

Fetching & Pulling

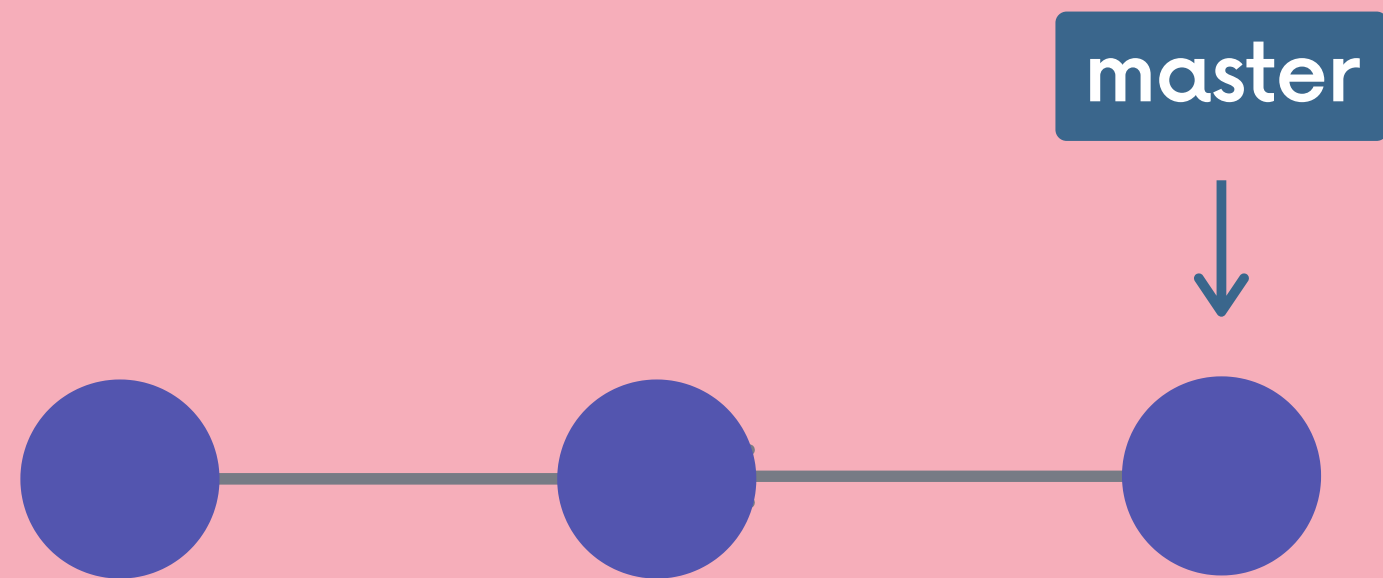


A Closer Look At Cloning



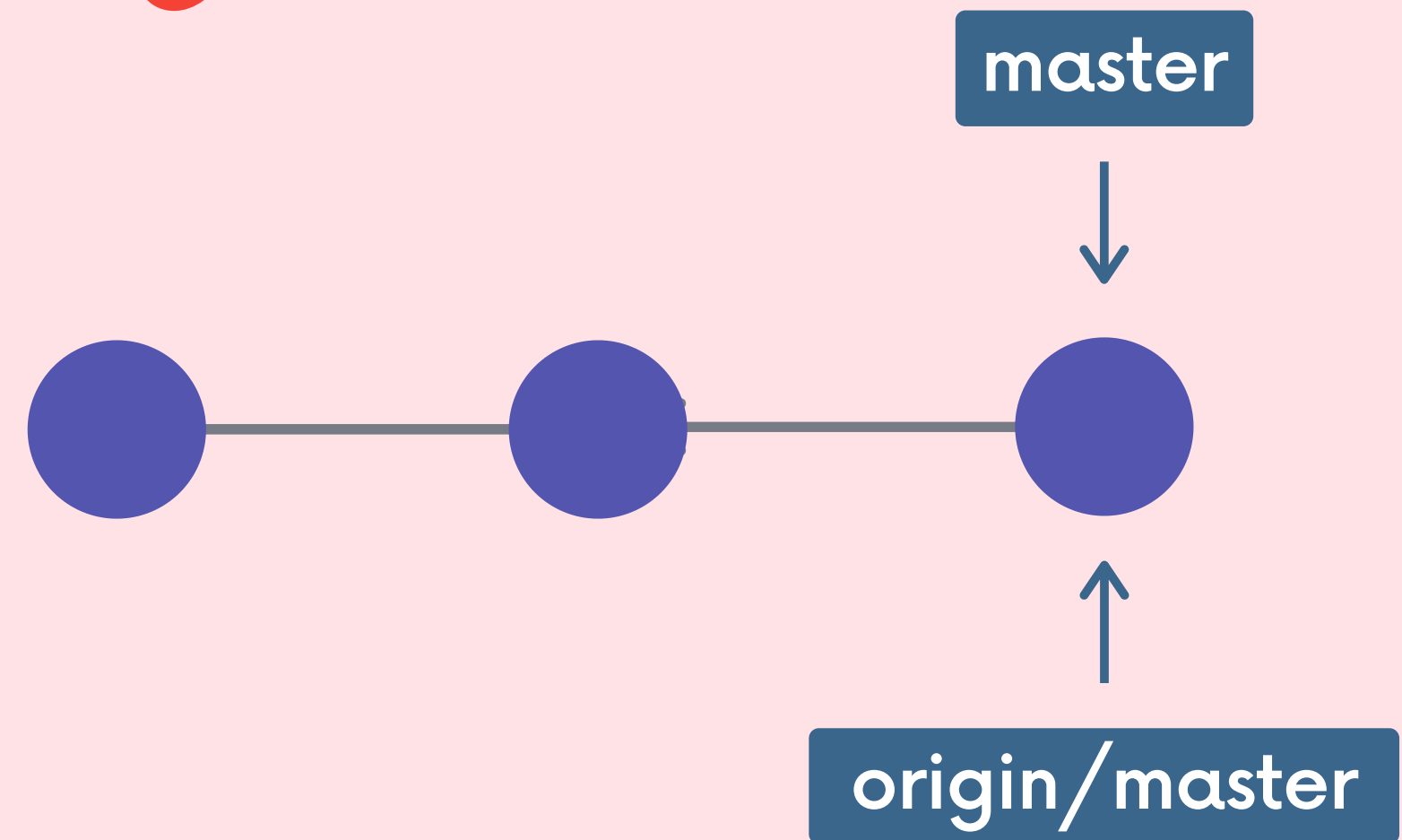
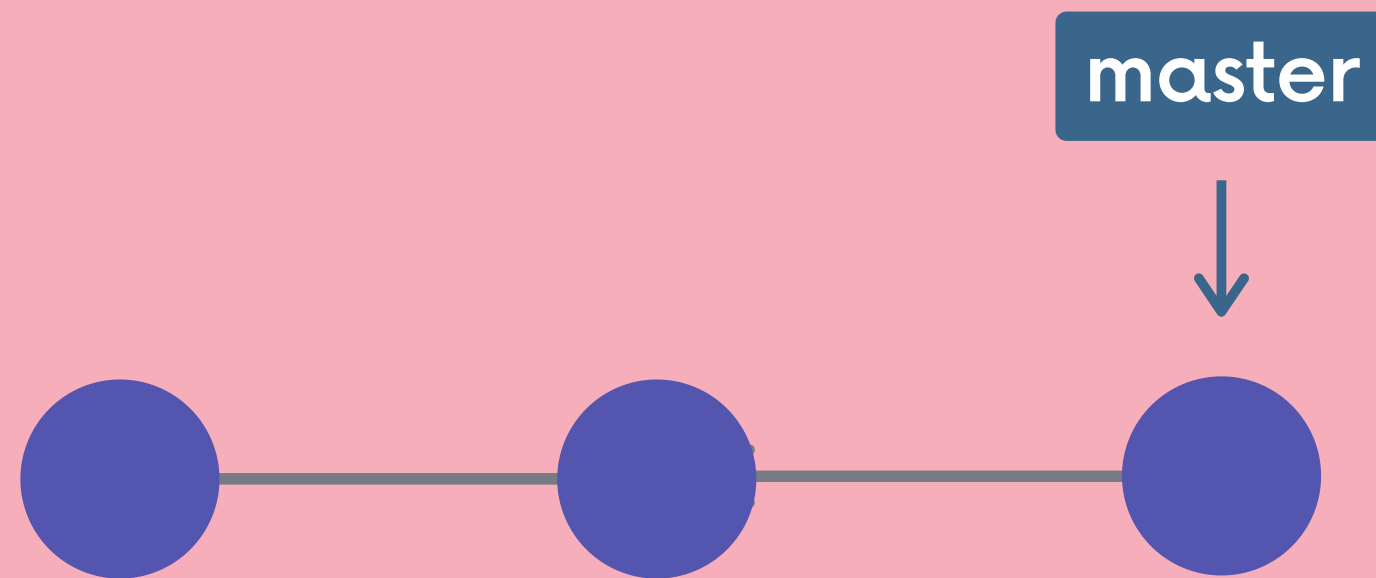
Github Repo

My Computer

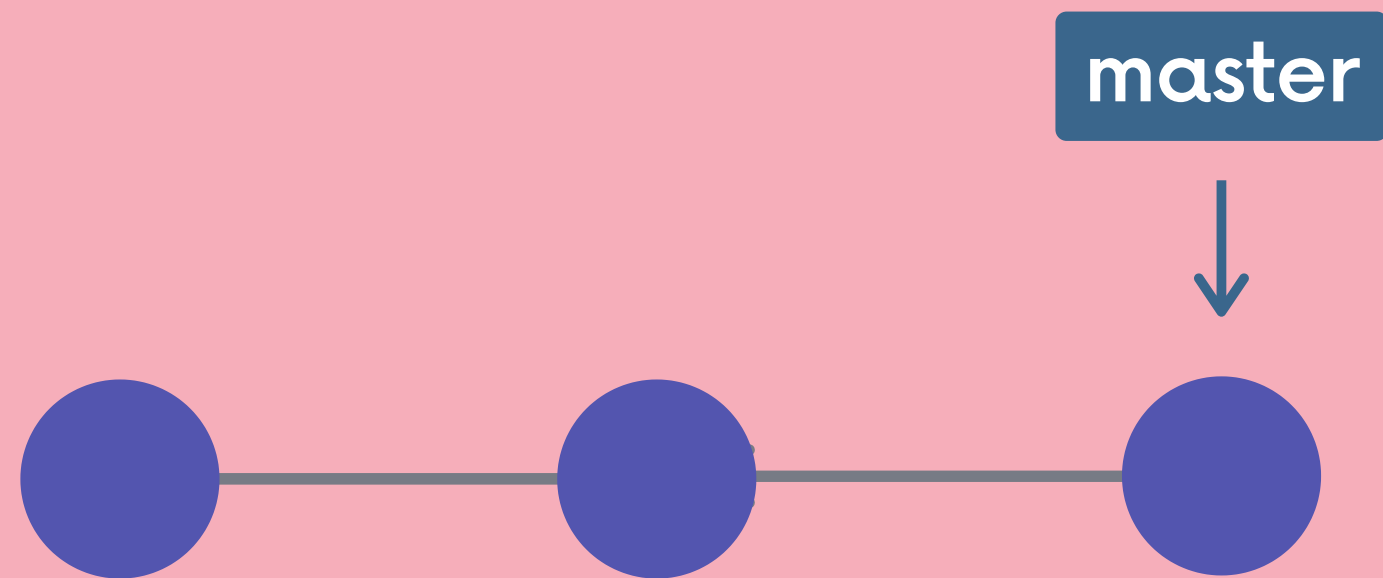


Github Repo

My Computer

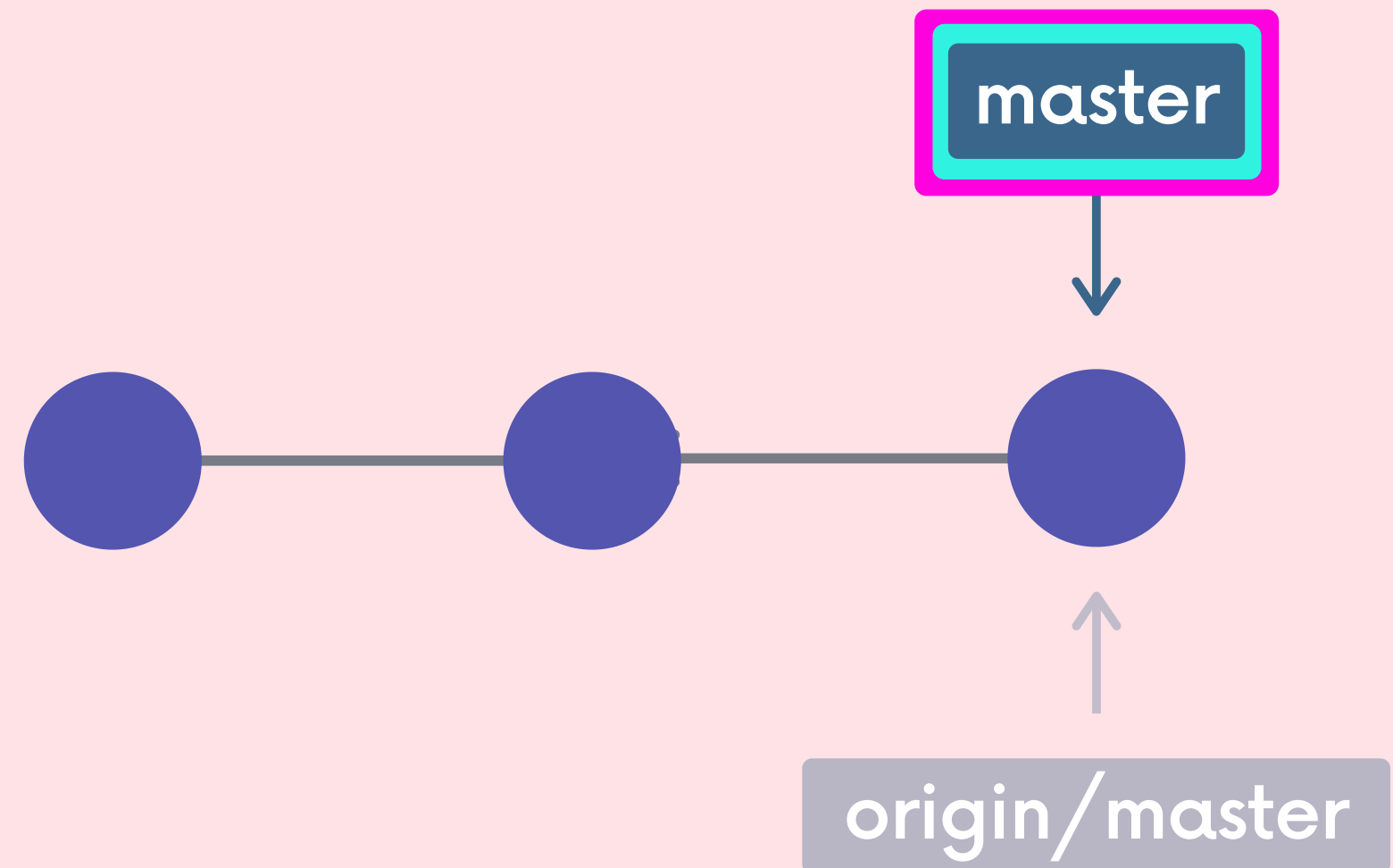


Github Repo



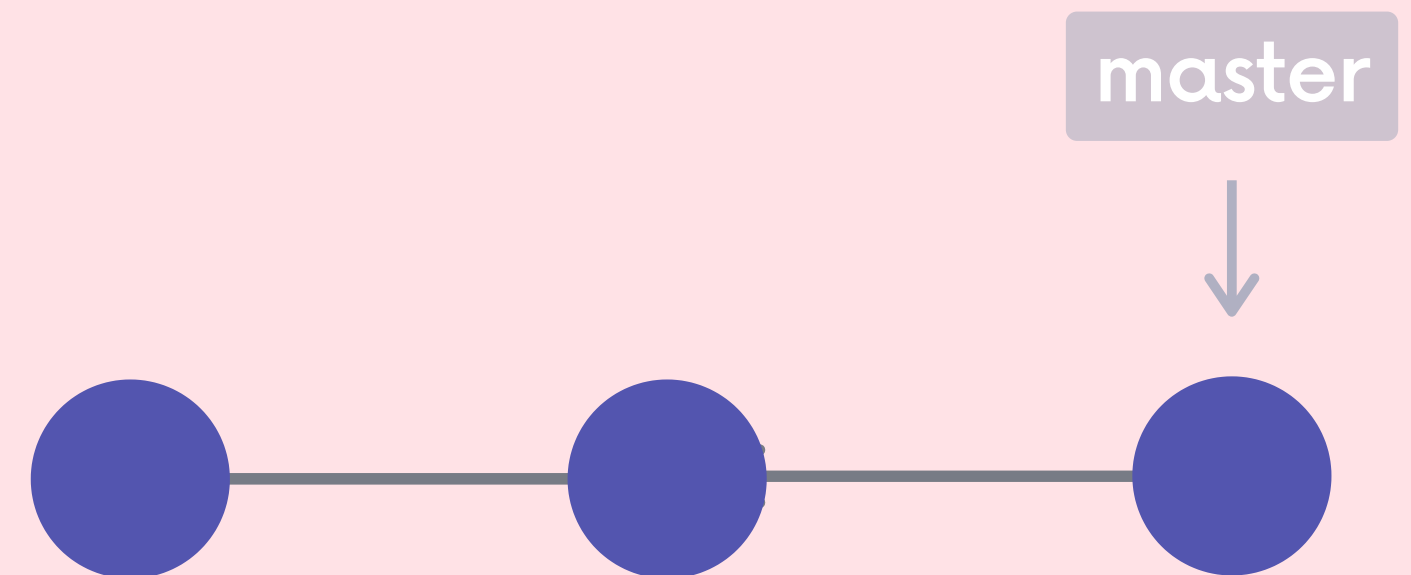
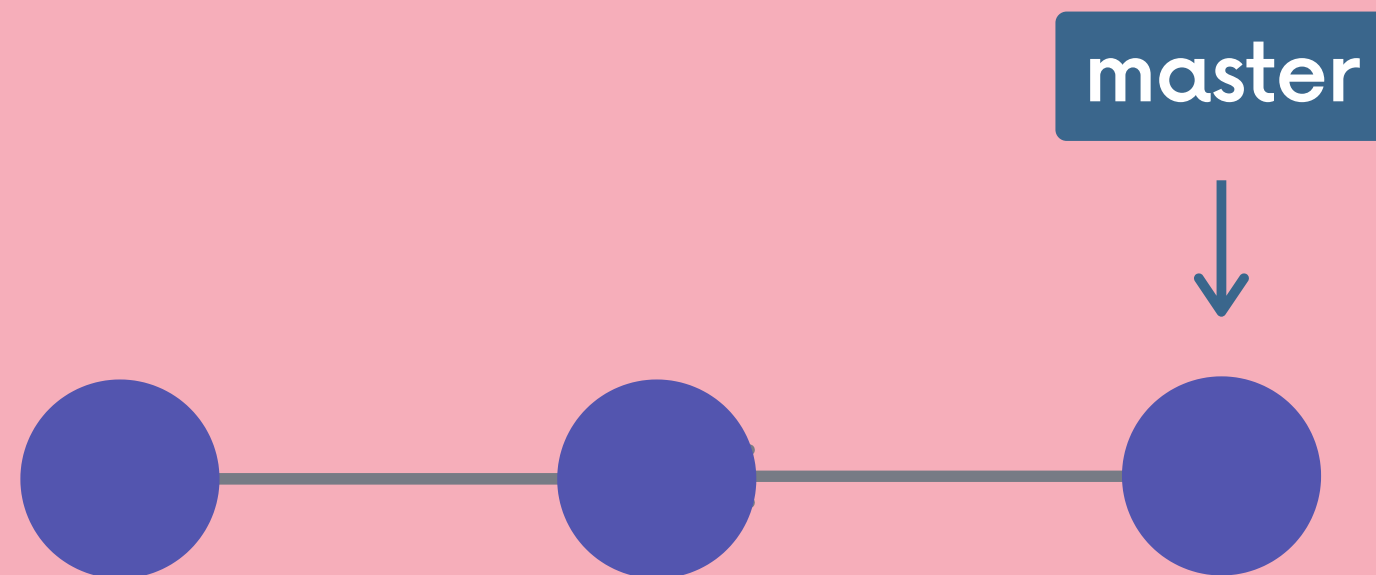
My Computer

A regular branch reference.
I can move this around myself.



Github Repo

My Computer



This is a "Remote Tracking Branch". It's a reference to the state of the master branch on the remote. I can't move this myself. It's like a bookmark pointing to the last known commit on the master branch on origin



Remote Tracking Branches

"At the time you last communicated with this remote repository, here is where x branch was pointing"

They follow this pattern `<remote>/<branch>`.

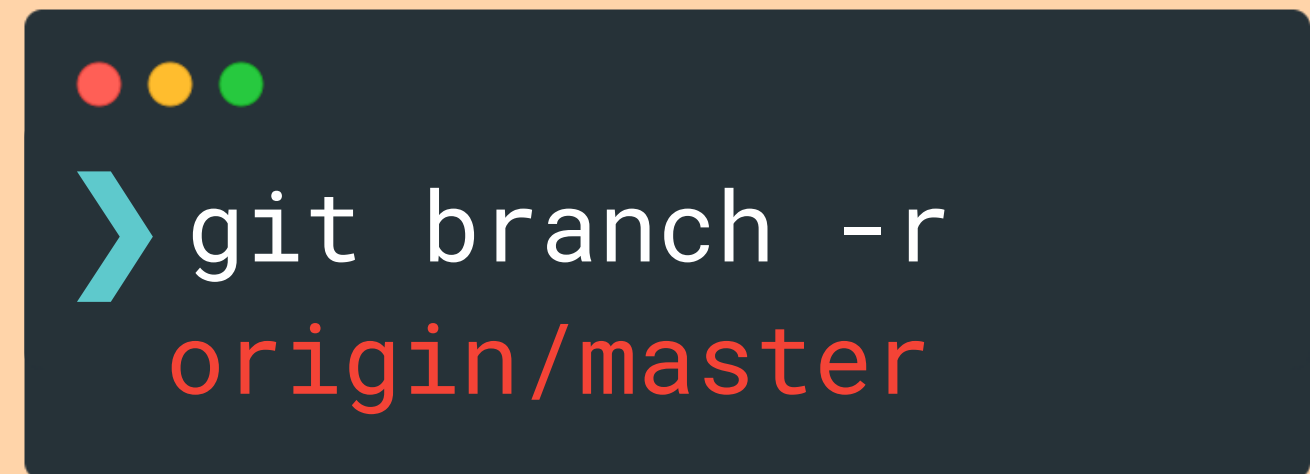
- **origin/master** references the state of the master branch on the remote repo named origin.
- **upstream/logoRedesign** references the state of the logoRedesign branch on the remote named upstream (a common remote name)





Remote Branches

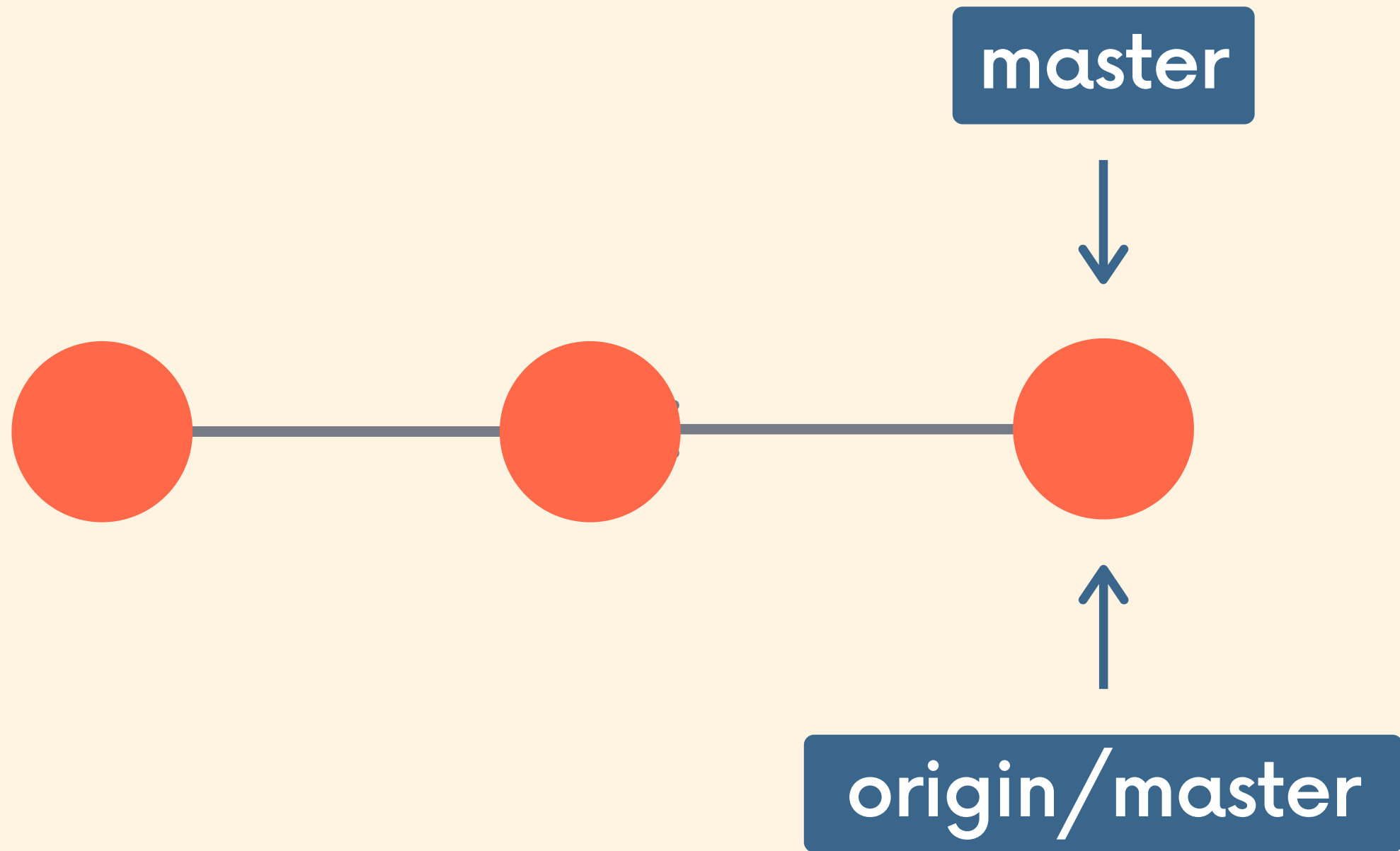
Run `git branch -r` to view the remote branches our local repository knows about.

A dark-themed terminal window with three colored window control buttons (red, yellow, green) in the top-left corner. A light blue prompt character is followed by the command `git branch -r` and its output `origin/master` in red text.

```
> git branch -r  
origin/master
```

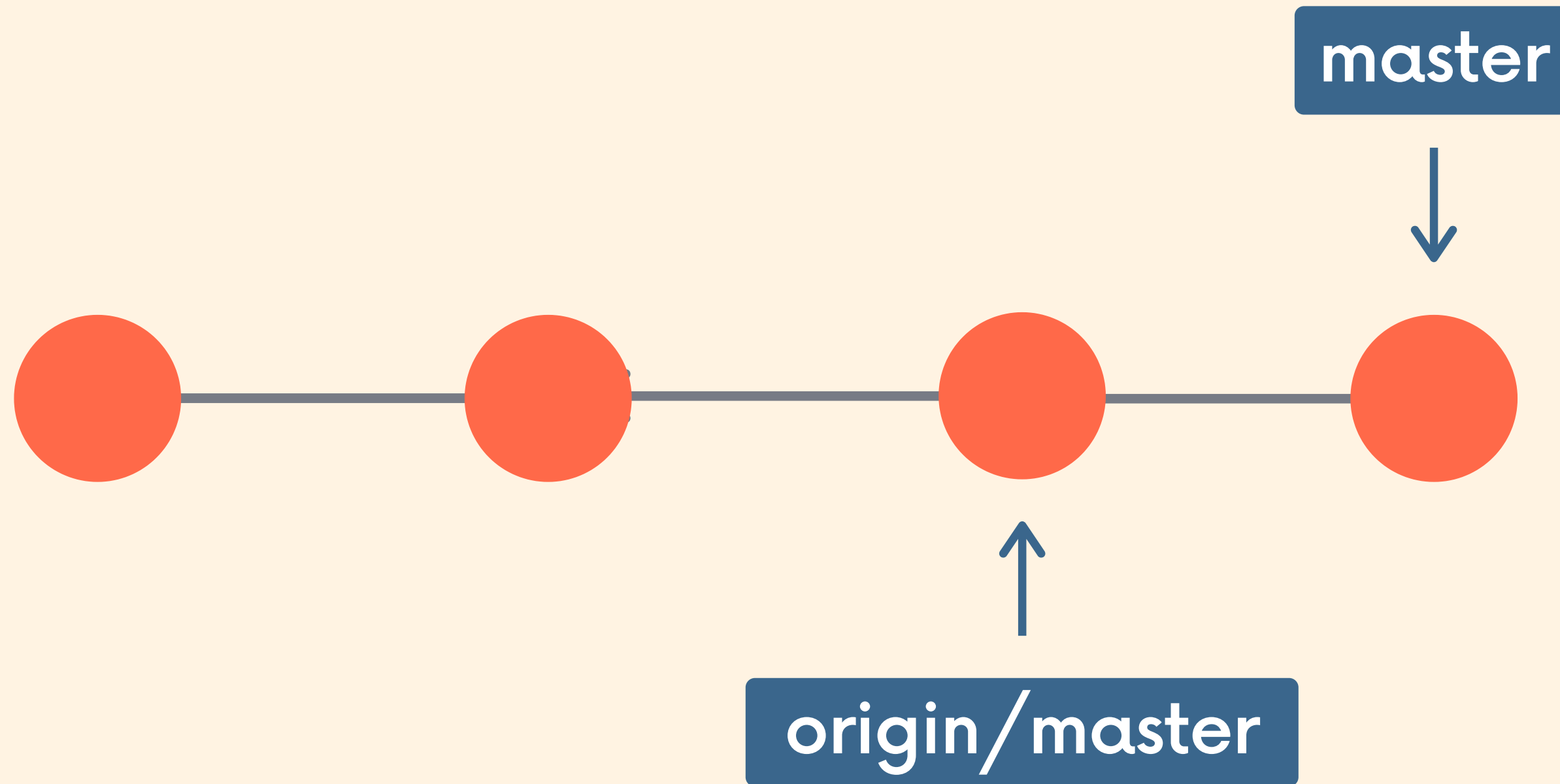


My Computer



My Computer

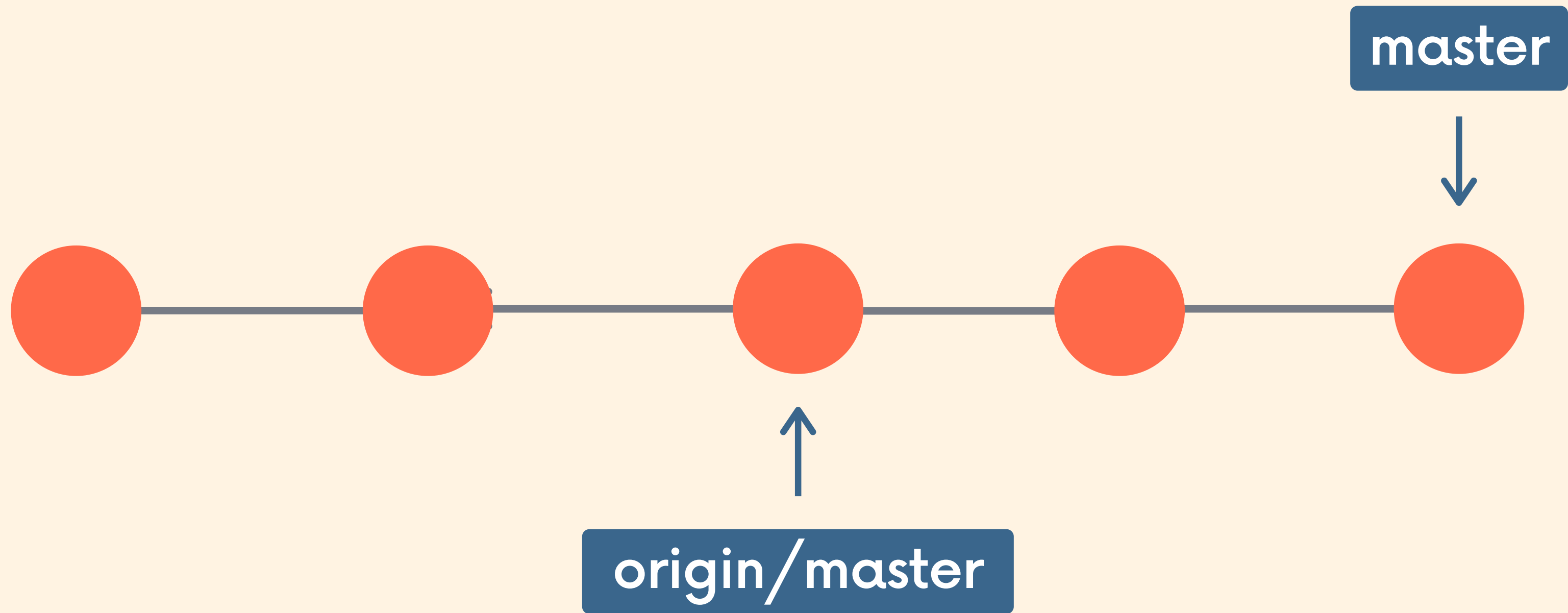
I make a new commit locally. My master branch reference updates, like always.



The remote reference stays the same

My Computer

I make another commit, and the local branch reference moves again.



Remote reference doesn't move!

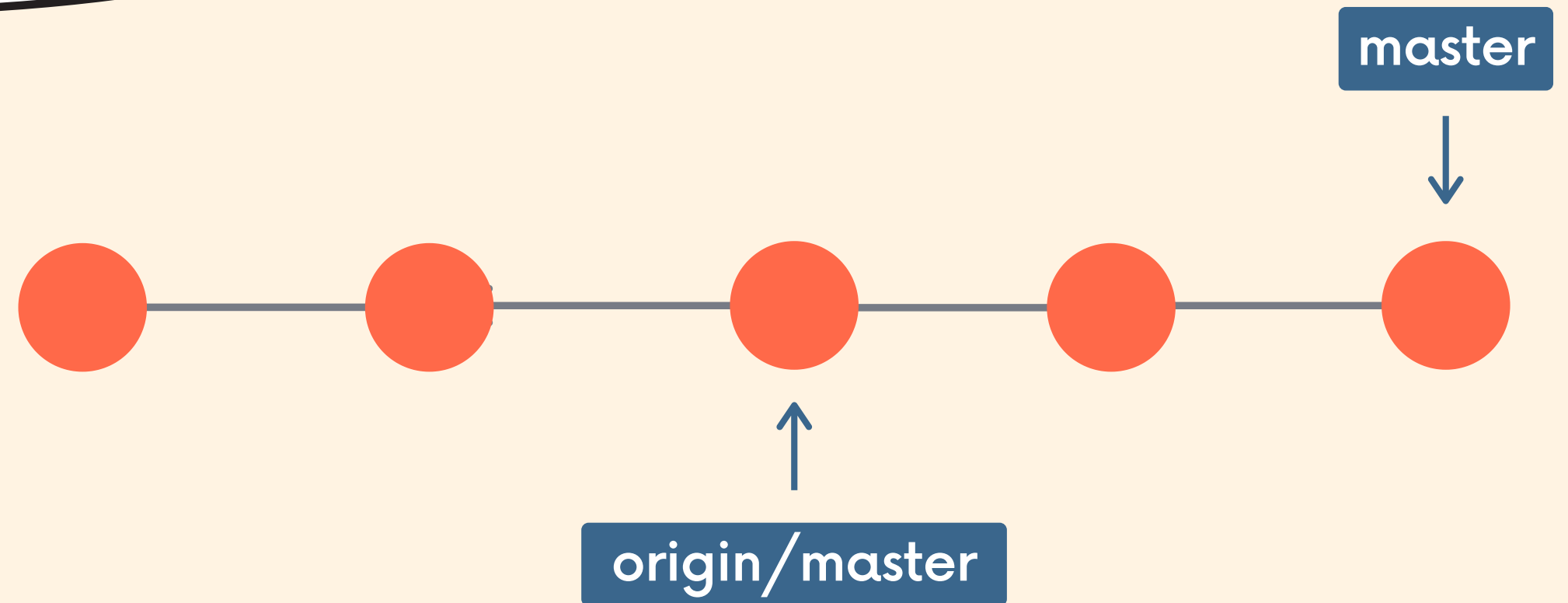


When I run `git status`

```
● ● ●  
>git status  
On branch master  
Your branch is ahead of 'origin/master' by 2  
commits.  
  (use "git push" to publish your local commits)
```



hmm...what did this project look like when I first cloned this repo?





You can checkout these remote branch pointers



```
>git checkout origin/master
```

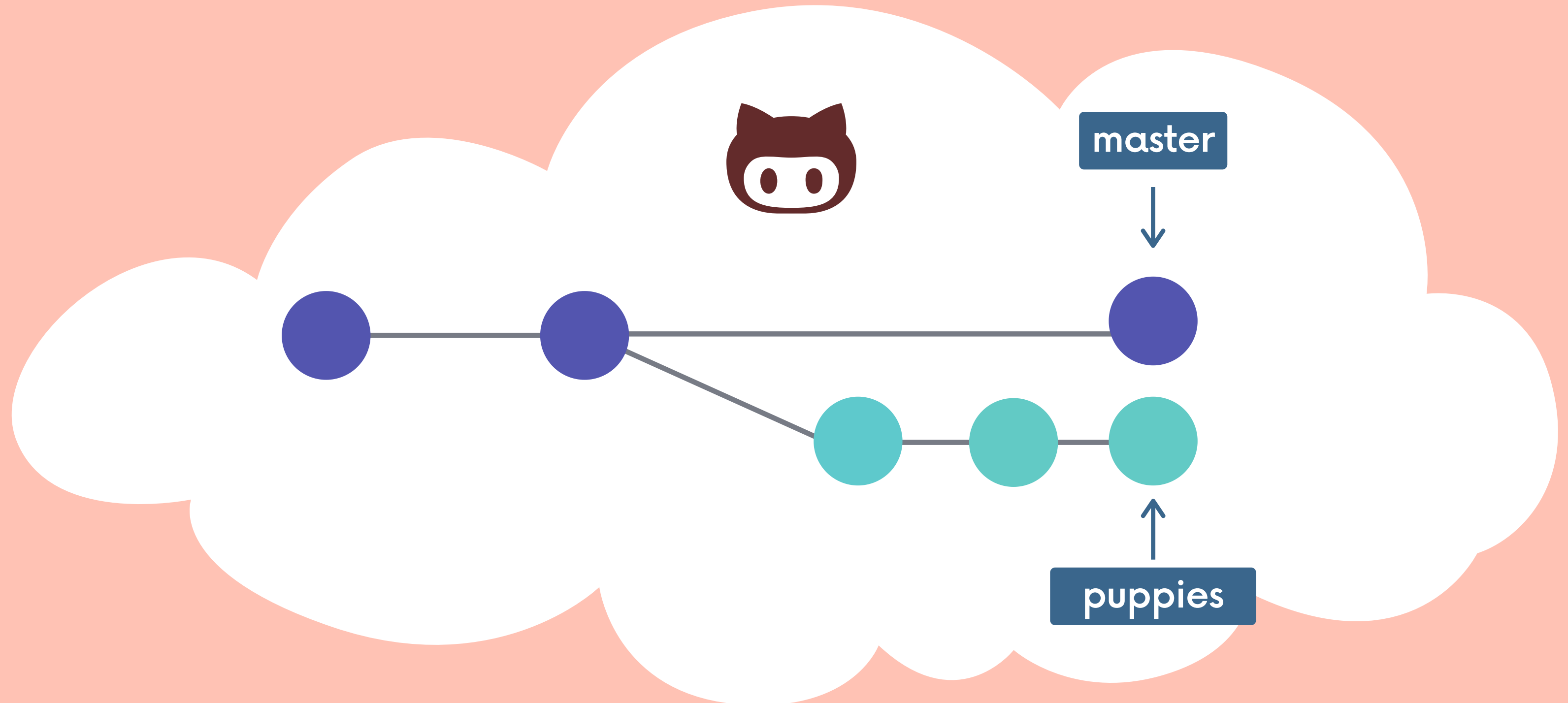
```
Note: switching to 'origin/master'.
```

```
You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this blah blah blah blah
```

Detached HEAD! Don't panic. It's fine.



Suppose I Just Cloned This Github Repo





Remote Branches

Once you've cloned a repository, we have all the data and Git history for the project at that moment in time. However, that does not mean it's all in my workspace!

The github repo has a branch called **puppies**, but when I run **git branch** I don't see it on my machine! All I see is the master branch. What's going on?



```
git branch
*master
```





Remote Branches

Run `git branch -r` to view the remote branches our local repository knows about.

```
git branch -r  
origin/master  
origin/puppies
```



Workspace

master

By default, my master branch is already tracking origin/master.

I didn't connect these myself!

Remote

origin/master

origin/puppies





I want to work on the puppies branch locally!

I could checkout `origin/puppies`, but that puts me in detached HEAD.

I want my own local branch called `puppies`, and I want it to be connected to `origin/puppies`, just like my local `master` branch is connected to `origin/master`.

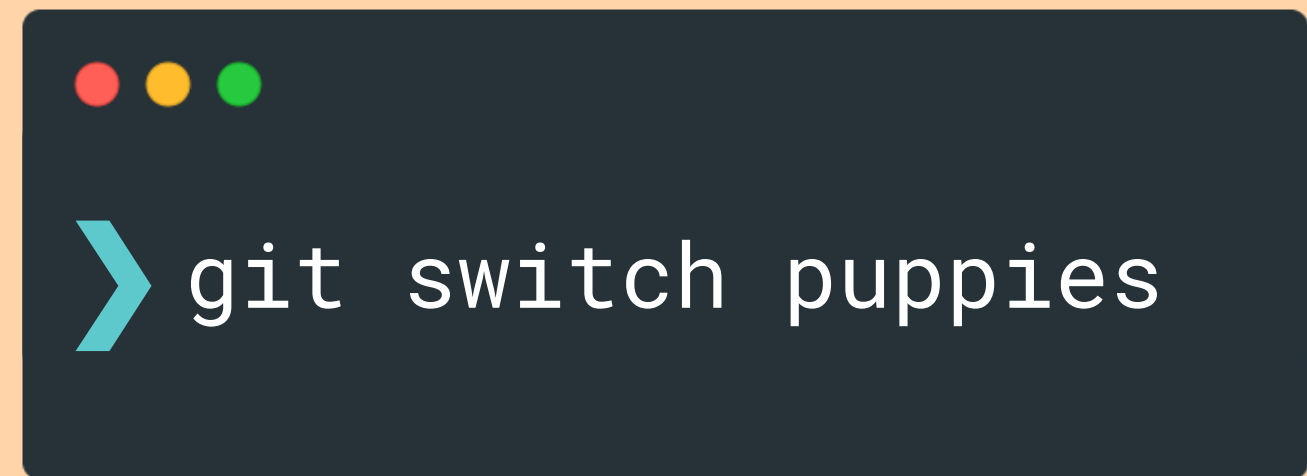




It's super easy!

Run `git switch <remote-branch-name>` to create a new local branch from the remote branch of the same name.

`git switch puppies` makes me a local puppies branch AND sets it up to track the remote branch `origin/puppies`.



```
> git switch puppies
```





```
> git switch puppies
```

```
Branch 'puppies' set up to track remote  
branch 'puppies' from 'origin'.  
Switched to a new branch 'puppies'
```

Workspace

master

puppies

Remote

origin/master

origin/puppies





NOTE!

the new command **git switch** makes this super easy to do!
It used to be slightly more complicated using **git checkout**

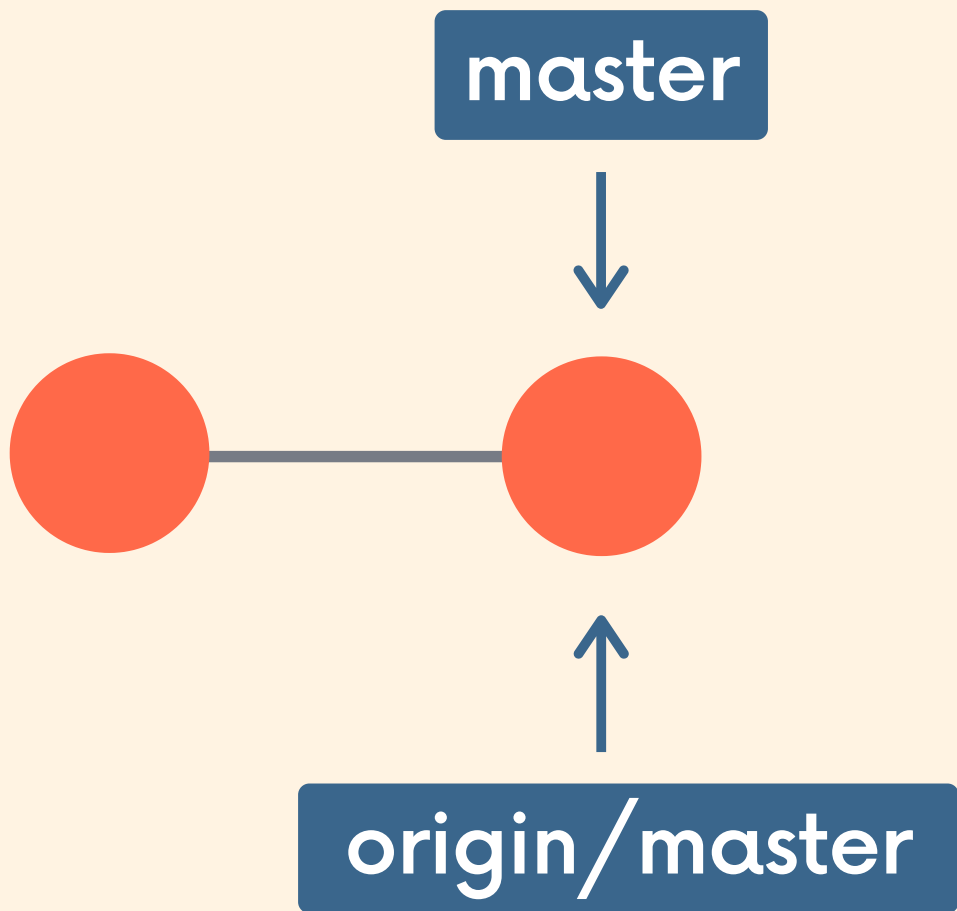
```
git checkout --track origin/puppies
```



Github



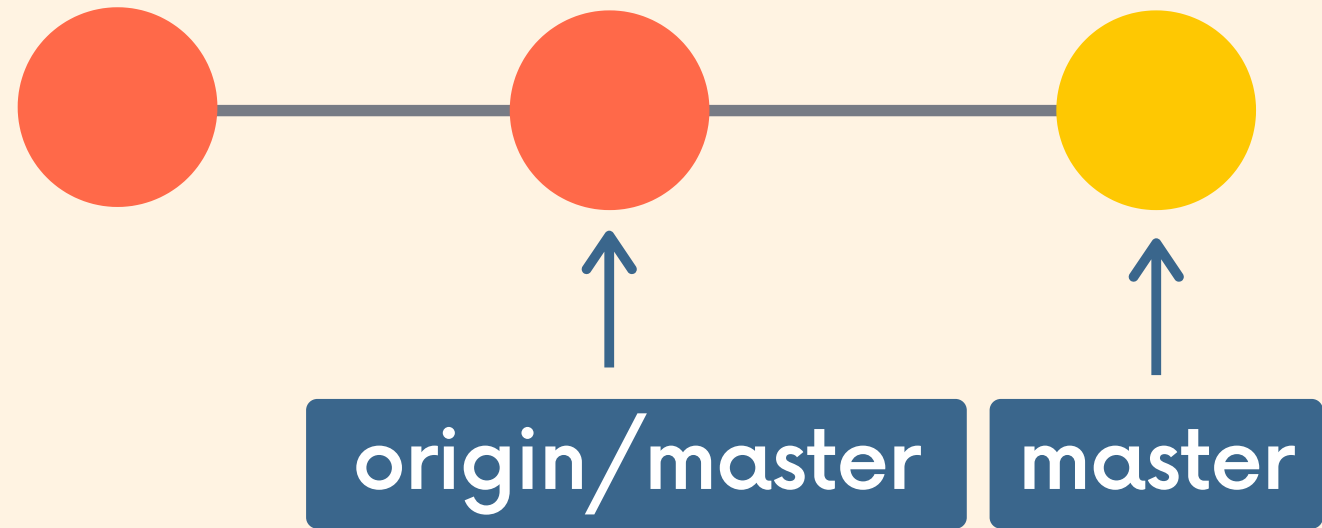
Local



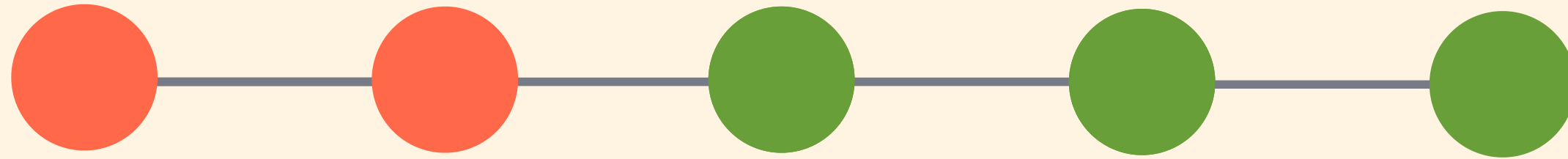
Github



Local

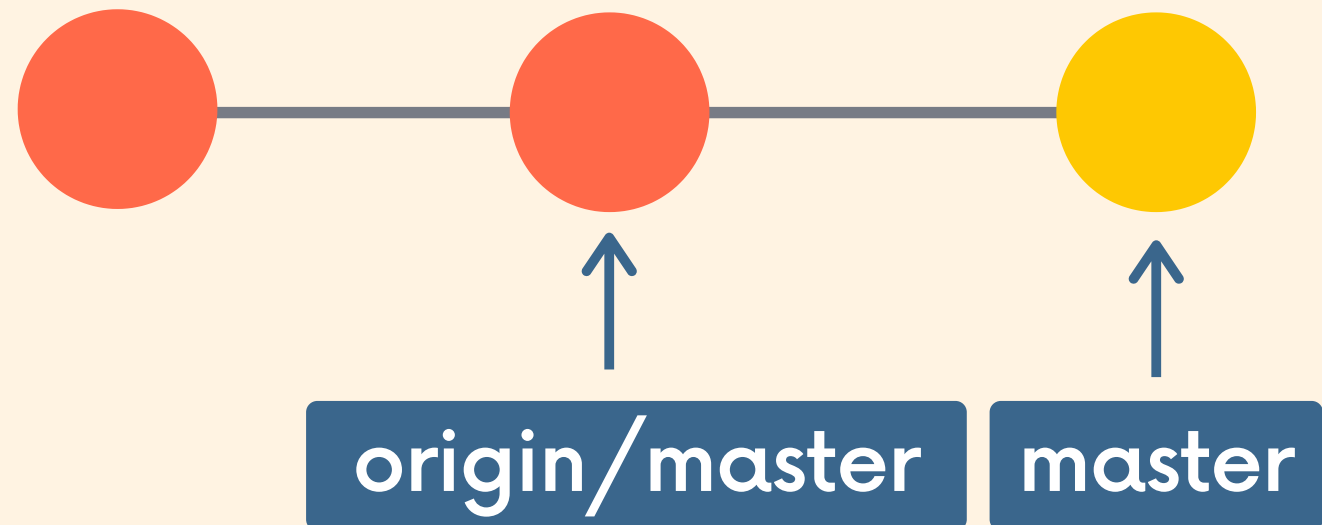


Github

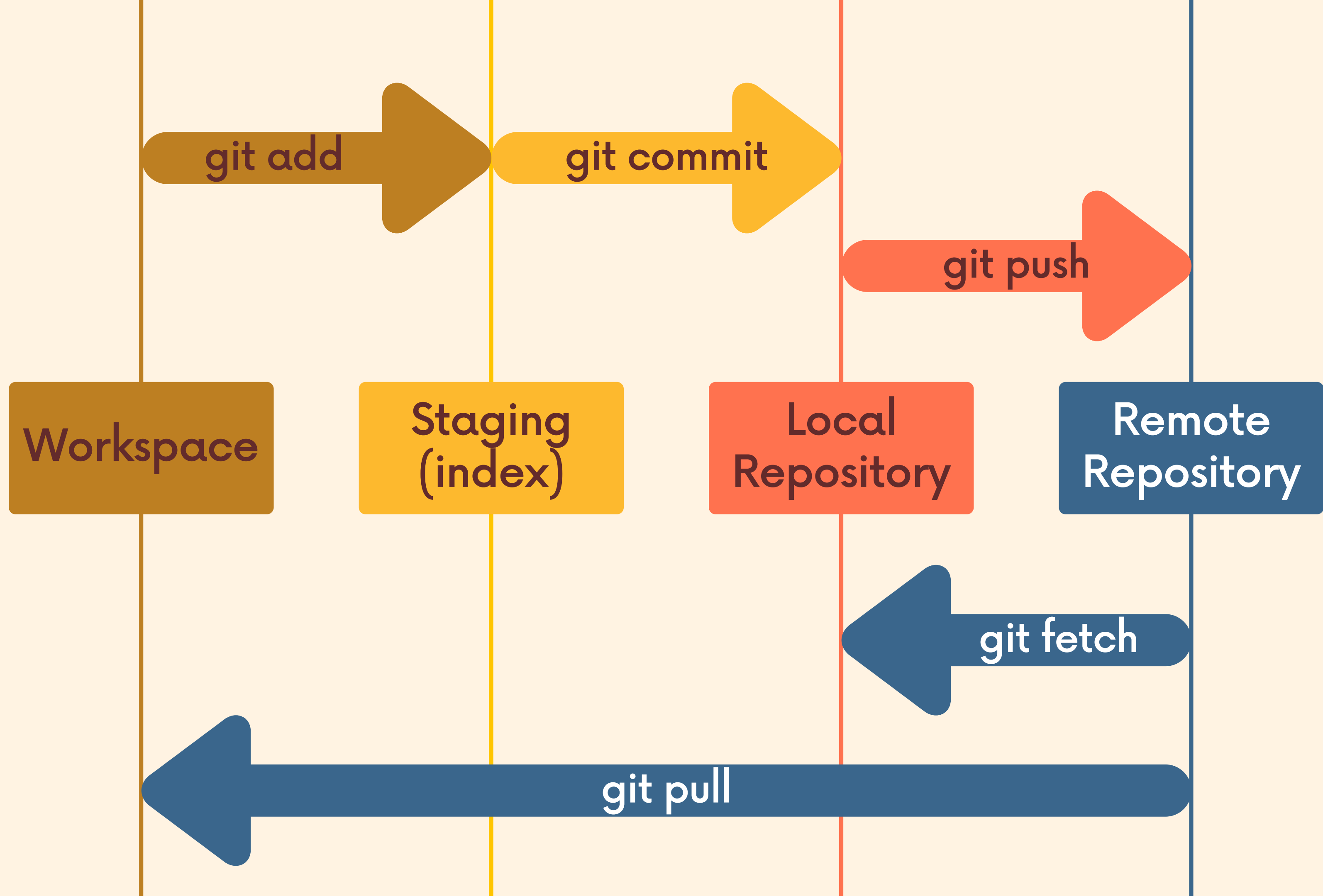


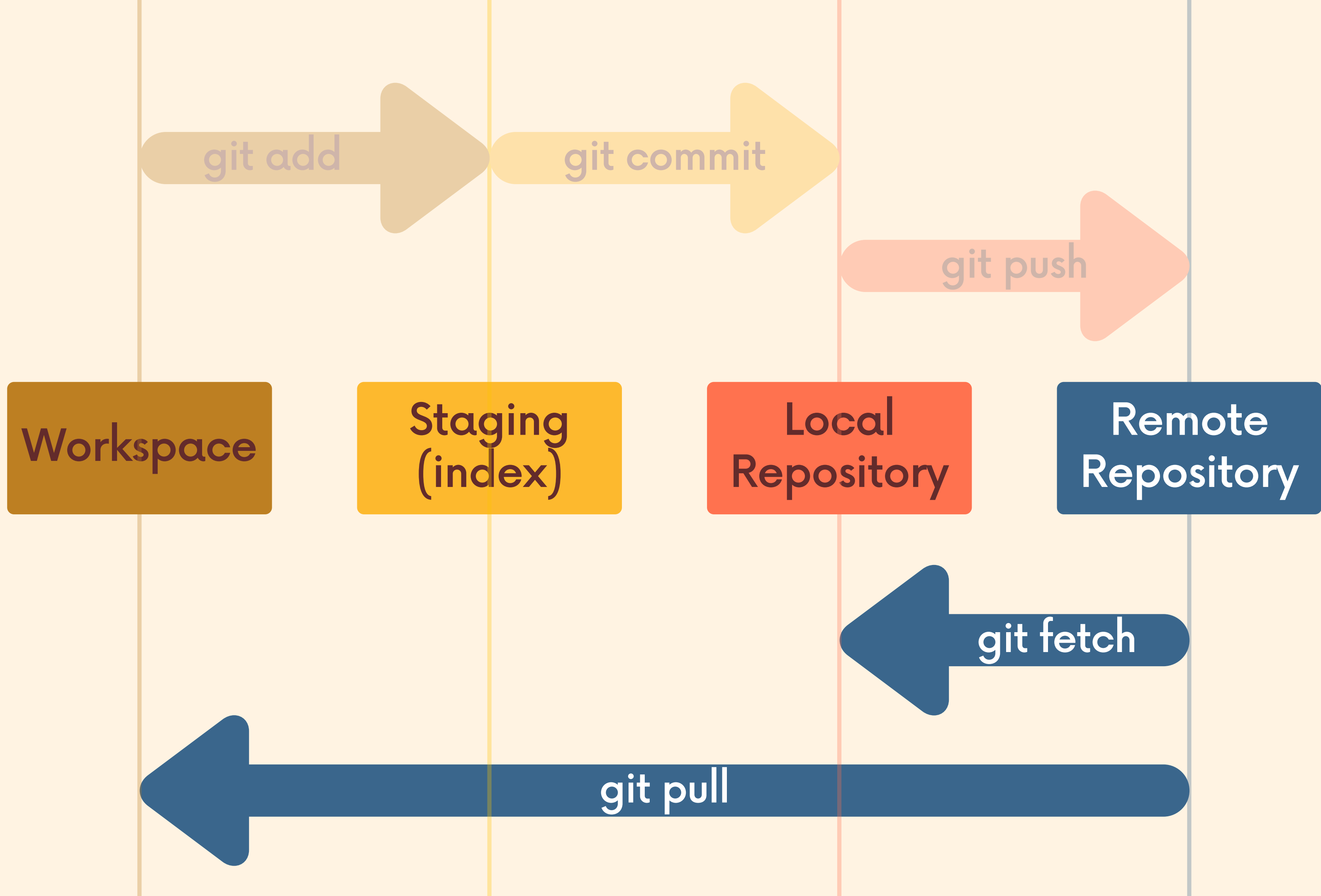
Uh oh! The remote repo has changed! A teammate has pushed up changes to the master branch, but my local repo doesn't know!

Local



How do I get those changes???







Fetching

Fetching allows us to download changes from a remote repository, BUT those changes will not be automatically integrated into our working files.

It lets you see what others have been working on, without having to merge those changes into your local repo.

Think of it as "please go and get the latest information from Github, but don't screw up my working directory."





Git Fetch

The `git fetch <remote>` command fetches branches and history from a specific remote repository. It only updates remote tracking branches.

`git fetch origin` would fetch all changes from the origin remote repository.

```
> git fetch <remote>
```

If not specified, `<remote>` defaults to origin

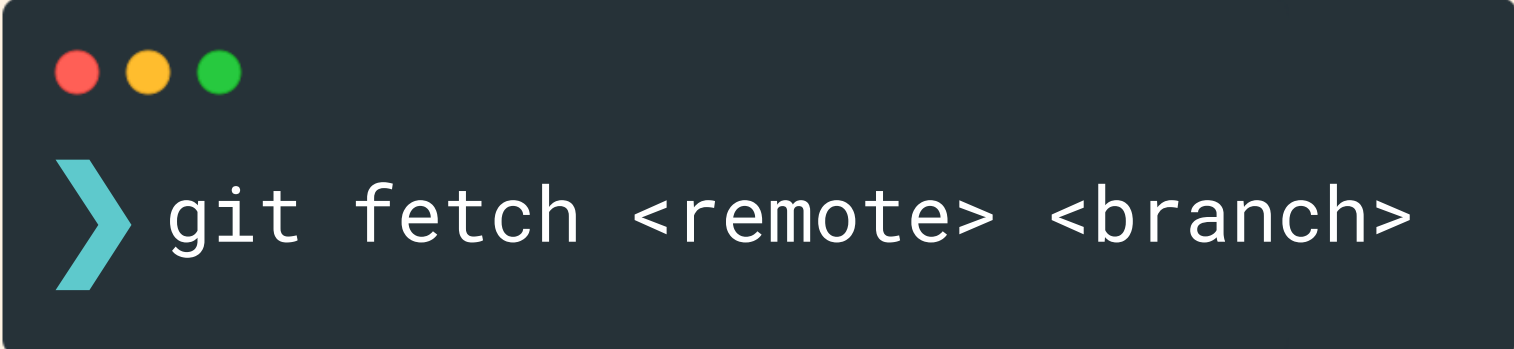




Git Fetch

We can also fetch a specific branch from a remote using `git fetch <remote> <branch>`

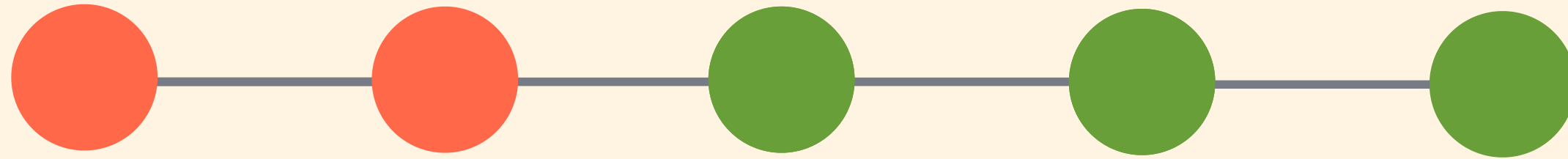
For example, `git fetch origin master` would retrieve the latest information from the master branch on the origin remote repository.



```
git fetch <remote> <branch>
```

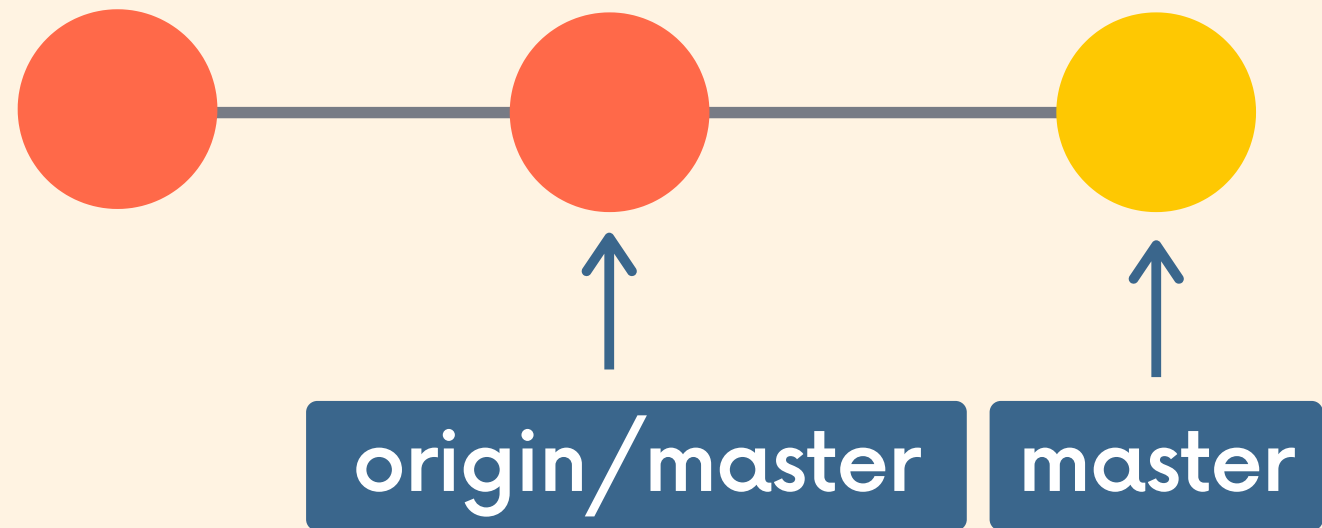


Github



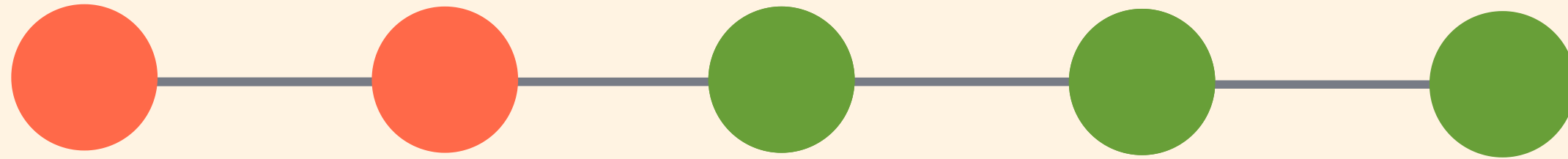
Uh oh! The remote repo has changed! A teammate has pushed up changes to the master branch, but my local repo doesn't know!

Local

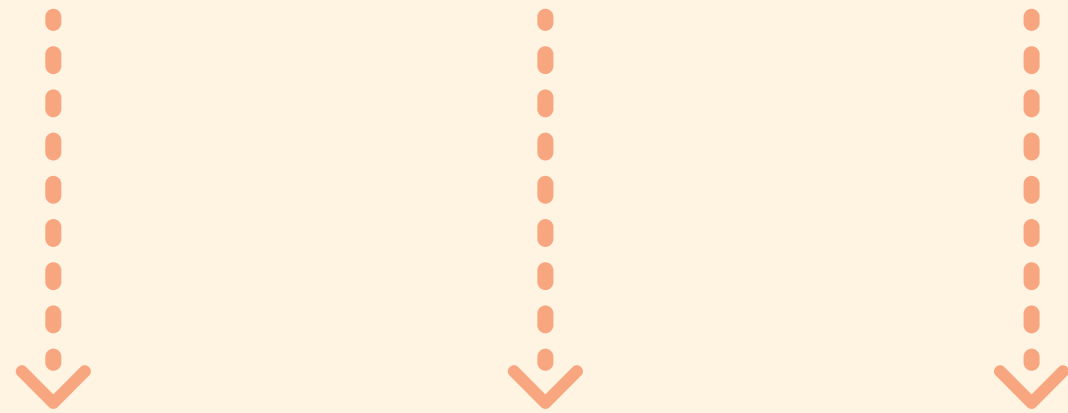


How do I get those changes???

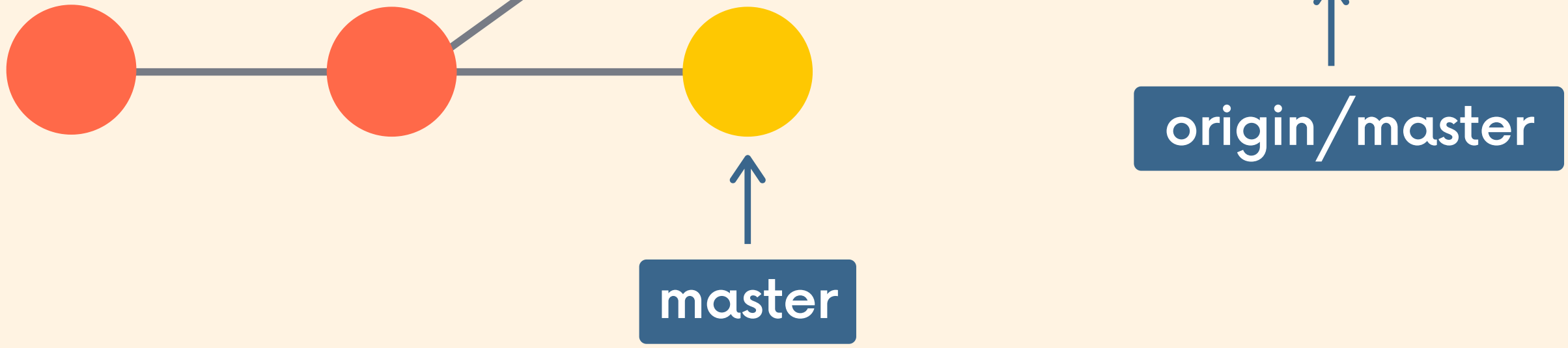
Github



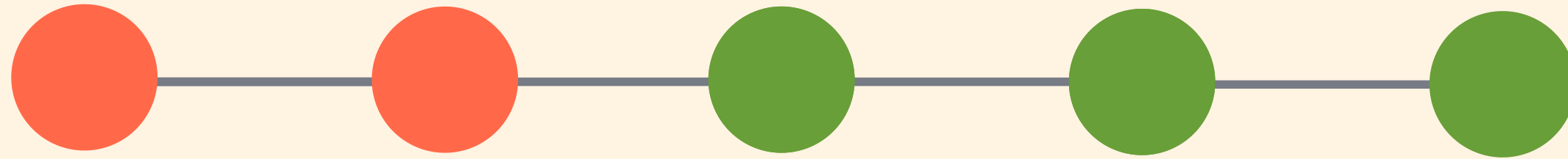
```
> git fetch origin master
```



Local

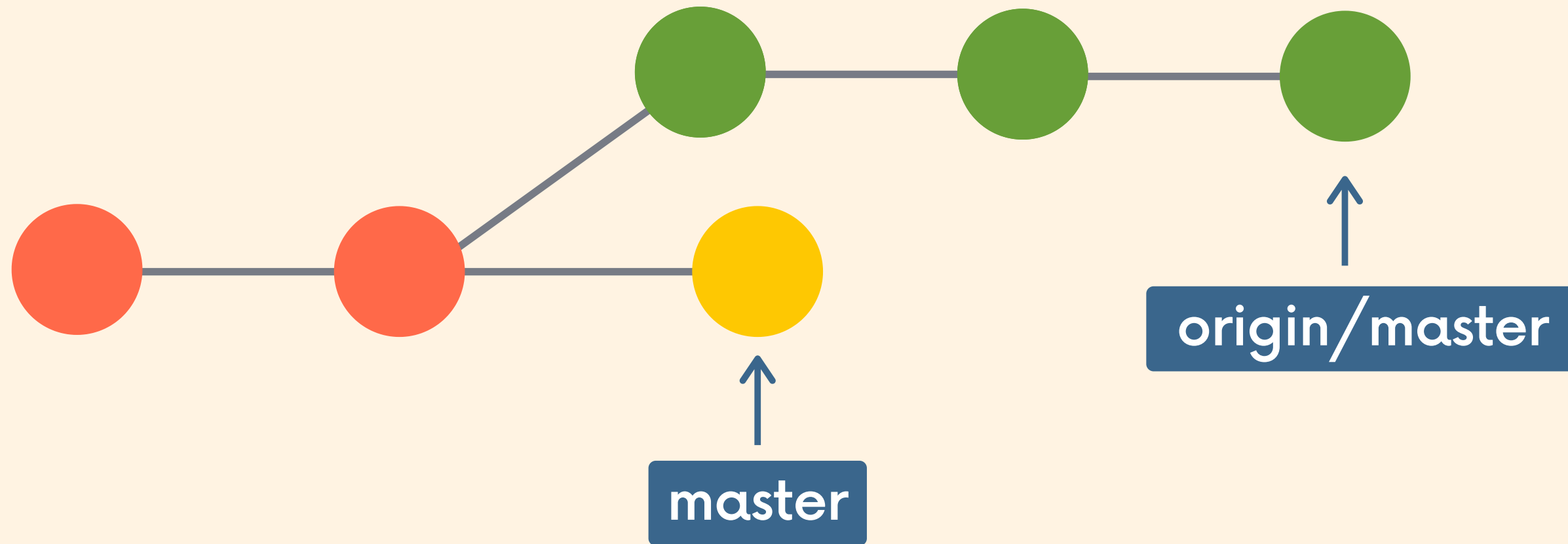


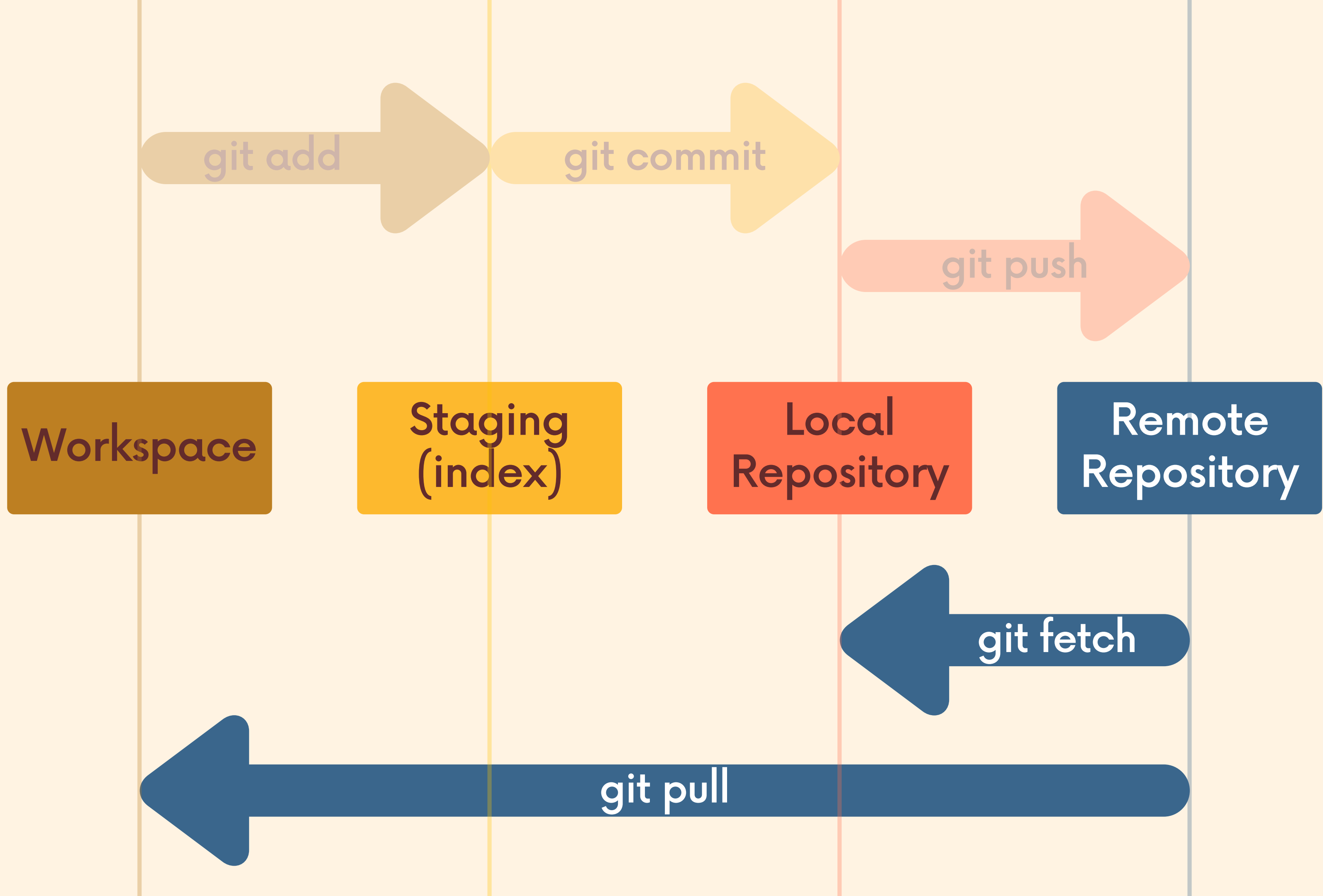
Github



I now have those changes on my machine, but if I want to see them I have to **checkout origin/master**. My master branch is untouched!

Local







Pulling

`git pull` is another command we can use to retrieve changes from a remote repository. Unlike `fetch`, `pull` actually updates our HEAD branch with whatever changes are retrieved from the remote.

"go and download data from Github AND immediately update my local repo with those changes"



git pull = git fetch + git merge

update the remote tracking branch
with the latest changes from the
remote repository


update my current branch with
whatever changes are on the remote
tracking branch



git pull

To pull, we specify the particular remote and branch we want to pull using **git pull <remote> <branch>**. Just like with git merge, it matters WHERE we run this command from. Whatever branch we run it from is where the changes will be merged into.

git pull origin master would fetch the latest information from the origin's master branch and merge those changes into our current branch.

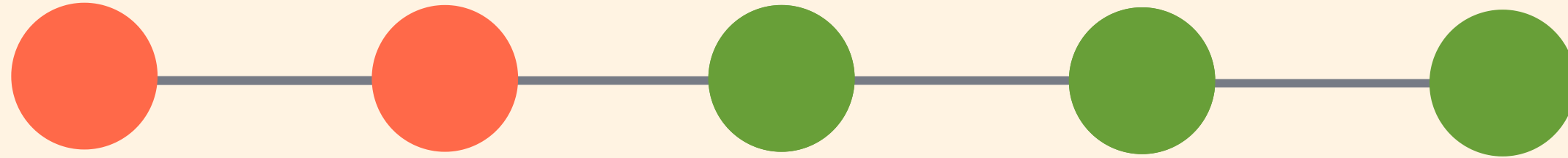


```
> git pull <remote> <branch>
```

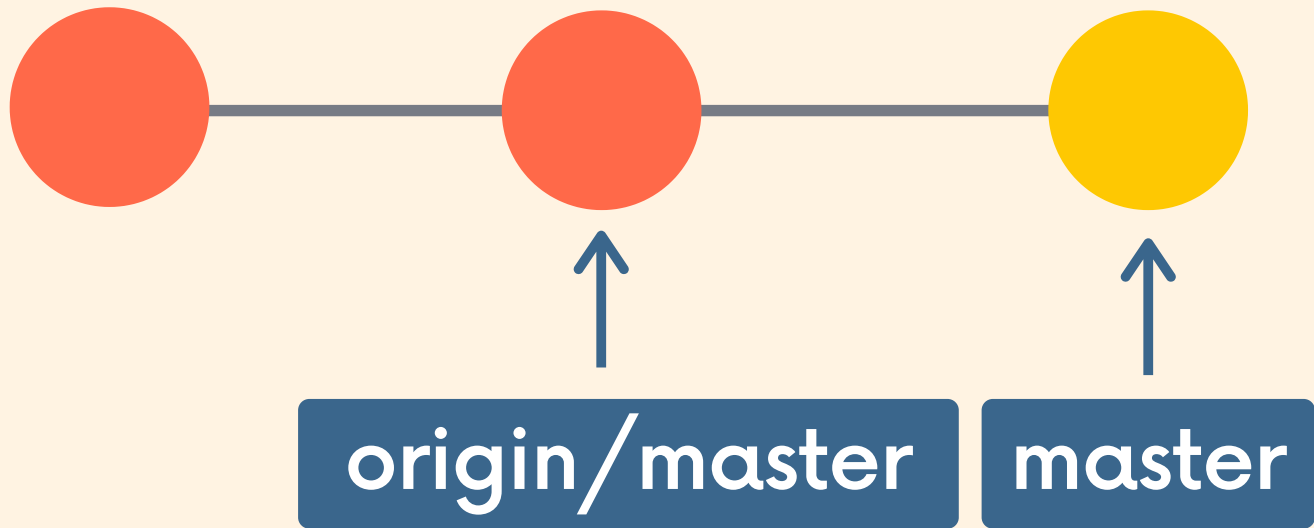


**pulls can result in
merge conflicts!!**

Github



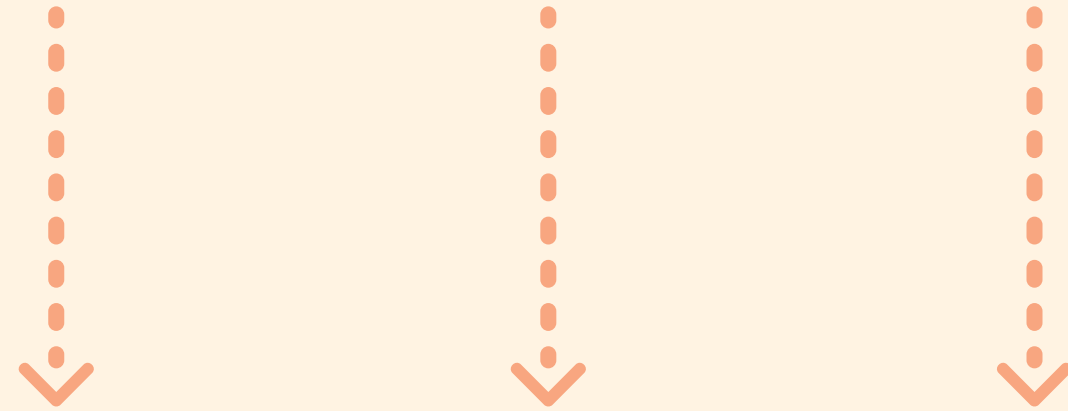
Local



Github



```
git pull origin master
```



origin/master



Local



master

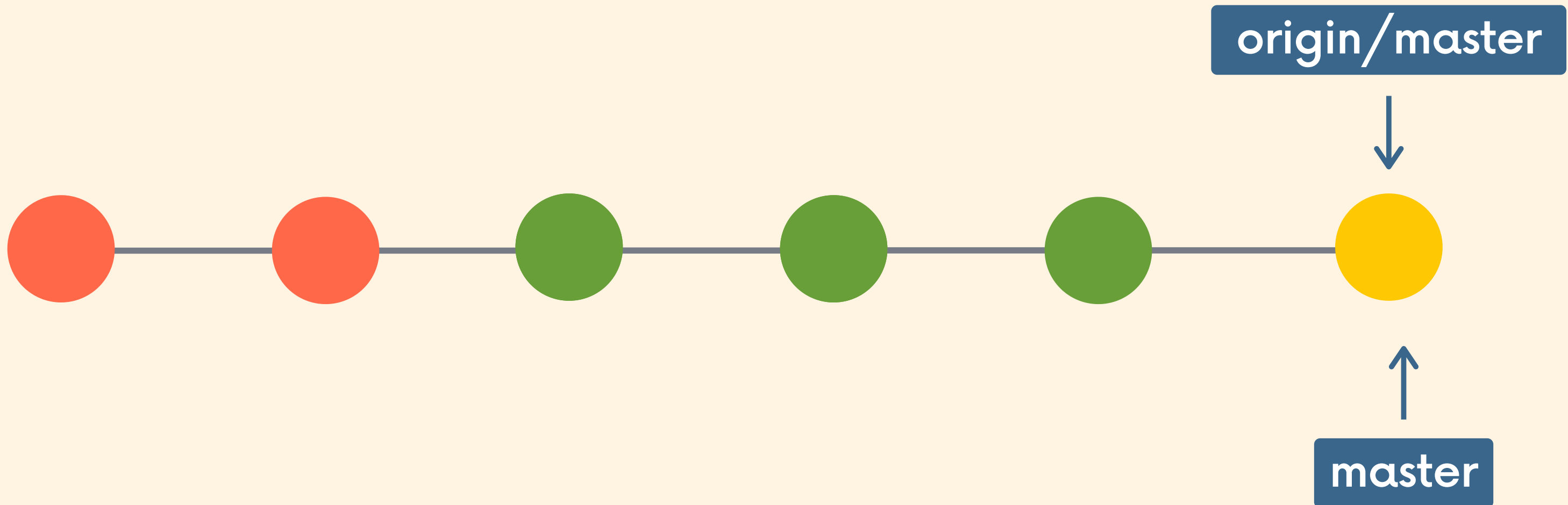


Github



I now have the latest commits from origin/master on my local master branch (assuming I pulled while on my master branch)

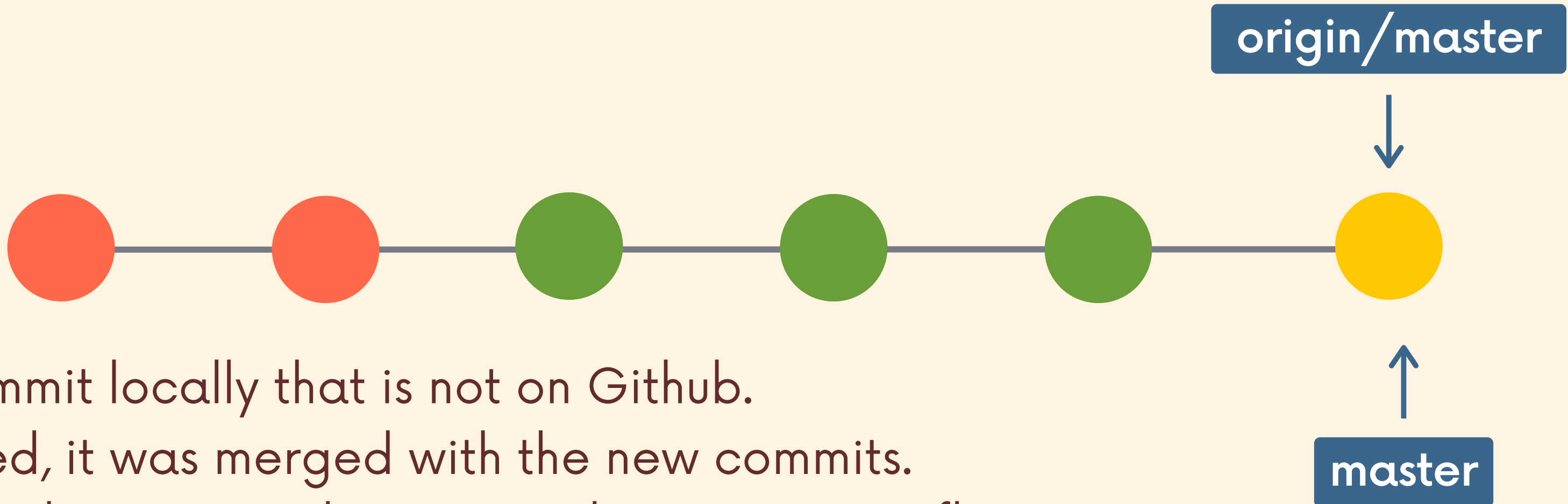
Local



Github



Local



I have a commit locally that is not on Github.
When I pulled, it was merged with the new commits.
As with any other merge, this can result in merge conflicts.



An even easier syntax!

If we run `git pull` without specifying a particular remote or branch to pull from, git assumes the following:

- remote will default to origin
- branch will default to whatever tracking connection is configured for your current branch.

Note: this behavior can be configured, and tracking connections can be changed manually. Most people dont mess with that stuff

```
> git pull
```



Workspace

Remote

master

origin/master

puppies

origin/puppies

When I'm on my local master branch...

```
> git pull
```

pulls from origin/master automatically

Workspace

master

puppies

Remote

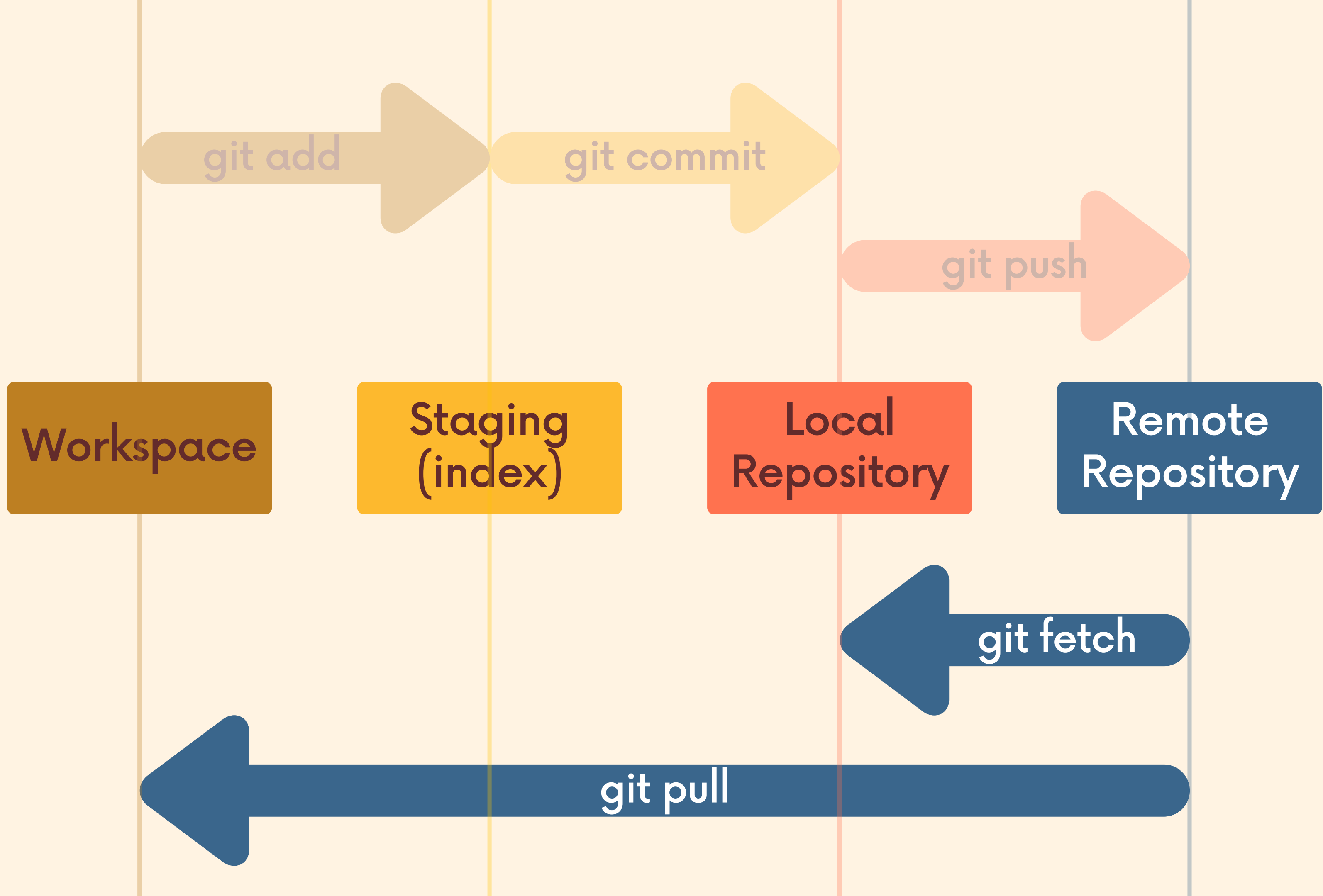
origin/master

origin/puppies

When I'm on my local puppies branch...

```
> git pull
```

pulls from origin/puppies automatically





git fetch

- Gets changes from remote branch(es)
- Updates the remote-tracking branches with the new changes
- Does not merge changes onto your current HEAD branch
- Safe to do at anytime

git pull

- Gets changes from remote branch(es)
- Updates the current branch with the new changes, merging them in
- Can result in merge conflicts
- Not recommended if you have uncommitted changes!

