# Solution M5: HashiCorp Tools

Bellow you can find just one possible solution to the challenges in the homework assignment

Please note, that because both tasks are related, there is just one set of files

## Task 1: Nomad Cluster

Even though it is not considered a good practice, we can create a Nomad cluster where all nodes are acting both as server and client

A key resource to help us achieve the desired result, can be found here: <https://www.nomadproject.io/docs/configuration>

The solution has many files, but the most important ones are the **Vagrantfile** and **Nomad’s** configuration file

There is also a set of **Consul**-related files

All files are included as an accompanying compressed archive

In its final version the **Vagrantfile** has the following content:

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

$script = <<SCRIPT

echo "\* Install dependencies ..."

sudo dnf install -y unzip curl wget vim

echo "\* Turn off the firewall ..."

sudo systemctl disable --now firewalld

echo "\* Turn off SELinux ..."

sudo setenforce 0

echo "\* Fetching Consul ..."

CONSUL\_VERSION=1.10.2

cd /tmp/

curl -sSL https://releases.hashicorp.com/consul/${CONSUL\_VERSION}/consul\_${CONSUL\_VERSION}\_linux\_amd64.zip > consul.zip

echo "\* Installing Consul ..."

unzip consul.zip

sudo chmod +x consul

sudo mv consul /usr/bin/consul

sudo mkdir -p /etc/consul.d

sudo chmod a+w /etc/consul.d

sudo mkdir -p /opt/consul

sudo chmod a+w /opt/consul

echo "\* Fetching Nomad ..."

NOMAD\_VERSION=1.1.4

cd /tmp/

curl -sSL https://releases.hashicorp.com/nomad/${NOMAD\_VERSION}/nomad\_${NOMAD\_VERSION}\_linux\_amd64.zip -o nomad.zip

echo "\* Installing Nomad ..."

unzip nomad.zip

sudo chmod +x nomad

sudo mv nomad /usr/bin/nomad

sudo mkdir -p /etc/nomad.d

sudo chmod a+w /etc/nomad.d

sudo mkdir -p /opt/nomad

sudo chmod a+w /opt/nomad

echo "\* Set hostname's IP to made advertisement Just Work"

sudo sed -i -e "s/.\*server.\*/$(ip route get 192.168.50 | awk '{print $NF;exit}') $(hostname)/" /etc/hosts

sudo sed -i -e "s/.\*client.\*/$(ip route get 192.168.50 | awk '{print $NF;exit}') $(hostname)/" /etc/hosts

SCRIPT

$docker = <<DOCKER

echo "\* Install BASH Completion ..."

sudo dnf install -y bash-completion

echo "\* Add Docker repository ..."

sudo dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

echo "\* Install Docker ..."

sudo dnf install -y docker-ce docker-ce-cli

echo "\* Adjust Docker configuration ..."

sudo mkdir -p /etc/docker

echo '{ "hosts": ["tcp://0.0.0.0:2375", "unix:///var/run/docker.sock"] }' | sudo tee /etc/docker/daemon.json

sudo mkdir -p /etc/systemd/system/docker.service.d/

echo [Service] | sudo tee /etc/systemd/system/docker.service.d/docker.conf

echo ExecStart= | sudo tee -a /etc/systemd/system/docker.service.d/docker.conf

echo ExecStart=/usr/bin/dockerd | sudo tee -a /etc/systemd/system/docker.service.d/docker.conf

echo "\* Enable and start Docker ..."

sudo systemctl enable --now docker

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

sudo usermod -aG docker vagrant

echo "\* Install Docker Compose ..."

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

echo "\* Add BASH Completion for Docker Compose ..."

sudo curl -L https://raw.githubusercontent.com/docker/compose/1.29.2/contrib/completion/bash/docker-compose -o /etc/bash\_completion.d/docker-compose

DOCKER

$nomad = <<NOMAD

echo "\* Set vm.max\_map\_count to a higher value ..."

sudo sysctl -w vm.max\_map\_count=262144

echo "\* Copying template configuration files ..."

sudo cp /vagrant/nomad/server.hcl /etc/nomad.d/server.hcl

echo "\* Adjust the configuration ..."

sudo sed -i -e "s/%SERVER\_ADDRESS%/$(ip route get 192.168.50 | awk '{print $(NF-2);exit}')/" /etc/nomad.d/server.hcl

sudo sed -i -e "s/%CLIENT\_INTERFACE%/$(ip route | grep 192.168.50 | cut -d ' ' -f 3)/" /etc/nomad.d/server.hcl

echo "\* Starting nomad agent ..."

sudo bash -c "nomad agent -config=/etc/nomad.d/server.hcl > /tmp/nomad 2>&1 &"

NOMAD

$consulsrv = <<CONSULSRV

echo "\* Copying template configuration files ..."

sudo cp /vagrant/consul/server.json /etc/consul.d/server.json

echo "\* Adjust the configuration ..."

sudo sed -i -e "s/%SERVER\_ADDRESS%/$(ip route get 192.168.50 | awk '{print $(NF-2);exit}')/" /etc/consul.d/server.json

echo "\* Starting consul agent ..."

sudo bash -c "consul agent -config-dir /etc/consul.d > /tmp/consul 2>&1 &"

CONSULSRV

$consulclnt = <<CONSULCLNT

echo "\* Copying template configuration files ..."

sudo cp /vagrant/consul/client.json /etc/consul.d/client.json

echo "\* Adjust configuration of Consul ..."

sudo sed -i -e "s/%CLIENT\_ADDRESS%/$(ip route get 192.168.50 | awk '{print $(NF-2);exit}')/" /etc/consul.d/client.json

echo "\* Reload services definitions ..."

sudo systemctl daemon-reload

echo "\* Starting consul agent ..."

sudo bash -c "consul agent -config-dir /etc/consul.d -advertise $(ip route get 192.168.50 | awk '{print $(NF-2);exit}') > /tmp/consul 2>&1 &"

CONSULCLNT

Vagrant.configure("2") do |config|

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

    vb.memory = "3072"

  end

  config.vm.box = "shekeriev/centos-8-minimal"

  config.vm.synced\_folder ".", "/vagrant", type: "virtualbox"

  config.vm.provision "shell", inline: $script, privileged: false

  config.vm.provision 'shell', inline: $docker, privileged: false

  config.vm.define 'node1' do |node1|

    node1.vm.hostname = 'node1'

    node1.vm.network 'private\_network', ip: '192.168.50.2'

    node1.vm.provision "shell", inline: $consulsrv, privileged: false

    node1.vm.provision "shell", inline: $nomad, privileged: false

  end

  config.vm.define 'node2' do |node2|

    node2.vm.hostname = 'node2'

    node2.vm.network 'private\_network', ip: '192.168.50.3'

    node2.vm.provision 'shell', inline: $consulclnt, privileged: false

    node2.vm.provision 'shell', inline: $nomad, privileged: false

  end

  config.vm.define 'node3' do |node3|

    node3.vm.hostname = 'node3'

    node3.vm.network 'private\_network', ip: '192.168.50.4'

    node3.vm.provision 'shell', inline: $consulclnt, privileged: false

    node3.vm.provision 'shell', inline: $nomad, privileged: false

  end

end

And the **Nomad** configuration contains the following:

bind\_addr = "%SERVER\_ADDRESS%" # the default

addresses {

    http = "%SERVER\_ADDRESS%"

    rpc = "%SERVER\_ADDRESS%"

    serf = "%SERVER\_ADDRESS%"

}

# Advertise an accessible IP address so the server is reachable by other servers

# and clients. The IPs can be materialized by Terraform or be replaced by an

# init script.

advertise {

    http = "%SERVER\_ADDRESS%:4646"

    rpc = "%SERVER\_ADDRESS%:4647"

    serf = "%SERVER\_ADDRESS%:4648"

}

# Increase log verbosity

log\_level = "DEBUG"

# Setup data dir

data\_dir = "/opt/nomad"

# Enable the server

server {

  enabled = true

  # Self-elect, should be 3 or 5 for production

  bootstrap\_expect = 3

}

# Enable the client

client {

  enabled = true

  network\_interface = "%CLIENT\_INTERFACE%"

}

# Enable Docker volumes

plugin "docker" {

  config {

    volumes {

      enabled = true

    }

  }

}

As usual, we start the environment with:

**vagrant up**

Once all machines are up and running, we can use a **Nomad** client either on our host, or installed on one of the nodes (for example node 1), and check the status of the cluster with:

**nomad server members -address="http://192.168.50.2:4646"**

We can ease ourselves by exporting an environment variable containing the address of one of the nodes:

**export NOMAD\_ADDR=http://192.168.50.2:4646**

Then the former command will become:

**nomad server members**

For the commands that follow, we will assume that the environment variable has been set correctly

The list of client nodes can be seen by executing this:

**nomad node status**

## Task 2: Workload on Nomad Cluster

In order to reuse the infrastructure created in Task 1, we must copy the contents of the **site** folder to every node in the cluster

This should be done automatically during the execution of the **vagrant up** command. The folder must appear on every node as **/vagrant/site**

If this is not the case, then we can copy it manually on every node with:

**scp -r site vagrant@<node-ip>:.**

Please note, that the job expects the **site** folder to be in **/vagrant/site**

If not, you should adjust the path

You should add/adjust the following variable in the **index.php** file

   $host = $\_ENV['NOMAD\_IP\_db'];

Then, we can check the information on how to use **Docker** with **Nomad** on this URL:

<https://www.nomadproject.io/docs/drivers/docker.html>

Should we need a reference for **Nomad Jobs**, we can use this URL:

<https://www.nomadproject.io/docs/job-specification/job.html>

Finally, we can create a **homework.nomad** job file with the following content:

job "homework" {

  datacenters = ["dc1"]

  type = "service"

  group "homework" {

    count = 1

    network {

      port "http" {

        static = 80

      }

      port "db" {

        static = 3306

      }

    }

    task "dob-mysql" {

      driver = "docker"

      config {

        image = "shekeriev/dob-w3-mysql"

        ports = ["db"]

      }

      env {

        MYSQL\_ROOT\_PASSWORD = "12345"

      }

      resources {

        cpu    = 500 # 500 MHz

        memory = 256 # 256MB

      }

      service {

        name = "dob-mysql"

        tags = ["global", "db"]

        port = "db"

        check {

          name     = "alive"

          type     = "tcp"

          interval = "10s"

          timeout  = "2s"

        }

      }

    }

    task "dob-php" {

      driver = "docker"

      config {

        image = "shekeriev/dob-w3-php"

        ports = ["http"]

        mounts = [

          {

            type = "bind"

            target = "/var/www/html"

            source = "/vagrant/site"

            readonly = true

          }

        ]

      }

      resources {

        cpu    = 500 # 500 MHz

        memory = 256 # 256MB

      }

      service {

        name = "dob-php"

        tags = ["global", "web"]

        port = "http"

        check {

          name     = "alive"

          type     = "tcp"

          interval = "10s"

          timeout  = "2s"

        }

      }

    }

  }

}

We can check what will be executed with:

**nomad plan homework.nomad**

And then execute the job with:

**nomad job run homework.nomad**