Kubernetes Setup Using Ansible and Vagrant

Intissar EL AHMER

08/02/2021

M2 – Data Science

Objective

This report post describes the steps required to setup a multi node Kubernetes cluster for development purposes. This setup provides a production-like cluster that can be setup on your local machine.

Why do we require multi node cluster setup?

Multi node Kubernetes clusters offer a production-like environment which has various advantages. Even though Minikube provides an excellent platform for getting started, it doesn't provide the opportunity to work with multi node clusters which can help solve problems or bugs that are related to application design and architecture. Minikube provides a local instance of a single-node kubernetes install, allowing you to configure and interact with it as you would a full cluster.

For instance, Ops can reproduce an issue in a multi node cluster environment, Testers can deploy multiple versions of an application for executing test cases and verifying changes. These benefits enable teams to resolve issues faster which make them more agile.

Why use Vagrant and Ansible?

Vagrant is a tool that will allow us to create a virtual environment easily and it eliminates pitfalls that cause the works-on-my-machine phenomenon. It can be used with multiple providers such as Oracle VirtualBox, VMware, Docker, and so on. It allows us to create a disposable environment by making use of configuration files.

A vagrant is a tool to automatically create and configure portable and playable virtual machines.One of the things in favor of Vagrant, against other DevOps tools like Docker, is that any computer scientist/programmer/developer (even those who use Windows) understands it the first time: Vagrant is an automation of deployment that will configure and automate the creation of virtual machines.

Vagrant is a tool to manage virtual machine environments, and allows you to configure and use reproducible work environments on top of various virtualization and cloud platforms. It also has integration with Ansible as a provisioner for these virtual machines, and the two tools work together well.

Ansible is an infrastructure automation engine that automates software configuration management. It is agentless and allows us to use SSH keys for connecting to remote machines. Ansible playbooks are written in yaml and offer inventory management in simple text files Ansible performs the installation of the required packages for kubernetes in the nodes.

Ansible is an open-source automation tool, or platform, used for IT tasks such as configuration management, application deployment, intraservice orchestration, and provisioning. Automation is crucial these days, with IT environments that are too complex and often need to scale too quickly for system administrators and developers to keep up if they have to do everything manually. Automation simplifies complex tasks, not just making developers' jobs more manageable but allowing them to focus attention on other tasks that add value to an organization. In other words, it frees up time and increases efficiency. And Ansible, as noted above, is rapidly rising to the top in the world of automation tools. Let's look at some of the reasons for Ansible's popularity.

What is Vagrantfile?

The primary function of the Vagrantfile is to describe the type of machine required for a project, and how to configure and provision these machines.
The Vagrantfile is the main configuration file that builds the nodes (infrastructure) and installs an ssh key allowing access to them without a password.

The syntax of Vagrantfiles is Ruby, but knowledge of the Ruby programming language is not necessary to make modifications to the Vagrantfile, since it is mostly a simple variable assignment. In fact, Ruby is not even the most popular community Vagrant is used within, which should help show you that despite not having Ruby knowledge, people are very successful with Vagrant.

Vagrant vs Kubernetes?

Kubernetes vs Vagrant, are primarily classified as "Container" and "Virtual Machine Management" tools respectively. Kubernetes is an open source orchestration system for Docker containers. It handles scheduling onto nodes in a compute cluster and actively manages workloads to ensure that their state matches the users declared intentions. On the other hand, Vagrant is detailed as " A tool for building and distributing development environments ". Kubernetes is a container orchestration system that manages containers at scale. Initially developed by Google based on its experience running containers in production, Kubernetes is open source and actively developed by a community around the world.

What is Kubernetes?

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

The name Kubernetes originates from Greek, meaning helmsman or pilot. Google open-sourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.

Prerequisites

- Vagrant should be installed on your machine. (www.vagrantup.com)
- Oracle VirtualBox can be used as a Vagrant provider ($ sudo apt install virtualbox) or make use of similar providers as described in Vagrant's official documentation.Vagrant can work with multiple OS virtualisation providers. Vagrant and VirtualBox integration works flawlessly and it is free.
- Ansible should be installed in your machine to automate installation of OS, Docker & Kubernetes dependencies quickly on multiple VMs. Refer to the Ansible

installation guide for platform specific installation. ($sudo apt install ansible)

Setup overview

We will be setting up a Kubernetes cluster that will consist of one master and two worker nodes. All the nodes will run Ubuntu Xenial 64-bit OS and Ansible playbooks will be used for provisioning.

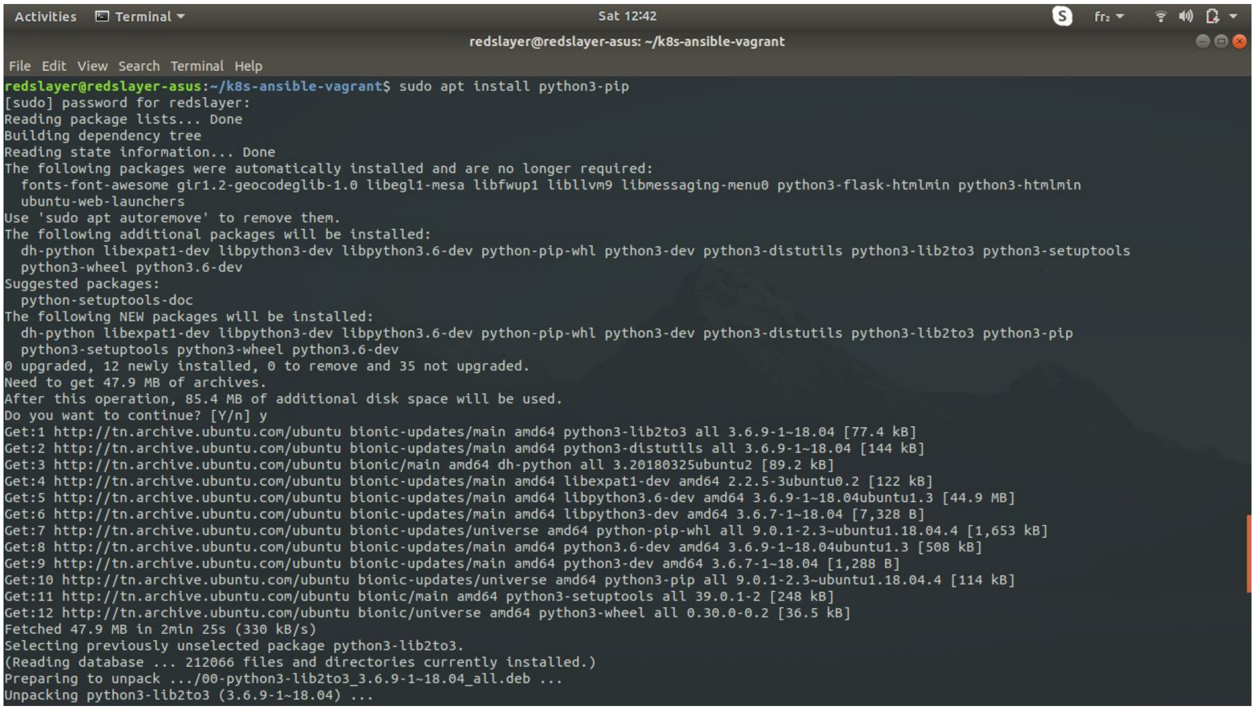
One master node

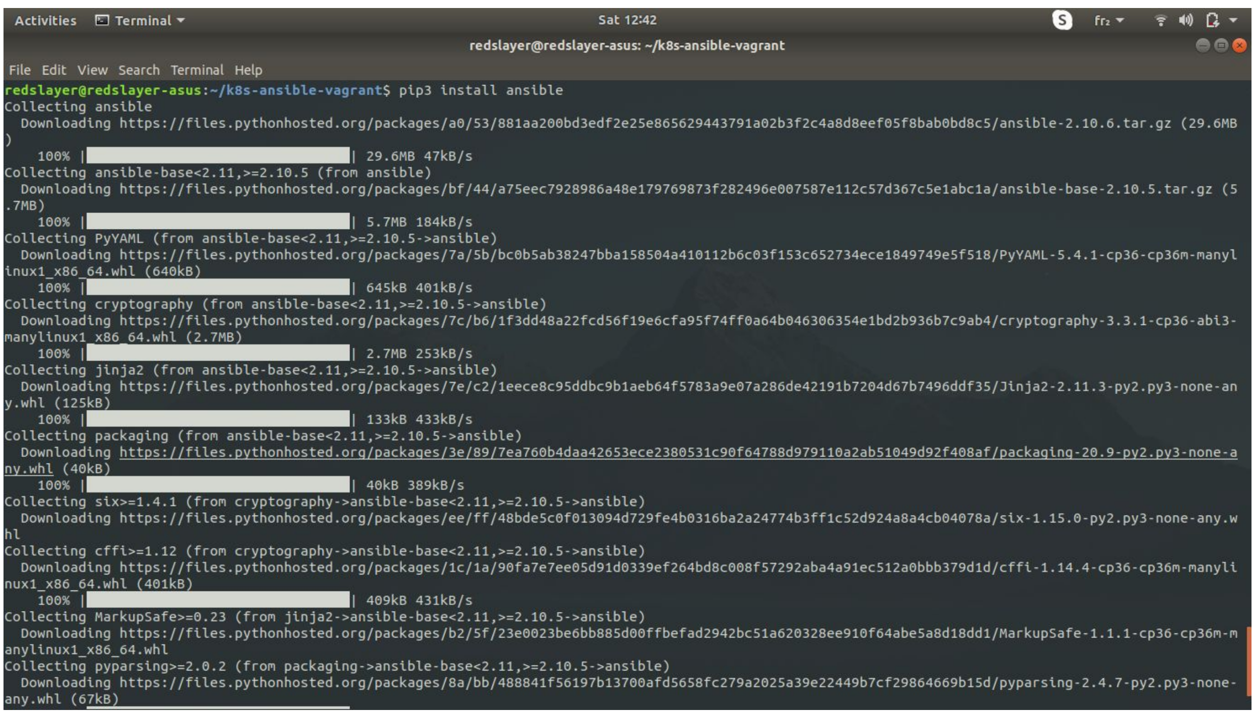
The master node (a *node* in Kubernetes refers to a server) is responsible for managing the state of the cluster. It runs Etcd, which stores cluster data among components that schedule workloads to worker nodes.

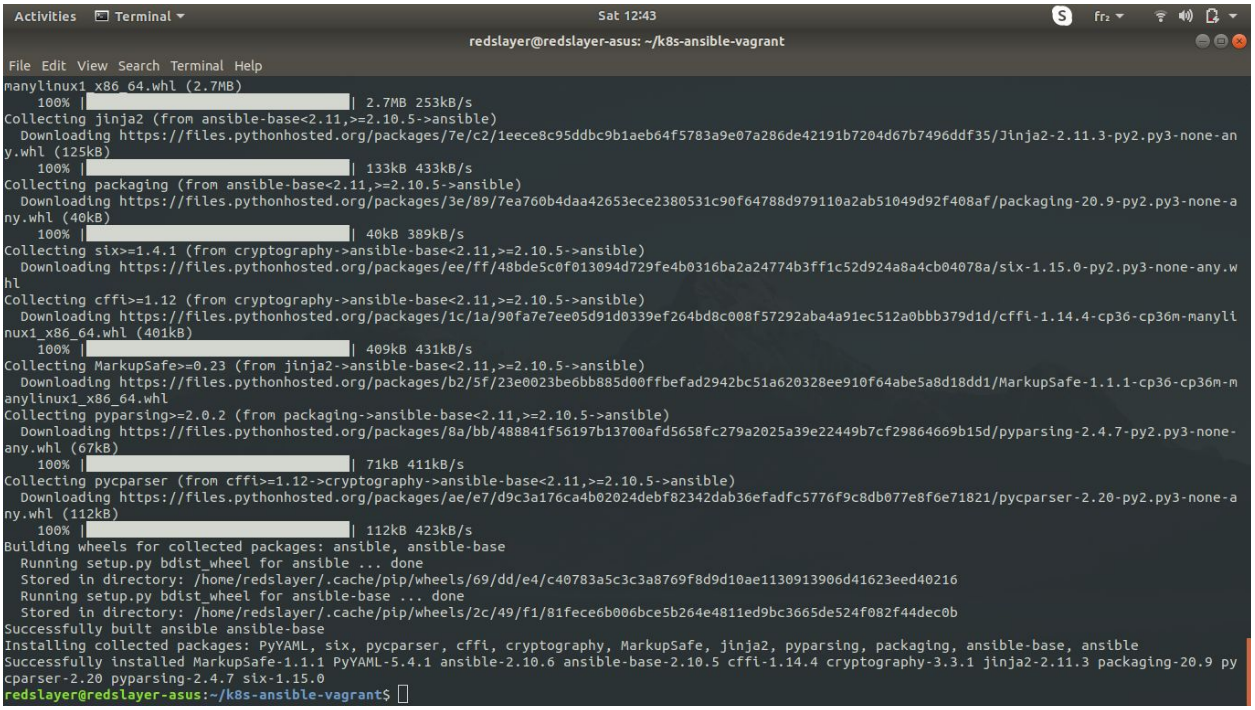
Two worker nodes

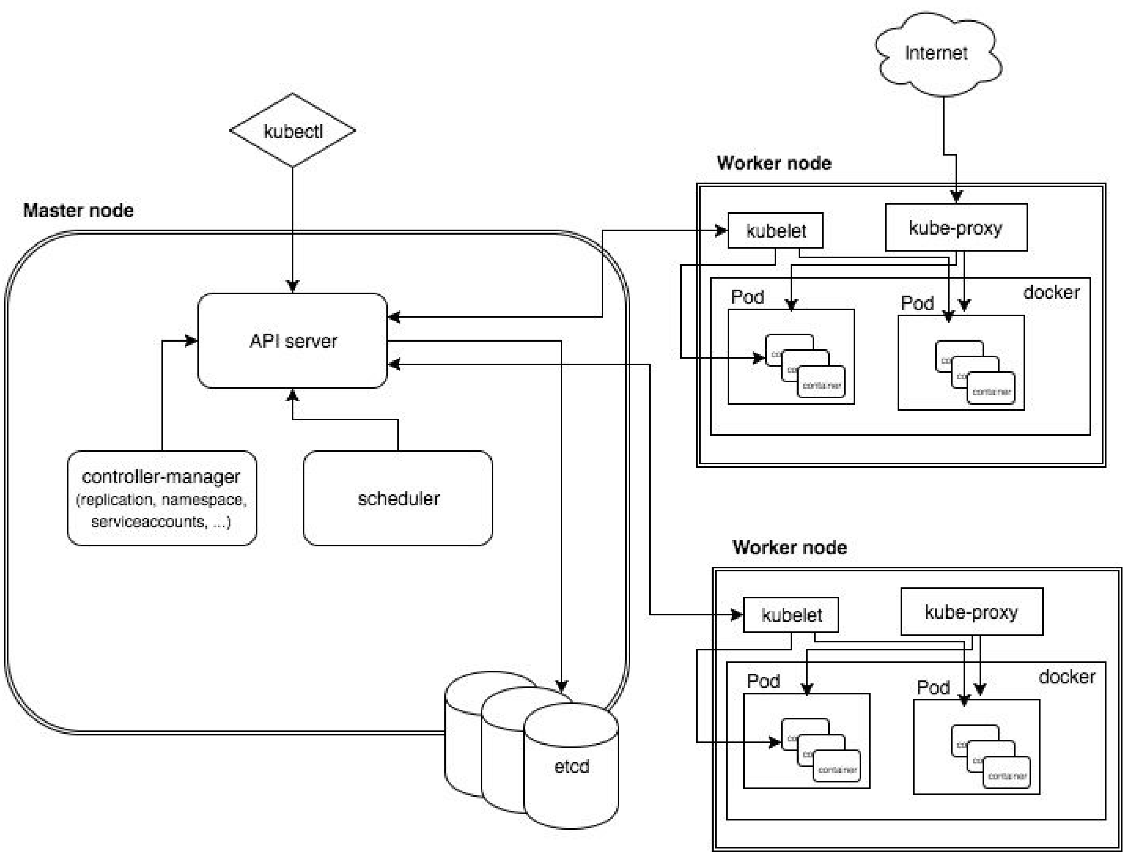
Worker nodes are the servers where your *workloads* (i.e. containerized applications and services) will run. A worker will continue to run your workload once they're assigned to it, even if the master goes down once scheduling is complete. A cluster's capacity can be increased by adding workers.

After completing this guide, you will have a cluster ready to run containerized applications, provided that the servers in the cluster have sufficient CPU and RAM resources for your applications to consume. Almost any traditional Unix application including web applications, databases, daemons, and command line tools can be containerized and made to run on the cluster.


Activities
Terminal
Sat 12:42
redslayer@redslayer-asus: ~/k8s-ansible-vagrant
File Edit View Search Terminal Help
redslayer@redslayer-asus:~/k8s-ansible-vagrant$ sudo apt install python3-pip
[sudo] password for redslayer:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
fonts-font-awesome gir1.2-geocodeglib-1.0 libegl1-mesa libfwup1 libllvm9 libmessaging-menu0 python3-flask-htmlmin python3-htmlmin
ubuntu-web-launchers
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
dh-python libexpat1-dev libpython3-dev libpython3.6-dev python-pip-whl python3-dev python3-distutils python3-lib2to3 python3-setuptools
python3-wheel python3.6-dev
Suggested packages:
python-setuptools-doc
The following NEW packages will be installed:
dh-python libexpat1-dev libpython3-dev libpython3.6-dev python-pip-whl python3-dev python3-distutils python3-lib2to3 python3-pip
python3-setuptools python3-wheel python3.6-dev
0 upgraded, 12 newly installed, 0 to remove and 35 not upgraded.
Need to get 47.9 MB of archives.
After this operation, 85.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://tn.archive.ubuntu.com/ubuntu bionic-updates/main amd64 python3-lib2to3 all 3.6.9-1~18.04 [77.4 kB]
Get:2 http://tn.archive.ubuntu.com/ubuntu bionic-updates/main amd64 python3-distutils all 3.6.9-1~18.04 [144 kB]
Get:3 http://tn.archive.ubuntu.com/ubuntu bionic/main amd64 dh-python all 3.20180325ubuntu2 [89.2 kB]
Get:4 http://tn.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libexpat1-dev amd64 2.2.5-3ubuntu0.2 [122 kB]
Get:5 http://tn.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libpython3.6-dev amd64 3.6.9-1~18.04ubuntu1.3 [44.9 MB]
Get:6 http://tn.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libpython3-dev amd64 3.6.7-1~18.04 [7,328 B]
Get:7 http://tn.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 python-pip-whl all 9.0.1-2.3~ubuntu1.18.04.4 [1,653 kB]
Get:8 http://tn.archive.ubuntu.com/ubuntu bionic-updates/main amd64 python3.6-dev amd64 3.6.9-1~18.04ubuntu1.3 [508 kB]
Get:9 http://tn.archive.ubuntu.com/ubuntu bionic-updates/main amd64 python3-dev amd64 3.6.7-1~18.04 [1,288 B]
Get:10 http://tn.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 python3-pip all 9.0.1-2.3~ubuntu1.18.04.4 [114 kB]
Get:11 http://tn.archive.ubuntu.com/ubuntu bionic/main amd64 python3-setuptools all 39.0.1-2 [248 kB]
Get:12 http://tn.archive.ubuntu.com/ubuntu bionic/universe amd64 python3-wheel all 0.30.0-0.2 [36.5 kB]
Fetched 47.9 MB in 2min 25s (330 kB/s)
Selecting previously unselected package python3-lib2to3.
(Reading database ... 212066 files and directories currently installed.)
Preparing to unpack .../00-python3-lib2to3_3.6.9-1~18.04_all.deb ...
Unpacking python3-lib2to3 (3.6.9-1~18.04) ...


Activities
Terminal
Sat 12:42
redslayer@redslayer-asus: ~/k8s-ansible-vagrant
File Edit View Search Terminal Help
redslayer@redslayer-asus:~/k8s-ansible-vagrant$ pip3 install ansible
Collecting ansible
Downloading https://files.pythonhosted.org/packages/a0/53/881aa200bd3edf2e25e865629443791a02b3f2c4a8d8eef05f8bab0bd8c5/ansible-2.10.6.tar.gz (29.6MB)
100% | 29.6MB 47kB/s
Collecting ansible-base<2.11,>=2.10.5 (from ansible)
Downloading https://files.pythonhosted.org/packages/bf/44/a75eec7928986a48e179769873f282496e007587e112c57d367c5e1abc1a/ansible-base-2.10.5.tar.gz (5.7MB)
100% | 5.7MB 184kB/s
Collecting PyYAML (from ansible-base<2.11,>=2.10.5->ansible)
Downloading https://files.pythonhosted.org/packages/7a/5b/bc0b5ab38247bba158504a410112b6c03f153c652734ece1849749e5f518/PyYAML-5.4.1-cp36-cp36m-manylinux1_x86_64.whl (640kB)
100% | 645kB 401kB/s
Collecting cryptography (from ansible-base<2.11,>=2.10.5->ansible)
Downloading https://files.pythonhosted.org/packages/7c/b6/1f3dd48a22fcd56f19e6cfa95f74ff0a64b046306354e1bd2b936b7c9ab4/cryptography-3.3.1-cp36-abi3-manylinux1_x86_64.whl (2.7MB)
100% | 2.7MB 253kB/s
Collecting jinja2 (from ansible-base<2.11,>=2.10.5->ansible)
Downloading https://files.pythonhosted.org/packages/7e/c2/1eece8c95ddbc9b1aeb64f5783a9e07a286de42191b7204d67b7496ddf35/Jinja2-2.11.3-py2.py3-none-any.whl (125kB)
100% | 133kB 433kB/s
Collecting packaging (from ansible-base<2.11,>=2.10.5->ansible)
Downloading https://files.pythonhosted.org/packages/3e/89/7ea760b4daa42653ece2380531c90f64788d979110a2ab51049d92f408af/packaging-20.9-py2.py3-none-any.whl (40kB)
100% | 40kB 389kB/s
Collecting six>=1.4.1 (from cryptography->ansible-base<2.11,>=2.10.5->ansible)
Downloading https://files.pythonhosted.org/packages/ee/ff/48bde5c0f013094d729fe4b0316ba2a24774b3ff1c52d924a8a4cb04078a/six-1.15.0-py2.py3-none-any.whl
Collecting cffi>=1.12 (from cryptography->ansible-base<2.11,>=2.10.5->ansible)
Downloading https://files.pythonhosted.org/packages/1c/1a/90fa7e7ee05d91d0339ef264bd8c008f57292aba4a91ec512a0bbb379d1d/cffi-1.14.4-cp36-cp36m-manylinux1_x86_64.whl (401kB)
100% | 409kB 431kB/s
Collecting MarkupSafe>=0.23 (from jinja2->ansible-base<2.11,>=2.10.5->ansible)
Downloading https://files.pythonhosted.org/packages/b2/5f/23e0023be6bb885d00ffbefad2942bc51a620328ee910f64abe5a8d18dd1/MarkupSafe-1.1.1-cp36-cp36m-manylinux1_x86_64.whl
Collecting pyparsing>=2.0.2 (from packaging->ansible-base<2.11,>=2.10.5->ansible)
Downloading https://files.pythonhosted.org/packages/8a/bb/488841f56197b13700afd5658fc279a2025a39e22449b7cf29864669b15d/pyparsing-2.4.7-py2.py3-none-any.whl (67kB)





High level Kubernetes architecture showing* a *cluster with a master and two

worker nodes

Before creating all the files, it should look like below.

```
Activities   Terminal ▾                Sat 12:28
redslayer@redslayer-asus: ~/k8s-ansible-vagrant/kubernetes-setup
File  Edit  View  Search  Terminal  Help
redslayer@redslayer-asus:~/k8s-ansible-vagrant$ sudo apt  install tree
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-font-awesome gir1.2-geocodeglib-1.0 libegl1-mesa libfwup1 libllvm9 libmessaging-menu0
  python3-flask-htmlmin python3-htmlmin ubuntu-web-launchers
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  tree
0 upgraded, 1 newly installed, 0 to remove and 35 not upgraded.
Need to get 40.7 kB of archives.
After this operation, 105 kB of additional disk space will be used.
Get:1 http://tn.archive.ubuntu.com/ubuntu bionic/universe amd64 tree amd64 1.7.0-5 [40.7 kB]
Fetched 40.7 kB in 0s (107 kB/s)
Selecting previously unselected package tree.
(Reading database ... 212059 files and directories currently installed.)
Preparing to unpack .../tree_1.7.0-5_amd64.deb ...
Unpacking tree (1.7.0-5) ...
Setting up tree (1.7.0-5) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
redslayer@redslayer-asus:~/k8s-ansible-vagrant$ tree
.
├── kubernetes-setup
│   ├── master-playbook.yml
│   └── node-playbook.yml
└── Vagrantfile

1 directory, 3 files
```

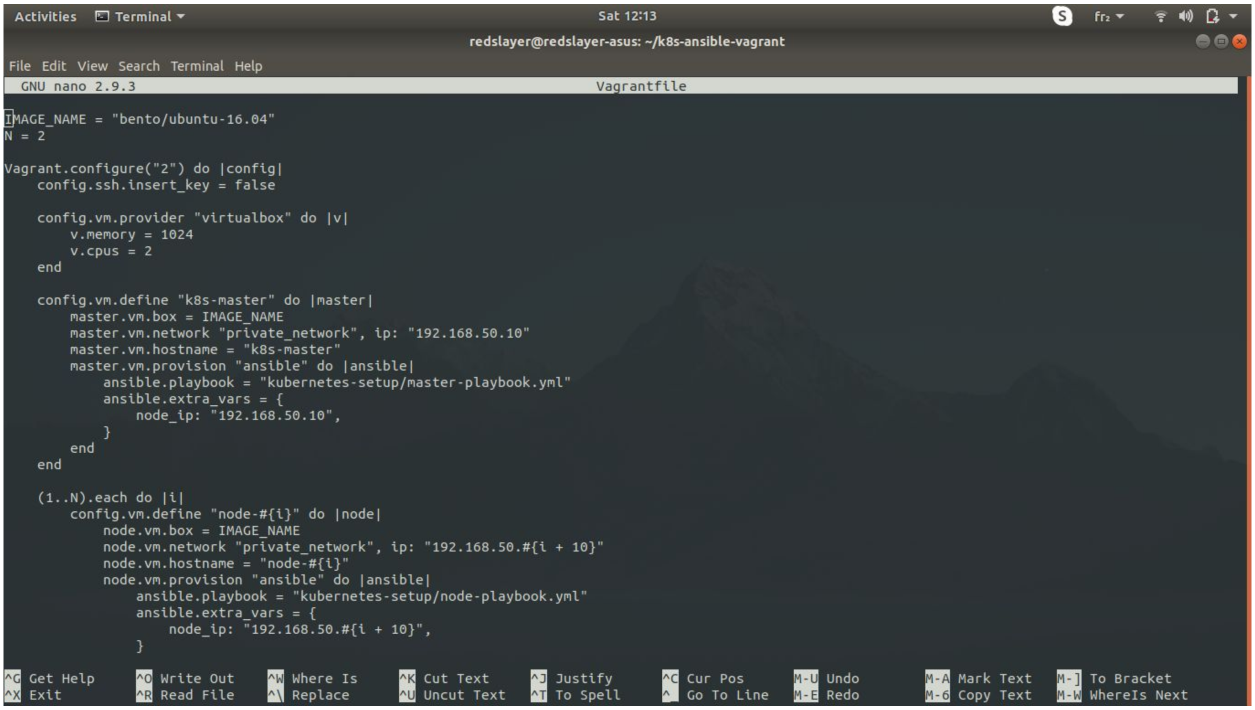
Step 1: Creating a Vagrantfile

Use the text editor of your choice and create a file named Vagrantfile, inserting the code below. The value of N denotes the number of nodes present in the cluster, it can be modified accordingly. In the below example, we are setting the value of N as 2.

At a minimum you need a master node and one worker node. But a single worker is of little use so I recommend at least 2 workers. Master and workers must be able to see each other, using stable IPs. Note the Vagrantfile also installs an hosts file (located in the same directory), as follows:

```
redslayer@redslayer-asus:~/k8s-ansible-vagrant$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       redslayer-asus

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
redslayer@redslayer-asus:~/k8s-ansible-vagrant$
```



Step 2: Create an Ansible playbook for Kubernetes master.

Create a directory named kubernetes-setup in the same directory as the Vagrantfile. Create two files named master-playbook.yml and node-playbook.yml in the directory kubernetes-setup.

In the file master-playbook.yml, add the code below.

Step 2.1: Install Docker and its dependent components.

Docker - a container runtime. It is the component that runs your containers. Support for other runtimes such as rkt is under active development in Kubernetes.

We will be installing the following packages, and then adding a user named “vagrant” to the “docker” group.

- docker-ce
- docker-ce-cli
- containerd.io

```
GNU nano 2.9.3                         master-playbook.yml

---
- hosts: all
  become: true
  tasks:
  - name: Install packages that allow apt to be used over HTTPS
    apt:
      name: "{{ packages }}"
      state: present
      update_cache: yes
    vars:
      packages:
      - apt-transport-https
      - ca-certificates
      - curl
      - gnupg-agent
      - software-properties-common

  - name: Add an apt signing key for Docker
    apt_key:
      url: https://download.docker.com/linux/ubuntu/gpg
      state: present

  - name: Add apt repository for stable version
    apt_repository:
      repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable
      state: present

  - name: Install docker and its dependecies
    apt:
      name: "{{ packages }}"
      state: present
      update_cache: yes
    vars:
```

```
      - docker-ce
      - docker-ce-cli
      - containerd.io
    notify:
      - docker status

  - name: Add vagrant user to docker group
    user:
      name: vagrant
      group: docker

  - name: Remove swapfile from /etc/fstab
    mount:
      name: "{{ item }}"
      fstype: swap
      state: absent
    with_items:
      - swap
      - none

  - name: Disable swap
    command: swapoff -a
    when: ansible_swaptotal_mb > 0

  - name: Add an apt signing key for Kubernetes
    apt_key:
      url: https://packages.cloud.google.com/apt/doc/apt-key.gpg
      state: present

  - name: Adding apt repository for Kubernetes
    apt_repository:
      repo: deb https://apt.kubernetes.io/ kubernetes-xenial main
      state: present
```

```yaml
- name: Add an apt signing key for Docker
  apt_key:
    url: https://download.docker.com/linux/ubuntu/gpg
    state: present

- name: Add apt repository for stable version
  apt_repository:
    repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable
    state: present

- name: Install docker and its dependecies
  apt:
    name: "{{ packages }}"
    state: present
    update_cache: yes
  vars:
    packages:
    - docker-ce
    - docker-ce-cli
    - containerd.io
  notify:
    - docker status

- name: Add vagrant user to docker group
  user:
    name: vagrant
    group: docker
```

Step 2.2: Kubelet will not start if the system has swap enabled, so we are disabling swap using the below code.

The Kubelet is the main Kubernetes service since it manages Docker according to Kubernetes rules. kubelet - a system service/program that runs on all nodes and handles node-level operations.

```yaml
- name: Remove swapfile from /etc/fstab
  mount:
    name: "{{ item }}"
    fstype: swap
    state: absent
  with_items:
    - swap
    - none

- name: Disable swap
  command: swapoff -a
  when: ansible_swaptotal_mb > 0
```

Step 2.3: Installing kubelet, kubeadm and kubectl using the below code.

We also need two command line clients: kubectl, the command line Kubernetes client, and kubeadm, the command line installer.

kubectl - a CLI tool used for issuing commands to the cluster through its API Server.

kubeadm init will expose a server (the apiserver) to other services at a given IP.

kubeadm - a CLI tool that will install and configure the various components of a cluster in a standard way and it will run for a while. It installs:

- etcd storing cluster configurations
- apiserver answering to `kubectl` requests
- controller managing the cluster
- scheduler deciding where containers should be placed

- dns for implementing service discovery
- proxy for locating services in the cluster

```
- name: Add an apt signing key for Kubernetes
  apt_key:
    url: https://packages.cloud.google.com/apt/doc/apt-key.gpg
    state: present

- name: Adding apt repository for Kubernetes
  apt_repository:
    repo: deb https://apt.kubernetes.io/ kubernetes-xenial main
    state: present
    filename: kubernetes.list

- name: Install Kubernetes binaries
  apt:
    name: "{{ packages }}"
    state: present
    update_cache: yes
  vars:
    packages:
      - kubelet
      - kubeadm
      - kubectl

- name: Configure node ip
  lineinfile:
    path: /etc/default/kubelet
    line: KUBELET_EXTRA_ARGS=--node-ip={{ node_ip }}
```

```
- name: Restart kubelet
  service:
    name: kubelet
    daemon_reload: yes
    state: restarted
```

Step 2.3: Initialize the Kubernetes cluster with kubeadm using the below code (applicable only on master node).

```
- name: Initialize the Kubernetes cluster using kubeadm
  command: kubeadm init --apiserver-advertise-address="192.168.50.10" --apiserver-cert-extra-sans="192.168.50.10"  --node-name k8s-master --p$
```

Step 2.4: Setup the kube config file for the vagrant user to access the Kubernetes cluster using the below code.

```
- name: Setup kubeconfig for vagrant user
  command: "{{ item }}"
  with_items:
  - mkdir -p /home/vagrant/.kube
  - cp -i /etc/kubernetes/admin.conf /home/vagrant/.kube/config
  - chown vagrant:vagrant /home/vagrant/.kube/config
```

Step 2.5: Setup the container networking provider and the network policy engine using the below code.

```
- name: Install calico pod network
  become: false
  command: kubectl create -f https://docs.projectcalico.org/v3.4/getting-started/kubernetes/installation/hosted/calico.yaml
```

Step 2.6: Generate kube join command for joining the node to the Kubernetes cluster and store the command in the file named join-command.

```
- name: Generate join command
  command: kubeadm token create --print-join-command
  register: join_command

- name: Copy join command to local file
  local_action: copy content="{{ join_command.stdout_lines[0] }}" dest="./join-command"
```

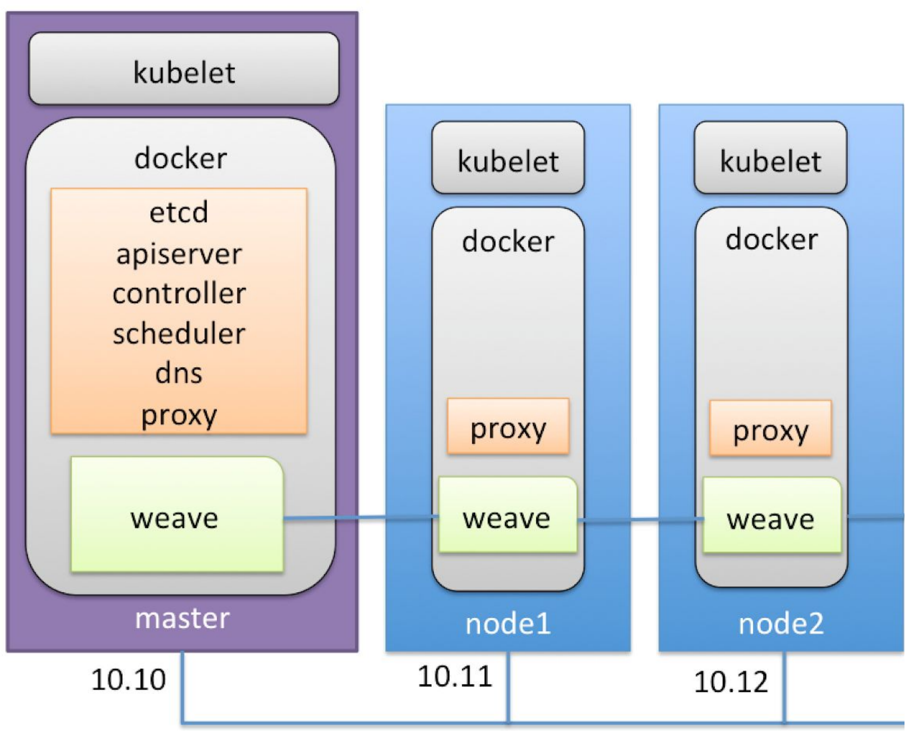
Step 2.7: Setup a handler for checking Docker daemon using the below code.

Docker is the essential service because Kubernetes is an orchestrator of Docker containers, so everything is built on top of Docker.

```
handlers:
  - name: docker status
    service: name=docker state=started
```

Step 3: Create the Ansible playbook for Kubernetes node.

Once the master is ready and nodes have installed the required software, we can complete the setup just executing kubeadm in each node with the join command, in order to complete the configuration.



Create a file named node-playbook.yml in the directory kubernetes-setup.

Add the code below into node-playbook.yml

Step 3.1: Start adding the code from Steps 2.1 to 2.3.

Step 3.2: Join the nodes to the Kubernetes cluster using below code.

```
- name: Copy the join command to server location
  copy: src=join-command dest=/tmp/join-command.sh mode=0777

- name: Join the node to cluster
  command: sh /tmp/join-command.sh
```

Step 3.3: Add the code from step 2.7 to finish this playbook.

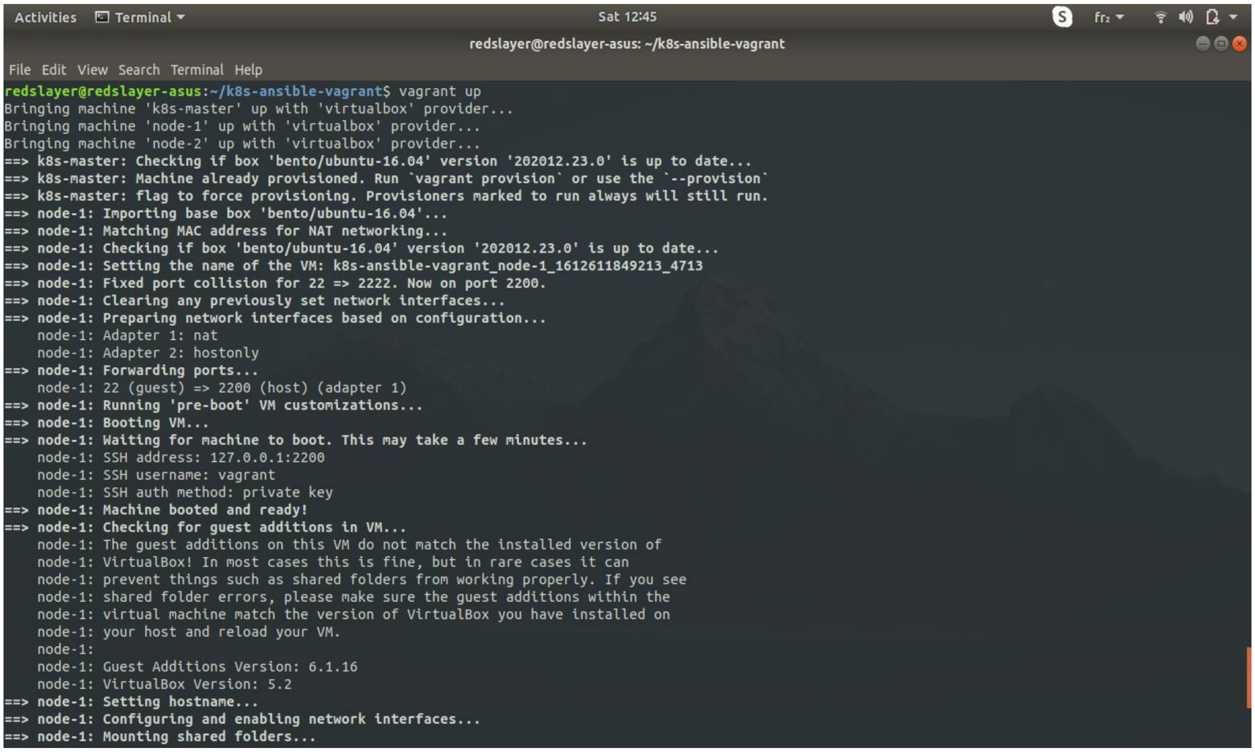
Step 4: Upon completing the Vagrantfile and playbooks follow the below steps.

Start the vagrant box-
$vagrant up will bring up all the nodes -

```
$ cd /path/to/Vagrantfile
$ vagrant up
```

For the first time, depending on your internet bandwidth, it may take about 10-30 minutes.

You may see similar output:



Once a provider is installed, you can use it by calling vagrant up with the --provider flag. This will force Vagrant to use that specific provider. No other configuration is necessary!

In normal day-to-day usage, the --provider flag is not necessary since Vagrant can usually pick the right provider for you. More details on how it does this is below.

$ vagrant up --provider=vmware_fusion

To re-run a playbook on an existing VM, just run:
$ vagrant provision

It's also easy to tear down the cluster using Vagrant's destroy command.

$ vagrant destroy -f will destroy the cluster.

Note that Vagrantfile is recreated whenever create-cluster.sh is executed. Any manual

changes made to Vagrantfile will be lost!

Upon completion of all the above steps, the Kubernetes cluster should be up and running. We can login to the master or worker nodes using Vagrant as follows:

```
$ ## Accessing master
$ vagrant ssh k8s-master
vagrant@k8s-master:~$ kubectl get nodes
NAME         STATUS   ROLES    AGE     VERSION
k8s-master   Ready    master   18m     v1.13.3
node-1       Ready    <none>   12m     v1.13.3
node-2       Ready    <none>   6m22s   v1.13.3

$ ## Accessing nodes
$ vagrant ssh node-1
$ vagrant ssh node-2
```

Step5: Generate SSH key for ansible (only need to run on ansible node).

Generate SSH key (during the ssh key generation it might ask for a passphrase so either you create a new passphrase or leave it empty)- You do not have to provide username and password every time you login/ssh into the other nodes.

```
redslayer@redslayer-asus:~/k8s-ansible-vagrant$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/redslayer/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/redslayer/.ssh/id_rsa.
Your public key has been saved in /home/redslayer/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Bi1+wG3VHbYudEv9IPB1GFufRX19QBkxMB87EweeuQY redslayer@redslayer-asus
The key's randomart image is:
+---[RSA 2048]----+
|          o.o+#@O|
|      . o . o.*+#O|
|       = +   E #.=|
|      . =   . * *.|
|       . S   . = .|
|        o     o   |
|                 |
|                 |
|                 |
+----[SHA256]-----+
redslayer@redslayer-asus:~/k8s-ansible-vagrant$
```

Once complete, check status of the VM’s with vagrant status

$ vagrant status

Step6: Verify the kubernetes nodes.

Let's check the nodes' status in our final step.

```
kubectl get nodes



NAME    STATUS   ROLES    AGE   VERSION
node1   Ready    master   13m   v1.18.2
node2   Ready    master   13m   v1.18.2
```

In master-playbook.yml i add --ignore-preflight-errors all : a list of checks whose errors will be shown as warnings.

```
- name: Initialize the Kubernetes cluster using kubeadm

command: kubeadm init --apiserver-advertise-address="192.168.50.10"
--apiserver-cert-extra-sans="192.168.50.10"  --node-name k8s-master
--pod-network-cidr=192.168.0.0/16 --ignore-preflight-errors all
```

A pod is an atomic unit that runs one or more containers. These containers share resources such as file volumes and network interfaces in common. Pods are the basic unit of scheduling in Kubernetes: all containers in a pod are guaranteed to run on the same node that the pod is scheduled on.

Each pod has its own IP address, and a pod on one node should be able to access a pod on another node using the pod's IP. Containers on a single node can communicate easily through a local interface. Communication between pods is more complicated, however, and requires a separate networking component that can transparently route traffic from a pod on one node to a pod on another.