CSCI 5380

Network Virtualization and Orchestration

Lab 10

Kubernetes Networking

University of Colorado Boulder

Department of Computer Science

Network Engineering

Professor Levi Perigo, Ph.D.

# **Summary**

With modern web services, users expect applications to be available 24/7, and developers expect to deploy new versions of those applications several times a day. Containerization helps package software to serve these goals, enabling applications to be released and updated in an easy and fast way without downtime. Kubernetes helps you make sure those containerized applications run where and when you want and helps them find the resources and tools they need to work. [Kubernetes](https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/) is a production-ready, open-source platform designed with Google's accumulated experience in container orchestration, combined with best-of-breed ideas from the community.

A Kubernetes cluster consists of two types of resources:

* Master node: The master coordinates all activities in your cluster, such as scheduling applications, maintaining applications' desired state, scaling applications, and rolling out new updates.
* Worker node: Each node has a Kubelet, which is an agent for managing the node and communicating with the Kubernetes master. The node should also have tools for handling container operations, such as Docker or rkt.

When you deploy applications on Kubernetes, you tell the master to start the application containers. The master schedules the containers to run on the cluster's nodes. The nodes communicate with the master using the Kubernetes API, which the master exposes. A Kubernetes *Pod* is a group of one or more Containers, tied together for the purposes of administration and networking. End users can also use the Kubernetes API directly to interact with the cluster.

To get started with Kubernetes development, you can use *[Minikube](https://github.com/kubernetes/minikube)* or *Kubeadm*. Minikube is a lightweight all-in-one Kubernetes implementation that creates a VM on your local machine and deploys a simple cluster containing only one node. The Minikube CLI provides basic operations for working with your cluster, including start, stop, status, and delete. Kubeadm is used for a multi-node setup where one node acts as Master Node and other nodes act as Worker nodes. Kubeadm tool helps you bootstrap your Kubernetes cluster.

# **Objective 1 – Environment Setup (VM and Kubeadm)**

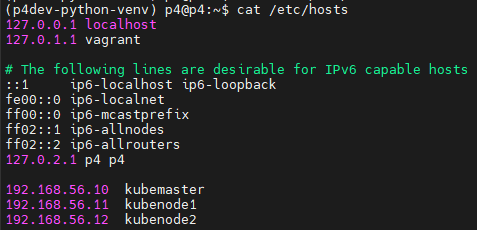
This objective focuses on setting up a multimode Kubernetes Cluster. You are required to follow the requirements mentioned in the “Before you Begin Section” stated [here](https://kubernetes.io/docs/setup/independent/install-kubeadm/). As you follow the below steps, please be sure to check the updated requirements in Kubernetes documentation available [here](https://kubernetes.io/docs/setup/independent/install-kubeadm/#before-you-begin).

**VM Setup:**

We will be using 3 VMs as three different Kubernetes cluster nodes described in Fig. 1. You can use your laptop provided you have 8 GB of RAM (2GB for each VM) and 6-8 CPU cores (2 cores for each VM). Alternatively, you may use your Lab server having relatively new Ubuntu Desktop/Minimal versions (Ubuntu 22.04 is recommended) with Virtualbox as your provider for setting up VMs.

Each VM should have bridged (or NAT) and host-only adapters for Internet and Internal connectivity. Host-only interface should be statically configured (192.168.56.X/24) and should have unique hostnames – “kubemaster” for kubernetes “master: VM 1”, “kubenode1” for kubernetes “worker1: VM 2” and “kubenode2” for kubernetes “worker2: VM 3”. Add entries in the file /etc/hosts on each VM, these new entries on should be the same on all VMs that highlight the host-only IP address and the hostname of all VMs. Additionally, please make sure SSH service is active across all VMs. Disable Swap across all VMs using “swapoff -a” command and comment the *swap* line (typically last line in the file) in /etc/fstab (do not modify any lines in this file as it contains boot information).

* Paste the screenshots showing the content of the file /etc/hosts on each VM after adding new entries. **[10 points]**

Master:  


Worker 1:

A screen shot of a computer

AI-generated content may be incorrect.

Worker 2:

A screen shot of a computer

AI-generated content may be incorrect.

* On each VM, ping the hostname of the other two VMs, and paste the screenshots indicating the ping results. **[10 points]**

Master:

A computer screen shot of a computer program

AI-generated content may be incorrect.

Worker 1:

**A computer screen with white text and numbers

AI-generated content may be incorrect.**

Worker 2:

A computer screen with white text and numbers

AI-generated content may be incorrect.

**Kubeadm Setup:**

Refer to Fig. 1 for high level steps regarding kubeadm installation steps on each VM.

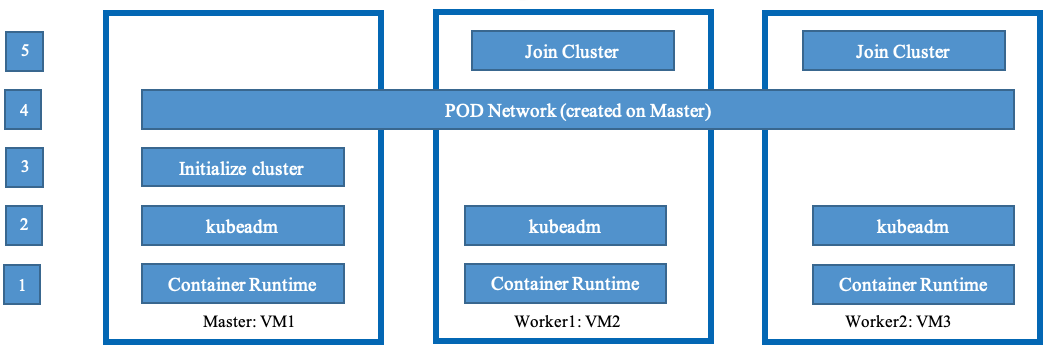


Fig. 1: Installation Steps

**Step 1:**

Before proceeding with Kubeadm setup, install a [container runtime](https://kubernetes.io/docs/setup/production-environment/container-runtimes/) for example containerd (recommended) or Docker across all the VMs.

**Step 2:**

There are plenty of resources online that can help you install Kubeadm across all VMs or you may follow below commands:

sudo apt-get update

sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list

deb https://apt.kubernetes.io/ kubernetes-xenial main

EOF

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

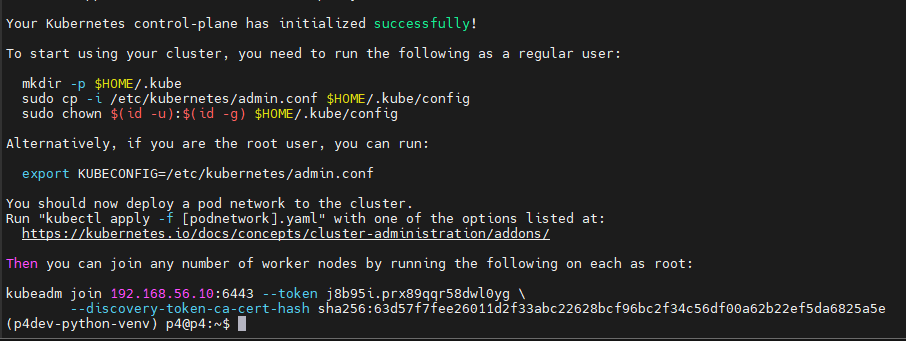
For more information, you may follow this [link](https://kubernetes.io/docs/setup/independent/install-kubeadm/#installing-kubeadm-kubelet-and-kubectl).

**Step 3: Initialize Kubernetes on the Master node (VM 1)**

Run the following command to initialize Kubernetes on the Master node:

kubeadm init --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=<Master Node’s Host-only Static IP>

* Paste the screenshot showing the output of the above command. **[10 points]**

****

Once the “kubeadm init” command completes its execution, run the required commands shown in the output to start using the cluster. The output will show a command in its output (toward the end) that starts with “kubeadm join --token”. Copy and save that command since we will be using it to join worker nodes to the Master node.

Kubeadm join command:

kubeadm join 192.168.56.10:6443 --token 60lot6.8rxic5a4za82ihrh \

--discovery-token-ca-cert-hash sha256:cfb60995900f1484985e15fb57736bfb9077a1bf09afda2ed72c20bf7c67d98c

* Run the command “kubectl cluster-info”, and paste the screenshot of its output. **[5 points]**

A screenshot of a computer

AI-generated content may be incorrect.

**Step 4: Install POD Network**

In this step you’ll setup a POD network so that Kubernetes *pods* can communicate with each other. You may use Calico or Flannel to setup this internal network. In this lab, we’ll be using Calico. On your Master node (VM 1) run following commands to setup [Calico](https://docs.tigera.io/calico/latest/getting-started/kubernetes/quickstart):

kubectl apply -f https://docs.projectcalico.org/v3.11/manifests/calico.yaml

Once a pod network has been installed, you can confirm that it is working by running the command “kubectl get pods --all-namespaces”. Paste the screenshot showing the output of this command. **[5 points]**

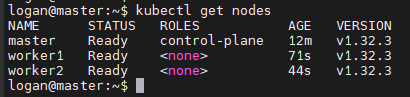
A screen shot of a computer

AI-generated content may be incorrect.

**Step 5: Join Worker nodes (VM 2 and VM 3) to Master node (VM 1)**

Here pastes the command (on VM 2 and VM 3) that you copied in Step 3 from the output of “kubeadm init” command. Once the commands are done executing, run “kubectl get nodes” command on Master node (VM 1) which should show VM 2 and VM 3 in the kubernetes cluster and their status is “Ready”.

* Paste the screenshot showing the output of running the command “kubectl get nodes” on Master node. **[10 points]**



# **Objective 2**

In this objective, you'll create a multi-node Kubernetes cluster of SDN controllers to achieve high-availability (HA) using Kubeadm as shown in the below Fig. 2 and scale the *Pods*. You may choose any SDN controller of your choice.

Kubernetes reference: <https://kubernetes.io/docs/concepts/>

Kubeadm

Master

VM-1

VM-3

VM-2

POD

POD

SDN Controller 2

POD

SDN Controller 1

SDN Controller 3

POD

Kubeadm-Worker 2

Kubeadm-Worker 1

VirtualBox

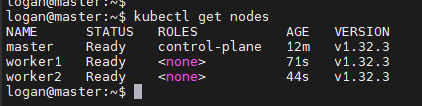
F

**Ubuntu Server**

Figure 2: Multi-node Kubernetes Cluster

Fig. 2: Multi-node Kubernetes cluster of SDN controllers

1. Paste the screenshot indicating number of nodes in the cluster. **[5 points]**



1. Create a Pod consisting of single SDN Controller by writing a YAML definition file of Pod/deployment. Paste the screenshot showing the output of some command indicating the created Pod. **[5 points]**



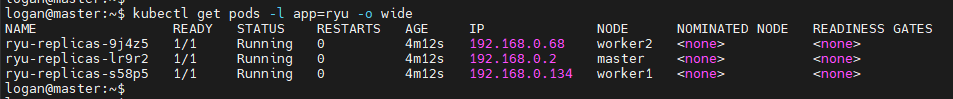


A screen shot of a computer program

AI-generated content may be incorrect.

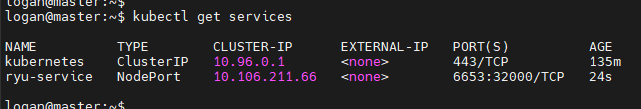
1. Create a ReplicaSet for a cluster of 3 Pods of SDN controllers (one in each Pod) by writing a ReplicaSet definition (YAML) file. Paste the screenshots of some commands indicating the 3 replicas of SDN controller, and the worker node on which each Pod was deployed? **[5 points]**





1. Create a NodePort service by writing a NodePort definition file to expose controller’s IP/Port to make an OpenFlow connection from a Mininet host outside the Kubernetes virtual network (You can run a Mininet agent on your laptop or using the Mininet VM from NetMan). Paste the screenshot of the output of command “kubectl get services” to indicate the created NodePort service. **[10 points]**





1. Create a Mininet network and connect the switches to the sdn controller application running on the 3 pods in the cluster. Paste the screenshot of your command to create the mininet network and the result of the command “pingall” in the mininet network. **[10 points]**

A screen shot of a computer

AI-generated content may be incorrect.

1. Indicate that controller failover works (by shutting down one of the Pods). Did Kubernetes maintain the ReplicaSet number defined in the definition file? Support your answer by pasting the related screenshots. **[5 points]**

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Submit all the required YAML definition files in step 2, 3, 4 along with the lab. **[10 points]**

Included in submission.

# **Extra Credit**

In this objective, you’ll deploy [Prometheus](https://prometheus.io/) and [Grafana](https://grafana.com/) on the Master node (VM1) to monitor and visualize the status of your cluster.

1. Create a Prometheus deployment and a NodePort service to expose Prometheus on some ports. Paste screenshots of the output of the commands “**kubectl get deployment** <-n namespace\_name (if needed)>” and “**kubectl get service** <-n namespace\_name (if needed)>” to indicate your successful creations of Prometheus. **[10 points]**

2. Create a Grafana deployment (set its data source to be your Prometheus server), and a NodePort service to expose Grafana on some ports. Paste screenshots of the output of the commands “**kubectl get deployment** <-n namespace\_name (if needed)>” and “**kubectl get service** <-n namespace\_name (if needed)>” to indicate your successful creations of Grafana. **[10 points]**

3. Now you should be able to access Prometheus and Grafana from your web browser, using any of your Master node IP address and the port you set for the services.

* After open your Prometheus webpage, go to “Status -> Targets”. Select “kubernetes-nodes” from the dropdown menu, it should show that all your nodes – 1 Master node and 2 Worker nodes are up, paste a screenshot of this page. **[5 points]**
* After open your Grafana webpage (admin: admin), import [this Grafana dashboard template](https://grafana.com/grafana/dashboards/8588-1-kubernetes-deployment-statefulset-daemonset-metrics/), set the refresh period to 1 minute. You should then see a chart showing the “Memory usage” of all nodes – 1 Master node and 2 Worker nodes, paste a screenshot of that chart. **[10 points]**

Total Score = \_\_\_\_\_\_\_\_ / 100 + 35