Lab 7 – Jenkins (DevOps, CI/CD tool)

Introduction:

Welcome to Lab 7! This simple exercise is designed to introduce you to Jenkins and continuous integration.

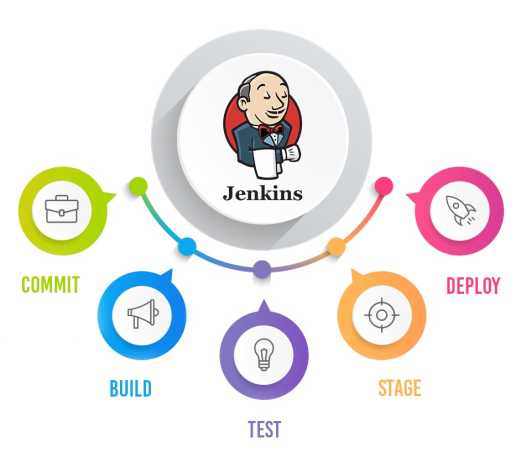
**General Instructions:**

1. All the screenshots must be uploaded to Edmodo before the stipulated deadline.
2. For each lab, screenshots need to be uploaded as mentioned under the “deliverables” section for each task **exactly** as mentioned.
3. If a task has some questions, these will not be evaluated but are extremely important in understanding the concepts.
4. Try to solve the tasks by yourselves, all relevant information to complete the tasks has been provided. In case you are stuck and not able to solve the issue, feel free to send email to pesu\_cc\_lab\_support@googlegroups.com

**What is Jenkins?**

Jenkins is an extensible, open source continuous integration server. It builds and tests your software continuously and monitors the execution and status of remote jobs, making it easier for team members and users to regularly obtain the latest stable code.

**What is Jenkins Pipeline?**



In simple words, Jenkins Pipeline is a combination of plugins that support the integration and implementation of continuous delivery pipelines using Jenkins.The pipeline as Code describes a set of features that allow Jenkins users to define pipelined job processes with code, stored and versioned in a source repository.

**Overview of the Experiment**

* Setup jenkins Using Docker.
* Set up a job in Jenkins to connect to your repository and build C++ hello.cpp.
* Set up second job to run the program after the build completes.

Prerequisite:

* Docker Installed on your system. (Refer to Task1 steps in the Lab5 manual)

**Task-1**

Aim: Set up Jenkins using Docker.

Deliverables:

1. 1A.png ( SS of the running Docker Container)

Steps:

* You will be given a Dockerfile.
* Put that Dockerfile in a folder and open a terminal in that folder.
* Build the dockerfile using this command: “sudo docker build -t jenkins:lab7 **.** “
* Run your container using this command “sudo docker run -p 8080:8080 -p 50000:50000 -it jenkins:lab7” (Note down the password shown on the terminal)
* Open URL: localhost:8080 on your browser.
* Enter the password shown on your terminal after running the container (You can set the password to ADMIN later).
  + In case you did not note down the password displayed on the terminal, you can find the password by connecting to the container (via “sudo docker exec –it <container\_id> /bin/bash”) and checking the file /var/Jenkins\_home **/secrets/initialAdminPassword inside the container**
* **Integrate GitHub to Jenkins**: When prompted for plugin installation, click on “Select Plugins to Install” and then search for Github and check the github option. (This step may take a few minutes to complete)
* Take the necessary SS.

**Task-2**

Aim: Set up a job in Jenkins to connect to your repository and build C++ hello.cpp.

Deliverables:

1. 2A.png ( picture showing the console output after the build is successful)
2. 2B.png (picture showing the Stable state of the task in Build History of Jenkins)

Steps:

* Navigate to Jenkins server. Click ***New Item***.
* Enter a name for your project, click *Freestyle Project*, then *OK*. *Note*: Please do not include a space.
* Name this something unique so there are no collisions.
* Select github project and Enter the repository URL. Use this repository link: https://github.com/sujeeth-cc/Lab7-Jenkins.git
* Set up *Source Code Management*, Select *git*. Enter the URL of git repository.
* Add “\*/main” in **Branches to build** (Do not delete the existing \*/master branch)
* Setting up *Build Triggers.* Select *Poll SCM*.
* Set up cron job by putting in “H/2 \* \* \* \*” in the Schedule box
* Set up *Build*. In **Add build** **step** pull-down menu, select *Execute Shell*.
* Enter “make -C original“(This will run the Makefile).
* Click *Save*.
* Click on build now.
* Take the required SS.

**Task-3**

Aim: The final step to this exercise is to set up a second job that automatically runs after the project builds. This is different from the other job because this will not have a git repository - it doesn't even build anything.

*Just a note: In a real-life scenario you wouldn't run a program through a build job just like this because I/O is not possible via this console. There are other tools people use at this step like SeleniumHQ, SonarQube, or a Deployment. The point of this is to show downstream/upstream job relationships.*

*Deliverables*

1. *3A.png (Console output of second job)*
2. *3B.png ( Status page of first job)*
3. *3C.png ( Build History of jenkins)*
4. *3D.png (Jenkins Dashboard)*

*Steps:*

* Create a new Job in Jenkins, Click *New Item* in the left panel
* Enter a name for your second job, click *Freestyle Project*, then *OK*.
* Go immediately to build step and select *Execute Shell*.
* Enter the following Command /var/jenkins\_home/workspace/<the name of your first project>/original/hello\_exec
* Save
* Set your first job to call the second.
* Go to your first job (i.e item) and open the *Configure* page in the pull down menu
* Scroll to bottom and add a Post-Build Action. Select ***Build other projects***.
* Enter the name of your second job.
* Save.
* Run your first job.
* Do this by clicking build now on the main page.
* After that successfully builds, go and check your second job. You should see it successfully run.
* Select a Build Job from History and go to the console log to see your program output. If you program has run there then you successfully set up a basic pipeline.
* Take the required SS.