CSE-322

Software Engineering Laboratory

Version Controlling with Git & GitHub

Md. Saidul Hoque Anik
anik@cse.uiu.ac.bd

Objective

1. Keep track of changes in a project. (Git)
2. Rollback if a problem occurs. (Git)
3. Push the changes to server so that everyone can see it. (Github)

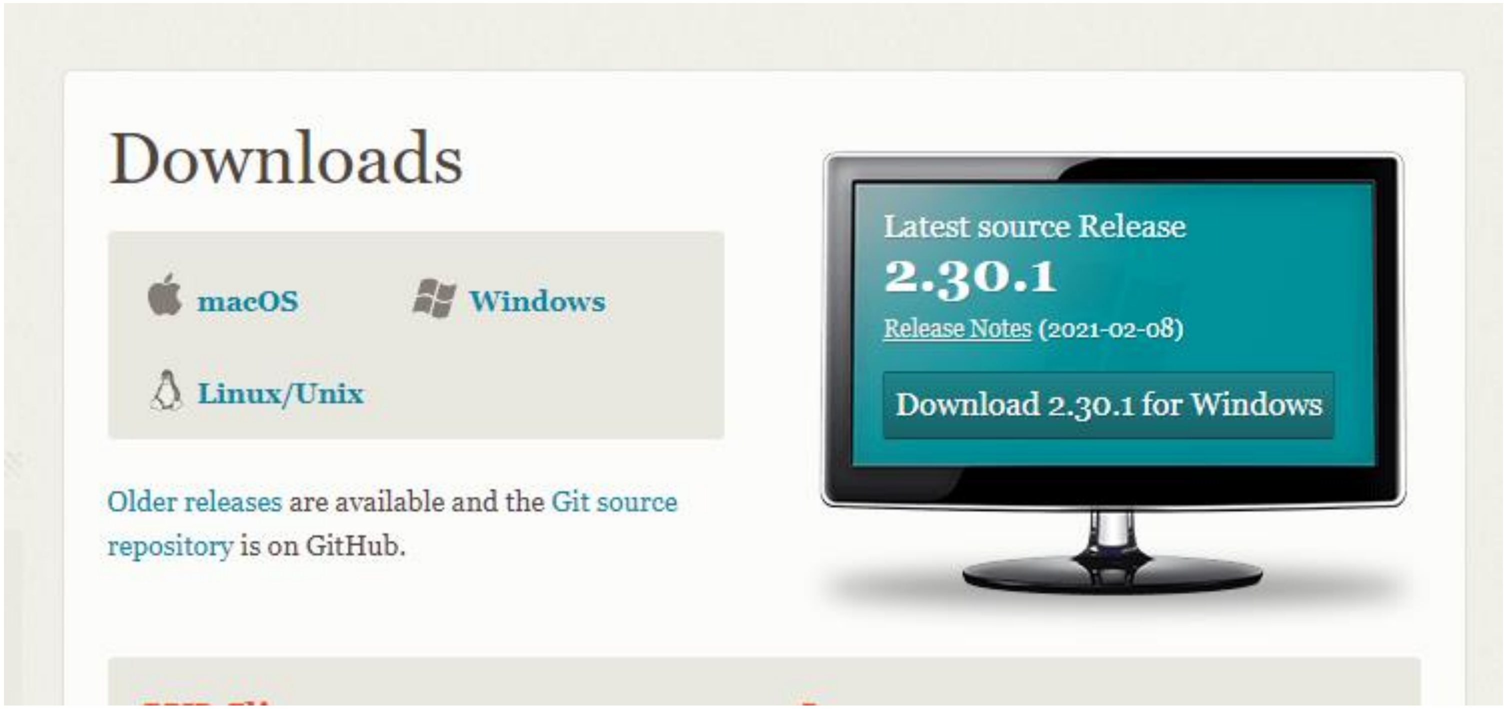
Git

- Git is a program that keeps track of the changes in your project.
- The changes are saved in your computer (local)

Git

Download and Install Git from here (don't download GUI client):

https://git-scm.com/downloads



Downloads

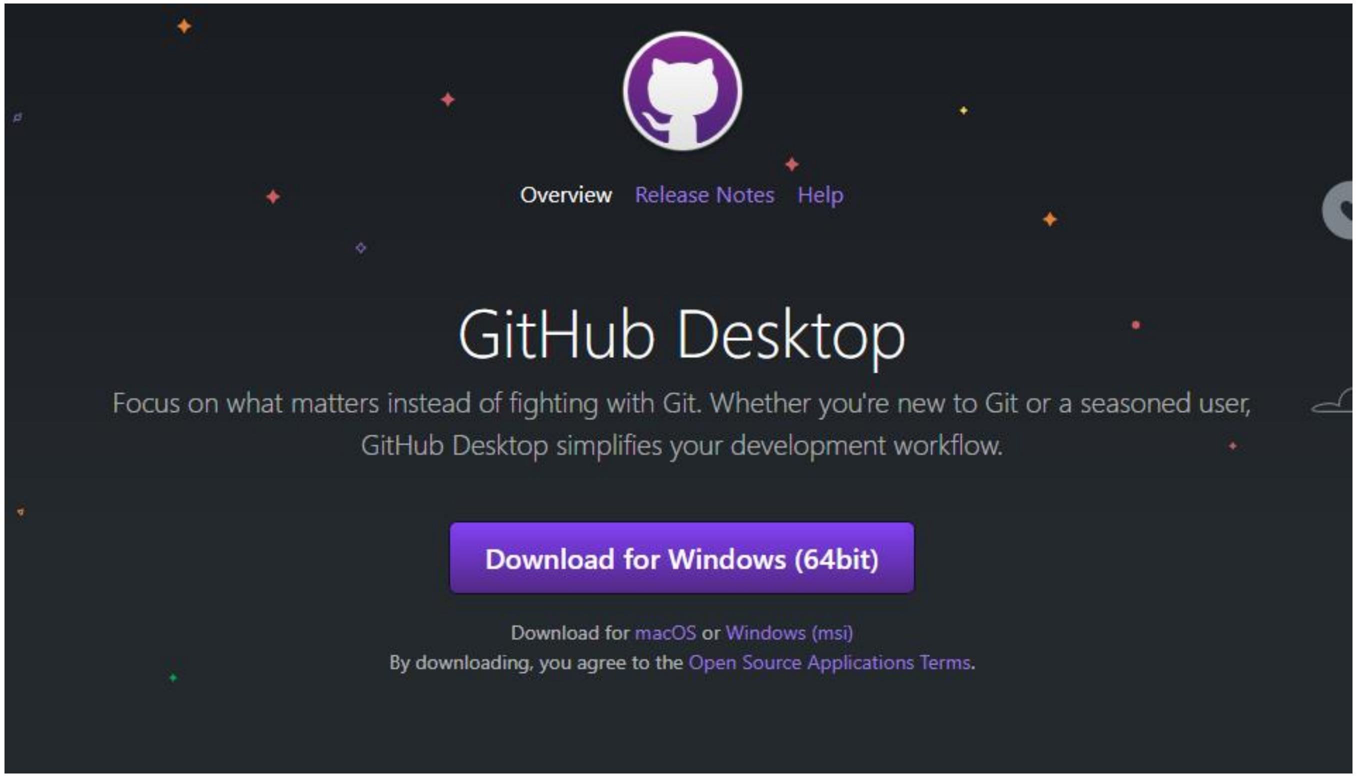
Older releases are available and the Git source repository is on GitHub.

Latest source Release
2.30.1
Release Notes (2021-02-08)

GitHub Desktop

Download and Install GitHub Desktop from here:

https://desktop.github.com/



GitHub Desktop

Focus on what matters instead of fighting with Git. Whether you're new to Git or a seasoned user, GitHub Desktop simplifies your development workflow.

Download for macOS or Windows (msi)
By downloading, you agree to the Open Source Applications Terms.

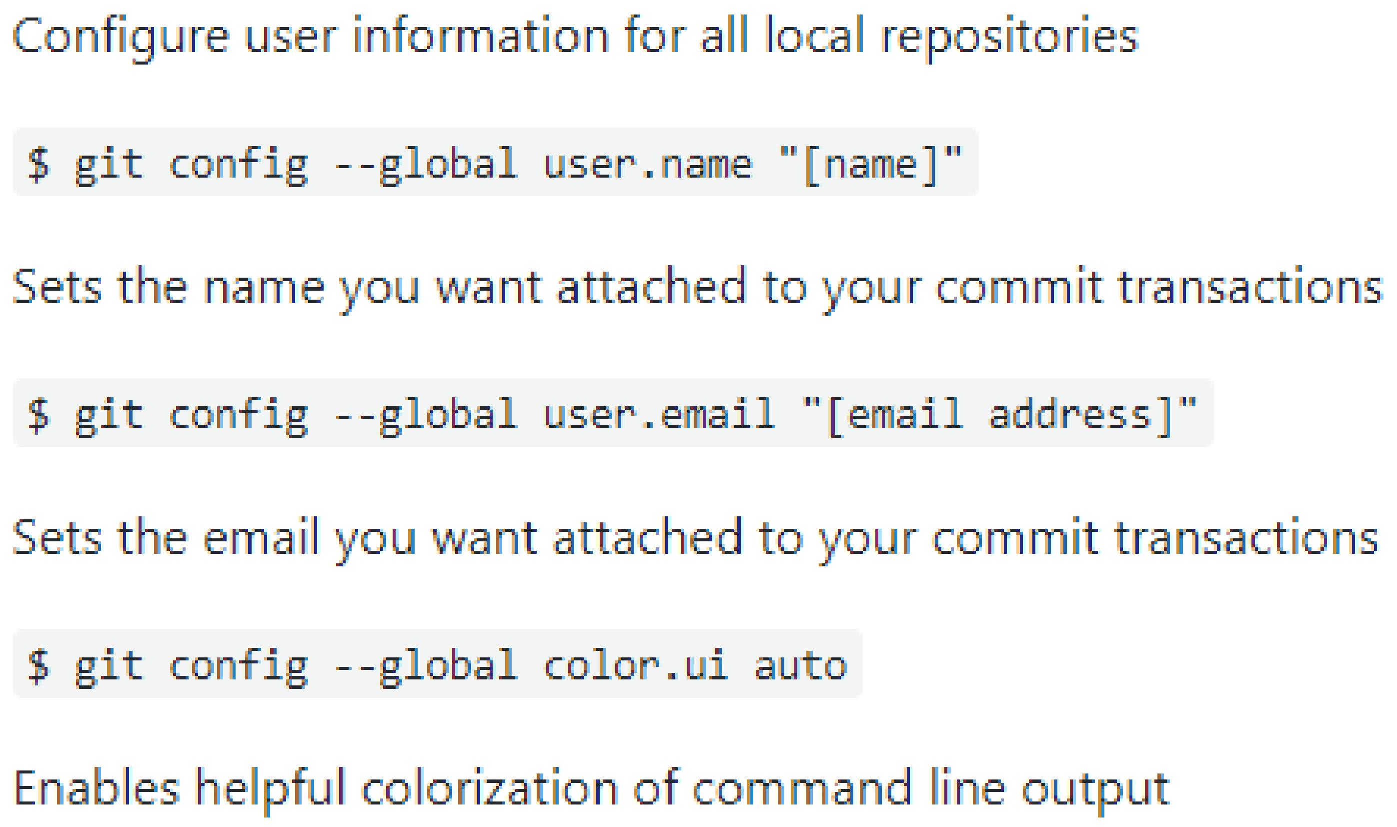
Git Initialization

Write the following codes in the command-line.

1. Check if Git is properly installed:

```
git --version
```

2. Put information about you (author in the Git):



Configure user information for all local repositories

```
$ git config --global user.name "[name]"
```

Sets the name you want attached to your commit transactions

```
$ git config --global user.email "[email address]"
```

Sets the email you want attached to your commit transactions

```
$ git config --global color.ui auto
```

Enables helpful colorization of command line output

Create Repository

A new repository can either be created locally, or an existing repository can be cloned. When a repository was initialized locally, you have to push it to GitHub afterwards.

```
$ git init
```

The git init command turns an existing directory into a new Git repository inside the folder you are running this command. After using the `git init` command, link the local repository to an empty GitHub repository using the following command:

```
$ git remote add origin [url]
```

Specifies the remote repository for your local repository. The url points to a repository on GitHub.

```
$ git clone [url]
```

Clone (download) a repository that already exists on GitHub, including all of the files, branches, and commits

Track Changes using Git

After making changes in your project file-

1. See the status using-
```
git status
```
2. Select the files using-
```
git add .
```
3. Commit the files using-
```
git commit –m “some message”
```

Useful Commands for tracking

```
$ git log
```

Lists version history for the current branch

```
$ git log --follow [file]
```

Lists version history for a file, beyond renames (works only for a single file)

```
$ git diff [first-branch]...[second-branch]
```

Shows content differences between two branches

```
$ git show [commit]
```

Outputs metadata and content changes of the specified commit

```
$ git add [file]
```

Snapshots the file in preparation for versioning

```
$ git commit -m "[descriptive message]"
```

Records file snapshots permanently in version history

To Revert to an older commit

1. Use **`git log`** to find out the hash code of the commit that you want to go to.
2. Use **`git checkout [commit]`** to view the state of the project on that commit
    - Use **`git checkout master`** to put the head back to the original position
3. Use **`git reset --hard [commit]`** to reset the project state to that commit and delete the commits in the front.

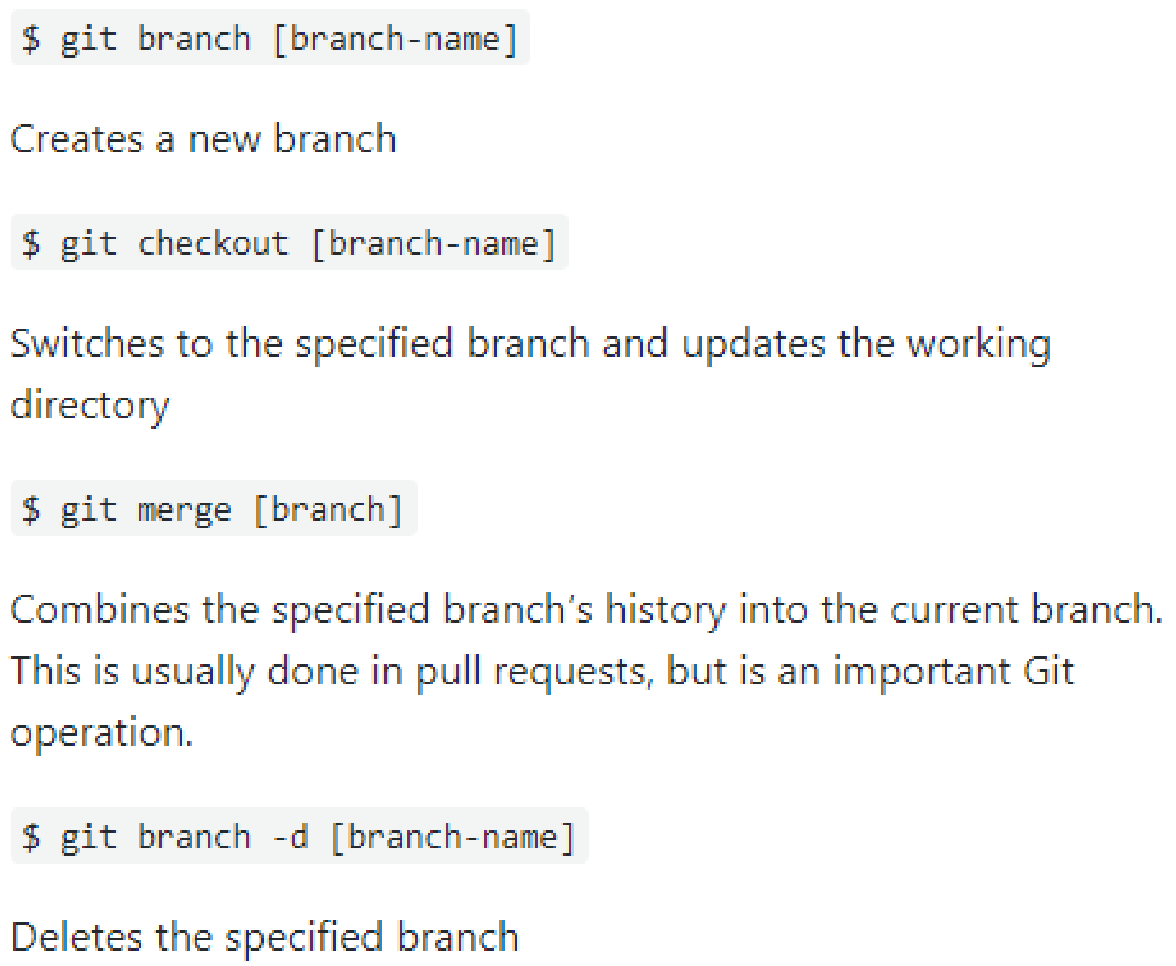
Use it with caution as it deletes all the commits between.

Useful Commands for Branching

Branches are an important part of working with Git. Any commits you make will be made on the branch you're currently "checked out" to. Use

```
git status
```

to see which branch that is.



```
$ git branch [branch-name]
```

Creates a new branch

```
$ git checkout [branch-name]
```

Switches to the specified branch and updates the working directory

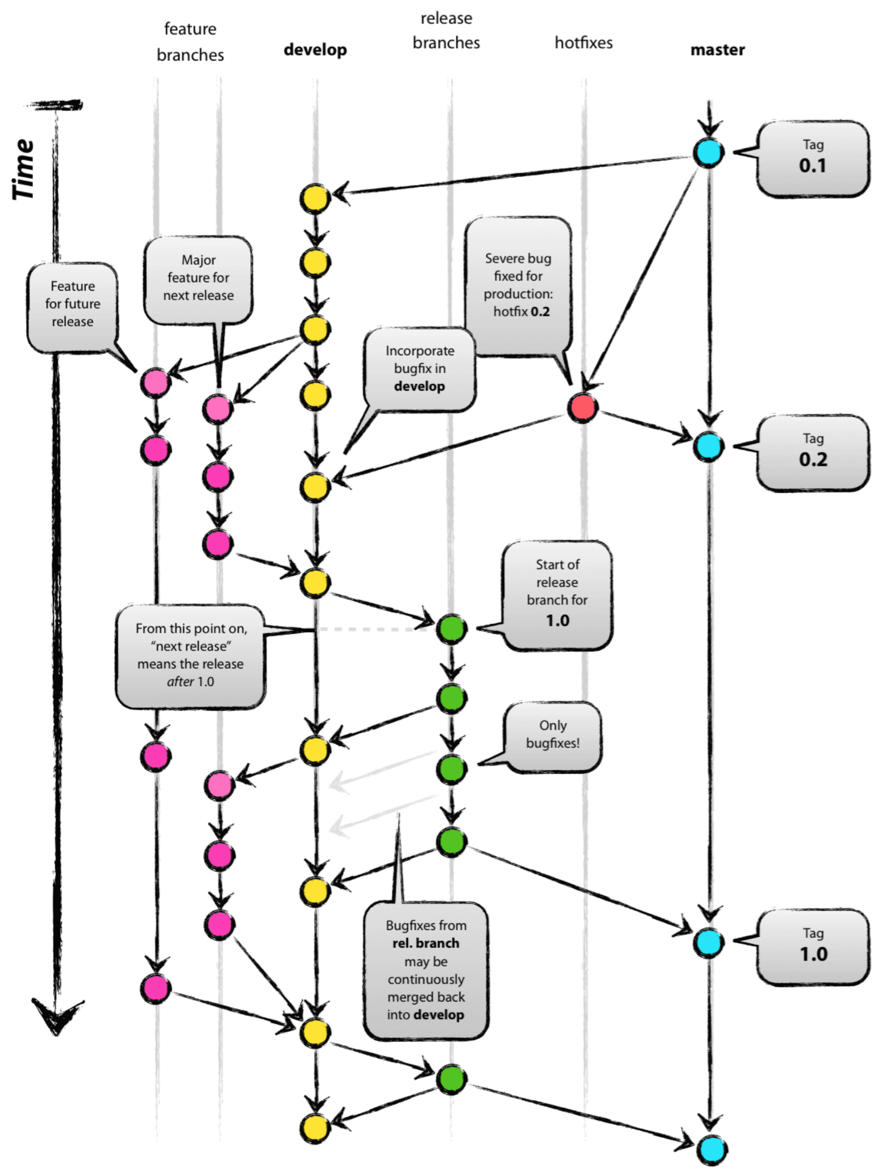
```
$ git merge [branch]
```

Combines the specified branch's history into the current branch. This is usually done in pull requests, but is an important Git operation.

```
$ git branch -d [branch-name]
```

Deletes the specified branch

Branching Strategy

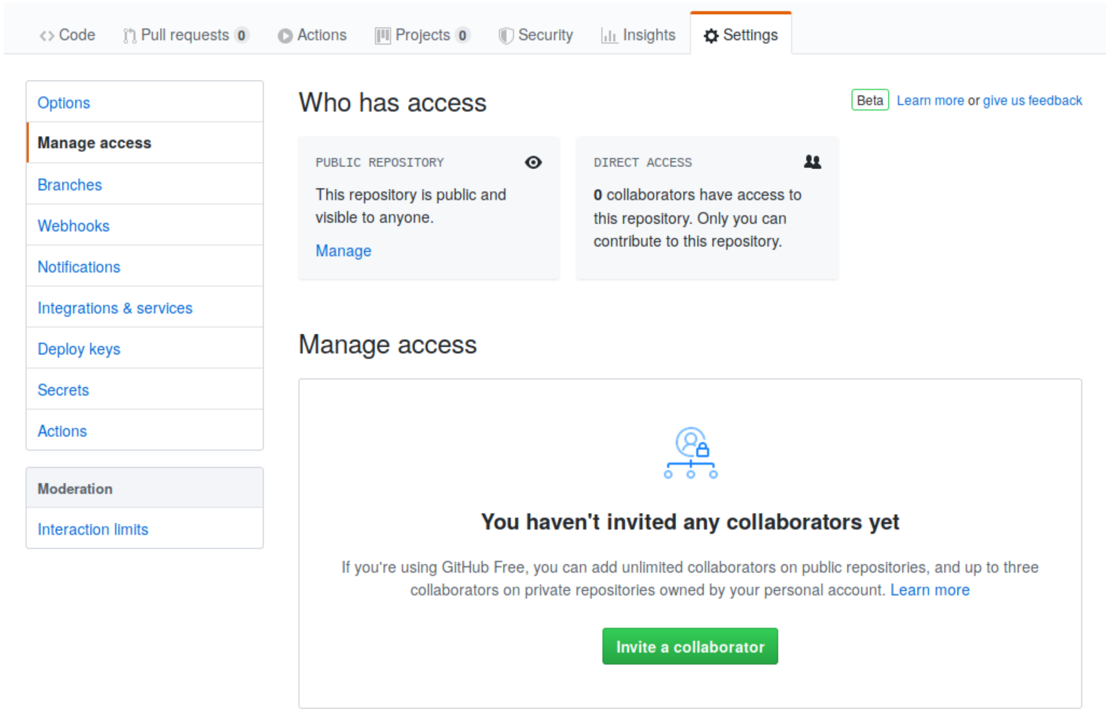


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Tag 0.2
Tag 1.0
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop

Adding Collaborator

If you want your team-mates to contribute your git repository, you'll have to add them as Collaborator in the GitHub repo.

1. Open the repo in the GitHub
2. Go to Settings > Manage Access (on the left pane) > Invite a collaborator
3. Add the GitHub accounts of your team-mates.



Who has access

PUBLIC REPOSITORY

This repository is public and visible to anyone.

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

You haven't invited any collaborators yet

If you're using GitHub Free, you can add unlimited collaborators on public repositories, and up to three collaborators on private repositories owned by your personal account. Learn more

Syncing Remote Branches

```
$ git fetch
```

Downloads all history from the remote tracking branches

```
$ git merge
```

Combines remote tracking branches into current local branch

```
$ git push
```

Uploads all local branch commits to GitHub

```
$ git pull
```

Updates your current local working branch with all new commits from the corresponding remote branch on GitHub. `git pull` is a combination of `git fetch` and `git merge`

Best Practice

1. Create a public repository in GitHub
2. The team members will clone the repo in their local PC
3. The master branch will ALWAYS be the working branch, not commit should be directly made to the master branch.
4. To create a feature, branch from your local repo and develop the feature in it. Commit every changes there.
5. After the feature is developed, merge it into the master and then publish it into the remote repository.

Concepts

- **Git**: an open source, distributed version-control system
- **GitHub**: a platform for hosting and collaborating on Git repositories
- **Commit**: a Git object, a snapshot of your entire repository compressed into a SHA
- **Branch**: a lightweight movable pointer to a commit
- **Clone**: a local version of a repository, including all commits and branches
- **Remote**: a common repository on GitHub that all team members use to exchange their changes
- **Fork**: a copy of a repository on GitHub owned by a different user
- **Pull request**: a place to compare and discuss the differences introduced on a branch with reviews, comments, integrated tests, and more
- **HEAD**: representing your current working directory, the HEAD pointer can be moved to different branches, tags, or commits when using git checkout

Reference

1. https://training.github.com/downloads/github-git-cheat-sheet/
2. https://swcarpentry.github.io/git-novice/08-collab/index.html
3. https://git-school.github.io/visualizing-git
4. https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners