Gitflow Tutorial

1 What is Gitflow?

Gitflow is simply an organized model for managing git branches for project releases and features.

The overall flow of Gitflow is:

1. A develop branch is created from master
2. A release branch is created from develop
3. Feature branches are created from develop
4. When a feature is complete it is merged into the develop branch
5. When the release branch is done it is merged into develop and master
6. If an issue in master is detected a hotfix branch is created from master
7. Once the hotfix is complete it is merged to both develop and master

To learn more about Gitflow, you can visit this Link:

`https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow`

2 Why does Gitflow exist?

- Ensures that only working code exists on master
- Provides avenue for multiple code reviews
- Prevents disappearing code (code overwritten by merge conflicts in the master branch)

3 What you will be doing today

For the sake of this tutorial, you will only be using the following branches:

- master
- release
- feature
- bugfix
- hotfix

4 Getting Started

4.1 Find A Partner

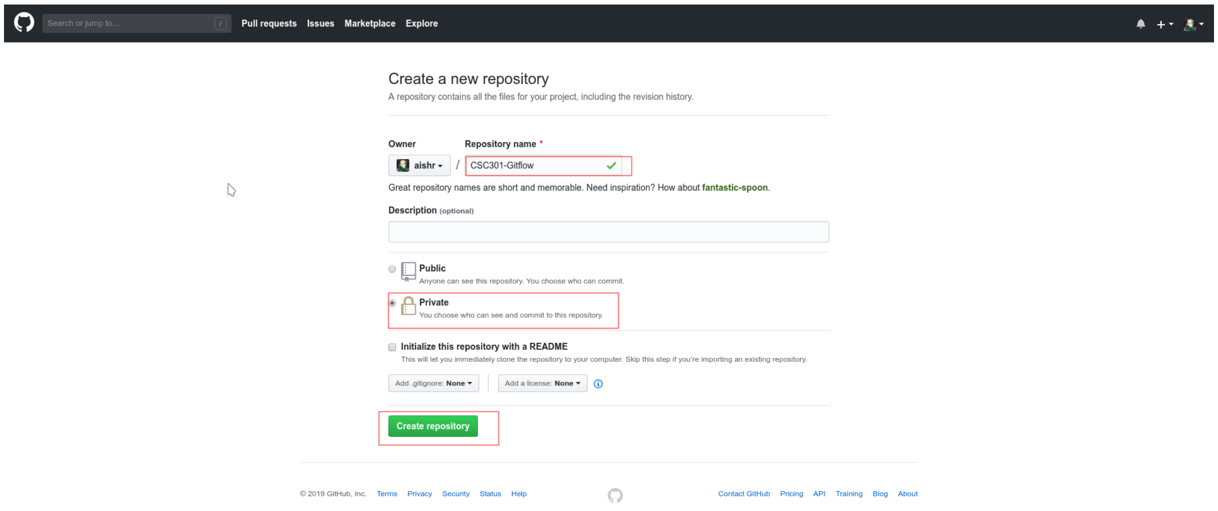
This lab will be done in pairs. Find somebody to work with for this lab. You should not be working on the same computer. Feel free to use personal devices instead of lab computers.

4.2 Creating a Git Repo

Between you and your partner, only one of you needs to create a git repo. Please use the same github account that you are currently using for the GitHub Classroom for this course.

On your home page, click on the + dropdown and select `New repository`

Once redirected, fill in the page as follows and click on `Create repository`



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: aishr / Repository name *: CSC301-Gitflow

Great repository names are short and memorable. Need inspiration? How about fantastic-spoon.

Description (optional)

Public — Anyone can see this repository. You choose who can commit.

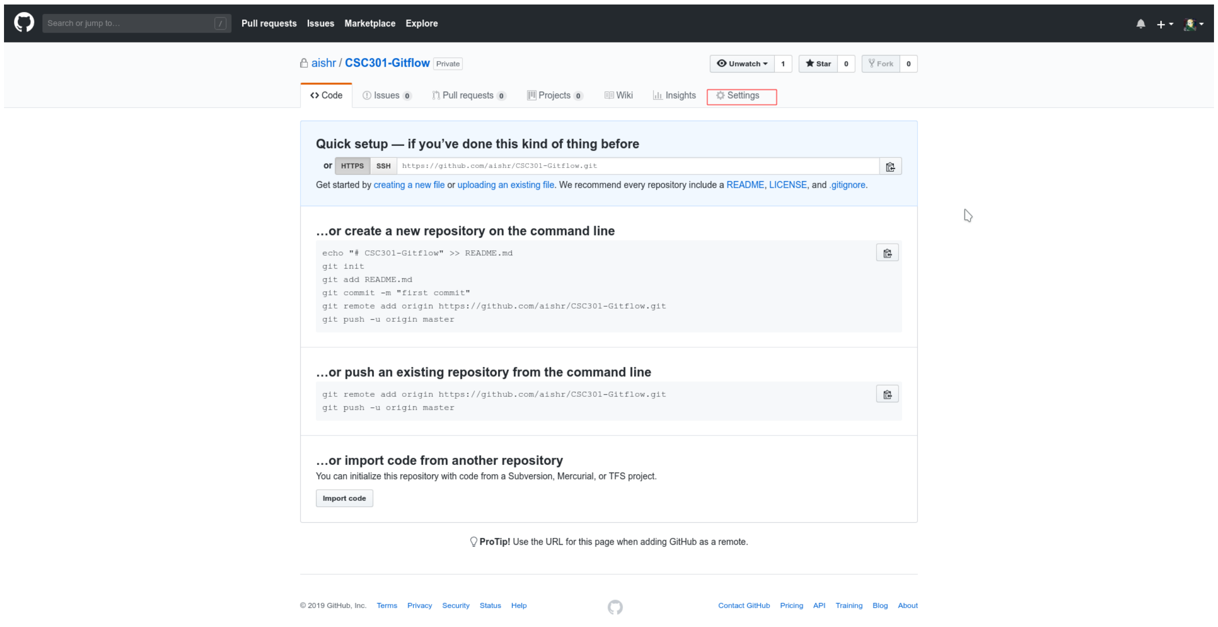
[✓ Private] — You choose who can see and commit to this repository.

Initialize this repository with a README — This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

4.3 Adding A Collaborator

Once your repo is created, you will be redirected to the following page:



aishr / CSC301-Gitflow Private

Quick setup — if you've done this kind of thing before

or HTTPS SSH https://github.com/aishr/CSC301-Gitflow.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# CSC301-Gitflow" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/aishr/CSC301-Gitflow.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/aishr/CSC301-Gitflow.git
git push -u origin master
```

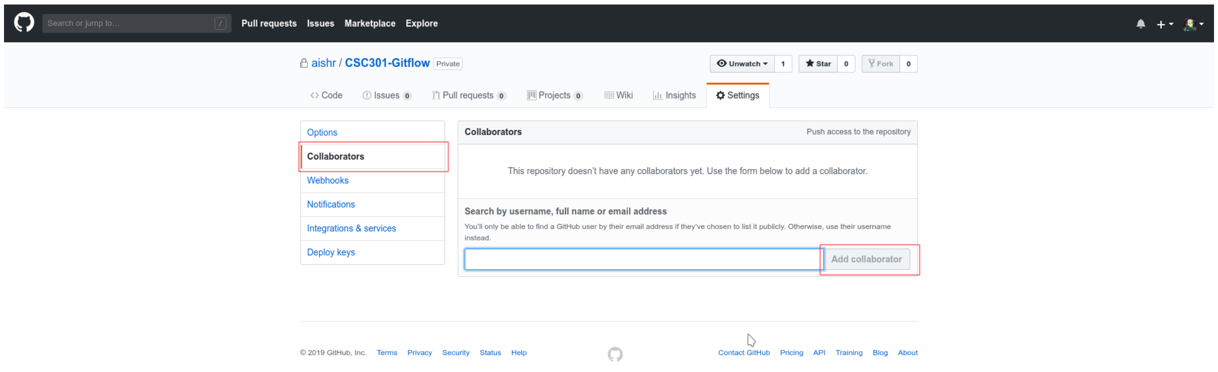
...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

ProTip! Use the URL for this page when adding GitHub as a remote.

© 2019 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

- On the top bar, click on `Settings`
- Click on Collaborators
- Enter in your partner's GitHub username
- Select the correct user from the dropdown
- Click `Add collaborator`



aishr / CSC301-Gitflow Private

Collaborators — Push access to the repository

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

© 2019 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

4.4 Adding Remote and Pushing

Inside of the provided folder (wherever you've put your starter code), run the following commands:

1. `git remote add origin https://github.com/user/repo`
2. `git push -u origin --all`.

It would be a good idea to verify that the changes have indeed been pushed to the remote.

Provided in the repository are an empty `master` branch and a `release/1.0` branch with some incomplete code.

5 Making Changes

5.1 Feature & Bigfix

Inside of the code you'll find two two tasks marked with `TODO`. Each person should complete one task on the correct branch specified in the comments. The goal of the lab in to get the completed code into the `master` branch. Once your changes are made you should both merge your branches into the release branch `release/1.0` and resolve any conflicts. This is the process you would be going through in a team environment to get everybody's changes into the same release. A release is a group of changes that will be sent to the production environment in a batch. At this point your release is complete and you should merge your release branch into master, delete your release branch and push all of this to your remote.

5.1.1 Summary

1. Each partner should complete a todo in your starter code on the correct branch
2. Merge both branches into your release branch
3. Resolve any merge conflicts
4. Once your release is complete with both changes and there are no merge conflicts, merge your release branch into master
5. Delete your release branch
6. Push all changed to your remote.

5.2 Hotfix

Finally you'll be making what's called a hotfix. These are emergency changes that need to be put into production as soon as possible. These branches are made off of master and merged directly to master. You'll make a branch off of `master` called `hotfix/PageContentFix` and edit the body of the web page and add an `<h1>` tag that simply says "HotFix". Merge this back into master and remove the branch then push all of your changes.

5.2.1 Summary

1. Branch off of master with the correct branch name
2. Make the correct changes to your hotfix branch
3. Merge your branch back into master
4. Remove the hotfix branch
5. Push all changes to remote