ETCD

Monday, August 19, 2024 6:11 PM

kubelet --> takes instructions fro the master and deploys them om the worker node

kube -prxy---> maintains the communication between services within the cluster.

ETCD

. ETCD in Kubernetes

Setup - Manual

To exit full screen, press Esc

```
wget -q --https-only \
    "https://github.com/coreos/etcd/releases/download/v3.3.9/etcd-v3.3.9-linux-amd64.tar.gz"
```

etcd.service

```
ExecStart=/usr/local/bin/etcd \\
  --name ${ETCD_NAME} \\
  --cert-file=/etc/etcd/kubernetes.pem \\
  --key-file=/etc/etcd/kubernetes-key.pem \\
  --peer-cert-file=/etc/etcd/kubernetes.pem \\
  --peer-key-file=/etc/etcd/kubernetes-key.pem \\
  --trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-client-cert-auth \\
  --client-cert-auth \\
  --initial-advertise-peer-urls https://${INTERNAL_IP}:2380 \\
  --listen-peer-urls https://${INTERNAL_IP}:2380 \\
  --listen-client-urls https://${INTERNAL_IP}:2379,https://127.0.0.1:2379 \\
  --advertise-client-urls https://${INTERNAL_IP}:2379 \\
  --initial-cluster-token etcd-cluster-0 \\
  --initial-cluster controller-0=https://${CONTROLLER0_IP}:2380,controller-1=https://${CONTROLLER1_IP}:2380 \\
  --initial-cluster-state new \\
  --data-dir=/var/lib/etcd
```

ETCD - Commands (Optional)

(Optional) Additional information about ETCDCTL Utility

ETCDCTL is the CLI tool used to interact with ETCD.

ETCDCTL can interact with ETCD Server using 2 API versions - Version 2 and Version 3. By default its set to use Version 2. Each version has different sets of commands.

For example ETCDCTL version 2 supports the following commands:

```
etcdctl backup
etcdctl cluster-health
etcdctl mk
etcdctl mkdir
etcdctl set
```

Whereas the commands are different in version 3

```
etcdctl snapshot save
etcdctl endpoint health
etcdctl get
etcdctl put
```

Kube api server

Monday, August 19, 2024 6:39 PM

Kube-api Server

1. Authenticate User
2. Validate Request
3. Retrieve data
4. Update ETCD
5. Scheduler
6. Kubelet

View api-server options - kubeadm

```
cat /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
spec:
  containers:
  - command:
    - kube-apiserver
    - --authorization-mode=Node,RBAC
    - --advertise-address=172.17.0.32
    - --allow-privileged=true
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --disable-admission-plugins=PersistentVolumeLabel
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
```

View api-server options

```
cat /etc/systemd/system/kube-apiserver.service
```

```
[Service]
ExecStart=/usr/local/bin/kube-apiserver \\
  --advertise-address=${INTERNAL_IP} \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --audit-log-maxage=30 \\
  --audit-log-maxbackup=3 \\
  --audit-log-maxsize=100 \\
  --audit-log-path=/var/log/audit.log \\
  --authorization-mode=Node,RBAC \\
  --bind-address=0.0.0.0 \\
  --client-ca-file=/var/lib/kubernetes/ca.pem \\
  --enable-admission-
plugins=Initializers,NamespaceLifecycle,NodeRestriction,LimitRanger,ServiceAccount,Defa
ultStorageClass,ResourceQuota \\
  --enable-swagger-ui=true \\
  --etcd-cafile=/var/lib/kubernetes/ca.pem \\
  --etcd-certfile=/var/lib/kubernetes/kubernetes.pem \\
  --etcd-keyfile=/var/lib/kubernetes/kubernetes-key.pem \\
  --etcd-
```

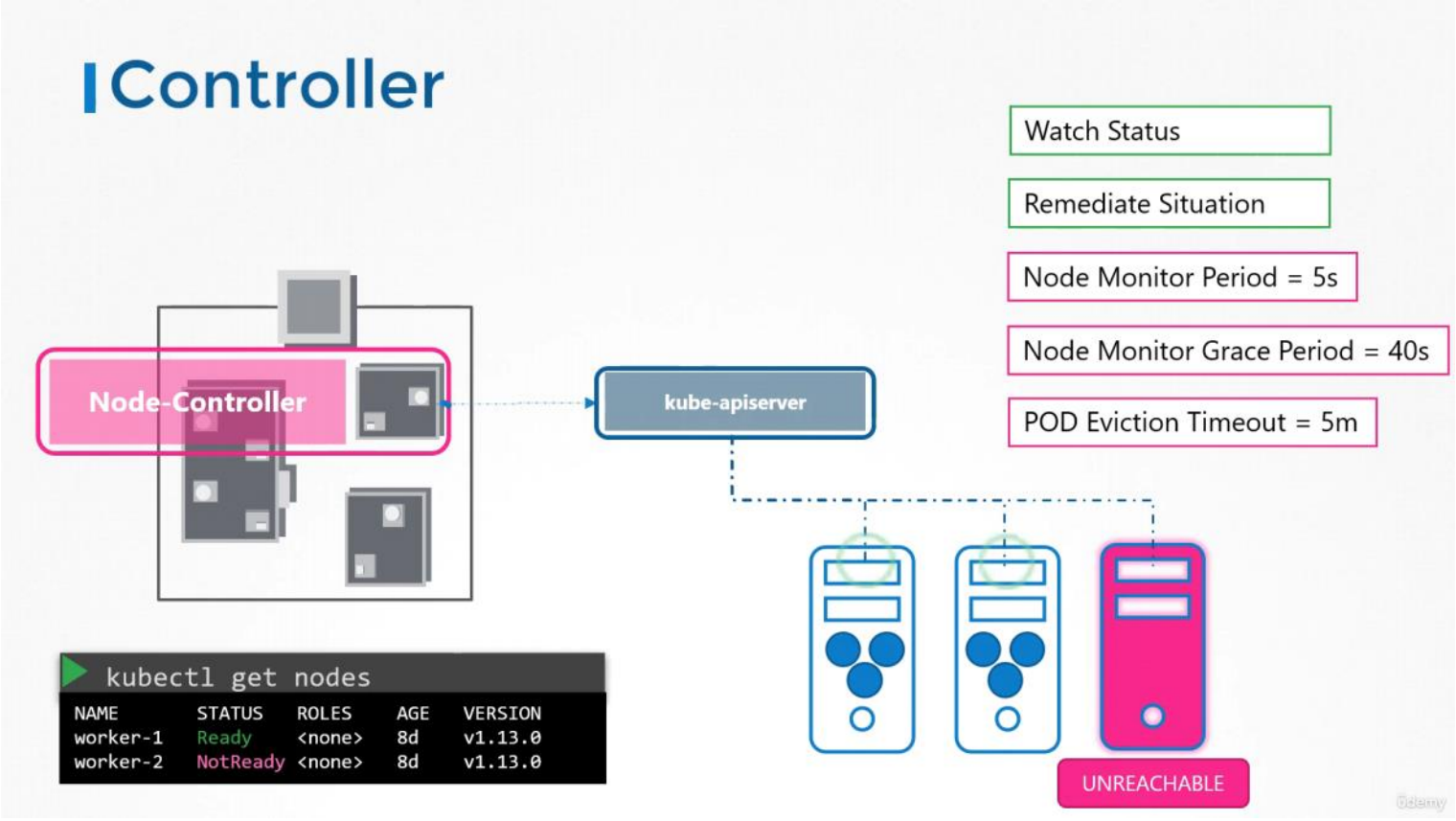
View api-server options

```
ps -aux | grep kube-apiserver
```

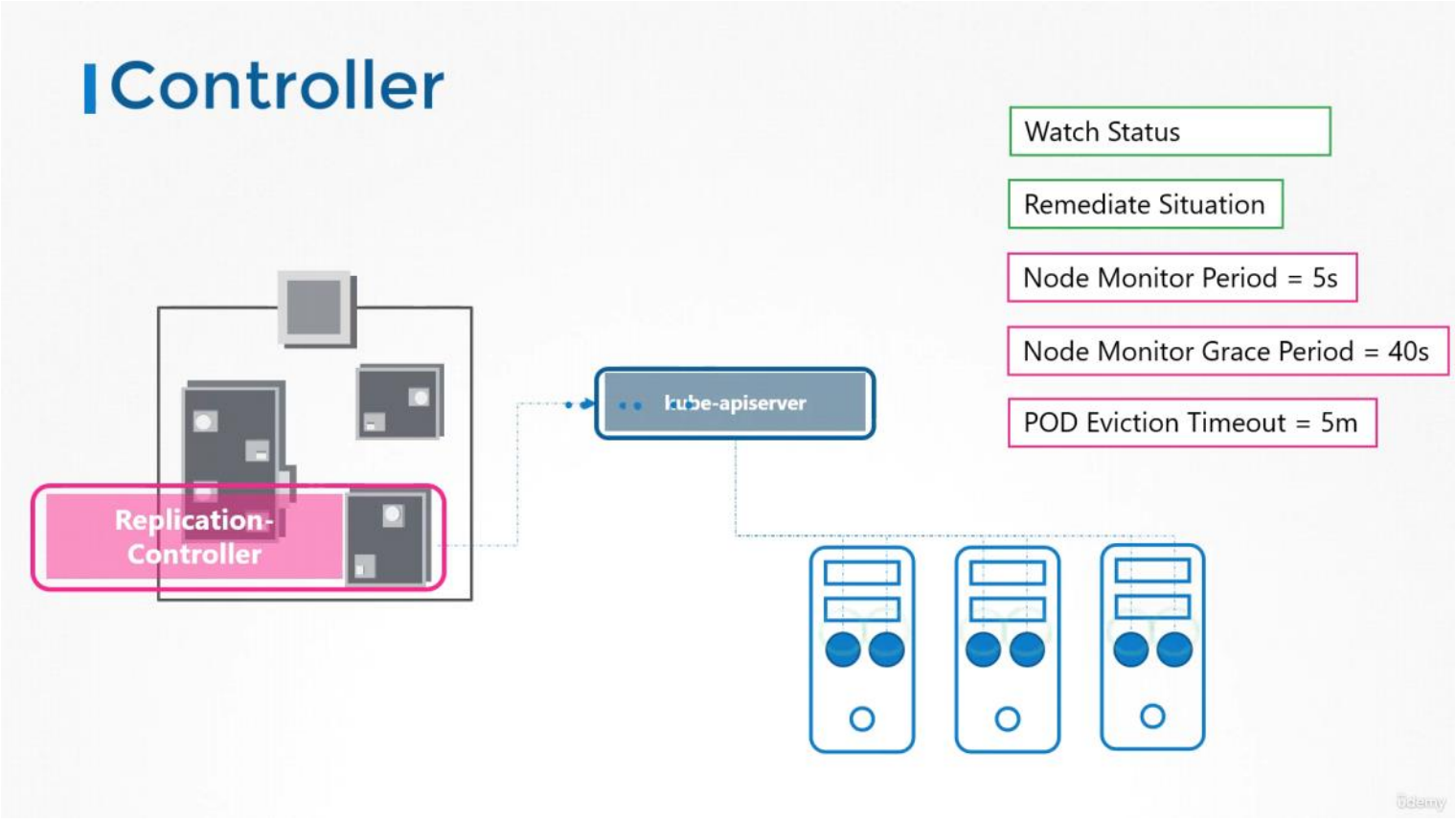
```
root      2348  3.3 15.4 399040 315604 ?       Ssl  15:46   1:22 kube-apiserver --authorization-mode=Node,RBAC --
advertise-address=172.17.0.32 --allow-privileged=true --client-ca-file=/etc/kubernetes/pki/ca.crt --disable-
admission-plugins=PersistentVolumeLabel --enable-admission-plugins=NodeRestriction--enable-bootstrap-token-
auth=true --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-
client.crt --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key --etcd-servers=https://127.0.0.1:2379 --
insecure-port=0 --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt --kubelet-client-
key=/etc/kubernetes/pki/apiserver-kubelet-client.key --kubelet-preferred-address-
types=InternalIP,ExternalIP,Hostname --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt --proxy-
client-key-file=/etc/kubernetes/pki/front-proxy-client.key--requestheader-allowed-names=front-proxy-client --
requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt --requestheader-extra-headers-prefix=X-Remote-
Extra- --requestheader-group-headers=X-Remote-Group --requestheader-username-headers=X-Remote-User --secure-
port=6443 --service-account-key-file=/etc/kubernetes/pki/sa.pub --service-cluster-ip-range=10.96.0.0/12 --tls-
cert-file=/etc/kubernetes/pki/apiserver.crt --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
```

Kube controller manager

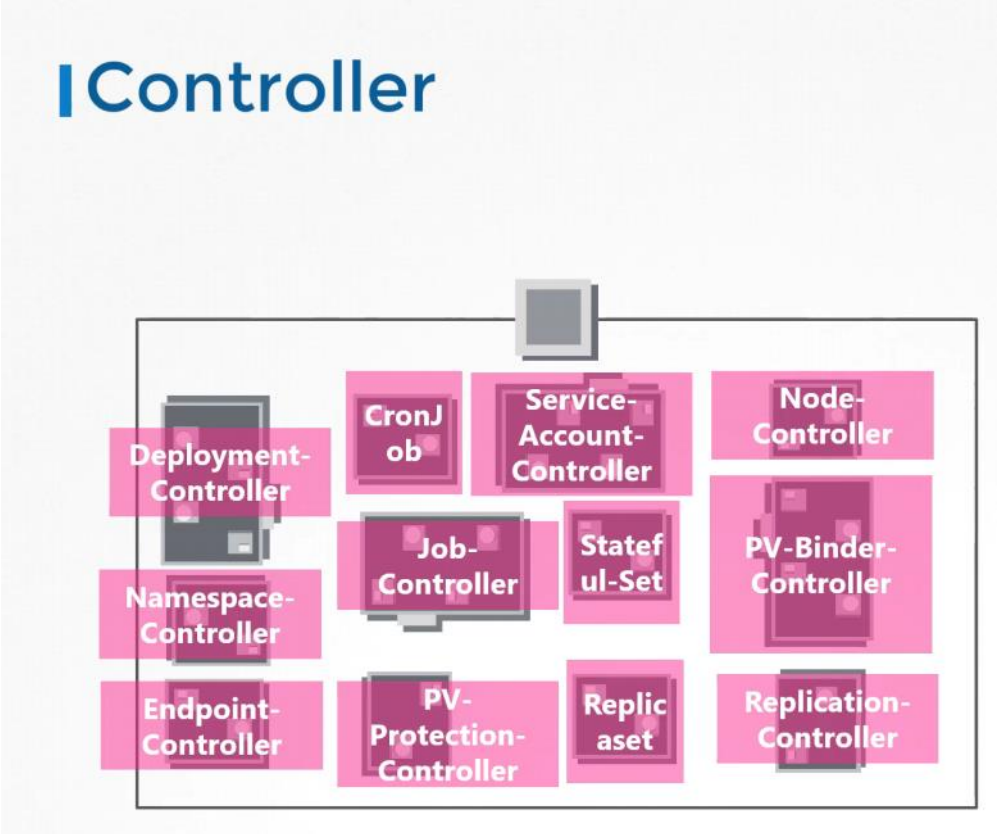
Monday, August 19, 2024 6:48 PM



monitors node every 5s
if not received signal it marks as unreachable, but waits for the signal for 40s
later it evicts the pods if it is still unreachable for 5m ie it gives 5m to come back alive



monitors health of replica sets--> replication controller



all these are inside kube controller manager

Installing kube-controller-manager

```
wget https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kube-controller-manager
```

kube-controller-manager.service

```
ExecStart=/usr/local/bin/kube-controller-manager \\
  --address=0.0.0.0 \\
  --cluster-cidr=10.200.0.0/16 \\
  --cluster-name=kubernetes \\
  --cluster-signing-cert-file=/var/lib/kubernetes/ca.pem \\
  --cluster-signing-key-file=/var/lib/kubernetes/ca-key.pem \\
  --kubeconfig=/var/lib/kubernetes/kube-controller-manager.kubeconfig \\
  --leader-elect=true \\
  --root-ca-file=/var/lib/kubernetes/ca.pem \\
  --service-account-private-key-file=/var/lib/kubernetes/service-account-key.pem \\
  --service-cluster-ip-range=10.32.0.0/24 \\
  --use-service-account-credentials=true \\
  --v=2
  --node-monitor-period=5s
  --node-monitor-grace-period=40s
  --pod-eviction-timeout=5m0s
  --controllers stringSlice     Default: [*]
  A list of controllers to enable. '*' enables all on-by-default controllers, 'foo' enables the controller
  named 'foo', '-foo' disables the controller named 'foo'.
  All controllers: attachdetach, bootstrapsigner, clusterrole-aggregation, cronjob, csrapproving,
  csrcleaner, csrsigning, daemonset, deployment, disruption, endpoint, garbagecollector,
  horizontalpodautoscaling, job, namespace, nodeipam, nodelifecycle, persistentvolume-binder,
  persistentvolume-expander, podgc, pv-protection, pvc-protection, replicaset, replicationcontroller,
```

wget it
run as a service

View kube-controller-manager - kubeadm

```
kubectl get pods -n kube-system
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-78fcdf6894-hwrq9	1/1	Running	0	16m
kube-system	coredns-78fcdf6894-rzhjr	1/1	Running	0	16m
kube-system	etcd-master	1/1	Running	0	15m
kube-system	kube-apiserver-master	1/1	Running	0	15m
kube-system	kube-controller-manager-master	1/1	Running	0	15m
kube-system	kube-proxy-lzt6f	1/1	Running	0	16m
kube-system	kube-proxy-zm5qd	1/1	Running	0	16m
kube-system	kube-scheduler-master	1/1	Running	0	15m
kube-system	weave-net-29z42	2/2	Running	1	16m
kube-system	weave-net-snmdl	2/2	Running	1	16m

View kube-controller-manager options - kubeadm

```
cat /etc/kubernetes/manifests/kube-controller-manager.yaml
```

```
spec:
  containers:
  - command:
    - kube-controller-manager
    - --address=127.0.0.1
    - --cluster-signing-cert-file=/etc/kubernetes/pki/ca.crt
    - --cluster-signing-key-file=/etc/kubernetes/pki/ca.key
    - --controllers=*,bootstrapsigner,tokencleaner
    - --kubeconfig=/etc/kubernetes/controller-manager.conf
    - --leader-elect=true
    - --root-ca-file=/etc/kubernetes/pki/ca.crt
    - --service-account-private-key-file=/etc/kubernetes/pki/sa.key
    - --use-service-account-credentials=true
```

View controller-manager options

```
cat /etc/systemd/system/kube-controller-manager.service
```

```
[Service]
ExecStart=/usr/local/bin/kube-controller-manager \\
  --address=0.0.0.0 \\
  --cluster-cidr=10.200.0.0/16 \\
  --cluster-name=kubernetes \\
  --cluster-signing-cert-file=/var/lib/kubernetes/ca.pem \\
  --cluster-signing-key-file=/var/lib/kubernetes/ca-key.pem \\
  --kubeconfig=/var/lib/kubernetes/kube-controller-manager.kubeconfig \\
  --leader-elect=true \\
  --root-ca-file=/var/lib/kubernetes/ca.pem \\
  --service-account-private-key-file=/var/lib/kubernetes/service-account-key.pem \\
  --service-cluster-ip-range=10.32.0.0/24 \\
  --use-service-account-credentials=true \\
  --v=2
Restart=on-failure
RestartSec=5
```

Outlook

Datadog Alerting

View controller-manager options

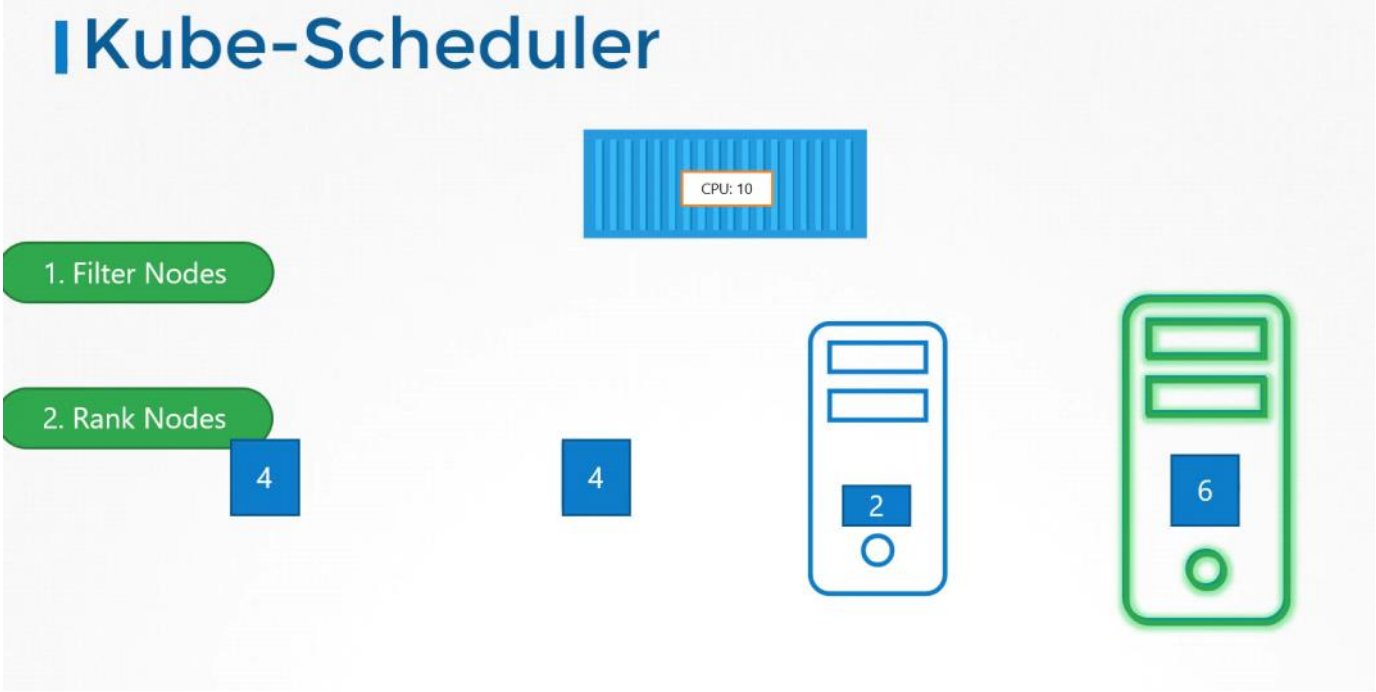
```
ps -aux | grep kube-controller-manager
```

```
root      1994  2.7  5.1 154360 105024 ?       Ssl  06:45   1:25 kube-controller-manager --address=127.0.0.1 --cluster-signing-cert-file=/etc/kubernetes/pki/ca.crt --cluster-signing-key-file=/etc/kubernetes/pki/ca.key --controllers=*,bootstrapsigner,tokencleaner --kubeconfig=/etc/kubernetes/controller-manager.conf --leader-elect=true --root-ca-file=/etc/kubernetes/pki/ca.crt --service-account-private-key-file=/etc/kubernetes/pki/sa.key --use-service-account-credentials=true
```

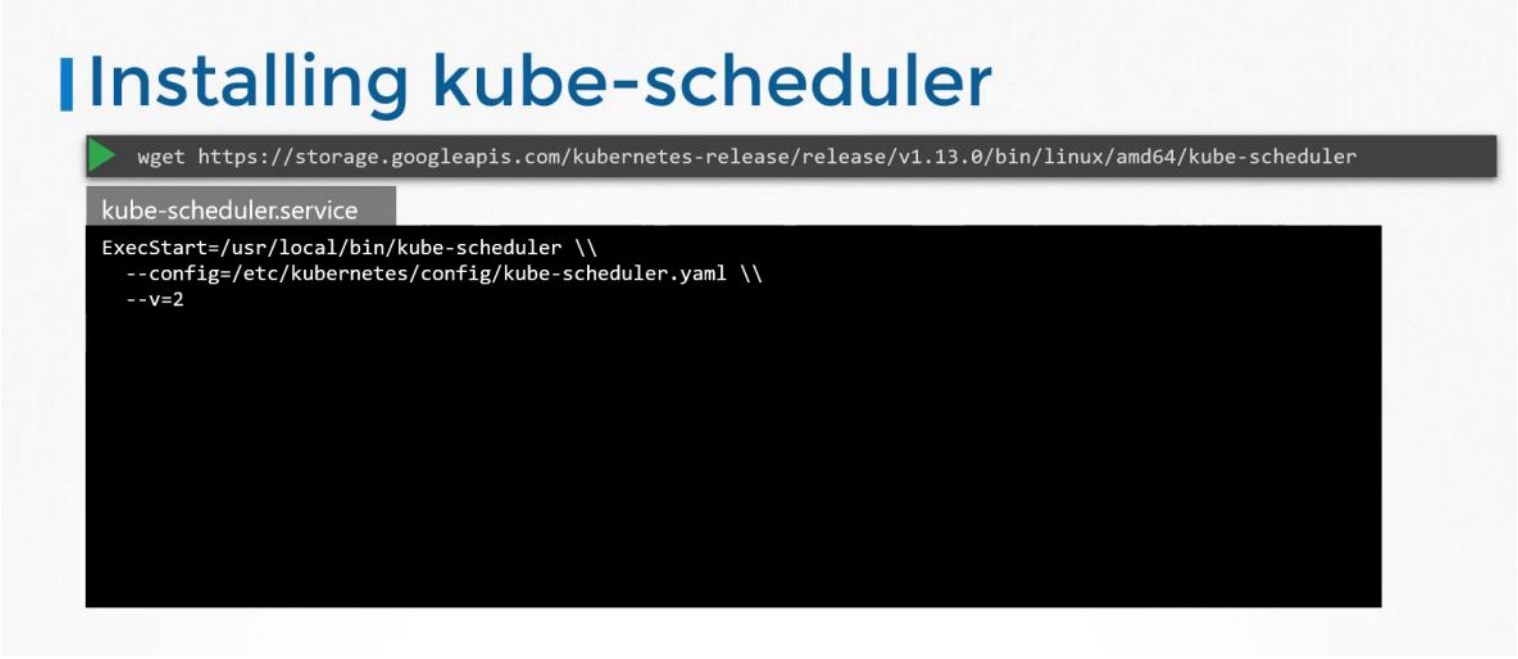
Kube Scheduler

Monday, August 19, 2024 6:59 PM

scheduler decides which pod goes on which node. but does not actually places them.
Kubelet creates the pod
scheduler makes sure which container goes where



filters based on cpu
jispe jyada space bacha uska rank jyada



View kube-scheduler options - kubeadm

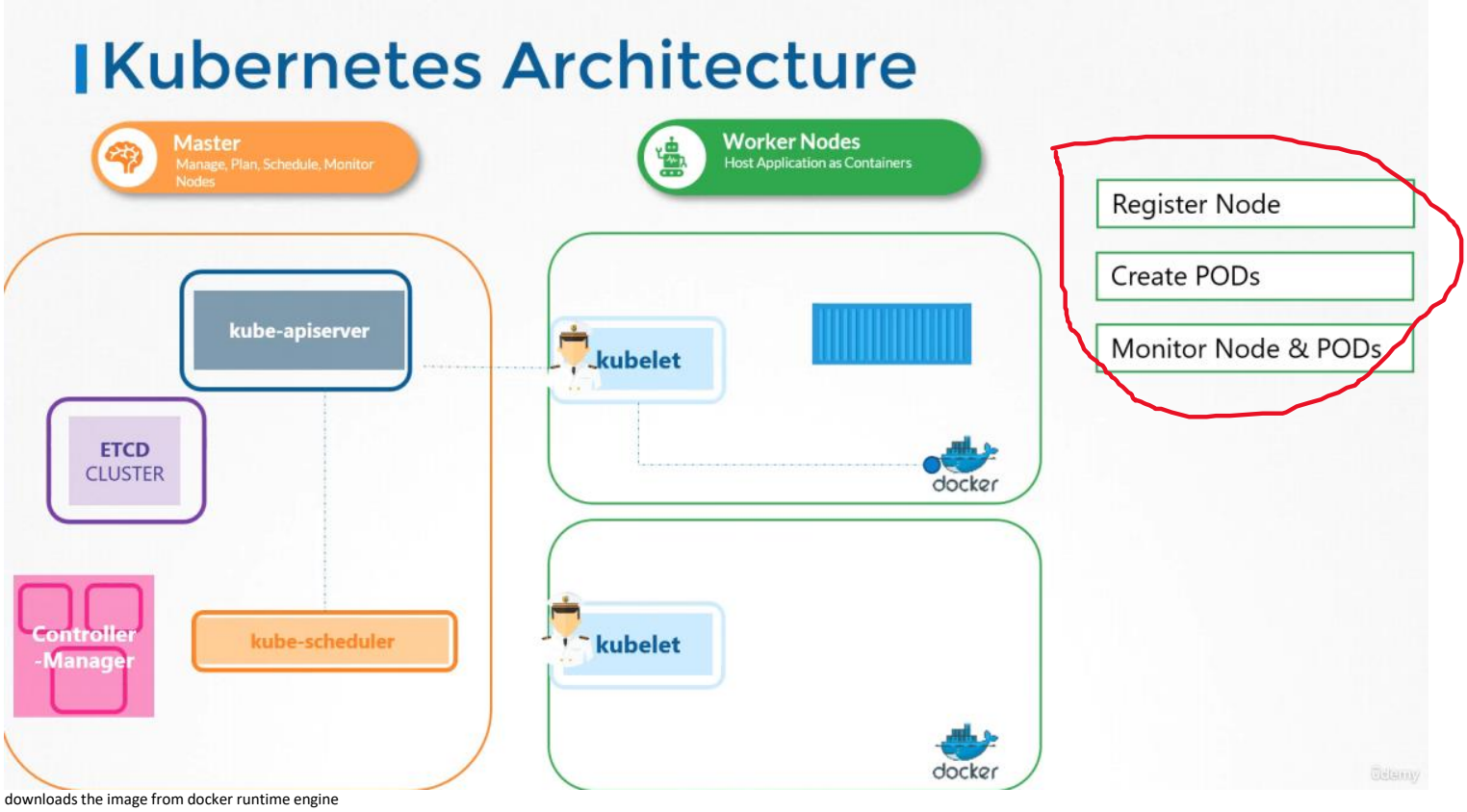
```
cat /etc/kubernetes/manifests/kube-scheduler.yaml
```

View kube-scheduler options

```
ps -aux | grep kube-scheduler
root      2477  0.8  1.6  48524 34044 ?        Ssl  17:31   0:08 kube-scheduler --
address=127.0.0.1 --kubeconfig=/etc/kubernetes/scheduler.conf --leader-elect=true
```

kubelet

Monday, August 19, 2024 7:07 PM



downloads the image from docker runtime engine
creates a pod
monitors its health

we have to manually install kubelet

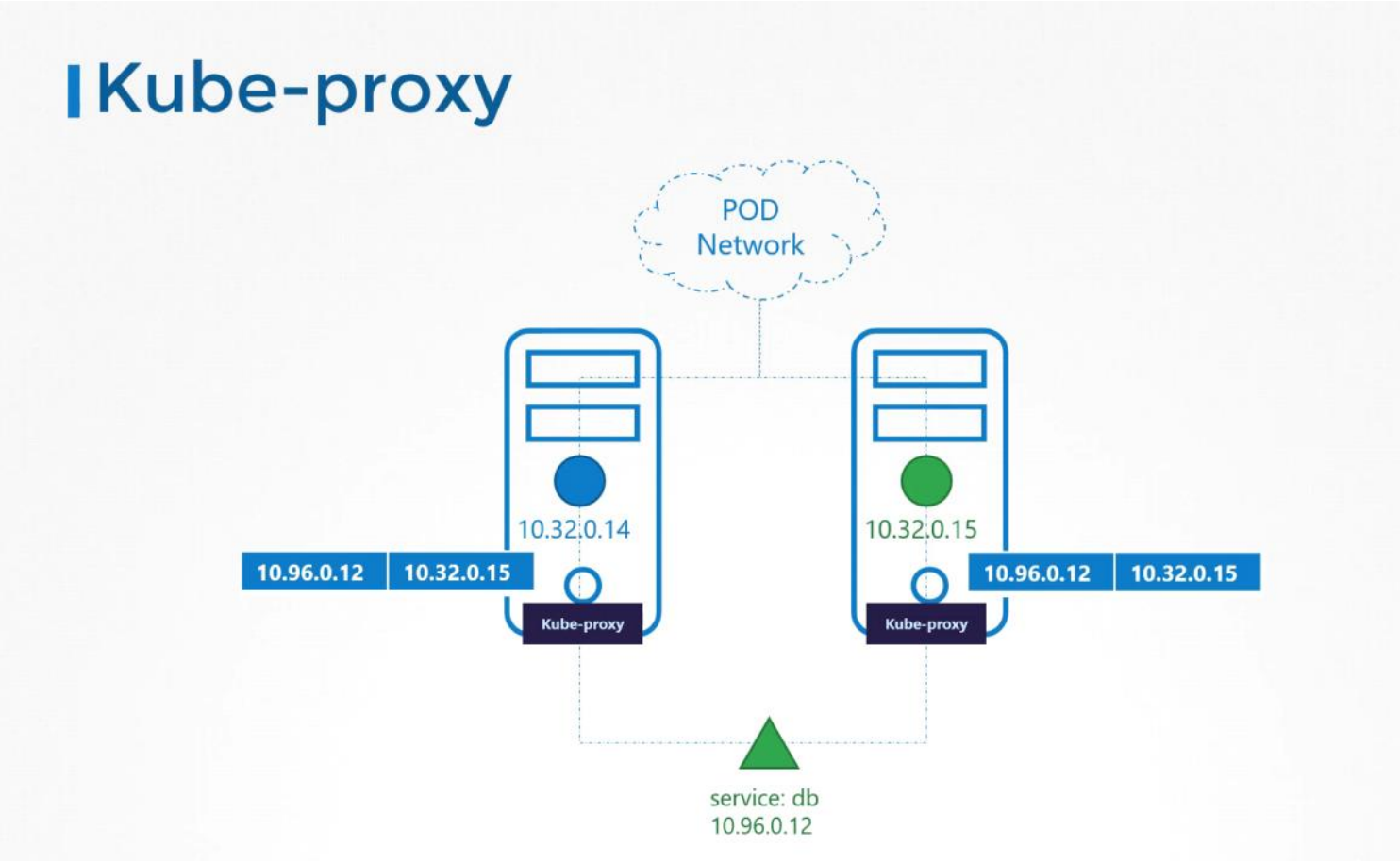
View kubelet options

```
ps -aux | grep kubelet
```

```
root      2095  1.8  2.4 960676 98788 ?        Ssl  02:32   0:36 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --config=/var/lib/kubelet/config.yaml --cgroup-driver=cgroupfs --cni-bin-dir=/opt/cni/bin --cni-conf-dir=/etc/cni/net.d --network-plugin=cni
```

Kube Proxy

Monday, August 19, 2024 7:07 PM



It creates IP table rules on each node on cluster to forward traffic of service 10.96.0.12 to the actual POD 10.32.0.15

Installing kube-proxy

```
wget https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kube-proxy
```

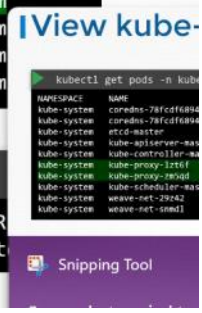
kube-proxy.service

```
ExecStart=/usr/local/bin/kube-proxy \\
  --config=/var/lib/kube-proxy/kube-proxy-config.yaml
Restart=on-failure
RestartSec=5
```

View kube-proxy - kubeadm

```
kubectl get pods -n kube-system
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
kube-system   coredns-78fcdf6894-hwrq9          1/1     Running   0          16m
kube-system   coredns-78fcdf6894-rzhjr          1/1     Running   0          16m
kube-system   etcd-master                       1/1     Running   0          15m
kube-system   kube-apiserver-master             1/1     Running   0          15m
kube-system   kube-controller-manager-master    1/1     Running   0          15m
kube-system   kube-proxy-lzt6f                  1/1     Running   0          16m
kube-system   kube-proxy-zm5qd                  1/1     Running   0          16m
kube-system   kube-scheduler-master             1/1     Running   0          15m
kube-system   weave-net-29z42                   2/2     Running   1          16m
kube-system   weave-net-snmdl                   2/2     Running   1          16m
```

```
kubectl get daemonset -n kube-system
NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
kube-proxy   2         2         2       2            2           beta.kubernet
```



Pods

Monday, August 19, 2024 9:51 PM

YAML in Kubernetes

pod-definition.yml

```
apiVersion: v1
kind:
metadata:



spec:
```

Kind	Version
POD	v1
Service	v1
ReplicaSet	apps/v1
Deployment	apps/v1

YAML in Kubernetes

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
      app: myapp
      type: front-end
spec:
  containers:
    - name: nginx-container
      image: nginx
```

Kind	Version
POD	v1
Service	v1
ReplicaSet	apps/v1
Deployment	apps/v1

```
kubectl create -f pod-definition.yml
```

what does the ready column in kubectl get pods indicate ?
It indicates running containers/total containers

kubectl delete pod webapp

kubectl run redis --image redis --dry-run=client -o yaml > r
ed.yml

kubectl create -f red.yml ---> when creating a resource from a file

- How many PODs exist on the system?
  In the current(default) namespace.

```
>>>controlplane ~ ➜  kubectl get ns -o wide
                    NAME            STATUS   AGE
                    kube-system      Active   16m
                    kube-public      Active   16m
                    kube-node-lease   Active   16m
                    default          Active   16m
```

```
>>> kubectl describe ns default
```

Replica set ensures that the desired number of pods are always running.

kubectl get rs -o yaml >rohan

kubectl scale --help

kubectl scale --replicas=5 -f rohan.yaml

kubectl edit replicaset

Deployments

Monday, August 19, 2024 11:38 PM

Definition

deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
  labels:
     app: myapp
     type: front-end
spec:
  template:
    metadata:
     name: myapp-pod
     labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx
  replicas: 3
  selector:
    matchLabels:
       type: front-end
```

Definition

deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
  labels:
     app: myapp
     type: front-end
spec:
  template:
    metadata:
     name: myapp-pod
     labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx
  replicas: 3
  selector:
    matchLabels:
       type: front-end
```

```
> kubectl create -f deployment-definition.yml
deployment "myapp-deployment" created
```

```
> kubectl get deployments
NAME              DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE

myapp-deployment   3         3         3            3           21s
```

```
> kubectl get replicaset
NAME                         DESIRED   CURRENT   READY     AGE

myapp-deployment-6795844b58   3         3         3         2m
```

```
> kubectl get pods
NAME                                READY     STATUS    RESTARTS   AGE
myapp-deployment-6795844b58-5rbj1   1/1       Running   0          2m
myapp-deployment-6795844b58-h4w55   1/1       Running   0          2m
myapp-deployment-6795844b58-lfjhv   1/1       Running   0          2m
```

here the deployment creates a replica set and which in turn creates pods

commands

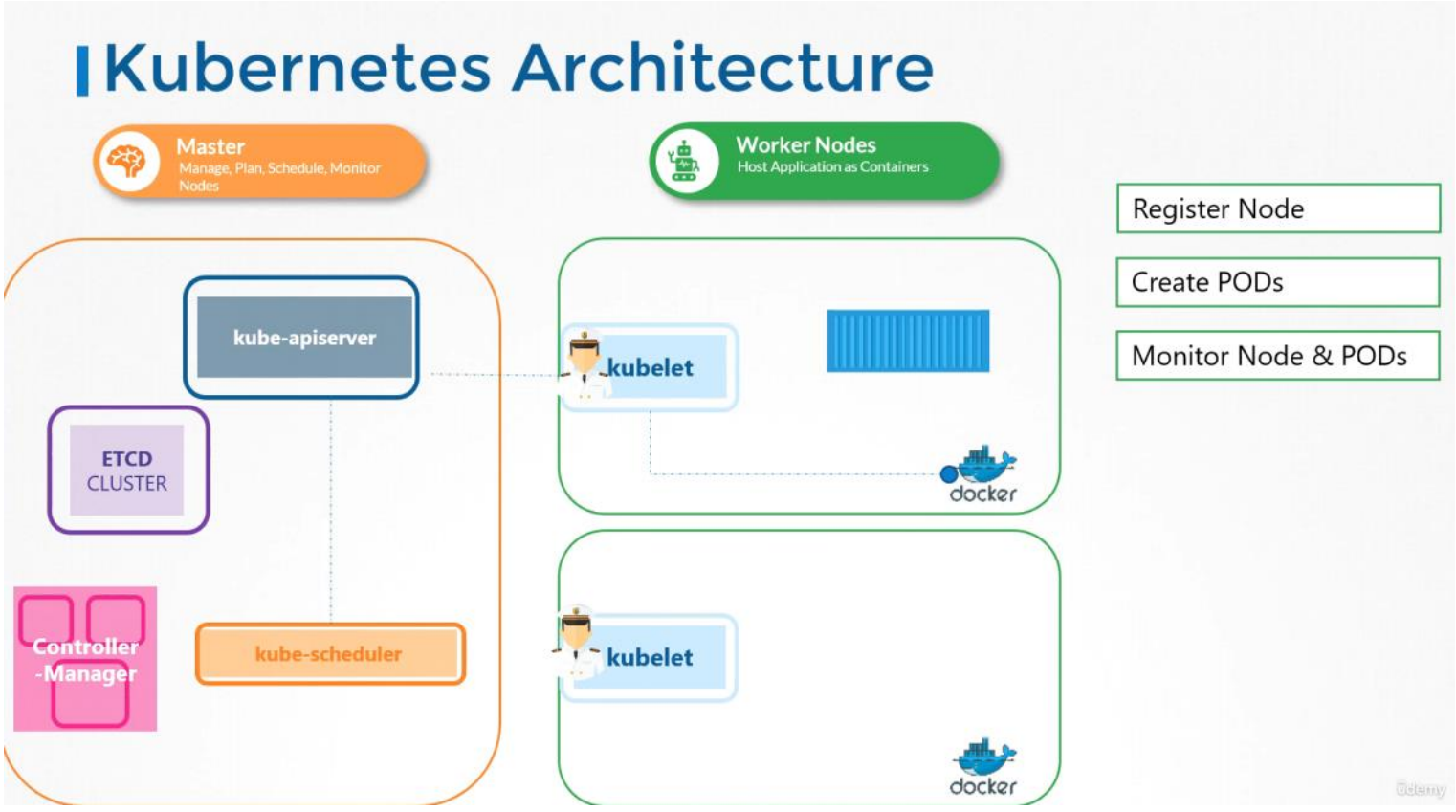
```
> kubectl get all
NAME                     DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/myapp-deployment   3         3         3            3           9h

NAME                               DESIRED   CURRENT   READY     AGE
rs/myapp-deployment-6795844b58     3         3         3         9h

NAME                                   READY     STATUS    RESTARTS   AGE
po/myapp-deployment-6795844b58-5rbjl   1/1       Running   0          9h
po/myapp-deployment-6795844b58-h4w55   1/1       Running   0          9h
po/myapp-deployment-6795844b58-lfjhv   1/1       Running   0          9h
```

Certification Tip!
Here's a tip!
As you might have seen already, it is a bit difficult to create and edit YAML files. Especially in the CLI. During the exam, you might find it difficult to copy and paste YAML files from browser to terminal. Using the kubectl run command can help in generating a YAML template. And sometimes, you can even get away with just the kubectl run command without having to create a YAML file at all. For example, if you were asked to create a pod or deployment with specific name and image you can simply run the kubectl run command.
Use the below set of commands and try the previous practice tests again, but this time try to use the below commands instead of YAML files. Try to use these as much as you can going forward in all exercises
Reference (Bookmark this page for exam. It will be very handy):
https://kubernetes.io/docs/reference/kubectl/conventions/
Create an NGINX Pod
kubectl run nginx --image=nginx
Generate POD Manifest YAML file (-o yaml). Don't create it(--dry-run)
kubectl run nginx --image=nginx --dry-run=client -o yaml
Create a deployment
kubectl create deployment --image=nginx nginx
Generate Deployment YAML file (-o yaml). Don't create it(--dry-run)
kubectl create deployment --image=nginx nginx --dry-run=client -o yaml
Generate Deployment YAML file (-o yaml). Don't create it(–dry-run) and save it to a file.
kubectl create deployment --image=nginx nginx --dry-run=client -o yaml > nginx-deployment.yaml
Make necessary changes to the file (for example, adding more replicas) and then create the deployment.
kubectl create -f nginx-deployment.yaml

OR
In k8s version 1.19+, we can specify the --replicas option to create a deployment with 4 replicas.
kubectl create deployment --image=nginx nginx --replicas=4 --dry-run=client -o yaml > nginx-deployment.yaml

Architecture

Monday, August 26, 2024 3:03 PM




```
        #ls -lrt

        echo $HelmInstallation


        if [ "${HelmInstallation}" = "Dryrun" ]
        then
           helm install --dry-run --debug tibco .
        elif [ "${HelmInstallation}" = "Upgrade" ]
        then
           helm upgrade -f values.yaml tibco . -n tibc
        fi
      '''
    }
}
```

Kubernetes

Tuesday, June 4, 2024 6:45 PM

Task Hint Solution 50:11

1 2 3 4 5 6 7 8 9 10 11 12 13

Create a new pod with the name `redis` and with the image `redis123`.
Use a pod-definition YAML file. And yes the image name is wrong!

Check

- Name: redis
- Image Name: redis123

Terminal 1

```
controlplane ~ ➜  kubectl run redis --image=redis123 --dry-run -o yaml

W0415 18:19:23.939739    9207 helpers.go:598] --dry-run is deprecated and can be replaced with --dry-run
=client.
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: redis
  name: redis
spec:
  containers:
  - image: redis123
    name: redis
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}

controlplane ~ ➜

controlplane ~ ➜  kubectl run redis --image=redis123 --dry-run=client -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: redis
  name: redis
spec:
  containers:
  - image: redis123
    name: redis
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}

controlplane ~ ➜
```

```
status: {}

controlplane ~ ➜  kubectl run redis --image=redis123 --dry-run=client -o yaml > redis.yaml
```

Check

- Name: redis
- Image Name: redis123

```
  restartPolicy: Always
status: {}

controlplane ~ ➜  kubectl create -f redis.yaml
pod/redis created

controlplane ~ ➜  kubectl get pods
NAME           READY   STATUS         RESTARTS   AGE
newpods-pnnx8  1/1     Running        0          8m16s
newpods-llstt  1/1     Running        0          8m16s
newpods-k87fx  1/1     Running        0          8m16s
nginx          1/1     Running        0          8m22s
redis          0/1     ErrImagePull   0          10s

controlplane ~ ➜
```

Task Hint Solution 48:29

1 2 3 4 5 6 7 8 9 10 11 12 13

Now change the image on this pod to `redis`.

Once done, the pod should be in a `running` state.

Check

- Name: redis
- Image Name: redis

Terminal 1 +

```
controlplane ~ ➜  cat redis.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: redis
  name: redis
spec:
  containers:
  - image: redis123
    name: redis
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}

controlplane ~ ➜  vi redis.yaml

controlplane ~ ➜  kubectl apply -f redis.yaml
Warning: resource pods/redis is missing the kubectl.kubernetes.io/last-applied-configuration annotation
which is required by kubectl apply. kubectl apply should only be used on resources created declaratively
 by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automat
ically.
pod/redis configured

controlplane ~ ➜

controlplane ~ ➜

controlplane ~ ➜  kubectl get pods
```

```
controlplane ~ ➜  kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
newpods-pnnx8  1/1     Running   0          9m38s
newpods-llstt  1/1     Running   0          9m38s
newpods-k87fx  1/1     Running   0          9m38s
nginx          1/1     Running   0          9m44s
redis          1/1     Running   0          92s

controlplane ~ ➜
```

Replication controller

Tuesday, June 4, 2024 6:53 PM

it ensures specified number of pods are running all the time

rc-definition.yml

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: myapp-rc
  labels:
      app: myapp
      type: front-end
spec:
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx
  replicas: 3
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
```

```
> kubectl create -f rc-definition.yml
replicationcontroller "myapp-rc" created
```

```
> kubectl get replicationcontroller
NAME       DESIRED   CURRENT   READY     AGE
myapp-rc   3         3         3         19s
```

```
> kubectl get pods
NAME            READY     STATUS    RESTARTS   AGE
myapp-rc-4lvk9  1/1       Running   0          20s
myapp-rc-mc2mf  1/1       Running   0          20s
myapp-rc-px9pz  1/1       Running   0          20s
```

replicaset-definition.yml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-replicaset
  labels:
      app: myapp
      type: front-end
spec:
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx
  replicas: 3
  selector:
    matchLabels:
      type: front-end
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
```

The match labels selectors simply matches the labels

```
> kubectl create -f replicaset-definition.yml
replicaset "myapp-replicaset" created
> kubectl get replicaset
NAME              DESIRED   CURRENT   READY     AGE
myapp-replicaset   3         3         3         19s
> kubectl get pods
NAME                   READY     STATUS    RESTARTS   AGE
myapp-replicaset-9dd19   1/1       Running   0          45s
myapp-replicaset-9jtpx   1/1       Running   0          45s
myapp-replicaset-hq84m   1/1       Running   0          45s
```

To scale replicas
you can edit the definition file
vi

Scale

```
> kubectl replace -f replicaset-definition.yml

> kubectl scale --replicas=6 -f replicaset-definition.yml

> kubectl scale --replicas=6 replicaset myapp-replicaset
```

replicaset-definition.yml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-replicaset
  labels:
      app: myapp
      type: front-end
spec:
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
```

```
> kubectl scale --replicas=6 replicaset myapp-replicaset
```

TYPE NAME

```
      app: myapp
      type: front-end
  spec:
    containers:
    - name: nginx-container
      image: nginx
replicas: 6
selector:
  matchLabels:
    type: front-end
```

the lower 2 commands wont make changes in the file.

commands

```
> kubectl create -f replicaset-definition.yml
```

```
> kubectl get replicaset
```

```
> kubectl delete replicaset myapp-replicaset
```

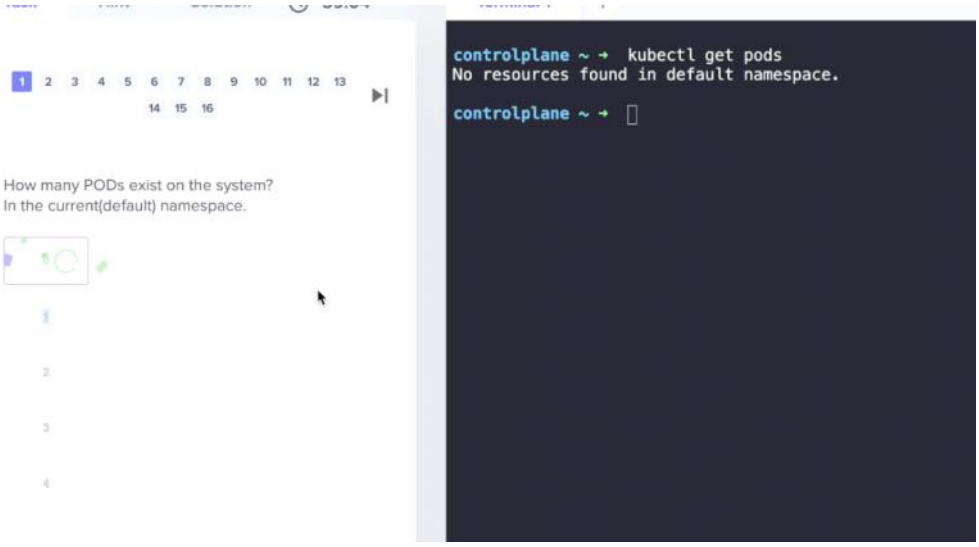
*Also deletes all underlying PODs

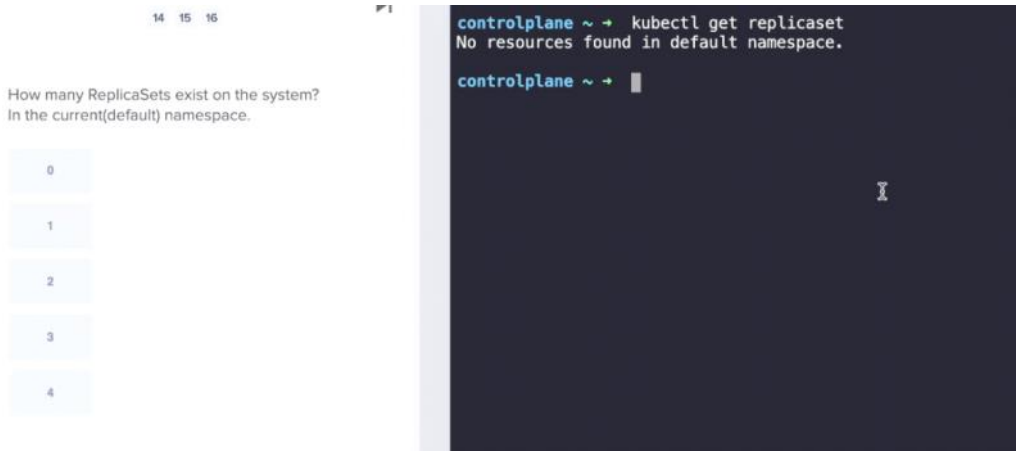
```
> kubectl replace -f replicaset-definition.yml
```

```
> kubectl scale -replicas=6 -f replicaset-definition.yml
```

Practice Test - ReplicaSets - Solution

Tuesday, June 4, 2024 7:35 PM





What is the image used to create the pods in the `new-replica-set` ?

- NGINX
- NEWPOD
- BUSYBOX-CONTAINER
- BUSYBOX777
- BUSYBOX-POD

```
NAME              DESIRED   CURRENT   READY   AGE
new-replica-set   4         4         0       9s

controlplane ~ ➜  kubectl describe replicaset new-replica-set
Name:         new-replica-set
Namespace:    default
Selector:     name=busybox-pod
Labels:       <none>
Annotations:  <none>
Replicas:     4 current / 4 desired
Pods Status:  0 Running / 4 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  name=busybox-pod
  Containers:
   busybox-container:
    Image:      busybox777
    Port:       <none>
    Host Port:  <none>
    Command:
      sh
      -c
      echo Hello Kubernetes! && sleep 3600
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
  Type    Reason            Age   From                   Message
  ----    ------            ----  ----                   -------
  Normal  SuccessfulCreate  60s   replicaset-controller  Created pod: new-replica-set-7r2qw
  Normal  SuccessfulCreate  60s   replicaset-controller  Created pod: new-replica-set-wkzjh
  Normal  SuccessfulCreate  60s   replicaset-controller  Created pod: new-replica-set-tn2mp
  Normal  SuccessfulCreate  60s   replicaset-controller  Created pod: new-replica-set-vpkh8

controlplane ~ ➜
```

How many PODs are READY in the `new-replica-set` ?

- 0
- 1
- 2
- 4
- 5


```
new-replica-set   4         4         0       9s

controlplane ~ ➜  kubectl describe replicaset new-replica-set
Name:         new-replica-set
Namespace:    default
Selector:     name=busybox-pod
Labels:       <none>
Annotations:  <none>
Replicas:     4 current / 4 desired
Pods Status:  0 Running / 4 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  name=busybox-pod
  Containers:
   busybox-container:
    Image:      busybox777
    Port:       <none>
    Host Port:  <none>
    Command:
      sh
      -c
      echo Hello Kubernetes! && sleep 3600
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
  Type    Reason            Age   From                   Message
  ----    ------            ----  ----                   -------
  Normal  SuccessfulCreate  60s   replicaset-controller  Created pod: new-replica-set-7r2qw
  Normal  SuccessfulCreate  60s   replicaset-controller  Created pod: new-replica-set-wkzjh
  Normal  SuccessfulCreate  60s   replicaset-controller  Created pod: new-replica-set-tn2mp
  Normal  SuccessfulCreate  60s   replicaset-controller  Created pod: new-replica-set-vpkh8
```

Why do you think the PODs are not ready?

- Application has errors
- Kubernetes is faulty
- Replicaset was not created correctly
- The image BUSYBOX777 doesn't exist

```
new-replica-set   4         4         0       9s

controlplane ~ ➜  kubectl describe replicaset new-replica-set
Name:         new-replica-set
Namespace:    default
Selector:     name=busybox-pod
Labels:       <none>
Annotations:  <none>
Replicas:     4 current / 4 desired
Pods Status:  0 Running / 4 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  name=busybox-pod
  Containers:
   busybox-container:
    Image:      busybox777
    Port:       <none>
    Host Port:  <none>
    Command:
      sh
      -c
      echo Hello Kubernetes! && sleep 3600
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
  Type    Reason            Age   From                   Message
  ----    ------            ----  ----                   -------
  Normal  SuccessfulCreate  60s   replicaset-controller  Created pod: new-replica-set-7r2qw
  Normal  SuccessfulCreate  60s   replicaset-controller  Created pod: new-replica-set-wkzjh
  Normal  SuccessfulCreate  60s   replicaset-controller  Created pod: new-replica-set-tn2mp
  Normal  SuccessfulCreate  60s   replicaset-controller  Created pod: new-replica-set-vpkh8

controlplane ~ ➜  kubectl describe pod new-replica-set-7r2qw
```

Why do you think the PODs are not ready?

- Application has errors
- Kubernetes is faulty
- Replicaset was not created correctly
- The image BUSYBOX777 doesn't exist

```
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-qc2d2 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  kube-api-access-qc2d2:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Normal   Scheduled  2m23s                default-scheduler  Successfully assigned default/new-replica-
set-7r2qw to controlplane
  Normal   Pulling    46s (x4 over 2m23s)  kubelet            Pulling image "busybox777"
  Warning  Failed     46s (x4 over 2m22s)  kubelet            Failed to pull image "busybox777": rpc err
or: code = Unknown desc = failed to pull and unpack image "docker.io/library/busybox777:latest": failed
to resolve reference "docker.io/library/busybox777:latest": pull access denied, repository does not exis
t or may require authorization: server message: insufficient_scope: authorization failed
  Warning  Failed     46s (x4 over 2m22s)  kubelet            Error: ErrImagePull
  Warning  Failed     34s (x6 over 2m21s)  kubelet            Error: ImagePullBackOff
  Normal   BackOff    23s (x7 over 2m21s)  kubelet            Back-off pulling image "busybox777"

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Delete any one of the 4 PODs.

Done

```
controlplane ~ ➜  kubectl get pods
NAME                   READY   STATUS             RESTARTS   AGE
new-replica-set-wkzjh  0/1     ImagePullBackOff   0          2m59s
new-replica-set-vpkh8  0/1     ImagePullBackOff   0          2m59s
new-replica-set-7r2qw  0/1     ImagePullBackOff   0          2m59s
new-replica-set-tn2mp  0/1     ImagePullBackOff   0          2m59s

controlplane ~ ➜  kubectl delete pod new-replica-set-wkzjh
pod "new-replica-set-wkzjh" deleted

controlplane ~ ➜
```

Why are there still 4 PODs, even after you deleted one?

I didn't delete it properly

ReplicaSet ensures that desired number of PODs always run

You cannot delete a POD from a ReplicaSet

```
controlplane ~ ➜  kubectl delete pod new-replica-set-wkzjh
pod "new-replica-set-wkzjh" deleted

controlplane ~ ➜  kubectl get pods
NAME                  READY   STATUS         RESTARTS   AGE
new-replica-set-hcmbw   0/1     ErrImagePull   0          17s
new-replica-set-vpkh8   0/1     ErrImagePull   0          3m26s
new-replica-set-7r2qw   0/1     ErrImagePull   0          3m26s
new-replica-set-tn2mp   0/1     ErrImagePull   0          3m26s

controlplane ~ ➜
```

Create a ReplicaSet using the `replicaset-definition-1.yaml` file located at `/root/`. There is an issue with the file, so try to fix it.

Check

- Name: replicaset-1

```
new-replica-set-7r2qw   0/1     ErrImagePull   0          3m26s
new-replica-set-tn2mp   0/1     ErrImagePull   0          3m26s

controlplane ~ ➜  ls /root
replicaset-definition-1.yaml  replicaset-definition-2.yaml  sample.yaml

controlplane ~ ➜  kubectl create -f /root/replicaset-definition-1.yaml
error: unable to recognize "/root/replicaset-definition-1.yaml": no matches for kind "ReplicaSet" in ver
sion "v1"

controlplane ~ ✖ vi /root/replicaset-definition-1.yaml

controlplane ~ ➜  cat /root/replicaset-definition-1.yaml
apiVersion: v1
kind: ReplicaSet
metadata:
  name: replicaset-1
spec:
  replicas: 2
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: nginx
        image: nginx

controlplane ~ ➜
```

1x 6:10 / 14:09

Use below command

```
controlplane ~ ✖ kubectl explain replicaset
KIND:     ReplicaSet
VERSION:  apps/v1

DESCRIPTION:
     ReplicaSet ensures that a specified number of pod replicas are running at
     any given time.

FIELDS:
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Fix the issue in the `replicaset-definition-2.yaml` file and create a `ReplicaSet` using it.

This file is located at `/root/`.

Check

- Name: replicaset-2

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: replicaset-2
spec:
  replicas: 2
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
```

Delete the two newly created ReplicaSets - `replicaset-1` and `replicaset-2`

Next

- ✔ Delete: replicaset-2
- ✔ Delete: replicaset-1

[redacted]

```
controlplane ~ ➜  kubectl get rs
NAME              DESIRED   CURRENT   READY   AGE
new-replica-set   4         4         0       10m
replicaset-1      2         2         2       3m4s
replicaset-2      2         2         2       22s

controlplane ~ ➜  kubectl delete rs replicaset-1
replicaset.apps "replicaset-1" deleted

controlplane ~ ➜  kubectl delete rs replicaset-2
replicaset.apps "replicaset-2" deleted

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Fix the original replica set `new-replica-set` to use the correct `busybox` image.

Either delete and recreate the ReplicaSet or Update the existing ReplicaSet and then delete all PODs, so new ones with the correct image will be created.

Check

- Replicas: 4

```
controlplane ~ ➜  kubectl edit rs new-replica-set
replicaset.apps/new-replica-set edited

controlplane ~ ➜

controlplane ~ ➜  kubectl get pods
NAME                    READY   STATUS             RESTARTS   AGE
new-replica-set-hcmbw   0/1     ImagePullBackOff   0          9m1s
new-replica-set-vpkh8   0/1     ImagePullBackOff   0          12m
new-replica-set-tn2mp   0/1     ImagePullBackOff   0          12m
new-replica-set-7r2qw   0/1     ImagePullBackOff   0          12m

controlplane ~ ➜  kubectl get rs
NAME              DESIRED   CURRENT   READY   AGE
new-replica-set   4         4         0       12m

controlplane ~ ➜  kubectl get rs
NAME              DESIRED   CURRENT   READY   AGE
new-replica-set   4         4         0       12m

controlplane ~ ➜  kubectl get pods
NAME                    READY   STATUS             RESTARTS   AGE
new-replica-set-hcmbw   0/1     ImagePullBackOff   0          9m19s
new-replica-set-vpkh8   0/1     ImagePullBackOff   0          12m
new-replica-set-tn2mp   0/1     ImagePullBackOff   0          12m
new-replica-set-7r2qw   0/1     ImagePullBackOff   0          12m
```

```
controlplane ~ ➜  kubectl get pods
NAME                    READY   STATUS             RESTARTS   AGE
new-replica-set-vpkh8   0/1     ImagePullBackOff   0          14m
new-replica-set-tn2mp   0/1     ImagePullBackOff   0          14m
new-replica-set-7r2qw   0/1     ImagePullBackOff   0          14m
new-replica-set-hcmbw   0/1     ErrImagePull       0          10m

controlplane ~ ➜  kubectl delete pod new-replica-set-vpkh8 new-replica-set-tn2mp new-replica-set-7r2qw n
ew-replica-set-hcmbw
pod "new-replica-set-vpkh8" deleted
pod "new-replica-set-tn2mp" deleted
pod "new-replica-set-7r2qw" deleted
pod "new-replica-set-hcmbw" deleted

controlplane ~ ➜

controlplane ~ ➜

controlplane ~ ➜  kubectl get pods
NAME                    READY   STATUS              RESTARTS   AGE
new-replica-set-8z7z5   0/1     ContainerCreating   0          12s
new-replica-set-f5gth   0/1     ContainerCreating   0          12s
new-replica-set-nsbgx   0/1     ContainerCreating   0          12s
new-replica-set-whhll   0/1     ContainerCreating   0          12s
```

```
new-replica-set-whhll   0/1     ContainerCreating   0          12s

controlplane ~ ➜  kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
new-replica-set-f5gth   1/1     Running   0          18s
new-replica-set-nsbgx   1/1     Running   0          18s
new-replica-set-8z7z5   1/1     Running   0          18s
new-replica-set-whhll   1/1     Running   0          18s

controlplane ~ ➜  kubectl get rs
NAME                DESIRED   CURRENT   READY   AGE
new-replica-set     4         4         4       15m

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Scale the ReplicaSet to 5 PODs.

Use `kubectl scale` command or edit the replicaset using `kubectl edit replicaset`.

Complete Next

✔ Replicas: 5

```
controlplane ~ ➜  kubectl scale rs new-replica-set --replicas=5
replicaset.apps/new-replica-set scaled

controlplane ~ ➜  kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
new-replica-set-f5gth   1/1     Running   0          55s
new-replica-set-nsbgx   1/1     Running   0          55s
new-replica-set-8z7z5   1/1     Running   0          55s
new-replica-set-whhll   1/1     Running   0          55s
new-replica-set-mwpnz   1/1     Running   0          4s

controlplane ~ ➜  kubectl edit rs new-replica-set
Edit cancelled, no changes made.

controlplane ~ ➜
```

Deployment

Wednesday, June 5, 2024 8:02 PM

Definition

deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
  labels:
      app: myapp
      type: front-end
spec:
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx
  replicas: 3
  selector:
    matchLabels:
      type: front-end
```

```
> kubectl create -f deployment-definition.yml
deployment "myapp-deployment" created
```

```
> kubectl get deployments
NAME              DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE

myapp-deployment   3         3         3            3           21s
```

```
> kubectl get replicaset
NAME                         DESIRED   CURRENT   READY     AGE

myapp-deployment-6795844b58   3         3         3         2m
```

```
> kubectl get pods
NAME                                READY     STATUS    RESTARTS   AGE
myapp-deployment-6795844b58-5rbjl   1/1       Running   0          2m
myapp-deployment-6795844b58-h4w55   1/1       Running   0          2m
myapp-deployment-6795844b58-lfjhv   1/1       Running   0          2m
```

1 2 3 4 5 6 7 8 9 10 11

What is the image used to create the pods in the new deployment?

NGINX

BUSYBOX-CONTAINER

BUSYBOX777

BUSYBOX-POD

```
frontend-deployment-7fd8cdb696-jbr44   0/1     ErrImagePull       0          59s

controlplane ~ ➜  kubectl describe pod frontend-deployment-7fd8cdb696-stmbx
Name:         frontend-deployment-7fd8cdb696-stmbx
Namespace:    default
Priority:     0
Node:         controlplane/172.25.0.4
Start Time:   Fri, 15 Apr 2022 19:14:37 +0000
Labels:       name=busybox-pod
              pod-template-hash=7fd8cdb696
Annotations:  <none>
Status:       Pending
IP:           10.42.0.10
IPs:
  IP:           10.42.0.10
Controlled By:  ReplicaSet/frontend-deployment-7fd8cdb696
Containers:
  busybox-container:
    Container ID:
    Image:          busybox888
    Image ID:
    Port:           <none>
    Host Port:      <none>
    Command:
      sh
      -c
      echo Hello Kubernetes! && sleep 3600
    State:          Waiting
      Reason:       ErrImagePull
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-w8929 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             False
```

1 2 3 4 5 6 7 8 9 10 11

Create a new Deployment using the `deployment-definition-1.yaml` file located at `/root/`. There is an issue with the file, so try to fix it.

Check

- Name: deployment-1

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-1
spec:
  replicas: 2
  selector:
    matchLabels:
      name: busybox-pod
  template:
    metadata:
      labels:
        name: busybox-pod
    spec:
      containers:
      - name: busybox-container
        image: busybox888
        command:
        - sh
        - "-c"
        - echo Hello Kubernetes! && sleep 3600
```

deployment was written. It is case sensitive

1 2 3 4 5 6 7 8 9 10 11

Create a new Deployment with the below attributes using your own deployment definition file.

Name: `httpd-frontend`;

Replicas: `3`;

Image: `httpd:2.4-alpine`

Check

- Name: httpd-frontend
- Replicas: 3
- Image: httpd:2.4-alpine

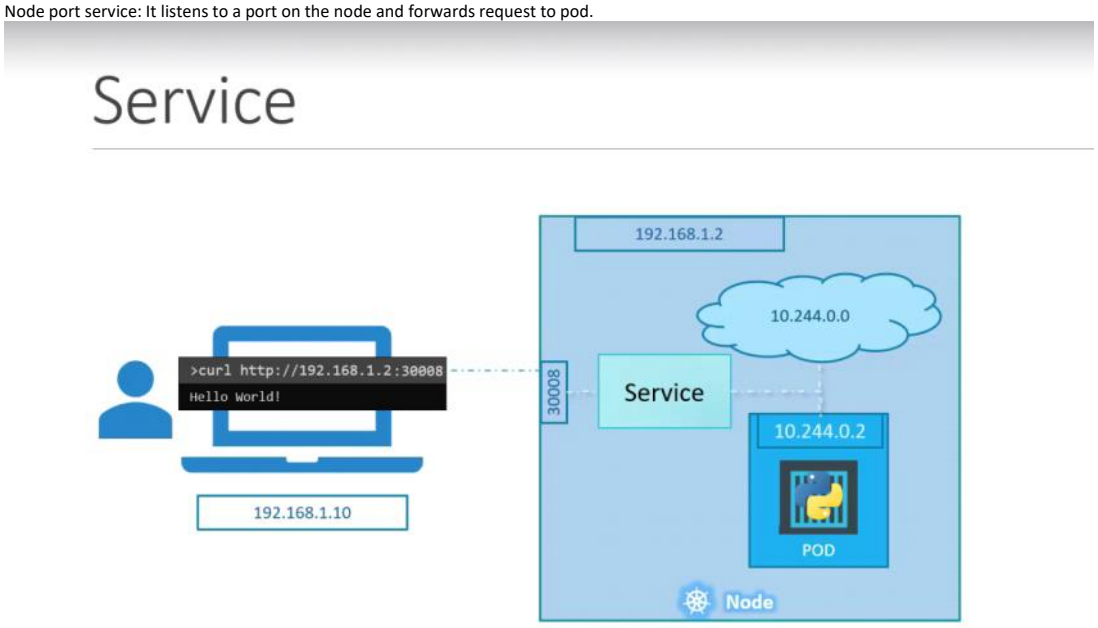
```
  kubectl create deployment my-dep --image=busybox --port=5701

Options:
      --allow-missing-template-keys=true: If true, ignore any errors in templates when a field or
map key is missing in the template. Only applies to golang and jsonpath output formats.
      --dry-run='none': Must be "none", "server", or "client". If client strategy, only print the
object that would be sent, without sending it. If server strategy, submit server-side request
without persisting the resource.
      --field-manager='kubectl-create': Name of the manager used to track field ownership.
      --image=[]: Image names to run.
  -o, --output='': Output format. One of:
json|yaml|name|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-file
.
      --port=-1: The port that this container exposes.
  -r, --replicas=1: Number of replicas to create. Default is 1.
      --save-config=false: If true, the configuration of current object will be saved in its
annotation. Otherwise, the annotation will be unchanged. This flag is useful when you want to
perform kubectl apply on this object in the future.
      --show-managed-fields=false: If true, keep the managedFields when printing objects in JSON or
YAML format.
      --template='': Template string or path to template file to use when -o=go-template,
-o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
      --validate=true: If true, use a schema to validate the input before sending it

Usage:
  kubectl create deployment NAME --image=image -- [COMMAND] [args...] [options]

Use "kubectl options" for a list of global command-line options (applies to all commands).

controlplane ~ ➜  kubectl create deployment httpd-frontend --image=httpd:2.4-alpine --replicas=3
deployment.apps/httpd-frontend created

controlplane ~ ➜  kubectl get deploy
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
frontend-deployment   0/4     4            0           5m58s
deployment-1          0/2     2            0           96s
httpd-frontend        0/3     3            0           6s
```

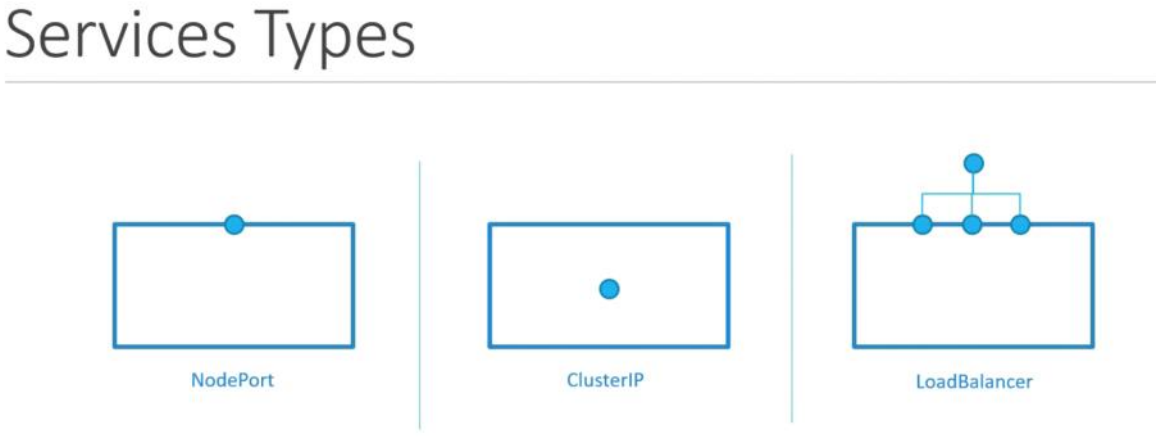
Services

Wednesday, June 5, 2024 9:51 PM

Notes

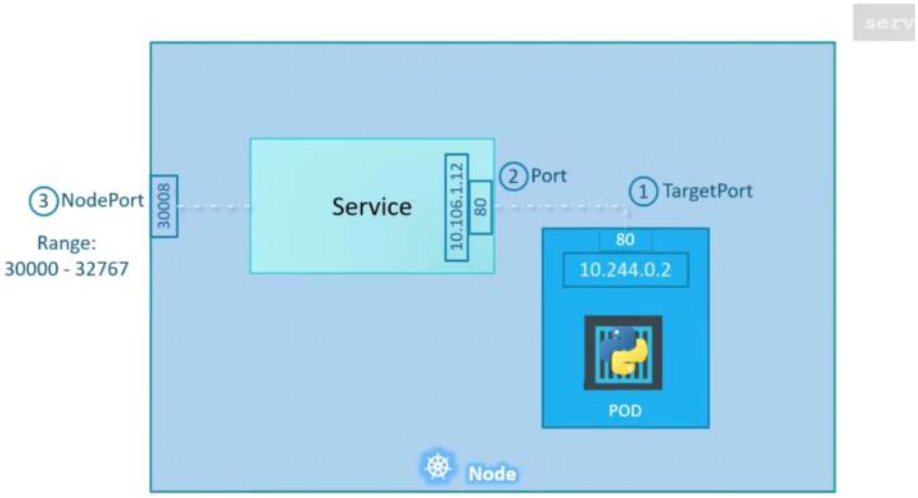
Node port service: It listens to a port on the node and forwards request to pod.



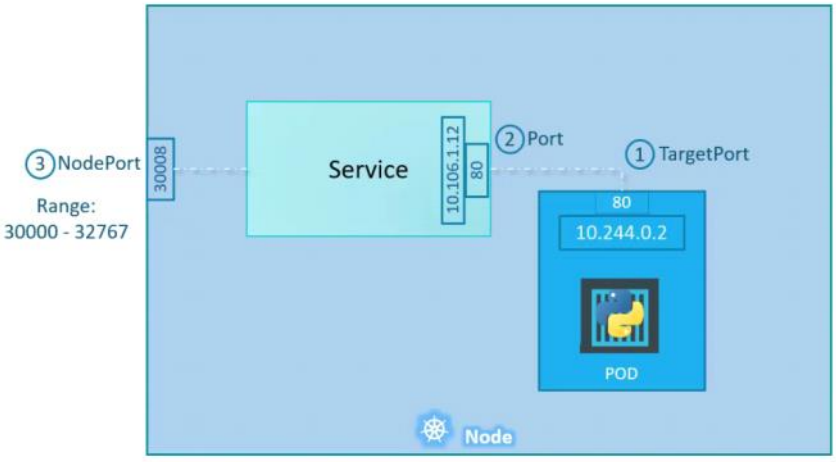


Service Node port

Service - NodePort



Service - NodePort



service-definition.yml

```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: NodePort
    ports:
     - targetPort: 80
      *port: 80
       nodePort: 30008
```

Service - NodePort

service-definition.yml

```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: NodePort
    ports:
     - targetPort: 80
       port: 80
       nodePort: 30008
    selector:
       app: myapp
       type: front-end
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:


spec:
  containers:
  - name: nginx-c
    image: nginx
```

Service - NodePort

service-definition.yml

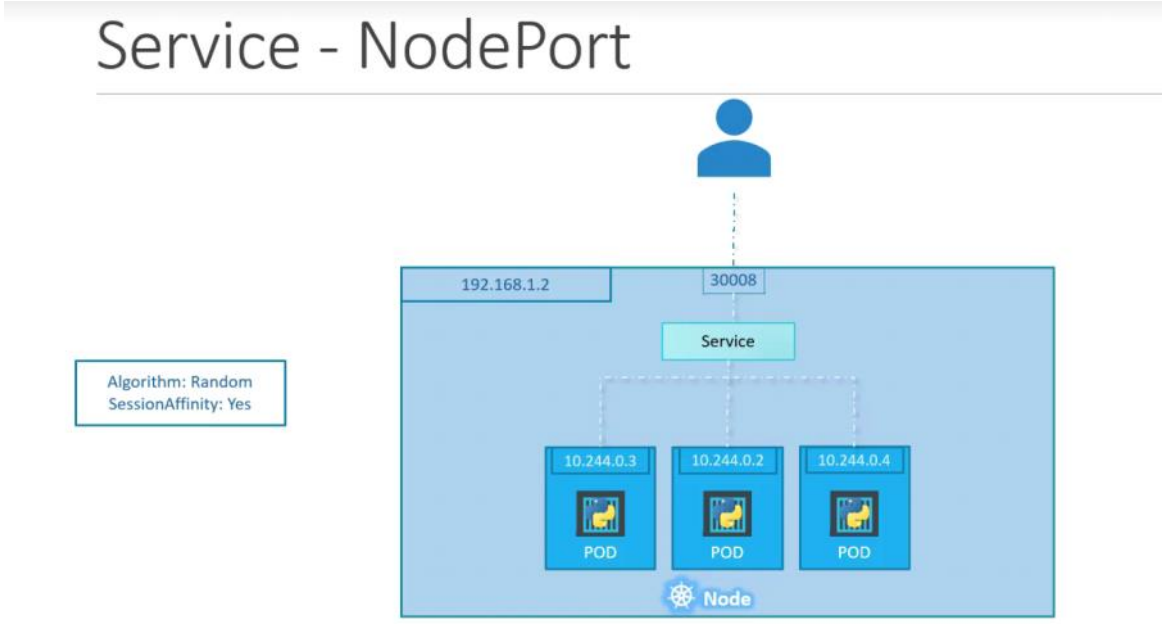
```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: NodePort
    ports:
     - targetPort: 80
       port: 80
       nodePort: 30008
    selector:
       app: myapp
       type: front-end
```

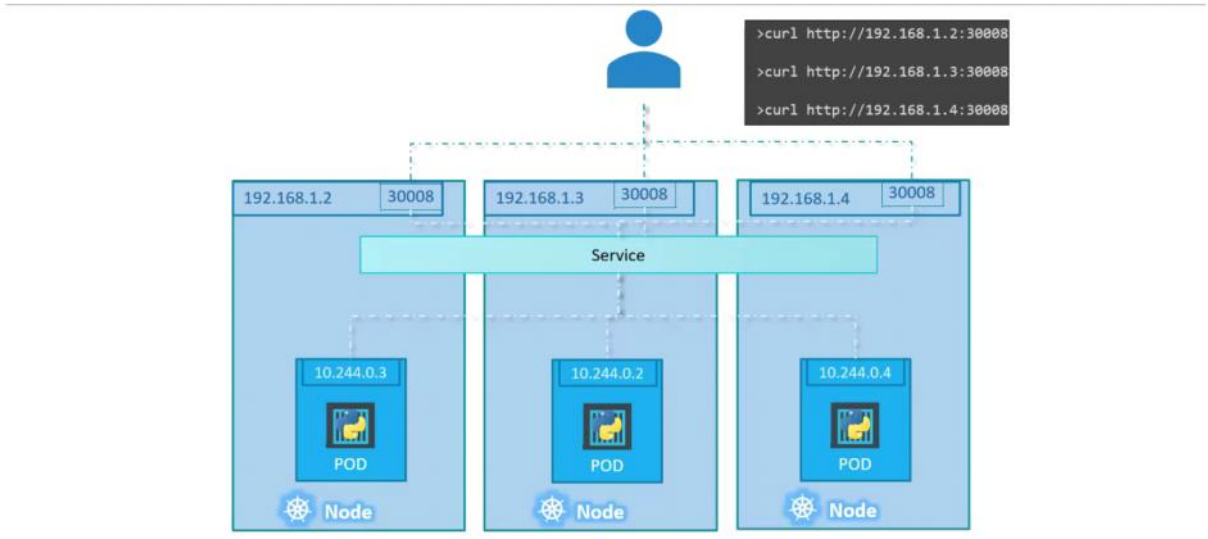
```
> kubectl create -f service-definition.yml
service "myapp-service" created
```

```
> kubectl get services
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP        16d
myapp-service   NodePort    10.106.127.123   <none>        80:30008/TCP   5m
```

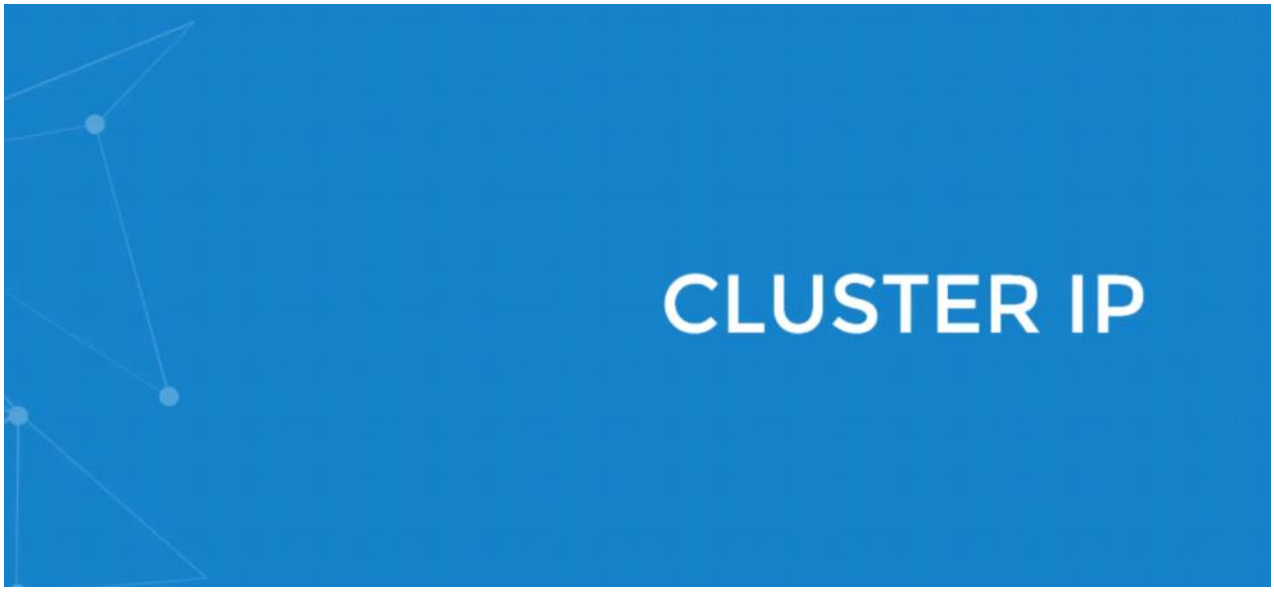
```
> curl http://192.168.1.2:30008
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
```

And then I use the port 30,008 to access the web server.

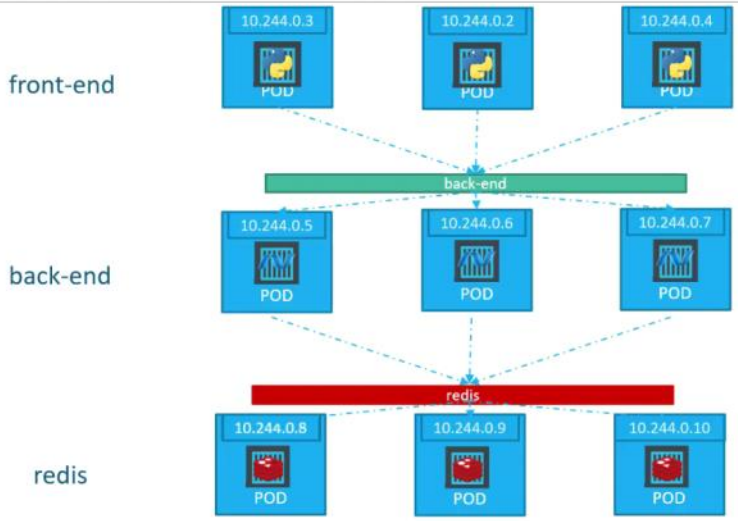

Service - NodePort
192.168.1.2
30008
Service
Algorithm: Random
SessionAffinity: Yes
10.244.0.3
10.244.0.2
10.244.0.4
POD
POD
POD
Node


>curl http://192.168.1.2:30008
>curl http://192.168.1.3:30008
>curl http://192.168.1.4:30008
192.168.1.2
30008
192.168.1.3
30008
192.168.1.4
30008
Service
10.244.0.3
10.244.0.2
10.244.0.4
POD
POD
POD
Node
Node
Node

Cluster IP


CLUSTER IP

ClusterIP



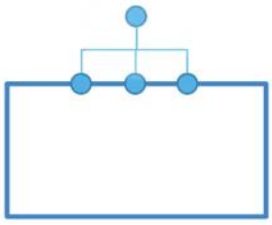
service-definition.yml

```yaml
apiVersion: v1
kind: Service
metadata:
    name: back-end

spec:
    type: ClusterIP
    ports:
     - targetPort: 80
       port: 80

    selector:
       app: myapp
       type: back-end
```

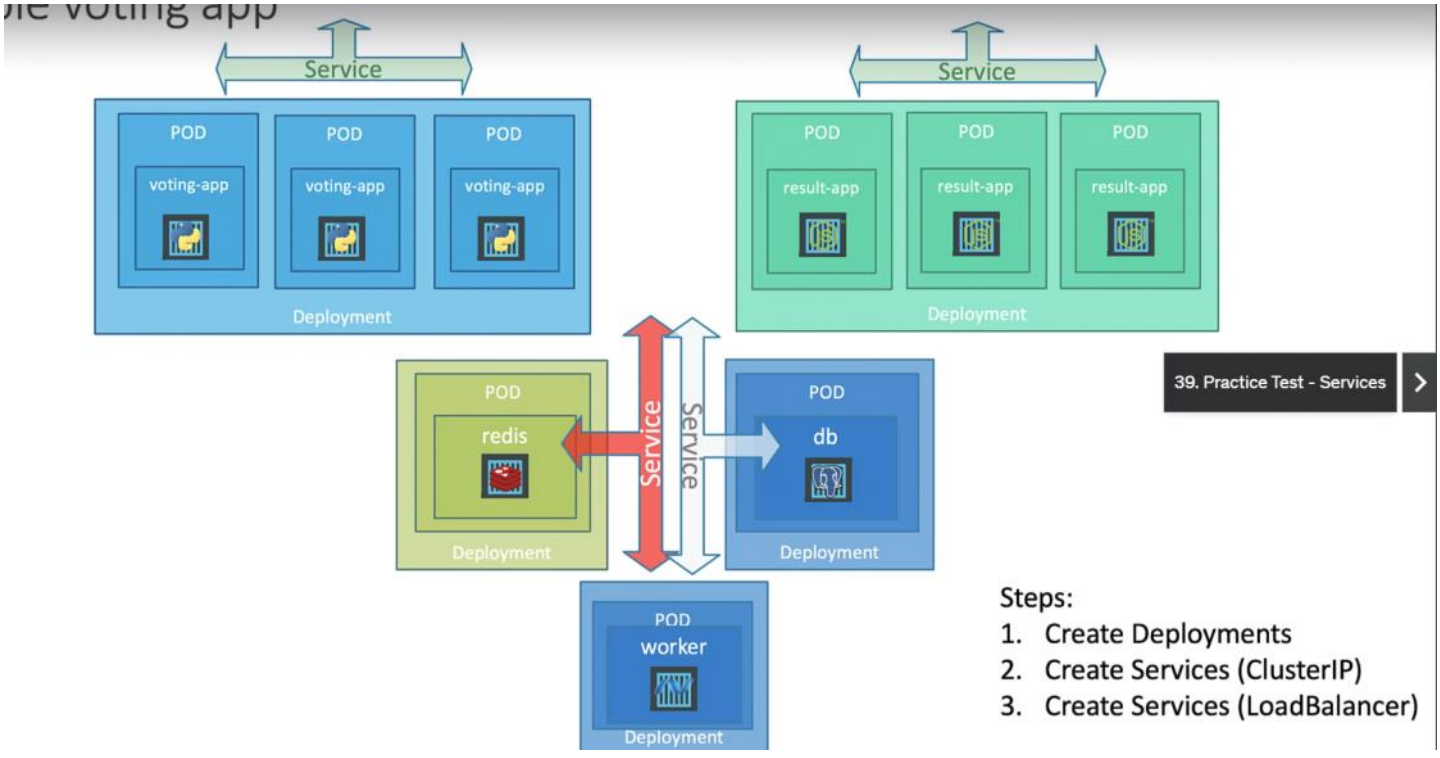
```
> kubectl create -f service-definition.yml
service "back-end" created
```

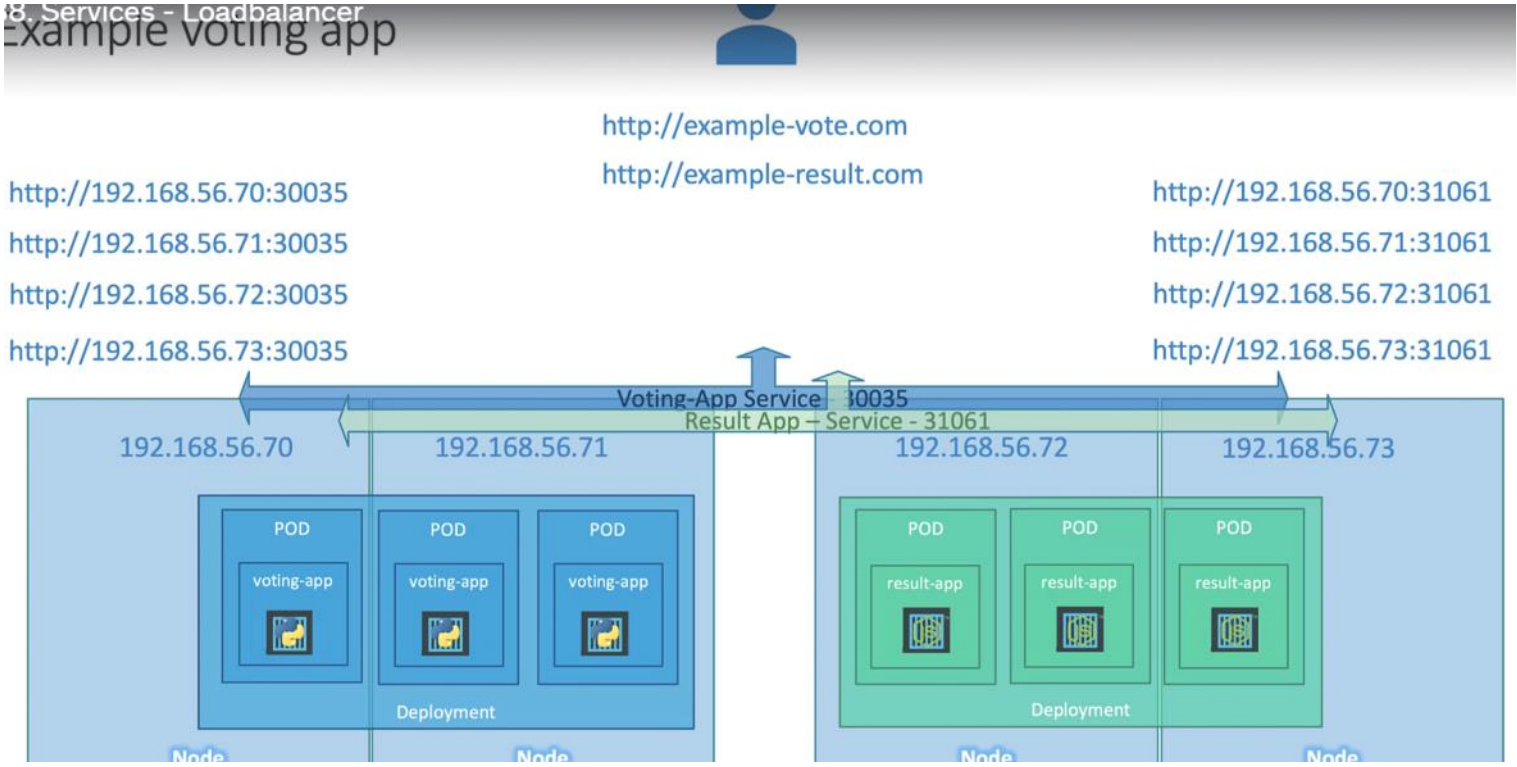
```
> kubectl get services
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP   16d
back-end      ClusterIP   10.106.127.123   <none>        80/TCP    2m
```

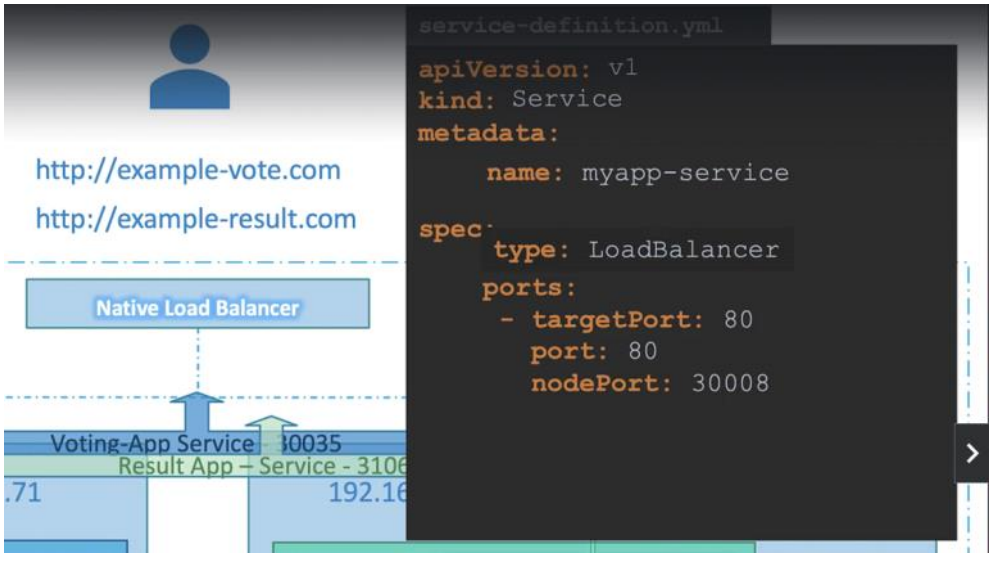
Service-
Load Balancer



6/5/2024 10:21 PM




8. Services - Loadbalancer
Example voting app
http://example-vote.com
http://example-result.com
http://192.168.56.70:30035
http://192.168.56.71:30035
http://192.168.56.72:30035
http://192.168.56.73:30035
http://192.168.56.70:31061
http://192.168.56.71:31061
http://192.168.56.72:31061
http://192.168.56.73:31061
Voting-App Service
Result App – Service - 31061
192.168.56.70
192.168.56.71
192.168.56.72
192.168.56.73
POD
voting-app
Deployment
POD
result-app
Deployment
Node


service-definition.yml
apiVersion: v1
kind: Service
metadata:
name: myapp-service
spec:
type: LoadBalancer
ports:
- targetPort: 80
port: 80
nodePort: 30008
http://example-vote.com
http://example-result.com
Native Load Balancer
Voting-App Service
Result App – Service - 3106

Practice- services

Wednesday, June 5, 2024 10:23 PM

1 2 3 4 5 6 7 8 9 10 11

How many Services exist on the system?
in the current(default) namespace

0

...

2

3

4

```
controlplane ~ ➜  kubectl get service
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   10m

controlplane ~ ➜  kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   10m

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

What is the `targetPort` configured on the `kubernetes` service?

8080

443

10.96.0.1

6443

```
controlplane ~ ➜  kubectl get service
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   10m

controlplane ~ ➜  kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   10m

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

What is the type of the default `kubernetes` service?

LoadBalancer

NodePort

ClusterIP

External

```
controlplane ~ ➜  kubectl get service
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   10m

controlplane ~ ➜  kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   10m

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

What is the `targetPort` configured on the `kubernetes` service?

8080

[redacted]

```
controlplane ~ ➜  kubectl describe svc kubernetes
Name:              kubernetes
Namespace:         default
Labels:            component=apiserver
```

1 2 3 4 5 6 7 8 9 10 11

What is the `targetPort` configured on the `kubernetes` service?

- 8080
- 443
- 10.96.0.1
- 6443

[redacted]

```
controlplane ~ ➜  kubectl describe svc kubernetes
Name:              kubernetes
Namespace:         default
Labels:            component=apiserver
                   provider=kubernetes
Annotations:       <none>
Selector:          <none>
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.43.0.1
IPs:               10.43.0.1
Port:              https  443/TCP
TargetPort:        6443/TCP
Endpoints:         10.53.180.9:6443
Session Affinity:  None
Events:            <none>

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

How many labels are configured on the `kubernetes` service?

- 0
- 1
- 2
- 3

```
controlplane ~ ➜  kubectl get service
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   10m

controlplane ~ ➜  kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   10m

controlplane ~ ➜  kubectl describe svc kubernetes
Name:              kubernetes
Namespace:         default
Labels:            component=apiserver
                   provider=kubernetes
Annotations:       <none>
Selector:          <none>
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.43.0.1
IPs:               10.43.0.1
Port:              https  443/TCP
TargetPort:        6443/TCP
Endpoints:         10.53.180.9:6443
Session Affinity:  None
Events:            <none>

controlplane ~ ➜
```

How many Endpoints are attached on the `kubernetes` service?

- 0
- 1
- 2
- 3

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   10m

controlplane ~ ➜  kubectl describe svc kubernetes
Name:              kubernetes
Namespace:         default
Labels:            component=apiserver
                   provider=kubernetes
Annotations:       <none>
Selector:          <none>
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4                                          e | 0 unavailable
IP:                10.43.0.1
IPs:               10.43.0.1
Port:              https  443/TCP
TargetPort:        6443/TCP
Endpoints:         10.53.180.9:6443
Session Affinity:  None
Events:            <none>

controlplane ~ ➜

  Volumes:         <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   simple-webapp-deployment-7b59598d59 (4/4 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  63s   deployment-controller  Scaled up replica set simple-webapp-deployment-
```

```
NewReplicaSet:   simple-webapp-deployment-7b59598d59 (4/4 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  63s   deployment-controller  Scaled up replica set simple-webapp-deployment-
7b59598d59 to 4
```

1 2 3 4 5 6 7 8 9 10 11

How many Deployments exist on the system now? in the current(default) namespace

0

2

3

4

```
controlplane ~ ➜  kubectl get deploy
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
simple-webapp-deployment   0/4     4            0           15s

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9 10 11

What is the image used to create the pods in the deployment?

NGINX

BUSYBOX-CONTAINER

kodekloud/simple-webapp:red

BUSYBOX777

BUSYBOX-POD

```
controlplane ~ ➜  kubectl get deploy
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
simple-webapp-deployment   0/4     4            0           15s

controlplane ~ ➜  kubectl describe deploy simple-webapp-deployment
Name:                   simple-webapp-deployment
Namespace:              default
CreationTimestamp:      Fri, 15 Apr 2022 20:35:47 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               name=simple-webapp
Replicas:               4 desired | 4 updated | 4 total | 4 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  name=simple-webapp
  Containers:
   simple-webapp:
    Image:        kodekloud/simple-webapp:red
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   simple-webapp-deployment-7b59598d59 (4/4 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  63s   deployment-controller  Scaled up replica set simple-webapp-deployment
7b59598d59 to 4
```

1 2 3 4 5 6 7 8 9 10 11

Create a new service to access the web application using the service-definition-1.yaml file

Name: webapp-service

Type: NodePort

targetPort: 8080

port: 8080

nodePort: 30080

selector: simple-webapp

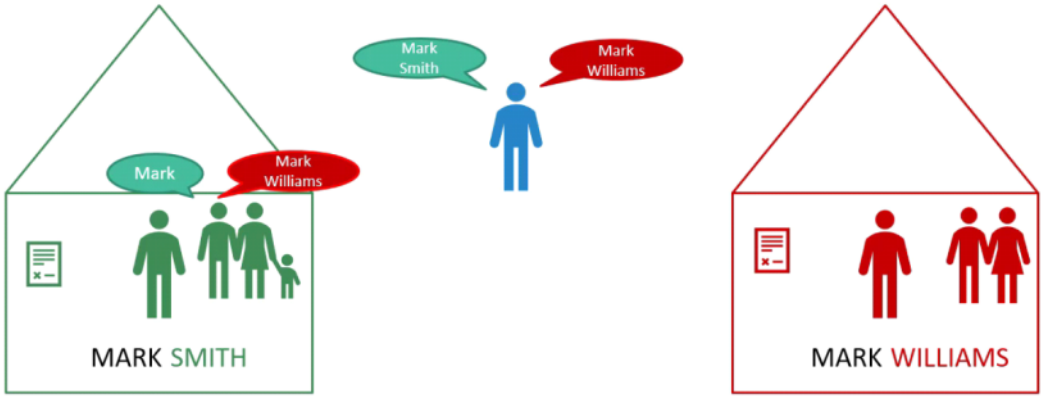
Check

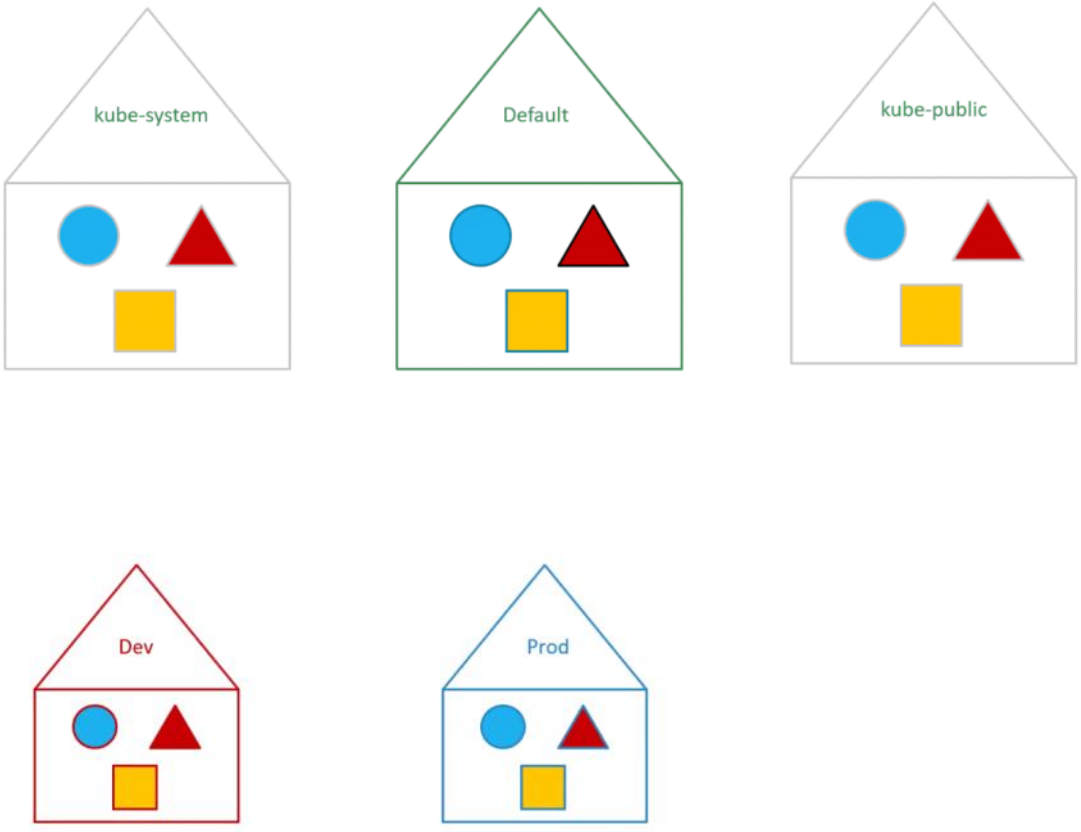
```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: webapp-service
spec:
  type: NodePort
  ports:
    - targetPort: 8080
      port: 8080
      nodePort: 30080
  selector:
    name: simple-webapp
```

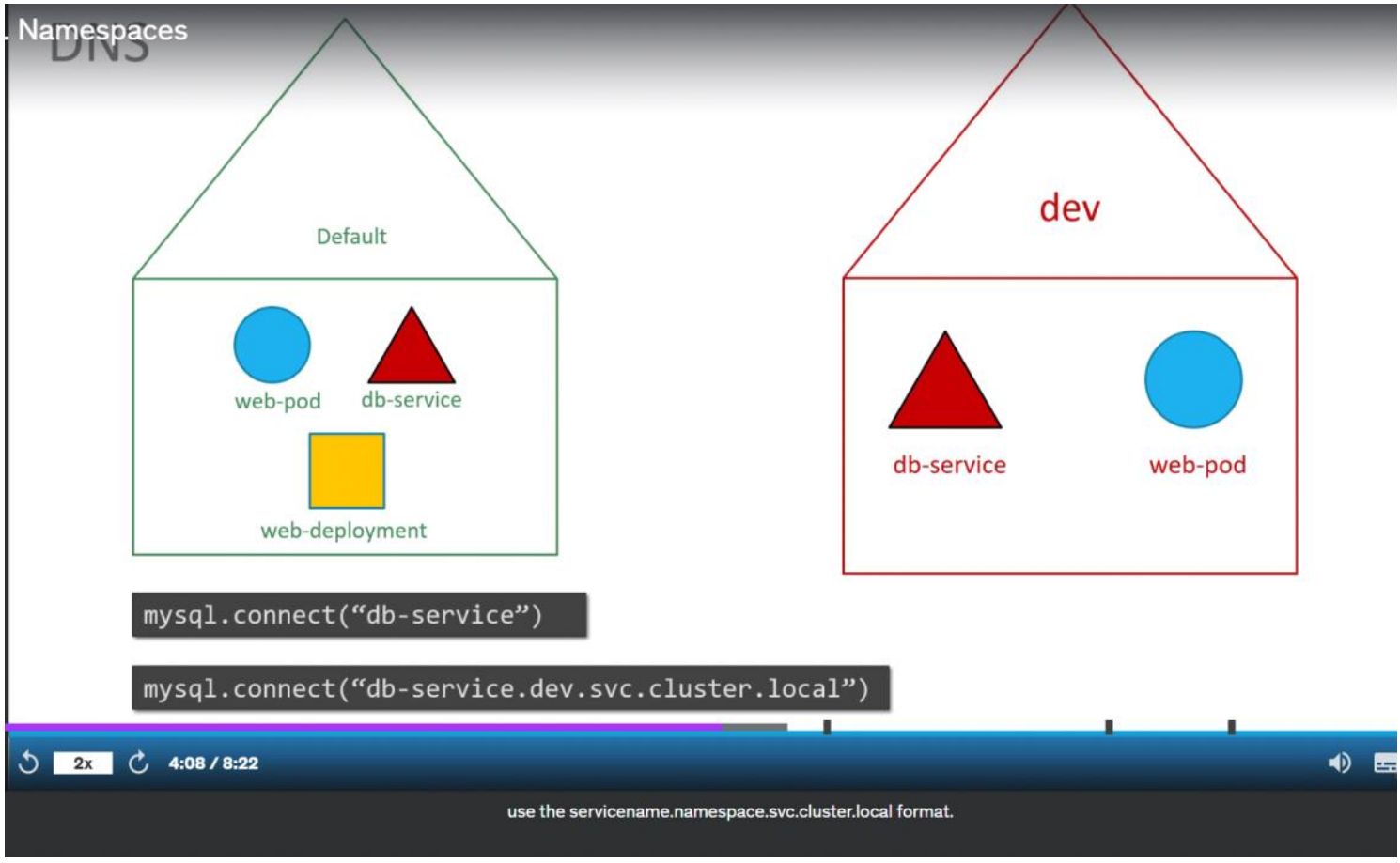
```
    name:

controlplane ~ ➜  vi service-definition-1.yaml

controlplane ~ ➜  kubectl create -f service-definition-1.yaml
```

Namespace

Wednesday, June 5, 2024 10:40 PM





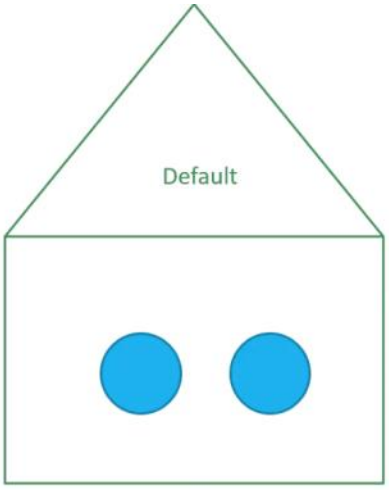


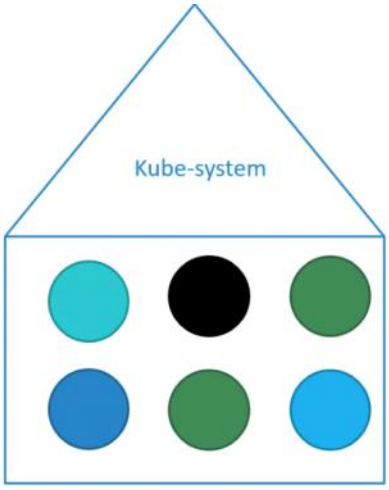
DNS

```
mysql.connect("db-service.dev.svc.cluster.local")
```

Service Name | Namespace | Service | domain

```
> kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
Pod-1     1/1     Running   0          3d
Pod-2     1/1     Running   0          3d
```





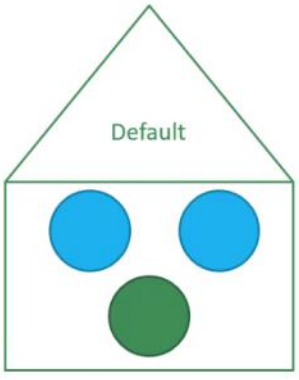

```
> kubectl get pods --namespace=kube-system
NAME                              READY   STATUS    RESTAR
coredns-78fcdf6894-92d52          1/1     Running   7
coredns-78fcdf6894-jx25g          1/1     Running   7
etcd-master                       1/1     Running   7
kube-apiserver-master             1/1     Running   7
kube-controller-manager-master    1/1     Running   7
kube-flannel-ds-amd64-hz4cf       1/1     Running   14
kube-proxy-4b8tn                  1/1     Running   7
kube-proxy-98db4                  1/1     Running   7
kube-proxy-jjrbs                  1/1     Running   7
kube-scheduler-master             1/1     Running   7
```

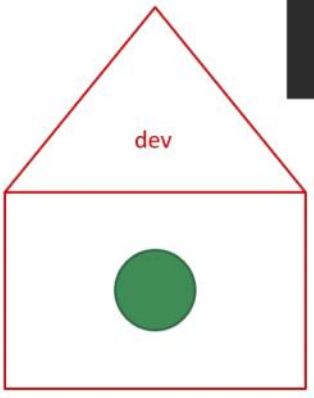
```
> kubectl create -f pod-definition.yml
pod/myapp-pod created
```

```
> kubectl create -f pod-definition.yml --namespace=dev
pod/myapp-pod created
```

pod-definition.yml

```yml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```



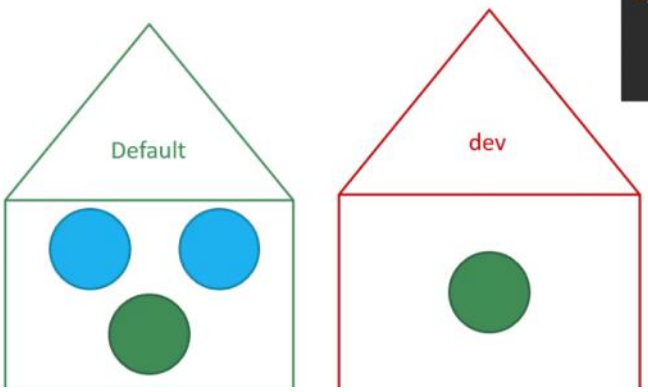


```
> kubectl create -f pod-definition.yml
pod/myapp-pod created
```

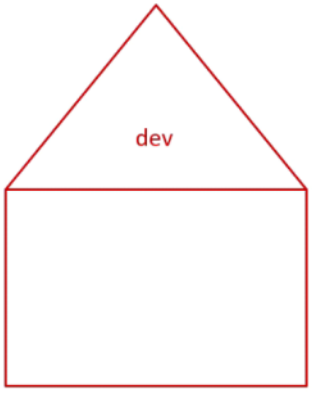
```
> kubectl create -f pod-definition.yml
pod/myapp-pod created
```

pod-definition.yml

```yml
apiVersion: v1
kind: Pod
metadata:
 name: myapp-pod
 namespace: dev
 labels:
    app: myapp
    type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```



Create Namespace



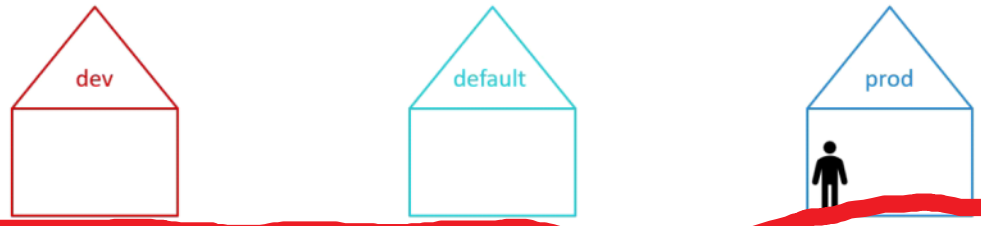
namespace-dev.yml

```yml
apiVersion: v1
kind: Namespace
metadata:
    name: dev
```

```
> kubectl create -f namespace-dev.yml
namespace/dev created
```

```
> kubectl create namespace dev
namespace/dev created
```

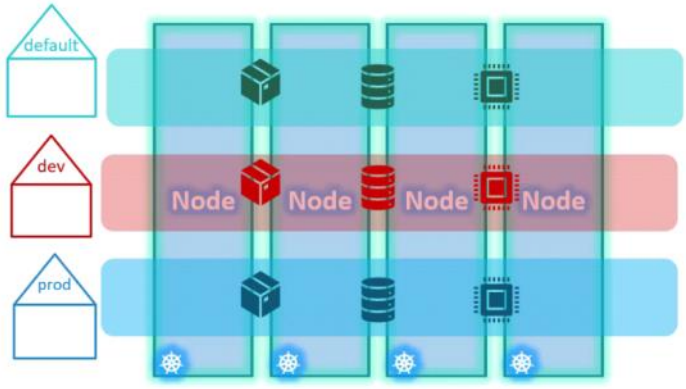
Switch




```
> kubectl get pods --namespace=dev
> kubectl get pods
> kubectl get pods --namespace=prod

> kubectl config set-context $(kubectl config current-context) --namespace=dev

> kubectl get pods
> kubectl get pods --namespace=default
> kubectl get pods --namespace=prod

> kubectl config set-context $(kubectl config current-context) --namespace=prod

> kubectl get pods --namespace=dev
> kubectl get pods --namespace=default
> kubectl get pods

> kubectl get pods --all-namespaces
```

To limit resource in a namespace:

Resource Quota



Compute-quota.yaml

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
    name: compute-quota
    namespace: dev

spec:
  hard:
    pods: "10"
    requests.cpu: "4"
    requests.memory: 5Gi
    limits.cpu: "10"
    limits.memory: 10Gi
```

```
> kubectl create -f compute-quota.yaml
```

Practice- namespaces

Thursday, June 6, 2024 1:27 AM

1 2 3 4 5 6 7

How many Namespaces exist on the system?

0

1

3

5

10

```
controlplane ~ ➜  kubectl get namespaces
NAME              STATUS   AGE
default           Active   6m50s
kube-system       Active   6m49s
kube-public       Active   6m49s
kube-node-lease   Active   6m49s
finance           Active   27s
marketing         Active   27s
dev               Active   27s
prod              Active   27s
manufacturing     Active   27s
research          Active   27s

controlplane ~ ➜  kubectl get ns
NAME              STATUS   AGE
default           Active   6m55s
kube-system       Active   6m54s
kube-public       Active   6m54s
kube-node-lease   Active   6m54s
finance           Active   32s
marketing         Active   32s
dev               Active   32s
prod              Active   32s
manufacturing     Active   32s
research          Active   32s

controlplane ~ ➜
```

How many pods exist in the `research` namespace?

0

1

2

3

4

5

[redacted]

```
controlplane ~ ➜  kubectl get pods --namespace=research
NAME    READY   STATUS             RESTARTS      AGE
dna-2   0/1     CrashLoopBackOff   3 (14s ago)   76s
dna-1   0/1     CrashLoopBackOff   3 (16s ago)   76s

controlplane ~ ➜  kubectl get pods -n=research
NAME    READY   STATUS             RESTARTS      AGE
dna-2   0/1     CrashLoopBackOff   3 (31s ago)   93s
dna-1   0/1     CrashLoopBackOff   3 (33s ago)   93s

controlplane ~ ➜
```

1 2 3 4 5 6 7

Create a POD in the `finance` namespace. Use the spec given below.

Check

- Name: redis
- Image Name: redis

```
controlplane ~ ➜  kubectl run redis --image=redis -n=finance
pod/redis created

controlplane ~ ➜  kubectl get pods -n=finance
NAME      READY   STATUS              RESTARTS   AGE
payroll   1/1     Running             0          2m20s
redis     0/1     ContainerCreating   0          8s

controlplane ~ ➜
```

1 2 3 4 5 6 7

Which namespace has the `blue` pod in it?

- marketing
- research
- manufacturing
- default

```
research          Active   2m38s

controlplane ~ ➜  kubectl get pods --all-namespaces
NAMESPACE       NAME                                       READY   STATUS            RESTARTS      AGE
kube-system     local-path-provisioner-6c79684f77-j5sqx    1/1     Running           0             9m5s
kube-system     coredns-5789895cd-bkj56                    1/1     Running           0             9m5s
kube-system     helm-install-traefik-crd-d2fg7             0/1     Completed         0             9m7s
kube-system     metrics-server-7cd5fcb6b7-kdgzb            1/1     Running           0             9m5s
kube-system     helm-install-traefik-c987s                 0/1     Completed         1             9m7s
kube-system     svclb-traefik-jptsh                        2/2     Running           0             8m30s
kube-system     traefik-6bb96f9bd8-w9wzr                   1/1     Running           0             8m30s
marketing       redis-db                                   1/1     Running           0             3m3s
dev             redis-db                                   1/1     Running           0             3m3s
manufacturing   red-app                                    1/1     Running           0             3m3s
finance         payroll                                    1/1     Running           0             3m3s
marketing       blue                                       1/1     Running           0             3m3s
research        dna-1                                      0/1     CrashLoopBackOff  4 (80s ago)   3m3s
research        dna-2                                      0/1     CrashLoopBackOff  4 (70s ago)   3m3s
finance         redis                                      1/1     Running           0             51s

controlplane ~ ➜  kubectl get pods -A
NAMESPACE       NAME                                       READY   STATUS      RESTARTS      AGE
kube-system     local-path-provisioner-6c79684f77-j5sqx    1/1     Running     0             9m21s
kube-system     coredns-5789895cd-bkj56                    1/1     Running     0             9m21s
kube-system     helm-install-traefik-crd-d2fg7             0/1     Completed   0             9m23s
kube-system     metrics-server-7cd5fcb6b7-kdgzb            1/1     Running     0             9m21s
kube-system     helm-install-traefik-c987s                 0/1     Completed   1             9m23s
kube-system     svclb-traefik-jptsh                        2/2     Running     0             8m46s
kube-system     traefik-6bb96f9bd8-w9wzr                   1/1     Running     0             8m46s
marketing       redis-db                                   1/1     Running     0             3m19s
dev             redis-db                                   1/1     Running     0             3m19s
manufacturing   red-app                                    1/1     Running     0             3m19s
finance         payroll                                    1/1     Running     0             3m19s
marketing       blue                                       1/1     Running     0             3m19s
finance         redis                                      1/1     Running     0             67s
research        dna-1                                      0/1     Completed   5 (96s ago)   3m19s
research        dna-2                                      0/1     Completed   5 (86s ago)   3m19s

controlplane ~ ➜
```

1 2 3 4 5 6 7

What DNS name should the Blue application use to access the database `db-service` in its own namespace - `marketing`.
You can try it in the webapplication UI. Use port `6379`.

- db-service
- blue-db-service
- db
- redis

```
controlplane ~ ➜  kubectl get pods -n=marketing
NAME       READY   STATUS    RESTARTS   AGE
redis-db   1/1     Running   0          4m14s
blue       1/1     Running   0          4m14s

controlplane ~ ➜  kubectl get svc -n=marketing
NAME           TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
blue-service   NodePort   10.43.82.162   <none>        8080:30082/TCP   4m27s
db-service     NodePort   10.43.134.33   <none>        6379:30758/TCP   4m27s

controlplane ~ ➜
```

```
controlplane ~ ➜  kubectl get svc -n marketing
NAME          TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
blue-service  NodePort   10.43.208.36   <none>        8080:30082/TCP   9m8s
db-service    NodePort   10.43.2.155    <none>        6379:30135/TCP   9m8s
```

Imperative vs Declarative

Thursday, June 6, 2024 9:37 AM

Infrastructure as Code

Imperative

```
1. Provision a VM by the name 'web-server'
2. Install NGINX Software on it
3. Edit configuration file to use port '8080'
4. Edit configuration file to web path '/var/www/nginx'
5. Load web pages to '/var/www/nginx' from GIT Repo - X
6. Start NGINX server
```

Declarative

```
VM Name: web-server
Package:  nginx:1.18
Port: 8080
Path: /var/www/nginx
Code: GIT Repo - X
```

g

Kubernetes

Imperative

```
> kubectl run --image=nginx nginx
> kubectl create deployment --image=nginx nginx
> kubectl expose deployment nginx --port 80
> kubectl edit deployment nginx
> kubectl scale deployment nginx --replicas=5
> kubectl set image deployment nginx nginx=nginx:1.18
> kubectl create -f nginx.yaml
> kubectl replace -f nginx.yaml
> kubectl delete -f nginx.yaml
```

Declarative

```
> kubectl apply -f nginx.yaml
```

Imperative Commands

Create Objects

```
> kubectl run --image=nginx nginx
```

```
> kubectl create deployment --image=nginx nginx
```

```
> kubectl expose deployment nginx --port 80
```

Update Objects

```
> kubectl edit deployment nginx
```

```
> kubectl scale deployment nginx --replicas=5
```

```
> kubectl set image deployment nginx nginx=nginx:1.18
```

Imperative Object Configuration Files

Create Objects

```
> kubectl create -f nginx.yaml
```

nginx.yaml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end-service
spec:
  containers:
  - name: nginx-container
    image: nginx:1.18
```

Update Objects

```
> kubectl edit deployment nginx
```

```
> kubectl replace -f nginx.yaml
```

```
> kubectl replace --force -f nginx.yaml
```

```
> kubectl create -f nginx.yaml
Error from server (AlreadyExists): error when creating "nginx.yaml": pods "myapp-pod" already exists
```

```
> kubectl replace -f nginx.yaml
Error from server (Conflict): error when replacing "nginx.yaml": Operation cannot be fulfilled on pods
"myapp-pod"
```

Declarative

Create Objects

```
> kubectl apply -f nginx.yaml
```

```
> kubectl apply -f /path/to/config-files
```

Update Objects

```
> kubectl apply -f nginx.yaml
```

nginx.yaml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end-service
spec:
  containers:
  - name: nginx-container
    image: nginx:1.18
```

Exam Tips

Create Objects

```
> kubectl apply -f nginx.yaml
```

```
> kubectl run --image=nginx nginx
```

```
> kubectl create deployment --image=nginx nginx
```

```
> kubectl expose deployment nginx --port 80
```

Update Objects

```
> kubectl apply -f nginx.yaml
```

```
> kubectl edit deployment nginx
```

```
> kubectl scale deployment nginx --replicas=5
```

```
> kubectl set image deployment nginx nginx=nginx:1.18
```

Certification Tips - Imperative Commands with Kubectl

While you would be working mostly the declarative way - using definition files, imperative commands can help in getting one time tasks done quickly, as well as generate a definition template easily. This would help save considerable amount of time during your exams.

Before we begin, familiarize with the two options that can come in handy while working with the below commands:

--dry-run: By default as soon as the command is run, the resource will be created. If you simply want to test your command , use the --dry-run=client option. This will not create the resource, instead, tell you whether the resource can be created and if your command is right.

-o yaml: This will output the resource definition in YAML format on screen.

Use the above two in combination to generate a resource definition file quickly, that you can then modify and create resources as required, instead of creating the files from scratch.

POD
Create an NGINX Pod
kubectl run nginx --image=nginx

Generate POD Manifest YAML file (-o yaml). Don't create it(--dry-run)
kubectl run nginx --image=nginx --dry-run=client -o yaml

Deployment
Create a deployment
kubectl create deployment --image=nginx nginx

Generate Deployment YAML file (-o yaml). Don't create it(--dry-run)
kubectl create deployment --image=nginx nginx --dry-run=client -o yaml

Generate Deployment with 4 Replicas
kubectl create deployment nginx --image=nginx --replicas=4

You can also scale a deployment using the kubectl scale command.
kubectl scale deployment nginx --replicas=4
Another way to do this is to save the YAML definition to a file and modify
kubectl create deployment nginx --image=nginx --dry-run=client -o yaml > nginx-deployment.yaml

You can then update the YAML file with the replicas or any other field before creating the deployment.

Service
Create a Service named redis-service of type ClusterIP to expose pod redis on port 6379
kubectl expose pod redis --port=6379 --name redis-service --dry-run=client -o yaml
(This will automatically use the pod's labels as selectors)
Or
kubectl create service clusterip redis --tcp=6379:6379 --dry-run=client -o yaml (This will not use the pods labels as selectors, instead it will assume selectors as **app=redis.** You cannot pass in selectors as an option. So it does not work very well if your pod has a different label set. So generate the file and modify the selectors before creating the service)

Create a Service named nginx of type NodePort to expose pod nginx's port 80 on port 30080 on the nodes:
kubectl expose pod nginx --type=NodePort --port=80 --name=nginx-service --dry-run=client -o yaml
(This will automatically use the pod's labels as selectors, but you cannot specify the node port. You have to generate a definition file and then add the node port in manually before creating the service with the pod.)
Or
kubectl create service nodeport nginx --tcp=80:80 --node-port=30080 --dry-run=client -o yaml
(This will not use the pods labels as selectors)
Both the above commands have their own challenges. While one of it cannot accept a selector the other cannot accept a node port. I would recommend going with the kubectl expose command. If you need to specify a node port, generate a definition file using the same command and manually input the nodeport before creating the service.

Practice- imperative

Thursday, June 6, 2024 9:49 AM

Task Hint Solution 58:40 Terminal 1 +

1 2 3 4 5 6 7 8 9

Deploy a pod named `nginx-pod` using the `nginx:alpine` image.
Use imperative commands only.

Check

- Name: nginx-pod
- Image: nginx:alpine

```
controlplane ~ ➜  kubectl run nginx-pod --image=nginx:alpine
pod/nginx-pod created

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9

Deploy a `redis` pod using the `redis:alpine` image with the labels set to `tier=db`.
Either use imperative commands to create the pod with the labels. Or else use imperative commands to generate the pod definition file, then add the labels before creating the pod using the file.

Complete Next

- Pod Name: redis
- Image: redis:alpine
- Labels: tier=db

```
controlplane ~ ➜  kubectl run redis --image=redis:alpine --labels="tier=db"
pod/redis created

controlplane ~ ➜
```

kubectl run redis --image redis:alpine -l tier=db

1 2 3 4 5 6 7 8 9

Create a service `redis-service` to expose the `redis` application within the cluster on port `6379`.
Use imperative commands.

Check

- Service: redis-service
- Port: 6379
- Type: ClusterIP

```
controlplane ~ ➜  kubectl create service --help
Create a service using a specified subcommand.
To exit full screen, press Esc
Aliases:
service, svc

Available Commands:
  clusterip    Create a ClusterIP service
  externalname Create an ExternalName service
  loadbalancer Create a LoadBalancer service
  nodeport     Create a NodePort service

Usage:
  kubectl create service [flags] [options]

Use "kubectl <command> --help" for more information about a given command.
Use "kubectl options" for a list of global command-line options (applies to all commands).

controlplane ~ ➜  kubectl create service clusterip
```

1 2 3 4 5 6 7 8 9

Create a service `redis-service` to expose the `redis` application within the cluster on port `6379`.

Use imperative commands.

Check

- Service: redis-service
- Port: 6379
- Type: ClusterIP

```
controlplane ~ ➜  kubectl expose redis --port 6379 --name redis-service
error: the server doesn't have a resource type "redis"
See 'kubectl expose -h' for help and examples

controlplane ~ ✖ kubectl expose pod redis --port 6379 --name redis-service
service/redis-service exposed

controlplane ~ ➜  kubectl get svc redis
Error from server (NotFound): services "redis" not found

controlplane ~ ✖ kubectl get svc redis-service
NAME            TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
redis-service   ClusterIP   10.43.56.187   <none>        6379/TCP   12s

controlplane ~ ➜  kubectl describe svc redis-service
Name:              redis-service
Namespace:         default
Labels:            tier=db
Annotations:       <none>
Selector:          tier=db
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.43.56.187
IPs:               10.43.56.187
Port:              <unset>  6379/TCP
TargetPort:        6379/TCP
Endpoints:         10.42.0.10:6379
Session Affinity:  None
Events:            <none>

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9

Create a deployment named `webapp` using the image `kodekloud/webapp-color` with `3` replicas.

Try to use imperative commands only. Do not create definition files.

Next

- Name: webapp
- Image: kodekloud/webapp-color
- Replicas: 3

```
controlplane ~ ➜  kubectl create deployment webapp image=kodekloud/webapp-color --replicas=3
Error: required flag(s) "image" not set
required flag(s) "image" not set

controlplane ~ ✖ kubectl create deployment webapp --image=kodekloud/webapp-color --replicas=3
deployment.apps/webapp created

controlplane ~ ➜

controlplane ~ ➜  kubctl get deploy
bash: kubctl: command not found

controlplane ~ ✖ kubctl get deployments
bash: kubctl: command not found

controlplane ~ ✖ kubectl get deploy
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
webapp   0/3     3            0           17s

controlplane ~ ➜  kubectl get deploy
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
webapp   3/3     3            3           23s

controlplane ~ ➜
```

kubectl create deployment webapp --image kodekloud/webapp-color --dry-run=client -o yaml > rohan

1 2 3 4 5 6 7 8 9

Create a new pod called `custom-nginx` using the `nginx` image and expose it on `container port 8080`.

Check

- Pod created correctly?

```
controlplane ~ ➜  kubectl run custom-nginx --image=nginx --port=8080
pod/custom-nginx created

controlplane ~ ➜
```

kubectl run custom-nginx --image nginx --port 8080 --dry-run=client -o yaml > shivam.yaml

1 2 3 4 5 6 7 8 9

Create a new namespace called `dev-ns`.
Use imperative commands.

Complete Next

- Namespace created?

```
controlplane ~ ➜  kubectl create namespace dev-ns
namespace/dev-ns created

controlplane ~ ➜
```

1 2 3 4 5 6 7 8 9

Create a new deployment called `redis-deploy` in the `dev-ns` namespace with the `redis` image. It should have `2` replicas.
Use imperative commands.

Check

- 'redis-deploy' created in the 'dev-ns' namespace?
- replicas: 2

```
controlplane ~ ➜  kubectl create deployment redis-deploy --image=redis --replicas=2 -n dev-ns
deployment.apps/redis-deploy created

controlplane ~ ➜  kubectl get deployment -n dev-ns
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
redis-deploy   2/2     2            2           12s

controlplane ~ ➜
```

kubectl create deployment redis-deploy -n dev-ns --image redis -r 5 --dry-run=client -o yaml > zoro.yaml

1 2 3 4 5 6 7 8 9

Create a pod called `httpd` using the image `httpd:alpine` in the default namespace. Next, create a service of type `ClusterIP` by the same name `(httpd)`. The target port for the service should be `80`.
Try to do this with as few steps as possible.

Check

- 'httpd' pod created with the correct image?
- 'httpd' service is of type 'ClusterIP'?
- 'httpd' service uses correct target port 80?
- 'httpd' service exposes the 'httpd' pod?

```
-o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
      --timeout=0s: The length of time to wait before giving up on a delete, zero means determine a
timeout from the size of the object
  -t, --tty=false: Allocate a TTY for the container in the pod.
      --wait=false: If true, wait for resources to be gone before returning. This waits for
finalizers.

Usage:
  kubectl run NAME --image=image [--env="key=value"] [--port=port] [--dry-run=server|client]
[--overrides=inline-json] [--command] -- [COMMAND] [args...] [options]

Use "kubectl options" for a list of global command-line options (applies to all commands).

controlplane ~ ➜  kubectl run httpd --image=httpd:alpine --port=80 --expose=true
service/httpd created
pod/httpd created

controlplane ~ ➜

controlplane ~ ➜  kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
nginx-pod                1/1     Running   0          12m
redis                    1/1     Running   0          10m
webapp-7b59bf687d-n7xxp  1/1     Running   0          5m4s
webapp-7b59bf687d-rds95  1/1     Running   0          5m4s
webapp-7b59bf687d-4gqmt  1/1     Running   0          5m4s
custom-nginx             1/1     Running   0          3m41s
httpd                    1/1     Running   0          8s

controlplane ~ ➜  kubectl get sv
error: the server doesn't have a resource type "sv"

controlplane ~ ✖ kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes      ClusterIP   10.43.0.1       <none>        443/TCP    20m
redis-service   ClusterIP   10.43.56.187    <none>        6379/TCP   6m35s
httpd           ClusterIP   10.43.112.233   <none>        80/TCP     15s

controlplane ~ ➜  kubectl describe
```

kubectl Apply

Thursday, June 6, 2024 10:19 AM

Kubectl Apply Command

Declarative

Create Objects

```
> kubectl apply -f nginx.yaml
```

```
> kubectl apply -f /path/to/config-files
```

Update Objects

```
> kubectl apply -f nginx.yaml
```

nginx.yaml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end-service
spec:
  containers:
  - name: nginx-container
    image: nginx:1.18
```

Kubectl Apply

Local file

nginx.yaml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end-service
spec:
  containers:
  - name: nginx-container
    image: nginx:1.19
```

```
> kubectl apply -f nginx.yaml
```

Last applied Configuration

```json
{
  "apiVersion": "v1",
  "kind": "Pod",
  "metadata": {
    "annotations": {},
    "labels": {
      "run": "myapp-pod",
      "type": "front-end-service"
    },
    "name": "myapp-pod",
  },
  "spec": {
    "containers": [
      {
        "image": "nginx:1.19",
        "name": "nginx-container"
      }
    ]
  }
}
```

Kubernetes

Live object configuration

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end-service
spec:
  containers:
  - name: nginx-container
    image: nginx:1.19
status:
  conditions:
  - lastProbeTime: null
    status: "True"
    type: Initialized
```

Kubectl Apply

Local file

nginx.yaml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end-service
spec:
  containers:
  - name: nginx-container
    image: nginx:1.18
```

```
> kubectl apply -f nginx.yaml
```

Last applied Configuration

Kubernetes

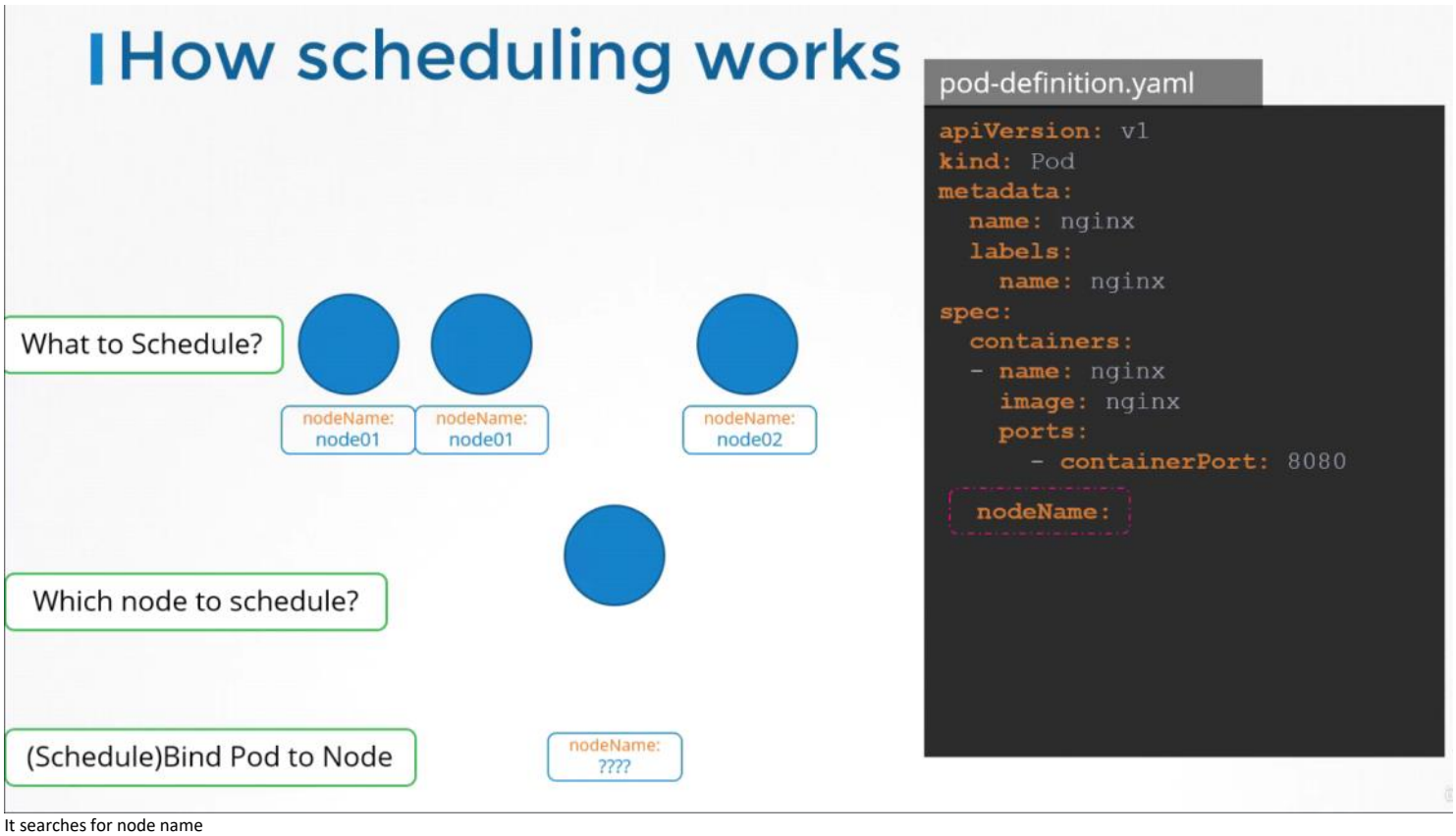
Live object configuration

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 annotations:
  kubectl.kubernetes.io/last-applied-configuration:
 {"apiVersion": "v1","kind": "Pod","metadata": {"annotations":
 {},"labels": {"run": "myapp-pod","type": "front-end-service"
    },"name": "myapp-pod"},"spec": {"containers": [{"image":
 "nginx:1.18","name": "nginx-container"}]}}
 labels:
    app: myapp
    type: front-end-service
spec:
  containers:
  - name: nginx-container
    image: nginx:1.18
status:
  conditions:
  - lastProbeTime: null
```

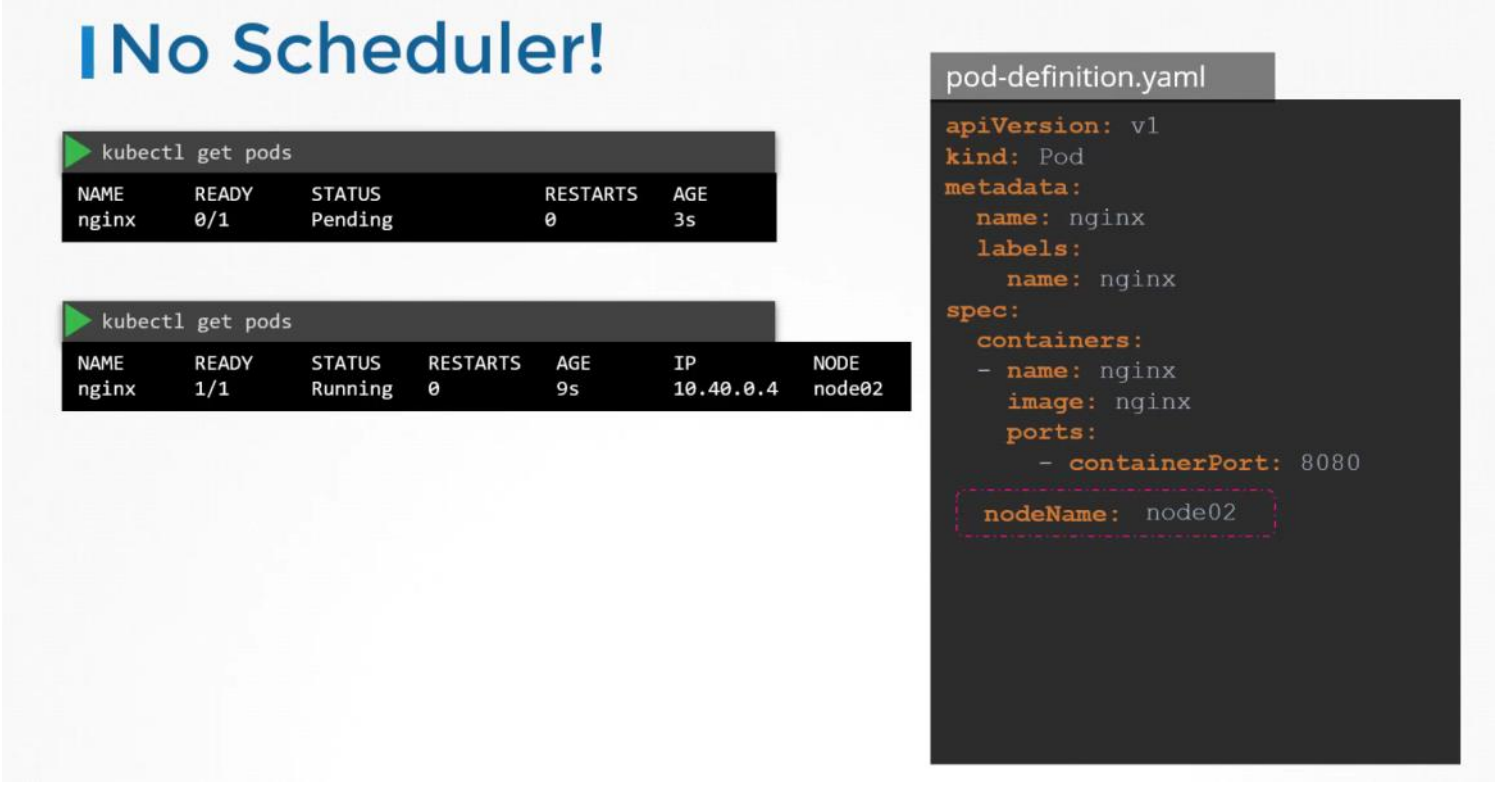
the live object configuration,

Scheduling

Friday, June 7, 2024 1:13 PM



It searches for node name



No Scheduler!

Pod-bind-definition.yaml

```yaml
apiVersion: v1
kind: Binding
metadata:
  name: nginx
target:
  apiVersion: v1
  kind: Node
  name: node02
```

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
      - containerPort: 8080
```

```
curl --header "Content-Type:application/json" --request POST --data '{"apiVersion":"v1", "kind": "Binding" …. }
http://$SERVER/api/v1/namespaces/default/pods/$PODNAME/binding/
```

01 02 03 04 05

A pod definition file `nginx.yaml` is given.
Create a pod using the file.

Only create the POD for now. We will inspect its status next.

...

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get nodes
NAME           STATUS   ROLES                  AGE   VERSION
controlplane   Ready    control-plane,master   14m   v1.20.0
node01         Ready    <none>                 14m   v1.20.0
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# ls
nginx.yaml  sample.yaml
root@controlplane:~# cat nginx.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  -  image: nginx
     name: nginx

root@controlplane:~#
root@controlplane:~# kubectl create -f nginx.yaml
pod/nginx created
root@controlplane:~#
root@controlplane:~#
```

What is the status of the created POD?

Scheduled

Error

Running

Pending

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# ls
nginx.yaml  sample.yaml
root@controlplane:~# cat nginx.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  -  image: nginx
     name: nginx

root@controlplane:~#
root@controlplane:~# kubectl create -f nginx.yaml
pod/nginx created
root@controlplane:~#
root@controlplane:~# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Pending   0          12s
root@controlplane:~#
```

Why is the POD in a pending state?

Inspect the environment for various kubernetes control plane components.

No nodes available

Incorrect Label

Taints & Tolerations

No Scheduler Present

```
Status:       Pending
IP:
IPs:          <none>
Containers:
  nginx:
    Image:        nginx
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-kpdm6 (ro)
Volumes:
  default-token-kpdm6:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-kpdm6
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:          <none>
root@controlplane:~# kubectl get pods -n kube-system
NAME                                   READY   STATUS    RESTARTS   AGE
coredns-74ff55c5b-csz5x                1/1     Running   0          16m
coredns-74ff55c5b-vzlnv                1/1     Running   0          16m
etcd-controlplane                      1/1     Running   0          16m
kube-apiserver-controlplane            1/1     Running   0          16m
kube-controller-manager-controlplane   1/1     Running   0          16m
kube-flannel-ds-9pcnm                  1/1     Running   0          16m
kube-flannel-ds-nzbns                  1/1     Running   0          16m
kube-proxy-2xblf                       1/1     Running   0          16m
kube-proxy-g7f5r                       1/1     Running   0          16m
root@controlplane:~#
```

if you cat nginx.yaml you will see no node name is given . so it wa	is not scheduled and was in pending state. You can schedule a pod on a node after creation of pod only by pod binding definition.yaml

01 02 03 04 05

Manually schedule the pod on `node01`.

Delete and recreate the POD if necessary.

Check

- Status: Running
- Pod: nginx

```
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeName: node01
  containers:
  - image: nginx
    name: nginx

~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

01 02 03 04 05

Manually schedule the pod on `node01`.

Delete and recreate the POD if necessary.

Check

- Status: Running
- Pod: nginx

```
root@controlplane:~# vi nginx.yaml
root@controlplane:~# kubectl replace --force -f nginx.yaml
pod "nginx" deleted
pod/nginx replaced
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get pods
NAME    READY   STATUS              RESTARTS   AGE
nginx   0/1     ContainerCreating   0          8s
root@controlplane:~# kubectl get pods
NAME    READY   STATUS              RESTARTS   AGE
nginx   0/1     ContainerCreating   0          10s
root@controlplane:~# kubectl get pods
NAME    READY   STATUS              RESTARTS   AGE
nginx   0/1     ContainerCreating   0          12s
root@controlplane:~# kubectl get pods --watch
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          19s
^Croot@controlplane:~#
root@controlplane:~#
```

01 02 03 04 05

Now schedule the same pod on the `controplane` node.

Delete and recreate the POD if necessary.

Check

- Status: Running
- Pod: nginx
- Node: controlplane?

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeName: controlplane
  containers:
  - image: nginx
    name: nginx

~
~
~
~
~
~
~
~
~
~
~
~
~
```

Now schedule the same pod on the `controplane` node.

Delete and recreate the POD if necessary.

Check

- Status: Running
- Pod: nginx
- Node: controlplane?

```
root@controlplane:~# kubectl get pods
NAME    READY   STATUS              RESTARTS   AGE
nginx   0/1     ContainerCreating   0          8s
root@controlplane:~# kubectl get pods
NAME    READY   STATUS              RESTARTS   AGE
nginx   0/1     ContainerCreating   0          10s
root@controlplane:~# kubectl get pods
NAME    READY   STATUS              RESTARTS   AGE
nginx   0/1     ContainerCreating   0          12s
root@controlplane:~# kubectl get pods --watch
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          19s
^Croot@controlplane:~#
root@controlplane:~#
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# vi nginx.yaml
root@controlplane:~#
root@controlplane:~# kubectl replace --force -f nginx.yaml
pod "nginx" deleted
pod/nginx replaced
root@controlplane:~#
root@controlplane:~# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          54s
root@controlplane:~# kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP          NODE          NOMINATED NODE   READINESS GATES
nginx   1/1     Running   0          58s   10.244.0.4   controlplane   <none>           <none>
root@controlplane:~#
```

Labels and Selectors

Friday, June 7, 2024 2:26 PM

Labels & Selectors

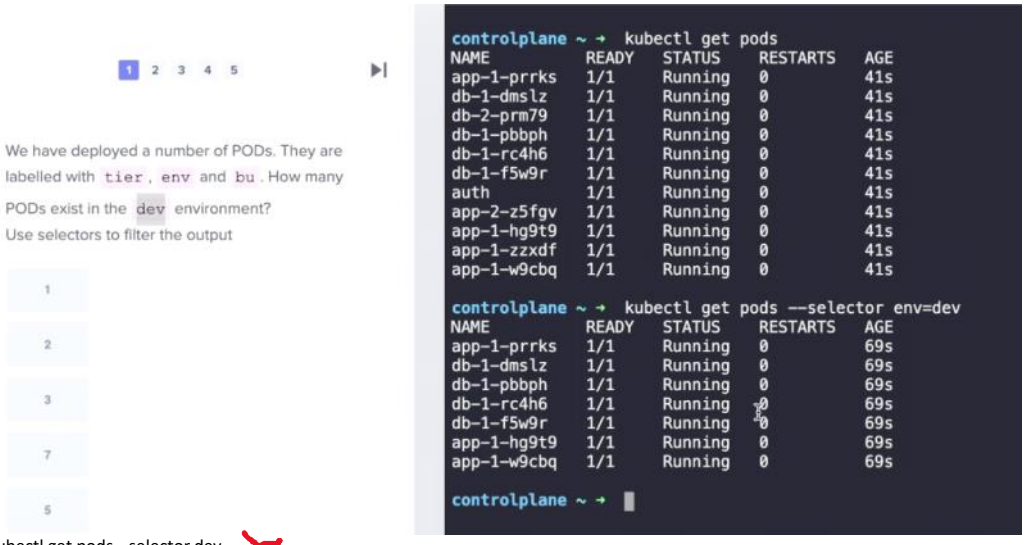
Labels

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp
  labels:
      app: App1
      function: Front-end


spec:
  containers:
  - name: simple-webapp
    image: simple-webapp
    ports:
      - containerPort: 8080
```

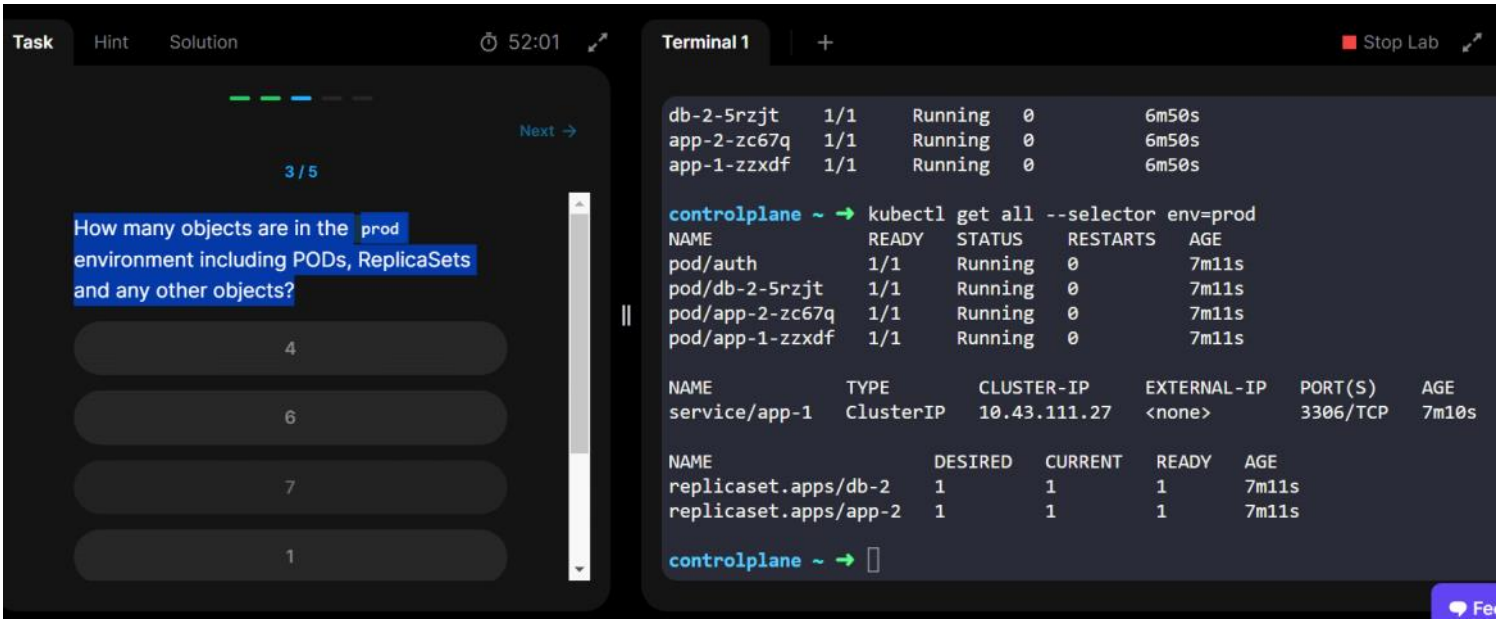
Select

```
kubectl get pods --selector app=App1
NAME                READY     STATUS      RESTARTS   AGE
simple-webapp        0/1    Completed   0          1d
```



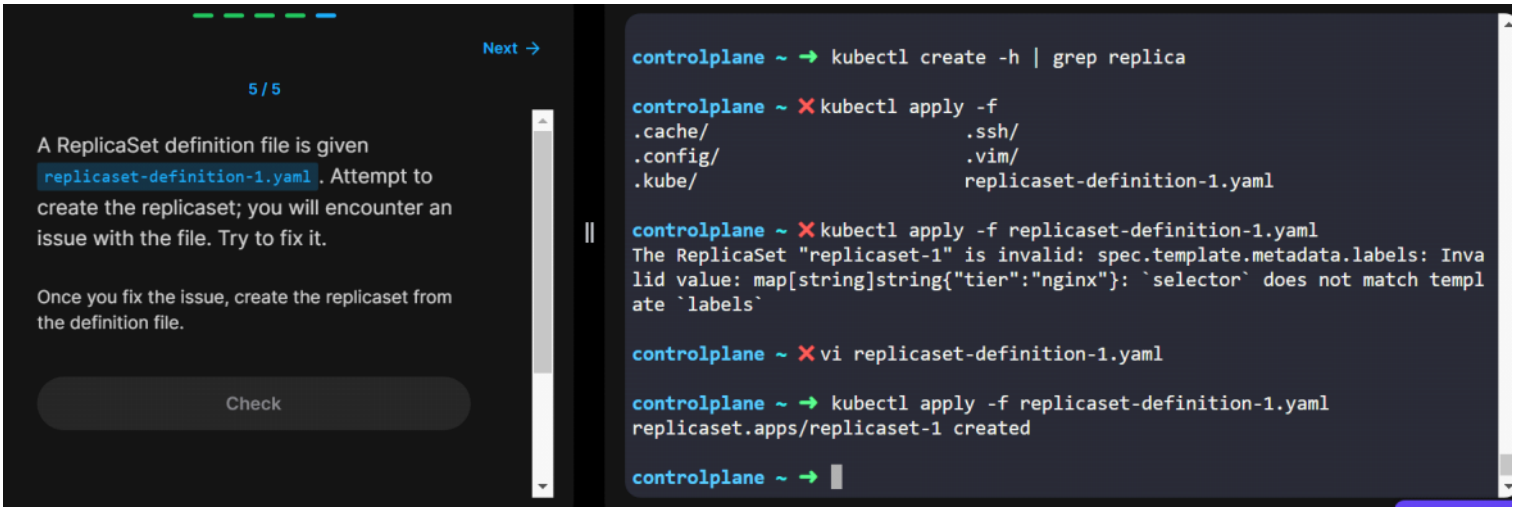
kubectl get pods --selector dev

- how many pods are in the dev env ?
  kubectl get pods --selector env=dev --no-headers | wc -l
  kubectl get pods -l env=prod
- kubectl get pods --selector bu=finance | wc -l



kubectl get all --selector env=prod,bu=finance,tier=frontend

kubectl get all -l env=prod,bu=finance,tier=frontend



kubectl apply -f replicaset-definition-1.yaml

Selector Practice

Friday, June 7, 2024 5:13 PM

1 2 3 4 5

We have deployed a number of PODs. They are labelled with `tier`, `env` and `bu`. How many PODs exist in the `dev` environment? Use selectors to filter the output

1

2

3

7

5

```
controlplane ~ ➜  kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
app-1-prrks   1/1     Running   0          41s
db-1-dmslz    1/1     Running   0          41s
db-2-prm79    1/1     Running   0          41s
db-1-pbbph    1/1     Running   0          41s
db-1-rc4h6    1/1     Running   0          41s
db-1-f5w9r    1/1     Running   0          41s
auth          1/1     Running   0          41s
app-2-z5fgv   1/1     Running   0          41s
app-1-hg9t9   1/1     Running   0          41s
app-1-zzxdf   1/1     Running   0          41s
app-1-w9cbq   1/1     Running   0          41s

controlplane ~ ➜  kubectl get pods --selector env=dev
NAME          READY   STATUS    RESTARTS   AGE
app-1-prrks   1/1     Running   0          69s
db-1-dmslz    1/1     Running   0          69s
db-1-pbbph    1/1     Running   0          69s
db-1-rc4h6    1/1     Running   0          69s
db-1-f5w9r    1/1     Running   0          69s
app-1-hg9t9   1/1     Running   0          69s
app-1-w9cbq   1/1     Running   0          69s

controlplane ~ ➜  kubectl get pods --selector env=dev | wc -l
8

controlplane ~ ➜  kubectl get pods --selector env=dev --no-headers
app-1-prrks   1/1   Running   0     110s
db-1-dmslz    1/1   Running   0     110s
db-1-pbbph    1/1   Running   0     110s
db-1-rc4h6    1/1   Running   0     110s
db-1-f5w9r    1/1   Running   0     110s
app-1-hg9t9   1/1   Running   0     110s
app-1-w9cbq   1/1   Running   0     110s

controlplane ~ ➜  kubectl get pods --selector env=dev --no-headers | wc -l
7

controlplane ~ ➜
```

1 2 3 4 5

How many PODs are in the `finance` business unit (`bu`)?

1

2

7

5

```
controlplane ~ ➜  kubectl get pods --selector bu=finance --no-headers | wc -l
6

controlplane ~ ➜
```

1 2 3 4 5

How many objects are in the `prod` environment including PODs, ReplicaSets and any other objects?

1

2

6

4

5

7

```
controlplane ~ ➜  kubectl get pods --selector env=prod --no-headers
db-2-prm79    1/1   Running   0     3m1s
auth          1/1   Running   0     3m1s
app-2-z5fgv   1/1   Running   0     3m1s
app-1-zzxdf   1/1   Running   0     3m1s

controlplane ~ ➜  kubectl get all --selector env=prod --no-headers
pod/db-2-prm79    1/1   Running   0     3m13s
pod/auth          1/1   Running   0     3m13s
pod/app-2-z5fgv   1/1   Running   0     3m13s
pod/app-1-zzxdf   1/1   Running   0     3m13s
service/app-1   ClusterIP   10.43.108.231   <none>   3306/TCP   3m12s
replicaset.apps/db-2    1     1     1     3m13s
replicaset.apps/app-2   1     1     1     3m13s

controlplane ~ ➜
```

Identify the POD which is part of the `prod` environment, the `finance` BU and of `frontend` tier?

app-1-zzxdf

auth

app-1-zk988

app-2-8zn86

db-2-c4qnn

[redacted]

```
controlplane ~ ➜  kubectl get all --selector env=prod,bu=finance,tier=frontend
NAME              READY   STATUS    RESTARTS   AGE
pod/app-1-zzxdf   1/1     Running   0          4m7s

controlplane ~ ➜
```

1 2 3 4 5

A ReplicaSet definition file is given `replicaset-definition-1.yaml`. Try to create the replicaset. There is an issue with the file. Try to fix it.

Check

- ReplicaSet: replicaset-1
- Replicas: 2

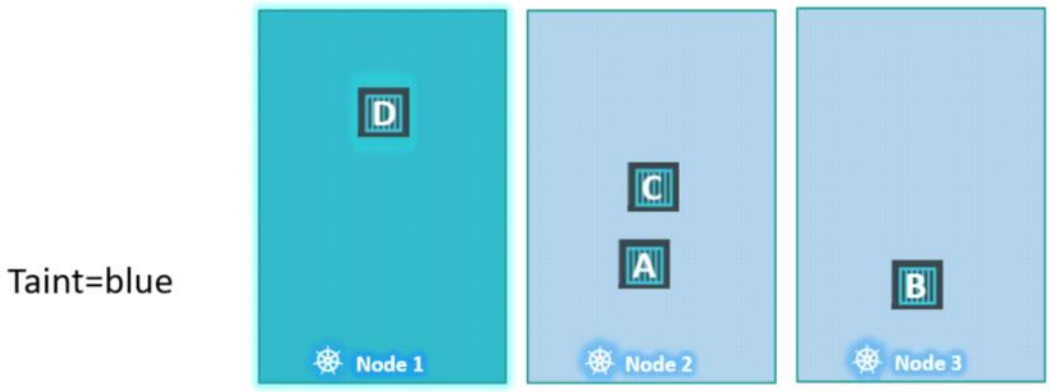
```
controlplane ~ ➜  cat replicaset-definition-1.yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
   name: replicaset-1
spec:
   replicas: 2
   selector:
      matchLabels:
        tier: front-end
   template:
     metadata:
       labels:
        tier: nginx
     spec:
       containers:
       - name: nginx
         image: nginx

controlplane ~ ➜  kubectl create -f replicaset-definition-1.yaml
The ReplicaSet "replicaset-1" is invalid: spec.template.metadata.labels: Invalid value: map[string]strin
g{"tier":"nginx"}: `selector` does not match template `labels`

controlplane ~ ✖ vi replicaset-definition-1.yaml
```

```
controlplane ~ ➜  kubectl create -f replicaset-definition-1.yaml
The ReplicaSet "replicaset-1" is invalid: spec.template.metadata.labels: Invalid value: map[string]strin
g{"tier":"nginx"}: `selector` does not match template `labels`

controlplane ~ ✖ vi replicaset-definition-1.yaml

controlplane ~ ➜  kubectl create -f replicaset-definition-1.yaml
replicaset.apps/replicaset-1 created

controlplane ~ ➜  kubectl get rs
NAME           DESIRED   CURRENT   READY   AGE
db-2           1         1         1       5m39s
db-1           4         4         4       5m39s
app-2          1         1         1       5m39s
app-1          3         3         3       5m40s
replicaset-1   2         2         0       10s

controlplane ~ ➜
```

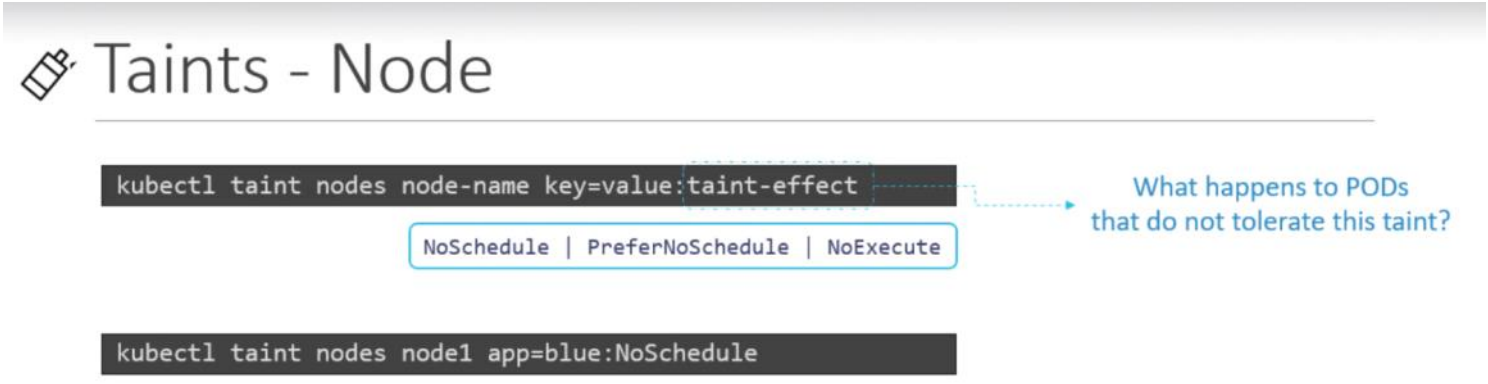
Taints and Tolerations

Friday, June 7, 2024 5:21 PM

Taints are set on node and tolerations are set on POD's



a



noschedule will not schedule pods on the node
prefer no schedule will not schedule pods on node but its not guaranteed.
no execute will not schedule pods on the node as well as it will evict any pods running on the node.

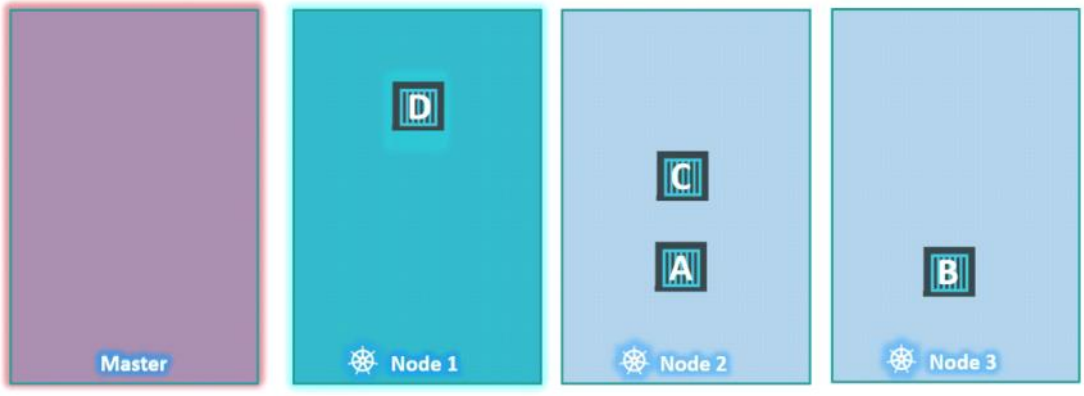
Tolerations - PODs

```
kubectl taint nodes node1          :
```

pod-definition.yml

```yml
apiVersion:
kind: Pod
metadata:
 name: myapp-pod
spec:
  containers:
  - name: nginx-container
    image: nginx

  tolerations:
  - key:"app"
    operator:"Equal"
    value:" blue"
    effect:" NoSchedule"
```

```
kubectl describe node kubemaster | grep Taint
Taints:             node-role.kubernetes.io/master:NoSchedule
```



Practice Taints and tolerations

Sunday, June 9, 2024 10:30 AM

01 02 03 04 05 06 07 08 09 10 11 12

How many `nodes` exist on the system?

Including the `controlplane` node.

```
root@controlplane:~# kubectl get nodes
NAME           STATUS   ROLES                  AGE   VERSION
controlplane   Ready    control-plane,master   17m   v1.20.0
node01         Ready    <none>                 16m   v1.20.0
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12

Do any taints exist on `node01` node?

Yes - NoExecute

No

Yes - NoSchedule

```
NAME           STATUS   ROLES                  AGE   VERSION
controlplane   Ready    control-plane,master   17m   v1.20.0
node01         Ready    <none>                 16m   v1.20.0
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl describe node node01
Name:               node01
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=node01
                    kubernetes.io/os=linux
Annotations:        flannel.alpha.coreos.com/backend-data: {"VNI":1,"VtepMAC":"8e:62:74:26:35:47"}
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 10.58.27.11
                    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:  Fri, 15 Apr 2022 22:57:19 +0000
Taints:             <none>
Unschedulable:      false
Lease:
  HolderIdentity:  node01
  AcquireTime:     <unset>
  RenewTime:       Fri, 15 Apr 2022 23:14:20 +0000
Conditions:
  Type                 Status  LastHeartbeatTime                 LastTransitionTime                Reaso
n                       Message
  ----                 ------  -----------------                 ------------------                -----
-                       -------
  NetworkUnavailable   False   Fri, 15 Apr 2022 22:57:25 +0000   Fri, 15 Apr 2022 22:57:25 +0000   Flann
elIsUp                  Flannel is running on this node
  MemoryPressure       False   Fri, 15 Apr 2022 23:12:35 +0000   Fri, 15 Apr 2022 22:57:19 +0000   Kubel
etHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure         False   Fri, 15 Apr 2022 23:12:35 +0000   Fri, 15 Apr 2022 22:57:19 +0000   Kubel
etHasNoDiskPressure     kubelet has no disk pressure
```

01 02 03 04 05 06 07 08 09 10 11 12

Create a taint on `node01` with key of `spray`, value of `mortein` and effect of `NoSchedule`

Check

- Key = spray
- Value = mortein
- Effect = NoSchedule

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl taint --he
```

01 02 03 04 05 06 07 08 09 10 11 12

Create a taint on `node01` with key of `spray`, value of `mortein` and effect of `NoSchedule`

Check

- Key = spray
- Value = mortein
- Effect = NoSchedule

```
root@controlplane:~#
root@controlplane:~# kubectl taint node node01 spray=mortein:NoSchedule
node/node01 tainted
root@controlplane:~#
root@controlplane:~# kubectl describe node node01
Name:               node01
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=node01
                    kubernetes.io/os=linux
Annotations:        flannel.alpha.coreos.com/backend-data: {"VNI":1,"VtepMAC":"8e:62:74:
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 10.58.27.11
                    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:  Fri, 15 Apr 2022 22:57:19 +0000
Taints:             spray=mortein:NoSchedule
Unschedulable:      false
Lease:
  HolderIdentity:  node01
  AcquireTime:     <unset>
  RenewTime:       Fri, 15 Apr 2022 22:16:10 +0000
```

kubectl taint nodes node01 spray=mortein:NoSchedule
kubectl describe node node01 | grep -i taint

01 02 03 04 05 06 07 08 09 10 11 12

What is the state of the POD?

Error

Ready

Pending

```
root@controlplane:~# kubectl run mosquito --image=nginx
pod/mosquito created
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
mosquito   0/1     Pending   0          6s
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12

Why do you think the pod is in a pending state?

Image is not available

Application Error

POD Mosquito cannot tolerate taint Mortein

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl describe pod mosquito
Name:         mosquito
Namespace:    default
Priority:     0
Node:         <none>
Labels:       run=mosquito
Annotations:  <none>
Status:       Pending
IP:
IPs:          <none>
Containers:
  mosquito:
    Image:        nginx
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-f6lxf (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  default-token-f6lxf:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-f6lxf
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age                From               Message
  ----     ------            ----               ----               -------
  Warning  FailedScheduling  45s (x2 over 45s)  default-scheduler  0/2 nodes are available: 1 node(s) ha
d taint {node-role.kubernetes.io/master: }, that the pod didn't tolerate, 1 node(s) had taint {spray: mo
rtein}, that the pod didn't tolerate.
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12

Create a new pod with the `nginx` image and pod name as `mosquito`.

Check

```
root@controlplane:~# kubectl run mosquito --image=nginx
pod/mosquito created
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
mosquito   0/1     Pending   0          6s
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12

Create another pod named `bee` with the `nginx` image, which has a toleration set to the

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl run bee --image=nginx --dry-run=client -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: bee
  name: bee
```

01 02 03 04 05 06 07 08 09 10 11 12

Create another pod named `bee` with the `nginx` image, which has a toleration set to the taint `mortein`.

Check

- Image name: nginx
- Key: spray
- Value: mortein
- Effect: NoSchedule
- Status: Running

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl run bee --image=nginx --dry-run=client -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: bee
  name: bee
spec:
  containers:
  - image: nginx
    name: bee
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
root@controlplane:~# kubectl run bee --image=nginx --dry-run=client -o yaml > bee.yaml
root@controlplane:~#
root@controlplane:~# cat bee.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: bee
  name: bee
spec:
  containers:
  - image: nginx
    name: bee
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
root@controlplane:~# vi be
```

01 02 03 04 05 06 07 08 09 10 11 12

Create another pod named `bee` with the `nginx` image, which has a toleration set to the taint `mortein`.

Check

- Image name: nginx
- Key: spray
- Value: mortein
- Effect: NoSchedule
- Status: Running

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: bee
  name: bee
spec:
  containers:
  - image: nginx
    name: bee
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  tolerations:
          - key: spray
            value: mortein
            effect: NoSchedule
            operator: Equal
status: {}
~
~
~
~
~
~
~
~
~
~
```

```
  restartPolicy: Always
status: {}
root@controlplane:~# vi bee.yaml
root@controlplane:~# kubectl create -f bee.yaml
```

```
root@controlplane:~# kubectl get pods
NAME        READY   STATUS              RESTARTS   AGE
bee         0/1     ContainerCreating   0          7s
mosquito    0/1     Pending             0          3m37s
root@controlplane:~# kubectl get pods
NAME        READY   STATUS              RESTARTS   AGE
bee         0/1     ContainerCreating   0          10s
mosquito    0/1     Pending             0          3m40s
root@controlplane:~# kubectl get pods --watch
NAME        READY   STATUS              RESTARTS   AGE
bee         0/1     ContainerCreating   0          13s
mosquito    0/1     Pending             0          3m43s
bee         1/1     Running             0          14s
^Croot@controlplane:~#
root@controlplane:~#
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10 11 12

Remove the taint on `controlplane`, which currently has the taint effect of `NoSchedule`.

Check

Node name: controlplane

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl describe node controlplane
Name:               controlplane
Roles:              control-plane,master
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=controlplane
                    kubernetes.io/os=linux
                    node-role.kubernetes.io/control-plane=
                    node-role.kubernetes.io/master=
Annotations:        flannel.alpha.coreos.com/backend-data: {"VNI":1,"VtepMAC":"f6:6e:ba:7d:23:ca"}
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 10.58.27.9
                    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:  Fri, 15 Apr 2022 22:56:44 +0000
Taints:             node-role.kubernetes.io/master:NoSchedule
Unschedulable:      false
Lease:
  HolderIdentity:  controlplane
  AcquireTime:     <unset>
  RenewTime:       Fri, 15 Apr 2022 23:21:49 +0000
Conditions:
  Type                 Status  LastHeartbeatTime                 LastTransitionTime                Reaso
n                       Message
  ----                 ------  -----------------                 ------------------                -----
-                       -------
  NetworkUnavailable   False   Fri, 15 Apr 2022 22:57:09 +0000   Fri, 15 Apr 2022 22:57:09 +0000   Flann
elIsUp                  Flannel is running on this node
  MemoryPressure       False   Fri, 15 Apr 2022 23:17:23 +0000   Fri, 15 Apr 2022 22:56:33 +0000   Kubel
etHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure         False   Fri, 15 Apr 2022 23:17:23 +0000   Fri, 15 Apr 2022 22:56:33 +0000   Kubel
etHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure          False   Fri, 15 Apr 2022 23:17:23 +0000   Fri, 15 Apr 2022 22:56:33 +0000   Kubel
etHasSufficientPID      kubelet has sufficient PID available
```

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl taint node controlplane node-role.kubernetes.io/master:NoSchedule-
node/controlplane untainted
root@controlplane:~#
```

kubectl taint node controlplane node-role.kubernetes.io/control-plane:NoSchedule-

01 02 03 04 05 06 07 08 09 10 11 12

Which node is the POD `mosquito` on now?

node02

controlplane

node01

```
root@controlplane:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
bee        1/1     Running   0          2m37s
mosquito   1/1     Running   0          6m7s
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE     IP          NODE          NOMINATED NODE   READINESS G
ATES
bee        1/1     Running   0          2m52s   10.244.1.2  node01        <none>           <none>
mosquito   1/1     Running   0          6m22s   10.244.0.4  controlplane  <none>           <none>
root@controlplane:~#
```

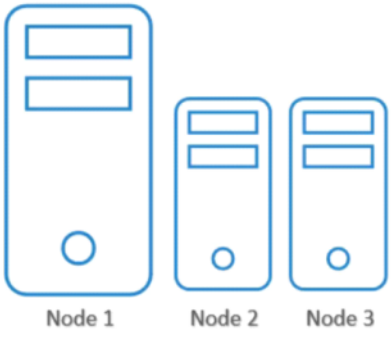
Node Selector

Tuesday, June 11, 2024 8:21 AM

Node Selectors

pod-definition.yml

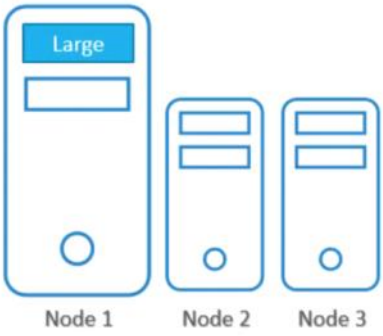
```yaml
apiVersion:
kind: Pod
metadata:
 name: myapp-pod
spec:
  containers:
  - name: data-processor
    image: data-processor

  nodeSelector:
    size: Large
```



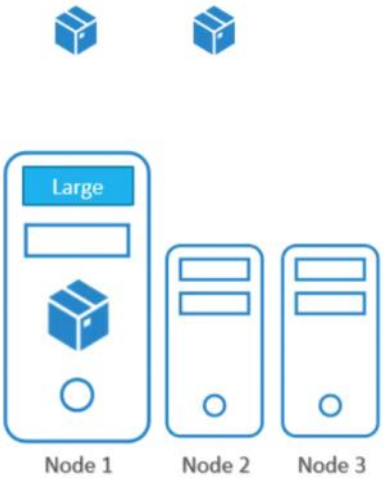
Label Nodes

```
kubectl label nodes <node-name> <label-key>=<label-value>
```

```
kubectl label nodes node-1 size=Large
```



Node Selector



pod-definition.yml

```yml
apiVersion:
kind: Pod
metadata:
 name: myapp-pod
spec:
  containers:
  - name: data-processor
    image: data-processor

  nodeSelector:
    size: Large
```

```
kubectl create -f pod-definition.yml
```

Node Affinity

Tuesday, June 11, 2024 8:43 AM

Node Affinity

pod-definition.yml

```yaml
apiVersion:
kind: Pod

metadata:
 name: myapp-pod
spec:

  containers:
  - name: data-processor
    image: data-processor

  nodeSelector:
    size: Large
```

pod-definition.yml

```yaml
apiVersion:
kind:

metadata:
 name: myapp-pod
spec:

 containers:
  - name: data-processor
    image: data-processor

 affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: size
          operator: In
          values:
          - Large
```

Node Affinity

pod-definition.yml

```yaml
apiVersion:
kind:

metadata:
 name: myapp-pod
spec:

 containers:
  - name: data-processor
    image: data-processor

 affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: size
          operator: NotIn
          values:
          - Small
```

Node Affinity

pod-definition.yml

```yaml
apiVersion:
kind:
metadata:
 name: myapp-pod
spec:
 containers:
  - name: data-processor
    image: data-processor
 affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: size
          operator: Exists
```

Node Affinity Types

Available:

requiredDuringSchedulingIgnoredDuringExecution

preferredDuringSchedulingIgnoredDuringExecution

Planned:

requiredDuringSchedulingRequiredDuringExecution

Node Affinity Types

Available:

requiredDuringSchedulingIgnoredDuringExecution

preferredDuringSchedulingIgnoredDuringExecution

	DuringScheduling	DuringExecution
Type 1	Required	Ignored
Type 2	Preferred	Ignored

Practice- Node Affinity

Tuesday, June 11, 2024 8:56 AM

01 02 03 04 05 06 07 08

How many Labels exist on node node01?

5

```
root@controlplane:~# kubectl describe node node01
Name:               node01
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=node01
                    kubernetes.io/os=linux
Annotations:        flannel.alpha.coreos.com/backend-data: {"VNI":1,"VtepMAC":"5a:c6:da:f2:19:f9"}
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 10.35.54.12
                    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:  Sat, 16 Apr 2022 17:27:31 +0000
Taints:             <none>
Unschedulable:      false
Lease:
  HolderIdentity:  node01
  AcquireTime:     <unset>
  RenewTime:       Sat, 16 Apr 2022 17:39:41 +0000
Conditions:
  Type                 Status  LastHeartbeatTime                 LastTransitionTime                Reaso
n                       Message
  ----                 ------  -----------------                 ------------------                -----
-                       -------
  NetworkUnavailable   False   Sat, 16 Apr 2022 17:27:38 +0000   Sat, 16 Apr 2022 17:27:38 +0000   Flann
elIsUp                  Flannel is running on this node
  MemoryPressure       False   Sat, 16 Apr 2022 17:37:46 +0000   Sat, 16 Apr 2022 17:27:31 +0000   Kubel
etHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure         False   Sat, 16 Apr 2022 17:37:46 +0000   Sat, 16 Apr 2022 17:27:31 +0000   Kubel
etHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure          False   Sat, 16 Apr 2022 17:37:46 +0000   Sat, 16 Apr 2022 17:27:31 +0000   Kubel
etHasSufficientPID      kubelet has sufficient PID available
  Ready                True    Sat, 16 Apr 2022 17:37:46 +0000   Sat, 16 Apr 2022 17:27:42 +0000   Kubel
etReady                 kubelet is posting ready status
Addresses:
  InternalIP:  10.35.54.12
```

01 02 03 04 05 06 07 08

What is the value set to the label `beta.kubernetes.io/arch` on `node01` ?

amd64

```
root@controlplane:~# kubectl describe node node01
Name:               node01
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=node01
                    kubernetes.io/os=linux
Annotations:        flannel.alpha.coreos.com/backend-data: {"VNI":1,"VtepMAC":"5a:c6:da:f2:19:f9"}
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 10.35.54.12
                    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:  Sat, 16 Apr 2022 17:27:31 +0000
Taints:             <none>
Unschedulable:      false
Lease:
  HolderIdentity:  node01
  AcquireTime:     <unset>
  RenewTime:       Sat, 16 Apr 2022 17:39:41 +0000
Conditions:
  Type                 Status  LastHeartbeatTime                 LastTransitionTime                Rea
n                       Message
  ----                 ------  -----------------                 ------------------                ---
-                       -------
  NetworkUnavailable   False   Sat, 16 Apr 2022 17:27:38 +0000   Sat, 16 Apr 2022 17:27:38 +0000   Fla
elIsUp                  Flannel is running on this node
  MemoryPressure       False   Sat, 16 Apr 2022 17:37:46 +0000   Sat, 16 Apr 2022 17:27:31 +0000   Kub
etHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure         False   Sat, 16 Apr 2022 17:37:46 +0000   Sat, 16 Apr 2022 17:27:31 +0000   Kub
etHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure          False   Sat, 16 Apr 2022 17:37:46 +0000   Sat, 16 Apr 2022 17:27:31 +0000   Kub
etHasSufficientPID      kubelet has sufficient PID available
  Ready                True    Sat, 16 Apr 2022 17:37:46 +0000   Sat, 16 Apr 2022 17:27:42 +0000   Kub
etReady                 kubelet is posting ready status
Addresses:
  InternalIP:  10.35.54.12
```

01 02 03 04 05 06 07 08

Apply a label `color=blue` to node `node01`

Check

color = blue

```
      --field-manager='kubectl-label': Name of the manager used to track field ownership.
      --field-selector='': Selector (field query) to filter on, supports '=', '==', and '!='.(e.g.
--field-selector key1=value1,key2=value2). The server only supports a limited number of field
queries per type.
  -f, --filename=[]: Filename, directory, or URL to files identifying the resource to update the
labels
  -k, --kustomize='': Process the kustomization directory. This flag can't be used together with -f
or -R.
      --list=false: If true, display the labels for a given resource.
      --local=false: If true, label will NOT contact api-server but run locally.
  -o, --output='': Output format. One of:
json|yaml|name|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-fi
e.
      --overwrite=false: If true, allow labels to be overwritten, otherwise reject label updates
that overwrite existing labels.
      --record=false: Record current kubectl command in the resource annotation. If set to false, do
not record the command. If set to true, record the command. If not set, default to updating the
existing annotation value only if one already exists.
  -R, --recursive=false: Process the directory used in -f, --filename recursively. Useful when you
want to manage related manifests organized within the same directory.
      --resource-version='': If non-empty, the labels update will only succeed if this is the
current resource-version for the object. Only valid when specifying a single resource.
  -l, --selector='': Selector (label query) to filter on, not including uninitialized ones, supports
'=', '==', and '!='.(e.g. -l key1=value1,key2=value2).
      --template='': Template string or path to template file to use when -o=go-template,
-o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].

Usage:
  kubectl label [--overwrite] (-f FILENAME | TYPE NAME) KEY_1=VAL_1 ... KEY_N=VAL_N
[--resource-version=version] [options]

Use "kubectl options" for a list of global command-line options (applies to all commands).
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl label node node01 color=blue
node/node01 labeled
root@controlplane:~#
root@controlplane:~# kubectl describe
```

01 02 03 04 05 06 07 08

Create a new deployment named `blue` with the `nginx` image and 3 replicas.

Next

```
root@controlplane:~# kubectl create deployment blue --image=nginx --replicas=3
deployment.apps/blue created
root@controlplane:~#
root@controlplane:~#
```

01 02 03 04 05 06 07 08

Which nodes `can` the pods for the `blue` deployment be placed on?

Make sure to check taints on both nodes!

controlplane and node01

node01

controlplane

```
root@controlplane:~# kubectl describe node node01 | grep Taints
Taints:             <none>
root@controlplane:~# kubectl get nodes
NAME           STATUS   ROLES                  AGE   VERSION
controlplane   Ready    control-plane,master   15m   v1.20.0
node01         Ready    <none>                 14m   v1.20.0
root@controlplane:~# kubectl describe node controlplane | grep Taints
Taints:             <none>
```

```
    metadata:
      creationTimestamp: null
      labels:
        app: blue
    spec:
      affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
          - matchExpressions:
            - key: disktype
              operator: In
              values:
              - ssd
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
```

to correct indentation press V and select all lines and then press shift + >

01 02 03 04 05 06 07 08

Set Node Affinity to the deployment to place the pods on `node01` only.

Check

- Name: blue
- Replicas: 3
- Image: nginx
- NodeAffinity: requiredDuringSchedulingIgnoredDuringExecutior
- Key: color
- values: blue

```
root@controlplane:~# kubectl edit deployment blue
deployment.apps/blue edited
root@controlplane:~#
```

01 02 03 04 05 06 07 08

Which nodes are the pods placed on now?

node01

controlplane and node01

controlplane

Master

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get pods -o wide
NAME                    READY   STATUS        RESTARTS   AGE    IP          NODE     NOMINATED NODE   R
EADINESS GATES
blue-566c768bd6-f8xzm   1/1     Running       0          16s    10.244.1.5  node01   <none>           <
none>
blue-566c768bd6-jzs95   1/1     Running       0          9s     10.244.1.7  node01   <none>           <
none>
blue-566c768bd6-sf9dk   1/1     Running       0          13s    10.244.1.6  node01   <none>           <
none>
blue-7bb46df96d-8x4px   1/1     Terminating   0          6m4s   10.244.1.2  node01   <none>           <
none>
root@controlplane:~# kubectl get pods -o wide
NAME                    READY   STATUS        RESTARTS   AGE     IP          NODE     NOMINATED NODE
READINESS GATES
blue-566c768bd6-f8xzm   1/1     Running       0          23s     10.244.1.5  node01   <none>
<none>
blue-566c768bd6-jzs95   1/1     Running       0          16s     10.244.1.7  node01   <none>
<none>
blue-566c768bd6-sf9dk   1/1     Running       0          20s     10.244.1.6  node01   <none>
<none>
blue-7bb46df96d-8x4px   0/1     Terminating   0          6m11s   <none>      node01   <none>
<none>
root@controlplane:~#
```

Task Solution 48:37

01 02 03 04 05 06 07 08

Create a new deployment named `red` with the `nginx` image and `2` replicas, and ensure it gets placed on the `controlplane` node only.

Use the label - `node-role.kubernetes.io/master` - set on the controlplane node.

Check

- Name: red
- Replicas: 2
- Image: nginx
- NodeAffinity: requiredDuringSchedulingIgnoredDuringExecutior
- Key: node-role.kubernetes.io/master
- Use the right operator

Terminal 1

```
root@controlplane:~# kubectl create deployment red --image=nginx --replicas=2 --dry-run=client -o yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: red
  name: red
spec:
  replicas: 2
  selector:
    matchLabels:
      app: red
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: red
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
root@controlplane:~# kubectl create deployment red --image=nginx --replicas=2 --dry-run=client -o yaml >
 red.yaml
```

Task Solution 47:26

01 02 03 04 05 06 07 08

Create a new deployment named `red` with the `nginx` image and `2` replicas, and ensure it gets placed on the `controlplane` node only.

Use the label - `node-role.kubernetes.io/master` - set on the controlplane node.

Check

- Name: red
- Replicas: 2
- Image: nginx
- NodeAffinity: requiredDuringSchedulingIgnoredDuringExecutior
- Key: node-role.kubernetes.io/master
- Use the right operator

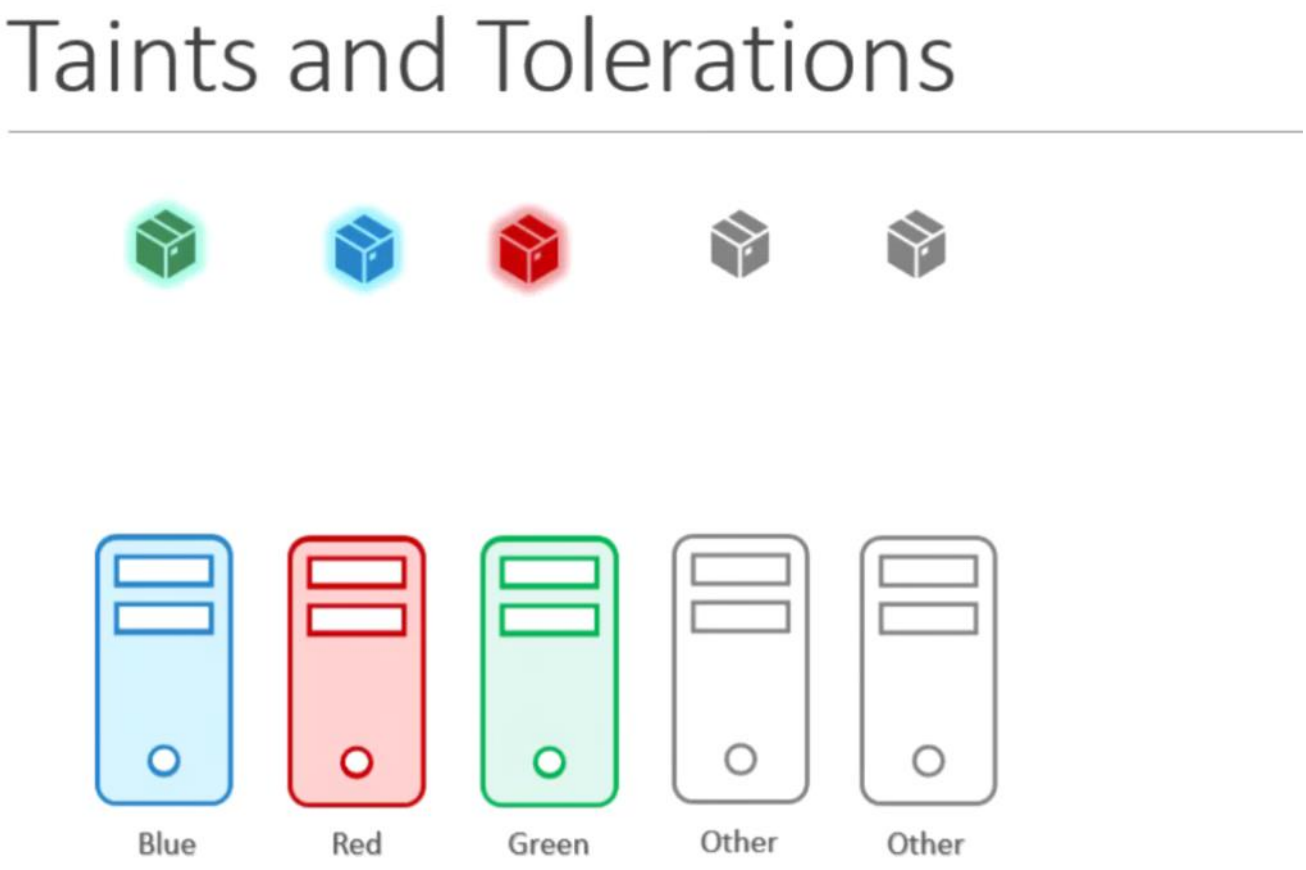
Terminal 1

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: red
  name: red
spec:
  replicas: 2
  selector:
    matchLabels:
      app: red
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: red
    spec:
        affinity:
            nodeAffinity:
              requiredDuringSchedulingIgnoredDuringExecution:
                nodeSelectorTerms:
                - matchExpressions:
                  - key: node-role.kubernetes.io/master
                    operator: Exists
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
~
~
~
~
~
~
~
:wq!
```

```
root@controlplane:~# vi red.yaml
root@controlplane:~#
root@controlplane:~# kubectl create -f red.yaml
error: error parsing red.yaml: error converting YAML to JSON: yaml: line 26: did not find expected key
root@controlplane:~# vi red.yaml
root@controlplane:~# vi red.yaml
root@controlplane:~# kubectl create -f red.yaml
deployment.apps/red created
root@controlplane:~#
```

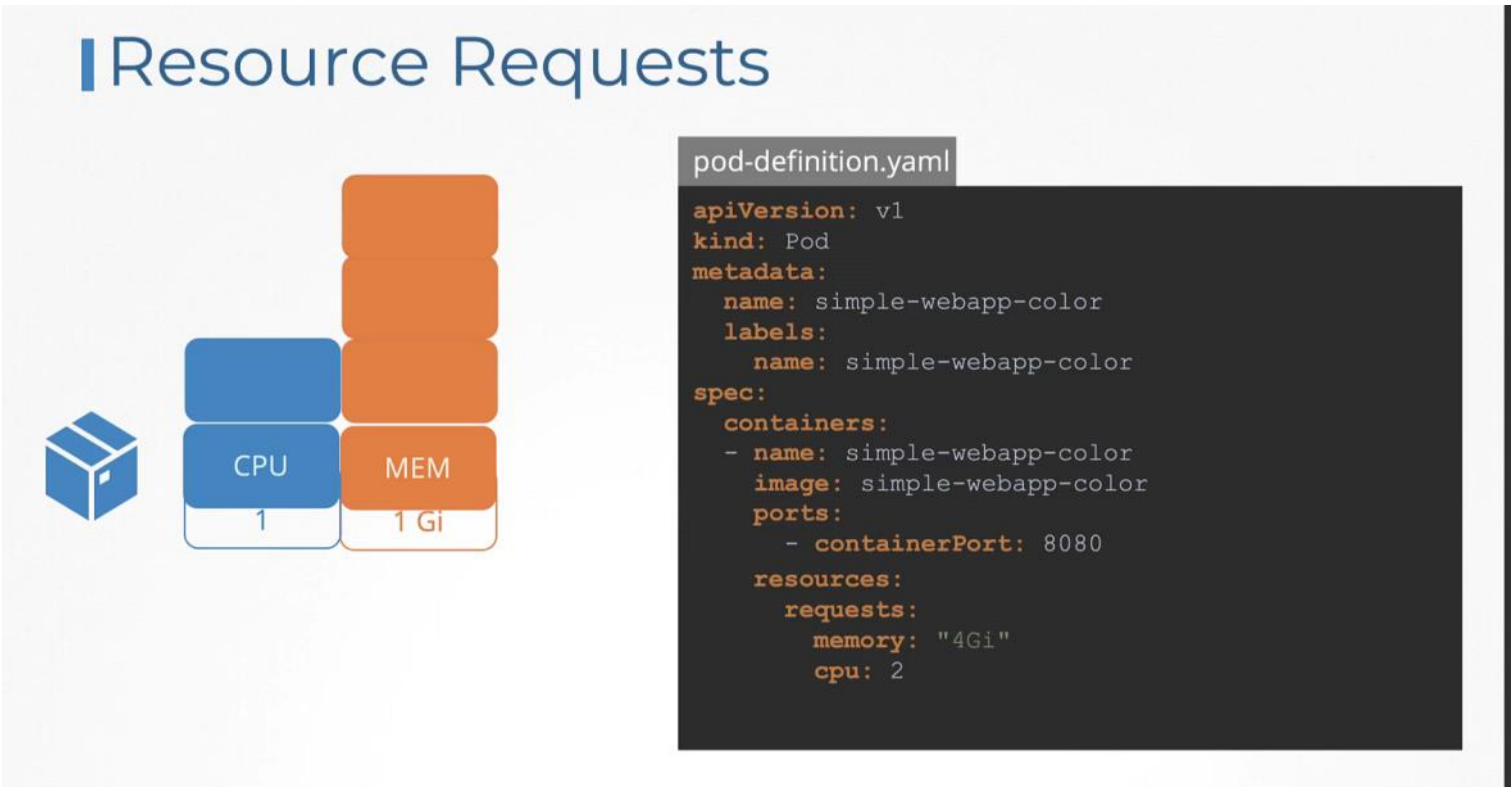
Taints and Tolerations

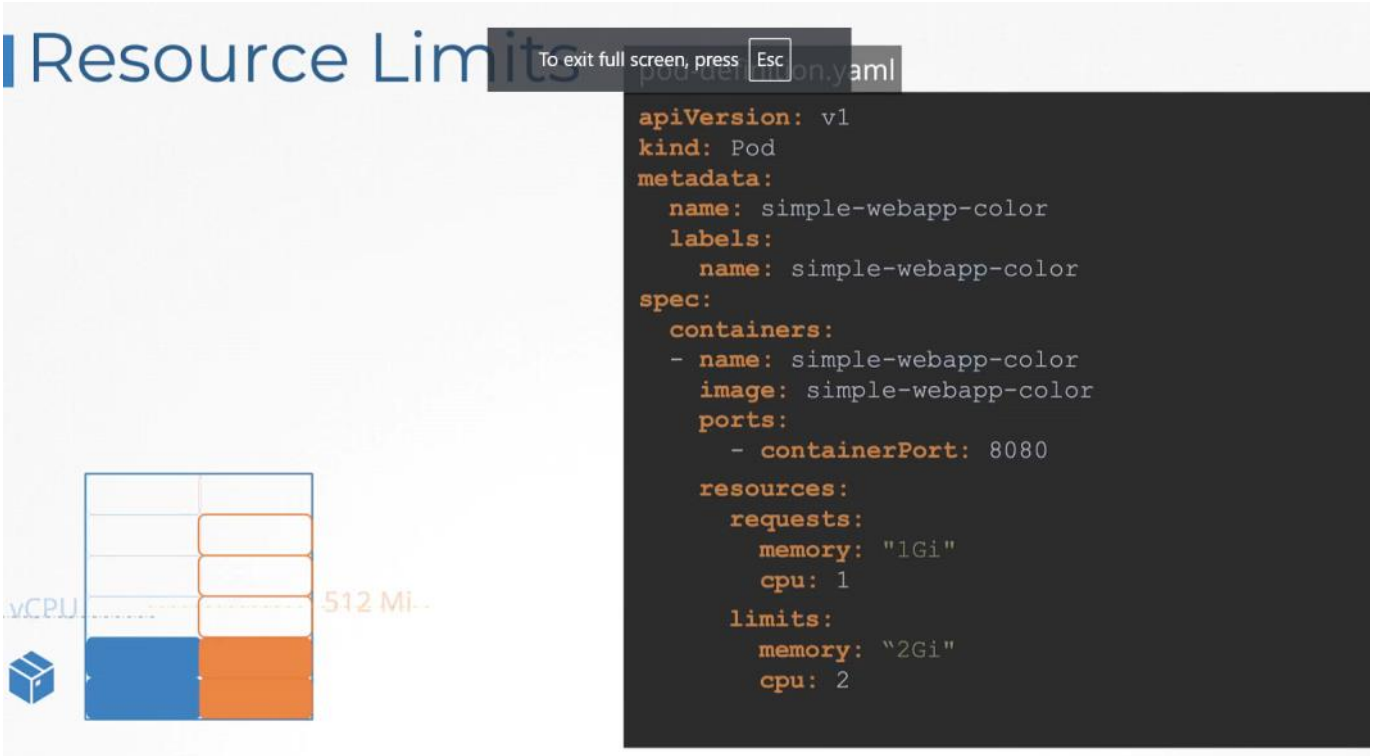
Tuesday, June 11, 2024 11:03 AM



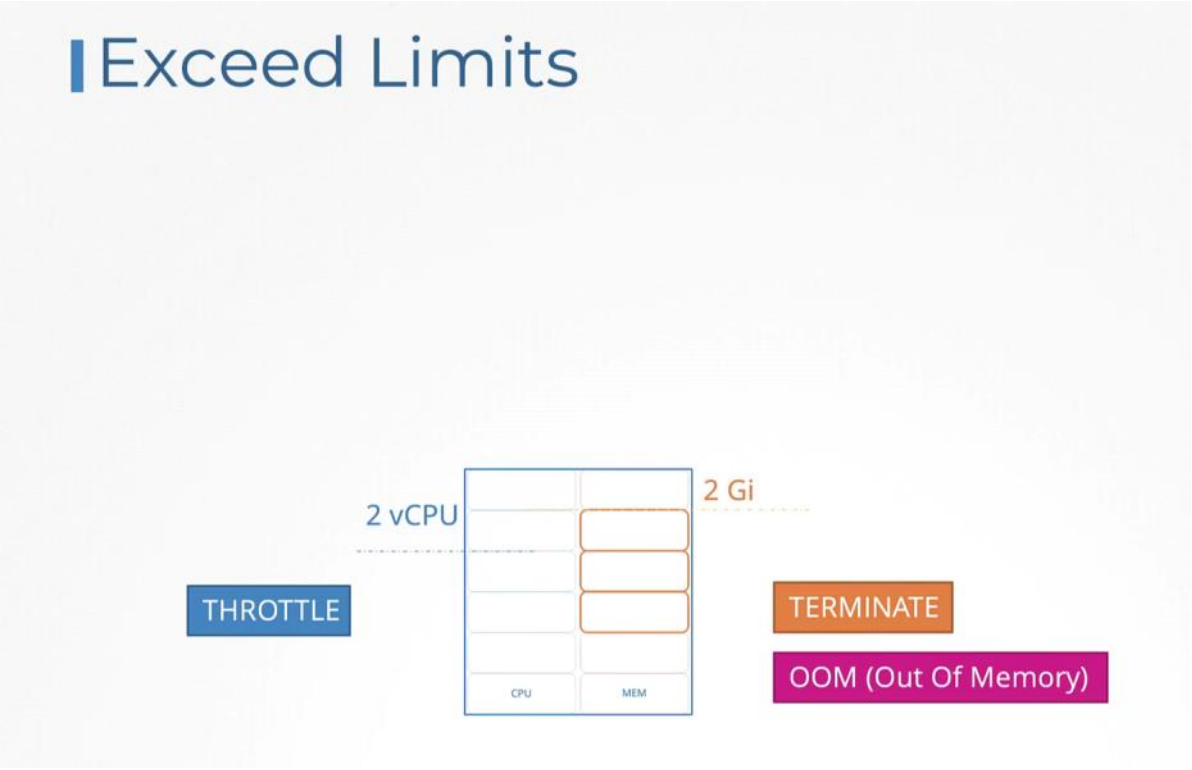
Resource req and limit

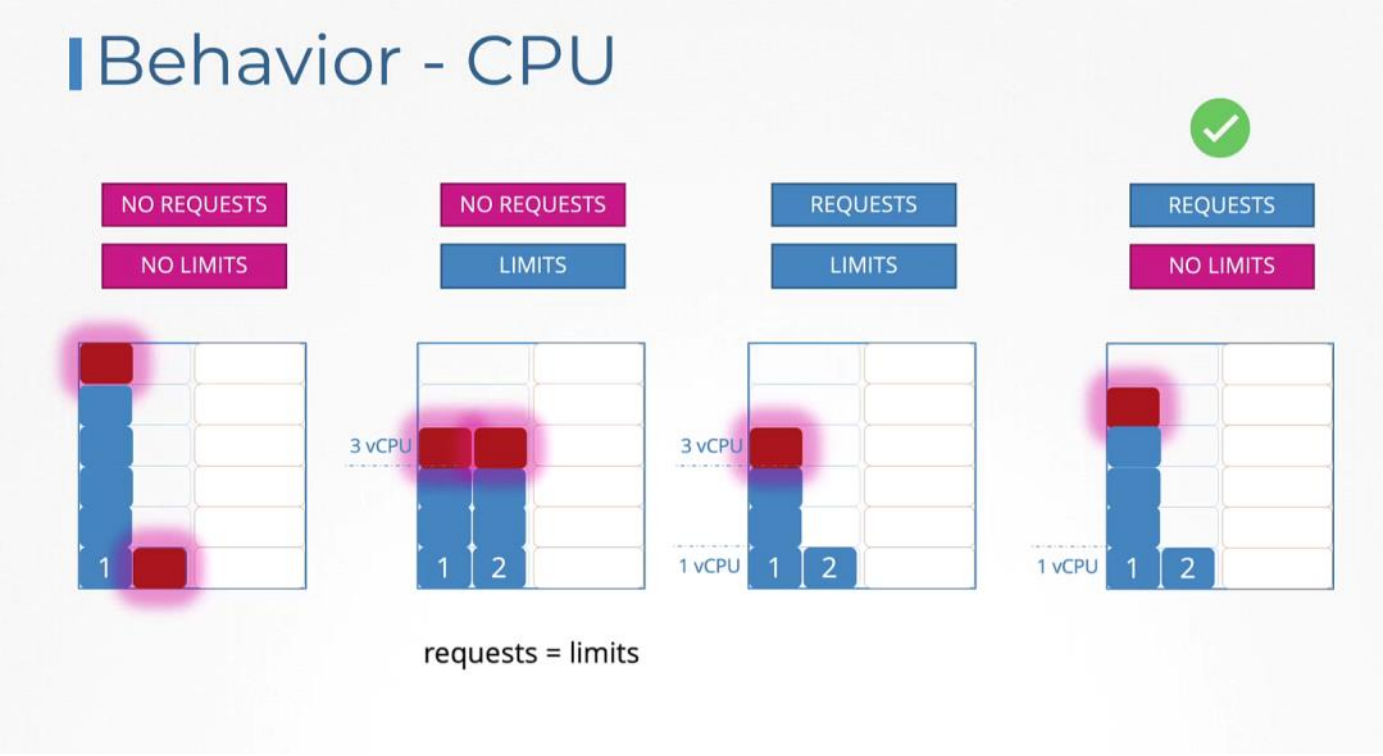
Tuesday, June 11, 2024 5:38 PM





resource req are the memory you will get guaranteed when creating a pod
limits are defined by you for each container in a pod
a container cannot use more cpu than the limit but it can use more memory than the cpu but it will get terminated and will show a OOM error





if there is no requests set then it will set req = limit

ideally you should set requests but no limits because sometimes it will require more than the limit set

Behavior - Memory

requests = limits

LimitRange

limit-range-cpu.yaml

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-resource-constraint
spec:
  limits:
  - default:
      cpu: 500m
    defaultRequest:
      cpu: 500m
    max:
      cpu: "1"
    min:
      cpu: 100m
    type: Container
```

LimitRange

limit-range-cpu.yaml

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-resource-constraint
spec:
  limits:
  - default:
      cpu: 500m                    limit
    defaultRequest:
      cpu: 500m                    request
    max:
      cpu: "1"                     limit
    min:
      cpu: 100m                    request
    type: Container
```

limit-range-memory.yaml

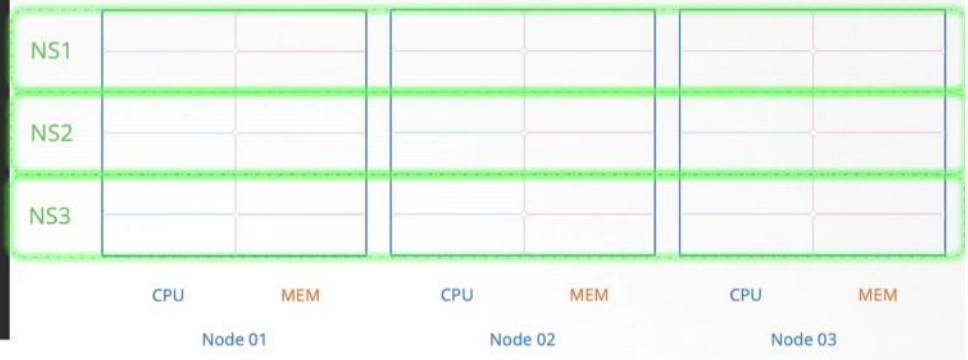
```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: memory-resource-constraint
spec:
  limits:
  - default:
      memory: 1Gi
    defaultRequest:
      memory: 1Gi
    max:
      memory: 1Gi
    min:
      memory: 500Mi
    type: Container
```

these limits and req set will only affect the newly created pods and not the existing ones.

Resource Quotas

resource-quota.yaml

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: my-resource-quota
spec:
  hard:
    requests.cpu: 4
    requests.memory: 4Gi
    limits.cpu: 10
    limits.memory: 10Gi
```



Practice Resource limits

Tuesday, June 11, 2024 6:33 PM

1 2 3 4 5 6 7

A pod called `rabbit` is deployed. Identify the CPU requirements set on the Pod in the current(default) namespace

1

2

3

4

0

```
controlplane ~ ➜  kubectl get pods
NAME     READY   STATUS             RESTARTS      AGE
rabbit   0/1     RunContainerError   2 (3s ago)   29s

controlplane ~ ➜

controlplane ~ ➜

controlplane ~ ➜  kubectl describe pod rabbg
```

1 2 3 4 5 6 7

A pod called `rabbit` is deployed. Identify the CPU requirements set on the Pod in the current(default) namespace

1

2

3

4

0

```
    Image:          ubuntu
    Image ID:       docker.io/library/ubuntu@sha256:9101220a875cee98b016668342c489ff0674f247f6ca20dfc91b
1c0f28581ae
    Port:           <none>
    Host Port:      <none>
    Args:
      sleep
      1000
    State:          Waiting
      Reason:       CrashLoopBackOff
    Last State:     Terminated
      Reason:       StartError
      Message:      failed to create containerd task: failed to create shim: OCI runtime create failed:
container_linux.go:380: starting container process caused: process_linux.go:545: container init caused:
process_linux.go:508: setting cgroup config for procHooks process caused: failed to write "200000": wri
e /sys/fs/cgroup/cpu,cpuacct/kubepods/burstable/podaca02fbb-199b-4917-b327-feceb73a1359/e10d0fa67ccbb4d
cb646c80400c72fba2a446a0679f73e6804c7a118b3dd236/cpu.cfs_quota_us: invalid argument: unknown
      Exit Code:    128
      Started:      Thu, 01 Jan 1970 00:00:00 +0000
      Finished:     Sat, 16 Apr 2022 18:00:44 +0000
    Ready:          False
    Restart Count:  2
    Limits:
      cpu:  2
    Requests:
      cpu:        1
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-zz492 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  kube-api-access-zz492:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
```

1 2 3 4 5 6 7

Delete the `rabbit` Pod.
Once deleted, wait for the pod to fully terminate.

Check

- Delete Pod rabbit

```
controlplane ~ ➜  kubectl delete pod rabbit
pod "rabbit" deleted
```

1 2 3 4 5 6 7

Another pod called `elephant` has been deployed in the default namespace. It fails to get to a running state. Inspect this pod and identify the `Reason` why it is not running.

Running

Ready

CrashLoopBackOff

OOMKilled

```
controlplane ~ ➜  kubectl get pod
NAME       READY   STATUS             RESTARTS      AGE
elephant   0/1     CrashLoopBackOff   1 (3s ago)    8s

controlplane ~ ➜  kubectl describe pod elephant
```

1 2 3 4 5 6 7

Another pod called `elephant` has been deployed in the default namespace. It fails to get to a running state. Inspect this pod and identify the `Reason` why it is not running.

- Running
- Ready
- CrashLoopBackOff
- OOMKilled

```
Start Time:   Sat, 16 Apr 2022 18:01:36 +0000
Labels:       <none>
Annotations:  <none>
Status:       Running
IP:           10.42.0.10
IPs:
  IP:  10.42.0.10
Containers:
  mem-stress:
    Container ID:  containerd://464974e43f3c45ee417bd9894c55113b99e0ac52d8ed9d5cdededb4da7ba2617
    Image:         polinux/stress
    Image ID:      docker.io/polinux/stress@sha256:b6144f84f9c15dac80deb48d3a646b55c7043ab1d83ea0a697c09
097aaad21aa
    Port:          <none>
    Host Port:     <none>
    Command:
      stress
    Args:
      --vm
      1
      --vm-bytes
      15M
      --vm-hang
      1
    State:          Waiting
      Reason:       CrashLoopBackOff
    Last State:     Terminated
      Reason:       OOMKilled
      Exit Code:    1
      Started:      Sat, 16 Apr 2022 18:01:41 +0000
      Finished:     Sat, 16 Apr 2022 18:01:41 +0000
    Ready:          False
    Restart Count:  1
    Limits:
      memory:  10Mi
    Requests:
      memory:     5Mi
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-prsdp (ro)
```

1 2 3 4 5 6 7

The status `OOMKilled` indicates that it is failing because the pod ran out of memory. Identify the memory limit set on the POD.

- Ok

```
Labels:       <none>
Annotations:  <none>
Status:       Running
IP:           10.42.0.10
IPs:
  IP:  10.42.0.10
Containers:
  mem-stress:
    Container ID:  containerd://464974e43f3c45ee417bd9894c55113b99e0ac52d8ed9d5cdededb4da7ba2617
    Image:         polinux/stress
    Image ID:      docker.io/polinux/stress@sha256:b6144f84f9c15dac80deb48d3a646b55c7043ab1d83ea0a697c0
097aaad21aa
    Port:          <none>
    Host Port:     <none>
    Command:
      stress
    Args:
      --vm
      1
      --vm-bytes
      15M
      --vm-hang
      1
    State:          Waiting
      Reason:       CrashLoopBackOff
    Last State:     Terminated
      Reason:       OOMKilled
      Exit Code:    1
      Started:      Sat, 16 Apr 2022 18:01:41 +0000
      Finished:     Sat, 16 Apr 2022 18:01:41 +0000
    Ready:          False
    Restart Count:  1
    Limits:
      memory:  10Mi
    Requests:
      memory:     5Mi
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-prsdp (ro)
Conditions:
```

1 2 3 4 5 6 7

The `elephant` pod runs a process that consume 15Mi of memory. Increase the limit of the `elephant` pod to 20Mi.

Delete and recreate the pod if required. Do not modify anything other than the required fields.

Check

- Pod Name: elephant
- Image Name: polinux/stress
- Memory Limit: 20Mi

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2022-04-16T18:01:36Z"
  name: elephant
  namespace: default
  resourceVersion: "934"
  uid: 5992055a-2211-4d16-b1a6-67cdaef010c0
spec:
  containers:
  - args:
    - --vm
    - "1"
    - --vm-bytes
    - 15M
    - --vm-hang
    - "1"
    command:
    - stress
    image: polinux/stress
    imagePullPolicy: Always
    name: mem-stress
    resources:
      limits:
        memory: 20Mi
      requests:
        memory: 5Mi
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-prsdp
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
-- INSERT (paste) --
```

```
controlplane ~ ➜  kubectl edit pod elephant
error: pods "elephant" is invalid
A copy of your changes has been stored to "/tmp/kubectl-edit-3376288381.yaml"
error: Edit cancelled, no valid changes were saved.

controlplane ~ ✖ cat v
```

```
controlplane ~ ➜  kubectl replace --force -f ^C

controlplane ~ ✖ kubectl replace --force -f /tmp/kubectl-edit-3376288381.yaml
pod "elephant" deleted
pod/elephant replaced

controlplane ~ ➜

controlplane ~ ➜  kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
elephant   1/1     Running   0          8s

controlplane ~ ➜
```

1 2 3 4 5 6 7

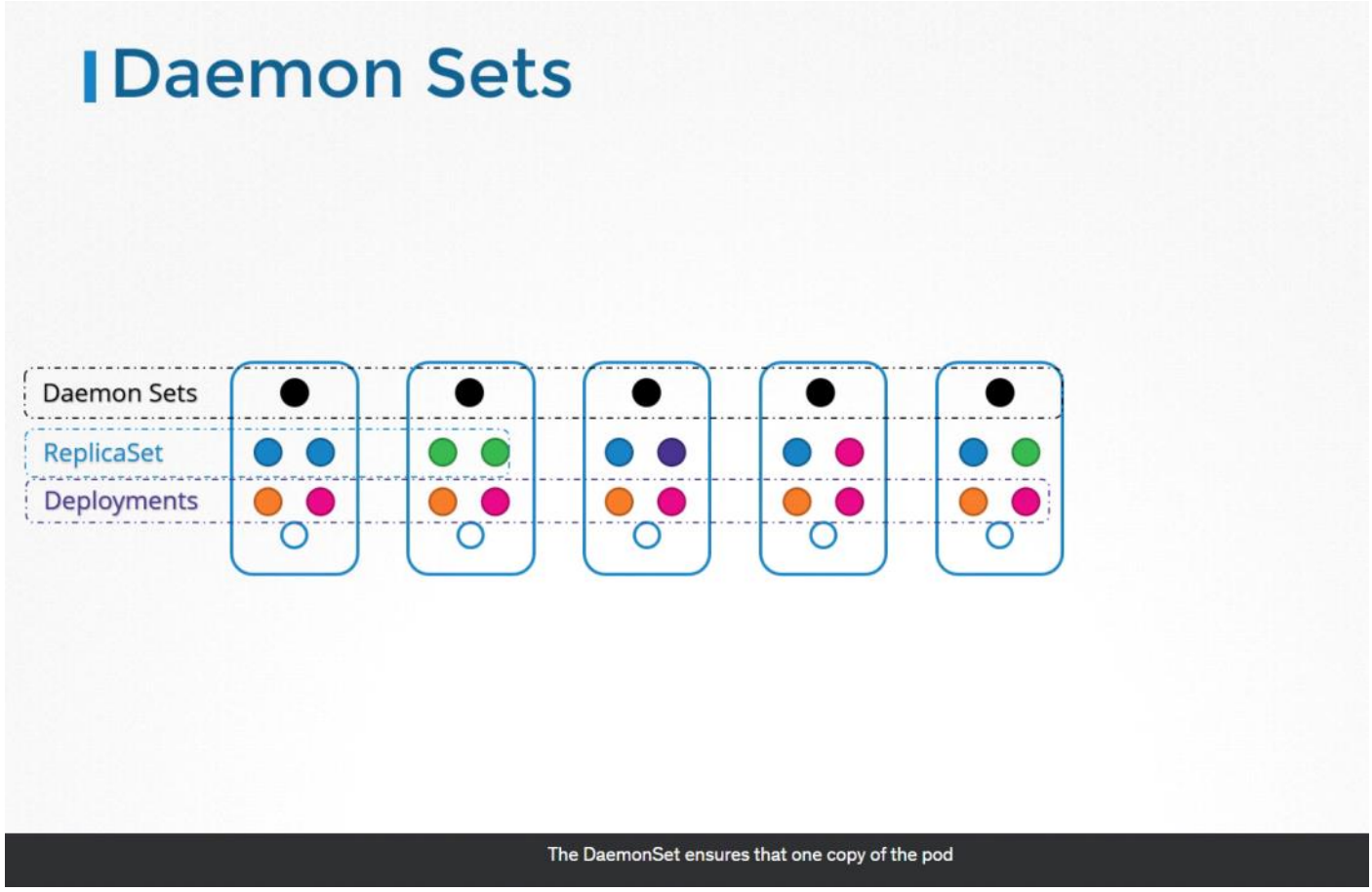
Inspect the status of POD. Make sure it's running

Ok

```
controlplane ~ ➜

controlplane ~ ➜  kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
elephant   1/1     Running   0          8s

controlplane ~ ➜  kubectl describe pod elephant
Name:         elephant
Namespace:    default
Priority:     0
Node:         controlplane/172.25.0.22
Start Time:   Sat, 16 Apr 2022 18:04:55 +0000
Labels:       <none>
Annotations:  <none>
Status:       Running
IP:           10.42.0.11
IPs:
  IP:  10.42.0.11
Containers:
  mem-stress:
    Container ID:  containerd://a3f3388244e2b84168ff5036d786795114dd49b22a173d828a2bdf8683978e
    Image:         polinux/stress
    Image ID:      docker.io/polinux/stress@sha256:b6144f84f9c15dac80deb48d3a646b55c7043ab1d83
```

Daemon sets

Wednesday, June 12, 2024 10:07 AM



DaemonSet Definition

daemon-set-definition.yaml

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: monitoring-daemon
spec:
  selector:
    matchLabels:
      app: monitoring-agent
  template:
    metadata:
      labels:
        app: monitoring-agent
    spec:
      containers:
      - name: monitoring-agent
        image: monitoring-agent
```

replicaset-definition.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: monitoring-daemon
spec:
  selector:
    matchLabels:
      app: monitoring-agent
  template:
    metadata:
      labels:
        app: monitoring-agent
    spec:
      containers:
      - name: monitoring-agent
        image: monitoring-agent
```

```
kubectl create -f daemon-set-definition.yaml
```

```
daemon-set Created
```

View DaemonSets

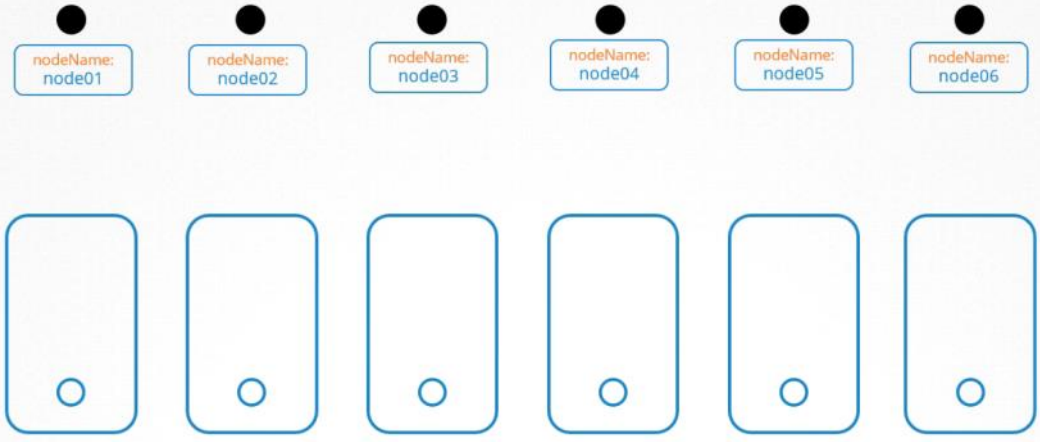
```
kubectl get daemonsets
NAME              DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   AGE
monitoring-daemon 1         1         1         1            1           41
```

```
kubectl describe daemonsets monitoring-daemon
Name:           monitoring-daemon
Selector:       name=monitoring-daemon
Node-Selector:  <none>
Labels:         name=monitoring-daemon
Desired Number of Nodes Scheduled: 2
Current Number of Nodes Scheduled: 2
Number of Nodes Scheduled with Up-to-date Pods: 2
Number of Nodes Scheduled with Available Pods: 1
Number of Nodes Misscheduled: 0
Pods Status:  2 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:           app=monitoring-agent
  Containers:
```

kubectl get daemonsets --all-namespaces

Deamonsets make sure that there is one copy of the container on each pod in cluster

How does it work?



Earlier we used to add node name property to the pods and the pods would land on the nodes. now we use node affinity rule for this purpose.

Practice Daemon sets

Wednesday, June 12, 2024 10:22 AM

01 02 03 04 05 06

How many `DaemonSets` are created in the cluster in all namespaces?

Check all namespaces

- 3
- 1
- 4
- 2

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get daemonsets
No resources found in default namespace.
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get daemonsets -A
NAMESPACE     NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
      AGE
kube-system   kube-flannel-ds   1         1         1       1            1           <none>
      24m
kube-system   kube-proxy        1         1         1       1            1           kubernetes.io/os=li
nux   24m
root@controlplane:~#
```

01 02 03 04 05 06

Which namespace are the `DaemonSets` created in?

- weave-net
- kube-public
- kube-proxy
- default
- kube-system

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get daemonsets
No resources found in default namespace.
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get daemonsets -A
NAMESPACE     NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
      AGE
kube-system   kube-flannel-ds   1         1         1       1            1           <none>
      24m
kube-system   kube-proxy        1         1         1       1            1           kubernetes.io/os=li
nux   24m
root@controlplane:~#
```

01 02 03 04 05 06

Which of the below is a `DaemonSet` ?

- scheduler
- coredns
- etcd-master
- kube-flannel-ds

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get daemonsets
No resources found in default namespace.
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get daemonsets -A
NAMESPACE     NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
      AGE
kube-system   kube-flannel-ds   1         1         1       1            1           <none>
      24m
kube-system   kube-proxy        1         1         1       1            1           kubernetes.io/os=li
nux   24m
root@controlplane:~#
```

01 02 03 04 05 06

On how many nodes are the pods scheduled by the DaemonSet `kube-proxy`

3

0

2

1

```
root@controlplane:~#
root@controlplane:~# kubectl get daemonsets -A
NAMESPACE     NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
      AGE
kube-system   kube-flannel-ds   1         1         1       1            1           <none>
      24m
kube-system   kube-proxy        1         1         1       1            1           kubernetes.io/os=li
nux   24m
root@controlplane:~# kubectl describe daemonsets kube-proxy
Error from server (NotFound): daemonsets.apps "kube-proxy" not found
root@controlplane:~# kubectl describe daemonsets kube-proxy -n kube-system
Name:           kube-proxy
Selector:       k8s-app=kube-proxy
Node-Selector:  kubernetes.io/os=linux
Labels:         k8s-app=kube-proxy
Annotations:    deprecated.daemonset.template.generation: 1
Desired Number of Nodes Scheduled: 1
Current Number of Nodes Scheduled: 1
Number of Nodes Scheduled with Up-to-date Pods: 1
Number of Nodes Scheduled with Available Pods: 1
Number of Nodes Misscheduled: 0
Pods Status:  1 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:           k8s-app=kube-proxy
  Service Account:  kube-proxy
  Containers:
   kube-proxy:
    Image:      k8s.gcr.io/kube-proxy:v1.20.0
    Port:       <none>
    Host Port:  <none>
    Command:
```

01 02 03 04 05 06

What is the image used by the POD deployed by the `kube-flannel-ds` DaemonSet?

nginx

quay.io/coreos/flannel:v0.13.1-rc1

weave-kube:2.0

weave-kube:2.4

```
root@controlplane:~# kubectl describe ds kube-flannel-ds -n kube-system
Name:           kube-flannel-ds
Selector:       app=flannel
Node-Selector:  <none>
Labels:         app=flannel
                tier=node
Annotations:    deprecated.daemonset.template.generation: 1
Desired Number of Nodes Scheduled: 1
Current Number of Nodes Scheduled: 1
Number of Nodes Scheduled with Up-to-date Pods: 1
Number of Nodes Scheduled with Available Pods: 1
Number of Nodes Misscheduled: 0
Pods Status:  1 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:           app=flannel
                    tier=node
  Service Account:  flannel
  Init Containers:
   install-cni:
    Image:      quay.io/coreos/flannel:v0.13.1-rc1
    Port:       <none>
    Host Port:  <none>
    Command:
      cp
    Args:
      -f
      /etc/kube-flannel/cni-conf.json
      /etc/cni/net.d/10-flannel.conflist
    Environment:  <none>
    Mounts:
      /etc/cni/net.d from cni (rw)
```

01 02 03 04 05 06

Deploy a DaemonSet for `FluentD` Logging.

Use the given specifications.

Check

- Name: elasticsearch
- Namespace: kube-system
- Image: k8s.gcr.io/fluentd-elasticsearch:1.20

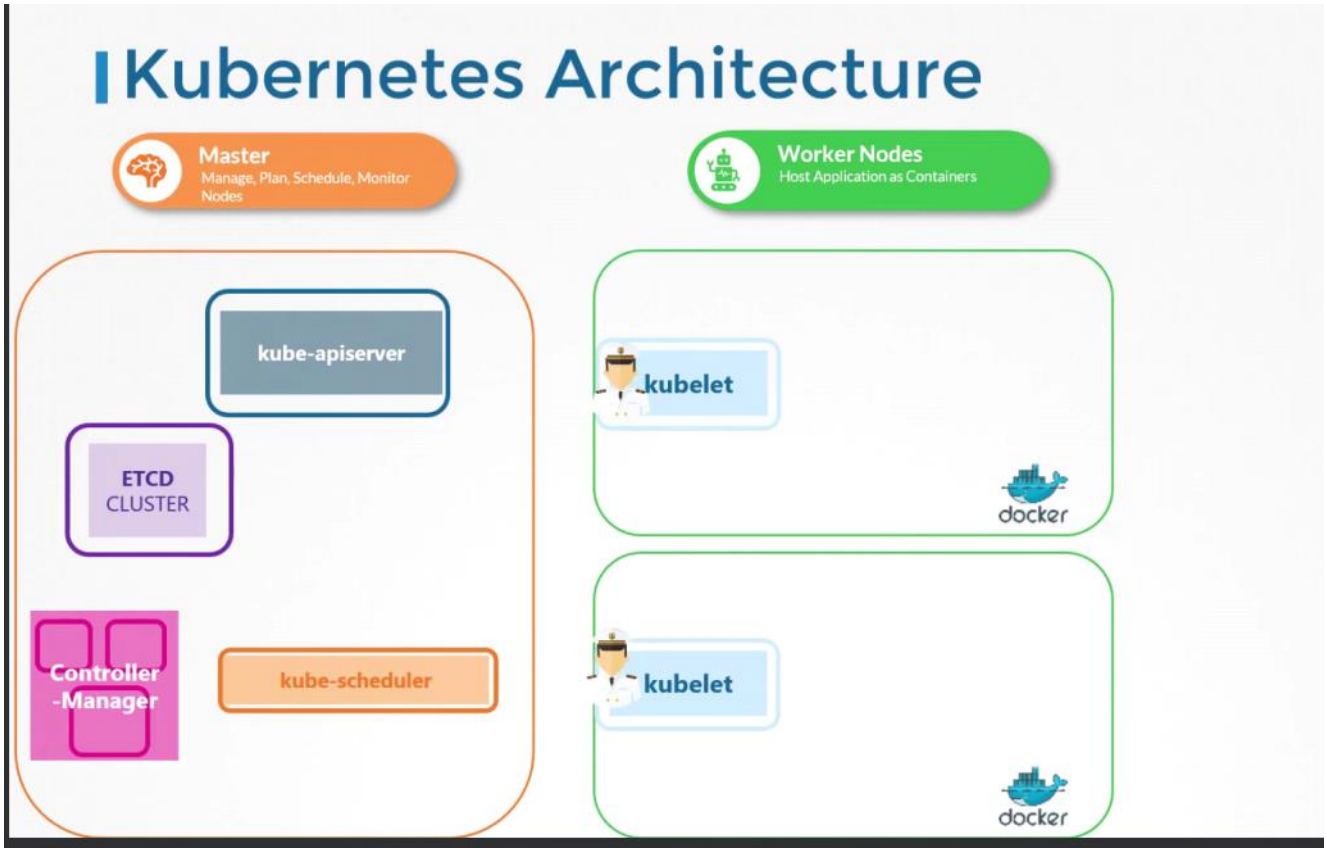
```
root@controlplane:~# kubectl create daemonset
Error: must specify one of -f and -k

error: unknown command "daemonset"
See 'kubectl create -h' for help and examples
root@controlplane:~# kubectl create deployment elasticsearch -n kube-system --image=k8s.gcr.io/fluentd-e
lasticsearch:1.20 --dry-run=client -o yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: elasticsearch
  name: elasticsearch
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: elasticsearch
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: elasticsearch
    spec:
      containers:
      - image: k8s.gcr.io/fluentd-elasticsearch:1.20
        name: fluentd-elasticsearch
        resources: {}
status: {}
root@controlplane:~# kubectl create deployment elasticsearch -n kube-system --image=k8s.gcr.io/fluentd-e
lasticsearch:1.20 --dry-run=client -o yaml > fluentd.yaml
root@controlplane:~#
root@controlplane:~# vi fluentd.yaml
root@controlplane:~#
```

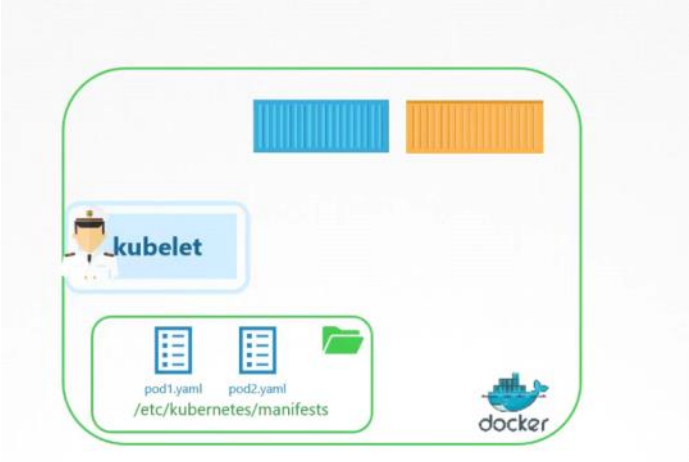
```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  creationTimestamp: null
  labels:
    app: elasticsearch
  name: elasticsearch
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: elasticsearch
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: elasticsearch
    spec:
      containers:
      - image: k8s.gcr.io/fluentd-elasticsearch:1.20
        name: fluentd-elasticsearch
        resources: {}
~
~
~
```

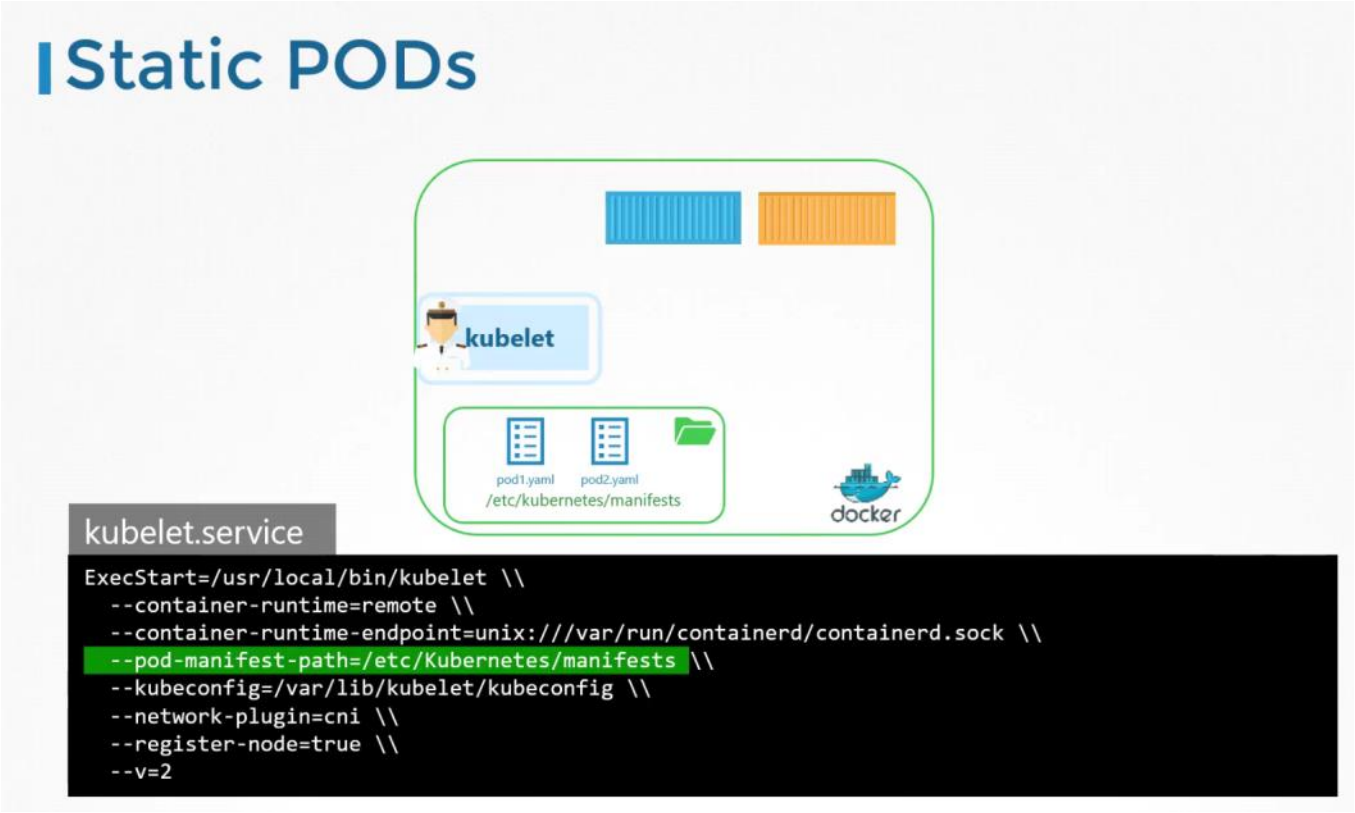
```
root@controlplane:~#
root@controlplane:~# vi fluentd.yaml
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl create -f fluentd.yaml
daemonset.apps/elasticsearch created
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get ds
No resources found in default namespace.
root@controlplane:~# kubectl get ds -n kube-system
NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
elasticsearch     1         1         0       1            0           <none>                   10s
kube-flannel-ds   1         1         1       1            1           <none>                   29m
kube-proxy        1         1         1       1            1           kubernetes.io/os=linux   29m
root@controlplane:~#
```

Static Pods

Wednesday, June 12, 2024 8:06 PM

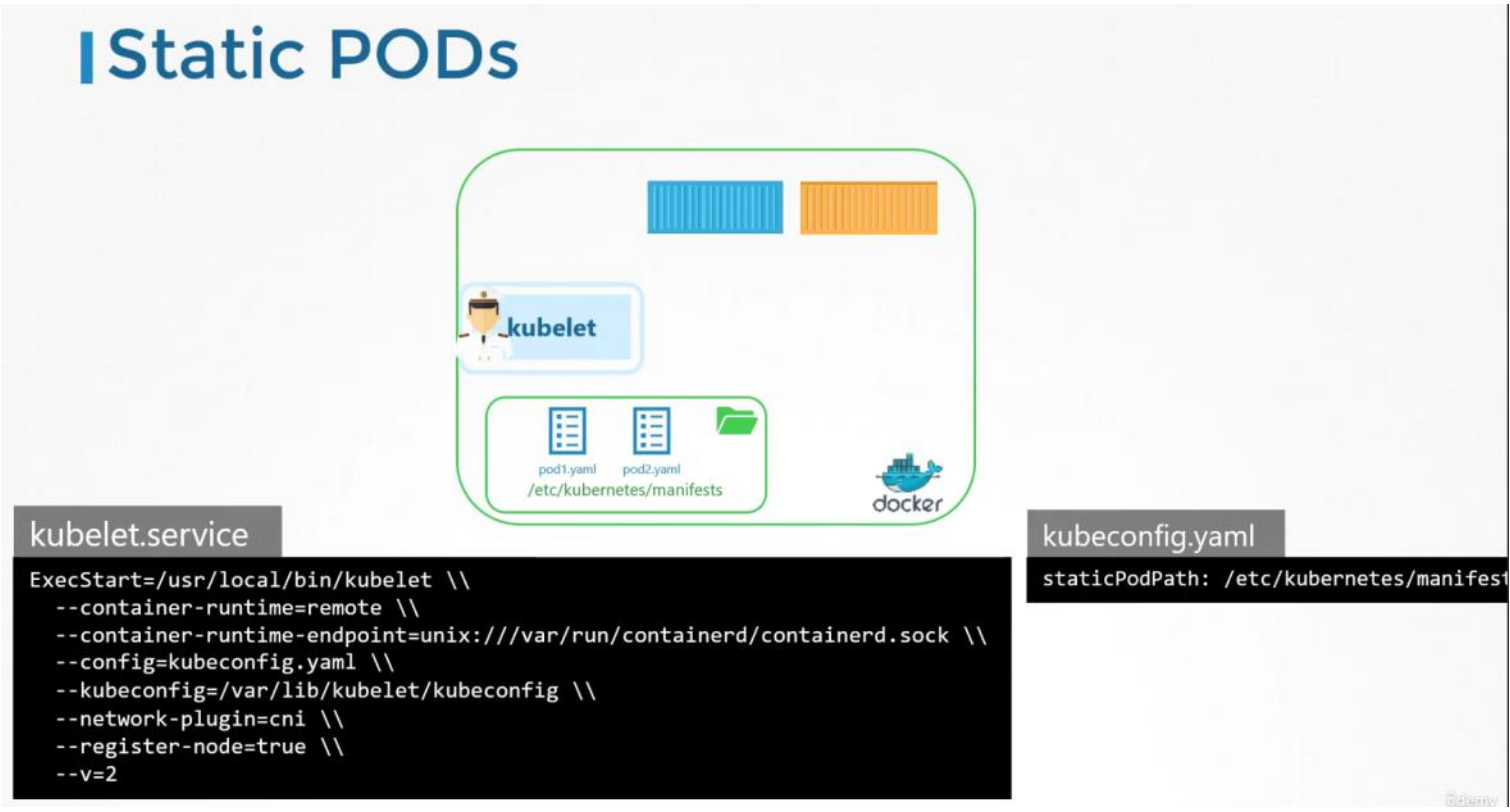






How to create a pod without the help of kubeapi server(how to provide pod definnition file to kubelet without kube api server)

static pods are created by the kubelet without the intervention of kube api server



Static PODs

Note

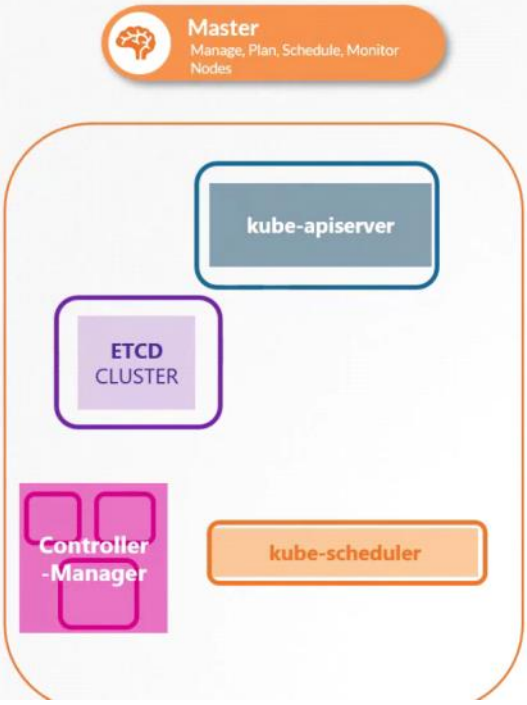
Use the relevant command for each runtime

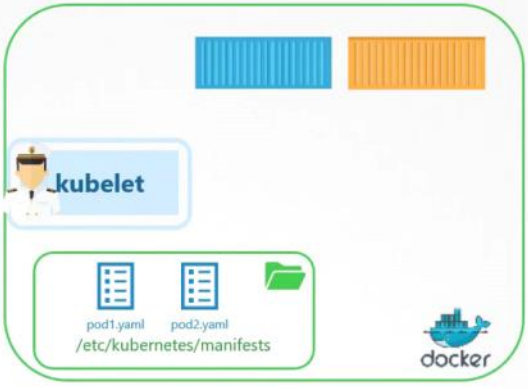
cri-o `crictl ps`

containerd `nerdctl ps`

```
docker ps
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS
PORTS               NAMES
8e5d4c4db7b6        busybox                "sh -c 'echo Hello K…"   20 seconds ago      Up 20 seconds
k8s_myapp-container_myapp-pod-host01_default_48e37fb432f2e06350e76786bd0bac66_0
f6737e1149cb        k8s.gcr.io/pause:3.1   "/pause"                 24 seconds ago      Up 23 seconds
k8s_POD_myapp-pod-host01_default_48e37fb432f2e06350e76786bd0bac66_0
```

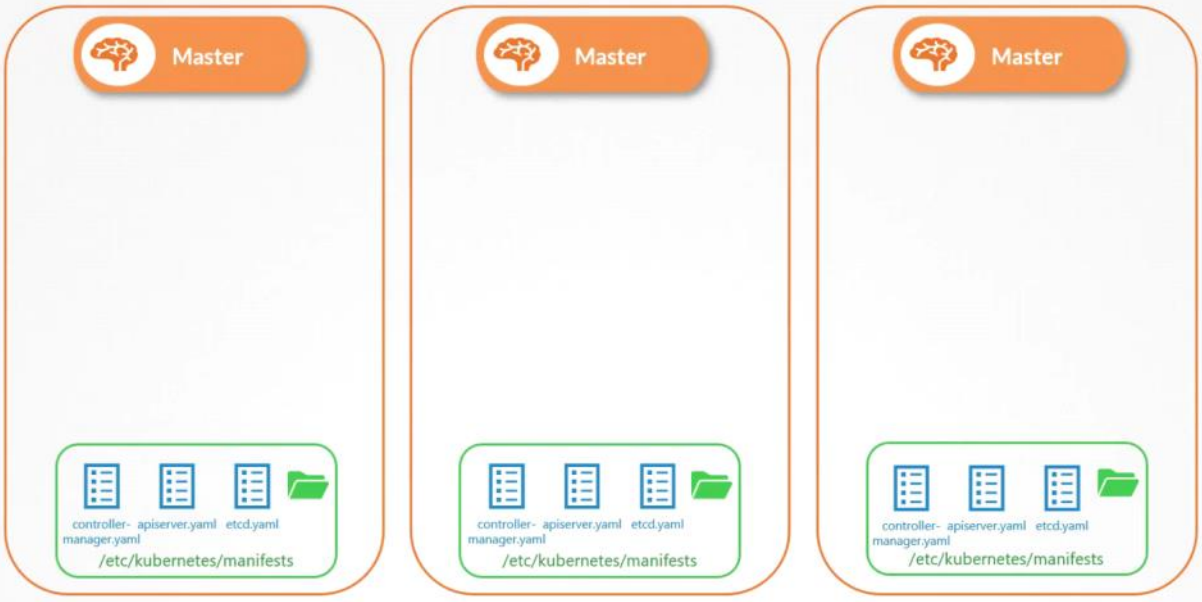
Static PODs



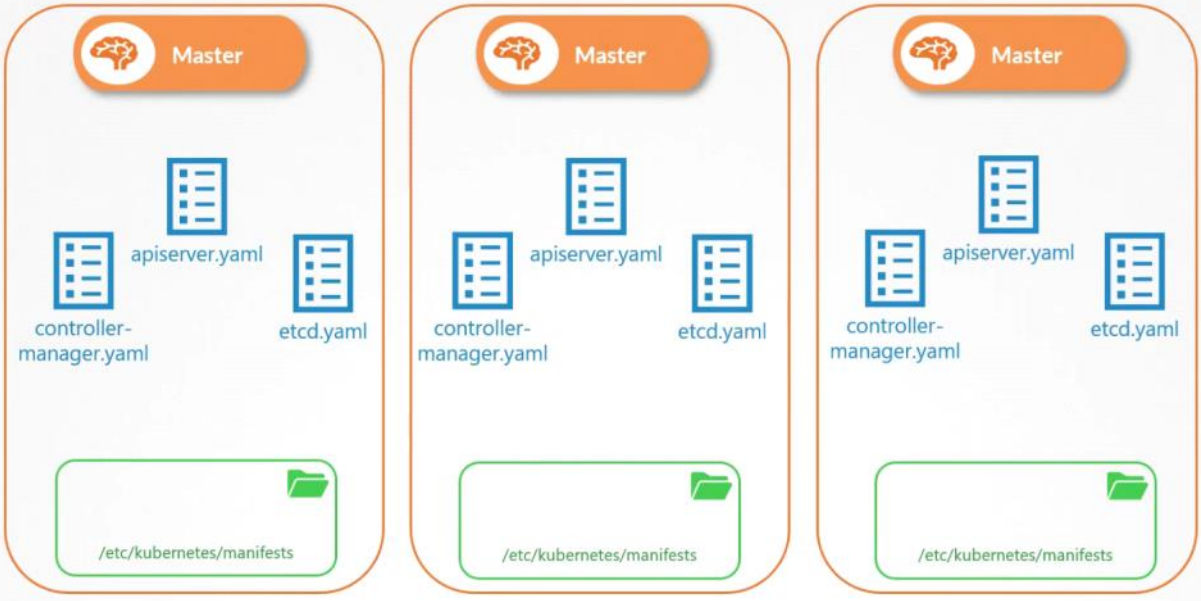



```
kubectl get pods
NAME               READY   STATUS              RESTARTS   AGE
static-web-node01  0/1     ContainerCreating   0          29s
```

Use Case



Use Case



Use Case

```
kubectl get pods -n kube-system
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
kube-system   coredns-78fcdf6894-hwrq9          1/1     Running   0          16m
kube-system   coredns-78fcdf6894-rzhjr          1/1     Running   0          16m
kube-system   etcd-master                       1/1     Running   0          15m
kube-system   kube-apiserver-master             1/1     Running   0          15m
kube-system   kube-controller-manager-master    1/1     Running   0          15m
kube-system   kube-proxy-lzt6f                  1/1     Running   0          16m
kube-system   kube-proxy-zm5qd                  1/1     Running   0          16m
kube-system   kube-scheduler-master             1/1     Running   0          15m
kube-system   weave-net-29z42                   2/2     Running   1          16m
kube-system   weave-net-snmdl                   2/2     Running   1          16m    -
```

Static PODs vs DaemonSets

Static PODs	DaemonSets
Created by the Kubelet	Created by Kube-API server (DaemonSet Controller)
Deploy Control Plane components as Static Pods	Deploy Monitoring Agents, Logging Agents on nodes
Ignored by the Kube-Scheduler	

Practice static pods

Wednesday, June 12, 2024 8:27 PM

01 02 03 04 05 06 07 08 09 10

How many static pods exist in this cluster in all namespaces?

4

1

2

3

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get pods -A
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
kube-system   coredns-74ff55c5b-brwnd                    1/1     Running   0          9m44s
kube-system   coredns-74ff55c5b-nfg4f                    1/1     Running   0          9m43s
kube-system   etcd-controlplane                          1/1     Running   0          9m54s
kube-system   kube-apiserver-controlplane                1/1     Running   0          9m54s
kube-system   kube-controller-manager-controlplane       1/1     Running   0          9m54s
kube-system   kube-flannel-ds-95zmd                      1/1     Running   0          9m37s
kube-system   kube-flannel-ds-ct7qn                      1/1     Running   0          9m14s
kube-system   kube-proxy-c5zpk                           1/1     Running   0          9m44s
kube-system   kube-proxy-wclgr                           1/1     Running   0          9m14s
kube-system   kube-scheduler-controlplane                1/1     Running   0          9m54s
root@controlplane:~#
root@controlplane:~# kubectl get pod kube-apiserver-controlplane -n kube-system
NAME                          READY   STATUS    RESTARTS   AGE
kube-apiserver-controlplane   1/1     Running   0          10m
root@controlplane:~# kubectl get pod kube-apiserver-controlplane -n kube-system -o yaml
```

One way is to find the node name at the last of pod name here kubeapiserver-controlplane have node name at the end.
secondly you can look at the yaml file.

```
            f:lastProbeTime: {}
            f:lastTransitionTime: {}
            f:status: {}
            f:type: {}
          k:{"type":"Ready"}:
            .: {}
            f:lastProbeTime: {}
            f:lastTransitionTime: {}
            f:status: {}
            f:type: {}
        f:containerStatuses: {}
        f:hostIP: {}
        f:phase: {}
        f:podIP: {}
        f:podIPs:
          .: {}
          k:{"ip":"10.38.102.6"}:
            .: {}
            f:ip: {}
        f:startTime: {}
    manager: kubelet
    operation: Update
    time: "2022-04-16T18:52:45Z"
  name: kube-apiserver-controlplane
  namespace: kube-system
  ownerReferences:
  - apiVersion: v1
    controller: true
    kind: Node
    name: controlplane
    uid: fbab9414-1ae8-4f64-bc94-456dd8569e6c
  resourceVersion: "467"
  uid: 26acf5fa-82dc-4fef-adae-e208089ce9af
spec:
  containers:
  - command:
```

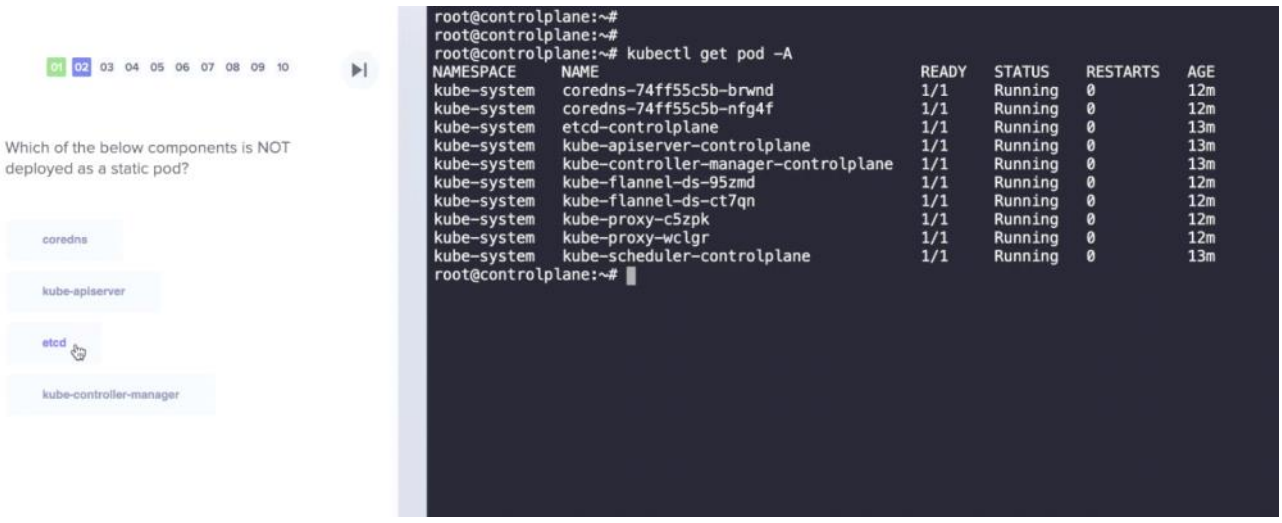
here in owner refrences kind is a node so its is a static pod.

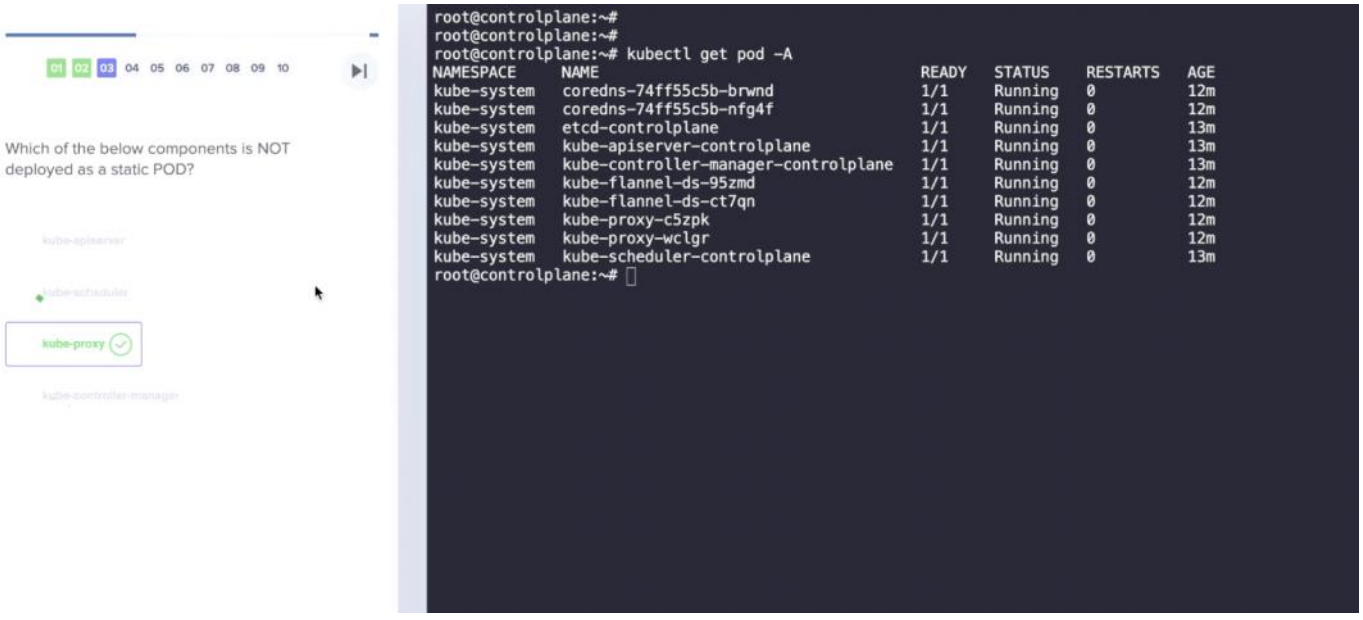
```
            .: {}
            f:lastProbeTime: {}
            f:lastTransitionTime: {}
            f:status: {}
            f:type: {}
        f:containerStatuses: {}
        f:hostIP: {}
        f:phase: {}
        f:podIP: {}
        f:podIPs:
          .: {}
          k:{"ip":"10.244.0.3"}:
            .: {}
            f:ip: {}
        f:startTime: {}
    manager: kubelet
    operation: Update
    time: "2022-04-16T18:53:13Z"
  name: coredns-74ff55c5b-brwnd
  namespace: kube-system
  ownerReferences:
  - apiVersion: apps/v1
    blockOwnerDeletion: true
    controller: true
    kind: ReplicaSet
    name: coredns-74ff55c5b
    uid: c2935e74-2f0f-44b4-b023-57e228a19b6d
  resourceVersion: "580"
  uid: 2d5c8057-c721-4ddb-8910-c00674004670
```

here in owner refrences kind is a replica set so its not a pod.

```
    kind: ReplicaSet
    name: coredns-74ff55c5b
    uid: c2935e74-2f0f-44b4-b023-57e228a19b6d
  resourceVersion: "580"
  uid: 2d5c8057-c721-4ddb-8910-c00674004670
spec:
  containers:
  - args:
    - -conf
    - /etc/coredns/Corefile
    image: k8s.gcr.io/coredns:1.7.0
    imagePullPolicy: IfNotPresent
```

here in owner refrences kind is a replica set so its not a pod.





01 02 03 04 05 06 07 08 09 10

On which nodes are the static pods created currently?

- controlplane
- node01
- controlplane & node01
- All Nodes

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get pod -A
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
kube-system   coredns-74ff55c5b-brwnd                    1/1     Running   0          12m
kube-system   coredns-74ff55c5b-nfg4f                    1/1     Running   0          12m
kube-system   etcd-controlplane                          1/1     Running   0          13m
kube-system   kube-apiserver-controlplane                1/1     Running   0          13m
kube-system   kube-controller-manager-controlplane       1/1     Running   0          13m
kube-system   kube-flannel-ds-95zmd                      1/1     Running   0          12m
kube-system   kube-flannel-ds-ct7qn                      1/1     Running   0          12m
kube-system   kube-proxy-c5zpk                           1/1     Running   0          12m
kube-system   kube-proxy-wclgr                           1/1     Running   0          12m
kube-system   kube-scheduler-controlplane                1/1     Running   0          13m
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10

What is the path of the directory holding the static pod definition files?

- /etc/kubernetes/manifests
- /etc/kubernetes/pods
- /etc/kubelet/manifests
- /var/kubelet/manifests

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get pod -A
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
kube-system   coredns-74ff55c5b-brwnd                    1/1     Running   0          12m
kube-system   coredns-74ff55c5b-nfg4f                    1/1     Running   0          12m
kube-system   etcd-controlplane                          1/1     Running   0          13m
kube-system   kube-apiserver-controlplane                1/1     Running   0          13m
kube-system   kube-controller-manager-controlplane       1/1     Running   0          13m
kube-system   kube-flannel-ds-95zmd                      1/1     Running   0          12m
kube-system   kube-flannel-ds-ct7qn                      1/1     Running   0          12m
kube-system   kube-proxy-c5zpk                           1/1     Running   0          12m
kube-system   kube-proxy-wclgr                           1/1     Running   0          12m
kube-system   kube-scheduler-controlplane                1/1     Running   0          13m
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# cat /var/lib/
apt/            docker/         etcd/           pam/            sudo/
cni/            docker-engine/  git/            polkit-1/       systemd/
containerd/     dockershim/     kubelet/        python/         ucf/
dbus/           dpkg/           misc/           rancher/        vim/
root@controlplane:~# cat /var/lib/kubelet/
config.yaml        device-plugins/    pki/               plugins_registry/  pods/
cpu_manager_state  kubeadm-flags.env  plugins/           pod-resources/
root@controlplane:~# cat /var/lib/kubelet/
```

```
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
    enabled: false
  webhook:
    cacheTTL: 0s
    enabled: true
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
cgroupDriver: cgroupfs
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
imageMinimumGCAge: 0s
kind: KubeletConfiguration
logging: {}
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
resolvConf: /run/systemd/resolve/resolv.conf
rotateCertificates: true
runtimeRequestTimeout: 0s
shutdownGracePeriod: 0s
shutdownGracePeriodCriticalPods: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s
root@controlplane:~#
```

Run the command ps -aux | grep kubelet and identify the config file - --

config=/var/lib/kubelet/config.yaml. Then check in the config file for staticPodPath.

01 02 03 04 05 06 07 08 09 10

How many pod definition files are present in the manifests folder?

1

4

2

3

```
  webhook:
    cacheTTL: 0s
    enabled: true
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
cgroupDriver: cgroupfs
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
imageMinimumGCAge: 0s
kind: KubeletConfiguration
logging: {}
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
resolvConf: /run/systemd/resolve/resolv.conf
rotateCertificates: true
runtimeRequestTimeout: 0s
shutdownGracePeriod: 0s
shutdownGracePeriodCriticalPods: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s
root@controlplane:~#
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10

What is the docker image used to deploy the kube-api server as a static pod?

k8s.gcr.io/kube-apiserver-amd64:v1.10.3

docker.io/kube-apiserver-amd64:v1.11.3

k8s.gcr.io/kube-apiserver:v1.20.0

```
  webhook:
    cacheTTL: 0s
    enabled: true
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
cgroupDriver: cgroupfs
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
imageMinimumGCAge: 0s
kind: KubeletConfiguration
logging: {}
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
resolvConf: /run/systemd/resolve/resolv.conf
rotateCertificates: true
runtimeRequestTimeout: 0s
shutdownGracePeriod: 0s
shutdownGracePeriodCriticalPods: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# ls /etc/kubernetes/manifests
etcd.yaml  kube-apiserver.yaml  kube-controller-manager.yaml  kube-scheduler.yaml
root@controlplane:~# cat /etc/kubernetes/manifests/kube-apiserver.yaml
```

01 02 03 04 05 06 07 08 09 10

Create a static pod named `static-busybox` that uses the `busybox` image and the command `sleep 1000`

Check

- Name: static-busybox
- Image: busybox

```
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: static-busybox
  name: static-busybox
spec:
  containers:
  - command:
    - sleep
    - "1000"
    image: busybox
    name: static-busybox
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
root@controlplane:~# kubectl run static-busybox --image=busybox --restart=Never --dry-run=client -o yaml
  --command -- sleep 1000
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: static-busybox
  name: static-busybox
spec:
  containers:
  - command:
    - sleep
    - "1000"
    image: busybox
    name: static-busybox
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Never
status: {}
root@controlplane:~# kubectl run static-busybox --image=busybox --restart=Never --dry-run=client -o yaml
  --command -- sleep 1000 > static-busybox.yal
```

```
  restartPolicy: Never
status: {}
root@controlplane:~# kubectl run static-busybox --image=busybox --restart=Never --dry-run=client -o yaml
  --command -- sleep 1000 > static-busybox.yaml
root@controlplane:~# cp static-busybox.yaml /etc/kubernetes/manifests/
root@controlplane:~#
root@controlplane:~# kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
static-busybox-controlplane   0/1     Pending   0          4s
root@controlplane:~#
```

Create a pod definition file in the manifests folder. To do this, run the command:
kubectl run --restart=Never --image=busybox static-busybox --dry-run=client -o yaml --command --sleep 1000 > /etc/kubernetes/manifests/static-busybox.yaml

01 02 03 04 05 06 07 08 09 10

Edit the image on the static pod to use `busybox:1.28.4`

Check

- Name: static-busybox
- Image: busybox:1.28.4

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# vi /etc/kubernetes/manifests/static-busybox.yaml
```

```
root@controlplane:~# vi /etc/kubernetes/manifests/static-busybox.yaml
root@controlplane:~#
root@controlplane:~# kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
static-busybox-controlplane   0/1     Pending   0          5s
root@controlplane:~# kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
static-busybox-controlplane   0/1     Pending   0          7s
root@controlplane:~# kubectl get pods --watch
NAME                      READY   STATUS    RESTARTS   AGE
static-busybox-controlplane   0/1     Pending   0          10s
static-busybox-controlplane   0/1     ErrImagePull   0          12s
static-busybox-controlplane   0/1     ErrImagePull   0          14s
static-busybox-controlplane   0/1     ImagePullBackOff   0          15s
^Croot@controlplane:~# vi /etc/kubernetes/manifests/static-busybox.yaml
root@controlplane:~# kubectl get pods --watch
NAME                      READY   STATUS    RESTARTS   AGE
static-busybox-controlplane   0/1     Pending   0          3s
static-busybox-controlplane   0/1     CreateContainerConfigError   0          5s
static-busybox-controlplane   1/1     Running                      0          8s
^Croot@controlplane:~#
root@controlplane:~#
root@controlplane:~#
```

```
static-busybox-controlplane   0/1     CreateContainerConfigError   0          5s
static-busybox-controlplane   1/1     Running                     0          8s
^Croot@controlplane:~#
root@controlplane:~#
root@controlplane:~#
```

01 02 03 04 05 06 07 08 09 10

We just created a new static pod named static-greenbox. Find it and delete it.

This question is a bit tricky. But if you use the knowledge you gained in the previous questions in this lab, you should be able to find the answer to it.

Check

Static pod deleted

```
root@controlplane:~# kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
static-busybox-controlplane   1/1     Running   0          47s
static-greenbox-node01        1/1     Running   0          16s
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl delete pod static-greenbox-node01
pod "static-greenbox-node01" deleted
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
static-busybox-controlplane   1/1     Running   0          69s
static-greenbox-node01        1/1     Running   0          3s
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# ls /etc/kubernetes/manifests/
etcd.yaml  kube-apiserver.yaml  kube-controller-manager.yaml  kube-scheduler.yaml  static-busybox.yaml
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get nodes
NAME           STATUS   ROLES                  AGE   VERSION
controlplane   Ready    control-plane,master   22m   v1.20.0
node01         Ready    <none>                 21m   v1.20.0
root@controlplane:~#
root@controlplane:~# kubectl get nodes  -o wide
NAME           STATUS   ROLES                  AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE
        KERNEL-VERSION   CONTAINER-RUNTIME
controlplane   Ready    control-plane,master   22m   v1.20.0   10.38.102.6   <none>        Ubuntu 18.04.
5 LTS   5.4.0-1069-gcp   docker://19.3.0
node01         Ready    <none>                 21m   v1.20.0   10.38.102.8   <none>        Ubuntu 18.04.
5 LTS   5.4.0-1069-gcp   docker://19.3.0
root@controlplane:~#
root@controlplane:~# ssh 10.38.102.8
The authenticity of host '10.38.102.8 (10.38.102.8)' can't be established.
ECDSA key fingerprint is SHA256:u9OdNaCmrfntgjseysL28yF4sjvpXR6aev+/cLrHsvU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.38.102.8' (ECDSA) to the list of known hosts.
root@node01:~#
```

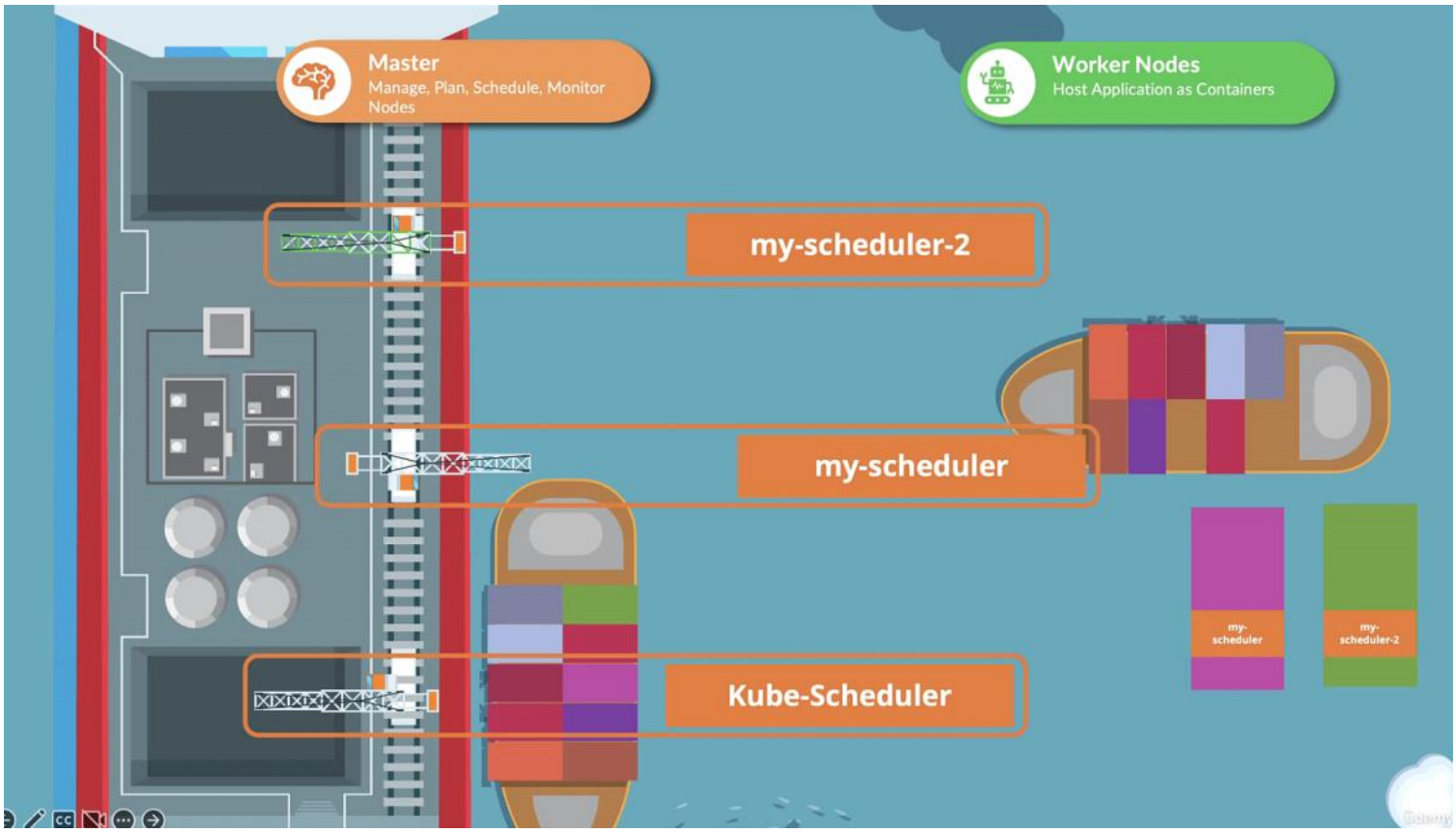
```
root@node01:~#
root@node01:~#
root@node01:~# ls /etc/kubernetes/manifests/
root@node01:~#
root@node01:~#
root@node01:~# cat /var/lib/
apt/           docker/        git/           polkit-1/      systemd/
cni/           docker-engine/ kubelet/       python/        ucf/
containerd/    dockershim/    misc/          rancher/       vim/
dbus/          dpkg/          pam/           sudo/
root@node01:~# cat /var/lib/
apt/           docker/        git/           polkit-1/      systemd/
cni/           docker-engine/ kubelet/       python/        ucf/
containerd/    dockershim/    misc/          rancher/       vim/
dbus/          dpkg/          pam/           sudo/
root@node01:~# cat /var/lib/kubelet/
config.yaml        device-plugins/    pki/               plugins_registry/  pods/
cpu_manager_state  kubeadm-flags.env  plugins/           pod-resources/
root@node01:~# cat /var/lib/kubelet/config.yaml
```

```
shutdownGracePeriodCriticalPods: 0s
staticPodPath: /etc/just-to-mess-with-you
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s
root@node01:~# cd /etc/just-to-mess-with-you
root@node01:/etc/just-to-mess-with-you# ls
greenbox.yaml
root@node01:/etc/just-to-mess-with-you#
root@node01:/etc/just-to-mess-with-you#
root@node01:/etc/just-to-mess-with-you# rm greenbox.yaml
root@node01:/etc/just-to-mess-with-you#
root@node01:/etc/just-to-mess-with-you#
root@node01:/etc/just-to-mess-with-you# ls
root@node01:/etc/just-to-mess-with-you#
root@node01:/etc/just-to-mess-with-you# exit
logout
```

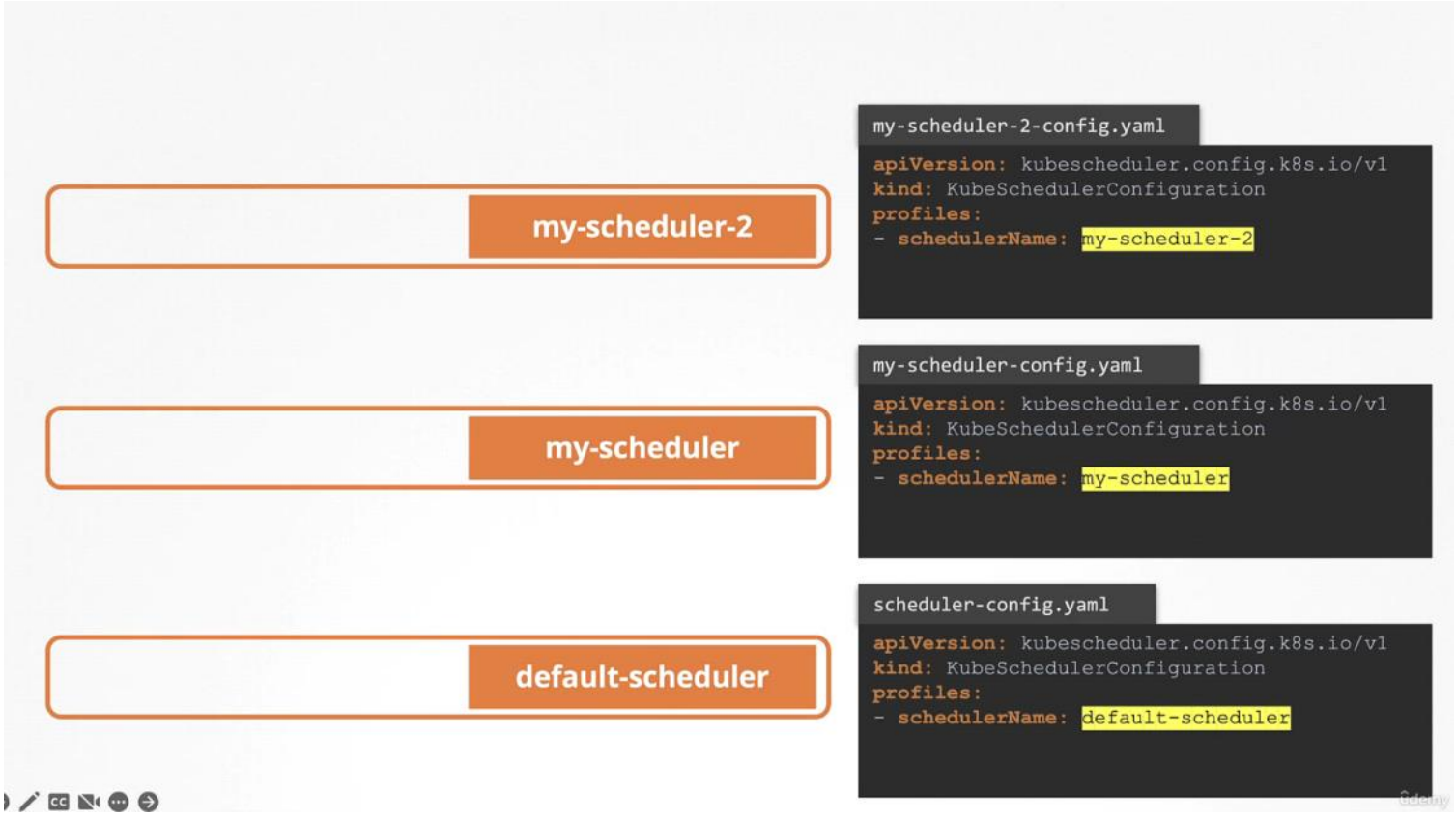
Identify which node the static pod is created on, ssh to the node and delete the pod definition file.
If you don't know the IP of the node, run the kubectl get nodes -o wide command and identify the IP.
Then, SSH to the node using that IP.
For static pod manifest path look at the file /var/lib/kubelet/config.yaml on node01 ---> kubelet configuration

Multiple schedulers

Thursday, June 13, 2024 8:50 AM



s



Deploy Additional Scheduler

```
wget https://storage.googleapis.com/kubernetes-release/release/v1.12.0/bin/linux/amd64/kube-scheduler
```

kube-scheduler.service

```
ExecStart=/usr/local/bin/kube-scheduler \\
  --config=/etc/kubernetes/config/kube-scheduler.yaml
```

my-scheduler-2.service

```
ExecStart=/usr/local/bin/kube-scheduler \\
  --config=/etc/kubernetes/config/my-scheduler-2-config.yaml
```

my-scheduler-2-config.yaml

```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- schedulerName: my-scheduler-2
```

my-scheduler.service

```
ExecStart=/usr/local/bin/kube-scheduler \\
  --config=/etc/kubernetes/config/my-scheduler-config.yaml
```

my-scheduler-config.yaml

```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- schedulerName: my-scheduler
```

Deploy Additional Scheduler as a Pod

my-custom-scheduler.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-custom-scheduler
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-scheduler
    - --address=127.0.0.1
    - --kubeconfig=/etc/kubernetes/scheduler.conf
    - --config=/etc/kubernetes/my-scheduler-config.yaml
    image: k8s.gcr.io/kube-scheduler-amd64:v1.11.3
    name: kube-scheduler
```

my-scheduler-config.yaml

```yaml
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- schedulerName: my-scheduler
leaderElection:
  leaderElect: true
  resourceNamespace: kube-system
  resourceName: lock-object-my-scheduler
```

Use Custom Scheduler

```
kubectl get pods --namespace=kube-system
NAME                              READY   STATUS    RESTARTS   AGE
coredns-78fcdf6894-bk4ml          1/1     Running   0          1h
coredns-78fcdf6894-ppr6m          1/1     Running   0          1h
etcd-master                       1/1     Running   0          1h
kube-apiserver-master             1/1     Running   0          1h
kube-controller-manager-master    1/1     Running   0          1h
kube-proxy-dgbgv                  1/1     Running   0          1h
kube-proxy-fptbr                  1/1     Running   0          1h
kube-scheduler-master             1/1     Running   0          1h
my-custom-scheduler               1/1     Running   0          9s
weave-net-4tfpt                   2/2     Running   1          1h
weave-net-6j6zs                   2/2     Running   1          1h
```

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  -  image: nginx
     name: nginx
   schedulerName: my-custom-scheduler
```

```
kubectl create -f pod-definition.yaml
```

```
kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
nginx    0/1     Pending   0          6s
```

```
kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
nginx    1/1     Running   0          6s
```

you have to add the name of scheduler in po-definnition.yaml as scheduler-name:
the name of schedueler may vary on how you create it, either as a deployment or as a pod.

View Events

```
kubectl get events -o wide
LAST SEEN   COUNT   NAME       KIND  TYPE    REASON     SOURCE                MESSAGE
9s          1       nginx.15   Pod   Normal  Scheduled  my-custom-scheduler   Successfully assigned default/nginx to node01
8s          1       nginx.15   Pod   Normal  Pulling    kubelet, node01       pulling image "nginx"
2s          1       nginx.15   Pod   Normal  Pulled     kubelet, node01       Successfully pulled image "nginx"
2s          1       nginx.15   Pod   Normal  Created    kubelet, node01       Created container
2s          1       nginx.15   Pod   Normal  Started    kubelet, node01       Started container
```

you can view which scheduler picked up the pod by looking at the reason and source.

View Scheduler Logs

To exit full screen, press Esc

```
kubectl logs my-custom-scheduler --name-space=kube-system
```

```
I0204 09:42:25.819338       1 server.go:126] Version: v1.11.3
W0204 09:42:25.822720       1 authorization.go:47] Authorization is disabled
W0204 09:42:25.822745       1 authentication.go:55] Authentication is disabled
I0204 09:42:25.822801       1 insecure_serving.go:47] Serving healthz insecurely on 127.0.0.1:10251
I0204 09:45:14.725407       1 controller_utils.go:1025] Waiting for caches to sync for scheduler controller
I0204 09:45:14.825634       1 controller_utils.go:1032] Caches are synced for scheduler controller
I0204 09:45:14.825814       1 leaderelection.go:185] attempting to acquire leader lease  kube-system/my-custom-scheduler...
I0204 09:45:14.834953       1 leaderelection.go:194] successfully acquired lease kube-system/my-custom-scheduler
```

practice multiple schedulers

Thursday, June 13, 2024 9:44 AM

1 2 3 4 5 6

What is the name of the POD that deploys the default kubernetes scheduler in this environment?

- scheduler
- kube-scheduler
- kube-scheduler-controlplane
- etcd-master

```
root@controlplane ~ ➜  kubectl get pods
No resources found in default namespace.

root@controlplane ~ ➜  kubectl get pods -A
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
kube-system   coredns-64897985d-6qwf5                    1/1     Running   0          6m14s
kube-system   coredns-64897985d-9rdt8                    1/1     Running   0          6m14s
kube-system   etcd-controlplane                          1/1     Running   0          6m25s
kube-system   kube-apiserver-controlplane                1/1     Running   0          6m30s
kube-system   kube-controller-manager-controlplane       1/1     Running   0          6m31s
kube-system   kube-flannel-ds-rl72v                      1/1     Running   0          6m14s
kube-system   kube-proxy-tzwkm                           1/1     Running   0          6m14s
kube-system   kube-scheduler-controlplane                1/1     Running   0          6m25s

root@controlplane ~ ➜
```

1 2 3 4 5 6

What is the image used to deploy the kubernetes scheduler?
Inspect the kubernetes scheduler pod and identify the image

- scheduler:1.0
- kube-scheduler:1.13
- k8s.gcr.io/kube-scheduler:v1.23.0

```
root@controlplane ~ ➜  kubectl get pods
No resources found in default namespace.

root@controlplane ~ ➜  kubectl get pods -A
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
kube-system   coredns-64897985d-6qwf5                    1/1     Running   0          6m14s
kube-system   coredns-64897985d-9rdt8                    1/1     Running   0          6m14s
kube-system   etcd-controlplane                          1/1     Running   0          6m25s
kube-system   kube-apiserver-controlplane                1/1     Running   0          6m30s
kube-system   kube-controller-manager-controlplane       1/1     Running   0          6m31s
kube-system   kube-flannel-ds-rl72v                      1/1     Running   0          6m14s
kube-system   kube-proxy-tzwkm                           1/1     Running   0          6m14s
kube-system   kube-scheduler-controlplane                1/1     Running   0          6m25s

root@controlplane ~ ➜  kubectl describe pod kube-scheduler-controlplane -n kube-system
```

1 2 3 4 5 6

We have already created the `ServiceAccount` and `ClusterRoleBinding` that our custom scheduler will make use of.
Checkout the following kubernetes objects:

`ServiceAccount` : my-scheduler (kube-system namespace)
`ClusterRoleBinding` : my-scheduler-as-kube-scheduler
`ClusterRoleBinding` : my-scheduler-as-volume-scheduler

Ok

```
root@controlplane ~ ➜  kubectl get sa my-scheduler
Error from server (NotFound): serviceaccounts "my-scheduler" not found

root@controlplane ~ ✖ kubectl get sa my-scheduler -n kube-system
NAME           SECRETS   AGE
my-scheduler   1         26s

root@controlplane ~ ➜
```

1 2 3 4 5 6

Let's create a configmap that the new scheduler will employ using the concept of `ConfigMap as a volume`.
Create a configmap with name `my-scheduler-config` using the content of file `/root/my-scheduler-config.yaml`

Check

- ConfigMap my-scheduler-config created ?

```
root@controlplane ~ ➜  cat /root/my-scheduler-config.yaml
apiVersion: kubescheduler.config.k8s.io/v1beta2
kind: KubeSchedulerConfiguration
profiles:
  - schedulerName: my-scheduler
leaderElection:
  leaderElect: false

root@controlplane ~ ➜

root@controlplane ~ ➜

root@controlplane ~ ➜  kubectl create configmap --name my-scheduler-config --from-file=/root/my-schedule
r-config.yaml -n kube-system
Error: unknown flag: --name
See 'kubectl create configmap --help' for usage.

root@controlplane ~ ✖ kubectl create configmap  my-scheduler-config --from-file=/root/my-scheduler-confi
g.yaml -n kube-system
configmap/my-scheduler-config created

root@controlplane ~ ➜

root@controlplane ~ ➜
```

1 2 3 4 5 6

Deploy an additional scheduler to the cluster following the given specification.

Use the manifest file provided at `/root/my-scheduler.yaml`. Use the same image as used by the default kubernetes scheduler.

Check

- Name: my-scheduler
- Status: Running
- Correct image used?

```
NAMESPACE     NAME                                      READY   STATUS    RESTARTS   AGE
kube-system   coredns-64897985d-6qwf5                   1/1     Running   0          10m
kube-system   coredns-64897985d-9rdt8                   1/1     Running   0          10m
kube-system   etcd-controlplane                         1/1     Running   0          10m
kube-system   kube-apiserver-controlplane               1/1     Running   0          10m
kube-system   kube-controller-manager-controlplane      1/1     Running   0          10m
kube-system   kube-flannel-ds-rl72v                     1/1     Running   0          10m
kube-system   kube-proxy-tzwkm                          1/1     Running   0          10m
kube-system   kube-scheduler-controlplane               1/1     Running   0          10m

root@controlplane ~ ➜  kubectl describe pod kube-scheduler-controlplane -n kube-system | grep Image
    Image:         k8s.gcr.io/kube-scheduler:v1.23.0
    Image ID:      docker-pullable://k8s.gcr.io/kube-scheduler@sha256:af8166ce28baa7cb902a2c0d16da865d5d
7c892fe1b41187fd4be78ec6291c23

root@controlplane ~ ➜  vi my-scheduler.yaml

root@controlplane ~ ➜  kubectl create -f my-scheduler.yaml
pod/my-scheduler created

root@controlplane ~ ➜

root@controlplane ~ ➜

root@controlplane ~ ➜  kubectl get pods
No resources found in default namespace.

root@controlplane ~ ➜  kubectl get pods -n kube-system
NAME                                      READY   STATUS    RESTARTS   AGE
coredns-64897985d-6qwf5                   1/1     Running   0          11m
coredns-64897985d-9rdt8                   1/1     Running   0          11m
etcd-controlplane                         1/1     Running   0          11m
kube-apiserver-controlplane               1/1     Running   0          11m
kube-controller-manager-controlplane      1/1     Running   0          11m
kube-flannel-ds-rl72v                     1/1     Running   0          11m
kube-proxy-tzwkm                          1/1     Running   0          11m
kube-scheduler-controlplane               1/1     Running   0          11m
my-scheduler                              1/1     Running   0          7s

root@controlplane ~ ➜
```

1 2 3 4 5 6

A POD definition file is given. Use it to create a POD with the new custom scheduler.

File is located at `/root/nginx-pod.yaml`

Check

- Uses custom scheduler
- Status: Running

```
root@controlplane ~ ➜

root@controlplane ~ ➜

root@controlplane ~ ➜  kubectl get pods
No resources found in default namespace.
```

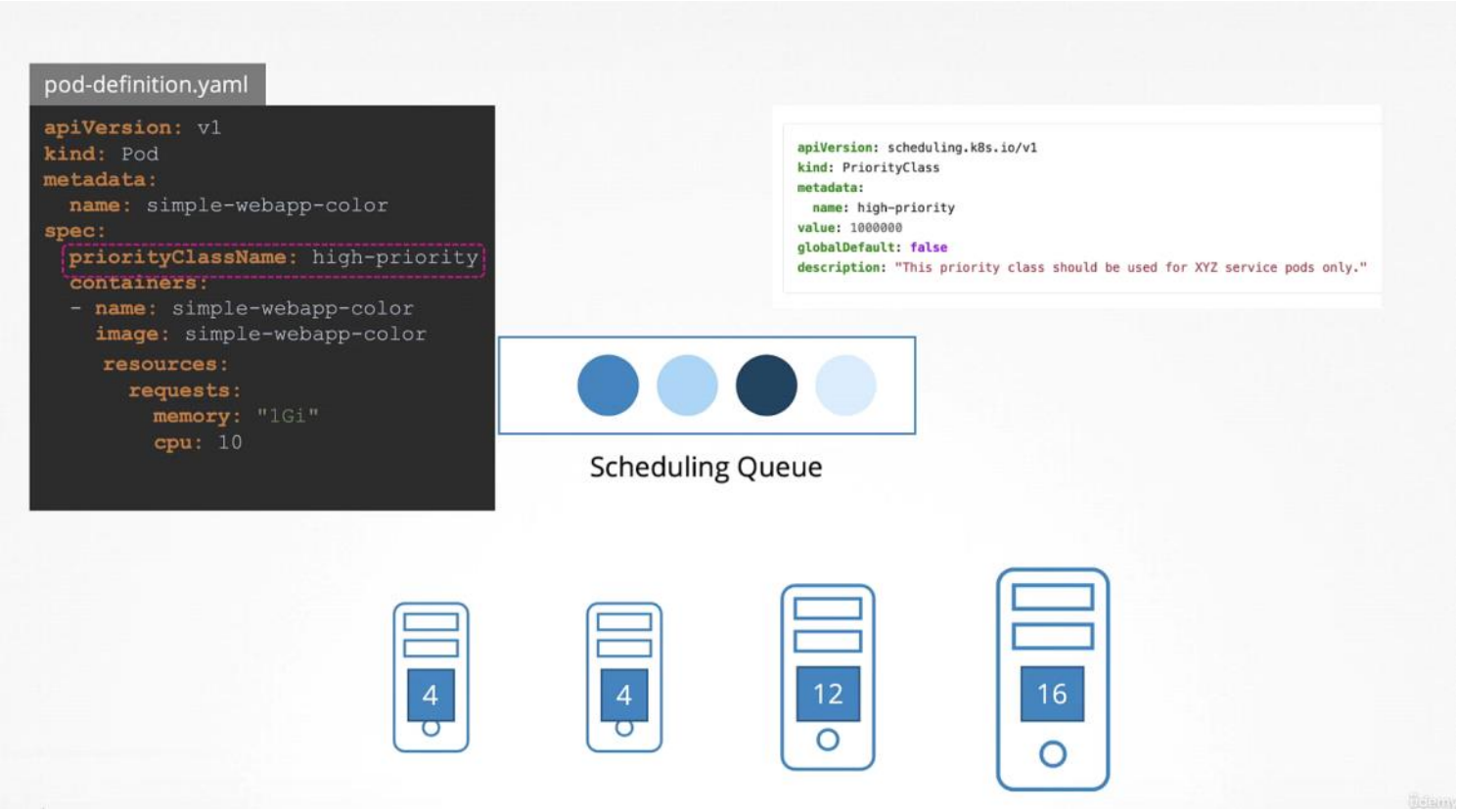
[redacted]

```
root@controlplane ~ ➜

root@controlplane ~ ➜

root@controlplane ~ ➜  ls
my-scheduler-config.yaml  my-scheduler.yaml  nginx-pod.yaml

root@controlplane ~ ➜  cat nginx-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx

root@controlplane ~ ➜  vi nginx-pod.yaml

root@controlplane ~ ➜
```

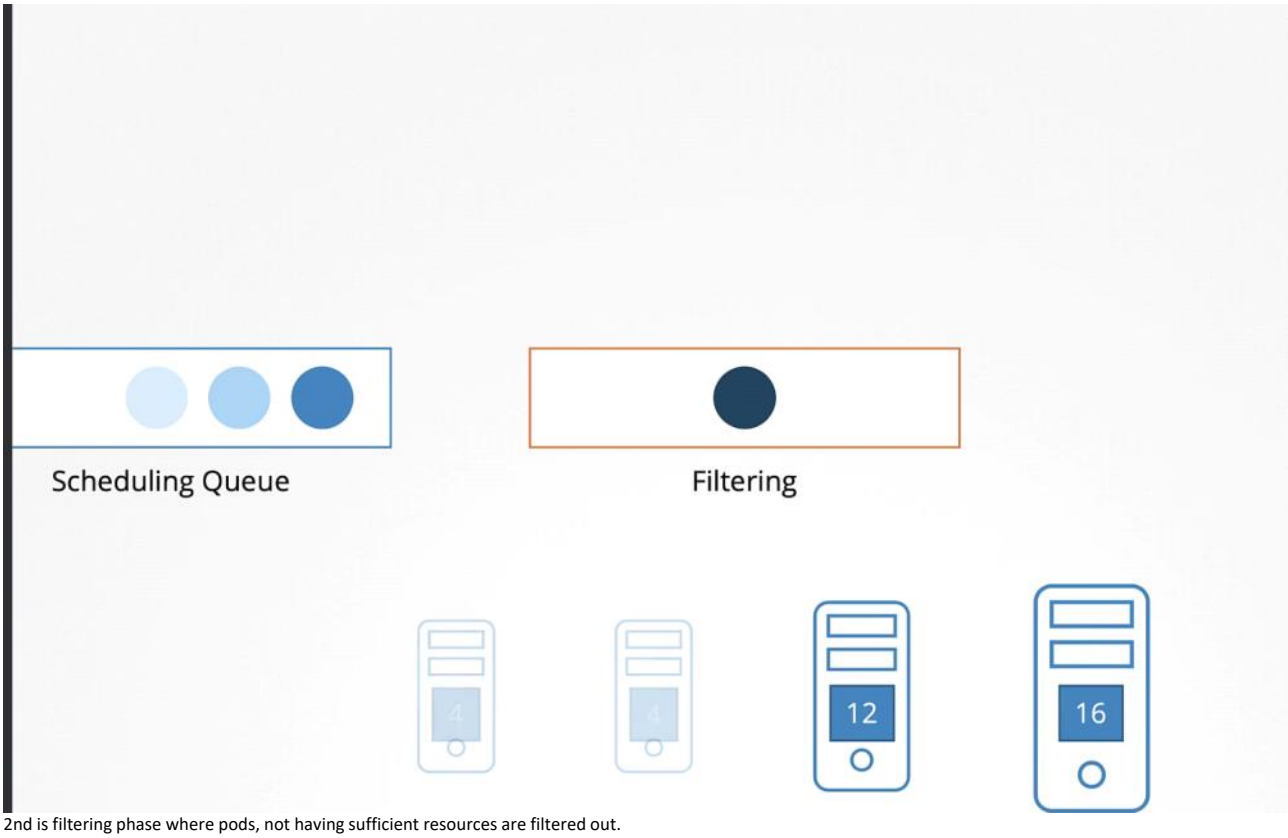
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  schedulerName: my-schedu
  containers:
  - image: nginx
    name: nginx
~
~
~
~
~
~
~
```

Scheduler Profiles

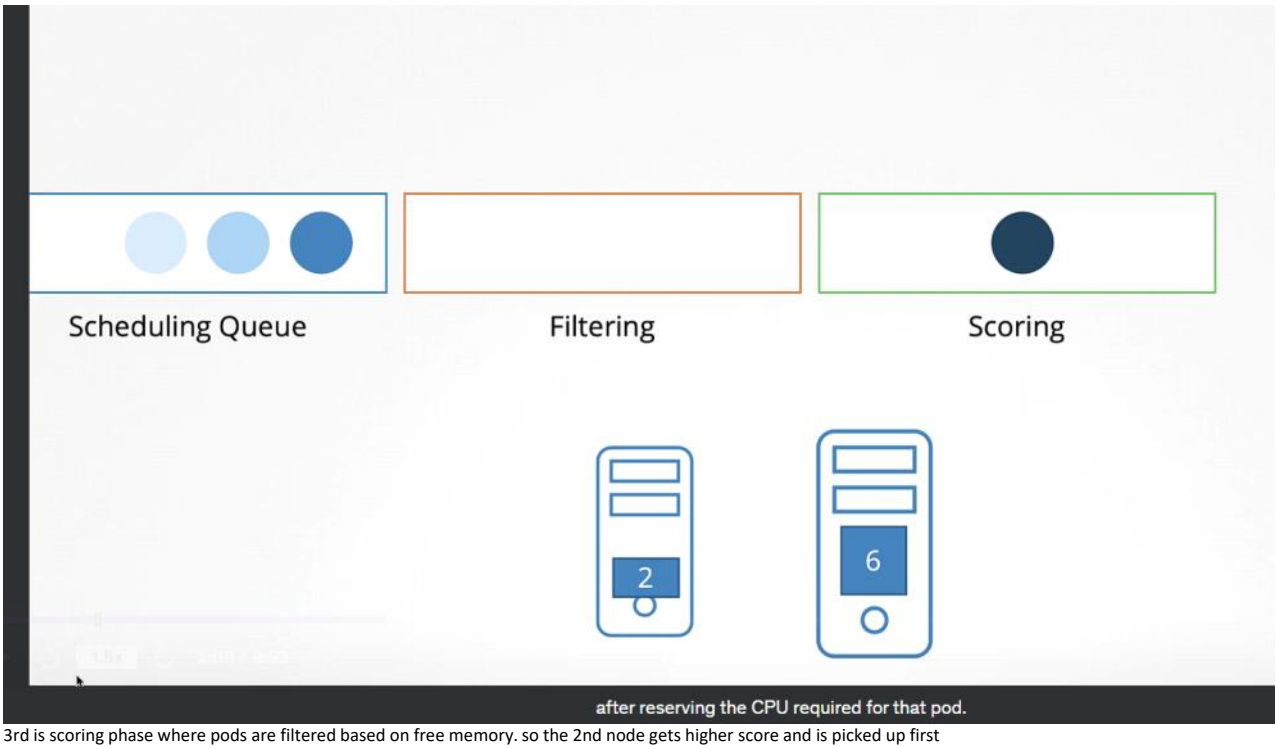
Thursday, June 13, 2024 10:10 AM



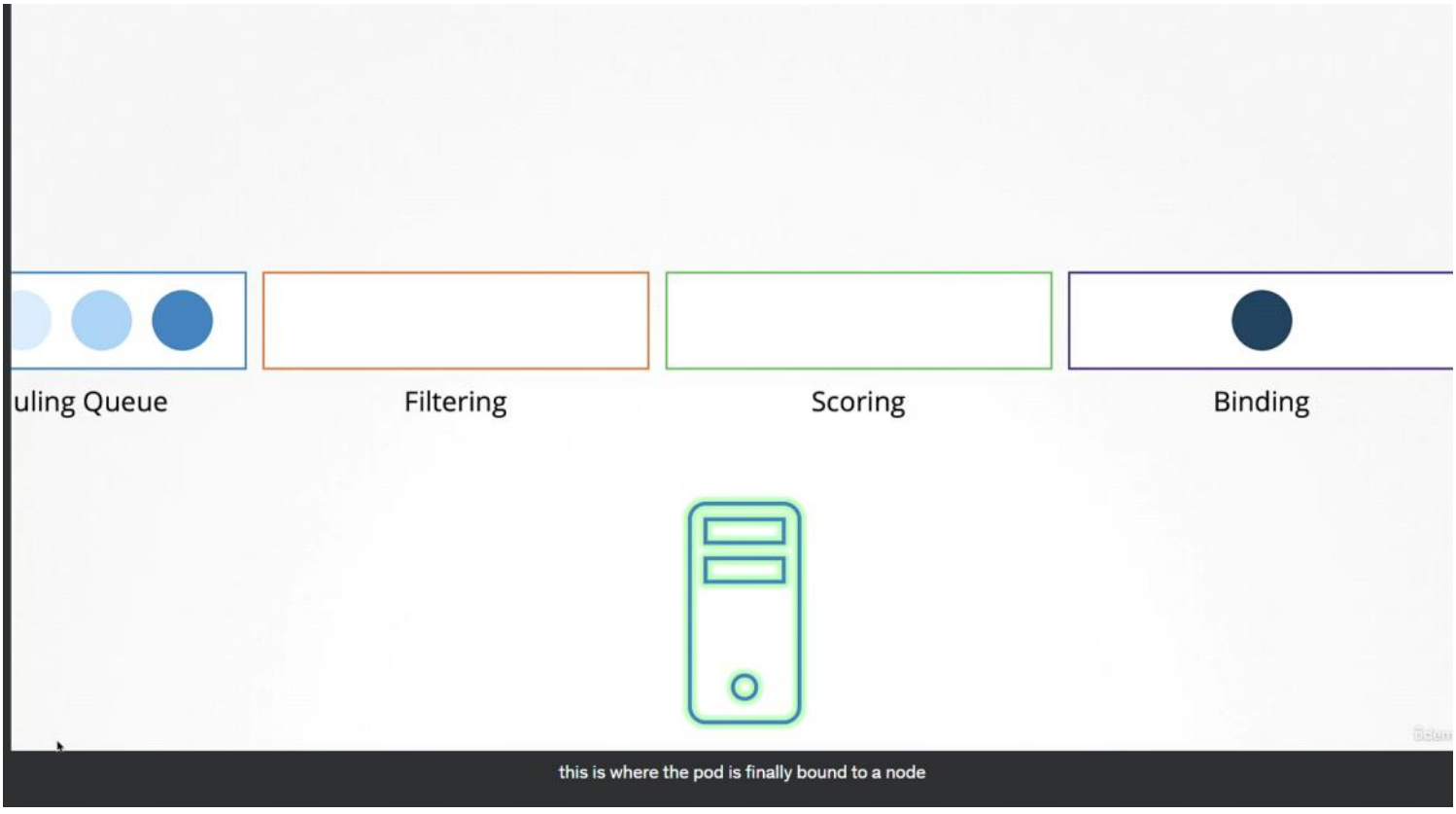
1st the que is based on priority high-prioruty gets scheduled first



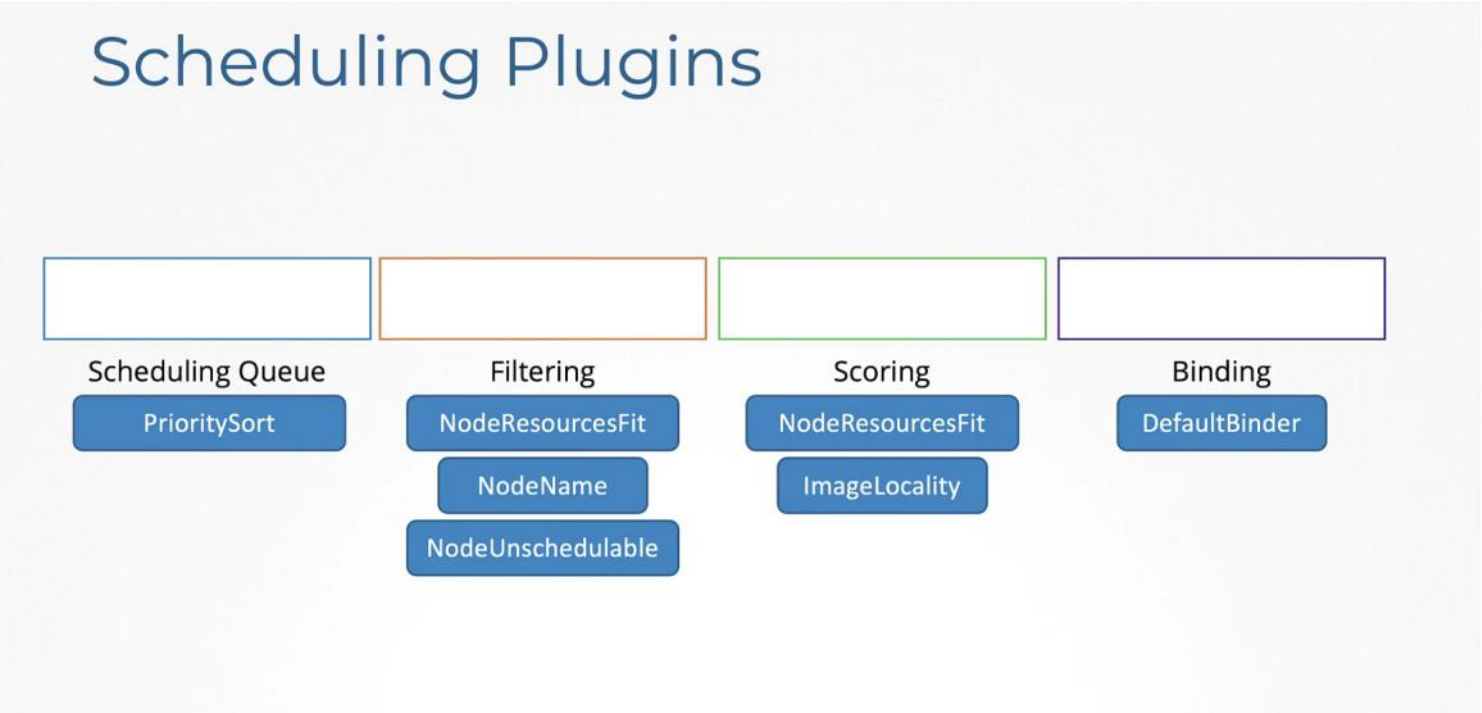
2nd is filtering phase where pods, not having sufficient resources are filtered out.



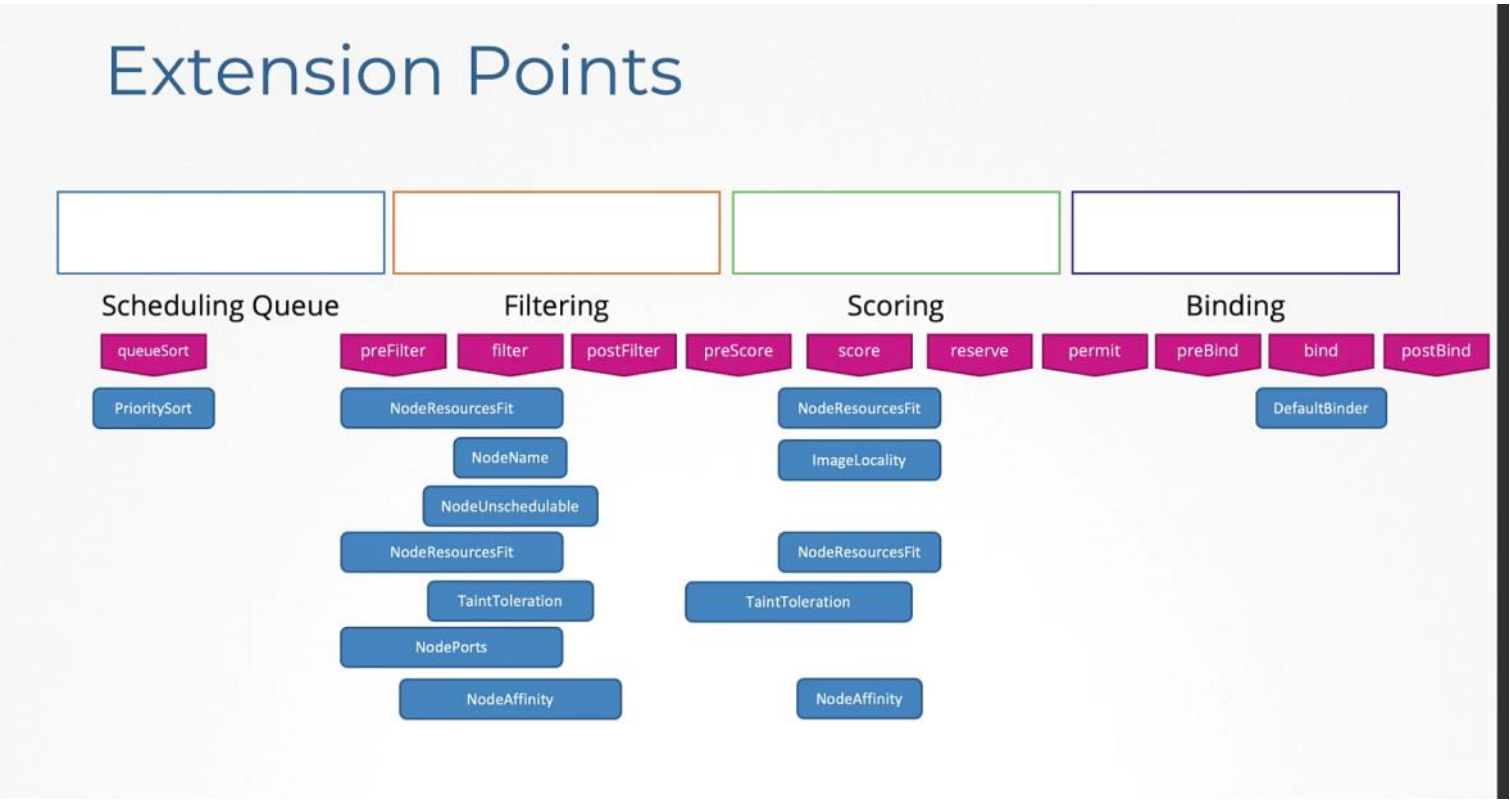
3rd is scoring phase where pods are filtered based on free memory. so the 2nd node gets higher score and is picked up first



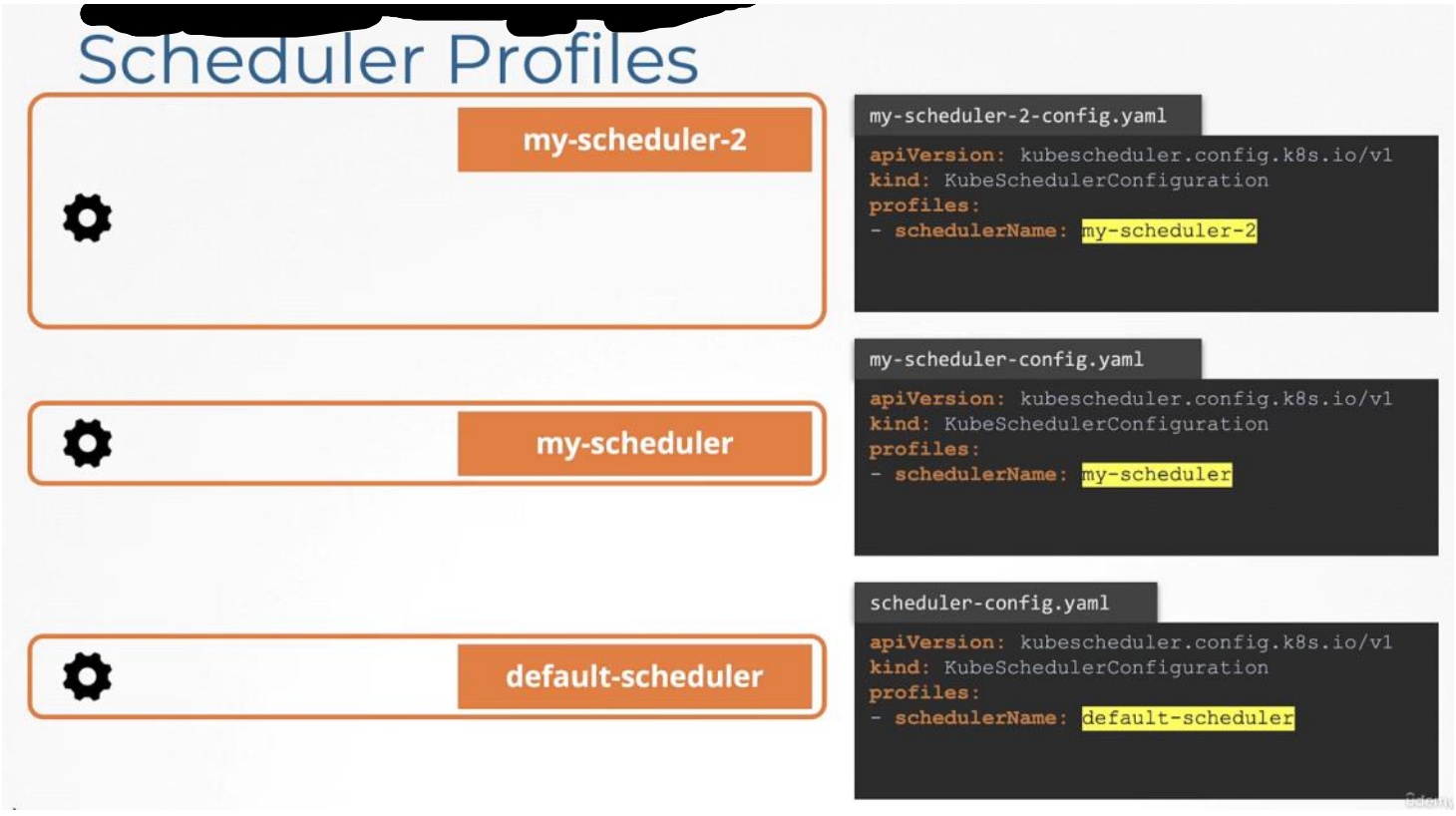
last phase is where pod is bound to a node with higher score.

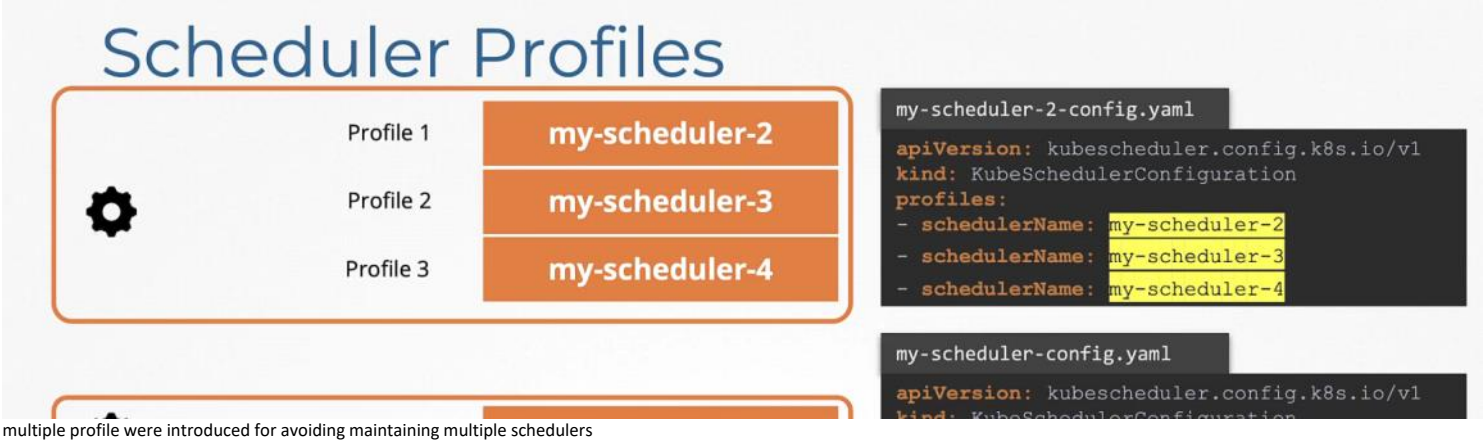


marked I blue are the plugijngs used during scheduling

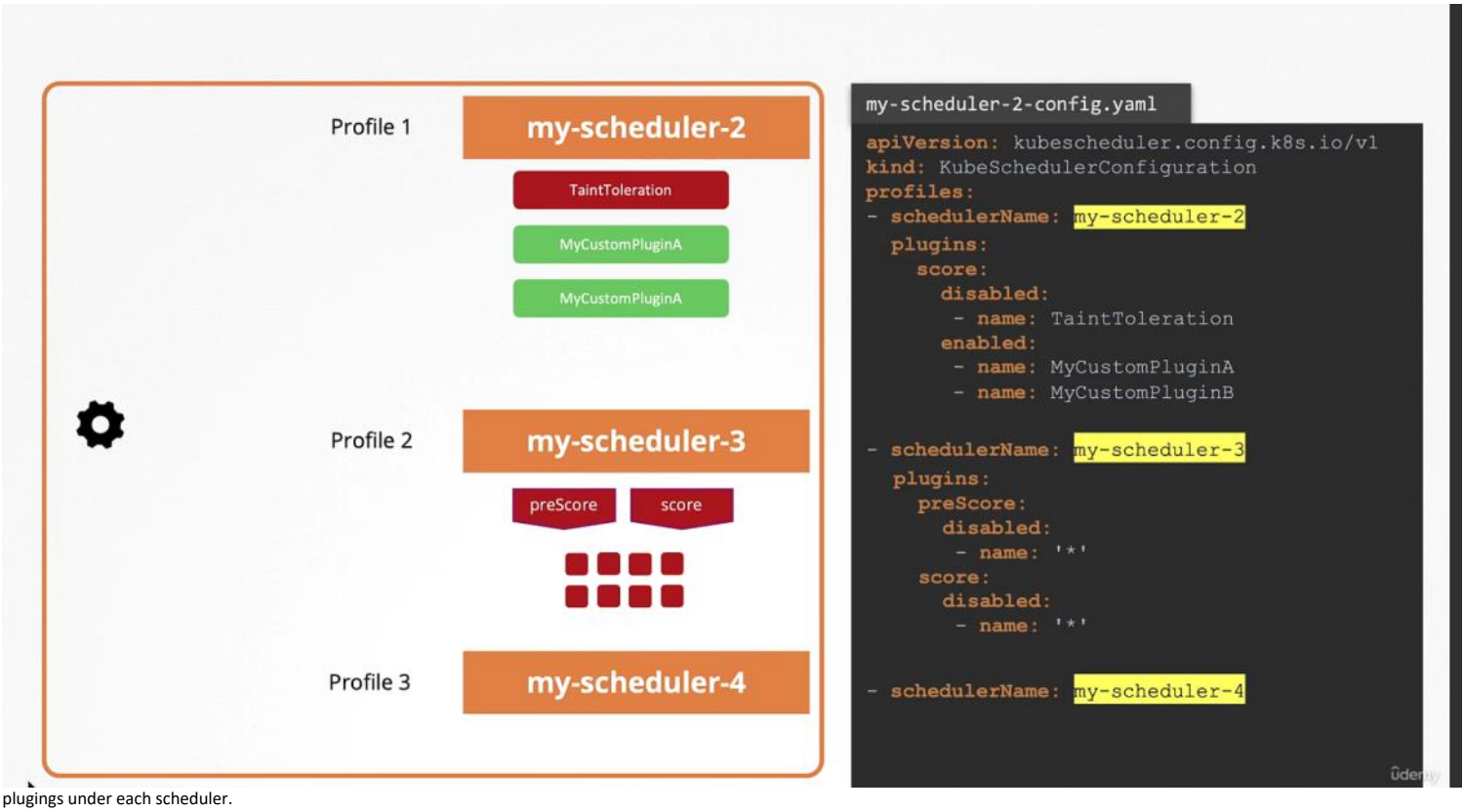


[redacted]





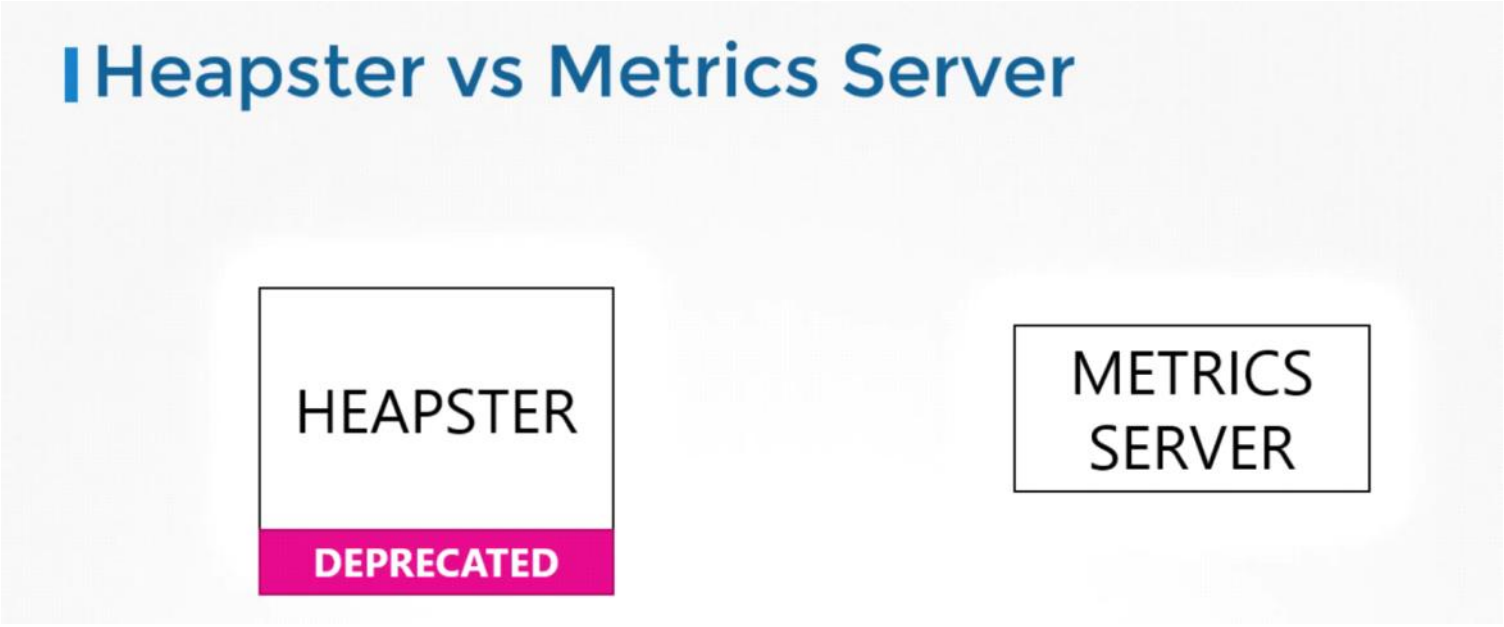
multiple profile were introduced for avoiding maintaining multiple schedulers

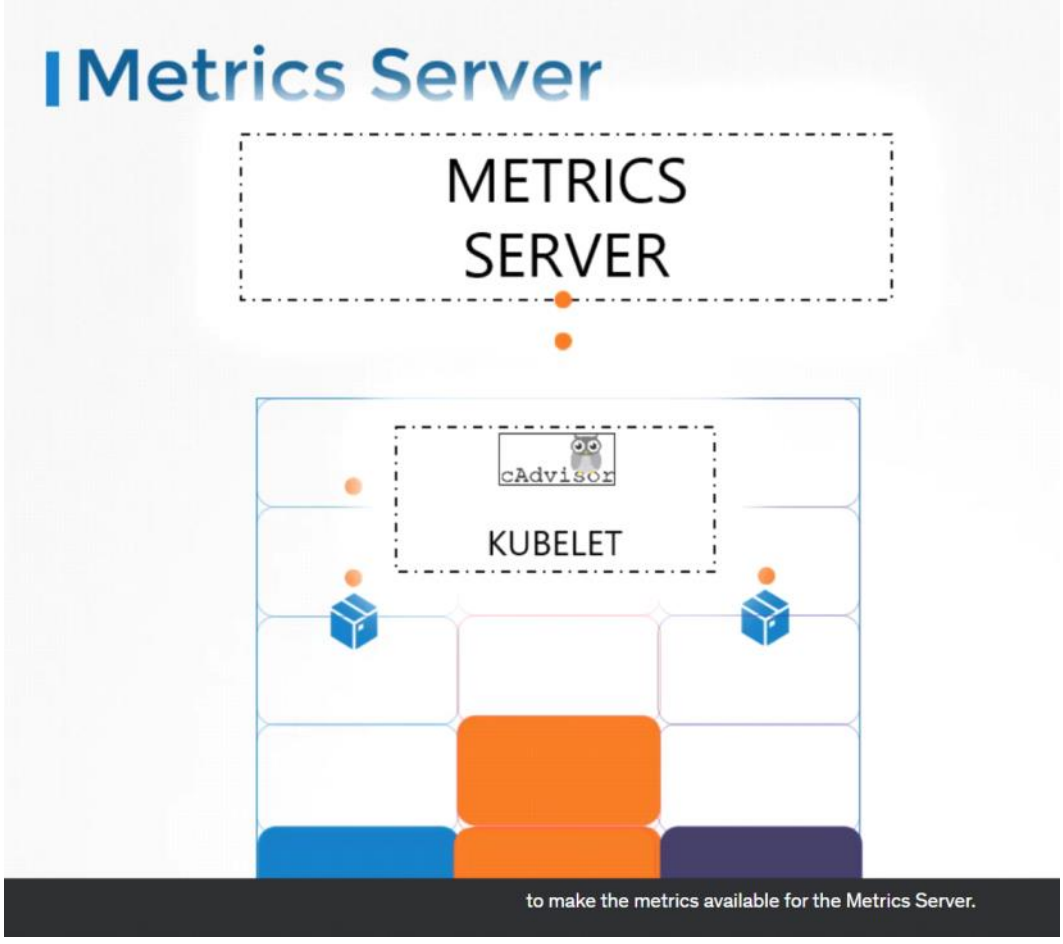


plugings under each scheduler.

Monitoring Cluster Components

Friday, June 14, 2024 8:42 AM





Metrics Server – Getting Started

minikube

```
minikube addons enable metrics-server
```

others

```
git clone https://github.com/kubernetes-incubator/metrics-server.git
```

```
kubectl create -f deploy/1.8+/
clusterrolebinding "metrics-server:system:auth-delegator" created
rolebinding "metrics-server-auth-reader" created
apiservice "v1beta1.metrics.k8s.io" created
serviceaccount "metrics-server" created
deployment "metrics-server" created
service "metrics-server" created
clusterrole "system:metrics-server" created
clusterrolebinding "system:metrics-server" created
```

View

```
kubectl top node
NAME          CPU(cores)   CPU%      MEMORY(bytes)   MEMORY%
kubemaster    166m         8%        1337Mi          70%
kubenode1     36m          1%        1046Mi          55%
kubenode2     39m          1%        1048Mi          55%
```

```
kubectl top pod
NAME          CPU(cores)   CPU%      MEMORY(bytes)   MEMORY%
nginx         166m         8%        1337Mi          70%
redis         36m          1%        1046Mi          55%
```

Practice Monitoring Cluster Components

Friday, June 14, 2024 8:52 AM

01 02 03 04 05 06 07 08

We have deployed a few PODs running workloads. Inspect them.

Wait for the pods to be ready before proceeding to the next question.

Ok

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get pods
NAME       READY   STATUS              RESTARTS   AGE
elephant   1/1     Running             0          20s
lion       1/1     Running             0          20s
rabbit     0/1     ContainerCreating   0          20s
root@controlplane:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
elephant   1/1     Running   0          27s
lion       1/1     Running   0          27s
rabbit     1/1     Running   0          27s
root@controlplane:~#
```

01 02 03 04 05 06 07 08

Let us deploy metrics-server to monitor the PODs and Nodes. Pull the git repository for the deployment files.

Run: `git clone https://github.com/kodekloudhub/kubernetes-metrics-server.git`

Ok

```
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# kubectl get pods
NAME       READY   STATUS              RESTARTS   AGE
elephant   1/1     Running             0          20s
lion       1/1     Running             0          20s
rabbit     0/1     ContainerCreating   0          20s
root@controlplane:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
elephant   1/1     Running   0          27s
lion       1/1     Running   0          27s
rabbit     1/1     Running   0          27s
root@controlplane:~# git clone https://github.com/kodekloudhub/kubernetes-metrics-server.git
Cloning into 'kubernetes-metrics-server'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 24 (delta 4), reused 0 (delta 0), pack-reused 12
Unpacking objects: 100% (24/24), done.
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# ls
kubernetes-metrics-server  sample.yaml
root@controlplane:~# cd kubernetes-metrics-server/
root@controlplane:~/kubernetes-metrics-server# ls
README.md                    auth-reader.yaml              metrics-server-service.yaml
aggregated-metrics-reader.yaml  metrics-apiservice.yaml    resource-reader.yaml
auth-delegator.yaml          metrics-server-deployment.yaml
root@controlplane:~/kubernetes-metrics-server#
```

01 02 03 04 05 06 07 08

Deploy the metrics-server by creating all the components downloaded.

Run the `kubectl create -f .` command from within the downloaded repository.

Complete

Next

```
root@controlplane:~#
root@controlplane:~# kubectl get pods
NAME       READY   STATUS              RESTARTS   AGE
elephant   1/1     Running             0          20s
lion       1/1     Running             0          20s
rabbit     0/1     ContainerCreating   0          20s
root@controlplane:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
elephant   1/1     Running   0          27s
lion       1/1     Running   0          27s
rabbit     1/1     Running   0          27s
root@controlplane:~# git clone https://github.com/kodekloudhub/kubernetes-metrics-server.git
Cloning into 'kubernetes-metrics-server'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 24 (delta 4), reused 0 (delta 0), pack-reused 12
Unpacking objects: 100% (24/24), done.
root@controlplane:~#
root@controlplane:~#
root@controlplane:~# ls
kubernetes-metrics-server  sample.yaml
root@controlplane:~# cd kubernetes-metrics-server/
root@controlplane:~/kubernetes-metrics-server# ls
README.md                      auth-reader.yaml                metrics-server-service.yaml
aggregated-metrics-reader.yaml metrics-apiservice.yaml         resource-reader.yaml
auth-delegator.yaml            metrics-server-deployment.yaml
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server# kubectl create -f .
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
serviceaccount/metrics-server created
deployment.apps/metrics-server created
service/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
root@controlplane:~/kubernetes-metrics-server#
```

01 02 03 04 05 06 07 08

It takes a few minutes for the metrics server to start gathering data.

Run the `kubectl top node` command and wait for a valid output.

```
root@controlplane:~/kubernetes-metrics-server# kubectl top node
NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
controlplane   470m         1%     1252Mi          0%
node01         57m          0%     349Mi           0%
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server# kubectl top pod
NAME       CPU(cores)   MEMORY(bytes)
elephant   20m          32Mi
lion       1m           18Mi
rabbit     131m         252Mi
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
```

01 02 03 04 05 06 07 08

Identify the node that consumes the `most` CPU.

node01

node02

node03

controlplane

```
root@controlplane:~/kubernetes-metrics-server# kubectl top node
NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
controlplane   470m         1%     1252Mi          0%
node01         57m          0%     349Mi           0%
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server# kubectl top pod
NAME       CPU(cores)   MEMORY(bytes)
elephant   20m          32Mi
lion       1m           18Mi
rabbit     131m         252Mi
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
```

01 02 03 04 05 06 07 08

Identify the node that consumes the `most` Memory.

controlplane

node03

node01

node02

```
root@controlplane:~/kubernetes-metrics-server# kubectl top node
NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
controlplane   470m         1%     1252Mi          0%
node01         57m          0%     349Mi           0%
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server# kubectl top pod
NAME       CPU(cores)   MEMORY(bytes)
elephant   20m          32Mi
lion       1m           18Mi
rabbit     131m         252Mi
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
```

01 02 03 04 05 06 07 08

Identify the POD that consumes the `most` Memory.

tiger

rabbit

lion

elephant

```
root@controlplane:~/kubernetes-metrics-server# kubectl top node
NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
controlplane   470m         1%     1252Mi          0%
node01         57m          0%     349Mi           0%
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server# kubectl top pod
NAME       CPU(cores)   MEMORY(bytes)
elephant   20m          32Mi
lion       1m           18Mi
rabbit     131m         252Mi
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
```

01 02 03 04 05 06 07 08

Identify the POD that consumes the `least` CPU.

lion

tiger

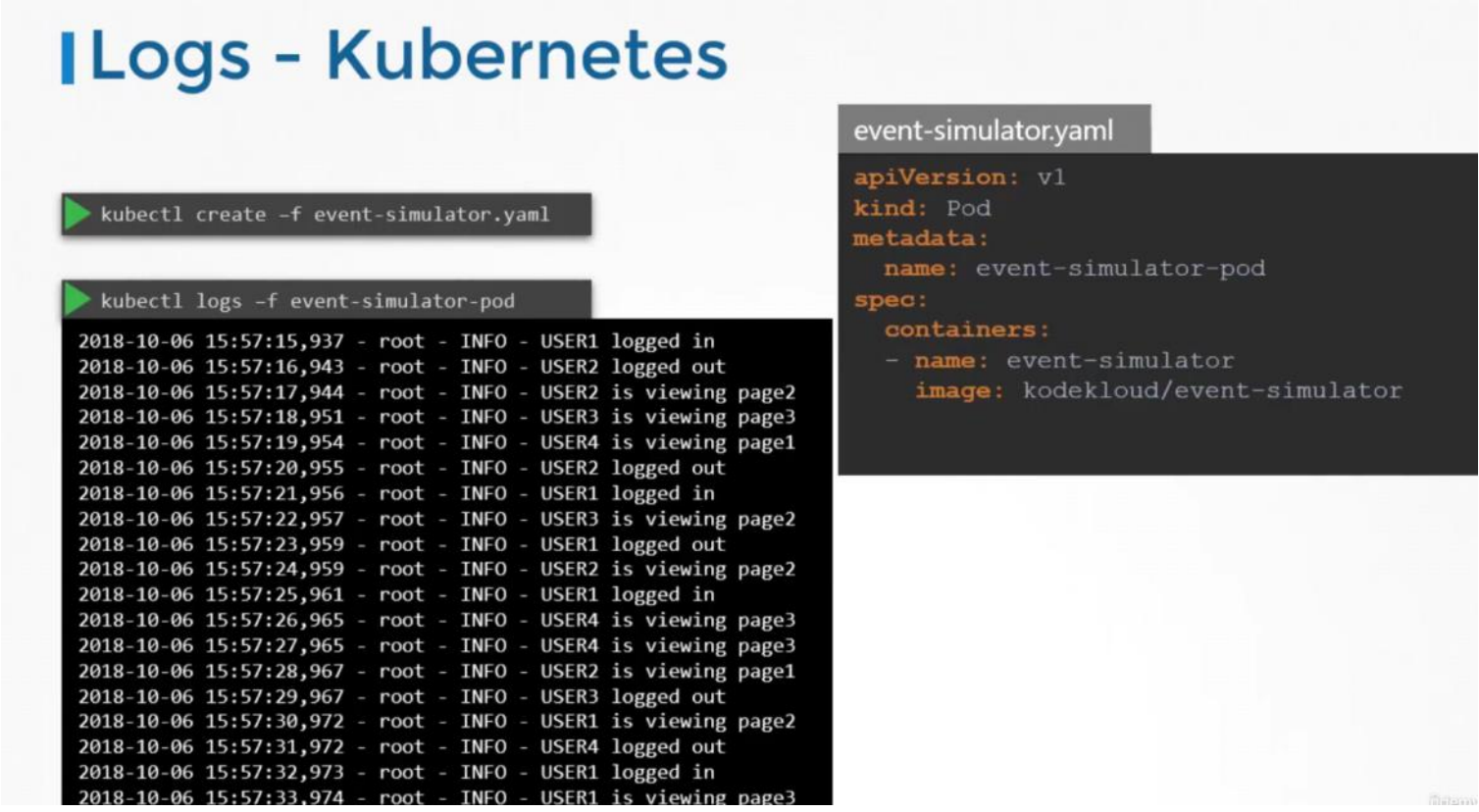
elephant

rabbit

```
root@controlplane:~/kubernetes-metrics-server# kubectl top node
NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
controlplane   470m         1%     1252Mi          0%
node01         57m          0%     349Mi           0%
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server# kubectl top pod
NAME       CPU(cores)   MEMORY(bytes)
elephant   20m          32Mi
lion       1m           18Mi
rabbit     131m         252Mi
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
root@controlplane:~/kubernetes-metrics-server#
```

Managing application logs

Friday, June 14, 2024 9:06 AM



Logs - Kubernetes

```
kubectl logs -f event-simulator-pod event-simulator
```

```
2018-10-06 15:57:15,937 - root - INFO - USER1 logged in
2018-10-06 15:57:16,943 - root - INFO - USER2 logged out
2018-10-06 15:57:17,944 - root - INFO - USER2 is viewing page2
2018-10-06 15:57:18,951 - root - INFO - USER3 is viewing page3
2018-10-06 15:57:19,954 - root - INFO - USER4 is viewing page1
2018-10-06 15:57:20,955 - root - INFO - USER2 logged out
2018-10-06 15:57:21,956 - root - INFO - USER1 logged in
2018-10-06 15:57:22,957 - root - INFO - USER3 is viewing page2
2018-10-06 15:57:23,959 - root - INFO - USER1 logged out
2018-10-06 15:57:24,959 - root - INFO - USER2 is viewing page2
2018-10-06 15:57:25,961 - root - INFO - USER1 logged in
2018-10-06 15:57:26,965 - root - INFO - USER4 is viewing page3
2018-10-06 15:57:27,965 - root - INFO - USER4 is viewing page3
2018-10-06 15:57:28,967 - root - INFO - USER2 is viewing page1
2018-10-06 15:57:29,967 - root - INFO - USER3 logged out
```

event-simulator.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: event-simulator-pod
spec:
  containers:
  - name: event-simulator
    image: kodekloud/event-simulator
  - name: image-processor
    image: some-image-processor
```

you need to specify name of container if there are multiple to see the logs.

Practice Logging

Friday, June 14, 2024 9:14 AM

1 2 3 4

We have deployed a POD hosting an application. Inspect it. Wait for it to start.

Ok

```
controlplane ~ ➜  kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
webapp-1   1/1     Running   0          51s

controlplane ~ ➜
```

1 2 3 4

A user - USER5 - has expressed concerns accessing the application. Identify the cause of the issue.
Inspect the logs of the POD

Item Out of Stock

Account Locked due to Many Failed Attempts

Account does not exist

Application Crashed

```
controlplane ~ ➜  kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
webapp-1   1/1     Running   0          51s

controlplane ~ ➜  kubectl logs webapp-1
```

```
[2022-04-16 19:55:56,920] INFO in event-simulator: USER1 is viewing page3
[2022-04-16 19:55:57,921] INFO in event-simulator: USER2 is viewing page2
[2022-04-16 19:55:58,923] WARNING in event-simulator: USER5 Failed to Login as the account is locked due
 to MANY FAILED ATTEMPTS.
[2022-04-16 19:55:58,923] WARNING in event-simulator: USER7 Order failed as the item is OUT OF STOCK.
[2022-04-16 19:55:58,923] INFO in event-simulator: USER2 is viewing page2
[2022-04-16 19:55:59,925] INFO in event-simulator: USER3 logged out
[2022-04-16 19:56:00,926] INFO in event-simulator: USER2 is viewing page2
[2022-04-16 19:56:01,947] INFO in event-simulator: USER1 is viewing page2
[2022-04-16 19:56:02,949] INFO in event-simulator: USER4 logged in
[2022-04-16 19:56:03,950] WARNING in event-simulator: USER5 Failed to Login as the account is locked due
 to MANY FAILED ATTEMPTS.
[2022-04-16 19:56:03,950] INFO in event-simulator: USER3 logged out
[2022-04-16 19:56:04,950] INFO in event-simulator: USER1 logged out
[2022-04-16 19:56:05,951] INFO in event-simulator: USER2 logged out
[2022-04-16 19:56:06,952] WARNING in event-simulator: USER7 Order failed as the item is OUT OF STOCK.
[2022-04-16 19:56:06,953] INFO in event-simulator: USER2 is viewing page2
[2022-04-16 19:56:07,954] INFO in event-simulator: USER2 is viewing page2
[2022-04-16 19:56:08,955] WARNING in event-simulator: USER5 Failed to Login as the account is locked due
 to MANY FAILED ATTEMPTS.
[2022-04-16 19:56:08,955] INFO in event-simulator: USER4 logged in
[2022-04-16 19:56:09,957] INFO in event-simulator: USER4 logged out
[2022-04-16 19:56:10,958] INFO in event-simulator: USER2 is viewing page1
[2022-04-16 19:56:11,959] INFO in event-simulator: USER1 is viewing page3
[2022-04-16 19:56:12,961] INFO in event-simulator: USER3 logged in
[2022-04-16 19:56:13,961] WARNING in event-simulator: USER5 Failed to Login as the account is locked due
 to MANY FAILED ATTEMPTS.
[2022-04-16 19:56:13,962] INFO in event-simulator: USER3 is viewing page2
[2022-04-16 19:56:14,963] WARNING in event-simulator: USER7 Order failed as the item is OUT OF STOCK.
[2022-04-16 19:56:14,963] INFO in event-simulator: USER1 is viewing page3
```

1 2 3 4

We have deployed a new POD - webapp-2 - hosting an application. Inspect it. Wait for it to start.

Ok

```
controlplane ~ ➜  kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
webapp-1   1/1     Running   0          110s
webapp-2   2/2     Running   0          9s

controlplane ~ ➜
```

1 2 3 4

A user is reporting issues while trying to purchase an item. Identify the user and the cause of the issue. Inspect the logs of the webapp in the POD

- USER1 - Item Out of Stock
- USER30 - Item Out of Stock
- USER4 - Account does not exist
- USER2 - Application Crashed

```
controlplane ~ ➜  kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
webapp-1   1/1     Running   0          110s
webapp-2   2/2     Running   0          9s

controlplane ~ ➜  kubectl logs webapp-2
error: a container name must be specified for pod webapp-2, choose one of: [simple-webapp db]

controlplane ~ ✖ kubectl logs webapp-2 simple-webapp
```

1 2 3 4

A user is reporting issues while trying to purchase an item. Identify the user and the cause of the issue. Inspect the logs of the webapp in the POD

- USER1 - Item Out of Stock
- USER30 - Item Out of Stock
- USER4 - Account does not exist
- USER2 - Application Crashed

```
[2022-04-16 19:57:24,773] INFO in event-simulator: USER2 logged in
[2022-04-16 19:57:25,774] INFO in event-simulator: USER1 logged in
[2022-04-16 19:57:26,775] WARNING in event-simulator: USER5 Failed to Login as the account is locked due
 to MANY FAILED ATTEMPTS.
[2022-04-16 19:57:26,776] INFO in event-simulator: USER4 is viewing page1
[2022-04-16 19:57:27,777] INFO in event-simulator: USER1 is viewing page1
[2022-04-16 19:57:28,779] WARNING in event-simulator: USER30 Order failed as the item is OUT OF STOCK.
[2022-04-16 19:57:28,779] INFO in event-simulator: USER4 is viewing page3
[2022-04-16 19:57:29,780] INFO in event-simulator: USER1 logged in
[2022-04-16 19:57:30,781] INFO in event-simulator: USER3 logged in
[2022-04-16 19:57:31,783] WARNING in event-simulator: USER5 Failed to Login as the account is locked due
 to MANY FAILED ATTEMPTS.
[2022-04-16 19:57:31,783] INFO in event-simulator: USER2 logged in
[2022-04-16 19:57:32,784] INFO in event-simulator: USER3 logged out
[2022-04-16 19:57:33,785] INFO in event-simulator: USER4 logged in
[2022-04-16 19:57:34,786] INFO in event-simulator: USER3 is viewing page1
[2022-04-16 19:57:35,787] INFO in event-simulator: USER3 logged out
[2022-04-16 19:57:36,788] WARNING in event-simulator: USER5 Failed to Login as the account is locked due
 to MANY FAILED ATTEMPTS.
[2022-04-16 19:57:36,789] WARNING in event-simulator: USER30 Order failed as the item is OUT OF STOCK.
[2022-04-16 19:57:36,789] INFO in event-simulator: USER3 logged in
[2022-04-16 19:57:37,790] INFO in event-simulator: USER1 logged in
[2022-04-16 19:57:38,791] INFO in event-simulator: USER1 is viewing page1
[2022-04-16 19:57:39,792] INFO in event-simulator: USER4 is viewing page2
[2022-04-16 19:57:40,794] INFO in event-simulator: USER1 logged in
[2022-04-16 19:57:41,794] WARNING in event-simulator: USER5 Failed to Login as the account is locked due
 to MANY FAILED ATTEMPTS.
[2022-04-16 19:57:41,794] INFO in event-simulator: USER4 is viewing page1
[2022-04-16 19:57:42,796] INFO in event-simulator: USER1 logged out
[2022-04-16 19:57:43,797] INFO in event-simulator: USER1 is viewing page3
[2022-04-16 19:57:44,798] WARNING in event-simulator: USER30 Order failed as the item is OUT OF STOCK.
[2022-04-16 19:57:44,799] INFO in event-simulator: USER1 logged in
[2022-04-16 19:57:45,800] INFO in event-simulator: USER2 is viewing page1
[2022-04-16 19:57:46,801] WARNING in event-simulator: USER5 Failed to Login as the account is locked due
 to MANY FAILED ATTEMPTS.
[2022-04-16 19:57:46,802] INFO in event-simulator: USER4 logged out
[2022-04-16 19:57:47,803] INFO in event-simulator: USER3 logged out
[2022-04-16 19:57:48,804] INFO in event-simulator: USER1 is viewing page2

controlplane ~ ➜
```