**Session - 72**

it is zero downtime strategy, for example if the current running version is blue, we will create green version with similar capacity and we will do health check. then we switch the routing from blue to green if everything is good. blue can be still backup version to rollback if we get any issues in green.

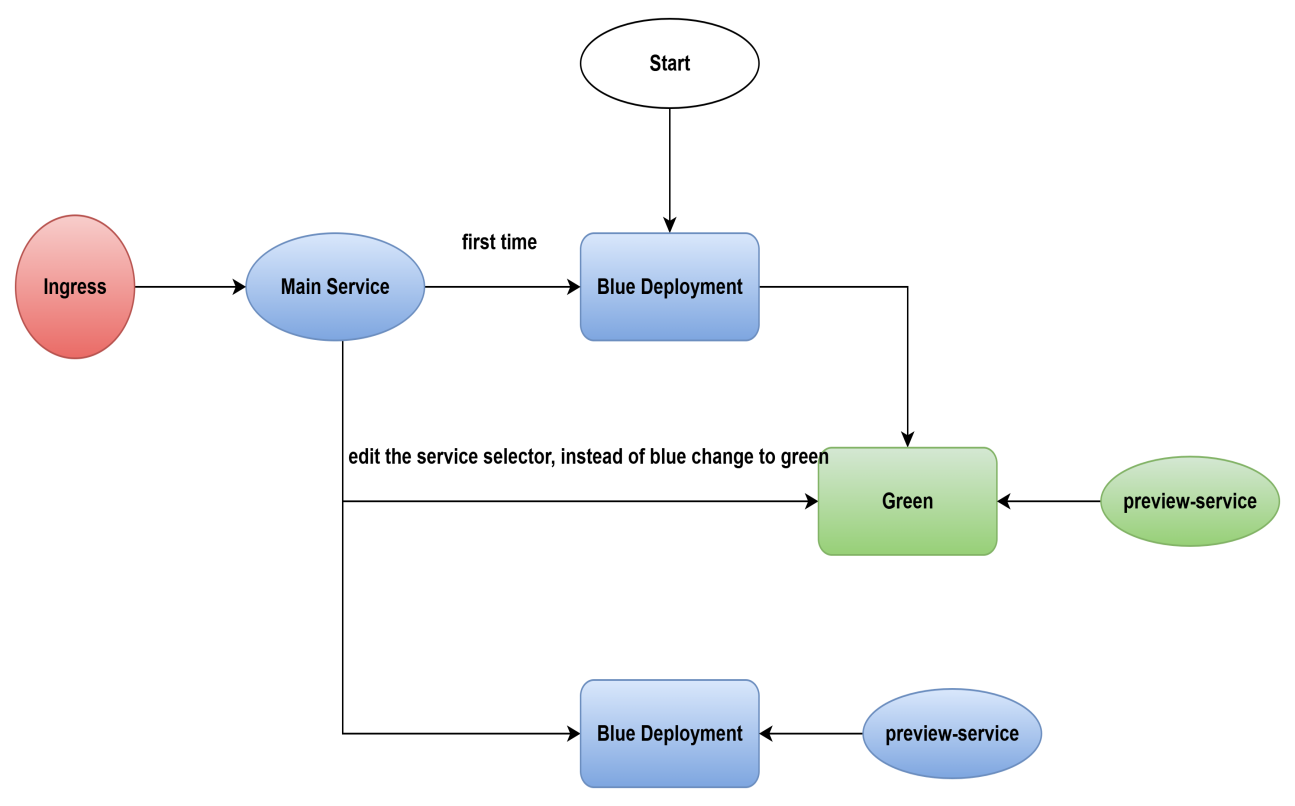
we are using this in our eks upgrade. where blue is the running node group, we create green node group. upgrade control plane and green node group switch the workloads from blue to green.

purpose: service-demo

project: roboshop

environment: dev

color: blue

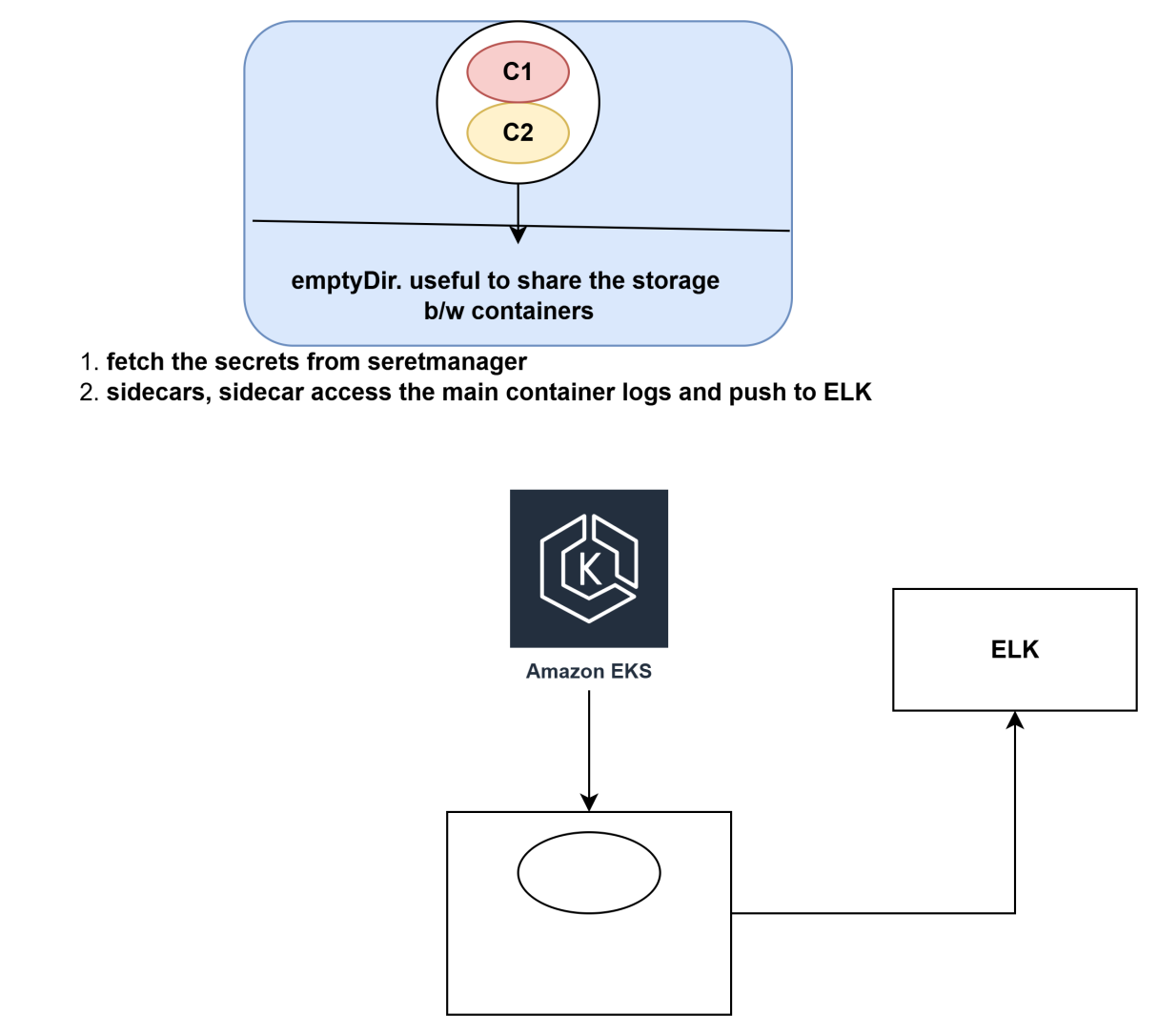


--> This is about blue deployment.

--> ephemeral volumes:

- emptyDir

- hostPath --> don't use this, use only for daemonset



ReplicaSet

Deployment

StatefulSet

DaemonSet

DaemonSet make sure a replica runs on each and every node, this is mostly useful to collect the host logs and we can send them external log storage.

liveness probe -> a person is alive or not. just heart beat. if not alive k8 attempts restart

readiness probe -> when he is ready to work. is pod ready to accept the traffic. then end point gets registered with service

k8 uses liveness probe and rediness probe for self healing.

**k8-resources/18-daemonset.yaml**

apiVersion: apps/v1

kind: DaemonSet

metadata:

name: fluentd-elasticsearch

namespace: kube-system

labels:

k8s-app: fluentd-logging

spec:

selector:

matchLabels:

name: fluentd-elasticsearch

template:

metadata:

labels:

name: fluentd-elasticsearch

spec:

tolerations:

# these tolerations are to have the daemonset runnable on control plane nodes

# remove them if your control plane nodes should not run pods

- key: node-role.kubernetes.io/control-plane

operator: Exists

effect: NoSchedule

- key: node-role.kubernetes.io/master

operator: Exists

effect: NoSchedule

containers:

- name: fluentd-elasticsearch

# this pod collects the host logs and send them to ELK

image: quay.io/fluentd\_elasticsearch/fluentd:v5.0.1

resources:

limits:

memory: 200Mi

requests:

cpu: 100m

memory: 200Mi

volumeMounts:

- name: varlog

mountPath: /var/log

readOnly: true

# it may be desirable to set a high priority class to ensure that a DaemonSet Pod

# preempts running Pods

# priorityClassName: important

terminationGracePeriodSeconds: 30

volumes:

- name: varlog

hostPath:

path: /var/log

**--> aws eks update-kubeconfig --region us-east-1 --name roboshop-dev**

**--> kubectl get nodes**

**--> git clone k8-resources.git**

**--> cd k8-resources/**

**--> kubectl apply -f 18-daemonset.yaml**

**--> kubectl get pods**

**--> kubectl get pods -n kube-system**

--> catalogue -> mongodb

/health --> 200 then it is live

success code but mongodb false, then it is not ready

**k8 architecture**

**--------------------**

master and nodes

**19-live-readiness.yaml**

apiVersion: v1

kind: Pod

metadata:

labels:

test: liveness

name: liveness-http

spec:

containers:

- name: liveness

image: nginx

livenessProbe:

tcpSocket:

port: 80

initialDelaySeconds: 30

periodSeconds: 10

readinessProbe:

httpGet:

path: /

port: 80

initialDelaySeconds: 40

periodSeconds: 3

**--> cd k8-resources**

**--> git pull**

**--> kubectl apply -f 19-live-readiness.yaml**

**--> kubectl get pods**

**--> watch kubectl get pods**

**-->** NetworkPolicy

frontend -> catalogue -> mongodb

mongodb\_catalogue

**20-network-policy.yaml**

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: mongodb-accept-catalogue

namespace: roboshop

spec:

podSelector:

matchLabels:

tier: database

component: mongodb

policyTypes:

- Ingress

ingress:

- from:

- podSelector:

matchLabels:

component: catalogue

project: roboshop

tier: app

ports:

- protocol: TCP

port: 27017

**--> kubectl apply -f 20-network-policy.yaml**

**--> kubectl create namespace roboshop**

**--> kubectl apply -f 20-network-policy.yaml**

**--> kubectl get np -n roboshop**

**--> kubectl get networkpolicy -n roboshop**

**--> kubectl describe networkpolicy mongodb-accept-catalogue -n roboshop**

**-->**  catalogue -> mongodb

/health --> 200 then it is live

success code but mongodb false, then it is not ready

NetworkPolicy

frontend -> catalogue -> mongodb

mongodb\_catalogue

**k8 architecture**

**--------------------**

master and nodes

master

===========

api server --> all cluster interactions comes to api server. validates RBAC and provide the response. if you are creating pods, api server validates and handover to scheduler.

scheduler --> it a component to decide where your pod should run. it checks for taints and tolerations, selectors like nodeaffinity, pod affinity, node readiness,etc.

control manager

replica controller -> make sures desired number of pod replicas run all the time.

node controller --> checks desired number of nodes are running

ServiceAccount controller --> Create default ServiceAccounts for new namespaces.

EndpointSlice controller -> Populates EndpointSlice objects (to provide a link between Services and Pods).

etcd --> key value storage for k8 cluster. all k8 configs are here.

**Node components**

**================**

kubelet --> agent running in every node. connects the nodes and master and makes sure containers are running inside pod.

kube-proxy --> networking rules how to send traffic to pods

Container runtime --> containerd, runc, etc. runs the containers

add-ons: adds capabilities to the cluster. vpc-cni, dns, metric server, etc.

cart -> catalogue

cart -> catalogue:8080

http:://catalogue-preview:8080/health --> success

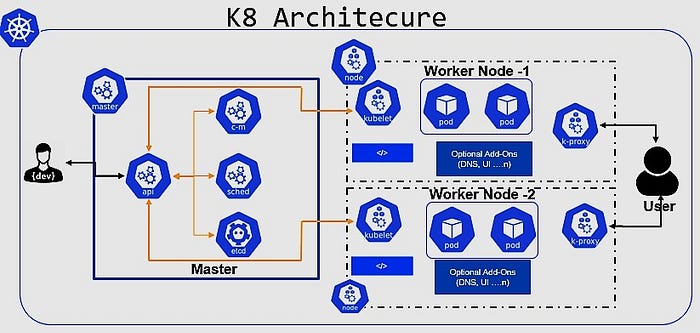
**Kubernetes Architecture and Components**

Kubernetes is extremely flexible and is capable of being deployed in many different configurations. It supports clusters as small as a single node and as large as a few thousand. It can be deployed using either physical or virtual machines on premises or in the cloud.

Kubernetes consists of two main components:

- Master (Control Plane)

Worker Nodes



1. **What does a NodePort service do in kubernates?**

**ANS))** Exposes a service on a ephemeral port an each node’s IP

1. **Which attribute is mandatory when defining an EC2 instance in Terraform?**

**ANS))** aml

1. **How do you check the events of a POD?**

**ANS))** kubectl describe pod

1. **Which component of the kubernates architecture manages the actual state of the cluster?**

**ANS))** etcd

1. **What is the function of the kube-proxy component in kubernates?**

**ANS))** Manages network rules on nodes

1. **What is a key use case for using DaemonSets in Kubernates?**

**ANS))** Running a copy of a pod on call notes

1. **Which Component is configured to create volume dynamically?**

**ANS))** StorageClass

1. **How can you achieve high availability for your application in kubernates?**

**ANS))** Use multiple replicas and spread them across nodes

1. **Which kubernates feature allows you to specify the condition under which a pod can be scheduled on a node?**

**ANS))** Taints and tolerations

1. **What does a kubernates readiness probe check?**

**ANS))** If a container is ready to start accepting traffic

1. **Which kubernates resource is used to manage a group of identical pods?**

**ANS))** Replicaset

1. **What is the function of a kubernates ingress resource?**

**ANS))** Manage external access to service in a cluster

1. **What command would you use to delete a namespace and all its resources?**

**ANS))** Kubectl delete namespace <name>

1. **Which command roll back a deployment to a previous revision?**

**ANS))** Kubectl rollout undo

1. **Which resource you should use to mount the volume to POD?**

**ANS))** PVC