Container Orchestration using Kubernetes

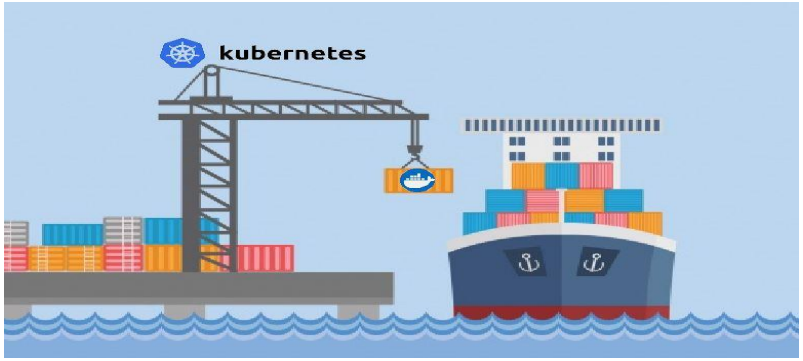
Name: Raghav Sharma
Student ID: 200459151
CS476- Software Development Project

Contents

- Kubernetes Overview
- Containers – Docker
- Container Orchestration?
- Kubernetes Concepts – PODs | ReplicaSets | Deployment | Services
- Networking in Kubernetes
- Kubernetes Management - Kubectl
- Kubernetes Definition Files - YAML

kubernetes or K8s

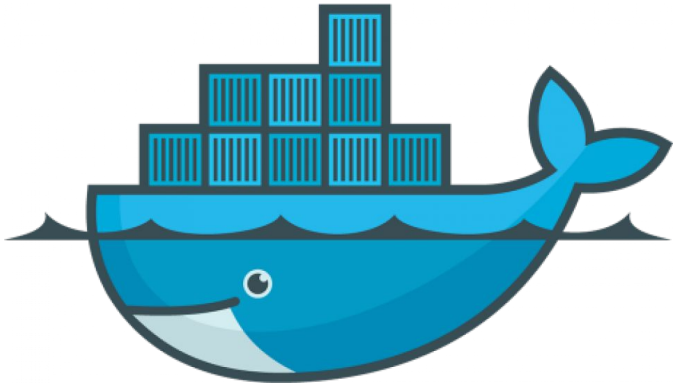
- Kubernetes also known as K8s was built by Google based on their experience running containers in production. It is now an open-source project and is arguably one of the best and most popular container orchestration technologies out there.
- To understand Kubernetes, we must first understand two things – Container and Orchestration. Once we get familiarized with both of these terms we would be in a position to understand what kubernetes is capable of. We will start looking at each of these next.



Containers, Why do you need containers?

Containers are completely isolated environments, as in they can have their own processes or services, their own network interfaces, their own mounts, just like Virtual machines, except that they all share the same OS kernel.
Solves :-

- Compatibility/Dependency
- Long setup time
- Different Dev/Test/Prod environments



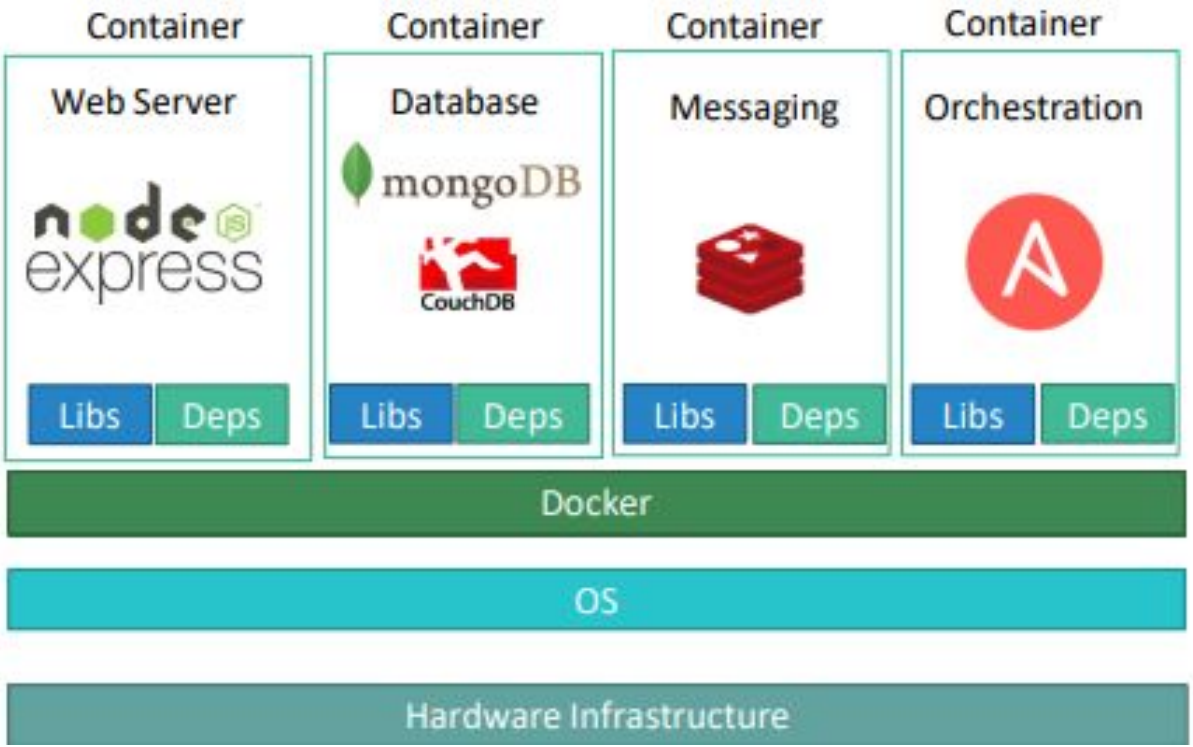
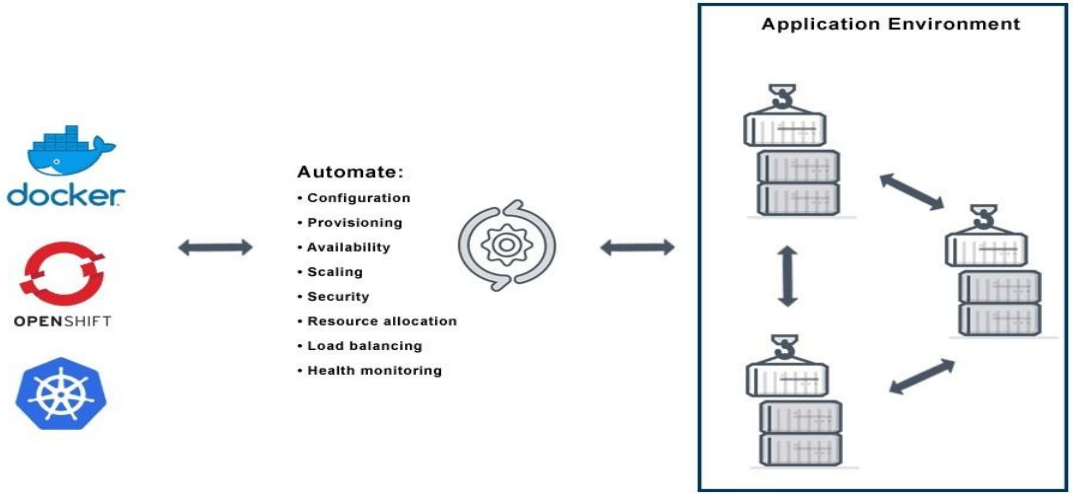


Figure: System Architecture using containers

Container Orchestration

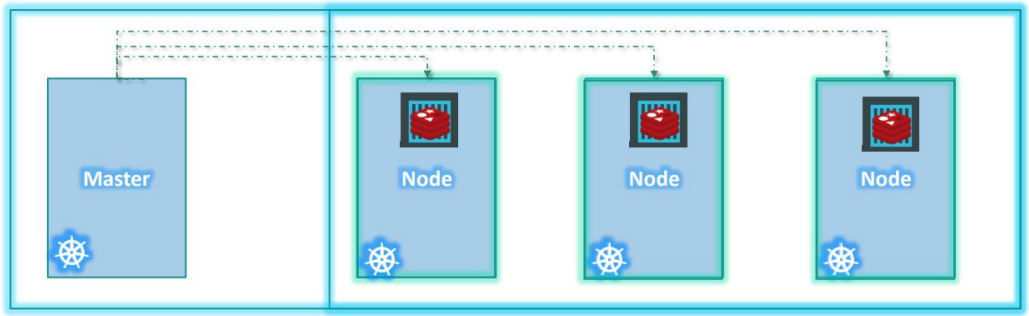


- To enable these functionalities you need an underlying platform with a set of resources. The platform needs to orchestrate the connectivity between the containers and automatically scale up or down based on the load. This whole process of automatically deploying and managing containers is known as Container Orchestration.

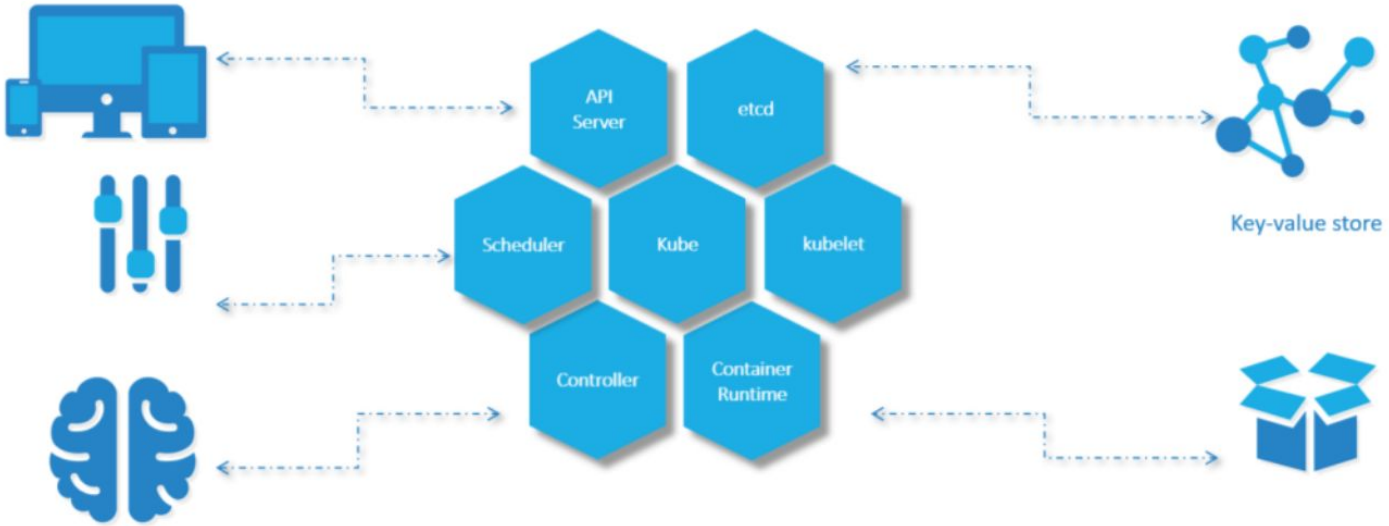
Orchestration Technologies

Some Important terms in Kubernetes

- A **node** is a machine – physical or virtual – on which kubernetes is installed. A node is a worker machine and this is were containers will be launched by kubernetes.
- A **cluster** is a set of nodes grouped together. This way even if one node fails you have your application still accessible from the other nodes.
- The **master** is another node with Kubernetes installed in it, and is configured as a Master. The master watches over the nodes in the cluster and is responsible for the actual orchestration of containers on the worker nodes.



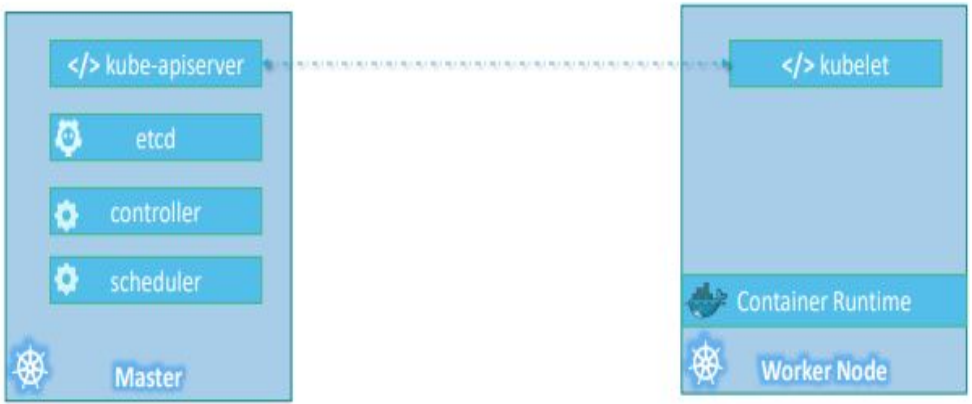
Components in Kubernetes



Master vs Worker Nodes

Components in Master Node: *API server, etcd, controller, Scheduler*

Components in Worker Node: *Kubelet and Container runtime*



kubectl

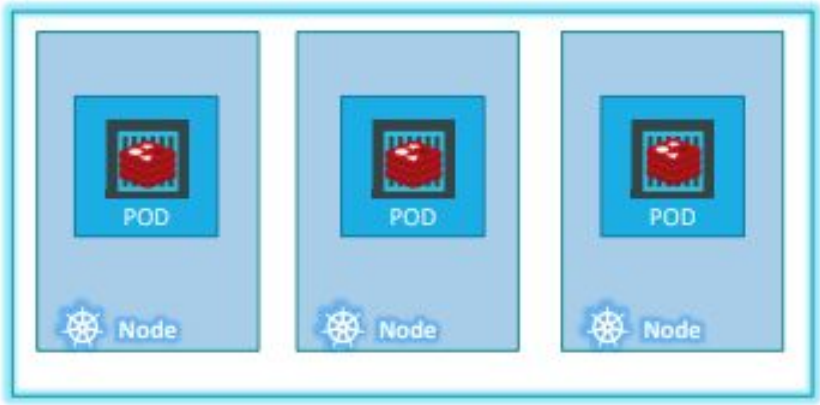
- Kubernetes has command line utilities known as the kube command line tool or kubectl or kube control.
- The kube control tool is used to deploy and manage applications on a kubernetes cluster, to get cluster information, get the status of nodes in the cluster and many other things.

Basic commands:

- kubectl run hello-minikube
- kubectl cluster-info
- kubectl get nodes

I used play-with-k8s.com to do all the K8 stuff.

POD in K8s



The containers are encapsulated into a Kubernetes object known as PODs. A POD is a single instance of an application. A POD is the smallest object, that you can create in kubernetes.

POD in K8s

```
kubectl run nginx --image nginx
```

```
kubectl get pods
```

```
C:\Kubernetes>kubectl get pods
NAME                    READY     STATUS              RESTARTS   AGE
nginx-8586cf59-whssr    0/1       ContainerCreating   0          3s
```

```
C:\Kubernetes>kubectl get pods
NAME                    READY     STATUS    RESTARTS   AGE
nginx-8586cf59-whssr    1/1       Running   0          8s
```

What this command really does is it deploys a docker container by creating a POD. So it first creates a POD automatically and deploys an instance of the nginx docker image.
This is one way of creating Pods, another prefered way is to create a pods using a yaml file.

POD in K8s

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod-pod
  labels:
    app: myapp
spec:
  containers:
    - name: nginx-container
      image: nginx
—----------------------------------------------
```

Command: kubectl create –f pod-definition.yml

POD in K8s

```
> kubectl get pods
NAME            READY     STATUS    RESTARTS   AGE
myapp-pod       1/1       Running   0          20s
```

```
> kubectl describe pod myapp-pod
Name:         myapp-pod
Namespace:    default
Node:         minikube/192.168.99.100
Start Time:   Sat, 03 Mar 2018 14:26:14 +0800
Labels:       app=myapp
              name=myapp-pod
Annotations:  <none>
Status:       Running
IP:           10.244.0.24
Containers:
  nginx:
    Container ID:   docker://830bb56c8c42a86b4bb70e9c1488fae1bc38663e4918b6c2f5a783e7688b8c9d
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:4771d09578c7c6a65299e110b3ee1c0a2592f5ea2618d23e4ffe7a4cab1ce5de
    Port:           <none>
    State:          Running
      Started:      Sat, 03 Mar 2018 14:26:21 +0800
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-x95w7 (ro)
Conditions:
  Type          Status
  Initialized   True
  Ready         True
  PodScheduled  True
Events:
  Type    Reason                 Age   From               Message
  ----    ------                 ----  ----               -------
  Normal  Scheduled              34s   default-scheduler  Successfully assigned myapp-pod to minikube
  Normal  SuccessfulMountVolume  33s   kubelet, minikube  MountVolume.SetUp succeeded for volume "default-token-x95w7"
  Normal  Pulling                33s   kubelet, minikube  pulling image "nginx"
  Normal  Pulled                 27s   kubelet, minikube  Successfully pulled image "nginx"
  Normal  Created                27s   kubelet, minikube  Created container
  Normal  Started                27s   kubelet, minikube  Started container
```

ReplicaSet in K8s

- What is the purpose of a replicaSet?
  Let's go back to our first scenario were we had a single POD running our application. What if for some reason, our application crashes and the POD fails?

- Users will no longer be able to access our application. To prevent users from losing access to our application, we would like to have more than one instance or POD running at the same time in the case if one fails we still have our application running on the other one.

- The replicaSet controller helps us run multiple instances of a single POD in the kubernetes cluster thus providing High Availability.

- Even if you have a single POD, the replicaSet can help by automatically bringing up a new POD when the existing one fails. Thus the replication controller ensures that the specified number of PODs are running at all times. Even if it's just 1 or 100.

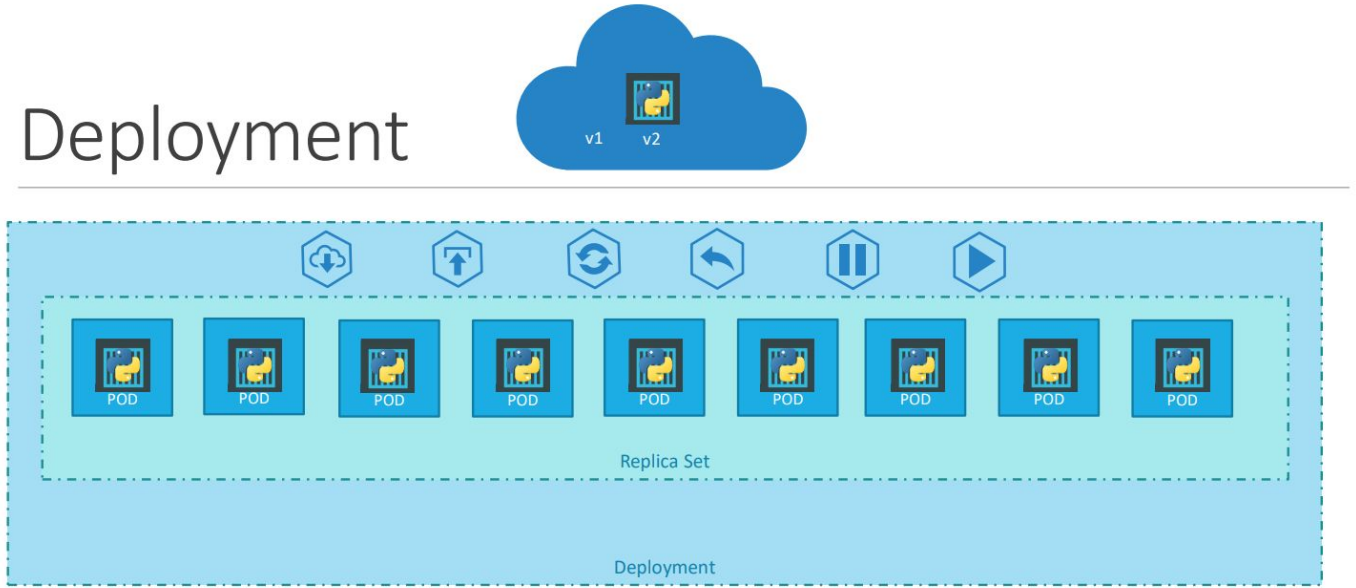
Replica Set in K8s

replicaset-definition.yml

```
apiVersion: apps/v1

kind: ReplicaSet

metadata:
  name: myapp-replicaset
  labels:
    app: myapp
    type: front-end

spec:
  replicas: 3
  selector:
    matchLabels:
      type: front-end
  template:
    metadata:
      Name: myapp-pod
      labels:
        name: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx
```

```
kubectl create –f replicaset-definition.yml

kubectl get replicaset

kubectl replace -f replicaset-definition.yml

kubectl scale --replicas=6 –f replicaset-definition.yml

kubectl scale --replicas=6 replicaset myapp-replicaset

kubectl delete replicaset myapp-replicaset
```

Deployment in K8s



Deployment in K8s

deployment-definition.yml

```
apiVersion: apps/v1

kind: Deployment

metadata:
  name: myapp-deployment
  labels:
    app: myapp
    type: front-end

spec:
  replicas: 3
  selector:
    matchLabels:
      type: front-end
  template:
    metadata:
      Name: myapp-pod
      labels:
        name: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx
```

```
> kubectl create -f deployment-definition.yml
deployment "myapp-deployment" created
```

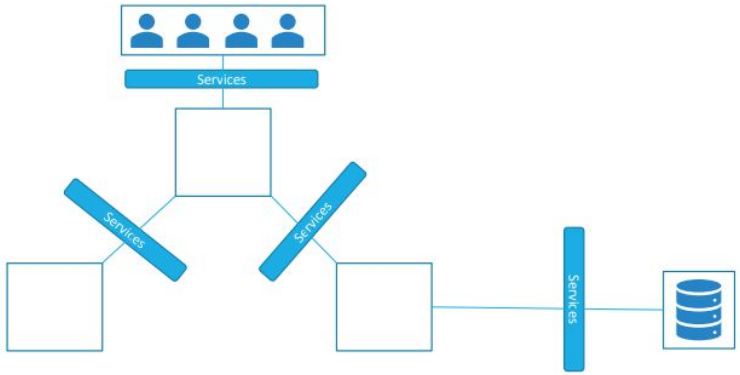
```
> kubectl get deployments
NAME              DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
myapp-deployment  3         3         3            3           21s
```

```
> kubectl get replicaset
NAME                          DESIRED   CURRENT   READY     AGE
myapp-deployment-6795844b58   3         3         3         2m
```

```
> kubectl get pods
NAME                              READY     STATUS    RESTARTS   AGE
myapp-deployment-6795844b58-5rbjl   1/1       Running   0          2m
myapp-deployment-6795844b58-h4w55   1/1       Running   0          2m
myapp-deployment-6795844b58-1fjhv   1/1       Running   0          2m
```

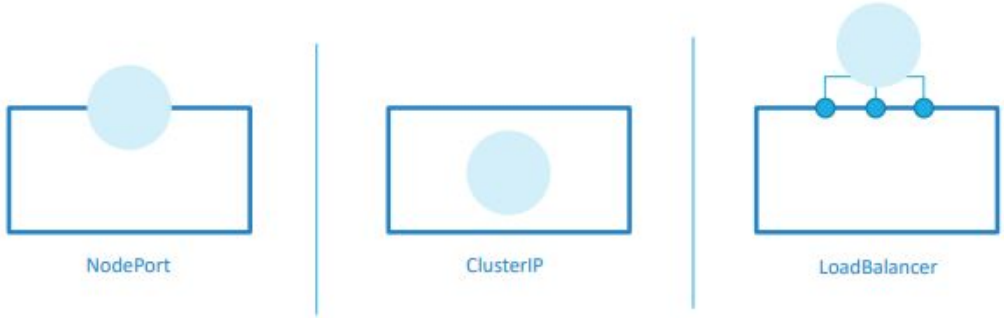
Services in K8s

- Kubernetes Services enable communication between various components within and outside of the application.
- Kubernetes Services helps us connect applications together with other applications or users.
- Services enable the front-end application to be made available to users, it helps communication between back-end and front-end PODs, and helps in establishing connectivity to an external data source.



Types of Services

- NodePort were the service makes an internal POD accessible on a Port on the Node.
- The second is ClusterIP – and in this case the service creates a virtual IP inside the cluster to enable communication between different services such as a set of front-end servers to a set of backend servers.
- The third type is a LoadBalancer, were it provisions a load balancer for our service in supported cloud providers.



References

- https://kubernetes.io/docs/home/
- https://medium.com/the-programmer/kubernetes-fundamentals-for-absolute-beginners-architecture-components-1f7cda8ea536
-
- https://www.vmware.com/topics/glossary/content/container-orchestration.html#:~:text=Container%20orchestration%20is%20the%20automation,networking%2C%20load%20balancing%20and%20more.
- Vohra, Deepak. (2016). Kubernetes Microservices with Docker. 10.1007/978-1-4842-1907-2.

THANK YOU !