M5: Jenkins Advanced

# Assignment

You are expected to create the following

* A setup with two virtual machines – one with **Jenkins** and another with **Docker** installed just like on the practice

*\* You can adjust their parameters in order to fit within your available resources*

* On the **Docker** machine you must deploy **Gitea** (as we did during the practice)
  + Create a repository to host your version of the **BGApp** application
  + Enable **Webhooks**
* On the **Jenkins** machine create a pipeline to build the **BGApp** application. There should be steps for:
  + Downloading the project from your **Gitea** repository
  + Using one **Docker Compose** file to
    - Build the images
    - Create a common network
    - Run the containers (the web container to publish port on **8080**)
  + Testing the application for reachability and that (after a short wait) one of cities (for example Sofia) is displayed
  + Stopping the application and removing the containers
  + Publishing the images to **Docker Hub**
  + Using another **Docker Compose** file to
    - Create a common network
    - Run the containers (the web container to publish port on **80**)

As usual, try to do the infrastructure part as automated as possible. Of course, using **Vagrant**

For the **Jenkins** part, try to automate it as much as possible. Ideally, there should be a **Jenkinsfile** hosted in the repository from which you create the pipeline (try to use the **CLI** to automate this as well)

# Solutions

## Task 1

* A setup with two virtual machines – one with **Jenkins** and another with **Docker** installed just like on the practice

*\* You can adjust their parameters in order to fit within your available resources*

* Single virtual machine with Jenkins and Docker installed

### Vagrantfile

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

# vi: set ft=ruby :

Vagrant.configure(2) do |config|

  config.ssh.insert\_key = false

  config.vm.define "jenkins" do |jenkins|

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

    jenkins.vm.hostname = "jenkins.do1.lab"

    jenkins.vm.provider :virtualbox do |vb|

      vb.customize ["modifyvm", :id, "--memory", "3072"]

    end

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

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

    jenkins.vm.provision "shell", path: "add\_hosts.sh"

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

  end

  config.vm.define "docker" do |docker|

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

    docker.vm.hostname = "docker.do1.lab"

    docker.vm.provider :virtualbox do |vb|

      vb.customize ["modifyvm", :id, "--memory", "3072"]

    end

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

    docker.vm.provision "shell", path: "add\_hosts.sh"

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

  end

end

### add\_hosts.sh

#!/bin/bash

echo "192.168.99.101 jenkins.do1.lab jenkins" >> /etc/hosts

echo "192.168.99.102 docker.do1.lab docker" >> /etc/hosts

### Install-jenkins.sh

#!/bin/bash

echo "\* Update repositories and install Java"

apt-get update

apt-get install -y fontconfig openjdk-11-jre

echo "\* Add and import the repository key"

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

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 "\* Update repositories and install Jenkins"

apt-get update

apt-get install -y jenkins

echo "\* admin password is:"

cat /var/lib/jenkins/secrets/initialAdminPassword

### install-docker.sh

#!/bin/bash

echo "\* Add required packages"

apt-get update

apt-get install -y ca-certificates curl gnupg lsb-release git

echo "\* Add repository key"

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

echo "\* Add the 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 the packages (Java, git, Docker)"

apt-get update

apt-get install -y fontconfig openjdk-11-jre git docker-ce docker-ce-cli containerd.io

echo "\* Adjust the group membership"

usermod -aG docker vagrant

# echo "\* Install Docker Compose"

# mkdir -p /home/vagrant/.docker/cli-plugins/

# curl -SL https://github.com/docker/compose/releases/download/v2.2.3/docker-compose-linux-x86\_64 -o /home/vagrant/.docker/cli-plugins/docker-compose

# chmod +x /home/vagrant/.docker/cli-plugins/docker-compose

echo "\* Install Docker Compose V2 for all users"

mkdir -p /usr/local/lib/docker/cli-plugins

curl -SL https://github.com/docker/compose/releases/download/v2.2.3/docker-compose-linux-x86\_64 -o /usr/local/lib/docker/cli-plugins/docker-compose

chmod +x /usr/local/lib/docker/cli-plugins/docker-compose

echo "\* Get docker-compose version"

docker compose version

### Post-installation

Then open [**http://192.168.99.101:8080**](http://192.168.99.101:8080)in a browser

* Get default jenkins admin password

**sudo cat /var/lib/jenkins/secrets/initialAdminPassword**

* On the next screen, click on the **Install suggested plugins** option
* Create the new administrator user

- Username: **admin**

- Password: **Password1**

- Full name: DevOps Administrator

- E-mail address: [doadmin@do1.lab](mailto:doadmin@do1.lab)

* Install the default set of plugins

### Preparation (on the CLI)

#### Docker machine

Open a new session

**vagrant ssh docker**

Now, we must create a new **jenkins** user with **Password1** as its password

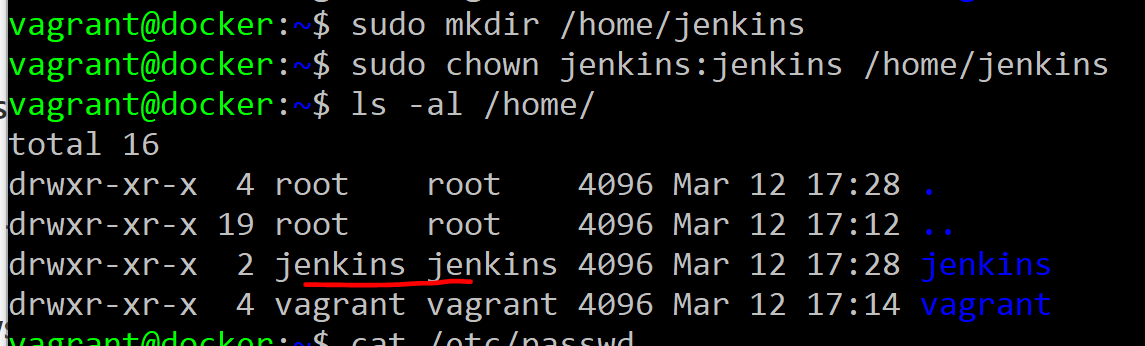
**sudo useradd jenkins**

**sudo passwd jenkins**

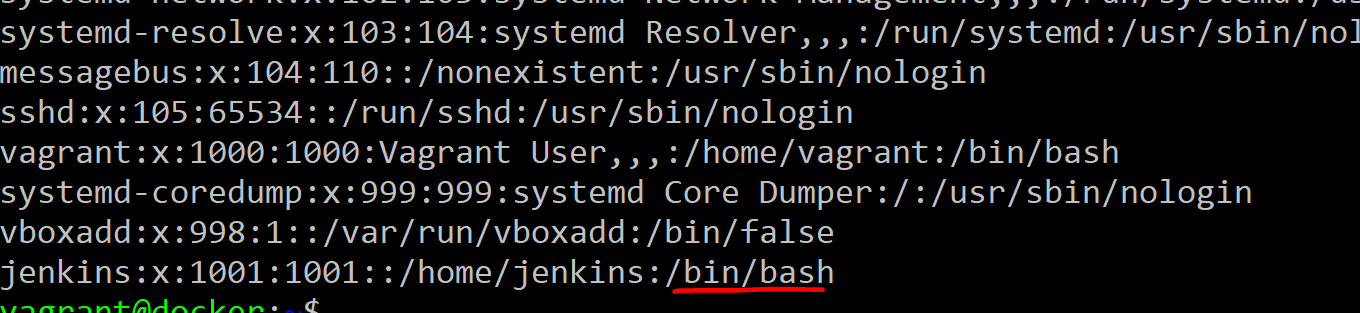
Create home folder for jenkins user

**sudo mkdir /home/jenkins**

**sudo chown jenkins:jenkins /home/jenkins**



Fix **/etc/passwd**



Grant the new user **sudo** permissions

**sudo visudo**

Enter the following

**jenkins ALL=(ALL) NOPASSWD: ALL**

Save and close the file

Add the **jenkins** user to the **docker** group

**sudo usermod -aG docker jenkins**

Switch to the **jenkins** user to test

**su - jenkins**

Exit the **jenkins** user session

**exit**

Now, we can close the session to the **docker** machine

**exit**

#### Jenkins machine

And open a new session to the **jenkins** machine

**vagrant ssh jenkins**

Make sure that the user has password (for example, **Password1**)

**sudo passwd jenkins**

And that the shell is set to **/bin/bash**

Switch to the **jenkins** user with

**su - jenkins**

Generate a public/private key pair

**ssh-keygen -t rsa -m PEM**

Copy the **SSH** key to the **jenkins** machine

**ssh-copy-id jenkins@jenkins.do1.lab**

Test the connectivity

**ssh jenkins@jenkins.do1.lab**

Close the test connection

**exit**

Copy the **SSH** key to the **docker** machine

**ssh-copy-id jenkins@docker.do1.lab**

Test the connectivity

**ssh jenkins@docker.do1.lab**

Close the test connection

**exit**

Exit the **jenkins** session

**exit**

Finally, we can close the session to the **jenkins** machine

**Exit**

### Preparation (in Jenkins)

#### Credentials

Navigate to **Manage Jenkins** and then to **Manage Credentials**

Then click **Jenkins** and then finally **Global**

Let’s add the first set

Click the **Add Credentials** button

Select **Username with password**

Enter **jenkins** for **Username**

Enter <jenkins\_password>(**Password1)** for **Password**

Set the **Description** to **Local user with password**

Click **OK**

Now, add the second set

Click the **Add Credentials** button

Change the **Kind** to **SSH Username with private key**

In the **Description** box enter **Credentials from file**

Enter **jenkins** for **Username**

Select **Enter directly**

Then click the **Add** button

And paste the contents of the private key file which can be extracted (on the **jenkins** machine) with

**sudo cat /var/lib/jenkins/.ssh/id\_rsa**

Finally, click the **OK**

#### Plugins

Return to the initial page of the **Jenkins** portal

Navigate to **Manage Jenkins** and then to **Manage Plugins**

Switch to **Available** and search for **SSH**

Select it and click **Download now and install after restart**

Then select **Restart Jenkins when installation is complete and no jobs are running**

#### Hosts

Once back, navigate to **Manage Jenkins** and then to **Configure System**

Scroll down to the **SSH remote hosts /** **SSH sites** section

Click the **Add** button once to create a record

- hostname: **jenkins.do1.lab**

- port: **22**

- credentials: **jenkins (Credentials from file)** or **jenkins (Local user with password)**

And then once more for an additional record

- hostname: **docker.do1.lab**

- port: **22**

- credentials: **jenkins (Credentials from file)** or **jenkins (Local user with password)**

Check the connectivity for each one of the hosts with the **Check connection** button

Confirm with the **Save** button

#### Slave Host

Now, we can register (or add) the **docker** machine in **Jenkins**

Open the **Jenkins** portal or return to the home page

Navigate to **Manage Jenkins** and then to **Manage Nodes and Clouds**

Click on the **New Node** command

Enter **docker-node** for **Node** name

And select the **Permanent Agent** option

Confirm with **Create**

Next, set the description to **Docker slave node**

And enter the following values in the appropriate places

Set the **# of executors** to **4**

Use **/home/jenkins** for **Remote root directory**

Enter **docker-node** in **Labels**

For **Usage** set **Only build jobs with label expression matching this node**

Set **Launch method** to **Launch slave agents via SSH**

For **Host** enter **docker.do1.lab**

Now, for Credentials select **jenkins (Credentials from file)**

Leave the **Host Key Verification Strategy** to **Known hosts file**

Confirm with the **Save** button

Next, click on the newly added host and then on the logs for the host.

Monitor the process of agent installation. Once done, continue with the next steps.

## Task 2

* On the **Docker** machine you must deploy **Gitea** (as we did during the practice)
  + Create a repository to host your version of the **BGApp** application
  + Enable **Webhooks**

### Install Gitea

Open a terminal session to the **docker** machine

**vagrant ssh docker**

Execute the following to copy locally the **docker-compose.yml** file

**cp /vagrant/docker-compose.yml .**

Explore it and then start it

**docker compose up -d**

Now, return on the host and open a browser tab and navigate to [**http://192.168.99.102:3000**](http://192.168.99.102:3000)

Change at least the following two options:

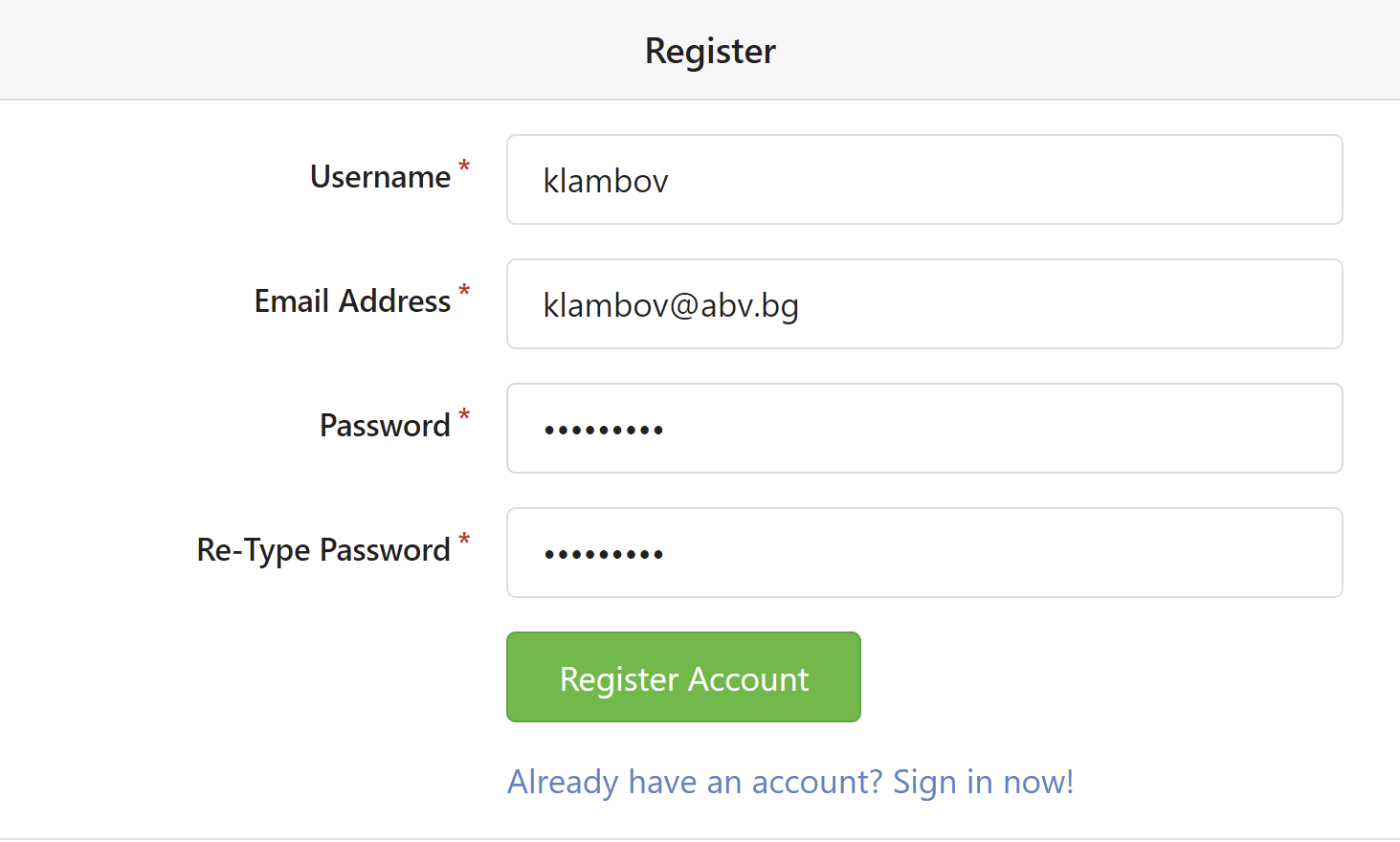
* Set **Server Domain** to **192.168.99.102**
* Set **Gitea Base URL** to **http://192.168.99.102:3000/**

Accept the rest of the proposed values and click the **Install Gitea** button

Once, the page is refreshed note that initially it may show an error. Refresh it a few times

Click the **Need an account? Register now.** link to create an account

Enter the required data and click the **Register Account** button



### Create a repository to host your version of the BGApp

Let’s do something more complex and complete

Navigate to **Gitea**

Click on the **+** sign next to the account menu (top right)

Select **New Repository**

Click **Migrate repository**

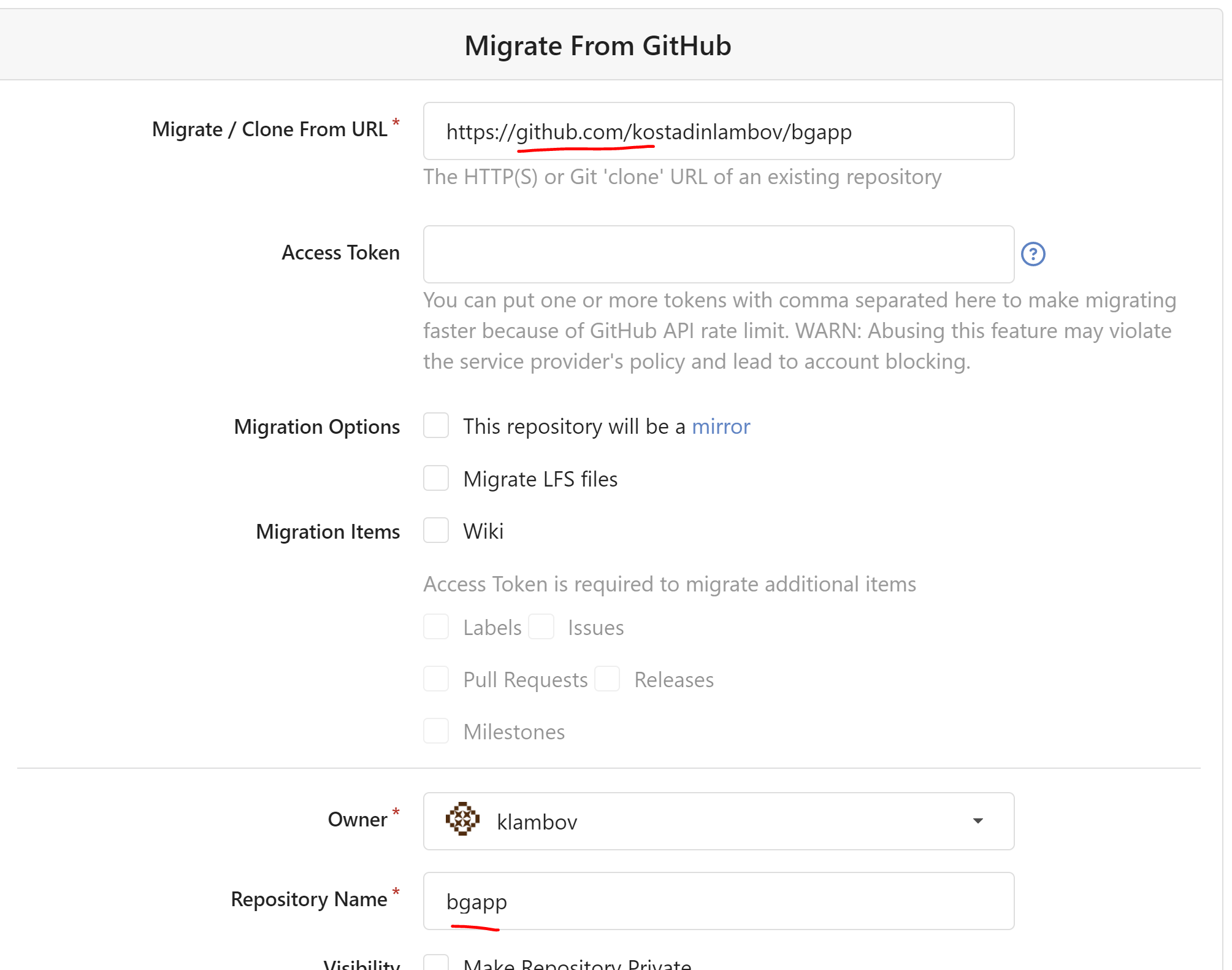
Click on **GitHub**

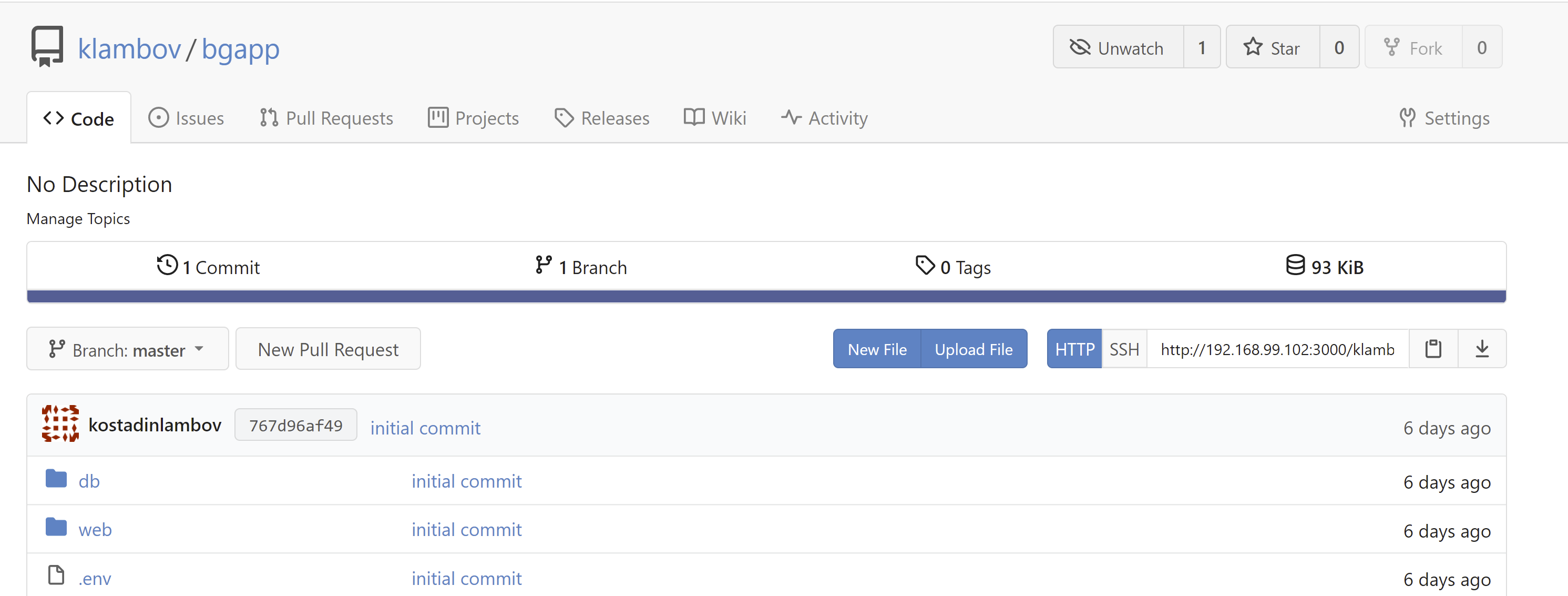
Enter [**https://github.com/kostadinlambov/bgapp**](https://github.com/kostadinlambov/bgapp)in the **Migrate / Clone From URL** field

Fill **bgapp** in the **Repository Name** field

Click **Migrate Repository**

Once done, explore the structure and the code





Return to the **Dashboard** in **Jenkins**

Click **New Item**

Then select **Pipeline** and enter **BgApp-Pipeline** for name and click **OK**

Select **GitHub project** and paste the repository URL

For example, **http://192.168.99.102:3000/klambov/bgapp**

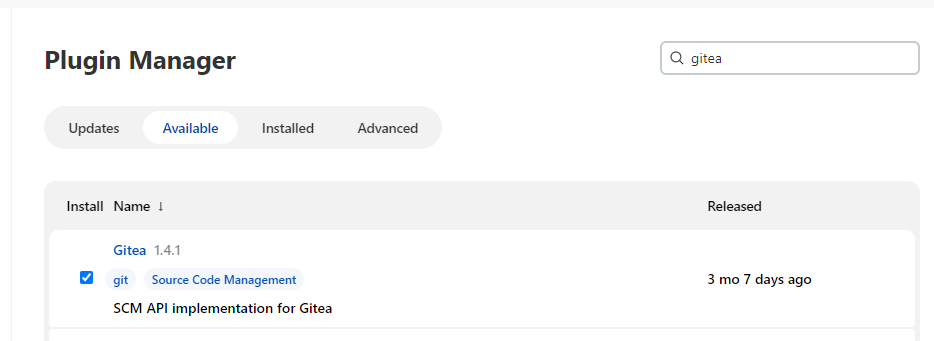
Scroll down to the **Pipeline** section and paste the your Pipeline.

### Enable Webhooks

Return to the **Dashboard** of **Jenkins**

Navigate to **Manage Jenkins** > **Manage Plugins**

Switch to **Available** and search for **gitea**



Select it and click **Download now and install after restart**

Then you may select the **Restart Jenkins when installation is complete and no jobs are running** option

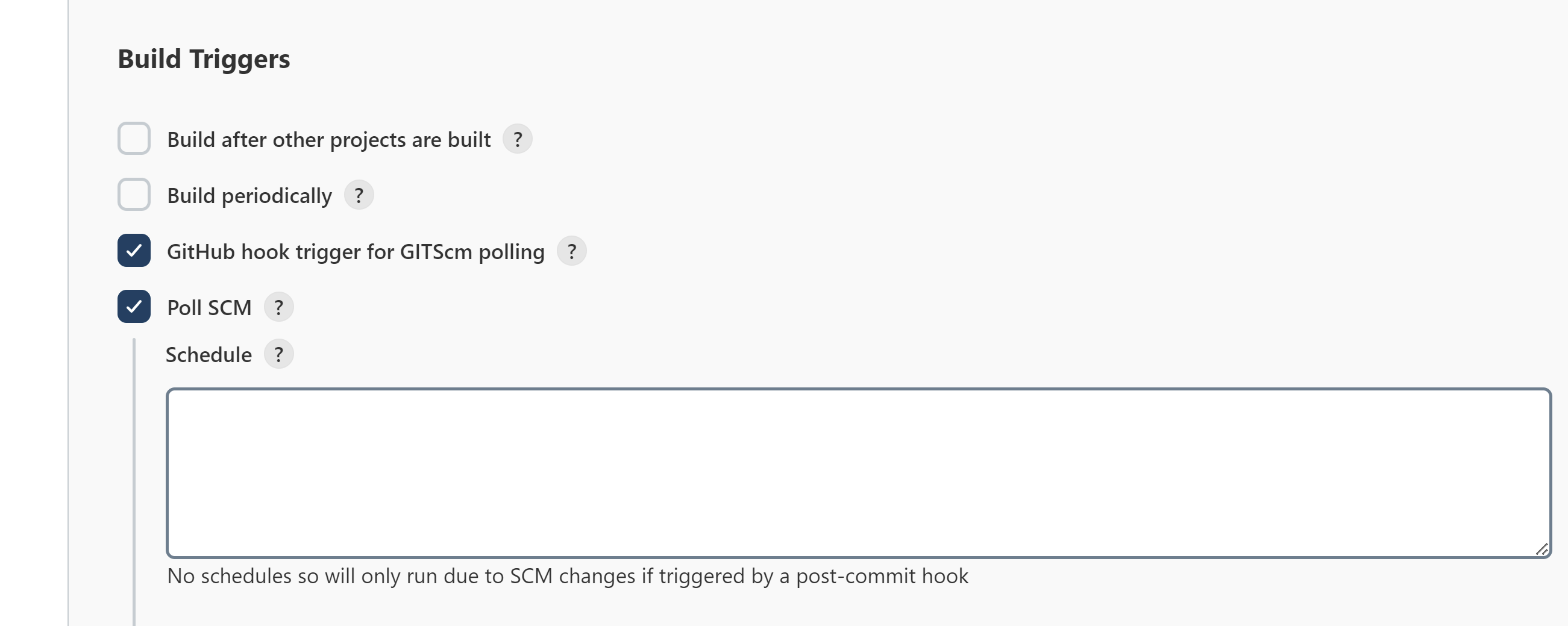
Once the system is ready, log in back again

Return to the **Dashboard** and click on the **BgApp** job

Then click **Configure**

Scroll down to **Build Triggers** section

Select both **GitHub hook trigger for GitScm polling** and **Poll SCM**

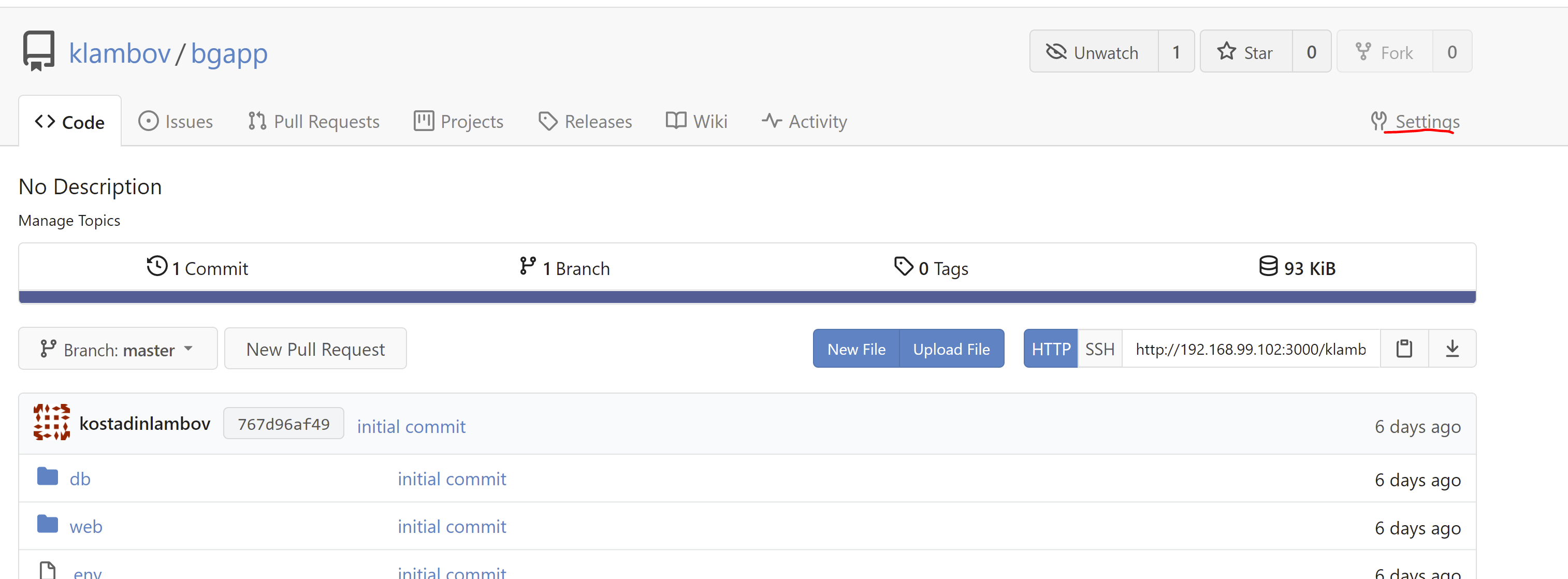


Click **Save** to confirm the changes

Now, return to **Gitea**

Go to the corresponding repository

Click **Settings** (far right corner)

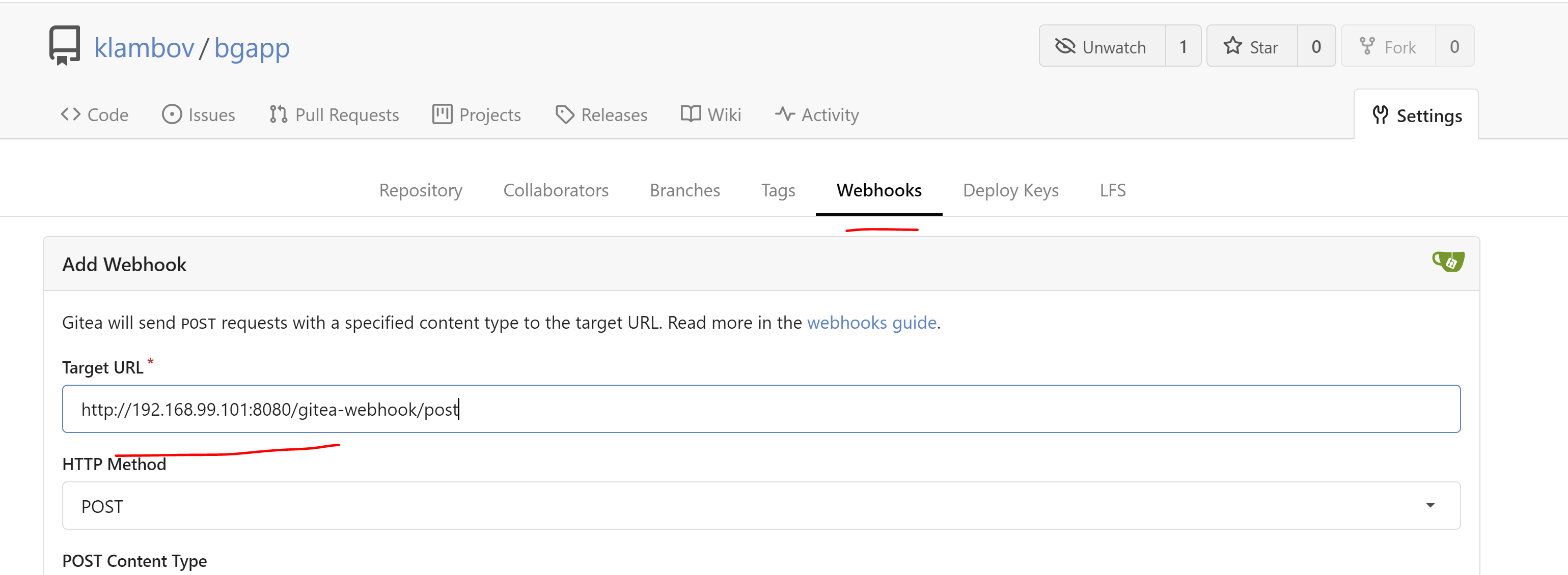


Switch to **Webhooks**

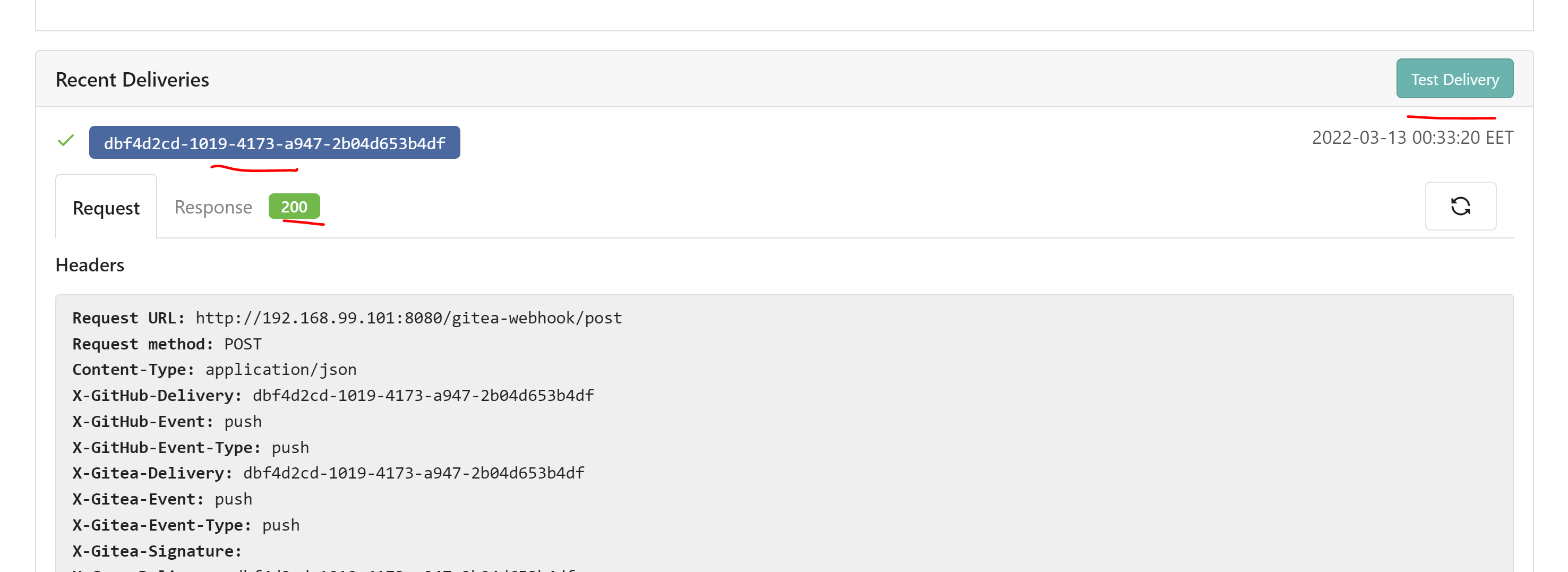
Click **Add Webhook** and select **Gitea**

Enter <http://192.168.99.101:8080/gitea-webhook/post> in the **Target URL** field

Then click the **Add Webhook** button



Click back on the hook to enter in edit mode

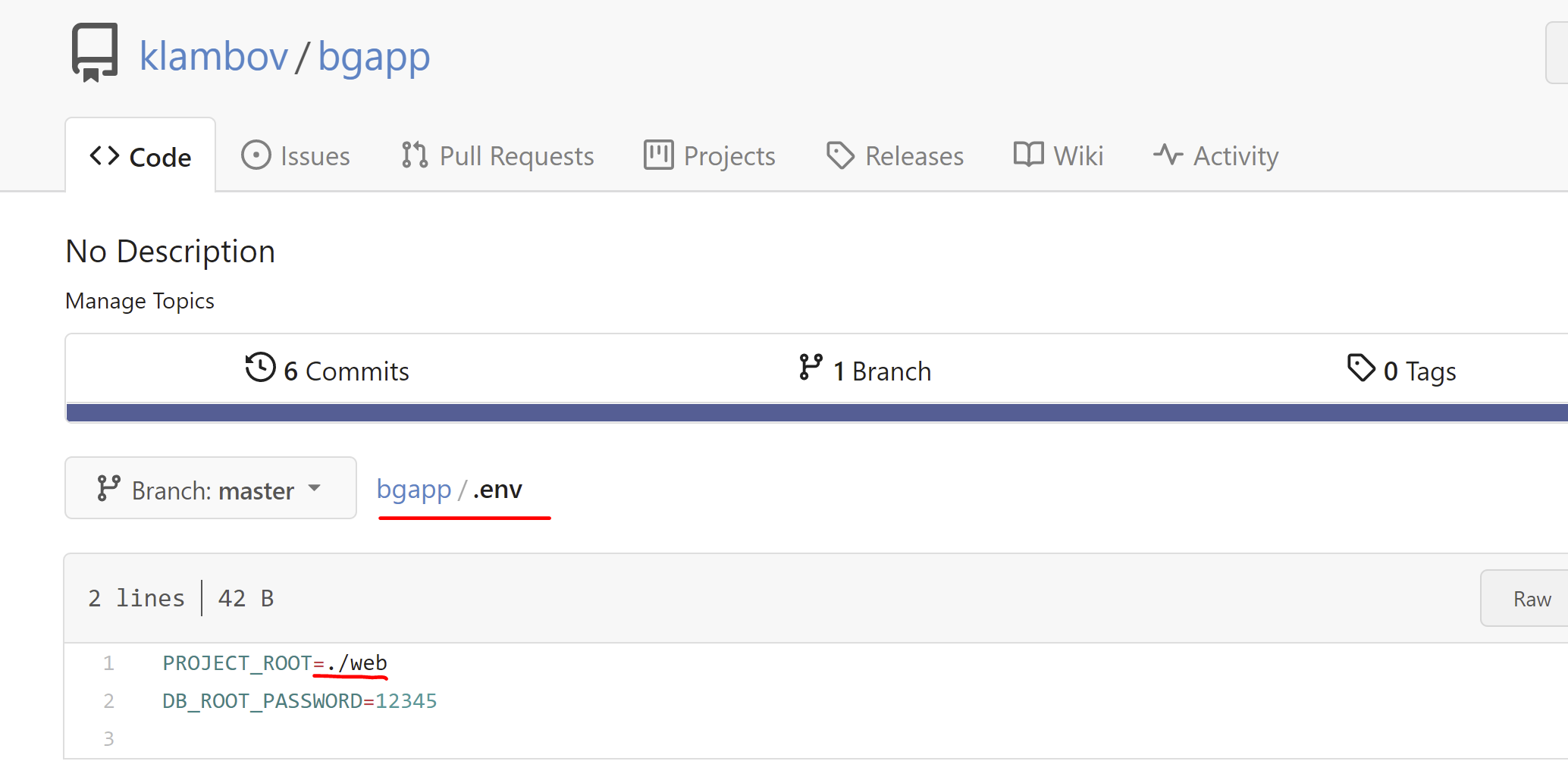


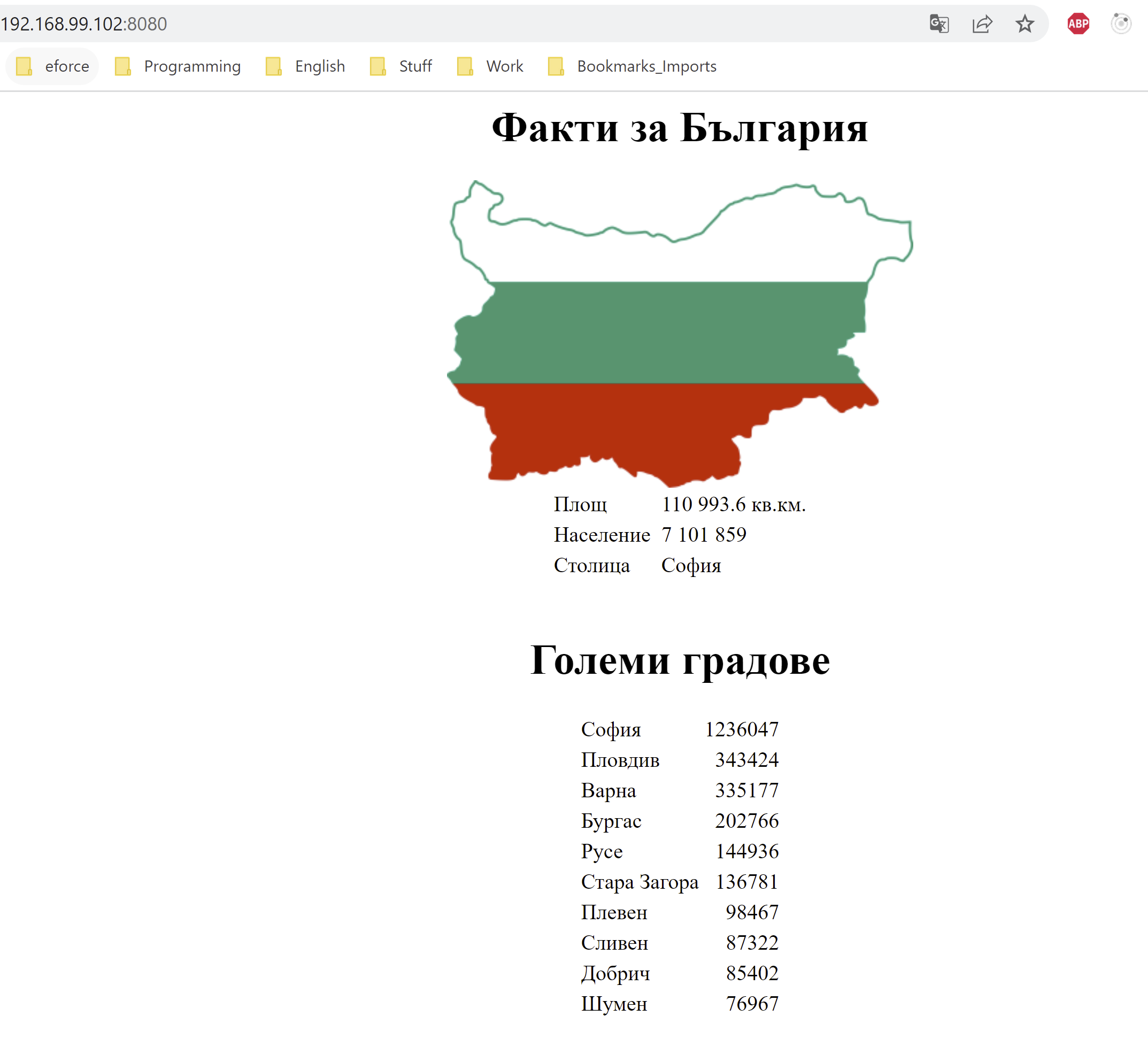
## Task 3

* On the **Jenkins** machine create a pipeline to build the **BGApp** application. There should be steps for:
  + Downloading the project from your **Gitea** repository
  + Using one **Docker Compose** file to
    - Build the images
    - Create a common network
    - Run the containers (the web container to publish port on **8080**)
  + Testing the application for reachability and that (after a short wait) one of cities (for example Sofia) is displayed
  + Stopping the application and removing the containers
  + Publishing the images to **Docker Hub**
  + Using another **Docker Compose** file to
    - Create a common network
    - Run the containers (the web container to publish port on **80**)
* Jenkins pipeline to build the BGApp application used in M3 and M4. There should be steps for:
  + Downloading the project from your remote repository
  + Building the images
  + Creating a common network
  + Running the containers

### Fix project root folder

We have to fix the PROJECT\_ROOT env variable in the repository to:  
**PROJECT\_ROOT=./web**





### docker-compose.yaml

version: "3.8"

services:

    web:

        build:

            context: .

            dockerfile: Dockerfile.web

        ports:

            - 8080:80

        volumes:

            - "${PROJECT\_ROOT}:/var/www/html:ro"

        networks:

            - app-network

        depends\_on:

            - db

    db:

        build:

            context: .

            dockerfile: Dockerfile.db

        networks:

            - app-network

        environment:

            MYSQL\_ROOT\_PASSWORD: "${DB\_ROOT\_PASSWORD}"

networks:

    app-network:

### Add Docker Hub token

Now, let’s publish the image to **Docker Hub** if the test part succeeds

Log in to **Docker Hub**

Then go to **Account Settings** and then **Security**

Click the **New Access Token** button

Enter **Jenkins** in the **Access Token Description**

And click **Generate**

Click the **Copy and Close** button and store the token somewhere safe

Return to **Jenkins**

Navigate to **Manage Jenkins** > **Manage Credentials**

Then click on **Jenkins** > **Global credentials**

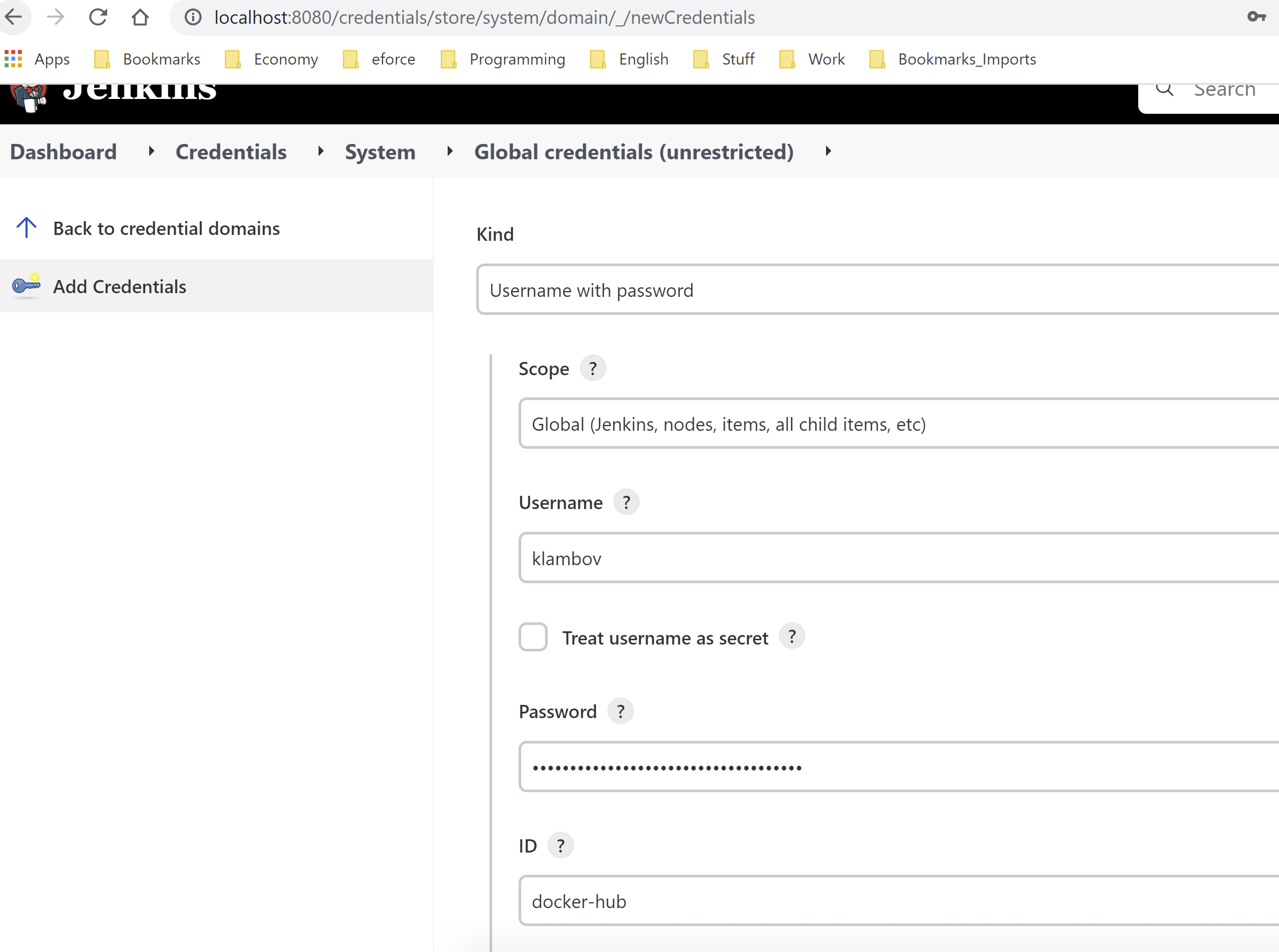
Finally, click the **Add Credentials** link

Enter your **Docker Hub** username in the **Username** field

Then paste the token in the **Password** field

Enter **docker-hub** in the **ID** and **Description** field

Finally, click **OK**



Add the following just after the **agent** section

    environment

    {

        DOCKERHUB\_CREDENTIALS=credentials('docker-hub')

    }

And then, the following two stages after the **Test** stage

        stage('Login')

        {

            steps

            {

                sh 'echo $DOCKERHUB\_CREDENTIALS\_PSW | docker login -u $DOCKERHUB\_CREDENTIALS\_USR --password-stdin'

            }

        }

        stage('Push')

        {

            steps

            {

                sh 'docker image tag img-calc <username>/supercalc'

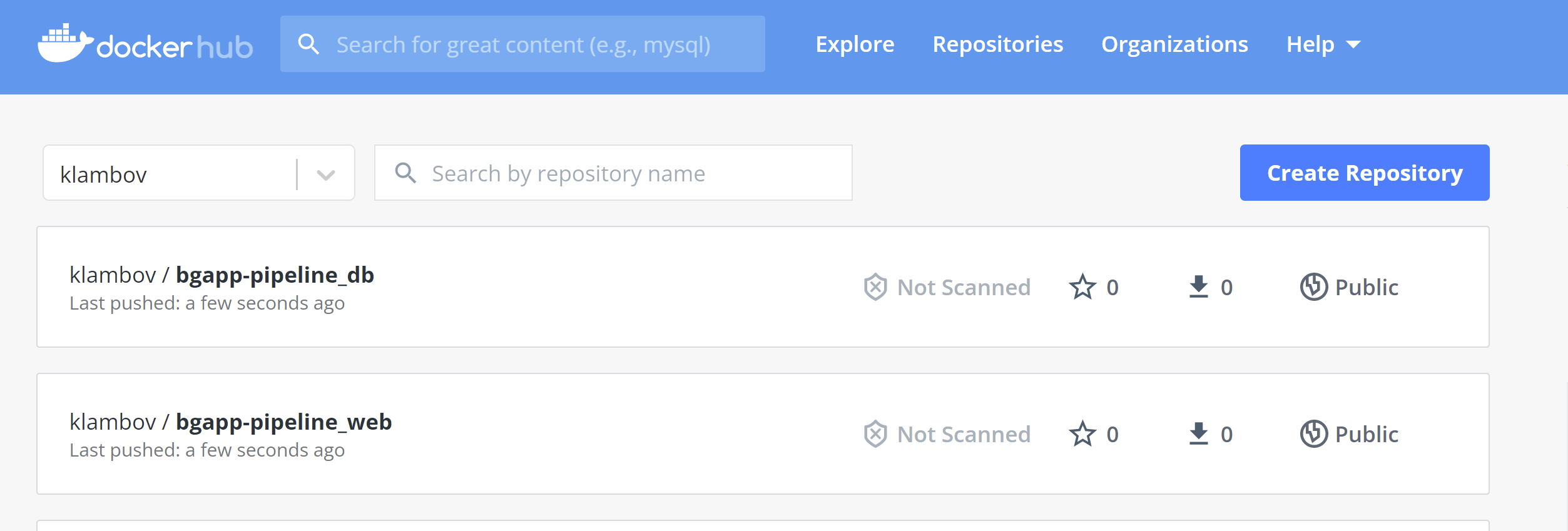
                sh 'docker push <username>/supercalc'

            }

        }

Click the **Save** button

### Push the images to Docker Hub



### docker-compose.images.yaml

version: "3.7"

networks:

  bgapp-net:

services:

  web:

    image: klambov/bgapp-pipeline\_web

    container\_name: web

    restart: always

    networks:

      - bgapp-net

    volumes:

      - "${PROJECT\_ROOT}:/var/www/html:ro"

    ports:

      - "80:80"

    depends\_on:

      - db

  db:

    image: klambov/bgapp-pipeline\_web

    restart: always

    environment:

      - MYSQL\_ROOT\_PASSWORD=12345

    networks:

      - bgapp-net

### Jenkinsfile

pipeline

{

    agent

    {

        label 'docker-node'

    }

    environment

    {

        DOCKERHUB\_CREDENTIALS=credentials('docker-hub')

    }

    stages

    {

        stage('Clone the project')

        {

             steps

            {

                 git branch: 'master', url: 'http://192.168.99.102:3000/klambov/bgapp'

            }

        }

        stage('Start the BgApp')

        {

            steps

            {

                sh 'docker compose up -d'

            }

        }

        stage('Test the application')

        {

            steps

            {

                script

                {

                    echo 'Test #1 - reachability'

                    sh 'echo $(curl --write-out "%{http\_code}" --silent --output /dev/null http://localhost:8080) | grep 200'

                    echo 'Test #2 - test if София is displayed'

                    sh "curl  http://localhost:8080 | grep София"

                }

            }

        }

        stage('Stop the BgApp and remove the containers')

        {

            steps

            {

                sh 'docker compose down -v'

            }

        }

         stage('Login')

        {

            steps

            {

                sh 'echo $DOCKERHUB\_CREDENTIALS\_PSW | docker login -u $DOCKERHUB\_CREDENTIALS\_USR --password-stdin'

            }

        }

        stage('Push images')

        {

            steps

            {

                sh 'echo "Push web image"'

                sh 'docker image tag bgapp-pipeline\_web klambov/bgapp-pipeline\_web'

                sh 'docker push klambov/bgapp-pipeline\_web'

                sh 'echo "Push db image"'

                sh 'docker image tag bgapp-pipeline\_db klambov/bgapp-pipeline\_db'

                sh 'docker push klambov/bgapp-pipeline\_db'

            }

        }

        stage('Start the BgApp with the images from Docker Hub')

        {

            steps

            {

                sh 'docker compose -f docker-compose.images.yaml up -d'

            }

        }

    }

}