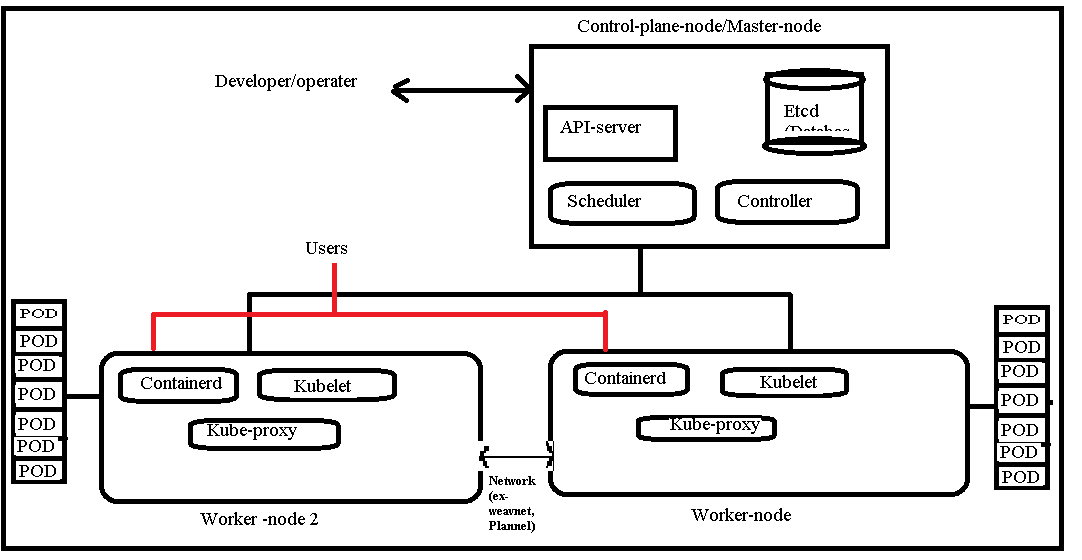
**Kubernetes Notes**

**What is kubernetes?**

Kubernetes is an open-source container orchestration tool for automating deployment, scaling, and management of containerized applications.

Different orchestration tool

* Docker swarm
* Kubernetes(k8s)
* Mesosphere Docs



**There are no of way create the K8s cluster setup**

**1.** **Installing Kubernetes with deployment tools (This also can install in cloud and local)**

* Bootstrapping clusters with kubeadm
* Installing Kubernetes with kOps
* Installing Kubernetes with Kubespray

**2. Installing kubernetes with cloud servers**

* AWS cloud -EKS
* Azure cloud -AKS
* GCP cloud -GKE

**Kubernetes (k8s) Installation and 1node cluster setup with use of utility (kubeadm)** Bootstrapping clusters with kubeadm

**\*\*\*\*\*Step 1- Launch 2 Vm (Instances) \*\*\*\*\*\*\*\*\***

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

(Note- instances required the atlease 2 CPUS & 2 GB RAM)

1. Control-plane-node

2. Worker node

**\*\*\*\*\*Step 2- in the Control plane node instance \*\*\*\*\*\*\*\***

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

***%%% -----Sub step 1----- %%%***

Install Kubeadm, kubelet, kubectl (You will find all the steps in the google ===> download kubeadm for ubuntu ====> Select the kubernetes official

(https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/))

Note: Follow the instruction for the Downloading steps of kubelet, kubeadm & kubectl

**# Task- 1**-Update the apt package index and install packages needed to use the Kubernetes apt repository:

sudo apt-get update

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

**#Task-2**-Download the Google Cloud public signing key:

sudo curl -fsSLo /etc/apt/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg

**#Task-3**-Add the Kubernetes apt repository:

echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

**#Task-4**-Update apt package index, install kubelet, kubeadm and kubectl, and pin their version:

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

***%%%%---Sub step 2-----%%%%%***

Install the Containerd ====> Search in google kubesimplify >>>>blogs>>>search for containerd ( Note- follow all the instruction in the website)

**#Task1**

cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf

overlay

br\_netfilter

EOF

**#Task2**

sudo modprobe overlay

sudo modprobe br\_netfilter

**#Task3--** Setup required sysctl params, these persist across reboots.

cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf

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

net.ipv4.ip\_forward = 1

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

EOF

**#Task4**-- Apply sysctl params without reboot

sudo sysctl --system

**#Task -5**---Install and configure containerd **(Change the sudo apt-key add ) to(sudo tee /etc/apt/trusted.gpg.d/myrepo.asc)**

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/myrepo.asc

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

sudo apt update -y

sudo apt install -y containerd.io

sudo mkdir -p /etc/containerd

containerd config default | sudo tee /etc/containerd/config.toml

**#Task-6**- Start containerd

sudo systemctl restart containerd

sudo systemctl enable containerd

Note----- the kubeadm .kubectl, kubelet & containerd is installed

If you want to check the where all the downloads are available

which kubeadm

which kubelet

which kubectl

which containerd

or

kubeadm version

containerd version

**\*\*\*\*\*STEP 3\*\*\*\*\*\*\*\*(Run the command to init)**

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

kubeadm init (it will create the control node with all installed components)

"if you face any error in the following command and you want to reset all the changes of kubeadm you can use " kubeadm reset" this will remove all the changes happened in the kubeadm init phase ......."

***%%%%--Sub Step -1--%%%%%% (Copy the command from the output)***

V.V.imp ---Note: Copy the output of the kubeadm to set up the further

This is the output of all the phases run by kubeadm init

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

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user (if you are the regular user or like any added user follow the steps)

mkdir -p $HOME/.kube

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

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

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

[**https://kubernetes.io/docs/concepts/cluster-administration/addons/**](https://kubernetes.io/docs/concepts/cluster-administration/addons/)

**(**Go to the above url and select the whichever podnetwork you want 🡺 click on that 🡺click on installation 🡺 copy the command and install the pod network)

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.84.219:6443 --token yogaa6.0mmu83mkrs1zxu3x --discovery-token-ca-cert-hash sha256:14a643aa67df2630ec102f1a24766bd7c36a4ba226d0d5fbaaa806e37fd0d31

**\*\*\*\*\*\*\*\*\*Step-4\*\*\*\*\*\*\*\*\*\*\*\*\* (Execute the command in Worker-node)**

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

***%%% -----Sub step 1----- %%%***

Download **kubeadm, kubectl, kubelet and containerd in the Worker-node**

Note: You can write the shell script and run that file (Note- in file the starting should be #!/bin/bash

Step 1 – create 2 files with .sh file extension (ex- create.sh, containerd.sh)

Step 2- give the file execution permission (chmod +x <file name>)

Step 3- Run the file (./<file name>

**kubectl is the command for every kubernetes need to run this command**

K8s Commands:

🡪(This all commands will run in control-plane-node)

1- To list the nodes

**kubectl get nodes**

2- To list the nodes details ( By this command you can see the all the nodes details like (Name, status, Roles, Age, Version, Internal and external IP,Os image, Kernel-version, Container runtime(ex-containerd)

**kubectl get nodes -o wide**

3-To list all the pods(Control-plane pods) (Including k8s own pods(API server, Controller, Scheduler & etcd ) they are also in the 1-cointainer pod)

**kubectl get pods --all-namespaces**

4- To list only the pods (it will show only the worker nodes pods/created pods)

**kubectl get pods**

Tokens (in kubernetes we will connect the nodes by using the tokens (in starting creating the cluster while doing the kubeadm init command will generate the token )(that will be only for 24 hrs) ) (Note--- Token will valid for 24 hrs only)

5- how to list all the tokens (it will list token and also the joining command )

kubeadm token list

6-how to create only the token

kubeadm token create

7-how to create token along with joining command

kubeadm token create --print-join-command

**------------------------------------------PODS------------------------------------------**

Pods is a group of one or more container with a shared execution environment in Kubernetes (shared Storage, Network, Resources & specification for running the container)

In the Kubernetes can't create container directly you need to create the pod and pod had the IP address (Container don’t have IP in k8s), Pods Consists one or more container

Two pod can talk each other with the external network interface (eth0) and two applications or containers in the singe pod will talk each other with localhost (lo)

**You can create a pod with 3 types/methods**

* **Pod object/method**
* **Deployment object/method**
* **service object/method**

--To see the network interface

sudo apt-get install net-tools (it will download the ifconfig file)

--To view the network interfaces and local host

ifconfig

**Project 1- via Direct giving Pod object/method**

Create pod and deploy the Nginx image in the pod (Single container pod in 1node cluster)

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

Total project 1(breffing)

Step 1- clone the code from GitHub

Step 2 - build the code using maven

Step 3- create image using the Dockerfile (contains -ubuntu, nginx server, deploy war file to www.html and CMD start nginx) and push to docker hub

---Till now old process

Step 4 - Create the Manifest file (ex- pod.yml/pod.yaml) (take images only nginx-server)

Step 5 - Create pod (with single container pod) with nginx server

Note- Already Build the 1 node cluster (control-plane-node and worker node)

===================================================

Write manifest file (pod.yml) (Note- yaml file is case sensitive, indentation sensitive)

Note-1:- If you have 2 words you need to give 1st letter of 1st word with small case and 2nd word with first letter with caps

----Yam syntax is case sensitive (ex- bharathBata), ex2 - mukeshKumar

Note-2:- Yaml syntax is indentation sensitive (The suggested syntax for YAML files is to use 2 spaces for indentation)

Ex- abcd: xyz (Semicolon should be after the abcd only and space then next text)

Ex- The suggested syntax for YAML files is to use 2 spaces for indentation

-metadata:

name:

-spec:

name:

**How to write the Manifest file (Yaml file) with Pod object/method**

Step 1- Create the file with .yml/yaml extension (Ex: abcd.yaml/abcd.yml)

Step 2- GO the file and the write the script inside the file (yaml syntax is Case and Indentation sensitive) .

abcd.yml

apiVersion: v1

kind: Pod

metadata:

name: nginx-pod

lables:

env: prod

version: 1.2.0

branch: xyz

client: flipkart

spec:

containers:

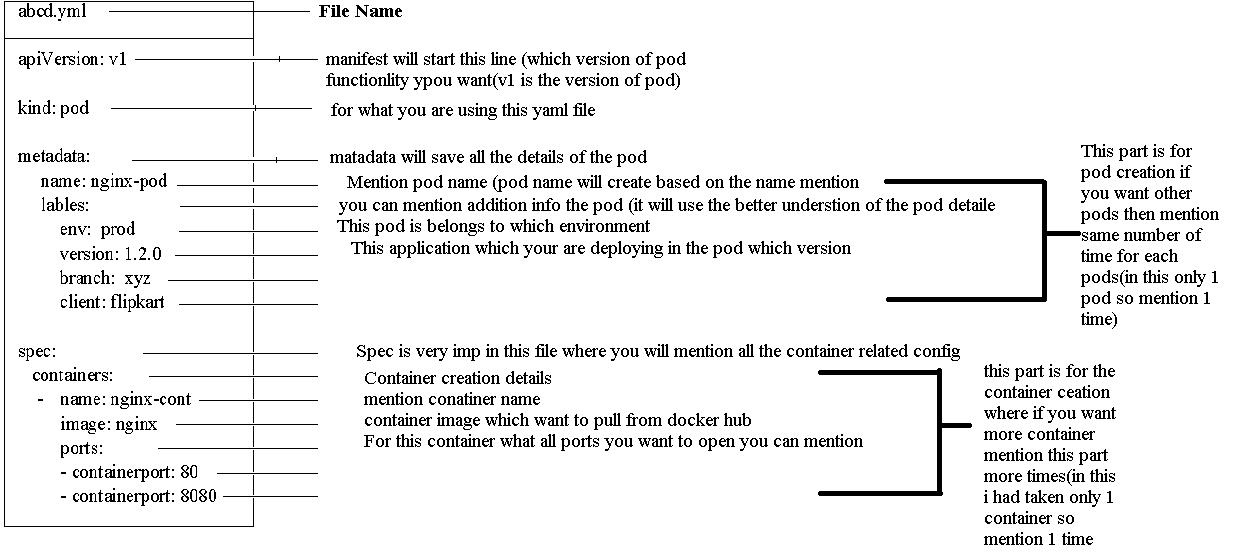
* name: nginx-cont

image: nginx

ports:

- containerport: 80

- containerport: 8080



Note: apiVersion (“V” should be in caps only) and Pod (“P” should be in caps in kind)

**Manifest files description:**

**-**Straight away we can see four top level resources:

* apiVersion
* kind
* metadata
* spec

**apiVersion:**

* Tells the API server about what version of the YAML is used to create the object(POD object in this case)
* Pod is currently in the version v1 API group.

**Kind:**

* Tells us the kind of object being deployed. In this case, we are creating a POD object.
* It tells the control plane what type of object is being defined.

**Metadata:**

* This section has again two sub-sections i.e. name and the labels
* You can name the pod using the “name” key
* Using the labels you can identify the pod.

Note: If you want to create more pods you can specify one more time.

**Spec:**

* This is where we can specify details about the containers that will run in the

Pod.

* In this section you can specify container details (like Name, Image, Port no, etc)

Note: You can create one or more containers by repeating the steps (like Name, Image, Port no, etc)

**K8s Commads**

🡪How to execute the .yaml file

kubectl apply –f <filename>

🡪How to check the pod full details

kubectl get pods –o wide

🡪How to delete the pod

kubectl get pod <pod name>

(or)

You can delete through the Yaml file also

kubectl delete -f <yaml File name with you want to delete>

🡪How to check the pod is running (hit google /but it won’t work you can check here)

curl http://< ip of pod>:<port-no>

🡪How to check the pod creation phases (Interview questions) How do you troubleshoot when Pod creation fails

kubectl describe pod <pod name>

go and check the events (All the information about the Pod)

🡪How to view the pod creation status in real-time.

kubectl get pods –watch

(it will continuously watch and gives the status output)

**NOTE: In real time we won’t create pods with the pod object way this is just learning the basic**

Because - If a Pod fails, it is not automatically get the replicas. Because of this, we usually deploy them via higher-level objects such as Deployments.

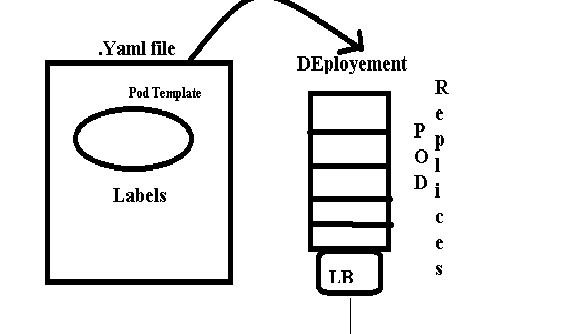
- Deployment adds things like "scalability" (scale-up/down), "self-healing", "rolling updates" and "rollbacks" and makes Kubernetes so powerful.

**Project-2-Creating Deployments & Services object/method**

Advantages: (Deployment)

* Using the Deployment method you can create the replaces of the pod (means you can create “n” number of pod )
* Using the Deployment method you can enable the Controller to do self-healing process
* Using the Deployment you will get the advantage of the doing Scalling up and down, Rolling back etc

Deployment uses the pod templates to create the replaces and it will allow the Pod version is apps/v1



**In Kubernetes, what is the difference between a service and a deployment?**

In Kubernetes a deployment is a method of launching a pod with containerized applications and ensuring that the necessary number of replicas is always running on the cluster.

On the other hand, a service is responsible for exposing an interface to those pods, which enables network access from either within the cluster or between external processes and the service.

**How to write the Manifest file with using the Deployment (example)**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: nginx-pod**

**version: 1.2.0**

**spec:**

**replicas: 10**

**selector:**

**matchLabels:**

**app: nginx-prod-pod**

**template:**

**metadata:**

**labels:**

**app: nginx-prod-pod**

**spec:**

**containers:**

**- name: nginx-cont**

**image: nginx**

**ports:**

**ContainerPort: 80**

**Manifest files description:**

**-**Straight away we can see four top level resources:

* apiVersion
* kind
* metadata
* spec

**apiVersion:**

* Tells the API server about what version of the YAML is used to create the object(POD object in this case)
* Pod is currently in the version v1 API group.

**Kind:**

* Tells us the kind of object being deployed. In this case, we are creating a POD object.
* It tells the control plane what type of object is being defined.

**Metadata:**

* This section has again two sub-sections i.e. name and the labels
* You can name the pod using the “name” key
* Using the labels you can identify the pod.

Note: If you want to create more pods you can specify one more time.

**Spec1:**

* This is where we can specify details about the containers that will run in the

Pod.

* In this section you can specify container details (like Name, Image, Port no, etc)

Note: You can create one or more containers by repeating the steps (like Name, Image, Port no, etc)

**replicas:**

**-** This is where you will mention how many pod you want to create

**selector**:

- This is where you will mention which labels you want to select to create the replicas

**matchLable:**

- This is where you need to mention the labels info.

**Commands:**

* **To run the deployment file**

**kubectl apply –f deployment.yaml**

* **To view the deployment**

**kubectl get deployment**

* **To view the deployment widely**

**kubectl get deployment –o wide**

* **To view the deployment compete details**

**kubectl describe deployment < deployment name>**

* **To delete the deployment**

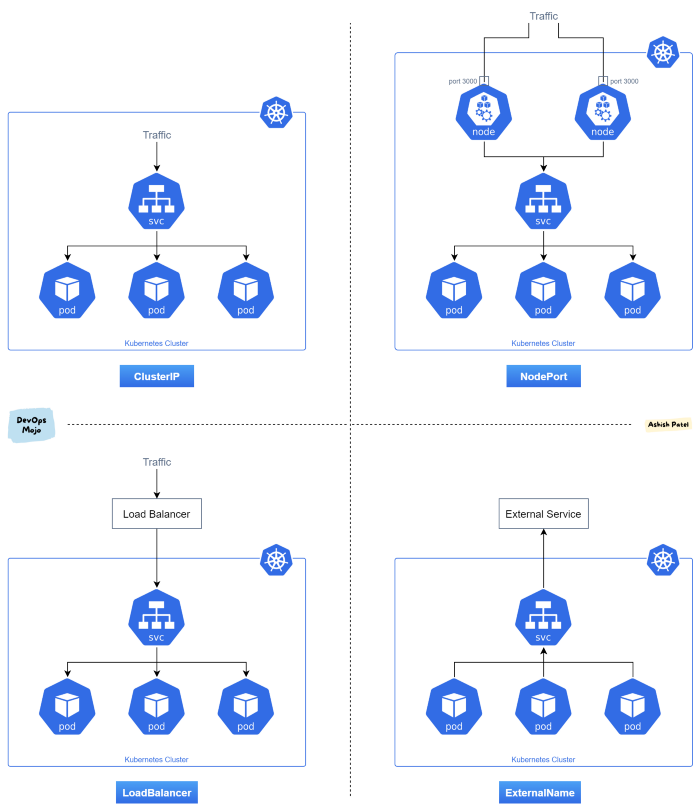
**kubectl delete deployment < deployment name>**

**kubectl delete –f < deployment.yaml file>**

**Project -3- Services method**

There are 4 Types of Services in Kubernetes

* Cluster Ip
* Nodeport
* Load balancer
* External name
* Head less



1. ClusterIP ----- ClusterIP is the default and most common service type.

Kubernetes will assign a cluster-internal IP address to ClusterIP service. This makes the service only reachable within the cluster.

You cannot make requests to service (pods) from outside the cluster. You can optionally set cluster IP in the service definition file.

\*Use Cases\*

Inter service communication within the cluster. For example, communication between the front-end and back-end components of your app.

2. NodePort:

NodePort service is an extension of ClusterIP service. A ClusterIP Service, to which the NodePort Service routes, is automatically created.

It exposes the service outside of the cluster by adding a cluster-wide port on top of ClusterIP.

NodePort exposes the service on each Node’s IP at a static port (the NodePort). Each node proxies that port into your Service. So, external traffic has access to fixed port on each Node. It means any request to your cluster on that port gets forwarded to the service.

You can contact the NodePort Service, from outside the cluster, by requesting <NodeIP>:<NodePort>.

Node port must be in the range of 30000–32767. Manually allocating a port to the service is optional. If it is undefined, Kubernetes will automatically assign one.

If you are going to choose node port explicitly, ensure that the port was not already used by another service.

\*Use Cases\*

When you want to enable external connectivity to your service.

Using a NodePort gives you the freedom to set up your own load balancing solution, to configure environments that are not fully supported by Kubernetes, or even to expose one or more nodes’ IPs directly.

Prefer to place a load balancer above your nodes to avoid node failure.

3. LoadBalancer:

LoadBalancer service is an extension of NodePort service. NodePort and ClusterIP Services, to which the external load balancer routes, are automatically created.

It integrates NodePort with cloud-based load balancers. It exposes the Service externally using a cloud provider’s load balancer.

Each cloud provider (AWS, Azure, GCP, etc) has its own native load balancer implementation. The cloud provider will create a load balancer, which then automatically routes requests to your Kubernetes Service.

Traffic from the external load balancer is directed at the backend Pods. The cloud provider decides how it is load balanced.

Every time you want to expose a service to the outside world, you have to create a new LoadBalancer and get an IP address.

\*Use Cases\*

When you are using a cloud provider to host your Kubernetes cluster.

4. ExternalName:

Services of type ExternalName map a Service to a DNS name, not to a typical selector such as my-service.

You specify these Services with the `spec.externalName` parameter.

It maps the Service to the contents of the externalName field (e.g. foo.bar.example.com), by returning a CNAME record with its value.

No proxying of any kind is established.

Use Cases

This is commonly used to create a service within Kubernetes to represent an external datastore like a database that runs externally to Kubernetes.

You can use that ExternalName service (as a local service) when Pods from one namespace to talk to a service in another namespace.

**How to write the manifest file for the service (Service type is NodePort)**

**apiVersion: v1**

**kind: service**

**metadata:**

**name: nginx-lb**

**labels:**

**app: nginx-pod-np**

**version: 1.8.3**

**spec:**

**selector:**

**app: nginx-prod-pod**

**type: NodePort**

**ports:**

**- nodePort: 31000**

**port: 80**

**targetPort: 80**

**Commands:**

* **To run the service file**

**kubectl apply –f service.yaml**

* **To view the services**

**kubectl get service**

* **To view the service widely**

**kubectl get service –o wide**

* **To view the service compete details**

**kubectl describe service <service name>**

* **To delete the service**

**kubectl delete service <service name>**

**kubectl delete –f <service.yaml file>**

* To view nodes, pods, deployment, service, StatefulSet

kubectl get nodes,pods,deploymets,services,Statefulset

Now the service is also created you can hit from the outside world (All the pods are connected to the Nodeport service where you can access from the google)

🡪 In the above manifest file I had given NodePort port number is 31000 and it is assigned to the worker public ip

WorkerpublicIP:31000 (you can search from google)

Ex—<http://34.125.163.68:31000>

**If you are not getting anything from browser ( Error) Check the firewall of the server**

**GCP console 🡪 All service 🡪 VPC network 🡪Firewall 🡪 Click on the security group 🡪 EDIT 🡪 go to the 🡪 source IPV4 range (enter the cider 0.0.0.0/0) 🡪 Enter TCP port range (31000)🡪Save**

**Till now what I did that are the on-premises Clusters (But that are Not recommended)**

**Reason-**

**🡪 In the on-premises cluster, you need to manage the cluster – it will be difficult to manage where in the upgrade of the version when this cluster is in the production and testing environment.**

**🡪 Adding worker node to the cluster need to take care (need to add manually).**

**GKE (Google Kubernetes Engine)**

**–This services provided by google where Managing of the cluster will taken by the cloud provider (means adding the worker node based on the traffic and get the upgradation of the version will taken by them)**

**How to create the cluster by using GKE**

**Step 1- GO the GCP console and go for the Kubernetes engine service 🡪click on Cluster option 🡪click on Create option**

**In creation, there are two types to create cluster**

1. **Autopilot (Google manages your cluster)**

**It work on pay-per-pod kubernetes cluster where GKE manages your nodes with minimal configuration required.**

1. **Standard (You manages your cluster)**

**It works on pay-per-node kubernetes cluster where you configure and manages the nodes.**

**Step 2- (In this case I had selected the standard ) 🡪 It will ask the Cluster details and fill the details 🡪 click on create 🡪 It will take 5 min to create the cluster.**

**(you can check what is happening by clicking the name of the cluster)**

**Step 3- Once cluster created 🡪 click on 3 dot button 🡪 click on connect 🡪 Ypiu will get the popup page where you will get the Command line access key (Copy that) 🡪click on Run in cloud shell option 🡪 you will command line terminal (that you can use extend extend to get in browser)**

**Step 4- In the CLI terminal you will get the pop up of the Access key authentication they you past the access key which you copied to get access for command line terminal**

**Step 5- Check the nodes by using the view commands (kubectl get nodes)**

**Step 6- Create the Deployment and service manifest files**

**🡪 deployment.yaml**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: gamutkart-dep**

**labels:**

**app: gamutkart-prod**

**spec:**

**replicas: 10**

**selector:**

**matchLabels:**

**app: gamutkart-prod**

**template:**

**metadata:**

**labels:**

**app: gamutkart-prod**

**spec:**

**containers:**

**- name: gamutkart-cont**

**image: bharathBata/gamutkart-ima**

**ports:**

**- containerPort: 8080**

**command: [“/bin/sh”]**

**args: [“-c”, “/root/apache-tomcat-8.5.38/bin/startup.sh; while true; do sleep 1; done;”]**

**🡪 service.yaml**

**apiVersion: v1**

**kind: service**

**metadata:**

**name: gamutkart-ser**

**labels:**

**app: gamutkart-prod**

**spec:**

**selector:**

**app: gamutkart-prod**

**type: LoadBalancer**

**ports:**

**targetPort: 8080**

**port: 8080**

**🡪execute both the files**

**kubectl apply –f deploymeny.yaml**

**kubectl apply -f service.yaml**

**After creating the pod and LoadBalancer then hit in google to check the enable to the outside world**

**LoadBalancer external ip address(service):<port number>/warfile name**

**ex-** [**http://34.136.250.235:8080/gamutgurus/**](http://34.136.250.235:8080/gamutgurus/)

**🡪 How to check the external ip (service)**

**kubectl get service**

**Create the StatefulSet object for creating pods**

* **Step-1 - for achieving this you need to get the service first**
* **step -2 – you need to edit the deployment files “in the kind you need mention”**

**kind: StatefulSet**

**in the spec you need to mention**

**spec:**

**replicas:**

**serviceName: <mention which service you want to tag>**

**ex- serviceName: gamutkart-ser**

**apiVersion: apps/v1**

**kind: StatefulSet**

**metadata:**

**name: gamutkart-dep**

**labels:**

**app: gamutkart-prod**

**spec:**

**replicas: 10**

**serviceName: gamutkart-ser**

**selector:**

**matchLabels:**

**app: gamutkart-prod**

**template:**

**metadata:**

**labels:**

**app: gamutkart-prod**

**spec:**

**containers:**

**- name: gamutkart-cont**

**image: harathBata/gamutkart-ima**

**ports:**

**- containerPort: 8080**

**command: [“/bin/sh”]**

**args:  [“-c”, “/root/apache-tomcat-8.5.38/bin/startup.sh; while true; do sleep 1; done;”]**

**\*Interview Question\* Difference between Deployment and StatefulSet**

|  |  |
| --- | --- |
| **Deployment** | **StatefulSet** |
| **The deployment object creates the pod with random names** | **The StatefulSet object creates the pod with Predicted/fixed name** |
| **If the pod got deleted in k8s has self-healing where it will create a new pod with new name** | **If the pod got deleted in k8s has self-healing where it will create a new pod with deleted pod name only**  **Ex- if abcd pod deleted then it will create a new pod with the abcd name only** |
| **If any volume is attached to the pod and that pod got deleted the deployment object will create only a new pod it won’t attach the volume to the pod automatically** | **If any volume is attached to the pod and that pod got deleted the in the StatefulSet object will create the new pod and attach the volume automatically.** |
| **Deployment object create the pod simultaneously (without any order/without following any sequences)**  **(it will start creating all the pods at a time)** | **StatefulSet will create the pods Sequences/ in order (it will create first 1st pod and then 2nd pod)** |
| **It is a readwritemany and readonlymany (it will share the volume with other pod so any pod can read write the volume at any times it will leads to duplicate)** | **It is readwriteonce (Means it will read and write the volume only once** |
| **They don’t have deducted volume and will share the volume each other** | **They had deducted volume and wont share each other** |

**Commands:**

* **To run the StatefulSet file**

**kubectl apply –f StatefulSet.yaml**

* **To view the StatefulSet**

**kubectl get StatefulSet**

* **To view the StatefulSet widely**

**kubectl get StatefulSet –o wide**

* **To view the StatefulSet compete details**

**kubectl describe s StatefulSet <service name>**

* **To delete the StatefulSet**

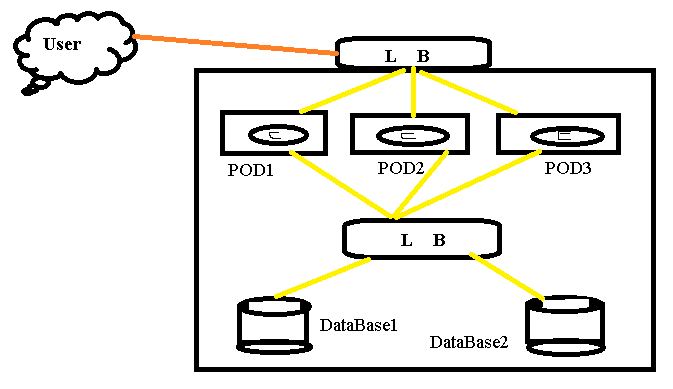
**kubectl delete StatefulSet < StatefulSet name>**

**kubectl delete –f < StatefulSet.yaml file>**

**====Till now we had created the 1 tire application cluster===**

**2 tire application Cluster**

**Structure of the application**

****

**This will be the critical project (Where I am taking the sample application from the Kubernetes to practices)**

**(**[**https://docs.vmware.com/en/VMware-Cloud-Foundation/services/vcf-developer-ready-infrastructure-v1/GUID-6F184EC5-AFC1-4D0A-A5D5-1E31EE938438.html**](https://docs.vmware.com/en/VMware-Cloud-Foundation/services/vcf-developer-ready-infrastructure-v1/GUID-6F184EC5-AFC1-4D0A-A5D5-1E31EE938438.html)**)**

**deploy a Kubernetes guestbook application using MongoDB 🡪 VMware**

**So need to write the manifest file**

1. **front-end Deployment yaml file (to Create the pods)**
2. **front-end service yaml file (to Create load balancer)**
3. **Back-end Mango DB yaml file (To Create the data base)**
4. **Back-end DB service yaml file (to create the load balancer to access the DB)**
5. **Create the Deployment file for the frontend pods**

**(This all information will be given by the Developer)**

**Image name: paulczar/gb-frontend:v5**

**No of pods: 3**

**No of the container in Pod: 1**

**Memory allocation: 100mb**

**CPU allocation: 100m**

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

**frontend-deployment.yaml**

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

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: guestbook-dep**

**labels:**

**name: guestbook**

**component: frontend**

**spec:**

**replicas: 3**

**selector:**

**matchLabels:**

**gb-pod: guestbook-pod**

**template:**

**metadata:**

**labels:**

**gb-pod: guestbook-pod**

**spec:**

**containers:**

**- name: guestbook-cont**

**image: paulczar/gb-frontend:v5**

**resources:**

**requests:**

**memory: 100Mi**

**cpu: 100m**

**env:**

**- name: GET\_HOSTS\_FROM**

**value: dns**

**ports:**

**- containerPort: 80**

**Explication:**

- Pod name given is in starting metadata**: guestbook-dep**

- In the pod related labels give one more

name : guestbook

component: frontend

-Attached the labels for the container and pod is **gb-pod: gustbook-pod (where the selector can select this tagged pod for futher use)**

**-attached deducted memory and cpu for the cointainer**

**- Given environment variable for the image (where this image required the env variable)**

**env:**

**name: GET\_HOME\_FROM**

**value: dns**

**given port number as 80(where this front-end code is use html code)**

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

1. **Create the frontend-service for the load balancer**

**Service type is =LoadBalancer**

**Pod tag is gb-pod: guestbook-pod (for all this tagged pods need to assig for the this load balancer )**

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

**frontend-service.yaml**

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

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: guestbook-frontend-service**

**labels:**

**name: guestbook**

**spec:**

**selector:**

**gb-pod: guestbook-pod**

**type: LoadBalancer**

**ports:**

**- port: 80**

1. **Backend –deployment (This is to create the database pods)**

**(This info is given by dev team)**

**- 2 pods for the database**

**- 1 container for the pod**

**- image: mongo:4.2**

**- allocate deducted memory and cpu: 100mb and 100m**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: guestbook-mango-dep**

**labels:**

**name: Database-mango**

**component: backend**

**spec:**

**replicas: 2**

**selector:**

**matchLabels:**

**mg-pod: guestbook-db**

**template:**

**metadata:**

**labels:**

**mg-pod: guestbook-db**

**spec:**

**containers:**

**- name: mango-cont**

**image: mongo:4.2**

**resources:**

**requests:**

**memory: 100Mi**

**cpu: 100m**

**args:**

**- --bind\_ip**

**- 0.0.0.0**

**ports:**

**- containerPort: 27017**

**Explication:**

- Pod name given is in starting metadata**: guestbook-mango-dep**

- In the pod related labels give one more

name : guestbook-mango

component: backend

-Attached the labels for the container and pod is **mg-pod: guestbook-db (where the selector can select this tagged pod for further use)**

**-attached deducted memory and cups for the container**

**- after creating the container it will pass the commands “args”**

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

**args:**

**- --bind\_ip**

**- 0.0.0.0**

**=====This means after creating it will run this command which mean it will bind ip to the container in this case giving all IPs to access the DB**

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

**-Where mangoDB port number is 27107 so that port is given to the container**

**4: Backend service (Create the load balancer for the database)**

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

**backend-service.yaml**

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

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: guestbook-back-ser**

**labels:**

**mg-pod: guestbook-db**

**spec:**

**selector:**

**mg-pod: guestbook-db**

**type: ClusterIP**

**ports:**

**- targetPort: 27107**

**port: 27107**

**======The end of the kubernetes guestbook application using mongoDB database ======**

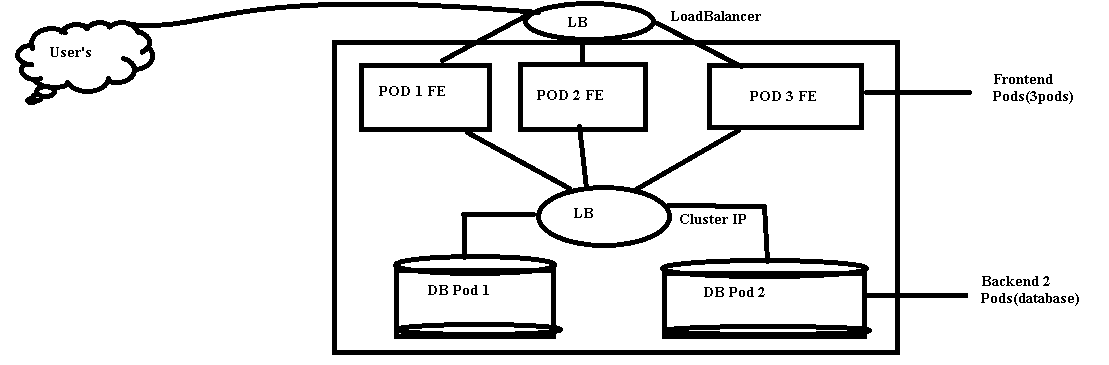
**Difference between NodePort and Load Balancer**

|  |  |  |
| --- | --- | --- |
| **SL/No** | **Node Port** | **Load Balancer** |
| **1** | **We can get the NodePort service as free** | **Load Balancer is the add-on service need to get from the cloud services providers(Chargeable)** |
| **2** | **Node port will take the External ip (where any one can access from the external world) for any one worker node Ip** | **Load Balancer will get fixed ip for the cloud service providers in load balancer only** |
| **3** | **If the worker node fails which ip is attached to service then ip will change with other worker node** | **Where the worker node fails also ip won’t change because it had allocated the dedicated ip to the service** |
|  |  |  |

**Cluster IP – Where servers will not give access to pod to access from external world**

**(This service is used for the accessing form the internal only1)**

**Ex- (use case) – I have cluster with 2 tire application which has frond end and backend (database) where only front should have access to the outside world and backend should have only to access to front end pods**

****