Lab: Understanding and Working with PODs

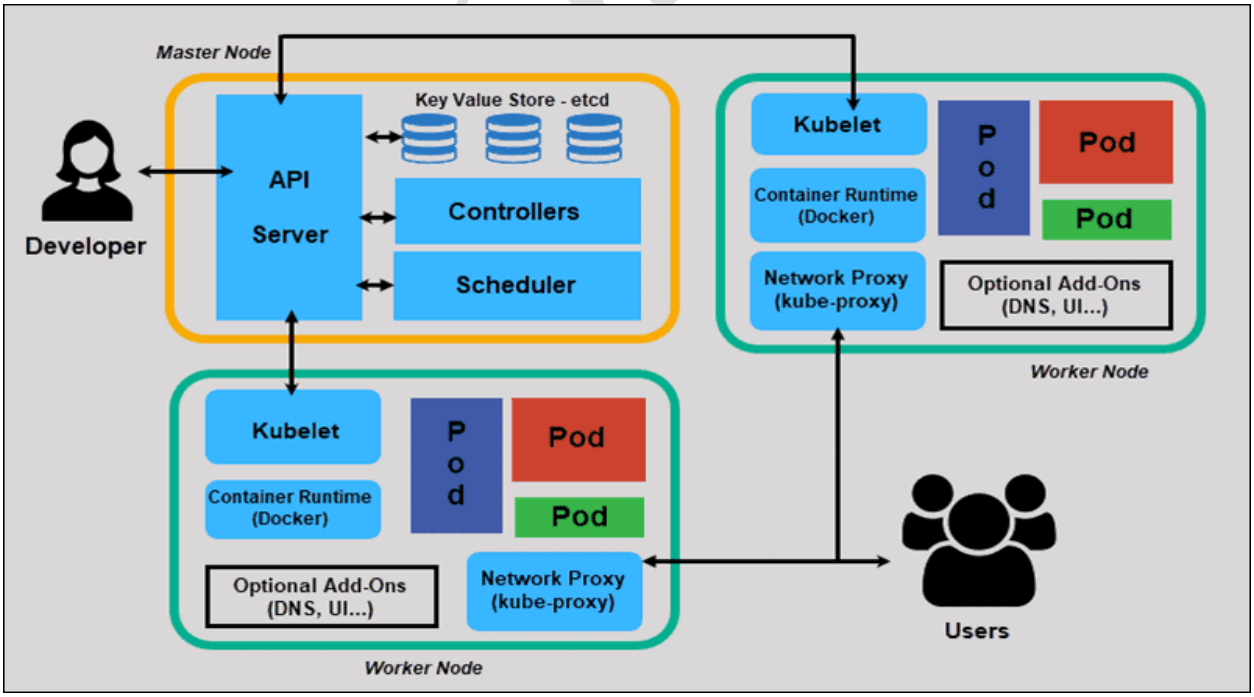
Introduction:

Pods are the Smallest Deployable Units of computing that can be created and managed in Kubernetes.

A Pod (as in a pod of whales or pea pod) is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. A Pod's contents are always co-located and co-scheduled, and run in a shared context.

A Pod models an application-specific "logical host": it contains one or more application containers which are relatively tightly coupled. In non-cloud contexts, applications executed on the same physical or virtual machine are analogous to cloud applications executed on the same logical host.

The shared context of a Pod is a set of Linux namespaces, cgroups, and potentially other facets of isolation - the same things that isolate a container. Within a Pod's context, the individual applications may have further sub-isolations applied. A Pod is similar to a set of containers with shared namespaces and shared filesystem volumes.



Objective:

- **Creating A Pod Using Imperative Method**
- **Creating A Pod Using Declarative Method**
- **Generate the Pod Manifest in the YAML& JSON Format**
- **Create a Temporary Pod**

Ensure that you have logged-in as **root** user on **eoc-controller** node.

1 Creating A Pod with Imperative Method.

```
# kubectl run web01 --image nginx
```

Output:

```
[root@eoc-controller ~]#kubectl run web01 --image nginx
pod/web01 created
```

1.1 Let's **list** the pod status by executing the below command.

```
# kubectl get pods
```

Output:

```
[root@eoc-controller ~]#kubectl get pods
NAME     READY    STATUS     RESTARTS    AGE
web01    1/1      Running    0           51s
```

2 Create A Pod Using Declarative Method.

2.1 Let's **view** the yaml manifest file by executing below command.

```
# cat -n ~/kubernetes/pod-webserver.yml
```

Output:

```
[root@eoc-controller ~]#cat -n ~/kubernetes/pod-webserver.yml
     1  apiVersion: v1
     2  kind: Pod
     3  metadata:
     4    name: webserver
     5  spec:
     6    containers:
     7    - name: webserver
     8      image: nginx
```

2.2 Let's **create** a pod containing a **nginx** server using **webserver.yml** file by executing the below command.

```
# kubectl create -f ~/kubernetes/pod-webserver.yml
```

Output:

```
[root@eoc-controller ~]#kubectl create -f ~/kubernetes/pod-webserver.yml
pod/webserver created
```

2.3 Let's **list** all of the **pods** which are in running state.

```
# kubectl get pods
```

Output:

```
[root@eoc-controller ~]#kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
web01       1/1     Running   0          4m23s
webserver   1/1     Running   0          29s
```

2.4 To **check** the **detailed** output of yaml&json by executing the below command.

```
# kubectl get pods -o wide
```

Output:

```
[root@eoc-controller ~]#kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE     IP           NODE        NOMINATED NODE   READINESS GATES
web01       1/1     Running   0          11m     10.32.0.2    eoc-node1   <none>           <none>
webserver   1/1     Running   0          7m14s   10.32.0.3    eoc-node1   <none>           <none>
```

```
# kubectl get pod webserver -o yaml
```

Output:

```
[root@eoc-controller ~]#kubectl get pod webserver -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2023-08-30T10:06:13Z"
  name: webserver
  namespace: default
  resourceVersion: "7155"
  uid: 52829765-c81f-43c4-89e3-4c133ca678b8
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: webserver
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-bczs8
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: eoc-node1
  preemptionPolicy: PreemptLowerPriority
  priority: 0
```

```
# kubectl get pod webserver -o json
```

Output:

```
[root@eoc-controller ~]#kubectl get pod webserver -o json
{
    "apiVersion": "v1",
    "kind": "Pod",
    "metadata": {
        "creationTimestamp": "2023-08-30T10:06:13Z",
        "name": "webserver",
        "namespace": "default",
        "resourceVersion": "7155",
        "uid": "52829765-c81f-43c4-89e3-4c133ca678b8"
    },
    "spec": {
        "containers": [
            {
                "image": "nginx",
                "imagePullPolicy": "Always",
                "name": "webserver",
                "resources": {},
                "terminationMessagePath": "/dev/termination-log",
                "terminationMessagePolicy": "File",
                "volumeMounts": [
                    {
                        "mountPath": "/var/run/secrets/kubernetes.io/serviceaccount",
                        "name": "kube-api-access-bczs8",
                        "readOnly": true
                    }
```

2.5 Let's **capture** Pod IP to a variable by running below command.

```
# kubectl get pod webserver -o jsonpath='{.status.podIP}'
```

```
# podIP=$(kubectl get pod webserver -o
jsonpath='{.status.podIP}')
```

```
# echo $podIP
```

Output:

```
[root@eoc-controller ~]#echo $podIP
10.32.0.3
```

Note: **Wait till the pod status changes to "running" status.**

2.6 Let's **access** the pod using the ip displayed in previous step.

```
 # curl $podIP
```

Output:

```
[root@eoc-controller ~]#curl $podIP
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
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>
```

2.7 Let's **describe** the pod by executing bellow command and check pod details.

```
# kubectl describe pod webserver
```

Output:

```
[root@eoc-controller ~]#kubectl describe pod webserver
Name:             webserver
Namespace:        default
Priority:         0
Service Account:  default
Node:             eoc-node1/192.168.100.12
Start Time:       Wed, 30 Aug 2023 06:06:13 -0400
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               10.32.0.3
IPs:
  IP:  10.32.0.3
Containers:
  webserver:
    Container ID:   containerd://c863a94d9238d76c6ba1f5b35bf3a170d8bb7589d6edf41b09b2eb2dba28da06
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:104c7c5c54f2685f0f46f3be607ce60da7085da3eaa5ad22
594295e9c
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Wed, 30 Aug 2023 06:06:16 -0400
    Ready:          True
    Restart Count:  0
    Environment:    <none>
```

2.8 Let's **check** the **logs** of the webserver application.

```
# kubectl logs webserver
```

Output:

```
[root@eoc-controller ~]#kubectl logs webserver
/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: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/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
2023/08/30 10:06:16 [notice] 1#1: using the "epoll" event method
2023/08/30 10:06:16 [notice] 1#1: nginx/1.25.2
2023/08/30 10:06:16 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2023/08/30 10:06:16 [notice] 1#1: OS: Linux 4.18.0-512.el8.x86_64
2023/08/30 10:06:16 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/08/30 10:06:16 [notice] 1#1: start worker processes
2023/08/30 10:06:16 [notice] 1#1: start worker process 29
2023/08/30 10:06:16 [notice] 1#1: start worker process 30
2023/08/30 10:06:16 [notice] 1#1: start worker process 31
2023/08/30 10:06:16 [notice] 1#1: start worker process 32
10.46.0.0 - - [30/Aug/2023:10:19:13 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/7.61.1" "-"
```

2.9 Let's **dry-run** the creation of the pod by executing the below command.

```
# kubectl run web02 --image nginx --dry-run=client
```

Output:

```
[root@eoc-controller ~]#kubectl run web02 --image nginx --dry-run=client
pod/web02 created (dry run)
```

Note: **Pod is actually not created it just gives the confirmation that pod can be created.**

3 Generate the Pod Manifest in the YAML & JSON Format

3.1 Let's **generate** the pod manifest in the **yml** & **json** format by executing the below commands.

```
# kubectl run web02 --image nginx --dry-run=client -o yaml |
tee web02-pod.yml
```

Output:

```
[root@eoc-controller ~]#kubectl run web02 --image nginx --dry-run=client -o yaml |
 tee web02-pod.yml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: web02
  name: web02
spec:
  containers:
  - image: nginx
    name: web02
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

```
# kubectl run web02 --image nginx --dry-run=client -o json |
tee web02-pod.json
```

Output:

```
[root@eoc-controller ~]#kubectl run web02 --image nginx --dry-run=client -o json |
 tee web02-pod.json
{
    "kind": "Pod",
    "apiVersion": "v1",
    "metadata": {
        "name": "web02",
        "creationTimestamp": null,
        "labels": {
            "run": "web02"
        }
    },
    "spec": {
        "containers": [
            {
                "name": "web02",
                "image": "nginx",
                "resources": {}
            }
        ],
        "restartPolicy": "Always",
        "dnsPolicy": "ClusterFirst"
    },
    "status": {}
}
```

3.2 Let's **create** a pod using the **web02-pod.json** file which we saved in the above step.

```
# kubectl create -f web02-pod.json
```

Output:

```
[root@eoc-controller ~]#kubectl create -f web02-pod.json
pod/web02 created
```

3.3 Let's **list** all of the pods which are in running state by executing the below command.

```
# kubectl get pods
```

Output:

```
[root@eoc-controller ~]#kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
web01       1/1     Running   0          42m
web02       1/1     Running   0          64s
webserver   1/1     Running   0          39m
```

3.4 Let's **list** the pods with **default labels**.

```
# kubectl get pods --show-labels
```

utput:

```
[root@eoc-controller ~]#kubectl get pods --show-labels
NAME        READY   STATUS    RESTARTS   AGE   LABELS
web01       1/1     Running   0          43m   run=web01
web02       1/1     Running   0          89s   run=web02
webserver   1/1     Running   0          39m   <none>
```

3.5 Let's **label** the pod webserver.

```
# kubectl label pod webserver app=webserver
```

Output:

```
[root@eoc-controller ~]#kubectl label pod webserver app=webserver
pod/webserver labeled
```

3.6 Let's **list** the pods with default labels.

```
# kubectl get pods --show-labels
```

Output:

```
[root@eoc-controller ~]#kubectl get pods --show-labels
NAME        READY   STATUS    RESTARTS   AGE     LABELS
web01       1/1     Running   0          49m     run=web01
web02       1/1     Running   0          7m34s   run=web02
webserver   1/1     Running   0          45m     app=webserver
```

3.7 Let's **overwrite** the existing label with new label.

```
# kubectl label pods web01 web02 app=webserver2
```

Output:

```
[root@eoc-controller ~]#kubectl label pods web01 web02 app=webserver2
pod/web01 labeled
pod/web02 labeled
```

3.8 Let's **list** the overwrite labels.

```
# kubectl get pods --show-labels
```

Output:

```
[root@eoc-controller ~]#kubectl get pods --show-labels
NAME        READY   STATUS    RESTARTS   AGE   LABELS
web01       1/1     Running   0          71m   app=webserver2,run=web01
web02       1/1     Running   0          29m   app=webserver2,run=web02
webserver   1/1     Running   0          67m   app=webserver
```

3.9 Let's **list** the pods using labels by executing below command.

```
# kubectl get pods -l app=webserver2
```

Output:

```
[root@eoc-controller ~]#kubectl get pods -l app=webserver2
NAME    READY   STATUS    RESTARTS   AGE
web01   1/1     Running   0          64m
web02   1/1     Running   0          22m
```

3.10 Annotations

You can use Kubernetes annotations to attach arbitrary non-identifying metadata to objects. Clients such as tools and libraries can retrieve this metadata.

```
# kubectl annotate pod webserver
repository=https://raw.githubusercontent.com/EyesOnCloud/kuber
netes/pod-webserver.yml
```

Output:

```
[root@eoc-controller ~]#kubectl annotate pod webserver repository=https://raw.githubusercon
tent.com/EyesOnCloud/kubernetes/pod-webserver.yml
pod/webserver annotated
```

3.11 Let's **describe** the pod to verify.

```
# kubectl describe pod webserver
```

Output:

```
[root@eoc-controller ~]#kubectl describe pod webserver
Name:             webserver
Namespace:        default
Priority:         0
Service Account:  default
Node:             eoc-node1/192.168.100.12
Start Time:       Wed, 30 Aug 2023 06:06:13 -0400
Labels:           app=webserver
Annotations:      repository: https://raw.githubusercontent.com/EyesOnCloud/kubernetes/pod-webserver.yml
Status:           Running
IP:               10.32.0.3
IPs:
  IP:  10.32.0.3
```

3.12 Let's **delete** the pod using label by executing below command

```
# kubectl delete pod -l app=webserver2
```

Output:

```
[root@eoc-controller ~]#kubectl delete pod -l app=webserver2
pod "web01" deleted
pod "web02" deleted
```

3.13 Let's **get into a shell** to the running container.

```
# kubectl exec webserver -it -c webserver -- /bin/bash
```

Output:

```
[root@eoc-controller ~]#kubectl exec webserver -it -c webserver -- /bin/bash
root@webserver:/#
```

3.14 Let's do **ls** inside the running container.

```
# ls
```

Output:

```
root@webserver:/# ls
bin   docker-entrypoint.d   home   lib64   mnt   root  srv  usr
boot  docker-entrypoint.sh  lib    libx32  opt   run   sys  var
dev   etc                   lib32  media   proc  sbin  tmp
```

3.15 In your shell **create an index.html** file in the /usr/share/nginx/html directory.

```
# echo 'Hello Webserver' > /usr/share/nginx/html/index.html
```

3.16 Let's **verify** by executing the below command.

```
# curl http://localhost/
```

Output:

```
root@webserver:/# echo 'Hello Webserver' > /usr/share/nginx/html/index.html
root@webserver:/# curl http://localhost/
Hello Webserver
```

```
# exit
```

3.17 Let's **expose** the Pod on Port 80 by running below command.

```
# kubectl expose pod webserver --port=80
```

Output:

```
[root@eoc-controller ~]#kubectl expose pod webserver --port=80
service/webserver exposed
```

3.18 Lets check the **service** by executing below command.

```
# kubectl get service
```

Output:

```
[root@eoc-controller ~]#kubectl get service
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP   155m
webserver    ClusterIP   10.101.225.146   <none>        80/TCP    37s
```

```
# curl SERVICEIP
```

Note: Use your service ip.

Output:

```
[root@eoc-controller ~]#curl 10.101.225.146
Hello Webserver
```

3.19 Lets **check** the endpoind by executiing below command.

```
# kubectl get ep webserver
```

Output:

```
[root@eoc-controller ~]#kubectl get ep webserver
NAME        ENDPOINTS       AGE
webserver   10.32.0.3:80    2m10s
```

3.20 Let's **delete** pod by executing the below command.

```
# kubectl delete pod webserver
```

Output:

```
[root@eoc-controller ~]#kubectl delete pod webserver
pod "webserver" deleted
```

3.21 Let's **delete** the **service** by executing below command

```
# kubectl delete service webserver
```

Output:

```
[root@eoc-controller ~]#kubectl delete service webserver
service "webserver" deleted
```

4 Creating a Temporary pod

4.1 Let's **create** a temporary pod by executing below command.

```
# kubectl run temporary-pod -i -t --image=busybox:1.28.0 --rm
-- sh
```

```
# ls
```

```
# ps
```

Output:

```
[root@eoc-controller ~]#kubectl run temporary-pod -i -t --image=busybox:1.28.0 --rm -- sh
If you don't see a command prompt, try pressing enter.
/ # ls
bin   dev   etc   home  proc  root  sys   tmp   usr   var
/ # ps
PID   USER     TIME  COMMAND
    1 root      0:00 sh
    9 root      0:00 ps
```

4.2 Exit out of the pod will also delete the pod.

```
# exit
```

Output:

```
/ # exit
Session ended, resume using 'kubectl attach temporary-pod -c temporary-pod -i -t' command wh
en the pod is running
pod "temporary-pod" deleted
```