# Practice M8: Exam Prep – Option 4

During this practice we will assume that we are working in Linux environment. It could be a physical machine or a virtual one. The distribution of choice is not that important, but it will be better to stick to some of the well supported distributions.

Most of the steps can be executed in Windows and/or macOS environment as well, either directly or in a VM.

The infrastructure can be built on-premise or in the cloud (AWS, GCP, Azure). Any combination of tools is acceptable.

## Assignment

Prepare a clustered environment to host a containerized web application. Application’s images must be built in an automated way – check every 3 minutes and if there are changes, build the images, publish them, and then re-deploy them. The cluster should be monitored – a dashboard, showing the utilization must be created.

## Possible Solution

We will implement one possible solution. It will include the following set of technologies:

* it will be hosted locally in VirtualBox
* the infrastructure and cluster will be built with Vagrant + Shell Scripts
* the cluster will be based on Nomad and Consul and will have one master and two worker nodes
* for monitoring we will use Elastic Stack
* logs and metrics will be collected with Filebeat and Metricbeat
* Jenkins and Docker will be used for image building

#### Infrastructure & Cluster

For this part, we will base our solution on the **Vagrantfile** created during **Part 3** of **Module 5**. Instead of just modifying it, we will build an entirely new version. It can be seen in the corresponding folder.

Let’s start the process of exploration and creation:

* Go to folder **M8/M8-4** if not there already and examine **Vagrantfile** file
* If you see it applicable, do some changes
* Once ready, execute:

**vagrant up**

* Wait until everything is up and running. This can take between 15 and 20 minutes
* Meanwhile you can install **nomad** binary locally if you don’t have it already:

**curl -sSL https://releases.hashicorp.com/nomad/0.8.7/nomad\_0.8.7\_linux\_amd64.zip -o nomad.zip**

**unzip nomad.zip**

**sudo chmod +x nomad**

**sudo mv nomad /usr/bin/nomad**

* Now, once the infrastructure is up and running, try to connect and ask for information:

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

**nomad node status -address="http://192.168.50.2:4646"**

* Then, we can open two browser tabs, the first for **Nomad**, pointing to:

**http://192.168.50.2:4646/ui**

* And a second one, for the **Consul** part of the cluster, pointing to:

**http://192.168.50.2:8500/ui**

#### Monitoring

Now that we have a working infrastructure, it is time to take care of the monitoring part. For this task we must deploy an **Elastic Stack**. We will reuse the images created and published during the first part of this practice, but instead of **Kubernetes** applications we will develop a **Nomad** job files.

***// Under development…***

#### CD/CI Preparation

We will deploy **Jenkins** in our **Nomad** cluster. For this purpose, we will use a two-component solution with master and slave. The master portion will be deployed as **Nomad** job:

* Being in the **M8/M8-4** folder, let’s examine the **jenkins/jenkins-master.nomad** file
* Adjust its content as you see fit
* Once ready with the adjustments, let’s aks for the plan of the job:

**nomad plan -address="http://192.168.50.2:4646" jenkins/jenkins-master.nomad**

* Now, we are ready to run the job:

**nomad job run -address="http://192.168.50.2:4646" jenkins/jenkins-master.nomad**

* We can check in the Web UI of both Consul and Nomad what we have so far

Now that we have a Jenkins master up and running, we must configure it a bit:

* First, let’s check what is the URL of **Jenkins**:

**curl http://192.168.50.2:8500/v1/catalog/service/jenkins-master?pretty=true**

* Open a browser tab and navigate the IP in the **ServiceAddress** field and use the port **8080**
* Go to **Manage Jenkins > Configure System**
* Scroll down, in the **Cloud** section click on **Add new a new cloud**, and select **Nomad**
* In the **Name** field enter something unique, like jenkins-nomad-master
* In the **Nomad URL** field enter the IP of our master node - <http://192.168.50.2:4646>
* Click on Test connection
* Then you can click on Save if you do not want or do not need to do additional changes. In our case, we will click on Add in the Slave Templates section
* Then we will fill the following (you can experiment with the values):
  + CPU – enter for example **300**
  + Memory – **256** should be fine
  + Labels – enter **jenkins-nomad-slave**
  + Disk – **300**
  + Priority – **50**
  + Usage – select **Only build jobs with label expressions matching this node**
  + Workspace root – enter **/home/builder**
  + Image – enter **shekeriev/jenkins-nomad-slave:latest** or other Jenkins slave image
  + Click on **Save**

#### CD/CI – build and apply – with Git, Jenkins and Docker

We will create a similar job, as we did in the last section of M8 Exam Prep Practice:

* Go to **New Item**
* Select **Pipeline** for type
* In the **Enter an item name** set some name, for example **Pipeline-Docker-Nomad**
* In the pipeline script enter (or copy it from the **M8/M8-4/jenkins/jenkins-pipeline.txt** file):

**def imagetag = new Date().format('yyyyMMdd.HHmmss')**

**def image = "shekeriev/nomad-jenkins:${imagetag}"**

**def docker\_node = "tcp://192.168.50.3"**

**def nomad\_master = "http://192.168.50.2:4646"**

**def reg\_user = "YOUR-USER-NAME"**

**def reg\_pass = "YOUR-SECURE-PASS"**

**node("jenkins-nomad-slave")**

**{**

**stage("Build Docker image")**

**{**

**git 'https://github.com/shekeriev/simple-docker-image.git'**

**sh "DOCKER\_HOST=${docker\_node} docker build -t ${image} ."**

**}**

**stage ("Push to Docker Hub")**

**{**

**sh "DOCKER\_HOST=${docker\_node} docker login -u ${reg\_user} -p ${reg\_pass}"**

**sh "DOCKER\_HOST=${docker\_node} docker push ${image}"**

**}**

**stage ("Apply the changes with Nomad")**

**{**

**sh "sed 's/%DOCKER-IMAGE%/${imagetag}/g' -i nomad/amazing-web-app.nomad"**

**sh "nomad job run --address='${nomad\_master}' nomad/amazing-web-app.nomad"**

**}**

**}**

* Don’t forget to adjust the values to your situation. For example set/change the user name and password for the Docker registry
* Click **Save**
* Click on **Build Now**
* Check the result either on the command line and/or in the browser

#### Clean up

Don’t forget to clean up. With this approach, there is just one step required 😊

**vagrant destroy --force**