### Canary Deployment on Kubernetes

Create a deployment using below yaml to deploy pods for our web-blue app

vi web-blue.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: web-blue

spec:

replicas: 3

selector:

matchLabels:

app: web-blue

type: web-app

strategy:

type: RollingUpdate

template:

metadata:

labels:

app: web-blue

type: web-app

spec:

containers:

- image: mandarct/web-blue:v1

name: web-blue

ports:

- containerPort: 80

protocol: TCP

A screenshot of a computer

Description automatically generated with medium confidence

Deploy the above deployment to the Kubernetes cluster in the default namespace

kubectl apply -f web-blue.yaml

Verify that pods are running

**kubectl get po**

NAME READY STATUS RESTARTS AGE

web-blue-5657b94c87-cqkfz 1/1 Running 0 12m

web-blue-5657b94c87-rwcfj 1/1 Running 0 12m

web-blue-5657b94c87-vgsqv 1/1 Running 0 12m

Schematic

Description automatically generated with medium confidence

Create a service of type Load-balancer to expose above deployment using below yaml

vi svc-web-lb.yaml

apiVersion: v1

kind: Service

metadata:

name: web-app-svc-lb

spec:

ports:

- port: 80

protocol: TCP

targetPort: 80

selector:

type: web-app

type: LoadBalancer

ports:

- port: 80

targetPort: 80

Deploy this Load-Balancer service to the default namespace

kubectl apply -f svc-web-lb.yaml

Verify the service is created of type load-balancer

kubectl get svc web-app-svc-lb

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

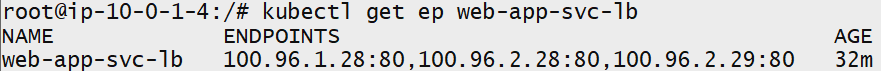
web-app-svc-lb LoadBalancer 100.67.144.247 a72cd7c5e674044e4b09e34ae1848acd-702623717.ap-south-1.elb.amazonaws.com 80:30229/TCP 18m

Verify the end-point object is created pointing to the IP address for web-blue pods

**kubectl get ep web-app-svc-lb**

NAME ENDPOINTS AGE

web-app-svc-lb 100.96.1.28:80,100.96.2.28:80,100.96.2.29:80 32m



Search for a72cd7c5e674044e4b09e34ae1848acd-702623717.ap-south-1.elb.amazonaws.com in

AWS -> EC2 Dashboard -> Load balancers

Verify that a new ELB has been created in AWS. Wait for 2 minutes for the ELB instances to be in-service.

Graphical user interface, text, application, email

Description automatically generated

Test the ELB DNS URL from your browser, you should get below response from the web-blue app pods.

Graphical user interface, text, application, email

Description automatically generated

Create another deployment using below yaml

vi web-green.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: web-green

spec:

replicas: 3

selector:

matchLabels:

app: web-green

strategy:

type: RollingUpdate

template:

metadata:

labels:

app: web-green

type: web-app

spec:

containers:

- image: mandarct/web-green:v1

name: web-green

ports:

- containerPort: 80

protocol: TCP

Deploy the above deployment to the Kubernetes cluster in the default namespace

kubectl apply -f web-green.yaml

Verify pods running for web-green deployment as well

**kubectl get po**

NAME READY STATUS RESTARTS AGE

web-blue-5657b94c87-cqkfz 1/1 Running 0 25m

web-blue-5657b94c87-rwcfj 1/1 Running 0 25m

web-blue-5657b94c87-vgsqv 1/1 Running 0 25m

web-green-76df95dbcd-4bnkf 1/1 Running 0 27m

web-green-76df95dbcd-57v5x 1/1 Running 0 27m

web-green-76df95dbcd-rhmvk 1/1 Running 0 27m

Text, table

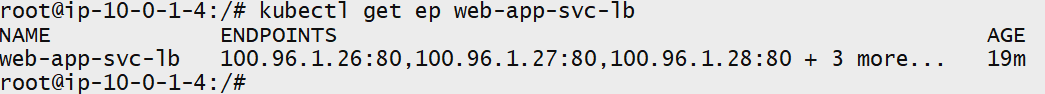
Description automatically generated

Verify that the end-points for the existing load-balancer service are updated with pods for web-green deployment

root@ip-10-0-1-4:/# kubectl get ep web-app-svc-lb

NAME ENDPOINTS AGE

web-app-svc-lb 100.96.1.26:80,100.96.1.27:80,100.96.1.28:80 + 3 more... 19m



Hit the load balancer (ELB) URL from web-browser, multiple times. You should be below 2 outputs, as the traffic is routed between the 2 deployments (web-blue & web-green)

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Once we have deployed both blue and green versions of our deployments, we notice that pods are created with below labels. Our Load balancer service is created with matching labels for ‘type=web-app’, so the traffic is distributed (load balanced) across both the versions of our deployments

A picture containing graphical user interface

Description automatically generated

If you delete the web-green deployment, load-balancer will start sending traffic only to the blue pods

**kubectl delete deploy web-green**

deployment.apps "web-green" deleted

Icon

Description automatically generated with low confidence

The end point object for load balancer service will be back pointing only to the IP address for web-blue pods

**kubectl get ep web-app-svc-lb**

NAME ENDPOINTS AGE

web-app-svc-lb 100.96.1.28:80,100.96.2.28:80,100.96.2.29:80 32m

A picture containing text, orange, close

Description automatically generated

Same can be verified by describing the service

**kubectl describe svc web-app-svc-lb**

Name: web-app-svc-lb

Namespace: default

Labels: <none>

Annotations: kubectl.kubernetes.io/last-applied-configuration:

{"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"name":"web-app-svc-lb","namespace":"default"},"spec":{"ports":[{"port":8...

Selector: type=web-app

Type: LoadBalancer

IP: 100.67.144.247

LoadBalancer Ingress: a72cd7c5e674044e4b09e34ae1848acd-702623717.ap-south-1.elb.amazonaws.com

Port: <unset> 80/TCP

TargetPort: 80/TCP

NodePort: <unset> 30229/TCP

Endpoints: 100.96.1.28:80,100.96.2.28:80,100.96.2.29:80

Session Affinity: None

External Traffic Policy: Cluster

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal EnsuringLoadBalancer 38m service-controller Ensuring load balancer

Normal EnsuredLoadBalancer 38m service-controller Ensured load balancer

Try hitting the ELB URL from web-browser multiple times, you should only see the response from web-app-blue.

Graphical user interface, application

Description automatically generated