

# Launch Single Node Kubernetes Cluster

## CKA Syllabus:

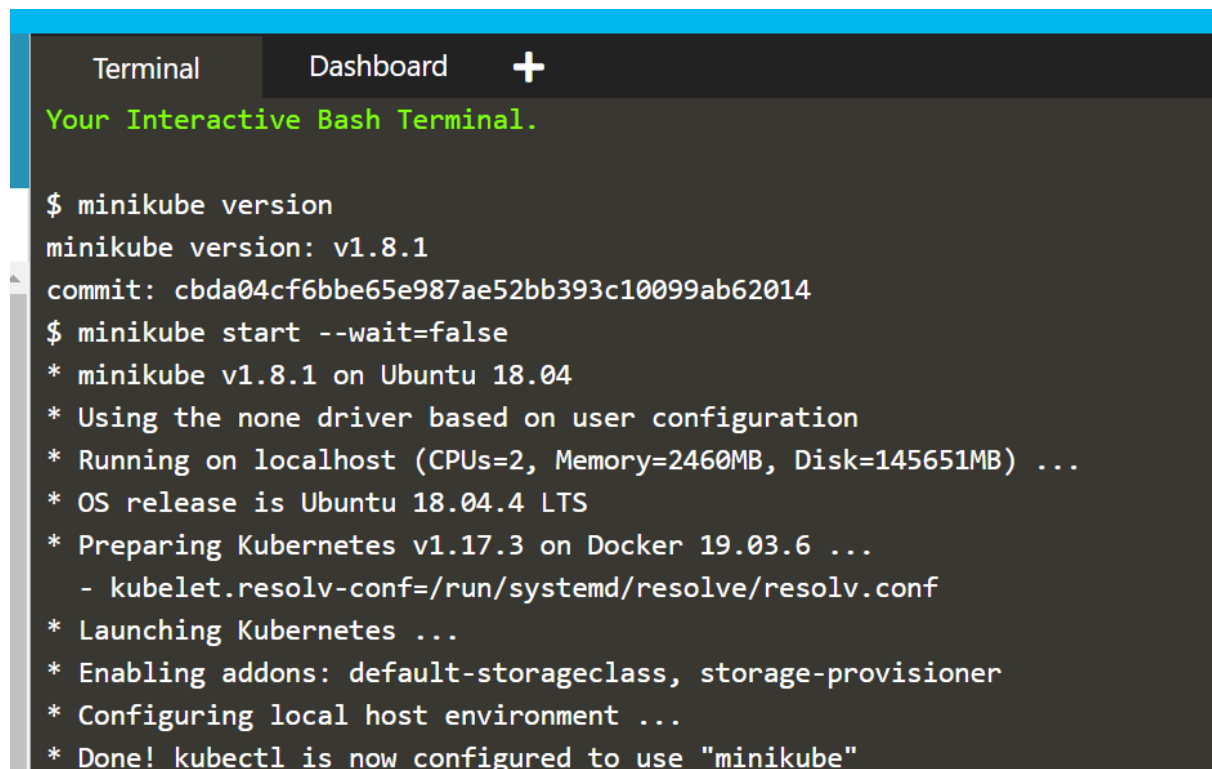
### Installation, Configuration and Validation

Goal:

Design a Kubernetes cluster

Step 1:

Start Minikube

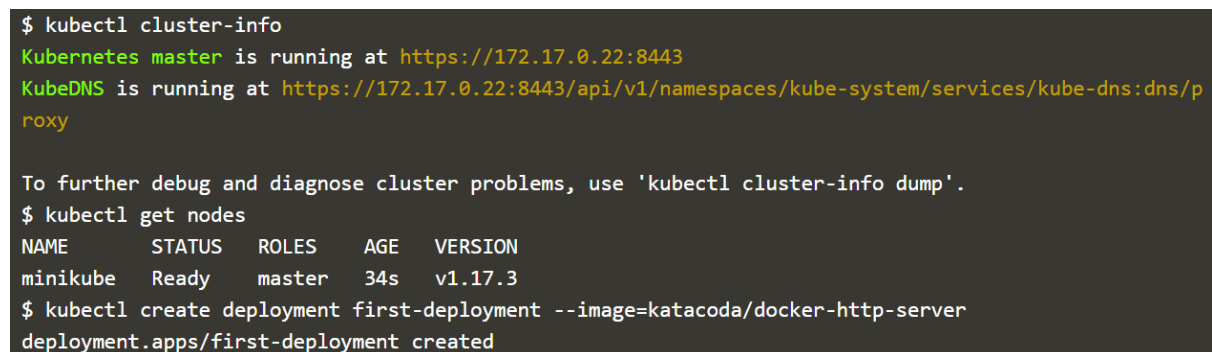


The screenshot shows a terminal window with a dark background and a light blue header bar. The header bar contains the word 'Terminal' in white, 'Dashboard' in white, and a white plus sign. Below the header bar, the text 'Your Interactive Bash Terminal.' is displayed in green. The terminal shows the following commands and output:

```
$ minikube version
minikube version: v1.8.1
commit: cbda04cf6bbe65e987ae52bb393c10099ab62014
$ minikube start --wait=false
* minikube v1.8.1 on Ubuntu 18.04
* Using the none driver based on user configuration
* Running on localhost (CPUs=2, Memory=2460MB, Disk=145651MB) ...
* 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



The screenshot shows a terminal window with a dark background. The terminal shows the following commands and output:

```
$ kubectl cluster-info
Kubernetes master is running at https://172.17.0.22:8443
KubeDNS is running at https://172.17.0.22:8443/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   34s   v1.17.3
$ kubectl create deployment first-deployment --image=katacoda/docker-http-server
deployment.apps/first-deployment created
```

Step 3:

## Deploy Containers

```
$ kubectl create deployment first-deployment --image=katacoda/docker-http-server
deployment.apps/first-deployment created
```

The command below finds the allocated port and executes a HTTP request.

```
$ export PORT=$(kubectl get svc first-deployment -o go-template='{{range.spec.ports}}{{if .nodePort}}
{{.nodePort}}{{"\n"}}{{end}}{{end}}')
$ echo "Accessing host01:$PORT"
Accessing host01:31817
$ curl host01:$PORT
<h1>This request was processed by host: first-deployment-666c48b44-l81rs</h1>
```

Step 4: Dashboard

## Overview

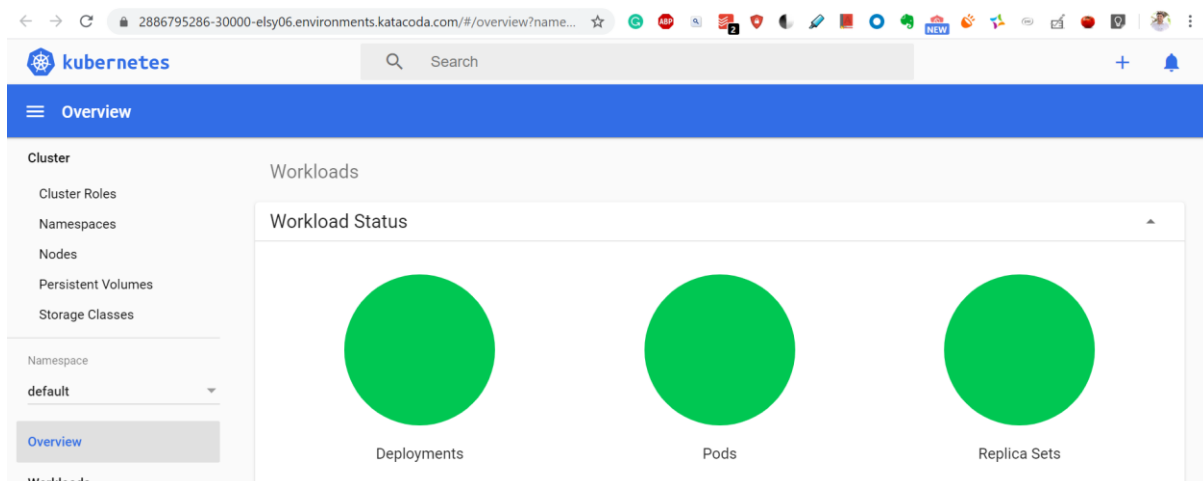


Fig: 1

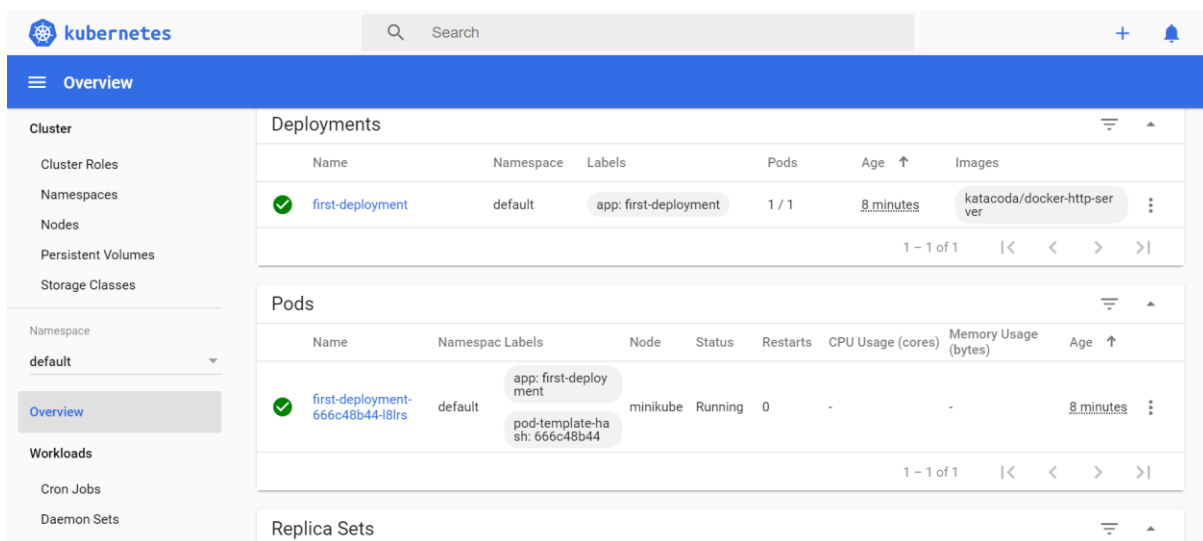


Fig: 2

The screenshot shows the Kubernetes Dashboard Overview page. The left sidebar contains navigation links for Cluster, Namespaces, Nodes, Persistent Volumes, Storage Classes, Workloads, Cron Jobs, and Daemon Sets. The main content area is divided into two sections: 'Replica Sets' and 'Discovery and Load Balancing'.

**Replica Sets Table:**

Name	Namespace	Labels	Pods	Age	Images
first-deployment-666c48b44	default	app: first-deployment pod-template-hash: 666c48b44	1 / 1	8 minutes	katacoda/docker-http-server

**Services Table:**

Name	Namespace	Labels	Cluster IP	Internal Endpoints	External Endpoints	Age
first-deployment	default	app: first-deployment	10.97.205.219	first-deployment:80 TCP first-deployment:311 TCP	-	7 minutes

Fig: 3

The screenshot shows the Kubernetes Dashboard Overview page. The left sidebar contains navigation links for Cluster, Namespaces, Nodes, Persistent Volumes, Storage Classes, Workloads, Cron Jobs, Daemon Sets, Deployments, and Jobs. The main content area is divided into two sections: 'Services' and 'Config and Storage'.

**Services Table:**

Name	Namespace	Labels	Cluster IP	Internal Endpoints	External Endpoints	Age
first-deployment	default	app: first-deployment	10.97.205.219	first-deployment:80 TCP first-deployment:311 TCP	-	8 minutes
kubernetes	default	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	9 minutes

**Secrets Table:**

Name	Namespace	Labels	Type	Age
default-token-zt6z7	default	-	kubernetes.io/service-account-token	9 minutes

Fig: 4

Pods:

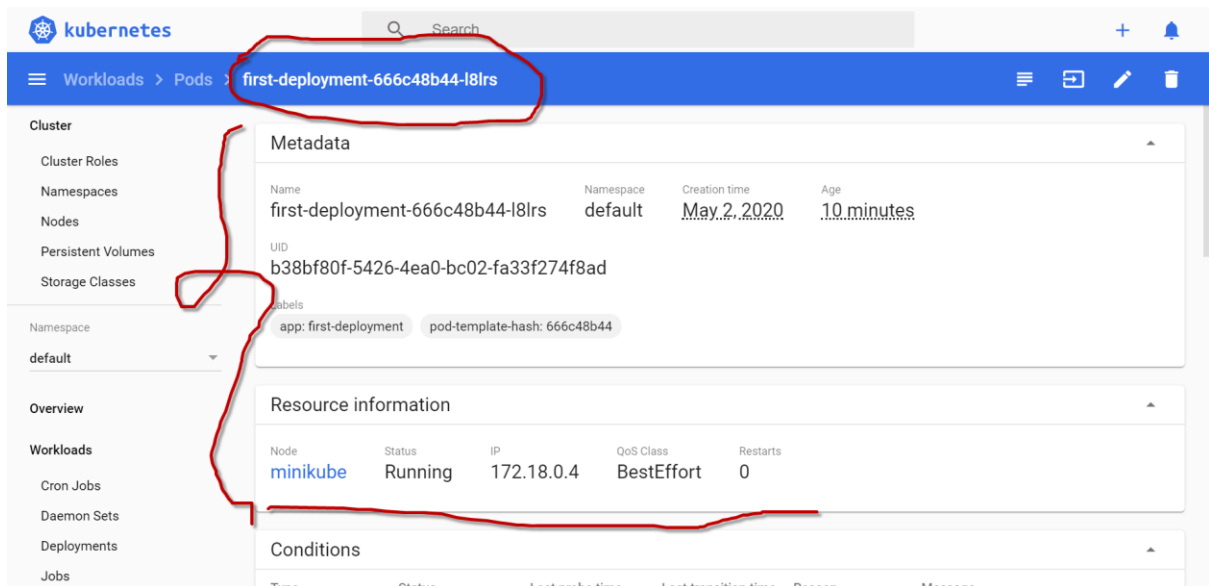


Fig: 5

Services:

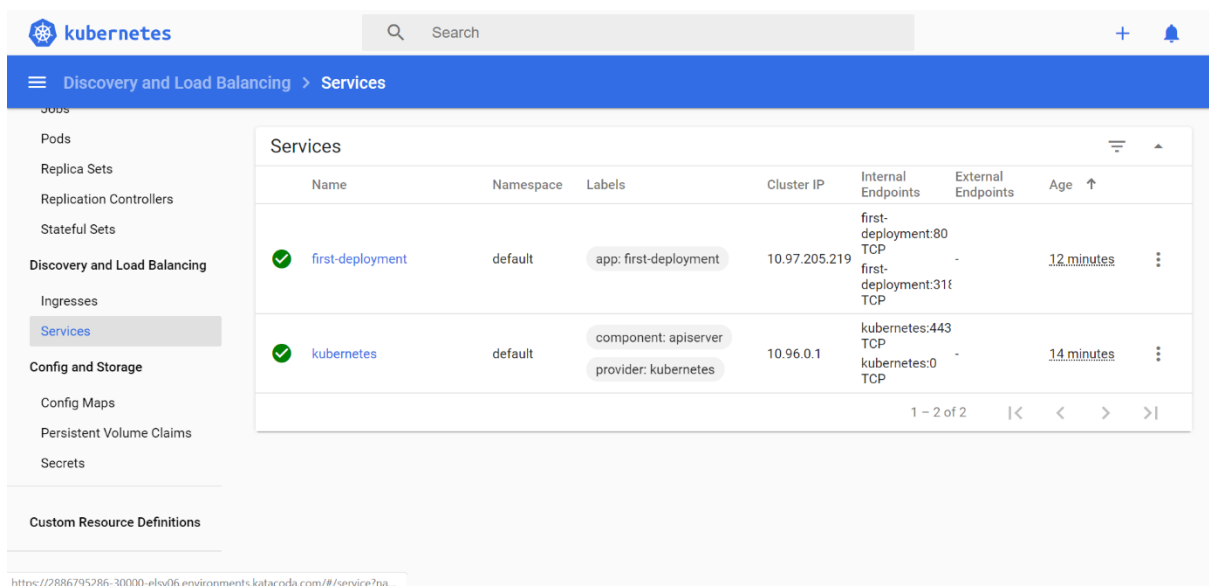


Fig: 6

Learning objectives:

**This scenario has explained how to launch a Single Node Kubernetes cluster. Using Minikube is ideal for development environments and testing Kubernetes locally.**

References:

<https://www.katacoda.com/courses/kubernetes/launch-single-node-cluster>

<https://kubernetes.io/docs/setup/learning-environment/minikube/>