



Kubernetes

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

About the Tutorial

Kubernetes is a container management technology developed in Google lab to manage containerized applications in different kind of environments such as physical, virtual, and cloud infrastructure. It is an open source system which helps in creating and managing containerization of application.

This tutorial provides an overview of different kind of features and functionalities of Kubernetes and teaches how to manage the containerized infrastructure and application deployment.

Audience

This tutorial has been prepared for those who want to understand the containerized infrastructure and deployment of application on containers. This tutorial will help in understanding the concepts of container management using Kubernetes.

Prerequisites

We assume anyone who wants to understand Kubernetes should have an understating of how the Docker works, how the Docker images are created, and how they work as a standalone unit. To reach to an advanced configuration in Kubernetes one should understand basic networking and how the protocol communication works.

Copyright & Disclaimer

© Copyright 2018 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

About the Tutorial	i
Audience	i
Prerequisites	i
Copyright & Disclaimer	i
Table of Contents	ii
BASIC KUBERNETES	1
1. Kubernetes – Overview	2
2. Kubernetes – Architecture	3
Kubernetes – Cluster Architecture	3
Kubernetes – Master Machine Components	3
Kubernetes – Node Components	4
Kubernetes –	5
Master and Node Structure	5
3. Kubernetes – Setup	6
Prerequisites	6
Install Docker Engine	7
Install etcd 2.0	7
Configure kube-apiserver	9
Configure the kube Controller Manager	9
Kubernetes Node Configuration	10
4. Kubernetes – Images	12
5. Kubernetes – Jobs	14
Creating a Job	14
Scheduled Job	15
6. Kubernetes – Labels & Selectors	17
Labels	17
Selectors	17
7. Kubernetes – Namespace	19
8. Kubernetes – Nodes	21
9. Kubernetes – Service	23
10. Kubernetes – Pod	26
Types of Pod	26
11. Kubernetes – Replication Controller	28
12. Kubernetes – Replica Sets	30

13. Kubernetes – Deployments	32
14. Kubernetes – Volumes	34
Persistent Volume and Persistent Volume Claim	35
15. Kubernetes – Secrets.....	39
Using Secrets	40
16. Kubernetes – Network Policy	43
ADVANCED KUBERNETES	45
17. Kubernetes – API.....	46
Adding API to Kubernetes.....	46
API Changes	46
API Versioning.....	47
18. Kubernetes – Kubectl	48
Setting Kubectl.....	48
Configuring Kubectl	48
Verifying the Setup	49
19. Kubernetes – Kubectl Commands.....	50
20. Kubernetes – Creating an App	58
21. Kubernetes – App Deployment	61
22. Kubernetes – Autoscaling.....	67
23. Kubernetes – Dashboard Setup	69
24. Kubernetes – Monitoring	74

Basic Kubernetes

1. Kubernetes – Overview

Kubernetes is an open source container management tool hosted by Cloud Native Computing Foundation (CNCF). This is also known as the enhanced version of Borg which was developed at Google to manage both long running processes and batch jobs, which were earlier handled by separate systems.

Kubernetes comes with a capability of automating deployment, scaling of application, and operations of application containers across clusters. It is capable of creating container-centric infrastructure.

Features of Kubernetes

Following are some of the important features of Kubernetes.

- Continuous development, integration and deployment
- Containerized infrastructure
- Application-centric management
- Auto-scalable infrastructure
- Environment consistency across development testing and production
- Loosely coupled infrastructure, where each component can act as a separate unit
- Higher density of resource utilization
- Predictable infrastructure which is going to be created

One of the key components of Kubernetes is, it can run application on clusters of physical and virtual machine infrastructure. It also has the capability to run applications on cloud.

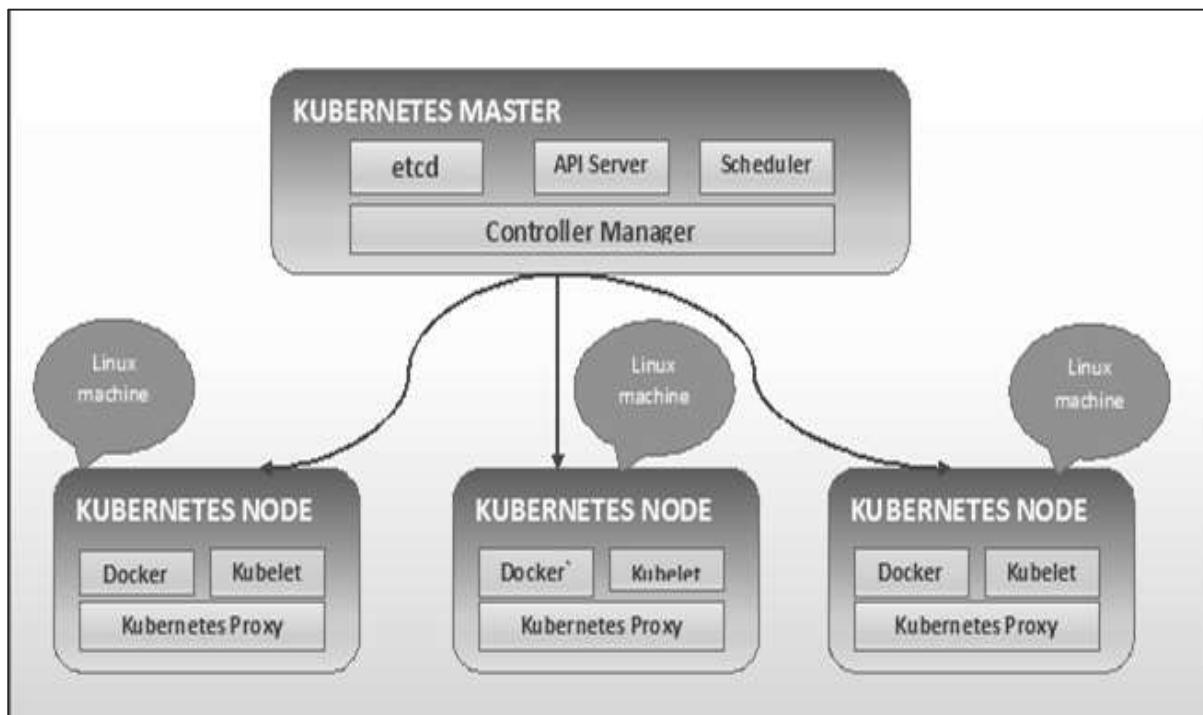
It helps in moving from host-centric infrastructure to container-centric infrastructure.

2. kubernetes – Architecture

In this chapter, we will discuss the basic architecture of Kubernetes.

Kubernetes – Cluster Architecture

As seen in the following diagram, Kubernetes follows client-server architecture. Wherein, we have master installed on one machine and the node on separate Linux machines.



The key components of master and node are defined in the following section.

Kubernetes – Master Machine Components

Following are the components of Kubernetes Master Machine.

etcd

It stores the configuration information which can be used by each of the nodes in the cluster. It is a high availability key value store that can be distributed among multiple nodes. It is accessible only by Kubernetes API server as it may have some sensitive information. It is a distributed key value Store which is accessible to all.

API Server

Kubernetes is an API server which provides all the operation on cluster using the API. API server implements an interface, which means different tools and libraries can readily

communicate with it. **Kubeconfig** is a package along with the server side tools that can be used for communication. It exposes Kubernetes API.

Controller Manager

This component is responsible for most of the controllers that regulate the state of cluster and perform a task. In general, it can be considered as a daemon which runs in a non-terminating loop and is responsible for collecting and sending information to API server. It works toward getting the shared state of cluster and then make changes to bring the current status of the server to the desired state. The key controllers are replication controller, endpoint controller, namespace controller, and service account controller. The controller manager runs different kind of controllers to handle nodes, endpoints, etc.

Scheduler

This is one of the key components of Kubernetes master. It is a service in master responsible for distributing the workload. It is responsible for tracking utilization of working load on cluster nodes and then placing the workload on which resources are available and accept the workload. In other words, this is the mechanism responsible for allocating pods to available nodes. The scheduler is responsible for workload utilization and allocating pod to new node.

Kubernetes — Node Components

Following are the key components of Node server which are necessary to communicate with Kubernetes master.

Docker

The first requirement of each node is Docker which helps in running the encapsulated application containers in a relatively isolated but lightweight operating environment.

Kubelet Service

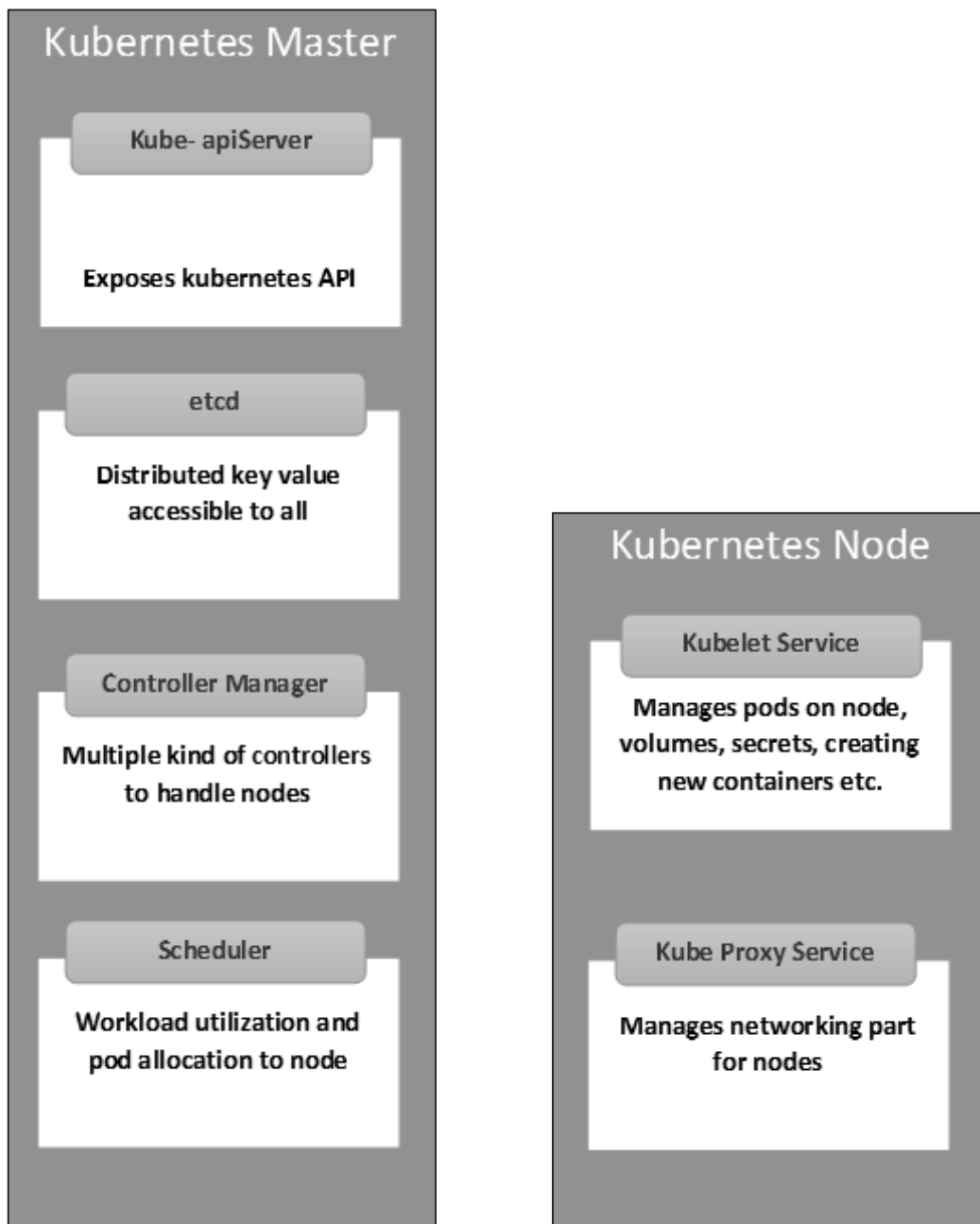
This is a small service in each node responsible for relaying information to and from control plane service. It interacts with **etcd** store to read configuration details and write values. This communicates with the master component to receive commands and work. The **kubelet** process then assumes responsibility for maintaining the state of work and the node server. It manages network rules, port forwarding, etc.

Kubernetes Proxy Service

This is a proxy service which runs on each node and helps in making services available to the external host. It helps in forwarding the request to correct containers and is capable of performing primitive load balancing. It makes sure that the networking environment is predictable and accessible and at the same time it is isolated as well. It manages pods on node, volumes, secrets, creating new containers' health checkup, etc.

Kubernetes — Master and Node Structure

The following illustrations show the structure of Kubernetes Master and Node.



3. Kubernetes – Setup

It is important to set up the Virtual Datacenter (vDC) before setting up Kubernetes. This can be considered as a set of machines where they can communicate with each other via the network. For hands-on approach, you can set up vDC on **PROFITBRICKS** if you do not have a physical or cloud infrastructure set up.

Once the IaaS setup on any cloud is complete, you need to configure the **Master** and the **Node**.

Note: The setup is shown for Ubuntu machines. The same can be set up on other Linux machines as well.

Prerequisites

Installing Docker: Docker is required on all the instances of Kubernetes. Following are the steps to install the Docker.

Step 1: Log on to the machine with the root user account.

Step 2: Update the package information. Make sure that the apt package is working.

Step 3: Run the following commands.

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates
```

Step 4: Add the new GPG key.

```
$ sudo apt-key adv \
    --keyserver hkp://ha.pool.sks-keyservers.net:80 \
    --recv-keys 58118E89F3A912897C070ADB76221572C52609D

$ echo "https://apt.dockerproject.org/repo ubuntu-trusty main" | sudo tee
/etc/apt/sources.list.d/docker.list
```

Step 5: Update the API package image.

```
$ sudo apt-get update
```

Once all the above tasks are complete, you can start with the actual installation of the Docker engine. However, before this you need to verify that the kernel version you are using is correct.

Install Docker Engine

Run the following commands to install the Docker engine.

Step 1: Logon to the machine.

Step 2: Update the package index.

```
$ sudo apt-get update
```

Step 3: Install the Docker Engine using the following command.

```
$ sudo apt-get install docker-engine
```

Step 4: Start the Docker daemon.

```
$ sudo apt-get install docker-engine
```

Step 5: To verify if the Docker is installed, use the following command.

```
$ sudo docker run hello-world
```

Install etcd 2.0

This needs to be installed on Kubernetes Master Machine. In order to install it, run the following commands.

```
$ curl -L https://github.com/coreos/etcd/releases/download/v2.0.0/etcd
-v2.0.0-linux-amd64.tar.gz -o etcd-v2.0.0-linux-amd64.tar.gz ->1

$ tar xzvf etcd-v2.0.0-linux-amd64.tar.gz -----> 2

$ cd etcd-v2.0.0-linux-amd64 ----->3
$ mkdir /opt/bin ----->4
$ cp etcd* /opt/bin ----->5
```

In the above set of command:

- First, we download the **etcd**. Save this with specified name.
- Then, we have to un-tar the tar package.
- We make a dir. inside the /opt named bin.
- Copy the extracted file to the target location.

Now we are ready to build Kubernetes. We need to install Kubernetes on all the machines on the cluster.

```
$ git clone https://github.com/GoogleCloudPlatform/kubernetes.git
$ cd kubernetes
$ make release
```

The above command will create a **_output** dir in the root of the kubernetes folder. Next, we can extract the directory into any of the directory of our choice /opt/bin, etc.

Next, comes the networking part wherein we need to actually start with the setup of Kubernetes master and node. In order to do this, we will make an entry in the host file which can be done on the node machine.

```
$ echo "<IP address of master machine> kube-master
      < IP address of Node Machine>" >> /etc/hosts
```

Following will be the output of the above command.

```
root@boot2docker:/etc# cat /etc/hosts
127.0.0.1 boot2docker localhost localhost.local

# The following lines are desirable for IPv6 capable hosts
# (added automatically by netbase upgrade)

::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters
ff02::3  ip6-allhosts

10.11.50.12 kube-master
10.11.50.11 kube-minion
```

Now, we will start with the actual configuration on Kubernetes Master.

First, we will start copying all the configuration files to their correct location.

```
$ cp <Current dir. location>/kube-apiserver /opt/bin/
$ cp <Current dir. location>/kube-controller-manager /opt/bin/
$ cp <Current dir. location>/kube-kube-scheduler /opt/bin/
$ cp <Current dir. location>/kubecfg /opt/bin/
$ cp <Current dir. location>/kubectl /opt/bin/
$ cp <Current dir. location>/kubernetes /opt/bin/
```

The above command will copy all the configuration files to the required location. Now we will come back to the same directory where we have built the Kubernetes folder.

```

$ cp kubernetes/cluster/ubuntu/init_conf/kube-apiserver.conf /etc/init/
$ cp kubernetes/cluster/ubuntu/init_conf/kube-controller-manager.conf
/etc/init/
$ cp kubernetes/cluster/ubuntu/init_conf/kube-kube-scheduler.conf /etc/init/

$ cp kubernetes/cluster/ubuntu/initd_scripts/kube-apiserver /etc/init.d/
$ cp kubernetes/cluster/ubuntu/initd_scripts/kube-controller-manager
/etc/init.d/
$ cp kubernetes/cluster/ubuntu/initd_scripts/kube-kube-scheduler /etc/init.d/

$ cp kubernetes/cluster/ubuntu/default_scripts/kubelet /etc/default/
$ cp kubernetes/cluster/ubuntu/default_scripts/kube-proxy /etc/default/
$ cp kubernetes/cluster/ubuntu/default_scripts/kubelet /etc/default/

```

The next step is to update the copied configuration file under /etc. dir.

Configure etcd on master using the following command.

```
$ ETCD_OPTS="-listen-client-urls=http://kube-master:4001"
```

Configure kube-apiserver

For this on the master, we need to edit the **/etc/default/kube-apiserver** file which we copied earlier.

```

$ KUBE_APISERVER_OPTS="--address=0.0.0.0 \
--port=8080 \
--etcd_servers=<The path that is configured in ETCD_OPTS> \
--portal_net=11.1.1.0/24 \
--allow_privileged=false \
--kubelet_port= < Port you want to configure> \
--v=0"

```

Configure the kube Controller Manager

We need to add the following content in **/etc/default/kube-controller-manager**.

```

$ KUBE_CONTROLLER_MANAGER_OPTS="--address=0.0.0.0 \
--master=127.0.0.1:8080 \
--machines=kube-minion \ -----> #this is the kubernatics node

```

```
--v=0
```

Next, configure the kube scheduler in the corresponding file.

```
$ KUBE_SCHEDULER_OPTS="--address=0.0.0.0 \
--master=127.0.0.1:8080 \
--v=0"
```

Once all the above tasks are complete, we are good to go ahead by bring up the Kubernetes Master. In order to do this, we will restart the Docker.

```
$ service docker restart
```

Kubernetes Node Configuration

Kubernetes node will run two services the **kubelet** and the **kube-proxy**. Before moving ahead, we need to copy the binaries we downloaded to their required folders where we want to configure the kubernetes node.

Use the same method of copying the files that we did for kubernetes master. As it will only run the kubelet and the kube-proxy, we will configure them.

```
$ cp <Path of the extracted file>/kubelet /opt/bin/
$ cp <Path of the extracted file>/kube-proxy /opt/bin/
$ cp <Path of the extracted file>/kubecfg /opt/bin/
$ cp <Path of the extracted file>/kubectl /opt/bin/
$ cp <Path of the extracted file>/kubernetes /opt/bin/
```

Now, we will copy the content to the appropriate dir.

```
$ cp kubernetes/cluster/ubuntu/init_conf/kubelet.conf /etc/init/
$ cp kubernetes/cluster/ubuntu/init_conf/kube-proxy.conf /etc/init/

$ cp kubernetes/cluster/ubuntu/initd_scripts/kubelet /etc/init.d/
$ cp kubernetes/cluster/ubuntu/initd_scripts/kube-proxy /etc/init.d/

$ cp kubernetes/cluster/ubuntu/default_scripts/kubelet /etc/default/
$ cp kubernetes/cluster/ubuntu/default_scripts/kube-proxy /etc/default/
```

End of ebook preview
If you liked what you saw...
Buy it from our store @ **<https://store.tutorialspoint.com>**