# Solution M2: Container Orchestration

## Task 1

We are going to create a 2-node cluster

Before we begin, we must create one machine with the following parameters:

* 2 CPU
* 2 GB RAM
* 10 GB HDD
* 1 network adapter set to NAT or NAT network
* *1 network adapter set to host only mode*

### Create the VM template

We must install the **OS**, which in our case will be **Ubuntu 20.04**, and do some fine-tuning

Of course, instead of manual installation, we will use a generic template with clean installation

* Once we have the machine ready, we must log on and start doing some work
* First, make sure that the auto-update routines are turned off

**sudo systemctl disable --now apt-daily-upgrade.timer**

**sudo systemctl disable --now apt-daily.timer**

**sudo systemctl disable --now unattended-upgrades.service**

**sudo sed -i s/1/0/g /etc/apt/apt.conf.d/20auto-upgrades**

* Update package info

**sudo apt-get update**

* Install required system packages

**sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common**

* Install **Docker** official GPG key

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -**

* Add the **Docker** repository

**sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"**

* Update package info

**sudo apt-get update**

* Install **Docker**

**sudo apt-get install docker-ce docker-ce-cli containerd.io**

* Modify Docker configuration

**cat << EOF | sudo tee /etc/docker/daemon.json**

**{**

**"exec-opts": ["native.cgroupdriver=systemd"]**

**}**

**EOF**

* Reload and restart the daemon

**sudo systemctl daemon-reload**

**sudo systemctl restart docker**

**sudo systemctl enable docker**

* Install **Kubernetes** official GPG key

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

* Add **Kubernetes** repository

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

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

**EOF**

* Update package info

**sudo apt-get update**

* Install **Kubernetes** packages

**sudo apt-get install -y kubelet kubeadm kubectl**

* Just in any case change the following system setting:

**cat << EOF | sudo tee /etc/sysctl.d/k8s.conf**

**net.bridge.bridge-nf-call-ip6tables = 1**

**net.bridge.bridge-nf-call-iptables = 1**

**EOF**

* Apply the changes

**sudo sysctl --system**

* One last step – turn off the swap:

**sudo swapoff -a**

**sudo sed -i '/swap/ s/^/#/' /etc/fstab**

* Now we are ready to turn off the machine and clone it:

**sudo shutdown now**

### Clone VM

* Create two clones
* Do not forget to tick the option to reinitialize the **MAC** address
* Last, but not least, choose full clone

### Final touches

* Start all two machines
* Set the hostname of all three to **node-X.k8s** where **X** is a number between **1** and **2**
* *Enable the second network adapter and set its* ***IP*** *address to* ***192.168.89.10X****, where* ***X*** *is between* ***1*** *and* ***2***
* Add records for all two hosts in the **/etc/hosts** file

### Initialize the Kubernetes master (control-plane) node

* Initiate the initialization process on the master node:

**sudo kubeadm init --apiserver-advertise-address=192.168.89.101 --pod-network-cidr 10.244.0.0/16**

* Once the initialization process is over, in order to configure our environment, we must execute:

**mkdir -p $HOME/.kube**

**sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config**

**sudo chown $(id -u):$(id -g) $HOME/.kube/config**

* Remove the schedule taint from the node:

**kubectl taint nodes --all node-role.kubernetes.io/master-**

### Install plugin for the pod network

We will install the **Calico** (<https://docs.projectcalico.org/getting-started/kubernetes/quickstart>) pod network plugin

* First, will install the **Tigera Calico** operator and custom resource definitions

**kubectl create -f https://docs.projectcalico.org/manifests/tigera-operator.yaml**

* Then, we will download the custom resource definition

**wget https://docs.projectcalico.org/manifests/custom-resources.yaml**

* Next, we will edit the **custom-resources.yaml** file and will make sure that the CIDR matches ours. We can use:

**sed -i 's/192.168.0.0\/16/10.244.0.0\/16/g' custom-resources.yaml**

* Finally, we will create the resource

**kubectl create -f custom-resources.yaml**

### Join the worker node

* On the worker node execute the generated **join** command:

**sudo kubeadm join 192.168.89.101:6443 --token ydgfpy.zkebb0osukiwcsft \**

**--discovery-token-ca-cert-hash sha256:9708e0227c1a173f5c113e3a4d09999b508004234c65e7503bdb155665ddecae**

* Once ready, we can check the status of the cluster by executing on the master this:

**kubectl get nodes**

We are done. We have a cluster of two nodes and all of them can run pods

## Task 2

With the following **Vagrantfile** (available as a separate file) we can solve the task:

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

$common = <<SCRIPT

echo "\* Disable auto-update timers and service if present ..."

systemctl disable --now apt-daily-upgrade.timer &> /dev/null || true

systemctl disable --now apt-daily.timer &> /dev/null || true

systemctl disable --now unattended-upgrades.service &> /dev/null || true

sed -i s/1/0/g /etc/apt/apt.conf.d/20auto-upgrades || true

echo "\* Add hosts ..."

echo "192.168.99.101 k8s1.dof.lab k8s1" >> /etc/hosts

echo "192.168.99.102 k8s2.dof.lab k8s2" >> /etc/hosts

echo "Install the required system packages ..."

apt-get update -y

apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-properties-common

echo "Install Docker official GPG key ..."

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -

echo "Add the Docker repository ..."

add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

echo "Install Docker ..."

apt-get update -y

apt-get install -y docker-ce docker-ce-cli containerd.io

echo "Modify Docker configuration ..."

cat << EOF | tee /etc/docker/daemon.json

{

  "exec-opts": ["native.cgroupdriver=systemd"]

}

EOF

systemctl daemon-reload

systemctl restart docker

systemctl enable docker

echo "Install Kubernetes official GPG key ..."

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

echo "Add Kubernetes repository ..."

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

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

EOF

echo "Install Kubernetes packages ..."

apt-get update -y

apt-get install -y kubelet kubeadm kubectl

echo "Change system settings ..."

cat << EOF | tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip\_forward = 1

EOF

sysctl --system

echo "Turn off the swap ..."

swapoff -a

sed -i '/swap/ s/^/#/' /etc/fstab

echo "\* Add vagrant user to docker group ..."

usermod -aG docker vagrant

SCRIPT

Vagrant.configure(2) do |config|

  config.ssh.insert\_key = false

  config.vm.provider "virtualbox" do |v|

    v.memory = 2048

    v.cpus = 2

  end

  config.vm.define "k8s1" do |k8s1|

    k8s1.vm.box = "shekeriev/ubuntu-20-04-server"

    k8s1.vm.hostname = "k8s1.dof.lab"

    k8s1.vm.network "private\_network", ip: "192.168.99.101"

    k8s1.vm.synced\_folder "vagrant/", "/vagrant"

    k8s1.vm.provision "shell", inline: $common

    k8s1.vm.provision "shell", inline: <<EOS

echo "\* Initialize Kubernetes cluster ..."

kubeadm init --apiserver-advertise-address=192.168.99.101 --pod-network-cidr 10.244.0.0/16

echo "\* Copy configuration for root ..."

mkdir -p /root/.kube

cp -i /etc/kubernetes/admin.conf /root/.kube/config

chown root:root /root/.kube/config

echo "\* Copy configuration for vagrant ..."

mkdir -p /home/vagrant/.kube

cp -i /etc/kubernetes/admin.conf /home/vagrant/.kube/config

chown vagrant:vagrant /home/vagrant/.kube/config

echo " Remove the schedule taint from the node ..."

kubectl taint nodes --all node-role.kubernetes.io/master-

echo "\* Install POD network plugin (Calico) ..."

kubectl create -f https://docs.projectcalico.org/manifests/tigera-operator.yaml

wget https://docs.projectcalico.org/manifests/custom-resources.yaml -O /tmp/custom-resources.yaml

sed -i 's/192.168.0.0/10.244.0.0/g' /tmp/custom-resources.yaml

kubectl create -f /tmp/custom-resources.yaml

echo "\* Install Dashboard ..."

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.3/aio/deploy/recommended.yaml

echo "\* Create Dashboard admin user ..."

cat << EOF > /vagrant/dashboard-admin-user.yml

apiVersion: v1

kind: ServiceAccount

metadata:

  name: admin-user

  namespace: kubernetes-dashboard

EOF

echo "\* Create Dashboard admin user role ..."

cat << EOF > /vagrant/dashboard-admin-role.yml

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

  name: admin-user

roleRef:

  apiGroup: rbac.authorization.k8s.io

  kind: ClusterRole

  name: cluster-admin

subjects:

- kind: ServiceAccount

  name: admin-user

  namespace: kubernetes-dashboard

EOF

echo "\* Add both the user and role ..."

kubectl apply -f /vagrant/dashboard-admin-user.yml

kubectl apply -f /vagrant/dashboard-admin-role.yml

echo "\* Save the user token ..."

kubectl -n kubernetes-dashboard describe secret $(kubectl -n kubernetes-dashboard get secret | grep admin-user | awk '{print $1}') > /vagrant/admin-user-token.txt

echo "\* Create custom token ..."

kubeadm token create abcdef.1234567890abcdef

echo "\* Save the hash to a file ..."

openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | openssl dgst -sha256 -hex | sed 's/^.\* //' > /vagrant/hash.txt

EOS

  end

  config.vm.define "k8s2" do |k8s2|

    k8s2.vm.box = "shekeriev/ubuntu-20-04-server"

    k8s2.vm.hostname = "k8s2.dof.lab"

    k8s2.vm.network "private\_network", ip: "192.168.99.102"

    k8s2.vm.synced\_folder "vagrant/", "/vagrant"

    k8s2.vm.provision "shell", inline: $common

    k8s2.vm.provision "shell", inline: <<EOS

echo "\* Join the worker node (k8s2) ..."

kubeadm join 192.168.99.101:6443 --token abcdef.1234567890abcdef --discovery-token-ca-cert-hash sha256:`cat /vagrant/hash.txt`

EOS

  end

end