Namespace

Create namespace

Option 1: Imperatively

```
kubectl create namespace mynamespace
```

Options 2: Declaratively

```
apiVersion: v1
kind: Namespace
metadata:
  name: mynamespace
```

Create Pod in a namespace

```
root@ip-172-31-19-105:~/demo/pod# cat namespace-pod.yaml
apiVersion: v1                  # version of api-resource
kind: Pod                       # api-resourse
metadata:                       # information of the api-resource
  name: namespaced-pod
  namespace: mynamespace
spec:                            # configuration of the api-resource
  containers:
  - name: nginxcontainer
    image: nginx
```

List the pods in a namespace

```
kubectl get pod -n mynamespace
```

Delete Namespace

```
kubectl delete namespaces mynamespace
```

Labels and Selectors

Labels

Apply Labels

Apply the label “environment=production” label to worker 1 Node
Apply the label “environment=staging” label to worker 2 Node
Apply the label “location=india” label to all Nodes

Reference commands:
```
kubectl label nodes <one of the nodes’ name> environment=production
kubectl label nodes <the other nodes’ name> location=india
kubectl label nodes <the other nodes’ name> color=green
```

Get nodes with label information

```
kubectl get nodes --show-labels
```

Delete a label

```
kubectl label node master environment-
```

Update a label

```
kubectl label node master --overwrite location=usa
```

Selector

Select all the nodes with environment set to production
```
kubectl get nodes -l environment=production
```

Scheduling

nodeName

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    app: myapp
  name: pod
spec:
  nodeName: worker2  #Desired Node Name
  containers:
  - image: nginx
    name: pod
    ports:
    - containerPort: 80
```

nodeSelector

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  name: node-selector-pod
spec:
  nodeSelector:
      color: green  # Node labels
  containers:
  - image: nginx
    name: pod
    ports:
    - containerPort: 80
```

nodeAffinity

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  name: required-affinity-pod
spec:
  containers:
  - image: nginx
    name: pod
    ports:
    - containerPort: 80
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
```

```
        - matchExpressions:
          - key: color
            operator: In  # In, NotIn, Exists, DoesNotExist, Gt, Lt
            values:
            - red
            - green
```

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  name: preferred-affinity-pod
spec:
  containers:
  - image: nginx
    name: pod
    ports:
    - containerPort: 80
  affinity:
    nodeAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        preference:
          matchExpressions:
          - key: color
            operator: In
            values:
            - purple
```

Taints and Tolerations

Effects:

1. NoSchedule
2. PreferNoSchedule
3. NoExecute

Taint a node:

```
kubectl taint node worker2 type=gpu:NoSchedule
```

Tolerate the taint in a Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: test-taint-pod
spec:
  containers:
  - name: nginxcontainer
    image: nginx
```

```yaml
tolerations:
- key: type
  operator: Equal
  value: gpu
```

Untaint a node:

```
kubectl taint node ip-172-31-19-129 type=gpu:NoSchedule-
```

Logs

Print logs of specific containers in a pod:

```
kubectl logs [-f] <Podname> [containername]
kubectl logs -f multi-container <containername>
```

Print logs of all containers in a Pod

```
kubectl logs -f --all-containers multi-container
```

Exec

single container pod

```
kubectl exec declarative-pod -- printenv
```

interactively into a single container pod

```
kubectl exec -it declarative-pod -- /bin/sh
```

multi container pod (defaults to the first container)

```
kubectl exec multi-container-pod -- printenv
```

specific container in a multi container pod

```
kubectl exec multi-container-pod -c c2 -- printenv
```

Customization

Environment variables

```
apiVersion: v1
kind: Pod
metadata:
  name: envpod-declarative
spec:
  containers:
  - env:
    - name: KEY
      value: VALUE
    - name: KEY2
      value: Val2
    image: nginx
    name: envpod
    ports:
    - containerPort: 80
```

Custom Commands

```
apiVersion: v1
kind: Pod
metadata:
  name: customcommand
spec:
  containers:
  - image: alpine
    name: alpine
    command: ['sh','-c','echo "Hello Kubernetes" && sleep 100']
```

Resource Limits

```
apiVersion: v1
kind: Pod
metadata:
  name: resource-limit
spec:
  containers:
  - image: nginx
    name: nginx
    resources:
     limits:
      cpu: 0.5
```

Controllers

Replicaset

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: my-rs
spec:
 replicas: 3
 template:
  metadata:
   labels:
    app: my-rs
  spec:
   containers:
   - name: nginx
     image: nginx
     ports:
     - containerPort: 80
 selector:
  matchLabels:
   app: my-rs
```

Deployments

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: my-deployment
spec:
 replicas: 3
 template:
  metadata:
   labels:
    app: my-dep
  spec:
   containers:
   - name: nginx
     image: nginx:1.19
     ports:
     - containerPort: 80
 selector:
  matchLabels:
   app: my-dep
```

List the deployment:
```
kubectl get deployments
```
List pods in the deployment:
```
kubectl get pod
```
List the replicasets (which are part of deployment)
```
kubectl get rs
```
Get details of a Deployment
```
kubectl describe deployments.apps my-deployment
```
Scale a deployment
```
kubectl scale deployment my-deployment --replicas=5
```
Rollout a new version
```
kubectl set image deployment my-deployment nginx=nginx:1.20 --record
```
Check the rollout history
```
kubectl rollout history deployment my-deployment
```
Rollout another new version
```
kubectl set image deployment my-deployment nginx=nginx:1.21 --record
```
Check the rollout history
```
kubectl rollout history deployment my-deployment
```
Rollback to a specific version
```
kubectl rollout undo deployment my-deployment --to-revision 1
```

DaemonSet

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: my-ds
spec:
  template:
    metadata:
      labels:
        app: ds
    spec:
      containers:
      - image: nginx
        name: nginx
  selector:
    matchLabels:
      app: ds
```

List Daemonsets
```
kubectl get ds
```

Describe Daemonset
```
kubectl describe daemonsets.apps my-ds
```

Check the pods in the daemonset
```
kubectl get pods -o wide
```

Services

ClusterIP

```
kubectl expose deployment my-dep --name my-svc --port 80
```

OR

```
apiVersion: v1
kind: Service
metadata:
  name: my-declarative-svc
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: my-dep
```

NodePort

```
kubectl expose deployment my-dep --name my-nodeport-svc --port 80 --type
NodePort
```

OR

```
apiVersion: v1
kind: Service
metadata:
  name: nodeport-svc
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
    nodePort: 30001
  selector:
    app: my-dep
  type: NodePort
```

LoadBalancer

```
kubectl expose deployment my-dep --name my-lb-svc --port 80 --type LoadBalancer
```

List All services

```
kubectl get svc
```