Day 2

Info - In case you are curious to know how etcd databses in 3 master nodes synchronizes data automatically, you may check this

```
https://etcd.io/docs/v3.5/tutorials/how-to-setup-cluster/
```

Lab - Bash completion for oc commands (requires root privilege - hence I have already installed it for you)

In case of CentOS 7.x

```
oc completion bash > /etc/profile.d/
docker completion bash > /etc/profile.d/docker
exit
```

In case of RHEL

```
oc completion bash > /etc/bash_completion.d/
docker completion bash > /etc/bash_completion.d/docker
exit
```

How to check oc bash completion works in terminal?

- Try oc [space] [Tab] key
- Try docker [space] [Tab] key

Info - What happens when we create a deployment into openshift

The below command, oc is the client tool

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

```
- the oc client tool will make a REST call to API server requesting it to
create a deployment with name nginx using image bitnami/nginx:latest with 3
Pod instances
- the API Server receives the REST call from oc and then it creates a
deployment record named nginx
- the API Server sends a broadcasting event that a new deployment is
created
- the deployment controller receives the event from API Server, it then
```

```
sends a REST call to API Server requesting it to create a ReplicaSet with 3
Desired Pod using image bitnami/nginx:latest
- the API Server receives the REST call from Deployment Controller and then
it creates a ReplicaSet record in the etcd database
- the API Server sends a broadcasting event that a new ReplicaSet is
created
- the ReplicaSet Controller receives the event from API Server, it then
sends REST calls to API Server requesting it to create 3 Pods
- The API Server receives the REST call from ReplicaSet Controller and then
it creates 3 Pod records in the etcd database
- the API sends broadcasting events that 3 new Pods are created
- the Scheduler receives the event from API server, it then identifies
nodes where those 3 Pods can be deployed.
- the Scheduler makes a REST call to API Server with the Pod scheduling
recommendations
- the API Server receives the REST call from Scheduler, it then retrieves
the Pod entry from etcd database and updates the scheduling details on the
etcd database for the respective Pods
- the API Server will send broadcasting events that Pod sheduled to so and
so nodes
- the kubelet container agent running on the respective node receives the
event from API Server
- the kubelet pulls(downloads) the container image if it is not there
already, it then creates the container and start the container on the local
node where the kubelet is running
- the kubelet make a REST call to API Server sharing the status of the
containers, this happens in heart-beat like periodic fashion
- the API Server retrieves the Pod record from the etcd database and update
the Pod status based on the status update it received from kubelet
```

Lab - Creating a Pod using docker (Just for our understanding)

```
docker run -d --name nginx_pause --hostname nginx nginx
gcr.io/google_containers/pause:latest
docker run -d --name nginx --network=container:nginx_pause
bitnami/nginx:latest
docker ps
```

Let's find the IP Address of the nginx_pause container

```
docker inspect -f {{.NetworkSettings.IPAddress}} nginx_pause
docker exec -it nginx sh
hostname -i
exit
```

Expected output

```
[root@tektutor.org ~]# docker run -d --name nginx_pause --hostname nginx
gcr.io/google_containers/pause:latest
58cae658dba0f2d80671e527181e5ccc3a3c8c0a24230be1bc31d6d0cf98bc15

[root@tektutor.org ~]# docker ps
CONTAINER ID   IMAGE                                     COMMAND    CREATED
STATUS         PORTS     NAMES
58cae658dba0   gcr.io/google_containers/pause:latest   "/pause"   3 seconds
ago   Up 2 seconds             nginx_pause

[root@tektutor.org ~]# docker run -d --name nginx --
network=container:nginx_pause bitnami/nginx:latest
Unable to find image 'bitnami/nginx:latest' locally
latest: Pulling from bitnami/nginx
2f2e827ef32f: Pull complete
Digest:
sha256:476d94e773ee511bb789fafa69b2fcf39a91549dbe1386b1f64fb8f876ac3883
Status: Downloaded newer image for bitnami/nginx:latest
b44c9f65be28a22b9049b8ce66e83881d78bfb643c3ccb2bfb503a0a379ebd46

[root@tektutor.org ~]# docker ps
CONTAINER ID   IMAGE                                     COMMAND
CREATED              STATUS              PORTS     NAMES
b44c9f65be28   bitnami/nginx:latest
"/opt/bitnami/script…"   3 seconds ago        Up 2 seconds
nginx
58cae658dba0   gcr.io/google_containers/pause:latest   "/pause"
About a minute ago   Up About a minute             nginx_pause

[root@tektutor.org ~]# docker inspect nginx_pause | grep IPA
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.2",
                    "IPAMConfig": null,
                    "IPAddress": "172.17.0.2",

[root@tektutor.org ~]# docker exec -it nginx sh
$ hostname
nginx
$ hostname -i
172.17.0.2
$ exit
```

Lab - Creating a LoadBalancer service

For detailed instructions, you may check my medium blog

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

Things to note

- This requires Metallb operator installed in your Openshift cluster(I have already installed on all 3 rps servers)
- Metallb operator has to configured
    - we need create an address (range of available IP address)
    - we need to create a Metallb controller

Create an address pool - range of IP address the Metallb controller can assign to loadbalancer instances

```
cd ~/openshift-may-2024
git pull

cd Day2/metallb
oc apply -f addresspool.yml
```

Create a metallb controller

```
cd ~/openshift-may-2024
git pull

cd Day2/metallb
oc apply -f metallb.yml
```

Now you can check your loadbalancer service. Create a lb service

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

Accessing the nginx web server page via lb service

```
curl http://<load-balancer-service-external-ip>:8080
curl http://192.168.122.90:8080
```

Inof - What is Deployment?

- user applications are normally deployed as Deployment
- Deployment resource is managed by Deployment Controller
- When we deploy applications as deployment it automatically creates the below
    - Deployment

        - ReplicaSet (Per application version)
            - Pod1
            - Pod2
            - Pod3
- the deployment controller doesn't assure that always the Pods are distributed equally in all nodes

Info - What is DaemonSet?

- If we deploy our applications as Daemonset, then the DaemonSet Controller creates pods that matches the number of nodes in the openshift cluster.
- For instance, openshift runs one kube-proxy Pod in every node to support services (load-balancing)
- For instance, openshift runs one dns pod per node to support service discovery
- For instance, if we need to collect permance metrics all node, all pods running in a node. We have deploy one prometheus pod per node.

Lab - Deploying nginx into openshift in declarative style

```
oc project
oc get deploy
oc create deployment nginx --image=bitnami/nginx --replicas=3 -o yaml --
dry-run=client
oc create deployment nginx --image=bitnami/nginx --replicas=3 -o yaml --
dry-run=client > nginx-deploy.yml
oc apply -f nginx-deploy.yml

oc get deploy,rs,po
```

Expected output

```
[jegan@tektutor.org declarative-manifest-scripts]$ oc create deployment
nginx --image=bitnami/nginx --replicas=3 -o yaml --dry-run=client
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: nginx
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
```

```
    spec:
      containers:
      - image: bitnami/nginx
        name: nginx
        resources: {}
status: {}

oc create deployment nginx --image=bitnami/nginx --replicas=3 -o yaml --
dry-run=client > nginx-deploy.yml

[jegan@tektutor.org declarative-manifest-scripts]$ ls
nginx-deploy.yml

[jegan@tektutor.org declarative-manifest-scripts]$ oc apply -f nginx-
deploy.yml
deployment.apps/nginx created

[jegan@tektutor.org declarative-manifest-scripts]$ oc get deploy,rs,po
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx   0/3     3            0           4s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-bb865dc5f   3         3         0       4s

NAME                          READY   STATUS              RESTARTS   AGE
pod/nginx-bb865dc5f-bh24c   0/1     ContainerCreating   0          4s
pod/nginx-bb865dc5f-nmmq7   0/1     ContainerCreating   0          4s
pod/nginx-bb865dc5f-vm5cx   0/1     ContainerCreating   0          4s

[jegan@tektutor.org declarative-manifest-scripts]$ oc get po
NAME                    READY   STATUS    RESTARTS   AGE
nginx-bb865dc5f-bh24c   1/1     Running   0          2m2s
nginx-bb865dc5f-nmmq7   1/1     Running   0          2m2s
nginx-bb865dc5f-vm5cx   1/1     Running   0          2m2s
```

Lab - Creating cluster-ip internal service using declarative manifests yaml code

```
cd ~/openshift-may-2024
git pull
cd Day2/declarative-manifest-scripts

oc expose deploy/nginx --type=ClusterIP --port=8080 -o yaml --dry-
run=client
oc expose deploy/nginx --type=ClusterIP --port=8080 -o yaml --dry-
run=client > nginx-clusterip-svc.yml

oc apply -f nginx-clusterip-svc.yml

oc get svc
```

Expected output

```
[jegan@tektutor.org declarative-manifest-scripts]$ oc expose deploy/nginx --type=ClusterIP --port=8080 -o yaml --dry-run=client
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: nginx
  name: nginx
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    app: nginx
  type: ClusterIP
status:
  loadBalancer: {}

[jegan@tektutor.org declarative-manifest-scripts]$ oc expose deploy/nginx --type=ClusterIP --port=8080 -o yaml --dry-run=client > nginx-clusterip-svc.yml

[jegan@tektutor.org declarative-manifest-scripts]$ oc apply -f nginx-clusterip-svc.yml
service/nginx created

[jegan@tektutor.org declarative-manifest-scripts]$ oc get svc
NAME    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
nginx   ClusterIP   172.30.155.144           8080/TCP   3s
```

Lab - Creating nodeport external service using declarative manifests yaml code

```
cd ~/openshift-may-2024
git pull
cd Day2/declarative-manifest-scripts

oc expose deploy/nginx --type=NodePort --port=8080 -o yaml --dry-run=client
oc expose deploy/nginx --type=NodePort --port=8080 -o yaml --dry-run=client > nginx-nodeport-svc.yml

oc delete -f nginx-clusterip-svc.yml

oc apply -f nginx-nodeport-svc.yml
```

Expected output

```
[jegan@tektutor.org declarative-manifest-scripts]$ oc expose deploy/nginx --type=NodePort --port=8080 -o yaml --dry-run=client
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: nginx
  name: nginx
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    app: nginx
  type: NodePort
status:
  loadBalancer: {}

[jegan@tektutor.org declarative-manifest-scripts]$ oc expose deploy/nginx --type=NodePort --port=8080 -o yaml --dry-run=client > nginx-nodeport-svc.yml

[jegan@tektutor.org declarative-manifest-scripts]$ ls
nginx-clusterip-svc.yml  nginx-deploy.yml  nginx-lb-svc.yml  nginx-nodeport-svc.yml

[jegan@tektutor.org declarative-manifest-scripts]$ oc delete -f nginx-clusterip-svc.yml
service "nginx" deleted

[jegan@tektutor.org declarative-manifest-scripts]$ oc apply -f nginx-nodeport-svc.yml
service/nginx created

[jegan@tektutor.org declarative-manifest-scripts]$ oc get svc
NAME    TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
nginx   NodePort   172.30.125.21           8080:31960/TCP   3s
```

Lab - Creating loadbalancer external service using declarative manifests yaml code

```
cd ~/openshift-may-2024
git pull
cd Day2/declarative-manifest-scripts
```

```
oc expose deploy/nginx --type=LoadBalancer --port=8080 -o yaml --dry-
run=client
oc expose deploy/nginx --type=LoadBalancer --port=8080 -o yaml --dry-
run=client > nginx-lb-svc.yml

oc delete -f nginx-nodeport-svc.yml
oc apply -f nginx-lb-svc.yml
```

Expected output

```
[jegan@tektutor.org declarative-manifest-scripts]$ oc expose deploy/nginx -
-type=LoadBalancer --port=8080 -o yaml --dry-run=client
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: nginx
  name: nginx
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    app: nginx
  type: LoadBalancer
status:
  loadBalancer: {}

[jegan@tektutor.org declarative-manifest-scripts]$ oc expose deploy/nginx -
-type=LoadBalancer --port=8080 -o yaml --dry-run=client > nginx-lb-svc.yml

[jegan@tektutor.org declarative-manifest-scripts]$ ls
nginx-clusterip-svc.yml  nginx-deploy.yml  nginx-lb-svc.yml  nginx-
nodeport-svc.yml

[jegan@tektutor.org declarative-manifest-scripts]$ oc delete -f nginx-
nodeport-svc.yml
service "nginx" deleted

[jegan@tektutor.org declarative-manifest-scripts]$ oc apply -f nginx-lb-
svc.yml
service/nginx created

[jegan@tektutor.org declarative-manifest-scripts]$ oc get svc
NAME    TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)
AGE
nginx   LoadBalancer   172.30.219.56   192.168.122.90   8080:32090/TCP   3s
```

Lab - Creating a route in declarative style

```
cd ~/openshift-may-2024
git pull

cd Day2/declarative-manifest-scripts

oc expose svc/nginx -o yaml --dry-run=client
oc expose svc/nginx -o yaml --dry-run=client > nginx-route.yml

oc get svc
oc apply -f nginx-route.yml
oc get route

curl http://nginx-jegan.apps.ocp4.tektutor.org.labs
```

Expected output

```
nginx-clusterip-svc.yml  nginx-deploy.yml  nginx-lb-svc.yml  nginx-
nodeport-svc.yml

[jegan@tektutor.org declarative-manifest-scripts]$ oc get svc
NAME    TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)
AGE
nginx   LoadBalancer   172.30.219.56   192.168.122.90   8080:32090/TCP
15m

[jegan@tektutor.org declarative-manifest-scripts]$ oc expose svc/nginx -o
yaml --dry-run=client
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  creationTimestamp: null
  labels:
    app: nginx
  name: nginx
spec:
  port:
    targetPort: 8080
  to:
    kind: ""
    name: nginx
    weight: null
status: {}
[jegan@tektutor.org declarative-manifest-scripts]$ oc expose svc/nginx -o
yaml --dry-run=client > nginx-route.yml

[jegan@tektutor.org declarative-manifest-scripts]$ oc apply -f nginx-
route.yml
route.route.openshift.io/nginx created
```

```
[jegan@tektutor.org declarative-manifest-scripts]$ oc get route
NAME    HOST/PORT                                  PATH   SERVICES   PORT
TERMINATION   WILDCARD
nginx   nginx-jegan.apps.ocp4.tektutor.org.labs          nginx      8080
None

[jegan@tektutor.org declarative-manifest-scripts]$ curl http://nginx-
jegan.apps.ocp4.tektutor.org.labs
<!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>
```

Lab - Deleting deployment, service and route in declarative style

```
cd ~/openshift-may-2024
git pull

cd Day2/declarative-manifest-scripts
oc get all

oc delete -f nginx-route.yml
oc delete -f nginx-lb-svc.yml
oc delete -f nginx-deploy.yml

oc get all
```

Expected output

```
[jegan@tektutor.org declarative-manifest-scripts]$ pwd
/home/jegan/openshift-may-2024/Day2/declarative-manifest-scripts

[jegan@tektutor.org declarative-manifest-scripts]$ ls -l
total 20
-rw-r--r-- 1 jegan jegan 245 May 15 14:46 nginx-clusterip-svc.yml
-rw-r--r-- 1 jegan jegan 392 May 15 14:45 nginx-deploy.yml
-rw-r--r-- 1 jegan jegan 248 May 15 14:49 nginx-lb-svc.yml
-rw-r--r-- 1 jegan jegan 244 May 15 14:50 nginx-nodeport-svc.yml
-rw-r--r-- 1 jegan jegan 219 May 15 15:14 nginx-route.yml

[jegan@tektutor.org declarative-manifest-scripts]$ oc delete -f nginx-
route.yml
route.route.openshift.io "nginx" deleted

[jegan@tektutor.org declarative-manifest-scripts]$ oc delete -f nginx-lb-
svc.yml
service "nginx" deleted

[jegan@tektutor.org declarative-manifest-scripts]$ oc delete -f nginx-
deploy.yml
deployment.apps "nginx" deleted

[jegan@tektutor.org declarative-manifest-scripts]$ oc get all
Warning: apps.openshift.io/v1 DeploymentConfig is deprecated in v4.14+,
unavailable in v4.10000+
No resources found in jegan namespace.
```

Lab - Declaratively perform deployment scale up/down

```
cd ~/openshift-may-2024
git pull
cd Day2/declarative-manifest-scripts

oc apply -f nginx-deploy.yml
oc get deploy,rs,po
```

Now you can edit the nginx-deploy.yml file and update the replicas values from '3' to '5' and save and apply the changes.

```
vim nginx-deploy.yml
oc apply -f nginx-deploy.yml
oc get po
```

Now you can edit the nginx-deploy.yml file and update the replicas values from '5' to '3' and save and apply the changes.

```
vim nginx-deploy.yml
oc apply -f nginx-deploy.yml
oc get po
```

Cleanup

```
oc delete -f nginx-deploy.yml
```

Lab - Deploying an application into openshift using a spring-boot application jar from your local machine

```
cd ~/openshift-may-2024
git pull
cd Day2/spring-ms

mvn clean package
```

Make sure to use the jar file found under target folder into openshift web console using deploy app via jar file.

Lab - Deploying application using new-app from an existing Docker Hub Container Image

If you already have project in your name, delete it. Replace 'jegan' with your name.

```
oc delete project jegan
```

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

```
oc new-project jegan
oc new-app bitnami/nginx:latest
```

Expected output

```
[jegan@tektutor.org Day2]$ 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

[jegan@tektutor.org Day2]$ oc new-app bitnami/nginx:latest
--> Found container image 8c08fc2 (37 hours old) from Docker Hub for
"bitnami/nginx:latest"

    * An image stream tag will be created as "nginx:latest" that will track
this image

--> Creating resources ...
    imagestream.image.openshift.io "nginx" created
    deployment.apps "nginx" created
    service "nginx" created
--> Success
    Application is not exposed. You can expose services to the outside
world by executing one or more of the commands below:
     'oc expose service/nginx'
    Run 'oc status' to view your app.

[jegan@tektutor.org Day2]$ oc expose service/nginx
route.route.openshift.io/nginx exposed
```

Lab - Deploying Replicaset in declarative style

```
cd ~/openshift-may-2024
git pull
cd Day2/declarative-manifest-scripts

oc apply -f nginx-rs.yml
oc get deploy,rs,po

oc scale rs/nginx-rs --replicas=5
oc get po
oc scale rs/nginx-rs --replicas=3
oc get po

oc delete -f nginx-rs.yml
```

Expected output

```
[jegan@tektutor.org declarative-manifest-scripts]$ pwd
/home/jegan/openshift-may-2024/Day2/declarative-manifest-scripts
```

```
[jegan@tektutor.org declarative-manifest-scripts]$ ls -l
total 32
-rw-r--r-- 1 jegan jegan 488 May 15 15:35 deploy.yaml
-rw-r--r-- 1 jegan jegan 970 May 15 15:39 devfile.yaml
-rw-r--r-- 1 jegan jegan 245 May 15 14:46 nginx-clusterip-svc.yml
-rw-r--r-- 1 jegan jegan 392 May 15 15:20 nginx-deploy.yml
-rw-r--r-- 1 jegan jegan 248 May 15 14:49 nginx-lb-svc.yml
-rw-r--r-- 1 jegan jegan 244 May 15 14:50 nginx-nodeport-svc.yml
-rw-r--r-- 1 jegan jegan 219 May 15 15:14 nginx-route.yml
-rw-r--r-- 1 jegan jegan 348 May 15 17:37 nginx-rs.yml

[jegan@tektutor.org declarative-manifest-scripts]$ oc apply -f nginx-rs
error: the path "nginx-rs" does not exist
[jegan@tektutor.org declarative-manifest-scripts]$ oc apply -f nginx-rs.yml
replicaset.apps/nginx-rs created

[jegan@tektutor.org declarative-manifest-scripts]$ oc get deploy,rs,po
NAME                         DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-rs   3         3         3       55s

NAME                 READY   STATUS    RESTARTS   AGE
pod/nginx-rs-5hjp8   1/1     Running   0          55s
pod/nginx-rs-cxtmt   1/1     Running   0          55s
pod/nginx-rs-xcsfz   1/1     Running   0          55s

[jegan@tektutor.org declarative-manifest-scripts]$ oc scale
replicaset.apps/nginx-rs --replicas=5
replicaset.apps/nginx-rs scaled

[jegan@tektutor.org declarative-manifest-scripts]$ oc get po -w
NAME             READY   STATUS    RESTARTS   AGE
nginx-rs-2q4k9   1/1     Running   0          3s
nginx-rs-5hjp8   1/1     Running   0          116s
nginx-rs-cxtmt   1/1     Running   0          116s
nginx-rs-jk8np   1/1     Running   0          3s
nginx-rs-xcsfz   1/1     Running   0          116s

^C[jegan@tektutor.org declarative-manifest-scripts]$ oc scale
replicaset.apps/nginx-rs --replicas=3
replicaset.apps/nginx-rs scaled

[jegan@tektutor.org declarative-manifest-scripts]$ oc get po -w
NAME             READY   STATUS    RESTARTS   AGE
nginx-rs-5hjp8   1/1     Running   0          2m7s
nginx-rs-cxtmt   1/1     Running   0          2m7s
nginx-rs-xcsfz   1/1     Running   0          2m7s
^C[jegan@tektutor.org declarative-manifest-scripts]$ oc delete -f nginx-
rs.yml
replicaset.apps "nginx-rs" deleted
```

Lab - Creating a Pod declaratively

```
cd ~/openshift-may-2024
git pull
cd Day2/declarative-manifest-scripts
oc get po
oc apply -f pod.yml
oc get po

oc delete -f pod.yml
```

Expected output

```
[jegan@tektutor.org declarative-manifest-scripts]$ oc apply -f pod.yml
Warning: would violate PodSecurity "restricted:v1.24":
allowPrivilegeEscalation != false (container "my-nginx-container" must set
securityContext.allowPrivilegeEscalation=false), unrestricted capabilities
(container "my-nginx-container" must set securityContext.capabilities.drop=
["ALL"]), runAsNonRoot != true (pod or container "my-nginx-container" must
set securityContext.runAsNonRoot=true), seccompProfile (pod or container
"my-nginx-container" must set securityContext.seccompProfile.type to
"RuntimeDefault" or "Localhost")
pod/my-nginx-pod created

[jegan@tektutor.org declarative-manifest-scripts]$ oc get po
NAME             READY   STATUS    RESTARTS   AGE
my-nginx-pod     1/1     Running   0          5s
nginx-rs-lgj94   1/1     Running   0          5m16s
nginx-rs-n2vzr   1/1     Running   0          5m16s
nginx-rs-v8r2c   1/1     Running   0          5m16s

[jegan@tektutor.org declarative-manifest-scripts]$ oc delete pod my-nginx-
pod
pod "my-nginx-pod" deleted

[jegan@tektutor.org declarative-manifest-scripts]$ oc get po
NAME             READY   STATUS    RESTARTS   AGE
nginx-rs-lgj94   1/1     Running   0          5m41s
nginx-rs-n2vzr   1/1     Running   0          5m41s
nginx-rs-v8r2c   1/1     Running   0          5m41s
```