ASSIGNMENT

CodeKerdos

DEVOPS

Git and GitHub

ASSIGNMENT SUBMISSION GUIDLINES

1. You are required to submit answers to all the questions.
2. Create a folder named **DevOps-Assignment-Git** and for solution to each question create a sub-folder with name:
   Solution-*<QuestionNumber>* OR **Question-*<QuestionNumber>***
   and add the solution under these sub-folders.
   Now, your folder structure would look like this -

```
DevOps-Assignment-Git
   |
   |—— Solution-1
   |        |———index.html
   |
   |—— Solution-2
            |———index.html
```

3. To submit the solutions, **compress (zip)** the root/main folder and name the zip file with format:
   ***<StudentName>*_Batch*<YourBatchNumber>*_DevOps-Assignment-Git**
   Ex. - Ravi_Batch5_DevOps-Assignment-Git.
4. A **Google Form** will be shared with you to upload this ZIP file.
5. After evaluation **your scores and feedback** will be shared with you via email by your respective TAs.

DEADLINE

Please ensure that you provide your submissions by the mentioned deadline date.

Submissions received after this deadline will not be accepted, and failure to meet the deadline will result in not passing this module.

ASSIGNMENT QUESTIONS

Question 1

1. Based on what you have learnt in the class, do the following steps:
    a. Create a new folder
    b. Put the following files in the folder
        - *code.txt*
        - *log.txt*
        - *output.txt*
    c. Stage the *code.txt* and *output.txt* files
    d. Commit them
    e. And finally push them to GitHub
2. Please share the commands for the above steps.

Question 2

- Create a Git working directory with *feature1.txt* and *feature2.txt* in the master branch.
- Create 3 branches develop, feature1 and feature2.
- In develop branch create *develop.txt*, do not stage or commit it.

- Stash this file and check out to feature1 branch.
- Create a *new.txt* file in feature1 branch, stage and commit this file.
- Checkout to develop, unstash this file and commit.
- Please submit all the Git commands used to perform the above steps.

Question 3

1. Create a Git working directory, with the following branches:
   - develop
   - f1
   - f2
2. In the master branch, commit the *main.txt* file.
3. Put *develop.txt* in develop branch, *f1.txt* and *f2.txt* in f1 and f2.
4. Push all these branches to GitHub.main.txt
5. On local delete f2 branch.
6. Delete the same branch on GitHub as well.

Question 4

1. Put *master.txt* on master branch, stage and commit.
2. Create 3 branches: public 1, public 2 and private.
3. Put *public1.txt* on public 1 branch, stage and commit.
4. Merge public 1 on master branch.
5. Merge public 2 on master branch.
6. Edit *master.txt* on private branch, stage and commit.
7. Now update branch public 1 and public 2 with new master code in private.
8. Also update new master code on master.
9. Finally update all the code on the private branch.

Question 5

1. Create a Git flow workflow architecture on Git
2. Create all the required branches
3. Starting from the feature branch, push the branch to the master, following the architecture
4. Push a *urgent.txt* on master using hotfix

Question 6

1. Create a new folder and initialize it as a Git repository.
2. Inside the folder, create the following files:
    - *notes.txt*
    - *secrets.txt*
    - *image.png*
3. Now, create a ***.gitignore*** file with the following content.
    - *secrets.txt*
    - **.png*
4. Run ***git status*** and verify that.
    - *secrets.txt* and *image.png* are not shown and tracked.
    - Only *notes.txt* is Visible in Git status

Question 7

Set up SSH authentication with GitHub to push code without entering a password.

1. Generate an SSH key on your system.
2. Add the public key to your GitHub account under SSH Keys.
3. Test the SSH connection to GitHub.
4. Clone any repository using the SSH URL.

5. Make a small change, commit it, and push it to GitHub.
6. Ensure the push happens without asking for a username or password.
7. Submit a screenshot of:
    - SSH key added in GitHub
    - Terminal showing SSH connection success
    - Terminal showing successful Git push