DevOps Certification Training

Lesson 08: Need of Cloud in DevOps

Learning Objectives

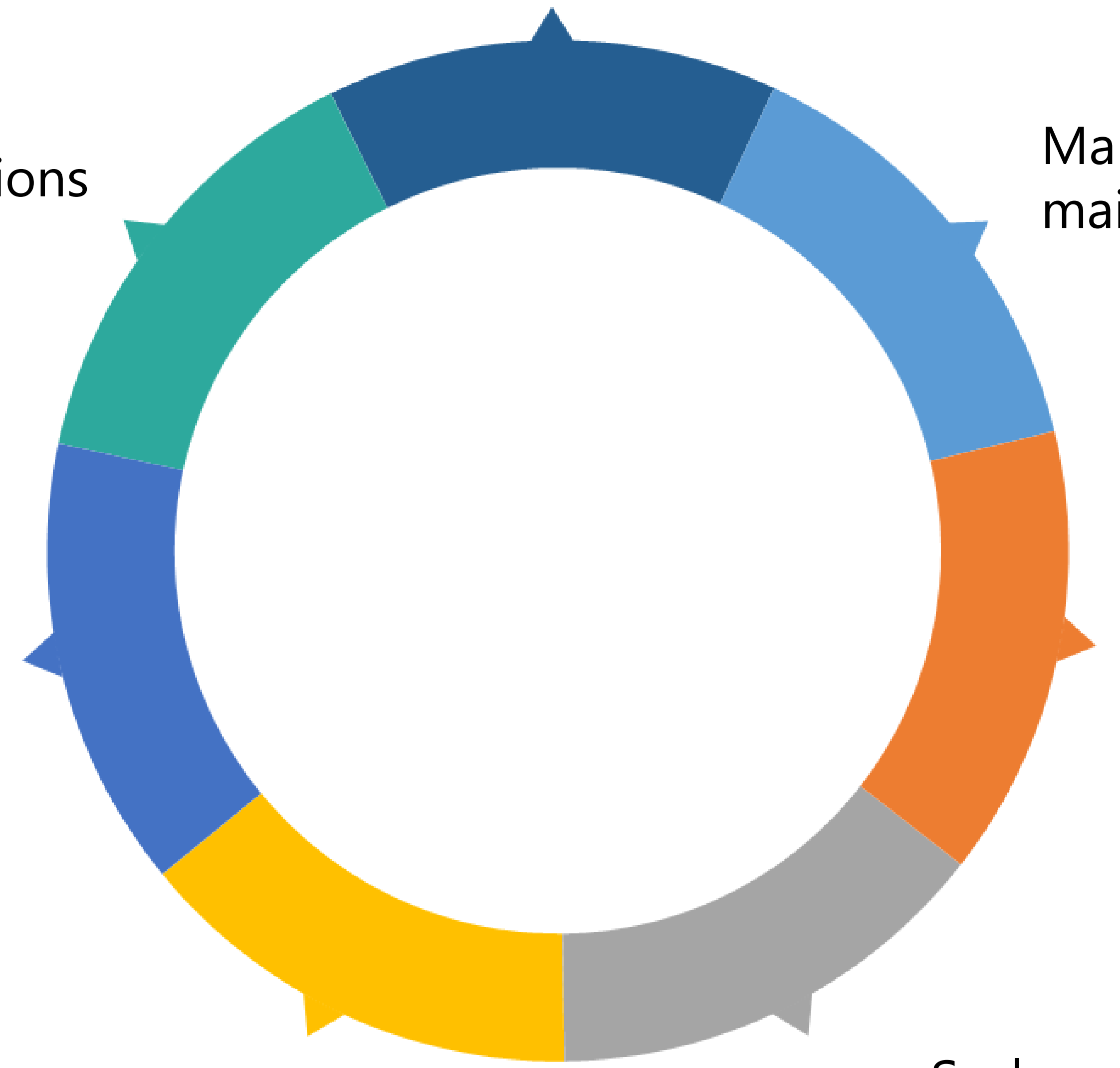
By the end of this lesson, you will be able to:

- ✓ Describe the concepts of cloud computing
- ✓ Explain the importance of cloud in DevOps
- ✓ Explain the need of AWS in DevOps
- ✓ Demonstrate the use of Kubernetes

Need of Cloud in DevOps

Overview of Cloud Computing

Cloud Computing



Parallel and distributed systems manages through the internet

Manages virtual resources with minimum maintenance

Shared pool of computing resources interconnected over networks

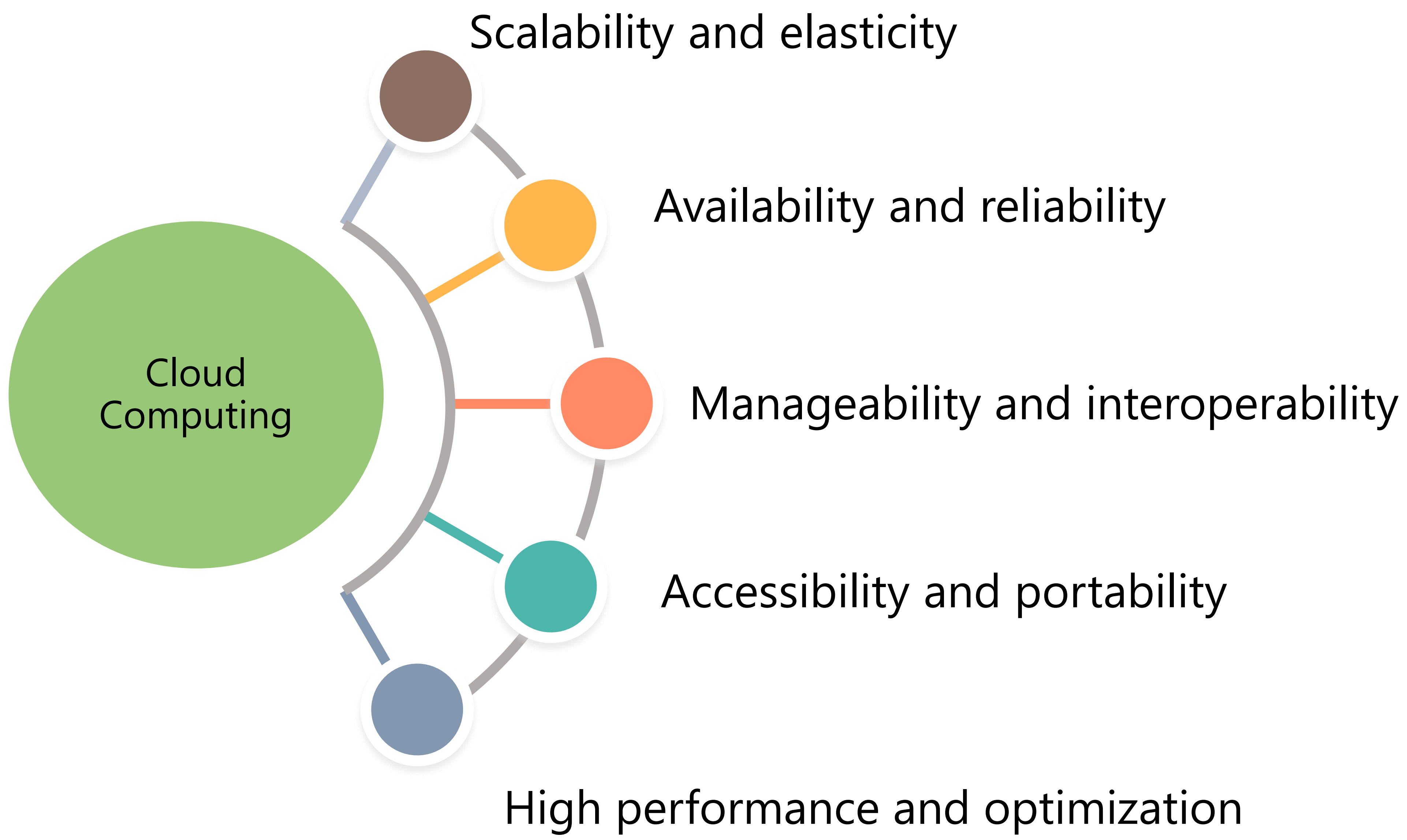
Scales up and down resources through the internet

Easy to implement and use

On-demand self service portal used to facilitate computing resources

Fast and inexpensive computing solutions

Cloud Computing Characteristics


Cloud Computing
Scalability and elasticity
Availability and reliability
Manageability and interoperability
Accessibility and portability
High performance and optimization

Cloud Computing Characteristics (Contd.)

- **Scalability**: Ability to handle increasing work load and variations in computing power
- **Elasticity**: Ability to process workload by provisioning and deprovisioning of infrastructure
- **Availability:** Availability of the system in an operational state and representing uptime of the infrastructure
- **Reliability:** Ability to maintain stable functionality of the system over a period of time
- **Manageability**: Ability to manage various components in a cloud network

Cloud Computing Characteristics (Contd.)

- **Interoperability**: Ability to integrates the systems without any restrictions
- **Accessibility**: Ability to provide an environment that can be accessed by multiple users
- **Portability**: Ability to access any service and environment from any platform
- **Performance**: Ability to design a high performance environment
- **Optimization**: Ability to stabilize the cloud environment and its efficiency

Advantages of Cloud Computing

1 Helps to manage infrastructure even without any infrastructure knowledge

2 Provides updated applications and services. It increases the efficiency of applications hosted on cloud

3 Provides cost-effective measures to set up the architecture to use cloud resources

4 Follows a pay-per-use model in which users pay only for the time they use cloud infrastructure

5 Stores applications and data on servers which are easily accessible through internet

6 Helps users easily access the changes made to the cloud from any network or system

7 Provides faster server performance which makes it quick and easy to boot systems

Disadvantages of Cloud Computing

1 Needs continuous internet connection

2 Provides users no control over the way cloud resources are shared and used which results in security threats

3 Does not work well in slow internet connections

4 Has no custom backup available for the data stored on it's servers

5 Lacks certain features found in desktop software such as Excel and Google Sheets

6 Performs slow in shared pools of infrastructure environment

Need of Cloud in DevOps

Cloud Services and Models

Cloud Service Models

PaaS Model

- Platform as a Service (PaaS) offers various types of development environments which can be used for product development.
- Examples: Google App Engine, Azure, and AWS Elastic Beanstalk

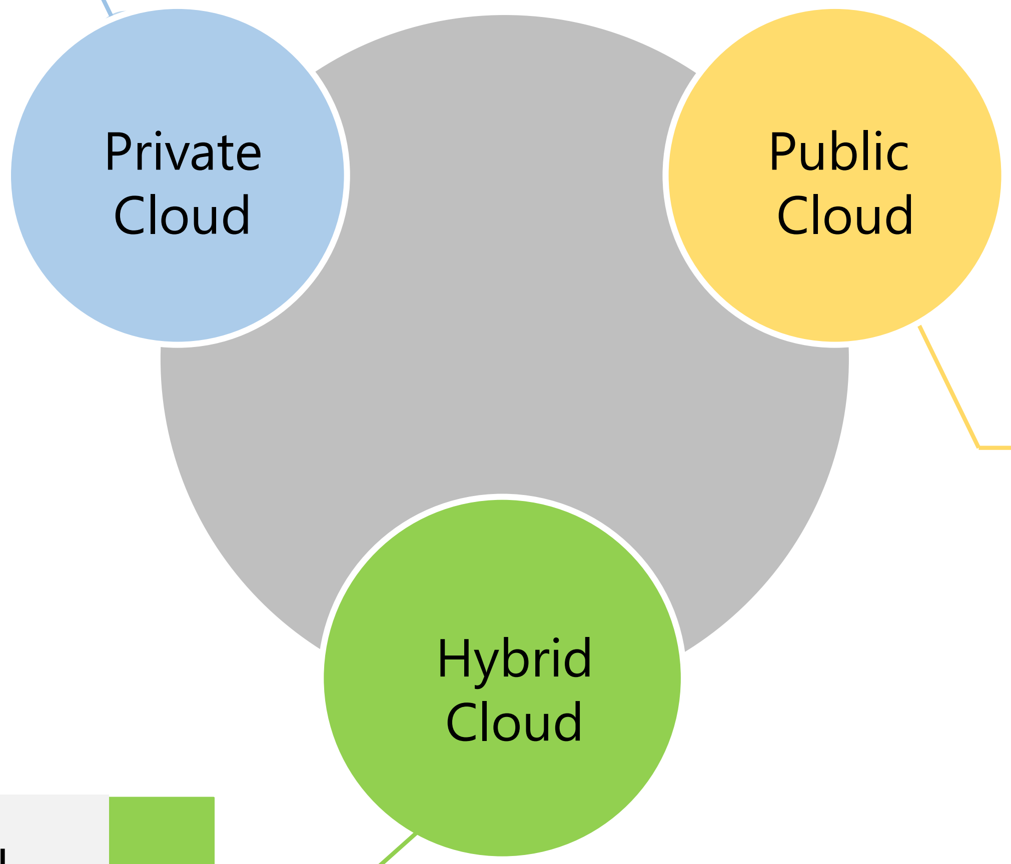
SaaS Model

- Software as a Service (SaaS) is an online service in which application software is provided to multiple users at a specific price.
- Examples: Google Apps, Salesforce.com, and CRM

IaaS Model

- Infrastructure as a Service (IaaS) is an online service which provides physical resources like servers, databases, load balancers, and storage.
- Examples: EC2, virtual machines, compute engines, load balancers, and firewalls

Cloud Models



Cloud infrastructure deployed on the private servers of organizations. It is secure and reliable. Examples: Data Centers and OpenNebula.

Cloud service on a public cloud is less secure than on a private network. Examples:AWS, Azure, Google Cloud.

Combination of both, public and private cloud. It is a composition of different cloud providers.

Popular Cloud Providers

amazon web services

Microsoft Azure

Google Cloud Platform

IBM **Cloud**

Alibaba Cloud

DigitalOcean

Need of Cloud in DevOps

Using AWS in DevOps

CI/CD in AWS

Continuous integration is a process of merging all working copies of developers to main repositories on a daily basis. It deals with building source codes and executing the initial test cases.

Continuous deployment performs automatic deployment once the build is successful. This is the last step in designing a complete workflow of build and deployment automation. It helps in quicker development and deployment.

CI/CD Services in AWS

AWS CodeCommit

AWS CodeDeploy

AWS CodeBuild

AWS CodePipeline

CI/CD Services in AWS (Contd.)

AWS CodeCommit

CodeCommit is a highly scalable and secure source code management system. It provides private Git repositories. Uptime and space utilization are the major drawbacks. Multiple repositories can be created in it.

AWS CodeBuild

AWS CodeBuild helps to implement continuous integration for the cloud applications. You can compile source codes, execute test cases, and generate binaries from source codes for deployment. Private build servers and continuous integration environment will not be created. It can be used for multiple applications and various platforms.

CI/CD Services in AWS (Contd.)

AWS CodePipeline

CodePipeline is a continuous delivery service which helps to automate builds, tests, and deploy your application. It helps to release the versions of applications as source code changes. It is a single solution to performing a complete CI/CD workflow.

AWS CodeDeploy

CodeDeploy is a cloud deployment service which helps to perform deployment automation to EC2 servers, Lambda, Elastic Beanstalk, and other physical systems. It supports easy deployment without manual errors. It eliminates the additional step of configuring Jenkins or other deploy tools.

Need of Cloud in DevOps

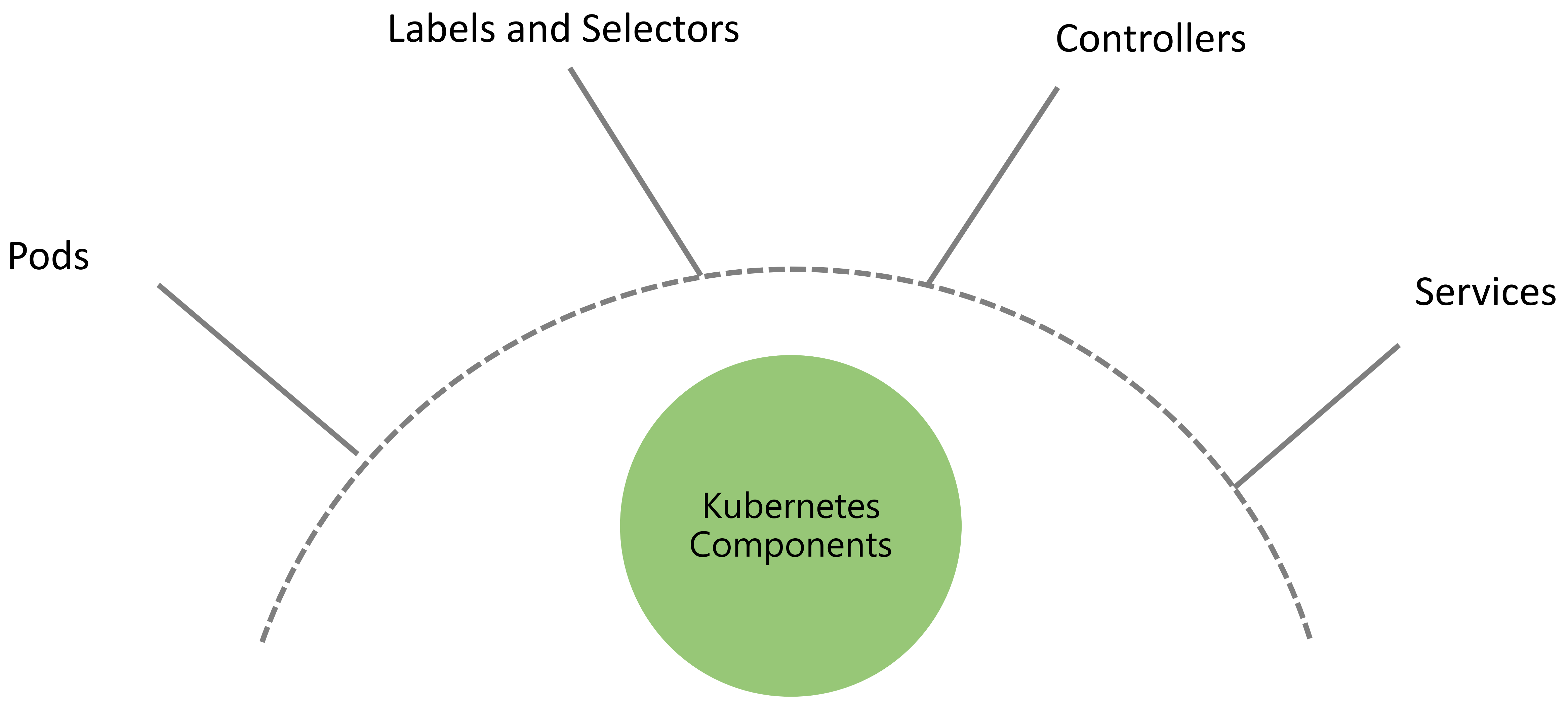
Kubernetes

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
- Same label can be used for multiple objects, so you should define and create unique labels for Kubernetes objects

Controllers

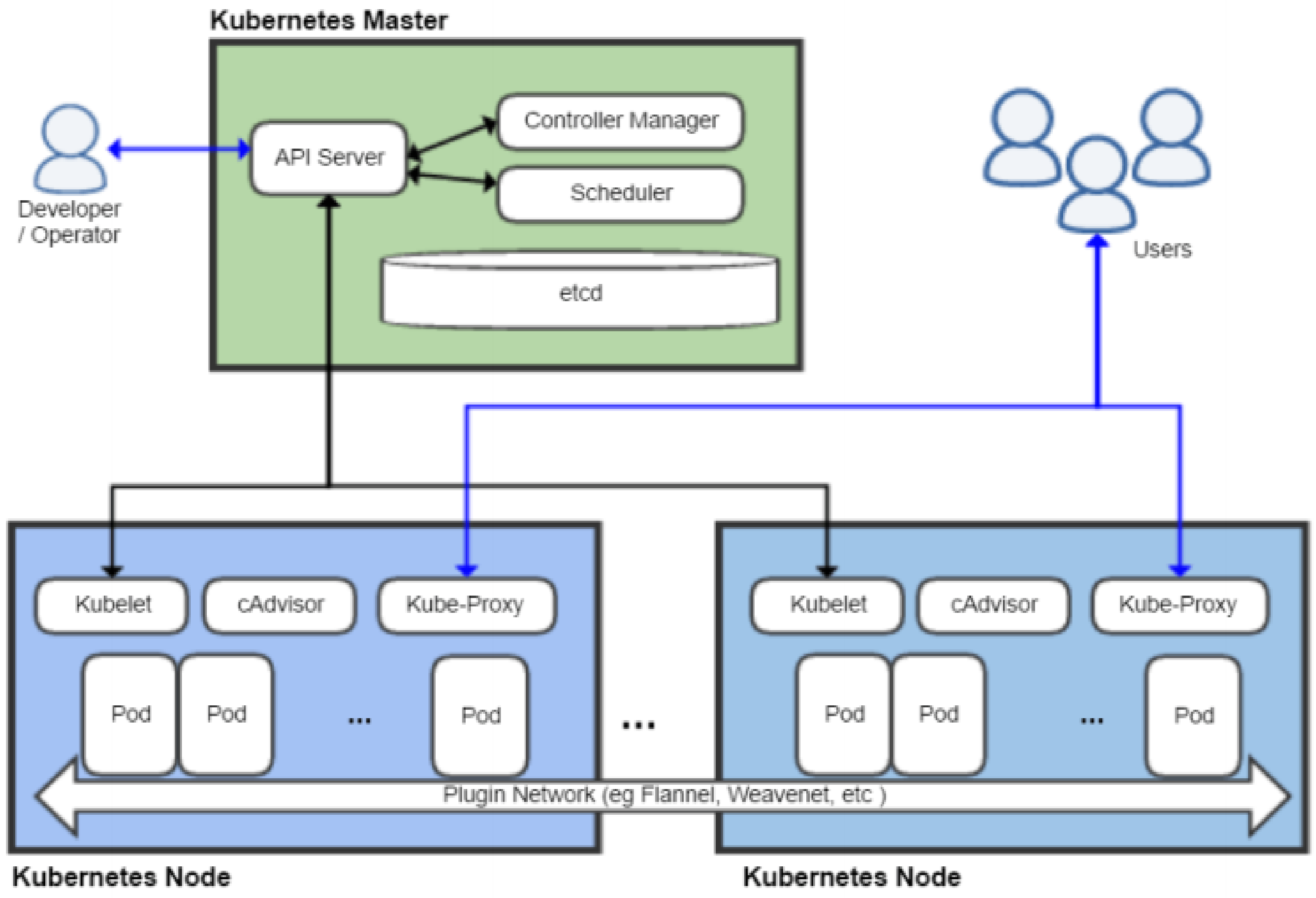
- Controllers bring pods to a specific state
- ReplicationController replicates and scales pods across Kubernetes clusters
- Controllers take care of availability of pods, and if it fails, a replacement pod gets created automatically
- DaemonSet controller ensures only one pod runs on each node
- Job 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

- 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 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 of 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.

simplilearn

Kubernetes Master Components (Contd.)

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 (Contd.)

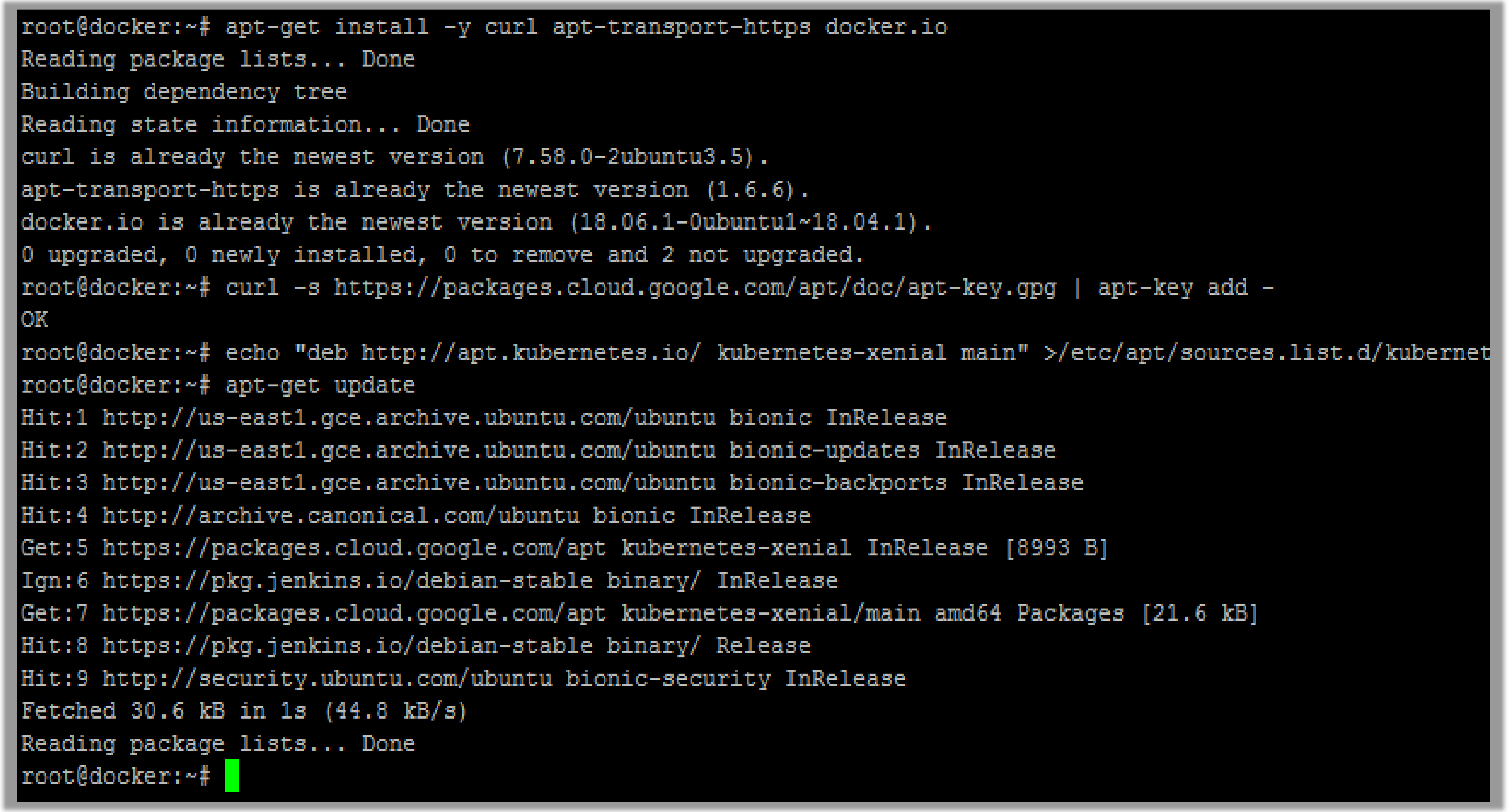
Kubelet

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.

Need of Cloud in DevOps

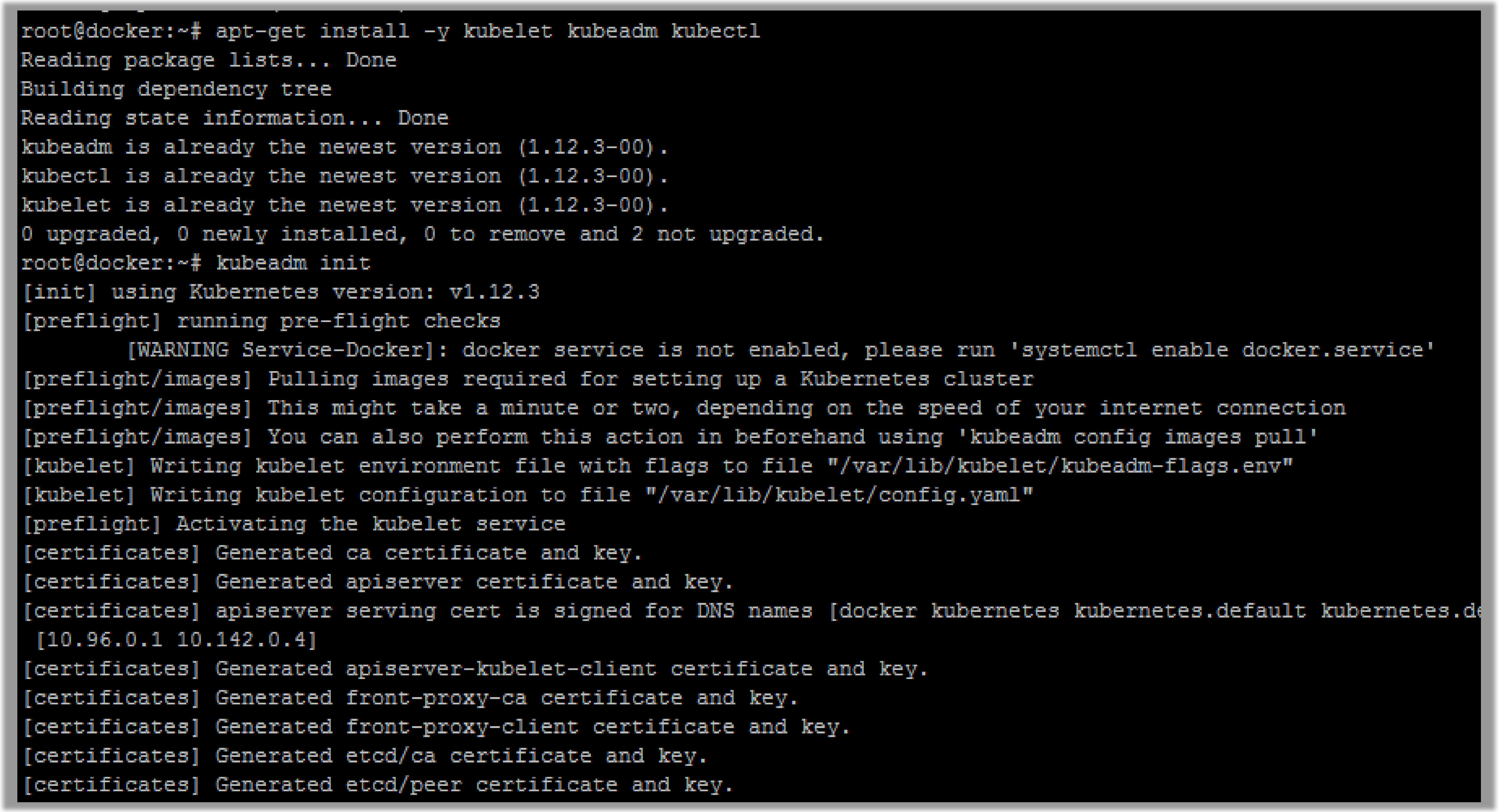
Demonstration of the “kubectl” Command

Kubernetes Installation



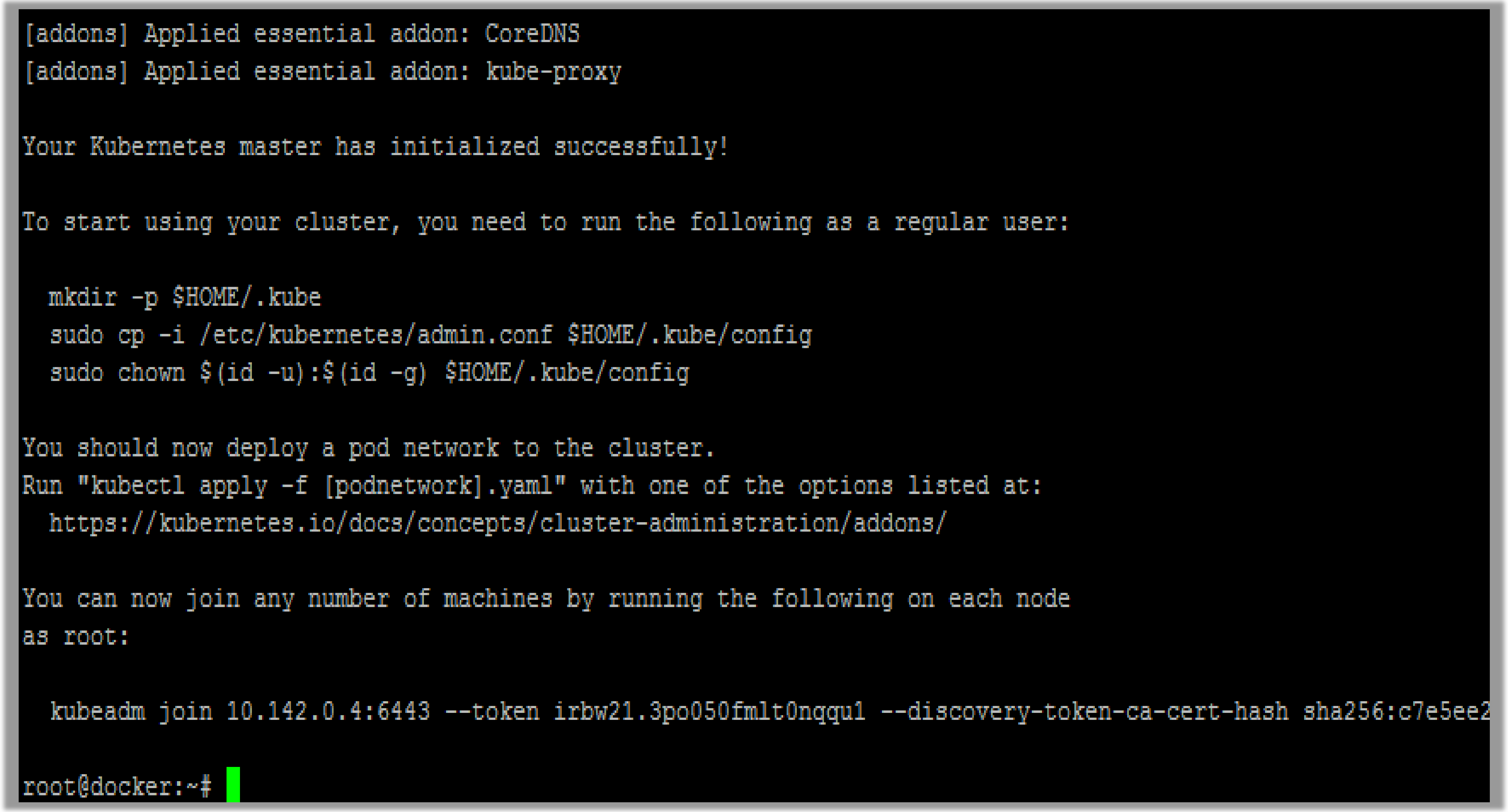
```
root@docker:~# apt-get install -y curl apt-transport-https docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
curl is already the newest version (7.58.0-2ubuntu3.5).
apt-transport-https is already the newest version (1.6.6).
docker.io is already the 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 (Contd.)



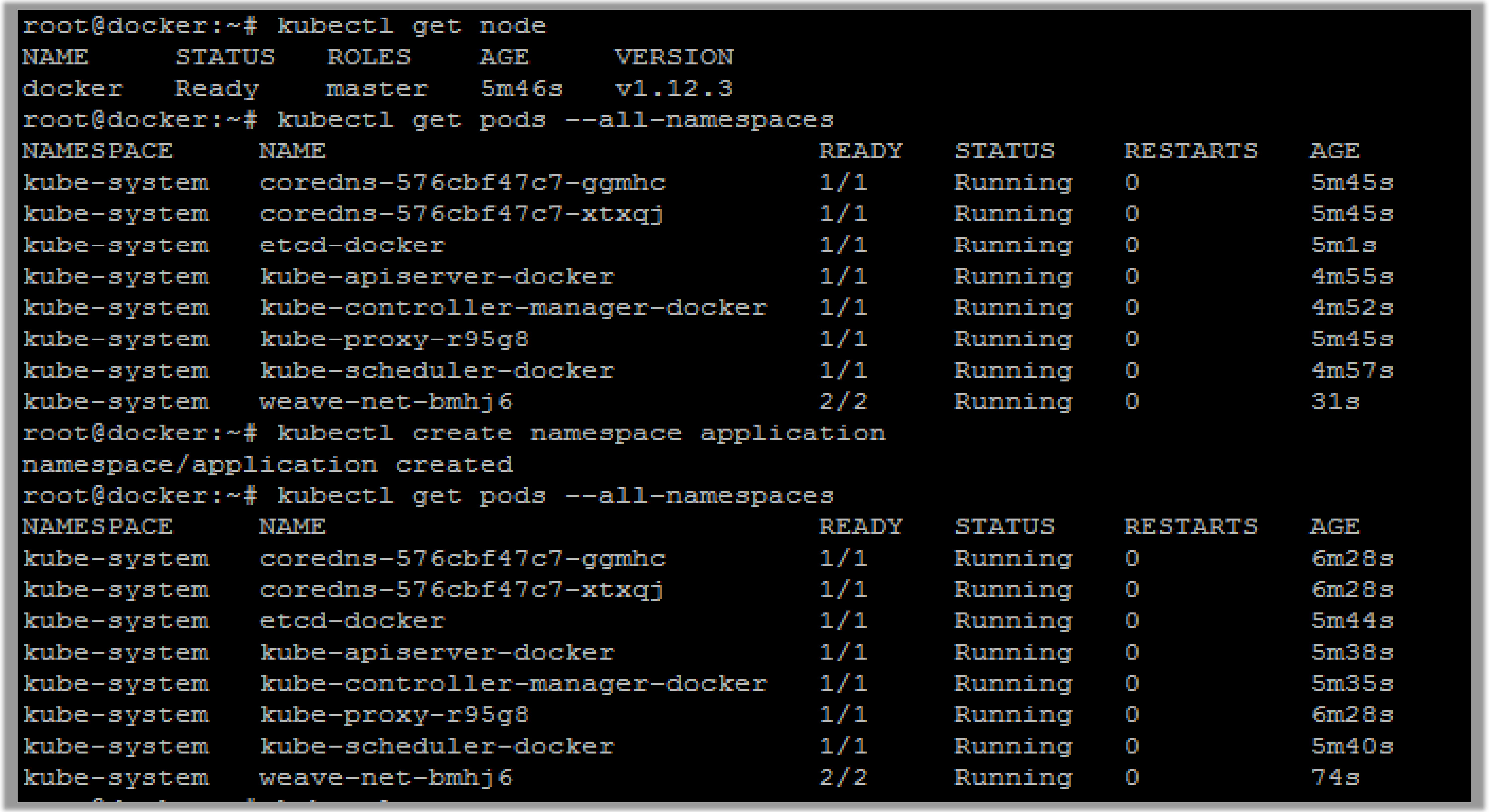
```
root@docker:~# apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree
Reading state information... Done
kubeadm is already the newest version (1.12.3-00).
kubectl is already the newest version (1.12.3-00).
kubelet is already the newest version (1.12.3-00).
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 (Contd.)



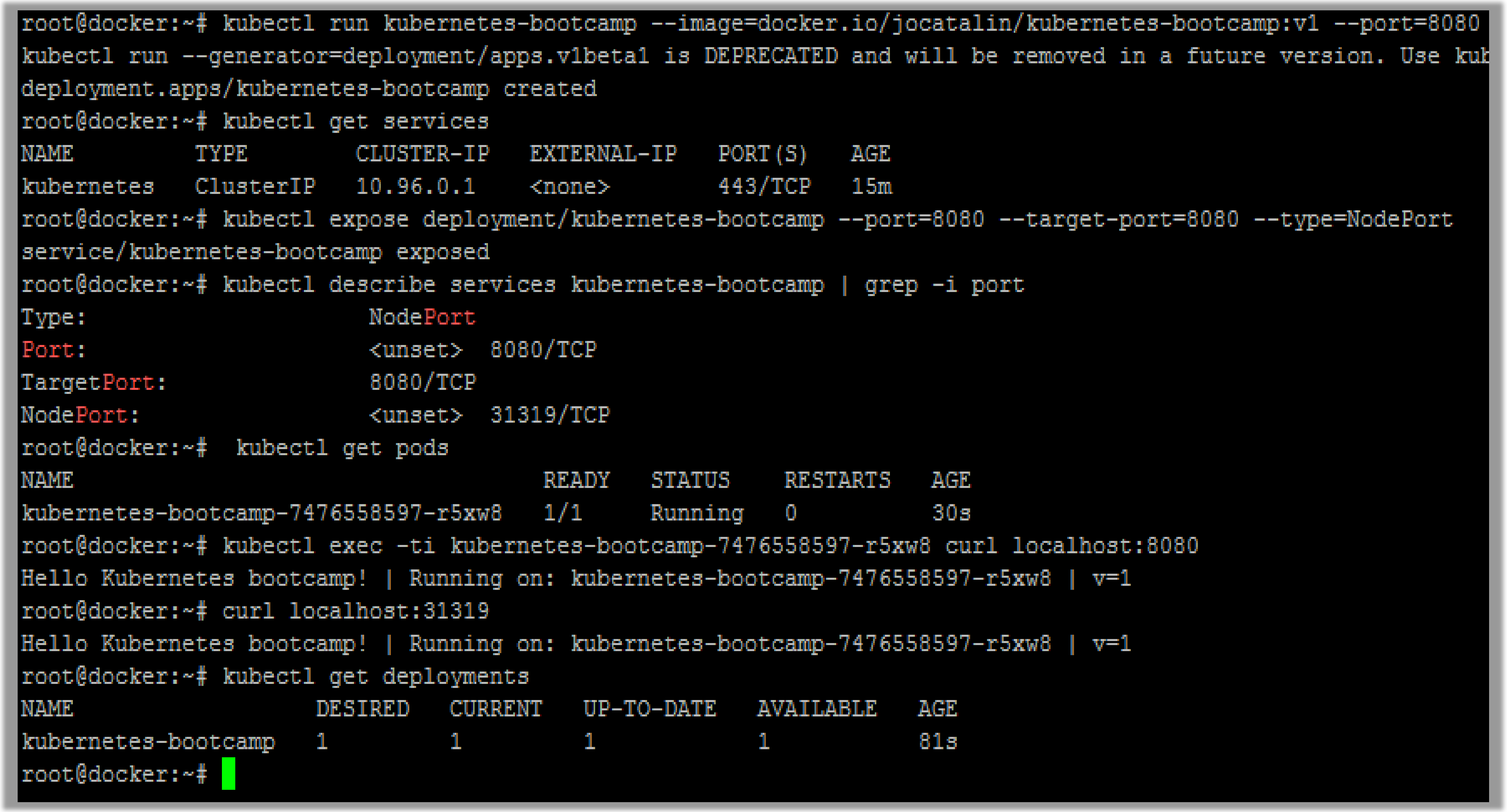
```
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

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 (Contd.)



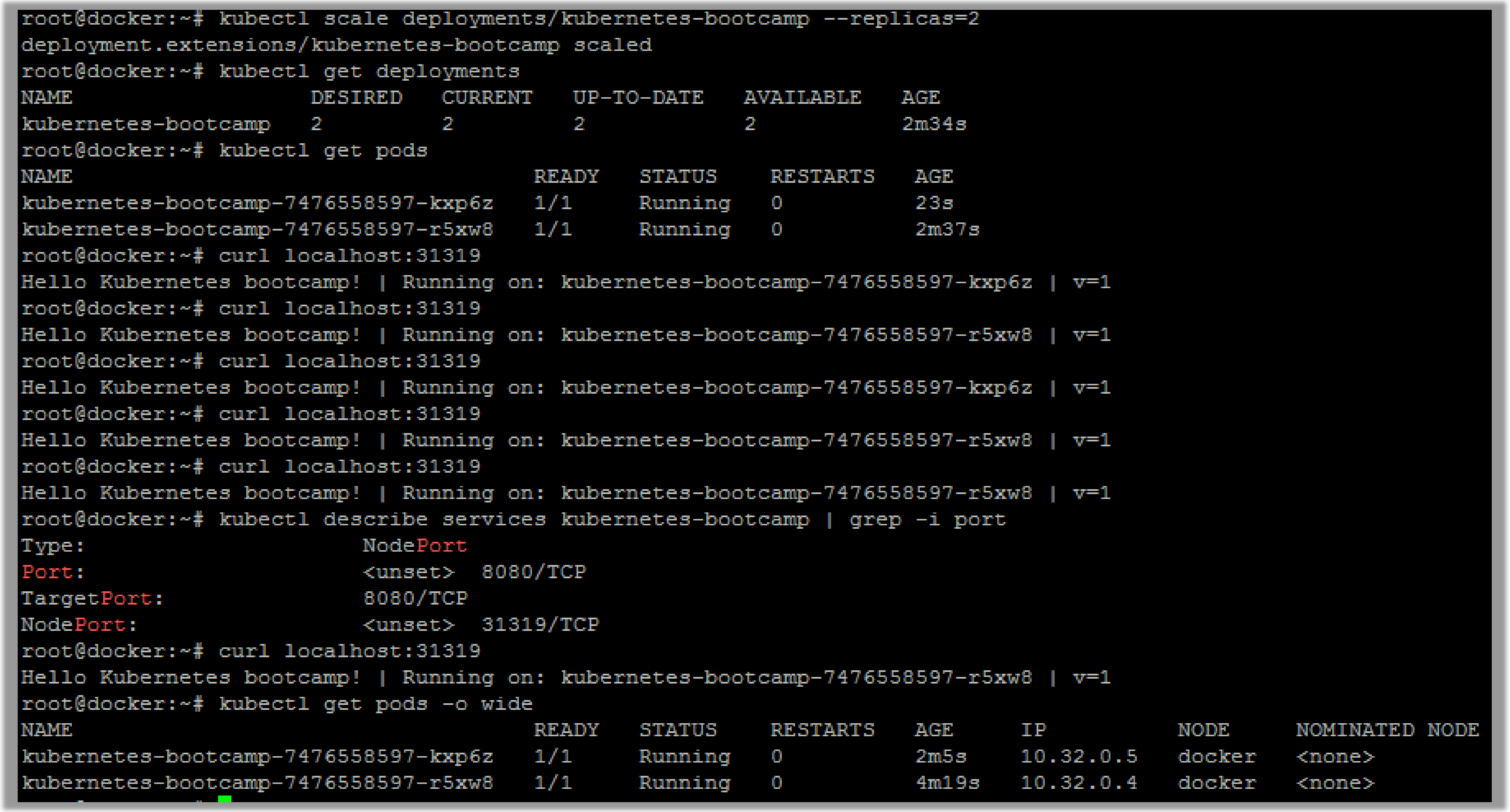
```
root@docker:~# kubectl get node
NAME      STATUS   ROLES    AGE     VERSION
docker    Ready    master   5m46s   v1.12.3
root@docker:~# kubectl get pods --all-namespaces
NAMESPACE     NAME                             READY   STATUS    RESTARTS   AGE
kube-system   coredns-576cbf47c7-ggmhc         1/1     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 (Contd.)



```
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 (Contd.)



```
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>
```

Assisted Practice

Duration: 40 mins

Add a Linux Node to the Kubernetes Cluster

Problem Statement: You are given a project to demonstrate the installation of Kubernetes, configure pods, and add a node to the Kubernetes cluster.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in their respective fields, and click **Login**.

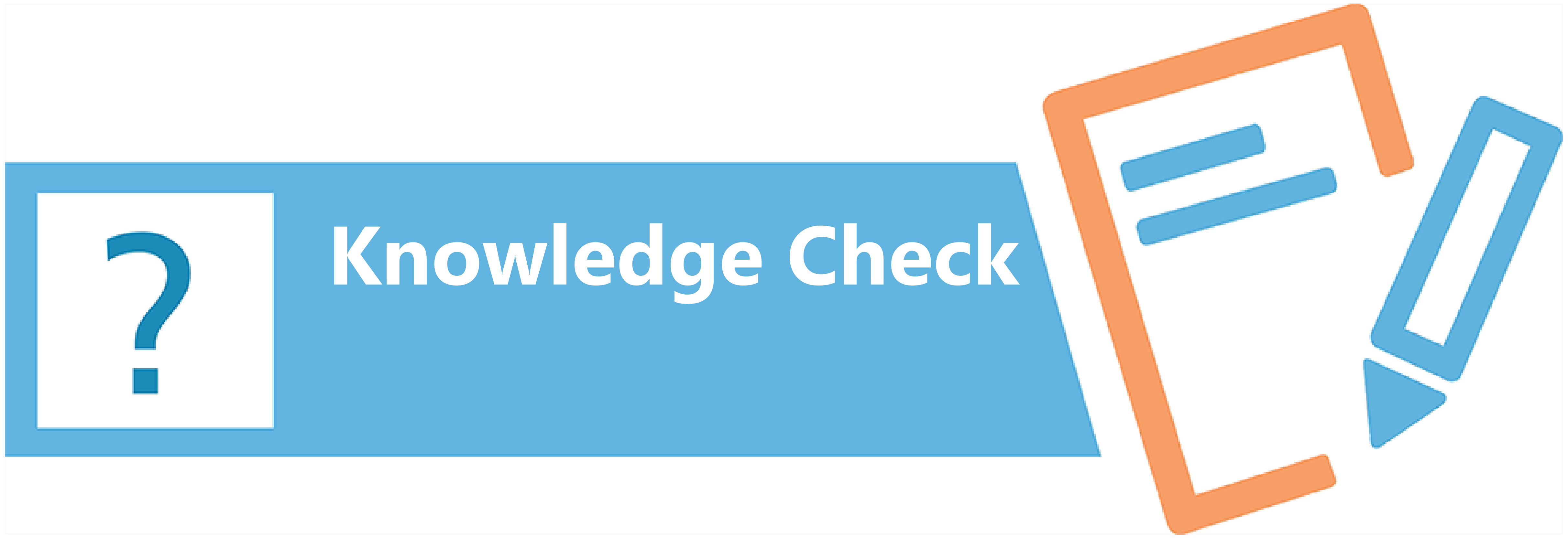
Assisted Practice: Guidelines to the Demonstration of Adding a Node to the Cluster

1. Login to your Ubuntu Lab, and open the terminal.
2. Install Docker from the official site and check the version.
3. Enable Docker service to start on system boot.
4. Install Kubernetes, and update the apt-get package.
5. Install the tools: kubelet, kubeadm, and kubectl.
6. Apply the network configurations to the Kubernetes cluster.
7. Check the result of the pods.
8. Add a node to the existing Kubernetes cluster.

Key Takeaways

You are now able to:

- ✓ Describe the concepts of cloud computing
- ✓ Explain the importance of cloud in DevOps
- ✓ Explain the need of AWS in DevOps
- ✓ Demonstrate the use of Kubernetes



Knowledge Check

Knowledge Check

1

Which one of the following cloud service models provides tools and environments to deploy applications?

a. PaaS

b. SaaS

c. IaaS

d. All of the above

Knowledge Check

1

Which one of the following cloud service models provides tools and environments to deploy applications?

a. PaaS

b. SaaS

c. IaaS

d. All of the above

The correct answer is **a. PaaS**

PaaS provides the platform that can be used to host an application without any setup.

Knowledge Check

2

Which of the following is the most important area of concern in cloud computing?

a. Security

b. Storage

c. Scalability

d. All of the above

Knowledge Check

2

Which of the following is the most important area of concern in cloud computing?

a. Security

b. Storage

c. Scalability

d. All of the above

The correct answer is **a. Security**

Security is one of the areas of concern which can impact environment on a high scale.

Knowledge Check

3

Which of the following will run on Kubernetes nodes?

a. Kubelet

b. kube-proxy

c. Pod

d. All of the above

Knowledge Check
3

Which of the following will run on Kubernetes nodes?

a. Kubelet

b. kube-proxy

c. Pod

d. All of the above

The correct answer is **d. All of the above**

All the mentioned components run on Kubernetes nodes. They are interconnected and work together to host applications on pods.

Knowledge Check 4

Which of the following is responsible for replication of controllers?

a. ReplicationController

b. DaemonSet controller

c. Node controller

d. All of the above

Knowledge Check

4

Which of the following is responsible for replication of controllers?

a. ReplicationController

b. DaemonSet controller

c. Node controller

d. All of the above

The correct answer is **a. ReplicationController**

ReplicationController is responsible for replicating pods in case any node fails.

Lesson-End Project

Host Docker on a Kubernetes Cluster

Duration: 40 mins

Problem Statement:

Perform the following:

- Clone the GitHub repository to build the image.
- Confirm if any of the pods are running in the Kubernetes cluster.
- Run the Docker app at port 80.
- Host the Docker image built on a Kubernetes cluster.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in their respective fields, and click **Login**.

Thank You