**Docker**

**=========================== Docker Installation on ubuntu ======================**

Create an AWS ubuntu18 instance

2 Open get.docker.com

3 Copy the below 2 commands and paste in Ubuntu terminal

curl -fsSL https://get.docker.com -o get-docker.sh

sh get-docker.sh

**=========================== Images & Containers =============================**

Windows-10 iso (Image)---by using this iso file we can install os to multiple laptops (containers)

**=========================== Multi-container-architecture ======================**

1. using --link option

2. using docker compose

3. docker-networking

4. python script

**=============================== Volumes ==================================**

1. simple docker volume (/var/lib/d6+ocker/volumes/75857b5bc1b0e578ce482b6e501af89b48bf5f16a706d6c290aa3455c0599268/\_data)

2. sharable docker volume

3. docker volume container

**========================= Docker-custmozed-images =========================**

we can create custmozed (including java, git-hub, ms-office) windows-10 iso. It is called as private

1. using docker commit command

2. using docker file

**========================= Docker-NETWORKING ==============================**

1.Bridge

2.Host only

3.Null

4.Overlay

**===================================Docker-registry===========================**

1.Public repository (hub.docker.com)

2.Private repository (created with in our organization)

**==================================Docker-swarm=============================**

1.Load Balancing

2.Scaling

3.Rolling-updates

4.Disaster recovery (Access & Handling failovers)

**============================== Docker-swarm-Networking =====================**

**=================================Docker-stack===============================**

Docker-compse+ swarm= Docker-stack

**Docker-commands**

**========================= Working on Images ================================**

1 To download a docker image

docker pull image\_name

2 To upload a docker image into docker hub

docker push image\_name

3 To see the list of all the docker images in docker hub

docker images / docker image ls

4 To delete a image

docker rmi image\_name/image\_id

5 To delete all the unused images,networks,volumes etc

docker system prune -a

6 To search for a docker image

docker search image\_name

7 To get detailed info about the docker image

docker inspect image\_name/image\_id

8 To create a docker image from a container

docker commit container\_name/container\_id image\_name

9 To create a image from a dockerfile

docker build -t image\_name .

**=========================== Working on Containers ===========================**

10 To see the list of running containers

docker container ls

11 To see the list of all the containers running and stopped

docker ps -a

12 To start a stopped container

docker start container\_name/container\_id

13 To stop a running container

docker stop container\_name/container\_id

14 To restart a container

docker restart container\_name/container\_id

To restart after 10 seconds

docker restart -t 10 container\_name/container\_id

15 To delete a stopped container

docker rm container\_name/container\_id

16 To delete a running container

docker rm -f container\_name/container\_id

17 To stop all running containers

docker stop $(docker ps -aq)

18 To delete all stopped containers

docker rm $(docker ps -aq)

19 To delete a containers running and stopped

docker rm -f $(docker ps -aq)

20 To get detailed info about any container

docker inspect container\_name/container\_id

21 To see the ports used by a container

docker port container\_name/container\_id

22 To see the logs genrated by a container

docker logs container\_name/container\_id

23 To execute an application in a container from outside the container

docker exec -it container\_name/container\_id command\_to\_be\_executed

Eg: To open bash shell in an already running container

docker exec -it container\_name/container\_id bash

24 To come out of a container without exit

ctrl+p,ctrl+q

25 To go into a container from which we have come out without exit

docker attach container\_name/container\_id

26 To create a container from an image

docker run image\_name/image\_id

Run command options

--name This is used to give a name for the container

-d Used to run the container in detached mode in the background

-it Used to open an interaxtive terminal in the container

-e Used for passing environment variables to the container

-v Used for attaching a volume to a container

--volumes-from Used for sharing volumes between multiple containers

-p Used for port mapping.This will link the container port(internal

port) with the host port(external port)

Eg: -p 8080:80 Here 8080 is the host port and 80 is the container port

-P Used for automatic port mapping ie the container port will be mapped

with a host port that is greater than 30000

-rm Used to delete a container on exit

--link Used to create a link between multiple containers so as to create

a multi container architecture

--network Used to start a container on a specific network

-m Used to specify the max memory that can be used by a container

-c Used to specify the cpus that should be alloted to a container

-h Used to specify a hostname to a container

**======================== Working on docker networks ===============================**

27 To see the list of all the docker networks

docker network ls

28 To create a new docker network

docker network create --driver driver\_type network\_name

28 To get detailed info about a network

docker network inspect network\_name/network\_id

29 To delete a network

docker network rm network\_name/network\_id

30 To attach a running container to a network

docker network connect network\_name/network\_id container\_name/container\_id

31 To remove a running container from a network

docker network disconnect network\_name/network\_id container\_name/container\_id

**========================== Working on docker volumes ===============================**

32 To see the list of all the docker volumes

docker volume ls

33 To create a docker volume

docker volume create volume\_name

34 To get detailed info about a volume

docker volume inspect volume\_name/volume\_id

35 To delete a volume

docker volume rm volume\_name/volume\_id

================================GHOST============================================



---

version: '3'

services:

DB:

image: mysql

environment:

MYSQL\_ROOT\_PASSWORD= pammi

GHOST:

image: ghost

ports:

- 3333: 2368

...

==============================JENKINS=============================================

---

version: '3'

services:

My\_jenkins:

image: jenkins

ports:

- 5050: 8080

QA\_Server:

image: tomcat

ports:

- 6060: 8080

PROD\_Server:

image: tomcat

ports:

- 7070: 8080

...

================================= LAMP ==========================================

---

version: '3'

services:

lamp\_db:

image: mysql

environment:

MYSQL\_ROOT\_PASSWORD: pammi

Apache:

image: httpd

ports:

- 1212: 80

PHP:

image: php:7.2-apache

ports:

- 1213: 80

...

==================================== SELENIUM =================================

---

version: '3'

services:

HUB:

image: selenium/hub

ports:

- 4444: 4444

Chrome:

image: selenium/node-chrome-debug

ports:

- 5901: 5900

Firefox:

image: selenium/node-firefox-debug

ports:

- 5902: 5900

...

==================================== WORDPRESS ==========================

---

version: '3'

services:

mydb:

image: mysql:5

environment:

MYSQL\_ROOT\_PASSWORD: pammi

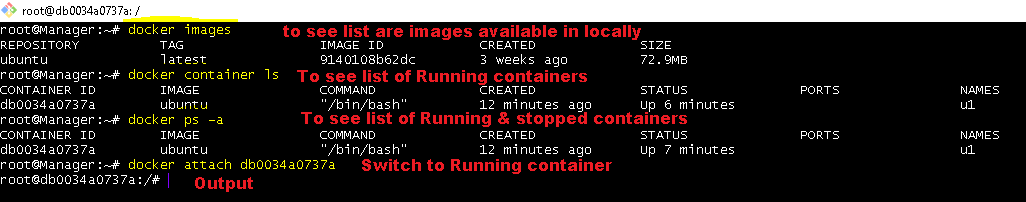
wordpress:

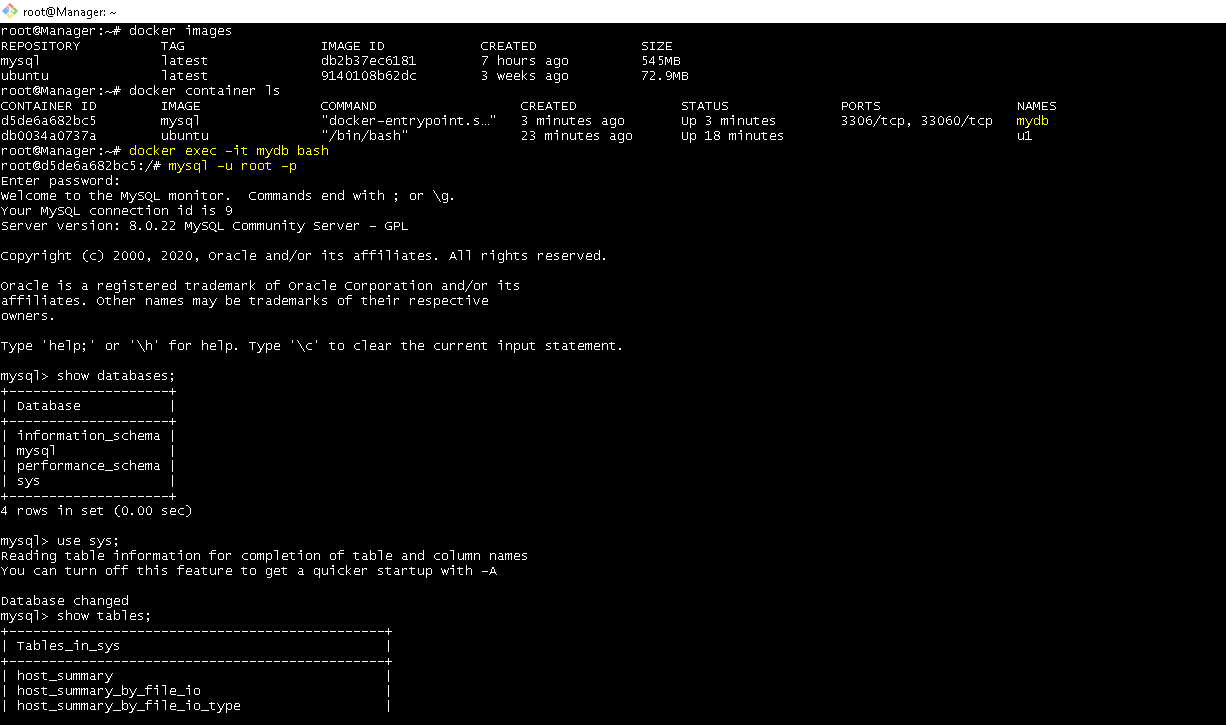
image: wordpress

ports:

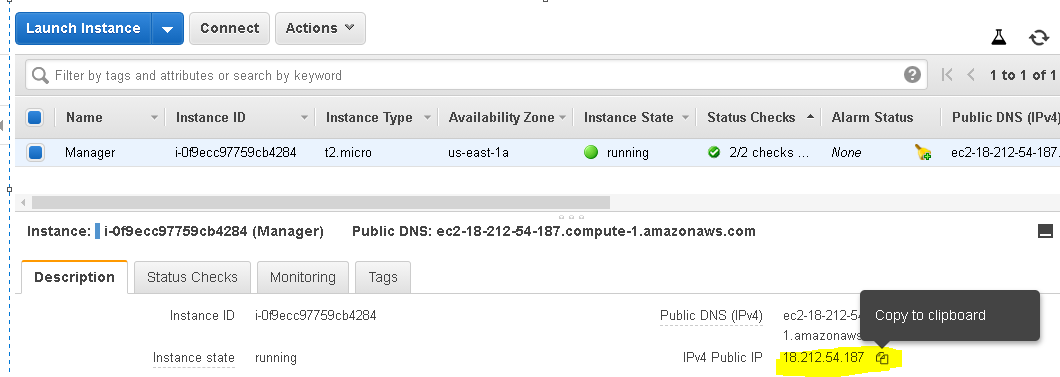
- 8888: 80

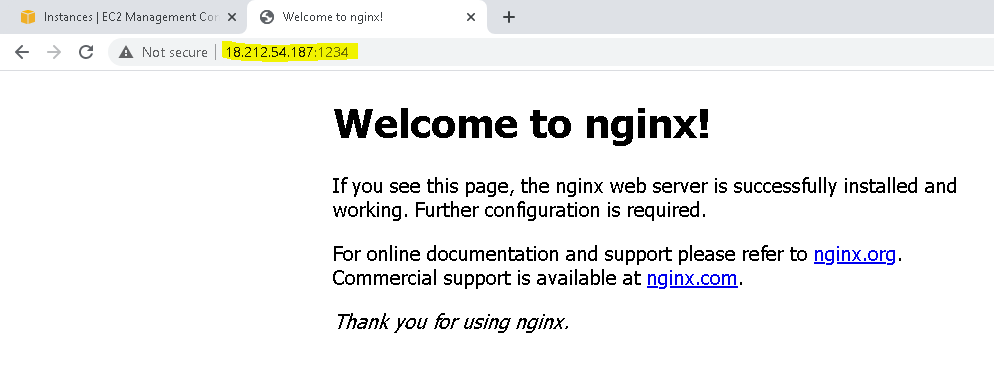
...







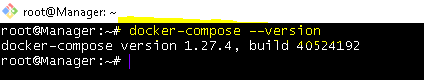




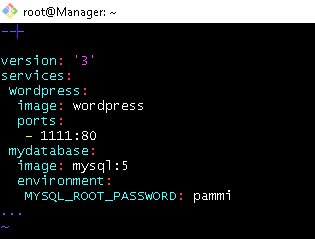
**Docker-Compose**

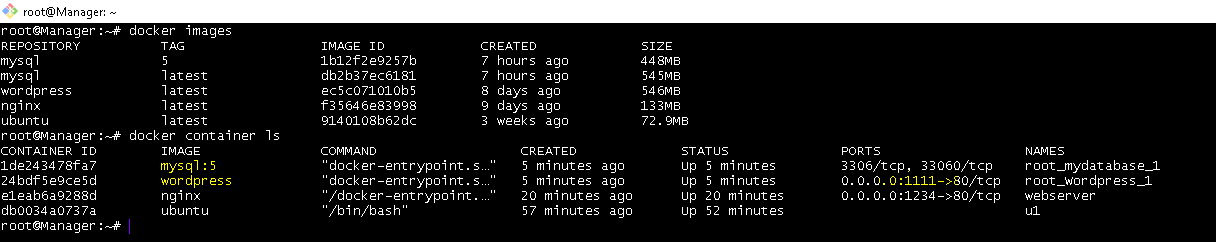
**Installation:**

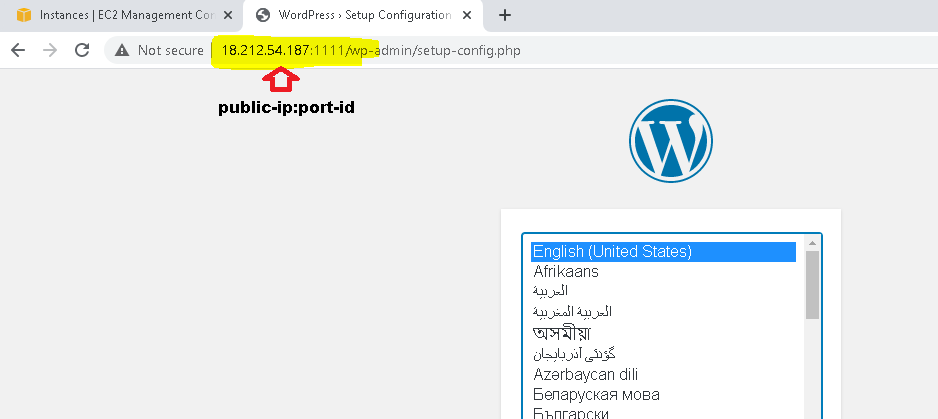
1. sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
2. sudo chmod +x /usr/local/bin/docker-compose



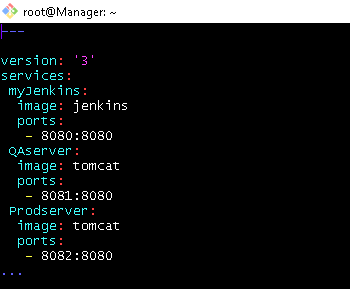
**WORDPRESS**

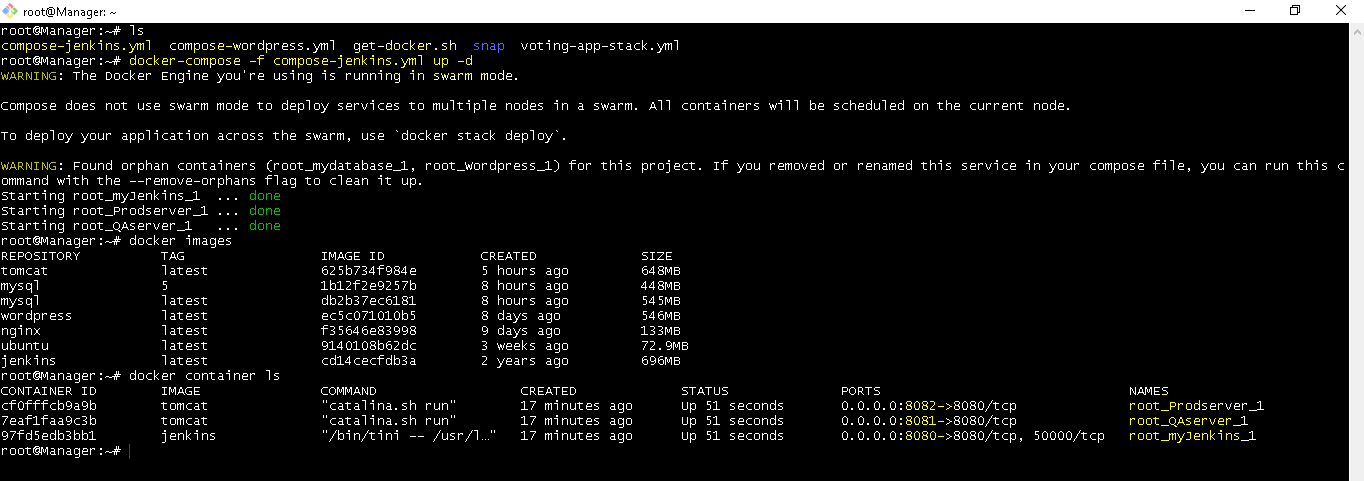


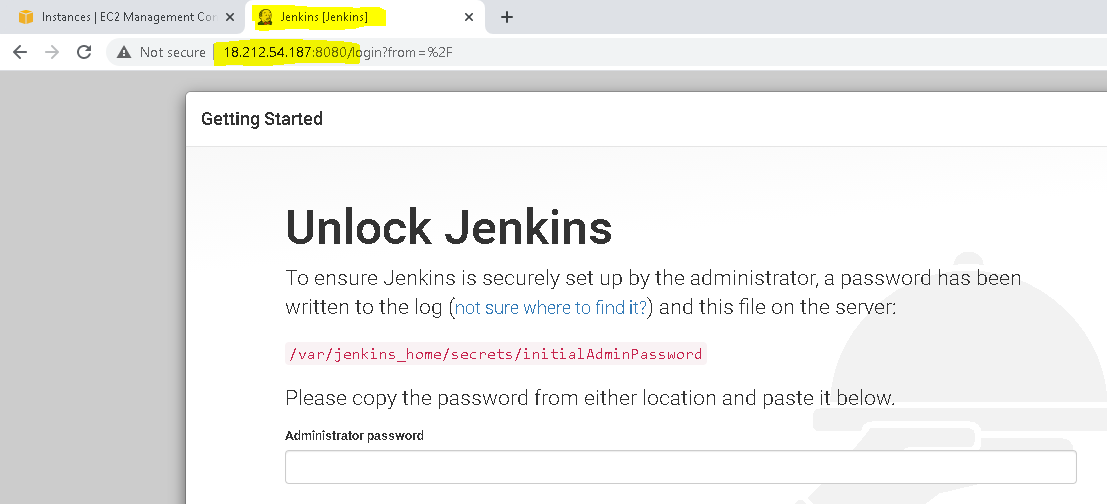




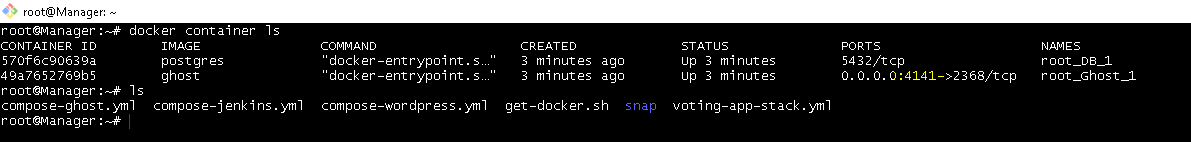
Jenkins-compose

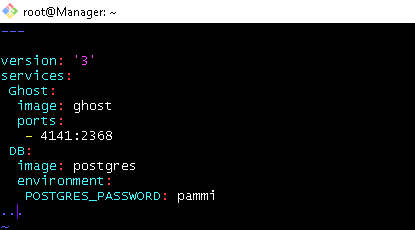


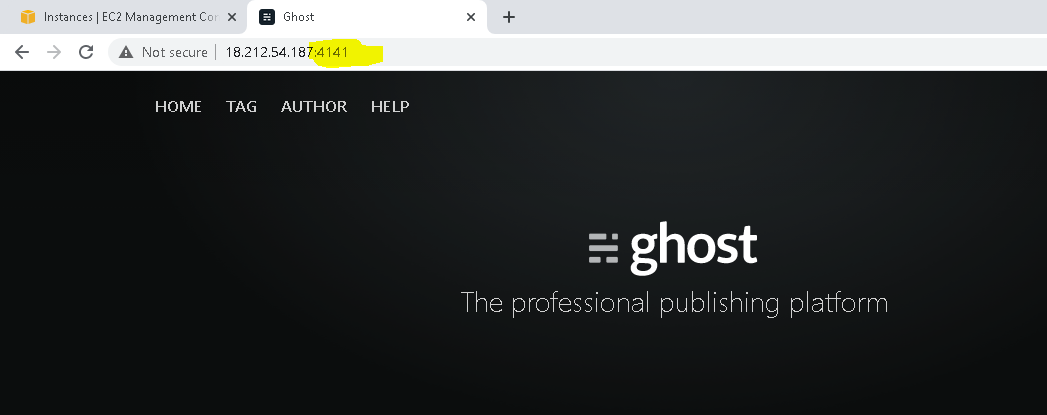




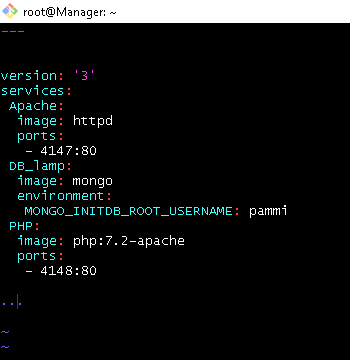
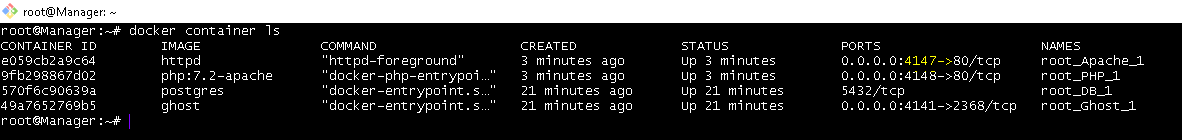
GHOST

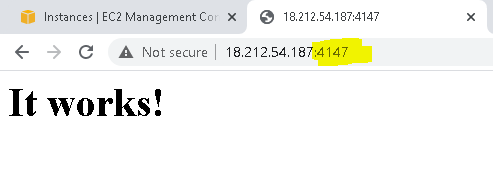




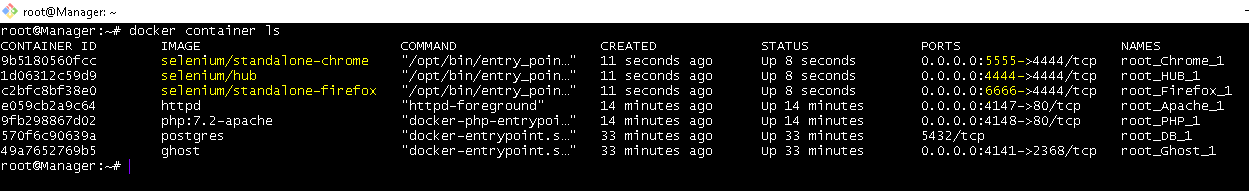


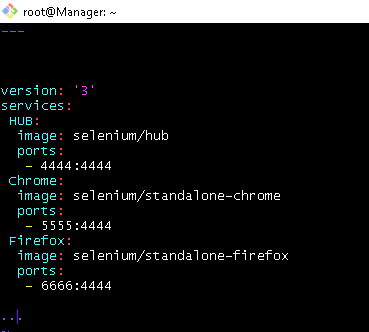
**LAMP --- Architecture**

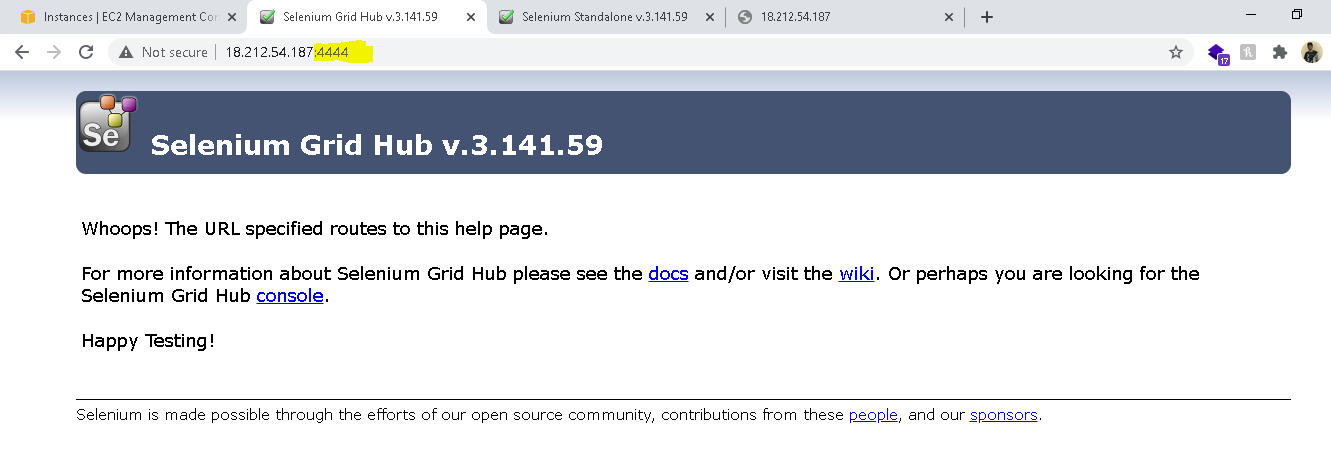


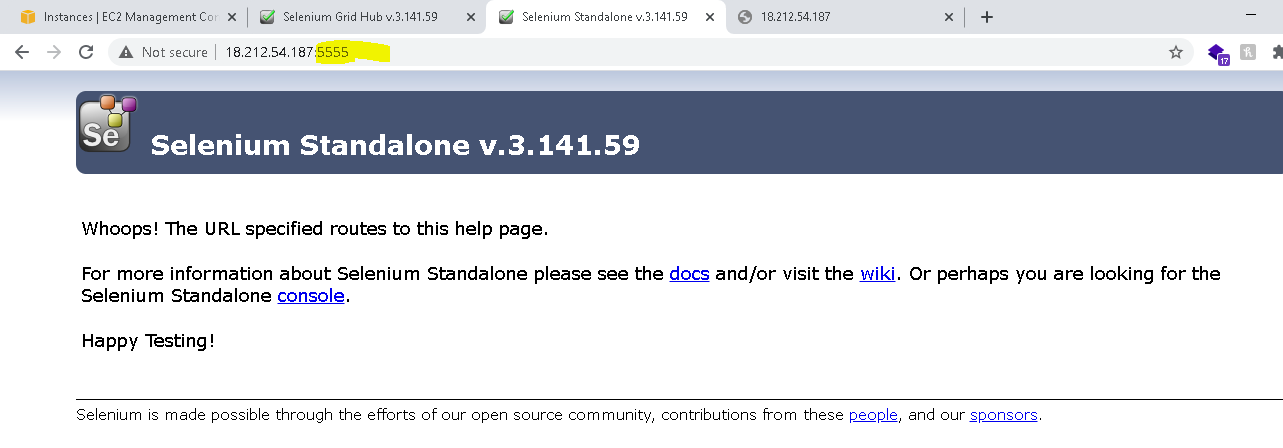


**SELENIUM PROJECT**









**Docker Swarm setup**

Manager

**Install docker on Manager, worker1, worker2**

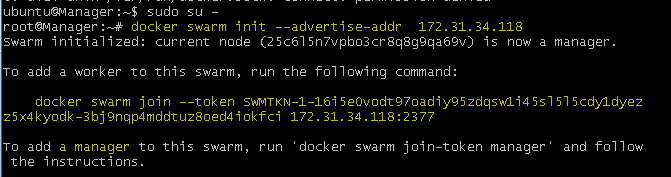
This script is meant for quick & easy install via:

# $ **curl -fsSL https://get.docker.com -o get-docker.sh**

**# $ sh get-docker.sh**

**Configure the Manager Node for Swarm Cluster Initialization**

**$ docker swarm init --advertise-addr <Private\_ip\_Manager>**

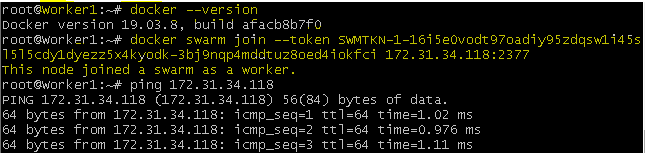


**Configure Worker Nodes to join the Swarm Cluster**

**Go to worker1**

Check docker is installed or not

# docker - - version

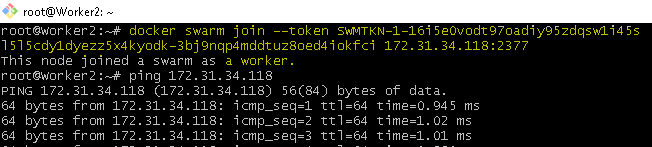
If docker is installed copy the token code from Manager and paste it in Worker1

**Go to worker2**

Check docker is installed or not

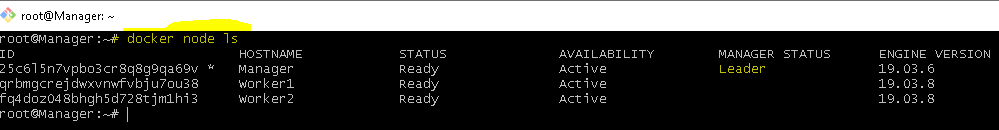
**# docker - - version**

If docker is installed copy the token code from Manager and paste it in Worker2



**Verify the Swarm Cluster**

**$ docker node ls**

****

**If at any time, you lost your join token, it can be retrieved by running the following command on the manager node for the manager token:**

**$ docker swarm join-token manager –q**

**The same way to retrieve the worker token run the following command on the manager node:**

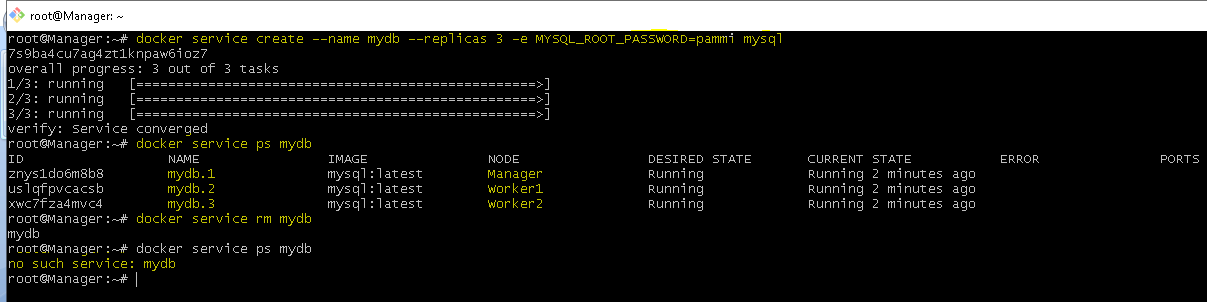
**docker swarm join-token worker –q**

**Load Balancing**

mysql(database)

**Start mysql with 3 replicas in swarm**

**docker service create --name mydb --replicas 3 -e MYSQL\_ROOT\_PASSWORD=pammi mysql**

****

**SCALING**

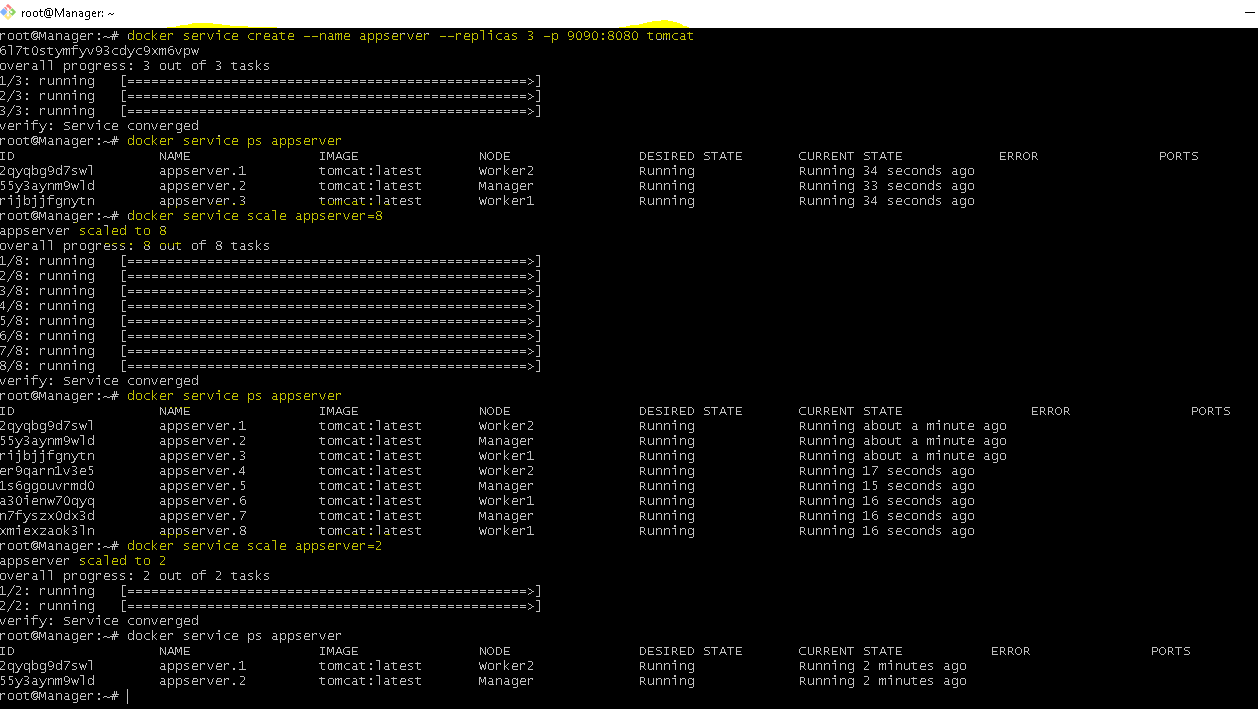
**By using scaling we can Increase or decrease the clusters**

**Creation# docker service create --name appserver --replicas 3 -p 9090:8080 tomcat**

**Validation# docker service ps appserver**

**Increase: docker service scale appserver=8**

**Decrease: docker service scale appserver=2**

****

**Rolling updates**

**Create: docker service create --name mydb --replicas 5 redis:3**

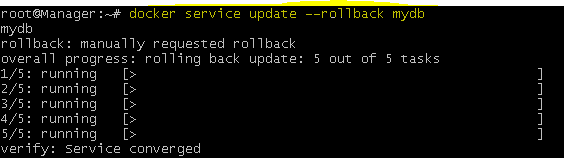
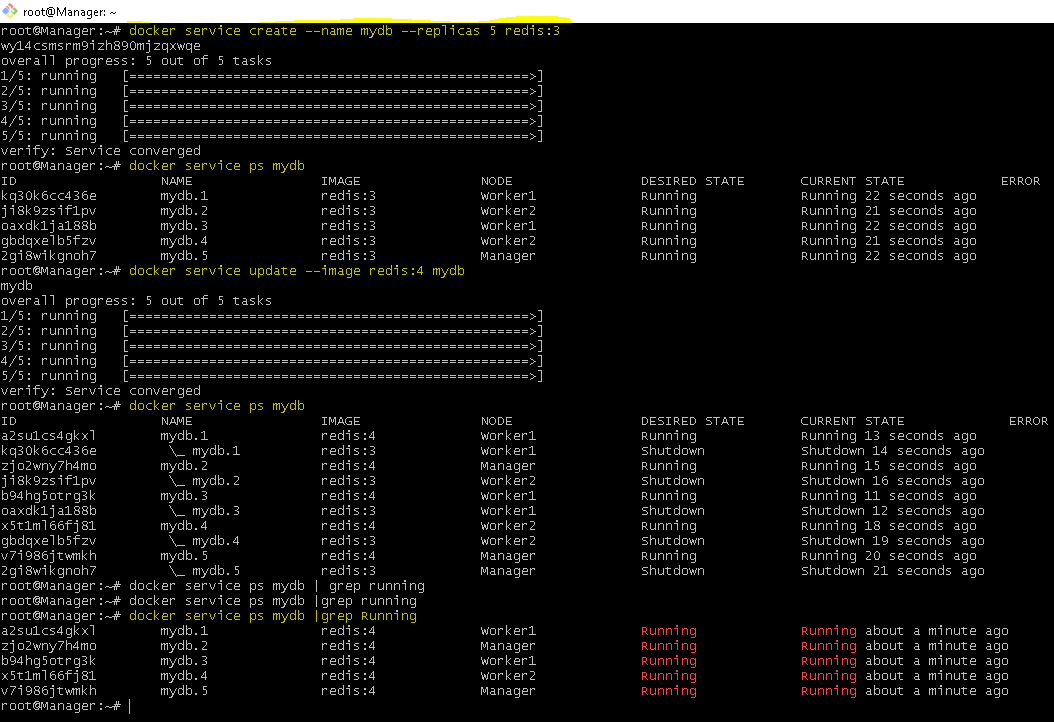
**Validation : docker service ps mydb**

**Upgrading: docker service update --image redis:4 mydb**

**Validation: docker service ps mydb |grep Running**

**Downgrading: docker service update --rollback mydb**

**Validation: docker service ps mydb |grep Running**

****

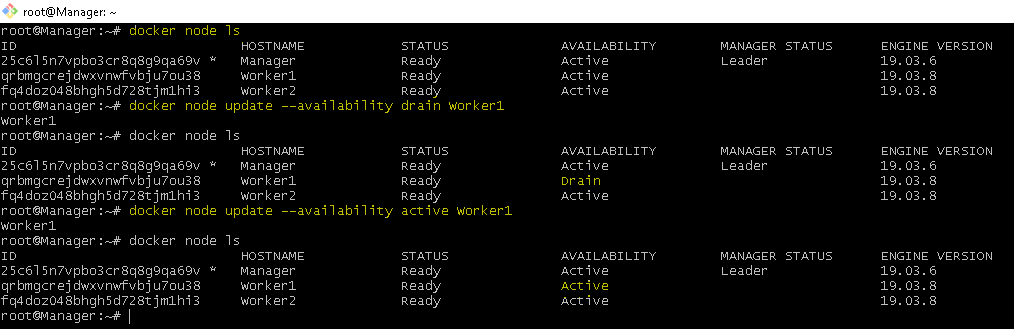
**Access( Bench Release)**

**Here we have 1 Manager & ( Worker1 , Worker2)**

**To check available status: root@Manager:~# docker node ls**

**De-Active: root@Manager:~# docker node update --availability drain Worker1**

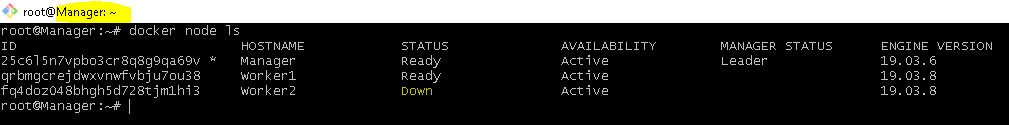
**Active: root@Manager:~# docker node update --availability active Worker1**

****

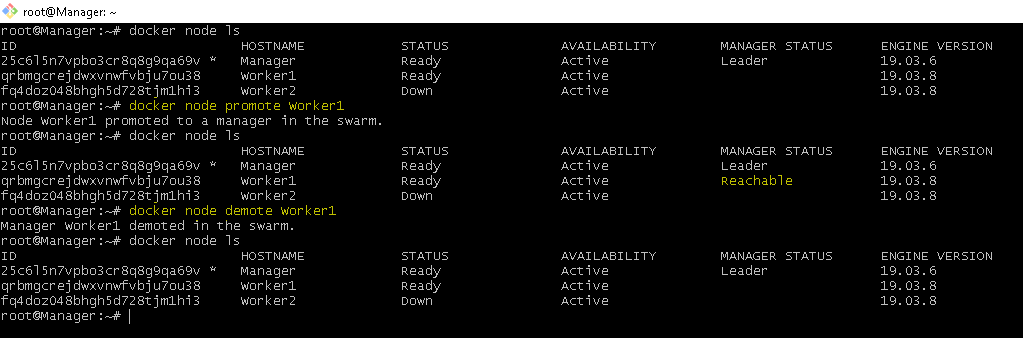
**Exit from Swarm**

**Here Worker2 want to exit from swarm**

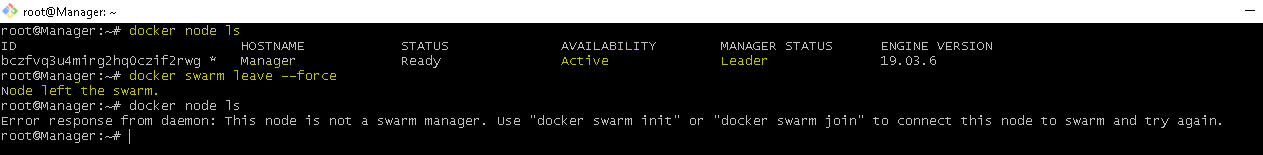
****

****

**Promotions&de-promote**

****

**Manager Leave the swarm**

****

**Docker Stack**

**Docker compose + Swarm = Stack **

**============================= Wordpress ===============================**

**---**

**version: '3'**

**services:**

**Wordpress:**

**image: wordpress**

**ports:**

**- 1111: 80**

**deploy:**

**replicas: 3**

**Mydb:**

**image: mysql5**

**environment:**

**MY\_ROOT\_PASSWORD: pammi**

**deploy:**

**replicas: 2**

**...**

**============================= Jenkins =============================**

**---**

**version: '3'**

**services:**

**Devserver:**

**image: jenkins**

**ports:**

**- 5050:8080**

**deploy:**

**replicas: 2**

**placements:**

**constrains:**

**- node.hostname == worker2**

**QAserver:**

**image: tomcat8**

**ports:**

**- 6060:8080**

**deploy:**

**replicas: 2**

**placements:**

**constarins:**

**- node.hostname == Manager**

**Prodserver:**

**image: tomcat8**

**ports:**

**- 7070:8080**

**deploy:**

**replicas: 3**

**placements:**

**constrains:**

**- node.hostname == worker1**

**=============================== LAMP ==================================**

**version: '3'**

**services:**

**apache:**

**image: apache2**

**ports:**

**- 2222:80**

**deploy:**

**replicas: 3**

**resources:**

**limits:**

**cpu: "0.05"**

**memory: "150m"**

**mydb\_lamp**

**image: mysql5**

**environment:**

**MY\_ROOT\_PASSWORD= pammi**

**deploy:**

**replicas: 2**

**resouces:**

**limits:**

**cpu: "0.06"**

**memory: "200m"**

**php:**

**image: php-7.2**

**ports:**

**-3333:80**

**deploy:**

**replicas: 4**

**resources:**

**limits:**

**cpu: "0.01"**

**memory: "75M"**

**Docker basics-**

**Servers: docker run attach container\_name/container\_id**

**Databases:**

**docker run --name mydb -d -e MYSQL\_ROOT\_PASSWORD=pammi mysql:5**

**Validation:**

**docker exec -it mydb bash**

**mysql -u root –p ------------Enter password**

**mysql> use sys;**

**Link options**

**Linking between to busybox**

**Create 2 busybox (b1 & b2)**

**Busybox(b1)** docker run --name b1 -it busybox

comeout out without exit from b1 (CTRL+P,CTRL+Q)

**Busybox(b2)** docker run --name b2 -it --link b1:my\_b1 busybox

**Validation:** ping b1

**Linking between mysq to wordpress**

**Create database(mysql) :**

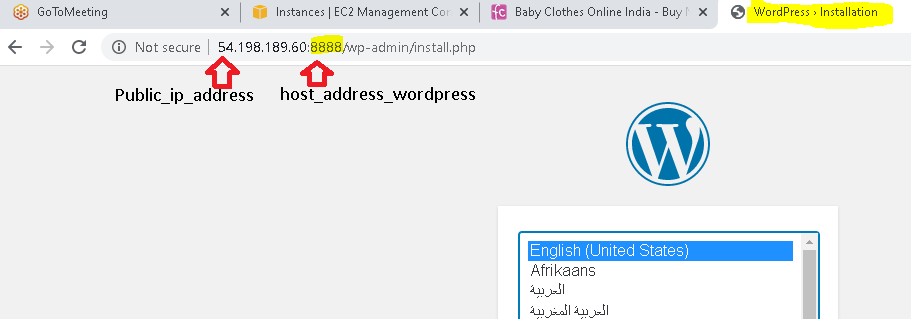
docker run --name mydb -d -e MYSQL\_ROOT\_PASSWORD=pammi mysql:5

**Create wordpress(webserver) :**

docker run --name mywordpress -d -p 8888:80 --link mydb:mysql wordpress

**Validation:**

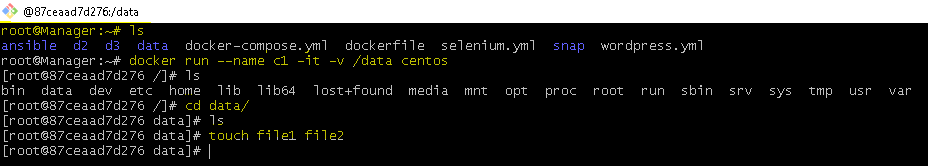
**IPv4 Public IP: 54.198.189.60 :8888**

****

**QA Server(Tomcat)**

VOLUME

**Simple docker volume :**

****

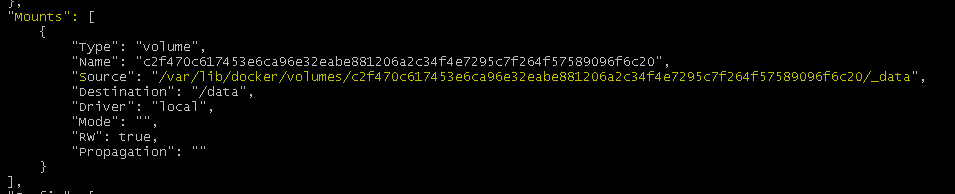
root@Manager:~# mkdir data/

root@Manager:~# docker run --name c1 -it -v /data centos

[root@87ceaad7d276 /]# cd data/

[root@87ceaad7d276 data]# touch file1 file2

root@Manager:~# docker inspect c1

****

**Go to Mounts--🡪 Copy source address**

"Source": **"/var/lib/docker/volumes/c2f470c617453e6ca96e32eabe881206a2c34f4e7295c7f264f57589096f6c20/\_data**",

**Delete container Centos(C1):**

root@Manager:~# docker ps

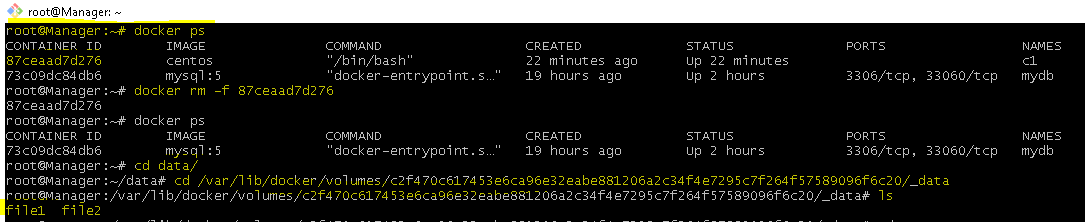
root@Manager:~# docker rm -f 87ceaad7d276 (or) C1

root@Manager:~# cd data/

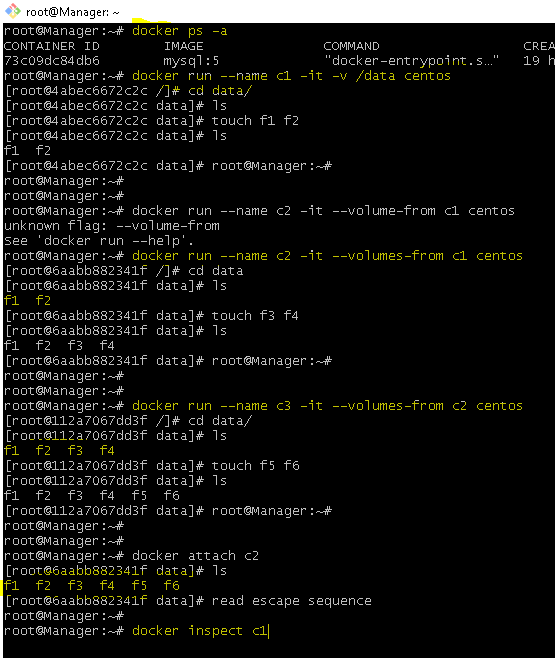
root@Manager:~/data# cd /var/lib/docker/volumes/c2f470c617453e6ca96e32eabe881206a2c34f4e7295c7f264f57589096f6c20/\_data

root@Manager:/var/lib/docker/volumes/c2f470c617453e6ca96e32eabe881206a2c34f4e7295c7f264f57589096f6c20/\_data# ls

Output: file1 file2

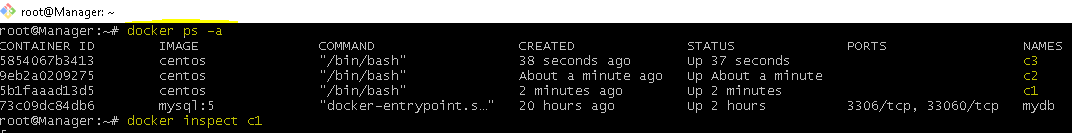
****

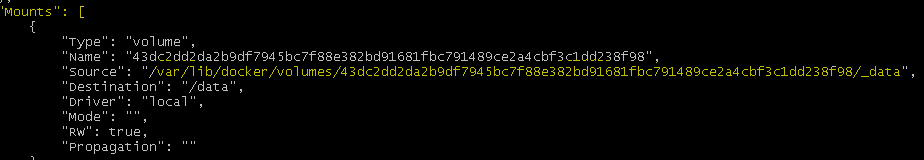
**Sharable docker volume:**

****

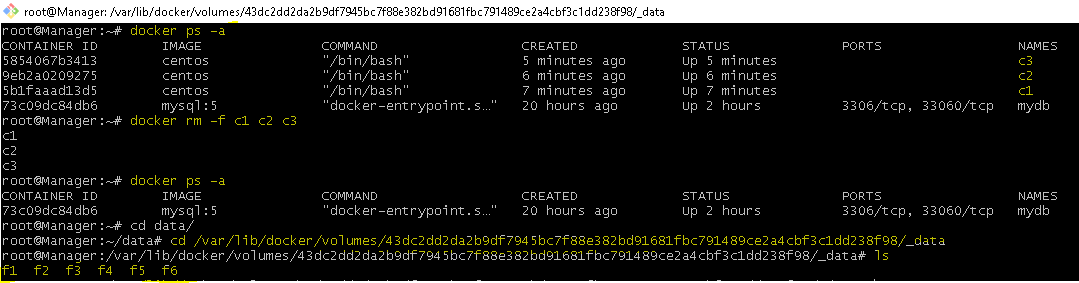
**Go to Mounts--🡪 Copy source address**

**/var/lib/docker/volumes/43dc2dd2da2b9df7945bc7f88e382bd91681fbc791489ce2a4cbf3c1dd238f98/\_data**





**Validation: After detelting the containers we can store the data in volumes**

****

**Docker Volume container: :**

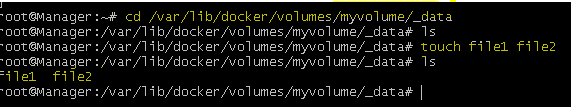
**Docker volume container are bi-directional i.e.,**

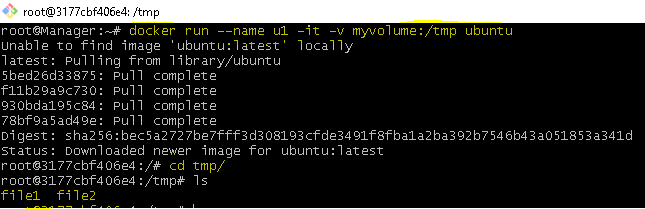
**data from the host machine can be copied into the containers**

**from the container we can save the data into host machine.**

/var/lib/docker/volumes/myvolume/\_data

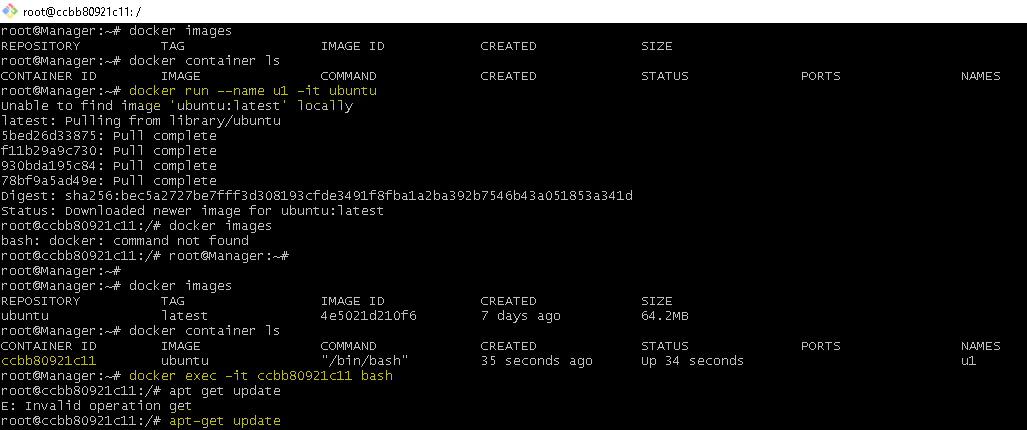
****

****

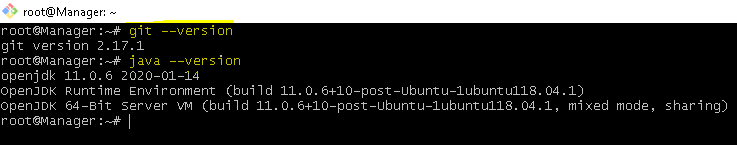
****

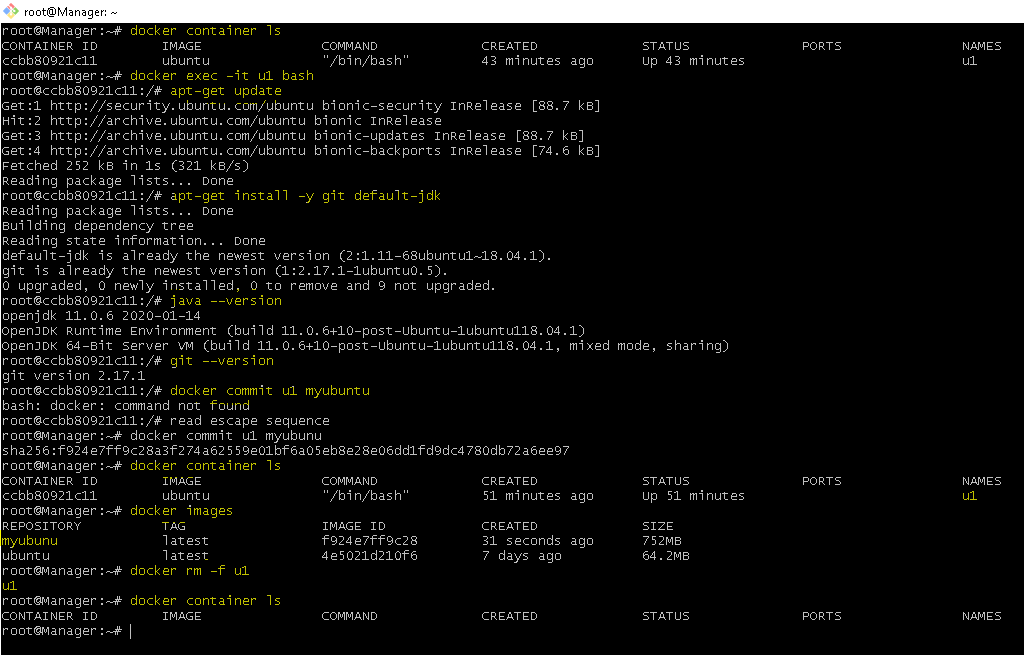
**Creating Docker customized Image**

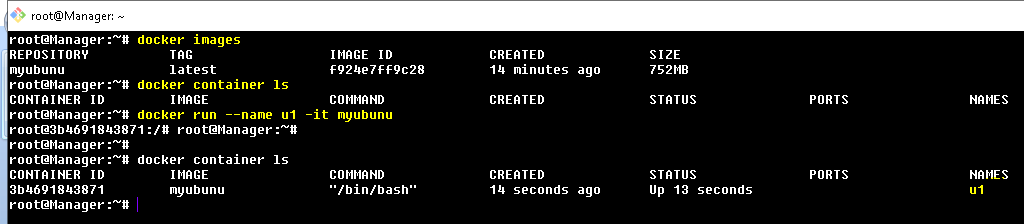
Customized image

****

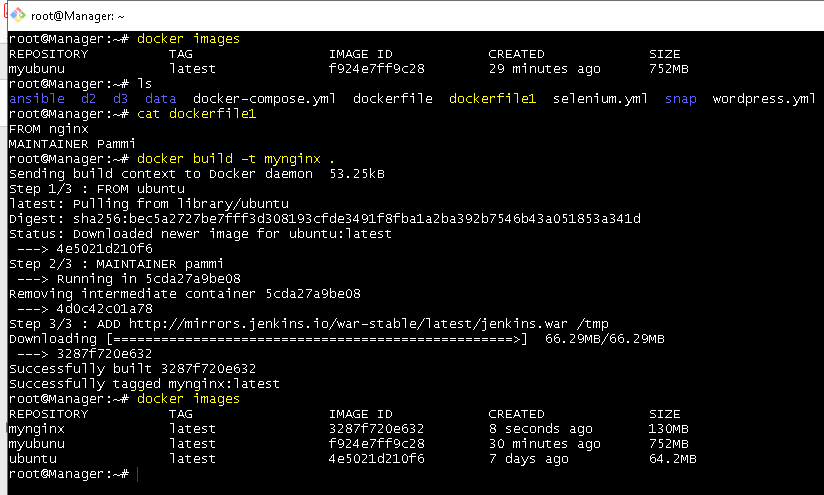
****

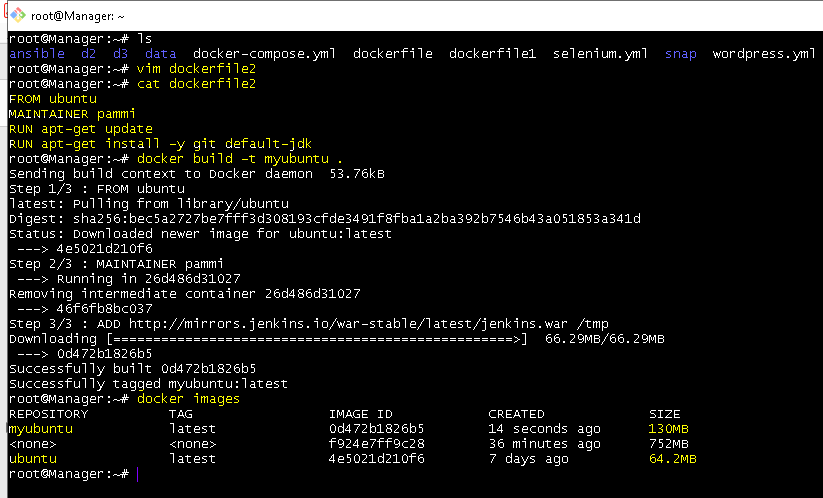
****

****

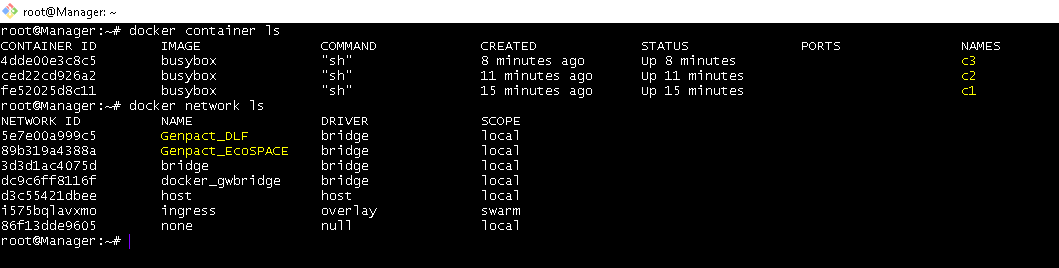
****

**Docker file**

****

****

**Docker Network**



**Git – setup**

**LR**

**LR**

**LR**

**LR**

**Local Machine(individual employee work space)**

**Send file from stagging area back to working Directory (unstage)**

****

**Working Directory**

**$pwd ----🡪 ALEKYA@DESKTOP-3PO147M MINGW64 ~/Desktop/intelliq**

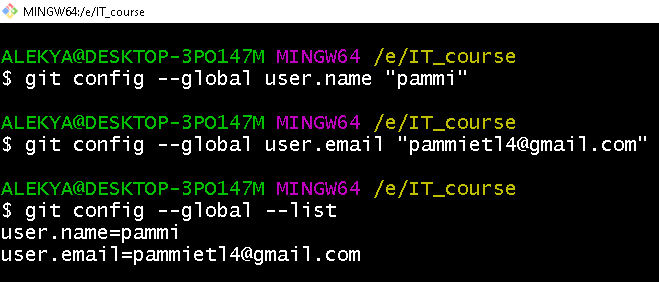
**$git init --🡪 to make intelliq as working directory(Master)**

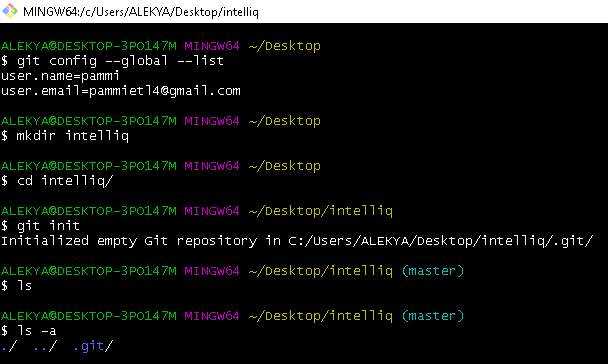
**$ cat .gitignore ----🡪 (private file storage area which cannot be access by git)**

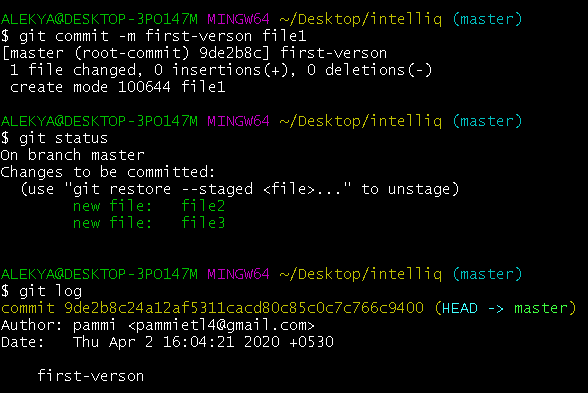
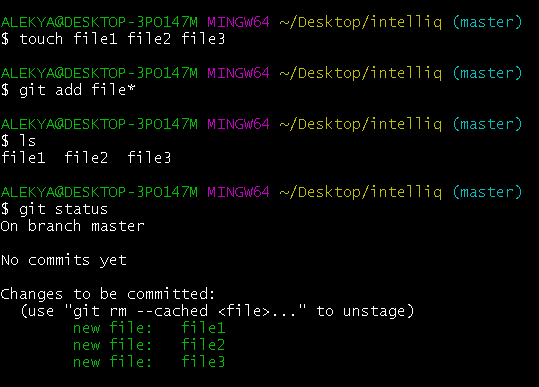
**File1**

**File2**

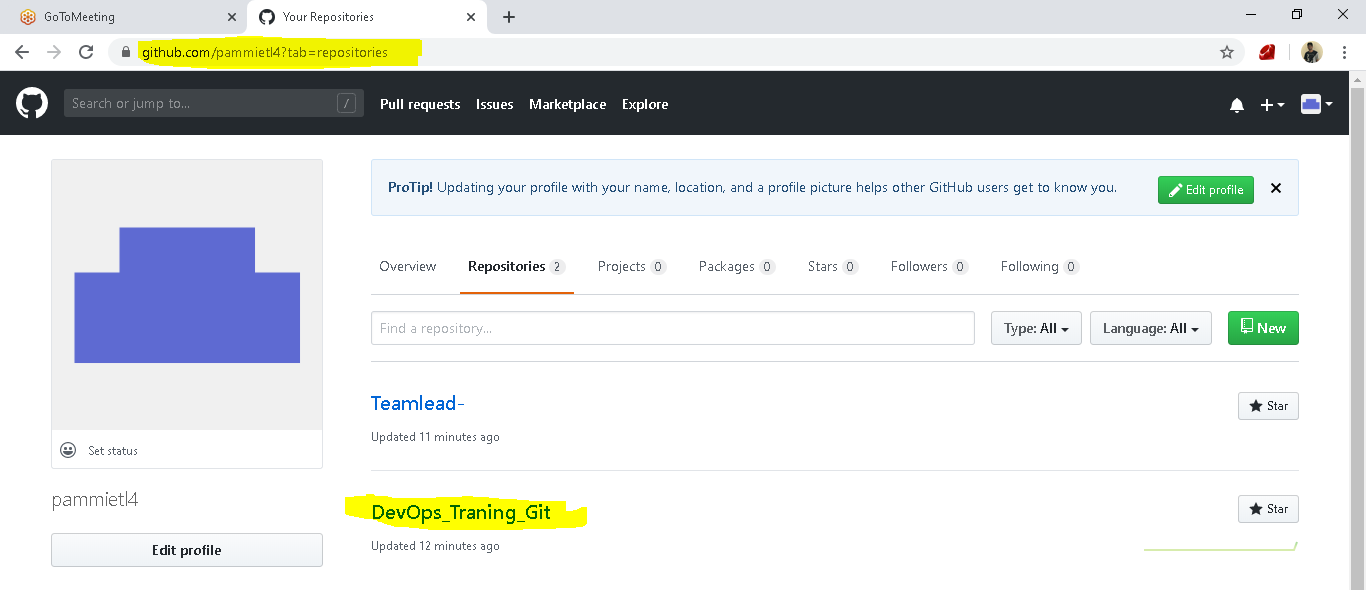
**Ctrl + d**

****

****

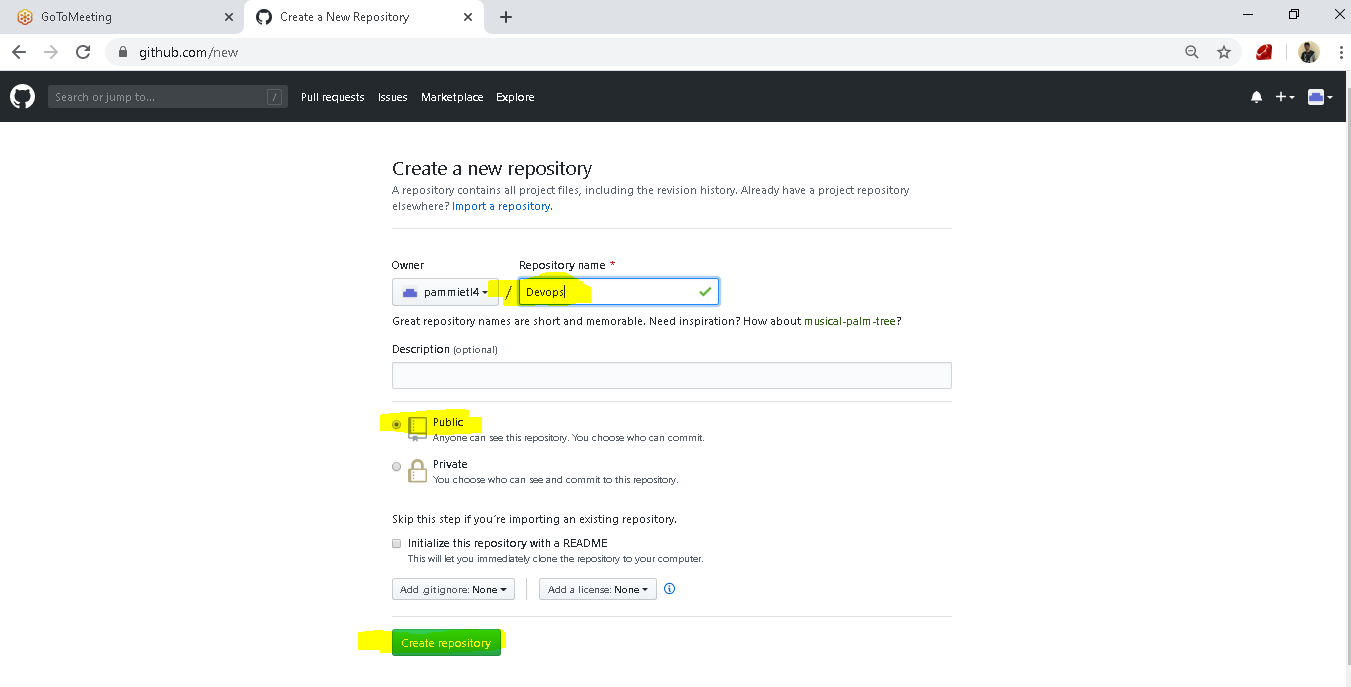
****

**Uploading code from Local Repository(LR) to Remote Repository(RR)**

****

**Local Repository**

**Steps**

****

**Go to Git bash**

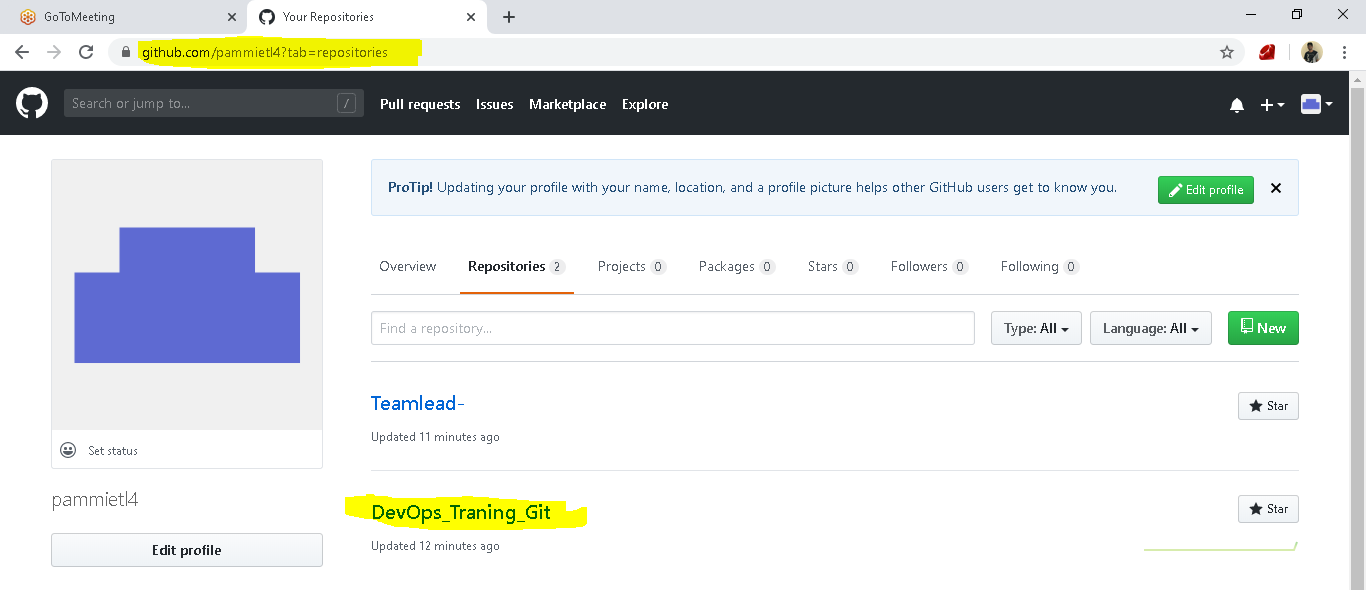
**git remote add origin https://github.com/pammietl4/Teamlead-.git**

**git push -u origin master**

**Downloading from Github to Local Repository**

Local Repository

**(LR)**

****

**We can download the data in 3 ways**

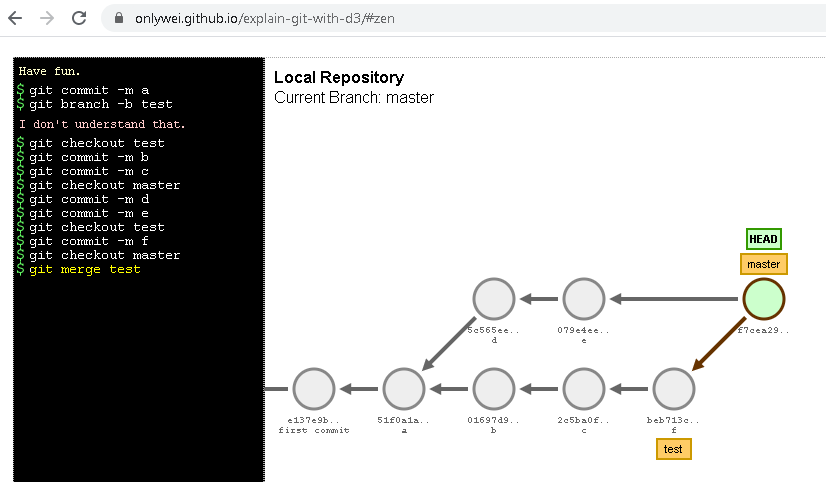
**Git Clone Git Fetch Git Pull**

**GIT RESET**

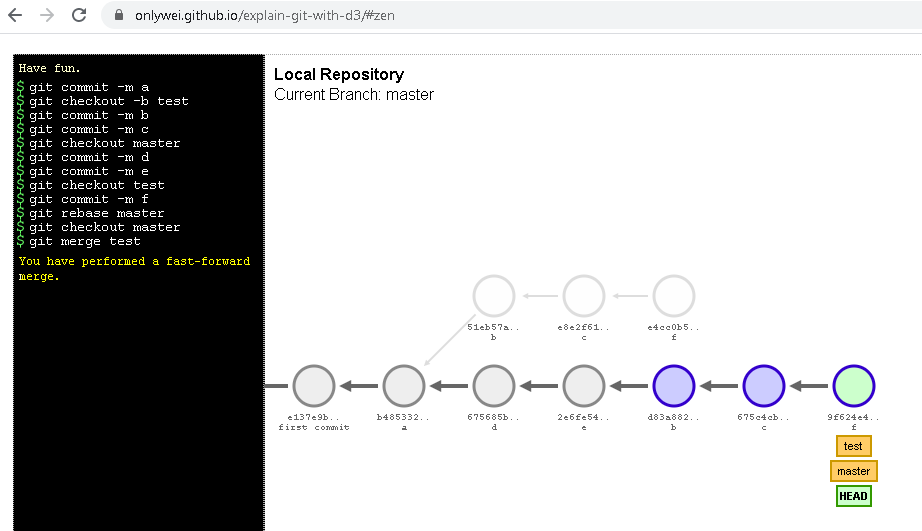
**Ordering the files**

[**https://onlywei.github.io/explain-git-with-d3/#zen**](https://onlywei.github.io/explain-git-with-d3/#zen)

**Git Merging: (it will be merged on time based)**

****

**Git Rebasing:**

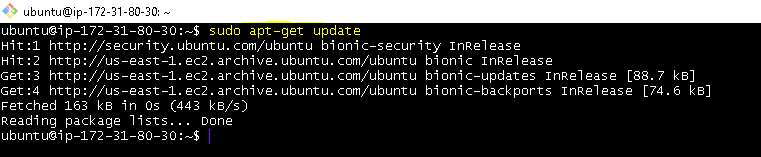
****

**Git Cherry-pick: here we can pick the required**

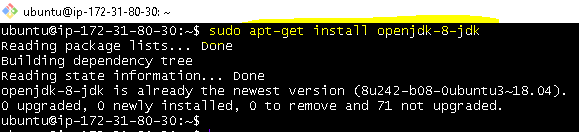
**Jenkins**

**Installing Jenkins:**

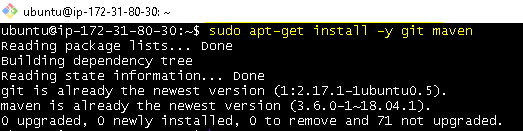
**Update the repository:** sudo apt-get update

****

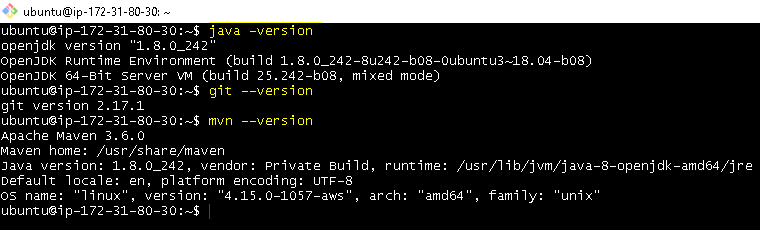
**Install java (version 8) :** sudo apt-get install openjdk-8-jdk

****

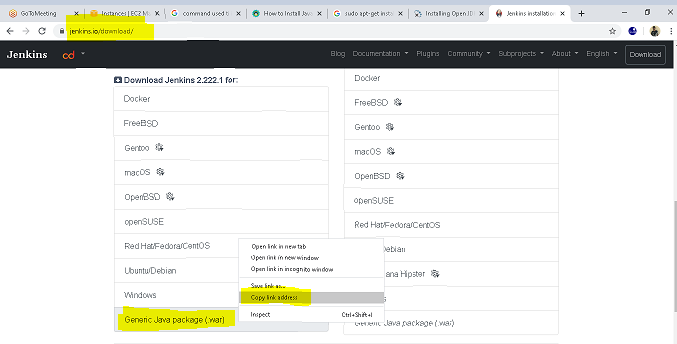
**Install git & maven:** sudo apt-get install -y git maven

****

**Validation:**

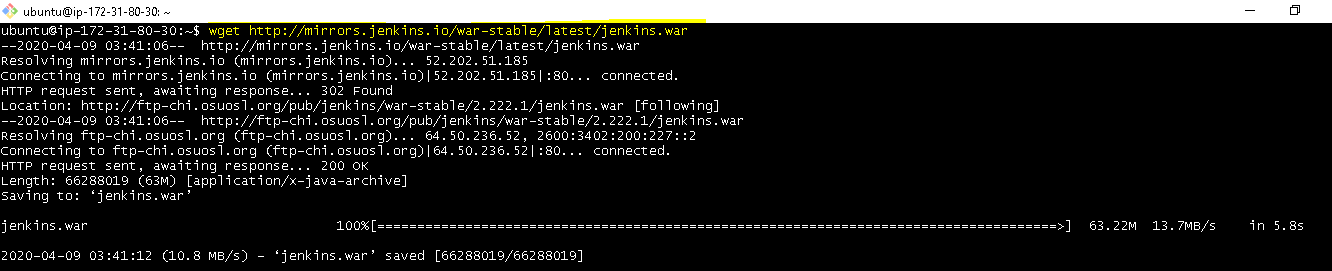
****

**Download Jenkins (copy the below url from chrome )**

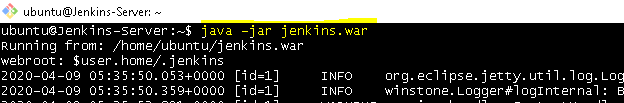
****

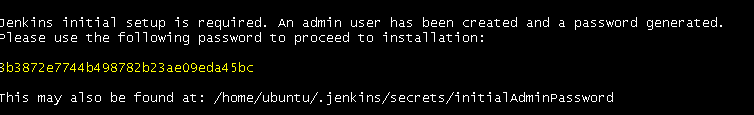
**ubuntu@ip-172-31-80-30:~$**

wget http://mirrors.jenkins.io/war-stable/latest/jenkins.war

****

**To start Jenkins:**

****

****

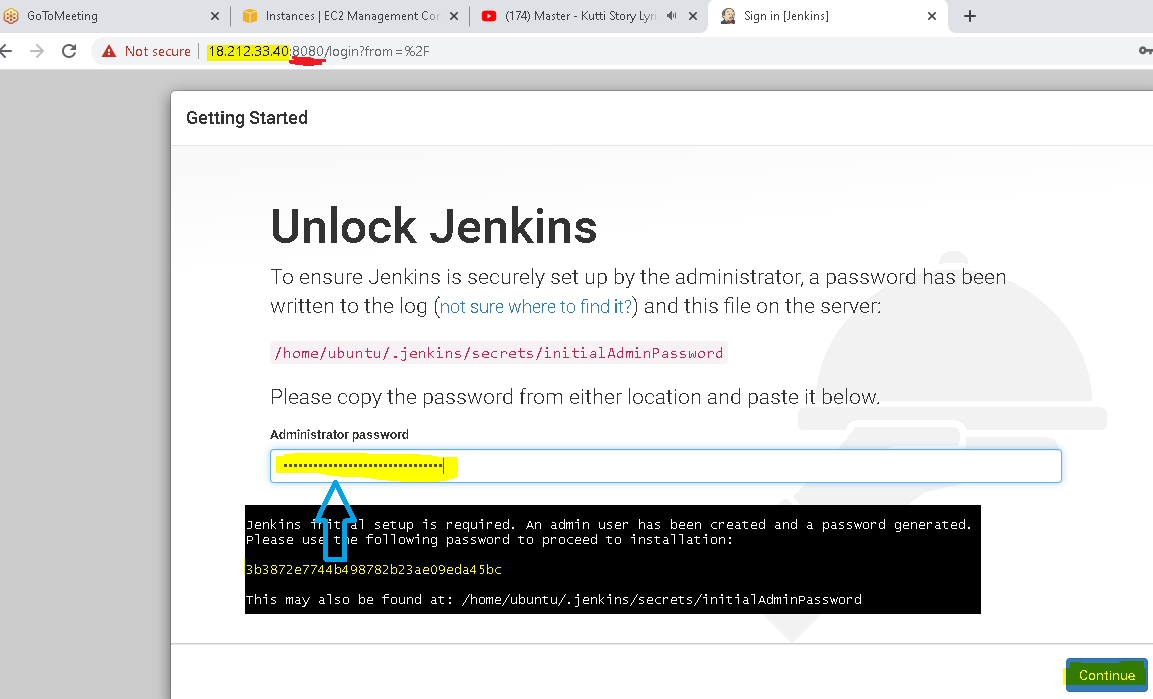
Please use the following password to proceed to installation:

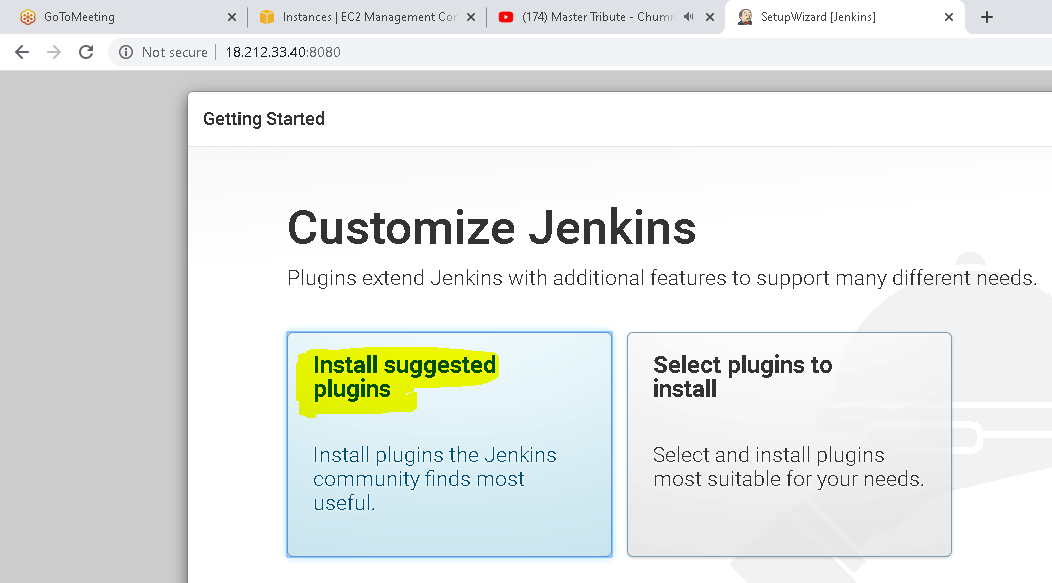
3b3872e7744b498782b23ae09eda45bc

This may also be found at: /home/ubuntu/.jenkins/secrets/initialAdminPassword

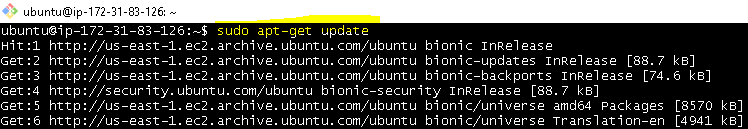
**To access jenkins from browser:**

**IPv4 Public IP 18.212.33.40 Jenkins port no: 8080**

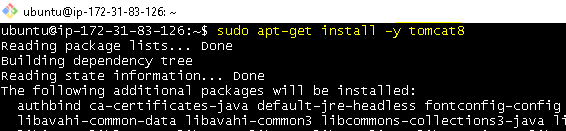
****

****

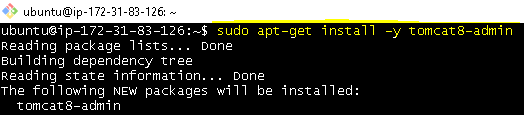
**QA-Server:**

****

ubuntu@ip-172-31-83-126:~$ **sudo apt-get install -y tomcat8**

****

ubuntu@ip-172-31-83-126:~$ **sudo apt-get install -y tomcat8-admin**

****

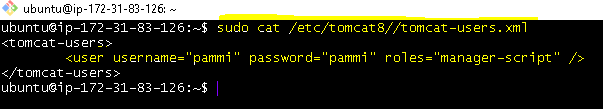
**Edit tomcat-users.xml file:**

ubuntu@ip-172-31-83-126:~$ **sudo vim /etc/tomcat8/tomcat-users.xml**

**<tomcat-users>**

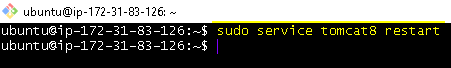
**<user username="pammi" password="pammi" roles="manager-script" />**

**</tomcat-users>**

****

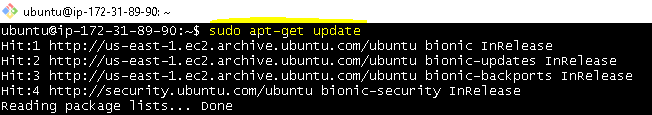
**Restart tomcat8**

ubuntu@ip-172-31-83-126:~$ **sudo service tomcat8 restart**

****

**Prod-server**

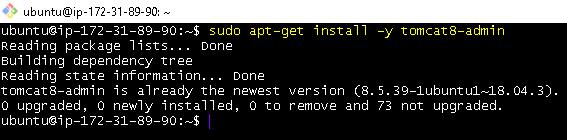
ubuntu@ip-172-31-89-90:~$ **sudo apt-get update**

****

ubuntu@ip-172-31-89-90:~$ **sudo apt-get install -y tomcat8**

****

ubuntu@ip-172-31-89-90:~$ **sudo apt-get install -y tomcat8-admin**

****

**Edit tomcat-users.xml file:**

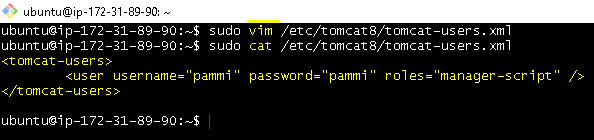
ubuntu@ip-172-31-89-90:~$ **sudo vim /etc/tomcat8/tomcat-users.xml**

ubuntu@ip-172-31-89-90:~$ **sudo cat /etc/tomcat8/tomcat-users.xml**

**<tomcat-users>**

**<user username="pammi" password="pammi" roles="manager-script" />**

**</tomcat-users>**



**Restart tomcat8**

****

