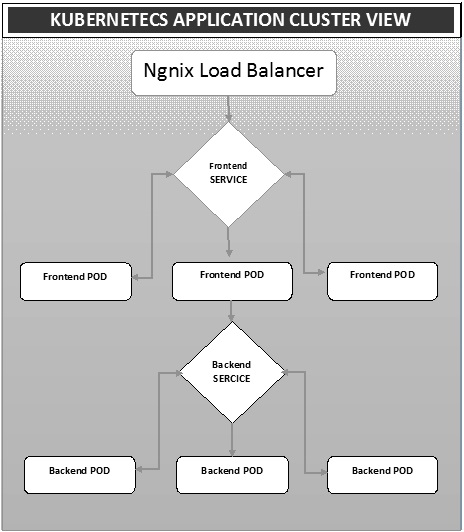
# Kubernetes - App Deployment

Deployment is a method of converting images to containers and then allocating those images to pods in the Kubernetes cluster. This also helps in setting up the application cluster which includes deployment of service, pod, replication controller and replica set. The cluster can be set up in such a way that the applications deployed on the pod can communicate with each other.

In this setup, we can have a load balancer setting on top of one application diverting traffic to a set of pods and later they communicate to backend pods. The communication between pods happen via the service object built in Kubernetes.



## Ngnix Load Balancer Yaml File

apiVersion: v1

kind: Service

metadata:

name: oppv-dev-nginx

labels:

k8s-app: omni-ppv-api

spec:

type: NodePort

ports:

- port: 8080

nodePort: 31999

name: omninginx

selector:

k8s-app: appname

component: nginx

env: dev

## Ngnix Replication Controller Yaml

apiVersion: v1

kind: ReplicationController

metadata:

name: appname

spec:

replicas: replica\_count

template:

metadata:

name: appname

labels:

k8s-app: appname

component: nginx

env: env\_name

spec:

nodeSelector:

resource-group: oppv

containers:

- name: appname

image: IMAGE\_TEMPLATE

imagePullPolicy: Always

ports:

- containerPort: 8080

resources:

requests:

memory: "request\_mem"

cpu: "request\_cpu"

limits:

memory: "limit\_mem"

cpu: "limit\_cpu"

env:

- name: BACKEND\_HOST

value: oppv-env\_name-node:3000

## Frontend Service Yaml File

apiVersion: v1

kind: Service

metadata:

name: appname

labels:

k8s-app: appname

spec:

type: NodePort

ports:

- name: http

port: 3000

protocol: TCP

targetPort: 3000

selector:

k8s-app: appname

component: nodejs

env: dev

## Frontend Replication Controller Yaml File

apiVersion: v1

kind: ReplicationController

metadata:

name: Frontend

spec:

replicas: 3

template:

metadata:

name: frontend

labels:

k8s-app: Frontend

component: nodejs

env: Dev

spec:

nodeSelector:

resource-group: oppv

containers:

- name: appname

image: IMAGE\_TEMPLATE

imagePullPolicy: Always

ports:

- containerPort: 3000

resources:

requests:

memory: "request\_mem"

cpu: "limit\_cpu"

limits:

memory: "limit\_mem"

cpu: "limit\_cpu"

env:

- name: ENV

valueFrom:

configMapKeyRef:

name: appname

key: config-env

## Backend Service Yaml File

apiVersion: v1

kind: Service

metadata:

name: backend

labels:

k8s-app: backend

spec:

type: NodePort

ports:

- name: http

port: 9010

protocol: TCP

targetPort: 9000

selector:

k8s-app: appname

component: play

env: dev

## Backed Replication Controller Yaml File

apiVersion: v1

kind: ReplicationController

metadata:

name: backend

spec:

replicas: 3

template:

metadata:

name: backend

labels:

k8s-app: beckend

component: play

env: dev

spec:

nodeSelector:

resource-group: oppv

containers:

- name: appname

image: IMAGE\_TEMPLATE

imagePullPolicy: Always

ports:

- containerPort: 9000

command: [ "./docker-entrypoint.sh" ]

resources:

requests:

memory: "request\_mem"

cpu: "request\_cpu"

limits:

memory: "limit\_mem"

cpu: "limit\_cpu"

volumeMounts:

- name: config-volume

mountPath: /app/vipin/play/conf

volumes:

- name: config-volume

configMap:

name: appname