How to Create a Kubernetes Cluster on Ubuntu 16.04 with kubeadm and Weave Net

Tammy Butow
Principal SRE

Introduction

Kubernetes is a system designed to manage applications built within containers across clustered environments. It handles the entire life cycle of a containerized application including deployment and scaling.

In this guide, we'll demonstrate how to get started by creating a Kubernetes cluster (v1.15) on Ubuntu 16.04. We will be using kubeadm to setup kubernetes. We will then deploy the Weaveworks Socks Shop Microservices Application as a demonstration of how to run microservices on Kubernetes.

The purpose of this tutorial is to enable you to run a demo microservices application on a kubernetes cluster you have created.

The overall feature state of kubeadm is **Beta** and will be graduated to **General Availability** (GA) in 2018.

Prerequisites

Before you begin this tutorial, you'll need the following:

- 3 Ubuntu 16.04 servers with 4GM RAM and private networking enabled

Step 1 - Get each server ready to run Kubernetes

We will start with creating three Ubuntu 16.04 servers. This will give you three servers to configure. To get this three member cluster up and running, you will need to select Ubuntu 16.04, 4GM RAM servers and enable Private Networking.

Create 3 hosts and call them kube-01, kube-02 and kube-03. You need to be running hosts with a minimum of 4GB RAM for the Weave Socks Shop Demo.

Set your hostnames for your servers as follows:

Server	Hostname
1	kube-01
2	kube-02
3	kube-03

Kubernetes will need to assign specialized roles to each server. We will setup one server to act as the master:

Hostname	Role
kube-01	Master
kube-02	Node
kube-03	Node

Step 2 - Set up each server in the cluster to run Kubernetes.

SSH to each of the servers you created. Proceed with executing the following commands as root. You may become the root user by executing sudo -i after SSH-ing to each host.

On each of the three Ubuntu 16.04 servers run the following commands as root:

```
apt-get update && apt-get install -y apt-transport-https
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-ke
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
apt-get update
apt-get install -y kubelet=1.15.4-00 kubeadm=1.15.4-00 kubectl=1.15.4-
```

Step 3 - Setup the Kubernetes Master

On the kube-01 node run the following command:

```
kubeadm init
```

This can take a minute or two to run, the result will look like this:

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
```

Your Kubernetes master has initialized successfully!

Run the following commands on kube-01:

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

Step 4 - Join your nodes to your Kubernetes cluster

You can now join any number of machines by running the kubeadm join command on each node as root. This command will be created for you as displayed in your terminal for you to copy and run.

An example of what this looks like is below:

```
kubeadm join --token 702ff6.bc7aacff7aacab17 174.138.15.158:6443 --dis
```

When you join your kube-02 and kube-01 nodes you will see the following on the node:

```
This node has joined the cluster:
* Certificate signing request was sent to master and a response was re
* The Kubelet was informed of the new secure connection details.
```

To check that all nodes are now joined to the master run the following command on the Kubernetes master kube-01:

```
kubectl get nodes
```

The successful result will look like this:

```
NAME      STATUS    ROLES     AGE       VERSION
kube-01   Ready     master    8m        v1.9.3
kube-02   Ready     <none>    6m        v1.9.3
kube-03   Ready     <none>    6m        v1.9.3
```

You will notice that the nodes do not have a role set on join, there is an open PR to resolve this.

Step 5 - Setup a Kubernetes Add-On For Networking Features And Policy

Kubernetes Add-Ons are pods and services that implement cluster features. Pods extend the functionality of Kubernetes. You can install addons for a range of cluster features including Networking and Visualization.

We are going to install the Weave Net Add-On on the kube-01 master which provides networking and network policy, will carry on working on both sides of a network partition, and does not require an external database. Read more about the Weave Net Add-on in the Weave Works Docs.

Next you will deploy a pod network to the cluster.

The options are listed at: https://kubernetes.io/docs/concepts/cluster-administration/addons/

Installing the Weave Net Add-On

Get the Weave Net yaml:

```
curl -o weave.yaml https://cloud.weave.works/k8s/v1.8/net.yaml
```

Inspect the yaml contents:

```
cat weave.yaml
```

On the kube-01 Kubernetes master node run the following commands:

```
kubectl apply -f weave.yaml
```

The result will look like this:

```
serviceaccount "weave-net" created
clusterrole "weave-net" created
clusterrolebinding "weave-net" created
role "weave-net" created
rolebinding "weave-net" created
daemonset "weave-net" created
```

It may take a minute or two for DNS to be ready, continue to check for DNS to be ready before moving on by running the following command:

```
kubectl get pods --all-namespaces
```

The successful result will look like this, every container should be running:

```
NAMESPACE     NAME                                READY     STATUS    RE
kube-system   etcd-kube-01                        1/1       Running   0
kube-system   kube-apiserver-kube-01              1/1       Running   0
kube-system   kube-controller-manager-kube-01     1/1       Running   0
kube-system   kube-dns-6f4fd4bdf-whbhd            3/3       Running   0
kube-system   kube-proxy-2hdhk                    1/1       Running   0
kube-system   kube-proxy-tvhjk                    1/1       Running   0
kube-system   kube-proxy-wspmv                    1/1       Running   0
kube-system   kube-scheduler-kube-01              1/1       Running   0
kube-system   weave-net-9ghn5                     2/2       Running   1
kube-system   weave-net-lh8tq                     2/2       Running   0
kube-system   weave-net-qhr25                     2/2       Running   0
```

Congratulations, now your Kubernetes cluster running on Ubuntu 16.04 is up and ready for you to deploy a microservices application.

Step 6 - Deploying The Weaveworks Microservices Sock Shop

Next we will deploy a demo microservices application to your kubernetes cluster.

First, on kube-01, clone the microservices sock shop git repo:

```
git clone https://github.com/microservices-demo/microservices-demo.git
```

Go to the microservices-demo/deploy/kubernetes folder:

```
kubectl create namespace sock-shop
```

You will see the following result:

```
namespace "sock-shop" created
```

Next apply the demo to your kubernetes cluster:

```
kubectl apply -f complete-demo.yaml
```

You will see the following result:

```
deployment "carts-db" created
service "carts-db" created
deployment "carts" created
service "carts" created
deployment "catalogue-db" created
service "catalogue-db" created
deployment "catalogue" created
service "catalogue" created
deployment "front-end" created
service "front-end" created
deployment "orders-db" created
service "orders-db" created
deployment "orders" created
service "orders" created
deployment "payment" created
service "payment" created
deployment "queue-master" created
service "queue-master" created
deployment "rabbitmq" created
```

Check to see if all of your pods are running:

```
kubectl get pods --namespace sock-shop
```

You will see the following result when all pods are ready, they will have the status of "Running":

```
NAMESPACE     NAME                                READY     STATUS
kube-system   etcd-kube-01                        1/1       Running
kube-system   kube-apiserver-kube-01              1/1       Running
kube-system   kube-controller-manager-kube-01     1/1       Running
kube-system   kube-dns-6f4fd4bdf-whbhd            3/3       Running
kube-system   kube-proxy-2hdhk                    1/1       Running
kube-system   kube-proxy-tvhjk                    1/1       Running
kube-system   kube-proxy-wspmv                    1/1       Running
kube-system   kube-scheduler-kube-01              1/1       Running
kube-system   weave-net-9ghn5                     2/2       Running
kube-system   weave-net-lh8tq                     2/2       Running
kube-system   weave-net-qhr25                     2/2       Running
sock-shop     carts-74f4558cb8-h9924              1/1       Running
sock-shop     carts-db-7fcddfbc79-v64fw           1/1       Running
sock-shop     catalogue-676d4b9f7c-55n4g          1/1       Running
sock-shop     catalogue-db-5c67cdc8cd-hvk96       1/1       Running
sock-shop     front-end-977bfd86-hq9x9            1/1       Running
sock-shop     orders-787bf5b89f-xfdl6             1/1       Running
```

Visit http://174.138.15.158:30001/ to see the Sock Shop working:

Conclusion

You have created a Kubernetes cluster and learned how to use the Kubernetes command-line tool kubectl. You then deployed Weave Socks Shop Microservices Application as a demonstration of how to run microservices on Kubernetes. You have now started to see how Kubernetes is designed to manage applications built within containers across clustered environments.

To create Gremlin attacks on Kubernetes follow our guide on "How To Install And Use Gremlin With Kubernetes". Join the Chaos Engineering Slack Community to discuss how Chaos Engineering can be practiced on Kubernetes.

Last Updated: February 27, 2018

Categories: SRE

Related

START
How to run a Status Check on a private endpoint using private network integrations
Introduction In this tutorial, we'll show you how to create and run a Status Check to monitor a service hosted on a...
Andre Newman
Technical Marketing Manager

START
How to use Gremlin with Amazon RDS
Amazon RDS is a managed relational database service that lets you easily deploy, scale, and replicate databases. You can...
Andre Newman
Technical Marketing Manager

START
Chaos Engineering using Gremlin on IBM Cloud
Managing the reliability of mission-critical cloud-native applications is a significantly different paradigm than that...
Gremlin
Chaos Engineer