Mastering git

Lesson 1

Irina Gulina

Tomas Tomecek

XKCD Source: https://explainxkcd.com/wiki/index.php/1597:_Git

Red Hat

Who are the lectors?

Tomas Tomecek

Sr. Principal Software Engineer
Field: Automation, Python, Infrastructure, AI
Interests: Mushrooms obviously 😁, hiking, gardening, snowboarding

ttomecek@redhat.com

tomas@tomecek.net

Red Hat

Irina Gulina

Sr. Software Quality Engineer
Field: RHEL + SAP + Cloud
Interests: DIY, knitting, baking, via ferrata

igulina@redhat.com

Red Hat

How did it start?

- Irina: Workshop "Git troubles: How to find, fix and avoid them", DevConf.CZ 2019
- Irina: "OSSDev: Advanced Git", October 2019
- Irina: Talk "Git Etiquette: Best Practices or Mind your Git Manners", Open House Red Hat, April 2020
- Tomas + Irina: Workshop "If you do force push…. May the force stay with you", DevConf.CZ 2020
- Tomas + Irina: “OSSDev: Advanced Git”, April 2021
- Tomas + Irina: “OSSDev: Mind your Git manners”, March 2022
- Tomas + Irina: “Mastering Git”, 6 weeks course in MUNI, winter 2022

All octocats illustrations come from: https://octodex.github.com/

Congratulations!

2nd iteration

Git is easy to use, if you know a few basic concepts

Focus on practical application

These slides are **NOT** study materials (explained later)

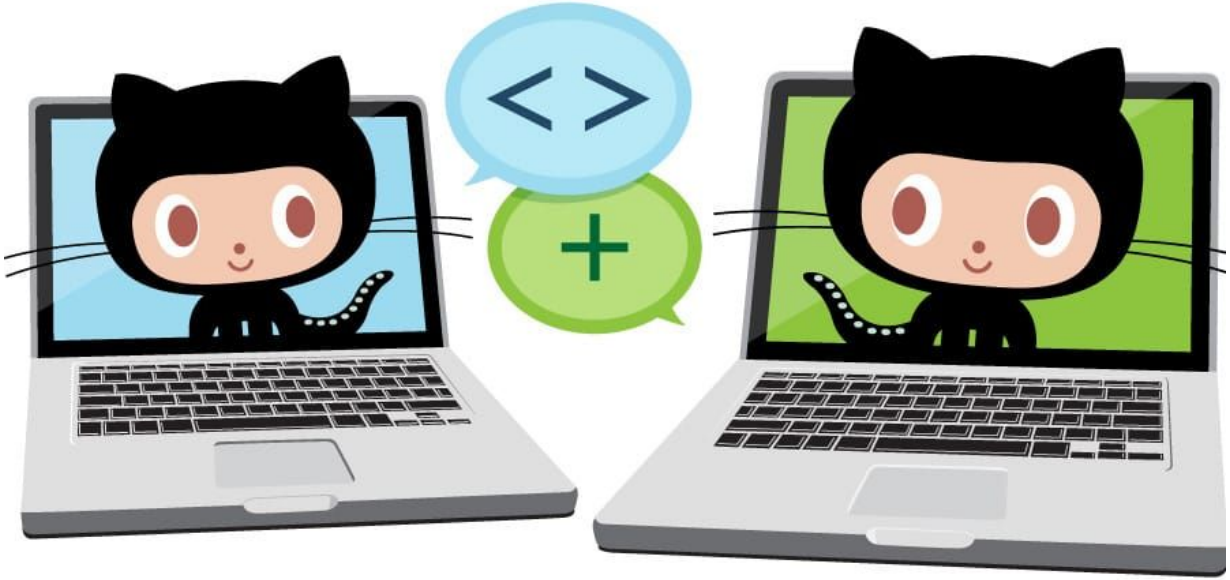
We don’t know MS Windows nor MacOS (containers/VMs)

In 6 classes you’ll learn the essentials so you can be productive

Tell us what you want to learn about

Homeworks: 5 + bonus task (1st class doesn’t have a homework. Yupeee!)
To pass the course: complete all 5HW, or 4HW + bonus

Class objective



About this course

Lesson 1 - Introduction of this course, organization, motivation, basics, commits

Lesson 2 - How does branching work in git

Lesson 3 - Working as a team with a git repository

Lesson 4 - Fixing mistakes

Lesson 5 - Git Etiquette

Lesson 6 - Git features and common open source git workflow

Today’s class

- Version Control and why should you care?
- Installing Git.
- The basics of Git Workflow. Cloning Repositories.
- Index.
- Art of commits.
- HTTPs and SSH.

- Lab: Installing Git. Configuring Git for local repositories. Securing your Git repo with SSH keys. Creating local repositories, adding files locally.
- Bonus: How to write a good README for your project.

What is git?

Distributed **version control system** for managing source code, i.e. it's a system that provides three important capabilities:

- Reversibility
- Concurrency
- Annotation

What is VCS?

Version control system is a system for managing the source code providing three important capabilities:

- **Reversibility - the ability to back up** to a saved, known-good state when you discover that some modification you did was a mistake or a bad idea.
- **Concurrency - the ability to have many people modifying the same collection of code** or documents knowing that conflicting modifications can be detected and resolved.
- **Annotation - attaching explanatory comments** about the intention behind each change to it and a **record of who was responsible for each change**.

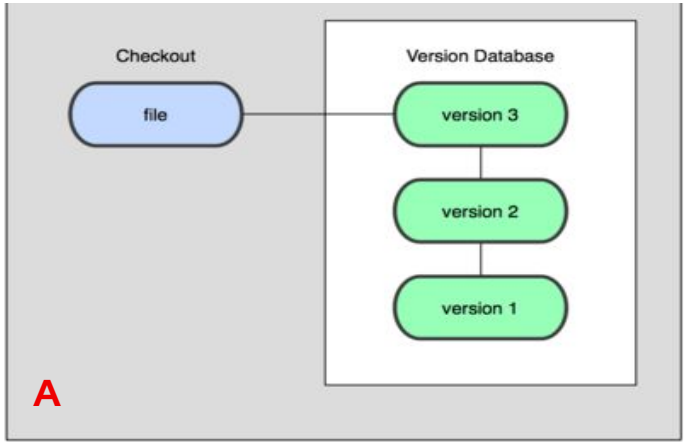
What problem does VCS solve?

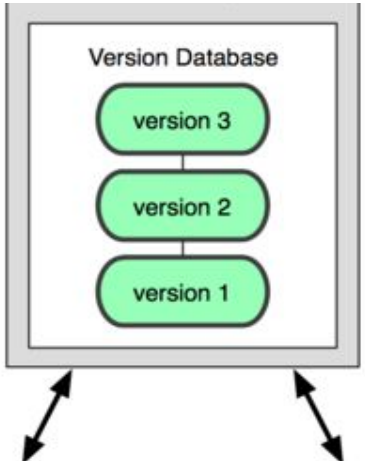
- Keep track of code history
- Collaborate on code as a team
- See who made which changes
- RECOVER

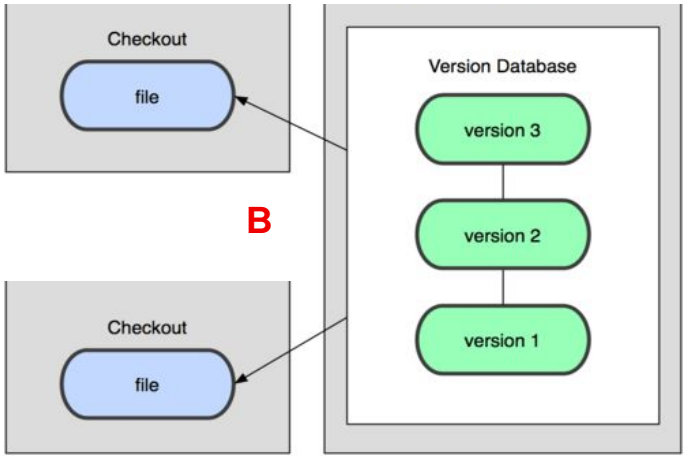
History

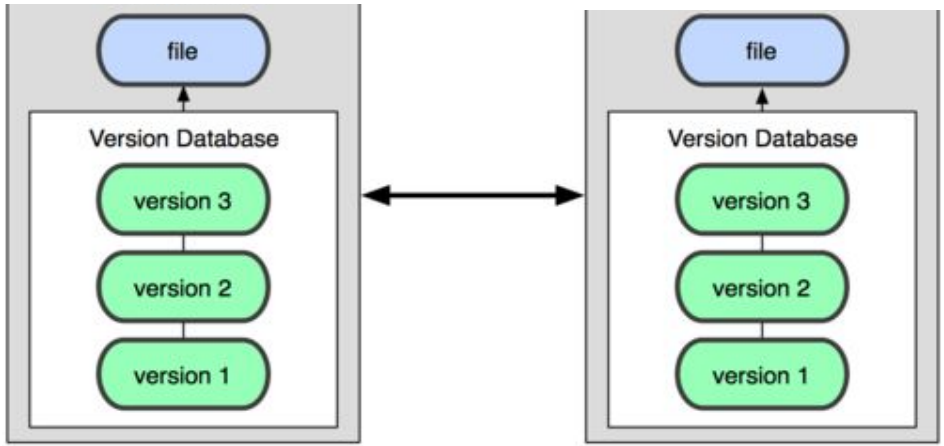
Generation	Networking	Operations	Concurrency	Examples
1st	None (Local)	One file at a time	Locks	RCS, SCCS
2nd	Centralized	Multi-file	Merge before commit	CVS, SourceSafe, Subversion, Team Foundation Server
3rd	Distributed	Changesets	Commit before merge	Bazaar, **Git,** Mercurial

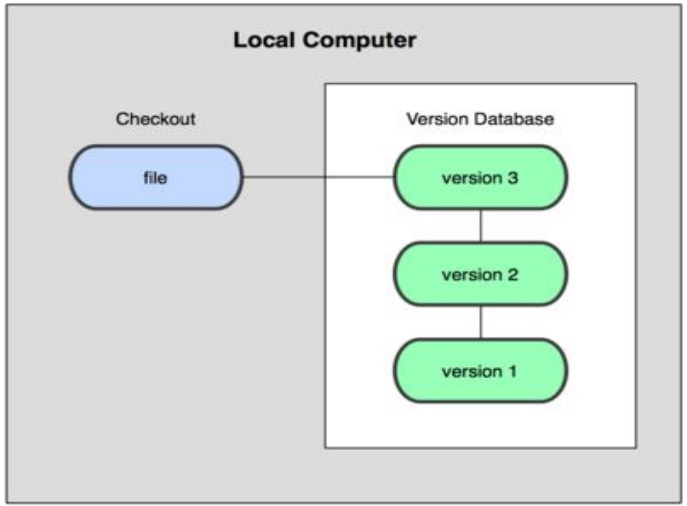
Eric Raymond's "Understanding Version-Control Systems"

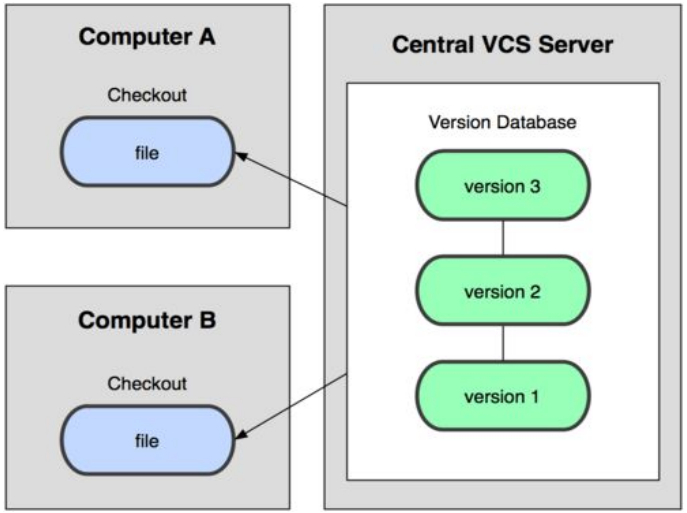

Checkout
file
Version Database
version 3
version 2
version 1
A

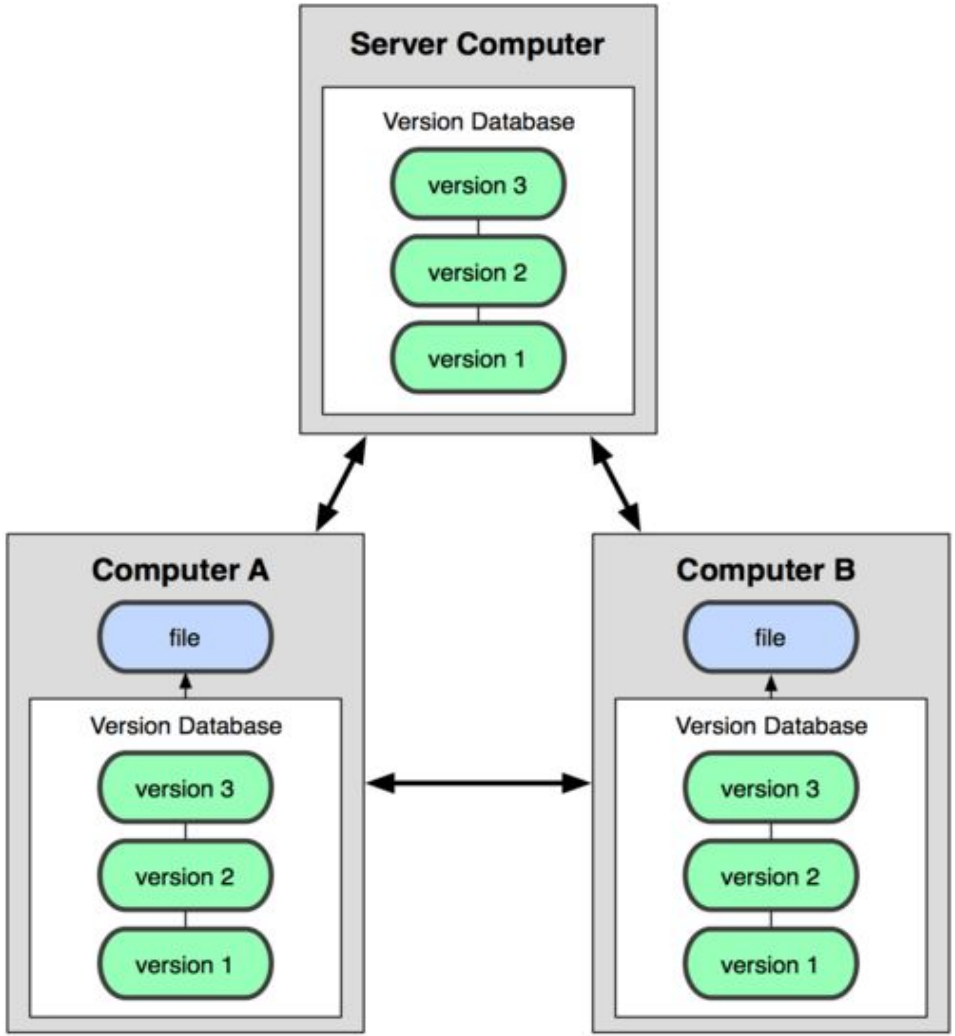

Version Database
version 3
version 2
version 1
C


Checkout
file
B
Checkout
file
Version Database
version 3
version 2
version 1


file
Version Database
version 3
version 2
version 1
file
Version Database
version 3
version 2
version 1


Local Computer
Checkout
file
Version Database
version 3
version 2
version 1


Computer A
Checkout
file
Computer B
Checkout
file
Central VCS Server
Version Database
version 3
version 2
version 1


Server Computer
Version Database
version 3
version 2
version 1
Computer A
file
Version Database
version 3
version 2
version 1
Computer B
file
Version Database
version 3
version 2
version 1

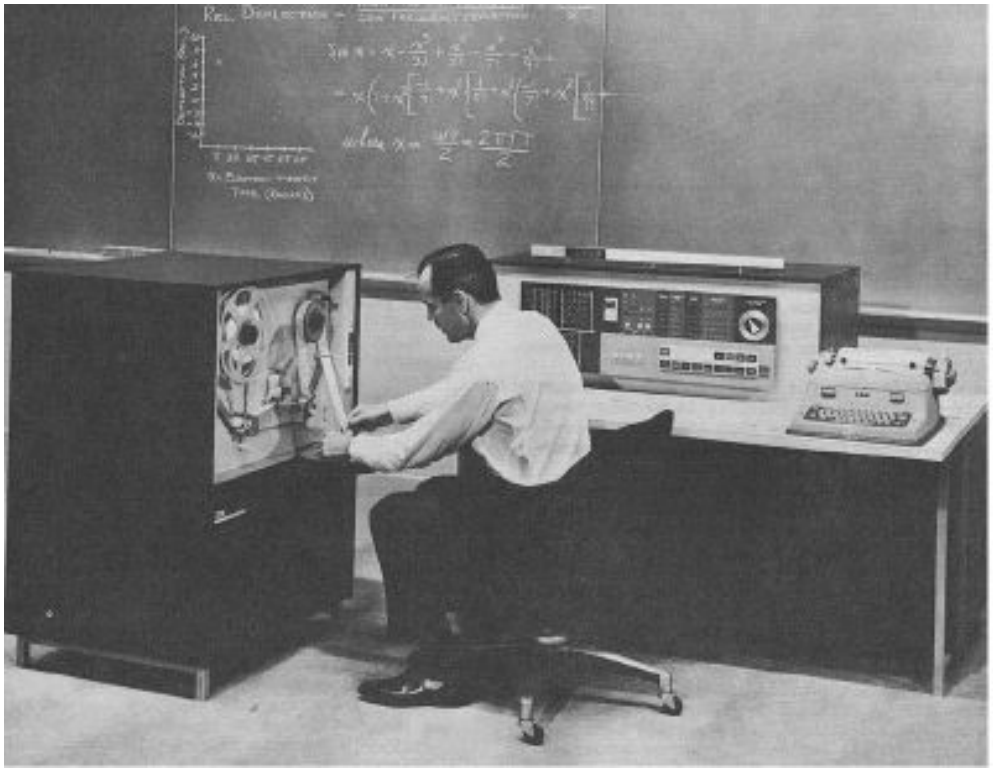
Before Version Control

Before Version Control

¯_(ツ)_/¯

Before Version Control

This 1959 IBM 1620 relied on paper tape to store data and programs



Reference: https://en.wikipedia.org/wiki/Punched_tape

With VCS

Reference https://www.reddit.com/r/ProgrammerHumor/comments/psqlij/chad_progr

With git

Midjourney prompt: “a picture of women sitting in front of computer and collaborating on a software project”

NAME

git - the stupid content tracker

DESCRIPTION

Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

How old is git?

- 11 August 1995
- 22 December 1999
- 14 April 2001
- 7 April 2005
- 13 June 2011
- 5 February 2016

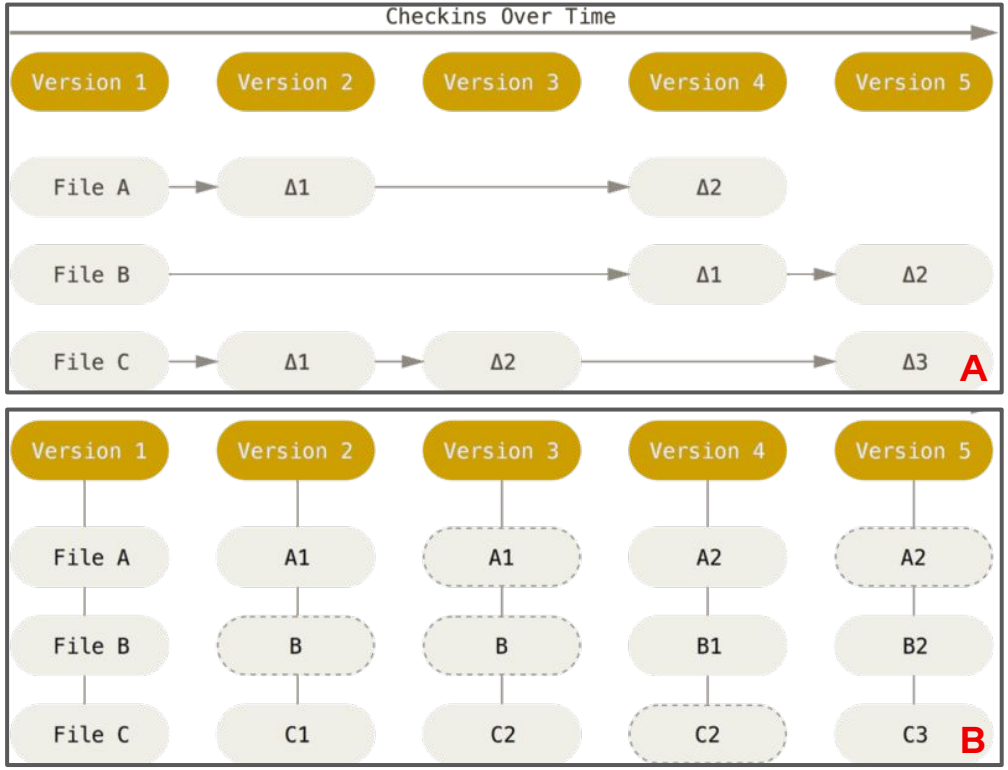
Source: https://octodex.github.com/

How old is git?

- 11 August 1995
- 22 December 1999
- 14 April 2001
- **7 April 2005**
- 13 June 2011
- 5 February 2016

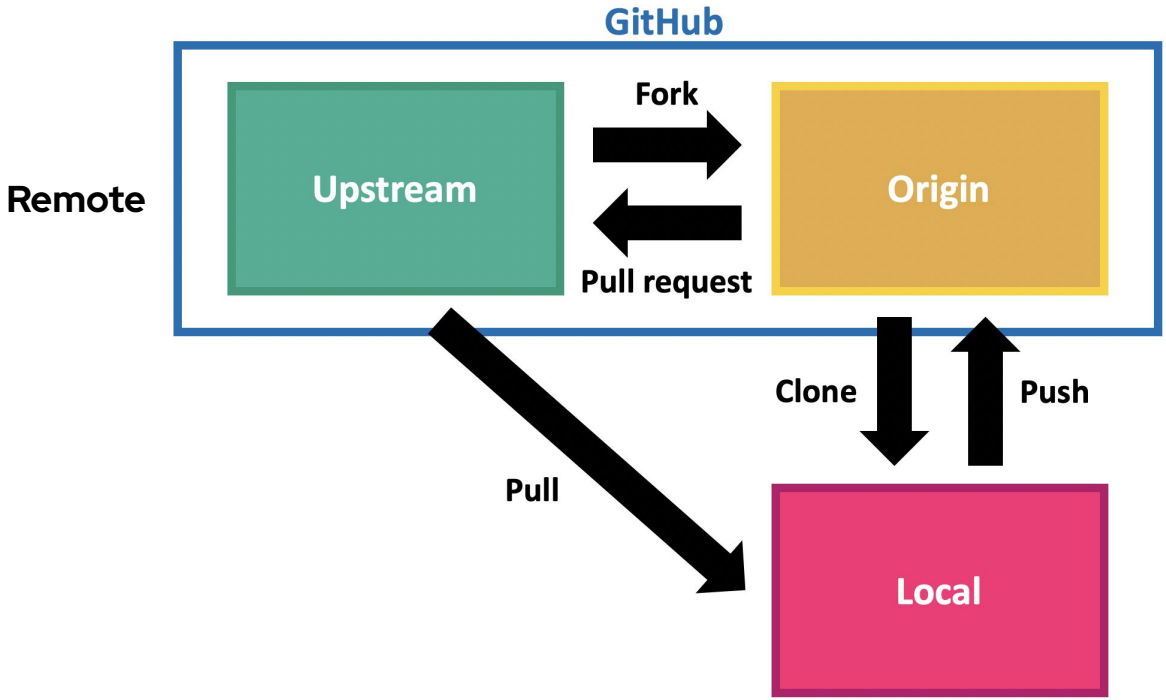
Source: https://octodex.github.com/

Snapshots, not differences

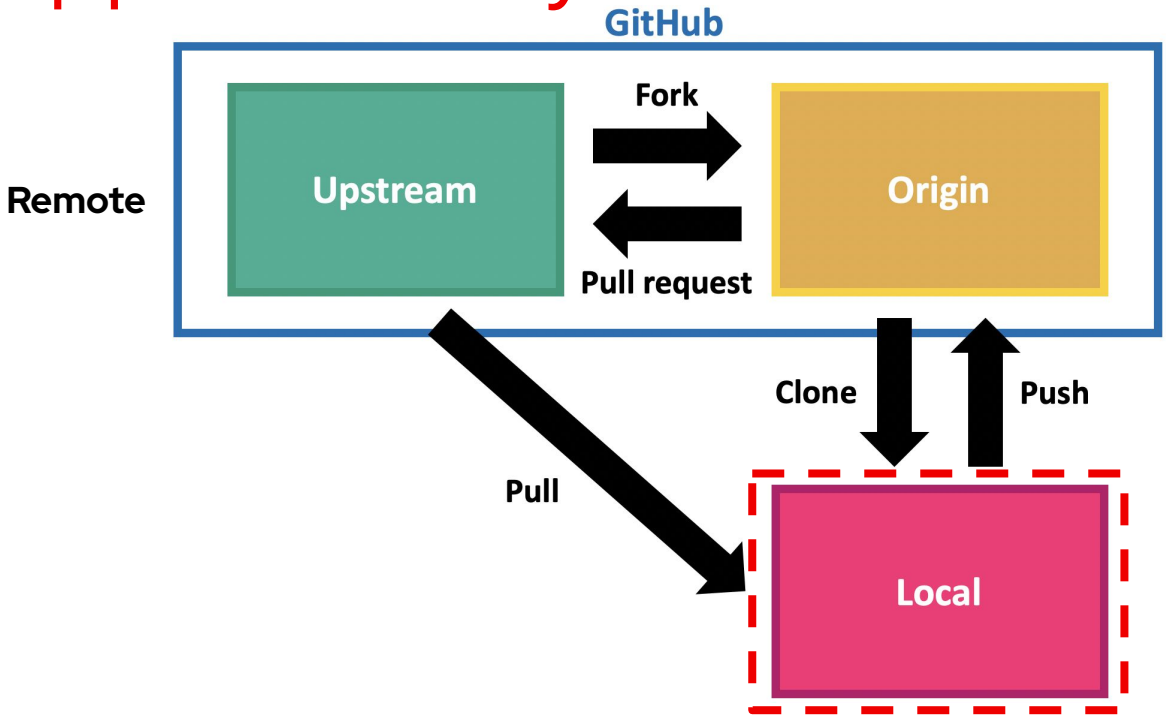


Red Hat

Git workflow



What happens locally?



Install git locally (Linux)

```
$ $package_manager $install_command git
```

E.g: `dnf install git`

On Windows

https://gitforwindows.org/

On Mac

```
$ brew install git
$ sudo port install git
```

Or a binary package shipped with Xcode

git config

git config

Sets conf variables determining git behavior

- System
- Global
- Local

git config

Sets conf variables determining git behavior

- System – all users and all repositories
- Global – current user and their repositories
- Local – specific repository

git config

A	System	1.	`.git/config`
B	Global	2.	`[path]/etc/gitconfig`
C	Local	3.	`~/.gitconfig or ~/.config/git/config`

git config

- System – all users and all repositories

  `[path]/etc/gitconfig`

- Global – current user and their repositories

  `~/.gitconfig or ~/.config/git/config`

- Local – specific repository

  `.git/config`

$ git config

Configure git with config command or directly editing the conf file.

- Identity
- Editor
- Commit
- Default branch name
- Merge tools
- Colored outputs (formatting and whitespace)
- Aliases

$ git config

Configure git with config command or directly editing the conf file.

```
$ git config –-list <--show-origin> <--system|global|local>
$ git config --global user.name "Mary Jane"
$ git config --global user.email mjane@example.com
$ git config --global init.defaultBranch main
```

```
$ git config --global --unset user.name
```

$ git config

Configure git with config command or directly editing the conf file.

```
$ git config core.editor

$ vi ~/.gitconfig
```

$ git config - Tricky question

If one git conf variable defined multiple times, which command will show the final say? For example, for a case of core.editor?

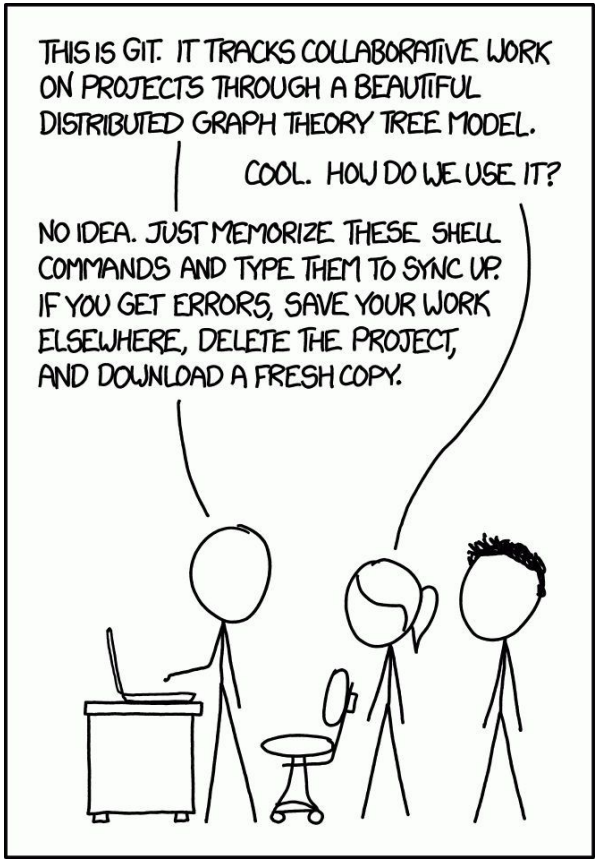
$ git config - Tricky question

If one git conf variable defined multiple times, which command will show the final say? For example, for a case of core.editor?

```
$ git config --show-origin core.editor
```

Documentation

- ~~Google~~
- ~~Stackoverflow~~
- `$ git <verb> --help and git help <verb>, e.g.`
  - `$ git help config`
  - `$ git config --help`
- `$ man git`
- `$ man git-<verb>, e.g. $ man git-config`
- User manual: file:///usr/share/doc/git/user-manual.html
- Pro Git Book by Scott Chacon and Ben Straub: https://git-scm.com/book/en/v2



XKCD Source: https://explainxkcd.com/wiki/index.php/1597:_Git

Git repository

- Create a new git repository locally
- Clone an existing repository

Create a local git repository

```
$ mkdir my_first_git_project

$ cd my_first_git_project

$ git init
```

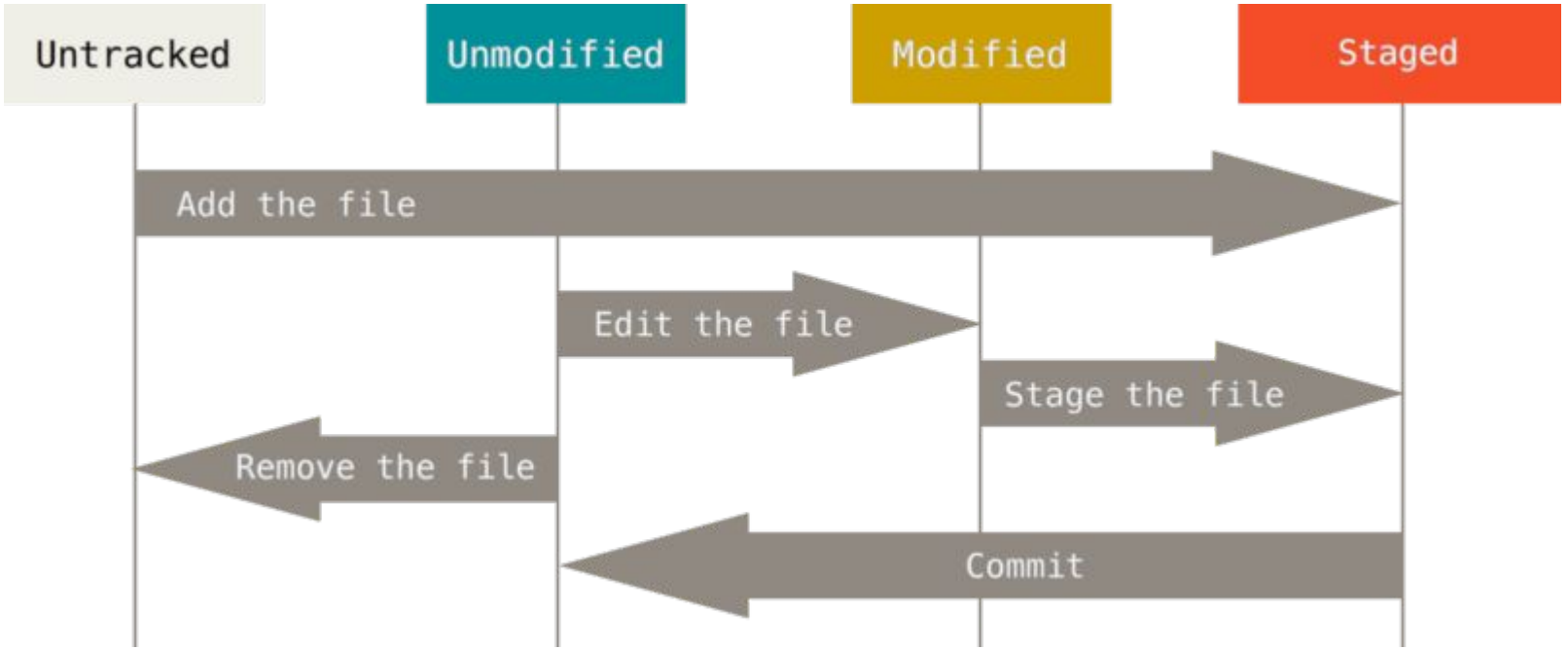
```
$ git init <directory>
```

git init

- convert an existing, unversioned project to a Git repository
- initialize a new, empty repository

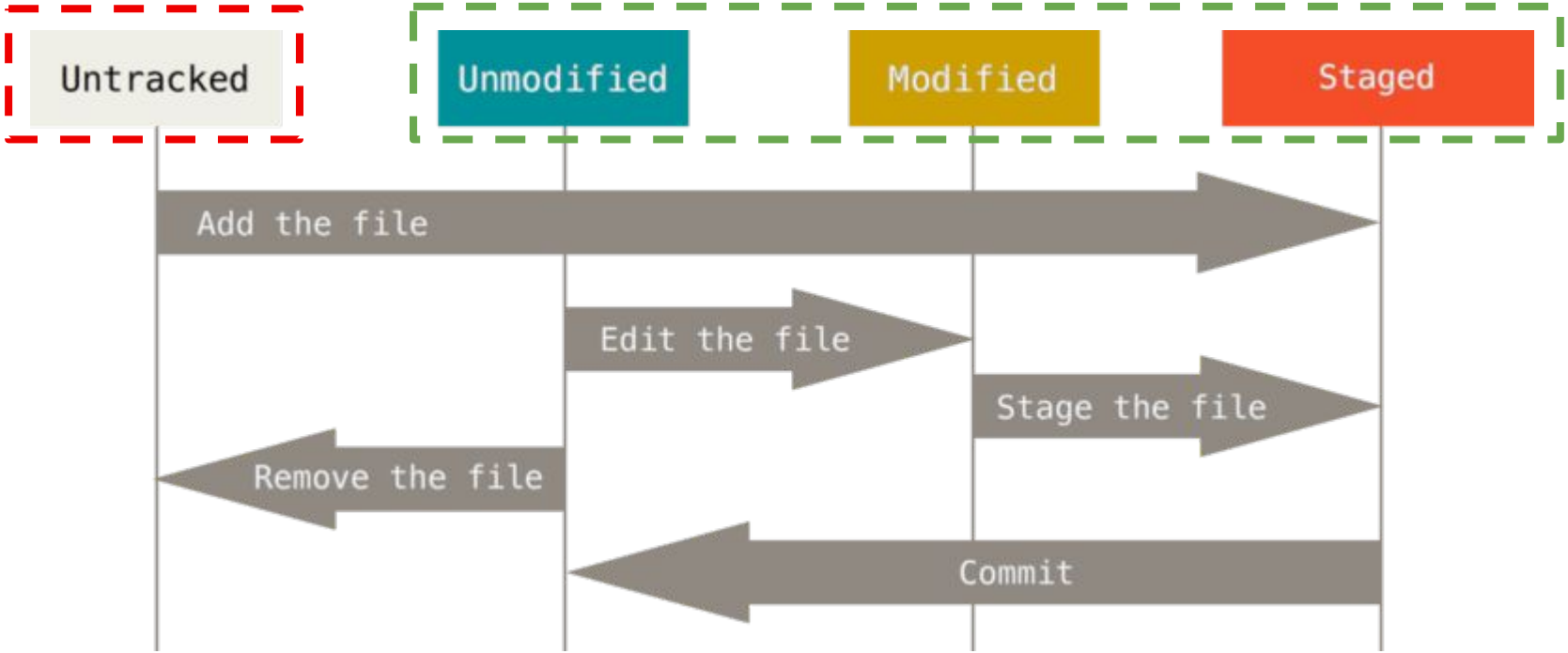
creates a .git subdirectory in the current working directory with metadata

Record changes to the repository



Reference: https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

Record changes to the repository



Reference: https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

What does this command do?

```
$ git status <-s>
```

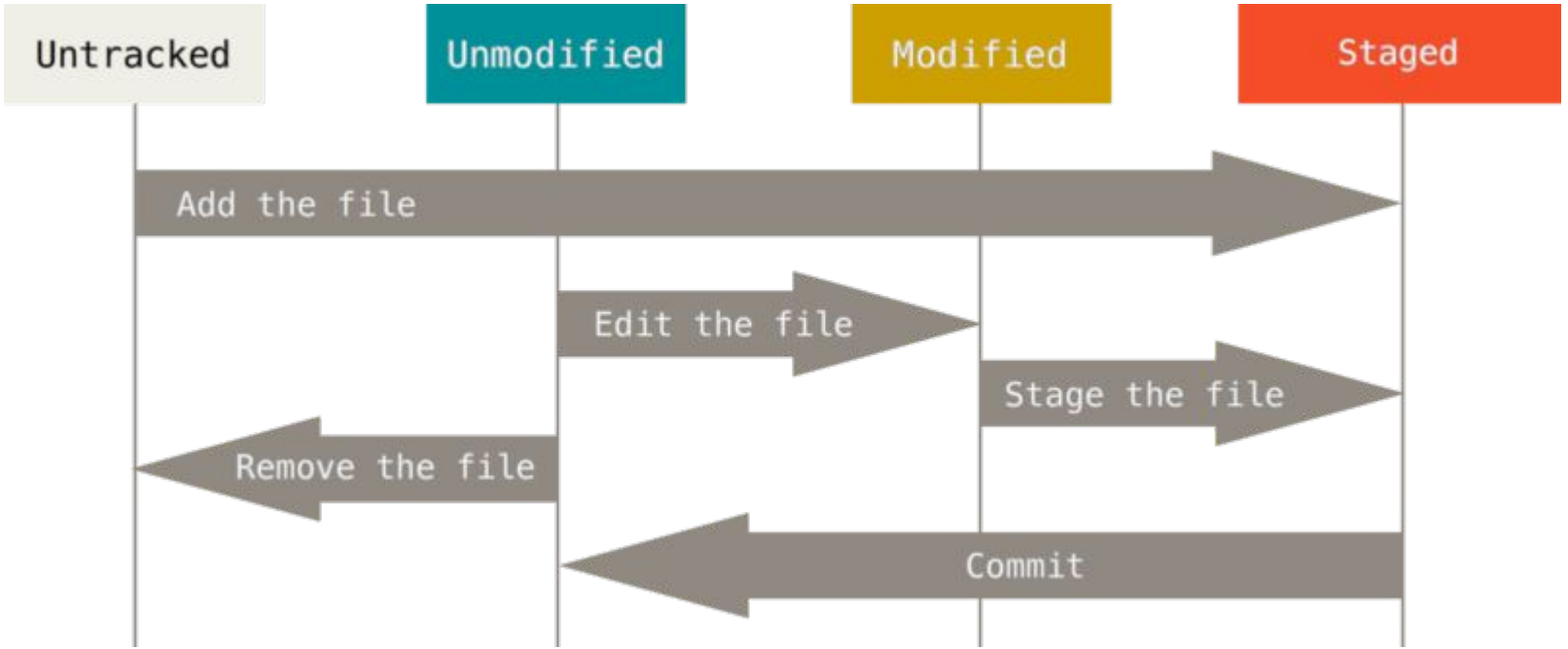

Reference: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Create file

```
$ echo ‘Mastering Git. First Lecture.’ >
README.md
```

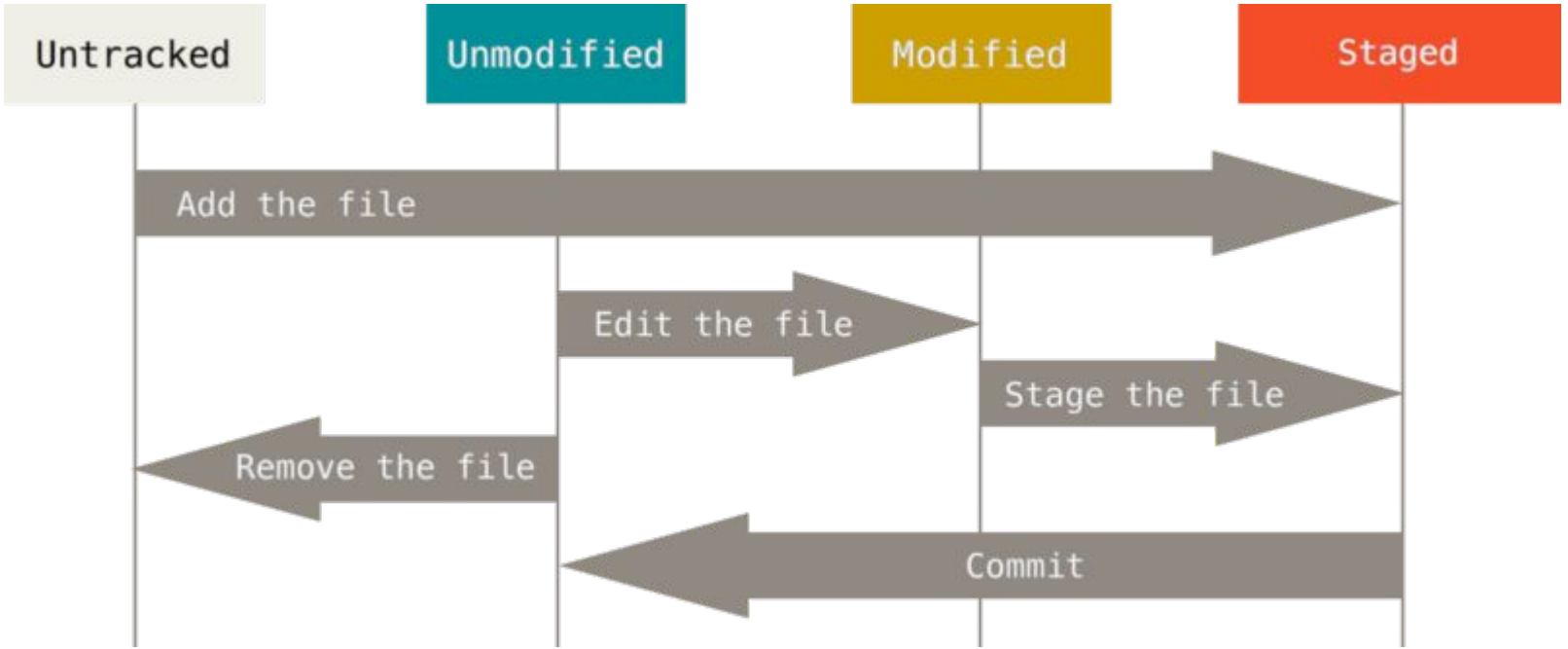

Reference: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

$ echo ‘Mastering Git. First Lecture’ > README.md



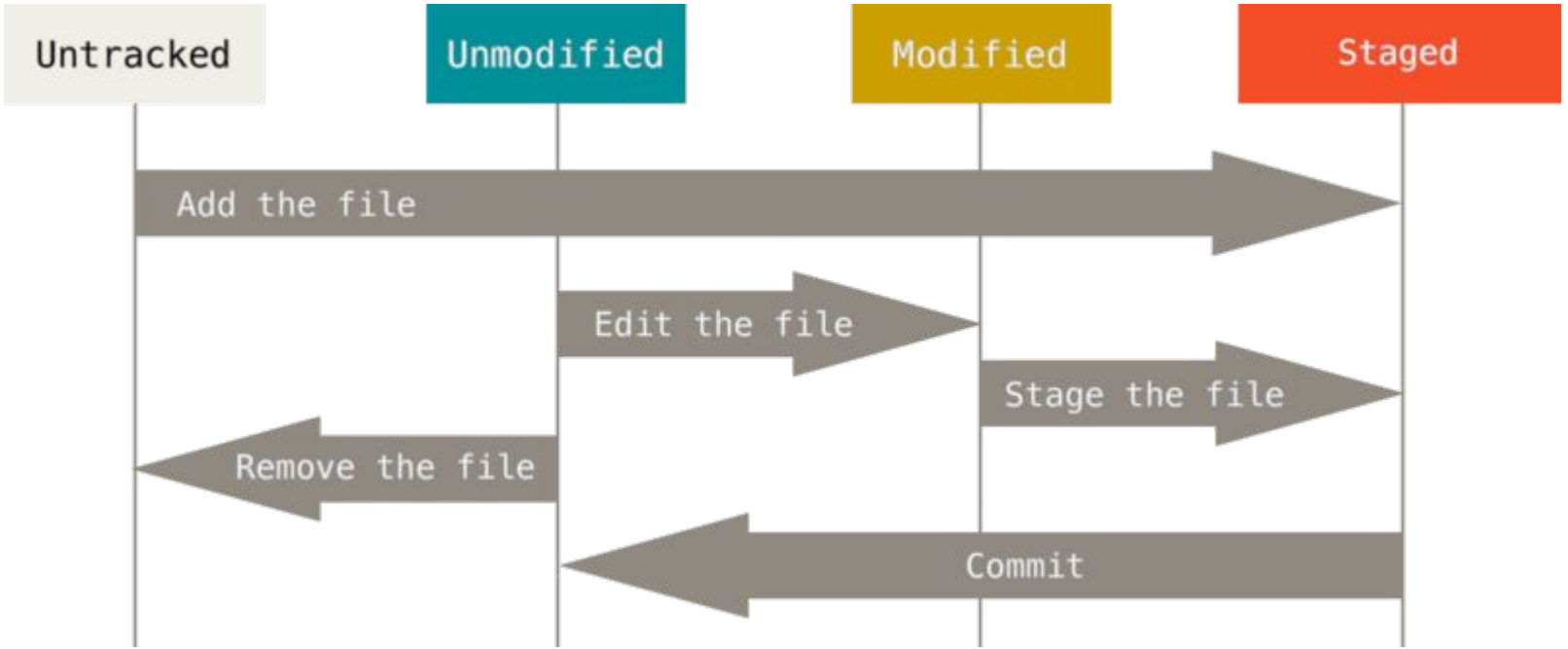
Reference: https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

$ git add README.md



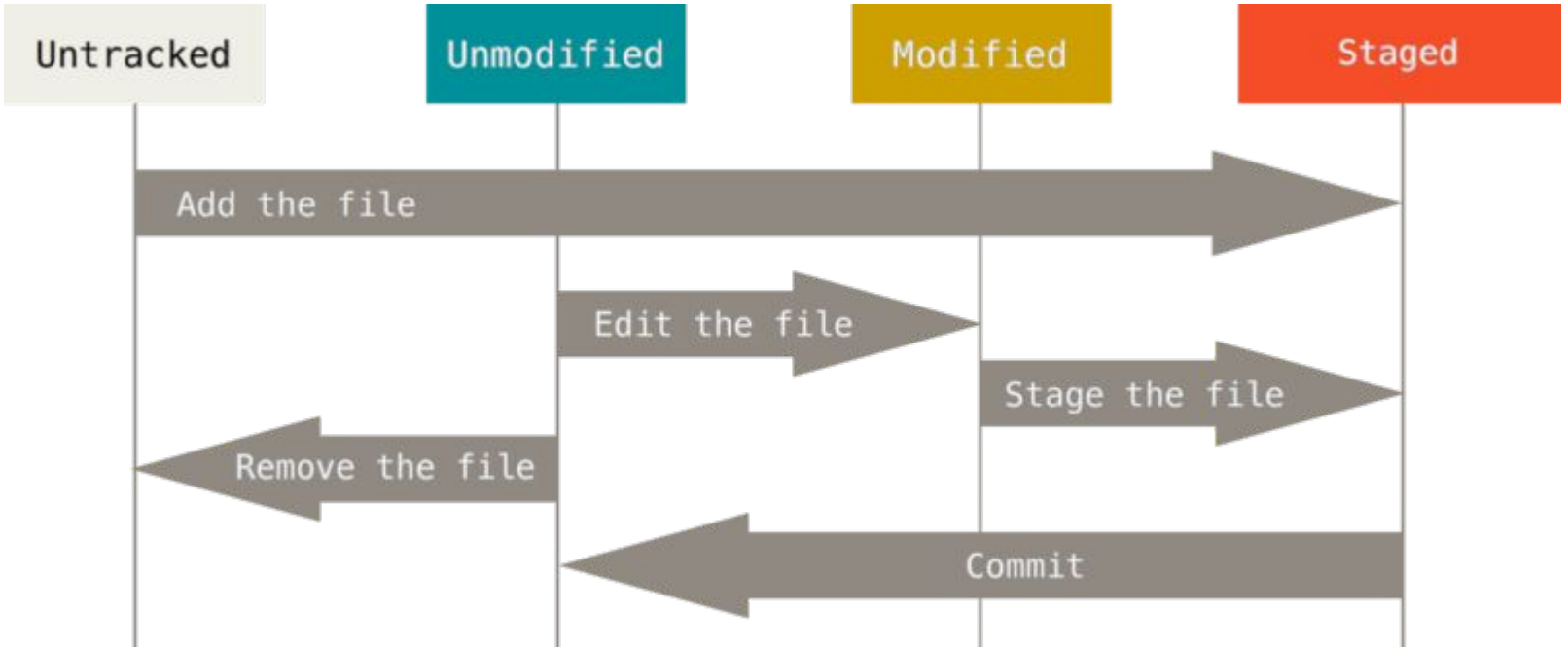
Reference: https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

$ git commit -m “Initial commit”



Reference: https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

```
$ echo $'\nThis is a new line' >> README.md
```



Reference: https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

Local workflow

Reference: https://stackoverflow.com/a/3690796/909579

First commit

$ echo ‘Mastering Git. First Lecture’ > README.md

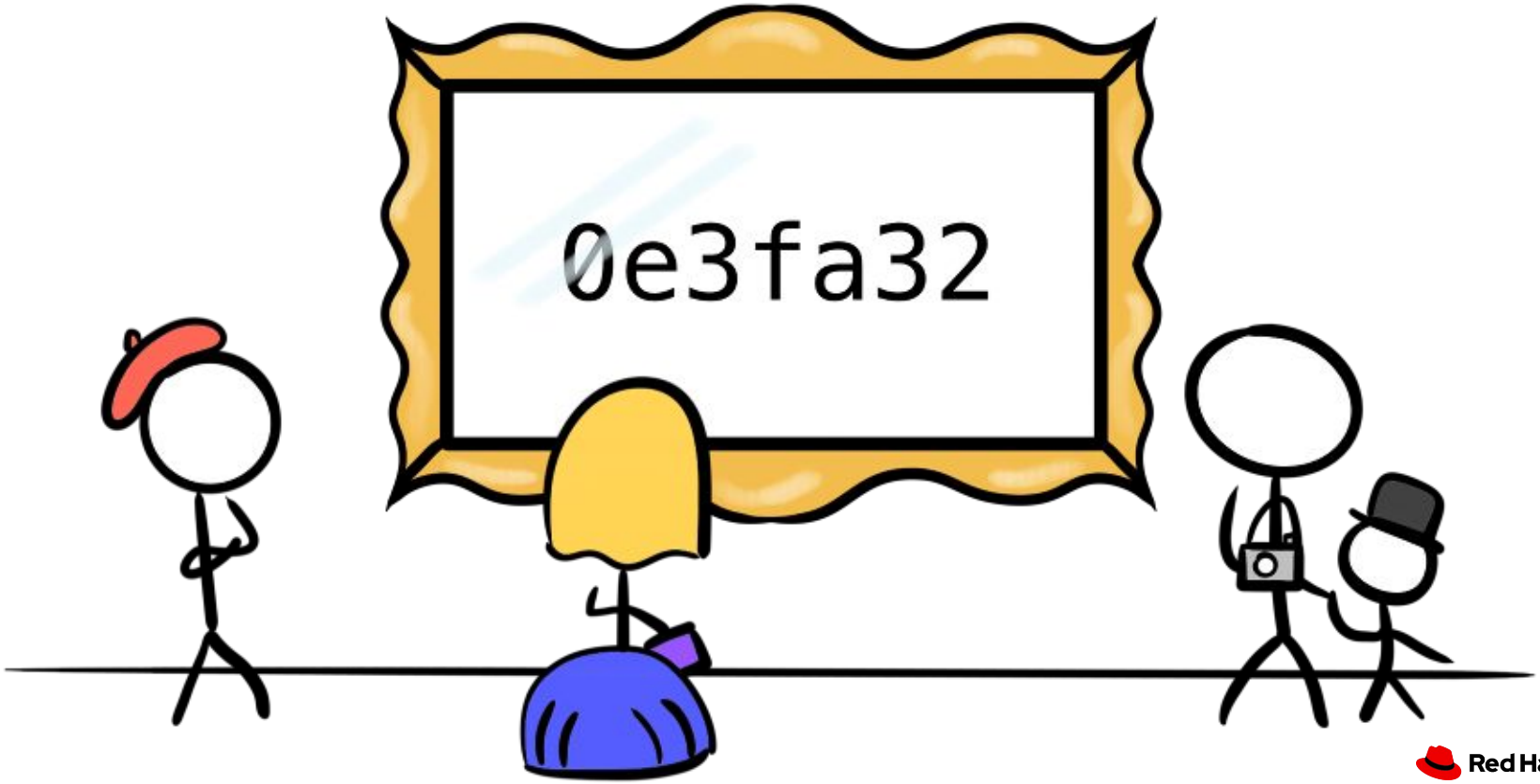
$ git status

$ git add README.md

$ git status

$ git commit -m “Initial commit”

Reference: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git



Red Hat

Commit ID = SHA-1 hash

```
$  git commit -m 'init commit'

[main c6b75cd] init commit

<...>

$ openssl zlib -d <
./.git/objects/61/8c0a1b1ed51b1f7a3456fc135311331ce41dbf | sha1sum
c6b75cd8d61e5be49fa4f23ee2025fed952e0aef  -
```

Commit ID content

$ openssl zlib -d <
./.git/objects/61/8c0a1b1ed51b1f7a3456fc135311331ce41dbf | sha1sum
c6b75cd8d61e5be49fa4f23ee2025fed952e0aef -

- Full content
- The ID of the previous commit or its merge
- *Commit and author* date
- *Committer and author's* name and email addresses
- Log message

Commit ID content

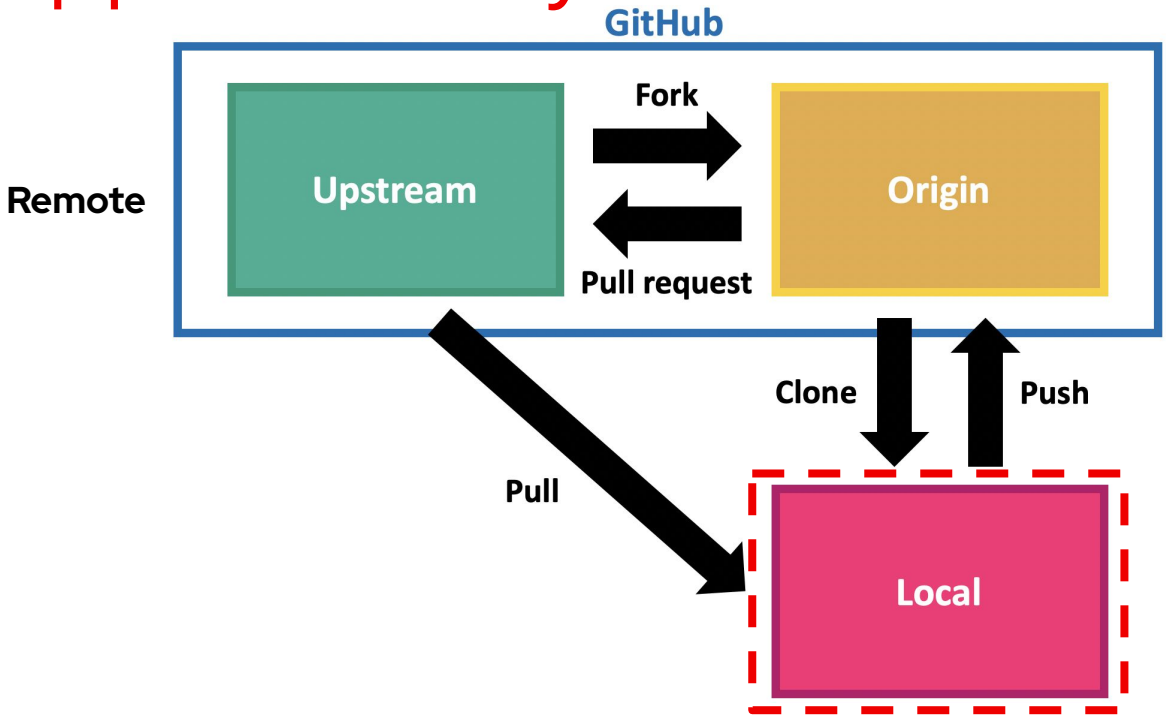
$ openssl zlib -d < ./.git/objects/61/8c0a1b1ed51b1f7a3456fc135311331ce41dbf | sha1sum

c6b75cd8d61e5be49fa4f23ee2025fed952e0aef -

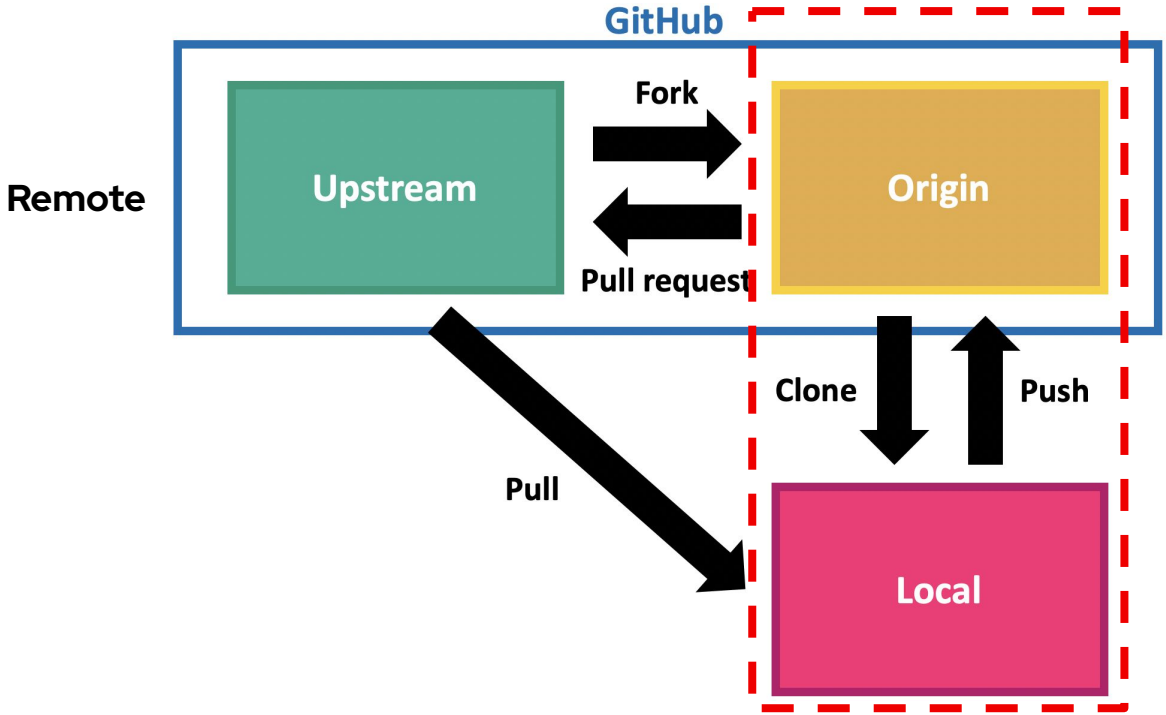
commit c6b75cd8d61e5be49fa4f23ee2025fed952e0aef
parent 6294ij1k53w43uf2fd94168ac410eca1e98 er34q
Author Irina Gulina <igulina@redhat.com> 1659644787 -0700
committer Irina Gulina<igulina2redhat.com> 1659644787 -0700

feature: my awesome feature

What happens locally?



Git workflow



What VCS hosting platforms do you know?

Origin + Local

- Create a new repository on GitHub (remote origin)
- Add remote origin locally
- Push the changes to remote origin

Origin + Local

- Create a new repository on GitHub (remote origin)
- Add remote origin locally
  - `$ git remote add origin git@github.com:<userID>/<repo_name>.git`
- Push the changes to remote origin

```
$ git branch -M main
$ git push -u origin main
```

Clone someone's git repository

$ exit from the previous repository (cd ..)

$ git clone https://github.com/libgit2/pygit2

Clone vs git init

`git clone` is dependent on `git init`

Local + Origin remote

Do it yourself, connect locally cloned repo with its origin remote version...

Delete and move files

```
$ git rm <file_name>

$ git mv <file_name>
```

Add files

```
$ git add --all
$ git add -A
$ git add .
```

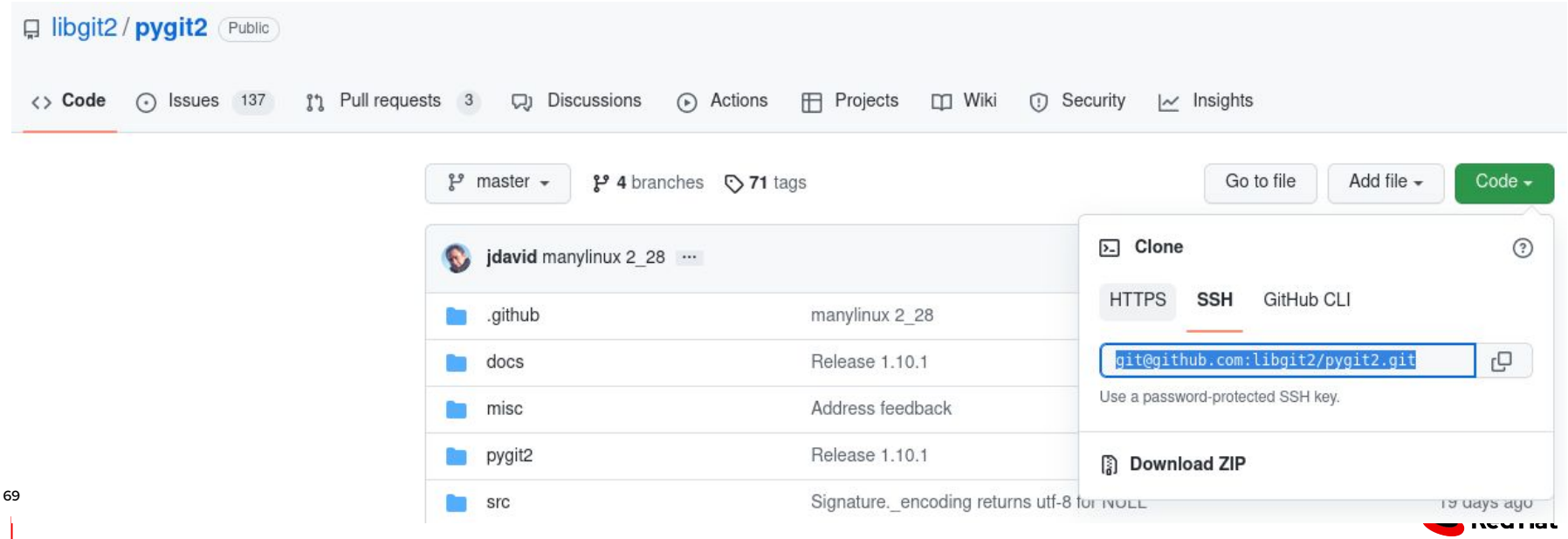
Add files

```
$  git commit -m 'repo cleanup'
[main c6b75cd] repo cleanup
<...>
 delete mode 100644 foo
 rename myfile.txt => roles/myfile.txt (100%)
```

HTTPS or SSH?

Clone someone's git repository

$ git clone git@github.com:libgit2/pygit2.git



HTTPS or SSH?

- Both are communication protocols.
- Both work for providing a reliable and secure connection
- Encrypted

Reference: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

HTTPS

- Smart - HTTPS
- username/password authentication
- Pros and Cons
  - Fast, efficient, firewalled approved
  - Writing username/password for authentication

Reference: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

SSH

- Smarter
- Private key-based authentication
- Pros and Cons
  i. Fast, efficient, firewall approved
  ii. One-time association: your key in a git forge
  iii. Auth is mandatory

Reference: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Do NOT use git:// and http://

No homework but requirements:

- gitlab.com account
- SSH key is in your account
- class repo is forked
- We have a UCO - GitLab mapping
  i. https://gitlab.com/redhat/research/mastering-git/-/issues/2
  ii. https://gitlab.com/redhat/research/mastering-git/-/issues/3

LAB

Bonus

How cool is Your README?

What is README?

Detailed description of a git project

- What is the project about
- What are the user cases
- How it is organized
- How to install and use it

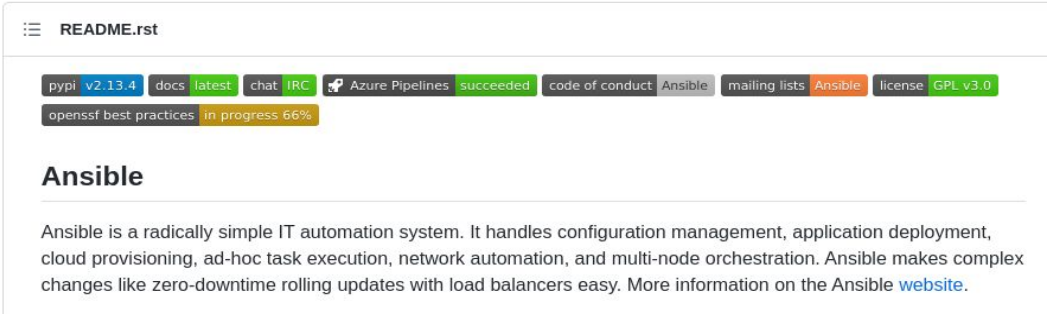
… and more

Why is README important?

- the first impression about your project
- improve engagement with the project
- help others get involved

How does a good README look?

- It should exist!
- No unique guide.
- Fairly brief but detailed.



- Answer **what**, **why**, and the **how** of the project.
  - ✓ TItle, description, demo, table of contents (optional), how to install and run the project, how to use the project - examples, user/pass requirements, credits, references, license (ref), badges (optional), howto contribute and project organization, **TESTS**, feedback, buy_me_coffee…

How does a good README look?

- Check out big popular projects. Use templates and generators, e.g.: https://github.com/kefranabg/readme-md-generator
- Located at the top level of the project directory
- Keep it up-to-date
- Peek a language
- Format and structure it

Questions?

We have Questions!

THANK YOU!