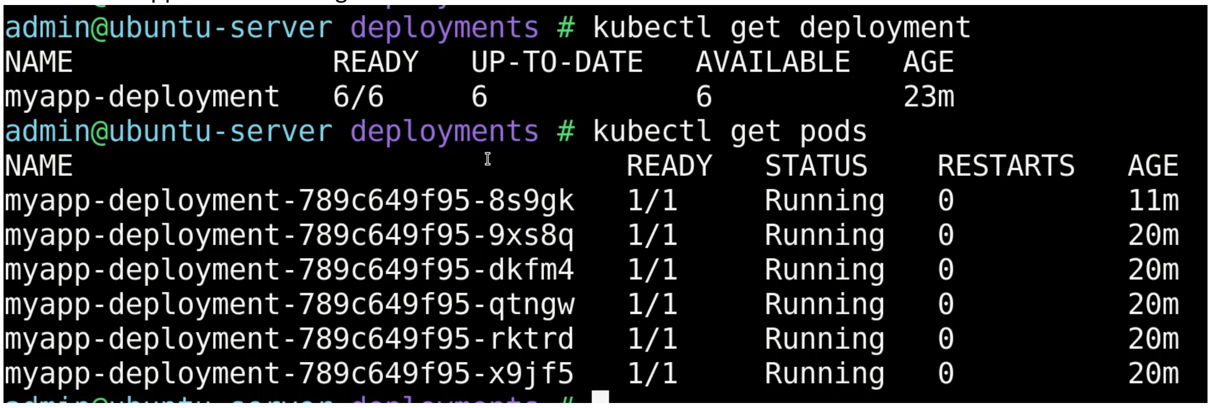
SERVICES :

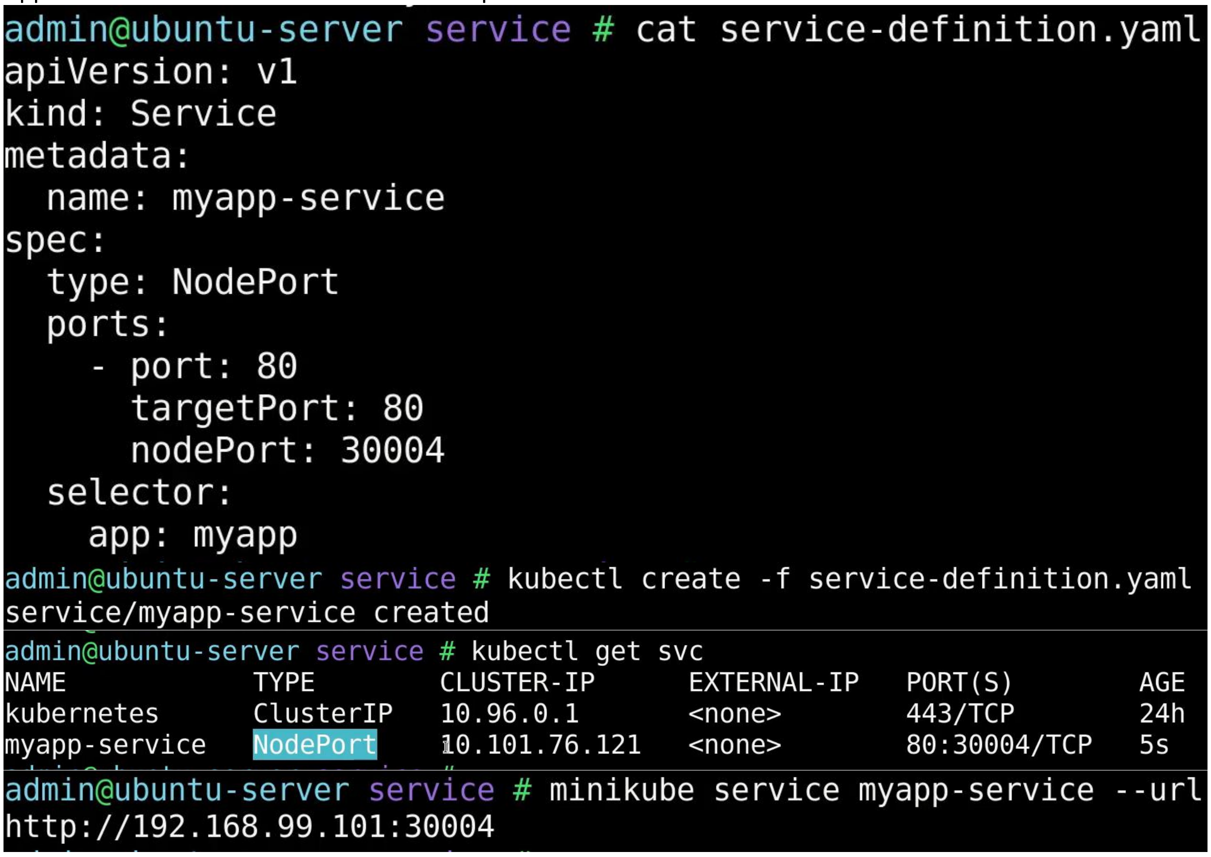
We have an application running on cluster .



```
admin@ubuntu-server deployments # kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
myapp-deployment   6/6     6            6           23m
admin@ubuntu-server deployments # kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
myapp-deployment-789c649f95-8s9gk   1/1     Running   0          11m
myapp-deployment-789c649f95-9xs8q   1/1     Running   0          20m
myapp-deployment-789c649f95-dkfm4   1/1     Running   0          20m
myapp-deployment-789c649f95-qtngw   1/1     Running   0          20m
myapp-deployment-789c649f95-rktrd   1/1     Running   0          20m
myapp-deployment-789c649f95-x9jf5   1/1     Running   0          20m
```

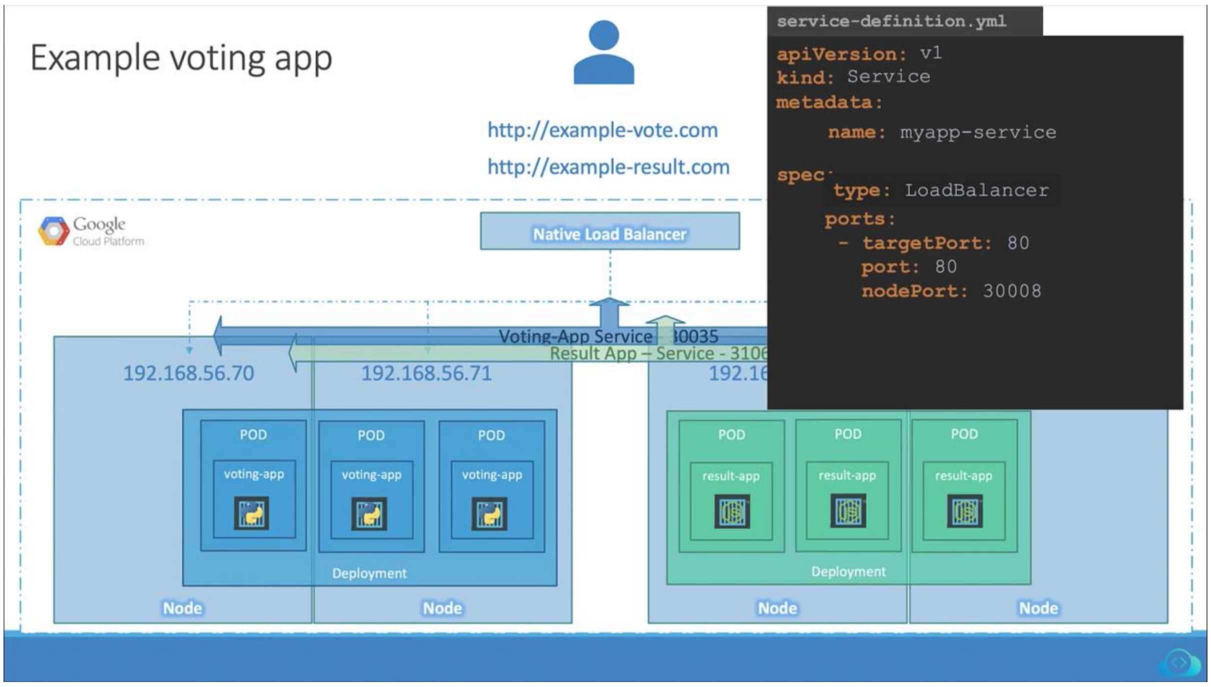
In order for end user to access our application we have to create service .

Application will be made accessible on nodeport .

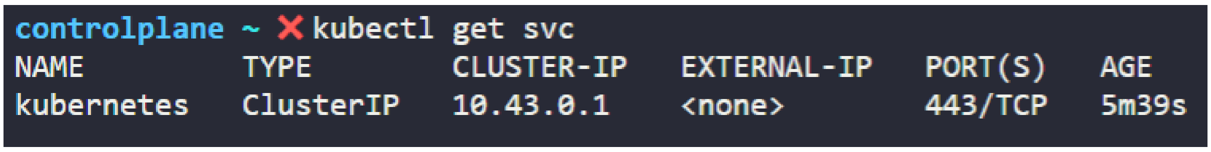


```
admin@ubuntu-server service # cat service-definition.yaml
apiVersion: v1
kind: Service
metadata:
  name: myapp-service
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30004
  selector:
    app: myapp
admin@ubuntu-server service # kubectl create -f service-definition.yaml
service/myapp-service created
admin@ubuntu-server service # kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1       <none>        443/TCP        24h
myapp-service   NodePort    10.101.76.121   <none>        80:30004/TCP   5s
admin@ubuntu-server service # minikube service myapp-service --url
http://192.168.99.101:30004
```

Url where our service will be accessible .

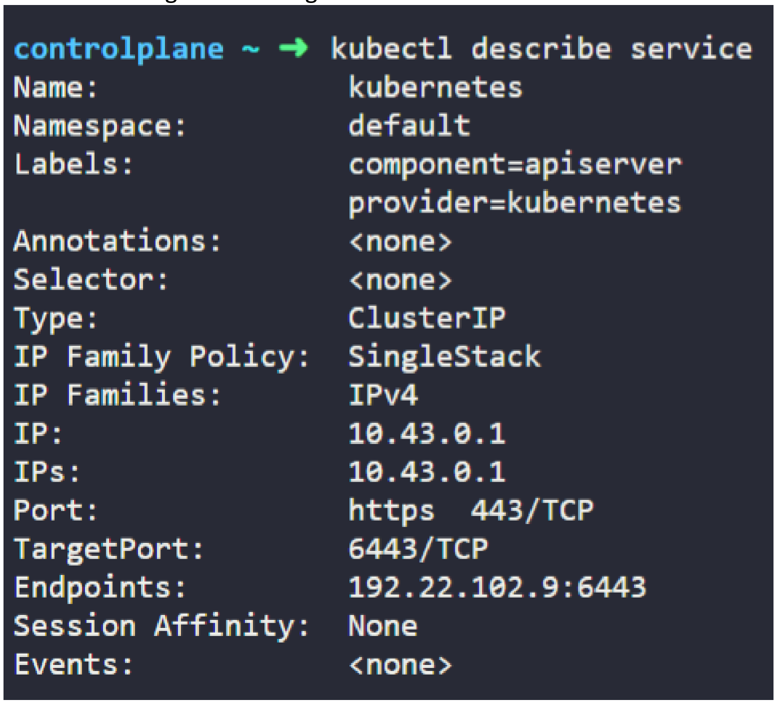


```
service-definition.yml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: LoadBalancer
  ports:
   - targetPort: 80
     port: 80
     nodePort: 30008
```



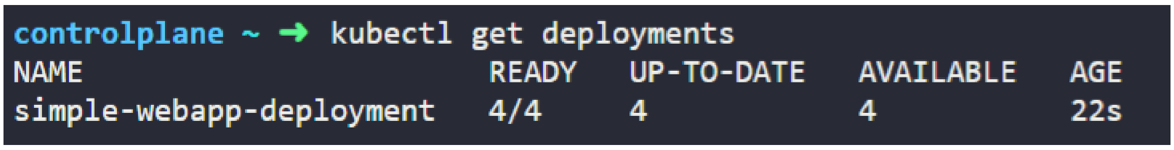
```
controlplane ~ ✗ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   5m39s
```

How many Services exist on the system? 1
What is the type of the default kubernetes service? Clusterip
What is the targetPort configured on the kubernetes service? 6443



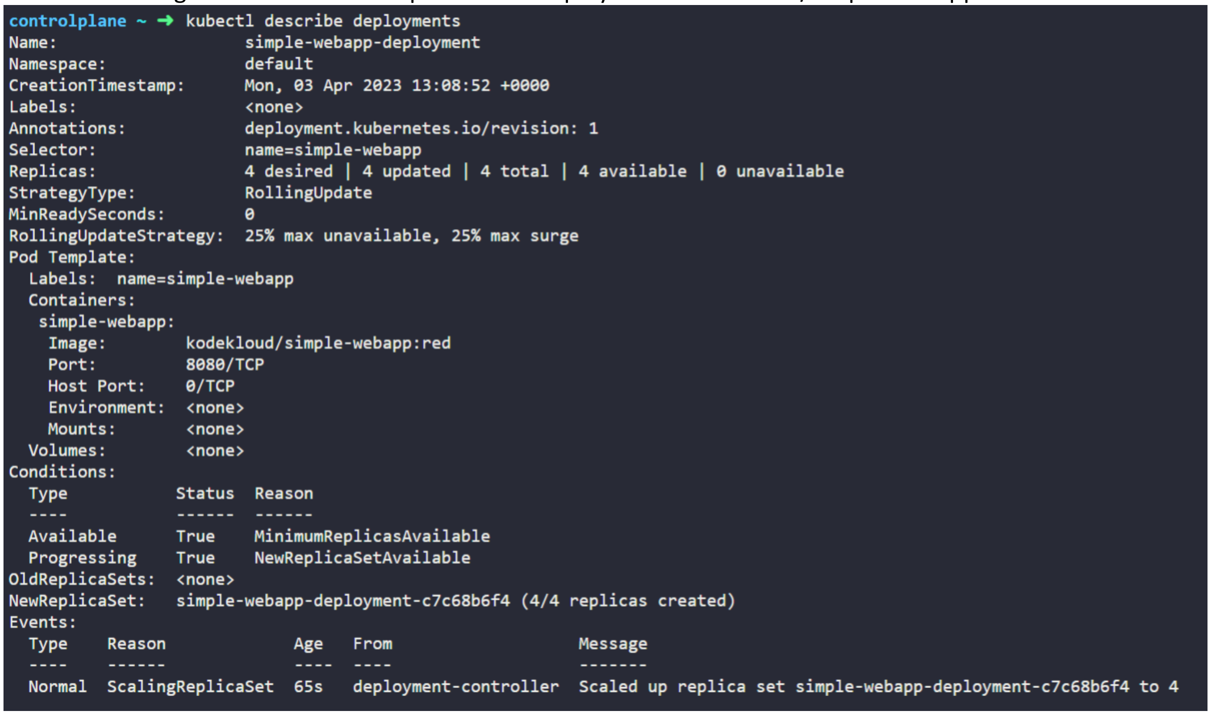
```
controlplane ~ ➜ kubectl describe service
Name:                 kubernetes
Namespace:            default
Labels:               component=apiserver
                      provider=kubernetes
Annotations:          <none>
Selector:             <none>
Type:                 ClusterIP
IP Family Policy:     SingleStack
IP Families:          IPv4
IP:                   10.43.0.1
IPs:                  10.43.0.1
Port:                 https  443/TCP
TargetPort:           6443/TCP
Endpoints:            192.22.102.9:6443
Session Affinity:     None
Events:               <none>
```

How many labels are configured on the kubernetes service? 2
How many Endpoints are attached on the kubernetes service? 1
How many Deployments exist on the system now? 1



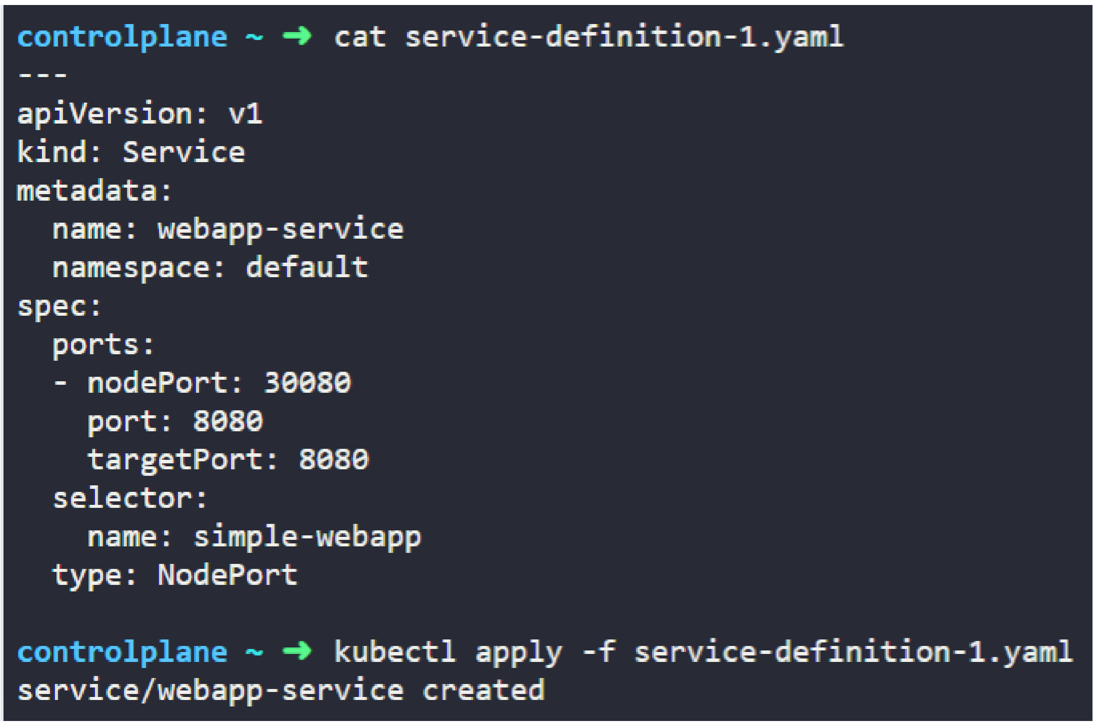
```
controlplane ~ ➜  kubectl get deployments
NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
simple-webapp-deployment   4/4     4            4           22s
```

What is the image used to create the pods in the deployment? kodekloud/simple-webapp:red

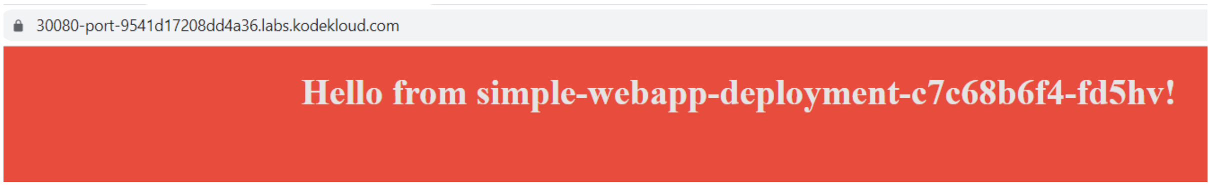


```
controlplane ~ ➜  kubectl describe deployments
Name:                   simple-webapp-deployment
Namespace:              default
CreationTimestamp:      Mon, 03 Apr 2023 13:08:52 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               name=simple-webapp
Replicas:               4 desired | 4 updated | 4 total | 4 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  name=simple-webapp
  Containers:
   simple-webapp:
    Image:        kodekloud/simple-webapp:red
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   simple-webapp-deployment-c7c68b6f4 (4/4 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  65s   deployment-controller  Scaled up replica set simple-webapp-deployment-c7c68b6f4 to 4
```

Create a new service to access the web application using the service-definition-1.yaml file.
Name: webapp-service
Type: NodePort
targetPort: 8080
port: 8080
nodePort: 30080
selector:
  name: simple-webapp



```
controlplane ~ ➜  cat service-definition-1.yaml
---
apiVersion: v1
kind: Service
metadata:
  name: webapp-service
  namespace: default
spec:
  ports:
  - nodePort: 30080
    port: 8080
    targetPort: 8080
  selector:
    name: simple-webapp
  type: NodePort

controlplane ~ ➜  kubectl apply -f service-definition-1.yaml
service/webapp-service created
```



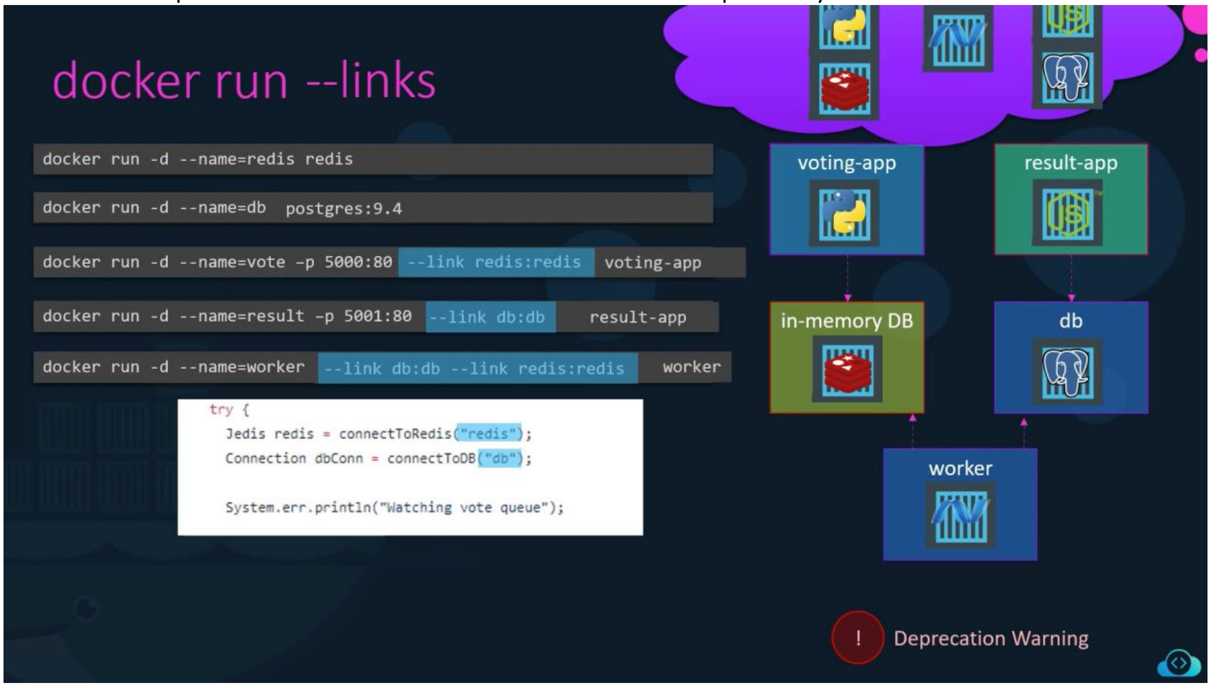
30080-port-9541d17208dd4a36.labs.kodekloud.com

Hello from simple-webapp-deployment-c7c68b6f4-fd5hv!

Web app is accessible after exposing service .

Microservices :

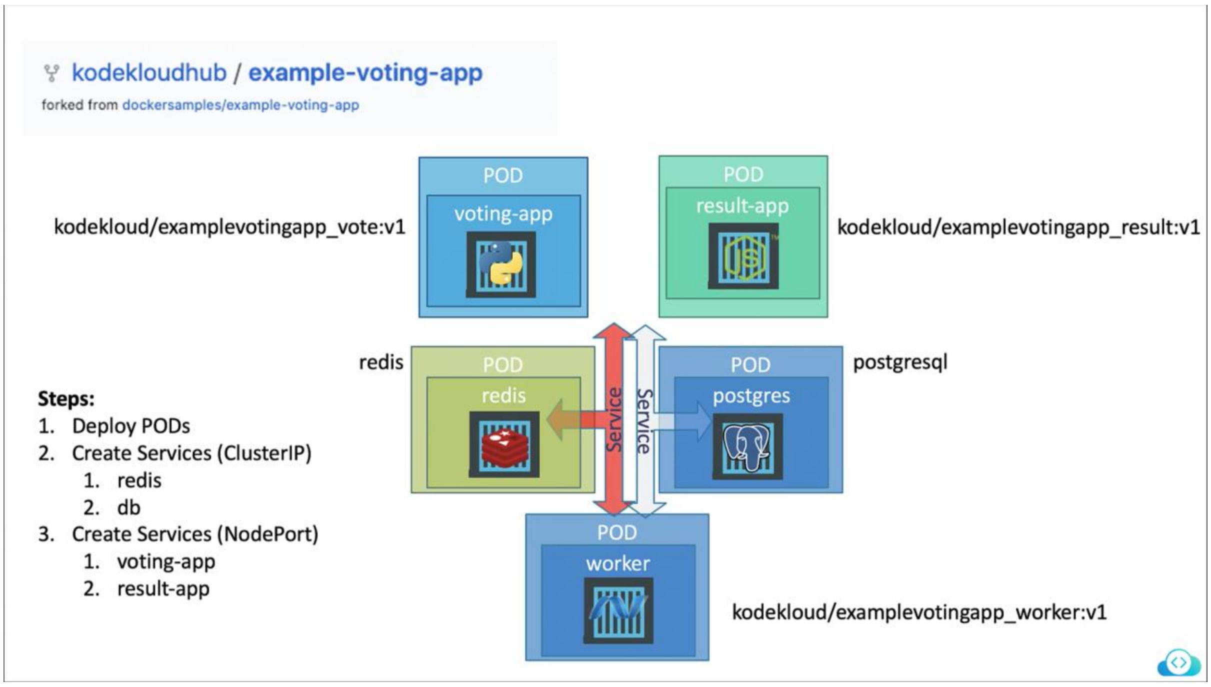
Links is cli option which can be used to link two containers together .
If 1 service is dependent on another service we can links to add dependency .



docker run --links

```
docker run -d --name=redis redis
docker run -d --name=db  postgres:9.4
docker run -d --name=vote -p 5000:80 --link redis:redis voting-app
docker run -d --name=result -p 5001:80 --link db:db  result-app
docker run -d --name=worker --link db:db --link redis:redis  worker
```

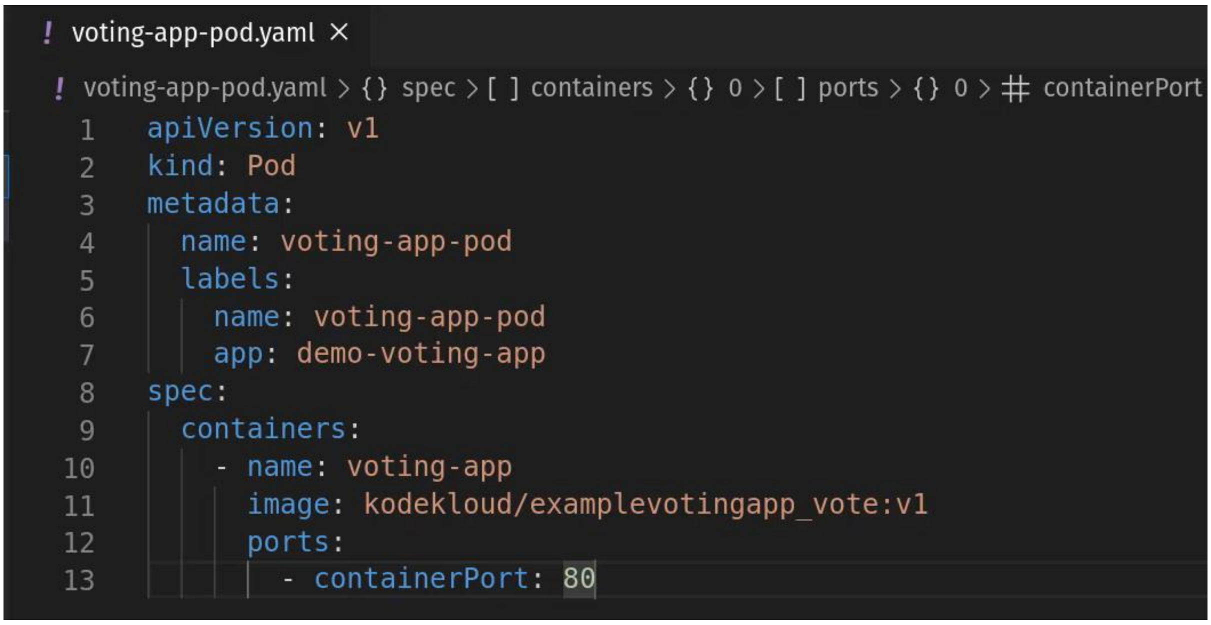
```
try {
  Jedis redis = connectToRedis("redis");
  Connection dbConn = connectToDB("db");

  System.err.println("Watching vote queue");
```



kodekloudhub / example-voting-app
forked from dockersamples/example-voting-app
POD
voting-app
kodekloud/examplevotingapp_vote:v1
POD
result-app
kodekloud/examplevotingapp_result:v1
redis
POD
redis
Service
Service
POD
postgres
postgresql
POD
worker
kodekloud/examplevotingapp_worker:v1

Steps:

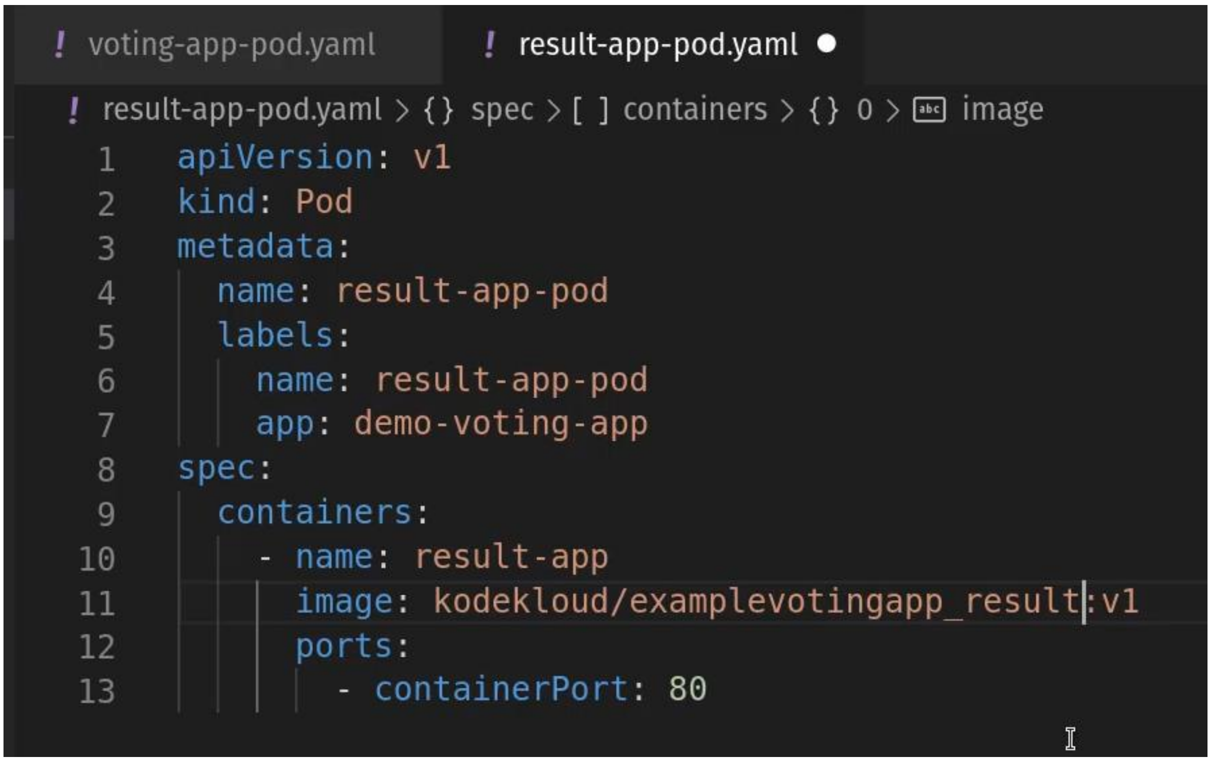
1. Deploy PODs
2. Create Services (ClusterIP)
   1. redis
   2. db
3. Create Services (NodePort)
   1. voting-app
   2. result-app



voting-app-pod.yaml

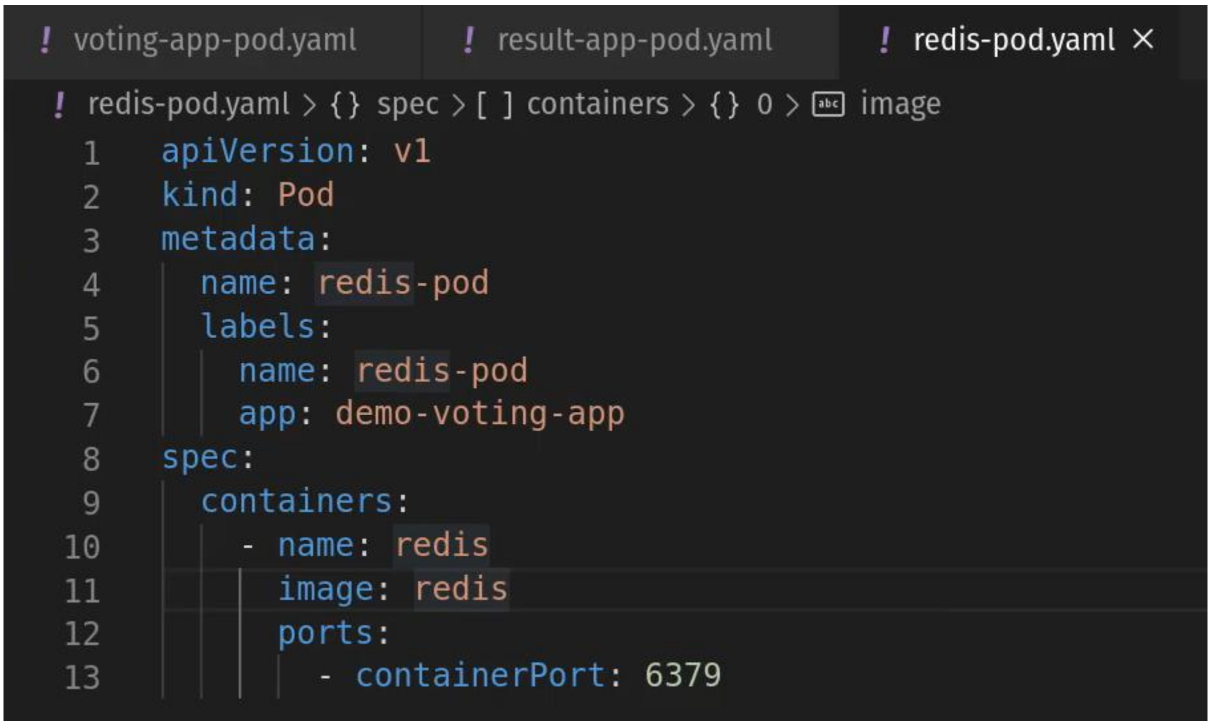
voting-app-pod.yaml > {} spec > [ ] containers > {} 0 > [ ] ports > {} 0 > # containerPort

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: voting-app-pod
  labels:
    name: voting-app-pod
    app: demo-voting-app
spec:
  containers:
    - name: voting-app
      image: kodekloud/examplevotingapp_vote:v1
      ports:
        - containerPort: 80
```


voting-app-pod.yaml | result-app-pod.yaml ●

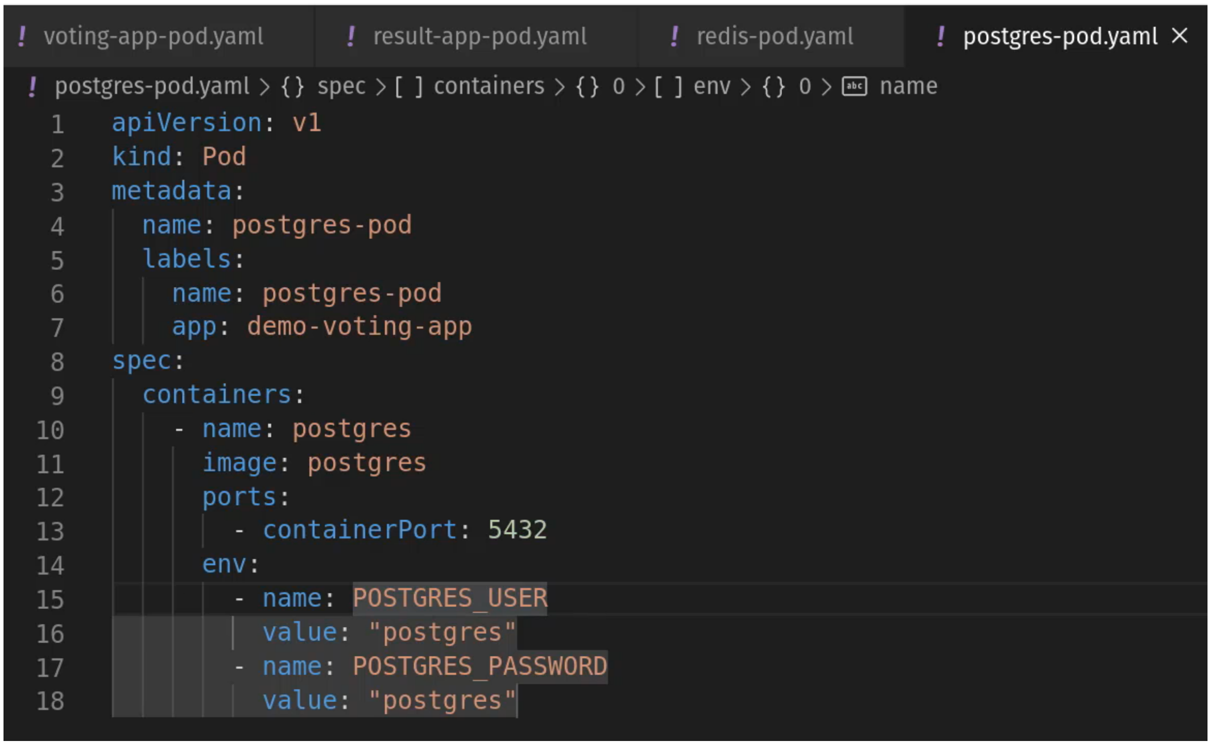
result-app-pod.yaml > {} spec > [ ] containers > {} 0 > image

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: result-app-pod
  labels:
    name: result-app-pod
    app: demo-voting-app
spec:
  containers:
    - name: result-app
      image: kodekloud/examplevotingapp_result:v1
      ports:
        - containerPort: 80
```

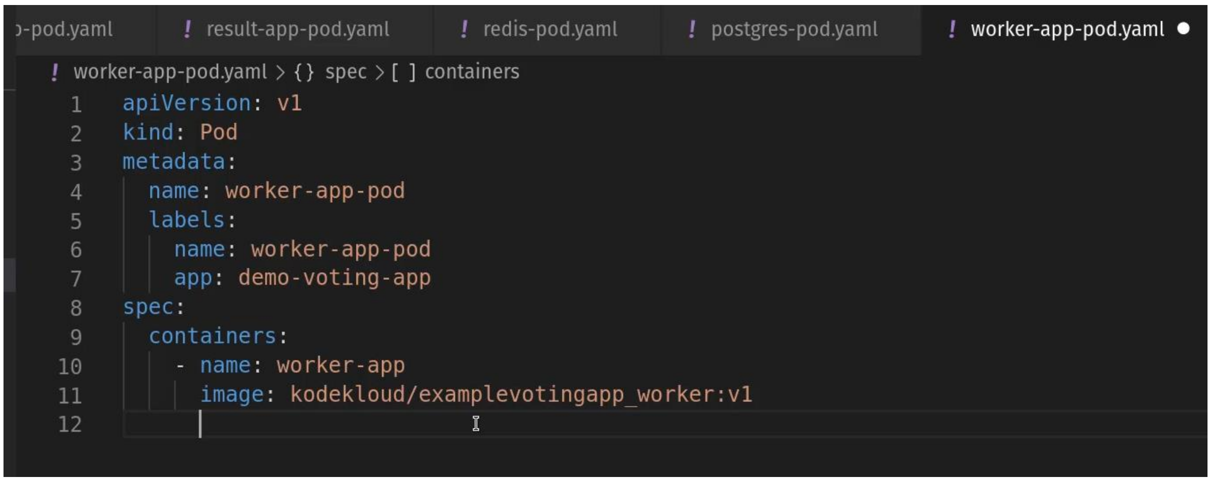

voting-app-pod.yaml | result-app-pod.yaml | redis-pod.yaml ×

redis-pod.yaml > {} spec > [ ] containers > {} 0 > image

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: redis-pod
  labels:
    name: redis-pod
    app: demo-voting-app
spec:
  containers:
    - name: redis
      image: redis
      ports:
        - containerPort: 6379
```

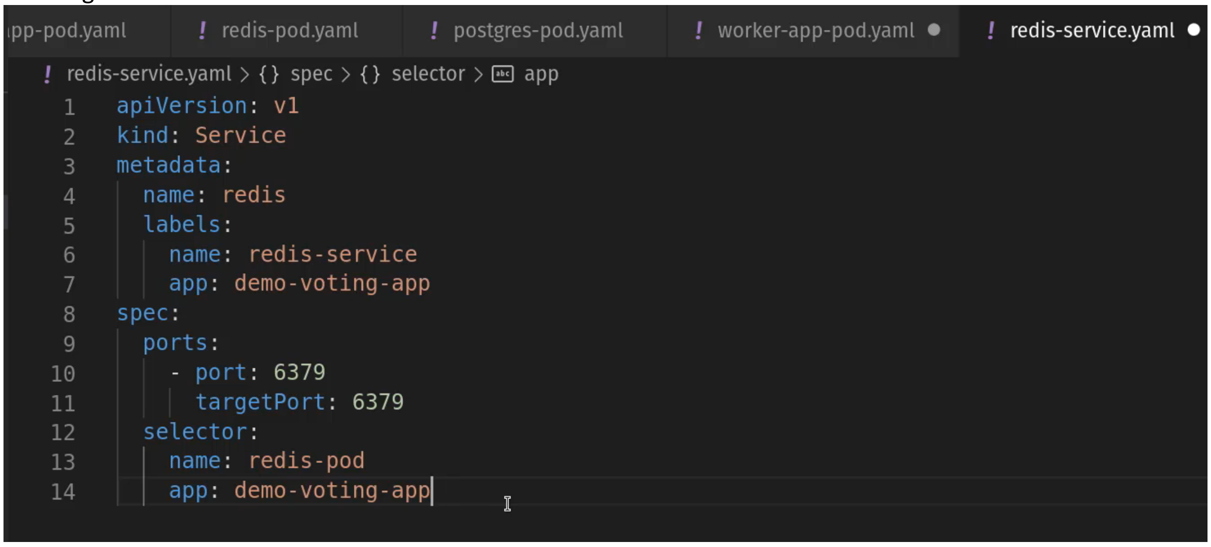


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: postgres-pod
  labels:
    name: postgres-pod
    app: demo-voting-app
spec:
  containers:
    - name: postgres
      image: postgres
      ports:
        - containerPort: 5432
      env:
        - name: POSTGRES_USER
          value: "postgres"
        - name: POSTGRES_PASSWORD
          value: "postgres"
```

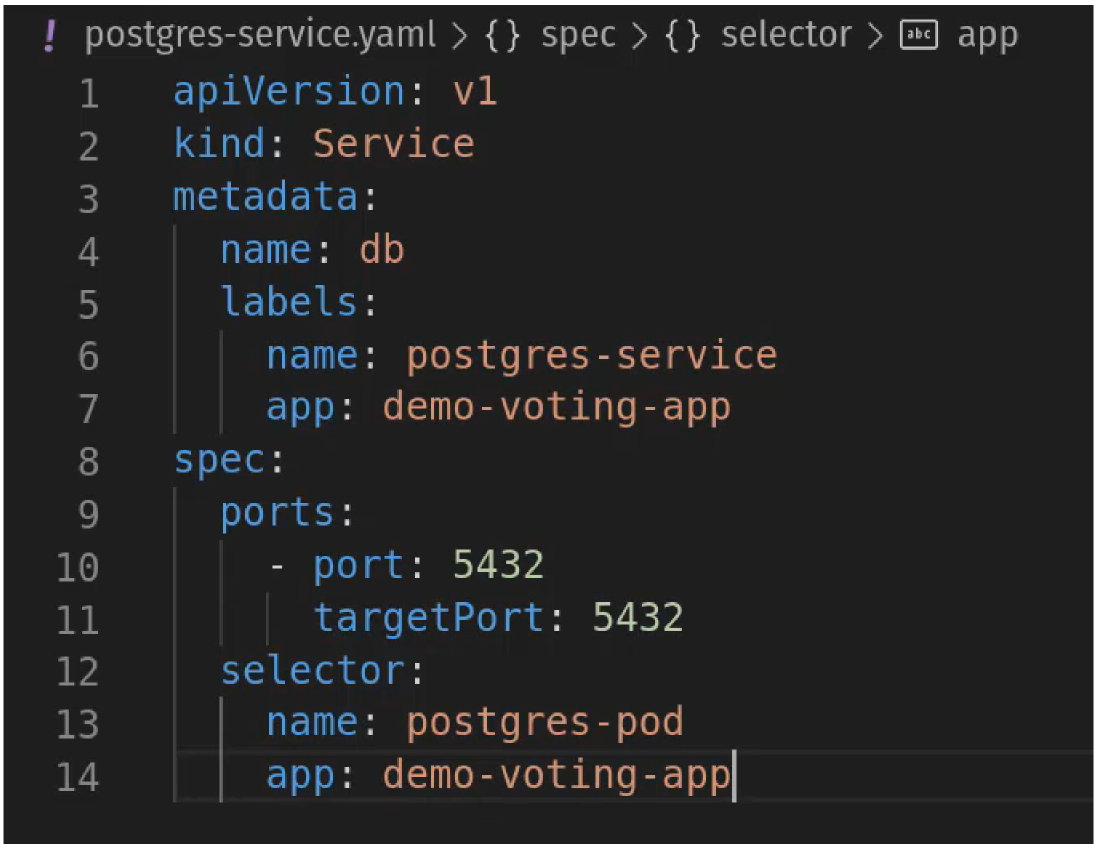


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: worker-app-pod
  labels:
    name: worker-app-pod
    app: demo-voting-app
spec:
  containers:
    - name: worker-app
      image: kodekloud/examplevotingapp_worker:v1

```

Creating services .



```yaml
apiVersion: v1
kind: Service
metadata:
  name: redis
  labels:
    name: redis-service
    app: demo-voting-app
spec:
  ports:
    - port: 6379
      targetPort: 6379
  selector:
    name: redis-pod
    app: demo-voting-app
```



```yaml
apiVersion: v1
kind: Service
metadata:
  name: db
  labels:
    name: postgres-service
    app: demo-voting-app
spec:
  ports:
    - port: 5432
      targetPort: 5432
  selector:
    name: postgres-pod
    app: demo-voting-app
```