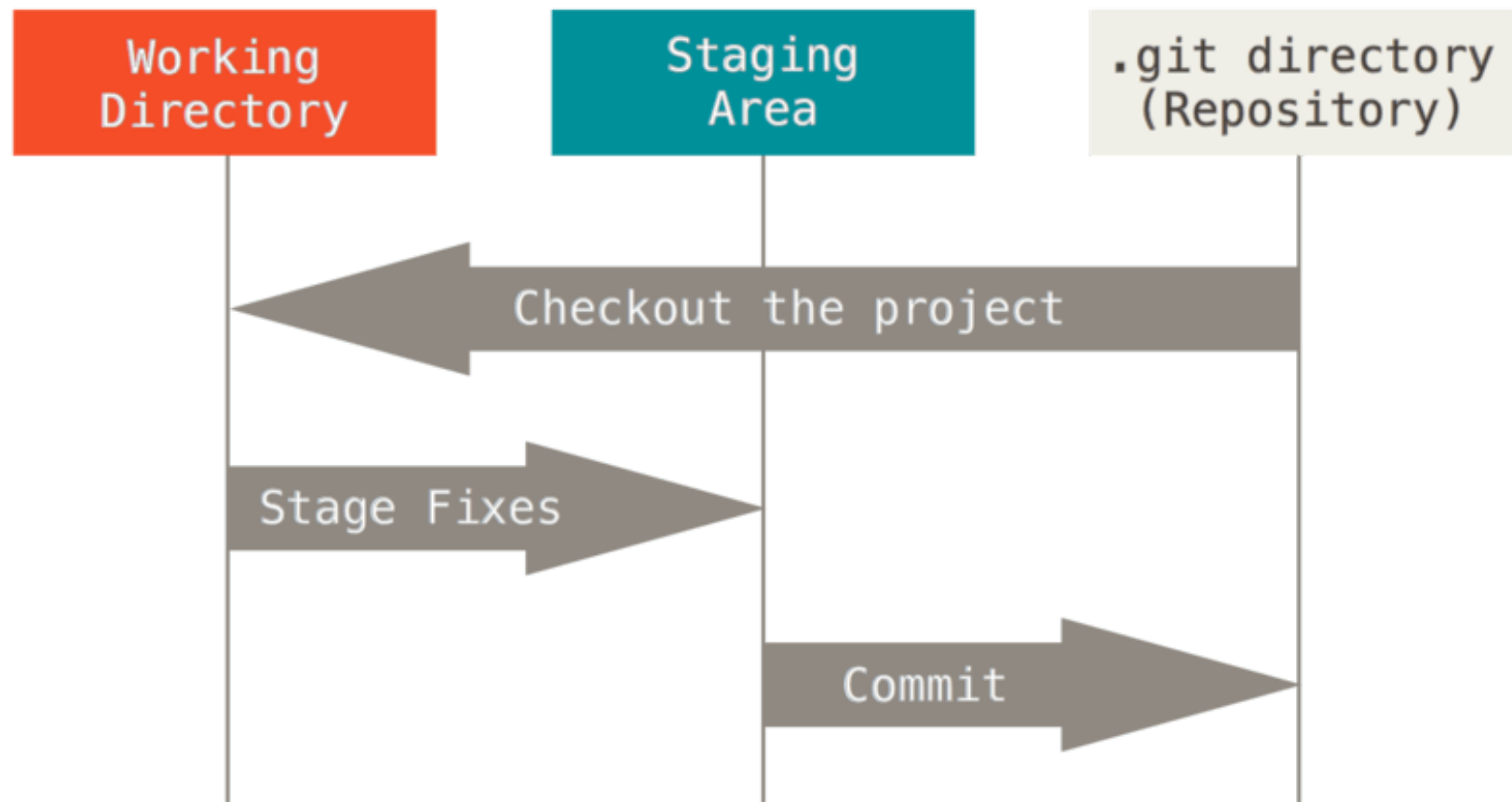


Intro to Git and Github

What is Git?

- Git is a method of version control. Version control is a snapshot of your folder at the stage you've 'saved' it. These 'snapshots' are stored in case you need to revert to an older version.
- Why do we need version control?
 - Works much better than saving multiple versions of a document when working alone
 - Helps save changes and prevent writing over other people's changes when working collaboratively

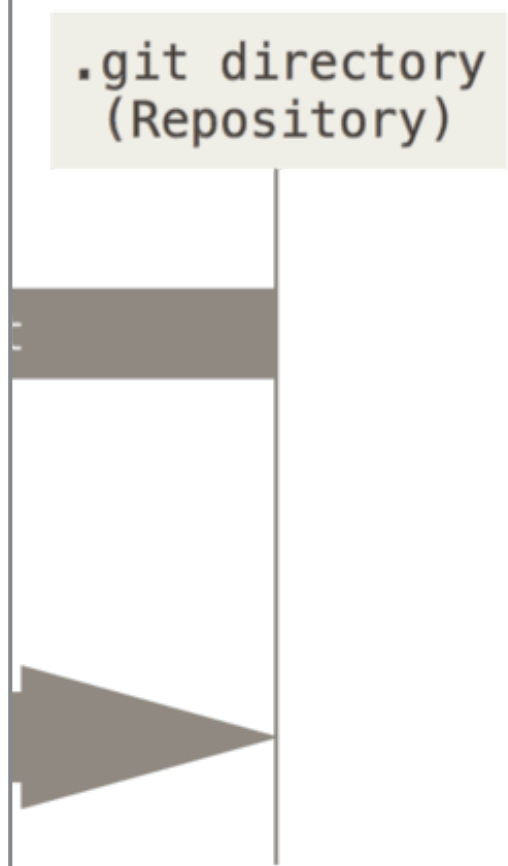
The parts of the Git flow



The parts of the Git flow

The Git directory is where Git stores the metadata and object database for your project.

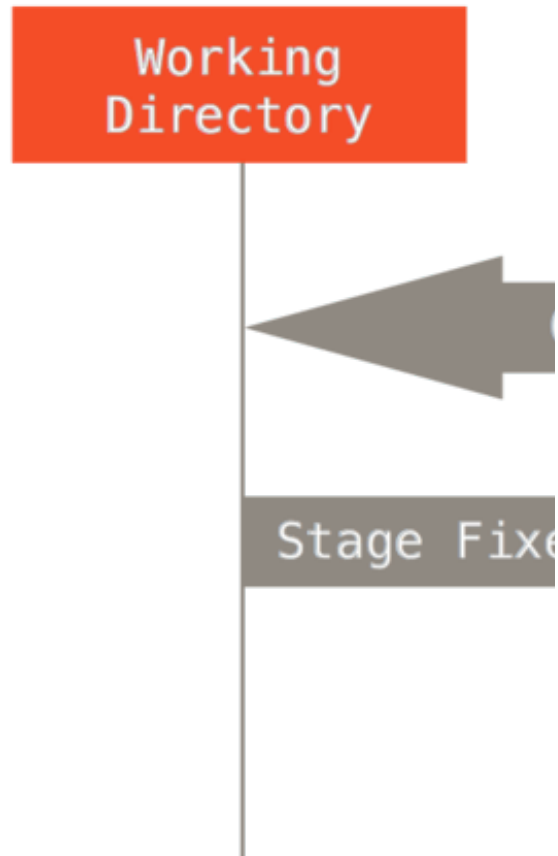
This is the most important part of Git, and it is what is copied when you clone a repository from another computer.



.git directory
(Repository)

The diagram illustrates the structure of the .git directory. It features a vertical line with a horizontal bar intersecting it. A label box is positioned at the top of the vertical line. Below the horizontal bar, a large arrow points from the left towards the vertical line.

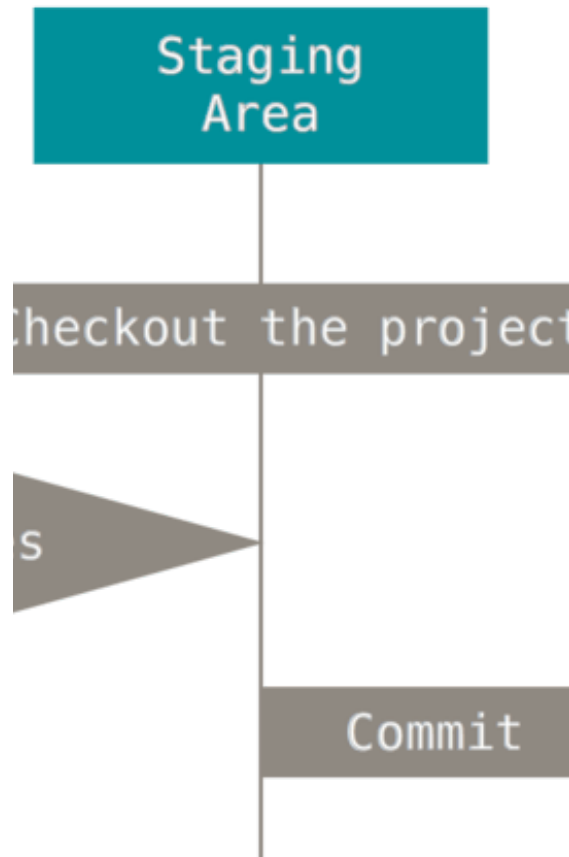
The parts of the Git flow



The working directory is a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.

The parts of the Git flow

The staging area is a file, generally contained in your Git directory, that stores information about what will go into your next commit. It's sometimes referred to as the "index".



What is a Git Workflow?

- You modify files in your working directory.
- You stage the files, adding snapshots of them to your staging area.
- You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

What's with that cat?



- Octocat is the mascot for GitHub.
- Git is a method of version control.
- GitHub is an website where you can store and share your GitHub repository (or repo). Github has an easy to use interface and offers collaboration and distribution tools, like 'forks', and presentation tools, like 'gists' and 'pages'.

What does it mean to 'fork' vs 'clone' a repo?

- A **clone** is simply a copy of a repository. No one but you sees any changes that are made to this copy, and it doesn't automatically refresh when the original version refreshes unless you tell it to.
- A **branch** is something that is within a repository. Conceptually, it represents a thread of development. Branches are usually temporary, and are intended to work on some small aspect of your main project. Any commits you make will stay on that branch until you merge them with the main branch. Branches can only be made by people authorized in the repository.
- A **fork** is a more public way of making a branch. Forking allows anybody make a local copy in their own git repository account. They can then make changes and when finished send a request to get the code accepted.

Great StackOverflow resource [here](#).

I broke it. How do I undo a commit?

- One big benefit of using git is so that you can revert to old versions of your work, if necessary.
- There are three types of 'reversions' - one in which you completely delete your change ('hard'), another in which you revise your commit before re-committing, and another in which you leave your files and index ('soft')
 - Type 1: `git reset --hard HEAD^`
Type 2: `git reset HEAD^`
Type 3: `git reset --soft HEAD^`
- You can also choose how far back you'd like to go in your commit log:
`git reset --soft HEAD~1` (~n refers to the nth previous commit)