# Docker Certified Associate (DCA)

Create by: **Hadi Tayanloo**

Phone : **+98-912-8387233**

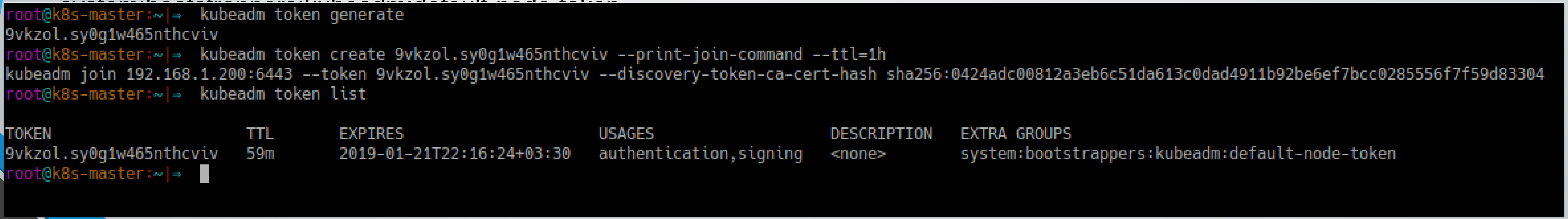
Linkedin: linkedin.com/in/htayanloo/

## Session 9

- Kubernetes Managment
- Kubernetes Pods
- Kubernetes Resource
- Kubernetes Volume

## Kubernetes Managment : Generate New token

- root@k8s-master:~|⇒  kubeadm token generate
- 9vkzol.sy0g1w465nthcviv
- root@k8s-master:~|⇒  kubeadm token create 9vkzol.sy0g1w465nthcviv --print-join-command --ttl=1h
- kubeadm join 192.168.1.200:6443 --token 9vkzol.sy0g1w465nthcviv --discovery-token-ca-cert-hash sha256:0424adc00812a3eb6c51da613c0dad4911b92be6ef7bcc0285556f7f59d83304
- root@k8s-master:~|⇒  kubeadm token list
- TOKEN                     TTL       EXPIRES                     USAGES                   DESCRIPTION   EXTRA GROUPS
- 9vkzol.sy0g1w465nthcviv   59m       2019-01-21T22:16:24+03:30   authentication,signing   <none>        system:bootstrappers:kubeadm:default-node-token

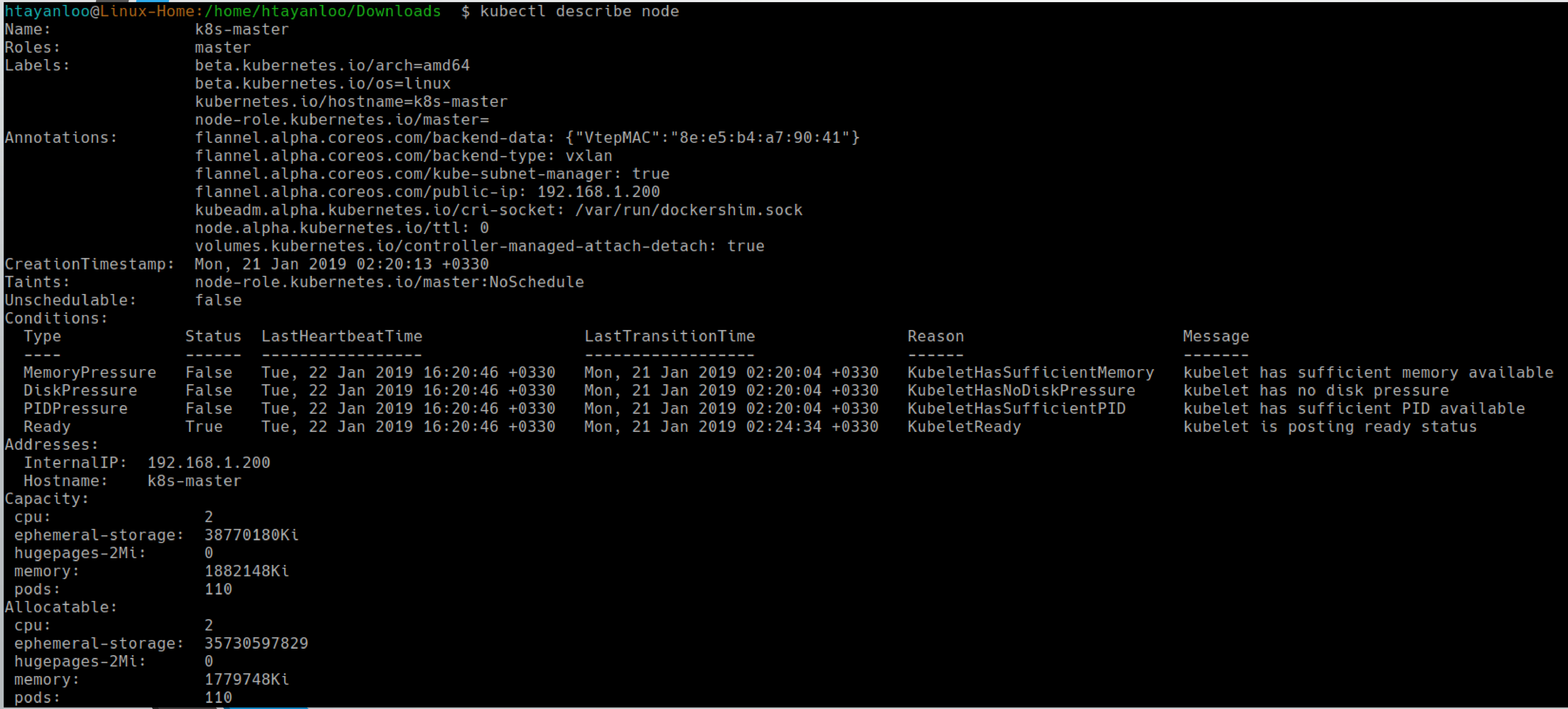


```
root@k8s-master:~|⇒  kubeadm token generate
9vkzol.sy0g1w465nthcviv
root@k8s-master:~|⇒  kubeadm token create 9vkzol.sy0g1w465nthcviv --print-join-command --ttl=1h
kubeadm join 192.168.1.200:6443 --token 9vkzol.sy0g1w465nthcviv --discovery-token-ca-cert-hash sha256:0424adc00812a3eb6c51da613c0dad4911b92be6ef7bcc0285556f7f59d83304
root@k8s-master:~|⇒  kubeadm token list

TOKEN                     TTL       EXPIRES                     USAGES                   DESCRIPTION   EXTRA GROUPS
9vkzol.sy0g1w465nthcviv   59m       2019-01-21T22:16:24+03:30   authentication,signing   <none>        system:bootstrappers:kubeadm:default-node-token
root@k8s-master:~|⇒
```

## The conditions field describes the status of all Running nodes.

- **OutOfDisk**
  - **True** if there is insufficient free space on the node for adding new pods, otherwise **False**
- **Ready**
  - **True** if the node is healthy and ready to accept pods, **False** if the node is not healthy and is not accepting pods, and Unknown if the node controller has not heard from the node in the last node-monitor-grace-period (default is 40 seconds)
- **MemoryPressure**
  - **True** if pressure exists on the node memory – that is, if the node memory is low; otherwise **False**
- **PIDPressure**
  - **True** if pressure exists on the processes – that is, if there are too many processes on the node; otherwise **False**
- **DiskPressure**
  - **True** if pressure exists on the disk size – that is, if the disk capacity is low; otherwise **False**
- **NetworkUnavailable**
  - **True** if the network for the node is not correctly configured, otherwise **False**
-

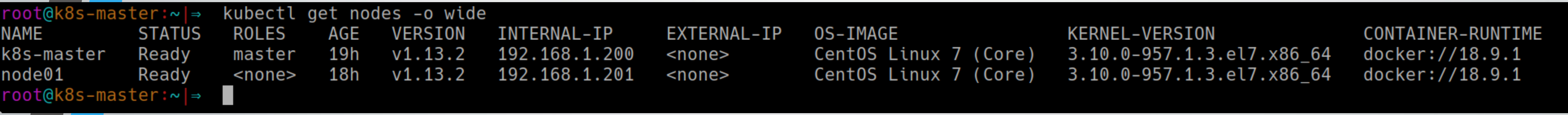
The conditions field describes the status of all Running nodes.



```
htayanloo@Linux-Home:/home/htayanloo/Downloads  $ kubectl describe node
Name:               k8s-master
Roles:              master
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/hostname=k8s-master
                    node-role.kubernetes.io/master=
Annotations:        flannel.alpha.coreos.com/backend-data: {"VtepMAC":"8e:e5:b4:a7:90:41"}
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 192.168.1.200
                    kubeadm.alpha.kubernetes.io/cri-socket: /var/run/dockershim.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Mon, 21 Jan 2019 02:20:13 +0330
Taints:             node-role.kubernetes.io/master:NoSchedule
Unschedulable:      false
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----             ------  -----------------                 ------------------                ------                       -------
  MemoryPressure   False   Tue, 22 Jan 2019 16:20:46 +0330   Mon, 21 Jan 2019 02:20:04 +0330   KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure     False   Tue, 22 Jan 2019 16:20:46 +0330   Mon, 21 Jan 2019 02:20:04 +0330   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure      False   Tue, 22 Jan 2019 16:20:46 +0330   Mon, 21 Jan 2019 02:20:04 +0330   KubeletHasSufficientPID      kubelet has sufficient PID available
  Ready            True    Tue, 22 Jan 2019 16:20:46 +0330   Mon, 21 Jan 2019 02:24:34 +0330   KubeletReady                 kubelet is posting ready status
Addresses:
  InternalIP:  192.168.1.200
  Hostname:    k8s-master
Capacity:
 cpu:                2
 ephemeral-storage:  38770180Ki
 hugepages-2Mi:      0
 memory:             1882148Ki
 pods:               110
Allocatable:
 cpu:                2
 ephemeral-storage:  35730597829
 hugepages-2Mi:      0
 memory:             1779748Ki
 pods:               110
```

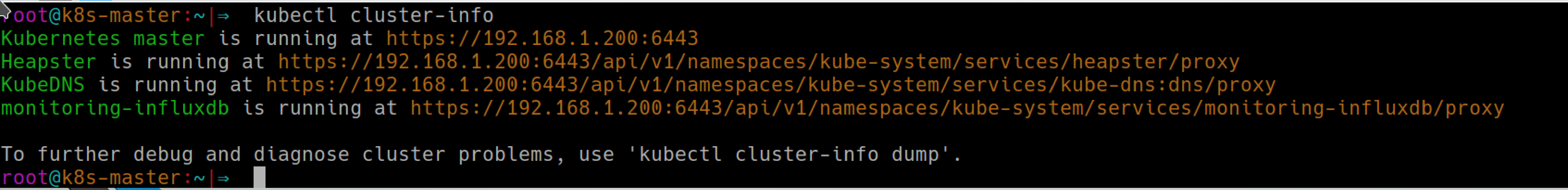
## Kubernetes node list

- root@k8s-master:~|⇒  kubectl get nodes -o wide
- NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME
- k8s-master Ready master 19h v1.13.2 192.168.1.200 <none> CentOS Linux 7 (Core) 3.10.0-957.1.3.el7.x86_64 docker://18.9.1
- node01 Ready <none> 18h v1.13.2 192.168.1.201 <none> CentOS Linux 7 (Core) 3.10.0-957.1.3.el7.x86_64 docker://18.9.1
-
-



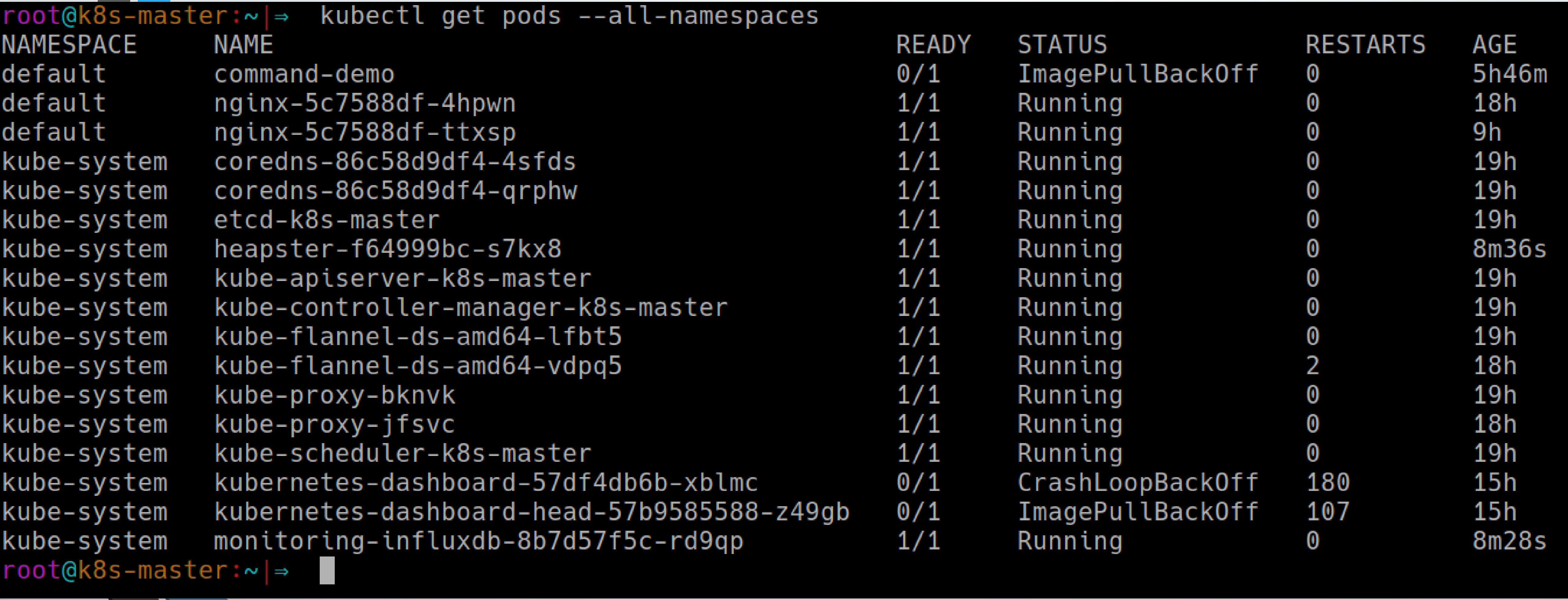
```
root@k8s-master:~|⇒  kubectl get nodes -o wide
NAME         STATUS   ROLES    AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE                KERNEL-VERSION               CONTAINER-RUNTIME
k8s-master   Ready    master   19h   v1.13.2   192.168.1.200   <none>        CentOS Linux 7 (Core)   3.10.0-957.1.3.el7.x86_64   docker://18.9.1
node01       Ready    <none>   18h   v1.13.2   192.168.1.201   <none>        CentOS Linux 7 (Core)   3.10.0-957.1.3.el7.x86_64   docker://18.9.1
root@k8s-master:~|⇒
```

## Kubernetes Cluster info



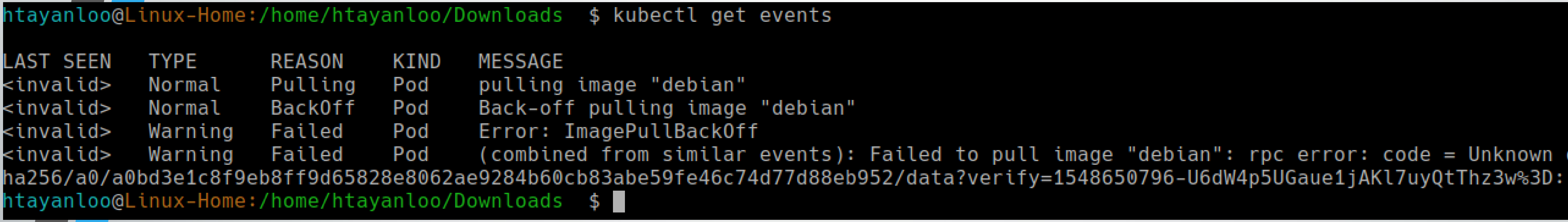
```
root@k8s-master:~|⇒  kubectl cluster-info
Kubernetes master is running at https://192.168.1.200:6443
Heapster is running at https://192.168.1.200:6443/api/v1/namespaces/kube-system/services/heapster/proxy
KubeDNS is running at https://192.168.1.200:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
monitoring-influxdb is running at https://192.168.1.200:6443/api/v1/namespaces/kube-system/services/monitoring-influxdb/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
root@k8s-master:~|⇒
```

## Get list of all Pods from any name space



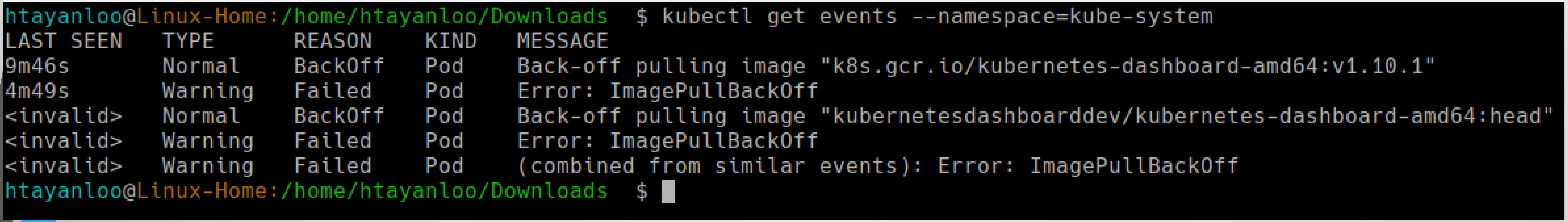
```
root@k8s-master:~|⇒  kubectl get pods --all-namespaces
NAMESPACE     NAME                                         READY   STATUS             RESTARTS   AGE
default       command-demo                                 0/1     ImagePullBackOff   0          5h46m
default       nginx-5c7588df-4hpwn                         1/1     Running            0          18h
default       nginx-5c7588df-ttxsp                         1/1     Running            0          9h
kube-system   coredns-86c58d9df4-4sfds                     1/1     Running            0          19h
kube-system   coredns-86c58d9df4-qrphw                     1/1     Running            0          19h
kube-system   etcd-k8s-master                              1/1     Running            0          19h
kube-system   heapster-f64999bc-s7kx8                      1/1     Running            0          8m36s
kube-system   kube-apiserver-k8s-master                    1/1     Running            0          19h
kube-system   kube-controller-manager-k8s-master           1/1     Running            0          19h
kube-system   kube-flannel-ds-amd64-lfbt5                  1/1     Running            0          19h
kube-system   kube-flannel-ds-amd64-vdpq5                  1/1     Running            2          18h
kube-system   kube-proxy-bknvk                             1/1     Running            0          19h
kube-system   kube-proxy-jfsvc                             1/1     Running            0          18h
kube-system   kube-scheduler-k8s-master                    1/1     Running            0          19h
kube-system   kubernetes-dashboard-57df4db6b-xblmc         0/1     CrashLoopBackOff   180        15h
kube-system   kubernetes-dashboard-head-57b9585588-z49gb   0/1     ImagePullBackOff   107        15h
kube-system   monitoring-influxdb-8b7d57f5c-rd9qp          1/1     Running            0          8m28s
root@k8s-master:~|⇒  
```

# kubectl get events



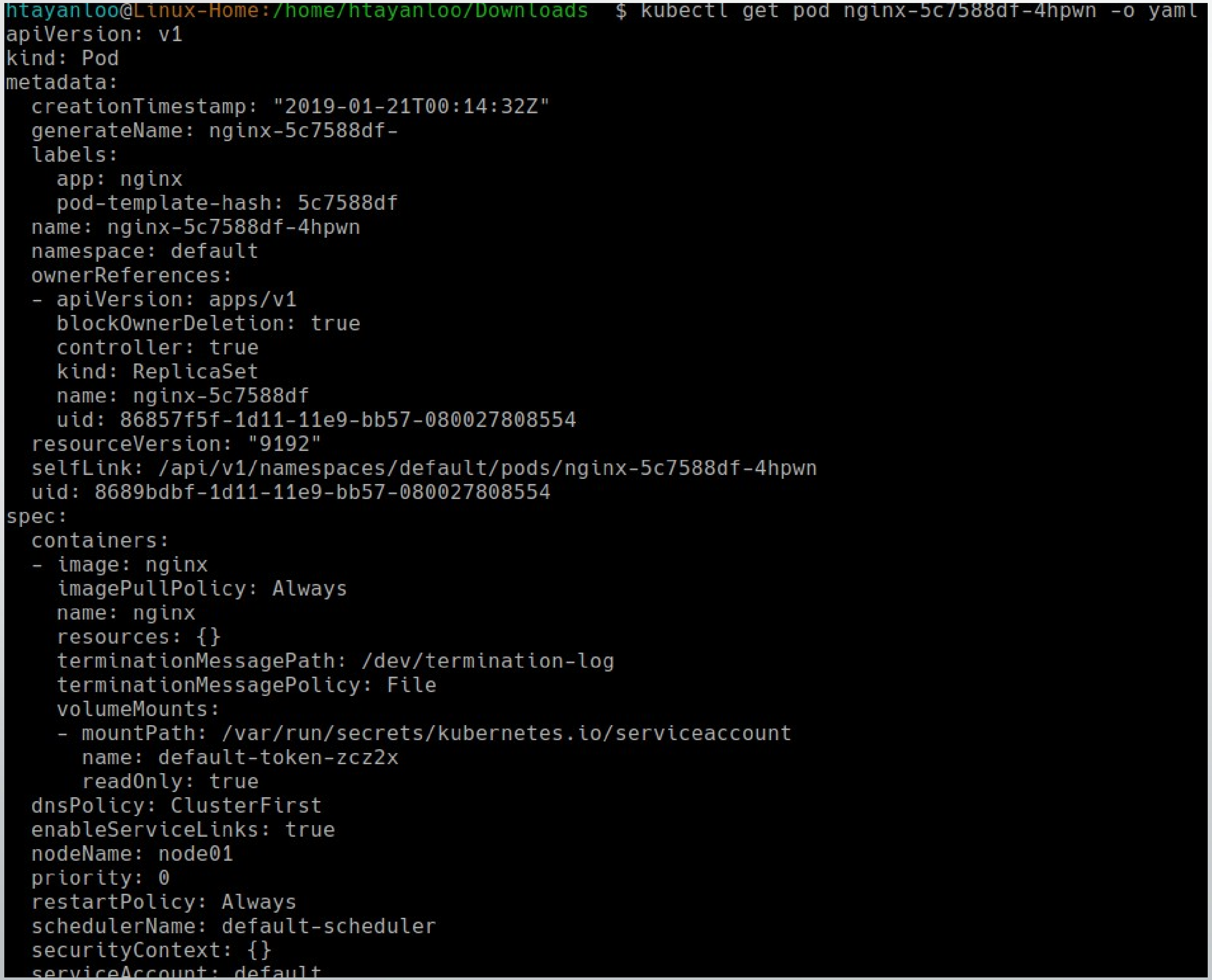
```
htayanloo@Linux-Home:/home/htayanloo/Downloads  $ kubectl get events

LAST SEEN   TYPE      REASON    KIND   MESSAGE
<invalid>   Normal    Pulling   Pod    pulling image "debian"
<invalid>   Normal    BackOff   Pod    Back-off pulling image "debian"
<invalid>   Warning   Failed    Pod    Error: ImagePullBackOff
<invalid>   Warning   Failed    Pod    (combined from similar events): Failed to pull image "debian": rpc error: code = Unknown
ha256/a0/a0bd3e1c8f9eb8ff9d65828e8062ae9284b60cb83abe59fe46c74d77d88eb952/data?verify=1548650796-U6dW4p5UGaue1jAKl7uyQtThz3w%3D:
htayanloo@Linux-Home:/home/htayanloo/Downloads  $
```

# kubectl get events --namespace=my-namespace



```
htayanloo@Linux-Home:/home/htayanloo/Downloads  $ kubectl get events --namespace=kube-system
LAST SEEN    TYPE       REASON     KIND    MESSAGE
9m46s        Normal     BackOff    Pod     Back-off pulling image "k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.1"
4m49s        Warning    Failed     Pod     Error: ImagePullBackOff
<invalid>    Normal     BackOff    Pod     Back-off pulling image "kubernetesdashboarddev/kubernetes-dashboard-amd64:head"
<invalid>    Warning    Failed     Pod     Error: ImagePullBackOff
<invalid>    Warning    Failed     Pod     (combined from similar events): Error: ImagePullBackOff
htayanloo@Linux-Home:/home/htayanloo/Downloads  $
```

# kubectl get pod POD_NAME -o yaml



```
htayanloo@Linux-Home:/home/htayanloo/Downloads  $ kubectl get pod nginx-5c7588df-4hpwn -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2019-01-21T00:14:32Z"
  generateName: nginx-5c7588df-
  labels:
    app: nginx
    pod-template-hash: 5c7588df
  name: nginx-5c7588df-4hpwn
  namespace: default
  ownerReferences:
  - apiVersion: apps/v1
    blockOwnerDeletion: true
    controller: true
    kind: ReplicaSet
    name: nginx-5c7588df
    uid: 86857f5f-1d11-11e9-bb57-080027808554
  resourceVersion: "9192"
  selfLink: /api/v1/namespaces/default/pods/nginx-5c7588df-4hpwn
  uid: 8689bdbf-1d11-11e9-bb57-080027808554
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: default-token-zcz2x
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: node01
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
```

# POD and Container

## Getting a shell to a Container

```
kubectl create -f https://k8s.io/examples/application/shell-demo.yaml
kubectl get pod shell-demo

kubectl exec -it shell-demo -- /bin/bash

root@shell-demo:/# ls /
```

## Running individual commands in a Container

```
kubectl exec shell-demo env

kubectl exec shell-demo ps aux
kubectl exec shell-demo ls /
kubectl exec shell-demo cat /proc/1/mounts
```

## Opening a shell when a Pod has more than one Container

If a Pod has more than one Container, use --container or -c to specify a Container in the kubectl exec command. For example, suppose you have a Pod named my-pod, and the Pod has two containers named main-app and helper-app. The following command would open a shell to the main-app Container.

```
kubectl exec -it my-pod --container main-app -- /bin/bash
```

## Assign Memory Resources to Containers and Pods

# Assign Memory Resources to Containers and Pods

## Senario-1

1. kubectl create namespace mem-example
2. kubectl create -f https://k8s.io/examples/pods/resource/memory-request-limit.yaml --namespace=mem-example
3. kubectl get pod memory-demo --namespace=mem-example
4. kubectl get pod memory-demo --output=yaml --namespace=mem-example
5. kubectl top pod memory-demo --namespace=mem-example
6. kubectl delete pod memory-demo --namespace=mem-example

## Exceed a Container’s memory limit

### Senario-2

1. kubectl create -f https://k8s.io/examples/pods/resource/memory-request-limit-2.yaml --namespace=mem-example
2. kubectl get pod memory-demo-2 –namespace=mem-example
3. kubectl get pod memory-demo-2 --output=yaml --namespace=mem-example
4. kubectl get pod memory-demo-2 –namespace=mem-example
5. kubectl describe pod memory-demo-2 –namespace=mem-example
6. kubectl describe nodes
7. kubectl delete pod memory-demo-2 --namespace=mem-example

## Specify a memory request that is too big for your Nodes

### Senario-3

1. kubectl create -f https://k8s.io/examples/pods/resource/memory-request-limit-3.yaml –namespace=mem-example
2. kubectl get pod memory-demo-3 –namespace=mem-example
3. kubectl describe pod memory-demo-3 –namespace=mem-example
4. kubectl delete pod memory-demo-3 --namespace=mem-example

## CleanUp Senario

- kubectl delete namespace mem-example
-

## Assign CPU Resources to Containers and Pods

1. kubectl create namespace cpu-example

## Assign CPU Resources to Containers and Pods

### Senario-4

1. kubectl create -f https://k8s.io/examples/pods/resource/cpu-request-limit.yaml --namespace=cpu-example
2. kubectl get pod cpu-demo --namespace=cpu-example
3. kubectl get pod cpu-demo --output=yaml --namespace=cpu-example
4. kubectl top pod cpu-demo –namespace=cpu-example
5. kubectl delete pod cpu-demo --namespace=cpu-example
6.

## Specify a CPU request that is too big for your Nodes

### Senario-5

1. kubectl create -f https://k8s.io/examples/pods/resource/cpu-request-limit-2.yaml --namespace=cpu-example
2. kubectl get pod cpu-demo-2 --namespace=cpu-example
3. kubectl describe pod cpu-demo-2 --namespace=cpu-example
4. kubectl delete pod cpu-demo-2 --namespace=cpu-example

## CleanUp Senario

- kubectl delete namespace cpu-example
-
-

## Configure Quality of Service for Pods

1. QoS classes:
   1. Guaranteed
   2. Burstable
   3. BestEffort

1. kubectl create namespace qos-example

## Create a Pod that gets assigned a QoS class of Guaranteed

1. For a Pod to be given a QoS class of Guaranteed:
    1. Every Container in the Pod must have a memory limit and a memory request, and they must be the same.
    2. Every Container in the Pod must have a CPU limit and a CPU request, and they must be the same.

# Assign CPU Resources to Containers and Pods

## Senario-6

1. kubectl create -f https://k8s.io/examples/pods/qos/qos-pod.yaml --namespace=qos-example
2. kubectl get pod qos-demo --namespace=qos-example –output=yaml
3. kubectl delete pod qos-demo --namespace=qos-example

## Create a Pod that gets assigned a QoS class of Burstable

1. A Pod is given a QoS class of Burstable if:

    1. The Pod does not meet the criteria for QoS class Guaranteed.
    2. At least one Container in the Pod has a memory or CPU request.

## Create a Pod that gets assigned a QoS class of Burstable

### Senario-7

1. kubectl create -f https://k8s.io/examples/pods/qos/qos-pod-2.yaml --namespace=qos-example

2. kubectl create -f https://k8s.io/examples/pods/qos/qos-pod-2.yaml --namespace=qos-example

3. kubectl delete pod qos-demo-2 --namespace=qos-example

## Create a Pod that gets assigned a QoS class of BestEffort

1. A Pod is given a QoS class of BestEffort if:

    1. For a Pod to be given a QoS class of BestEffort, the Containers in the Pod must not have any memory or CPU limits or requests.

## Create a Pod that gets assigned a QoS class of BestEffort

### Senario-8

1. kubectl create -f https://k8s.io/examples/pods/qos/qos-pod-3.yaml --namespace=qos-example

2. kubectl get pod qos-demo-3 --namespace=qos-example --output=yaml

3. kubectl delete pod qos-demo-3 --namespace=qos-example

## Create a Pod that has two Containers

### Senario-9

1. kubectl create -f https://k8s.io/examples/pods/qos/qos-pod-4.yaml --namespace=qos-example

2. kubectl get pod qos-demo-4 --namespace=qos-example --output=yaml

3. kubectl delete pod qos-demo-4 --namespace=qos-example

## CleanUp Senario

- kubectl delete namespace qos-example

## Configure a Pod to Use a Volume for Storage

## Configure a volume for a Pod

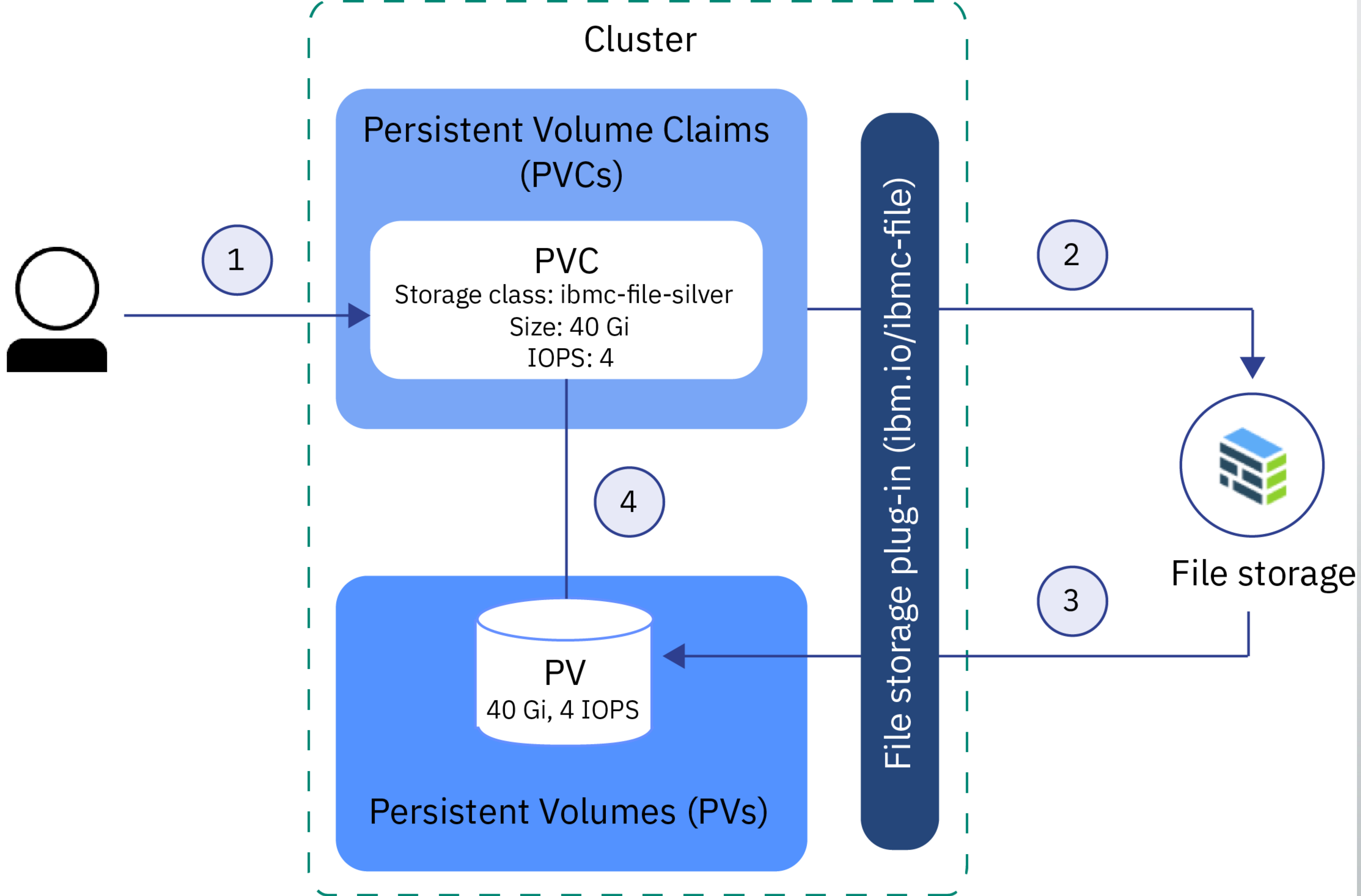
### Senario-10

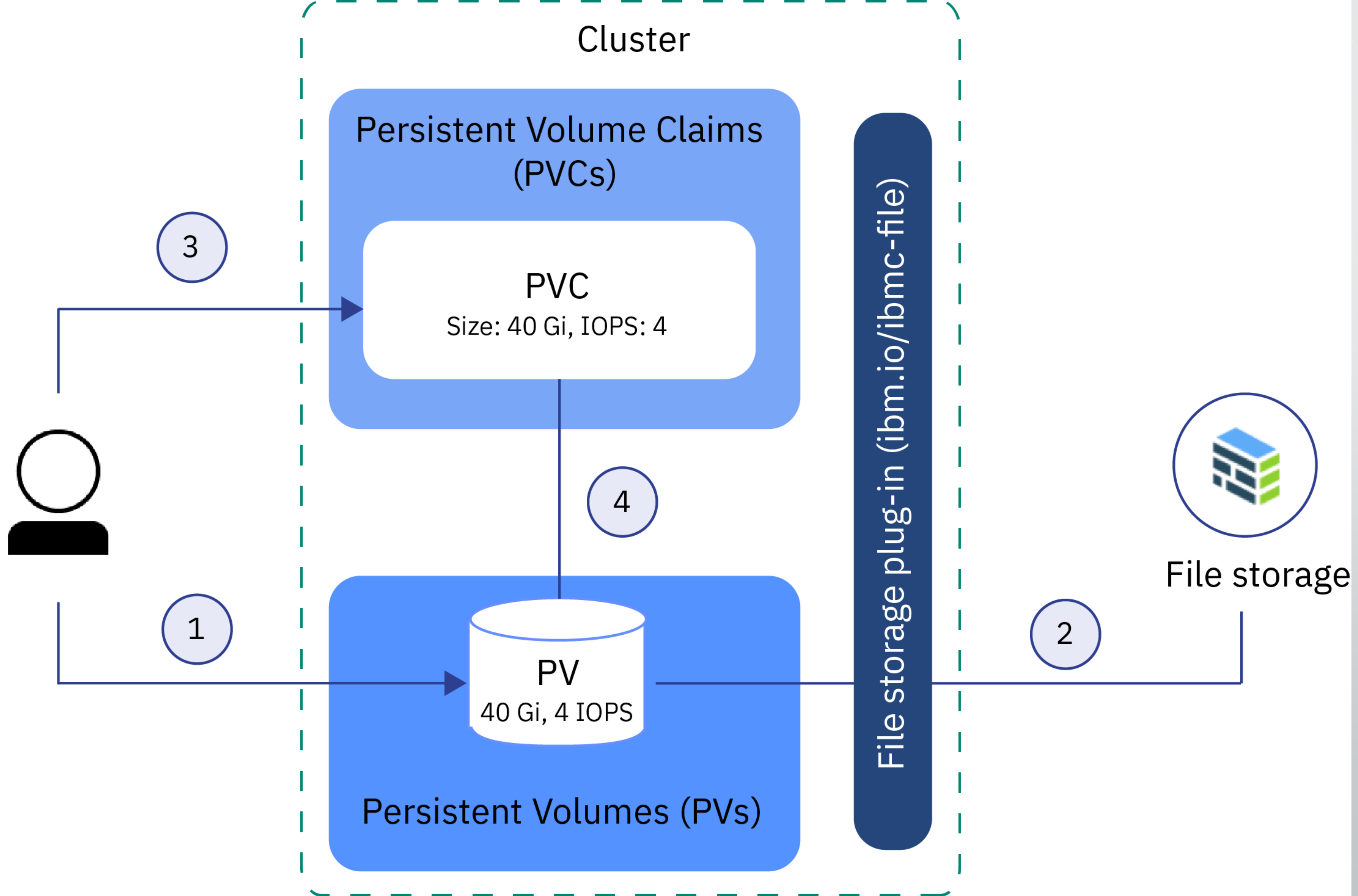
1. kubectl create -f https://k8s.io/examples/pods/storage/redis.yaml

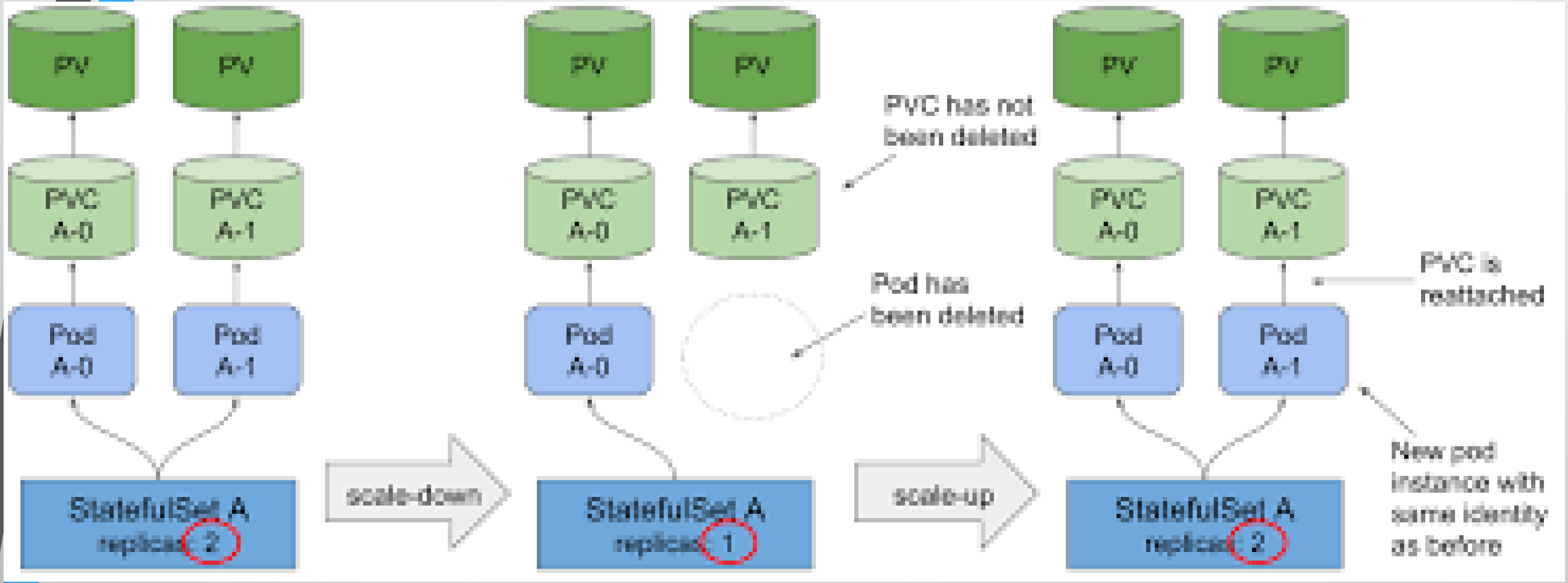
2. kubectl get pod redis –watch
3. 
4. kubectl exec -it redis -- /bin/bash

5. kubectl delete pod redis

## Configure a Pod to Use a PersistentVolume for Storage


Cluster
Persistent Volume Claims (PVCs)
PVC
Storage class: ibmc-file-silver
Size: 40 Gi
IOPS: 4
1
2
3
4
File storage plug-in (ibm.io/ibmc-file)
File storage
PV
40 Gi, 4 IOPS
Persistent Volumes (PVs)


Cluster
Persistent Volume Claims (PVCs)
3
PVC
Size: 40 Gi, IOPS: 4
4
1
PV
40 Gi, 4 IOPS
Persistent Volumes (PVs)
File storage plug-in (ibm.io/ibmc-file)
File storage
2


PV
PV
PVC A-0
PVC A-1
Pod A-0
Pod A-1
StatefulSet A replicas: 2
scale-down
PV
PV
PVC A-0
PVC A-1
Pod A-0
StatefulSet A replicas: 1
PVC has not been deleted
Pod has been deleted
scale-up
PV
PV
PVC A-0
PVC A-1
Pod A-0
Pod A-1
StatefulSet A replicas: 2
PVC is reattached
New pod instance with same identity as before

## Create a PersistentVolume

### Senario-11

1. mkdir /mnt/data
2. echo 'Hello from Kubernetes storage' > /mnt/data/index.html
3. kubectl create -f https://k8s.io/examples/pods/storage/pv-volume.yaml
4. kubectl get pv task-pv-volume

## Access Modes

1. ReadWriteOnce: The Volume can be mounted as read-write by a single node.
2. ReadOnlyMany: The Volume can be mounted read-only by many nodes.
3. ReadWriteMany: The Volume can be mounted as read-write by many nodes. PersistentVolumes that are backed by Compute Engine persistent disks don't support this access mode.
4.

# Create a PersistentVolumeClaim

## Senario-12

1. Step-1:
   1. mkdir /mnt/data
   2. echo 'Hello from Kubernetes storage' > /mnt/data/index.html
   3. kubectl create -f https://k8s.io/examples/pods/storage/pv-volume.yaml
   4. kubectl get pv task-pv-volume
2. Step-2:
   1. kubectl create -f https://k8s.io/examples/pods/storage/pv-claim.yaml
   2. kubectl get pv task-pv-volume
   3. kubectl get pvc task-pv-claim
3. Step-3
   1. kubectl create -f https://k8s.io/examples/pods/storage/pv-pod.yaml
   2. kubectl get pod task-pv-pod
   3. kubectl exec -it task-pv-pod -- /bin/bash
   4. root@task-pv-pod:/# apt-get update
   5. root@task-pv-pod:/# apt-get install curl
   6. root@task-pv-pod:/# curl localhost

## Configure Default Memory Requests and Limits for a Namespace

## Configure Default Memory Requests and Limits for a Namespace

### Senario-13

1. Create a namespace
    1. kubectl create namespace default-mem-example
2. Create a LimitRange and a Pod
    1. kubectl create -f https://k8s.io/examples/admin/resource/memory-defaults.yaml –namespace=default-mem-example

    2. kubectl create -f https://k8s.io/examples/admin/resource/memory-defaults-pod.yaml --namespace=default-mem-example

3. kubectl get pod default-mem-demo --output=yaml –namespace=default-mem-example
4. kubectl delete pod default-mem-demo --namespace=default-mem-example

## What if you specify a Container’s limit, but not its request?

### Senario-14

1. kubectl create -f https://k8s.io/examples/admin/resource/memory-defaults-pod-2.yaml --namespace=default-mem-example
2. kubectl get pod default-mem-demo-2 --output=yaml --namespace=default-mem-example
3.

## What if you specify a Container’s request, but not its limit?

### Senario-15

1. kubectl create -f https://k8s.io/examples/admin/resource/memory-defaults-pod-3.yaml --namespace=default-mem-example
2. 
3. kubectl get pod default-mem-demo-3 --output=yaml --namespace=default-mem-example
4. 
5.

## Configure a Pod Quota for a Namespace

## Configure a Pod Quota for a Namespace

### Senario-16

1. Create a namespace
   1. kubectl create namespace quota-pod-example
2. Create a ResourceQuota
   1. kubectl create -f https://k8s.io/examples/admin/resource/quota-pod.yaml --namespace=quota-pod-example

   2. kubectl get resourcequota pod-demo --namespace=quota-pod-example --output=yaml

3. kubectl create -f https://k8s.io/examples/admin/resource/quota-pod-deployment.yaml --namespace=quota-pod-example
4. kubectl get deployment pod-quota-demo --namespace=quota-pod-example –output=yaml
5. kubectl delete namespace quota-pod-example
6.

## Assign Pods to Nodes

## Configure a Pod Quota for a Namespace

### Senario-17

1. kubectl get nodes
2. kubectl label nodes <your-node-name> disktype=ssd
3. kubectl get nodes --show-labels
4. 
5. kubectl create -f https://k8s.io/examples/pods/pod-nginx.yaml
6. kubectl get pods --output=wide

## Running Automated Tasks with a CronJob

## Creating a Cron Job

### Senario-18

1. kubectl create -f https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/application/job/cronjob.yaml
2. 
3. kubectl get cronjob hello
4. kubectl get jobs --watch