Kubernetes

-> follows client-server architecture

Master Consists of

- Kube-apiserver -> Central management: recieve all Rest Req.
- etcd-storage -> store application metadata
- Kube-Controller manager -> Runs controller process in bg.
- Cloud-Controller manager -> " in underlying Cloud provider
- Kube-scheduler -> schedule the pods
- DNS-server

Node Consists of -> Communicates with master and report state

- Kubelet
- Kube-proxy

} on top of Docker

↳ Isolated networking and forwarding request

Kubectl -> Cli to interact with Kube-apiserver

Kubernetes objects

Pod -> One or more containers controlled by one application encapsulates containers.

Service → Gateway to request other physical pods

Deployment → desired replica set of a pod

Kubernetes objects control

- Declarative → Using Yaml file
- Iterative → Using CLI

Example Yaml file

Pod →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

apiVersion: v1 / kind: Pod → Version declration

metadata: name: nginx, labels: name: nginx → Pod name and labeling

spec: containers: → list of containers

- name: nginx → Indicates array

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-nginx
  labels:
    run: my-nginx
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    run: my-nginx
```

selector: run: my-nginx → discovering service

Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx
        ports:
        - containerPort: 80
```

→ newly implemented

→ template for replicas