



Prifysgol Cymru

Y Drindod Dewi Sant

University of Wales

Trinity Saint David

Trawsnewid Addysg, Trawsnewid Bywydau
Transforming Education, Transforming Lives

Version Control System

Network Programmability
(ACCB5029)

Applied Computing - WISA

Previously,

- ✓ Overview
- ✓ Review Linux and Python Skills
- ✓ Devnet Developer Environment
- ✓ Software Development & Design

This week,

- ✓ Implement software version control using GIT

Previous week's Tutorial and Independent learning

- Setup Lab environments
- Review Linux and Python
- Review Python installation, PIP, and Python virtual environments

Previously, we discussed about Devnet skills, and course work briefing. Followed by the last week's discussion on Software Development and Design. I hope you had a clarity on Software Development life cycle and its development methodologies. You must be able to identify various Devnet resources, navigate through basic Linux commands and python programming. This week, lets focus on Version Control Systems and discuss in detail about it, in addition to Git and GitHub.

✓ This week's Tutorial and Independent learning

- Review

✓ Independent learning

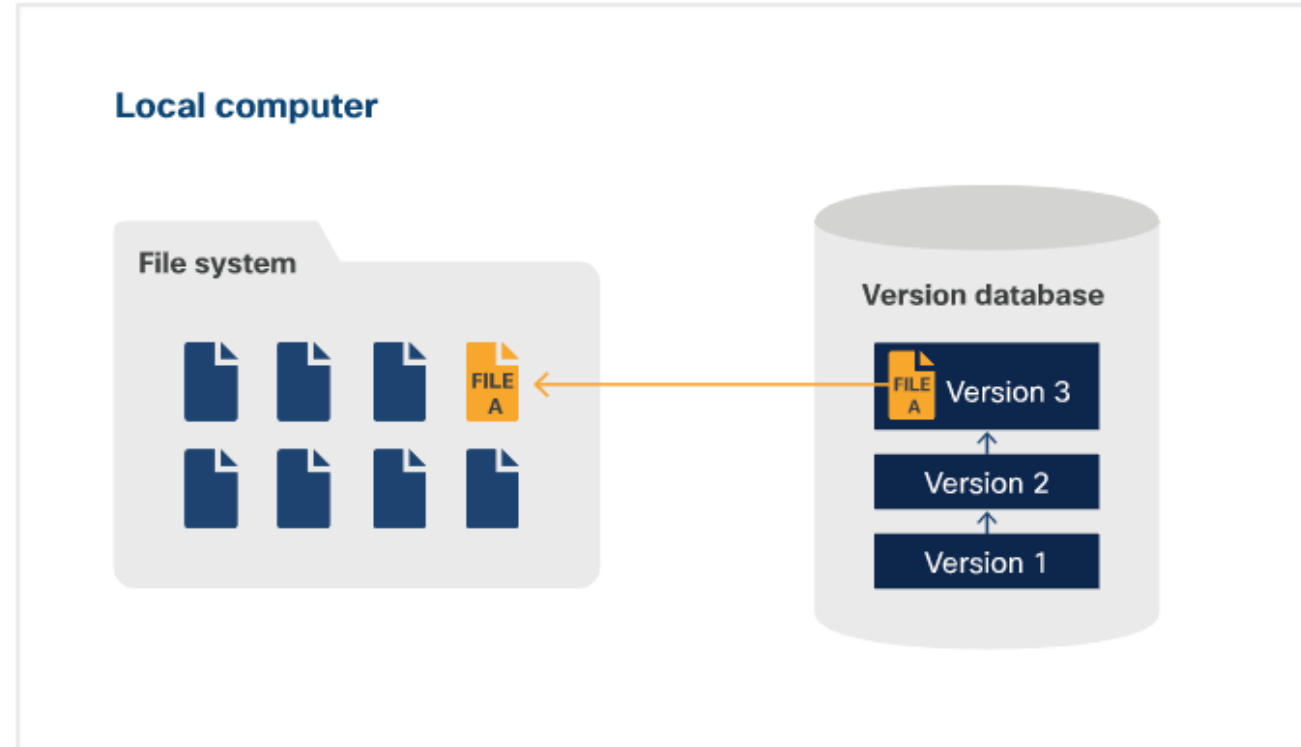
- Continue with Devnet Course in Cisco Networking Academy

Version Control System

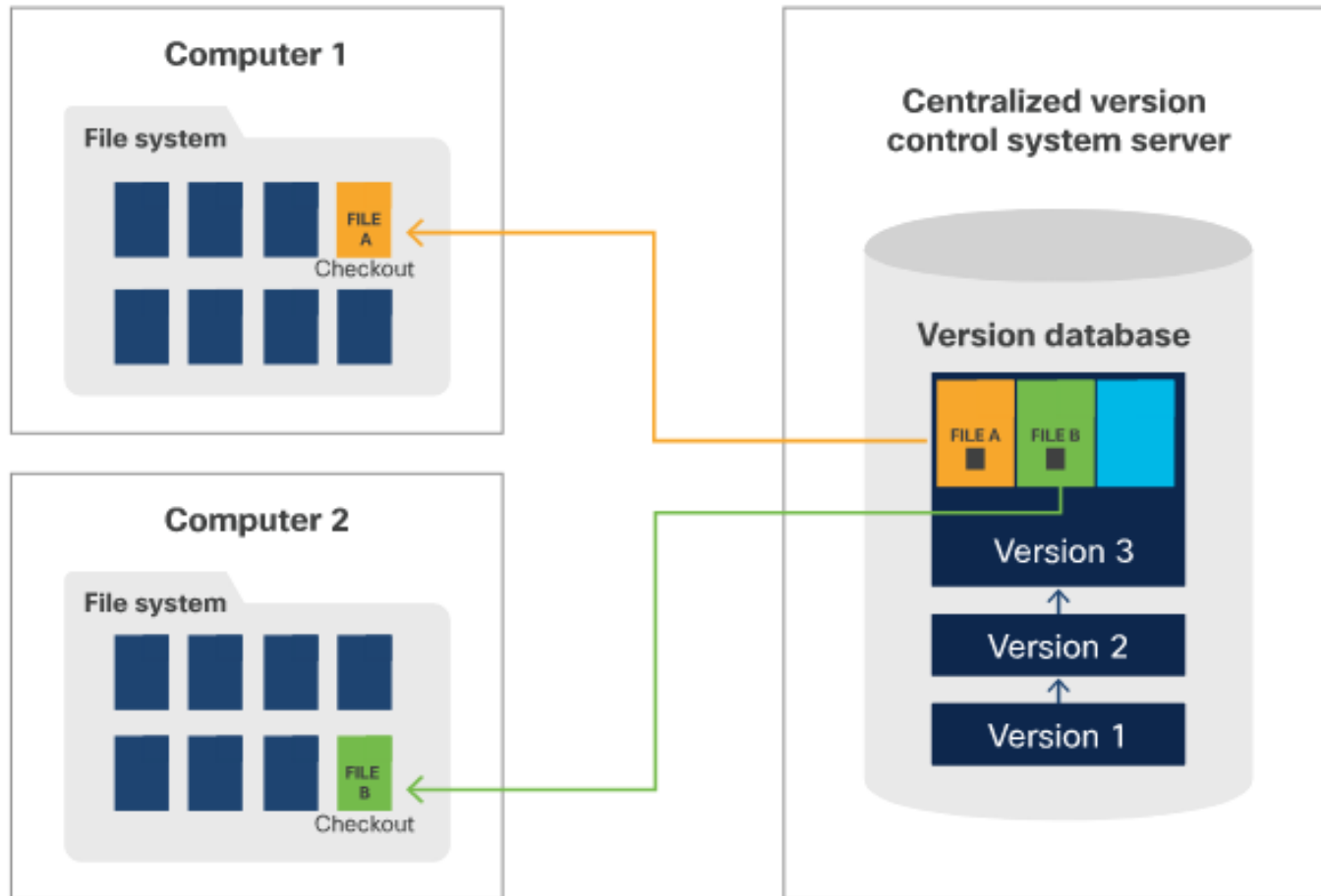
- **Version control**, also called version control systems, *revision control* or *source control*, is a way to **manage changes to a set of files in order to keep a history of those changes**.
- Benefits of version control are:
 - ✓ Enables collaboration
 - ✓ Accountability and visibility
 - ✓ Work in isolation
 - ✓ Safety
 - ✓ Work anywhere
- There are three types of version control systems:
 - ✓ Local
 - ✓ Centralized
 - ✓ Distributed

Local Version Control System

- LVCS uses a **simple database to keep track of all of the changes** to the file.
 - In most cases, the system stores the delta between the two versions of the file.
 - When the user wants to revert to the file, the delta is reversed to get to the requested version.



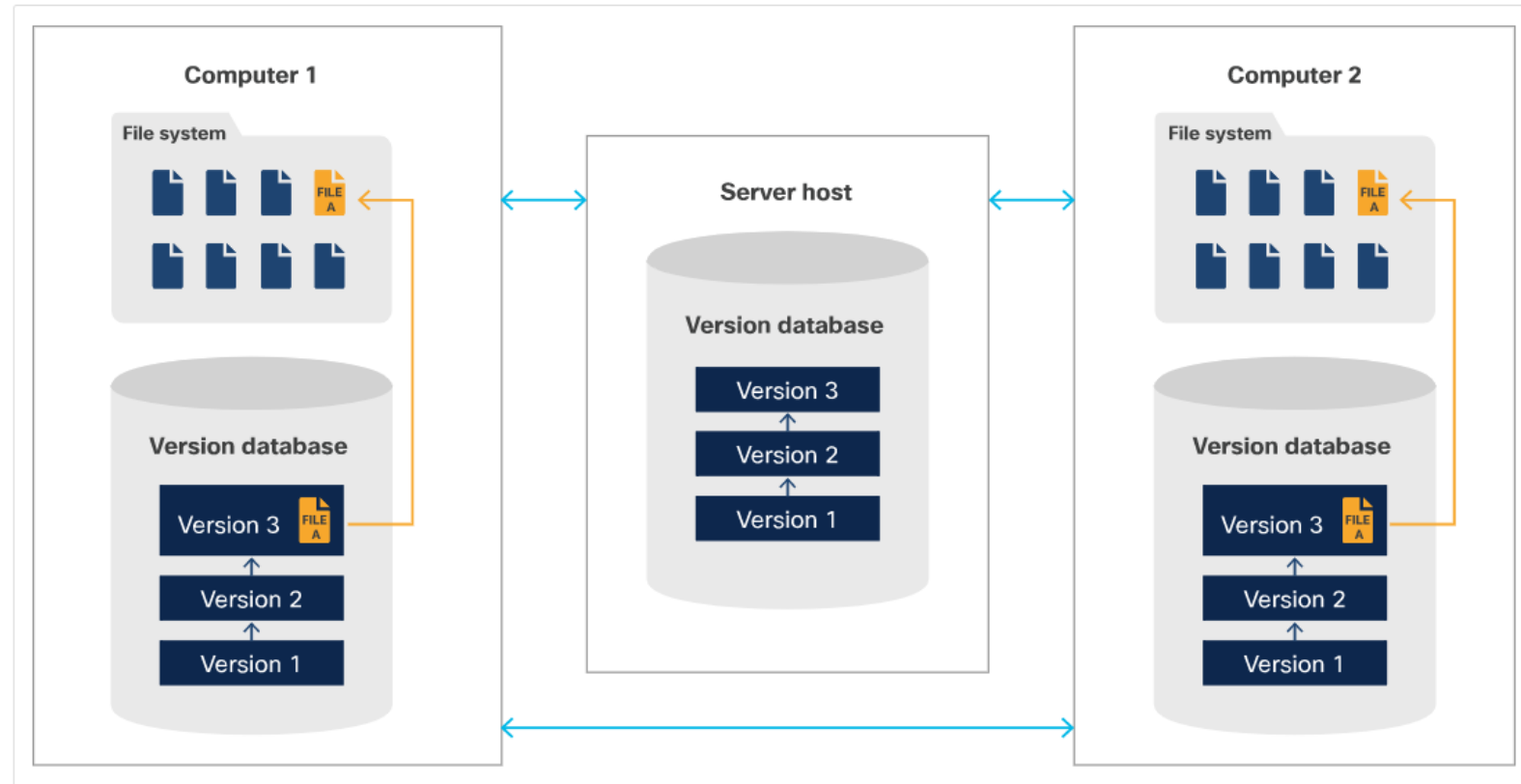
Centralized Version Control System (CVCS)



- CVCS uses a **server-client model**.
- The repository is stored in a centralized location, on a server.
- In CVCS, only one individual can work on a particular file at a time.
- An individual must check out the file to lock it and make the required changes and check in once done.

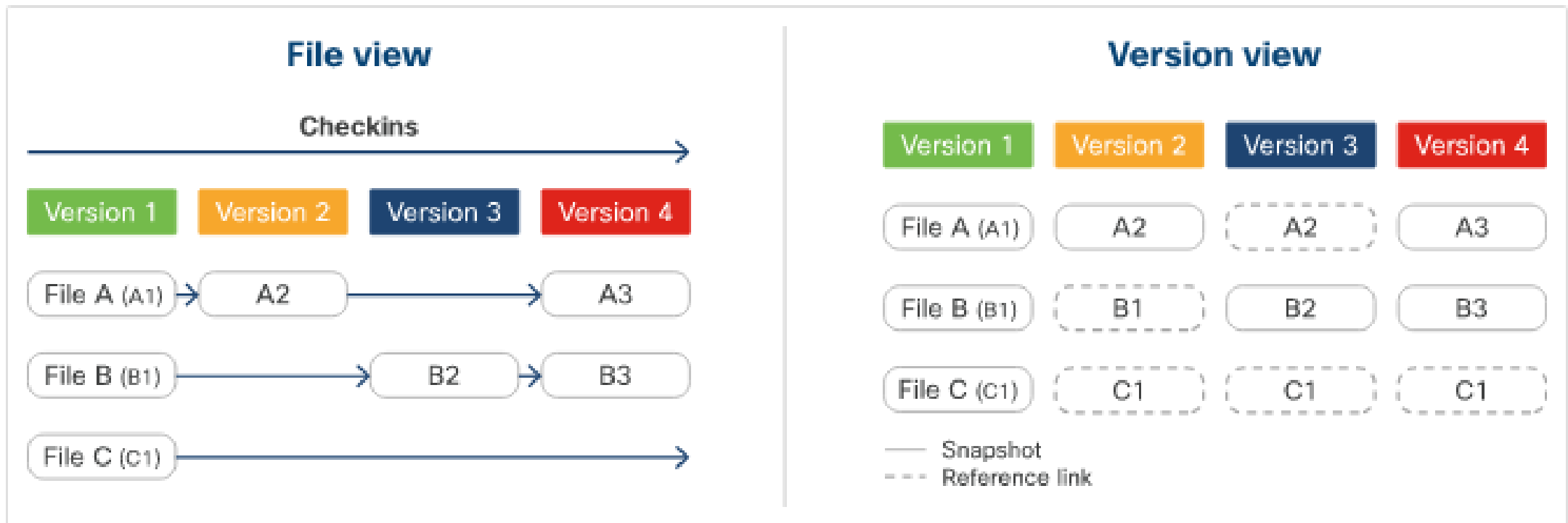
Distributed Version Control System

- DVCS is a **peer-to-peer model**.
- The repository can be stored on a client system, but it is usually stored in a repository hosting service.
- In DVCS, **every individual can work on any file, at the same time**, because the local file in the working copy is being modified. Hence, locking is not required.
- When the individual has made the changes, **they push the file to the main repository** that is in the repository hosting service, and the version control system detects any conflicts between file changes.



GIT

- **Git** is an open-source implementation of a distributed version control system.
- A **Git client must be installed on a client machine**. It is available for MacOS, Windows, and Linux/Unix.
- One key difference between Git and other version control systems is that **Git stores data as snapshots instead of differences** (the delta between the current file and the previous version).



GIT

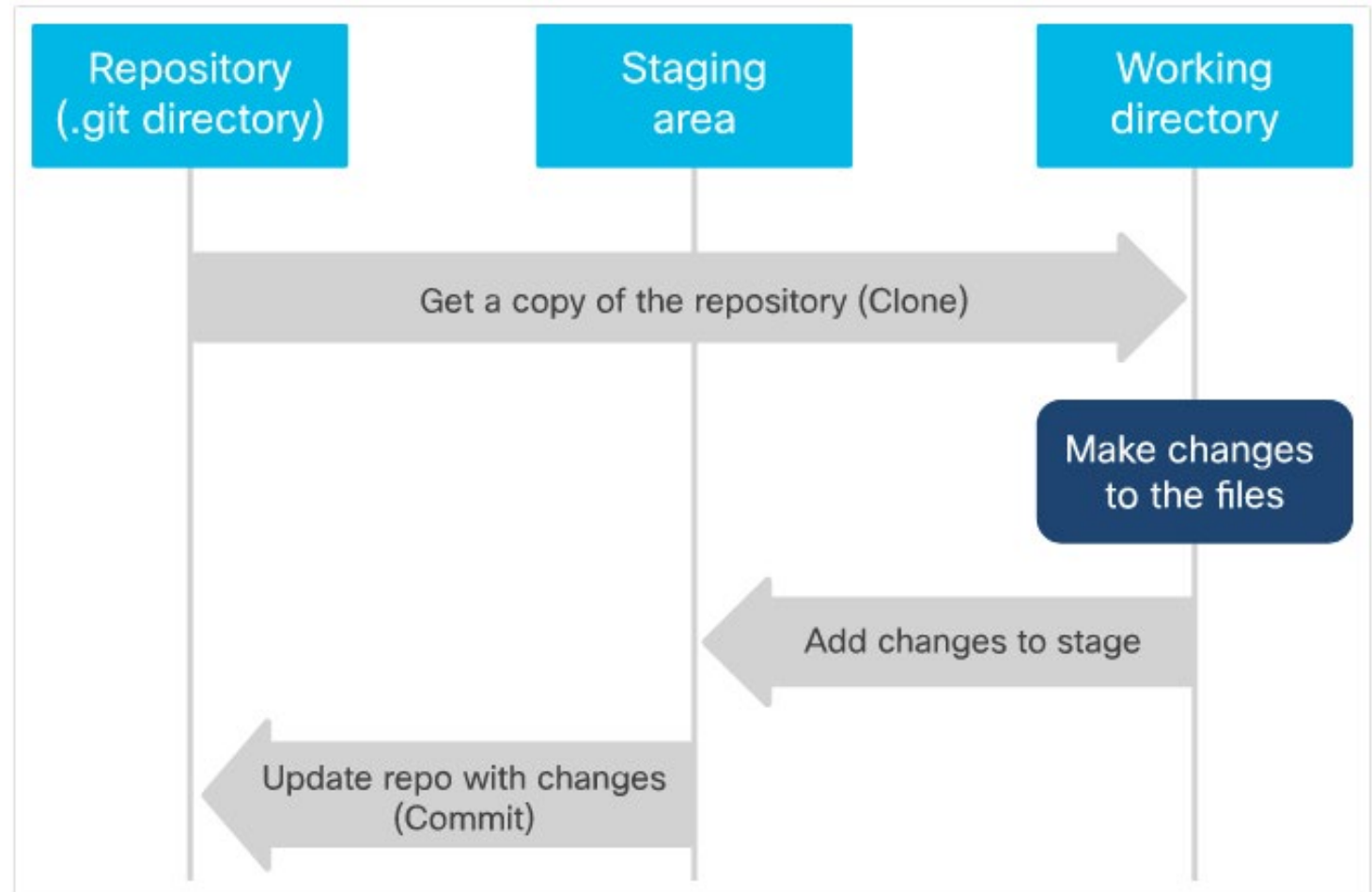
Git is organized by 3s- three stages and three states.

The three stages are:

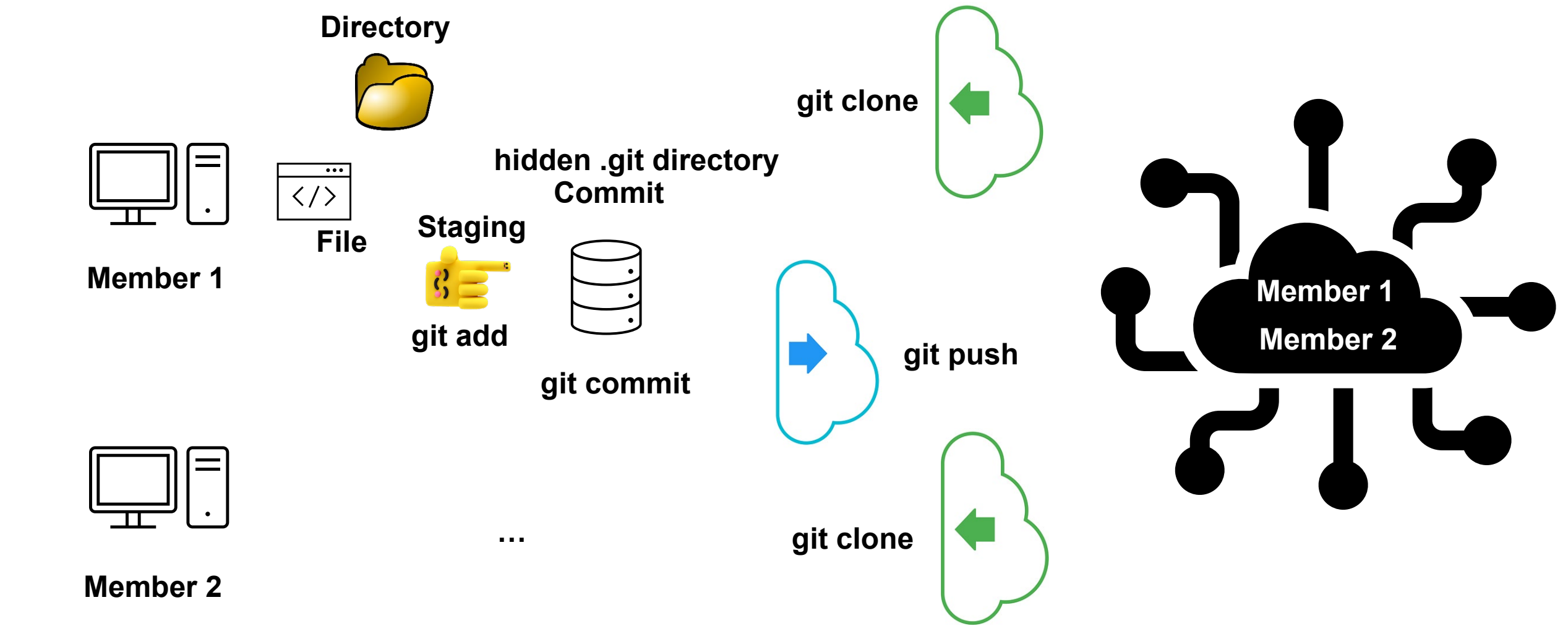
- Repository (the .git directory)
- Working directory
- Staging area

The three states are:

- ✓ Committed
- ✓ Modified
- ✓ Staged



GIT Repositories



1

```
devasc@labvm:~
File Edit View Search Terminal Help
devasc@labvm:~$ git config --global user.name "accb6019"
devasc@labvm:~$ git config --global user.email dummy.demo4class@gmail.com
devasc@labvm:~$ git config --list
user.name=accb6019
user.email=dummy.demo4class@gmail.com
devasc@labvm:~$
```

2

```
devasc@labvm:~/labs/devnet-src/git-second$ git init
Initialized empty Git repository in /home/devasc/labs/devnet-src/git-second/.git/
devasc@labvm:~/labs/devnet-src/git-second$ ls -la
.  ..  .git
```

3

```
devasc@labvm:~/labs/devnet-src/git-second$ echo "This is the first version of an example file for understanding">file1.txt
devasc@labvm:~/labs/devnet-src/git-second$ ls -la
total 16
drwxrwxr-x 3 devasc devasc 4096 Feb 16 06:21 .
drwxr-xr-x 16 devasc devasc 4096 Feb 16 06:12 ..
-rw-rw-r-- 1 devasc devasc 63 Feb 16 06:21 file1.txt
drwxrwxr-x 7 devasc devasc 4096 Feb 16 06:20 .git
devasc@labvm:~/labs/devnet-src/git-second$ cat file1.txt
This is the first version of an example file for understanding
devasc@labvm:~/labs/devnet-src/git-second$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  file1.txt

nothing added to commit but untracked files present (use "git add" to track)
```

4

```
devasc@labvm:~/labs/devnet-src/git-second$ git add file1.txt
devasc@labvm:~/labs/devnet-src/git-second$ git status
```

```
devasc@labvm:~/labs/devnet-src/git-second$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
  new file:   file1.txt
```

6

```
/git-second$ echo "Adding another line in the example file to check the version">>file1.txt
devasc@labvm:~/labs/devnet-src/git-second$ cat file1.txt
This is the first version of an example file for understanding
Adding another line in the example file to check the version
devasc@labvm:~/labs/devnet-src/git-second$ git add file1.txt
devasc@labvm:~/labs/devnet-src/git-second$ git commit -m "My second version with 2nd line"
[master d121cc0] My second version with 2nd line
1 file changed, 1 insertion(+)
```

7

```
devasc@labvm:~/labs/devnet-src/git-second$ git log
commit d121cc0ac0824070bab42b412c9b694d4980382e (HEAD -> master)
Author: accb6019 <dummy.demo4class@gmail.com>
Date: Thu Feb 16 06:28:38 2023 +0000

    My second version with 2nd line

commit 2fe27962640df7b7663e119e47f3d0faf32c2d37
Author: accb6019 <dummy.demo4class@gmail.com>
Date: Thu Feb 16 06:26:23 2023 +0000

    Committing my file to show an example
```

5

```
devasc@labvm:~/labs/devnet-src/git-second$ git commit -m "Committing my file to show an example"
[master (root-commit) 2fe2796] Committing my file to show an example
1 file changed, 1 insertion(+)
create mode 100644 file1.txt
```



GIT Repositories

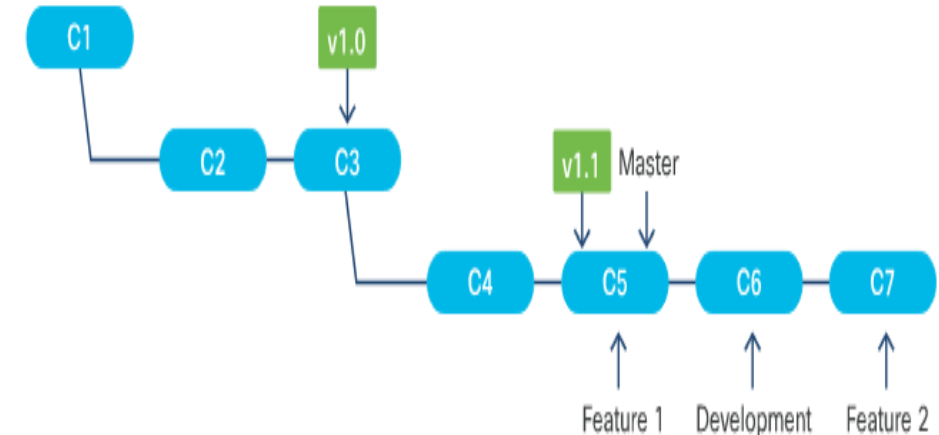
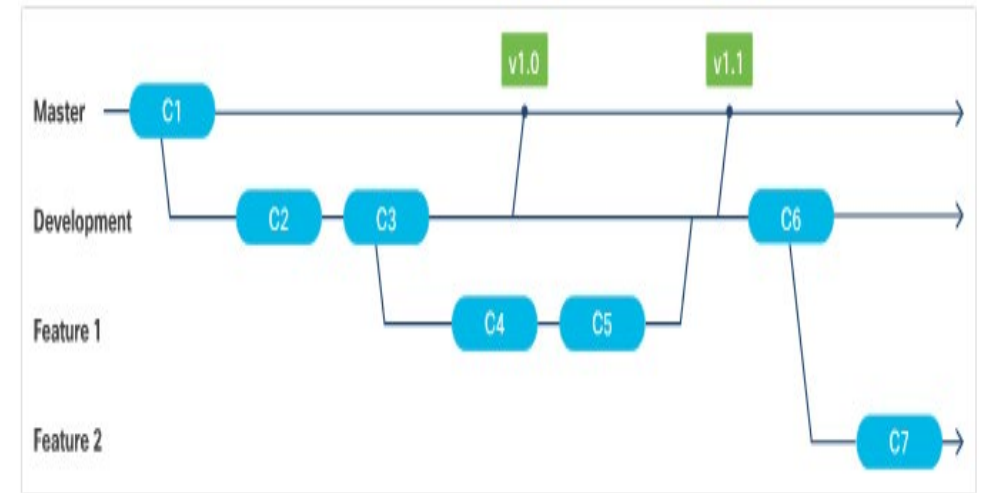
- Git has two types of repositories: **local and remote**.
- A **local repository is stored on the file system of a client machine**, which is the same one on which the git commands are being executed.
- A **remote repository is stored somewhere other than the client machine**, usually a server or repository hosting service.
- A **remote repository** with Git continues to be a **DVCS** because the remote repository **will contain the full repository**, which includes *the code and the file history*.
- When a client machine clones the repository, **it gets the full repository without requiring to lock it**, as in a CVCS.
- After the local repository is cloned from the remote repository or the remote repository is created from the local repository, **the two repositories are independent of each other until the content changes are applied to the other branch through a manual Git command execution**.

GIT Branching

Branching enables users to work on code independently without affecting the main code in the repository.

When a repository is created, the code is automatically put on a branch called Master.

- Branches can be local or remote, and they can be deleted and have their own history, staging area, and working directory.
- Git's branch creation is lightweight, and switching between branches is almost instantaneous.
- When a user goes from one branch to another, the code in their working directory and the files in the staging area change accordingly, but the repository (.git) directories remain unchanged.



GitHub

- While **Git** is an implementation of distributed version control and provides a command line interface, **GitHub** is a service provided by Microsoft that implements a repository hosting service with Git.
- In addition to providing the distributed version control and source code management functionality of Git, GitHub provides additional features such as:
 - ✓ code review
 - ✓ documentation
 - ✓ project management
 - ✓ bug tracking
 - ✓ feature requests
- GitHub introduced the concept of the 'pull request', which is a way of formalizing a request by a contributor to review changes such as new code, edits to existing code, etc., in the contributor's branch for inclusion in the project's main or other curated branches.

Feedback

Module Expectations and Daily Feedback - ACCB5029

Tue, Oct 07, 2025

<https://vevox.app/#/m/136436531>

Session ID: 136-436-531

