Gitflow Workflow Guide

Author: Jason Liu - ASLSTEAMHUB Group 18
Date: Feb 20th, 2025

Introduction

As a refresher, Gitflow workflow is a branching model for Git that provides a structured workflow for managing code development, releases, and maintenance. It allows us to have a unique branch for each feature that we are able to work on individually and to merge together in the dev branch for integration and testing before pushing to main.

Quick Guide

If you already know and are comfortable with Gitflow workflow, here are the commands as a quick reference. If you would like a more in-depth step-by-step guide, read from the Steps section.

```
git switch develop  # Ensure you're on the develop branch
git pull origin develop  # Get the latest changes
git switch -c feature/new-feature  # Create a new feature branch
git push -u origin feature/new-feature  # Push it to the remote repository
```

```
git add .  # Stage changes
git commit -m "Add new feature X"  # Commit changes
git push origin feature/new-feature  # Push changes to remote
```

```
git switch develop
git pull origin develop
git switch -c bugfix/fix-issue-123  # Create bugfix branch
git push -u origin bugfix/fix-issue-123
```

```
git add .
git commit -m "Fix issue #123"
git push origin bugfix/fix-issue-123
```

```
git switch develop
git pull origin develop
git merge feature/new-feature  # Merge the feature into develop
git push origin develop
```

```
git add .
git commit -m "Resolve merge conflict"
git push origin develop
```

```
git branch -d feature/new-feature  # Delete locally
git push origin --delete feature/new-feature  # Delete remotely
```

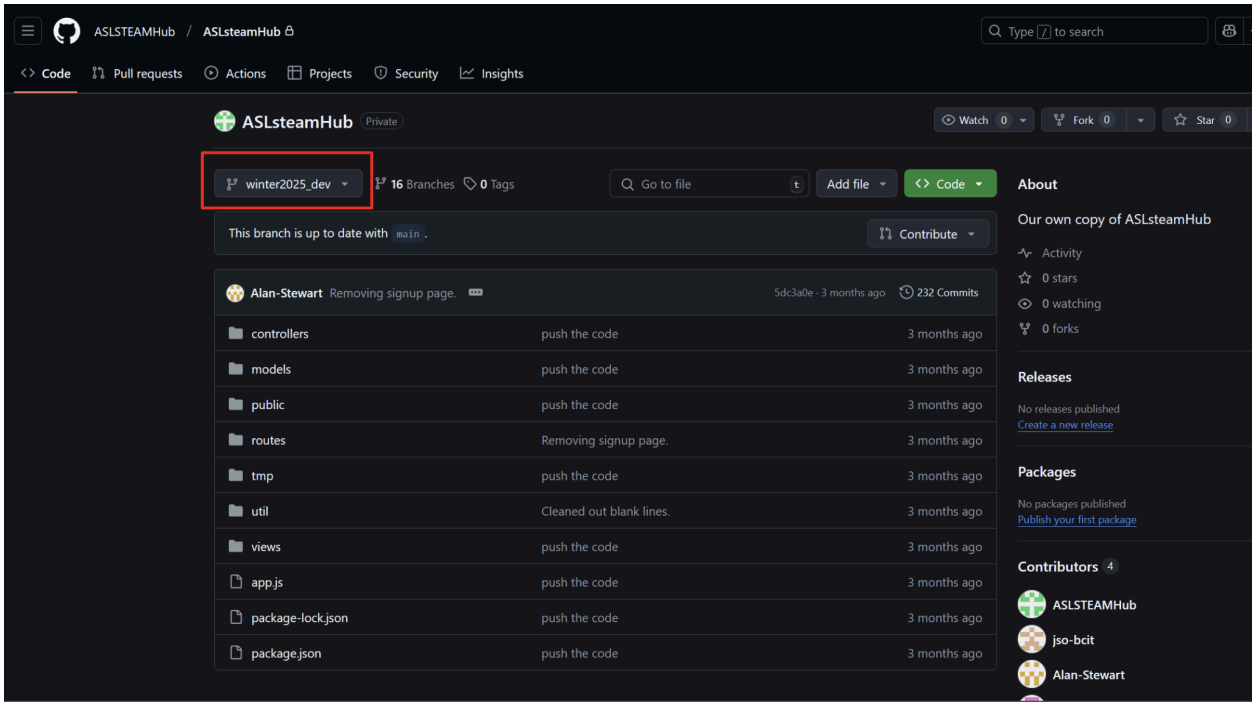
```
git switch develop
git pull origin develop
git switch -c release/v1.0  # Create a release branch
git push -u origin release/v1.0
```

```
git switch main
git merge release/v1.0  # Merge release into main
git tag v1.0  # Tag the release
git push origin main --tags  # Push the tag

git switch develop
git merge release/v1.0  # Merge release back into develop
git push origin develop
```

Steps

The dev branch that we will be working off of has already been created in the ASLSteamHub repository.



1. To work off of the dev branch, in the terminal, enter `git switch <your_devbranch_name_here>`. As an example for this document, we will be using `winter2025_dev`. Alternatively, you can use `git checkout winter2025_dev` as well.

```
PS C:\Users\Jason\Documents\BCIT CST\Term 4\COMP 3800 - Practicum 1\ASLsteamHub> git switch winter2025_dev
M       package-lock.json
M       package.json
Already on 'winter2025_dev'
Your branch is up to date with 'origin/winter2025_dev'.
PS C:\Users\Jason\Documents\BCIT CST\Term 4\COMP 3800 - Practicum 1\ASLsteamHub>
```

Note: `switch` is the newer method for changing branches as `checkout` is overloaded in terms of functionality. Either one works and is up to your preference.
To check which branch you are on, enter `git branch` or alternatively with `git status` for more additional info. Notice how the highlighted branch name changes.

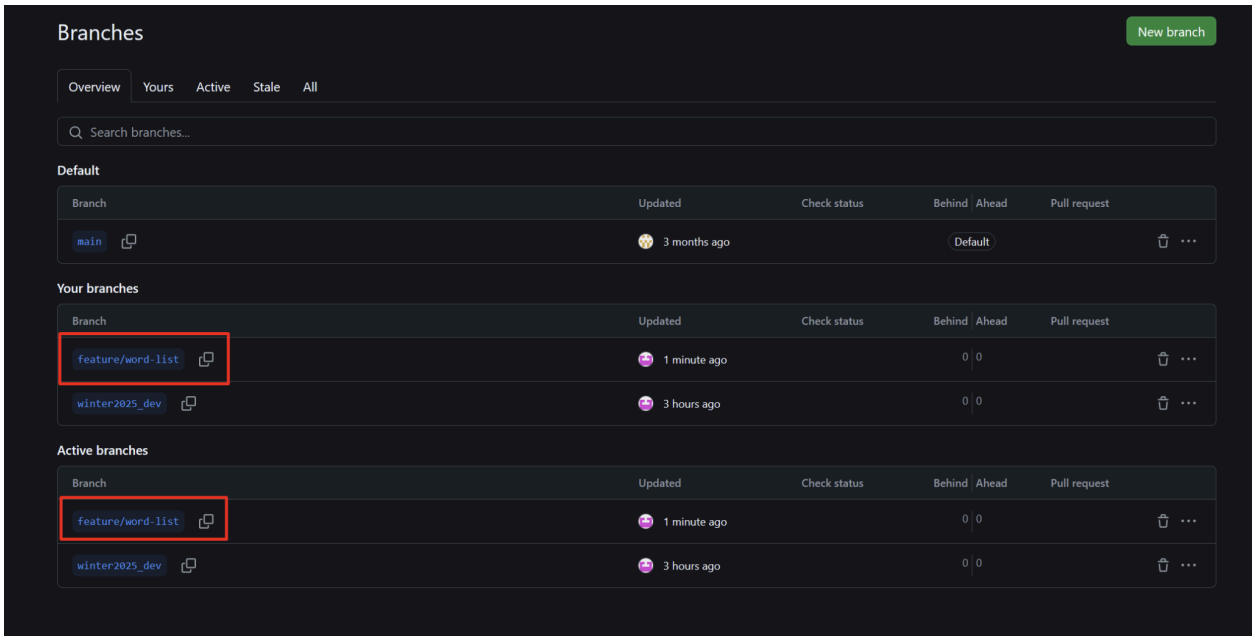
```
PS C:\Users\Jason\Documents\BCIT CST\Term 4\COMP 3800 - Practicum 1\ASLsteamHub> git branch
* main
  winter2025_dev
PS C:\Users\Jason\Documents\BCIT CST\Term 4\COMP 3800 - Practicum 1\ASLsteamHub> git switch winter2025_dev
M       package-lock.json
M       package.json
Switched to branch 'winter2025_dev'
Your branch is up to date with 'origin/winter2025_dev'.
PS C:\Users\Jason\Documents\BCIT CST\Term 4\COMP 3800 - Practicum 1\ASLsteamHub> git branch
  main
* winter2025_dev
PS C:\Users\Jason\Documents\BCIT CST\Term 4\COMP 3800 - Practicum 1\ASLsteamHub>
```

2. Ensure that you are on the `winter2025_dev branch` before creating a new branch. On our dev branch, make sure to pull to get the latest changes with `git pull origin winter2025_dev`.

3. Create a new branch with `git switch -c new-branch` or `git checkout -b new-branch`. We will follow the naming convention of `feature/new-feature` and `bugfix/fix-bug-description`. For example, creating a feature branch for the word lists would look like `git switch -c feature/word-list`.

4. Once the new branch has been created, enter `git push -u origin feature/new-feature` to push the newly created branch to the remote repository from the local repository.

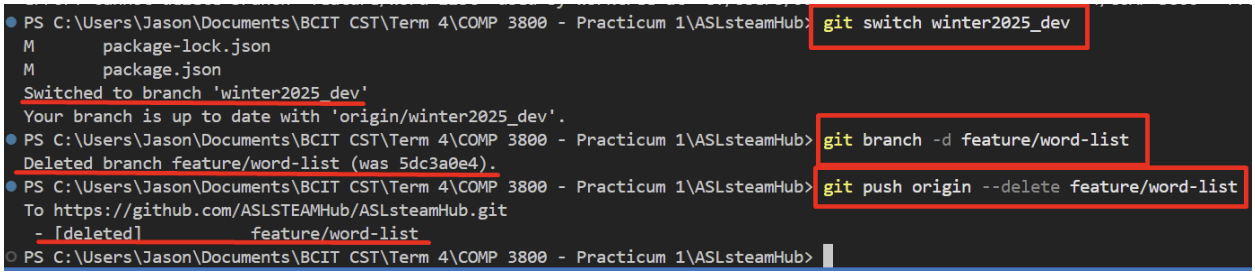
```
PS C:\Users\Jason\Documents\BCIT CST\Term 4\COMP 3800 - Practicum 1\ASLsteamHub> git switch -c feature/word-list
Switched to a new branch 'feature/word-list'
PS C:\Users\Jason\Documents\BCIT CST\Term 4\COMP 3800 - Practicum 1\ASLsteamHub> git push -u origin feature/word-list
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature/word-list' on GitHub by visiting:
remote:      https://github.com/ASLSTEAMHub/ASLsteamHub/pull/new/feature/word-list
remote:
To https://github.com/ASLSTEAMHub/ASLsteamHub.git
 * [new branch]        feature/word-list -> feature/word-list
branch 'feature/word-list' set up to track 'origin/feature/word-list'.
```

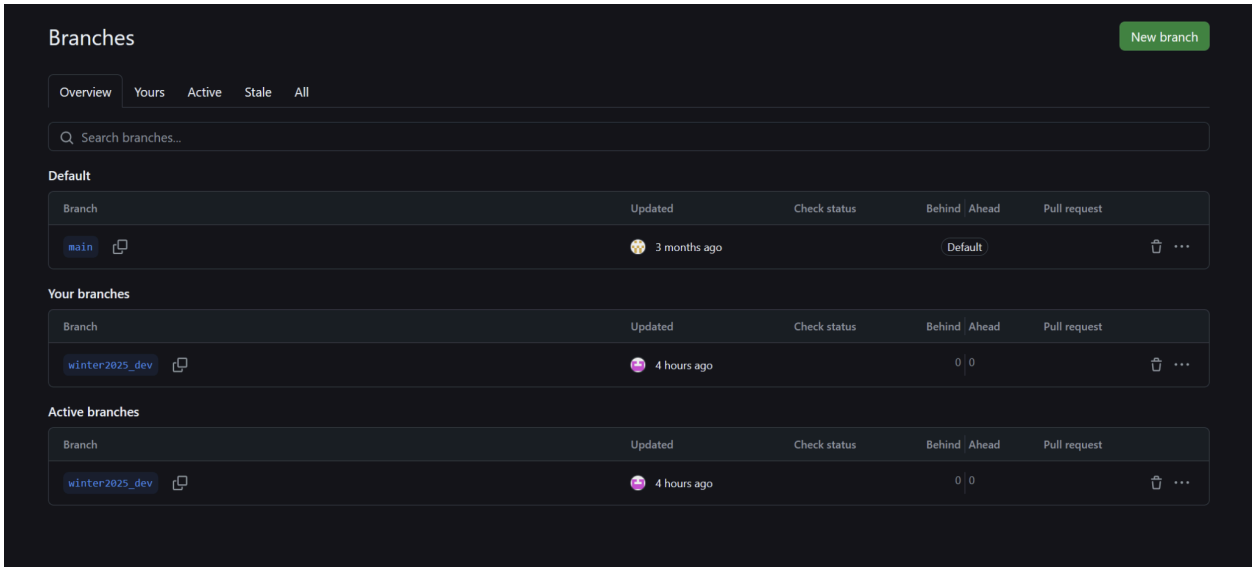


Note how the newly created branch is now visible in the remote github repository.

5. Add and commit work with the usual commands of `git add` and `git commit -m`. To push, make sure to use the same branch. For example, on the `feature/word-list branch`, enter `git push origin feature/word-list`.

6. If for any reason you want to delete a branch, move to a different branch first (`winter2025_dev`), then enter `git branch -d feature/new-feature` to delete locally, then `git push origin --delete feature/new-feature` to delete remotely. For example, to delete the `feature/word-list` branch locally, enter `git branch -d feature/word-list`. Then delete it remotely by entering `git push origin --delete feature/word-list`.





Note how the `feature/word-list` branch is now gone.

7. Once a feature or bugfix is complete, merge it into the dev branch by switching to `winter2025_dev`, pulling to make sure everything is up to date, then enter `git merge feature/new-feature`, then push with `git push origin develop`. Resolve any conflicts, add, commit, and push again.

8. Once integration and testing is complete in the dev branch, we can then merge the production-ready code into `main`. Optionally, a release branch can be created as well for final checks, minor fixes, documentation. Switch to the `main` branch and merge the dev branch/release branch with main. Add and push production tags, then push the changes to the remote repository.