Day 1

First Day - Feedback

```
https://survey.zohopublic.com/zs/y4DHH9
```

What is Boot Loaders

- it is system utility that get's installed in Master Boot Record(MBR)
- in our Hard Disk(storage) we have Sector 0, Byte 0 which is called as Master Boot Record(MBR)
- the boot loader software is installed in the MBR
- When a system is booted, first the BIOS Power On Self Test(POST) will happen
- Examples
    - LILO
    - GRUB 1/2

What is dual/multi booting?

- installing dual or multiple OS on the same machine
- we can boot into only one OS at a time
- at any point of time only one Operating System can be active
- you need to install GRUB or similar boot loader to support booting into any one of the OS installed on your laptop/desktop

What is Hypervisor?

- is nothing but Virtualization
- Virtualization is a Hardware + Software technology
- General Purpose Processors (x86, x86_64) - 32 bit and 64 bit Processors
    - AMD (AMD-V - Virtualization support)
    - Intel (VT-X - Virtualiation support)
- Apple Silicon Processor - Based on ARM Processor (M1/M2 - supports virtualization)
- Many OS can be actively running at the same time on the same laptop/desktop/workstation/server
- Examples
    - VMWare
        - Workstation - Type 2 Hypervisor (Linux/Windows)
        - Fusion - Type 2 Hypervisor (Mac OS-X)
        - vSphere/vCenter - Type 1 Hypervisor a.k.a Bare-metal Hypervisors
    - Oracle Virtualbox - Type 2 Hypervisor (Linux/Mac/Windows - Free)
    - Parallels - Type 2 Hypervisor (Mac OS-X)
    - each OS runs in a separate Virtual Machine (VM)
    - each Virtual Machine requires dedicated hardware resources
        - Virtual CPU Cores
        - RAM - Actual

        - Storage - Actual
        - Network (Virtual - Software defined Network cards)
        - Graphics Card (Virtual - Software defined Network cards)
    - Because every Virtual Machines requires dedicated hardware resources, this type of Virtualization is called Heavy-weight Virtualization

Processor Packaging

- Processors comes in 2 packages
- Single Chip Module (SCM) - One Processor per IC
- Multiple Chip Module (MCM) - Many Processors per IC

CPU Cores

- each Processor may host many CPU Cores
- For instance, these days AMD/Intel has Processors that support 128/256/512 cores in a single Processor

Physical vs Logical/Virtual Cores

- each modern Physical CPUs are capable of running multiple threads parallely
- Hyperthreading - each physical CPU supports 2/4/8 virtual cores

To support 1000 OS, how many physical machines are required in the absence of Virtualization Technology

- 1000 Physical machines are required
- data-center maintenance
- cost of each server
- power consumption
- real-estate cost - lease/rent
- Uninterupped power supply (UPS/INverter)
- Air Conditioning
- Sound proofing

To support 1000 OS, how many least number of physical machines are required with Virtualization support?

- 1 Server ie enough technically
- 8 Socket Server Motherboard
- Assume in each each Processor Socket we have installed a MCM packaged Processors
    - MCM IC with 4 Processors
    - Each Processor supports 128 Physical CPU Cores
    - each Socket - how many cores - 512 Physical CPU cores
    - In 8 Socket - how many cores - 512 x 8 = 4096 Physical CPU Cores
    - In 8 Sockets - how many virtual cores - 4096 x 2 = 8192 virtual/logical cores
- RAM/Storage

Container Overview

What they are not

- it is not Hypervisor
- it is not a Virtual Machine
- containers don't represent Operating System

What they are

- it is an application virtualization technology
- it is a single application process
- each container represents one application
- one application may require one to many containers
- every container has its own network stack
- every container has its own software defined network(virtual) network card
- every container get its own file system
- every container get its own port range (0-65535)
- they get their own IP address
- For example
    - Oracle DB Server can run in a single container
    - MySQL DB Server can run in a single container
    - Tomcat Server in a single container
    - REST/SOAP API can run inside a container
    - Webservice can run inside a container
    - one Microservice per container
    - one Application Server can run in a container
    - one Web Server can run in a container
    - one Message Queue Server can run in a container
    - Apache Kakfa Server can run in a container

Containers don't have

- their own hardware resources
- their own OS Kernel

What is Container Runtime?

- it is a low-level software
- it is not so user-friendly
- it manages container images and containers
- normally no end-users use the container runtimes directly
- Examples
    - runC
    - CRI-O

What is Container Engine?

- is a high-level software
- it offers user-friendly commands to manage container images and containers

- internally they depend on Container Runtimes to manage images and containers
- end-users like us, normally will use Container Engines as they are very user-friendly
- Examples
    - Docker is a Container Engine that depends on containerd which internally depends on runC container Runtime
    - Podman is a Container Engine that depends on CRI-O container Runtime

Docker Overview

- is a container engine
- developed in Go language
- developed by a company called Docker Inc
- Docker comes in 2 flavours
    1. Community Edition - Docker CE (Free)
    2. Enterprise Edition - Docker EE (Paid)

Podman Overview

- is opensource but primarily maintained by Red Hat
- CoreOS was the company that had 2 products
    - CoreOS - Operating system optimized for Container Orchestration Platforms
    - rkt - pronounced as rocket, which is a container runtime software
- Red Hat acquired CoreOS, they kind replaced rkt with CRI-o Container Runtime
- Red Hat CoreOS is now called Red Hat Enterprise Core OS (RHCOS)
- RHCOS is the Operating System used in Red Hat Openshift 4.x onwards
- Podman is opensource container engine maintained by Red Hat
- it is an alternate product for Docker

Docker High Level Architecture

- follows client/server architecture
- client runs in user-space
- while server runs in kernel space (as root user)
- end-users use only the client software to interact with the Docker server
- Docker server runs in the background as Service(daemon)

Container Orchestration Platform Overview

- High-Level Features
    - provides a platform where you could your applications and make them Highly Available (HA)
    - it has built-in monitoring features to check the health of your application and repair them on need basis
        - readiness check (it is live and ready to server user request)
        - liveliness check (the application is running but may not ready to support user requests)
    - scale up/down
        - adding more instances of your application when the user-traffic increases
        - removing idle application instances when the user-traffic reduces
    - rolling update

        - upgrade your already live application from one version to other without any downtime
    - rollback
        - downgrade your already latest live application from latest version to older versions without any downtime
    - expose your container applications workloads to external world via Services (Service discovery)
    - Services 2 types
        - Internal (accessible within the orchestraion platform only) and
        - External (accessible outside the orchestrtion platform as well)
- can manage the application life cycle within the Orchestration platform
- Examples
    - Docker SWARM
        - a light-weight
        - free
        - Docker Inc's native Orchestration platform
        - it supports only Docker containerized application workloads
    - Google Kubernetes
        - opensource container orchestration platform
        - it supports managing different containerized application workloads
            - microservices running in docker container
            - tomcat running in containerd container
            - oracle db server running in LXC container
            - also supports Podman
        - initial days Kubernetes by default was supported Docker
    - Red Hat OpenShift
        - is developed on top of Google Kubernetes
        - it is an enterprise software that requires paid license

Red Hat OpenShift - Container Orchestration Platform High-Level Architecture

- Openshift cluster has 2 types of nodes
    1. Master
    2. Worker
- Control Plane Components runs only in the master node
- Worker Nodes - this is where user application will be deployed by default
- It is also possible to configure the master nodes to allow deploying user applications apart from control plane components
- client tools
    - oc (openshift client tool)
    - kubectl (kubernetes client tool)

What are the Control Plane Compenents in OpenShift/Kubernetes

1. API Server
2. etcd key-value datastore/database
3. scheduler
4. controller managers (a collection of many controllers)

API Server

- this is a collection REST APIs for every features supported by OpenShift
- stores the entire cluster status, user application status,nodes status, etc into the etcd database
- API Server is the only components normally has access to etcd databases
- all Openshift components interact with API Server by making a REST API call
- API Server responds to REST calls via events
- Whenever API servers makes any change in etcd database, it will be followed by a broadcasting event about the change it made in the etcd db
- oc/kubectl client tools will also talk to API server only

etcd database

- key/value database
-

Scheduler

Controller Managers

- it is a collection of many controllers
- controllers are applications that run continuously in an infinite loop waiting for events
    - new deployment created
    - new Pod created
    - Pod deleted
    - deployment deleted
    - Deployment scaled up
    - Deployment scaled down
- every Resource in Openshift is managed by one Controller
- Example
    - Deployment is a resource in Kubernetes/Openshift that represents an application deployment
    - Deployment Controller is the controller that monitors, manages, repairs the Deployment resource
    - ReplicaSet controller monitors, manages and repairs ReplicaSet resource
    - Endpoint Controller
    - Job Controller
    - CronJob Controller
    - DaemonSet Controller
    - StatefulSet Controller
    - ReplicationController

What is a Pod?

- a collection of many related containers
- inside each container one application or component will be running
- multiples are hosted/running
- it is record/yaml definition stored in etcd database
- is the smallest unit that can be deployed into Openshift/Kubernetes
- For instance
    - If you deploy Jenkins, jenkins will run inside a container which is part of a Pod
- Unlike container, where each container gets IP address(es), in Kubernetes/Openshift IP address(es) are assigned on the Pod level not on the container level
- In other words, the containers running with the same Pod shares the same IP address and ports

OpenShift resources

- Deployment
- ReplicaSet
- Pod
- all the above are resources (yaml definitions) records stored in etcd database
-

Lab - Find the Openshift version details

```
oc version
```

Expected output

```
jegan@tektutor.org $ oc version
Client Version: 4.15.12
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: 4.15.12
Kubernetes Version: v1.28.9+2f7b992
```

Lab - Listing the openshift nodes in the cluster

```
oc get nodes
```

Expected output

```
jegan@tektutor.org $ oc get nodes
NAME                                STATUS   ROLES
AGE    VERSION
master-1.ocp4.tektutor.org.labs   Ready    control-plane,master,worker
7d1h   v1.28.9+2f7b992
master-2.ocp4.tektutor.org.labs   Ready    control-plane,master,worker
```

```
7d1h   v1.28.9+2f7b992
master-3.ocp4.tektutor.org.labs   Ready    control-plane,master,worker
7d1h   v1.28.9+2f7b992
worker-1.ocp4.tektutor.org.labs   Ready    worker                         7d
v1.28.9+2f7b992
worker-2.ocp4.tektutor.org.labs   Ready    worker                         7d
v1.28.9+2f7b992
```

Lab - Finding the IP address of nodes, finding the OS installed on the nodes

You can use the wide mode to find

- Ip address of the nodes
- OS installed on the nodes
- CRI-O Container Runtime version
- Node Status
- Node roles (master/worker)

```
oc get nodes -o wide
```

Expected output

```
jegan@tektutor.org $ oc get nodes -o wide
NAME                              STATUS   ROLES
AGE    VERSION           INTERNAL-IP       EXTERNAL-IP   OS-IMAGE
KERNEL-VERSION                 CONTAINER-RUNTIME
master-1.ocp4.tektutor.org.labs   Ready    control-plane,master,worker
7d1h   v1.28.9+2f7b992   192.168.122.35            Red Hat Enterprise Linux
CoreOS 415.92.202404302054-0 (Plow)   5.14.0-284.64.1.el9_2.x86_64   cri-
o://1.28.6-2.rhaos4.15.git77bbb1c.el9
master-2.ocp4.tektutor.org.labs   Ready    control-plane,master,worker
7d1h   v1.28.9+2f7b992   192.168.122.112           Red Hat Enterprise Linux
CoreOS 415.92.202404302054-0 (Plow)   5.14.0-284.64.1.el9_2.x86_64   cri-
o://1.28.6-2.rhaos4.15.git77bbb1c.el9
master-3.ocp4.tektutor.org.labs   Ready    control-plane,master,worker
7d1h   v1.28.9+2f7b992   192.168.122.36            Red Hat Enterprise Linux
CoreOS 415.92.202404302054-0 (Plow)   5.14.0-284.64.1.el9_2.x86_64   cri-
o://1.28.6-2.rhaos4.15.git77bbb1c.el9
worker-1.ocp4.tektutor.org.labs   Ready    worker                         7d
v1.28.9+2f7b992   192.168.122.102           Red Hat Enterprise Linux CoreOS
415.92.202404302054-0 (Plow)   5.14.0-284.64.1.el9_2.x86_64   cri-
o://1.28.6-2.rhaos4.15.git77bbb1c.el9
worker-2.ocp4.tektutor.org.labs   Ready    worker                         7d
v1.28.9+2f7b992   192.168.122.38            Red Hat Enterprise Linux CoreOS
415.92.202404302054-0 (Plow)   5.14.0-284.64.1.el9_2.x86_64   cri-
o://1.28.6-2.rhaos4.15.git77bbb1c.el9
```

Lab - Each container gets an IP address in Docker (Not in Openshift/Kubernetes)

```
docker run -dit --name c1 --hostname c1 ubuntu:16.04 /bin/bash
docker run -dit --name c2 --hostname c2 ubuntu:16.04 /bin/bash
docker ps
docker inspect -f {{.NetworkSettings.IPAddress}} c1
docker inspect -f {{.NetworkSettings.IPAddress}} c2
docker inspect c2 | grep IPA
docker inspect c1 | grep IPA
```

Expected output

```
 jegan@tektutor.org $ docker run -dit --name c1 --hostname c1 ubuntu:16.04
/bin/bash
21d42db717ee5122e048dfd631521738410cb268941f27048e9229e6d607323a
 jegan@tektutor.org $ docker run -dit --name c2 --hostname c2 ubuntu:16.04
/bin/bash
4fc195efc2650c814f995a0cce706c4c6e9ef9a2f8c0d673f3bcaf9e8e27e669
 jegan@tektutor.org $ docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED         STATUS
PORTS     NAMES
4fc195efc265   ubuntu:16.04   "/bin/bash"   2 seconds ago   Up 1 second
c2
21d42db717ee   ubuntu:16.04   "/bin/bash"   7 seconds ago   Up 6 seconds
c1
 jegan@tektutor.org $ docker inspect -f {{.NetworkSettings.IPAddress}} c1
172.17.0.2
 jegan@tektutor.org $ docker inspect -f {{.NetworkSettings.IPAddress}} c2
172.17.0.3
 jegan@tektutor.org $ docker inspect c2 | grep IPA
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.3",
                    "IPAMConfig": null,
                    "IPAddress": "172.17.0.3",
 jegan@tektutor.org $ docker inspect c1 | grep IPA
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.2",
                    "IPAMConfig": null,
                    "IPAddress": "172.17.0.2",
```

Lab - Creating a Pod in docker

In case of a Pod, only one container in the group of containers (Pod) has a network interface. The other containers within the Pod shares the infra/secret/pause container's network, hence they all share the same IP address.

```
docker run -d --name nginx --network=container:nginx_pause
gcr.io/google_containers/pause:latest
docker run -d --name nginx --network=container:nginx_pause nginx:latest
docker ps
docker inspect {{.NetworkSettings.IPAddress}} nginx_pause
docker inspect {{.NetworkSettings.IPAddress}} nginx
```

Lab - Creating a project in Openshift

Things to note

```
- Project is created for every project team by the Administrators
- Inside a project multiple applications can be deployed
- project access can be restricted to a users
- only those team members who are part of a team can access a specific
project and resources under the project.
- projects helps seggregate one team's deployments from others
```

In the below command replace 'jegan' with your name.

```
oc new-project jegan
```

Expected output

```
 jegan@tektutor.org $ oc new-project jegan
Already on project "jegan" on server
"https://api.ocp4.tektutor.org.labs:6443".

You can add applications to this project with the 'new-app' command. For
example, try:

    oc new-app rails-postgresql-example

to build a new example application in Ruby. Or use kubectl to deploy a
simple Kubernetes application:

    kubectl create deployment hello-node --image=registry.k8s.io/e2e-test-
images/agnhost:2.43 -- /agnhost serve-hostname
```

Lab - Listing all projects

```
oc projects
oc get projects
```

```
oc get project
oc get namespaces
oc get namespace
oc get ns
```

Expected output

```
jegan@tektutor.org $ oc projects
You have access to the following projects and can switch between them with
' project ':

    default
  * jegan
    knative-eventing
    knative-serving
    knative-serving-ingress
    kube-node-lease
    kube-public
    kube-system
    openshift
    openshift-apiserver
    openshift-apiserver-operator
    openshift-authentication
    openshift-authentication-operator
    openshift-cloud-controller-manager
    openshift-cloud-controller-manager-operator
    openshift-cloud-credential-operator
    openshift-cloud-network-config-controller
    openshift-cloud-platform-infra
    openshift-cluster-csi-drivers
    openshift-cluster-machine-approver
    openshift-cluster-node-tuning-operator
    openshift-cluster-samples-operator
    openshift-cluster-storage-operator
    openshift-cluster-version
    openshift-config
    openshift-config-managed
    openshift-config-operator
    openshift-console
    openshift-console-operator
    openshift-console-user-settings
    openshift-controller-manager
    openshift-controller-manager-operator
    openshift-dns
    openshift-dns-operator
    openshift-etcd
    openshift-etcd-operator
    openshift-host-network
    openshift-image-registry
    openshift-infra
    openshift-ingress
    openshift-ingress-canary
```

```
    openshift-ingress-operator
    openshift-insights
    openshift-kni-infra
    openshift-kube-apiserver
    openshift-kube-apiserver-operator
    openshift-kube-controller-manager
    openshift-kube-controller-manager-operator
    openshift-kube-scheduler
    openshift-kube-scheduler-operator
    openshift-kube-storage-version-migrator
    openshift-kube-storage-version-migrator-operator
    openshift-machine-api
    openshift-machine-config-operator
    openshift-marketplace
    openshift-monitoring
    openshift-multus
    openshift-network-diagnostics
    openshift-network-node-identity
    openshift-network-operator
    openshift-node
    openshift-nutanix-infra
    openshift-oauth-apiserver
    openshift-openstack-infra
    openshift-operator-lifecycle-manager
    openshift-operators
    openshift-ovirt-infra
    openshift-ovn-kubernetes
    openshift-route-controller-manager
    openshift-serverless
    openshift-service-ca
    openshift-service-ca-operator
    openshift-user-workload-monitoring
    openshift-vsphere-infra

Using project "jegan" on server "https://api.ocp4.tektutor.org.labs:6443".
```

Lab - Switching between projects

```
oc project default
oc project jegan
```

Lab - Finding the current active project

```
oc project
```

Lab - Deleting a project (kindly delete only projects you created)

Deleting a project, automatically deletes all the resources under the project.

```
oc get projects | grep jegan
oc delete project jegan
```

Expected output

```
jegan@tektutor.org $ oc get projects | grep jegan
jegan                                                                   Active
jegan@tektutor.org $ oc delete project jegan
project.project.openshift.io "jegan" deleted
```

Lab - First deployment

```
oc new-project jegan
oc project jegan
oc create deployment nginx --image=nginx:latest --replicas=3
```

Listing the deployment

```
oc get deployments
oc get deployment
oc get deploy
```

Listing the replicasets

```
oc get replicasets
oc get replicaset
oc get rs
```

Listing the pods

```
oc get pods
oc get pod
oc get po
```

Expected output

```
 jegan@tektutor.org $ oc project
Using project "jegan" on server "https://api.ocp4.tektutor.org.labs:6443".
 jegan@tektutor.org $ oc create deployment nginx --image=nginx:latest --
replicas=3
deployment.apps/nginx created
 jegan@tektutor.org $ oc get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   0/3     3            0           6s
 jegan@tektutor.org $ oc get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   0/3     3            0           7s
 jegan@tektutor.org $ oc get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   0/3     3            0           10s
 jegan@tektutor.org $ oc get replicasets
NAME               DESIRED   CURRENT   READY   AGE
nginx-56fcf95486   3         3         0       23s
 jegan@tektutor.org $ oc get replicaset
NAME               DESIRED   CURRENT   READY   AGE
nginx-56fcf95486   3         3         0       24s
 jegan@tektutor.org $ oc get rs
NAME               DESIRED   CURRENT   READY   AGE
nginx-56fcf95486   3         3         0       26s
 jegan@tektutor.org $ oc get pods
NAME                     READY   STATUS             RESTARTS     AGE
nginx-56fcf95486-8mx9j   0/1     CrashLoopBackOff   1 (8s ago)   29s
nginx-56fcf95486-94vsp   0/1     CrashLoopBackOff   1 (9s ago)   29s
nginx-56fcf95486-ffj6q   0/1     CrashLoopBackOff   1 (8s ago)   29s
 jegan@tektutor.org $ oc get pod
NAME                     READY   STATUS             RESTARTS      AGE
nginx-56fcf95486-8mx9j   0/1     CrashLoopBackOff   1 (9s ago)    30s
nginx-56fcf95486-94vsp   0/1     CrashLoopBackOff   1 (10s ago)   30s
nginx-56fcf95486-ffj6q   0/1     CrashLoopBackOff   1 (9s ago)    30s
 jegan@tektutor.org $ oc get po
NAME                     READY   STATUS             RESTARTS      AGE
nginx-56fcf95486-8mx9j   0/1     CrashLoopBackOff   1 (11s ago)   32s
nginx-56fcf95486-94vsp   0/1     CrashLoopBackOff   1 (12s ago)   32s
nginx-56fcf95486-ffj6q   0/1     CrashLoopBackOff   1 (11s ago)   32s
 jegan@tektutor.org $ oc get po
NAME                     READY   STATUS             RESTARTS      AGE
nginx-56fcf95486-8mx9j   0/1     CrashLoopBackOff   3 (39s ago)   108s
nginx-56fcf95486-94vsp   0/1     CrashLoopBackOff   3 (43s ago)   108s
nginx-56fcf95486-ffj6q   0/1     CrashLoopBackOff   3 (35s ago)   108s
 jegan@tektutor.org $ oc get po
NAME                     READY   STATUS             RESTARTS      AGE
nginx-56fcf95486-8mx9j   0/1     Error              4 (60s ago)   2m9s
nginx-56fcf95486-94vsp   0/1     CrashLoopBackOff   4 (20s ago)   2m9s
nginx-56fcf95486-ffj6q   0/1     Error              4 (56s ago)   2m9s
 jegan@tektutor.org $ oc get po
NAME                     READY   STATUS             RESTARTS      AGE
nginx-56fcf95486-8mx9j   0/1     Error              4 (63s ago)   2m12s
nginx-56fcf95486-94vsp   0/1     CrashLoopBackOff   4 (23s ago)   2m12s
```

```
nginx-56fcf95486-ffj6q   0/1     Error             4 (59s ago)   2m12s
 jegan@tektutor.org $ oc get po
NAME                     READY   STATUS             RESTARTS      AGE
nginx-56fcf95486-8mx9j   0/1     Error              4 (64s ago)   2m13s
nginx-56fcf95486-94vsp   0/1     CrashLoopBackOff   4 (24s ago)   2m13s
nginx-56fcf95486-ffj6q   0/1     Error              4 (60s ago)   2m13s
 jegan@tektutor.org $ oc get po
NAME                     READY   STATUS             RESTARTS      AGE
nginx-56fcf95486-8mx9j   0/1     Error              4 (65s ago)   2m14s
nginx-56fcf95486-94vsp   0/1     CrashLoopBackOff   4 (25s ago)   2m14s
nginx-56fcf95486-ffj6q   0/1     Error              4 (61s ago)   2m14s
 jegan@tektutor.org $ oc get po
NAME                     READY   STATUS             RESTARTS      AGE
nginx-56fcf95486-8mx9j   0/1     Error              4 (65s ago)   2m14s
nginx-56fcf95486-94vsp   0/1     CrashLoopBackOff   4 (25s ago)   2m14s
nginx-56fcf95486-ffj6q   0/1     Error              4 (61s ago)   2m14s
 jegan@tektutor.org $ oc get po
NAME                     READY   STATUS             RESTARTS      AGE
nginx-56fcf95486-8mx9j   0/1     Error              4 (66s ago)   2m15s
nginx-56fcf95486-94vsp   0/1     CrashLoopBackOff   4 (26s ago)   2m15s
nginx-56fcf95486-ffj6q   0/1     Error              4 (62s ago)   2m15s
 jegan@tektutor.org $ oc logs nginx-56fcf95486-ffj6q
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to
perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-
default.sh
10-listen-on-ipv6-by-default.sh: info: can not modify
/etc/nginx/conf.d/default.conf (read-only file system?)
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-
resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-
templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-
processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/05/27 10:53:40 [warn] 1#1: the "user" directive makes sense only if
the master process runs with super-user privileges, ignored in
/etc/nginx/nginx.conf:2
nginx: [warn] the "user" directive makes sense only if the master process
runs with super-user privileges, ignored in /etc/nginx/nginx.conf:2
2024/05/27 10:53:40 [emerg] 1#1: mkdir() "/var/cache/nginx/client_temp"
failed (13: Permission denied)
nginx: [emerg] mkdir() "/var/cache/nginx/client_temp" failed (13:
Permission denied)
 jegan@tektutor.org  ~  oc whoami
system:admin
 jegan@tektutor.org $ oc get po
NAME                     READY   STATUS             RESTARTS        AGE
nginx-56fcf95486-8mx9j   0/1     CrashLoopBackOff   6 (2m59s ago)   9m19s
nginx-56fcf95486-94vsp   0/1     CrashLoopBackOff   6 (3m10s ago)   9m19s
nginx-56fcf95486-ffj6q   0/1     CrashLoopBackOff   6 (2m55s ago)   9m19s
 jegan@tektutor.org  ~  oc get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
```

```
nginx   0/3     3            0           11m
 jegan@tektutor.org $ oc describe deploy/nginx
Name:                   nginx
Namespace:              jegan
CreationTimestamp:      Mon, 27 May 2024 16:21:38 +0530
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=nginx
Replicas:               3 desired | 3 updated | 3 total | 0 available | 3
unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:latest
    Port:
    Host Port:
    Environment:
    Mounts:
  Volumes:
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      False   MinimumReplicasUnavailable
  Progressing    True    ReplicaSetUpdated
OldReplicaSets:
NewReplicaSet:   nginx-56fcf95486 (3/3 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  11m   deployment-controller  Scaled up replica
set nginx-56fcf95486 to 3
```

Lab - Deleting a deployment

```
oc delete deploy/nginx
oc get deploy,rs,po
```

Expected output

```
jegan@tektutor.org $ oc delete deploy/nginx
deployment.apps "nginx" deleted
jegan@tektutor.org $ oc get deploy,rs,po
No resources found in jegan namespace.
```

Lab - Creating nginx with bitnami image

```
oc project
oc create deployment nginx --image=bitnami/nginx:latest --replicas=3
oc get deploy,rs,po
oc get po -w
oc get po
```

Expected output

```
jegan@tektutor.org $ oc project
Using project "jegan" on server "https://api.ocp4.tektutor.org.labs:6443".

jegan@tektutor.org $ oc get deploy,rs,po
No resources found in jegan namespace.

jegan@tektutor.org $ oc create deployment nginx --
image=bitnami/nginx:latest --replicas=3
deployment.apps/nginx created

jegan@tektutor.org $ oc get deploy,rs,po
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx   0/3     3            0           5s

NAME                               DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-66c775969   3         3         0       5s

NAME                         READY   STATUS              RESTARTS   AGE
pod/nginx-66c775969-j7t8w   0/1     ContainerCreating   0          5s
pod/nginx-66c775969-knrhv   0/1     ContainerCreating   0          5s
pod/nginx-66c775969-xp48p   0/1     ContainerCreating   0          5s

jegan@tektutor.org $ oc get po -w
NAME                    READY   STATUS              RESTARTS   AGE
nginx-66c775969-j7t8w   1/1     Running             0          16s
nginx-66c775969-knrhv   0/1     ContainerCreating   0          16s
nginx-66c775969-xp48p   1/1     Running             0          16s
nginx-66c775969-knrhv   1/1     Running             0          17s
^C%

jegan@tektutor.org $ oc get po
NAME                    READY   STATUS    RESTARTS   AGE
nginx-66c775969-j7t8w   1/1     Running   0          23s
nginx-66c775969-knrhv   1/1     Running   0          23s
nginx-66c775969-xp48p   1/1     Running   0          23s
```

Lab - Finding the Pod IP address and the node where it is running

```
oc get po -o wide
```

Expected output

```
jegan@tektutor.org $ oc get po -o wide
NAME                    READY   STATUS    RESTARTS   AGE     IP
NODE                              NOMINATED NODE   READINESS GATES
nginx-66c775969-j7t8w   1/1     Running   0          3m27s   10.129.0.250
master-2.ocp4.tektutor.org.labs
nginx-66c775969-knrhv   1/1     Running   0          3m27s   10.128.3.94
worker-2.ocp4.tektutor.org.labs
nginx-66c775969-xp48p   1/1     Running   0          3m27s   10.131.1.17
worker-1.ocp4.tektutor.org.labs
```

Lab - Port-forwarding for local access - generally used by developers for testing purpose(Not for production)

In the below command, port 8080 is the port where nginx web server is listening inside the Pod container. Port 9080 is the port on the local machine. When request comes to the port 9080, it is forwarded on the port 8080 on the Pod container.

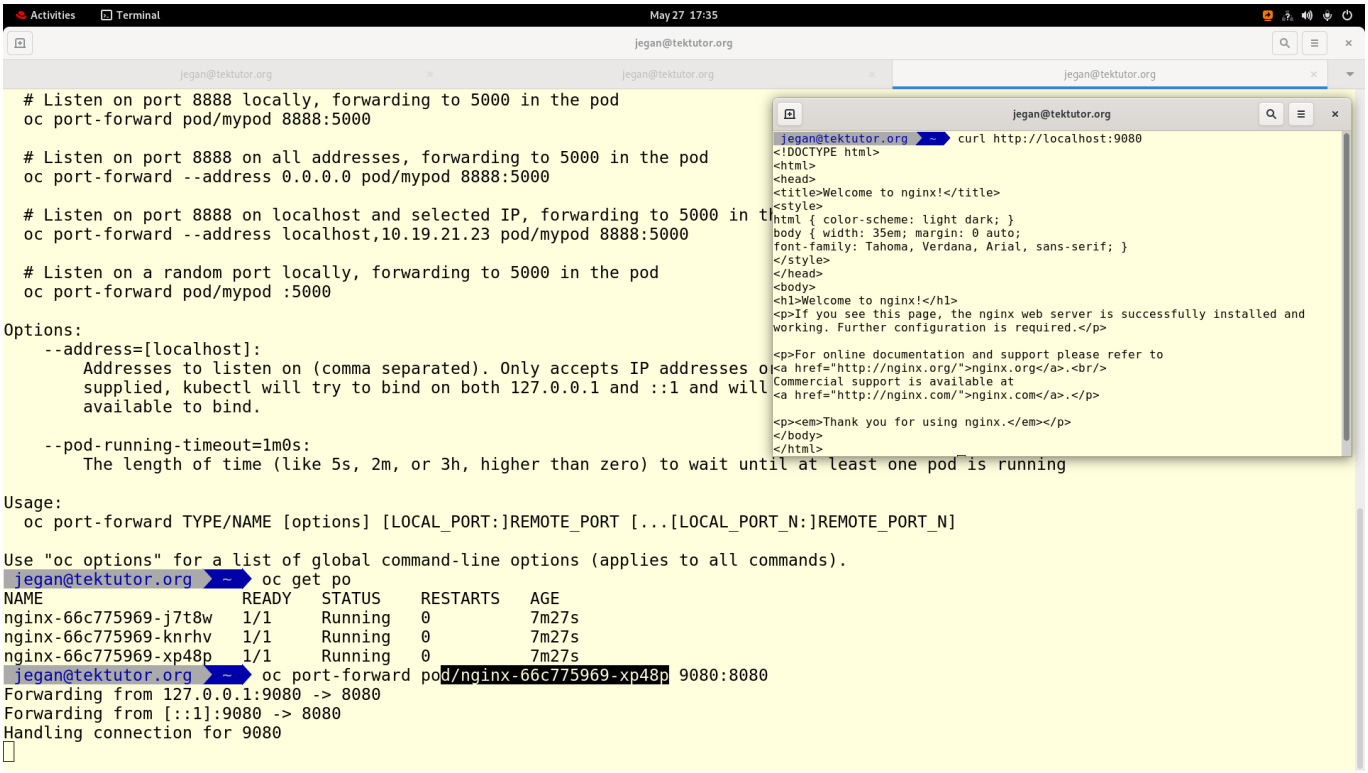
The port 9080 can be changed to any port of your choice that is available on the localhost.

```
oc get po
oc port-forward pod/nginx-66c775969-xp48p 9080:8080
```

Accessing the web page from another terminal window

```
curl http://localhost:9080
```

Expected output



Lab - Creating an internal service for nginx deployment

```
oc get svc
oc get deploy,rs,po
oc expose deploy/nginx --type=ClusterIP --port=8080
oc get services
oc get service
oc get svc
oc describe svc/nginx
```

Expected output