#90DaysOfDevOps Challenge - Day 9 - Deep Dive in Git & GitHub for DevOps Engineers

Welcome to Day 9 of the #90DaysOfDevOps challenge. Today, we will take a deep dive into Git & GitHub, two essential tools in the DevOps ecosystem. We will explore the importance of Git, understand the difference between the main branch and master branch, and clarify the distinction between Git and GitHub. Get ready to enhance your version control skills and strengthen your collaboration capabilities.

What is Git and why is it important?

Git is a distributed version control system designed to track changes in source code during software development. It provides a robust and efficient way to manage the codebase, track revisions, and facilitate collaboration among team members. With Git, you can easily work on different branches, merge changes, and revert to previous versions, ensuring a streamlined and organized development process. Git's importance lies in its ability to improve code quality, enable collaboration, and facilitate project management in an efficient and scalable manner.

What is the difference between Main Branch and Master Branch?

Historically, the default branch name in Git repositories was "master." However, recognizing that the term "master" has negative connotations due to its association with slavery, many projects and organizations have transitioned to using alternative branch names like "main" or "primary" as a more inclusive and respectful terminology. Therefore, the difference between the main branch and the master branch is essentially a matter of naming convention, where "main" is considered a more inclusive and preferred term.

Can you explain the difference between Git and GitHub?

Git and GitHub are often used interchangeably, but they serve different purposes in the software development process. Git is a distributed version control system that allows developers to manage and track changes to source code. It provides a local repository on each developer's machine, enabling them to work offline, commit changes, and branch out for different features or experiments.

On the other hand, GitHub is a web-based hosting service for Git repositories. It provides a centralized platform for collaboration, code sharing, and project management. GitHub allows multiple developers to work on the same codebase, collaborate through pull requests, manage issues, and host documentation. It also provides additional features like project boards, continuous integration, and deployment workflows, making it a powerful tool for DevOps teams.

How do you create a new repository on GitHub?

To create a new repository on GitHub, follow these steps:

1. Visit the GitHub website (https://github.com/) and log in to your account (or create a new account if you don't have one).
2. Click on the "+" button on the top-right corner of the page and select "New repository" from the dropdown menu.

3. Provide a name for your repository, choose whether it should be public or private, and add an optional description.
4. Select any additional options you prefer, such as initializing the repository with a README file, adding a .gitignore file, or choosing a license.
5. Click on the "Create repository" button to create your new repository.

Congratulations! You have successfully created a new repository on GitHub.

What is the difference between a local and a remote repository? How to connect local to remote?

A local repository refers to the repository stored on your local machine, typically created using Git. It contains all the necessary files, commits, branches, and version history related to a project. The local repository allows you to work on code, make changes, create new branches, and commit your changes locally without the need for an internet connection.

A remote repository, such as those hosted on platforms like GitHub, acts as a central server where you can store and share your code with others. It serves as a collaboration hub, enabling multiple developers to access, contribute, and synchronize their work on a project. Remote repositories allow for team collaboration, version control, and easy sharing of code across different machines.

To connect your local repository to a remote repository on GitHub, follow these steps:

1. Create a new repository on GitHub using the steps mentioned earlier.
2. On your local machine, navigate to the directory containing your Git repository using the command line or a Git client.
3. Add the remote repository URL to your local repository using the command:

   ```
   git remote add origin <remote_repository_url>
   ```

   Replace `<remote_repository_url>` with the URL of your remote repository on GitHub.
4. Verify the connection between your local and remote repositories using the command:

   ```
   git remote -v
   ```

   This will display the remote repository URL associated with your local repository.
5. You can now push your local commits to the remote repository using the command:

   ```
   git push origin <branch_name>
   ```

Replace `<branch_name>` with the name of the branch you want to push.

By following these steps, you have successfully connected your local repository to a remote repository on GitHub.

Tasks

Now, let's dive into the hands-on exercises to solidify your understanding of Git and GitHub.

Exercise 1: Set your user name and email address

Before proceeding, it's important to set your user name and email address, which will be associated with your commits. Use the following commands:

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

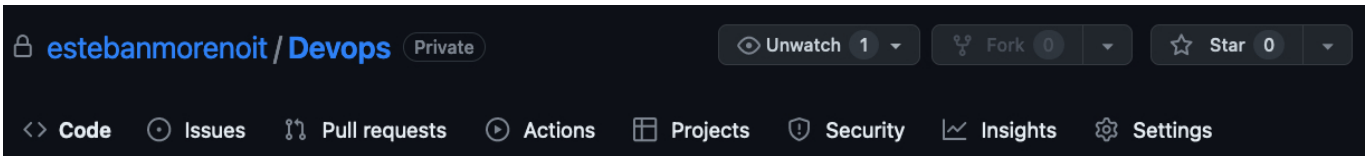
Replace "Your Name" with your actual name and "yourname@example.com" with your email address.

To check if your Git configuration is set up correctly, you can use the following command:

```
git config --list
```

Exercise 2: Create a repository named "Devops" on GitHub

1. Visit the GitHub website and log in to your account.
2. Follow the steps mentioned earlier to create a new repository named **"Devops"** on GitHub.



Exercise 3: Connect your local repository to the repository on GitHub

1. On your local machine, navigate to the directory where you want to create your local repository.
2. Initialize a new Git repository using the command:

```
git init
```

3. Add the remote repository URL to your local repository using the command:

```
git remote add origin <remote_repository_url>
```

Replace `<remote_repository_url>` with the URL of your "Devops" repository on GitHub.

4. Verify the connection between your local and remote repositories using the command:

```
git remote -v
```

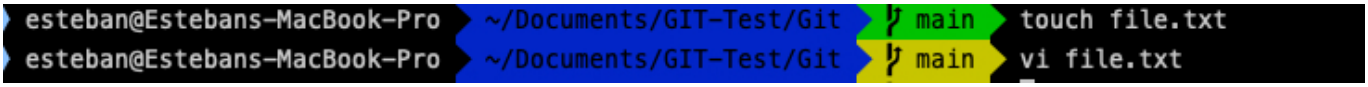
This should display the remote repository URL associated with your local repository.

Exercise 4: Create a new file in Devops/Git/Day-02.txt & add some content to it

1. Create a new file named "Day-02.txt" within the "DevOps/Git" directory in your local repository.
2. Add some content to the file, such as:

```
This is a test file
```

3. Save the file.



Exercise 5: Push your local commits to the repository on GitHub

1. Commit your changes using the following commands:

```
git add .
git commit -m "Added new file"
```

2. Push your local commits to the "Devops" repository on GitHub using the command:

```
git push origin master
```

Congratulations! You created a repository on GitHub, connected your local repository to the remote repository, created a new file, and pushed your changes to GitHub.

In this article, we explored the significance of **Git** and its role in **version contro**l. We clarified the difference between the **main branch** and the **master branch**, and demystified the distinction between **Git** and **GitHub**. Additionally, we walked through the steps to create a **new repository** on GitHub, understand the difference between **local** and **remote repositories**, and connect them seamlessly.

Get ready for Day 10, where we'll dive deeper into advanced Git concepts and explore powerful collaboration workflows using branches, merging, and pull requests. Stay tuned!