Introduction to Git and Github

What is Git?

What is Git?

Git is a version control system that allows you to track changes in your code and collaborate with others.

What is GitHub?

What is GitHub?

GitHub is a web-based platform that provides hosting for Git repositories and offers additional features such as issue tracking and project management.

Why are they important?

Why are they important?

- Using version control makes it easier to track changes and revert to previous versions if necessary.
- Collaborating with others on code is much easier when using version control.

Important concepts

Here are the topics we will discuss today

- Repository
- Stage
- Commit
- Push
- Pull
- Branch
- Merge
- Conflict
- Pull request
- Additional features in Git

What is a repository?

Repository

A repository is a directory or folder that contains files and metadata for a project under version control.

- **Local Repository**: A Git repository on a local machine.
- **Remote Repository**: A Git repository on a remote server, such as Github.

Syncing: Keeping the local and remote repositories up-to-date with each other.

How do you create a repository?

Creating a repository on GitHub

1. Log in to your GitHub account and click on the "+" sign on the top right corner of your screen.
2. Select "New Repository" from the dropdown menu.
3. Enter a name for your repository.
4. Provide a description for your repository.
5. Choose whether you want your repository to be public or private.
6. Select "Initialize this repository with a README".
7. Click on the "Create Repository" button to create your new repository.
8. Copy the repository URL as HTTPS in the next page.

Initializing a repository using git

1. On your computer, create a directory with the name “git-tutorial”.
2. Open your command prompt or terminal window.
3. Navigate to the directory.
4. Run the command "`git init`" to initialize a new Git repository in the directory.
5. Create a README.md file using the command “`git add README.md`”
6. Run the command “`git remote add origin <URL>`”

Why do we need a README file?

Why do we need a README

- Provides context
- Helps with documentation
- Enhances collaboration
- Shows professionalism

Cloning

Cloning

- Cloning a repository on GitHub allows you to make a copy of the repository on your local machine.
- This is useful if you want to work on the code locally, make changes, and then push those changes back to the GitHub repository.
- You can a clone a repository using this command:
  - `git clone <URL>`

Best practices

- Use version control for all projects
- Keep repositories clean and organized
- Collaborate effectively

Basic operations

Staging and Committing

Staging

- Staging is the process of preparing changes in your code before committing them to your Git repository.
- It allows you to review and organize your changes.

Committing

- A commit is a way of saving changes to your local Git repository.
- Each commit represents a snapshot of your project at a particular point in time.
- Commits are like checkpoints, allowing you to track changes to your project over time.

How to stage your changes

- To stage changes in Git, you use the "git add" command followed by the name of the file(s) you want to stage.
- For example, to stage a file named "file.txt", you would run the command "`git add index.html`".

Staging Multiple Files

Staging Multiple Files

- You can also stage multiple files at once by separating their names with spaces, like this: "`git add index.html style.css script.js`".
- You can also stage all your files that have been changed using “`git add .`”

Git status

- Git Status is a command that displays the current status of your Git repository.
- To check the status of your repository, run the command "`git status`" in your terminal.
- The output will also tell you if you have any untracked files in your repository.

Exercise

- Create a new file named "newfile.txt" in your local repository.
- Add some content to the file, such as "Hello, world!".
- Use the "`git status`" command to check the status of your changes.
- Stage the changes by running the command "`git add newfile.txt`".
- Use the "`git status`" command again to verify that the changes have been staged successfully.

How to commit changes

- Change the content on "newfile.txt" to “Hello, your name”
- Stage the changes you want to commit by using the git add command.
- Commit the changes using the git commit command, along with a descriptive commit message.
  - `git commit -m “message”`

Commit the changes in your repository

Best practices

- Commit frequently
- Only stage what changes you’re confident with
- Write descriptive commit messages
- Use atomic commits (let it represent a single contained change)
- Use simple present tense
    - Example : “Update Login Page”

Push

Git push

Pushing changes in Git refers to sending your local changes to the remote repository.

Pushing changes allows others to access your changes and collaborate on the project.

How to push changes

- Push the changes to the remote repository using the git push command.
  - `git push`
- If prompted, enter your GitHub username and password to authenticate the push

Push the changes

Pull

Git pull

- Pulling in Git refers to retrieving and merging changes from a remote repository to your local repository.
- This is useful when others have made changes to the remote repository that you want to incorporate into your local repository.

Exercise

- Modify the content of the file you pushed on GitHub.
- Commit the changes to the repository on GitHub using the GitHub UI.
- On your local machine, navigate to the repository directory in your terminal and use the `git pull` command to retrieve the changes from the remote repository:
- Check the contents of the file you created on GitHub to confirm that the changes have been successfully pulled to your local repository.

Branch

Branch

- A branch is a separate line of development that allows you to work on a feature or fix without affecting the main codebase.
- It allows multiple developers to work on the same codebase without conflicting change
- Two ways to create a branch:

```
1.  git checkout -b [your-name].NewFile
2.  git branch [your-name].NewFile

    git checkout [your-name].NewFile
```

- Switch to a branch:
  - `git checkout branch_name`

Exercise

- Create a new branch using the command
  - `git branch  [your-name].NewFile`
- Switch to the new branch using the command
  - `git checkout [your-name].NewFile`
- Make changes to the code and commit the changes using the command
- Push the changes

Best practices

- Branch off of the main branch
- Keep branches small and focused (delete them after they’ve served their purpose)
- Delete old branches

How do we combine people’s work?

Merge

Merging

- A merge is the process of combining changes from one branch into another branch.
- It is used to integrate the changes made in one branch into the main codebase and vice versa.

How to merge branches

- Checkout the branch you want to merge into (e.g., main):

```
git checkout main
```

- Pull any changes from the remote repository:

```
git pull
```

- Merge the other branch into the current branch (e.g., feature-branch):

```
git merge feature-branch
```

Conflict

Conflicts in git

A conflict occurs when two or more branches have made changes to the same code in a conflicting manner.

Git is designed to handle conflicts and allows you to resolve them manually.

Bringing changes from your feature branch to main

Pull Request

Pull request

- A Pull Request is a feature in GitHub that allows users to propose changes to a repository and collaborate on those changes with other users
- It is often used in open-source projects where multiple people may be contributing to the same codebase.

How to create a pull request

- Navigate to your repository on GitHub and click the "New pull request" button.
- Review your changes:
- Submit your Pull Request
- Respond to feedback
- Congratulations, you've successfully prepared a Pull Request on GitHub!

Create a pull request from your feature branch to the main branch

More on Git

Rebase

Rebase

Rebasing is the process of moving a branch to a new base commit. It is used for combining multiple commits on to a single one. It has the same effect on the code as merge at the end of the day.

- Command: `git rebase <branch-name>`

Inside the repository, configure Git to always rebase when pulling changes by setting the pull.rebase configuration option to true(**Use git config pull.rebase true**). You can add “--global” to the command if you want to set it up for all repositories.

Rebase Example

1. Alice commits a bugfix on top of a commit by Bob:

```
    git checkout bob-branch
    git commit -m "Bugfix"
```

2. Bob rebases his branch on to main:

```
    git checkout main
    git pull origin main
    git checkout bob-branch
    git rebase main
```

3. Alice needs to update her code with the changes from Bob’s branch:

```
    git checkout alice-branch
    git rebase bob-branch
```

Merge VS Rebase

Merge Vs Reset

- Git merge combines the changes from one branch into another by creating a new commit that has both branch's changes.
- Git rebase integrates the changes from one branch into another by replaying the changes on top of the destination branch, creating a linear history.
- Git merge is simpler and preserves the original history of the source branch, while Git rebase produces a cleaner, linear history but can be more complex and requires more attention to potential conflicts.

Best practices: Use Git merge when you want to preserve the history of the source branch, and use Git rebase when you want to maintain a cleaner, linear project history.

Exercise

Step 1 - Clone a repository

Step 2 - Set `git config pull.rebase true`

Step 3 - Create a new branch

Step 4 - Add your name on name.txt (The names should be in alphabetical order)

Step 5 - Stage and commit your changes

Step 6 - Go to the main branch and pull from the remote repository

Step 7 - Merge your branch with the main branch (You can’t use git merge)

Stash

Stash

Git stash allows you to temporarily save changes that are not ready to be committed, giving you the ability to switch branches or to work on other tasks.

What stashing?

Stash

- keep a clean commit history
- Organize your work
- Avoid losing changes.
    - Saves changes that are not ready to be committed.
    - Saves changes when switching branches or working on other tasks.

Using Git Stash

How to use Git Stash

1. Stash changes using the "`git stash`" command
2. List stashes using the "`git stash list`" command
3. Apply a stash using the "`git stash apply`" command
4. Delete a stash using the "`git stash drop`" command

Git stash apply vs Git stash pop

Git stash apply vs Git stash pop

`Git stash apply` applies the changes from a stash to the working directory, without deleting the stash.

`Git stash pop` restores the most recently stashed changes to the working directory and deletes them from the stash.

Question

Scenario: You are working on a feature branch and you need to switch to a different branch to address an urgent bug fix. However, you have some uncommitted changes in your working directory that you don't want to lose.

Question: How can you make sure you don’t lose your changes

Question: After switching back to your feature branch, you want to apply the changes you stashed earlier. What command would you use to retrieve the changes?

Revert

Git Revert

- Git revert simply creates a new commit that is the opposite of an existing commit.
- Instead of deleting a pre-existing commit, what git revert does is create a new commit that negates the changes caused by the pre-existing commit.
- Command: `git revert <commit-id>`

Reset

Git Reset

- git reset is used when we want to change our repository to a previous commit and discard any changes after that.
- Before we change the state of our repository, we need to figure out which commit we want to go back to.
- To get a list of the commit history we do `git log`.
- After figuring out the commit, in its simplest form, we run git reset <commit-id>.

Conventions

Committing

- Use simple present tense
  - Example : “Update Login Page”
- Add a tag (refer to “conventional commit rules”)
  - Example : fix(mobile): Update Login Page.

Creating a branch:

- The branch name should follow the pattern [your-name].[task-name].

Creating a pull request

- Add a label when creating a pull request

Thank you