# **Lecture:** Introduction to Git and GitHub (Part 1)

- **Teacher:** Sir Arsalan Shah
- **Typist:** Yousuf Naveed

## Cloning Repositories

- To clone a repository from GitHub, follow these steps:

1. Go to your GitHub repository.
2. Copy the repository URL from the "Code" button.
3. After copying the Repository's URL, use the following command to clone the repository: **`git clone YourRepoURL`**

- To clone from a specific branch, use the following command:

**`git clone -b BranchName YourRepoURL`**

## Removing Files

There are two methods to remove files:

1. From the staging area
2. From the working directory.

To remove a file from the staging area, you can use the following command:

**`git rm YourFileName`**

To remove a file from the working directory, use the following command:

**`git rm --cached YourFileName`**

> Removing a file from the working directory means that the file will become untracked.

# Undo Changes

Undoing changes can be done in three different cases:

1. Reset from the staging area:
    - For one file: **`git reset YourFileName`**
    - For all files: **`git reset`**

2. Reset from the last commit: **`git reset head~1`**

3. Reset from a specific commit:
    - First, use `git log` to copy the commit hash of the file you want to reset.
    - Then, use the following command: **`git reset YourCommit'sHash`**
    - If you want to reset the code along with the file, use: **`git reset –hard YourCommit'sHash`**

# Git Merge

Merging two branches can be done in two ways:

1. **Merge via a Pull Request:**
    - Navigate to your GitHub repository.
    - Click on "Compare & pull request."
    - If there are no conflicts, click "Create pull request," and then "Merge pull request."
    - Confirm the merge.

2. **Merging with potential conflicts:**
    - Ensure there are no conflicts in both branches by adding and committing all files.
    - If conflicts arise, use the `git merge BranchName` command to manually resolve them.

# Pull Command

To update your local repository with changes from the remote repository, use the

pull command: **`git pull origin master`**

# Alias Command

You can create alias commands for lengthy ones, making them shorter for

convenience.

**For example:**

`git config --global alias.st status`

Now, instead of typing **`git status`**, you can use **`git st`**.

# Conclusion

In this second part of our "Introduction to Git and GitHub" series, we delved deeper into essential Git concepts and advanced techniques. From cloning repositories and removing files to understanding undoing changes and merging branches, you've gained a strong foundation in using Git for version control and collaborating on coding projects.

These skills are not only valuable for individual developers but also essential when working with teams on software development projects. Git and GitHub provide powerful tools for tracking changes, managing code, and ensuring a seamless collaborative workflow.

As you continue your journey into the world of Git and GitHub, remember that practice makes perfect. The more you use these tools, the more proficient you'll become. Embrace version control as a fundamental aspect of modern software development, and you'll find it indispensable in your coding endeavors.