# CSTP:1206

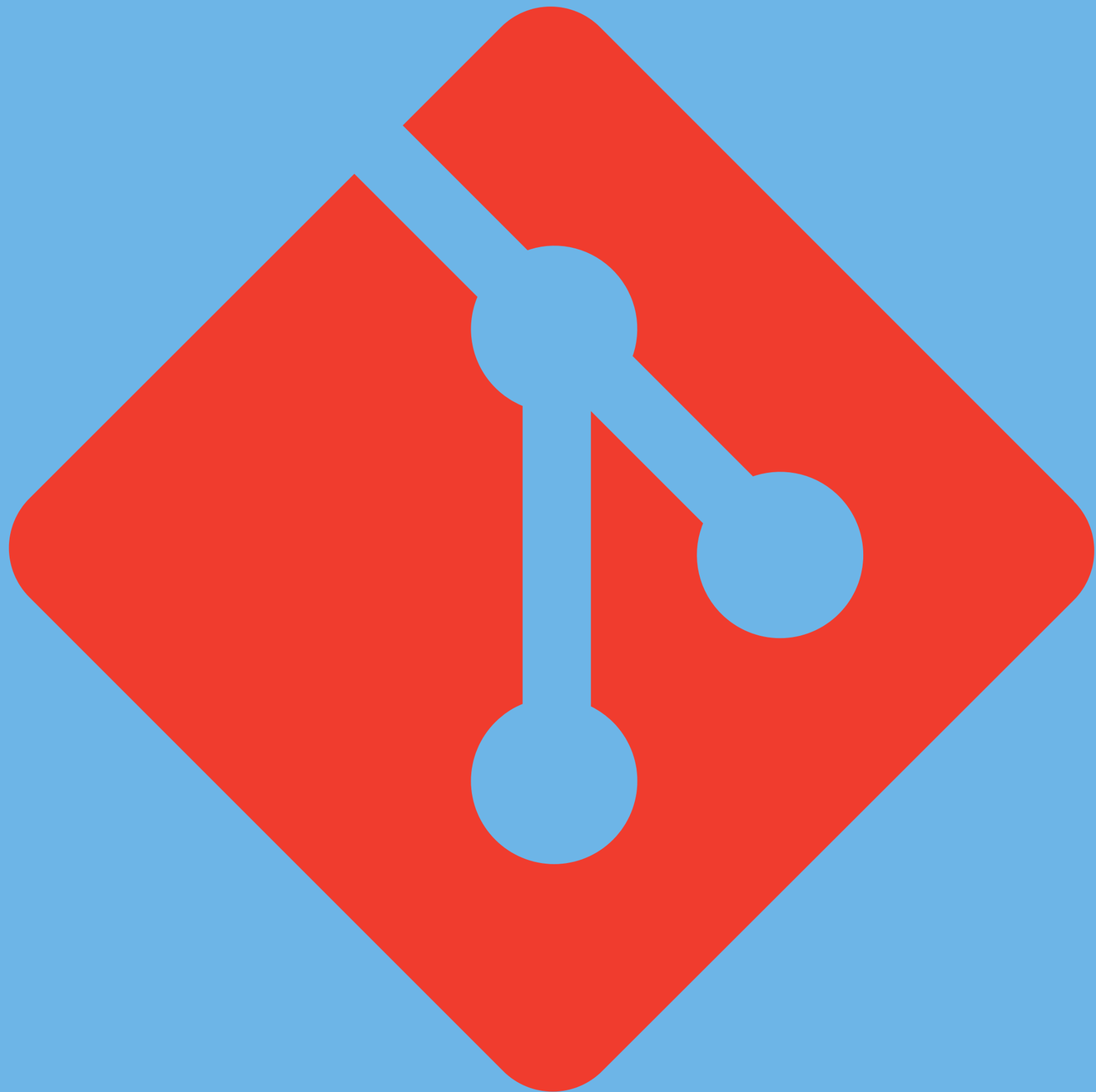
Introduction to Internet Programming & Web Applications

## Introduction to GIT and Github


**Prabhjyot Gambhir**
**(PRABH)**

# What is GIT ?

# Why is GIT ?



Almost every programming profession and industry uses Git, making it the most used tool overall.

# But Before learning GIT, what exactly is a Version Control System ?

**Version control** is a software that tracks and manages changes to files over time


You leave out version control.....
everyone starts losing thier minds!
quickmeme.com

In most systems, version control enables users to review previous versions of files, compare changes between versions, undo changes, and do a lot more.

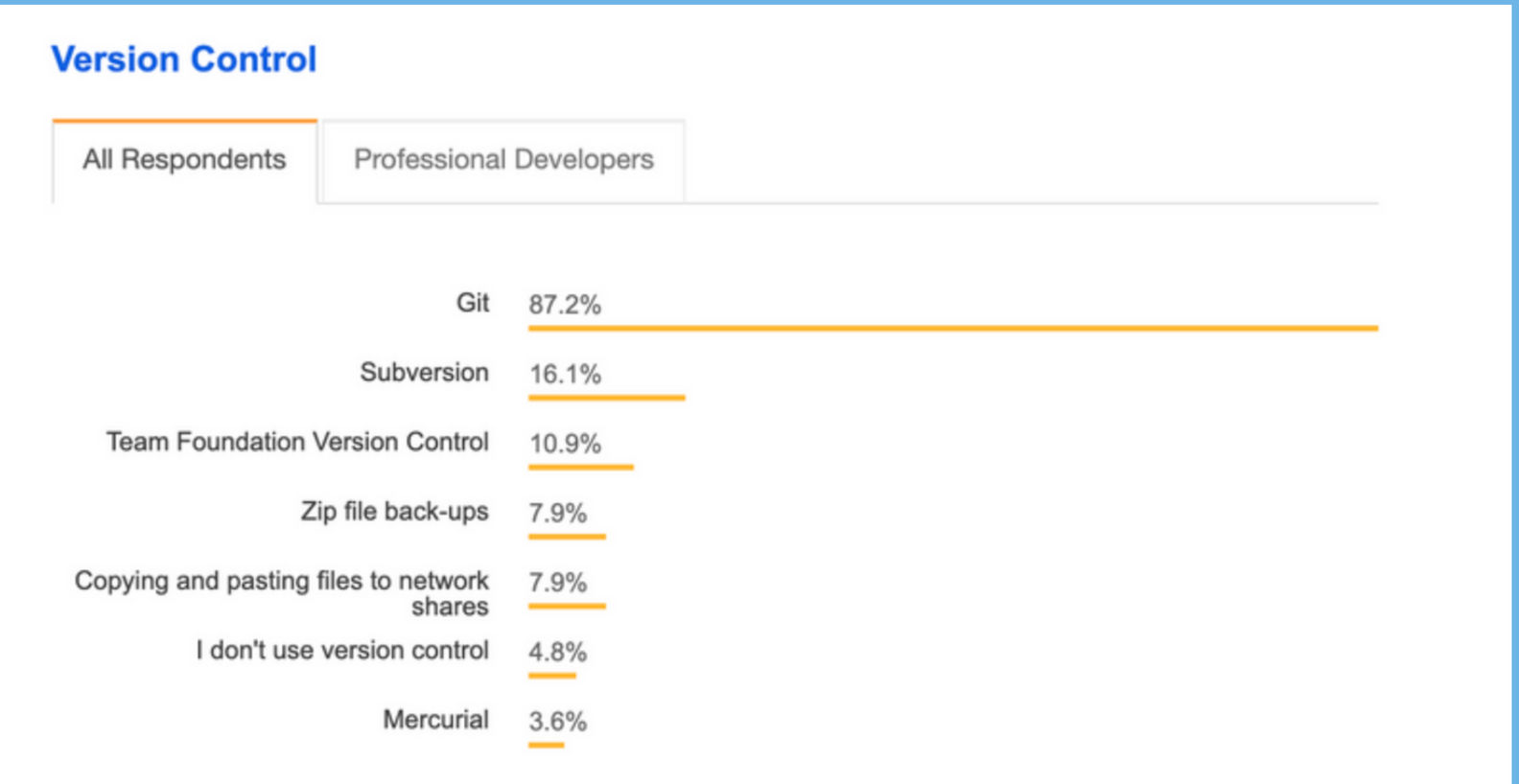
**GIT is one of the Version Control Systems.**
**There are others as well like**

- **Mercurial**
- **Subversion**
- **CVS**

GIT

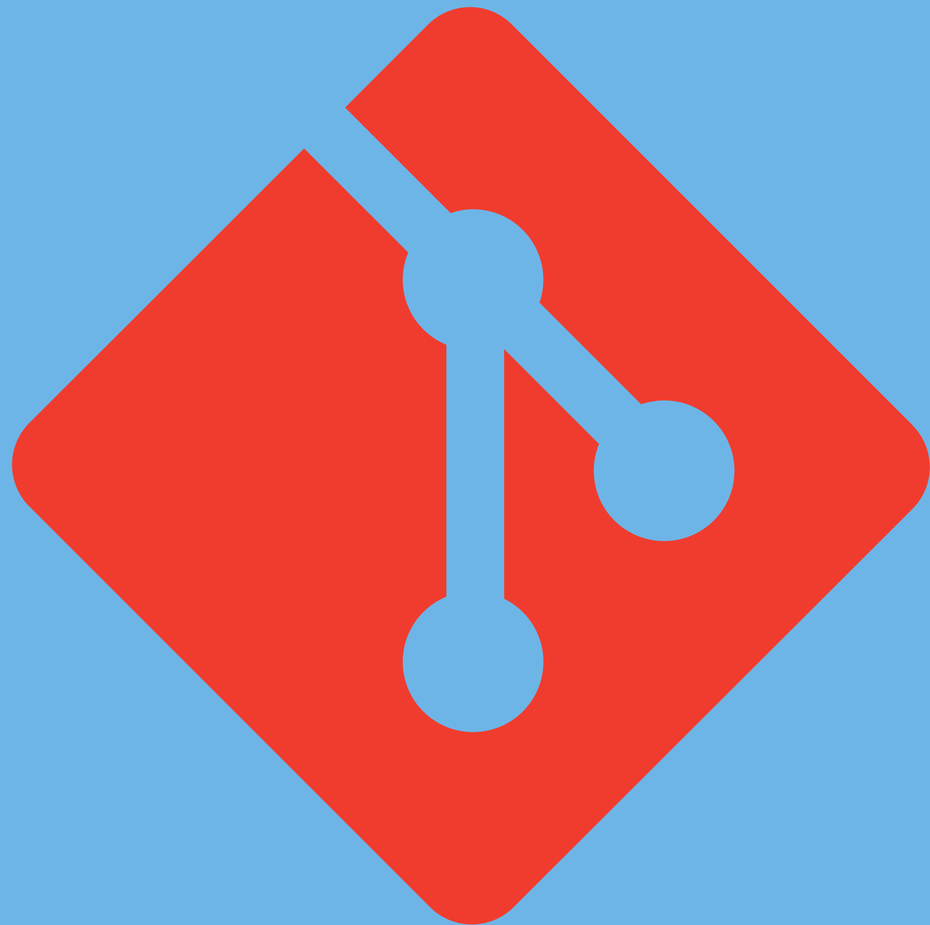
# GIT visibly leads marginally!

**Nearly 90% of respondents to Stack Overflow's 2018 Developer Survey indicated that Git was their preferred version management solution. Git is so commonly used that the poll hasn't even bothered to inquire about version control in the past few years.**



# GIT-TO-THE-POINT !

But why do we exactly need GIT ?



# So GIT helps us in!

Track changes in multiple files

Time Travel to previous versions

Collaborate with others

Compare versions in project

Revert to previous change

Combine Changes


Red's house
Player Red
Time 38:04
Badges 8
Pokédex 493/493
Yes
No
Would you like to save the game?


final.**psd**


updatedfinal.**psd**

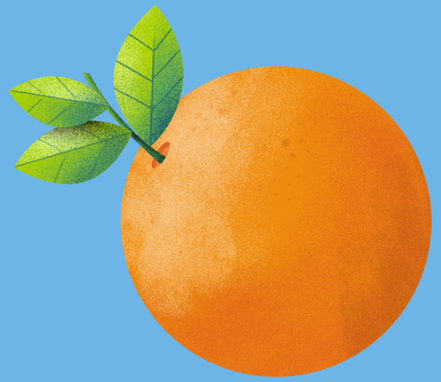

reallyfinal.**psd**


seriouslyfinal.**psd**


pleaseletthisb
efinal.**psd**


-$=#%{>*]^+%.**psd**

# GIT ≠ GITHUB



# GIT 

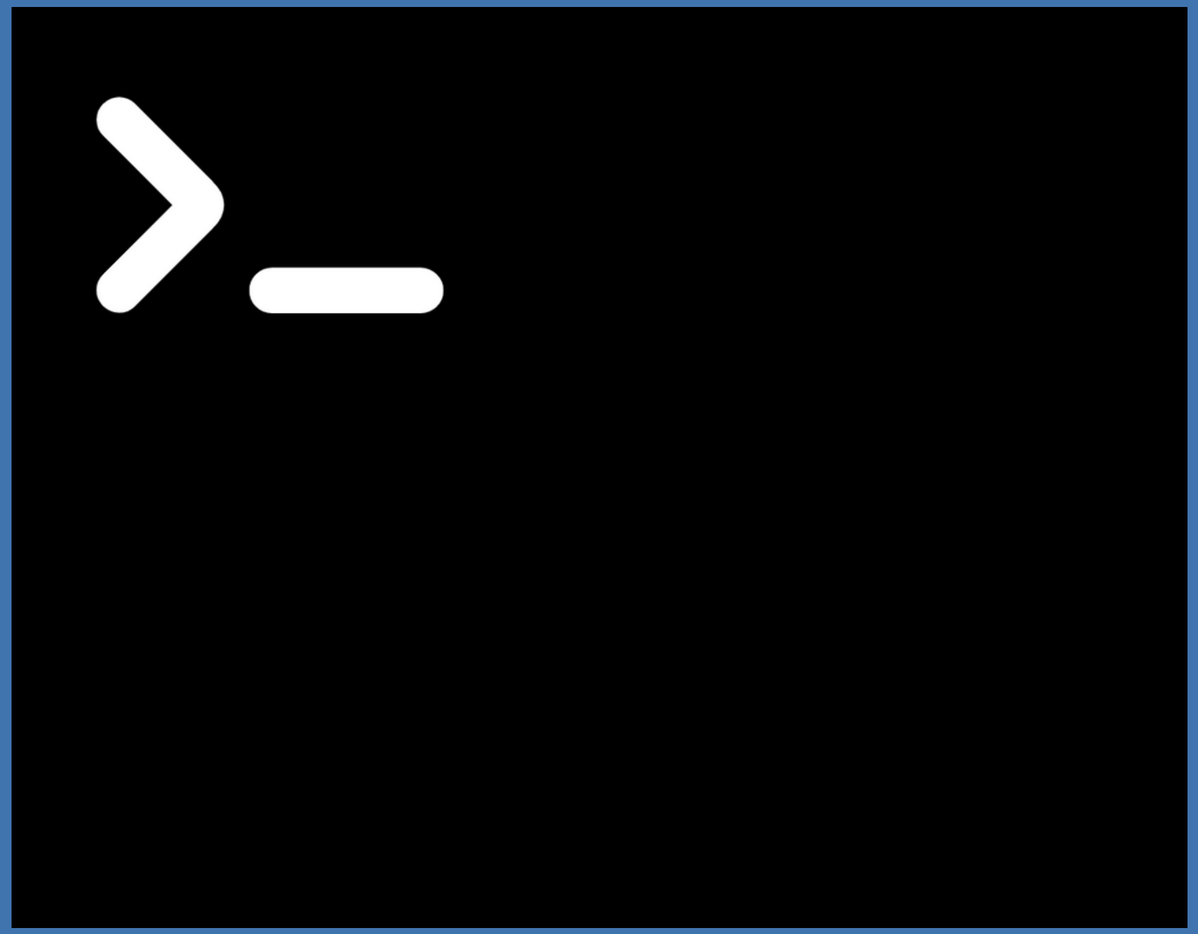
Git is the version control software that runs locally on your machine. You don't need to register for an account. You don't need the internet to use it. You can use Git without ever touching Github

# GITHUB 

Github is a service that hosts Git repositories in the cloud and makes it easier to collaborate with other people. You do need to sign up for an account to use Github. It's an online place to share work that is done using Git.

# Installing GIT

**GIT is primarily a CLI based tool.We execute several git commands in a Unix shell to use it. Although it's not the most user-friendly environment, git is built on this.a**



# Graphical User Interfaces

**Companies have developed graphical user interfaces for Git over the past few years, enabling individuals to use Git without needing to be command-line experts.**

**Popular Git GUIs are as follows:**

- **Use Github Desktop**
- **SourceTree**
- **Tower**
- **GitKraken**
- **Ungit**

# Installing GIT

But first check if you have git already installed in your computer by running the command

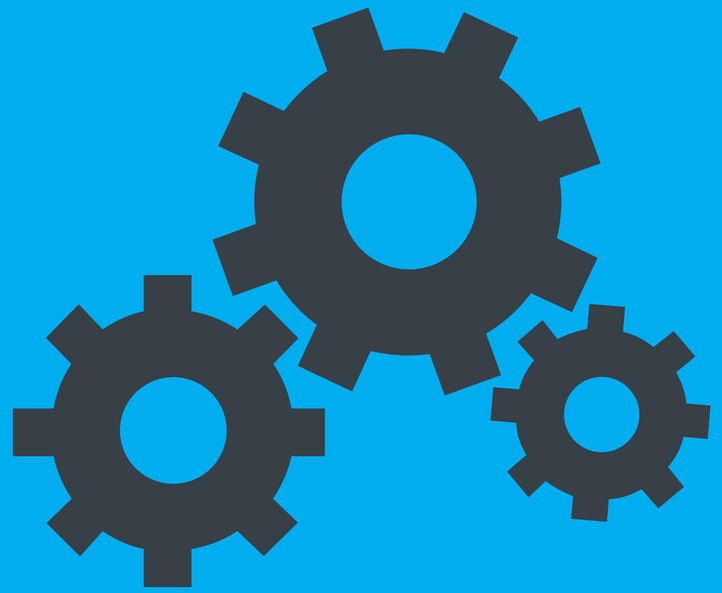
# MAC USERS

https://sourceforge.net/projects/git-osx-installer/

# WINDOW USERS

https://git-scm.com/download/win

# NOW LETS CONFIGURE GIT



# Configure your name

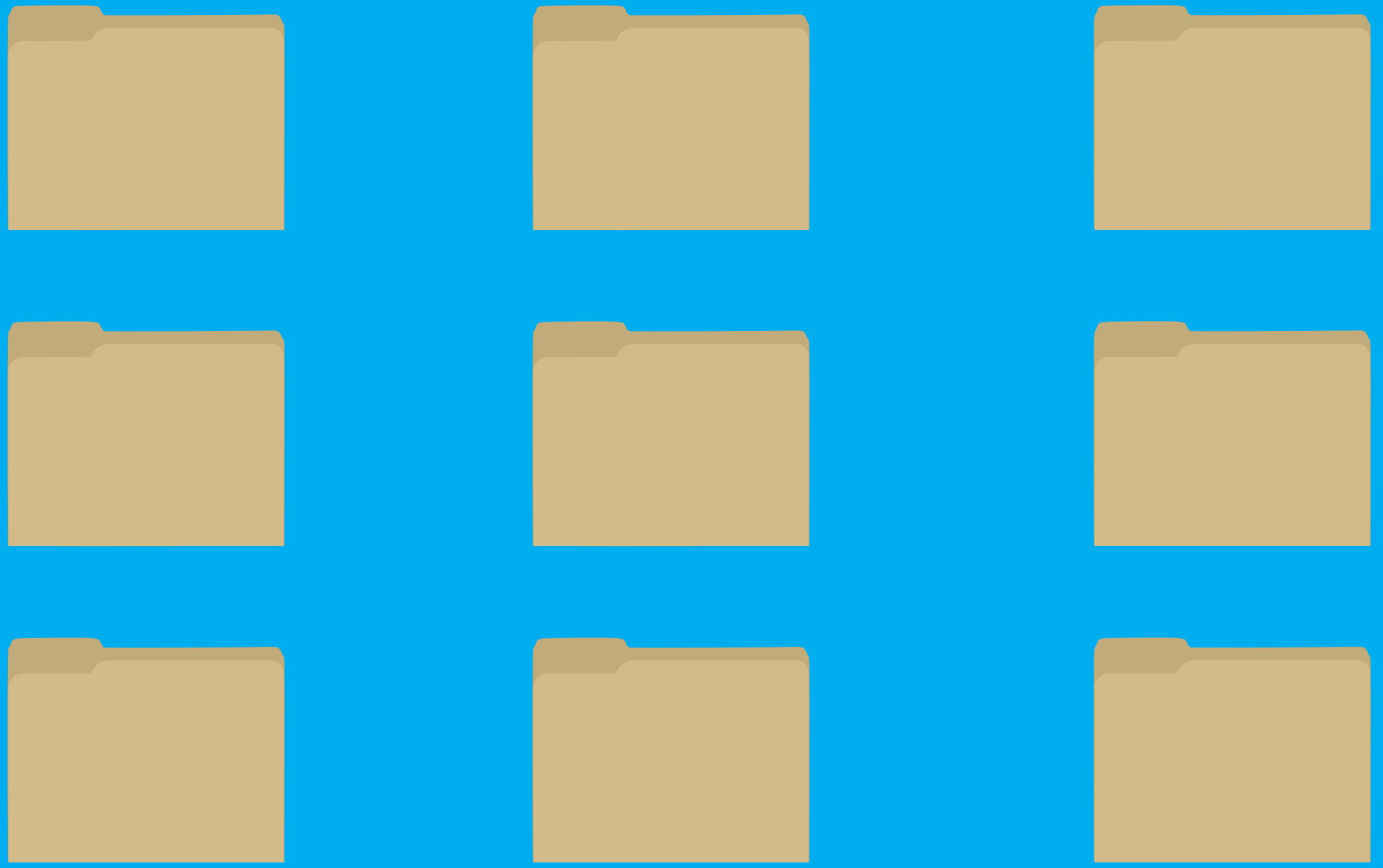
```
prabhjyotgambhir@Prabhs-MacBook-Pro ~ % git config --global user.name "Prabhjyot Gambhir"
```

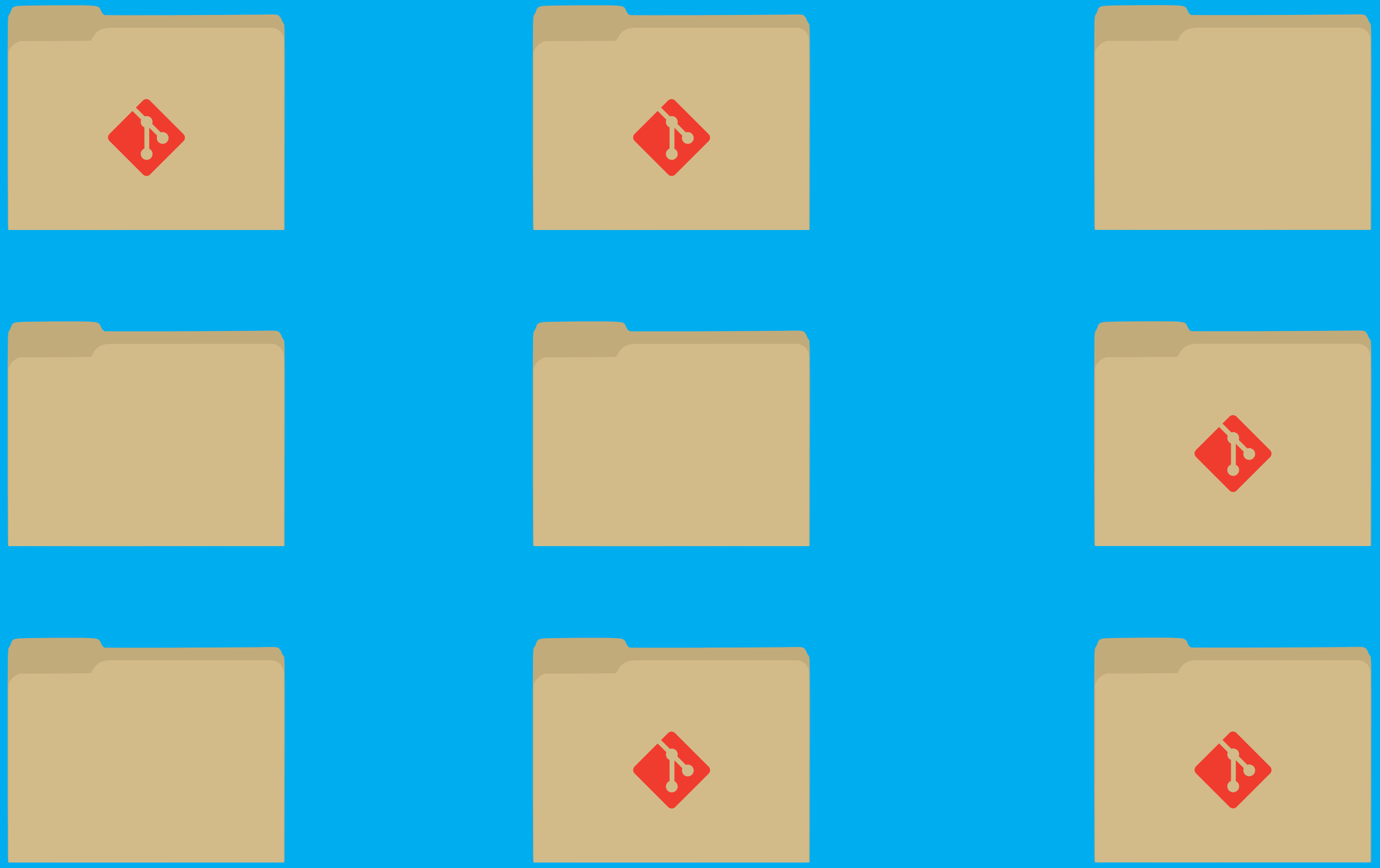
# Configure your Email

```
prabhjyotgambhir@Prabhs-MacBook-Pro ~ % git config --global user.email "knowprabhjyot@gmail.com"
```

# GIT REPOSITORY ?

A workspace called a Git "Repo" is used to track and manage files inside of folders.
A fresh git repository must be created each time we want to use Git with a project, application, etc. We can store as many repos as we need on our computer, each with its own history and contents.





# First Git Command

## Git Init

**To start a new git repository, use git init. We must setup a repo first before we can perform any git-related actions! You just need to do this once for each project. Initialize the repository in the project's top-level folder.**

```
prabhjyotgambhir@Prabhs-MacBook-Pro hello % git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/prabhjyotgambhir/Desktop/hello/.git/
prabhjyotgambhir@Prabhs-MacBook-Pro hello %
```


WARNING
DO NOT INIT A REPO INSIDE OF A REPO!
Before running git init, use git status to verify that you are not currently inside of a repo.

# Second Git Command

## Git status

**Git status provides details on a git repository's current status and its contents.**
**Although we don't currently have any repos to check the progress of, it is still incredibly helpful.**

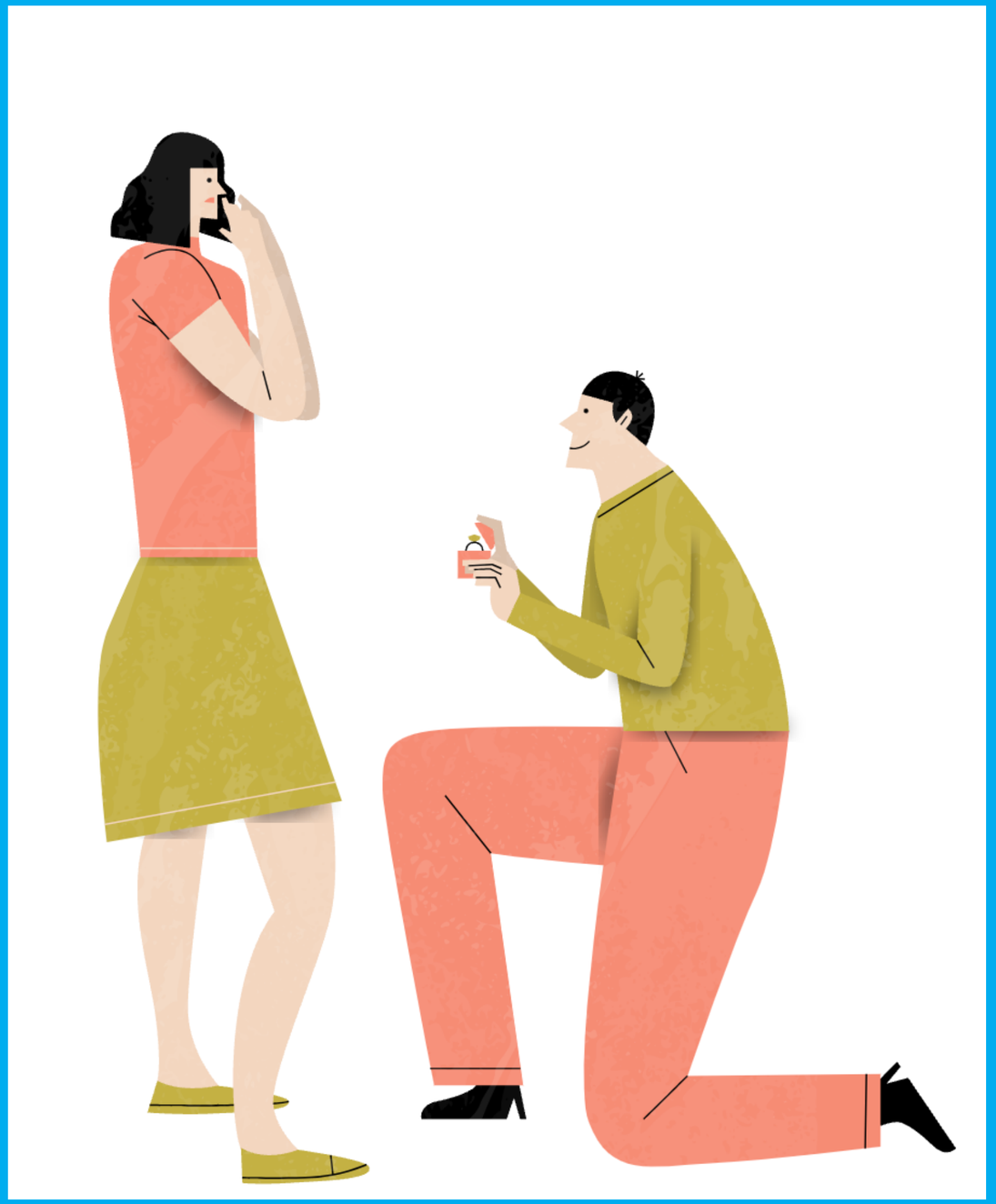
```
prabhjyotgambhir@Prabhs-MacBook-Pro hello % git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
prabhjyotgambhir@Prabhs-MacBook-Pro hello %
```

# GIT Commit

## The most important GIT feature

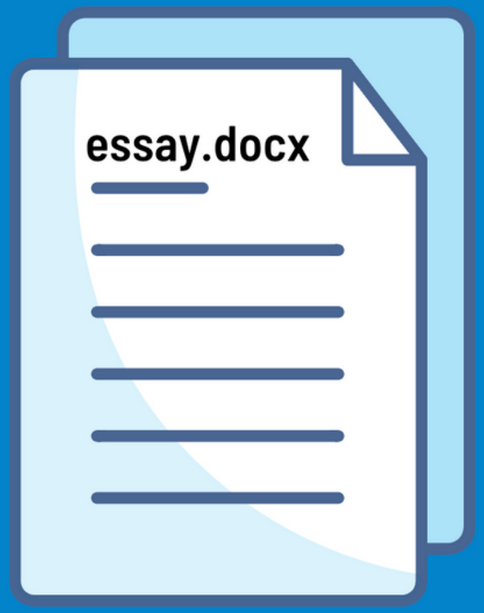


# GIT Commit

Similar like saving a game in a video game, committing is the same. We are momentarily taking a snapshot of a git repository.

We save a single file's state when we save a file. Git allows us to save the combined state of several files and directories.a

# Save




# Commit



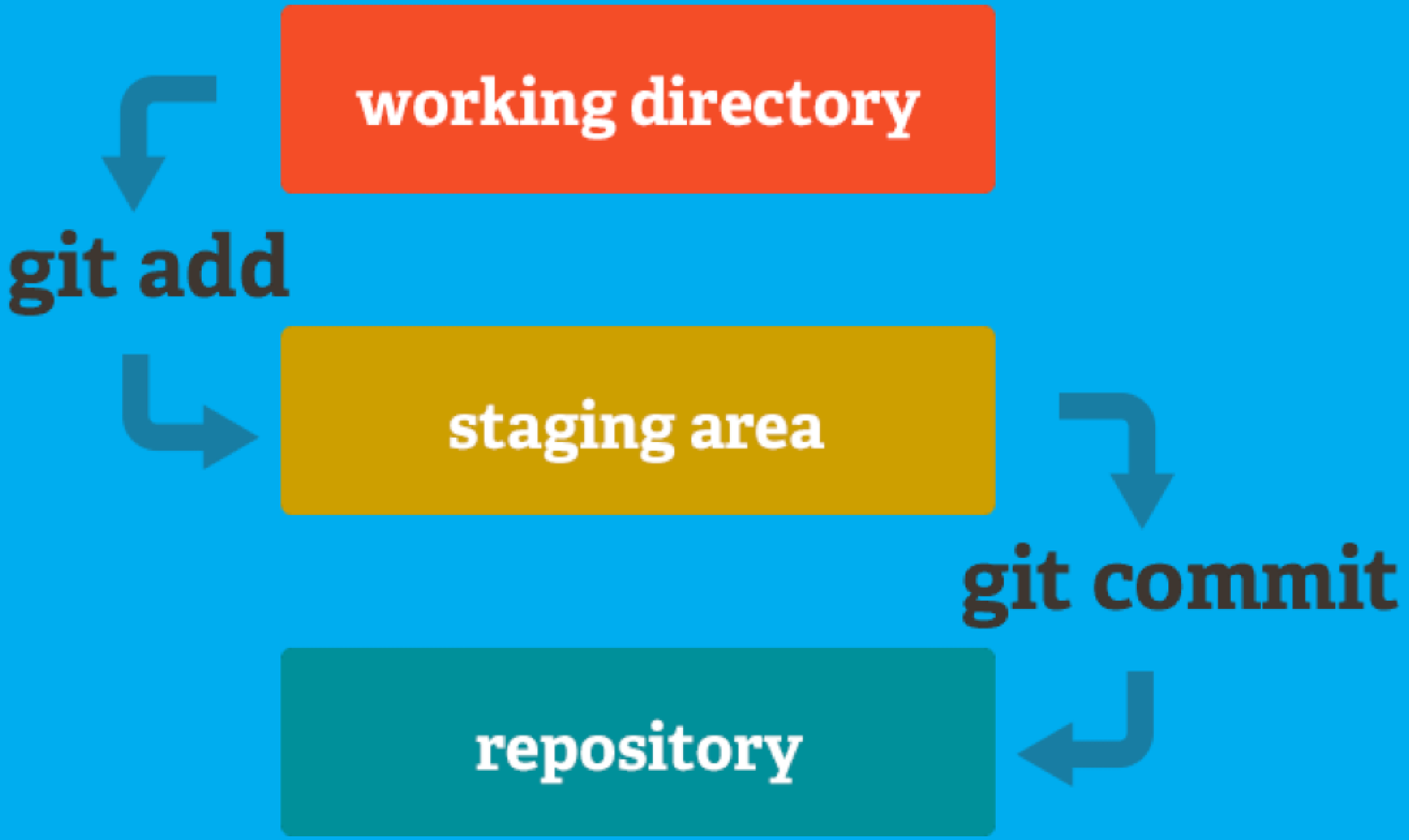


# The Basic Git Flow

Work on some files

Add Changes

Commit


working directory
git add
staging area
git commit
repository

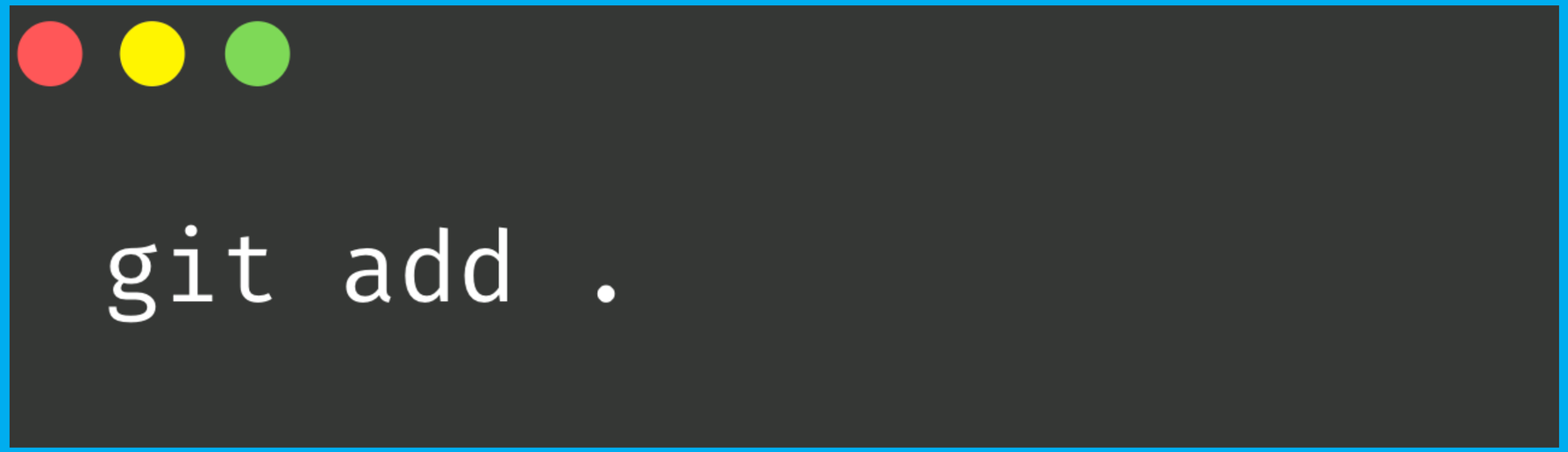
# Git Add

Use git add to add specific files to the staging area. Separate files with spaces to add multiple files at once

You can think of the staging area as a prep table where Git will take the next commit. Files on the Staging Index are poised to be added to the repository.

# Git Add

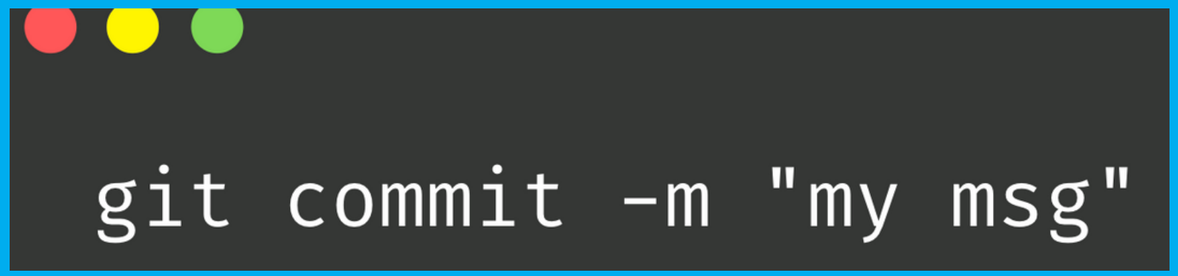
**Use git add . to stage all changes at once**



# Git Commit

We use the git commit command to actually commit changes from the staging area.

When making a commit, we need to provide a commit message that summarizes the changes and work snapshotted in the commit

# GIT IGNORE

Create a.gitignore file at the repository's root. To instruct Git which files and folders to ignore, we can write patterns inside the file:

.DS Store will ignore files named .DS Store

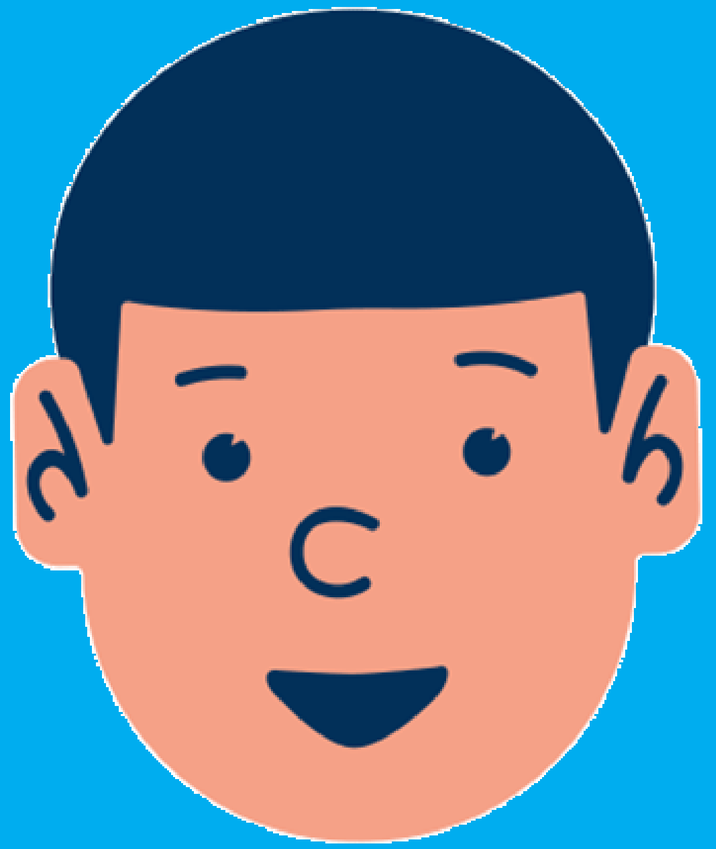
folderName/ will not consider a whole directory.

*.log will ignore any files with the .log extension

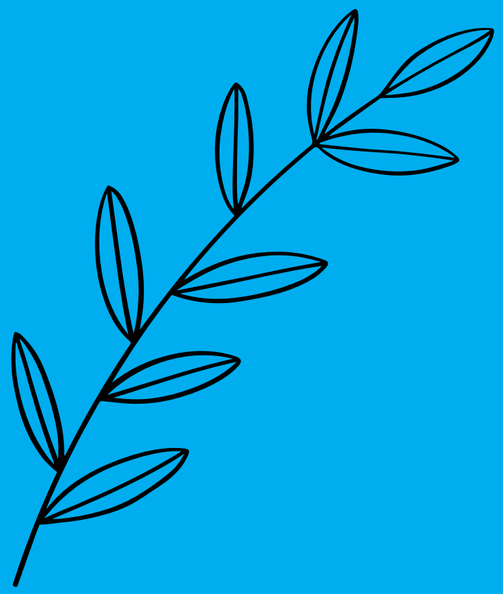
# HEAD

The word HEAD appears frequently in Git.
To refer to the current "place" in your repository, HEAD is merely a pointer. It designates a specific branch reference.

HEAD has always pointed to your most recent commit on the master branch, but shortly we'll see that we can move about and that HEAD will change!



# Creating Branches

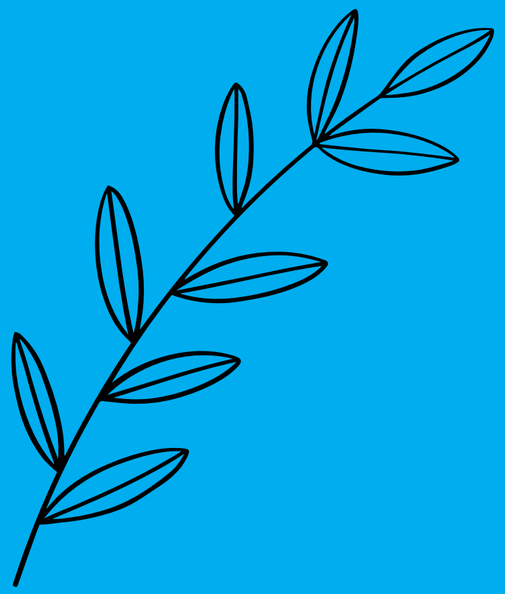


To create a new branch based on the current HEAD, use git branch "branch-name"

Simply said, this makes the branch. You are not transferred to that branch (the HEAD remains the same).

```
git branch <branch-name>
```

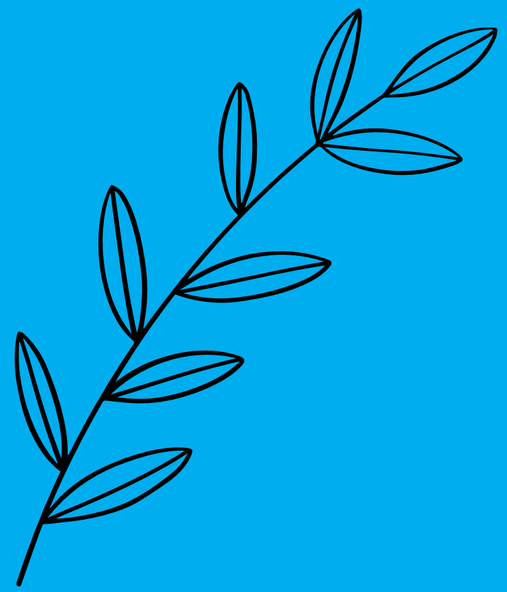
# Changing Branches



Once you have created a new branch,
use git switch <branch-name> to switch to it.

```
git switch <branch-name>
```

# Another way of changing



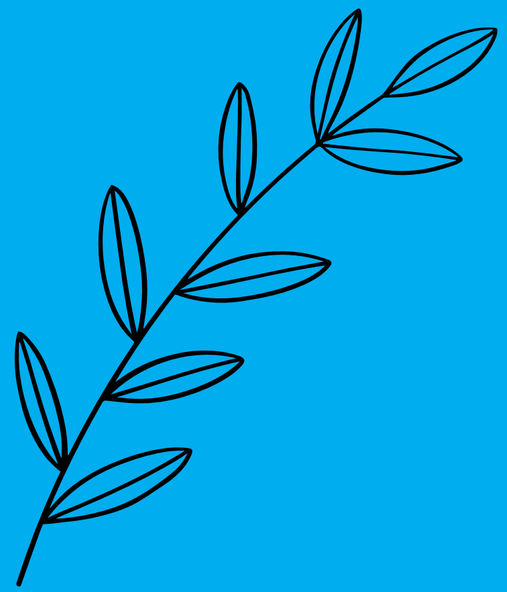
```
git checkout <branch-name>
```

In the past, we switched branches with git checkout branch-name>. This still functions.

The decision was taken to create a standalone switch command because it is significantly easier and does a fraction of what the checkout command does.

If you use checkout as opposed to switch, you will see older instructions and documents. Both are now employed.

# Create and Switch

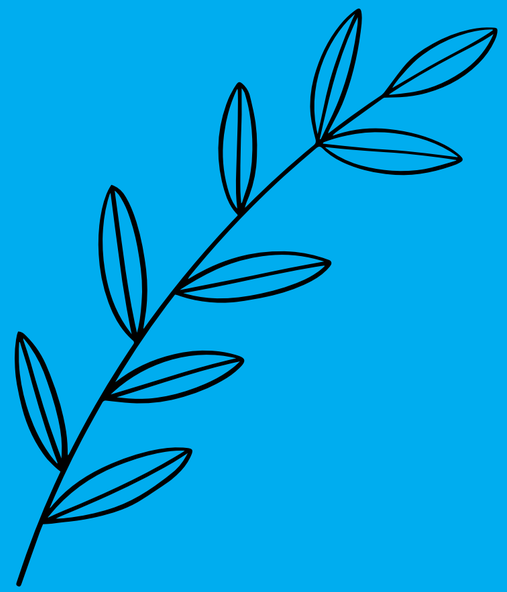


Use git switch with the -c flag to create a new branch AND switch to it all in one go.

Remember -c as short for "create"

```
git switch -c <branch-name>
```

# Delete Branch



You can also Delete a branch from your local by using this command

```
git branch -d bugfix
```

# GIT PRO BOOK

https://git-scm.com/book/en/v2

https://education.github.com/git-cheat-sheet-education.pdf