NeuroNex

Data Joint

Workshop 2018
Day 2

Sponsored by

NeuroNex Innovation Award
#1707359

Presented by
Edgar Y. Walker
April 20, 2018

DataJoint

Session 4:
Pipeline Design Exercises

Session 4 Goals

1. Practice designing a data pipeline from scratch based on project requirements
2. Practice the “design discussions” in a team
3. Recognize common design patterns

DataJoint

Session 5: Extending existing pipelines

Session 5 Goals

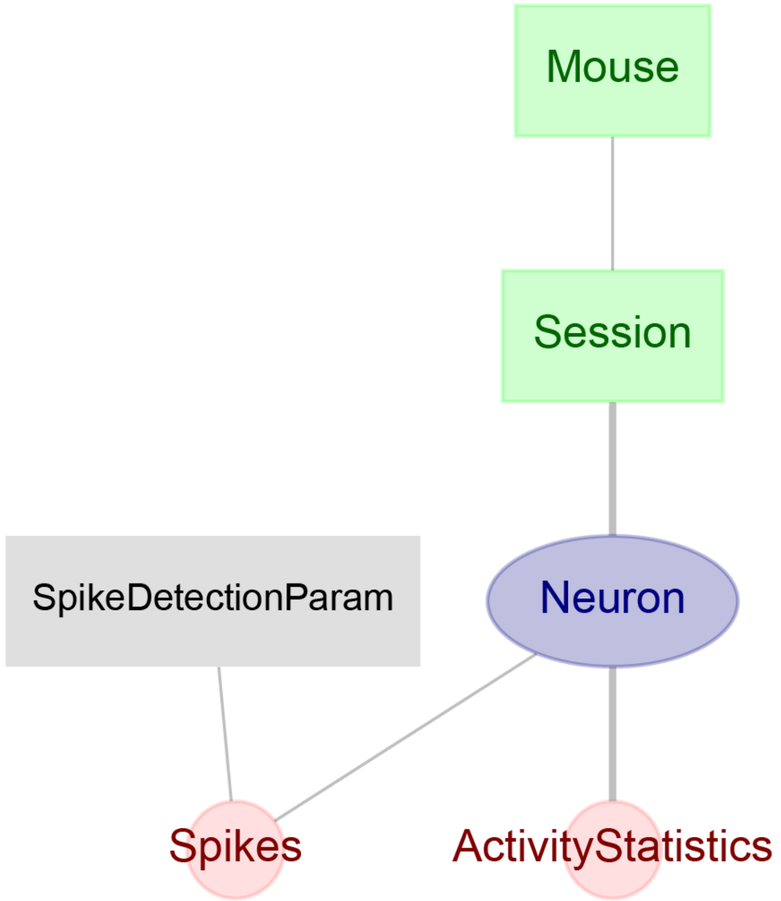
1. Learn to explore an existing data pipeline
2. Extend an existing data pipeline with your own computation

DataJoint

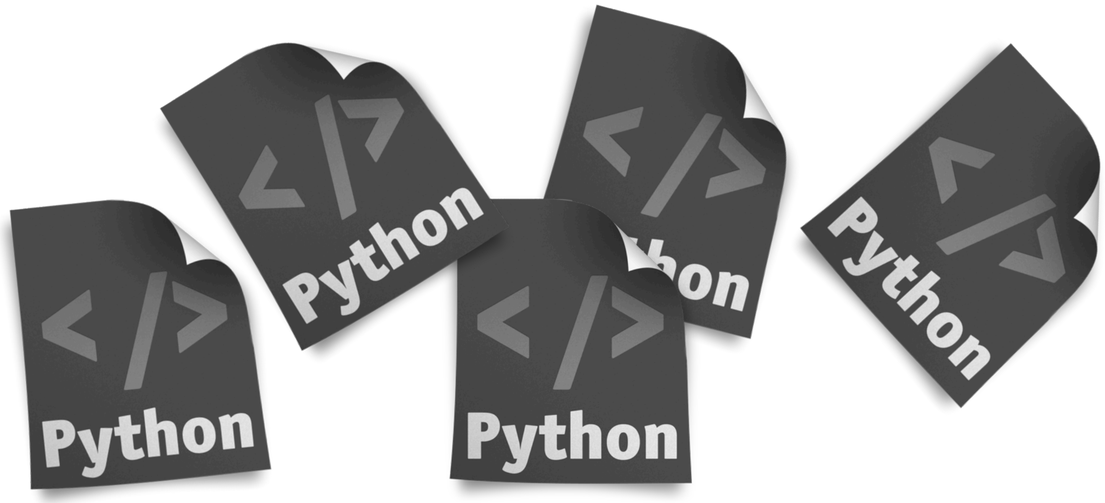
Session 6:
Best practices in scientific computations and data sharing

Computation results depend on your code

- What an Imported/Computed table gets populated with depends on your code! (e.g. `make` function)
- If you change your code midway, your table can end up containing results from multiple versions of your code!
- If you change your code, you should delete old entries and re-populate
- You need a good code versioning practice!



Keeping track of your code is more critical than ever!!

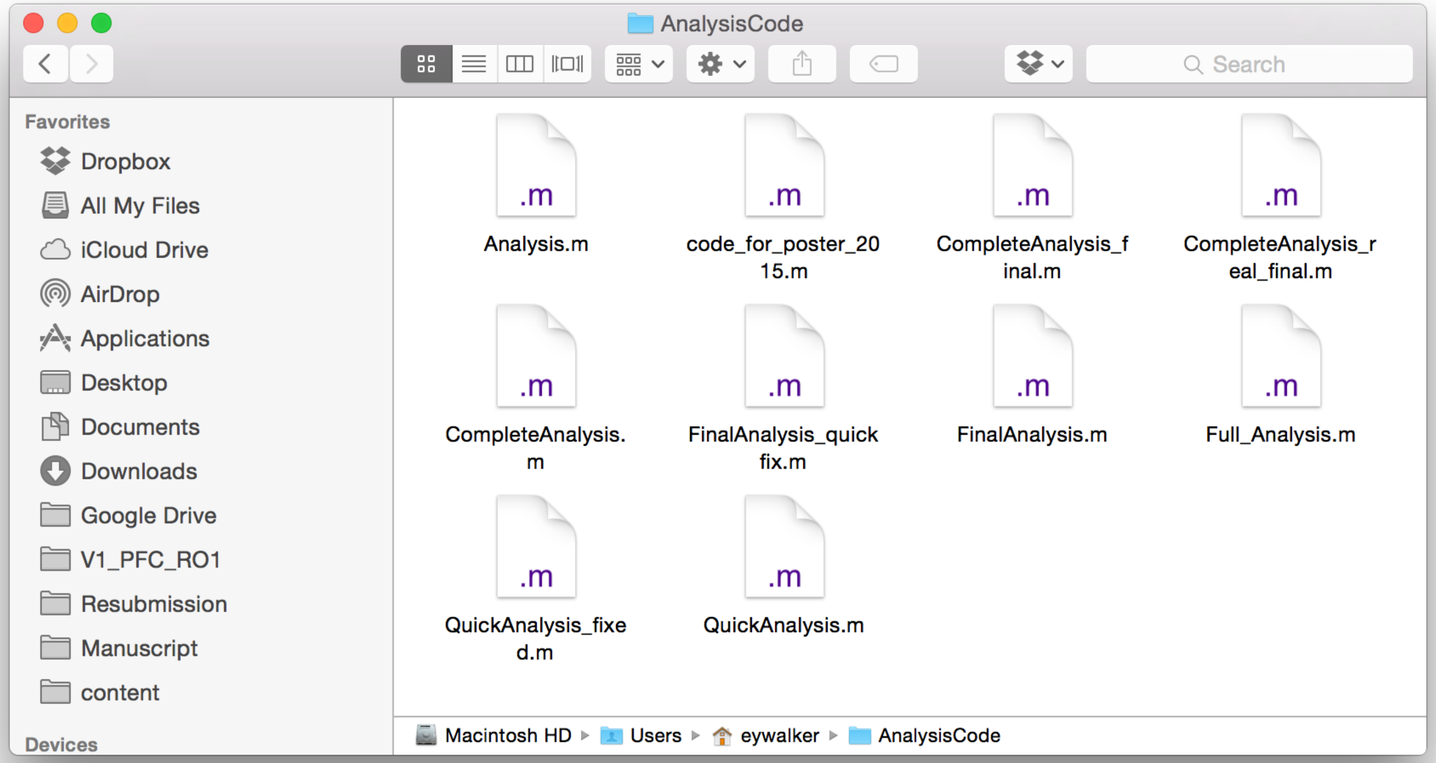


What's wrong with 'keeping a copy' approach?

- Keeping multiple copies of the same file could work for a very short term and a small project
- …but, it can very quickly get out of control!
- Copied file approach also does not tell you clear history of changes

Common but not-so-good approaches

- Multiple versions of the same file in a folder



Introducing you to…

What's Git?

- Formal definition:
  - Git is a distributed **version control system**
- Practical definition:
  - Git is a piece of software the let's you **keep track of changes you made to a project** (e.g. experiment code) with an ability to easily revert back to previous versions

You should use Git if you…

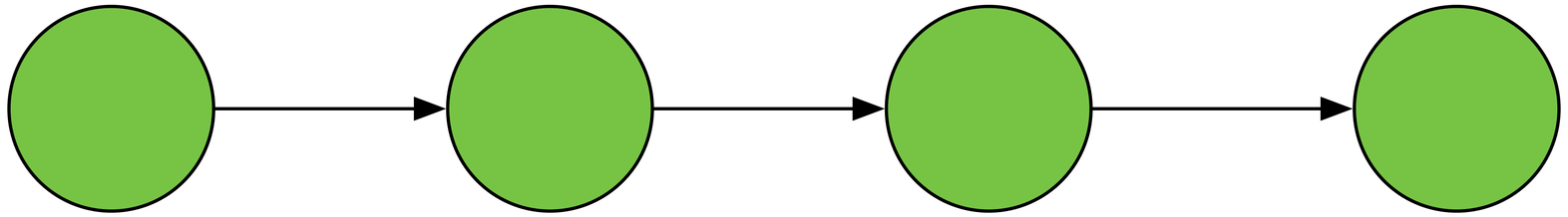
- ever made a change to a file that you wish you could undo
- want to “try out” changes without causing complete catastrophe
- want to keep track of different versions of files
- want to know when the changes were introduced
- collaborate with other people on editing files
- edit files on multiple computers
- wish you could go back in time

How to think about Git

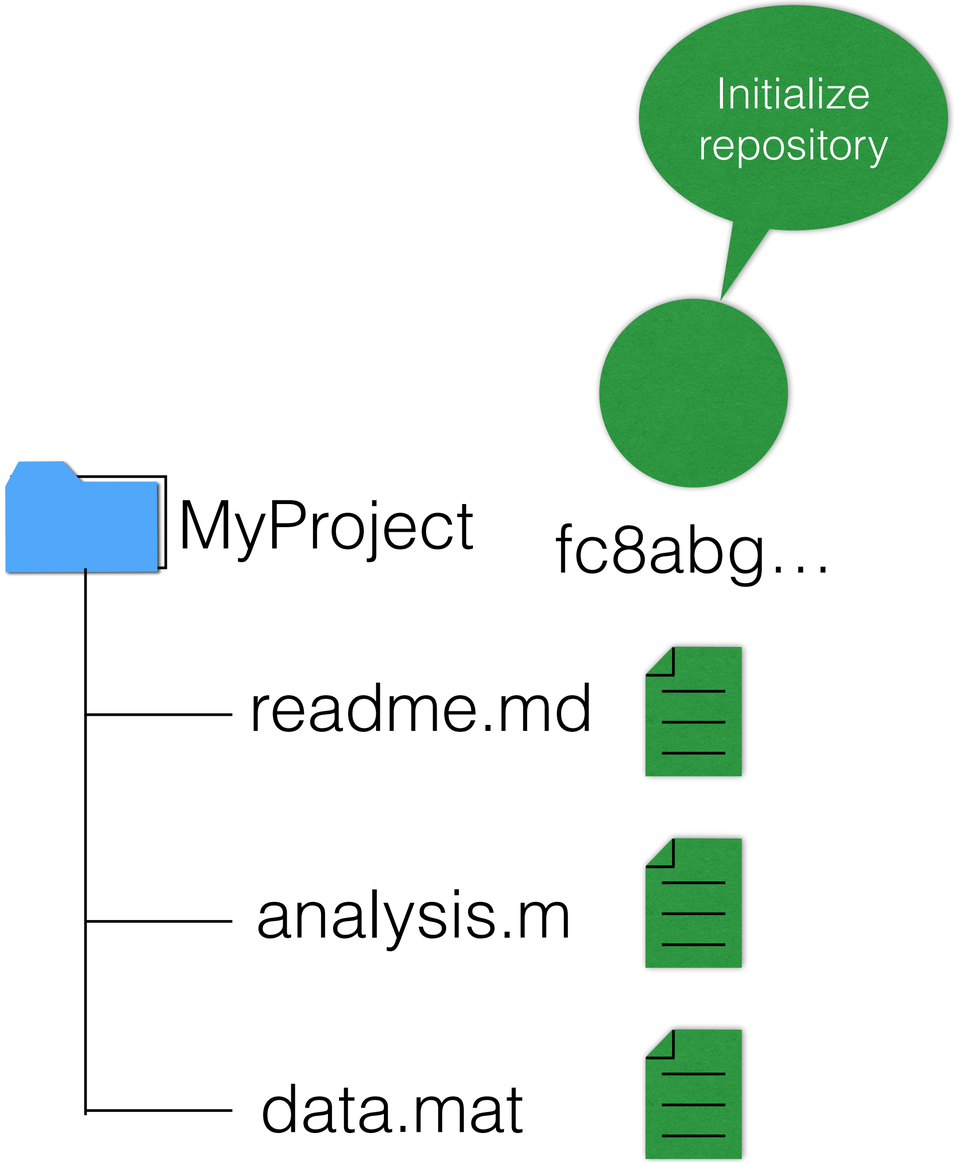
- Git is your…
  - File back up system
  - Change logging system
  - File/code sharing system

How does Git work?

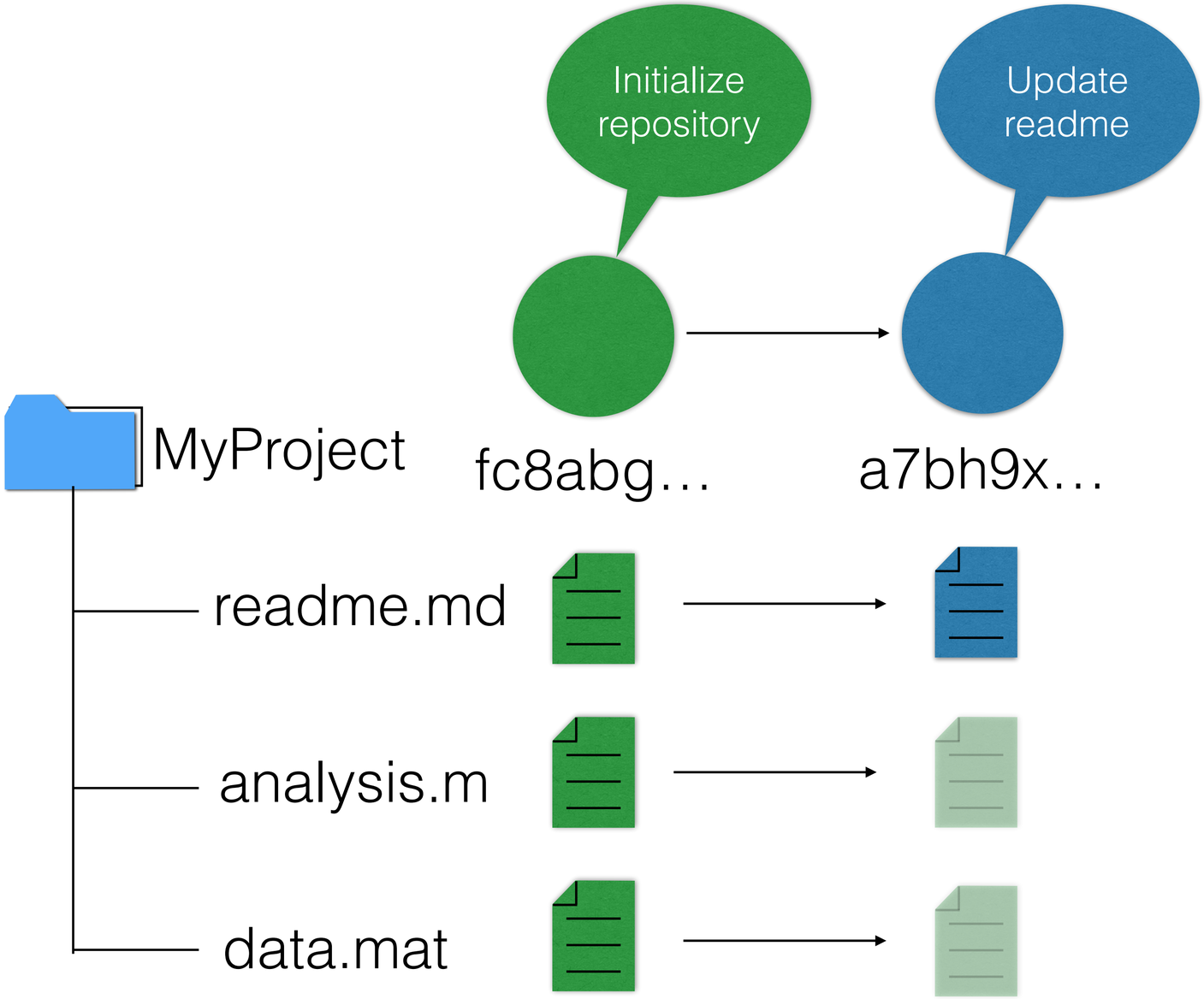
- works at the level of a folder or a project **“repository”**
- Keeps a **local** history of **commits**
- **Commits** are a collection of changes to files inside the **repository**



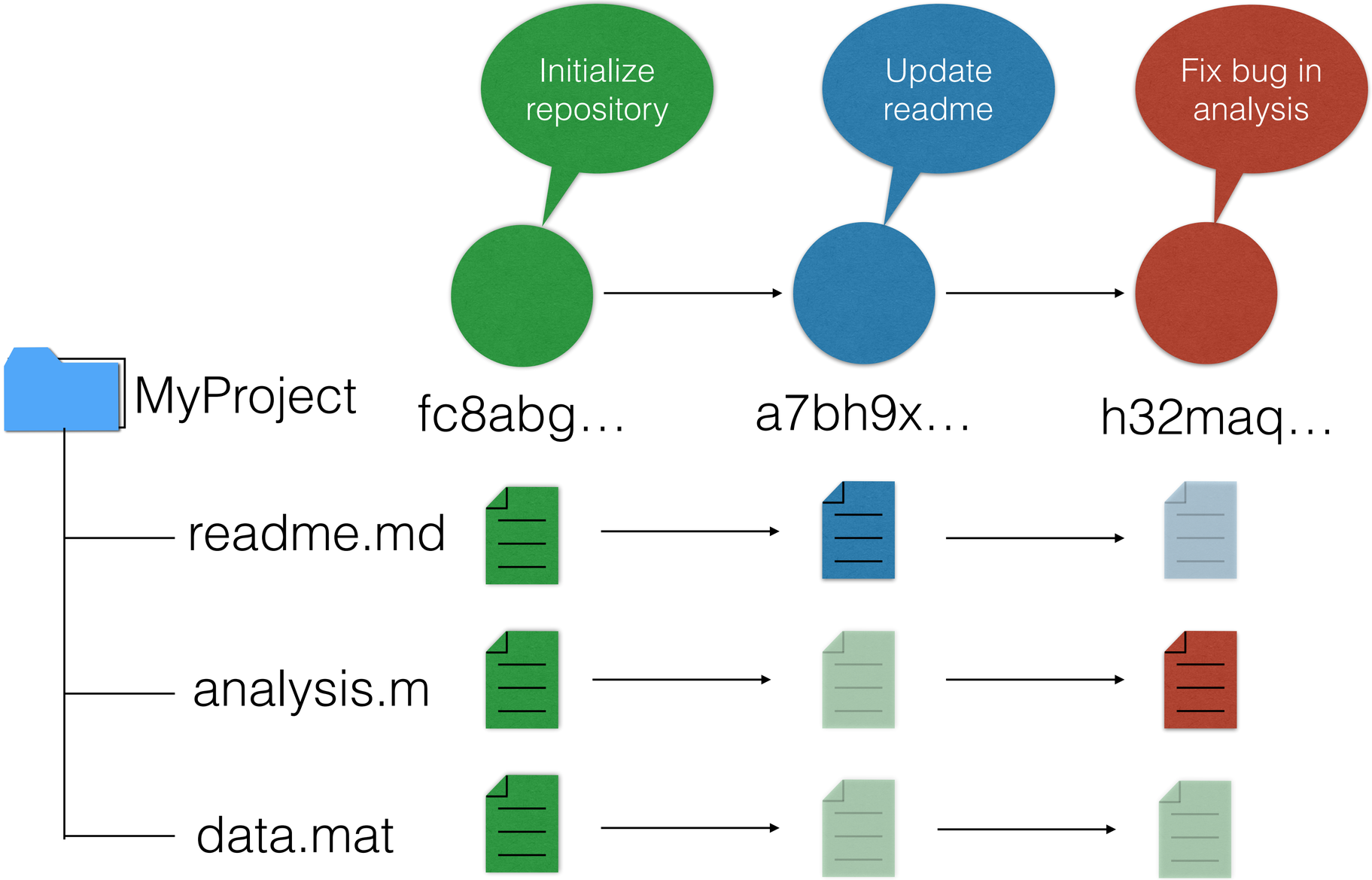
How does Git work?



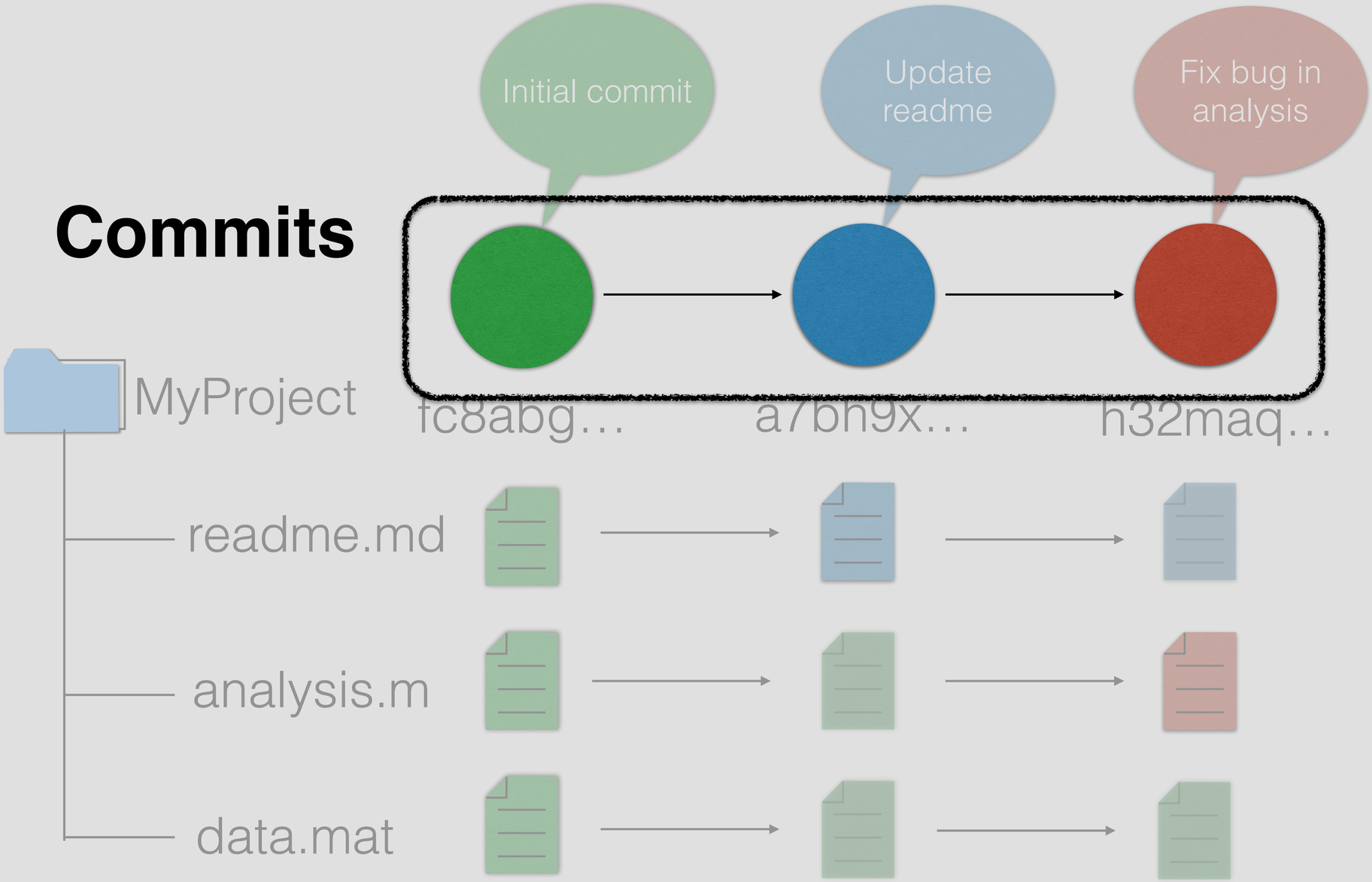
How does Git work?

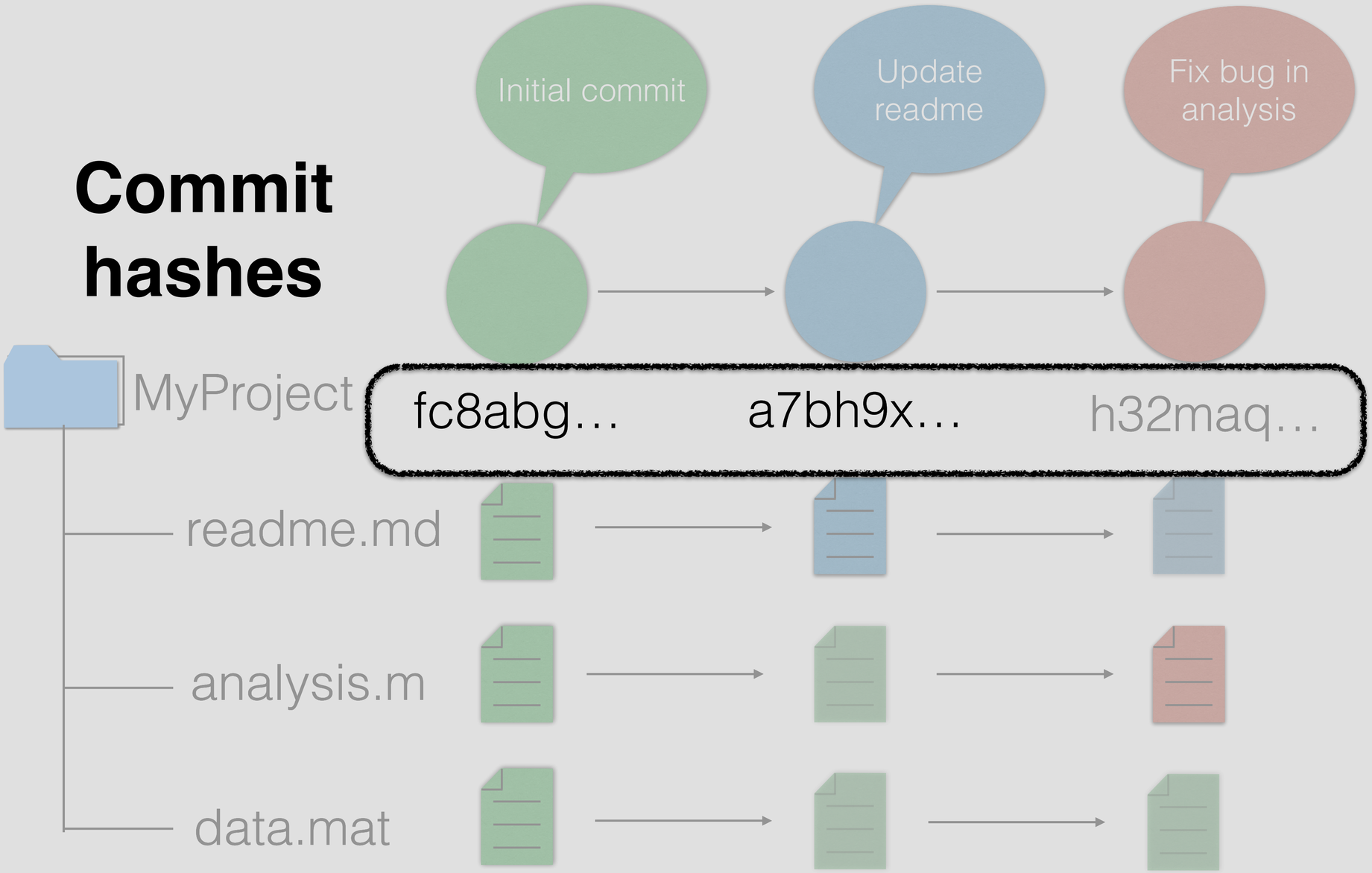


How does Git work?

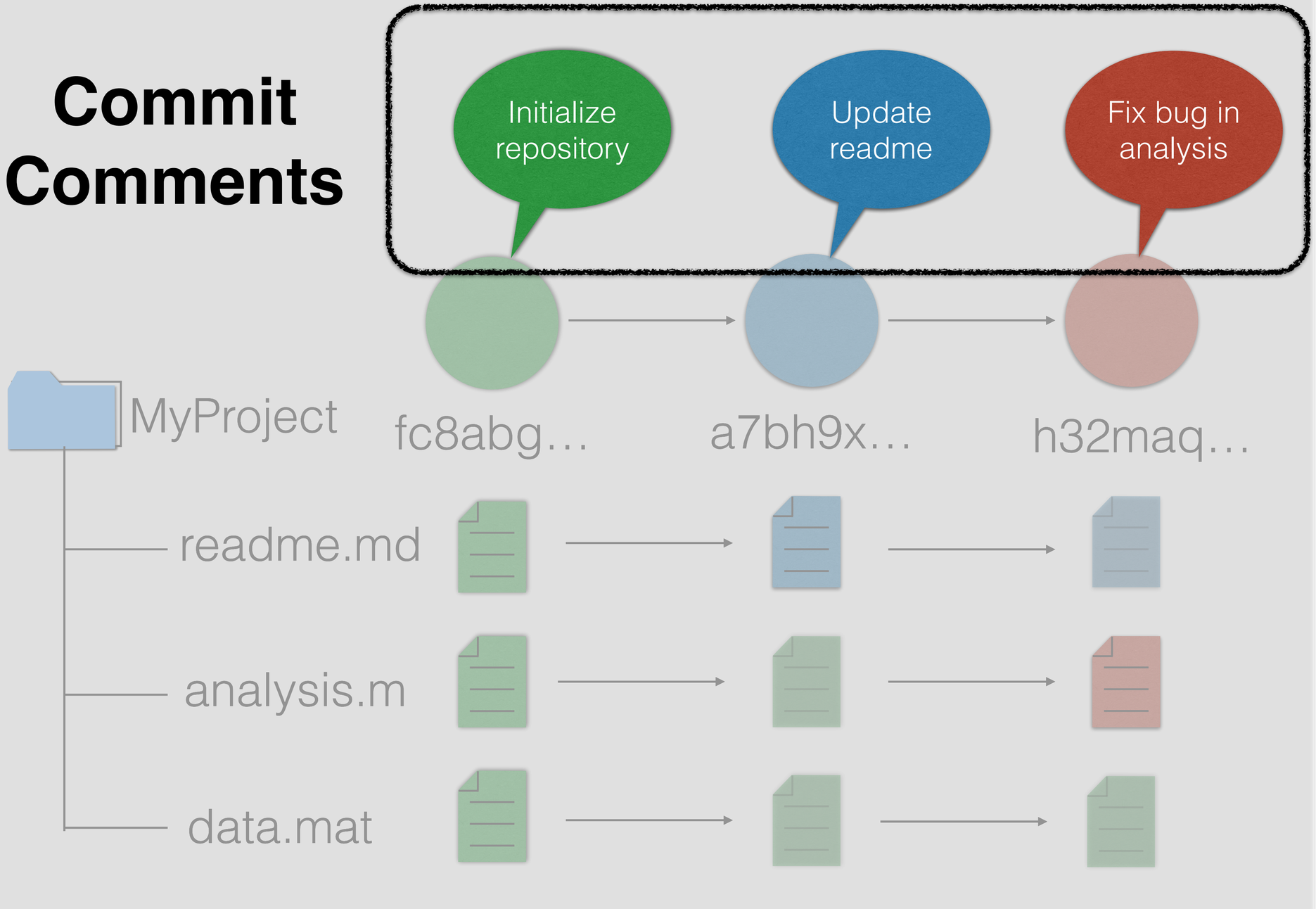


Commits


Initial commit
Update readme
Fix bug in analysis
MyProject
fc8abg…
a7bh9x…
h32maq…
readme.md
analysis.m
data.mat


Commit hashes
Initial commit
Update readme
Fix bug in analysis
MyProject
fc8abg…
a7bh9x…
h32maq…
readme.md
analysis.m
data.mat

Commit Comments



Always keep a backup

- Git is great, but all your history is **local to your computer**
- If your computer breaks, **all history is lost!**
- You want to **keep a “remote” copy!**

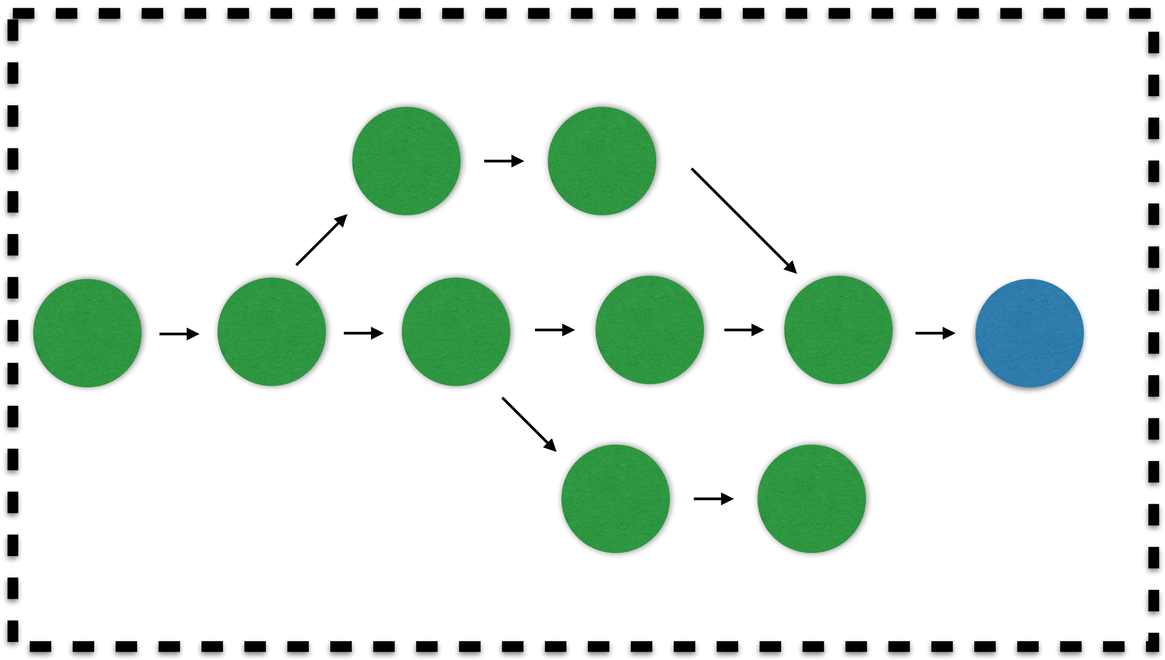
Say hello to GitHub

- GitHub is an online Git repository hosting service
- You can make a remote copy of your Git repository
- You can **“push”** and **“pull”** change to/from GitHub
- Collaborators can **“clone”** your repository to their local machines
- Perfect way to backup your work and collaborate with others!

How does GitHub work?

Edgar

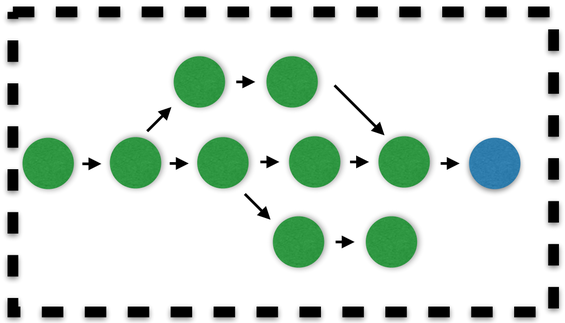
pipeline



You (usually) start with your local Git repository

How does GitHub work?

pipeline



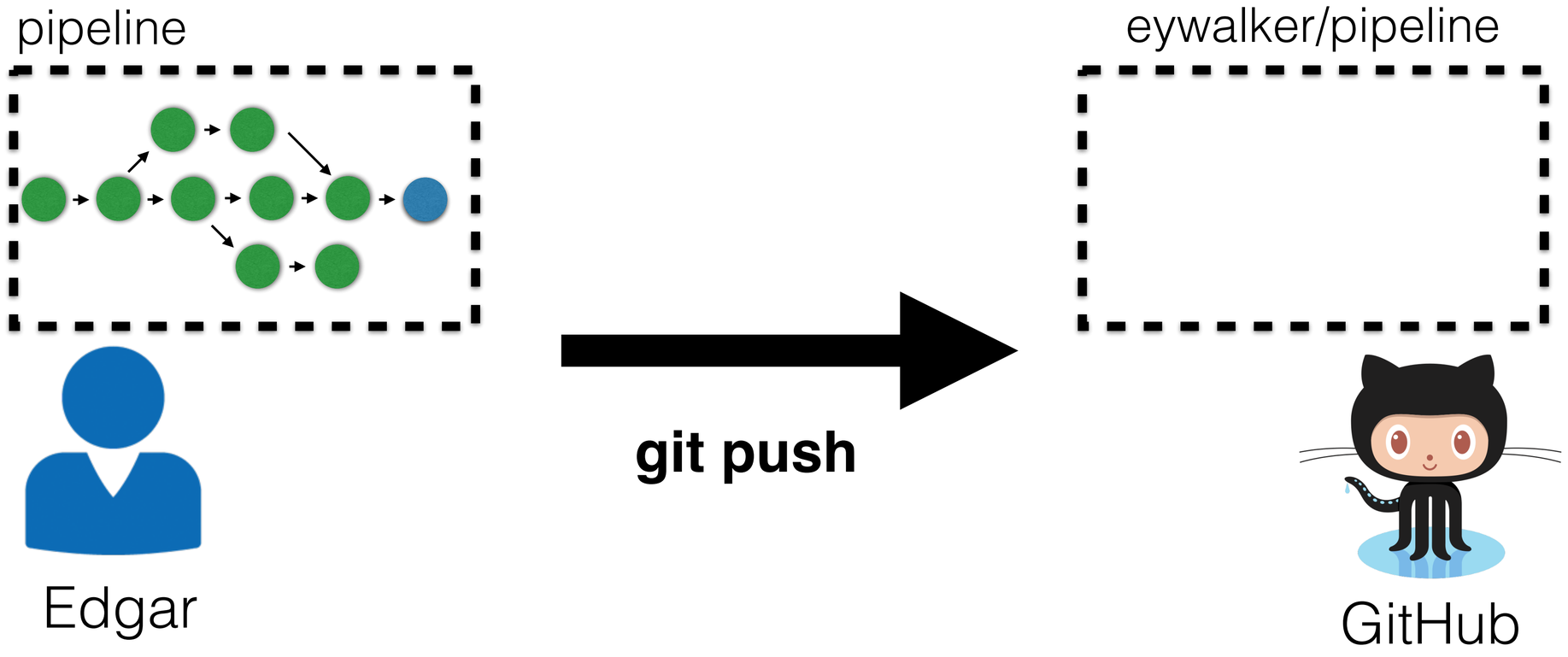
Edgar

eywalker/pipeline

GitHub

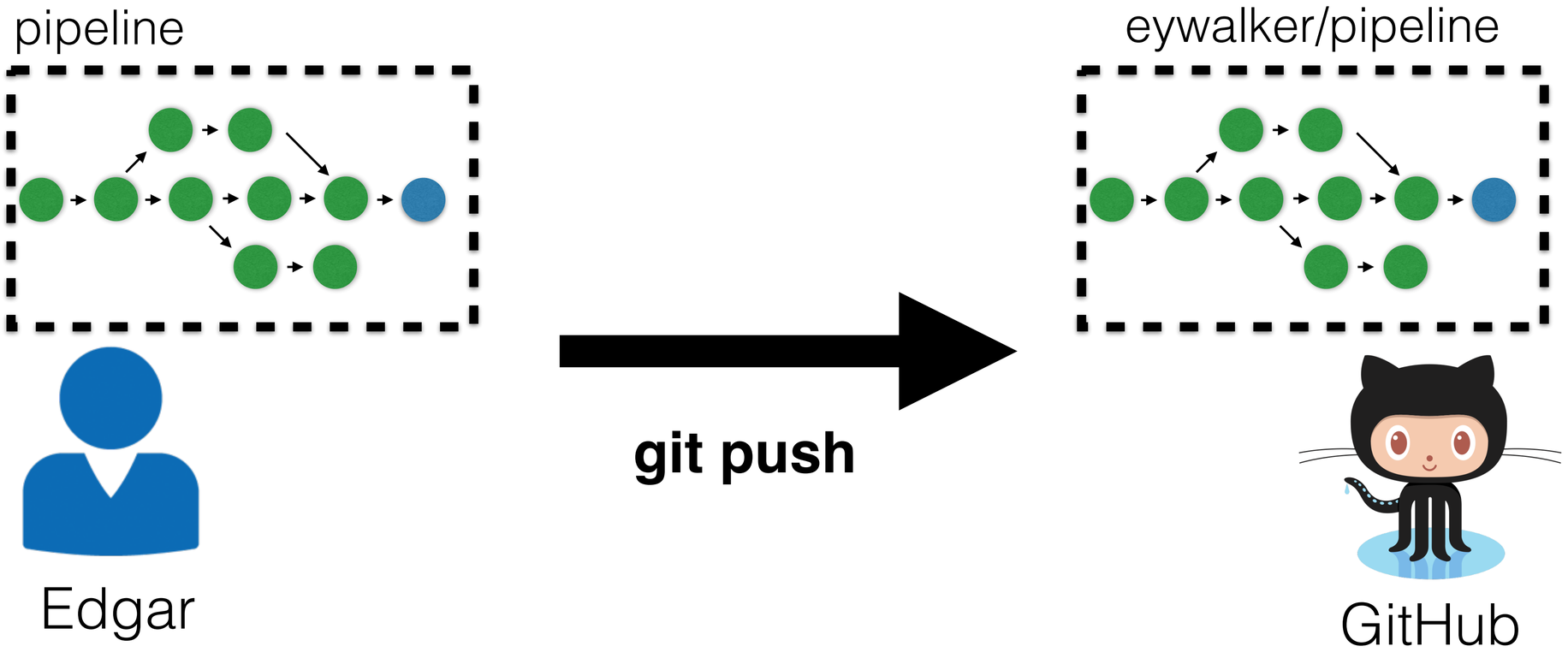
You set up an empty repository on GitHub under your username

How does GitHub work?



You **push** your local repository to the **GitHub repository**

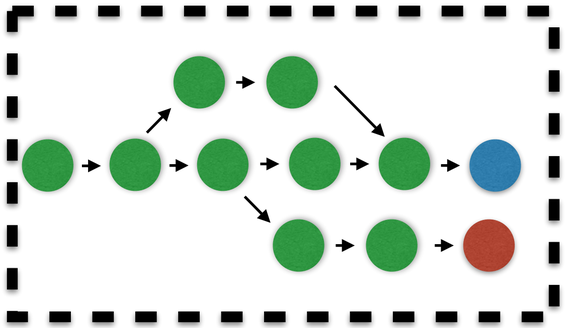
How does GitHub work?



Now the GitHub repository is synched with your local repository!

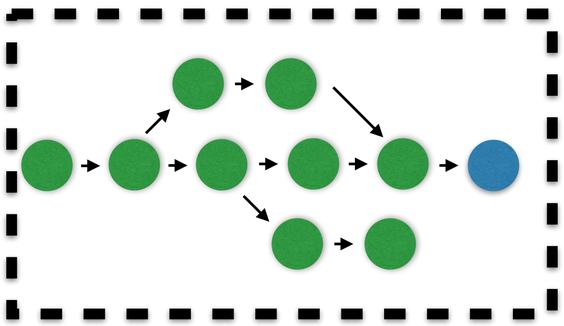
How does GitHub work?

pipeline



Edgar

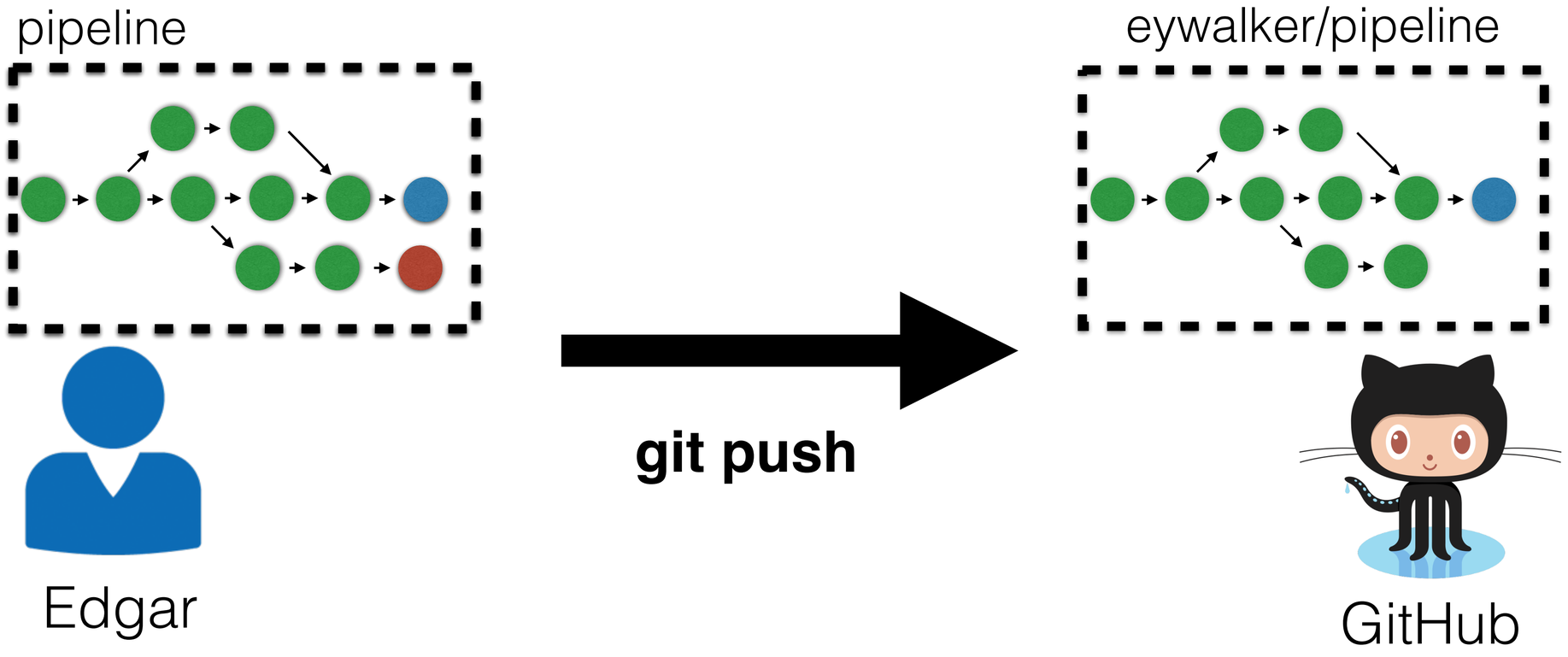
eywalker/pipeline



GitHub

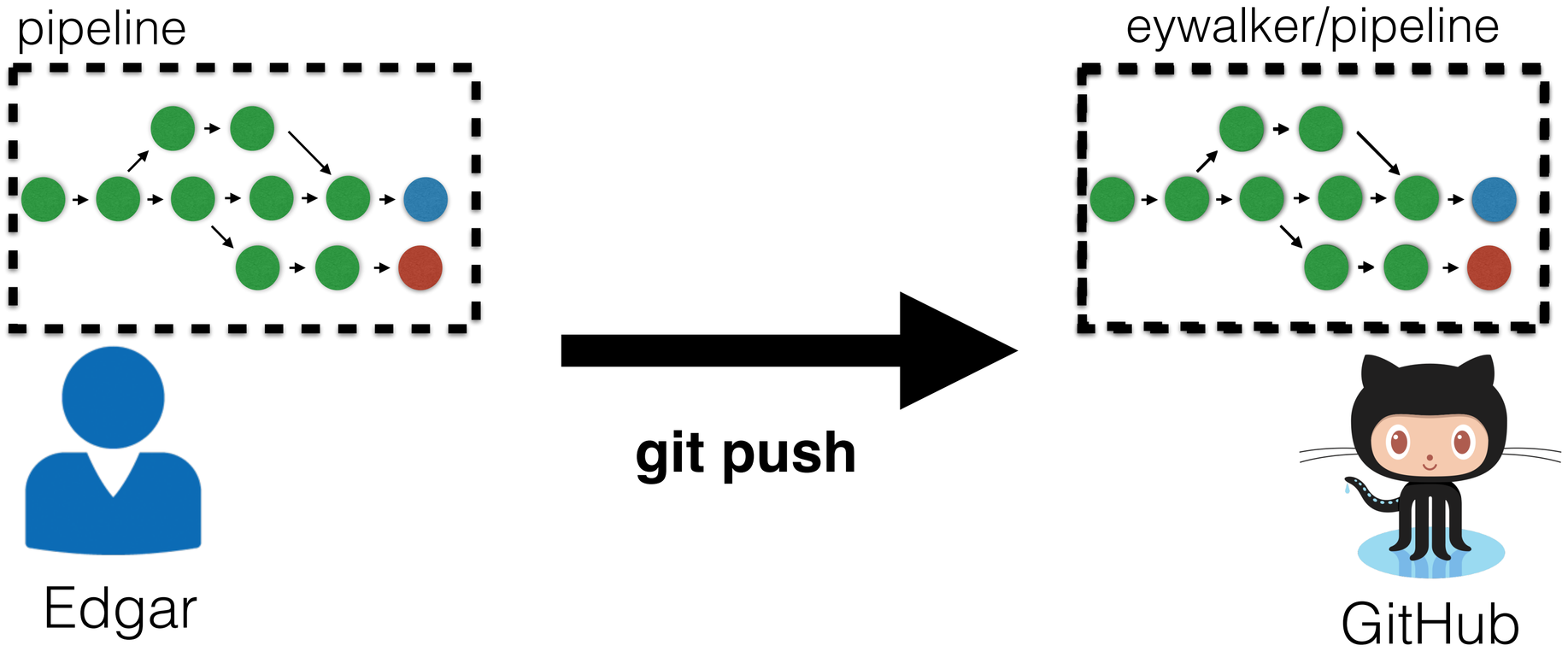
If you make some changes…

How does GitHub work?



Push again to the **origin**

How does GitHub work?



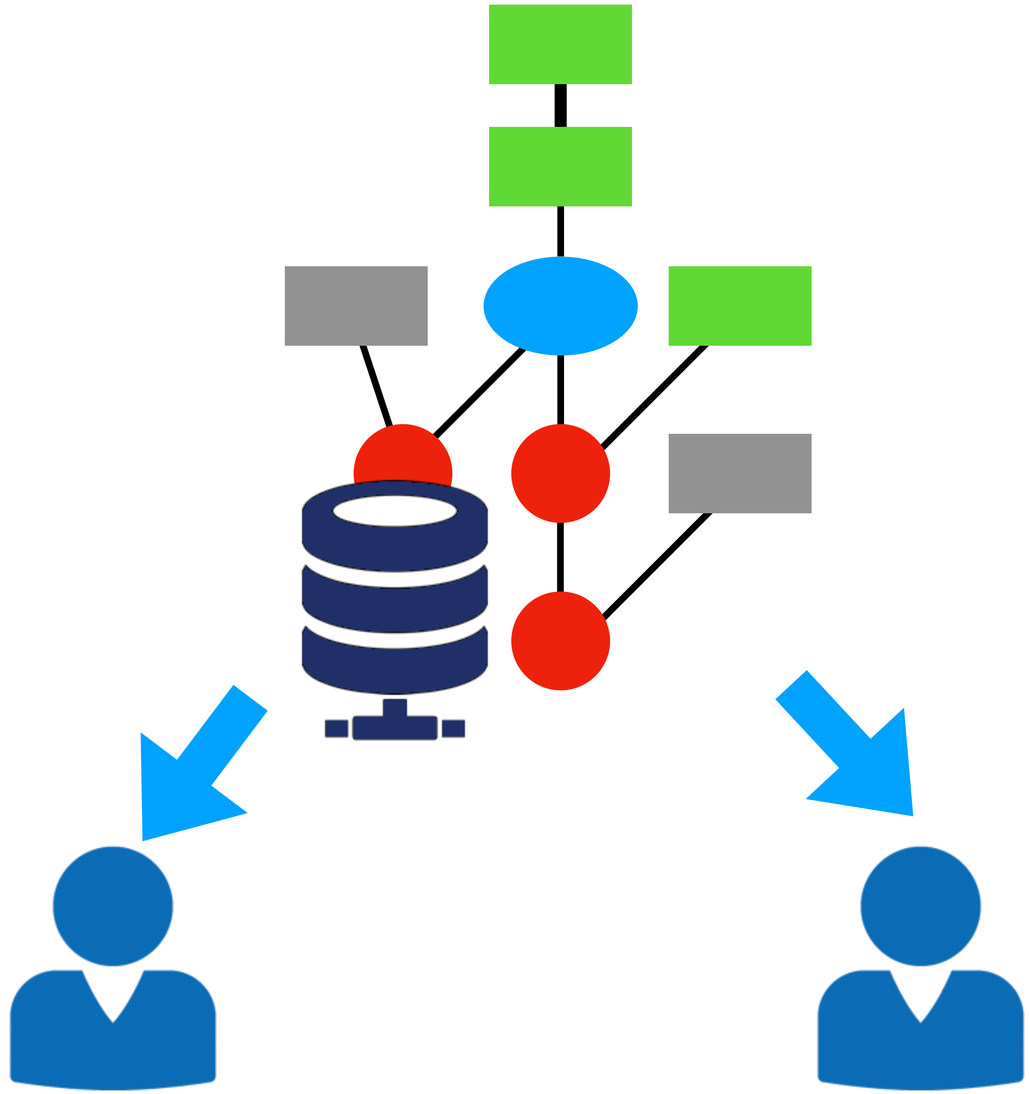
Push again to the **origin**

Keeping track of your code

- Git and GitHub together let's you keep track of changes you make to your code (e.g. data pipeline definitions)
- GitHub provides you with a backup of your **entire history of your code**
- It's also a great way to share your code!

Sharing a DataJoint pipeline: Read only

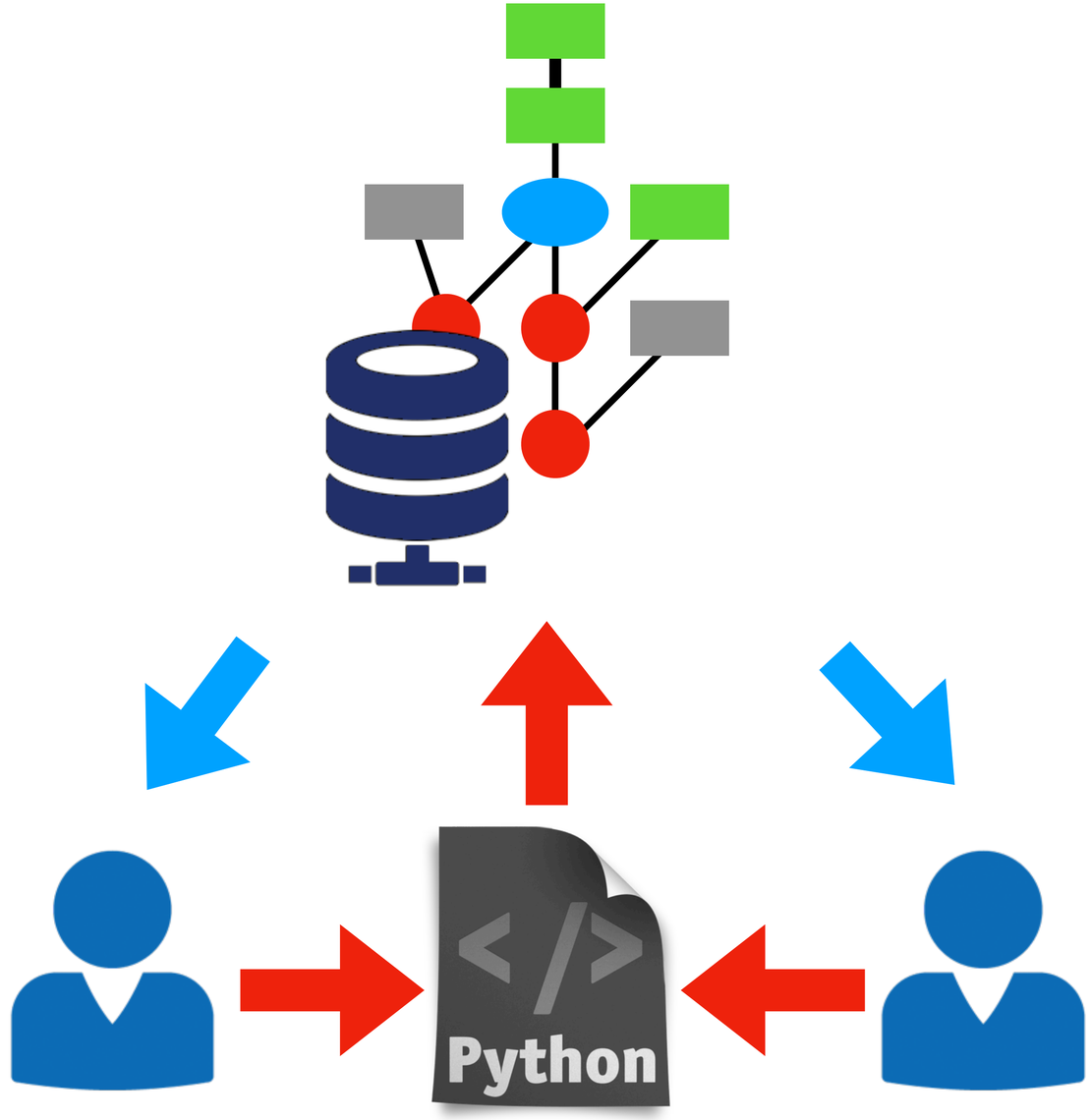
- Sharing a data pipeline can be as easy as providing an access to the DataJoint database
- For read-only access, you don't even need to know the code used to create the tables!
- Your collaborator can happily define new tables in their own schema, extending the data pipeline



What if you do want the other person to be able to **insert data** into your data pipeline?

Sharing a DataJoint pipeline Read and Write

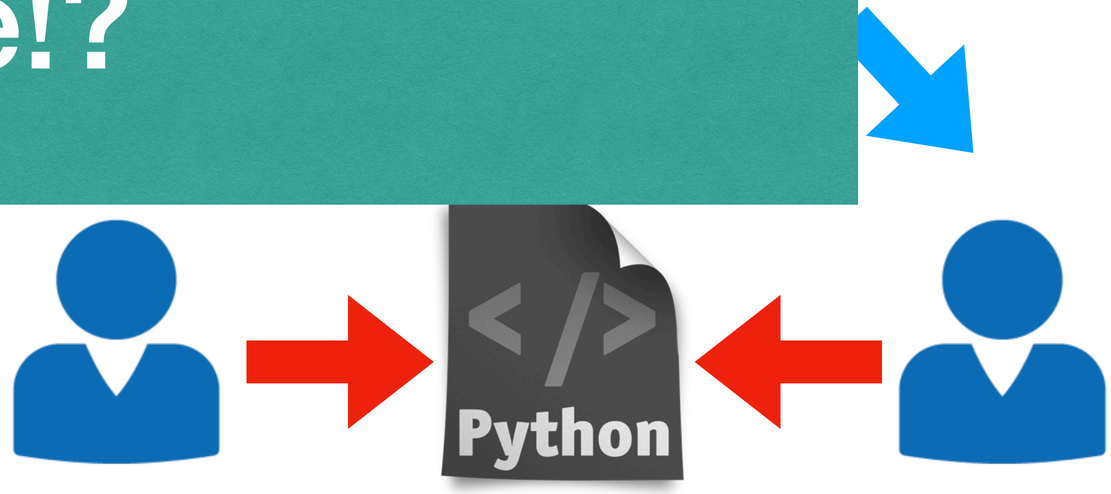
- To insert data into a data pipeline, the code for the pipeline needs to be shared
- Everyone should use same table definitions to ensure consistency in entered data!



Sharing a DataJoint pipeline Read and Write

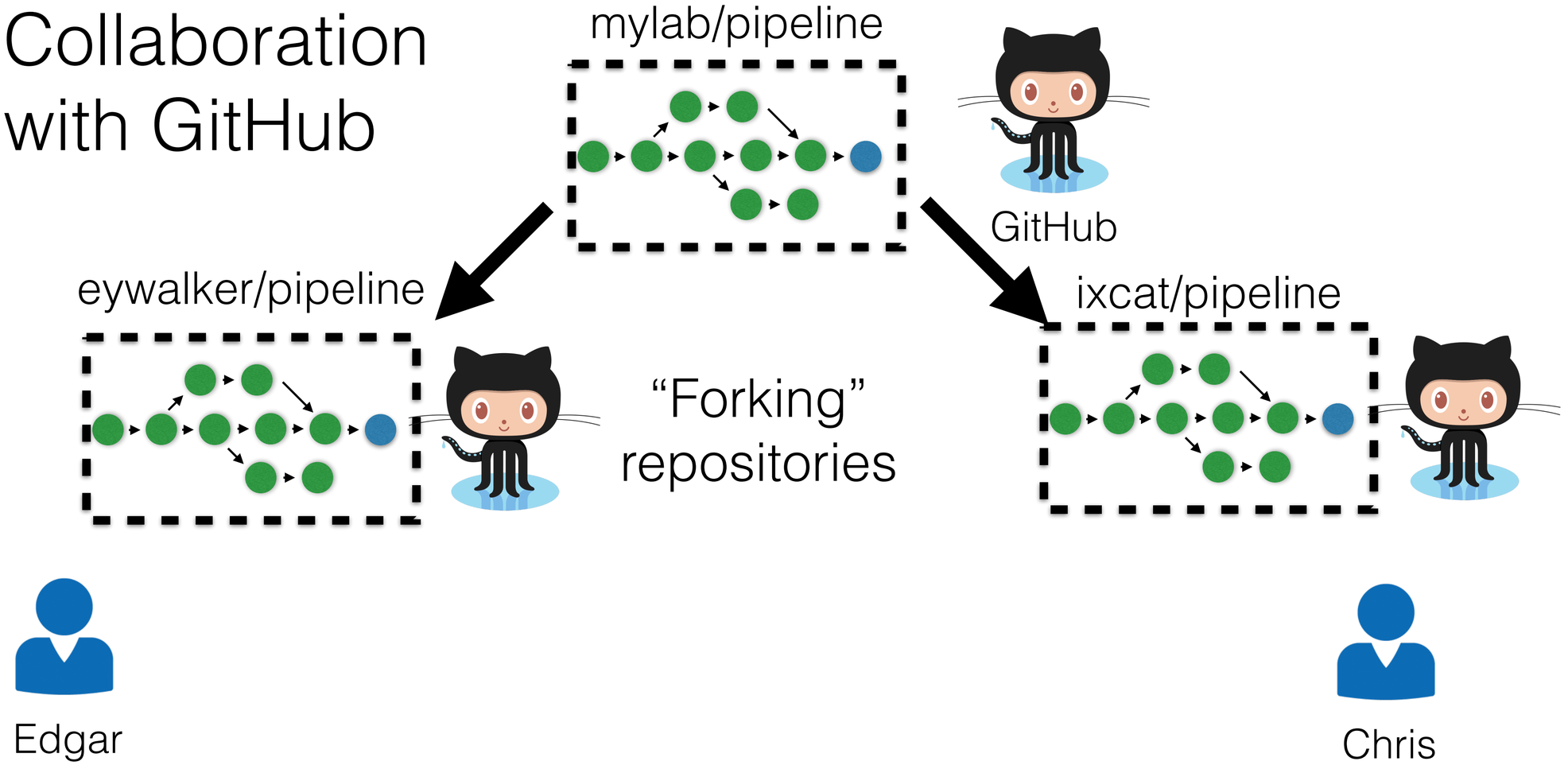
- To
  the
  be
- Eve
  de
  in entered data!

But how can we ensure that they are all using the same code!?

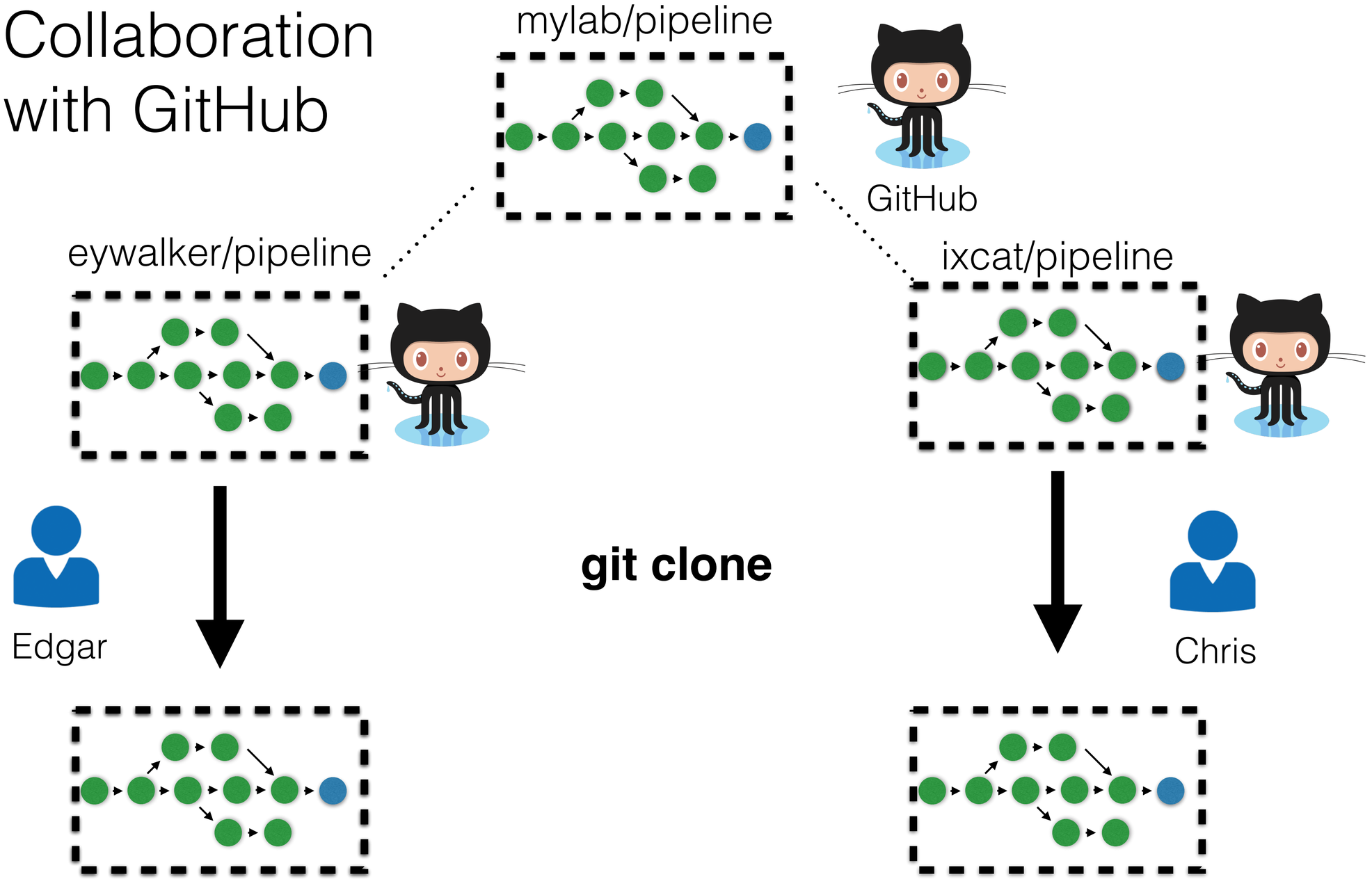


Collaborative Workflow in GitHub

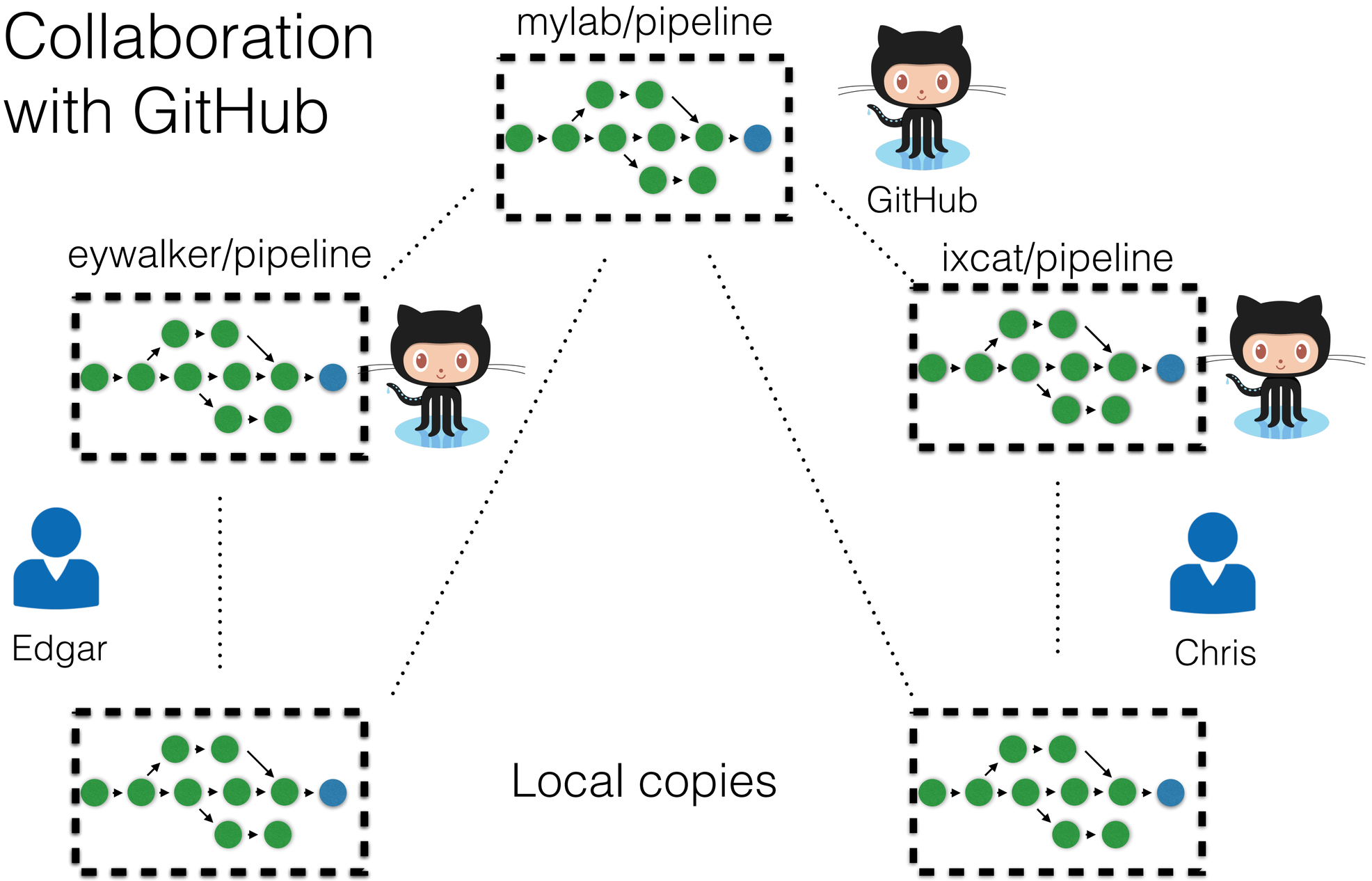
Collaboration with GitHub



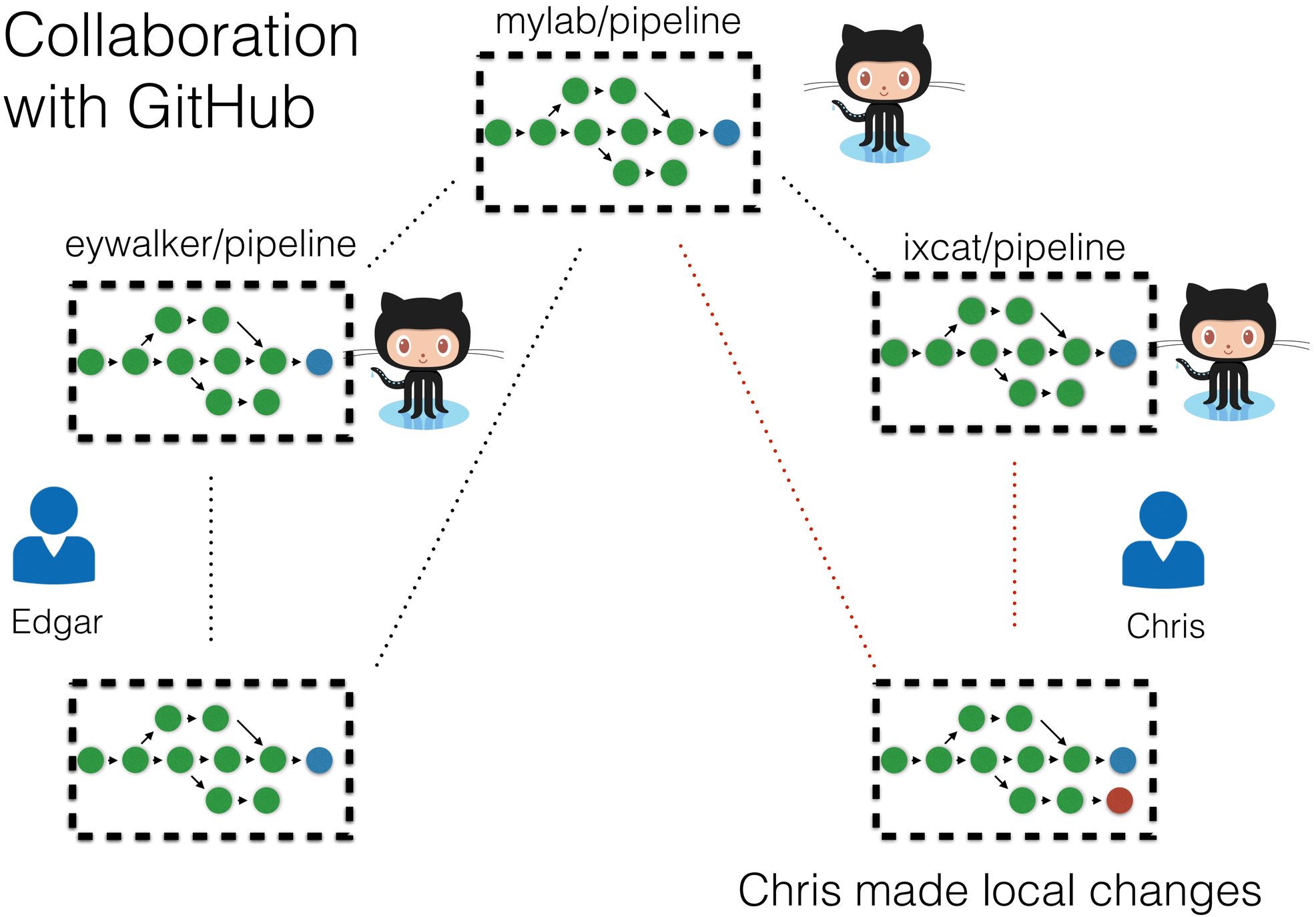
Collaboration with GitHub



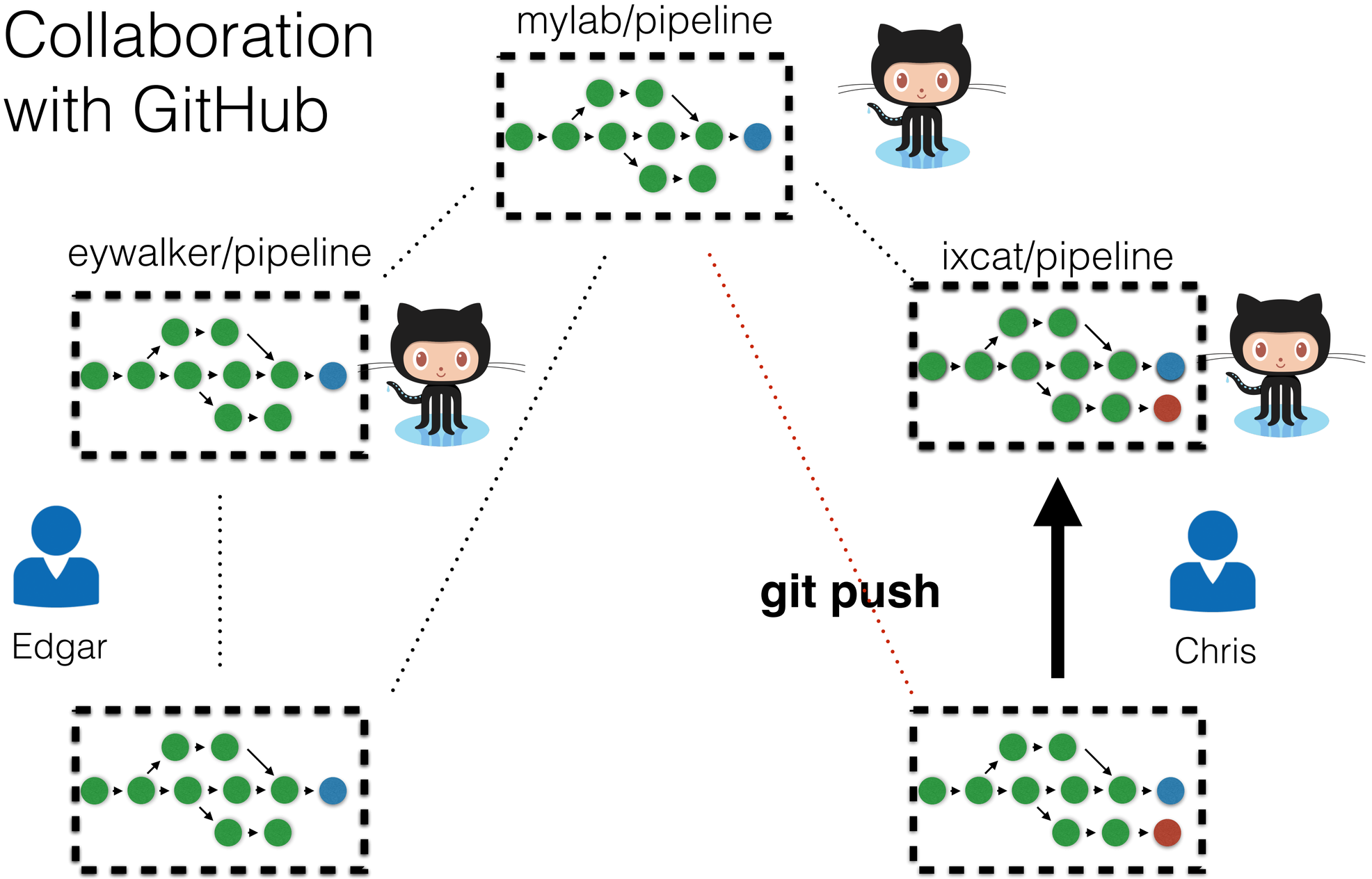
Collaboration with GitHub



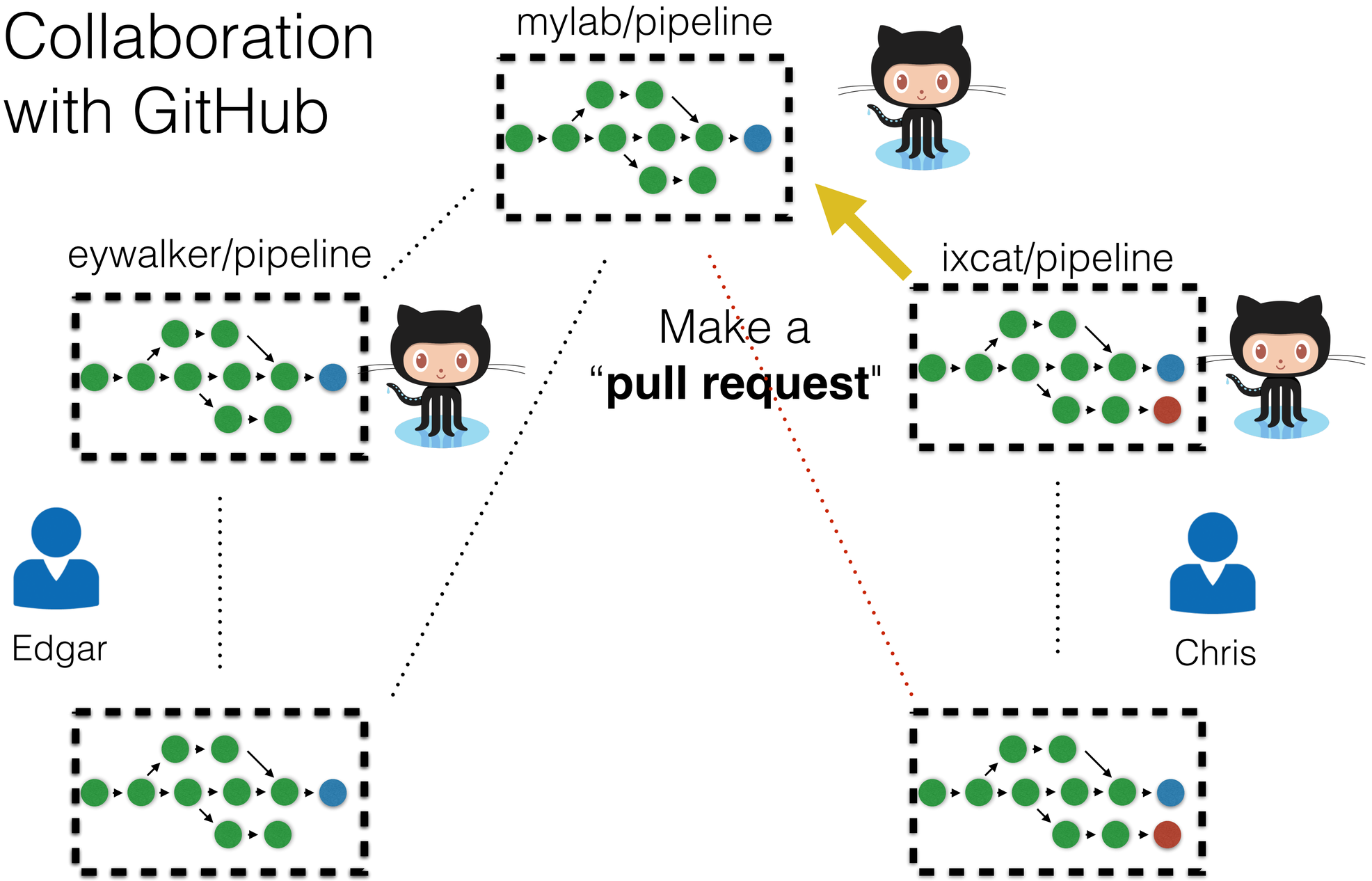
Collaboration with GitHub



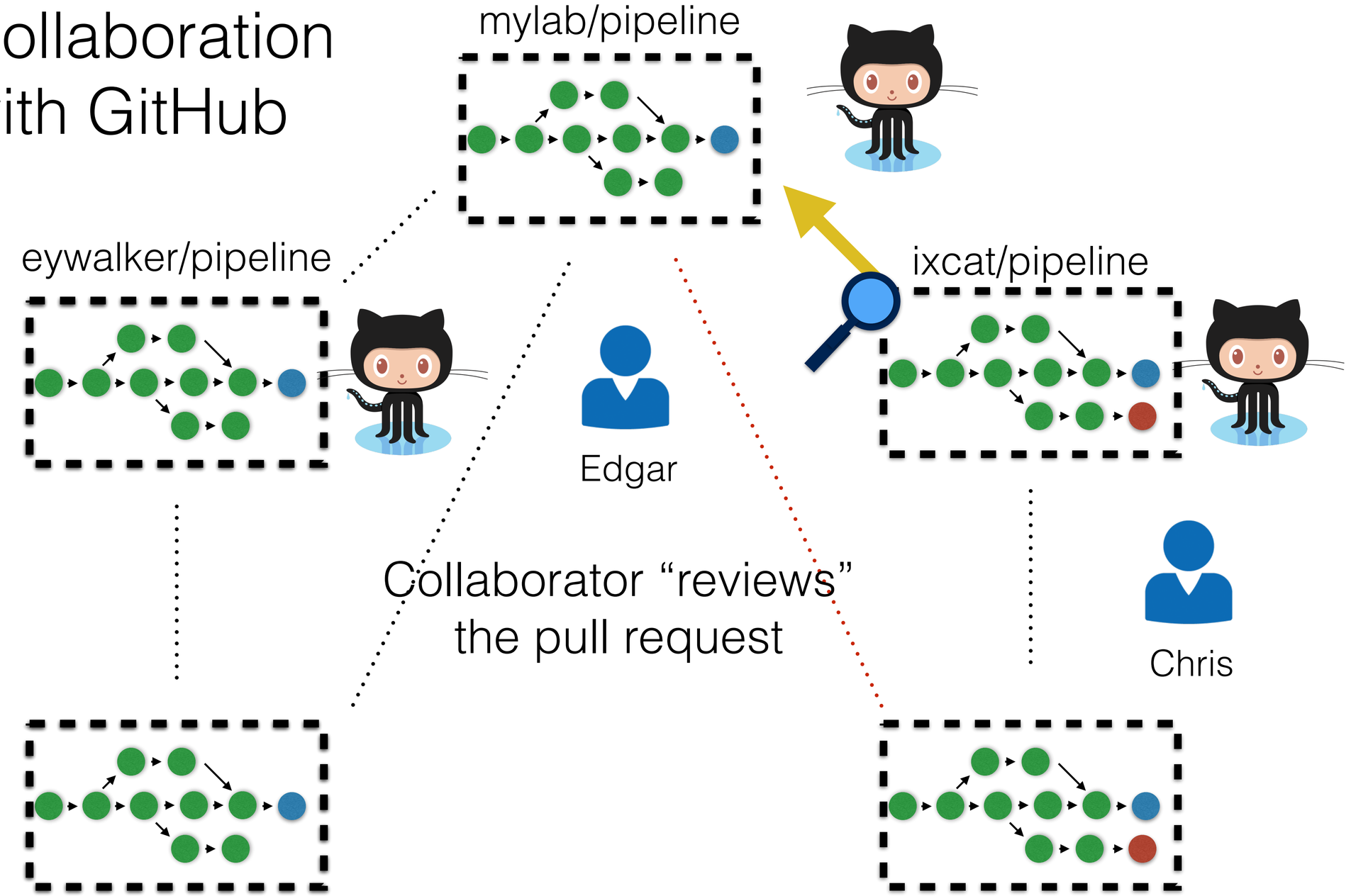
Collaboration with GitHub



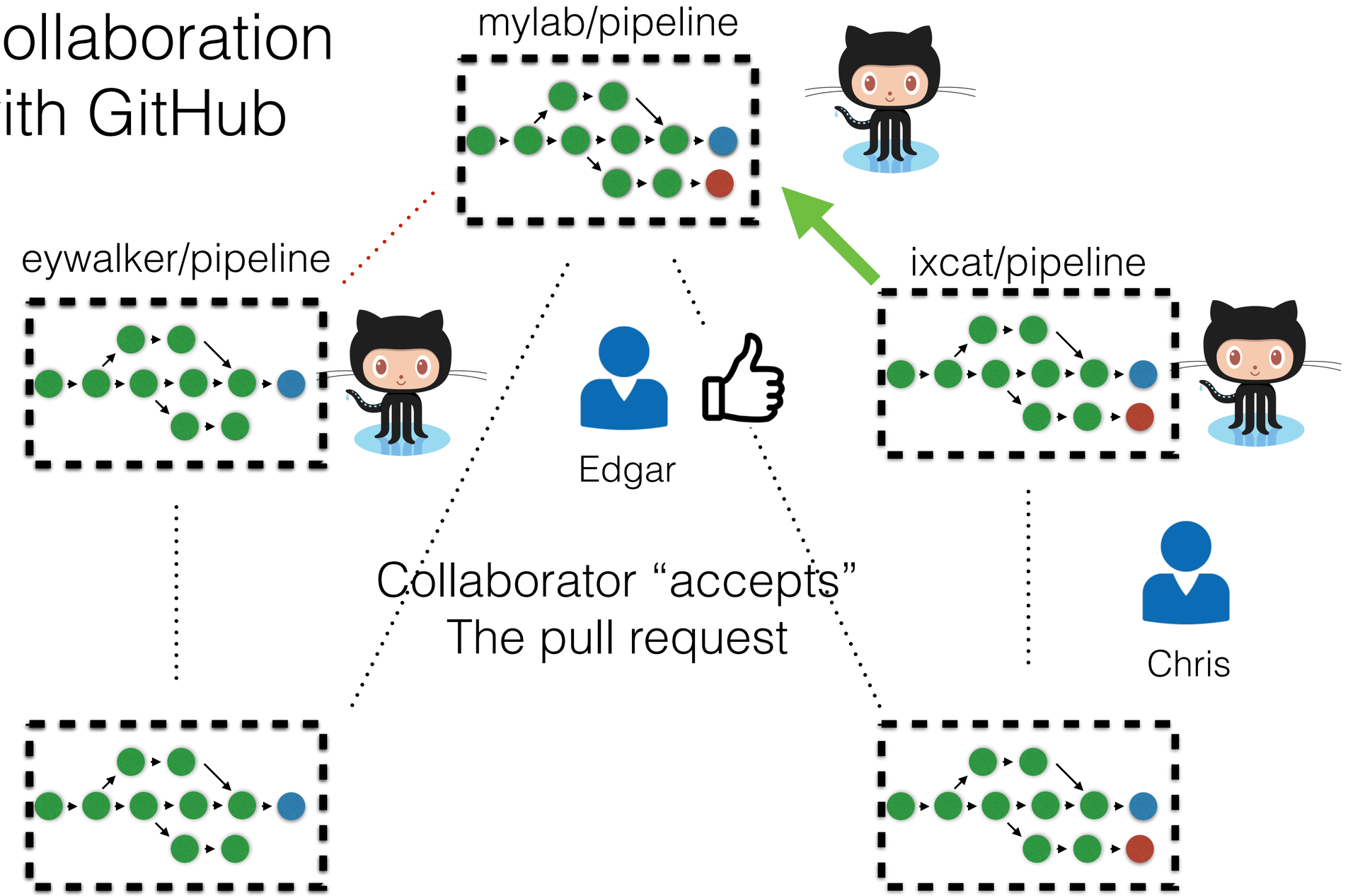
Collaboration with GitHub



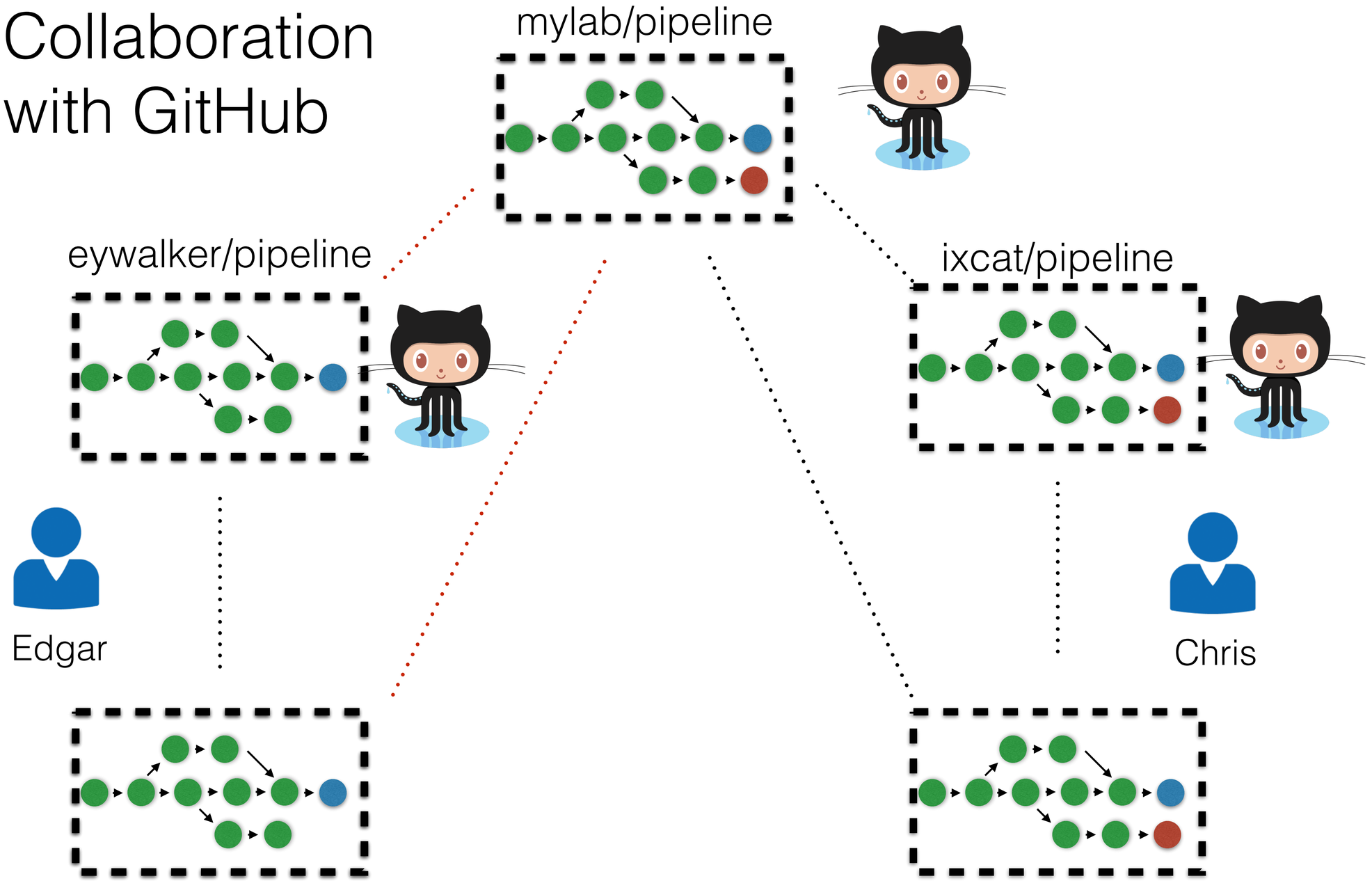
Collaboration with GitHub



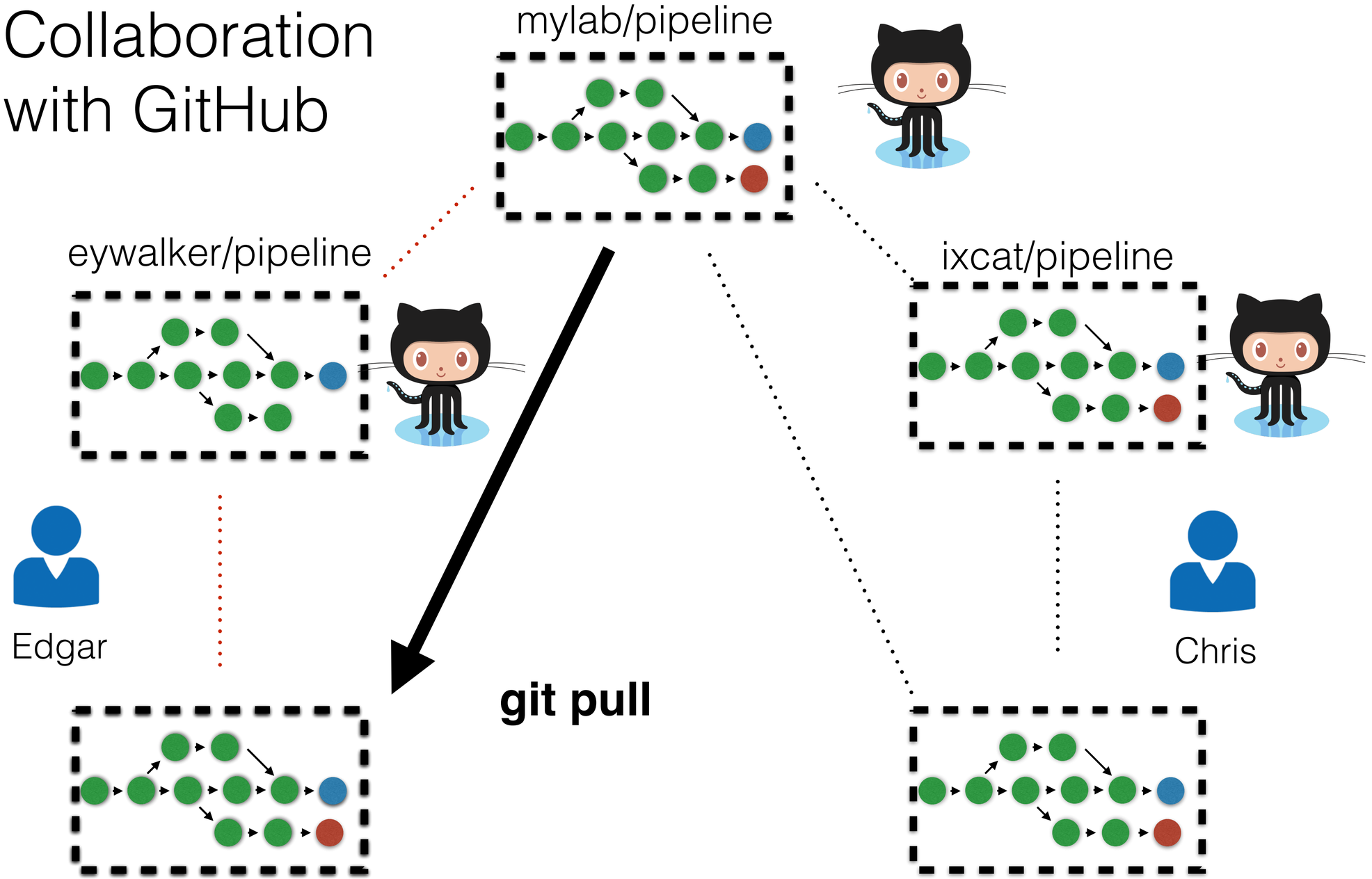
Collaboration with GitHub



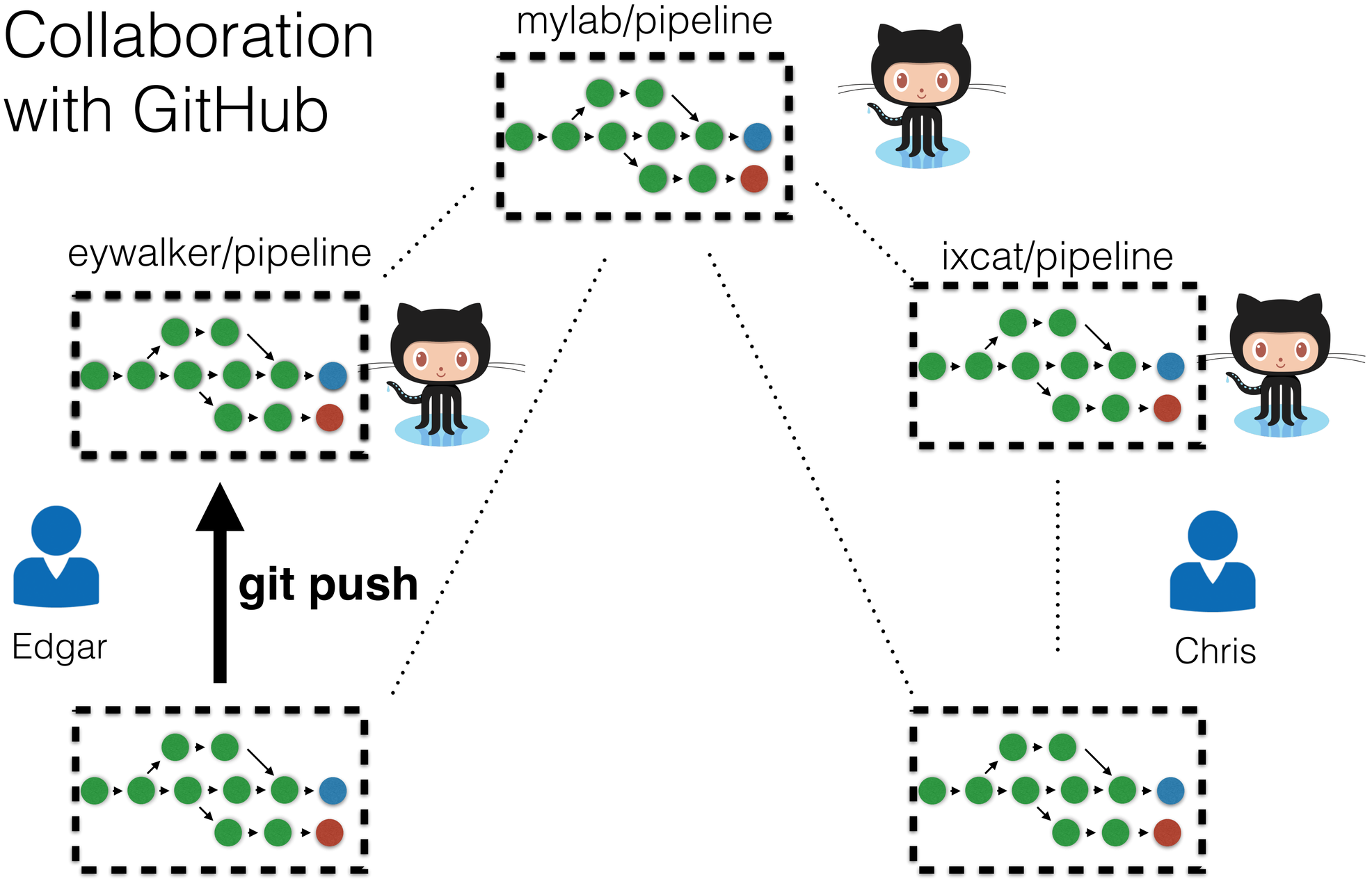
Collaboration with GitHub



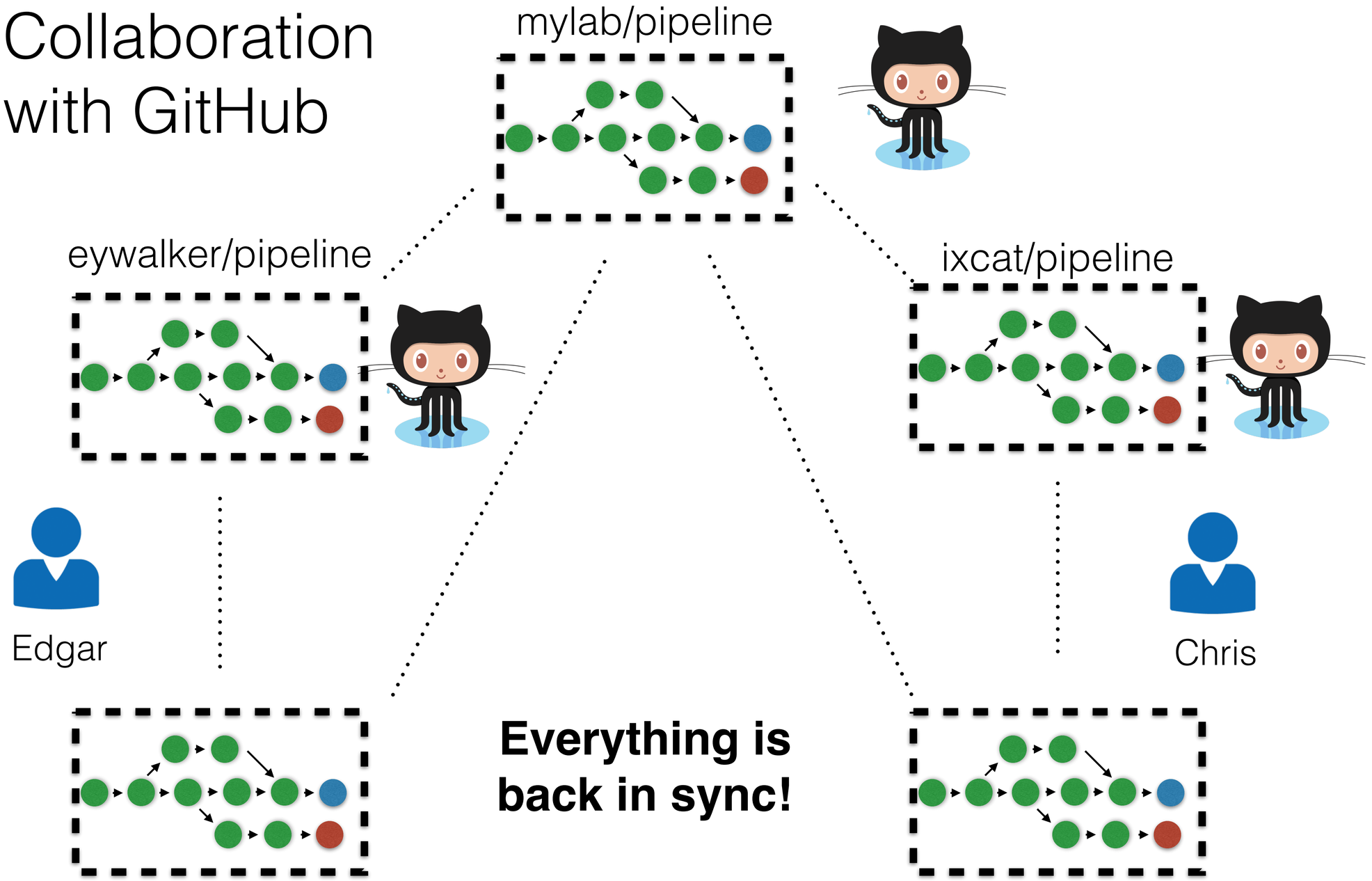
Collaboration with GitHub



Collaboration with GitHub



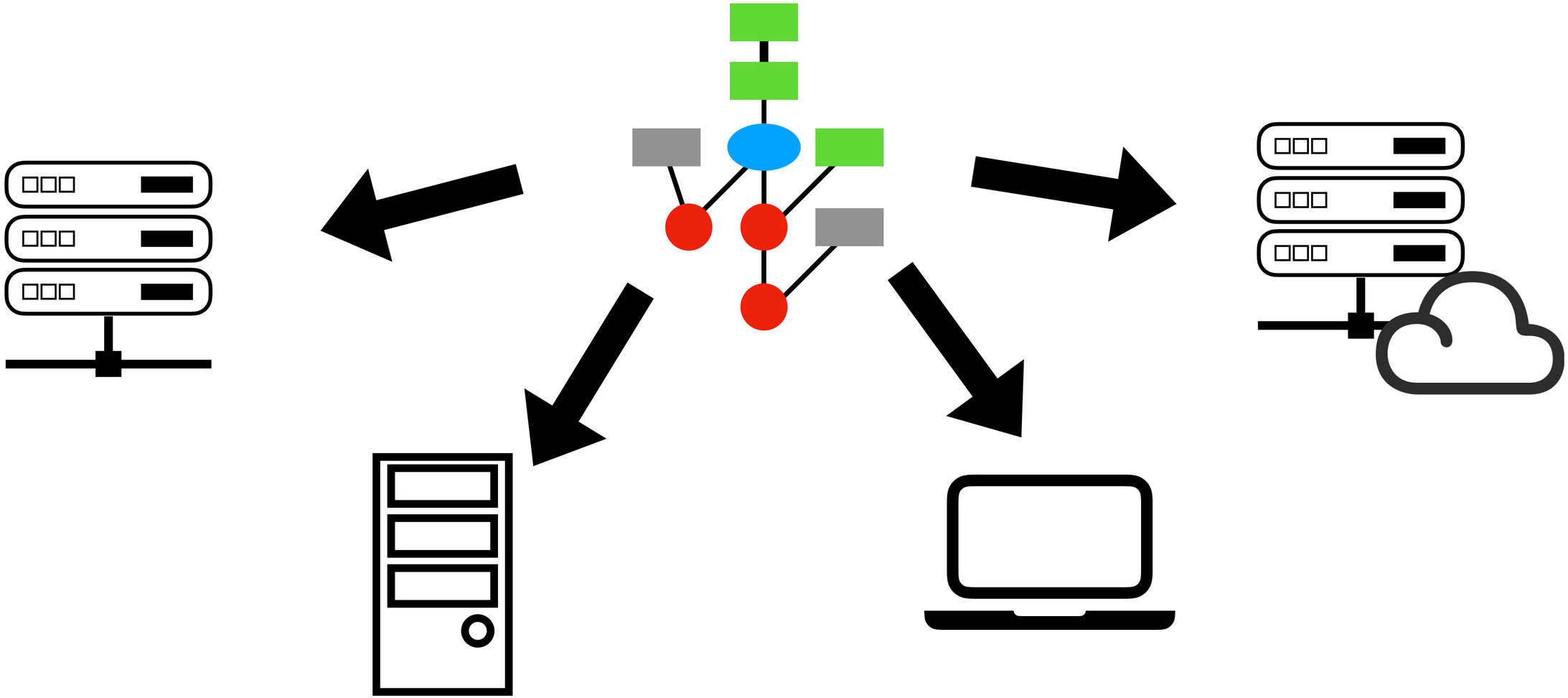
Collaboration with GitHub



Sharing your pipeline code

- If multiple people are going to work on a common pipeline, good code sharing practice is **absolutely essential!!**
- GitHub pull request workflow is a popular and effective way to maintain a well-reviewed data pipeline code
- But ultimately everything depends on each and every member of the group following the best practices
- Nothing replaces good communication!

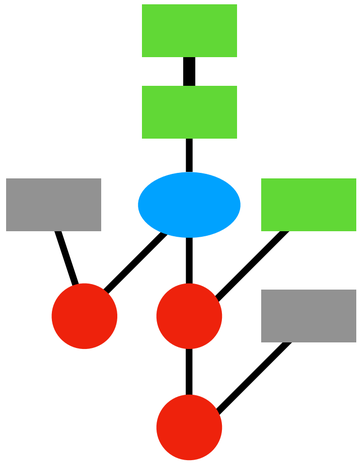
Running your data pipeline



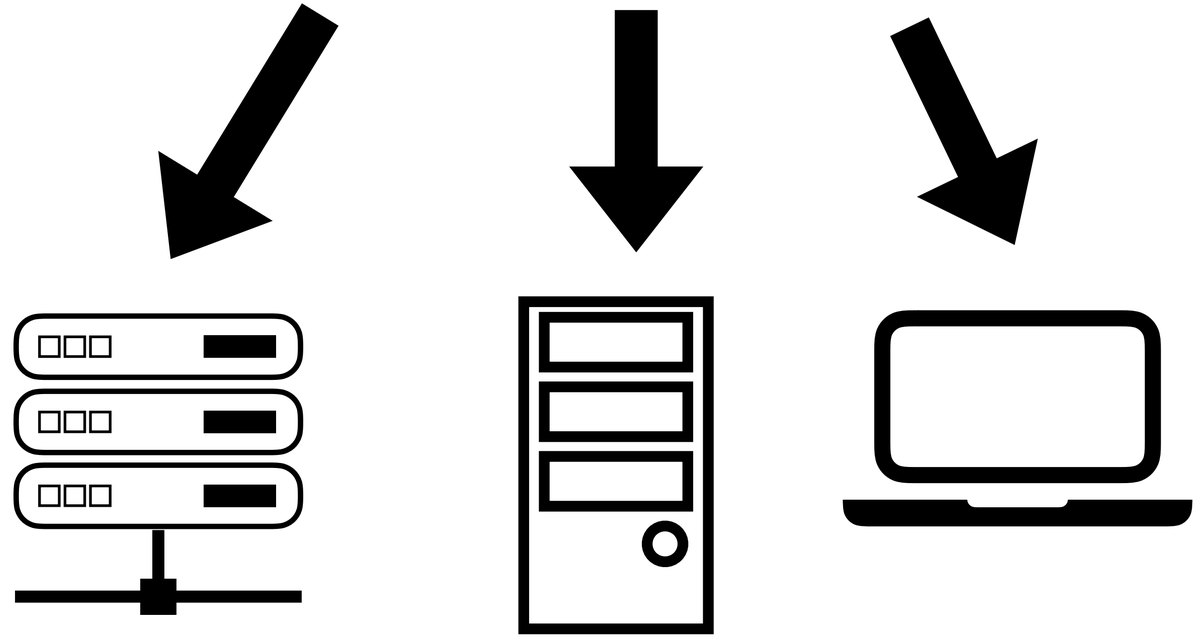
You can “run” your pipeline across multiple machines

- Running the pipeline is almost equivalent to calling “populate” on your Imported and Computed tables

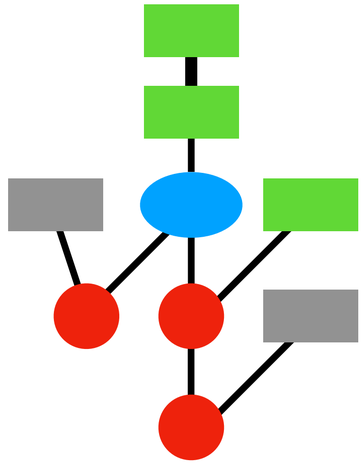
- “populate” can be run with keyword “reserve_jobs=True” to automatically distribute the task across any number of machines!



populate(reserve_jobs=True)

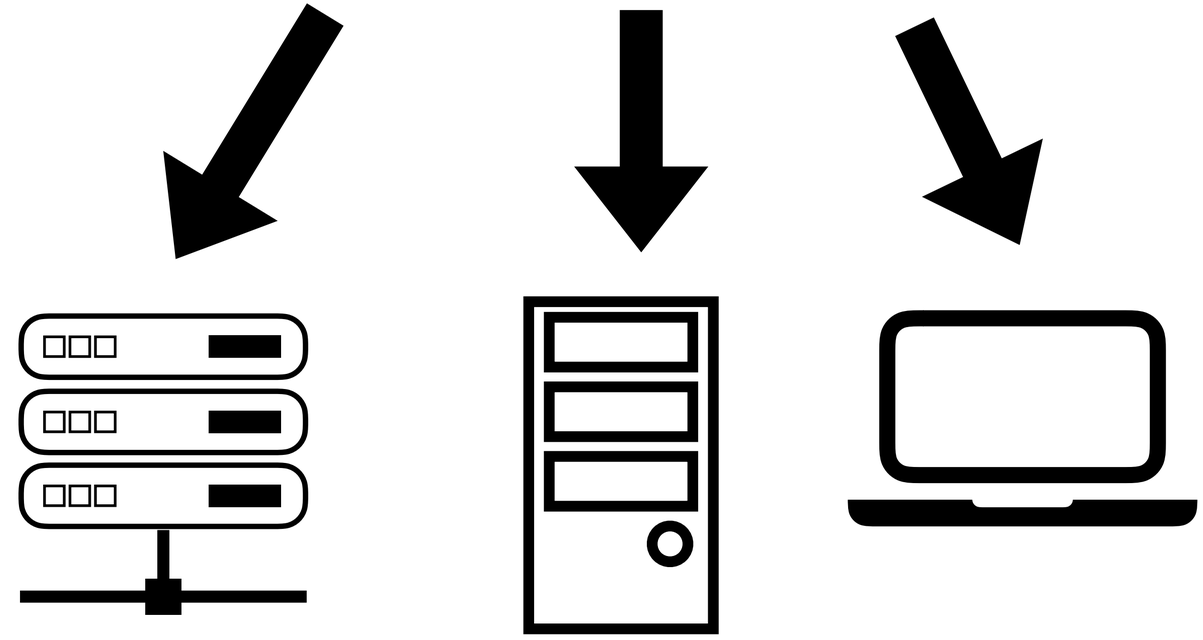


But how do you ensure they are computing the “same” thing?



- Ensure that they all have same code for the data pipeline
  - Git/GitHub can be used to ensure same version of pipeline is running
- …but having same code does **not** guarantee same results!

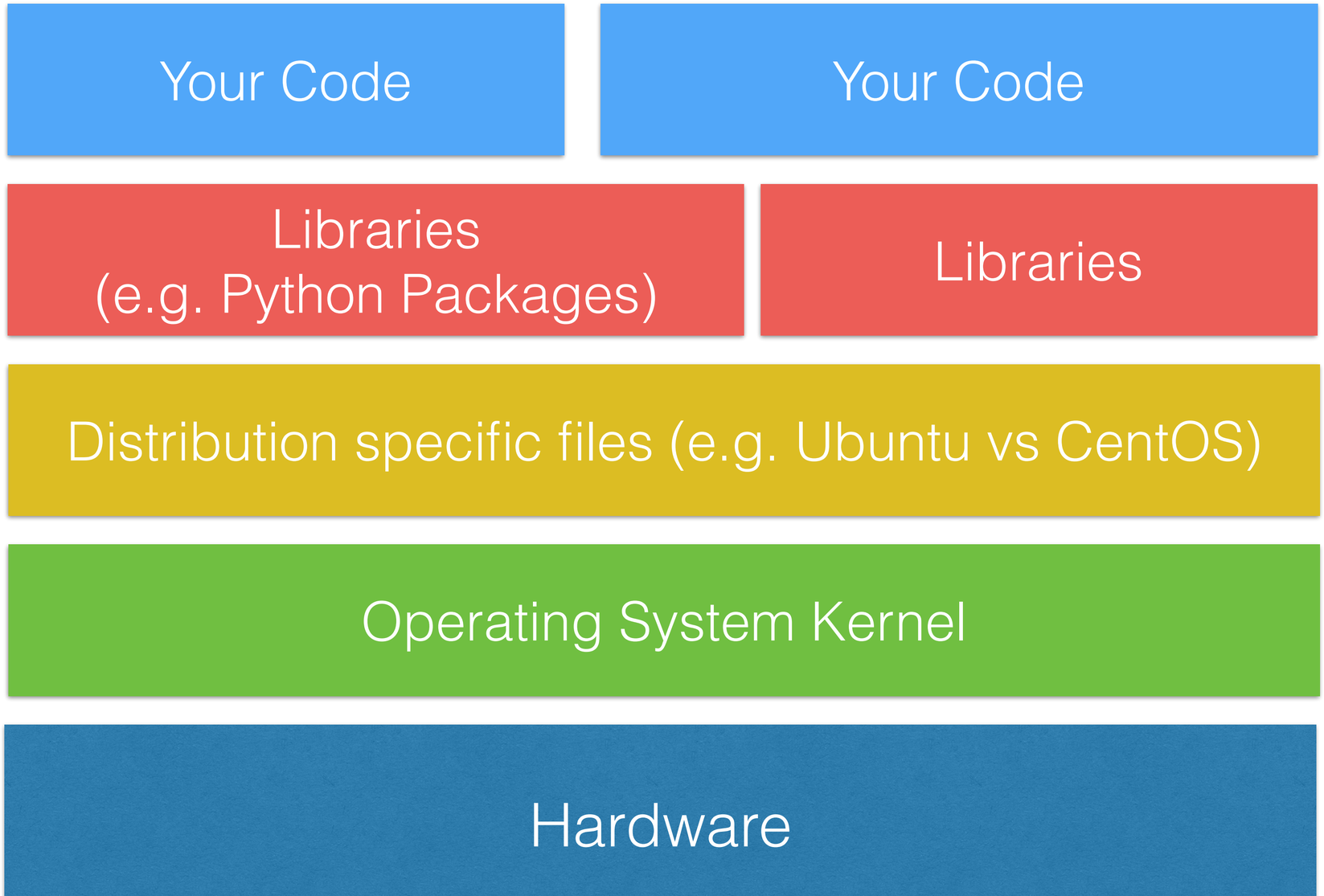
populate(reserve_jobs=True)



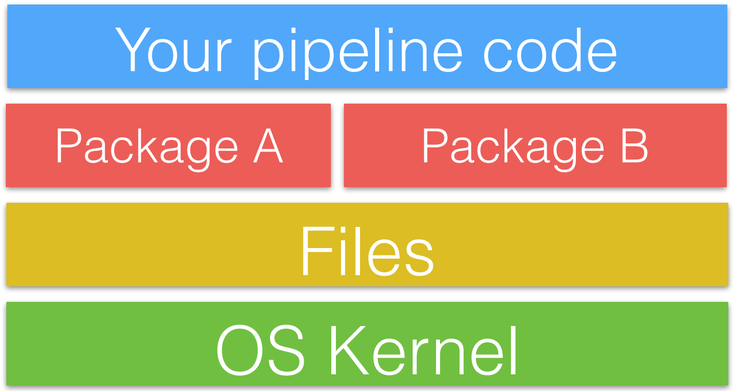
Code is half of the equation to the reproducibility

- Your code depends on many things **outside of your code**
  - Other libraries
  - File systems
  - Operating system
- Your code runs in **an environment**

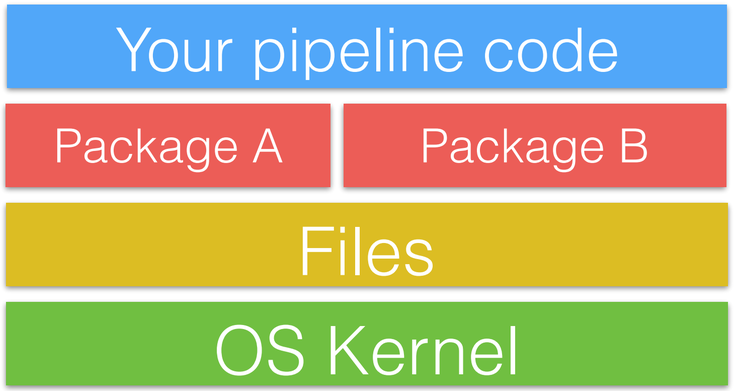
Code environment



Ideal setup

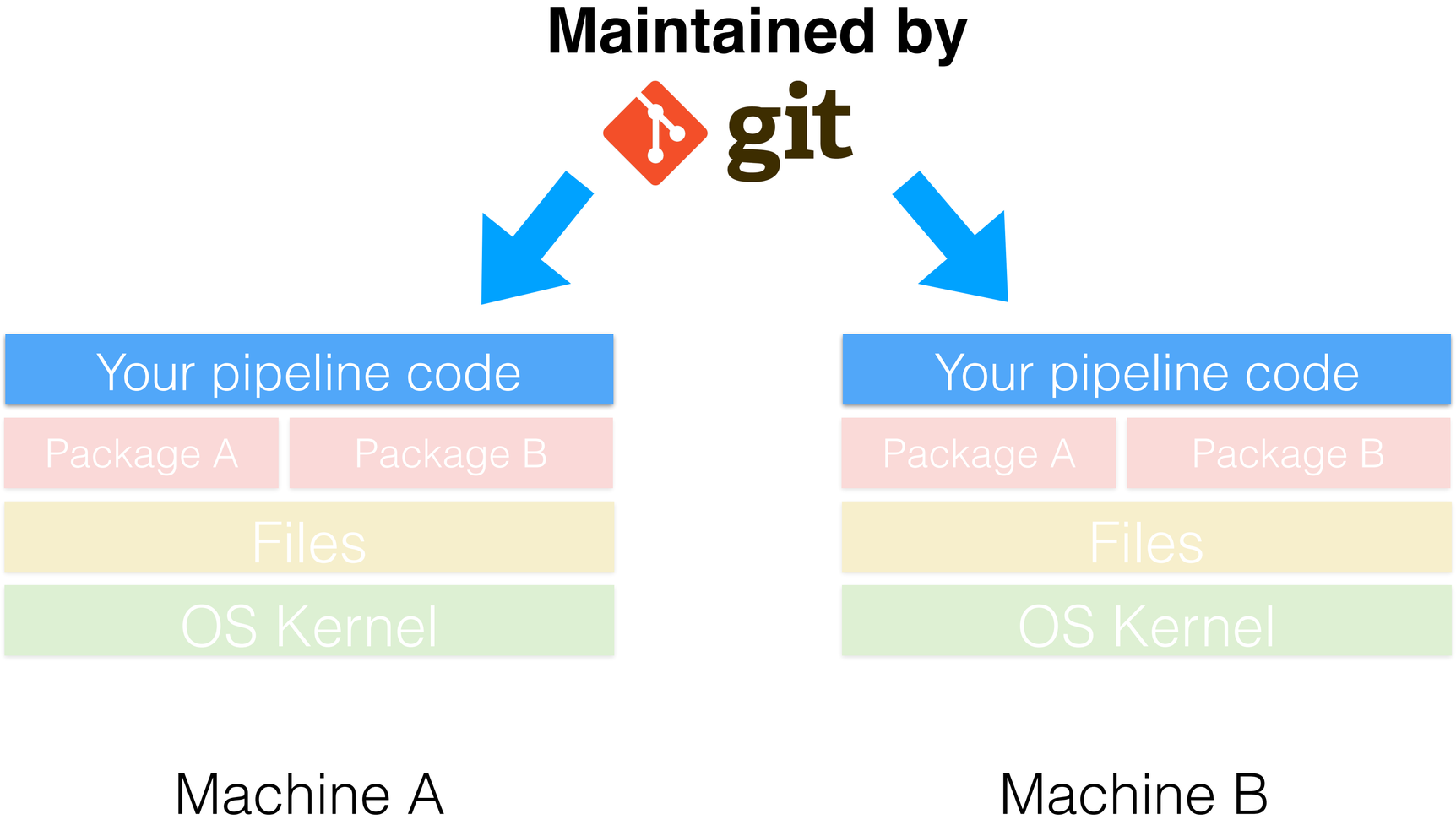


Machine A

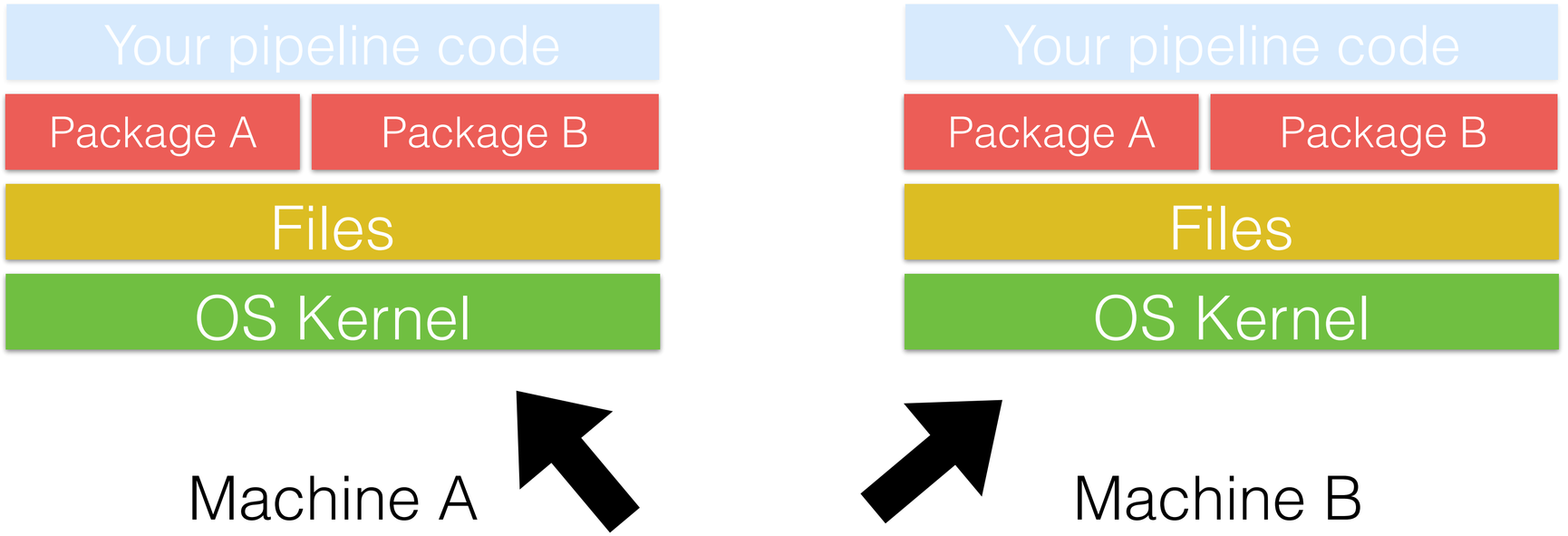


Machine B

Ideal setup

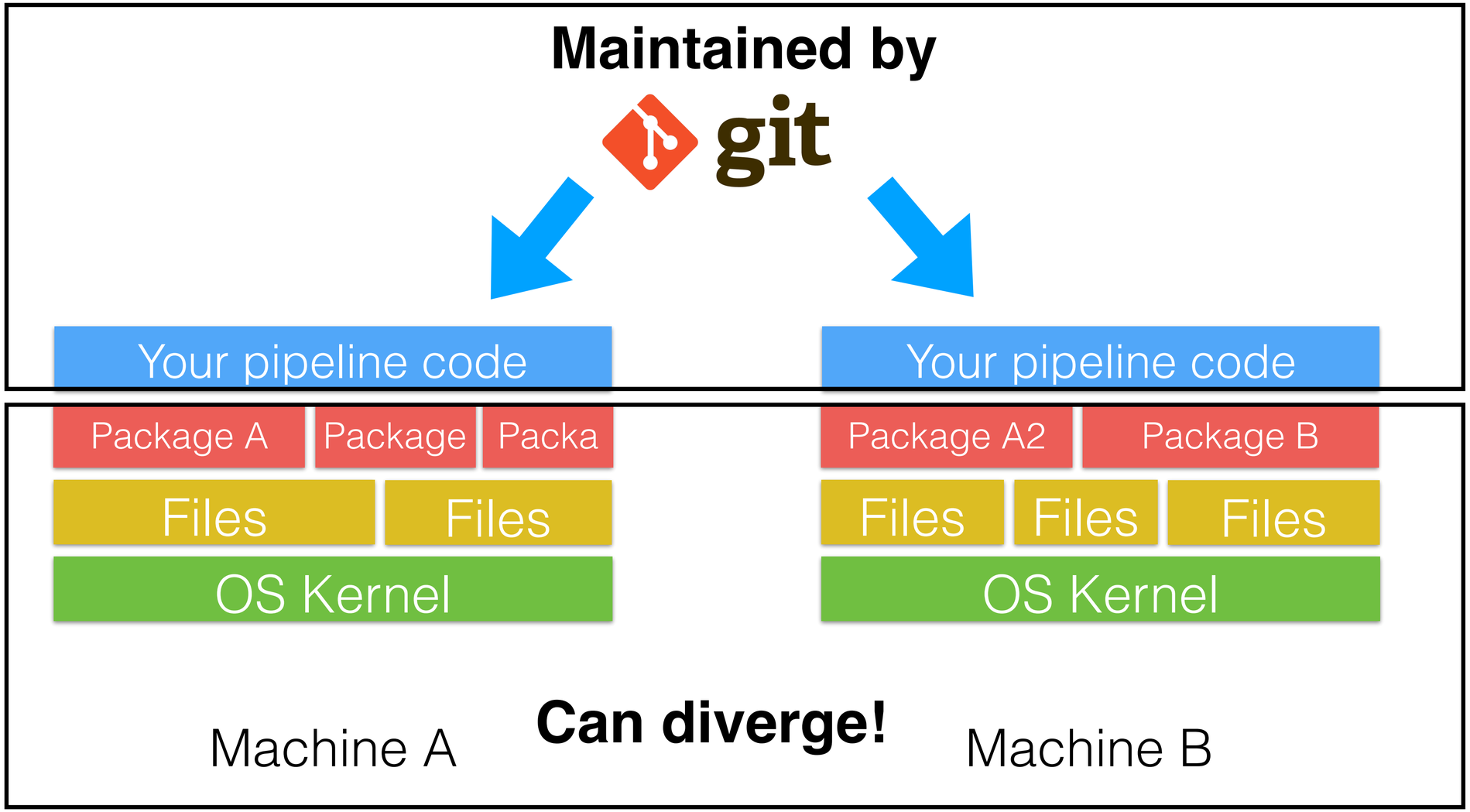

Maintained by
git
Your pipeline code
Package A
Package B
Files
OS Kernel
Your pipeline code
Package A
Package B
Files
OS Kernel
Machine A
Machine B

Ideal setup

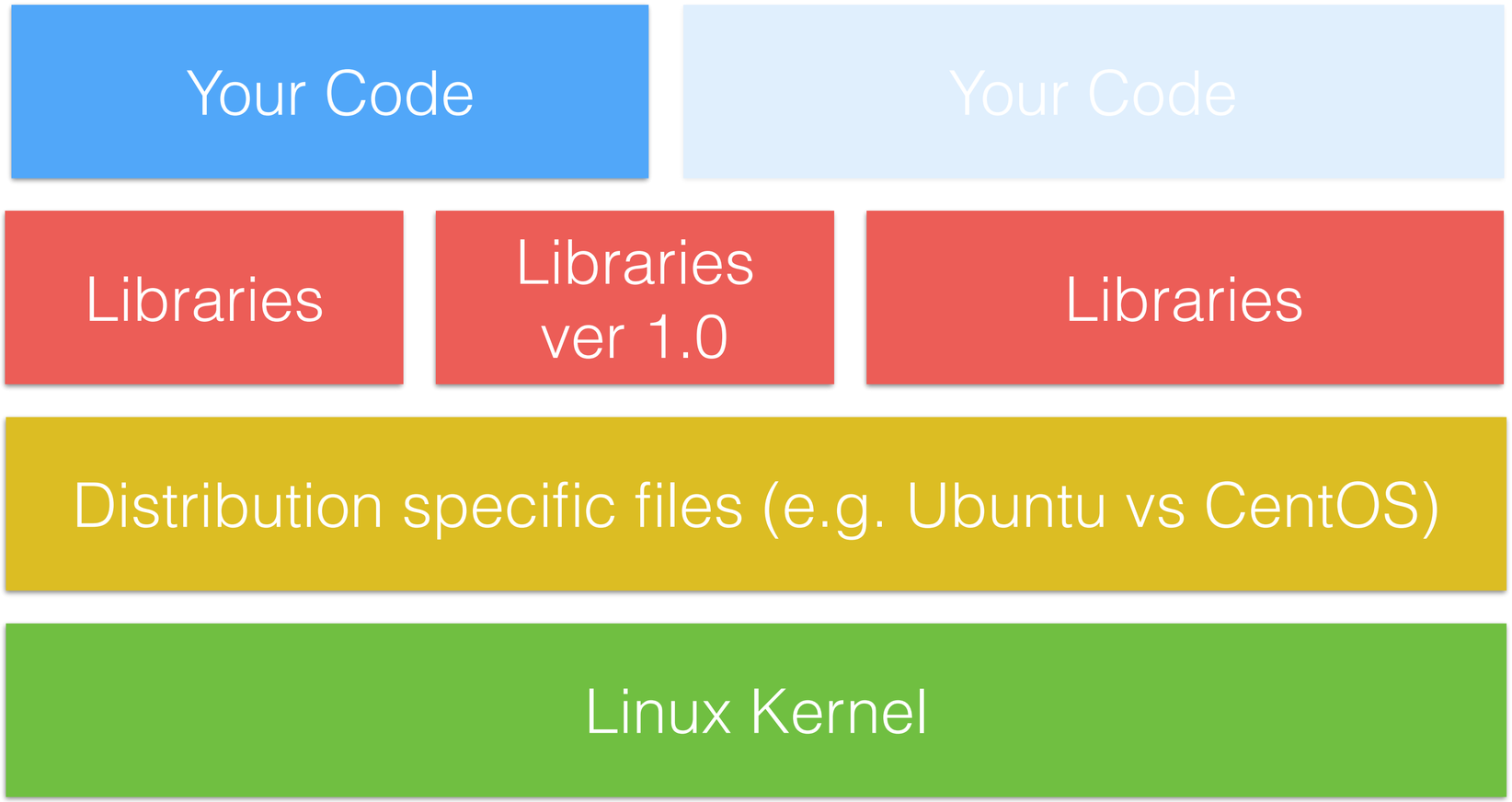


Not maintained by anything!

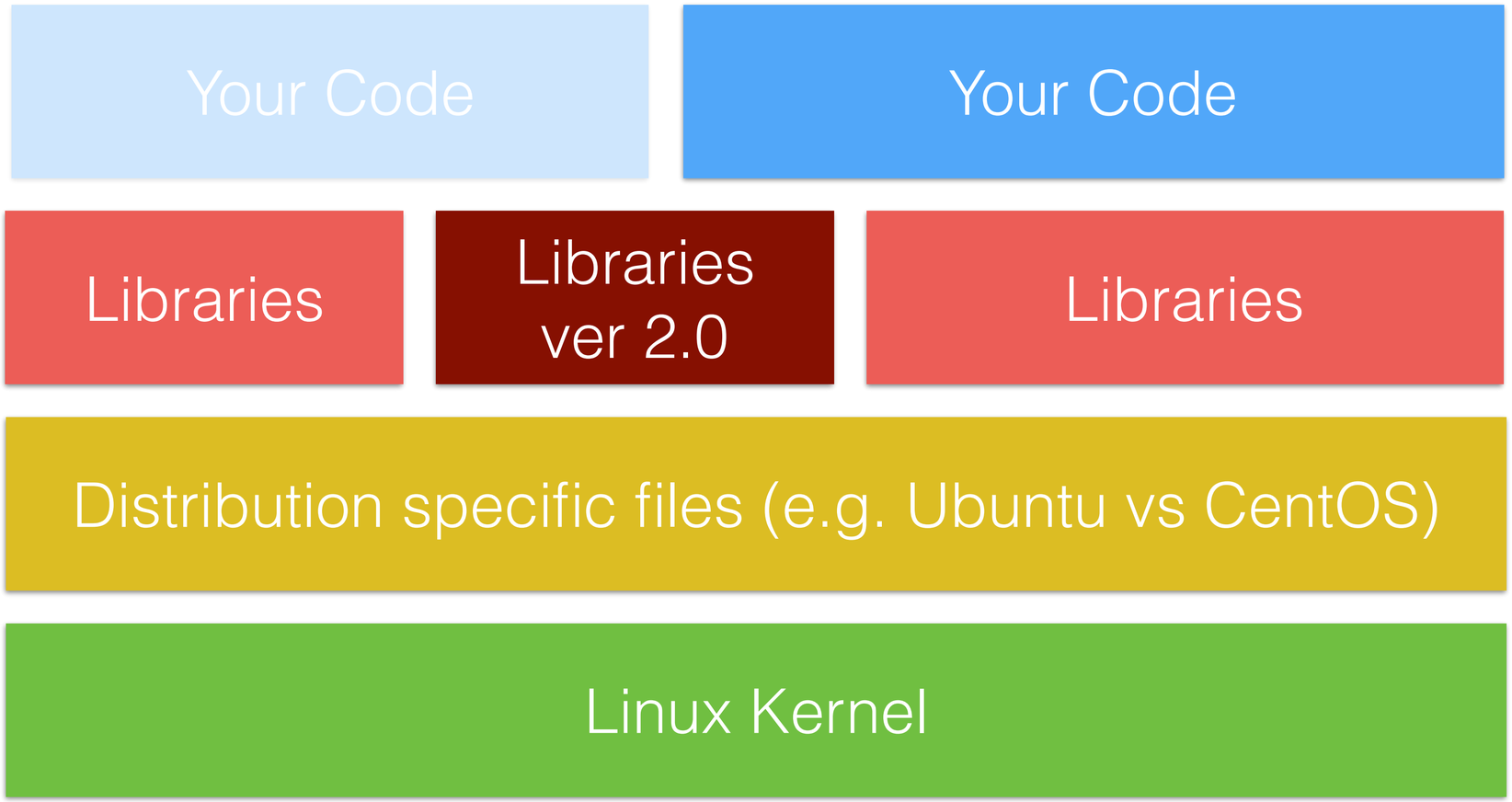
What tends to happen



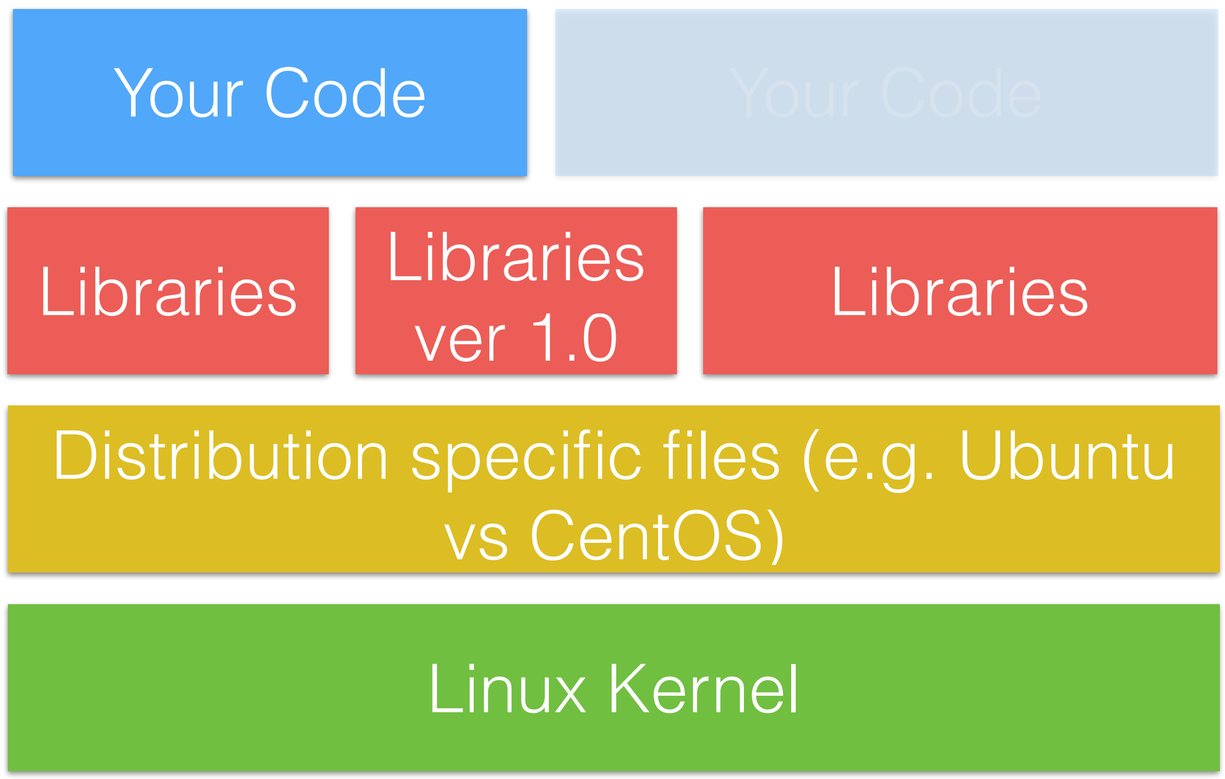
Libraries can conflict

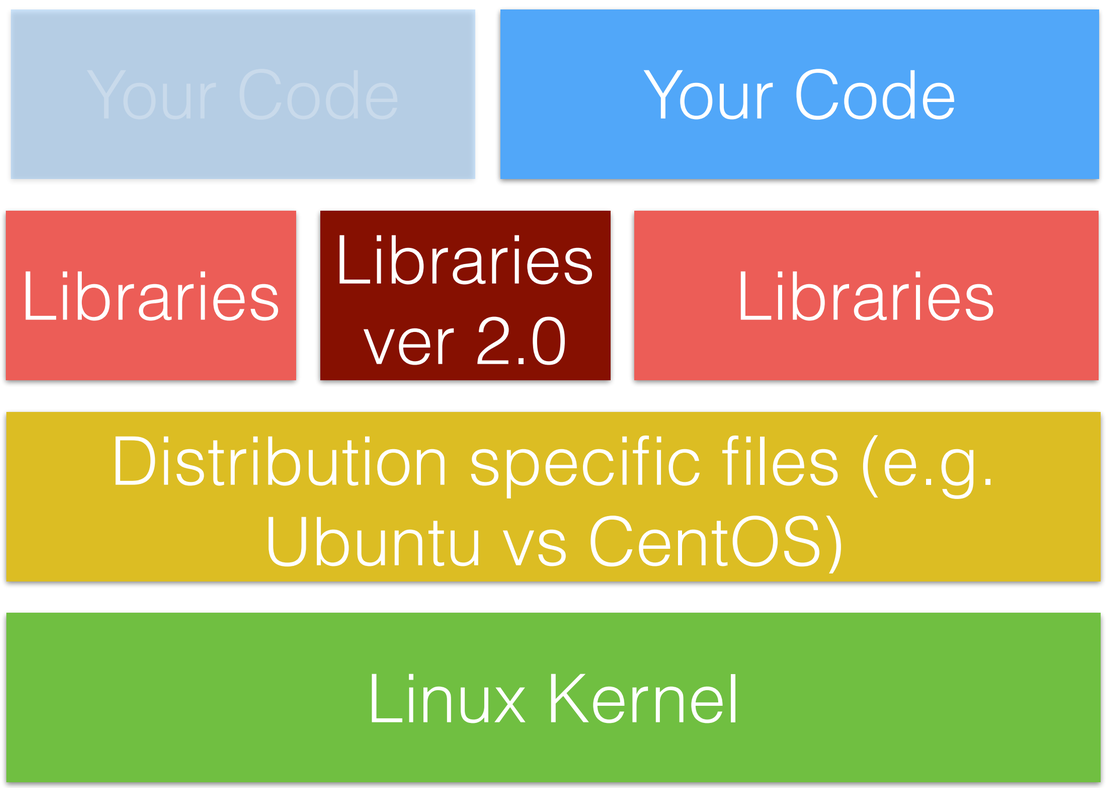


Libraries can conflict

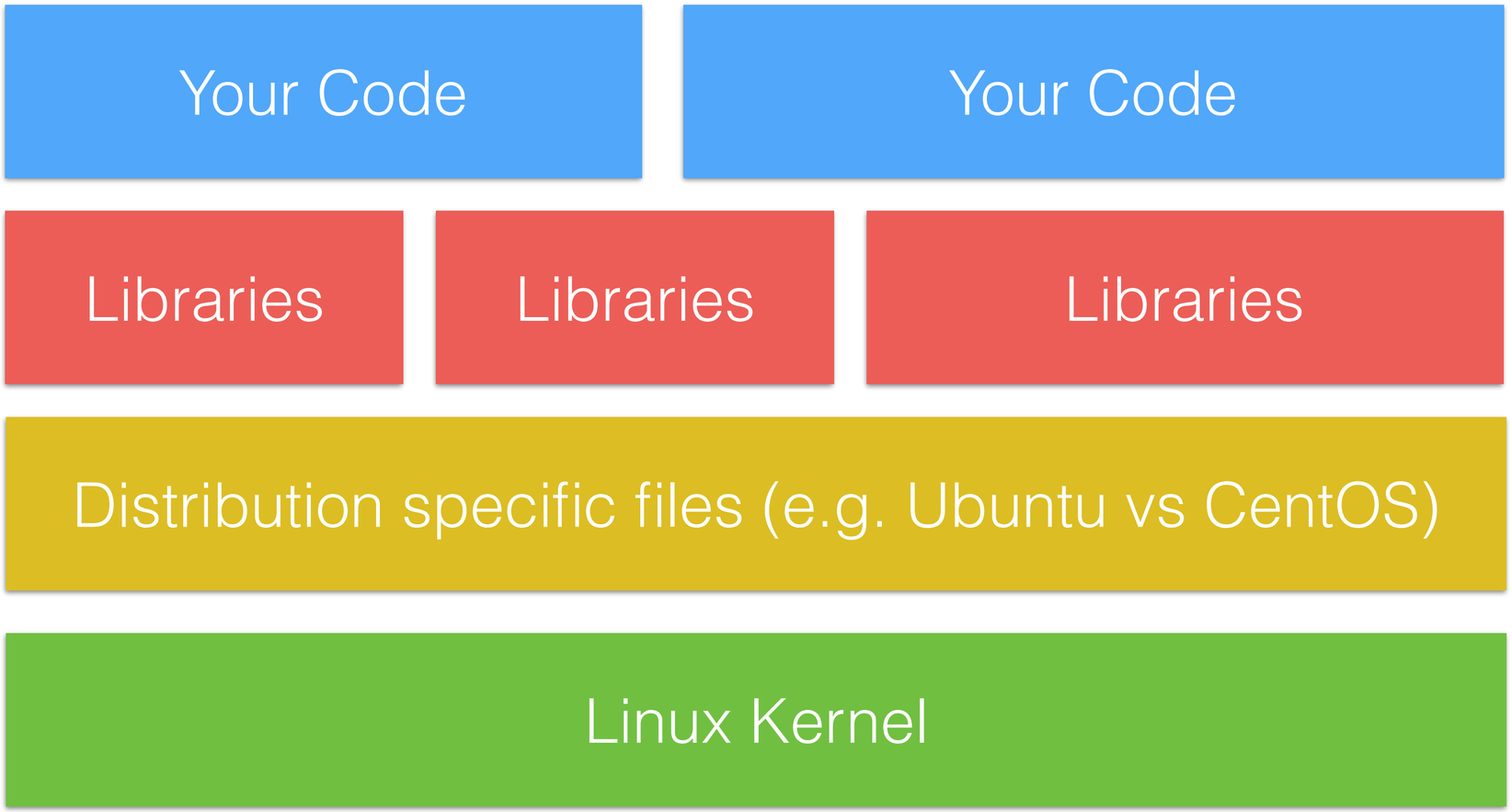


How can they live on a same machine?

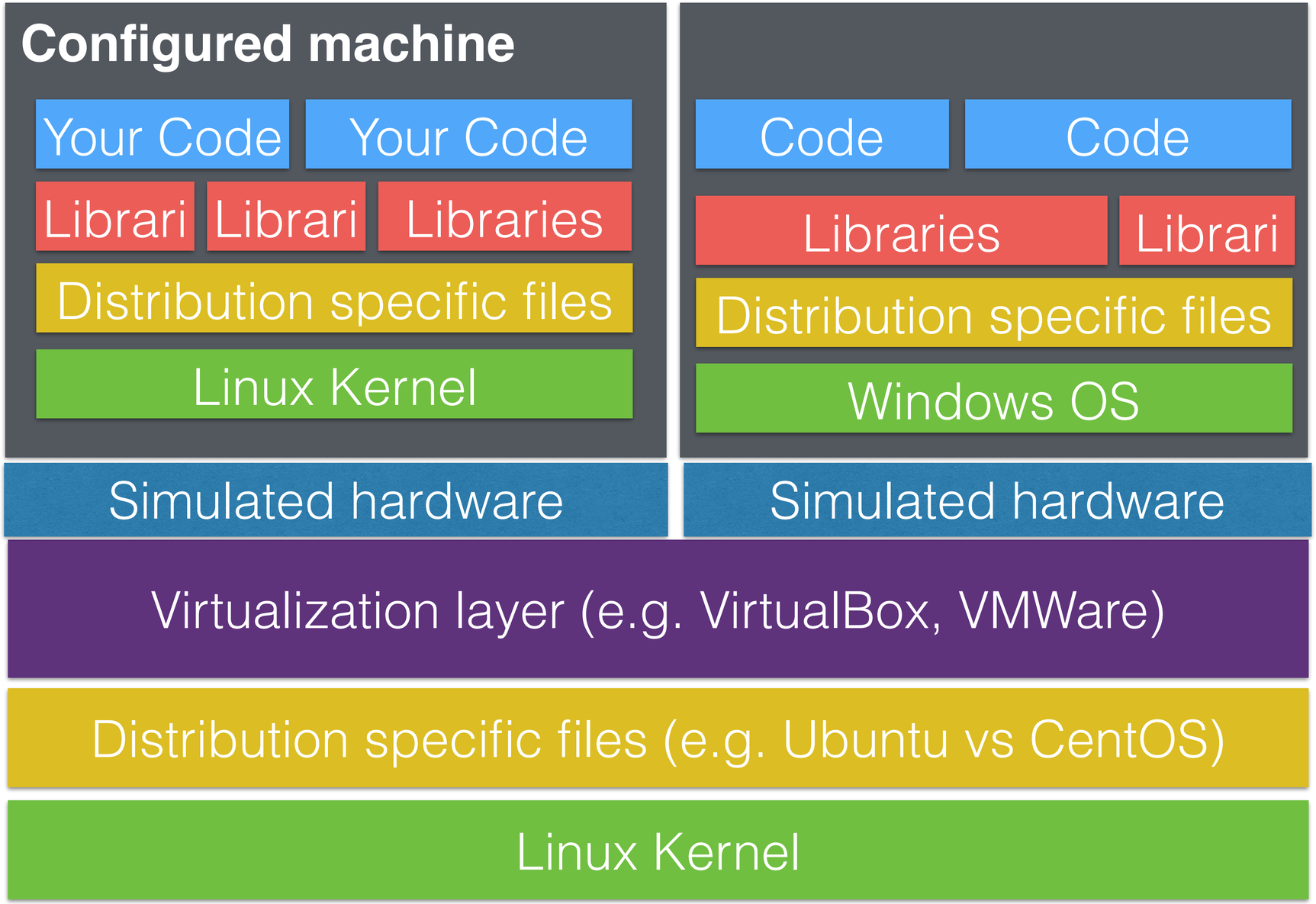




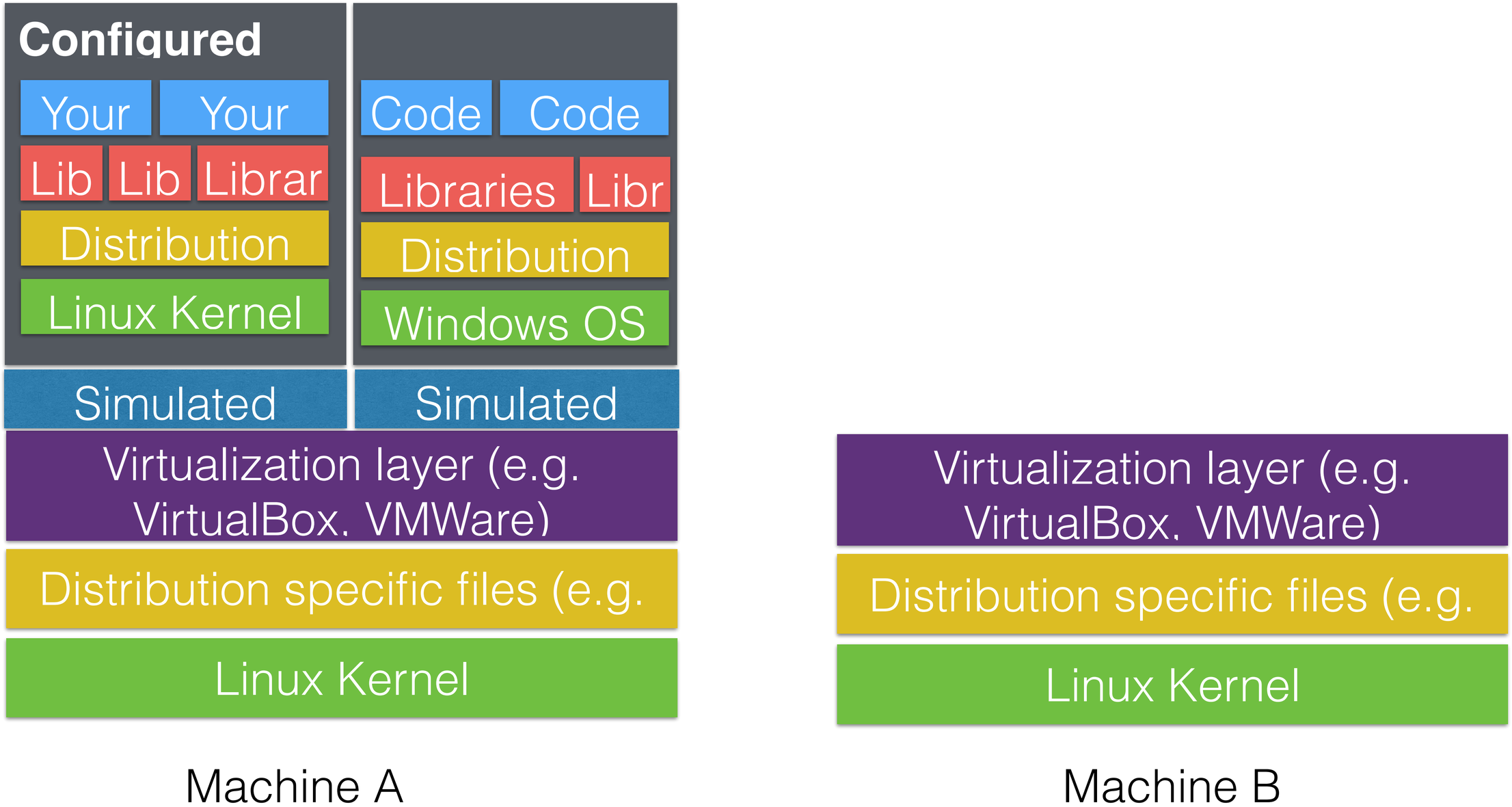
Potential solution: Virtual Machines



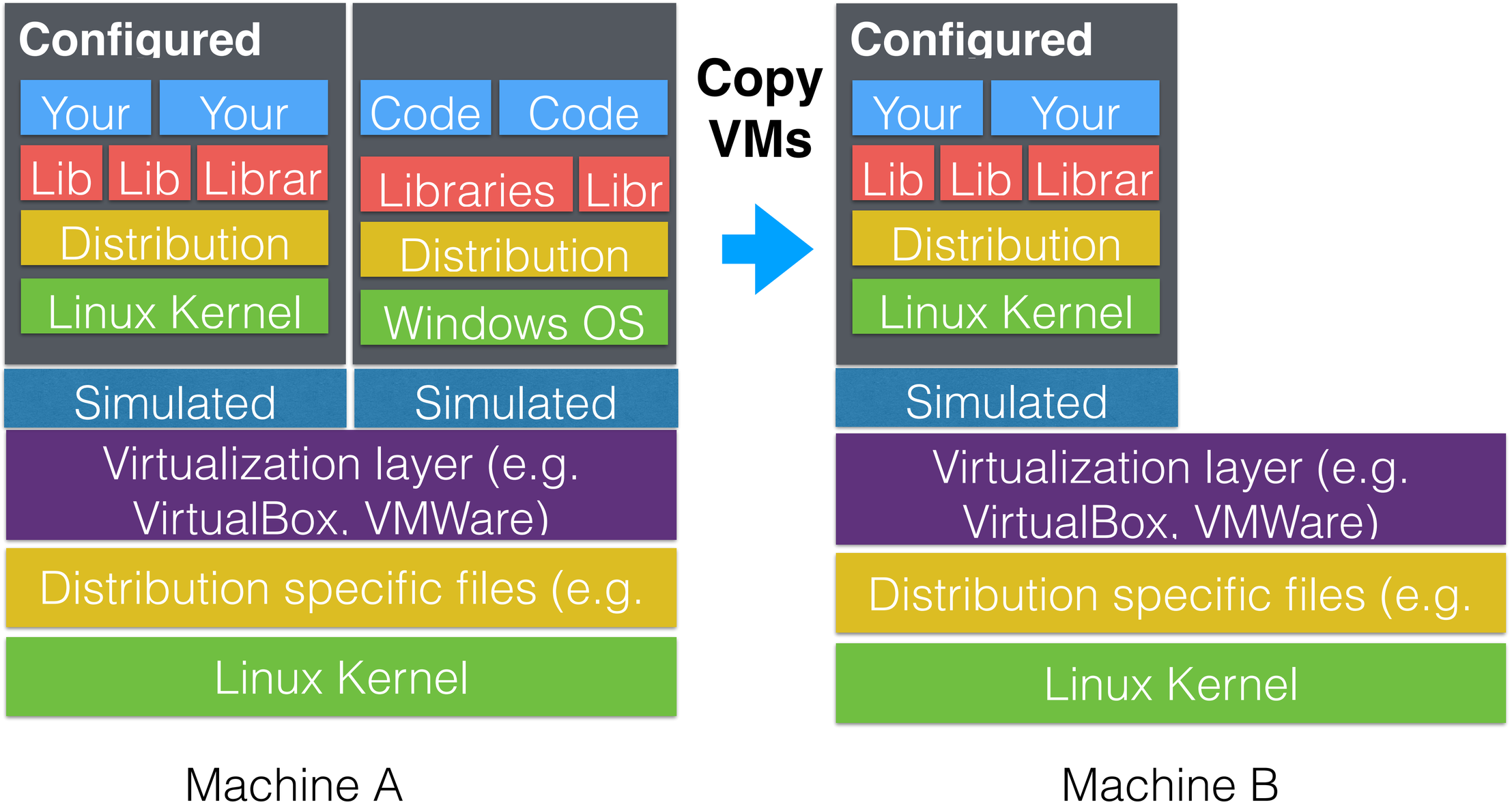
Potential solution: Virtual Machines



Potential solution: Virtual Machines



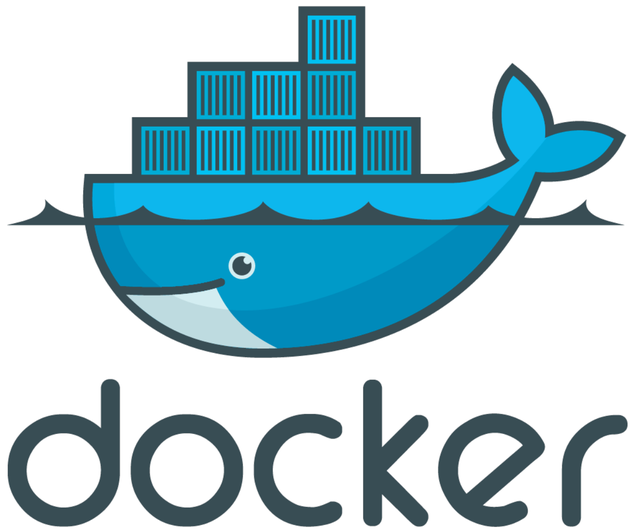
Potential solution: Virtual Machines



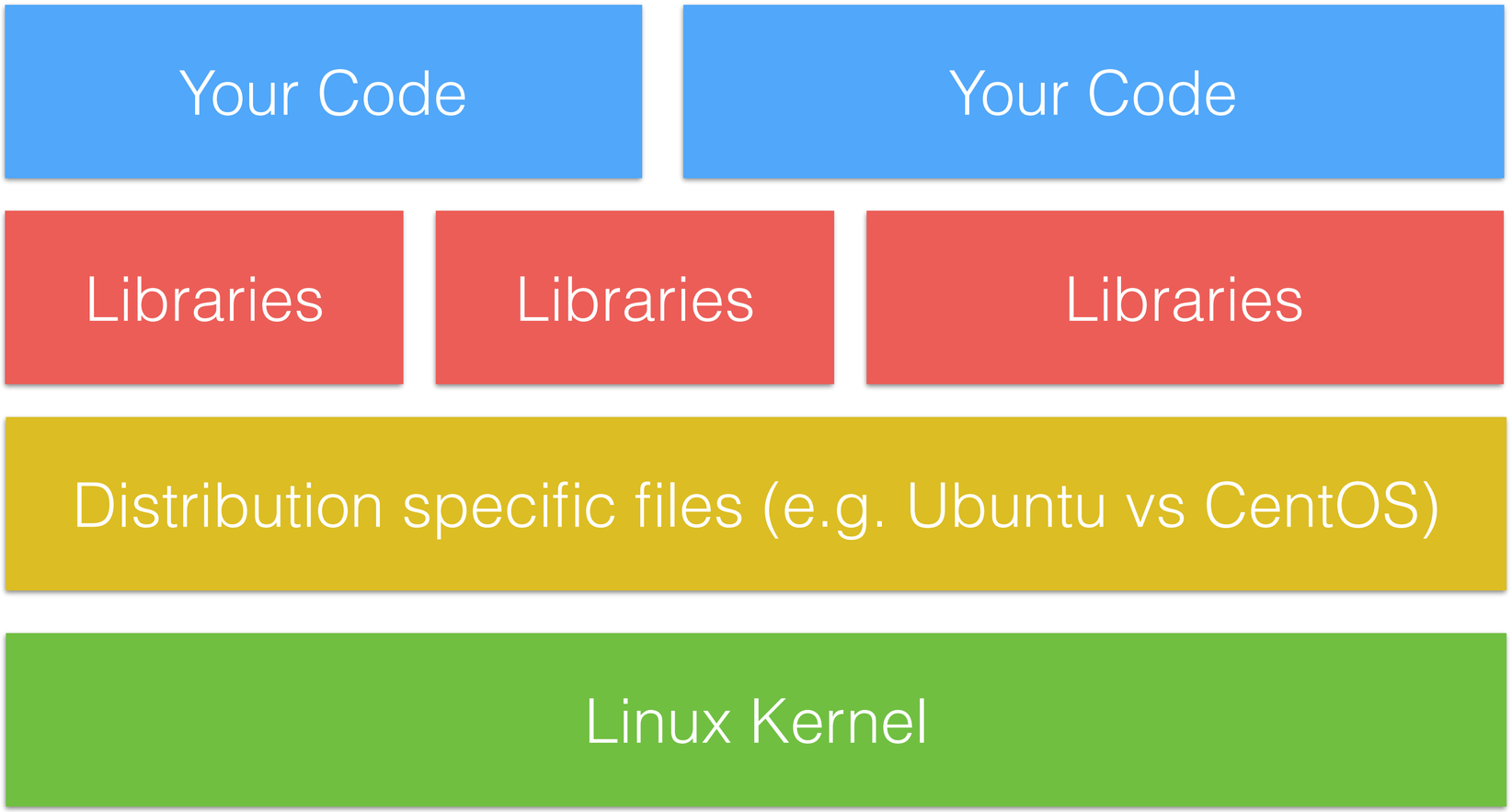
Problem with VMs

- **HEAVY!!**
  - A lot of extra layers causes significant performance drop
- Each virtual machine is also huge and not the easiest thing to share (60~100GB)
- Distribution of virtual machine image becomes an issue

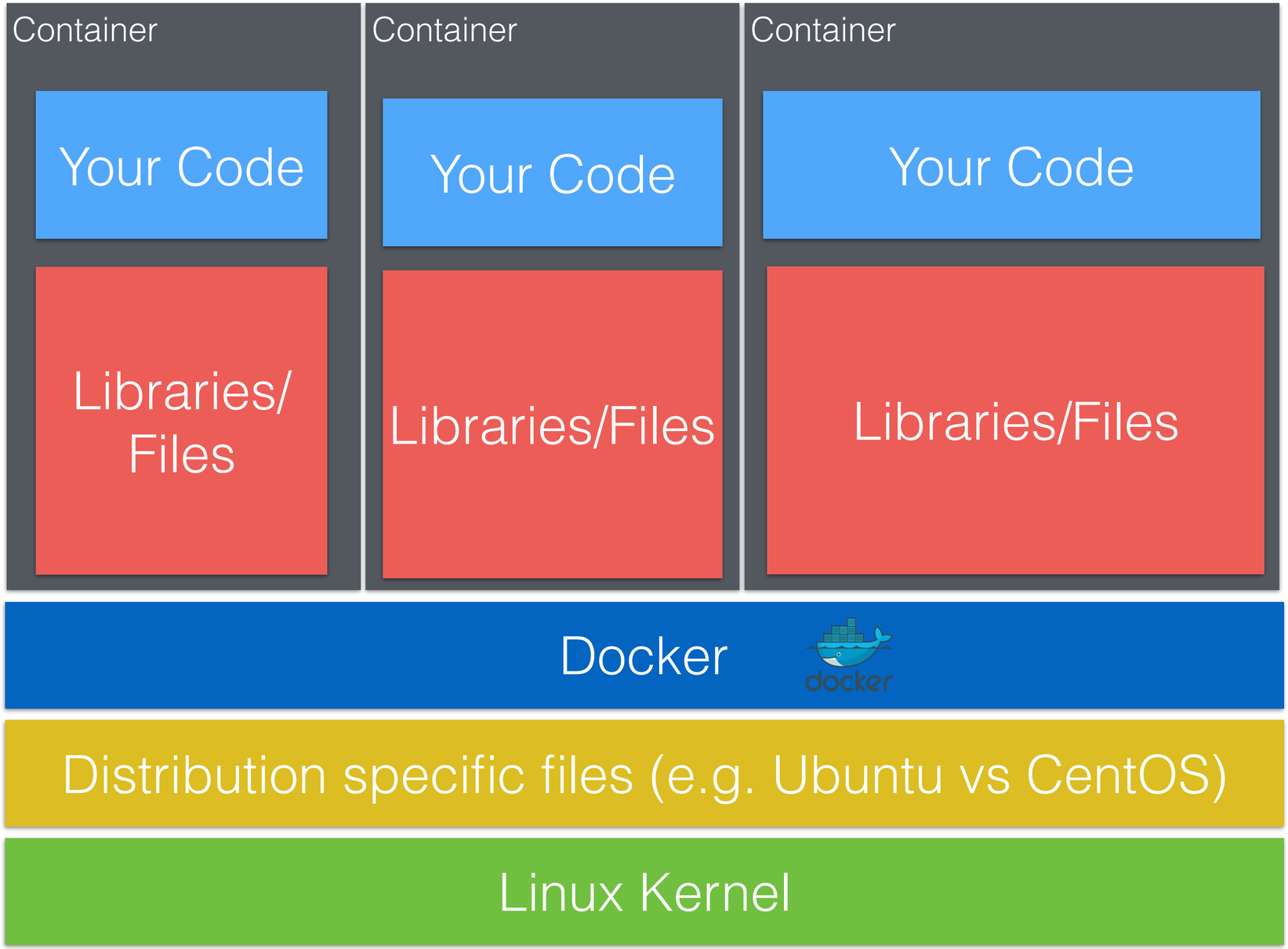
Real solution



The real solution: Docker!



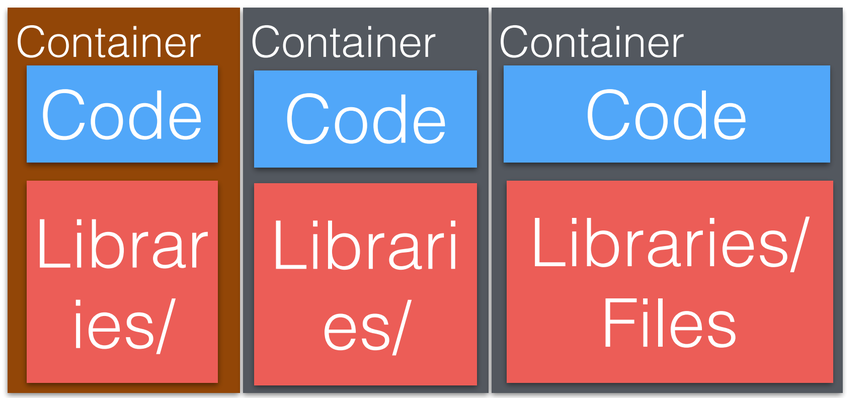
The real solution: Docker!



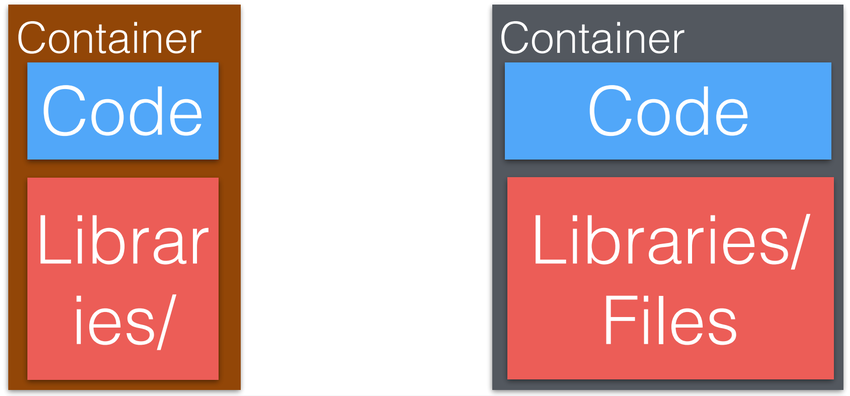
Docker is the way to go!

- Significantly fewer layers needed
- Your code is packaged with all the libraries and files it needs as a **container**
- You can easily distribute the **images of the container**
- Container image is relatively lightweight (typically 200-600MB, with “big” ones ~1GB)

You share containers across machines

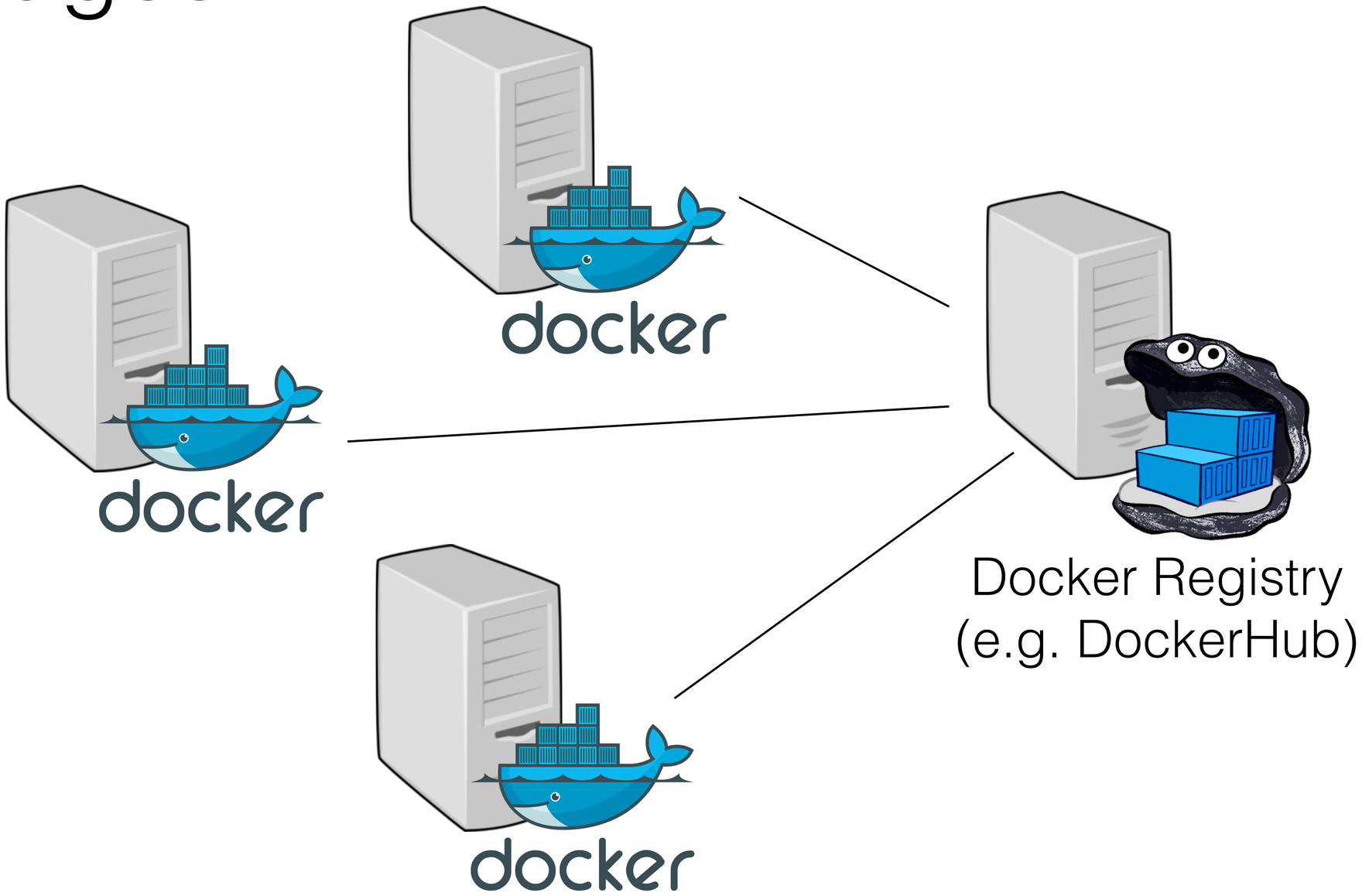


Machine A

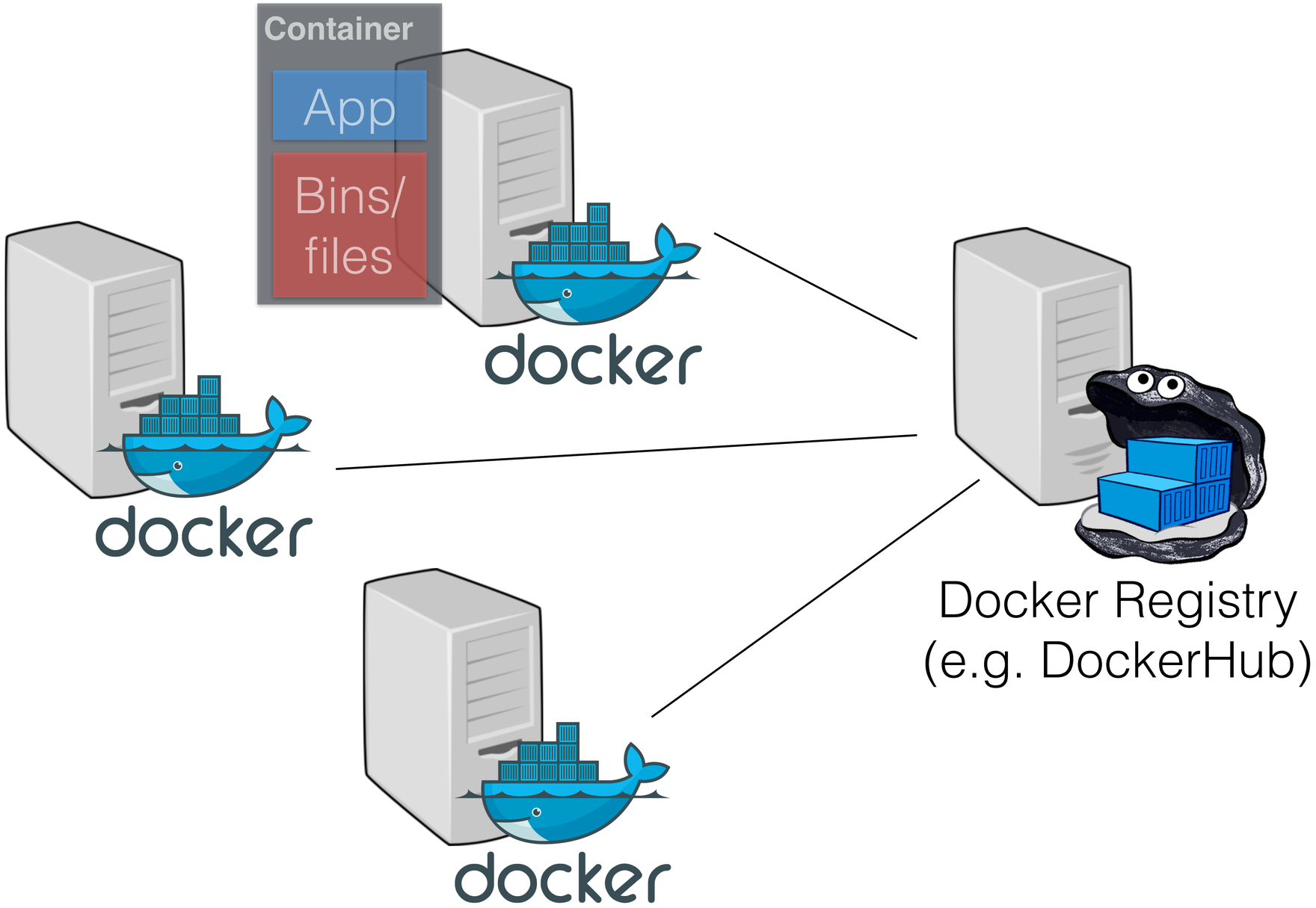


Machine B

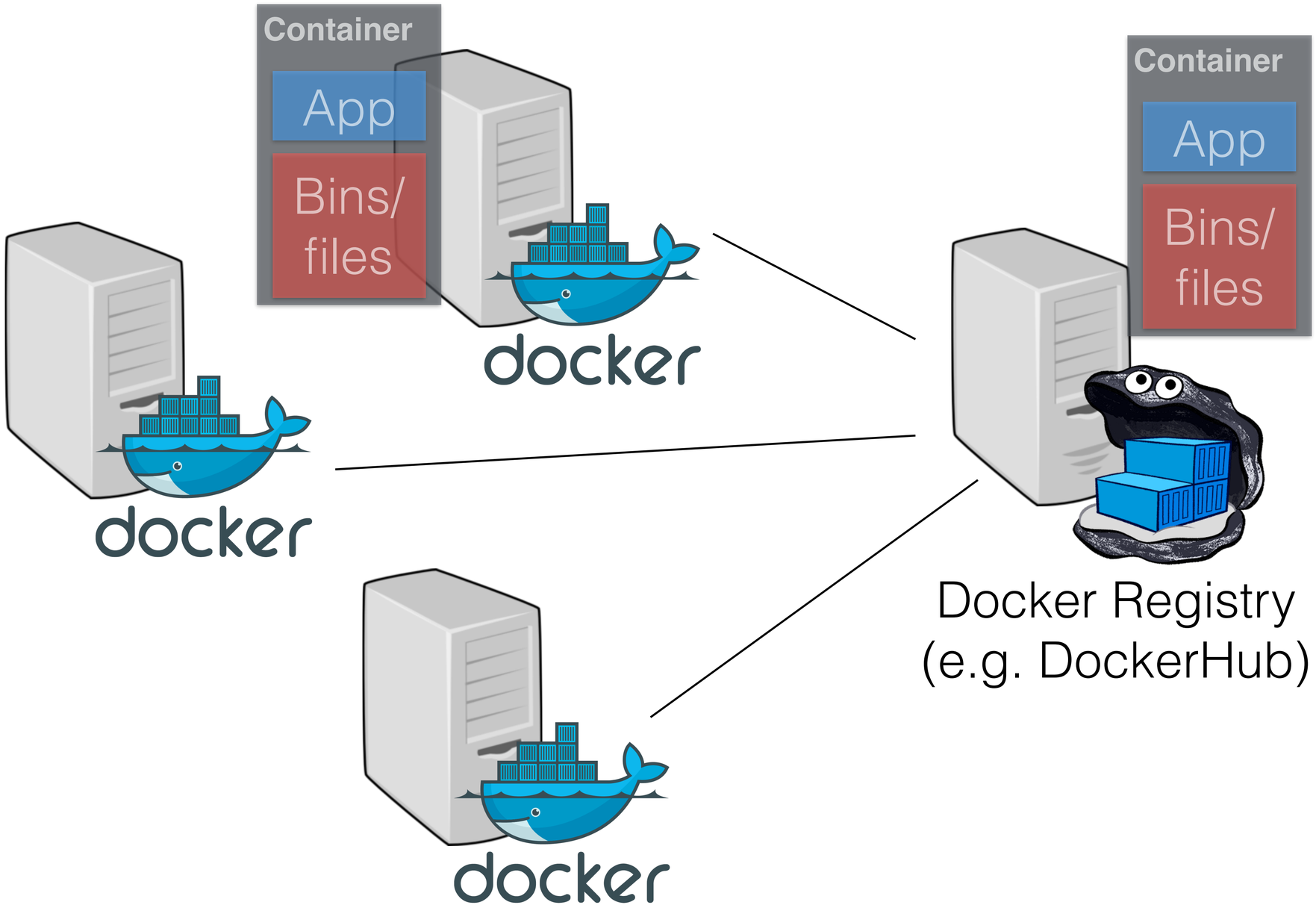
Distribution of Docker Images



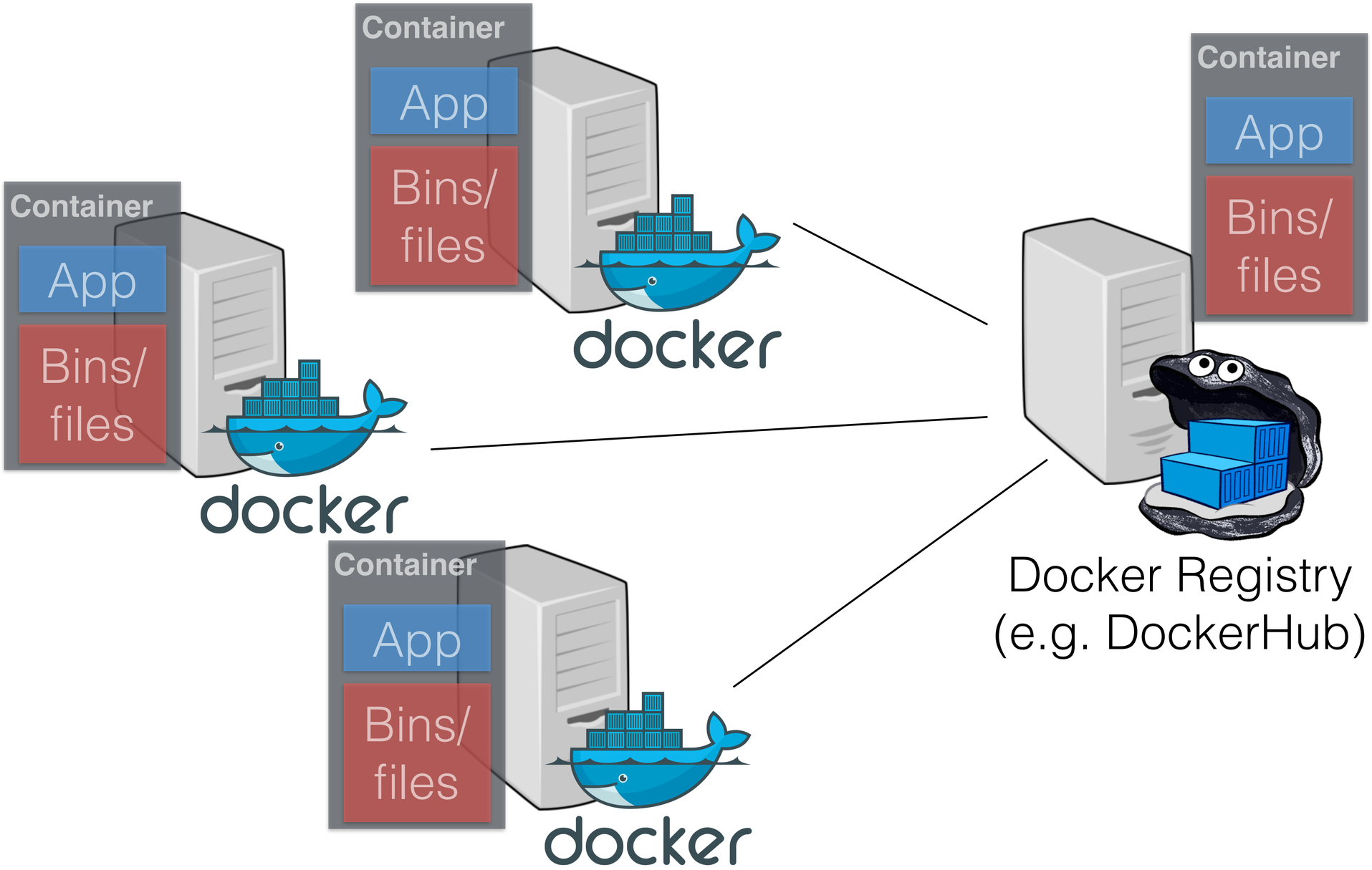
One machine “builds” the image



Push the image to the registry

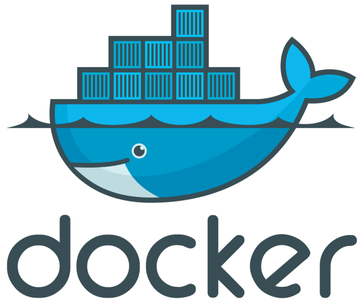


Other machines pull the image



Sharing data, code, and environment

- DataJoint let's you share data easily
- Git/GitHub let's you share and maintain code
- Docker let's you share the whole environment for running you code



DataJoint

Recap of Day 2

We covered even more!

- Practiced designing a data pipeline in a team based on project requirements
- Learned to explore an existing data pipeline and extend it with new analysis
- Discussed best practices in computations and in sharing data, code and environments
  - Looked at tools/technologies to support this
- Learned up-and-coming developments in DataJoint

But this is only the beginning…

- Now it’s your turn to start using DataJoint!
- Nothing teaches you a tool better than using it in your own project
- We would be thrilled to learn how you are going to use DataJoint and would be happy to provide any help!
- Joint the community of DataJoint users by joining our Slack group

- Sign up at DataJoint.io to receive free a tutorial database account and to receive email news letters for future workshop and other announcements about DataJoint

- Documentation and tutorials are available at https://docs.datajoint.io and https://tutorials.datajoint.io

- More learning resources are up and coming!

NeuroNex DataJoint Workshop

NeuroNex Innovation Award
#1707359

- All workshop materials including Jupyter notebooks and presentation slides are made available on our GitHub repository: https://github.com/datajoint/neuronex_workshop_2018

Thank you!!