

# git workflows (for collaboration)



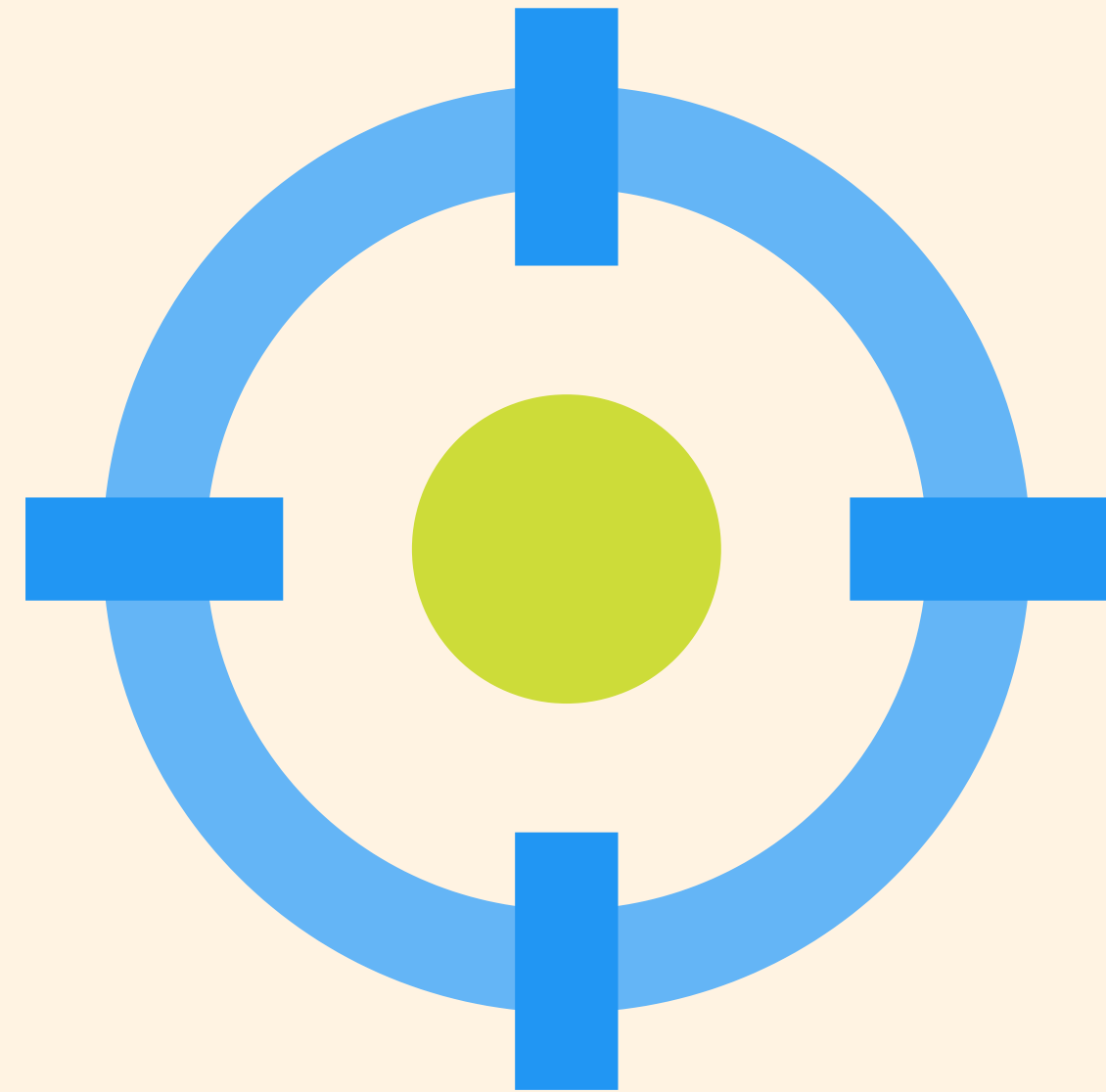


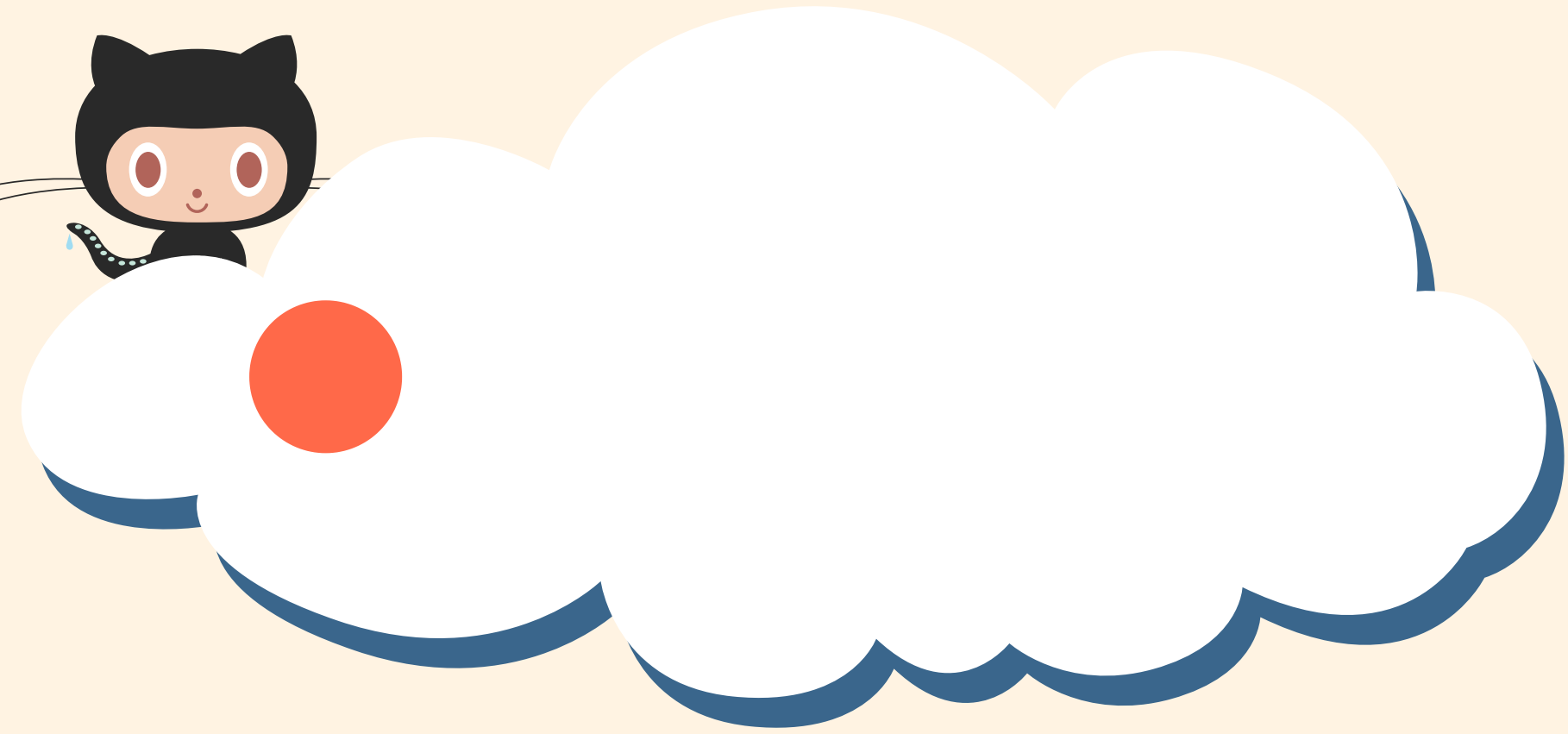
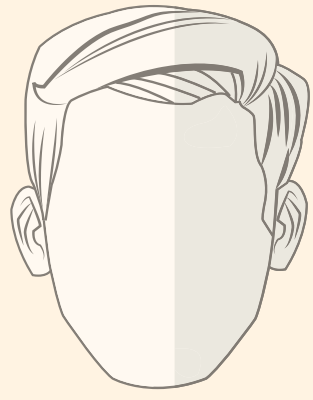
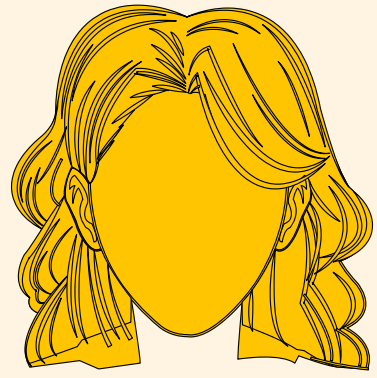
# Centralized Workflow

AKA Everyone Works On Master/Main  
AKA The Most Basic Workflow Possible

The simplest collaborative workflow is to have everyone work on the master branch (or main, or any other SINGLE branch).

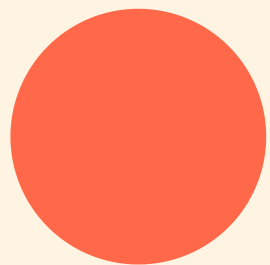
It's straightforward and can work for tiny teams, but it has quite a few shortcomings!

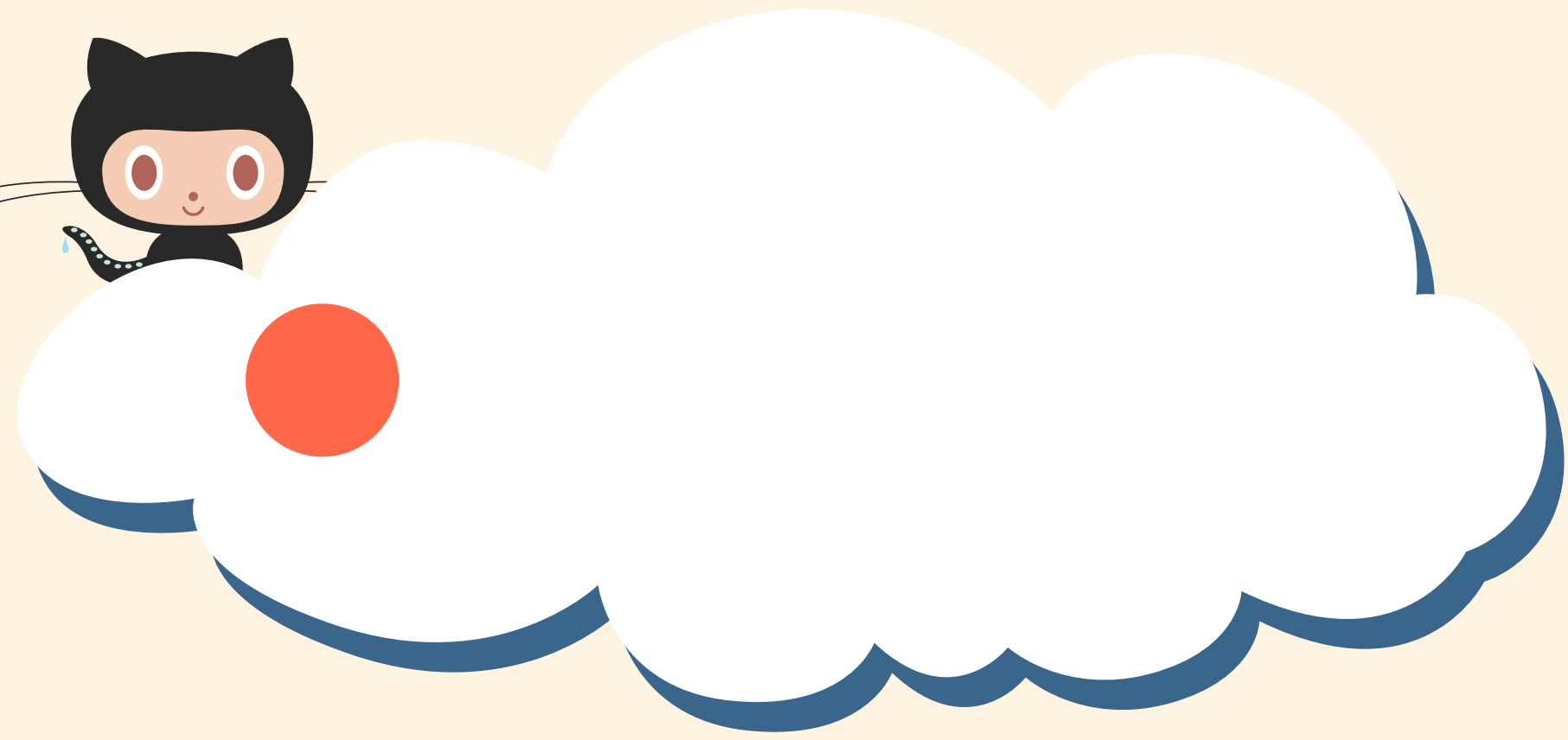
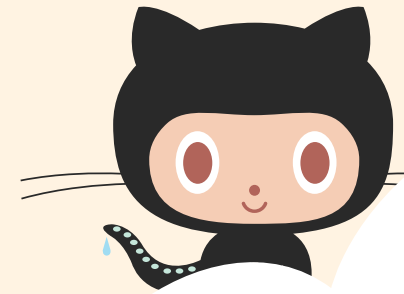
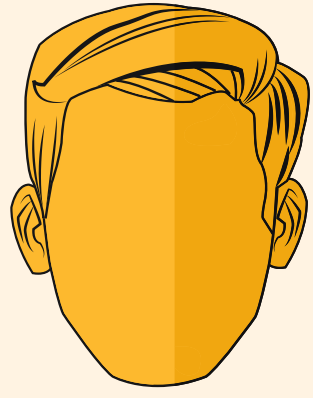
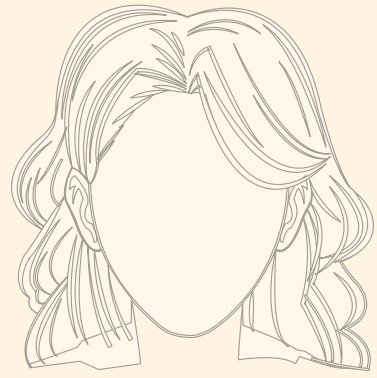




Pamela clones the repo

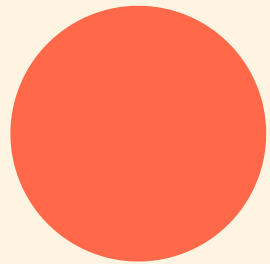
master



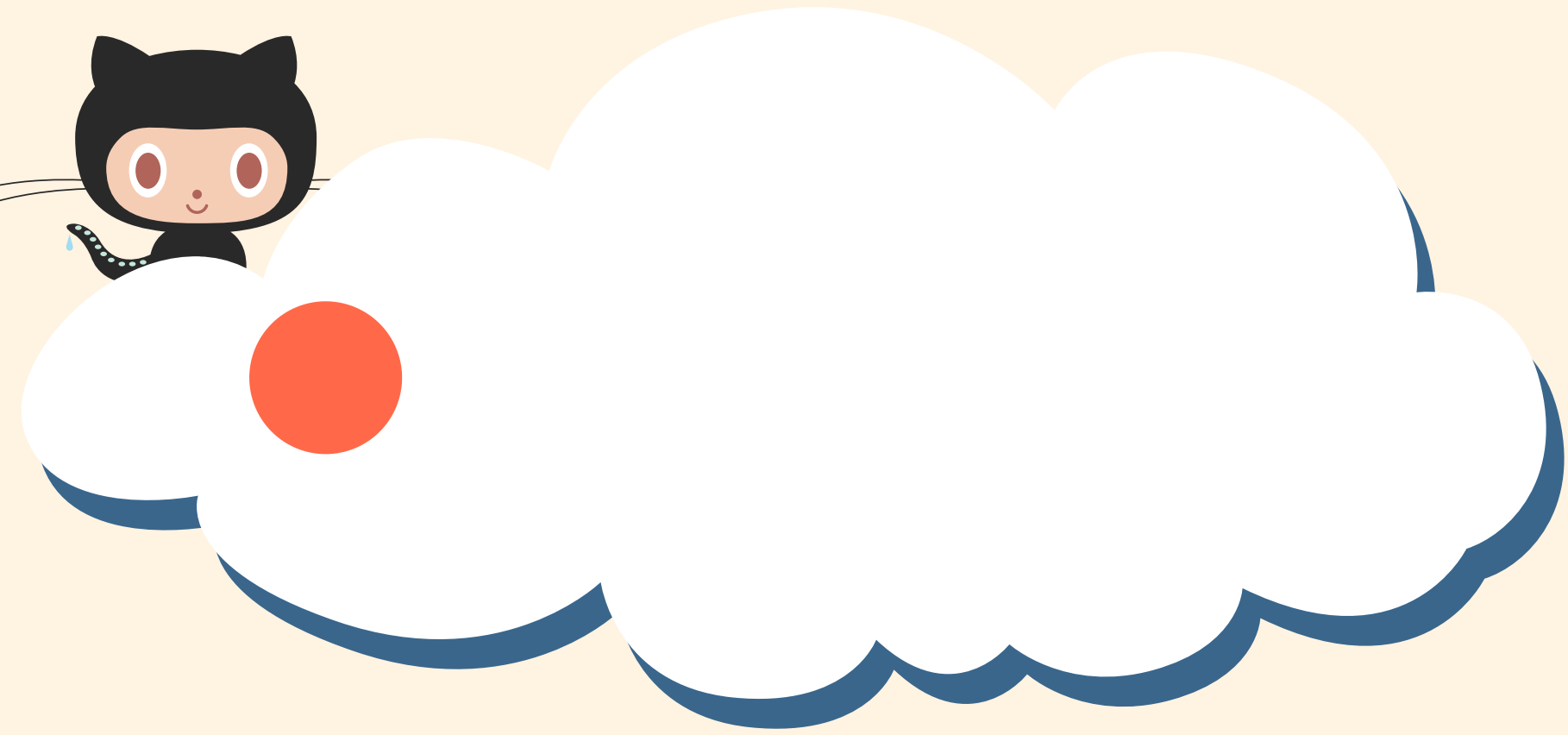
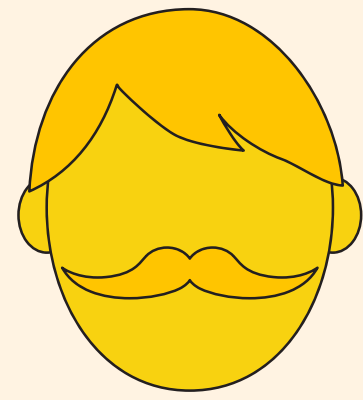
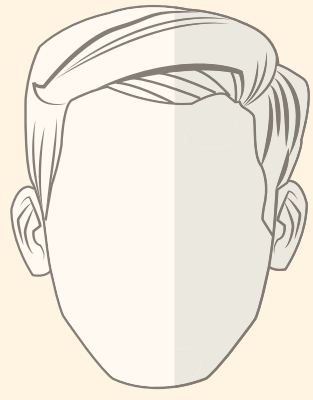
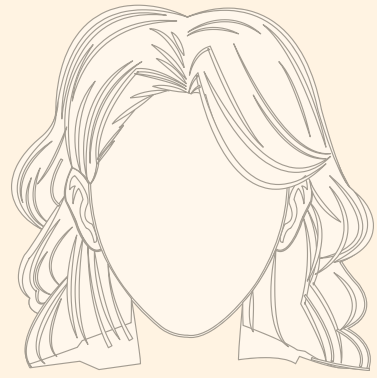


David clones the repo

master

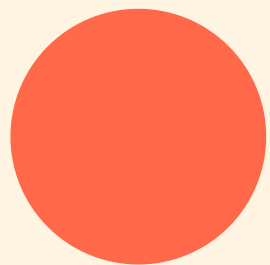


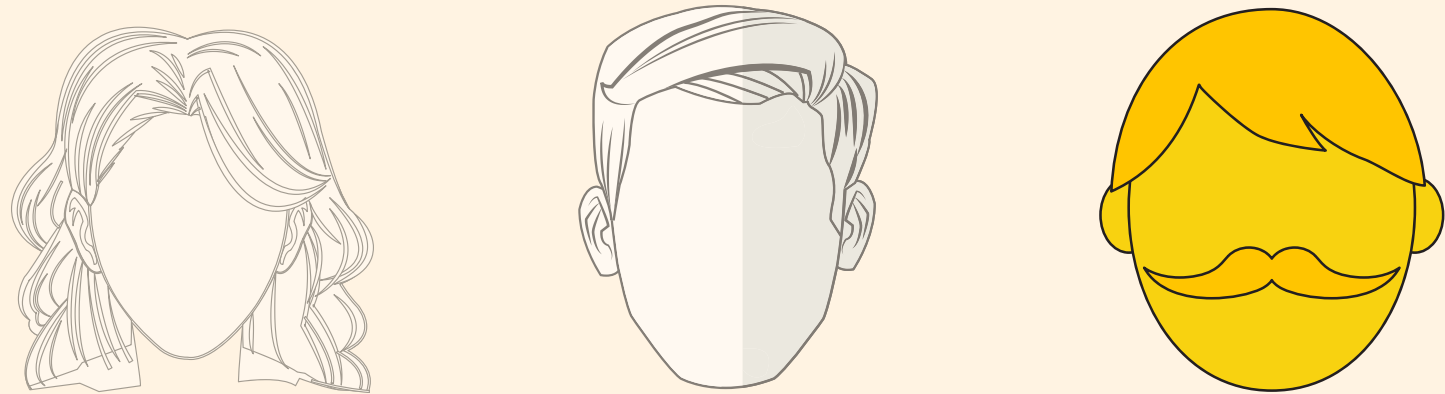




Forrest clones the repo

master

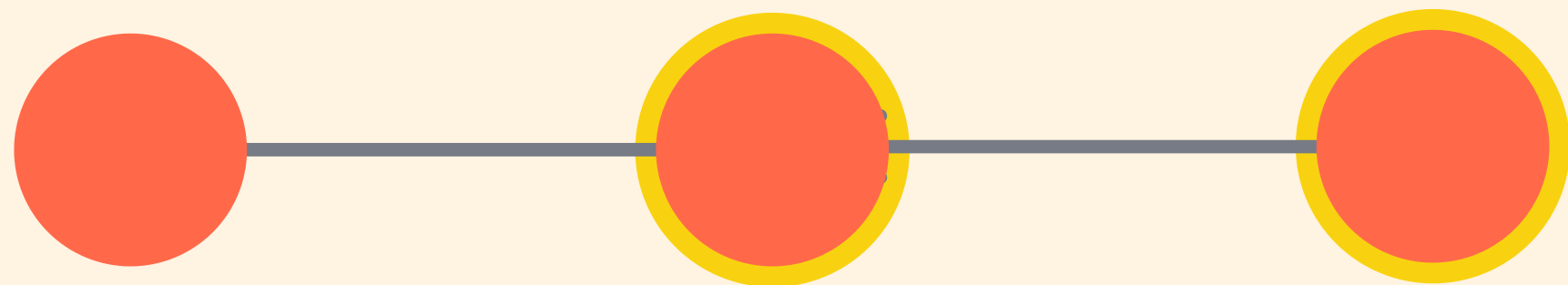


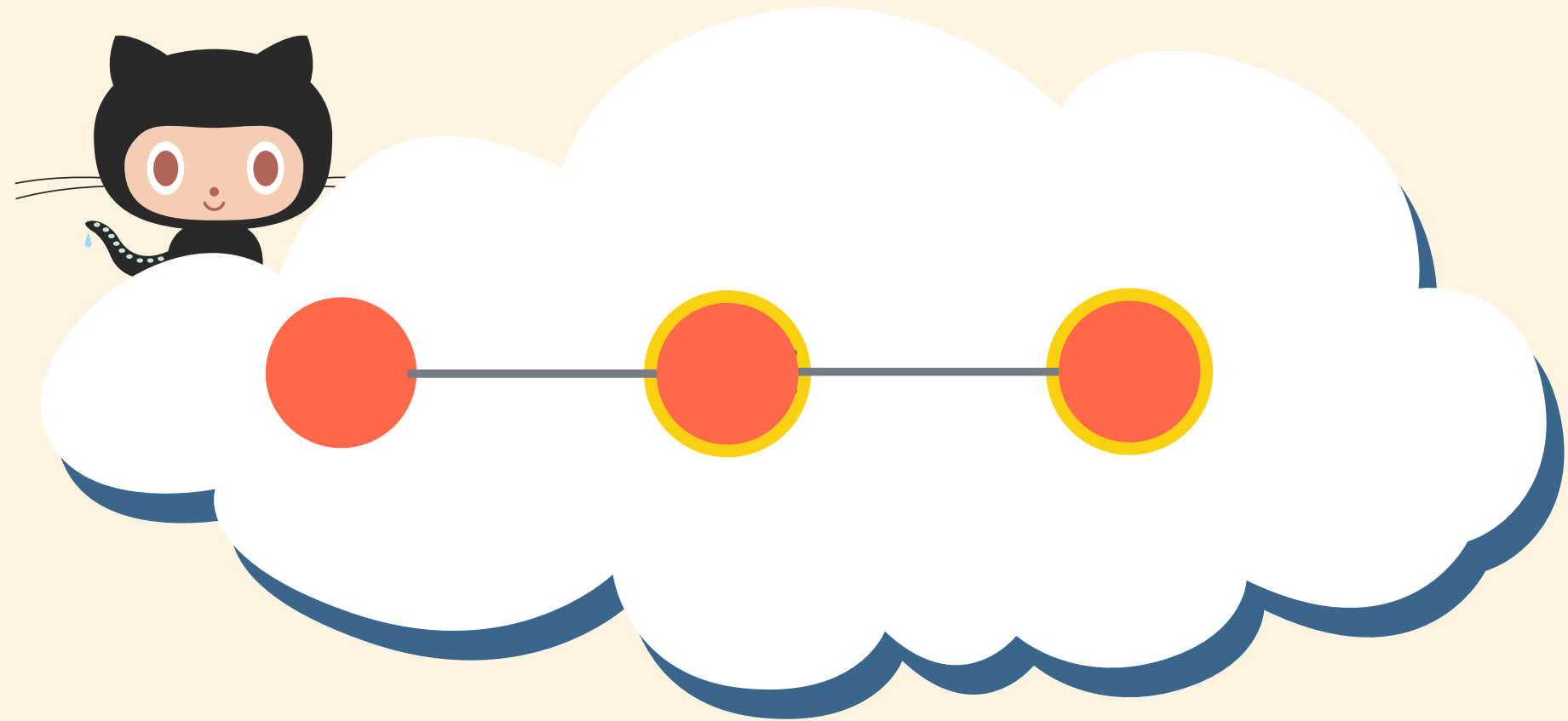
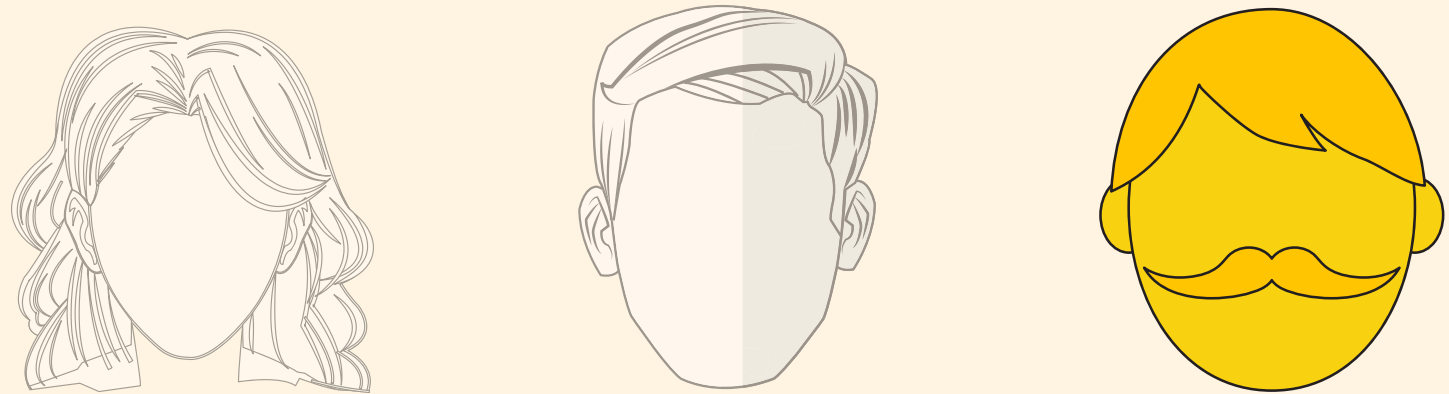


**Forrest** gets to work on a new feature! Adding and Committing all day long!

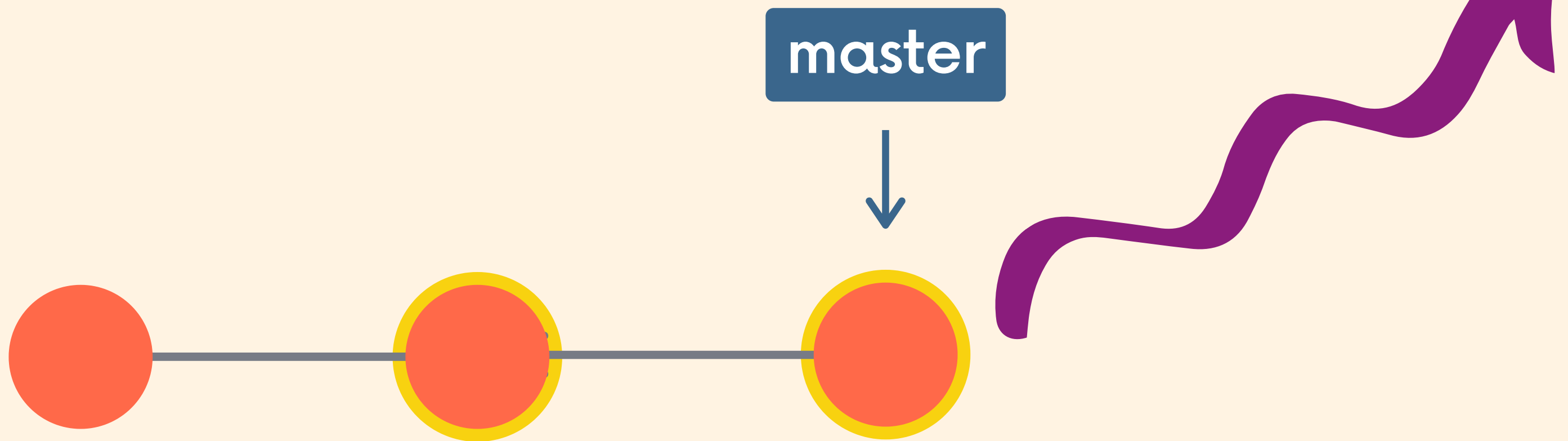


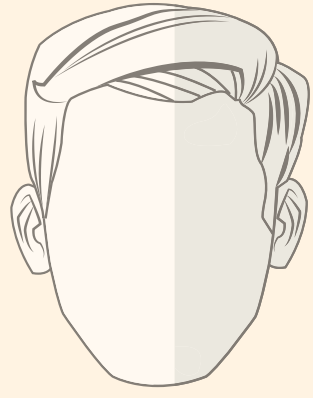
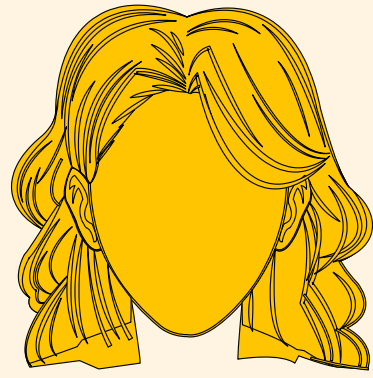
master



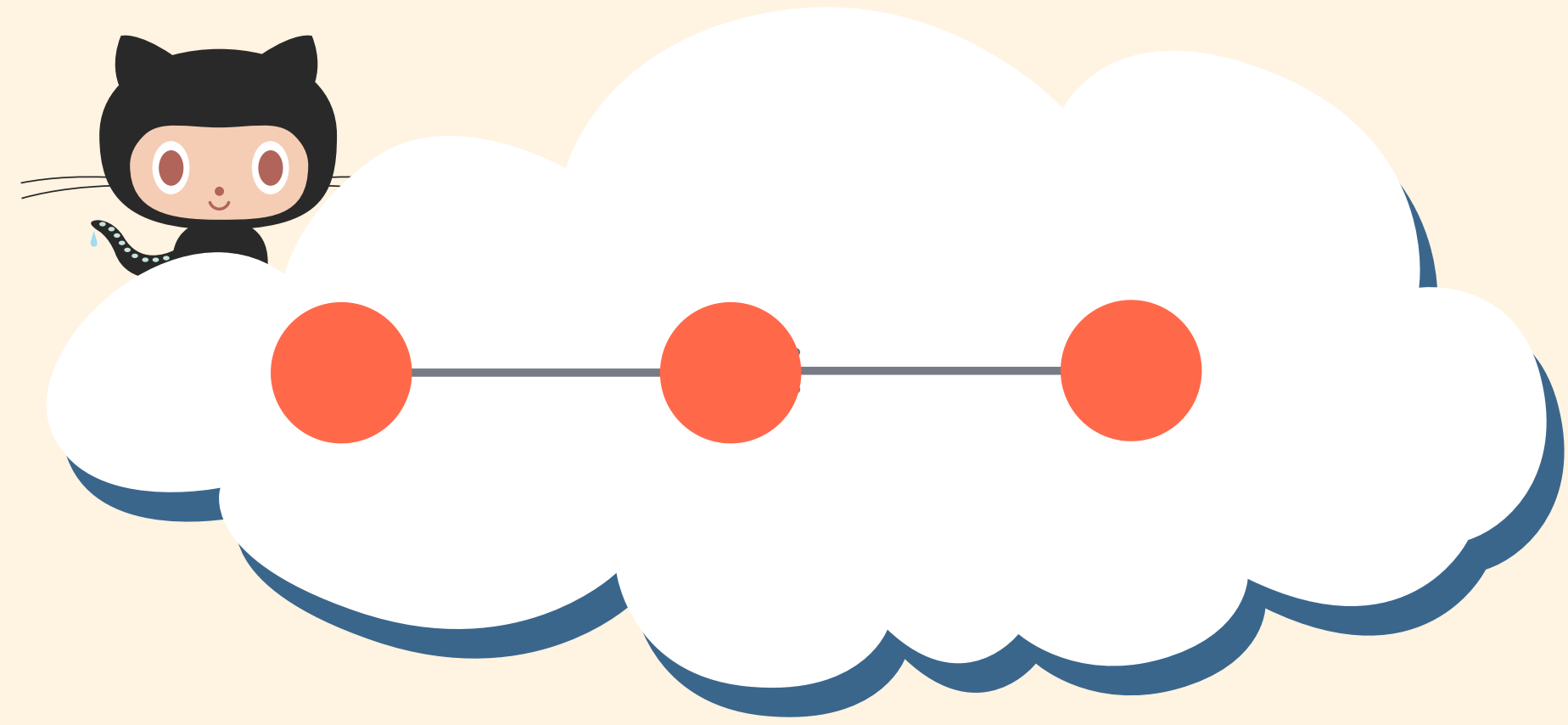


Forrest now pushes up his new work to Github



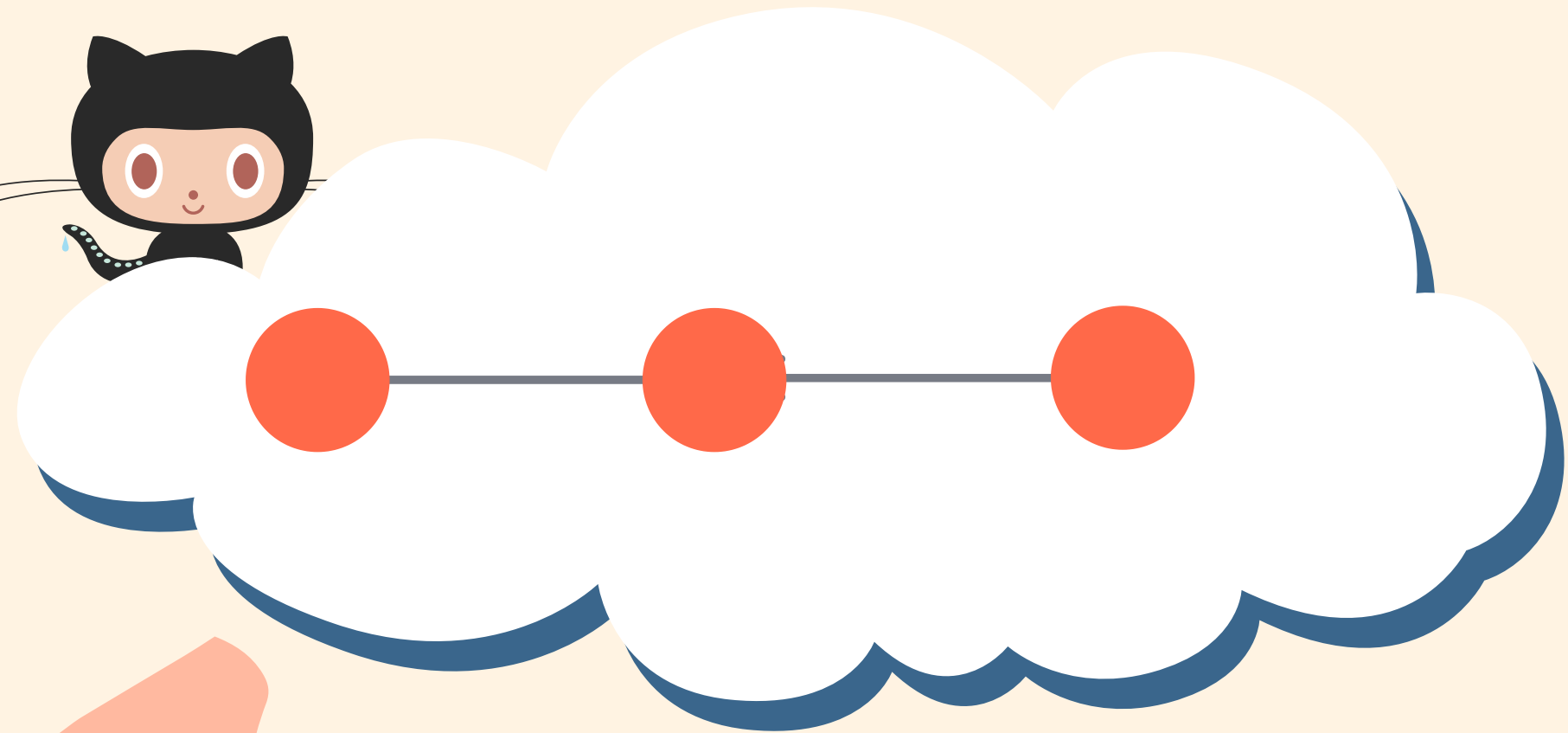
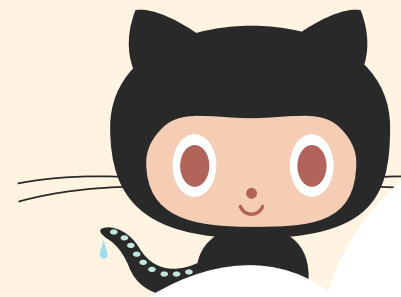
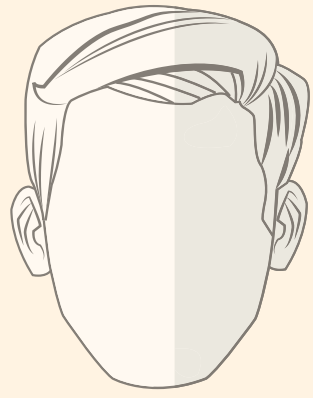
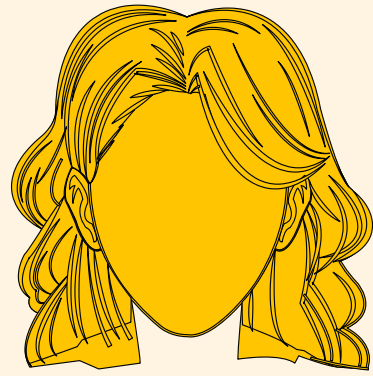


Pamela has also been hard at work on her own new feature.



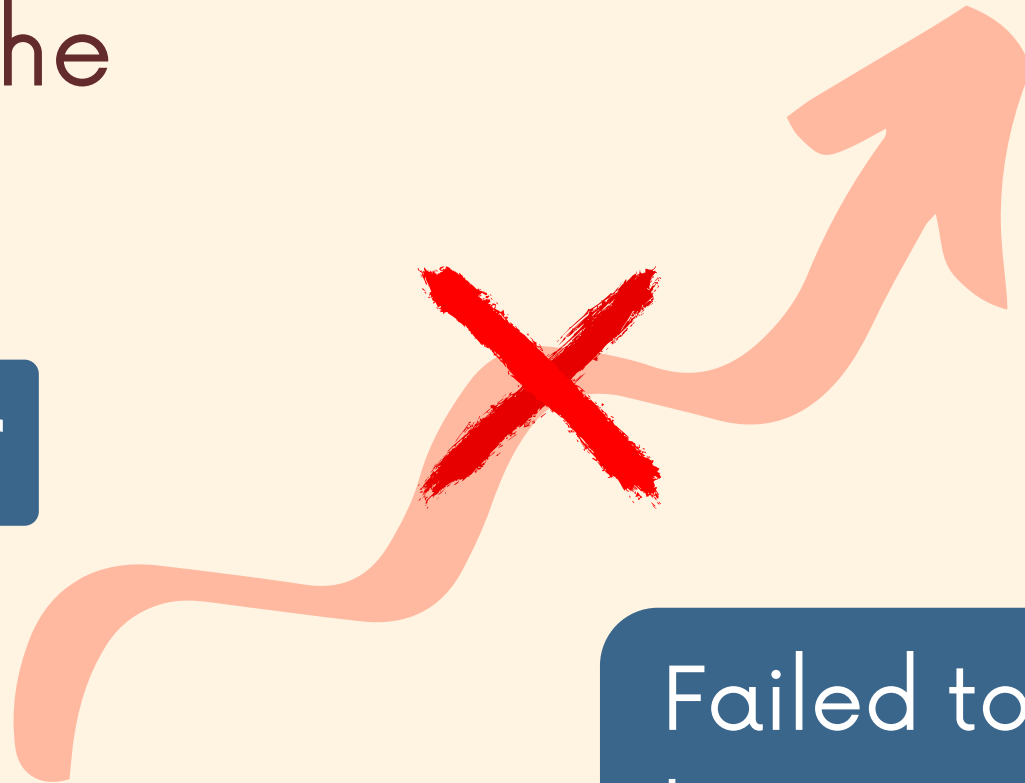
master



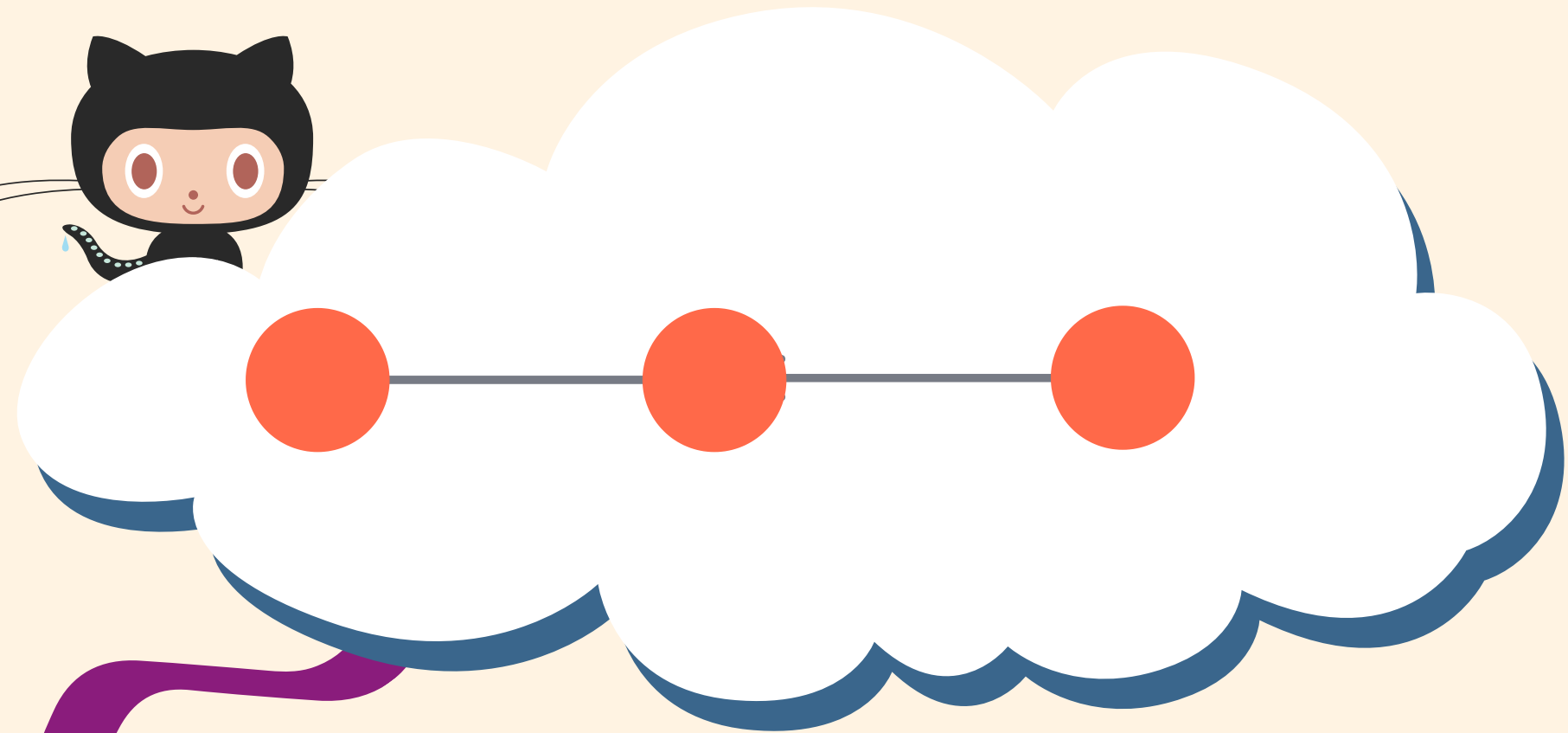
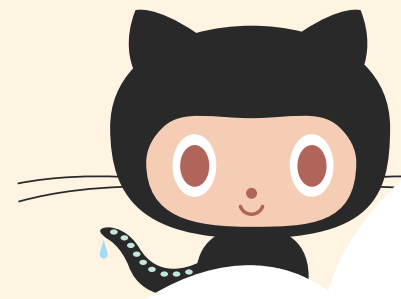
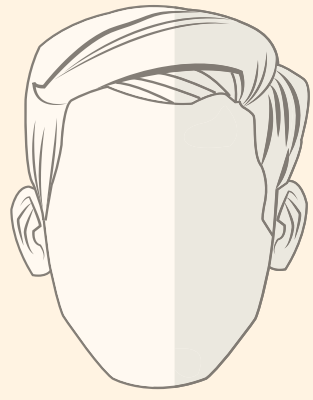
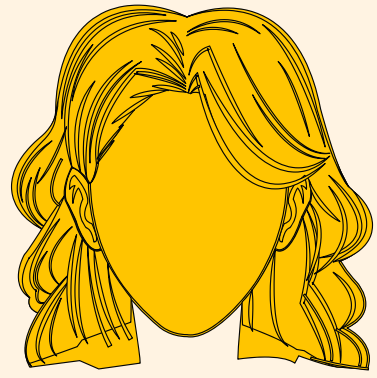


She tries to **push** her new work up to Github, but she runs into trouble!

master

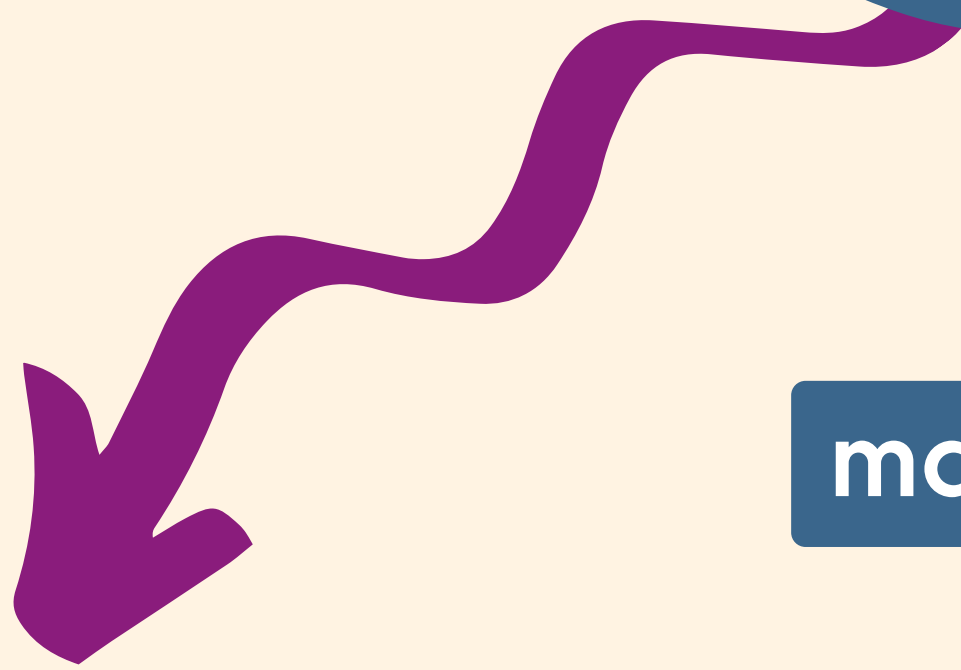


Failed to push. Updates were rejected because the tip of your current branch is behind its remote counterpart. Merge the remote changes (e.g. 'git pull') before pushing again.



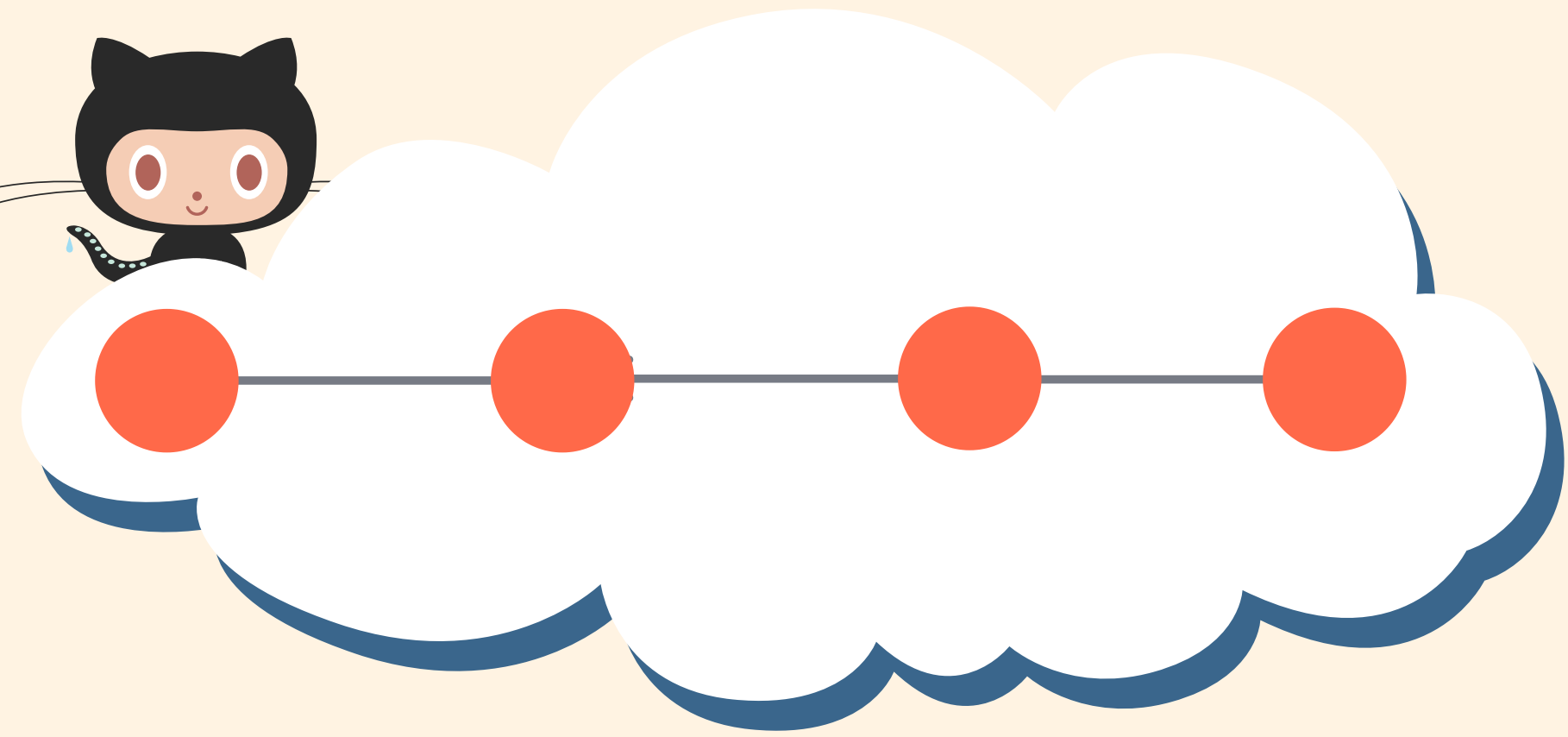
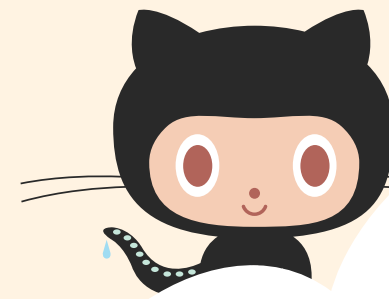
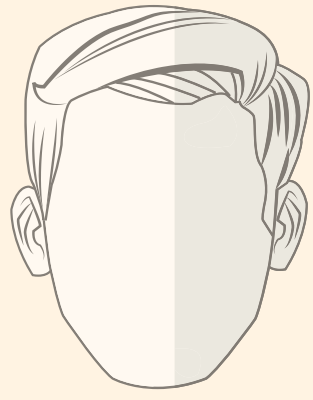
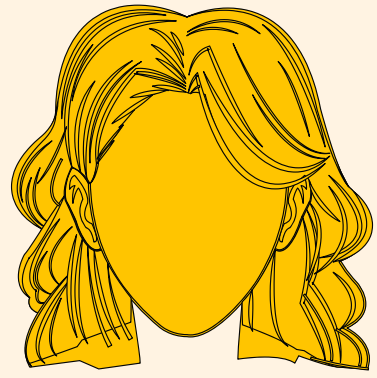
So she pulls to get the changes from origin master

Forrest's work must be merged in. Hopefully this goes relatively smoothly!



master

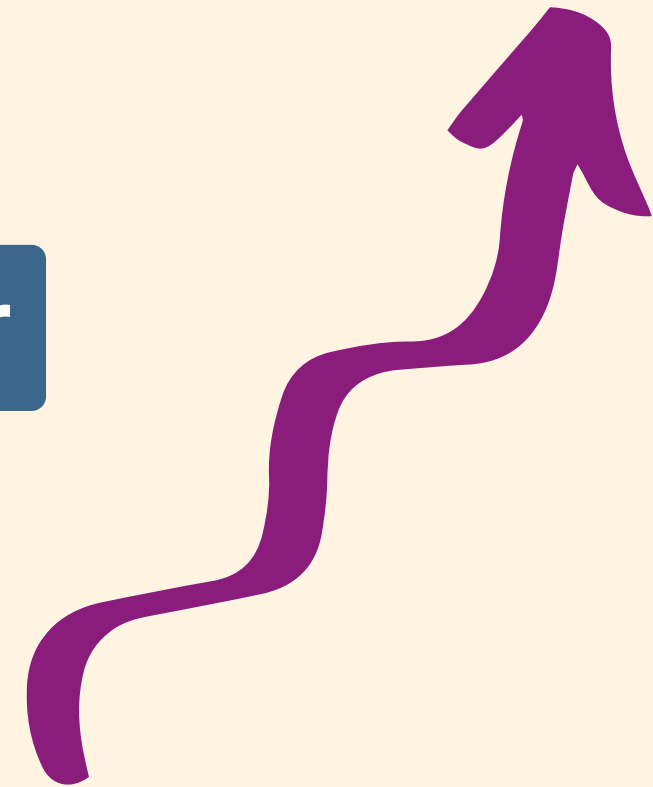


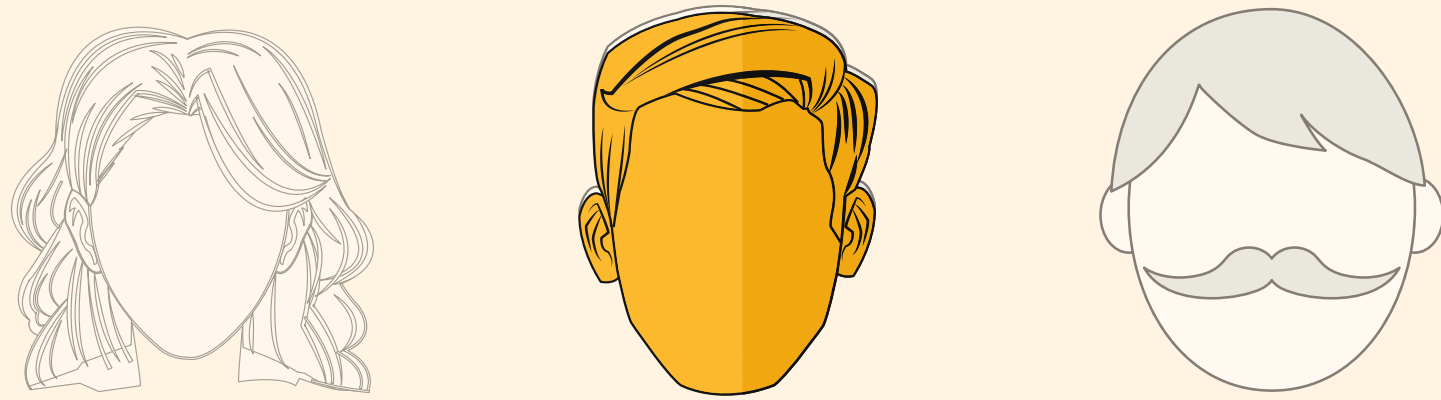


Now that she has merged the latest work from Github, she can push her master branch up!



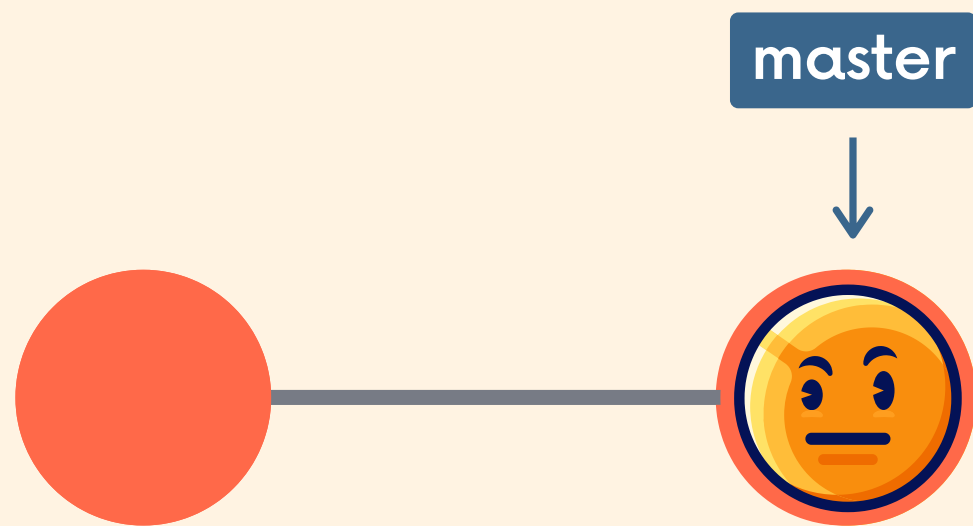
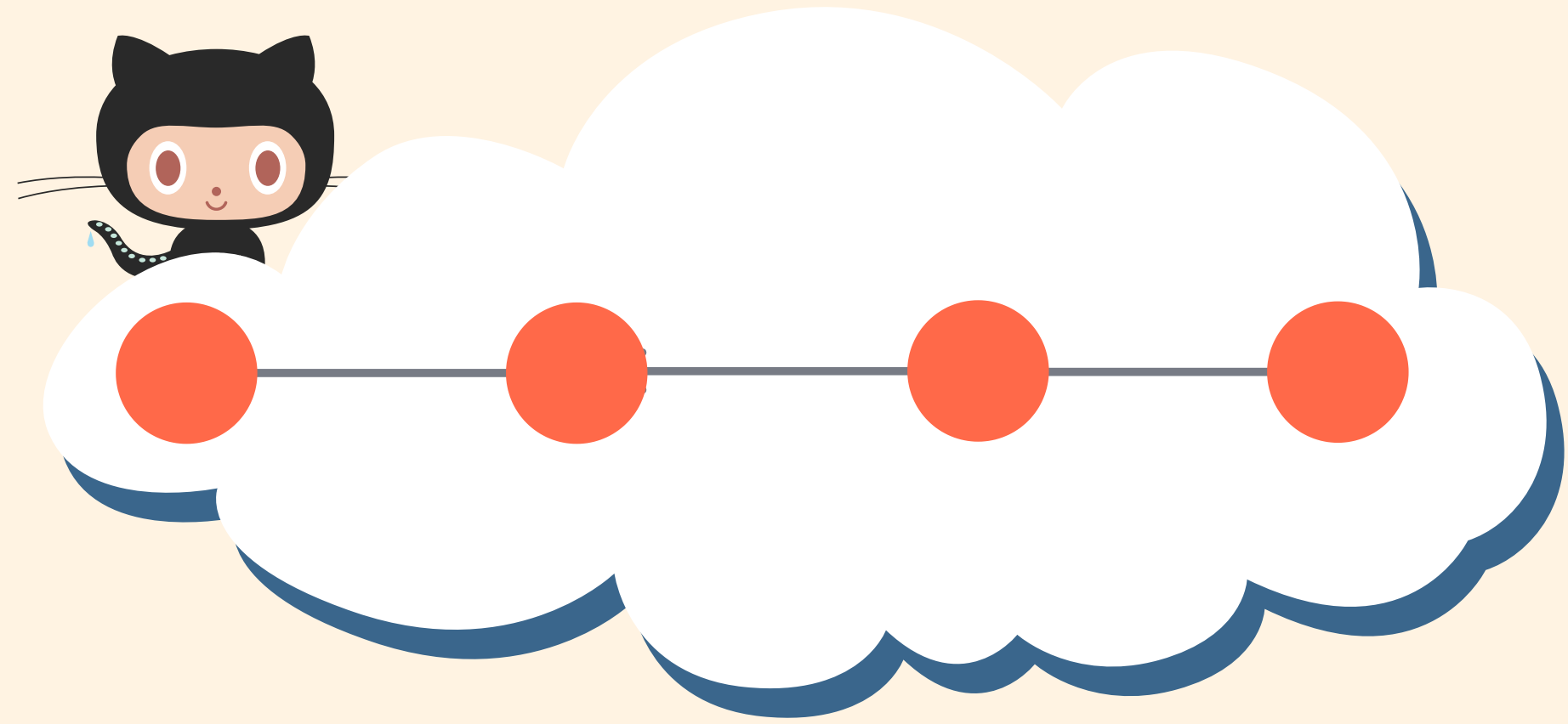
master



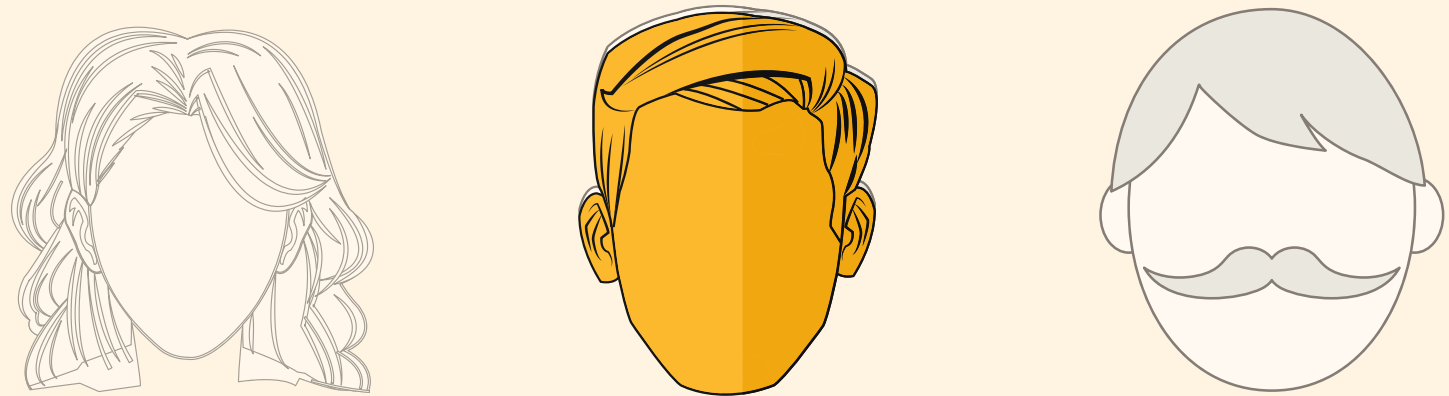


David working on a new feature, but is having some doubts.

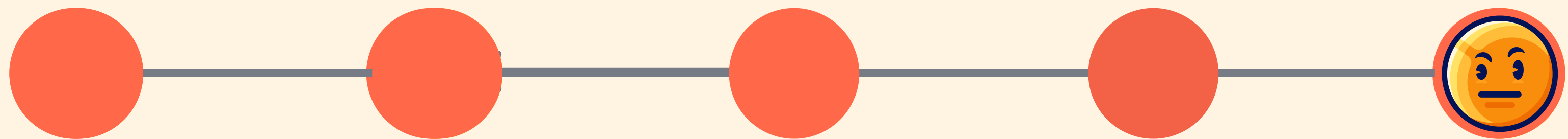
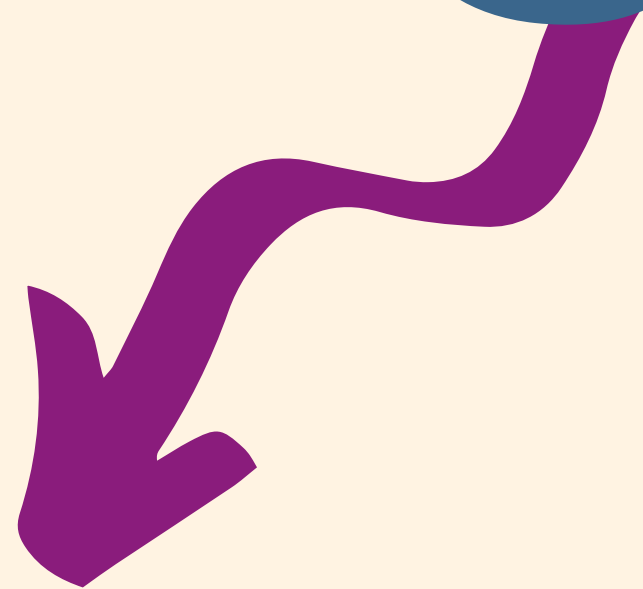
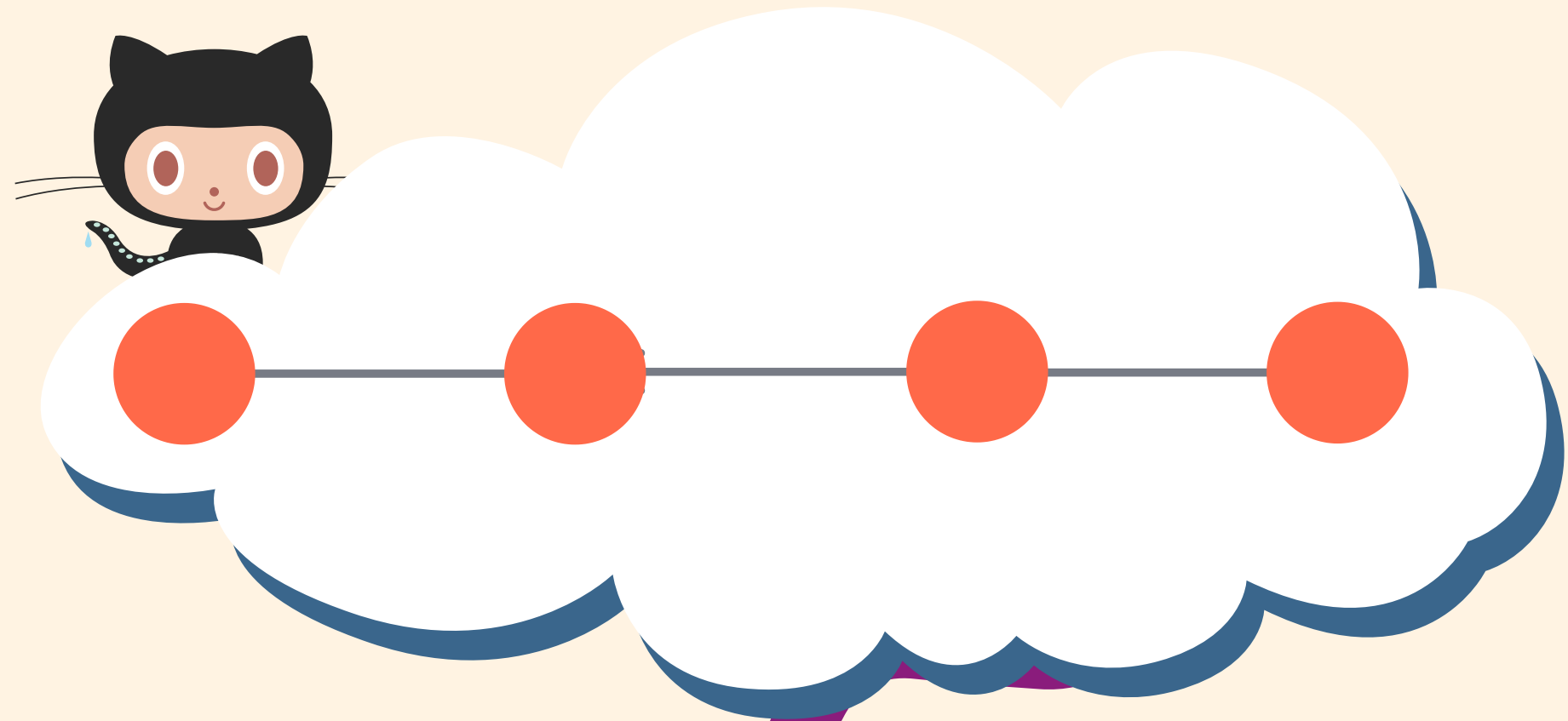
He'd like to share his commits with the rest of the team to start a discussion.

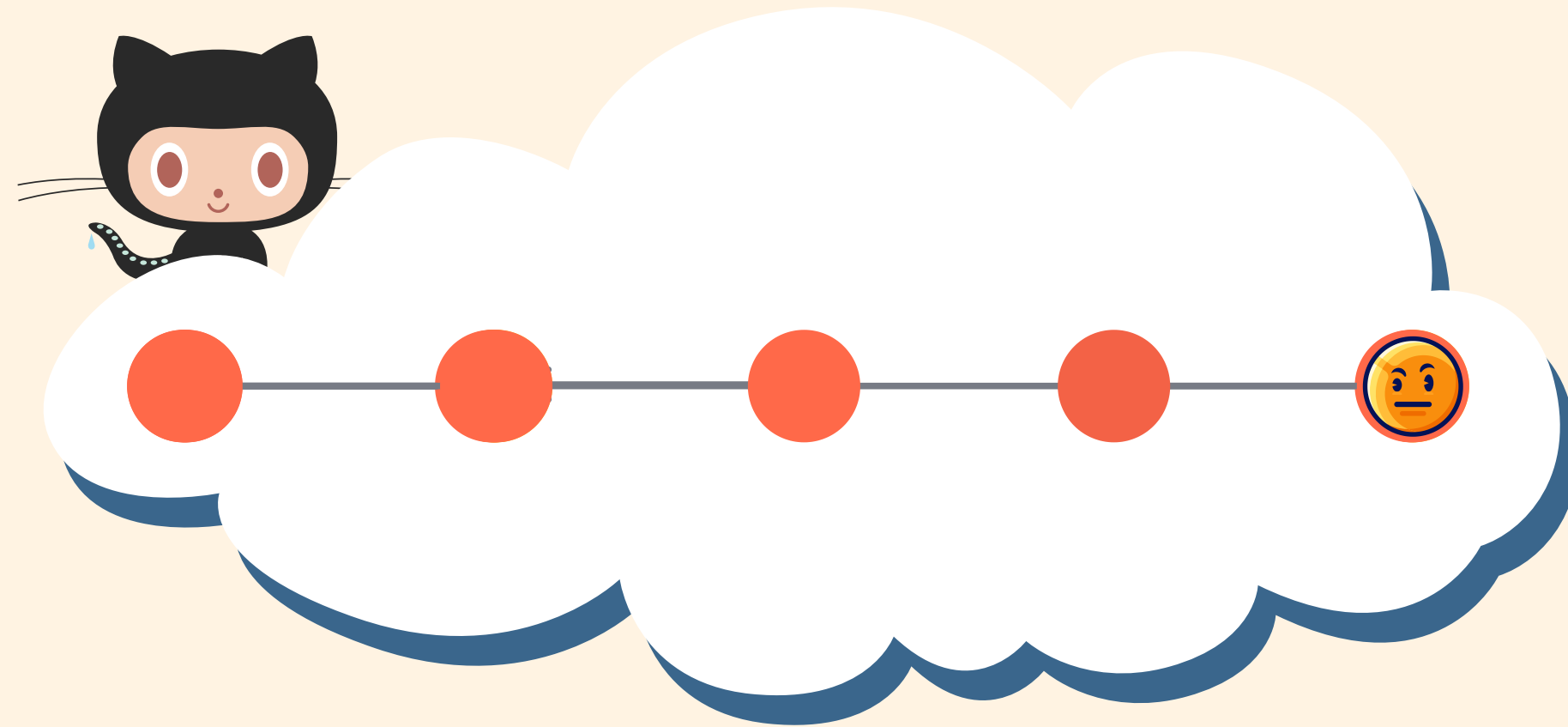
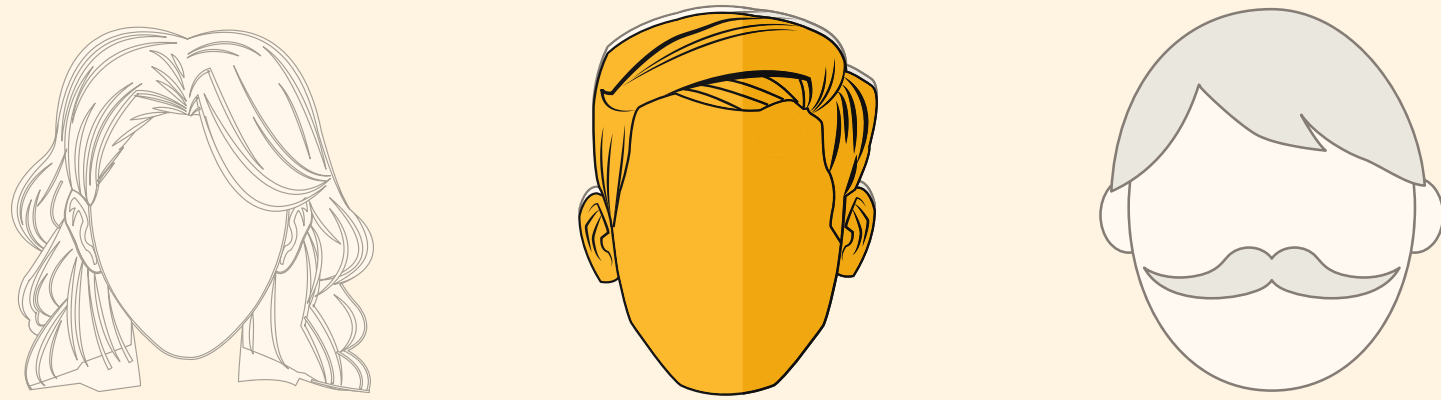




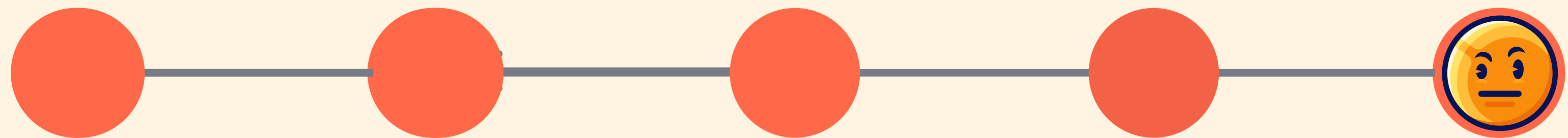


Before he can even share these iffy commits, he has to **pull** from Github and **merge** them in to master





Now he can finally push his work up to Github. His teammates can pull to get his new commits.





# The Problem

While it's nice and easy to only work on the master branch, this leads to some serious issues on teams!

- Lots of time spent resolving conflicts and merging code, especially as team size scales up.
- No one can work on anything without disturbing the main codebase. How do you try adding something radically different in? How do you experiment?
- The only way to collaborate on a feature together with another teammate is to push incomplete code to master. Other teammates now have broken code...





# Enter

# Feature

# Branches

DON'T WORK ON MASTER SILLY GOOSE!





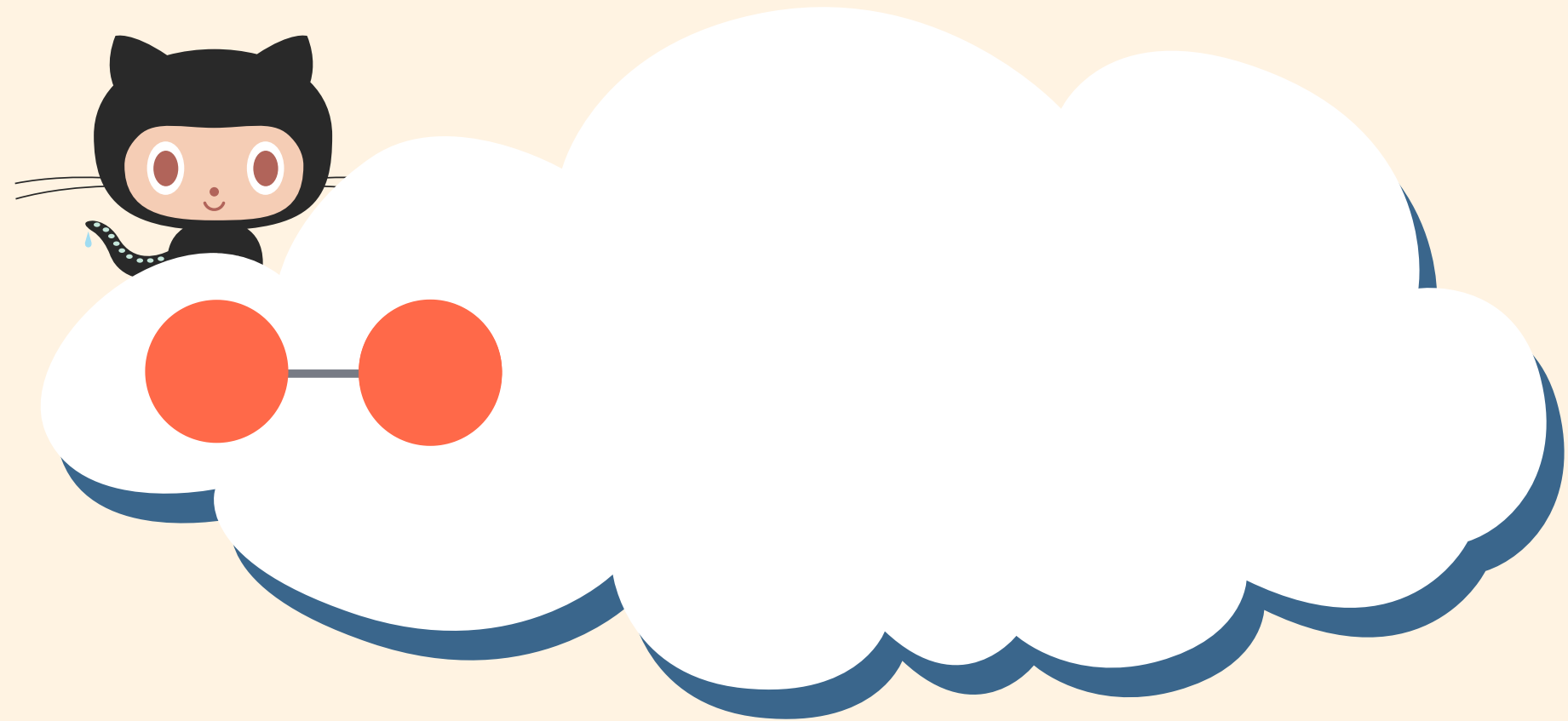
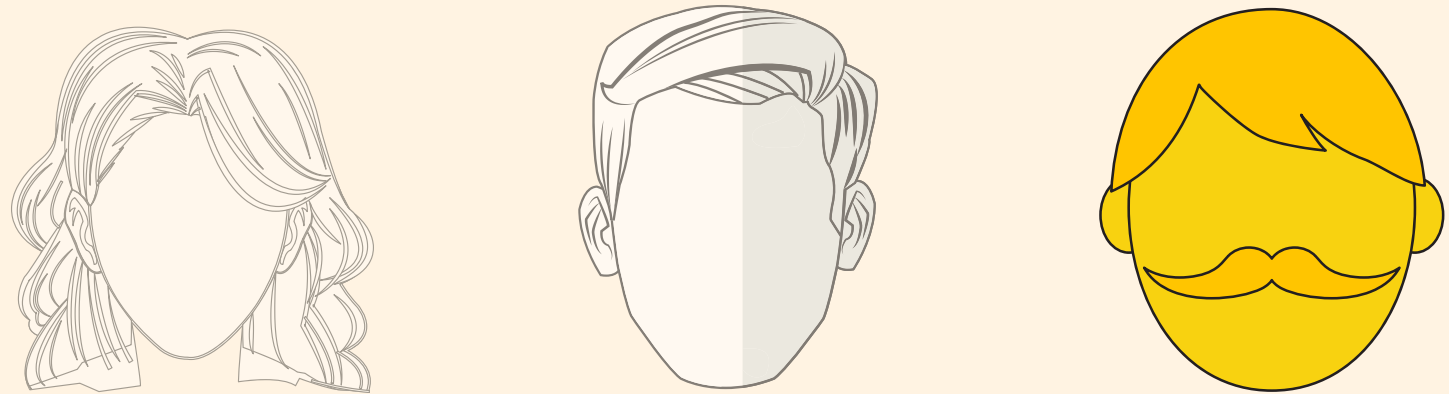


# Feature Branches

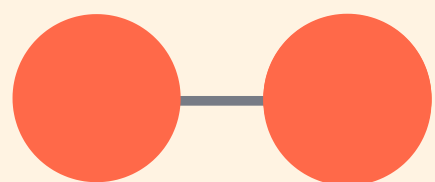
Rather than working directly on master/main, all new development should be done on separate branches!

- Treat master/main branch as the official project history
- Multiple teammates can collaborate on a single feature and share code back and forth without polluting the master/main branch
- Master/main branch won't contain broken code (or at least, it won't unless someone messes up)

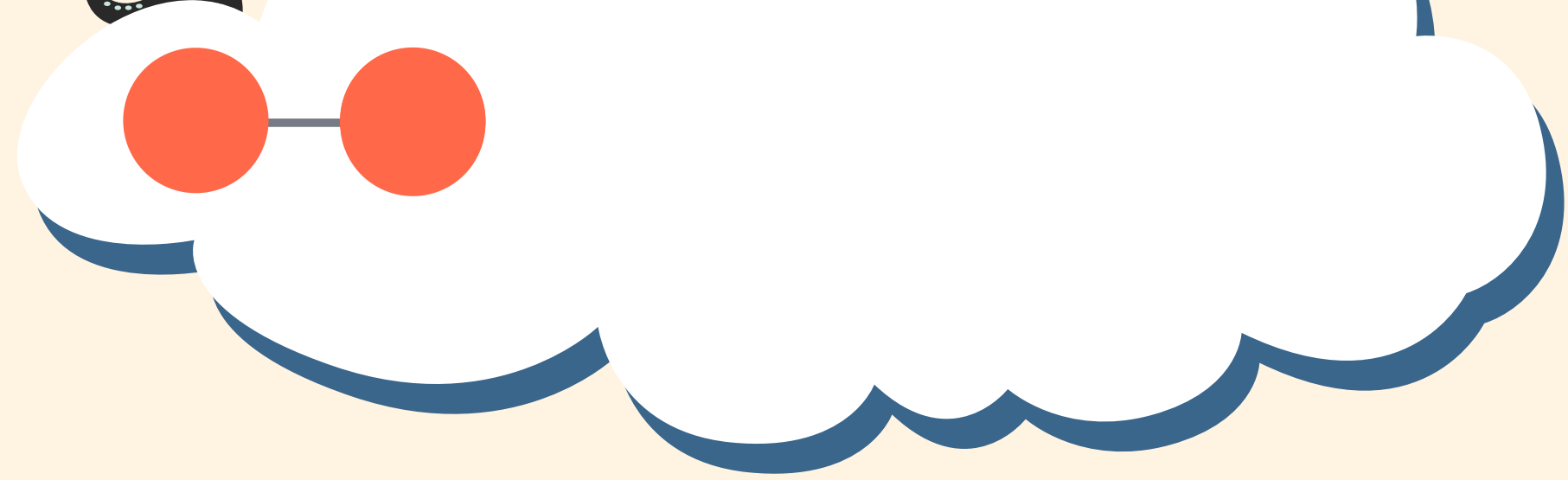
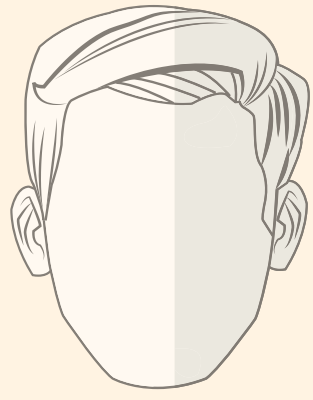
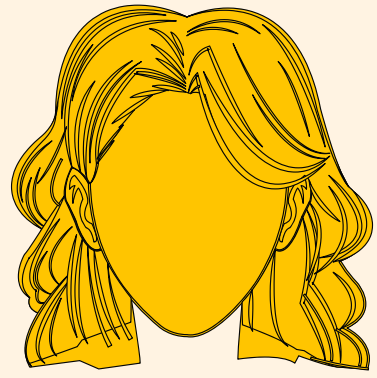




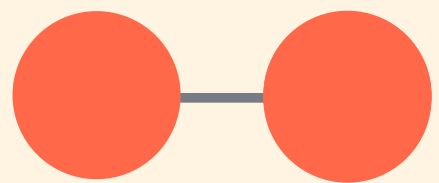
Forrest clones the repo



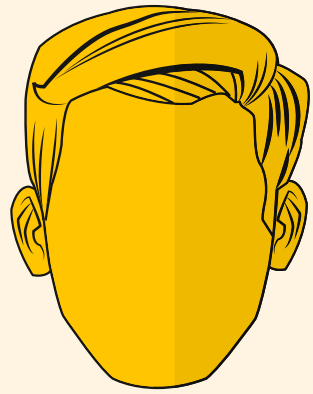
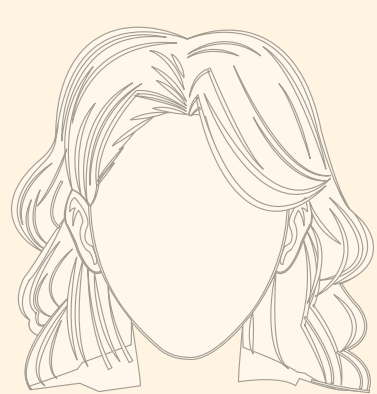
master



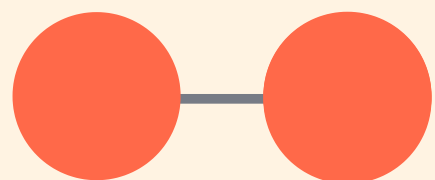
Pamela clones the repo



master

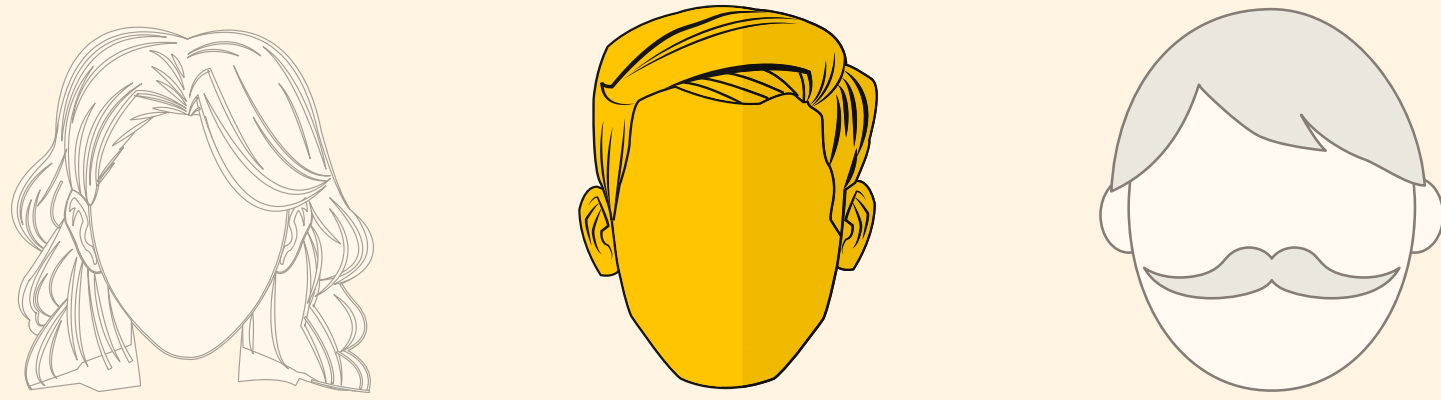


David clones the repo

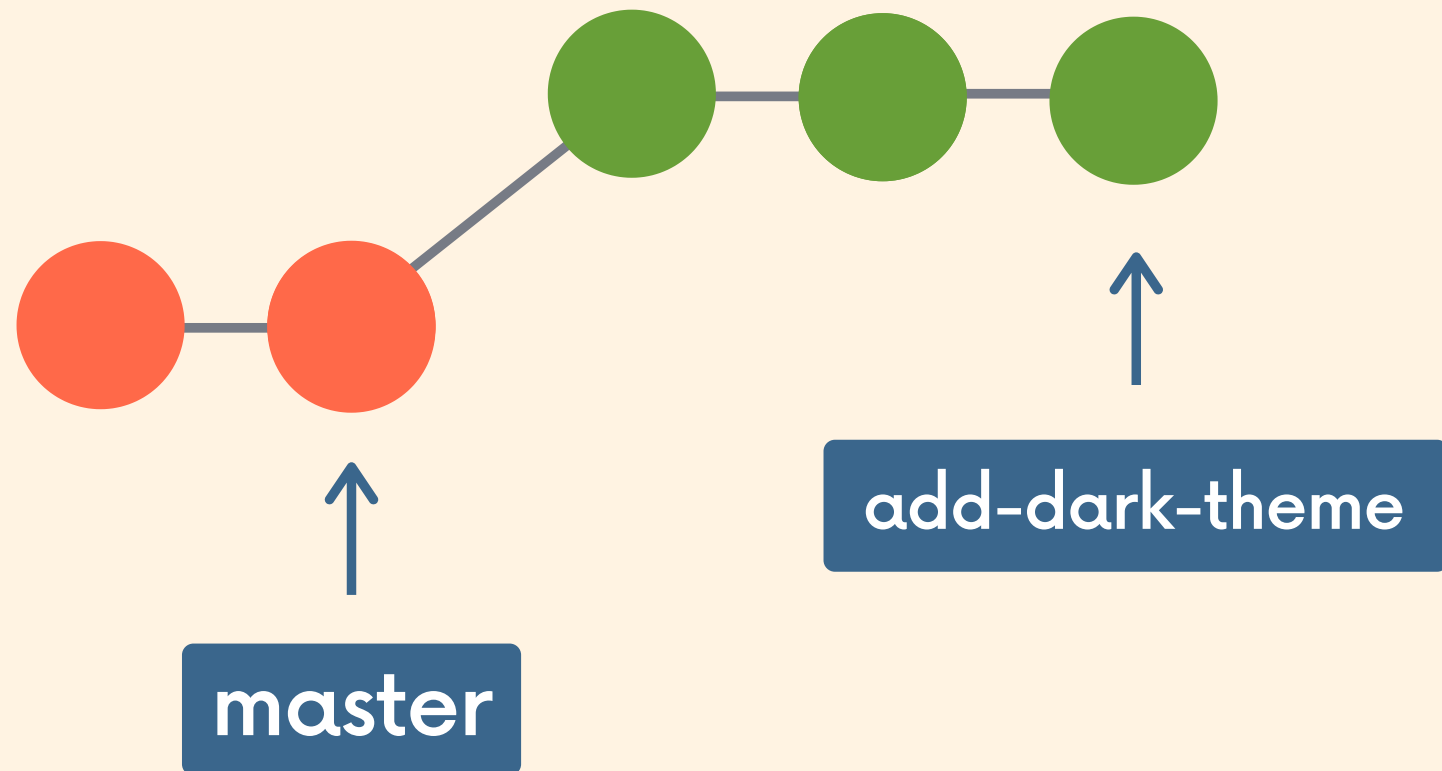
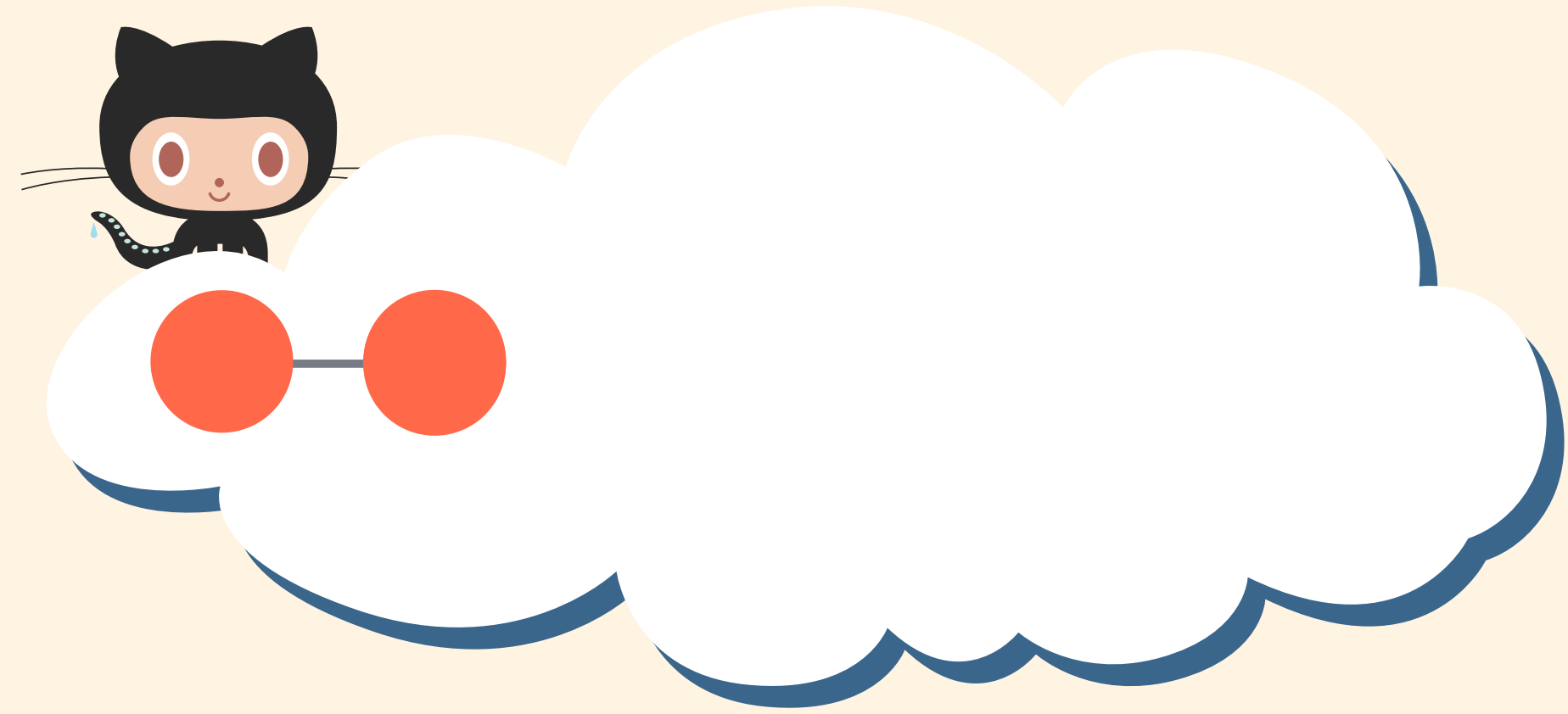


master

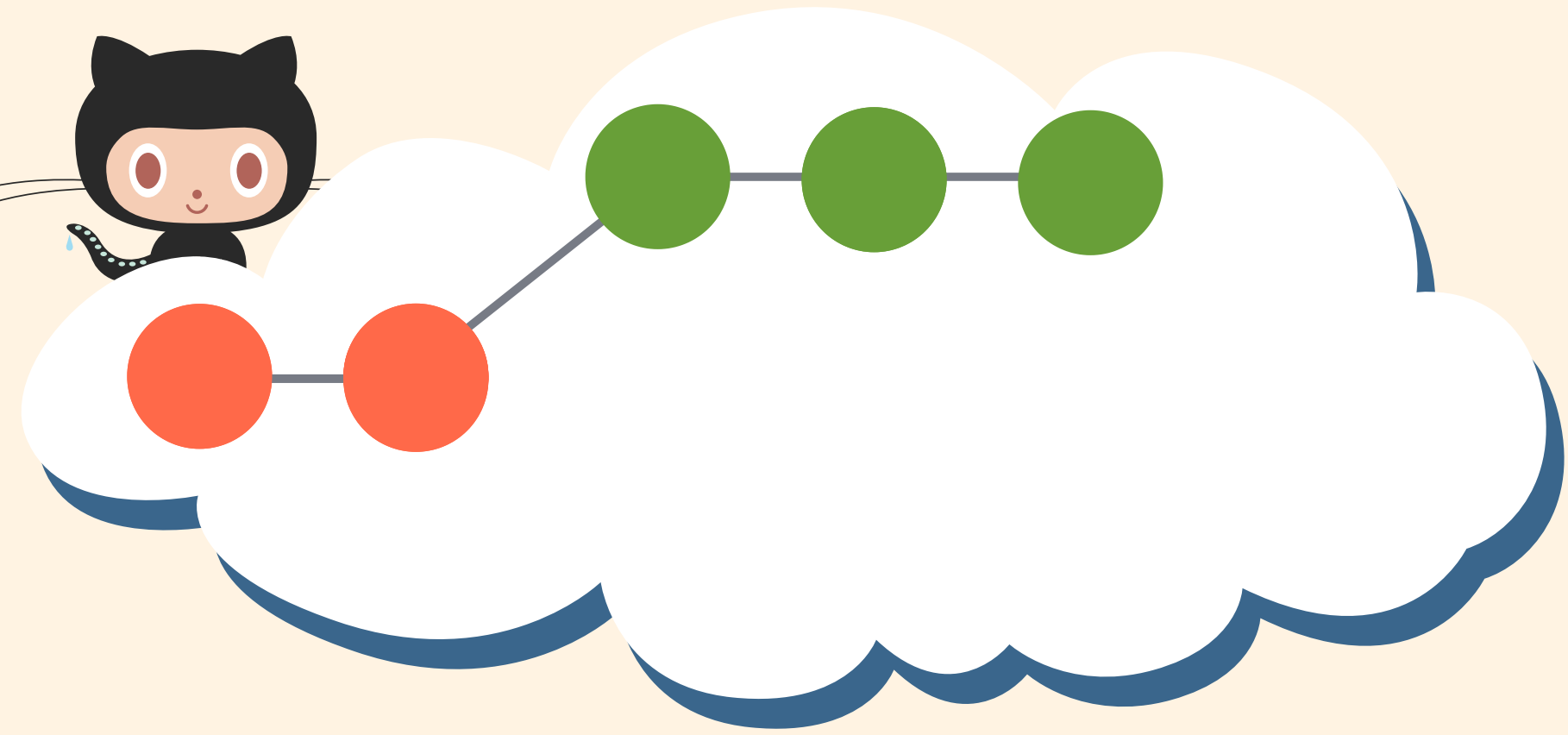
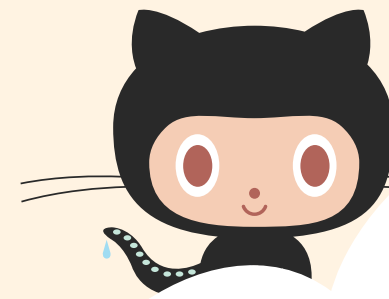
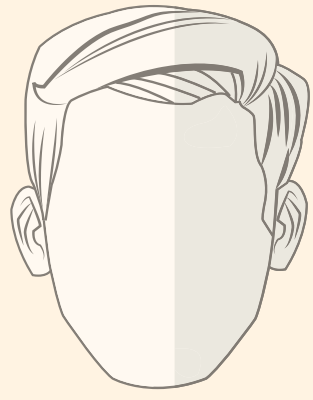
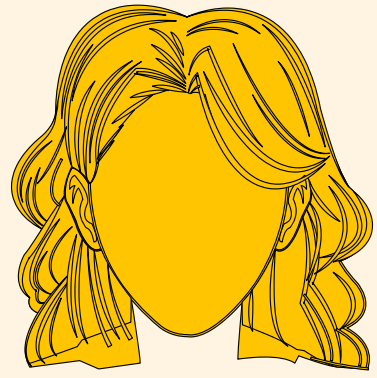




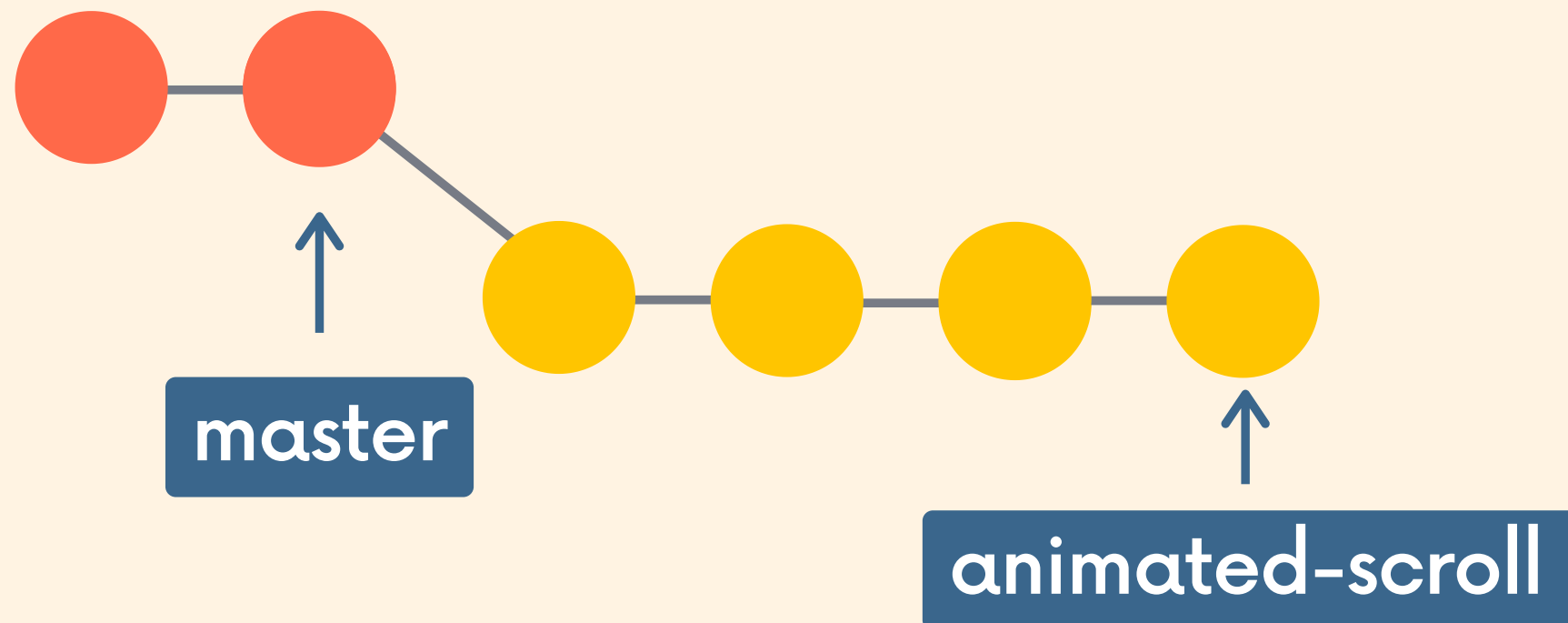
David starts work on a new feature. He does all this work on a separate branch!







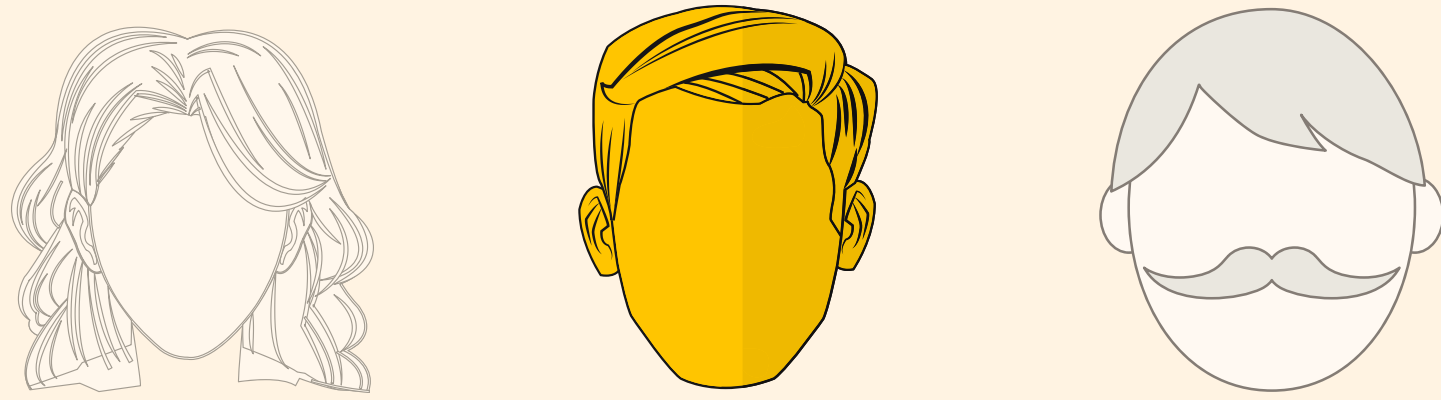
**Pamela** is hard at work on her own new feature. Just like everyone else, she's working on a separate feature branch rather than master.



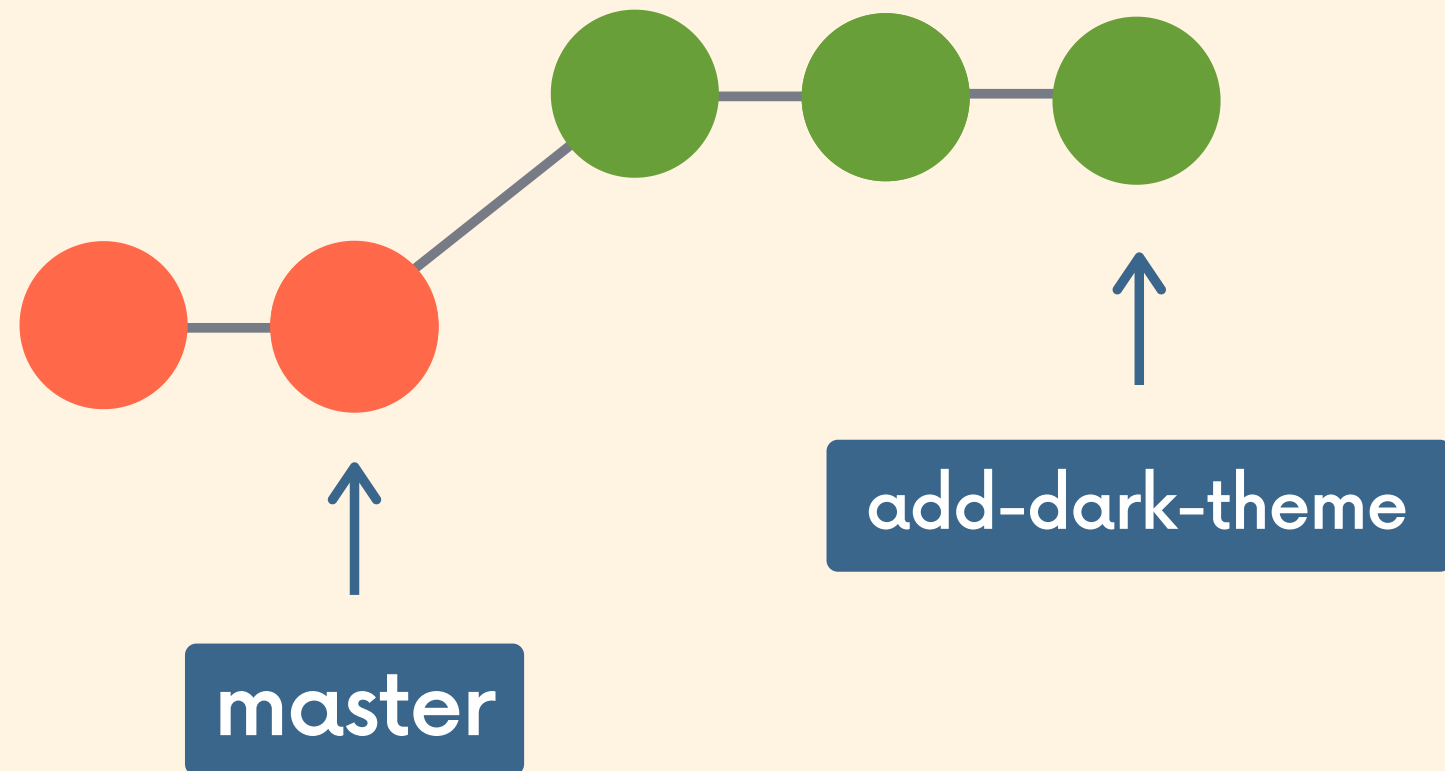
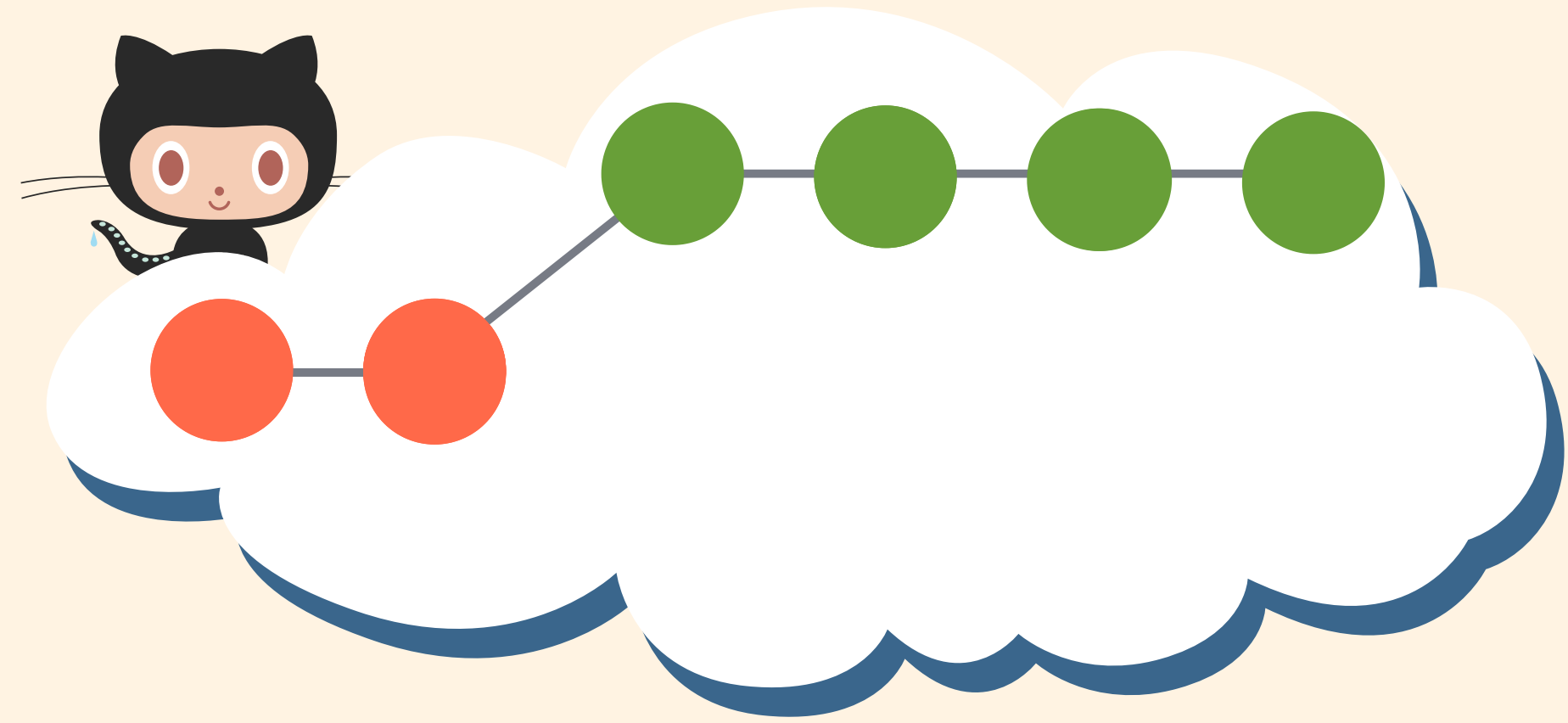


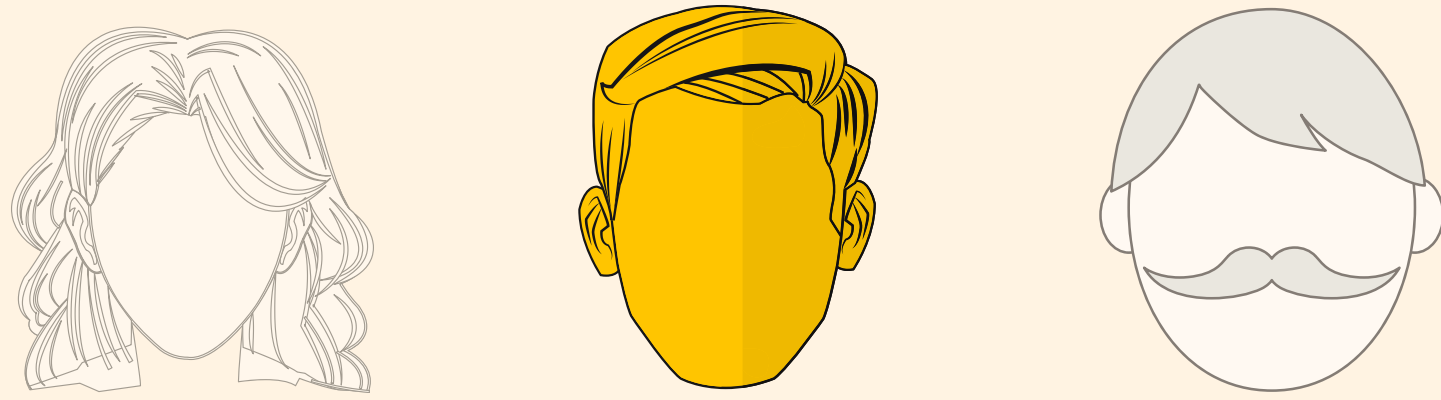




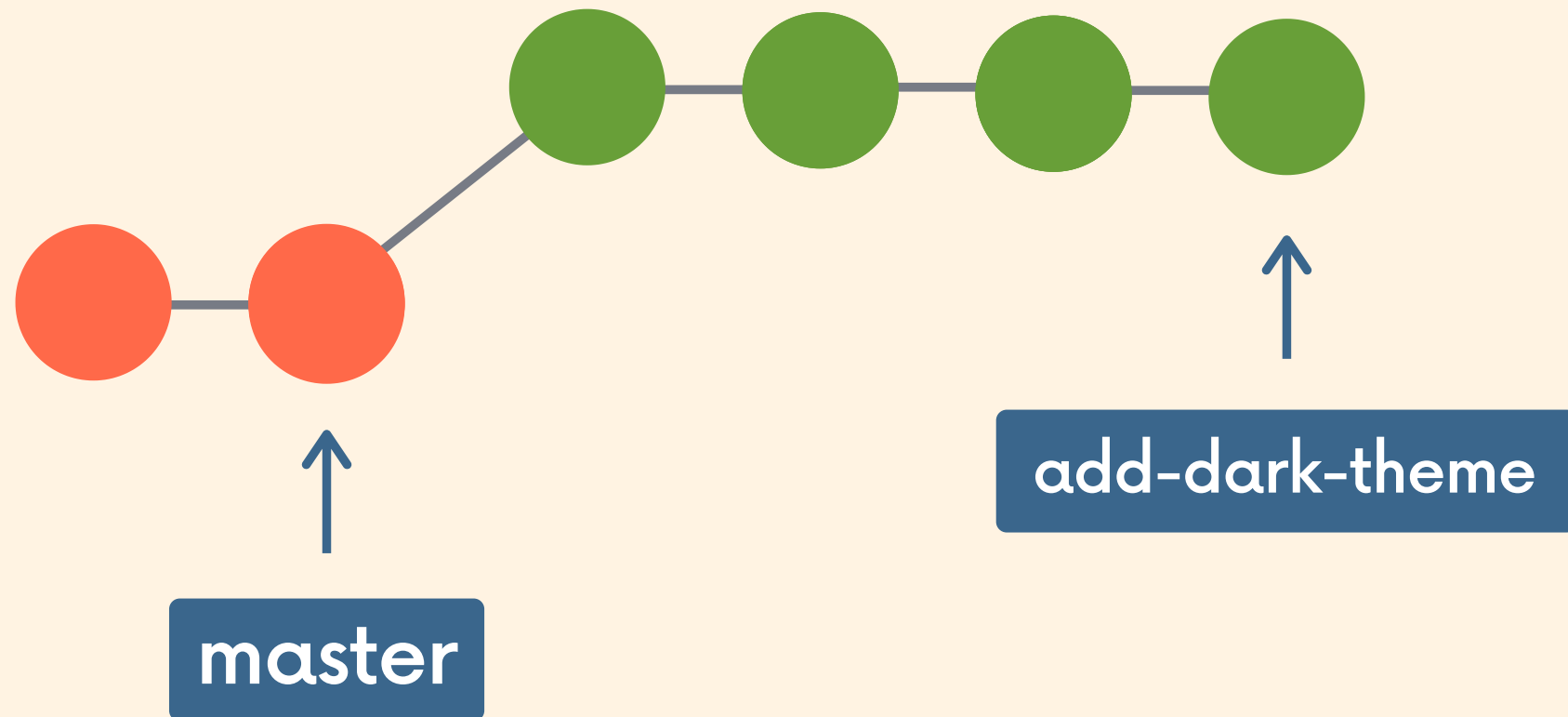
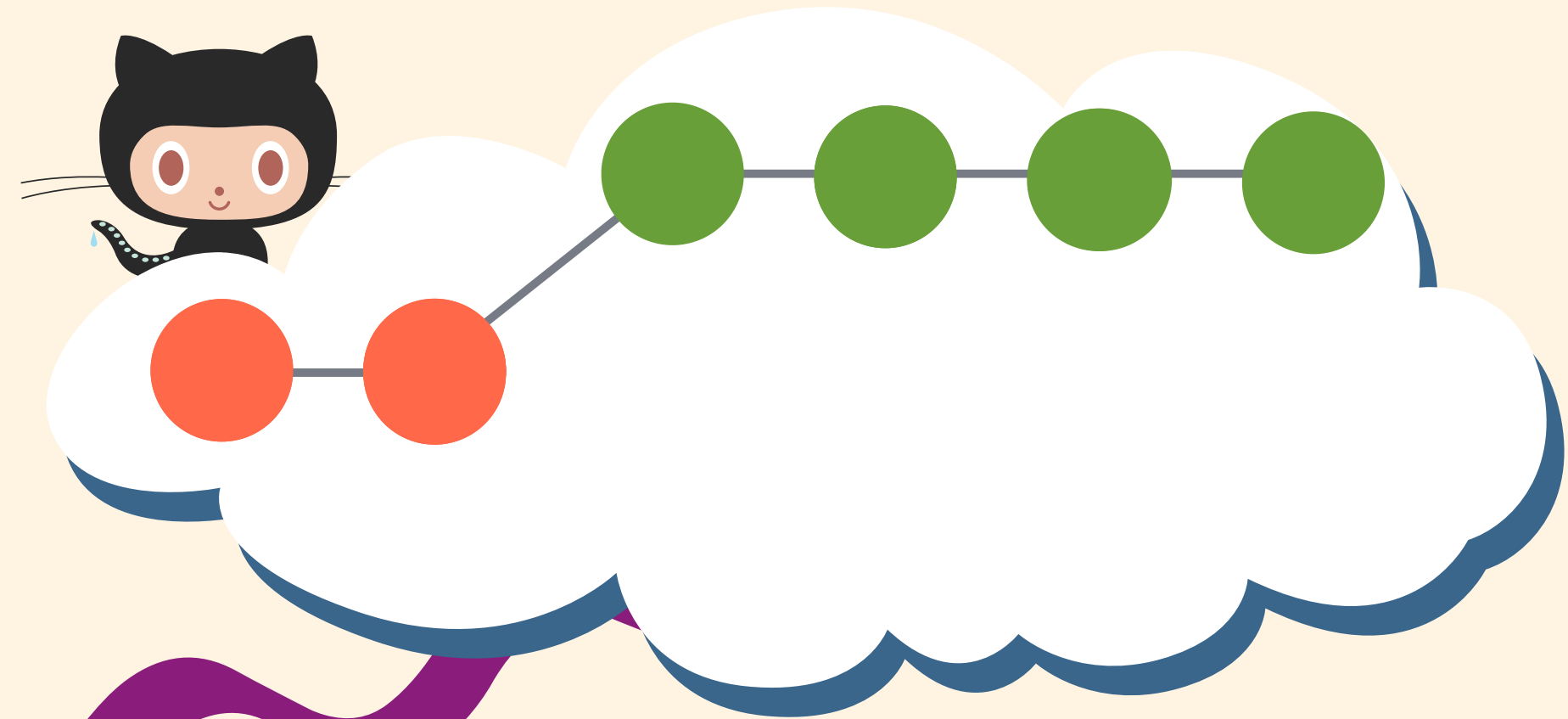


**David** returns to work the next morning. After an hour on reddit he decides he should actually do some work.

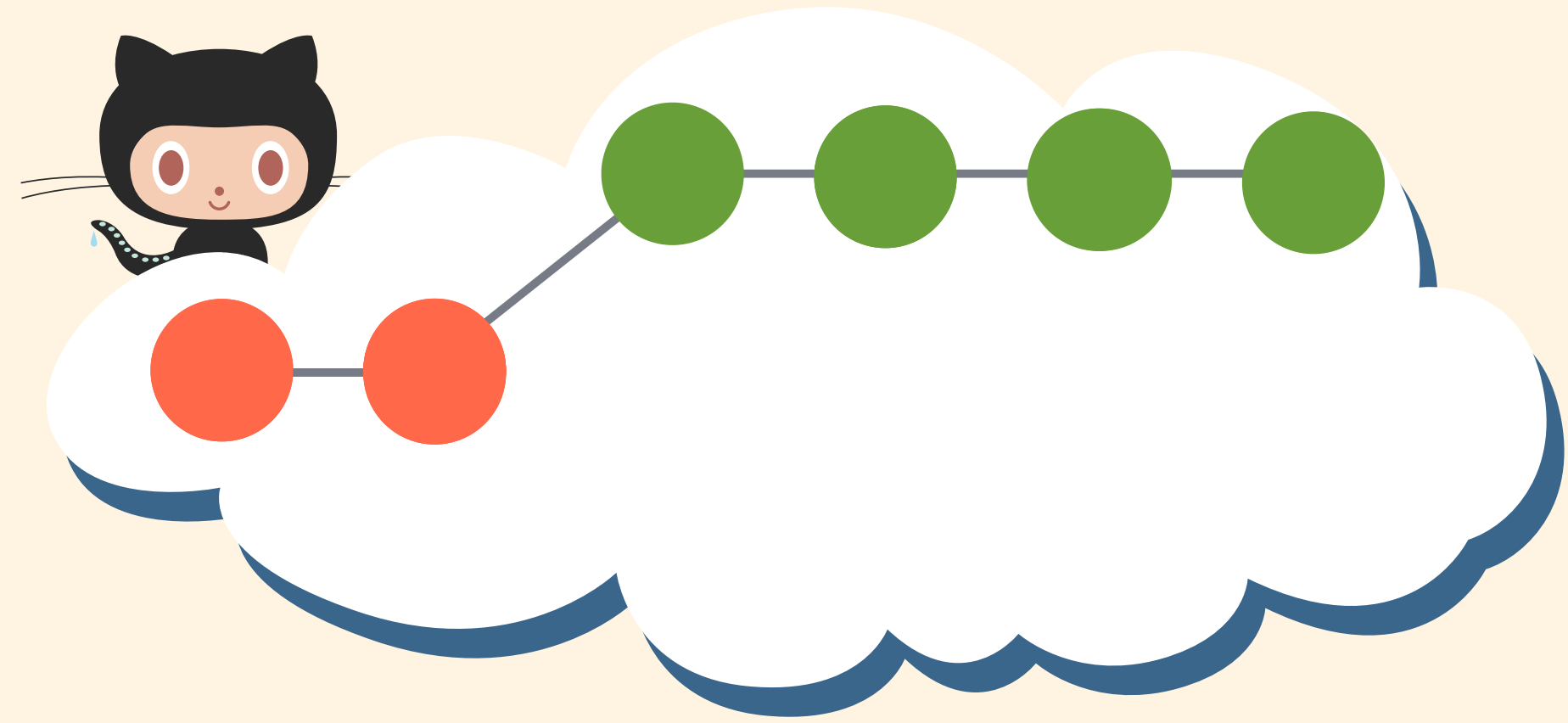
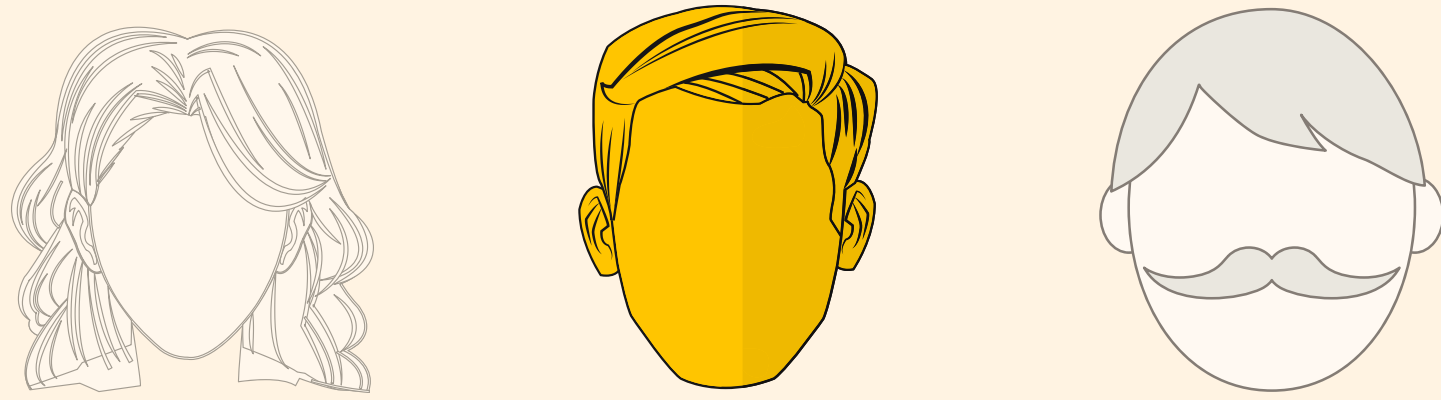




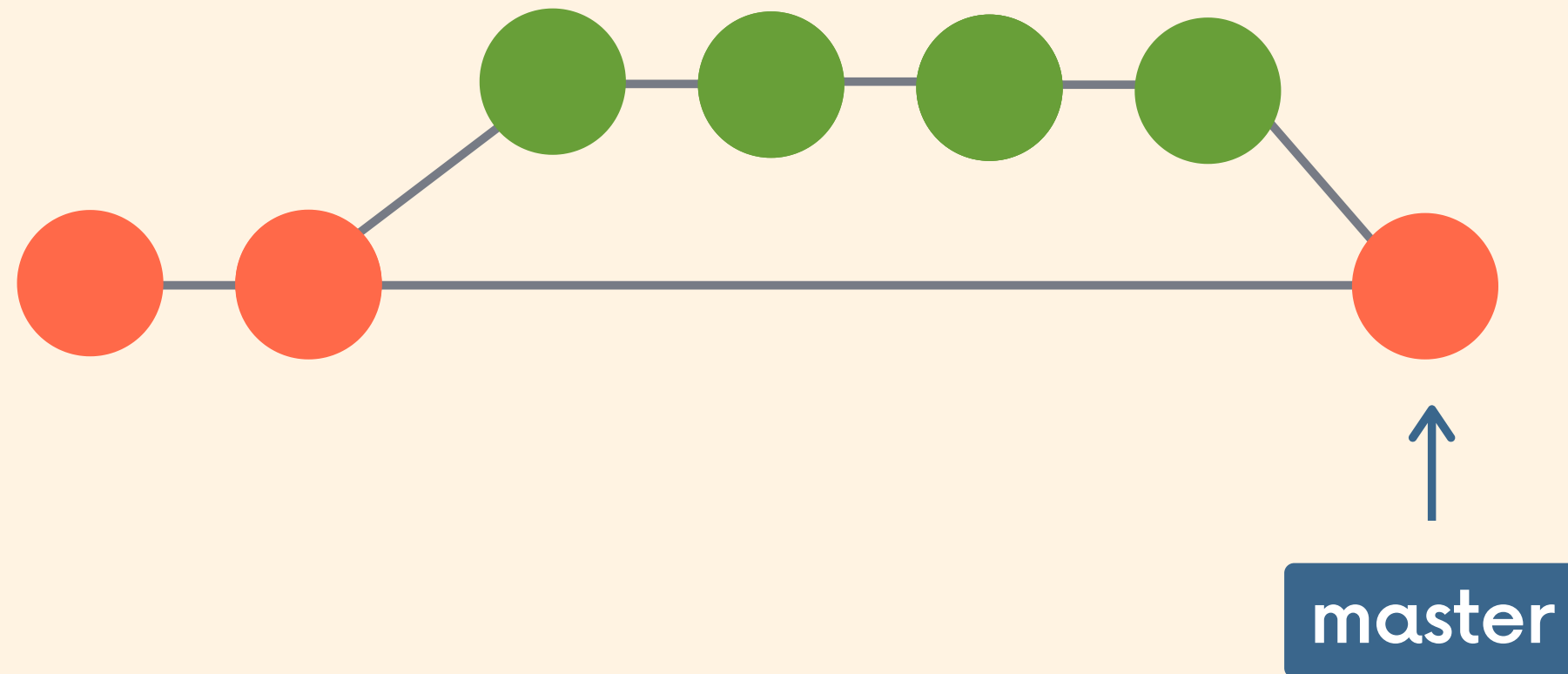
**David** fetches from Github and sees that there is new work on the add-dark-theme branch. He pulls the changes down and continues work.



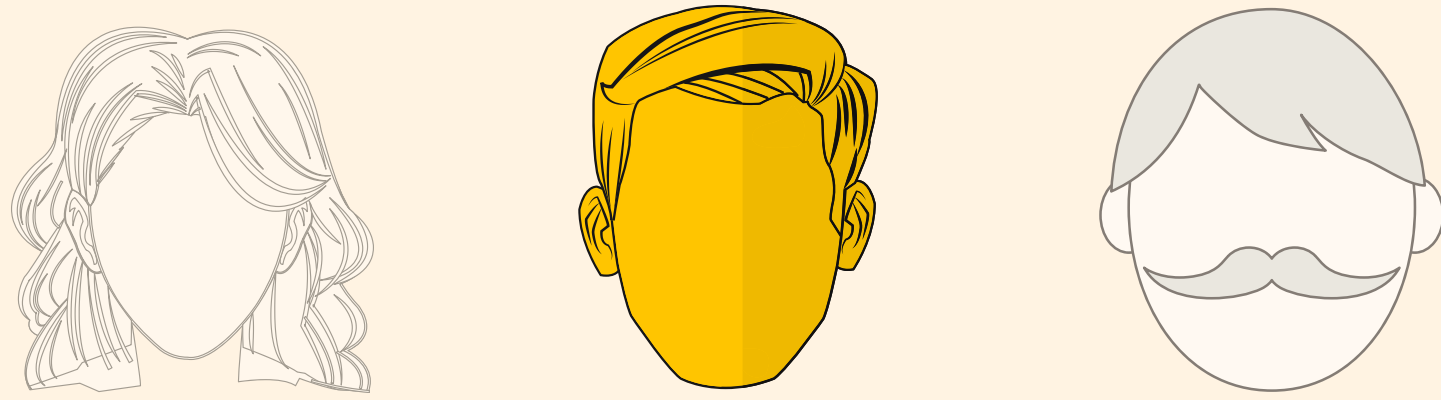




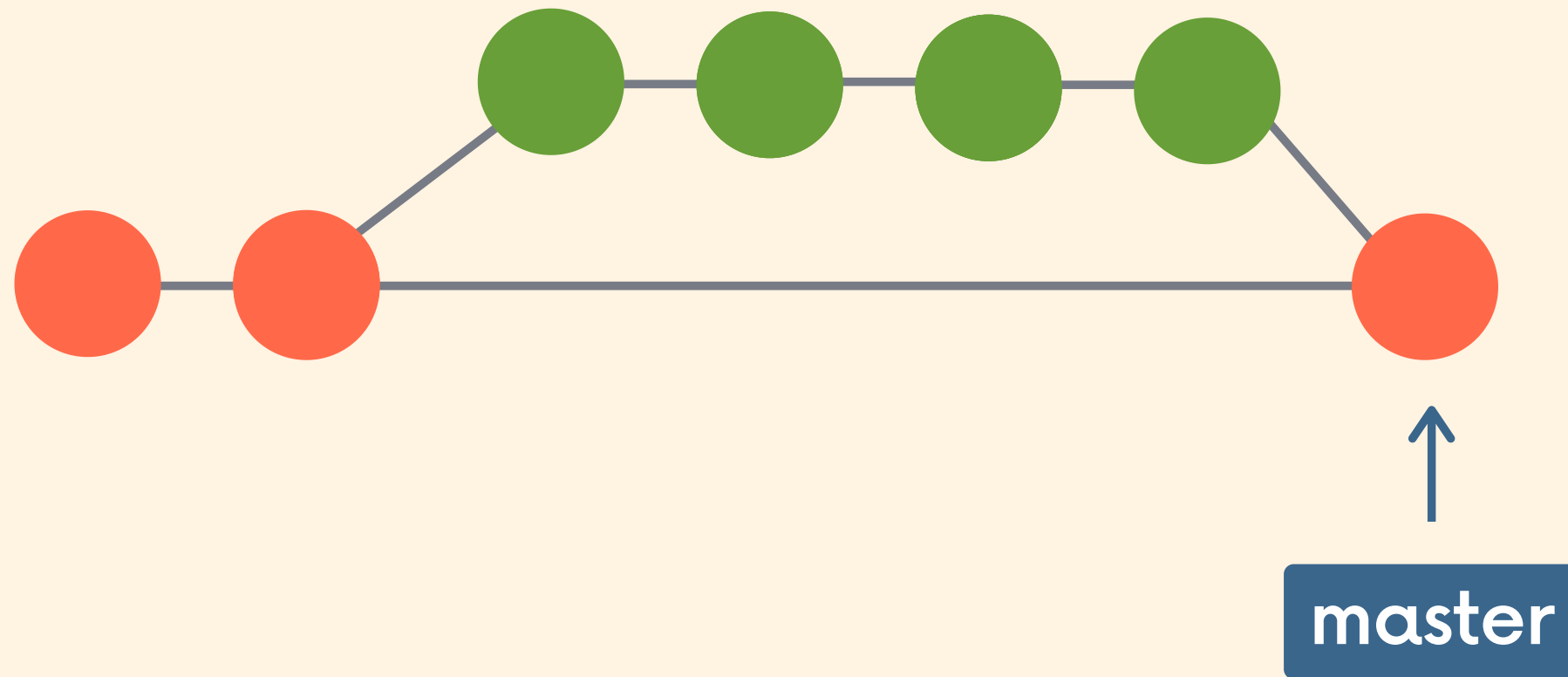
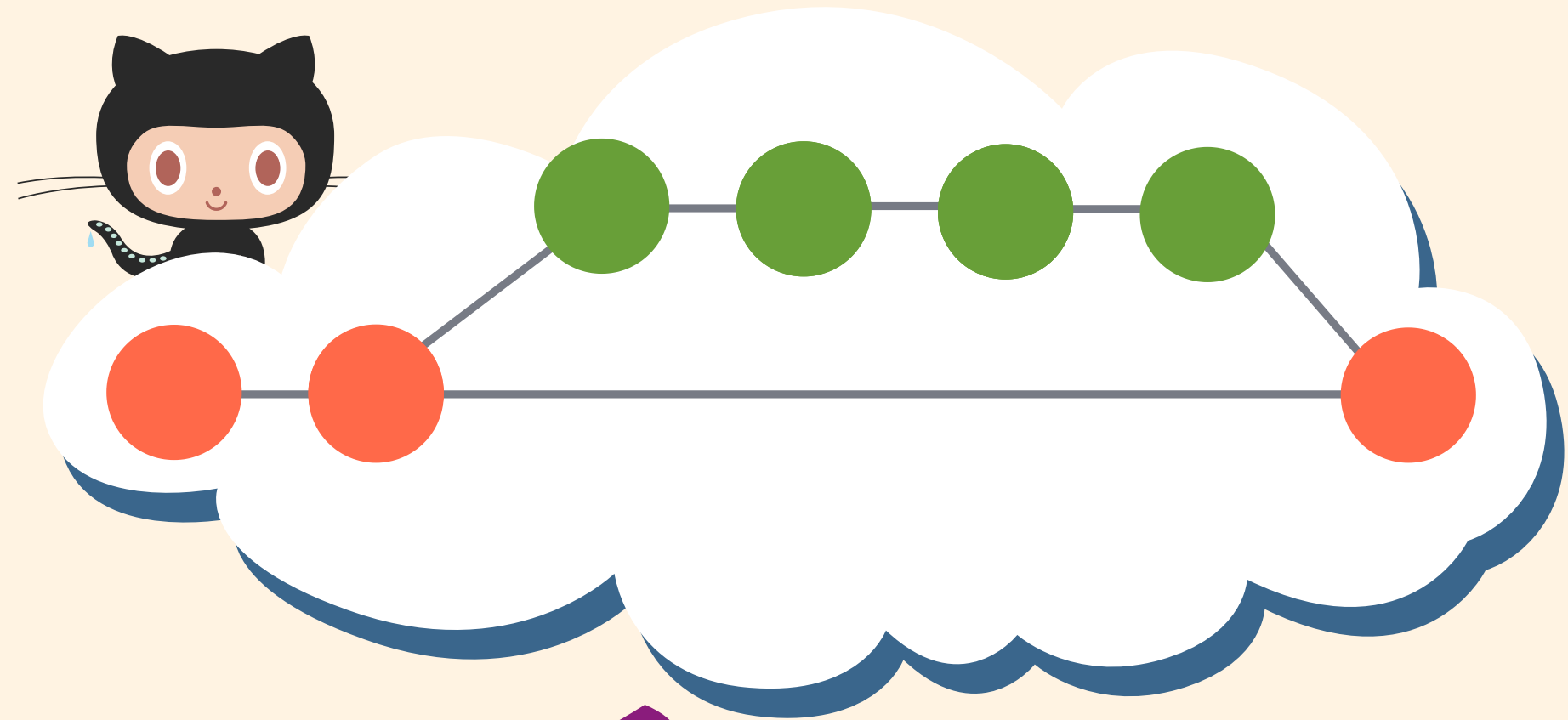
David decides he is happy with the new feature, so he merges it into master!



\*At a real company, you don't just decide you are "happy with" a feature. There are mechanisms for code approval and merging we'll discuss shortly!



David pushes up the updated master branch to Github. The others can now pull down the changes.





# Feature Branch Naming

There are many different approaches for naming feature branches. Often you'll see branch names that include slashes like **bug/fix-scroll** or **feature/login-form** or **feat/button/enable-pointer-events**

Specific teams and projects usually have their own branch naming conventions. To keep these slides simple and concise, I'm just going to ignore those best-practices for now.

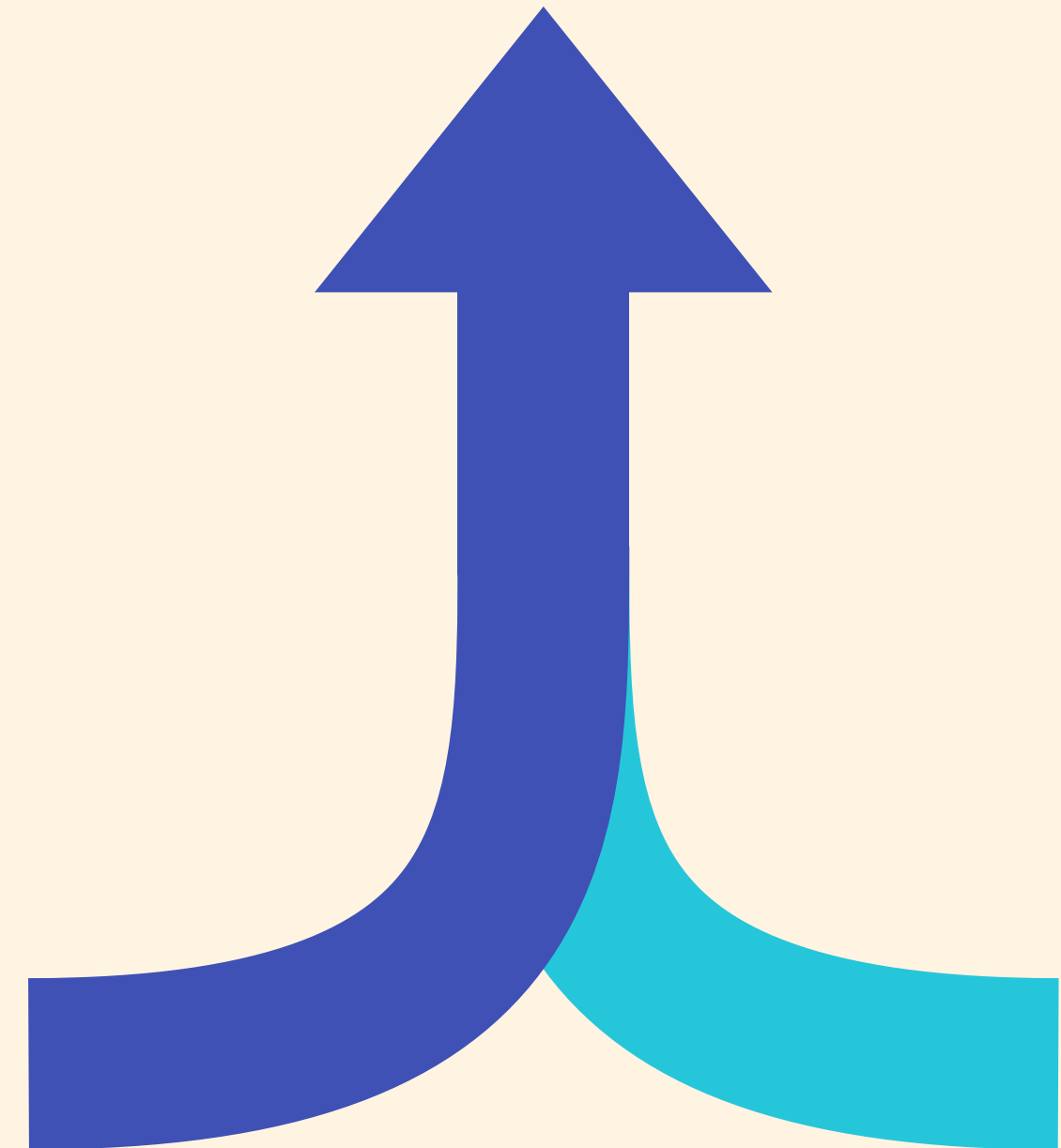




# Merging In Feature Branches

At some point new the work on feature branches will need to be merged in to the master branch!  
There are a couple of options for how to do this...

1. Merge at will, without any sort of discussion with teammates. **JUST DO IT WHENEVER YOU WANT.**
2. Send an email or chat message or something to your team to discuss if the changes should be merged in.
3. **Pull Requests!**







# Pull Requests

Pull Requests are a feature built in to products like Github & Bitbucket. **They are not native to Git itself.**

They allow developers to alert team-members to new work that needs to be reviewed. They provide a mechanism to approve or reject the work on a given branch. They also help facilitate discussion and feedback on the specified commits.

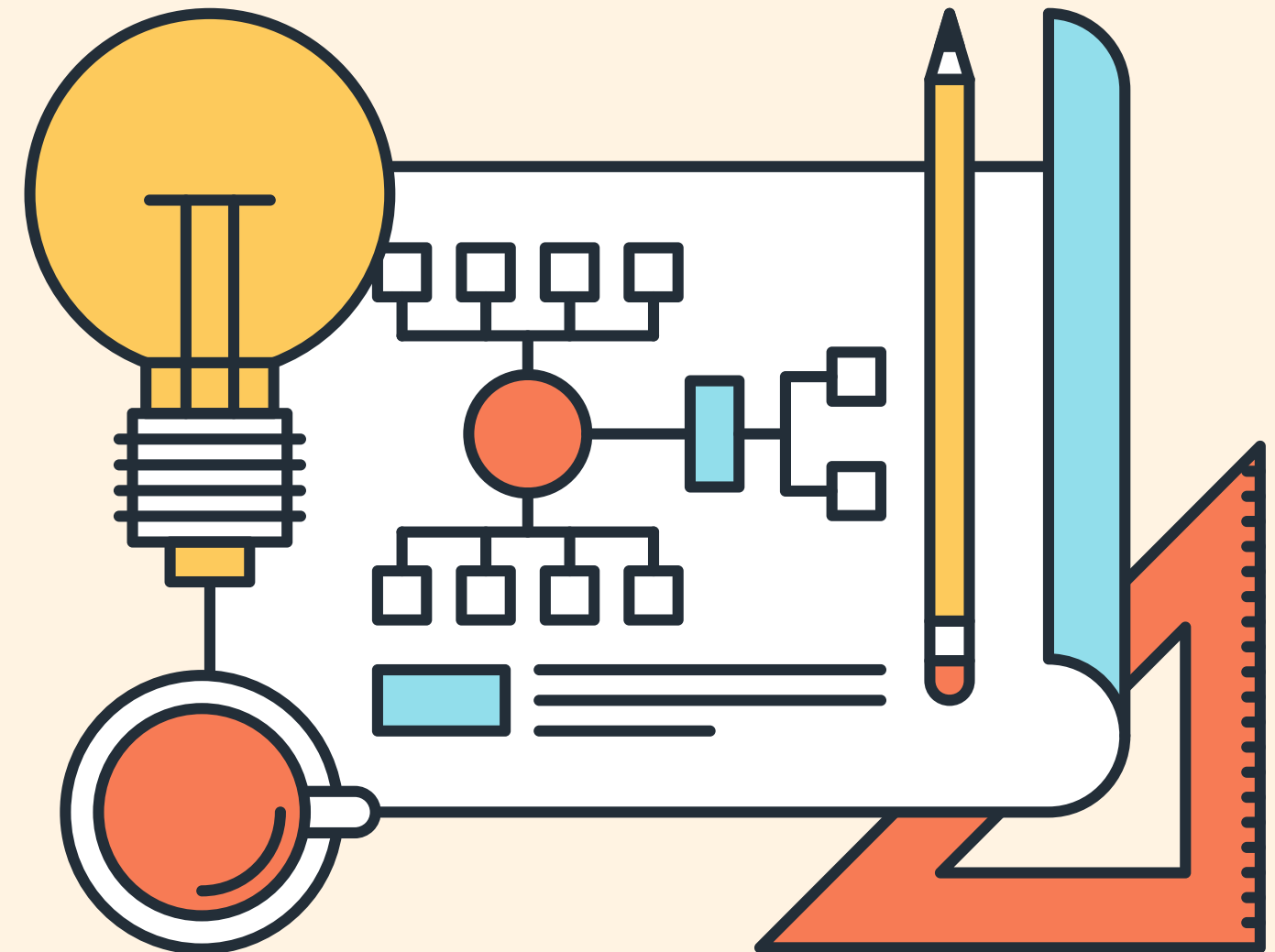
"I have this new stuff I want to merge in to the master branch...what do you all think about it?"



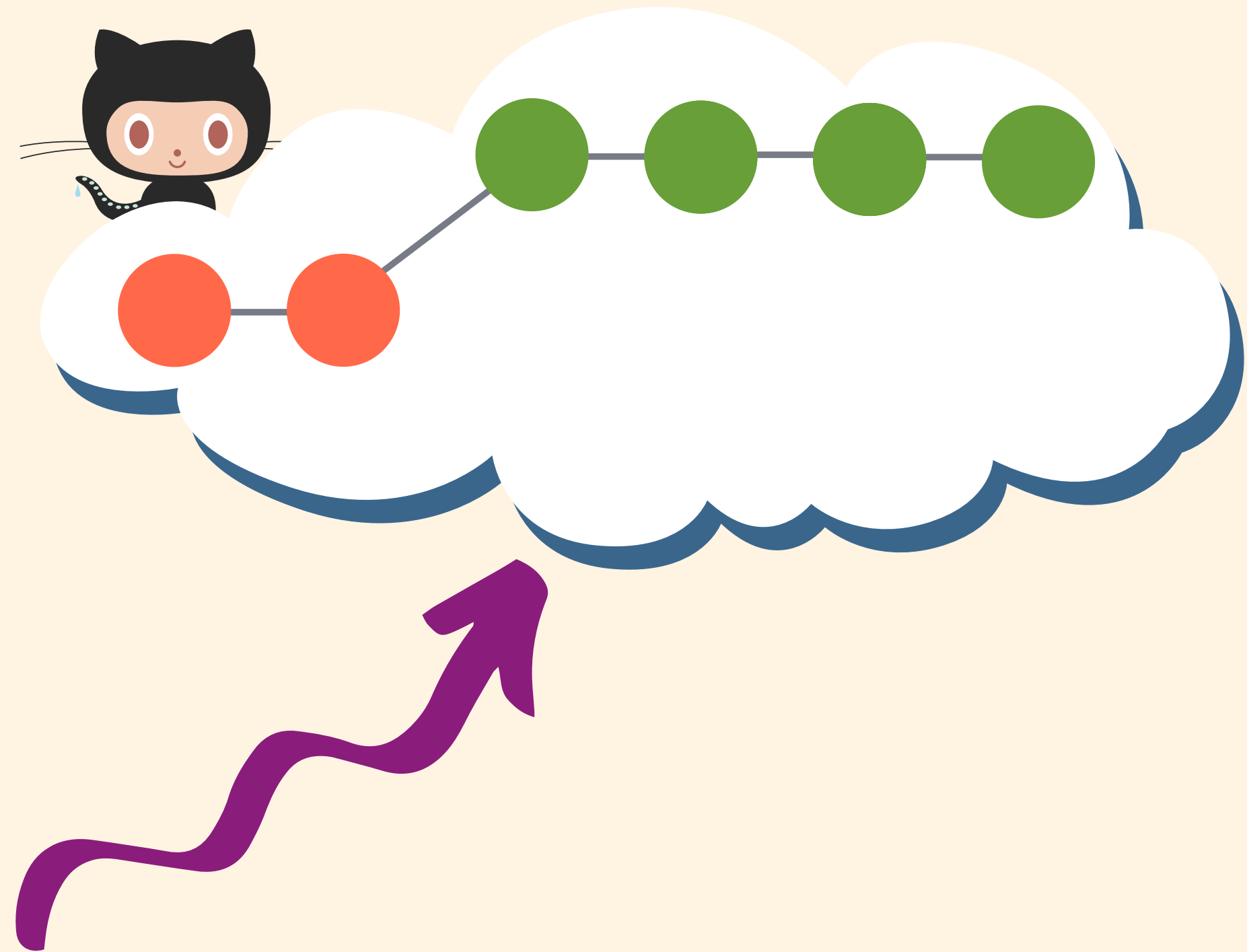
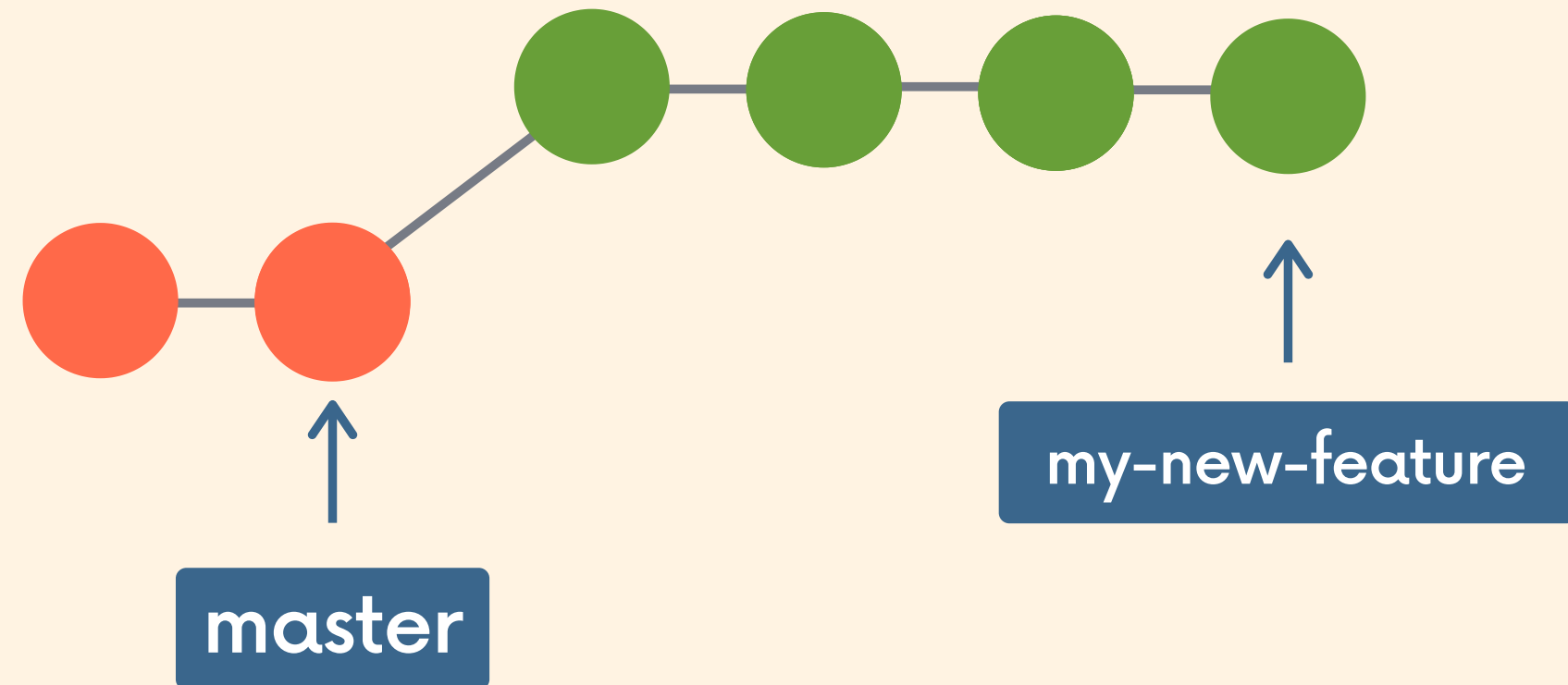


# The Workflow

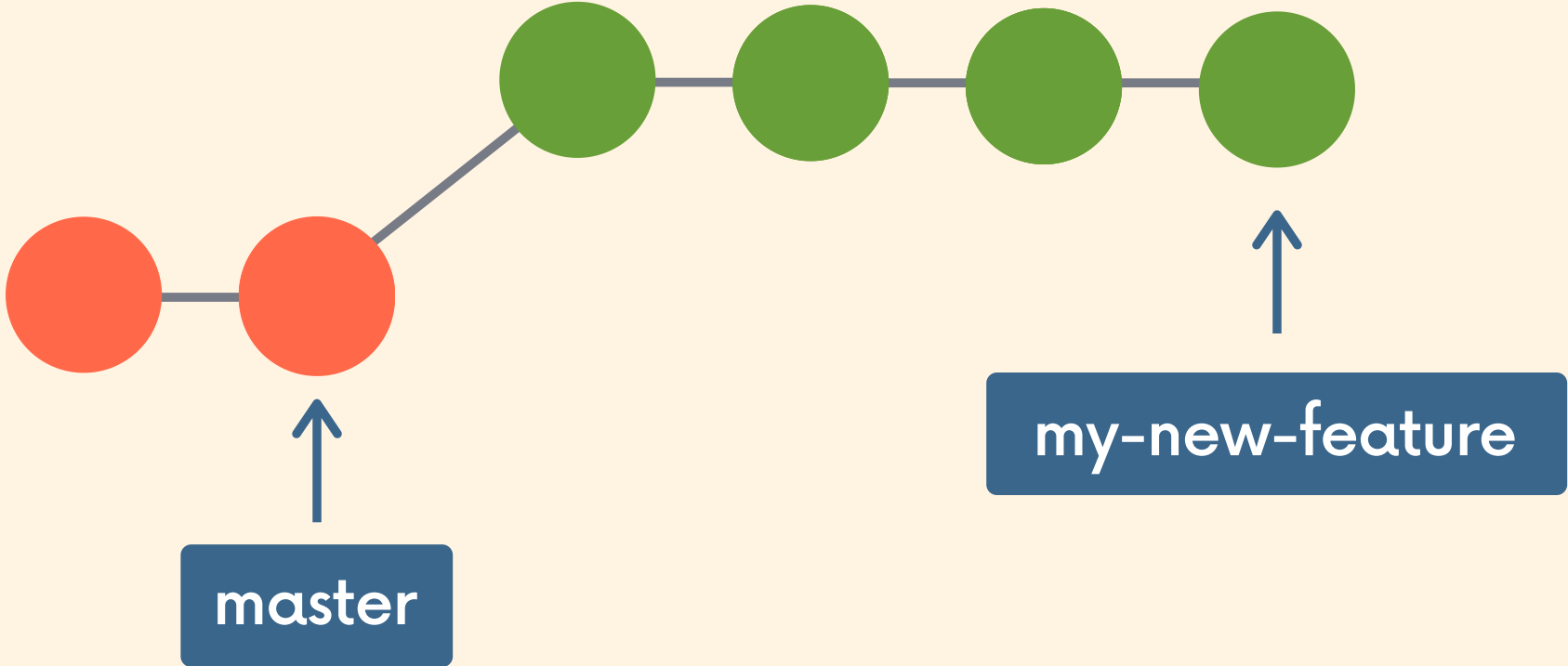
1. Do some work locally on a feature branch
2. Push up the feature branch to Github
3. Open a pull request using the feature branch just pushed up to Github
4. Wait for the PR to be approved and merged. Start a discussion on the PR. This part depends on the team structure.



I push my feature branch up to Github, so that I can open a Pull Request



# My Github



my-new-feature had recent pushes less than a minute ago

[Compare & pull request](#)

my-new-feature ▾

5 branches 0 tags

[Go to file](#)

[Add file ▾](#)

[Code ▾](#)

This branch is 2 commits ahead of master.

[Pull request](#) [Compare](#)

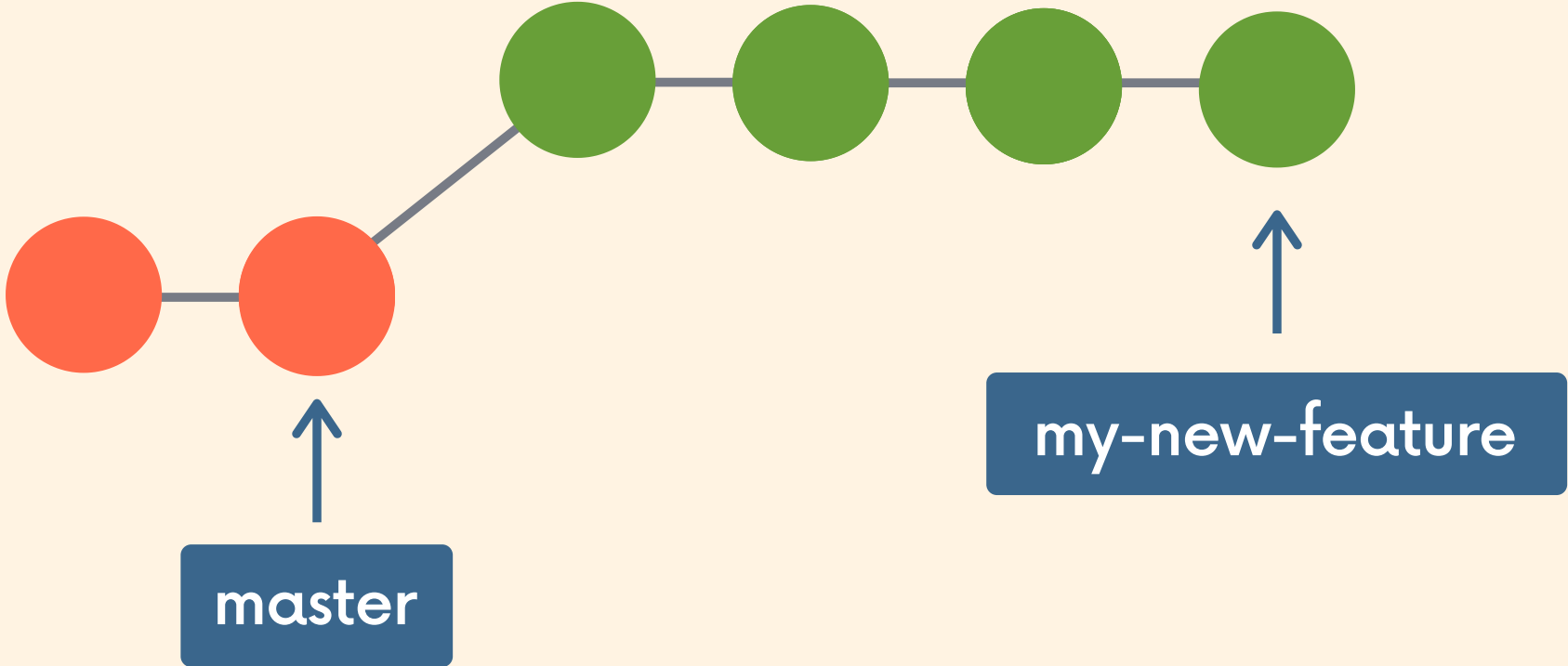
**Colt Steele** and **Colt Steele** add another new file

d3faa08 32 seconds ago 9 commits

anotherNewFile.txt	add another new file	32 seconds ago
newfeature.txt	add new feature	1 minute ago
playlist.txt	merge	5 days ago



# My Github



my-new-feature had recent pushes less than a minute ago [Compare & pull request](#)

my-new-feature ▾ 5 branches 0 tags [Go to file](#) [Add file ▾](#) [Code ▾](#)

This branch is 2 commits ahead of master. [Pull request](#) [Compare](#)


	<b>Colt Steele</b> and <b>Colt Steele</b> add another new file	d3f
	anotherNewFile.txt	add another new file
	newfeature.txt	add new feature
	playlist.txt	merge


I click the PR button

# My Github

## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

 base: master ▾ ← compare: my-new-feature ▾ ✓ **Able to merge.** These branches can be automatically merged.



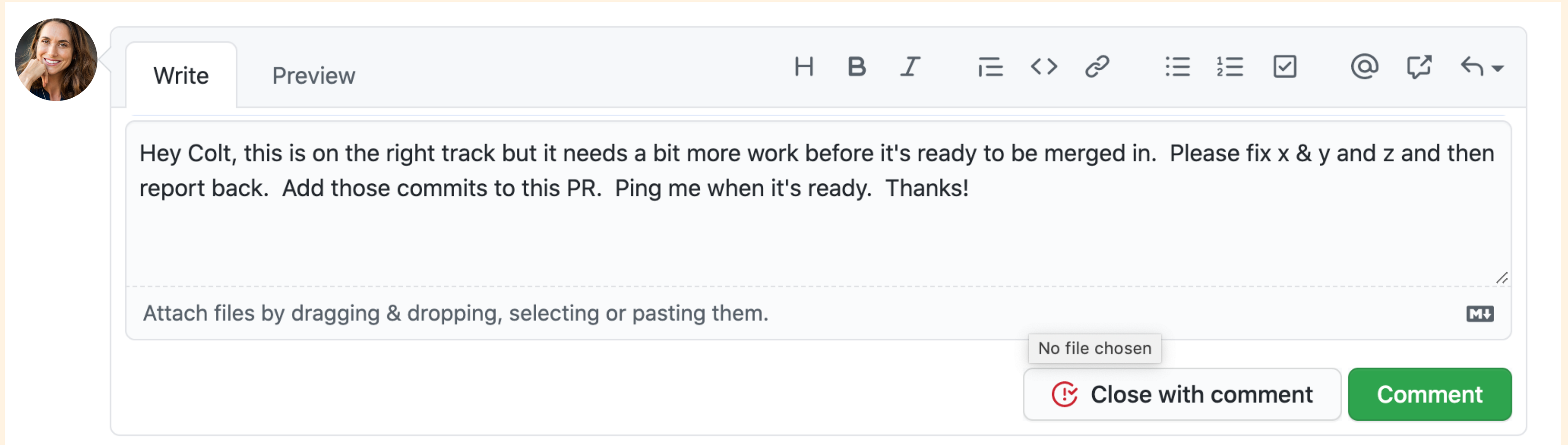
Write Preview H B *I* ≡ <> 🔗 ☰ ☰ ☑ @ 🗨 ↶

Leave a comment

Attach files by dragging & dropping, selecting or pasting them. 📎

Create pull request

# My Boss's Github...



The screenshot shows a GitHub comment interface. On the left is a circular profile picture of a woman. The comment area has two tabs: 'Write' (active) and 'Preview'. A rich text editor toolbar is visible with icons for bold, italic, code, link, list, and other formatting options. The main text area contains the following message: "Hey Colt, this is on the right track but it needs a bit more work before it's ready to be merged in. Please fix x & y and z and then report back. Add those commits to this PR. Ping me when it's ready. Thanks!". Below the text is a dashed line and a file upload area with the text "Attach files by dragging & dropping, selecting or pasting them." and a "M↓" icon. At the bottom right, there is a "No file chosen" button, a "Close with comment" button with a red warning icon, and a green "Comment" button.

My boss leaves some feedback for me. She asks me to make a couple changes before she merges the pull request.



**Colt** commented 2 minutes ago • edited ▾



Added in new feature. This should be descriptive and helpful.



**Colt Steele** added 2 commits 6 days ago



add indie songs

399d0b9



add hot chip

75e6c6d



**Boss** commented 44 seconds ago

Author



Hey Colt, this is on the right track but it needs a bit more work before it's ready to be merged in. Please fix x & y and z and then report back. Add those commits to this PR. Ping me when it's ready. Thanks!



**Colt** commented 21 seconds ago

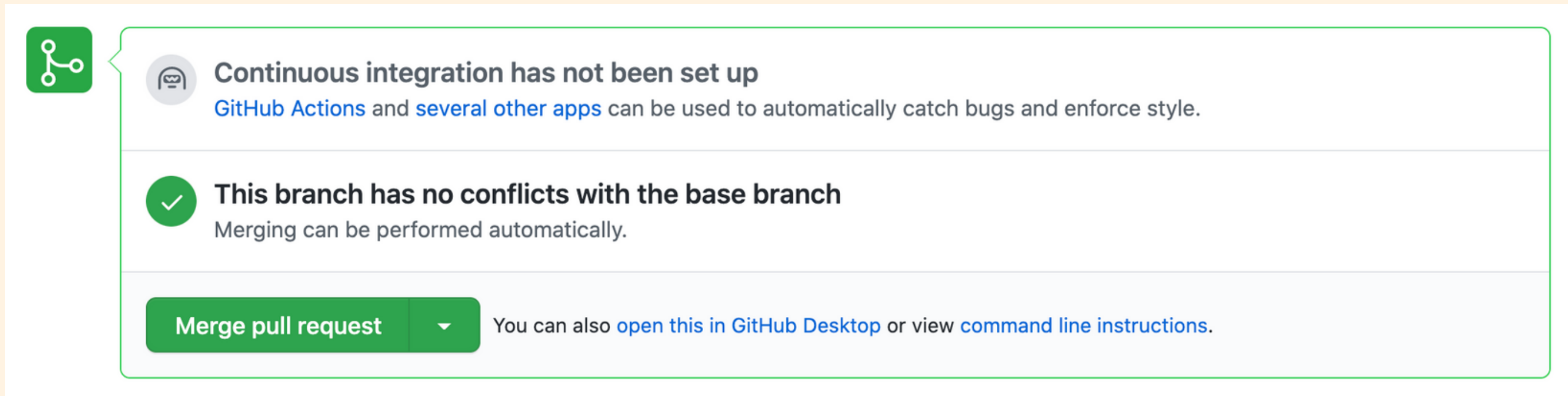
Author



Ok, sounds good! I'll get to work on fixing x & y & z!

I can respond! We can discuss and give feedback!

# My Boss's Github...

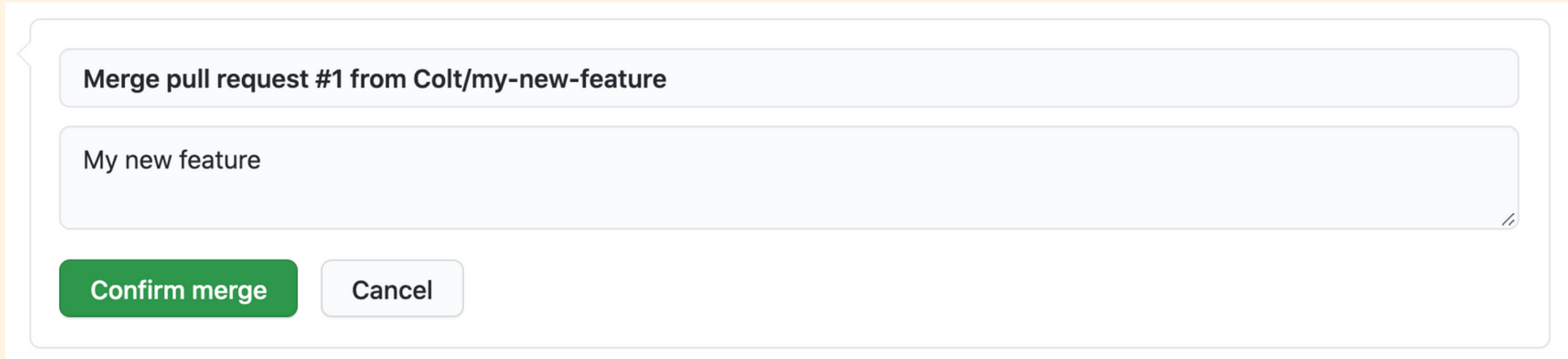


The screenshot shows a GitHub pull request interface. On the left, there is a green icon of a branching diagram. The main content area is divided into three sections:

- Continuous integration has not been set up**  
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.
- This branch has no conflicts with the base branch**  
Merging can be performed automatically.
- Merge pull request** (button) with a dropdown arrow. To the right of the button, it says: "You can also [open this in GitHub Desktop](#) or view [command line instructions](#)."

Once I make the requested changes, my boss (or whoever is in charge of merging) can merge in my pull request!

# My Boss's Github...



Merge pull request #1 from Colt/my-new-feature

My new feature

**Confirm merge** Cancel

The image shows a GitHub merge pull request dialog box. It has a title bar that says "Merge pull request #1 from Colt/my-new-feature". Below the title bar is a text input field containing the text "My new feature". At the bottom of the dialog box, there are two buttons: a green button labeled "Confirm merge" and a white button labeled "Cancel".

Once I make the requested changes, my boss (or whoever is in charge of merging) can merge in my pull request!

The above text will be used in the resulting merge commit.

# My Boss's Github...



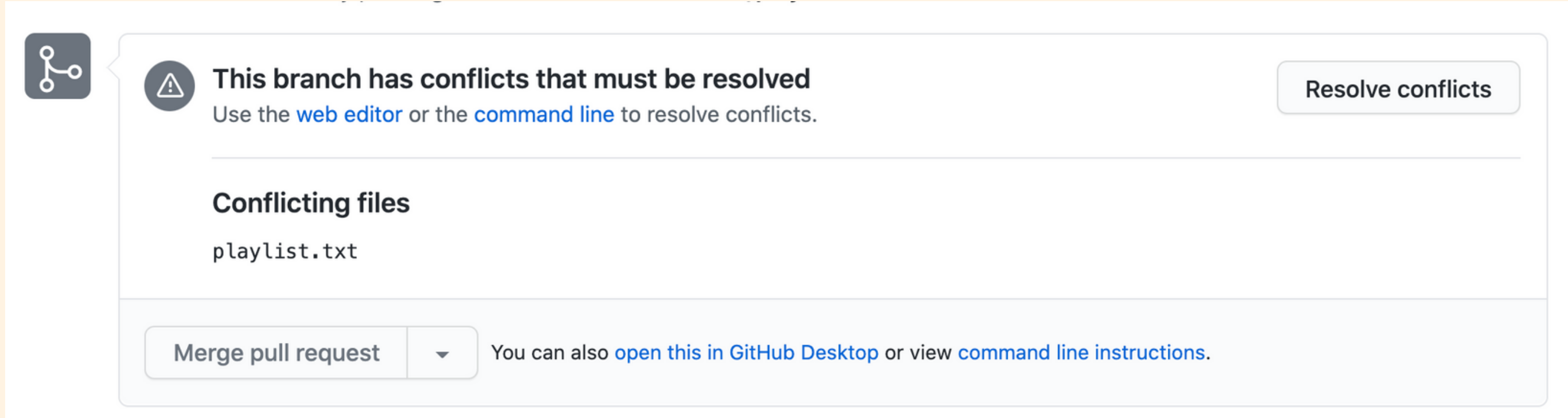
## Pull request successfully merged and closed

You're all set—the `my-new-feature` branch can be safely deleted.

Delete branch

The changes from the my-new-feature branch have now been merged into the master branch!!!

# My Boss's Github...



The screenshot shows a GitHub pull request interface. On the left, there is a dark square icon with a white branching diagram. To its right is a warning icon (a triangle with an exclamation mark) followed by the text "This branch has conflicts that must be resolved". Below this, it says "Use the [web editor](#) or the [command line](#) to resolve conflicts." In the top right corner of the notification box is a button labeled "Resolve conflicts". Below the main text is a section titled "Conflicting files" with the filename "playlist.txt" listed underneath. At the bottom left of the notification box is a button labeled "Merge pull request" with a dropdown arrow. To the right of this button is the text "You can also [open this in GitHub Desktop](#) or view [command line instructions](#)."

Just like any other merge, **sometimes there are conflicts** that need to be resolved when merging a pull request. This is fine. Don't panic.

You can perform the merge and fix the conflicts on the command line like normal, or you can use Github's interactive editor.



# My Boss's Local Machine

My boss can merge the branch and resolve the conflicts locally...

Switch to the branch in question. Merge in master and resolve the conflicts.

```
git fetch origin  
git switch my-new-feature  
git merge master  
fix conflicts!
```

Switch to master. Merge in the feature branch (now with no conflicts). Push changes up to Github.

```
git switch master  
git merge my-new-feature  
git push origin master
```

# Don't Worry

Github Gives You Instructions  
If You Forget What To Do!





### Continuous integration has not been set up

[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.



### This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

#### ✓ Create a merge commit

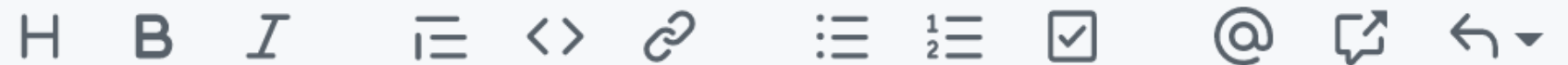
All commits from this branch will be added to the base branch via a merge commit.

#### Squash and merge

The 3 commits from this branch will be combined into one commit in the base branch.

#### Rebase and merge

The 3 commits from this branch will be rebased and added to the base branch.



When merging, we can perform a Squash & Merge which will reduce all the commits from the feature branch down into a single commit on master.

isting them.



Close pull request

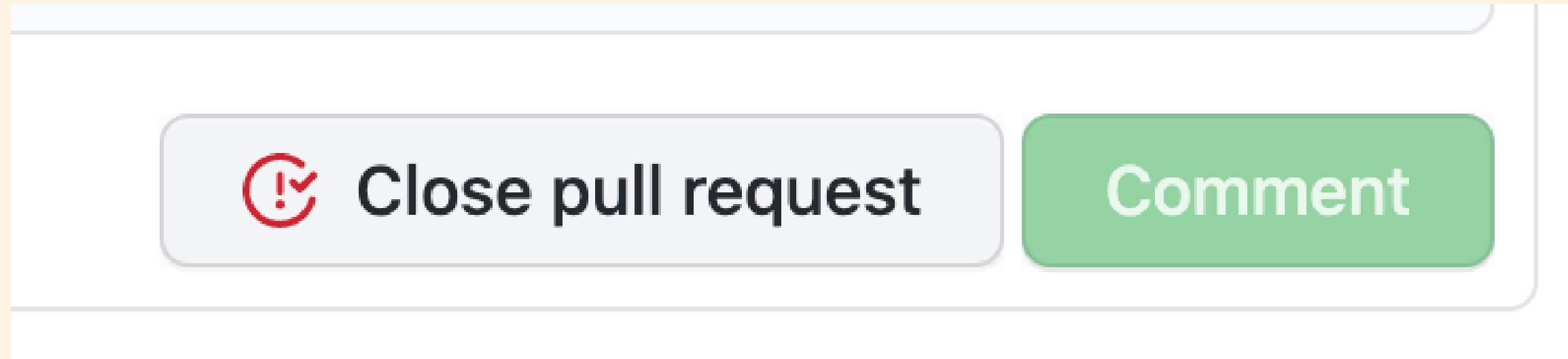
Comment



Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

# My Boss's Github...

My boss can also decide to close my PR instead :(





# Recap-ing Pull Requests

**Pull Requests** are a fancy way of requesting changes from one branch be merged into another branch.

Tools like Github & Bitbucket allow us to generate pull requests via an online interface. Team members can then view the changes and decide to merge them in or reject them. PR's also provide a place to discuss the changes and provide feedback.

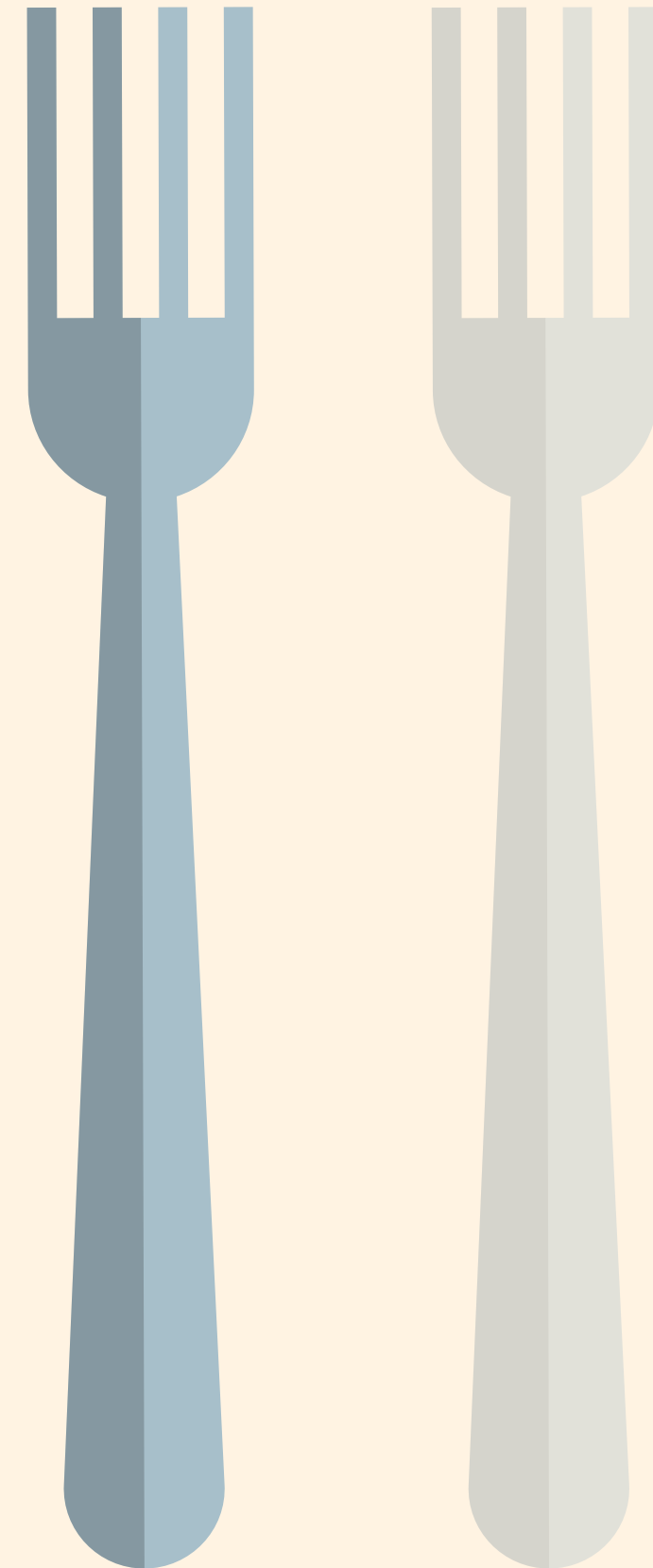




# Fork & Clone: Another Workflow

The "fork & clone" workflow is different from anything we've seen so far. Instead of just one centralized Github repository, every developer has their own Github repository in addition to the "main" repo. Developers make changes and push to their own forks before making pull requests.

It's very commonly used on large open-source projects where there may be thousands of contributors with only a couple maintainers.



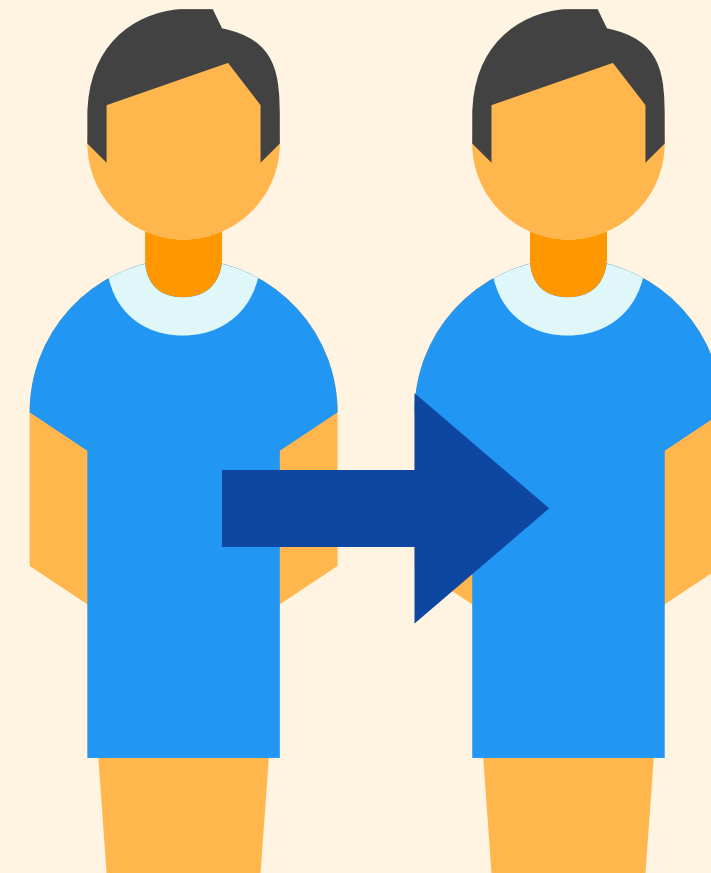


# Forking

Github (and similar tools) allow us to create personal copies of other peoples' repositories. We call those copies a "fork" of the original.

When we fork a repo, we're basically asking Github "Make me my own copy of this repo please"

As with pull requests, forking is not a Git feature. The ability to fork is implemented by Github.





# This repo is not mine. I want a copy!

aapatre / Automatic-Udemy-Course-Enroller-GET-PAID-UDEMY-COURSES-for-FREE

Watch 52 Star 1k Fork 155

Code Issues 3 Pull requests 1 Discussions Actions Projects 1 Wiki Security Insights

master 2 branches 6 tags Go to file Add file Code

aapatre Update README.md 2 35e5cb1 4 days ago 345 commits

.github	Merge pull request #146 from aapatre/develop	6 days ago
core	Wait for enroll button to be clickable after selecting state/province	10 days ago
tests	Adding unittests	13 days ago
.coveragerc	Adding more unittests	23 days ago
.deepsources.toml	Create .deepsources.toml	16 days ago
.gitignore	fixed deepsource ignore	16 days ago
.restyled.yaml	Adding restyled configuration	last month
CHANGELOG.md	fixed typo	6 days ago
CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.md	15 days ago
CONTRIBUTING.md	Create CONTRIBUTING.md	15 days ago
LICENSE	Initial commit	3 months ago
README.md	Update README.md	4 days ago
SECURITY.md	Update SECURITY.md	15 days ago
logconfig.ini	Added logging	13 days ago
pull_request_template.md	Create pull_request_template.md	15 days ago
pyproject.toml	Bump version in pyproject	18 days ago
pytest.ini	Adding more OS tests	15 days ago

### About

Do you want to LEARN NEW STUFF for FREE? Don't worry, with the power of web-scraping and automation, this script will find the necessary Udemy coupons & enroll you for PAID UDEMY COURSES, ABSOLUTELY FREE!

Readme  
GPL-3.0 License

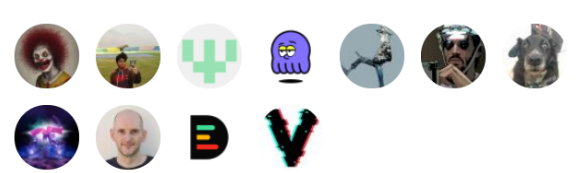
### Releases 6

v1.0.0 Latest  
6 days ago  
+ 5 releases

### Packages

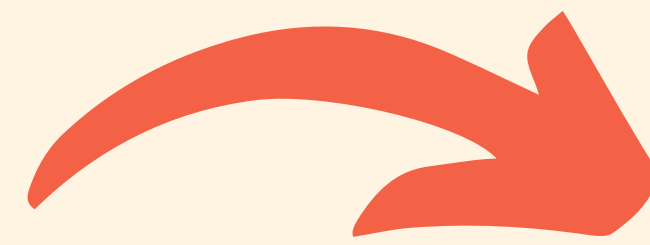
No packages published

### Contributors 11





# I click the "fork" button



aapatre / Automatic-Udemy-Course-Enroller-GET-PAID-UDEMY-COURSES-for-FREE

Watch 52 Star 1k Fork 155

Code Issues 3 Pull requests 1 Discussions Actions Projects 1 Wiki Security Insights

master 2 branches 6 tags Go to file Add file Code

aapatre Update README.md 2 35e5cb1 4 days ago 345 commits

.github	Merge pull request #146 from aapatre/develop	6 days ago
core	Wait for enroll button to be clickable after selecting state/province	10 days ago
tests	Adding unittests	13 days ago
.coveragerc	Adding more unittests	23 days ago
.deepsources.toml	Create .deepsources.toml	16 days ago
.gitignore	fixed deepsource ignore	16 days ago
.restyled.yaml	Adding restyled configuration	last month
CHANGELOG.md	fixed typo	6 days ago
CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.md	15 days ago
CONTRIBUTING.md	Create CONTRIBUTING.md	15 days ago
LICENSE	Initial commit	3 months ago
README.md	Update README.md	4 days ago
SECURITY.md	Update SECURITY.md	15 days ago
logconfig.ini	Added logging	13 days ago
pull_request_template.md	Create pull_request_template.md	15 days ago
pyproject.toml	Bump version in pyproject	18 days ago
pytest.ini	Adding more OS tests	15 days ago

**About**

Do you want to LEARN NEW STUFF for FREE? Don't worry, with the power of web-scraping and automation, this script will find the necessary Udemy coupons & enroll you for PAID UDEMY COURSES, ABSOLUTELY FREE!

Readme  
GPL-3.0 License

**Releases** 6

v1.0.0 Latest  
6 days ago

+ 5 releases

**Packages**

No packages published

**Contributors** 11

Watch 52 Star 1k Fork 155



# Now I have my very own copy!

Colt / [Automatic-Udemy-Course-Enroller-GET-PAID-UDEMY-COURSES-for-FREE](#)

forked from [aapatre/Automatic-Udemy-Course-Enroller-GET-PAID-UDEMY-COURSES-for-FREE](#)

Watch 0 Star 0 Fork 156

Code Pull requests Actions Projects Wiki Security Insights Settings

master 2 branches 6 tags Go to file Add file Code

This branch is even with aapatre:master. Pull request Compare

**aapatre** Update README.md 35e5cb1 4 days ago 345 commits

.github	Merge pull request <a href="#">aapatre#146</a> from aapatre/develop	6 days ago
core	Wait for enroll button to be clickable after selecting state/province	10 days ago
tests	Adding unittests	13 days ago
.coveragerc	Adding more unittests	23 days ago
.deepsources.toml	Create .deepsources.toml	16 days ago
.gitignore	fixed deepsource ignore	16 days ago
.restyled.yaml	Adding restyled configuration	last month
CHANGELOG.md	fixed typo	6 days ago
CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.md	15 days ago
CONTRIBUTING.md	Create CONTRIBUTING.md	15 days ago
LICENSE	Initial commit	3 months ago
README.md	Update README.md	4 days ago
SECURITY.md	Update SECURITY.md	15 days ago
logconfig.ini	Added logging	13 days ago

### About

Do you want to LEARN NEW STUFF for FREE? Don't worry, with the power of web-scraping and automation, this script will find the necessary Udemy coupons & enroll you for PAID UDEMY COURSES, ABSOLUTELY FREE!

Readme

GPL-3.0 License

### Releases

6 tags

[Create a new release](#)

### Packages

No packages published

[Publish your first package](#)

### Languages

Python 100.0%

# The original repo...

 [aapatre / Automatic-Udemy-Course-Enroller-GET-PAID-UDEMY-COURSES-for-FREE](#)

# My newly-created fork...

 [Colt / Automatic-Udemy-Course-Enroller-GET-PAID-UDEMY-COURSES-for-FREE](#)

forked from [aapatre/Automatic-Udemy-Course-Enroller-GET-PAID-UDEMY-COURSES-for-FREE](#)



# Now What?

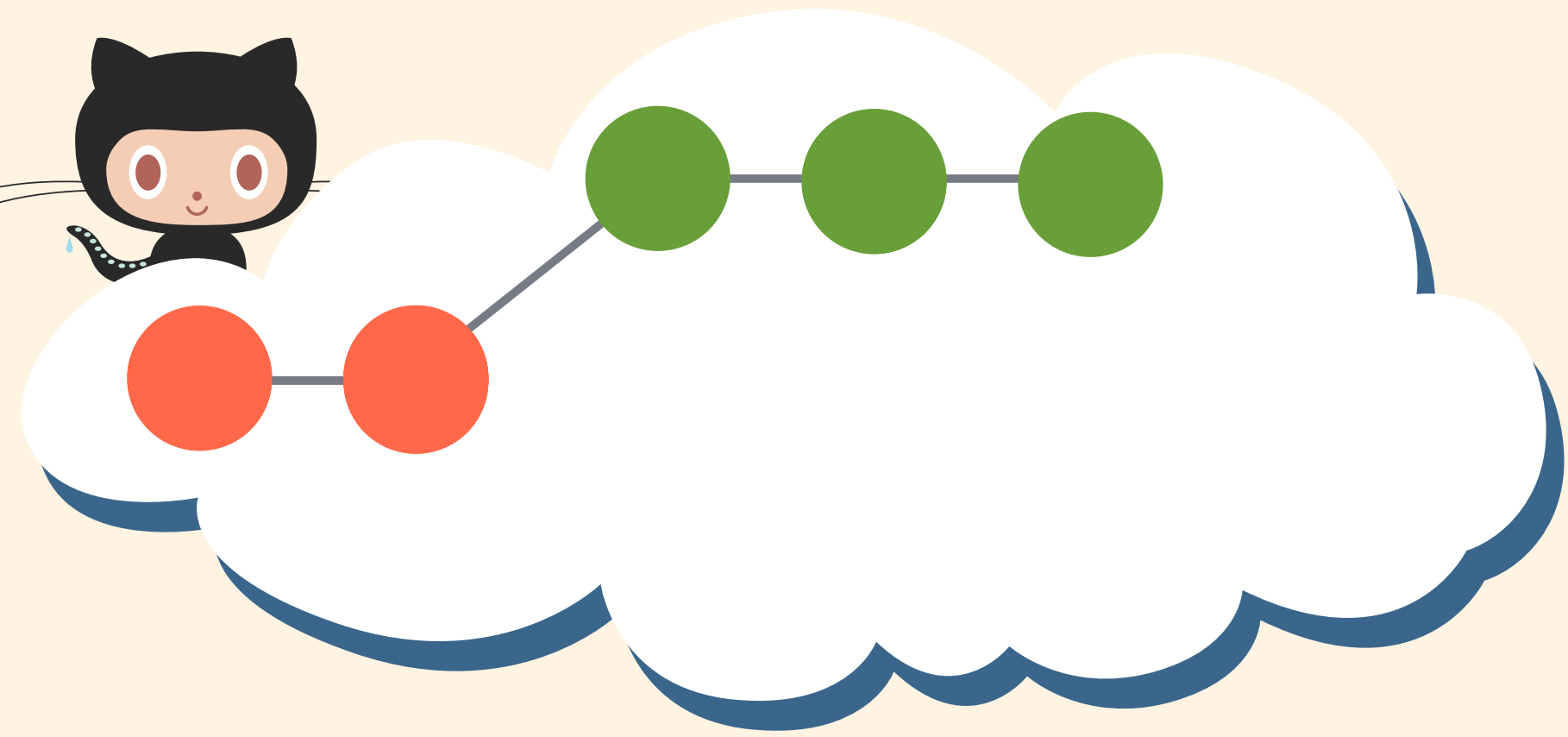
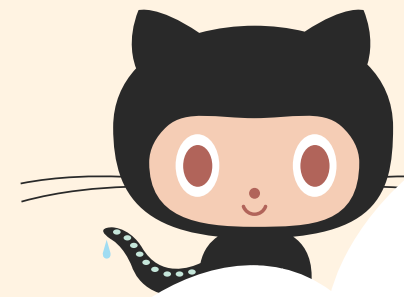
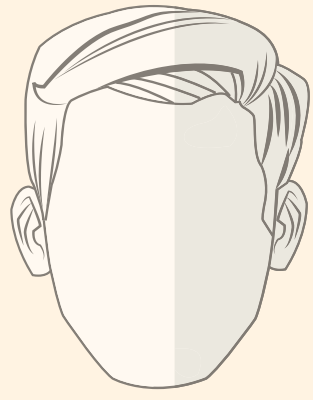
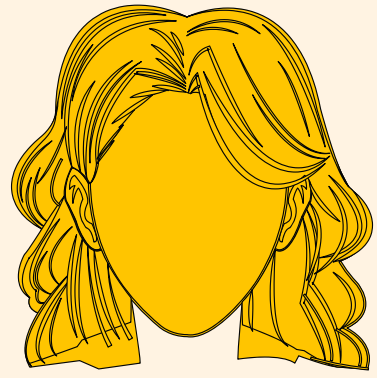
Now that I've forked, I have my very own copy of the repo where I can do whatever I want!

I can clone my fork and make changes, add features, and break things without fear of disturbing the original repository.

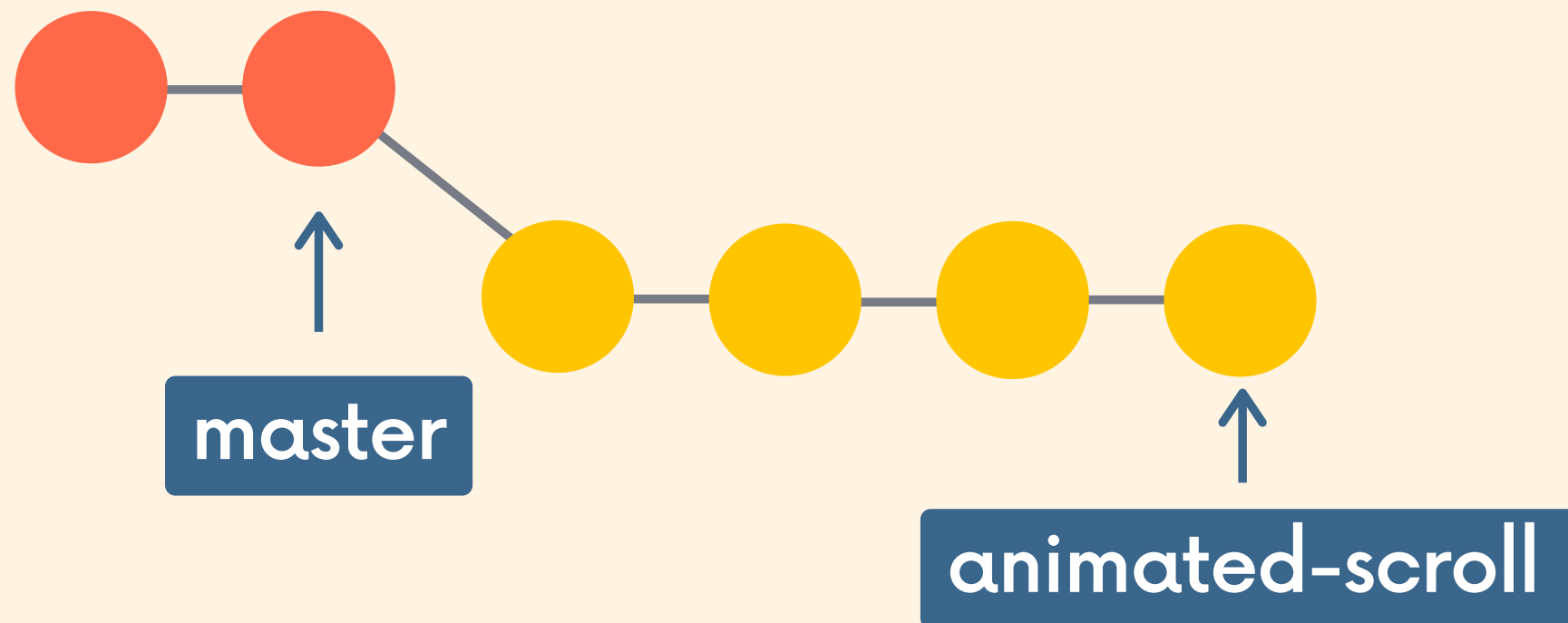
If I do want to share my work, I can make a pull request from my fork to the original repo.





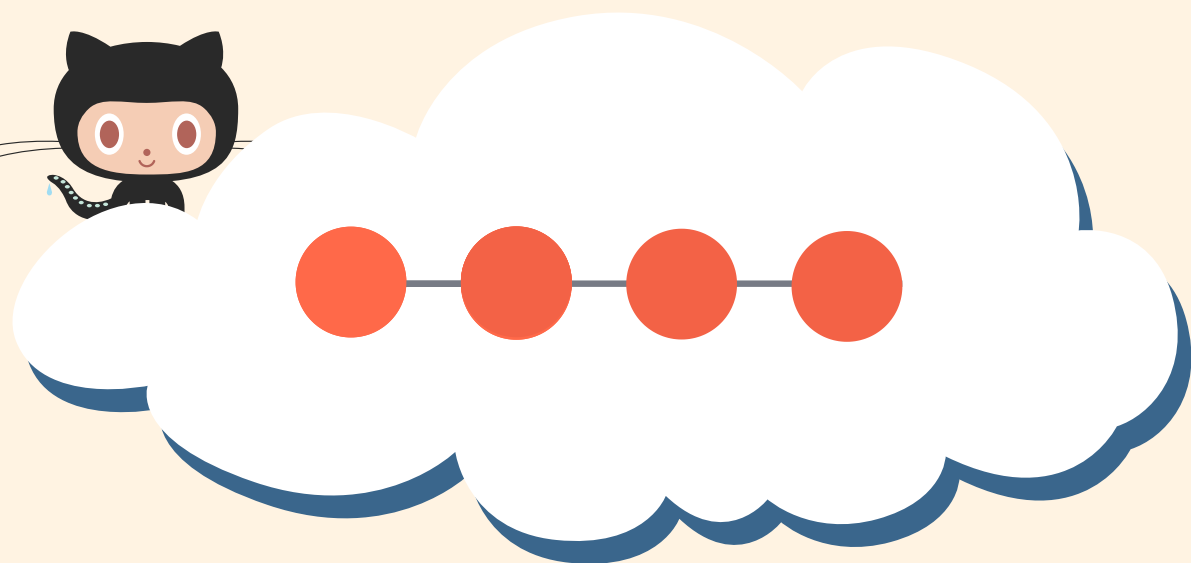
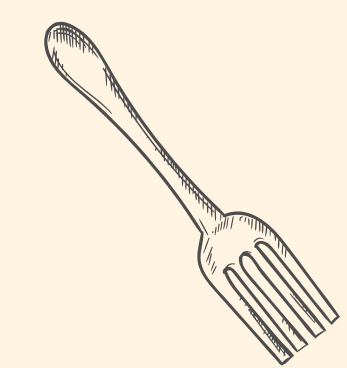
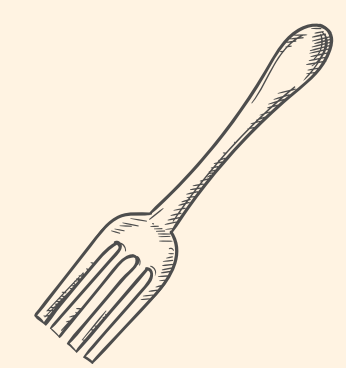


**Pamela** is hard at work on her own new feature. Just like everyone else, she's working on a separate feature branch rather than master.

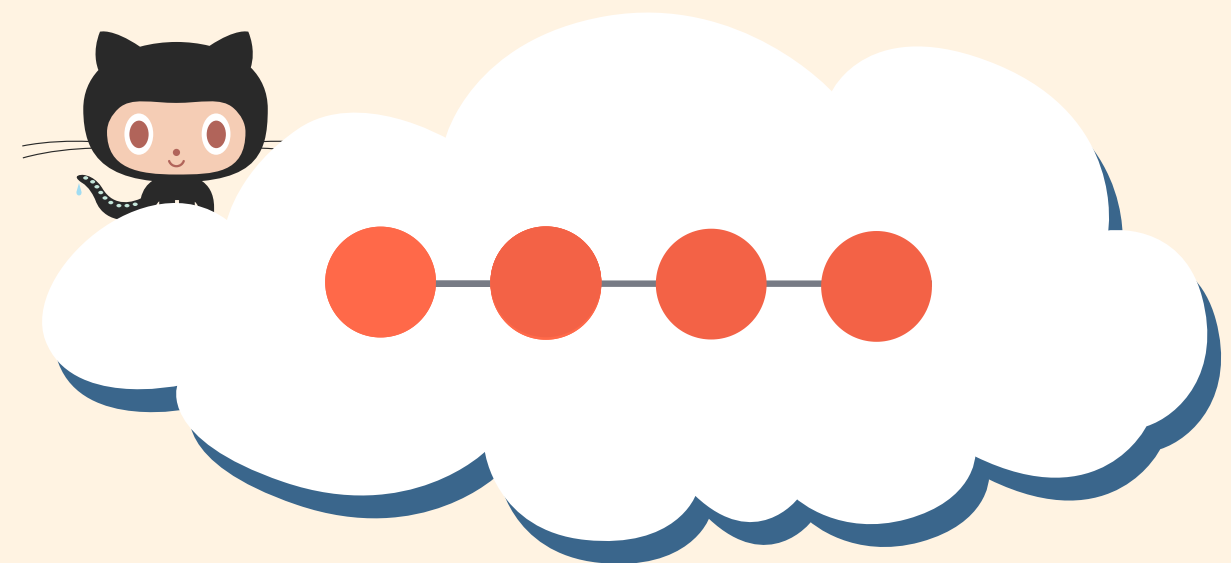




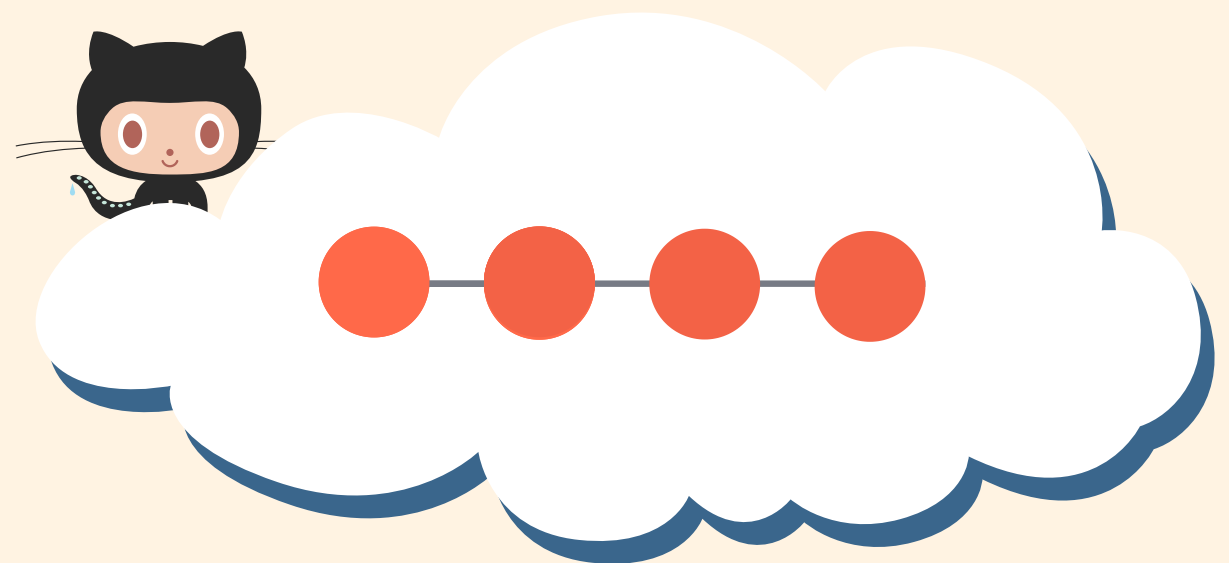
The Official Project Repo



My Fork



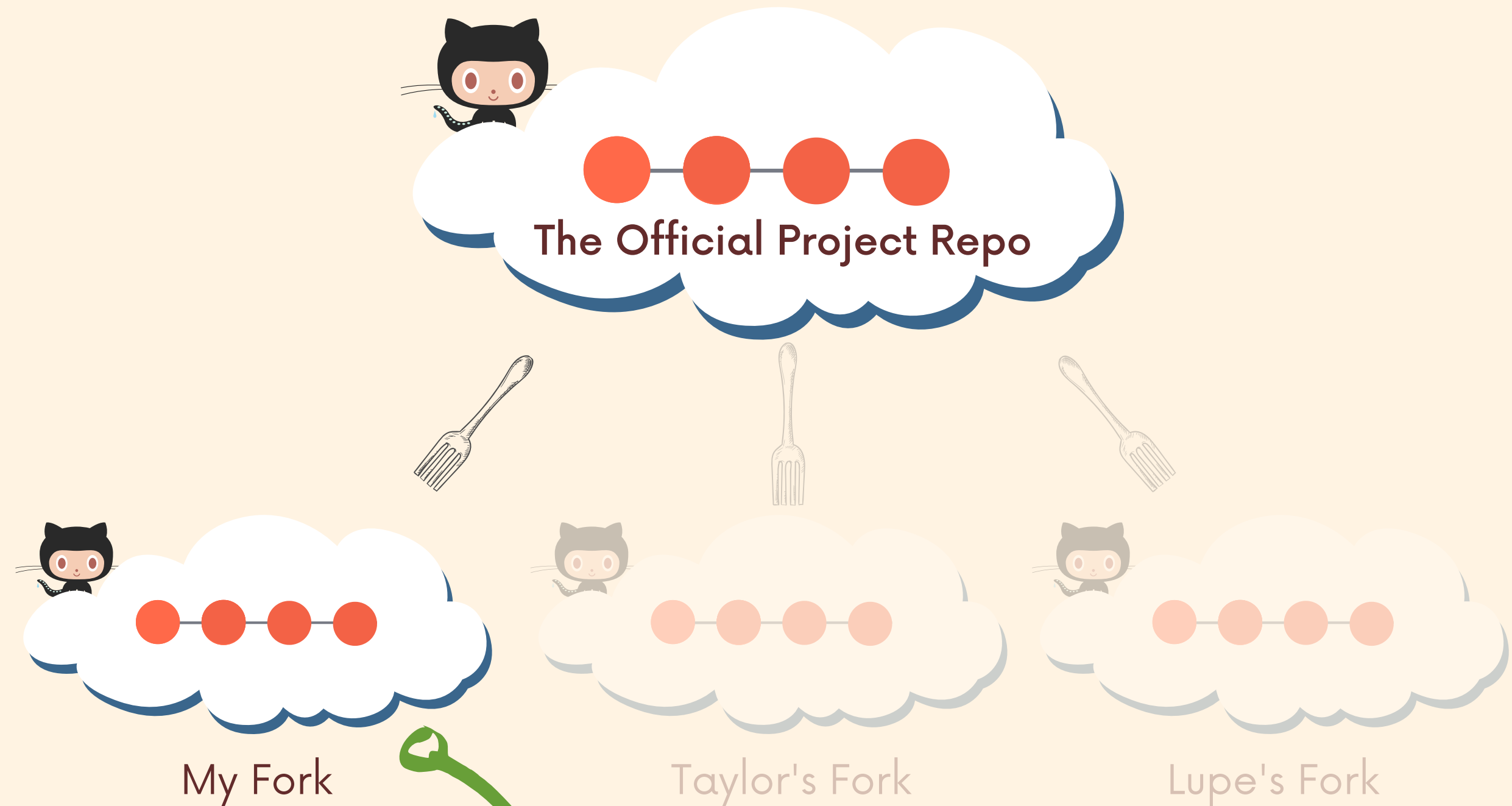
Taylor's Fork



Lupe's Fork

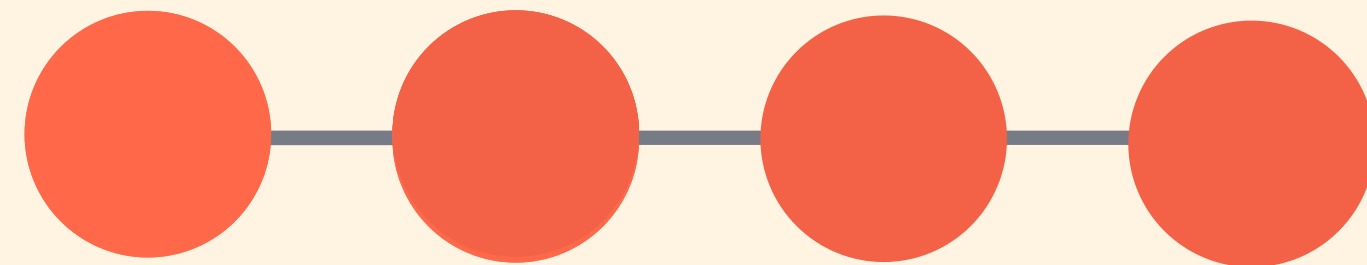


# Github

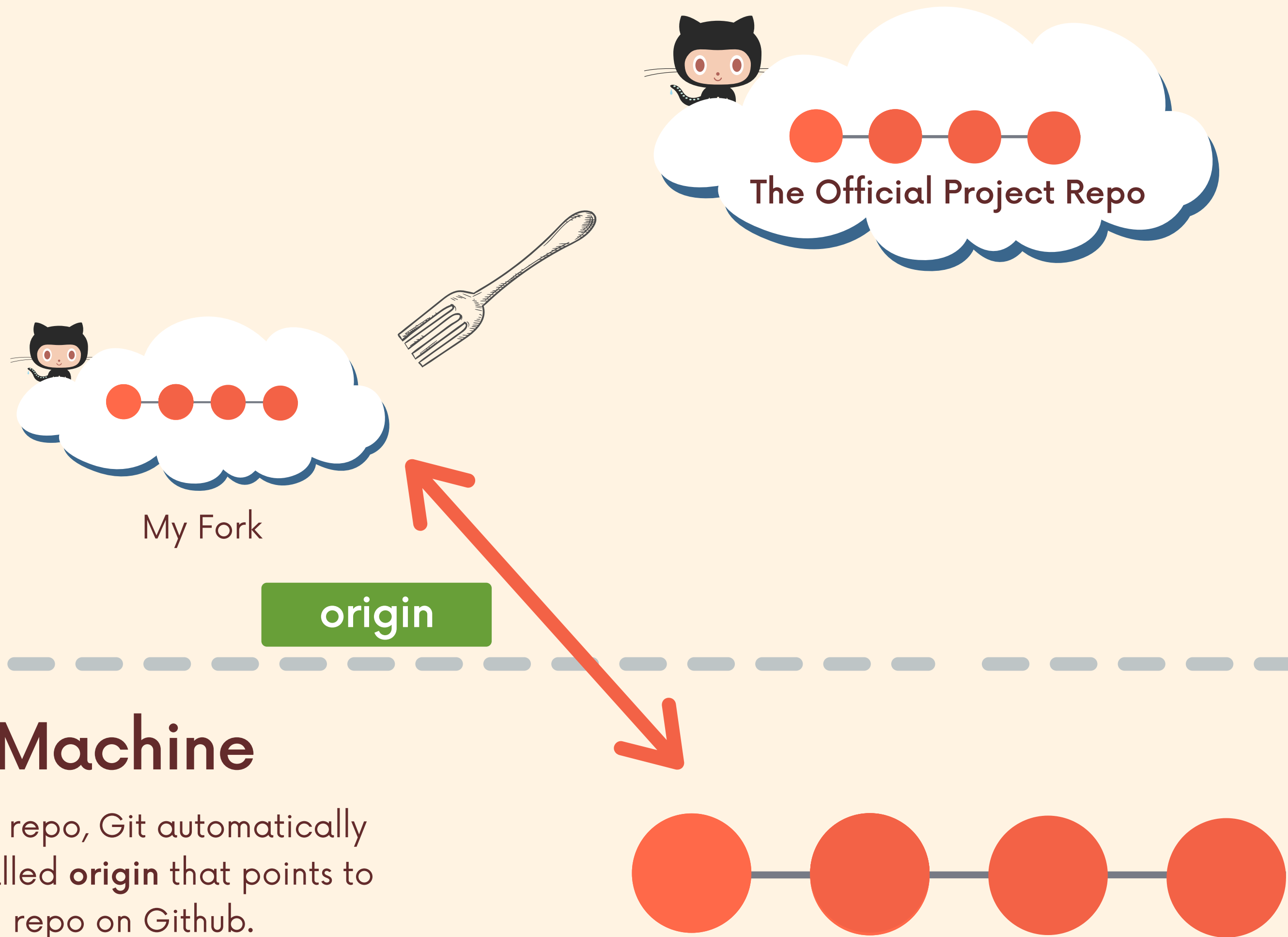


## My Local Machine

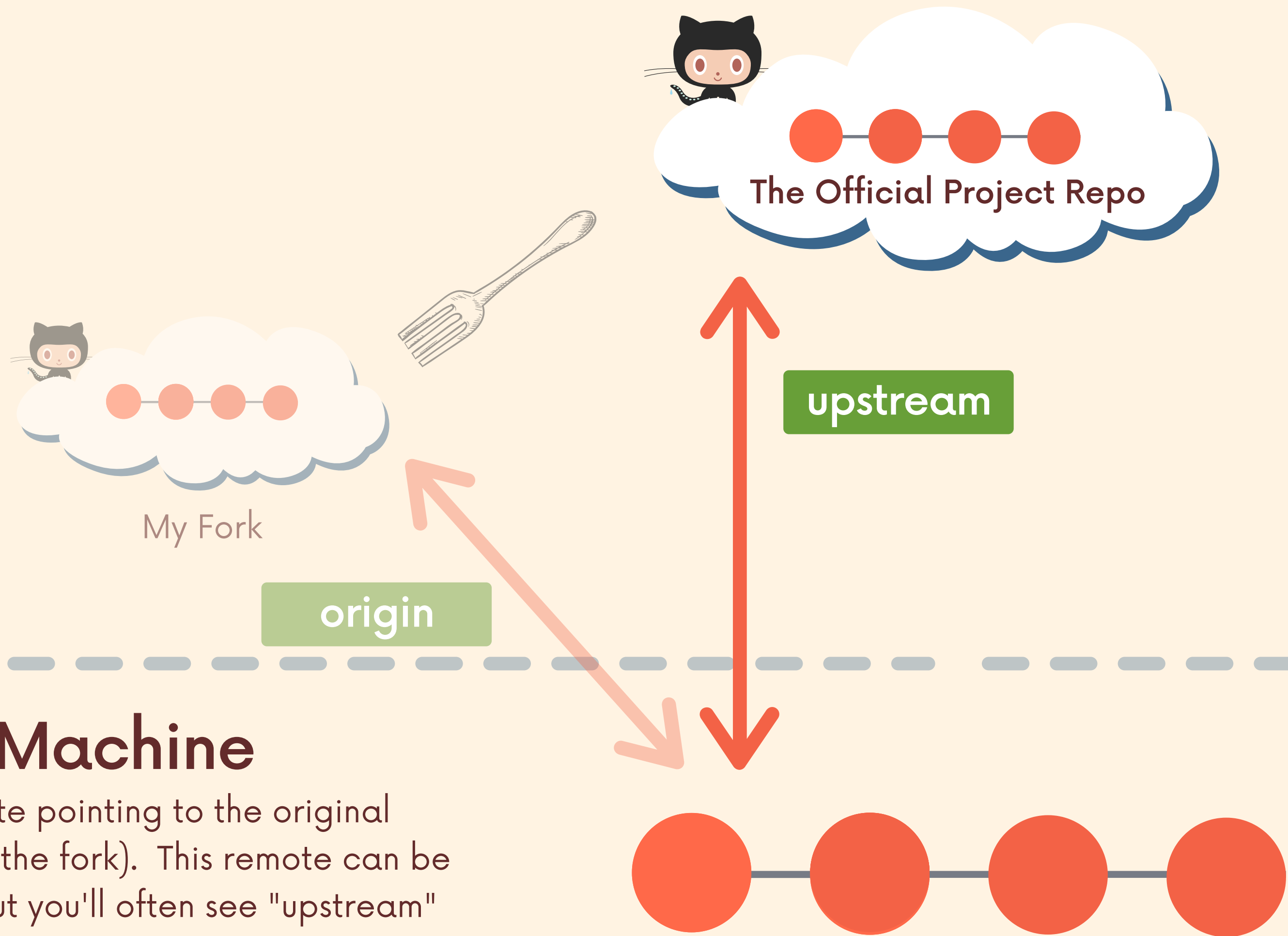
I make a clone of my forked repository so that I can start working on it locally



# Github



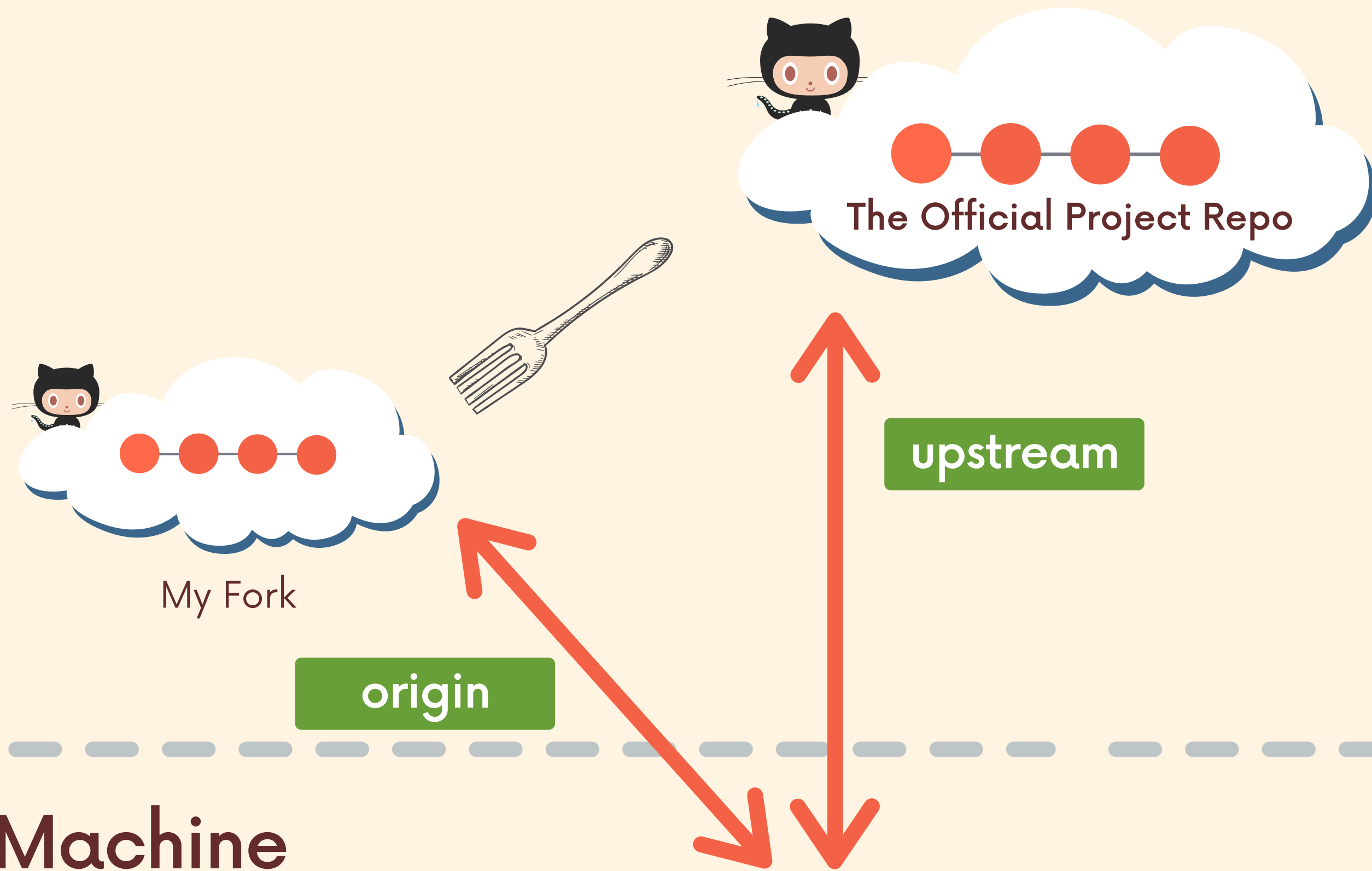
# Github



## My Local Machine

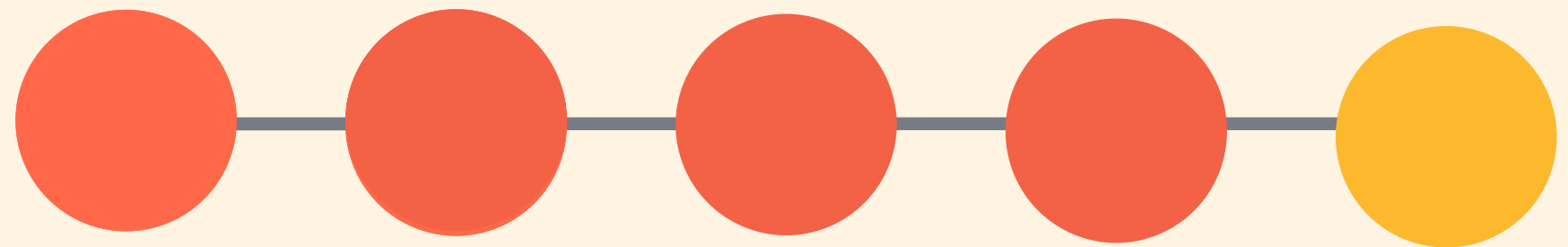
Next, I add a remote pointing to the original project repo (NOT the fork). This remote can be named anything, but you'll often see "upstream" or "original" used.

# Github



## My Local Machine

I do some new work locally. Normally, I would work on a feature branch, but only to keep the diagrams simpler I'm not going to!



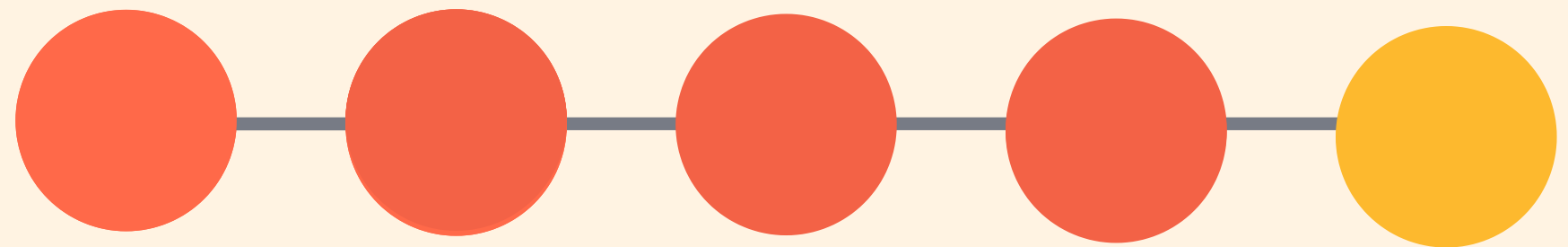
# Github



origin

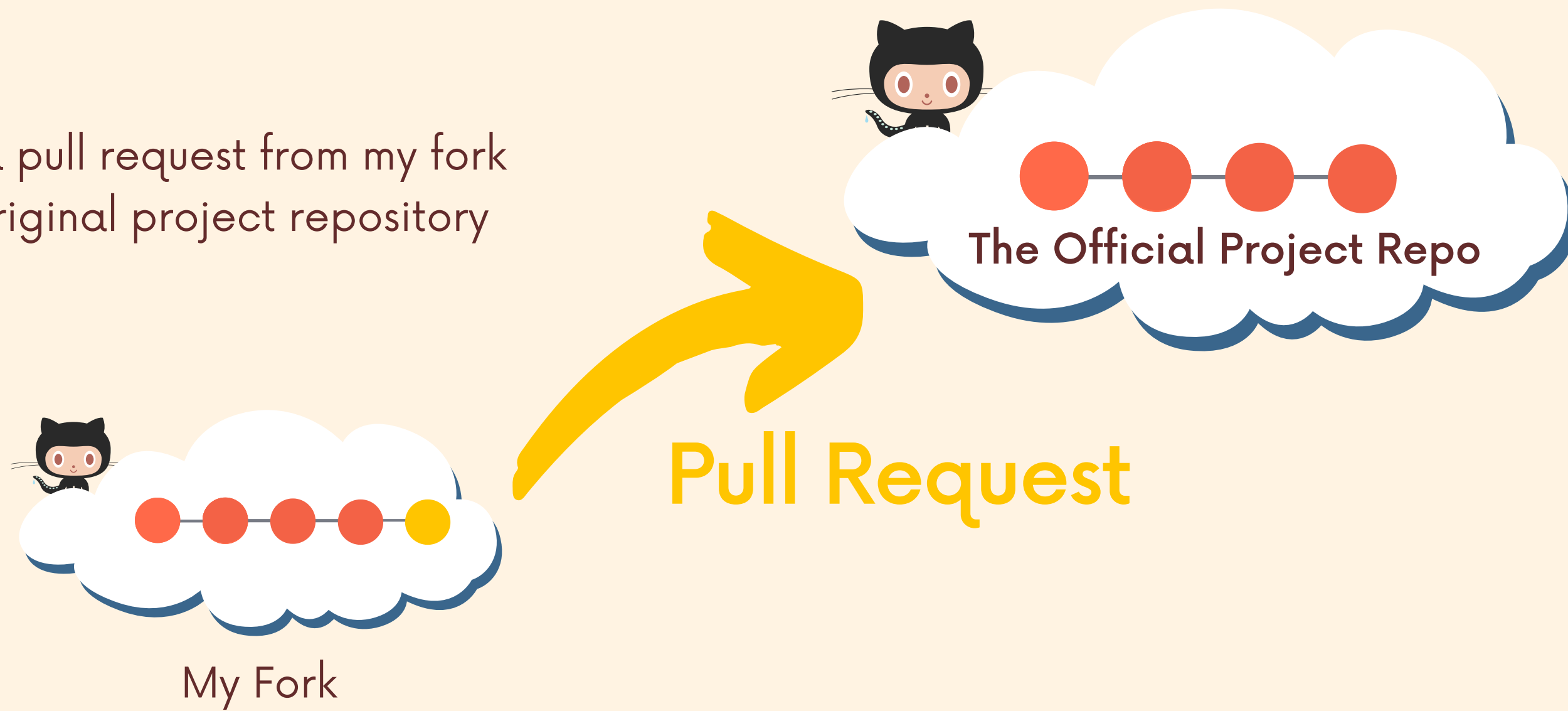
## My Local Machine

To share my changes with others, I cannot push to upstream. I don't have permission!  
But I can **push to origin** (my Github fork)



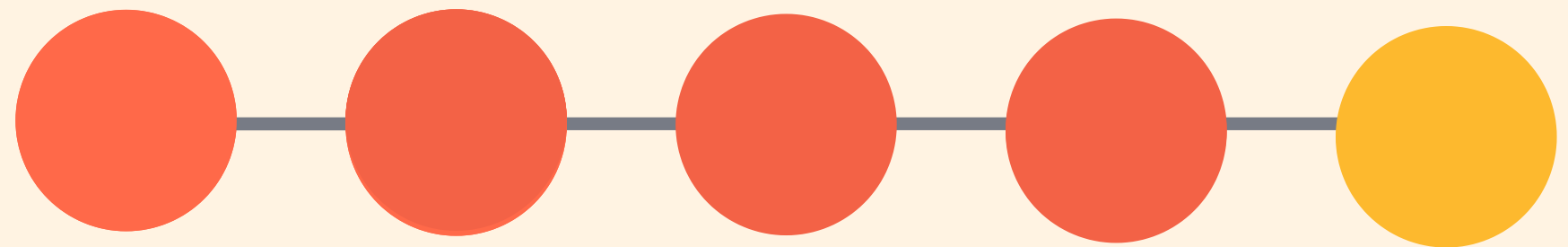
# Github

Next, I can make a pull request from my fork on Github to the original project repository



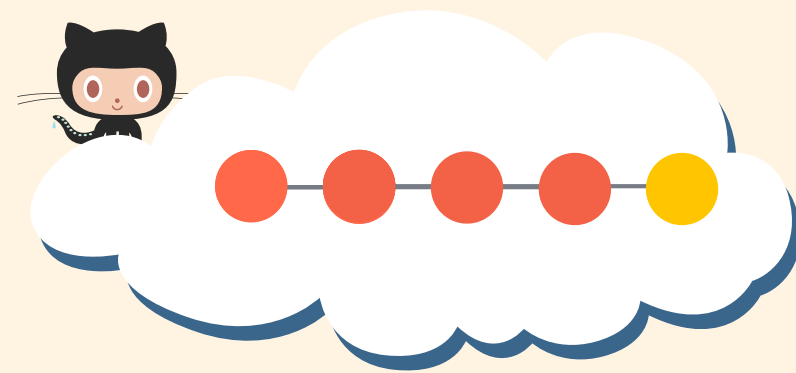
---

## My Local Machine



# Github

Now I wait to hear from the project maintainers! Do they want me to make further changes? It turns out they accept and merge my pull request! Woohoo!



My Fork



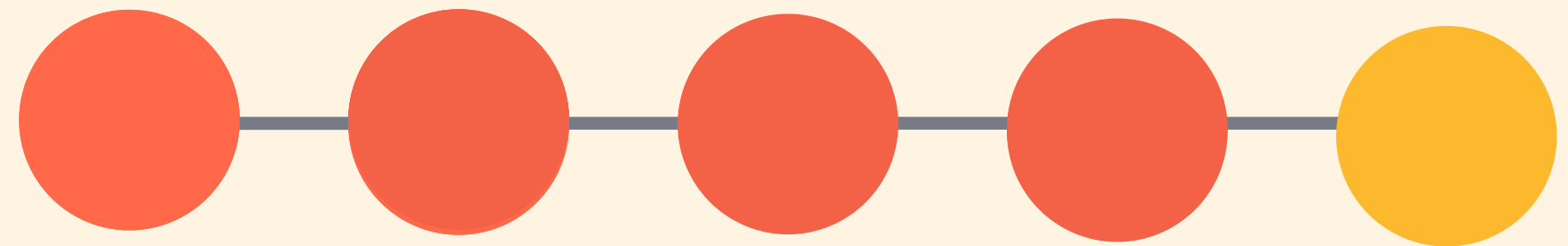
Pull Request Accepted!!!



The Official Project Repo

---

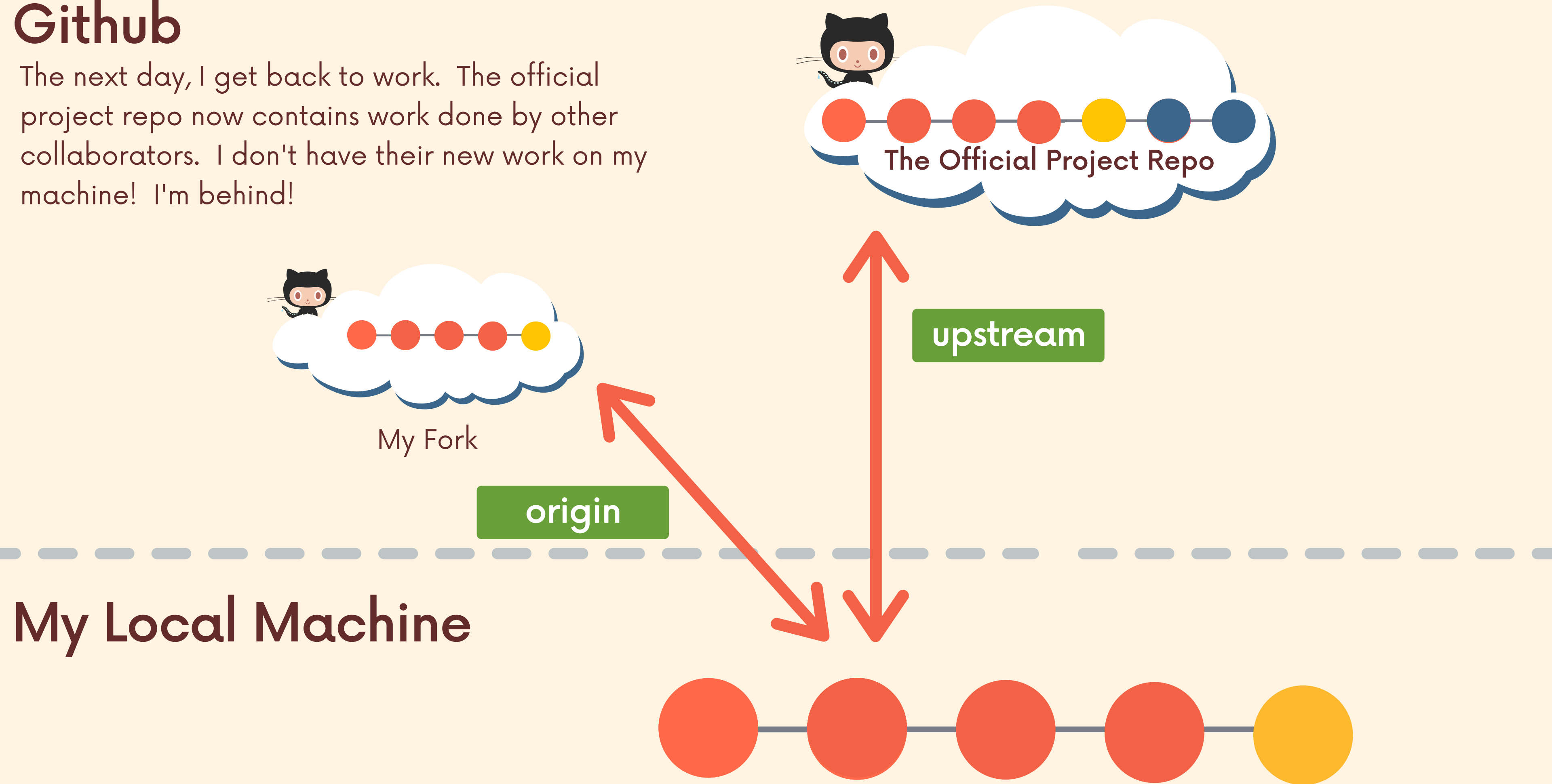
## My Local Machine



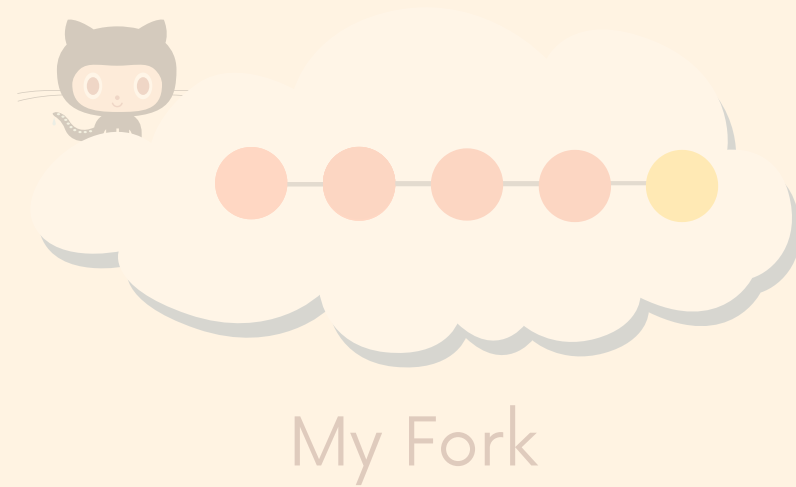
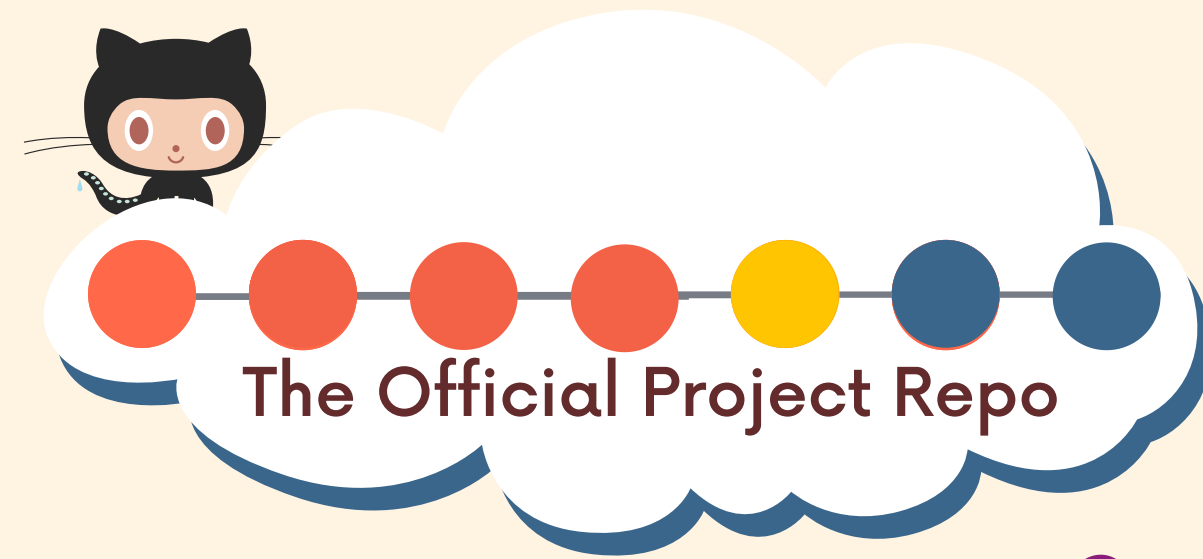


# Github

The next day, I get back to work. The official project repo now contains work done by other collaborators. I don't have their new work on my machine! I'm behind!



# Github



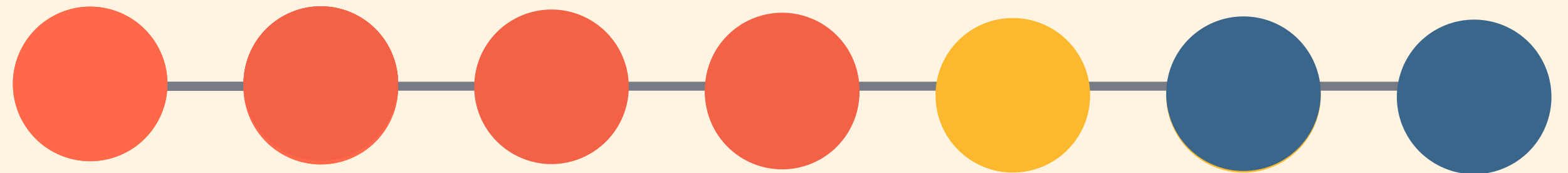
upstream

origin

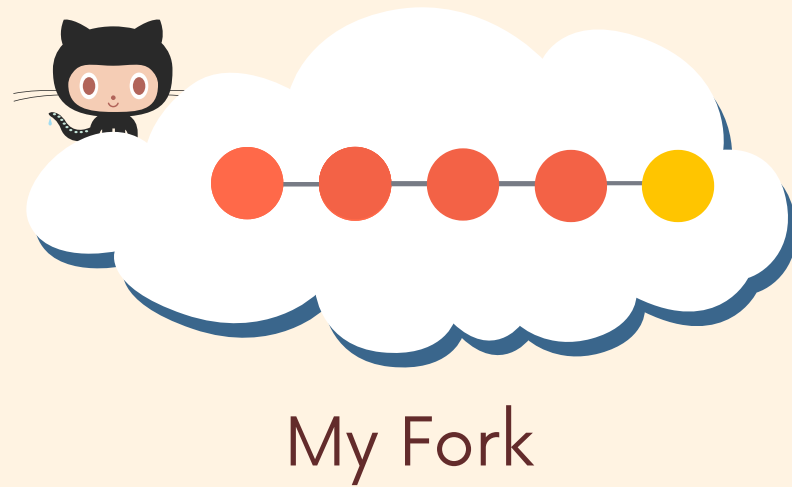
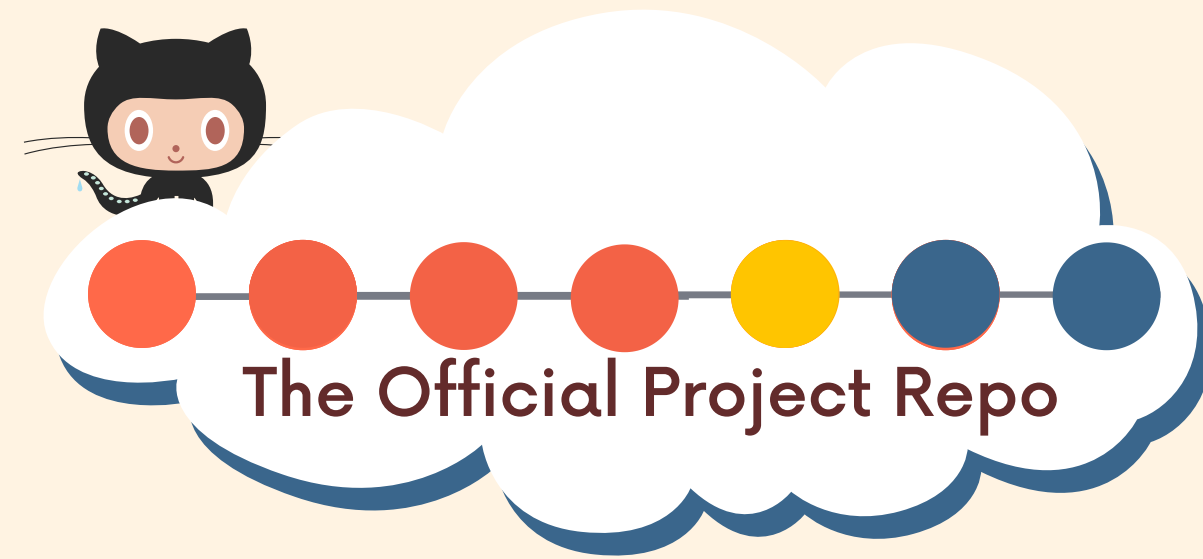


## My Local Machine

All I need to do is **pull from upstream** (the original repo) to get the latest changes in my local repo.

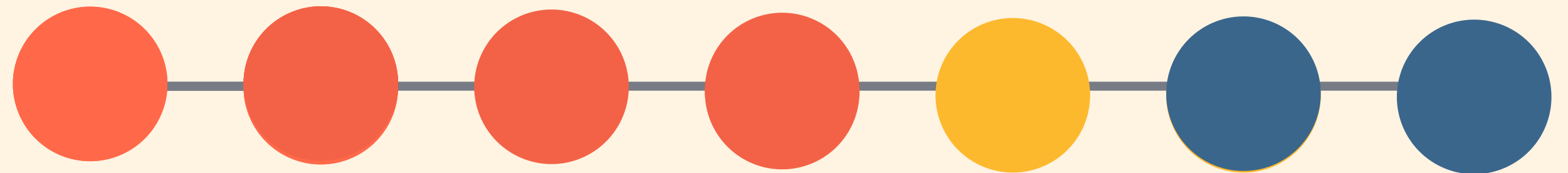


# Github

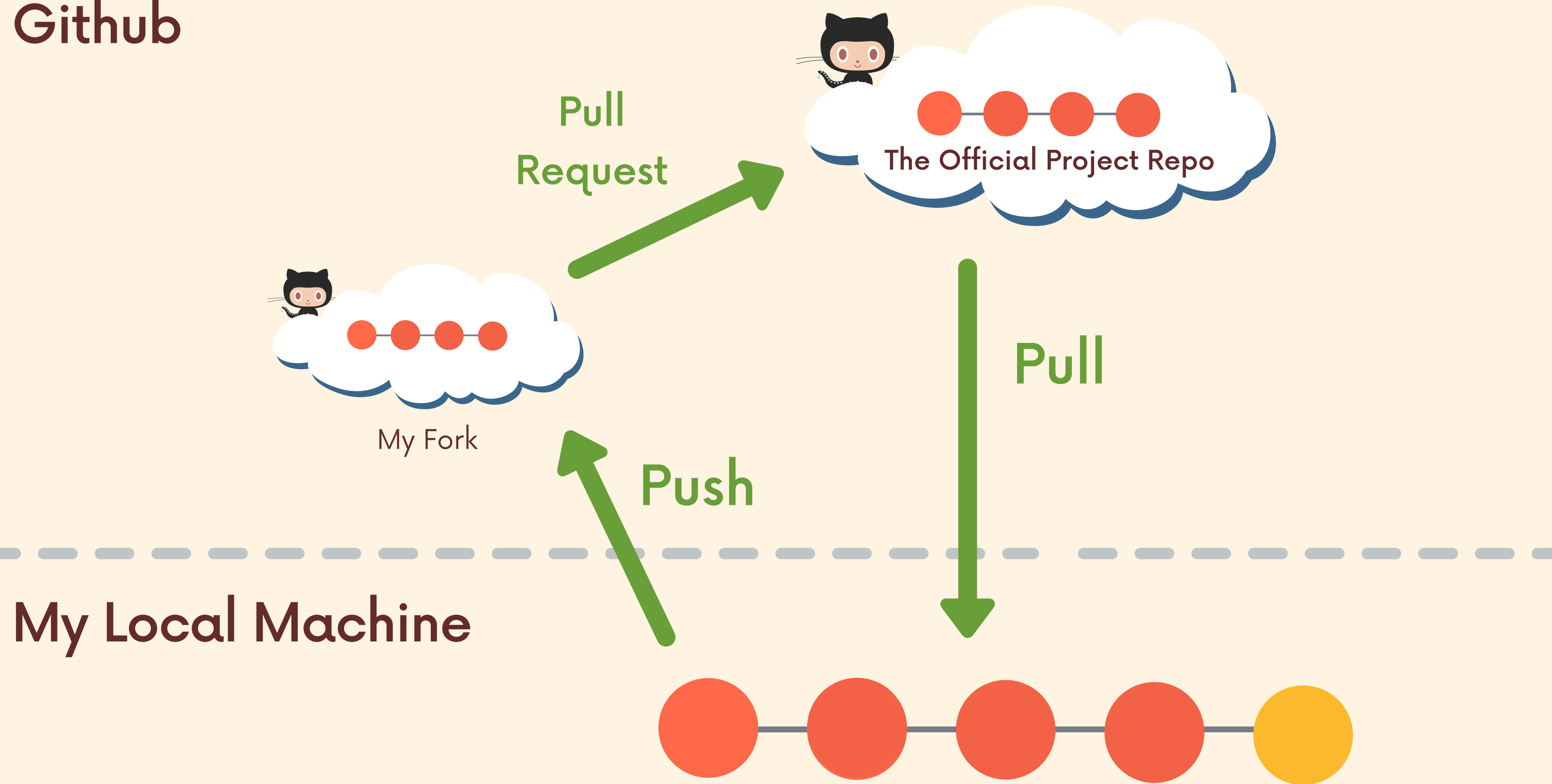


## My Local Machine

Now I have the latest changes from the upstream repo! I can work on some new features locally without working about being out of date.



# Github





# To Summarize!

- 1.1 fork the original project repo on Github
- 2.1 clone my fork to my local machine
- 3.1 add a remote pointing to the original project repo.  
This remote is often named upstream.
- 4.1 make changes and add/commit on a feature branch on my local machine
- 5.1 push up my new feature branch to my forked repo (usually called origin)
- 6.1 open a pull request to the original project repo containing the new work on my forked repo
7. Hopefully the pull request is accepted and my changes are merged in!





# An Even Briefer Summary

1. FORK THE PROJECT
2. CLONE THE FORK
3. ADD UPSTREAM REMOTE
4. DO SOME WORK
5. PUSH TO ORIGIN
6. OPEN PR

