**Introduction to Git: A Beginner's Lesson**

**What is Git?**

Git is a distributed version control system used to track changes in source code during software

development. It allows multiple developers to work on a project simultaneously without interfering

with each other's changes.

**Why Use Git?**

1. Collaboration: Multiple developers can work on the same project.

2. Version Control: Keeps track of every change made to the code.

3. Backup: Provides a reliable backup of your code.

4. Branching and Merging: Allows you to work on different features simultaneously.

**Basic Git Terminology**

Repository (Repo): A directory which contains your project work, including a .git folder, where Git

stores all the metadata and object database.

Commit: A snapshot of your repository at a specific point in time.

Branch: A movable pointer to a commit. The default branch name in Git is main.

Clone: A copy of a repository.

Push: Sending your committed changes to a remote repository.

Pull: Fetching and merging changes from a remote repository to your local repository.

Merge: Combining changes from different branches.

**Setting Up Git**

1. Install Git: Download and install Git from https://git-scm.com/.

**Introduction to Git: A Beginner's Lesson**

2. Configure Git:

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

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

**Basic Git Commands**

**Git Local:**

**Initializing a Repository**

git init

**Checking the Status**

git status

**Adding Changes**

git add filename

or add all changes

git add .

**Committing Changes**

git commit -m "Your commit message"

**Introduction to Git: A Beginner's Lesson**

**Viewing Commit History**

git log

**Branching**

git branch <branch\_name>

git checkout <branch\_name>

**or create and switch**

git checkout -b <branch\_name>

**Merging**

git merge <branch\_name>

**Switch commit**

git checkout <commit-hash>

**Git Cloud:**

**Cloning a Repository**

git clone <https://github.com/user/repo.git>

**Or, push the local repository to remote**

git remote add origin <https://github.com/user/repo.git>

git branch -M main

git push -u origin main

**Why You Need git branch -M main:**

* **Setting the Correct Default Branch**: If your local branch is named something other than main (for example, master or development), you need to rename it to main to match the naming convention, especially if the remote repository is expected to have a main branch. By running git branch -M main, you ensure that your local branch matches the desired naming standard for the remote repository.
* **Consistency Across Environments**: Many developers are moving towards using main as the primary branch name due to inclusivity and standardization. Renaming your branch helps maintain consistency across different environments and developers.
* **Required for the Push**: The push command will attempt to push the currently checked-out branch. If your current branch is not named main, and you intend to push a branch named main to the remote, you'll need to rename it first.

**Benefits of Using -u:**

1. **Simplified Commands**: After running git push -u origin main, you can simply use git push or git pull in the future without having to specify the remote or branch name.
2. **Tracking Branches**: It establishes a tracking relationship, allowing you to see the status of the remote branch relative to your local branch with commands like git status.

**Pushing Changes**

git push origin <branch\_name>

**Or, if used -u before**

git push

**Pulling Changes**

git pull origin <branch\_name>

**Or, if used -u before**

git pull

**How to Back to specific commit and push to remote:**git revert <commit-hash>

git push

**Basic Workflow**

1. Clone the repository (if you haven't already):

git clone https://github.com/user/repo.git

cd repo

**Introduction to Git: A Beginner's Lesson**

2. Create a new branch for your work:

git checkout -b new-feature

3. Make changes to your files.

4. Check the status of your changes:

git status

5. Add your changes to the staging area:

git add .

6. Commit your changes:

git commit -m "Add new feature"

7. Push your changes to the remote repository:

git push origin new-feature

8. Create a pull request on the remote repository to merge your changes.

**Conclusion**

Git is a powerful tool that can greatly enhance your productivity and collaboration in

development. This lesson covers the basics, but there is much more to learn. As you become more

comfortable with Git, explore more advanced features like rebasing, cherry-picking, and using

hooks. Happy coding!