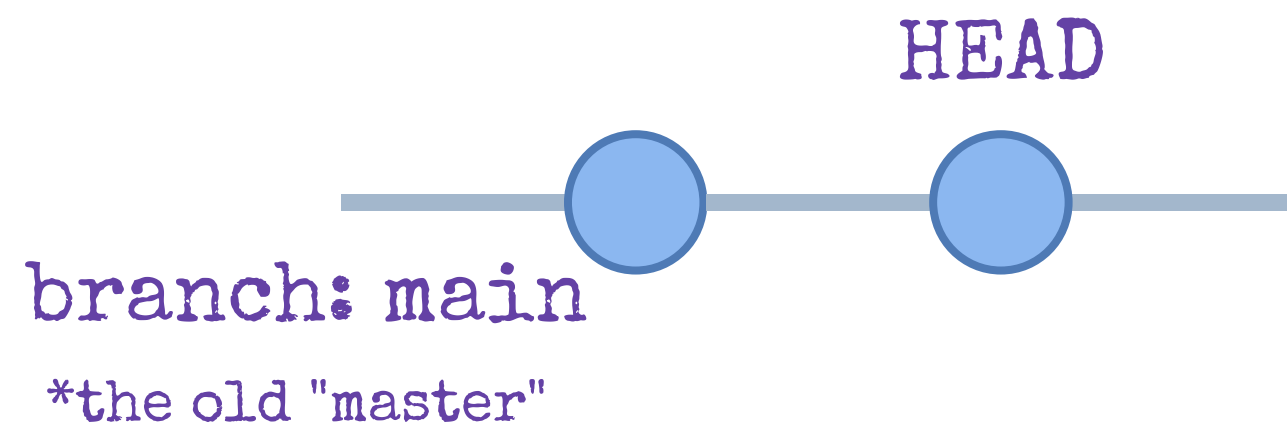
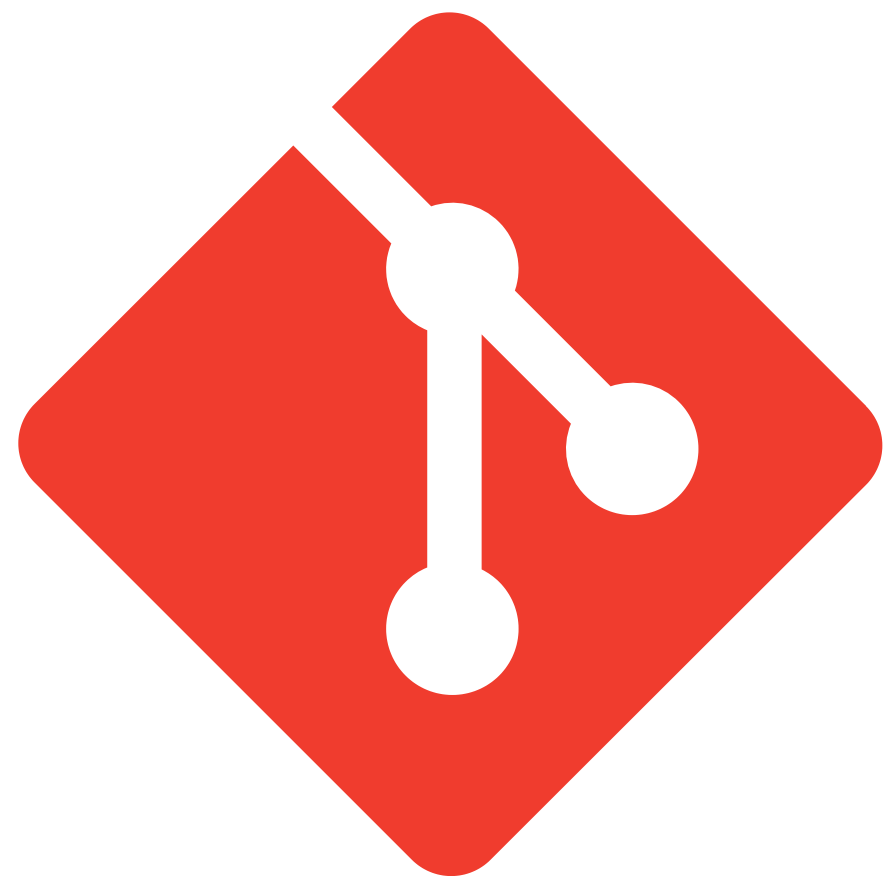


# Git

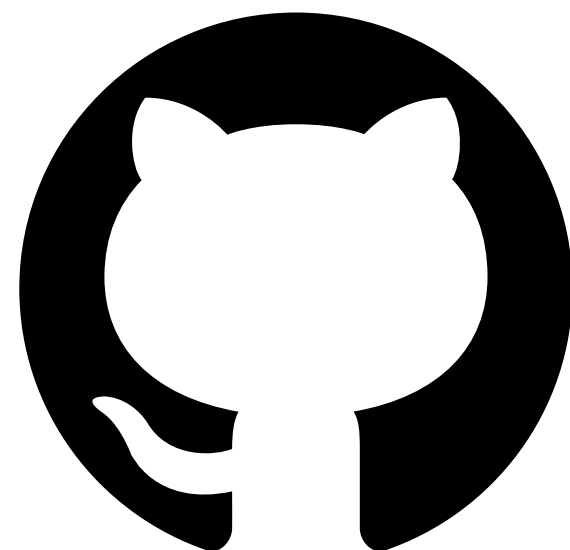


## Concept of a "Stack"

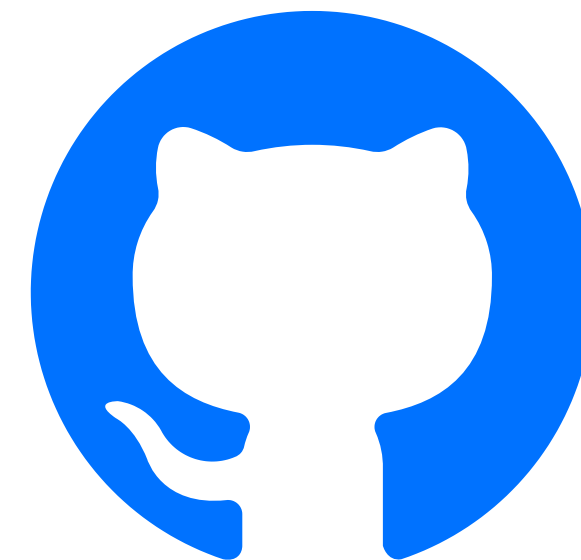




Git



GitHub



GitHub  
Enterprise



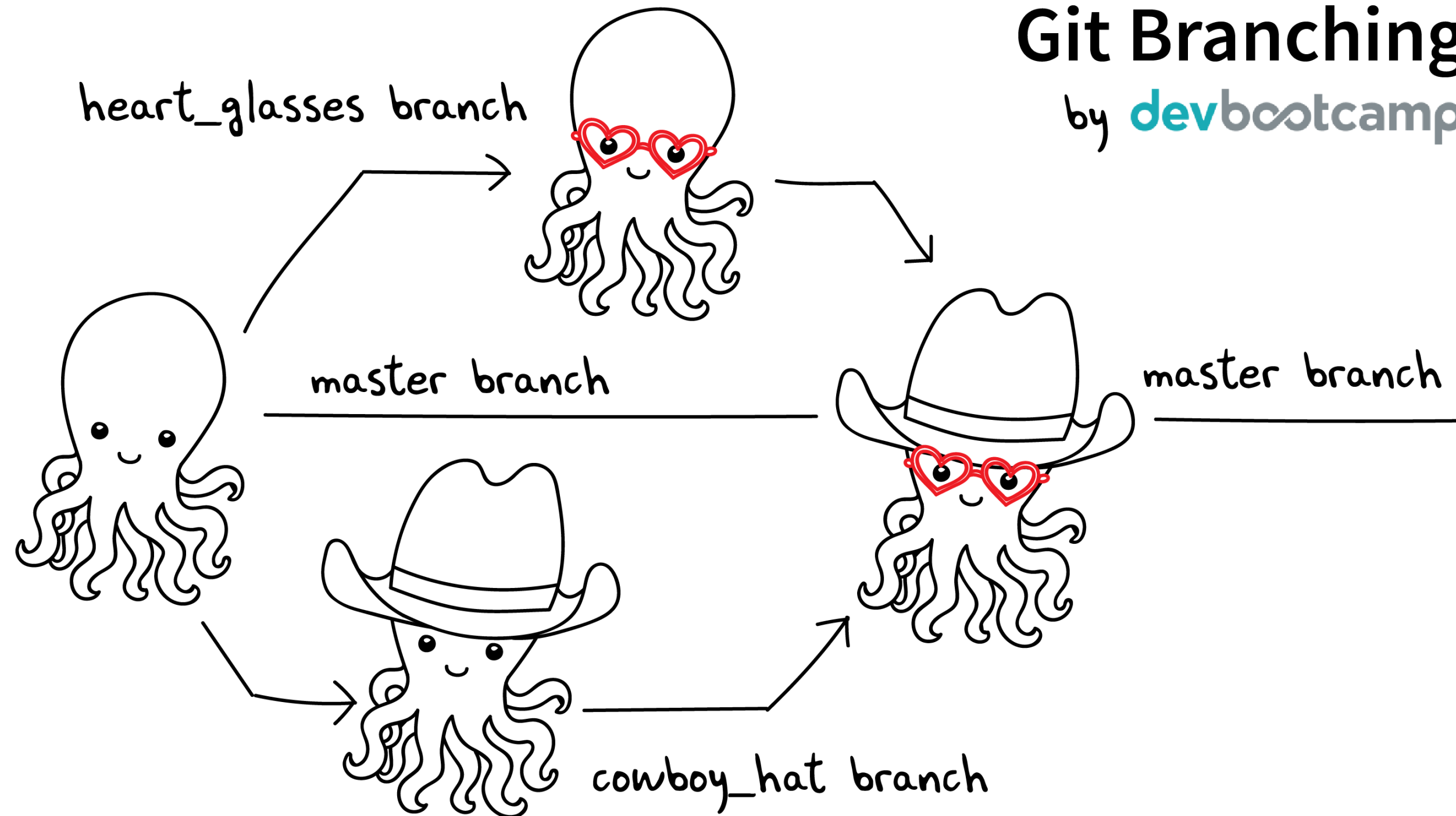
GitLab



Azure  
Repos

# Git Branching

by **dev**bootcamp

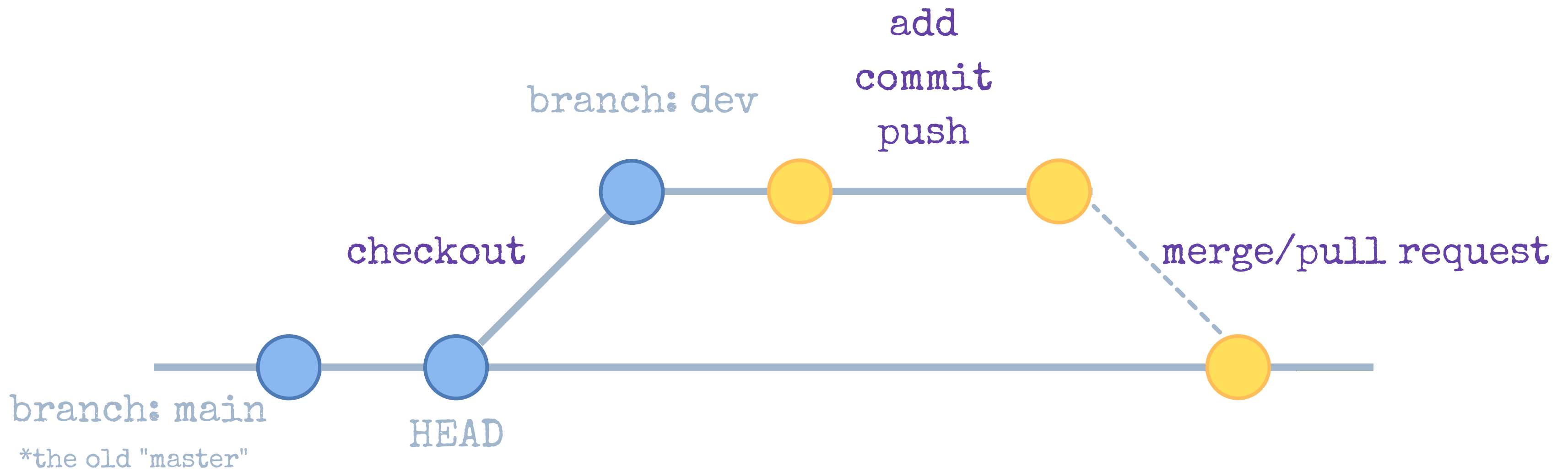


THIS IS GIT. IT TRACKS COLLABORATIVE WORK  
ON PROJECTS THROUGH A BEAUTIFUL  
DISTRIBUTED GRAPH THEORY TREE MODEL.

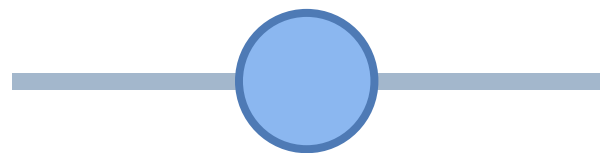
COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL  
COMMANDS AND TYPE THEM TO SYNC UP.  
IF YOU GET ERRORS, SAVE YOUR WORK  
ELSEWHERE, DELETE THE PROJECT,  
AND DOWNLOAD A FRESH COPY.

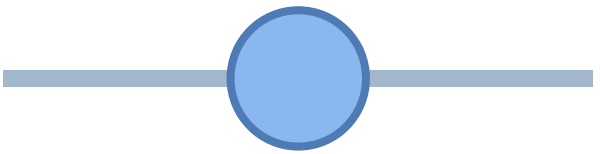




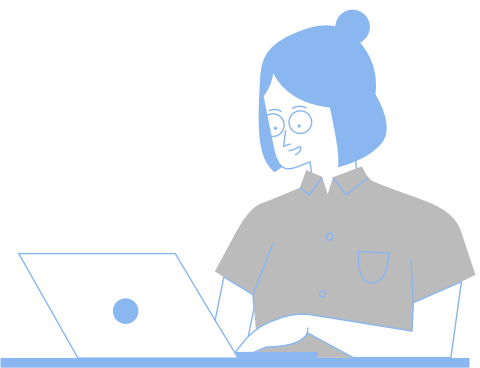
# Full Story



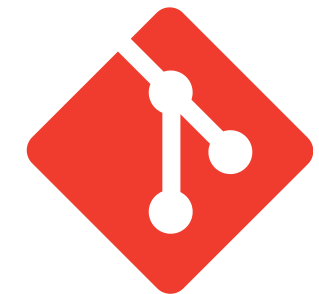
remote branch: origin/main



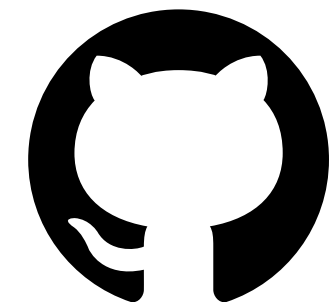
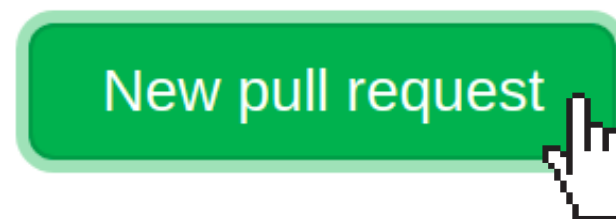
local branch: main



```
$ git clone [https://...]
$ git branch
    main
$ git branch lime
$ git checkout lime
...
[code change]
...
$ git add .
$ git commit -m "[message]"
$ git push origin lime
```

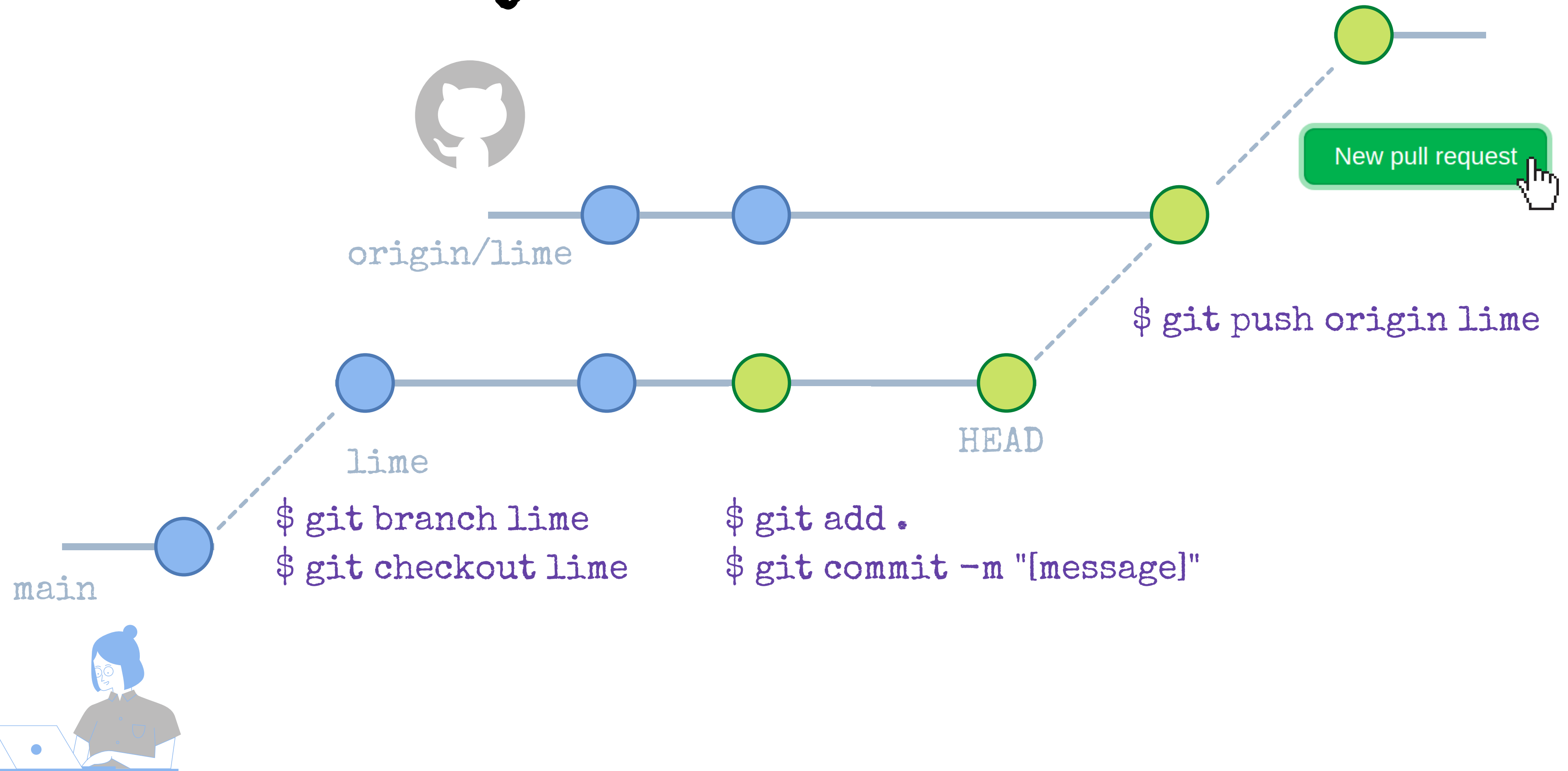


Git



GitHub

# Full Story



# Resources

Help is your best friend

```
$ git [command] --help
```



## About

## Documentation

Reference  
**Book**  
Videos  
External Links

## Downloads

## Community

This book is available in [English](#).

Full translation available in

[azərbaycan dili](#),  
[български език](#),  
[Deutsch](#),  
[Español](#),  
[Français](#),

## Book

The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here.

All content is licensed under the [Creative Commons Attribution Non Commercial Share Alike 3.0 license](#).

Print versions of the book are available on [Amazon.com](#).



2nd Edition (2014)

## Download Ebook



### 1. Getting Started

- 1.1 [About Version Control](#)
- 1.2 [A Short History of Git](#)
- 1.3 [What is Git?](#)
- 1.4 [The Command Line](#)
- 1.5 [Installing Git](#)
- 1.6 [First-Time Git Setup](#)
- 1.7 [Getting Help](#)
- 1.8 [Summary](#)

## GitHub Guides

[Video Guides](#)

[GitHub Help](#)

[GitHub.com](#)



## Understanding the GitHub flow

GitHub flow is a lightweight, branch-based workflow that supports teams and projects where deployments are made regularly. This guide explains how and why GitHub flow works.

🕒 5 minute read



## Hello World

The easiest way to get started with GitHub. In this guide you'll complete a time honored "Hello World" exercise, and learn GitHub essentials.

🕒 10 minute read

<https://git-scm.com/book/en/v2>

<https://guides.github.com/>



# Pulling Data from remote



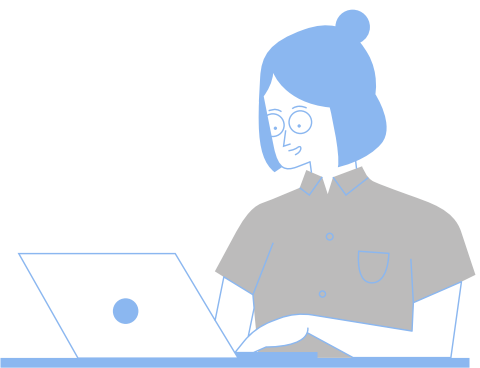
```
$ git remote update
```

```
$ git status --untracked-files=no
```

On branch lime

Your branch is behind 'origin/lime' by 1  
commit, and can be fast-forwarded.

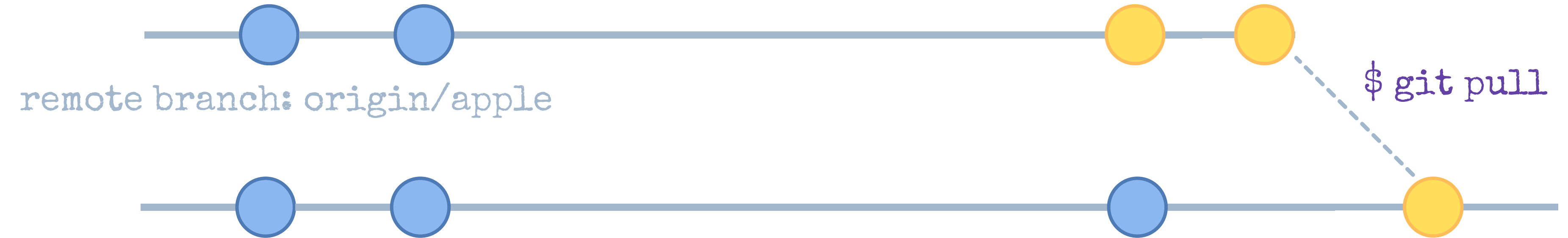
(use "git pull" to update your local branch)



# Pulling Data from remote



HEAD

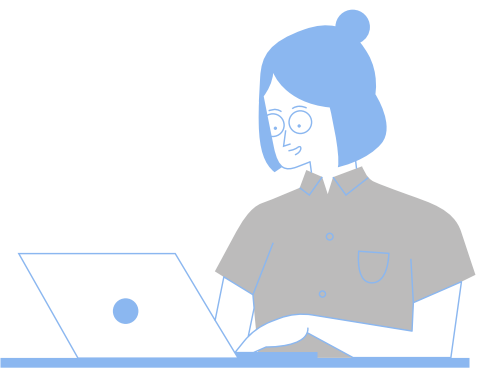


remote branch: origin/apple

\$ git pull

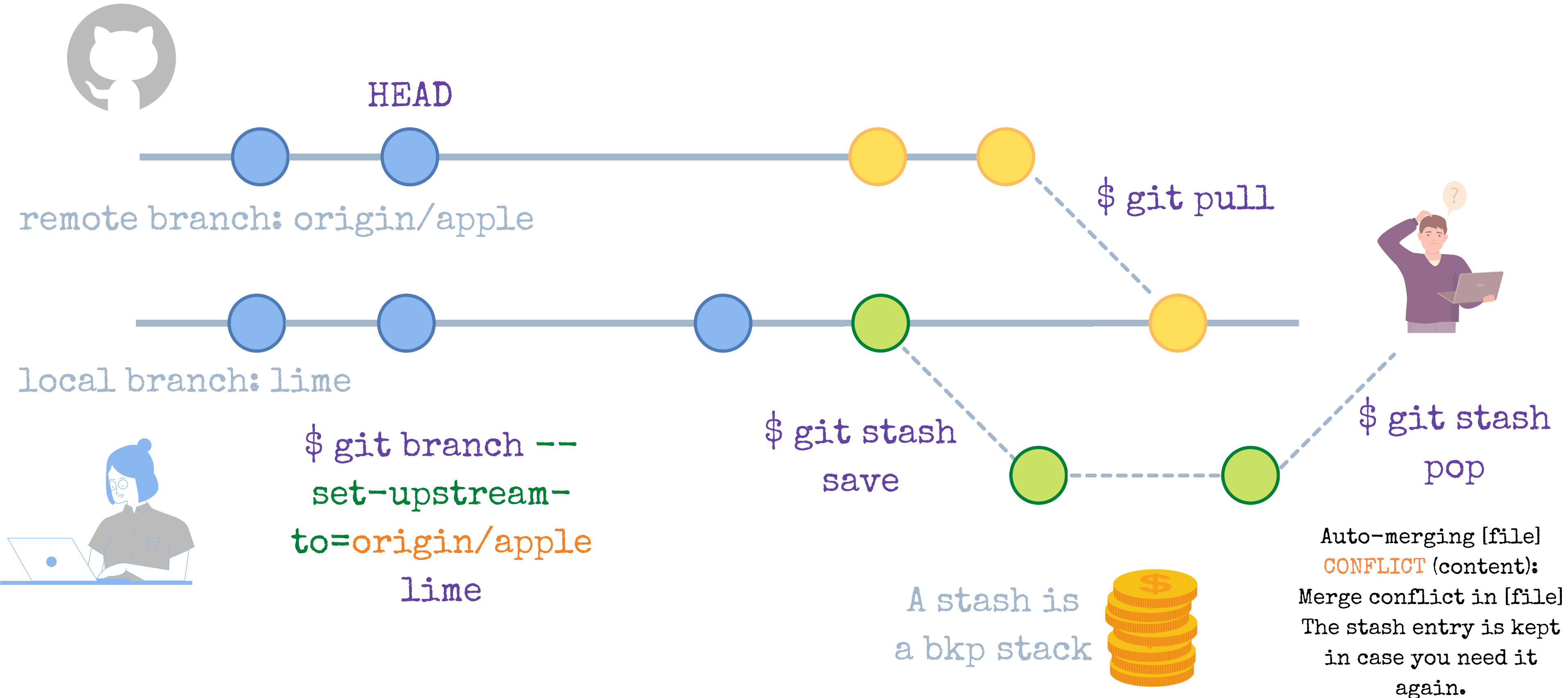
local branch: lime

```
$ git branch --set-upstream-  
to=origin/apple lime
```



Branch 'lime' set up  
to track remote  
branch 'apple' from  
'origin'.

# Pulling Data from remote



(1) Solve inconsistencies manually where git issued warnings:

A

<<<<<< Updated upstream

B

=====

C

>>>>>> Stashed changes

(2) Add and push:

```
$ git add .
```

```
$ git push origin HEAD:apple
```

This is the safest method available!

# How to ignore files!

Ignore certain files:

```
file.py
```

Ignore directories:

```
dir/
```

Enforce exceptions:

```
!custom.c
```

```
!dir/
```

Wild cards:


```
*.c
```

Dir and subdirs:

```
dir/*/**
```

# How to ignore files!

Owner \*

 felipepenha ▾

/


Repository name \*


test ✓

Great repository names are short and memorable. Need inspiration? How about **legendary-succotash**?

Description (optional)

Test

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

☒ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▾

☒ **.gitignore template**

Filter ignores...

Processing

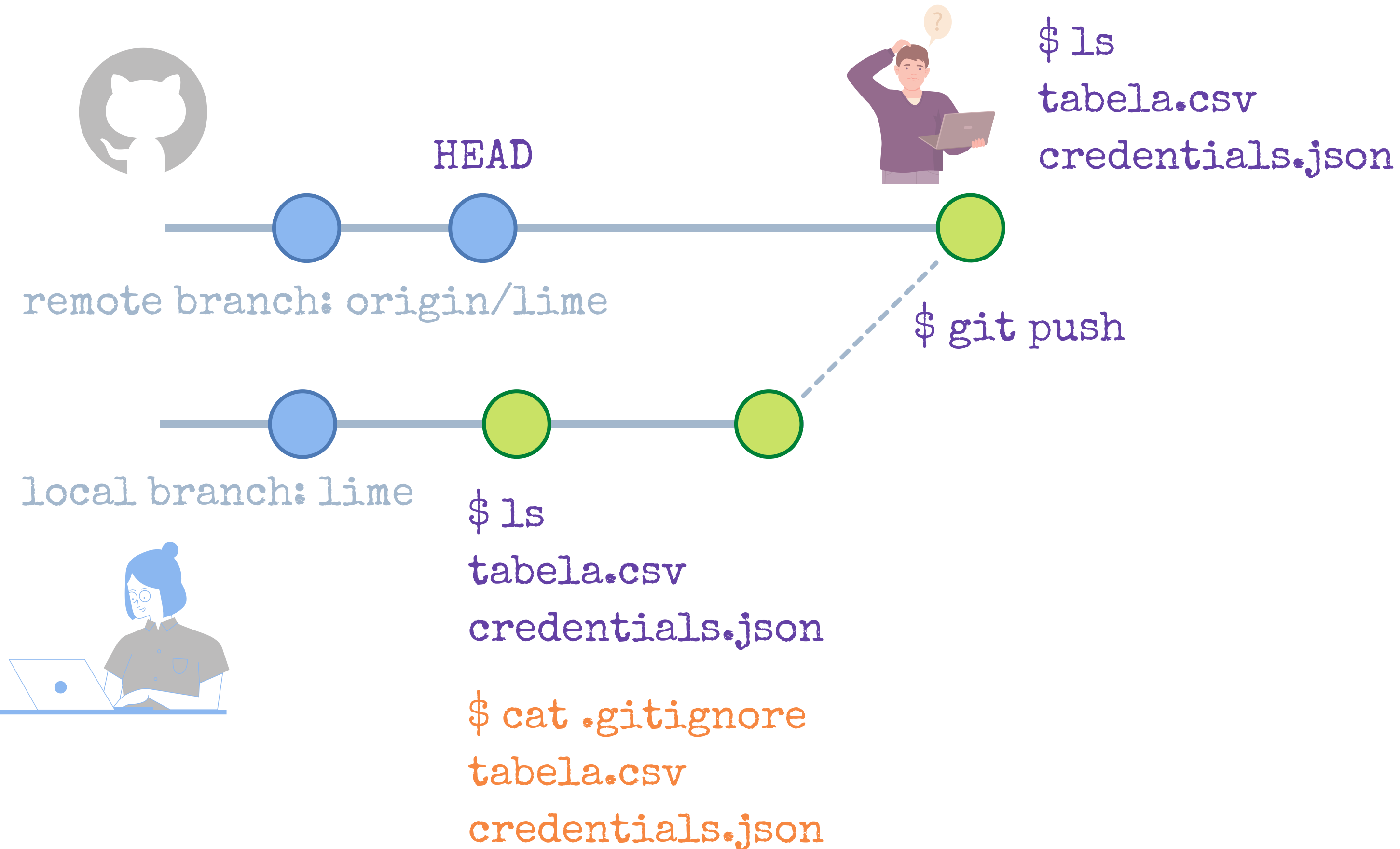
PureScript

✓ Python

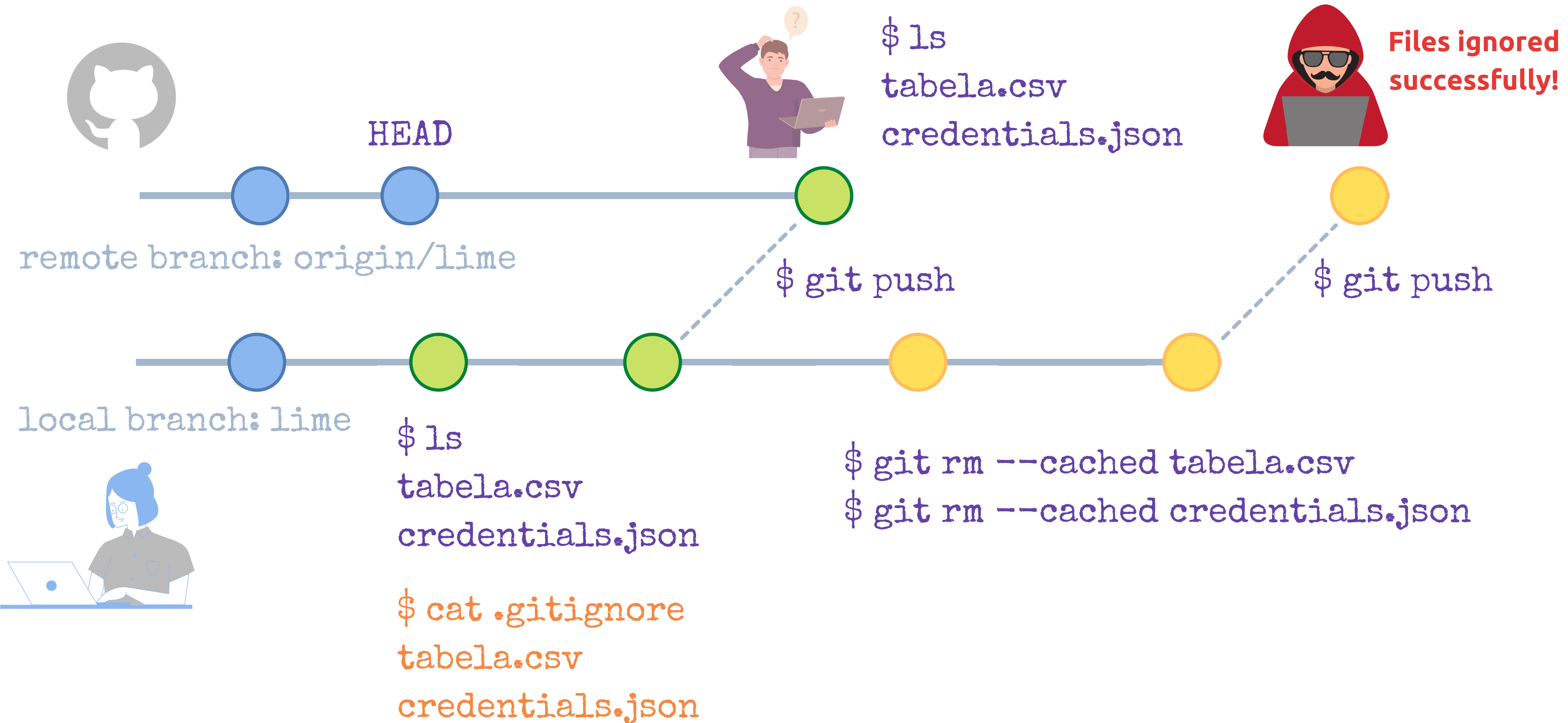
Qooxdoo

This repository will be initialized with the default name in your [settings](#).

# How to ignore files!



# How to ignore files!





# Tags!

- tags are unique
- no pattern is enforced
- but there are some conventions:

Semantic Versioning 2.0.0

<https://semver.org/>

Given a version number MAJOR.MINOR.PATCH, increment the:

MAJOR version when you make incompatible API changes,

MINOR version when you add functionality in a backwards compatible manner, and

PATCH version when you make backwards compatible bug fixes.

Major version zero (0.y.z) is for initial development. Anything MAY change at any time. The public API SHOULD NOT be considered stable.

# Tags!

```
$ git tag --annotate [x.y.z]  
--message "[message]"
```

How to push with the tag?

```
$ git push origin [branch_name] tag [x.y.z]
```

# Tags!

```
git clone --branch [tag_name] [clone_url]
```

```
git checkout tags/[tag_name] -b [new_branch]
```

```
git fetch --tags
```

```
git tag --list
```

# •git

•git is usually built upon

\$ git init

or

\$ git clone

For example, log information for  
is stored here:

\$ ls -a logs/refs/heads/

which is also displayed here:

\$ git log

# •git

```
find .git/objects -type f
```

```
$ git cat-file -p [branch-name]^{tree}
```

[file is altered]

```
$ git hash-object -w [file]
```

```
f2ba8f84ab5c1bce84a7b441cb1959cfc7093b7f
```

```
$ find .git/objects -type f | grep "b7f"
```

```
.git/objects/f2/ba8f84ab5c1bce84a7b441cb1959cfc7093b7f
```

```
$ git cat-file -p [old hash] > test.txt
```