



It's Time To Learn

Git Basics



A Quick High-Level

Conceptual Overview

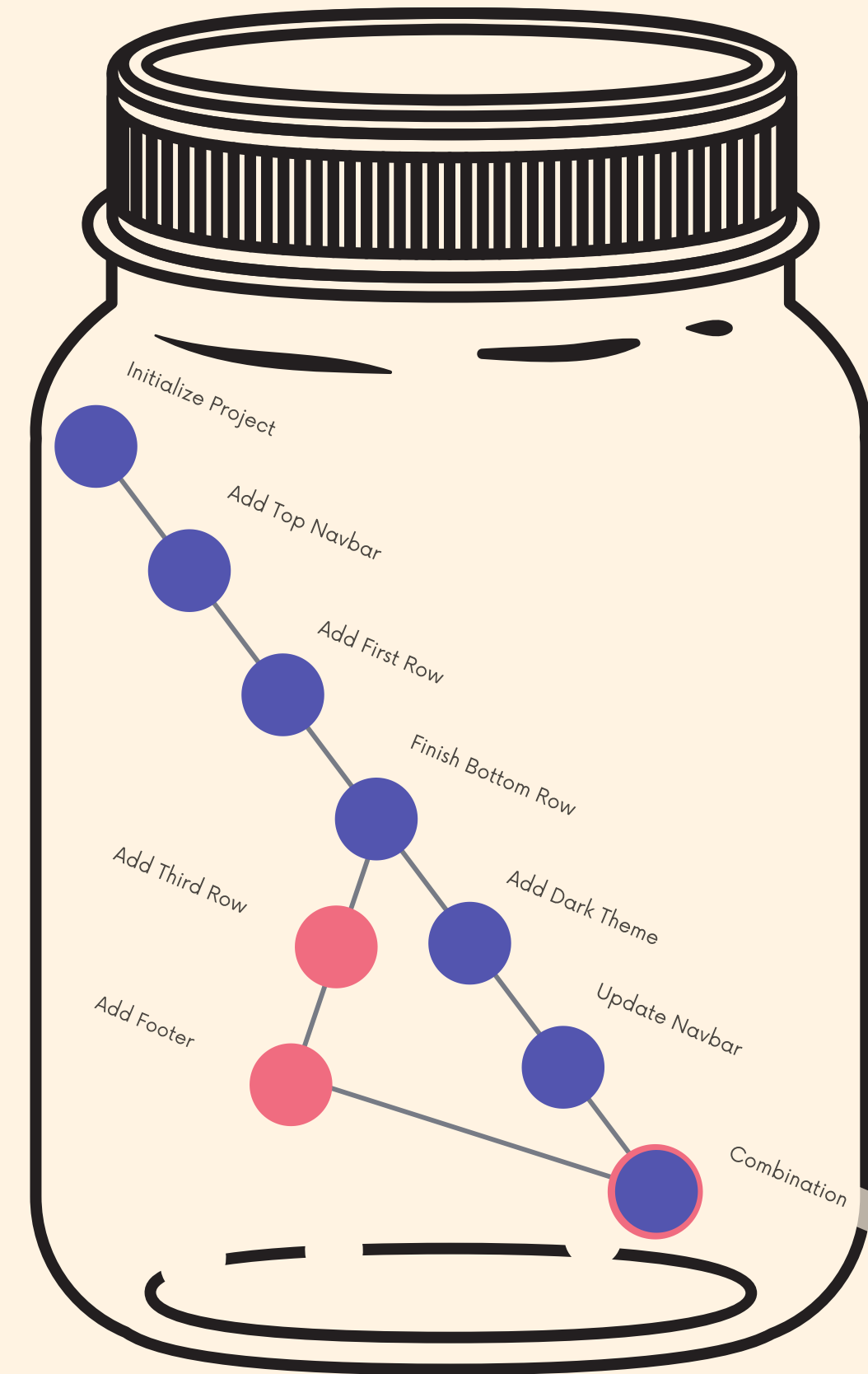




Repository

A Git "Repo" is a workspace which tracks and manages files within a folder.

Anytime we want to use Git with a project, app, etc we need to create a new git repository. We can have as many repos on our machine as needed, all with separate histories and contents













Portfolio Website

Startup Idea

My First Movie Script

Symphony #17

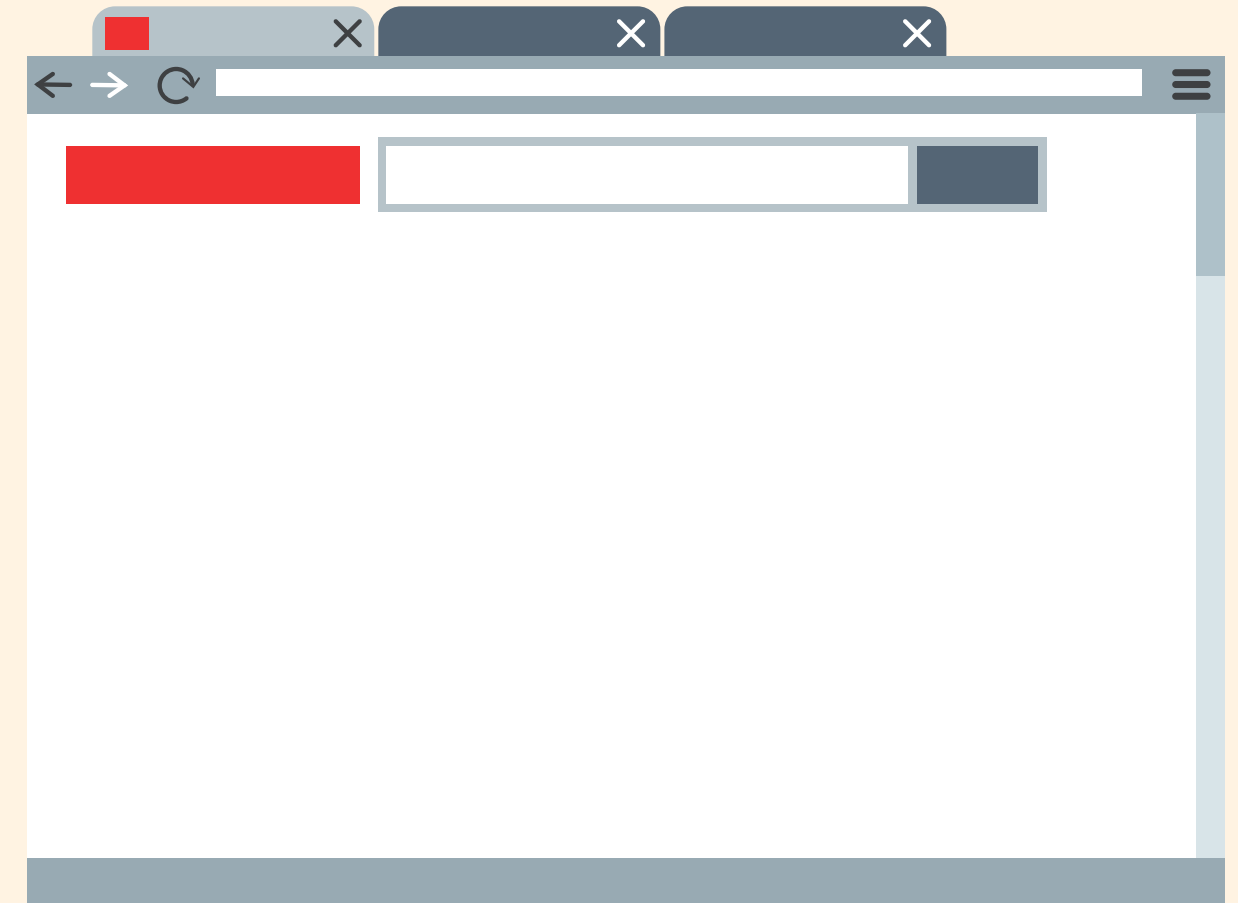
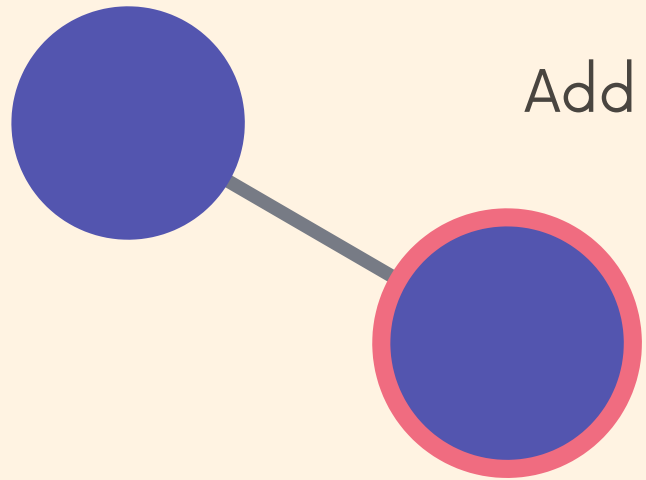
Reddit Clone App



This is my repo!

Initialize Project

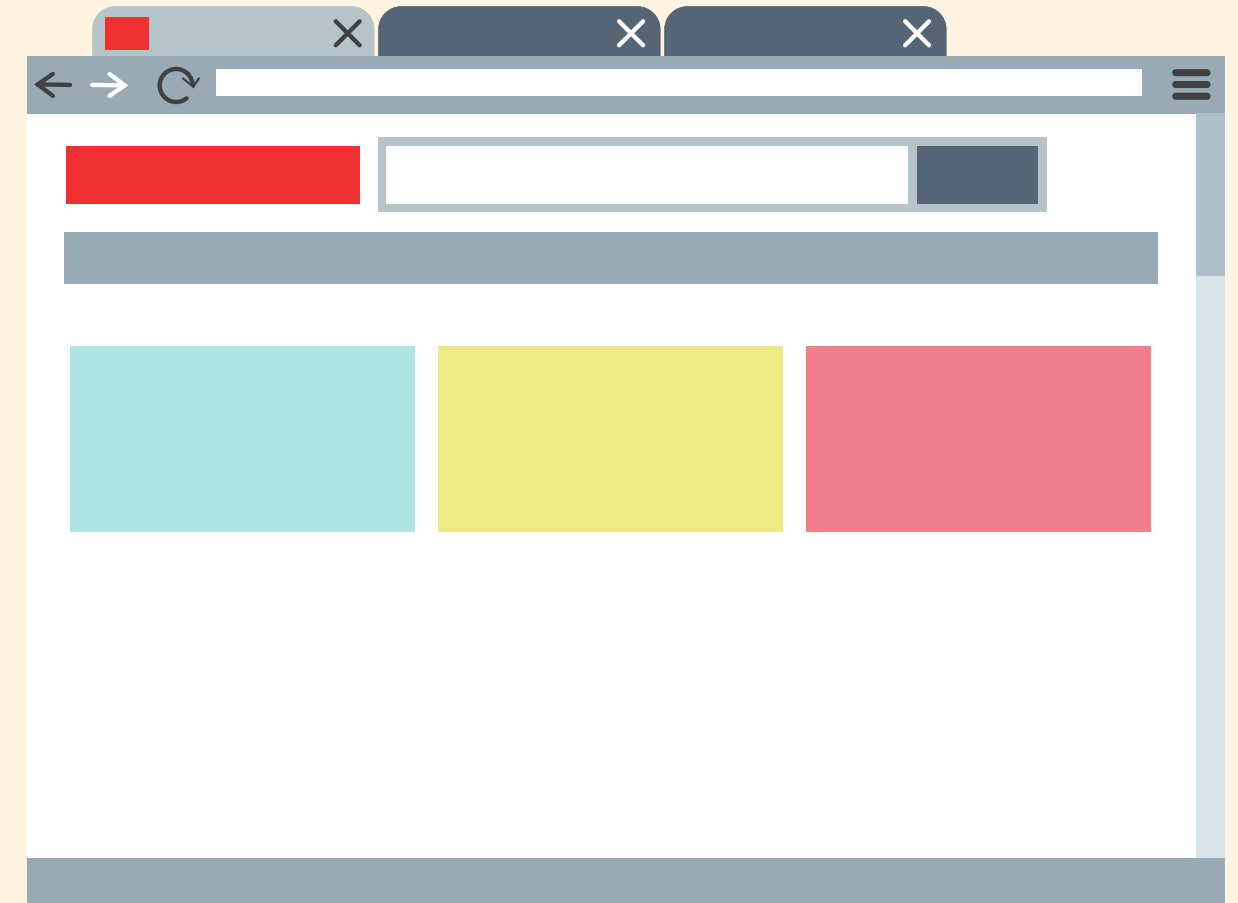
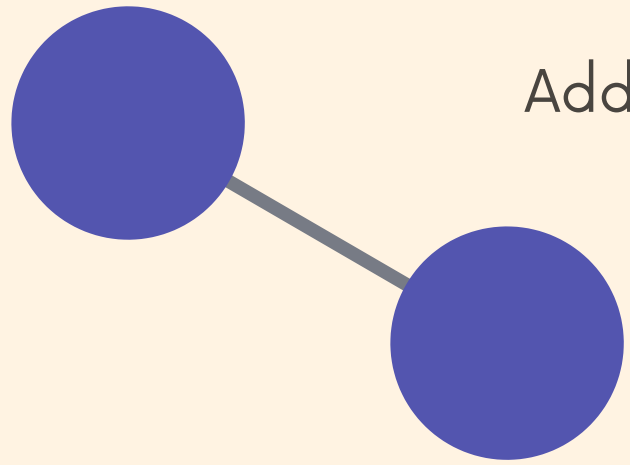
Add Top Navbar



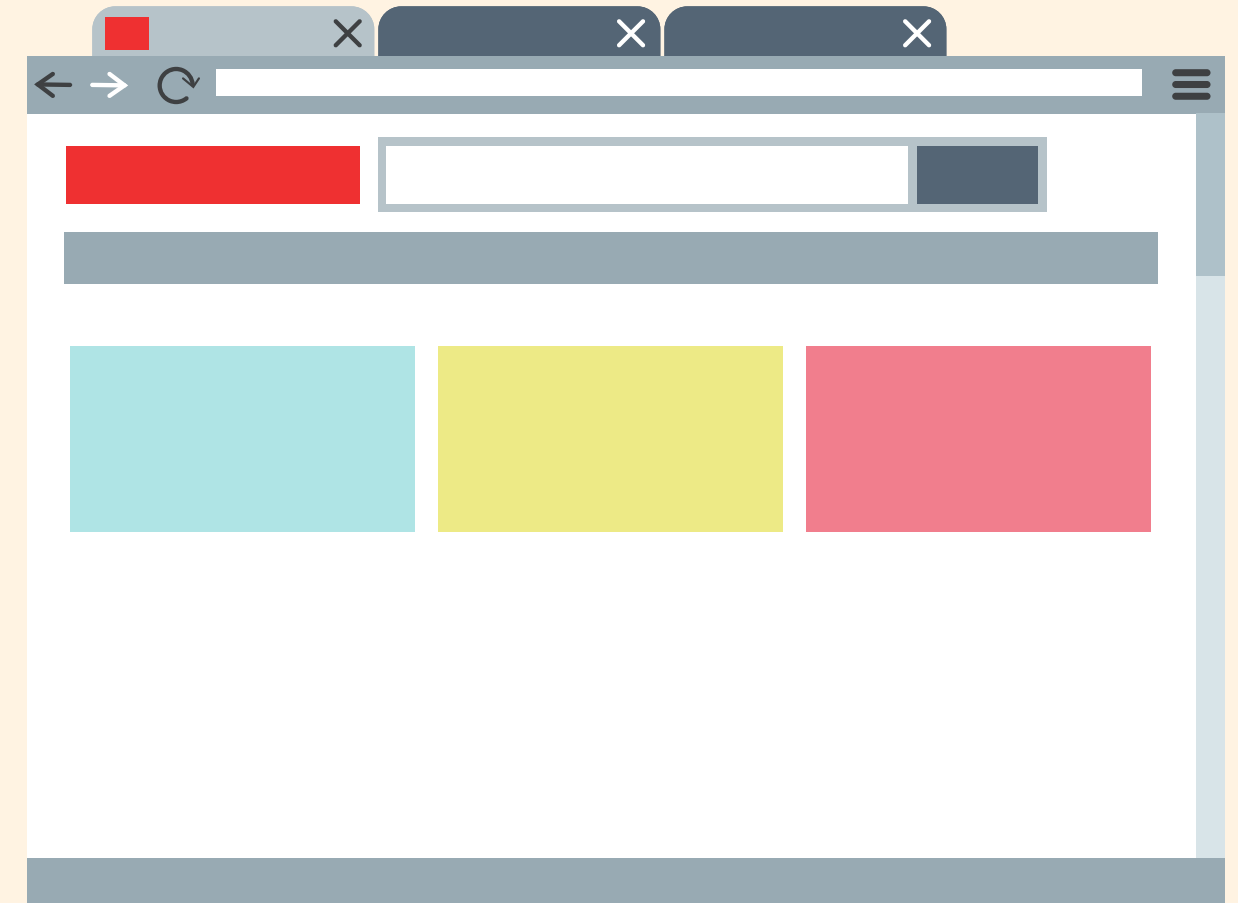
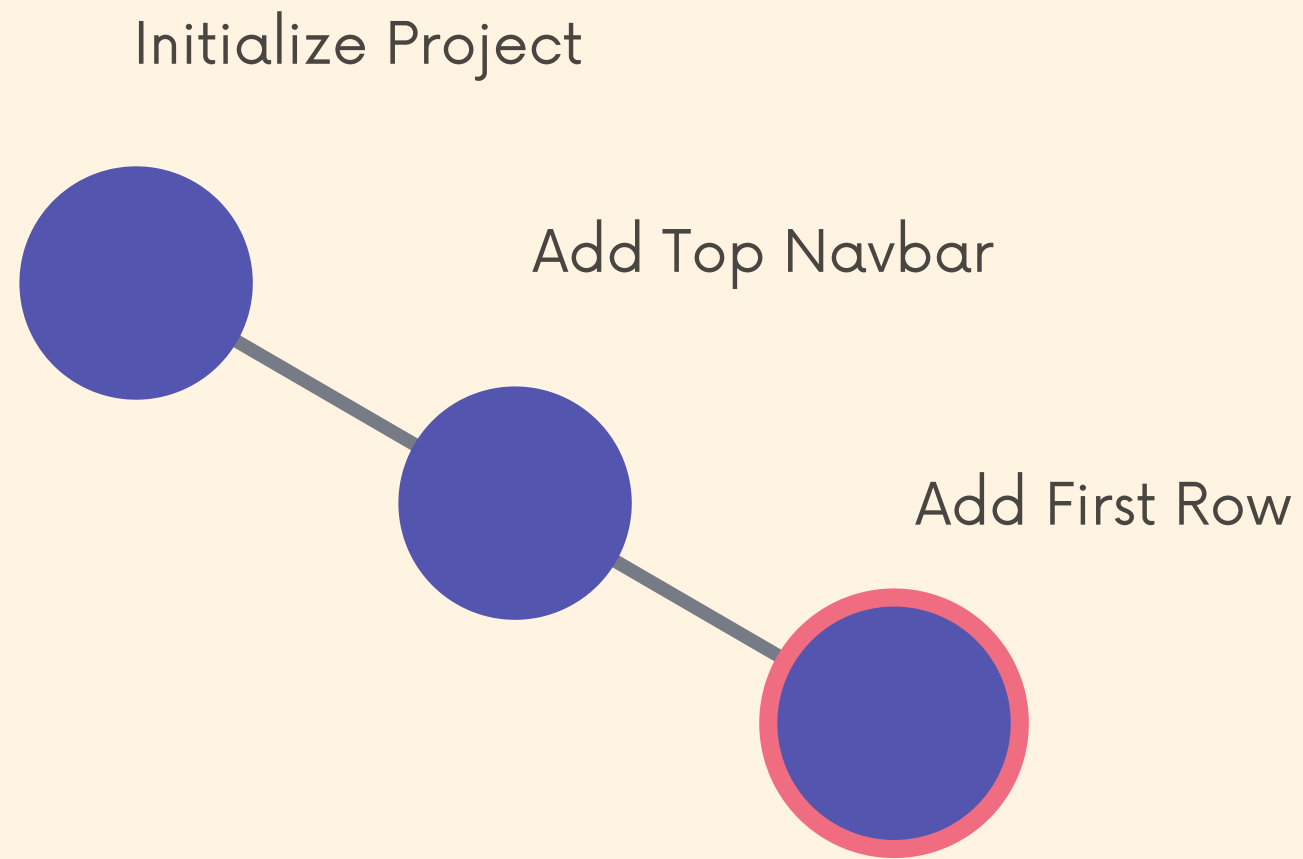
I add some content

Initialize Project

Add Top Navbar



Add A Commit





Committing

Making a commit is similar to making a save in a video game. We're taking a snapshot of a git repository in time.

When saving a file, we are saving the state of a single file. With Git, we can save the state of multiple files and folders together.



Save



Commit





The Basic Git Workflow

Work On Stuff

Make new files, edit
files, delete files, etc

Add Changes

Group specific
changes together, in
preparation of
committing

Commit

Commit everything
that was previously
added





Our First Git Command!

git status gives information on the current status of a git repository and its contents

It's very useful, but at the moment we don't actually have any repos to check the status of!





If We Try It...



```
> git status
```

```
fatal: not a git repository  
(or any of the parent directories)
```





Our Actual First Git Command!

Use **git init** to create a new git repository. Before we can do anything git-related, we must initialize a repo first!

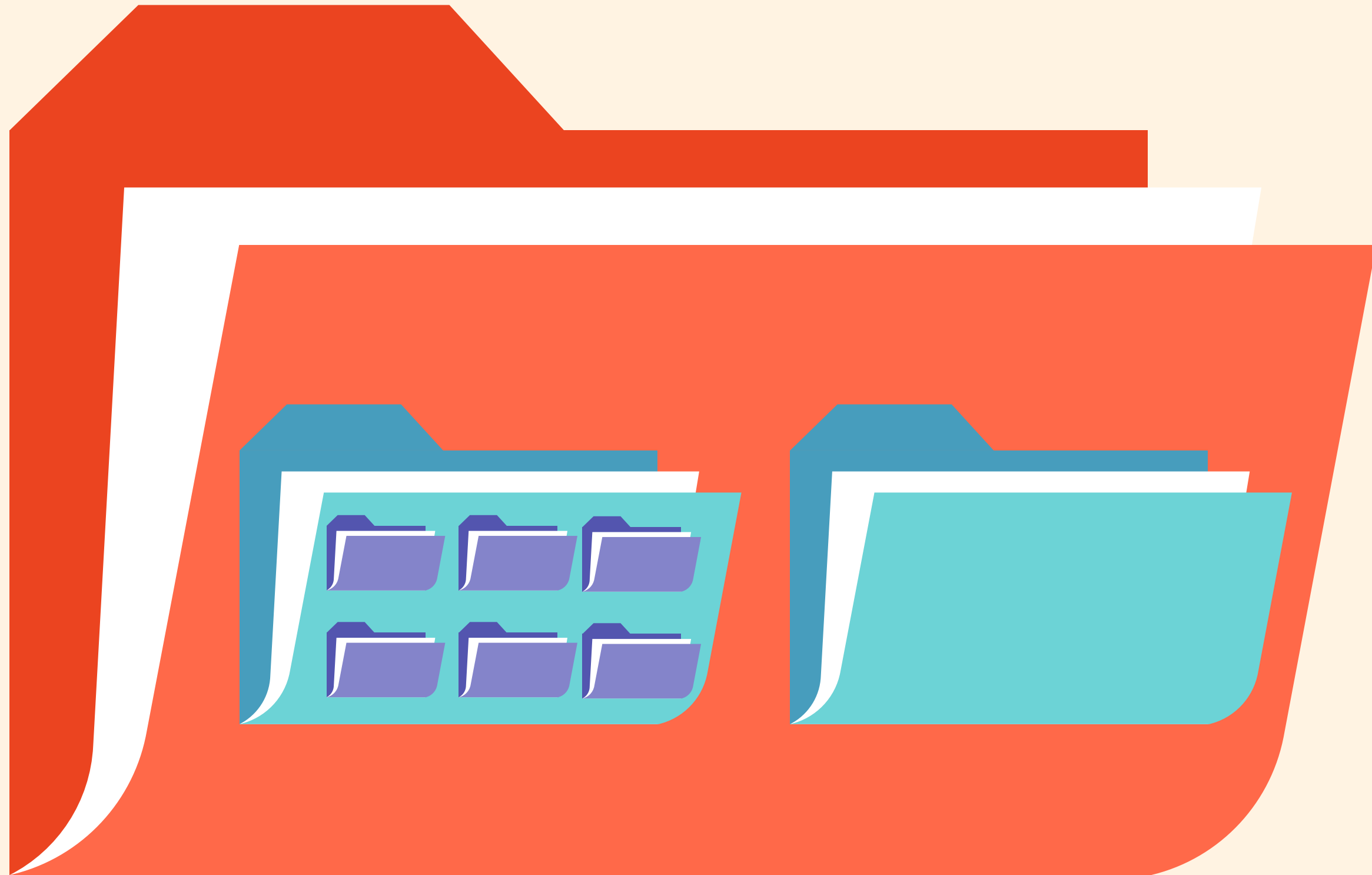
This is something you do once per project. Initialize the repo in the top-level folder containing your project



```
> git init
```



Git Tracks A Directory and All Nested Subdirectories



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.



Git Is Now Watching Us!

Try making a new file in your repo, and then run `git status` again.

You'll see that git noticed the file, but it is "untracked"





Committing

Making a commit is similar to making a save in a video game. We're taking a snapshot of a git repository in time.

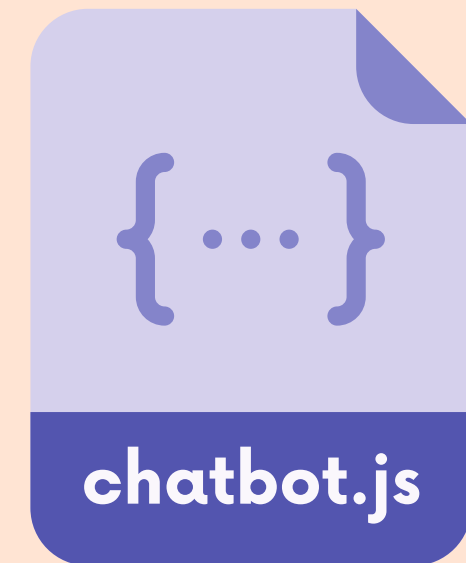
When saving a file, we are saving the state of a single file. With Git, we can save the state of multiple files and folders together.



Save



Commit





deleted team.html

modified about.html

modified about.css

created navbar.html

created navbar.css

created navbar.js

created logo.jpg

I made changes in 7
different files this morning

deleted team.html

modified about.html

modified about.css

Add new team members
to about page

created navbar.html

created navbar.css

created navbar.js

created logo.jpg

deleted team.html

modified about.html

modified about.css

created navbar.html

created navbar.css

created navbar.js

created logo.jpg

Add branded navbar



The Basic Git Workflow

Work On Stuff

Make new files, edit files, delete files, etc

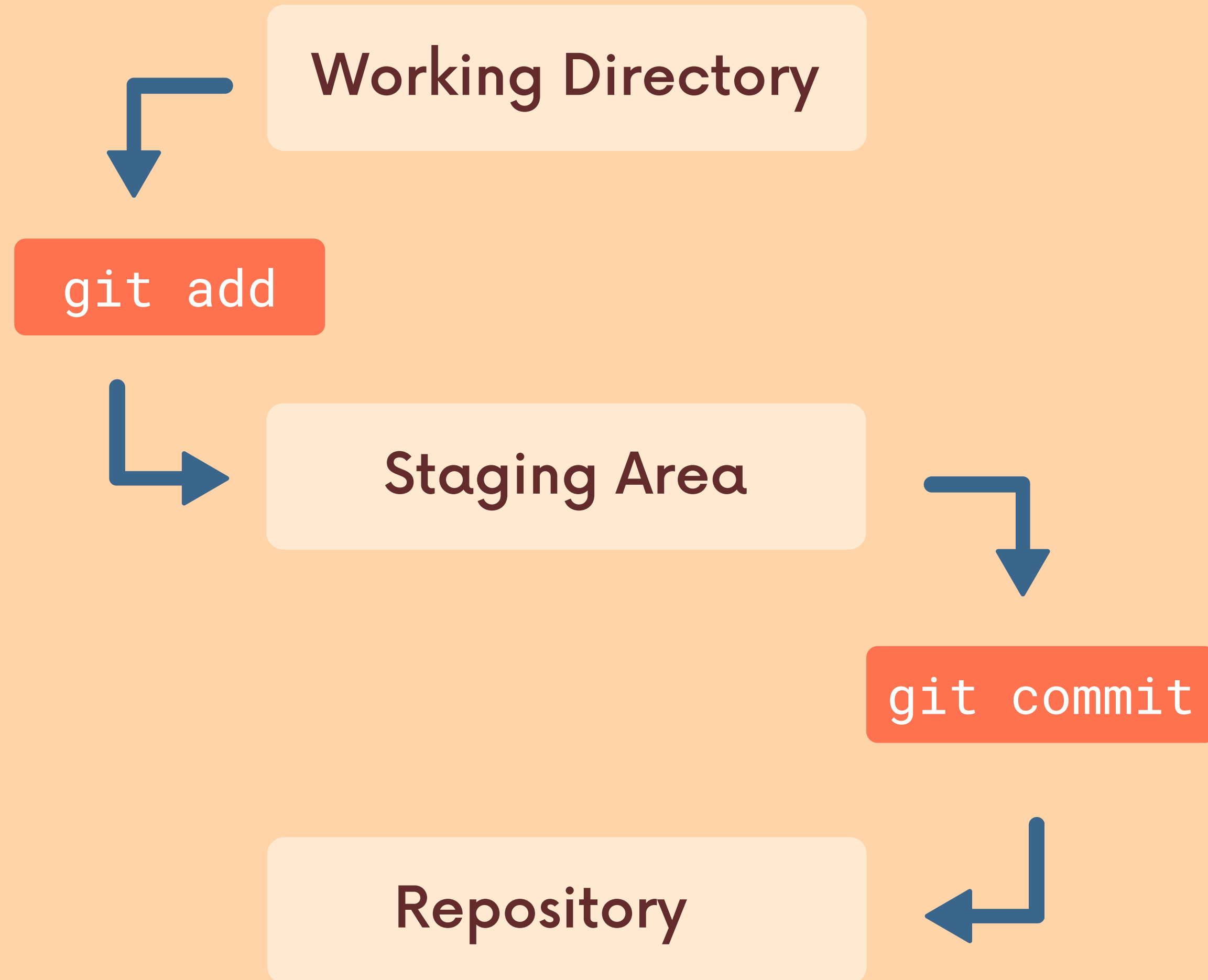
Add Changes

Group specific changes together, in preparation of committing

Commit

Commit everything that was previously added







Adding

We use the `git add` command to stage changes to be committed.

It's a way of telling Git, "please include this change in our next commit"



Working Directory

deleted team.html

modified about.html

modified about.css

created navbar.html

created navbar.css

created navbar.js

created logo.jpg

Staging Area

Repository

Working Directory

deleted team.html

modified about.html

modified about.css

created navbar.css

created navbar.js

created logo.jpg

Staging Area

created navbar.html

Repository

Working Directory

deleted team.html

modified about.html

modified about.css

created navbar.js

created logo.jpg

Staging Area

created navbar.html

created navbar.css

Repository

Working Directory

deleted team.html

modified about.html

modified about.css

created logo.jpg

Staging Area

created navbar.html

created navbar.css

created navbar.js

Repository

Working Directory

deleted team.html

modified about.html

modified about.css

Staging Area

created navbar.html

created navbar.css

created navbar.js

created logo.jpg

Repository

Working Directory

deleted team.html

modified about.html

modified about.css

Staging Area

Repository

created navbar.html

created navbar.css

created navbar.js

created logo.jpg

Add branded navbar

Working Directory

deleted team.html

modified about.html

modified about.css

Staging Area

Repository

Add branded navbar

Working Directory

modified about.html

modified about.css

Staging Area

deleted team.html

Repository

Add branded navbar

Working Directory

modified about.css

Staging Area

deleted team.html

modified about.html

Repository

Add branded navbar

Working Directory

Staging Area

deleted team.html

modified about.html

modified about.css

Repository

Add branded navbar

Working Directory

Staging Area

Repository

deleted team.html

modified about.html

modified about.css

Add team members
to about page

Add branded navbar

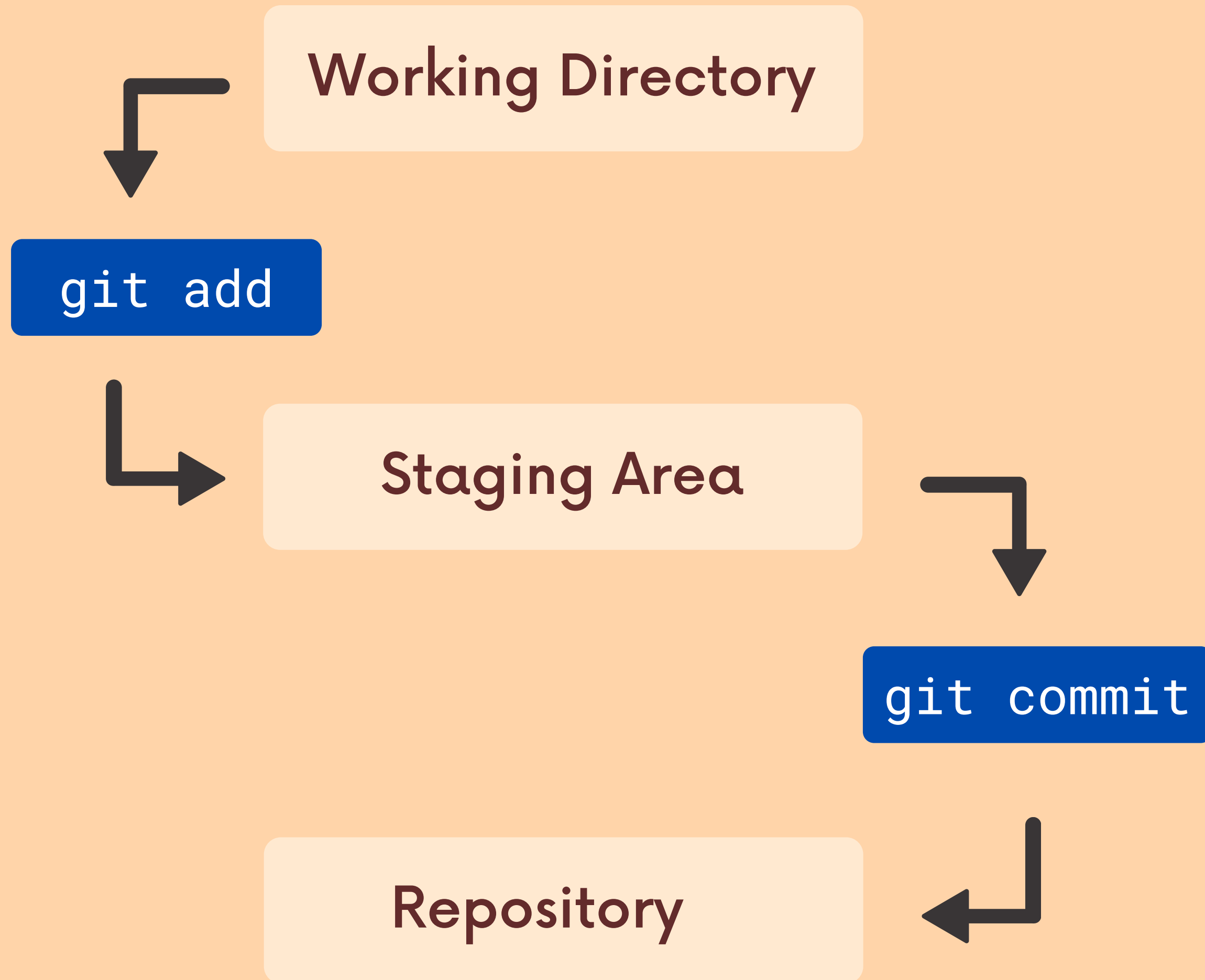
Working Directory

Staging Area

Repository

Add team members
to about page

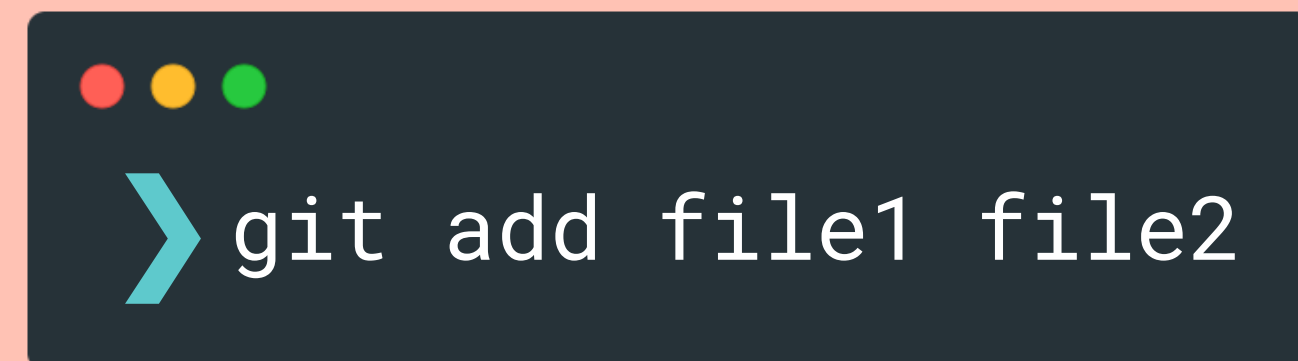
Add branded navbar





Adding

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

A dark blue terminal window with rounded corners and three colored window control buttons (red, yellow, green) in the top-left corner. It contains a light blue prompt character followed by the command `git add file1 file2` in white text.

```
> git add file1 file2
```





Adding

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

A terminal window icon with three colored dots (red, yellow, green) in the top left corner.

```
> git add .
```



Working Directory

modified navbar.html

modified about.html

created lisa.jpg

Staging Area

Repository

Add team members
to about page

Add branded navbar

Working Directory

modified about.html

created lisa.jpg

Staging Area

modified navbar.html

Repository

Add team members
to about page

Add branded navbar

Working Directory

modified about.html

created lisa.jpg

Staging Area

Repository

modified navbar.html

Fixed navbar typo

Add team members
to about page

Add branded navbar

Working Directory

modified about.html

created lisa.jpg

Staging Area

Repository

Fixed navbar typo

Add team members
to about page

Add branded navbar

Working Directory

created lisa.jpg

Staging Area

modified about.html

Repository

Fixed navbar typo

Add team members
to about page

Add branded navbar

Working Directory

Staging Area

modified about.html

created lisa.jpg

Repository

Fixed navbar typo

Add team members
to about page

Add branded navbar

Working Directory

Staging Area

Repository

modified about.html

created lisa.jpg

Add new hire Lisa to
the about page

Fixed navbar typo

Add team members
to about page

Add branded navbar

Working Directory

Staging Area

Repository

Add new hire Lisa to the about page

Fixed navbar typo

Add team members to about page

Add branded navbar



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

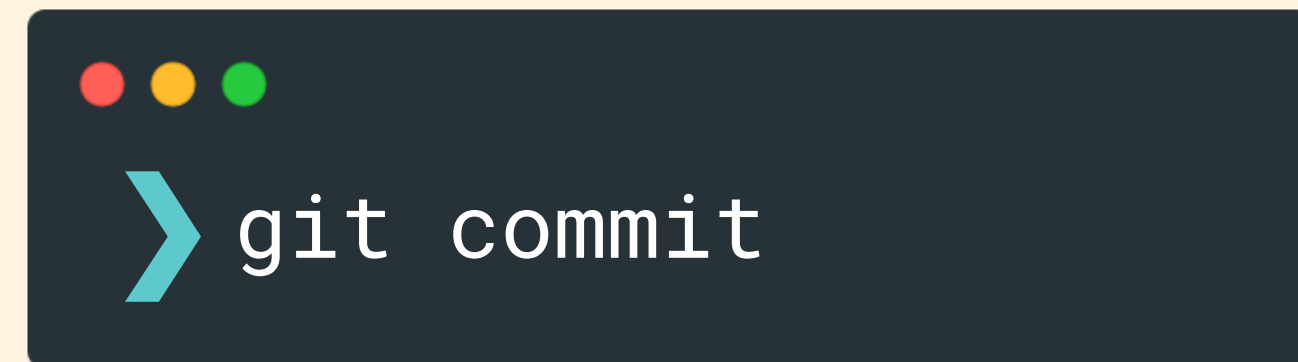




Git Commit

Running **git commit** will commit all staged changes. It also opens up a text editor and prompts you for a commit message.

This can be overwhelming when you're starting out, so instead you can use...





Git Commit

The `-m` flag allows us to pass in an inline commit message, rather than launching a text editor.

We'll learn more about writing good commit messages later on.

A dark-themed terminal window icon with three colored dots (red, yellow, green) in the top left corner.

```
> git commit -m "my message"
```

