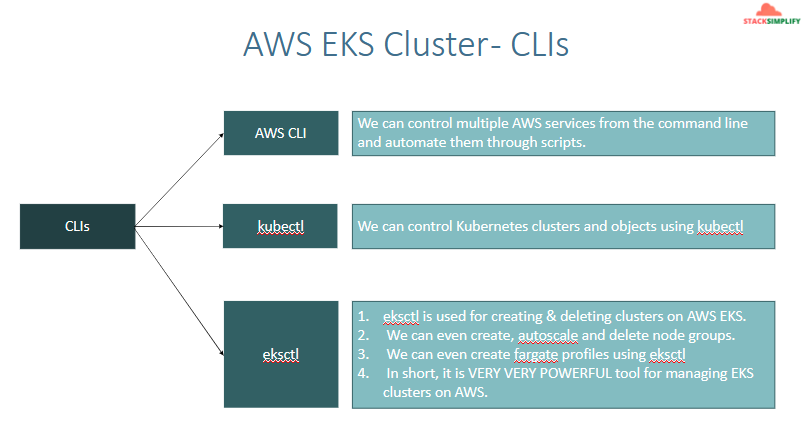
**EKS**

**INSTALL AWS, KUBECTL & EKSCTL CLI's INSTALL AWS CLI -**



**INSTALL AWS**

Reference-1: <https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html>

Reference-2: <https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html>

Reference: <https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-mac.html>

**MAC**

# Download Binary

curl "<https://awscli.amazonaws.com/AWSCLIV2.pkg>" -o "AWSCLIV2.pkg"

# Install the binary

sudo installer -pkg ./AWSCLIV2.pkg -target /

aws --version

**Windows 10**

- The AWS CLI version 2 is supported on Windows XP or later.

- The AWS CLI version 2 supports only 64-bit versions of Windows.

# Download Binary:

<https://awscli.amazonaws.com/AWSCLIV2.msi>

#Install the downloaded binary (standard windows install)

aws –version

OR

# Alternatively, you can run the msiexec command to run the MSI installer

C:\> msiexec.exe /i <https://awscli.amazonaws.com/AWSCLIV2.msi>

C:\> msiexec.exe /i <https://awscli.amazonaws.com/AWSCLIV2.msi> /qn

C:\> aws --version

**Configure AWS Command Line using Security Credentials**

- Go to AWS Management Console --> Services --> IAM

- Select the IAM User: kalyan - Click on \*\*Security credentials\*\* tab

- Click on \*\*Create access key\*\*

- Copy Access ID and Secret access key

- Go to command line and provide the required details ```

aws configure

AWS Access Key ID [None]: ABCDEFGHIAZBERTUCNGG (Replace your creds when prompted)

AWS Secret Access Key [None]: uMe7fumK1IdDB094q2sGFhM5Bqt3HQRw3IHZzBDTm (Replace your creds when prompted)

Default region name [None]: us-east-1

Default output format [None]: json ```

- Test if AWS CLI is working after configuring the above ```

aws ec2 describe-vpcs

```

**INSTALL KUBECTL CLI**

To get the exact Kubectl client version based on our EKS Cluster version

- Reference: <https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html>

**MAC - Install and configure kubectl**

# Download the Package

mkdir kubectlbinary

cd kubectlbinary

curl -o kubectl <https://amazon-eks.s3.us-west-2.amazonaws.com/1.16.8/2020-04-16/bin/darwin/amd64/kubectl>

# Provide execute permissions

chmod +x ./kubectl

# Set the Path by copying to user Home Directory

mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin echo 'export PATH=$PATH:$HOME/bin' >> ~/.bash\_profile

# Verify the kubectl version

kubectl version --short --client

**Windows 10 - Install and configure kubectl**

- Install kubectl on Windows 10

```

mkdir kubectlbinary

cd kubectlbinary

curl -o kubectl.exe <https://amazon-eks.s3.us-west-2.amazonaws.com/1.16.8/2020-04-16/bin/windows/amd64/kubectl.exe>

```

- Update the system \*\*Path\*\* environment variable

C:\Users\KALYAN\Documents\kubectlbinary

- Verify the kubectl client version

kubectl version --short --client

kubectl version –client

**INSTALL EKSCTL CLI**

- Reference: https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/

**Mac**

# Install Homebrew on MacOs

/bin/bash -c "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install.sh>)"

# Install the Weaveworks Homebrew tap.

brew tap weaveworks/tap

# Install the Weaveworks Homebrew tap.

brew install weaveworks/tap/eksctl

# Verify eksctl version

eksctl version

```

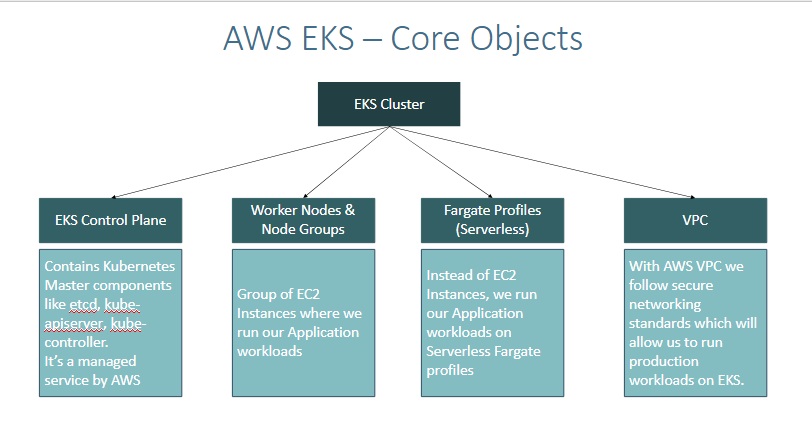
**windows or linux**

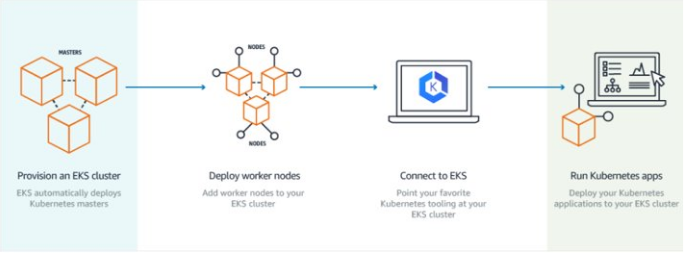
- For windows and linux OS, you can refer below documentation link.

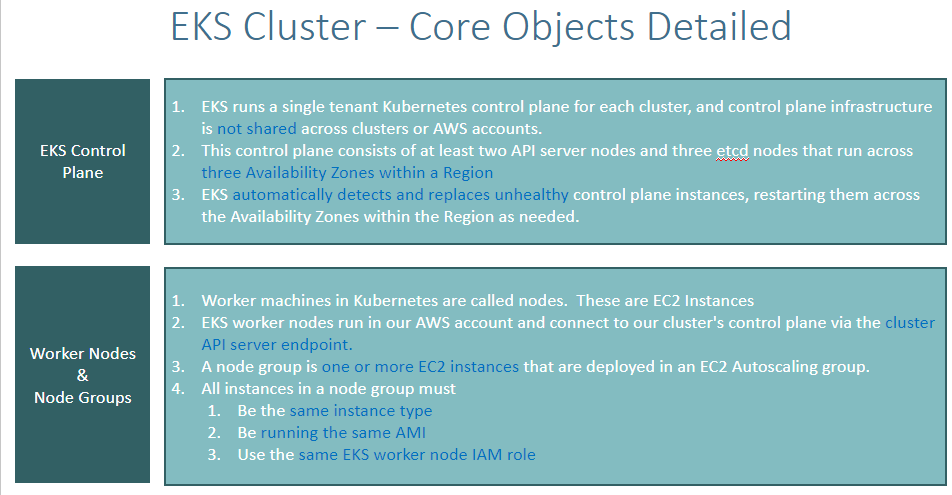
<https://github.com/eksctl-io/eksctl/blob/main/README.md#installation>

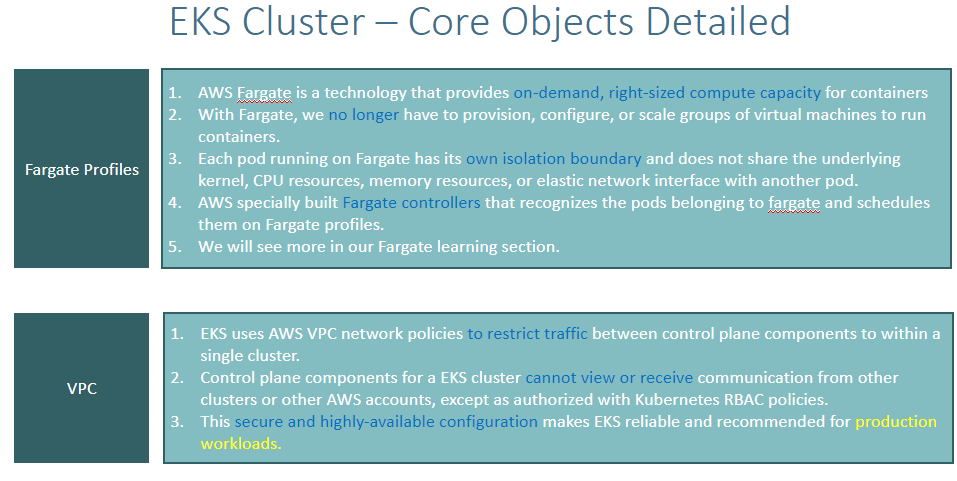
- \*\*Reference:\*\*

<https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html#installing-eksctl> - <https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html>









## **CREATING EKS CLUSTER USING EKSCTL**

**Create Cluster**  
eksctl create cluster --name=eksdemo1 \  
 --region=us-east-1 \  
 --zones=us-east-1a,us-east-1b \  
 --without-nodegroup   
  
# Get List of clusters  
eksctl get cluster

## **Create & Associate IAM OIDC Provider for our EKS Cluster**

* To enable and use AWS IAM roles for Kubernetes service accounts on our EKS cluster, we must create & associate OIDC identity provider.
* To do so using eksctl we can use the below command.
* Use latest eksctl version (as on today the latest version is 0.21.0)  
    
  # Replace with region & cluster name  
  eksctl utils associate-iam-oidc-provider \  
   --region us-east-1 \  
   --cluster eksdemo1 \  
   --approve

## **Create EC2 Keypair**

* Create a new EC2 Keypair with name as kube-demo
* This keypair we will use it when creating the EKS NodeGroup.
* This will help us to login to the EKS Worker Nodes using Terminal.

## **Create Node Group with additional Add-Ons in Public Subnets**

* These add-ons will create the respective IAM policies for us automatically within our Node Group role.

# Create Public Node Group   
eksctl create nodegroup --cluster=eksdemo1 \  
 --region=us-east-1 \  
 --name=eksdemo1-ng-public1 \  
 --node-type=t3.medium \  
 --nodes=2 \  
 --nodes-min=2 \  
 --nodes-max=4 \  
 --node-volume-size=20 \  
 --ssh-access \  
 --ssh-public-key=kube-demo \  
 --managed \  
 --asg-access \  
 --external-dns-access \  
 --full-ecr-access \  
 --appmesh-access \  
 --alb-ingress-access   
**Note**: eksctl create cluster --help  
 eksctl create nodegroup –help  
 eksctl create --help

## **Verify Cluster & Nodes**

### Verify NodeGroup subnets to confirm EC2 Instances are in Public Subnet

* Verify the node group subnet to ensure it created in public subnets
* Go to Services -> EKS -> eksdemo -> In details select Node group -eksdemo1-ng1-public
* Click on Associated subnet in **Details** tab
* Click on **Route Table** Tab.
* We should see that internet route via Internet Gateway (0.0.0.0/0 -> igw-xxxxxxxx)

### Verify Cluster, NodeGroup in EKS Management Console

* Go to Services -> Elastic Kubernetes Service -> eksdemo1

### List Worker Nodes

# List EKS clusters  
eksctl get cluster  
# List NodeGroups in a cluster  
eksctl get nodegroup --cluster=<clusterName>  
# List Nodes in current kubernetes cluster  
kubectl get nodes -o wide  
# Our kubectl context should be automatically changed to new cluster  
kubectl config view --minify

### Verify Worker Node IAM Role and list of Policies

* Go to Services -> EC2 -> Worker Nodes
* Click on **IAM Role associated to EC2 Worker Nodes**

### Verify Security Group Associated to Worker Nodes

* Go to Services -> EC2 -> Worker Nodes
* Click on **Security Group** associated to EC2 Instance which contains remote in the name.

### Verify CloudFormation Stacks

* Verify Control Plane Stack & Events
* Verify NodeGroup Stack & Events

### Login to Worker Node using Keypai kube-demo

* Login to worker node

# For MAC or Linux or Windows10  
ssh -i kube-demo.pem ec2-user@<Public-IP-of-Worker-Node>  
# For Windows 7  
Use putty  
**Note:** To view disk utilisation df-h

## **Update Worker Nodes Security Group to allow all traffic**

* We need to allow All Traffic on worker node security group

## References:

* <https://docs.aws.amazon.com/eks/latest/userguide/enable-iam-roles-for-service-accounts.html>
* <https://docs.aws.amazon.com/eks/latest/userguide/create-service-account-iam-policy-and-role.html>

# **DELETE EKS CLUSTER AND NODE GROUP**

## **Delete Node Group**

* We can delete a nodegroup separately using below eksctl delete nodegroup  
  # Delete Node Group  
  eksctl delete nodegroup --cluster=eksdemo1 --name=eksdemo1-ng-public1

## **Delete Cluster**

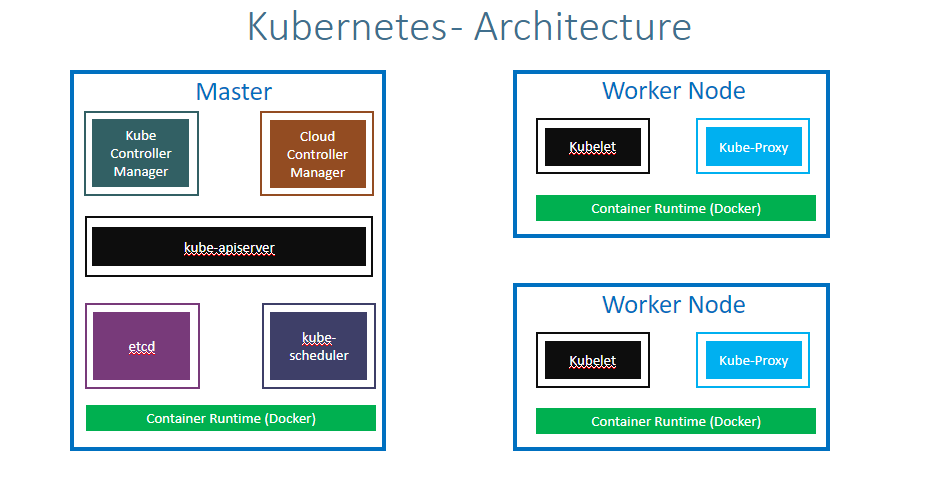
* We can delete cluster using eksctl delete cluster

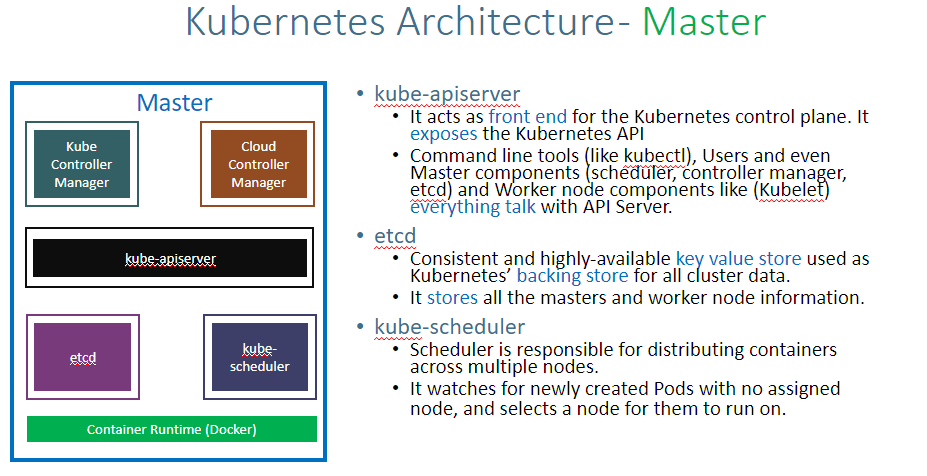
# Delete Cluster  
eksctl delete cluster eksdemo1

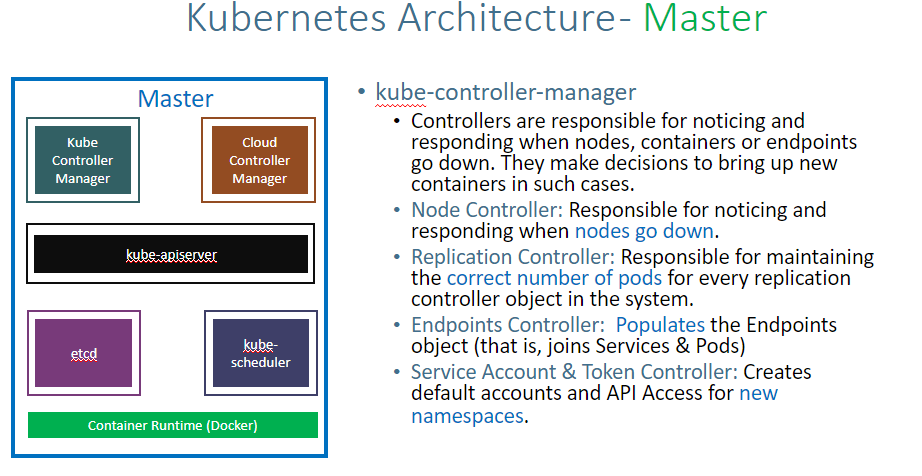
### Note-1: Rollback any Security Group Changes

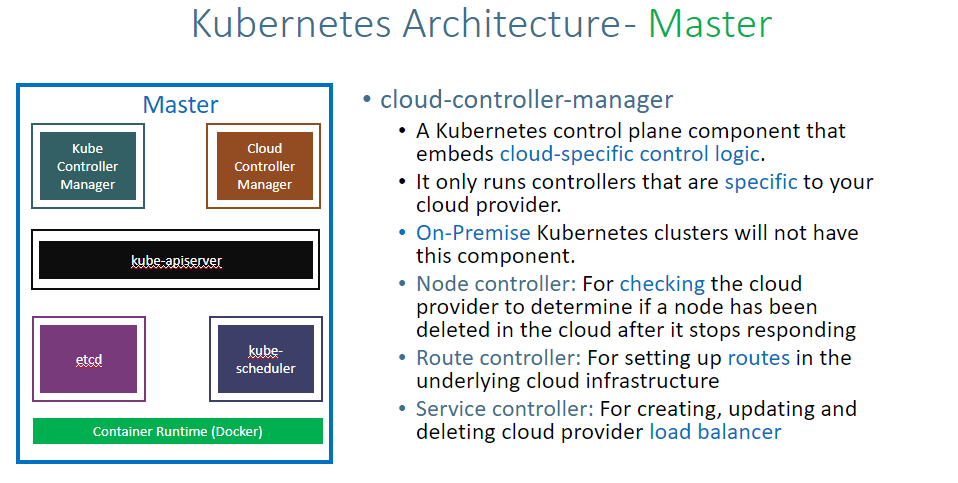
### Note-2: Rollback any EC2 Worker Node Instance Role - Policy changes

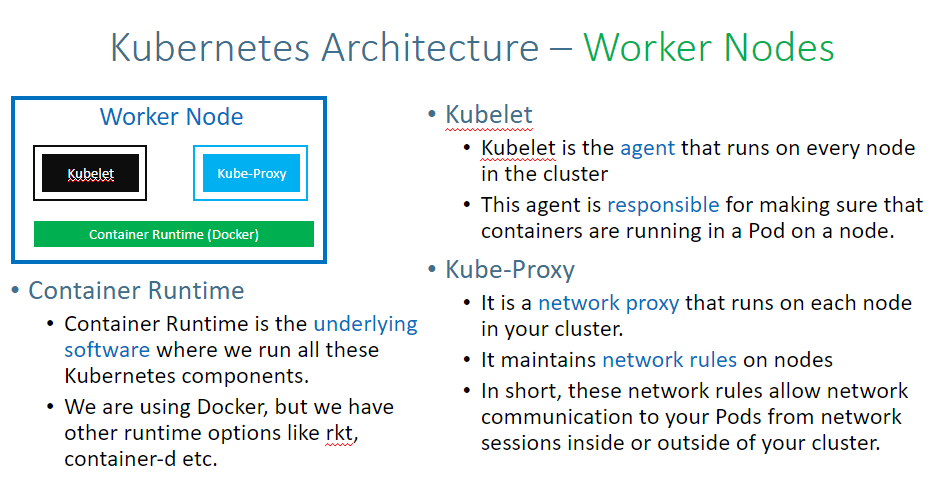
EBS CSI Driver

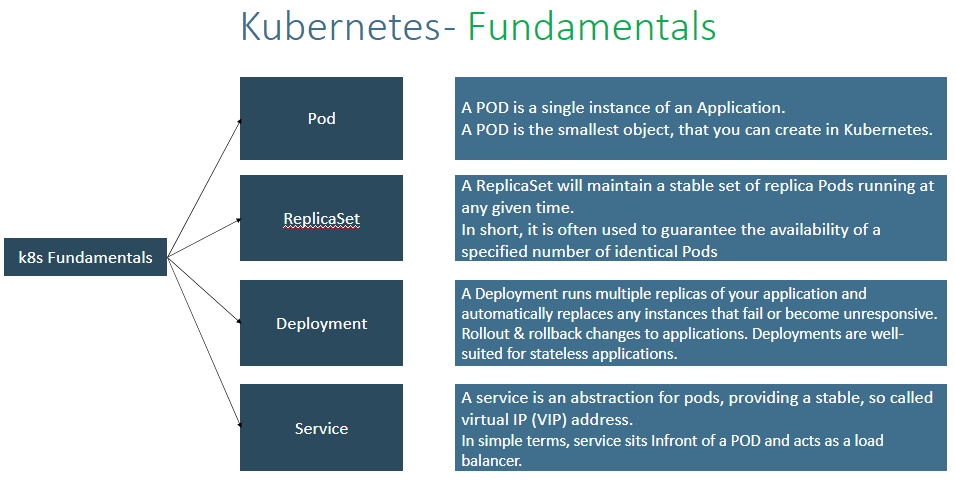


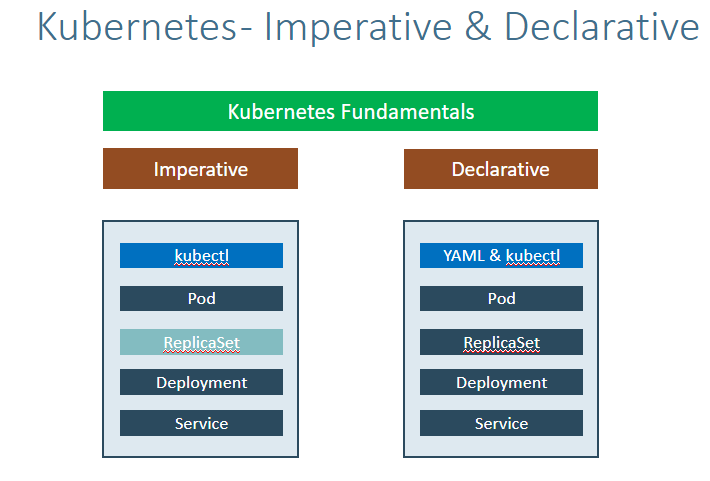


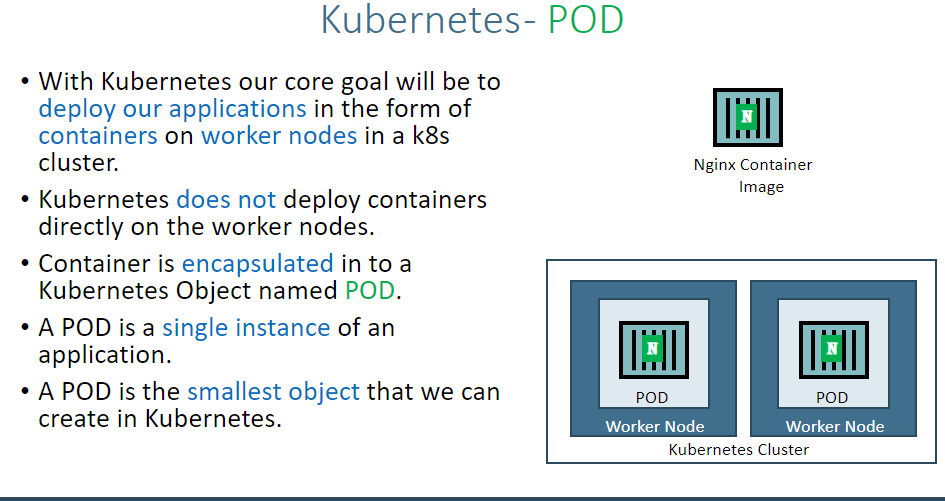


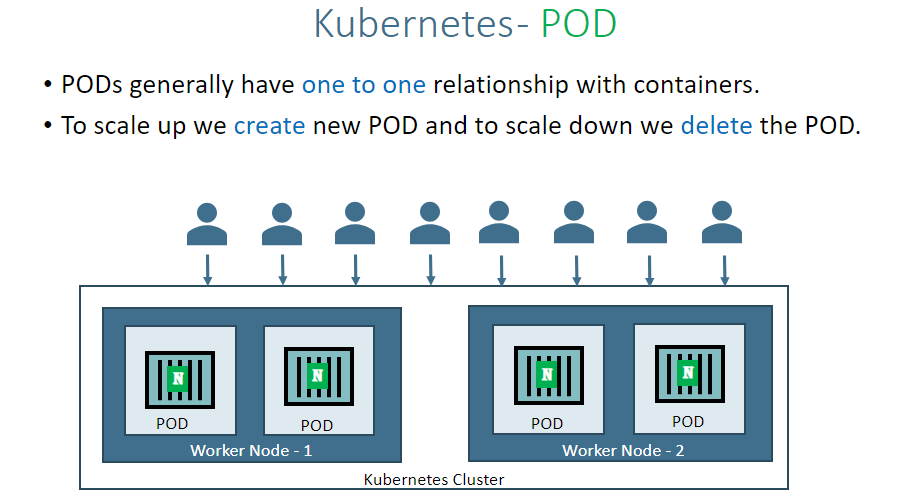


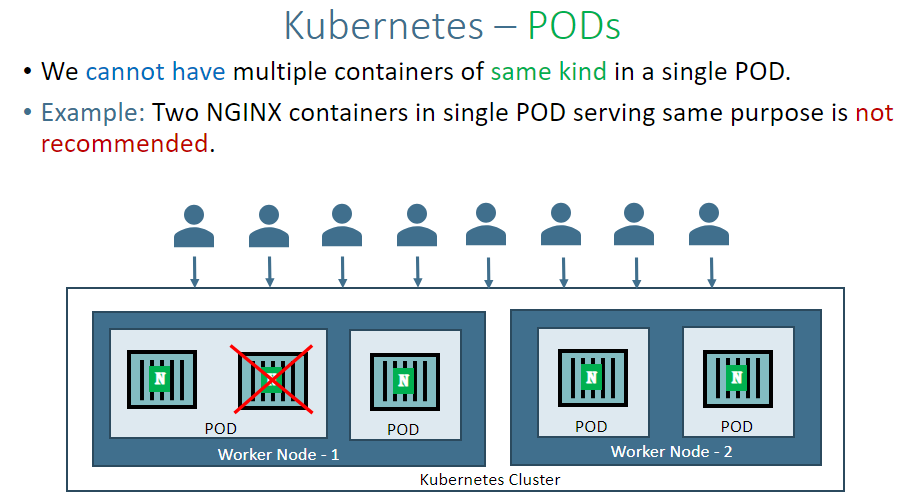


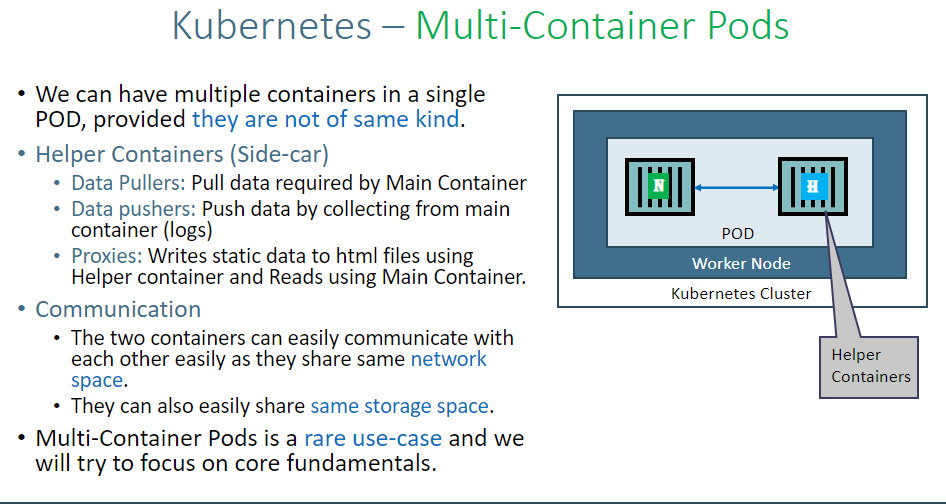












**IMPERATIVE METHOD**

**Create a Pod**

kubectl run <desired-pod-name> --image <Container-Image> --generator=run-pod/v1

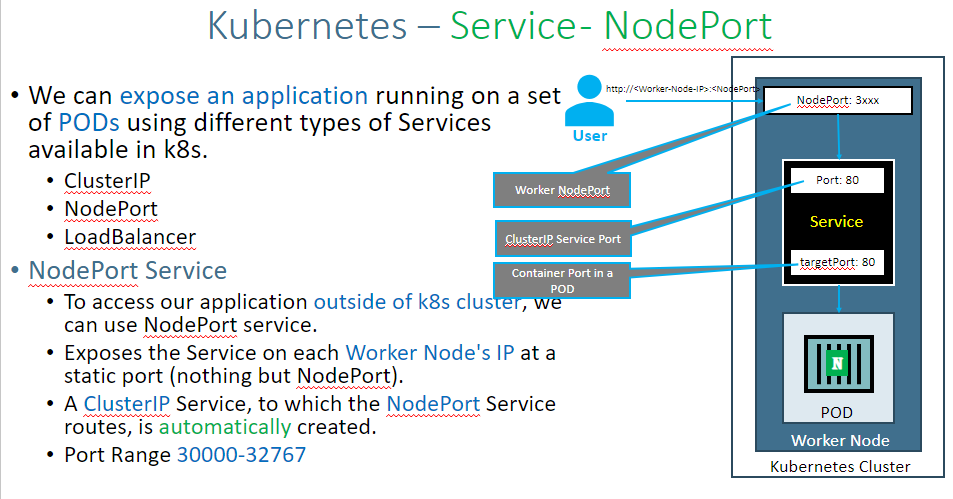
kubectl run my-first-pod --image stacksimplify/kubenginx:1.0.0 --generator=run-pod/v1

* With Kubernetes 1.18 version, there is lot clean-up to kubectl run command.
* The below will suffice to create a Pod as a pod without creating deployment. We dont need to add --generator=run-pod/v1

**List Pods**  
kubectl get pods

**Describe the Pod**  
kubectl describe pod my-first-pod

**Delete Pod**  
kubectl delete pod my-first-pod



**Ports**

* **Port:** Port on which node port service listens in Kubernetes cluster internally
* **TargetPort:** We define container port here on which our application is running.
* **NodePort:** Worker Node port on which we can access our application.

**Expose Pod as a Service**  
kubectl expose pod <Pod-Name> --type=NodePort --port=80 --name=<Service-Name>  
kubectl expose pod my-first-pod --type=NodePort --port=80 --name=my-first-service

**Get Service Info**  
kubectl get service

Access the Application using Public IP

http://<node1-public-ip>:<Node-Port>

If target-port is not defined, by default and for convenience, the **targetPort** is set to the same value as the **port** field.

Expose Pod as a Service with Container Port (--taret-port)  
kubectl expose pod my-first-pod --type=NodePort --port=81 --target-port=80 --name=my-first-service3

### **Verify Pod Logs** kubectl logs my-first-pod

### **Connect to Container in a POD** kubectl exec -it <pod-name> -- /bin/bash kubectl exec -it my-first-pod -- /bin/bash

**Running individual commands in a Container**

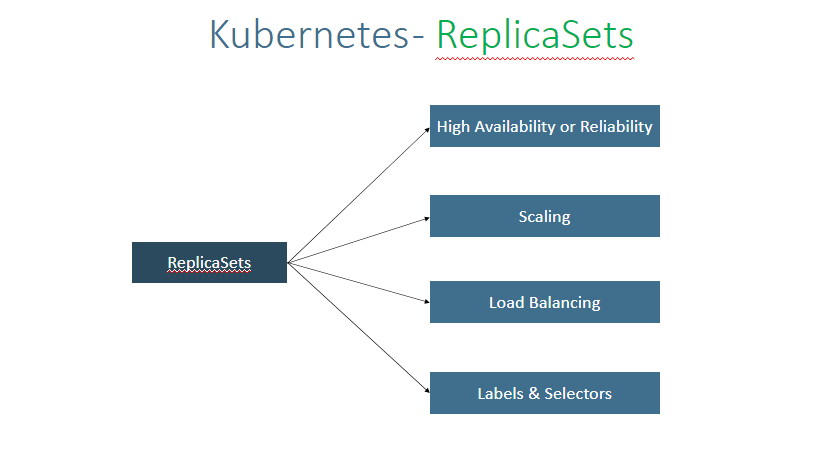
kubectl exec -it <pod-name> env  
  
kubectl exec -it my-first-pod env  
kubectl exec -it my-first-pod ls  
kubectl exec -it my-first-pod cat /usr/share/nginx/html/index.html

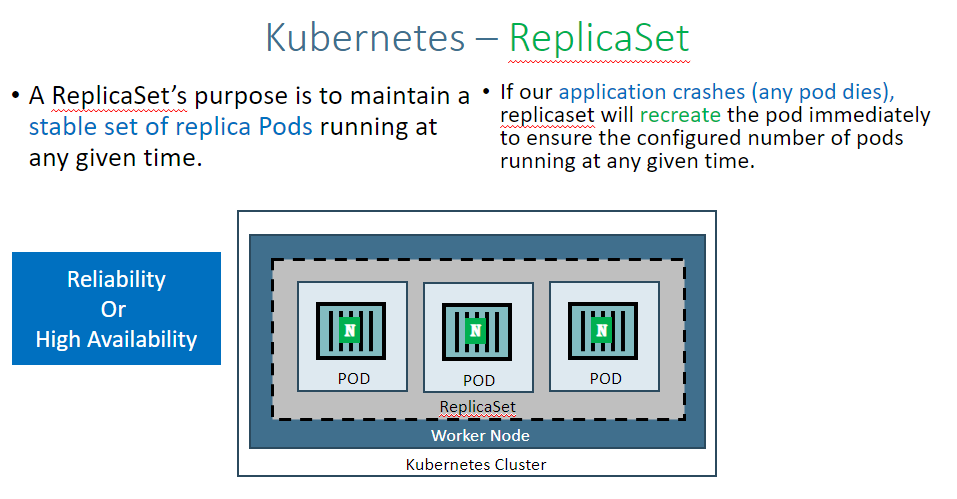
## **Get YAML Output of Pod & Service**

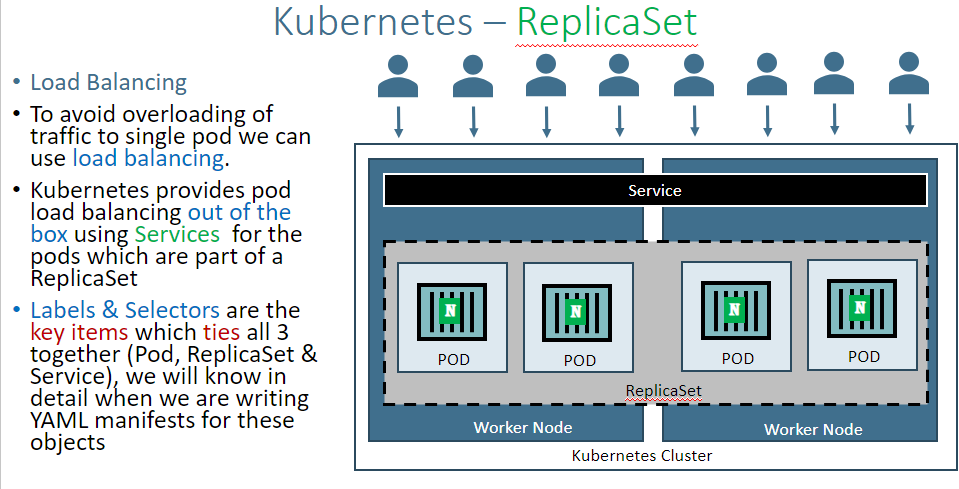
Get pod definition YAML output  
kubectl get pod my-first-pod -o yaml   
  
Get service definition YAML output  
kubectl get service my-first-service -o yaml

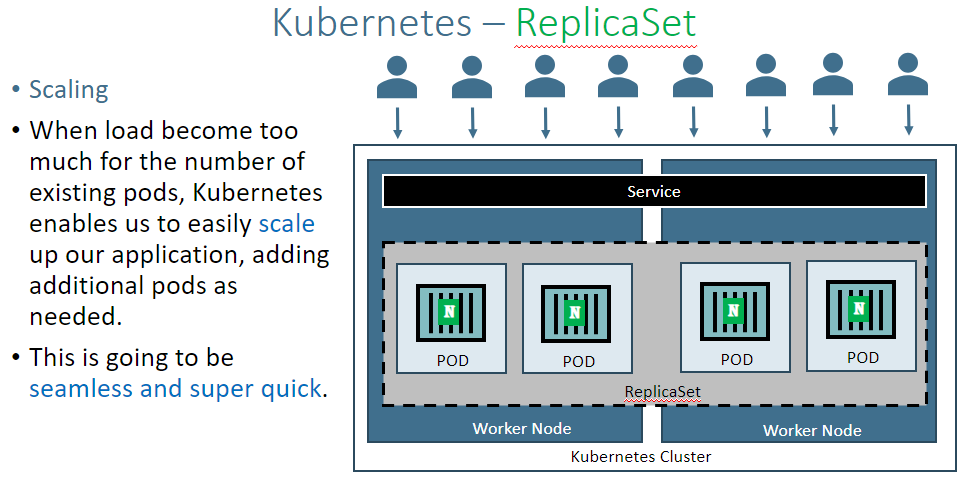
**Delete Services**  
kubectl delete svc my-first-service  
  
**Delete Pod**  
kubectl delete pod my-first-pod

**REPLICASET**









**Create ReplicaSet**

kubectl create -f replicaset-demo.yml

**replicaset-demo.yml**

apiVersion: apps/v1  
kind: ReplicaSet  
metadata:  
 name: my-helloworld-rs  
 labels:  
 app: my-helloworld  
spec:  
 replicas: 3  
 selector:  
 matchLabels:  
 app: my-helloworld  
 template:  
 metadata:  
 labels:  
 app: my-helloworld  
 spec:  
 containers:  
 - name: my-helloworld-app  
 image: stacksimplify/kube-helloworld:1.0.0

**Get list of ReplicaSets**

kubectl get replicaset

### **Describe ReplicaSet**

Describe the newly created ReplicaSet

kubectl describe rs <replicaset-name>  
  
kubectl describe rs my-helloworld-rs

**Expose ReplicaSet with a service (NodePort Service) to access the application externally (from internet)**  
kubectl expose rs <ReplicaSet-Name> --type=NodePort --port=80 --target-port=8080 --name=<Service-Name-To-Be-Created>  
kubectl expose rs my-helloworld-rs --type=NodePort --port=80 --target-port=8080 --name=my-helloworld-rs-service

Verify the new pod got created automatically  
kubectl get pods (Verify Age and name of new pod)

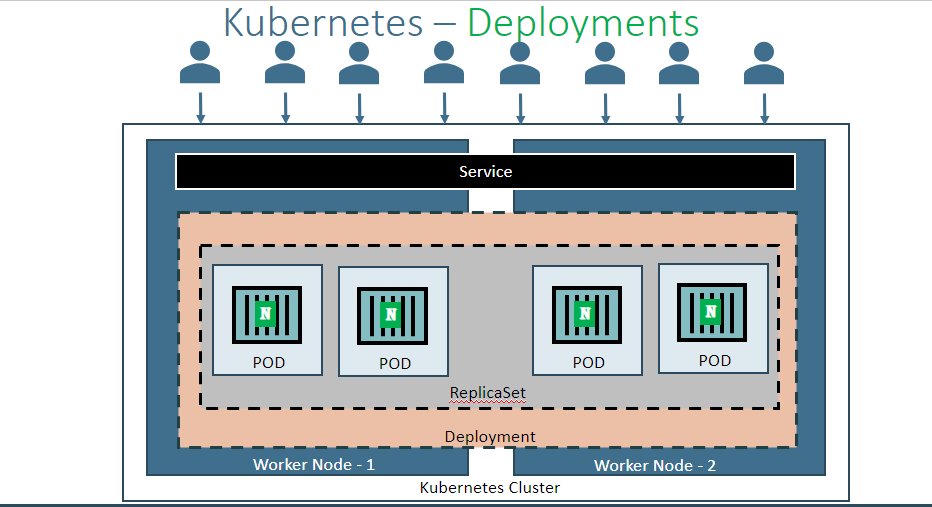
To change the no of replica - Make change in yaml fil

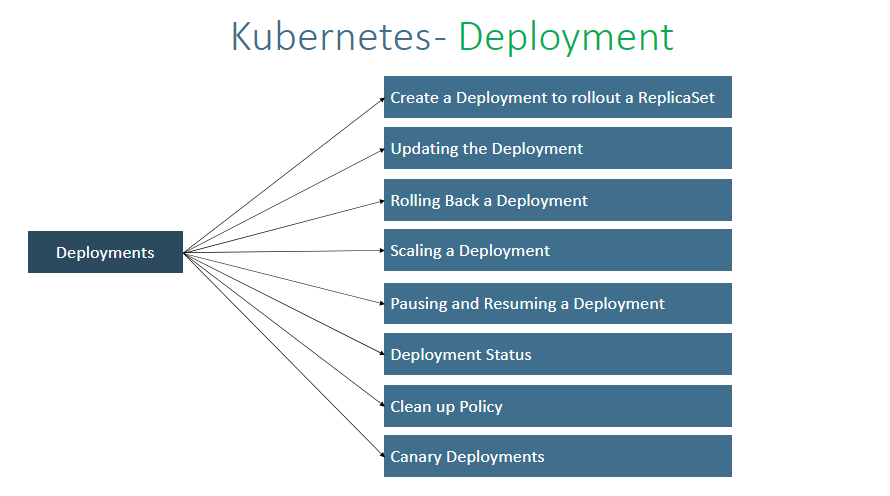
**Apply latest changes to ReplicaSet**  
kubectl replace -f replicaset-demo.yml

**Delete ReplicaSe**t  
kubectl delete rs <ReplicaSet-Name>

**Delete Service**  
kubectl delete svc <service-name>

## **Deployment**





**Create Deployment**  
kubectl create deployment <Deplyment-Name> --image=<Container-Image>  
kubectl create deployment my-first-deployment --image=stacksimplify/kubenginx:1.0.0   
  
**Verify Deployment**  
kubectl get deployments  
  
**Describe Deployment**  
kubectl describe deployment <deployment-name>  
kubectl describe deployment my-first-deployment  
  
**Verify ReplicaSet**  
kubectl get rs

**Scale Up the Deployment**  
kubectl scale --replicas=20 deployment/<Deployment-Name>  
kubectl scale --replicas=20 deployment/my-first-deployment   
  
**Scale Down the Deployment**  
kubectl scale --replicas=10 deployment/my-first-deployment