Day 1

About our lab environment

- OnPrem Production grade Red Hat OpenShift setup
- System Configuration
    - 48 virtual cores
    - 755 GB RAM
    - 17 TB HDD Storage
- Oracle Linux v9.4 64-bit OS
- KVM Hypervisor
- 7 Virtual machines
    - Master 1 with RHEL Core OS (8 Cores, 128GB RAM, 500 GB HDD)
    - Master 2 with RHEL Core OS (8 Cores, 128GB RAM, 500 GB HDD)
    - Master 3 with RHEL Core OS (8 Cores, 128GB RAM, 500 GB HDD)
    - Worker 1 with RHEL Core OS (8 Cores, 128GB RAM, 500 GB HDD)
    - Worker 2 with RHEL Core OS (8 Cores, 128GB RAM, 500 GB HDD)
    - HAProxy Load Balancer, Bind DNS - Bastion Virtual Machine
    - BootStrap Virtual Machine
        - installs control plane components in master nodes
        - creates an etcd cluster configured etcd pod instances from respective master nodes
        - configures the HAProxy Load Balancer to act as a load-balancer the master nodes
        - helps in installing worker node components and join to the worker nodes to the openshift cluster

What is dual-booting or multi-booting?

- Let's say we have a laptop with Windows 10 pre-installed and we need to do some prototype in Ubuntu
- You can use a system utility called Boot Loader like LILO, Grub 1, Grub2
- When we install boot loaders, it gets installed in your hard disk Master Boot Record(MBR) - Sector 0, Byte 0 in your hard disk (512 bytes)
- When we boot our machine, BIOS POST (Power On Self Test), once the BIOS is loaded, BIOS will initialize all hardwares and then it then it instructs the CPU to load and run the Boot loader application from MBR
- The boot application starts scanning for Operating Systems installed on your Hard disk(s), if it finds more than one OS then it gives a menu for you to choose which OS you wish to boot into
- Only one OS can be active at any point of time Examples
- LILO
- GRUB
- BootCamp

Hypervisor Overview

```
- is hardware virtualization technology
- this helps us run multiple Operating System on a laptop/desktop/workstation/server
- i.e more than one OS can be actively running on the same machine
- this type of virtualization is considered heavy-weight as each Virtual Machine(VM - Guest OS) requires dedicated hardware resources
  - CPU Cores
  - RAM
  - Storage ( HDD/SDD )
- each VM represents a fully functional Operating System
- the OS that runs within Virtual Machine is referred as Guest OS
- each Guest OS has its own dedicated OS Kernel
- there are two of Hypervisors
  1. Type 1
  - Bare-Metal Hypervisors
  - Used in Servers & Workstations
  - Virtual Machines(Guest OS) can be created directly on top of hardware with no Host OS requirement
  - Examples
    - VMWare vSphere/vCenter
  2. Type 2
  - used in Laptops/Desktops/Workstations
  - Examples
    - Oracle VirtualBox ( supported in Windows/Linux/Mac OS-X )
    - Parallels ( supported in Mac OS-X )
    - KVM ( supported in all Linux Distributions )
    - VMWare Fusion ( supported in Mac OS-X - Free, earlier it was a paid software, not actively maintained anymore )
    - VMWare Workstation Pro ( was a paid software but now it is Free, works in Linux & Windows )
    - Microsoft Hyper-V
```

Linux Kernel Container Features

1. Namespace

   - helps in isolating one container from other containers running on the same OS

2. Control Group(CGroups)

   - to ensure every container shares the hardware resources co-operatively Control Groups are used
   - if this isn't not done, at times certain containers uses all the hardwares resources leaving other containers to starve
   - helps in applying resource quota restrictions like
     - we can restrict a container on how many CPU Cores it can use at the max at any point of time
     - we can restrict how much RAM a container use at the max
     - we can restrict, how much storage a container can use at the max

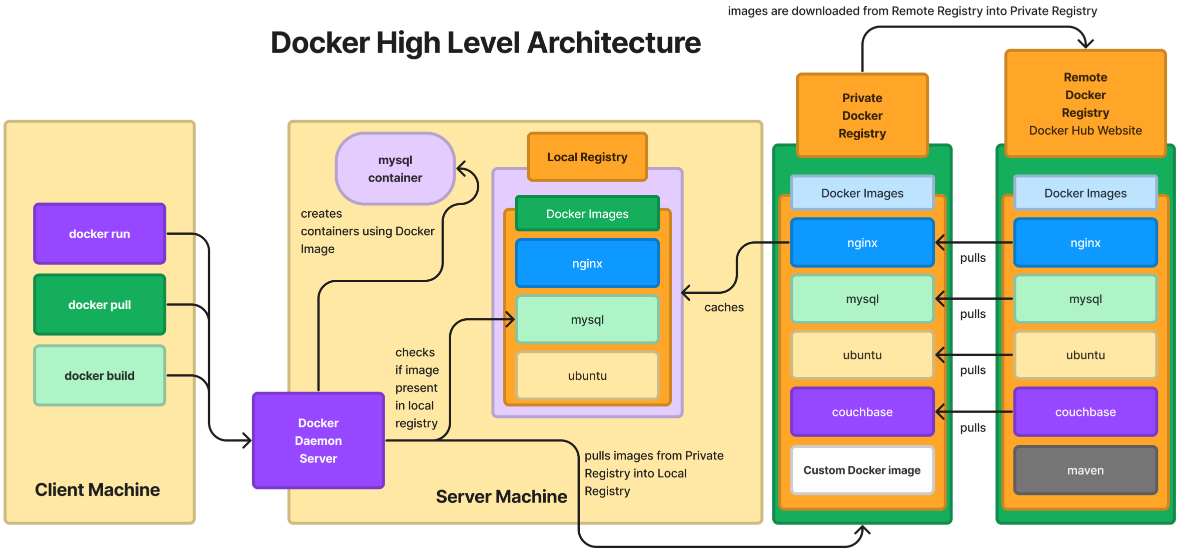
Container Overview

```
- is an application virtualization technology
- each container represents one application
- containers can't run a OS inside it, they can only run a single
application
- hence, containers are not a replacement for Hypervisor(Virtualization)
- in practical world, Virtualization and Containerization are used in
combination, hence they are complimenting technology and not a competing
technology
- is considered light-weight virtualization
- containers running in the Host OS, shares the hardware resources on the
underlying Host OS
- containers don't get their own hardware resources
- contianers does'nt have OS Kernel
- containers depend on the Host OS Kernel for any OS functionality
```

Docker Overview

```
- Docker is a Container Engine
- Developed in Golang by a company called Docker Inc
- comes in 2 flavours
  1. Docker Community Edition - Docker CE ( Free )
  2. Docker Enterprise Edition - Docker EE ( Paid )
- follows Client/Server Architecture
- Docker Registry
  - collection of many Docker Images
- Supports 3 types of Docker Registries
  1. Local Docker Registry
  2. Private Docker Registry
    - setup using JFrog Artifactory or Sonatype Nexus
  3. Remote Docker Registry
    - website maintained by Docker Inc
    - provides many opensource docker images
```

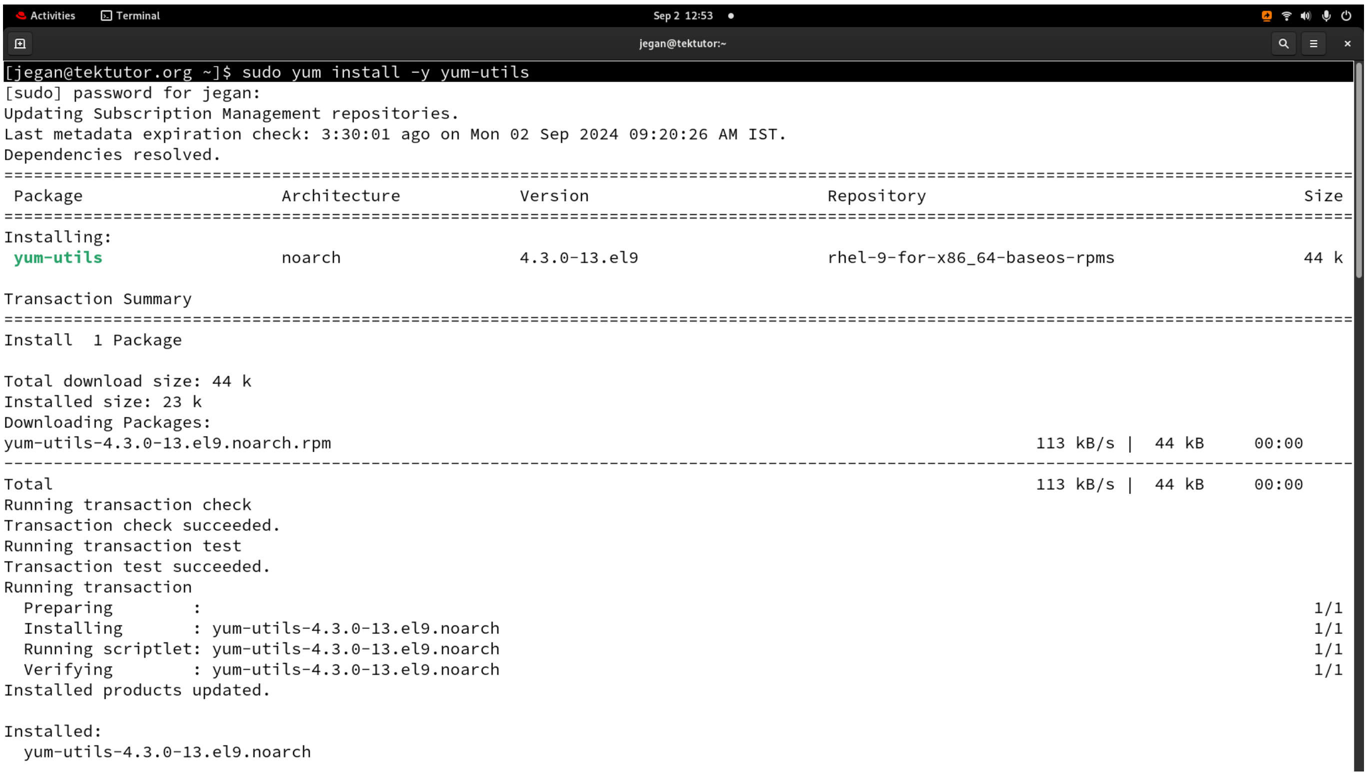
High-Level Docker Architecture



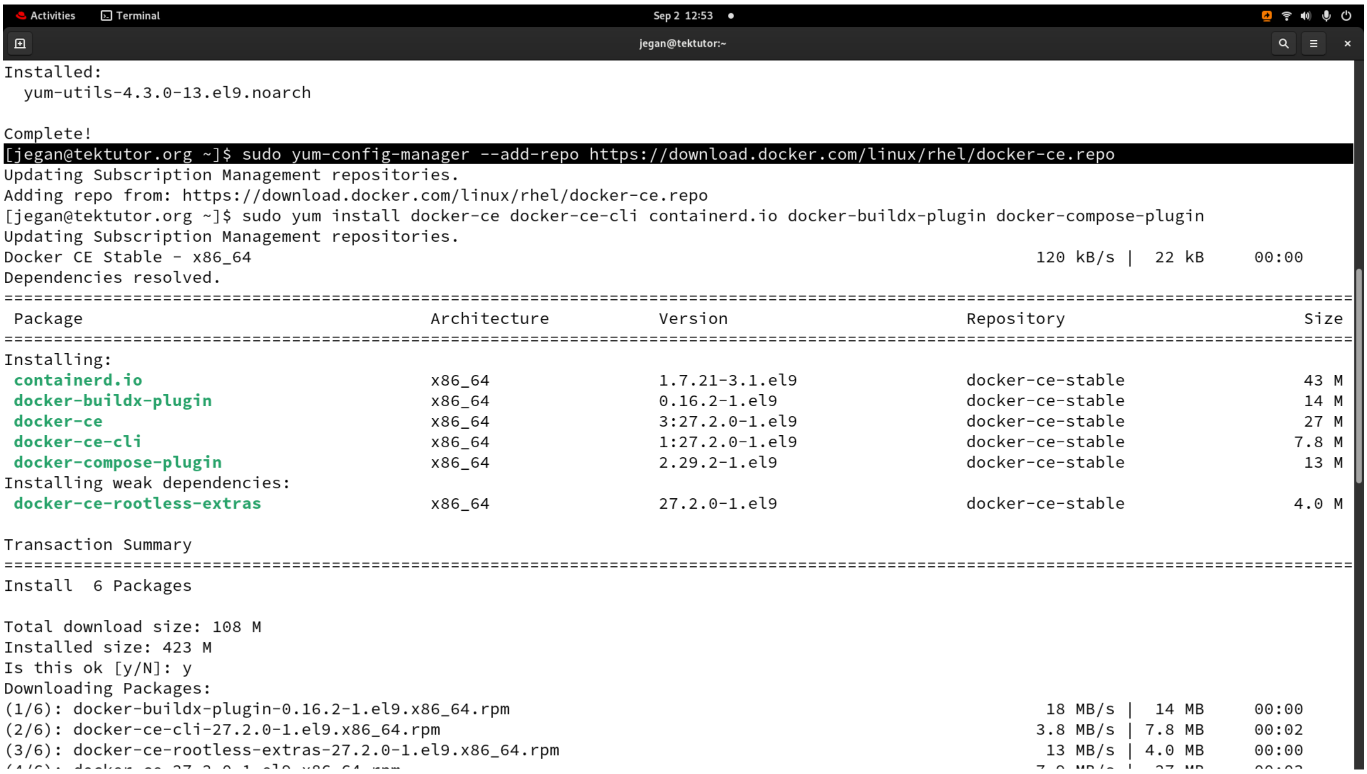
Demo - Installing Docker Community edition in RHEL or Oracle Linux

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo
https://download.docker.com/linux/rhel/docker-ce.repo
sudo yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
sudo usermod -aG docker $USER
sudo systemctl enable docker
sudo systemctl start docker
sudo systemctl status docker
docker --version
docker info
docker images
```

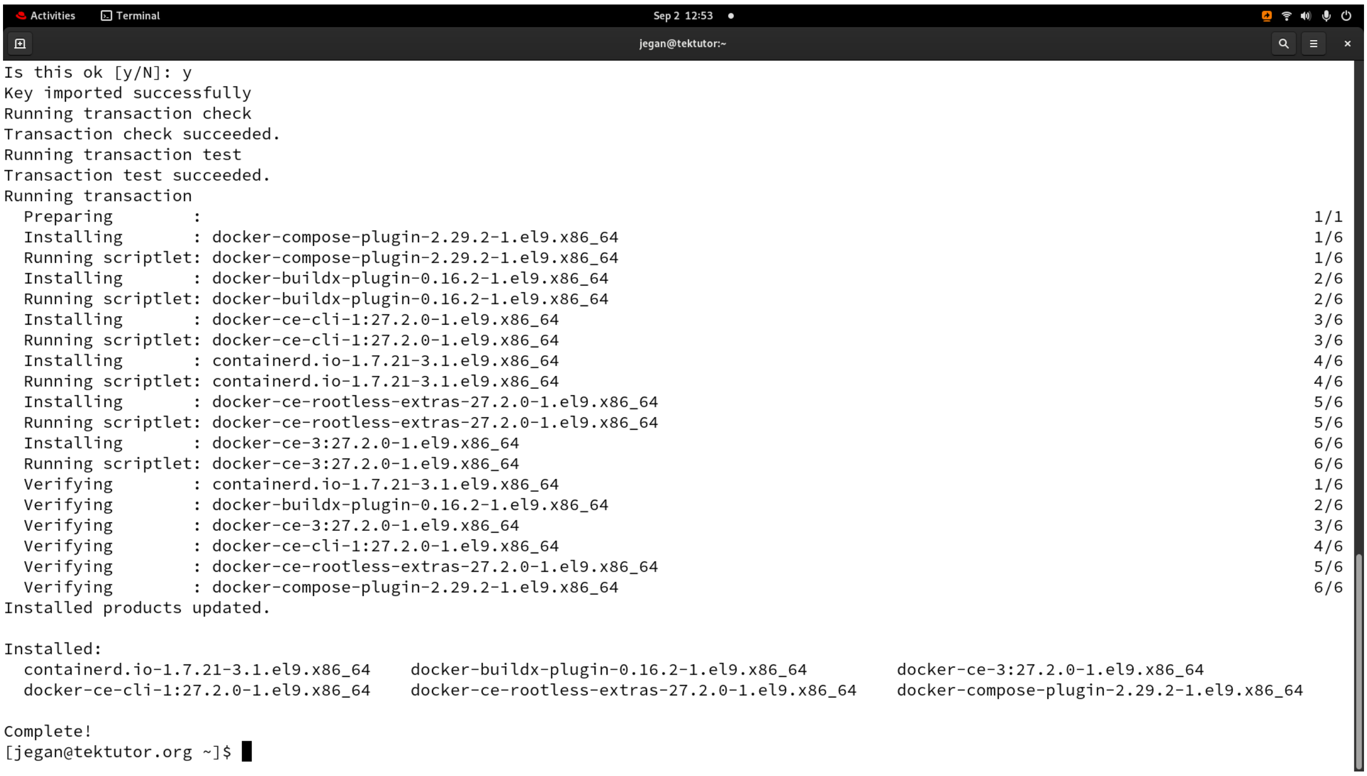
Expected output



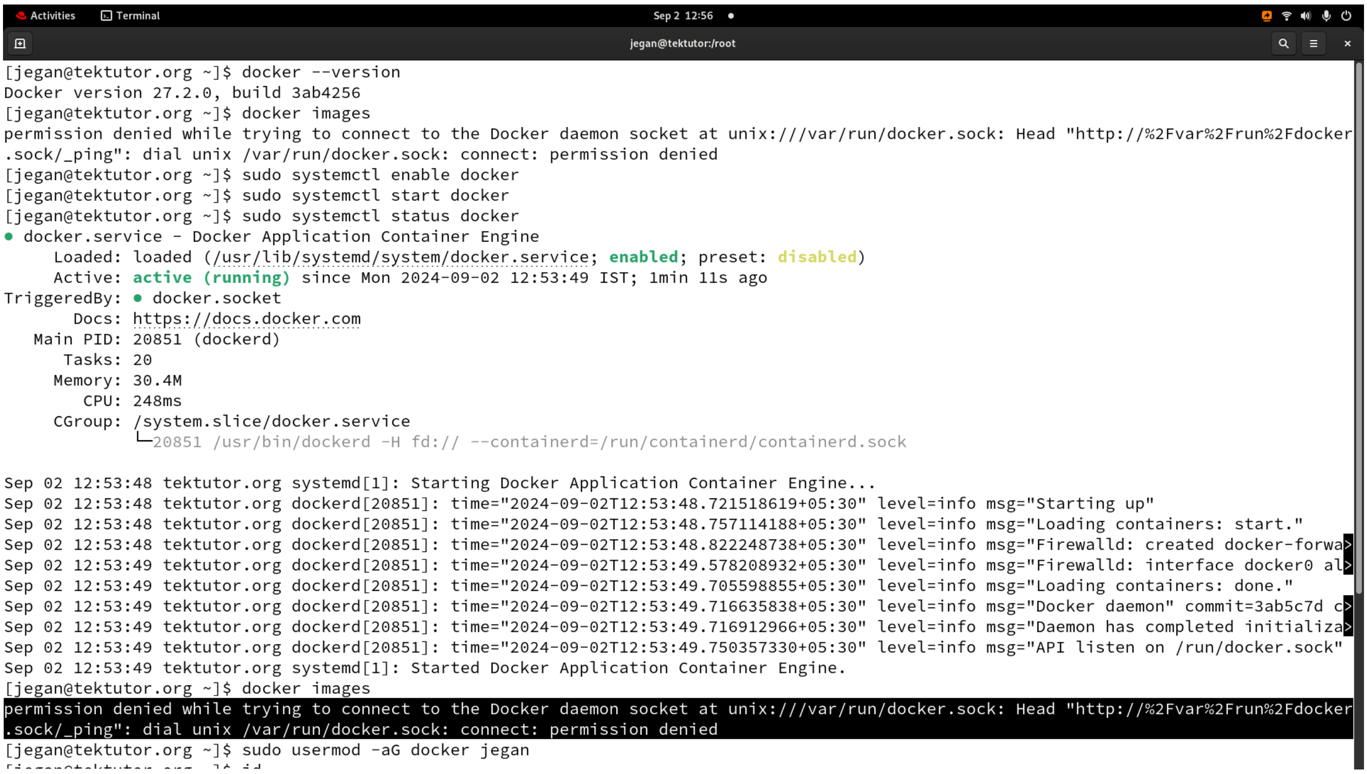
```
[jegan@tektutor.org ~]$ sudo yum install -y yum-utils
[sudo] password for jegan:
Updating Subscription Management repositories.
Last metadata expiration check: 3:30:01 ago on Mon 02 Sep 2024 09:20:26 AM IST.
Dependencies resolved.
========================================================================================================================================
 Package                    Architecture            Version                         Repository                                   Size
========================================================================================================================================
Installing:
 yum-utils                  noarch                  4.3.0-13.el9                    rhel-9-for-x86_64-baseos-rpms                44 k

Transaction Summary
========================================================================================================================================
Install  1 Package

Total download size: 44 k
Installed size: 23 k
Downloading Packages:
yum-utils-4.3.0-13.el9.noarch.rpm                                                                         113 kB/s |  44 kB     00:00
----------------------------------------------------------------------------------------------------------------------------------------
Total                                                                                                     113 kB/s |  44 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                                1/1
  Installing       : yum-utils-4.3.0-13.el9.noarch                                                                                  1/1
  Running scriptlet: yum-utils-4.3.0-13.el9.noarch                                                                                  1/1
  Verifying        : yum-utils-4.3.0-13.el9.noarch                                                                                  1/1
Installed products updated.

Installed:
  yum-utils-4.3.0-13.el9.noarch
```



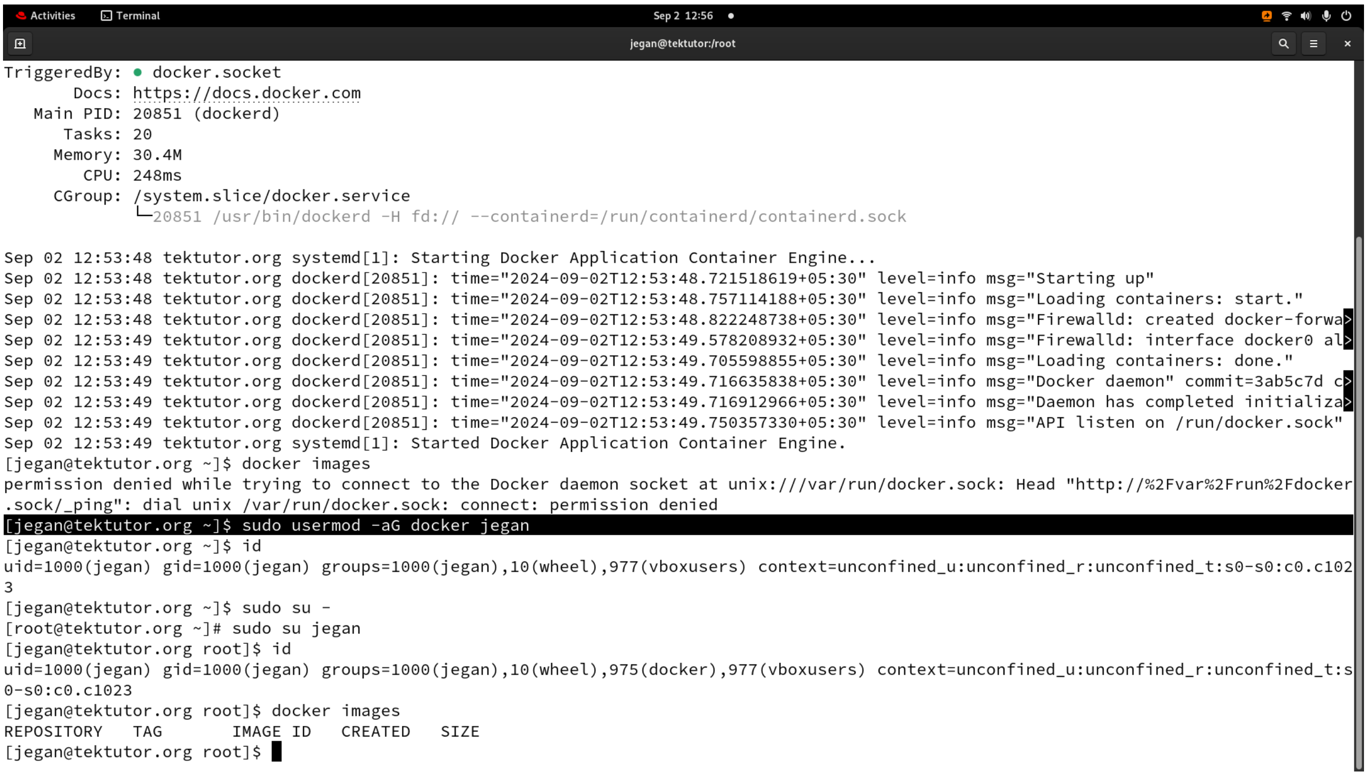
```
Installed:
  yum-utils-4.3.0-13.el9.noarch

Complete!
[jegan@tektutor.org ~]$ sudo yum-config-manager --add-repo https://download.docker.com/linux/rhel/docker-ce.repo
Updating Subscription Management repositories.
Adding repo from: https://download.docker.com/linux/rhel/docker-ce.repo
[jegan@tektutor.org ~]$ sudo yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Updating Subscription Management repositories.
Docker CE Stable - x86_64                                                                                 120 kB/s |  22 kB     00:00
Dependencies resolved.
========================================================================================================================================
 Package                                   Architecture            Version                         Repository                    Size
========================================================================================================================================
Installing:
 containerd.io                             x86_64                  1.7.21-3.1.el9                  docker-ce-stable              43 M
 docker-buildx-plugin                      x86_64                  0.16.2-1.el9                    docker-ce-stable              14 M
 docker-ce                                 x86_64                  3:27.2.0-1.el9                  docker-ce-stable              27 M
 docker-ce-cli                             x86_64                  1:27.2.0-1.el9                  docker-ce-stable             7.8 M
 docker-compose-plugin                     x86_64                  2.29.2-1.el9                    docker-ce-stable              13 M
Installing weak dependencies:
 docker-ce-rootless-extras                 x86_64                  27.2.0-1.el9                    docker-ce-stable             4.0 M

Transaction Summary
========================================================================================================================================
Install  6 Packages

Total download size: 108 M
Installed size: 423 M
Is this ok [y/N]: y
Downloading Packages:
(1/6): docker-buildx-plugin-0.16.2-1.el9.x86_64.rpm                                                        18 MB/s |  14 MB     00:00
(2/6): docker-ce-cli-27.2.0-1.el9.x86_64.rpm                                                              3.8 MB/s | 7.8 MB     00:02
(3/6): docker-ce-rootless-extras-27.2.0-1.el9.x86_64.rpm                                                   13 MB/s | 4.0 MB     00:00
(4/6): docker-ce-27.2.0-1.el9.x86_64.rpm                                                                  7.0 MB/s |  27 MB     00:03
```



```
Is this ok [y/N]: y
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                                   1/1
  Installing       : docker-compose-plugin-2.29.2-1.el9.x86_64                                                                         1/6
  Running scriptlet: docker-compose-plugin-2.29.2-1.el9.x86_64                                                                         1/6
  Installing       : docker-buildx-plugin-0.16.2-1.el9.x86_64                                                                          2/6
  Running scriptlet: docker-buildx-plugin-0.16.2-1.el9.x86_64                                                                          2/6
  Installing       : docker-ce-cli-1:27.2.0-1.el9.x86_64                                                                               3/6
  Running scriptlet: docker-ce-cli-1:27.2.0-1.el9.x86_64                                                                               3/6
  Installing       : containerd.io-1.7.21-3.1.el9.x86_64                                                                               4/6
  Running scriptlet: containerd.io-1.7.21-3.1.el9.x86_64                                                                               4/6
  Installing       : docker-ce-rootless-extras-27.2.0-1.el9.x86_64                                                                     5/6
  Running scriptlet: docker-ce-rootless-extras-27.2.0-1.el9.x86_64                                                                     5/6
  Installing       : docker-ce-3:27.2.0-1.el9.x86_64                                                                                   6/6
  Running scriptlet: docker-ce-3:27.2.0-1.el9.x86_64                                                                                   6/6
  Verifying        : containerd.io-1.7.21-3.1.el9.x86_64                                                                               1/6
  Verifying        : docker-buildx-plugin-0.16.2-1.el9.x86_64                                                                          2/6
  Verifying        : docker-ce-3:27.2.0-1.el9.x86_64                                                                                   3/6
  Verifying        : docker-ce-cli-1:27.2.0-1.el9.x86_64                                                                               4/6
  Verifying        : docker-ce-rootless-extras-27.2.0-1.el9.x86_64                                                                     5/6
  Verifying        : docker-compose-plugin-2.29.2-1.el9.x86_64                                                                         6/6
Installed products updated.

Installed:
  containerd.io-1.7.21-3.1.el9.x86_64    docker-buildx-plugin-0.16.2-1.el9.x86_64         docker-ce-3:27.2.0-1.el9.x86_64
  docker-ce-cli-1:27.2.0-1.el9.x86_64    docker-ce-rootless-extras-27.2.0-1.el9.x86_64    docker-compose-plugin-2.29.2-1.el9.x86_64

Complete!
[jegan@tektutor.org ~]$ 
```



```
[jegan@tektutor.org ~]$ docker --version
Docker version 27.2.0, build 3ab4256
[jegan@tektutor.org ~]$ docker images
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http://%2Fvar%2Frun%2Fdocker
.sock/_ping": dial unix /var/run/docker.sock: connect: permission denied
[jegan@tektutor.org ~]$ sudo systemctl enable docker
[jegan@tektutor.org ~]$ sudo systemctl start docker
[jegan@tektutor.org ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: disabled)
     Active: active (running) since Mon 2024-09-02 12:53:49 IST; 1min 11s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 20851 (dockerd)
      Tasks: 20
     Memory: 30.4M
        CPU: 248ms
     CGroup: /system.slice/docker.service
             └─20851 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Sep 02 12:53:48 tektutor.org systemd[1]: Starting Docker Application Container Engine...
Sep 02 12:53:48 tektutor.org dockerd[20851]: time="2024-09-02T12:53:48.721518619+05:30" level=info msg="Starting up"
Sep 02 12:53:48 tektutor.org dockerd[20851]: time="2024-09-02T12:53:48.757114188+05:30" level=info msg="Loading containers: start."
Sep 02 12:53:48 tektutor.org dockerd[20851]: time="2024-09-02T12:53:48.822248738+05:30" level=info msg="Firewalld: created docker-forwa>
Sep 02 12:53:49 tektutor.org dockerd[20851]: time="2024-09-02T12:53:49.578208932+05:30" level=info msg="Firewalld: interface docker0 al>
Sep 02 12:53:49 tektutor.org dockerd[20851]: time="2024-09-02T12:53:49.705598855+05:30" level=info msg="Loading containers: done."
Sep 02 12:53:49 tektutor.org dockerd[20851]: time="2024-09-02T12:53:49.716635838+05:30" level=info msg="Docker daemon" commit=3ab5c7d c>
Sep 02 12:53:49 tektutor.org dockerd[20851]: time="2024-09-02T12:53:49.716912966+05:30" level=info msg="Daemon has completed initializa>
Sep 02 12:53:49 tektutor.org dockerd[20851]: time="2024-09-02T12:53:49.750357330+05:30" level=info msg="API listen on /run/docker.sock"
Sep 02 12:53:49 tektutor.org systemd[1]: Started Docker Application Container Engine.
[jegan@tektutor.org ~]$ docker images
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http://%2Fvar%2Frun%2Fdocker
.sock/_ping": dial unix /var/run/docker.sock: connect: permission denied
[jegan@tektutor.org ~]$ sudo usermod -aG docker jegan
[jegan@tektutor.org ~]$ id
```

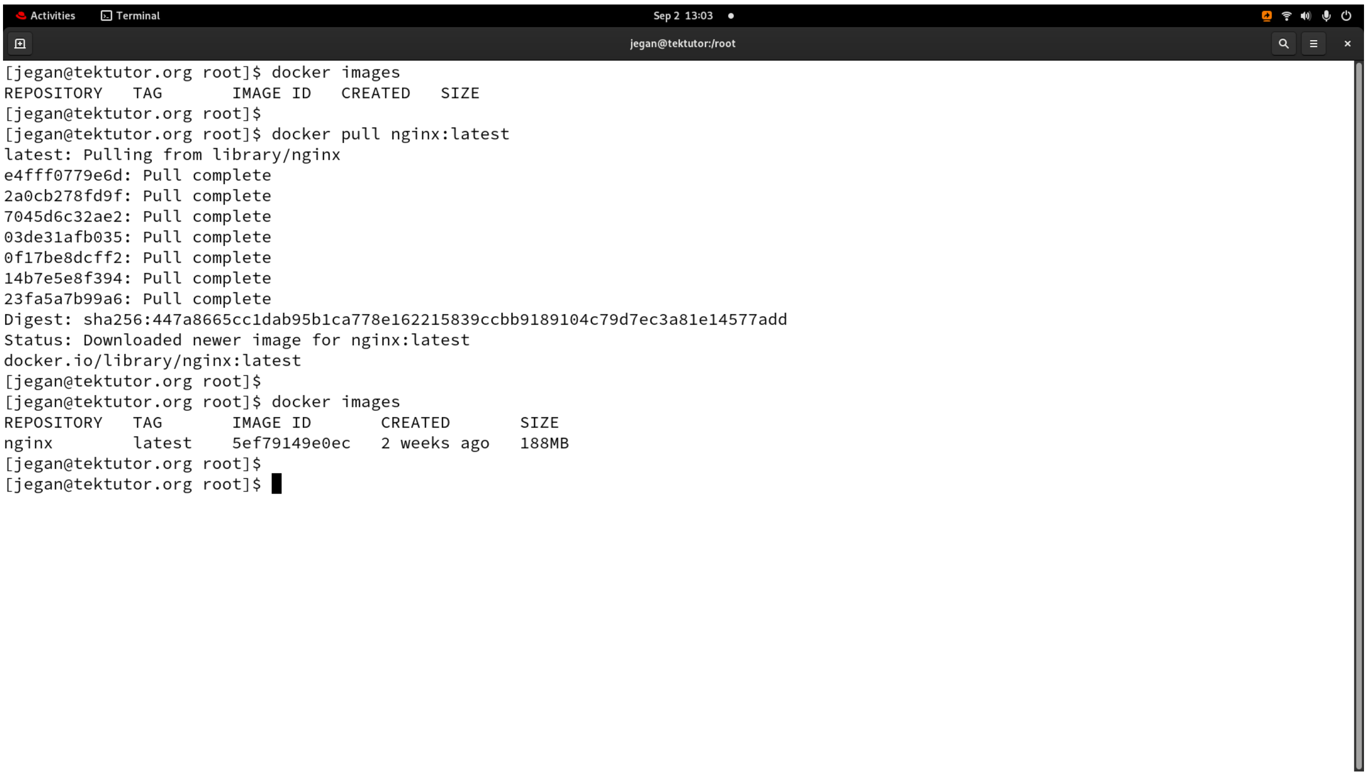


```
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 20851 (dockerd)
      Tasks: 20
     Memory: 30.4M
        CPU: 248ms
     CGroup: /system.slice/docker.service
             └─20851 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Sep 02 12:53:48 tektutor.org systemd[1]: Starting Docker Application Container Engine...
Sep 02 12:53:48 tektutor.org dockerd[20851]: time="2024-09-02T12:53:48.721518619+05:30" level=info msg="Starting up"
Sep 02 12:53:48 tektutor.org dockerd[20851]: time="2024-09-02T12:53:48.757114188+05:30" level=info msg="Loading containers: start."
Sep 02 12:53:48 tektutor.org dockerd[20851]: time="2024-09-02T12:53:48.822248738+05:30" level=info msg="Firewalld: created docker-forwa>
Sep 02 12:53:49 tektutor.org dockerd[20851]: time="2024-09-02T12:53:49.578208932+05:30" level=info msg="Firewalld: interface docker0 al>
Sep 02 12:53:49 tektutor.org dockerd[20851]: time="2024-09-02T12:53:49.705598855+05:30" level=info msg="Loading containers: done."
Sep 02 12:53:49 tektutor.org dockerd[20851]: time="2024-09-02T12:53:49.716635838+05:30" level=info msg="Docker daemon" commit=3ab5c7d c>
Sep 02 12:53:49 tektutor.org dockerd[20851]: time="2024-09-02T12:53:49.716912966+05:30" level=info msg="Daemon has completed initializa>
Sep 02 12:53:49 tektutor.org dockerd[20851]: time="2024-09-02T12:53:49.750357330+05:30" level=info msg="API listen on /run/docker.sock"
Sep 02 12:53:49 tektutor.org systemd[1]: Started Docker Application Container Engine.
[jegan@tektutor.org ~]$ docker images
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http://%2Fvar%2Frun%2Fdocker
.sock/_ping": dial unix /var/run/docker.sock: connect: permission denied
[jegan@tektutor.org ~]$ sudo usermod -aG docker jegan
[jegan@tektutor.org ~]$ id
uid=1000(jegan) gid=1000(jegan) groups=1000(jegan),10(wheel),977(vboxusers) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c102
3
[jegan@tektutor.org ~]$ sudo su -
[root@tektutor.org ~]# sudo su jegan
[jegan@tektutor.org root]$ id
uid=1000(jegan) gid=1000(jegan) groups=1000(jegan),10(wheel),975(docker),977(vboxusers) context=unconfined_u:unconfined_r:unconfined_t:s
0-s0:c0.c1023
[jegan@tektutor.org root]$ docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
[jegan@tektutor.org root]$ 
```

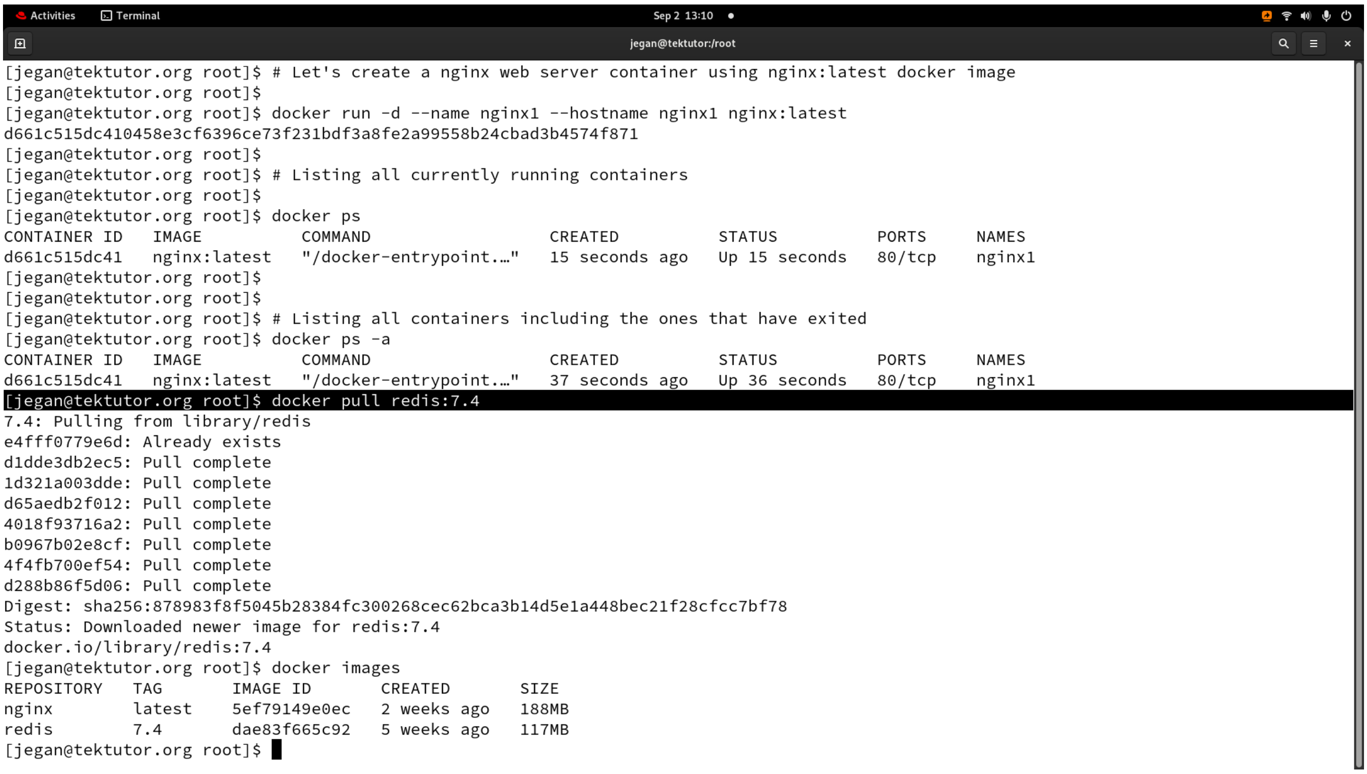
Lab - Download a docker image from Docker Remote Registry to Local Docker Registry

```
docker images
docker pull nginx:latest
docker images
docker pull redis:7.4
docker images
```

Expected output



```
[jegan@tektutor.org root]$ docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$ docker pull nginx:latest
latest: Pulling from library/nginx
e4fff0779e6d: Pull complete
2a0cb278fd9f: Pull complete
7045d6c32ae2: Pull complete
03de31afb035: Pull complete
0f17be8dcff2: Pull complete
14b7e5e8f394: Pull complete
23fa5a7b99a6: Pull complete
Digest: sha256:447a8665cc1dab95b1ca778e162215839ccbb9189104c79d7ec3a81e14577add
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
nginx        latest    5ef79149e0ec   2 weeks ago   188MB
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$
```

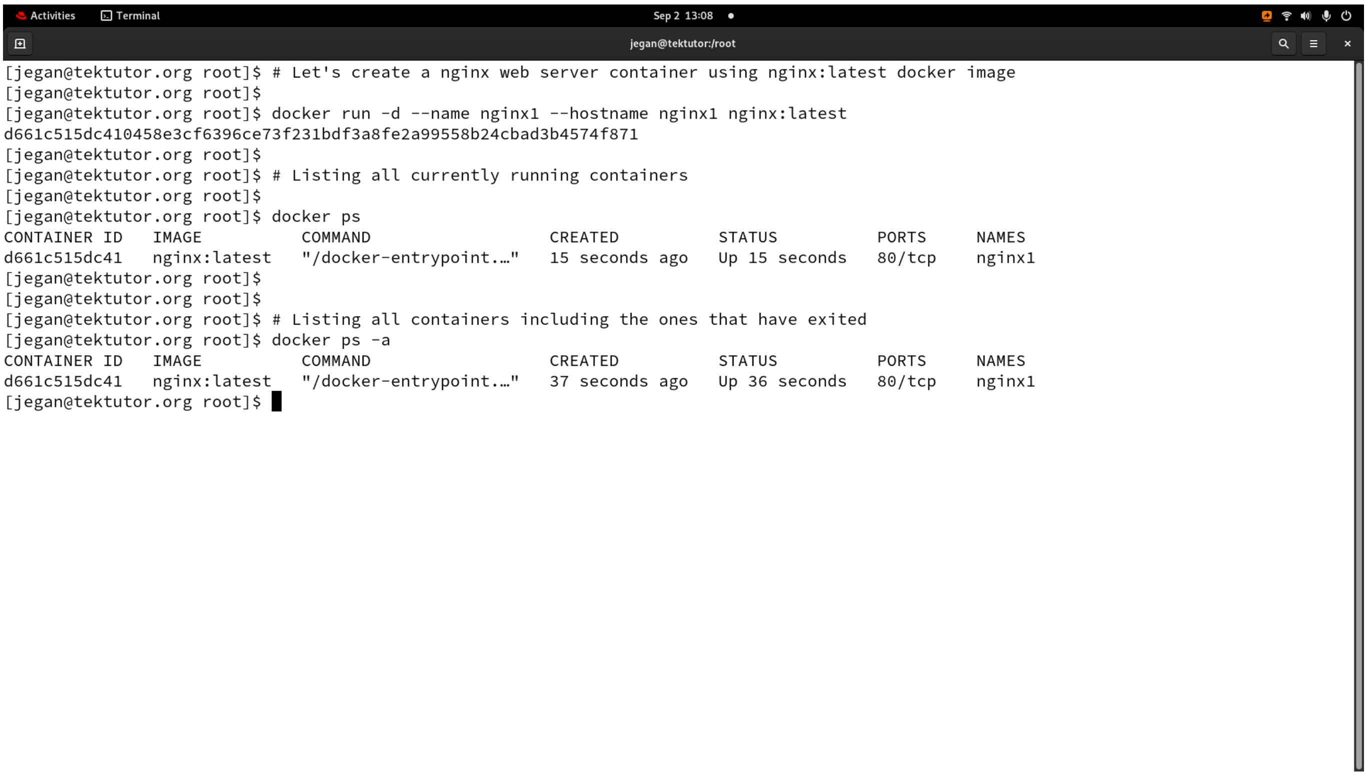


```
[jegan@tektutor.org root]$ # Let's create a nginx web server container using nginx:latest docker image
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$ docker run -d --name nginx1 --hostname nginx1 nginx:latest
d661c515dc410458e3cf6396ce73f231bdf3a8fe2a99558b24cbad3b4574f871
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$ # Listing all currently running containers
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
d661c515dc41   nginx:latest   "/docker-entrypoint.…"   15 seconds ago   Up 15 seconds   80/tcp    nginx1
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$ # Listing all containers including the ones that have exited
[jegan@tektutor.org root]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
d661c515dc41   nginx:latest   "/docker-entrypoint.…"   37 seconds ago   Up 36 seconds   80/tcp    nginx1
[jegan@tektutor.org root]$ docker pull redis:7.4
7.4: Pulling from library/redis
e4fff0779e6d: Already exists
d1dde3db2ec5: Pull complete
1d321a003dde: Pull complete
d65aedb2f012: Pull complete
4018f93716a2: Pull complete
b0967b02e8cf: Pull complete
4f4fb700ef54: Pull complete
d288b86f5d06: Pull complete
Digest: sha256:878983f8f5045b28384fc300268cec62bca3b14d5e1a448bec21f28cfcc7bf78
Status: Downloaded newer image for redis:7.4
docker.io/library/redis:7.4
[jegan@tektutor.org root]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
nginx        latest    5ef79149e0ec   2 weeks ago   188MB
redis        7.4       dae83f665c92   5 weeks ago   117MB
[jegan@tektutor.org root]$
```

Lab - Creating an nginx container

```
docker run -d --name nginx1 --hostname nginx1 nginx:latest
docker ps
docker ps -a
```

Expected output



```
[jegan@tektutor.org root]$ # Let's create a nginx web server container using nginx:latest docker image
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$ docker run -d --name nginx1 --hostname nginx1 nginx:latest
d661c515dc410458e3cf6396ce73f231bdf3a8fe2a99558b24cbad3b4574f871
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$ # Listing all currently running containers
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
d661c515dc41   nginx:latest   "/docker-entrypoint.…"   15 seconds ago   Up 15 seconds   80/tcp    nginx1
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$
[jegan@tektutor.org root]$ # Listing all containers including the ones that have exited
[jegan@tektutor.org root]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
d661c515dc41   nginx:latest   "/docker-entrypoint.…"   37 seconds ago   Up 36 seconds   80/tcp    nginx1
[jegan@tektutor.org root]$ 
```

Container Image Overview

```
- Container image is template or specification or blueprint of a container
- It is similar to Windows-12-OS-DVD-image.iso
- Just like how we are able to use windows DVD iso image and install
Windows 12 OS on multiple machine, on the similar fashion, using a
Container Image one can create multiple containers
- Containers are served by Container Registries
- Examples
  - Ubuntu Container Image ( ubuntu:24.04 )
  - Nginx Container Image ( nginx:latest )
```

Container Registry Overview

```
- Docker supports 3 types of Container Registries
  1. Local Docker Registry
  2. Private Docker Registry and
  3. Remote Docker Registry ( Docker Hub Website -  hub.docker.com )
- Local Docker Registry is just a folder, where all the docker images are
stored
- Private Docker Registry is server that can be setup using JFrog
Artifactory or Sonatype Nexus
- Remote Docker Registry is a website that has a whole lot of Docker Images
```

Containers Overview

```
- Container is an instance of one Container Image
- Whatever software tools are present in the Container Image are readily available for use in any container intance
- containers gets its own Private IP address
- containers get its own file system
- containers supports one or more shells
- containers has its own network stack ( 7 OSI Layers )
- containers has its own software defined network cards ( Network Interface Card - NICs )
- each container represents one instance of an application
- containers are not OS
- some containers may appear like a OS, but technically they are just application process that runs in a separate namespace
```

Info - Container Runtime

```
- is a low-level software that is used to manage Containers and Images
- it is not so user-friendly, hence end-users generally don't use this directly
- examples
  - runC
  - CRI-O
```

Info - Container Engines

```
- is a high-level software that is used to manage containers and images
- it internally uses Container Runtime to manage containers and images
- it is very user-friendly
- examples
  - Docker
  - Podman
  - containerd
```

Container Orchestration Overview

```
- Container Orchestration platform tools helps us manage our containerized application workloads
- instead of manually creating containers and managing them, we could use Container Orchestration Platforms to manage them for us
- Examples
  1. Docker SWARM - Docker's native Container Orchestration Platform
  2. Google Kubernetes - Free & Production grade Container Orchestration Platform
  3. Red Hat Openshift - Enterprise Paid software - Red Hat's Distribution of Kubernetes
```

```
- General features supported
  - provides an environment to make your containerized application works Highly Available (HA)
  - based on traffic to one's application individual applications can be scaled up/down on demand manually/automatically
  - upgrading/downgrading your already live applications from one version to the other without any down time using Rolling update
  - exposing your applications within cluster or to external world using services
- it is a self-healing platform
  - it can repair itself to some extent
  - it can repair your application when it is found faulty
```

Info - Docker SWARM Overview

```
- this is Docker Inc's Container Orchestration Platform
- it only supports managing Docker containerized application workloads
- it is pretty easy to install and learn
- can be installed on a laptop with pretty basic configuation as well as it is very light weight
- good for POC or learning purpose
- not production grade
```

Info - Google Kubernetes Overview

```
- free and opensource container orchestration platform developed by Google along with many open source contributors
- it is production grade
- free for personal and commercial use
- as it is opensource, we won't get support from Google
- only supports command line interface (CLI)
- doesn't support web console
- Kubernetes does provide some basic Dashboard but it is considered a security vulnerability, hence no one uses the Kubernetes Dashboard
- Rancher is opensource webconsole for Google Kubernetes
- Kubernetes supports RBAC, Custom Resource Definition (CRD) to extend the Kubernetes API to add new type of resources
- Kubernetes also allows us to develop and deploy custom operators to manage our custom resources
- Kubernetes operators - is a combination of many Custom Resources + Customer Controllers
- supports inbuilt monitoring features
  - it can check the health of our application, when it finds your application is not responding, it can repair it or replace it with another good healthy instance of your application
  - it supports inbuilt load-balancing
- supports any container runtime/engine as long as there is an implemention of CRI(Container Runtime Interface) to interact with those runtime/engine
```

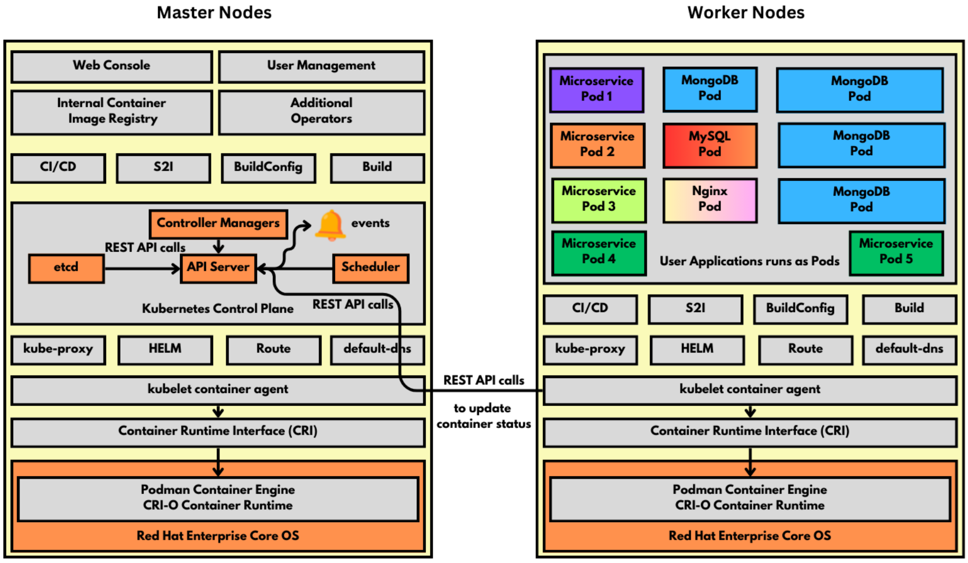
Info - OKD

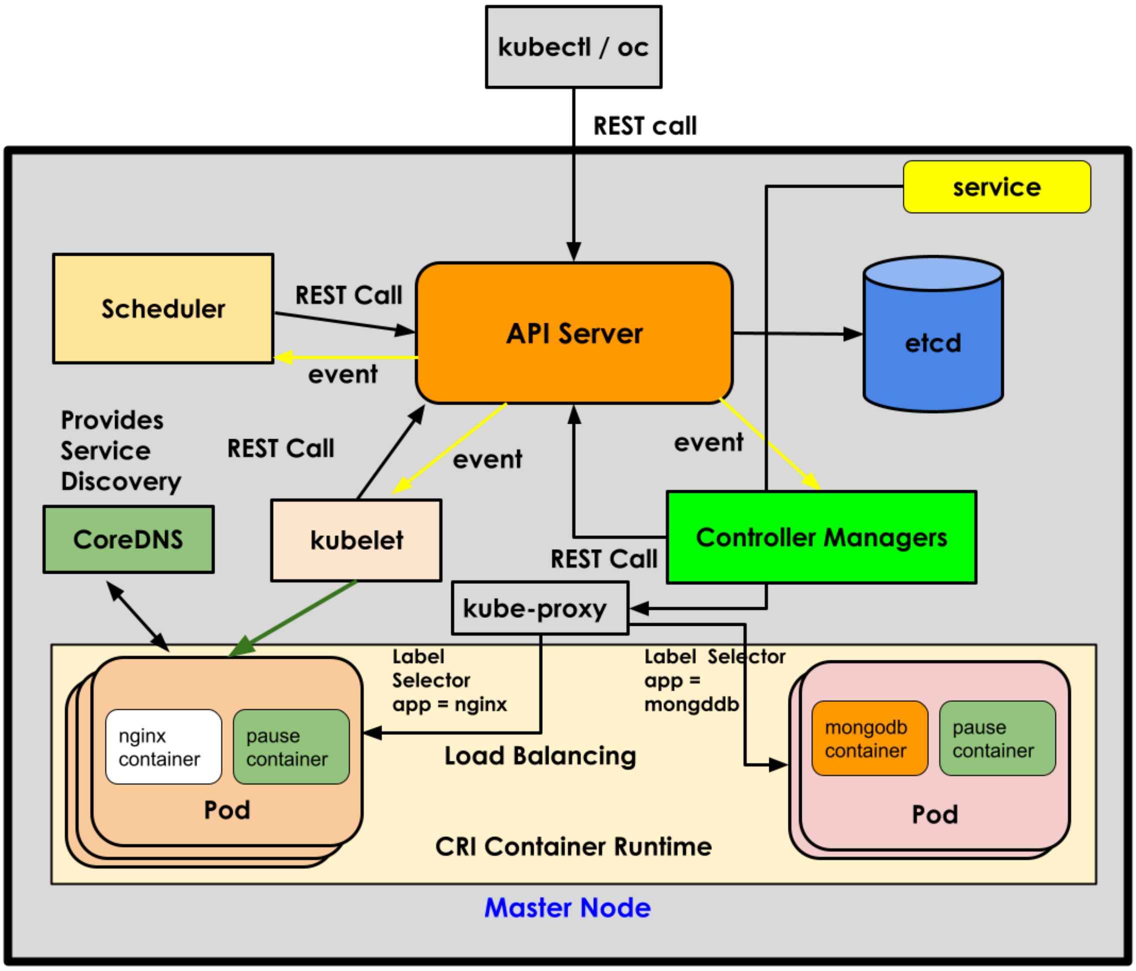
```
- is opensource Container Orchestration Platform maintained by opensource
community
- is developed on top of Google Kubernetes
- supports both CLI and webconsole
- comes with in-built internal Image Registry
- you only get community support
- supports user-management
- supports deploying application from source code is called S2I (Source to
Image)
- supports CI/CD
- support Virtualization
```

Red Hat Openshift Overview

```
- Red Hat Openshift is developed on top of Opeensource OKD ( which in turn
is developed on top of opensource Kubernetes )
- supports command line interface and webconsole(GUI)
- supports Role based access control (RBAC), hence multiple users can be
created with different level of access to Openshift cluster
- supports many additional features on top of all the Kubernetes features
  1. User Management
  2. Pre-integrated monitoring tools ( Prometheus & Grafana Dashboards )
  3. Out of the box - Private Container Registry
  4. Routes - a new feature only available in Openshift which is developed
on top of Kubernetes Ingress
  5. Using the Kubernetes operators, the Red Hat Openshift team has many
additional features on top of Kubernetes
- Openshift version upto 3 - supported many different container
runtime/engines including Docker
- Openshift version 4 and above - only supports Podman Container Engine and
CRI-O Container Runtime
- Due to security vulnerabilities issues in Docker,
```

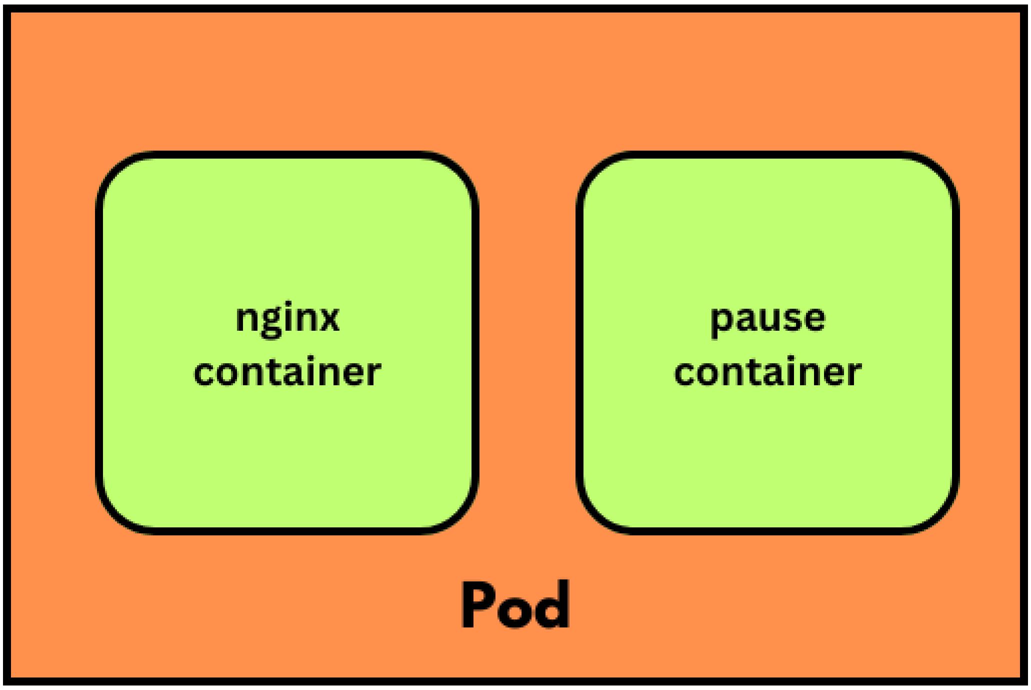
High-Level Openshift Architecture





Info - Pod Overview

```
- Pod is a collection of related containers
- it is configuration object that is stored and maitained within etcd key-
value database by API server
- as per recommended industry best practice, one main application per Pod
is better
- Pod is the smallest unit that can be deployed within Kubernetes/Openshift
- all the containers within a Pod, shares the IP address, ports, etc.,
- each Pod get a its own dedicated port range 0-65535
```



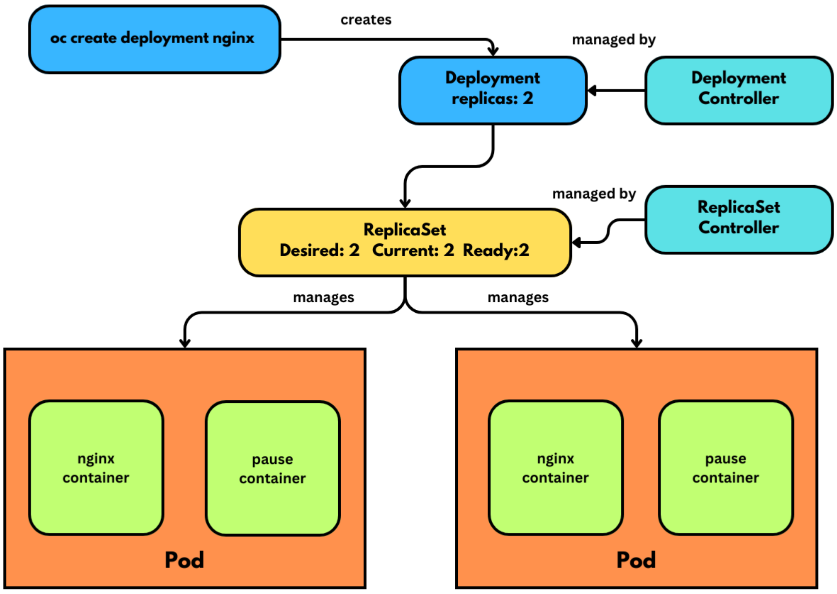
Info - ReplicaSet Overview

```
- it is a configuration object that is stored and maintained within etcd
key-value database by API server
- it tells how many instance of a particular Pod type should be running at
point of time
- this configuration object is consumed by ReplicaSet controller
- it the ReplicaSet controller which manages the Pod instances
- the ReplicaSet controller is the one which supports scale up/down
```

Info - Deployment Overview

```
- stateless applications are normally deployed as a deployment
- it is a configuration object that is stored and maintained within etcd
database by API server
- it describes
  - name of the application deployment
```

```
    - number of Pod instances expected to be running
    - container image and version that must be used to create Pod
  - it supports rolling update
  - this configuration object is consumed by Deployment Controller
```



Kube config file

- the oc/kubectl client tools requires a config file that has connection details to the API Server(load balancer)
- the config file is generally kept in user home directory, .kube folder and the default name of kubeconfig is config
- optionally we could also use the --kubeconfig flag with the oc command to point to a config file
- it is also possible to use a KUBECONFIG environment variable to point to the config file
- Just to give an idea, it is possible that your Kubernetes/OpenShift is running in AWS/Azure but you could install oc/kubectl client tool on your laptop with a config file and still run all the oc/kubectl commands from your laptop without going to aws/azure

To print the content of kubeconfig file

```
cat ~/.kube/config
```

About Red Hat Enterprise Core OS (RHCOS)

- an optimized operating system created especially for the use of Container Orchestration Platforms
- each version of RHCOS comes with a specific version of Podman Container Engine and CRI-O Container Runtime
- RHCOS enforces many best practices and security features

- it allows writing to only folders the application will has read/write access
- if an application attempts to modify a read-only folder RHCOS will not allow those applications to continue running
- RHCOS also reserves many Ports for the internal use of Openshift
- User applications will not have write access to certain reserved folders, user applications are allowed to perform things as non-admin users only, only certain special applications will have admin/root access

Points to remember

- Red Hat Openshift uses RedHat Enterprise Linux Core OS
- RHCOS has many restrictions or insists best practises
- RHEL Core OS reserves ports under 1024 for its internal use
- Many folders within the OS is made as ready only
- Any application Pod attempts to perform write operation on those restricted folders will not be allowed to run
- For detailed documentation, please refer official documentation here https://docs.openshift.com/container-platform/4.8/architecture/architecture-rhcos.html

Info - Pod Lifecycle

- Pending - Container image gets downloads or there are no Persistent Volume to bound and claim them
- Running - The Pod is scheduled to a node and all containers in the Pod are up and running
- Succeeded - All containers in the Pod have terminated succesfully and not be restarted
- Failed - All containers in the Pod have terminated but one or more containers terminated with non-zero status or was terminated by Openshift
- Unknown - For some reason, the state of the Pod could not be obtained may be there is some problem in communicating to the node where the Pod is running

Info - Container Lifecycle

- Waiting - pulling the container image
- Running - container is running without issues
- Terminated - container in the Terminated state began execution and then either ran to completion or failed for some reason

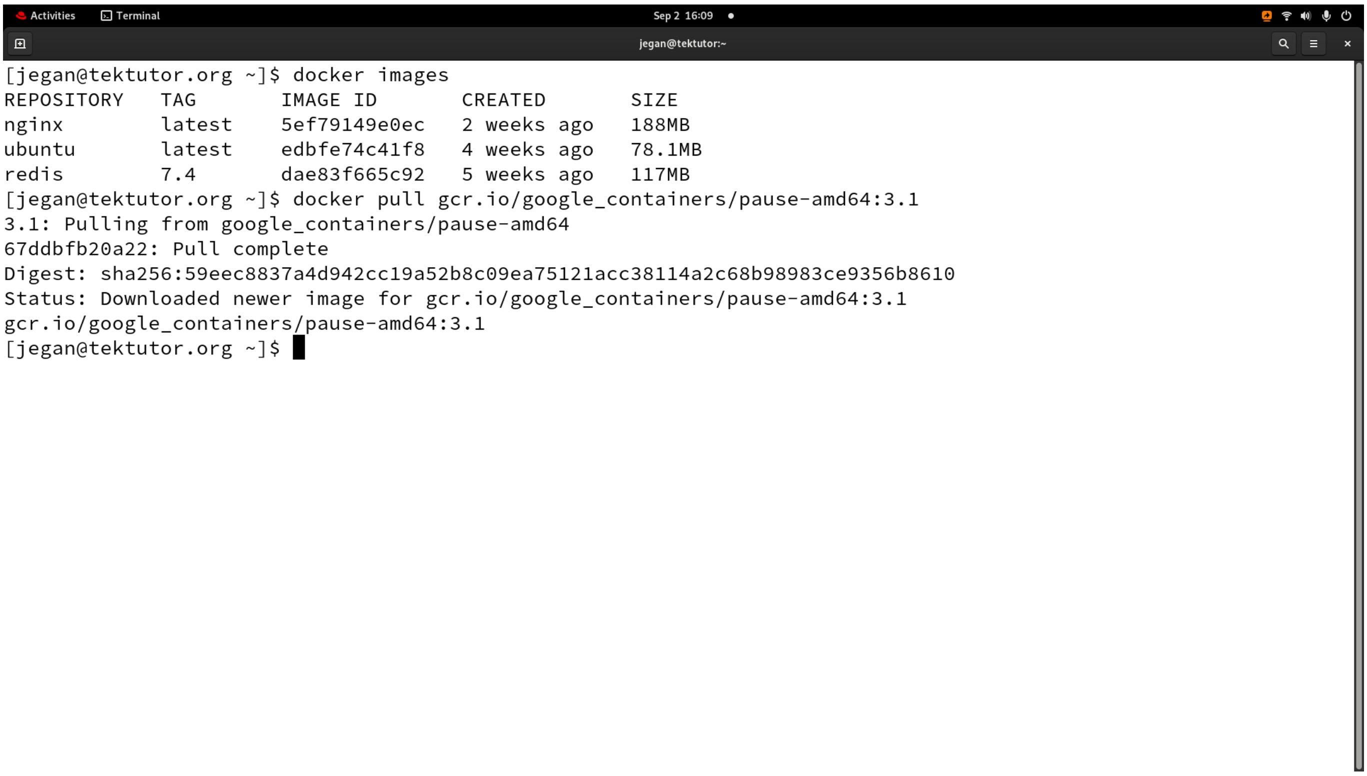
Lab - Checking the openshift tool version

```
oc version
kubectl version
oc get nodes
oc get nodes -o wide
```

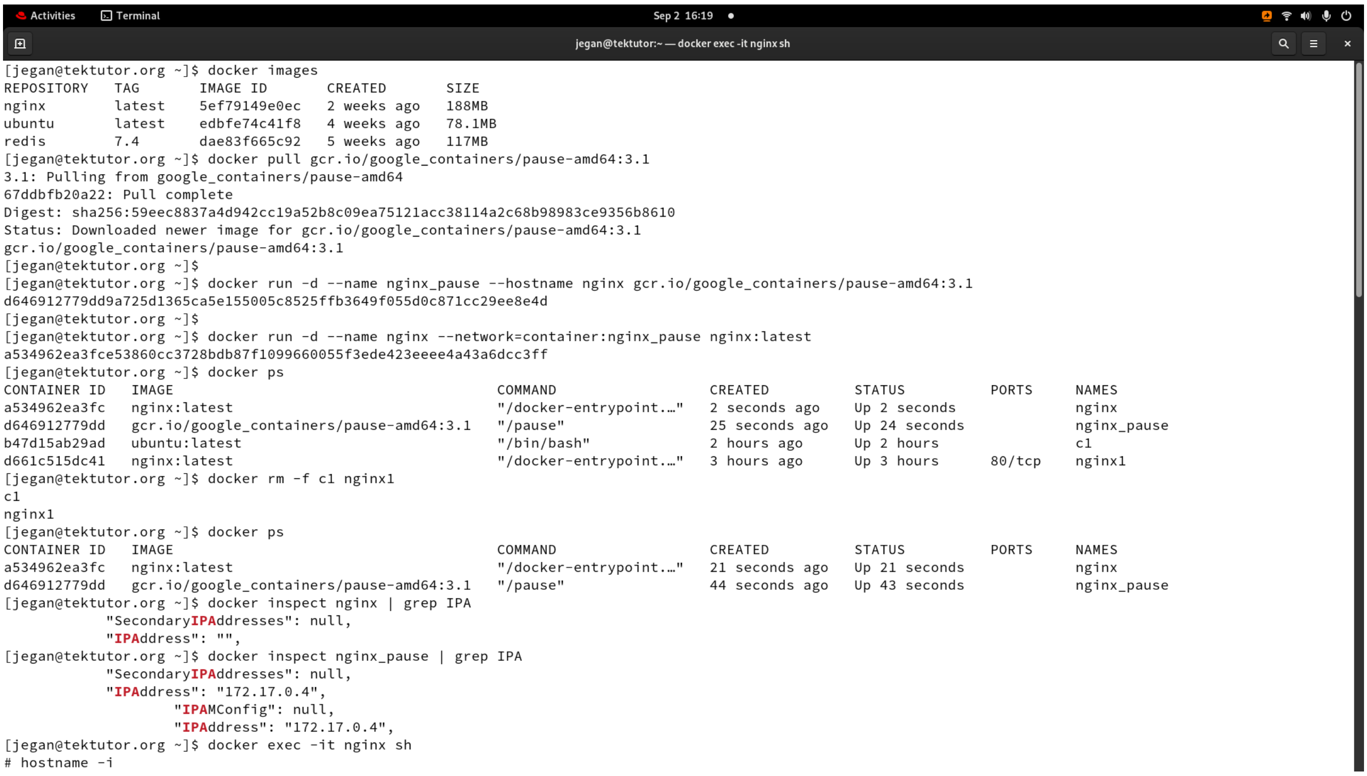
Lab - Creating a Pod in docker

```
docker images
docker pull gcr.io/google_containers/pause-amd64:3.1
docker images

docker run -d --name nginx_pause --hostname nginx
gcr.io/google_containers/pause-amd64:3.1
docker run -d --name nginx --network=container:nginx_pause nginx:latest
docker ps
docker inspect nginx_pause | grep IPA
docker exec -it nginx sh
hostname -i
hostname
apt update && apt install net-tools iputils-ping
ifconfig
exit
```

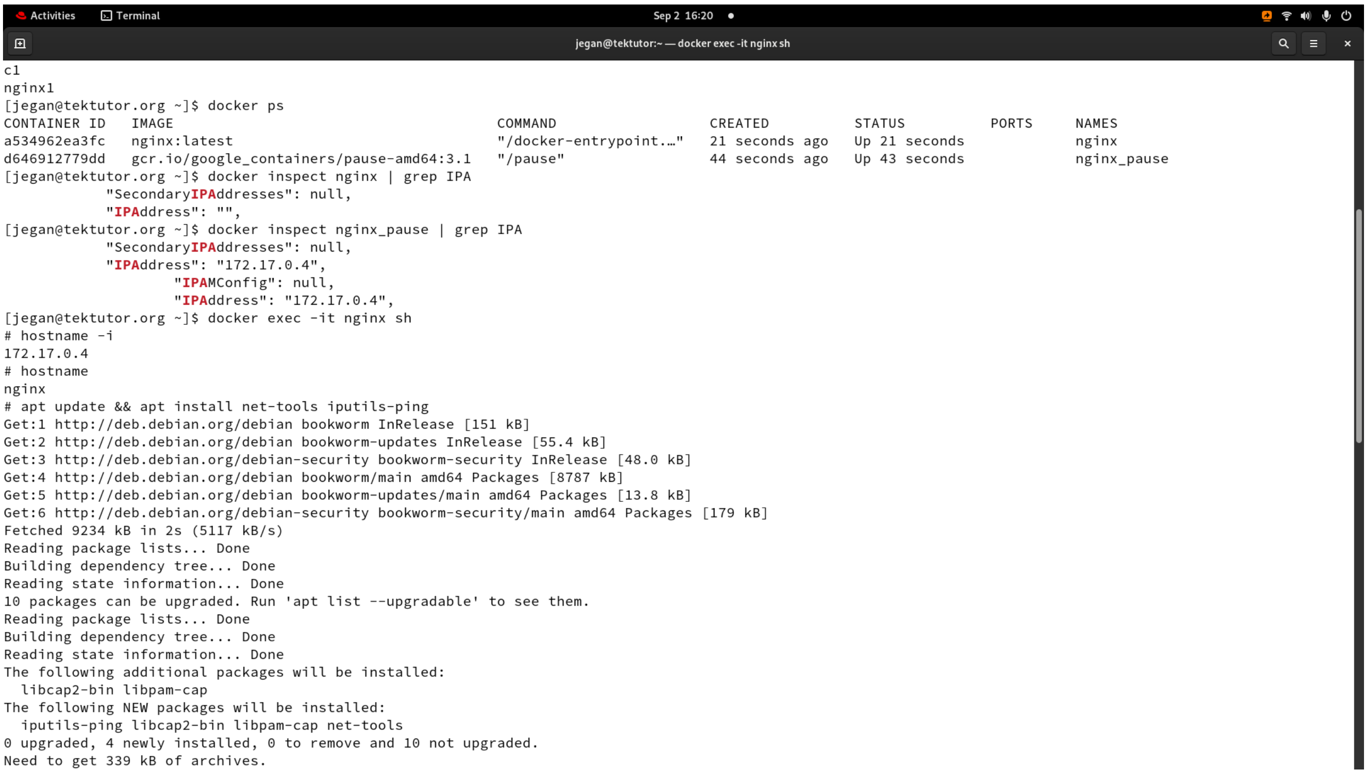
Expected output



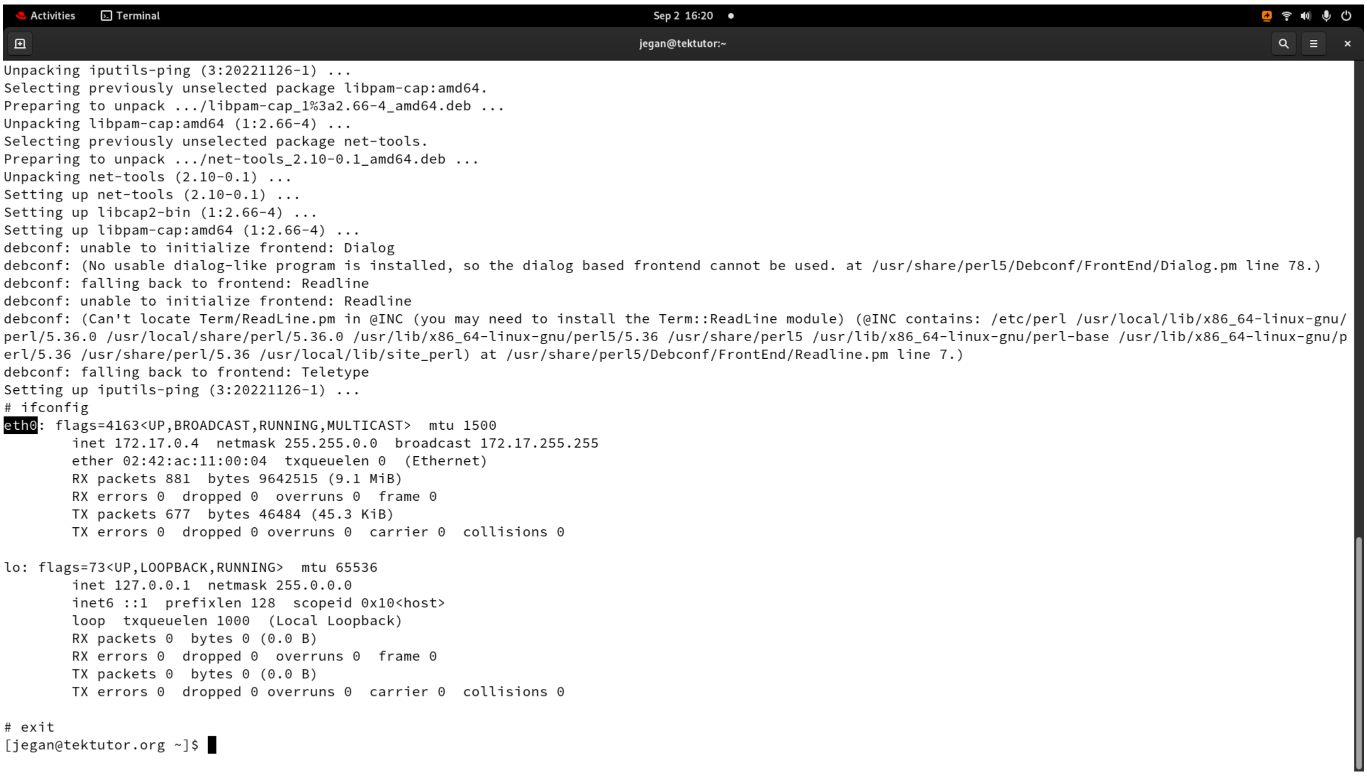
```
[jegan@tektutor.org ~]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
nginx        latest    5ef79149e0ec   2 weeks ago   188MB
ubuntu       latest    edbfe74c41f8   4 weeks ago   78.1MB
redis        7.4       dae83f665c92   5 weeks ago   117MB
[jegan@tektutor.org ~]$ docker pull gcr.io/google_containers/pause-amd64:3.1
3.1: Pulling from google_containers/pause-amd64
67ddbfb20a22: Pull complete
Digest: sha256:59eec8837a4d942cc19a52b8c09ea75121acc38114a2c68b98983ce9356b8610
Status: Downloaded newer image for gcr.io/google_containers/pause-amd64:3.1
gcr.io/google_containers/pause-amd64:3.1
[jegan@tektutor.org ~]$ 
```



```
[jegan@tektutor.org ~]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
nginx        latest    5ef79149e0ec   2 weeks ago   188MB
ubuntu       latest    edbfe74c41f8   4 weeks ago   78.1MB
redis        7.4       dae83f665c92   5 weeks ago   117MB
[jegan@tektutor.org ~]$ docker pull gcr.io/google_containers/pause-amd64:3.1
3.1: Pulling from google_containers/pause-amd64
67ddbfb20a22: Pull complete
Digest: sha256:59eec8837a4d942cc19a52b8c09ea75121acc38114a2c68b98983ce9356b8610
Status: Downloaded newer image for gcr.io/google_containers/pause-amd64:3.1
gcr.io/google_containers/pause-amd64:3.1
[jegan@tektutor.org ~]$ 
[jegan@tektutor.org ~]$ docker run -d --name nginx_pause --hostname nginx gcr.io/google_containers/pause-amd64:3.1
d646912779dd9a725d1365ca5e155005c8525ffb3649f055d0c871cc29ee8e4d
[jegan@tektutor.org ~]$ 
[jegan@tektutor.org ~]$ docker run -d --name nginx --network=container:nginx_pause nginx:latest
a534962ea3fce53860cc3728bdb87f1099660055f3ede423eeee4a43a6dcc3ff
[jegan@tektutor.org ~]$ docker ps
CONTAINER ID   IMAGE                                      COMMAND                  CREATED          STATUS          PORTS     NAMES
a534962ea3fc   nginx:latest                               "/docker-entrypoint.…"   2 seconds ago    Up 2 seconds              nginx
d646912779dd   gcr.io/google_containers/pause-amd64:3.1   "/pause"                 25 seconds ago   Up 24 seconds             nginx_pause
b47d15ab29ad   ubuntu:latest                              "/bin/bash"              2 hours ago      Up 2 hours                c1
d661c515dc41   nginx:latest                               "/docker-entrypoint.…"   3 hours ago      Up 3 hours      80/tcp    nginx1
[jegan@tektutor.org ~]$ docker rm -f c1 nginx1
c1
nginx1
[jegan@tektutor.org ~]$ docker ps
CONTAINER ID   IMAGE                                      COMMAND                  CREATED          STATUS          PORTS     NAMES
a534962ea3fc   nginx:latest                               "/docker-entrypoint.…"   21 seconds ago   Up 21 seconds             nginx
d646912779dd   gcr.io/google_containers/pause-amd64:3.1   "/pause"                 44 seconds ago   Up 43 seconds             nginx_pause
[jegan@tektutor.org ~]$ docker inspect nginx | grep IPA
            "SecondaryIPAddresses": null,
            "IPAddress": "",
[jegan@tektutor.org ~]$ docker inspect nginx_pause | grep IPA
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.4",
                    "IPAMConfig": null,
                    "IPAddress": "172.17.0.4",
[jegan@tektutor.org ~]$ docker exec -it nginx sh
# hostname -i
```



```
c1
nginx1
[jegan@tektutor.org ~]$ docker ps
CONTAINER ID   IMAGE                                      COMMAND                  CREATED          STATUS          PORTS     NAMES
a534962ea3fc   nginx:latest                               "/docker-entrypoint.…"   21 seconds ago   Up 21 seconds             nginx
d646912779dd   gcr.io/google_containers/pause-amd64:3.1   "/pause"                 44 seconds ago   Up 43 seconds             nginx_pause
[jegan@tektutor.org ~]$ docker inspect nginx | grep IPA
            "SecondaryIPAddresses": null,
            "IPAddress": "",
[jegan@tektutor.org ~]$ docker inspect nginx_pause | grep IPA
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.4",
                    "IPAMConfig": null,
                    "IPAddress": "172.17.0.4",
[jegan@tektutor.org ~]$ docker exec -it nginx sh
# hostname -i
172.17.0.4
# hostname
nginx
# apt update && apt install net-tools iputils-ping
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8787 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [13.8 kB]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [179 kB]
Fetched 9234 kB in 2s (5117 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
10 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcap2-bin libpam-cap
The following NEW packages will be installed:
  iputils-ping libcap2-bin libpam-cap net-tools
0 upgraded, 4 newly installed, 0 to remove and 10 not upgraded.
Need to get 339 kB of archives.
```



```
Unpacking iputils-ping (3:20221126-1) ...
Selecting previously unselected package libpam-cap:amd64.
Preparing to unpack .../libpam-cap_1%3a2.66-4_amd64.deb ...
Unpacking libpam-cap:amd64 (1:2.66-4) ...
Selecting previously unselected package net-tools.
Preparing to unpack .../net-tools_2.10-0.1_amd64.deb ...
Unpacking net-tools (2.10-0.1) ...
Setting up net-tools (2.10-0.1) ...
Setting up libcap2-bin (1:2.66-4) ...
Setting up libpam-cap:amd64 (1:2.66-4) ...
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (Can't locate Term/ReadLine.pm in @INC (you may need to install the Term::ReadLine module) (@INC contains: /etc/perl /usr/local/lib/x86_64-linux-gnu/
perl/5.36.0 /usr/local/share/perl/5.36.0 /usr/lib/x86_64-linux-gnu/perl5/5.36 /usr/share/perl5 /usr/lib/x86_64-linux-gnu/perl-base /usr/lib/x86_64-linux-gnu/p
erl/5.36 /usr/share/perl/5.36 /usr/local/lib/site_perl) at /usr/share/perl5/Debconf/FrontEnd/Readline.pm line 7.)
debconf: falling back to frontend: Teletype
Setting up iputils-ping (3:20221126-1) ...
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.4  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:04  txqueuelen 0  (Ethernet)
        RX packets 881  bytes 9642515 (9.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 677  bytes 46484 (45.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

# exit
[jegan@tektutor.org ~]$ 
```

Info - Kubernetes/Openshift Control Plane Components

The control plane components only runs in master node

```
1. API Server
2. etcd database
3. Controller Managers
4. Scheduler
```

API Server

```
- API Server is the heart of Kubernetes/Openshift
- supports REST API for all the features supported by OpenShift
- all the control plane components interacts only with API Server by making a REST call
- API Server stores/updates/retrieves the cluster status, application status into etcd database
- each change that is done in the etcd database will result in API Server raising events
- all the Openshift components will be communicate only to API Server
- other components are not allowed to communicate with each other directly
- every components communication flows via API Server only in Kubernetes/Openshift
- API Server maintains the nodes, cluster, application status in the etcd database
- only API Server will have access to etcd database
- In our openshift cluster, 3 master nodes are there, hence 3 API Servers i.e one API Server per master node is there
- API Server sends broadcasting events whenever any update happens in the
etcd database
  - new record added
  - existing record updated
  - existing record deleted
```

Etcd database

```
- key/value database
- it is an independent opensource database used in Kubernetes & Openshift
- generally etcd database works as a cluster i.e group of etcd instances
works together as a cluster
- when one of the etcd db instance is updates, the data is synchronized
automatically on other etcd instances within the etcd cluster
- minimum number of instances required to create an etcd cluster is 3
```

Controller Managers

```
- it is a collection of many controllers
- Controllers are deployed as Pods
- Each Controller watches the cluster
  - when new resources are created
  - when existing resources in etcd are edited/updated
  - when existing resource are deleted from etcd
- Based on the events from API server the Controller job to reconcile based
on updated configuration objects
- Some controllers within Controller Manager Pod
  - Deployment Controller
  - ReplicaSet Controller
  - EndPoitnt Controller
  - Job Controller
  - StatefulSet Controller
  - DaemonSet Controller
- Each Controller manages one type of Kubernetes/OpenShift resource
- For example
  - Deployment Controller manages Deployment resource
- Deployment Controller watches for events related to Deployment Resource
  - New deployment created
  - Depoyment edited
  - Deployment deleted
- ReplicaSet Controller watches for events from API Server related to
ReplicaSet Resource
  - New ReplicaSet created
  - ReplicaSet edited
  - ReplicaSet deleted
```

Scheduler

```
- this is the component that is responsible to find a healthy node where a
new Pod can be deployed
- Scheduler sends its scheduling recommendataion to API Server
- API Server updates the scheduling info on each Pod stored in the etcd
database
- API Server broadcasts events for each Pod deployed onto some node
- Kubelet is the container agent that runs in every node ( master and
worker nodes)
- kubelet downloads the container image, creates and starts the container
- kubelet keeps monitoring the status of the container running on the local
node and reports the status on a heart-beat like periodic fashion to the
API Server
```

Info - Understanding Red Hat Openshift installation

```
- Assume we wish to have 3 master nodes with 3 worker nodes
- each of these nodes can be a Virtual Machine in on-prem server or it
could be a physical in your office or it could be an ec2 in aws or it could
be an azure vm in azure cloud
- OpenShift master nodes will accept only Red Hat Enterprise Core OS as the
Operating System
- The Red Hat Enterprise Core OS  (RHCOS)
  - it comes Podman Container Engine and CRI-O Container Runtime pre-
installed
- Openshift worker nodes has opt for either Red Hat Enterprise Linux (RHEL)
or Red Hat Enterprise Core Linux (RHCOS)
- Bastion Virtual Machine (Helper Virtual Machine)
  - Can be any Linux OS
  - In our lab, I have installed Fedora 39 Linux
  - This is where we will install DNS Server
  - This is where we will install DHCP Server
  - This is where we will install HAProxy Load Balancer server
  - The HA Proxy Load Balancer will host the following
    - Red Hat Enterprise Core OS Image
    - Red Hat Openshift master node ignition file
    - Red Hat Openshift worker node ignition file
    - Red Hat Openshift bootstrap node ignition file
- BootStrap Virtual Machine ( Virtual Machine - used to setup Openshift
master and worker nodes )
  - Red Hat Enterprise Core OS
  - In order to install openshift, the openshift installer creates a simple
Kubernetes cluster within BootStrap Virtual Machine
  - The BootStrap Virtual Machine - Kubernetes cluster installs the
required openshift master node components into master-1, master-2 and
master-3 VMs
  - Once the Kubernetes cluster running in BootStrap VMs finds the master-
1, master-2 and master-3 nodes in openshift are found to be stable, the
Kubernetes cluster running in BootStrap VM is no more required
  - The sole purpose of BootStrap Virtual Machine is to create a 3 node
master cluster in Openshift
```

```
  - In most case, people prefer disposing this VM once the 3 openshift master nodes are up and running as a cluster
- Master Node 1
  - We need to Red Hat Enterprise Core OS as an Operating Sytem
  - RHCOS comes with
    - preinstalled Podman Container Engine and CRI-O container runtime
  - Control Plane Components will be running
    - API server (Pod)
    - etcd database (Pod)
    - controller managers (Pod)
    - scheduler (Pod)
- Master Node 2
  - We need to Red Hat Enterprise Core OS as an Operating Sytem
  - RHCOS comes with
    - preinstalled Podman Container Engine and CRI-O container runtime
  - Control Plane Components will be running
    - API server (Pod)
    - etcd database (Pod)
    - controller managers (Pod)
    - scheduler (Pod)
- Master Node 3
  - We need to Red Hat Enterprise Core OS as an Operating Sytem
  - RHCOS comes with
    - preinstalled Podman Container Engine and CRI-O container runtime
  - Control Plane Components will be running
    - API server (Pod)
    - etcd database (Pod)
    - controller managers (Pod)
    - scheduler (Pod)
  - kube-proxy (Pod)
  - kubelet (service)
  - coreDNS (Pod)
- Worker Node 1
  - We need to Red Hat Enterprise Core OS as an Operating Sytem
  - RHCOS comes with
    - preinstalled Podman Container Engine and CRI-O container runtime
  - kube-proxy (Pod)
  - kubelet (service)
  - coreDNS (Pod)
- Worker Node 2
  - We need to Red Hat Enterprise Core OS as an Operating Sytem
  - RHCOS comes with
    - preinstalled Podman Container Engine and CRI-O container runtime
  - kube-proxy (Pod)
  - kubelet (service)
  - coreDNS (Pod)
- Worker Node 3
  - We need to Red Hat Enterprise Core OS as an Operating Sytem
  - RHCOS comes with
    - preinstalled Podman Container Engine and CRI-O container runtime
  - kube-proxy (Pod)
  - kubelet (service)
  - coreDNS (Pod)
```

Info - Openshift client tools

```
oc - is the command-line openshift client
kubectl - is the kubernetes command-line client
both oc and kubectl are supported in openshift cluster
```

OpenShift resources

- Deployment (K8s resource)
- ReplicaSet (K8s resource)
- Pod (K8s resource)
- Job (K8s resource)
- DaemonSet (K8s resource)
- StatefulSet (K8s resource)
- Build (OpenShift resource - Custom Resource added by OpenShift)
- ImageStream (OpenShift resource - Custom Resource added by OpenShift)
- DeploymentConfig (OpenShift resource - Custom Resource added by OpenShift)

Deployment command looks like this

```
oc create deployment nginx --image=bitnami/nginx:latest --replicas=3
```

Deployment

- This is a JSON/YAML definition which is stored in etcd database
- The deployment is managed by Deployment Controller
- when we applications, they are deployed as Deployment with Kubernetes/OpenShift
- Deployment Controller creates ReplicaSet, which is then managed by ReplicaSet Controller
- Deployment has one or more ReplicaSet(s)

ReplicaSet

- This is a JSON/YAML definition which is stored in etcd database
- The ReplicaSet is managed by ReplicaSet Controller
- ReplicaSet capture details like
    - How many Pod instances are desired?
- ReplicaSet Controller reads the ReplicaSet definition and learns the desired Pod instance count
- ReplicaSet Controller creates so many Pod definition as indicated in the ReplicaSet
- ReplicaSet Controller ensures the desired Pod count matches with the actual Pod count, whenever a Pod crashes, it is the responsibility of ReplicaSet Controller to ensure the desired and actual Pods are equal
- ReplicaSet has one or more Pods

Pod

- is a collection of one or more Containers
- IP address is assigned on the Pod level not on the Container level
- If two containers are in the same Pod, there will be sharing IP Address of the Pod
- within container, application are deployment (tomcat,mysql, nginx these are applications)
- recommended best practice,only one application should be there in a Pod
- Pods are scheduled by Scheduler onto some Node
- every Pod has a Network Stack and Network Interface Card (NIC)

Kubelet

- is a daemon service that interacts with the Container Runtime on the current node/server where kubelet is running
- kubelet downloads the required container image and creates the Pod containers
- kubelet frequently reports the status of Pod container status to the API server
- kubelet also monitors the health of POds running on the node and ensures they are healthy
- kubelet will there on every node (master and worker nodes)

kube-proxy

- is a Pod that runs one instance per node (both master and worker nodes)
- provides load-balancing a group of similar Pods

Core DNS

- is a Pod that runs usually on master nodes but in some installation, it might even run on worker node
- Core DNS offers Service Discovery i.e application will be connect to group of Pods by Service name

Kubectl

- is a client tool used to create and manage deployments and services in Kubernetes
- it also works in OpenShift
- it make REST call to API Server

OC

- is a client tool used to create and manage Openshift resources in OpenShift
- it makes REST call to API Server