Kubernetes (k8s)

About me

Kshithija Liyanage

Senior Infrastructure Engineer

TECHSER division

LSEG Technology (Previously MIT)

We do consultancy for

Servers

OS and Containerized platforms

Cloud

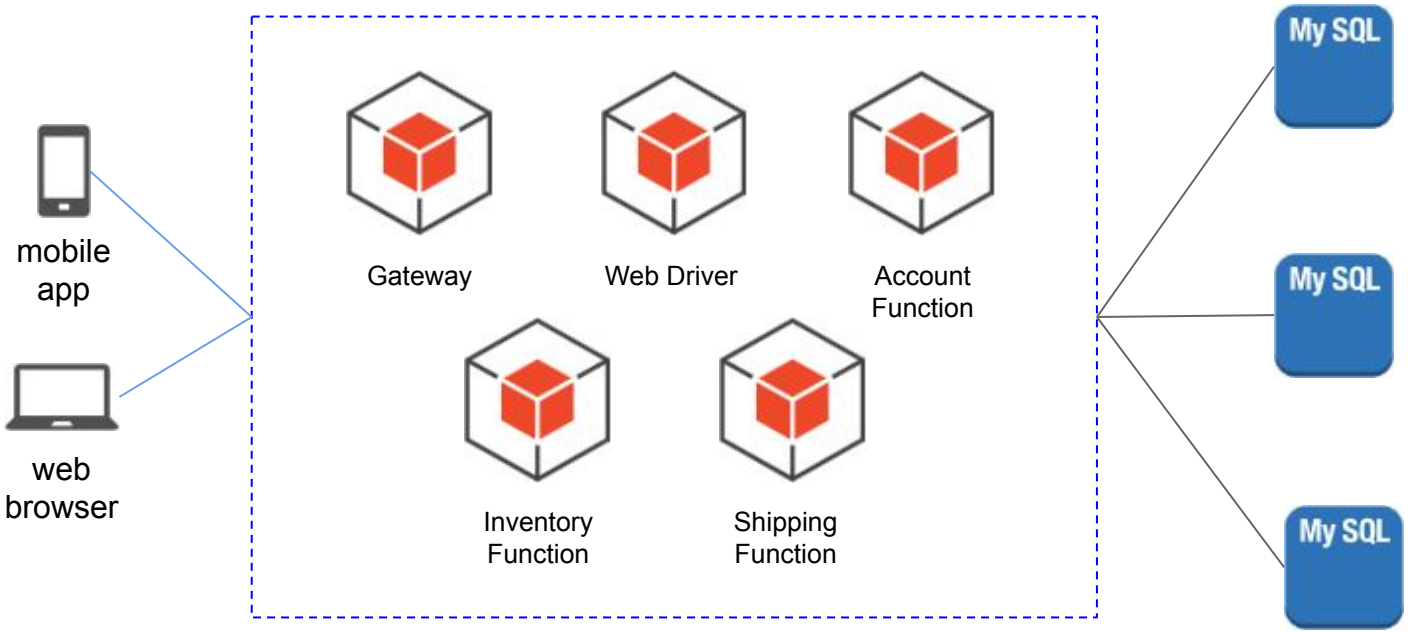
Other system software

Flow of the presentation

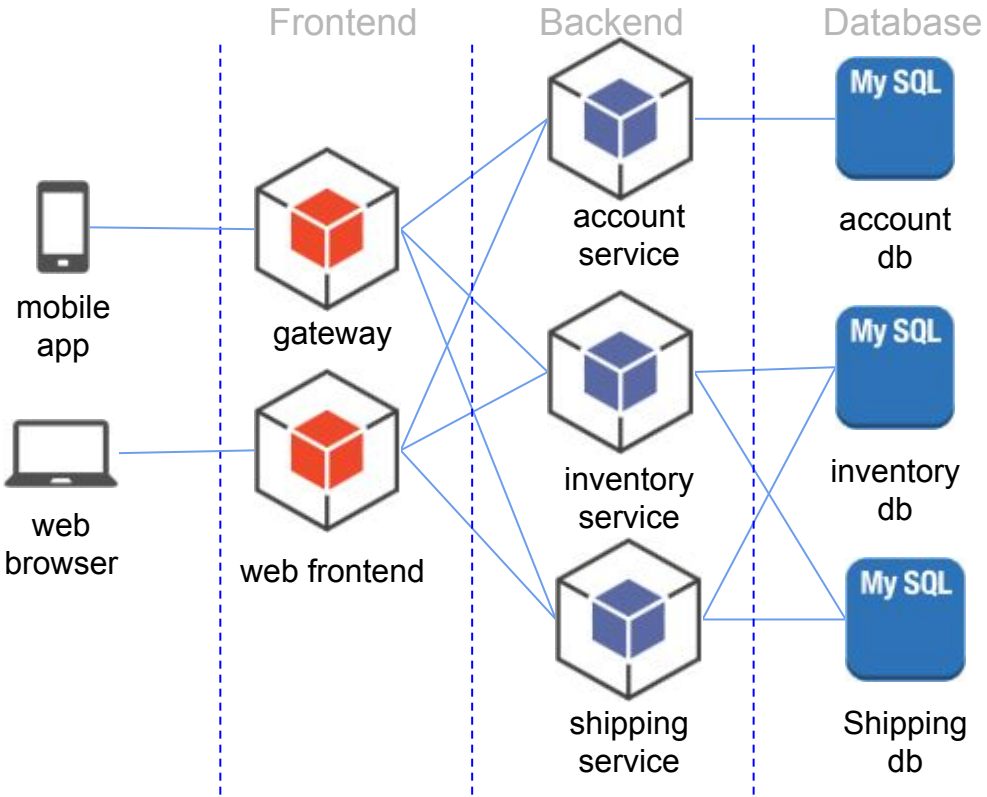
- Microservices architecture
- Microservices with Kubernetes
- Learn by reverse engineering
- First microservice in kubernetes
- Scaling
- High availability and FT
- Rolling upgrades
- CI/CD

Microservice architecture

One-thing that do everything - Monolithic

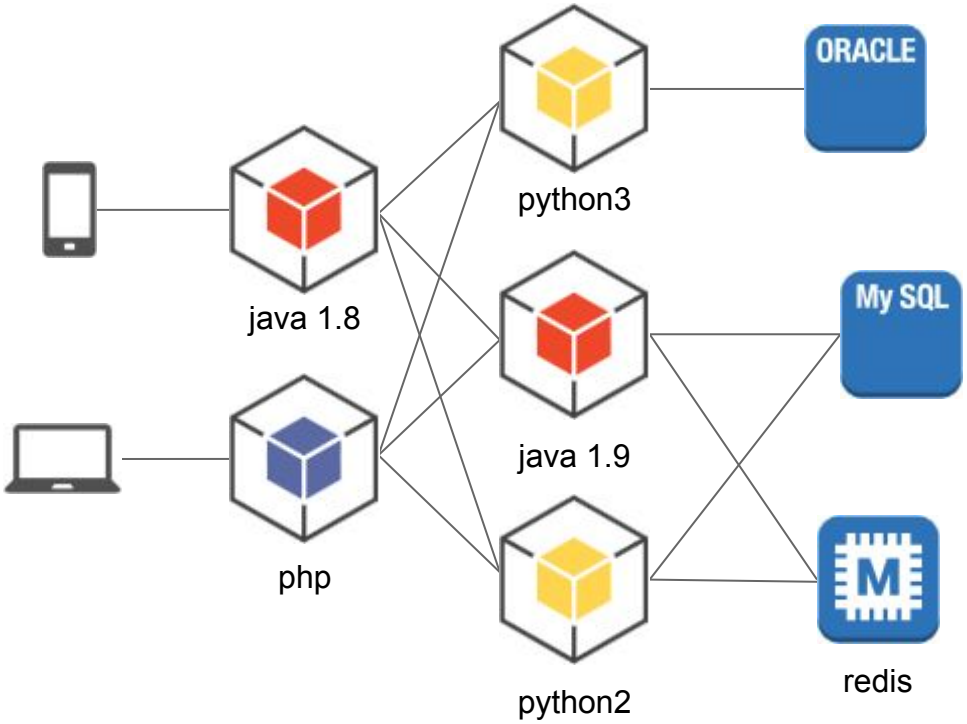


Do one thing and do it right - Microservices

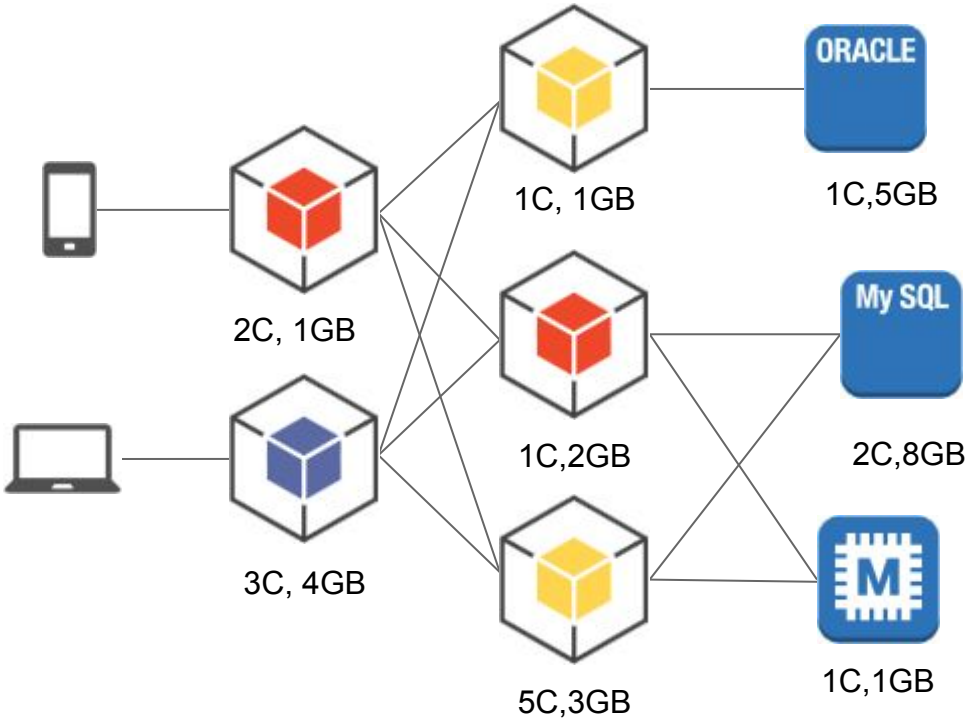


Microservices with Kubernetes

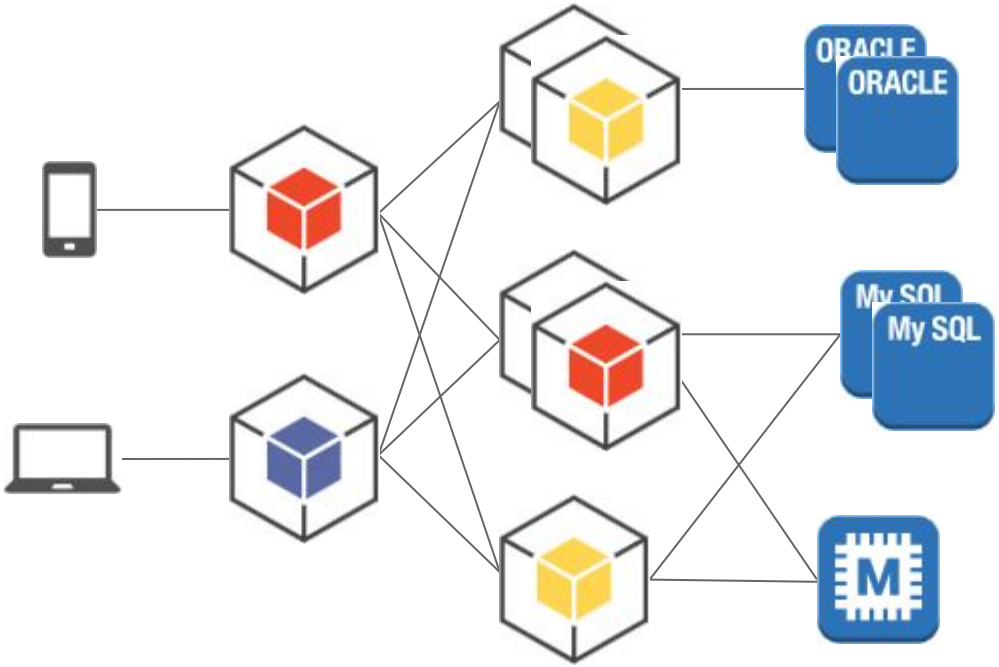
Technologies coexists



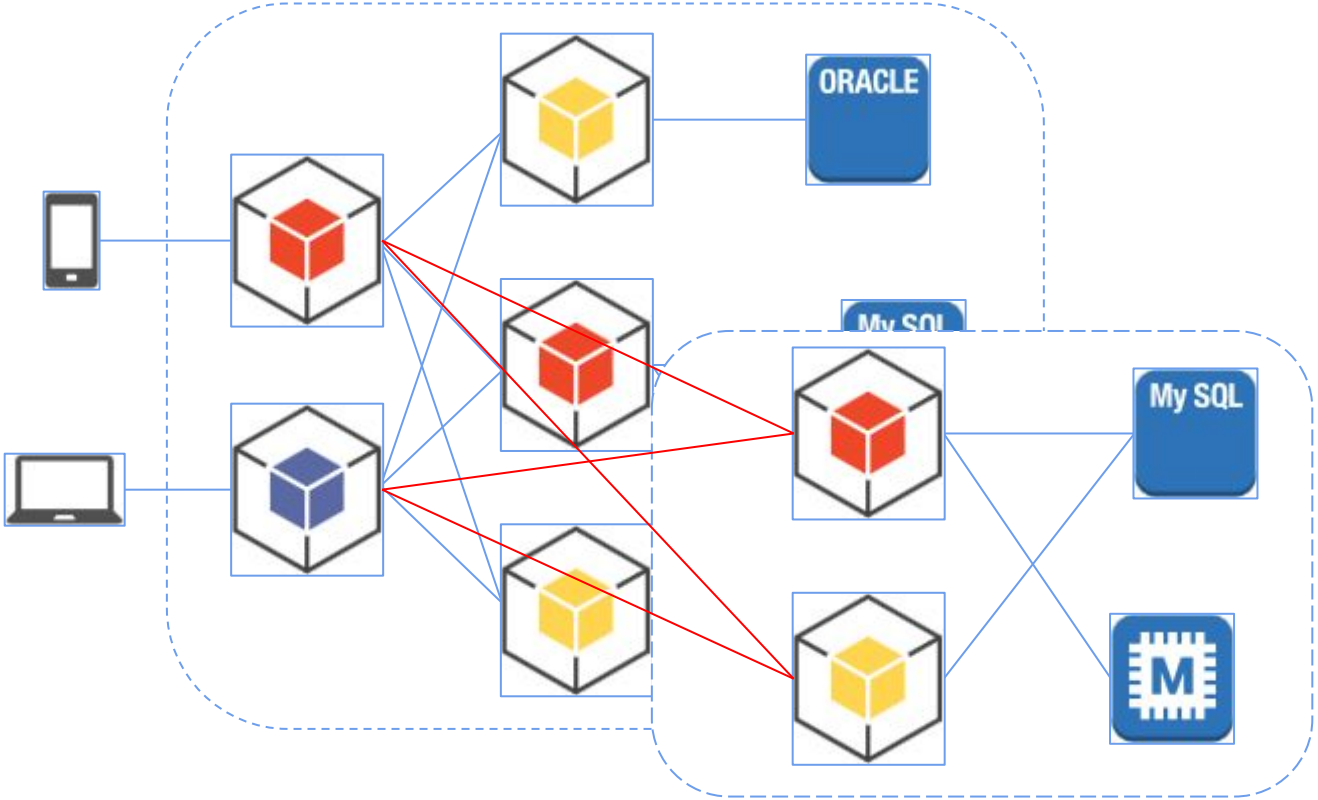
No resource overprovisioned



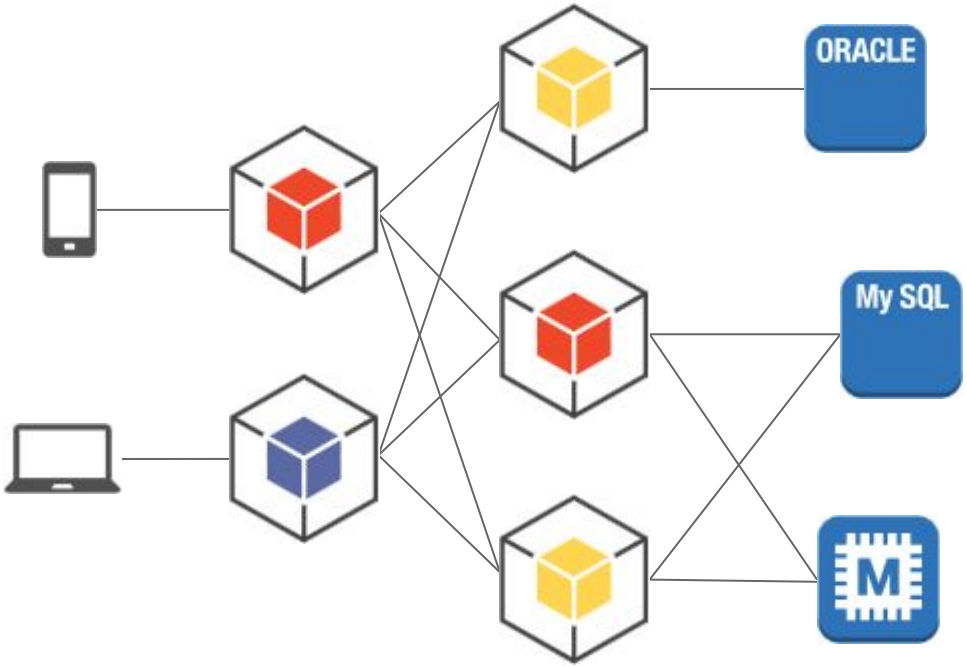
Scalable



Scalable ctd.

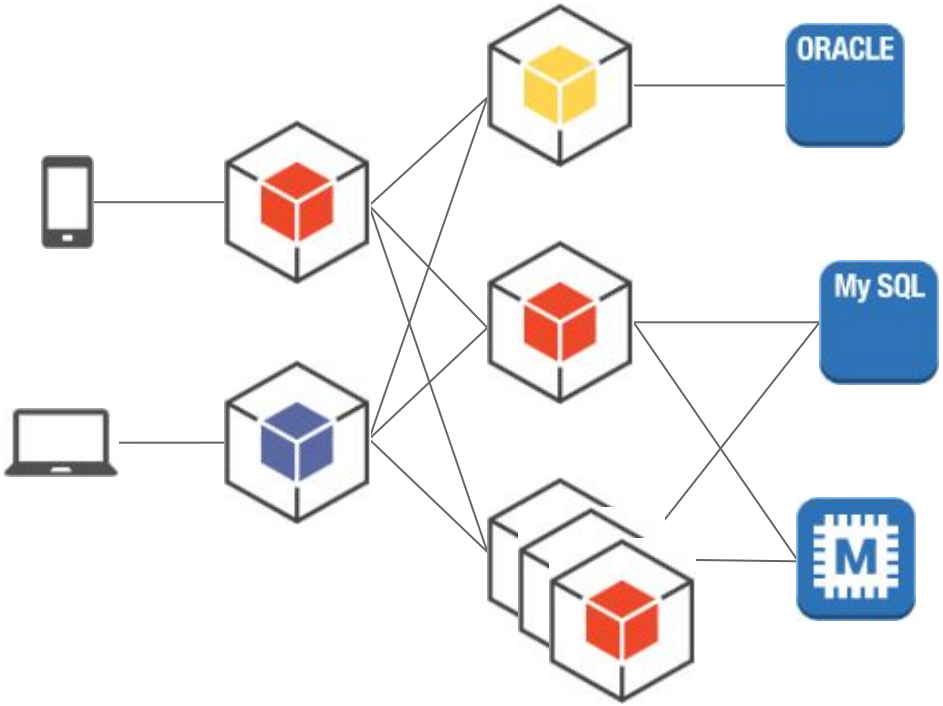


Rolling upgrade



v1.0.0

Highly available and Fault tolerant



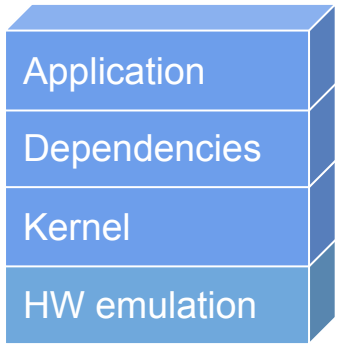
Kubernetes

- Kubernetes being the state-of-the-art platform for microservices that support above traits
- It is called the orchestration platform

What is running

- These are called pods
- It’s a stripped-down version of VM

```
kshithija-hq@kshithijahq-ThinkPad-X230:~$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
maeno-frontend-8f4cd9f-5rvcf      1/1     Running   2          2d17h
maeno-frontend-8f4cd9f-jxw2c      1/1     Running   2          2d17h
kshithija-hq@kshithijahq-ThinkPad-X230:~$
```



Learn by reverse engineering

Let's install kubernetes

Minikube

Get Started: https://minikube.sigs.k8s.io/docs/start/

Kubeadm

Install Docker: https://docs.docker.com/engine/install/

Install Kubeadm:
https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

Use a cloud service

Lab Environment: http://labs.play-with-k8s.com/

Minikube

- Easiest way to run Kubernetes locally
- Its a single-node Virtual Machine

```
PS C:\Windows\system32> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\Windows\system32>
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> minikube start
* minikube v1.23.2 on Microsoft Windows 10 Pro 10.0.19043 Build 19043
* minikube 1.24.0 is available! Download it: https://github.com/kubernetes/minikube/releases/tag/v1.24.0
* To disable this notice, run: 'minikube config set WantUpdateNotification false'

* Using the hyperv driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Restarting existing hyperv VM for "minikube" ...
* Preparing Kubernetes v1.22.2 on Docker 20.10.8 ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\Windows\system32>
```

Lets install kubectl

Linux:

Install Kubectl: https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/

Windows:

Install Kubectl: https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/

Helloworld to kubernetes

Install Frontend:

- kubectl run maeno-frontend --image=kshithija/maeno-fe:v1.0.0 --requests=cpu=200m

Install Backend:

- kubectl run maeno-backend --image=kshithija/maeno-be:v1.0.0 --requests=cpu=100m

This is called imperative method

Deploy backend using a config files

kubectl run maeno-backend --image=kshithija/maeno-be:v1.0.0

=

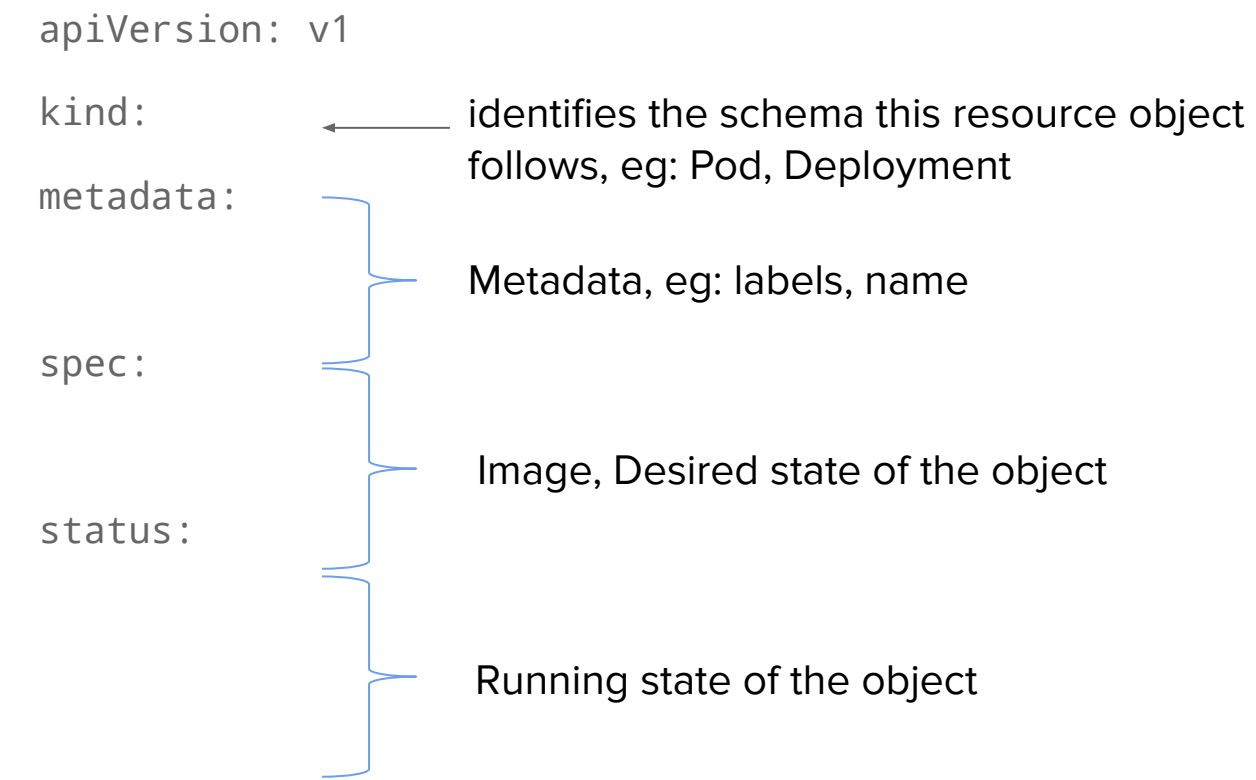
kubectl apply -f maeno-be.yaml

maeno-be.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: maeno-backend
  name: maeno-backend
spec:
  containers:
  - image: kshithija/maeno-be:v1.0.0
    name: maeno-backend
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

This is called declarative method

Syntax


apiVersion: v1
kind:
identifies the schema this resource object follows, eg: Pod, Deployment
metadata:
Metadata, eg: labels, name
spec:
Image, Desired state of the object
status:
Running state of the object

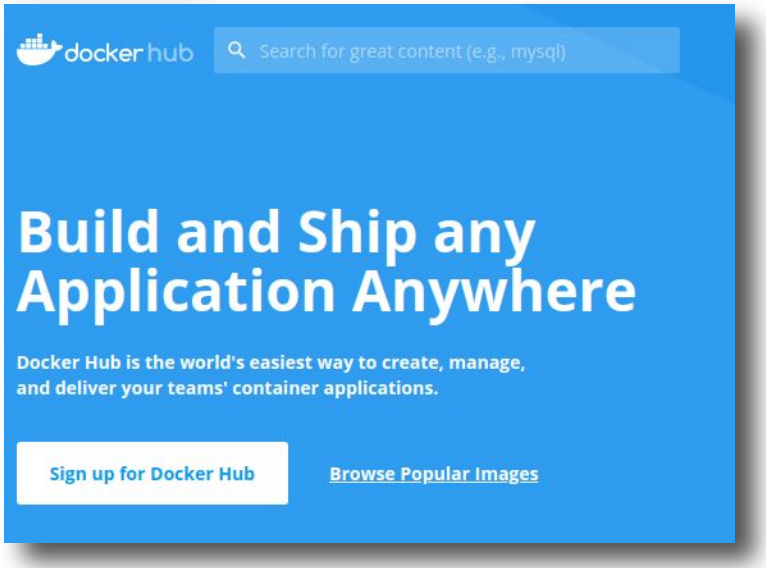
First micro service

Two-tier microservice application flow

Pod on rest is called 'image'

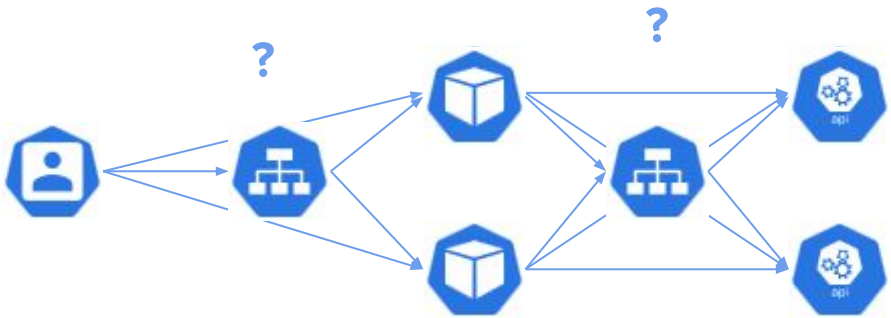
kubectl run maeno-backend --image=kshithija/maeno-be:v1.0.0

```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: maeno-backend
  name: maeno-backend
spec:
  containers:
  - image: kshithija/maeno-be:v1.0.0
    name: maeno-backend
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```



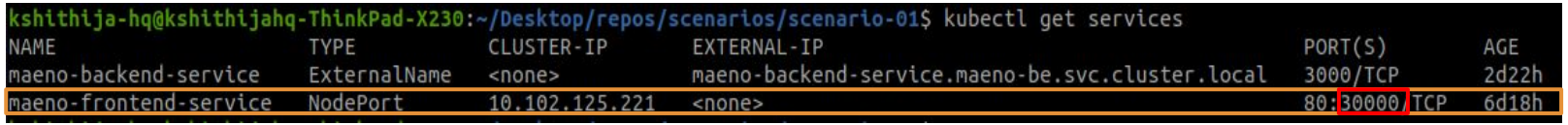
https://hub.docker.com/

Connectivity between pods



What is a ‘Service’

‘Services’ provide connectivity and load balancing



Scaling

It's not just the PODs and the Services

Deployment

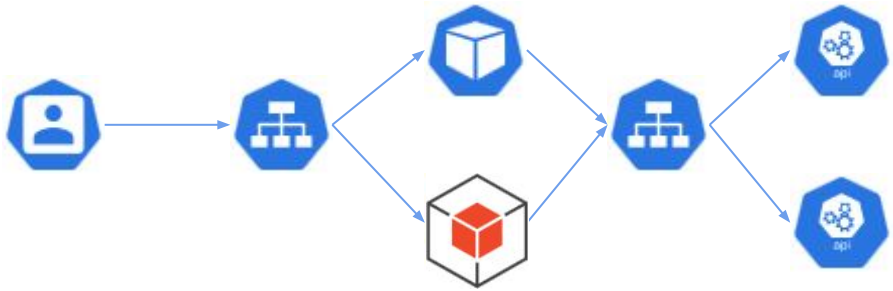
Replica Set

```
kshithija-hq@kshithijahq-ThinkPad-X230:~/Desktop/repos/scenarios/scenario-01$ kubectl get all
NAME                                 READY   STATUS    RESTARTS   AGE
pod/maeno-frontend-8f4cd9f-5rvcf     1/1     Running   2          2d19h
pod/maeno-frontend-8f4cd9f-jxw2c     1/1     Running   2          2d19h

NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP                                    PORT(S)        AGE
service/maeno-backend-service        ExternalName   <none>           maeno-backend-service.maeno-be.svc.cluster.local   3000/TCP       2d23h
service/maeno-frontend-service       NodePort       10.102.125.221   <none>                                         80:30000/TCP   6d19h

NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/maeno-frontend       2/2     2            2           6d19h

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/maeno-frontend-8f4cd9f     2         2         2       2d19h
```

This is all that we provisioned so far

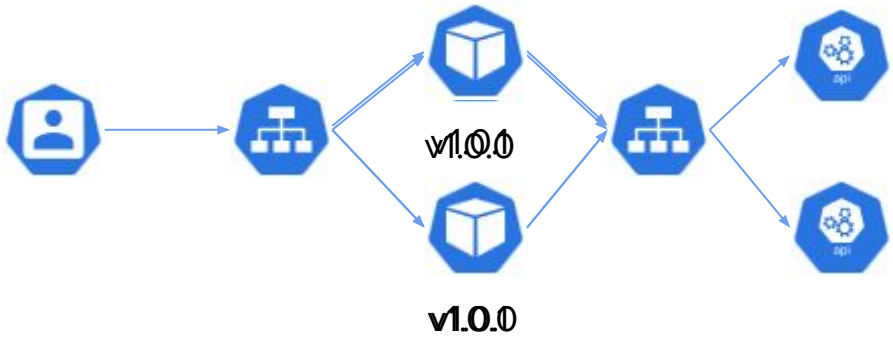
High availability

Let's kill a process



Rolling upgrade

Let's upgrade the frontend



Kubernetes in production

[Amazon EKS]

[Google GKE]

[Azure AKS]

CI/CD