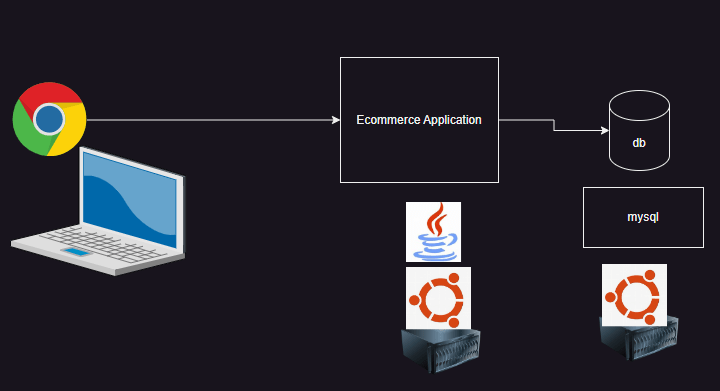
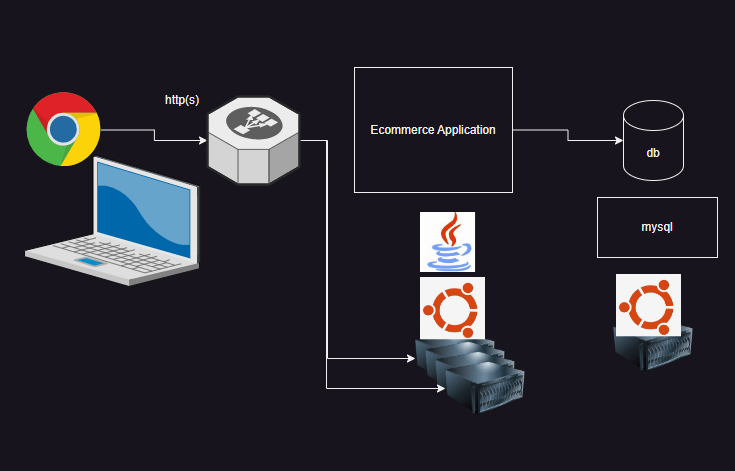
October 16, 2024

DevOps Classroom notes 16/Oct/2024

**Story of Kubernetes (k8s)**

**Monolithic Systems**

* Lets assume we are builing an ecommerce system  
  
* Ecommerce has different modules
  + user registration and management
  + administration
  + catalog
  + warehousing
  + logistics
  + cart
  + payment
  + notifications
* Scaling: During the seasonal sale, we have more than normal users accesing our [application](https://directdevops.blog/2024/10/16/devops-classroom-notes-16-oct-2024/), we need to scale [servers](https://directdevops.blog/2024/10/16/devops-classroom-notes-16-oct-2024/)  
  
* Application Updates:
  + We need to increase number of servers if we need zero downtime
  + Rollout the new version on servers (atleast one server should be up)
  + In the cases of problems with latest version (Rolling back) but the above steps are challenging and require expertise

**Microservices**

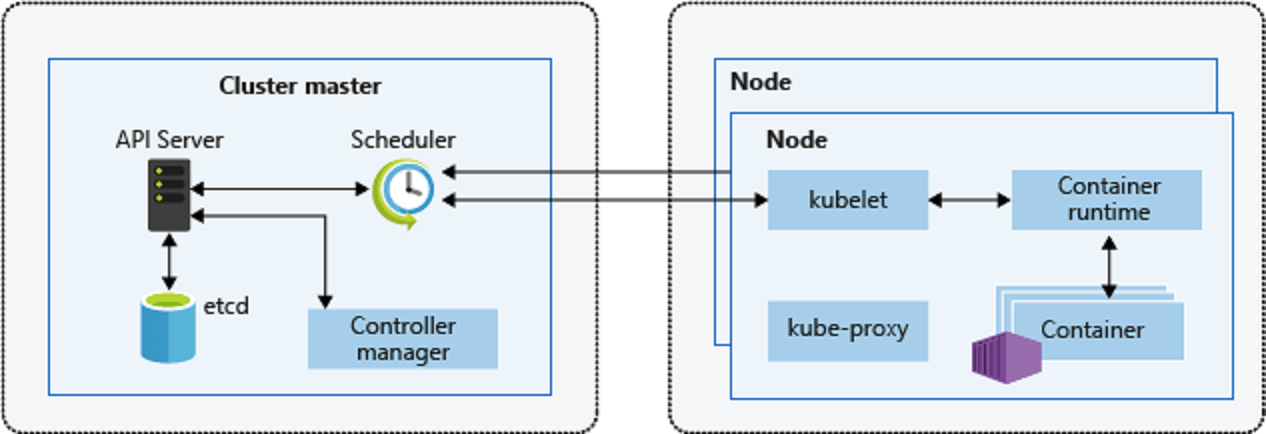
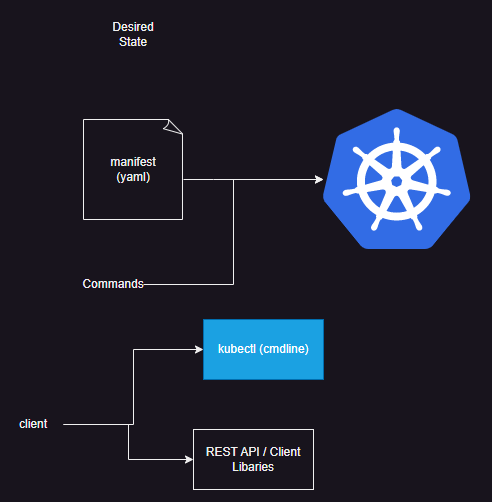
* Applications are broken down into individually runnable services
* Read the comic book [Refer Here](https://cloud.google.com/kubernetes-engine/kubernetes-comic)
* Google has a history of running containers and to operate on this containers they have built orchestration systems
  + OMEGA
  + BORG
* Docker was released and Was a popular option to run containers
* Google has written the orchestration system based on their experince with containers in GOlang which is called as Kubernetes and made it opensource.
* Kubernetes, also known as K8s, is an open source system for automating deployment, scaling, and management of containerized applications.
* Running containers for Production Scenarios is what k8s does.
* Alternatives to k8s
  + Docker Swarm
  + Apache Mesos
  + AWS ECS
* Kubernetes creates a cluster which is combination of multiple nodes categorized as
  + Worker node (node):
    - Here the application workloads are executed
  + Master Node:
    - They manage the cluster

**Share this:**

October 17, 2024

# DevOps Classroom notes 17/Oct/2024

## Kubernetes Architecture

* [Refer Here](https://directdevops.blog/2019/10/09/kubernetes-introduction/) for introduction to k8s  
  
* We express our desired state in yaml or imperative command and use a client (kubectl) to communicate with k8s and k8s does the rest  
  
* For master and node components [Refer Here](https://directdevops.blog/2019/10/10/kubernetes-master-and-node-components/)

## Ways to setup Kubernetes

* Single System (Developer): This is workstation environment for developmental purposes, Here we have following major options
  + minikube
  + kind
* Self Hosted
  + kubeadm
  + kubespray (install k8s using ansible )
* [Cloud](https://directdevops.blog/2024/10/17/devops-classroom-notes-17-oct-2024/) Hosted

Cloud hosting

* + AWS: Elastic Kubernetes Services (EKS)
  + Azure: Azure Kuberenetes Services (AKS)
  + GCP: Google Kubernetes Engine (GKE)

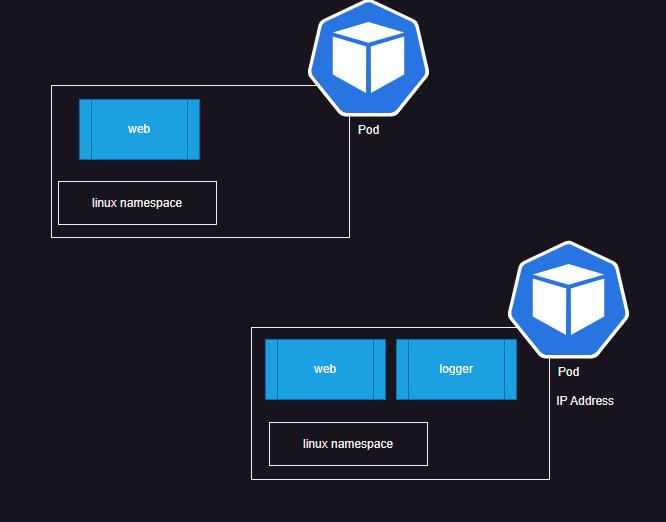
## Kubernetes stopped supporting docker

* kubernetes uses a interface called as CRI (Container Runtime Interface)
* kubernetes used to ship dockershim to interact with docker containers
* In K8s release 1.21 dockershim was removed [Refer Here](https://kubernetes.io/blog/2020/12/02/dont-panic-kubernetes-and-docker/)
* Mirantis and many other communties have developed CRI for docker [Refer Here](https://github.com/Mirantis/cri-dockerd)

## Kubernetes workloads

* Pod
* ReplicaSet/ReplicationController
* Deployment
* DaemonSet
* StatefulSet
* Service
* Ingress
* Labels
* Namespaces
* …

# Pod

* This is the smallest unit of creation by k8s
* A Pod has one or more containers in it
* Every Pod gets a unique ipaddress  
  

October 18, 2024

# DevOps Classroom notes 18/Oct/2024

## K8s Installation (Self hosted)

* Lab setup
  + 3 [ubuntu](https://directdevops.blog/2024/10/18/devops-classroom-notes-18-oct-2024/) 22.04
  + ensure they can reach each other
* For installation of k8s we will be using kubeadm
* High Level overview
  + install docker on all nodes
  + install cri-dockerd on all nodes
  + installing kubeadm, kubectl on all nodes
  + initialize the cluster on master node and this command gives join command which we will be executing on nodes
  + Configure kubectl
  + For Networking between Pods, Kubernetes needs CNI-Plugins
  + We will be installing Flannel CNI Plugins
* [Refer Here](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/) for kubeadm install
* for user data

#!/bin/bash

curl -fsSL https://get.docker.com -o install-docker.sh

sh install-docker.sh

* Installing CRI-dockerd [Refer Here](https://github.com/Mirantis/cri-dockerd/releases) for releases page on all nodes

cd /tmp

wget https://github.com/Mirantis/cri-dockerd/releases/download/v0.3.15/cri-dockerd\_0.3.15.3-0.ubuntu-jammy\_amd64.deb

sudo dpkg -i cri-dockerd\_0.3.15.3-0.ubuntu-jammy\_amd64.deb

* Install kubeadm, kubectl, kubelet on all nodes [Refer Here](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/#installing-kubeadm-kubelet-and-kubectl)

sudo apt-get update

# apt-transport-https may be a dummy package; if so, you can skip that package

sudo apt-get install -y apt-transport-https ca-certificates curl gpg

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

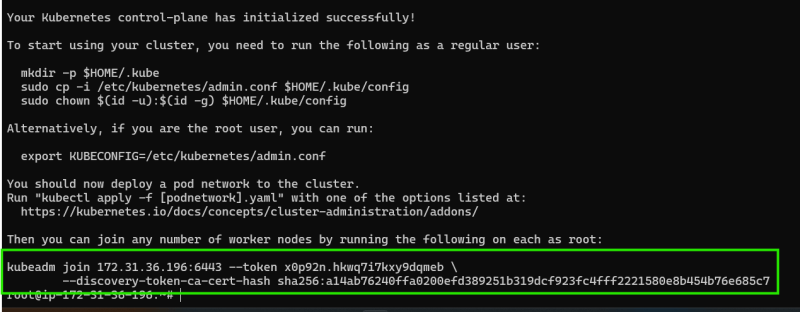
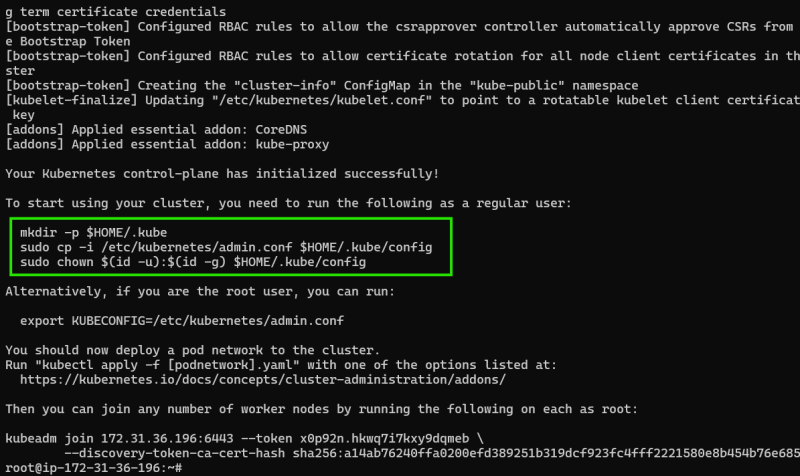
sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

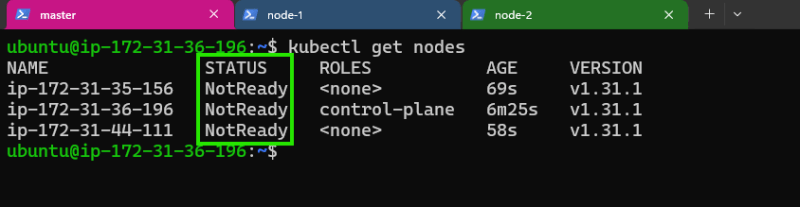
sudo systemctl enable --now kubelet

* Now login into master node and initialize the cluster [Refer Here](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/) and become a root user

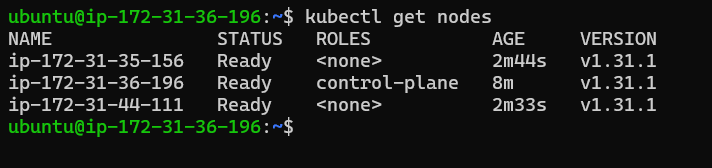
kubeadm init --pod-network-cidr=10.244.0.0/16 --cri-socket "unix:///var/run/cri-dockerd.sock"

* Now save the join command  
  
* Lets configure kubectl on master node. become a normal user  
  
* Login into node 1 and execute the join command

kubeadm join 172.31.36.196:6443 --token x0p92n.hkwq7i7kxy9dqmeb --discovery-token-ca-cert-hash sha256:a14ab76240ffa0200efd389251b319dcf923fc4fff2221580e8b454b76e685c7 --cri-socket "unix:///var/run/cri-dockerd.sock"

* repeat the same on node 2
* Now login into node 1 and execute kubectl get nodes  
  
* To fix the not ready status, we need to install pod network, lets install flannel on master node

kubectl apply -f https://github.com/coreos/flannel/raw/master/Documentation/kube-flannel.yml



### K8s Interfaces

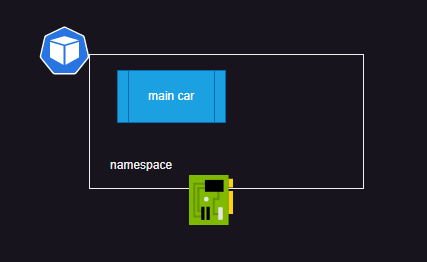
* CRI (Container Runtime Interface)
* CNI (Container Network Interface)
* CSI (Container [Storage](https://directdevops.blog/2024/10/18/devops-classroom-notes-18-oct-2024/) Interface)

October 19, 2024

# DevOps Classroom notes 19/Oct/2024

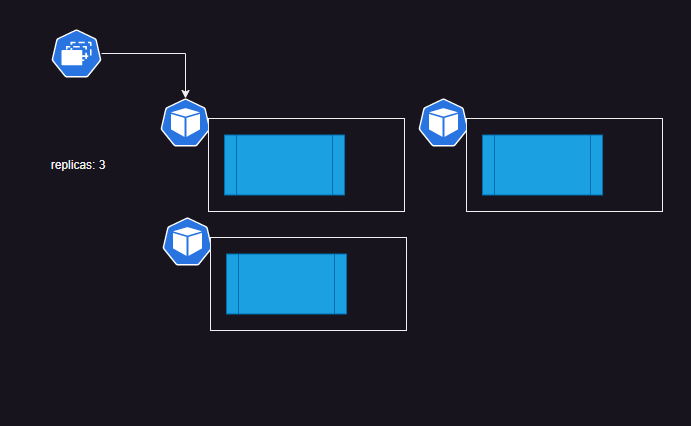
## Kubernetes Major Workloads

### Pod

* [Refer Here](https://kubernetes.io/docs/concepts/workloads/pods/) for official docs
* Pods will have one or more containers
* Each Pod gets an IP Address which is shared to the container(s)
* Pods will two types of containers in it
  + init containers: They are created in a sequence and they are expected to finish in some finite time. Generally we do precondition checks over here
  + containers: These are where we run our application images and they should run forever and they are created in parallel
* In Pod the desired state is containers, when containers fail [k8s](https://directdevops.blog/2024/10/19/devops-classroom-notes-19-oct-2024/) will continuously restart containers
* There is one more type of container called as ephemeral container which is used for debugging purposes  
  

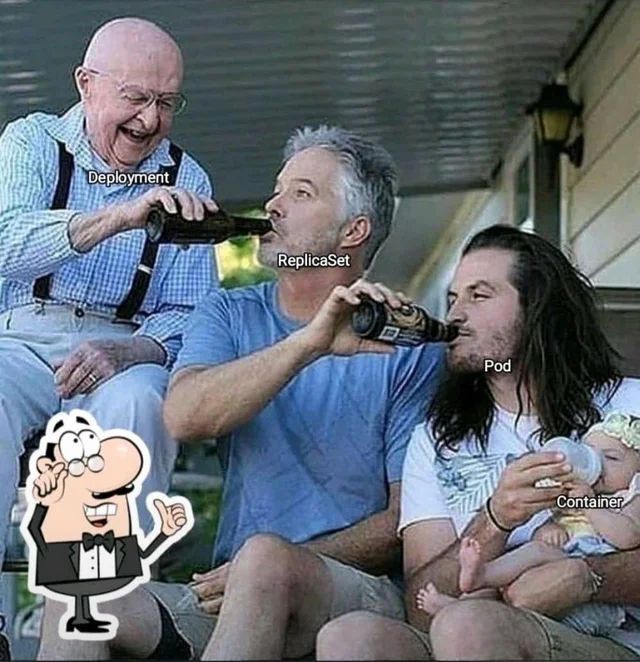
# Controllers

### ReplicaSet

* [Refer Here](https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/)  
  

### [Deployment](https://directdevops.blog/2024/10/19/devops-classroom-notes-19-oct-2024/)

* [Refer Here](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/) for deployment



### Daemonset

* This will create a pod on every node (or selected nodes) in k8s cluster
* [Refer Here](https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/)

### Statefulset

* Statefulsets are used to run stateful applications and they have capability to manage volumes as well

## Label

* this is a key value pair which can be attached to any k8s object
* k8s can query objects with the help of labels

## Service

[Refer Here](https://kubernetes.io/docs/concepts/services-networking/service/) for official docs

## Kubernetes API Server and Clients

* kubectl uses config in ~/.kube/config to get the cluster information and secrets to connect with api server [Refer Here](https://kubernetes.io/docs/reference/kubectl/)
* kubectl command line tool cheatsheet [Refer Here](https://kubernetes.io/docs/reference/kubectl/quick-reference/)
* kuberentes also provides client libraries in various languages [Refer Here](https://kubernetes.io/docs/reference/using-api/client-libraries/)
* kubectl has two styles of resource creation
  + imperative:
    - we construct a command to create a resource
    - quick and convient way to perform one time or infrequent operations.
  + declarative
    - We write a manifest in yaml format with our desired state
    - Declarative way is useful for automation, repetition and having history of changes
* Kubernetes api versioning scheme [Refer Here](https://kubernetes.io/docs/reference/using-api/#api-versioning)
* To group releated resources k8s use apiGroups [Refer Here](https://kubernetes.io/docs/reference/using-api/#api-groups)
  + core
  + apps
  + batch
  + …
* API Version: APIGROUP/version for all groups other than core. For core the APIVERSION = version
* When we are writing manifest we specify (generally)
  + apiVersion: Api version of Resource
  + kind: Type of Resource
  + metadata: name and labels
  + spec: here we define what we want
* Once we execute this kuberentes adds the fifth field status

### Share this:

October 19, 2024

# DevOps Classroom notes 19/Oct/2024

## YAML

* This is data representation format
* YAML uses key value or name value pairs to represent data
* values can be of following types
  + simple/scalar
    - text
    - number
    - bool
  + complex
    - list/array
    - map/dictionary/object
* YAML syntax is heavily inspired from [python](https://directdevops.blog/2024/10/19/devops-classroom-notes-19-oct-2024-2/) and json (javascript object notation)
* YAML files will have extension of .yaml or .yml
* yaml basic syntax

key: <value>

* Sample YAML

---

title: "Venom The Last Dance" # Text

year: 2024 # number

budget: 110.5 # number

imaxRelease: yes # boolean

genre: # list

- Action

- Adventure

- Fantasy

- Sci-Fi

starcast: # object

Venom: Tom Hardy # Text

Military Member: Chiwetel Ejiofor # Text

Scientist: Juno Temple # Text

## Writing Kubernetes Manifests

* Kuberentes manifests are strictly structured by api-references
* [Refer Here](https://kubernetes.io/docs/reference/) for reference pages and [Refer Here](https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.31/) for 1.31 api reference

### Hello Pod

* Lets start by writing a pod which will run nginx container
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/88b45e19c2a398bb9ee447facb6d1aece9c7813b) for changeset

---

apiVersion: v1

kind: Pod

metadata:

name: hello-pod

spec:

containers:

- name: web

image: nginx:1.27

* To create pods from manifest kubectl apply -f <filename.yaml>
* To get basic information

kubectl get pods <pod-name>

* to get little more info

kubectl get pods <pod-name> -o wide

* To get detailed information

kubectl describe pods <pod-name>

* To get podinfo in yaml format

kubectl get pods <pod-name> -o yaml

* To watch the changes

kubectl get pods -w

### Create a Pod with two containers

* pod name activity-2
* containers:
  + ngnix
  + alpine with command sleep 1d

---

apiVersion: v1

kind: Pod

metadata:

name: activity-2

spec:

containers:

- name: web

image: nginx:1.27

- name: sidecar

image: alpine

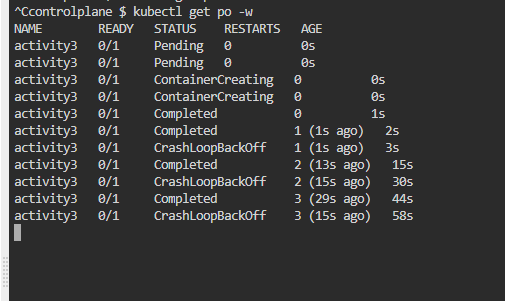
args: # cmd

- sleep

- 1d

* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/a52cb5033e0a6e9f058640bd86a80e1dad9ba6ef)

### Activity 3: Writing a Pod with a container with exits

* K8s tries restarting pod after crashloopbackoff  
  
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/a925bfc85ab2e0ac1fcfa1a68f81695b338b31af) for changes
* in the arguments lets pass sleep 10s

### Init containers

* These represent checking for preconditions

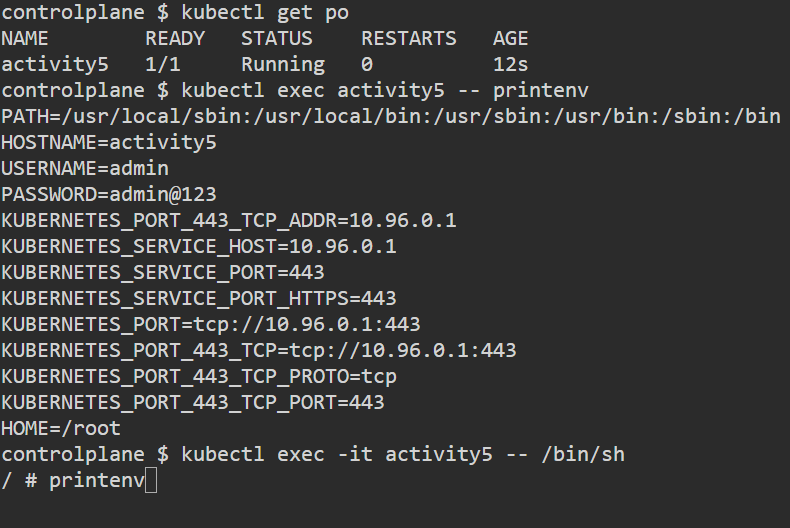
### Activity 4:

* Lets write a pod spec with nginx and alpine with sleep 1d
* init containers
  + init1 alpine with sleep 10s
  + init2 alpine with sleep 10s
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/2af4fc44791e92bebdc31379f9e7a5657c8bb369) for spec

### Activity 5

* create a pod with
  + container:
    - name: test
    - image: alpine
    - args: sleep 1d
  + we need to pass the following environmental variables
    - USERNAME = admin
    - PASSWORD = admin@123
* equivelent docker commnad

docker run -e "USERNAME=admin" -e "PASSWORD=admin@123" alpine sleep 1d

* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/52d04e1ddd9512ad43eb886d56356c3533200c16)  
  

### Lets start adding labels

* [Refer Here](https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/) for official docs
* lets create a pod with nginx container and labels
  + env: dev
  + app: web
  + release: v1.6.9

---

apiVersion: v1

kind: Pod

metadata:

name: web

labels:

env: dev

app: web

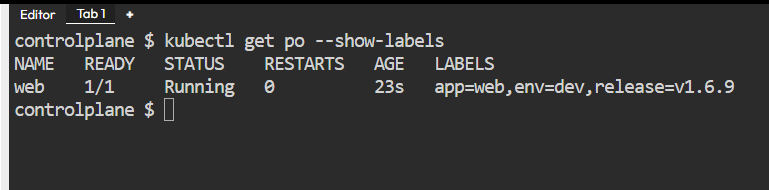
release: v1.6.9

spec:

containers:

- name: web

image: nginx:1.27



### Label selectors

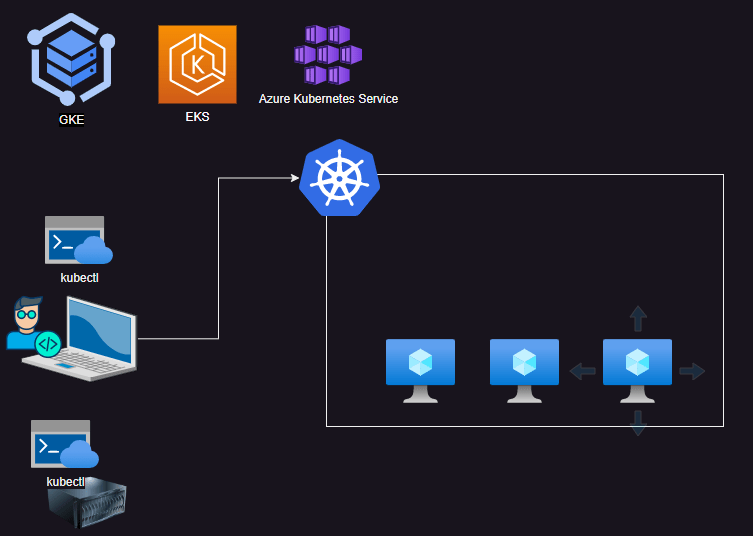
* [Refer Here](https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#label-selectors) for official docs

### Share this:

October 20, 2024

DevOps Classroom notes 20/Oct/2024

**Managed Kubernetes clusters**

* Cloud providers offer managed kubernetes clusters
* Cloud providers manage the control plane i.e. we need not install k8s and the control plane will be a black box  
  

**Azure Kubernetes Services (AKS)**

* AKS is managed k8s offered by Azure [Refer Here](https://learn.microsoft.com/en-us/azure/aks/what-is-aks)
* Lets setup AKS Cluster
* Pre-reqs:
  + Azure CLI is installed and configured
* [Refer Here](https://learn.microsoft.com/en-us/azure/aks/learn/quick-kubernetes-deploy-cli) for steps
* Azure cli user data

#!/bin/bash

sudo apt-get update

sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release -y

sudo mkdir -p /etc/apt/keyrings

curl -sLS https://packages.microsoft.com/keys/microsoft.asc |

gpg --dearmor | sudo tee /etc/apt/keyrings/microsoft.gpg > /dev/null

sudo chmod go+r /etc/apt/keyrings/microsoft.gpg

AZ\_DIST=$(lsb\_release -cs)

echo "Types: deb

URIs: https://packages.microsoft.com/repos/azure-cli/

Suites: ${AZ\_DIST}

Components: main

Architectures: $(dpkg --print-architecture)

Signed-by: /etc/apt/keyrings/microsoft.gpg" | sudo tee /etc/apt/sources.list.d/azure-cli.sources

sudo apt-get update

sudo apt-get install azure-cli -y

* Commands to create AKS cluster from bash (linux/mac)

# variables

export MY\_RESOURCE\_GROUP\_NAME="myAKSResourceGroup"

export REGION="eastus"

export MY\_AKS\_CLUSTER\_NAME="myAKSCluster"

export MY\_DNS\_LABEL="mydnslabel"

# Create a resource group

az group create --name $MY\_RESOURCE\_GROUP\_NAME --location $REGION

az aks create --resource-group $MY\_RESOURCE\_GROUP\_NAME --name $MY\_AKS\_CLUSTER\_NAME --node-count 1 --node-vm-size "Standard\_B2ms" --generate-ssh-keys

# install kubectl

az aks install-cli

# get kube config

az aks get-credentials --resource-group $MY\_RESOURCE\_GROUP\_NAME --name $MY\_AKS\_CLUSTER\_NAME

* Commands to create AKS cluster from powershell using Azure CLI

$MY\_RESOURCE\_GROUP\_NAME="myAKSResourceGroup"

$REGION="eastus"

$MY\_AKS\_CLUSTER\_NAME="myAKSCluster"

$MY\_DNS\_LABEL="mydnslabel"

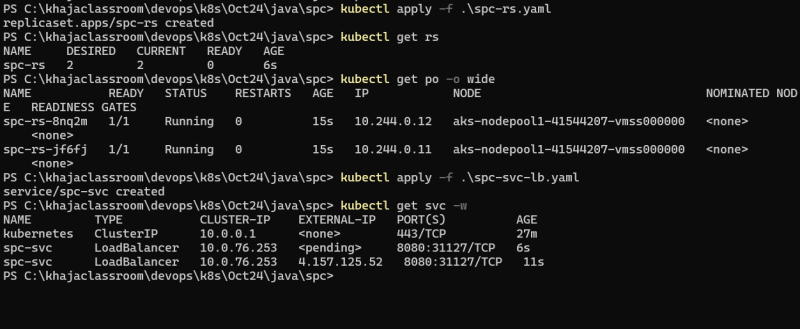
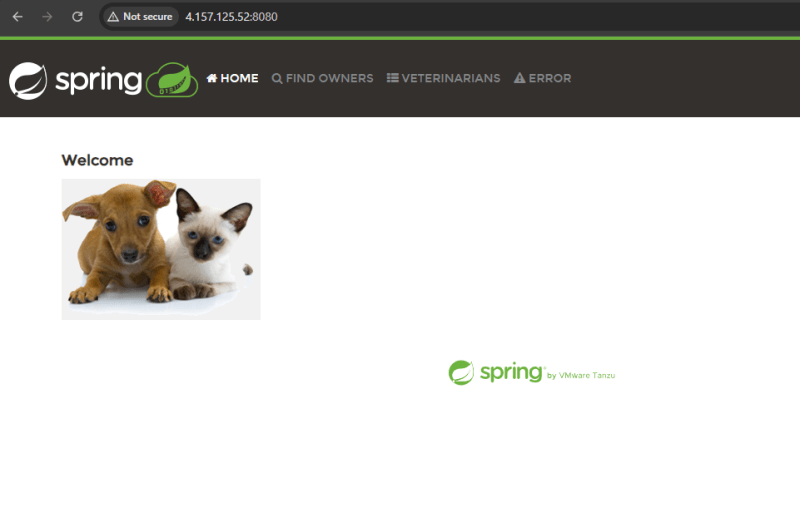
az group create --name $MY\_RESOURCE\_GROUP\_NAME --location $REGION

az aks create --resource-group $MY\_RESOURCE\_GROUP\_NAME --name $MY\_AKS\_CLUSTER\_NAME --node-count 1 --node-vm-size "Standard\_B2ms" --generate-ssh-keys

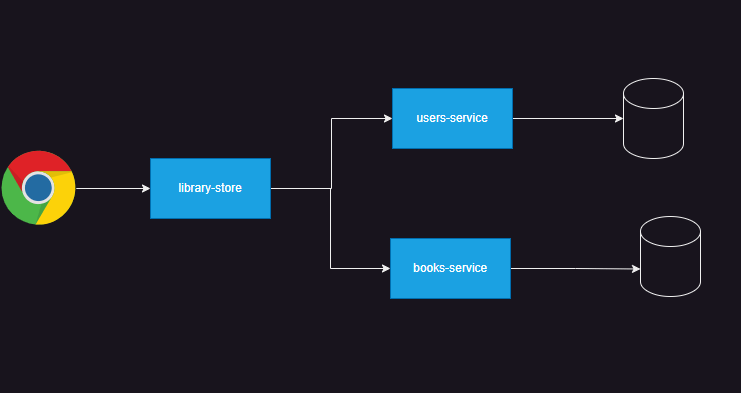
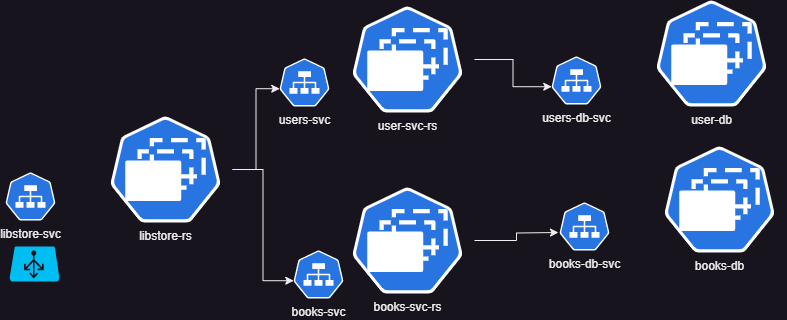
az aks install-cli

az aks get-credentials --resource-group $MY\_RESOURCE\_GROUP\_NAME --name $MY\_AKS\_CLUSTER\_NAME

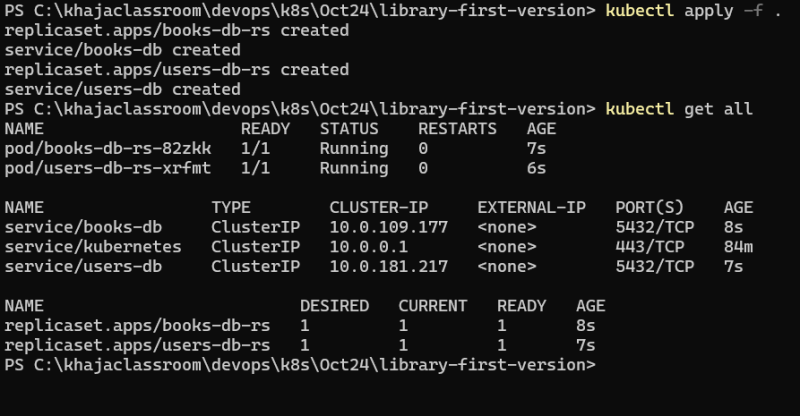
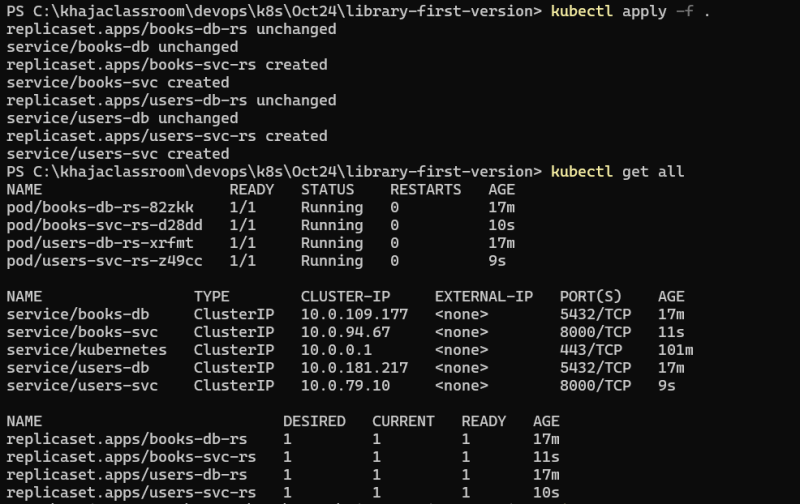
**Lets deploy spc using replicaset and a service of type load balancer**

* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/e588993ad1d1dc435c8441bc70590dcbe938f320) for changes  
    
  

**Library Application**

* Architecture  
  
* Technical stack
  + users-service: REST API
    - python
    - fast api
  + user-db:
    - postgres
  + books-service: REST API
    - python
    - fast api
  + user-db:
    - postgres
  + library-frontend:
    - react js
* To run books database
  + image: postgres:15-alpine
  + environmental variables:
    - POSTGRES\_USER: user
    - POSTGRES\_PASSWORD: password
    - POSTGRES\_DB: booksdb
  + port: 5432
* To run books service
  + image: shaikkhajaibrahim/libbookssvc:1.0
  + environmental variables:
    - DATABASE\_URL: “postgresql://:@:5432/“
    - SECRET\_KEY: ‘YtDEVWnL35aAIP-5yxeLjAZ49R920-mMNDfwPyWULu63HFsYzo0f-LO2InxC8eu428k’
  + port: 8000
* To run user database
  + image: postgres:15-alpine
  + environmental variables:
    - POSTGRES\_USER: user
    - POSTGRES\_PASSWORD: password
    - POSTGRES\_DB: usersdb
  + port: 5432
* To run users servics:
  + image: shaikkhajaibrahim/libuserssvc:1.0
  + environmental variables:
    - DATABASE\_URL: “postgresql://:@:5432/“
    - SECRET\_KEY: ‘YtDEVWnL35aAIP-5yxeLjAZ49R920-mMNDfwPyWULu63HFsYzo0f-LO2InxC8eu428k’
  + port: 8000
* To run library webstore
  + image: shaikkhajaibrahim/libwebstore:1.0
  + environmental variables
    - REACT\_APP\_BACKEND\_API\_URL: http://:8000/api/v1
    - REACT\_APP\_BOOKS\_API\_URL: http://:8000/api/v1/books
    - REACT\_APP\_USERS\_API\_URL: http://:8000/api/v1/users
  + port: 3000  
    

**First Version of Library:**

* Lets create a users database replicaset
* Lets create a books database replicaset  
  
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/7bc97a9f07d2afe451d1d88afd275ecea65be851) for the changes done
* Now lets create rs for users service and books service [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/433c9699375995904ab49639225e7d41f51b3f4d) for changes  
  
* Lets add rs and service for library store [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/3a91c088f48ee5663251d58a1fd632fa0cc6db3e) for changes
* To create a admin user run this from any container

curl -X 'POST' \

'http://users-svc:8000/api/v1/users/' \

-H 'accept: application/json' \

-H 'Content-Type: application/json' \

-d '{

"username": "qtdevops",

"email": "admin@admin.com",

"user\_type": "administrator",

"password": "admin@123"

}'

**Share this:**

October 22, 2024

# DevOps Classroom notes 22/Oct/2024

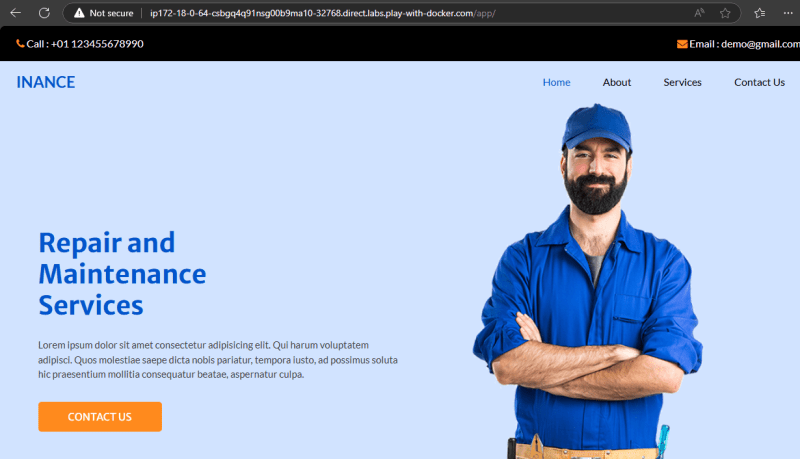
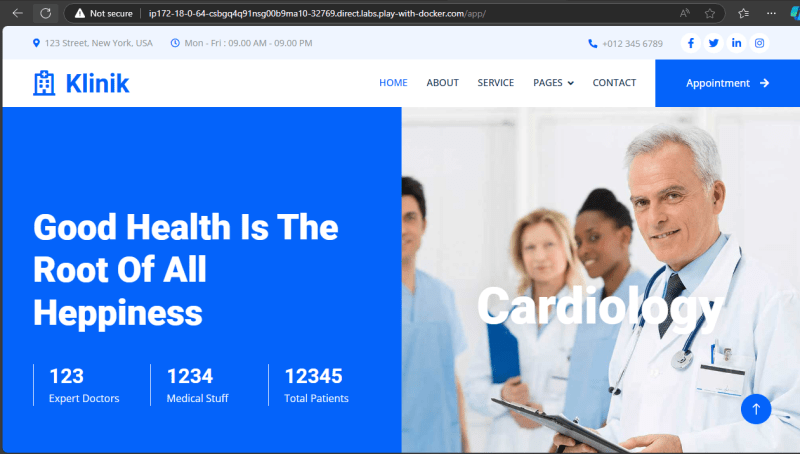
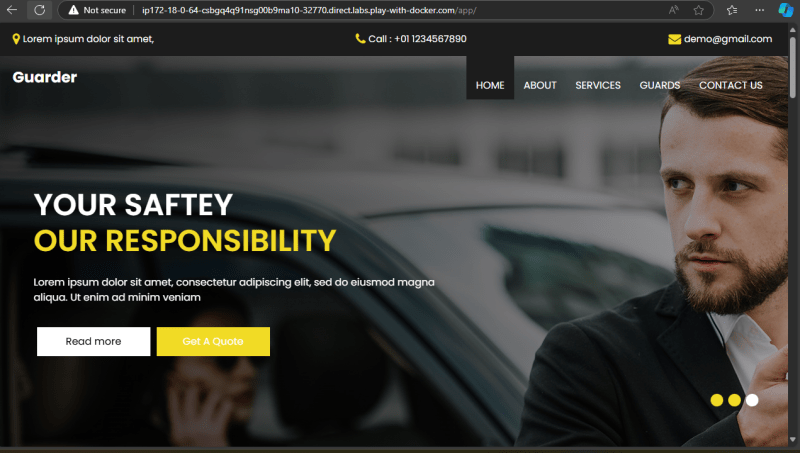
## Phippy Goes to Zoo

* [Refer Here](https://www.cncf.io/wp-content/uploads/2020/08/Phippy-Goes-To-The-Zoo.pdf) for the story book

### [Deployments](https://directdevops.blog/2024/10/22/devops-classroom-notes-22-oct-2024/)

* [Refer Here](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/) for official docs
* Deployments create replica sets, replicasets create pods which in turn runs the containers
* Deployments are suitable for stateless applications
* Deployments come with two strategies
  + Recreate
  + RollingUpdates (Default)
* In Rolling updates
  + We can rollout New versions
  + undo rollout (rollbacks)

### To understand Deployments

* Lets use image shaikkhajaibrahim/dmtest
* Application runs of /app on port 80
* tag 1.0  
  
* tag 2.0  
  
* tag 3.0  
  
* Lets create a [Deployment](https://directdevops.blog/2024/10/22/devops-classroom-notes-22-oct-2024/) with 4 replicas of version 1.0 with a service exposed to external world
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/bf1f3413ded4a7bc0676bf13ac37455c5d2658e2) for the manifests to deploy version 1.0
* For deployment to record the change cause lets use the annotation [Refer Here](https://kubernetes.io/docs/reference/labels-annotations-taints/#change-cause)
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/44b73899ac746dbba0153eb6f59788b49140ff64) for changes done to add change cause
* To find the deployment status we use kubeclt rollout
* To access the application externally use nodeport or loadbalancer
* To deploy the new version change the tag and change cause annotation
* Exercise:
  + Convert
    - users-service from replicaset to deployment
    - books-service from replicaset to deployment
    - library-web-store from replicaset to deployment

**Annotations**

* [Refer Here](https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations/) for official docs
* Annotations add metadata to objects which are generally used by tools

October 23, 2024

DevOps Classroom notes 23/Oct/2024

**Config Maps and Secrets**

* [Refer Here](https://kubernetes.io/docs/concepts/configuration/configmap/) for official docs of config maps
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/8df06a2c3965aaead05c6ba8a24456276394008c) for the changes done to read the values from config maps into pods
* Config maps gives us flexibility to separate configuration from Pod
* The sensitive information is still in plain text, To solve this Secrets for [k8s](https://directdevops.blog/2024/10/23/devops-classroom-notes-23-oct-2024/) gives an base64encoding based approach to store sensitive information
* [Refer Here](https://kubernetes.io/docs/concepts/configuration/secret/) for official docs of secrets
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/9e92cdf9405f7e1676a432c5415ba9fc5e5e1174) for changes done to use secrets
* The production approach for storing sensitive information will be
  + Use an external secrets manager like azure key vault, aws secrets manager, gcp secrets manager or hashicorp vault
  + Use secrets CSI Driver of a vendor to get the sensitive information into k8s as [storage](https://directdevops.blog/2024/10/23/devops-classroom-notes-23-oct-2024/)

**Health Checks or Probes in Kubernetes**

* In K8s we have 3 types of Probes
  + Liveness Probe:
    - Determines if the container is running or not
    - If Probe fails the container is restarted
  + Readiness Probe
    - Determines if the application is running or not
    - If probe fails, this container will not recieve requests from service
  + Startup Probe
    - Determines if the container starup is complete or not
    - If this probe fails no further probes are executed
* [Refer Here](https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/) for official docs
* Configuring probes [Refer Here](https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/#configure-probes)
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/e3d8911298fad3ae03430da77591f8de3980369f) for the changes.

**Share this:**

October 24, 2024

DevOps Classroom notes 24/Oct/2024

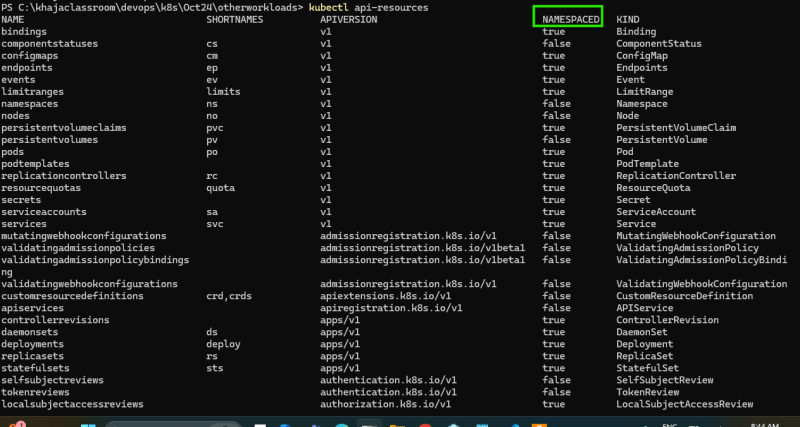
**Daemonsets**

* Daemonsets are used to run a pod on each node or selected nodes
* They are useful for running agent like [softwares](https://directdevops.blog/2024/10/24/devops-classroom-notes-24-oct-2024/) in container
* Daemonset support rolling updates like deployments
* Lets write a simple spec to run a dameonset
* [Refer Here](https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/) for offical docs
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/5a151ac398e2db41533890ebbcb80e042219383c) for the changes done

**Jobs and CronJobs**

* [Refer Here](https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/) for cronjob demo
* Lets write a cronjob which runs every day at 11:30 PM  
  [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/2272726eb85835223ba43f3146d8787ccad0e71b)

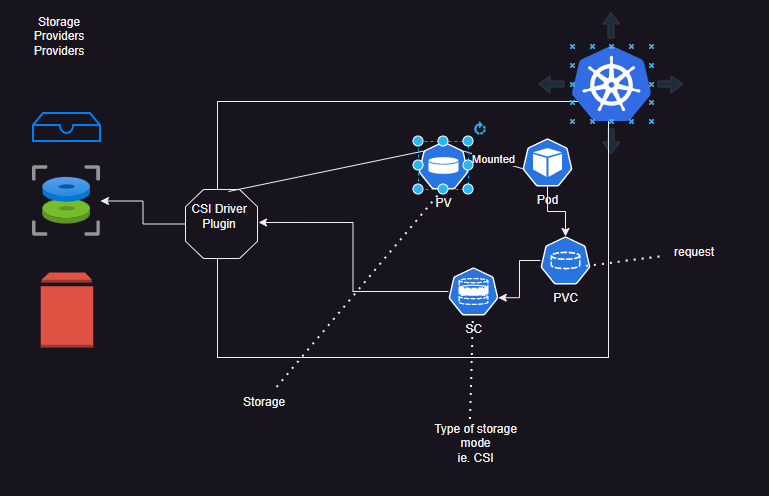
**Namespace**

* [Refer Here](https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/) for official docs
* In kubernetes we have api-resources
* Resources are classified into two scopes
  + Namespace scoped resources:
    - Namespace true indicates namespace scoped resources
  + Cluster scoped resources
    - Namespaced false indicates cluster scoped resources  
      
* Exercise:
  + create a qa namespace (use manifest)
  + create a nginx deployment with 3 replicas in qa namespace
  + create a service with matches the labels in qa namespace
  + create an alpine pod in default namespace, try pinging to the nginx service in qa namespace with name not ip

October 25, 2024

DevOps Classroom notes 25/Oct/2024

**Kubernetes Storage**

* CSI is a [storage](https://directdevops.blog/2024/10/25/devops-classroom-notes-25-oct-2024/) interface for [k8s](https://directdevops.blog/2024/10/25/devops-classroom-notes-25-oct-2024/)
* List of CSI drivers [Refer Here](https://kubernetes-csi.github.io/docs/drivers.html)  
  
* [Refer Here](https://kubernetes.io/docs/concepts/storage/persistent-volumes/) for official docs of PVs
* [Refer Here](https://kubernetes.io/docs/concepts/storage/storage-classes/) for official docs of Storage Classes
* Access Modes
  + RWO – ReadWriteOnce (Suitable for block storages like EBS, Azure Disk, Persistent Disk)
  + ROX – ReadOnlyMany (Any disk)
  + RWX – ReadWriteMany (Typically fileshares or blob storages)
  + RWOP – ReadWriteOncePod (Suitable for block storages like EBS, Azure Disk, Persistent Disk)
* [Refer Here](https://learn.microsoft.com/en-us/azure/aks/azure-csi-disk-storage-provision) for using Azure Disk as PV

**Lets create a mysql Pod where we create a persitent volume dynamically**

* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/298f33dd8375379ee8a54fb8f7fb3b0546fb091a) for the changes done.
* Exercise: Try running mongodb with pvc and postgres with pvc

**Necesity for database cluster**

* To have HA we run database in multiple servers in the case of k8s multiple pods
* IN db cluster we have multiple servers and each server will have its own storage
* So we need to create pods and each pod should have its own persistent volume
* challenge to create cluster with deployments:
  + managing multiple PVs
  + pod names are not predictable: database clusters generally will have predictable endpoints (read endpoint, write endpoint)
* So we need a way to create
  + multiple pods and pvs
  + pod names should be predicatable
* all of the above are acheived with stateful sets

**Share this:**

October 26, 2024

# DevOps Classroom notes 26/Oct/2024

## Lens

* [Refer Here](https://k8slens.dev/) for [kubernetes](https://directdevops.blog/2024/10/26/devops-classroom-notes-26-oct-2024/) lens IDE

# Statefulset

* [Refer Here](https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/) for official docs
* Stateful sets create multiple pods with a predicatable name in a sequential order (0 to n)
* Rolling updates will be performed in a reverse order (n-0)
* Each Pod will raise a PVC to get a PV
* Generally we will have a headless service to access specific pod
* Statefulsets are widely used to create database clusters and any [application](https://directdevops.blog/2024/10/26/devops-classroom-notes-26-oct-2024/) with state.
* Since we acces pod using headless service the libarary application will have a DATABASE URI changed to postgresql://user:password@booksdb-0.booksdb-svc:5432/booksdb
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/366a22e7a5cc969c04fad69514401c48e0c62d04) for the changes done to move away from replicaset to stateful set for database pods in library application.
* For executions watch classroom recording

# Scheduling pods on Nodes

In Kubernetes, scheduling pods on nodes is essential to ensure resources are utilized efficiently and application requirements are met. Here are key techniques and strategies used for scheduling pods:

### 1. ****Node Selector****

* The simplest way to control which nodes a pod is scheduled on.
* Specify nodeSelector in the pod specification to define node labels the pod requires.
* Only nodes with the specified labels will be considered for the pod.

yaml  
spec:  
nodeSelector:  
disktype: ssd

### 2. ****Node Affinity and Anti-Affinity****

* **Node Affinity**: A more flexible and expressive method than nodeSelector. Allows specifying rules for preferred and required node selection.
  + Required (hard constraint): The pod will only be scheduled if nodes match.
  + Preferred (soft constraint): The scheduler tries to match but may skip if unavailable.
* **Anti-Affinity**: Ensures that certain pods are not placed on the same node or close to others. Useful for high availability.

yaml  
affinity:  
nodeAffinity:  
requiredDuringSchedulingIgnoredDuringExecution:  
nodeSelectorTerms:  
- matchExpressions:  
- key: disktype  
operator: In  
values:  
- ssd

### 3. ****Pod Affinity and Anti-Affinity****

* Controls pod placement based on the presence of other pods.
* **Pod Affinity**: Specifies pods that should be placed together for performance or resource-sharing needs.
* **Pod Anti-Affinity**: Ensures pods are spread out across nodes to prevent resource contention.

yaml  
affinity:  
podAffinity:  
requiredDuringSchedulingIgnoredDuringExecution:  
- labelSelector:  
matchLabels:  
app: frontend  
topologyKey: "kubernetes.io/hostname"

### 4. ****Taints and Tolerations****

* **Taints**: Applied to nodes to repel certain pods, setting constraints on which nodes can host which pods.
* **Tolerations**: Allow pods to “tolerate” taints, enabling them to be scheduled on tainted nodes.
* Useful for workloads requiring dedicated nodes or isolation.

yaml  
tolerations:  
- key: "example-key"  
operator: "Equal"  
value: "example-value"  
effect: "NoSchedule"

### 5. ****Resource Requests and Limits****

* Ensure nodes have the required CPU and memory resources for each pod.
* Pods with high resource requests will only be scheduled on nodes with sufficient capacity, preventing overloading.

yaml  
resources:  
requests:  
memory: "64Mi"  
cpu: "250m"  
limits:  
memory: "128Mi"  
cpu: "500m"

### 6. ****Priority and Preemption****

* Allows higher-priority pods to preempt lower-priority ones if the cluster is under resource pressure.
* Priority classes define which pods are most important, ensuring critical applications are scheduled even in tight resource situations.

### 7. ****Topology Spread Constraints****

* Enables even distribution of pods across nodes in the cluster, reducing single points of failure.
* Can control pod spread across various zones or availability domains.

yaml  
topologySpreadConstraints:  
- maxSkew: 1  
topologyKey: "zone"  
whenUnsatisfiable: DoNotSchedule  
labelSelector:  
matchLabels:  
app: frontend

### 8. ****Custom Schedulers****

* Kubernetes allows using custom schedulers for unique workload requirements.
* Useful for applications needing custom placement strategies, such as data locality.

# Administrative Activity: Upgrading K8s clusters

### Self hosted

* Always go through release notes to figure out what has changed in the new version
* backup the etcd cluster and persistent volumes
* cordon the node
* drain the node
* upgrade by executing linux commands
* uncordon and make it available for scheduling

### Managed [k8s](https://directdevops.blog/2024/10/26/devops-classroom-notes-26-oct-2024/) cluster

* Follow the cloud documentations

### Manifests

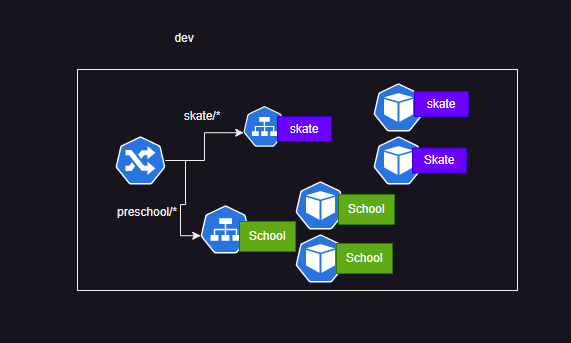
* K8s manifests are static in nature, so every change has to be done in the file and then added to version control
* Reusability becomes a problem
* To solve this problem, we have two options
  + Kustomize
  + Helm: (Package manager for kubernetes):
* NextSteps:
  + ingress and ingress controller
  + Authentication and Authorization
  + Autoscaling:
    - Horizontal Pod Autoscaling
    - Vertical Pod Autoscaling
    - Node Autoscaling (Managed [K8s](https://directdevops.blog/2024/10/26/devops-classroom-notes-26-oct-2024/) Clusters)

### Share this:

October 27, 2024

# DevOps Classroom notes 27/Oct/2024

# Ingress and Ingress Controllers

* [Refer Here](https://kubernetes.io/docs/concepts/services-networking/ingress/) for official docs and [Refer Here](https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/) for ingress [Controllers](https://directdevops.blog/2024/10/27/devops-classroom-notes-27-oct-2024/)
* Lets look at our apps
  + shaikkhajaibrahim/preschool:1.1:
    - This runs a preshool app on path /preschool on port 80
  + shaikkhajaibrahim/skateboard:1.0
    - This runs a skate board app on path /skate on port 80  
      
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/bf0900e67b8cb61ec620901648601b8d3583d98c) for the [deployment](https://directdevops.blog/2024/10/27/devops-classroom-notes-27-oct-2024/) manifest with images and service with clusterip
* K8s doesnot have a default ingress implementation, for this exercise lets use nginx ingress controller

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/cloud/deploy.yaml

* Now lets write ingress rules [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/e341bebfa6e8aa40cc588d08c486ad99e691a150)

# Installing nginx ingress controller

To install the NGINX Ingress Controller using kubectl, follow these steps:

## Steps to Install NGINX Ingress Controller

1. **Create a Namespace**:  
   First, create a namespace for the NGINX Ingress Controller:  
   bash  
   kubectl create namespace ingress-nginx
2. **Apply the NGINX Ingress Controller Manifest**:  
   Use the following command to deploy the NGINX Ingress Controller by applying the official manifest:  
   bash  
   kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/cloud/deploy.yaml
3. **Verify the Deployment**:  
   Check that the NGINX Ingress Controller pods are running:  
   bash  
   kubectl get pods --namespace ingress-nginx
4. **Check the Service**:  
   To see if the NGINX Ingress Controller has been assigned a public IP address, run:  
   bash  
   kubectl get svc --namespace ingress-nginx  
   Look for the service of type LoadBalancer and note its EXTERNAL-IP. It may take a few minutes for the IP to be assigned.
5. **Troubleshooting**:  
   If the EXTERNAL-IP status shows as Pending, you can investigate further with:  
   bash  
   kubectl describe svc -n ingress-nginx ingress-nginx-controller

By following these steps, you will have successfully installed the NGINX Ingress Controller on your Kubernetes cluster. This setup allows you to manage incoming traffic and route it to your services based on defined rules.

Citations:  
[1] <https://help.hcl-software.com/devops/test/hub/10.2.0/docs/topics/t_install_nginx.html>  
[2] <https://spacelift.io/blog/kubernetes-ingress>  
[3] <https://docs.k0sproject.io/v1.25.4+k0s.0/examples/nginx-ingress/>  
[4] <https://gcore.com/docs/cloud/kubernetes/networking/install-and-set-up-the-nginx-ingress-controller>

November 2, 2024

# DevOps Classroom notes 02/Nov/2024

# CRD (Custom Resource Definitions) & Operators

* Kubernetes gives the extension mechanism where we create
  + CRD (Custom resource definition)
  + Operators: [Refer Here](https://sdk.operatorframework.io/)
    - They create reconcile loop
* Lifecycle => CRD + Operator

# Problem 1: Static YAML Manifests

* K8s manifests are static in nature
* During deployments we will have changes to handle
  + image tags
  + labels
  + namespaces
* We have to manually change the manifests

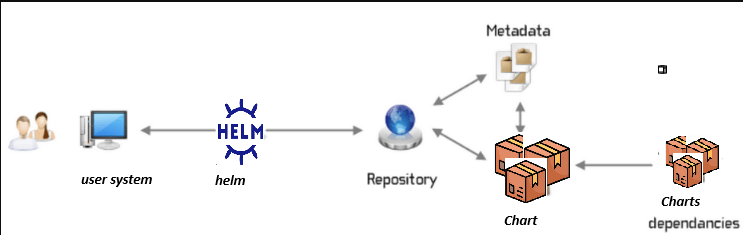
# Problem 2: No Reusability

* Manifest YAML files are not reusable

# Solutions

1. Helm:
2. This works as a package manager to kubernetes
3. we need to install helm
4. Kustomize:
5. This works as if manages multiple environments
6. This works with native kubectl

# Helm

* [Refer Here](https://helm.sh/) for official docs and [Refer Here](https://helm.sh/docs/intro/install/) to install helm
* In helm  
  
* Components:
  + Helm (client)
  + Repository (Which hosts the charts)
  + Chart (An individual package)

## Create a helm chart for basic deployment

* YAML Manifest

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.27

resources:

limits:

memory: "128Mi"

cpu: "500m"

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: nginx-svc

spec:

type: ClusterIP

selector:

app: nginx

ports:

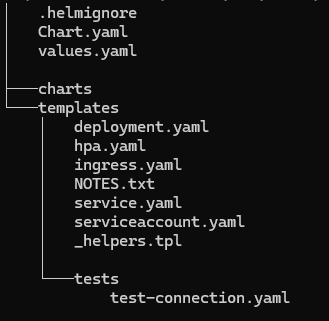
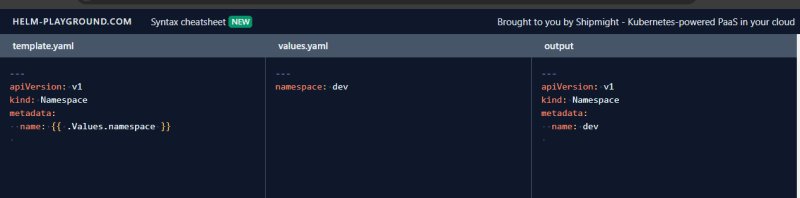
- port: 80

targetPort: 80

### Helm Charts

* Lets create a helm chart

helm create activity1

* A folder called as activity1 is create with following content  
  
* [Refer Here](https://helm.sh/docs/topics/charts/) for official docs on Charts
* Helm use go templating syntax [Refer Here](https://helm.sh/docs/chart_template_guide/)
* Use helm online validator to verify expression [Refer Here](https://helm-playground.com/#t=LQhQFMA8EMFsAcA24BcACA3h4aB0A1aRAV3AGdcBLAF3FjLQB81qB7ATTkSbQDtLeAE3C9qaAExoAvlNBA&v=LQhQEsBcFMFsGcBcACA2gIgGbgE70ugDTLrzQDGA9gHYAm6AuqEA)
* Helm creates manifests and passes it to the kubectl after replacing dynamic expression (template expressions)  
  
* [Refer Here](https://helm-playground.com/cheatsheet.html) for synaxes used in Helm
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/6e46af8af09223c7589d7475f20c3bdc4b10f030) for the helm chart.
* Now create a chart repository and push the chart to repository

# Kustomize

* Kustomize simplifies templating without strange expressions
* [Refer Here](https://kustomize.io/) for official docs
* [Refer Here](https://kubernetes.io/docs/tasks/manage-kubernetes-objects/kustomization/) for kustomize documents
* [Refer Here](https://kubernetes.io/docs/tasks/manage-kubernetes-objects/kustomization/)
* [Refer Here](https://github.com/kubernetes-sigs/kustomize/tree/master/examples/multibases) for multi base

# Elastic Kubernetes Service (EKS)

* This is kubernetes as a service offering from AWS
* AWS Service Integrations: (Private integrations)
  + AWS has VPC CNI where every pod gets an ip address from vpc range
  + AWS services can be linked to AWS load Balancers
  + AWS EKS provides Application Load Balancer Ingress controller
  + AWS EKS provides necessary CSI implementations (StorageClasses) to use
    - AWS EBS Disks
    - AWS EFS
    - AWS FSx
    - AWS S3
* EKS offers cluster node autoscaling
  + Cluster autoscaler
  + Karpenter
* EKS can be integrated with IAM to bring in AWS users or with external identity servers
* What are different EKS offerings ?
  + EKS (Completely on AWS)
    - Compute options
      * EC2
      * Fargate
  + EKS on Outposts (Hybrid cloud)
  + EKS Anywhere (distribution)
    - Onprem
* EKS creation options
  + Console
  + cli
  + eksctl [Refer Here](https://eksctl.io/)
  + terraform (CI/CD)

CI/CD services

### Share this:

November 3, 2024

# DevOps Classroom notes 03/Nov/2024

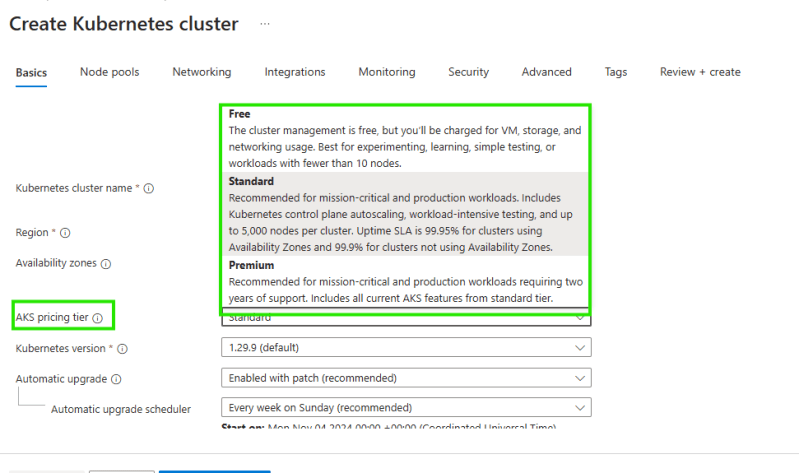
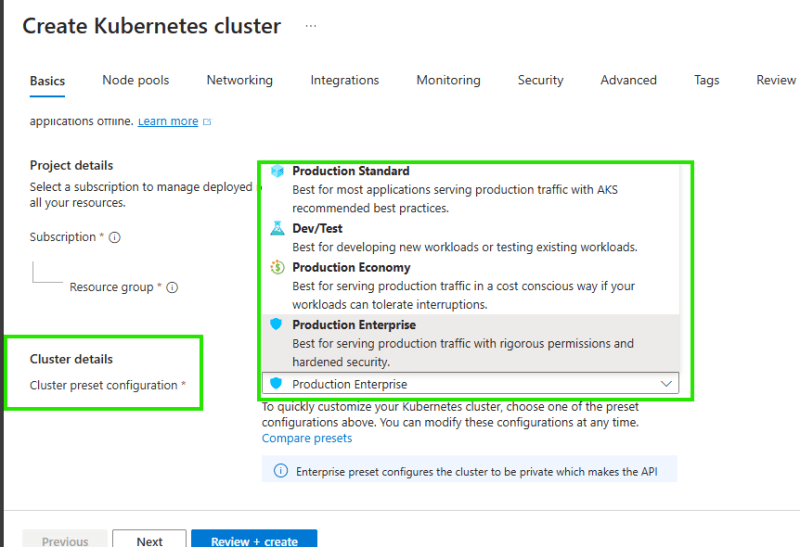
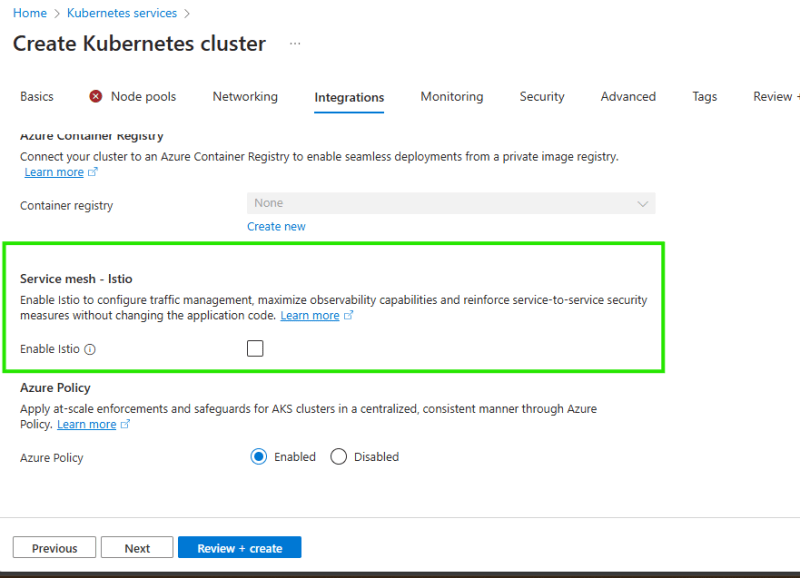
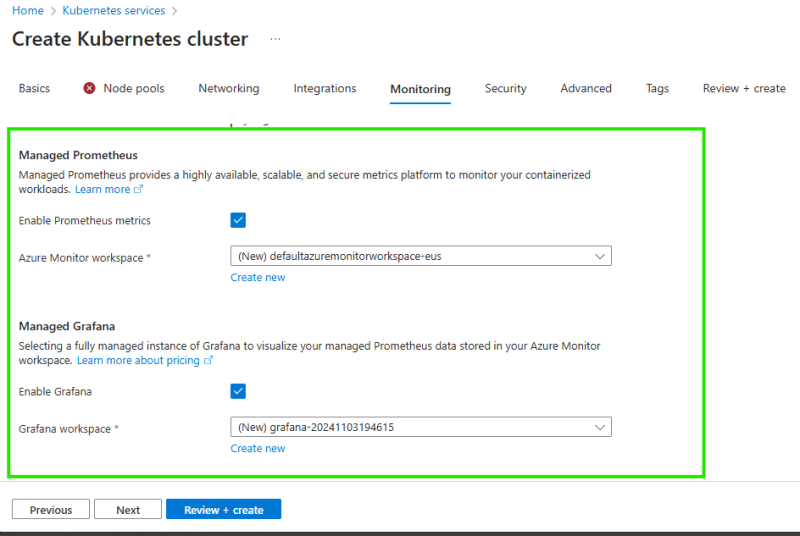
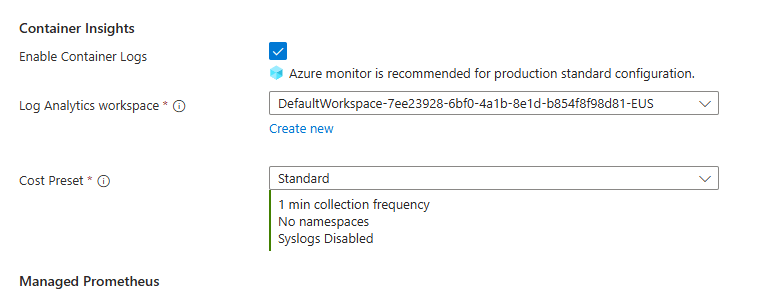
## Elastic Kubernetes Services (EKS)

* [Refer Here](https://eksctl.io/) for eksctl and [Refer Here](https://eksctl.io/installation/) for installation
* [Refer Here](https://eksctl.io/usage/schema/) for schema of eksctl
* Ensure AWS CLI is installed and user is configured
* To store data in persistent volume ensure you have configured proper iam roles [Refer Here](https://docs.aws.amazon.com/eks/latest/userguide/ebs-csi.html)
* [Refer Here](https://github.com/aws/eks-charts) for helm charts
* Find helm charts for setting up
  + storage classes
  + use aws secrets manager mounting secrets
* AWS Service Integrations
  + Network based resources
    - Examples
      * RDS
      * EC2
      * ECS
    - either create the resources and eks in same network or create a peering connection (private)
  + Non network based resources
    - Examples
      * S3
      * Dynamodb
    - Create vpc endpoints
  + IAM Roles: Ensure you create necessary iam roles to k8s cluster
* [Refer Here](https://github.com/asquarezone/KubernetesZone/commit/a7a214213bf706def3805d3045634fa2e37a3fc9) for EKS cluster config
* Cluster Autoscaling: Increasing number of nodes
  + AWS provides two options
    - cluster autoscaler: is autoscaling (like asg)
    - karpenter: it automatically increases number of nodes with fast provisioning [Refer Here](https://karpenter.sh/) and [Refer Here](https://karpenter.sh/docs/getting-started/getting-started-with-karpenter/) for steps
* Backups:
  + [Refer Here](https://velero.io/) for velero
  + [Refer Here](https://aws.amazon.com/blogs/containers/backup-and-restore-your-amazon-eks-cluster-resources-using-velero/) velero with eks
* Upgrades:

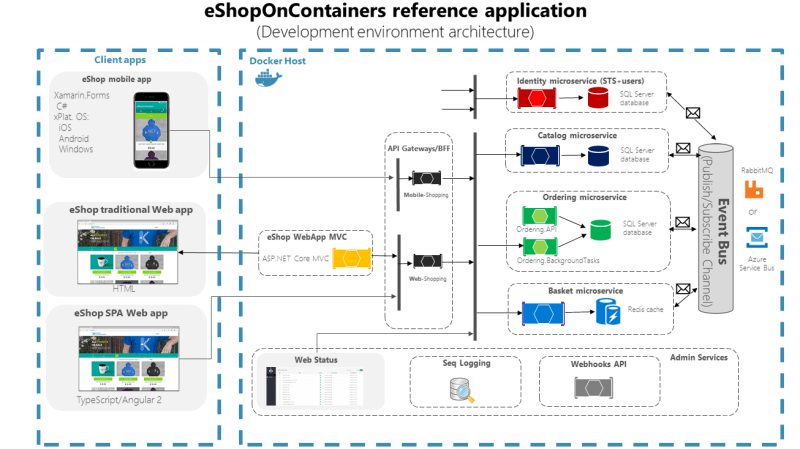
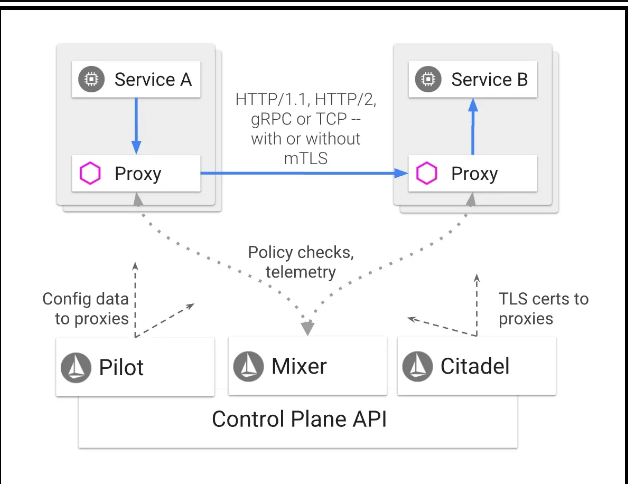
## Storing Secrets in K8s

* K8s secrets are just encoded values, so we need vaults
* Popular vaults
  + Hashicorp vault
  + aws secrets manager
  + azure key vault
* All the secrets can be mounted with the help of
  + CSI Driver
  + SecretProvider

# AKS

* AKS Pricing Tier  
    
    
    
    
  
* Operations:
  + Storage Classes and CSI [Refer Here](https://learn.microsoft.com/en-us/azure/aks/csi-storage-drivers)
  + Azure Key Vault (Secrets) [Refer Here](https://learn.microsoft.com/en-us/azure/aks/csi-secrets-store-driver)
  + Backup:
    - Velero (Opensource tool)
    - [Refer Here](https://learn.microsoft.com/en-us/azure/backup/azure-kubernetes-service-backup-overview) for overview and [Refer Here](https://learn.microsoft.com/en-us/azure/backup/azure-kubernetes-service-cluster-backup) for backup natively
  + Upgrades: [Refer Here](https://learn.microsoft.com/en-us/azure/aks/upgrade-cluster)
* Integrations:
  + [Refer Here](https://learn.microsoft.com/en-us/azure/service-connector/tutorial-python-aks-sql-database-connection-string?tabs=azure-portal&pivots=workload-id) for Azure SQL
  + Azure storage account [Refer Here](https://learn.microsoft.com/en-us/azure/service-connector/tutorial-python-aks-storage-workload-identity?tabs=azure-portal#create-service-connection-with-service-connector-preview)

# Service Mesh

* Consider this architecture  
  
* [Refer Here](https://aws.amazon.com/what-is/service-mesh) for service mesh
* Features
  + mTLS
  + Circuit breaker
  + Traffic Splitting
  + A/B Testing
  + Retry logic
  + Fault injections
  + Network Observability
* Tools
  + Istio: installation ([Refer Here](https://istio.io/latest/docs/setup/install/helm/)) Custom Resources (Virtual Service, Destination Rules, Gateway)  
    
  + linkerd
* Istio on Azure [Refer Here](https://learn.microsoft.com/en-us/azure/aks/istio-about) and on AWS [Refer Here](https://aws.amazon.com/blogs/opensource/getting-started-with-istio-on-amazon-eks/)