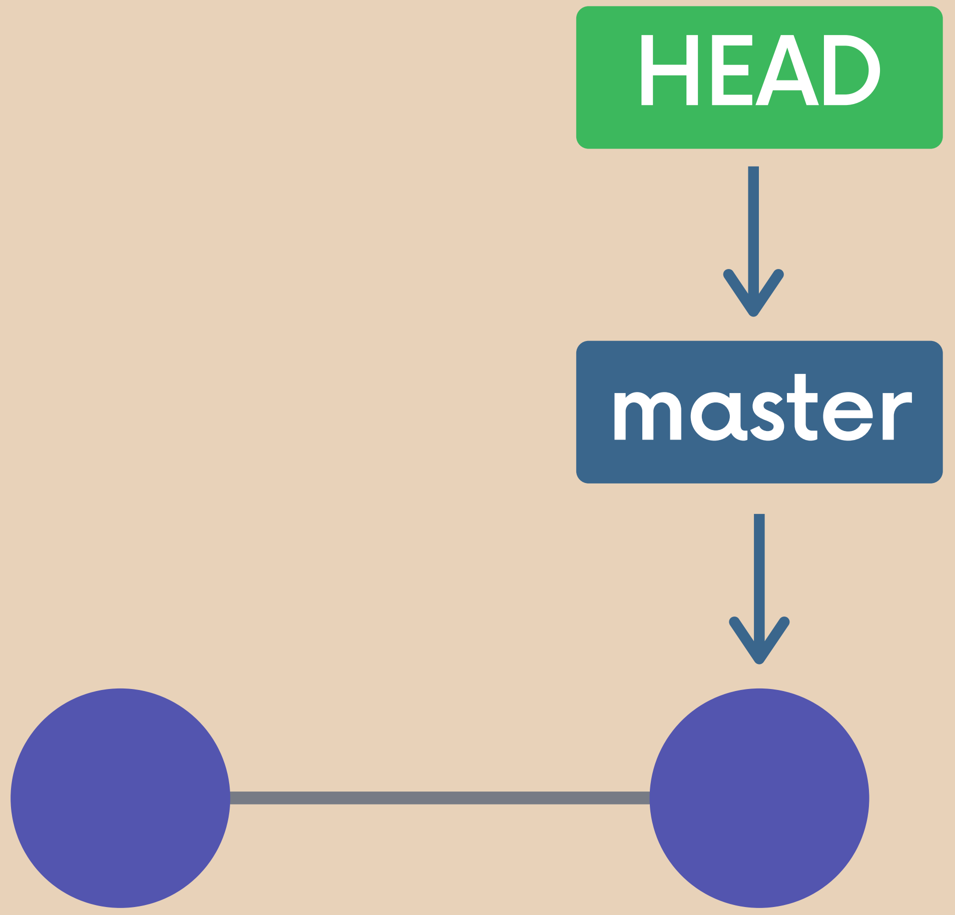
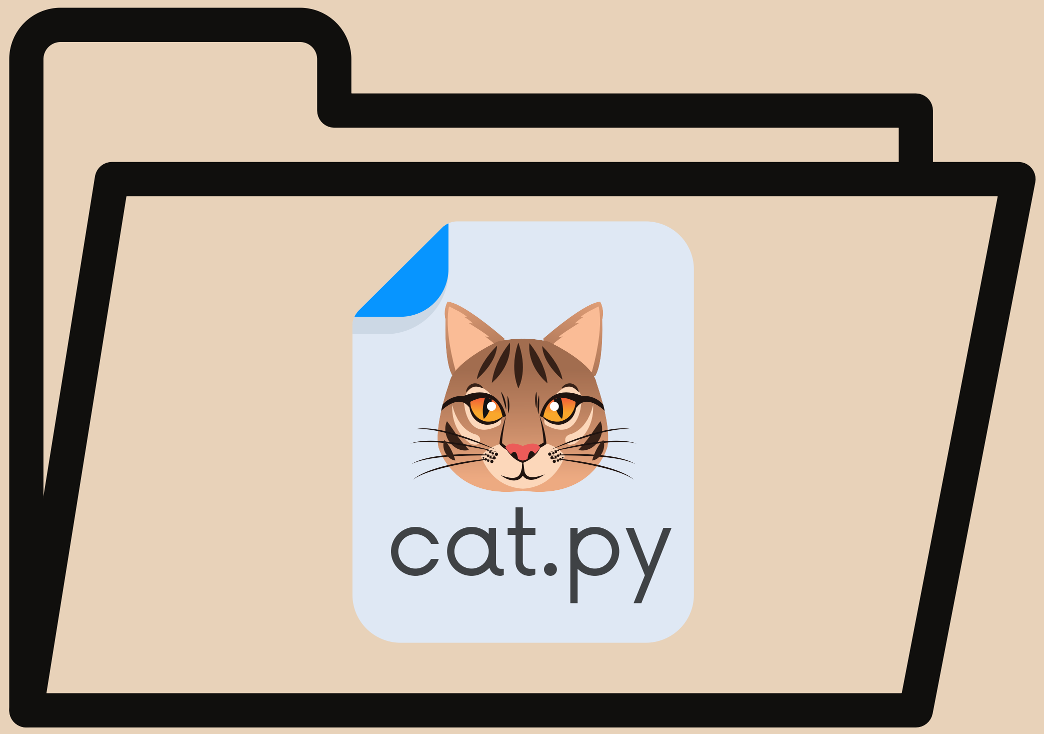
Git Stashing

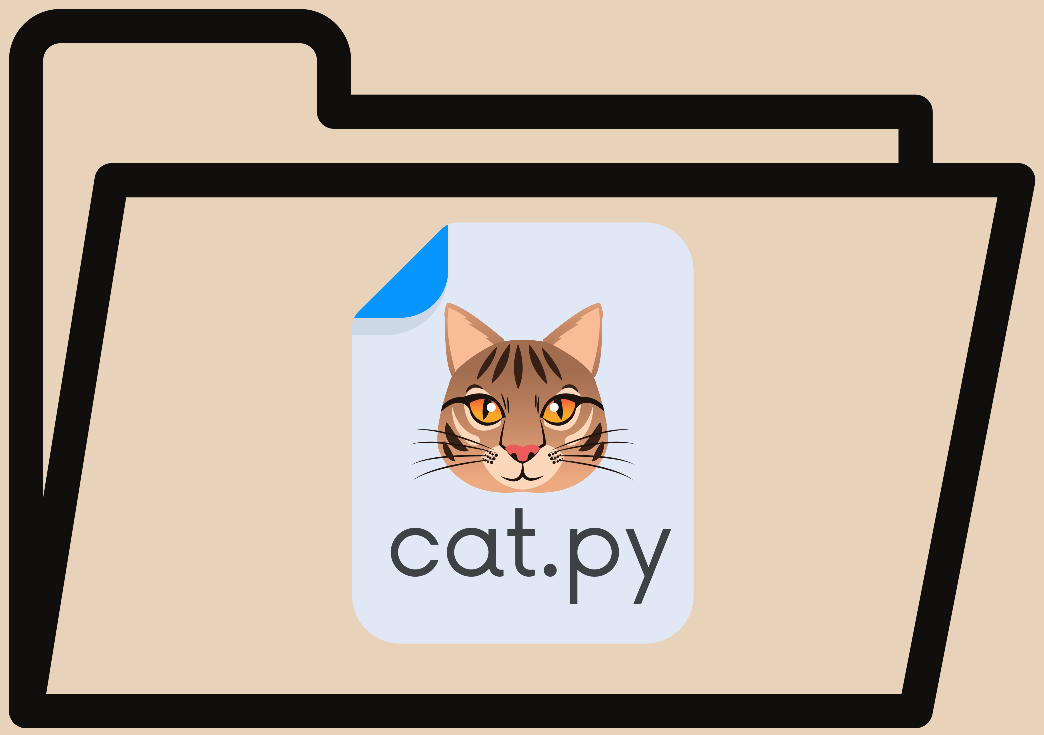
Imagine...

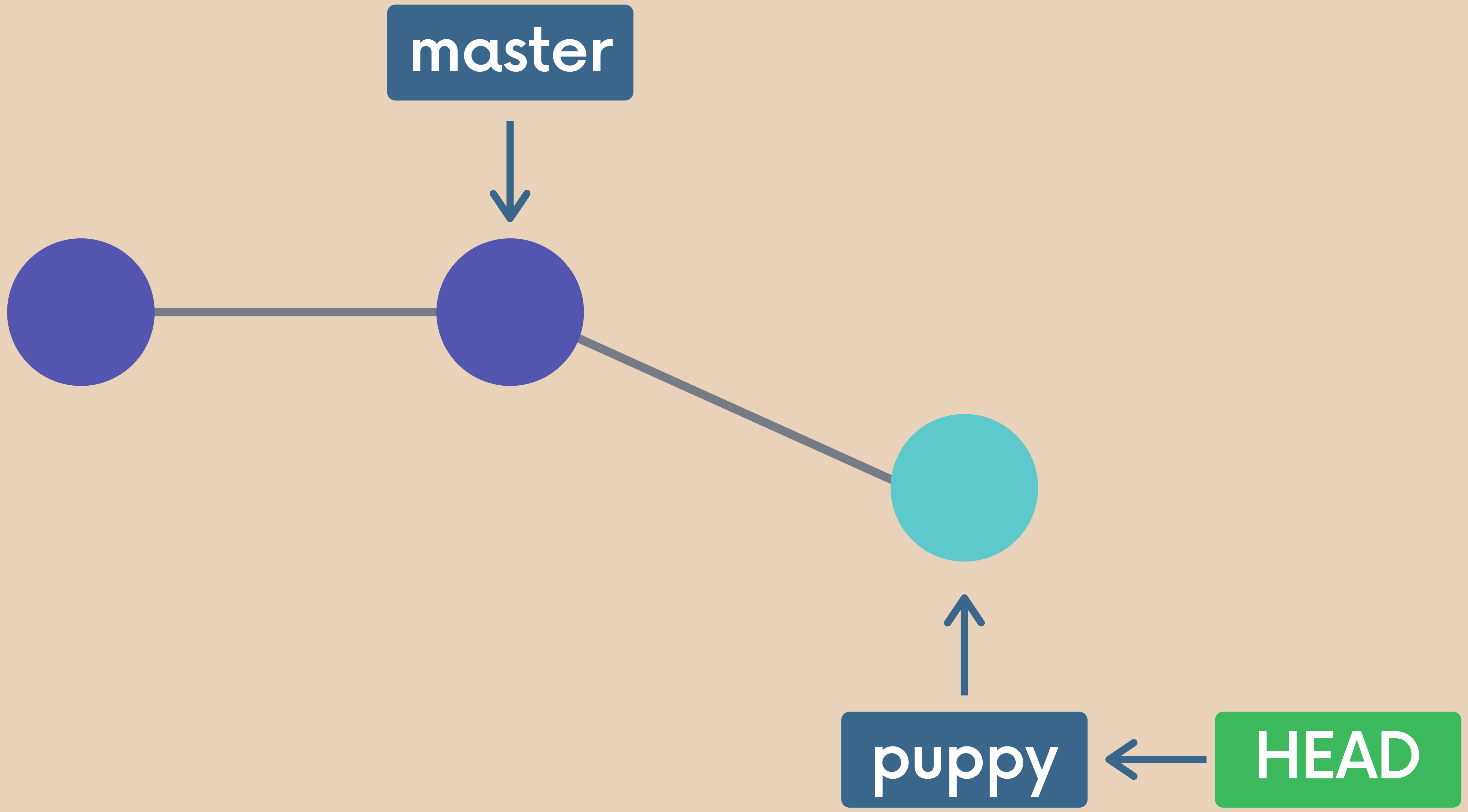
I'm on master



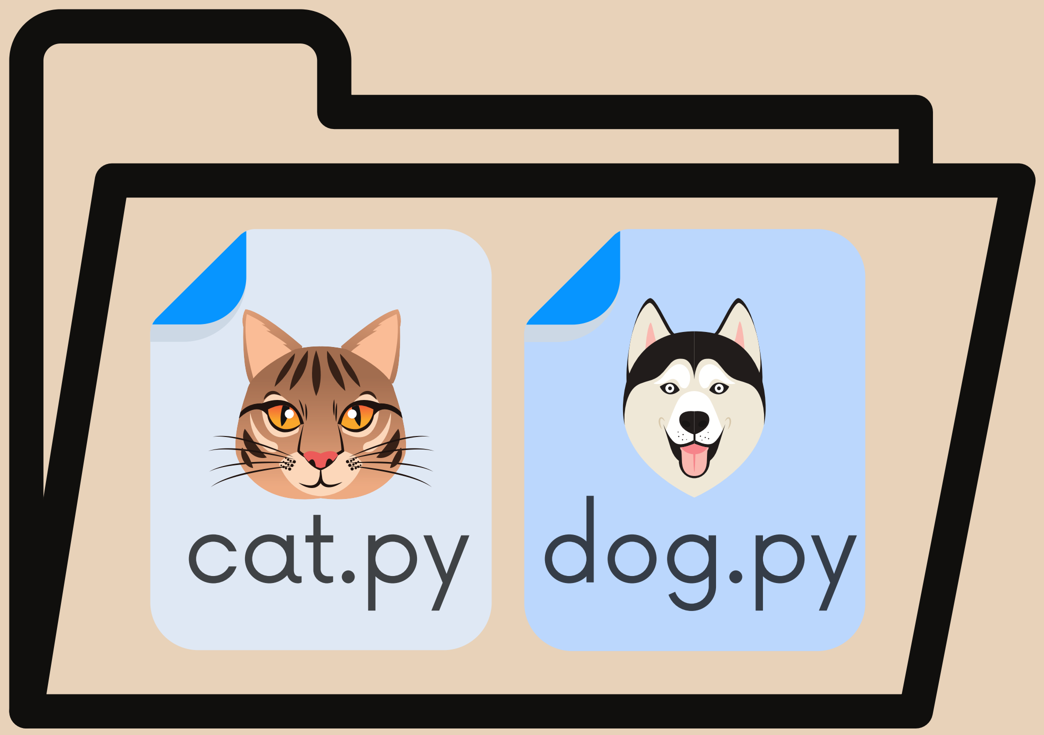


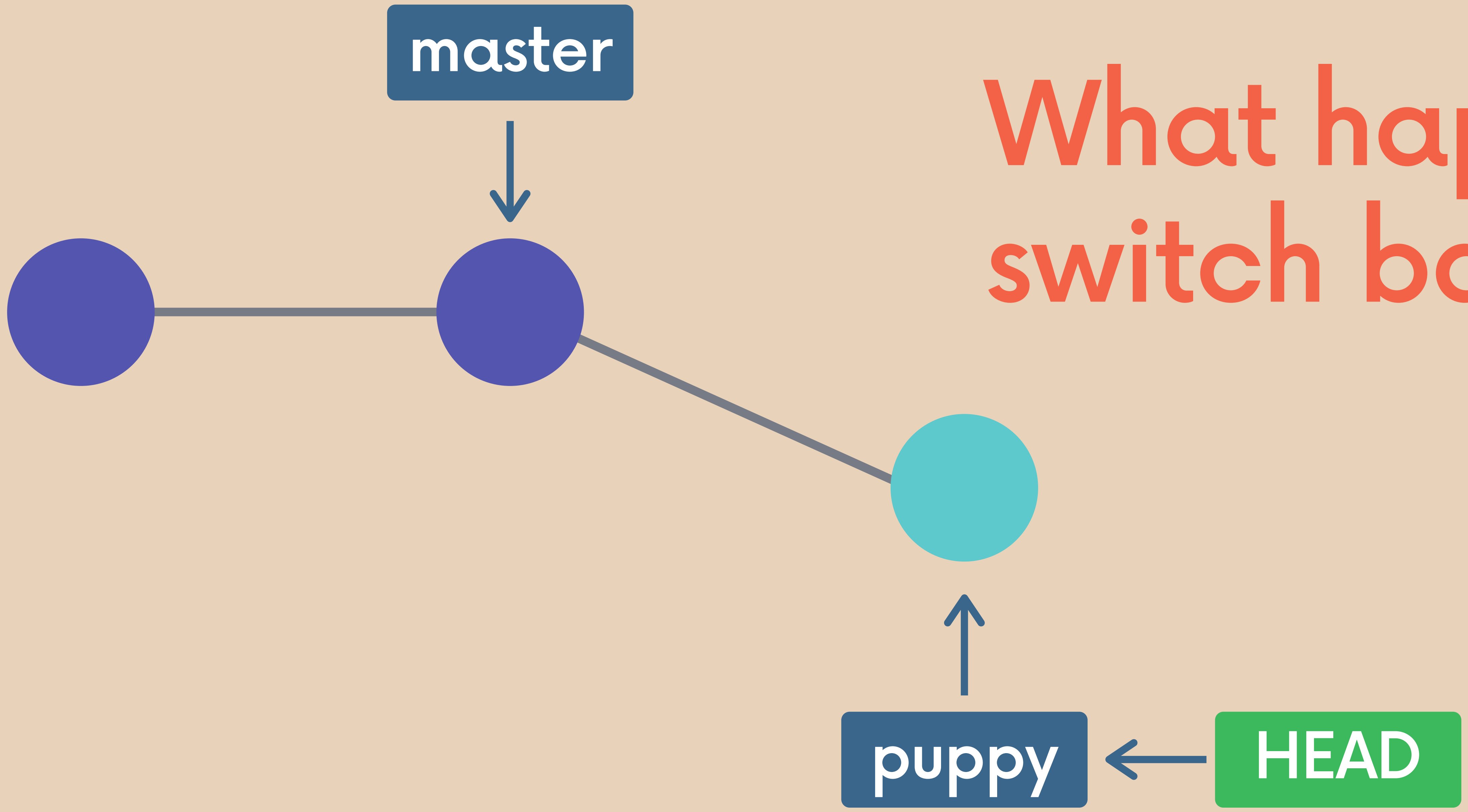
I make a new branch and switch to it





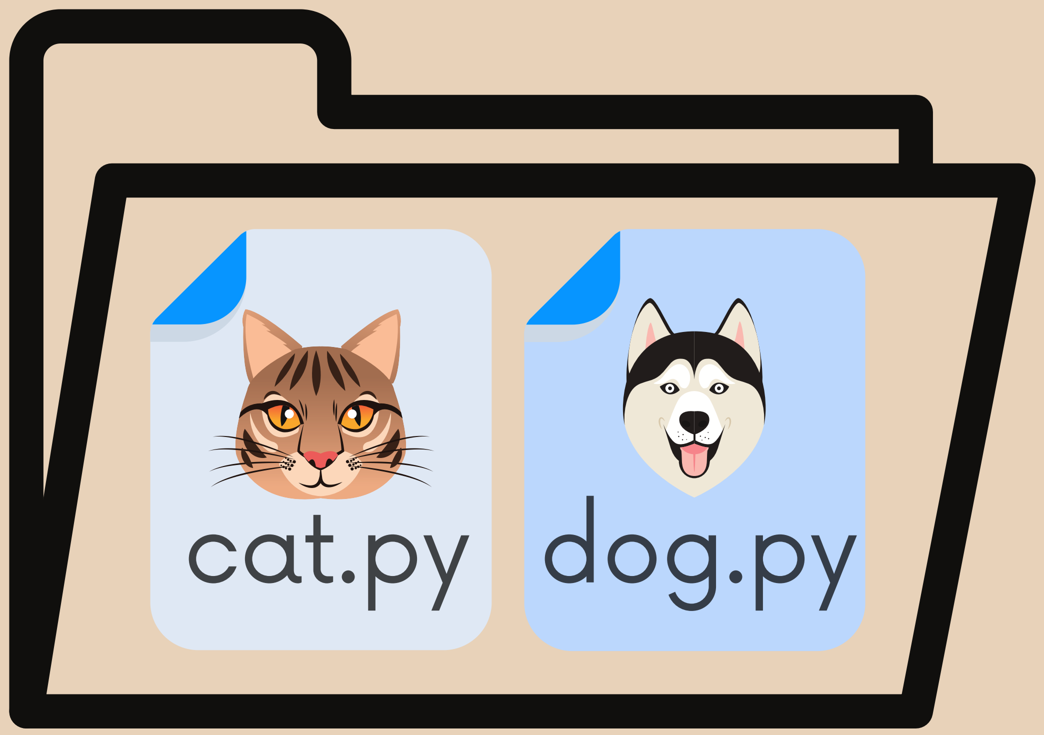
I do some new work, but don't make any commits

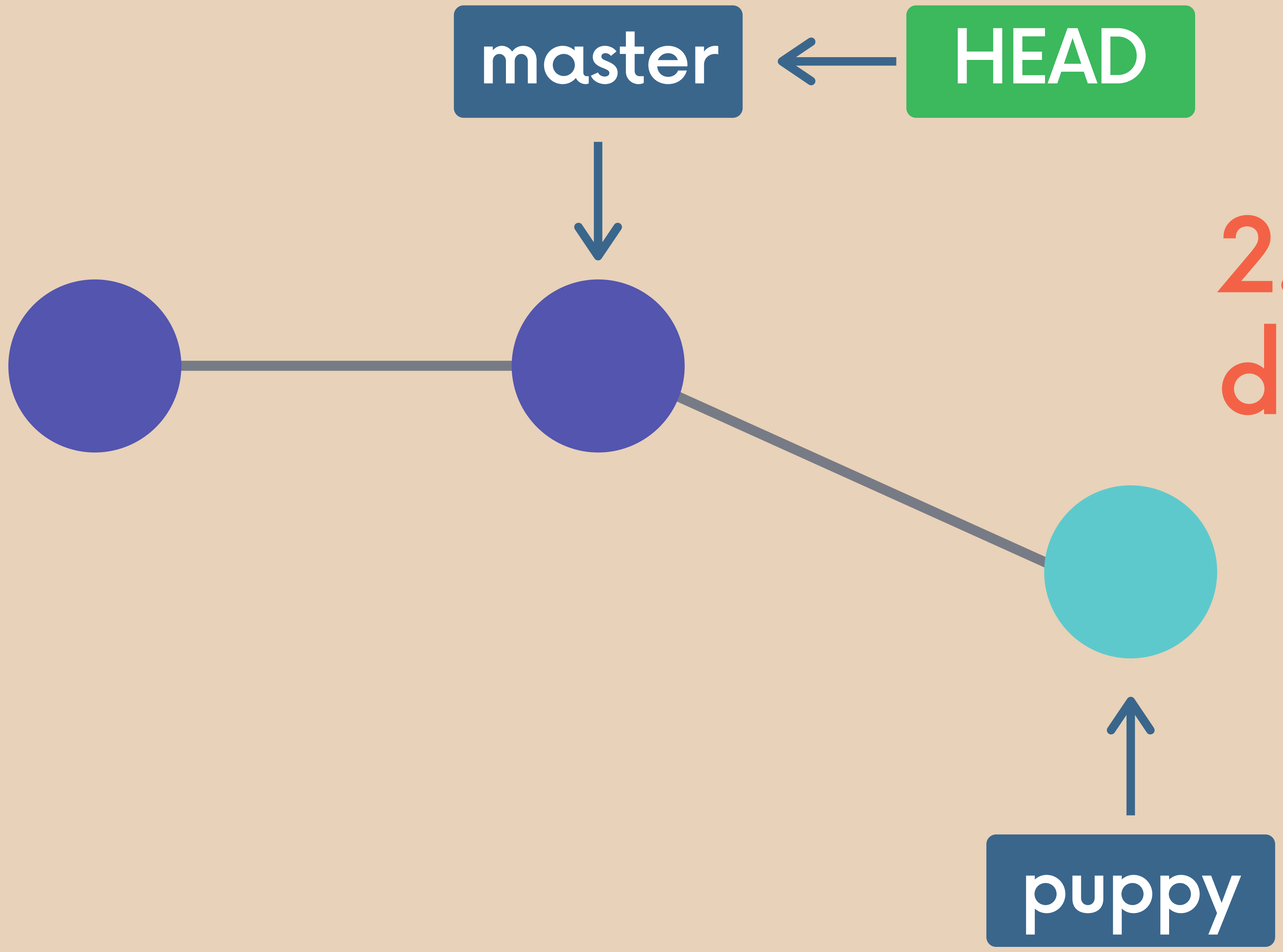




What happens when I switch back to master?

1. My changes come with me to the destination branch





2. Git won't let me switch if it detects potential conflicts

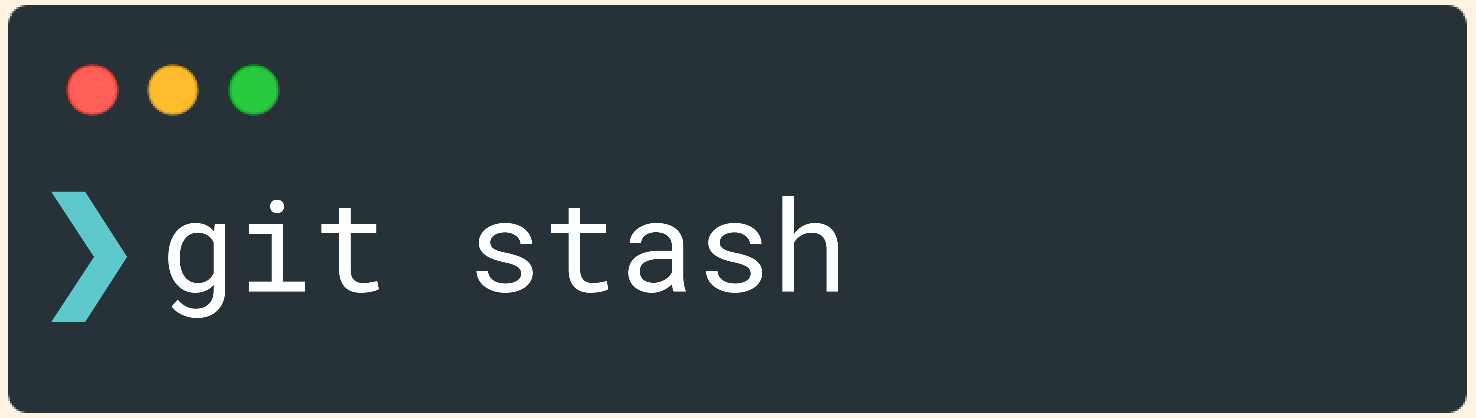
Stashing

Git provides an easy way of stashing these uncommitted changes so that we can return to them later, without having to make unnecessary commits.

Git Stash

git stash is super useful command that helps you save changes that you are not yet ready to commit. You can stash changes and then come back to them later.

Running git stash will take all uncommitted changes (staged and unstaged) and stash them, reverting the changes in your working copy.

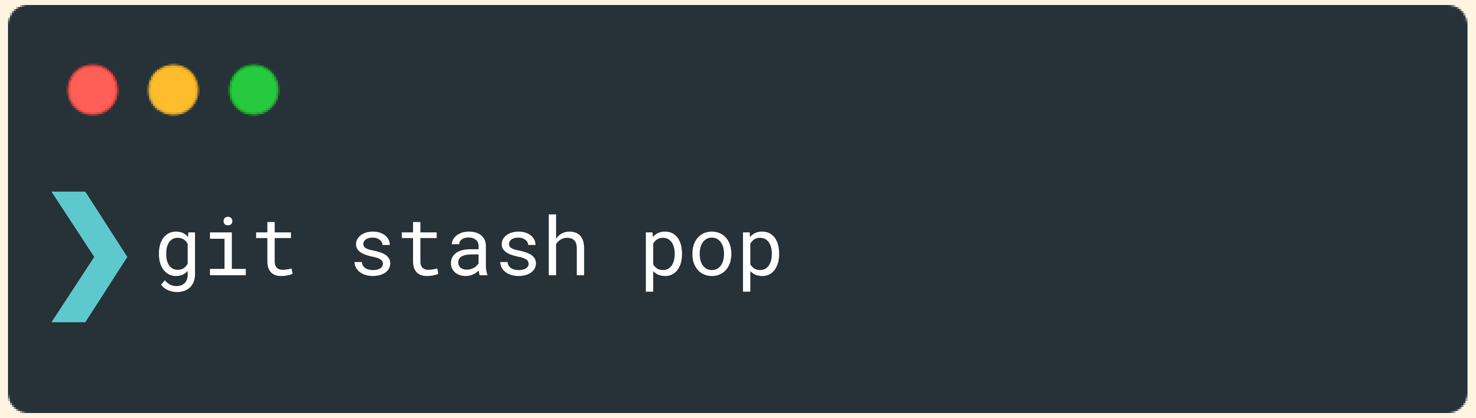


```
❯ git stash
```

You can also use **git stash save** instead

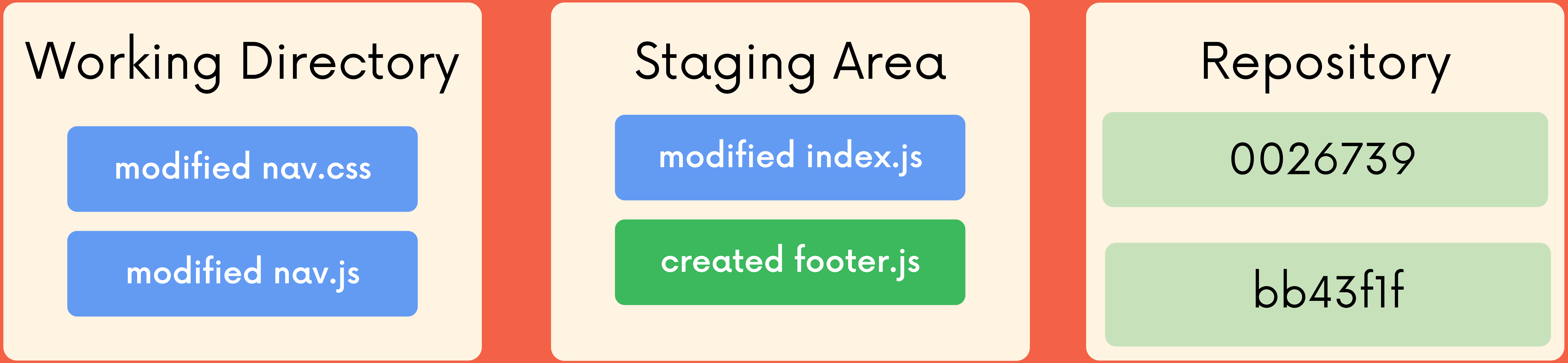
Stashing

Use **git stash pop** to remove the most recently stashed changes in your stash and re-apply them to your working copy.



```
❯ git stash pop
```

Branch: bugfix

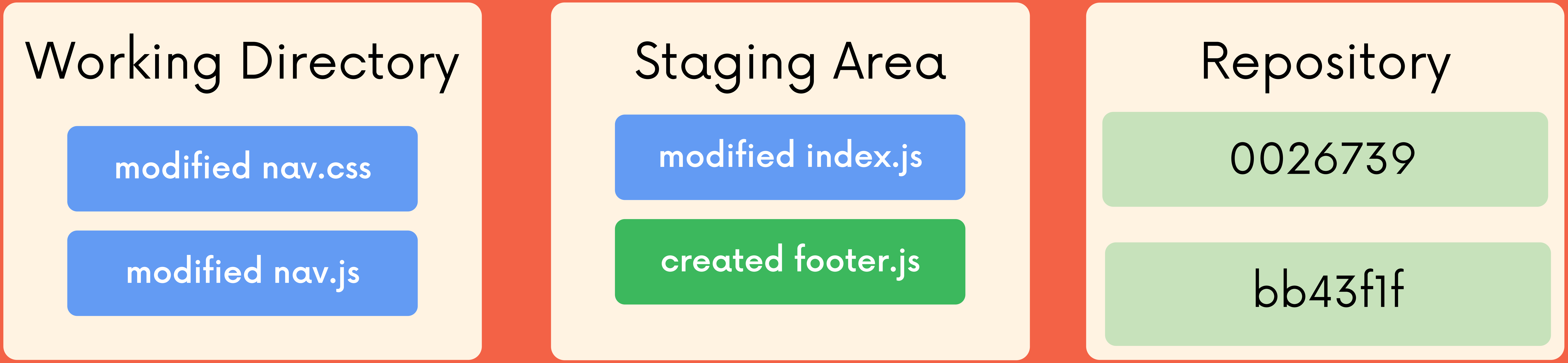


I'm working away on fixing a bug,
but then...

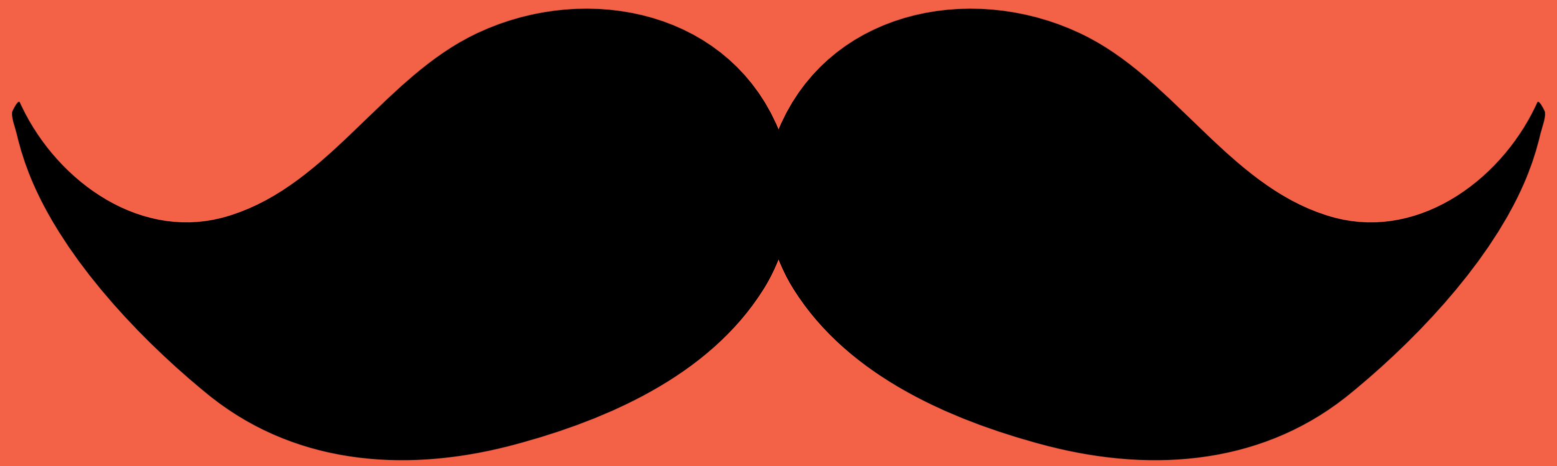
HELP!

Hey, can you take a look at what I merged into master and make sure I didn't screw up everything?

Branch: bugfix



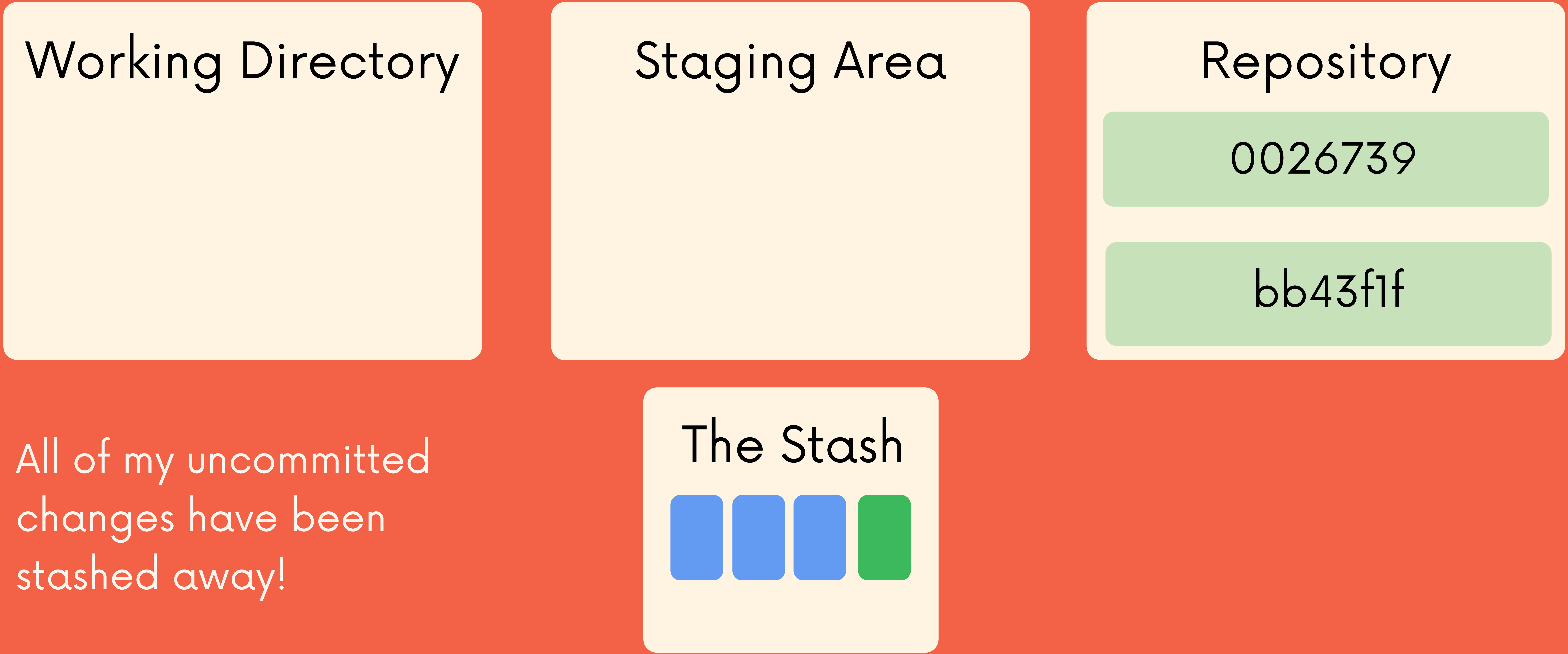
I'm not at all ready to commit these changes, but I also don't want to take them with me to **master**...

I Can 

Stash Them!

Branch: bugfix

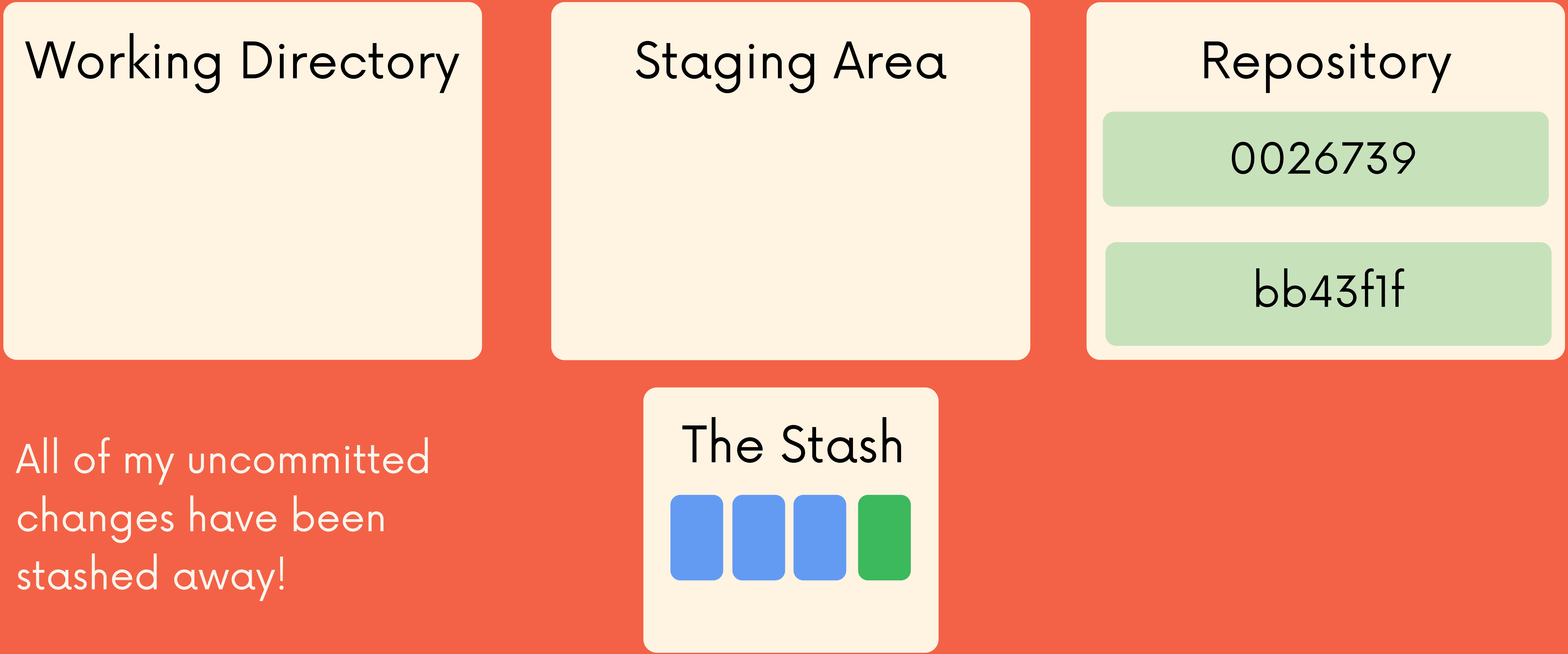
```
❯ git stash
```



All of my uncommitted changes have been stashed away!

Branch: bugfix

```
❯ git stash
```



All of my uncommitted changes have been stashed away!

Stashing

Now that I have stashed my changes, I can switch branches, create new commits, etc.

I head over to master and take a look at my coworker's changes.

When I'm done, I can re-apply the changes I stashed away at any point

Branch: bugfix

```
❯ git stash pop
```

Working Directory

- modified nav.css
- modified nav.js

Staging Area

- modified index.js
- created footer.js

Repository

- 0026739
- bb43f1f

The Stash

My stashed changes are restored!

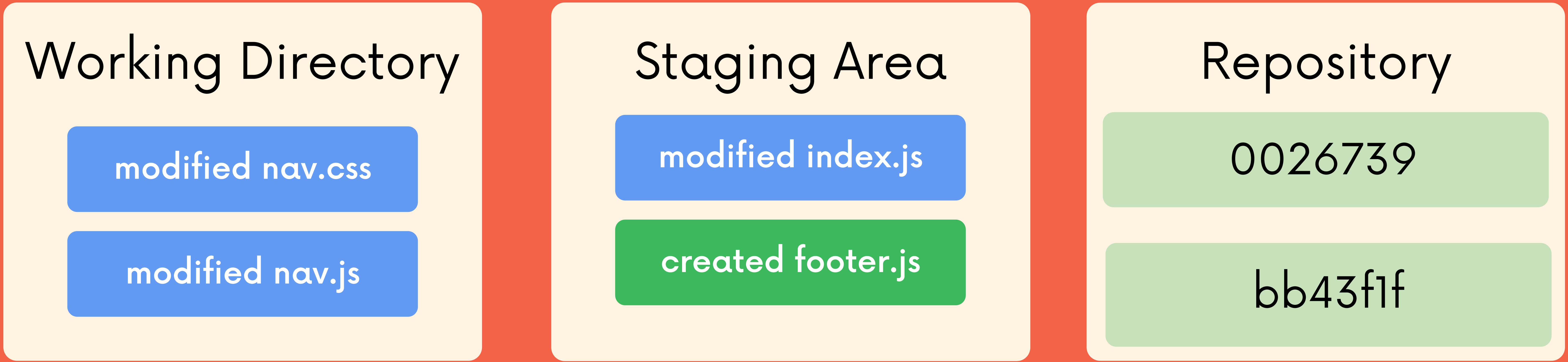
Stash Apply

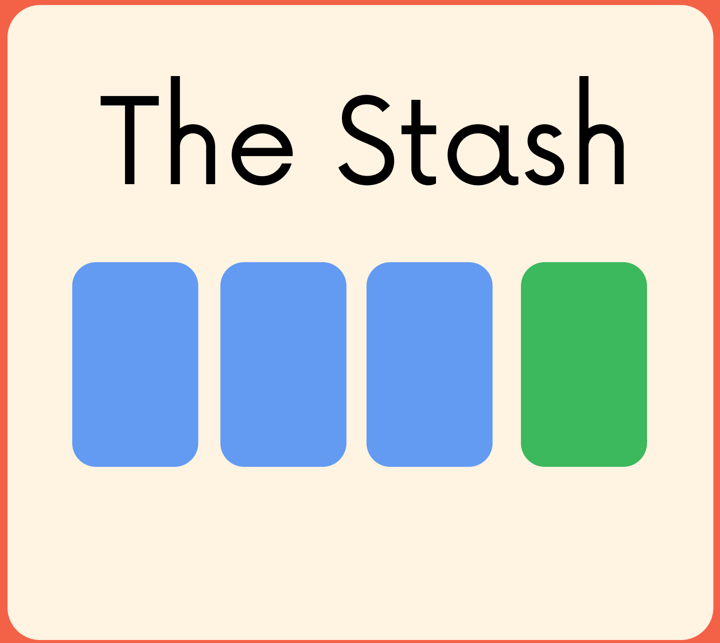
You can use **git stash apply** to apply whatever is stashed away, without removing it from the stash. This can be useful if you want to apply stashed changes to multiple branches.

```
❯ git stash apply
```

Branch: bugfix

```
❯ git stash apply
```



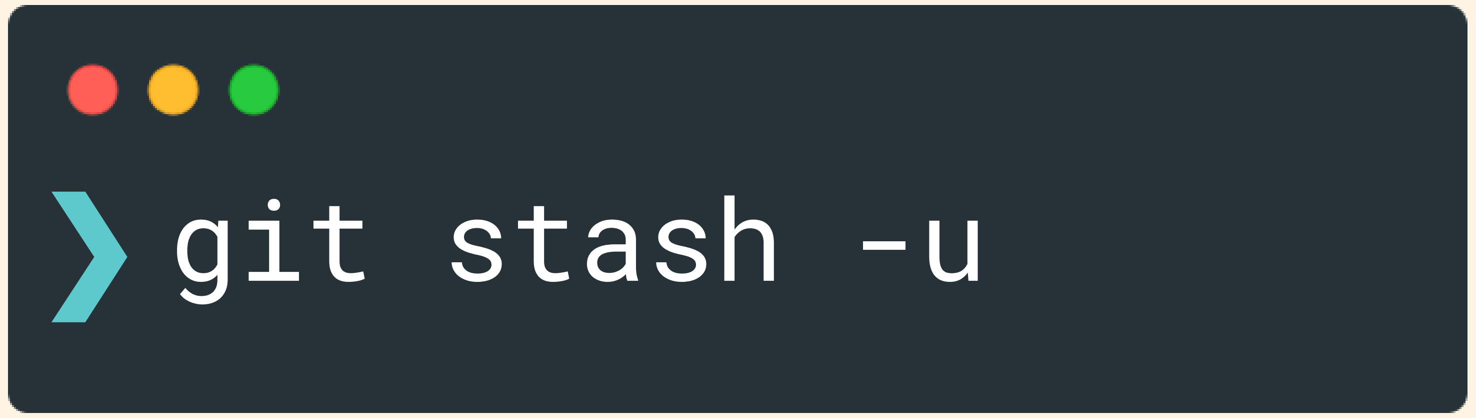


My stashed changes are restored!

Heads Up!

If you have untracked files (files that you have never checked in to Git), they will not be included in the stash.

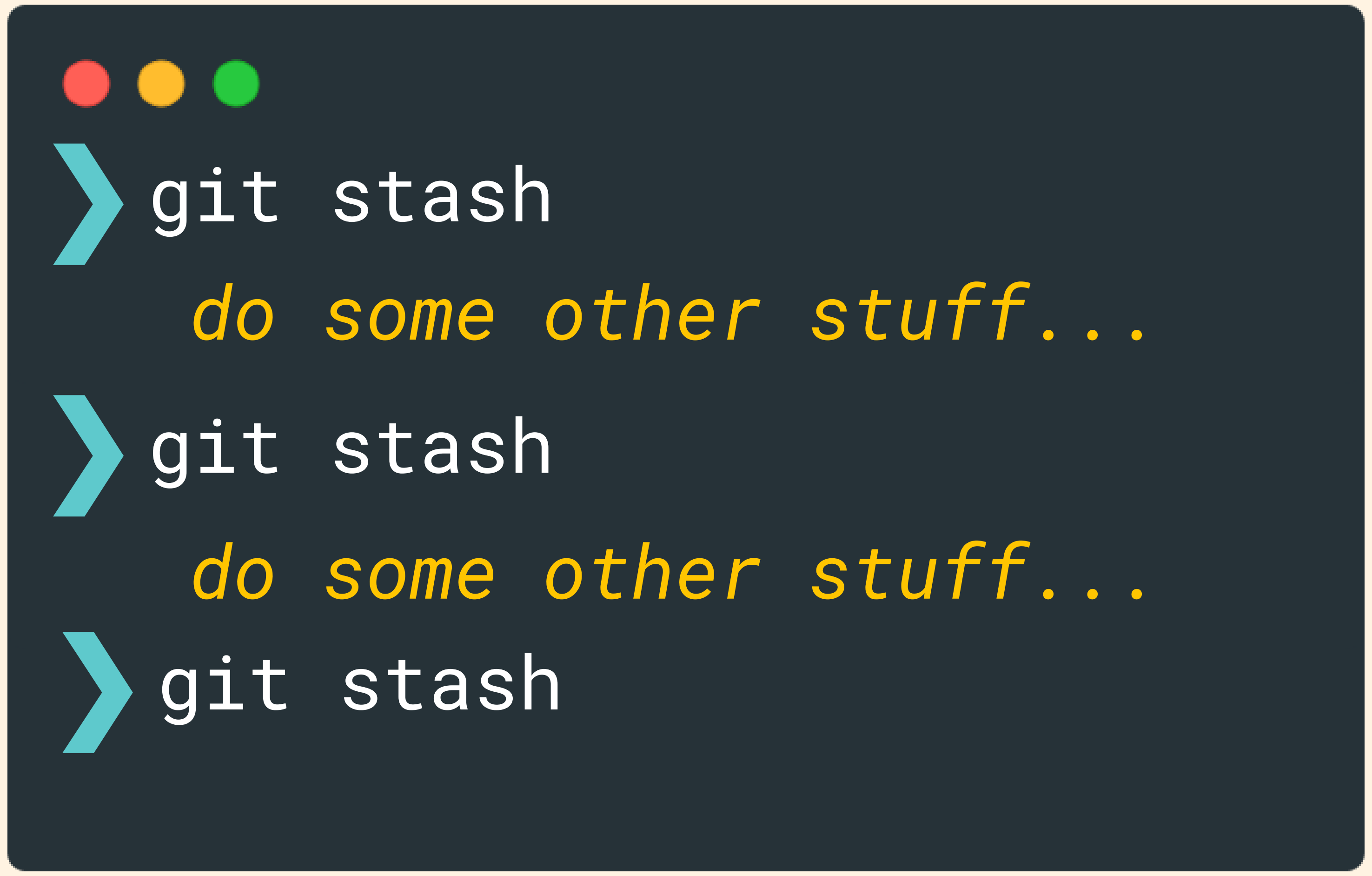
Fortunately, you can use the **`-u`** option to tell git stash to include those untracked files.



```
❯ git stash -u
```

Stashing Multiple Times

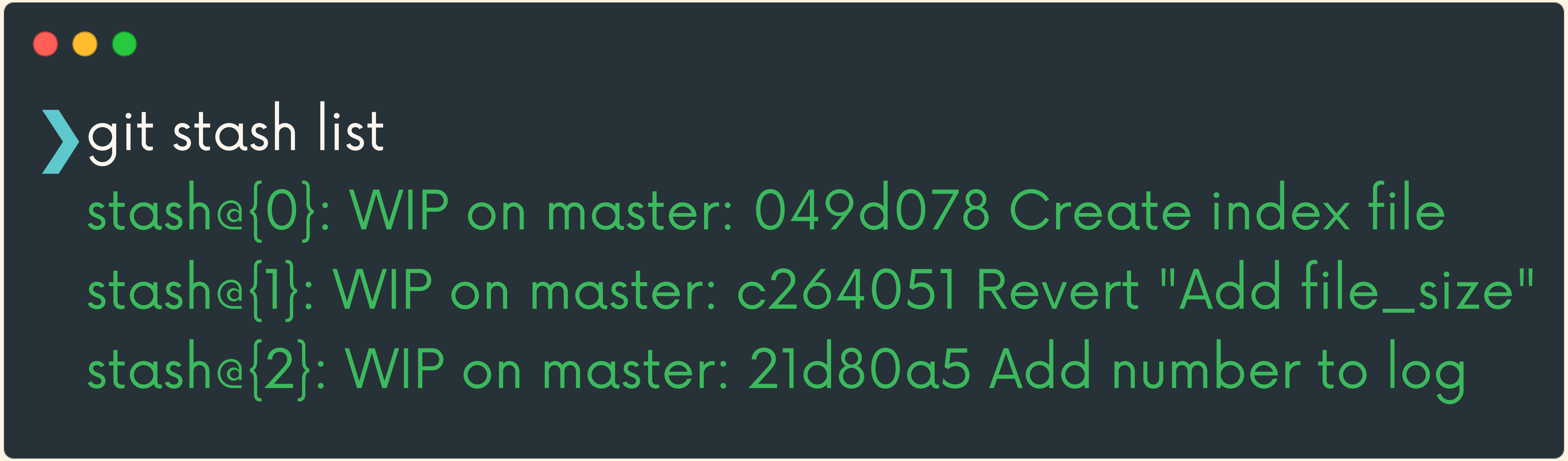
You can add multiple stashes onto the stack of stashes. They will all be stashed in the order you added them.



```
❯ git stash
   do some other stuff...
❯ git stash
   do some other stuff...
❯ git stash
```

Viewing Stashes

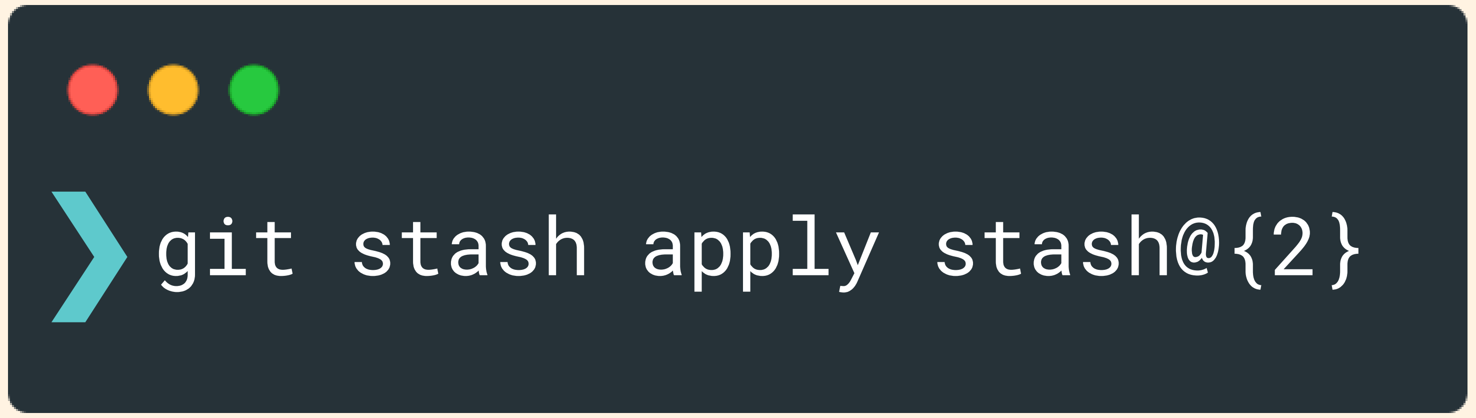
run **git stash list** to view all stashes



```
❯git stash list
  stash@{0}: WIP on master: 049d078 Create index file
  stash@{1}: WIP on master: c264051 Revert "Add file_size"
  stash@{2}: WIP on master: 21d80a5 Add number to log
```

Applying Specific Stashes

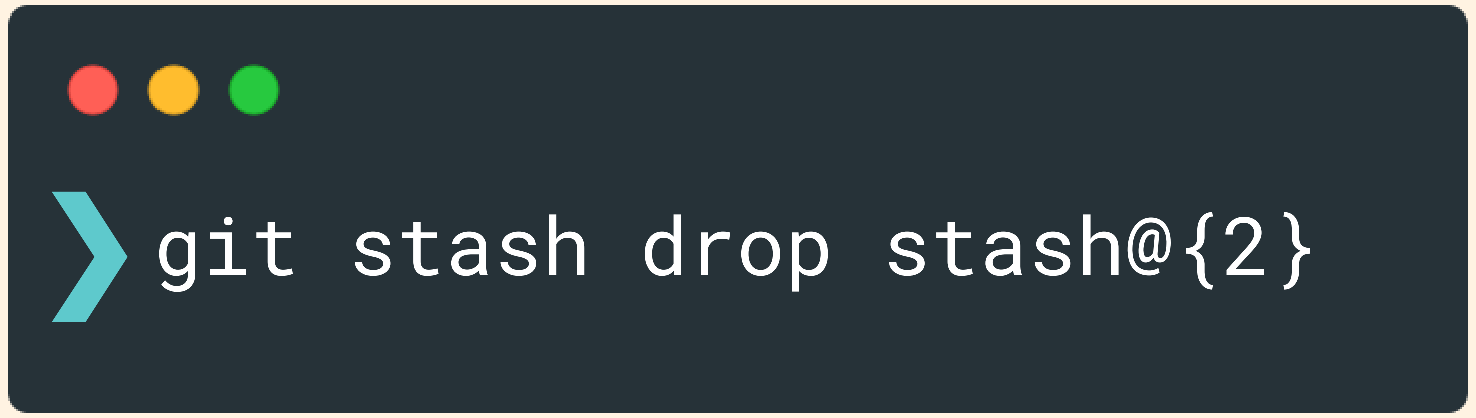
git assumes you want to apply the most recent stash when you run **git stash apply**, but you can also specify a particular stash like **git stash apply stash@{2}**



```
❯ git stash apply stash@{2}
```

Dropping Stashes

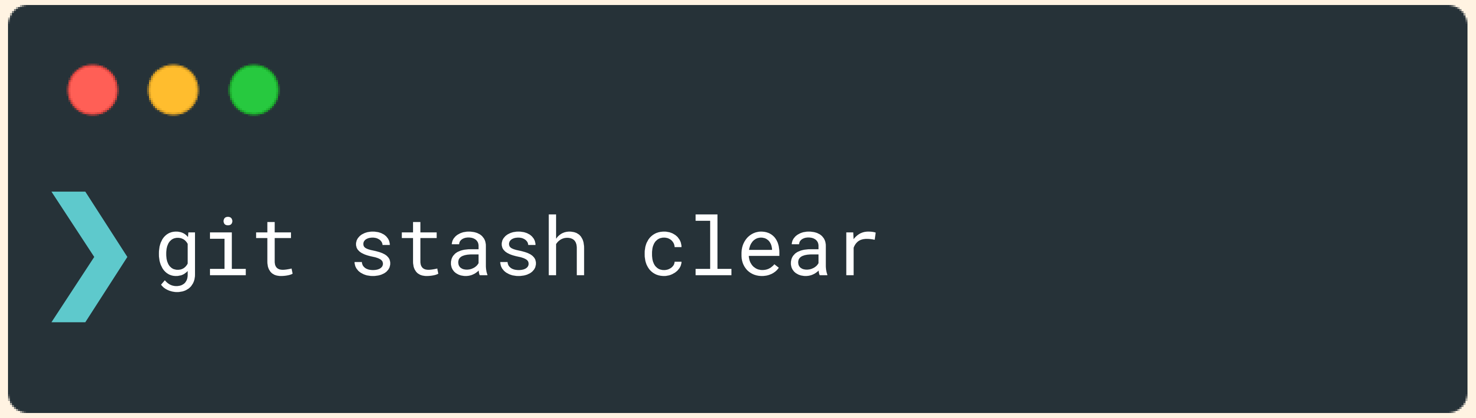
To delete a particular stash, you can use
git stash drop <stash-id>



```
git stash drop stash@{2}
```

Clearing The Stash

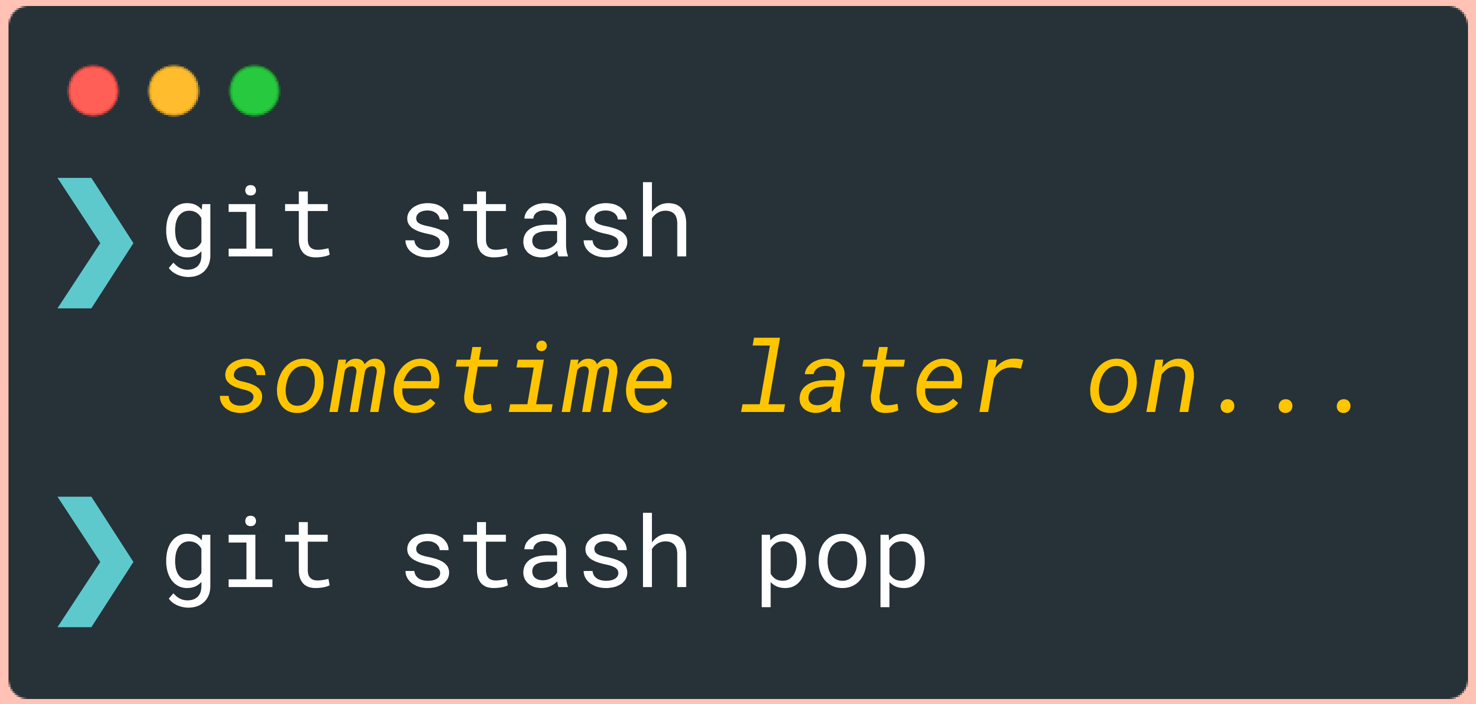
To clear out all stashes, run **git stash clear**



```
❯ git stash clear
```

Do I Really Need This?

99% of the time, I just use **git stash** and **git stash pop**.
Just my personal experience, YMMV!



```
❯ git stash
  sometime later on...
❯ git stash pop
```