

Git is a distributed version control system used to track changes to a code base. When git was invented in 2005 by Linus Torvalds it was unique from other version control systems because it was distributed, all developers can have access to the full history of the code base. It enables multiple people to work on a project at the same time. There are many features in git that make this a possibility. One of those features are "pulls", used to incorporate upstream changes with a local repository. Repositories are a collection of files that make up a project.

When code is "pulled" from a repository, it retrieves a copy of the remote repository, typically this is the "origin." This is done to sync the changes made on a local repository with changes that others have made. A commit is an image of the repository at a point in time. "Push" is a command used to upload changes from a local repository to a remote repository. Changes and commits to a repository are executed through the standard Git commit process within a local repository. The standard commit process consists of three steps: edit, stage and commit. This staging allows developers to make changes in smaller commits and continue to work in their local repository.

Another feature of git is merge; this feature is used to combine two branches. When the merge feature is used to combine two branches this can cause a merge conflict to occur when local changes are in direct conflict with upstream commits. Such as, if one person is working and then pushes their work to the main branch then another who is also working tries to push after the first person. The second person will not be able to push to the main branch because the second person's history is now in conflict with the central repository. Git then requires this conflict to be manually rectified otherwise the second person would be overwriting official commits. This can be done by the second person pulling the first person's updates and integrating their work into it.

Git workflows are practices and conventions that a team will follow when git is used on a project. There are many different possible workflows and this makes it a challenge to choose a workflow when git is being utilized in a workspace. Some of the challenges that need to be taken into consideration are, if the workflow will scale well with the size of the team, if mistakes are made can they easily be rectified and if the workflow creates any unnecessary difficulties for a team. Some of the workflows are centralized, predetermined or decentralized. It depends on the needs of the team and project which workflow will be used.

A Centralized workflow is most useful for small teams. This workflow has one main branch and all of the work is done on the main branch. The centralized workflow is the basis for all other popular git workflows in the sense that there is a centralized repo that developers will push their local changes and retrieve other's changes. A Feature branching Workflow has different isolated branches, so features can be worked on before merging with the main branch. A pull request is made when there is a request to merge changes to code that have been completed on one branch to another typically this would be from a feature branch to the main branch.

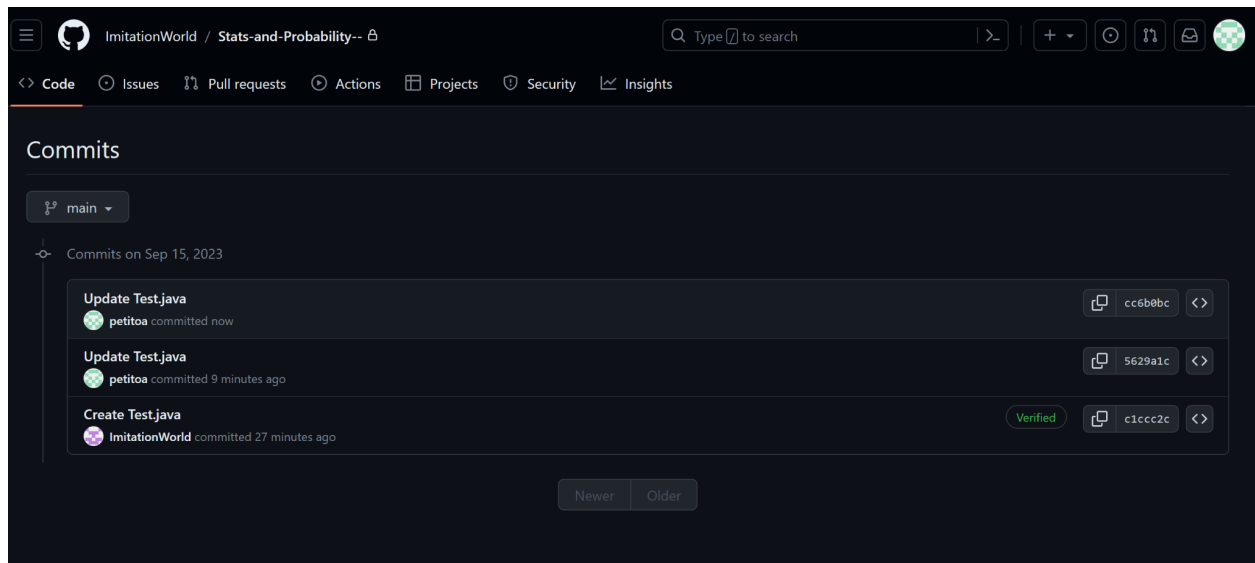
A Gitflow Workflow has predetermined branches and a model that revolves around the release of a project. This workflow outlines the project and has a strict branching that will never change and will be used to implement features and concepts. In a Forking Workflow, each developer pushes their completed features to their individual public repositories, rather than a centrally shared one. In the Forking Workflow each developer will have two branches one is a private local branch and the other is the public server side branch. The advantage of a forking workflow is that changes can be implemented without having to have all developers push to a central repository. Git has many features like "pulls," "commits," and "merges," that enable multiple developers and many different workflows.

Bibliography:

Atlassian. “Git Workflow: Atlassian Git Tutorial.” *Atlassian*, www.atlassian.com/git/tutorials/comparing-workflows. Accessed 16 Sept. 2023.

Atlassian. “Git Workflow: Atlassian Git Tutorial.” *Atlassian*, www.atlassian.com/git/tutorials/comparing-workflows. Accessed 16 Sept. 2023.

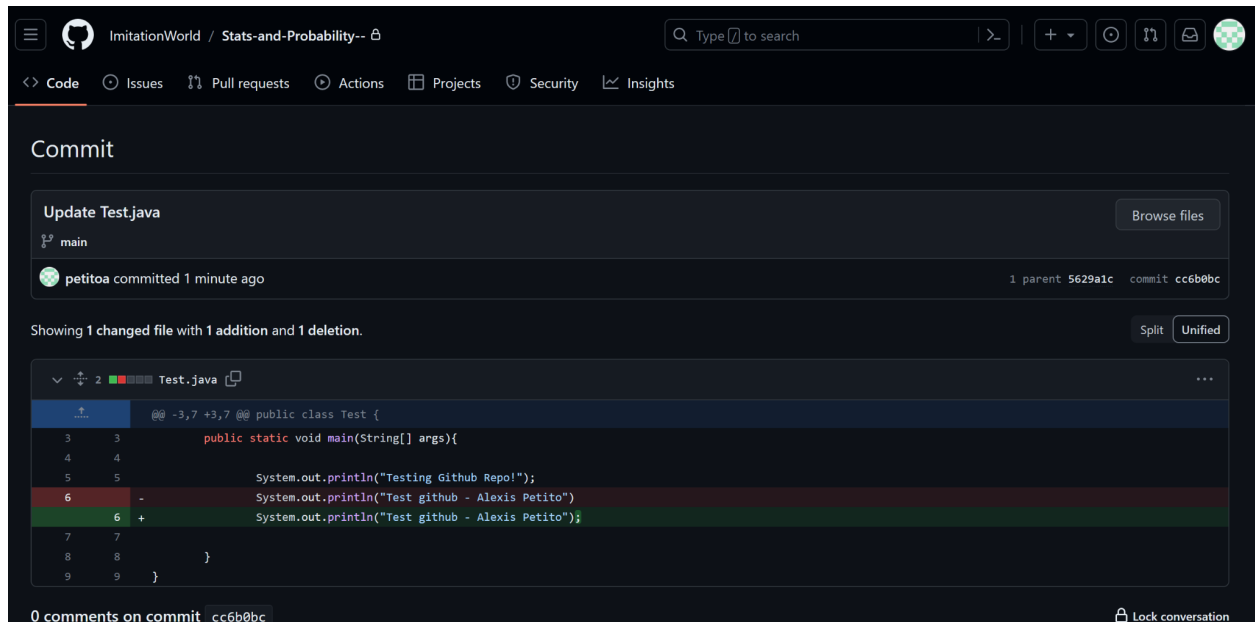
GitHub assignment make commit to another repository:



This screenshot shows the commit history of a GitHub repository named 'Stats-and-Probability--' by user 'ImitationWorld'. The interface is in dark mode. At the top, there's a search bar and navigation tabs for Code, Issues, Pull requests, Actions, Projects, Security, and Insights. The 'Commits' tab is active, showing a list of commits on the 'main' branch for the date 'Sep 15, 2023'. The commits are:

- Update Test.java** by petitoa, committed now (hash: cc6b0bc).
- Update Test.java** by petitoa, committed 9 minutes ago (hash: 5629a1c).
- Create Test.java** by ImitationWorld, committed 27 minutes ago (hash: c1ccc2c, marked as Verified).

Navigation buttons for 'Newer' and 'Older' commits are at the bottom.



This screenshot shows the details of a specific commit in the same repository. The commit is titled 'Update Test.java' by 'petitoa', committed 1 minute ago, with hash 'cc6b0bc'. It has one parent commit, '5629a1c'. Below the commit info, it states 'Showing 1 changed file with 1 addition and 1 deletion.' and provides a 'Split' / 'Unified' view toggle. The file 'Test.java' is shown with a diff view. The diff highlights changes between the parent and the current commit:

```
@@ -3,7 +3,7 @@ public class Test {
3      public static void main(String[] args){
4
5      System.out.println("Testing Github Repo!");
6      - System.out.println("Test github - Alexis Petito");
6      + System.out.println("Test github - Alexis Petito");
7
8      }
9  }
```

At the bottom, it shows '0 comments on commit cc6b0bc' and a 'Lock conversation' button.