LINUX Command :-

Copy file from AWS remote Server
scp -i mirafraa.pem ubuntu@18.206.201.43:/home/ubuntu/nginx.tar.bz2 /home/pramod/Downloads/

C:\>pscp -i yourkey.ppk yourfilename ubuntu@18.206.201.43:/home/Pramod/down/

Cut line from one variable in linux
$export SERVER=ca47dwviss007.caremore.com
$echo $SERVER
$echo $SERVER | cut -d \. -f 1

Delete Files older than 10 days
$sudo find /tmp -type f -atime +10 -delete
Delete all files and folders older than 10 days
$find /tmp -ctime +10 -exec rm -rf {} +
Delete file with size:- $find . -name "*.tif" -type 'f' -size -160k -delete
Delete file with extension and & Size
find -type f \(-name "*zip" -o -name "*tar" -o -name "*gz" \) -size +1M -exec rm {} +
Apache Server Error Log
$ sudo tail -100 /var/log/apache2/error.log
$ sudo grep -i invalid /var/log/apache2/error.log
$ tail -f access_log | grep "HTTP/1\.[01]\" 50." >> /tmp/log_error_capture.txt &
Increment variable with +1
$variable: N=`expr $N + 1` Increment N by one.

File Permissions:-
chmod - modify file access rights
chown - change file ownership
chgrp - change a file's group ownership

0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read +Write	rw-
7	Read + Write +Execute	rwx

Networking Commands:
Check the live port status in server
$ lsof -I:22
$ sudo netstat -tulpn | grep LISTEN
$ netstat -atun
$ ifconfig
$ ovs-vsctl show
$ ip route

```
$ netstat -nr
$ cat /etc/network/interfaces
$ sudo ifconfig eth0 192.168.1.200 netmask 255.255.255.0
$ sudo route add default gw 192.168.1.1 eth0
$ route -n
```

Check the system configuration

```
$ lscpu
```

Add Port in firewall permanently

```
$ sudo firewall-cmd --permanent --add-port={9200, 9300}/tcp
$ sudo firewall-cmd --reload
```

Check OS version

```
$ cat /etc/os-release
```

View the user list in linux

```
$ cut -d: -f1 /etc/passwd
$ awk -F: '{ print $1}' /etc/passwd
$ adduser <username>
$ passwd
```

Give sudo privilege

```
$ usermod -aG sudo <username>
```

Login this user

```
$ sudo su - username
```

User non expiry

```
$ passwd -x -1 <username>
```

USE SSH connection between two system : SSh connection use password authentication

```
# Enable system with username & password
$ vi /etc/ssh/sshd_config
```

Change the passwordauthentication yes then save the file

```
# permitRootlogin yes
# passwordAuthentication yes
```

Restart the ssh

```
$ systemctl restart ssh
$ systemctl enable ssh
# Generate ssh key
$ ssh-keygen
$ ssh-copy-id user@host_ip_address

$ service sshd restart
```

To edit the sudo user access for no password authentication

```
$ vi /etc/sudoers
```

Add this end of file

```
<Username>     ALL=(ALL) NOPASSWD:ALL
```

To remove forcefully folder or file

```
$ sudo rm -rf folderName
```

Remove all file & folder together in current directory

```
$ rm -rf *
```

Disk Related command
$ sudo lsblk
$ df -h
$ du -hs *
Linux File system
Ext2., jfs. ReiserFS. XFS. Btrfs.

Format and Mount an Attached EBS Volume
$ lsblk
$ sudo file -s /dev/xvdf
$ sudo file -s /dev/xvda1 (If the device has a file system, the command shows information about the file system type.)
$ sudo yum install xfsprogs
$ sudo mkfs -t xfs /dev/xvdf
$ sudo mkdir /data
$ sudo mount /dev/xvdf /data
To mount an attached volume automatically after reboot
$ sudo cp /etc/fstab /etc/fstab.orig
$ sudo lsblk -o +UUID (find the UUID of the device)
$ sudo vim /etc/fstab (Edit the fstab file and add UUID of device)
Example:- { UUID=aebf131c-6957-451e-8d34-ec978d9581ae /data xfs defaults,nofail 0 2 }
$ sudo mount -a

docker pull docker.bintray.io/jfrog/artifactory-oss

Aws Instance ip:- 3.13.79.163

ssh -i ubuntuaws.pem ubuntu@3.16.82.170

ssh -i /Users/pramods/ubuntuaws.pem ubuntu@3.13.113.208

Docker Tool

Docker Role in Devops

i. Build the Application
ii. Run the Application
iii. Ship the Application

RUN executes command(s) in a new layer and creates a new image. E.g., it is often used for installing software packages.
CMD sets default command and/or parameters, which can be overwritten from command line when docker container runs.

ENTRYPOINT configures a container that will run as an executable.

Container is a system / A running state of a image is a container
Container using union file system
Task we do:
1- Manage Images
2- Manage Container
3- Manage Network
4- Docker Compose
5- Docker Volume
6- Docker image Creation

- Docker build
- Docker pull
- Docker run

Home Directory of Docker: (/var/lib/docker)
/Users/pramods/Downloads

Install Docker In one command

$ sudo curl -fsSL get.docker.com | /bin/bash

Install Docker In Instance

Refer the link :-
https://docs.docker.com/install/linux/docker-ce/ubuntu/
https://hub.docker.com/

First, update your existing list of packages:
$ sudo apt-get update
Next, install a few prerequisite packages which let apt use packages over HTTPS:
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
Then add the GPG key for the official Docker repository to your system:
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Add the Docker repository to APT sources:
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
Next, update the package database with the Docker packages from the newly added repo:
$ sudo apt-get update
Make sure you are about to install from the Docker repo instead of the default Ubuntu repo:
$ apt-cache policy docker-ce

Finally, install Docker:
$ sudo apt-get install -y docker-ce

Start the services of Docker
$ sudo service docker start

Check that it's running:
$ sudo systemctl status docker

Give user to docker permission
$ sudo usermod -aG docker <username>
$ sudo gpasswd -a jenkins docker
$ sudo chmod 777 /var/run/docker.sock
$ sudo usermod -aG docker $USER

Commands to Install Docker Machine

1. $ base=https://github.com/docker/machine/releases/download/v0.16.0 &&
curl -L $base/docker-machine-$(uname -s)-$(uname -m) >/tmp/docker-machine &&
sudo install /tmp/docker-machine /usr/local/bin/docker-machine

2. $ docker-machine version

Reference : https://docs.docker.com/machine/install-machine/#install-machine-directly

Commands to Install Docker Compose
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.25.3/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose
$ docker-compose --version

Reference : https://docs.docker.com/compose/install/#install-compose

Install Docker Image from hub.docker.com

$ docker pull ubuntu (by default latest version of image will pull)
$ docker pull ubuntu:18.04 (for particular version pull)

To view the running container
$ docker ps
To view the all container
$ docker ps -a
Create container with login in same container
$ docker run -ti --name=(name of container) (image id) /bin/bash
Ex:- $ docker run -ti --name=test 7565dg3546 /bin/bash
Create container with detachmode

$ docker run -tid --name=(name of container) (image id) /bin/bash
Ex:- $ docker run -tid --name=test 7565dg3546 /bin/bash
To inspect / View the container
$ docker inspect <container name>
Ex:- $ docker inspect test1
To inspect / View the Image
$ docker inspect <image name / ID>
Ex:- $ docker inspect ubuntu
Login to particular container
$ docker attach <container name>
Ex:- $ docker attach test1
Login with execute
$ docker exec -ti <container name> /bin/bash
Ex:- $ docker exec -ti test1 /bin/bash
To View the container logs
$ docker logs <container name>
Ex:- $ docker logs test1
To stop the container
$ docker stop <container name>
Ex:- $ docker stop test1
To come out from running container
Hold :- CTRL+P+Q
Remove the container
$ docker rm <container name>
Ex:- $ docker rm test
Remove forcefully container
$ docker rm -f <container name>
Ex:- $ docker rm -f test

Copy files from local to inside container
$ docker cp (file name) (container ID):/file name

Create container with port forwarding command
$ docker run -ti --name=(container name) -p 80:80 (image id) /bin/bash
Ex:- $ docker run -ti --name=test2 -p 80:80 75623cd1 /bin/bash

Make an Apache2 web-server in container
First make a container with port forwarding
$ docker run -ti --name=(container name) -p 80:80 (image id) /bin/bash
$ apt-get update
$ apt-get install apache2 -y
$ dpkg -l apache2
$ service apache2 status

$ service apache2 start
$ apt-get update
$ apt-get install vim (install VI editor)
$ apt-get install curl (its will view of localhost page)

Bridge Network is default network of container
To view the Docker network
$ docker network ls

How to Build a custom image for Docker container :-
$ docker image build ubuntu pramodksahoo/ubuntulatest:0.0.0.2

$ docker tag [image id]

To create a container with specify the volume alocate
$ docker run -tid --name=mysql4 -e MYSQL_ALLOW_EMPTY_PASSWORD=True --mount source=mysql-db1,destination=/var/lib/mysql mysql

$ docker container run -d --name=nginxbind --mount type=bind,source=$(pwd),target=/app nginx

To enter my sql database
$ mysql -u root -pmypassword -h 172.17.0.3 -P 3306

To mount Local volume with docker container volume, (data will sync automatically between them)
$ docker container run -d --name=nginx2-bind -p 81:80 --mount type=bind,source="$(pwd)",target=/usr/share/nginx/html nginx

Docker Compose YML file

```
services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: mypassword
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpressuser
      MYSQL_PASSWORD: wordpress

  wordpress:
    depend-on:
```

```
    - db
  image: wordpress:latest
  ports:
    - 8080:80
  restart: always
  environment:
    WORDPRESS_DB_HOST: db:3306
    WORDPRESS_DB_USER: wordpressuser
    WORDPRESS_DB_PASSWORD: wordpress

volumes:
  db_data:
```

```
$ docker-compose up -d
```

YML script for Docker-compose

```
version: "3.1"
services:
  web:
    image: guard:latest
    ports:
      - "9999:9999"
      - 587:587
    depends_on:
      - db
    volumes:
      - /home/ubuntu/guard/logs1:/workspace/logs
    restart: always
    environment:
      SONIM_DB_HOST:
jdbc:mysql://db:3306/sonim_dev?useSSL=false&serverTimezone=UTC&useLegacyDatetimeCode=false&
createDatabaseIfNotExist=true
      SPRING_PROFILES_ACTIVE: production
      SPRING_RESET_ACTIVATION_URL: http://3.135.219.111:2906/reset-password?token=
      SPRING_CREATE_ACTIVATION_URL: http://3.135.219.111:2906/create-password?token=
      SONIM_SUPPORT_EMAIL: devops.mirafra@gmail.com
      SONIM_MAIL_HOST: smtp.gmail.com
      SONIM_MAIL_PORT: 587
      SONIM_MAIL_USERNAME: devopspramod100@gmail.com
      SONIM_MAIL_PASSWORD: vcmjyviwghbcfrmf
      SONIM_BFT_URL: http://10.16.1.212:8080/bft
      SONIM_BFT_USER: sonim_jenkins
```

```
    SONIM_BFT_PASSWORD: S0n1m@1234
  db:
    image: percona
    restart: always
    ports:
      - "3366:3306"
    volumes:
      - /home/ubuntu/guard/db1:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: sonim

  adminer:
    image: adminer
    restart: always
    ports:
      - 9998:8080
```

Bluprints@1234
Another Docker yml file:

```
version : '3'

services: #Each entry in the services section will create a separate container when docker-compose is run
  distro:
    image: alpine #Image would be download at RunTime
    restart : always #Directive is used to indicate that the container should always restart
    container_name: Custom_alpine #Directive is used to override the randomly generated container name and replace it with a name that is easier to remember and work with.
    entrypoint: tail -f /dev/null #tail -f is an ongoing process, so it will run indefinitely and prevent the container from stopping. The default entrypoint is overridden to keep the container running.
```

```
$ docker-compose -f custome-application.yml up -d
```

Docker Swarm in Cluster

Text Direction : SetUp Docker on Swarm Docker Nodes
Commands to Install Docker on Linux Machine
Please execute the all commands in sequence as they given.

```
$ sudo apt-get update
```
To Install Docker Swarm

$ docker swarm init

To create a new network
$ docker network create <network Driver> <Network Name>

To retrieve the join command including the join token , find the tokin key of swarm manager
$ docker swarm join-token worker

To Install visualizer in docker swarm
$ docker run -it -d -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock dockersamples/visualizer

To assign container to particular Node (Like :- manager , worker)
$ docker service create --name=webserver --constraint node.role==manager --replicas=3 nginx

Forcibly remove an inaccessible node from a swarm
$ docker node rm --force <node name>

Docker PROJECT

$ docker service create - -name vote -p 5000:80 - -network front_end_ntw - -replicas 5 dockersamples/examplevotingapp_vote:before

$ docker service create --name=redis --network front_end_ntw --replicas 5 redis:3.2

$ docker service create --name worker --network front_end_ntw --network back_end_ntw dockersamples/examplevotingapp_worker:latest

$ docker service create --name=db --network back_end_ntw --mount type=volume,source=db-data,target=/var/lib/postgresql/data postgres:9.4

$ docker service create --name=result --network back_end_ntw -p 5001:80 dockersamples/examplevotingapp_result:before

Jenkins Tool

Install Jenkins Server

If it's cloud machine then allow the port 8080 in firewall
$ sudo apt-get update
$ sudo apt install default-jdk

$ sudo apt-get install openjdk-8-jdk (Jenkin support only java 8)
$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
$ sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \
  /etc/apt/sources.list.d/jenkins.list'
$ sudo apt-get update
$ sudo apt-get install jenkins
$ sudo systemctl start jenkins (start the Jenkins)
$ sudo systemctl status jenkins
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword (copy the key to unlock jenkins)

To Uninstall jenkin from ubuntu

$ sudo apt-get remove --purge jenkins

Start the Jenkin Server in manual port in mac

$ java -jar /Users/pramod/Downloads/jenkins.war --httpPort=9090 (desktop manual post)

JDK file Path for Java- MAC OS

/Library/Java/JavaVirtualMachines/jdk1.8.0_211.jdk/Contents/Home/
Installation Folder
/Users/pramods/eclipse/java-2019-06

start jenkins service in MAC OS
$ sudo launchctl load /Library/LaunchDaemons/org.jenkins-ci.plist
stop jenkins service in MAC OS
$ sudo launchctl unload /Library/LaunchDaemons/org.jenkins-ci.plist

Remove jenkins logs
$ cd /var/log/jenkins
$ rm -rf jenkins.log
$ du -hs *
$ df -h

JDK file Path for Java- ubuntu

/usr/lib/jvm/java-1.8.0-openjdk-amd64

If jenkin will stop the process for full of memory then add
$ sudo fallocate -l 1G /swapfile
$ sudo chmod 600 /swapfile
$ sudo mkswap /swapfile
$ sudo swapon /swapfile

Dockerizing Jenkins in AWS server - Docker-compose.yml

```
version: '2'
services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    user: jenkins
    volumes:
      - /home/ubuntu/sonim_jenkins/jenkins/data:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    environment:
      - JENKINS_HOST_HOME="/data/jenkins"
      - JAVA_OPTS="-Djava.awt.headless=true -Djava.io.tmpdir=/var/jenkins_home/tmp"
      - TMP=/var/jenkins_home/tmp
      - TEMP=/var/jenkins_home/tmp
      - TMPDIR=/var/jenkins_home/tmp
    ports:
      - "8080:8080"
      - "50000:50000"
```

MAVEN Install in ubuntu

Login to root
$ mkdir /opt/maven
$ cd /opt/maven
$ wget http://mirrors.estointernet.in/apache/maven/maven-3/3.8.1/binaries/apache-maven-3.8.1-bin.tar.gz
$ tar -xvzf apache-maven-3.8.1-bin.tar.gz
$ cd apache-maven-3.8.1/
$ pwd
Then copy the path
$ ls -la (to verify the .bash_profile) if not then
$ vi .bash_profile
#User specific environment and startup programs
JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
M2_HOME=/opt/maven/apache-maven-3.8.1
M2=$M2_HOME/bin

PATH=$PATH:$JAVA_HOME:$M2_HOME:$M2:$HOME/bin

export PATH

To check the path
$ echo $PATH

MAven Home path - ubuntu
/opt/maven/apache-maven-3.8.1

Tomcat server install in MAC OS
Download from :- https://tomcat.apache.org/download-80.cgi
Then Extract the folder and rename it, then move to home directory
Give the full permission to bin folder
$ chmod 777 /Users/pramods/tomcat/bin/catalina.sh
$ chmod 777 /Users/pramods/tomcat/bin/startup.sh
Then Run:-
$ /Users/pramods/tomcat/bin/startup.sh
Default port will : 8080
To change the port of tomcat
Go to tomcat folder then enter conf folder then open server.xml file and edit there port number

Tomcat server install in Ubuntu
Before install check java installed or not,
$ java -version
If not the install java
$ apt-get install java-1.8*

Change the path to $ cd /opt/
$ ls
If wget not found then install wget
$ apt-get install wget
Then download
$ wget http://apachemirror.wuchna.com/tomcat/tomcat-9/v9.0.30/bin/apache-tomcat-9.0.30.tar.gz

$ ls
$ tar -zvxf apache-tomcat-9.0.30.tar.gz
$ ls
$ cd apache-tomcat-8.5.42
Then start the service
$ cd bin

Check tomcat is running or not

$ ps -ef | grep java

$ ps -ef | grep tomcat
$ echo $PATH
Set a soft link
$ ln -s /opt/apache-tomcat-8.5.43/bin/startup.sh /usr/local/bin/tomcatup
$ ln -s /opt/apache-tomcat-8.5.43/bin/shutdown.sh /usr/local/bin/tomcatdown

Tomcat Ubuntu Location
/opt/apache-tomcat-8.5.42/bin
To change the port no of Tomcat server
$ vi /opt/apache-tomcat-8.5.42/conf/server.xml
To enable manager GUI
Change the file in context.xml
$ find / -name context.xml
$ vi /opt/apache-tomcat-8.5.43/webapps/manager/META-INF/context.xml
Then comment the velu <!-- -->
$ vi /opt/apache-tomcat-8.5.43/webapps/host-manager/META-INF/context.xml
 Then comment the velu <!-- -->
Tomcat User role configure:-
Go to tomcat folder then enter conf folder then open tomcat-user.xml file and edit
$ vi /opt/apache-tomcat-8.5.43/conf/tomcat-users.xml
```
  <role rolename="manager-gui"/>
  <role rolename="manager-script"/>
  <role rolename="manager-status"/>
  <user username="admin" password="admin" roles="manager-gui, manager-script, manager-jmx,
manager-status"/>
  <user username="deployer" password="deployer" roles="manager-script"/>
  <user username="tomcat" password="tomcat@123" roles="manager-gui"/>
```

Oracle login password- Ks36tdDJmQ96RNj

Jenkin-docker shell command :-
```
sudo docker rm -f $(sudo docker ps -a -q)
sudo docker build /home/ubuntu/workspace/hello-world -t test
sudo docker run -it -p 82:80 -d test
```

Dockerfile for httpd service
```
FROM httpd
ADD ./devopsIQ /usr/local/apache2/htdocs/devopsIQ
```

172.31.10.216 netmask 255.255.240.0

Ansible Tool

Concept comand:-
Rebooting - ansible all -a “/sbin/reboot”
Copy file - ansible all -m copy -a “src=/home/dan dest=/tmp/home”
Create User - ansible all -m user -a “name=testuser password=testuser”
Remove User - ansible all -m user -a “name=testuser state=absent”
Change file permission - ansible all -m file -a “dest=/home/dan/file1.txt mode=777”
Install Package - ansible -s all -m yum -a “name=httpd state=latest”
Start a service - ansible all -m service -a “name=httpd state=started”
Stop a service - ansible all -m service -a “name=httpd state=stopped”

Installation Ansible in AWS EC2

The best way to get Ansible for Ubuntu is to add the project's PPA (personal package archive) to your system. To do this effectively, we need to install the software-properties-common package, which will give us the ability to work with PPAs easily. (This package was called python-software-properties on older versions of Ubuntu.)

Create SSH connection with Password less authentication

User login with root
$ sudo -i
$ passwd root
Enter password
$ vim /etc/ssh/sshd_config
#Auth

Installation on RHEL

$ sudo su -
$ yum update -y
$ rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

https://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/epel-release-7-11.noarch.rpm
$ yum install ansible -y
$ ansible --version

Public key:-

Installation on UBUNTU

$ sudo apt-get update
$ sudo apt-get install software-properties-common
$ sudo apt-add-repository ppa:ansible/ansible
$ sudo apt-get update
$ sudo apt-get install ansible

$ ansible --version

Configure Server with SSH connection
In server Node :-
$ sudo find / -name "id_rsa.pub"
Then, if there isnt, we generate one using the command
$ ssh-keygen -t rsa -b 4096 -C "instance-1"
For Conformation
$ sudo find / -name "id_rsa.pub"
Then edit the key
$ sudo vi /home/ubuntu/.ssh/id_rsa.pub
then copied the contents of the file to a textfile and store it

Go to client Node:-
$ sudo find / -name '*authorize*'
For my case, it is in ./.ssh/authorized_keys. And then we open that using sudo vi
$ sudo vi ./.ssh/authorized_keys
Then prest

Login as a ansadmin user on master and generate ssh key (Master)
$ ssh-keygen
$ cd ~/.ssh
$ ls
Copy keys onto all ansible client nodes (Master)
$ ssh-copy-id <target-server ip>

Edit the hosts file to enter the host ip
$ sudo vi /etc/ansible/hosts
[all_hosts]
<target-server ip>
<target-server ip>

.YML file for playbook

```
---
- hosts: 172.31.32.32
  become: true
  tasks:
  - name: copy jar/war onto tomcat servers
        copy:
          src: /opt/playbooks/webapp/target/webapp.war
          dest: /opt/apache-tomcat-8.5.42/webapps
```

CHEF Tool

1- Create chef server
Can create a host server from → https://api.chef.io

2- Setup a workstation
Download Development Kit from → https://downloads.chef.io/
Install DK in respective OS

3- Download starter kit from organization tab (from chef server login)
4- Extract zip file from download

Chef Command for Node Connect
$ knife bootstrap 18.224.95.128 ubuntu -P --identity-file /Users/pramod/ubuntuaws.pem -N ubuntu
$ knife bootstrap 18.224.95.128 --sudo -X ubuntu -i Users/pramod/ubuntuaws.pem -N ubuntu2

.htacess file

```
RewriteEngine on
RewriteCond %{HTTPS} off
RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

RewriteCond %{HTTP_HOST} !^www\.
RewriteRule ^(.*)$ https://www.%{HTTP_HOST}/$1 [R=301,L]

RewriteCond $1 !^(index\.php|resources|robots\.txt)
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php?/$1 [L,QSA]
```

Kubernetes In Ubuntu Server

Get IP address:-
$ ifconfig -a
The product_uuid can be checked by using the command

```
$ sudo cat /sys/class/dmi/id/product_uuid

Sudo apt-get update -y
sudo apt install -y apt-transport-https
Sudo su –

swapoff -a
OR:-

sudo vim /etc/fstab
#/dev/mapper/hakase--labs--vg-swap_1 none            swap    sw                  0         0


modprobe br_netfilter   (enable IP Table)
sysctl -p
sysctl net.bridge.bridge-nf-call-iptables=1

# Install Docker CE
## Set up the repository:
### Install packages to allow apt to use a repository over HTTPS
apt-get update && apt-get install apt-transport-https ca-certificates curl software-properties-common

### Add Docker’s official GPG key
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -

### Add Docker apt repository.
add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) \
  stable"
 ## Install Docker CE.
apt-get update && apt-get install docker-ce=18.06.2~ce~3-0~ubuntu
 # Setup daemon.
cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
```

```
mkdir -p /etc/systemd/system/docker.service.d

# Restart docker.
systemctl daemon-reload
systemctl restart docker

usermod -aG docker ubuntu

systemctl restart docker
apt-get install -y kubelet kubeadm kubectl Kubernetes-cni

apt-get update && apt-get install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
apt-get update
apt-get install -y kubelet kubeadm kubectl
apt-mark hold kubelet kubeadm kubectl

systemctl daemon-reload
systemctl start kubelet
systemctl enable kubelet.service
```

Kubernetes Projects: GOOGLE PRIVATE CLOUD

Create a Cluster first

Login to cluster
$ gcloud container clusters get-credentials standard-cluster-1 --zone us-central1-a

Create a docker Image for container
$ ./mvnw com.google.cloud.tools:jib-maven-plugin:build
-Dimage=gcr.io/$GOOGLE_CLOUD_PROJECT/spring-boot-example:v1

Run A pod
$ kubectl run <Name_spring-boot> --image=gcr.io/my-project-kubernetes-245411/spring-boot-example:v1
--port=8080

$ kubectl get deployment

To expose with external IP for port forwarding
$ kubectl expose deployment <Name_spring-boot> --type=LoadBalancer

Scale up the pods in deployment
$ kubectl scale deployment <Name_spring-boot> --replicas=3

Deployment from yml file
$ kubectl apply -f test.yaml

Delete the services
$ kubectl delete service <Name_spring-boot>

Create a dashboard for cluster running
$ kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-beta1/aio/deploy/recommended.yaml

kubectl create clusterrolebinding cluster-admin-binding \
--clusterrole cluster-admin \
--user jenkins-gke@my-project-kubernetes-245411.iam.gserviceaccount.com

SONARQUBE

Setup SonarQube
> Provision Instance (Minimum 1 core CPU & 2 GB RAM)
> Install MySql version, java
> Install sonarqube
  > Provision mysql RDS instance
  > Create necessary BDS & Users

IN RHL System
$ yum update
$ yum install mysql java-1.8* -y
# Download the LTS version of sonarqube
$ cd /opt/
$ wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.9.1.zip
$ unzip <sonarqube_file_name>
# Rename the sonarqube directory
$ mv <sonarqube-7.9.1> sonarqube

# user add for sonarqube
$ useradd sonaradmin
$ passwd sonaradmin
$ chown -R sonaradmin:sonaradmin sonarqube

GCP IAM role create

```
gcloud iam service-accounts add-iam-policy-binding \
  stackhives@my-project-kubernetes-245411.iam.gserviceaccount.com \
  --member=user:stackhives\
  --role=roles/iam.serviceAccountUser
```

```
kops create cluster \
--state=${KOPS_STATE_STORE} \
--node-count=2 \
--master-size=t2.micro \
--node-size=t2.micro \
--zones=us-east-1b,us-east-1c \
--name=${KOPS_CLUSTER_NAME} \
--dns private \
--master-count 1
```

```
apiVersion: v1
kind: Pod
metadata:
  name: first-app
  labels:
    app: nodeapp
spec:
  containers:
    - name: nodeapp
      image: kammana/nodeapp:v1
      ports:
        - containerPort: 8080
```

Packstack For OpenStack

https://wiki.openstack.org/wiki/Packstack

./pack build sonim-ui-application --env NODE_ENV=development --builder cloudfoundry/cnb

How to install E-Elasticsearch L-Logstah K-Kibana Stack on Ubuntu Linux

http://www.cyberkeeda.com/2020/01/how-to-install-elk-stack-on-ubuntu.html?m=1&fbclid=IwAR2PNmlgABSB_WqIVsJQgGOjvjVM3j25xqZDfmPp5hONEMqej_fRvWSpT4Y