# Solution M4: Jenkins

This is one possible and fully automated solution of the tasks included in the homework

All steps that follow assume that we decided to base our solution on **Debian 11** box

Please note, that this solution is intentionally far from being an optimal one

Here, the emphasis is put on readability and not on optimization or speed

You must adjust some values like IP addresses, image (or box) names, repository names, credentials, etc. to match your use case

## Environment

Prepare the environment by creating a **Vagrantfile** with the following content

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

Vagrant.configure(2) do |config|

  config.ssh.insert\_key = false

  config.vm.define "jdhost" do |jdhost|

    jdhost.vm.box="shekeriev/debian-11"

    jdhost.vm.hostname = "jdhost.do1.lab"

    jdhost.vm.provider :virtualbox do |vb|

      vb.customize ["modifyvm", :id, "--memory", "4096"]

    end

    jdhost.vm.network "private\_network", ip: "192.168.99.100"

    jdhost.vm.network "forwarded\_port", guest: 8080, host: 8080

    jdhost.vm.provision "shell", path: "install\_common.sh"

    jdhost.vm.provision "shell", path: "install\_docker.sh"

    jdhost.vm.provision "shell", path: "install\_jenkins.sh"

  end

end

We will need to create a **install\_common.sh** file to install and configure the necessary packages

#!/bin/bash

echo "\* Update repositories and install common packages"

apt-get update

apt-get install -y ca-certificates curl gnupg lsb-release fontconfig openjdk-11-jre git

In addition, we will create a file named **install\_docker.sh** to prepare and install **Docker**

#!/bin/bash

echo "\* Add Docker repository key"

curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo "\* Add Docker repository"

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian $(lsb\_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null

echo "\* Install Docker"

apt-get update

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

echo "\* Adjust group membership"

usermod -aG docker vagrant

And a file named **install\_jenkins.sh** to prepare and install **Jenkins**

#!/bin/bash

echo "\* Add Jenkins repository key"

curl -fsSL https://pkg.jenkins.io/debian/jenkins.io.key | tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo "\* Add Jenkins repository"

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian binary/ | tee /etc/apt/sources.list.d/jenkins.list > /dev/null

echo "\* Install Jenkins"

apt-get update

apt-get install -y jenkins

echo "\* Adjust jenkins user"

usermod -s /bin/bash jenkins

echo -e 'Password1\nPassword1' | passwd jenkins

echo "\* Adjust group membership"

usermod -aG docker jenkins

echo "\* Restart the service to reflect the mmembership change"

systemctl restart jenkins

echo "\* admin password is:"

cat /var/lib/jenkins/secrets/initialAdminPassword

Of course, we could create just one file, or use inline shell for example

So far, we have the base infrastructure

We can bring it up with

**vagrant up**

Then open [**http://192.168.99.100:8080**](http://192.168.99.100:8080)in a browser

Create the initial administrator user. For example, **admin** with password **Password1**

Install the default set of plugins

We are ready to continue with the pipeline

## Jenkins Pipeline

Navigate to **Dashboard** and click **New Item**

Enter **Homework** as name, select **Pipeline** and click **OK**

Use the following script for the pipeline

pipeline

{

    agent any

    stages

    {

        stage('Clone the project')

        {

            steps

            {

                sh 'rm -rf bgapp || true'

                sh 'git clone https://github.com/shekeriev/bgapp.git'

            }

        }

        stage('Prepare the network')

        {

            steps

            {

                sh 'docker network ls | grep appnet || docker network create appnet'

            }

        }

        stage('Build the web image')

        {

            steps

            {

                sh 'cd bgapp && docker image build -t img-web -f Dockerfile.web .'

            }

        }

        stage('Run the web component')

        {

            steps

            {

                sh 'docker container rm -f web || true'

                sh 'docker container run -d --name web --net appnet -p 9090:80 -v $(pwd)/bgapp/web:/var/www/html:ro img-web'

            }

        }

        stage('Build the db image')

        {

            steps

            {

                sh 'cd bgapp && docker image build -t img-db -f Dockerfile.db .'

            }

        }

        stage('Run the db component')

        {

            steps

            {

                sh 'docker container rm -f db || true'

                sh 'docker container run -d --name db --net appnet -e MYSQL\_ROOT\_PASSWORD=12345 img-db'

            }

        }

    }

}

Click Save to store the configuration

Click Build Now to run the job

After a while, the build will end with success

Open a browser tab and navigate to <http://192.168.99.100:9090>

Our application should be there and working