



UNIVERSITY WITH A PURPOSE

UNIVERSITY OF PETROLEUM & ENERGY STUDIES
Dehradun

APPLICATION
CONTAINERIZATION

Name: Rakshit Kapoor

Course: B. TECH CSE DevOps (2018-22)

Roll no.: R171218082

Sapid: 500067642

Experiment-8

Launch Single Node Kubernetes Cluster

Step-1: Start Minikube

```
$ minikube start --wait=false
* minikube v1.8.1 on Ubuntu 18.04
* Using the none driver based on existing profile
* Reconfiguring existing host ...
* Using the running none "minikube" bare metal machine ...
* OS release is Ubuntu 18.04.4 LTS
* Preparing Kubernetes v1.17.3 on Docker 19.03.6 ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
* Launching Kubernetes ...
* Enabling addons: default-storageclass, storage-provisioner
* Configuring local host environment ...
* Done! kubectl is now configured to use "minikube"
$
```

Step-2: Cluster info

```
$ kubectl cluster-info
Kubernetes master is running at https://172.17.0.19:8443
KubeDNS is running at https://172.17.0.19:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
$ kubectl get nodes
NAME                STATUS    ROLES    AGE      VERSION
minikube            Ready     master   2m10s    v1.17.3
$
```

Step-3: Deploy Containers

```
$ kubectl create deployment first-deployment --image=katacoda/docker-http-server
deployment.apps/first-deployment created
$ kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
first-deployment-666c48b44-djc8t    0/1     ContainerCreating   0           3s
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
first-deployment-666c48b44-djc8t    1/1     Running   0           5m16s
$ kubectl expose deployment first-deployment --port=80 --type=NodePort
service/first-deployment exposed
$ export PORT=$(kubectl get svc first-deployment -o go-template='{{range.spec.ports}}{{if .nodePort}}{{.nodePort}}{{"\n"}}{{end}}{{end}}')
$ echo "Accessing host01:$PORT"
Accessing host01:31013
$ curl host01:$PORT
<h1>This request was processed by host: first-deployment-666c48b44-djc8t</h1>
$
```

Step-4: Dashboard

```
$ minikube addons enable dashboard
* The 'dashboard' addon is enabled
$ kubectl apply -f /opt/kubernetes-dashboard.yaml
namespace/kubernetes-dashboard configured
service/kubernetes-dashboard-katacoda created
$ kubectl get pods -n kubernetes-dashboard -w
NAME                                READY   STATUS    RESTARTS   AGE
dashboard-metrics-scraper-7b64584c5c-h95q5  1/1     Running   0          6s
kubernetes-dashboard-79d9cd965-dgr9d        1/1     Running   0          5s
$ kubectl get pods -n kubernetes-dashboard -w
```

The screenshot displays the Kubernetes Dashboard interface. The left sidebar contains navigation links for Cluster, Namespace, Workloads, and Config and Storage. The main content area shows the 'Workload Status' section with three green circular indicators for Deployments, Pods, and Replica Sets. Below this, there are three tables: Deployments, Pods, and Replica Sets, each showing a single entry for 'first-deployment'.

Cluster

- Cluster Roles
- Namespaces
- Nodes
- Persistent Volumes
- Storage Classes

Namespace

default

Overview

Workloads

- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

Discovery and Load Balancing

- Ingresses

Services

Config and Storage

- Config Maps
- Persistent Volume Claims

Workload Status

Deployments

Name	Namespace	Labels	Pods	Age	Images
first-deployment	default	app: first-deployment	1 / 1	10 minutes	katacoda/docker-http-server

Pods

Name	Namespace	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Age
first-deployment-666c48b44-qjz8t	default	app: first-deployment pod-template-hash: 666c48b44	minikube	Running	0	-	-	10 minutes

Replica Sets

Name	Namespace	Labels	Pods	Age	Images
first-deployment-666c48b44	default	app: first-deployment pod-template-hash: 666c48b44	1 / 1	10 minutes	katacoda/docker-http-server