Version Control Software

15 SEP, 2020

RAJA SEKHAR TADEPALLI

AGENDA

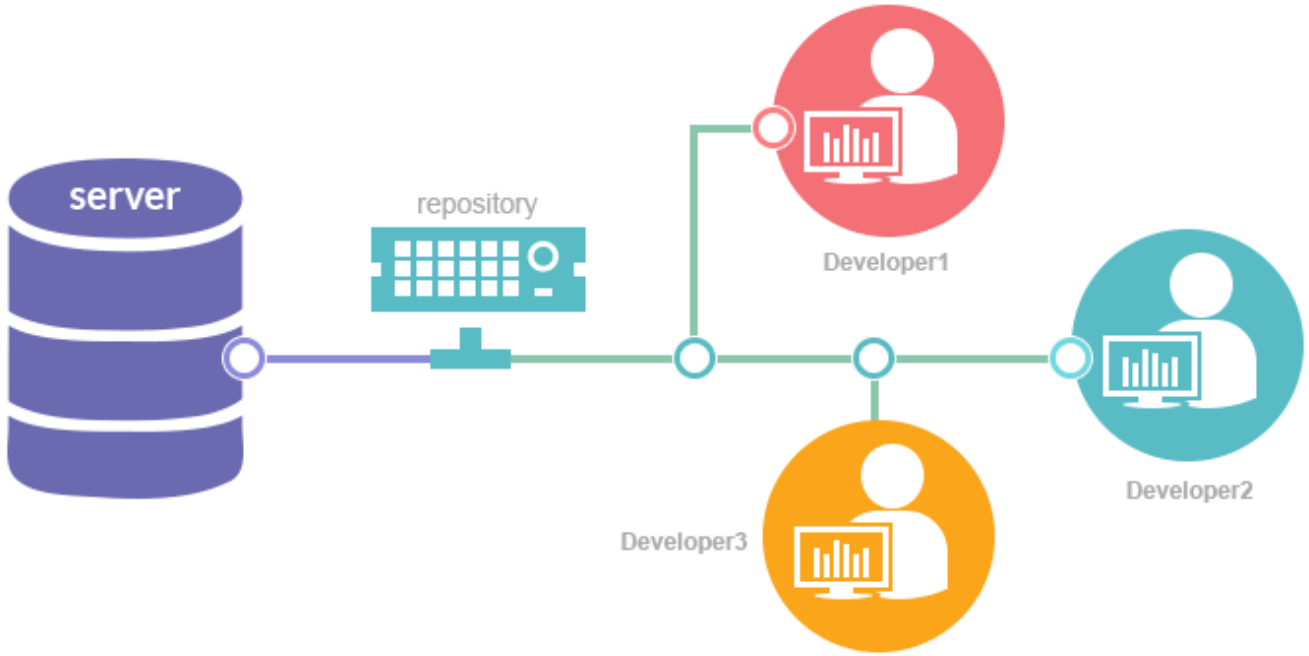
1. Introduction of Trainer
2. VCS
3. GIT
4. GIT Exercise & Commands
5. Q&A

TRAINER INTRODUCTION

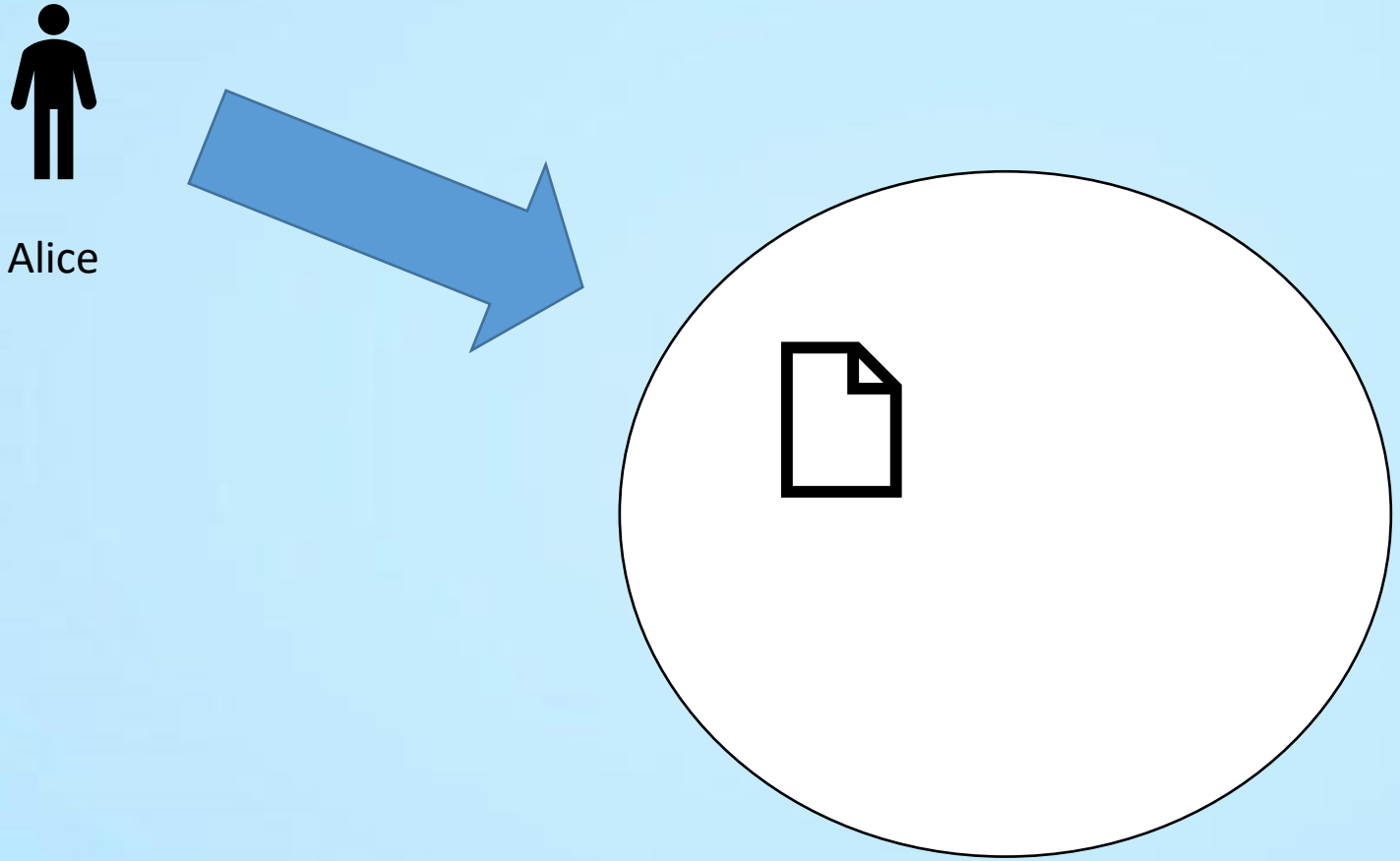
RAJA SEKHAR TADEPALLI

- Lead Resource Development Lab Head in Epam
- 12+ years software development experience.
- Developed various kinds of Applications using Microsoft Stack

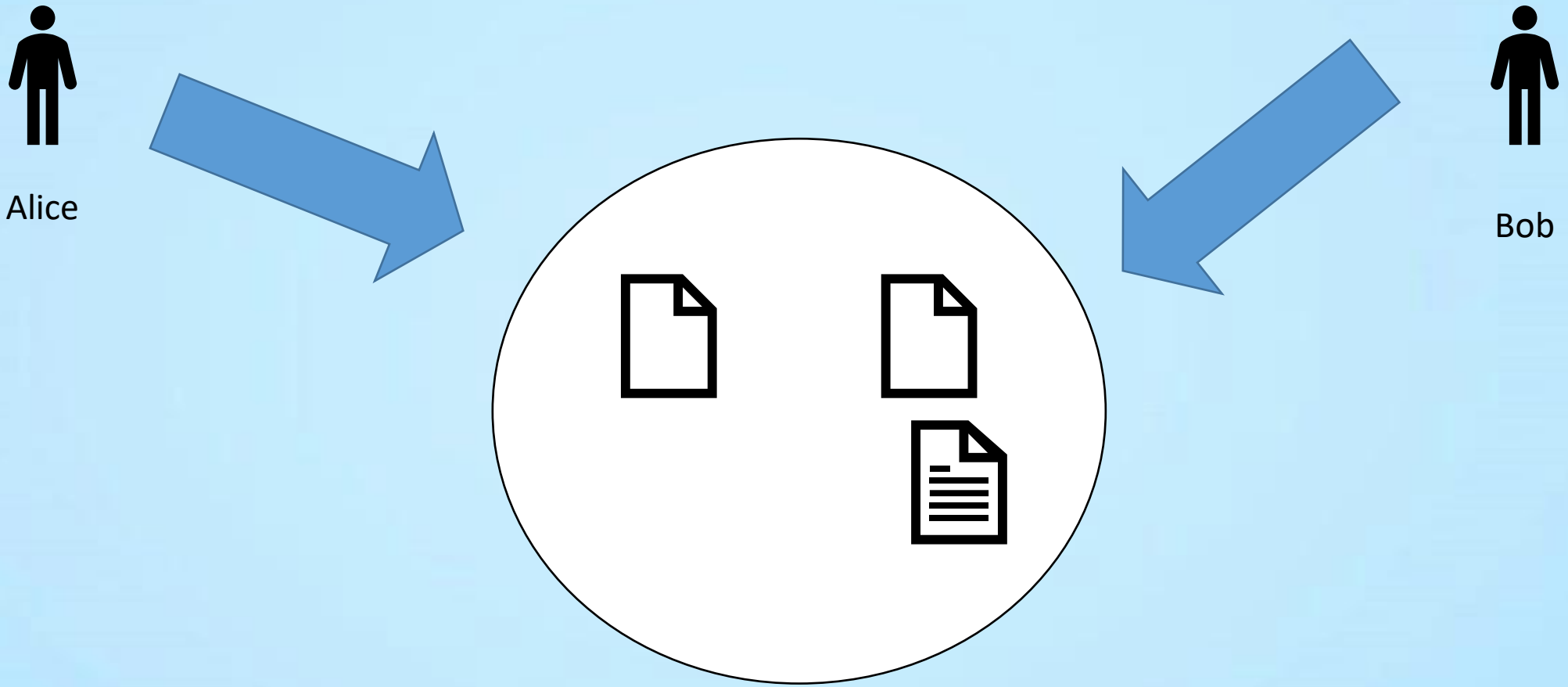
VERSION CONTROL SYSTEM (VCS)



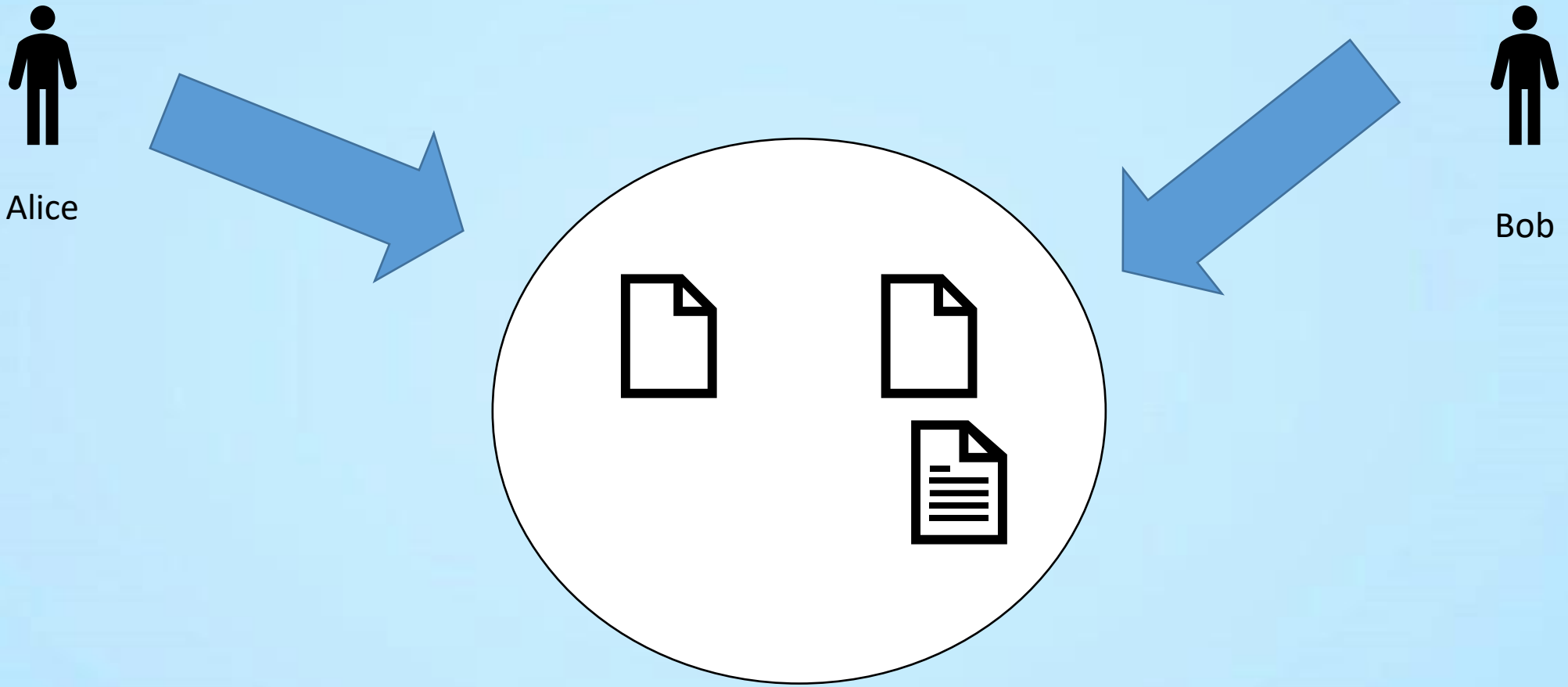
WHY DO WE NEED A VCS?



WHY DO WE NEED A VCS?



WHY DO WE NEED A VCS?

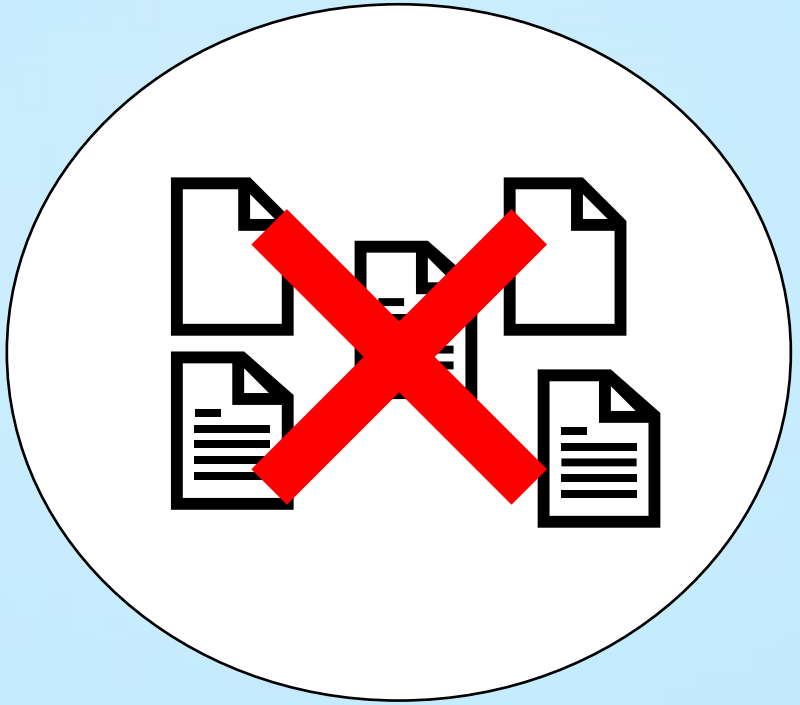


WHY DO WE NEED A VCS?

Alice

Bob

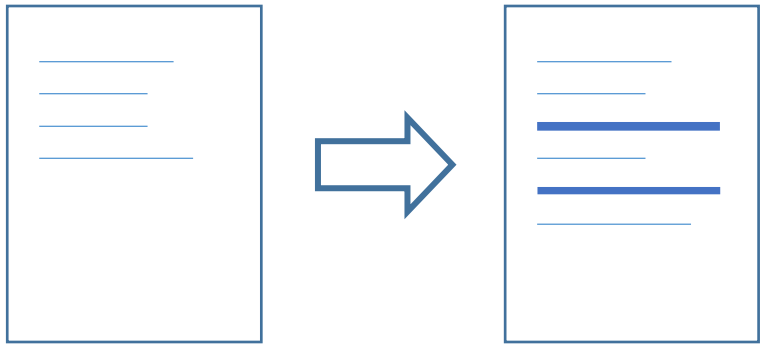
John



Who replaced the files? When ?

WHY DO WE NEED A VCS?

VCSs Track File Changes

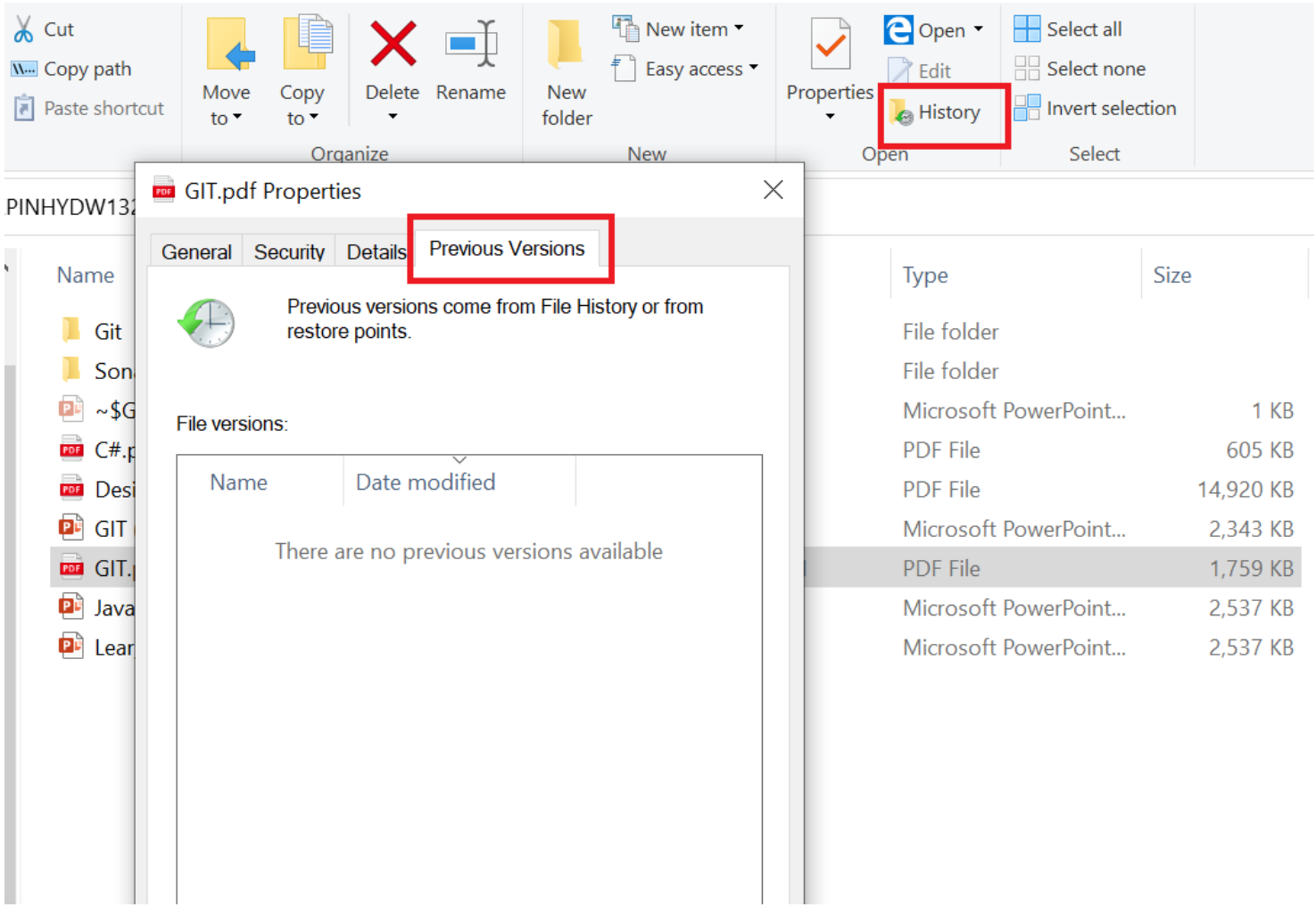


Code is organized within a repository.

VCSs Tell Us:

- Who made the change?
  - So you know whom to blame
- What has changed (added, removed, moved)?
  - Changes within a file
  - Addition, removal, or moving of files/directories
- Where is the change applied?
  - Not just which file, but which version or branch
- When was the change made?
  - Timestamp
- Why was the change made?
  - Commit messages

Basically, the Five W's



EXAMPLE OF WINDOWS FILE HISTORY

BRIEF HISTORY OF VERSION CONTROL SOFTWARE

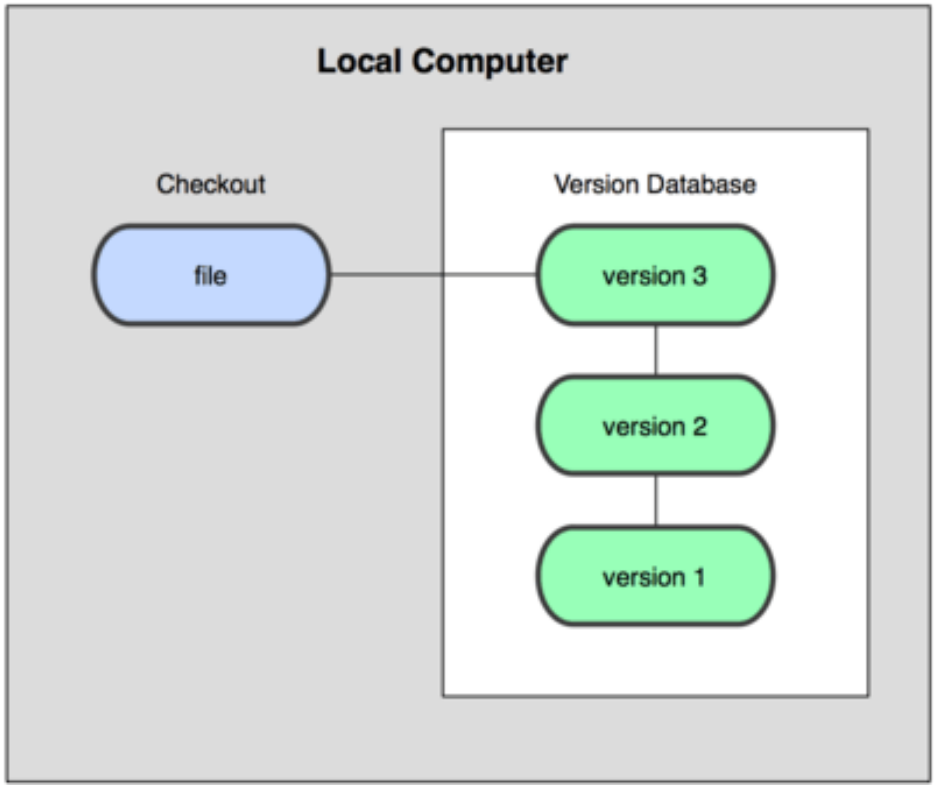
- First Generation – Local Only
  - SCCS – 1972
    - Only option for a LONG time
  - RCS – 1982
    - For comparison with SCCS, see this 1992 forum link
- Second Generation – Centralized
  - CVS – 1986
    - Basically a front end for RCS
  - SVN – 2000
    - Tried to be a successor to CVS
  - Perforce – 1995
    - Proprietary, but very popular for a long time
- Second Generation (Cont.)
  - Team Foundation Server – 2005
    - Microsoft product, proprietary
    - Good Visual Studio integration
- Third Generation – Decentralized
  - BitKeeper – 2000
  - GNU Bazaar – 2005
    - Canonical/Ubuntu
  - Mercurial – 2005
  - Git – 2005
  - Team Foundation Server – 2013

TYPES OF VCS

- Local Version Control System
- Centralized Version Control System (CVCS)
- Distributed/ Decentralized Version Control System(DVCS)

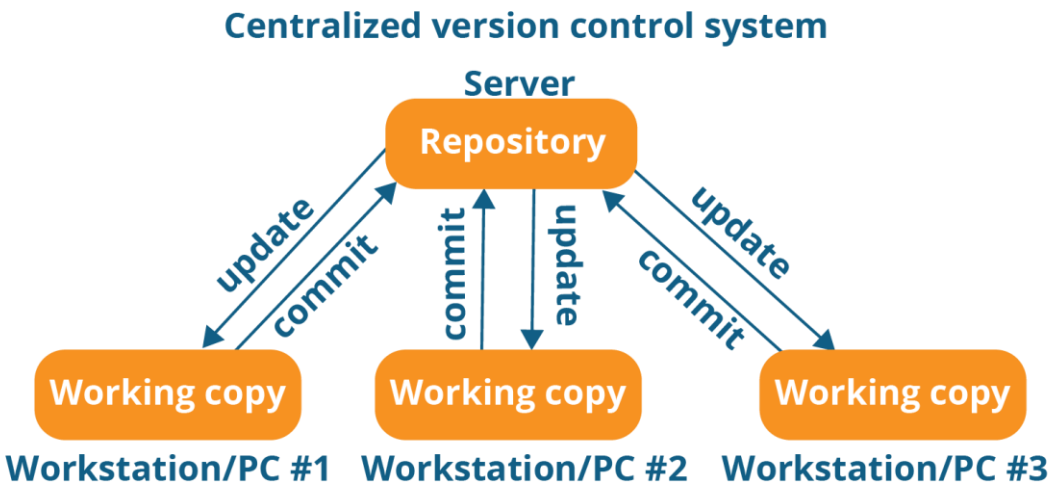
LOCAL VCS

- Visual Source Safe is a tool which track the versions but only on local machine. Basically I would like to check in and check out on my local machine when I am coding on my personal project from Home.



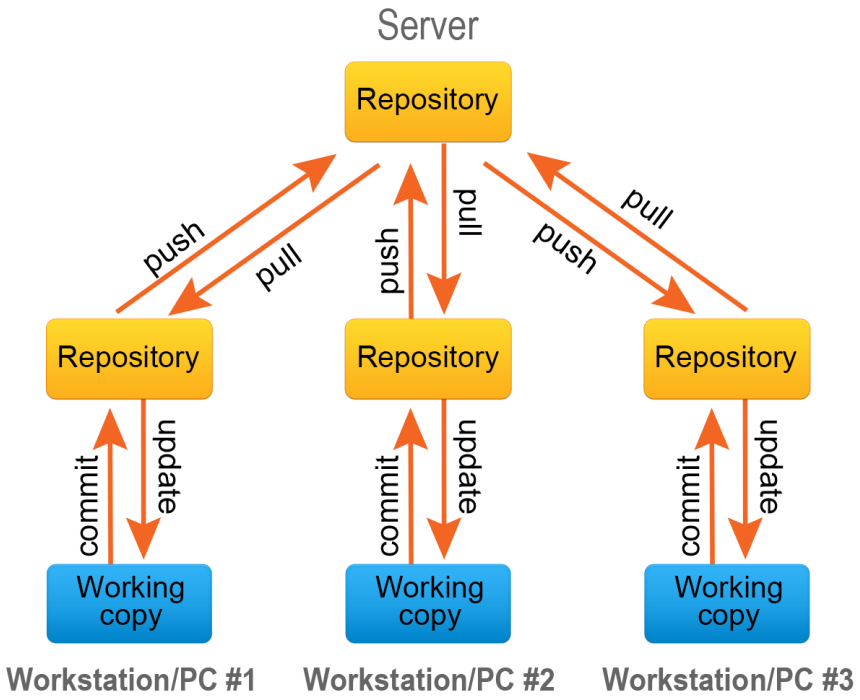
CENTRALIZED VCS

- In Subversion, CVS, Perforce, etc. A central server repository (repo) holds the "official copy" of the code
  - the server maintains the sole version history of the repo
- You make "checkouts" of it to your local copy
  - you make local modifications
  - your changes are not versioned
- When you're done, you "check in" back to the server
  - your check-in increments the repo's version

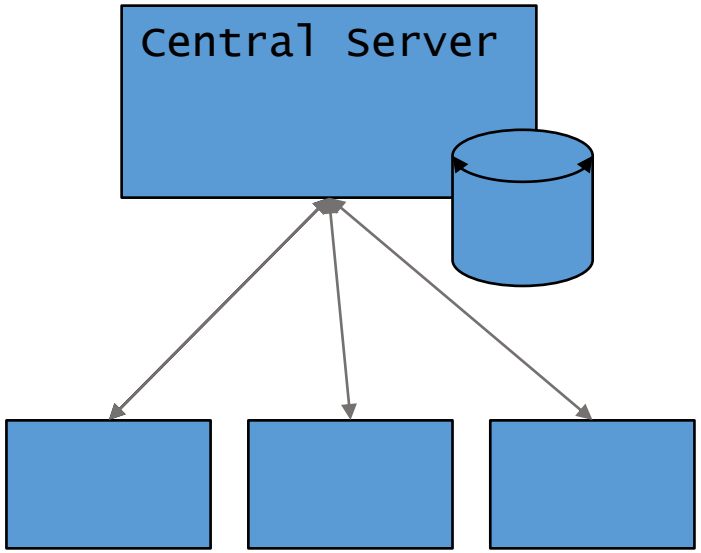


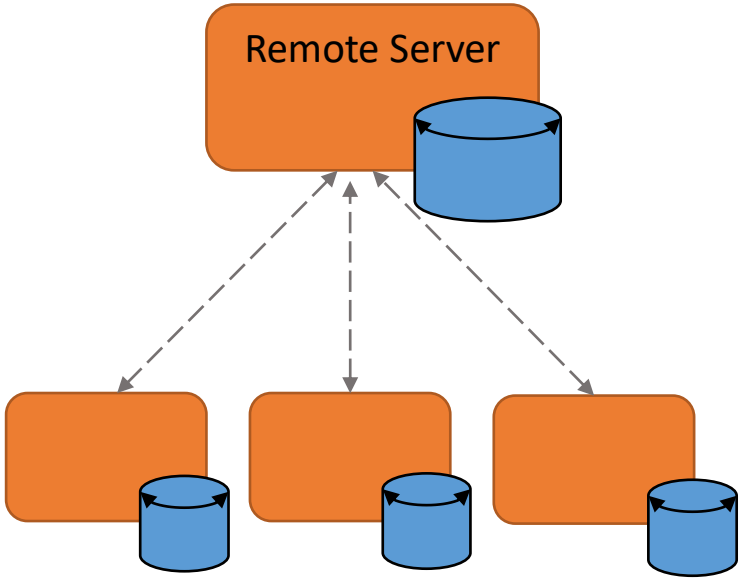
DISTRIBUTED VCS (GIT)

- In git, mercurial, etc., you don't "checkout" from a central repo
  - you "clone" it and "pull" changes from it
- Your local repo is a complete copy of everything on the remote server
  - yours is "just as good" as theirs
- Many operations are local:
  - check in/out from local repo
  - commit changes to local repo
  - local repo keeps version history
- When you're ready, you can "push" changes back to server



CENTRALIZED VC VS. DISTRIBUTED VC






git

WHAT IS GIT?

Git (/gɪt/)[7] is a distributed version-control system for tracking changes in source code during software development.[8] It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows[*clarification needed*] .[9][10][11]

GIT

- Created by Linus Torvalds, creator of Linux, in 2005
  - Came out of Linux development community
  - Designed to do version control on Linux kernel

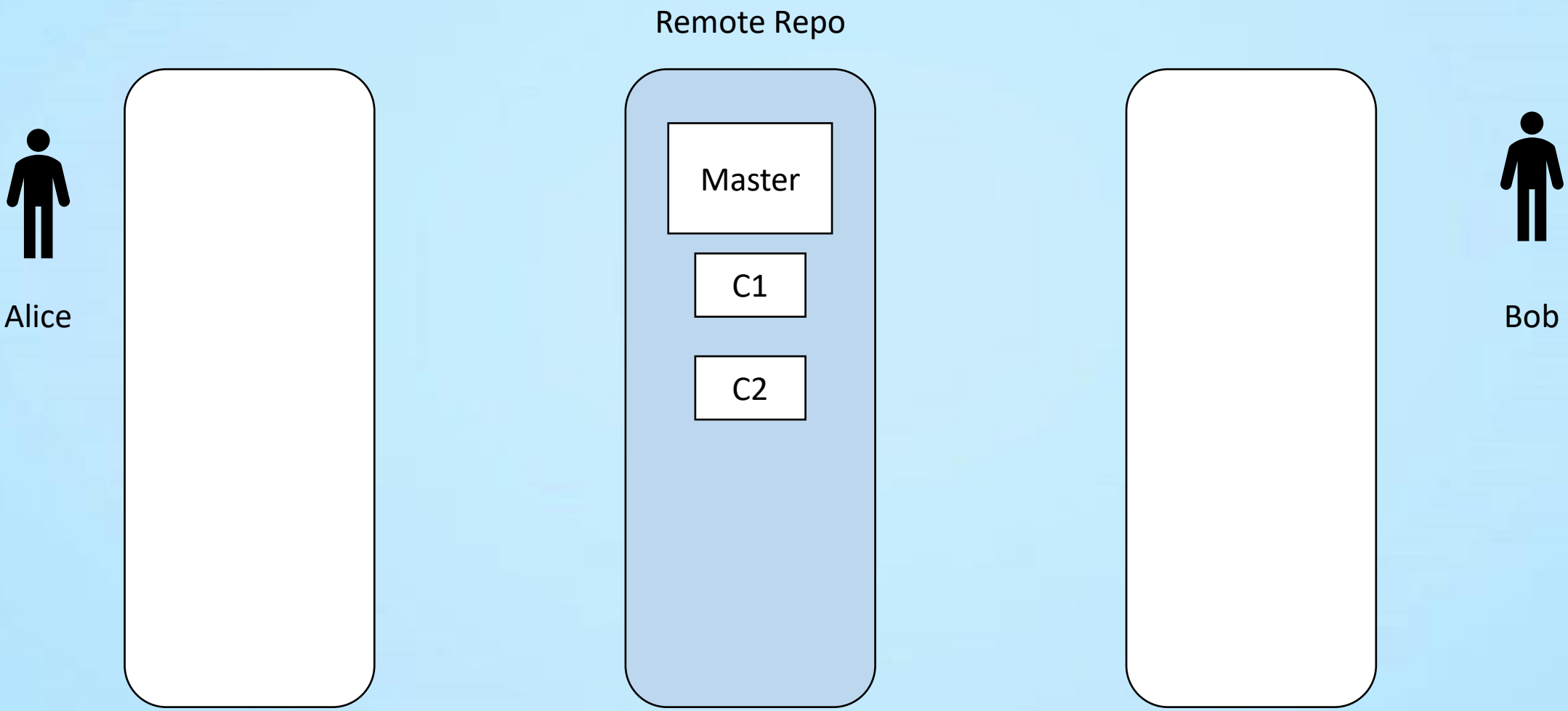
- Goals of Git:
  - Speed
  - Support for non-linear development (thousands of parallel branches)
  - Fully distributed
  - Able to handle large projects efficiently

INSTALLING/LEARNING GIT

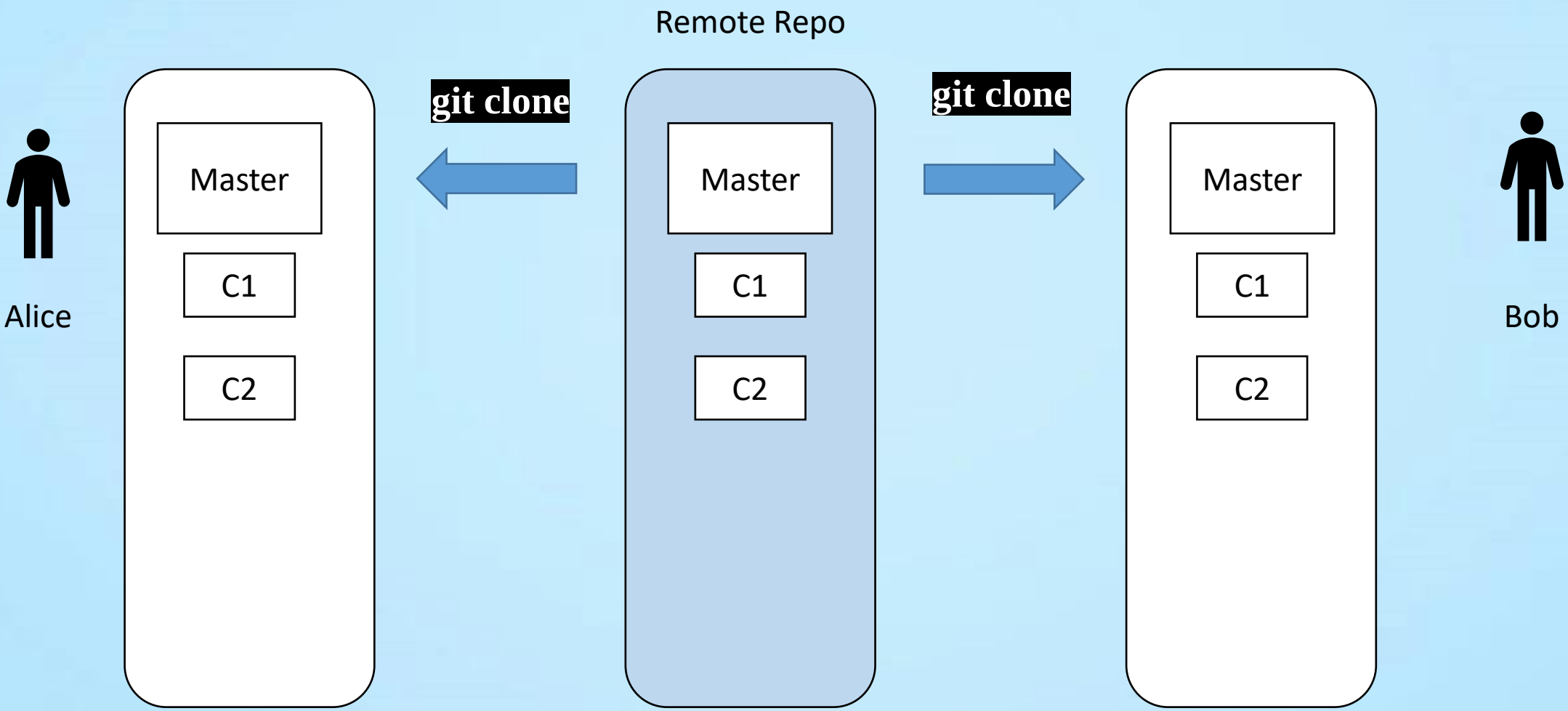
- Git website: http://git-scm.com/
- Free on-line book: http://git-scm.com/book
  - Reference page for Git: http://gitref.org/index.html
  - Git tutorial: http://schacon.github.com/git/gittutorial.html
  - Git for Computer Scientists: http://eagain.net/articles/git-for-computer-scientists/
- At command line: (where verb = config, add, commit, etc.) – git help verb


DEMO

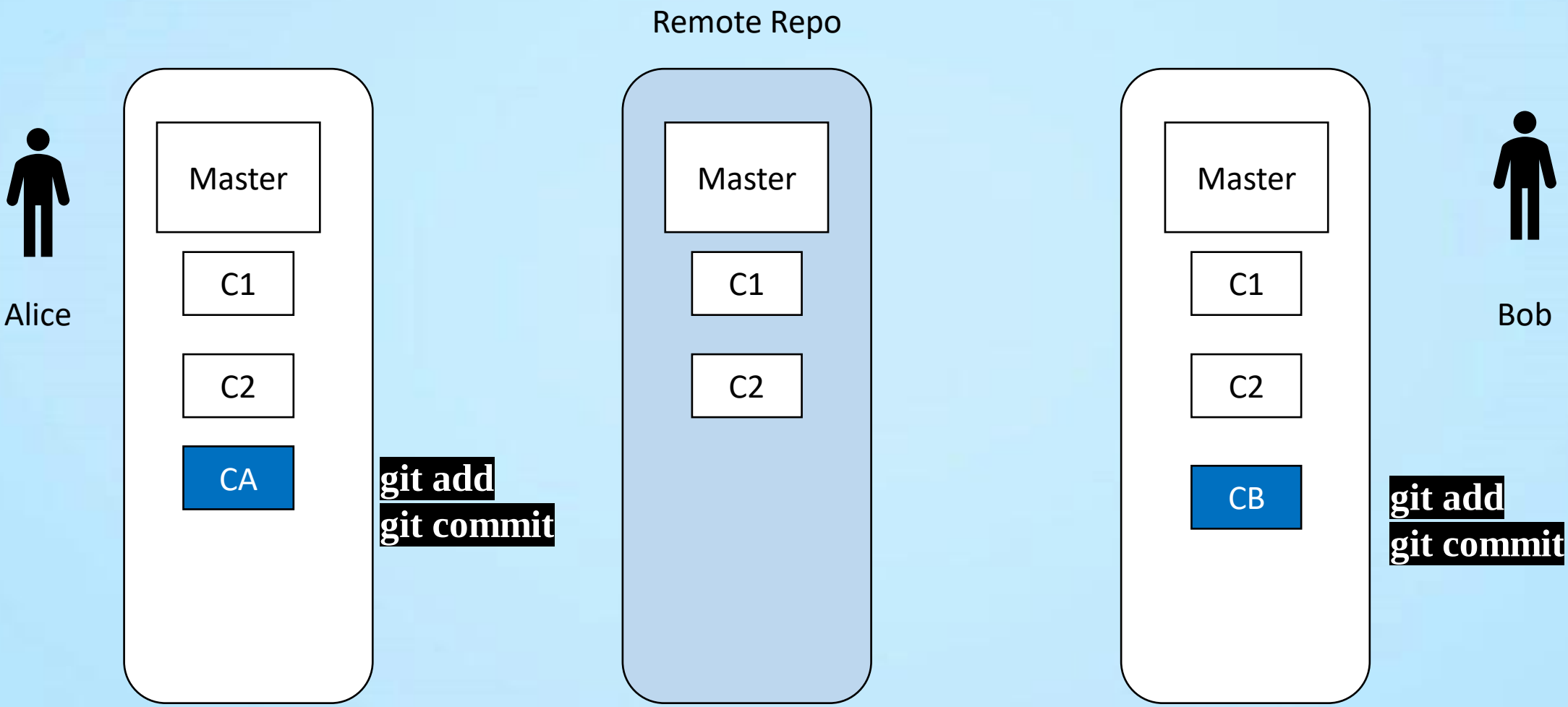
COLLABORATE



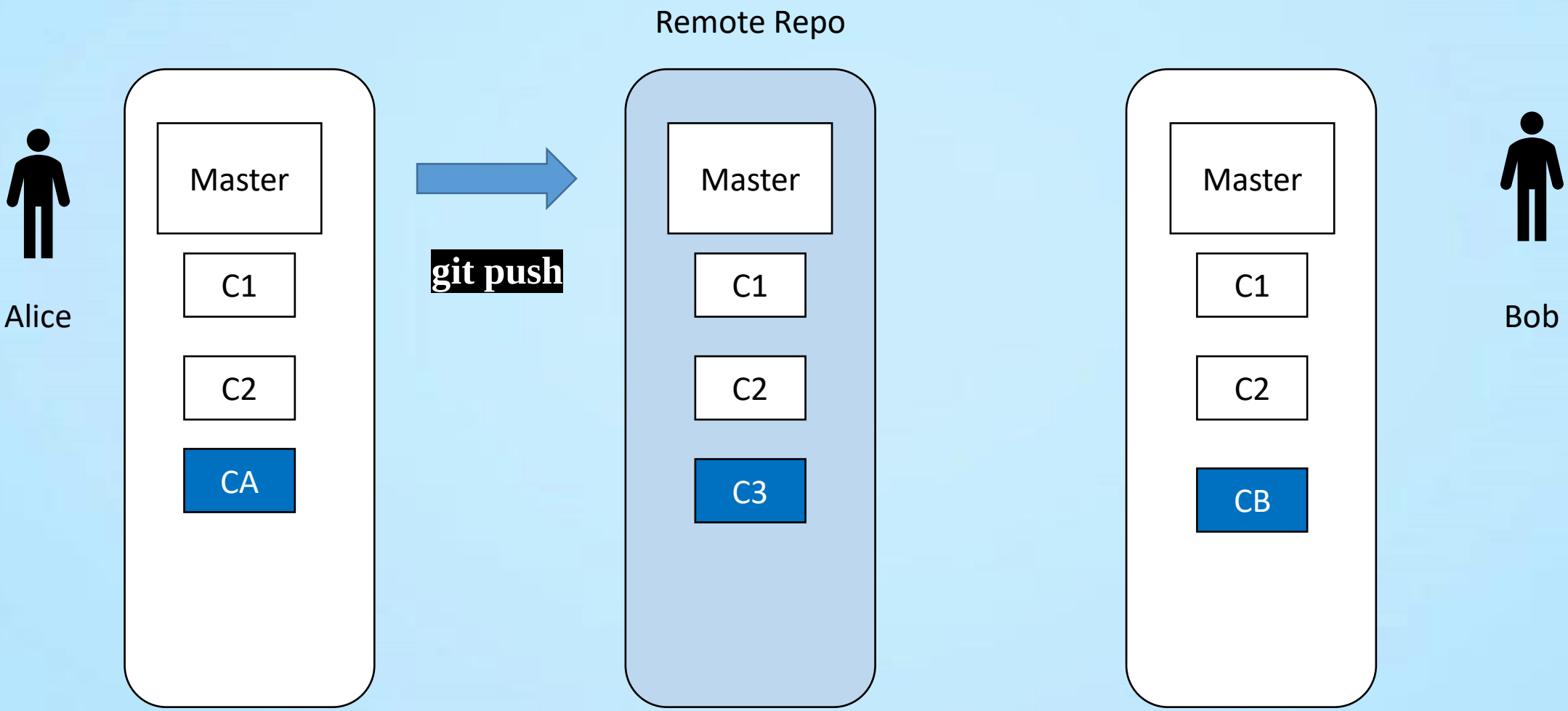
COLLABORATE



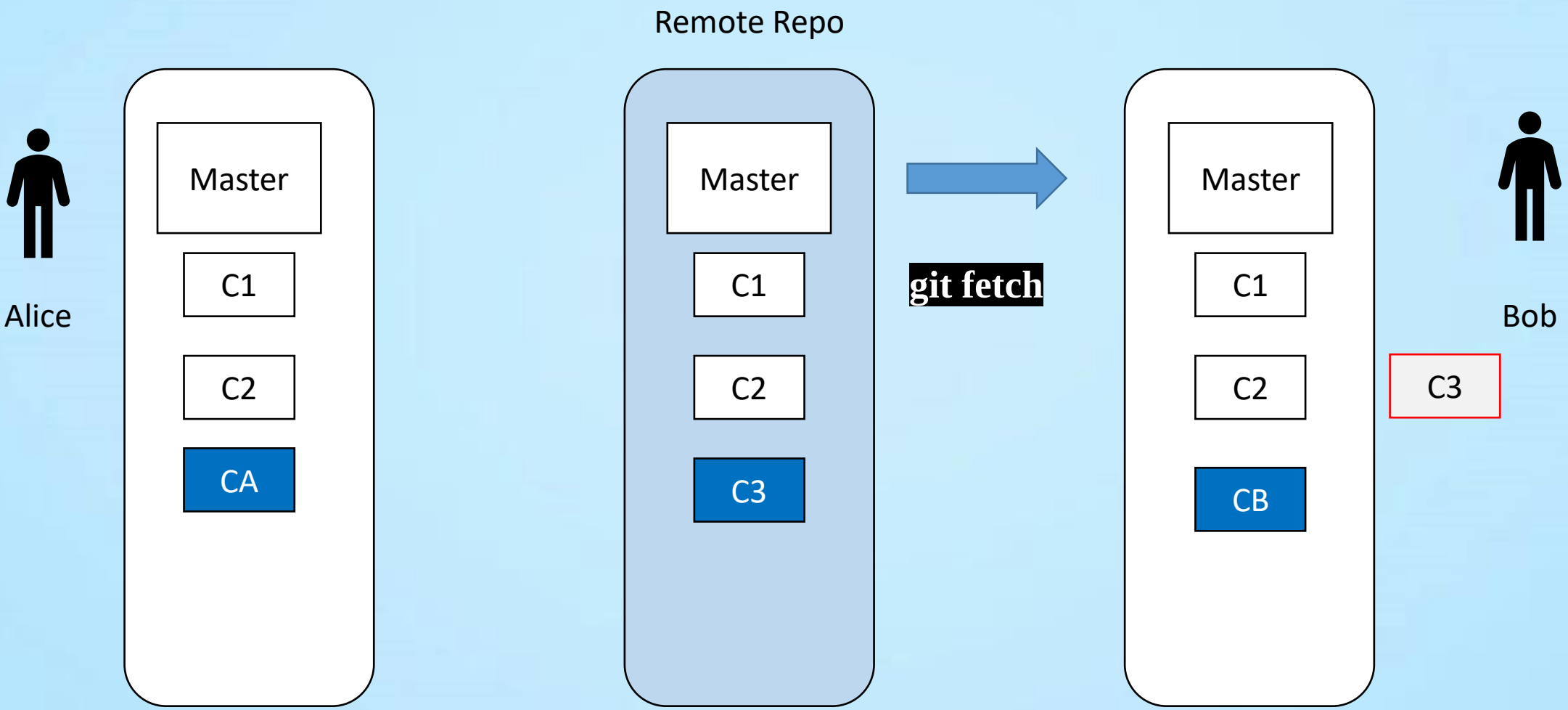
COLLABORATE



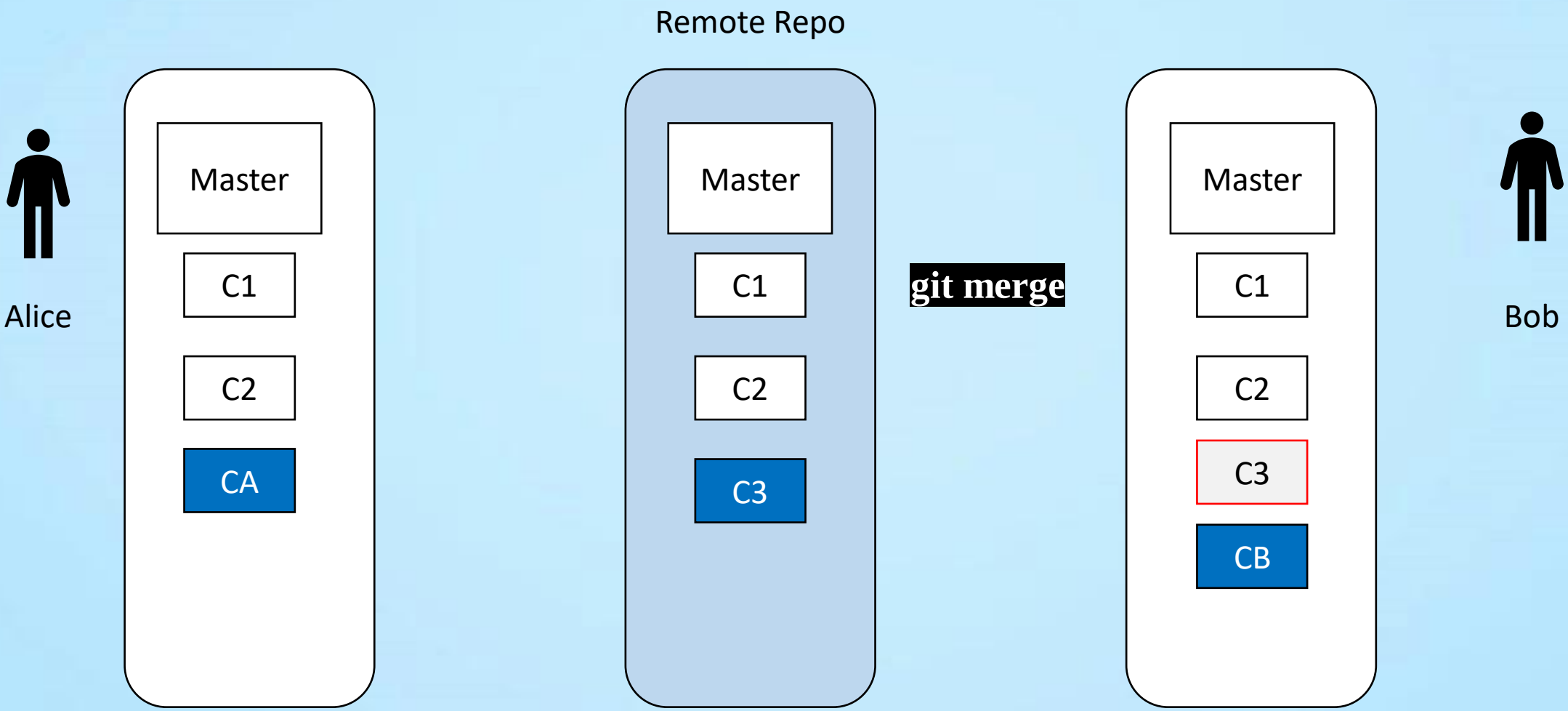
COLLABORATE



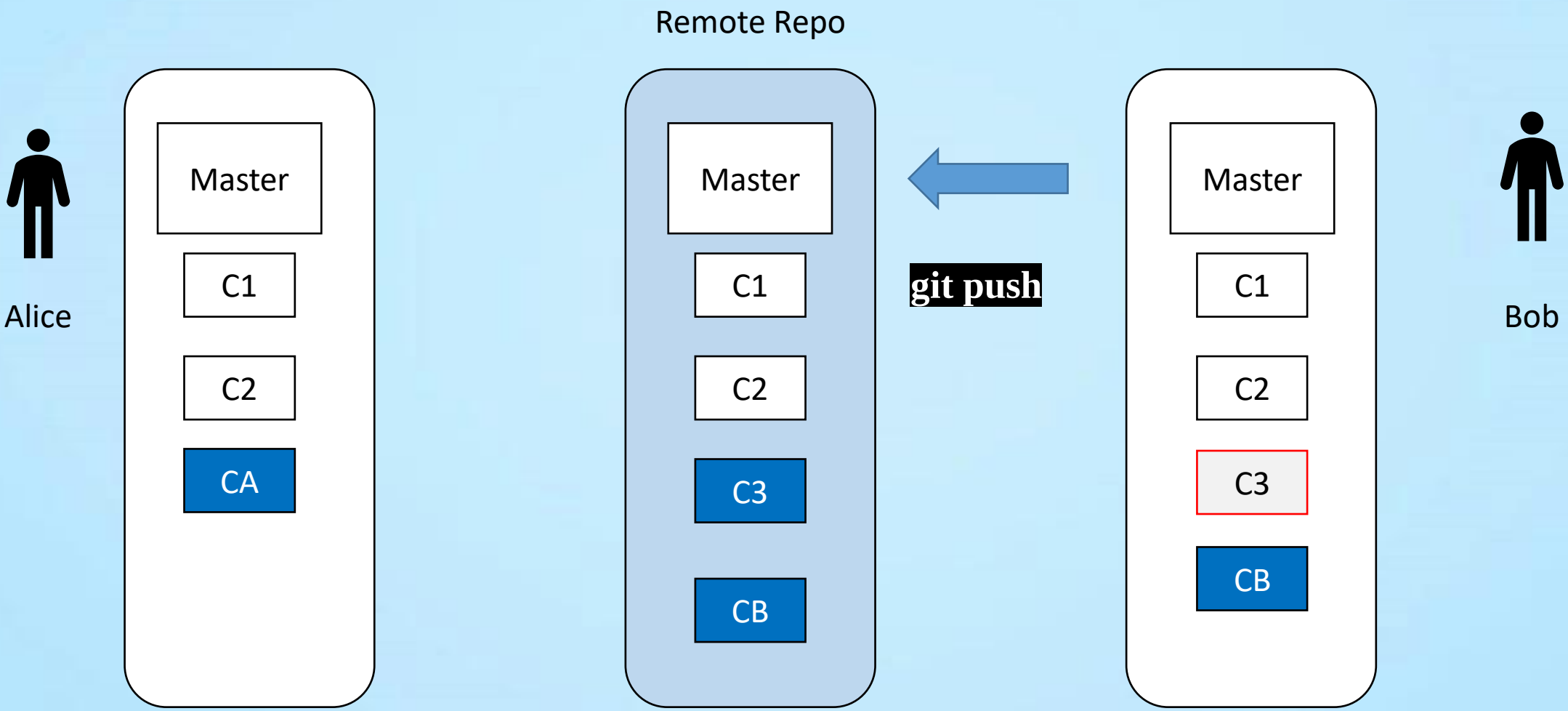
COLLABORATE



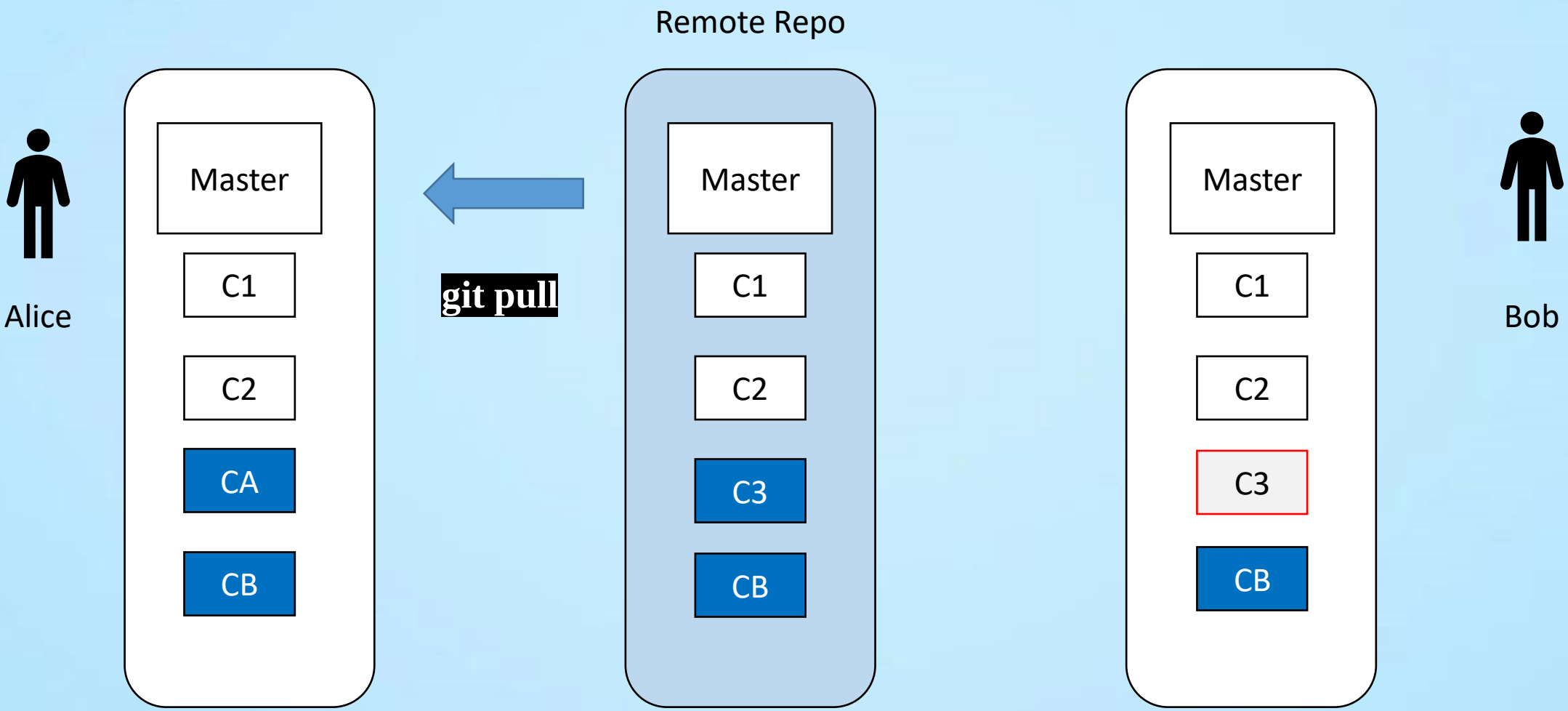
COLLABORATE



COLLABORATE



COLLABORATE



GIT COMMANDS

CREATE YOUR FIRST REPOSITORY

- Create a project folder in your local machine
  - `cd "Directory Name"`
- Then we initialize it as a Git Repository
  - git init

GIT Can Now Track the changes inside our project folder

COMMIT

- The "**commit**" command is used to save your changes to the local repository.

Note that you have to explicitly tell **Git** which changes you want to include in a **commit** before running the "**git commit**" command. This **means** that a file won't be automatically included in the next **commit** just because it was changed.

CREATE YOUR FIRST COMMIT

- First create a file **“hello.txt”** containing

  Hello

- Then run the following commands

```
$git add hello.txt
$git commit –m “Initial Commit”
```

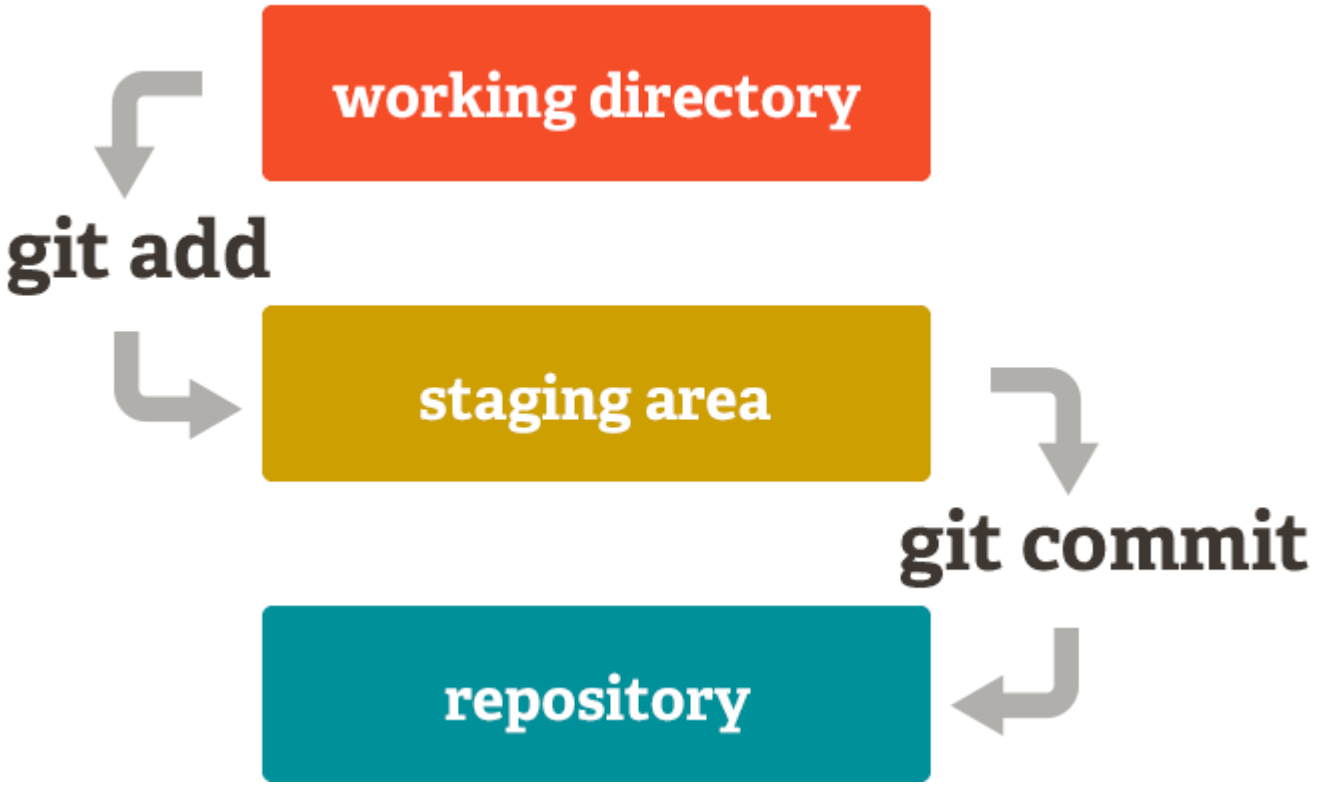
VIEW THE REPOSITORY

- View the repository history

```
$git log

$git log –graph –pretty = oneline
```

WHAT HAPPENED?



CREATE YOUR SECOND REPOSITORY

- Modify “hello.txt” to add “world”

Hello World

Then run the following commands

$git add Hello.txt

$git commit –m “Make hello.txt more exciting”

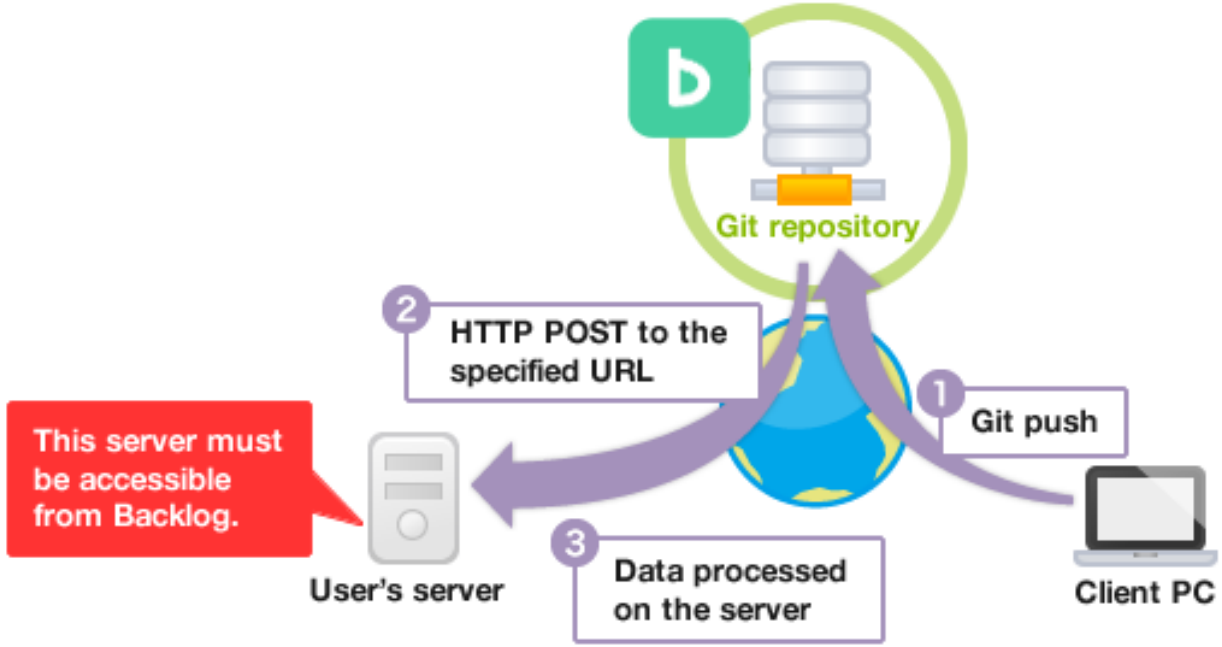
FILE STATUS

```
$ git status
# On branch master
nothing to commit, working directory clean
```

Interaction w/ remote repo

- **Push** your local changes to the remote repo.
- **Pull** from remote repo to get most recent changes.
  – (fix conflicts if necessary, add/commit them to your local repo)
- To fetch the most recent updates from the remote repo into your local repo, and put them into your working directory:
  – git pull origin master
- To put your changes from your local repo in the remote repo:
  – git push origin master

GIT


Git repository
2
HTTP POST to the specified URL
This server must be accessible from Backlog.
1
Git push
User's server
3
Data processed on the server
Client PC


me
myeclipse
ORACLE
JDEVELOPER
IJ


git


Bitbucket


GitLab

VIEWING/UNDOING CHANGES

- To view status of files in working directory and staging area:
  - `git status` or `git status -s` (short version)
- To see what is modified but unstaged:
  - `git diff`
- To see a list of staged changes:
  - `git diff --cached`
- To see a log of all changes in your local repo:
  - `git log` or `git log --oneline` (shorter version)

```
1677b2d Edited first line of readme
258efa7 Added line to readme
0e52da7 Initial commit
```

  - `git log -5` (to show only the 5 most recent updates), etc.

MERGE CONFLICTS

- The conflicting file will contain <<< and >>> sections to indicate where Git was unable to resolve a conflict:

```
<<<<<<< HEAD:index.html
<div id="footer"> todo: message here </div>
=======
<div id="footer">
    thanks for visiting our site
</div>
>>>>>>> SpecialBranch:index.html
```

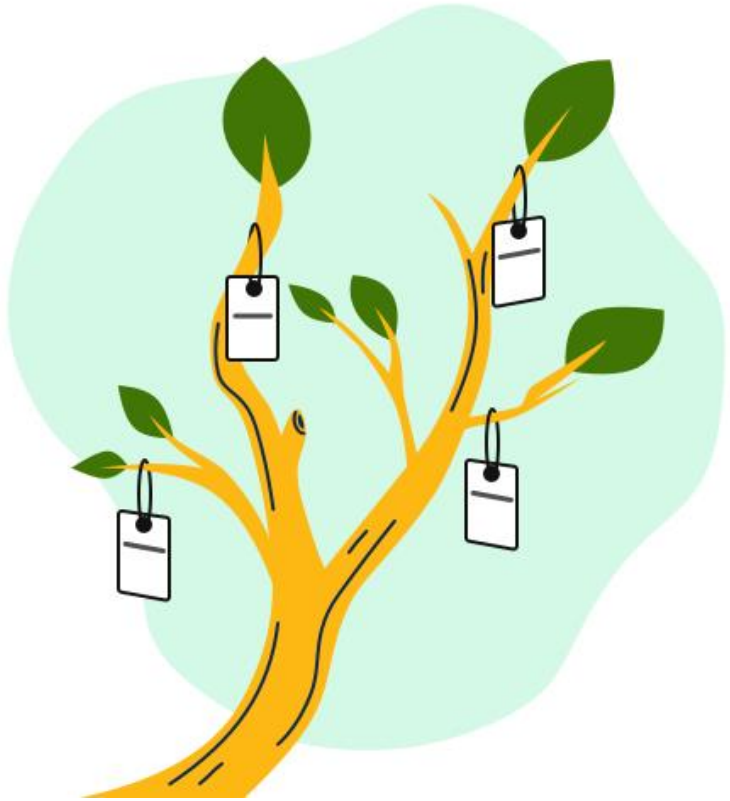
Branch 1's version

Branch 2's version

- Find all such sections and edit them to the proper state (whichever of the two versions is newer / better / more correct).

BRANCHING AND MERGING

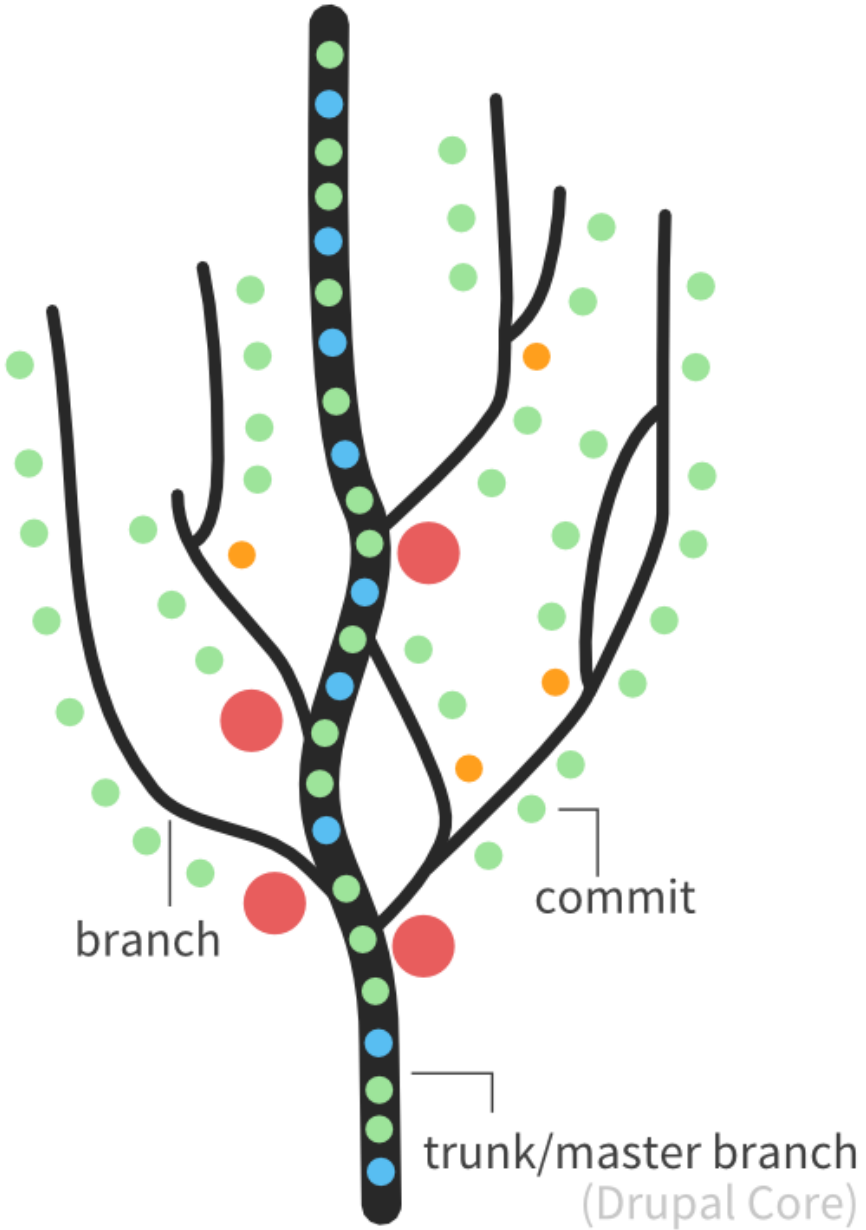
- Git uses branching heavily to switch between multiple tasks.
- To create a new local branch:
  - `git branch name`
- To list all local branches: (* = current branch)
  - `git branch`
- To switch to a given local branch:
  - `git checkout branchname`
- To merge changes from a branch into the local master:
  - `git checkout master`
  - `git merge branchname`



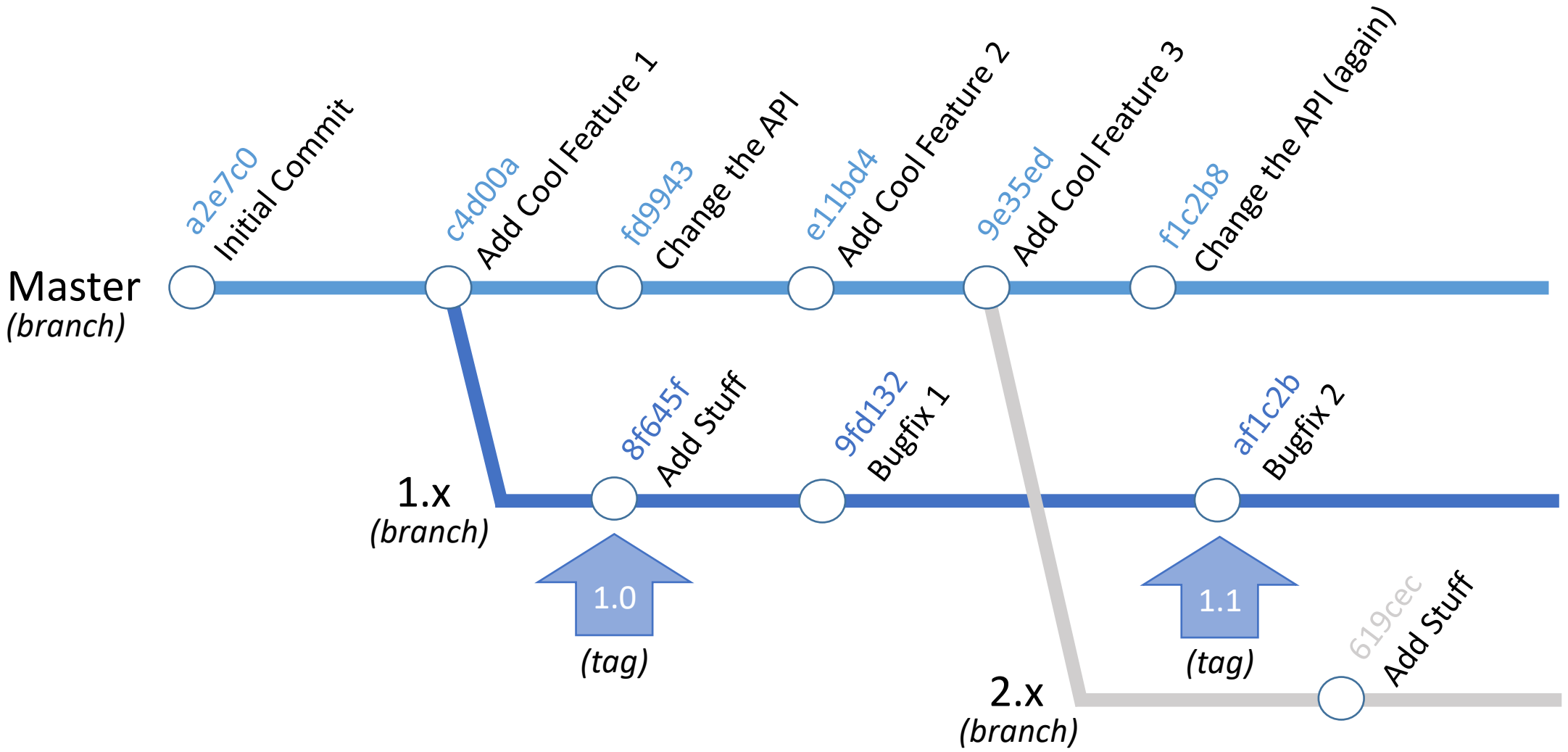
BRANCHES AND COMMITS

With this growing tree in mind, the branches on the tree would correspond to... that's right... branches in Git. The only difference is that in Git, you can decide when and where new branches split off and also branches can merge together in Git. The red paper tags at the start of each branch allows you to identify that particular branch. The green-colored tags at the ends of branches are known in Git as tags.

To continue the analogy (and translate it into software) each branch on the tree corresponds to a different major version of Drupal core (for example 6.x or 7.x). As bugs are fixed in a given release, the branch where the changes are committed would be growing longer (more revisions on that branch). Whenever a release is made, we take the most recent changes of the files (the revisions on the end of the branch) and add a tag to identify it (a marker with a word on it, in this case, based on the version number of the release) and tie it around the end of the branch. Therefore, for a contributed Drupal module, the tag at the base of the branch might be 7.x-1.0, but at the end of the branch it would be a higher number such as 7.x-1.9.



BRANCHES, COMMITS, AND TAGS, OH MY!



BEST PRACTICES

BEST PRACTICES

- Write good commit messages Otherwise, the commit log is useless.
  - You should be able to get an idea what changed and why just by looking at the commit messages
- Commit related changes
  - This would make it easier to rollback changes. Use a separate branch for each new bug/feature request.
- Commit often
- Do not commit sensitive information
  - Passwords, settings, database dumps etc.
- Avoid publishing half-done work as it can lead to broken builds. If you need to push those changes (e.g. you want a backup), put them in a branch then push that branch. Or consider using git stash.

BEST PRACTICES

- Communicate! Version control systems aims to improve communication between team members, not replace it. Always consult with the other person whenever you encounter a merge conflict.
- Do not commit commented-out debug code.
    - It’s messy. It’s ugly. It’s unprofessional
- Do not commit large binaries.

GIT ECLIPSE INTEGRATION

GIT ECLIPSE INTEGRATION

1. Create GIT Repository
2. Eclipse – Go to Perspective and Clone the Project
3. Create Project in Eclipse
4. Team => Share => Add to Git Repository
5. Commit and Push the Changes to GitHub.


REGENT STREET W1
Superdry
T-SHIRT GALLERY OPEN
Superdry
LEICESTER ARMS

THANK YOU!