#90DaysOfDevOps Challenge - Day 8 - Basic Git & GitHub for DevOps Engineers

What is Git?

Git is a **version control system** that allows you to track changes to files and coordinate work on those files among multiple people. It is commonly used for software development, but it can be used to track changes to any set of files.

With Git, you can keep a record of who made changes to what part of a file, and you can revert to earlier versions of the file if needed. Git also makes it easy to collaborate with others, as you can share changes and merge the changes made by different people into a single version of a file.

What is GitHub?

GitHub is a web-based platform that provides hosting for version control using Git. It is a subsidiary of Microsoft, and it offers all the distributed version control and source code management (SCM) functionality of Git, as well as adding its own features. GitHub is a very popular platform for developers to share and collaborate on projects, and it is also used for hosting open-source projects.

What is Version Control? How many types of version controls do we have?

Version control is a system that tracks changes to a file or set of files over time so that you can recall specific versions later. It allows you to revert files to a previous state, revert the entire project to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more.

There are two main types of version control systems: centralized version control systems and distributed version control systems.

1. A **Centralized Version Control System (CVCS)** uses a central server to store all the versions of a project's files. Developers "check out" files from the central server, make changes, and then "check-in" the updated files. Examples of CVCS include Subversion and Perforce.

2. A **Distributed Version Control System (DVCS)** allows developers to "clone" an entire repository, including the entire version history of the project. This means that they have a complete local copy of the repository, including all branches and past versions. Developers can work independently and then later merge their changes back into the main repository. Examples of DVCS include Git, Mercurial, and Darcs.

Why do we use distributed version control over centralized version control?

1. **Better collaboration:** In a DVCS, every developer has a full copy of the repository, including the entire history of all changes. This makes it easier for developers to work together, as they don't have to constantly communicate with a central server to commit their changes or to see the changes made by others.

2. **Improved speed:** Because developers have a local copy of the repository, they can commit their changes and perform other version control actions faster, as they don't have to communicate with a central server.
3. **Greater flexibility:** With a DVCS, developers can work offline and commit their changes later when they do have an internet connection. They can also choose to share their changes with only a subset of the team, rather than pushing all of their changes to a central server.
4. **Enhanced security:** In a DVCS, the repository history is stored on multiple servers and computers, which makes it more resistant to data loss. If the central server in a CVCS goes down or the repository becomes corrupted, it can be difficult to recover the lost data.

Overall, the decentralized nature of a DVCS allows for greater collaboration, flexibility, and security, making it a popular choice for many teams.

Task 1: Install Git

To begin, we need to install Git on your computer. Git is available for various operating systems, including Windows, macOS, and Linux. Follow the steps below:

1. Visit the official Git website at **https://git-scm.com/downloads**.
2. Download the appropriate installer for your operating system.
3. Run the installer and follow the on-screen instructions to complete the installation.
4. After the installation is complete, open a terminal or command prompt and type `git --version` to verify that Git is installed correctly. You should see the version number displayed.

Congratulations! You have successfully installed Git on your computer. Now, let's move on to the next task.

Task 2: Create a GitHub Account

GitHub is a widely used platform for hosting Git repositories and collaborating on projects. If you don't already have a GitHub account, follow these steps to create one:

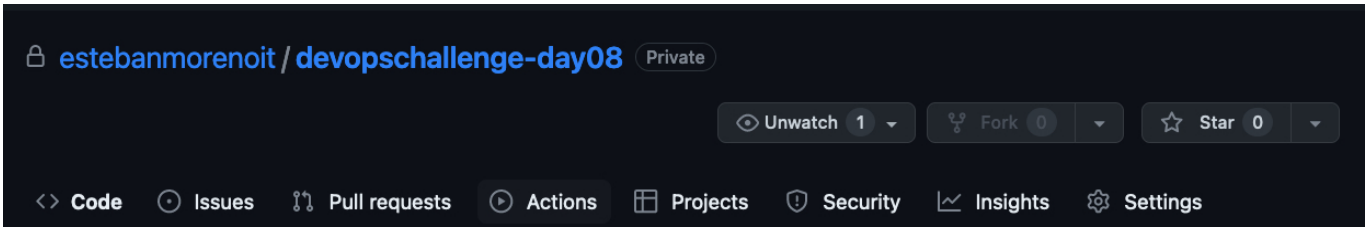
1. Open your web browser and go to **https://github.com**.
2. On the GitHub homepage, click on the "Sign up" button.
3. Fill in the required information, including your desired username, email address, and password.
4. Choose a plan (either Free or one of the paid plans) based on your needs.
5. Complete the verification process, which may involve solving a CAPTCHA or verifying your email address.

Once you have completed these steps, you will have successfully created a GitHub account.

Exercise 1: Create a New Repository on GitHub

To begin, we will create a new repository on GitHub, which will serve as the central location for storing and managing your code. Follow the steps below:

1. Open your web browser and go to **https://github.com**.
2. Log in to your GitHub account.
3. On the GitHub homepage, click on the "+" button in the top-right corner and select "New repository" from the dropdown menu.
4. Give your repository a meaningful name.
5. Optionally, describe your repository to provide more context.
6. Choose the repository's visibility (public or private) based on your requirements.
7. Click on the "Create repository" button to create the repository.

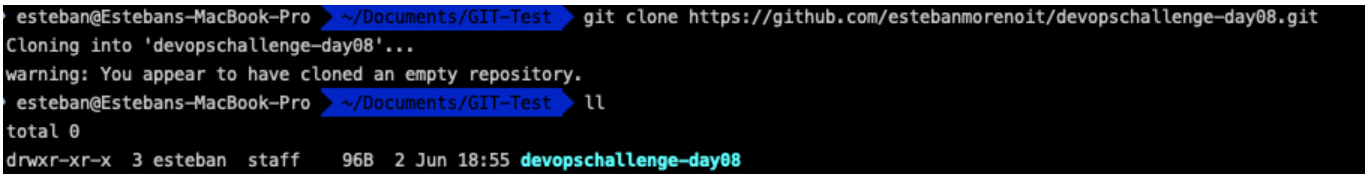


Congratulations! You have successfully created a new repository on GitHub. Now, let's move on to the next exercise.

Exercise 2: Clone the Repository to Your Local Machine

To work on the repository locally, we need to clone it to your computer. Follow these steps:

1. On the repository page on GitHub, click on the "Code" button.
2. Copy the URL of the repository.
3. Open a terminal or command prompt on your local machine.
4. Navigate to the directory where you want to clone the repository.
5. Use the `git clone` command followed by the repository URL to clone the repository. For example: `git clone https://github.com/your-username/your-repository.git`
6. Press Enter to execute the command.

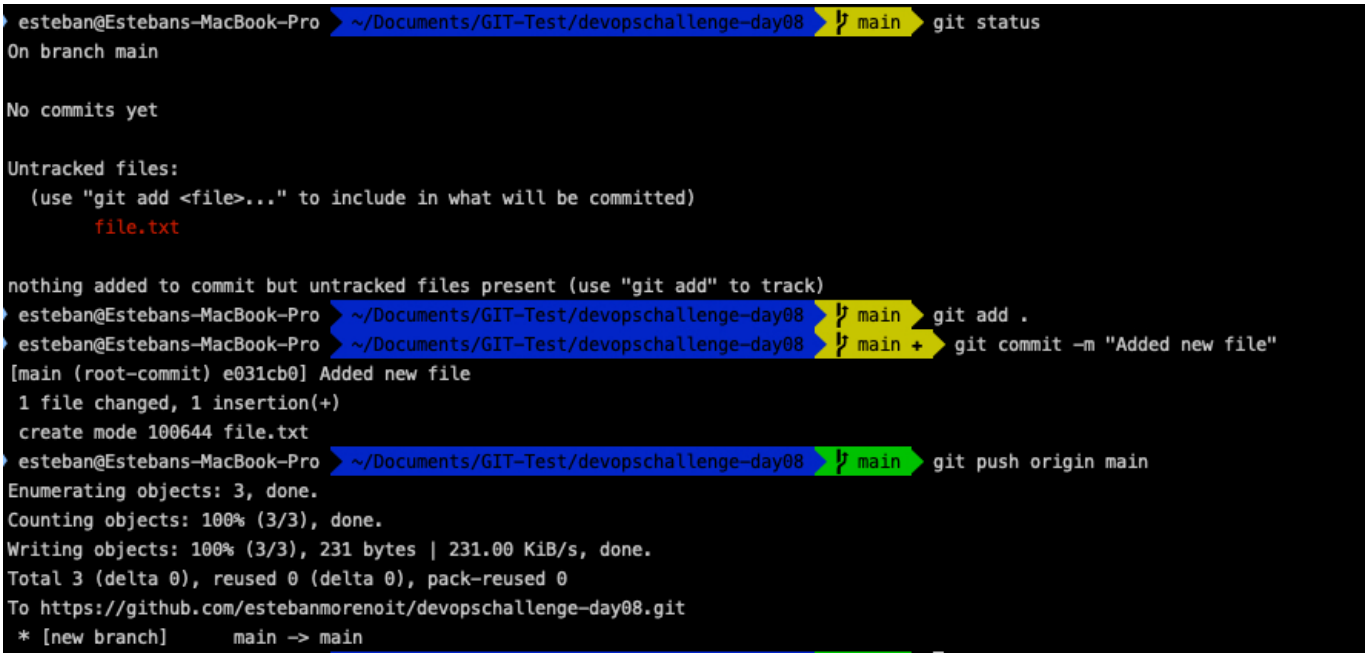


Great! You have successfully cloned the repository to your local machine. Let's proceed to the next exercise.

Exercise 3: Make Changes, Commit, and Push

Now that you have the repository cloned locally, you can make changes to the files and commit them to track the modifications. Follow these steps:

1. Open the cloned repository in your preferred text editor or IDE.
2. Make the desired changes to the files in the repository.
3. Save the changes.
4. Open a terminal or command prompt in the root directory of the cloned repository.
5. Use the `git status` command to view the changes you made. It will show the modified files.
6. Use the `git add` command followed by the file names to stage the changes for commit. For example: `git add filename.txt` or `git add .` to stage all changes.
7. Use the `git commit` command to commit the changes with a meaningful message describing the modifications. For example: `git commit -m "Added new feature"` or `git commit -m "Fixed a bug"`
8. Finally, use the `git push` command to push the committed changes back to the repository on GitHub. For example: `git push origin main` or `git push origin master`, depending on the branch name.



Congratulations! You've successfully completed Day 8 of the #90DaysOfDevOps challenge. Today, you learned the basics of Git and GitHub, including creating a new repository, cloning it to your local machine, making changes, committing them, and pushing them back to GitHub. These fundamental exercises are essential for version control and collaborative software development.

Stay tuned for Day 9, where we'll take a deep dive into Git & GitHub for DevOps Engineers. We'll cover advanced topics, branching strategies, merging, resolving conflicts, and more. It's going to be an exciting and insightful day!