Course Introduction

Bryan Krausen

WELCOME

INSTRUCTOR

Bryan Krausen

Welcome! I'm a Premier Udemy Instructor and a seasoned technologist (25+ years) who is passionate about cloud infrastructure and security. I help organizations use technology to solve real-world problems.

← CONNECT WITH ME

Why Trust Me?

I've always been a teacher at heart and have taught hundreds of thousands of people

I Consult with VERY Large Organizations

I've designed, implemented, fixed, and managed it all ☺

I've used Git Throughout my Entire Career

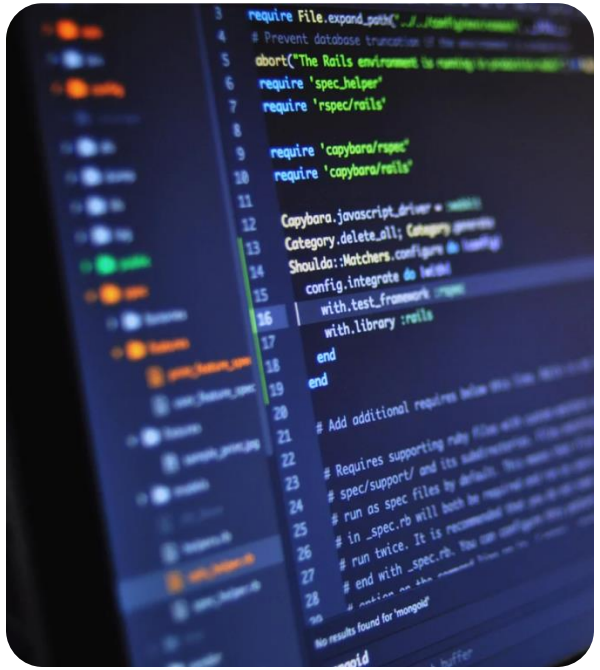
And you probably will, too!

I'm VERY Involved with the Community

I've been part of many community-based programs and spoken at many large events

How Do We Get There?

(Short) Lectures to Build Understanding



Demos for Real-World Examples

Hands-On Labs to Apply Skills

Course Goals

Get you working with Git fast without taking too much of your time.

Get familiar with the Basic Git Concepts

Quickly get up to speed on basic Git terms, features, and usage

Learn the Fundamental Git Workflow

Learn how to start a new project and start working with existing projects.

Collaborate with Others to Develop Code

Learn how to put all these skills to work and work with others in your organization

COURSE OVERVIEW

INTRODUCTION

Learn about common terminology and usage of Git

PREP YOUR ENVIRONMENT

Install and Configure Git and related tools

GIT BASICS

Learn the basic functionality of using Git in your environment

BEYOND THE BASICS

Take it a step further and learn additional features and commands

COLLABORATION

Learn strategies and features to iterate on code and work with team members

ADDITIONAL CONTENT

More content so you can keep improving on your skills

What is Git?

Bryan Krausen

The best way to learn Git is to first do only very basic things and not even look at some of the things you can do until you are familiar with and confident about the basics...

Linus Torvalds
Git Creator (and the Linux kernel)

What is Git?

Git is a distributed version control system (VCS) that tracks file changes. It enables multiple developers to collaborate effectively and work on the most up-to-date version of the project.

Branching

Create a separate copy of the files that you (or a team) are working on

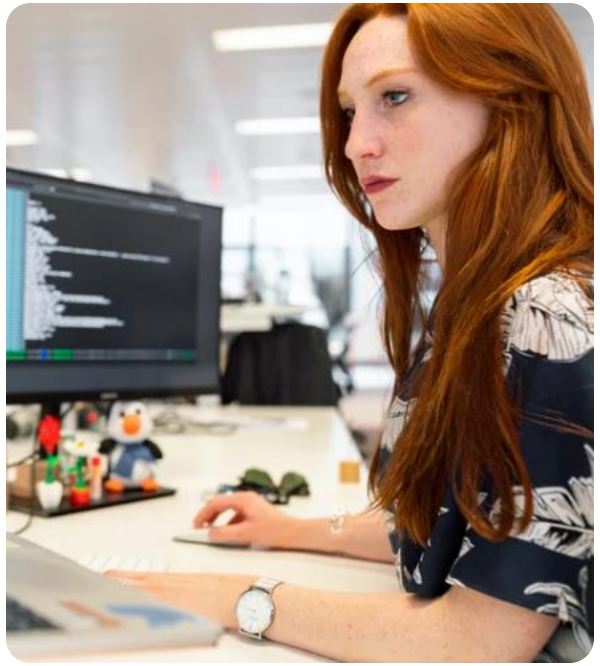
Edit Files Separately

Edit files independently in your personal branch without impacting the original files or other people's work

Merge in Changes

Safely bring your changes back into the main copy of files so your changes don't impact others.

Real-World Uses of Git

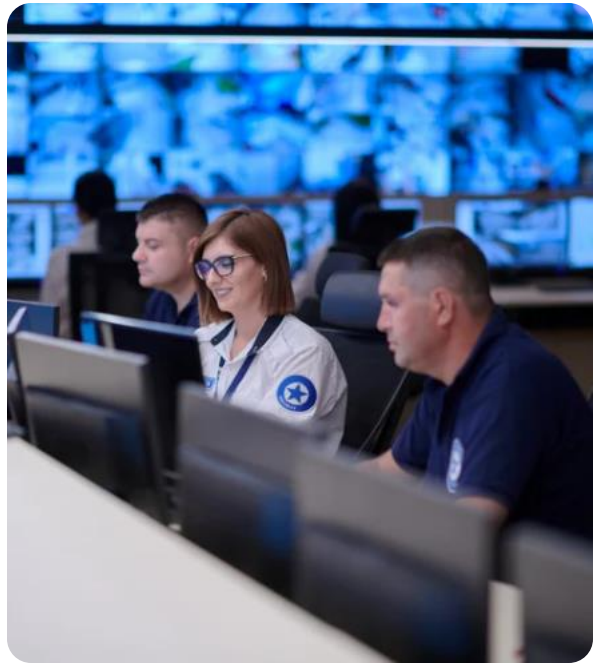


Software Developers

Uses Git to save work, fix mistakes, and build features safely.

DevOps Engineers

Manage infrastructure as code (Terraform, Ansible), pipelines, and configuration files.



AI/ML Engineers

Uses Git to track experiments, manage model code, and share results with the team.

Real-World Uses of Git

Local Machine
(Project Files/Code)

Git Repository
(Project/File Storage)

Common Terminology

Objects or Concepts

Repository

A place for your project that contains all files and versions.

Commit

A saved snapshot of your code at a point in time

Branch

Your own copy of the repository files used to work independently.

Pull Request

Request for your code to be merged into another repo or branch

Remote

A version of your repo hosted somewhere (GitHub, GitLab, etc.)

Staging

When changes go before committing them.

Common Terminology

Actions – Things You Do with Git

Clone

A local copy of a repository.

Push

Upload your commits from your local copy to the repo.

Commit

A snapshot of your changes, a version of your project.

Pull

Fetch and get changes from a repository to your local copy.

Merge

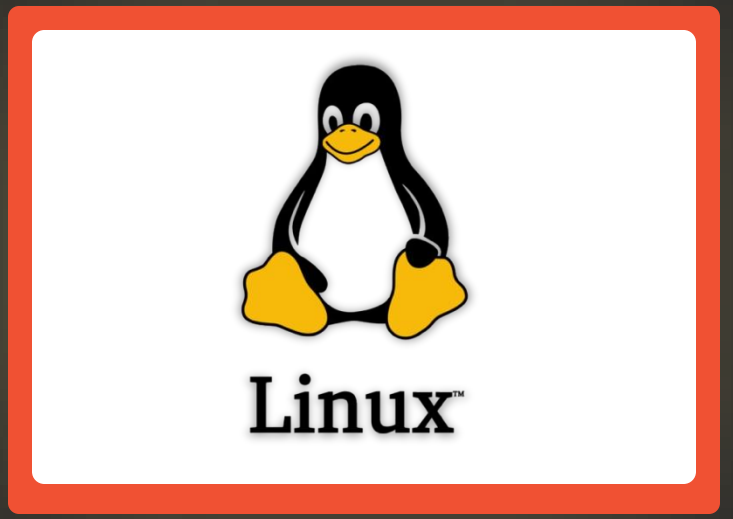
Combine changes from one branch to another branch.

Fork

A local copy of a public repository you can use to make changes.

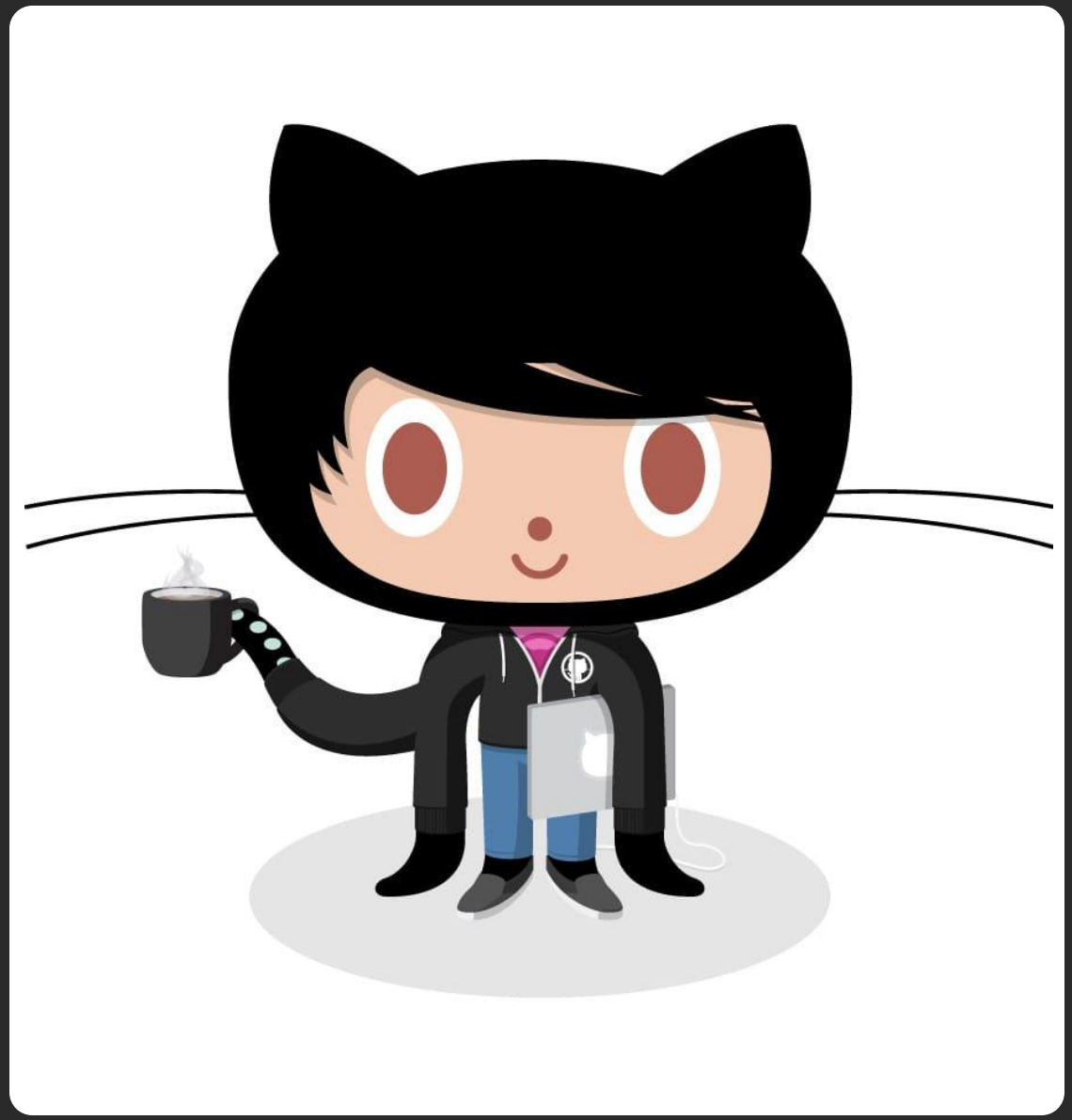
Platform Support

Install it where you need - you'll have an identical workflow regardless of platform



Difference Between Git and GitHub

Bryan Krausen



What is GitHub?

A cloud-based platform where you can store, share, and work together with others to write code.

- **Store Your Code in Hosted Repositories**
  Let GitHub store your code in private or public repositories.
- **Track and Manage Changes to Your Code**
  Maintain and iterate on your code over time.
- **Collaborate on Projects**
  Let others review your code, make suggestions, and approve changes to the project.



Differences between Git and GitHub?

Location

Git works locally on your machine.

GitHub provides remote and cloud-based access.

Purpose

Git is a version control system.

GitHub is a platform for hosting and collaborating on Git repositories.

Collaboration

Git handles code tracking.

GitHub enables team workflows like pull requests, reviews, and issue tracking.

What Else Does GitHub Offer?

GitHub is much more than just Git Repositories

Organizations (collaboration)

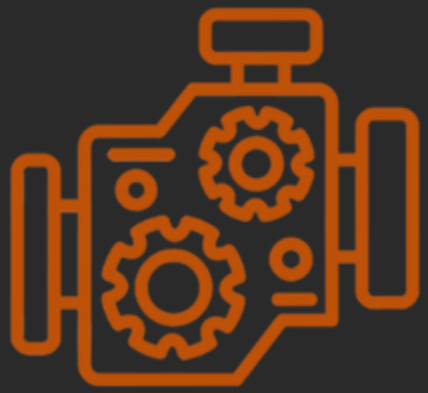
GitHub Actions (CI/CD)

Security Features

Technical Support

Images: octodex.github.com

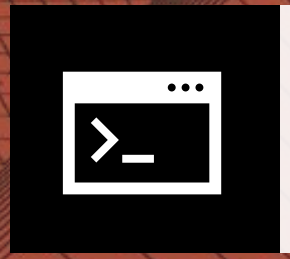
Summary



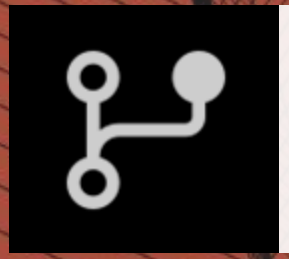
Git

Git powers GitHub

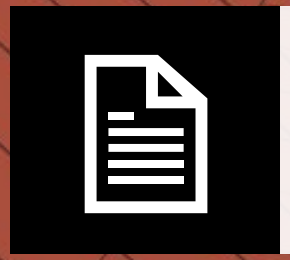
SUMMARY



Git is a local version control tool commonly used by developers DevOps, and other IT-related roles

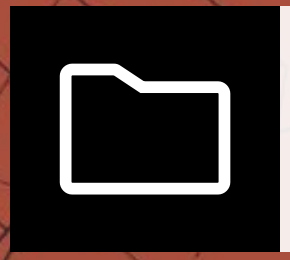


You'll work with repositories, branches, and commits which are Git-related objects



Git tracks changes to files and enables you to work collaboratively with teammates

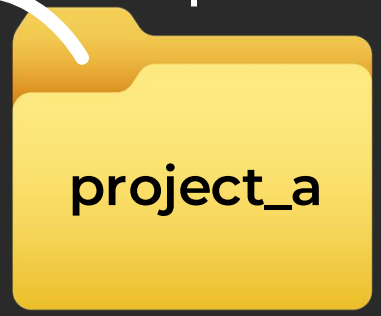
Git works offline, while GitHub adds collaboration and other features online.

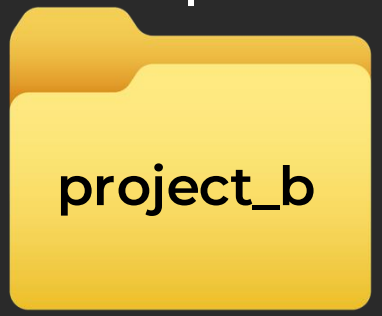


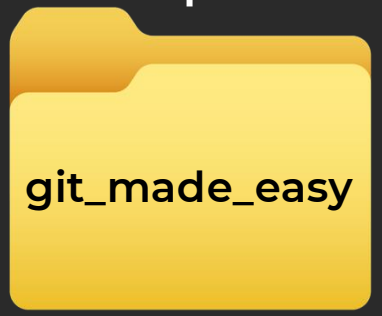
Project files are stored and managed in a Git repository using actions like push, pull, commit, merge, and fork.

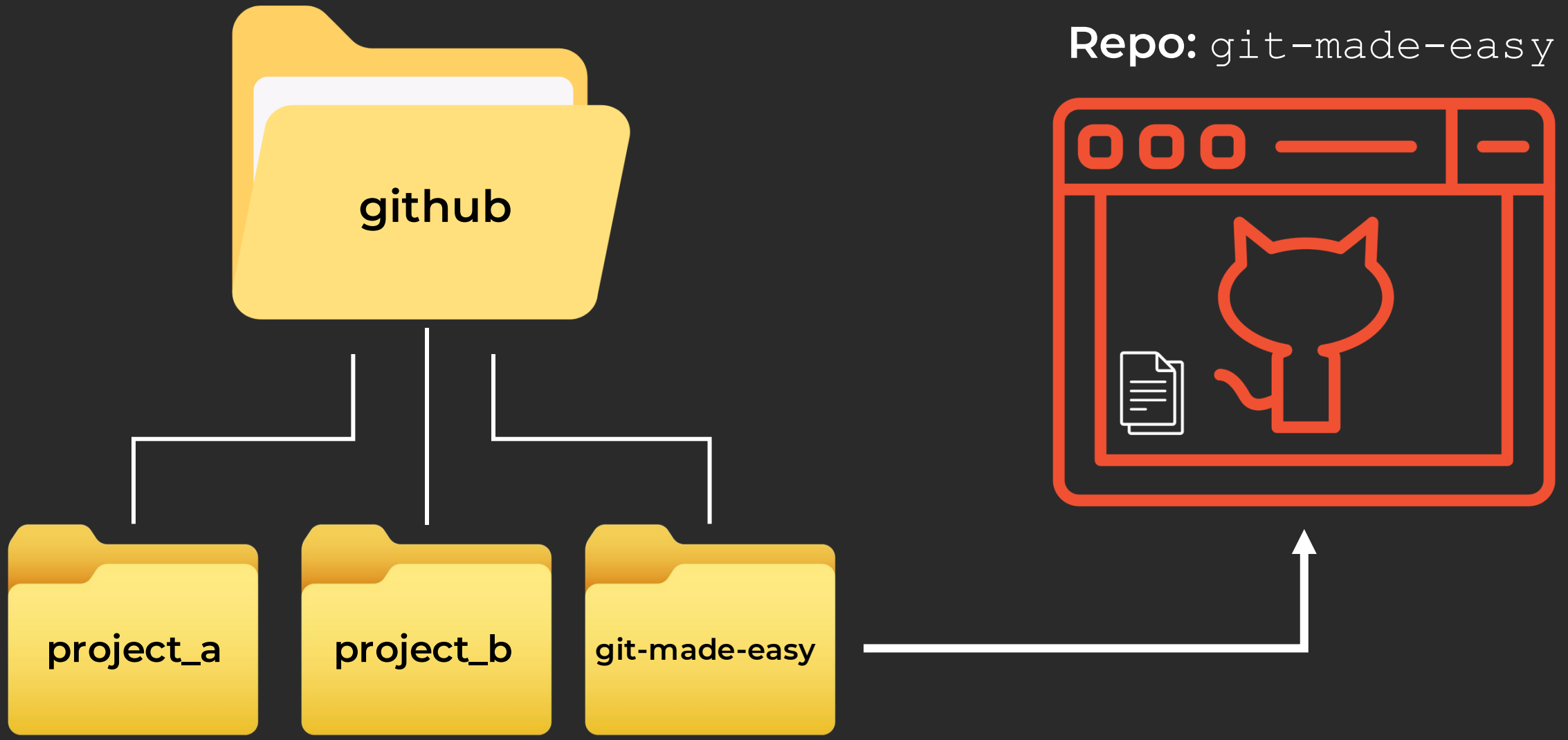
Git is the tool that enables platforms like GitHub, GitLab, BitBucket, and more.

Your Local Computer/Laptop





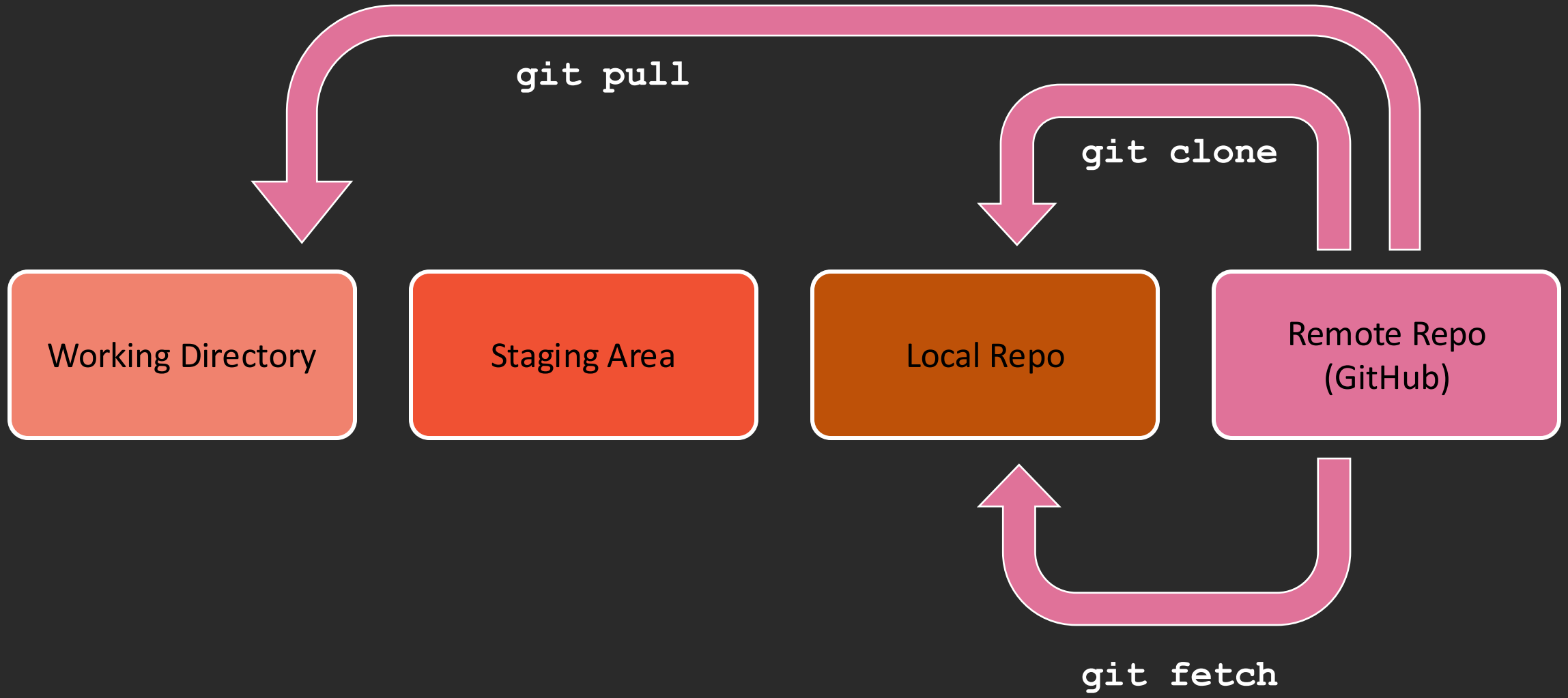




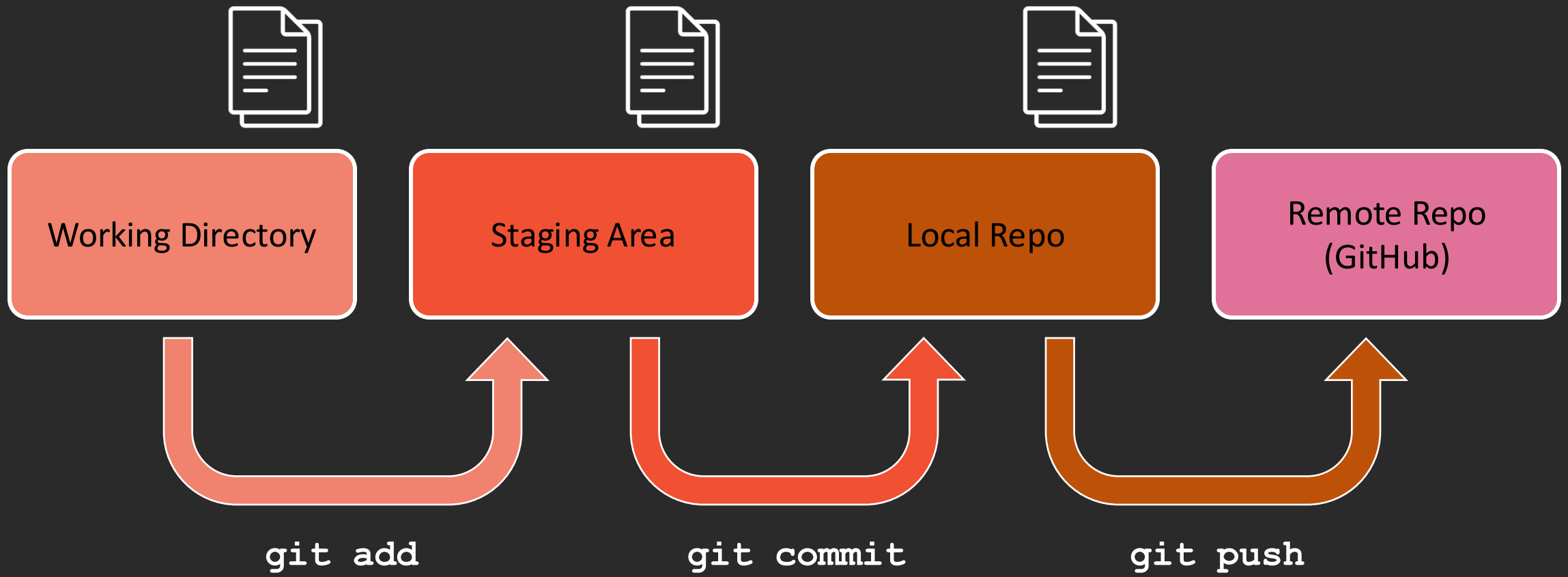
The Git Workflow

Bryan Krausen

Git Workflow – Remote to Local



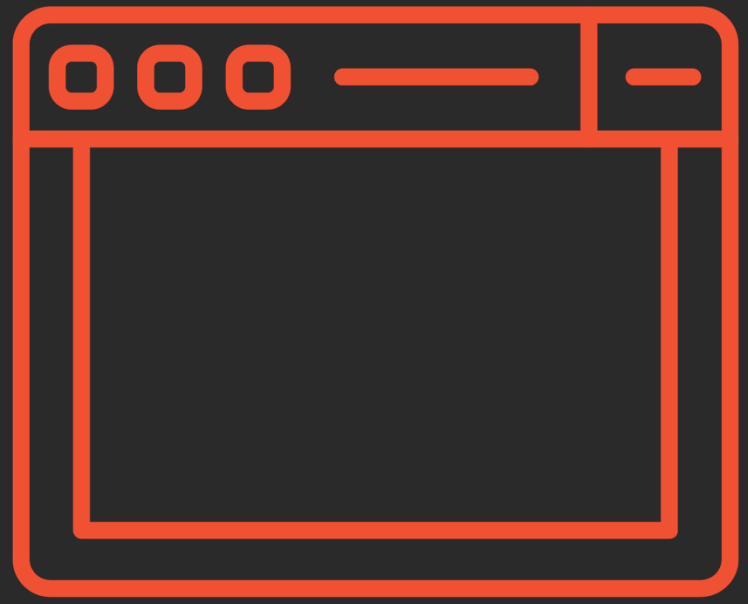
Git Workflow – Local to Remote



Branching

Bryan Krausen

Common Scenario



Your Local Computer/Laptop

Corporate Website

Your Local
Computer/Laptop

git push

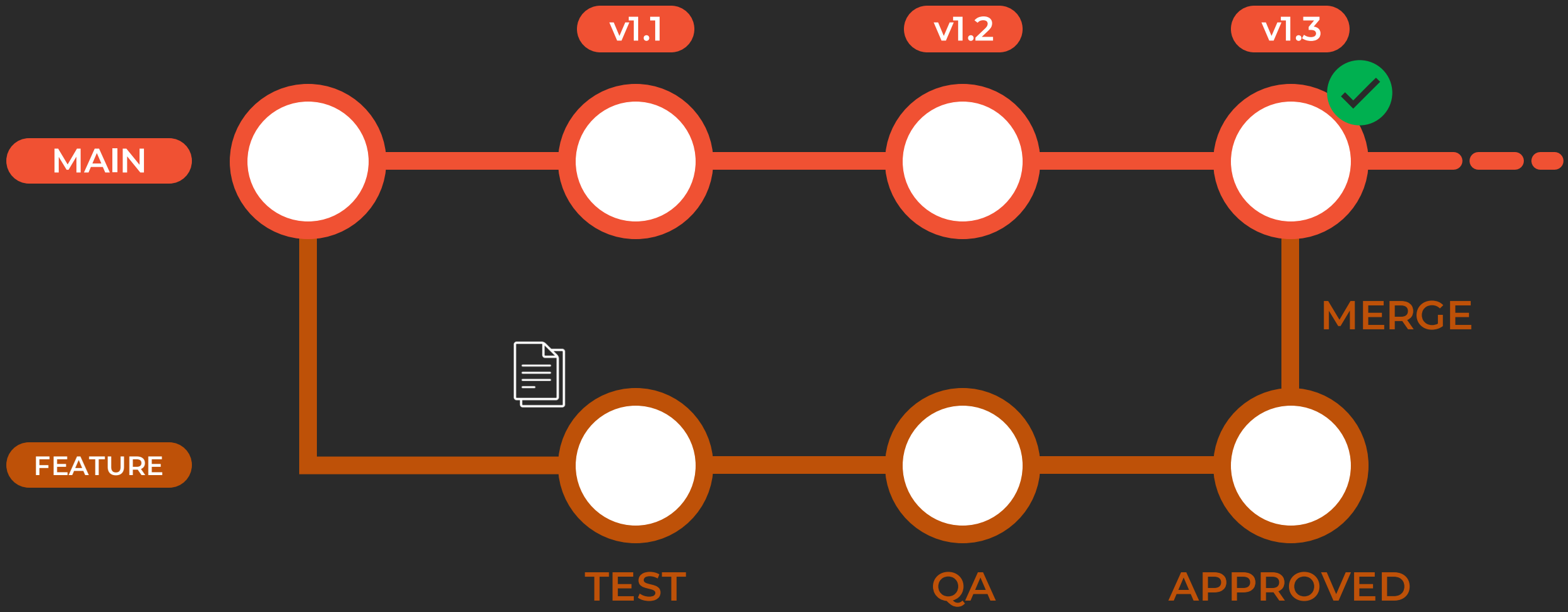
v1.1

v1.2

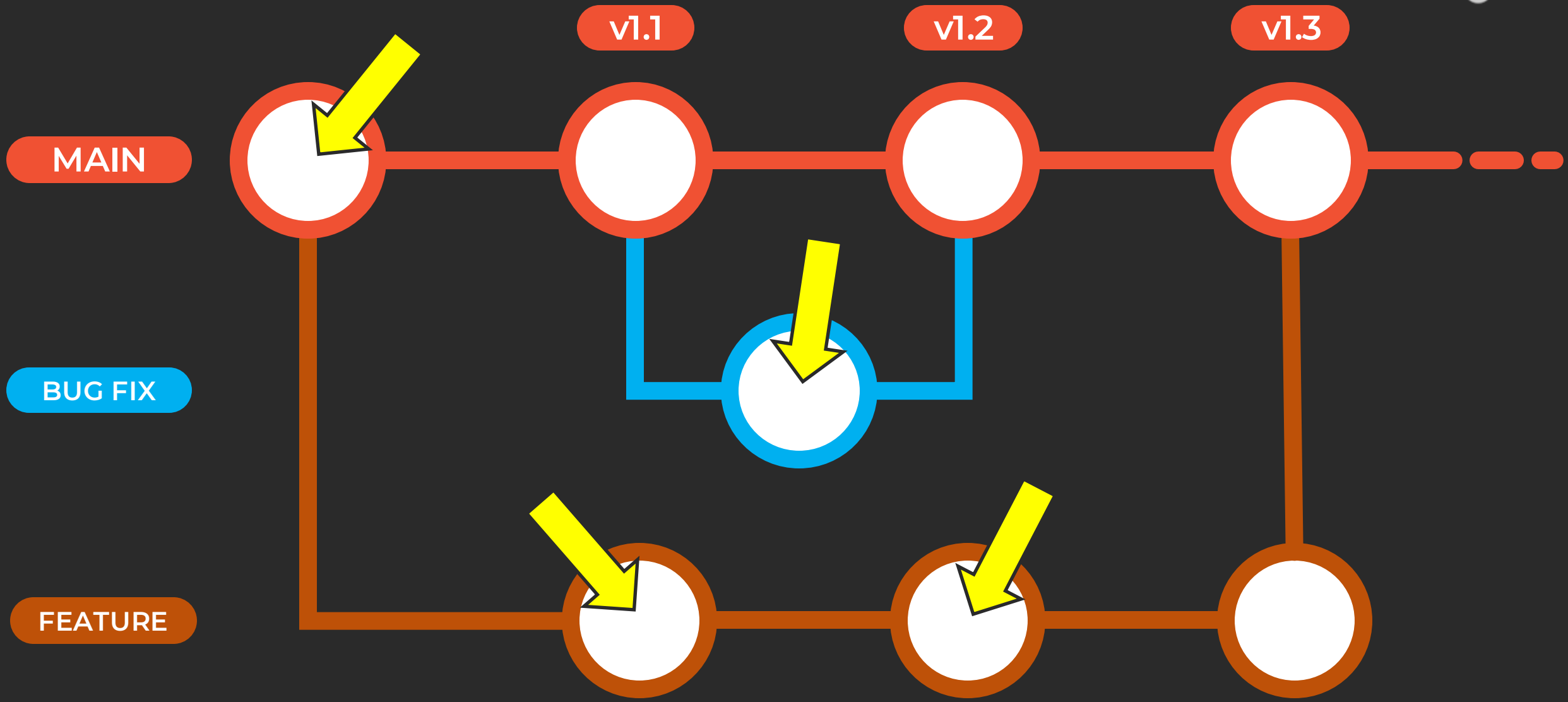
v1.3

MAIN

Branching



Branching

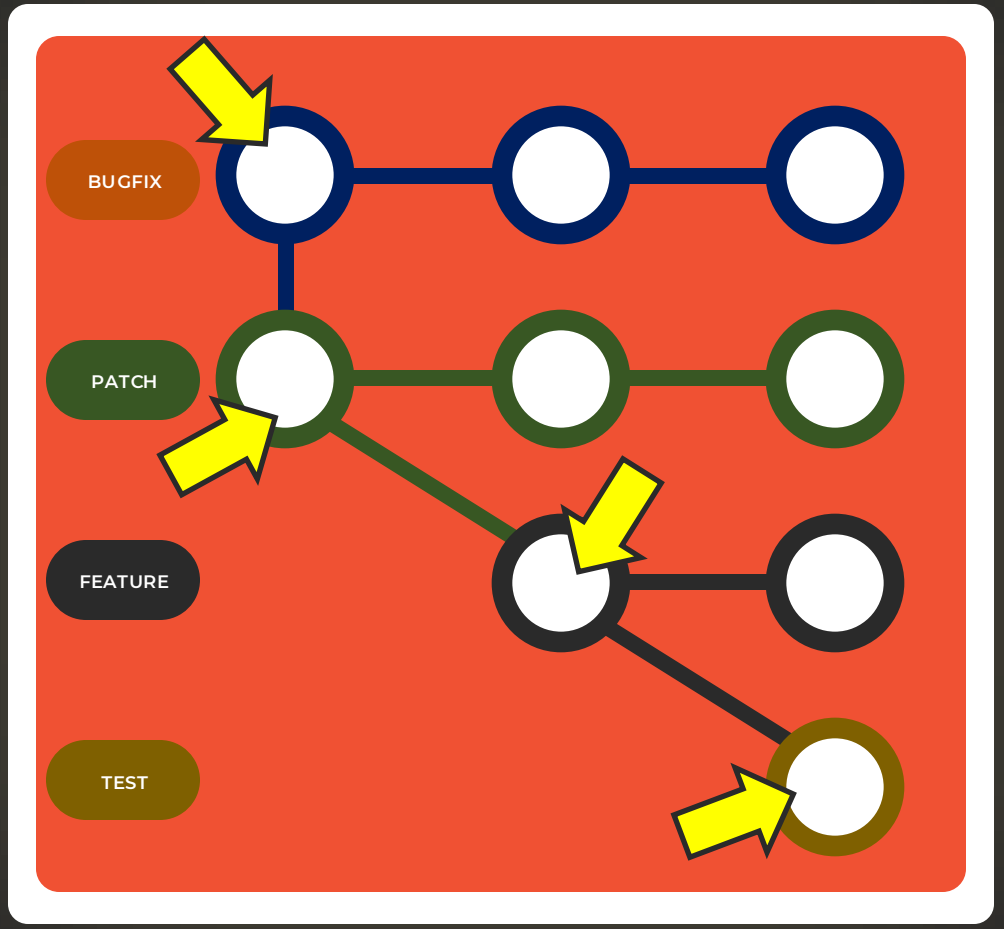


What is HEAD ??

HEAD is simply a pointer to your current branch

HEAD tells Git where you're working and makes sure your files match the current branch

When you switch branches, HEAD moves to point to the new branch.



Common Commands

Create a New Branch

TERMINAL

```
$ git branch <name>

$ git branch develop
$ git branch bug-fix
```

- creates a new branch based on the current branch
- does not switch to that branch

List Existing Branches

TERMINAL

```
$ git branch
  main
* develop
  bug-fix
```

- will list the branches on your local machine
- The * indicates the current branch – also known as HEAD

Common Commands

Switch to a Branch

TERMINAL

```
$ git switch <name>

$ git switch bug-fix
Switched to branch 'bug-fix'
```

- switches HEAD pointer to the desired branch
- updates the working directory to match the branch

Create & Switch to a New Branch

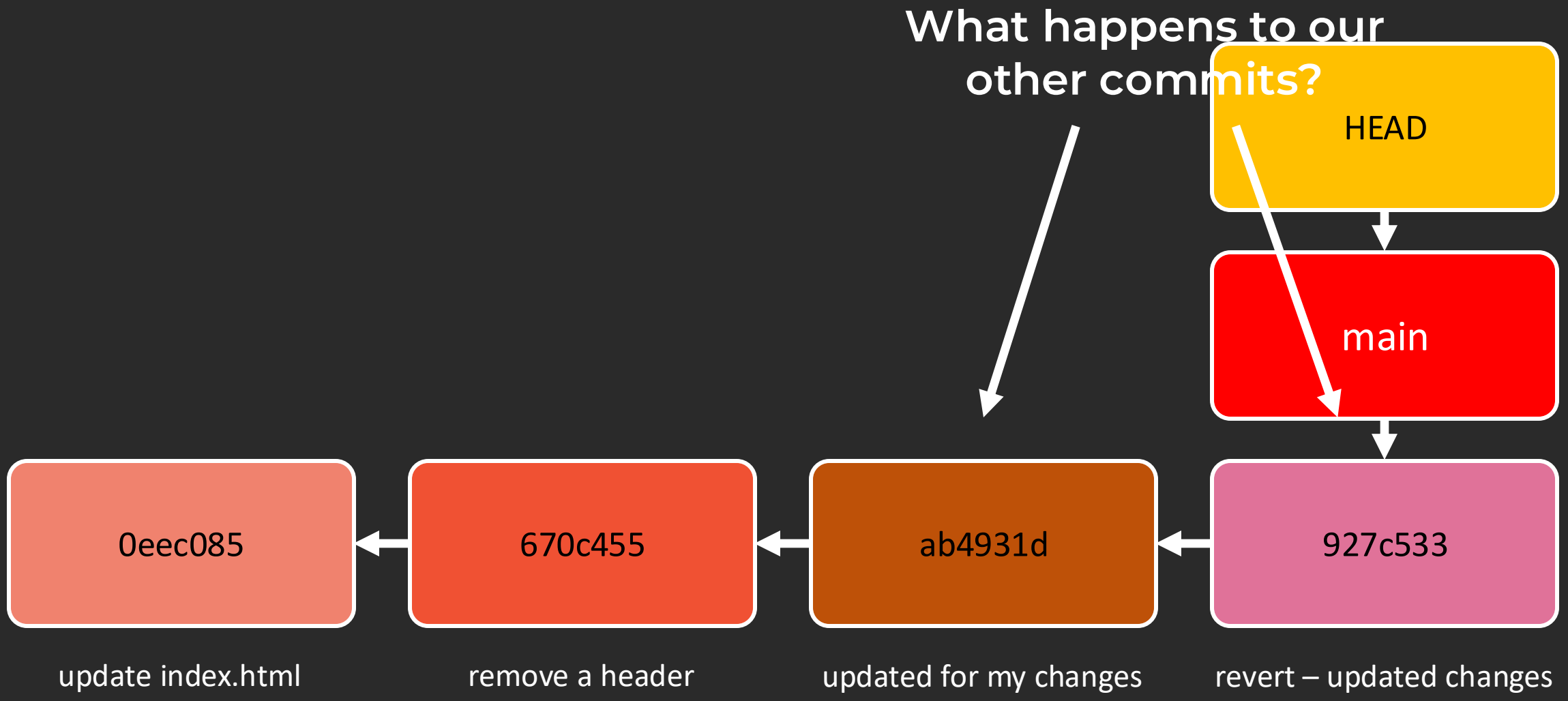
TERMINAL

```
$ git switch -c <name>

$ git switch -c patch
Switched to a new branch 'patch'
```

- creates a new branch
- immediately switch HEAD point to the new branch

Git Reset

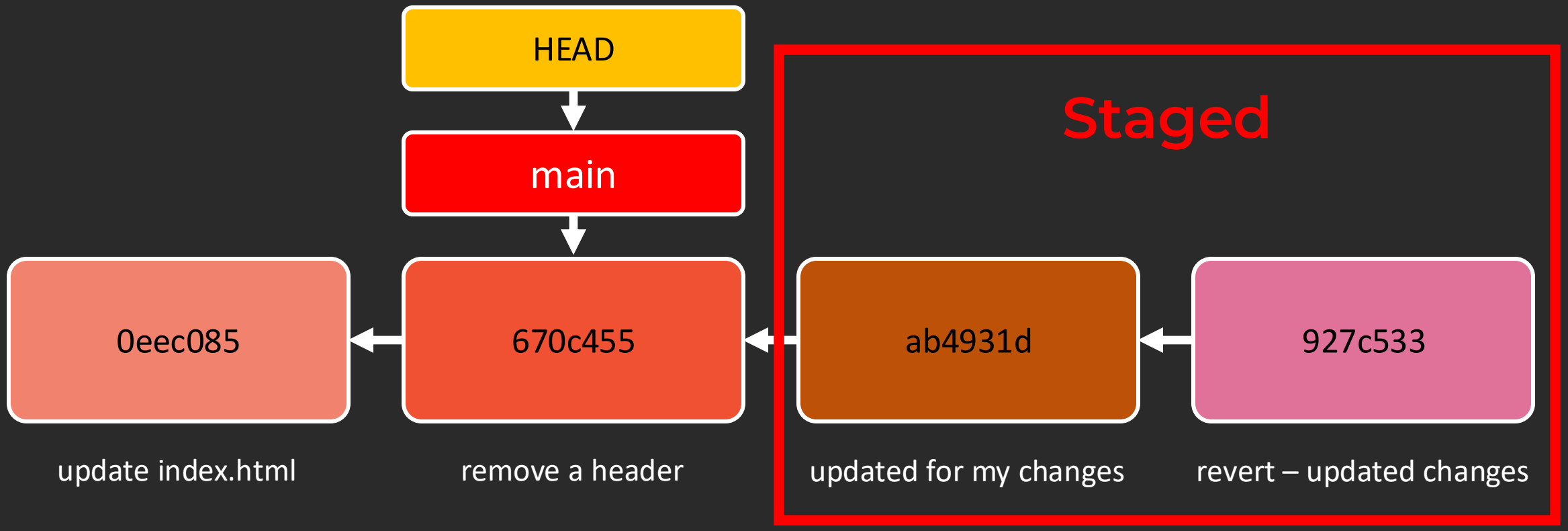


Git Reset

```
git reset --soft <commit>
git reset --mixed <commit>  (default)
git reset --hard <commit>
```

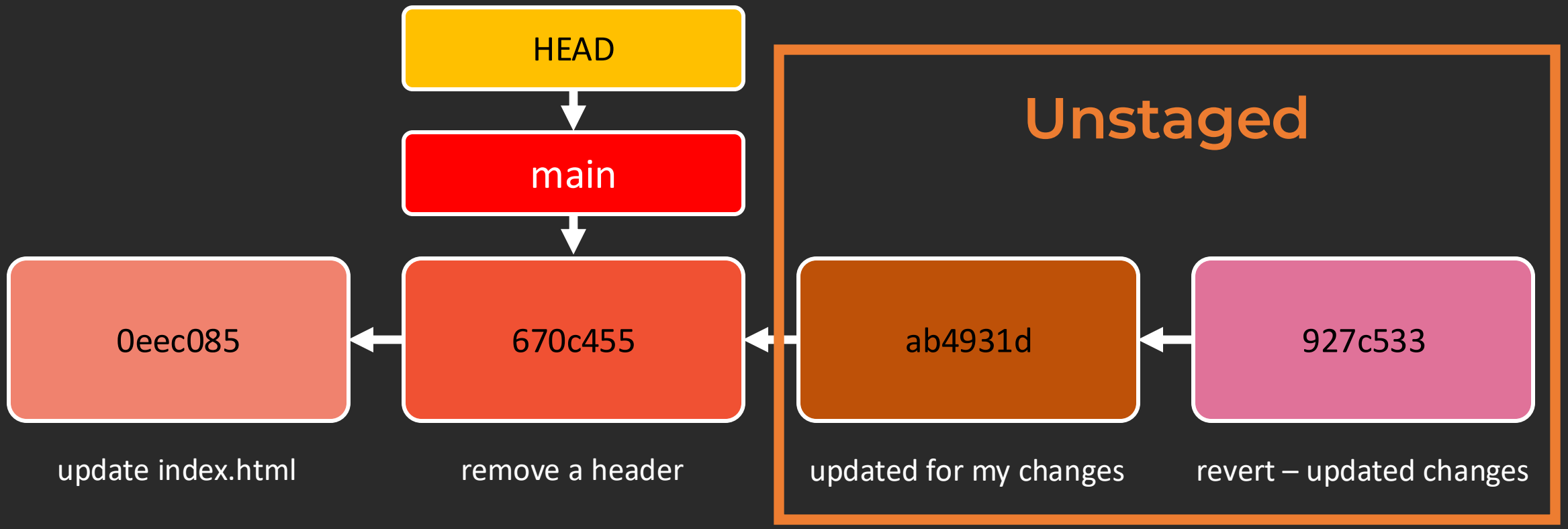
git reset --soft

Undo all changes between HEAD and the commit

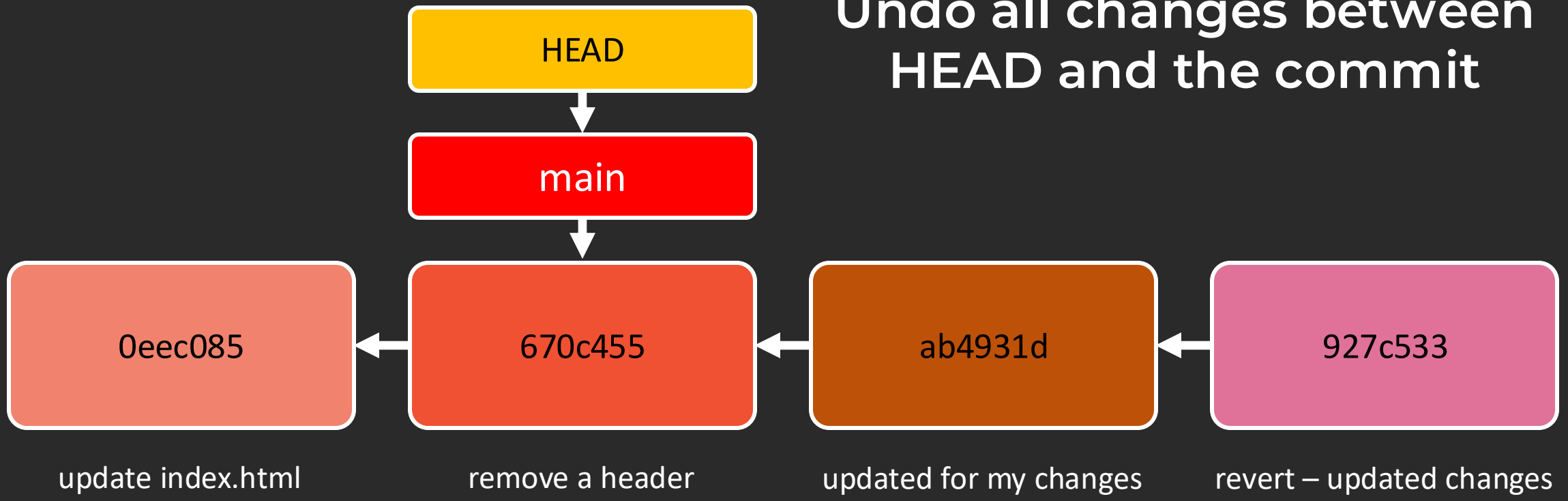


git reset --mixed

Undo all changes between HEAD and the commit

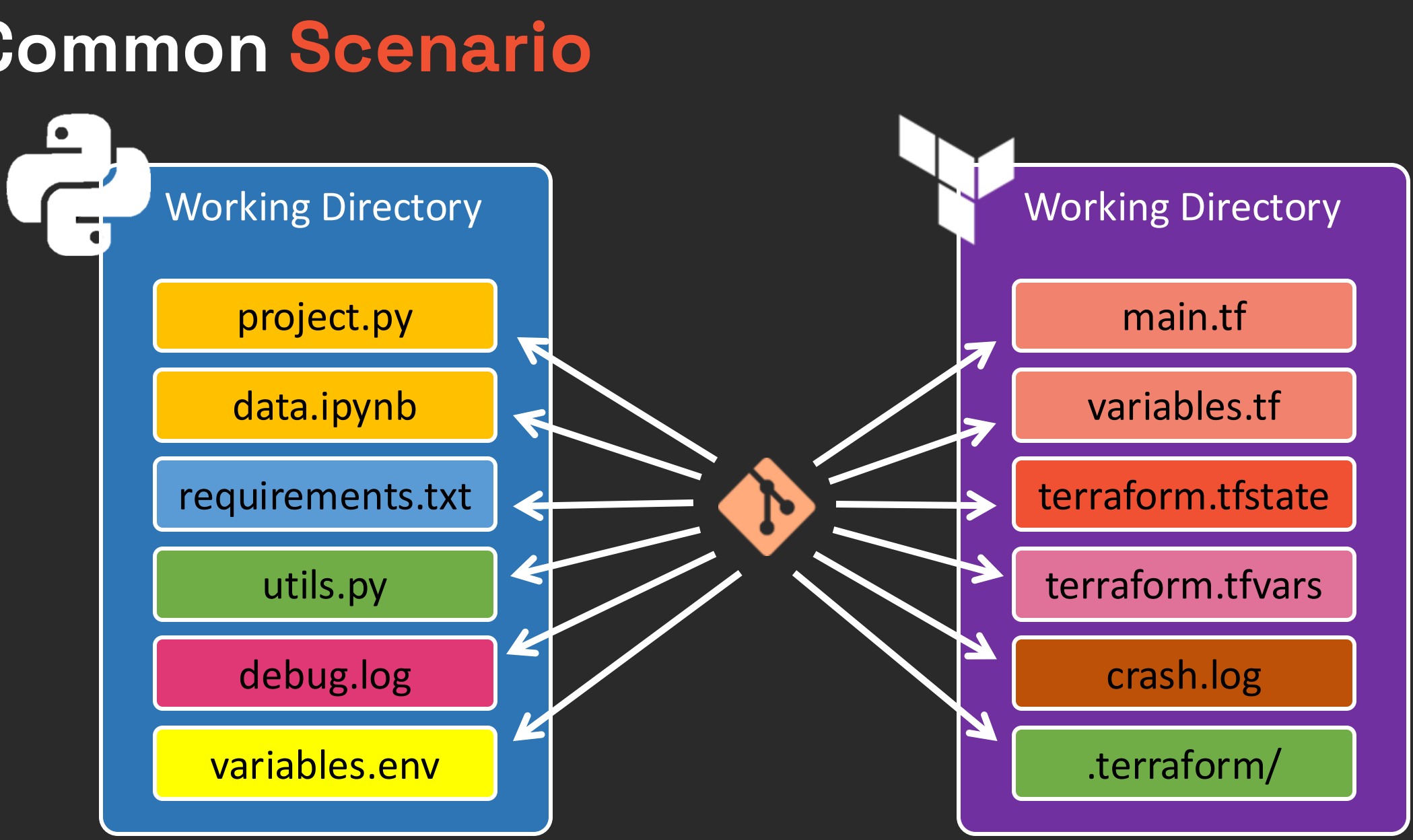


git reset --hard

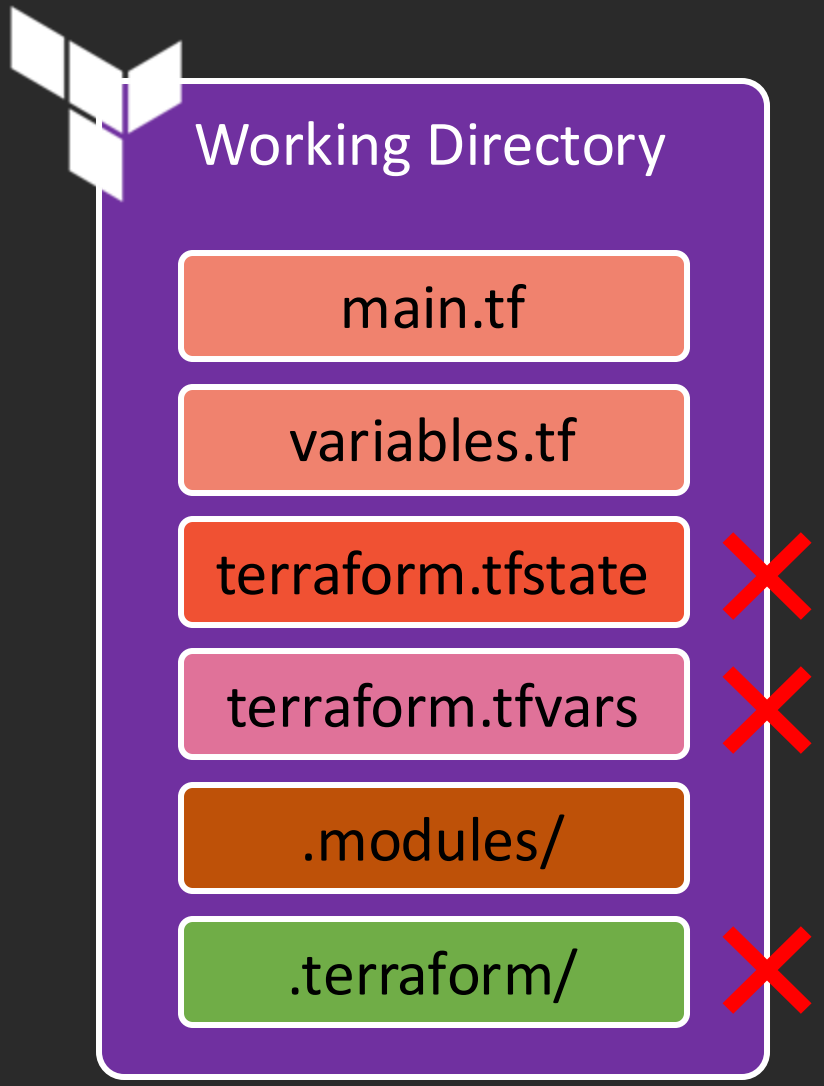


Undo all changes between HEAD and the commit

Common Scenario

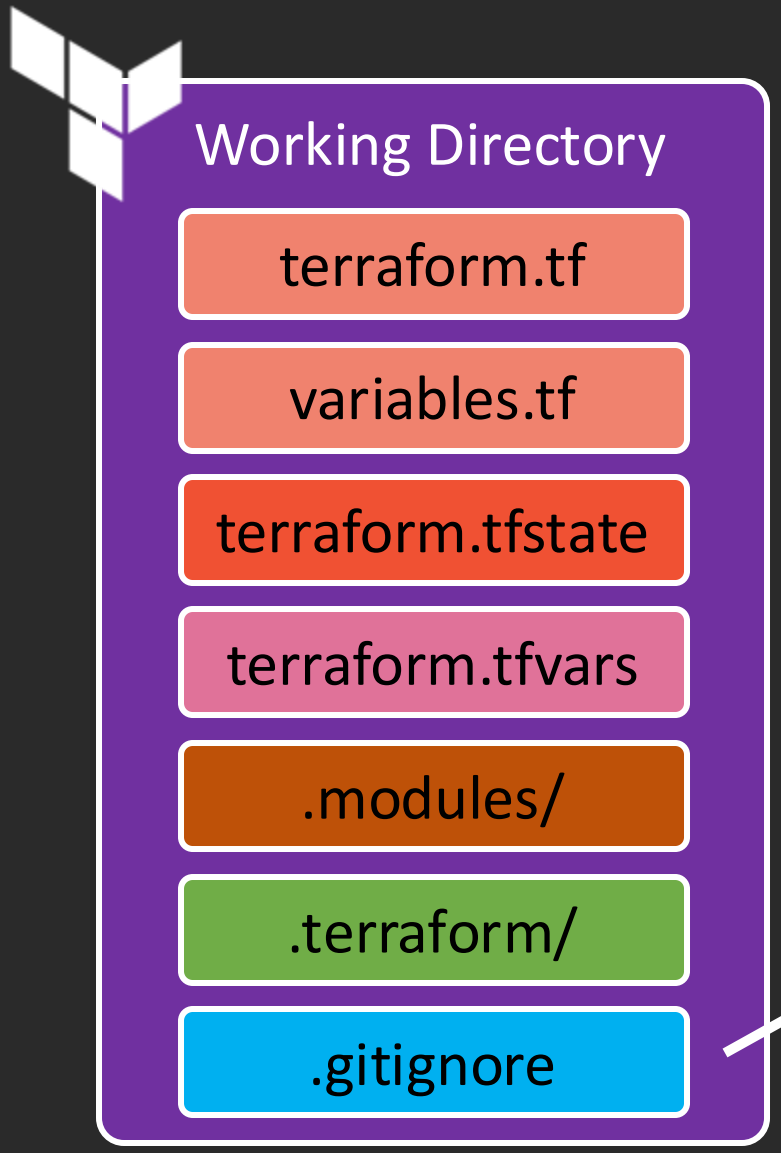


Tracking Files



Remote Repository

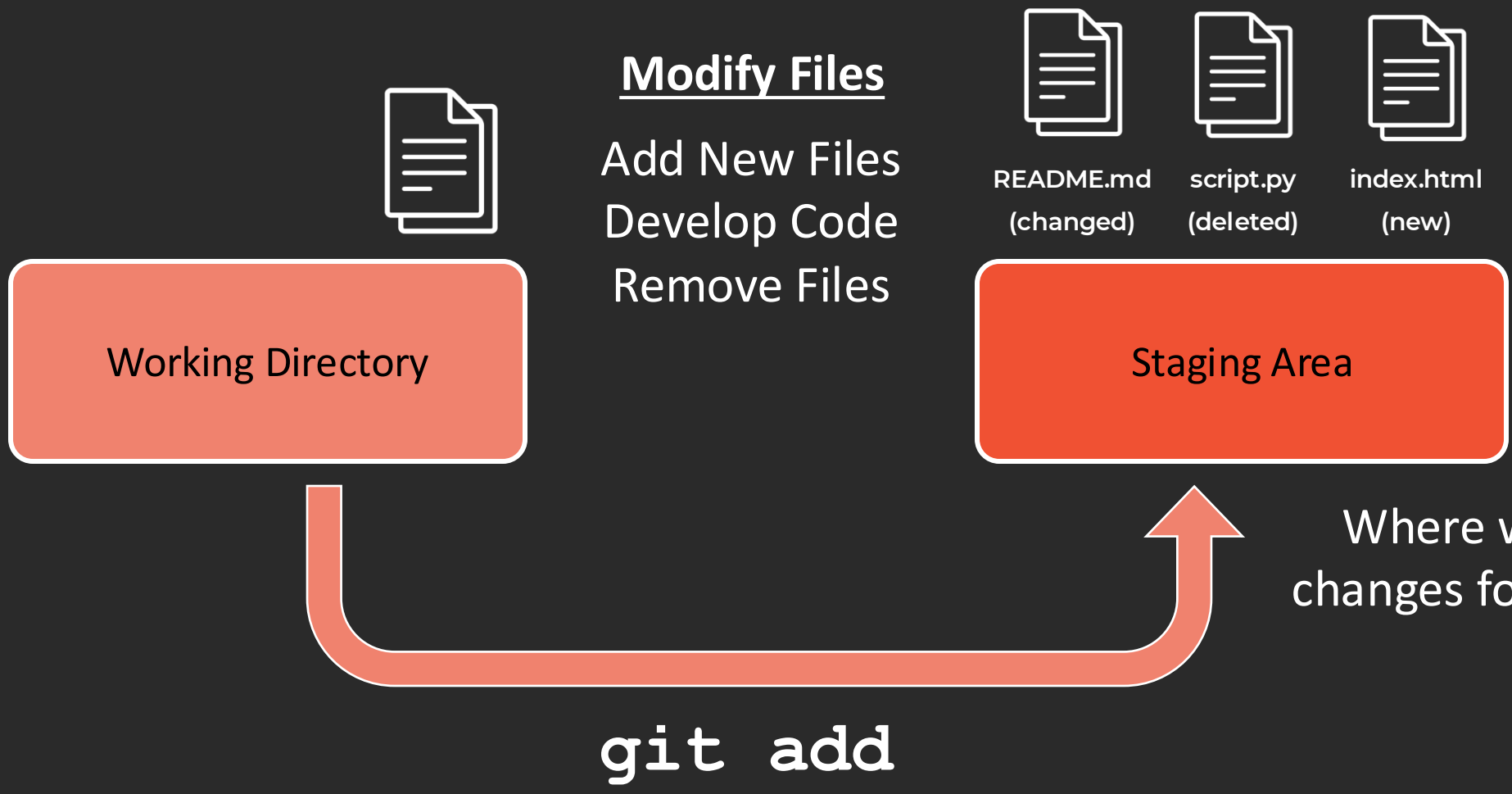
GitIgnore



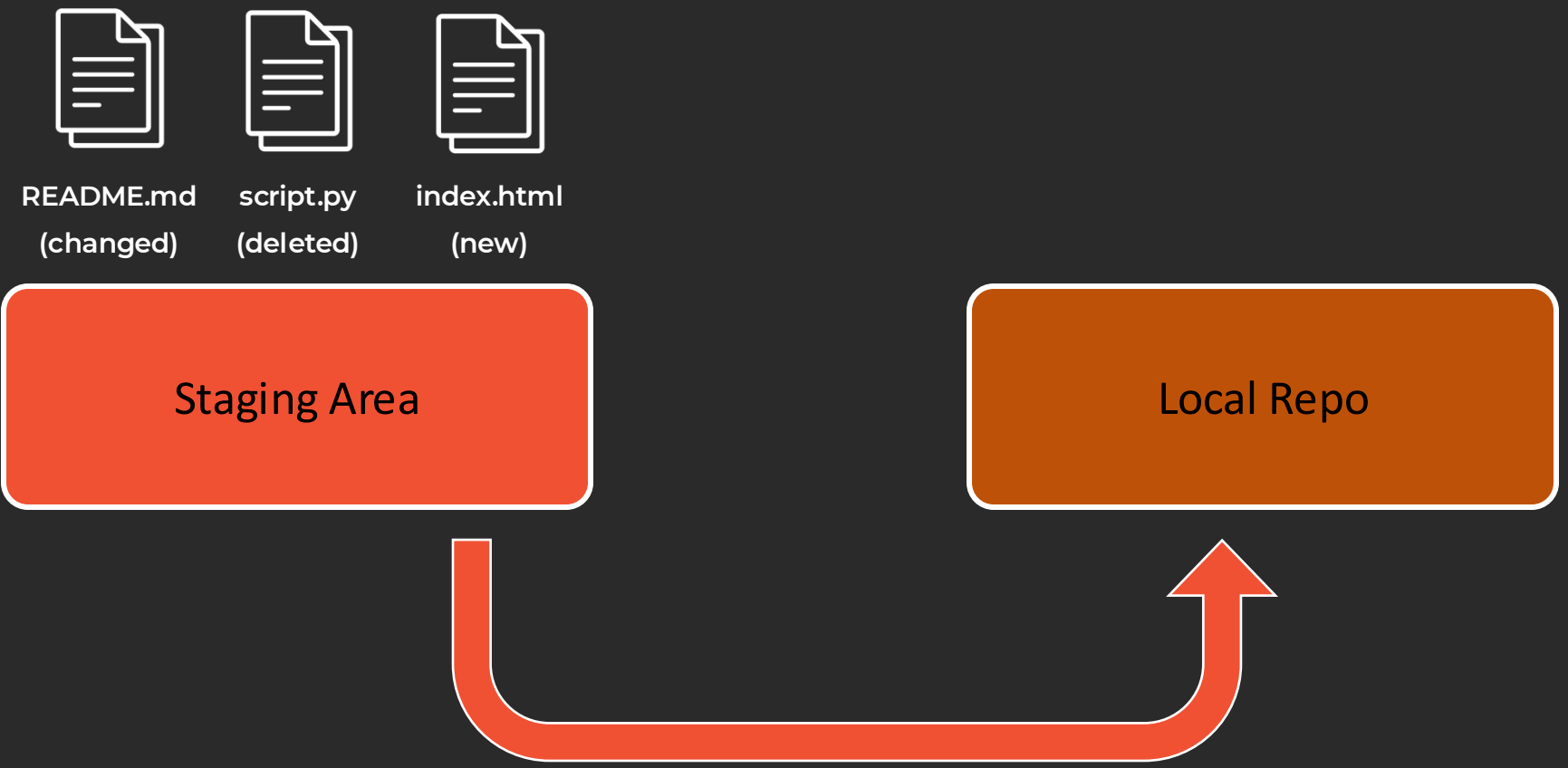
List of files or file types we don't want Git to track

```
.terraform/
*.tfstate
*.tfvars
crash.log
```

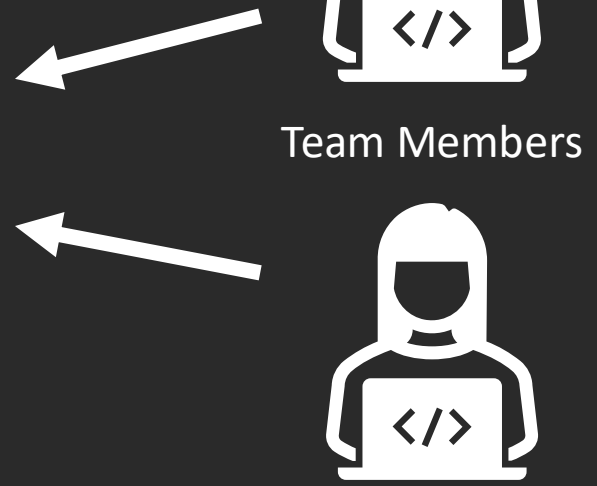
Git Add



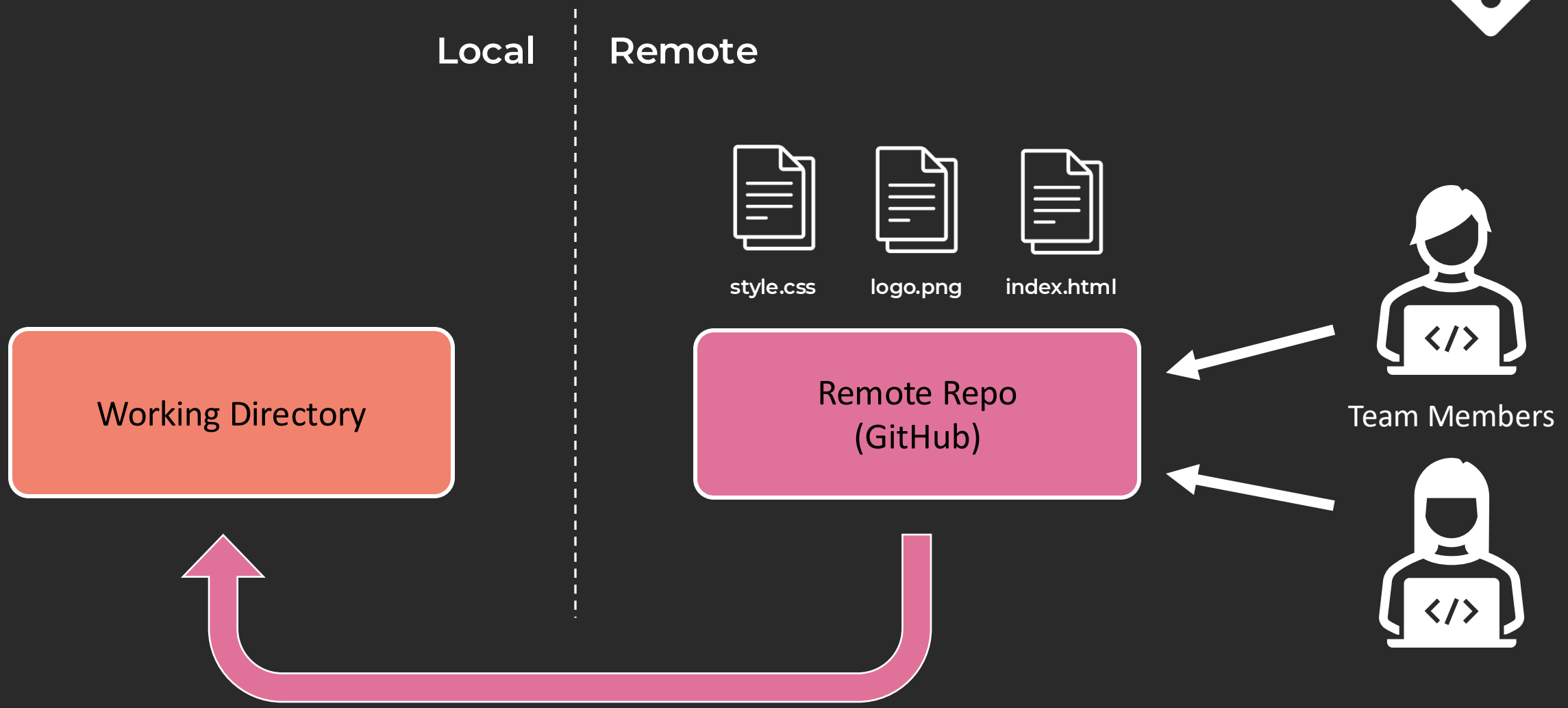
Git Commit



Git Push

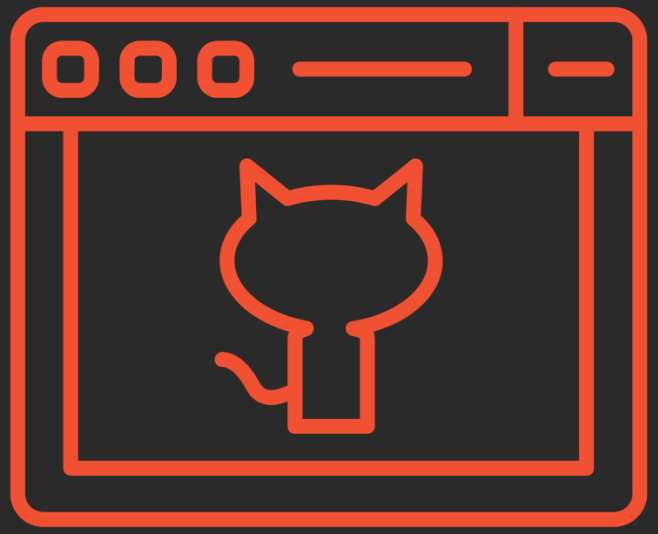


Git Pull



Git Fork

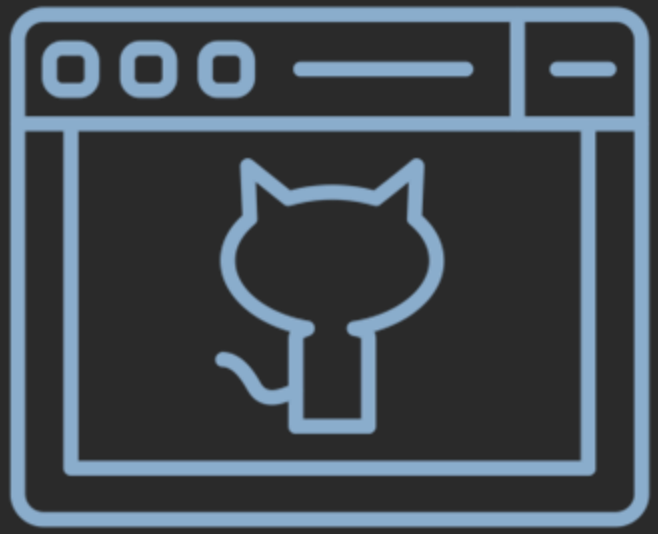
Your Account



Fork

makes a copy

Not Your Account

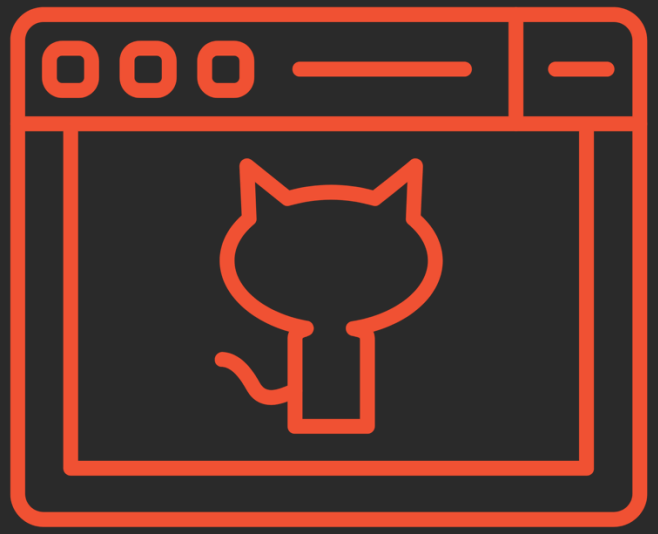


- git-made-easy
- project_a

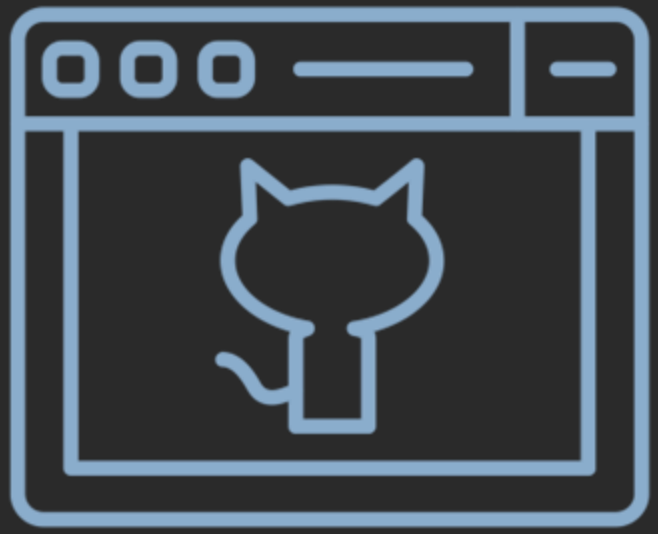
- data_project
- personal_project
- helpful_repo

Git Fork – Fetch Changes

Your Account



Not Your Account



git-made-easy

project_a

helpful_repo

Does NOT sync changes automatically

data_project

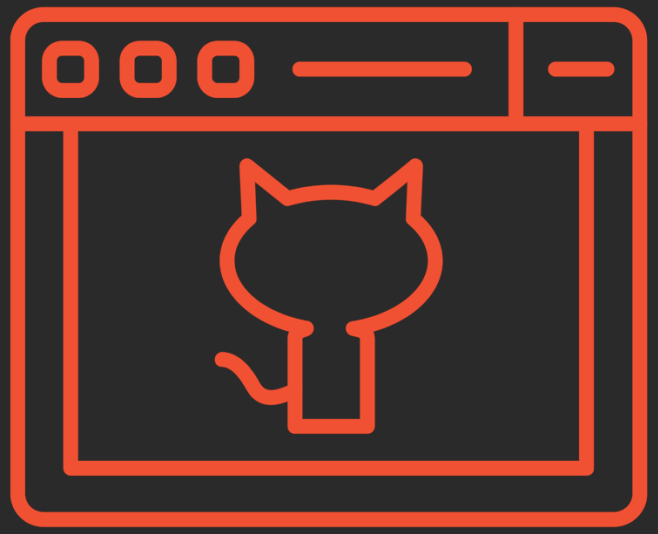
personal_project

helpful_repo

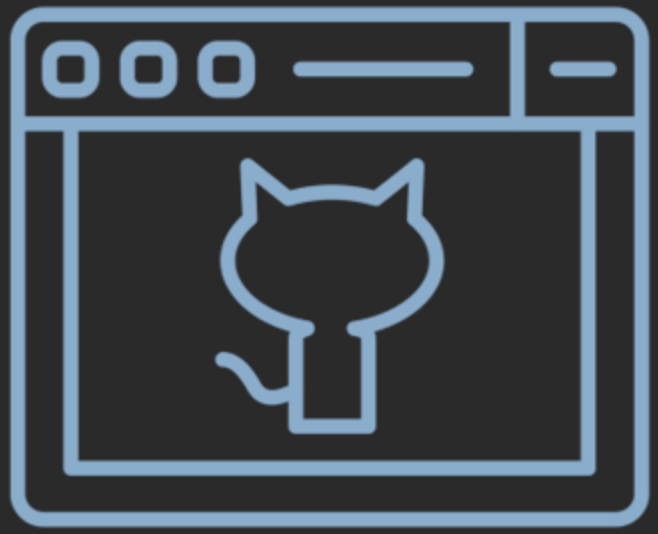
Upstream repo

Git Fork – Contributing Back

Your Account



Not Your Account



git-made-easy

project_a

helpful_repo

Must submit a Pull Request to Contribute

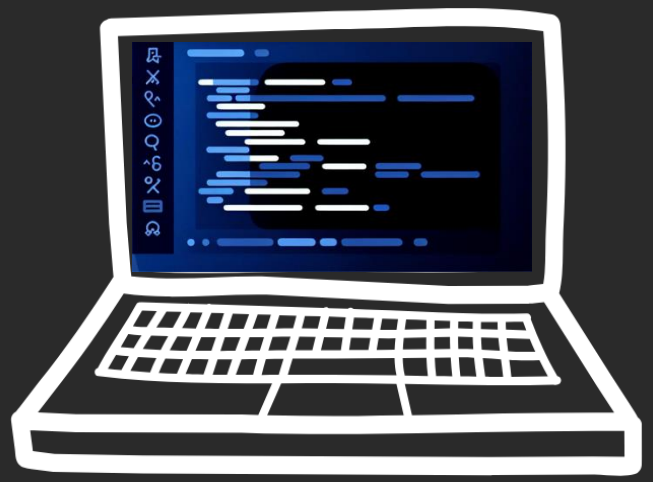
data_project

personal_project

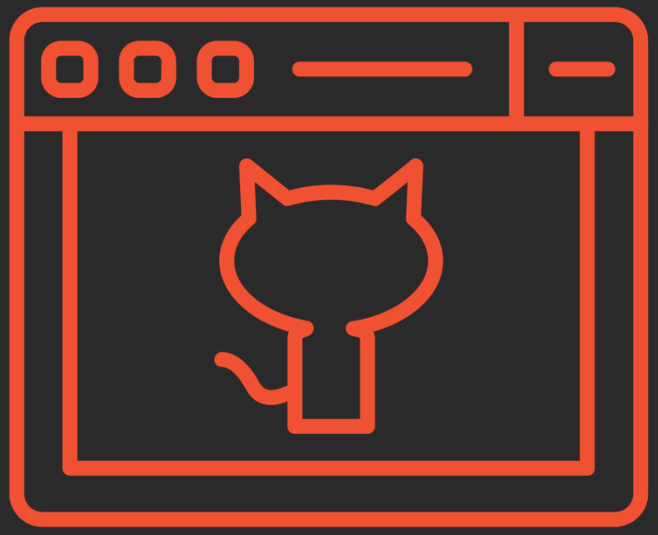
helpful_repo

Upstream repo

Pull Requests



Your Local
Computer/Laptop



git-made-easy