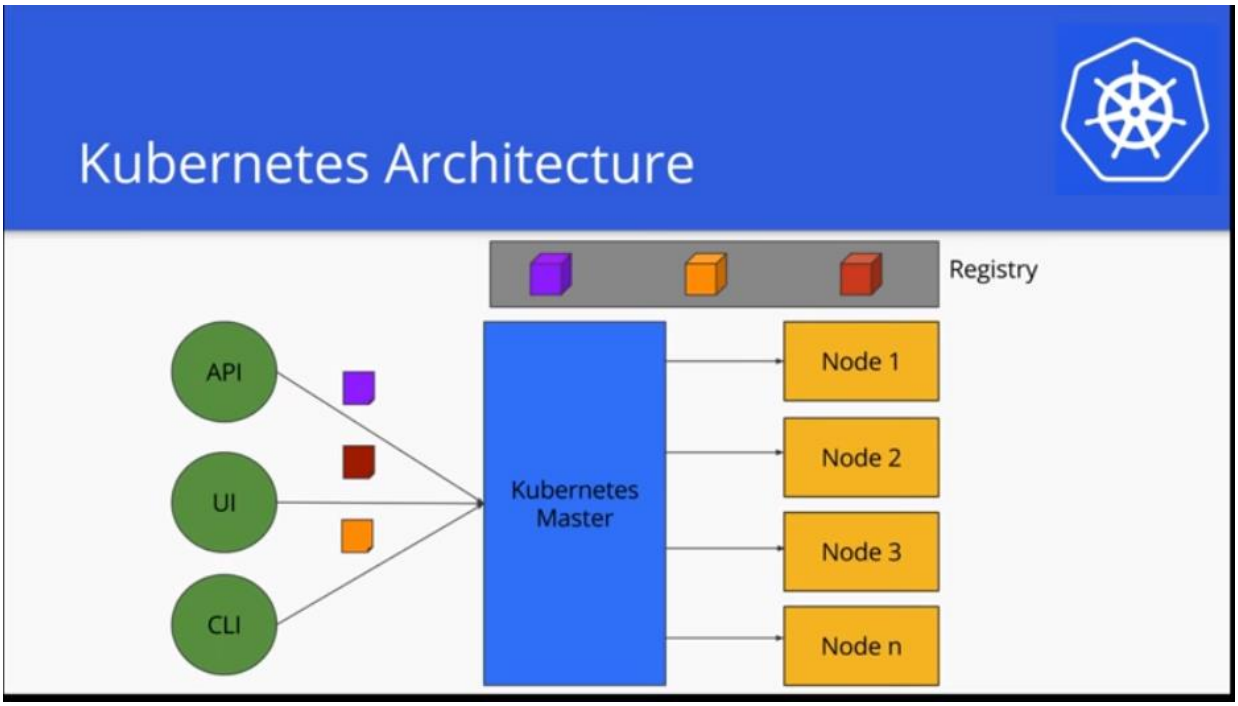
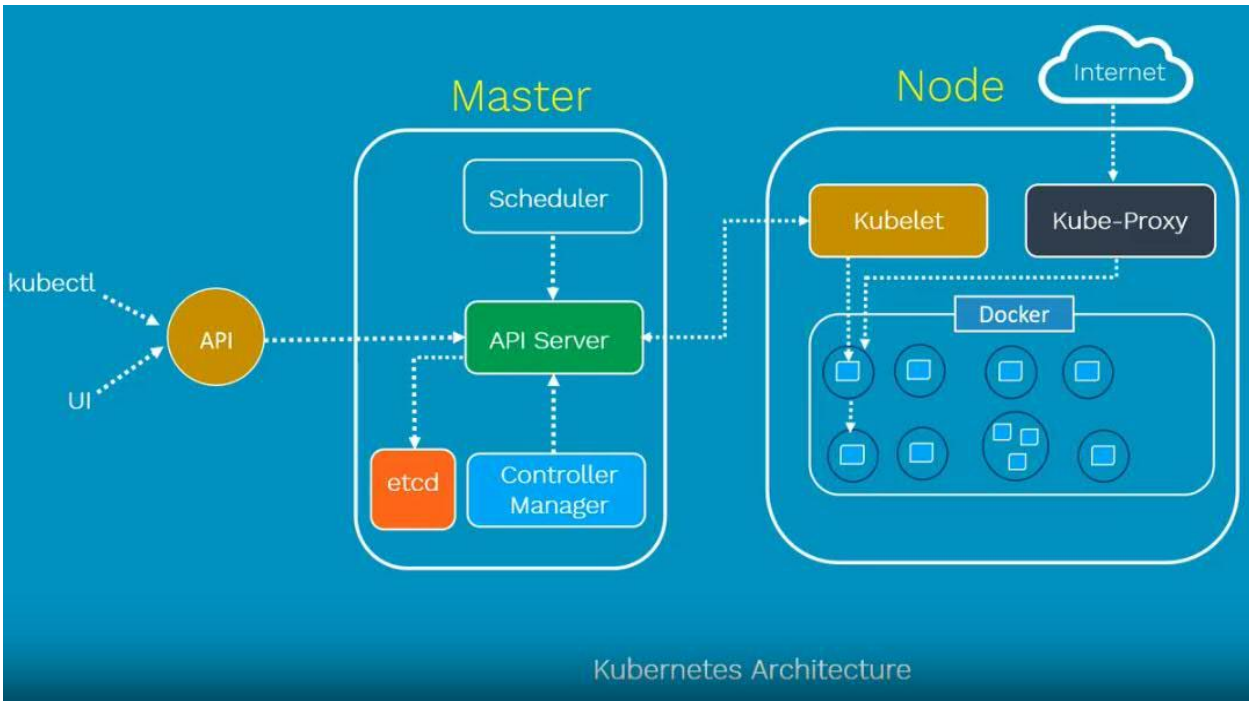
Kubernetes

Features of Kubernetes

Following are some of the important features of Kubernetes.

- Continues development, integration and deployment
- Containerized infrastructure
- Application-centric management
- Auto-scalable infrastructure
- Environment consistency across development testing and production
- Loosely coupled infrastructure, where each component can act as a separate unit
- Higher density of resource utilization
- Predictable infrastructure which is going to be created.





Pod scaling:

Pods Tight coupling and Loose coupling:

Pods life cycle: Pending, Running, Succeeded, Failed:

Pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-pod →name of the pod
spec:
  containers:
    - name: hello-ctrl – name of the container
      image: Jenkins → name of the images
      ports: - container port number
        - containerPort: 8080
```

1. NodePort
2. ClusterIP
3. LoadBalancer

kubectl get nodes

kubectl get create -f pod.yml

kubectl get pods

kubectl describe pods

kubectl get pods -o wide

kubectl het pods/hello-pod

kubectl get pods –all-namespaces

kubectl delete pods/hello-pod

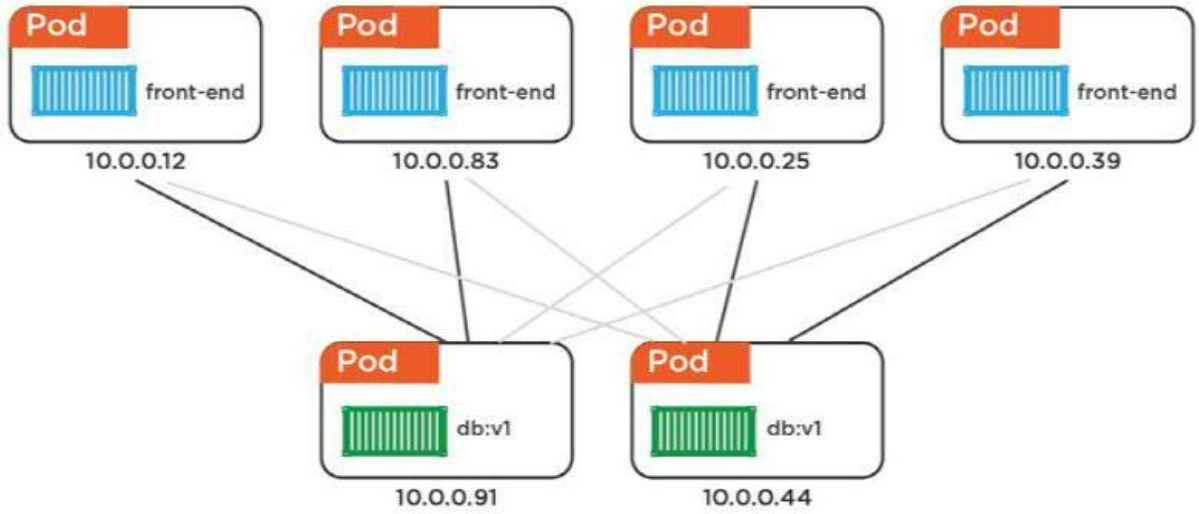
Replication Controller:

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-rc
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

 # kubectl scale rc nginx-rc --replicas=5

 # kubectl get po -l app=nginx-app

Service:

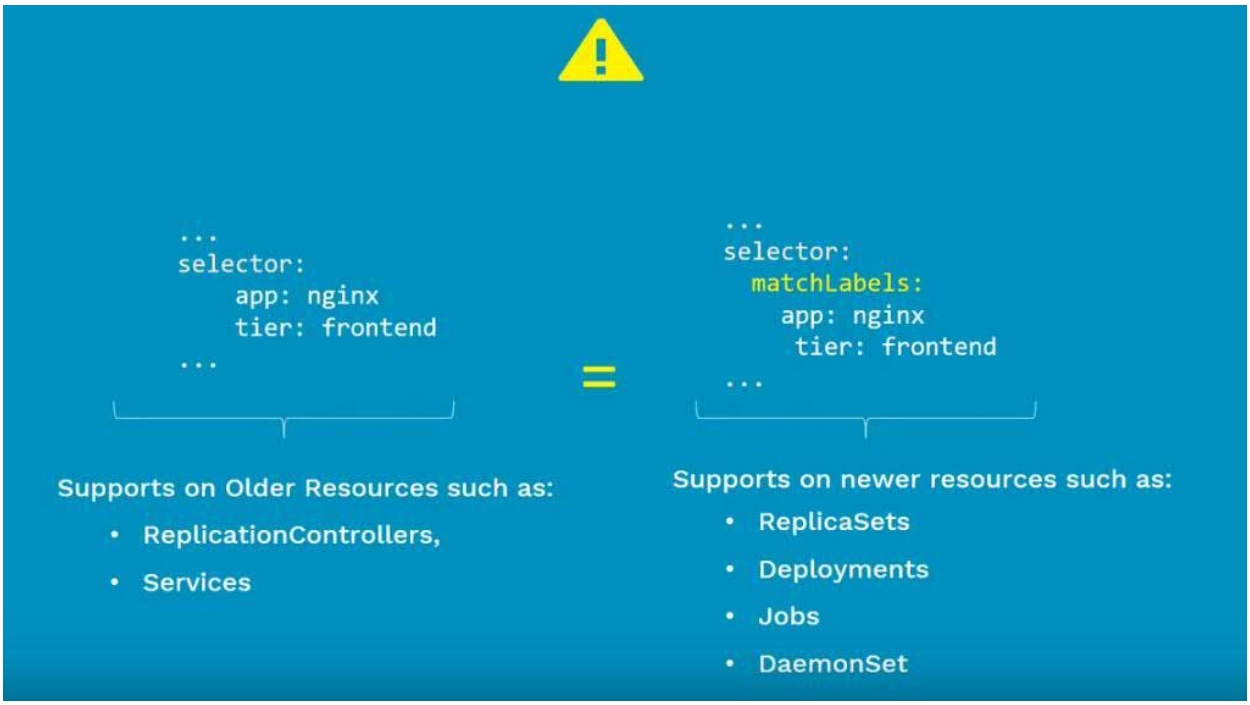



```
apiVersion: v1
kind: Service
metadata:
  name: nodeport-svc
  labels:
    app: nginx-app
spec:
  selector:
    app: nginx-app
  type: NodePort
  ports:
  - nodePort: 31000
    port: 80
    targetPort: 80
```

```
apiVersion: v1
kind: Service
metadata:
  name: nodeport-svc
  labels:
    app: nginx-app
spec:
  selector:
    app: nginx-app
  type: LoadBalancer
  ports:
  - nodePort: 31000
    port: 80
    targetPort: 80
```

ReplicaSet:

Equality Based and **Set-Based**



Deployment:

1. Recreate
2. RollingUpdate
3. Canary
4. Blue/Green

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deploy
  labels:
    app: nginx-app
spec:
  replicas: 3
  selector:
```

```
  matchLabels:
    app: nginx-app
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
      - name: nginx-pod
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

We can update using in 2 ways. Using **kubectl set** and **kubectl edit** commands.

1st option: using kubectl set command.

 # kubectl set image deploy nginx-deploy nginx-pod =nginx:1.9.1

Or

 # kubectl set image deploy nginx-deploy nginx- pod =nginx:1.9.1 --record

2nd option: using kubectl edit command.

 # kubectl edit deploy nginx-deployment

#kubectl rollout history deployment/nginx-deployment

Scale Up and down:

 # kubectl scale deployment deploy nginx-deployment --replicas=5

Kubernetes Cluster creation:

Step1: Create required no.of instances. Minimum 2 should require (1 master and 1 slave).

Step2:

Let's disable SELinux and SWAP. To disable those, our instances needs few port numbers to be opened on all the instances. (6443, 10250).

Execute below on all nodes (master and all worker).

```
# systemctl stop firewalld

# systemctl disable firewalld

# swapoff -a

# sed -i.bak -r 's/(.+ swap .+)/#\1/' /etc/fstab

# setenforce 0

#  sed -i 's/enforcing/disabled/g' /etc/selinux/conf
```

Step3: Now it's time to install required packages for Kubernetes:

```
# cat <<EOF > /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64

enabled=1

gpgcheck=1

repo_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

exclude=kube*

EOF

# yum update -y
```

If the above command doesn't work, then open the file by using below command and paste the content and save, else ignore. Then try the above command. So it will work.

Opening file: vi /etc/yum.repos.d/kubernetes.repo

Content to save:

```
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64/
enabled=1
```

```
gpgcheck=1
repo_gpgcheck=0
gpgkey=https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
```

 # yum install -y docker kubeadm kubelet kubectl --disableexcludes=Kubernetes

If Docker is not installed using above command, then install it manually using below commands:

sudo yum install yum-utils

sudo yum-config-manager --enable rhui-REGION-rhel-server-extras

sudo yum install docker -y

 # systemctl enable docker && systemctl start docker

 # systemctl enable kubelet && systemctl start kubelet

 # cat <<EOF > /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

EOF

 # sysctl net.bridge.bridge-nf-call-iptables=1

If above command doesn't work, then first run below command and then try:

 # modprobe br_netfilter sudo sysctl -p

 # sysctl net.ipv4.ip_forward=1

 # sysctl --system

 # systemctl daemon-reload

 # systemctl restart kubelet

We can copy and run all the commands at once and run on master node. If it works fine, then run the same commands on all worker nodes.

Step4: Lets configure Kubernetes master node (run below command only on master node):

 # kubeadm init --pod-network-cidr=10.240.0.0/16

Then copy the generated commands to a note pad:

Now run below commands on master to make kubectl to work.

Run few(below) commands on master node:

 # mkdir -p $HOME/.kube

 # sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

 # sudo chown $(id -u):$(id -g) $HOME/.kube/config

To make the network work between the nodes on Kubernetes configuration, we can use different plugins like weave, flannel and calico.

Lets try to add flannel plugin:

 # kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/v0.9.1/Documentation/kube-flannel.yml

If the above command doesn't work, then try below one.

 # kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-manifests/kube-flannel-rbac.yml

If you want to use weave plugin instead of flannel, then use below command.

kubectl apply --filename https://git.io/weave-kube-1.6

These network plugin commands make's DNS up.

Make sure the DNS is up or not by running below command.

To confirm pod network is successfully installed or not, run below command.

 # kubectl get pods –all-namespaces

Step5: Lets join all worker nodes to cluster by running the join command on all worker nodes.

Repeat the same step on all worker nodes.

Step6: Testing.

To make sure on which worker node our pod is running. User below command.

 # kubectl get nodes

Ingress:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
        - path: /jenkins
          backend:
            serviceName: jenkins-svc
            servicePort: 8080
        - path: /nginx
          backend:
            serviceName: nginx-svc
            servicePort: 80
```