What is Git?

Git is a version control system that allows you to keep track of changes made to a project over time.

What is GitHub?

GitHub is popular hosting service for Git repositories. GitHub allows you to store your local Git repositories in the cloud. With GitHub, you can backup your personal files, share your code, and collaborate with others.

Step 1: Install Git for Mac

To install Git on Mac, we first need to install Homebrew.

Step 1.1: Install Homebrew

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

Step 1.2: Install Git

```
brew install git
```

Step 2: Configure Git and Github

Step 2.1: Set up Git

For Git to work properly, we need to configure Git with the following commands. Enter your own information inside the quotes. It’s a good idea to use the same email as your Github account email.

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

To verify that the output matches your name and email address.

```
git config --get user.name
git config --get user.email
```

Step 2.2: Create Github account or Sign in

Go to GitHub.com and create an account. If you already have one, sign in.

Step 2.3: Create SSH Key

Check if you have an SSH key

```
ls ~/.ssh/id_rsa.pub
```

If you receive a message "No such file or directory", you'll need to create a new one. **Create SSH** To create SSH key, run the following command

```
ssh-keygen -C <youremail>
```

- When it asks a location to save the generated key, press Enter.
- Next, it will ask you for a password; enter one if you wish, but it's not required.

Step 2.4: Link Your SSH Key with GitHub

- Log into GitHub and click **Settings**
- Click **SSH and GPG keys** and click **New SSH Key**
- Run this following command in your terminal, copy the output into the key field and click click **Add SSH key**

```
cat ~/.ssh/id_rsa.pub
```

- Follow this Github article to test your key

Git Project

A Git project has three parts:

- A Working Directory: where you'll be doing all the work: creating, editing, deleting and organizing files
- A Staging Area: where you'll list changes you make to the working directory
- A Repository: where Git permanently stores those changes as different versions of the project

Git commands

- `git init` initialize a new Git repository
- `git status` check the status of changes
- `git add filename` adds files from the working directory to the staging area
- `git add filename_1 filename_2` add multiple files to the staging area with a single command:
- `git diff filename` shows the difference between the working directory and the staging area
- `git commit` permanently stores changes from the staging area inside the repository. the option `-m` followed by a message.

```
git commit -m "First commit"
```

Standard commit messages:

- Must be in quotation marks
- Written in the present tense
- Should be brief (50 characters or less) when using -m

`git log` shows a list of all previous commits. The message includes

- A 40-character code, called a SHA, that uniquely identifies the commit. This appears in orange text.

- The commit author
- The date and time of the commit
- The commit message

Backtrack Git

In Git, HEAD commit is the commit you're currently on.

- `git checkout HEAD filename` restores the file in your working directory to look exactly as it did when you last made a commit. We can also use `git checkout -- filename`
- `git reset HEAD filename` unstages file changes in the staging area.
- `git reset commit_SHA` resets to a previous commit in your commit history. This command works by using the first 7 characters of the SHA of a previous commit.

Git branch

Git allows users to create branches to experiment with versions of a project. To list all a Git project's branches, use this command `git branch`

Create a new branch

```
git branch new_branch
```

Switch to the new branch

```
git checkout branch_name
```

Commit on a new branch

All the commands you do on master branch, you can also do on this branch.

- To add files to the staging area `git add filename`
- To commit `git commit -m "Commit message"`

Merge

Firstly, we need to switch to master branch `git checkout master` Then use this command to merge the new branch to a master branch `git merge branch_name`

Delete branch

After the branch has been integrated into master, you can delete it.

```
git branch -d branch_name
```

If the feature branch is never merged into master branch, we use the -D option

```
git branch -D branchname
```

Git workflow on your own

- Initiate Git or Clone GitHub to local machine

```
git clone https://github.com/USER_NAME/REPOSITORY_NAME.git
```

- Create a branch to work on a new project feature and push it to the server:

```
git branch <name of the branch>
git push -u origin <name of the branch>
```

For example, if your branch is “develop”, you can enter the following command.

```
git branch develop
git push -u origin develop
```

- Switch to a the branch

```
git checkout <name of the branch>
```

- Make changes on the working files
- Commit changes:

```
git status
git add --a
git commit -m "Add your comment here"
```

- After commit push branch to remote:

```
git checkout master
git merge <name of the branch>
git push -u- origin master
```

Git Teamwork

A remote (Git repository) can live on the web, on a shared network or even in a separate folder on your local computer.

git clone

To create a local copy of a remote:

```
git clone remote_location clone_name
```

- `remote_location` tells Git where to go to find the remote.

- `clone_name` is the name you give to the directory in which Git will clone the repository

git remote

See a list of a Git project's remotes with the command: `git remote -v`

git fetch

To fetch work from the remote into the local copy: `git fetch`

git merge

To merge origin/master into your local branch: `git merge origin/master`

git push

To push a local branch to the origin remote: `git push origin <branch_name>`

Git workflow teamwork

1. Fetch and merge changes from the remote
2. Create a branch to work on a new project feature
3. Develop the feature on your branch and commit your work
4. Fetch and merge from the remote again (in case new commits were made while you were working)
5. Push your branch up to the remote for review

Resources:

- Set up Git
- Git introduction
- Codeacademy