GIT FUNDAMENTALS

What and Why GIT?

- Distributed Version Control System to track versions of files.
- Enables multiple developers to work on the same project.
- Maintains a history of changes.
- Allows branching and merging for feature development.

What is GIT?

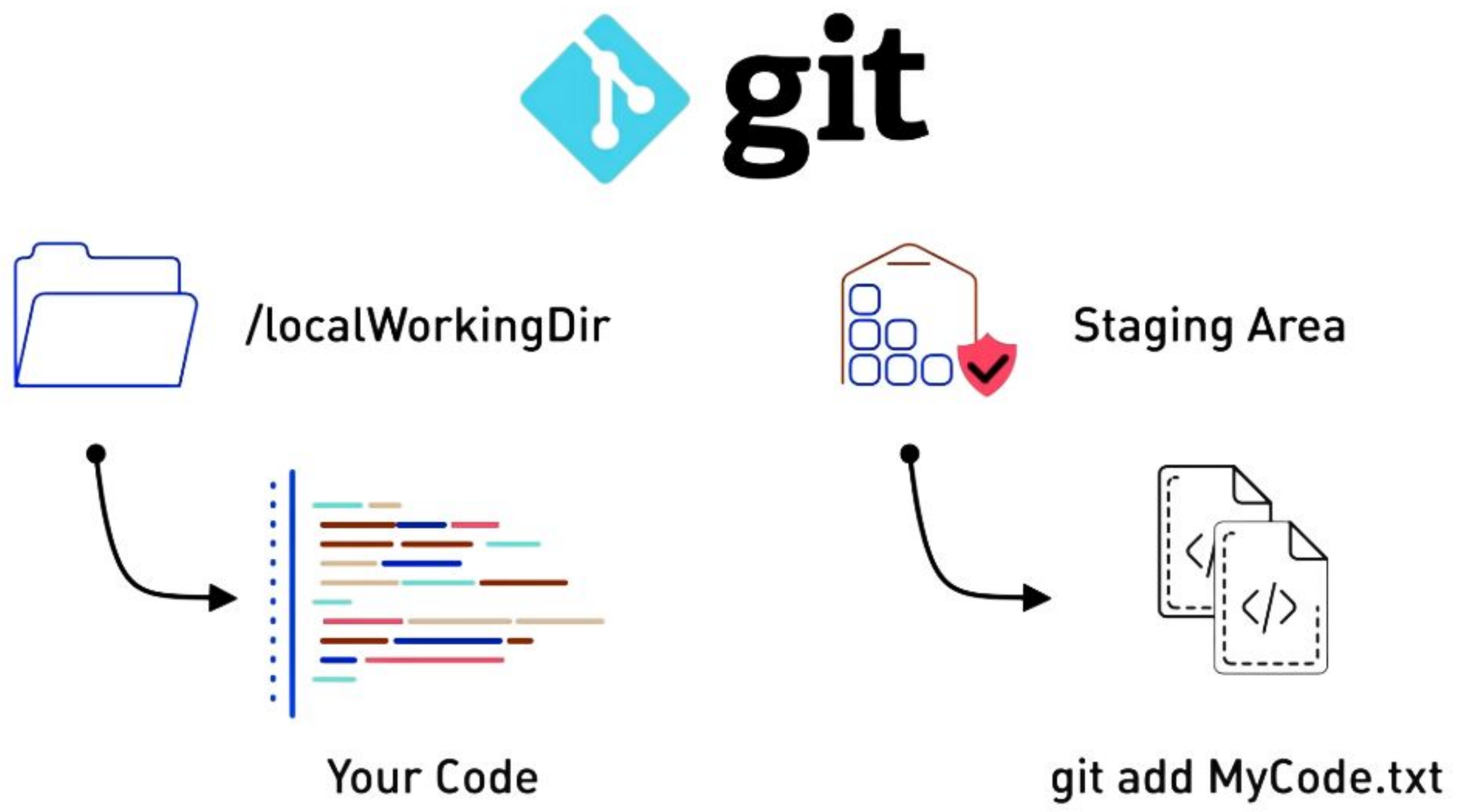


Image source: https://www.youtube.com/watch?v=e9lnsKot_SQ&ab_channel=ByteByteGo

What is GIT?

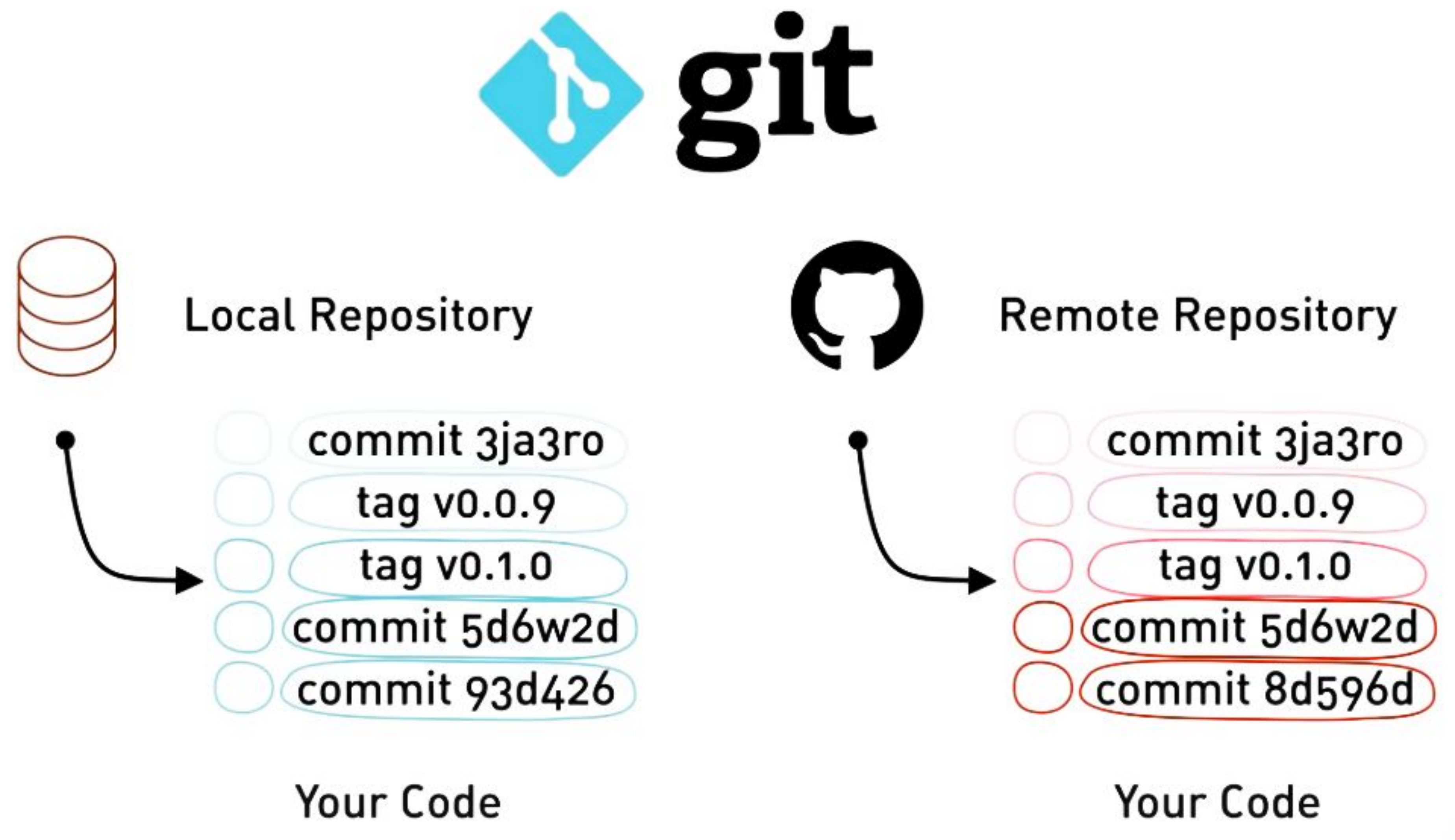


Image source: https://www.youtube.com/watch?v=e9lnsKot_SQ&ab_channel=ByteByteGo

GIT?

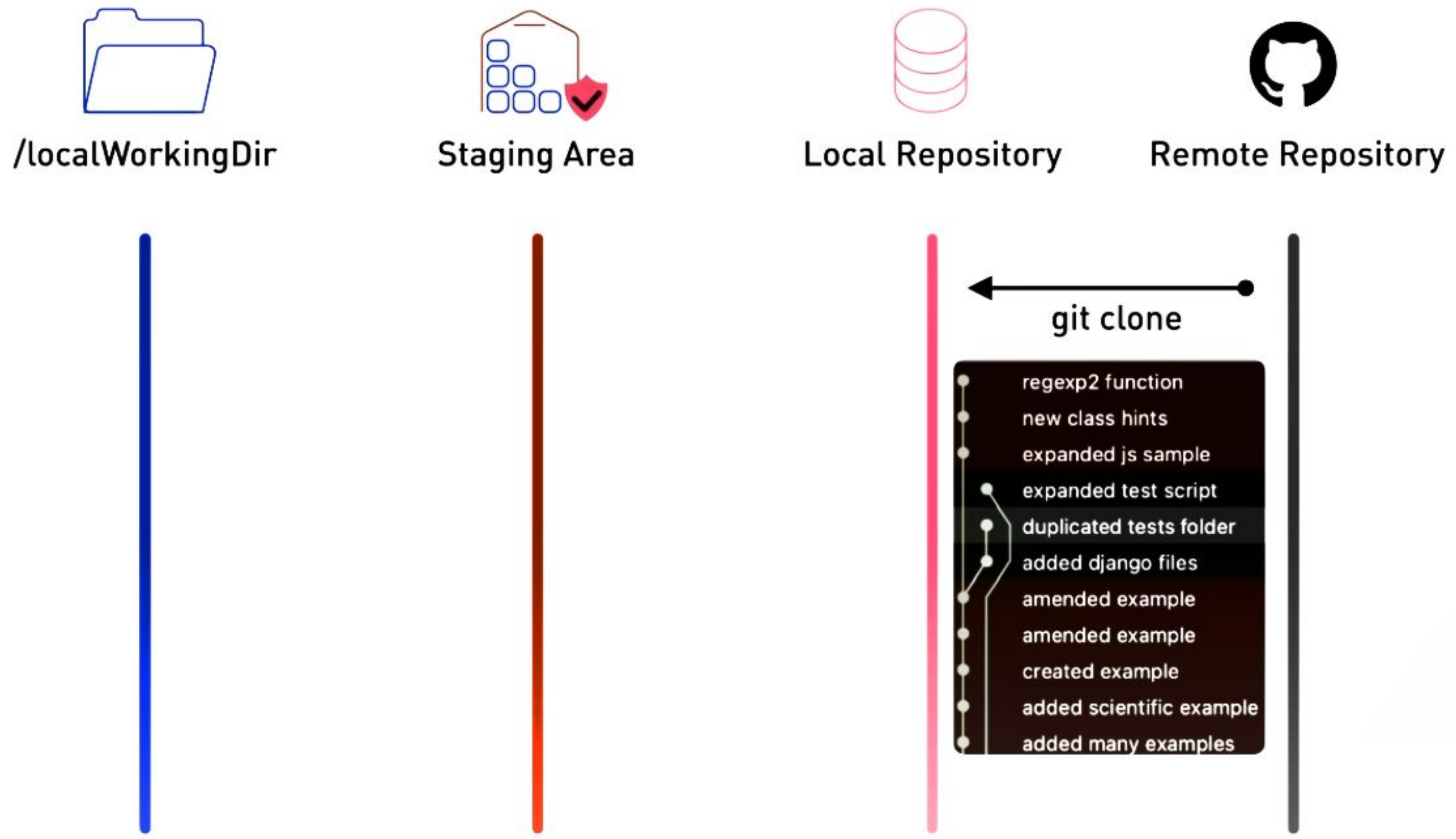


Image source: https://www.youtube.com/watch?v=e9lnsKot_SQ&ab_channel=ByteByteGo

GIT?

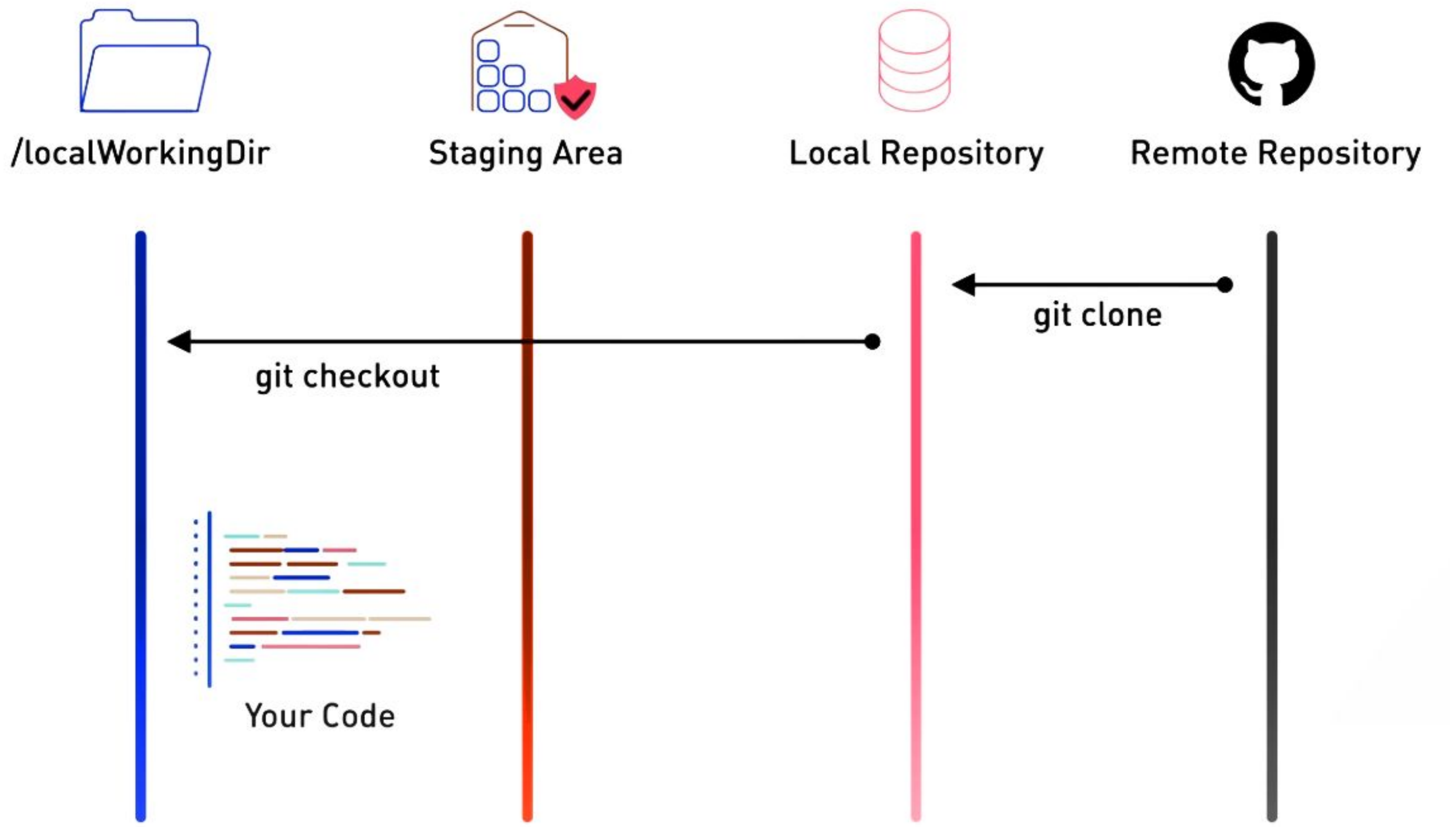


Image source: https://www.youtube.com/watch?v=e9lnsKot_SQ&ab_channel=ByteByteGo

GIT?

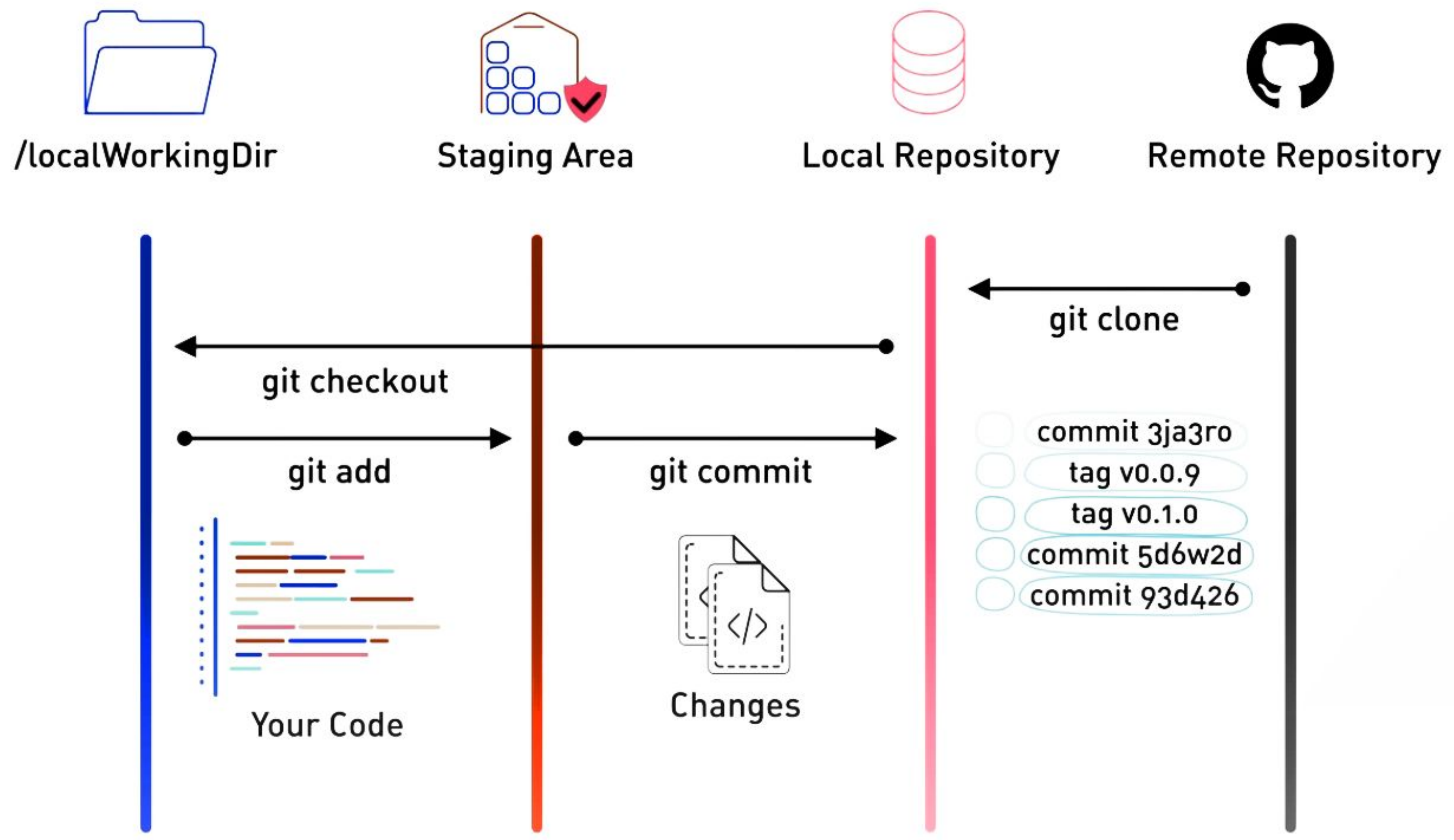


Image source: https://www.youtube.com/watch?v=e9lnsKot_SQ&ab_channel=ByteByteGo

GIT?

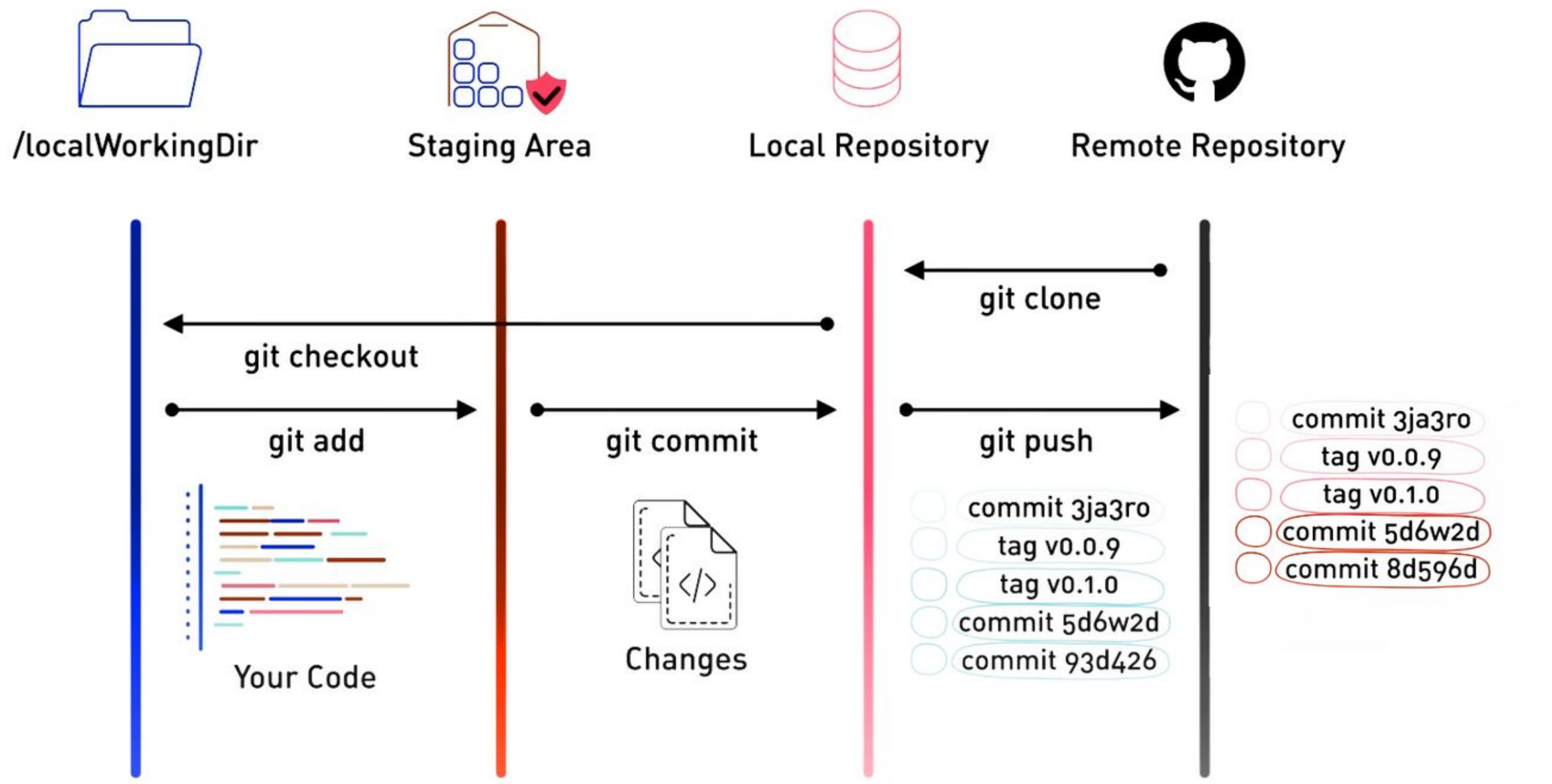


Image source: https://www.youtube.com/watch?v=e9lnsKot_SQ&ab_channel=ByteByteGo

GIT?

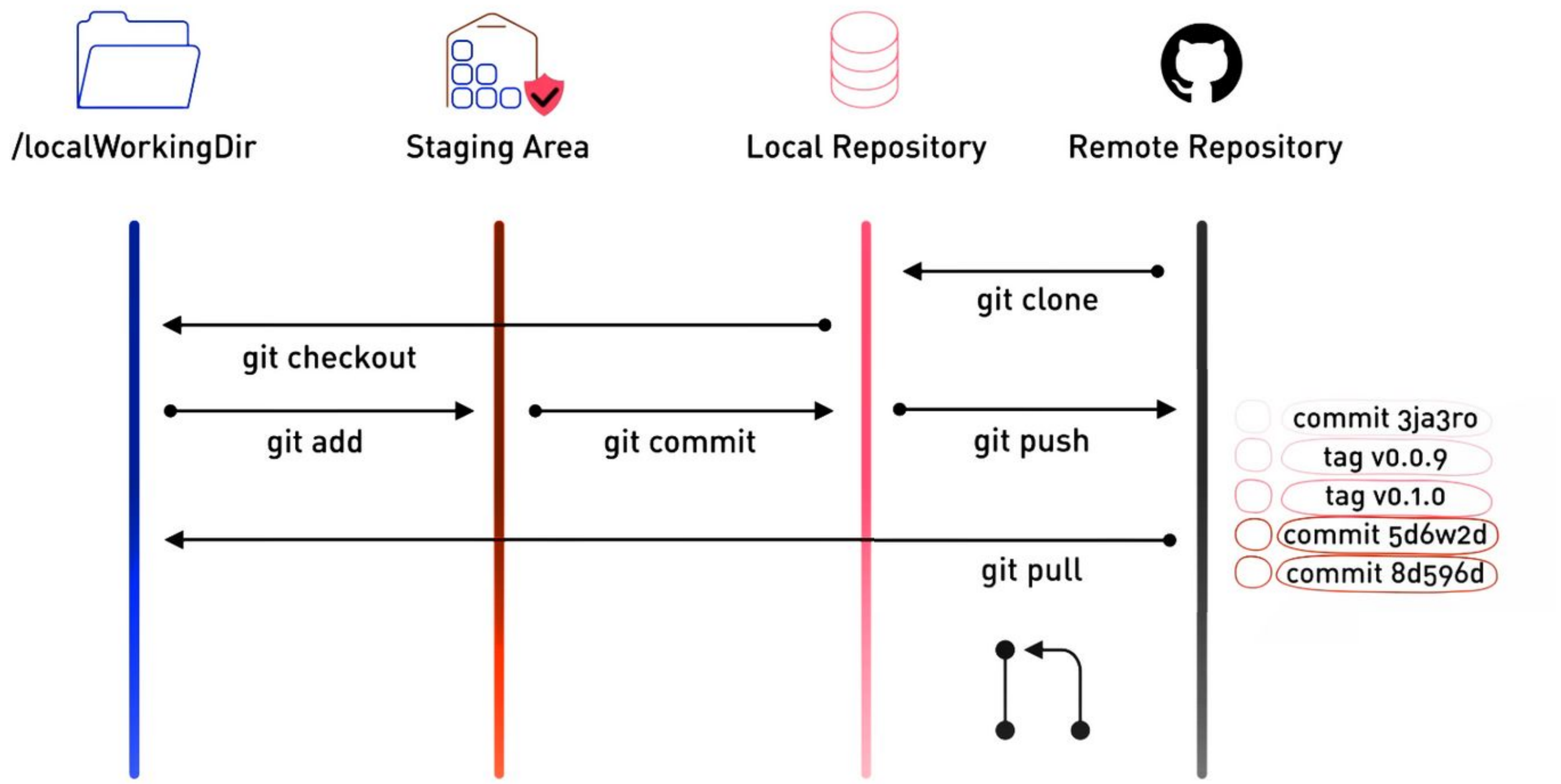


Image source: https://www.youtube.com/watch?v=e9lnsKot_SQ&ab_channel=ByteByteGo

GitHub

WHAT IS GITHUB?

GitHub

- A web-based platform for version control using Git.
- Provides cloud-based storage for Git repositories Facilitates collaboration with features like pull requests, issues, and wikis.
- Getting Started:
  - Create an account at github.com
  - Explore public repositories or create your own repository.

WHY USE GIT & GITHUB?

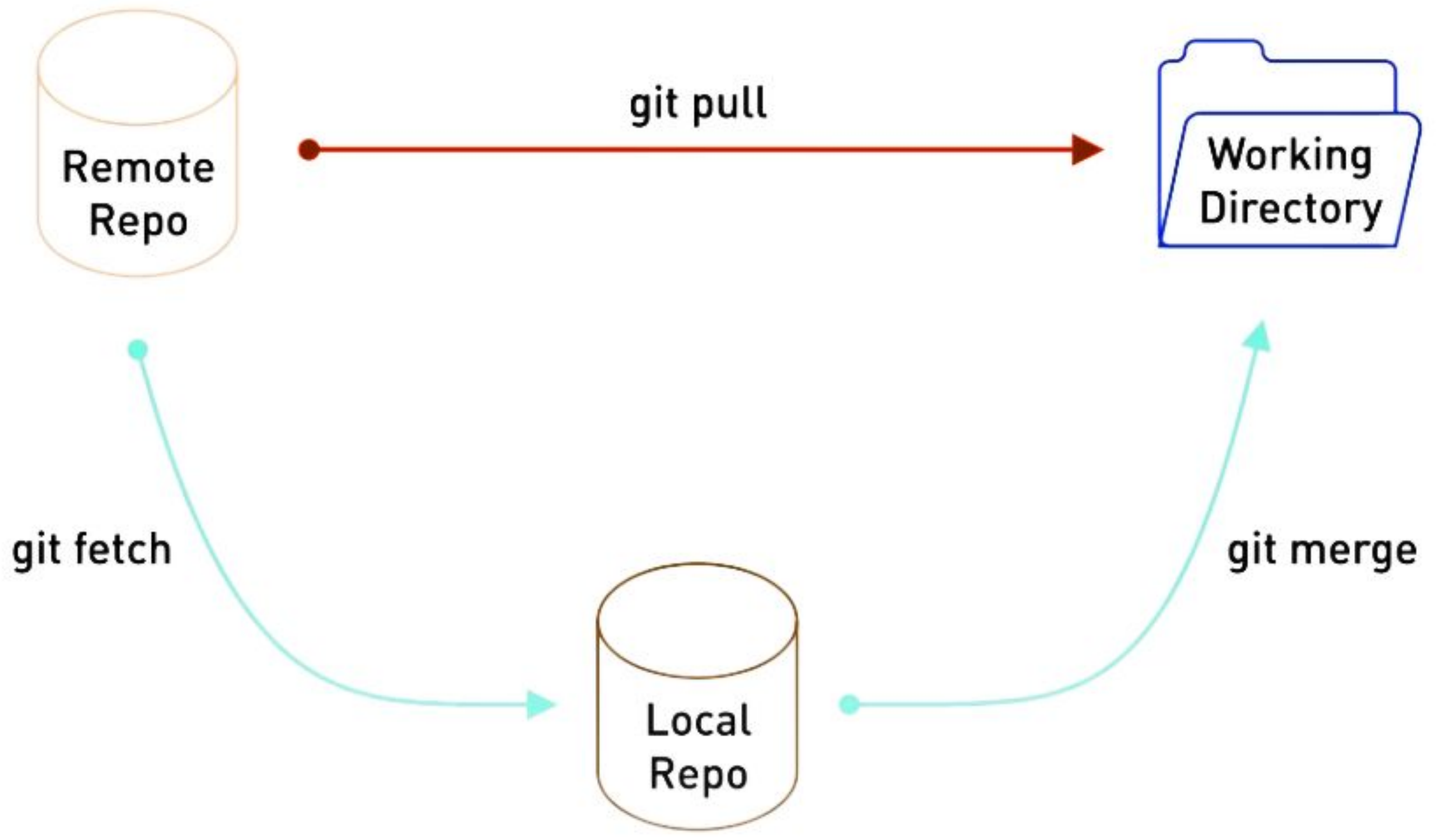


Image source: https://www.youtube.com/watch?v=e9lnsKot_SQ&ab_channel=ByteByteGo

INSTALLATION

Installing Git on Different Platforms:

- Windows: Download from git-scm.com. Use Git Bash or Git GUI for operations.
- Mac: Use Homebrew: brew install git
- Linux: Use package manager: sudo apt-get install git (Debian/Ubuntu) or sudo dnf install git (Fedora)

Verify Installation:

- Run git --version in the terminal to ensure Git is installed.

</>
SSH

SSH KEYS

- What are SSH Keys?
  - Secure Shell (SSH) keys are a pair of cryptographic keys used for secure communication
  - Allows secure connection to remote repositories without entering passwords
- Key Components:
  - Public Key: Shared with the server Private
  - Key: Kept secure on your machine

SETTING UP SSH KEYS ON MAC AND LINUX

- **Generate SSH Keys:**
  - Run `ssh-keygen -t ed25519 -C "your_email@example.com"`
  - Default location: ~/.ssh/id_ed25519
- **Add SSH Key to SSH Agent:**
  - Start the agent: `eval "$(ssh-agent -s)"`
  - Add key: `ssh-add ~/.ssh/id_ed25519`
- **Add SSH Key to GitHub:**
  - Copy public key: `cat ~/.ssh/id_ed25519.pub` Add key in
  - GitHub under Settings > SSH and GPG keys

SETTING UP SSH KEYS ON WINDOWS

- Generate SSH Keys:
  - Use Git Bash to run ssh-keygen -t ed25519 -C "your_email@example.com"
  - Default location: C:\Users\YourName\.ssh\id_ed25519
- Add SSH Key to SSH Agent:
  - Start the agent: eval "$(ssh-agent -s)"
  - Add key: ssh-add ~/.ssh/id_ed25519
- Add SSH Key to GitHub:
  - Copy public key: cat ~/.ssh/id_ed25519.pub or use a text editor
  - Add key in GitHub under Settings > SSH and GPG keys

CREATING A REPOSITORY

- Creating a New Repository on GitHub:
    - Navigate to GitHub and click New repository
    - Initialize with a README file or add .gitignore
- Creating a Local Repository:
    - Use `git init` in your project directory. This
    - creates a .git folder to track changes

CLONING A REPOSITORY

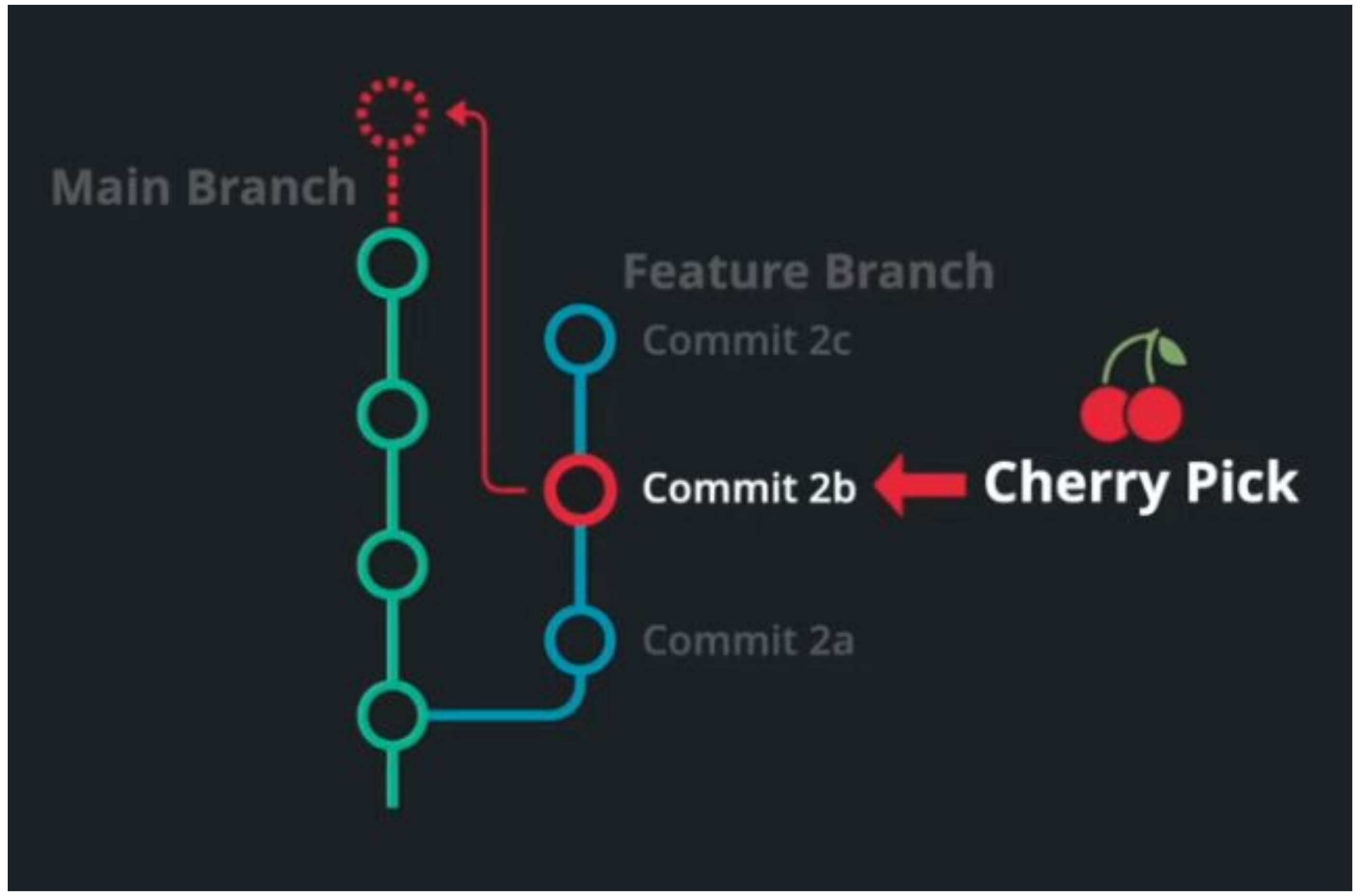
- What is Cloning?
  - Copying a repository from GitHub (or another Githost) to your local machine
- How to Clone a Repository:
  - Use the command: git clone <repository-url>
  - Example: git clone https://github.com/user/repo.git
  - Cloning creates a local copy with all the history and branches

ADDING CODE TO REPOSITORY

- Checking Status:
  - Use *git status* to see changes in the working directory
- Adding Changes:
  - Stage changes with *git add <file>* or *git add .* to add all changes
  - Committing Changes: Use *git commit -m "commit message"* to
  - save changes
- Pushing to GitHub:
  - Use *git push origin main* (or the current branch name) to update the remote repository

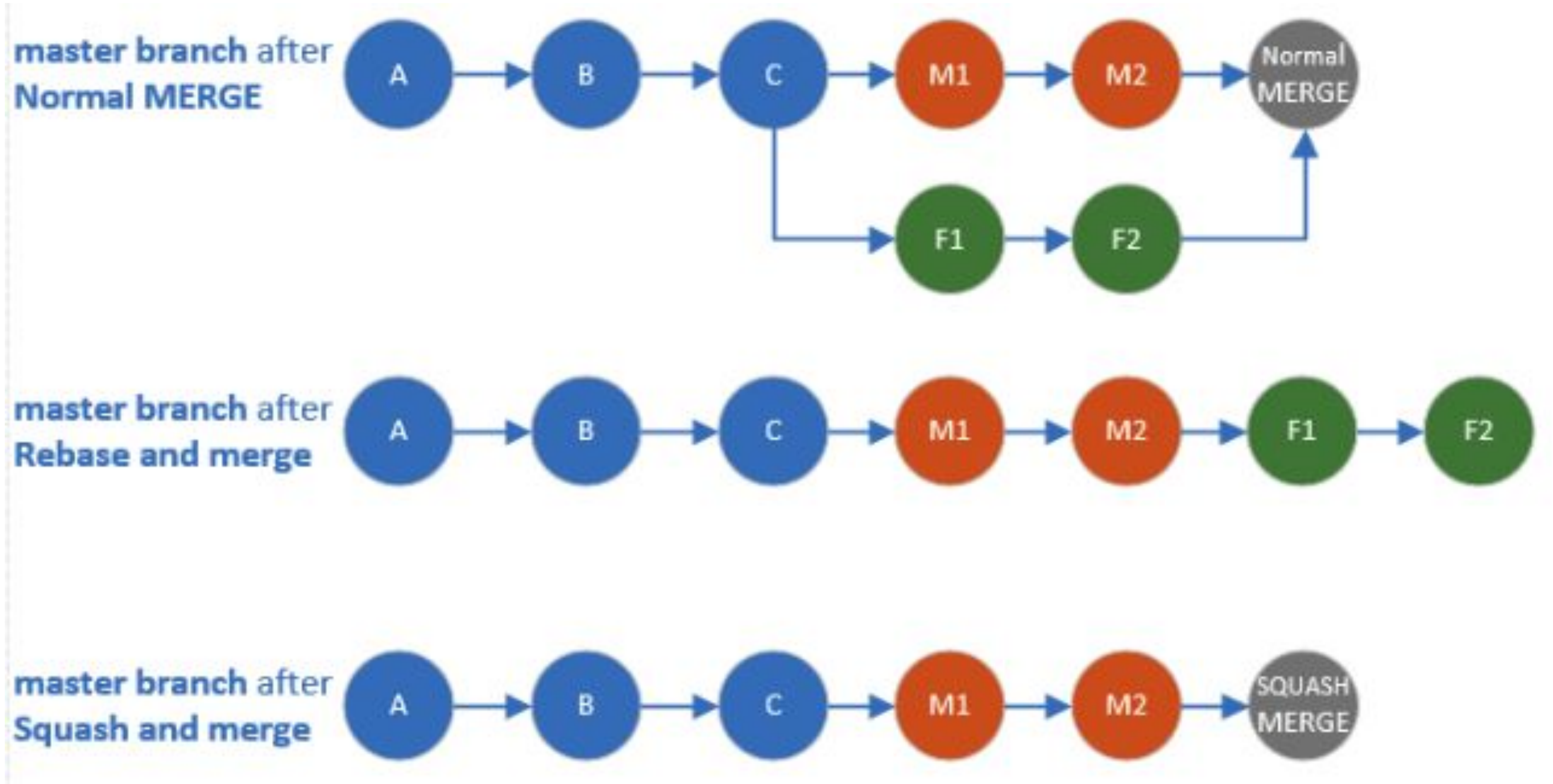
SOME GIT COMMANDS

- Git Branch
  - Create, list, or switch between different development branches
  - Command: `git branch branch-name`
- Git Cherry-Pick
  - Apply a specific commit from another branch to your current branch
  - Command: `git cherry-pick <commit-hash>`
- Git Diff
  - Show differences between commits, branches, or working directory
  - Command: `git diff HEAD~1`   # compare with previous commit
  - Command: `git diff branch1 branch2` # compare two branches
- Git Amend
  - Modify the most recent commit (message or content)
  - Command: `git commit --amend`



SOME GIT COMMANDS

- Git Stash
  - Temporarily save uncommitted changes.
  - Command: `git stash`
- Git Patch
  - Create and apply patch files containing changes that can be shared or applied elsewhere
  - Command: `git format-patch HEAD~3`   # create patch files for last 3 commits
  - Command: `git apply patch-file.patch`   # apply a patch file
- Git Reset
  - Undo commits by moving HEAD pointer backwards
  - Command: `git reset --soft HEAD~1` # keep changes staged
  - Command: `git reset --hard HEAD~1` # discard all changes
- Git Reflog
  - View complete history of HEAD movements, including "lost" commits
  - Command: `git reflog`
- Git Merge vs Git Rebase
  - Merge creates a merge commit vs Rebase replays commits linearly
  - Command: `git merge feature-branch` # creates merge commit
  - Command: `git rebase main`   # replays commits on top of main


master branch after Normal MERGE
A
B
C
M1
M2
Normal MERGE
F1
F2
master branch after Rebase and merge
A
B
C
M1
M2
F1
F2
master branch after Squash and merge
A
B
C
M1
M2
SQUASH MERGE