Evans Mutwiri

5 April 2022

Title: Version Control

[GitHub Link](https://github.com/EvansMutwiri/my--internship.git)

Version Control Report

**Branching and Merging**

Branching means diverging from the main line of development and continuing to do work without changing anything in the main line(branch).

It’s typical to create a new branch and want to switch to that new branch at the same time — this can be done in one operation with

Deleting a may be used for a reason such as when a feature branch has already been merged into the development branch and it is no longer useful.

Git stores data as a series of snapshots. Git stores a commit object that contains the author name and email address, the message and pointers to that commit or commits that directly came before this commit(its parent or parents) - for context the initial commit has zero parents, one parent for a normal commit and multiple parents for a commit that results from a merge of two or more branches. This creates a tree object in the repository.

When you create a commit this commit creates a pointer to the commit that came before it.

The default branch is **main**(formerly master). When you create a branch a new pointer to the same commit you’re currently on is created. To know the exact branch you're on, git keeps a pointer called HEAD.

$ git log --oneline

This shows where the pointers are pointing. By default the HEAD points to the main branch otherwise points to the current branch.

$ git checkout testing

$ git switch testing

$ git switch ch -

These git commands can be easily executed on **GitHub** in a gui.

In **GitHub** switching between branches is as much as a click. To view all branches navigate to the main page of the repository and click NUMBER branches. This lists all the active branches to navigate to any just click on its name.

Other options available in GitHub are **renaming a branch**, **changing the default branch** and **delete/restore branches**.

**Pull request** is an attempt to add your own changes to the repository of another owner. To make such a request, you need to take someone’s project, create a separate branch, and then offer your changes to merge it with the rest and be part of the main project. Once a pull request is opened, you can discuss and review the potential changes with other collaborators before changes are merged with the rest of the code.

**Merging**

When a change is pushed to a feature branch a compare and pull request button will appear on GitHub. When you click it, it directs to another page where you choose the base branch and the branch to compare with. Add a merge commit message and description then click create pull request. Once a pull request is submitted, GitHub compares the two branches for differences and where no conflicts are present, then the merge can be completed easily.

**Comparison**

The Compare View brings all information needed to determine what changed over a series of commits onto a single page: a condensed commit list in chronological order, followed by a rollup diff of all changes between the two points, followed by any relevant commit comments. All in the same place and with a single well-defined URL.

In the compare view there are two menus to pick the base and compare.

It is possible to compare across branches, tags, commits, forks, and dates. The compare view eliminates the need to use some git commands such as git log, git diff…

It is a useful feature to use when reviewing branches before merging.

**Stashing**

Git stash - Stash the changes in a dirty working directory away, can be used in cases where we need to switch in between branches and at the same time not wanting to lose edits in the current branch.

Right clicking the changed files folder, then clicking stash all changes basically saves your modifications away and reverts the working directory to match the HEAD commit, hence providing a clean working directory for other tasks.

The modifications stashed away can be listed, inspected with and restored. This will help resume working where you had last left before switching to other branches.

Stashing is useful for scenarios such as conflicts with upstream changes where git pull fails to overwrite changes, interrupted workflows i.e. when something has to be done in the middle of something else going on, testing partial commits or saving changes for future use.

**Rebasing**

The purpose of rebase and merge is to integrate changes from one branch to another, but they do it differently. When rebase is used the result is a linear history with content from both branches. The golden rule of rebase is never to use it for branches available to other developers because it will lead to out of sync copies of repositories.

**Github actions**

GitHub Actions are event-driven, which implies you can run a series of commands in response to a specific event. For example, you can run a command that executes a software testing script every time someone creates a pull request for a repository.

GitHub Actions provides the flexibility to automate all of your software workflows, and it now includes world-class CI/CD. You can build, test, and deploy your code directly from GitHub. Allow code reviews, branch management, and issue triaging work the way you need.

**Creating a workflow to trigger github actions**

GitHub offers pre-configured workflow templates where you can modify to create your own continuous integration workflow. GitHub examines your code and suggests CI templates that may be useful for your repository. If one's repository contains Node.js code, for example, you'll see recommendations for Node.js projects. Workflow templates can be used as a starting point for creating your own custom workflow or as-is.

The [actions/starter-workflows](https://github.com/actions/starter-workflows) repository contains a complete list of workflow templates.

To create a workflow will require two steps:

1. Create a new file called github-actions.yml in the.github/workflows directory of your GitHub repository(create the folder too).
2. In the github-actions.yml file, write the YAML content. YAML is a data serialization language that is often used for writing configuration files. The content of this file will be the specifics of the workflow including: the trigger event eg: on [push] , the jobs and the virtual machine it will run on.

**Jobs** are a set of steps in a workflow that execute on the same runner. They can be shell scripts or an action that will run. By default they run in parallel unless a job takes a dependency on another job.

1. Choose Create a new branch for this commit and begin a pull request. Then, to make a pull request, go to File > Propose New File. The push event will be triggered when you push commits to a branch.

**CODEOWNERS** file is used to define individuals or teams responsible for code in a repository. They review pull requests that modify code that they own.