Academic Tasks (29462)

SET A

Academic Task Number: 3 | **Course Code:** INT 331
Date of allotment: __/11/2024 | **Course Title: Fundamentals of DevOps**
Academic Task Type: Practical | **Maximum Marks: 30**

Question 1:

You are part of a team collaborating on a project in Git. The team is divided into two groups, Team A and Team B, each working on their respective branches. The task is to manage changes collaboratively in a single file.:

1. **Branch Creation:**
   Create two branches, **team-a-branch** and **team-b-branch**, from the main branch.

2. **Team A's Changes:**
   In team-a-branch, add the line "**Team A made a change**" to **shared.txt**, commit the change, and merge it into main.

3. **Team B's Changes:**
   In team-b-branch, pull the latest main changes, add the line "**Team B made a change**" to shared.txt, and commit the change.

4. **Conflict Resolution:**
   If a conflict occurs when merging team-b-branch into main, resolve the conflict, ensuring both Team A's and Team B's changes are retained.

Question 2:

You want to compare the changes made between the last commit and the current state of your working directory.

1. What command would you use to see these changes?

2. How can you use the same command to include differences in staged files?

Academic Tasks (29462)

Question 3:

You have uncommitted changes in your working directory but need to switch to another branch without committing these changes.

1. What command would you use to temporarily save your changes using Git Stash?
2. How can you list all the stashed changes in your repository?
3. What command would you use to reapply the most recently stashed changes

Academic Tasks (29462)

SET B

Academic Task Number: 3 | **Course Code:** INT 331
Date of allotment: __/11/2024 | **Course Title: Fundamentals of DevOps**
Academic Task Type: Practical | **Maximum Marks: 30**

Question 1

Scenario: Team Collaboration on a Document:
Create two branches, team-x and team-y, from main. Team X adds a new paragraph to collab.txt, commits, and merges their changes into main. Team Y then updates the same file with a new line, commits, and merges it into main. Resolve conflicts if they occur.

Question 2:

You need to compare two specific commits in your Git history, identified by their hashes (abc123 and def456).

1. What command would you use to view the differences between these two commits?
2. How can you add a file name to this command to limit the comparison to a specific file?

Question 3: You have uncommitted changes in your working directory but need to switch to another branch without committing these changes.

1. What command would you use to temporarily save your changes using Git Stash?
2. How can you list all the stashed changes in your repository?
3. What command would you use to reapply the most recently stashed changes?

OR

Academic Tasks (29462)

Question 4:

Scenario: You are working on a project called **"FeatureX"** in a Git repository. Your team follows a branching strategy where all new features are developed in feature branches and later merged into the develop branch.

1. Create a new branch named feature/FeatureX from the develop branch.
2. Make a change to a file (e.g., README.md) in the feature/FeatureX branch and commit it.
3. Meanwhile, another team member has updated the develop branch with a hotfix. Simulate this by creating a new branch named hotfix/UpdateDevelop, making a commit, and merging it into develop.
4. Merge the latest changes from develop into your feature/FeatureX branch to ensure you are up-to-date. Resolve any merge conflicts if they arise.
5. Finally, merge your feature/FeatureX branch into develop to integrate your changes.

Academic Tasks (29462)