

git workflows (for collaboration)



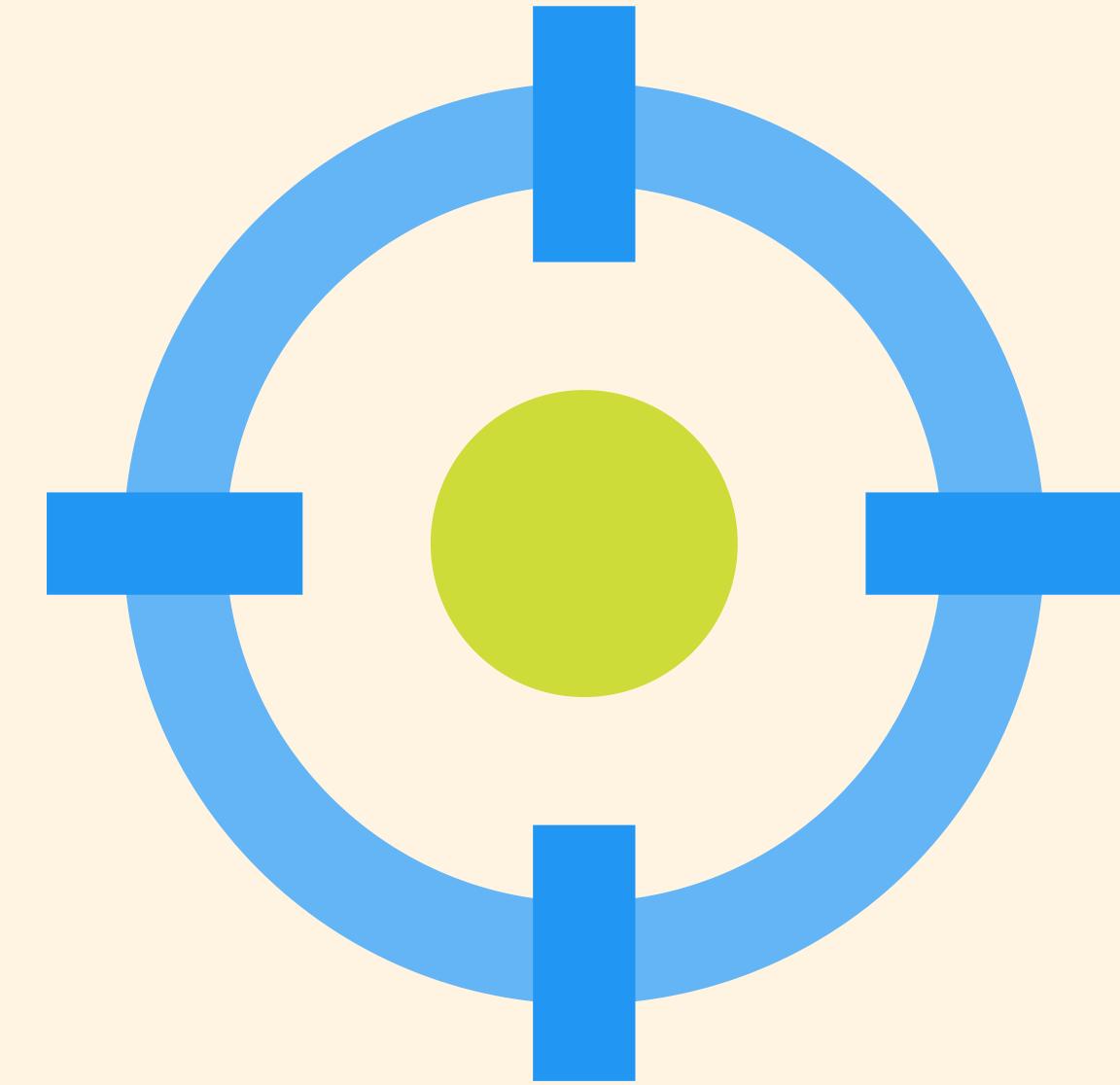


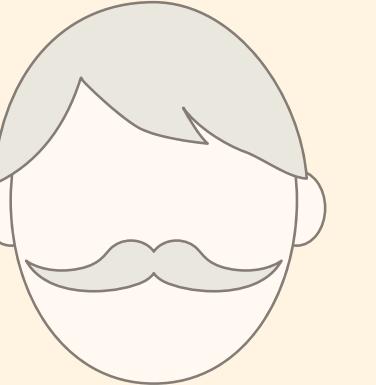
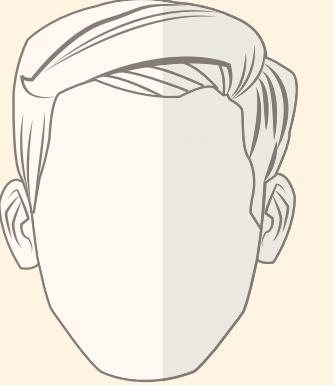
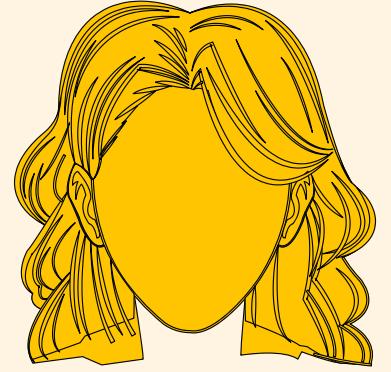
Centralized Workflow

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!

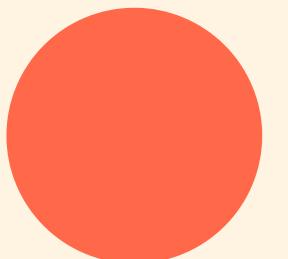


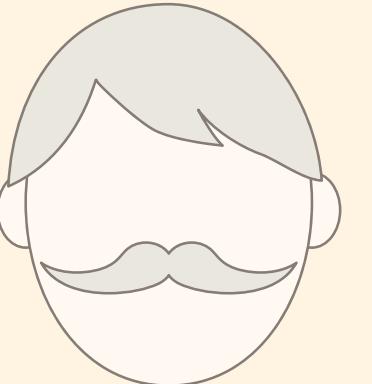
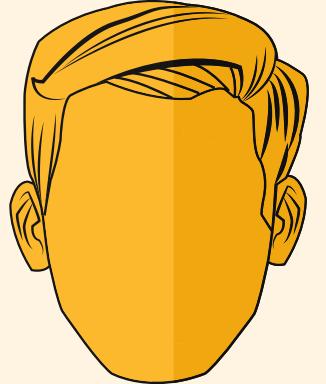


Pamela clones the repo



master

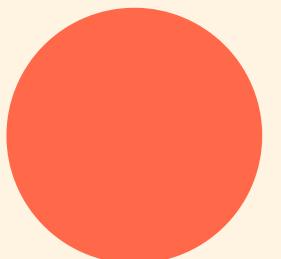


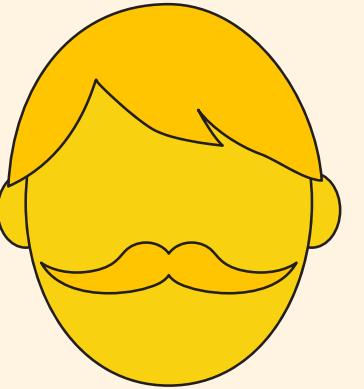
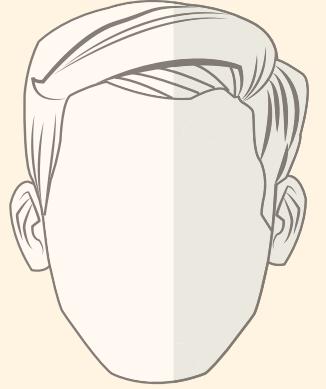
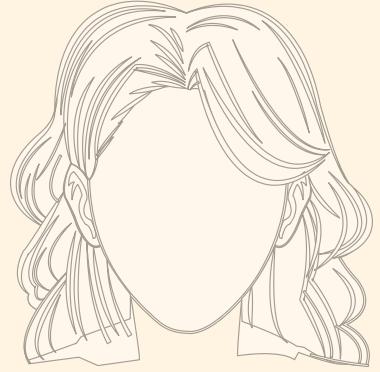


David clones the repo



master

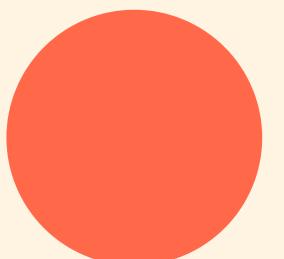


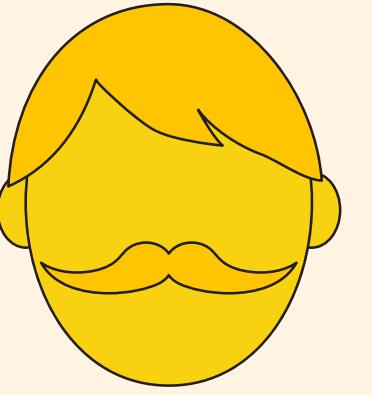
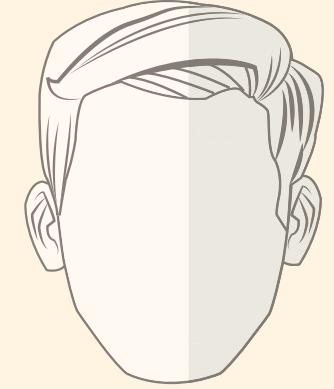


Forrest clones the repo

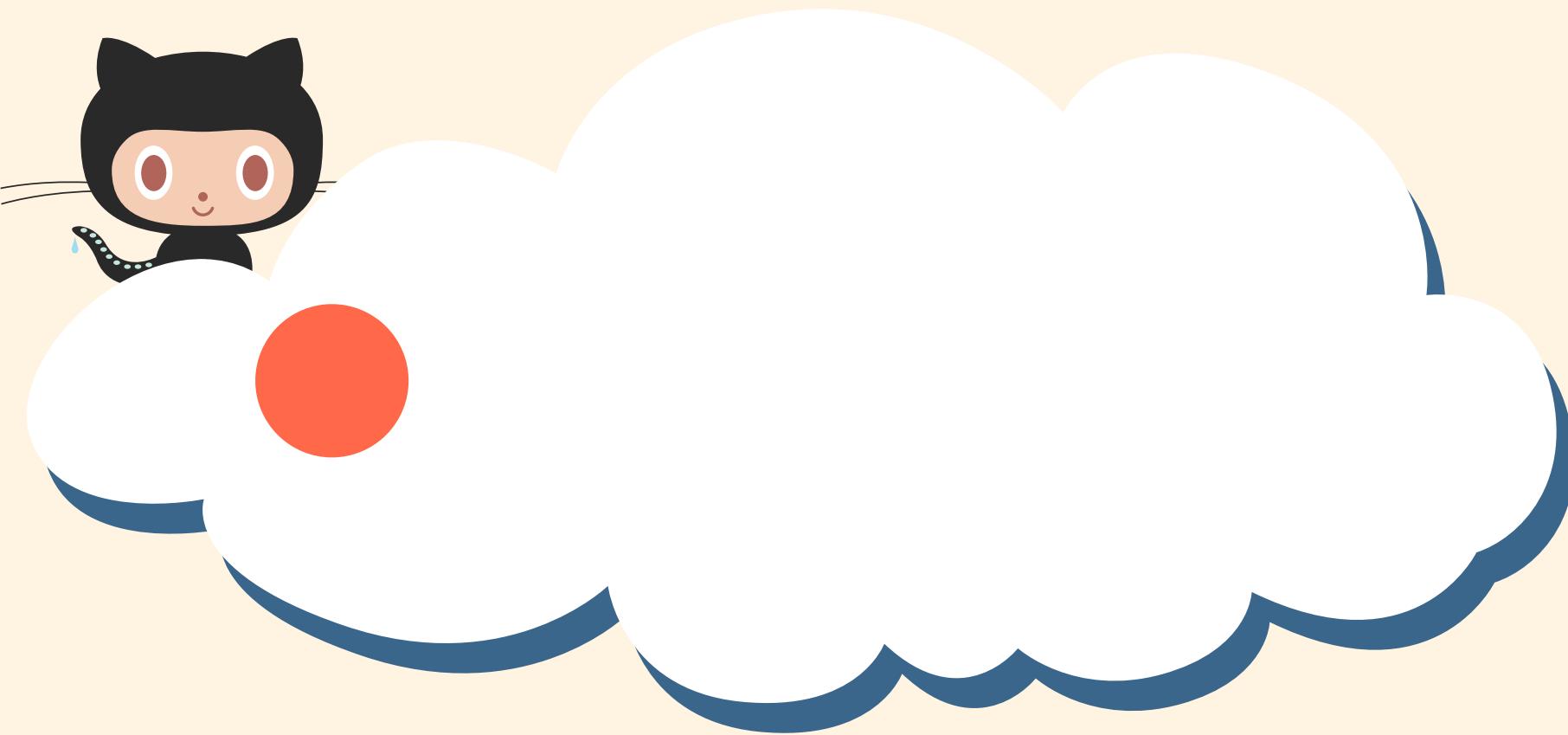


master

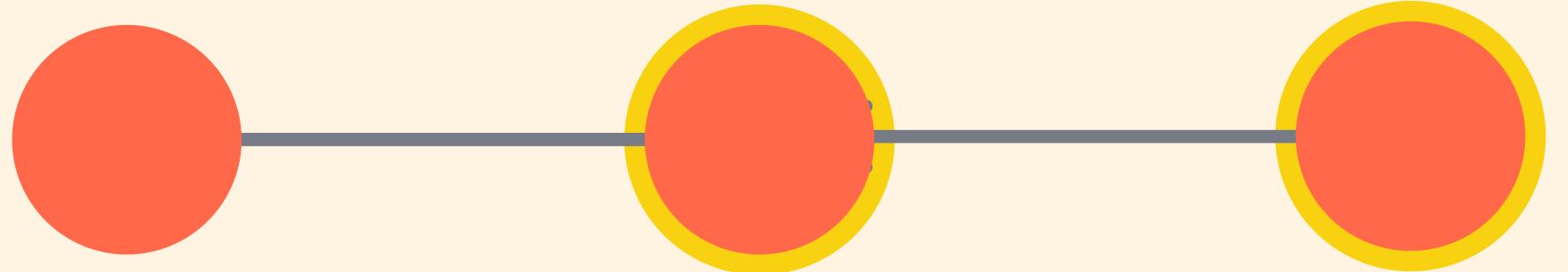


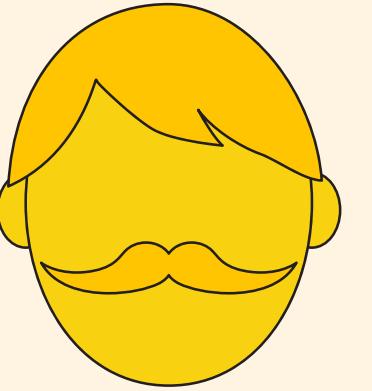
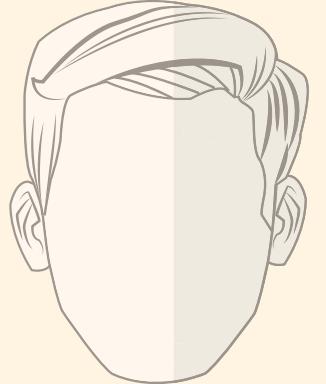
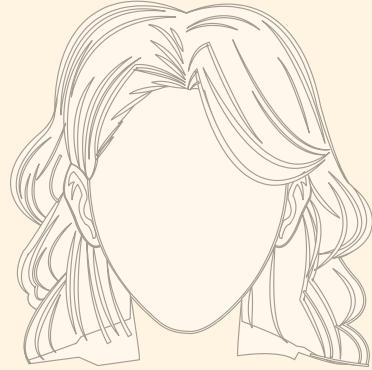


Forrest gets to work on a
new feature! Adding and
Committing all day long!

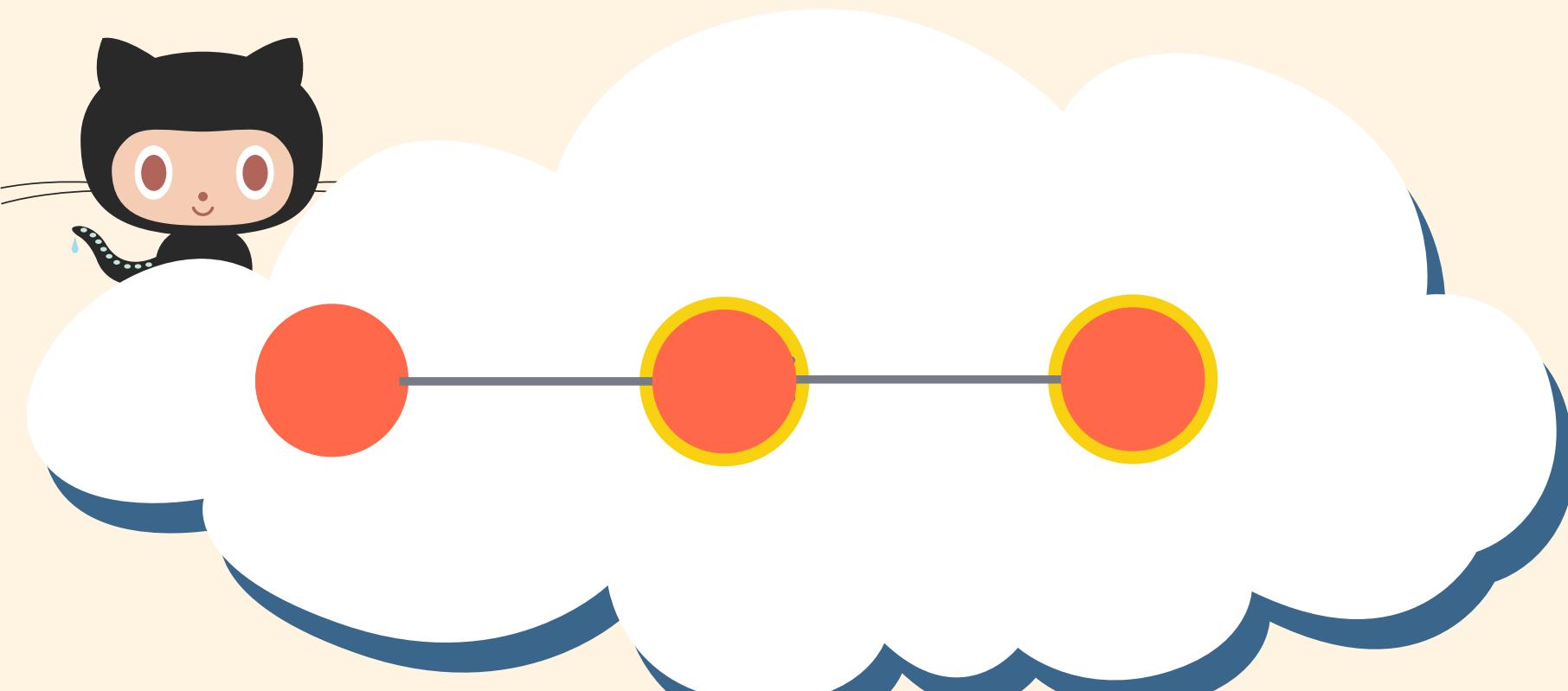
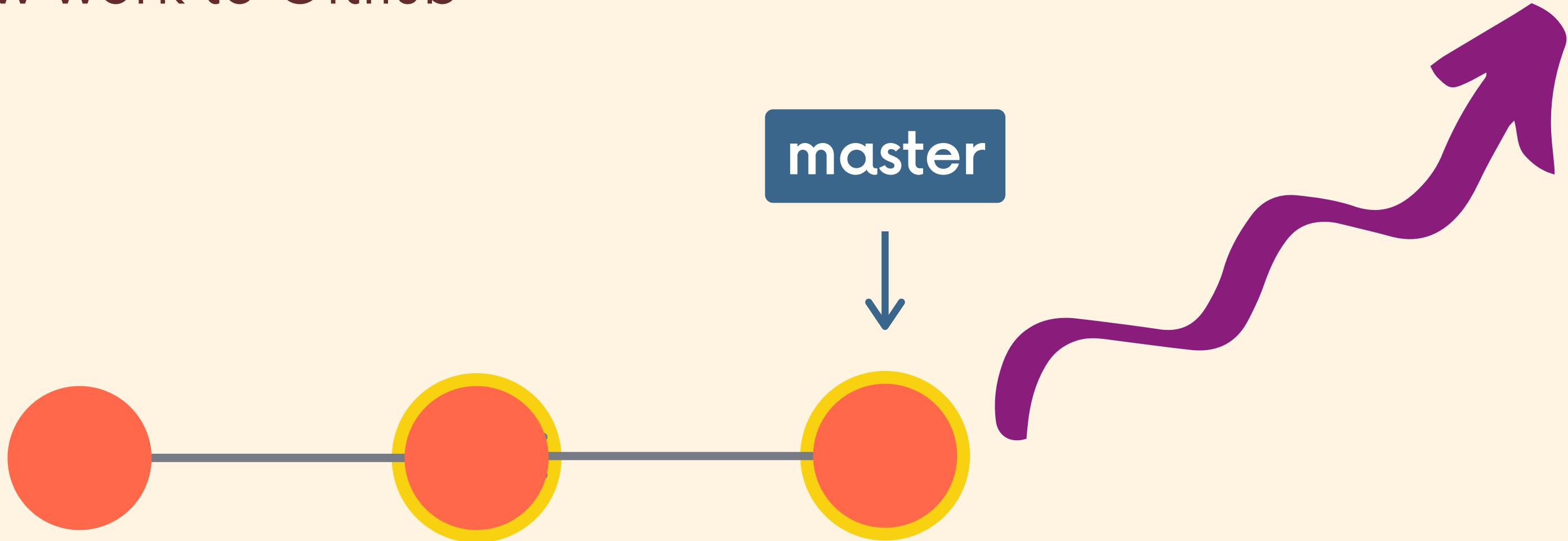


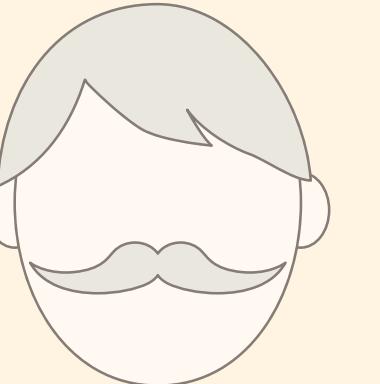
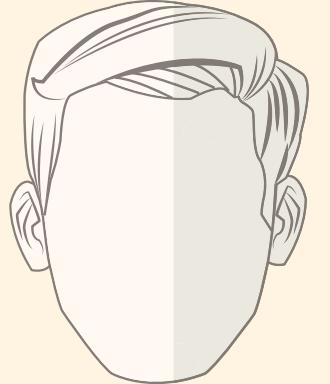
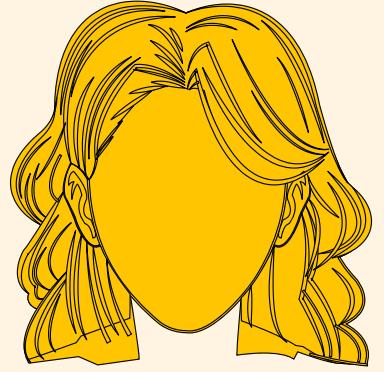
master



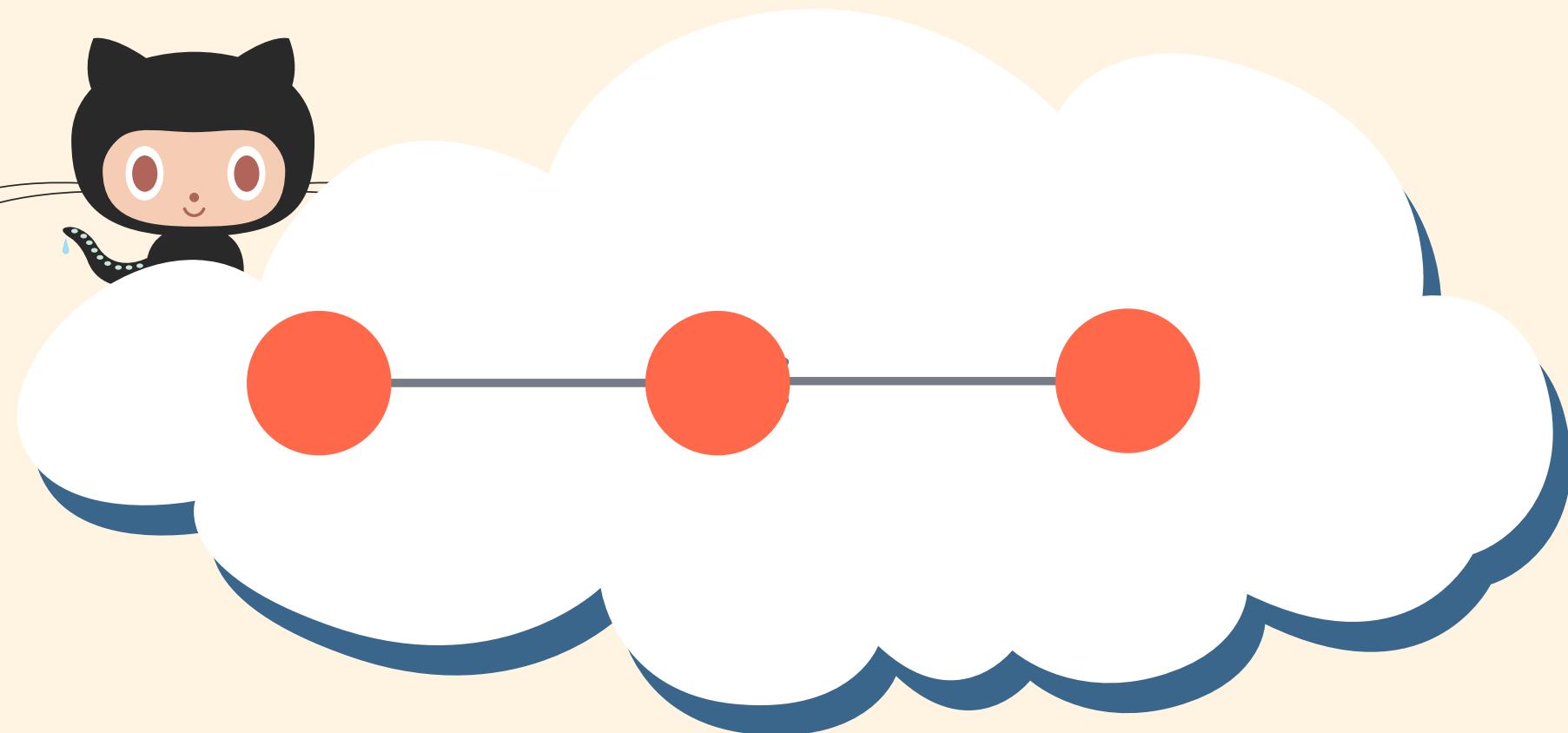
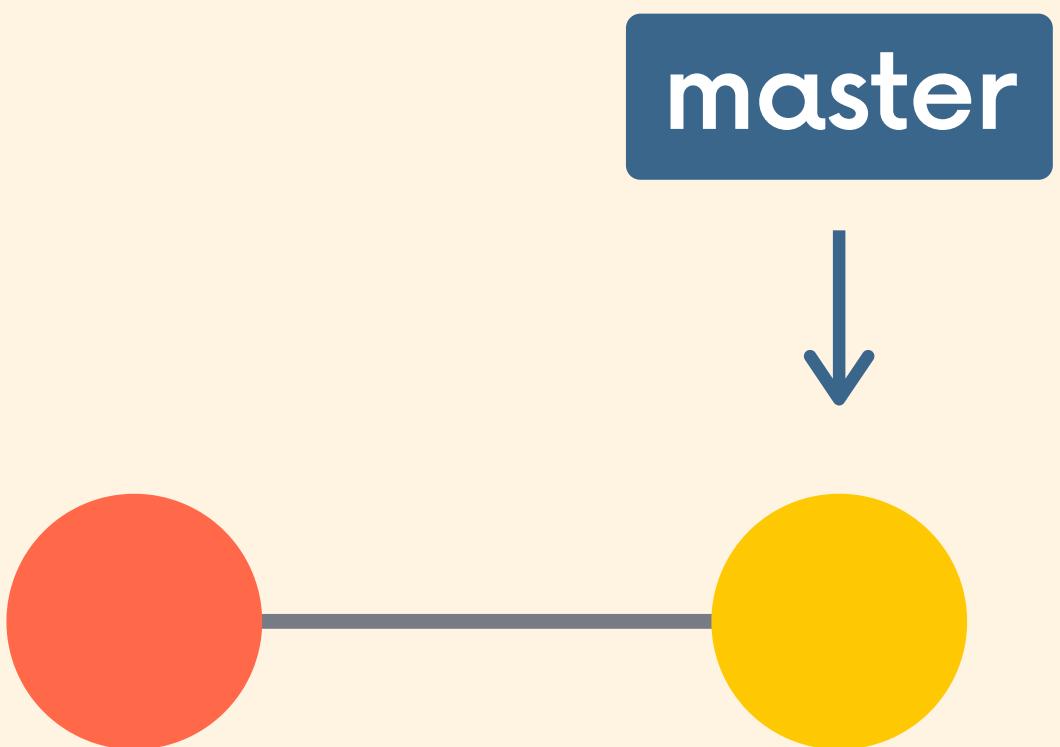


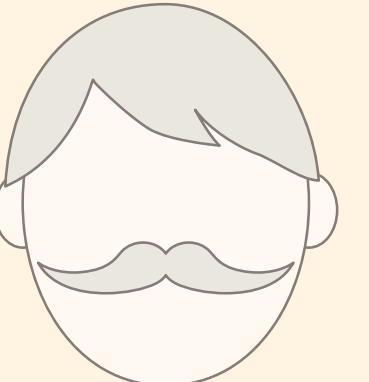
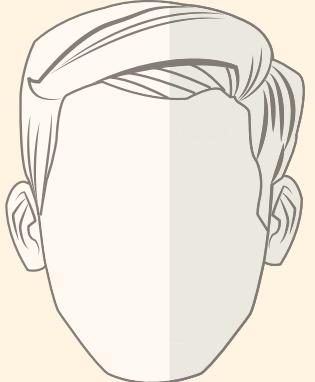
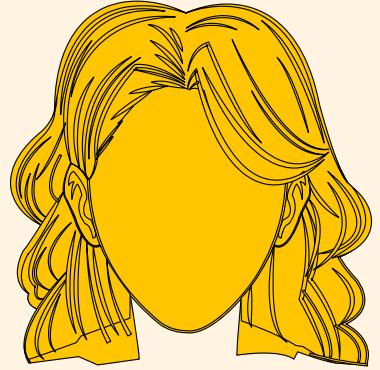
Forrest now pushes up his
new work to Github



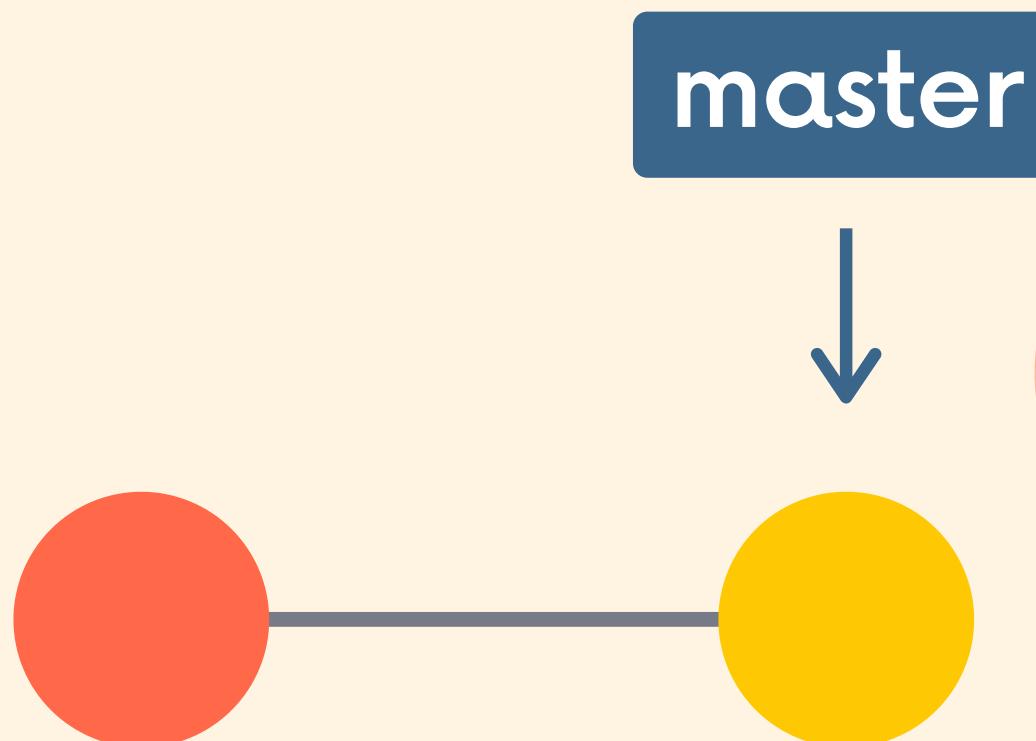


Pamela has also been
hard at work on her own
new feature.

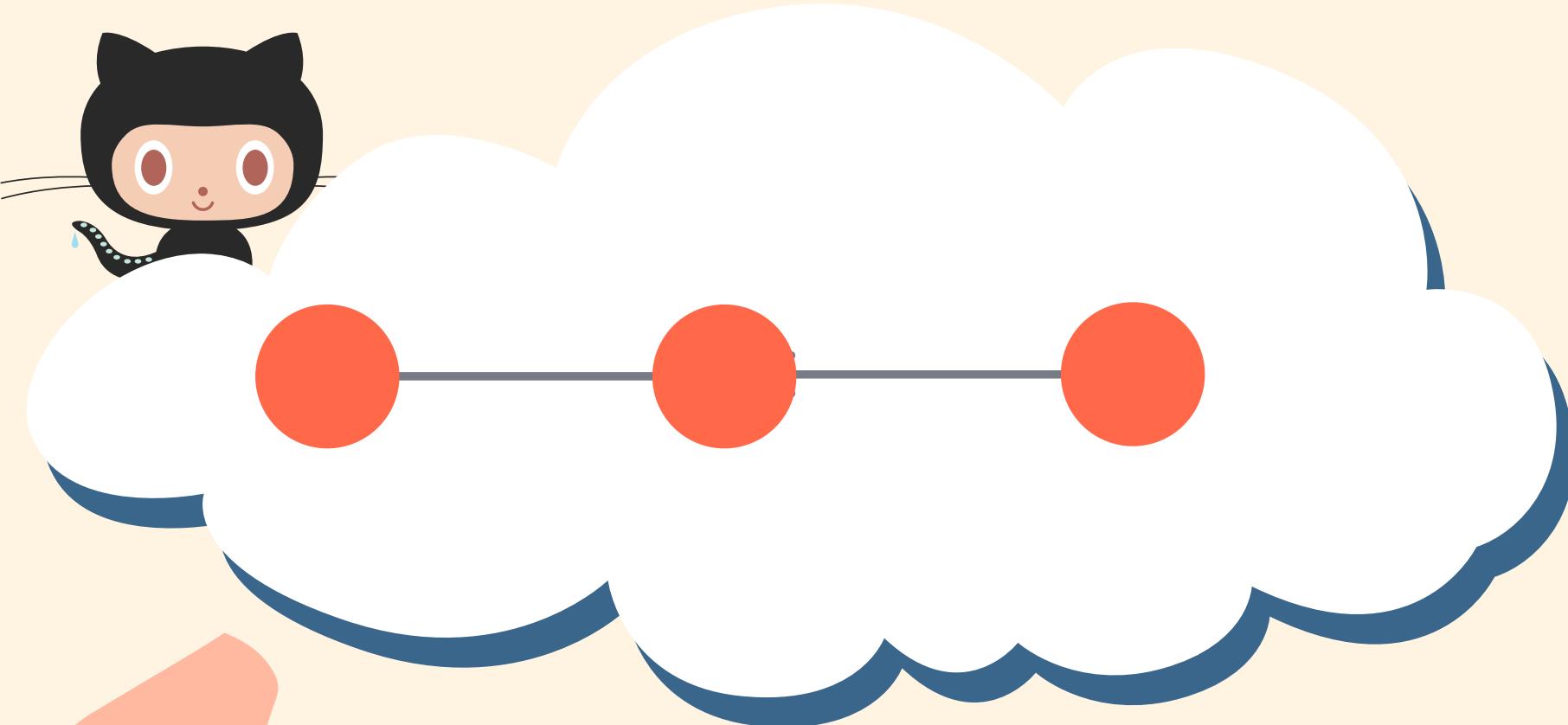


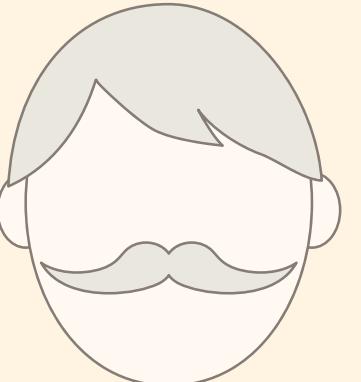
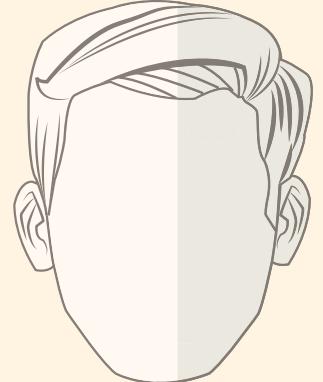
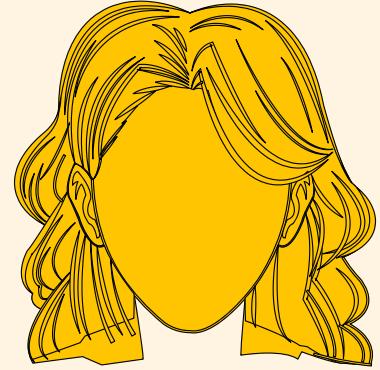


She tries to **push** her new work up to Github, but she runs into trouble!



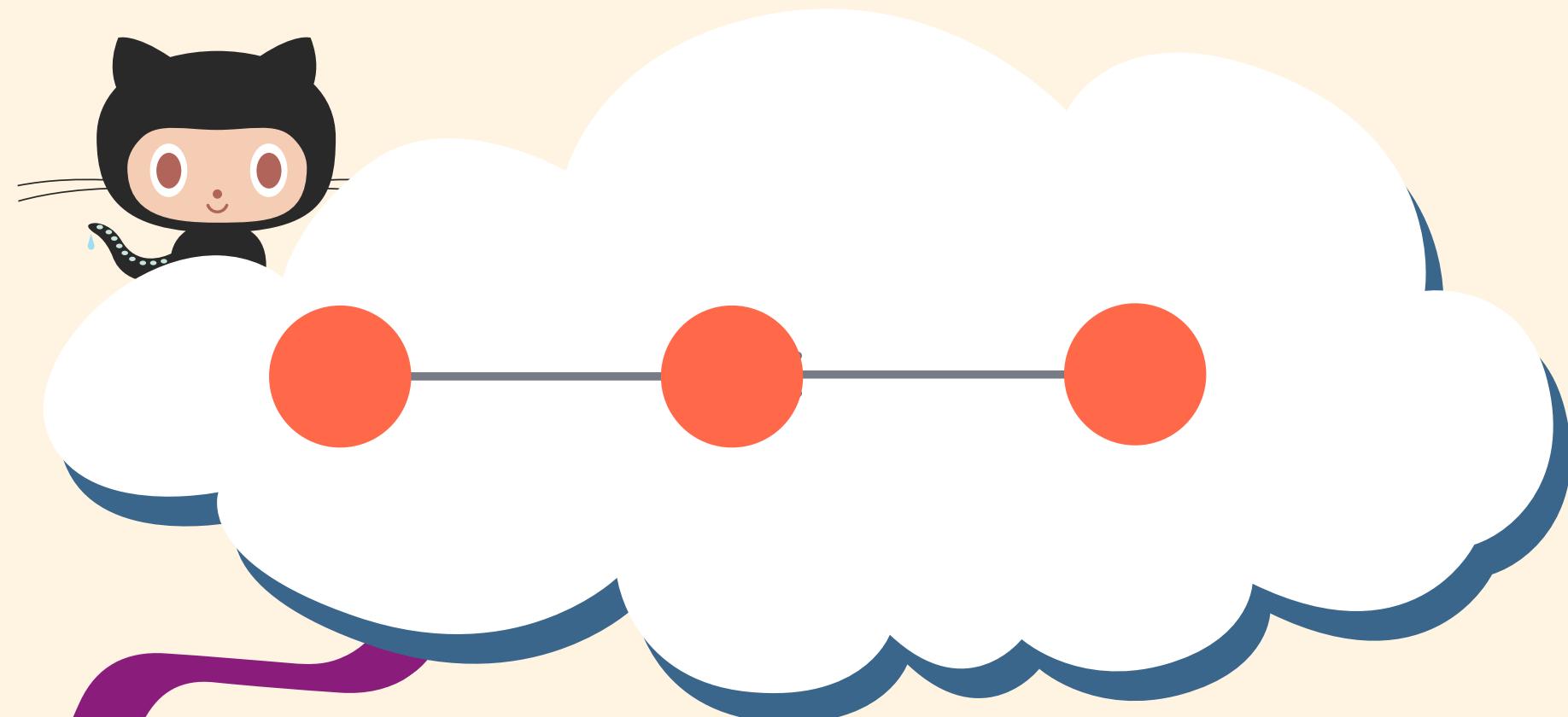
Failed to push. Updates were rejected because the tip of your current branch is behind its remote counterpart. Merge the remote changes (e.g. 'git pull') before pushing again.

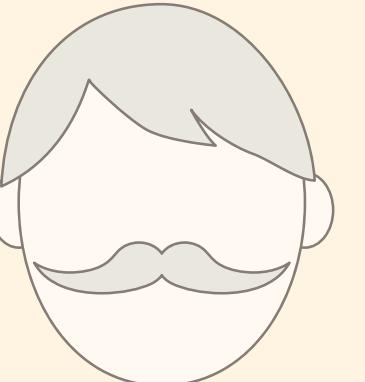
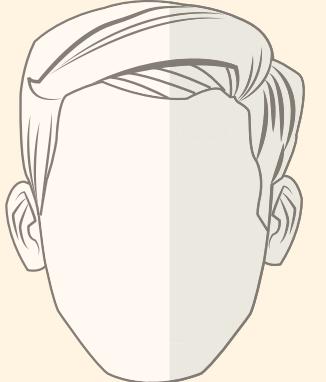
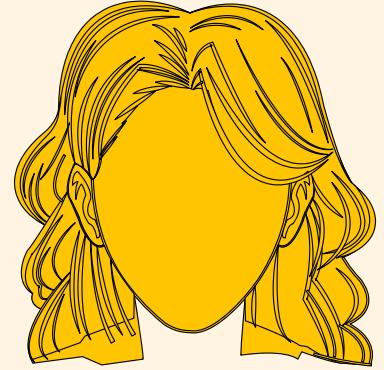




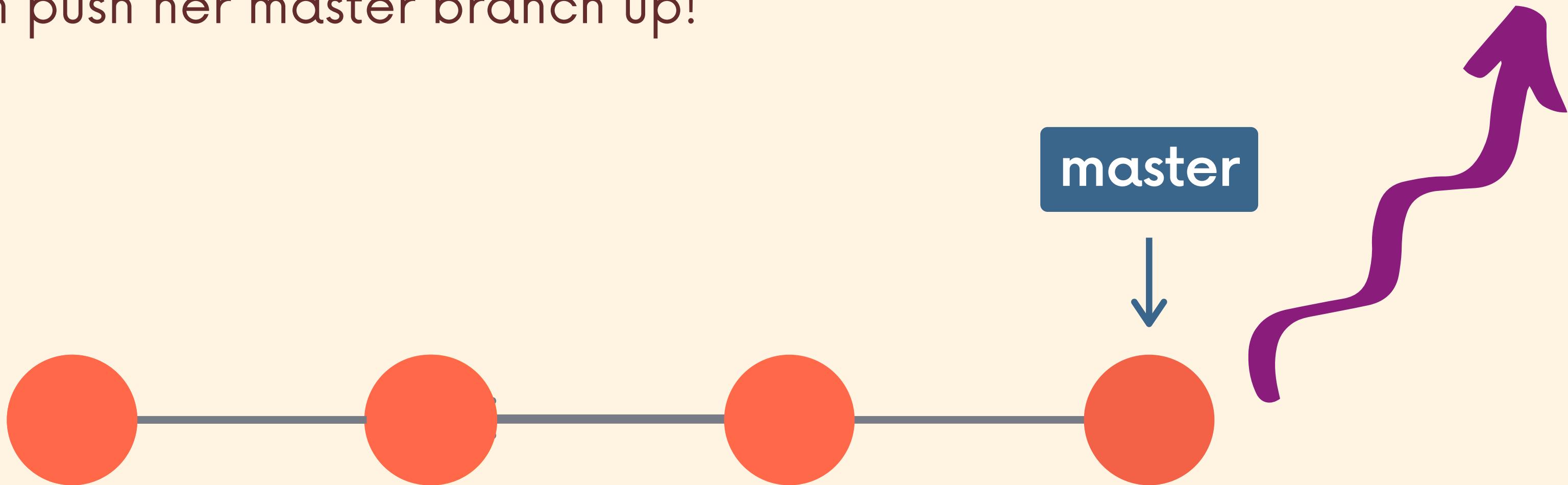
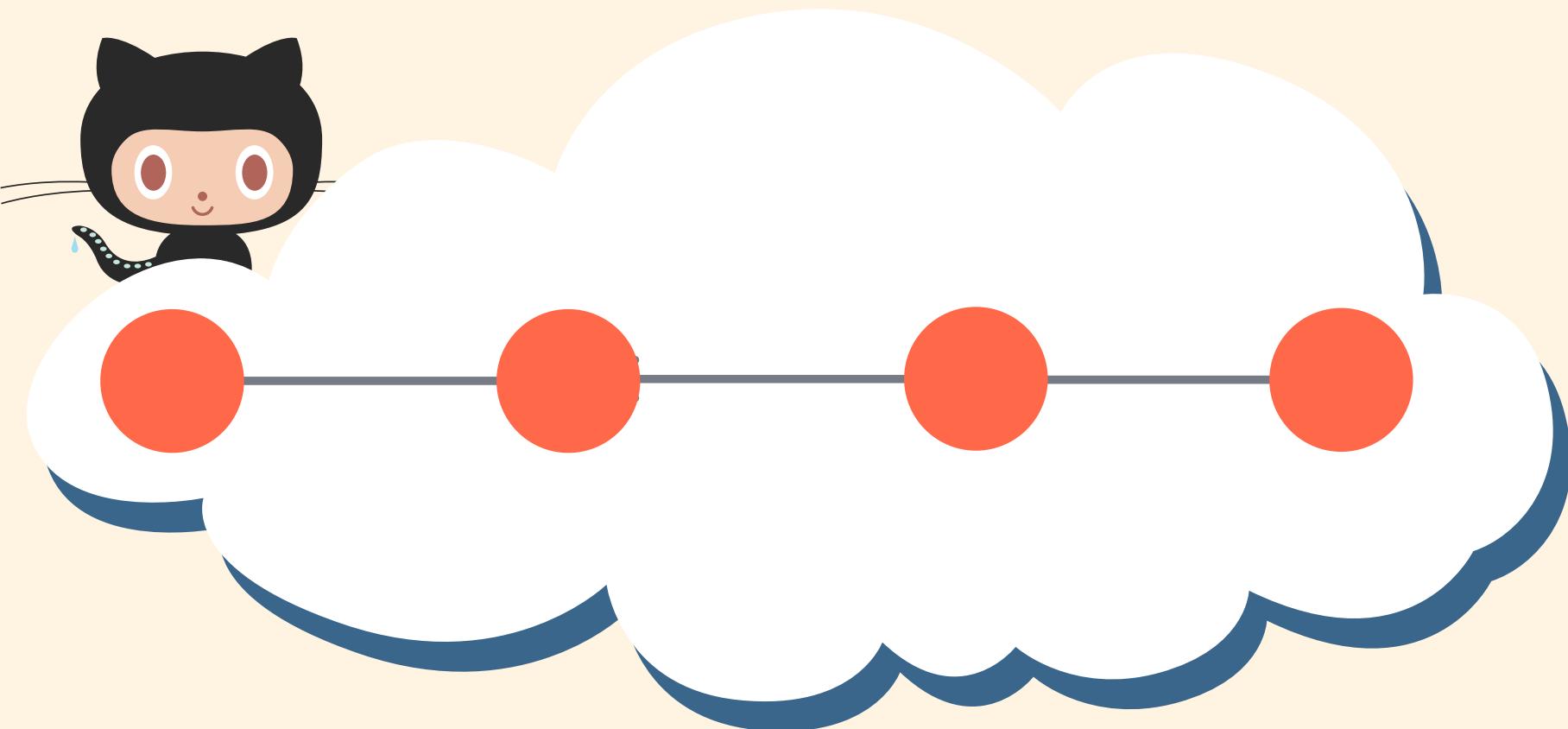
So she pulls to get the changes from origin master

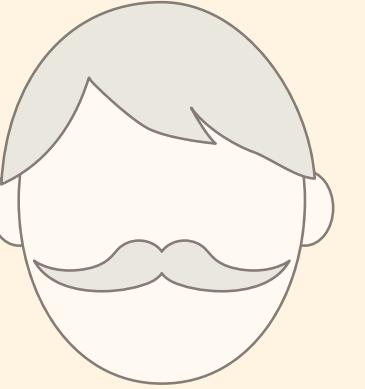
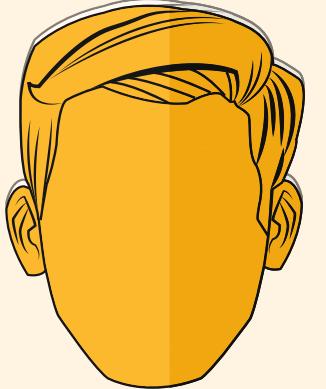
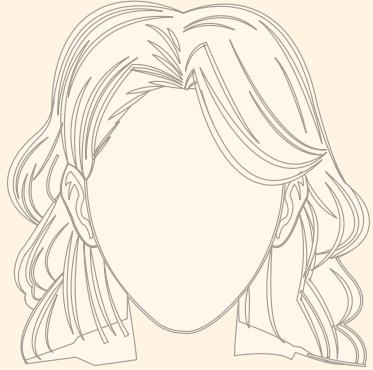
Forrest's work must be merged in. Hopefully this goes relatively smoothly!





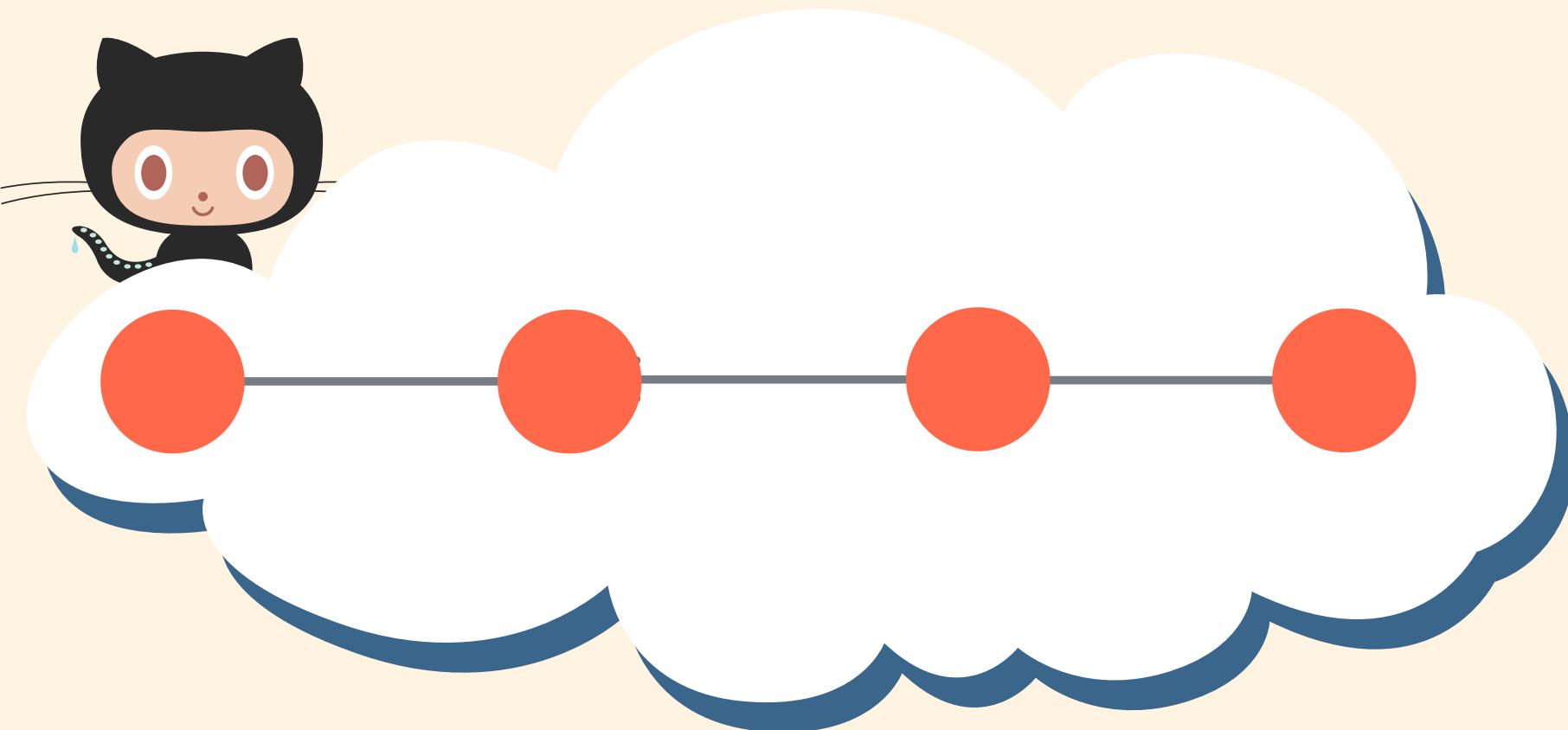
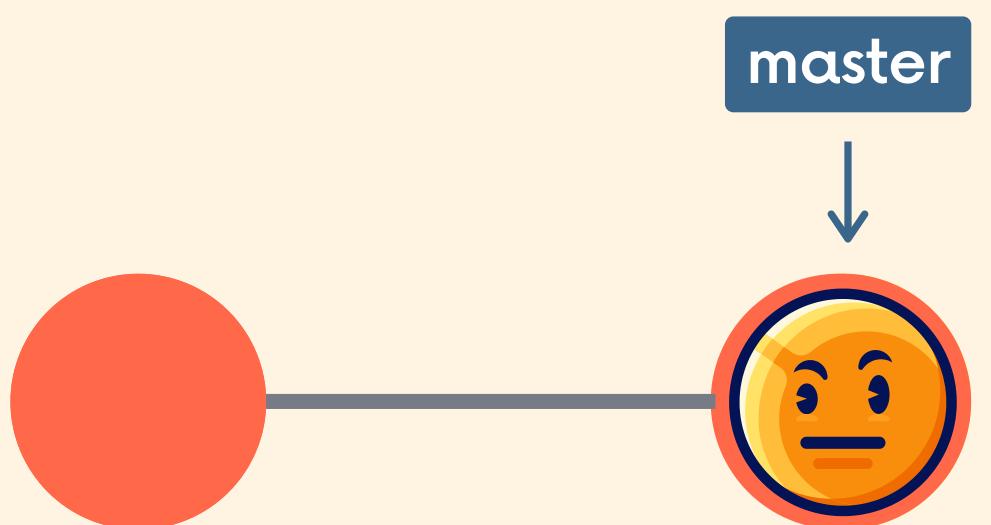
Now that she has merged the latest work from Github, she can push her master branch up!

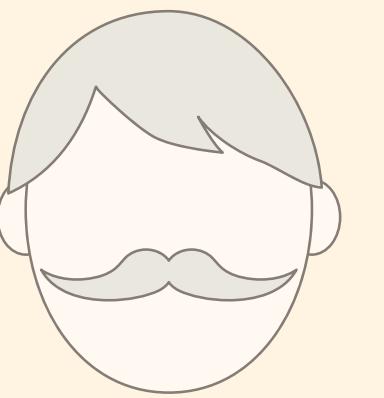
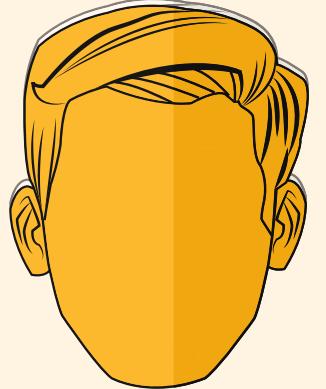
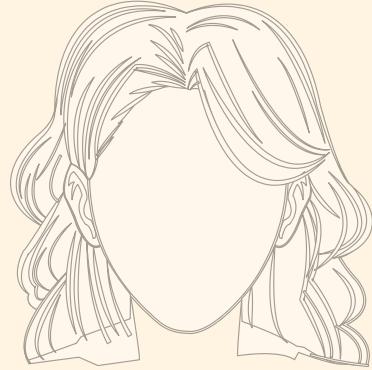




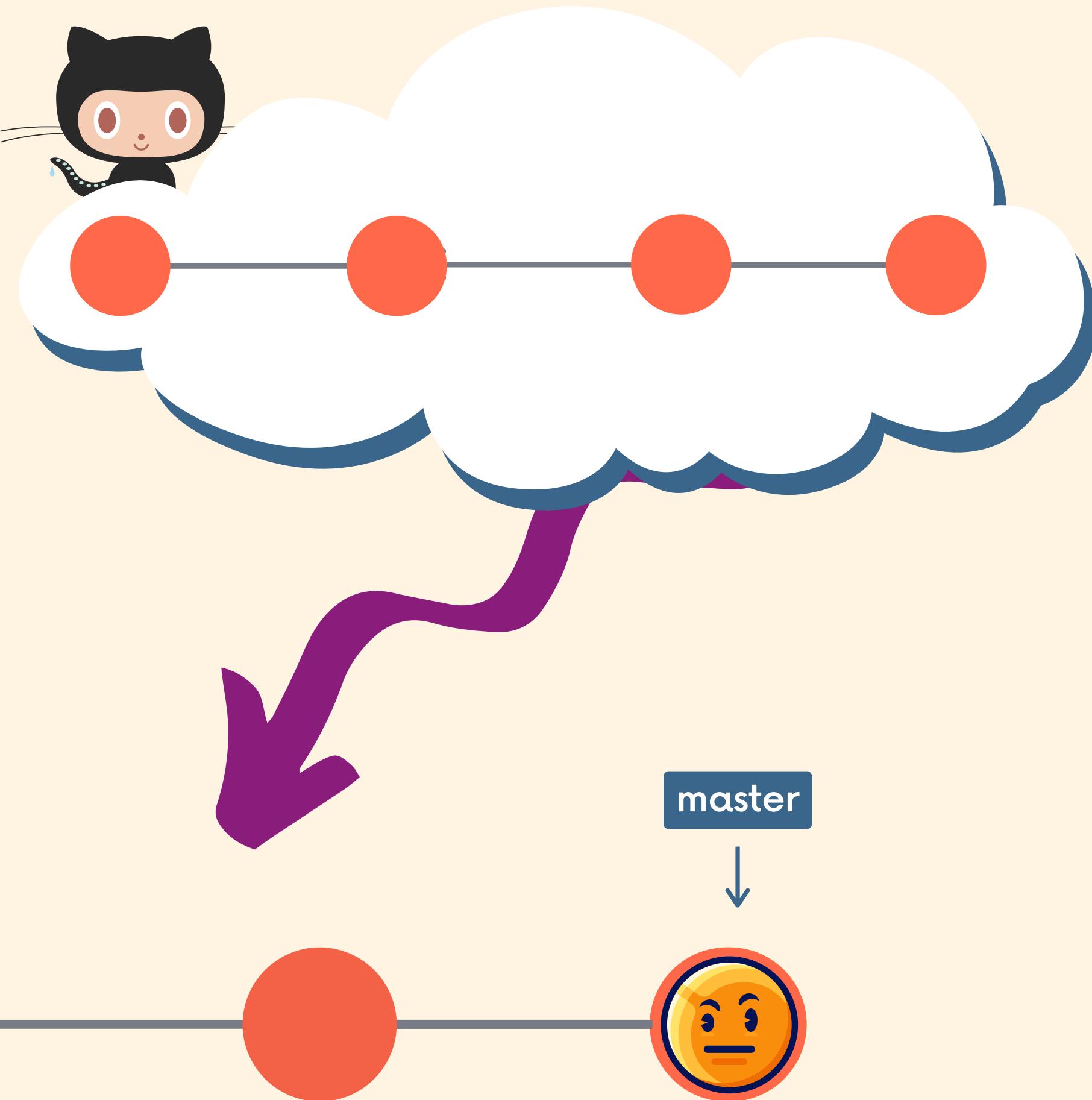
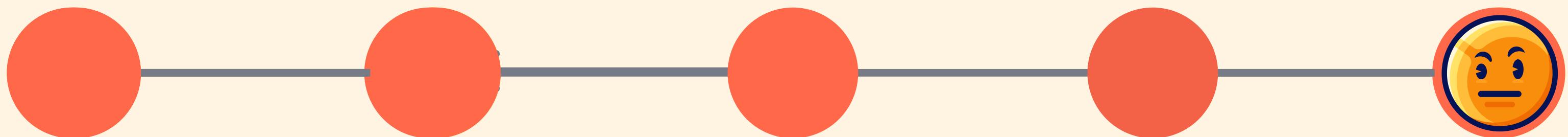
David working on a new feature, but is having some doubts.

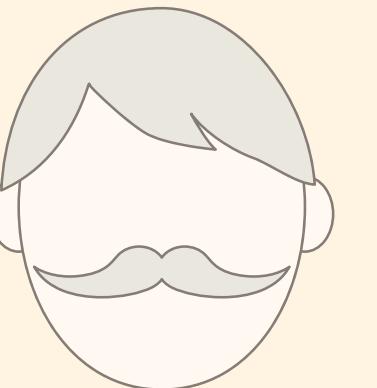
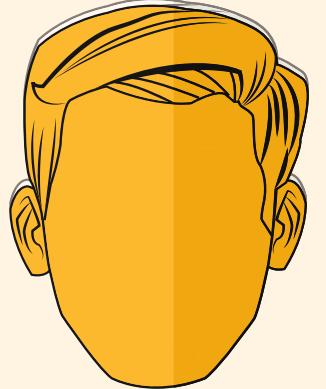
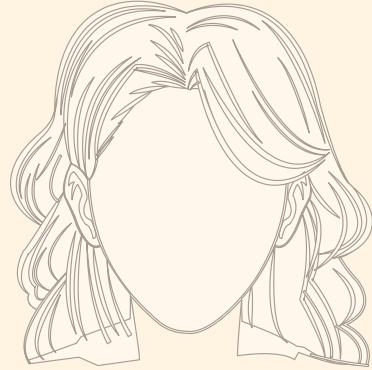
He'd like to share his commits with the rest of the team to start a discussion.



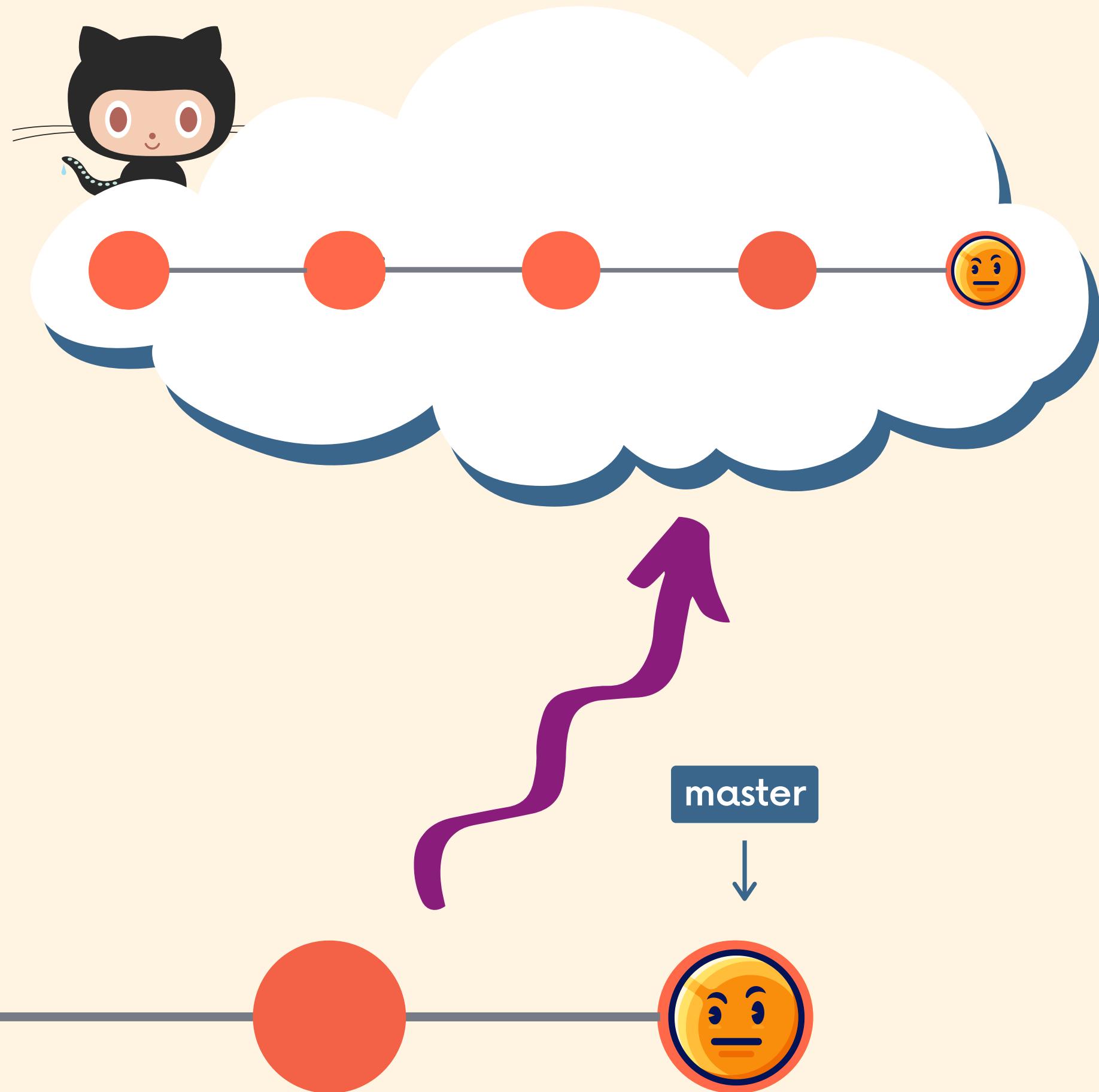


Before he can even share these iffy commits, he has to **pull** from Github and **merge** them in to master





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!



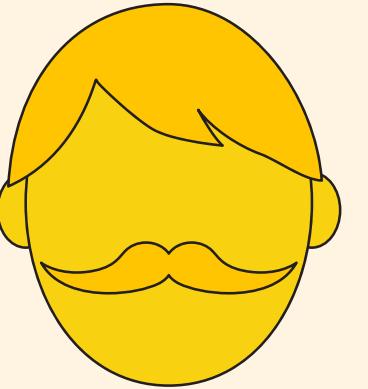
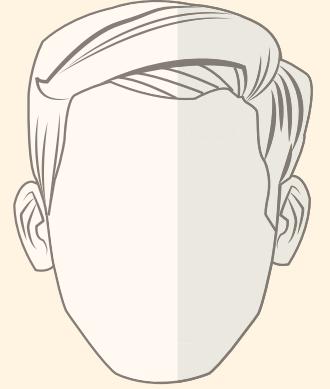
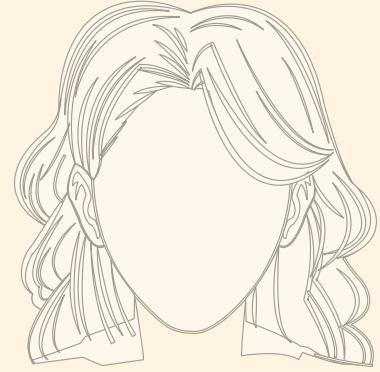


Feature Branches

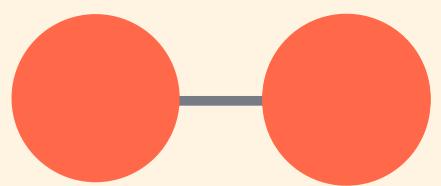
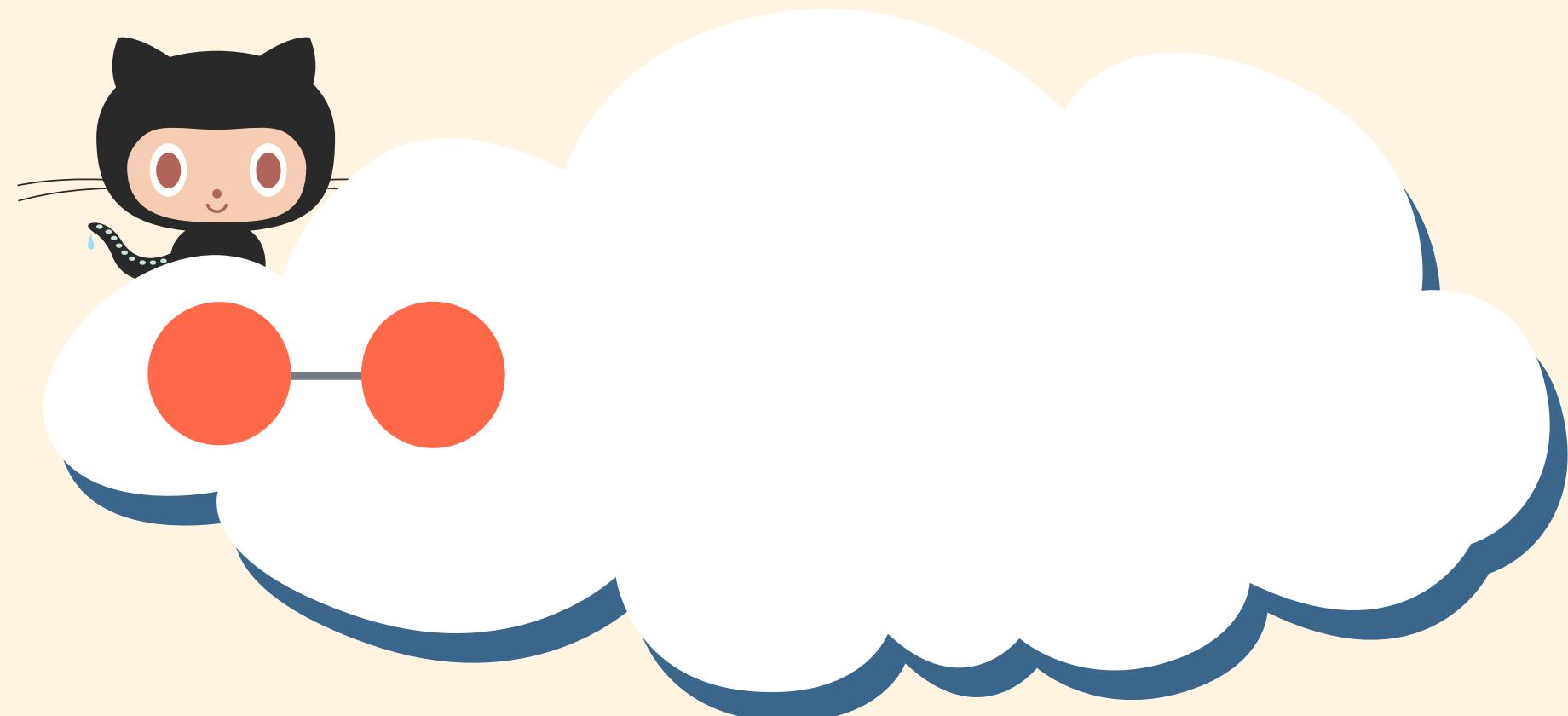
Rather than working directly on master/main, all new development should be done on separate branches!

- 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)

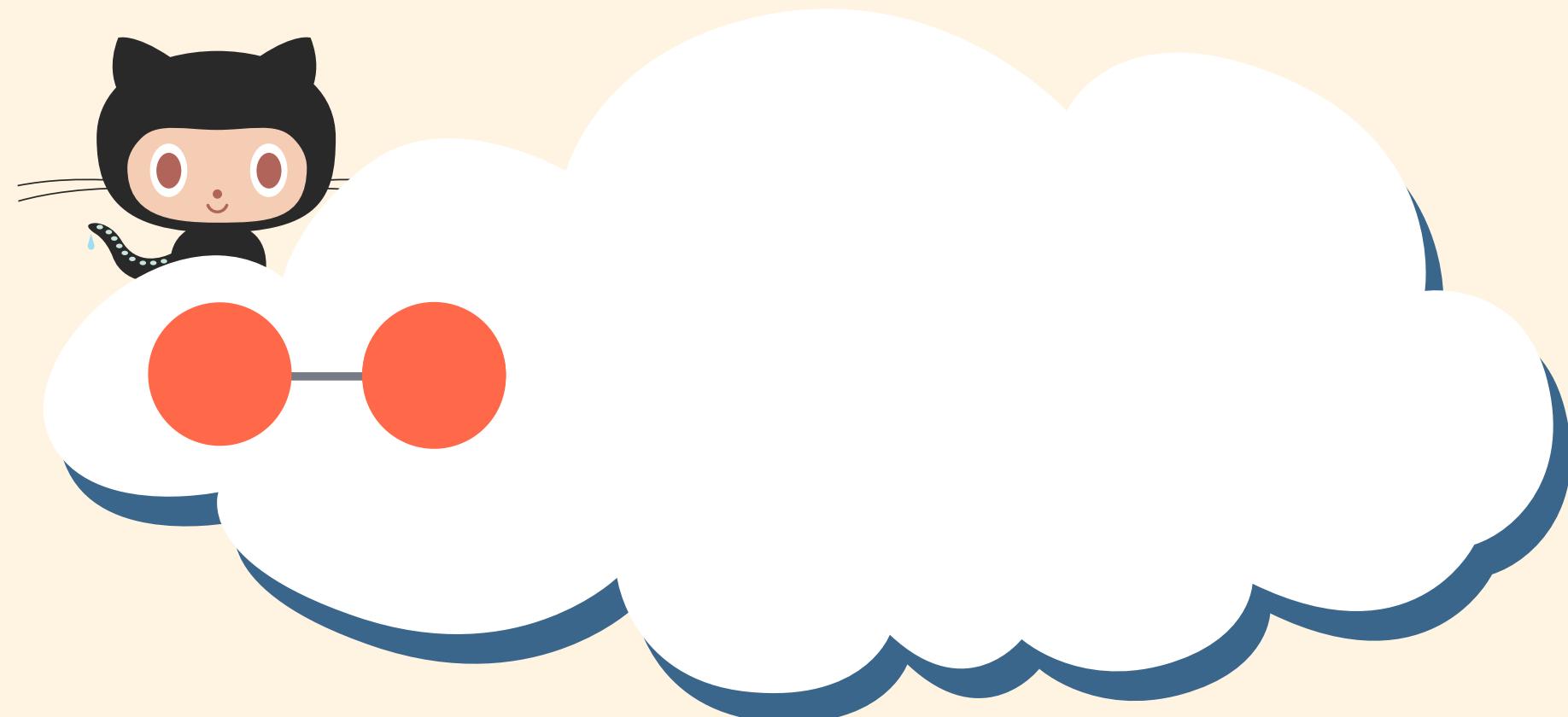
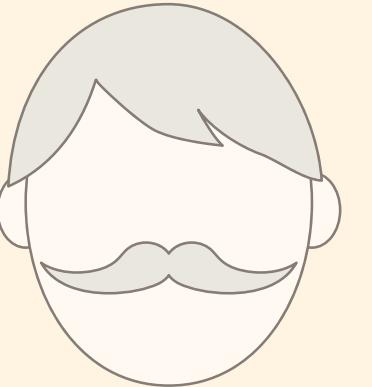
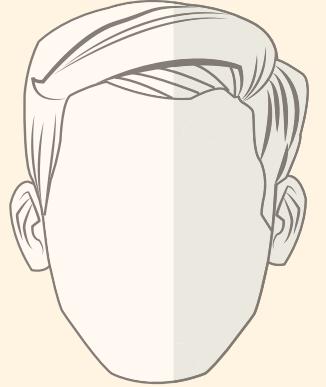
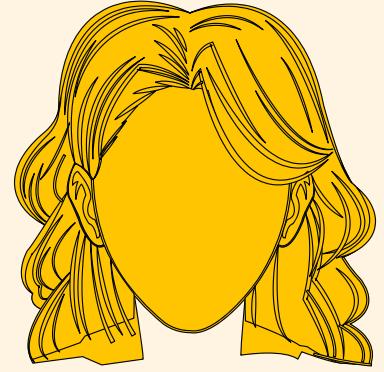




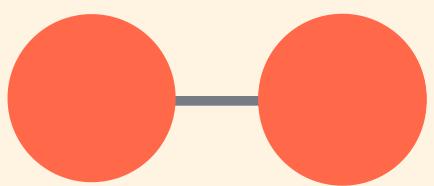
Forrest clones the repo



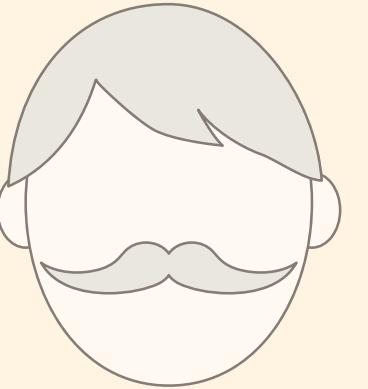
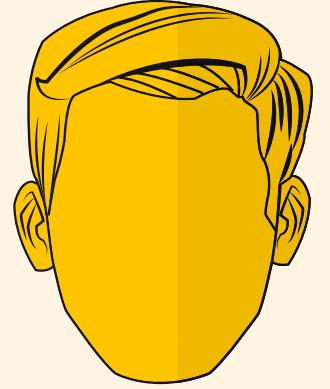
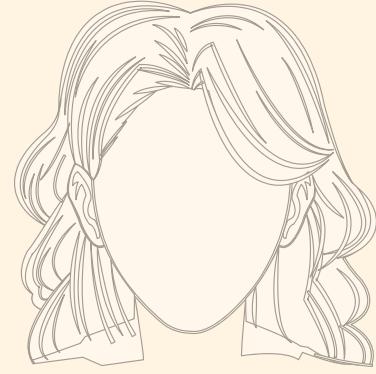
master



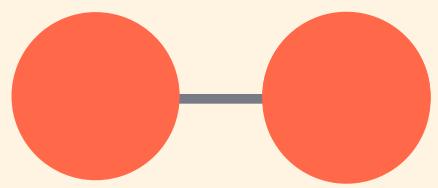
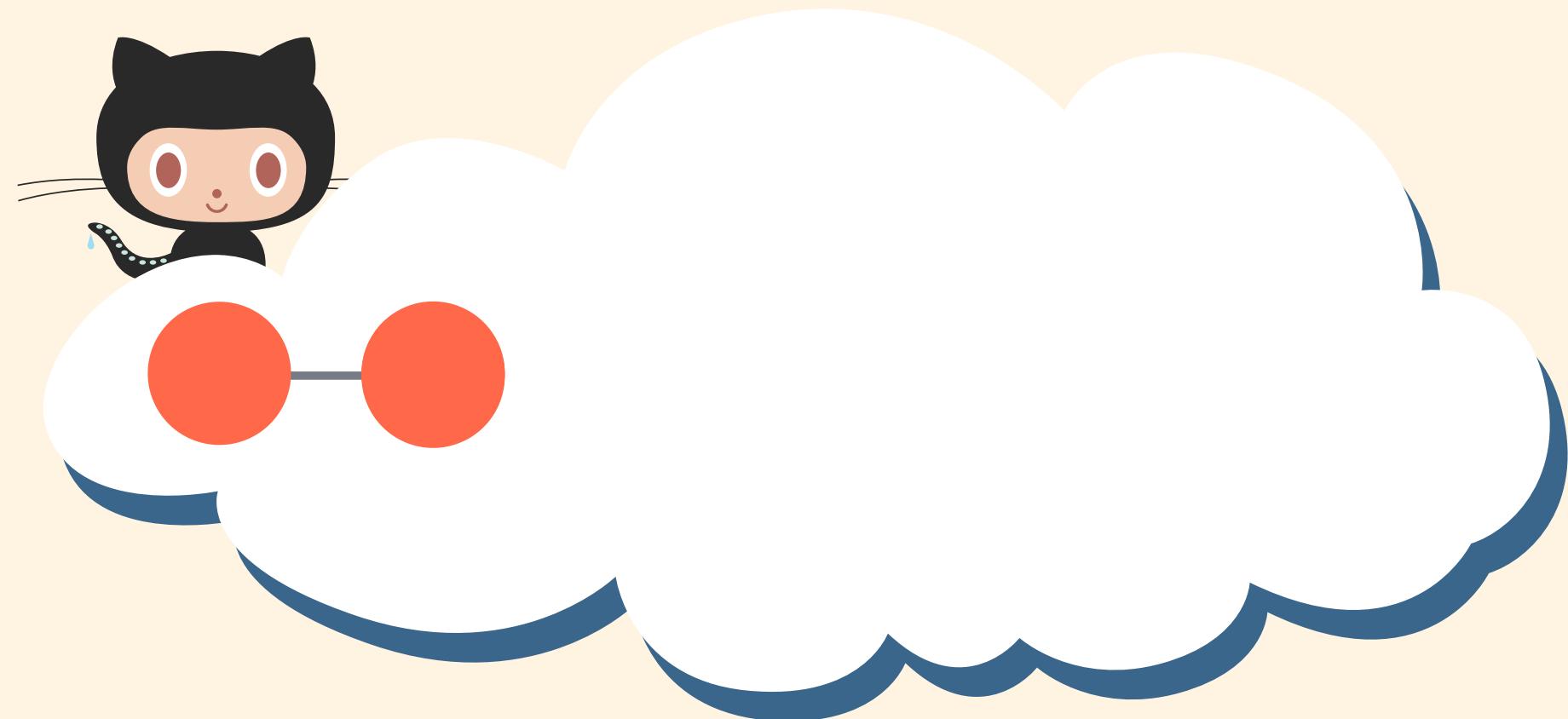
Pamela clones the repo



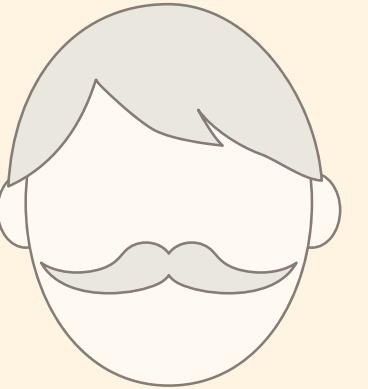
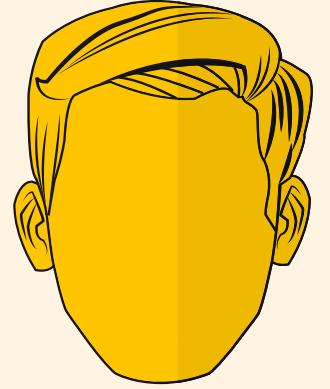
master



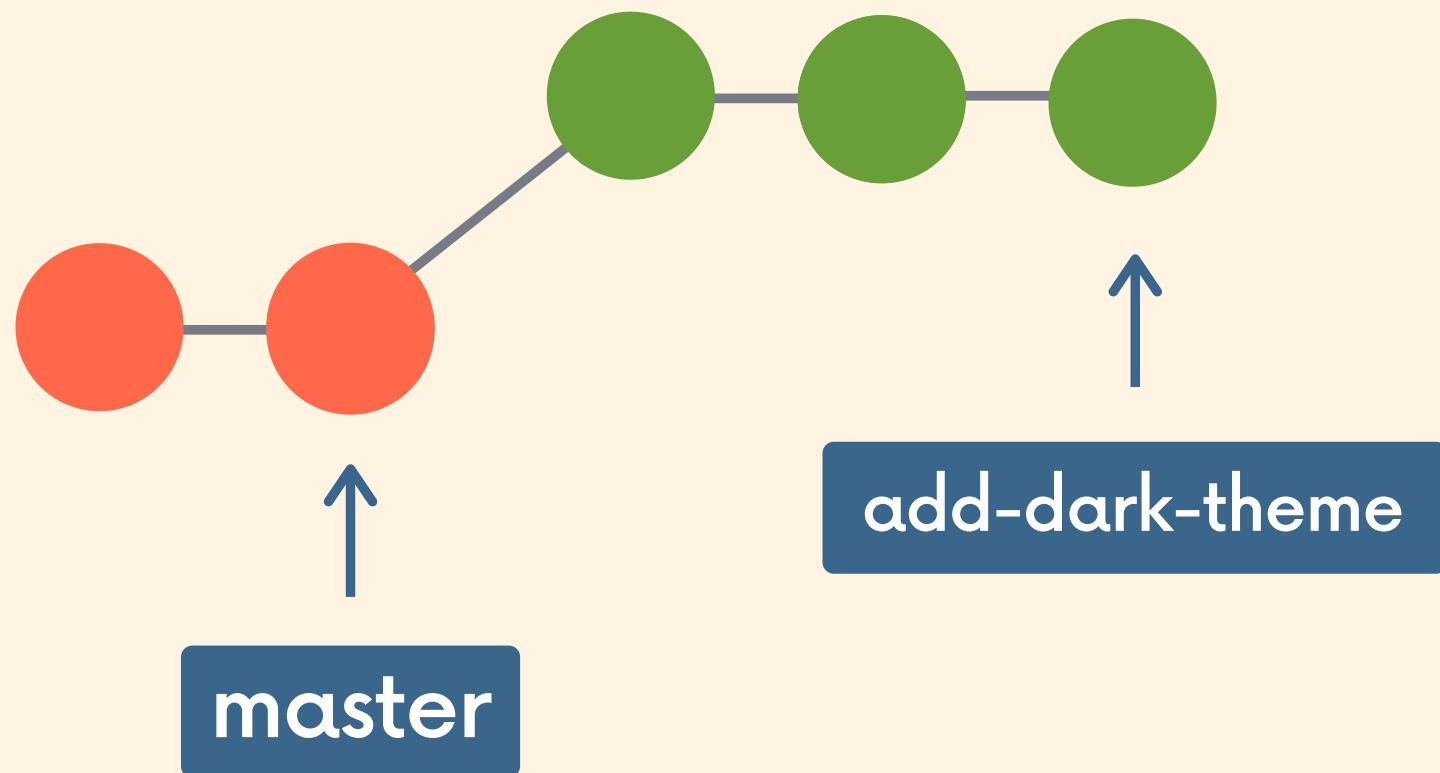
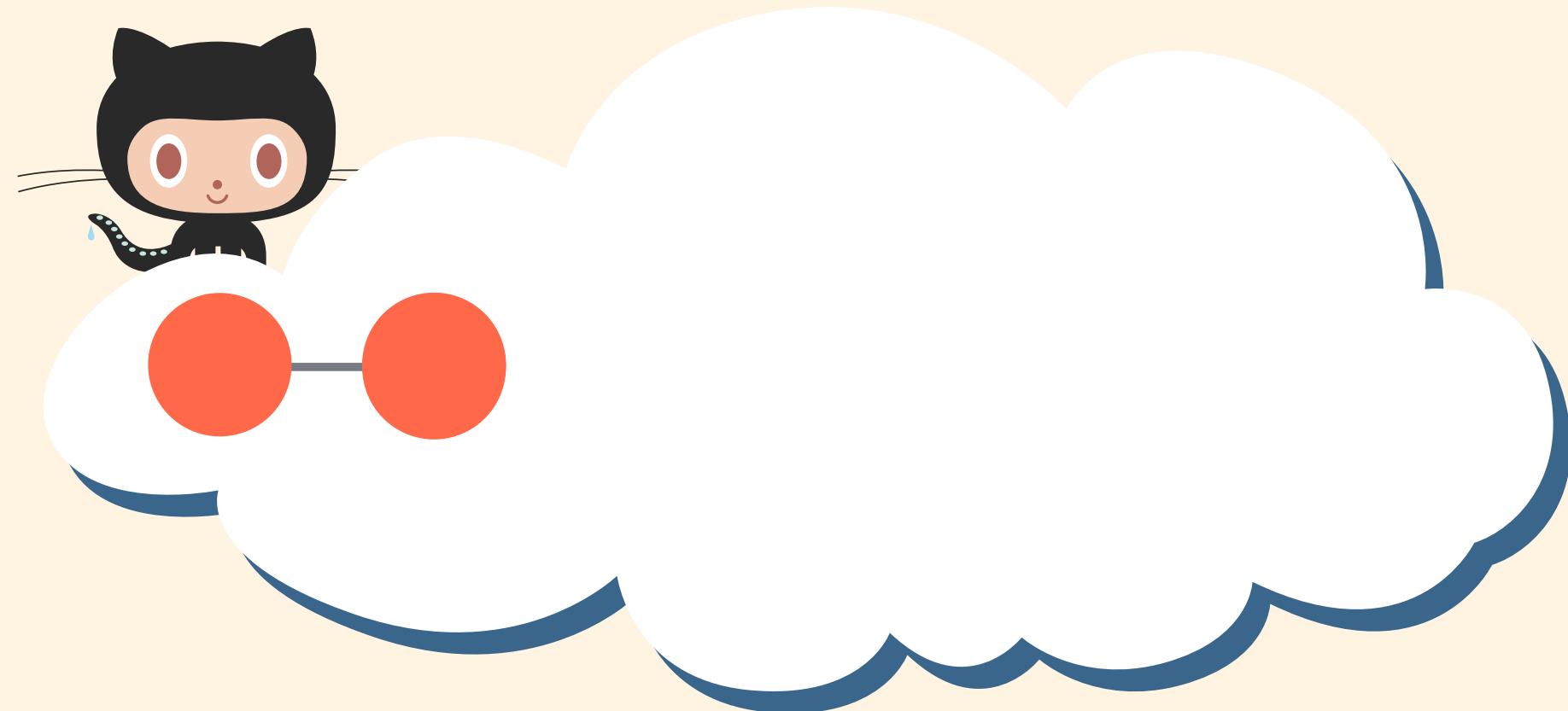
David clones the repo

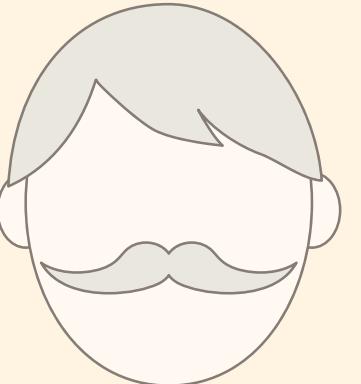
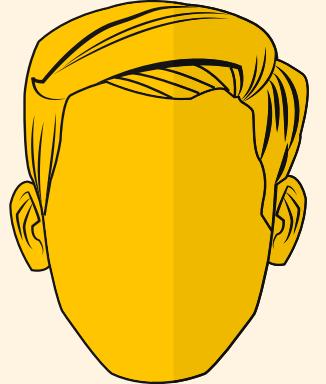
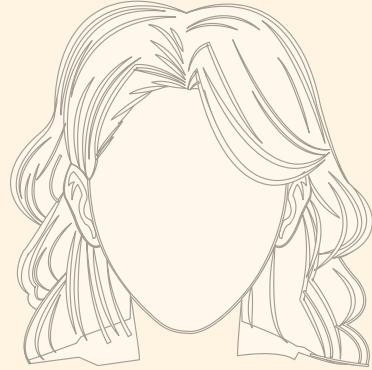


master

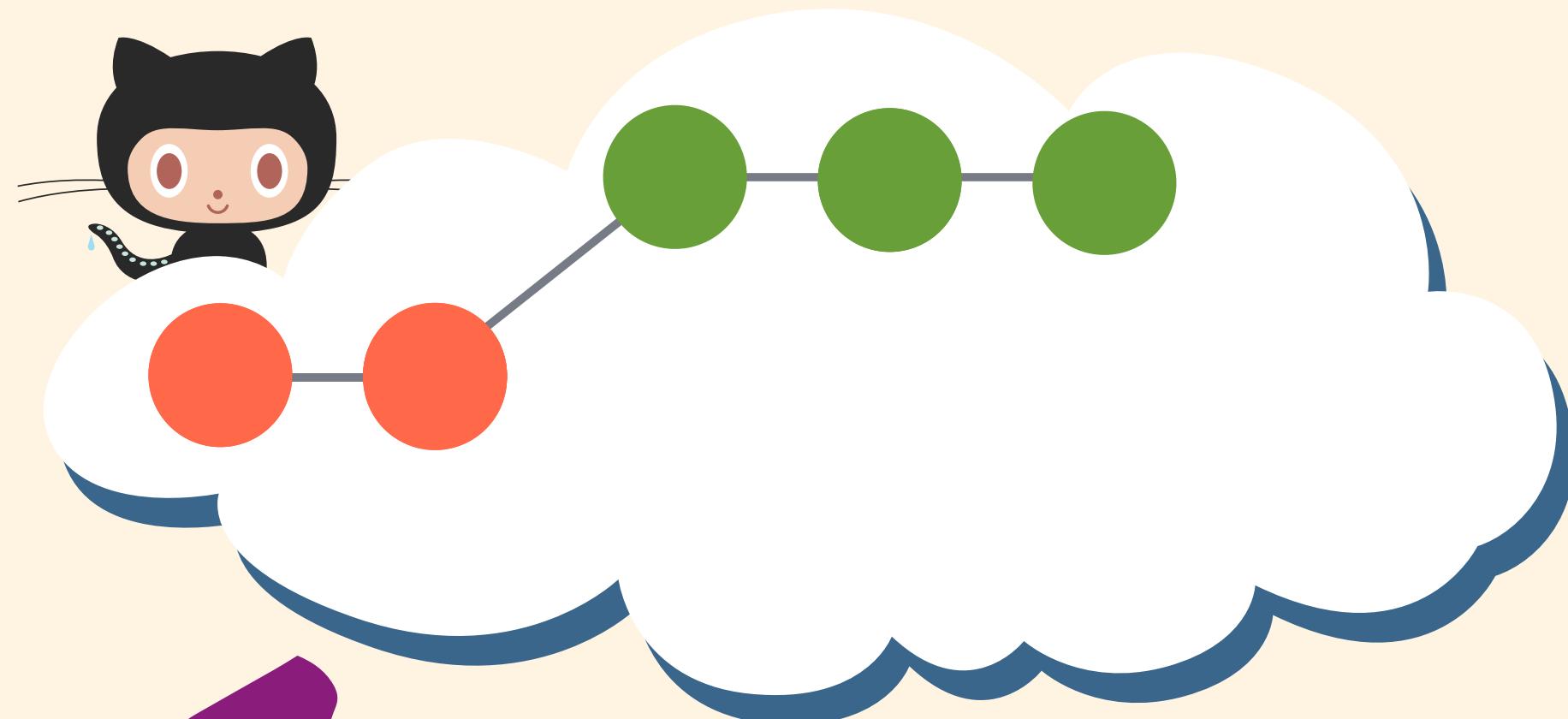
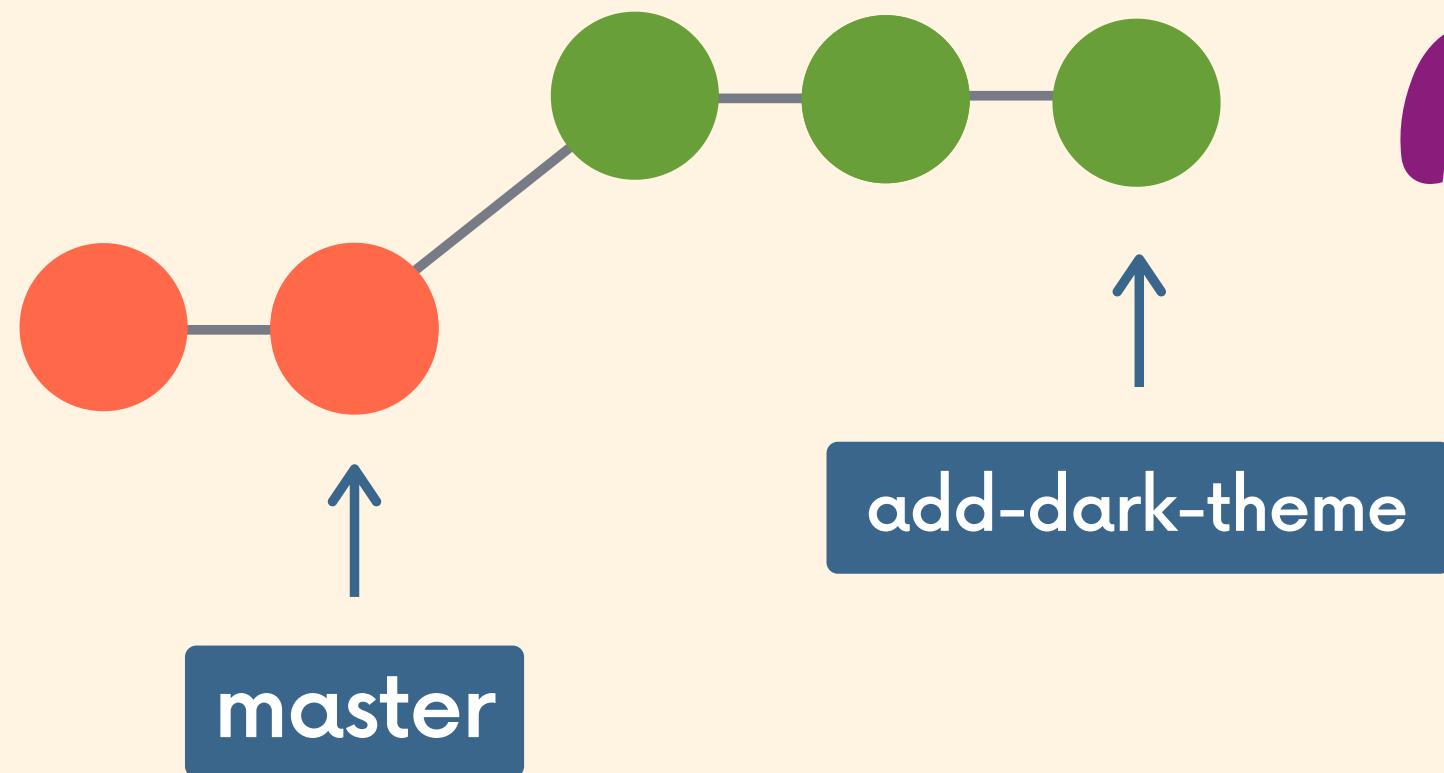


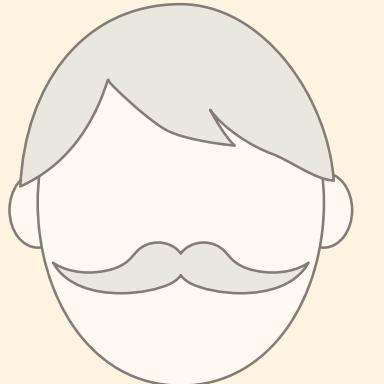
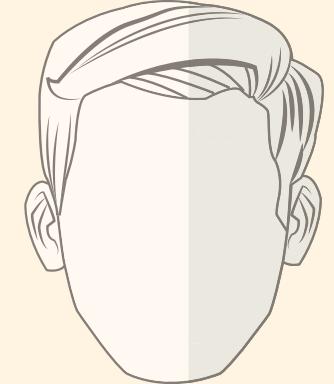
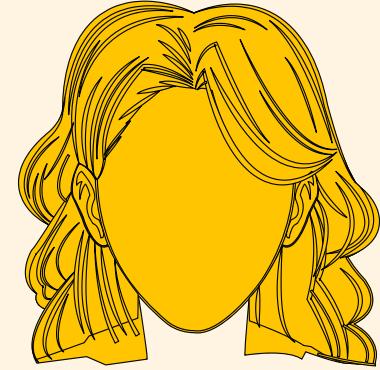
David starts work on a new feature. He does all this work on a separate branch!



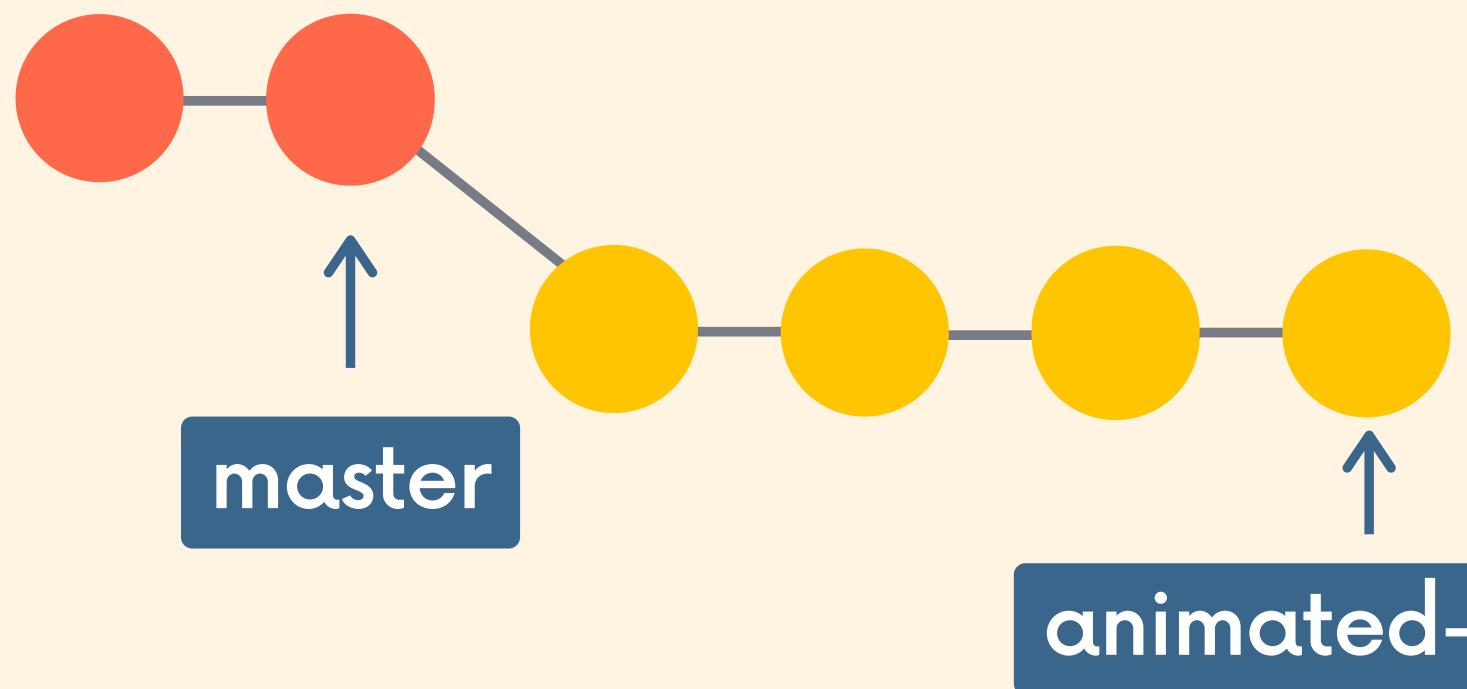
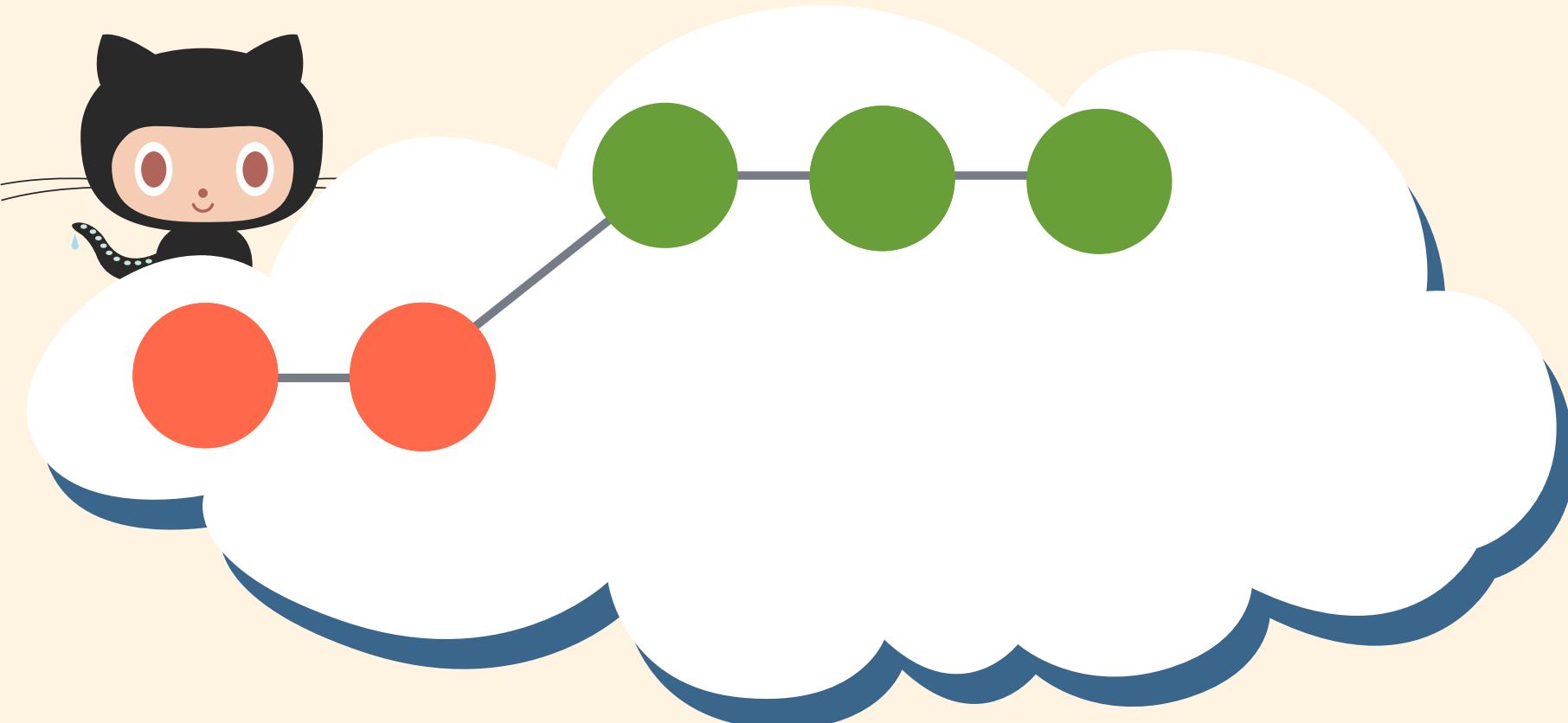


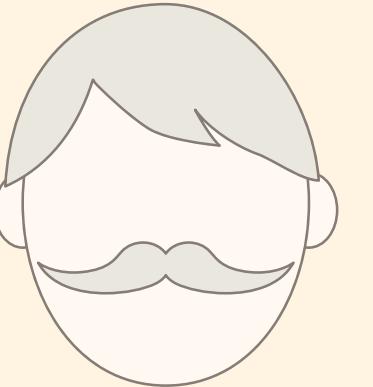
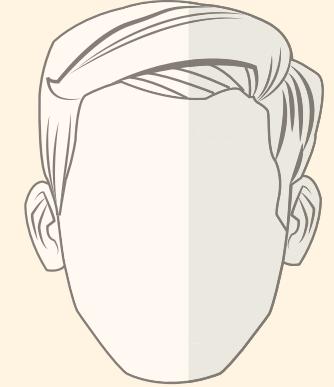
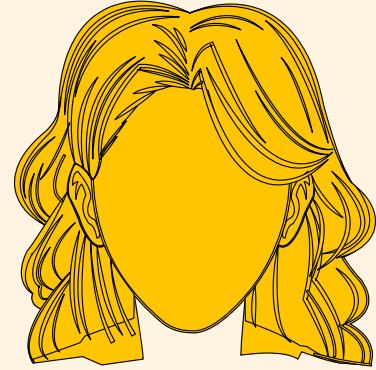
David wants Pamela to take a look at his new feature. Instead of merging it into master, he just pushes his feature branch up to Github!



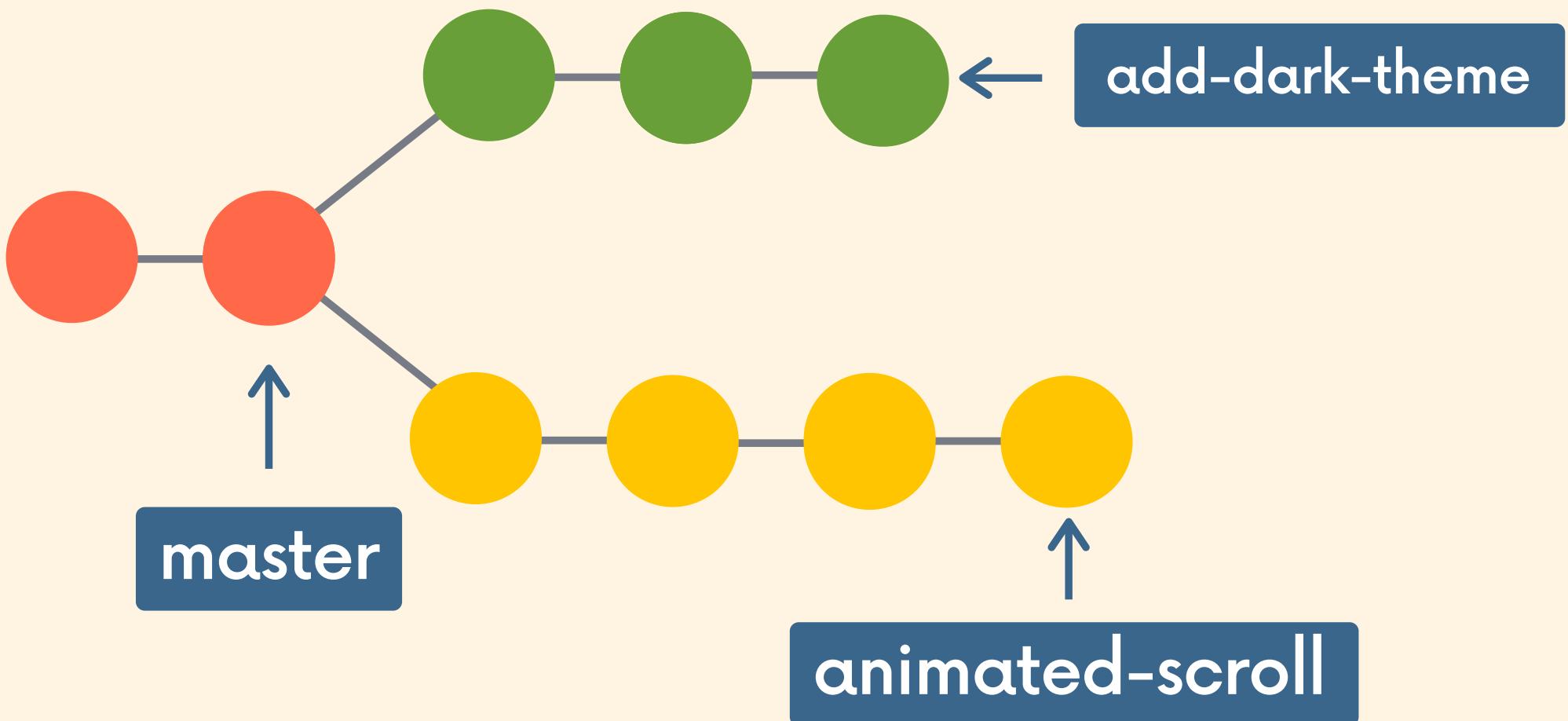
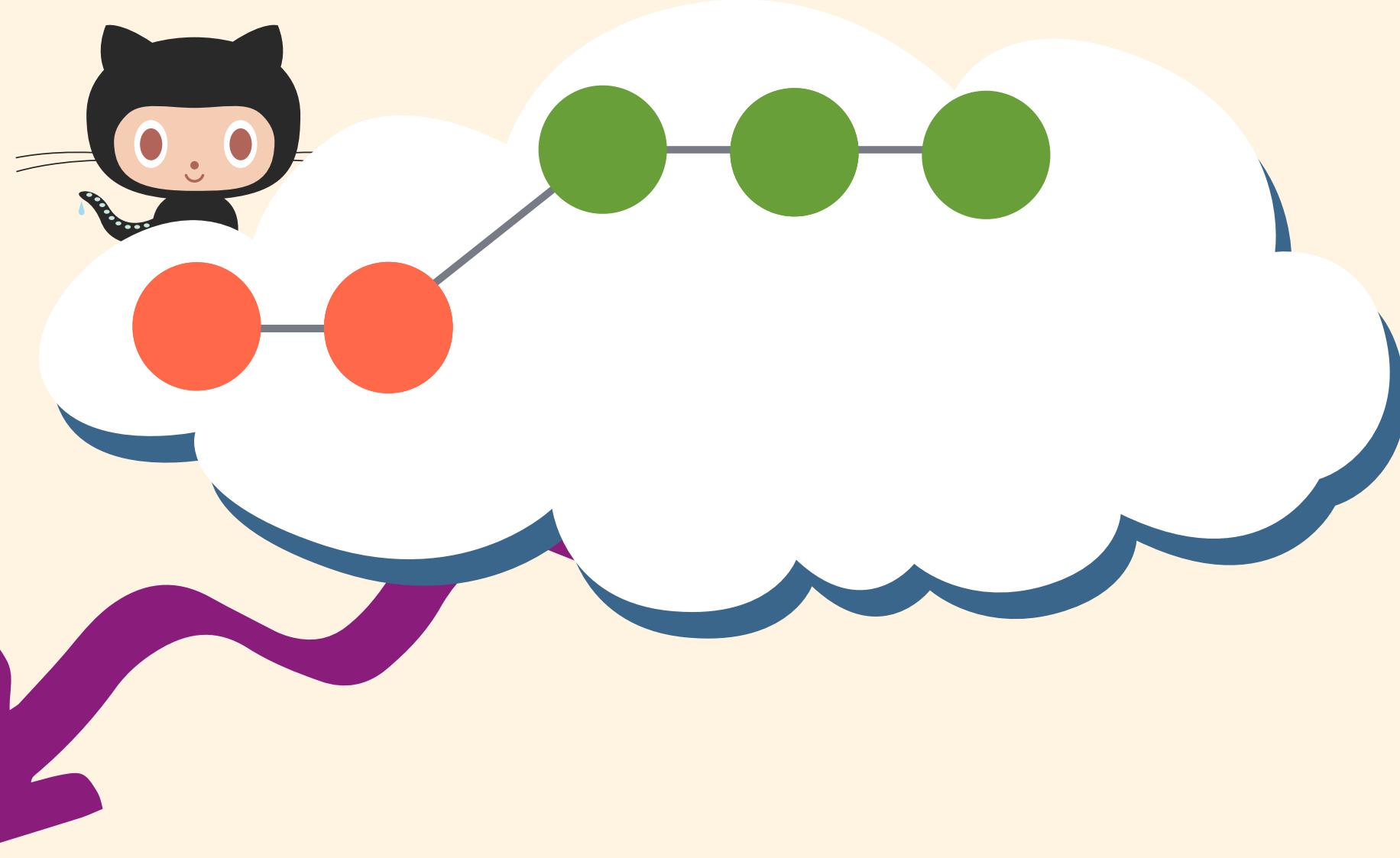


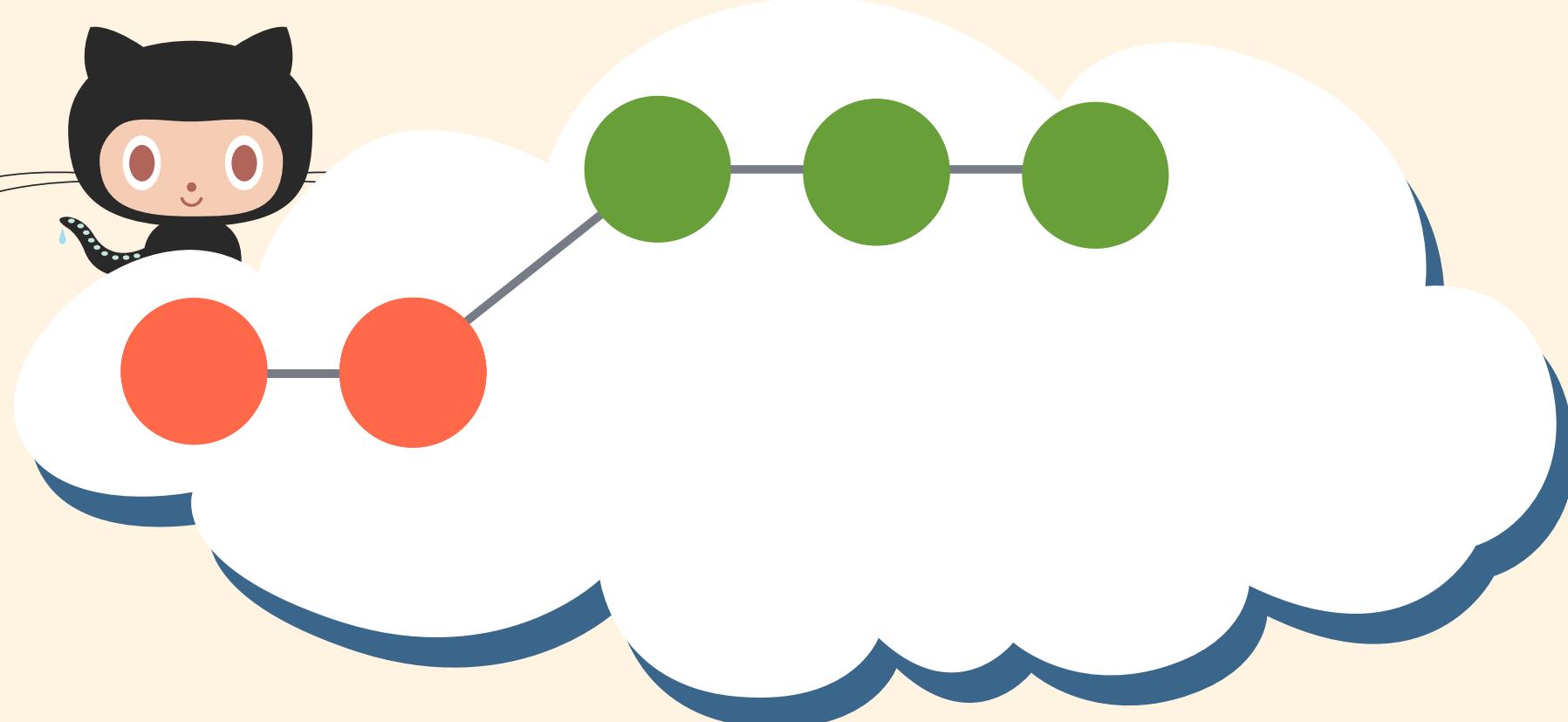
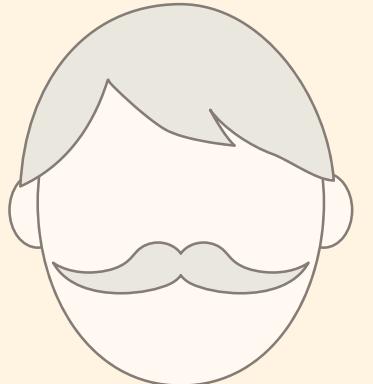
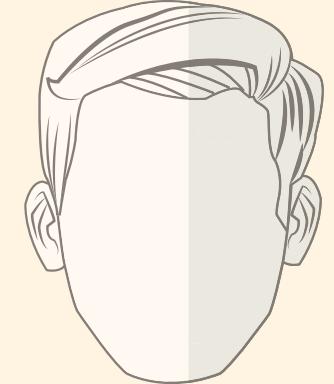
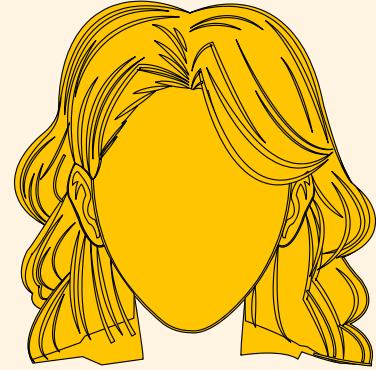
Pamela is hard at work on her own new feature. Just like everyone else, she's working on a separate feature branch rather than master.



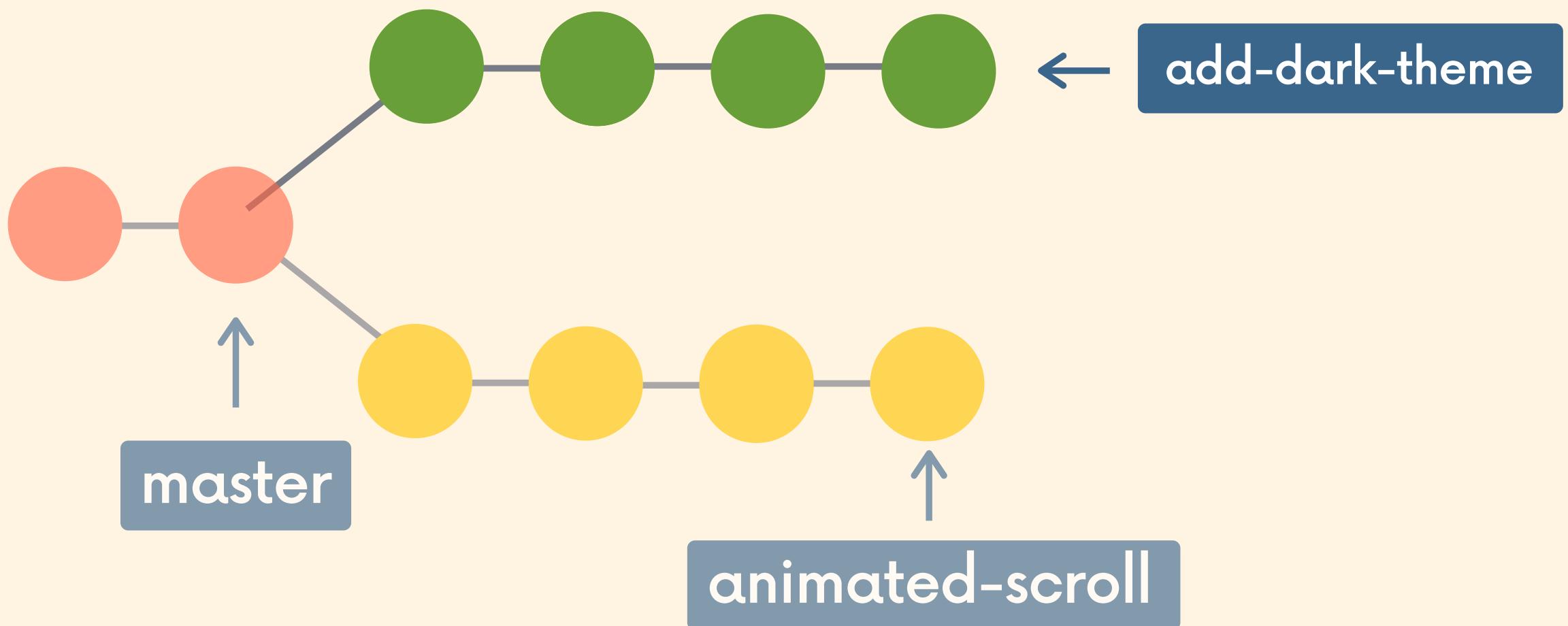


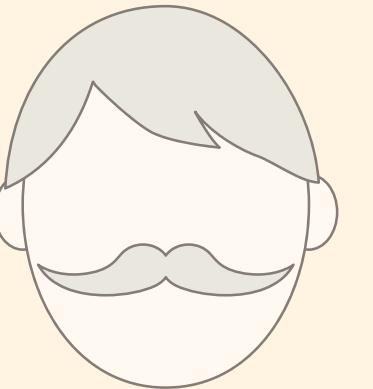
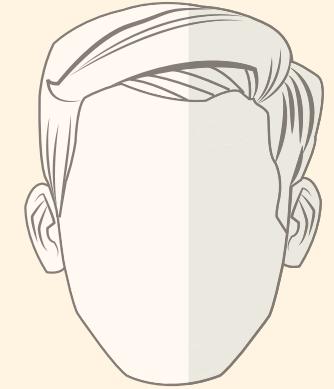
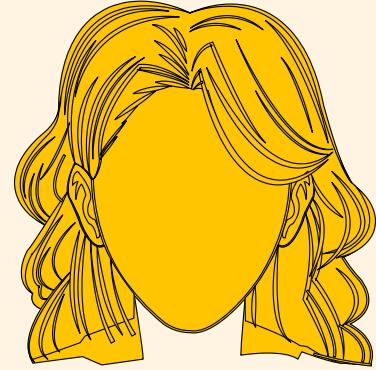
Pamela hears from David that he wants her to take a look at his new work. She pulls down his feature branch from Github.



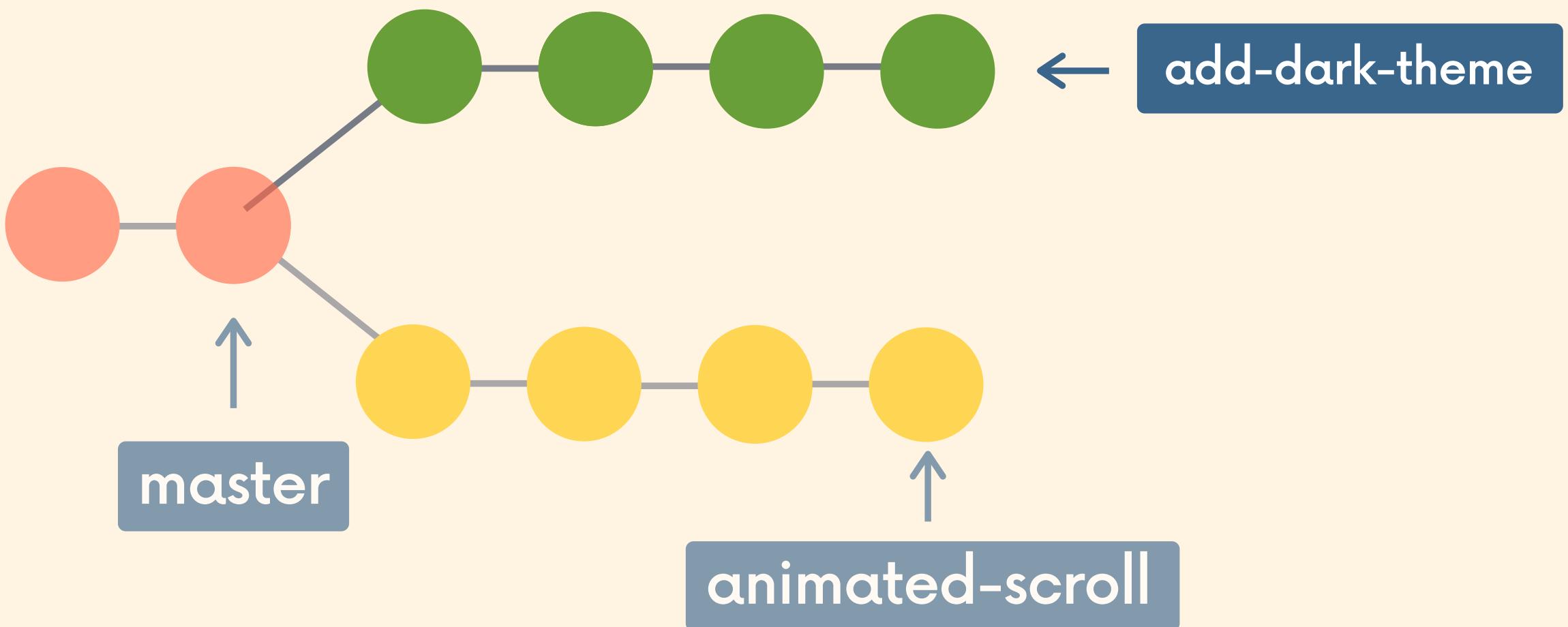
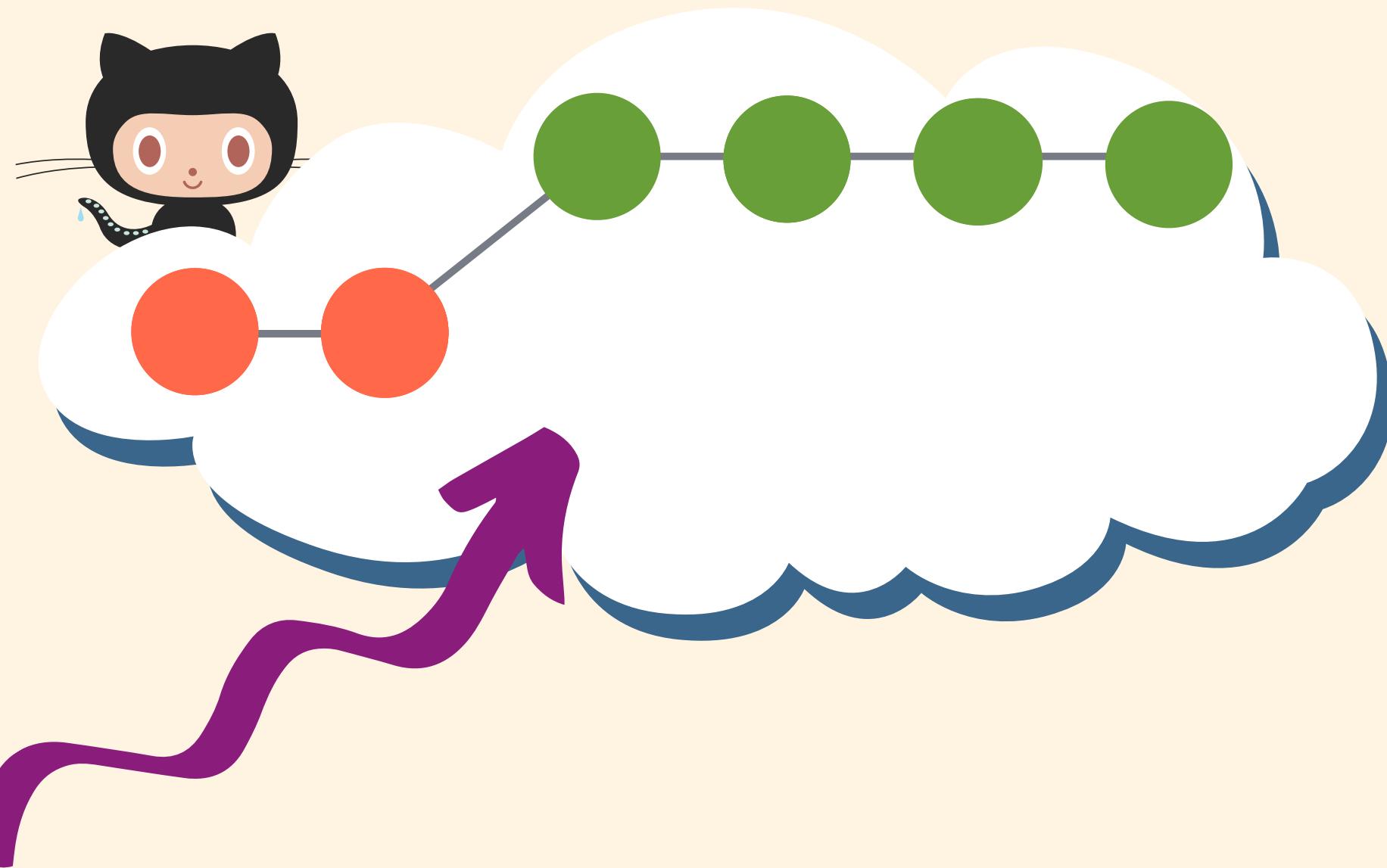


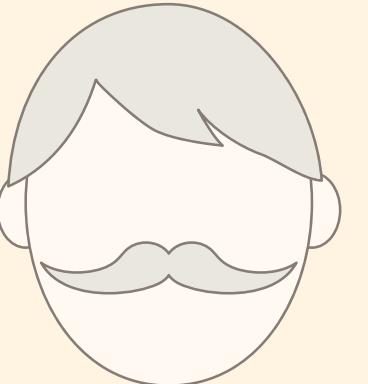
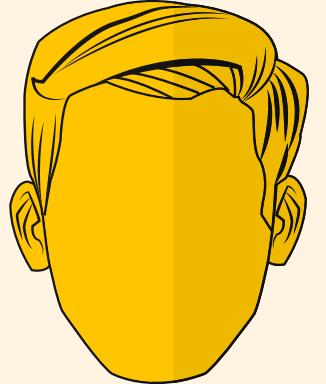
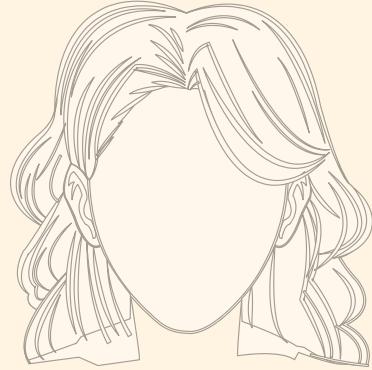
Pamela takes a look at the code and makes a couple improvements of her own. She adds/commits on the same feature branch.



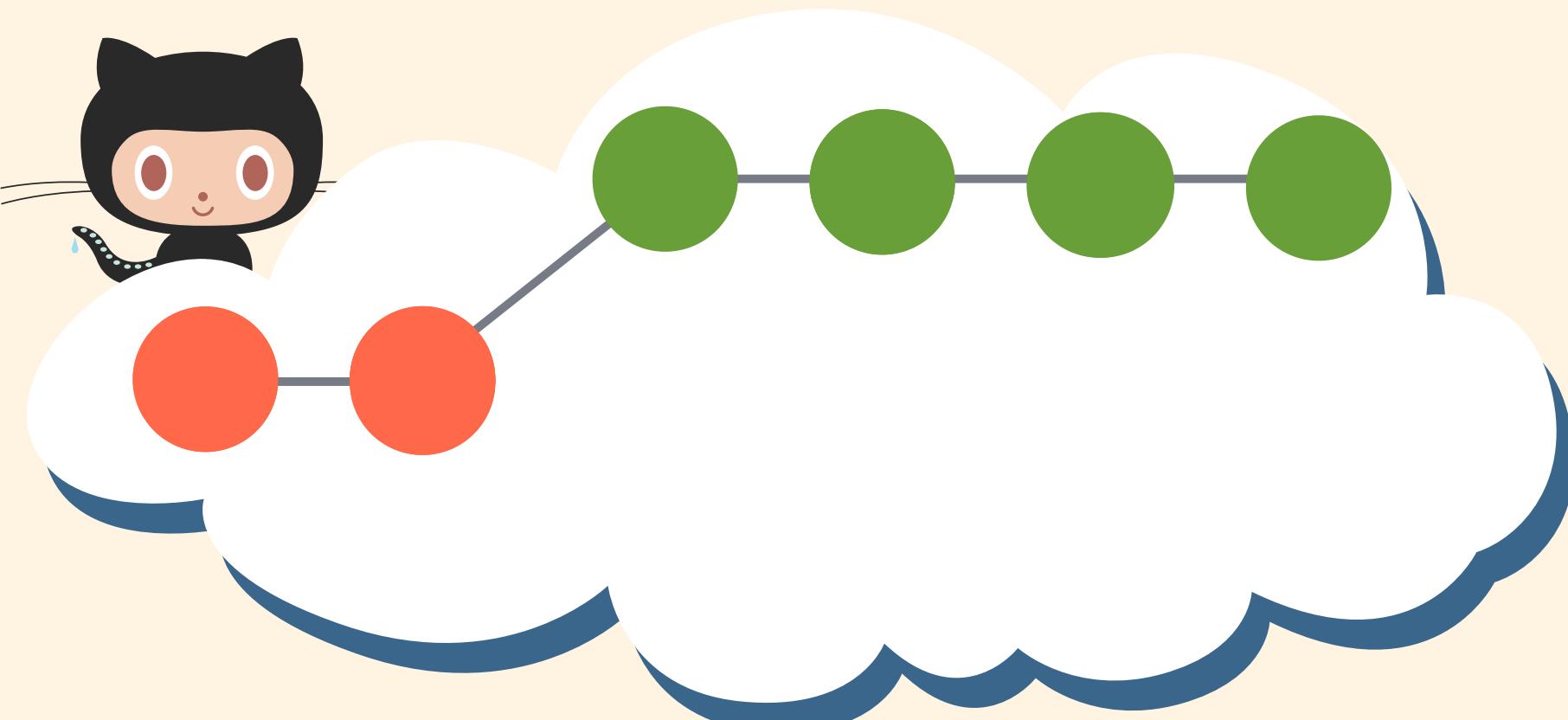
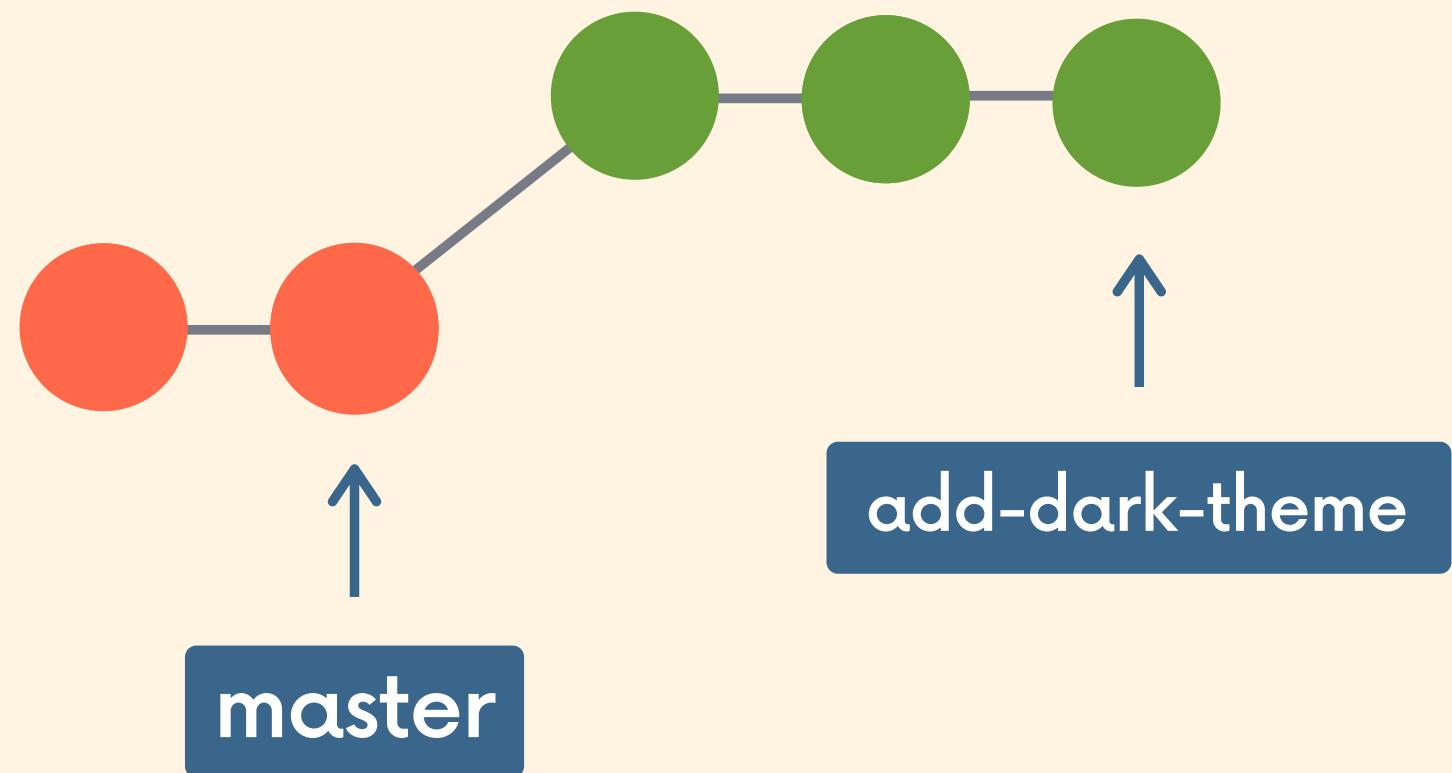


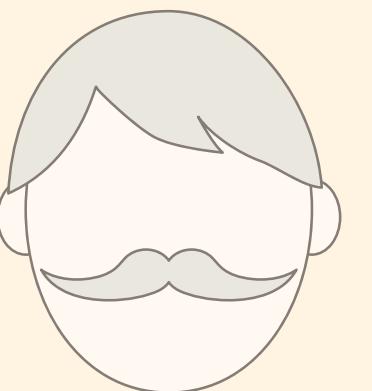
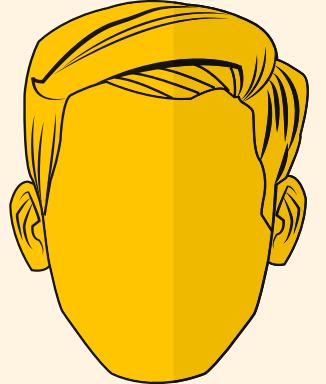
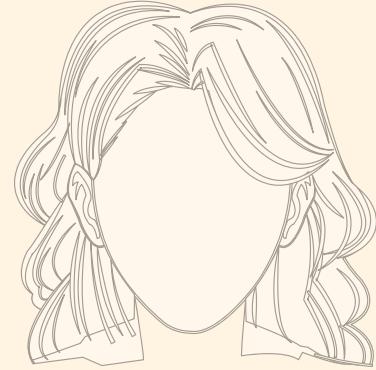
Pamela pushes up her new work on
the add-dark-theme feature branch
so that David can pull them down.



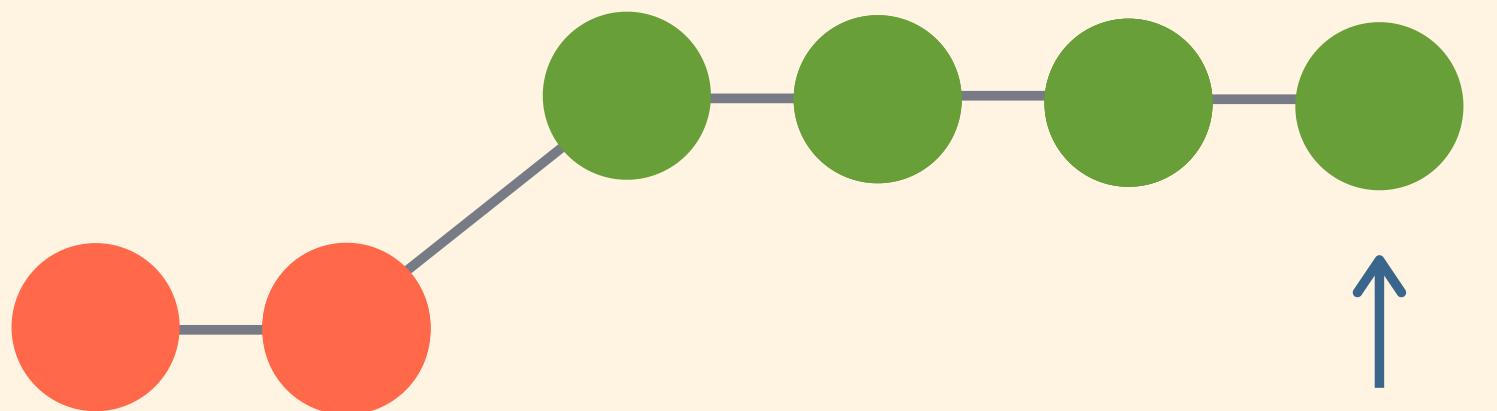
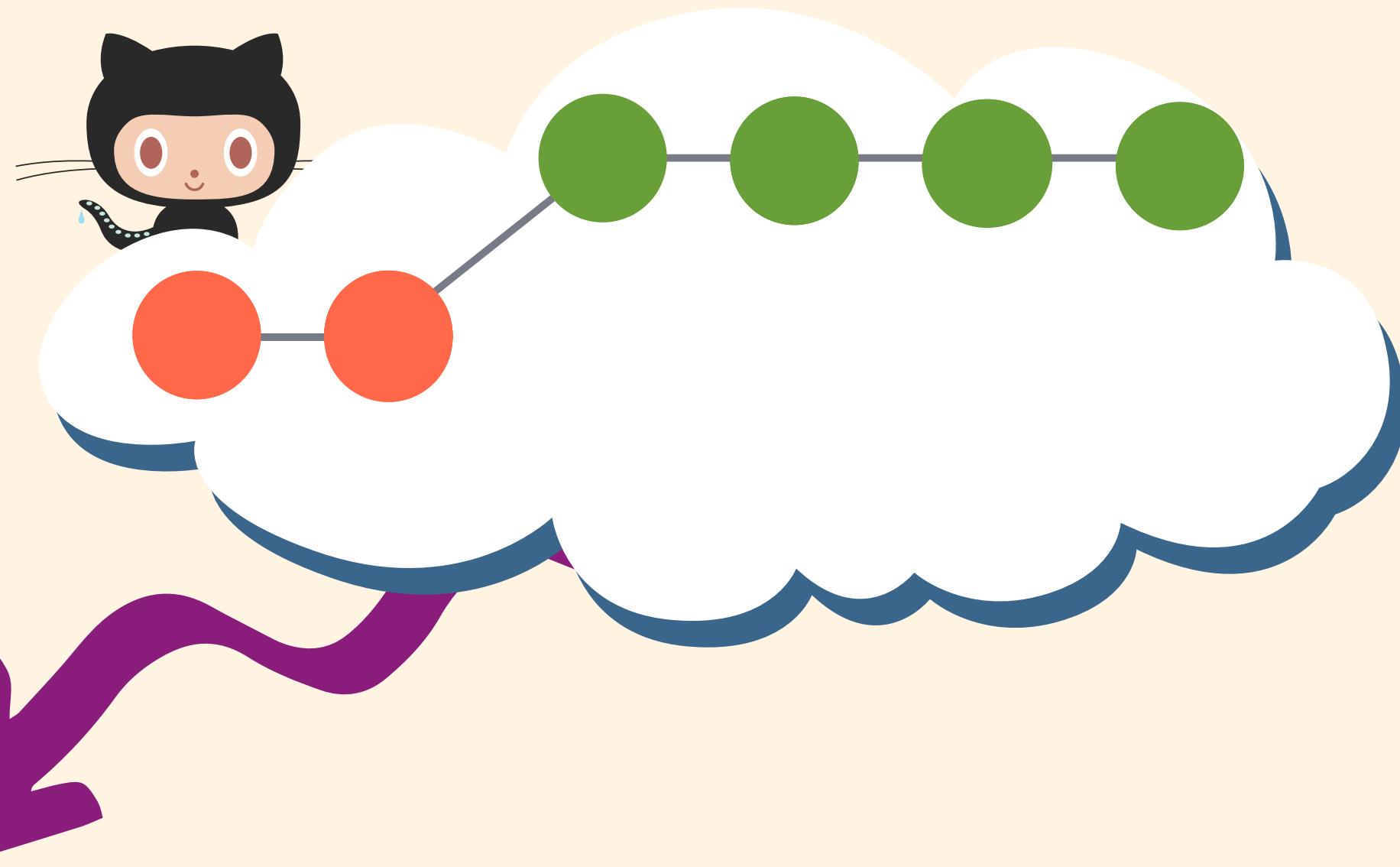


David returns to work the next morning. After an hour on reddit he decides he should actually do some work.



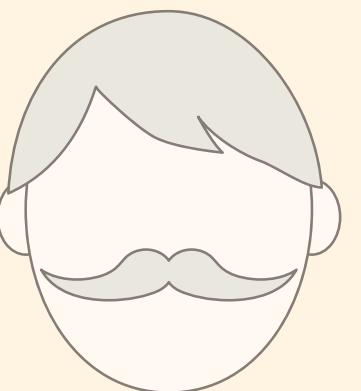
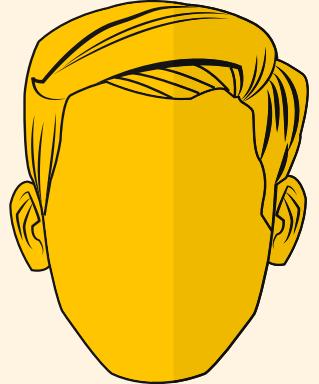
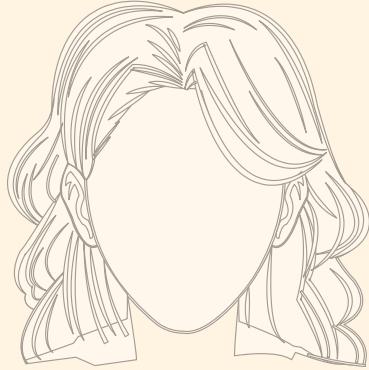


David fetches from Github and sees that there is new work on the add-dark-theme branch. He pulls the changes down and continues work.

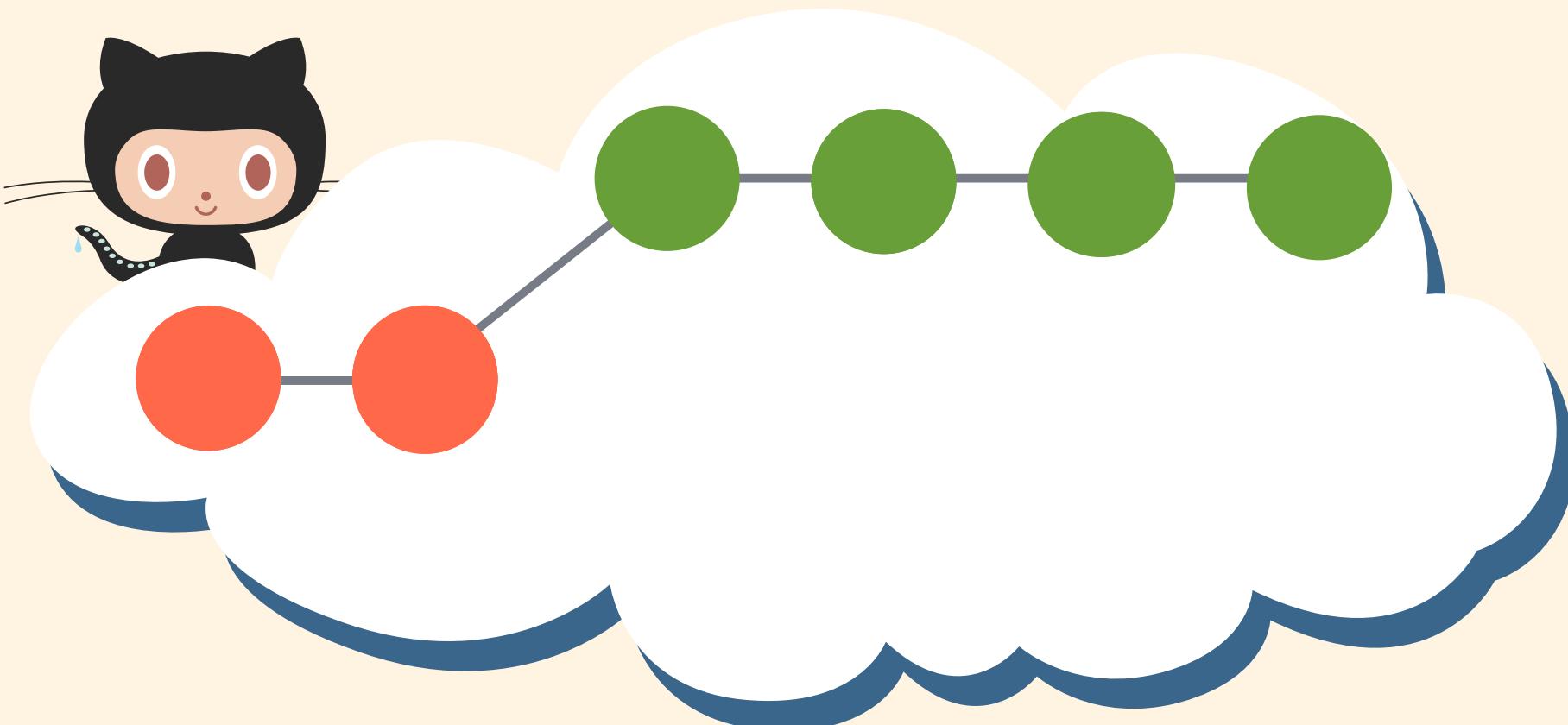
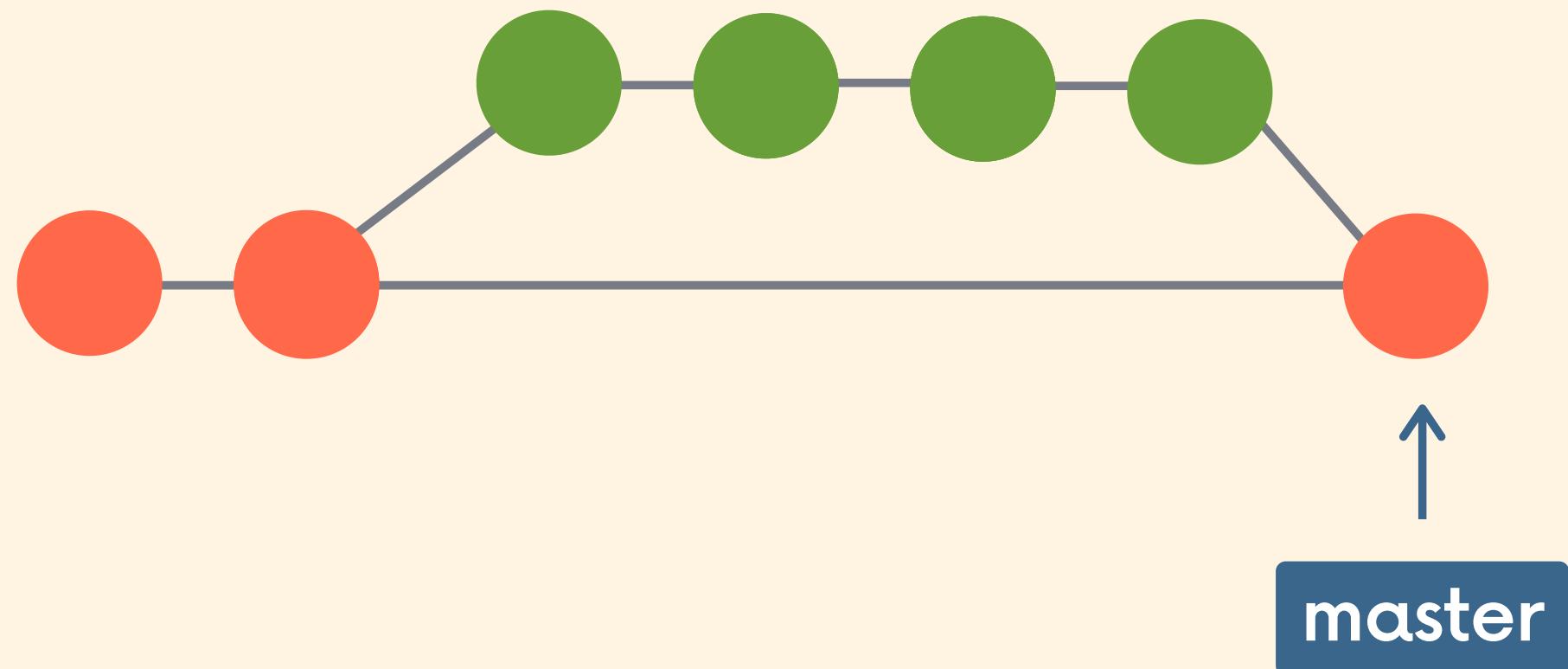


add-dark-theme

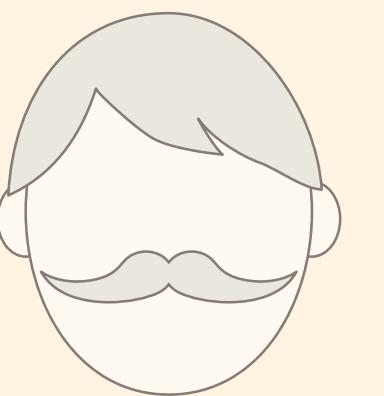
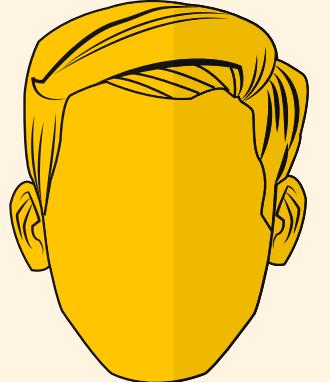
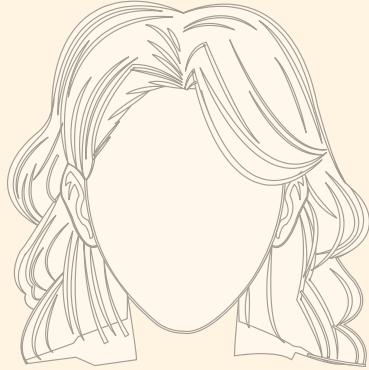
master



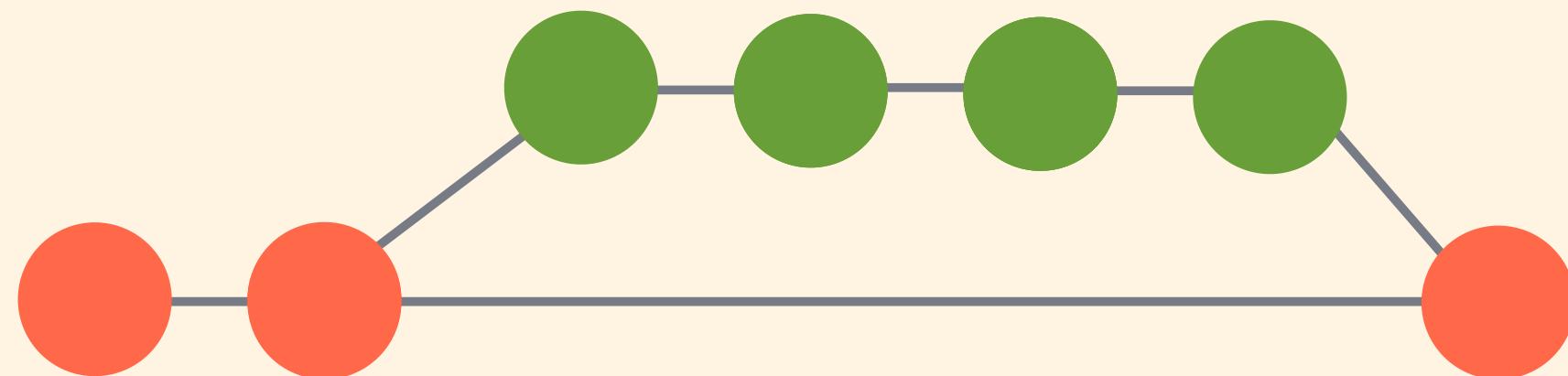
David decides he is happy with the new feature, so he merges it into master!



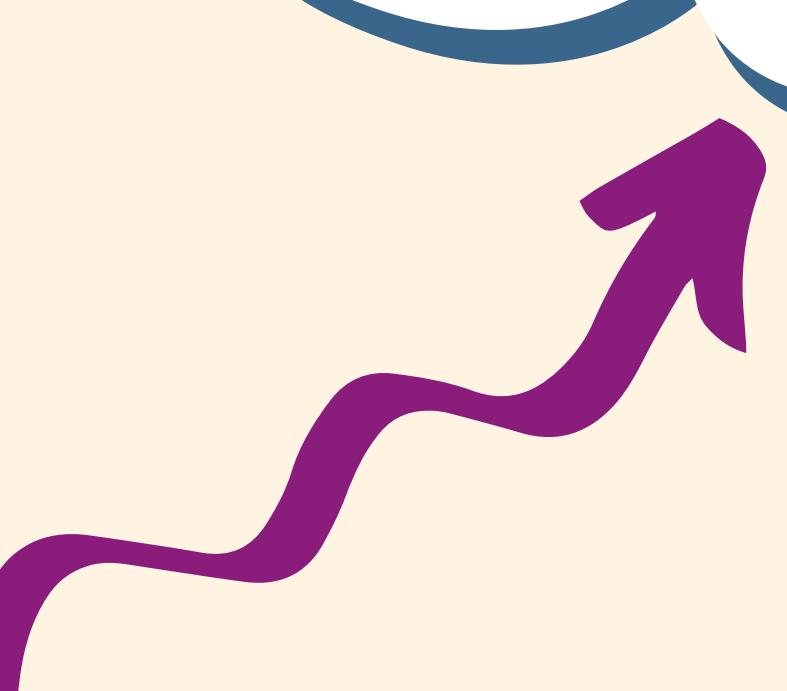
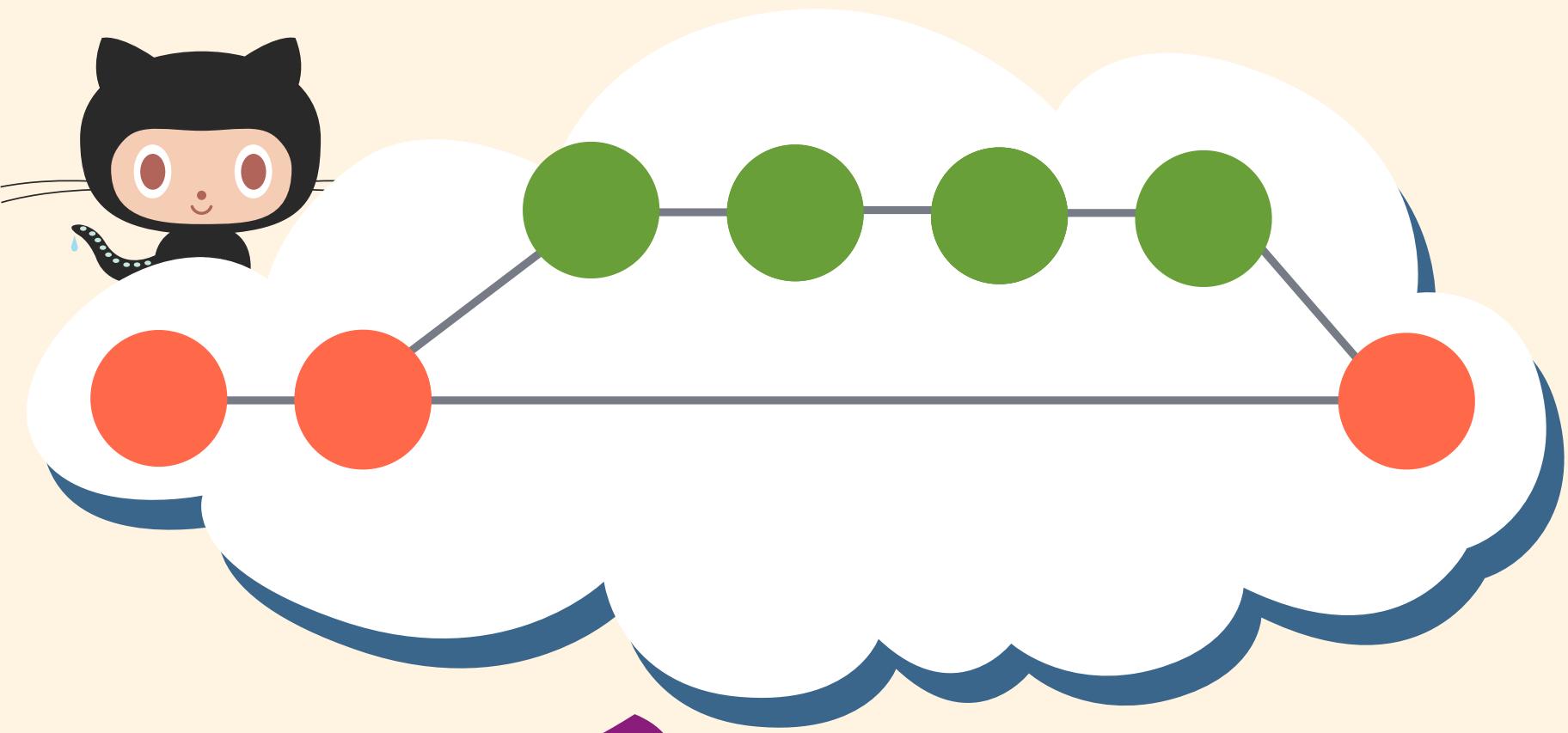
*At a real company, you don't just decide you are "happy with" a feature. There are mechanisms for code approval and merging we'll discuss shortly!



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



master





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.

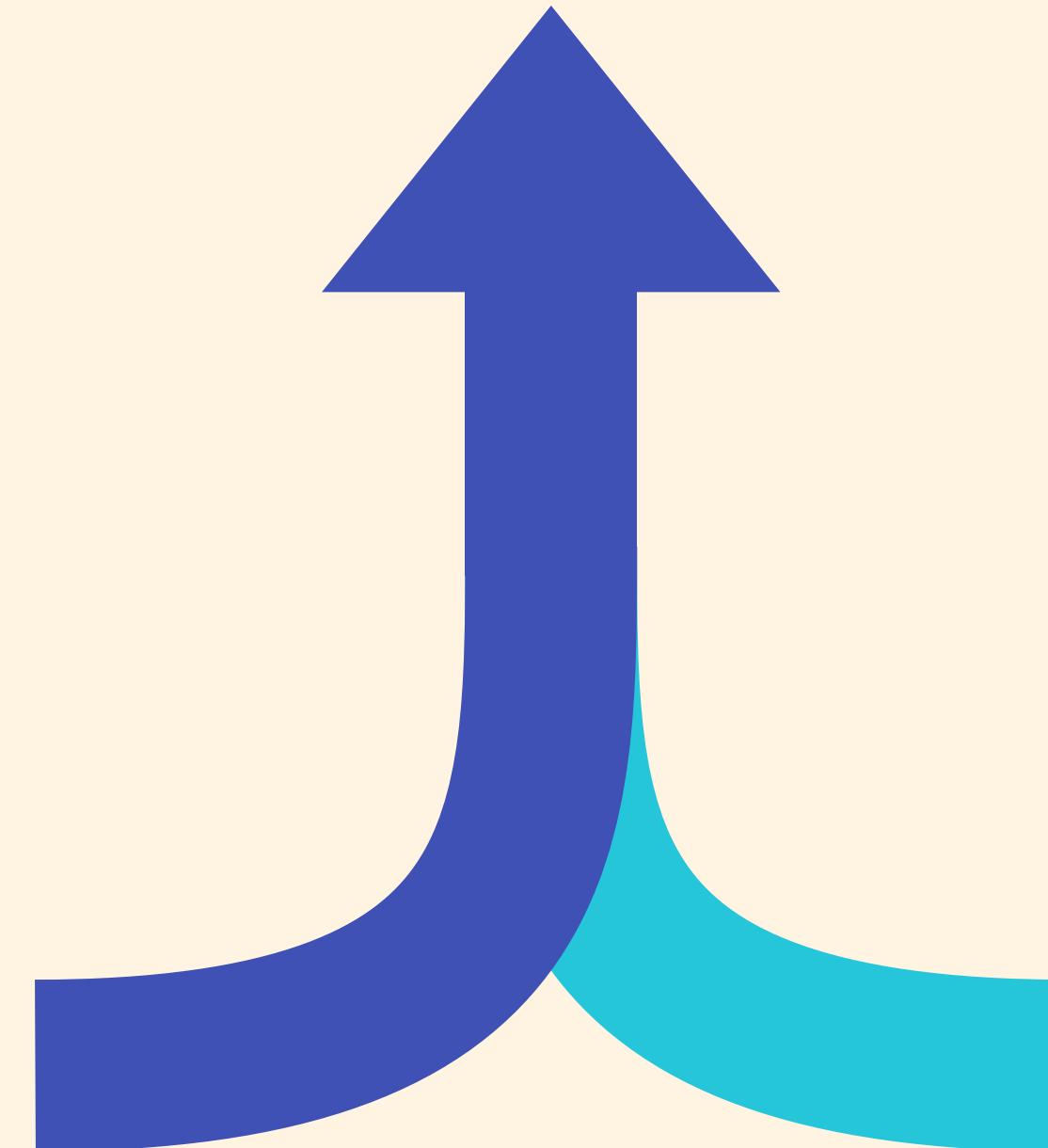




Merging In Feature Branches

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?"

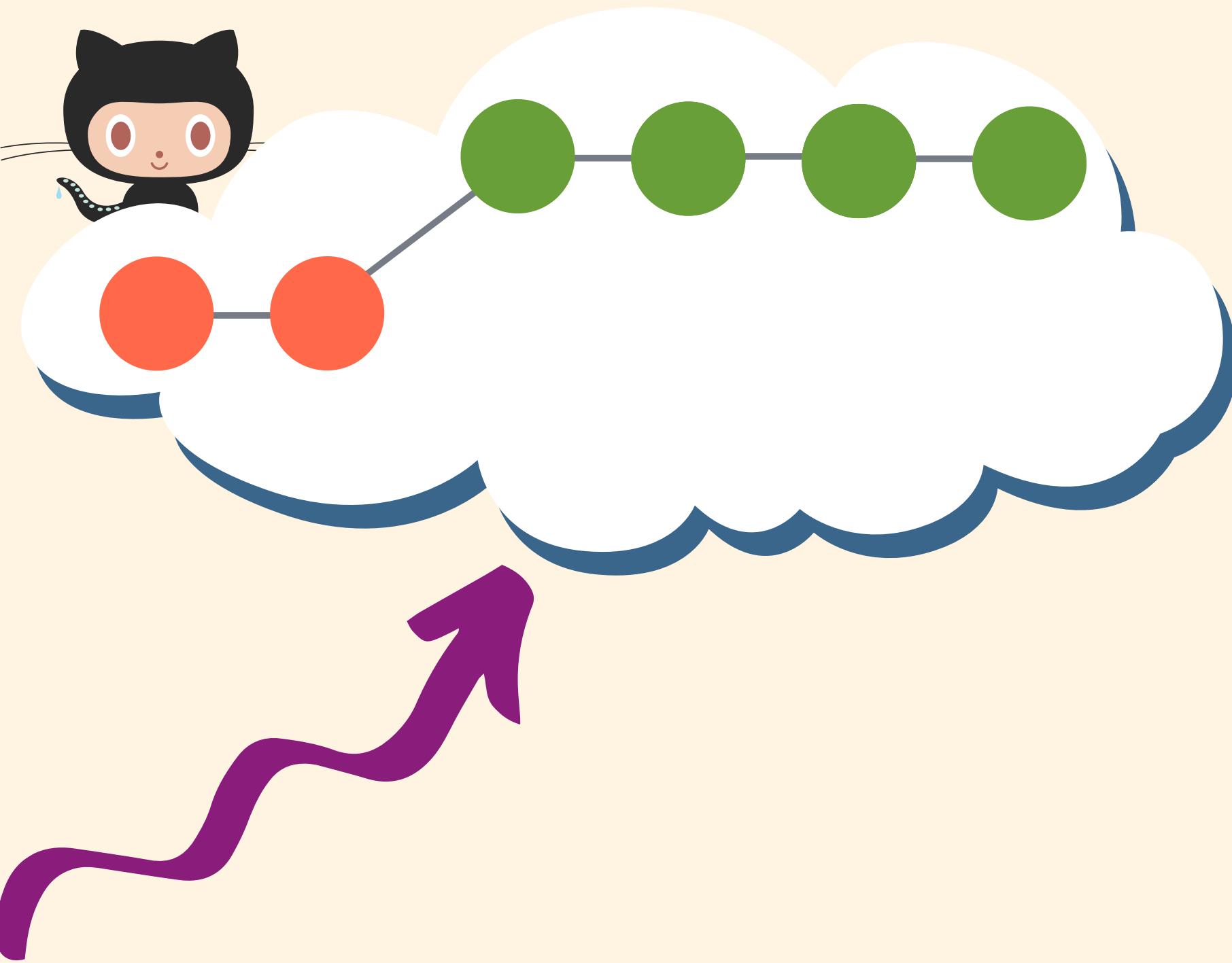
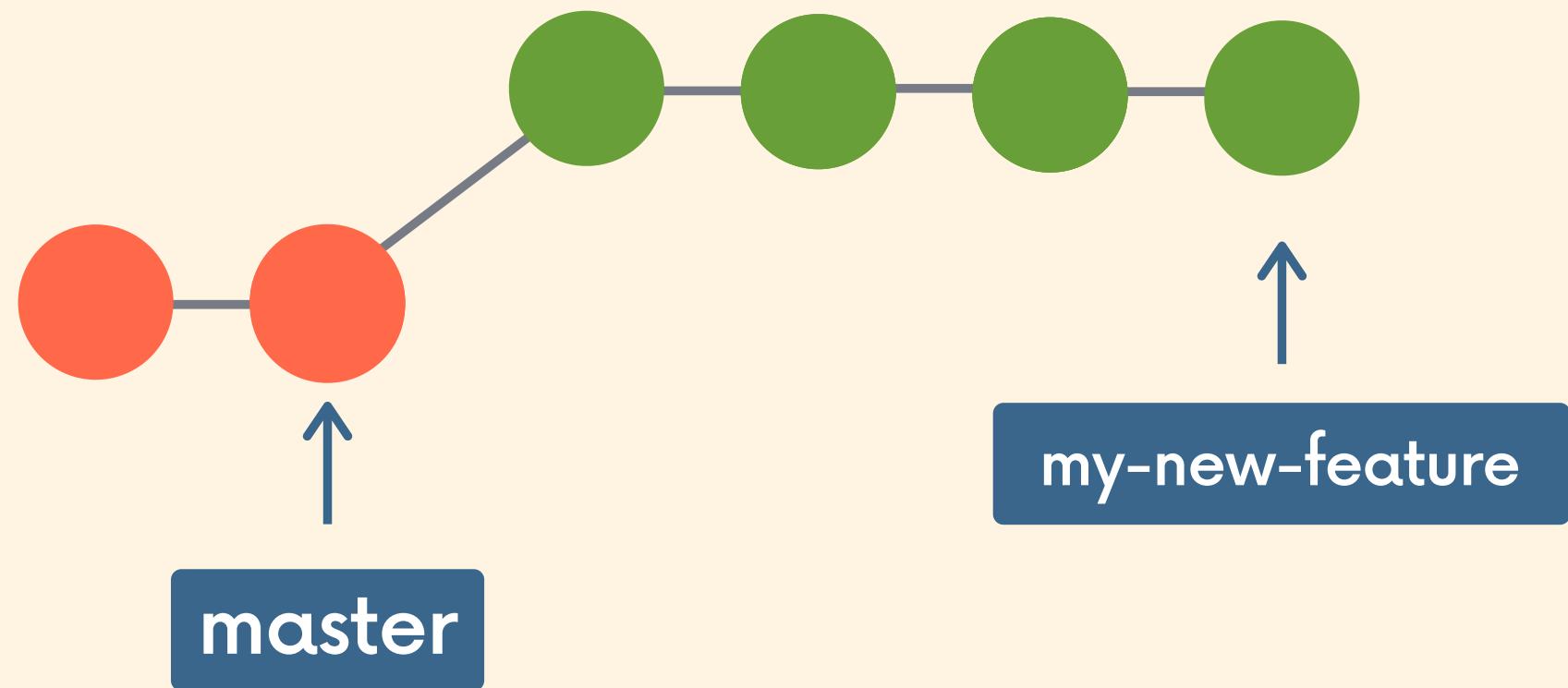


The Workflow

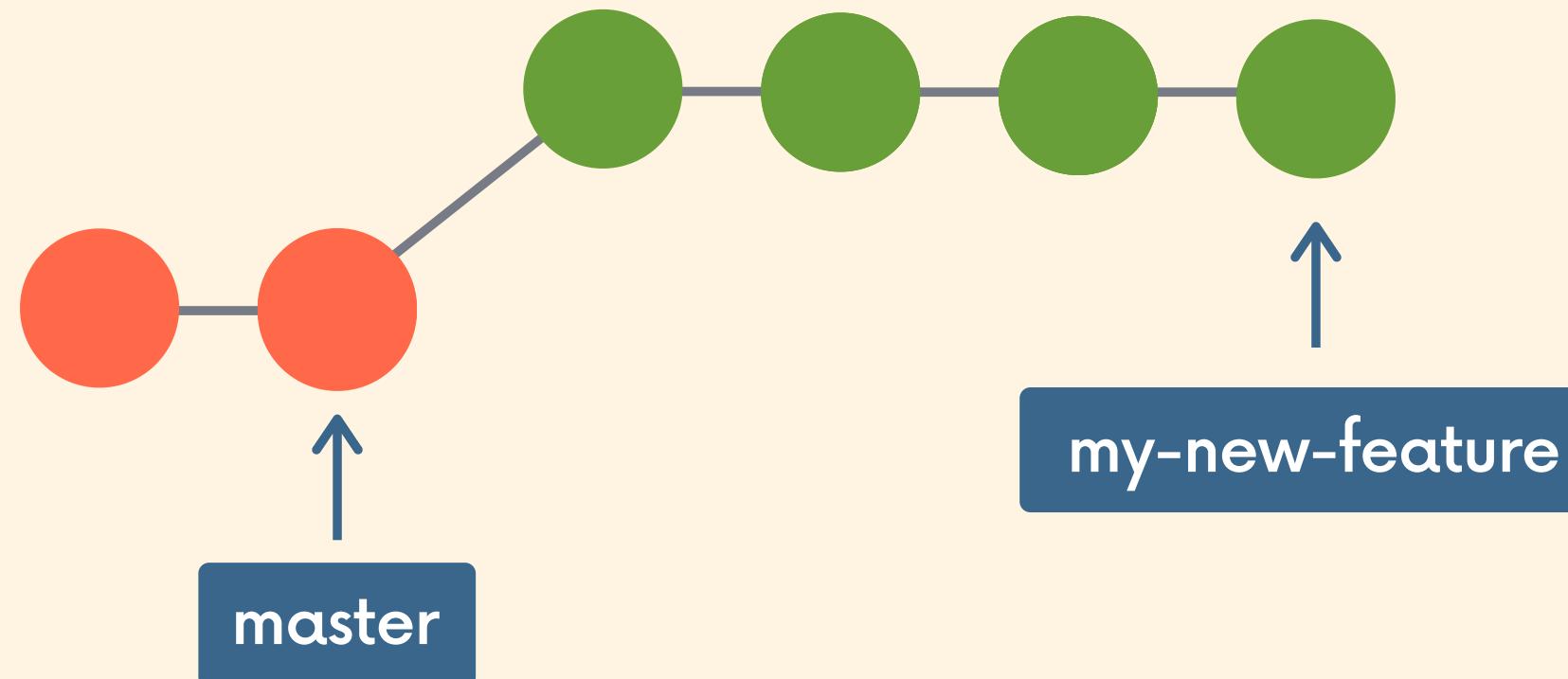
1. Do some work locally on a feature branch
2. Push up the feature branch to Github
3. Open a pull request using the feature branch just pushed up to Github
4. Wait for the PR to be approved and merged. Start a discussion on the PR. This part depends on the team structure.



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](#)

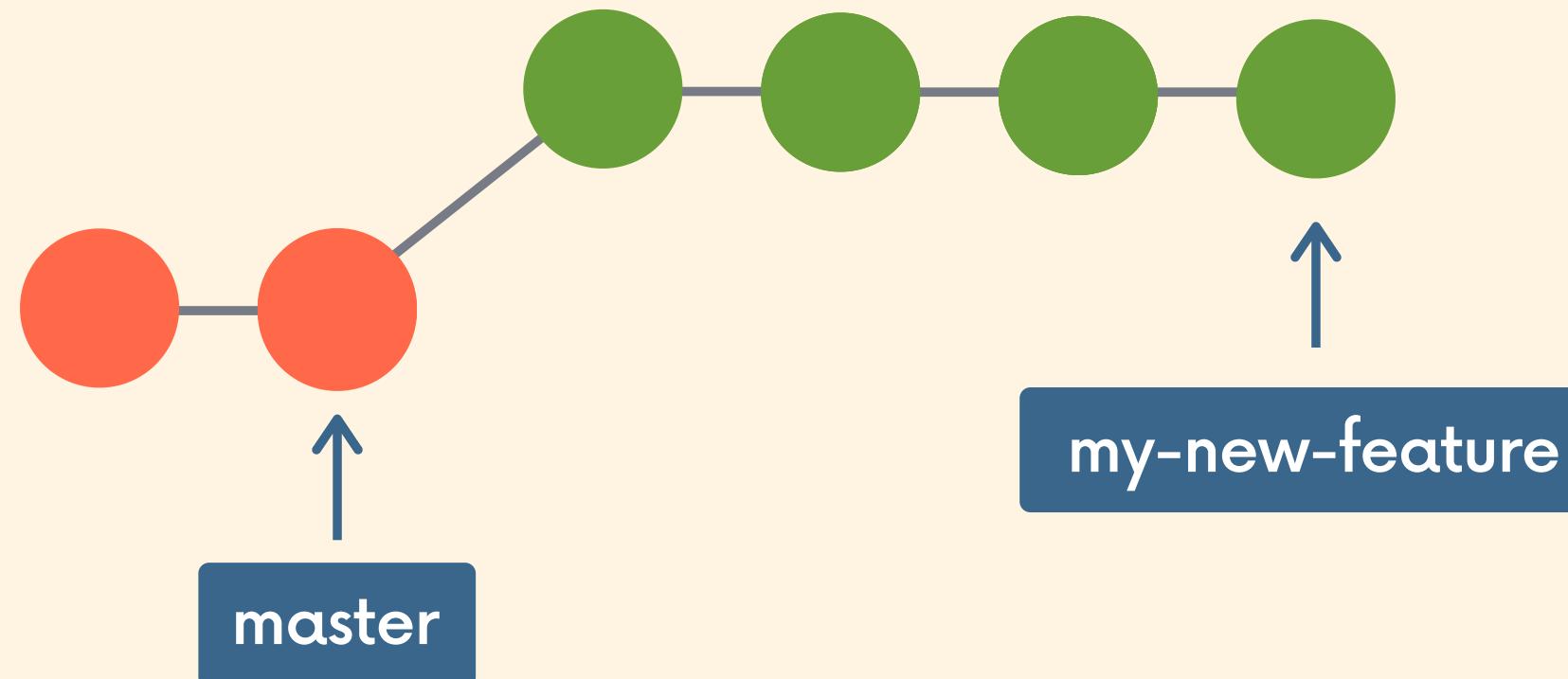
⚡ my-new-feature ▾ ⚡ 5 branches ⚡ 0 tags [Go to file](#) [Add file ▾](#) [Code ▾](#)

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](#)

Colt Steele and Colt Steele add another new file d3f

[anotherNewFile.txt](#) add another new file 32 seconds ago

[newfeature.txt](#) add new feature 1 minute ago

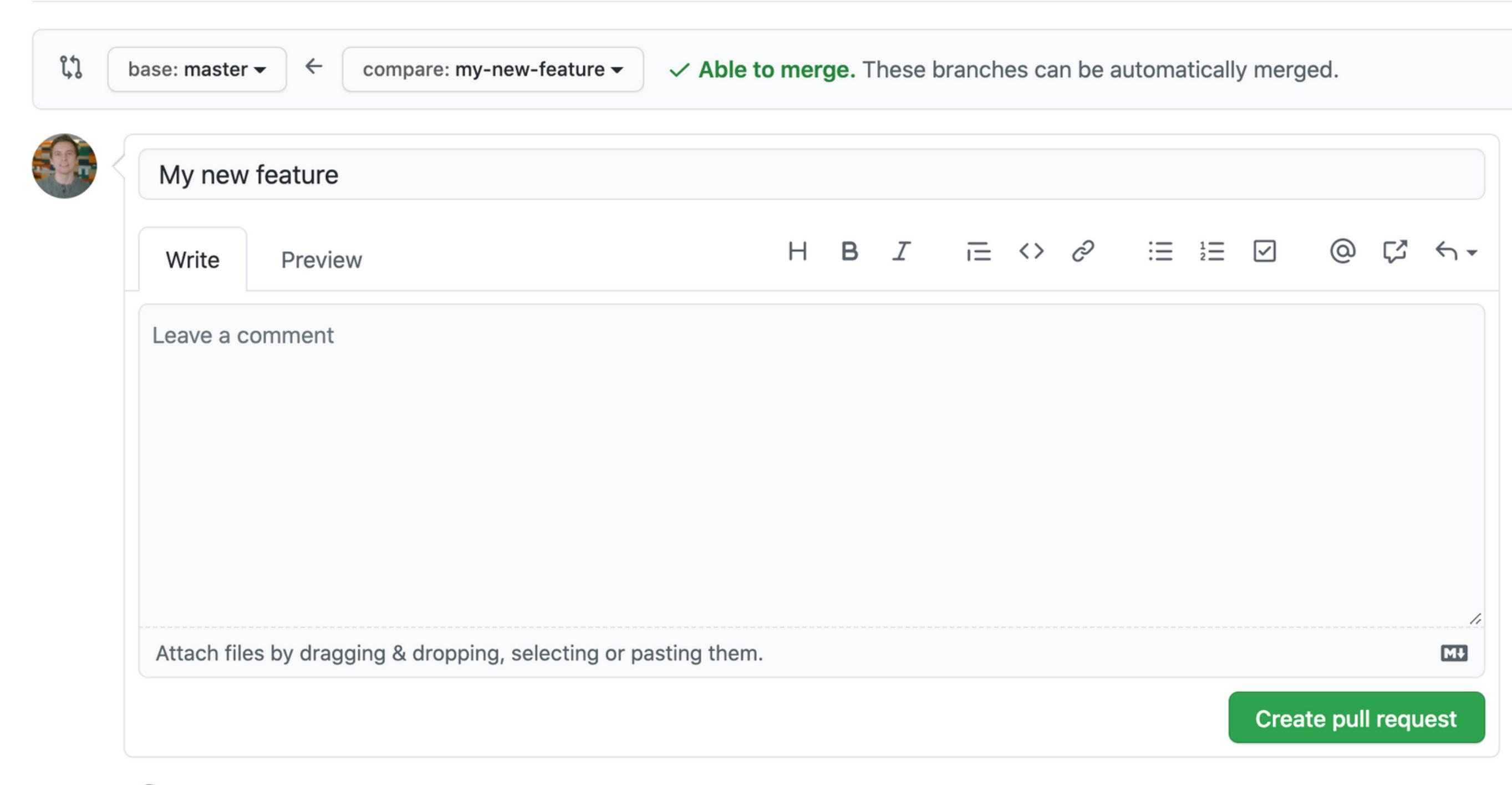
[playlist.txt](#) merge 5 days ago

I click the PR button

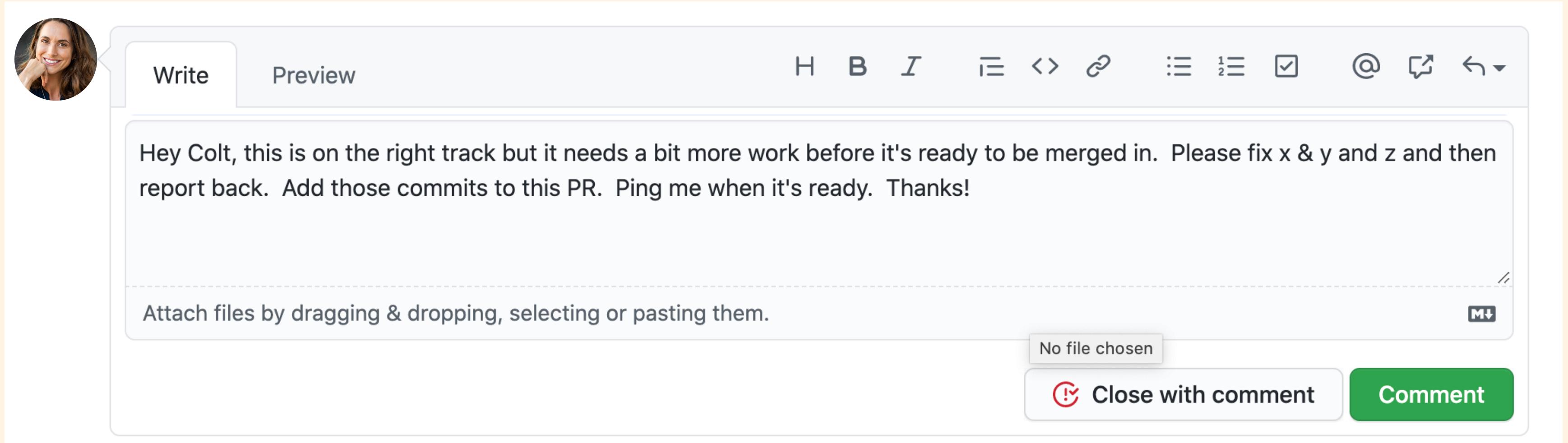
My Github

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](#).



My Boss's Github...



A screenshot of a GitHub pull request comment interface. At the top left is a circular profile picture of a woman with long brown hair, smiling. To her right are two tabs: "Write" (which is selected) and "Preview". Above the main text area are various rich text editing icons: bold (B), italic (I), heading (H), code block (code block icon), link (link icon), list (list icon), checkmark (checkmark icon), atmention (@), reply (reply icon), and back (back icon). 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 area is a dashed line separator. Underneath the separator, there is a placeholder text "Attach files by dragging & dropping, selecting or pasting them." followed by a file upload button with the text "No file chosen" and a "More" link. At the bottom right of the comment area are three buttons: a grey "Close with comment" button with a red exclamation mark 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

-o add indie songs

399d0b9

-o 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!



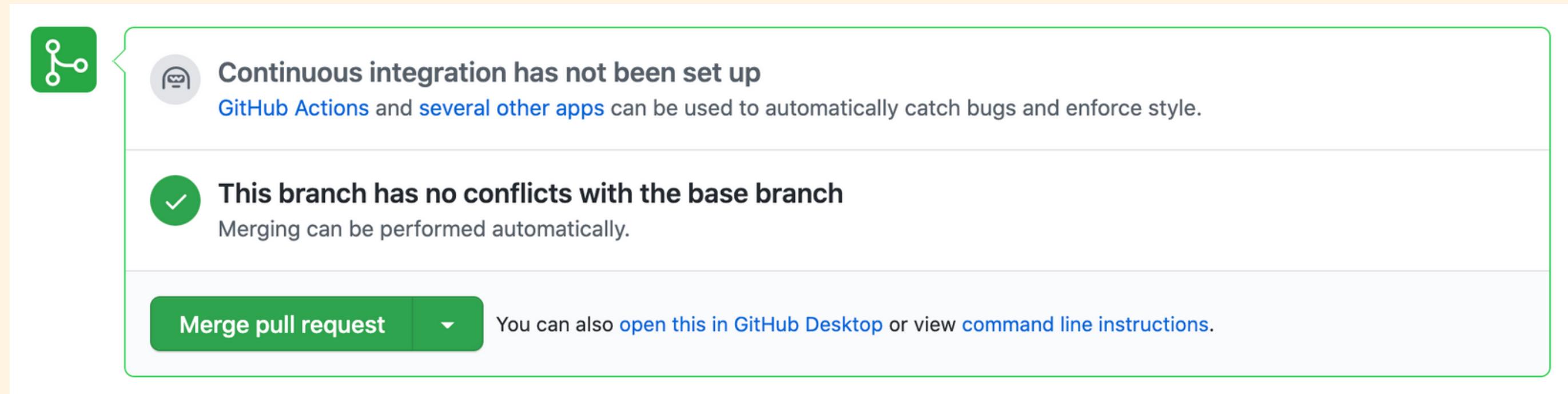
Colt commented 21 seconds ago

Author ...

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

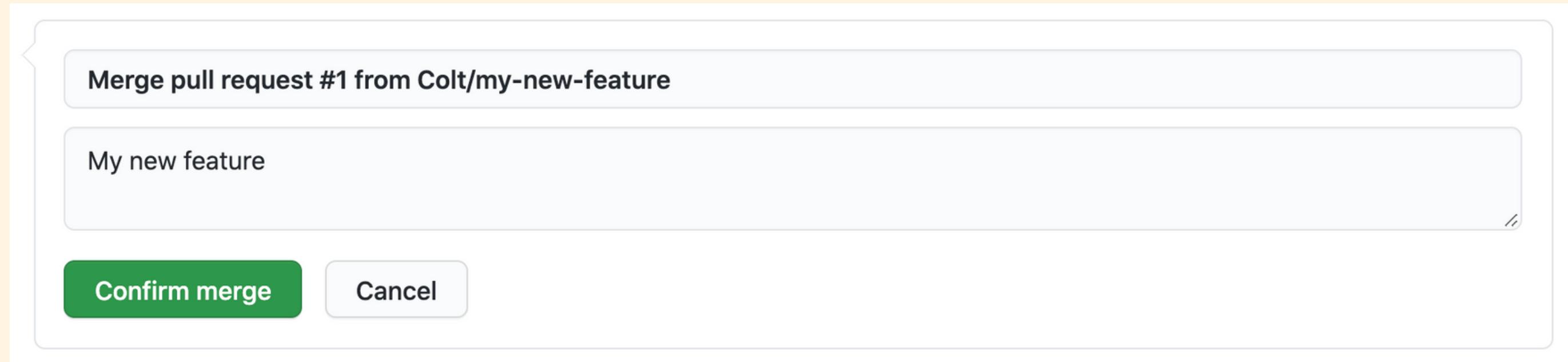
I can respond! We can discuss and give feedback!

My Boss's Github...



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...



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...



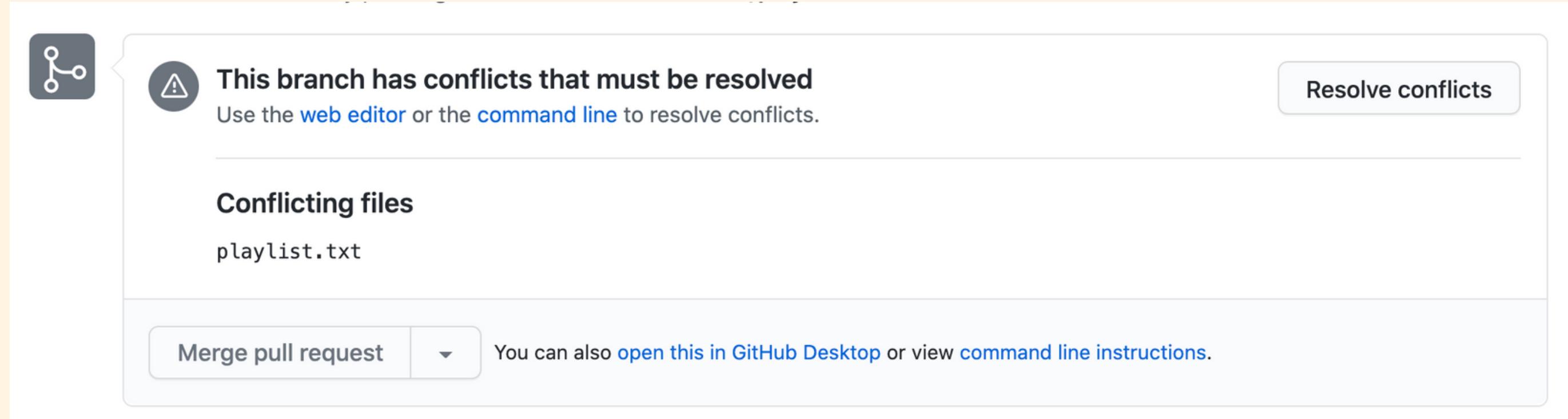
Pull request successfully merged and closed

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...



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](#).



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.

listing them



 Close pull request

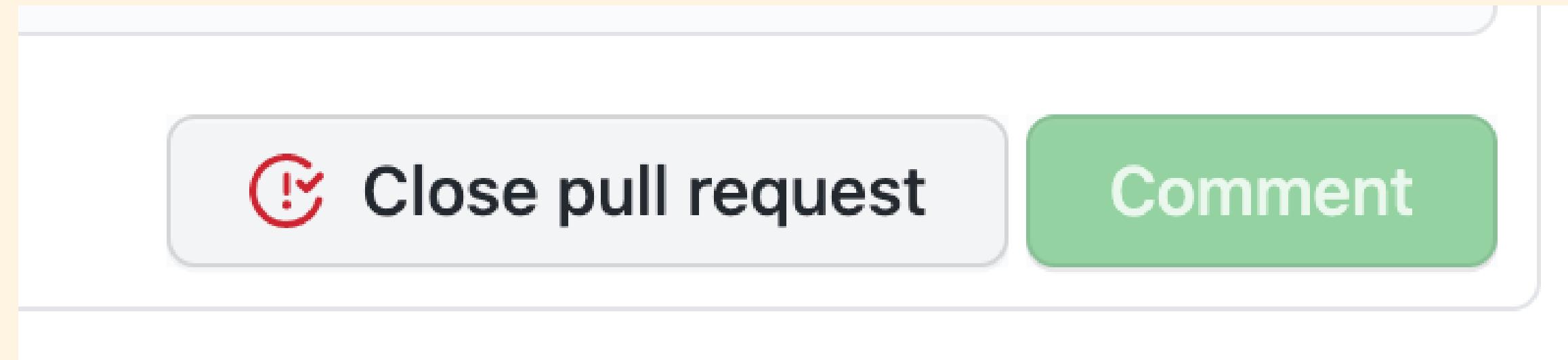
Comment



 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.

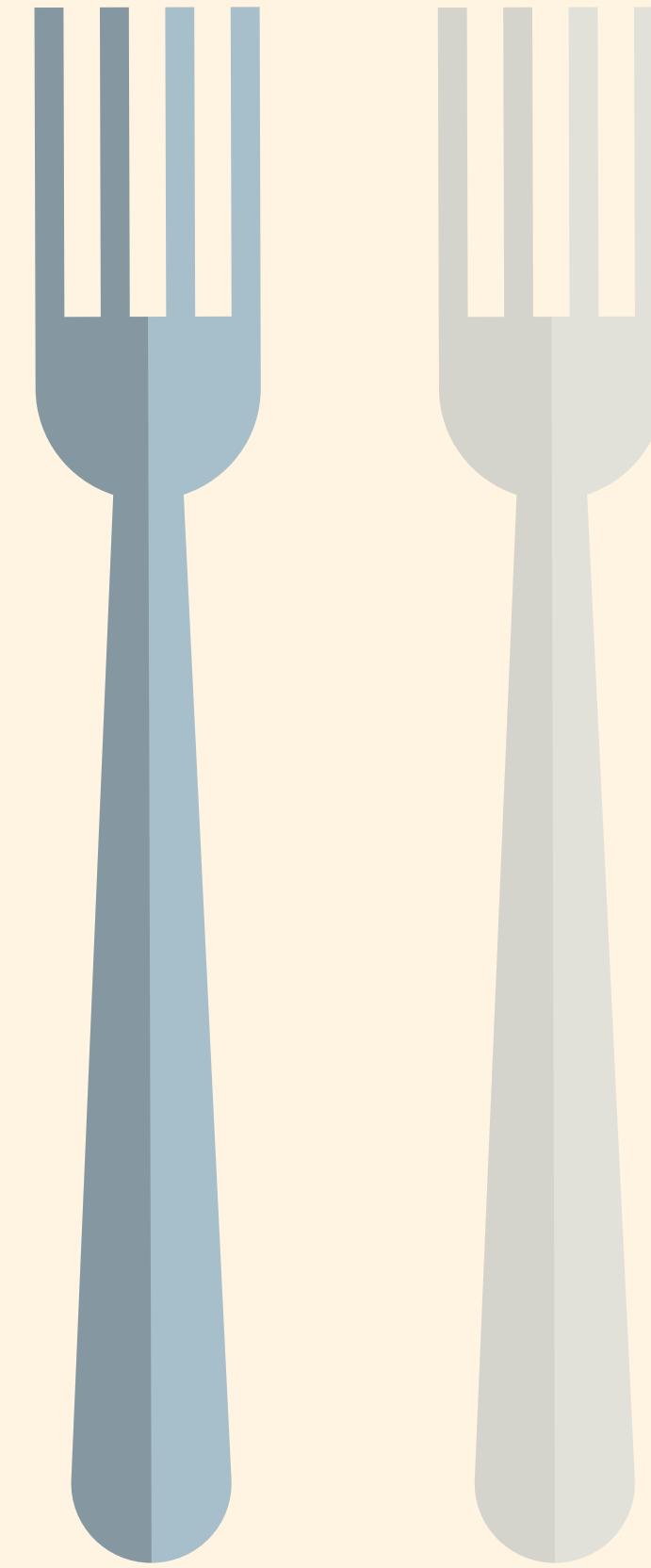




Fork & Clone: Another Workflow

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.

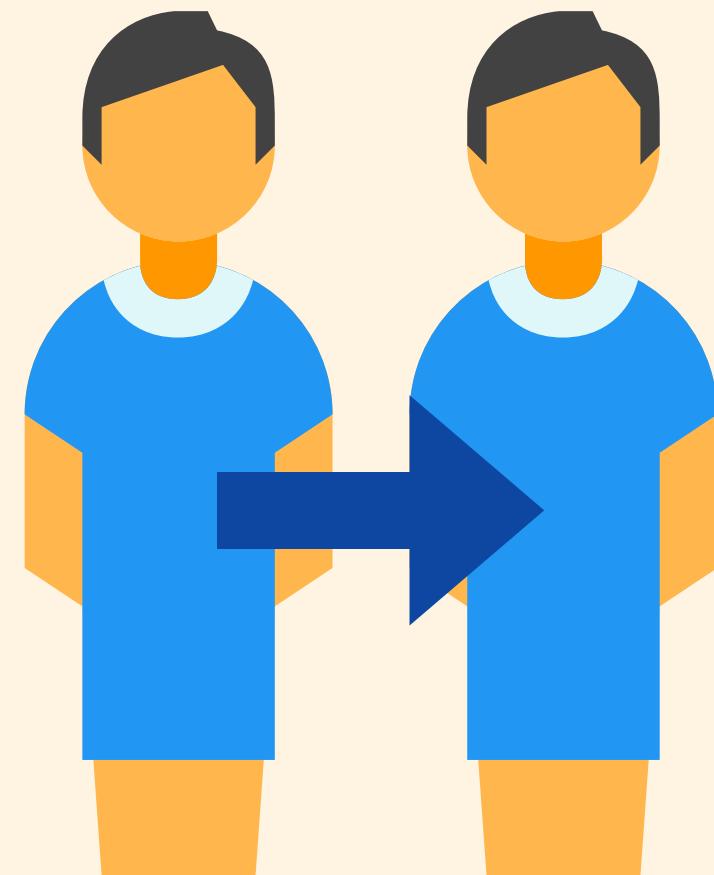


Forking

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
- .deepsource.toml Create .deepsource.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

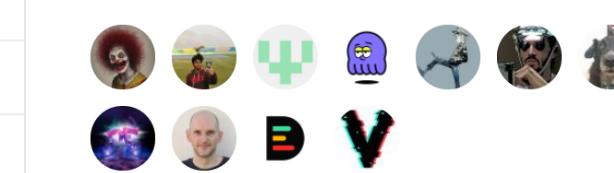
Releases 6

v1.0.0 Latest 6 days ago + 5 releases

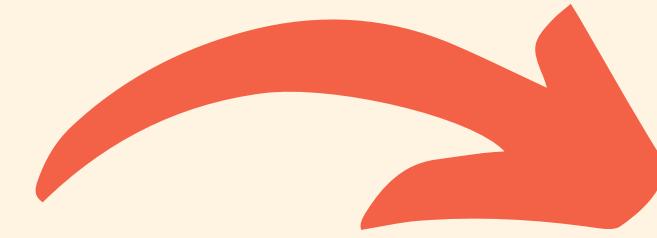
Packages

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 Pull requests Discussions Actions Projects Wiki Security Insights

master 2 branches 6 tags Go to file Add file Code

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

Initial commit 3 months ago

Update README.md 4 days ago

Added logging 13 days ago

Create pull_request_template.md 15 days ago

Bump version in pyproject 18 days ago

Adding more OS tests 15 days ago

Update SECURITY.md 15 days ago

logconfig.ini 13 days ago

pyproject.toml 15 days ago

pytest.ini 15 days ago

Merge pull request #146 from aapatre/develop 6 days ago

Wait for enroll button to be clickable after selecting state/province 10 days ago

Adding unittests 13 days ago

Adding more unittests 23 days ago

Create .deepsource.toml 16 days ago

fixed deepsource ignore 16 days ago

Adding restyled configuration last month

fixed typo 6 days ago

Create CODE_OF_CONDUCT.md 15 days ago

Create CONTRIBUTING.md 15 days ago

Initial commit 3 months ago

Update README.md 4 days ago

Update SECURITY.md 15 days ago

Added logging 13 days ago

Create pull_request_template.md 15 days ago

Bump version in pyproject 18 days ago

Adding more OS tests 15 days ago



 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 aapatre#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	
.deepsource.toml	Create .deepsource.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](#)

≡

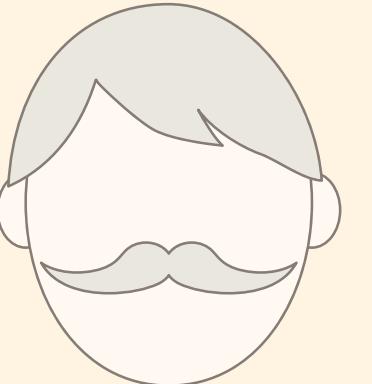
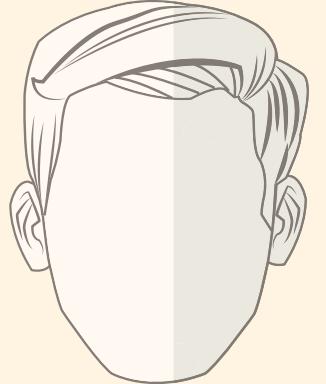
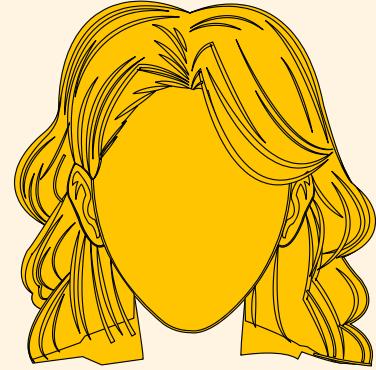
Now What?

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

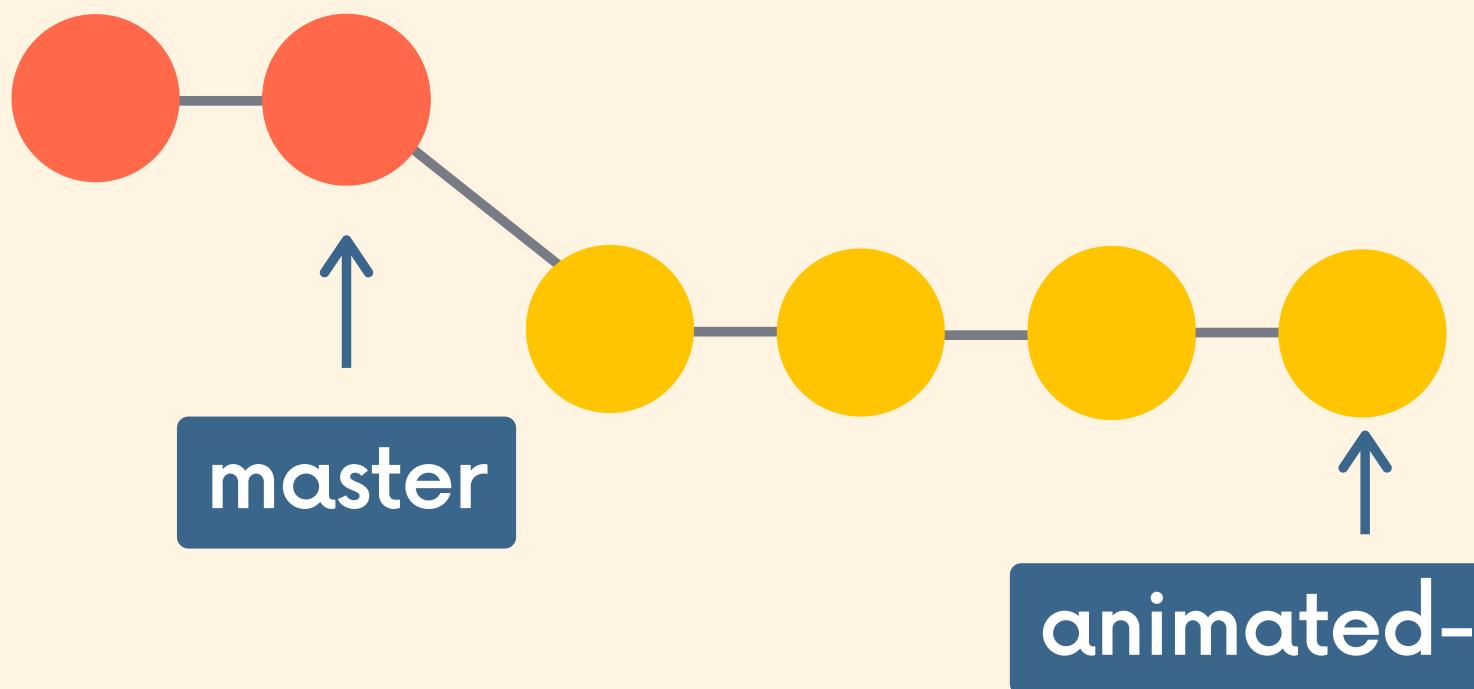
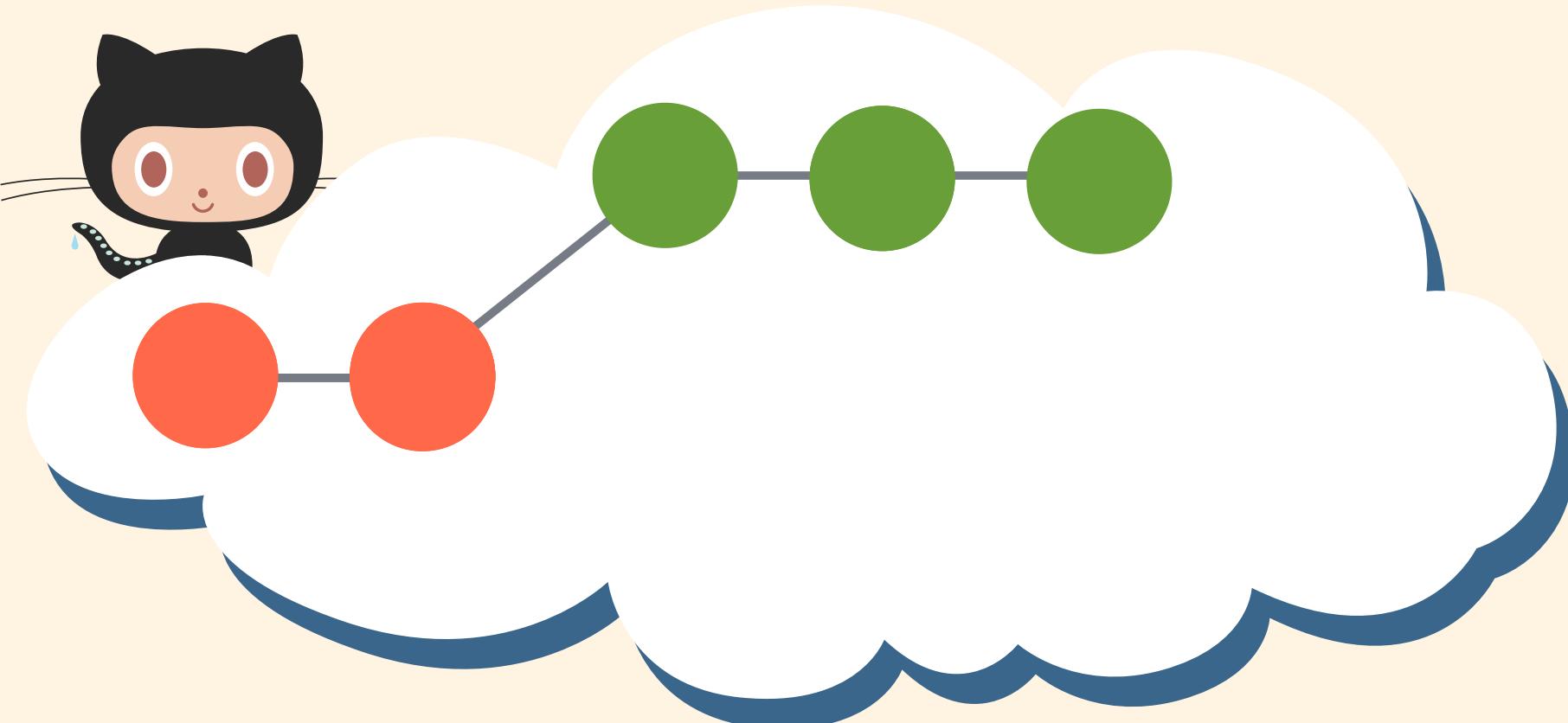
I can clone my fork and make changes, add features, and break things without fear of disturbing the original repository.

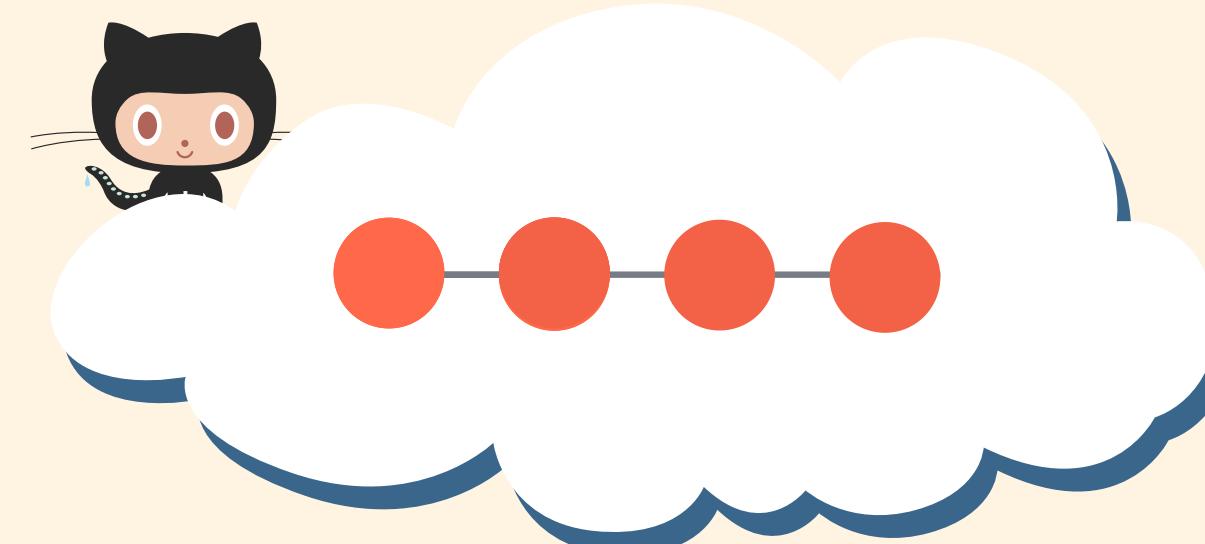
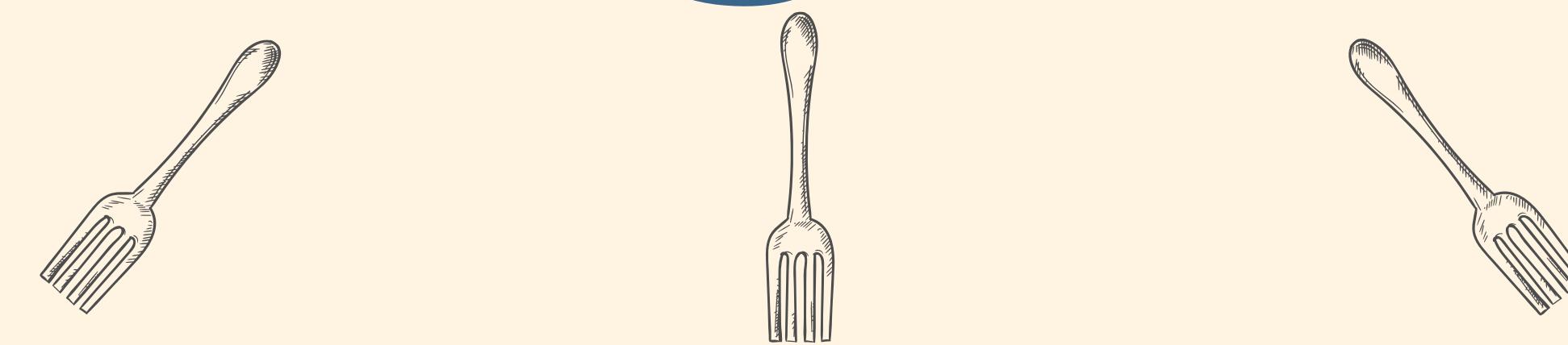
If I do want to share my work, I can make a pull request from my fork to the original repo.



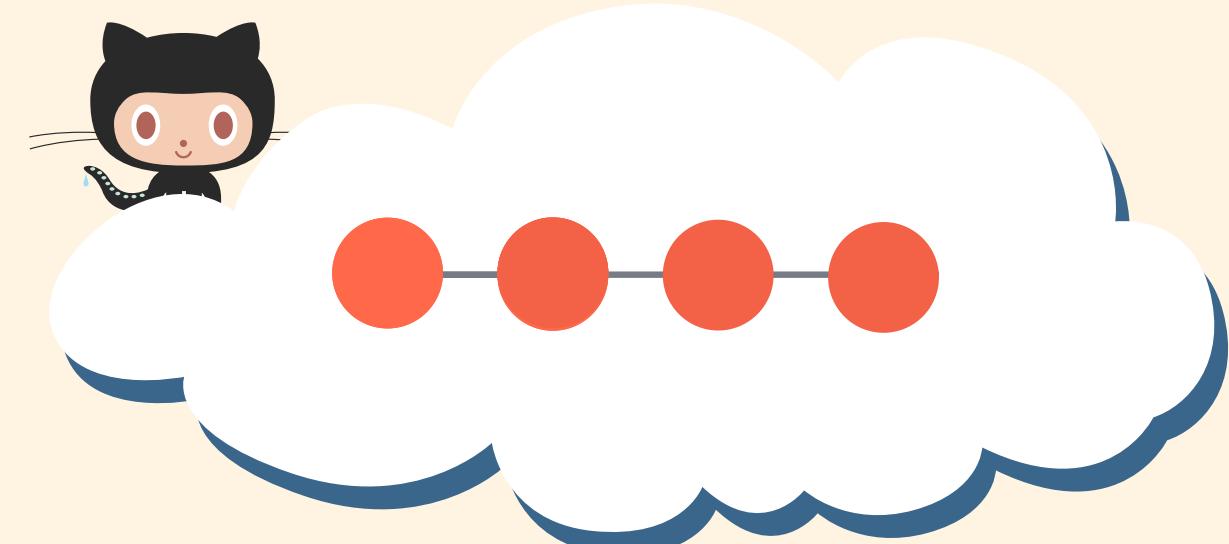


Pamela is hard at work on her own new feature. Just like everyone else, she's working on a separate feature branch rather than master.

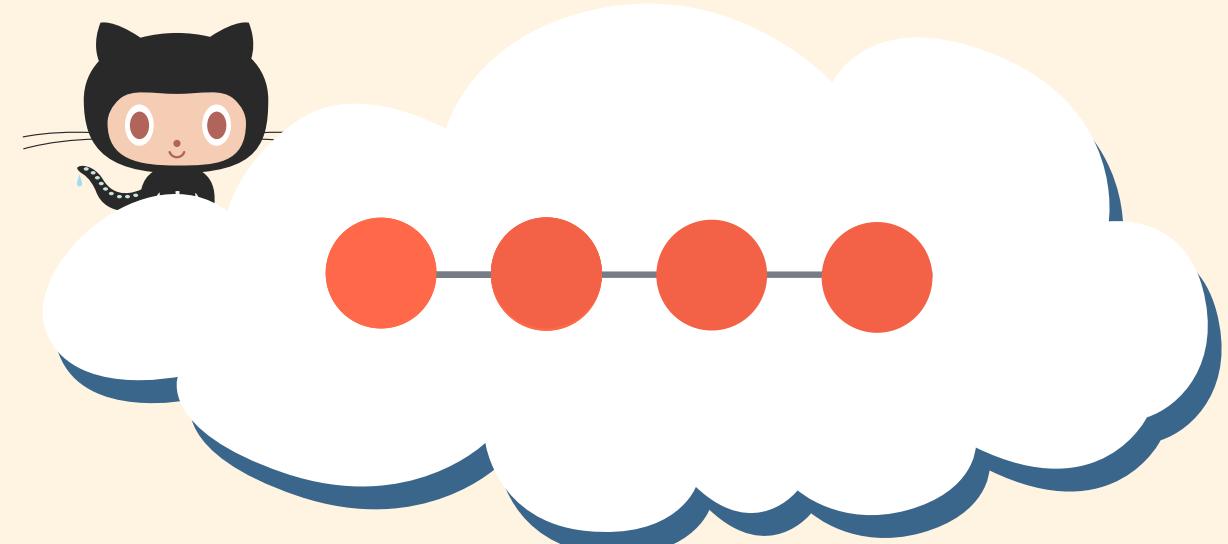




My Fork

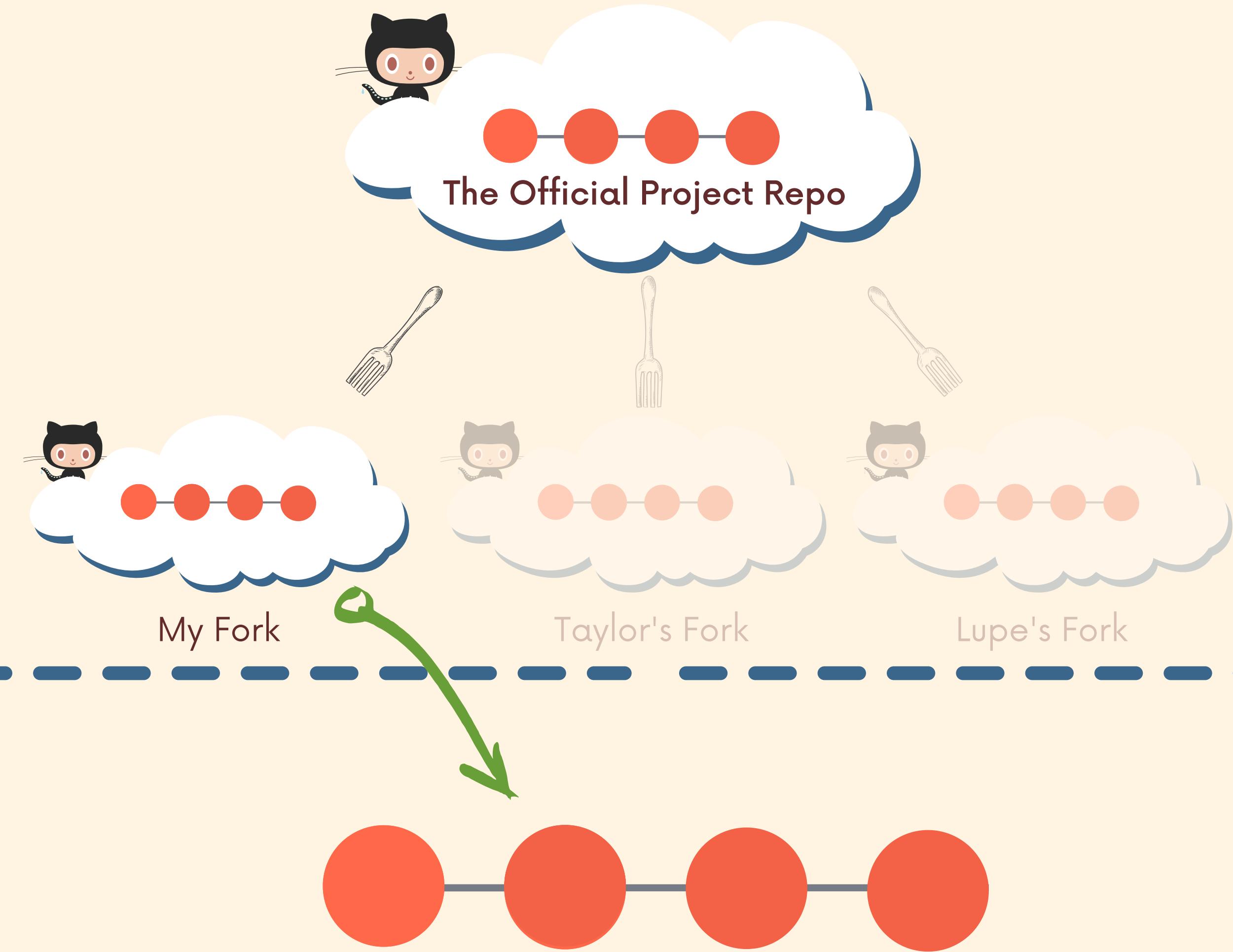


Taylor's Fork

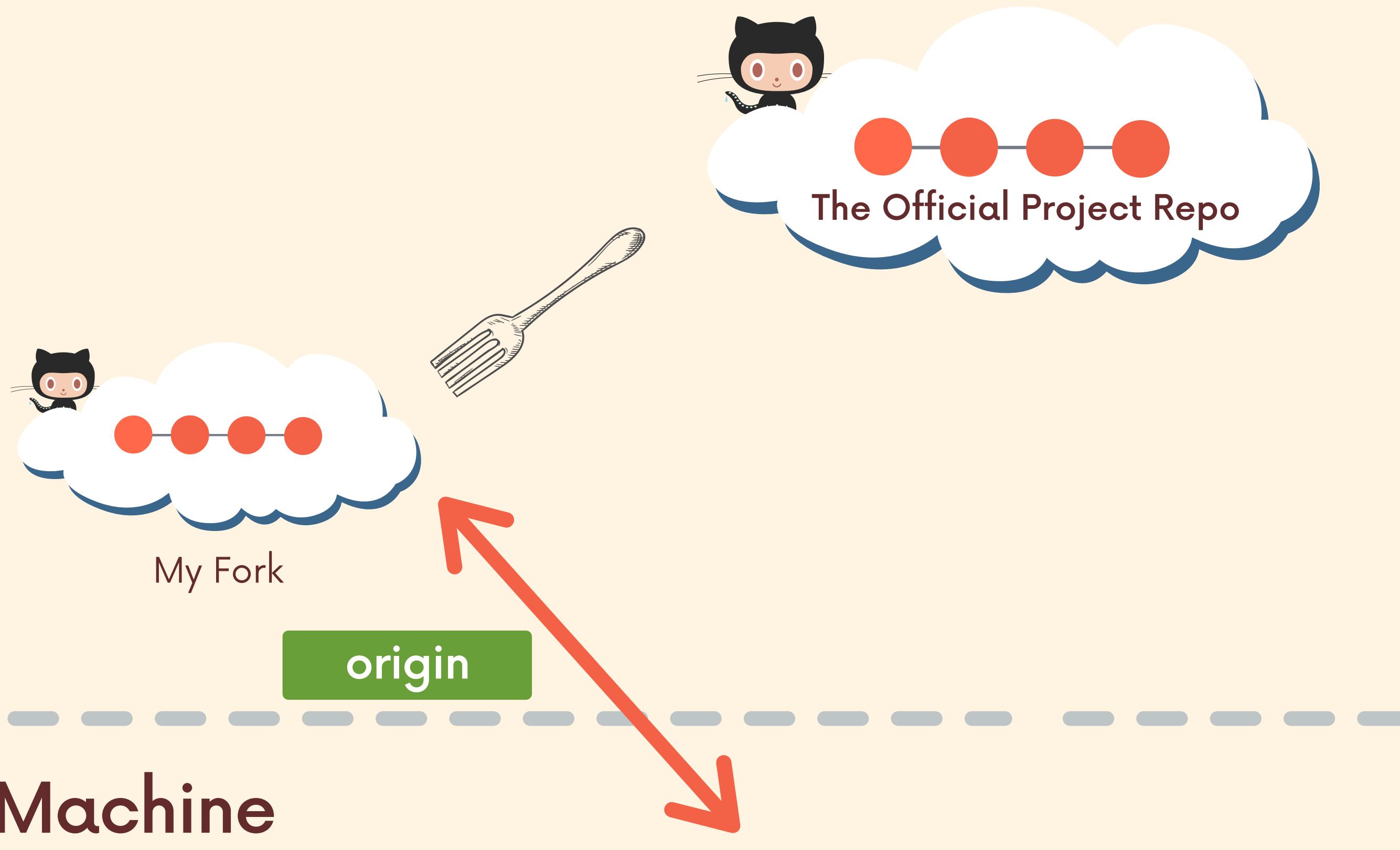


Lupe's Fork

Github



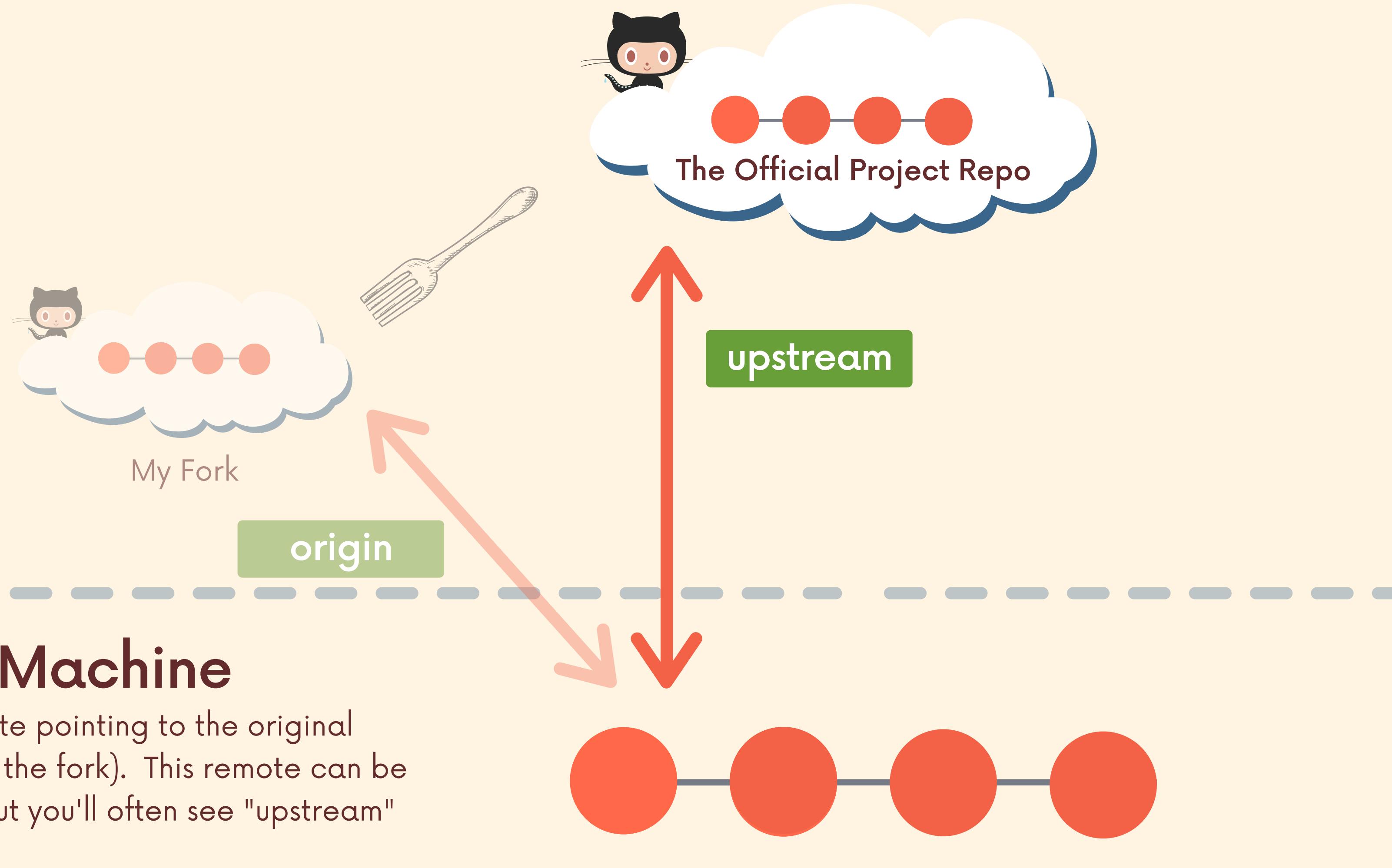
Github



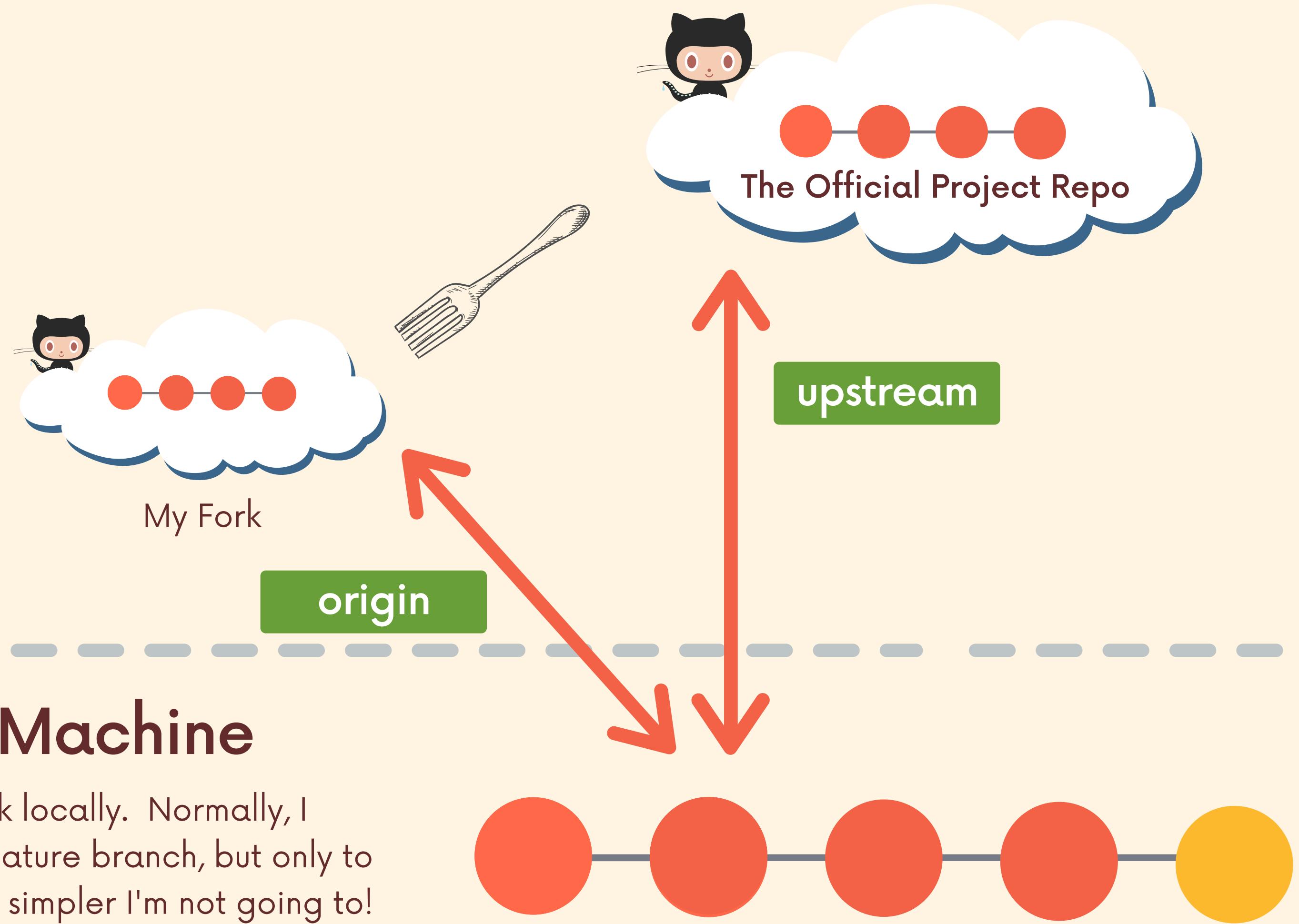
My Local Machine

When we clone a repo, Git automatically adds a remote called **origin** that points to our forked repo on Github.

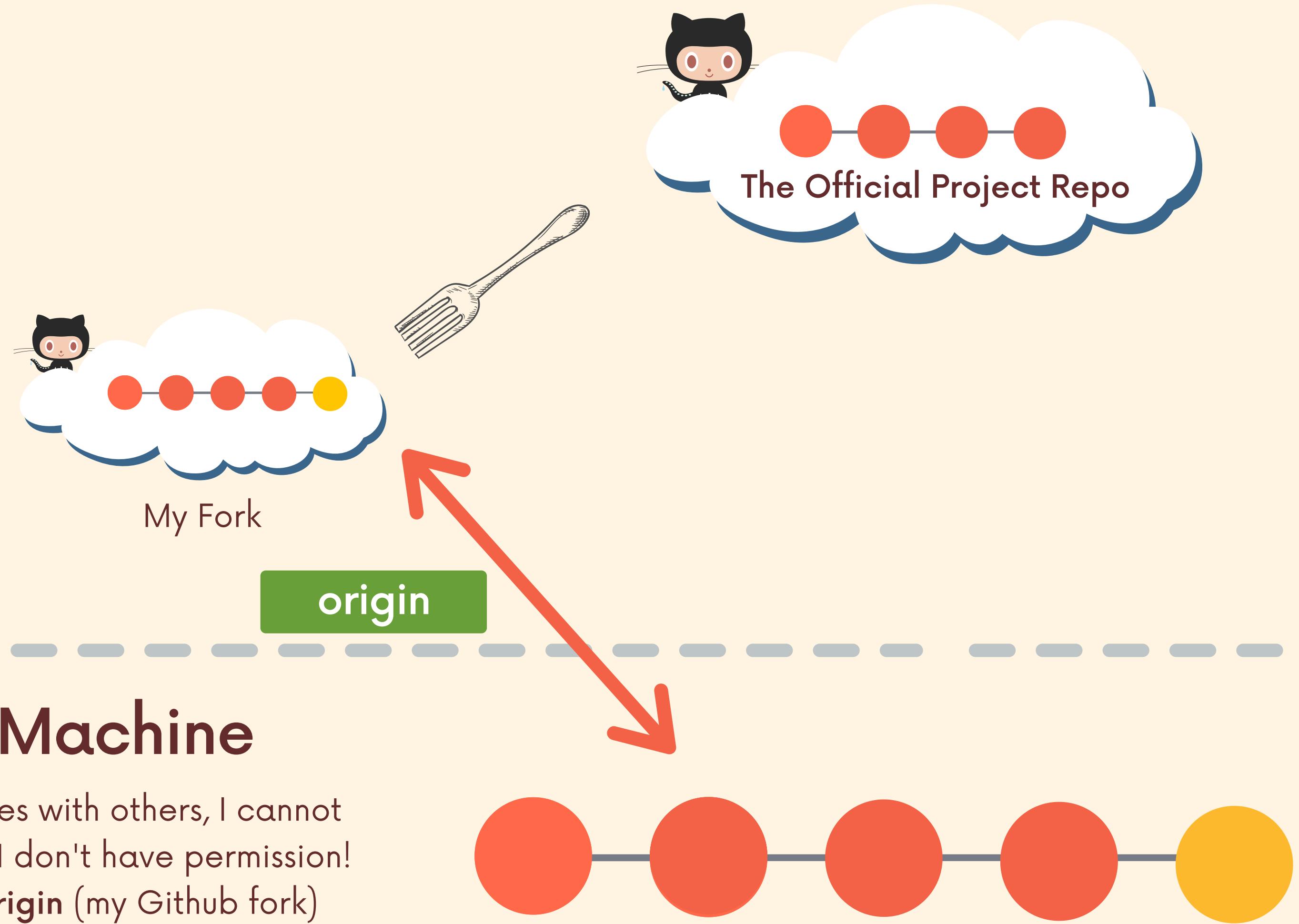
Github



Github

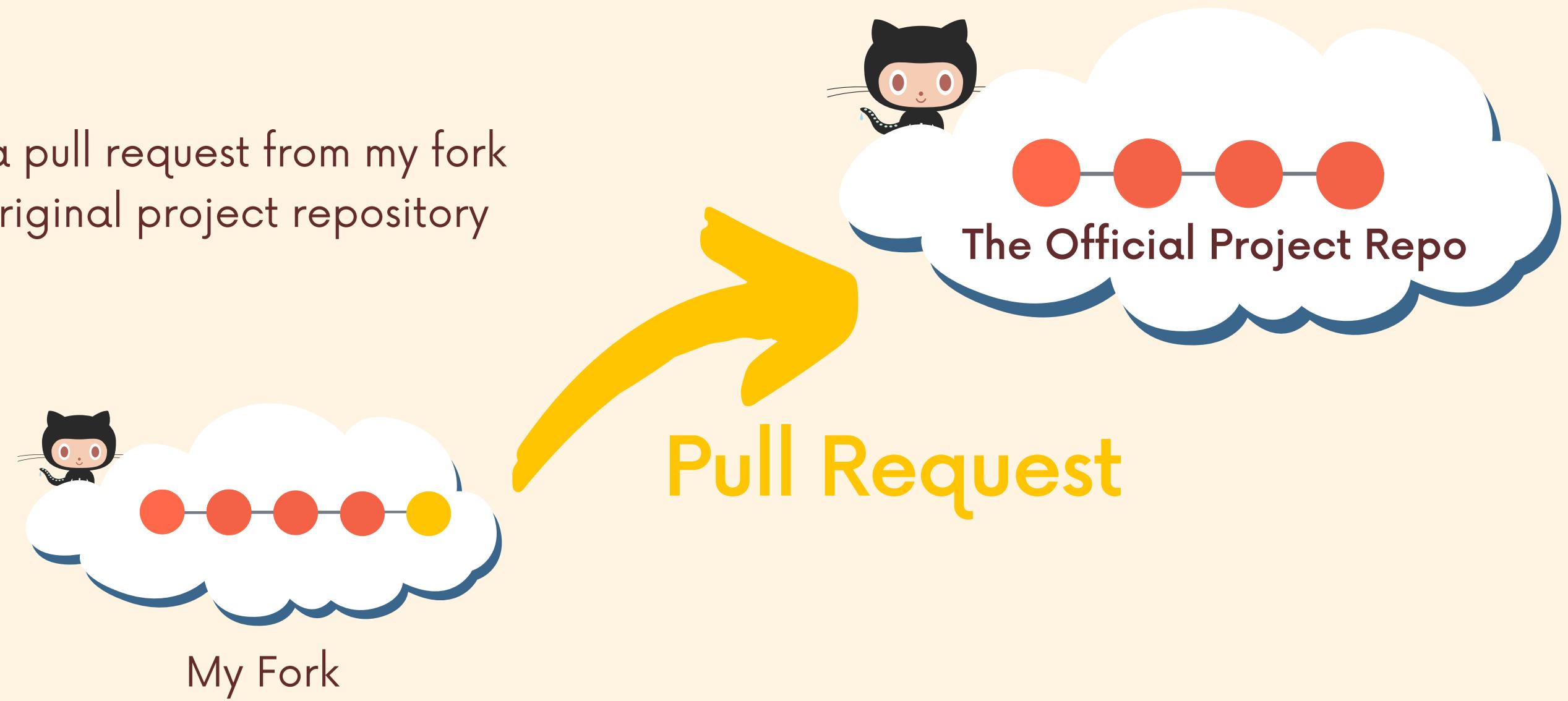


Github

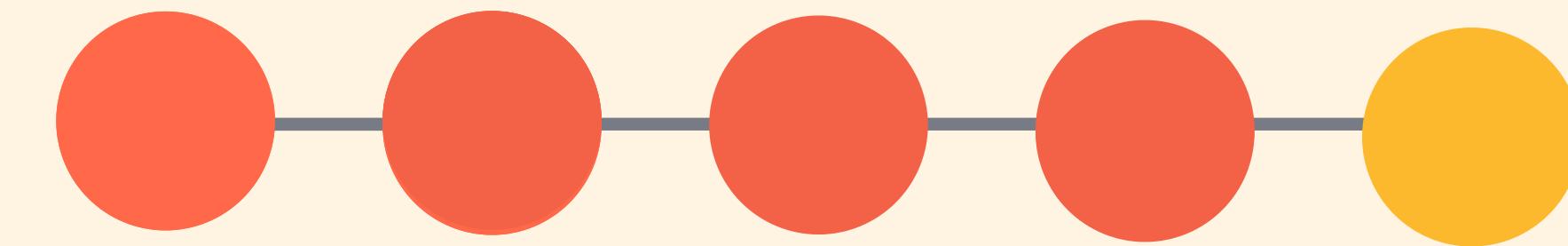


Github

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

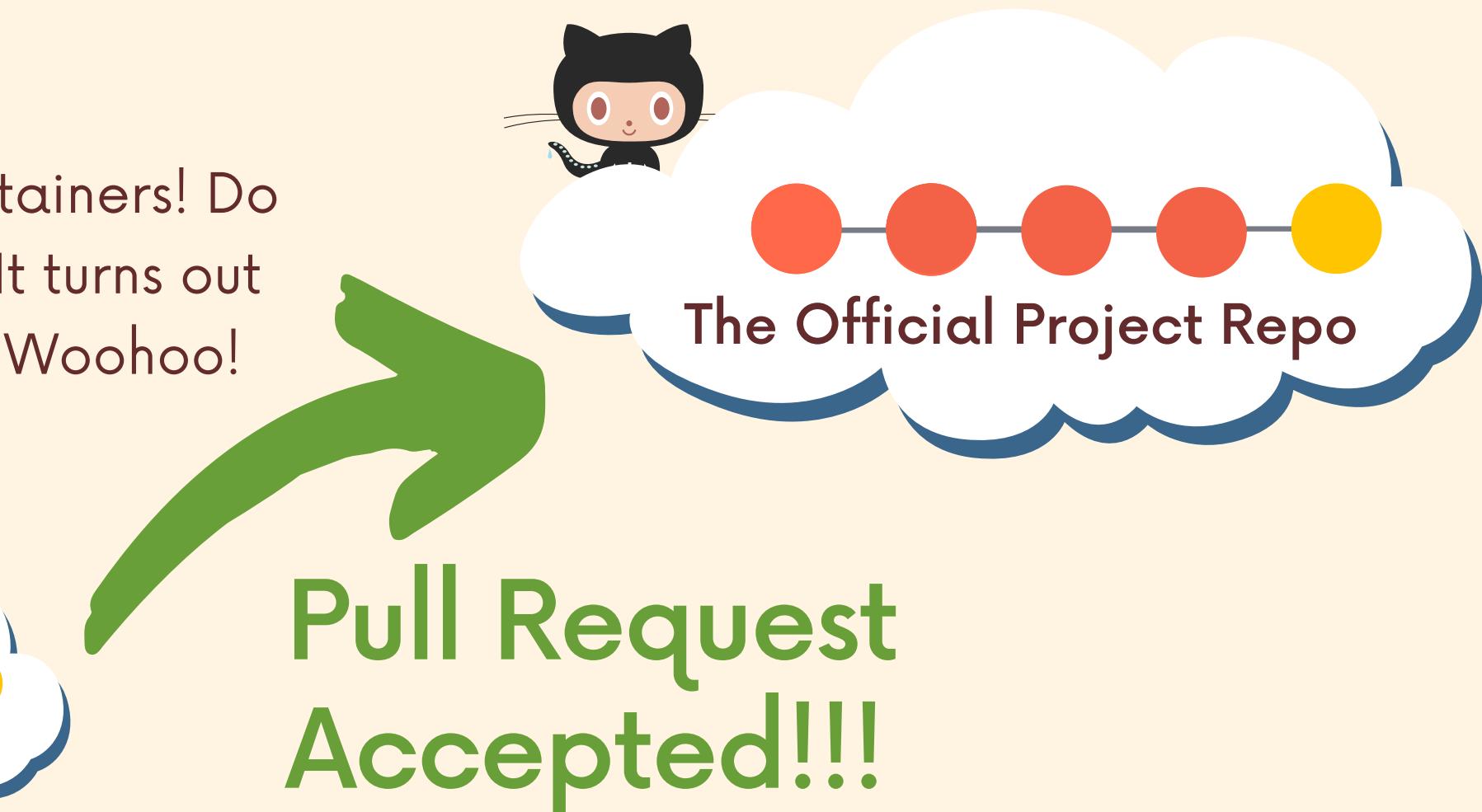
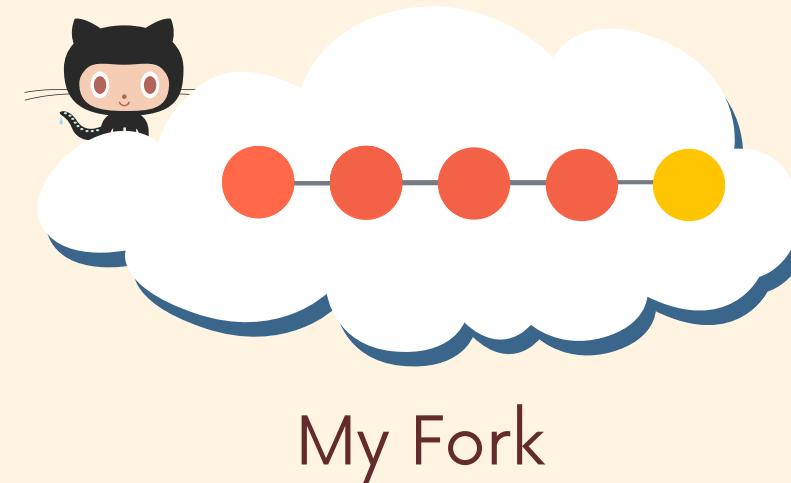


My Local Machine

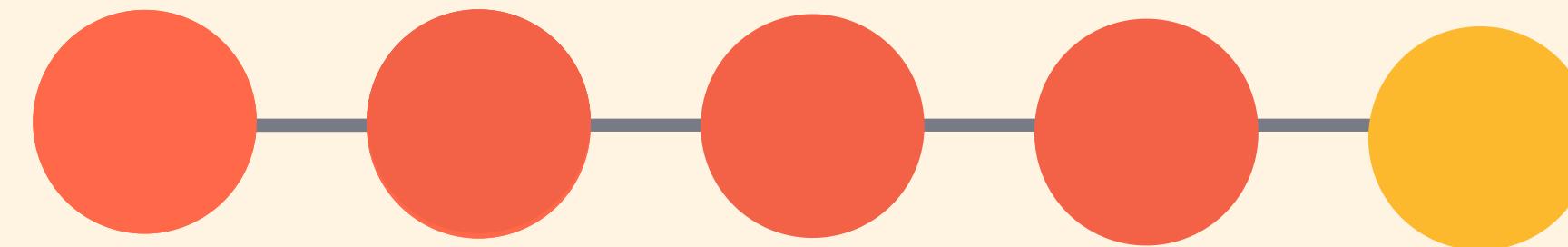


Github

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!

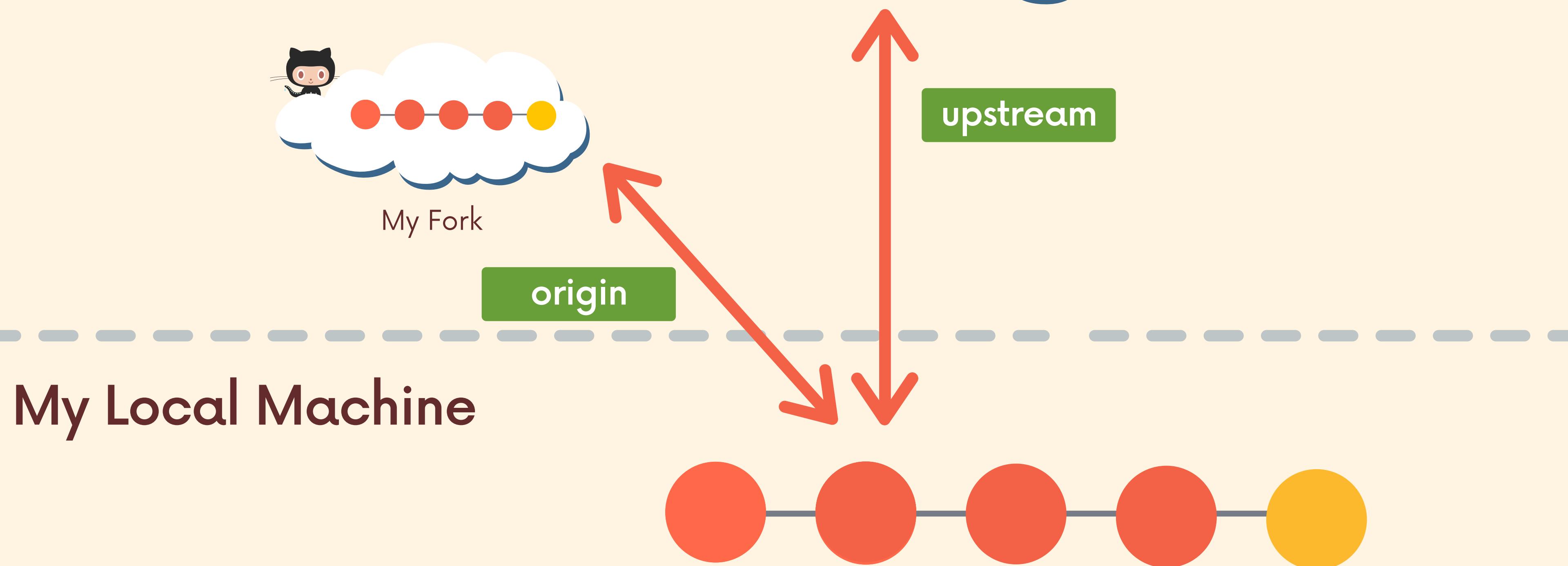


My Local Machine

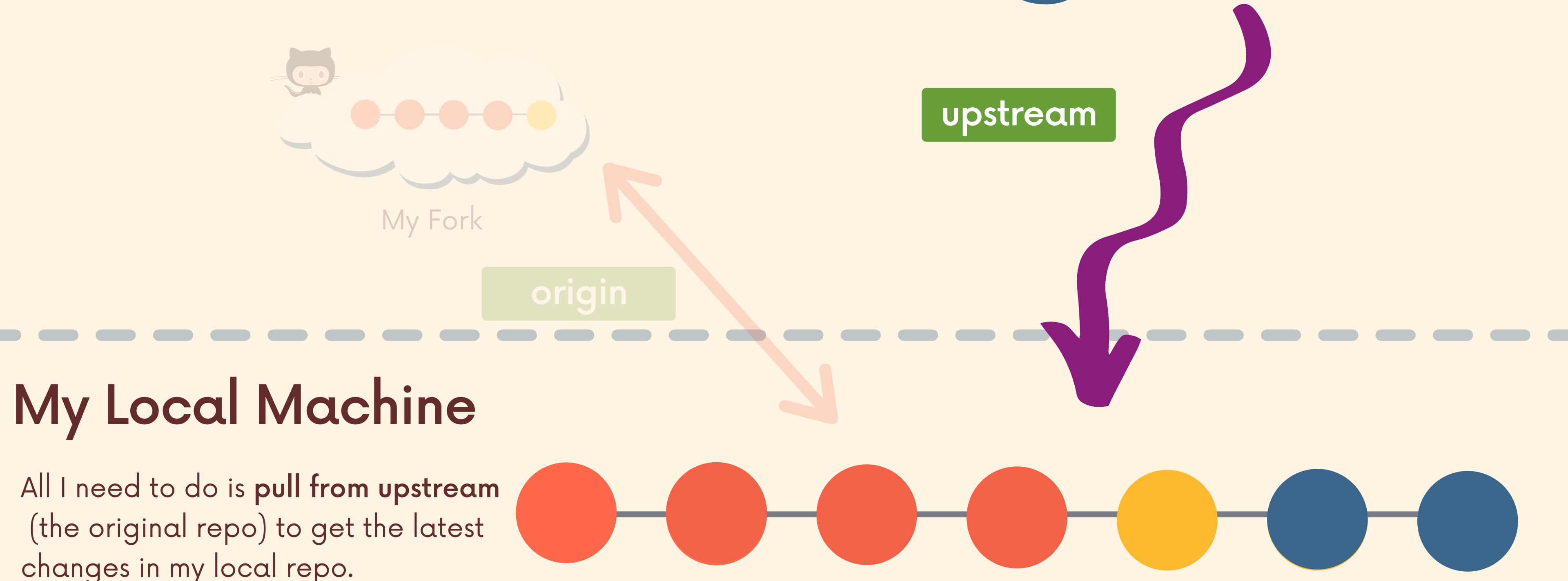


Github

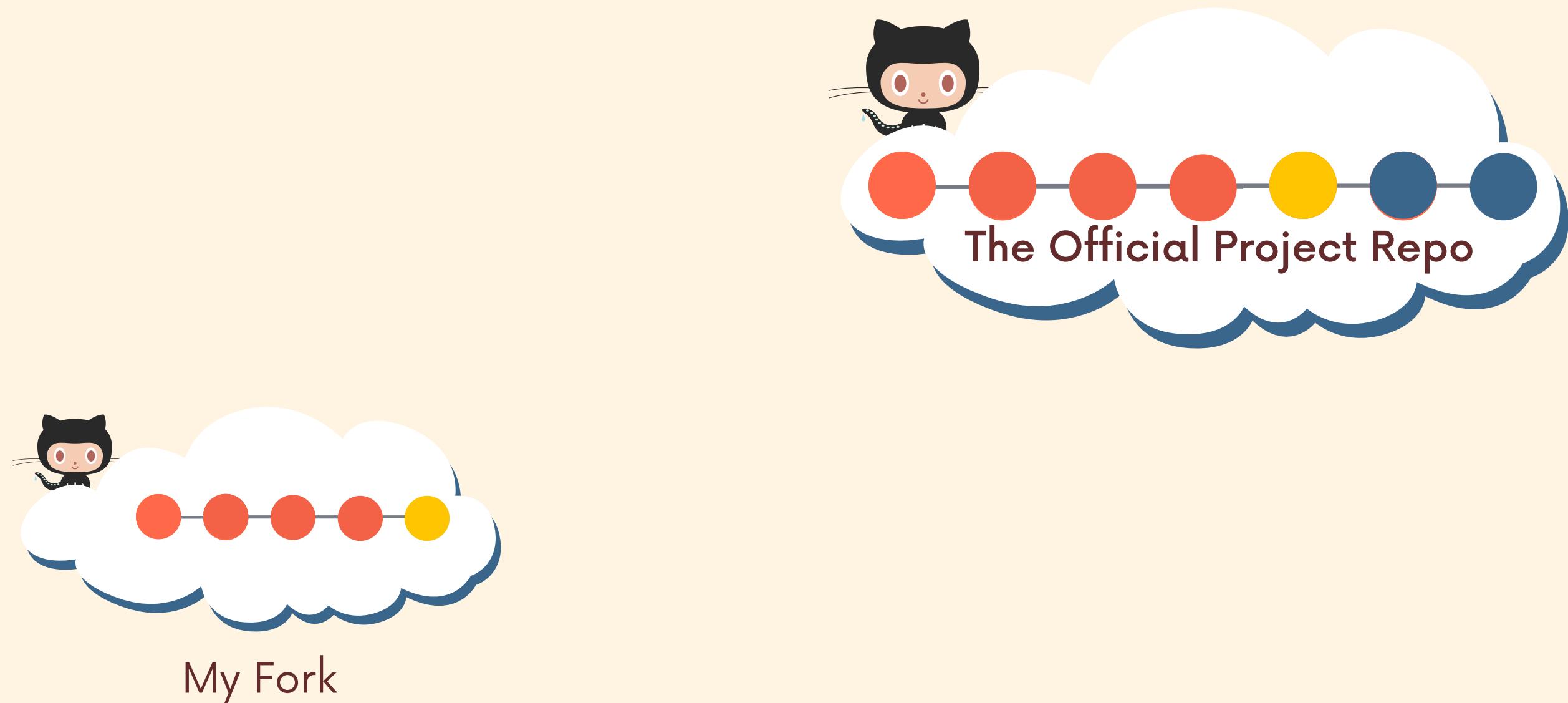
The next day, I get back to work. The official project repo now contains work done by other collaborators. I don't have their new work on my machine! I'm behind!



Github

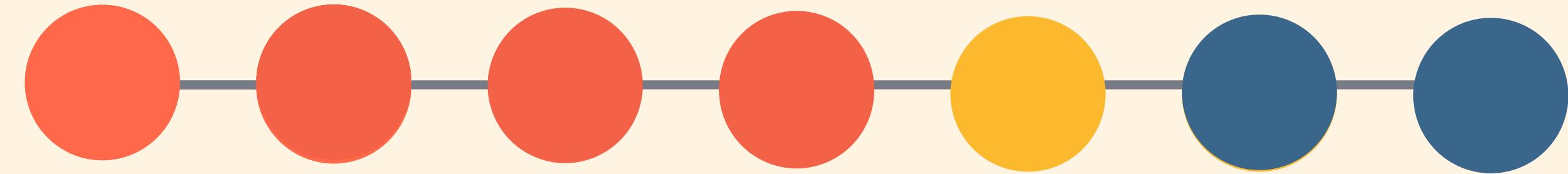


Github

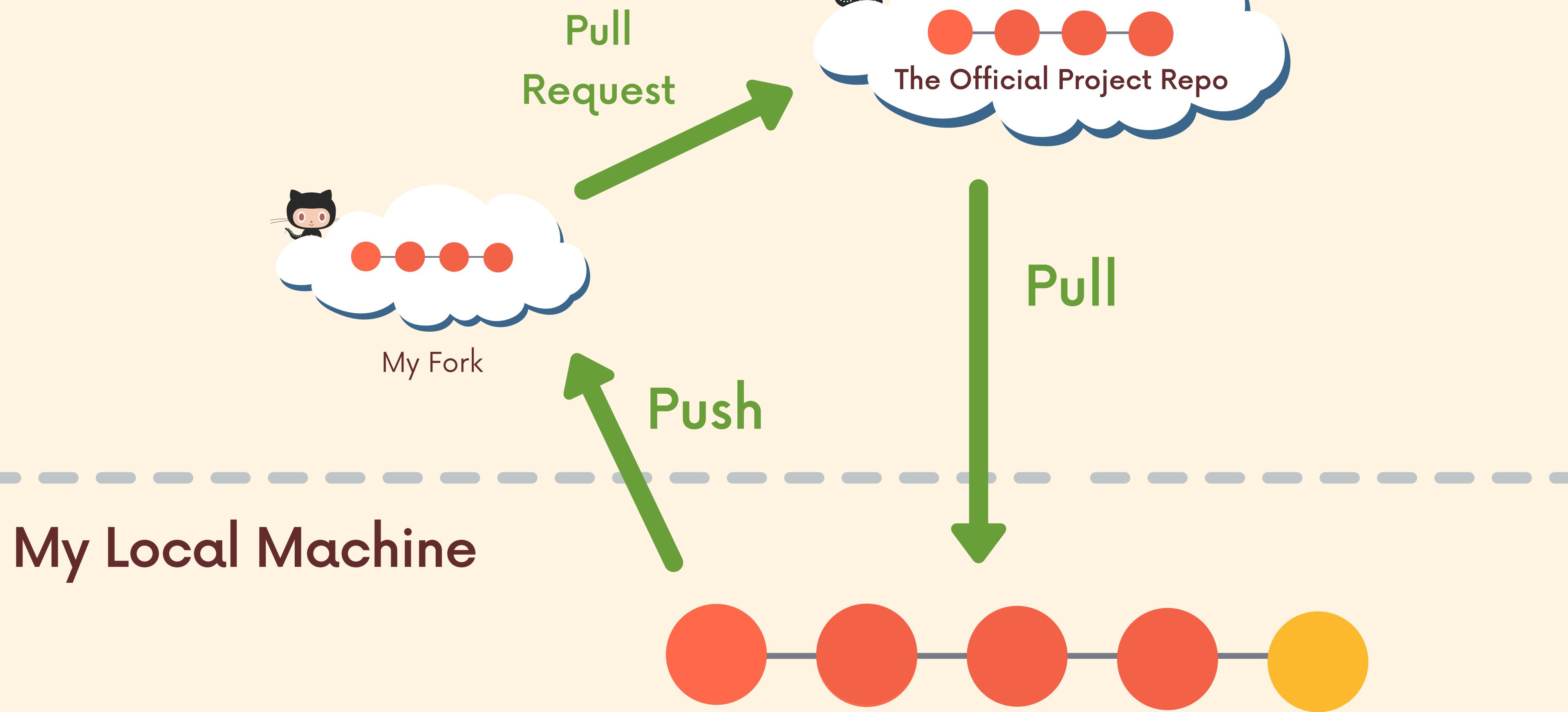


My Local Machine

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.I fork the original project repo on Github
- 2.I clone my fork to my local machine
- 3.I add a remote pointing to the original project repo.
This remote is often named upstream.
- 4.I make changes and add/commit on a feature
branch on my local machine
- 5.I push up my new feature branch to my forked repo
(usually called origin)
- 6.I 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

