Quick Reference Guide

Definitions

Local Repository -

A git repository that is on your local machine. It will only save changes that are made on YOUR computer.

Remote Repository -

A git repository that is accessible to many different people, such as GitHub. It will only contain changes that have been pushed to the remote repository.

Commit -

A set of saved changes that are stored in a Git repository. The repository could be local or remote.

Push -

Sending commits from your local Git repository to a remote Git repository, whereupon the remote Git repository is altered.

Cloning -

Cloning a repository is a copy paste of a remote Git repository into a local repository.

Pull -

Compares your local repository to the remote repository and adds all remote changes to your local repository.

Command / View	Shortcut
Changes	Ctrl+1
History	Ctrl+2
Push	Ctrl+P
Pull	Ctrl+Shift+P
Clone Repository	Ctrl+Shift+O
Show in Explorer	Ctrl+Shift+F

Master Git Manual

Section 1 - Using a Repository

This section covers the basics of how to use a GitHub repository. We assume that you have already created a GitHub account and have downloaded the GitHub Desktop app. Follow section 2 if you have yet to complete these tasks.

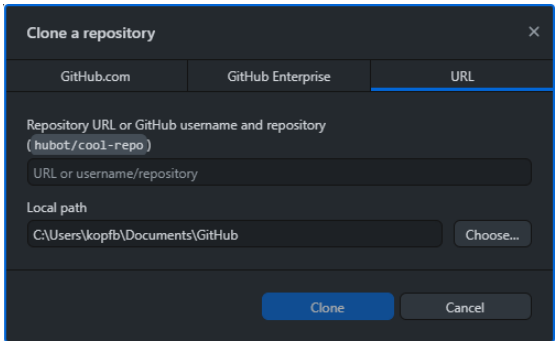
It is also assumed that you have been added to your GitHub project. If you are not, send your GitHub username to your Team Lead 1 so they can add you.

Cloning a Repository

1. Cloning a repository is the first thing you will do before working on your project (unless you already created the repository on your machine). A clone copies the remote repository into a local repository. This can be done on the GitHub web site and the GitHub desktop app.

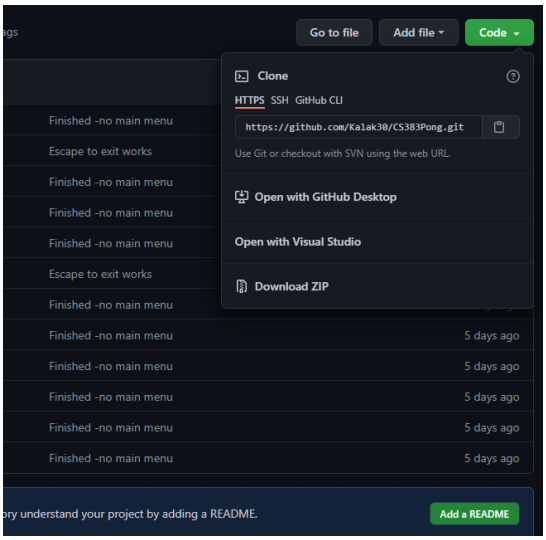
2. Desktop App

   Press **File > Clone Repository** (Ctrl+Shift+O) and select the URL tab. Enter the remote GitHub repository URL and the path for the local repository. Then press **clone**.



3. GitHub Website

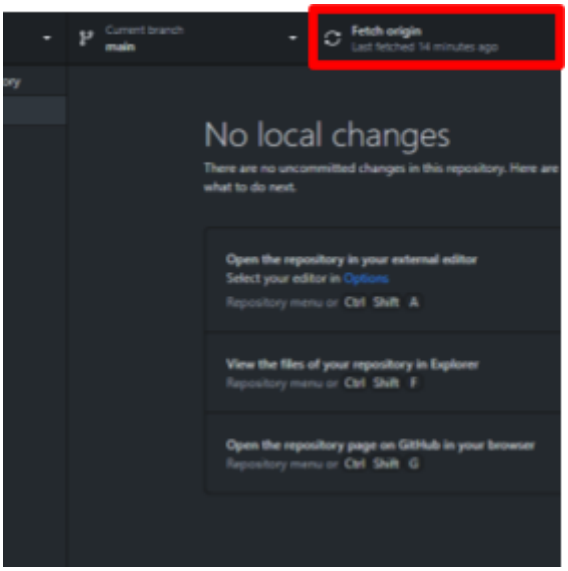
Navigate to the remote repository's main page and press the code dropdown. Then press the **Open with GitHub Desktop** button. This opens up the above picture with the remote repository's URL already entered. Finally press **clone**.



Pulling from a Repository

1. A Pull syncs your local repository and the remote repository by adding changes made in the remote repository to your local repository. This will allow you to stay up to date with your project.

2. On the GitHub desktop app, you can fetch the most recent updates, and pull them into your repo, by pressing the **Fetch origin** button in the top ribbon.



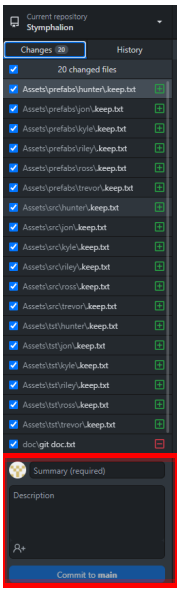
Master Git Manual

Committing Changes

1. A commit is a collection of saved changes to a local repository. Every commit has an associated commit message. The message explains why a particular set of changes was made. In the message, you should explain the reason/purpose of your commit.

2. How to make commits on the GitHub desktop app

   Once changes have been made in your local repository, they will appear on the left hand side of GitHub Desktop. Press **Ctrl+1** to make the changes list visible.

   At the bottom of the Changes panel, add a summary of changes made and a description of their purpose.

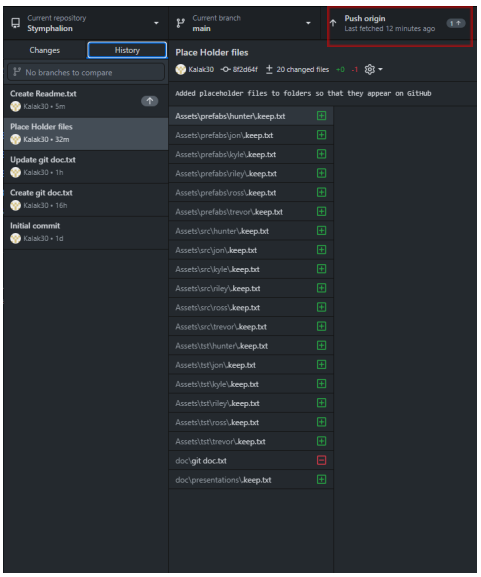
   Once you have adequately documented your changes you can commit your changes to a branch by pressing the **blue Commit button** at the bottom of the changes view.

   These changes can then be committed, or saved, to your **LOCAL** repository. This means that the changes will not yet appear on the remote GitHub repository.

Pushing Commits

1. A push in git is the action of cloning commits, saved changes, from your local repository to a remote repository. This is how you will keep the remote repository up to date.

2. Once you have one or more commits that you would like to confirm, you are ready to push those commits to the **remote repository**.

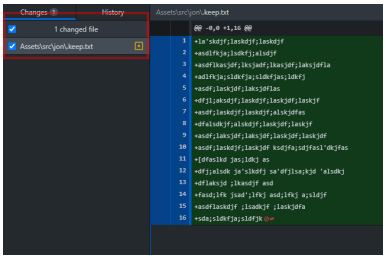
   To **push** your commits, press the Push origin button on the top ribbon. Make sure the repository you are pushing to is the repository for your game. You can also use the Push shortcut **Ctrl+P**.

Master Git Manual

Reverting Changes

1. When you mess something up and break your project right before a presentation you will probably want to revert to a stable, running build. You can revert your changes in several different ways.

2. Before Committing

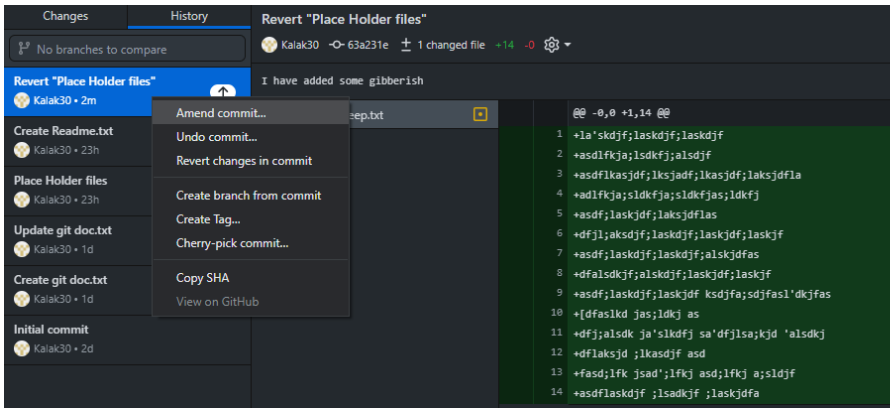


If you have yet to commit a change you can right click the file on the left side to discard all changes made to that file. You can also right click the top bar to discard all changes made to all files that have been selected.

You can also discard changes made to individual lines by right clicking the left hand side of the modified line, by the numbers and pressing the **discard line** button.

3. After Committing

Navigate to the history view in the GitHub desktop app. If you have yet to push your commit to the remote repository, then you can right click and press **undo commit**. This will turn all of the committed changes into uncommitted changes, and will appear in the changes view.



You can also right click and press the **revert changes in commit** to add a new commit reversing everything done in the selected commit. The reverted commit can then be pushed to the remote repository to roll back changes.

Master Git Manual

Branches

We will not be using branches for this project. The only reason a branch should be used is for releases to different platforms. For example, you may have a branch for an android release, windows release, vive release, etc.

Master Git Manual
Section 2 - Getting Started With Git

1. Go to https://github.com/join if you don't have an account, or **sign in** on Github.

2. Go to https://desktop.github.com to **download** Github Desktop. Once it's downloaded, run the installer called "GitHubDesktopSetup-x64.exe" and install.

3. After installing, **run** Github Desktop. From here there are two options:

    a. If you don't **already have a repository** Click "Create a New Repository on your hard drive…". Name your repository something appropriate, and add a description if you like. Choose the local path (on your hard drive) for where your repository will be stored.

        i. A "*.gitignore*" file is used to tell Github not to sync certain files back and forth. In the case of our Unity project, it will be useful to select the template Git Ignore file called "Unity". This will keep our project folder from getting too large.

        ii. Once you have your project set up on Github Desktop and linked with an online repository, you're ready to go!
            1. To get a **new *Unity* git project** set up, start by opening Unity Hub. Click the "New" button. From the *Create Project* screen, name your project, and under "location" choose the same folder as the folder for your Github repository. Commit your changes and push (sync with the repo) using the instructions from Section 1.

        iii. If you **already have a Unity project** set up, then locate the project file, make sure Unity is closed, then move the content of your Unity project folder into your local repository folder. Commit your changes and push (sync with the repo) using the instructions from Section 1.

    b. If you have already made the repository, and it's already online, and would like the files to be cloned to your computer, click "Clone a repository from the Internet…".