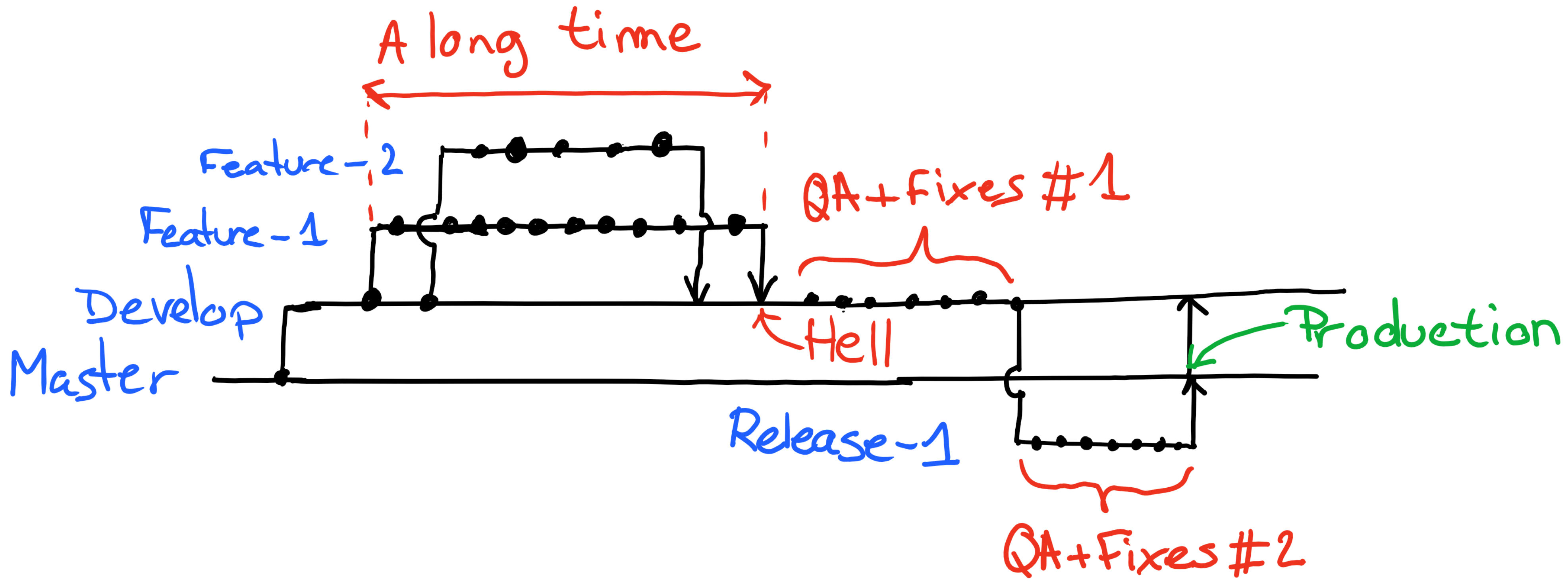

A long time
Feature-2
Feature-1
Develop
Master
QA+Fixes #1
Hell
Production
Release-1
QA+Fixes #2

Git

Outline

- What is version control?
- Why we need version control?
- What is Git?
- Getting Started with Git
  - `git init`
  - `git commit`
  - `git log`
- Basic Git Workflow

What is version control?

- **Version control systems** are a category of software tools that help a software team manage changes to source code over time.

- **Version control** software keeps track of every modification to the code in a special kind of database.

- It not only keeps the content of your modifications, but also keeps metadata about your changes (author, timestamps, etc.).

- Obsolete systems like Subversion, and Mercurial.

Why we need version control?

- **History**
  - A complete long-term change history of every file in your codebase.
  - Includes roll-back for when mistakes happen!
- **Branching and Merging**
  - Teams can benefit from the ability to work on independent streams of changes.
- **Traceability**
  - Being able to trace each change made to the software and connect it to project management and bug tracking software.
- **Collaboration**
- **Reproducibility**

What is Git?

- Git is a version control system.
    - Can record snapshots and track the content of a folder as it changes over time.
- Every time we **commit** a snapshot, Git records a snapshot of the **entire project**, saves it, and assigns it a version.
- These snapshots are kept inside a sub-folder called **.git**.

What is Git?

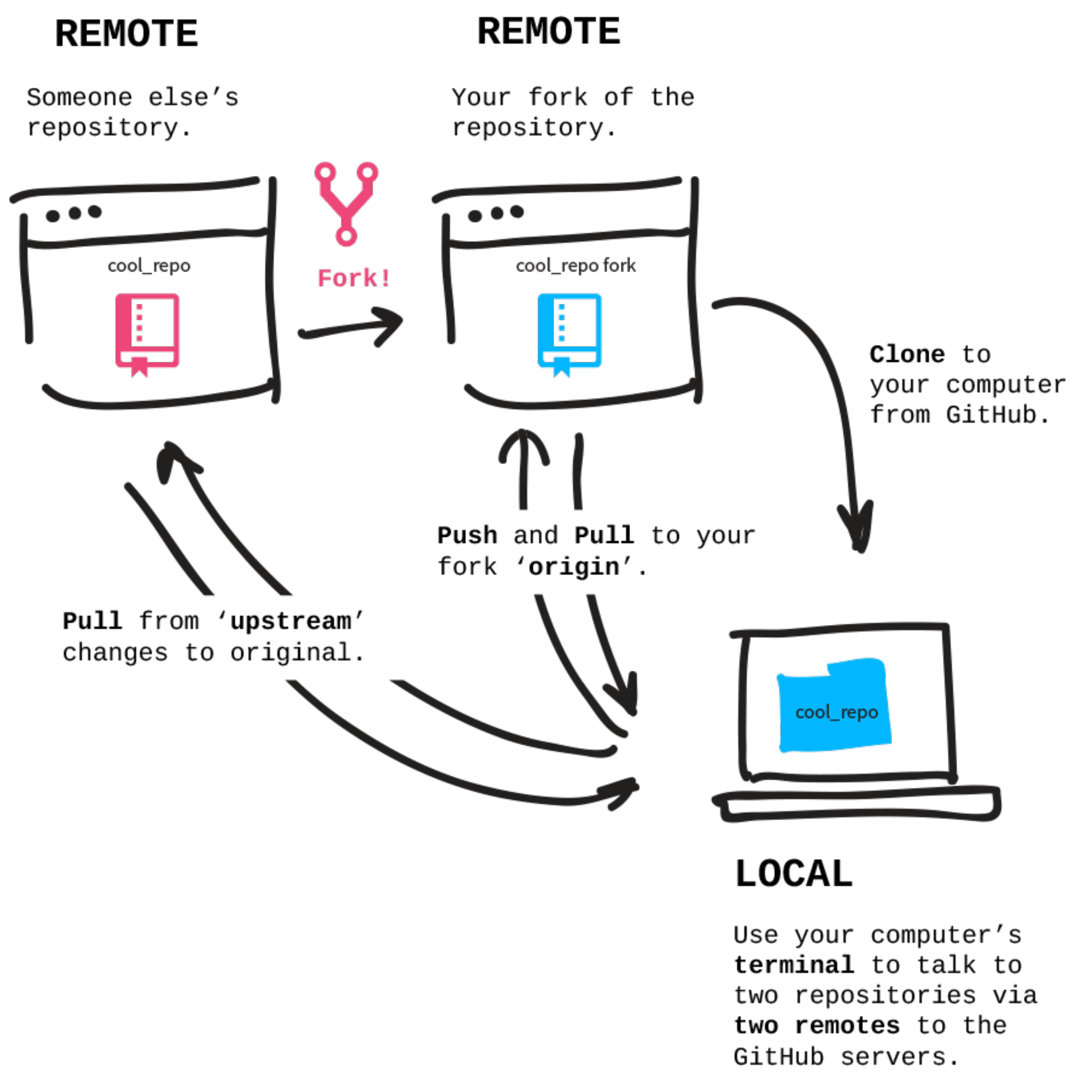
- Removing **.git** will remove the repository and history.
  - Your working directory and any remote copies remain unaffected.
- **.git** uses relative paths
  - You can move the repository to any other machine and it would still work!
- **Git** has multiple interfaces (CLI, GUI, web), and is shipped out of the box with many Linux-based systems.

What is Git?

- To check if git is installed, open up a terminal window and type the following
    - **`git —-version`**
- This will display the version number if **git** is installed.

Getting Started with Git

- We can **clone** an existing remote repository.
  - `git clone REMOTE_URL`
- We can **initialize** a new local repository.
  - `git init`


REMOTE
Someone else's repository.
cool_repo
Fork!
REMOTE
Your fork of the repository.
cool_repo fork
Clone to your computer from GitHub.
Push and Pull to your fork 'origin'.
Pull from 'upstream' changes to original.
cool_repo
LOCAL
Use your computer's terminal to talk to two repositories via two remotes to the GitHub servers.

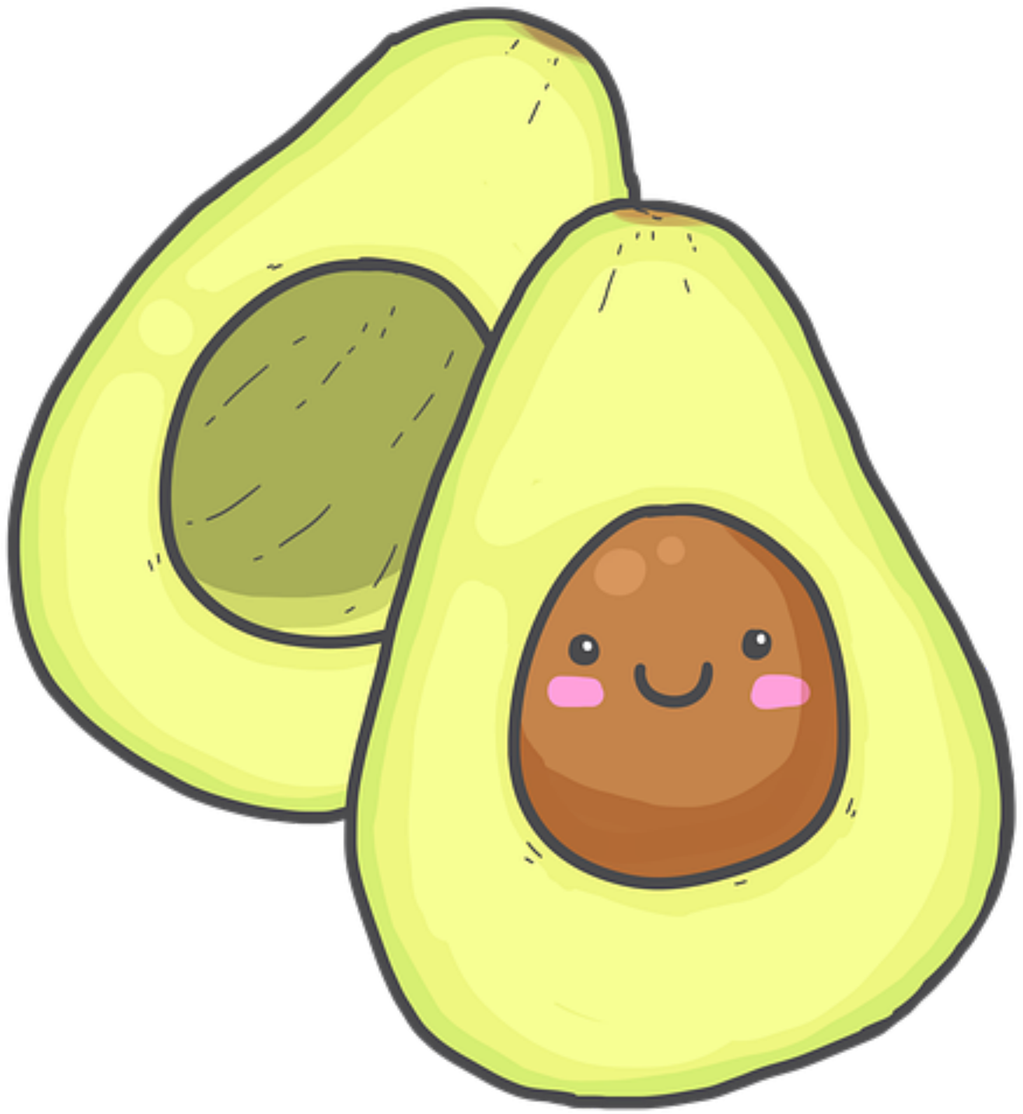
But before getting Started with Git…

- We need to configure Git.

```
$ git config --global user.name "Your Name"
$ git config --global user.email yourname@example.com
```

To-Do Task

Tracking a guacamole recipe with Git



Git Log

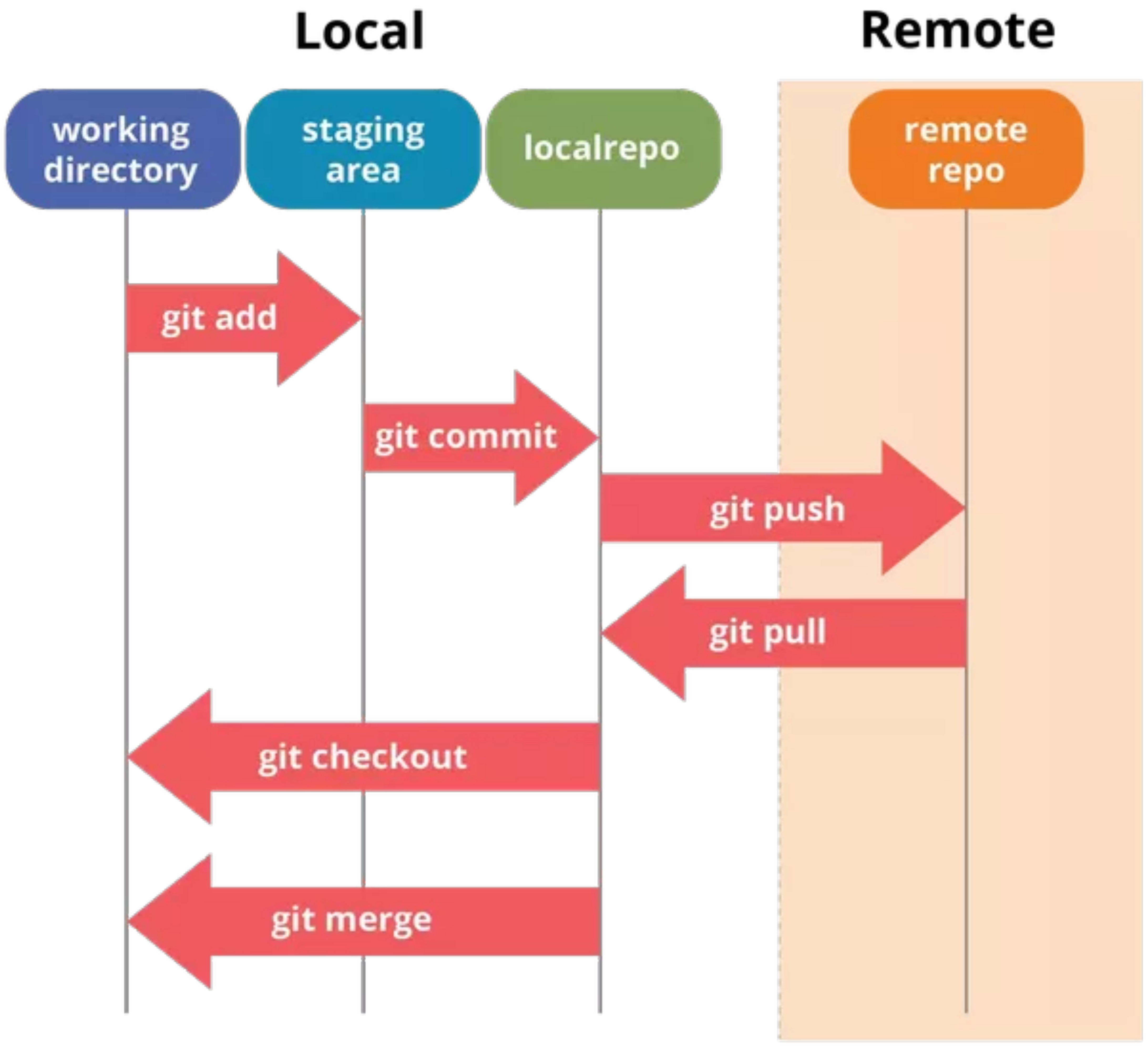
- We can use **`git log`** to display the history of the repository.
- Each **commit** is given a unique long hash as an identifier.
- Output is in *reverse chronological order*, i.e. newest commits on top.
- We will use the hashes when:
  - comparing versions
  - reverting changes

COMMENT	DATE
CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
ENABLED CONFIG FILE PARSING	9 HOURS AGO
MISC BUGFIXES	5 HOURS AGO
CODE ADDITIONS/EDITS	4 HOURS AGO
MORE CODE	4 HOURS AGO
HERE HAVE CODE	4 HOURS AGO
AAAAAAAA	3 HOURS AGO
ADKFJSLKDFJSDKLFJ	3 HOURS AGO
MY HANDS ARE TYPING WORDS	2 HOURS AGO
HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Basic Git Workflow

How do we use it?


Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout
git merge

Getting Help

- If you need help, you can use **`git help [command]`**.
  - **`git help commit`**
  - **`git help config`**
  - **`git help remote`**
- Use online resources.
  - https://guides.github.com/

In case of fire

1. git commit

2. git push

3. leave building