**What is Kubernetes ?**

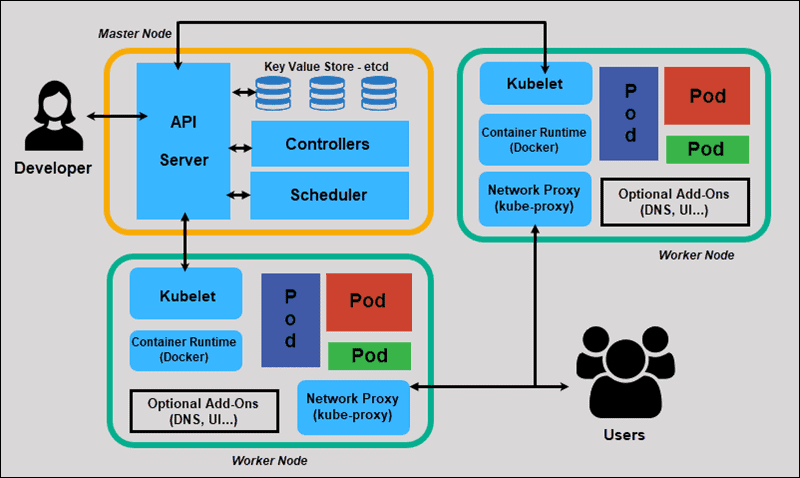
Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

**Purpose**

Containers are a good way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime. For example, if a container goes down, another container needs to start.

**Scope**

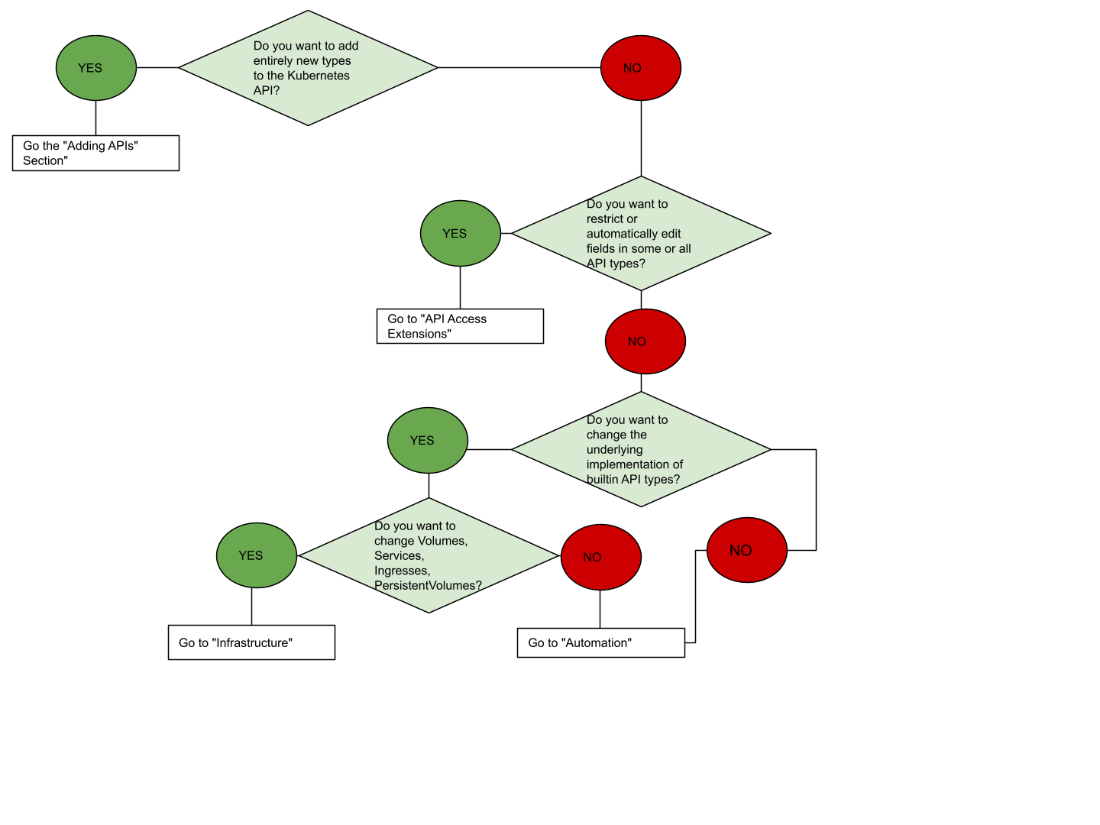
In this document we are going to implement application deployment with Kubernetes.



**Tools and Technologies**

1. Docker
2. Kubernetes
3. Jenkins
4. Git Hub
5. Java
6. Groovy

**Flowchart**



**Prerequisite for Kubernetes**

1. Ubuntu 20.04
2. Master node’s minimal required memory is 2GB and the worker node needs minimum is 1GB
3. Disable SWAP
4. Install Docker & configure

**Implementation steps**

1. **Install Kubernetes Servers**
2. **Install kubelet, kubeadm and kubectl**
3. **Disable Swap**
4. **Install Container runtime**
5. **Initialize master node**
6. Enable kubelet service.
7. These are the basic ‘kubeadm init’ options that are used to bootstrap cluster.

**Installation steps:**

## Assumptions

| **Role** | **FQDN** | **Private IP** | **OS** | **RAM** | **CPU** |
| --- | --- | --- | --- | --- | --- |
| Master | kmaster.example.com | 172.16.16.100 | Ubuntu 20.04 | 2G | 2 |
| Worker | kworker.example.com | 172.16.16.101 | Ubuntu 20.04 | 1G | 1 |

**On both Kmaster and Kworker**

##### Login as root user:

sudo su –

##### Disable Firewall

ufw disable

##### Disable swap

swapoff -a; sed -i '/swap/d' /etc/fstab

##### Update sysctl settings for Kubernetes networking

cat >>/etc/sysctl.d/kubernetes.conf<<EOF

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

EOF

sysctl --system

##### Install docker engine

{

apt install -y apt-transport-https ca-certificates curl gnupg-agent software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -

add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

apt update

apt install -y docker-ce=5:19.03.10~3-0~ubuntu-focal containerd.io

}

### Kubernetes Setup

##### Add Apt repository

{

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add –

echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" > /etc/apt/sources.list.d/kubernetes.list

}

##### Install Kubernetes components

apt update && apt install -y kubeadm=1.18.5-00 kubelet=1.18.5-00 kubectl=1.18.5-00

## On kmaster

##### Initialize Kubernetes Cluster

Update the below command with the ip address of kmaster

kubeadm init --apiserver-advertise-address=172.16.16.100 --pod-network-cidr=192.168.0.0/16 --ignore-preflight-errors=all

Note: private ip should be changed as per your instance

##### Deploy Calico network

kubectl --kubeconfig=/etc/kubernetes/admin.conf create -f https://docs.projectcalico.org/v3.14/manifests/calico.yaml

##### Cluster join command

kubeadm token create --print-join-command

##### To be able to run kubectl commands as non-root user

If you want to be able to run kubectl commands as non-root user, then as a non-root user perform these

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

## On Kworker

##### Join the cluster

Use the output from **kubeadm token create** command in previous step from the master server and run here.

## Verifying the cluster (On kmaster)

##### Get Nodes status

kubectl get nodes

##### Get component status

kubectl get cs

**POC Script :**

1. Deploy.yaml

---  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: nginx-deploy  
spec:  
 selector:  
 matchLabels:  
 app: nginx  
 replicas: 1  
 template:  
 metadata:  
 labels:  
 app: nginx  
 spec:  
 containers:  
 - name: nginx  
 image: devopstestlab/nginx-helloworld  
 ports:  
 - containerPort: 80

1. Svc.yaml

---  
kind: Service  
apiVersion: v1  
metadata:  
 name: nginx-svc  
spec:  
 selector:  
 app: nginx  
 ports:  
 - protocol: TCP  
 targetPort: 80  
 port: 80  
 type: NodePort

## ****deploymentservice.yml****

## 

|  |
| --- |
| apiVersion: apps/v1 |
|  | kind: Deployment |
|  | metadata: |
|  | name: nodeapp-deployment |
|  | labels: |
|  | app: nodeapp |
|  | spec: |
|  | replicas: 1 |
|  | selector: |
|  | matchLabels: |
|  | app: nodeapp |
|  | template: |
|  | metadata: |
|  | labels: |
|  | app: nodeapp |
|  | spec: |
|  | containers: |
|  | - name: nodeserver |
|  | image: thetips4you/nodeapp:latest |
|  | ports: |
|  | - containerPort: 3000   |  | | --- | | --- | |  |  | |  | apiVersion: v1 | |  | kind: Service | |  | metadata: | |  | name: nodeapp-service | |  | spec: | |  | selector: | |  | app: nodeapp | |  | type: LoadBalancer | |  | ports: | |  | - protocol: TCP | |  | port: 5000 | |  | targetPort: 3000 | |  | nodePort: 31110 | |

1. Pipeline:

pipeline {

environment {

dockerimagename = "01234567890123/nodeapp"

dockerImage = ""

}

agent any

stages {

stage('Checkout Source') {

steps {

git 'https://github.com/shazforiot/nodeapp\_test.git'

}

}

stage('Build image') {

steps{

script {

dockerImage = docker.build dockerimagename

}

}

}

stage('Pushing Image') {

environment {

registryCredential = 'dockerhublogin'

}

steps{

script {

docker.withRegistry( 'https://registry.hub.docker.com', registryCredential ) {

dockerImage.push("latest")

}

}

}

}

stage('Deploying App to Kubernetes') {

steps {

script {

kubernetesDeploy(configs: "deploymentservice.yml", kubeconfigId: "kubernetes")

}

}

}

}

}

**Conclusion:**

Kubernetes is a great tool for orchestrating containerised applications. It automates the very complex task of dynamically scaling an application in real time.

**Deploying A Mongo DB Stateful Application on Kubernetes:**

MongoDB is a general-purpose, document-based [NoSQL database program](https://phoenixnap.com/kb/what-is-nosql). As with other non-relational database management systems, MongoDB focuses on scalability and the speed of queries.

[Kubernetes](https://phoenixnap.com/kb/what-is-kubernetes) synergizes with MongoDB to create highly scalable and portable database deployments. These deployments are useful for working with a large amount of data and high loads.

**Requirements:**

* A Kubernetes cluster with [kubectl](https://phoenixnap.com/kb/kubectl-commands-cheat-sheet).
* Administrative access to your system.

==> **First we need to create a mongo.yaml**

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: mongodb-deployment  
 labels:  
 app: mongodb  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: mongodb  
 template:  
 metadata:  
 labels:  
 app: mongodb  
 spec:  
 containers:  
 - name: mongodb  
 image: mongo  
 ports:  
 - containerPort: 27017  
 env:  
 - name: MONGO\_INITDB\_ROOT\_USERNAME  
 valueFrom:  
 secretKeyRef:  
 name: mongodb-secret  
 key: mongo-root-username  
 - name: MONGO\_INITDB\_ROOT\_PASSWORD  
 valueFrom:   
 secretKeyRef:  
 name: mongodb-secret  
 key: mongo-root-password  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: mongodb-service  
spec:  
 selector:  
 app: mongodb  
 ports:  
 - protocol: TCP  
 port: 27017  
 targetPort: 27017

===> By using the below command we can execute this yaml file,

Kubectl apply –f mongo.yaml

===> Next we need to create a **mongo-secret.yaml for storing a Sensitive Information.**

**Vi mongo-secret.yaml**

apiVersion: v1  
kind: Secret  
metadata:  
 name: mongodb-secret  
type: Opaque  
data:  
 mongo-root-username: dXNlcm5hbWU=  
 mongo-root-password: cGFzc3dvcmQ=

**Kubectl apply –f mongo-secret.yaml**

**==> We need to create a mongo-configmap.yaml**

**Vi mongo-configmap.yaml**

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: mongodb-configmap  
data:  
 database\_url: mongodb-service

**==> we need to create a mongo-express.yaml for checking the connectivity between two servers.**

**Vi mongo-express.yaml**

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: mongo-express  
 labels:  
 app: mongo-express  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: mongo-express  
 template:  
 metadata:  
 labels:  
 app: mongo-express  
 spec:  
 containers:  
 - name: mongo-express  
 image: mongo-express  
 ports:  
 - containerPort: 8081  
 env:  
 - name: ME\_CONFIG\_MONGODB\_ADMINUSERNAME  
 valueFrom:  
 secretKeyRef:  
 name: mongodb-secret  
 key: mongo-root-username  
 - name: ME\_CONFIG\_MONGODB\_ADMINPASSWORD  
 valueFrom:   
 secretKeyRef:  
 name: mongodb-secret  
 key: mongo-root-password  
 - name: ME\_CONFIG\_MONGODB\_SERVER  
 valueFrom:   
 configMapKeyRef:  
 name: mongodb-configmap  
 key: database\_url  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: mongo-express-service  
spec:  
 selector:  
 app: mongo-express  
 type: LoadBalancer   
 ports:  
 - protocol: TCP  
 port: 8081  
 targetPort: 8081  
 nodePort: 30000

**Kubectl apply –f mongo-express.yaml**

**== By using public IP and port number we are able to access the application.**

**===> We need to execute this process in Jenkins Pipelines for Automation Purpose. Store all the files inside the Github Repo we Can call it whenever the job will be triggered.**

Pre-requisites:

· ==> EC2 Instance

**AWS EKS Setup**

1. Setup kubectl

a. Download kubectl version 1.20

b. Grant execution permissions to kubectl executable

c. Move kubectl onto /usr/local/bin

d. Test that your kubectl installation was successful

**Commands Used**

**I. curl -o kubectl** [**https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl**](https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl)

**II. chmod +x ./kubectl**

**III. mv ./kubectl /usr/local/bin**

**IV. kubectl version --short –client**

**2. Setup eksctl a. Download and extract the latest release b. Move the extracted binary to /usr/local/bin c. Test that your eksclt installation was successful**

**Commands Used:**

**I. curl --silent --location "**[**https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname**](https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname) **-s)\_amd64.tar.gz" | tar xz -C /tmp**

**II. sudo mv /tmp/eksctl /usr/local/bin**

**III. eksctl version**

**3. Create an IAM Role and attach it to EC2 instance IAM user should have access to IAM EC2 VPC CloudFormation**

**4. Create your cluster and nodes**

**Commands Used:**

**I. eksctl create cluster --name cluster-name \**

**II. --region region-name \**

**III. --node-type instance-type \**

**IV. --nodes-min 2 \**

**V. --nodes-max 2 \**

**VI. --zones <AZ-1>,<AZ-2>**

**For example:**

**eksctl create cluster --name POC-Kubernetes \**

**--region ap-south-1 \**

**--node-type t2.small \**

**5. Validate your cluster using by creating by checking nodes and by creating a pod.**

**Commands Used:**

**1. kubectl get nodes**

**2. kubectl run pod tomcat --image=tomcat**

**Stateless Application deployment:**

1. Deploy.yaml

---  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: nginx-deploy  
spec:  
 selector:  
 matchLabels:  
 app: nginx  
 replicas: 1  
 template:  
 metadata:  
 labels:  
 app: nginx  
 spec:  
 containers:  
 - name: nginx  
 image: devopstestlab/nginx-helloworld  
 ports:  
 - containerPort: 80

1. Svc.yaml

---  
kind: Service  
apiVersion: v1  
metadata:  
 name: nginx-svc  
spec:  
 selector:  
 app: nginx  
 ports:  
 - protocol: TCP  
 targetPort: 80  
 port: 80  
 type: NodePort

## ****deploymentservice.yml****

## 

|  |
| --- |
| apiVersion: apps/v1 |
|  | kind: Deployment |
|  | metadata: |
|  | name: nodeapp-deployment |
|  | labels: |
|  | app: nodeapp |
|  | spec: |
|  | replicas: 1 |
|  | selector: |
|  | matchLabels: |
|  | app: nodeapp |
|  | template: |
|  | metadata: |
|  | labels: |
|  | app: nodeapp |
|  | spec: |
|  | containers: |
|  | - name: nodeserver |
|  | image: thetips4you/nodeapp:latest |
|  | ports: |
|  | - containerPort: 3000   |  | | --- | | --- | |  |  | |  | apiVersion: v1 | |  | kind: Service | |  | metadata: | |  | name: nodeapp-service | |  | spec: | |  | selector: | |  | app: nodeapp | |  | type: LoadBalancer | |  | ports: | |  | - protocol: TCP | |  | port: 5000 | |  | targetPort: 3000 | |  | nodePort: 31110 | |

**Stateful Application Deployment:**

==> **First we need to create a mongo.yaml**

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: mongodb-deployment  
 labels:  
 app: mongodb  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: mongodb  
 template:  
 metadata:  
 labels:  
 app: mongodb  
 spec:  
 containers:  
 - name: mongodb  
 image: mongo  
 ports:  
 - containerPort: 27017  
 env:  
 - name: MONGO\_INITDB\_ROOT\_USERNAME  
 valueFrom:  
 secretKeyRef:  
 name: mongodb-secret  
 key: mongo-root-username  
 - name: MONGO\_INITDB\_ROOT\_PASSWORD  
 valueFrom:   
 secretKeyRef:  
 name: mongodb-secret  
 key: mongo-root-password  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: mongodb-service  
spec:  
 selector:  
 app: mongodb  
 ports:  
 - protocol: TCP  
 port: 27017  
 targetPort: 27017

===> By using the below command we can execute this yaml file,

Kubectl apply –f mongo.yaml

===> Next we need to create a **mongo-secret.yaml for storing a Sensitive Information.**

**Vi mongo-secret.yaml**

apiVersion: v1  
kind: Secret  
metadata:  
 name: mongodb-secret  
type: Opaque  
data:  
 mongo-root-username: dXNlcm5hbWU=  
 mongo-root-password: cGFzc3dvcmQ=

**Kubectl apply –f mongo-secret.yaml**

**==> We need to create a mongo-configmap.yaml**

**Vi mongo-configmap.yaml**

apiVersion: v1  
kind: ConfigMap  
metadata:  
 name: mongodb-configmap  
data:  
 database\_url: mongodb-service

**==> we need to create a mongo-express.yaml for checking the connectivity between two servers.**

**Vi mongo-express.yaml**

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: mongo-express  
 labels:  
 app: mongo-express  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: mongo-express  
 template:  
 metadata:  
 labels:  
 app: mongo-express  
 spec:  
 containers:  
 - name: mongo-express  
 image: mongo-express  
 ports:  
 - containerPort: 8081  
 env:  
 - name: ME\_CONFIG\_MONGODB\_ADMINUSERNAME  
 valueFrom:  
 secretKeyRef:  
 name: mongodb-secret  
 key: mongo-root-username  
 - name: ME\_CONFIG\_MONGODB\_ADMINPASSWORD  
 valueFrom:   
 secretKeyRef:  
 name: mongodb-secret  
 key: mongo-root-password  
 - name: ME\_CONFIG\_MONGODB\_SERVER  
 valueFrom:   
 configMapKeyRef:  
 name: mongodb-configmap  
 key: database\_url  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: mongo-express-service  
spec:  
 selector:  
 app: mongo-express  
 type: LoadBalancer   
 ports:  
 - protocol: TCP  
 port: 8081  
 targetPort: 8081  
 nodePort: 30000

**Kubectl apply –f mongo-express.yaml**

**== By using public IP and port number we are able to access the application.**

**\*\* Setting Up a Grafana and Prometheus for Application Monitoring**:

**Helm:**

**Installing Helm**

**1. $curl** [**https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3**](https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3) **> get\_helm.sh**

**2. $chmod 700 get\_helm.sh**

**3. $./get\_helm.sh**

**Logging and Monitoring:**

**Helm is used to deploy Prometheus and Grafana**

**Exposed the Grafana endpoint in AWS and able to access and configured the Dashboards based on the POD and NODES metrics, using Prometheus Datasource object.**

**1. $helm repo add prometheus-community 2. $helm repo add prometheus-community** [**https://prometheus-community.github.io/helm-charts**](https://prometheus-community.github.io/helm-charts)

**3. $helm upgrade -i prometheus prometheus-community/prometheus --namespace prometheus --set alertmanager.persistentVolume.storageClass="gp2",server.persistentVolume.storageClass="gp2"**

**4. $vi grafana.yaml**

**5. $kubectl create namespace prometheus**

**6. $helm upgrade -i prometheus prometheus-community/prometheus --namespace prometheus --set alertmanager.persistentVolume.storageClass="gp2",server.persistentVolume.storageClass="gp2"**

**7. $kubectl create -f grafana.yaml**

**8. $ kubectl create namespace prometheus**

**9. $helm install prometheus prometheus-community/prometheus --namespace prometheus --set alertmanager.persistentVolume.storageClass="gp2" --set server.persistentVolume.storageClass="gp2"**

**10. $kubectl port-forward -n prometheus deploy/prometheus-server 8080:9090**

**11. $kubectl create namespace grafana**

**12. $helm install grafana grafana/grafana --namespace grafana --set persistence.storageClassName="gp2" --set persistence.enabled=true --set**

**adminPassword='EKS!sAWSome' --values ${HOME}/environment/grafana/grafana.yaml --set service.type=LoadBalancer**

**13. $ls -lrt**

**14. $helm install grafana grafana/grafana --namespace grafana --set persistence.storageClassName="gp2" --set persistence.enabled=true --set adminPassword='EKS!sAWSome' --values root/grafana.yaml --set service.type=LoadBalancer**

**15. $helm install grafana grafana/grafana --namespace grafana --set persistence.storageClassName="gp2" --set persistence.enabled=true --set adminPassword='EKS!sAWSome' --values /root/grafana.yaml --set service.type=LoadBalancer**

**16. $kubectl get svc --namespace grafana -w grafana**

**17. $export SERVICE\_IP=$(kubectl get svc --namespace grafana grafana -o jsonpath='{.status.loadBalancer.ingress[0].ip}'**