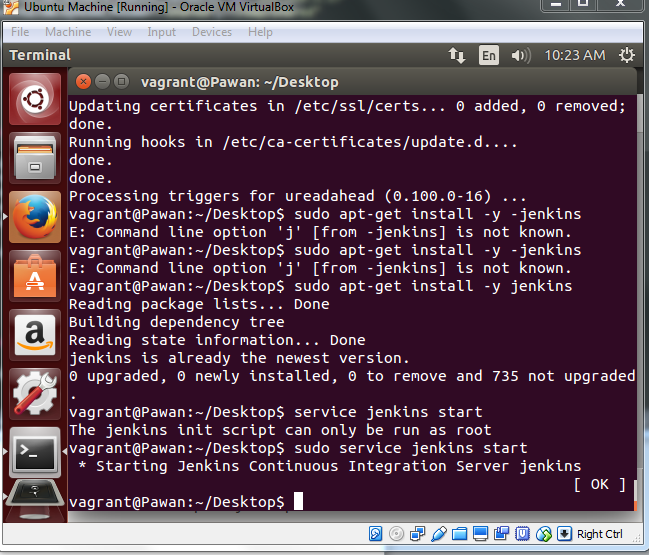
cd /home/vagrant/Desktop

sudo dpkg –i jenkins\_2.1\_all.deb

sudo apt-get install –y –f

sudo apt-get install –y jenkins

sudo service jenkins start





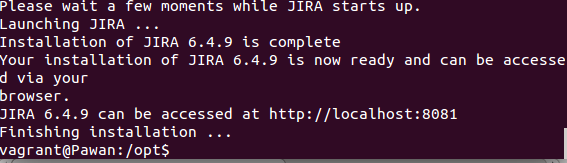
## Task 2 – Setting up Jira

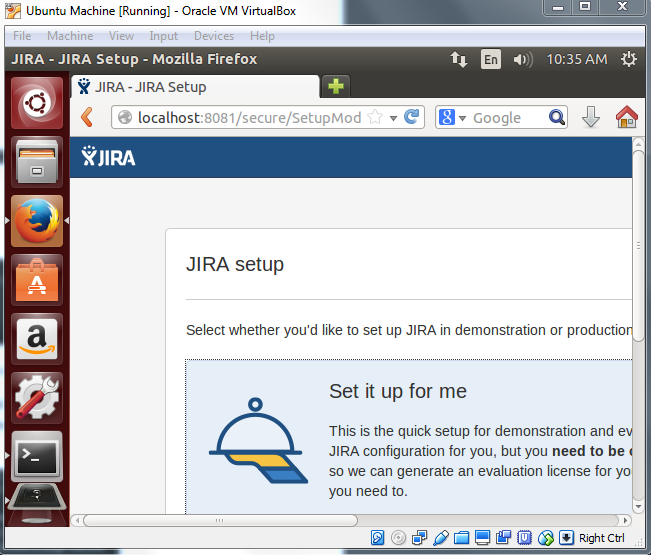
cd /opt/

sudo chmod a+x jira.bin

sudo ./jira.bin

Choose custom install, install on port 8081, custom port 80





# Task 4 – Installing Nexus

cd /opt/

sudo scp nexus-2.14.4-03-bundle.tar.gz /usr/local

cd /usr/local

sudo tar xvzf nexus-2.14.4-03-bundle.tar.gz

sudo ln –s nexus-2.14.4-03 nexus

echo "1" | sudo update-alternatives --config java

sudo chowm -R vagrant nexus\* sonatype-work

cd /usr/local/nexus

./bin/nexus console

./bin/nexus start

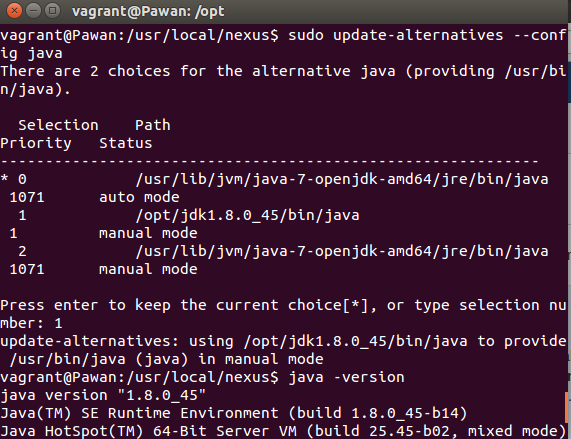
**Notes:**

nexus-3.\* didn’t work with this setup

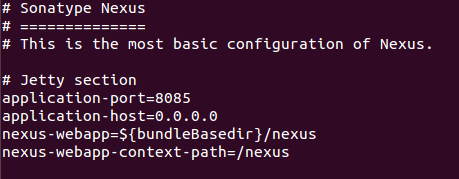
nexus won’t run with sudo, change a user to be owner of nexus and sonawork directory to run

nexus requires JVM 1.8. JVM might show as 1.7. Make sure java 1.8 is installed.

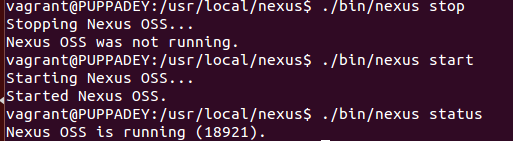
Use **sudo update-alternatives - -config java** and select the 1.8 version of java to run nexus

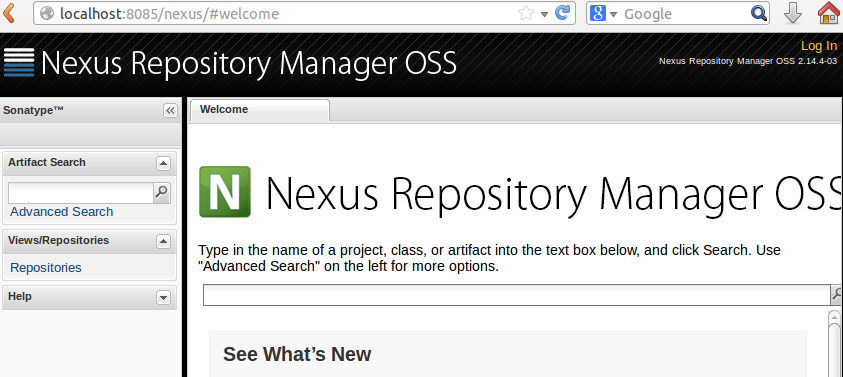


Nexus by default runs on port 8081, to change it edit the **conf/properties** file



Run nexus as follows:





# Task 5 – Install Zabbix

cd /opt/

wget http://repo.zabbix.com/zabbix/2.4/ubuntu/pool/main/z/zabbix-release/zabbix-release\_2.4-1+trusty\_all.deb

sudo dpkg -i zabbix-release\_2.4-1+trusty\_all.deb

sudo apt-get install -y zabbix-server-mysql zabbix-frontend-php php5-mysql

Edit **/etc/php5/apache2/php.ini**

post\_max\_size = 16M

max\_execution\_time = 300

max\_input\_time = 300

date.timezone = Europe/London

sudo service apache2 restart

Create **/etc/zabbix/apache.conf**

# Define /zabbix alias, this is the default  
<IfModule mod\_alias.c>  
Alias /zabbix /usr/share/zabbix  
</IfModule>

sudo cp /etc/zabbix/apache.conf /etc/apache2/conf-available/zabbix.conf

sudo a2enconf zabbix.conf

sudo service apache2 restart

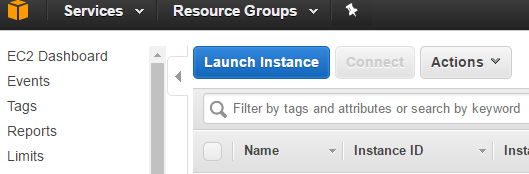
sudo service zabbix-server start

****

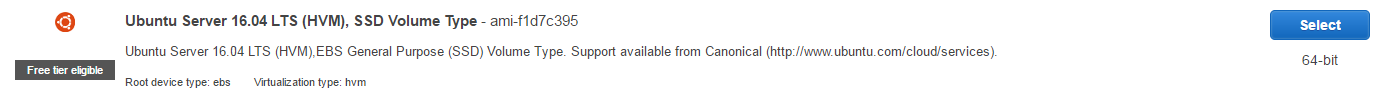
# Docker

## Creating AWS instance

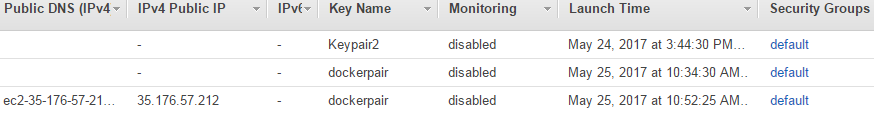
EC2 – Launch Instance

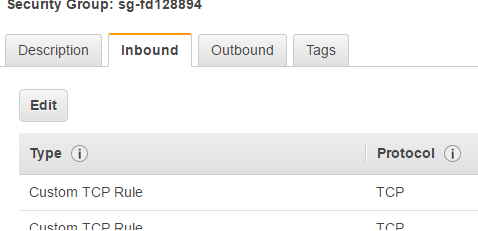


Install Ubuntu

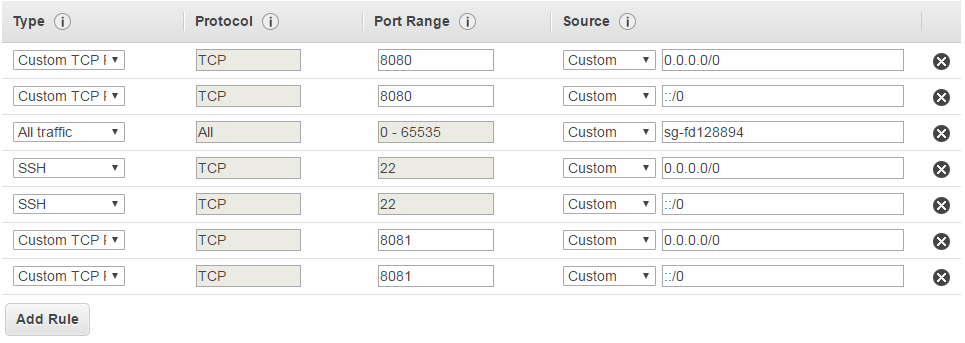


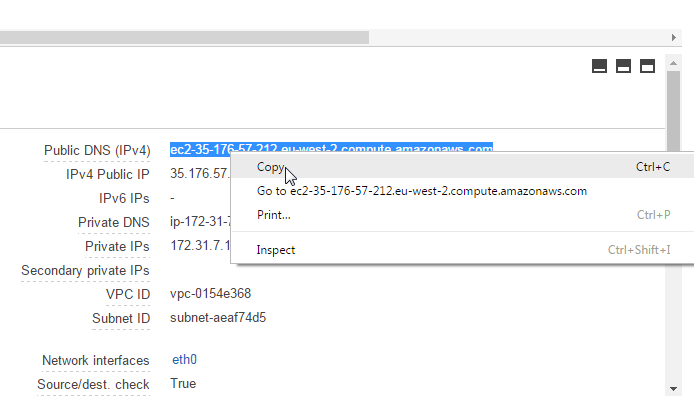
Go to security groups – default to configure firewall





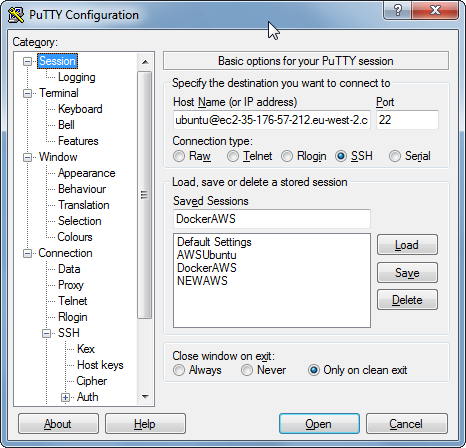
Add port 22 for SSH, and others as required e.g. 8080





The .pem file provided by AWS to access SSH can be broken down into a private/public key pair. The private key can be created by using the .pem file on PuTTYgen – click generate private key.

Put your privatekey to SSH-Auth on PuTTY Config. Insert the username (Ubuntu) public IP address and port 22 to access SSH.



## Task 1 – Install docker

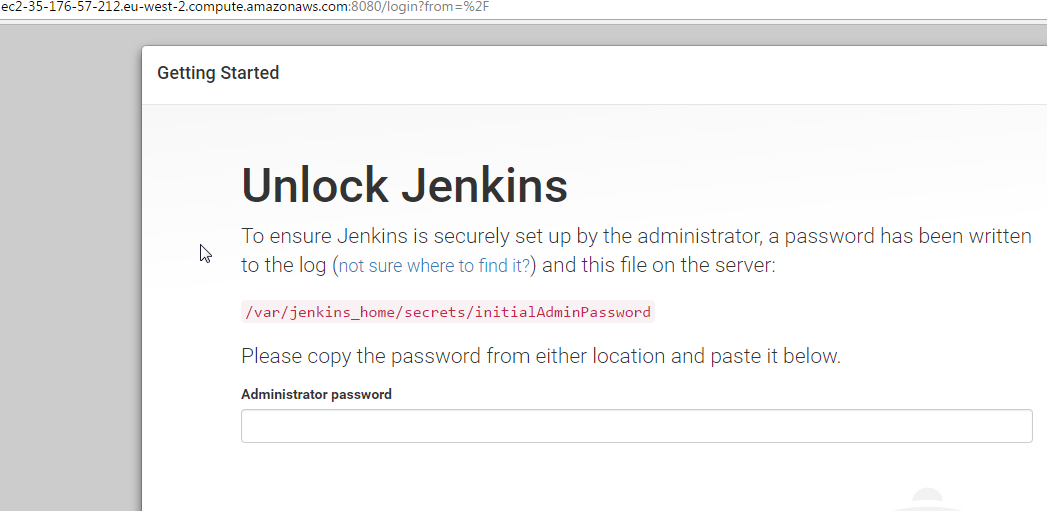
Get the script file to install docker and its dependencies from docker website and pipe it to the shell.

**wget -qO- https://get.docker.com/ | sh**

## Task 2 – Deploy a Jenkins Container

**docker pull jenkins**

**docker run -p 8080:8080 -p 50000:50000 jenkins**



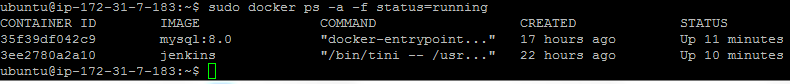
A container of Jenkins has been created, use ctrl+z to get out of the terminal.

**sudo docker ps –a**

**Lists all processes, find the container id of Jenkins**

**sudo docker start containerID**





## Task 3 – Create a dockerfile

FROM ubuntu:16.04

#Always update your running system

RUN sudo apt-get update -y

#You may or may not need to run these commands

RUN sudo apt-get install -y wget

RUN sudo apt-get install -y tar

#installs the libraries needed to run the GUI

RUN sudo apt-get install -y libgtk2.0

RUN sudo apt-get install -y mesa-utils

RUN sudo apt-get install -y libXtst6

#RUN sudo apt-get install -y openjdk-7-jre

#RUN java -version

#Now install the Java Compiler

#RUN sudo apt-get install -y openjdk-7-jdk

#RUN javac -version

#Add java from file and install

WORKDIR /opt

ADD files /opt

RUN sudo tar zxvf /opt/java.tar.gz

RUN sudo update-alternatives --install /usr/bin/java java /opt/jdk1.8.0\_74/bin/java 100

RUN sudo update-alternatives --install /usr/bin/javac javac /opt/jdk1.8.0\_74/bin/javac 100

# Install OpenJDK-8

RUN apt-get update && \

apt-get install -y openjdk-8-jdk && \

apt-get install -y ant && \

apt-get clean;

# Fix certificate issues

RUN apt-get update && \

apt-get install ca-certificates-java && \

apt-get clean && \

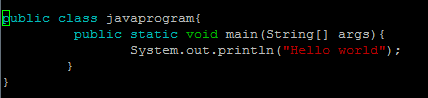
update-ca-certificates -f;

# Setup JAVA\_HOME -- useful for docker commandline

ENV JAVA\_HOME /usr/lib/jvm/java-8-openjdk-amd64/

RUN export JAVA\_HOME

Create java program



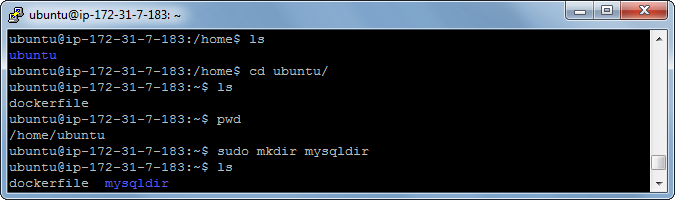
Compile and run it.





## Task 4 – Create your own linked container

To create mysql container that stores data in the host volume, create a directory



**docker run --name some-mysql -v /home/ubuntu/mysqldir:/var/lib/mysql -e MYSQL\_ROOT\_PASSWORD=my-secret-pw -d mysql:tag**

where **my-secret-pw** is password, **tag** is version number e.g. 8.0

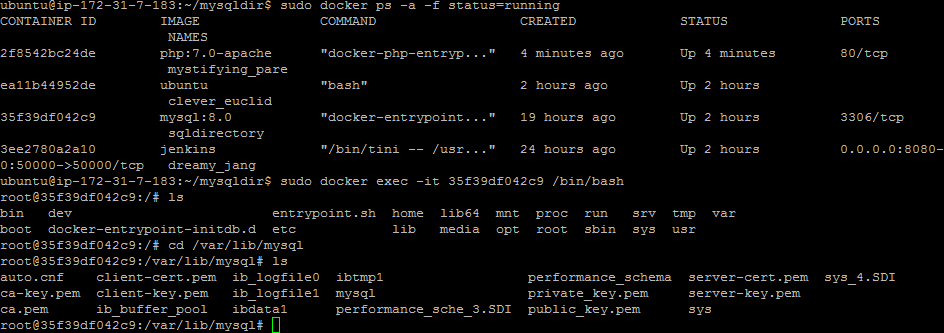
e.g. **docker run --name some-mysql -v /home/ubuntu/mysqldir:/var/lib/mysql -e MYSQL\_ROOT\_PASSWORD=abcde -d mysql:8.0**

Find running process with

**sudo docker ps -a -f status=running**

Execute interactive terminal –it inside a container using exec and containerID

**sudo docker exec -it containerID /bin/bash**

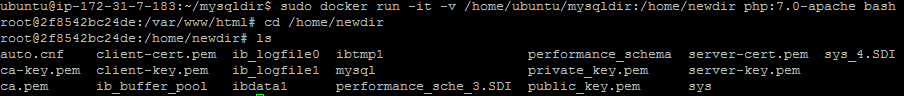
****

For PHP project, pull php image:

**sudo docker pull php**

When running the container mount the directory to a newdirectory in php file path

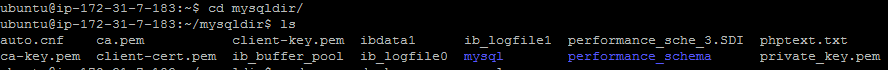
**sudo docker run -it -v /home/ubuntu/mysqldir:/home/newdir php:7.0-apache bash**

****

## Task 5 - Create your own docker-compose file

Create a directory to be used as shared folder.

/home/Ubuntu/mysqldir contains many files from task4

****

Create **docker-compose.yml** file as below:

**version : '2'**

**services:**

**db:**

**image: mysql:8.0**

**ports:**

**- "3333:3333"**

**volumes:**

**- /home/ubuntu/mysqldir:/var/www/html**

**environment:**

**MYSQL\_ROOT\_PASSWORD: passwerd**

**php:**

**image: php:7.0-apache**

**links:**

**- db:db**

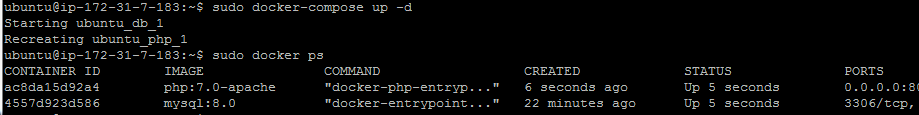
**ports:**

**- "80:80"**

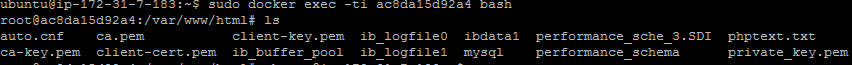
**volumes:**

**- /home/ubuntu/mysqldir:/var/www/html**

PHP an mysql are running



Execute the first container and check shared directory which contains files as expected.



Execute the second container and it can be seen that the directory is successfully mounted as the files expected are present.

