Part 3: Technical Explanation

What is Git?

Git is the most widely used version control system. Version control is software that helps a team manage changes to its source code. Git was created originally in 2005 to manage the Linux kernel. Nowadays it is used across the world for an unfathomably huge number of software projects.

Git tracks your changes to the source code via **commits**, and syncs with a remote source, like GitHub or Bitbucket, via **push** (to send your commits to the remote) or **pull** (to retrieve commits from the remote) commands.

What is GitHub?

GitHub is an online space to work collaboratively on projects and keep your code safe.

Most software requires teams of developers to work simultaneously in a safe way. GitHub is a tool that allows us to do that. It is a powerful tool on which all developers rely heavily.

Virtually every software company stores its code in GitHub or a similar service.

How do I use Git and GitHub?

This handy video can walk you through the below steps:

1. Create a new repository
2. Copy the link
3. In terminal, `cd` into the directory where you want your repo to live
4. Clone the repo using `git clone <repo name>`
5. `cd` into the repo
6. Add a file
7. Stage the changes in your current working directory using `git add .`
8. Commit the changes using `git commit -m "some message"`
9. Push the changes using `git push origin master`

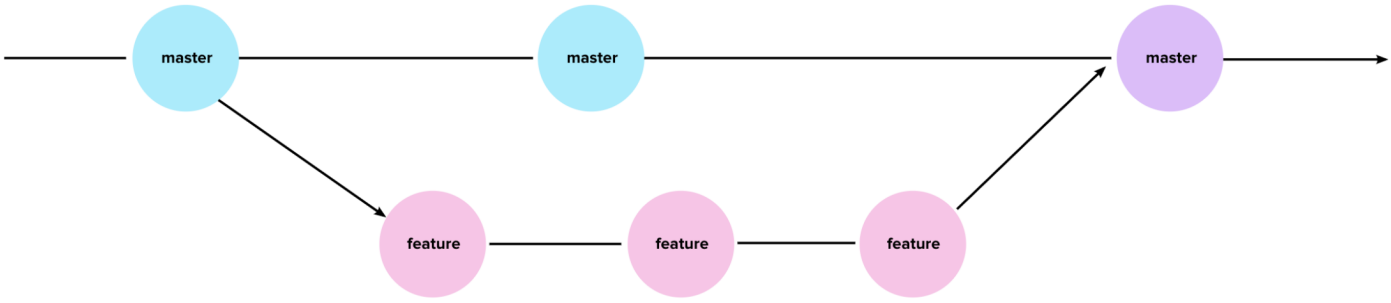
This will push your changes directly into the `master` branch. The `master` is the default branch of your repository, and as its name suggests, is the master source for your project. As you might have guessed, we do not want to push directly to our master branch, and in fact, when working on a team, you likely will not have access to do so.

To get around this, we use a workflow called **Feature Branching**. The main idea behind feature branching is that all feature development should take place in a dedicated branch rather than pushing directly to the master branch.

Here's how you can work in this kind of environment:

1. Create a new branch.
2. Then, all changes to our repository (additions, modifications, or deletion) are made inside this branch.
3. We then commit our changes, and push them to the remote source (GitHub).
4. Once the branch has been pushed to the remote, we open a Pull Request.
5. Once the Pull Request has been made, there will very often be automated tests that run to ensure that your modifications to not break the working application.
6. Pull Requests also initiate a discussion about the changes you've committed, and generally have to be approved by someone else on the team. That person (or entire team) will review your changes to make sure that everything is in order.
7. Once any concerns have been addressed and the Pull Request has been approved, the modified code is merged into the master branch.

A great benefit of this process is that, once merged, Pull Requests act as a record of changes to the codebase, just like a save file!



To create a new branch:

1. `cd` into the repo directory
2. Create and switch to a new branch using `git checkout -b new-branch-name`
3. Make the changes to your repository that you desire
4. Stage the changes using `git add -A`
5. Commit the changes using `git commit -m "some message"`
6. Push the changes using `git push origin new-branch-name`
7. Navigate to your GitHub repository in your browser

8. On the `Pull requests` tab, create a new pull request

To switch between existing branches, use `git checkout branch-name`. Note that you do not need the `-b` flag in your `git checkout` command this time because we are not creating a new branch.

To merge your changes into master, and update your local repository with those changes:

1. On GitHub, merge the Pull Request you created.
2. On your computer, switch back to your master branch using `git checkout master`
3. Pull the updates from the remote server with `git pull`