Git with it!
collaborative coding + version control with git

Dr. Fonti Kar + Robert Brady

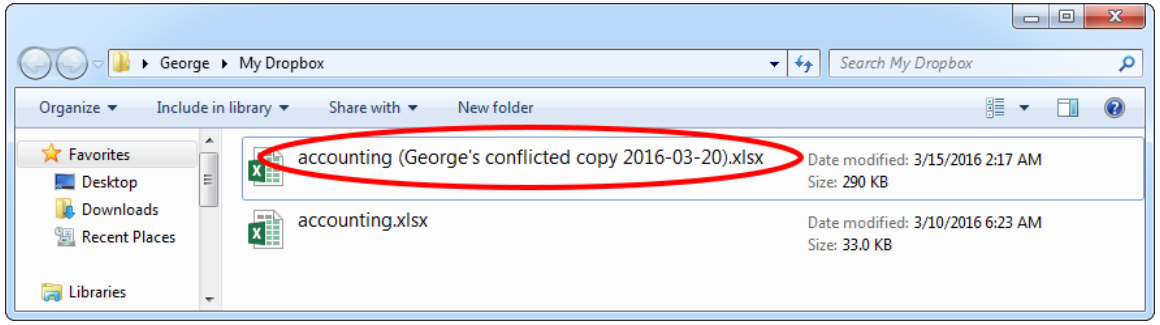
Illustrations by Allison Horst

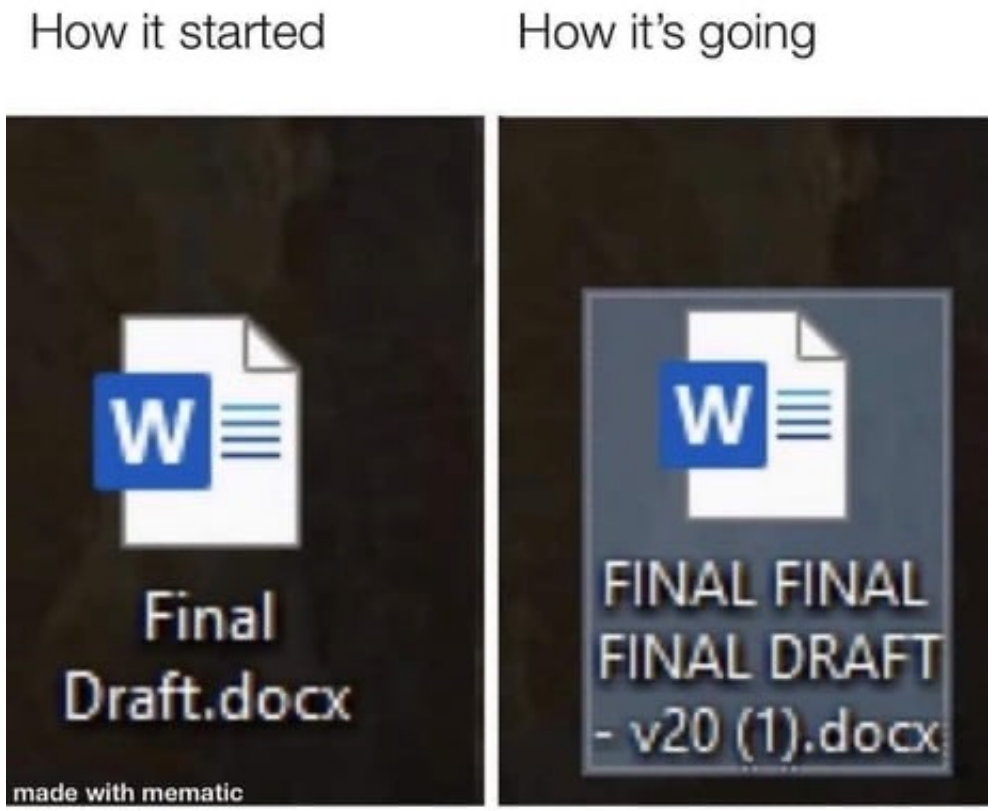
Overview

- Why git?
- Some new lingo
  - Repository
  - Add and Commit
  - Push and Pull
  - Branches
- Some demos
- Some follow along

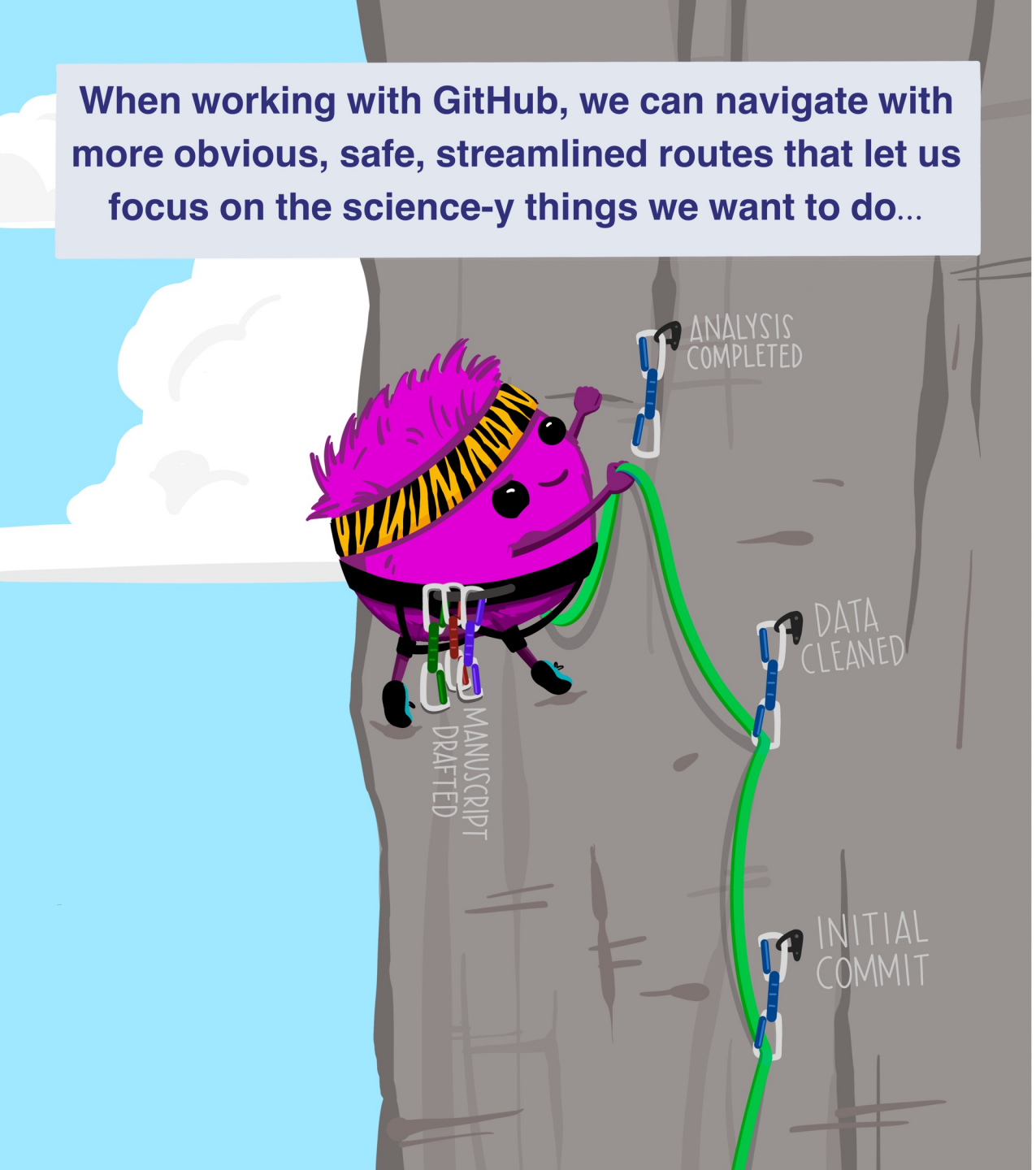
Why git?

- Tracks changes in complex code projects over time
- Keeps project directories tidy
- Allows for safe exploration without effecting existing work

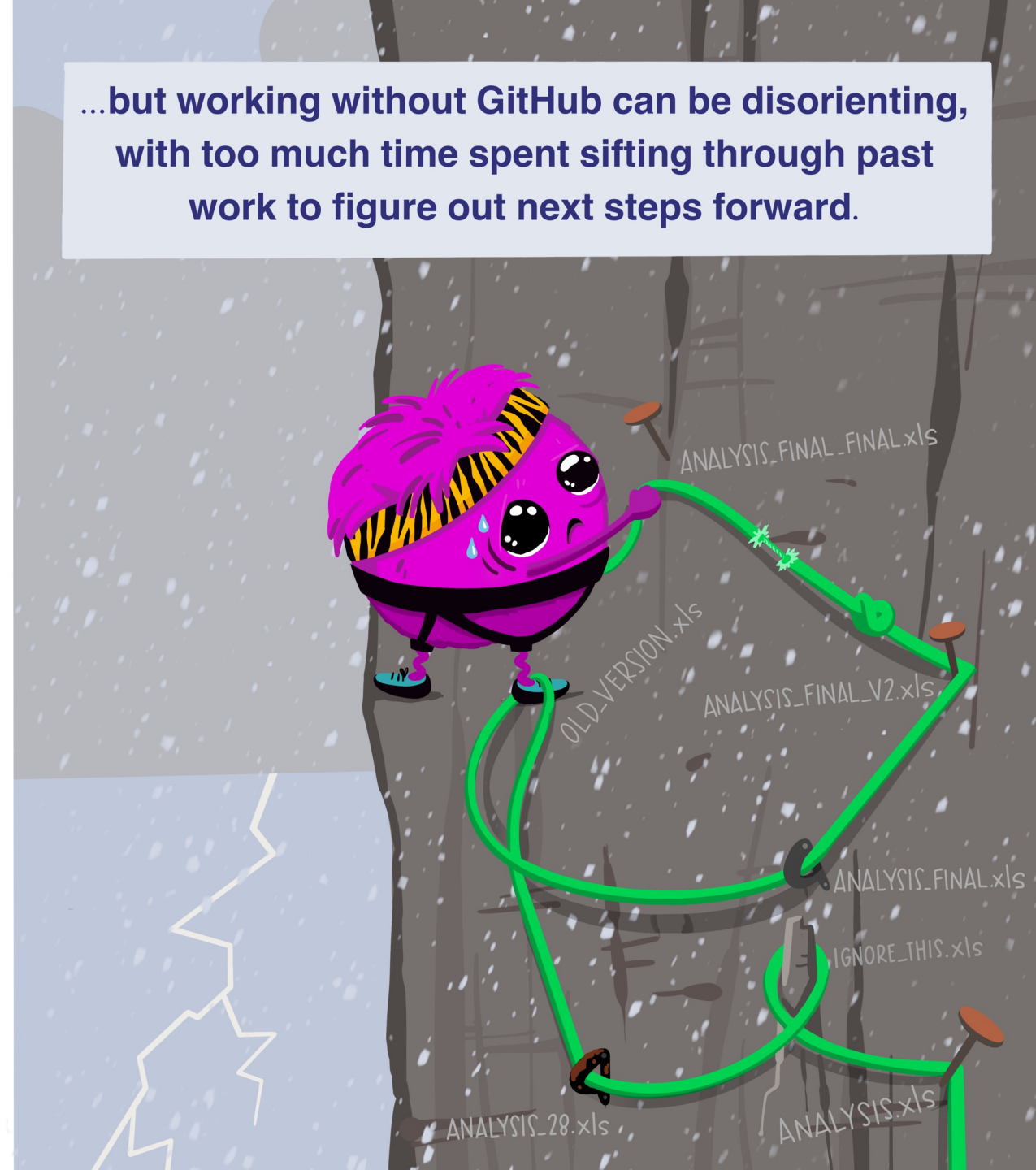




When working with GitHub, we can navigate with more obvious, safe, streamlined routes that let us focus on the science-y things we want to do...



...but working without GitHub can be disorienting, with too much time spent sifting through past work to figure out next steps forward.



Some new lingo

REPOSITORY

- AKA “repo”
- Two kinds:
  - Local – on your computer
    - Tracking your working directory
  - Remote – on a server
    - Public i.e. open source
    - Private

REPOSITORY

- AKA “repo”
- Two kinds:
  - Local – on your computer
    - Tracking your working directory
  - Remote – on a server
    - Public i.e. open source
    - Private

REPOSITORY

- AKA “repo”
- Two kinds:
  - Local – on your computer
    - Tracking your working directory
  - Remote – on a server
    - Public i.e. open source
    - Private

- Adding files
  - AKA "staging”
  - Initiates tracking on specific files

ADD AND COMMIT

ADD AND COMMIT

- Adding files
  - AKA "staging”
  - Initiates tracking on specific files
- Commit
  - An anchor of your changes
  - Include a helpful message

- Adding files
  - AKA "staging”
  - Initiates tracking on specific files
- Commit
  - An anchor of your changes
  - Include a helpful message

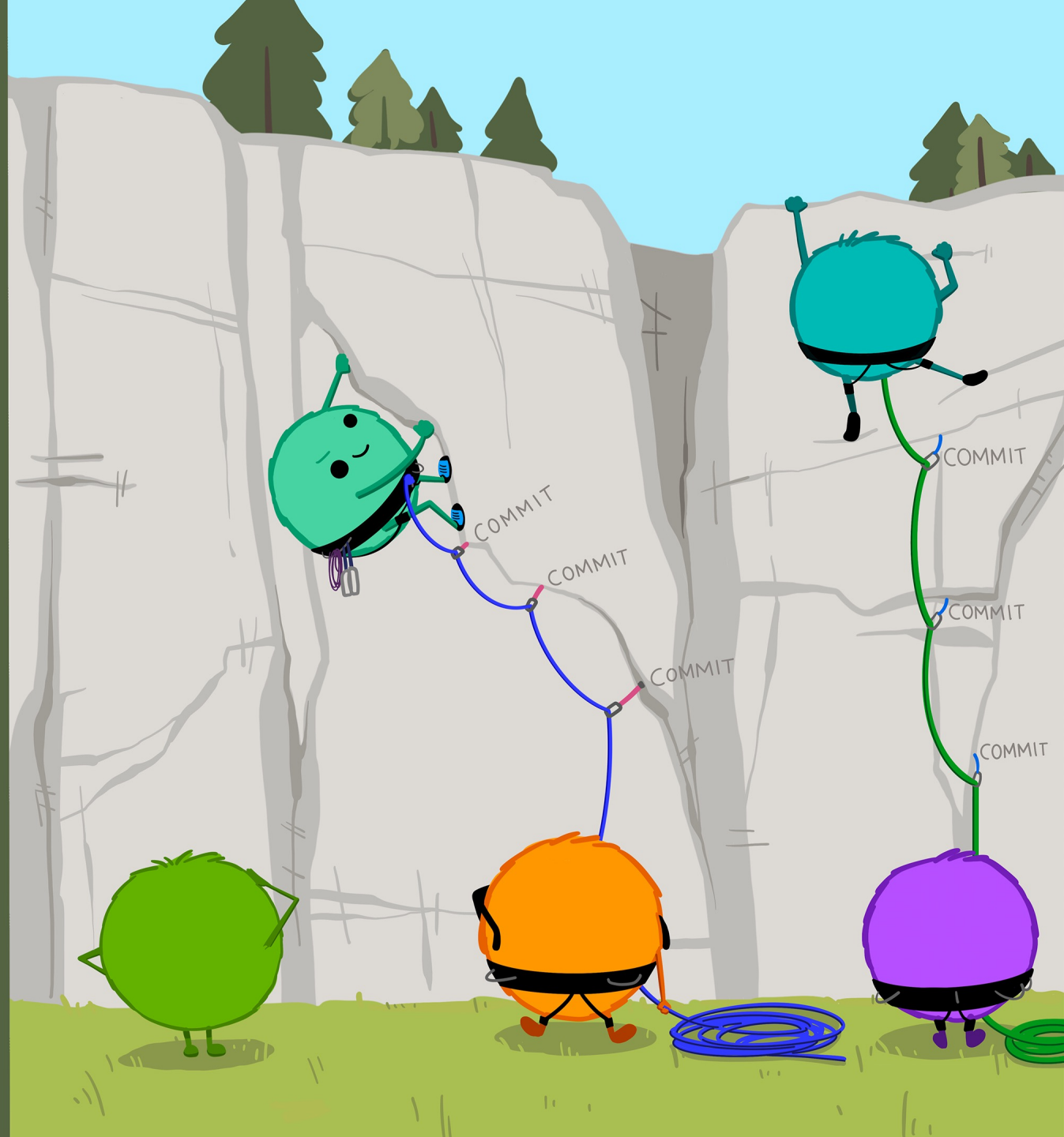
ADD AND COMMIT

- Adding files
  - AKA "staging”
  - Initiates tracking on specific files
- Commit
  - An anchor of your changes
  - Include a helpful message

ADD AND COMMIT

> Using a Git commit is like using anchors and other protection when climbing…**if you make a mistake, you can't fall past the previous commit**.
>
> Commits are also helpful to others, because **they show your journey, not just the destination**.

– HADLEY WICKHAM & JENNY BRYAN



Wickham & Bryan, RPackages (https://r-packages.org/preface.html)

- Pushing to server
  - AKA “publishing”
  - Send your work to remote repo i.e. GitHub

PUSH AND PULL

- Pushing to server
  - AKA “publishing”
  - Send your work to remote repo i.e. GitHub

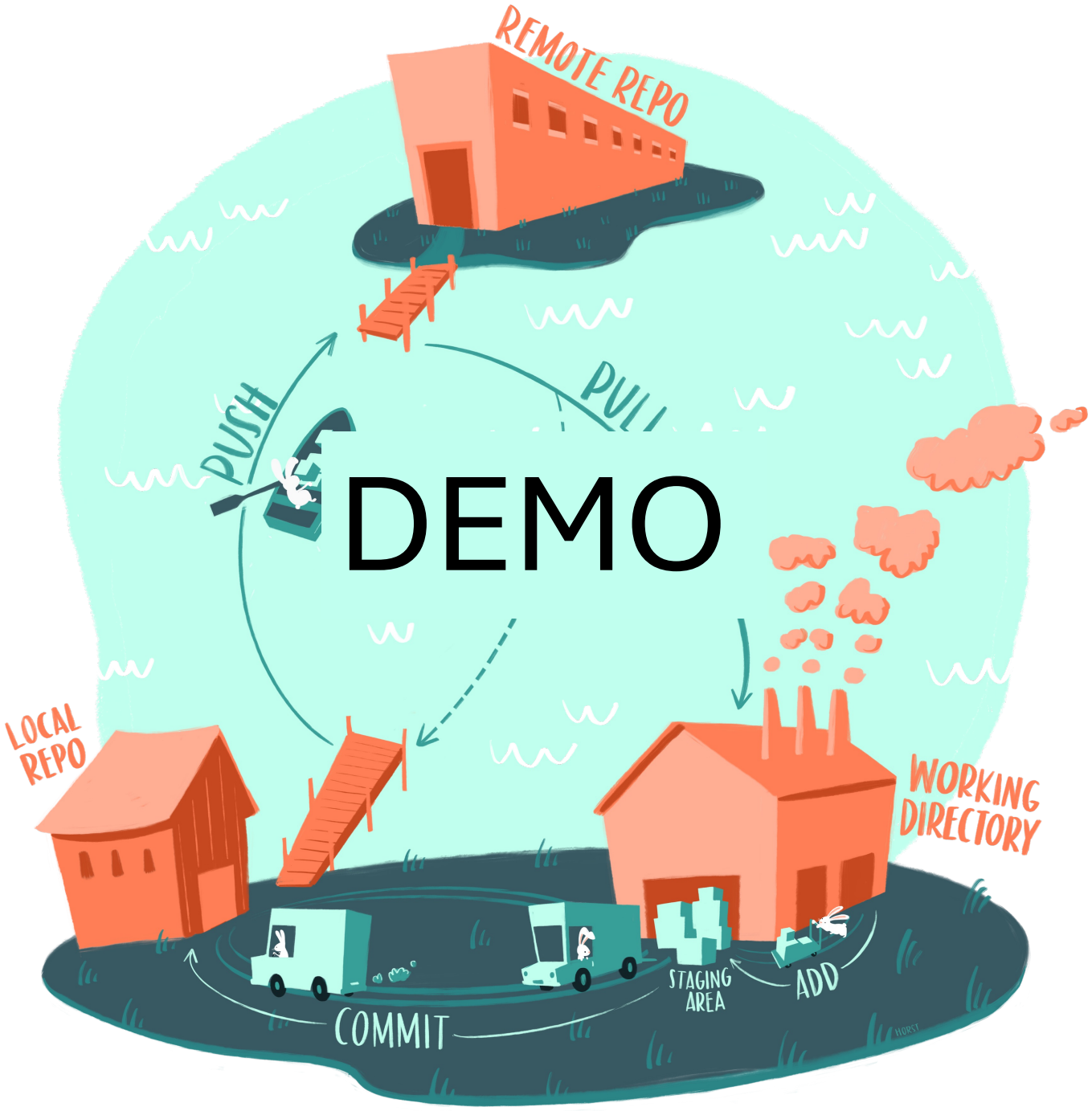
- Pulling from server
  - AKA “fetching”
  - Grab work from remote repo

PUSH AND PULL

- Pushing to server
  - AKA “publishing”
  - Send your work to remote repo i.e. GitHub

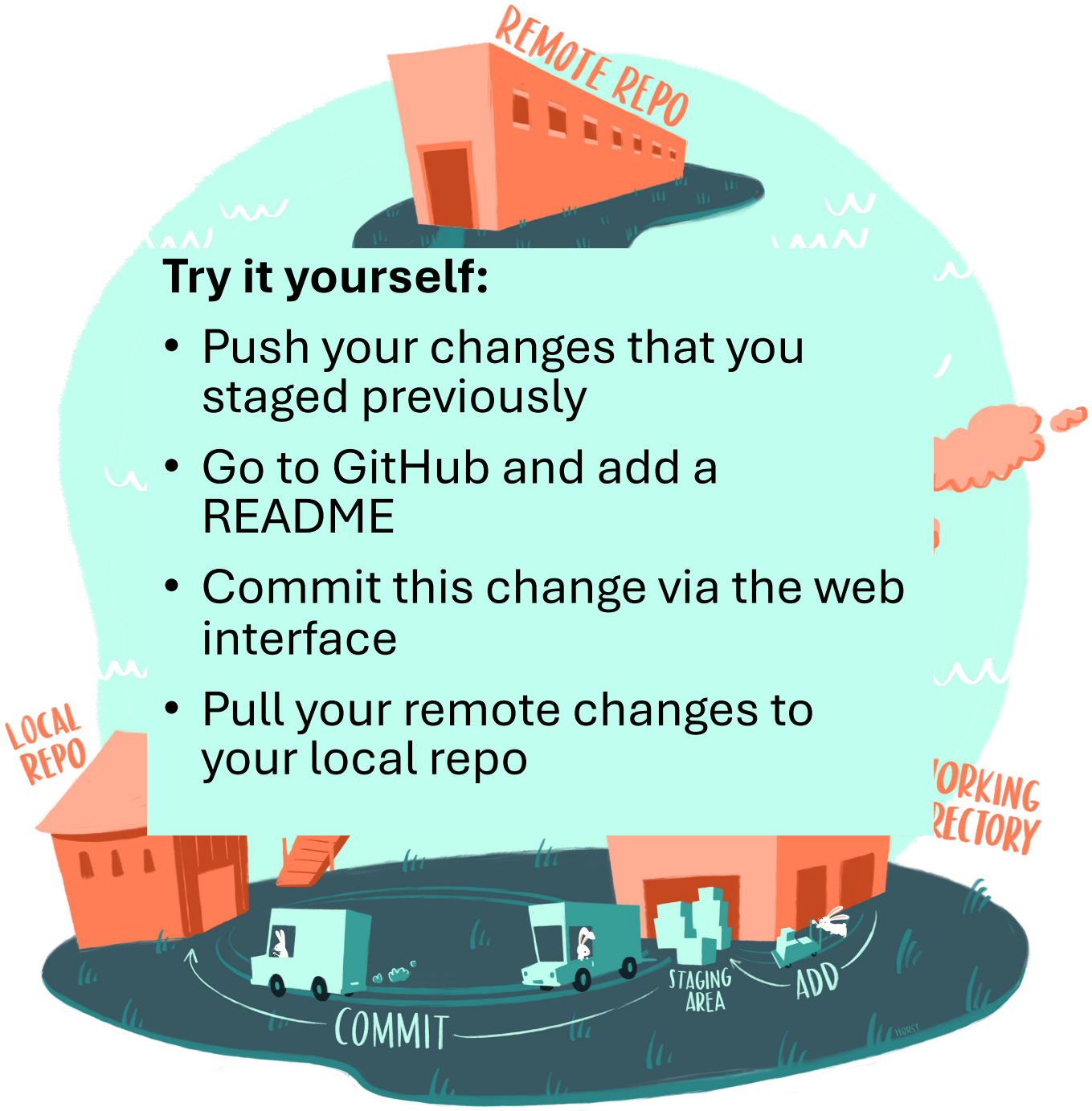
- Pulling from server
  - AKA “fetching”
  - Grab work from remote repo

PUSH AND PULL

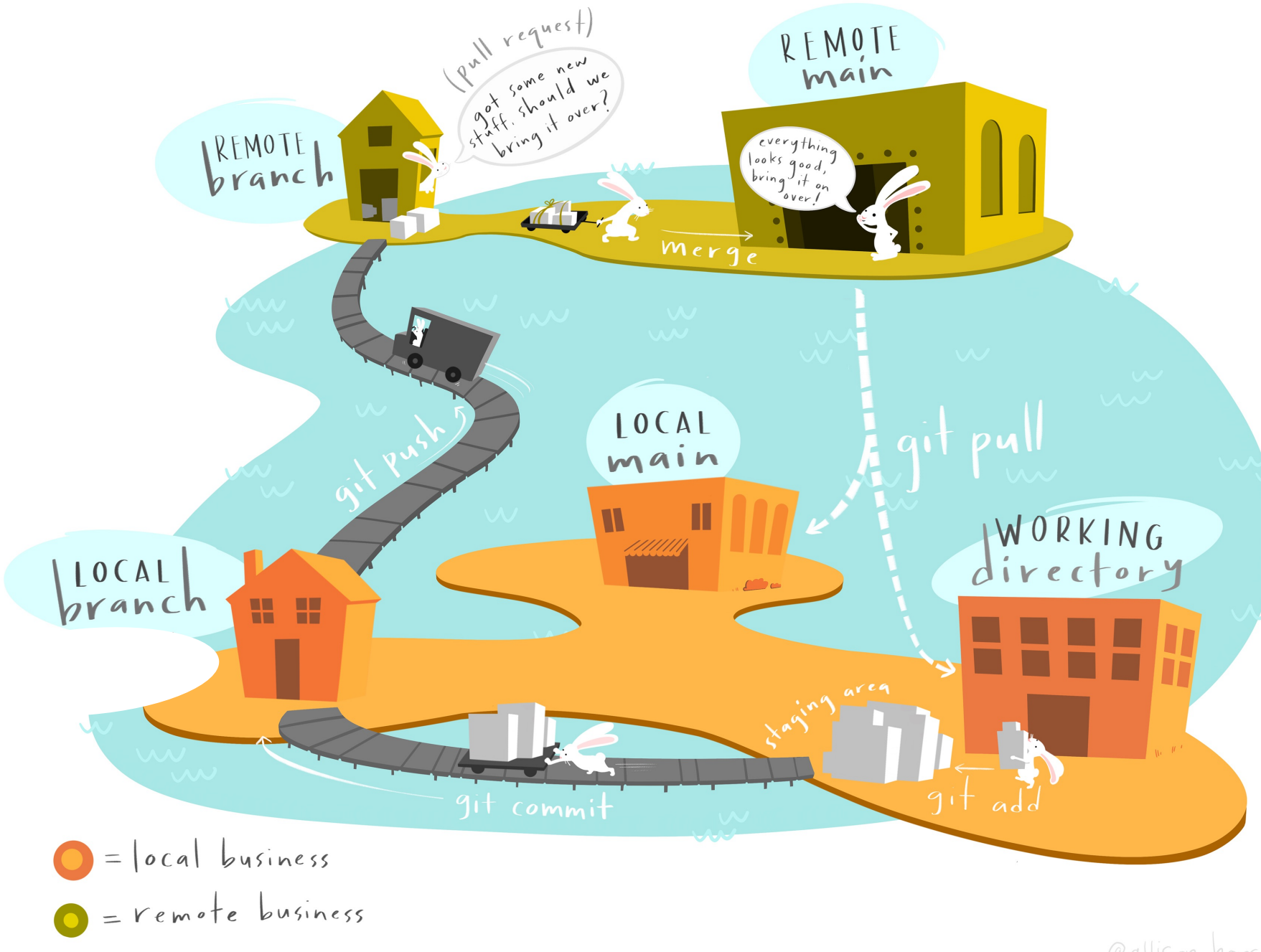


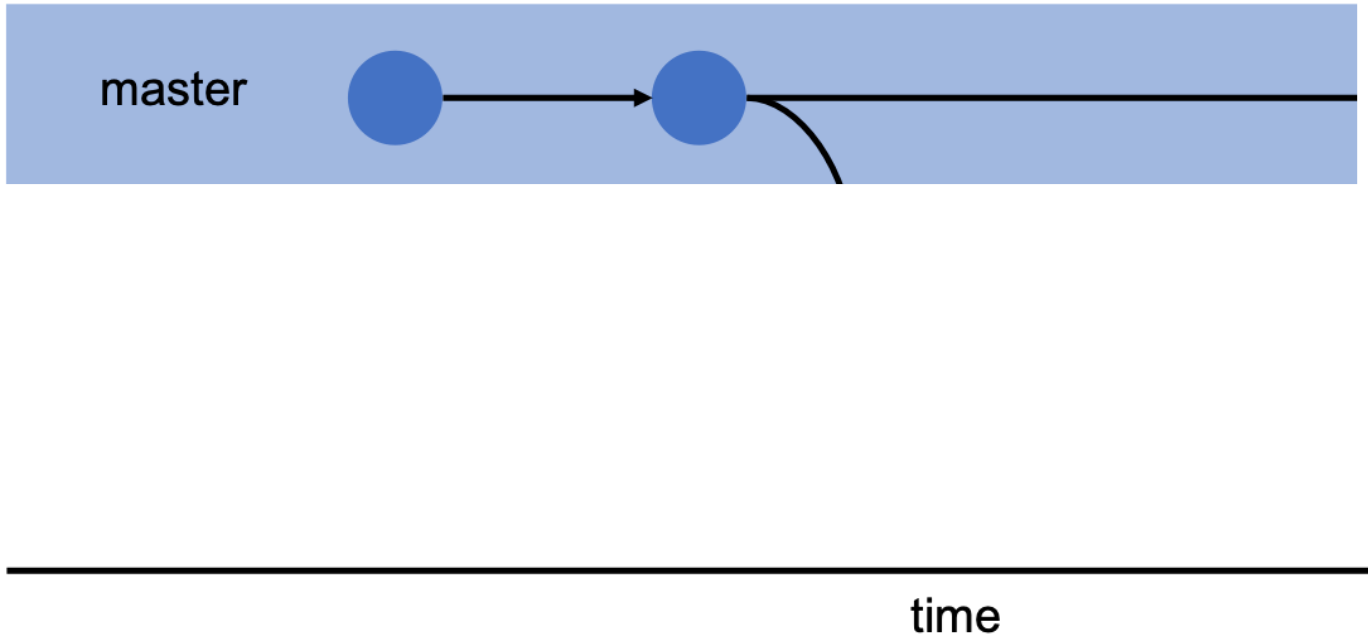
PUSH AND PULL

- Pushing to server
  - AKA “publishing”
  - Send your work to remote repo i.e. GitHub
- Pulling from server
  - AKA “fetching”
  - Grab work from remote repo



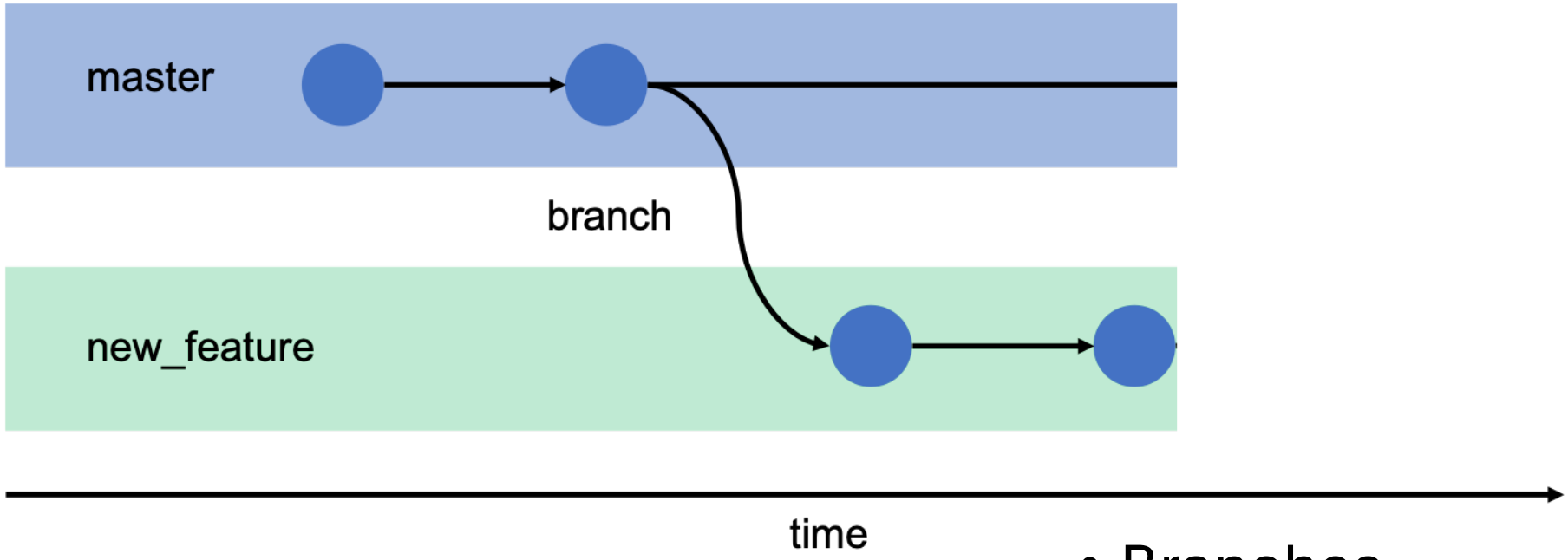
BRANCHES


REMOTE branch
(pull request)
got some new stuff. should we bring it over?
REMOTE main
everything looks good, bring it on over!
merge
git push
LOCAL main
git pull
LOCAL branch
WORKING directory
staging area
git commit
git add
= local business
= remote business
@allison_hors



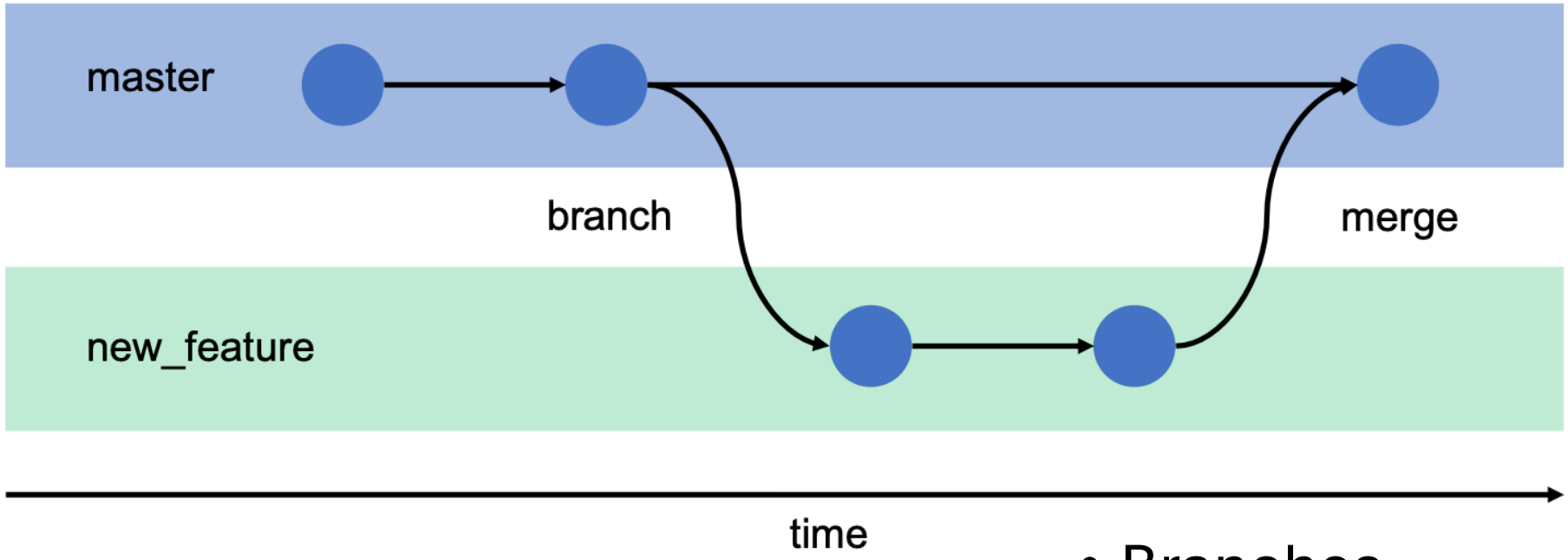
- Main trunk
  - Your “master” copy
  - Always in functional state

BRANCHES



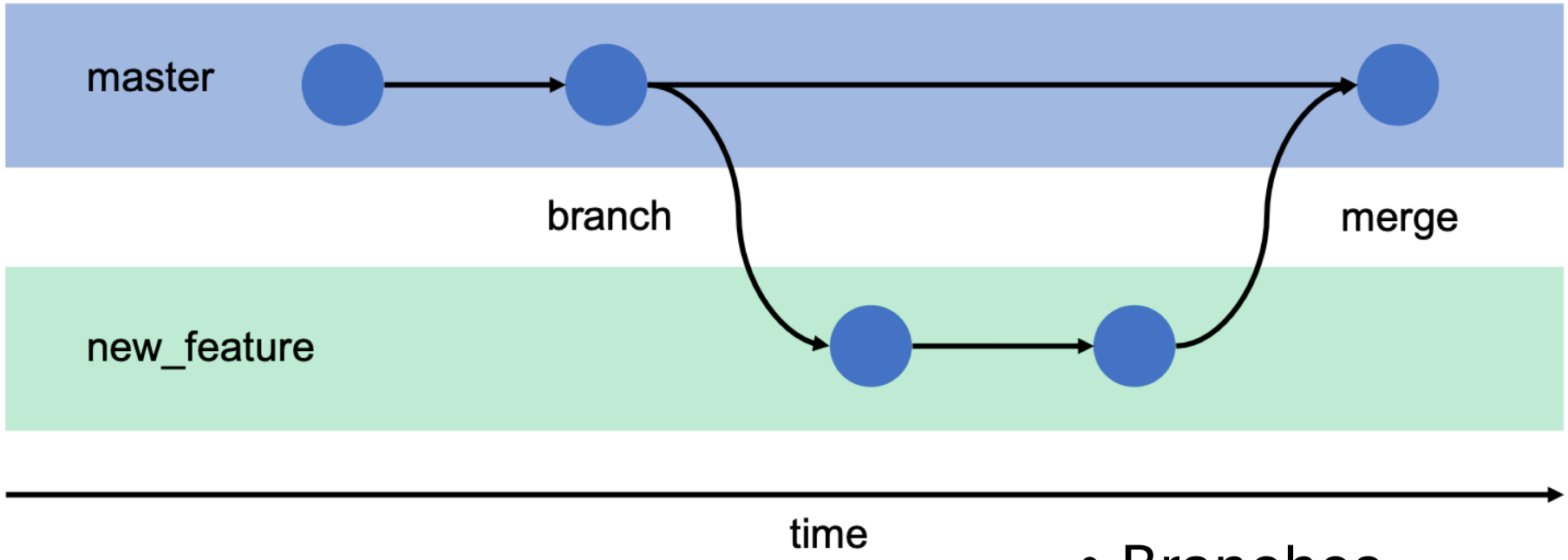
- Main trunk
  - Your “master” copy
  - Always in functional state
- Branches
  - Duplicate(s) of main trunk for new code development
  - Testing environment without different version of same files

BRANCHES



BRANCHES

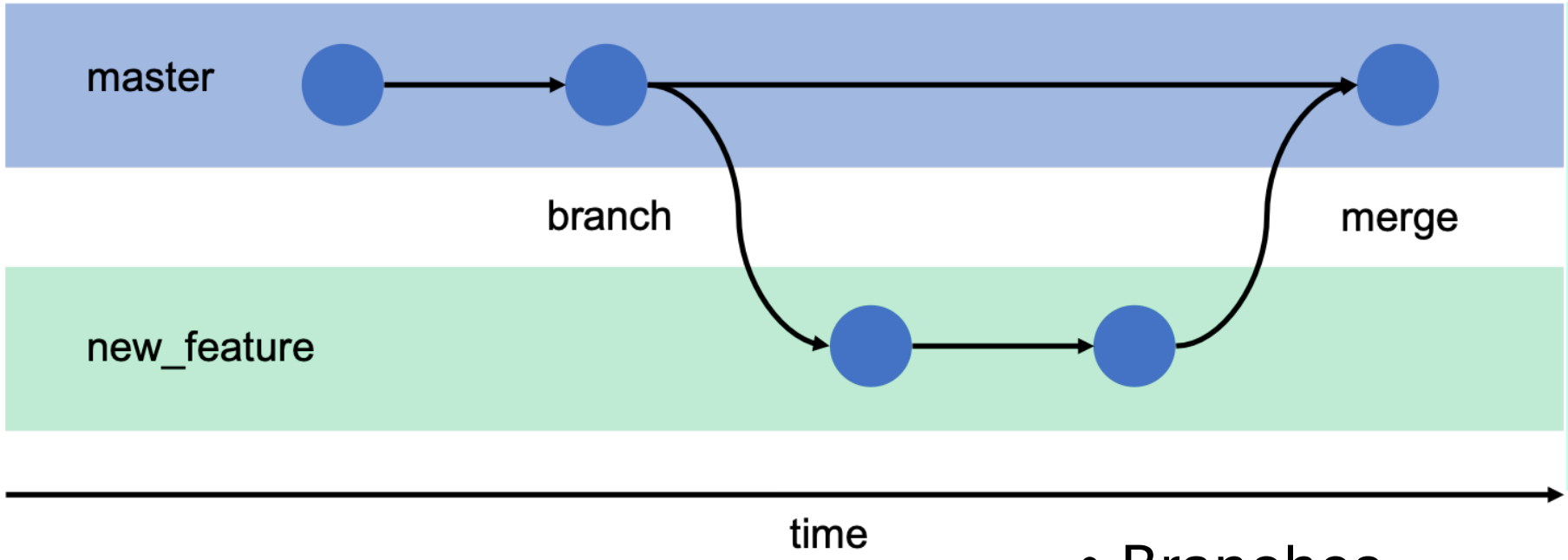
- Main trunk
  - Your “master” copy
  - Always in functional state
- Branches
  - Duplicate(s) of main trunk for new code development
  - Testing environment without different version of same files
- Merge
  - If new development is viable then join back to main trunk



DEMO

BRANCHES

- Main trunk
  - Your “master” copy
  - Always in functional state
- Branches
  - Duplicate(s) of main trunk for new code development
  - Testing environment without different version of same files
- Merge
  - If new development is viable then join back to main trunk

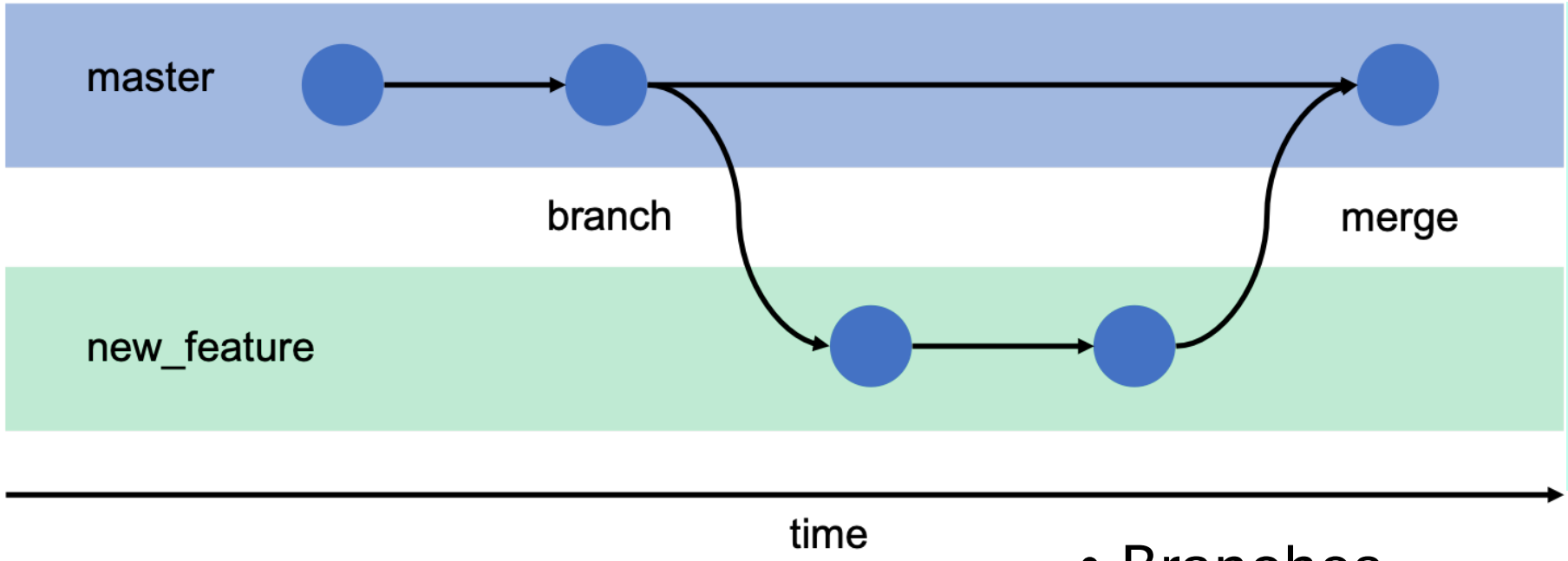


Try it yourself:

- Create a new branch
- Make some changes
- Create a pull request to merge changes back into main trunk

BRANCHES

- Main trunk
  - Your “master” copy
  - Always in functional state
- Branches
  - Duplicate(s) of main trunk for new code development
  - Testing environment without different version of same files
- Merge
  - If new development is viable then join back to main trunk



Try it yourself:

- Create a new branch
- Make some changes
- Create a pull request to merge changes back into main trunk

Bonus:

- Edit the same line of code on both the main branch and on the feature branch

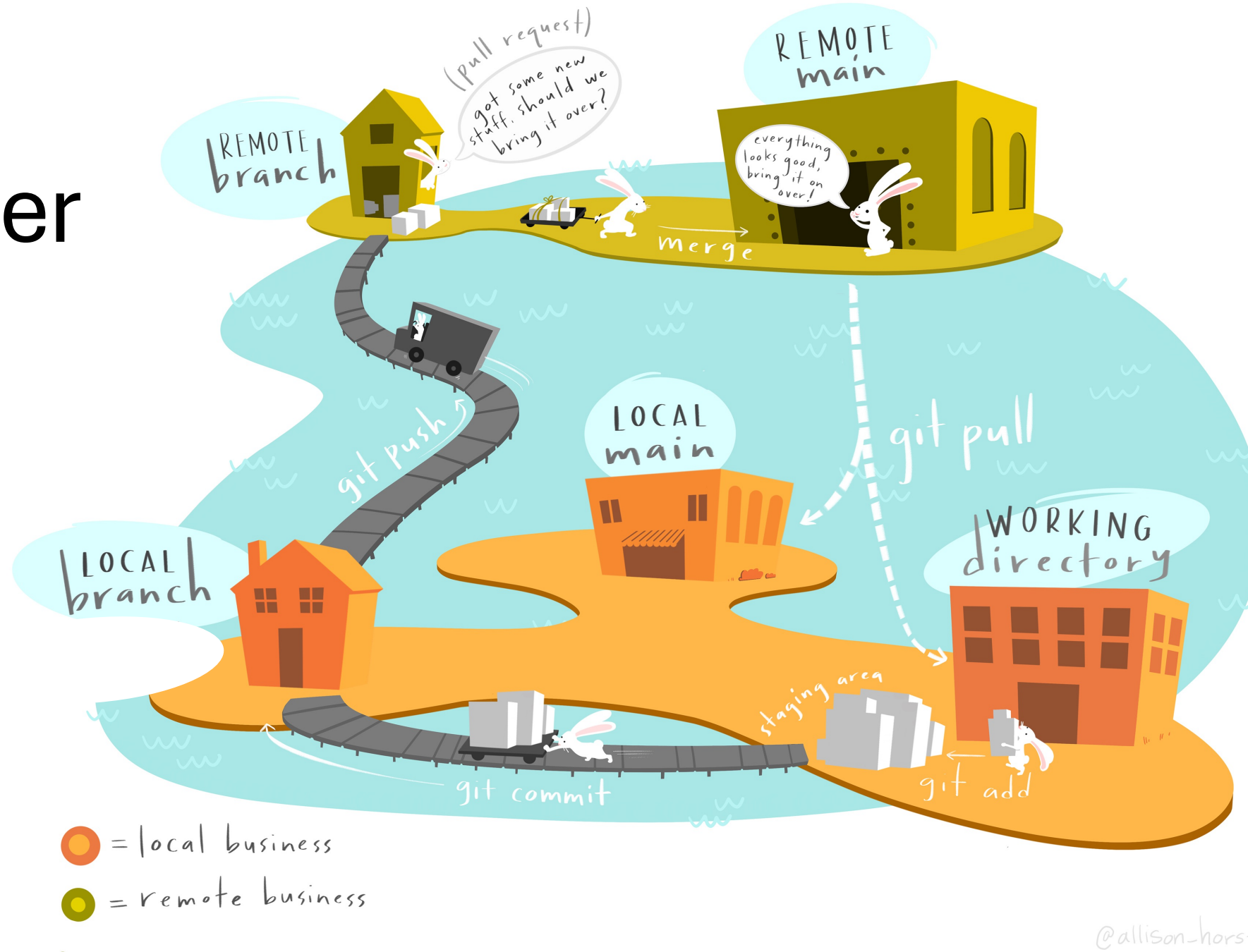
What happens?

- Main trunk
  - Your “master” copy
  - Always in functional state
- Branches
  - Duplicate(s) of new code devel
  - Testing environ different version of same files
- Merge
  - If new development is viable then join back to main trunk

BRANCHES

Put it together

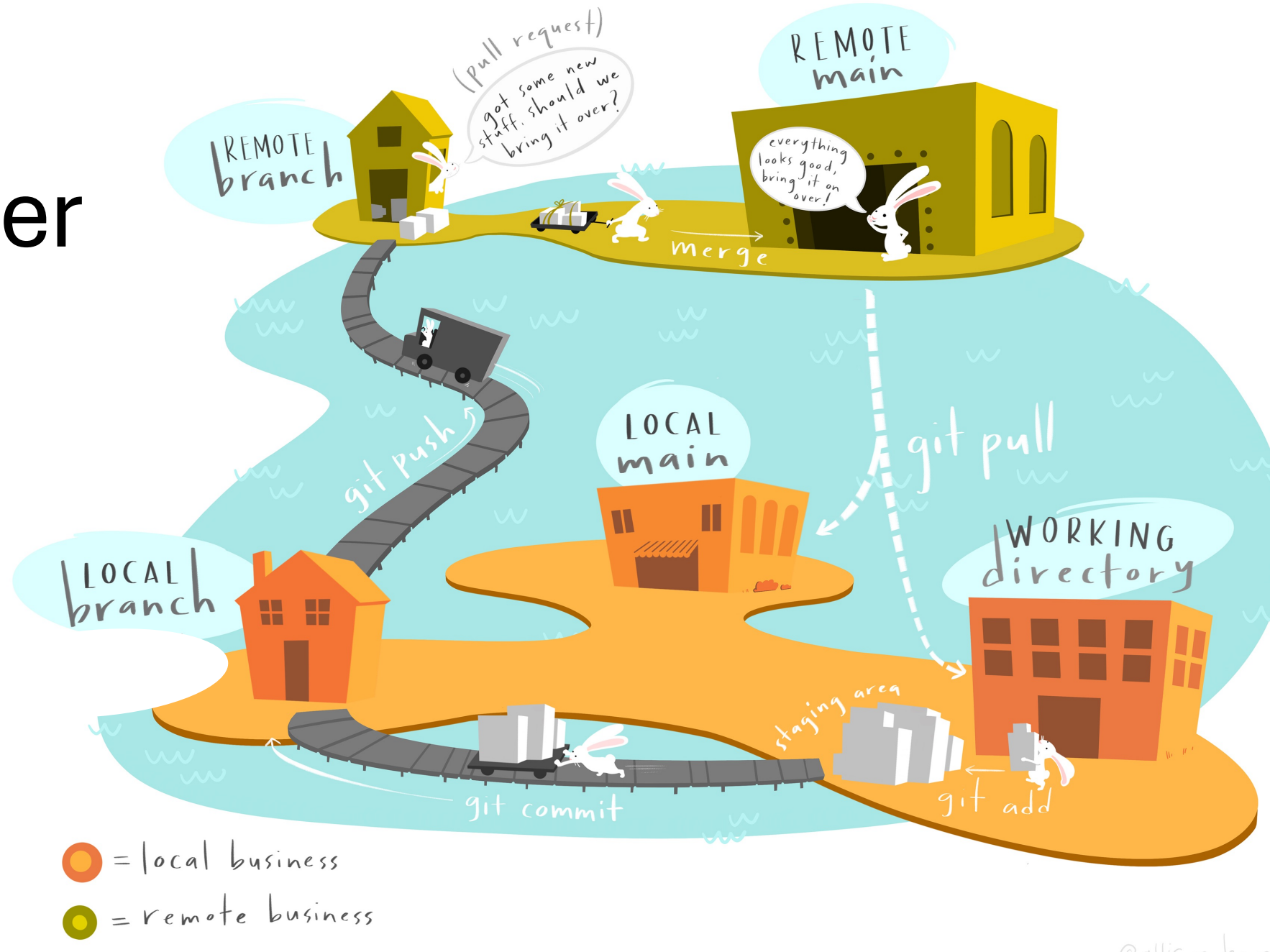
DEMO



Put it together

Try it yourself:

- Work in pairs
- Invite a collaborator to your repo
- Have them create a branch
- Make changes and introduce a merge conflict and create a pull request
- Create a pull request to merge changes back into main trunk



Summary

- Git is great
  - Version control
  - Safe explorations
  - Safe collaborations
- Some new lingo
  - Repository
  - Add and Commit
  - Push and Pull
  - Branches
  - Merge conflicts

Resources

- [Happy Git and GitHub for the useR]

Happy Git and GitHub for the useR

Search

Table of contents

Let's Git started

1 Why Git? Why GitHub?

2 Contributors

3 Workshops

Installation

Half the battle

4 Register a GitHub account

5 Install or upgrade R and RStudio

6 Install Git

7 Introduce yourself to Git

8 Install a Git client

Let's Git started

On this page

Let's Git started

License

View source

Edit this page