# Get Acquainted with Git

treehouse™


http://teamtreehouse.com/

# CSS

Cascading Style Sheets (CSS) forms the presentational layer of web pages. CSS allows you to apply visual styling to HTML elements with colors, fonts, layouts, and more.

Course
**Creating a Compass Extension**
Intermediate
• CSS

Course
**Compass Basics**
Intermediate
• CSS

Complete
**Sass Basics**
Intermediate
354
Points achieved


Course
**Modular CSS with Sass**
Intermediate
August 2014
• CSS


Search
Topics
All Topics
HTML
CSS
Design
JavaScript
Ruby
PHP
WordPress
iOS
Android
Development Tools
Business
Python
Content Type

# What is Git?

treehouse™

**version control system (vcs)**

System to keep track of changes to files.

Let's you undo mistakes, work on different versions of a project, and work with a team of people without stepping on each other's code.

treehouse™

# The Setup

treehouse™

# Installing Git

- **Mac**
  It's installed with Xcode or:
  http://git-scm.com/download/mac

- **Windows**
  http://git-scm.com/download/win

treehouse™

# Configuration

```
git config --global color.ui true
git config --global user.name <your_name>
git config --global user.email <your_email>
```

**Repository** or **“repo”**

The folder containing the files you are tracking — the folder for your project, for example.

treehouse™

# Create a repository

```
cd ~/Documents/my_project
git init
```

## Delete a repository

All git information is stored in a .git folder in the project's root. Just delete the .git folder.

treehouse™

**.gitignore**

A file which lists files and file extensions that Git should not track in the repository. “Ignore these files.” Place in the repository folder.

https://github.com/treehouse-dave/gitignore/archive/master.zip

treehouse™

# .gitignore (very minimum for a Mac)

```
.DS_Store
```

# The Basics

treehouse™

**commits**

Keep track of your changes by “committing” changed files to the repository.

treehouse™

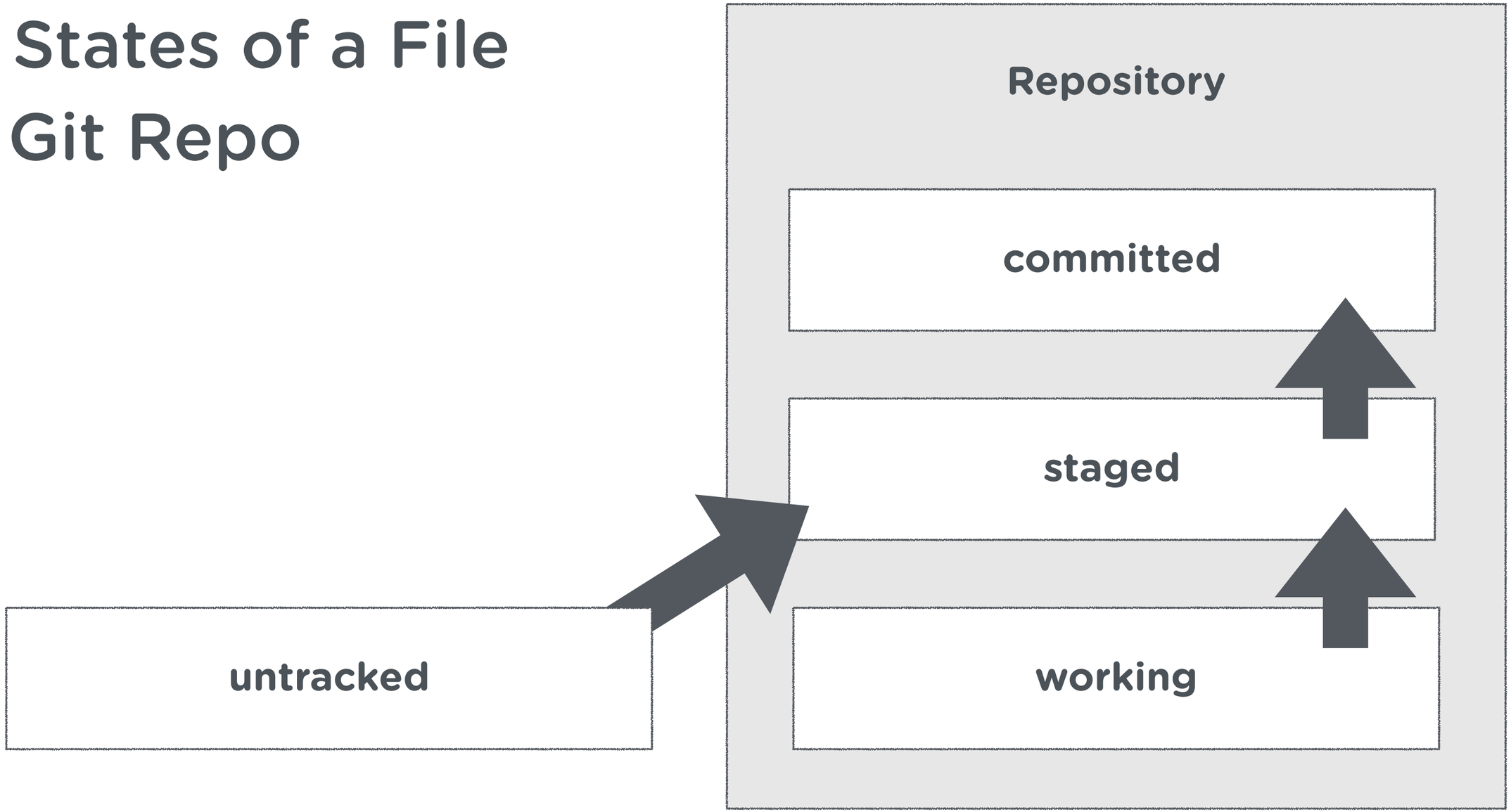
commit early, commit often

treehouse™

# Your First Commit

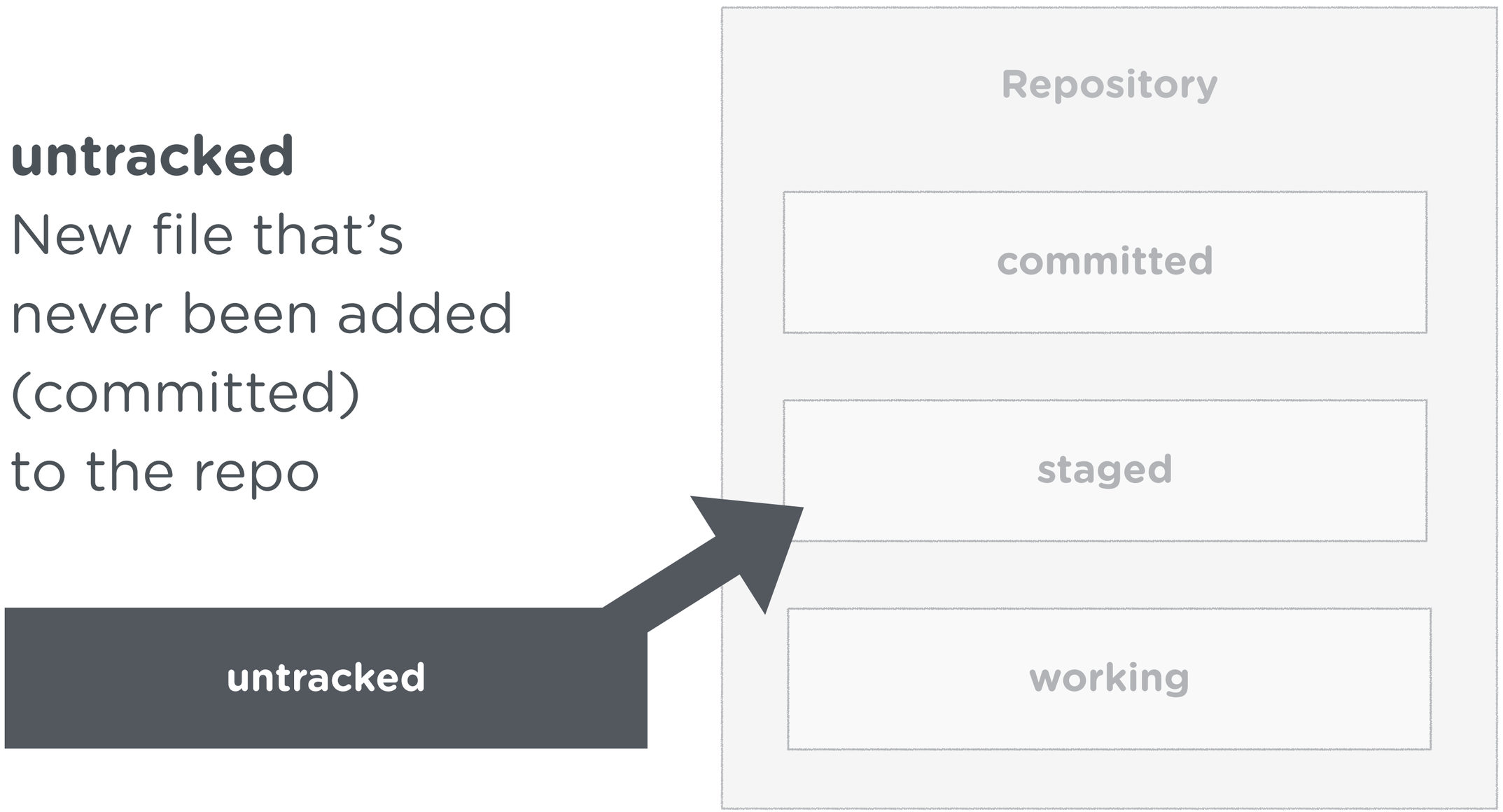
```
git add .
git commit -m "Initial Commit"
```

# The States of a File in a Git Repo



**untracked**

New file that's never been added (committed) to the repo



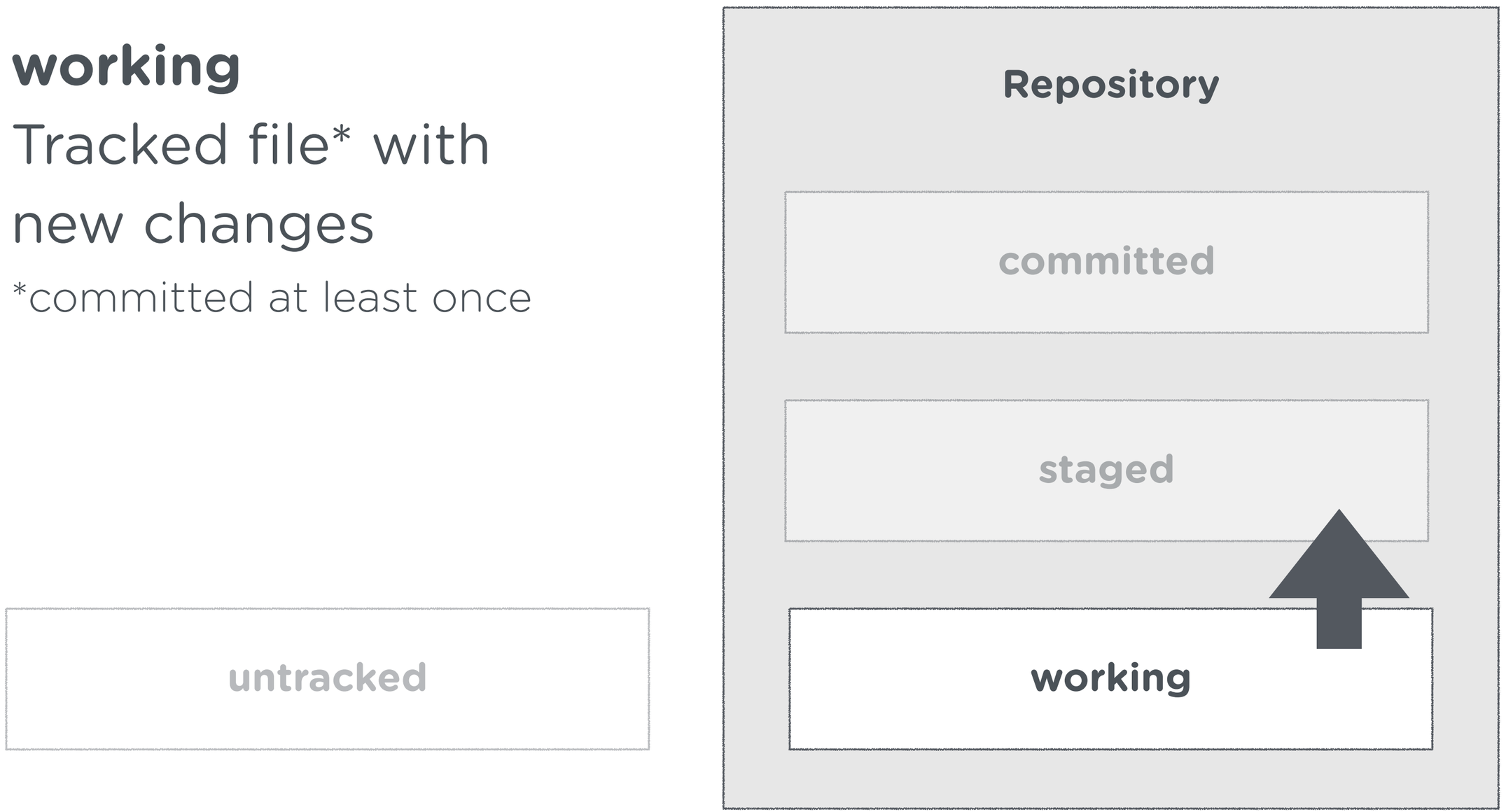
# What is Happening in the Repo?

```
git status
```

**working**

Tracked file* with new changes

*committed at least once

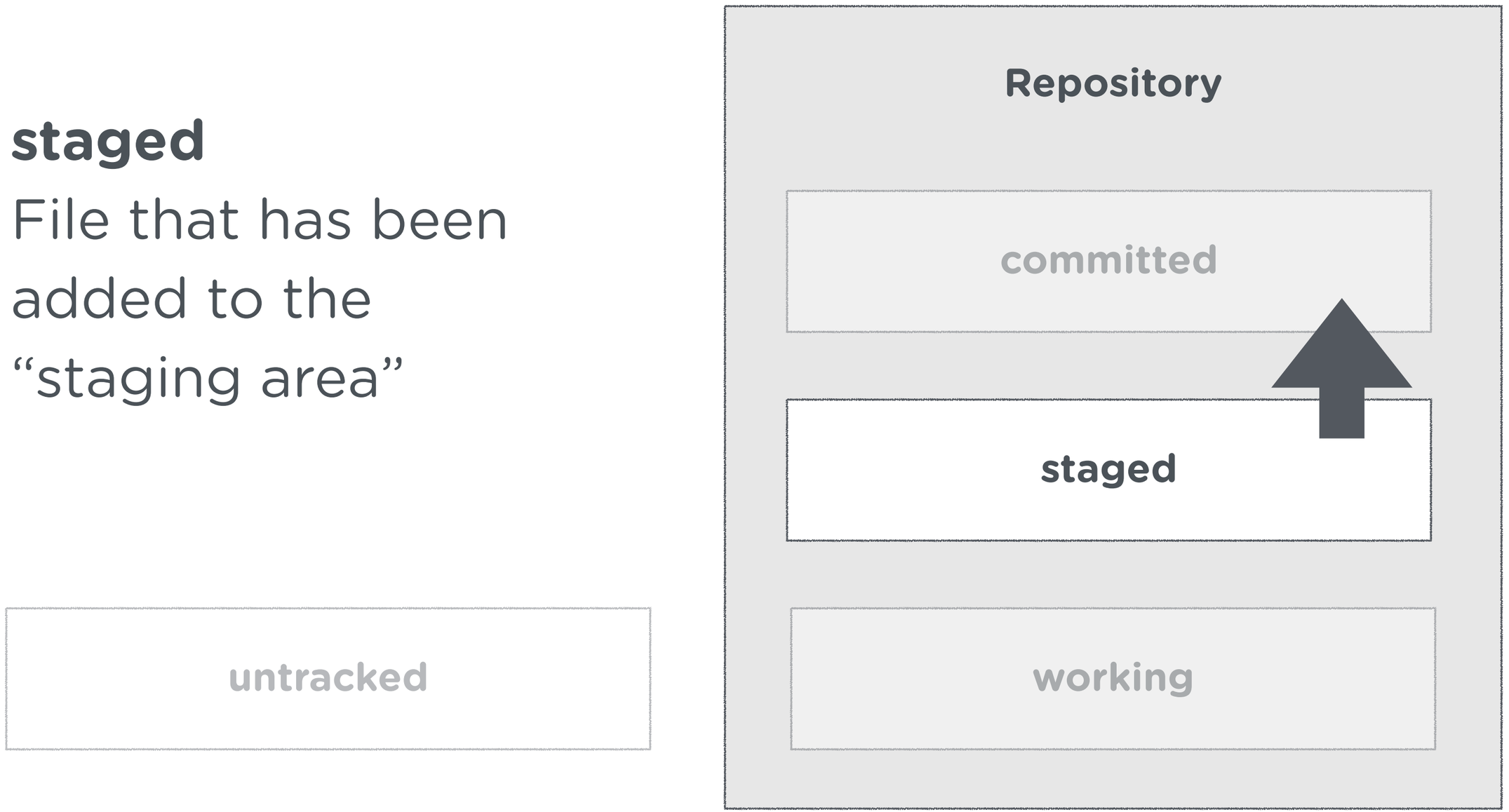


# Add Files

```
git add .
```

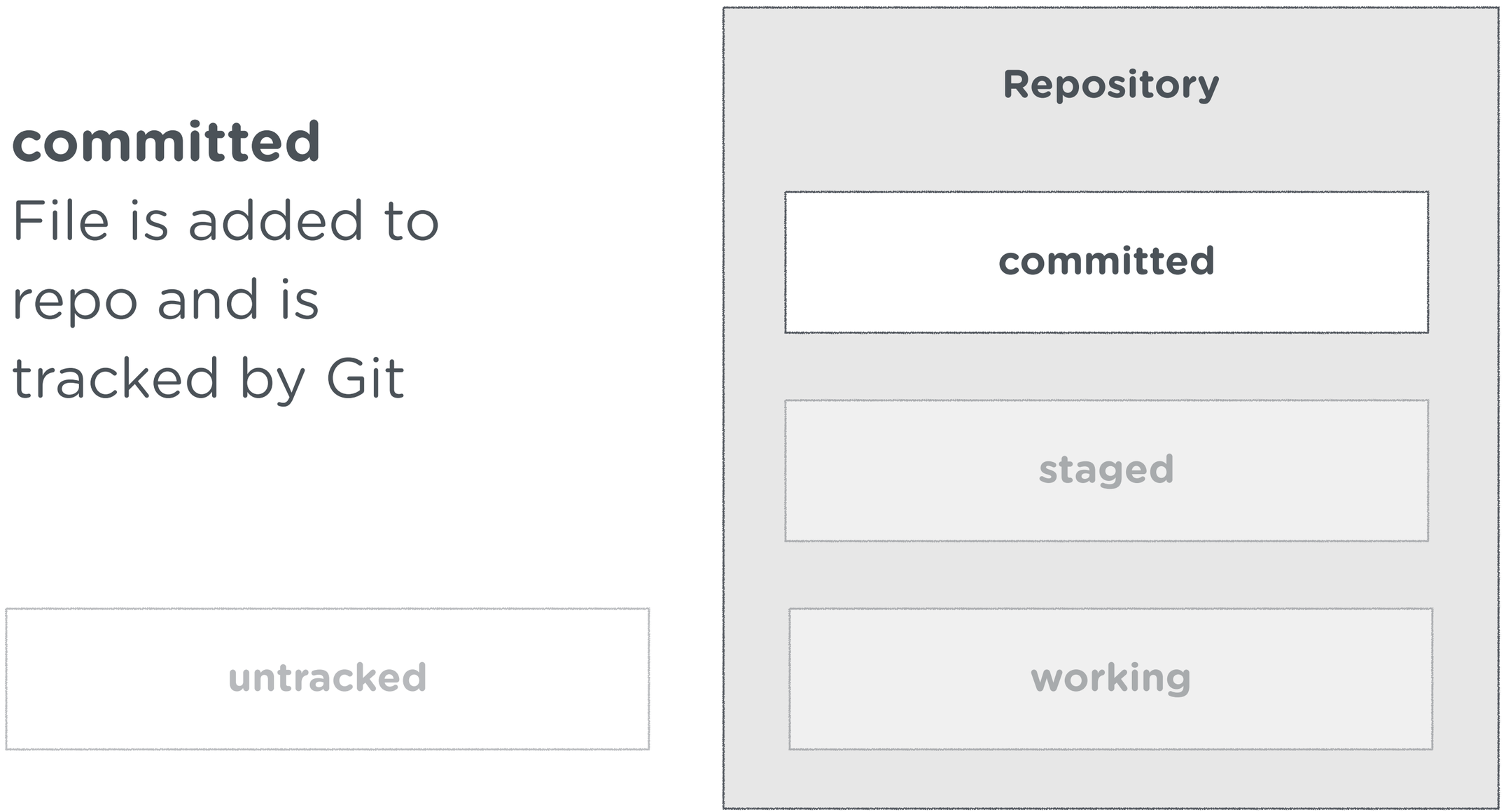
**staged**

File that has been added to the "staging area"



**committed**

File is added to repo and is tracked by Git

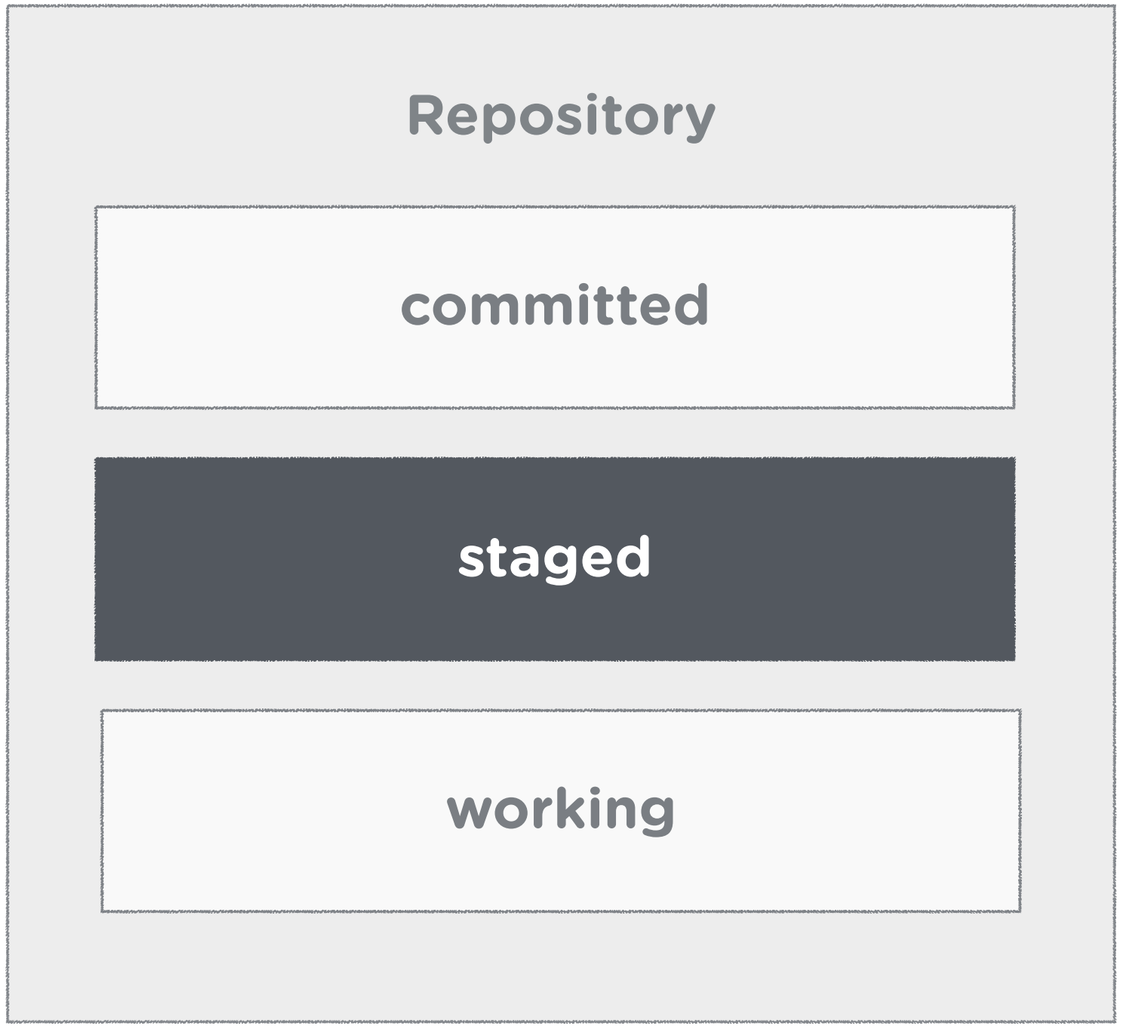


# Commit

```
git commit -m "Add JavaScript"
```

**staging**

Collect related files
to commit together




treehouse™

# Let’s Check Our Commits

```
git log
git log --oneline
```

# Check What You've Changed

```
git diff
git diff -- <file/directory>
```

# Keep Changes Separate

treehouse™

master

treehouse™

**master**

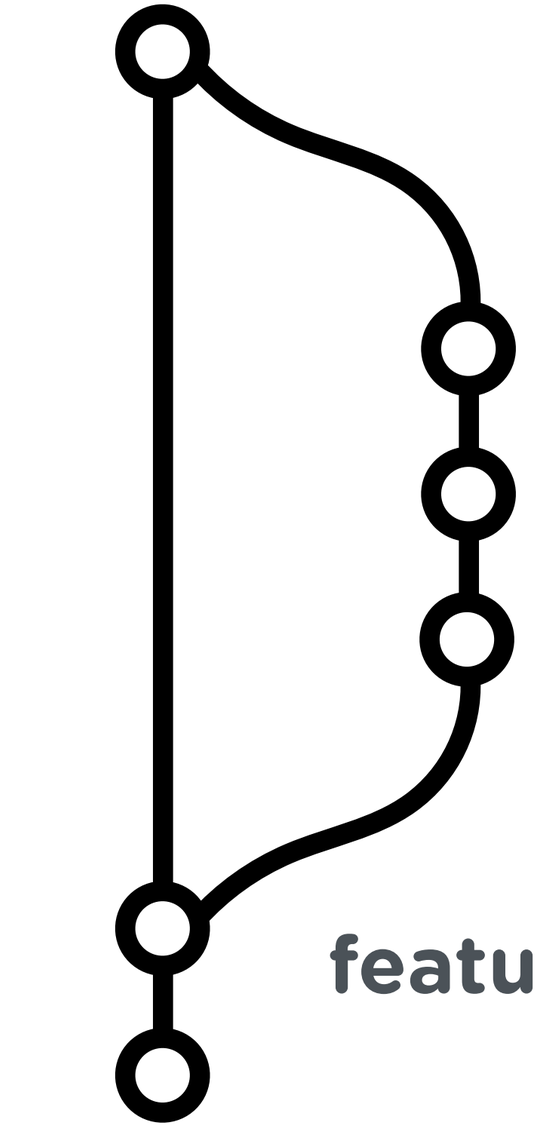
The main branch for a repository

treehouse™

**branch**

Separate path for development, bug fixes and adding features

treehouse™

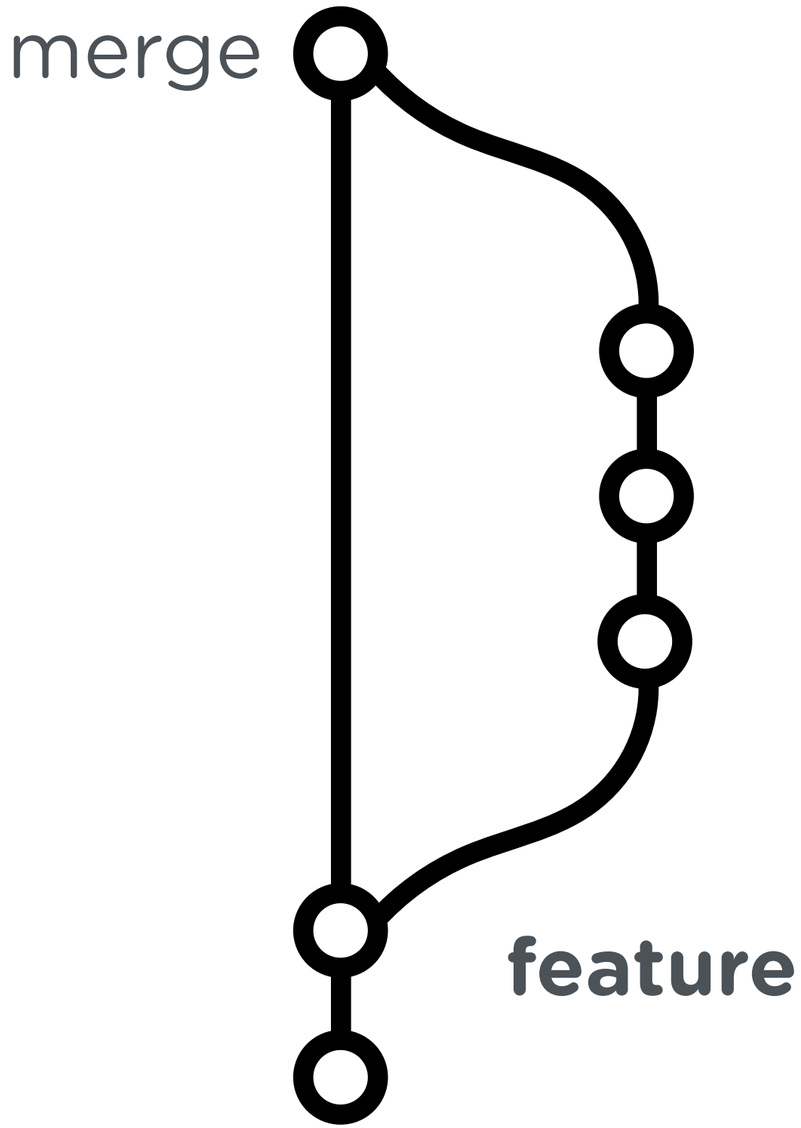



treehouse™

# A Simple Workflow

1. Create and checkout a new working branch
2. Work in that branch. Making changes; adding features
3. Once everything is working, merge changes back into master branch
4. Delete feature branch
5. Deploy master (push to web server)

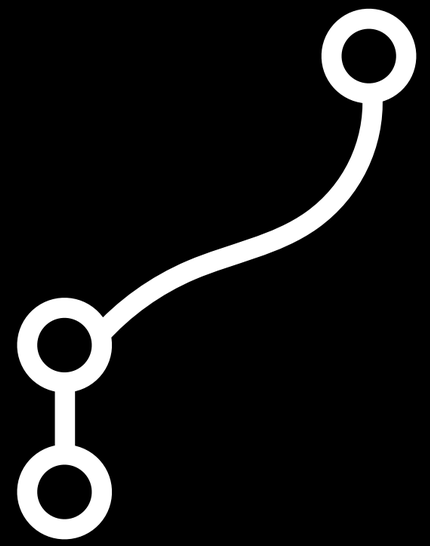
treehouse™




treehouse™

# Create and Checkout New Branch

```
git checkout -b <working_branch_name>
```



# What Branches Do I Have?

```
git branch
```

# When things go wrong

treehouse™

# I Messed Up My Working File (not yet staged)

```
git checkout <path/to/file>
```

# I Messed Up A File and Staged It

```
git reset HEAD <path/to/file>
git checkout <path/to/file>
```

# Arrgggg. I Messed Up File and Committed It!

```
git checkout HEAD^ <path/to/file>
```

# Oh man. I really messed up. Undo commit.*

```
git reset --hard HEAD^
```

**warning!** this changes a repo's history. Don't do this if you've published this commit to a shared repo (like Github)

# Merging Changes

treehouse™

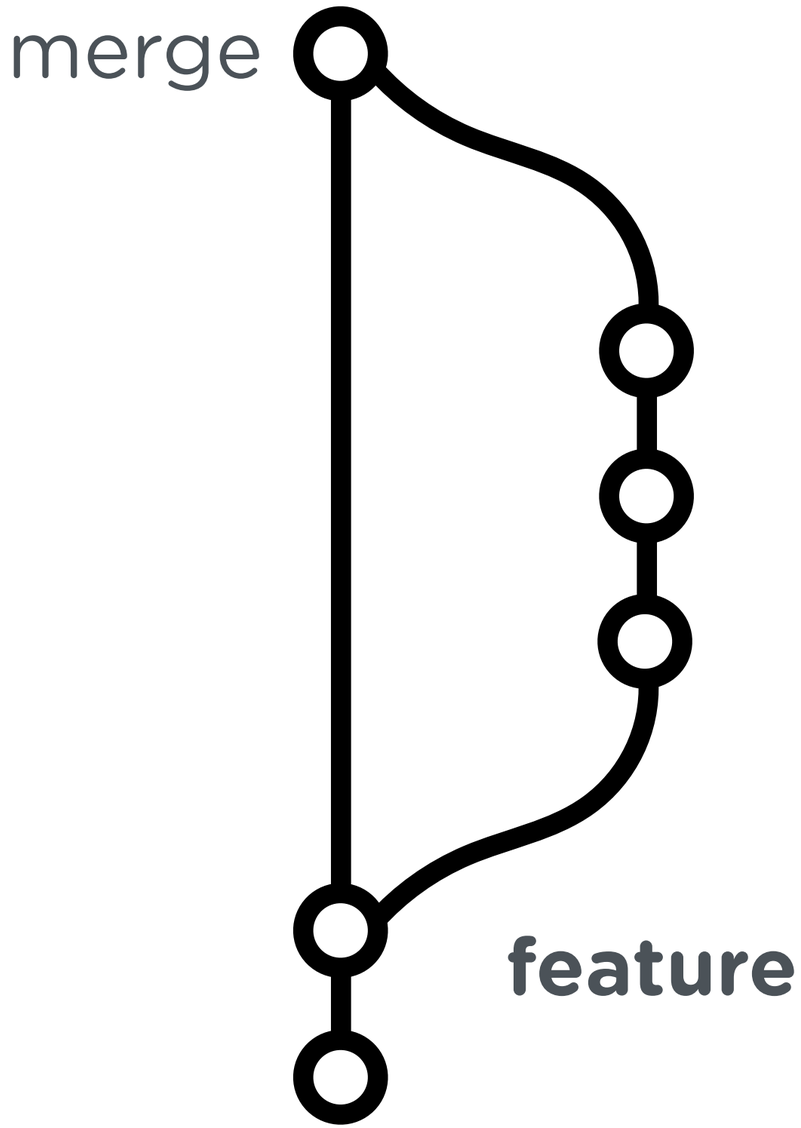
# A Simple Workflow

1. Create and checkout a new working branch

2. Work in that branch. Making changes; adding features

3. Once everything is working, merge changes back into master branch

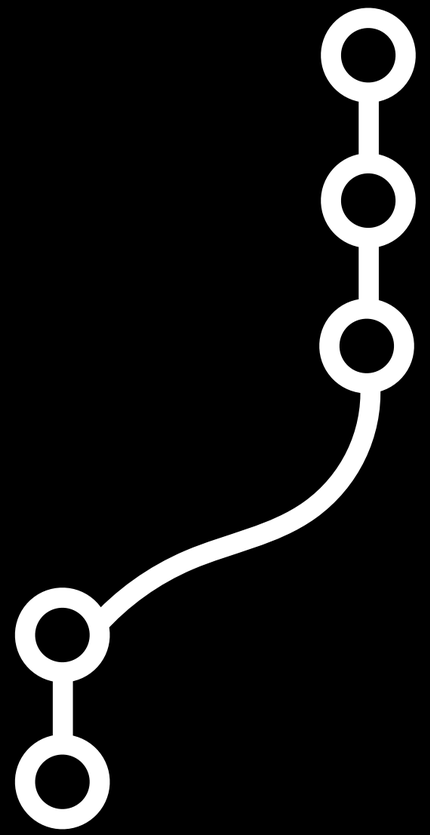
treehouse™




treehouse™

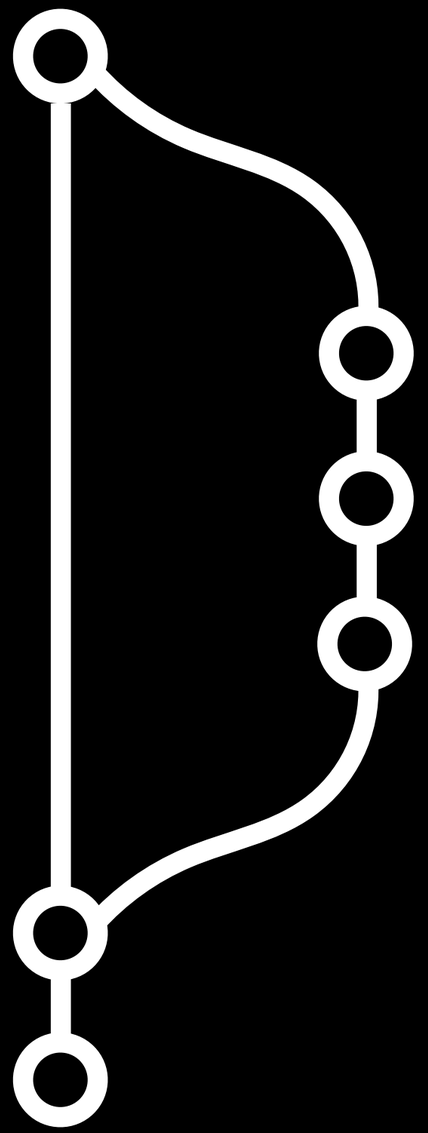
# Return to Master

```
git checkout master
```



# Merge Changes

```
git checkout master
git merge <working_branch_name>
```



# A Simple Workflow

1. Create and checkout a new working branch

2. Work in that branch. Making changes; adding features

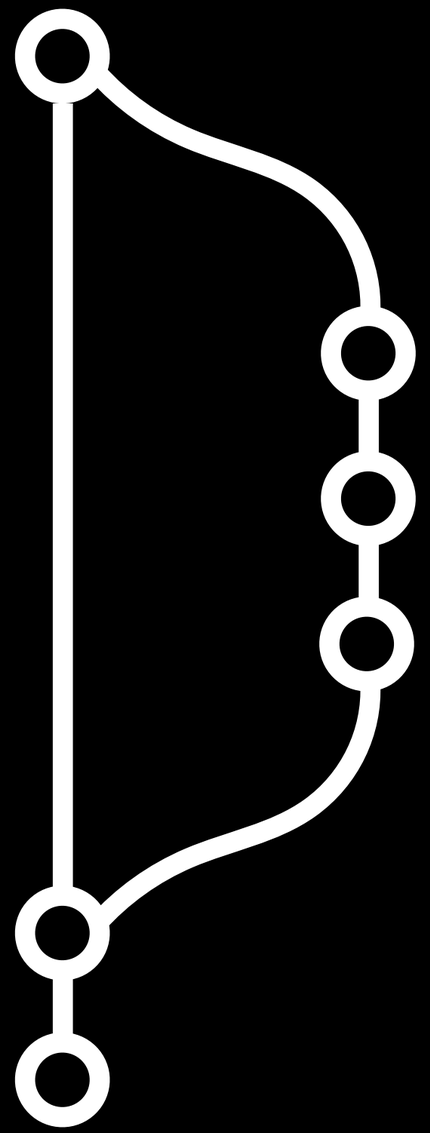
3. Once everything is working, merge changes back into master branch

4. Delete feature branch

treehouse™

# Delete Old Branch

```
git checkout master
git merge <working_branch_name>
git -d <working_branch_name>
```



# A Simple Workflow

1. Create and checkout a new working branch
2. Work in that branch. Making changes; adding features
3. Once everything is working, merge changes back into master branch
4. Delete feature branch
5. Deploy master (push to web server)

treehouse™

https://github.com/treehouse-dave/get-acquainted-with-git

treehouse™