MASTER-ACADEMY

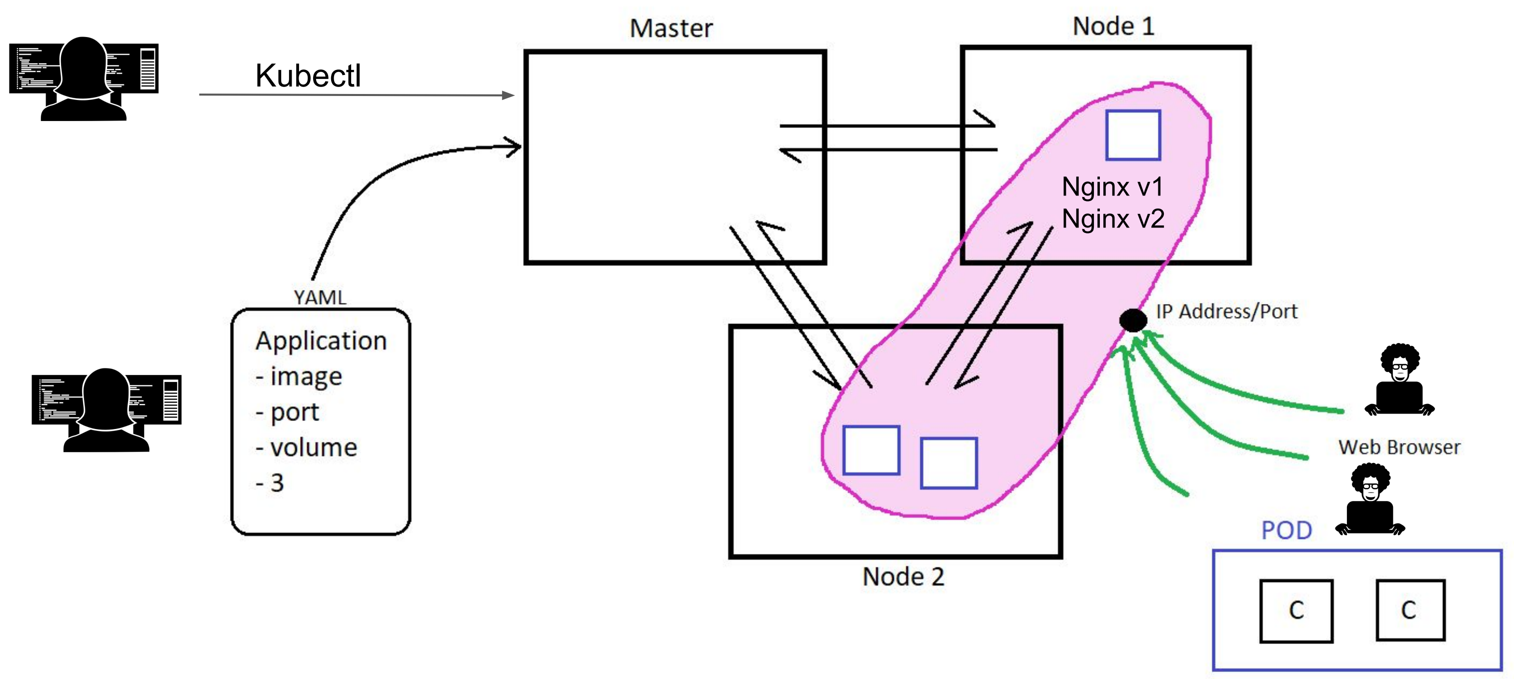
Container Orchestration using Kubernetes Part - 2

Week 4

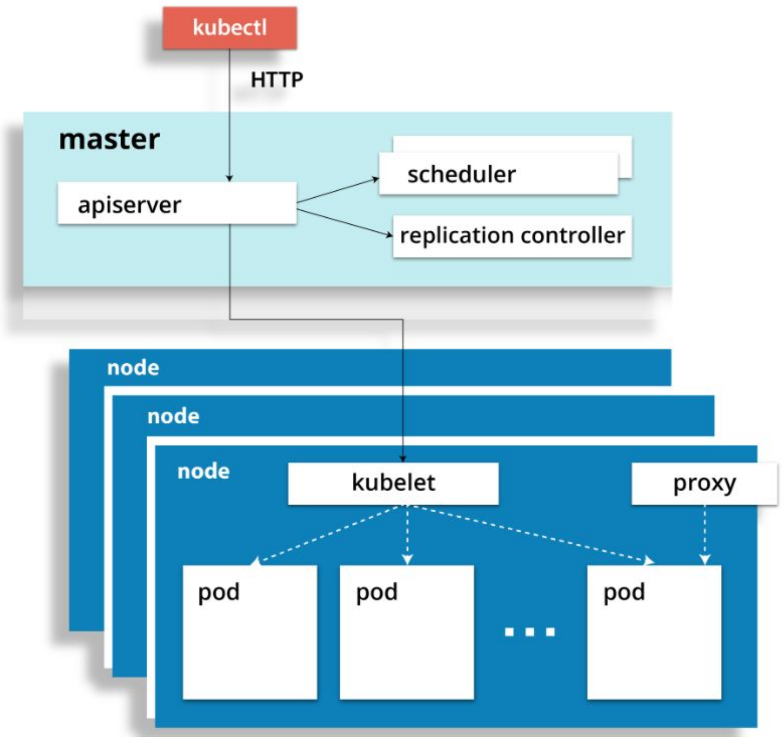
Topics

- **Part - 1**
- Container Orchestration Concept
- Introduction to Kubernetes
- Features and Advantages of Kubernetes
- Creating a Kubernetes Cluster
- **Part - 2**
- Deployment in Kubernetes
- Services in Kubernetes
- Rolling updates in Kubernetes
- Kubernetes Dashboard

Kubernetes Cluster Concept



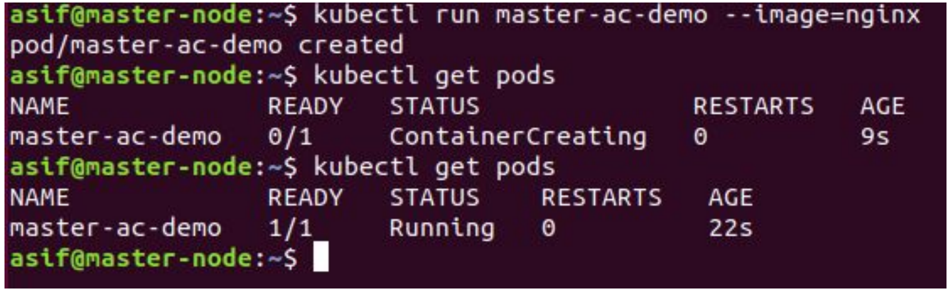
Deploying an Application



- In order to deploy an application we can use “kubectl” command
- A proxy is an agent that helps pods to communicate with each other and other components inside a cluster.
- We need to install a network for proxy to work. We can use Calico or Flannel.
- Kublet is an agent that runs on every node in a cluster
  - It is responsible for managing pods and their containers
  - Deals with pod specs YAML

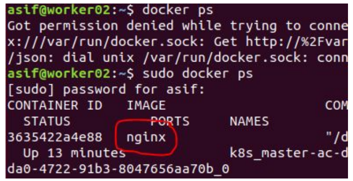
Creating a New Pod - Imperative

- Creating Pods using imperative commands:
  - kubectl run master-ac-demo --image=nginx

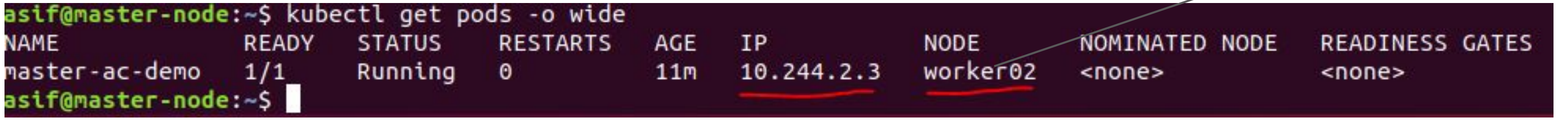


```
asif@master-node:~$ kubectl run master-ac-demo --image=nginx
pod/master-ac-demo created
asif@master-node:~$ kubectl get pods
NAME             READY   STATUS              RESTARTS   AGE
master-ac-demo   0/1     ContainerCreating   0          9s
asif@master-node:~$ kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
master-ac-demo   1/1     Running   0          22s
asif@master-node:~$
```

  - kubectl get pods -o wide



```
asif@worker02:~$ docker ps
Got permission denied while trying to conne
x:///var/run/docker.sock: Get http://%2Fvar
/json: dial unix /var/run/docker.sock: conn
asif@worker02:~$ sudo docker ps
[sudo] password for asif:
CONTAINER ID   IMAGE                      COM
  STATUS            PORTS     NAMES
3635422a4e88   nginx                      "/d
  Up 13 minutes               k8s_master-ac-d
da0-4722-91b3-8047656aa70b_0
```

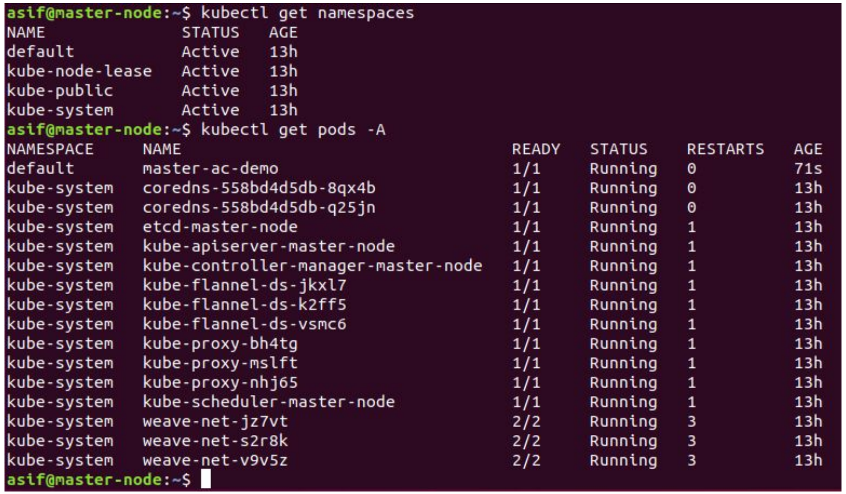


```
asif@master-node:~$ kubectl get pods -o wide
NAME             READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
master-ac-demo   1/1     Running   0          11m   10.244.2.3   worker02   <none>           <none>
asif@master-node:~$
```

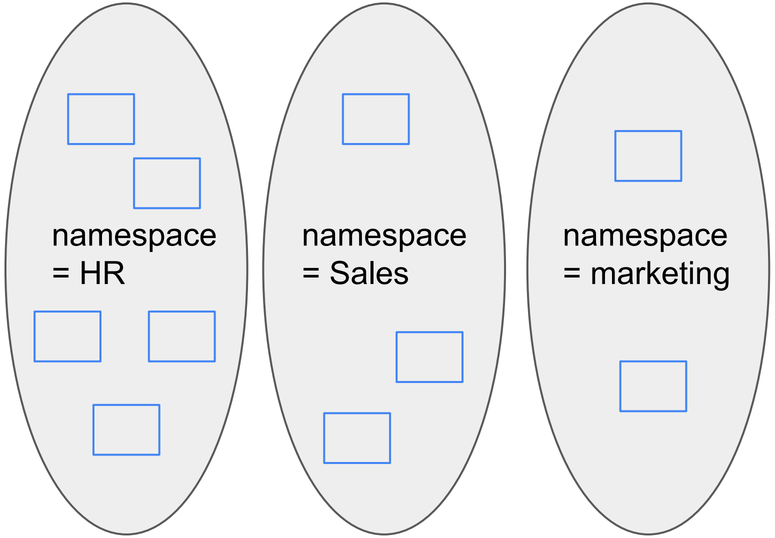
  - See what is inside a pod: kubectl describe pod master-ac-demo

What is a namespace in Kubernetes?

- A namespace is an abstracted domain where different teams can deploy their own pods
- One team cannot see other team's Pods



```
asif@master-node:~$ kubectl get namespaces
NAME              STATUS   AGE
default           Active   13h
kube-node-lease   Active   13h
kube-public       Active   13h
kube-system       Active   13h
asif@master-node:~$ kubectl get pods -A
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
default       master-ac-demo                        1/1     Running   0          71s
kube-system   coredns-558bd4d5db-8qx4b              1/1     Running   0          13h
kube-system   coredns-558bd4d5db-q25jn              1/1     Running   0          13h
kube-system   etcd-master-node                      1/1     Running   1          13h
kube-system   kube-apiserver-master-node            1/1     Running   1          13h
kube-system   kube-controller-manager-master-node   1/1     Running   1          13h
kube-system   kube-flannel-ds-jkxl7                 1/1     Running   1          13h
kube-system   kube-flannel-ds-k2ff5                 1/1     Running   1          13h
kube-system   kube-flannel-ds-vsmc6                 1/1     Running   1          13h
kube-system   kube-proxy-bh4tg                      1/1     Running   1          13h
kube-system   kube-proxy-mslft                      1/1     Running   1          13h
kube-system   kube-proxy-nhj65                      1/1     Running   1          13h
kube-system   kube-scheduler-master-node            1/1     Running   1          13h
kube-system   weave-net-jz7vt                       2/2     Running   3          13h
kube-system   weave-net-s2r8k                       2/2     Running   3          13h
kube-system   weave-net-v9v5z                       2/2     Running   3          13h
asif@master-node:~$
```

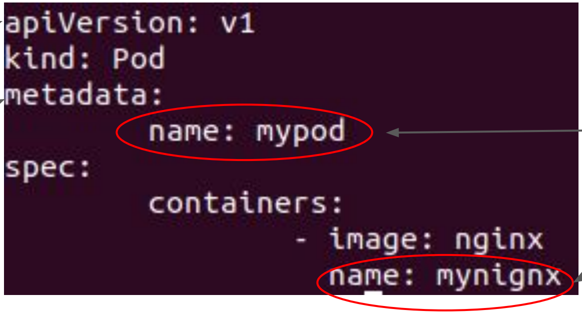


All Cluster components are inside kube-system namespace

Creating Pod from Configuration - Declarative

- Creating Pod using YAML file is called declarative way of defining resources
- In production it is NOT advised to run imperative commands to deploy a pod using the “kubectl run” command. Instead we use YAML or JSON file
- Complete Pod specifications should be written in a YAML file
  - Vim podSpec.yaml

We can define:
1. API version
2. Resource
3. Information
4. Specification



```
apiVersion: v1
kind: Pod
metadata:
        name: mypod
spec:
        containers:
                - image: nginx
                  name: mynignx
```

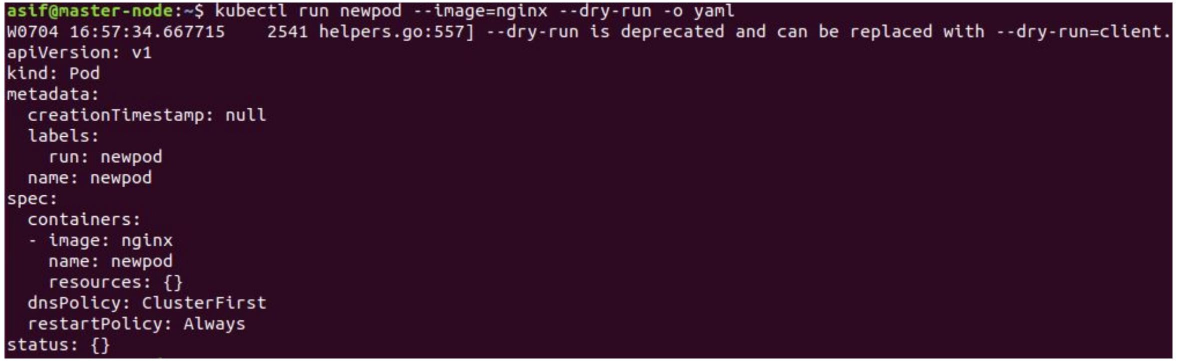
Define different names for Pod and container inside the Pod

Execute YAML file:
kubectl create -f podSpec.yaml
kubectl describe pod mypod

MASTER-ACADEMY

Creating Pod from Template

- Create a Template for Pods from imperative commands with dry run
  - kubectl run newpod --image=nginx --dry-run -o yaml



```
asif@master-node:~$ kubectl run newpod --image=nginx --dry-run -o yaml
W0704 16:57:34.667715    2541 helpers.go:557] --dry-run is deprecated and can be replaced with --dry-run=client.
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: newpod
  name: newpod
spec:
  containers:
  - image: nginx
    name: newpod
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

- Output the printout in a file
  - kubectl run newpod --image=nginx --dry-run -o yaml > newpodspec.yaml

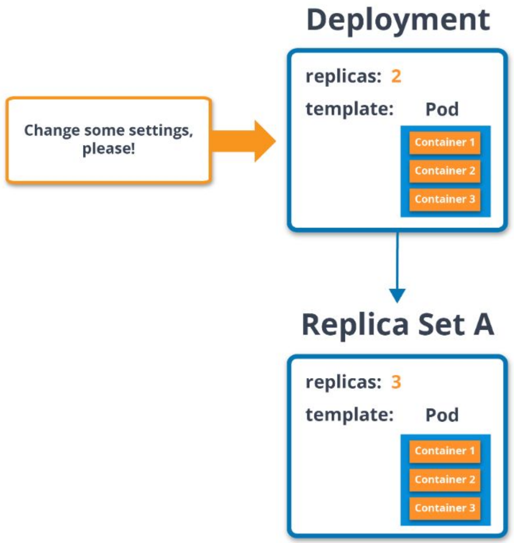
Deployment in Kubernetes

- There is a problem with just running the yaml file. There is a possibility that someone can delete that pod and it will not run again. High availability is not yet established. We use deployment to solve this problem

- Deployment can be defined to create new replica sets

- It can be defined to remove the existing deployment and use all their resources with new deployments

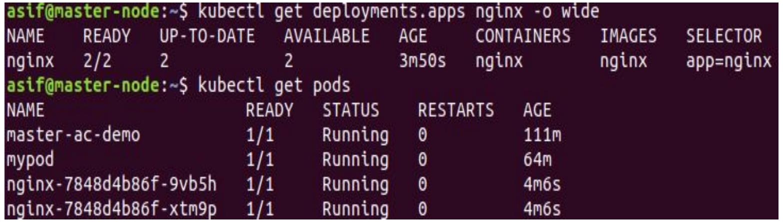
- Selector field defines how the pods management sequence is determined by deployment



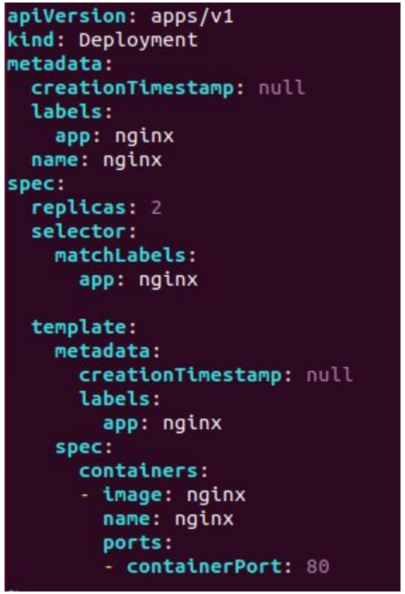
MASTER-ACADEMY

Deployment in Kubernetes (cont)

- Create a deployment using the following command
  - kubectl create deployment nginx --image=nginx --dry-run -o yaml > deployment.yaml
  - kubectl create -f deployment.yaml
  - kubectl get deployments.apps nginx -o wide
  - kubectl get pods



```
asif@master-node:~$ kubectl get deployments.apps nginx -o wide
NAME    READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS   IMAGES   SELECTOR
nginx   2/2     2            2           3m50s   nginx        nginx    app=nginx
asif@master-node:~$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
master-ac-demo           1/1     Running   0          111m
mypod                    1/1     Running   0          64m
nginx-7848d4b86f-9vb5h   1/1     Running   0          4m6s
nginx-7848d4b86f-xtm9p   1/1     Running   0          4m6s
```

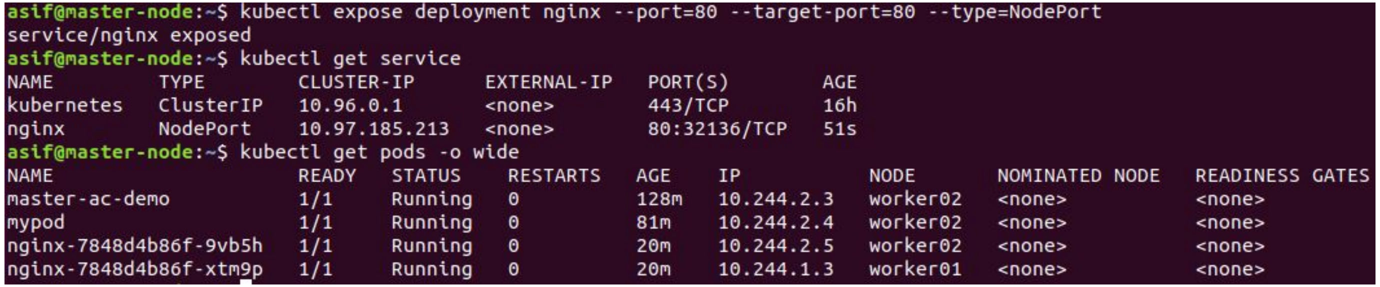


```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: nginx
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx

  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
        ports:
        - containerPort: 80
```

Service in Kubernetes

- Now that we have created a deployment or launched an application inside our cluster, how do we access the application?
  - kubectl expose deployment nginx --port=80 --target-port=80 --type=NodePort
  - OR kubectl create service NodePort --tcp=80:80
  - kubectl get service
  - kubectl get pods -o wide

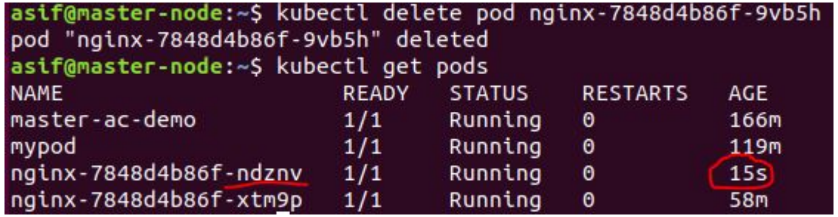


```
asif@master-node:~$ kubectl expose deployment nginx --port=80 --target-port=80 --type=NodePort
service/nginx exposed
asif@master-node:~$ kubectl get service
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        16h
nginx        NodePort    10.97.185.213   <none>        80:32136/TCP   51s
asif@master-node:~$ kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE    IP           NODE       NOMINATED NODE   READINESS GATES
master-ac-demo           1/1     Running   0          128m   10.244.2.3   worker02   <none>           <none>
mypod                    1/1     Running   0          81m    10.244.2.4   worker02   <none>           <none>
nginx-7848d4b86f-9vb5h   1/1     Running   0          20m    10.244.2.5   worker02   <none>           <none>
nginx-7848d4b86f-xtm9p   1/1     Running   0          20m    10.244.1.3   worker01   <none>           <none>
```

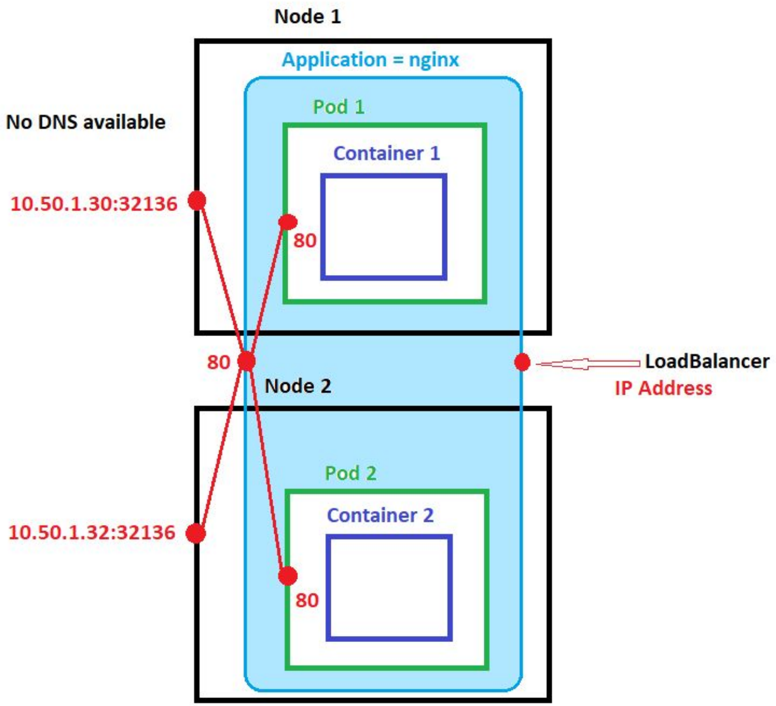
- Access the application from a browser, type: ip:port “ex: 10.50.1.30:32136”

Service in Kubernetes (cont)

- NodePort is not the ideal way for launching an application on the Kubernetes cluster.
- The user needs to know the IP address of each node inside the cluster which is not practical
- The solution is to use LoadBalancer instead of NodePort
- However Nodeport solves the problem of availability. How can we test it
  - kubectl delete pod nginx-7848d4b86f-9vb5h
  - kubectl get pods



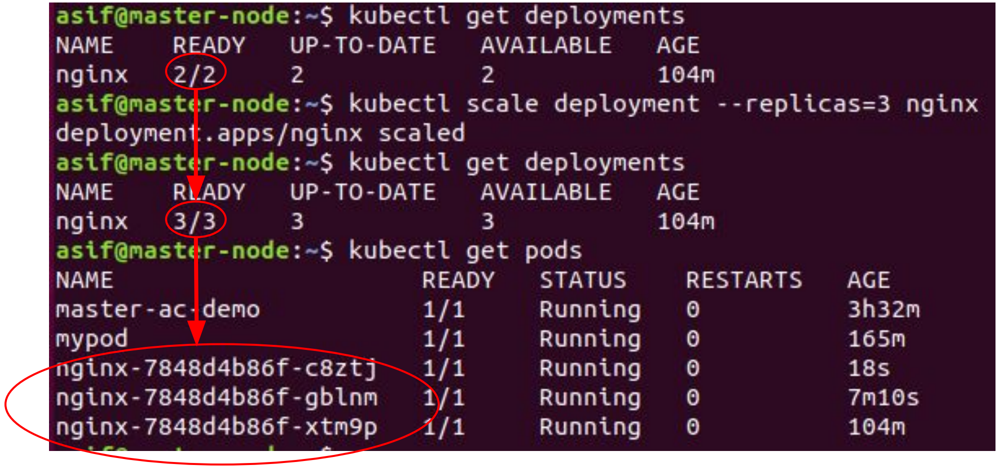
```
asif@master-node:~$ kubectl delete pod nginx-7848d4b86f-9vb5h
pod "nginx-7848d4b86f-9vb5h" deleted
asif@master-node:~$ kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
master-ac-demo            1/1     Running   0          166m
mypod                     1/1     Running   0          119m
nginx-7848d4b86f-ndznv    1/1     Running   0          15s
nginx-7848d4b86f-xtm9p    1/1     Running   0          58m
```



MASTER-ACADEMY

Scaling Using Replica Sets

- In order to scale up instances/pods we use the following command:
  - kubectl scale deployment --replicas=3 nginx
  - kubectl get deployments
  - kubectl get pods

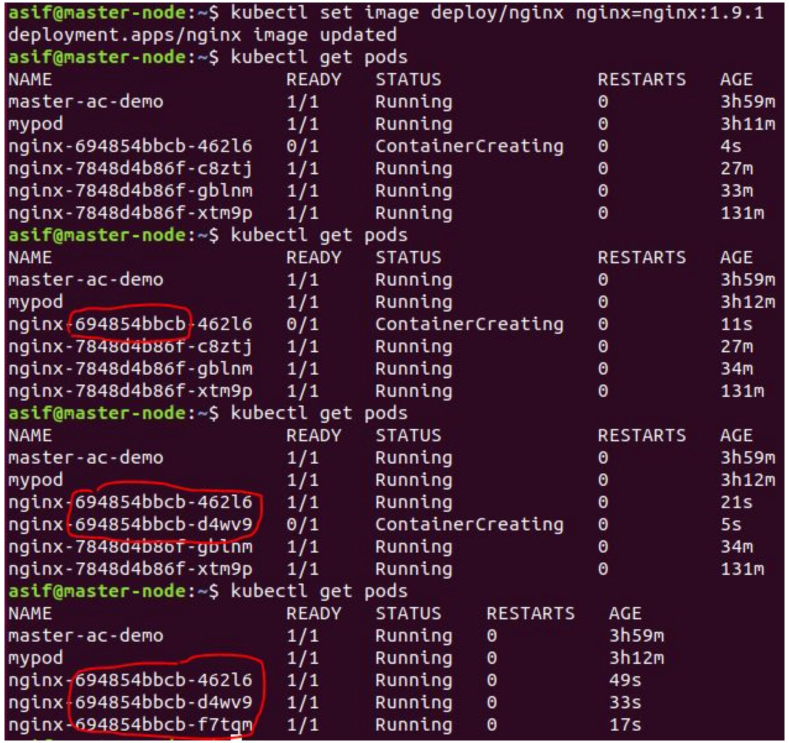


```
asif@master-node:~$ kubectl get deployments
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
nginx    2/2     2            2           104m
asif@master-node:~$ kubectl scale deployment --replicas=3 nginx
deployment.apps/nginx scaled
asif@master-node:~$ kubectl get deployments
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
nginx    3/3     3            3           104m
asif@master-node:~$ kubectl get pods
NAME                       READY   STATUS    RESTARTS   AGE
master-ac-demo             1/1     Running   0          3h32m
mypod                      1/1     Running   0          165m
nginx-7848d4b86f-c8ztj     1/1     Running   0          18s
nginx-7848d4b86f-gblnm     1/1     Running   0          7m10s
nginx-7848d4b86f-xtm9p     1/1     Running   0          104m
```

MASTER-ACADEMY

Kubernetes Rolling Update

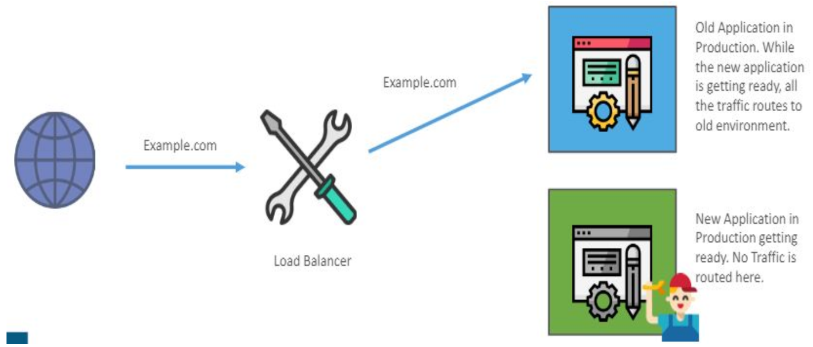
- Suppose we have a new version of our application "nginx"
- How do we update the application without any service disruption?
- Kubernetes does this automatically by using the following command:
  - kubectl set image deploy/nginx nginx=nginx:1.9.1

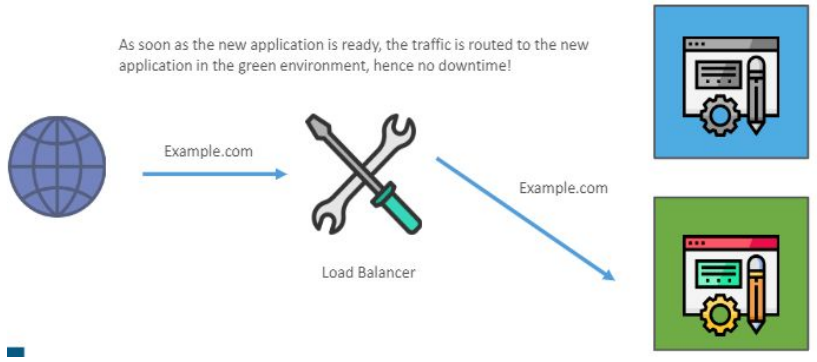


```
asif@master-node:~$ kubectl set image deploy/nginx nginx=nginx:1.9.1
deployment.apps/nginx image updated
asif@master-node:~$ kubectl get pods
NAME                     READY   STATUS              RESTARTS   AGE
master-ac-demo           1/1     Running             0          3h59m
mypod                    1/1     Running             0          3h11m
nginx-694854bbcb-462l6   0/1     ContainerCreating   0          4s
nginx-7848d4b86f-c8ztj   1/1     Running             0          27m
nginx-7848d4b86f-gblnm   1/1     Running             0          33m
nginx-7848d4b86f-xtm9p   1/1     Running             0          131m
asif@master-node:~$ kubectl get pods
NAME                     READY   STATUS              RESTARTS   AGE
master-ac-demo           1/1     Running             0          3h59m
mypod                    1/1     Running             0          3h12m
nginx-694854bbcb-462l6   0/1     ContainerCreating   0          11s
nginx-7848d4b86f-c8ztj   1/1     Running             0          27m
nginx-7848d4b86f-gblnm   1/1     Running             0          34m
nginx-7848d4b86f-xtm9p   1/1     Running             0          131m
asif@master-node:~$ kubectl get pods
NAME                     READY   STATUS              RESTARTS   AGE
master-ac-demo           1/1     Running             0          3h59m
mypod                    1/1     Running             0          3h12m
nginx-694854bbcb-462l6   1/1     Running             0          21s
nginx-694854bbcb-d4wv9   0/1     ContainerCreating   0          5s
nginx-7848d4b86f-gblnm   1/1     Running             0          34m
nginx-7848d4b86f-xtm9p   1/1     Running             0          131m
asif@master-node:~$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
master-ac-demo           1/1     Running   0          3h59m
mypod                    1/1     Running   0          3h12m
nginx-694854bbcb-462l6   1/1     Running   0          49s
nginx-694854bbcb-d4wv9   1/1     Running   0          33s
nginx-694854bbcb-f7tqm   1/1     Running   0          17s
```

Blue Green Deployment Model

- Blue-green deployment is a technique that reduces downtime and risk by running two identical production environments called Blue and Green
- The updated application gets setup in the new environment (Green), while old application remains in its own environment (Blue) untouched
- Traffic stays with the blue environment until the green environment is ready
- As soon as the new application is ready, the traffic is routed to the new application in the green environment, therefore there is no downtime


Example.com
Load Balancer
Example.com
Old Application in Production. While the new application is getting ready, all the traffic routes to old environment.
New Application in Production getting ready. No Traffic is routed here.


As soon as the new application is ready, the traffic is routed to the new application in the green environment, hence no downtime!
Example.com
Load Balancer
Example.com

End of Part 2