Lesson Plan

# Lab Setup

## Summary

## Introduction to Lab

In this module we will be creating the lab environment that will be used throughout the certification program. As part of the content we will be exploring together, each hands-on module will use this same lab environment. If you would like to take a break, or start and stop between the individual modules or module units, you can. This includes the ability for you to tear the lab down and then easily recreate it to save on resources when you are not working on this course if desired.

We will be creating 4 VMs (3 Kubernetes nodes and 1 standalone host) each with predefined static IP addresses:

- Control (198.19.0.1) - Kubernetes control-plane node
- Node1 (198.19.0.2) - Kubernetes worker node
- Node2 (198.19.0.3) - Kubernetes worker node
- Host1 (198.19.15.254) - General purpose host

(Note that each VM will also have a second dynamically allocated IP address, but in the course we will always use the static IP addresses listed above.)

## Prerequisites

To host the lab you will need a suitable physical machine (e.g. your laptop) or a virtual machine that supports nested virtualization (e.g. Azure VM).

**It is essential that you use the lab orchestrator provided in this course and not your own cluster since the course manifests and material are tailored to this specific environment.**

The suitable platforms for hosting the lab orchestrator listed here are not exhaustive, however they have been found to be the optimal minimum configuration to successfully host the lab. When possible we recommend using your local workstation/laptop to host the lab, but you can also use any cloud provider of your choice that supports nested virtualization. Additionally, if you've found another platform to function well for you please let us know in the **#academy** channel on the Calico Users slack.

## Local Workstation / Laptop

If you would like to use a local workstation, the following platforms have been validated:

- Linux with Snapcraft: https://snapcraft.io/docs/installing-snapd.
- Mac OS X Catalina and Big Sur.
    - Note: Mac OS X Mojave has issues related to networking, and will not result in a successful outcome.
- Windows 10.

The minimum hardware requirements for your local workstation are as follows:

- AMD or Intel Processor (64-bit).
- 4 Cores.
- 12GB of RAM.
- 40GB of free disk/storage on your root drive.

**Note: All lab materials will be obtained from the internet. As such, deployment times may vary depending on the speed of your connection.**

## Cloud Provider

If you would like to use a Cloud Provider, you must ensure that the cloud provider and VM chosen support nested virtualization. The following platforms have been validated:

- Microsoft Azure

The minimum hardware requirements for your cloud provider are as follows:

- Nested Virtualization.
- 4 vCPU.
- 12GB of RAM.

- 128GB of Storage.

Microsoft Azure

The following image / machine types have been validated to complete the certification program.

- Ubuntu Server 18.04 LTS
    - D4s_v3 - 4vCPU / 16G.

## Installing the Lab Orchestrator

To begin we will be installing Multipass. Multipass is a utility from Canonical that allows you to create Ubuntu VMs across a range of platforms in a uniform fashion. We recommend using the latest stable version of Multipass (version 1.5.0 at the time of writing). If any difficulty is encountered deploying the labs - please try this version of Multipass and let us know in the #academy slack channel that you encountered issues with a newer version.

Multipass installation instructions can be found here: https://multipass.run

**Note: If you're running on Windows, you must use the default Hyper-V option.** In addition, note that if you are running an old version of VMware Workstation/Player on Windows, you may find that VMware can not start VMs after using multipass due to its use of Hyper-V. If you experience this, please see the workaround instructions later in the "Managing Your Lab" module.

**Restart your workstation after installing Multipass and before installing the lab. It is essential you do not skip this step.**

## Validating the Lab Orchestrator Installation

After installing multipass and restarting your workstation, the installation can be validated by running the following commands:

```
multipass launch -n test1
```

Once the command is completed, the output should be the following:

```
Launched: test1
```

If your virtual machine did not launch, multipass allows you to inspect information about the instance.

```
multipass info test1
```

The info command for multipass can give information about the test1 instance.

If the instance is still starting, on some platforms multipass has issues starting the VM itself as part of the launch. We can force-start the instance by using the start command.

```
multipass start --all
```

If the VM does not show as having an IP address then there may be an incompatibility between multipass and your host network. If your network is slow then sometimes the multipass launch command can timeout and report failure while downloading the VM image, even though the VM creation is still in progress and may yet succeed.

Once the virtual machine has launched, try to execute a shell for test1.

```
multipass shell test1
```

A “Message of the Day” should display, with a command prompt at the very bottom:

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@test1:~$
```

At this prompt, you can type exit to log-out.

```
exit
```

After the launch is successful of the test1 instance, delete and purge this instance.

```
multipass delete test1
multipass purge
```

## Creating the Lab Environment

To provision the lab environment we will be using cloud-init from Canonical to customize our virtual machines.

### Quick start

If you're on Linux, Mac, or have access to a Bash shell on Windows you can follow these steps to get up and running quickly:

```
curl https://raw.githubusercontent.com/tigera/ccol1/main/control-init.yaml
| multipass launch -n control -m 2048M 20.04 --cloud-init -
curl https://raw.githubusercontent.com/tigera/ccol1/main/node1-init.yaml |
multipass launch -n node1 20.04 --cloud-init -
curl https://raw.githubusercontent.com/tigera/ccol1/main/node2-init.yaml |
multipass launch -n node2 20.04 --cloud-init -
curl https://raw.githubusercontent.com/tigera/ccol1/main/host1-init.yaml |
multipass launch -n host1 20.04 --cloud-init -
```

### Windows

If you're on Windows, in powershell you can follow these steps to get up and running:

```
Invoke-WebRequest -UseBasicParsing
'https://raw.githubusercontent.com/tigera/ccol1/main/control-init.yaml' |
Select-Object -Expand Content | multipass launch -n control -m 2048M 20.04
--cloud-init -
Invoke-WebRequest -UseBasicParsing
'https://raw.githubusercontent.com/tigera/ccol1/main/node1-init.yaml' |
Select-Object -Expand Content | multipass launch -n node1 20.04
--cloud-init -
Invoke-WebRequest -UseBasicParsing
'https://raw.githubusercontent.com/tigera/ccol1/main/node2-init.yaml' |
Select-Object -Expand Content | multipass launch -n node2 20.04
```

```
--cloud-init -
Invoke-WebRequest -UseBasicParsing
'https://raw.githubusercontent.com/tigera/ccol1/main/host1-init.yaml' |
Select-Object -Expand Content | multipass launch -n host1 20.04
--cloud-init -
```

## Starting the Instances

On some platforms, multipass requires you to start the VMs after they have been launched. We can do this by using the multipass start command.

```
multipass start --all
```

Throughout the deployments for the labs, the instances will reboot once provisioning is complete. As a result, you may have to wait a minute until the instance has fully provisioned. A quick way to check the current state of the cluster is to use the multipass list command.

```
multipass list
```

Example output:

```
Name                    State             IPv4             Image
control                 Running           172.17.78.3      Ubuntu 20.04 LTS
host1                   Running           172.17.78.6      Ubuntu 20.04 LTS
node1                   Running           172.17.78.7      Ubuntu 20.04 LTS
node2                   Running           172.17.78.12     Ubuntu 20.04 LTS
```

## Validating the Environment

To validate the lab has successfully started after all four instances we will enter the host1 shell:

```
multipass shell host1
```

Once you reach the command prompt of host1, run kubectl get nodes.

```
kubectl get nodes -A
```

Example output:

```
NAME      STATUS     ROLES    AGE    VERSION
node1     NotReady   <none>   4m44s  v1.18.10+k3s1
node2     NotReady   <none>   2m48s  v1.18.10+k3s1
control   NotReady   master   6m36s  v1.18.10+k3s1
```

Note the “NotReady” status. This is because we have not yet installed a CNI plugin to provide the networking.

The instance we will be using for the following labs will be host1 unless otherwise specified. Think of host1 as your primary entry point into the kubernetes ecosystem, with the other instances acting as the cluster in the cloud.