LAB | MLOps Deployment from PROD to DEV

3.13.2

SDA

INTRODUCTION

The objective of this exercise was to simulate the process of deploying code from a **production environment (PROD)** to a **development environment (DEV)**. The exercise involved working in pairs, where one student acted as the **developer** pushing the code, and the other as the **gatekeeper**, ensuring the code runs smoothly and correctly in the new environment.

Steps Involved:

1. Developer Role:

- **Push Code to the Repository:**
  The first step was to either create a new project or use an existing one. I created the project, copied the code to a folder, and initialized a **git repository**.
  - I created a virtual environment using `venv`.
  - Installed all the necessary dependencies using `pip`.
  - Generated a **requirements.txt** file to list all the installed packages.
  - Initialized the Git

    ```
    git init
    git add .
    git commit -m "Initial commit"
    git remote add origin <remote-repo-url>
    git push -u origin main
    ```

  - Created a **Pull Request (PR)** in GitHub to make the code available for review.

2. Gatekeeper Role:

- **Review and Pull Code:**
  Once the developer pushed the code to the remote repository and created a PR, my role as a gatekeeper was to review the changes, check if everything is correct, and merge the pull request if no issues were found.
  - After merging the PR, I pulled the latest changes into my **development environment**:

```
git pull origin main
```

- Set up the environment by creating a new **virtual environment** and installing the dependencies listed in the **requirements.txt** file:

```
pip install –r requirements.txt
Ran the project to ensure everything was working correctly
and without errors.
```

3. Testing and Feedback:

After setting up the environment and running the project, I tested it to verify that the code worked as expected in the **development** environment. I provided feedback to the developer if any issues were encountered during testing, and these were addressed before the final deployment.

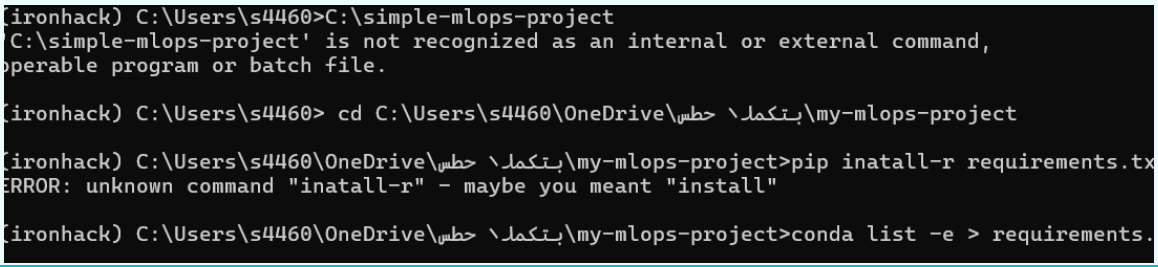
Challenges Faced:

- **Version Control Issues:**
  At some points, there were conflicts between local and remote repositories. Resolving these conflicts was an essential part of the process. Using proper Git commands like git pull, git push, and resolving merge conflicts helped ensure smooth synchronization between environments.
- **Dependency Issues:**
  Sometimes, there were dependencies listed in requirements.txt that weren't compatible with the version of Python in the development environment. It was important to manage versions carefully and update the dependencies where needed.

Outcome:

The project was successfully reviewed, merged, and tested in the development environment. The gatekeeper ensured that the code was fully functional and that the necessary dependencies were in place before deployment. By completing this exercise, I learned the importance of **version control** and **environment management** in the deployment process.

Workflow

1. 

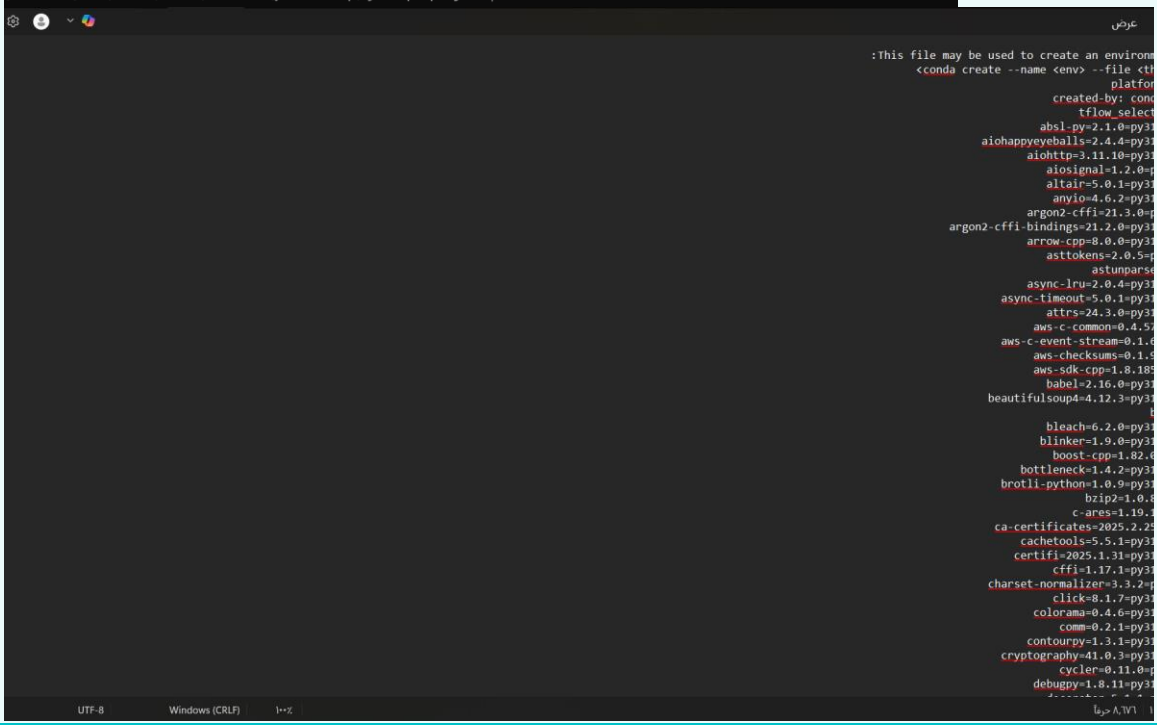
2.

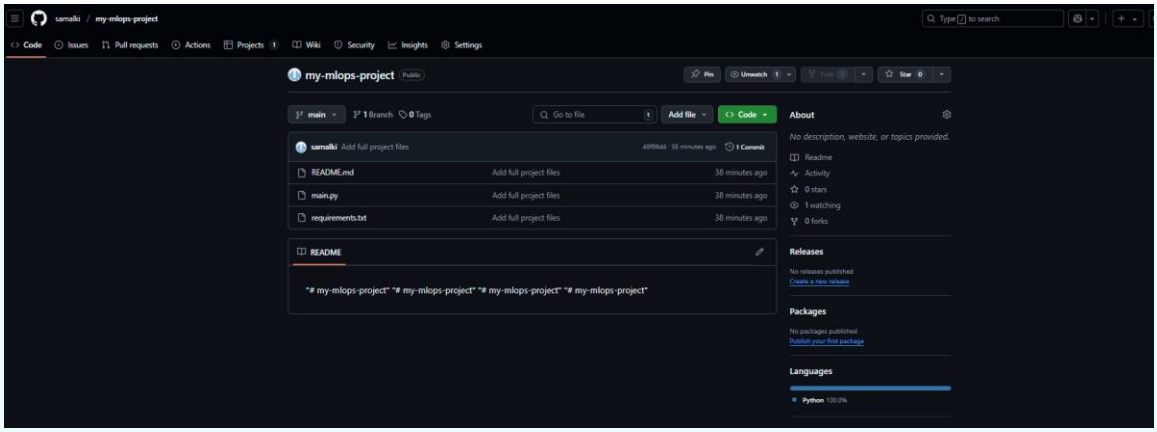
```
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   main.py
        new file:   requirements.txt
```

```
(ironhack) C:\Users\s4460\OneDrive\سطح المكتب\my-mlops-project>git commit -m "Add full project files"
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 's4460@Gam.(none)')

(ironhack) C:\Users\s4460\OneDrive\سطح المكتب\my-mlops-project>git config --global user.name "samalki"

(ironhack) C:\Users\s4460\OneDrive\سطح المكتب\my-mlops-project>git config --global user.email "sassmalki1@gmail.com"

(ironhack) C:\Users\s4460\OneDrive\سطح المكتب\my-mlops-project>git commit -m "Add full project files"
[main (root-commit) 48f86d4] Add full project files
 3 files changed, 293 insertions(+)
 create mode 100644 README.md
 create mode 100644 main.py
 create mode 100644 requirements.txt

(ironhack) C:\Users\s4460\OneDrive\سطح المكتب\my-mlops-project>git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 3.65 KiB | 1.82 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/samalki/my-mlops-project.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

(ironhack) C:\Users\s4460\OneDrive\سطح المكتب\my-mlops-project>
```

3.

4.



5. 

Conclusion:

This experience emphasized the significance of **effective collaboration** between the developer and gatekeeper roles, as well as the importance of keeping development and production environments synchronized. I also gained hands-on experience in using GitHub for version control and how to handle deployment in real-world MLOps workflows.