Lab Exercise: Basic GitHub Tasks for DevOps Beginners

Objective:

This lab will guide you through the essential GitHub tasks relevant for a beginner in **DevOps**. You will learn how to create a repository, clone it, make changes, commit those changes, push them to GitHub, and work with branches.

Prerequisites:

- A GitHub account (Sign up at github.com if you don't have one).
- Git installed on your local machine. (Download available at git-scm.com/downloads).

Step 1: Create a GitHub Repository

1. **Log in to GitHub**:
   - Navigate to GitHub and log in with your credentials.
2. **Create a new repository**:
   - In the upper-right corner, click the **+** icon and select **New repository**.
   - **Repository name**: Enter a name for your repository (e.g., devops-beginner).
   - **Description**: Optionally, add a description (e.g., "DevOps Beginner Repository").
   - **Visibility**: Choose **Public**.

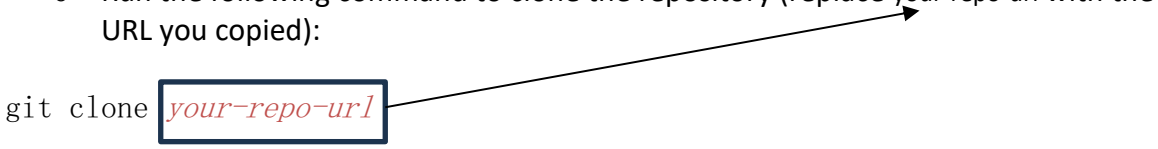
     Public Repository: *Anyone on the internet can view and access the repository including its code. For open-source projects where you want to share your work with others. Do not store sensitive data here! (Password, access keys, etc.)*

     Private Repository: *Only invited collaborators or members of the organization can access the repository. Ideal for private or sensitive work that you don't want to share with the public.*
   - Check the box that says **Initialize this repository with a README**.
   - Click **Create repository**.

Step 2: Clone the Repository to Your Local Machine

This allows offline development of code. You can develop features, fix bugs, and test locally, and push those changes up to the remote repository. It also provides a local backup of the project.

1. Copy the repository URL:
    - After creating the repository, you will be redirected to the repository's page.
    - Click the **Code** button and copy the repository's URL (it should look like https://github.com/username/repo-name.git).
2. Clone the repository:
    - Open a terminal or Git Bash on your local machine.
    - Run the following command to clone the repository (replace your-repo-url with the URL you copied):



    - You now have a local copy of your GitHub repository.
3. Navigate to the cloned directory:
    - Change the directory to your new repository:

```
cd devops-beginner
```

Step 3: Make Changes and Commit

1. Create a new file:

    This step simulates adding new code or a new file to your project.
    - In your local repository, create a new file called hello.txt and open it with a text editor.
    - Add the following content to the file:

```
Hello, DevOps Beginner!
```

2. Check the repository status:
    - Run the following command to check the status of your repository:

```
git status
```

    - You should see `hello.txt` listed as an untracked file. (i.e., a file that hasn't been added to version control yet).

3. Stage the file:
    - Stage the `hello.txt` file for commit by running:

      ```
      git add hello.txt
      ```

4. **Commit the changes**:
    - Add your user info and commit your changes with a message:

      ```
      git config --global user.name "Your Name"

      git config --global user.email "your-email@example.com"

      git add hello.txt

      git commit -m "Add hello.txt with a welcome message"
      ```

      The git commit command records the changes you have staged (in this case, the addition of hello.txt) in the repository's history. The -m flag allows you to add a commit message, which in this case is "Add hello.txt with a welcome message." This step finalizes the changes and adds them to your local Git repository, simulating saving a snapshot of your work.

Step 4: Push Changes to GitHub

1. **Push changes to GitHub**:
    - Push your local changes to the remote repository (GitHub):

      ```
      git push origin main
      ```

    - After the push, your changes should now appear in the GitHub repository.
2. **Verify changes on GitHub**:
    - Go back to GitHub and refresh your repository page. You should see the `hello.txt` file in the list of repository files.

Step 5: Work with Branches

Branches in GitHub allow you to work on different features or changes in isolation from the main codebase. This way, you can develop, test, and experiment without affecting the main project. Once the work on a branch is complete and stable, it can be merged back into the main branch.

1. **Create a new branch**:
    - In your terminal, create a new branch for feature development:

      ```
      git checkout -b new-feature
      ```

2. **Make changes in the new branch**:
   This simulates making updates or improvements to your code or project within the isolated branch. The changes remain within the new-feature branch and don't impact the main branch.

    - Open hello.txt and add another line:

      ```
      Welcome to the world of DevOps!
      ```

    - Save the file.

3. **Stage and commit the changes**:
    - Stage the changes:

      ```
      git add hello.txt
      ```

    - Commit the changes:

      ```
      git commit -m "Add a welcome message to hello.txt"
      ```

4. **Push the new branch to GitHub**:

   ```
   git push origin new-feature
   ```

Step 6: Create a Pull Request

1. **Navigate to your repository on GitHub**:
    - Go to your repository's page on GitHub.
2. **Create a pull request**:
    - GitHub will detect that you recently pushed a new branch (new-feature) and will prompt you to **Compare & pull request**. Click that button.
    - In the pull request, you can add comments explaining your changes.
    - Click **Create pull request**.

      GitHub detects the new branch (new-feature) and offers an option to compare it with the main branch and create a pull request. A pull request (PR) is a way to

propose merging the changes from your feature branch into the main branch. In this step, you provide a description of the changes you made, which simulates explaining your work to collaborators or reviewers.

3. **Review and merge the pull request**:
    - After creating the pull request, you or another team member can review the changes.
    - Once reviewed, click **Merge pull request**.
    - Confirm the merge, and your changes will be merged into the main branch.

Step 7: Clean Up Local Branch

1. **Switch back to the main branch**:
    - In your terminal, switch to the main branch:

      ```
      git checkout main
      ```

2. **Pull the latest changes from GitHub**:
    - Pull the latest changes to your local main branch:

      ```
      git pull origin main
      ```

3. **Delete the new-feature branch**:
    - Now that the changes have been merged, you can delete the new-feature branch:

      ```
      git branch -d new-feature
      ```

Step 8: Review Commit History

1. **View the commit history**:
    - To review your commit history, run the following command:

      ```
      git log --oneline
      ```

2. **Explore the commits**:
    - You will see a list of commits, including the messages you used when committing changes (e.g., "Add hello.txt" and "Add a welcome message to hello.txt").

Step 9: Clean Up Repository (Optional)

- **Delete the repository on GitHub**:
    1. If you want to clean up the repository, go to the repository page on GitHub.
    2. Click **Settings** (in the top-right corner).
    3. Scroll down to the **Danger Zone** and click **Delete this repository**.
    4. Confirm the deletion.