Version Control

We will learn

- Version Control – Concepts, Terminology
- Why use Version Control
- GitHub
- GitLab
- Apache SVN

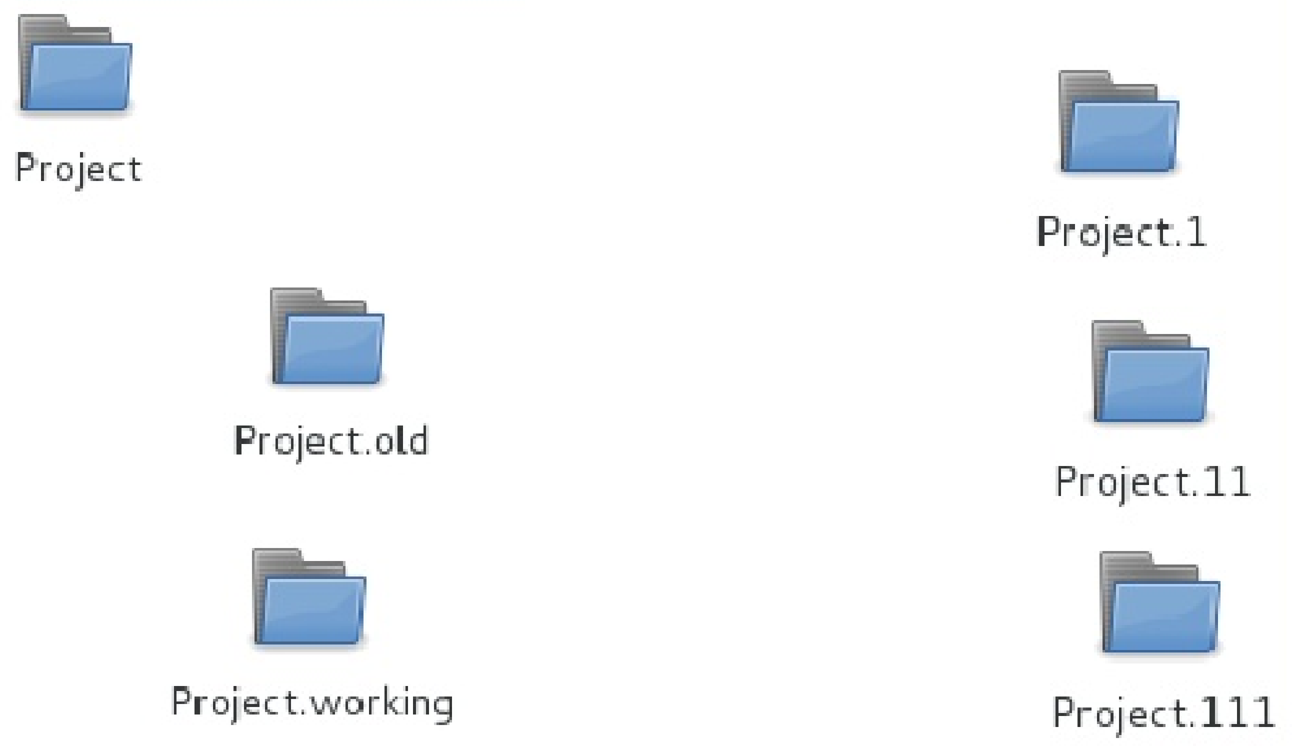
Version Control

- Manage changes to files, documents, images and source code
- Also Known as :
    - Revision Control
    - Source Control
    - Version Management
    - Configuration Management

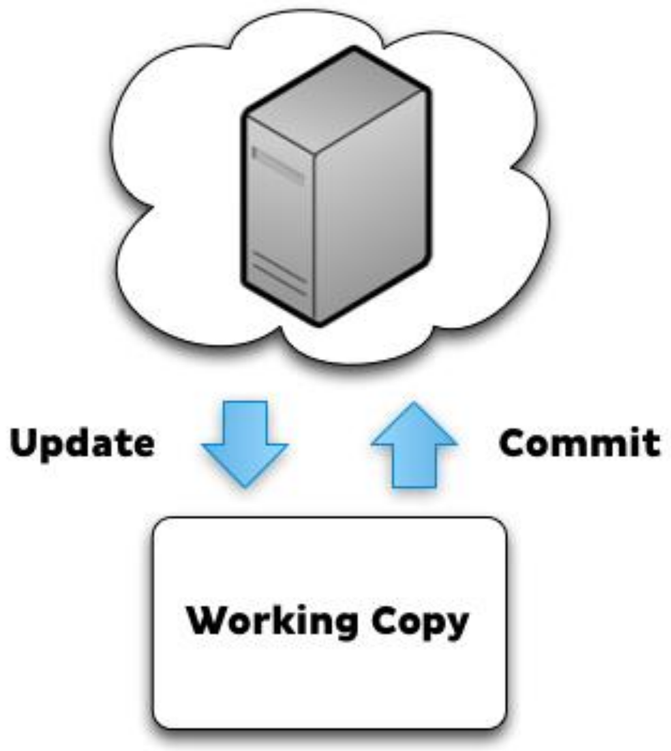
Why Version Control

- Undo, incremental backup of changes
- Integrate several sources or subsystems or source code together
- Team Collaboration
  - Code without interference
  - Integrate code of Multiple developers
  - Know Who Did, What and When
  - Time Travel – Go back to any previous version
- Keep your source code secure
- Troubleshooting
- Productivity

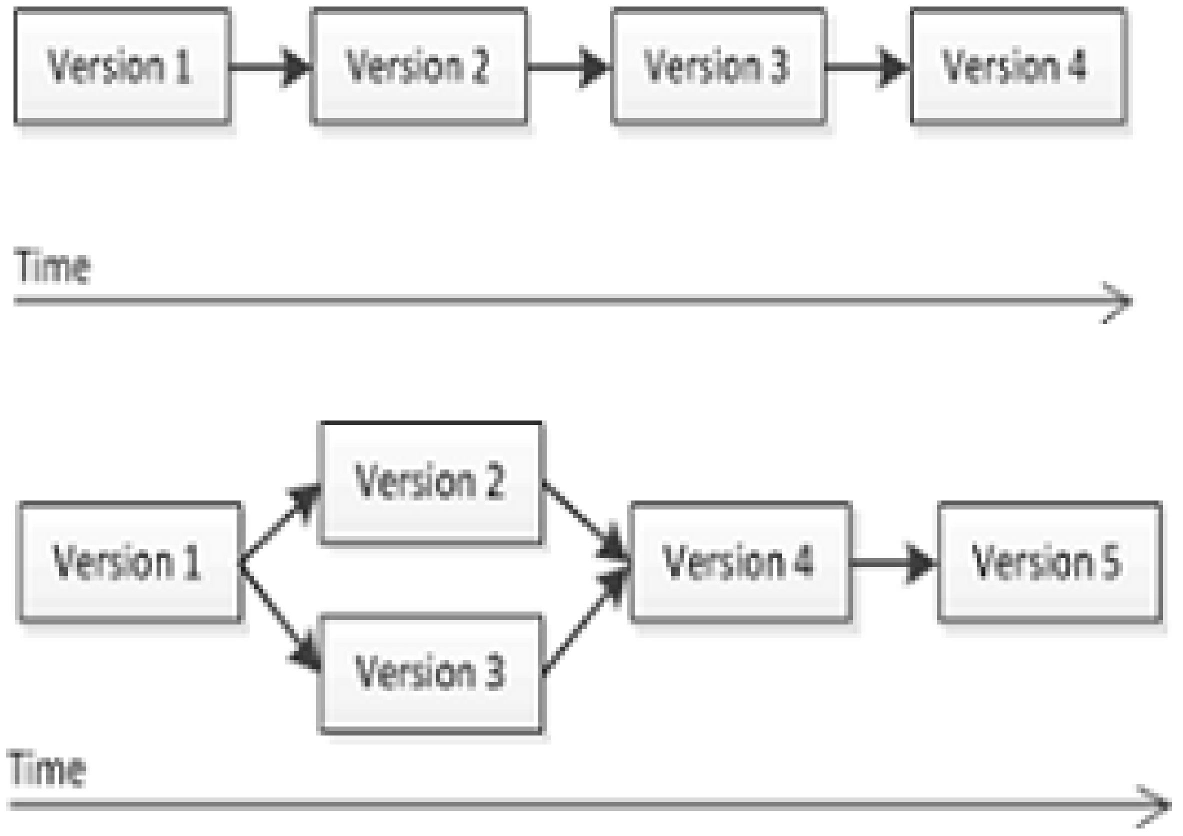
Have you or your team created folder like this for same project?



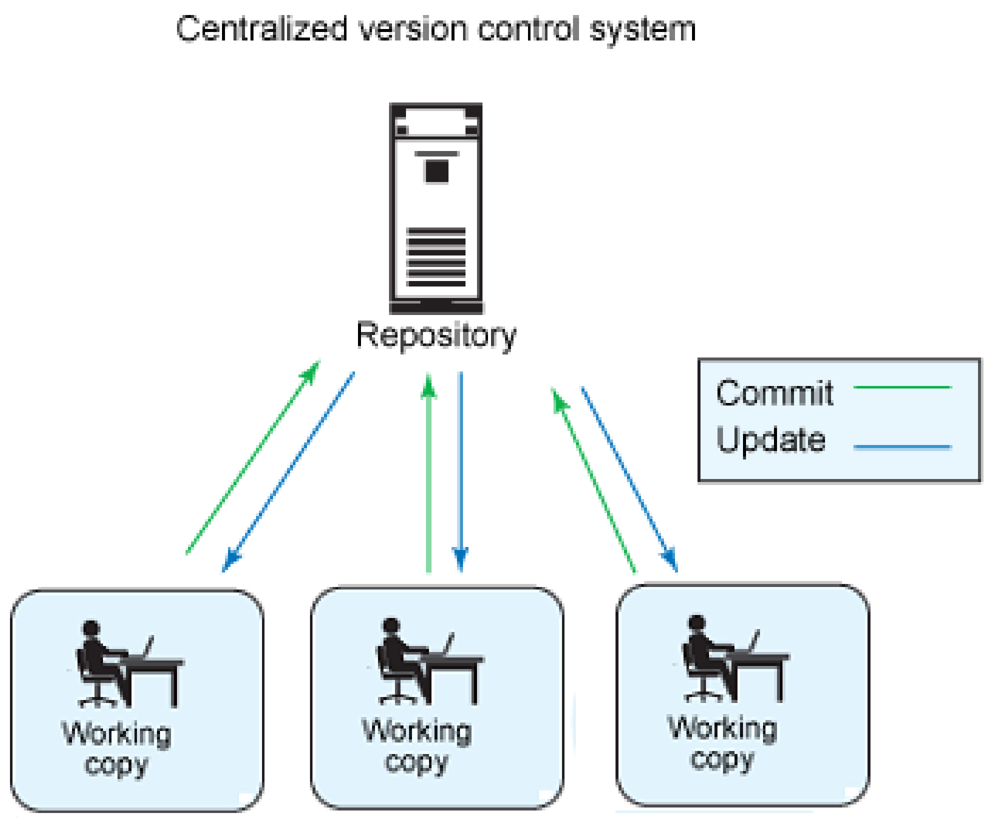
Repositories and working copies

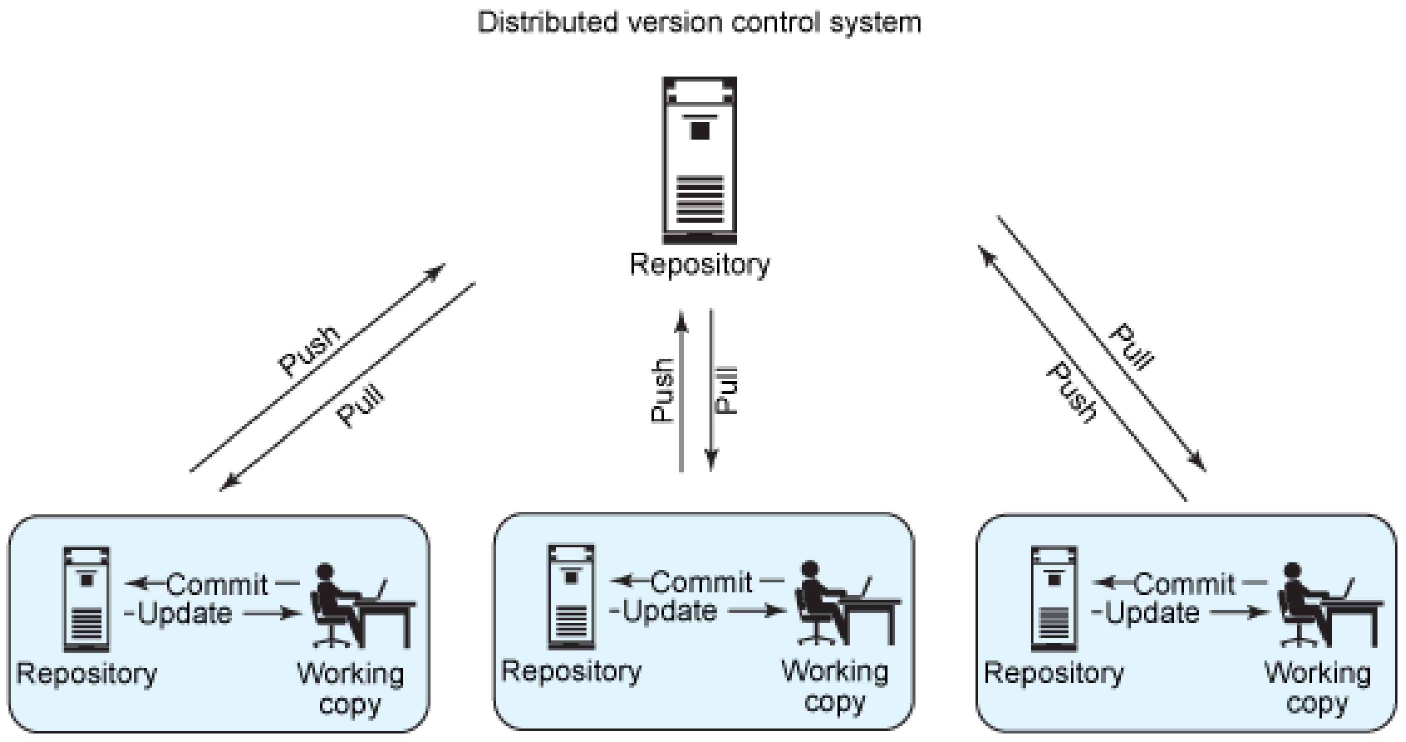


Repository keeps linear and branching history



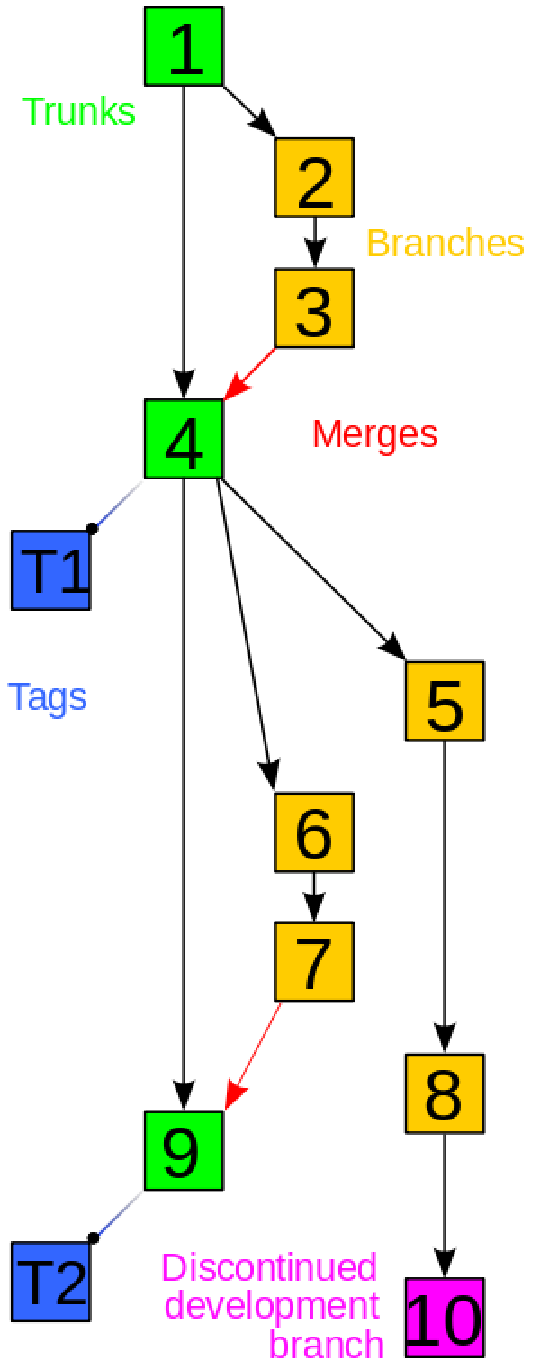
Centralized and Distributed Version Control





How does Version Control work?

- Repository
- Revisions
- Tree baseline
- Branches
- Tags



Git

- A distributed version control system
- Git is the most popular version control system in the industry
- A proper and detailed understanding of Git will allow you to make a transition to any other distributed VCS easily.
- Most popular VCS are similar to Git
- Command-line Tool (Accessible with Terminal on the MAC or Git Bash on Windows)

Installing Git

- You need Git installed on your system, and you can access it in a UNIX Terminal, either the Terminal on the Mac or Git Bash on Windows.
- Download Git from the following link:
  https://git-scm.com/downloads

Version Control Terminology

1. Version Control System (VCS)
2. Source/Software Control Management(SCM)
3. Repository
4. Commit
5. SHA
6. Working Directory
7. Checkout
8. Staging Area/Index
9. Branch

Version Control Terminology

1. **Version Control System :**

A VCS allows you to: revert files back to a previous state, revert the entire project back to a previous state, review changes made over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more.

2. **Repository:**

A directory that contains your project work which are used to communicate with Git. Repositories can exist either locally on your computer or as a remote copy on another computer.

3. Commit
Git thinks of its data like a set of snapshots of a mini file system.
Think of it as a save point during a video game.

4. SHA
A SHA is basically an ID number for each commit.
Ex. E2adf8ae3e2e4ed40add75cc44cf9d0a869afeb6

5. Working Directory
The files that you see in your computer's file system. When you open your project files up on a code editor, you're working with files in the Working Directory.

6. Checkout

When content in the repository has been copied to the Working Directory. It is possible to checkout many things from a repository; a file, a commit, a branch, etc.

7. Staging Area

You can think of the staging area as a prep table where Git will take the next commit. Files on the Staging Index are poised to be added to the repository.

8. Branch

A branch is when a new line of development is created that diverges from the main line of development. This alternative line of development can continue without altering the main line.

Let’s dive into the good stuff now

GIT COMMANDS:

(THINGS WE’LL COVER)

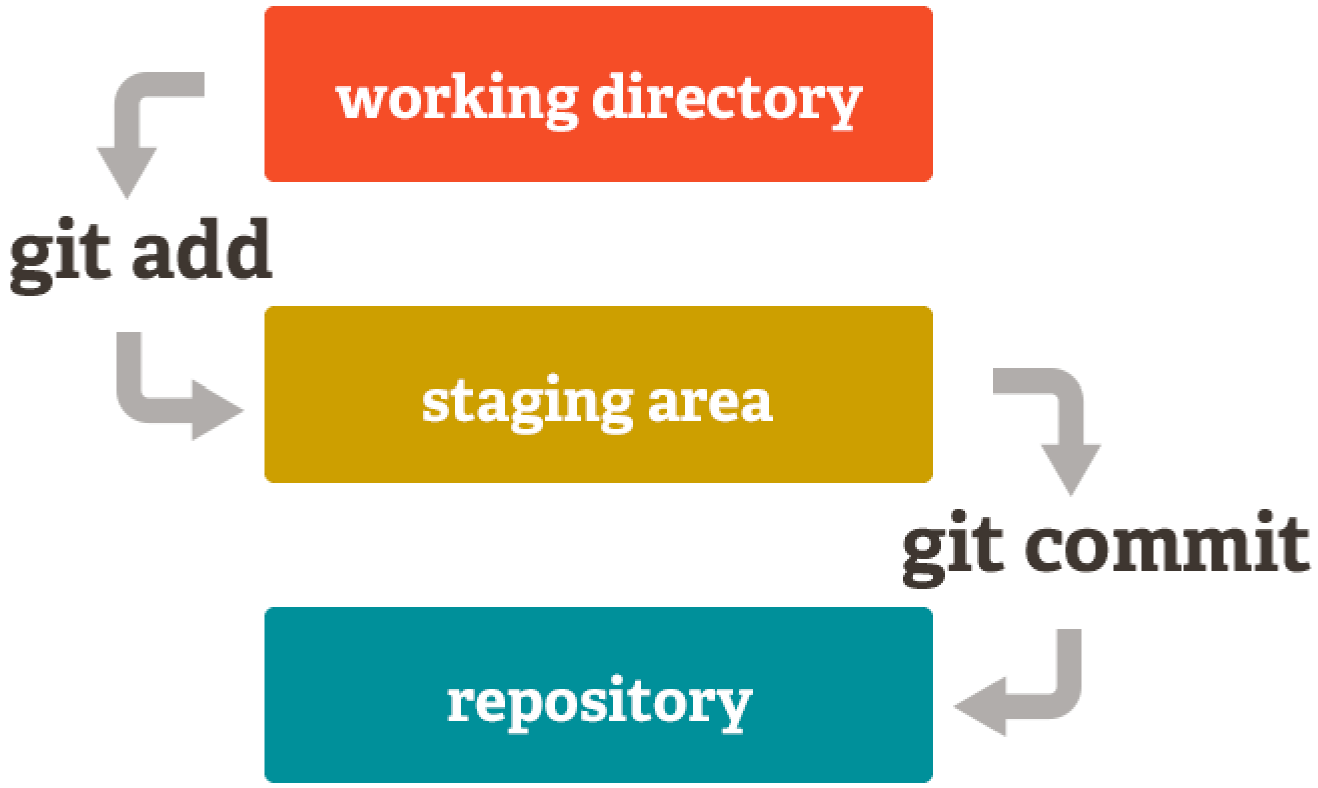
- BASIC GIT COMMANDS
- REMOTE REPOSITORY – PUSH & PULL
- BRANCHING, TAGGING AND MERGING
- DISASTER HAS STRUCK!

Basic Git Commands

- git init – *Initialize a Git repository/working directory*
- git status – *Status of your working directory*
- git add <filename> or git add . *(for all files in your working directory)*
- git commit – *Stash changes in your working directory*
- git log – *View your commit history*
- git clone – *Create an identical copy*

CREATED REPO AND MADE AN INITIAL COMMIT
SO I GUESS YOU COULD SAY THINGS ARE GETTING PRETTY SERIOUS

Basic Git model locally



GitHub

- It's a hosting medium/website for your Git repositories
- Offers powerful collaborative abilities
- A good indicator of what you code/how much you code/quality of your code

Working with a remote repository

- Remote?

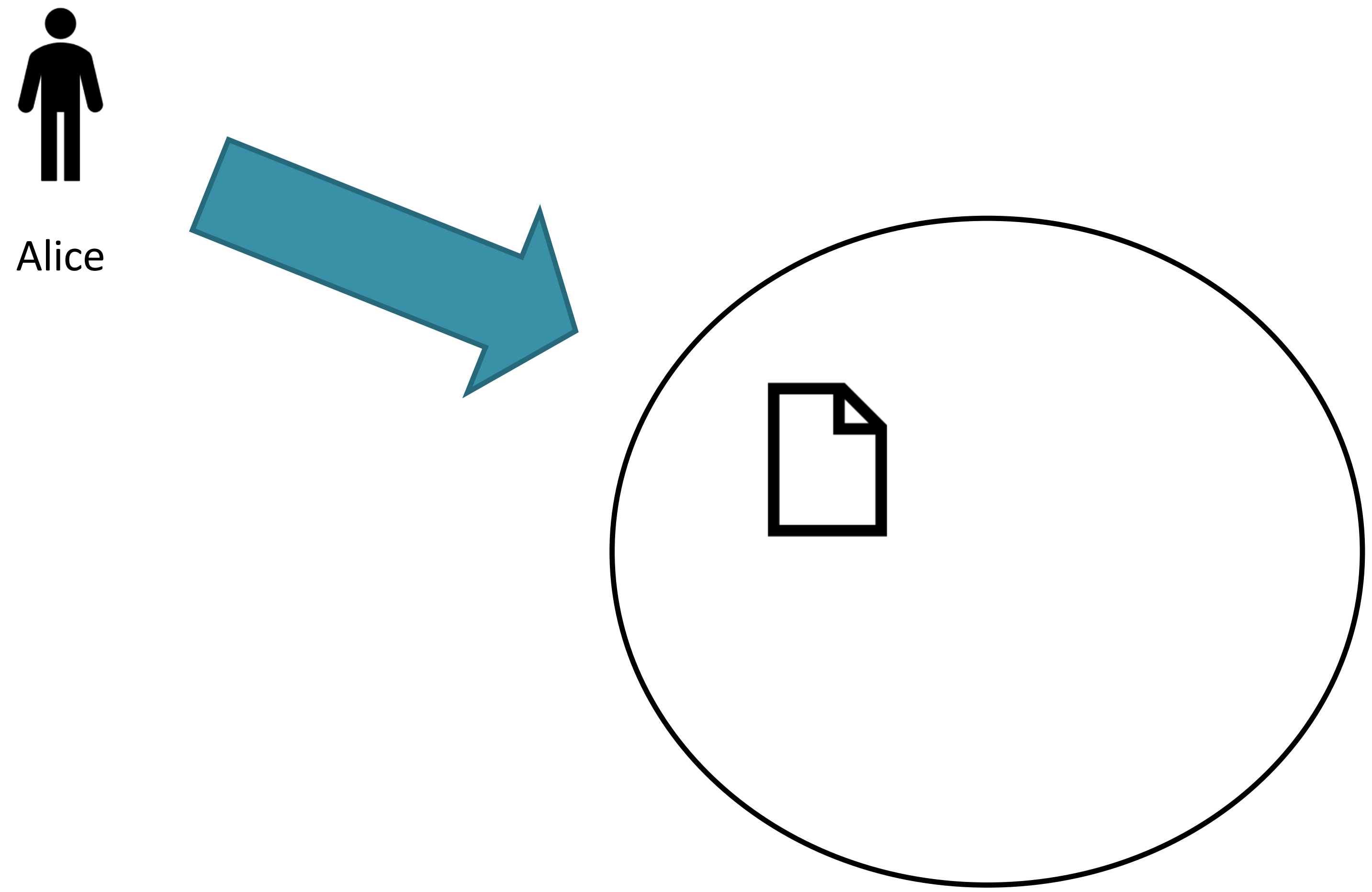
It’s the place where your code is stored.

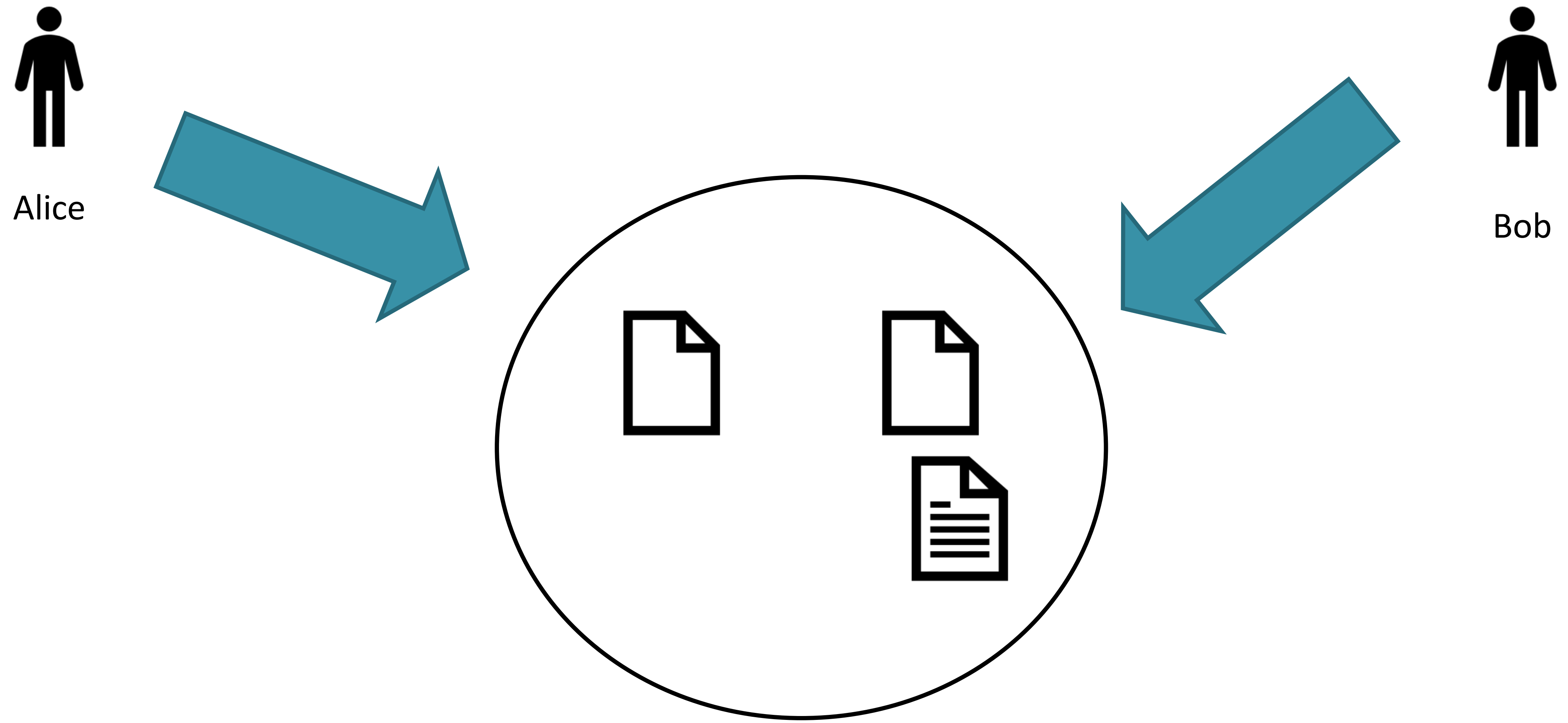
By default, remote name is origin and default branch is master.

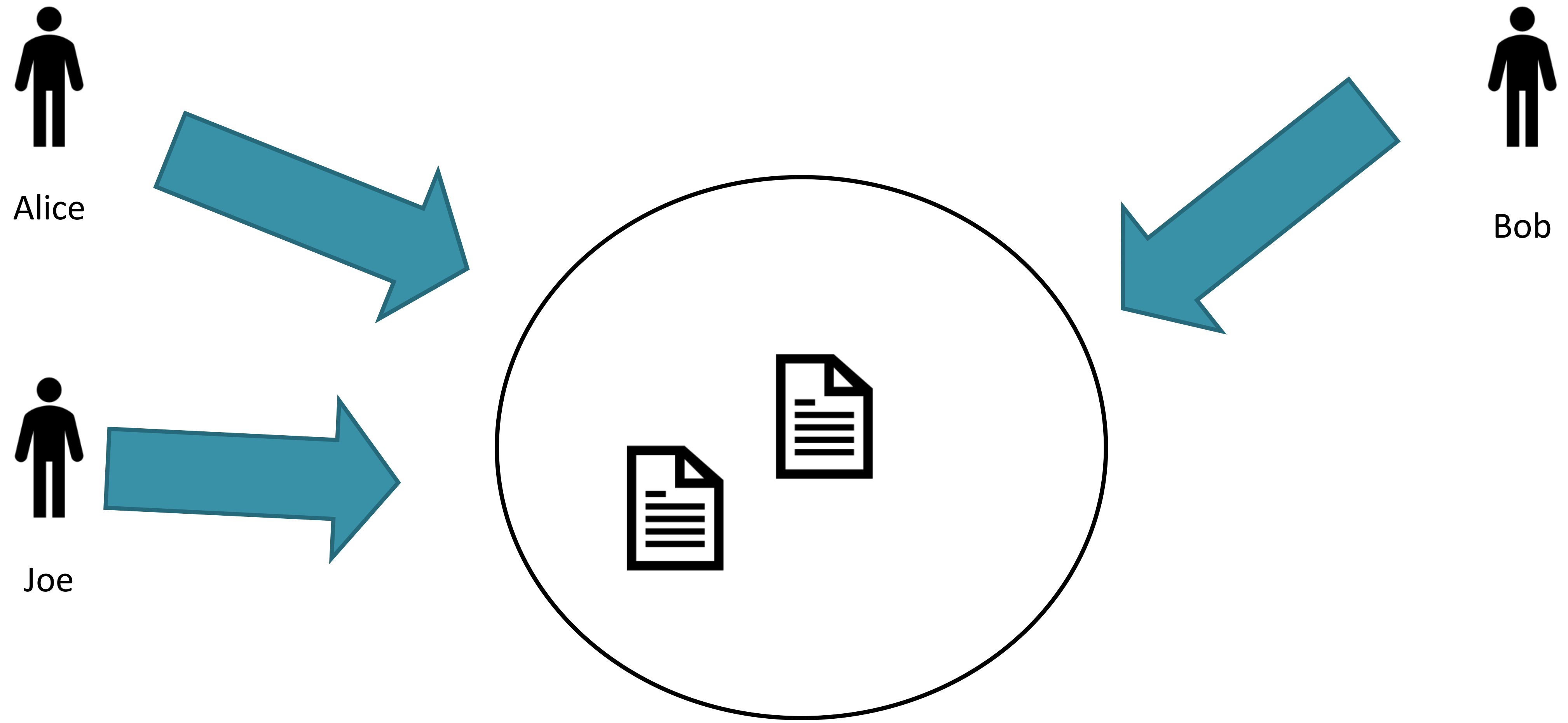
- Certain things that come to play, namely collaboration.

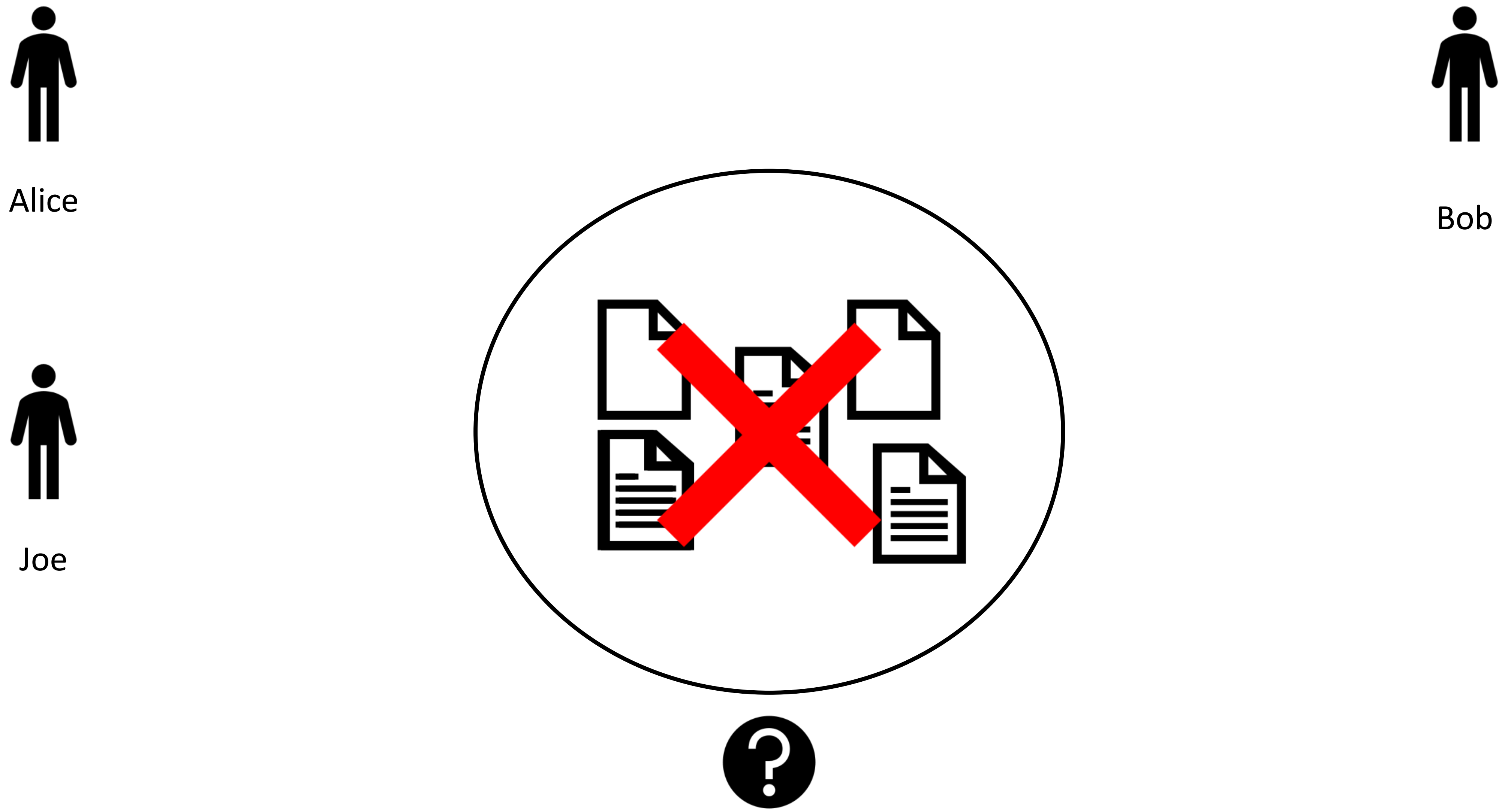
How are we going to handle that with Git.

So here comes, push, pull, branching, merging, **forking**.


Alice


Alice
Bob


Alice
Bob
Joe



Who replaced the files? When ?

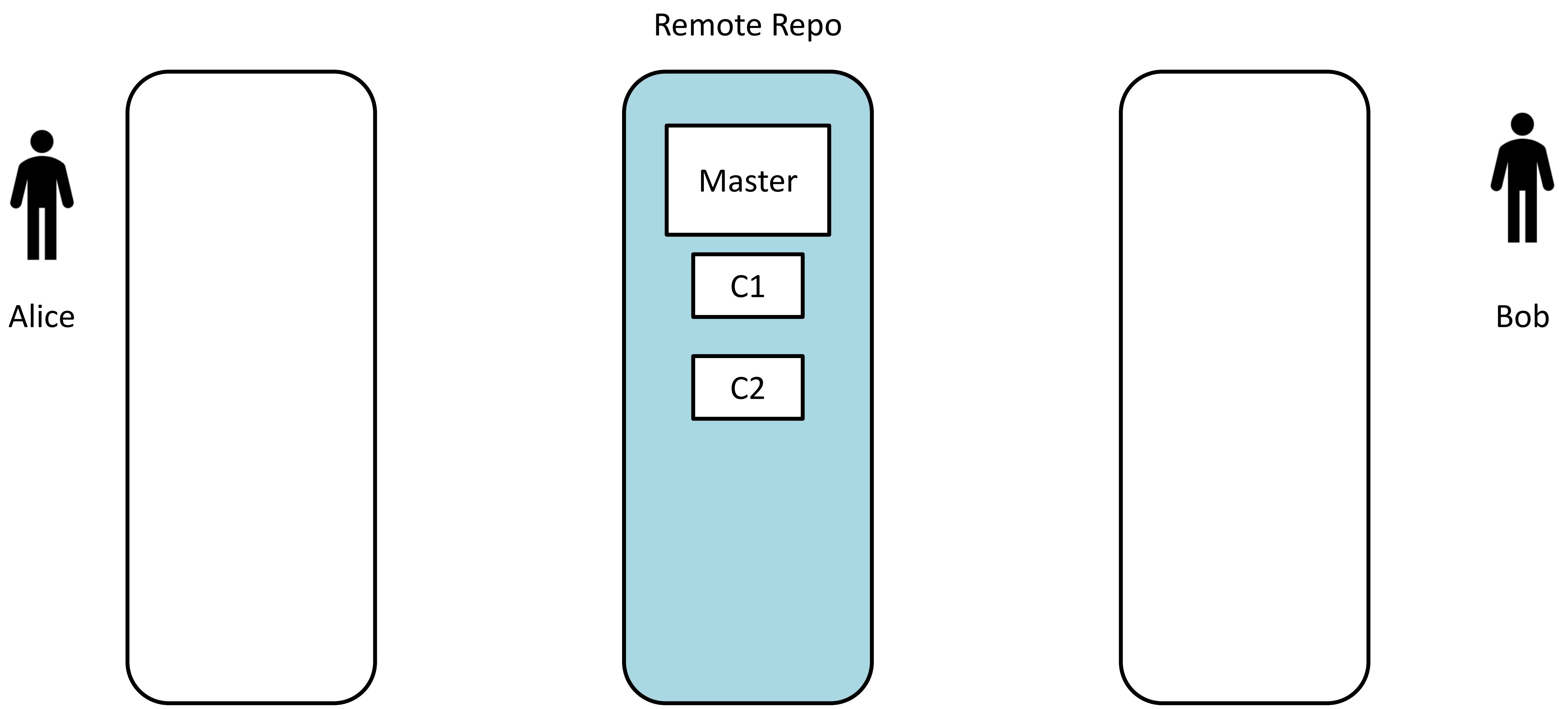
How to access GitHub

1. Access it on github.com
2. Create an account if you don’t already have one.
3. GitHub Clone link:

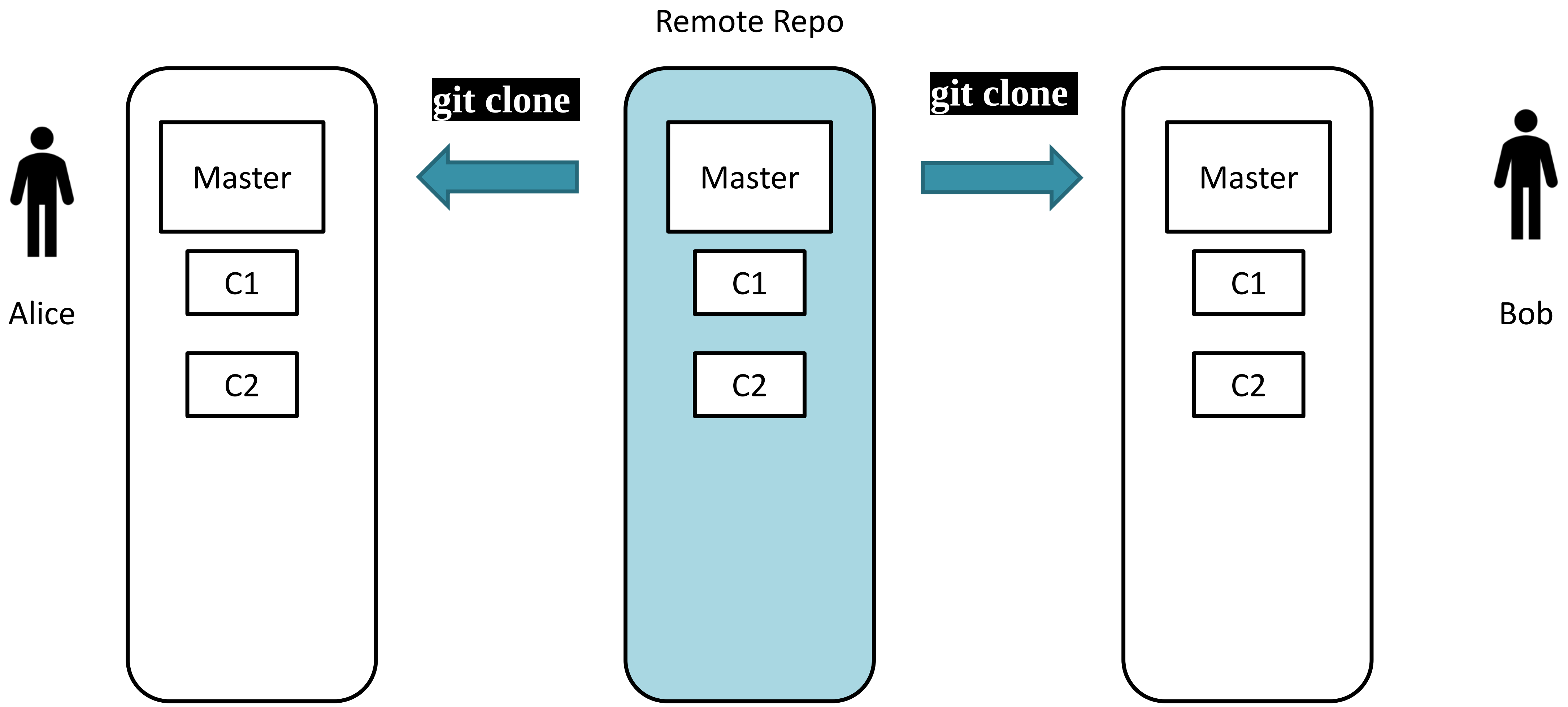
    github.com/accountname/repository_name

    Ex: github.com/amaharjantbc/tbcl3test.git

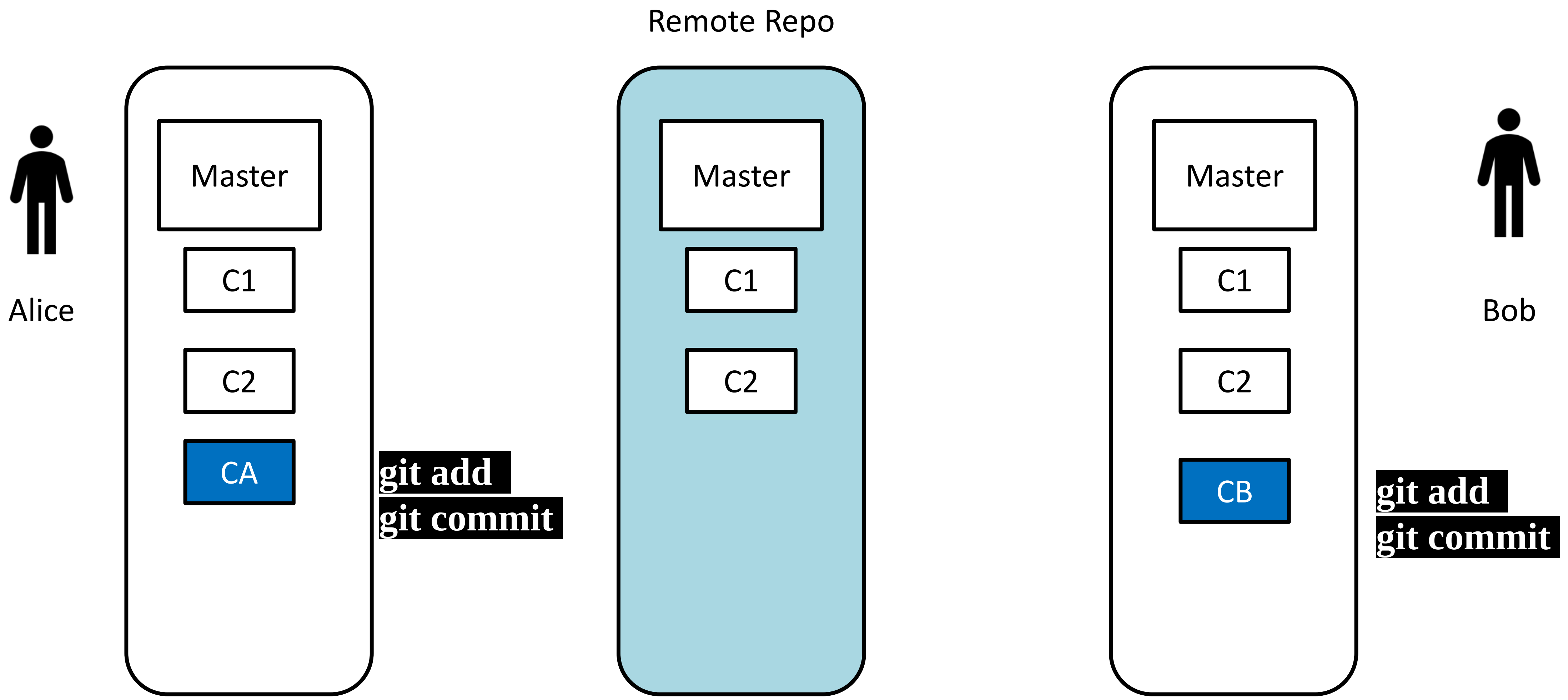
Collaboration with GitHub



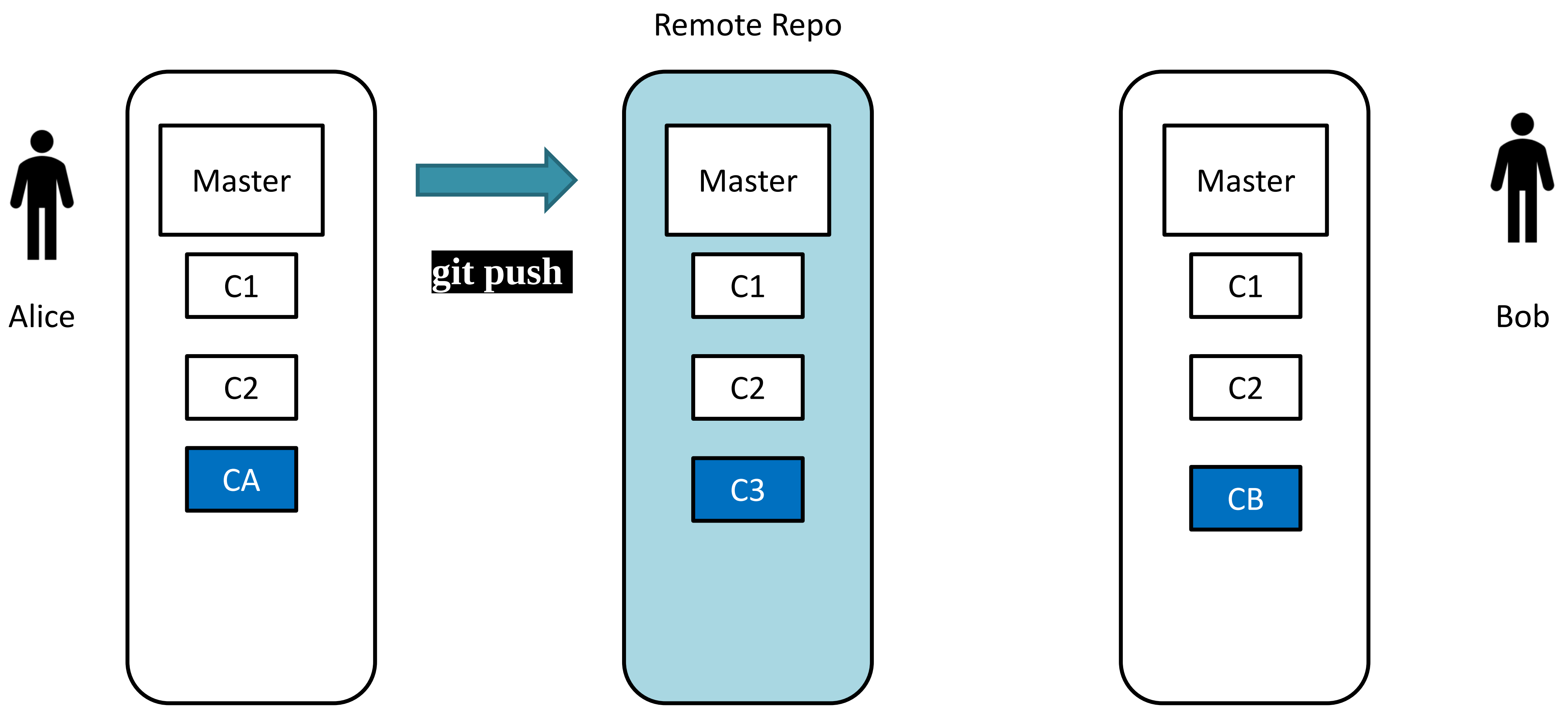
Collaborate



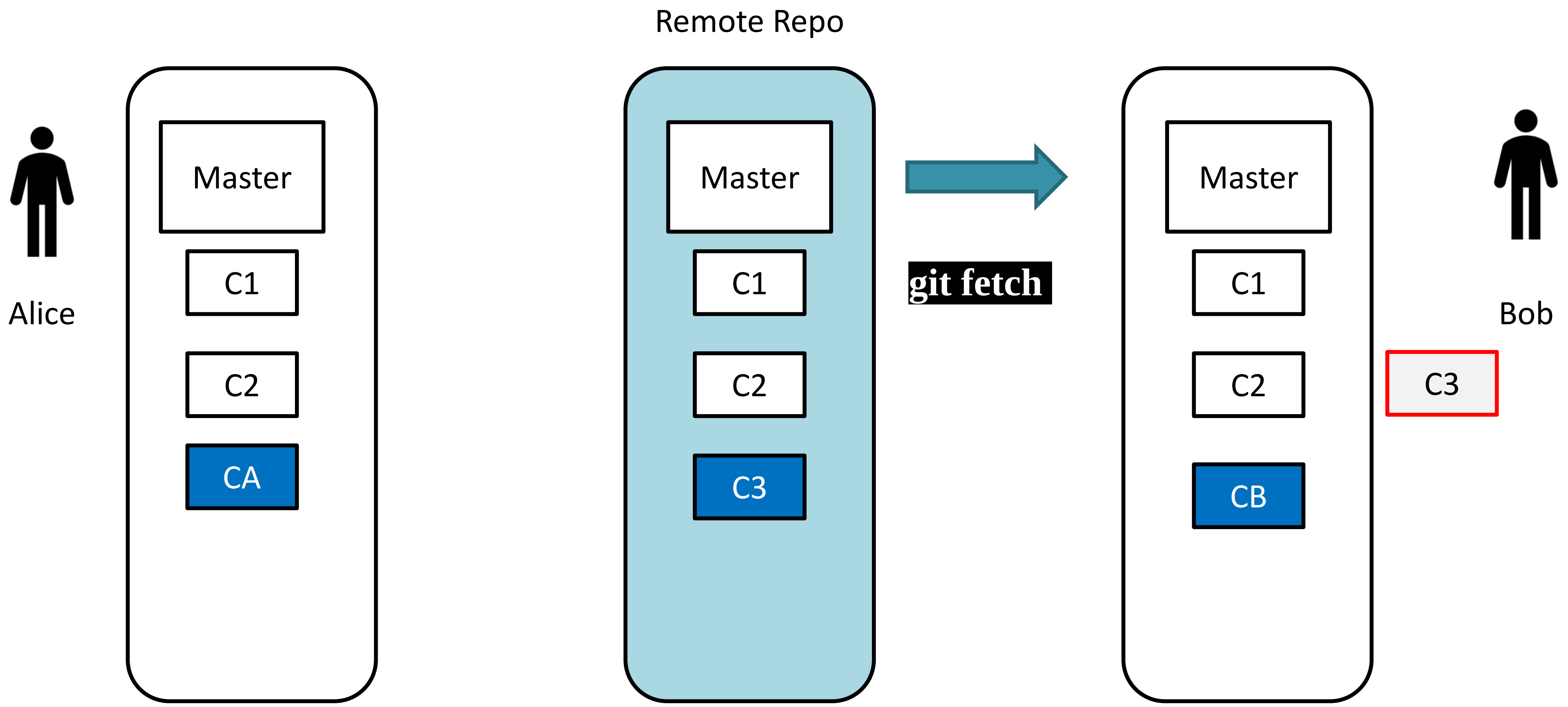
Collaborate



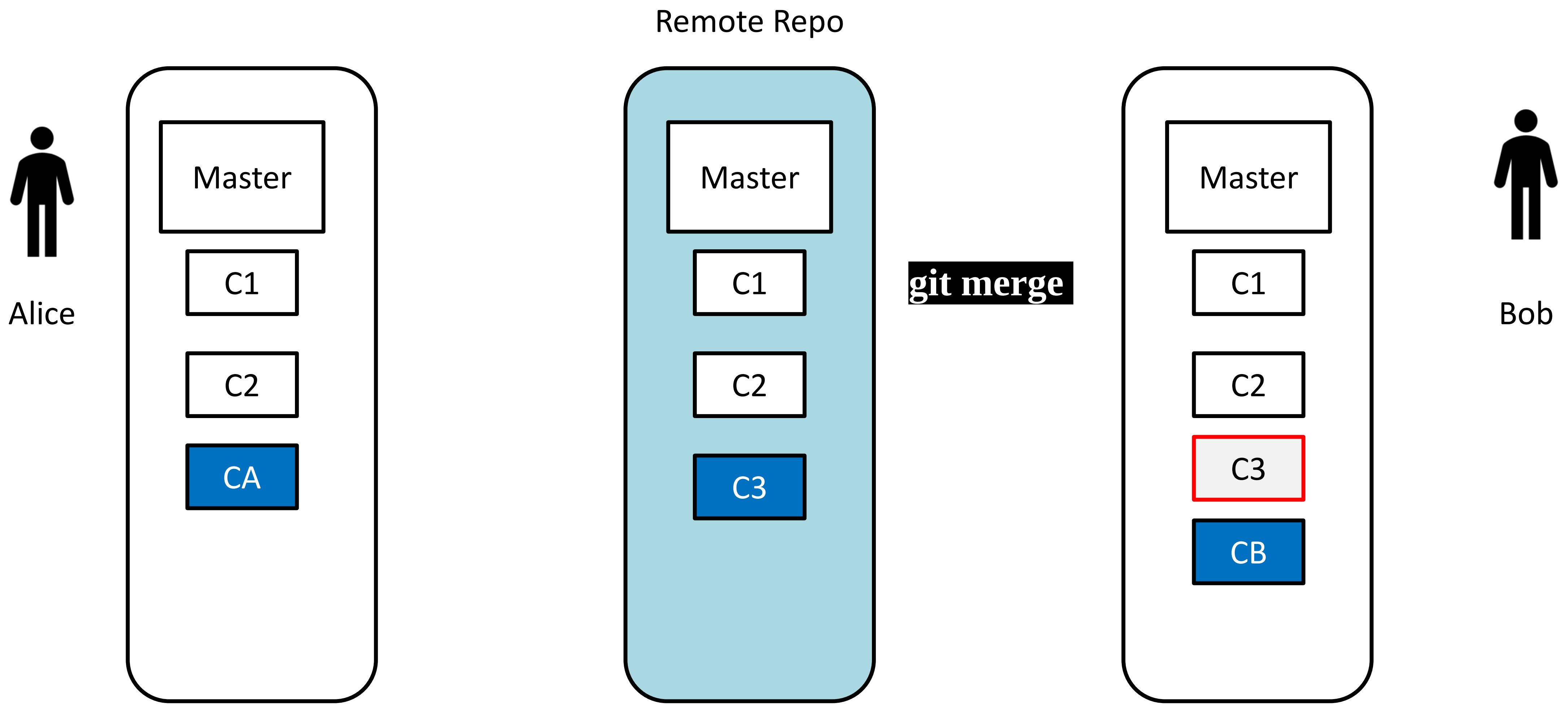
Collaborate



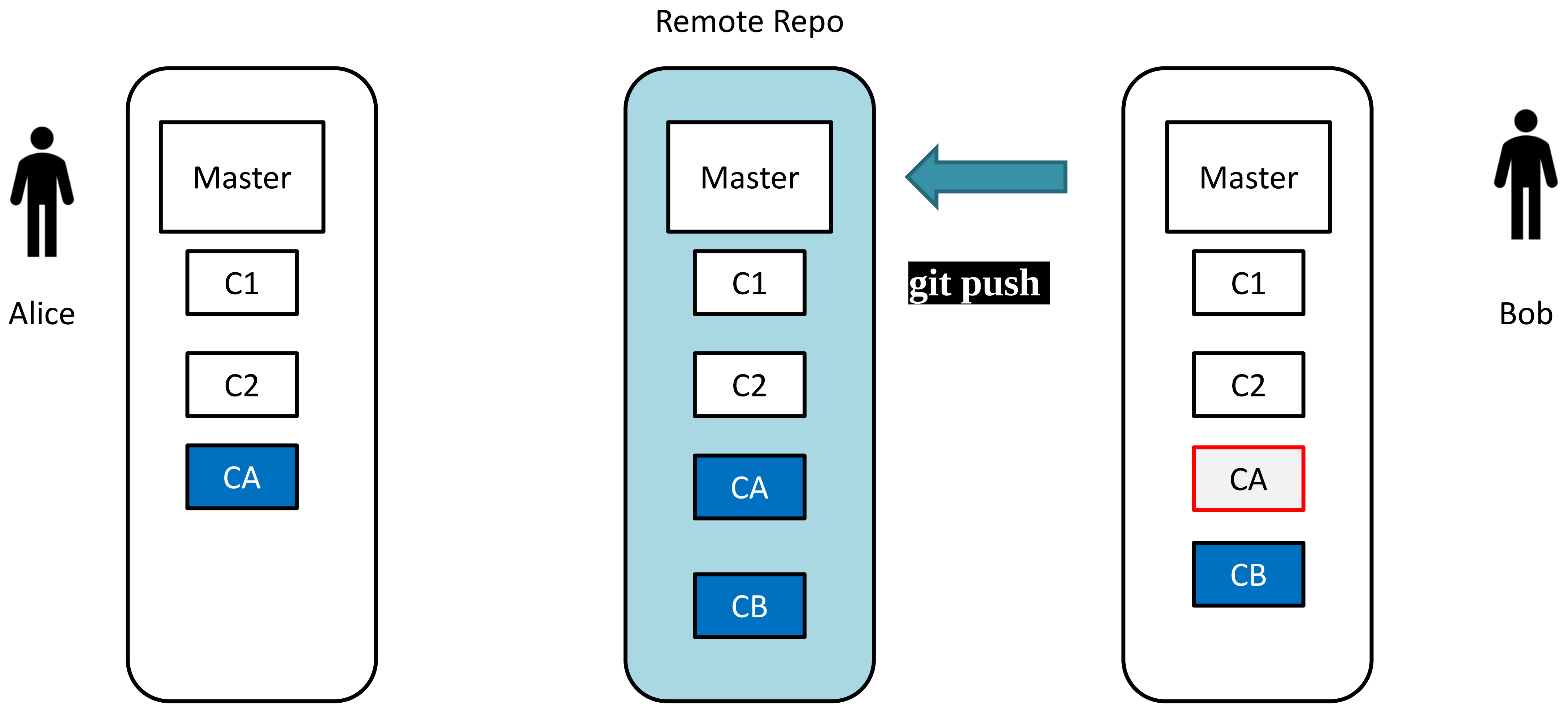
Collaborate



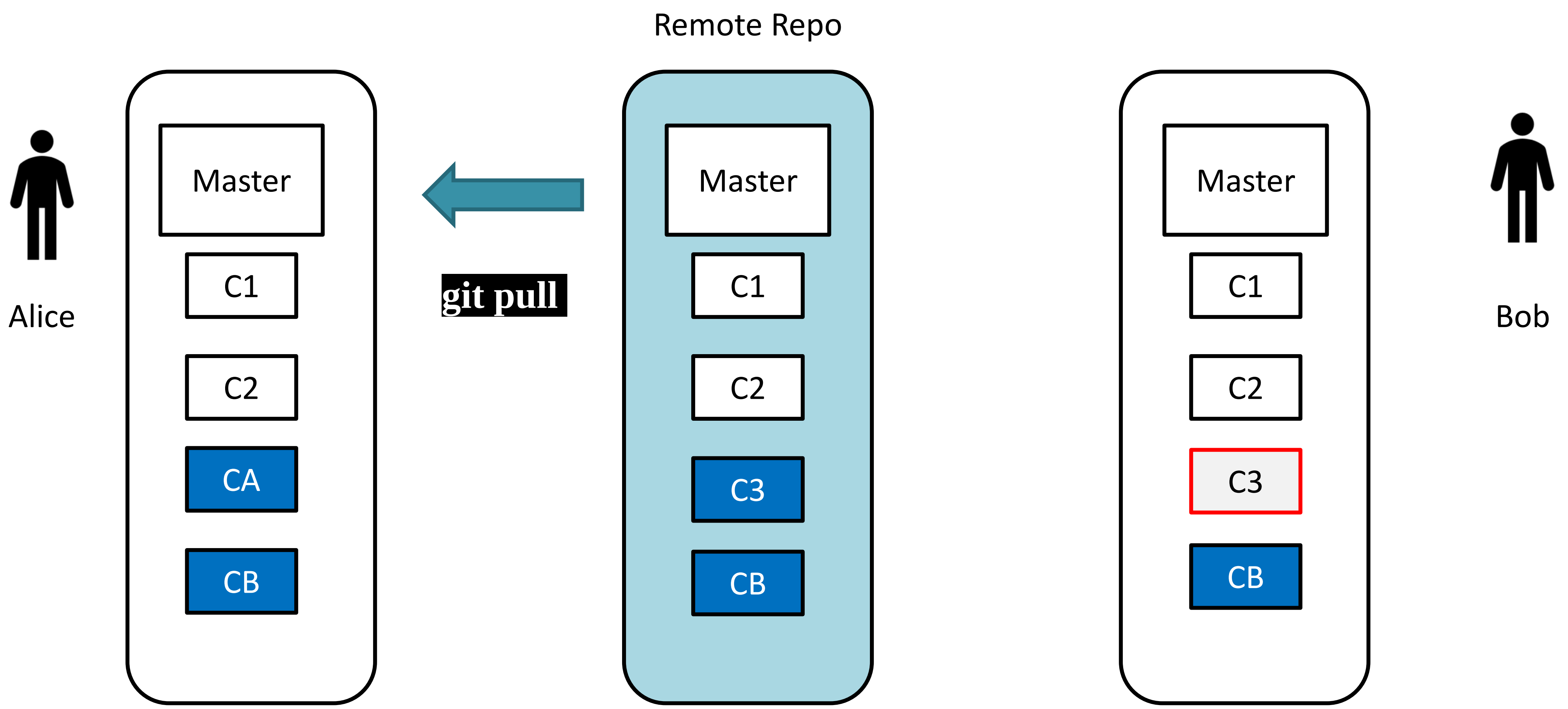
Collaborate



Collaborate



Collaborate



More on Collaborative Git commands

- Deleting a branch

git branch –d <branchname>

- Merge conflicts! How do you resolve?
- Pull requests ?
- git tag –a v1.0 / git tag –a v1.0 <commit sha>

Dealing with Merge Conflicts

- Two typical cases of merge conflicts

1. Normal merge where you’re a collaborator
2. Pull request (handled by the repository owner)

More to learn in Git/Forward Steps

- Firstly, I highly recommended building a project from scratch with Git integrated and if possible, with other programmers.
- As far as Git goes, this PowerPoint covers the basics to help you get started, but as you work with version control, you will encounter new commands required.
- Rebasing – concept you could cover.

GitHub Demo

More Git commands

- git push – push your changes into the remote repository
- git pull – pull your latest changes from the remote repository
- git branch – view branches in your repository
- git branch <branchname> - create a branch
- git checkout <branchname> - move to that branch
- git merge <branchname> - merge into that branch
- git revert <commit sha>

GitLab

- GitLab is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking and continuous integration/continuous deployment pipeline features
- an open-source license, developed by GitLab Inc

Apache SVN or Tortoise SVN

- a software versioning and revision control system distributed as open source under the Apache License.
- Software developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation
- GitLab is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking and continuous integration/continuous deployment pipeline[7] features, using an open-source license, developed by GitLab Inc