**Домашна Работа бр. 2   
Анета Станковска  
Индекс: 249012**

**Опис на CI pipeline и контејнеризација на:**

**1. git engine (gitea, gitlab...)**

**2. jenkins**

**3. nexus repository**

**Поврзување на сервисите со цел имплементација на CI за одредена апликација во рамки на домашна 3.**

**\* Втората домашна се надоврзува на првата, односно се користи истата основа (истиот бекенд сервис и истиот фронтенд сервис) со тоа што сега и во фронтенд сервисот додадов Dockerfile со цел да можам да го исконфигурирам и овој сервис.**

**Инсталација на Jenkins**

* За да може да користам Jenkins на мојата локална машина најпрво треба да го инсталирам. Со оглед на тоа што веќе имам инсталирано Docker на машината, ја стартувам Docker Desktop апликацијата и потоа во терминал ја извршувам следната команда: **docker pull jenkins/jenkins:lts** за да download-ирам Jenkins.**A computer screen shot of a black screen

  Description automatically generated**
* За да верифицирам дека се е инсталирано, само ја повикувам командата docker images, по што го добивам следното:  
  **A black background with white text

  Description automatically generated**
* Следно креирам admin user и password за Jenkins со помош на командата: **docker run -p 8080:8080 -v jenkins\_home:/var/jenkins\_home --name jenkins jenkins/jenkins:lts** . Jenkins ни дозволува и да го смениме paddword-от долколку сакаме за време на setup процесот.
* Во browser отварам <http://localhost:8080/> за да се логирам на Jenkins.
* **A screenshot of a computer

  Description automatically generated**
* Инсталирам default plugins од кои во моментот ми треба Pipeline plugin-от, потоа Jenkins ме прашува дали сакам да креирам user или да продолжам како admin. Одбирам да продолжам како admin. Откако ќе преминам на следниот чекор, конфигурирам инстанца.
* A screenshot of a computer

  Description automatically generated
* Откако ќе ја зачувам конфигурацијата, го добивам почетниот екран.
* Додавам system credentials што ќе ги користам во мојот Jenkinsfile
* A screenshot of a computer

  Description automatically generated

**Креирање на Jenkinsfile**

* Најпрво започнав со креирање на **Jenkinsfile** во root фолдерот, а тоа е Homework\_2 фолдерот.
* Jenkinsfile фајлот го сместив таму со цел да може да пристапи и до двата сервиси (server и client апликациите).
* Со оглед на тоа што во претходните чекори веќе го инсталирав и исконфигурирав Jenkins, ми останува само уште да го исконфигурирам самиот Jenkinsfile.
* Во самиот Jenkinsfile ги поставив сите потребни конфигурации за самиот pipeline како и за Docker.

**Креирање на docker-compose.yml фајл**

* Во root фолдерот креирав **docker-compose.yml** фајл.
* Целта на docker-compose.yml е да обезбеди декларативен начин за дефинирање и управување со повеќе Docker контејнери кои треба да работат заедно како дел од една апликација или проект. Овој фаjл ја опишува инфраструктурата на апликацијата, вклучувајќи ги контејнерите, нивните мрежи, волумените и другите конфигурации потребни за да функционираат правилно.
* Со една команда: **docker-compose up**, се стартуваат сите контејнери дефинирани во docker-compose.yml и се поврзуваат меѓусебно според дефинираните правила.
* Ја извршувам командата **docker-compose up** за да ги стартнам сите контејнери.

**Setup на Jenkins за GitHub repository**

* Во root фолдерот креирав **docker-compose.yml** фајл.

A screenshot of a computer

Description automatically generated

* Во оваа фаза нема да сетурам автоматски build, но доколку сакам, мопжам да сетирам и автоматски build, или тригерирање на build на одредено време.
* Run на setup за да може Jenkins container-от да го користи Docker daemon од host машината.
* A screen shot of a computer

  Description automatically generated
* Следно **docker exec -it jenkins bash**

**Steps to Configure the Pipeline**

1. **Set SCM to Git**:
   * In the dropdown next to SCM, select **Git**.
2. **Enter Your Repository URL**:
   * Add the HTTPS URL of your GitHub repository:

https://github.com/anetastankovska/FINKI-DevOps.git

1. **Set Credentials**:
   * If your repository is private, click **Add** next to the Credentials dropdown and add your GitHub credentials.
   * Select the credentials from the dropdown after adding them.
2. **Specify the Branch**:
   * Use the branch name you are working on (e.g., main or master).
3. **Set the Script Path**:
   * Enter the path to the Jenkinsfile inside your repository:

Copy code

Homework\_2/Jenkinsfile

1. **Enable Lightweight Checkout**:
   * Leave the **Lightweight checkout** option enabled. This ensures that only the Jenkinsfile and necessary files are fetched instead of the entire repository.

**Final Configuration**

After configuring, it should look something like this:

* **SCM**: Git
* **Repository URL**: https://github.com/anetastankovska/FINKI-DevOps.git
* **Branch**: main
* **Script Path**: Homework\_2/Jenkinsfile

A screenshot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

Screenshots од дел од командите се ставени во images фолдерот.

Source кодот е поставен на моето GitHub репо, во рамки на FINKI-DevOps репозиториум.

[ЛИНК до репо](https://github.com/anetastankovska/FINKI-DevOps/tree/main/Homework_1)

**Преглед на најважните команди**

* Build an image: docker build -t <image\_name>:<tag> .
* List images: docker images
* Run a container: docker run -d -p <host\_port>:<container\_port> <image\_name>:<tag>
* Run a container interactively: docker run -it <image\_name>:<tag> /bin/bash
* List running containers: docker ps
* List all containers (including stopped): docker ps -a
* Stop a container: docker stop <container\_id>
* Remove a container: docker rm <container\_id>
* Remove all stopped containers: docker container prune
* Remove an image: docker rmi <image\_name>:<tag>
* Login to Docker Hub: docker login
* Add tag to image: docker tag <existing\_image\_name>:<tag> <new\_image\_name>:<new\_tag>
* Commit changes in a container to a new image: docker commit <container\_id> <new\_image\_name>:<tag>
* Push an image to Docker Hub: docker push <image\_name>:<tag>
* Save an image as a tar file: docker save -o <file\_name>.tar <image\_name>:<tag>
* Load an image from a tar file: docker load -i <file\_name>.tar
* Remove unused images, containers, and networks: docker system prune
* Check container logs: docker logs <container\_id>
* Execute a command inside a running container: docker exec -it <container\_id> <command>