**EKSCTL Important commands**

**EKSCTL cluster creating**

**# Create Cluster, Create EKS Cluster using eksctl**

eksctl create cluster --name=eksdemo1 \

--region=us-east-1 \

--zones=us-east-1a,us-east-1b \

--without-nodegroup

--- note - It will take 15 to 20 minutes to create the Cluster Control Plane

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

eksctl utils associate-iam-oidc-provider \

--region us-east-1 \

--cluster eksdemo1\

--approve

--- **note** - 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)

**Node Group Creating**

**# 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** - 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.

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

**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

--- Important Note-1: We can’t stop our EC2 Instances which are in Kubernetes cluster unlike regular EC2 Instances. So, we need to delete the worker nodes (Node Group) if we are not using it during our learning process.

**Important Notes**

Note-1: Rollback any Security Group Changes

* When we create a EKS cluster using eksctl it creates the worker node security group with only port 22 access.
* When we progress through the course, we will be creating many NodePort Services to access and test our applications via browser.
* During this process, we need to add an additional rule to this automatically created security group, allowing access to our applications we have deployed.
* So, the point we need to understand here is when we are deleting the cluster using eksctl, its core components should be in same state which means roll back the change we have done to security group before deleting the cluster.
* In this way, cluster will get deleted without any issues, else we might have issues and we need to refer cloudformation events and manually delete few things. In short, we need to go to many places for deletions.

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

* When we are doing EBS Storage Section with EBS CSI Driver we will add a custom policy to worker node IAM role.
* When you are deleting the cluster, first roll back that change and delete it.
* This way we don't face any issues during cluster deletion.

**Delete Node Group**

--- **Note** - We can delete a nodegroup separately using below eksctl delete nodegroup.

**# List EKS Clusters**

--- eksctl get clusters

**# Capture Node Group name**

--- eksctl get nodegroup --cluster=<clusterName>

--- eksctl get nodegroup --cluster=eksdemo1

**# Delete Node Group**

--- eksctl delete nodegroup --cluster=<clusterName> --name=<nodegroupName>

--- eksctl delete nodegroup --cluster=eksdemo1 --name=eksdemo1-ng-public1

**Delete Cluster**

--- **note** - We can delete cluster using eksctl delete cluster

**# Delete Cluster**

--- eksctl delete cluster <clusterName>

--- eksctl delete cluster eksdemo1

**Create a Pod**

**# Template**

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

**# Replace Pod Name, Container Image**

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

* Important Note: Without --generator=run-pod/v1 it will create a pod with a deployment which is another core kubernetes concept which we will learn in next few minutes.

--- Important Note:

* 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**

**# To get list of pod names**

--- kubectl get pods

**# Alias name for pods is po**

--- kubectl get po

**#List pods with wide option which also provide Node information on which Pod is running**

--- kubectl get pods -o wide

**# Describe the Pod**

--- kubectl describe pod <Pod-Name>

--- kubectl describe pod my-first-pod

**# Delete Pod**

--- kubectl delete pod <Pod-Name>

--- kubectl delete pod my-first-pod

**Expose Pod with a Service**

**# 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

--- kubectl get svc

**Access the Application using Public IP**

--- **Error! Hyperlink reference not valid.**>

**Pod Logs**

**# Dump Pod logs**

--- kubectl logs <pod-name>

--- kubectl logs my-first-pod

**# Stream pod logs with -f option and access application to see logs**

--- kubectl logs <pod-name>

--- kubectl logs -f my-first-pod

**Connect to Container in a POD**

**# Connect to Nginx Container in a POD**

--- kubectl exec -it <pod-name> -- /bin/bash

--- kubectl exec -it my-first-pod -- /bin/bash

**# Execute some commands in Nginx container**

--- ls

--- cd /usr/share/nginx/html

--- cat index.html

--- exit

**Running individual commands in a Container**

**# Sample Commands**

--- 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

**Clean-Up**

**# Get all Objects in default namespace**

--- kubectl get all

**# Delete Services**

--- kubectl delete svc my-first-service

--- kubectl delete svc my-first-service2

--- kubectl delete svc my-first-service3

**# Delete Pod**

--- kubectl delete pod my-first-pod

**# Get all Objects in default namespace**

--- kubectl get all

**Create replicaset**

**#Create ReplicaSet**

--- kubectl create -f replicaset-demo.yml

**# Describe the newly created ReplicaSet**

--- kubectl describe rs/my-helloworld-rs

--- kubectl describe rs my-helloworld-rs

**Expose ReplicaSet as a Service**

**# Expose ReplicaSet as a Service**

--- 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

**# Get Service Info**

--- kubectl get service

--- kubectl get svc

**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

--- kubectl get deploy

**# Describe Deployment**

--- kubectl describe deployment <deployment-name>

--- kubectl describe deployment my-first-deployment

**# Verify ReplicaSet**

--- kubectl get rs

**# Verify Pod**

--- kubectl get po

**Scaling a Deployment**

**# Scale Up the Deployment**

--- kubectl scale --replicas=20 deployment/<Deployment-Name>

--- kubectl scale --replicas=20 deployment/my-first-deployment

**# Verify Deployment**

--- kubectl get deploy

**# Scale Down the Deployment**

--- kubectl scale --replicas=10 deployment/my-first-deployment

--- kubectl get deploy

**Expose Deployment as a Service**

**# Expose Deployment as a Service**

--- kubectl expose deployment <Deployment-Name> --type=NodePort --port=80 --target-port=80 --name=<Service-Name-To-Be-Created>

--- kubectl expose deployment my-first-deployment --type=NodePort --port=80 --target-port=80 --name=my-first-deployment-service

**# Get Service Info**

--- kubectl get svc

Observation: Make a note of port which starts with 3 (Example: 80:3xxxx/TCP). Capture the port 3xxxx and use it in application URL below.

**# Get Public IP of Worker Nodes**

--- kubectl get nodes -o wide

Observation: Make a note of "EXTERNAL-IP" if your Kubernetes cluster is setup on AWS EKS.

**Updating Application version V1 to V2 using "Set Image" Option**

**# Get Container Name from current deployment**

--- kubectl get deployment my-first-deployment -o yaml

**# Update Deployment - SHOULD WORK NOW**

--- kubectl set image deployment/<Deployment-Name> <Container-Name>=<Container-Image> --record=true

--- kubectl set image deployment/my-first-deployment kubenginx=stacksimplify/kubenginx:2.0.0 --record=true

**Verify Rollout Status (Deployment Status)**

--- **Observation:** By default, rollout happens in a rolling update model, so no downtime.

**# Verify Rollout Status**

--- kubectl rollout status deployment/my-first-deployment

**# Verify Deployment**

--- kubectl get deploy

**Update the Application from V2 to V3 using "Edit Deployment" Option**

--- **note** – this is the option we use most of the time, whenever we have running deployment.

# Edit Deployment

--- kubectl edit deployment/<Deployment-Name> --record=true

--- kubectl edit deployment/my-first-deployment --record=true

# Change From 2.0.0

    spec:

      containers:

      - image: stacksimplify/kubenginx:2.0.0

# Change To 3.0.0

    spec:

      containers:

      - image: stacksimplify/kubenginx:3.0.0

--- **Observation:** Rollout happens in a rolling update model, so no downtime.

**# Verify Rollout Status**

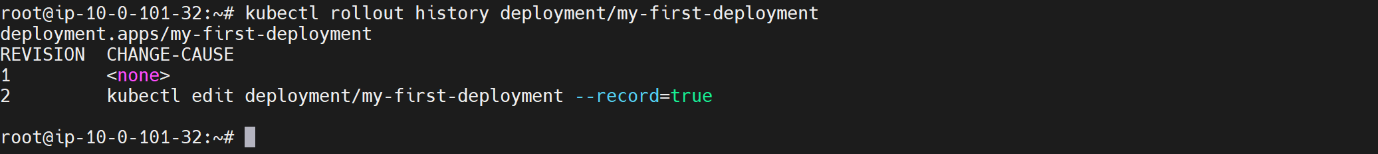
--- kubectl rollout status deployment/my-first-deployment

**Rollback a Deployment to previous version**

**# List Deployment Rollout History**

--- kubectl rollout history deployment/<Deployment-Name>

--- kubectl rollout history deployment/my-first-deployment



**Verify changes in each revision**

**# List Deployment History with revision information**

--- kubectl rollout history deployment/my-first-deployment --revision=1

--- kubectl rollout history deployment/my-first-deployment --revision=2

--- kubectl rollout history deployment/my-first-deployment --revision=3



--- **note** – it will clearly tell us, what happened in the revision 1

**Rollback to previous version**

--- Observation: If we rollback, it will go back to revision-2 and its number increases to revision-4

**# Undo Deployment**

--- kubectl rollout undo deployment/my-first-deployment

**# List Deployment Rollout History**

--- kubectl rollout history deployment/my-first-deployment

**Rollback to specific revision**

**# Rollback Deployment to Specific Revision**

--- kubectl rollout undo deployment/my-first-deployment --to-revision=3

**Pause Deployment and Two Changes**

**# Pause the Deployment**

--- kubectl rollout pause deployment/<Deployment-Name>

kubectl rollout pause deployment/my-first-deployment

**# Update Deployment - Application Version from V3 to V4**

--- kubectl set image deployment/my-first-deployment kubenginx=stacksimplify/kubenginx:4.0.0 --record=true

**# Check the Rollout History of a Deployment**

--- kubectl rollout history deployment/my-first-deployment

Observation: No new rollout should start, we should see same number of versions as we check earlier with last version number matches which we have noted earlier.

**# Get list of ReplicaSets**

--- kubectl get rs

Observation: No new replicaSet created. We should have same number of replicaSets as earlier when we took note.

**# Make one more change: set limits to our container**

--- kubectl set resources deployment/my-first-deployment -c=kubenginx --limits=cpu=20m,memory=30Mi

**Get the IAM role Worker Nodes using and associate this policy to that role**

**# Get Worker node IAM Role ARN**

--- kubectl -n kube-system describe configmap aws-auth

**# from output check rolearn**

--- rolearn: arn:aws:iam::180789647333:role/eksctl-eksdemo1-nodegroup-eksdemo-NodeInstanceRole-IJN07ZKXAWNN

**Deploy Amazon EBS CSI Driver**

--- **note** - Verify kubectl version, it should be 1.14 or later

--- kubectl version --client --short

**# Deploy EBS CSI Driver**

--- kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/deploy/kubernetes/overlays/stable/?ref=master"

**# Verify ebs-csi pods running**

--- kubectl get pods -n kube-system

**Create Persistent Volume Claims manifest**

**# Create Storage Class & PVC**

--- kubectl apply -f kube-manifests/

**# List Storage Classes**

--- kubectl get sc

**# List PVC**

--- kubectl get pvc

**# List PV**

--- kubectl get pv

**# list pv,sc,pvc**

--- kubectl get pc,pvc,sc

**Connect to MySQL Database**

**# Connect to MYSQL Database**

--- kubectl run -it --rm --image=mysql:5.6 --restart=Never mysql-client -- mysql -h mysql -pdbpassword11

**# Verify usermgmt schema got created which we provided in ConfigMap**

--- mysql> show schemas;