Day 1

Please share your first day feedback below

```
https://survey.zohopublic.com/zs/LjCU4j
```

SOLID - Design Principles

```
S - Single Responsibility Principle (SRP)
O - Open Closed Principle (OCP)
L - Liskov Substitution Principle (LSP)
I - Interface Seggregation Principle
D - Dependency Inject or Dependency Inversion or Inversion of Control (IOC)
```

Single Responsibility Principle (SRP)

- One Class should represent one Object/Entity
- One Component should represent one Object
- You are developed a Microservice in let's say springboot framework as a container
- the microservice could write some logs in a file
-

What is Boot Loader ?

- When we boot Laptop/Desktop/Workstation/Server, the BIOS it will perform POST (Power On Self Test)
- Once the BIOS POST operations are completed, the BIOS will instruct the CPU to load the Boot Loader utility that is installed on the Master Boot Record (MBR - Sector 0 Byte 0 on the Hard disk).
- It is tiny appliction which has to fit within 512 bytes
- Boot loader is the one which scans your hard disks looking for Operating systems installed and boots the OS
- Every OS whether it is Windows/Linux it will install the Boot Loader at the end of the OS installtion
- In case the MBR is corrupted then your OS will not boot at all
- Examples
    - LILO (Linux Loader)
    - GRUB 2 - Stable boot loader generally used in all Linux Distributions
    - BootCamp - commercial boot loader used on the Mac Laptops/Desktops

What is Dual/Multi-booting?

- In case, you have installed 2 or more OS on the same laptop/desktop, then the boot loader utility gives a menu asking which OS you want to boot into
- In case you have multiple OS, only one OS can be active at any point of time

- If you wish to boot into the other OS, you need to shutdown the currently running OS

What is Hypervisor?

- is virtualization technology
- we can run multiple OS on the same laptop/desktop/workstation/server
- many OS can be actively running at the same time on the same machine
- AMD Processor
    - the virtualization feature set is called AMD-V
- Intel Processor
    - the virtualization feature set is called VT-X
- need hardware and BIOS support to install Hypervisor(Virtualization) software
- each Virtual Machine represents 1 OS
- In case laption/desktop/workstation, we generally install Type 2 Hypervisor
- Type 2 Hypervisors require a Host OS installed on the machine
- The Virtualization software is installe on top of the Host OS
- The additional OS that we install are called Guest OS, which runs within a Virtual Machine
- In case of Servers, we can install Type 1 Hypervisor a.k.a Bare metal hypervisors
- Bare metal hypervisors doesn't require a OS to be installed to create virtual machine
- Type 1 Hypervisor Example
    - VMWare vSphere/vCenter
- Type 2 Hypervisors Examples VMWare (Paid - Commercial Product)
    - Fusion (installed top of Mac OS-X)
    - Workstation (installed on top of Windows/Linux) Oracle Virtualbox (installed on top of Mac/Windows/Linux - Free but not opensource)
- KVM - Free & Opensource Hypervisor works on all Linux Distributions
- This type of virtualization is called heavy-weight Virtualization
    - the reason for saying heavy-weight is because each Virtual Machines has to be allocated with dedicated hardware resources
        - CPU Cores (virtual cores)
        - RAM - Actual size
        - Network Card (virtual)
        - Graphics Card (virtual)

What is the deciding factor that limits the maximum number of Virtual Machines we can create on a machine?

- Processor - supports multiple CPU Cores
- depends on how many virtual CPU cores your system supports
- modern Processor with 4 Physical Cores are seen as 8 Virtual Cores by the Virtualization software
- Which mean on a laptop with 4 Physical CPU Cores, you can run 1 Host OS and 7 Virtual Machine

Docker Overview

- is an application virtualization technology
- unlinke the Hypervisor technology, this is light weight virtualization
- why light-weight?
    - because container's don't use dedicated hardware resources

    - all the containers that runs on the same host machine shares the hardwares on the host OS
    - containers are not Operating System
    - containers doesn't have their own OS Kernel
- containers get's their own IP Address
- each container represents one application or one application component (backend, frontend, web/app server, db server, etc)
- each container runs in a separate namespace
- containers are otherwise a normal application process
- Docker comes 2 flavours
    1. Docker Community Edition - Docker CE
    2. Docker Enterprise Edition - Docker EE
- Docker is developed in Go language by the company Docker Inc
- follows client/server architecture
- the server runs in the OS Kernel context (administrator), hence most container we create we will gain admin access within the containers, even in the case the user who created the container is a non-admin user

Podman Overview

- Podman is alternate to Docker
- Podman is also opensource maintained by Community headed by Red Hat
- stand-alone tool unlike Docker
- supports creating root-less container i.e running application as normal user (non-admin users)

What is a Container Image?

- is a binary file that has software pre-installed in it
- it comes with package manager like apt(apt-get),yum,rpm,dnf,etc.,
- containers are created using Container Image
- Example
    - Windows 11 ISO Disk Image,
    - similary we have ubuntu:16.04 container image
    - similarly we have nginx:1.18 is the Container Image that comes with nginx web server v1.18 pre-installed in it
- When we create a nginx container using nginx:1.18 container image, the container by default runs the nginx web server within it
- using container images, we can create any number of container
- containers are the running instances of the container image
- containers get an IP address

Container Orchestration Overview

- though containerized application workloads can be manually managed, in real-world no organization manages containers manually
- generally containerized applications are managed by Container Orchestration Platforms
- Examples
    - Docker SWARM
    - Google Kubernetes

    - AWS EKS - Elastic Kubernetes Service (Managed K8s Cluster from Amazon AWS)
    - Azure AKS - Azure Kubernetes Service (Managed K8s Cluster from Microsoft Azure)
    - Rancher Kubernetes (Kubernetes + Rancher Webconsole)
    - Red Hat OpenShift
    - AWS ROSA - Managed Red Hat OpenShift Cluster from Amazon AWS
    - Azure ARO - Managed Red Hat OpenShift cluster from Microsoft Azure

Docker SWARM

- Docker Inc's native Orchestration Platform
- supports only Docker containerized application workloads
- easy to install on regular laptops/desktops
- light-weight
- good for learning purpose, dev/qa environment
- not used in production

Google Kubernetes

- opensource and free Container Orchestration Platform developed and maintained by Opensource community led by Google
- supports many different Container Runtimes - e.g Docker, containerd, Podman
- also supports extending Kubernetes API by adding your own Custom Resources and Custom Controller
- supports packaging Custom Resources and Custom Controllers as Kubernetes Operators to extend Custom Kubernetes additional features
- generally Command-line only
- there is minimal Kubernetes Dashboard (webconsole) which doesn't support user management hence result in security issue, so normally organizations disable the Kubernetes dashboard
- Kubernetes doesn't support internal container registry out of the box, but can be configured to use external container registry

Red Hat OpenShift

- Red Hat's Kubernetes distribution
- Kubernetes + Many additional features developed on top of opensource Kubernetes
- all the features supported in Kubernetes also works in OpenShift
- OpenShift added many additional features on top of Kubernetes
- supports CLI and Web Console(GUI)
- comes with Internal Openshift Container Registry
- supports CI/CD within Openshift
- supports additional Custom Resources in Openshift
    - DeploymentConfig
    - Route
- self-healing platform
- supports in-built monitoring features
- supports in-built load-balancing
- supports high-availability for your applications deployed within Kubernetes/Openshift

- supports exposing your application only within the openshift cluster or outside the openshift cluster via services
- supports rolling update
- supports scaling up/down your application instance counts

Commonly Used OpenShift resources

- Pod
- ReplicaSet
- Deployment

What is OpenShift Cluster?

- OpenShift cluster is a collection of many machines
- each machine is referred as node in the Kubernetes/Openshift
- each node is either of type of Master or Worker
- Master nodes runs the Control Plane components
    - API Server
    - etcd database
    - scheduler
    - controller managers
- Worker nodes runs the user containerized application workloads

API Server

- this is a collection of all the Kubernetes/Openshift features in the form of REST APIs
- the client tools like oc/kubectl/kn/odo/helm will communicate to the API Server by making REST calls
- API Server saves the cluster status and application status in the etcd key/value database
- API Server is the only component that has access to the etcd databaase
- all the components in openshift will only communicate to API Server
- no two components are allowed to talk to each other directly, every communication happens only via API Server
- API Server sends broadcasting events each time
    - new record is inserted into the etcd database
    - an existing is modified
    - an existing is deleted

etcd database

- is an opensource key-value database used in Kubernetes/openshift
- can be used outside the scope of kubernetes/openshift as well
- normally works as a cluser of many etcd db nodes
- data gets synchronized when many etcd db servers works as a cluster
- one of the reasons why openshift recommends minimum 3 masters is due to the reason etcd requires/recommends a minimum of 3 nodes in the cluster

Scheduler

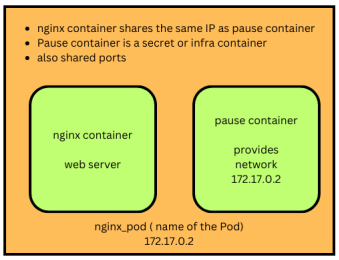
- this is the component that is responsible to find a healthy node where a new Pod can be deployed
- scheduler shares the scheduling recommendation about each Pod by making REST calls
- API Server sends broadcasting events whenever a new Pod is created in the etcd database
- API Server sends broadcasting events whenever a Pod is deleted in the etcd database
- API Server sends broadcasting events whenever a Pod status changes (for example -crashloop)

Controller Managers

- this is a collection of many Controllers
- For example
    - Deployment Controller
    - ReplicaSet Controller
    - StatefulSet Controller
    - DaemonSet Controller
    - Job Controller
    - CronJob Controller
    - Endpoint Controller
    - Storage Controller

What is a Pod?

- a collection of related containers
- each container represents one application or one application component
- Pod is the small unit that can be deployed in Kubernetes/OpenShift
- one application per Pod is the recommended best practice
- IP address is assigned on the Pod level, not on the container level
- hence, every container that is part of the same Pod shares the same IP address
- Pod is yaml/json resource that lives in etcd database
- the containers associated to the Pod runs in any nodes within the Openshift cluster



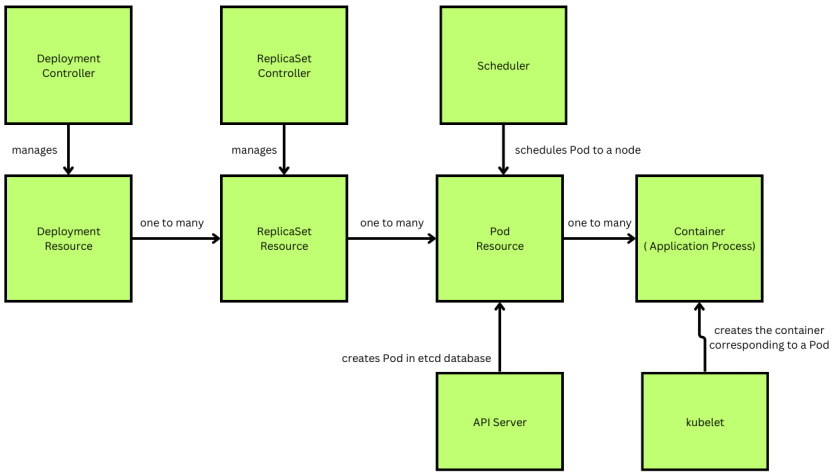
Pod

What is ReplicaSet?

- Replicaset is a resource that is stored and maintained in etcd database by API Server(Control Plane)
- ReplicaSet tells what is the desired number of Pods that be running at any point of time within Kubernetes/OpenShift
- ReplicaSet is managed by a controller called ReplicaSet Controller
- Scale up/down is suppored by ReplicaSet Controller

What is Deployment

- Deployment is a resourced stored and maintained in etcd database by API Server
- for each application we deploy in Openshift, a deployment will be created
- Deployment is managed by a controller called Deployment Controller
- the deploy has
    - an unique name
    - desired number of Pods that should be running
    - container image that must be used to deploy the Pod containers
- Rolling update is supported by Deployment Controller



What is a Controller?

- is a application that runs in a infinite loop
- it keeps looking for specific type of resources created within Openshift cluster in any namespace/project
- it has some special permissions to monitor certain resources in any project/namespace
- For example
    - Deployment Controller can detect Deployment resources created/edited/deleted/scaled up/down in any namespace/project
    - ReplicaSet Controller can detect ReplicaSet resources created in any namespace/project

What is a Master Node in Kubernetes/Openshift?

- In master node, control-plane components will be running

- Control Plane
- Control Planes components they provide the orchestration features
- Control Planes monitors, heals, manages the user-application deployed in the openshift
- wherever control plane components are running they are called master nodes
- In normal configurations, the user application won't be running/scheduled into master nodes
- In special cases, we can configure the master nodes to accept user application getting deployed into master nodes by removing the tains(conditions/restrictions)
- In our lab setup, user application will be deployed onto master as well as worker nodes
- master nodes are also called as controller nodes

What is a Worker Node in Kubernetes/Openshift?

- Worker Nodes is where user applications will be deployed
- there can any number of worker nodes
- worker nodes are also called as compute nodes

Info - OpenShift states

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.8/html/nodes/working-with-clusters#nodes-containers-events-list_nodes-containers-events

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

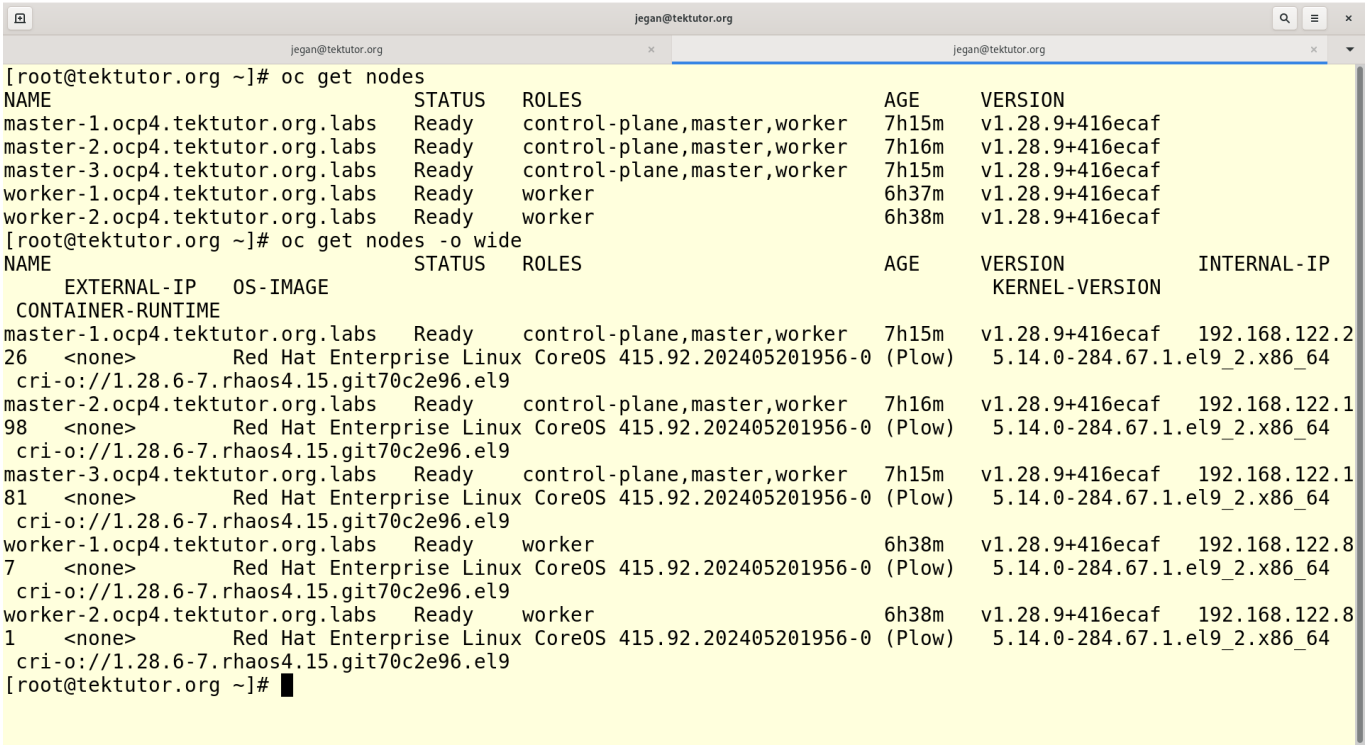
Info - Container Lifecycle

- 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

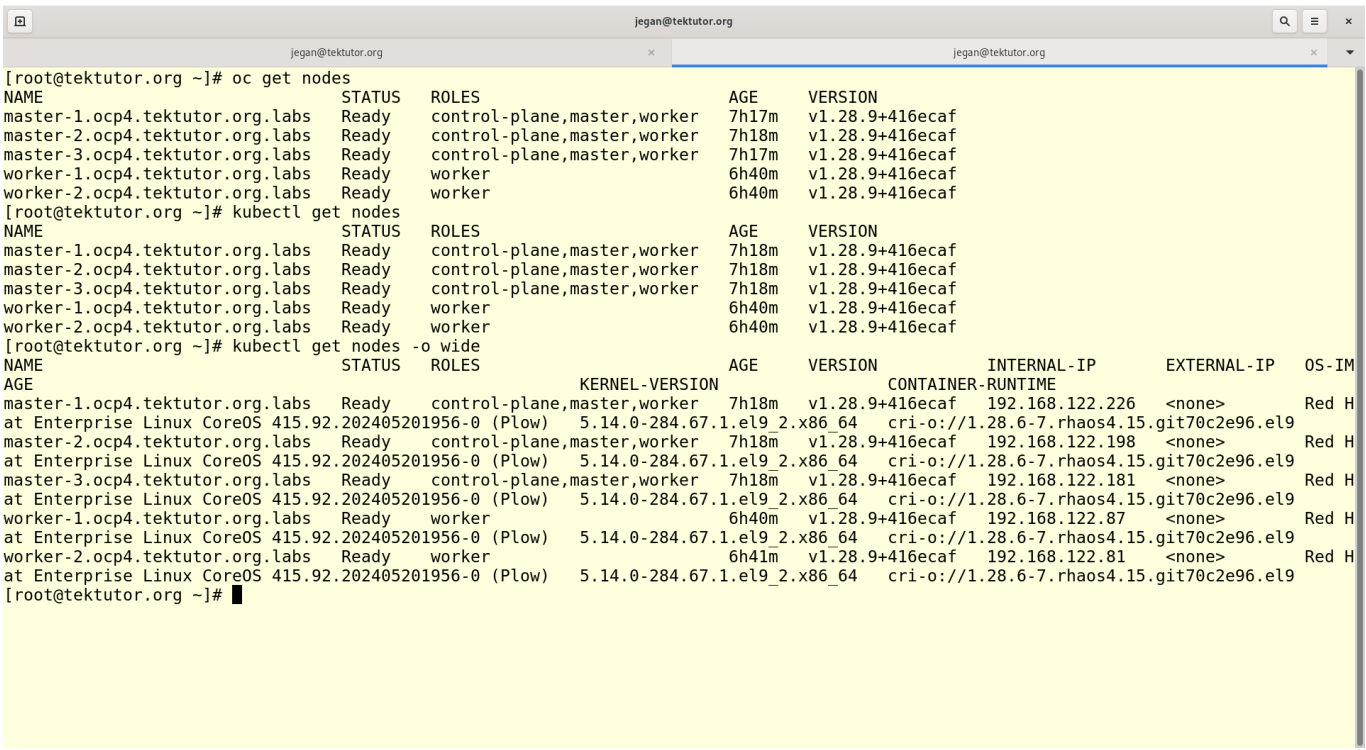
Lab - Listing the nodes in the Openshift cluster

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

Expected output



```
jegan@tektutor.org
jegan@tektutor.org  jegan@tektutor.org
[root@tektutor.org ~]# oc get nodes
NAME                              STATUS   ROLES                         AGE     VERSION
master-1.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h15m   v1.28.9+416ecaf
master-2.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h16m   v1.28.9+416ecaf
master-3.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h15m   v1.28.9+416ecaf
worker-1.ocp4.tektutor.org.labs   Ready    worker                        6h37m   v1.28.9+416ecaf
worker-2.ocp4.tektutor.org.labs   Ready    worker                        6h38m   v1.28.9+416ecaf
[root@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   7h15m   v1.28.9+416ecaf   192.168.122.2
26   <none>        Red Hat Enterprise Linux CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.x86_64
 cri-o://1.28.6-7.rhaos4.15.git70c2e96.el9
master-2.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h16m   v1.28.9+416ecaf   192.168.122.1
98   <none>        Red Hat Enterprise Linux CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.x86_64
 cri-o://1.28.6-7.rhaos4.15.git70c2e96.el9
master-3.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h15m   v1.28.9+416ecaf   192.168.122.1
81   <none>        Red Hat Enterprise Linux CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.x86_64
 cri-o://1.28.6-7.rhaos4.15.git70c2e96.el9
worker-1.ocp4.tektutor.org.labs   Ready    worker                        6h38m   v1.28.9+416ecaf   192.168.122.8
7    <none>        Red Hat Enterprise Linux CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.x86_64
 cri-o://1.28.6-7.rhaos4.15.git70c2e96.el9
worker-2.ocp4.tektutor.org.labs   Ready    worker                        6h38m   v1.28.9+416ecaf   192.168.122.8
1    <none>        Red Hat Enterprise Linux CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.x86_64
 cri-o://1.28.6-7.rhaos4.15.git70c2e96.el9
[root@tektutor.org ~]#
```



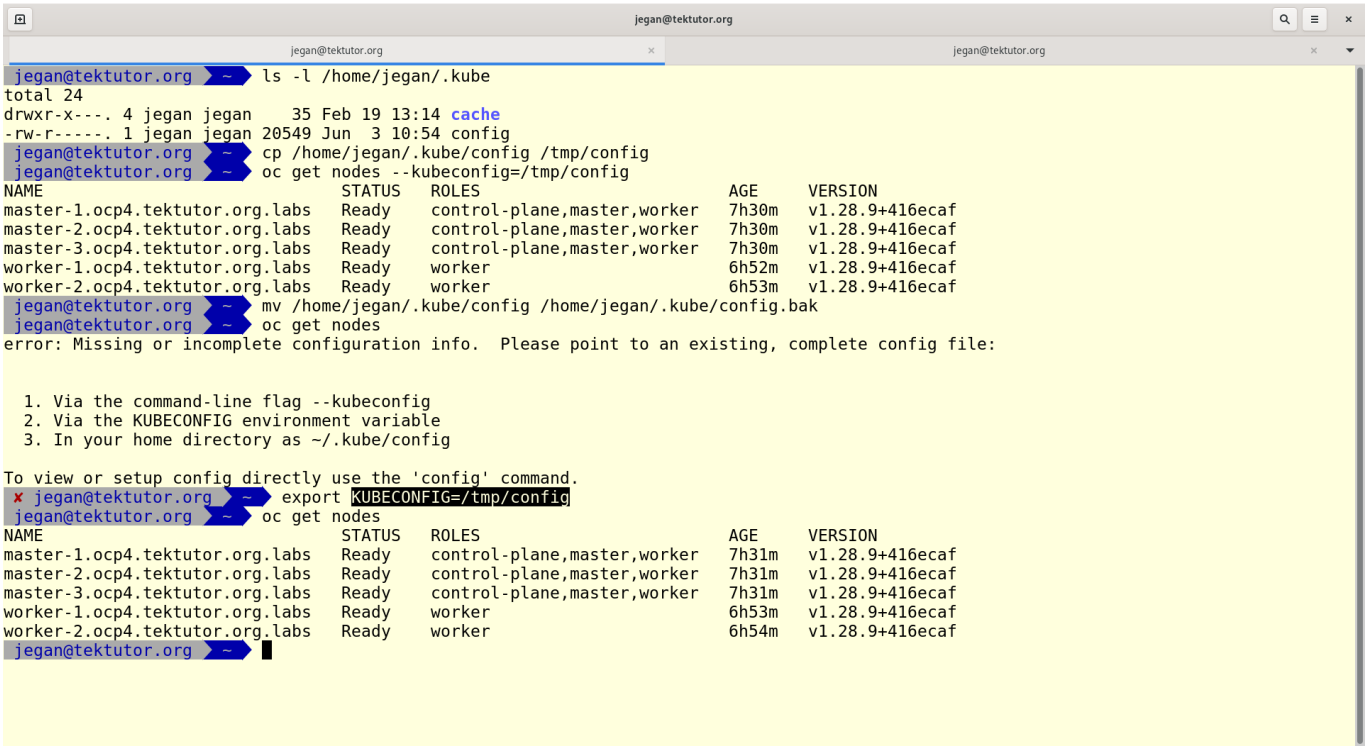
```
jegan@tektutor.org
jegan@tektutor.org  jegan@tektutor.org
[root@tektutor.org ~]# oc get nodes
NAME                              STATUS   ROLES                         AGE     VERSION
master-1.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h17m   v1.28.9+416ecaf
master-2.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h18m   v1.28.9+416ecaf
master-3.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h17m   v1.28.9+416ecaf
worker-1.ocp4.tektutor.org.labs   Ready    worker                        6h40m   v1.28.9+416ecaf
worker-2.ocp4.tektutor.org.labs   Ready    worker                        6h40m   v1.28.9+416ecaf
[root@tektutor.org ~]# kubectl get nodes
NAME                              STATUS   ROLES                         AGE     VERSION
master-1.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h18m   v1.28.9+416ecaf
master-2.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h18m   v1.28.9+416ecaf
master-3.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h18m   v1.28.9+416ecaf
worker-1.ocp4.tektutor.org.labs   Ready    worker                        6h40m   v1.28.9+416ecaf
worker-2.ocp4.tektutor.org.labs   Ready    worker                        6h40m   v1.28.9+416ecaf
[root@tektutor.org ~]# kubectl get nodes -o wide
NAME                              STATUS   ROLES                         AGE     VERSION           INTERNAL-IP       EXTERNAL-IP   OS-IM
AGE                                KERNEL-VERSION                CONTAINER-RUNTIME
master-1.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h18m   v1.28.9+416ecaf   192.168.122.226   <none>        Red H
at Enterprise Linux CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.x86_64   cri-o://1.28.6-7.rhaos4.15.git70c2e96.el9
master-2.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h18m   v1.28.9+416ecaf   192.168.122.198   <none>        Red H
at Enterprise Linux CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.x86_64   cri-o://1.28.6-7.rhaos4.15.git70c2e96.el9
master-3.ocp4.tektutor.org.labs   Ready    control-plane,master,worker   7h18m   v1.28.9+416ecaf   192.168.122.181   <none>        Red H
at Enterprise Linux CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.x86_64   cri-o://1.28.6-7.rhaos4.15.git70c2e96.el9
worker-1.ocp4.tektutor.org.labs   Ready    worker                        6h40m   v1.28.9+416ecaf   192.168.122.87    <none>        Red H
at Enterprise Linux CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.x86_64   cri-o://1.28.6-7.rhaos4.15.git70c2e96.el9
worker-2.ocp4.tektutor.org.labs   Ready    worker                        6h41m   v1.28.9+416ecaf   192.168.122.81    <none>        Red H
at Enterprise Linux CoreOS 415.92.202405201956-0 (Plow)   5.14.0-284.67.1.el9_2.x86_64   cri-o://1.28.6-7.rhaos4.15.git70c2e96.el9
[root@tektutor.org ~]#
```

Info - How the oc or kubectl learns the connection details to openshift cluster

```
- In the user home directory, there is hidden .kube folder which has the
config file
- In my case, /home/jegan/.kube/config is the location oc or kubectl will
first search for the config file
- It is also possible we could export an environment variable KUBECONFIG to
point to the location of the config file
```

```
- An alternate approach is to supply a switch in the command-line to point
the config file location
```

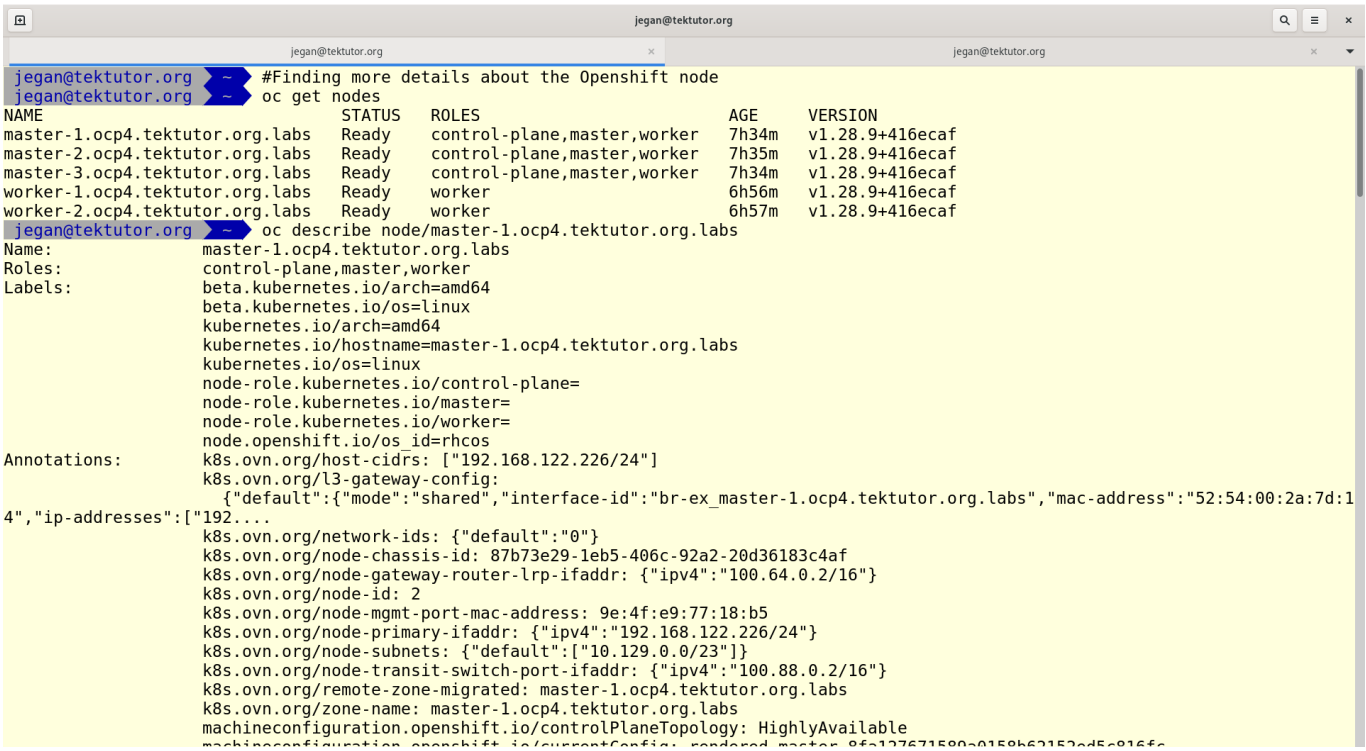
Expected output



Lab - Finding more details about an openshift node

```
oc get nodes
oc describe node/master-1.ocp4.tektutor.org.labs
oc describe node master-1.ocp4.tektutor.org.labs
```

Expected output



Lab - Editing a node - don't modify anything 😀

```
oc get nodes
oc edit node/worker-1.ocp4.tektutor.org.labs
```

Info - Openshift Projects

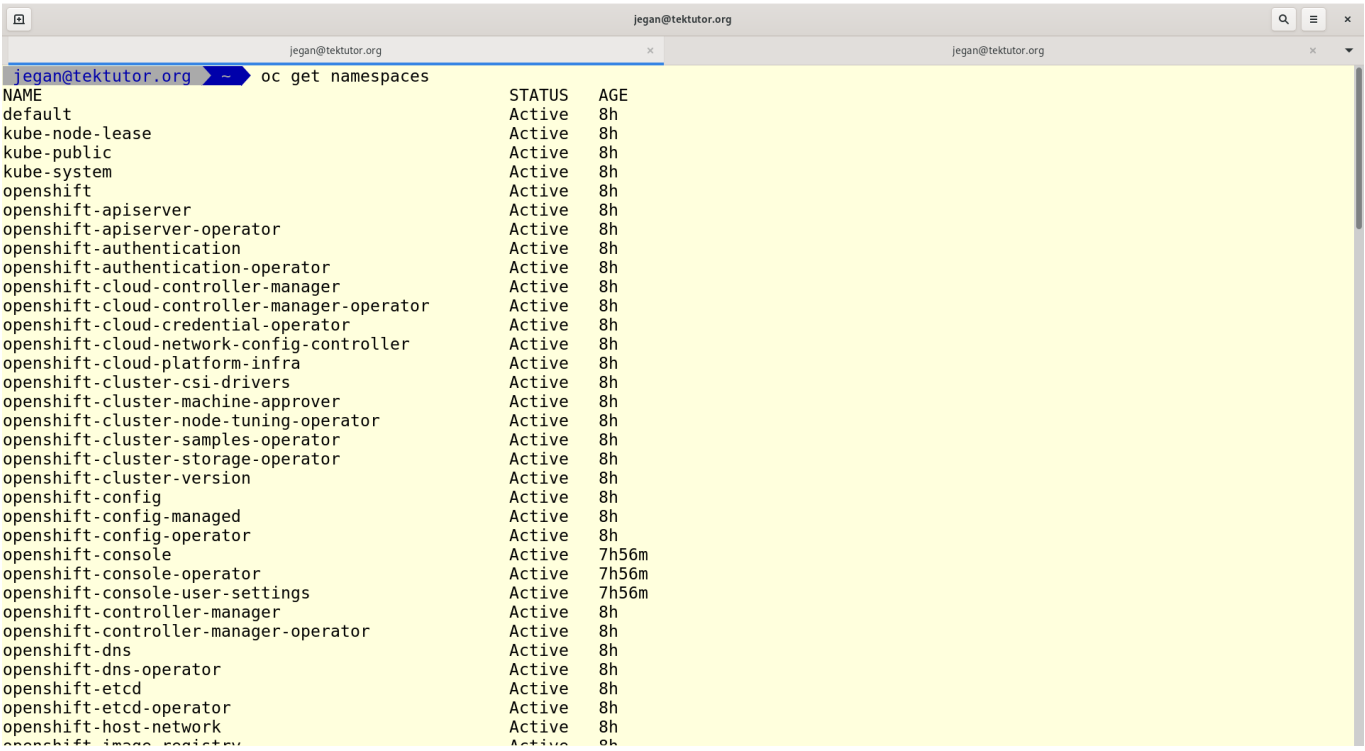
```
- In order to seggreate the application deployments made by different
teams, namespaces or projects are created
- For each team, atleast one project is created, so that multiple engineers
can work together
- many applications can deployed into a single openshift project
- Administrators can give access to projects the developers are working in,
while denying access to other users
- it is always recommended to create a separate before we deploy our
applications
- training specific request - each user can create one project in your name
```

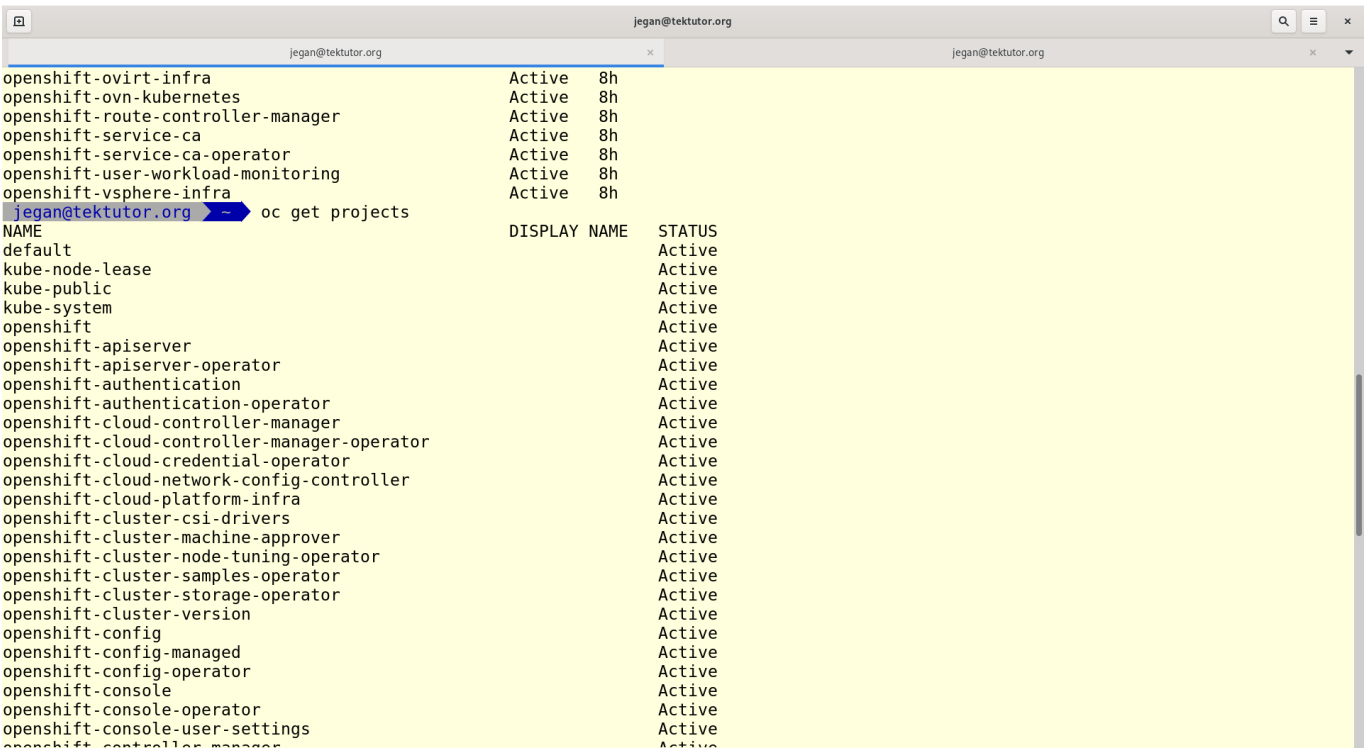
Lab - Listing the projects in openshift

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

```
oc get projects
oc get project
```

Expected output

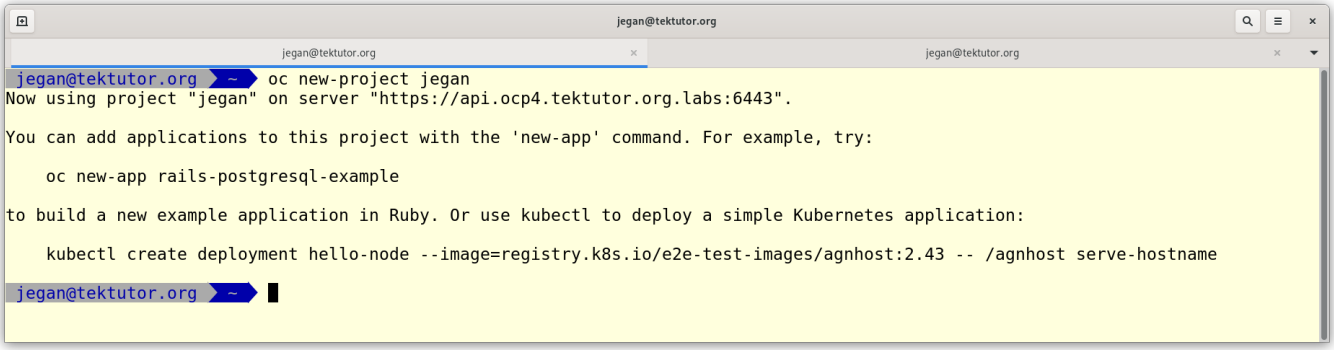




Lab - Creating a new project

```
oc new-project jegan
```

Expected output

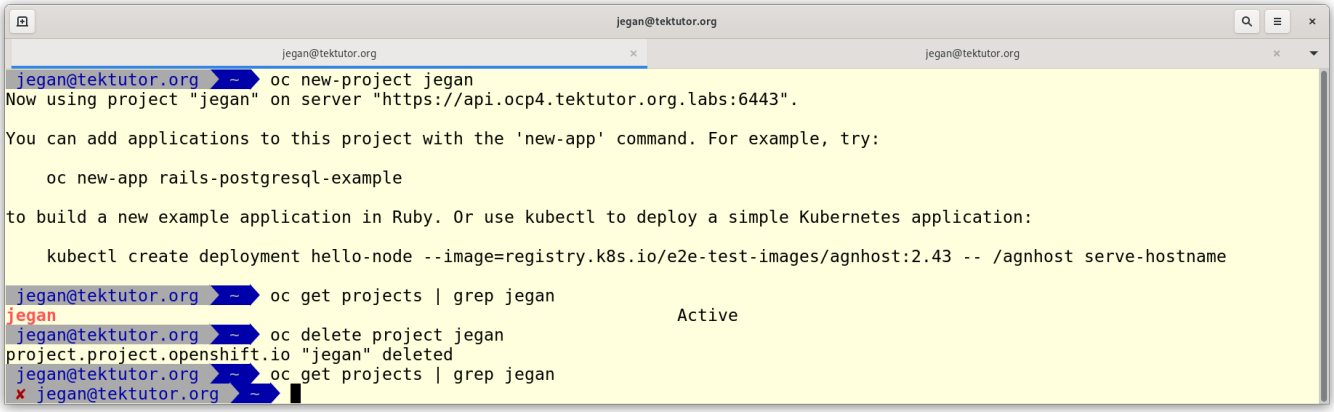


Lab - Deleting a project

When we delete a project, it deletes all the resources under the project. There is no way to recover/restore it once deleted.

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

Expected output



Lab - Switching between projects and finding the current active project

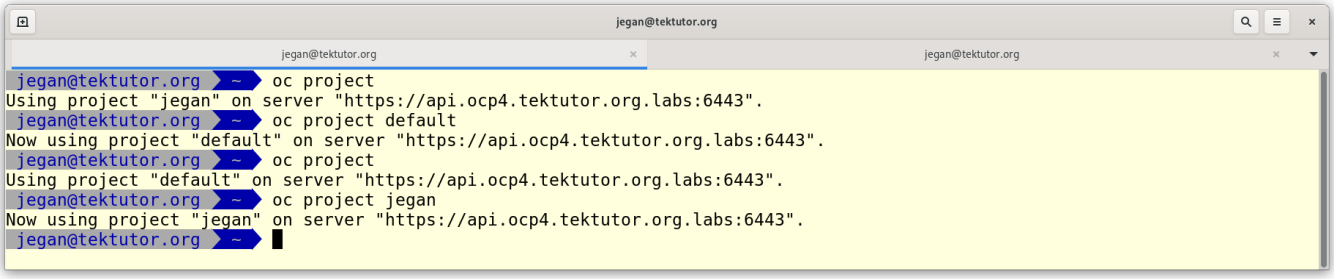
To find the currently active project

```
oc project
```

To switch between projects

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

Expected output



Lab - Deploying nginx web server into openshift in imperative style

The below command will create 3 types of resources in openshift.

1. Deployment by name nginx
2. ReplicaSet by nginx-
3. Pod (single Pod)

```
oc project
oc create deployment nginx --image=nginx:latest
```

Listing the deployments within your project.

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

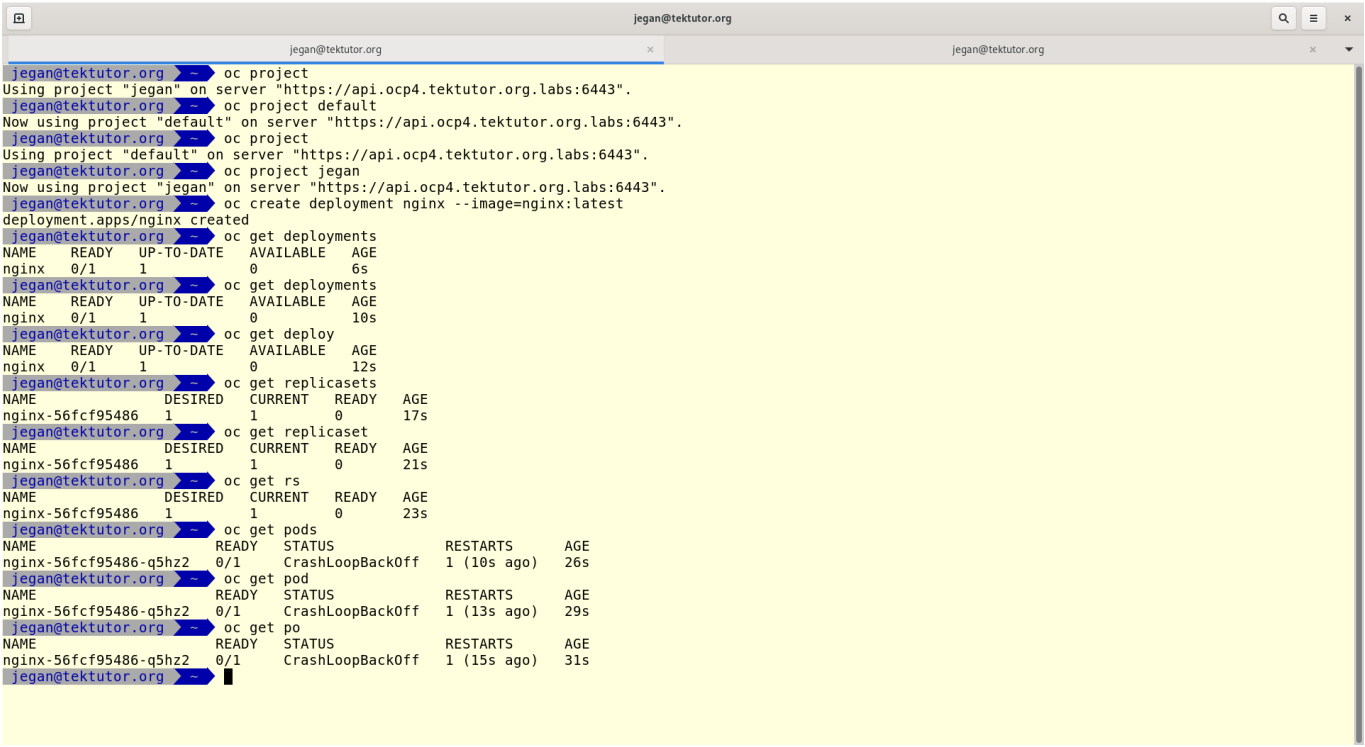
Listing the replicasets with within your project.

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

Listing the pods. Pod is one of the resource types supported by Kubernertes/Openshift. Within Pod, nginx container will be running.

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

Expected output



Things to note

- Deployment is one of the resource types supported by Kubernertes/Openshift.
- Deployment resource is stored inside the etcd database as a YAML document.
- ReplicaSet is one of the resource types supported by Kubernertes/Openshift.
- ReplicaSet resource is stored inside the etcd database as a YAML document.
- Pod resource is stored inside the etcd database as a YAML document.

Troubleshooting - understanding why the nginx Pod is crashing

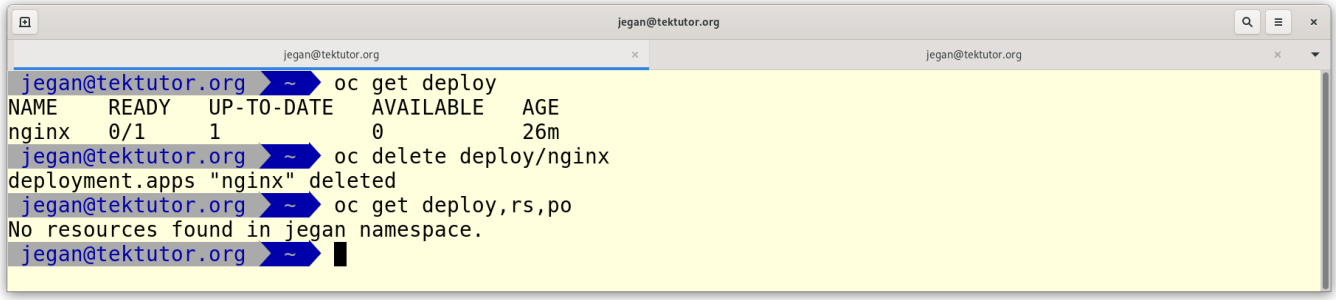
```
- if you notice the deployment command, we are using nginx:latest image
which works perfectly in kubernetes but not in openshift
- in openshift, the RedHat Enterprise Linux Core OS, won't allow normal
applications to do stuffs as administrators
- the nginx Pod container, seem to attempt to create a folder under /var
```

Lab - Deleting the nginx deployment

Deleting the nginx deploy, will also delete the replicaset and the respective pods. Once deleted, it can't be restored/recovered.

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

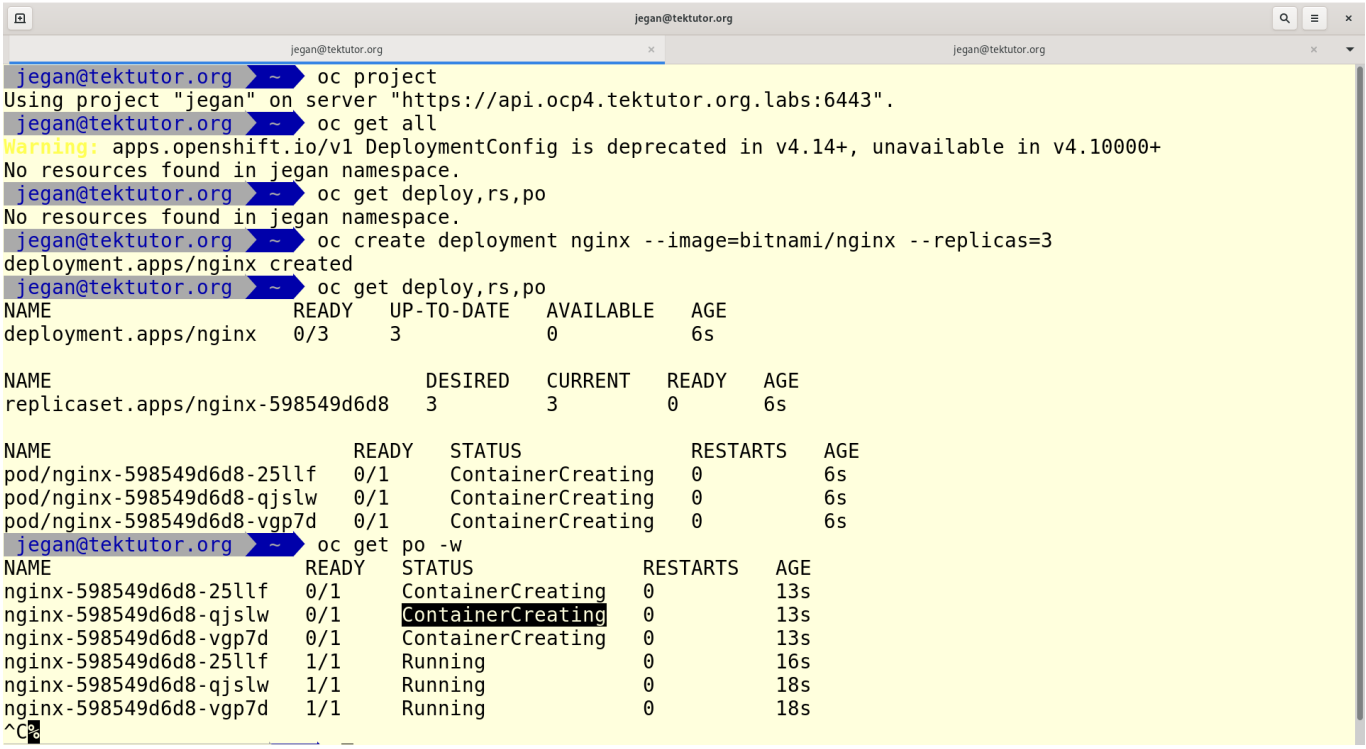
Expected output



Lab - Deploy nginx with bitnami image which is root-less

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

Expected output



Lab - Port-forwarding for developer testing

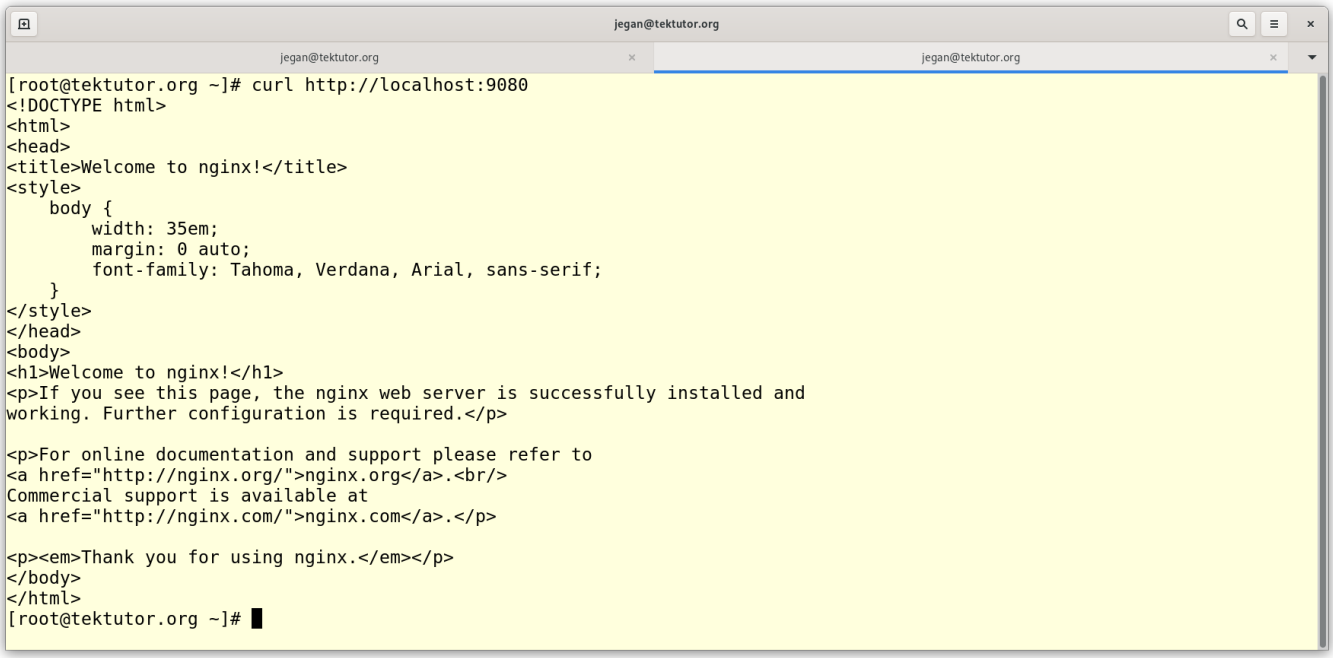
In the below command, the port 9080 is exposed on the local machine, while 8080 is the port used with the container by nginx web server.

```
oc get po
oc scale deploy/nginx --replicas=3
oc get po
oc port-forward pod/nginx-566b5879cb-566v9 9080:8080
```

From another terminal, we can access the web page served by the nginx pod container

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

Expected output



To stop the port-forward, you need to press Ctrl + C. Once stopped the page won't be accessible. This technique must be used only for testing purpose and not used in production.

For production use, we need to use services and routes.

Lab - Creating an internal service for nginx deployment

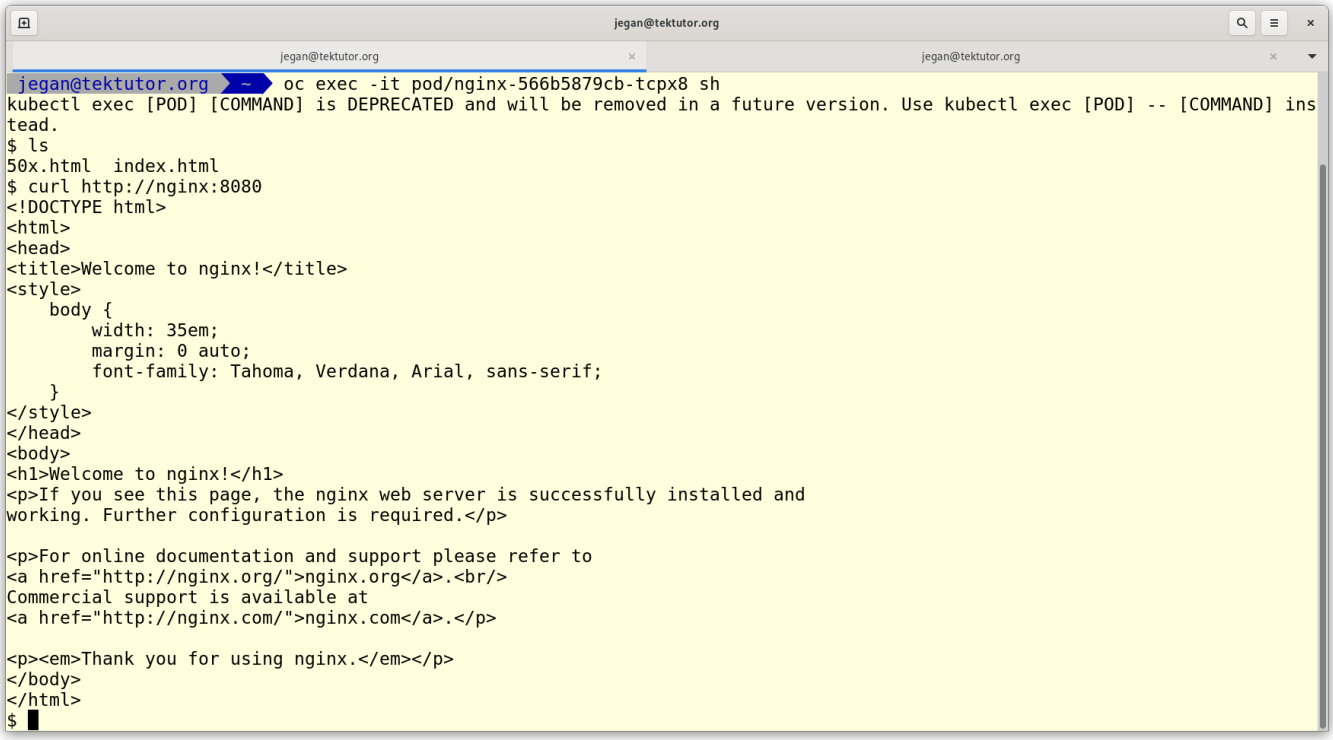
```
oc get deploy,po
oc expose deploy/nginx --type=ClusterIP --port=8080
oc get services
oc get service
oc get svc
oc describe svc/nginx
```

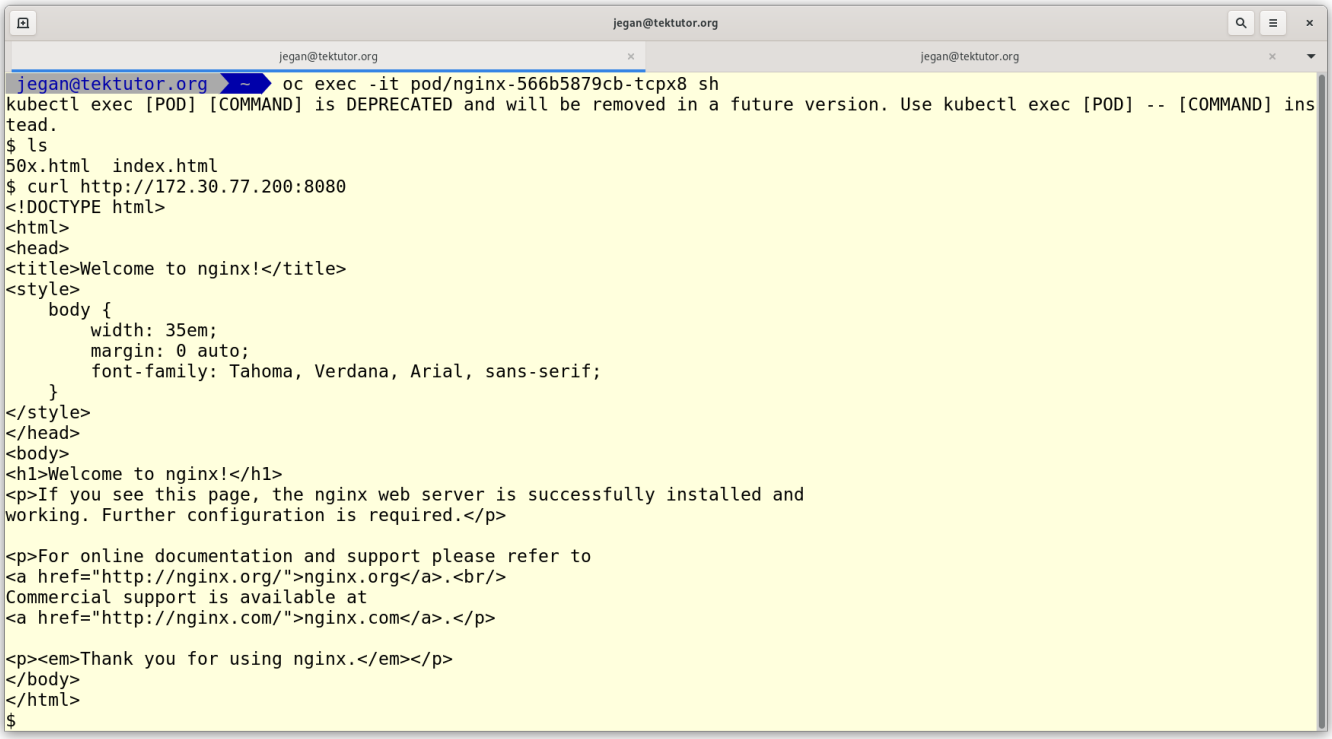
In order to access the service, we need to get inside some pod shell

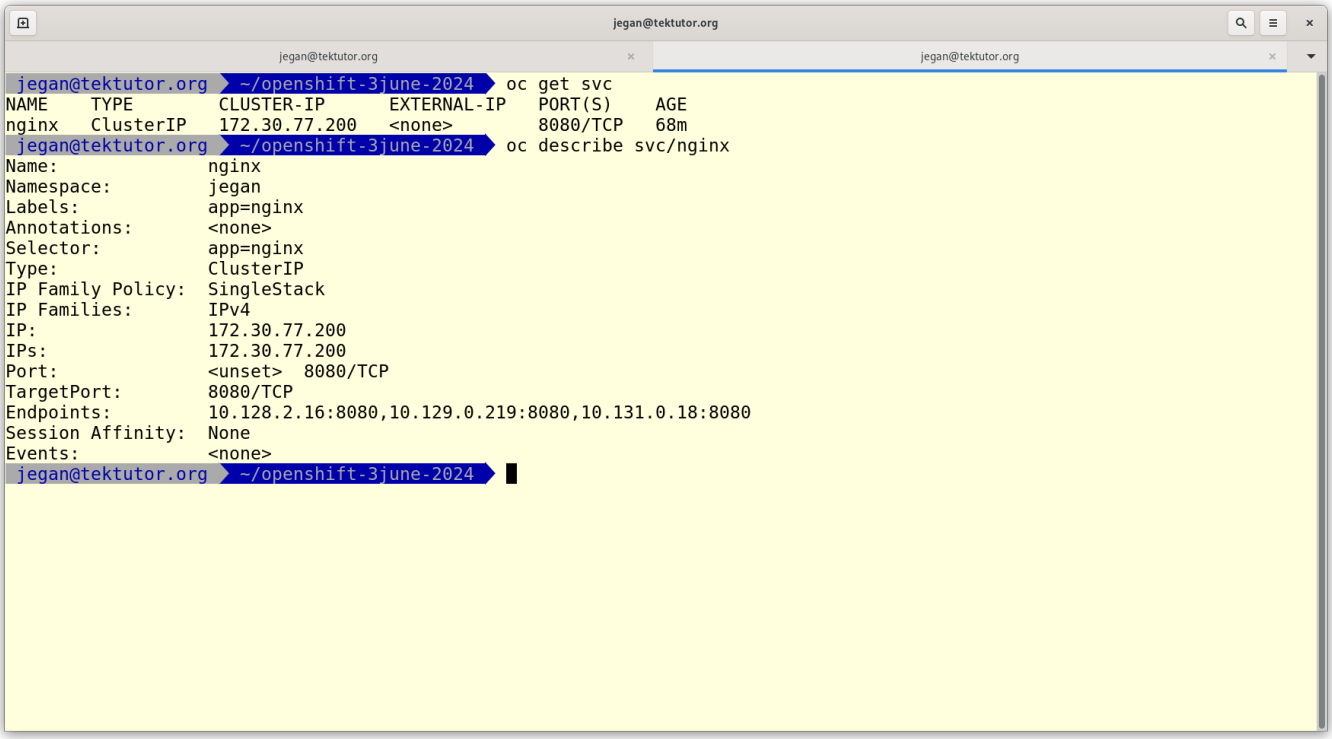
```
oc exec -it pod/nginx-566b5879cb-tcpx8 sh
ls
curl http://nginx:8080
curl http://172.30.77.200:8080
cat /etc/resolv.conf
```

In the above curl, the IP address 172.30.77.200 is the nginx service IP in my case. The nginx in the first curl command refers to the nginx service name. The default DNS service(172.30.0.10) resolves nginx service name to the IP address 172.30.77.200, aka Service Discovery.

Expected output







Lab - Creating an external nodeport service

First we need to delete the clusterip service we created earlier

```
oc get svc
oc delete svc/nginx
```

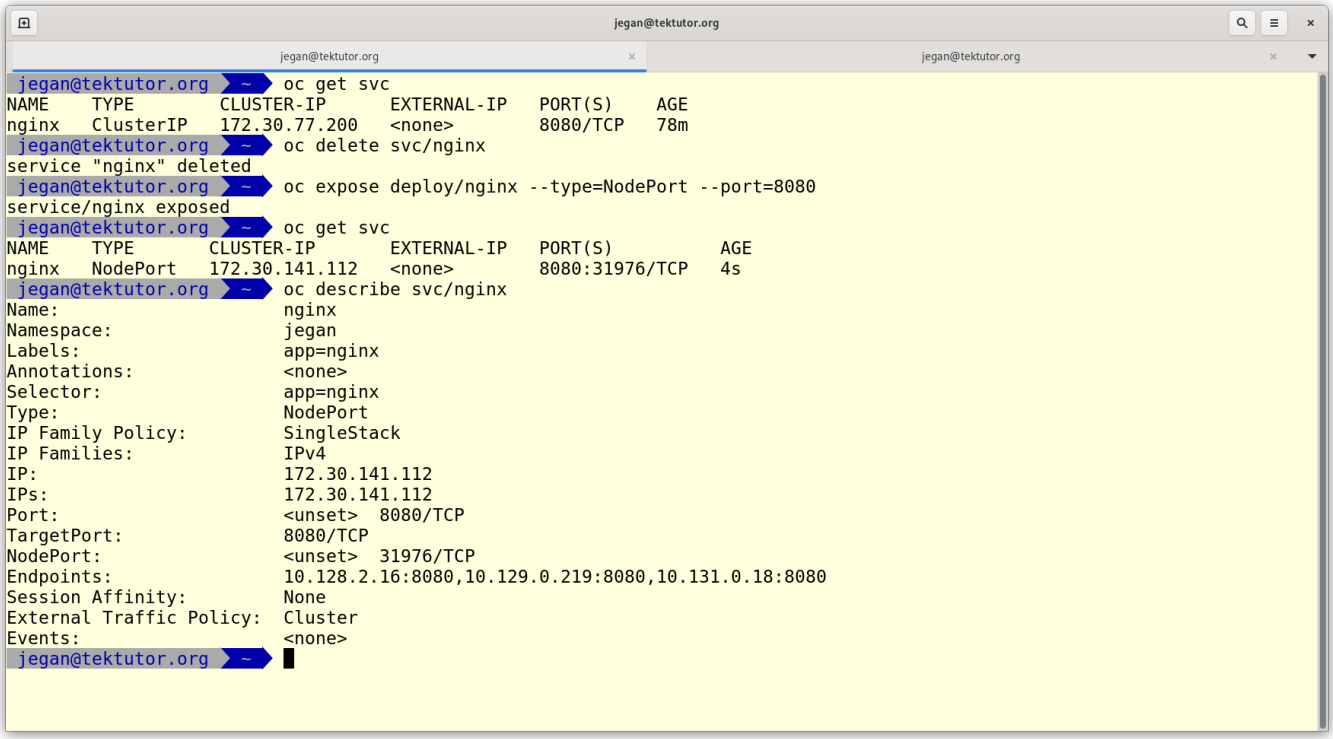
Let's create the nodeport service for nginx deployment

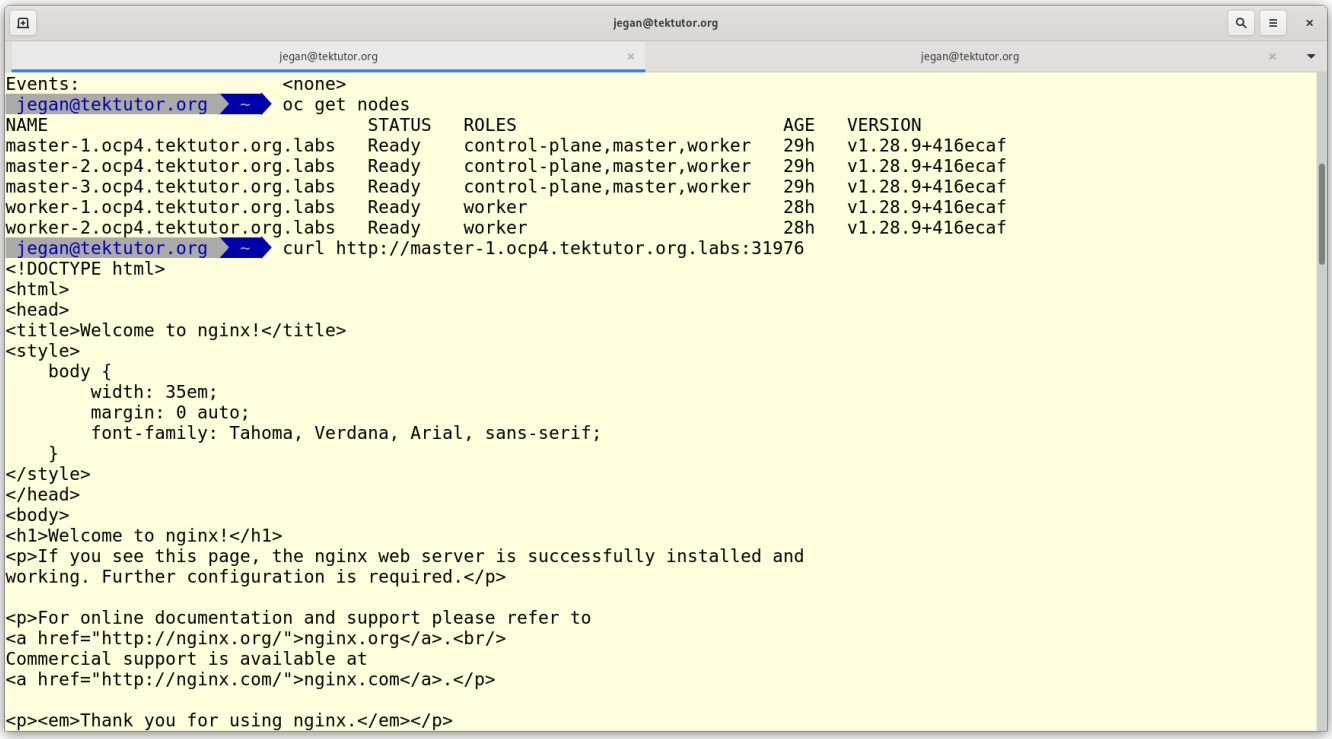
```
oc expose deploy/nginx --type=NodePort --port=8080
oc get svc
oc describe svc/nginx
```

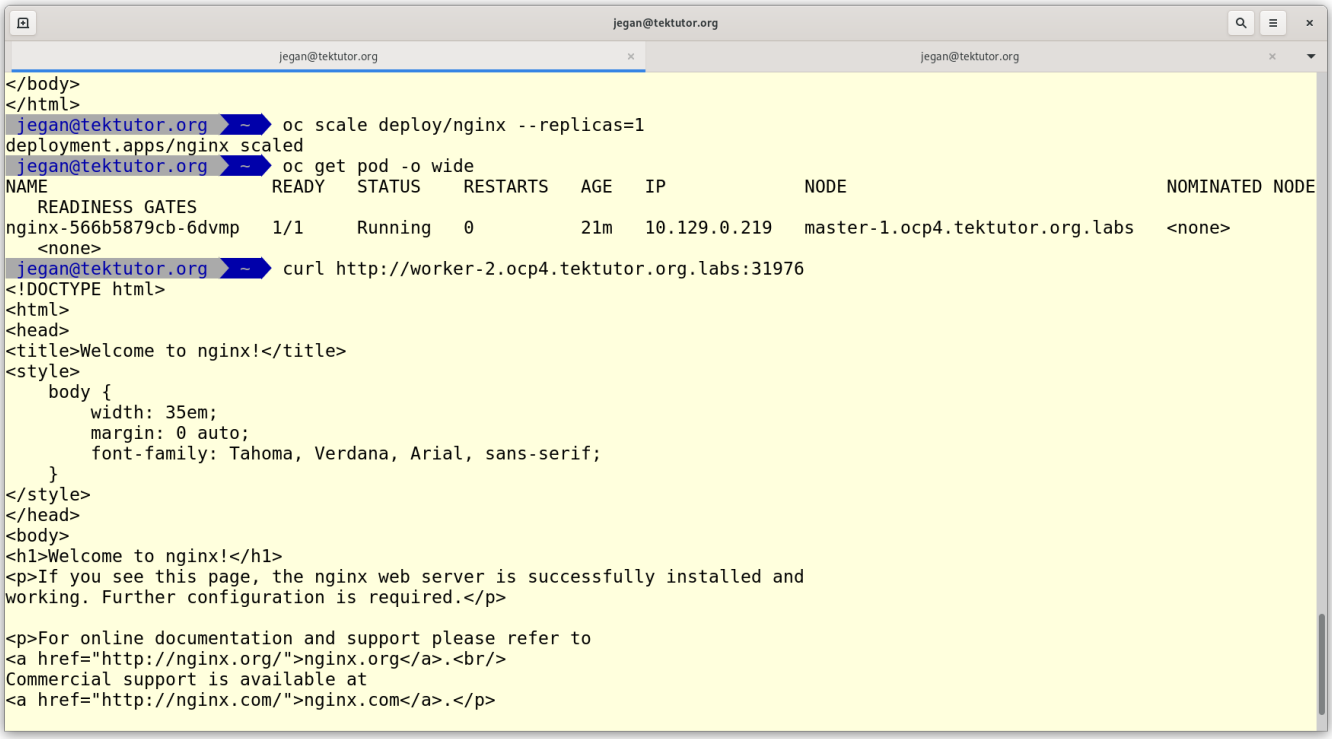
We can access the nodeport service using any one of the Node IP or the node hostnames

```
oc get nodes
curl http://master-1.ocp4.tektutor.org.labs:31976
curl http://master-2.ocp4.tektutor.org.labs:31976
curl http://master-3.ocp4.tektutor.org.labs:31976
curl http://worker-1.ocp4.tektutor.org.labs:31976
curl http://worker-2.ocp4.tektutor.org.labs:31976
```

Expected output







Info - Drawbacks of using NodePort service

```
- NodePort isn't end-user friendly as the end-user has to know the node ip
or the node hostname in order to access the service
- it is not developer friendly as well as the dev environment openshift may
have a single node openshift cluster while in prod environment the
openshift cluster may have many nodes, hence the way we access the nodeport
in different clusters for the same application will vary
- from the security point of view, the nodeport forces us to open up a
single port on every node for a nodeport service.  If we have to create 10
nodeport services for different microservice then we end opening 10 ports
```

```
on each of the nodes in the openshift cluster
- as we open more ports in the cluster, it is vulnerable for attacks
- the openshift solution for these problem is to use route
```

Lab - Creating a LoadBalancer external service

In order to support loadbalancer service in a on-prem setup, we need to install MetalLB operator. You may refer the instructions here

```
https://medium.com/tektutor/using-metallb-loadbalancer-with-bare-metal-openshift-onprem-4230944bfa35
```

First let's delete the nodeport service

```
oc get svc
oc delete svc/nginx
```

Let's create a loadbalancer service for the nginx deployment

```
oc get deploy,po
oc expose deploy/nginx --type=LoadBalancer --port=8080
```

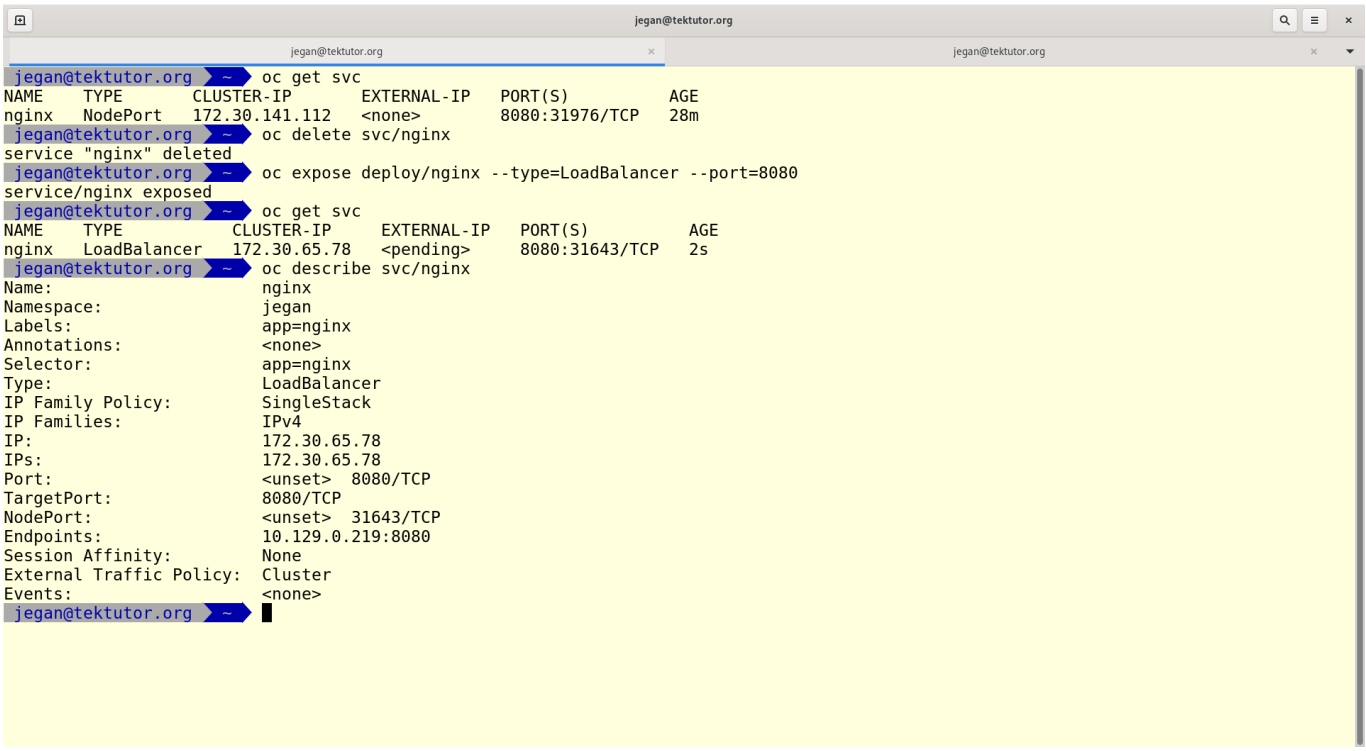
List the service

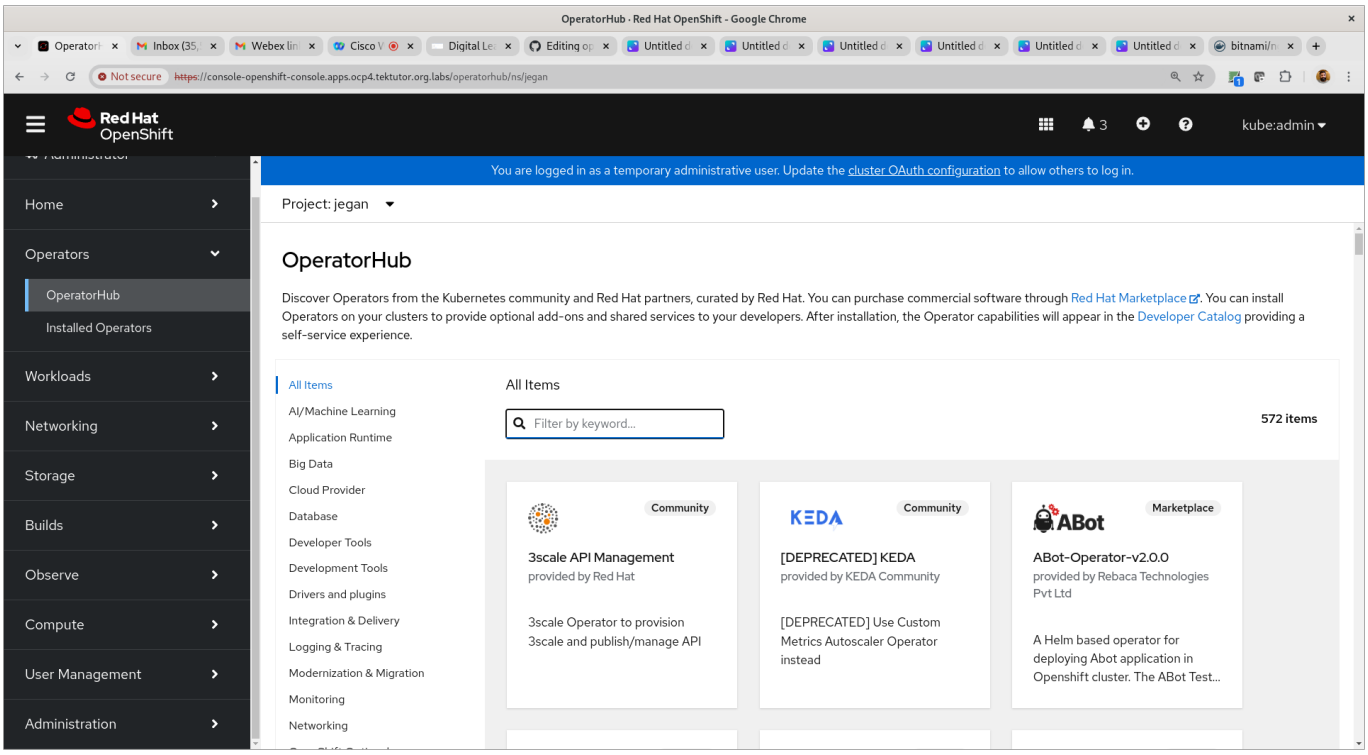
```
oc get svc
oc describe svc/nginx
```

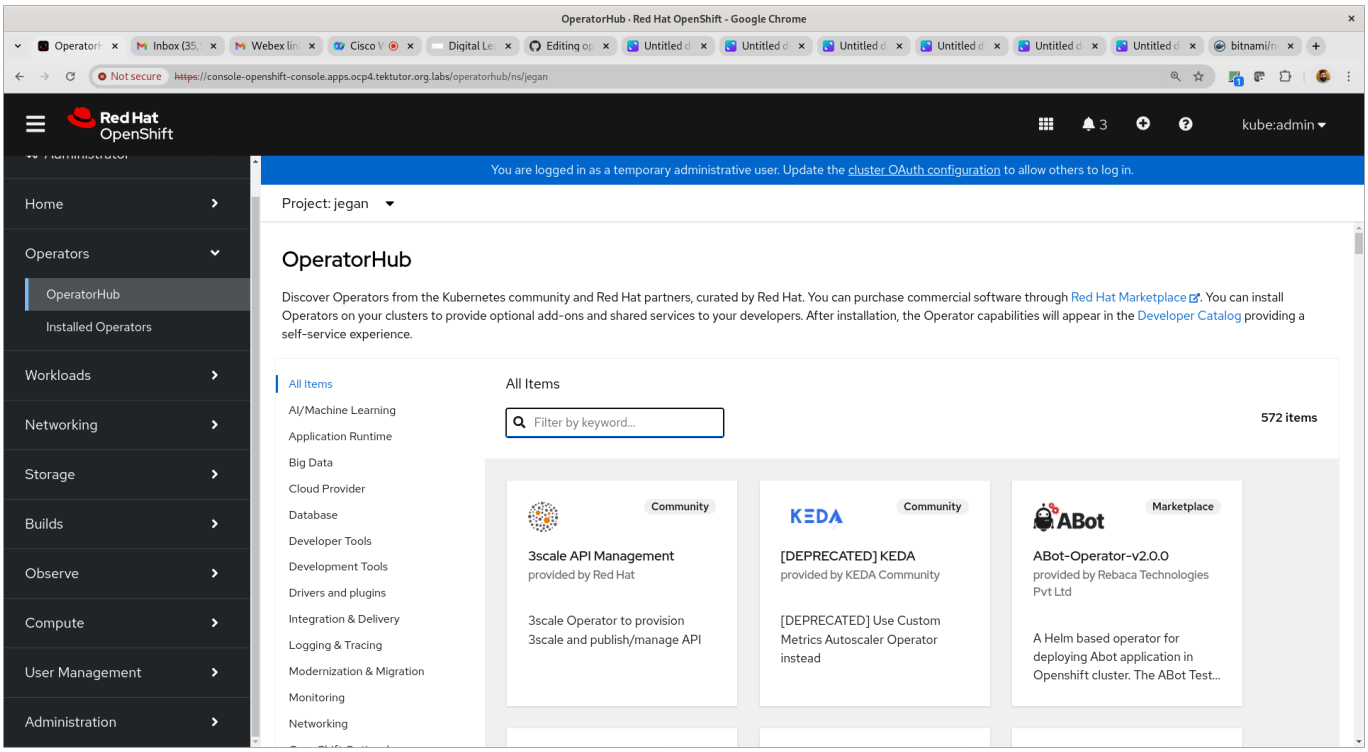
Accessing the service

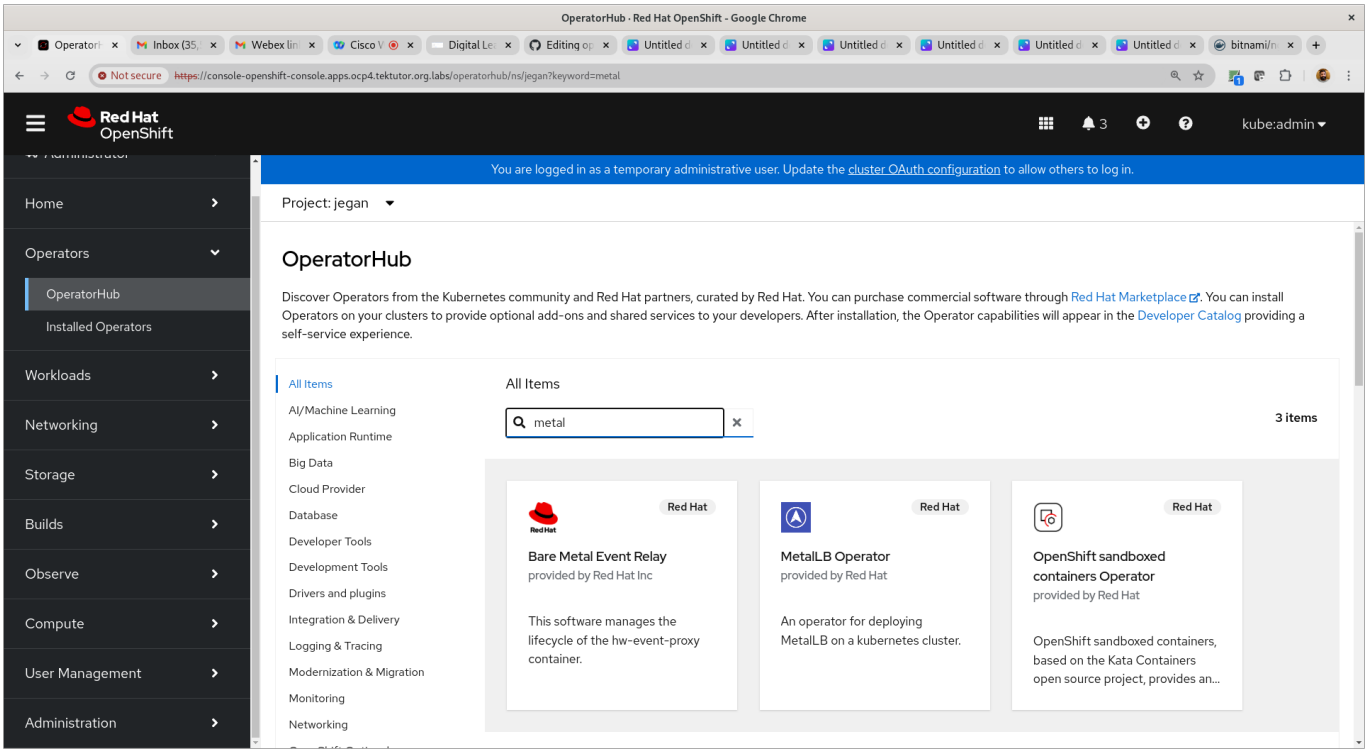
```
curl http://<external-ip-shown-in-loadbalancer-service>:8080
```

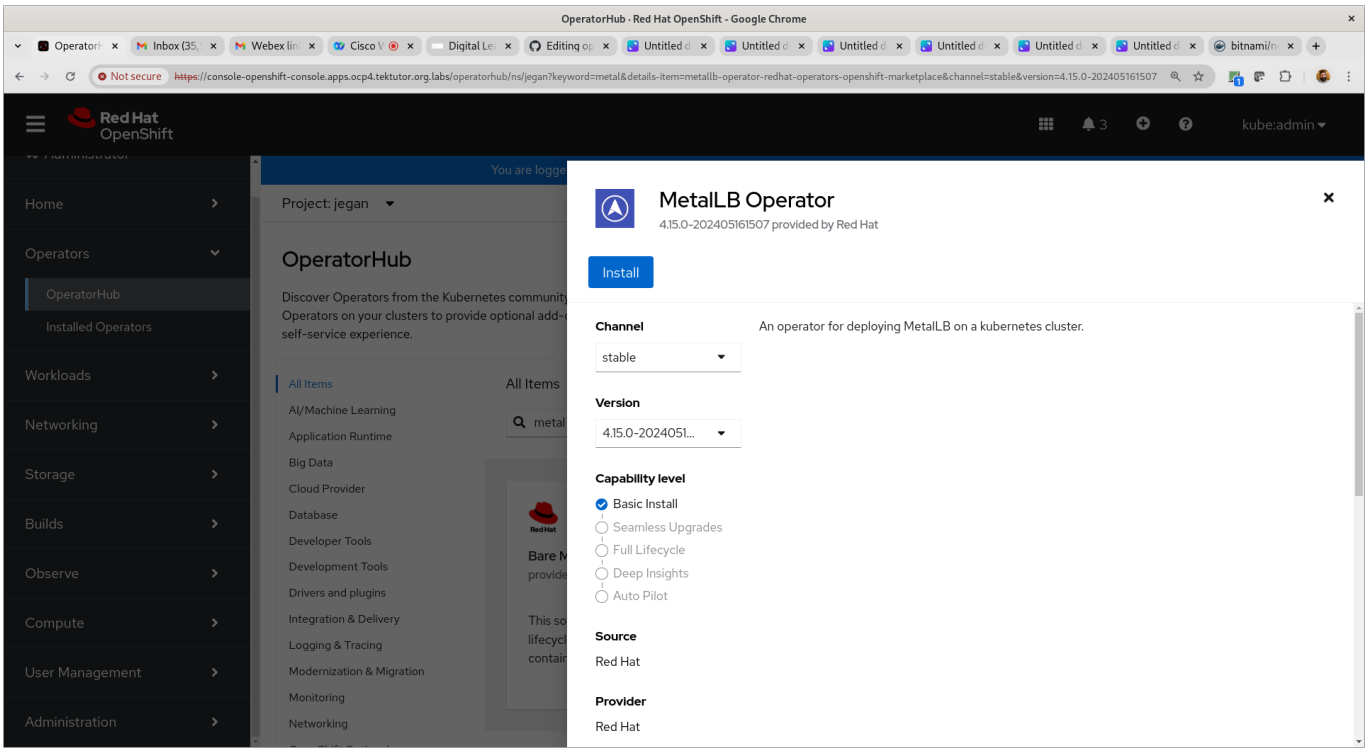
Expected output

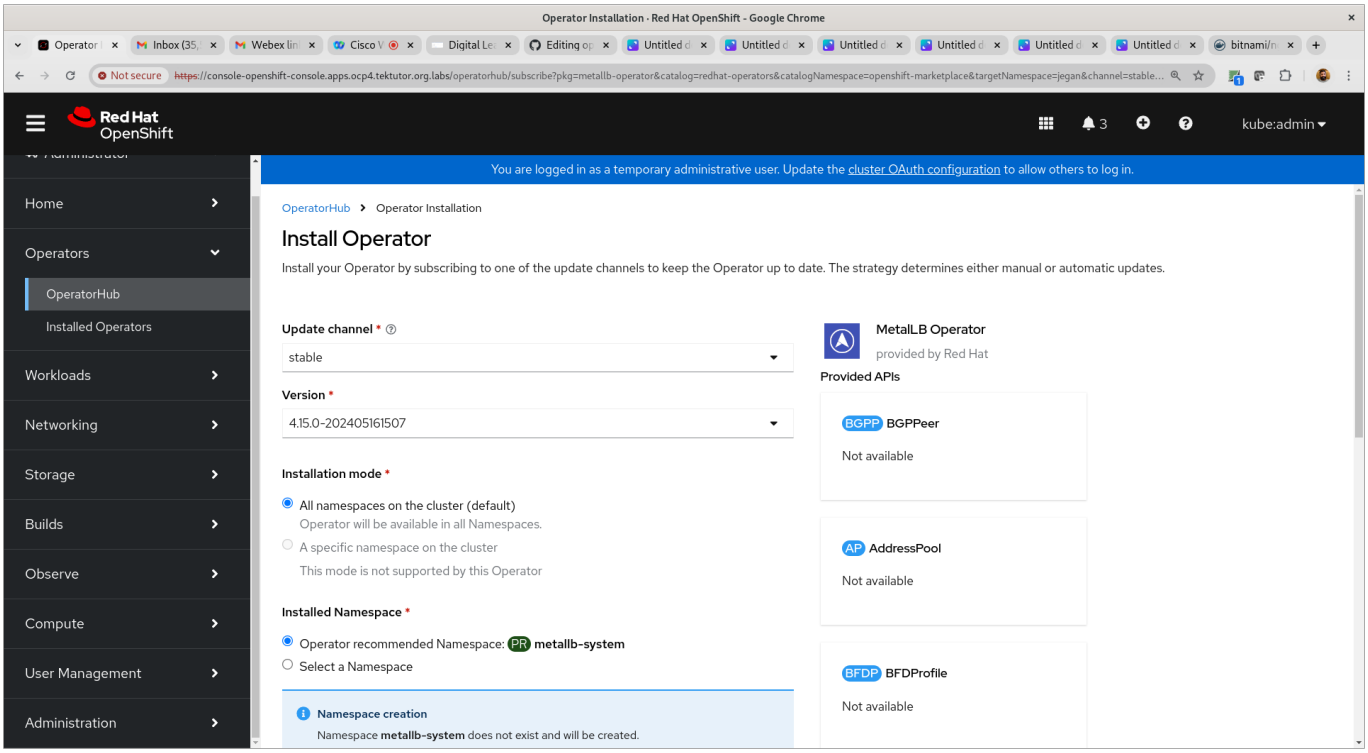


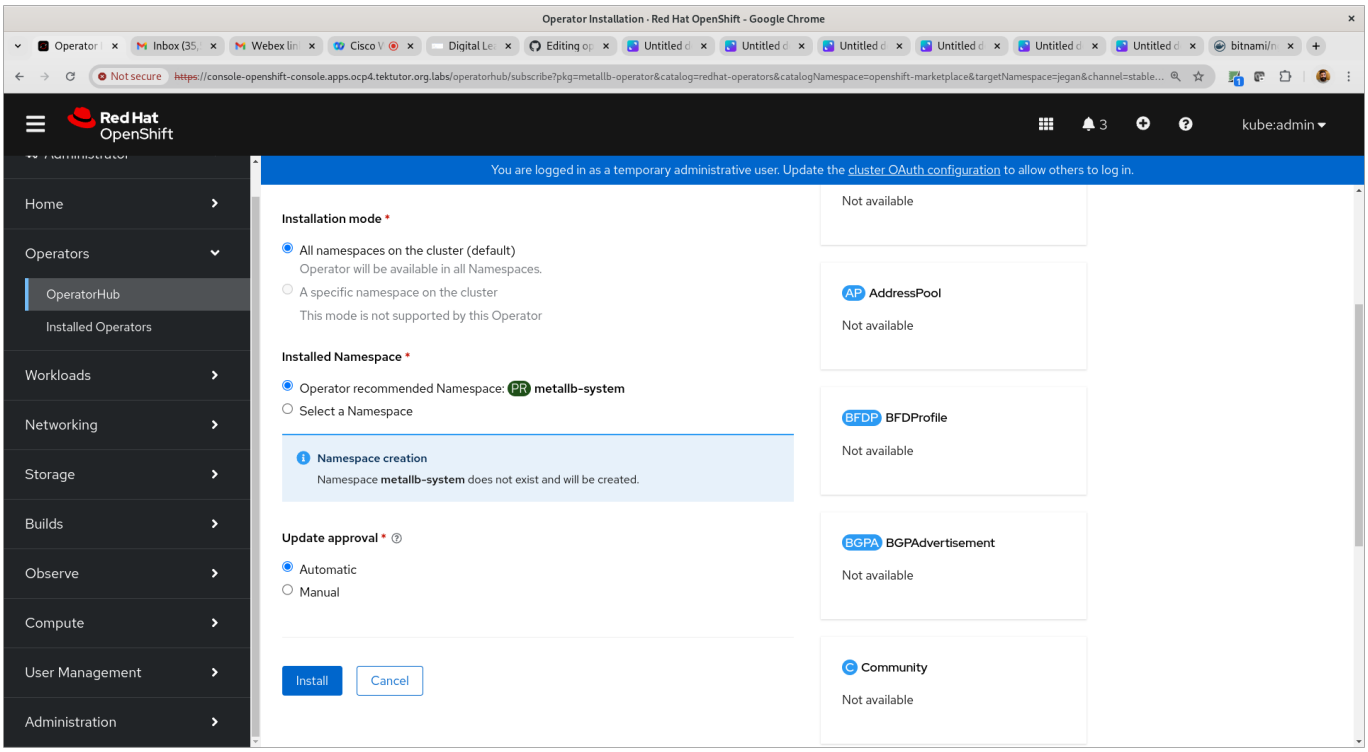


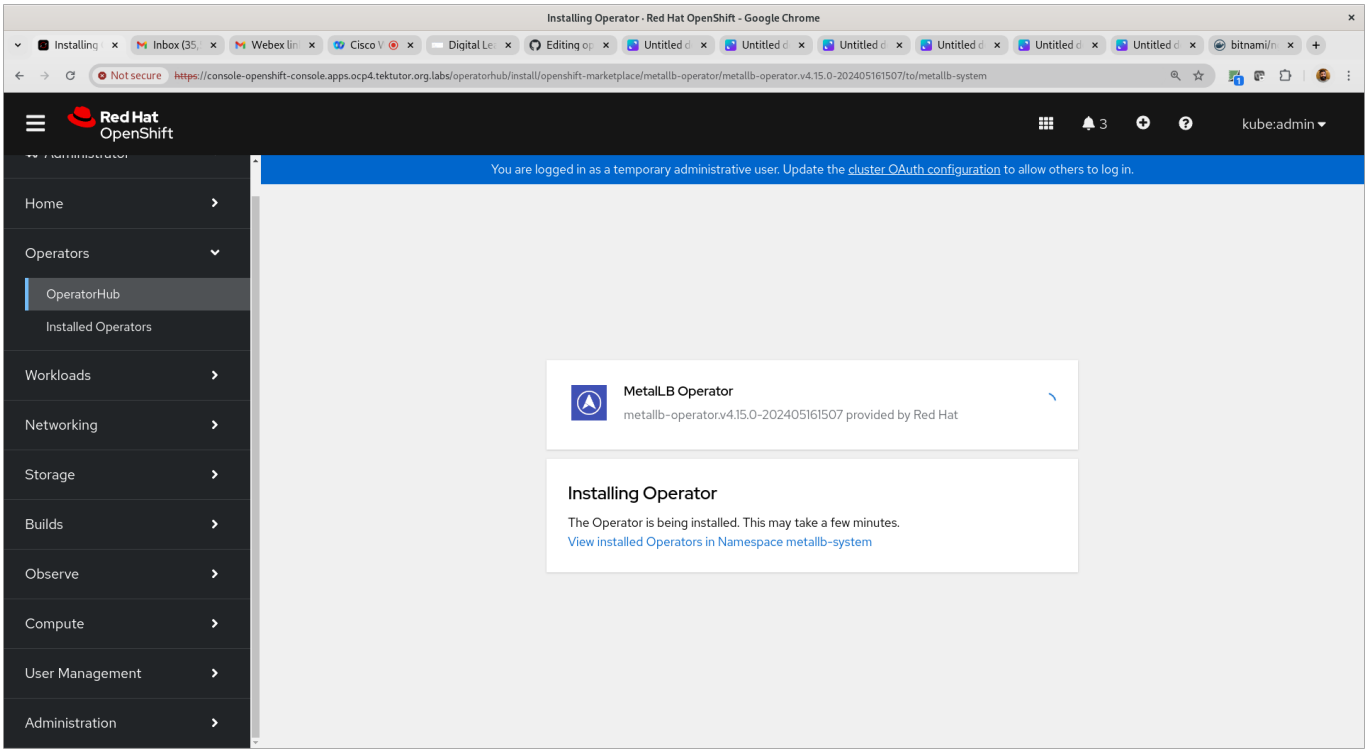

OperatorHub · Red Hat OpenShift - Google Chrome
Red Hat OpenShift
kube:admin
You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.
Project: jegan
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
OperatorHub
Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. You can purchase commercial software through Red Hat Marketplace. You can install Operators on your clusters to provide optional add-ons and shared services to your developers. After installation, the Operator capabilities will appear in the Developer Catalog providing a self-service experience.
All Items
AI/Machine Learning
Application Runtime
Big Data
Cloud Provider
Database
Developer Tools
Development Tools
Drivers and plugins
Integration & Delivery
Logging & Tracing
Modernization & Migration
Monitoring
Networking
All Items
Filter by keyword...
572 items
Community
3scale API Management
provided by Red Hat
3scale Operator to provision 3scale and publish/manage API
Community
[DEPRECATED] KEDA
provided by KEDA Community
[DEPRECATED] Use Custom Metrics Autoscaler Operator instead
Marketplace
ABot-Operator-v2.0.0
provided by Rebaca Technologies Pvt Ltd
A Helm based operator for deploying Abot application in Openshift cluster. The ABot Test...

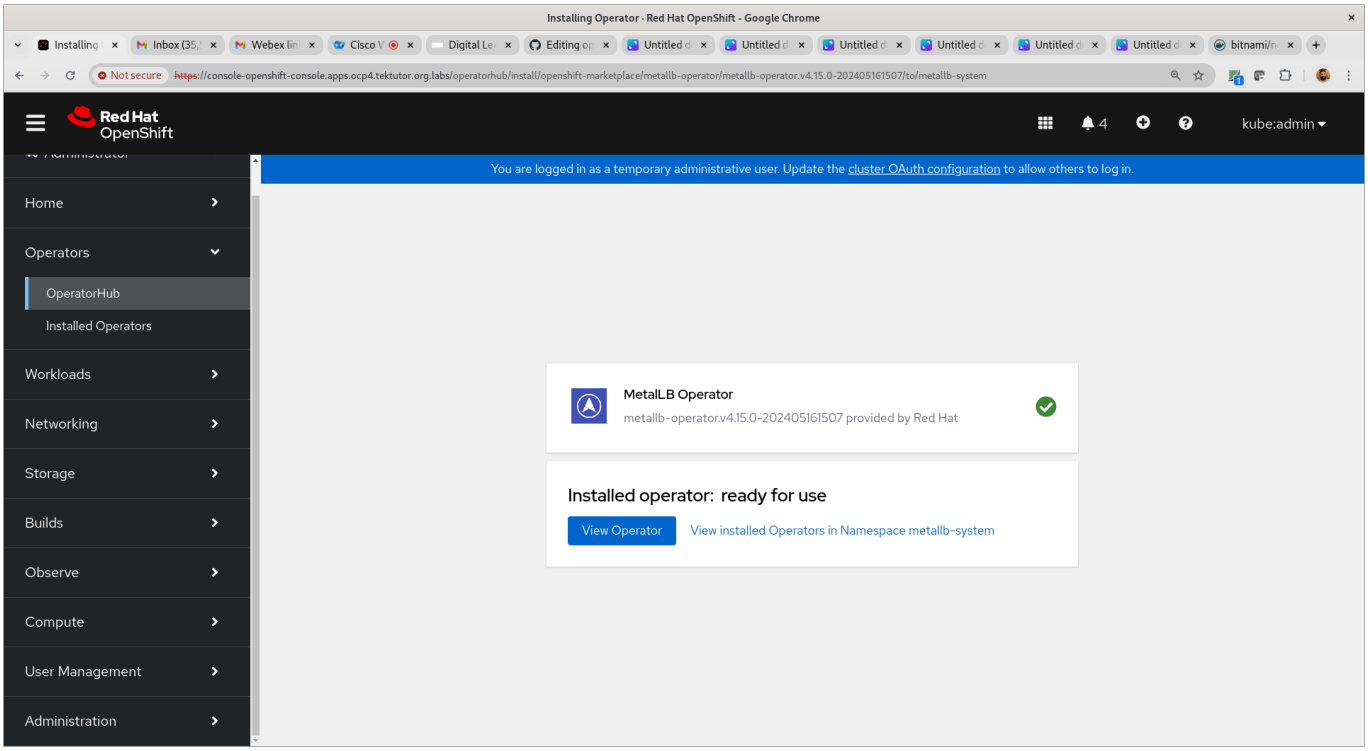

OperatorHub · Red Hat OpenShift - Google Chrome
Red Hat OpenShift
kube:admin
You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.
Project: jegan
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
OperatorHub
Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. You can purchase commercial software through Red Hat Marketplace. You can install Operators on your clusters to provide optional add-ons and shared services to your developers. After installation, the Operator capabilities will appear in the Developer Catalog providing a self-service experience.
All Items
AI/Machine Learning
Application Runtime
Big Data
Cloud Provider
Database
Developer Tools
Development Tools
Drivers and plugins
Integration & Delivery
Logging & Tracing
Modernization & Migration
Monitoring
Networking
All Items
metal
3 items
Red Hat
Bare Metal Event Relay
provided by Red Hat Inc
This software manages the lifecycle of the hw-event-proxy container.
Red Hat
MetalLB Operator
provided by Red Hat
An operator for deploying MetalLB on a kubernetes cluster.
Red Hat
OpenShift sandboxed containers Operator
provided by Red Hat
OpenShift sandboxed containers, based on the Kata Containers open source project, provides an...


OperatorHub · Red Hat OpenShift - Google Chrome
Red Hat OpenShift
kube:admin
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
Project: jegan
OperatorHub
Discover Operators from the Kubernetes community
Operators on your clusters to provide optional add-
self-service experience.
All Items
AI/Machine Learning
Application Runtime
Big Data
Cloud Provider
Database
Developer Tools
Development Tools
Drivers and plugins
Integration & Delivery
Logging & Tracing
Modernization & Migration
Monitoring
Networking
MetalLB Operator
4.15.0-202405161507 provided by Red Hat
Install
Channel
stable
An operator for deploying MetalLB on a kubernetes cluster.
Version
4.15.0-2024051...
Capability level
Basic Install
Seamless Upgrades
Full Lifecycle
Deep Insights
Auto Pilot
Source
Red Hat
Provider
Red Hat


Operator Installation · Red Hat OpenShift - Google Chrome
Red Hat OpenShift
kube:admin
You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.
OperatorHub › Operator Installation
Install Operator
Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.
Update channel *
stable
Version *
4.15.0-202405161507
Installation mode *
All namespaces on the cluster (default)
Operator will be available in all Namespaces.
A specific namespace on the cluster
This mode is not supported by this Operator
Installed Namespace *
Operator recommended Namespace: PR metallb-system
Select a Namespace
Namespace creation
Namespace metallb-system does not exist and will be created.
MetalLB Operator
provided by Red Hat
Provided APIs
BGPP BGPPeer
Not available
AP AddressPool
Not available
BFDP BFDProfile
Not available


Operator Installation - Red Hat OpenShift - Google Chrome
Red Hat OpenShift
kube:admin
You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
Installation mode *
All namespaces on the cluster (default)
Operator will be available in all Namespaces.
A specific namespace on the cluster
This mode is not supported by this Operator
Installed Namespace *
Operator recommended Namespace: metallb-system
Select a Namespace
Namespace creation
Namespace metallb-system does not exist and will be created.
Update approval *
Automatic
Manual
Install
Cancel
Not available
AddressPool
Not available
BFDProfile
Not available
BGPAdvertisement
Not available
Community
Not available


Installing Operator - Red Hat OpenShift - Google Chrome
Red Hat OpenShift
kube:admin
You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
MetalLB Operator
metallb-operator.v4.15.0-202405161507 provided by Red Hat
Installing Operator
The Operator is being installed. This may take a few minutes.
View installed Operators in Namespace metallb-system




```
jegan@tektutor.org ~/openshift-3june-2024 oc expose deploy/hello --type=LoadBalancer --port=8080
service/hello exposed
jegan@tektutor.org ~/openshift-3june-2024 oc get svc
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)          AGE
hello   LoadBalancer   172.30.28.71   192.168.122.91   8080:32356/TCP   3s
nginx   LoadBalancer   172.30.65.78   192.168.122.90   8080:31643/TCP   20m
jegan@tektutor.org ~/openshift-3june-2024 curl http://192.168.122.90:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
jegan@tektutor.org ~/openshift-3june-2024 curl http://192.168.122.91:8080
Greetings from Spring Boot!%
jegan@tektutor.org ~/openshift-3june-2024
```