Lesson 5: Git and Github

[Overview](#h.1r5gf9s5dw37)

[Git](#h.psosri64ismx)

[Concept](#h.ltu2npu08t22)

[What is Git?](#h.1uzv5hwvfw46)

[How does it work?](#h.jhghtlpcecw)

[Workflow](#h.8881mp6xwct2)

[Try Git Tutorial](#h.hpxbz9u59w0x)

[We Do: TTS Ruby](#h.q3l0zy38a2k8)

[GitHub](#h.7l3bt8c13s8e)

[Concept](#h.k3d2myu02bzp)

[We Do: TTS Ruby](#h.6s4t26jwu54e)

[Homework](#h.ny09d7xhxo42)

[Deleting in Git](#h.yedngtw5vmx3)

# Overview

1. Understand what Git is, and basic commands: git init, git status, git add, git commit, git log, git diff
2. Understand what GitHub is and how to start a new repo and push to it. Also note important areas like contributions and teams.

# Git

## Concept

(mostly taken from <http://git-scm.com/>)

### What is Git?

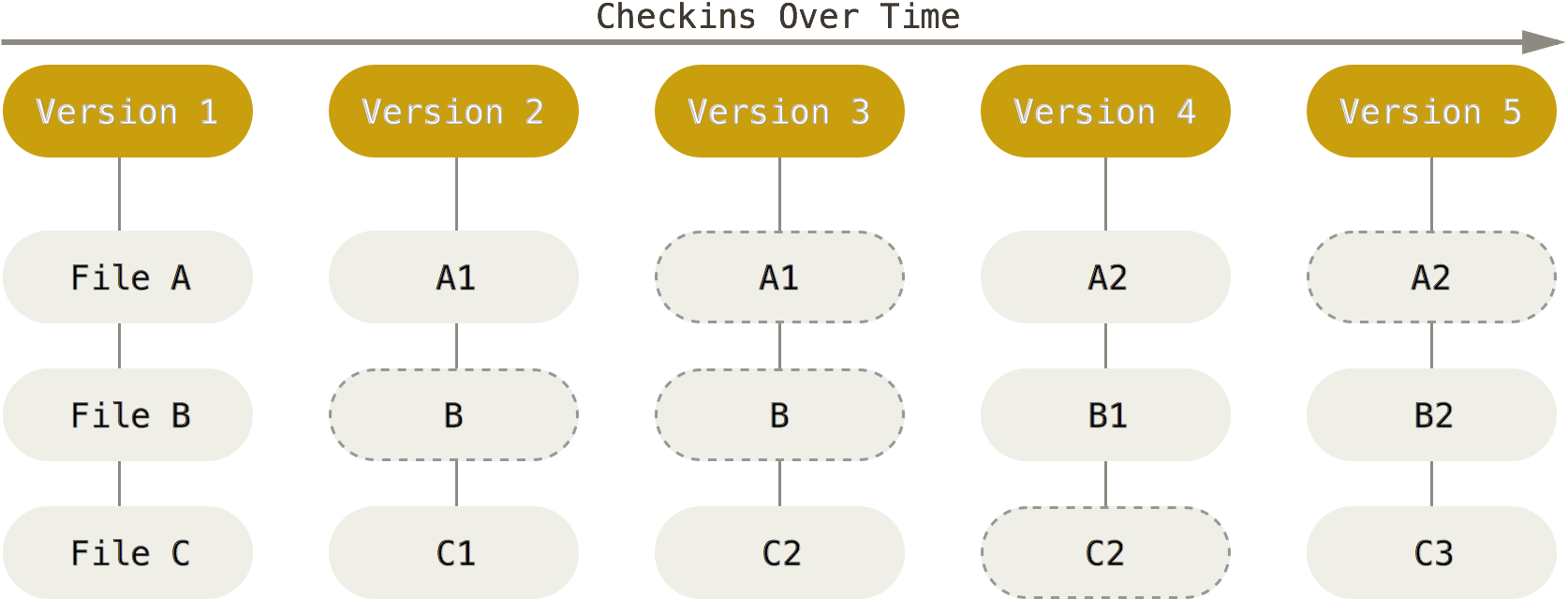
Git is a free and open source distributed version control system.

### How does it work?

Git thinks data like a **set of snapshots** of a miniature filesystem.

* Every time you **commit**, or save the state of your project in Git, it basically takes a picture of what all your files look like at that moment and **stores a reference** to that snapshot.
* To be efficient, if files have not changed, Git doesn’t store the file again, just a link to the previous identical file it has already stored.

Draw a simpler version of this on board:

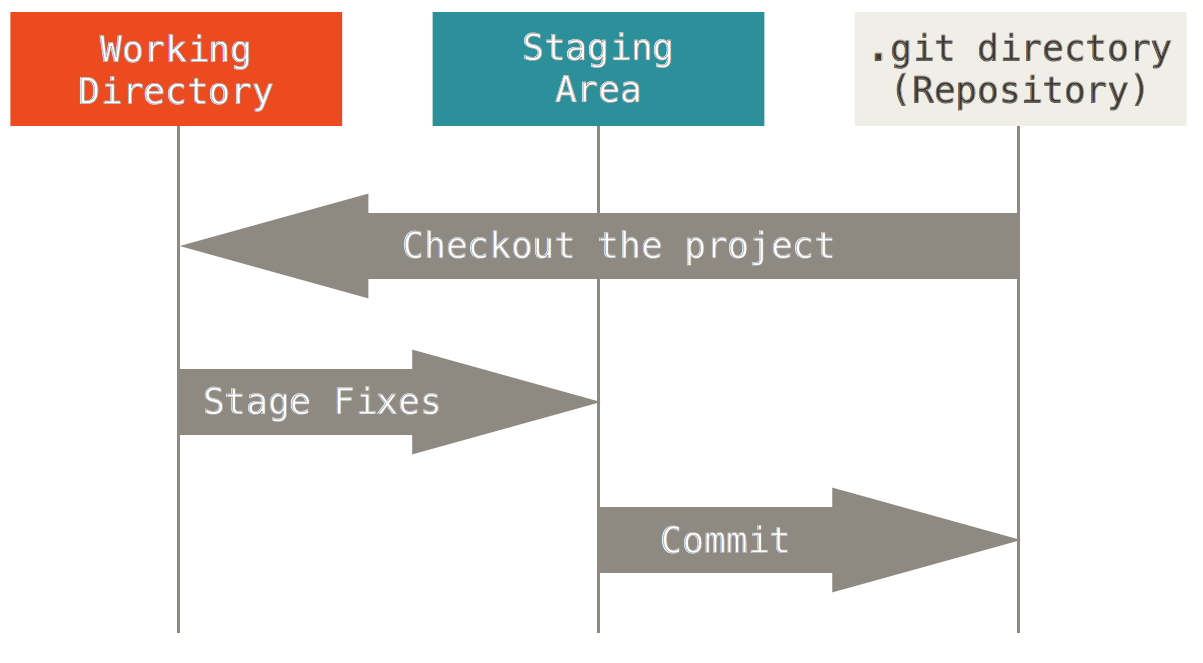


Git only needs local files and resources to operate – **you don’t need the internet**.

Git has **three main states** that your files can reside in:

* **committed** - the data is safely stored in your local database
* **modified** - you have changed the file but have not committed it to your database yet
* **staged** - you have marked a modified file in its current version to go into your next commit snapshot.

This leads us to the three main sections of a Git project: the Git directory, the working directory, and the staging area.



### Workflow

The basic Git workflow goes something like this:

1. You **modify** files in your working directory.
2. You **stage** the files, adding snapshots of them to your staging area. (**git add**)
3. You do a **commit**, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory. (**git commit -m “Note about what you changed”**)

## Try Git Tutorial

Work through <https://www.codeschool.com/courses/try-git> together through 1.9.

What are the major commands we just learned? git init, git add, git status, git commit

## We Do: TTS Ruby

Let’s turn our TTS Ruby folder into a Git repository. Pair with a classmate and write down what you would do to do this (5 minutes).

Do that together as a group.

# GitHub

(mostly taken from <http://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/>)

## Concept

GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features.

Git is a command-line tool with GitHub provides a **Web-based graphical interface**. It also provides access control and several **collaboration** features, such as a wikis and basic task management tools for every project.

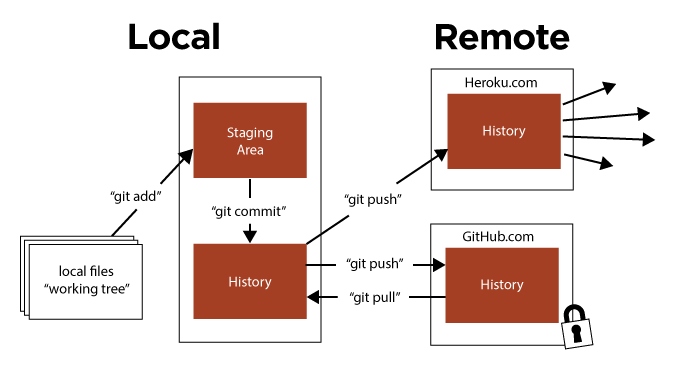
The flagship functionality of GitHub is “**forking**” – copying a repository from one user’s account to another. This enables you to take a project that you don’t have write access to and modify it under your own account. If you make changes you’d like to share, you can send a notification called a “pull request” to the original owner. That user can then, with a click of a button, merge the changes found in your repo with the original repo.

These three features – fork, pull request and merge – are what make GitHub so powerful.

Before GitHub, if you wanted to contribute to an open source project you had to manually download the project’s source code, make your changes locally, create a list of changes called a “patch” and then e-mail the patch to the project’s maintainer. The maintainer would then have to evaluate this patch, possibly sent by a total stranger, and decide whether to merge the changes.

When you submit a pull request, the project’s maintainer can see your profile, which includes all of your contributions on GitHub. If your patch is accepted, you get credit on the original site, and it shows up in your profile. It’s like a resume that helps the maintainer determine your reputation.

Draw this without Heroku:



## We Do: TTS Ruby

1. Login to GitHub.
2. Click the + sign to create a new repo.
3. Give it a unique name like “tts-ruby” and a good description.
4. Keep it listed publicly (free) and don’t initialize with a readme - you can always add it later.
5. Click “create repository”.
6. Don’t navigate away from this page. GitHub is super nice in that it gives you the exact commands you need to link your local repository with this new GitHub repository. (Read and evaluate the page).
7. Run the commands from the correct folder in command line.

# Homework

[Review and Git/Github Homework](https://docs.google.com/document/d/1gyqz2MUnldgPaoyt6JgFbI-WRo2995F2N0D0yoyuQXs/edit?usp=sharing)

# Deleting in Git

## git checkout

Deletes everything back to your last commit

## git stash

You made some changes and haven’t added or committed them yet but you want to quickly go back and view your last commit. Used a lot when switching between branches.

### git stash list

lists all current stashes

### git stash pop

Creates a new stash and gets rid of the last one

## git reset

You’ve committed a file (but haven’t pushed yet) but then change your mind and want to uncommit.

### git reset --soft

Goes back to a previous commit but keeps it staged. Doesn’t delete anything.

### git reset --hard

Unstages file and deletes all changes you made since last commit.

### git reset --mixed

Unstages file but keeps your changes. This is the default reset.

## git revert

You added, commited, and pushed and want to undo a commit (doesn’t have to be the last commit). This is for undoing commits when the code has already been pushed to a remote repository.

## git cherry-pick

You want to steal just one commit from another branch. Not really undoing, but easy way to grab just one thing.

## 

## git log

Shows history of commits in this repository

## git reflog

Shows history of all your local git commands

## git clean -f

Removes untracked files