FULL STACK

Kubernetes

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    cout << "Hello, world!" << endl;
    return 0;
}
```

A Day in the Life of a Full Stack Developer

All the desired results were achieved in the last sprint, and the sprint was marked as completed. Joe has been appreciated for the last project.

A new project which needs to be completed in a week. He needs to develop a project to schedule multiple pods and create multiple containers. He then has to deploy and add a Linux node to the Kubernetes cluster.

In this lesson, we will learn how to solve this real-world scenario and help Joe effectively complete his task.

Learning Objectives

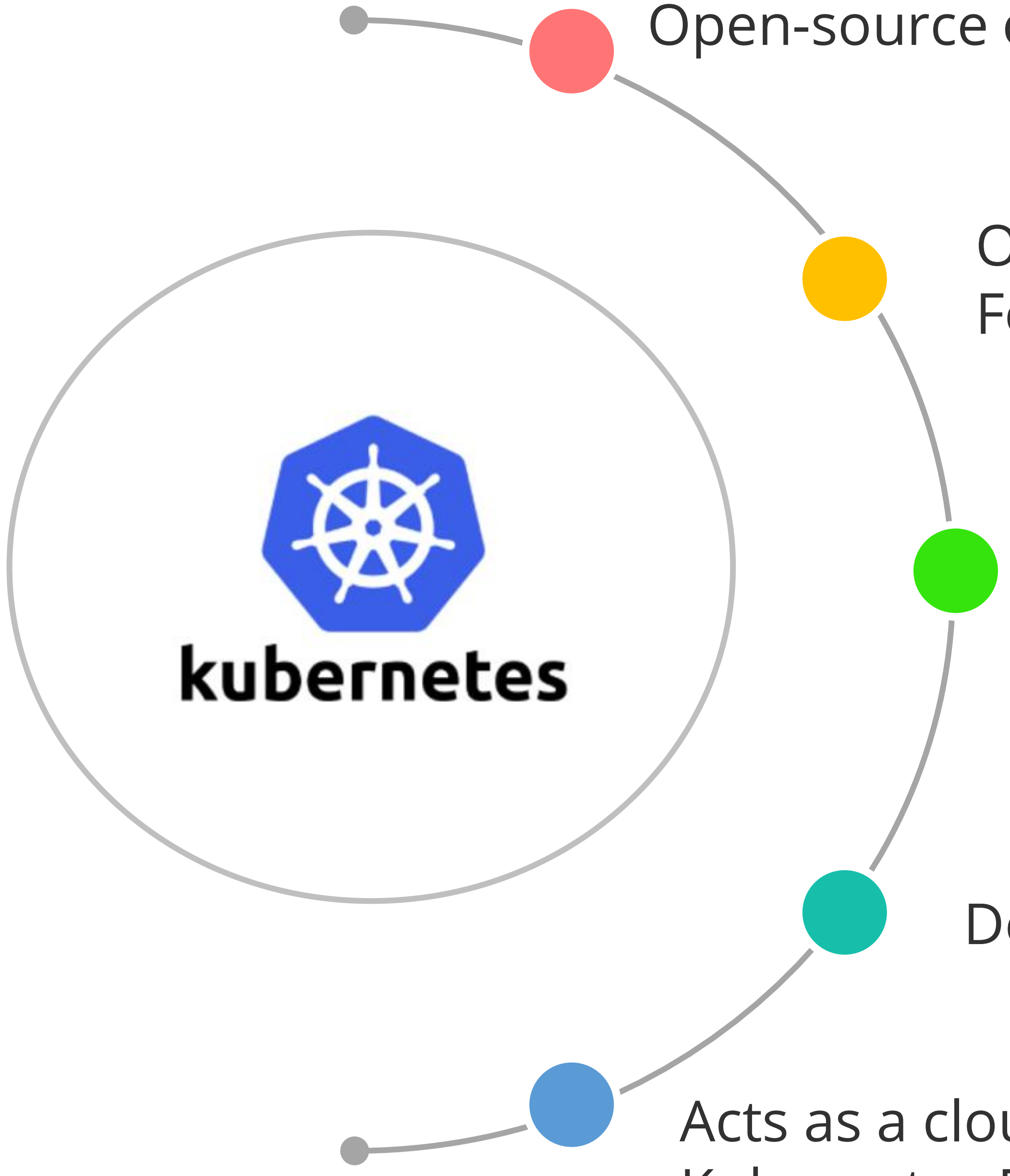
By the end of this lesson, you will be able to:

- List the importance of AWS S3 and Kubernetes
- Create an S3 bucket to host a web page
- Demonstrate the use of Kubernetes
- Deploy Spring application in AWS

FULL STACK

Kubernetes

Kubernetes



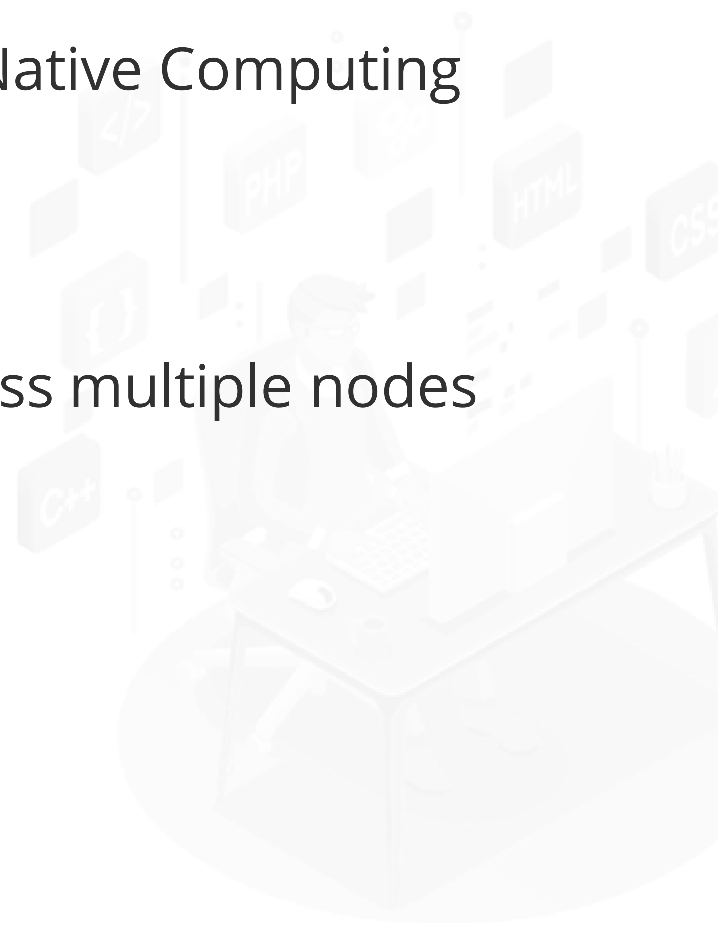
Open-source orchestration tool used to manage containerized applications

Originally designed by Google and acquired by Cloud Native Computing Foundation

Automates deployment and scales containers across multiple nodes in Kubernetes clusters

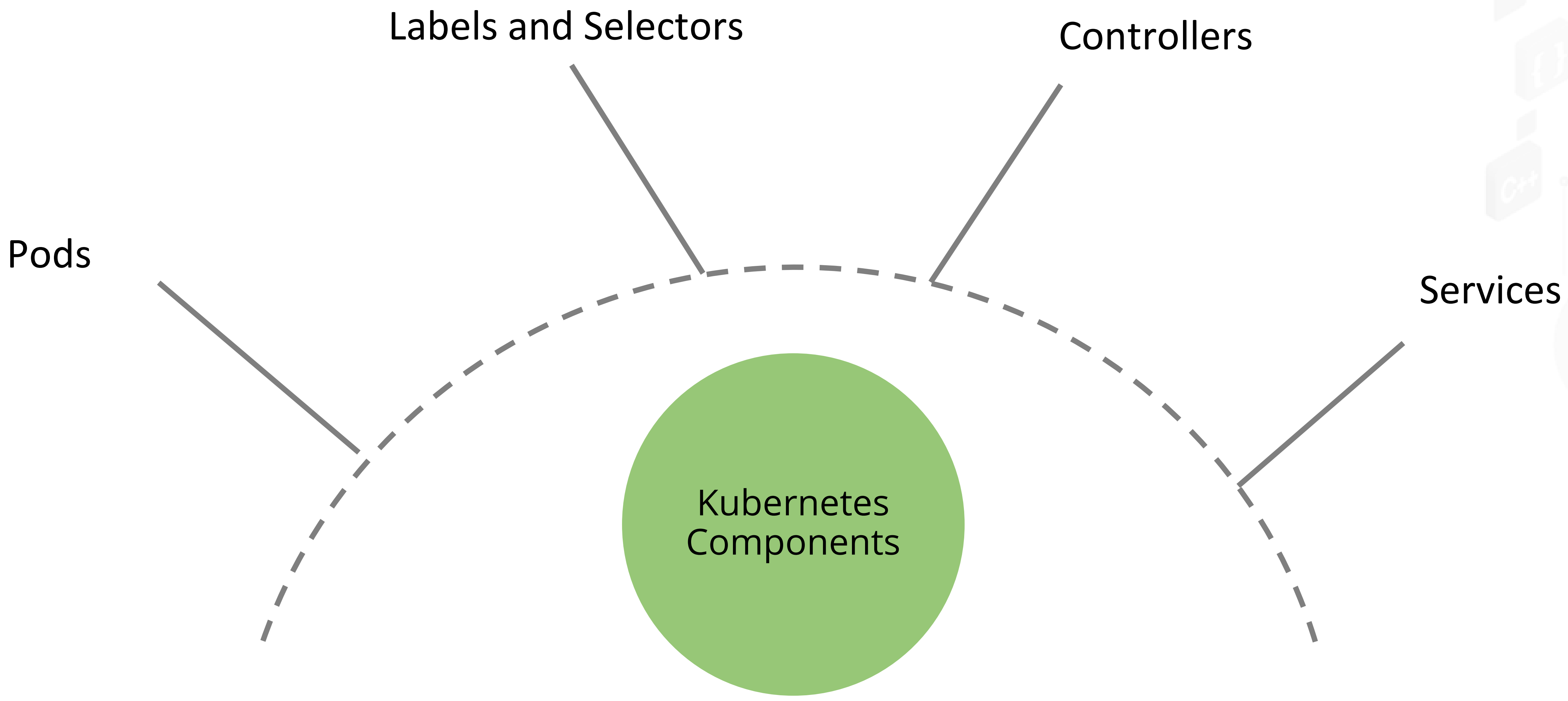
Designed in GO language

Acts as a cloud service in major cloud providers, such as EKS in AWS and Kubernetes Engine in GCP



Kubernetes Components

Kubernetes is a combination of various building blocks which collectively help to manage, deploy, and scale containerized applications.



Pods

- Pod is a basic scheduling unit in Kubernetes
- Each pod comprises one or more containers that can be initialized on any host
- Each pod is assigned a unique IP using which we can redirect traffic from outside to the pod
- Pods are managed using the kubelet command line in a Kubernetes cluster
- Containers in a pod can consist of multiple applications
- Pod templates are used to define how pods will be created and deployed
- Pods share physical resources from host machines in forms of CPU, RAM, and storage

Labels and Selectors

- Kubernetes attaches key-value pairs called labels for various objects such as services, pods, and nodes
- These labels can be used to locate a specific resource
- The same label can be used for multiple objects, so you should define and create unique labels for Kubernetes objects

Controllers

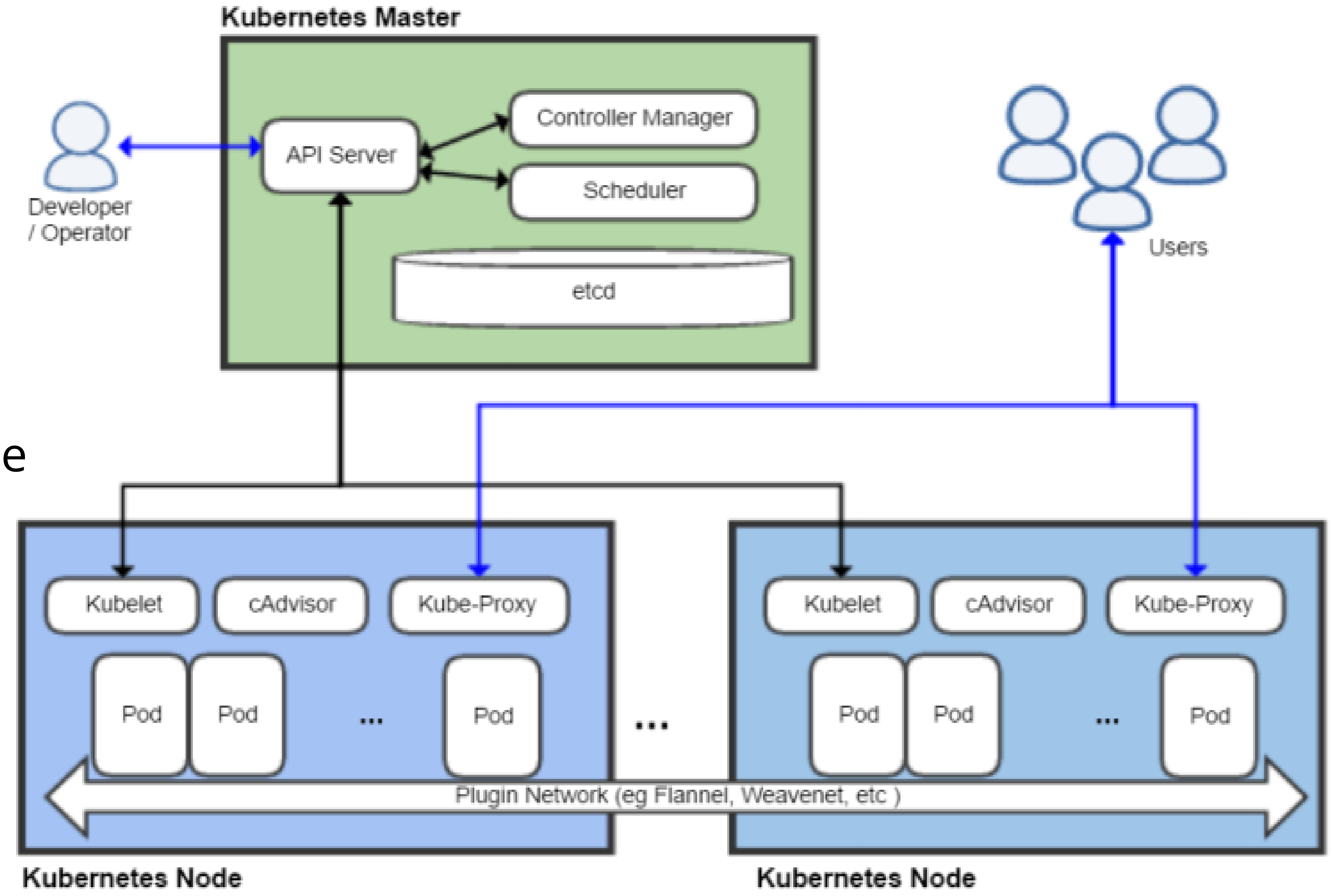
- Controllers bring pods to a specific state
- ReplicationController replicates and scales pods across Kubernetes clusters
- Controllers take care of the availability of pods, and if it fails, a replacement pod gets created automatically
- DaemonSet controller ensures only one pod runs on each node
- Job controller manages all the batch jobs of pods which are executed in a Kubernetes cluster
- Controllers manage pods using labels and selectors to identify resources

Services

- The collection of pods are bundled together in a service.
- Kubernetes allocates a unique port and DNS to each service. The port and DNS details are changed only if the service object is recreated.
- There can be multiple replicated pods in a service.
- In the case of multiple pods, an in-built load balancer is used to share the load between pods running on different nodes.
- A service implements high availability and load share for containerized applications.
- If a pod is terminated, a replacement pod will be initialized automatically.

Kubernetes Architecture

- Uses master-slave architecture
- Kubernetes master contains the following architecture components:
  - etcd
  - API server
  - Scheduler
  - Controller manager
- Kubernetes client contains the following architecture components:
  - Kubelet
  - cAdvisor
  - Pod
  - kube-proxy



Kubernetes Master Components

etcd

- etcd is a persistent, lightweight, and key-value data store.
- It stores the complete configuration data of a Kubernetes cluster. At any point in time, you can check the state of a cluster with the available data. This data store can be shared with other components.
- It provides a data layer in Kubernetes clusters.

API server

- API server supports Kubernetes API and processes all the requests from various components.
- It handles the REST requests and JSON requests and updates the state of each object in etcd.

Kubernetes Master Components

Scheduler

- Scheduler is the component of Kubernetes responsible for managing workloads in a cluster.
- It identifies the unutilized node and the process to schedule pods on unutilized nodes based on the requirements.
- It helps to manage all Kubernetes resources effectively.

Controller manager

- Controller manager manages all controllers in Kubernetes such as DaemonSet and ReplicationController.
- It interacts with the API server to create, edit, and delete any resources being managed.

Kubernetes Node Components

kube-proxy

- kube-proxy implements network proxy and acts as a load balancer in Kubernetes cluster.
- It helps to redirect traffic to a specific container in a pod based on the incoming port and IP details.

cAdvisor

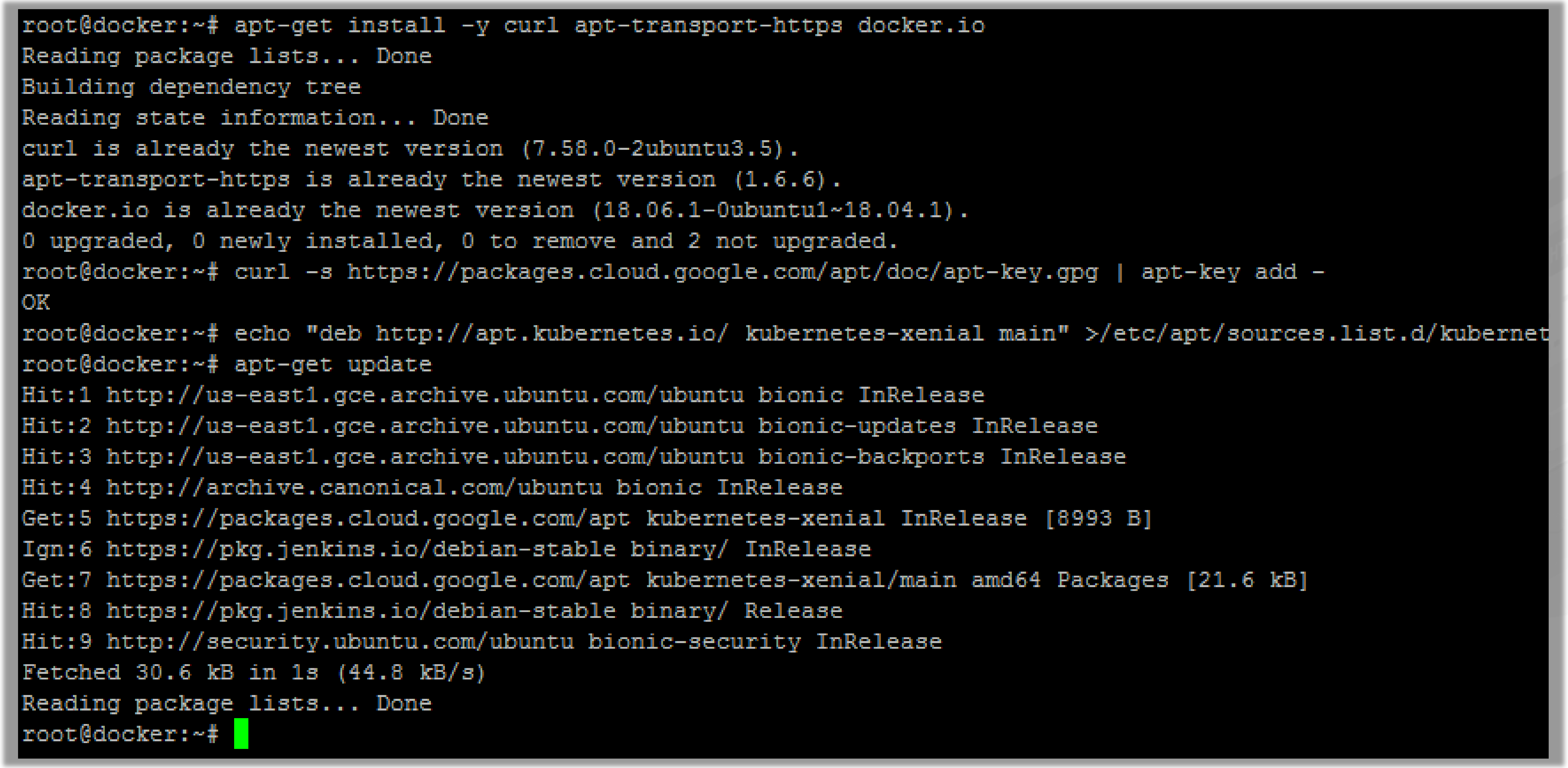
- cAdvisor is an agent that monitors and gathers resource usage and performance metrics such as CPU, memory, files, and network usage of containers on each node.

Kubernetes Node Components

Kubelet

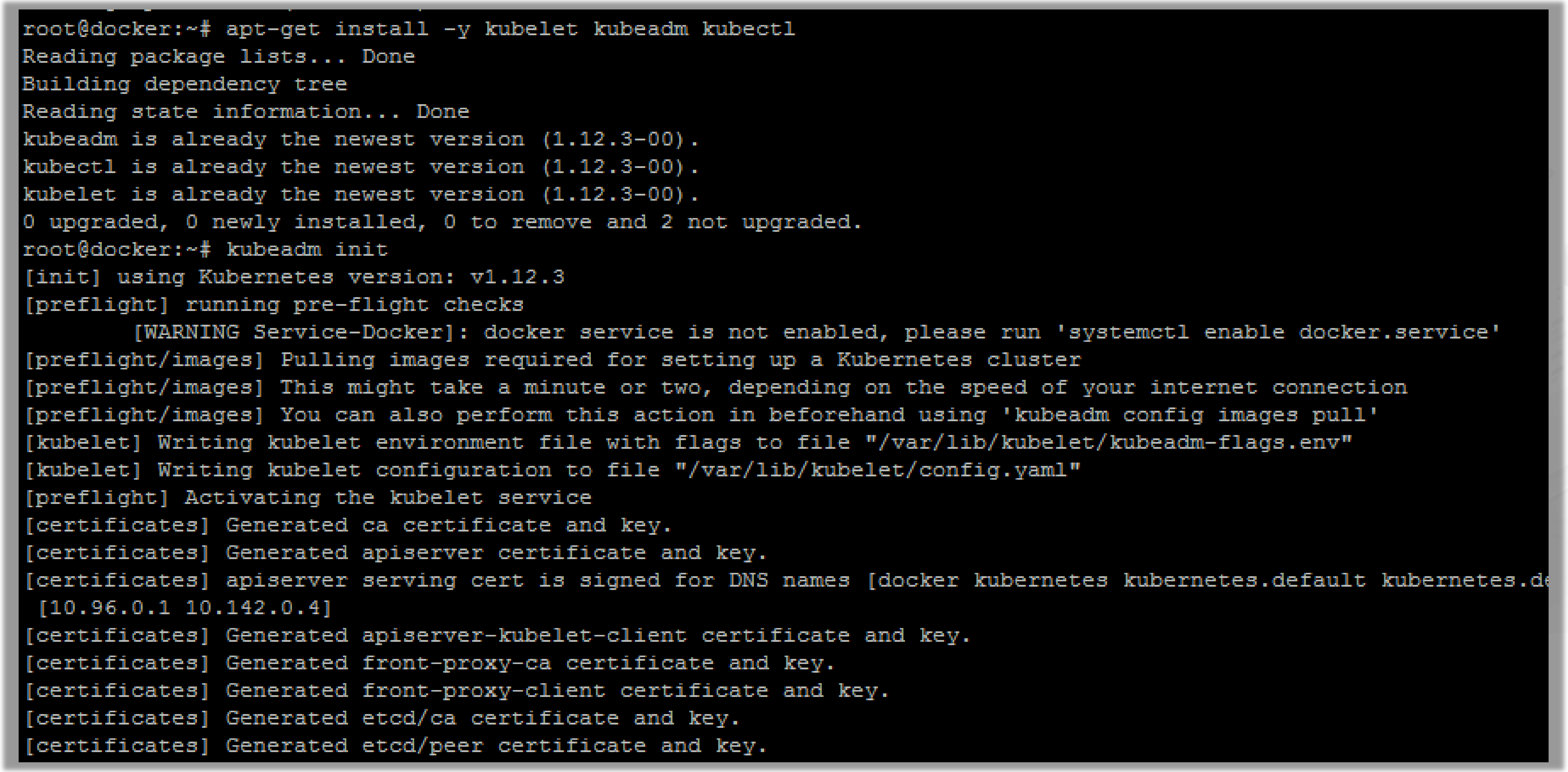
- Kubelet is responsible for the working of each node and ensuring the container’s health. It monitors how the pods start, stop, and are maintained.
- Once the master detects a node failure, the ReplicationController observes the change in state and launches pods on other healthy nodes.

Kubernetes Installation



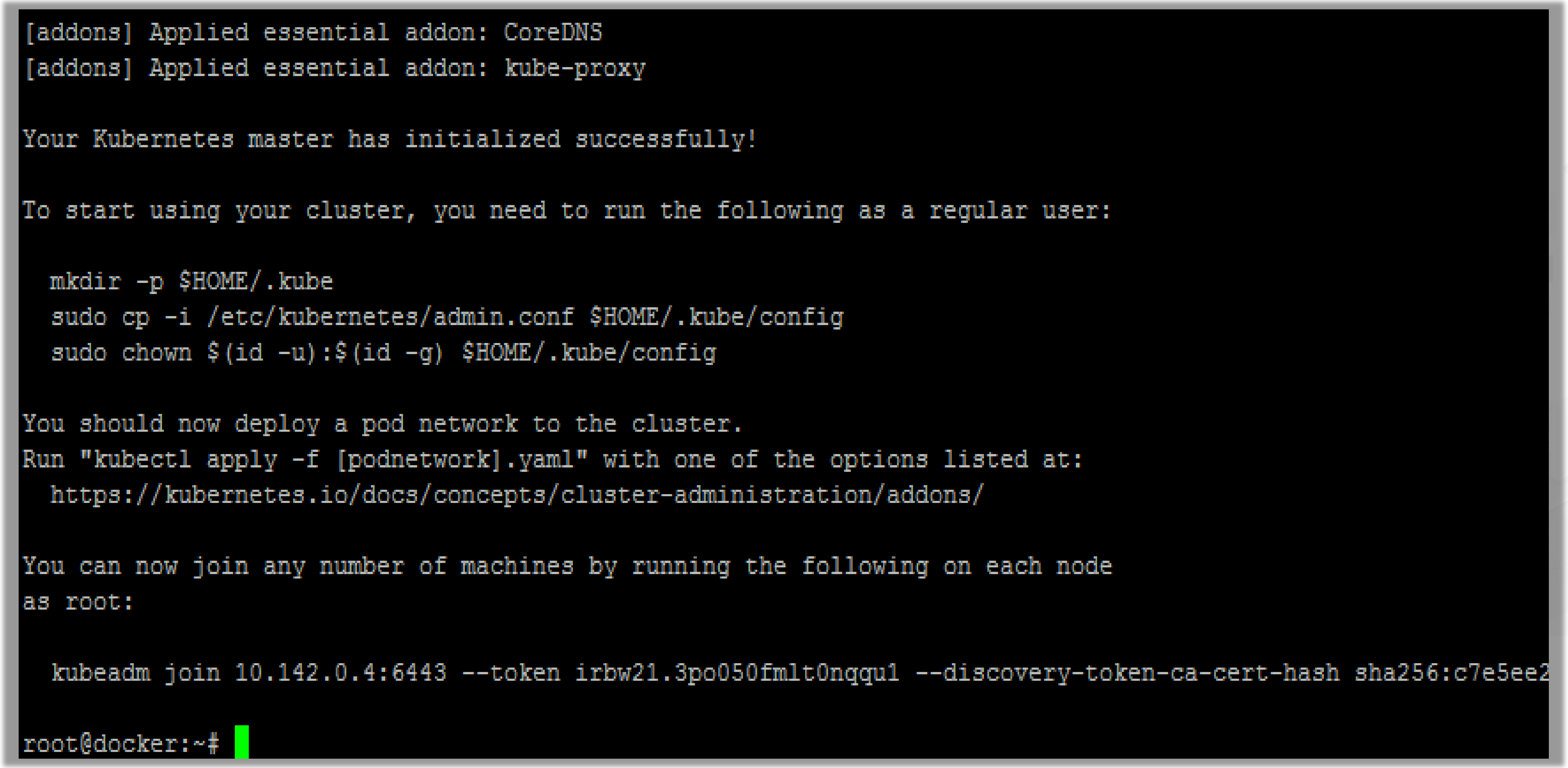
```
root@docker:~# apt-get install -y curl apt-transport-https docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
curl is already the newest version (7.58.0-2ubuntu3.5).
apt-transport-https is already the newest version (1.6.6).
docker.io is already the newest version (18.06.1-0ubuntu1~18.04.1).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
root@docker:~# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
OK
root@docker:~# echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" >/etc/apt/sources.list.d/kubernet
root@docker:~# apt-get update
Hit:1 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://archive.canonical.com/ubuntu bionic InRelease
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [8993 B]
Ign:6 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:7 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packages [21.6 kB]
Hit:8 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:9 http://security.ubuntu.com/ubuntu bionic-security InRelease
Fetched 30.6 kB in 1s (44.8 kB/s)
Reading package lists... Done
root@docker:~#
```

Kubernetes Installation



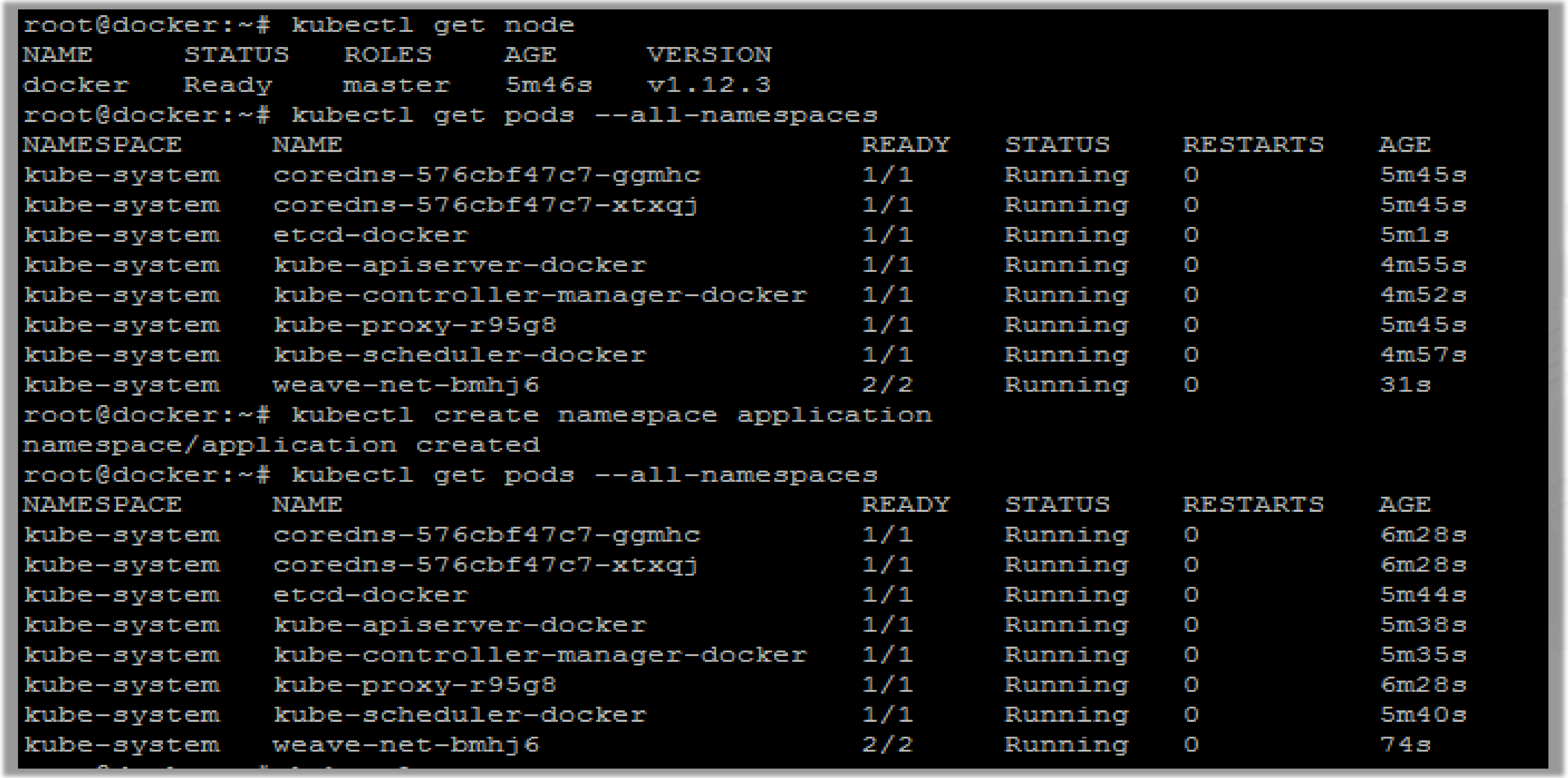
```
root@docker:~# apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree
Reading state information... Done
kubeadm is already the newest version (1.12.3-00).
kubectl is already the newest version (1.12.3-00).
kubelet is already the newest version (1.12.3-00).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
root@docker:~# kubeadm init
[init] using Kubernetes version: v1.12.3
[preflight] running pre-flight checks
        [WARNING Service-Docker]: docker service is not enabled, please run 'systemctl enable docker.service'
[preflight/images] Pulling images required for setting up a Kubernetes cluster
[preflight/images] This might take a minute or two, depending on the speed of your internet connection
[preflight/images] You can also perform this action in beforehand using 'kubeadm config images pull'
[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[preflight] Activating the kubelet service
[certificates] Generated ca certificate and key.
[certificates] Generated apiserver certificate and key.
[certificates] apiserver serving cert is signed for DNS names [docker kubernetes kubernetes.default kubernetes.de
 [10.96.0.1 10.142.0.4]
[certificates] Generated apiserver-kubelet-client certificate and key.
[certificates] Generated front-proxy-ca certificate and key.
[certificates] Generated front-proxy-client certificate and key.
[certificates] Generated etcd/ca certificate and key.
[certificates] Generated etcd/peer certificate and key.
```

Kubernetes Installation



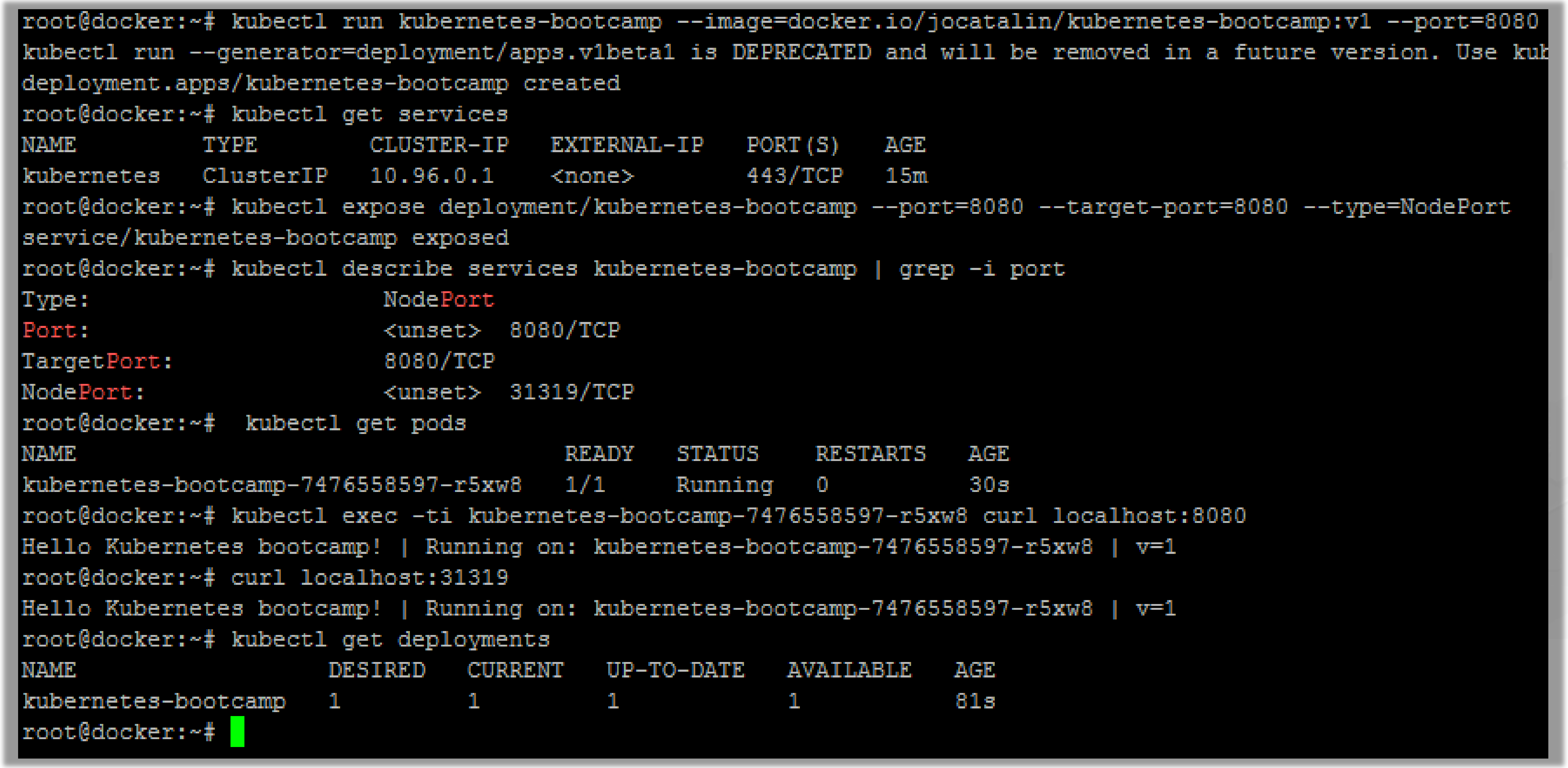
```
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join 10.142.0.4:6443 --token irbw21.3po050fmlt0nqqu1 --discovery-token-ca-cert-hash sha256:c7e5ee2

root@docker:~#
```

Kubernetes Installation



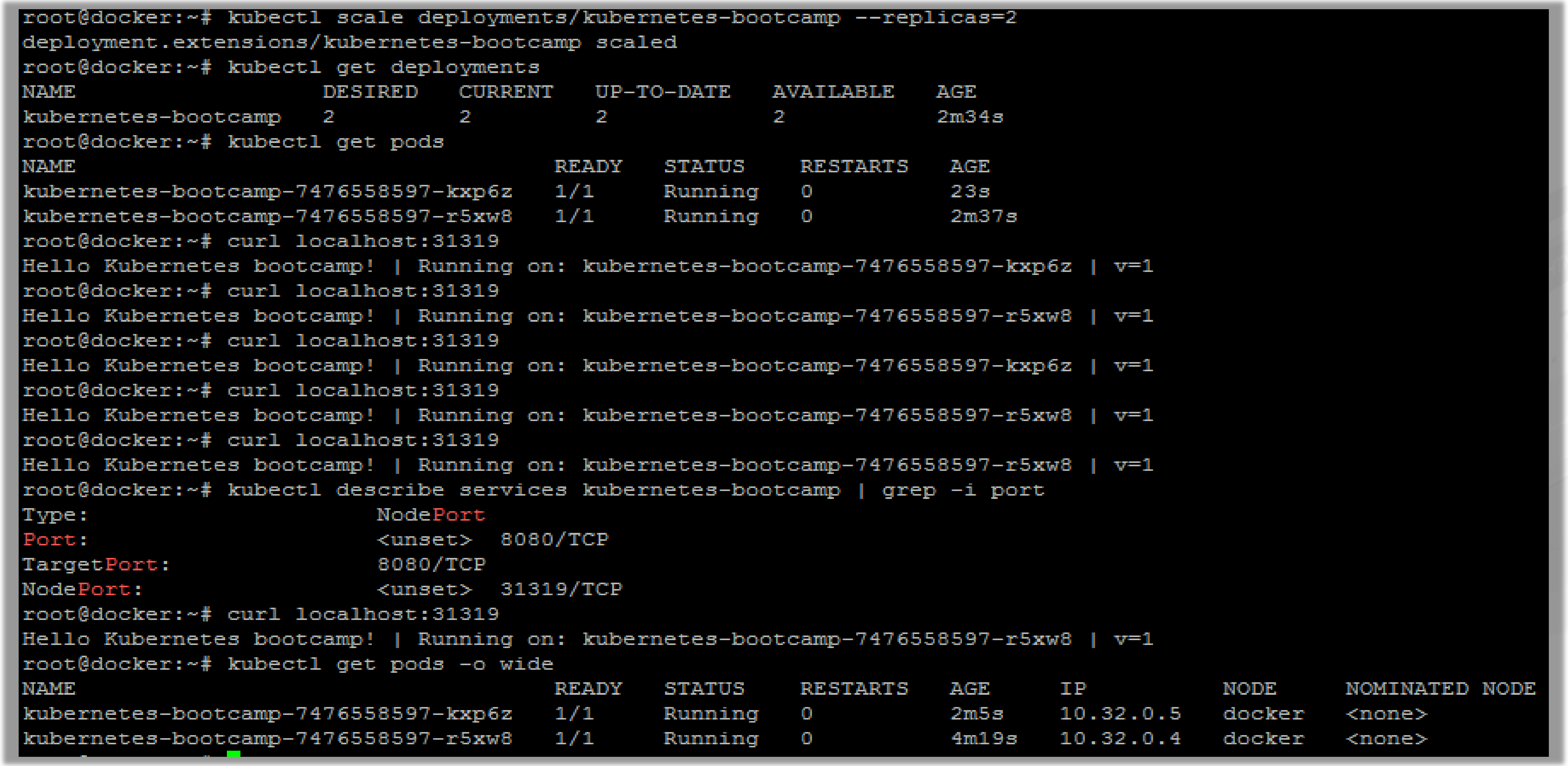
```
root@docker:~# kubectl get node
NAME      STATUS    ROLES     AGE       VERSION
docker    Ready     master    5m46s     v1.12.3
root@docker:~# kubectl get pods --all-namespaces
NAMESPACE     NAME                                 READY   STATUS     RESTARTS   AGE
kube-system   coredns-576cbf47c7-ggmhc             1/1     Running    0          5m45s
kube-system   coredns-576cbf47c7-xtxqj             1/1     Running    0          5m45s
kube-system   etcd-docker                          1/1     Running    0          5m1s
kube-system   kube-apiserver-docker                1/1     Running    0          4m55s
kube-system   kube-controller-manager-docker       1/1     Running    0          4m52s
kube-system   kube-proxy-r95g8                     1/1     Running    0          5m45s
kube-system   kube-scheduler-docker                1/1     Running    0          4m57s
kube-system   weave-net-bmhj6                      2/2     Running    0          31s
root@docker:~# kubectl create namespace application
namespace/application created
root@docker:~# kubectl get pods --all-namespaces
NAMESPACE     NAME                                 READY   STATUS     RESTARTS   AGE
kube-system   coredns-576cbf47c7-ggmhc             1/1     Running    0          6m28s
kube-system   coredns-576cbf47c7-xtxqj             1/1     Running    0          6m28s
kube-system   etcd-docker                          1/1     Running    0          5m44s
kube-system   kube-apiserver-docker                1/1     Running    0          5m38s
kube-system   kube-controller-manager-docker       1/1     Running    0          5m35s
kube-system   kube-proxy-r95g8                     1/1     Running    0          6m28s
kube-system   kube-scheduler-docker                1/1     Running    0          5m40s
kube-system   weave-net-bmhj6                      2/2     Running    0          74s
```

Kubernetes Installation



```
root@docker:~# kubectl run kubernetes-bootcamp --image=docker.io/jocatalin/kubernetes-bootcamp:v1 --port=8080
kubectl run --generator=deployment/apps.v1beta1 is DEPRECATED and will be removed in a future version. Use kub
deployment.apps/kubernetes-bootcamp created
root@docker:~# kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   15m
root@docker:~# kubectl expose deployment/kubernetes-bootcamp --port=8080 --target-port=8080 --type=NodePort
service/kubernetes-bootcamp exposed
root@docker:~# kubectl describe services kubernetes-bootcamp | grep -i port
Type:                     NodePort
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  31319/TCP
root@docker:~#  kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-7476558597-r5xw8   1/1     Running   0          30s
root@docker:~# kubectl exec -ti kubernetes-bootcamp-7476558597-r5xw8 curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-7476558597-r5xw8 | v=1
root@docker:~# curl localhost:31319
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-7476558597-r5xw8 | v=1
root@docker:~# kubectl get deployments
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1         1         1            1           81s
root@docker:~#
```

Kubernetes Installation



```
root@docker:~# kubectl scale deployments/kubernetes-bootcamp --replicas=2
deployment.extensions/kubernetes-bootcamp scaled
root@docker:~# kubectl get deployments
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   2         2         2            2           2m34s
root@docker:~# kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-7476558597-kxp6z   1/1     Running   0          23s
kubernetes-bootcamp-7476558597-r5xw8   1/1     Running   0          2m37s
root@docker:~# curl localhost:31319
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-7476558597-kxp6z | v=1
root@docker:~# curl localhost:31319
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-7476558597-r5xw8 | v=1
root@docker:~# curl localhost:31319
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-7476558597-kxp6z | v=1
root@docker:~# curl localhost:31319
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-7476558597-r5xw8 | v=1
root@docker:~# curl localhost:31319
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-7476558597-r5xw8 | v=1
root@docker:~# kubectl describe services kubernetes-bootcamp | grep -i port
Type:                     NodePort
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  31319/TCP
root@docker:~# curl localhost:31319
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-7476558597-r5xw8 | v=1
root@docker:~# kubectl get pods -o wide
NAME                                   READY   STATUS    RESTARTS   AGE     IP          NODE     NOMINATED NODE
kubernetes-bootcamp-7476558597-kxp6z   1/1     Running   0          2m5s    10.32.0.5   docker   <none>
kubernetes-bootcamp-7476558597-r5xw8   1/1     Running   0          4m19s   10.32.0.4   docker   <none>
```

Install Kubernetes

Duration: 30 min.

Problem Statement:
You are given a project to install and set up Kubernetes.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to install kubernetes:

1. Install prerequisite packages.

2. Install and configure Kubernetes.

Install Kubernetes on Cloud

Duration: 20 min.

Problem Statement:
You are given a project to install and set up Kubernetes on AWS cloud.

Assisted Practice: Guidelines

Steps to install kubernetes on cloud:

1. Login to your AWS Lab.
2. Create an AWS EKS cluster.
3. Set up **kubectl** command line with AWS EKS.

Web Hosting

Duration: 15 min.

Problem Statement:
You are given a project to demonstrate the use of AWS to explain web hosting.

Assisted Practice: Guidelines

Steps to perform web hosting:

1. Create a custom docker image.
2. Deploy a Spring Boot application to AWS EKS.

Deploy Your application

Duration: 20 min.

Problem Statement:
You are given a project to demonstrate the use of AWS to host a Spring application on it.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to deploy your application:

1. Set up EKS CTL command line and dependencies.
2. Create an EKS cluster using eksctl command line.
3. DEploy qan application to AWS EKS cluster.

Key Takeaways

- Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance.
- We can create up to 100 buckets in every AWS account. If you need to add more buckets, you can increase your account bucket limit to a maximum of 1,000 buckets by submitting a service limit increase.
- Kubernetes is an open-source container management tool that provides a platform for automating deployment, scaling, and operations of application containers across clusters of hosts.

Deploying ELK Stack on Docker Container

Duration: 30 min.

Problem Statement:

Your manager has asked you to create an elegant user interface for data analysis and data visualization as you have worked on ELK stack previously and have the idea of its working. This will help the DevOps team to monitor and analyze the application behavior.

LESSON-END PROJECT

Before the Next Class

You should know:

- Selenium automation with WebDriver
- TestNG and reports
- BDD with Cucumber
- Docker containerization
- Cloud basics and AWS concepts

Continuous Monitoring on Docker with ELK Stack

Duration: 240 min.

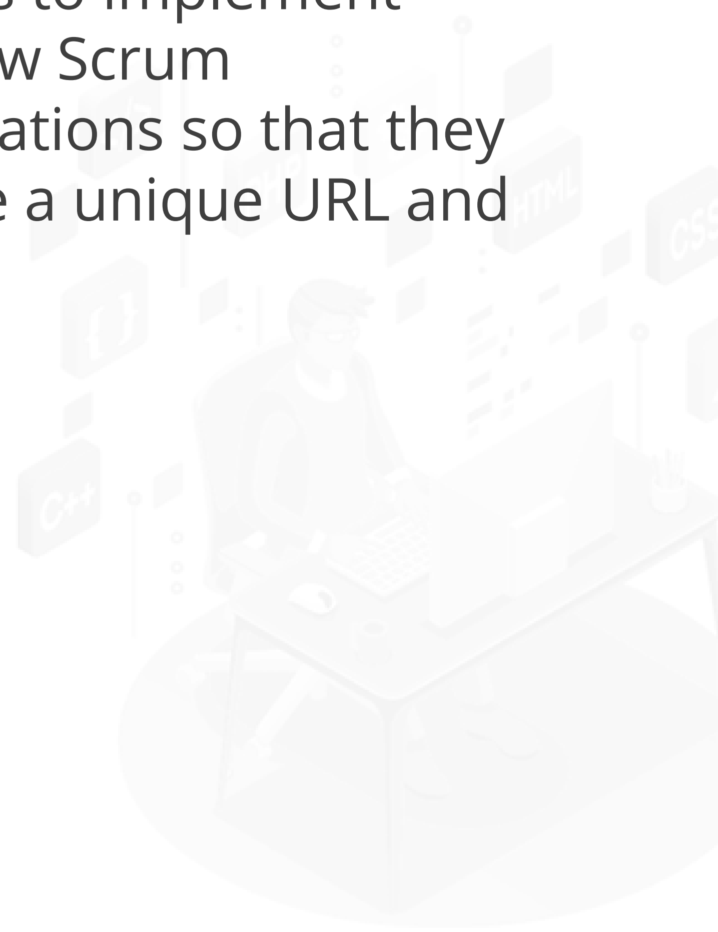
Project Objective:

Continuous Monitoring on Docker with ELK stack to monitor the application in real-time using Kibana.

PHASE-END PROJECT

Background of the Project Statement

XYZ Technology Solutions hired you as a DevOps Engineer. The company is undergoing an infrastructural change regarding the tools used in the organization. The company decides to implement DevOps to develop and deliver the products. Since XYZ is an agile organization, they follow Scrum methodology to develop the projects incrementally. They decide to dockerize their applications so that they can deploy them on Kubernetes. Each application, when deployed and exposed, will have a unique URL and port, using which we can access that application.



You Are Asked to Do

The application should have the following features:

- The application and its versions should be available on GitHub
- Commit the code multiple times and track their versions on GitHub
- Build the application in Docker and host it in Docker Hub
- Deploy ELK stack on Docker and push application logs to it
- Automate Docker build and deployment using Jenkins pipeline code

You Must Use the Following

IDE: Eclipse or IntelliJ

Programming language: Java

ELK Stack

Docker