Lab 6 - Continuous Integration with Jenkins

In this lab, you will explore how to perform Continuous Integration(CI) using Jenkins. You are highly recommended to use team virtual machine for this lab.

Jenkins has a wide range of functionalities that facilitate the software development process. In this lab, we will focus on the Continuous Integration aspect, which means updates to the code base are continuously tested to ensure the program's quality.

To receive credit for this lab, show your work to the TA during recitation.

Deliverables

- [ ] Show correct configuration of a Jenkins pipeline on **forked** **Lab5 Github Repo**. The build process must automatically fetch Jenkinsfile and run pipeline according to it.
- [ ] Complete the `jenkinsfile` to make the Jenkins pipeline test the repo during each build.
- [ ] Complete the `test_data_split` function in `test.py` to test data split step.

Java Installation

- Go to Jenkins Installation Page, select the Operating System you currently use. For your team server, please select linux.
- Under section Installation of Java, install java using command: `sudo apt install fontconfig openjdk-17-jre`.
- Show the success of installation by running `java --version`

Jenkins Installation

- In the same page for **Java Installation**, locate the installation command for Jenkins. For linux ubuntu system, it's:

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

- Use command `sudo systemctl start jenkins` to start Jenkins server.
- Show the success of Jenkins installation by running `sudo systemctl status jenkins`
- Enter `127.0.0.1:8080` in your browser and set up jenkins according to post installation wizard.
- Give Jenkins sudo permission by running `sudo visudo` and add line `jenkins ALL=(ALL) NOPASSWD: ALL` to this file.

Fork & Clone the Git Repository and Set Up Running Environment

- Please use a virtual environment management tool to for this project. In your team server, it is recommended to use miniconda. Create a python virtual environment and install `pytest`, `numpy`, `pandas`, and `sklearn` packages. If `git` is not installed in your team server, please run `sudo apt install git`. (You can use `venv` or `pipenv` instead of `conda` and adjust Jenkinsfile accordingly)
- Fork repository for lab 5 repository and clone it to your local machine.

Setting Up a Jenkins Pipeline

- Enter Jenkins dashboard: 127.0.0.1:8080 using your web browser.
- Clikc on `+ new item` button on the left.
- Use `mliplab5` as your project name and choose `pipeline` project.
- Under `General` section, click on `Github Project` and provide your **forked repository** http url.
- Under `pipeline`, click on `pipeline definition` and choose `Pipeline Script from SCM`. You need to create personal access token. Then choose git as your `SCM`. Add a `username password credential` with **any username** and your **personal access token as the password**.
- Change the `branch specifier` to `main`. Then, during each build, your Jenkins will pull code from Github and build upon it.

Complete the Jenkins File

- Read the Jenkinsfile carefully in the github repo. Then modifies **TODO** section to make Jenkins run pytest during each build.
- After the modification, **push** your changes into your **forked repository**.
- Show TA the modified Jenkins file to complete a deliverable.

Complete Test Case for Data Processing

- modified the **TODO** part in `test_utility.py` to test data split function. **Push the changes to Github**
- Build Jenkins project again and you should have a successful build.

Additional Resources

- Set up Jenkins Pipeline in SCM
- Use conda in Jenkins