Enterprise to the Power of Digital^{TM}

DevOps On Java Platform Using GIT Tool

Mahendra Pachpor

birlasoft

CK BIRLA GROUP

Engaged. Dependable. Challenger.

That's who we are…

birlasoft
with the IT edge of KPIT

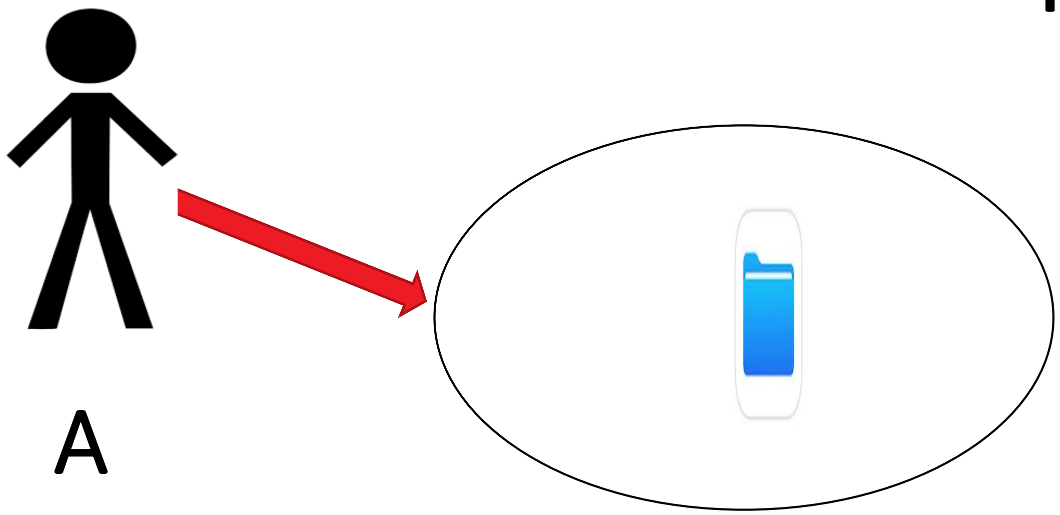
Session Objectives

- Introduction to version control systems
- Localized and centralized version control systems
- Distributed version control systems
- What is Git?
- Set-up Git in your system
- Create a repository in Git and GitHub
- Working with staging area
- Creating local repository for staging and committing files
- Maintain the log of commits performed
- Merge the commits locally
- Create branches and track files
- Sync with remote repository
- Pushing local repository updates to GIT remote repository

How Developers work ?

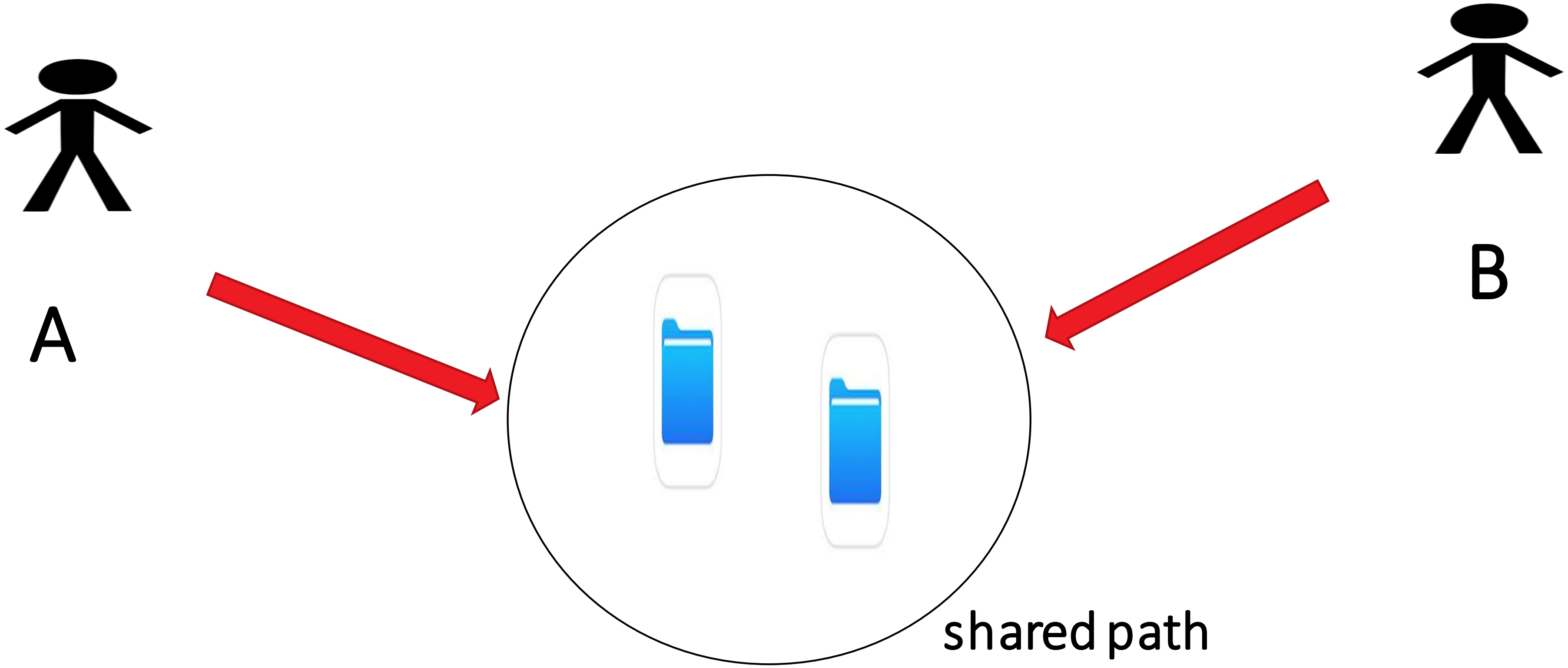
- Work in a team, probably on particular components ?
- Integrate the code together.
- Make copies of the files on local system
  or on the shared path

How Developers work ?

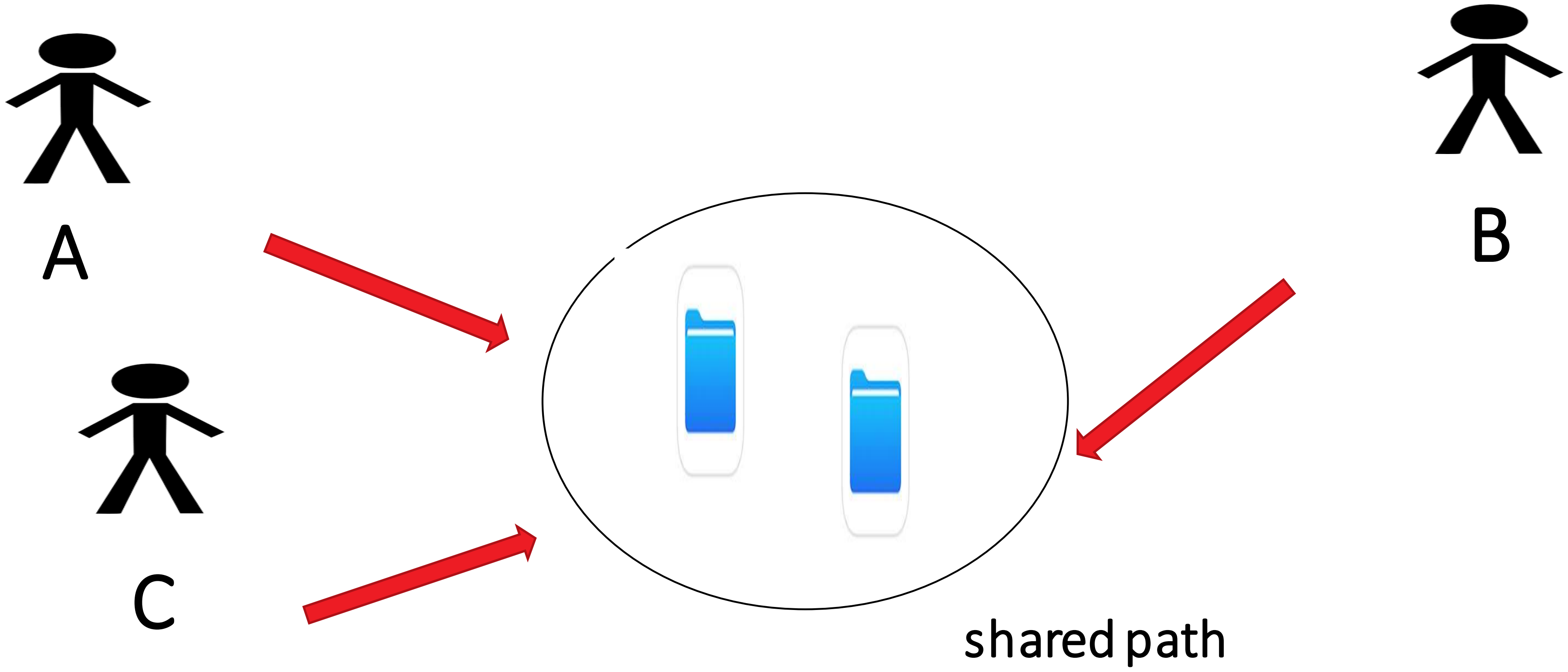


shared path

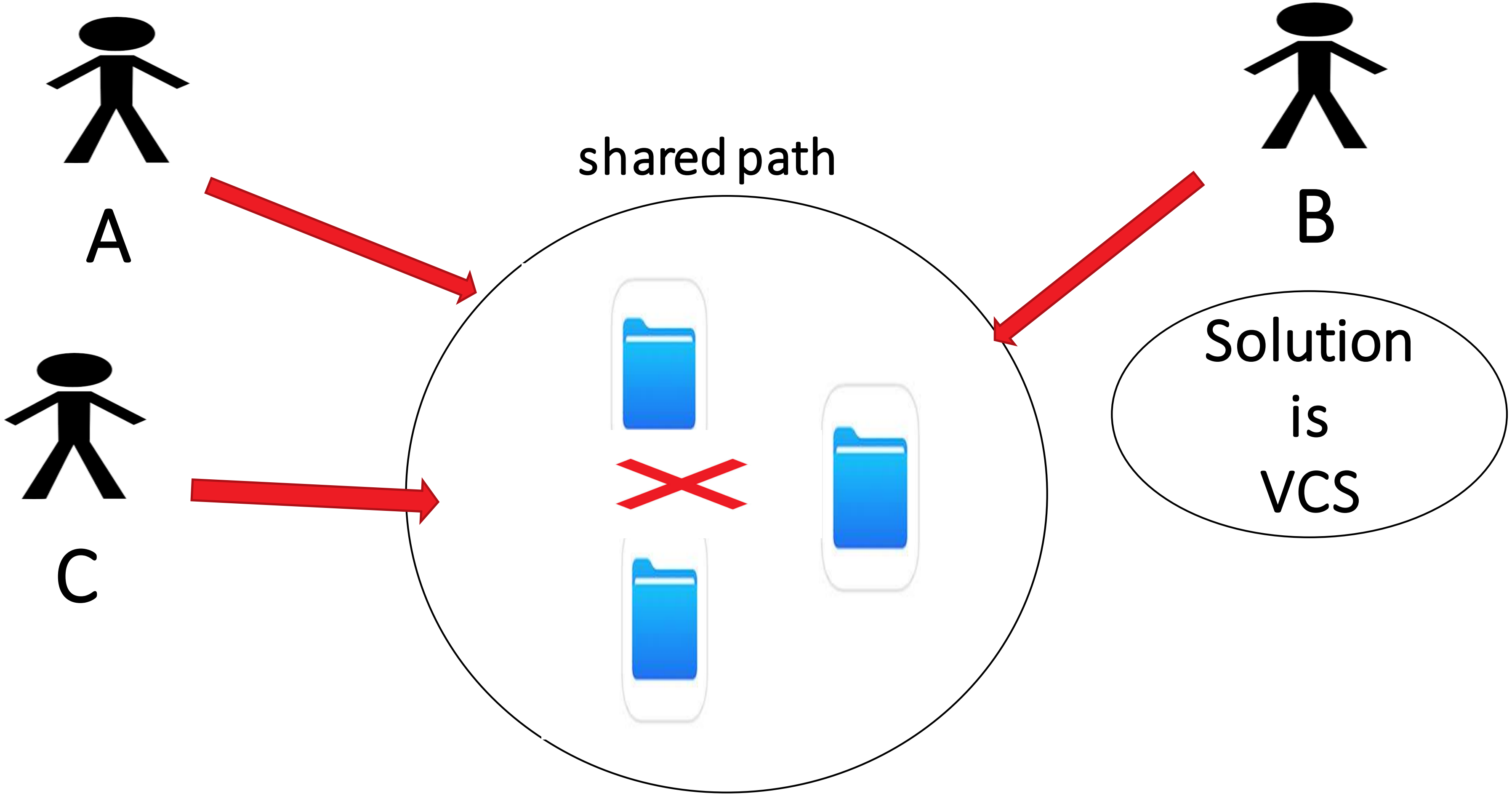
How Developers work ?



How Developers work ?



How Developers work ?



A

shared path

B

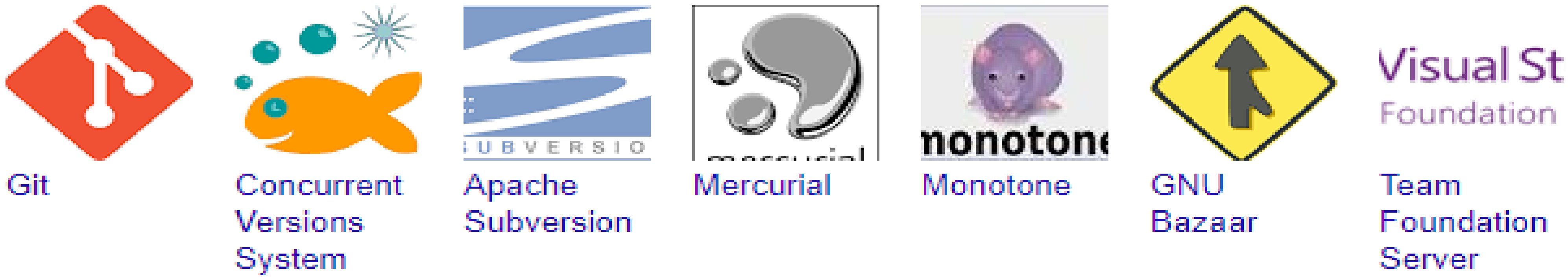
C

Solution is VCS

Who replaced files?
When ?

Version Control?

- A component of software configuration management
- System that records changes to a file or set of files over time so that you can recall specific versions easily.
- If you screw things up or lose files, you can easily recover for very little overhead.



Git

Concurrent Versions System

Apache Subversion

Mercurial

Monotone

GNU Bazaar

Team Foundation Server

Why Version Control?

- Developers can share codebase easily
- Keep track of all changes in development
- Compare code base.
- Maintain working copies of same code base
- Communicate changes made to code

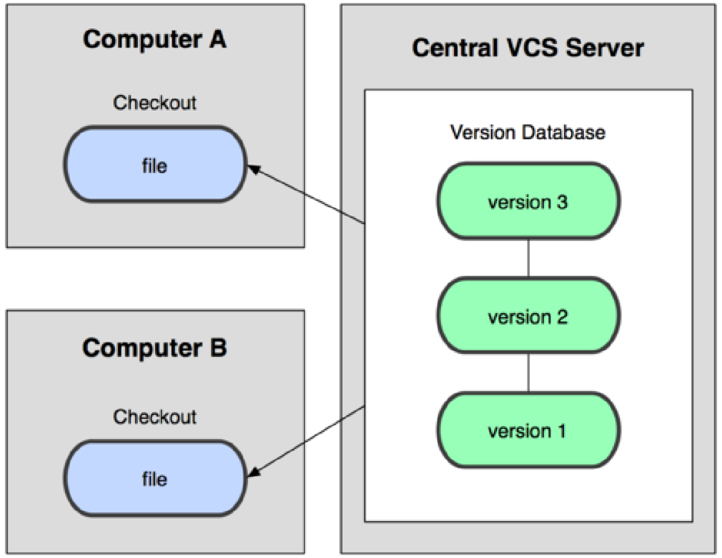
Makes working in a team easy!

Keep your code secure

Centralized VCS

- A central server repository (repo)
- Holds the "official copy" of the code
- Server maintains version history of the repo
- You make "checkouts" of it to your local copy
- When you're done, you "check in" back to the server

“your check in increments the repo's version”



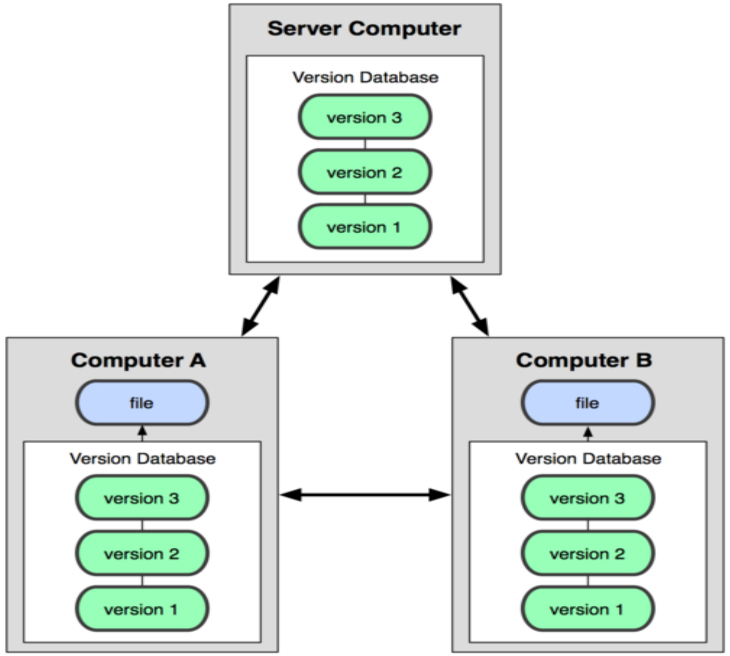
Distributed VCS (Git)

“In git no "checkout” from a central repo
you "**clone**" it and "**pull**" changes “

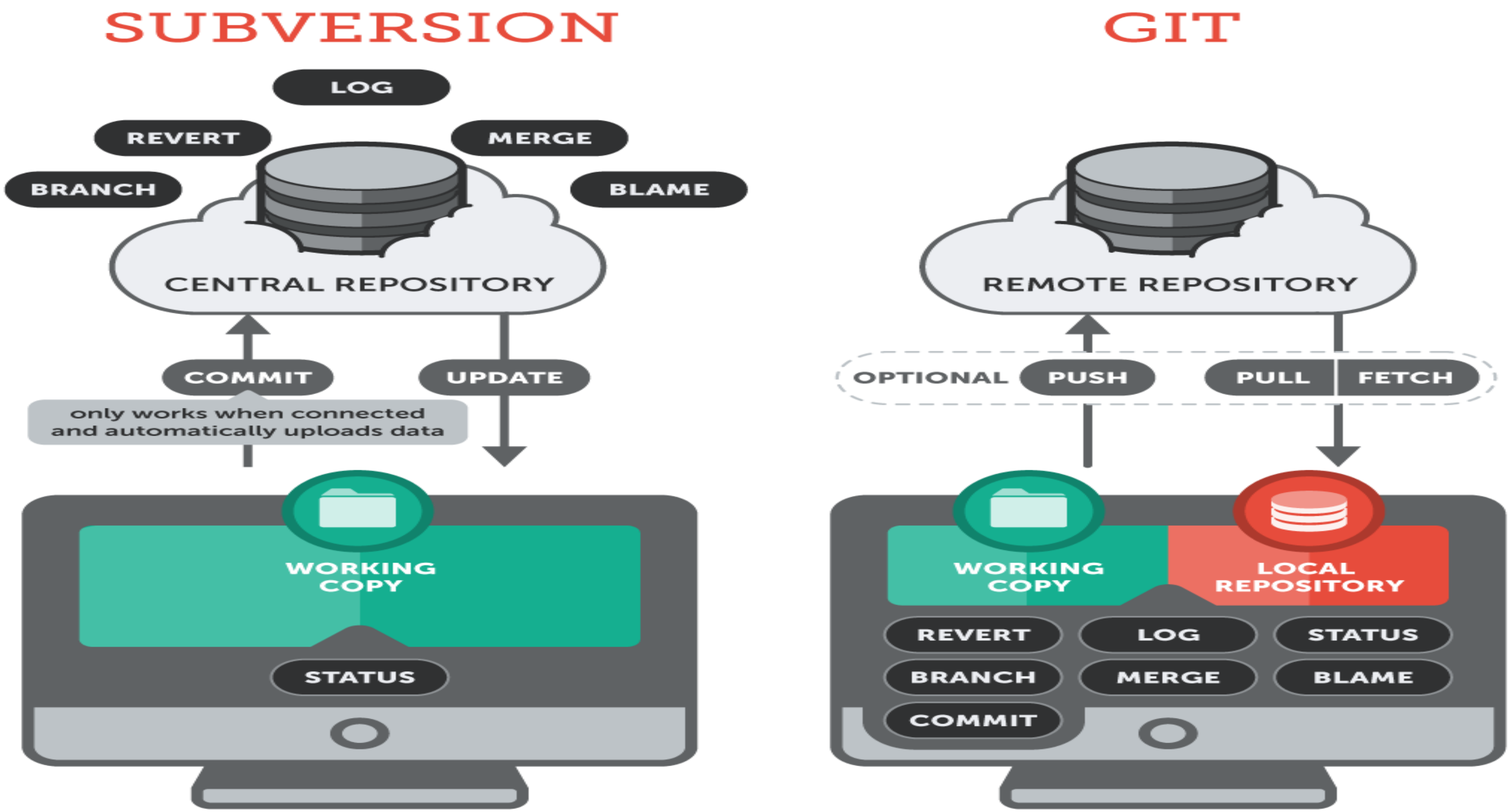
- Local repo is a complete copy of everything on the remote server

- Many operations are local:
  I. check in/out from local repo
  II. commit changes to local repo
  III. local repo keeps version history

- When you're ready, you can "push" changes back to server



Centralized vs.Distributed



Version Control Terminology

Version Control System (VCS) or (SCM)

- Repository
- Commit
- SHA
- Working Directory
- Checkout
- Staging Area/Index
- Branch

Version Control Terminology

Repository:

A directory that contains your project work which is used to communicate with Git.

Repositories can exist either locally on your computer or as a remote copy on another computer.

Commit

Git thinks of its data like a set of snapshots of a mini file system.

Version Control Terminology

SHA

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

Working Directory

The files that you see in your computer's file system.
Developer’s daily work on the files in the Working Directory.

Version Control Terminology

Checkout

When content in the repository has been copied to the Working Directory. (file, a commit, a branch..)

Staging Area

You can think of the staging area where Git will take the next commit. Files on the Staging are added to the repository.

Branch

A branch is when a new line of development is created
Developing new enhance model parallel to main root

What Is GIT

- Git is a distributed version control system.
- A tool that allows you to track your code history.
- An important tool to collaborate with other fellow developers
- Track down developers work who made the changes
  and what changes are done.

Organizations Using Github For Their Development Process?

Netflix

Amazon

Google

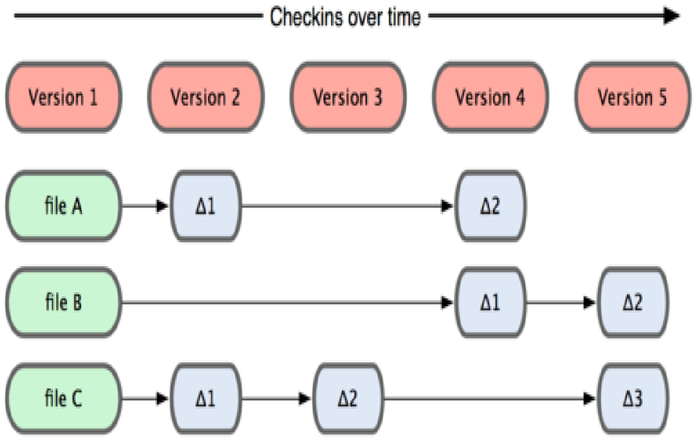
IBM

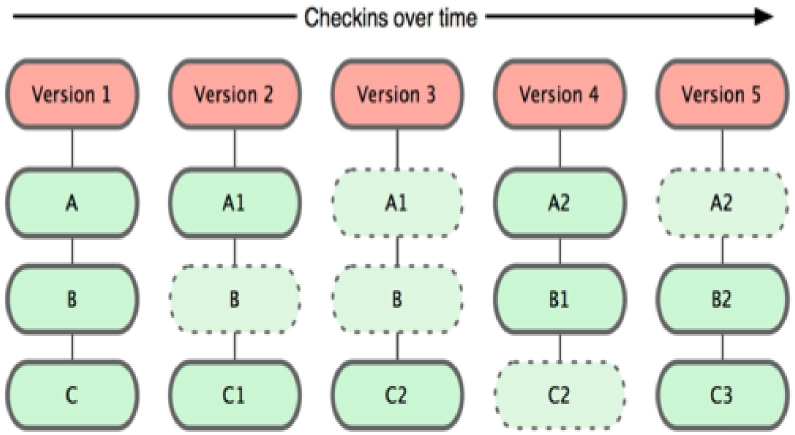
Why GIT

- Came out of Linux development community
- Designed to do version control on Linux kernel
- Goals of Git:
- Speed
- Support for non-linear development
- (thousands of parallel branches)
- Fully distributed
- Able to handle large projects efficiently

Created by
Linus Torvalds

Git Snapshots





Centralized VCS like Subversion track version data on each individual file.

-

Git keeps "snapshots" of the entire state of the project.

– Works faster

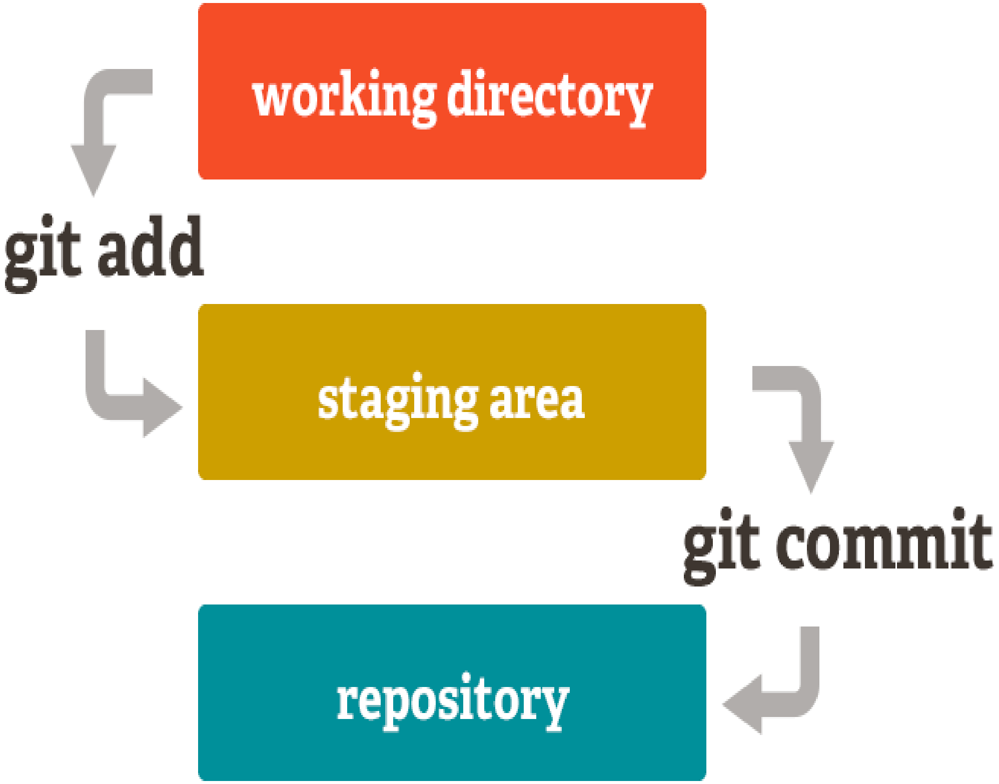
Git
Snapshots not
differences

Initial Git Configuration

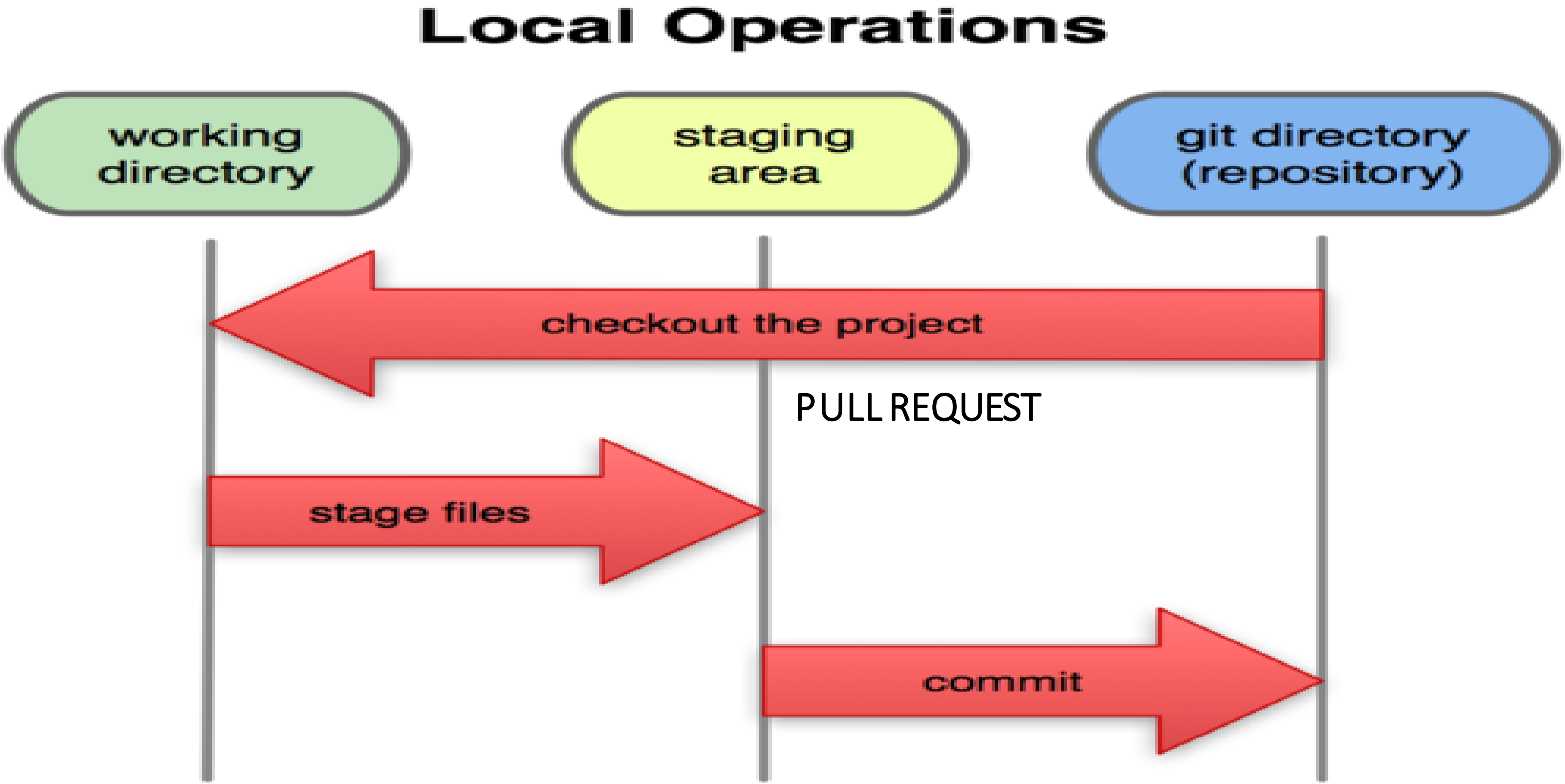
- Set the name and email for Git to use when you commit:
- git config --global user.name “User Name“
- git config --global user.email “User@domain.com”
- You can call git config –list to verify .
- Set the editor that is used for writing commit messages:
- git config --global core.editor (it is vim by default)

First time git setup

Basic Git Model Locally

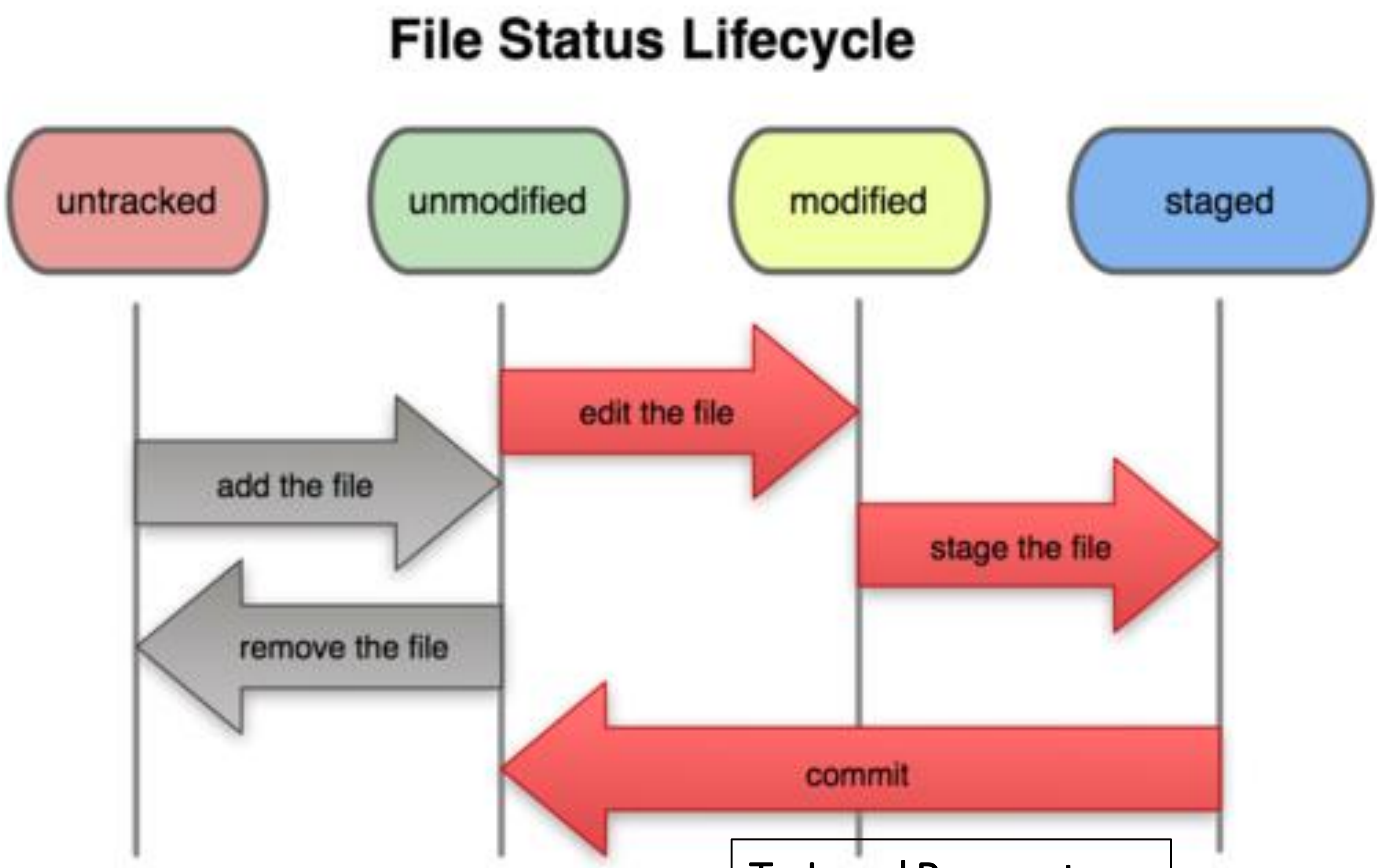


Local Git Areas



PULL REQUEST

Basic Git WorkFlow



To Local Reposotroy

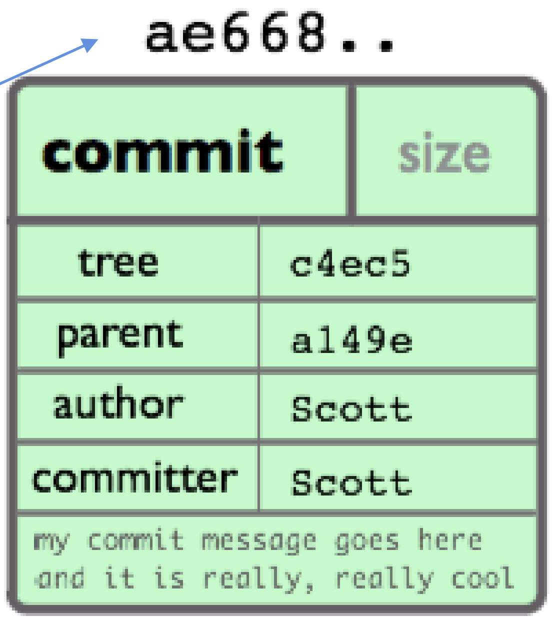
Git Commit Checksums

In Git, each user has their own copy of the repo, and commits changes to their local copy of the repo before pushing to the central server.

Git generates a unique **SHA-1 hash**

(**40 character string** of hex digits) for every commit.

Refers to commits by this ID rather than a version number.



ae668..

commit	size
tree	c4ec5
parent	a149e
author	Scott
committer	Scott

my commit message goes here
and it is really, really cool

Creating Git Repository

Two common scenarios:

- To create a new local Git repo in your current directory:

```
git init
```

- This will create a .git directory in your current directory.
- Then you can commit files in that directory into the repo.

```
git add filename
```

- git commit –m "*commit message*"

Creating a Git repo

- To clone a remote repo to your current directory:
- git clone *url-Remote-Repo localDirectoryName*
- This will create the given local directory, containing a working copy of the files from the repo,
- A .git directory

(hold the staging area and your actual local repo)

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** –oneline – View your commit history
- **git clone** – Create an identical copy
- Other commands like git show, git ignore, git diff etc.

GitHub

- It’s a hosting medium/website for your Git repositories
- It’s Remote Repository
- A good indicator of what you code/how much you code/quality of your code

Code review Project management

Team management Social coding

Documentation Code hosting

How To Access GitHub

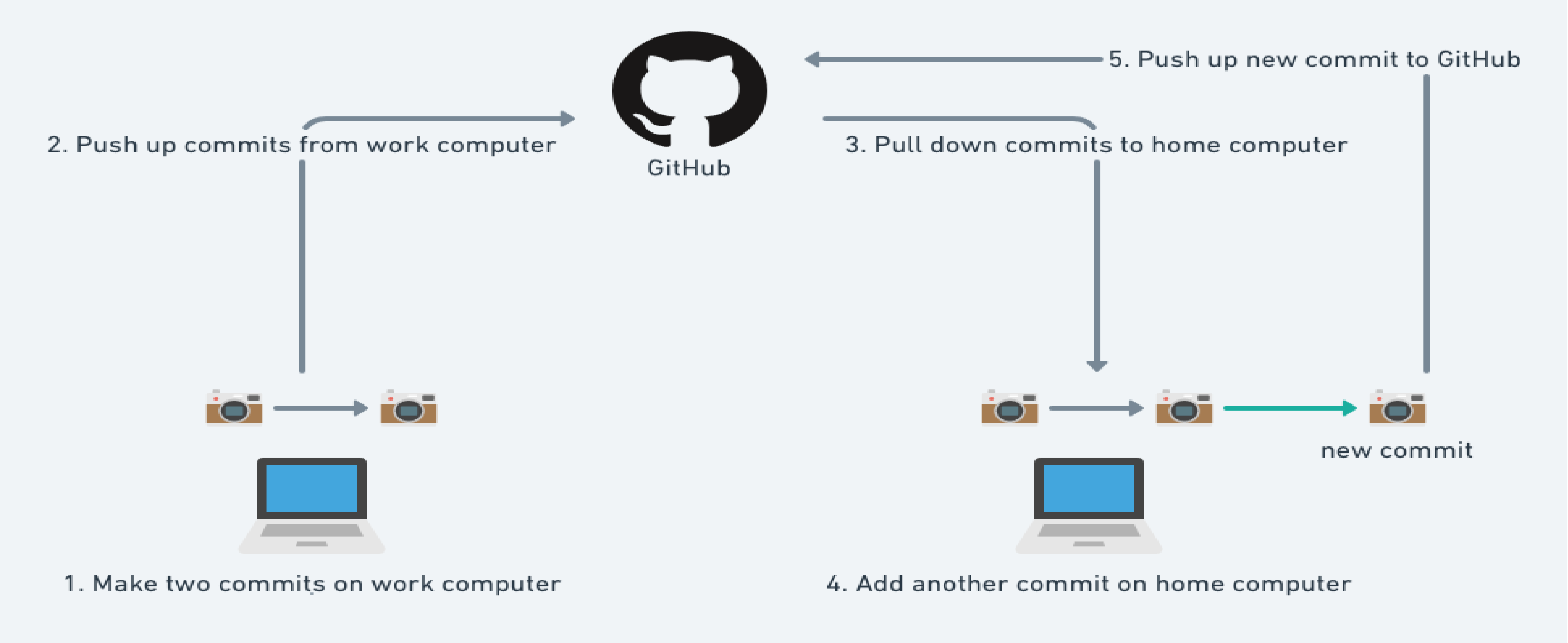
- Access it on github.com

- Create an account .

- GitHub Clone link:

  e.g. github.com/gitRep/GitIntro

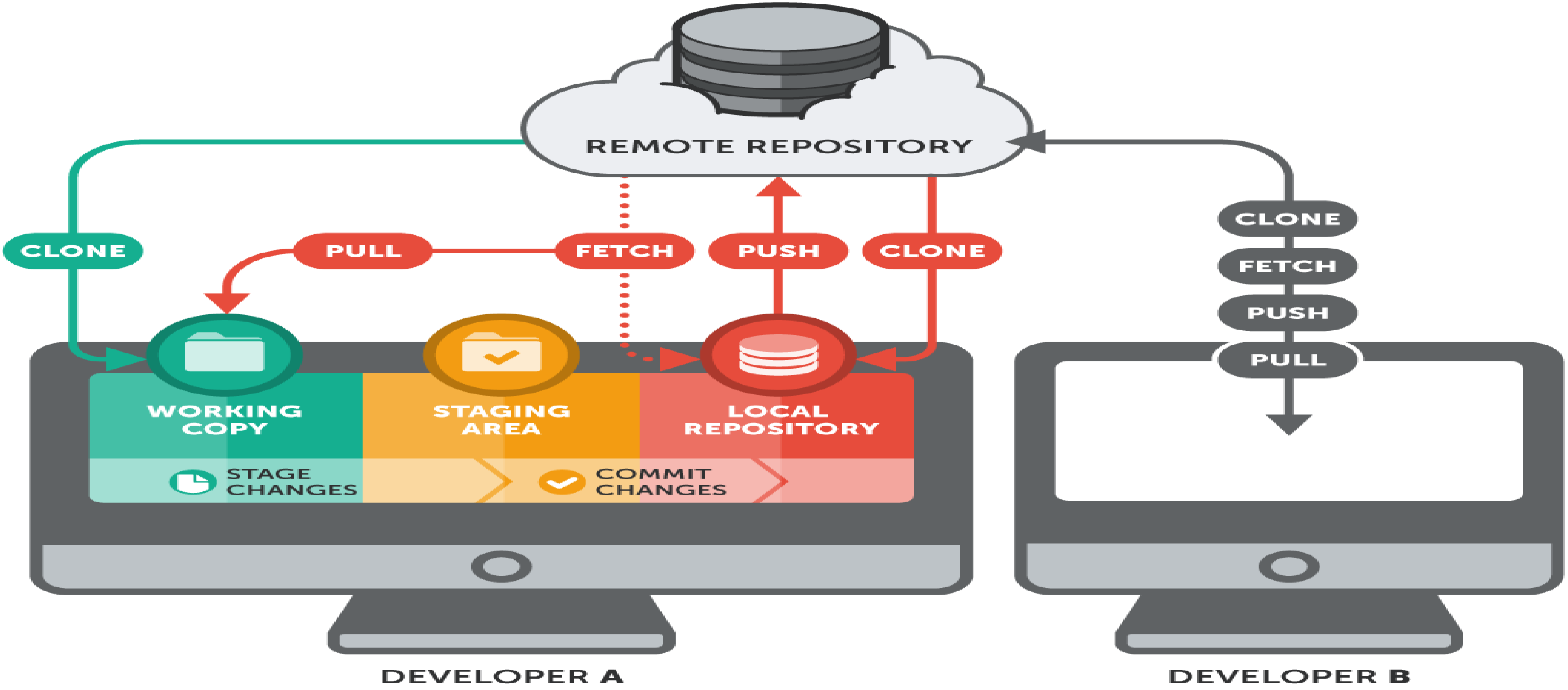
Using Git And Github In Development WorkFlows?


GitHub
5. Push up new commit to GitHub
2. Push up commits from work computer
3. Pull down commits to home computer
new commit
1. Make two commits on work computer
4. Add another commit on home computer

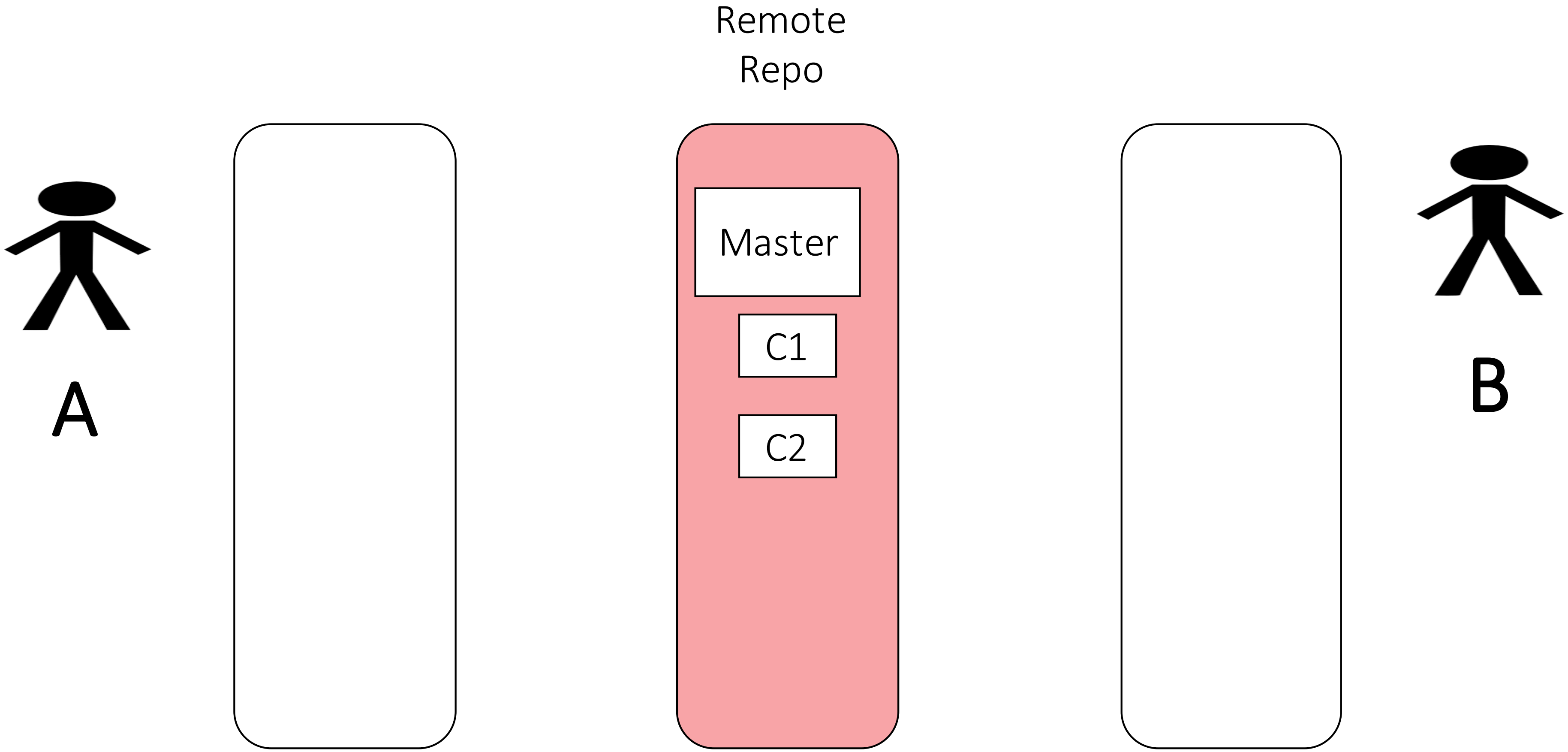
More Git Commands

- **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

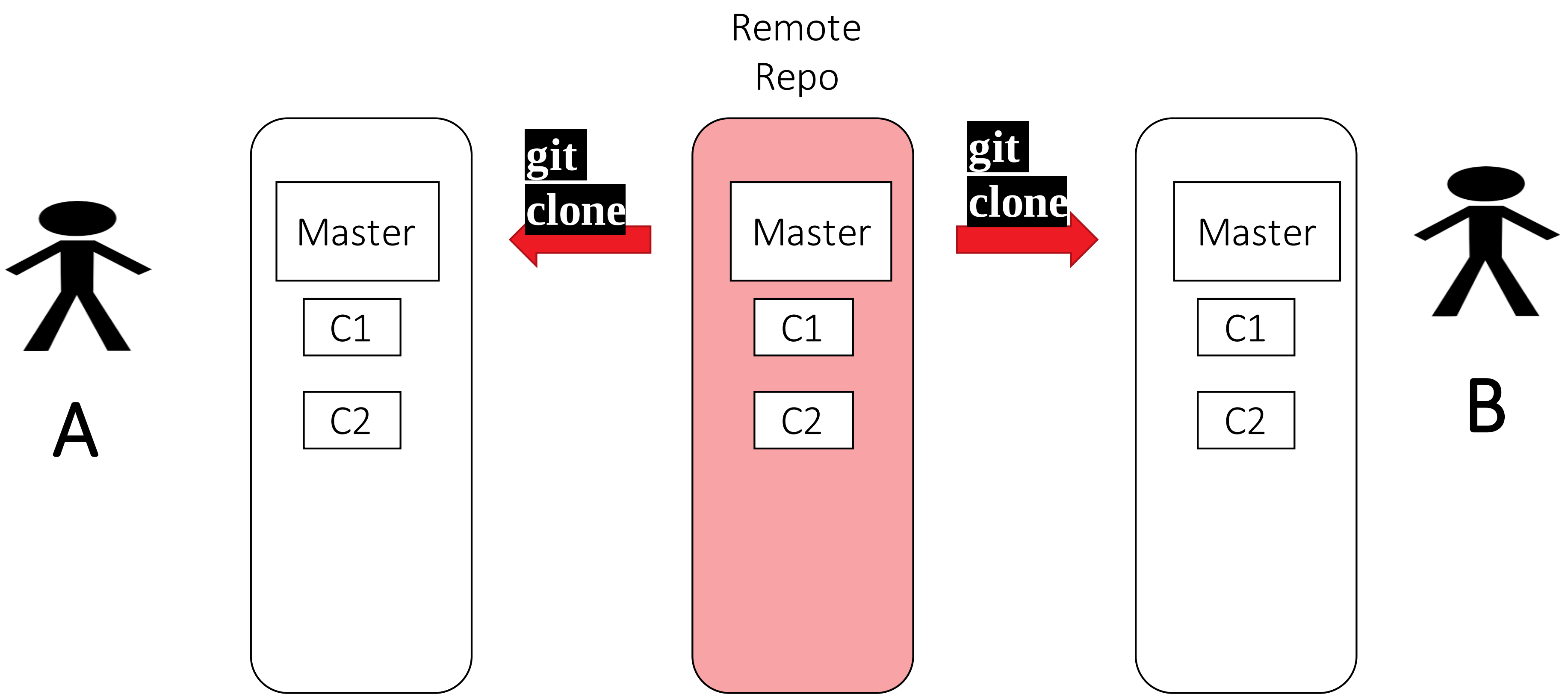
Git - Lifecycle



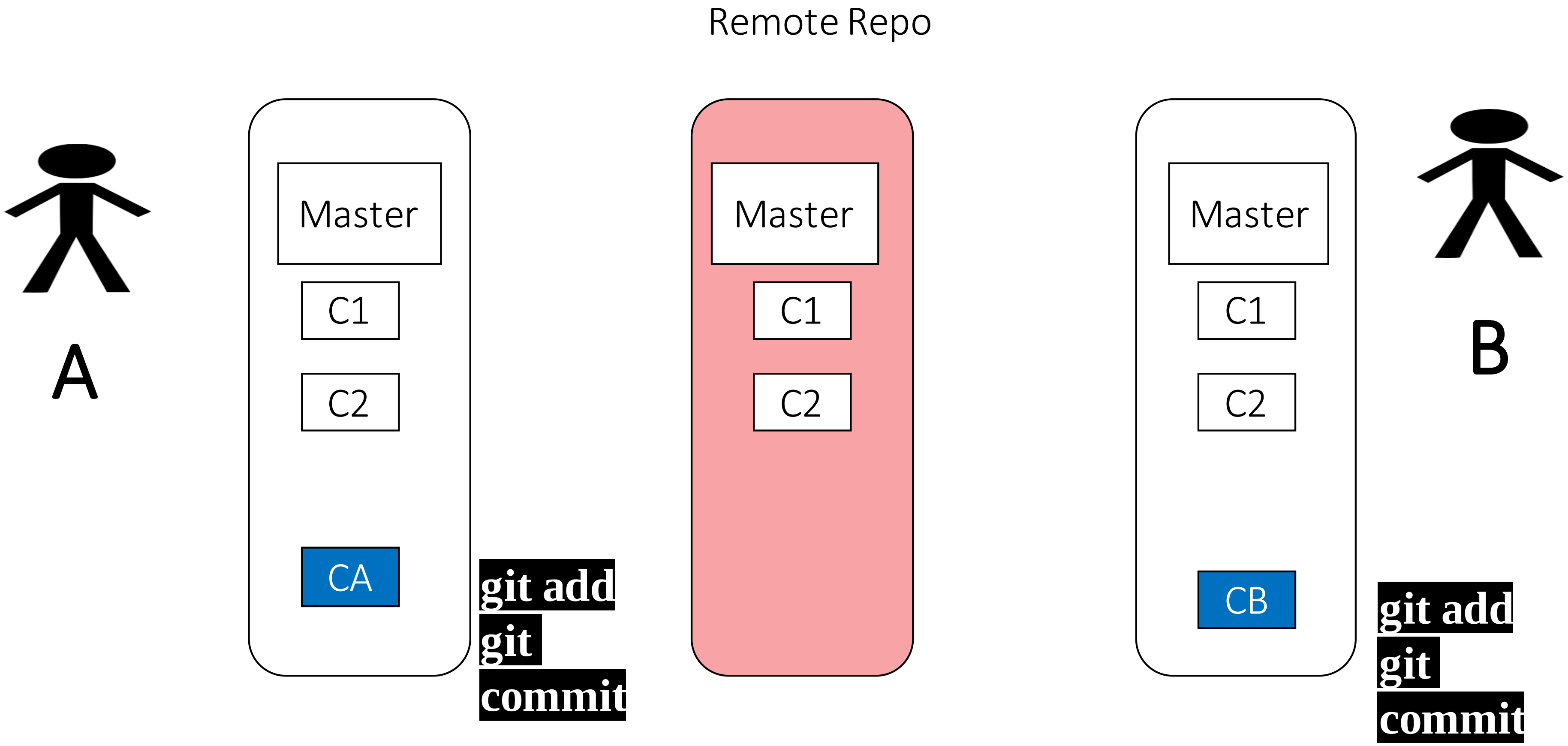
Collaboration With GitHub



Collaboration With GitHub



Collaboration With GitHub



Remote Repo

A

Master

C1

C2

CA

git add
git
commit

Master

C1

C2

Master

C1

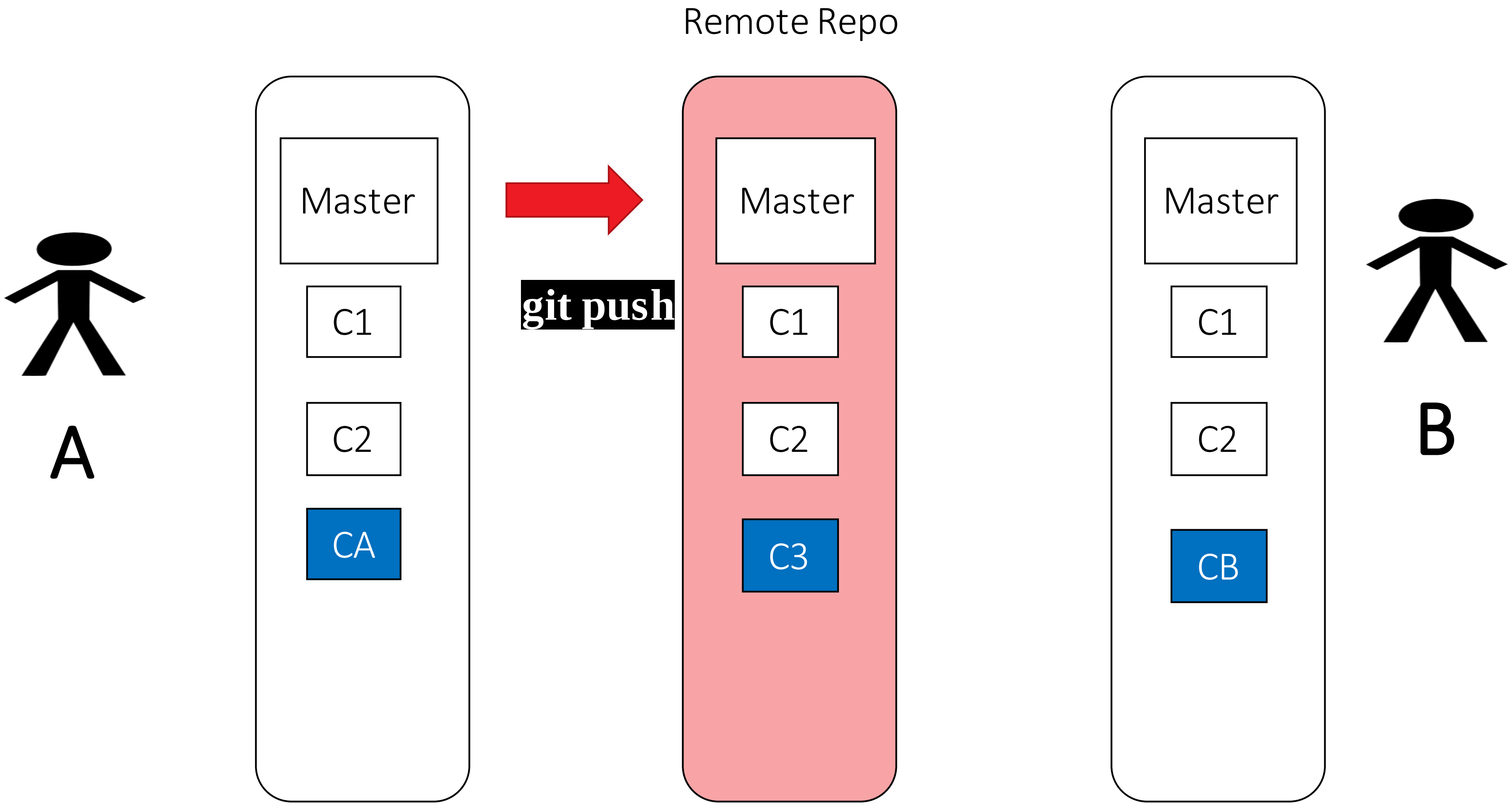
C2

CB

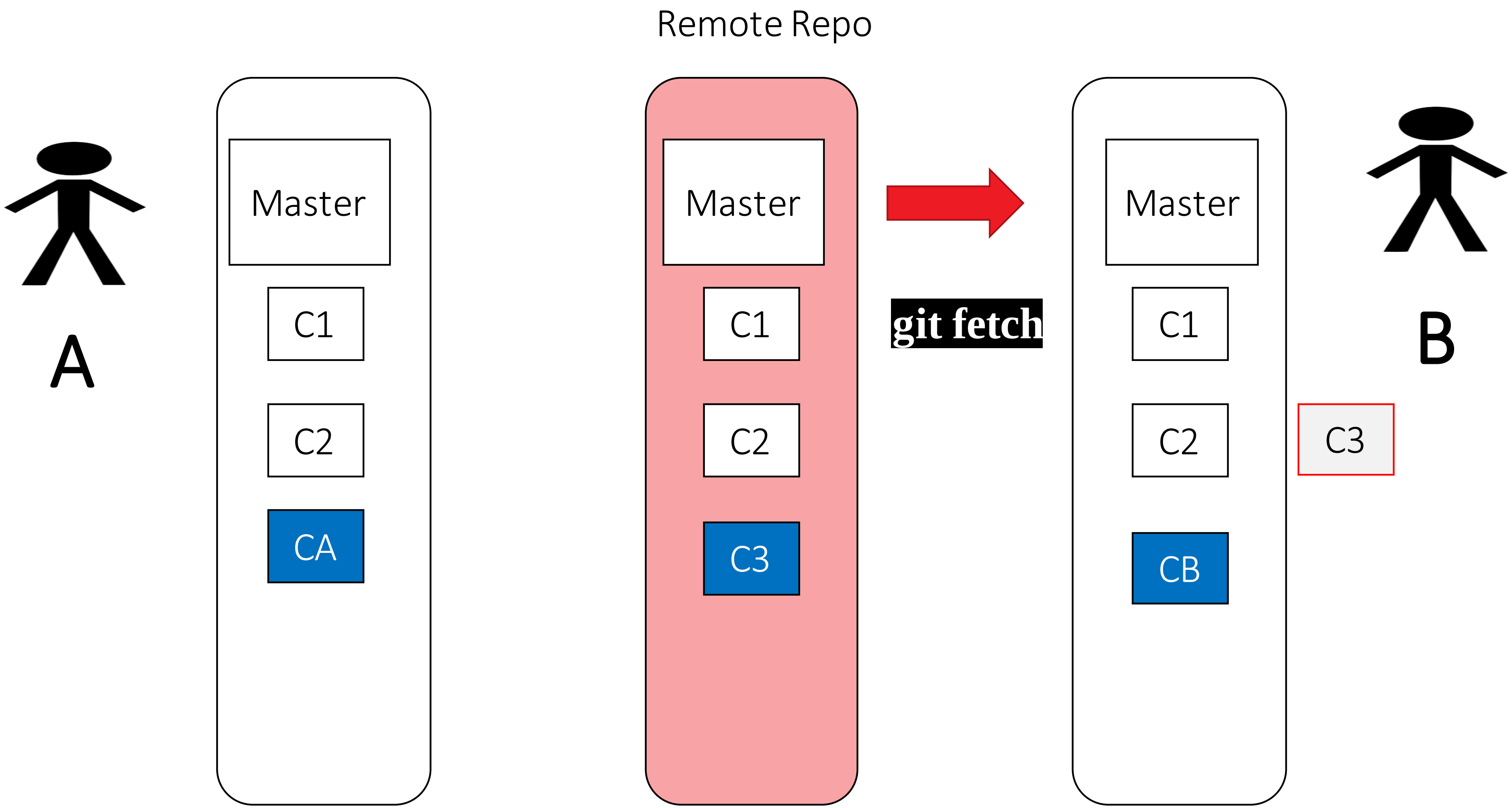
B

git add
git
commit

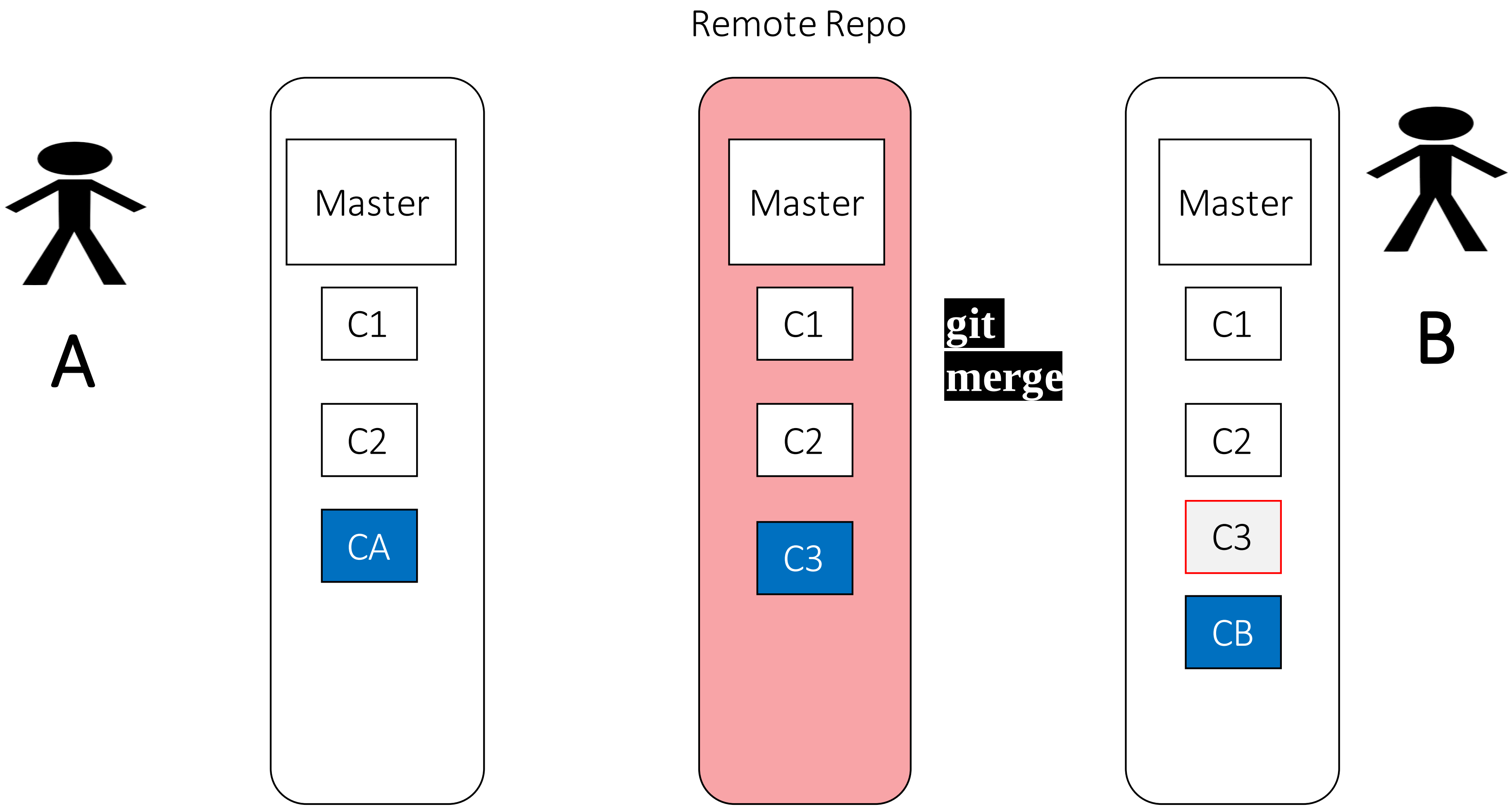
Collaboration With GitHub



Collaboration With GitHub



Collaboration With GitHub



Remote Repo

A

Master

C1

C2

CA

Master

C1

C2

C3

git merge

Master

C1

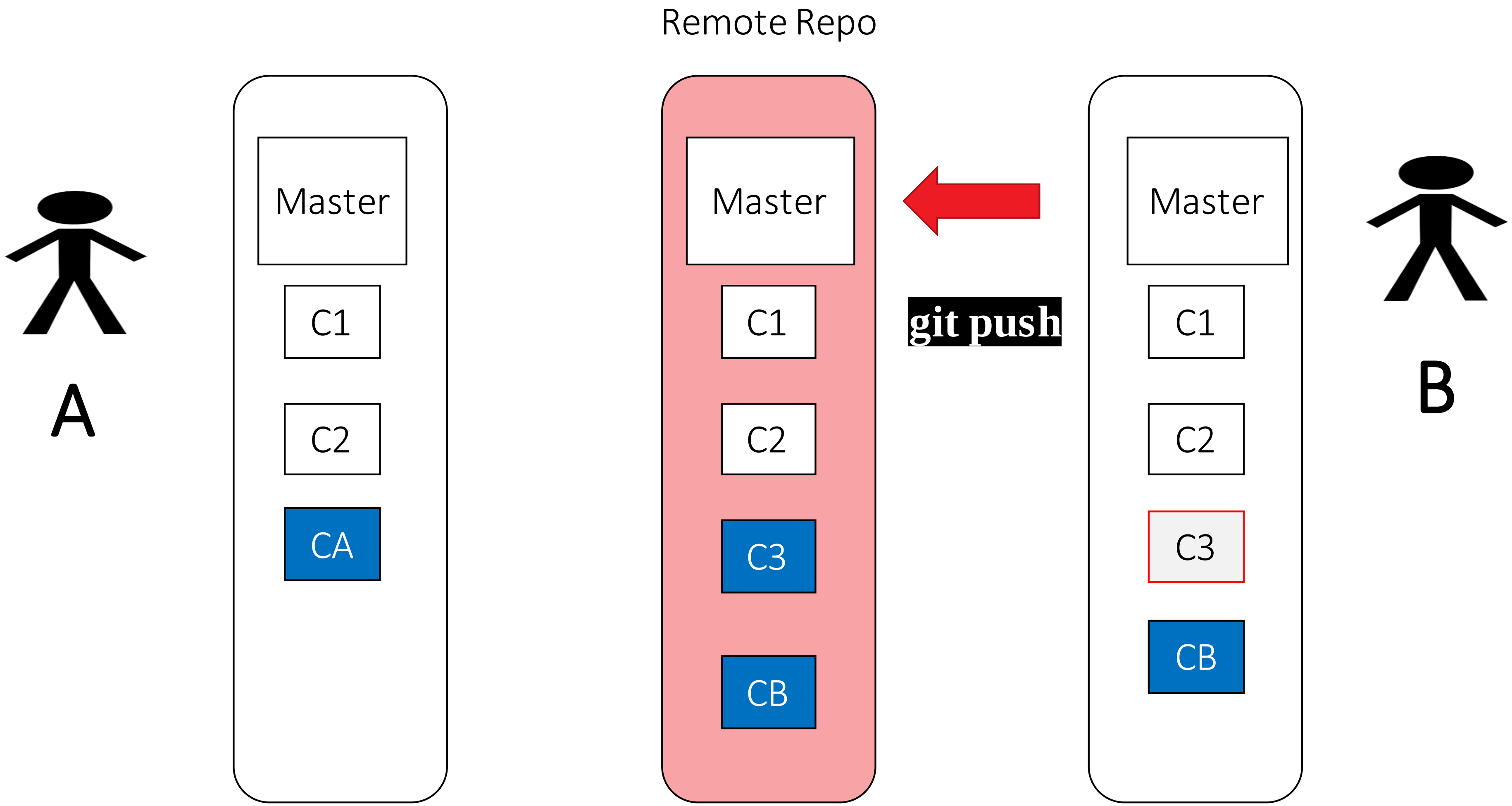
C2

C3

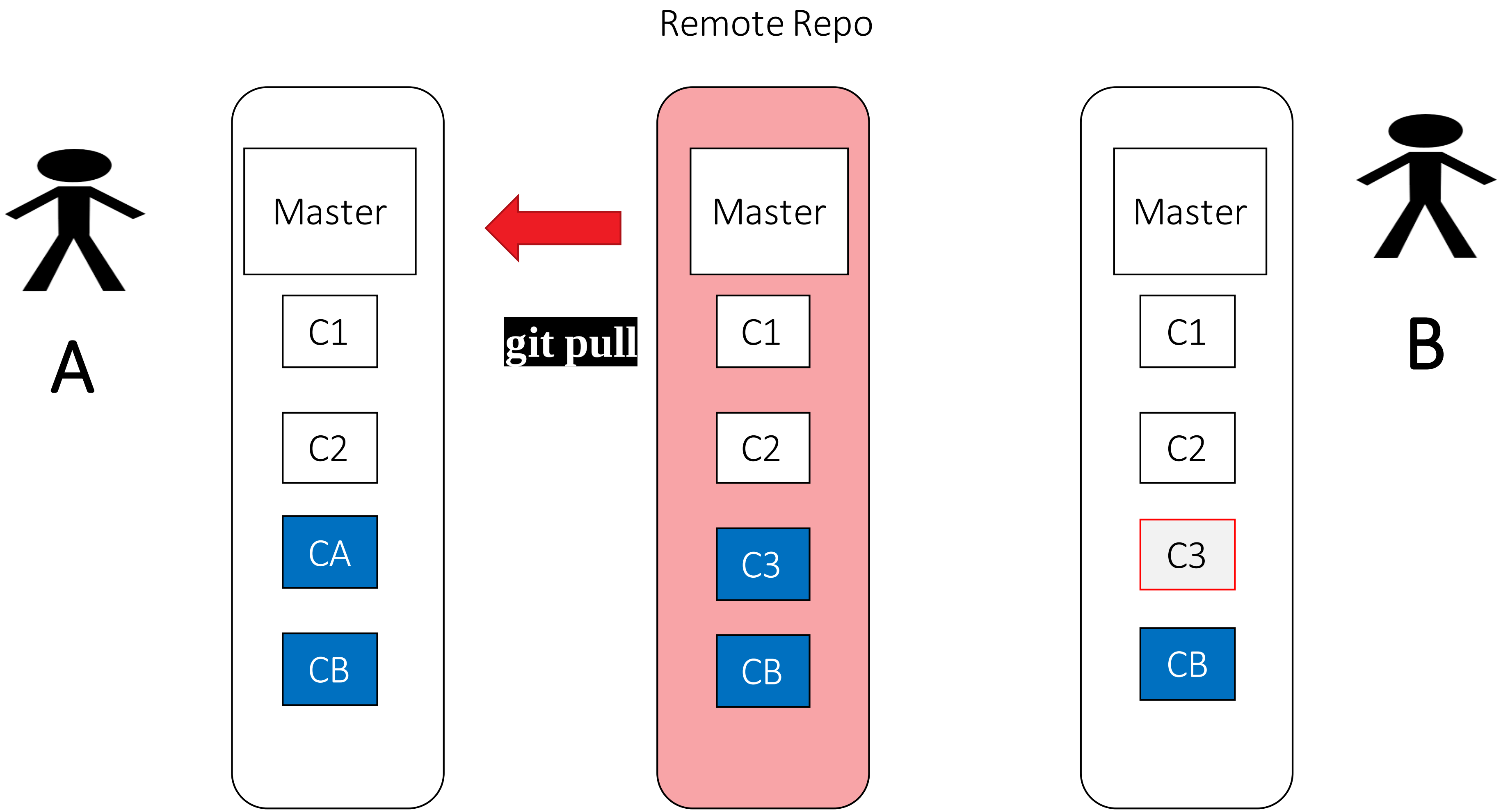
CB

B

Collaboration With GitHub



Collaboration With GitHub



Session Summary

- We discussed on Source Control fundamentals.
- We discussed on types of source control.
- We discussed on basic Git operations.
- We discussed on Git Staging and commit
- Git Branching
- Git Flow
- Git Hub Flow

Questions ?

Thank You

www.birlasoft.com

birlasoft

CK BIRLA GROUP