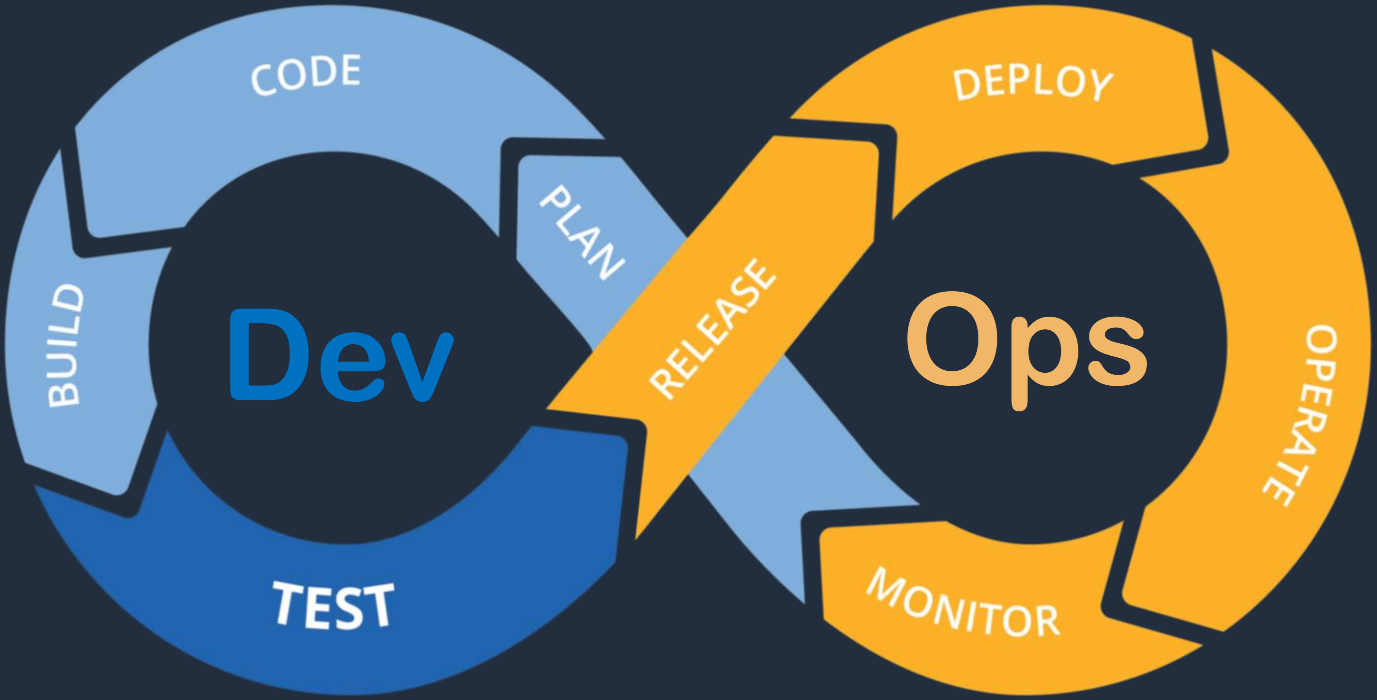
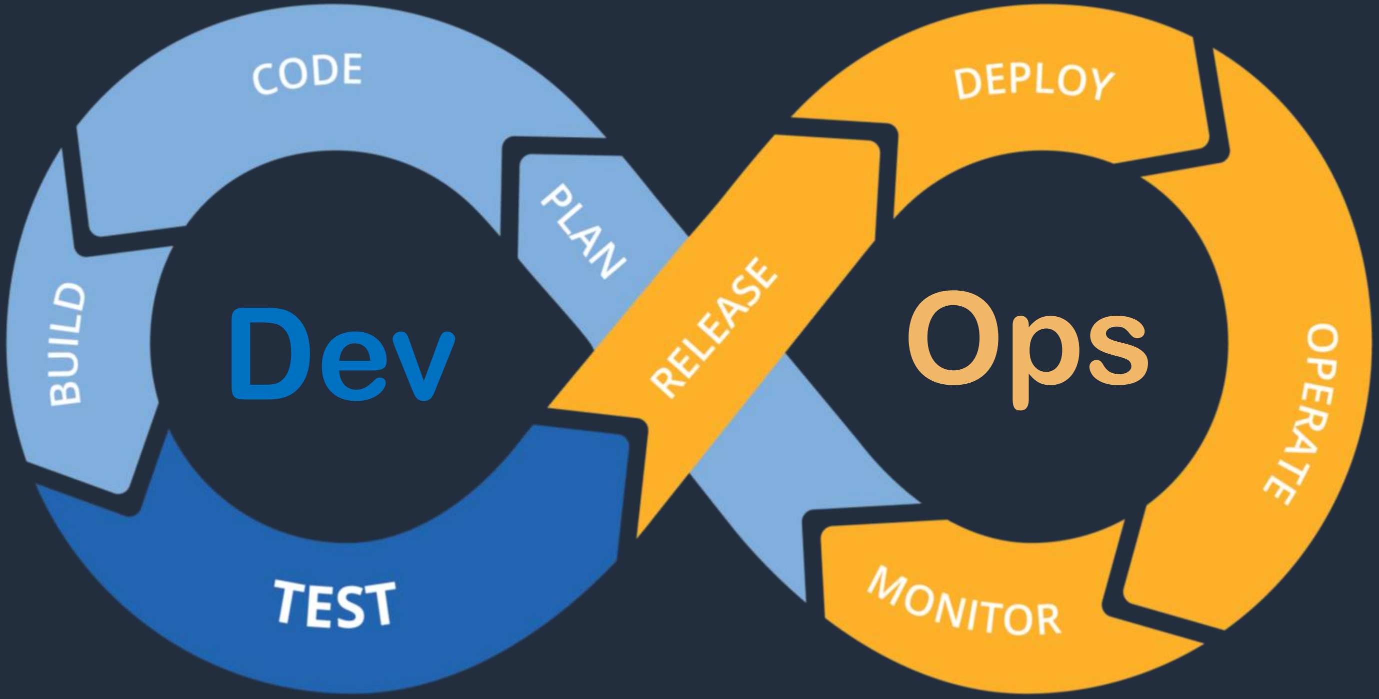

CODE
PLAN
BUILD
Dev
TEST
RELEASE
DEPLOY
Ops
OPERATE
MONITOR

DevOps

git

Version Control System (VCS)

Were to start my DevOps Journey?



Git and GitHub for DevOps Engineers

Git and GitHub for DevOps Engineers

Course Content Overview

- Getting Ready
  - Have a Query?
  - What is version control system
  - Setup git on windows and Linux systems
  - Working with gitbash

- Working with git
- Creating repository on git
  - Working with stages
  - Comparing changes in local repository

Course Content Overview

- Working with GitHub
  - Why GitHub and Creating a GitHub account
  - Creating a repository on GitHub
  - Cloning and push changes onto repository
- Working with Other Developer
  - How to work on other developers' code
  - Unable to push code onto remote repo?
  - Enabling ssh based authentication between Git and GitHub
  - How a Developer write code and push it onto remote repository

Course Content Overview

- Git Commits
  - How commit works
  - Knowing about specific commit
  - Committing changes from GitHub GUI

- Git Branches
  - Why do we need branches
  - Branching strategies & how branches works in DevOps workflow
  - Committing changes on branches
  - Resolving merge conflicts

Course Content Overview

- Working with Team
  - How Fork works
  - What is Pull Request and how it works
  - Working with private repositories
  - Protected branches and contributors
- Reverting changes
  - Revert changes from working directory
  - Reverting changes from Staging and Local Repository
  - .gitignore file

Course Content Overview

- Working with a Project
  - Project Overview
  - How to setup repositories for a new project
  - Allowing Developers to check-in code
  - Enabling DevOps workflow on Dev branch
  - Merging changes from Dev branch to UAT branch
  - Releasing code onto production

Course Objectives

- Git and GitHub
- DevOps Engineers activates on Git and GitHub
- How DevOps Engineers work with Developers
- How Git and GitHub works in the DevOps
- Real-time Project
- Bonus lecture

Before Starting

Prerequisites:

Linux Basics

AWS EC2 Service

GitHub URL : https://github.com/ravdy/git_course

Resources:

AWS Account

GitHub Account

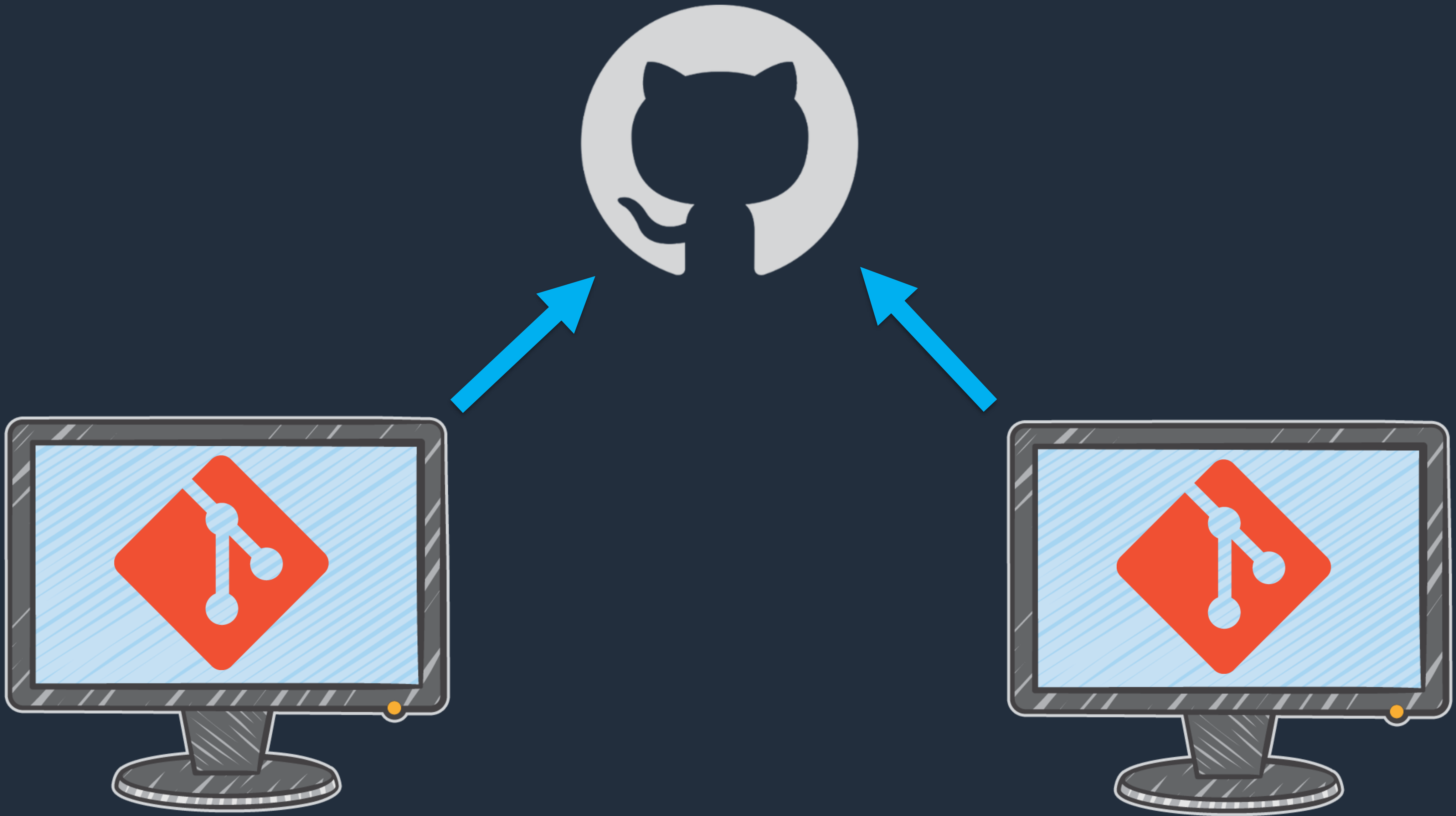
About Trainer

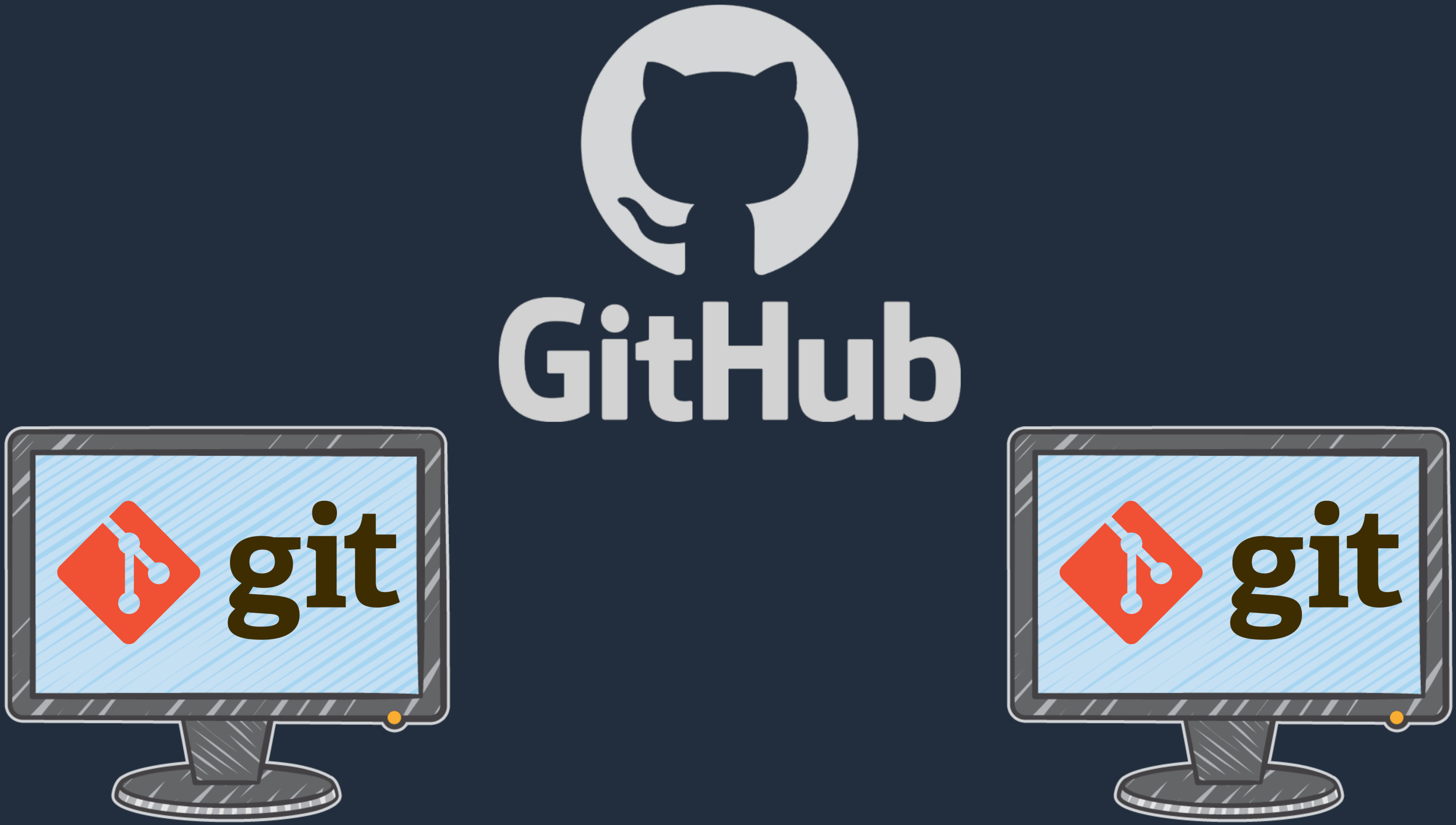
- AR Shankar
- Total 10+ Years of IT Experience
- Working as a DevOps Consultant in a leading IT Company
- Experience on DevOps, AWS and Linux
- Teaching more then **30,000** students on Udemy
- A YouTube Content Creator

Have a Query?

- Still reach out to me directly
  - Facebook group - https://www.facebook.com/groups/valaxytechnologies
  - LinkedIn group - https://www.linkedin.com/groups/13911419/
  - Instagram - https://www.instagram.com/valaxytechnologies
  - Facebook Page - https://www.facebook.com/ValaxyTechnologies/
  - LinkedIn Page - https://www.linkedin.com/in/valaxytechnologies/

Email - valaxytech@gmail.com




GitHub
git
git

The Problem

You want to write some <code/>
with a group of people

Solution: Version Control

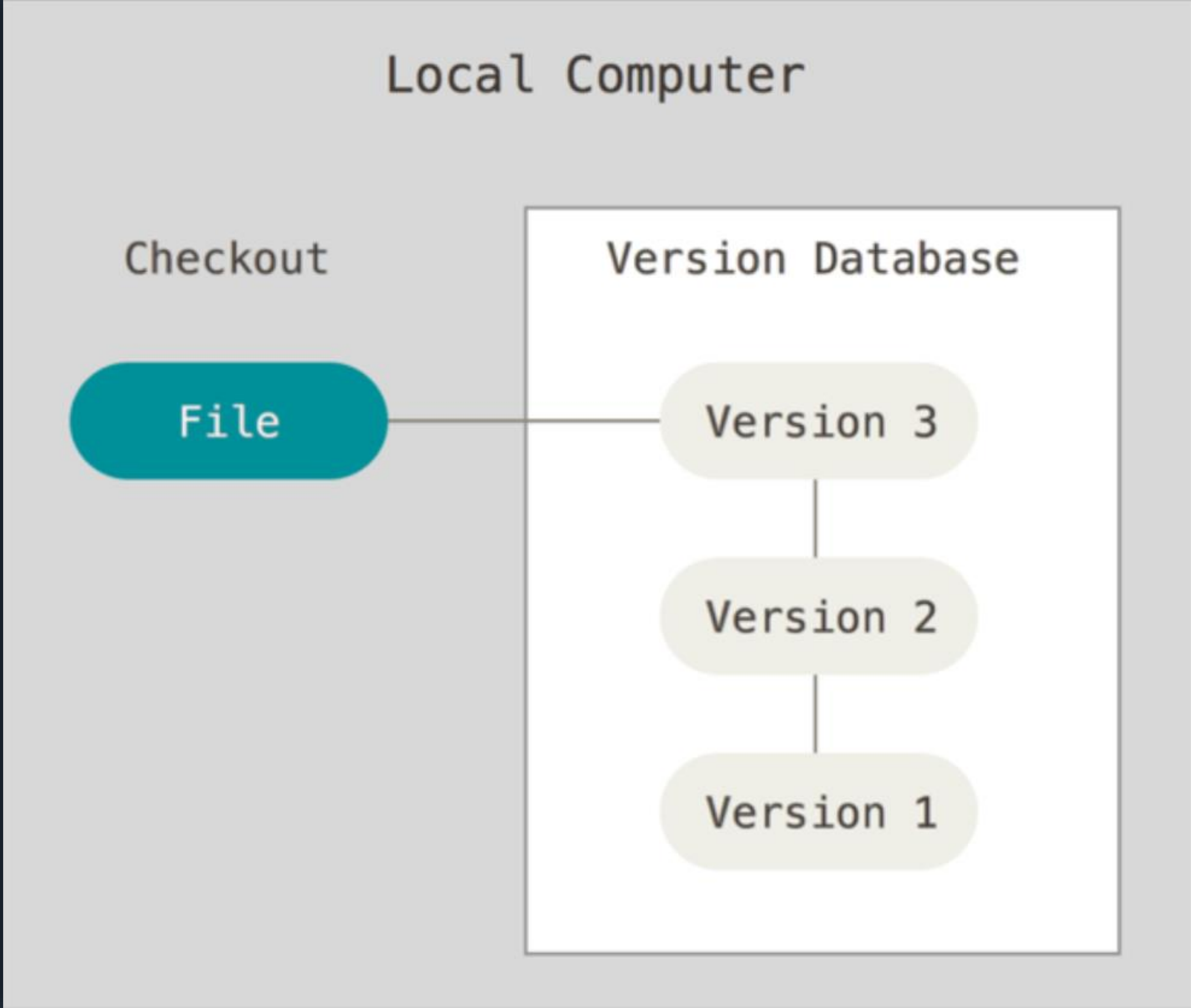
What is Version Control?

- A system that keeps **records of your changes**
- Allows for **collaborative development**
- Allows you to know **who made what changes** & **when**
- Allows you to **revert any changes**

Version Control System types

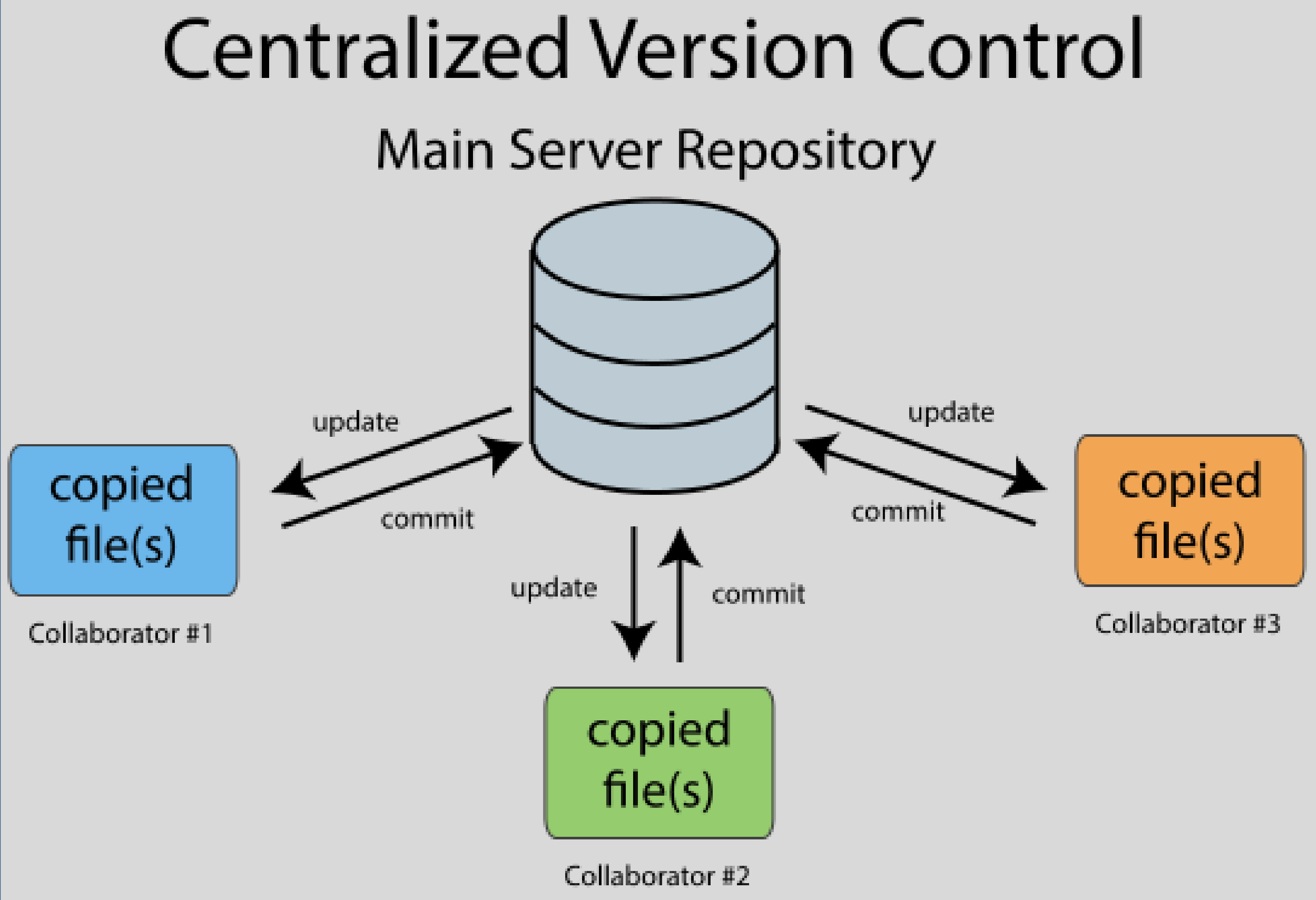
- Local
- Centralized
- Distributed

Local Version Control System



write some <code/>
with people in your team?

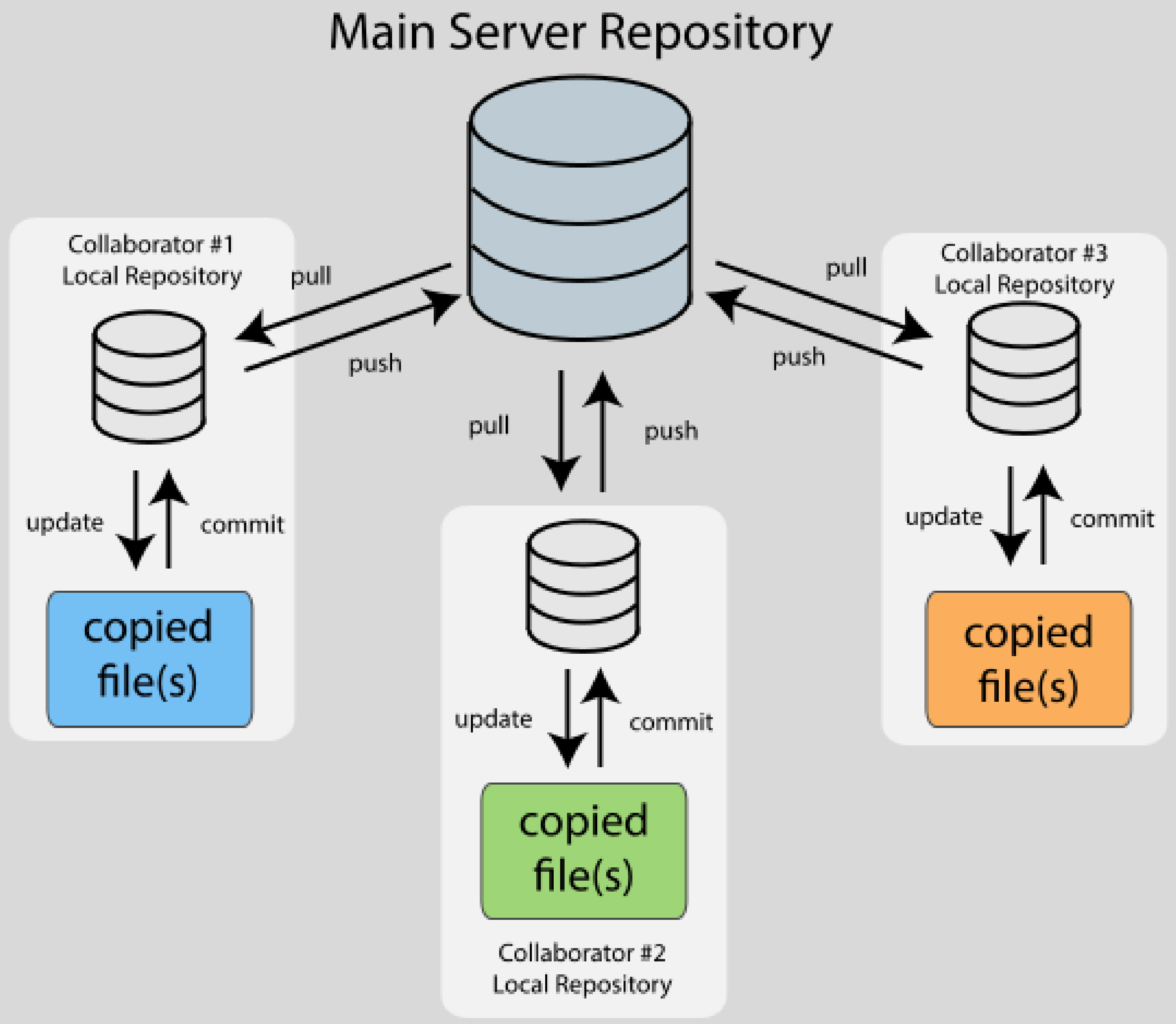
Centralized Version Control System



write some <code/>
with a distributed team?

Distributed Version Control System

Distributed Version Control



git

SUBVERSION

mercurial

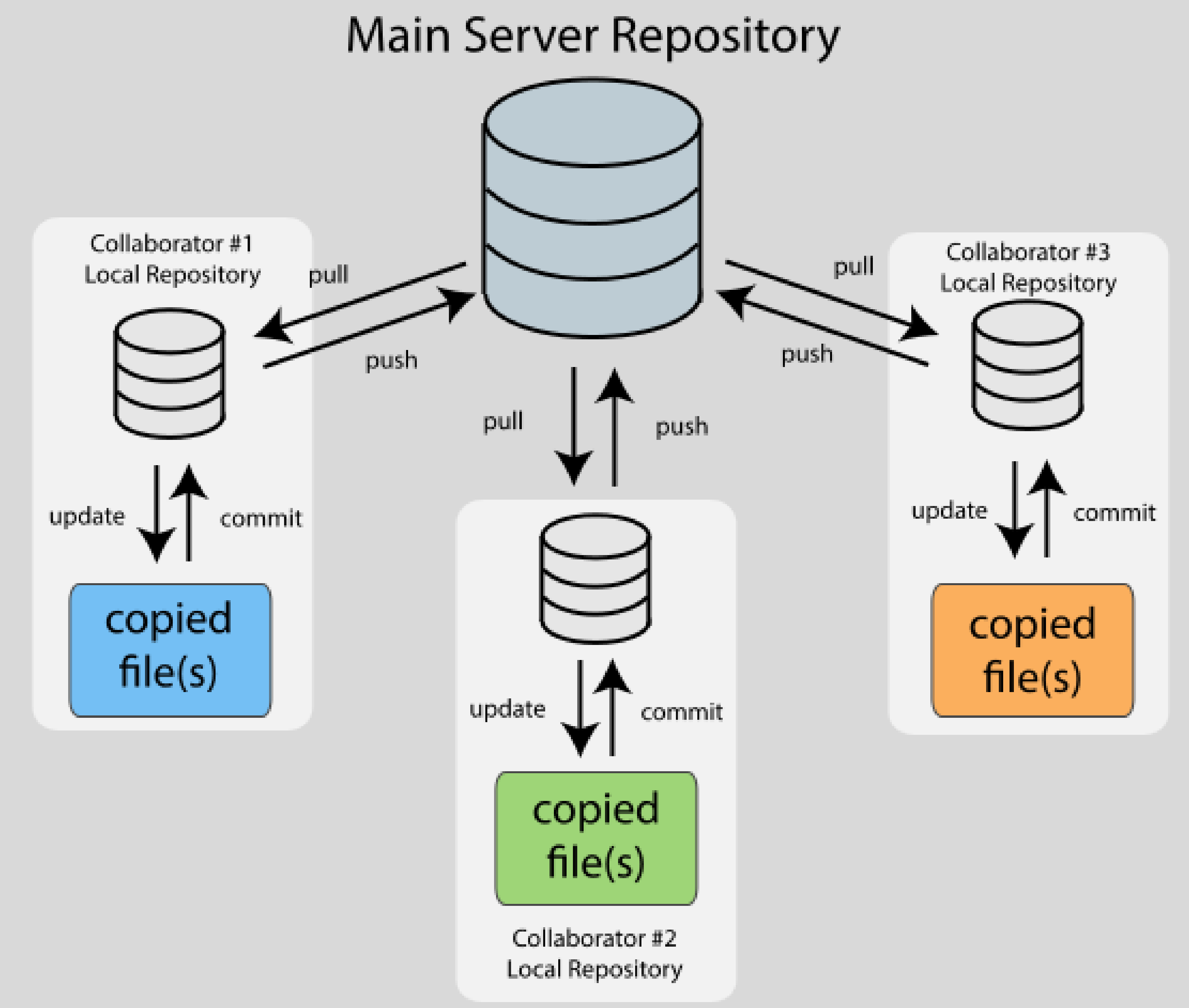
PERFORCE

Version everything.

git

Distributed Version Control System

Distributed Version Control



Get <code/>

From a Developer

Git Environment

- Developer 1: on Windows System
- Developer 2: on AWS Linux EC2 Instance

Install

On Windows System

Install

On Linux System

Introduce yourself to Git

- Start with below commands

```
git config –-global user.name “Shankar”

git config --global user.email “arsr319@gmail.com”
```

This is one time activity

- To display user details

```
git config –-global user.name
git config --global user.email
```

Before working with Git

- All git commands start with “git”
- You should introduce yourself to git
- Git works with repositories
- Most of the git commands should run inside a git repository

Creating Repository

A *repository,* encompasses the entire collection of files and folders associated with a project, along with each file’s revision history.

Create a repository on Git

```
git init .
```

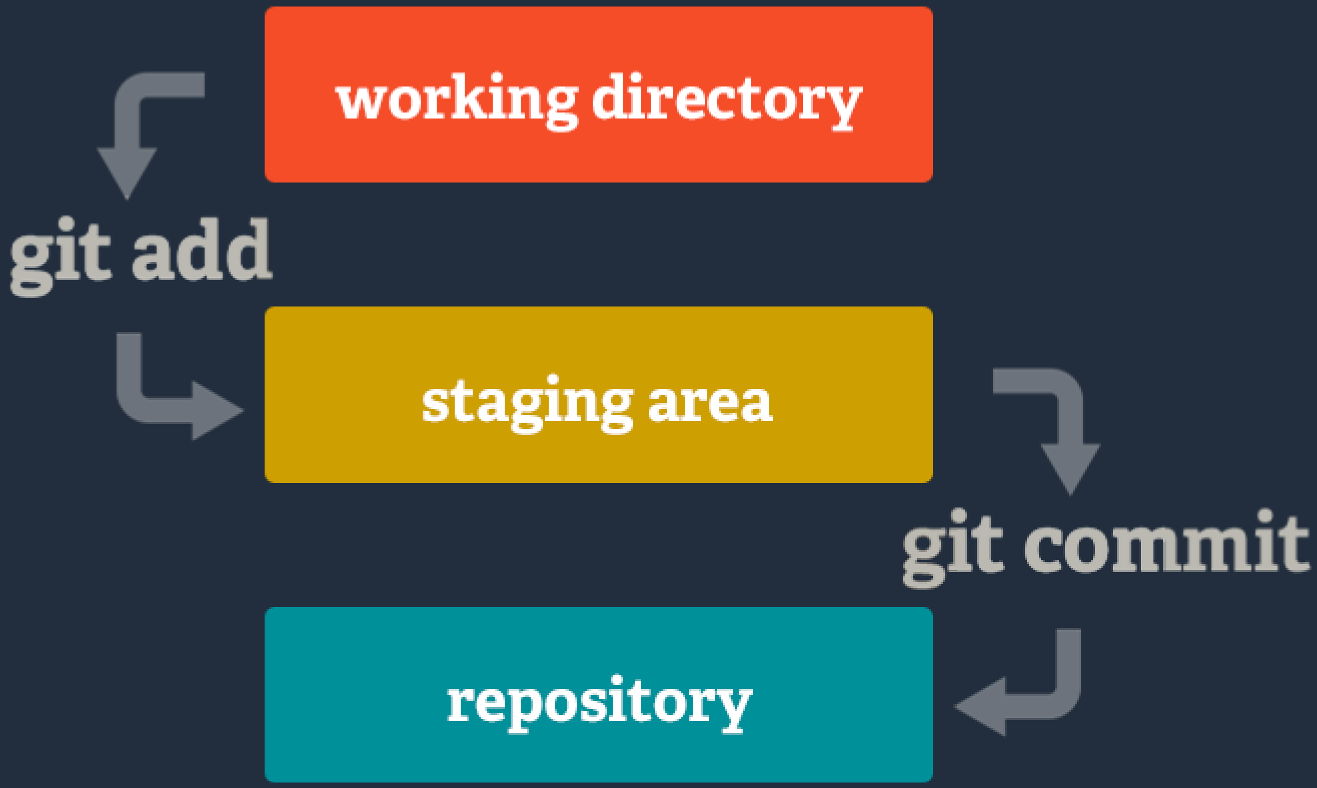
to create a repo without name

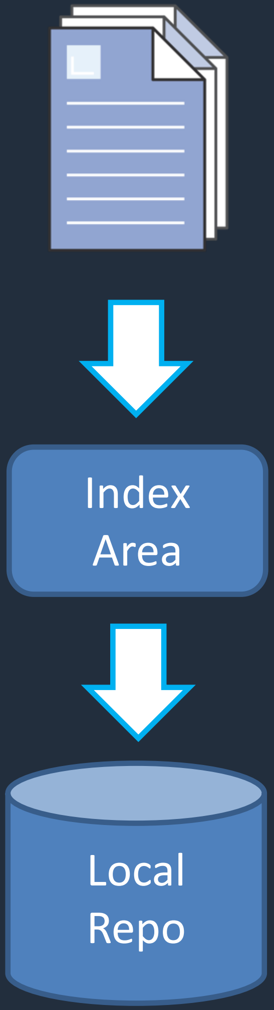
Add code to a Repository

Git stages

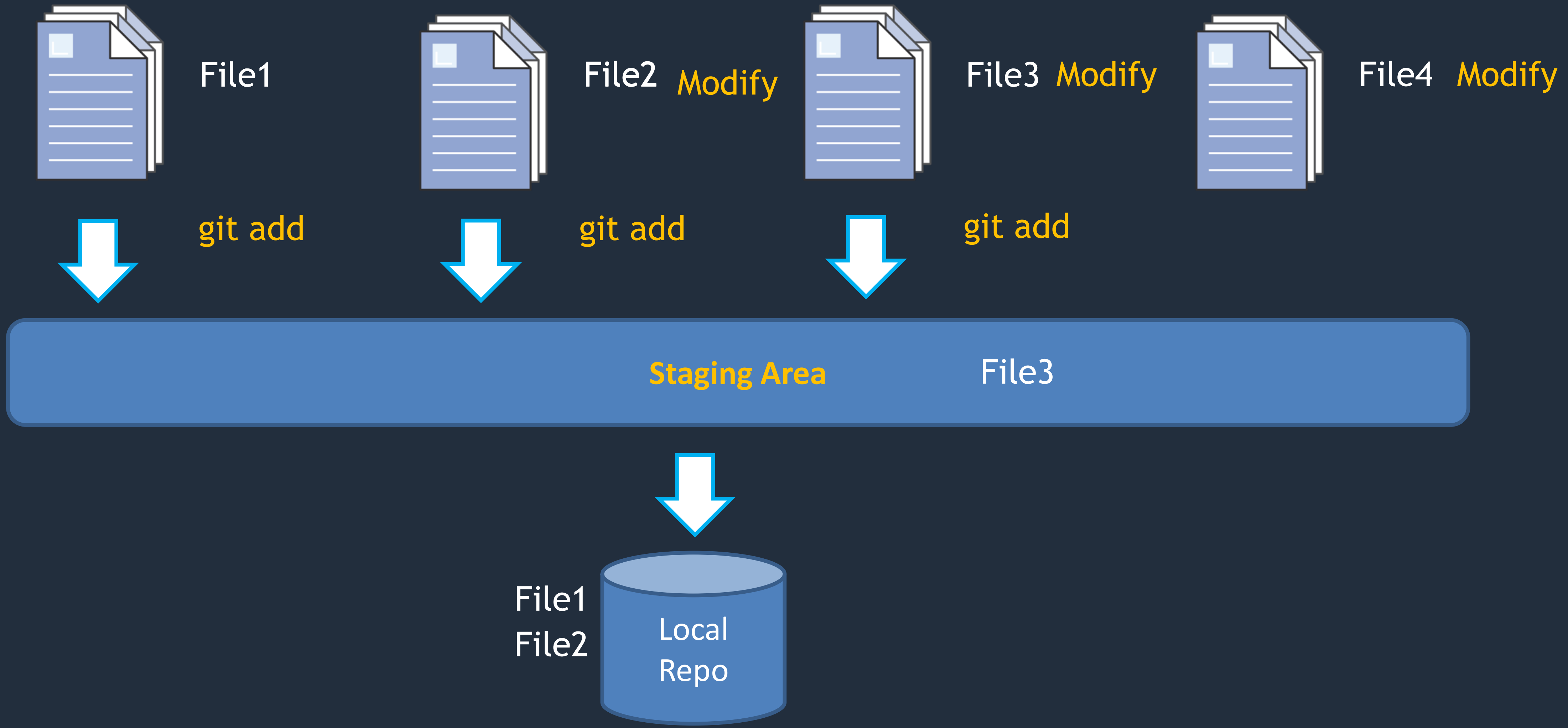
- git add - add files into staging area
- git commit - add changes to local VCS
- git status - shows current stage of repository
- git log - shows commit history

Git Stages

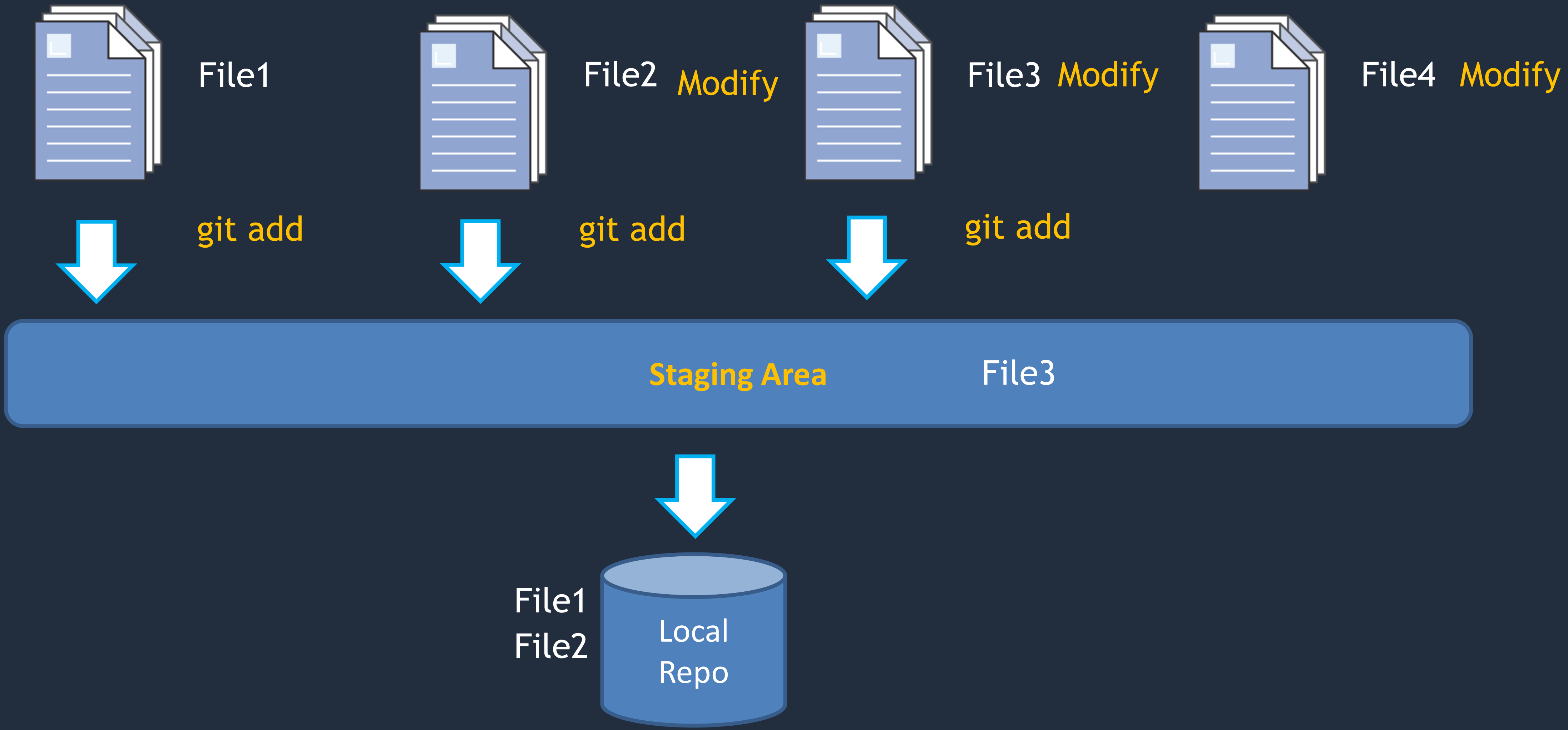

working directory
git add
staging area
git commit
repository


Index Area
Local Repo

Git Stages


File1
File2
Modify
File3
Modify
File4
Modify
git add
git add
git add
Staging Area
File3
File1
File2
Local Repo

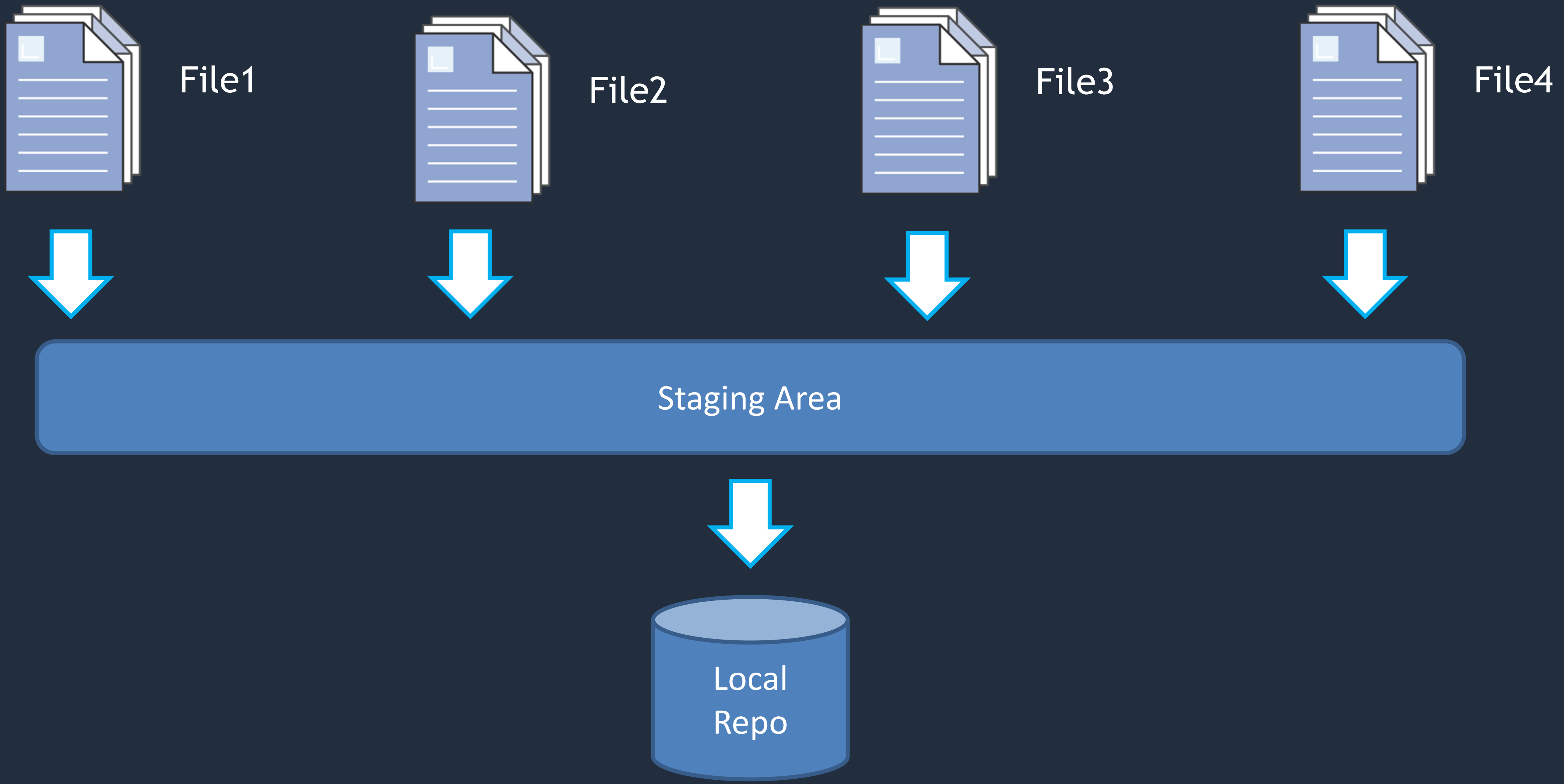
Git Stages


File1
File2
Modify
File3
Modify
File4
Modify
git add
git add
git add
Staging Area
File3
File1
File2
Local Repo

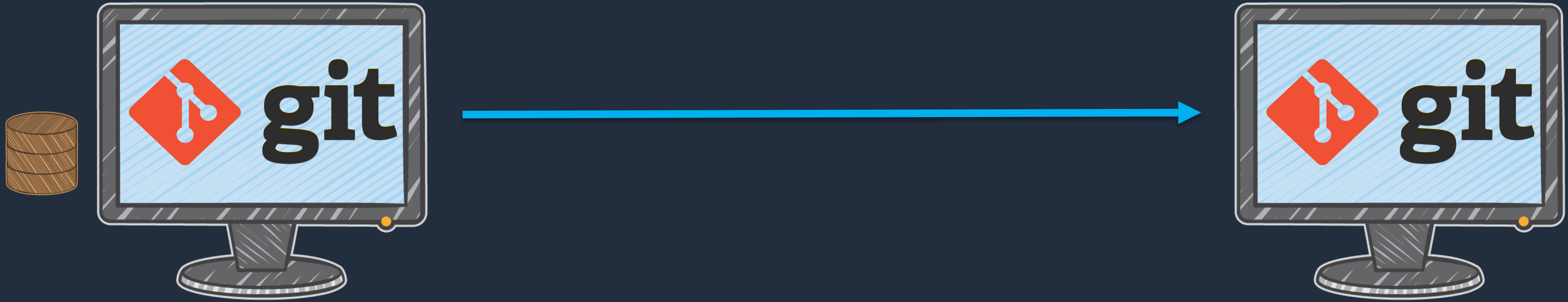
Git diff commands

- git diff – Compare changes of working directory with Staging area
- git diff --staged – Compare changes of Staging area with Local Repository
- git diff HEAD – Compare changes of working directory with Local Repository

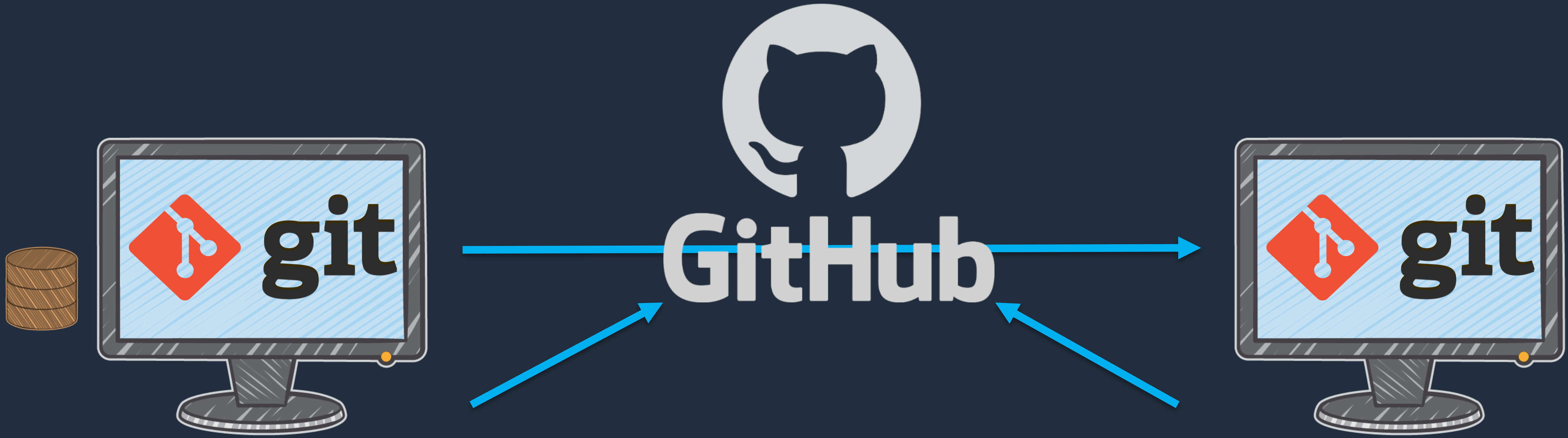
Git Stages



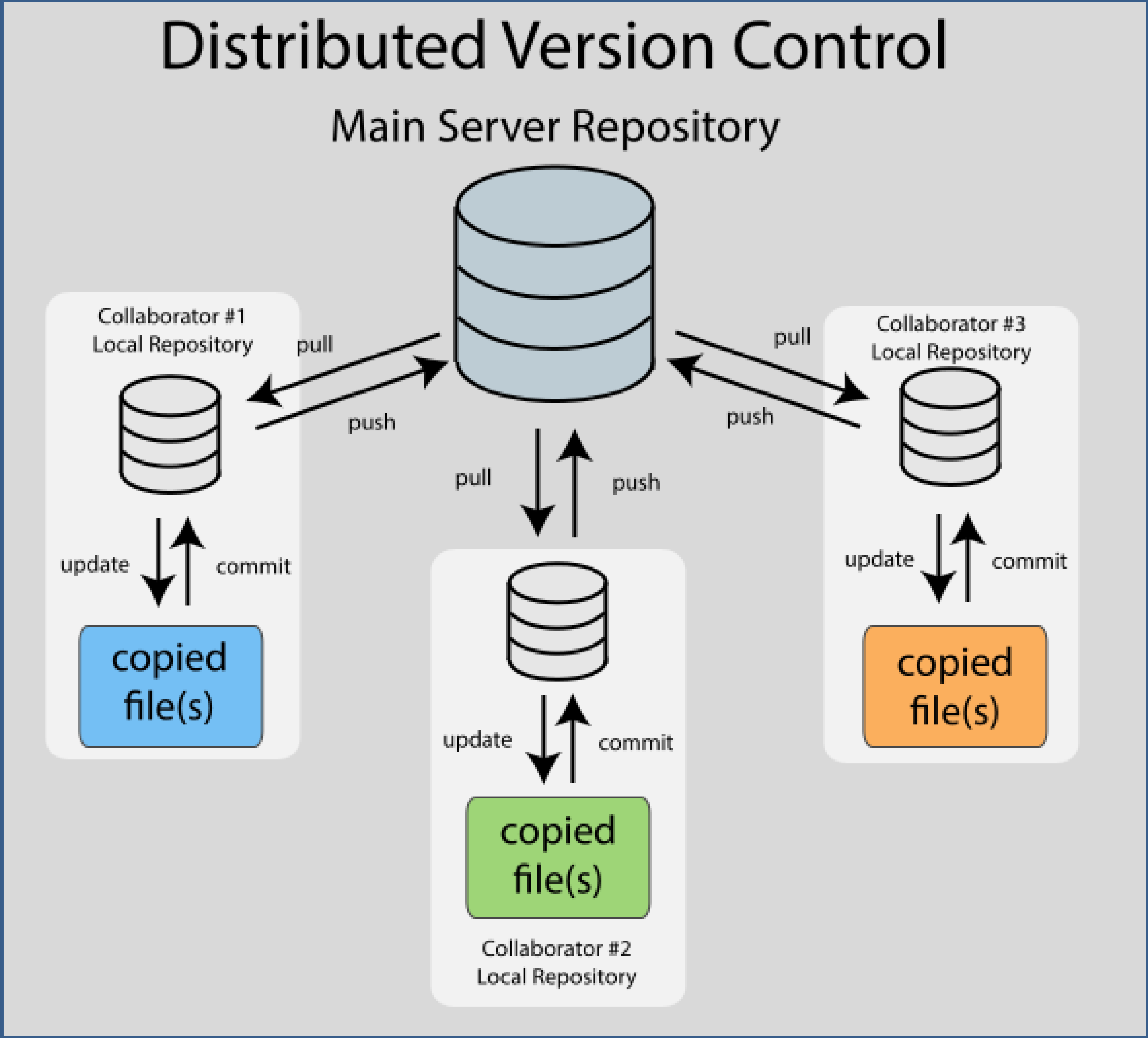
Working with Team



Working with Team

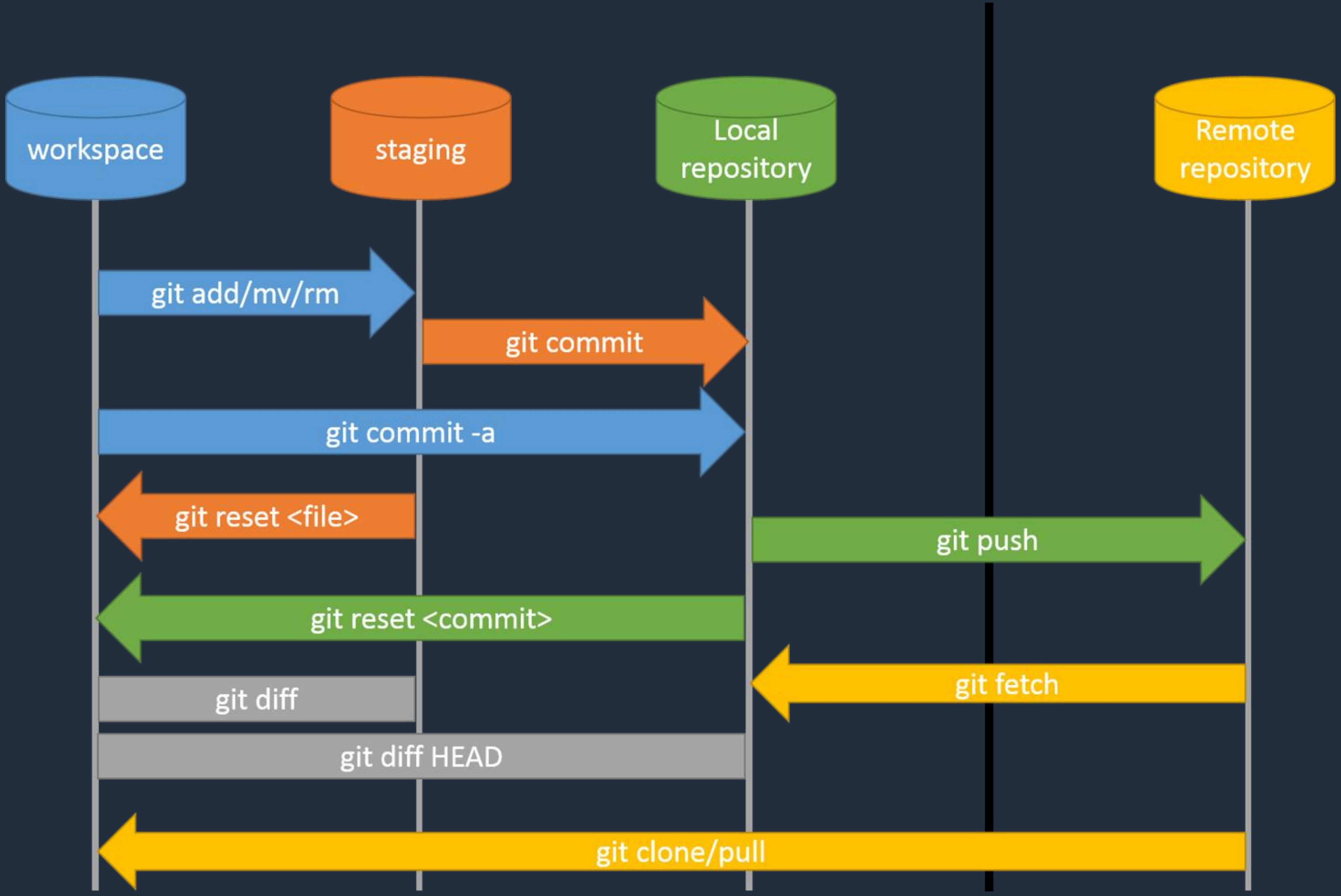


Distributed Version Control System



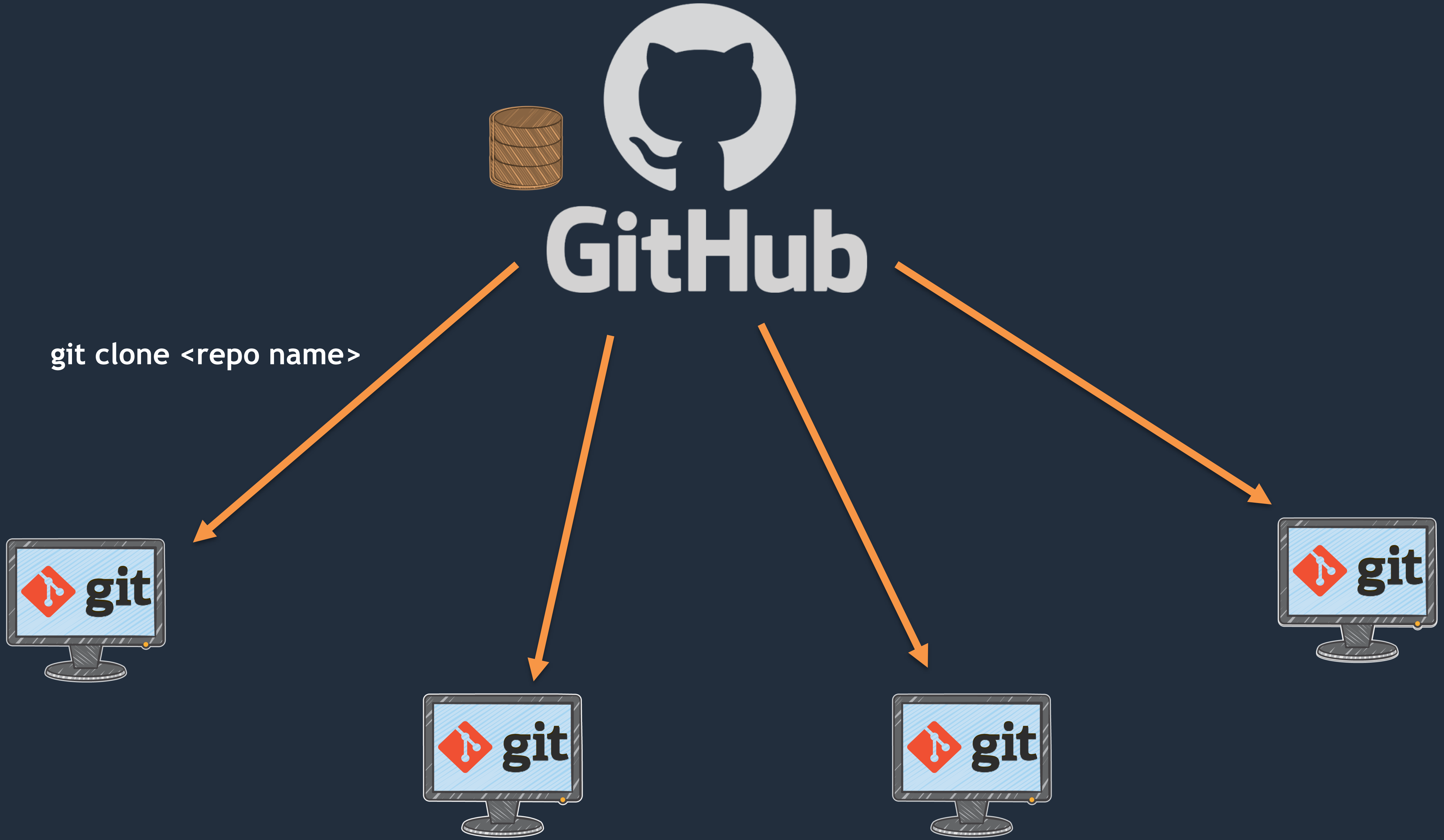
GitHub

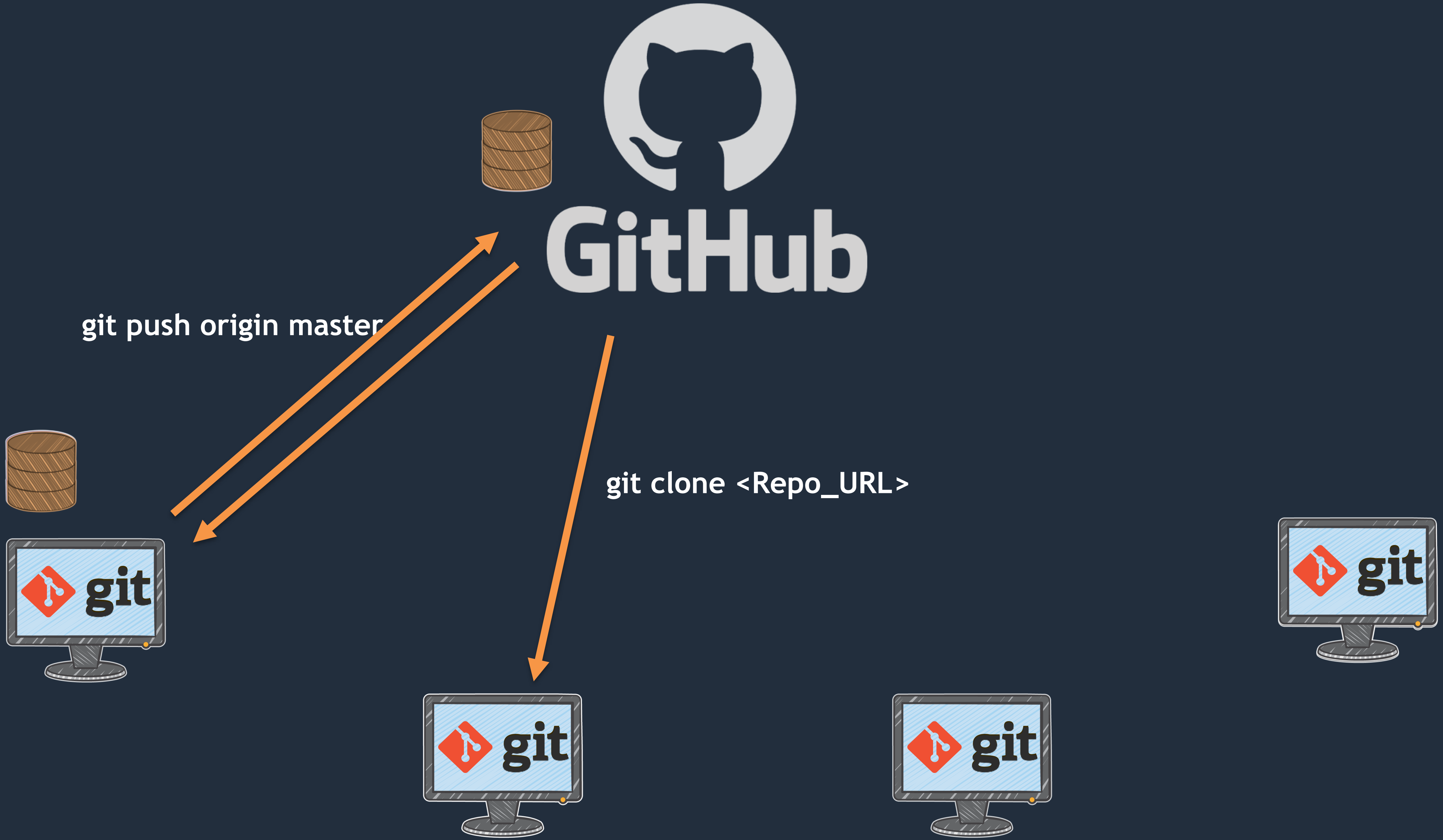
Cloning a Repo from GitHub


workspace
staging
Local repository
Remote repository
git add/mv/rm
git commit
git commit -a
git reset <file>
git push
git reset <commit>
git fetch
git diff
git diff HEAD
git clone/pull

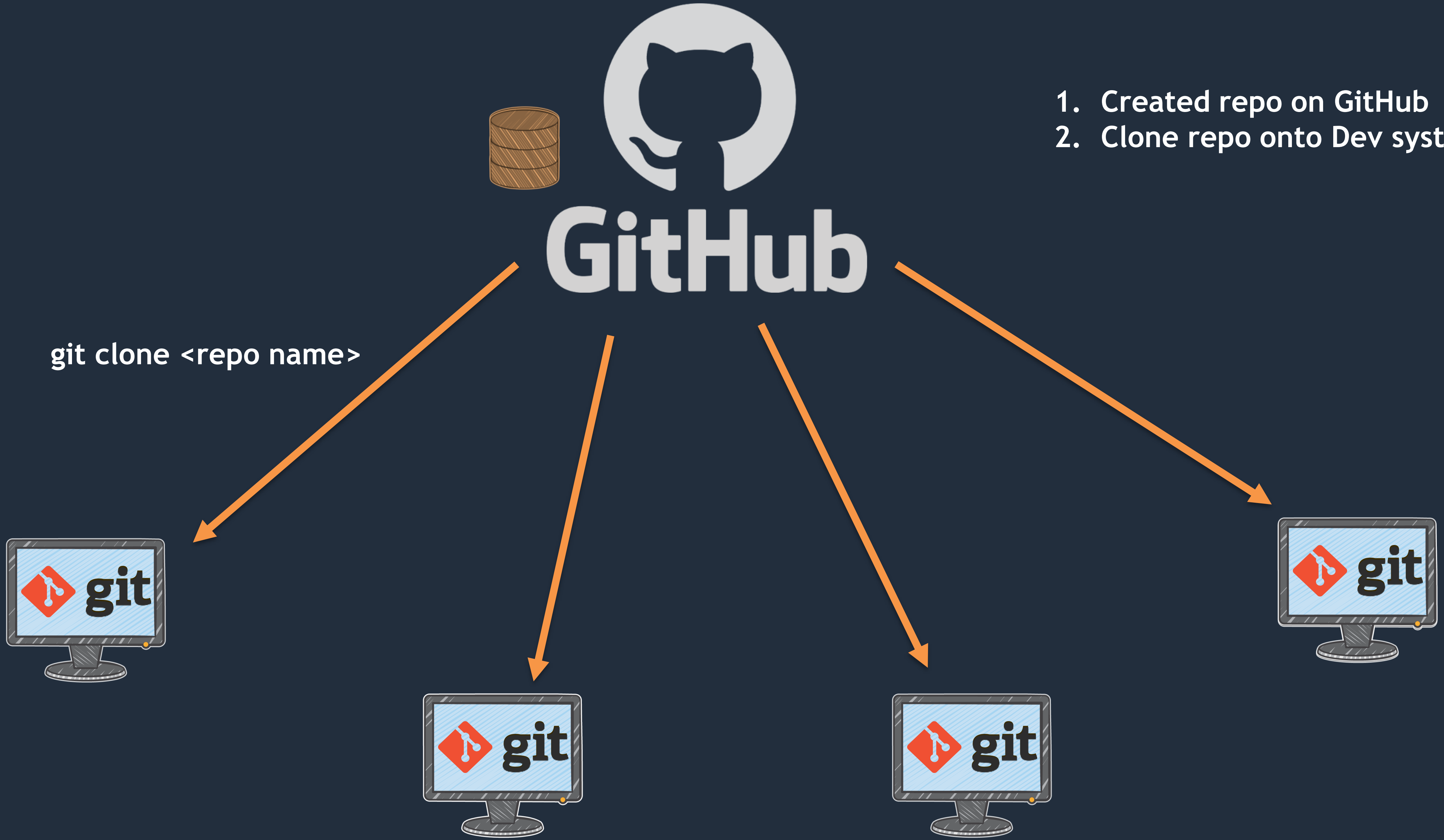
Push code onto GitHub Repository

Connecting to GitHub with SSH

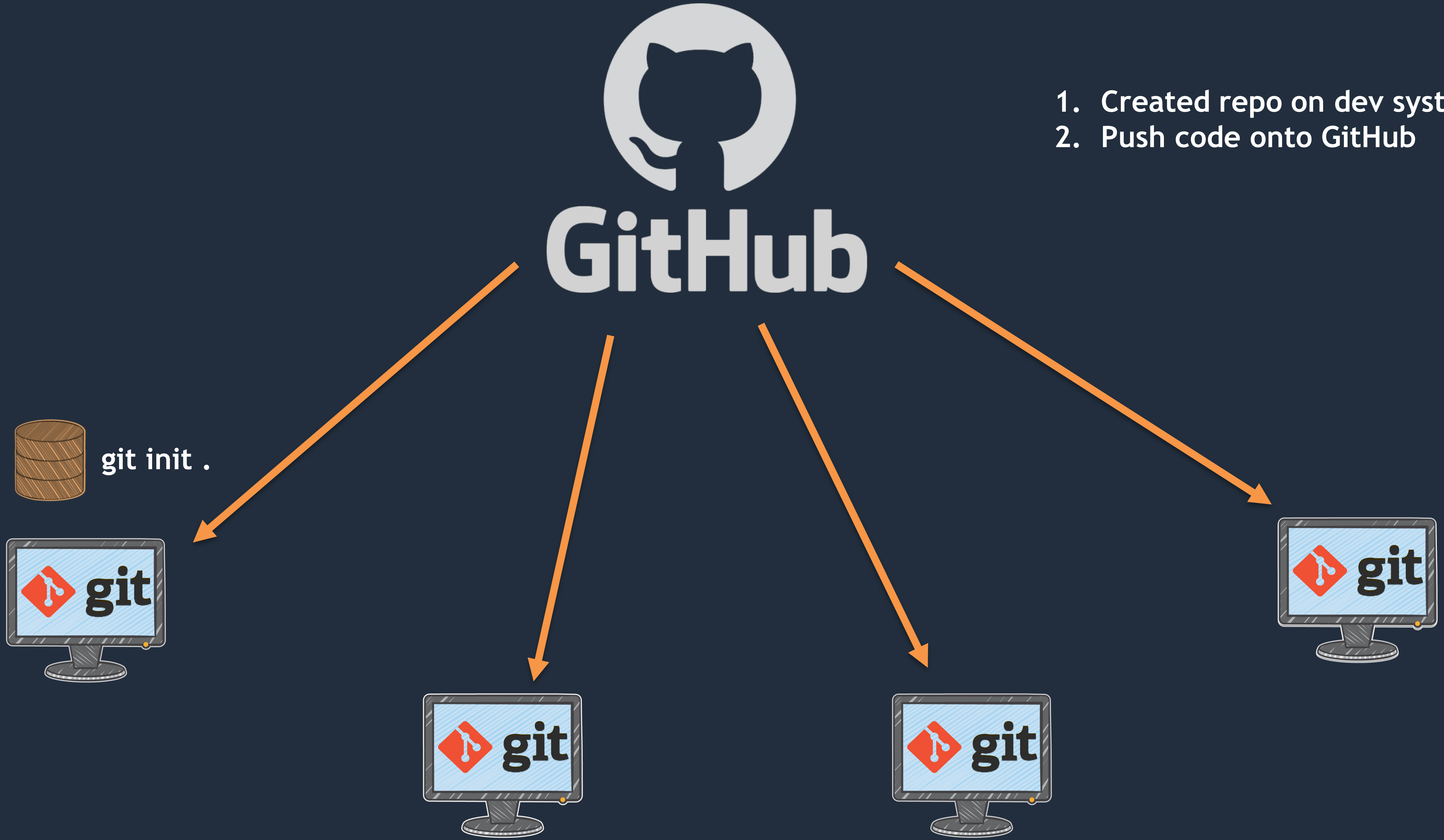

GitHub
git clone <repo name>
git
git
git
git


GitHub
git push origin master
git clone <Repo_URL>
git
git
git
git

1. Created repo on GitHub
2. Clone repo onto Dev system



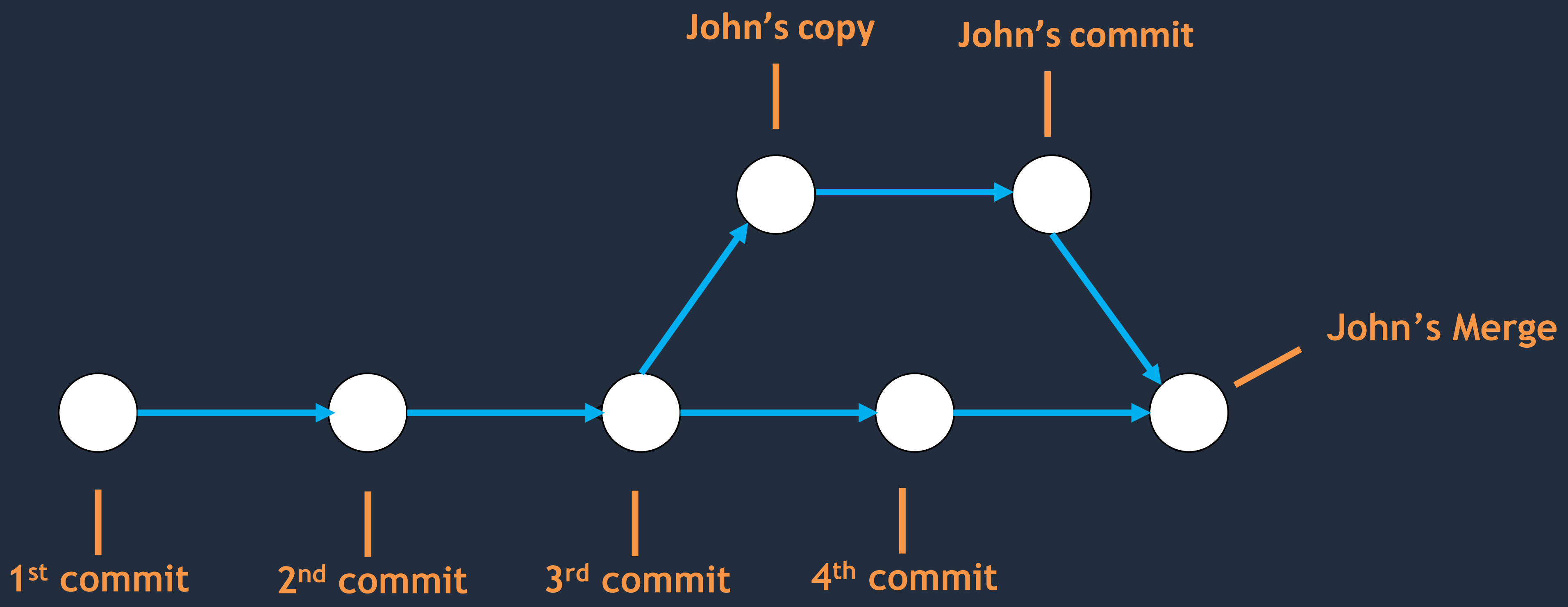
1. Created repo on dev system
2. Push code onto GitHub

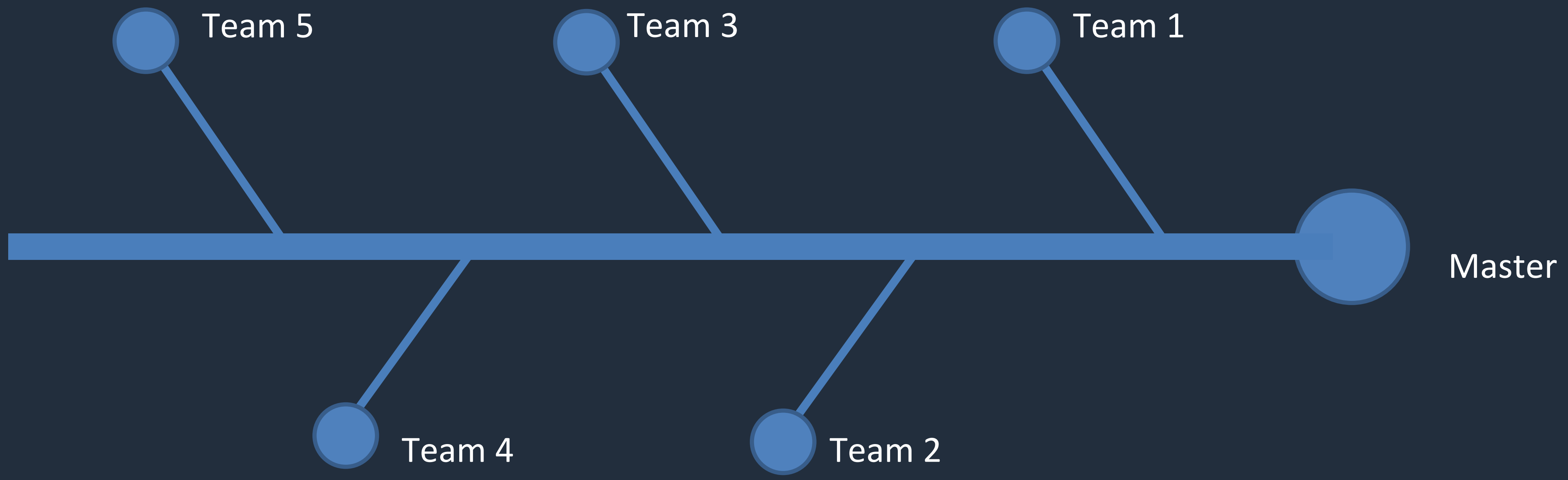


GitHub

Create a GitHub Account

Merge Conflicts




Team 5
Team 3
Team 1
Master
Team 4
Team 2

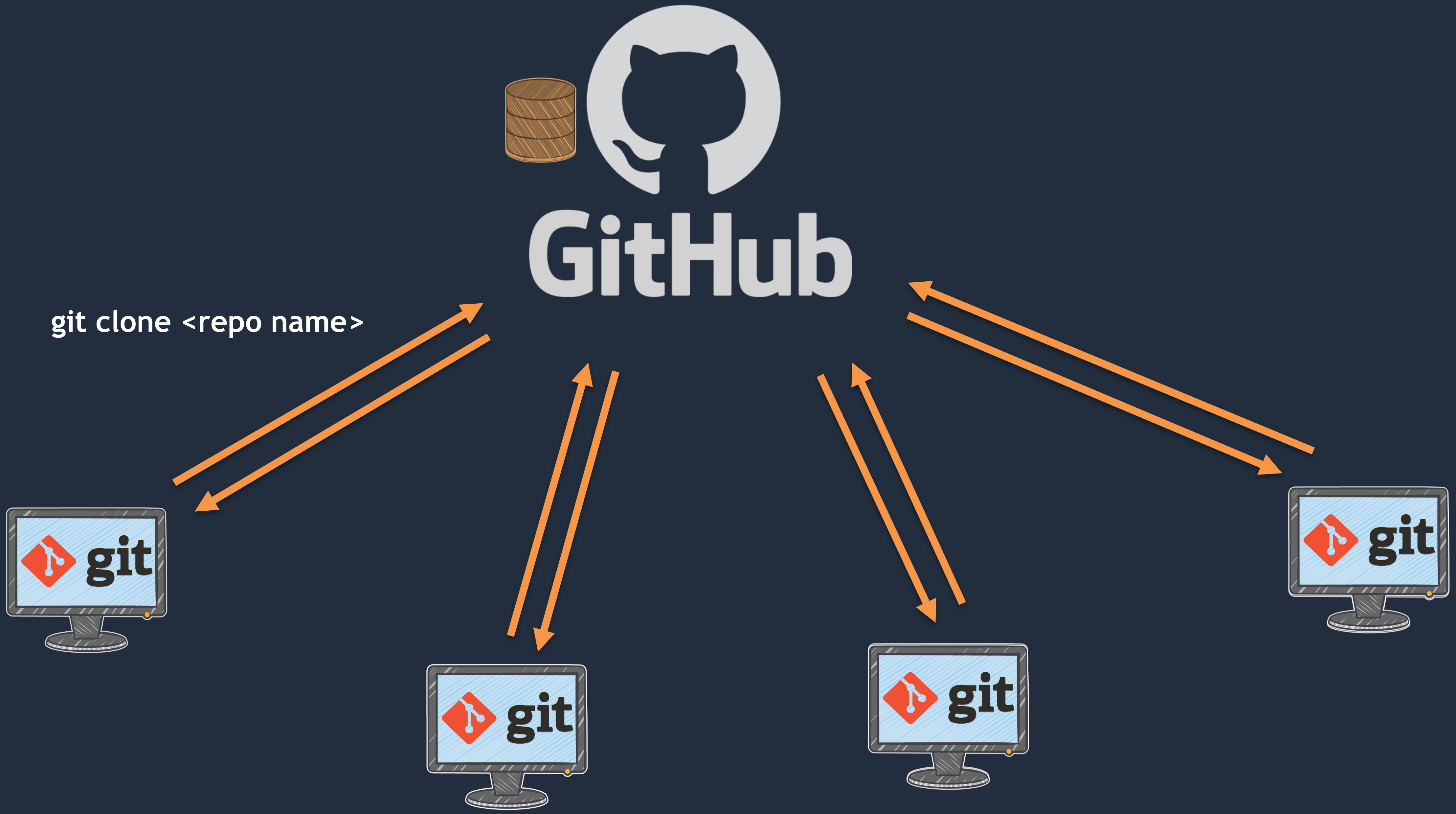
Git stages

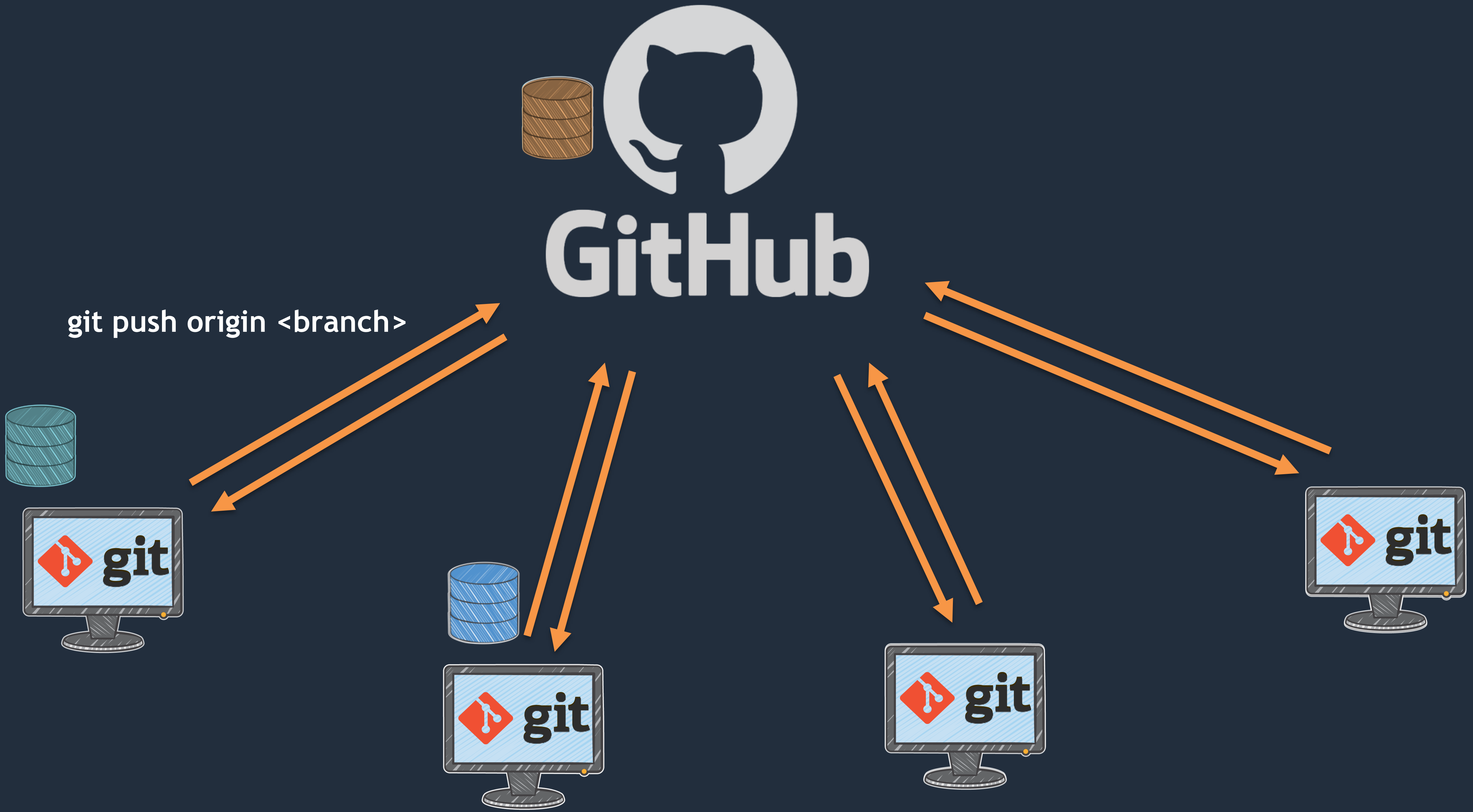
- Untraced - files created or updated but not part of version control system
- Staged - Files has been added to VCS but changes not yet commited
- Committed - Changes has been committed on local version control system
- Push - Changes has been updated in the centralized version control system

Most commonly,
forks are used to either propose changes to someone else's project

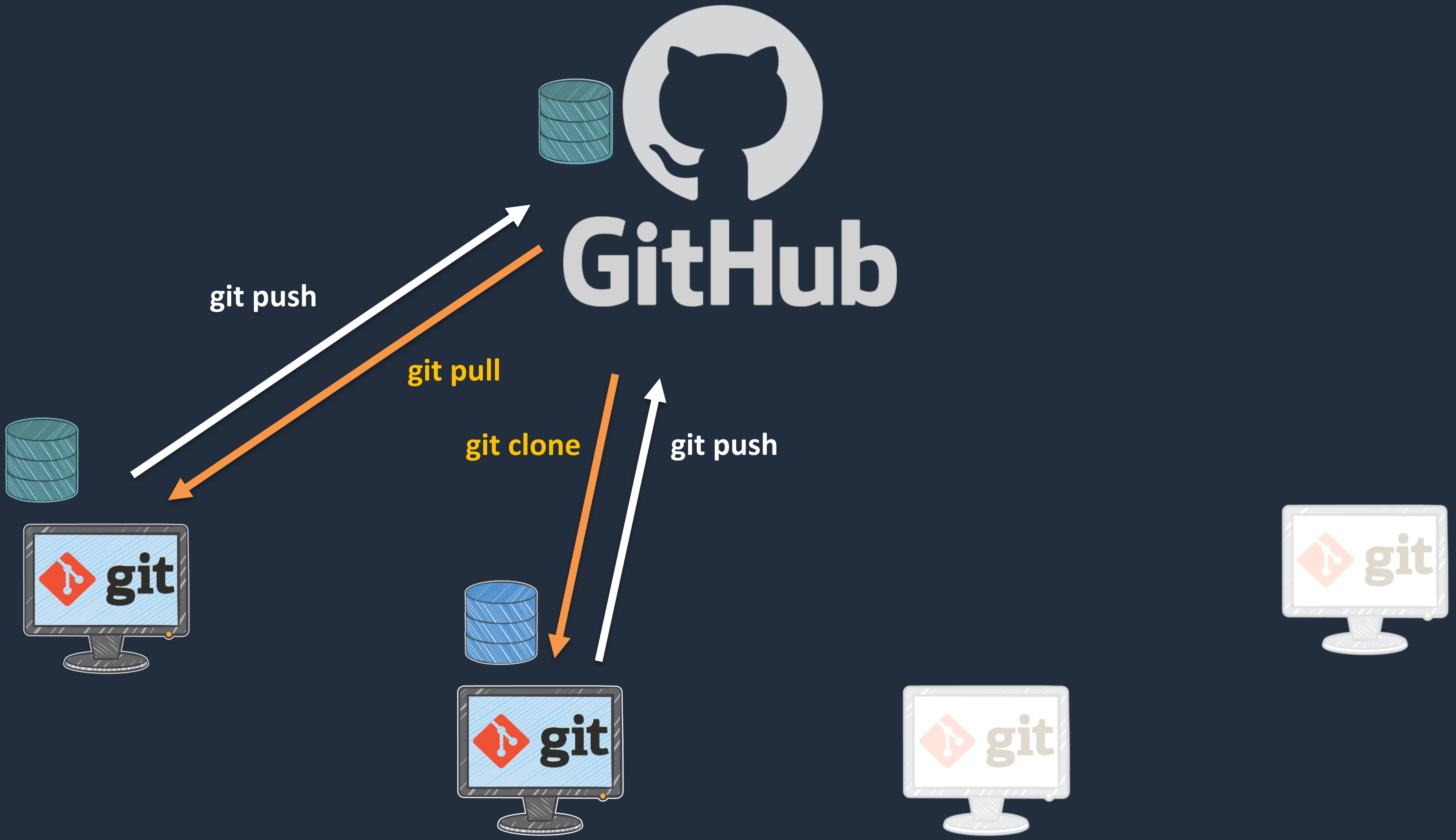
or

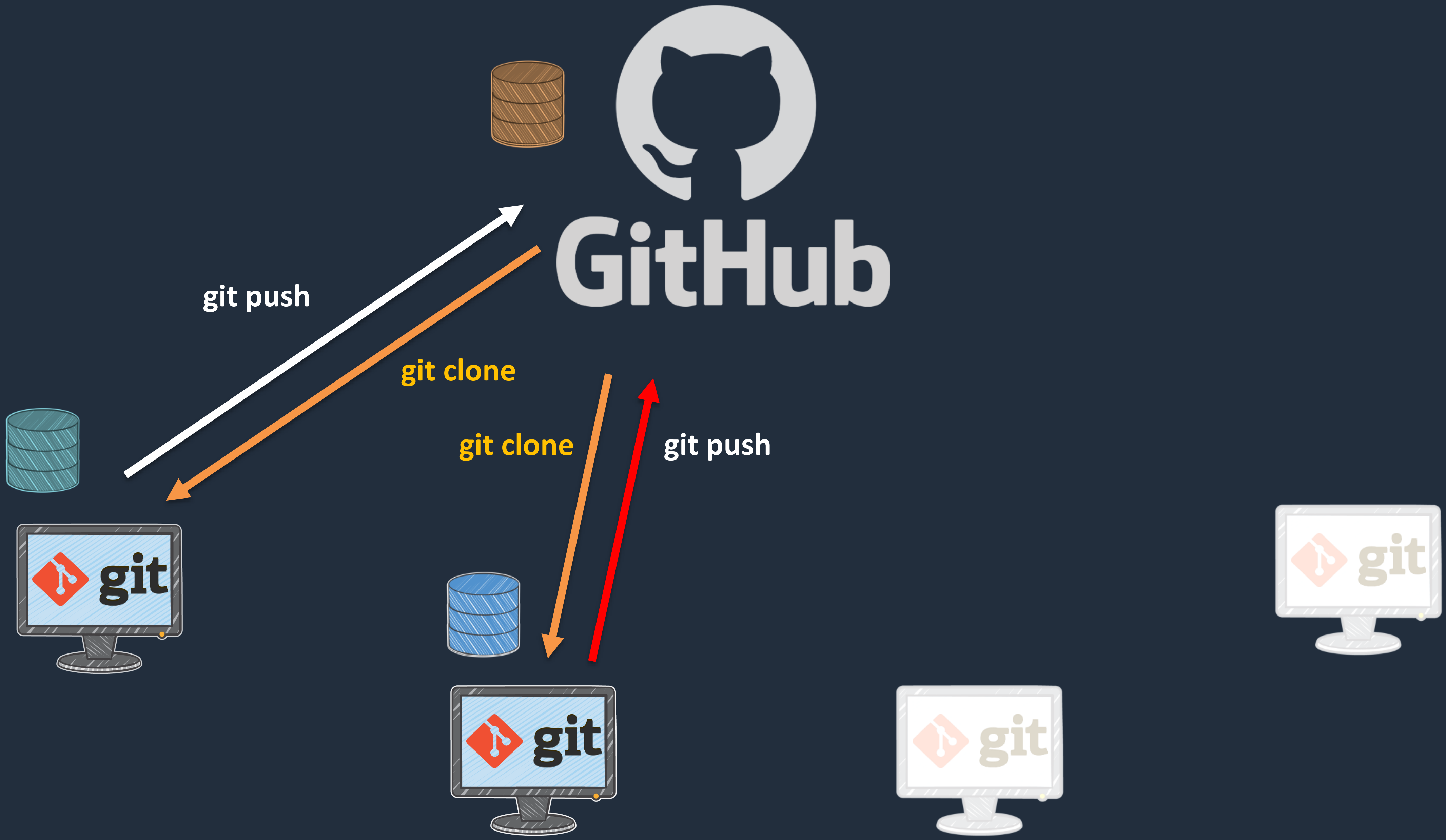
to use someone else's project as a starting point for your own idea.


GitHub
git clone <repo name>
git
git
git
git



git push origin <branch>


GitHub
git push
git pull
git clone
git push
git
git
git
git


GitHub
git push
git clone
git clone
git push
git
git
git
git

git - commonly used commands

git init .   - Create a Repo

git config --global user.name "USERNAME” - Setup username for git

git config --global user.email "USERNAME@EMAIL.COM" - Setup Email for git

git add .   - add changes to staging area

git commit -m "COMMIT_ID" - Commit changes to local repo

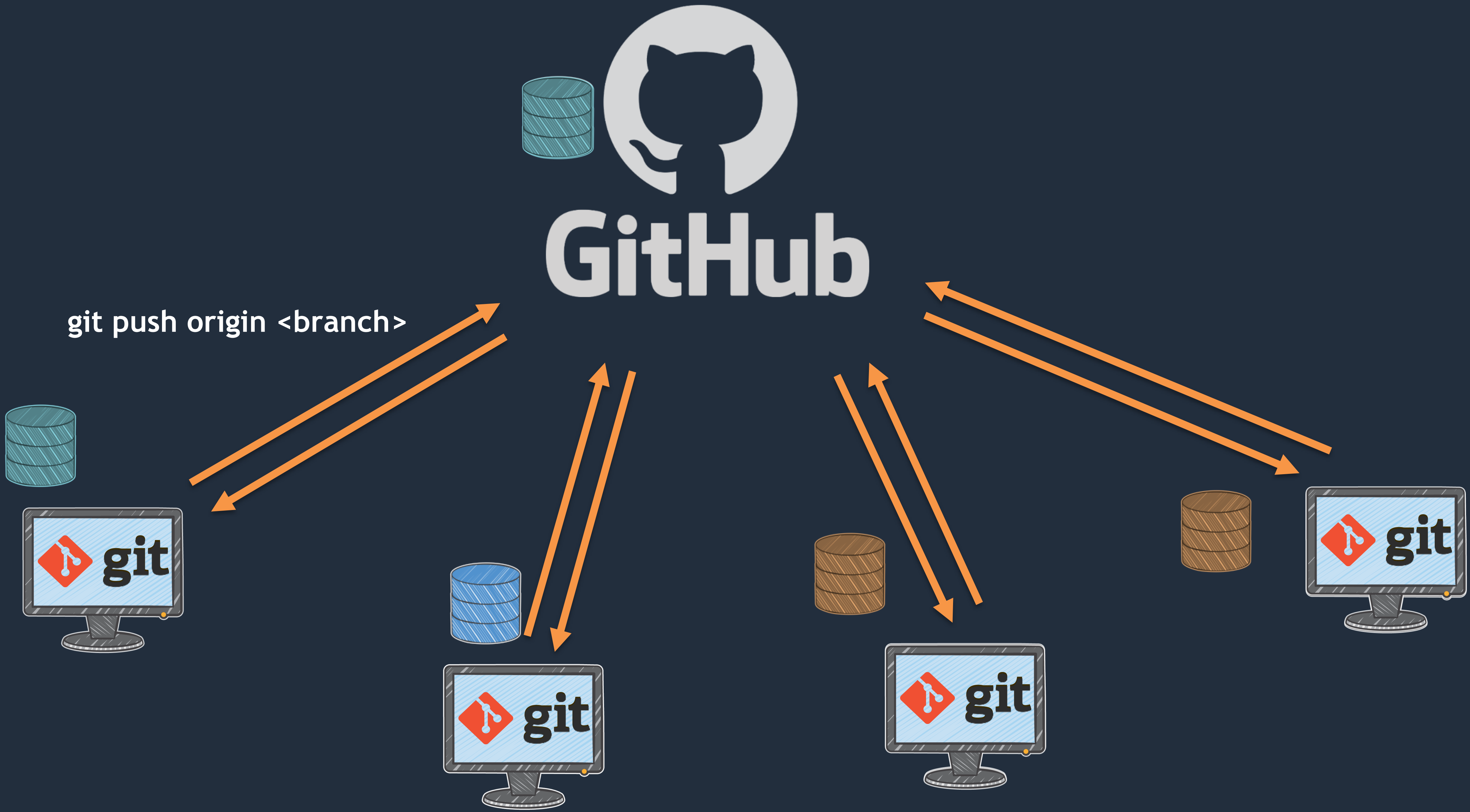
git push origin master - Push changes to remote Github repo

git remote add origin "GITHUB_REPO_URL”   – Associate a remote repo with local repo

git clone "GITHUB_REPO_URL”   – Clone remote repo to contributor system (1st time)

git pull - Pull updates from remote repo to contributor system (subsequent times)

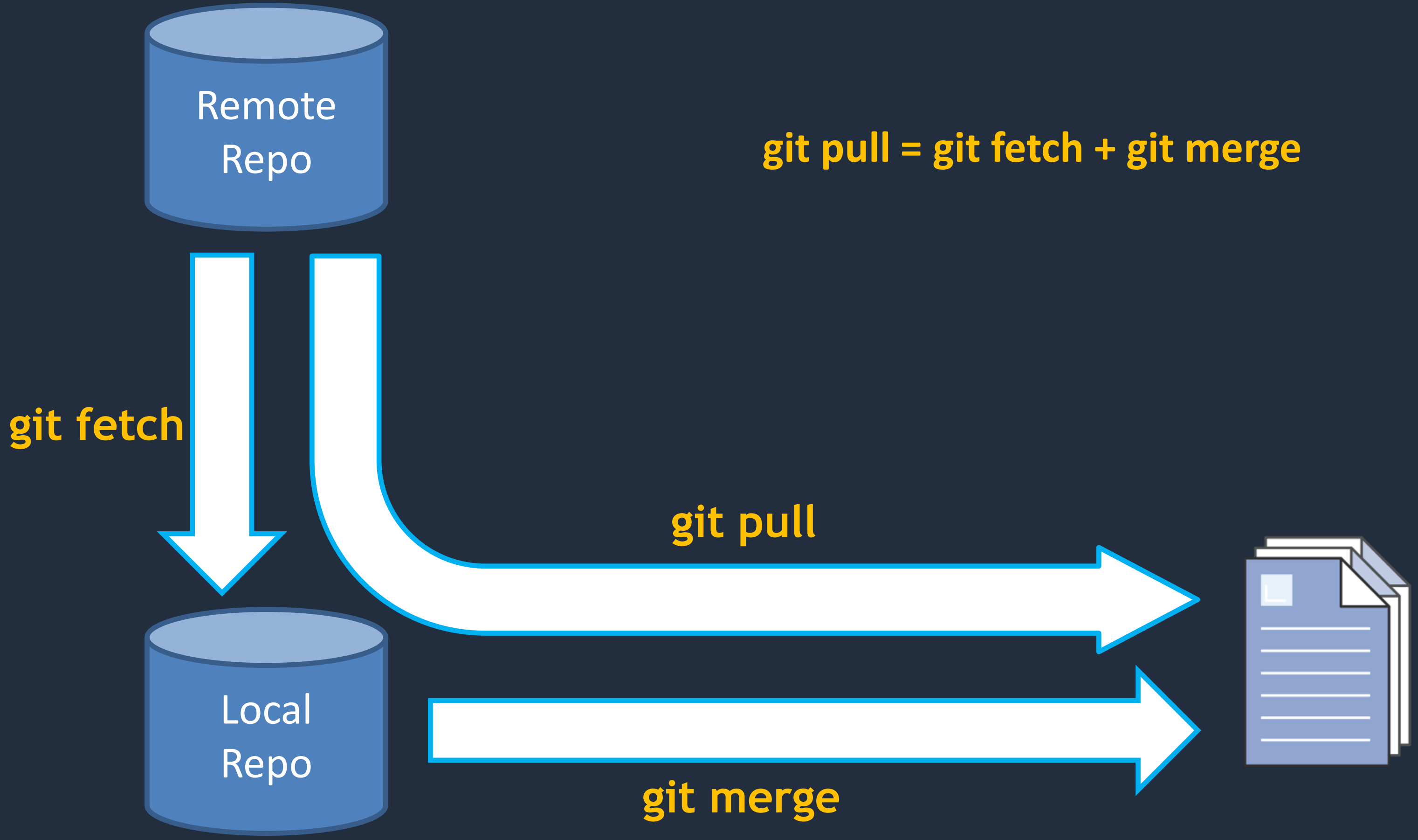
git log - list of the commits



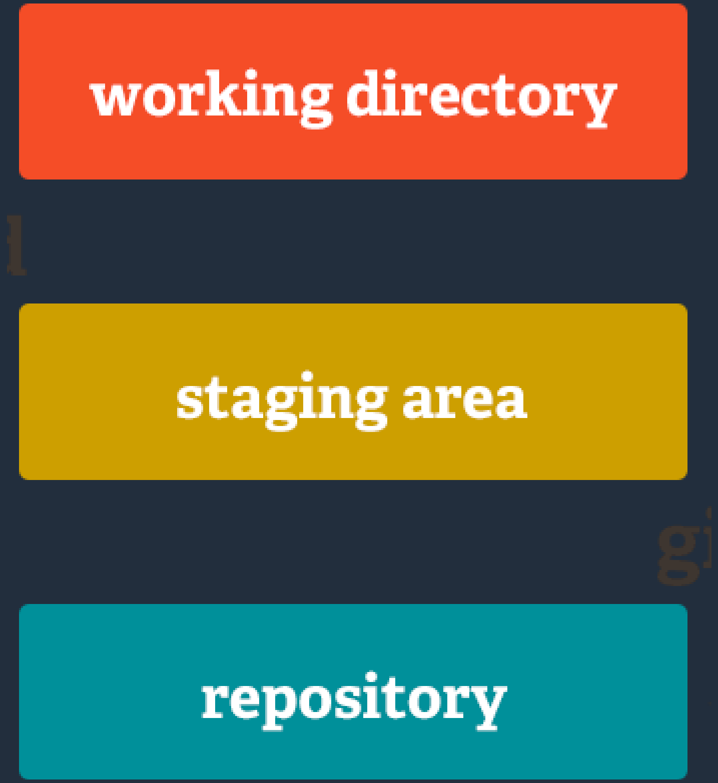
```
git push origin <branch>
```

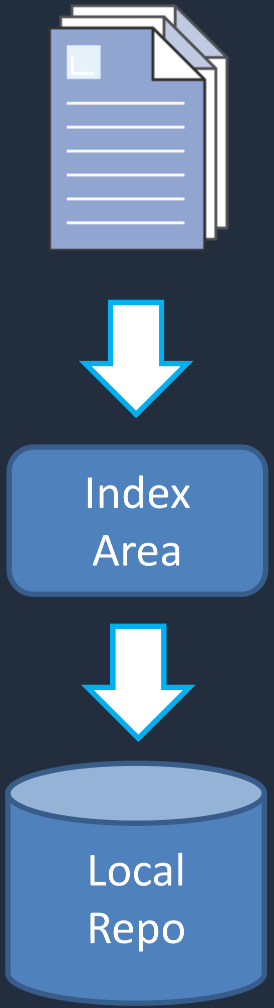
git fetch VS git pull

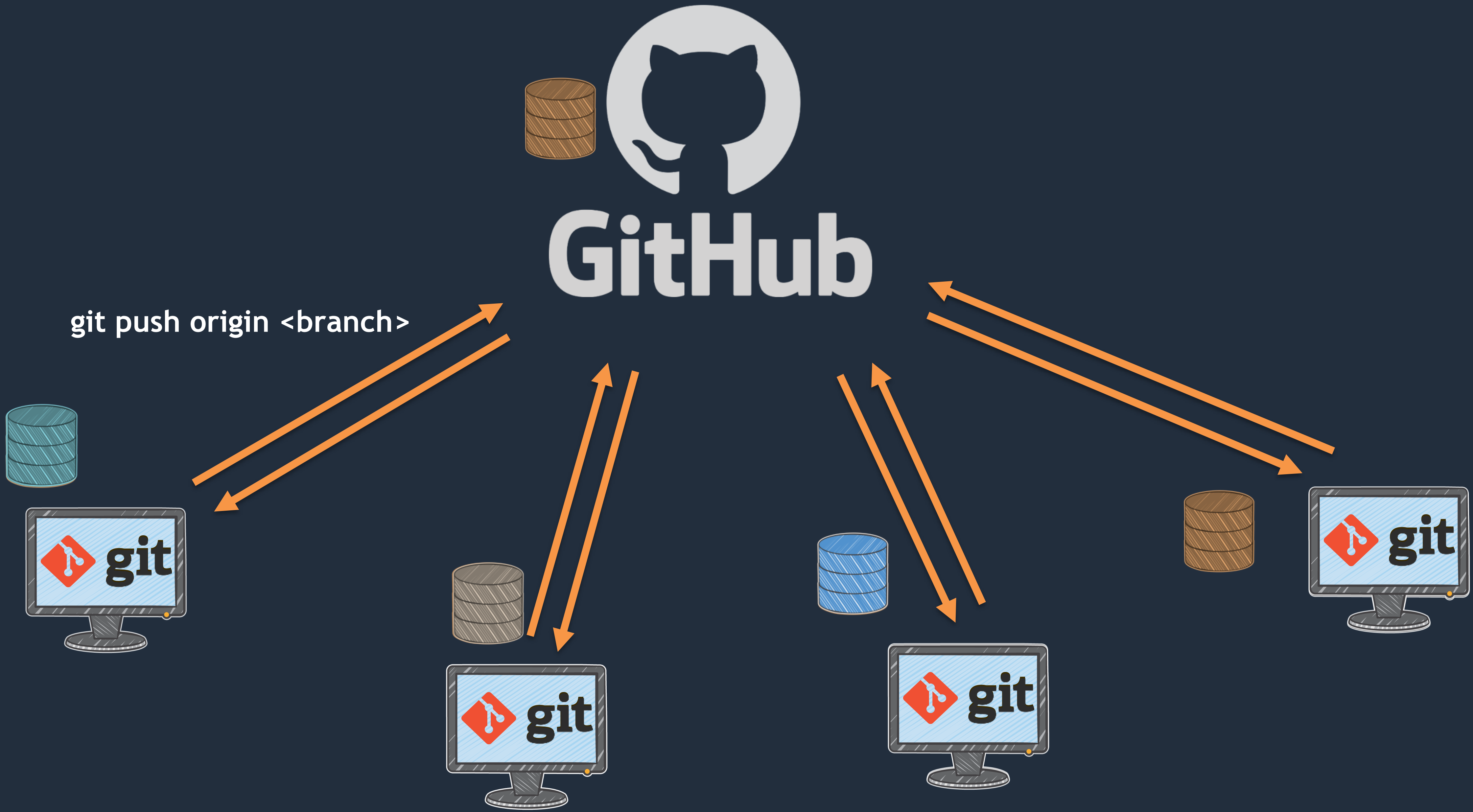
git pull = git fetch + git merge



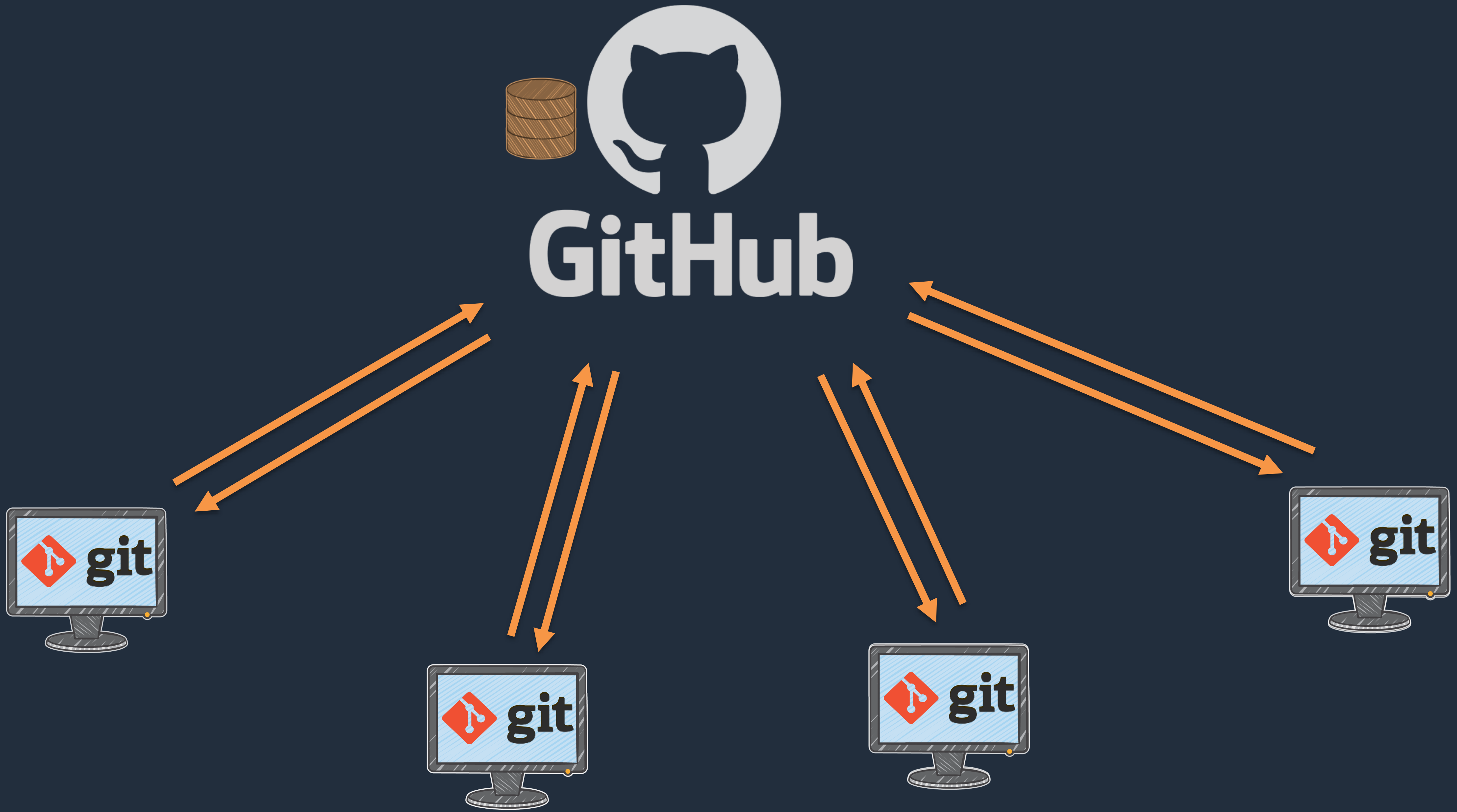
Git Stages


working directory
staging area
repository


Index Area
Local Repo



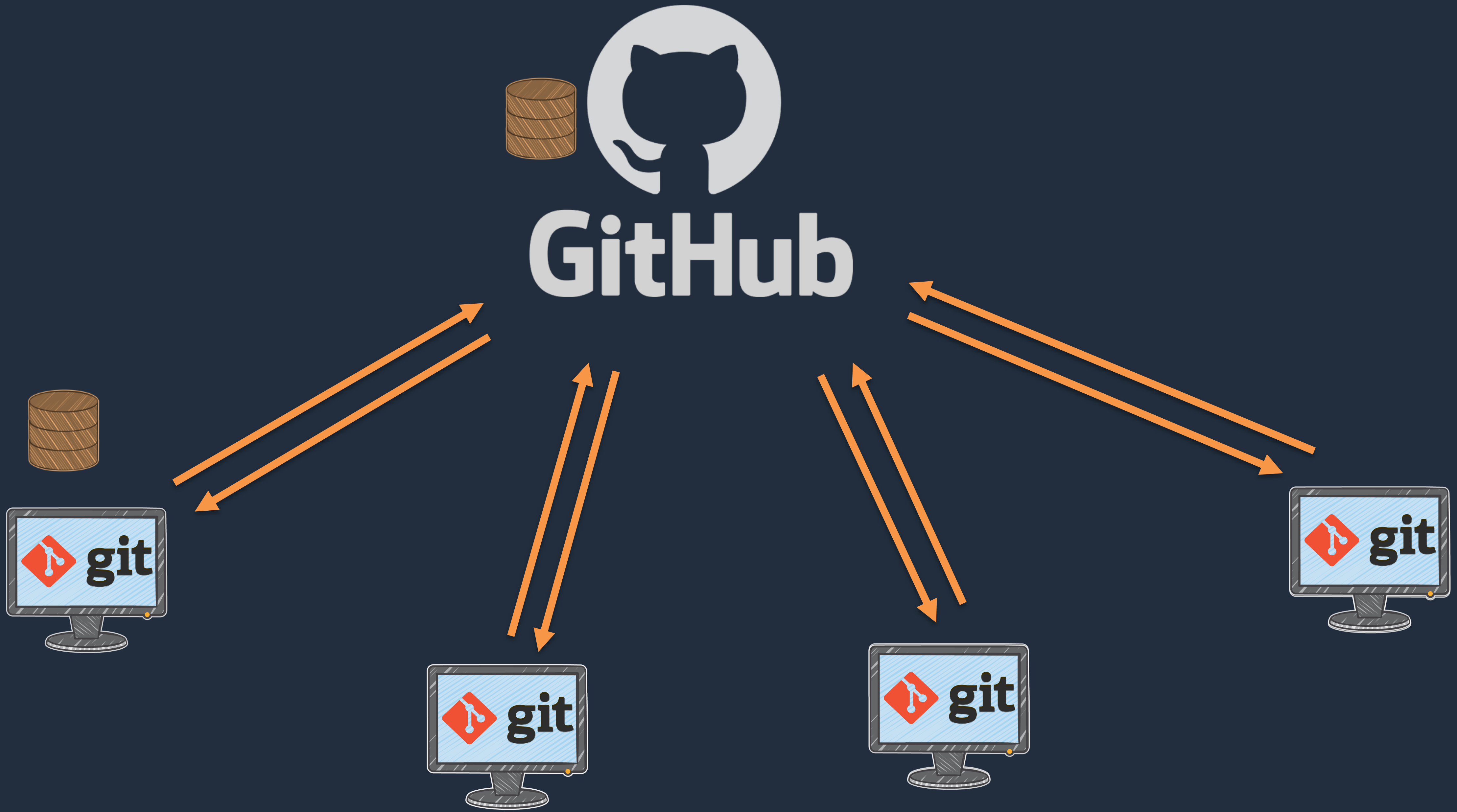
git push origin <branch>


GitHub
git
git
git
git

Create

Repository

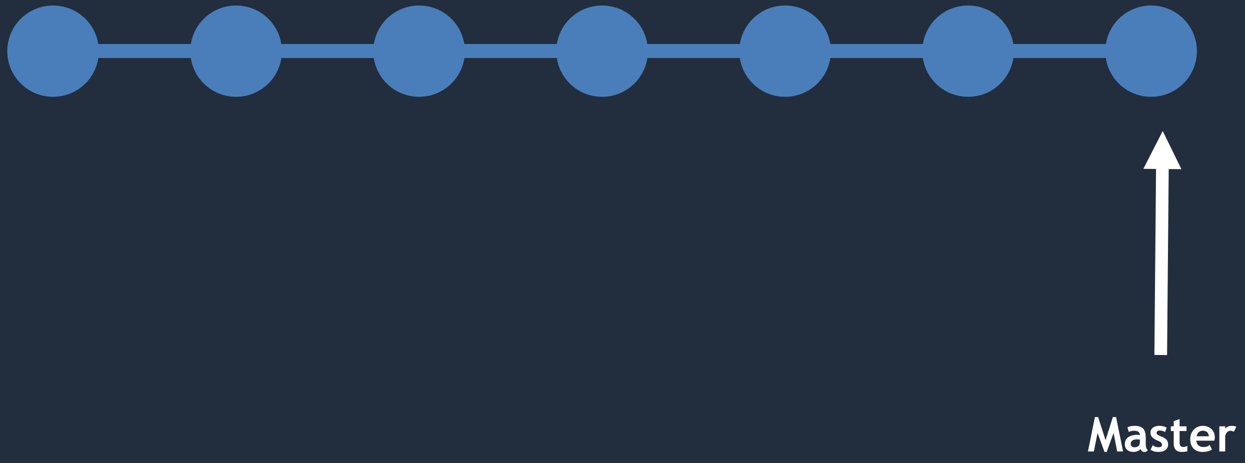
GitHub On GitHub

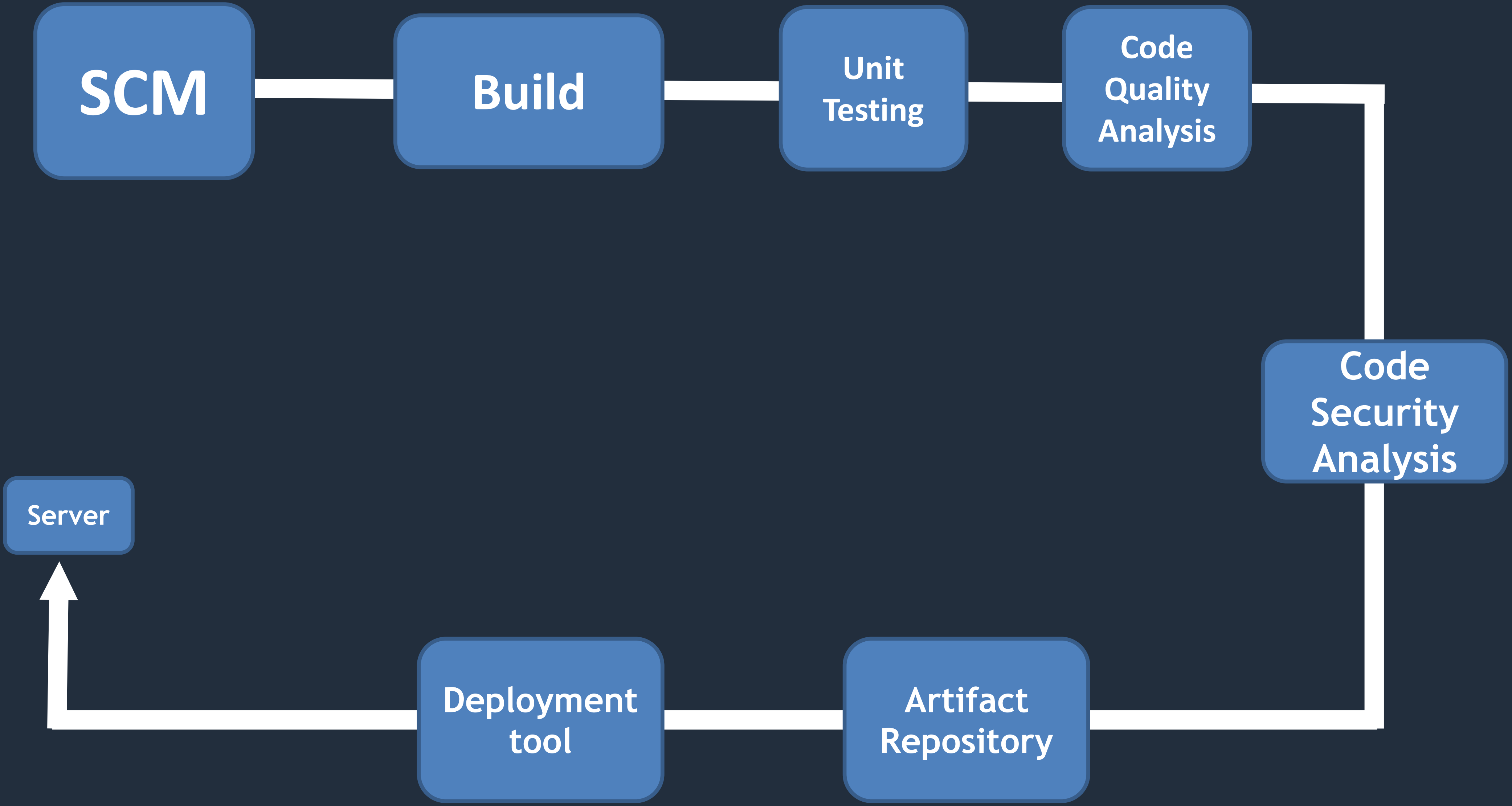

GitHub
git
git
git
git

git

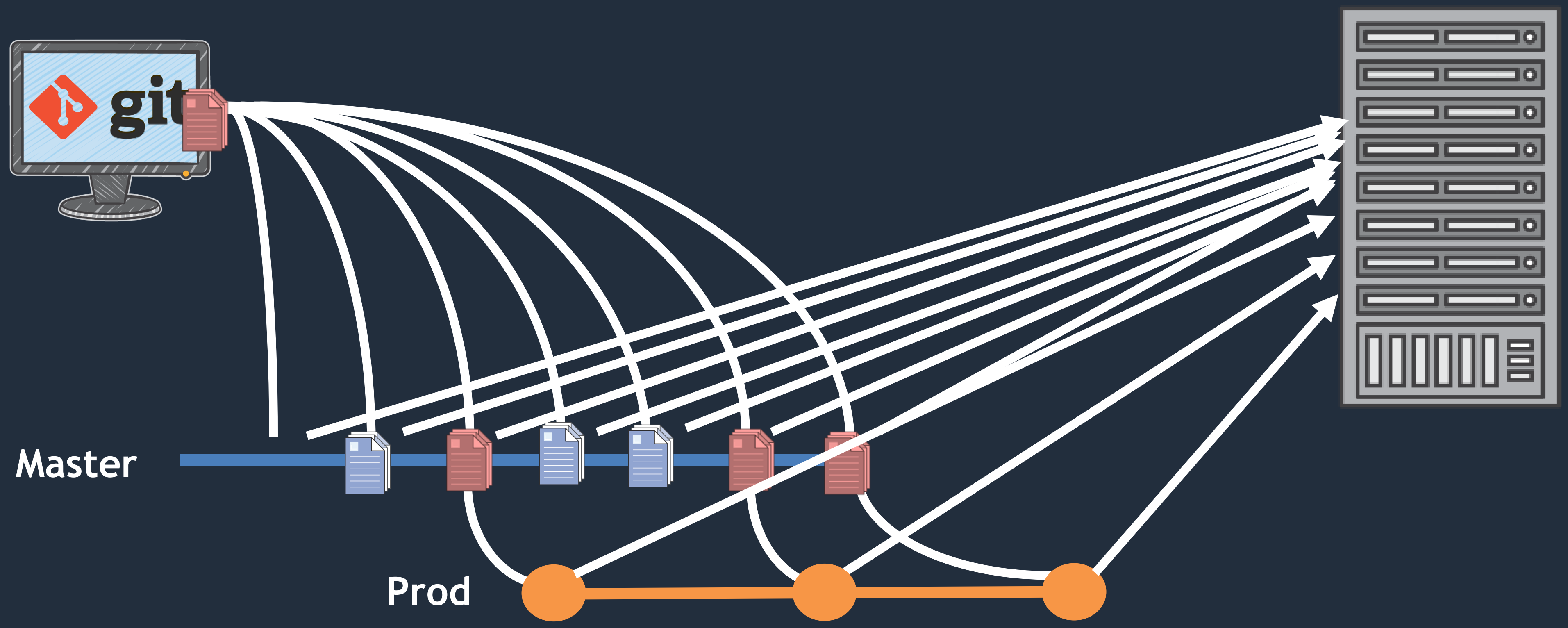
Branches

What is Branch?

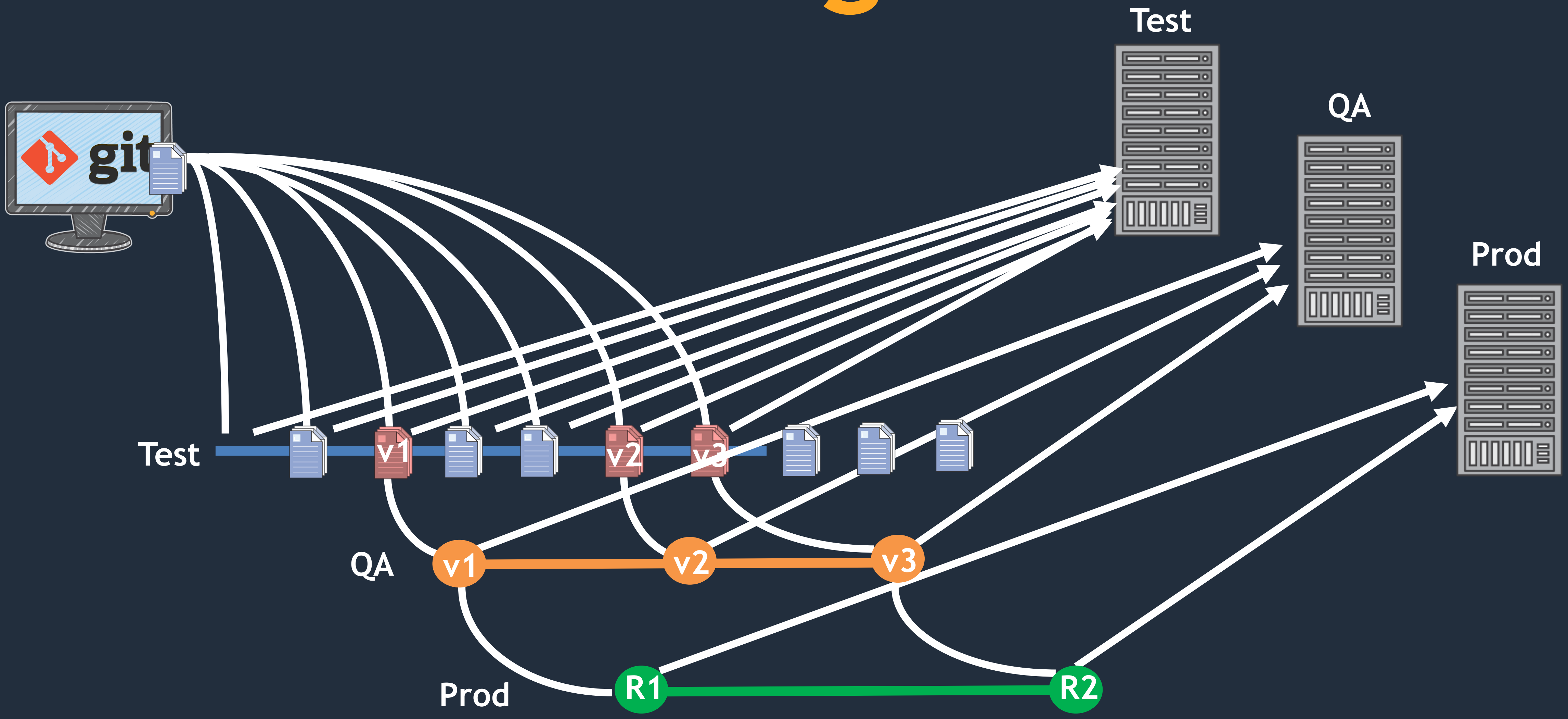



SCM
Build
Unit Testing
Code Quality Analysis
Code Security Analysis
Artifact Repository
Deployment tool
Server

Branches



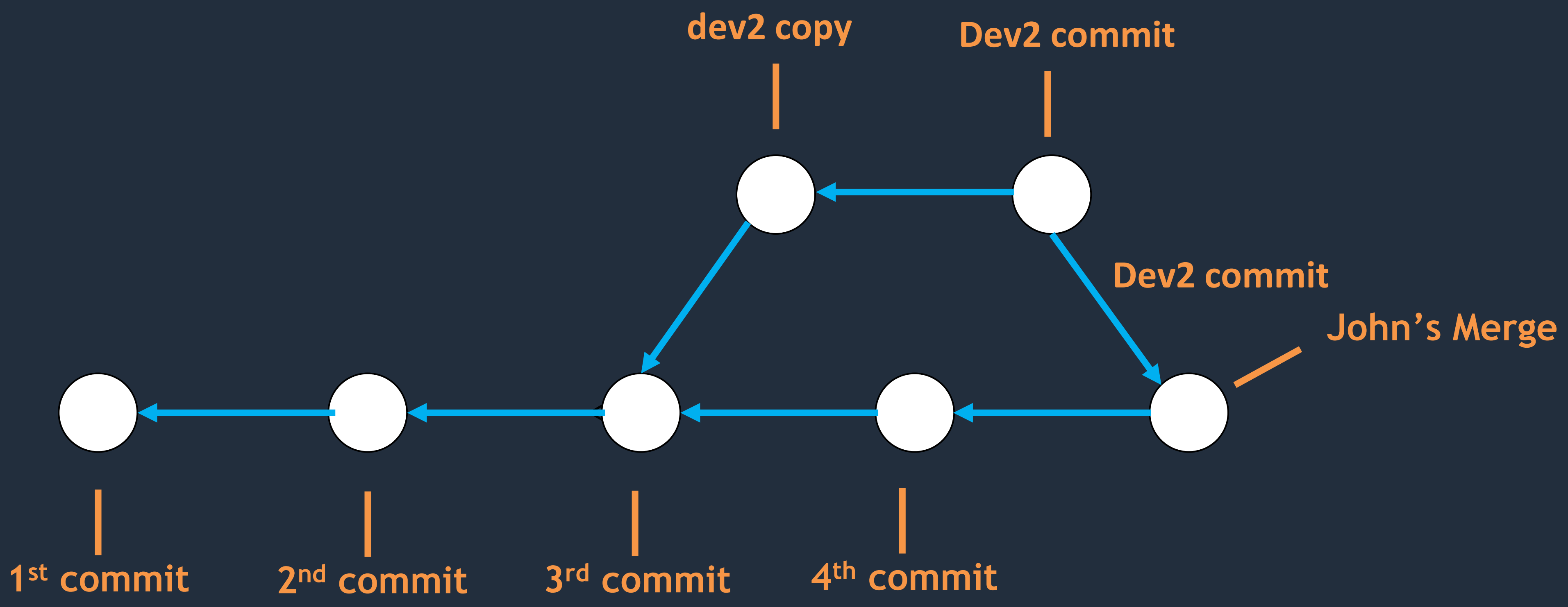
Branches Strategies



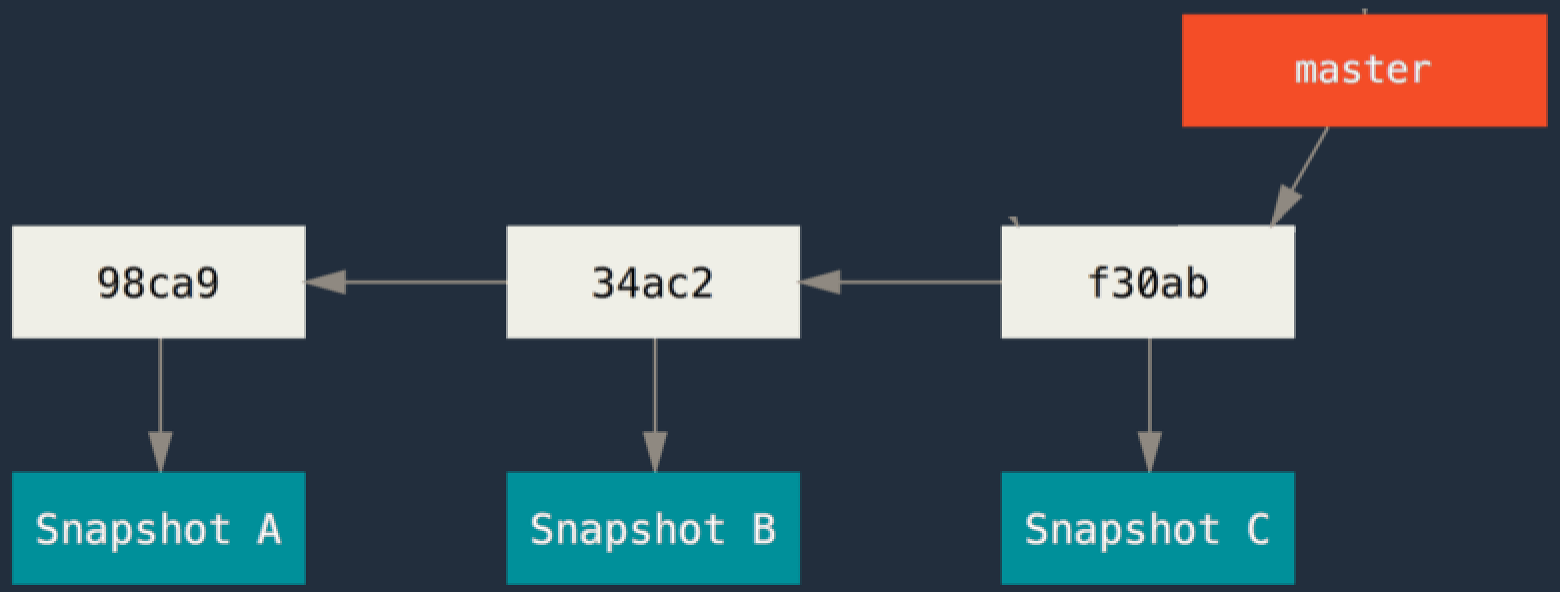
git

Merge Conflicts

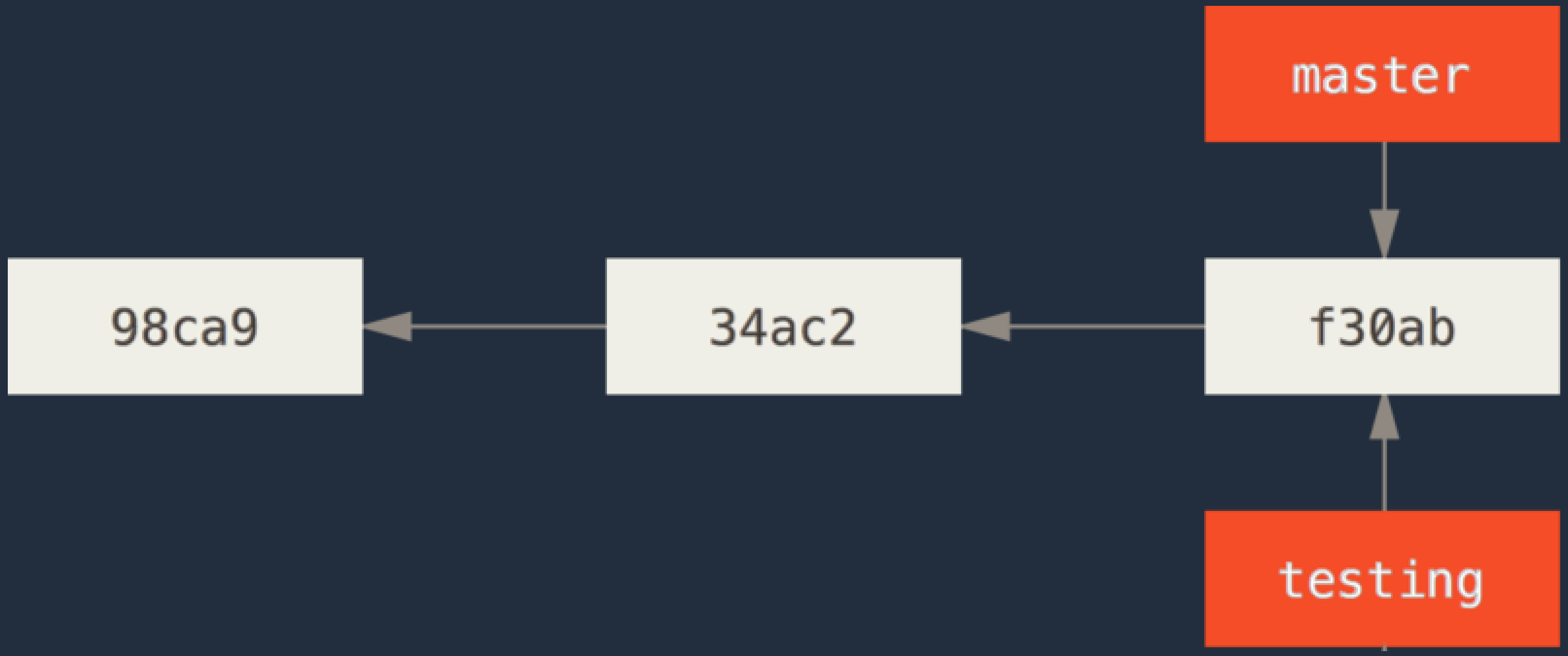
Merge Conflicts



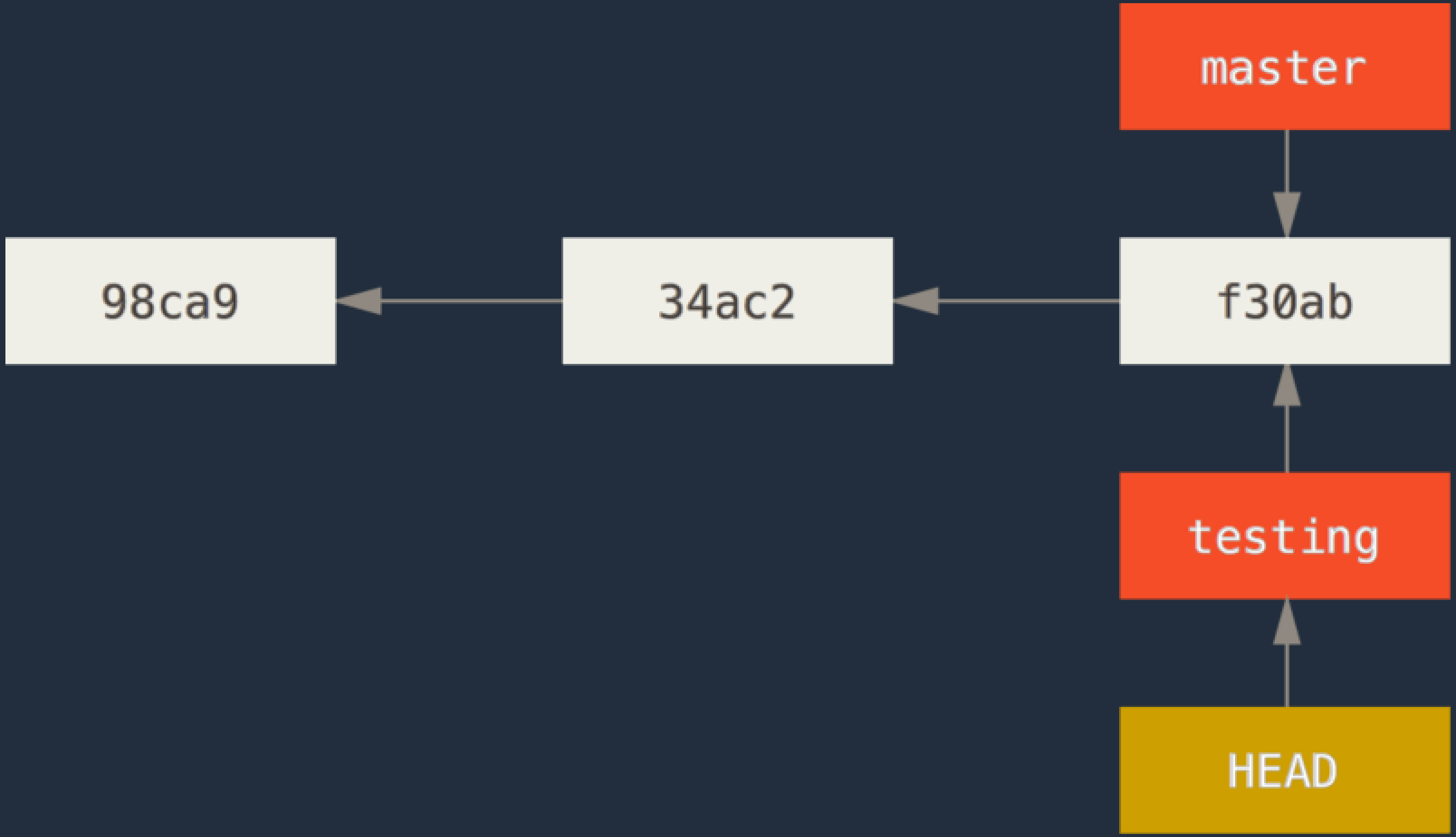
Branches



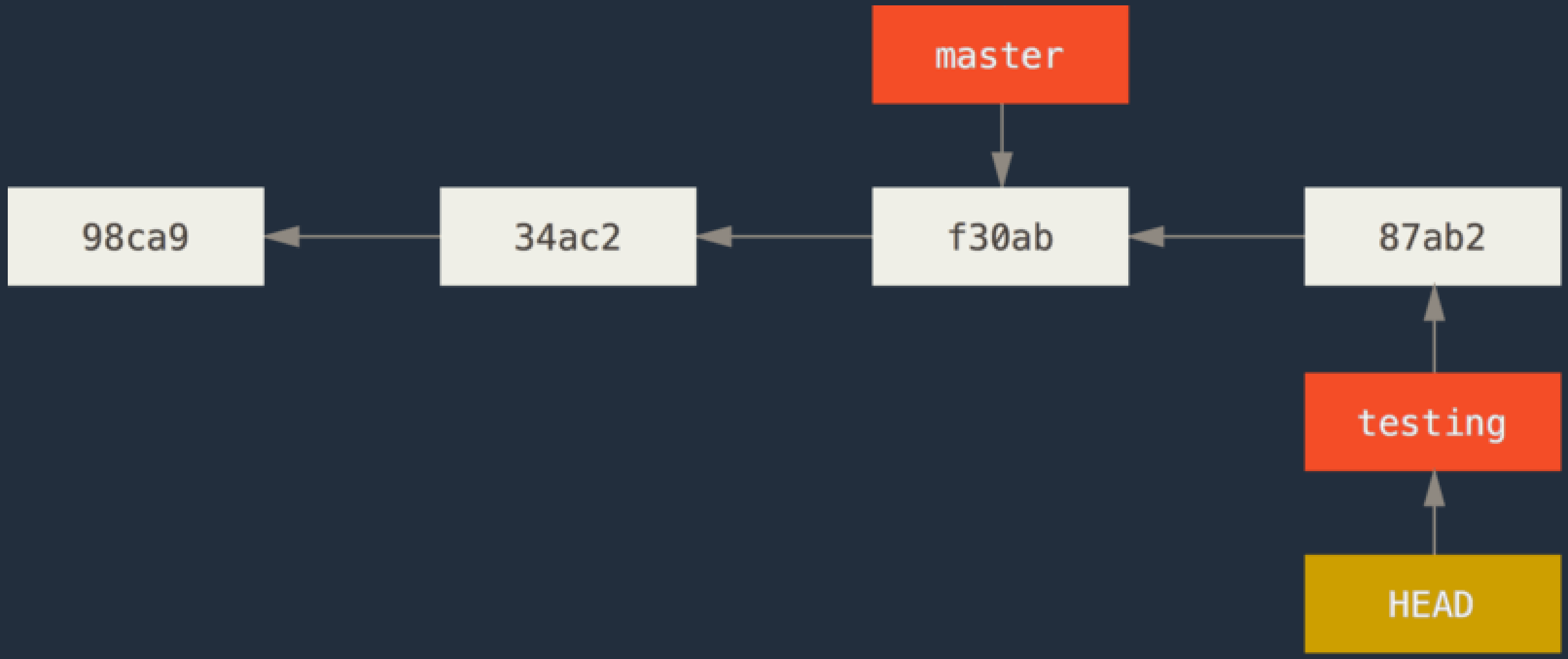
Creating Branch



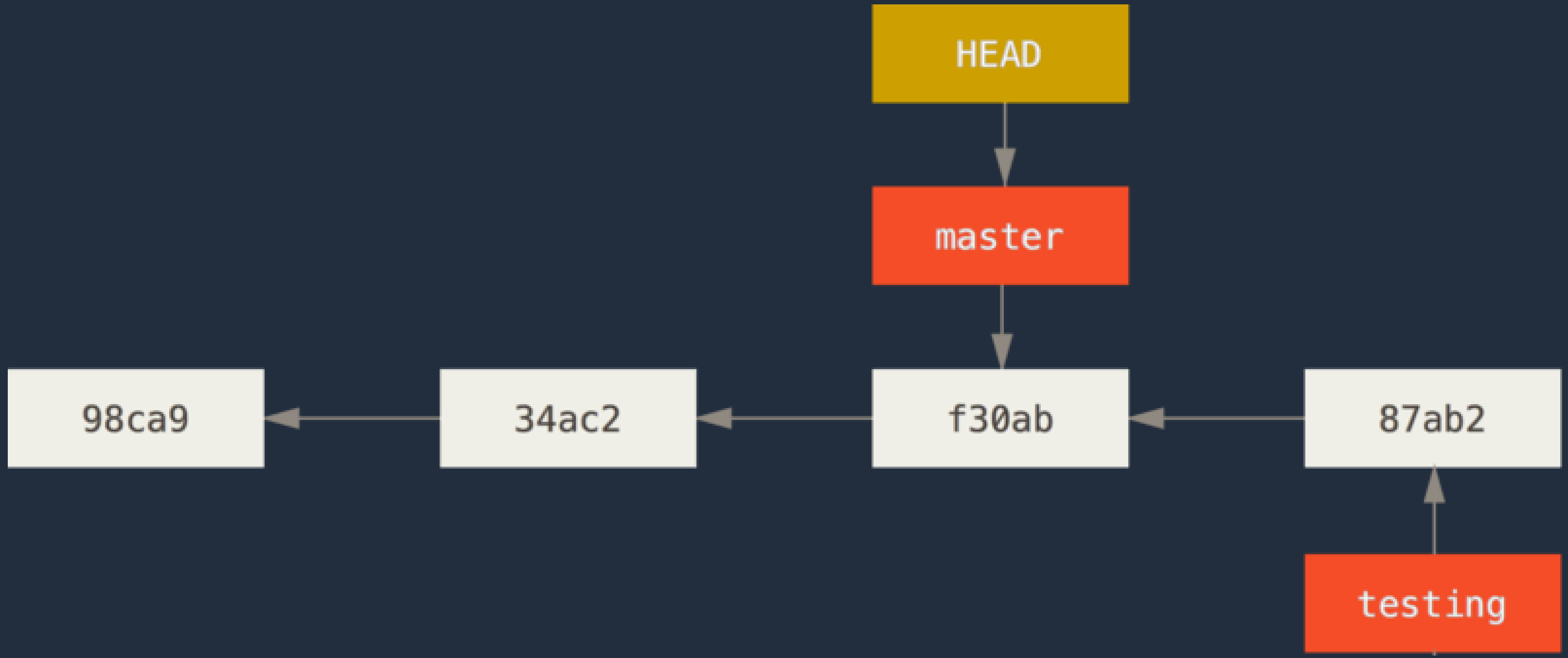
Switching Branch



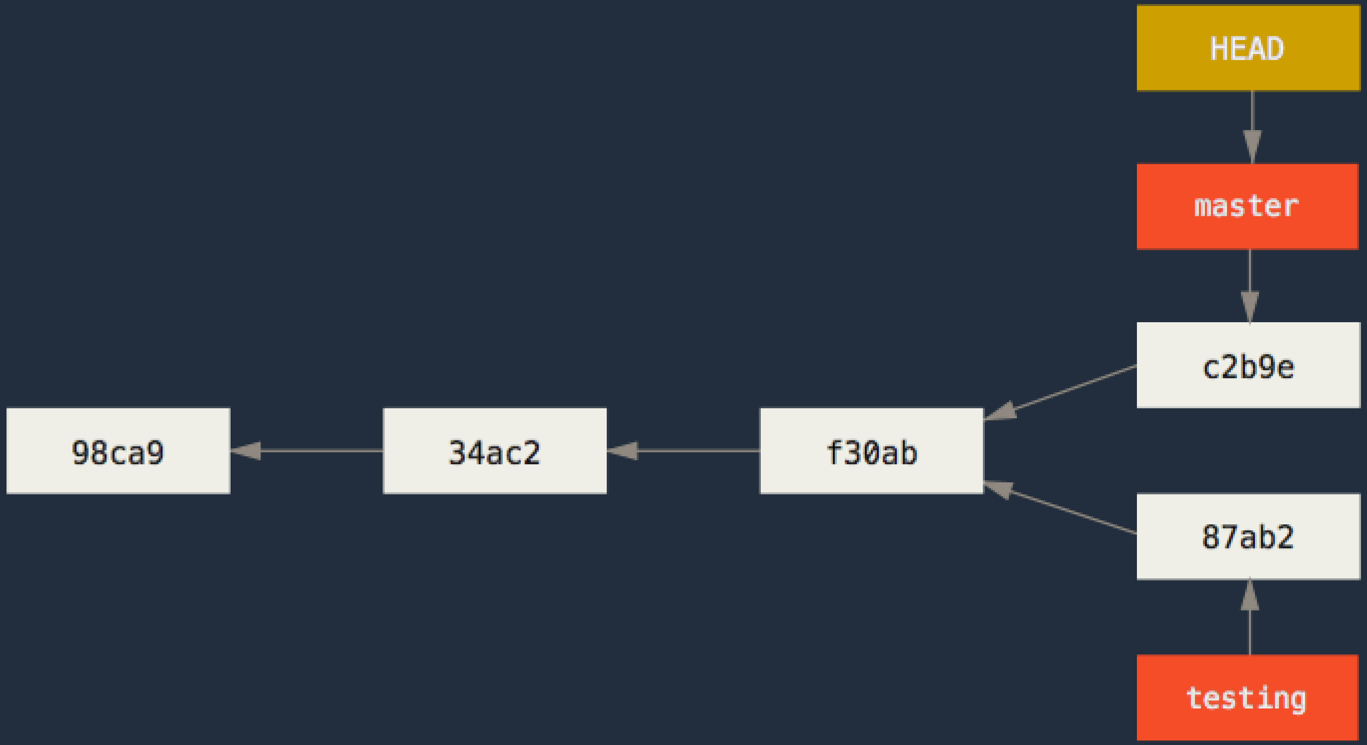
Commit a change on testing



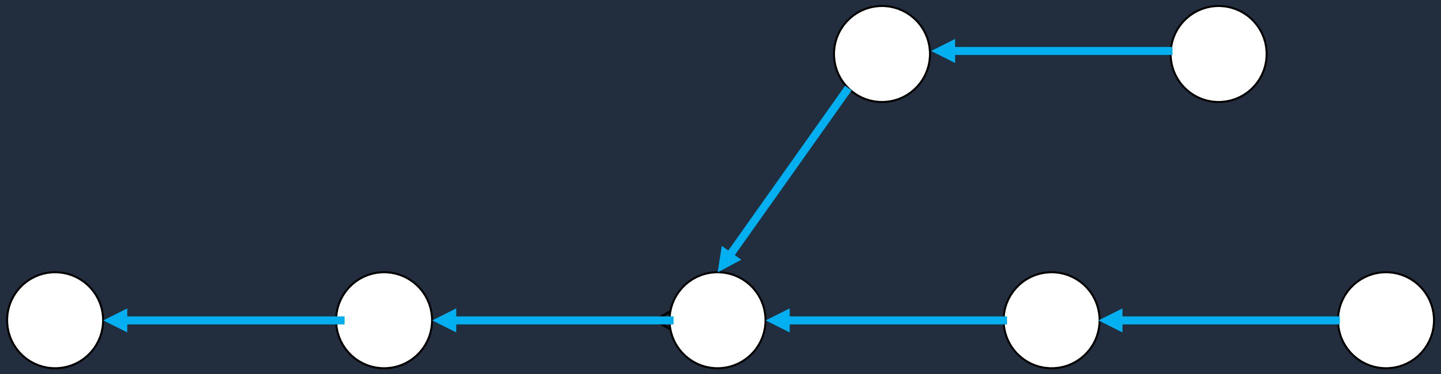
Switching branch

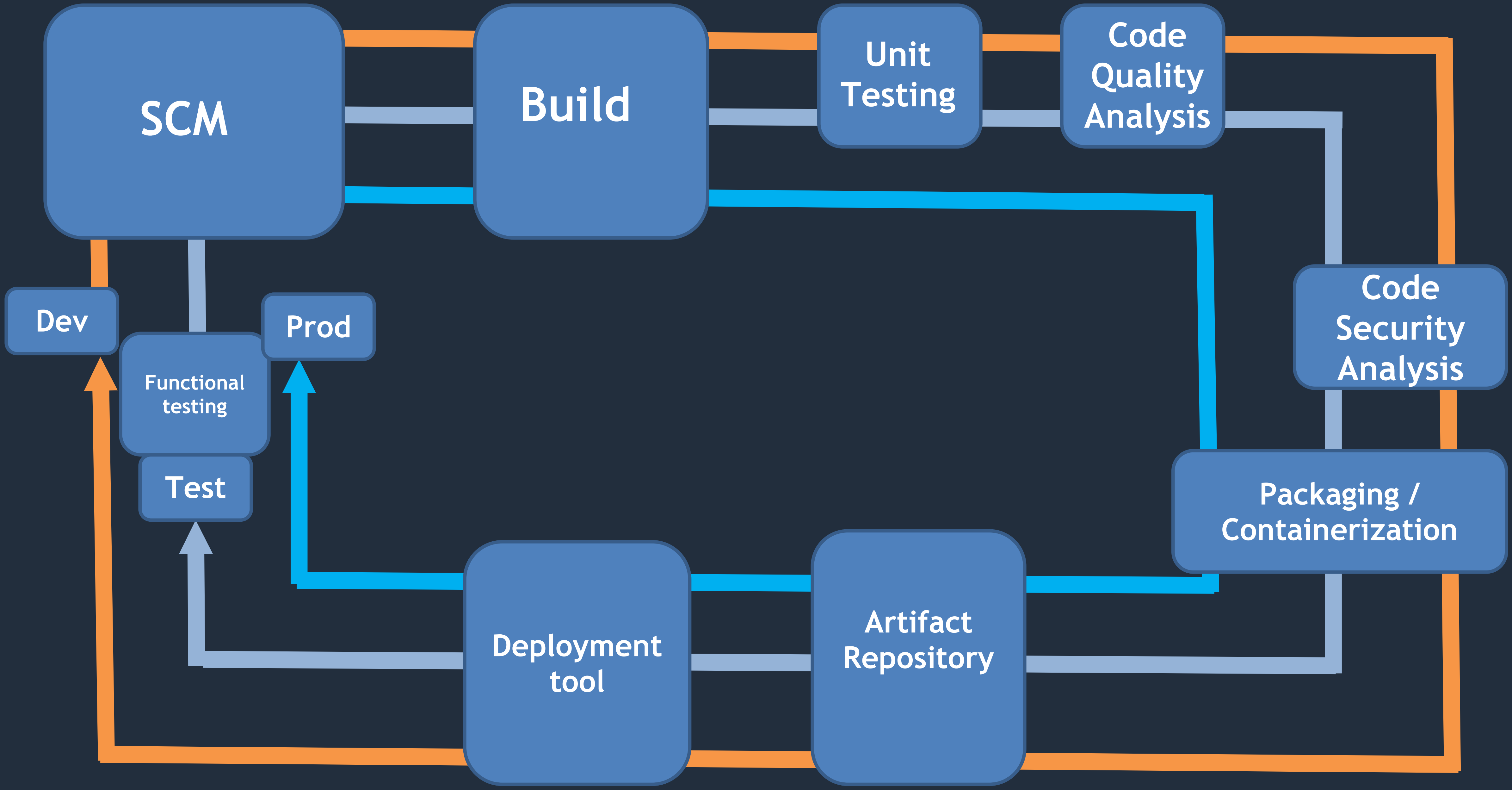


Committing changes



Branches




SCM
Build
Unit Testing
Code Quality Analysis
Code Security Analysis
Packaging / Containerization
Artifact Repository
Deployment tool
Dev
Functional testing
Test
Prod

GitHub

Forking a repository

GitHub

Working with Private Repository

GitHub

Protecting a Branch

GitHub

Collaborator

git

Tags

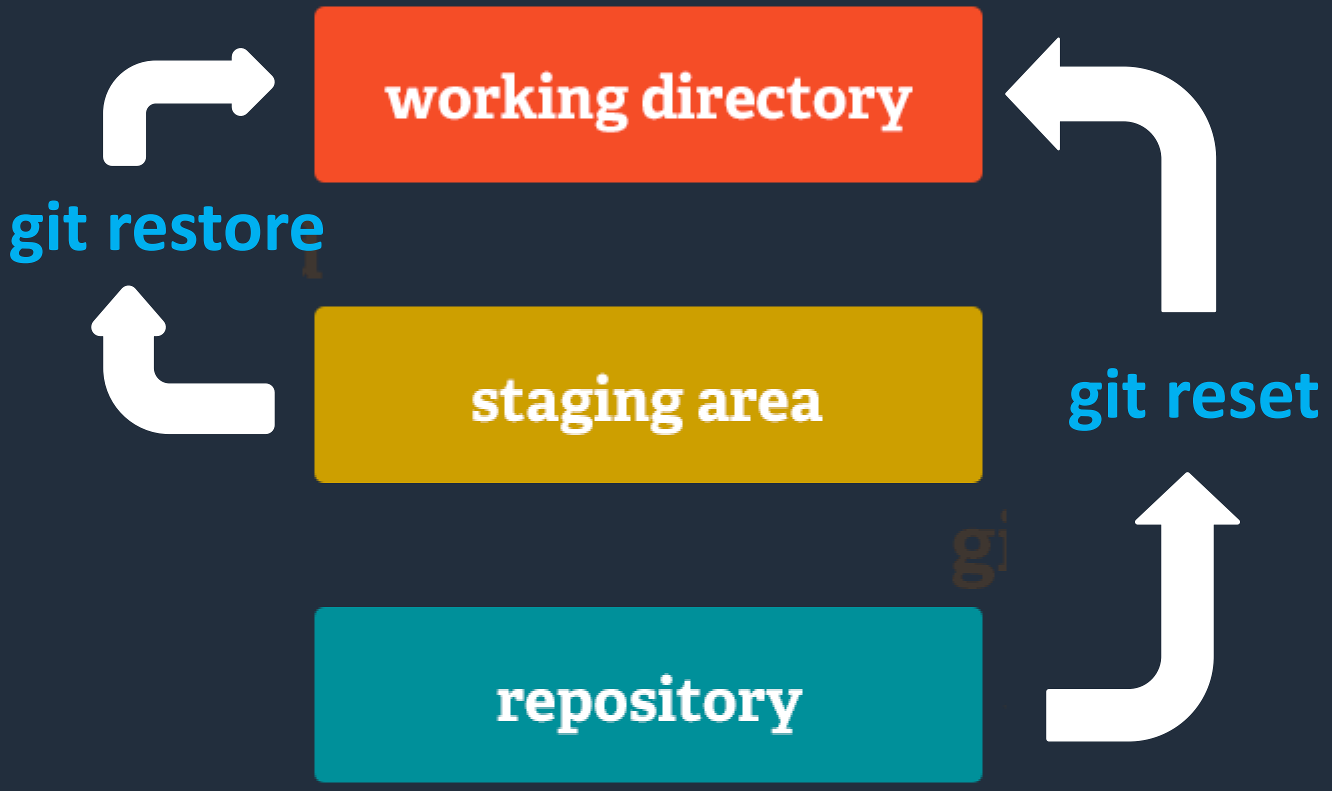
Tag a commit

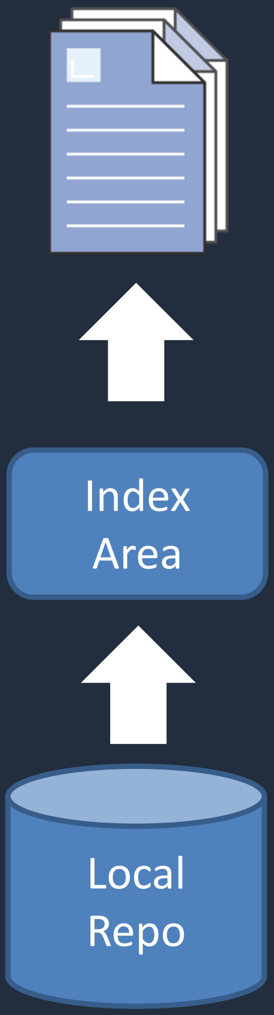
git Reverting Changes

git

.gitignore file

Revert Changes



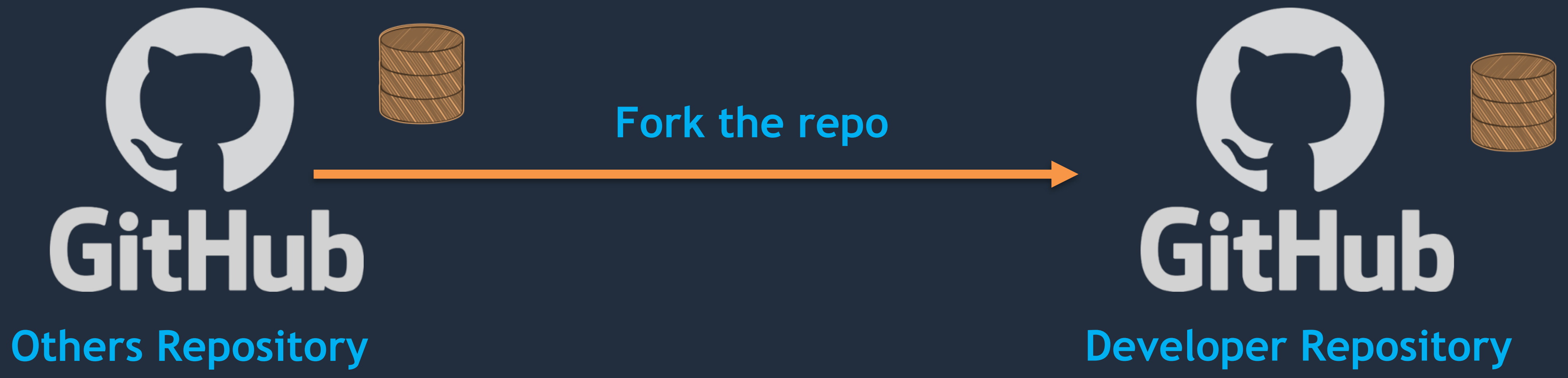


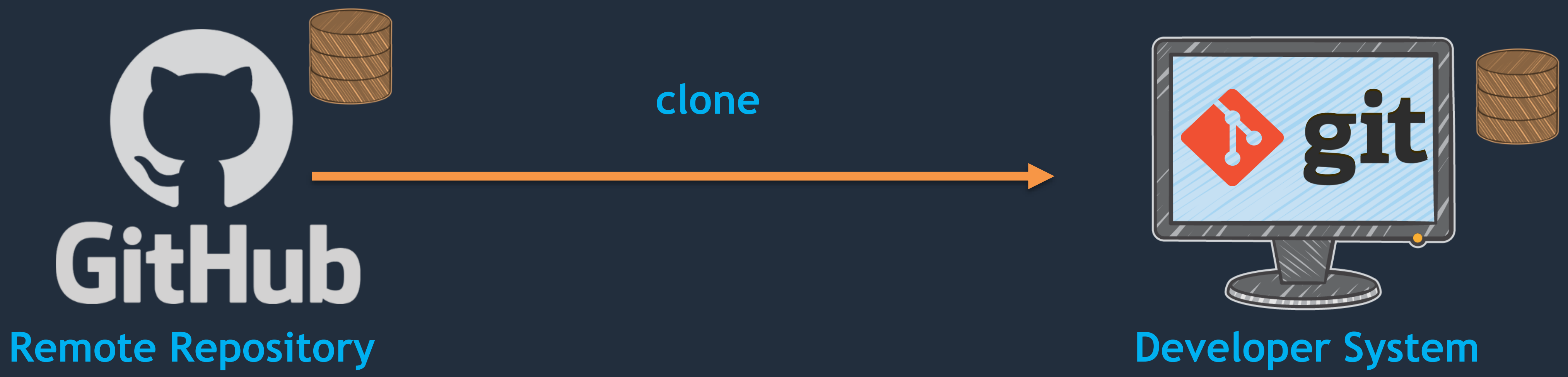
git tags

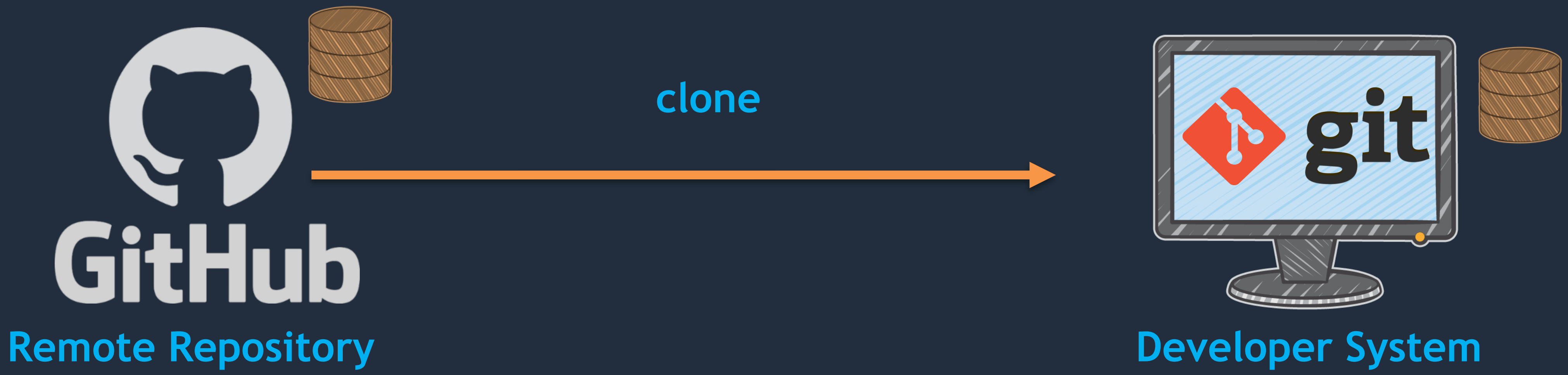
- Tags are ref's that point to specific commit in Git history.
- Tagging is generally used to capture a point in history that is used for a marked version release (i.e. v1.0.1).
- A tag is like a branch that doesn’t change.

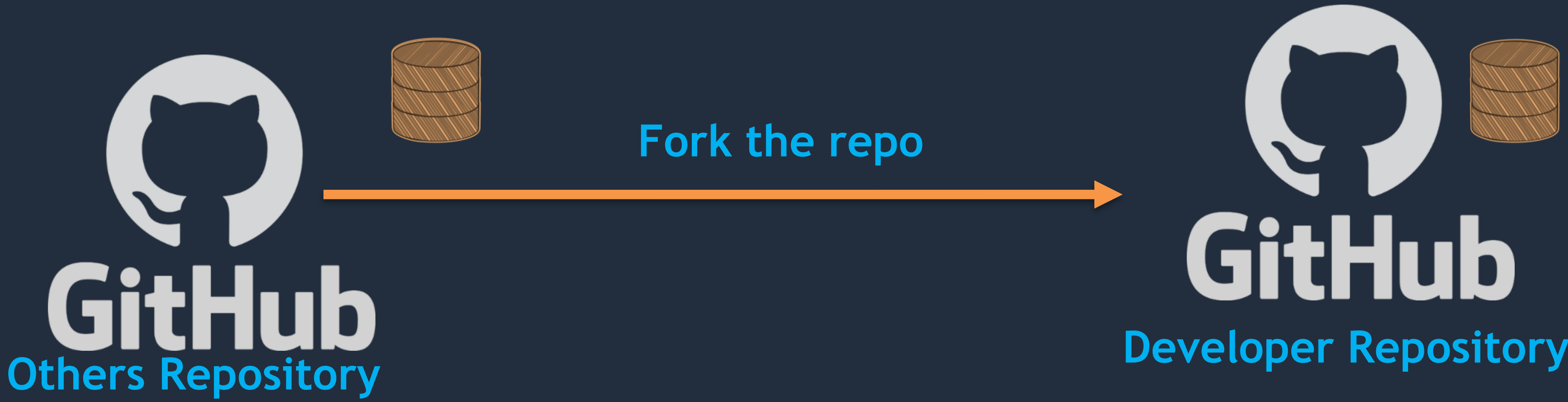
git tags

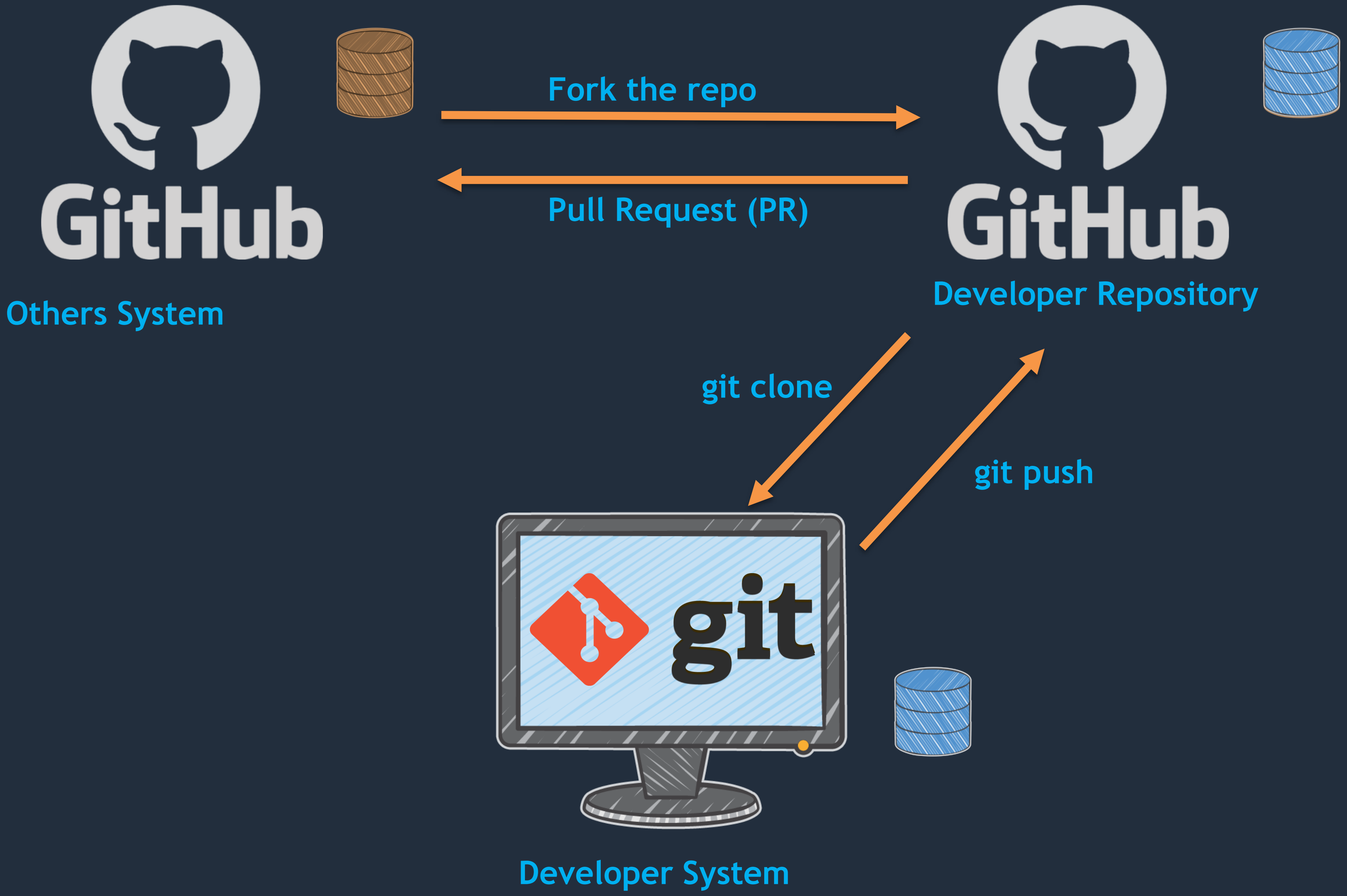
- git init - initializes a brand new Git repository and begins tracking an existing directory.
- git clone - creates a local copy of a project that already exists remotely.
- git add - stages a change.
- git commit - saves the snapshot to the project history and completes the change-tracking process.
- git push - updates the remote repository with any commits made locally to a branch.
- git status - shows the status of changes as untracked, modified, or staged.
- git pull - updates the local line of development with updates from its remote counterpart.
- git branch - shows the branches being worked on locally.
- git merge - merges lines of development together.


GitHub
Others Repository
Fork the repo
GitHub
Developer Repository


GitHub
Remote Repository
clone
git
Developer System

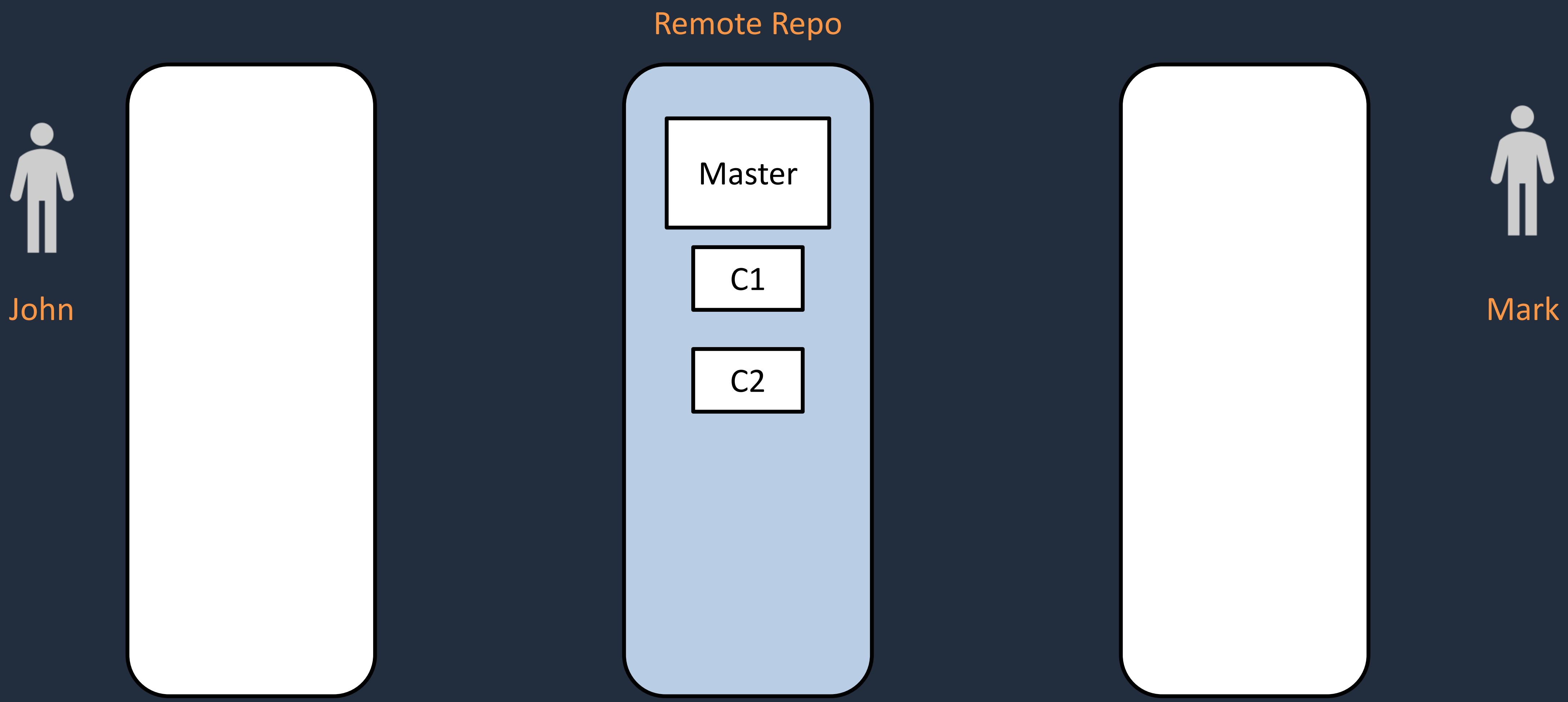

clone
GitHub
Remote Repository
git
Developer System

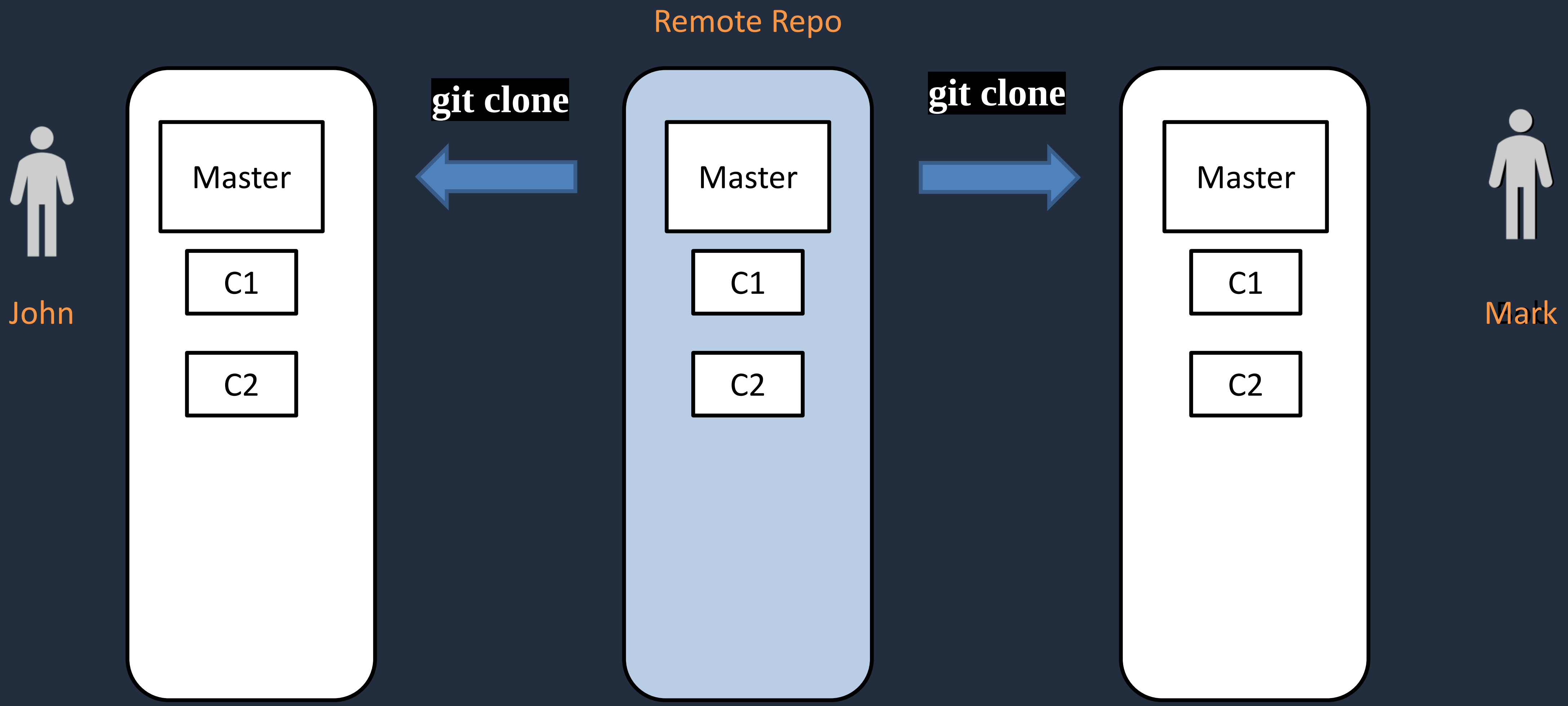

Fork the repo
GitHub
Others Repository
GitHub
Developer Repository

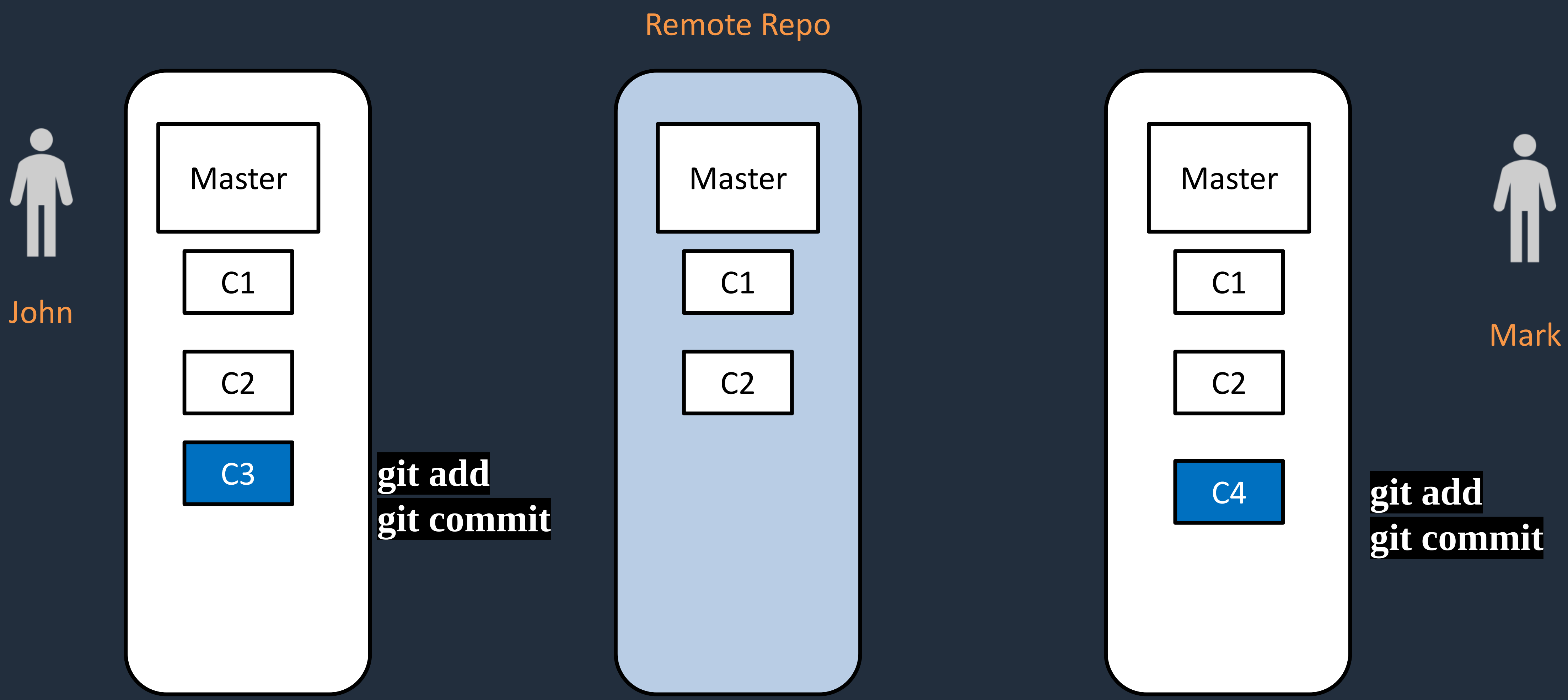

GitHub
Others System
Fork the repo
Pull Request (PR)
GitHub
Developer Repository
git clone
git push
git
Developer System

Basic Git Commands

- git init - initializes a brand new Git repository and begins tracking an existing directory.
- git clone - creates a local copy of a project that already exists remotely.
- git add - stages a change.
- git commit - saves the snapshot to the project history and completes the change-tracking process.
- git push - updates the remote repository with any commits made locally to a branch.
- git status - shows the status of changes as untracked, modified, or staged.
- git pull - updates the local line of development with updates from its remote counterpart.
- git branch - shows the branches being worked on locally.
- git merge - merges lines of development together.

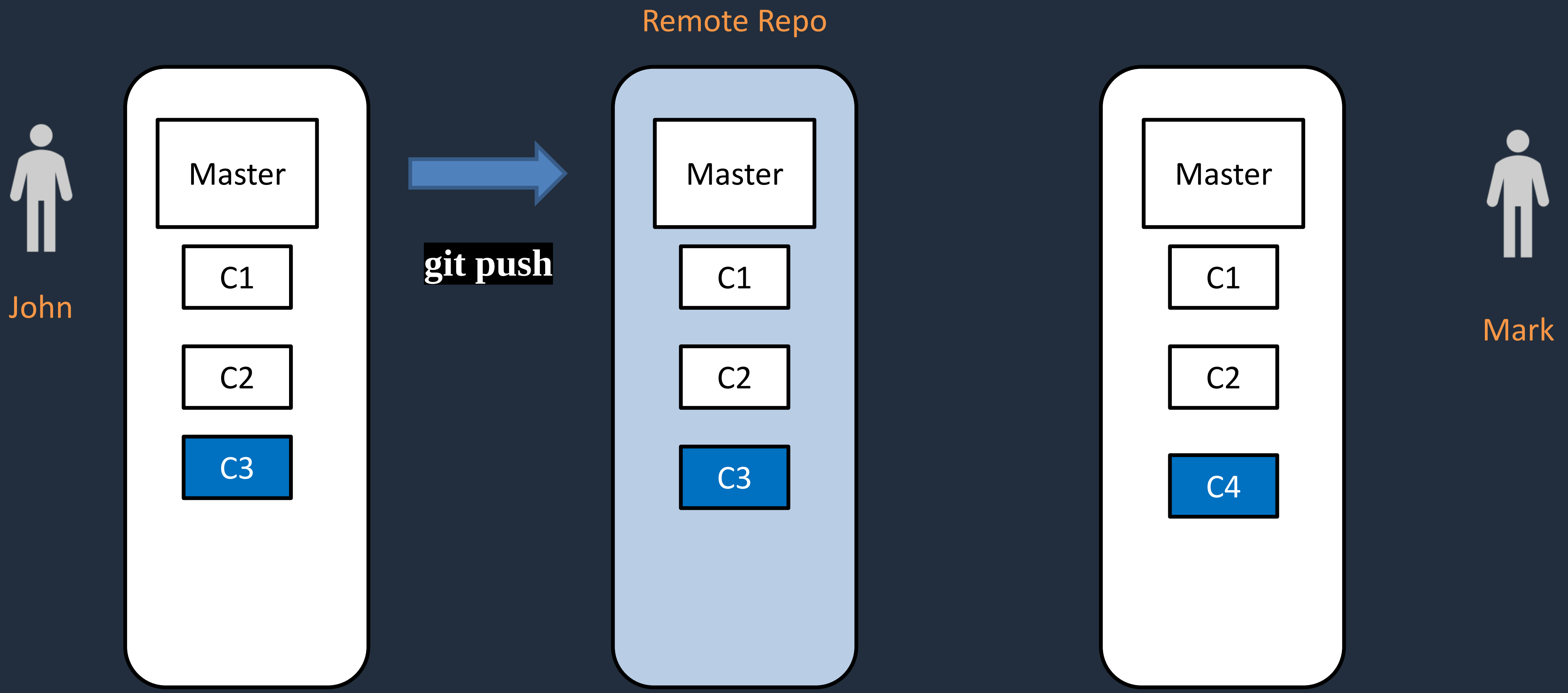

Remote Repo
Master
C1
C2
John
Mark

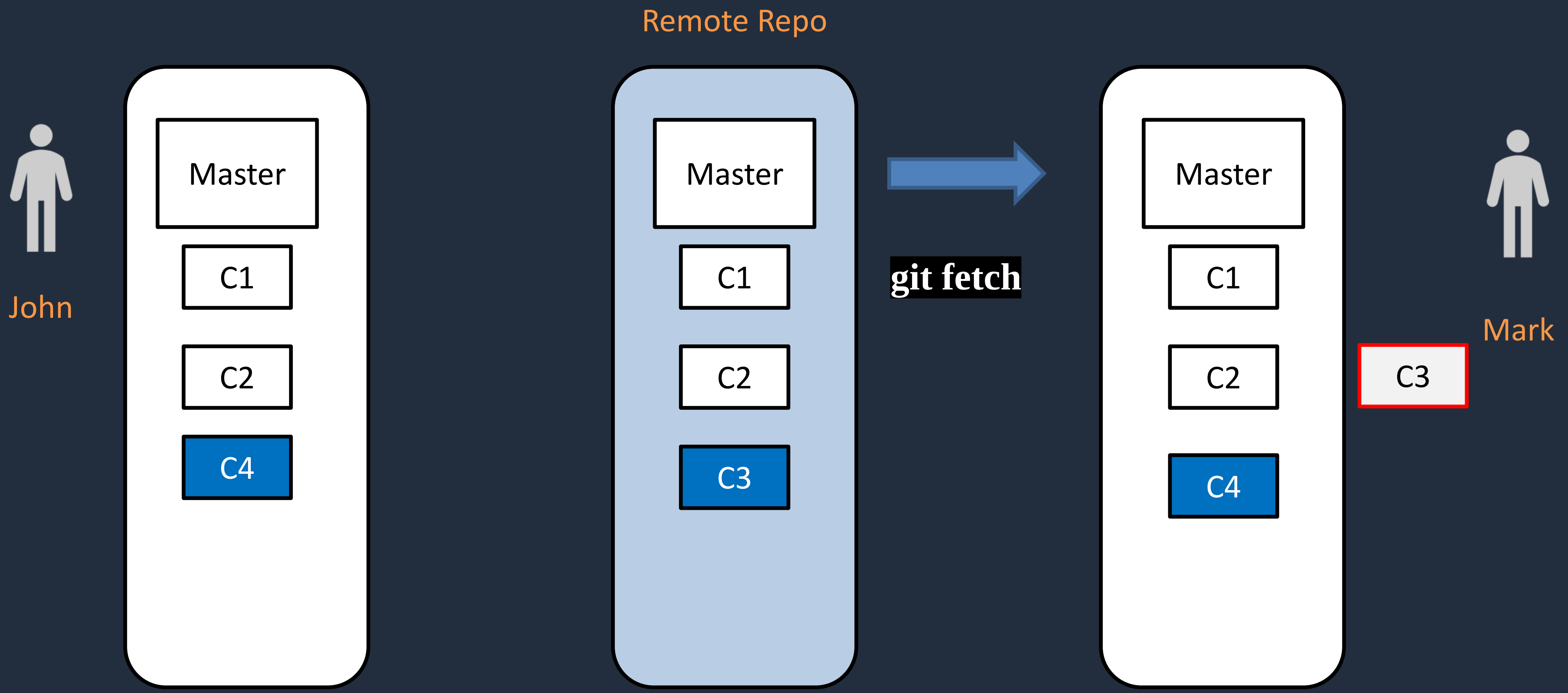

Remote Repo
git clone
git clone
John
Master
C1
C2
Master
C1
C2
Master
C1
C2
Mark

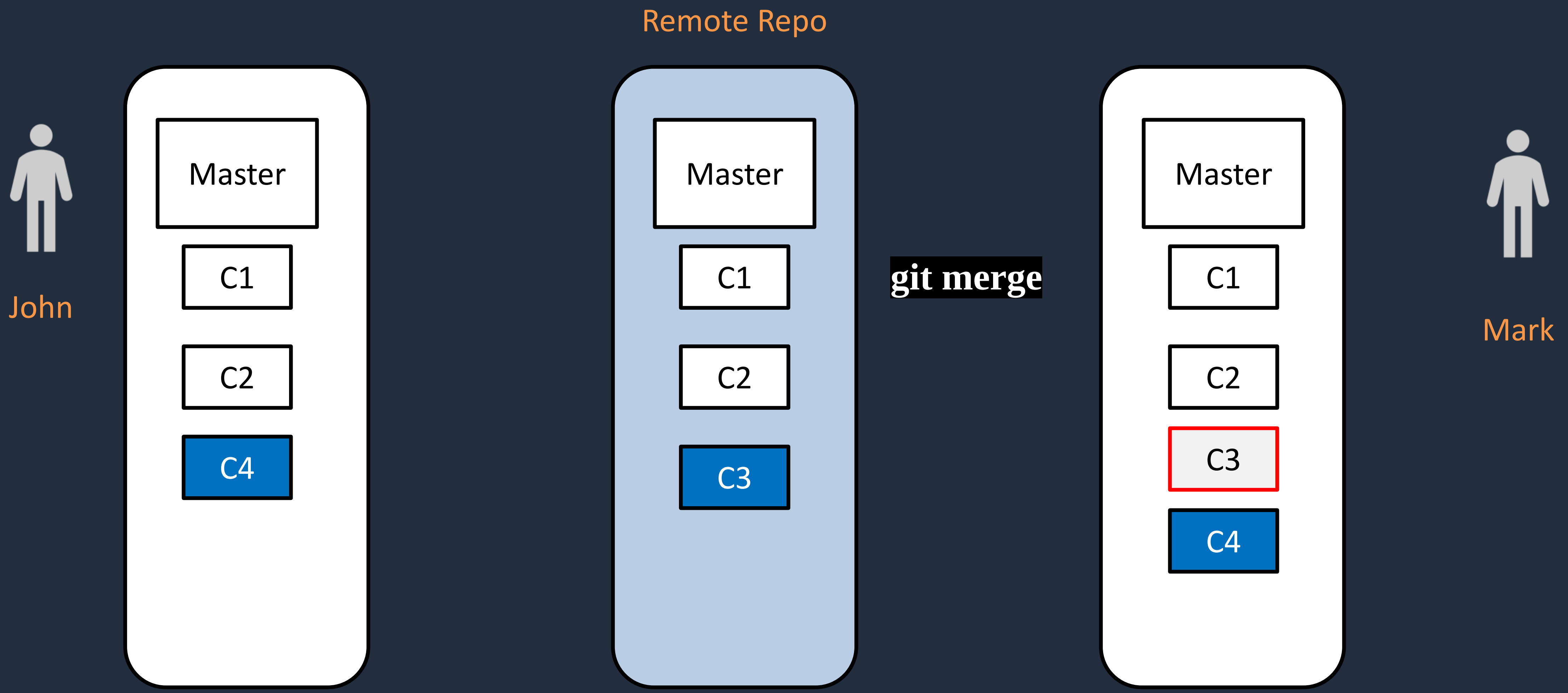


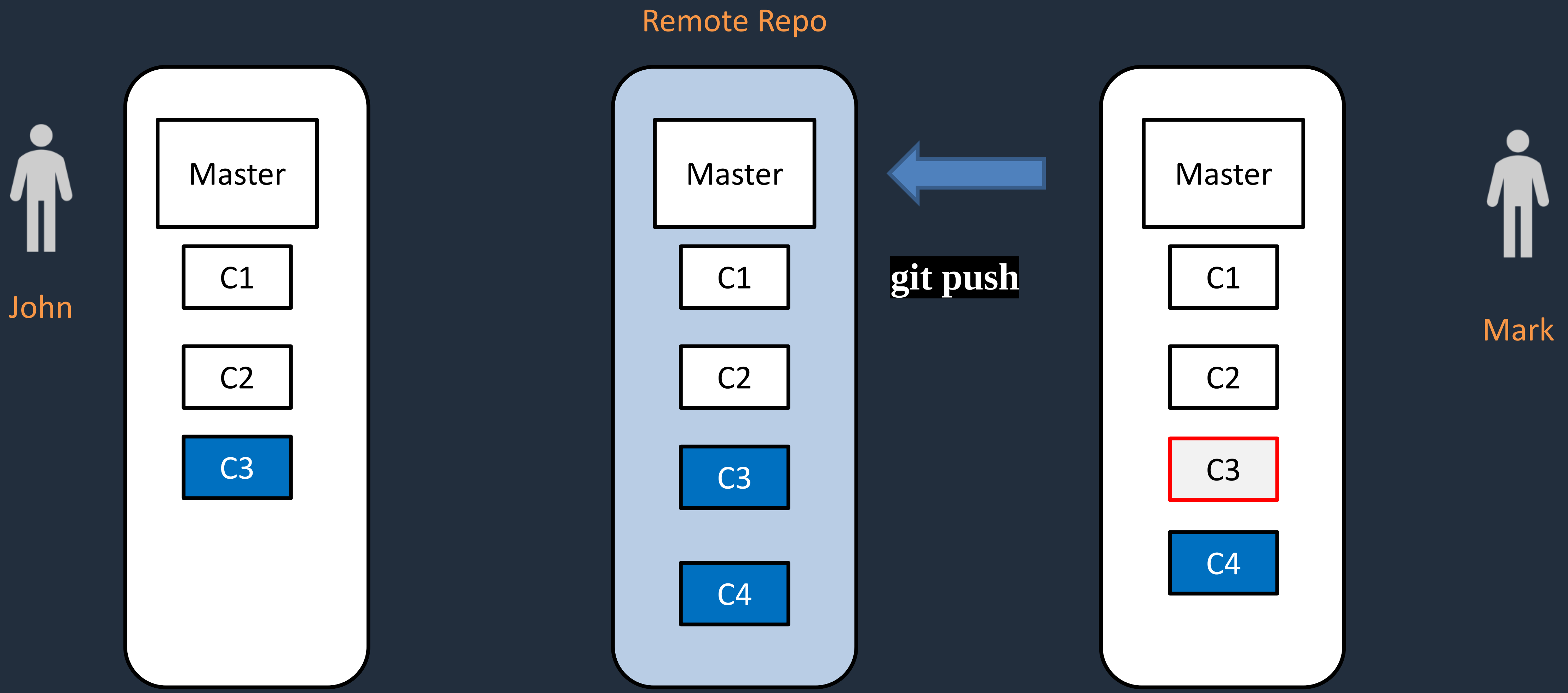
git add
git commit

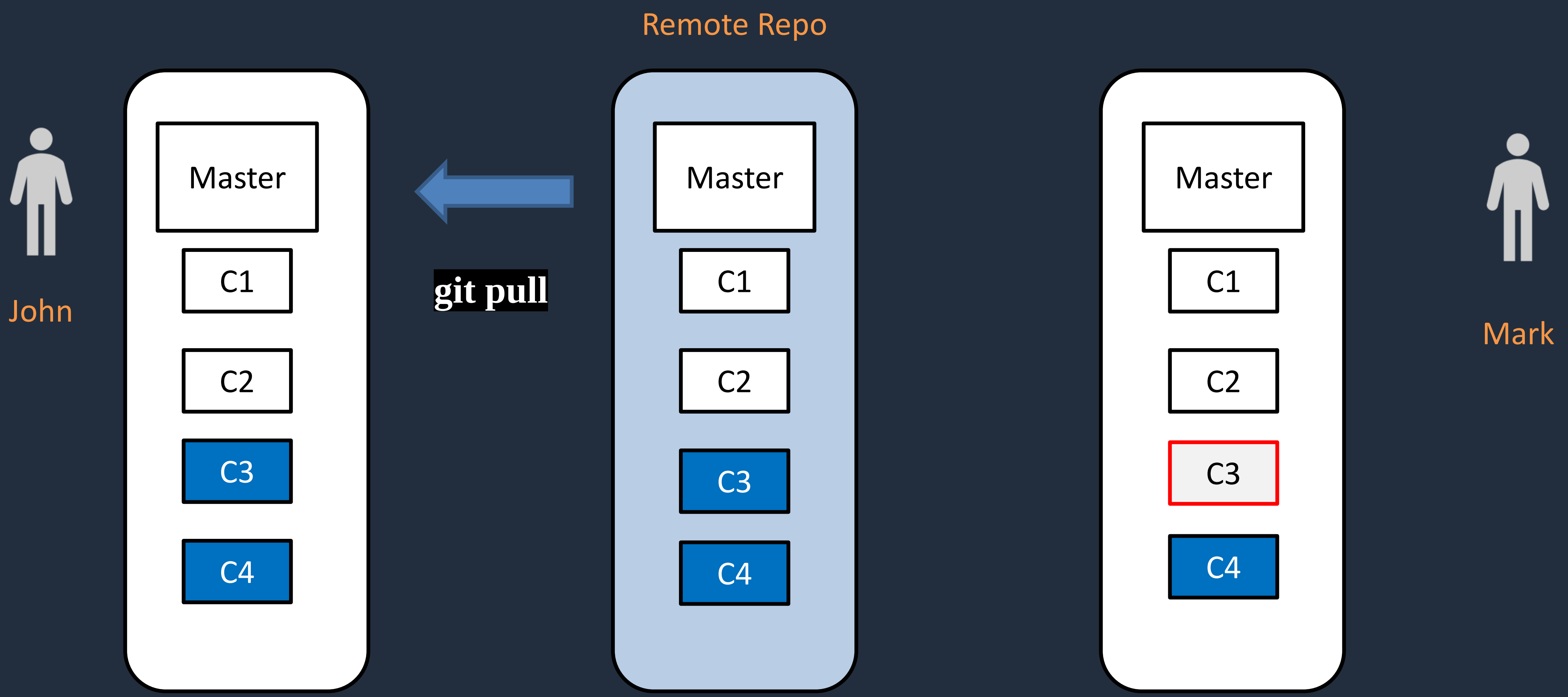
git add
git commit


Remote Repo
John
Master
C1
C2
C3
git push
Master
C1
C2
C3
Master
C1
C2
C4
Mark


Remote Repo
John
Master
C1
C2
C4
Master
C1
C2
C3
git fetch
Master
C1
C2
C4
C3
Mark


Remote Repo
John
Master
C1
C2
C4
Master
C1
C2
C3
git merge
Master
C1
C2
C3
C4
Mark


Remote Repo
John
Master
C1
C2
C3
Master
C1
C2
C3
C4
git push
Master
C1
C2
C3
C4
Mark


Remote Repo
John
Master
C1
C2
C3
C4
git pull
Master
C1
C2
C3
C4
Master
C1
C2
C3
C4
Mark

DevOps Engineer roles on git

DevOps Engineer, Can you create a repository for new project

Repository setup in the real-world

- ✓ Create a private repository
- ✓ Create 3 branches (Prod, UAT and Dev)
- ✓ Add team as collaborators to this repository.
- ✓ Enable SSH based authentication
- ✓ Protect Prod and UAT branches
- ✓ 1 approval needed to check-in code on UAT and 2 approvals needed to check-in code on to Prod
- ✓ Build and Deploy should be successful before check-in the code onto UAT as well as onto Prod

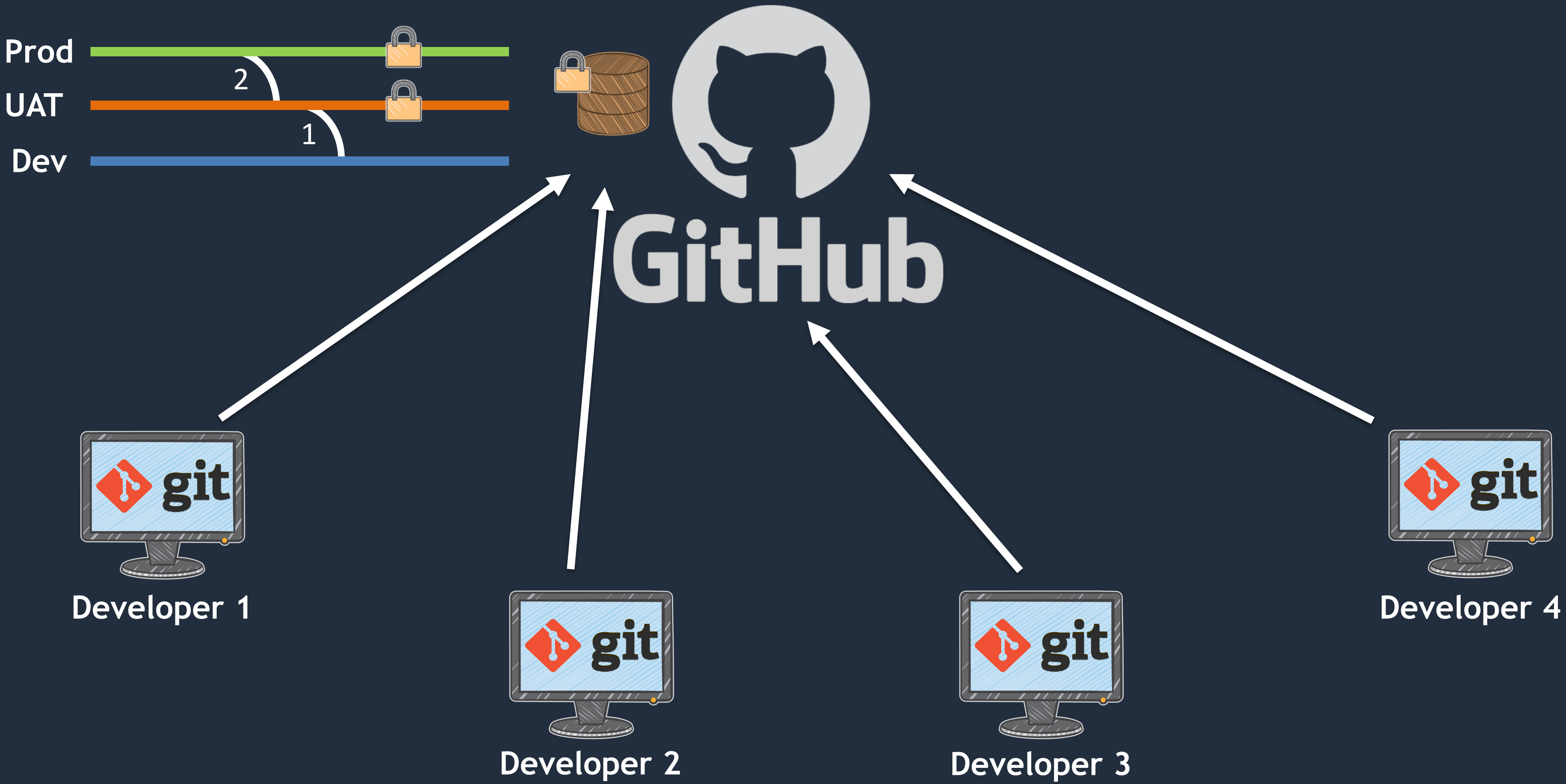
Project Modules

Developer 1 - Home Page

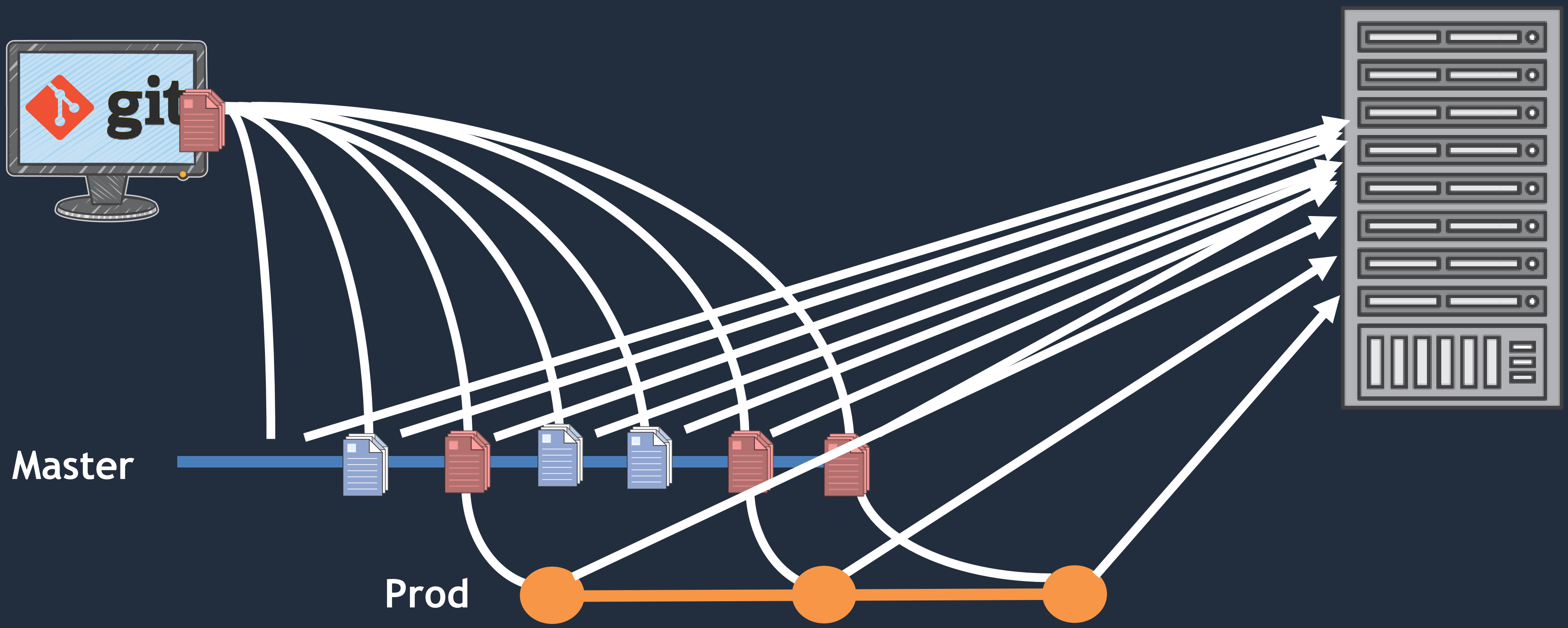
Developer 2 – Services

Developer 3 – Cars, drivers and Garage

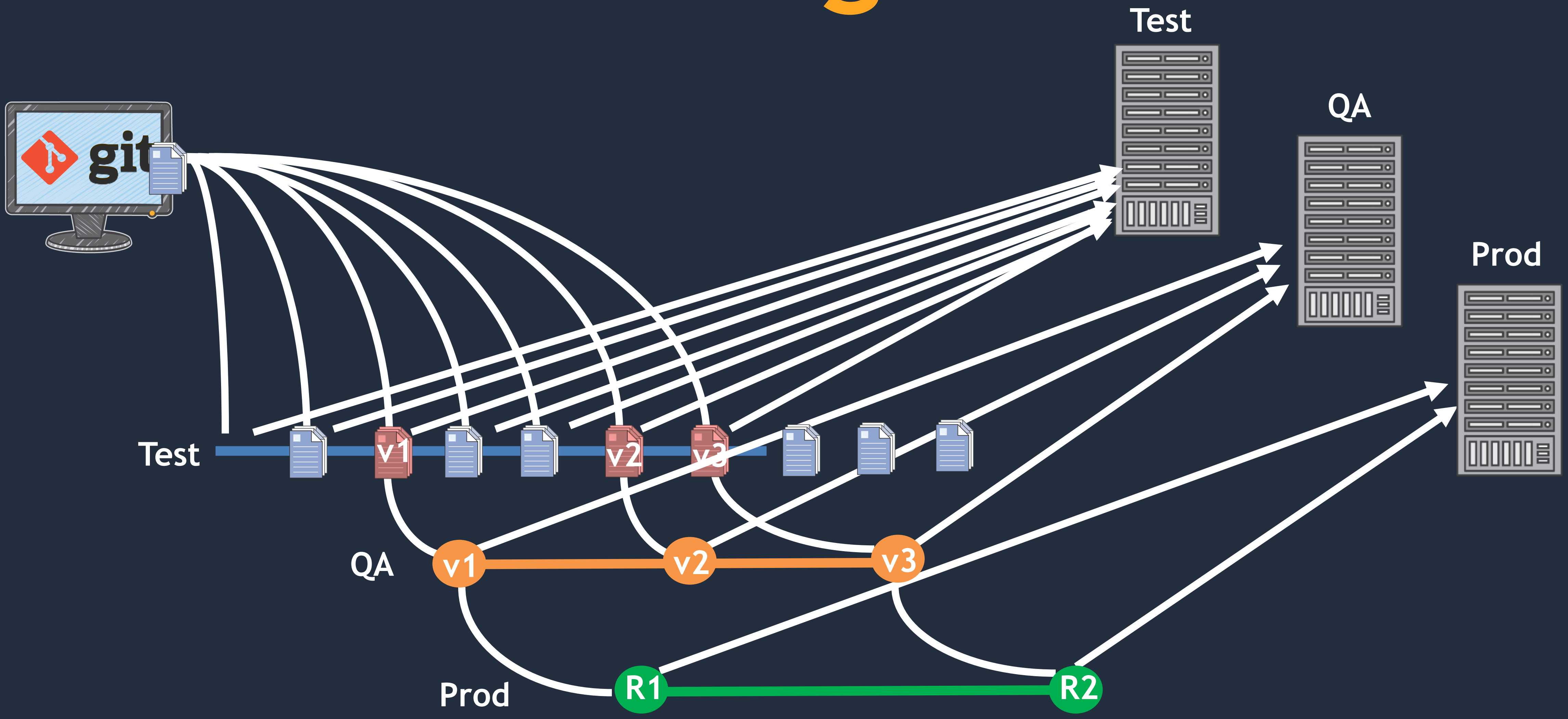
Developer 4 – Locations and contact us


Prod
2
UAT
1
Dev
GitHub
git
Developer 1
git
Developer 2
git
Developer 3
git
Developer 4

Branches



Branches Strategies



Thank You for watching this video

For help: Connect us using below mediums

twitter.com/valaxytechnologies

facebook.com/valaxytech

youtube.com/c/valaxytechnologies