Git Introduction

Git Journey

Git introduction
Git workflow
Local repo operations

Branches
Merge Conflicts
GitHub

Pull request

More Practice with Git

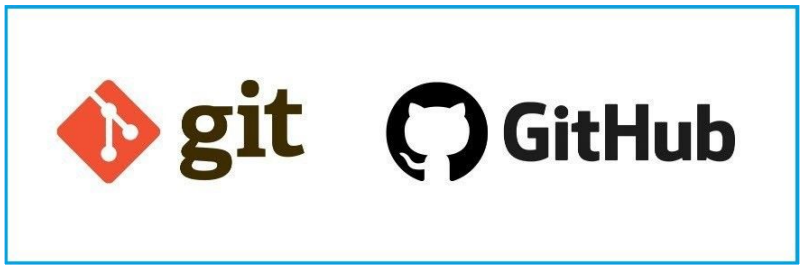


Table of Contents

What do you know about Git?

Let’s discuss about Git

What is Git?

Git is an open source distributed version control system

1 What's Version Control?

What's Version Control?

Version Control Systems

What comes to your mind when you hear this?

Students, write your response!

What's Version Control?

- Track changes on text files / source files for you
- Unlimited Undo / Redo
- Time Travel
- Collaborative development environment
- Compare and Blame
  - What changed
  - When it changed
  - Why it changed
  - Who changed it

What's Version Control?

Version Control Systems (VCS)

- **Tracks** and **records** changes to files over time
- Can track any type of file, but most commonly used for code
- Contains extra information such as date, author, and a message explaining the change

What's Version Control?

Benefits of Version Control Systems (VCS)

- Can **retrieve** previous version of files at any time
- Retrieve files that were accidentally deleted
- Can be used **locally**, or **collaboratively** with others

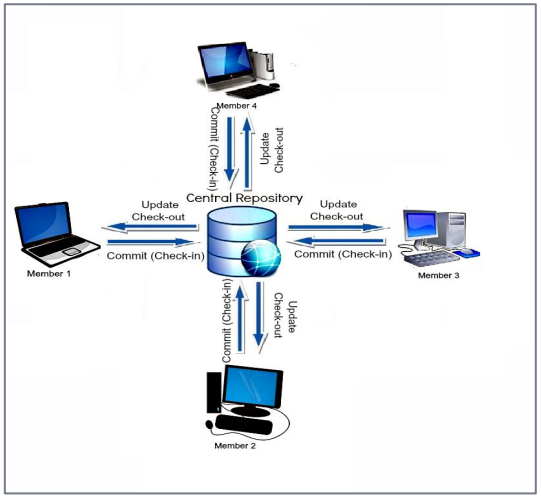
Version Control Systems

What is a "version control system"?

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.

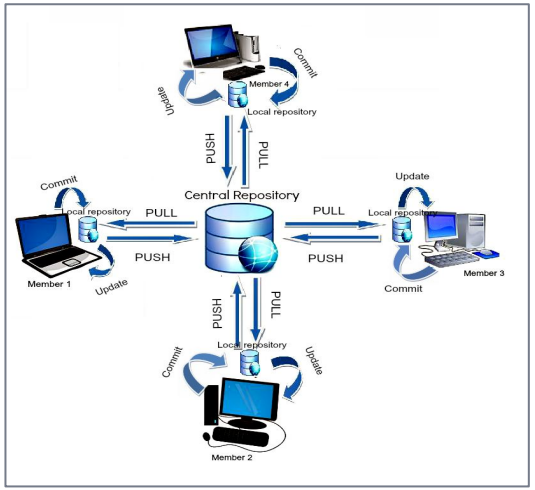
Centralized

You need to be connected to the server



Distributed

You can work while offline



Version Control Systems

- **Centralized**

In a centralized version control system (CVCS), a server acts as the main repository which stores every version of code. Using centralized source control, every user commits directly to the main branch, so this type of version control often works well for small teams, because team members have the ability to communicate quickly so that no two developers want to work on the same piece of code simultaneously. Strong communication and collaboration are important to ensure a centralized workflow is successful.

- **Distributed**

A distributed version control system (DVCS) is a type of version control where the complete codebase — including its full version history — is mirrored on every developer's computer. It's abbreviated DVCS.

2

What is Git?

What is Git?

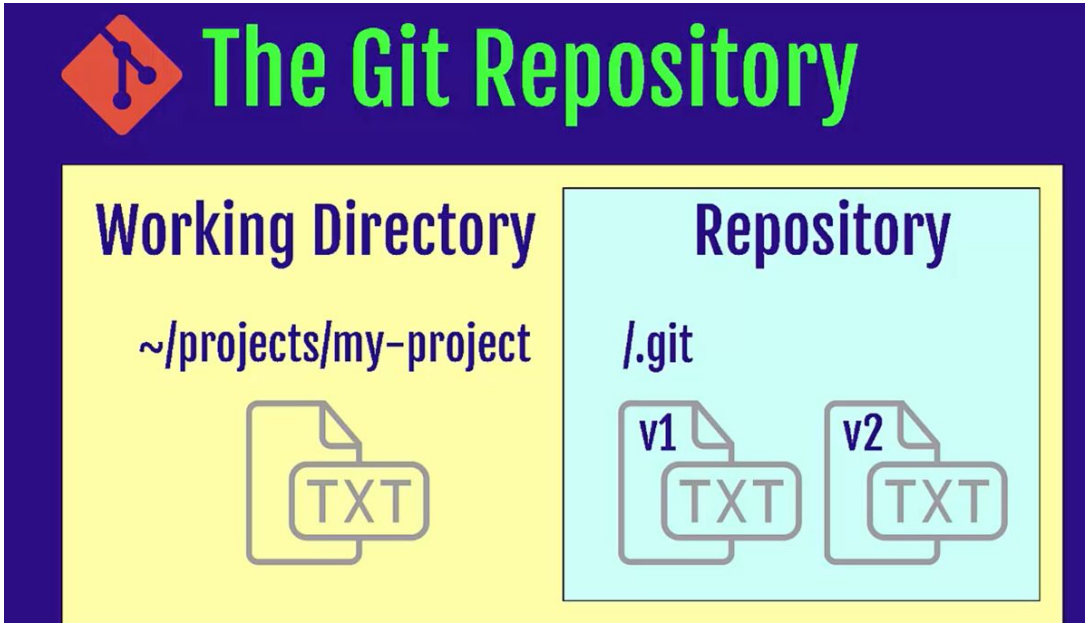
- Git is a DevOps tool/software used for source code management.
- It is a free and open-source version control system used to handle small to very large projects efficiently.
- Content Tracker and uses the concept of Distributed Version Control System (VCS)
- Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development.
- Linus Torvalds created Git in 2005 for the development of the Linux kernel.

Git Repository

What is a repository

- A directory or storage space where your projects can live.

- Local Repository
- Remote Repository



Git Repository

- Let’s check if you have git in your computer

```
git --version
```

- git needs your identity to mark/label changes / editor

```
git config --global user.name “Your Name”
```

```
git config --global user.email “Your Email”
```

```
git config --global core.editor “vim”
```

```
git config --list
```

Git Repository

- to create a new local repo

```
git init
```

- to see the commands

```
git help
```

- to see the status of your repo

```
git status
```

3 Workflow

Workflow

Working Directory

Where you work. Create new files, edit files delete files etc.

Staging Area (Index)

Before taking a snapshot, you're taking the files to a stage. Ready files to be committed.

Repository

Committed snapshots of your project will be stored here with a full version history.

File Stages

Committed | Unmodified changes from the last commit snapshot

Modified | Changes made to files since last commit snapshot

Staged | Changes marked to be added into the next commit snapshot

Stage modified files & commit changes

Create a new file

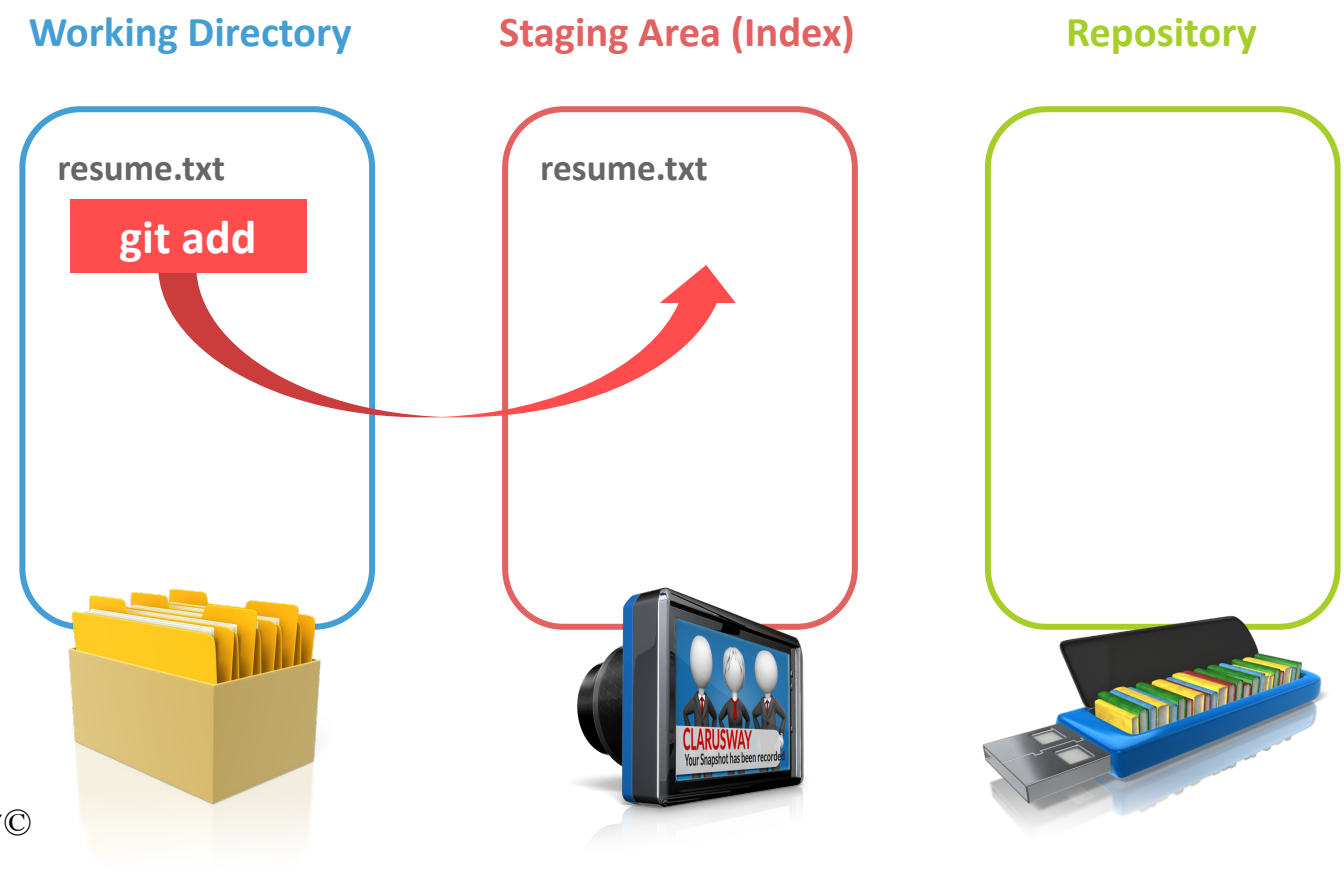
Working Directory

resume.txt
untracked file

Staging Area (Index)

Repository

Track/stage a file



Stage files options

- stage one file

```
git add filename
```

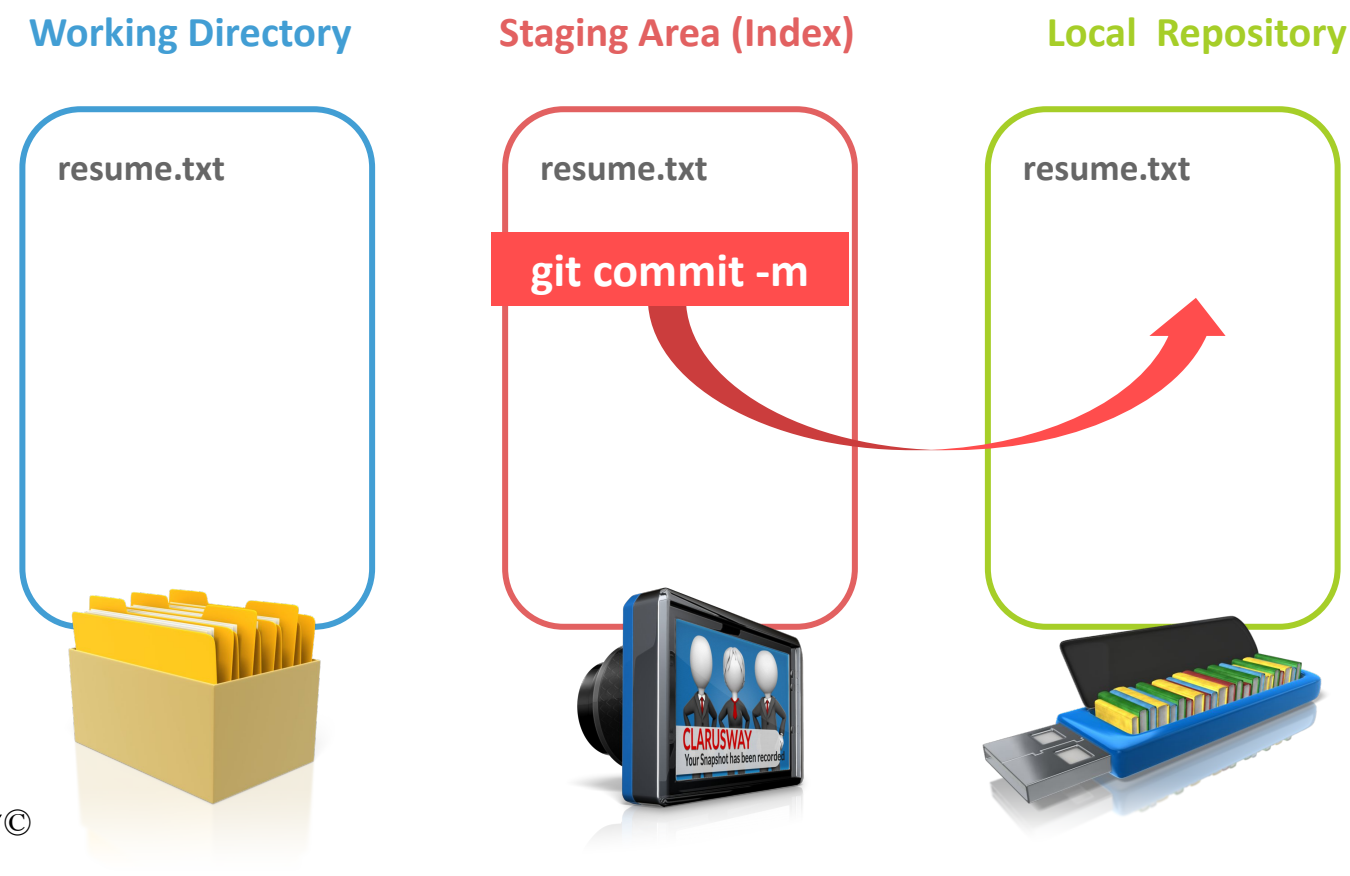
- stage all files (new, modified)

```
git add .
```

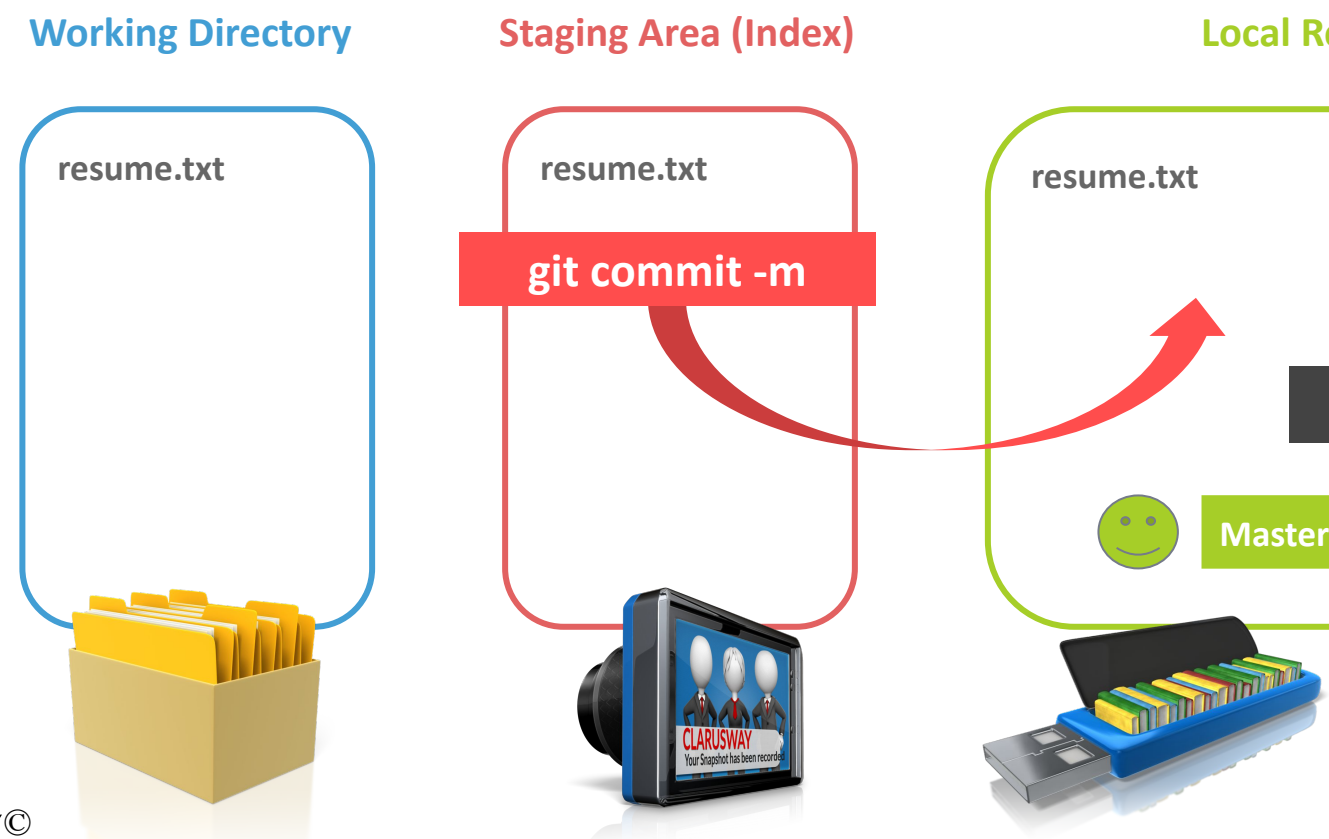
- stage modified and deleted files only

```
git add -u
```

Commit


Working Directory
Staging Area (Index)
Local Repository
resume.txt
resume.txt
git commit -m
resume.txt
CLARUSWAY
Your Snapshot has been recorded

Commit


Working Directory
Staging Area (Index)
Local Repository
resume.txt
resume.txt
resume.txt
git commit -m
git log
Master
CLARUSWAY
Your Snapshot has been recorded

Commit

- Commit the files on the stage

```
git commit -m ”message”
```

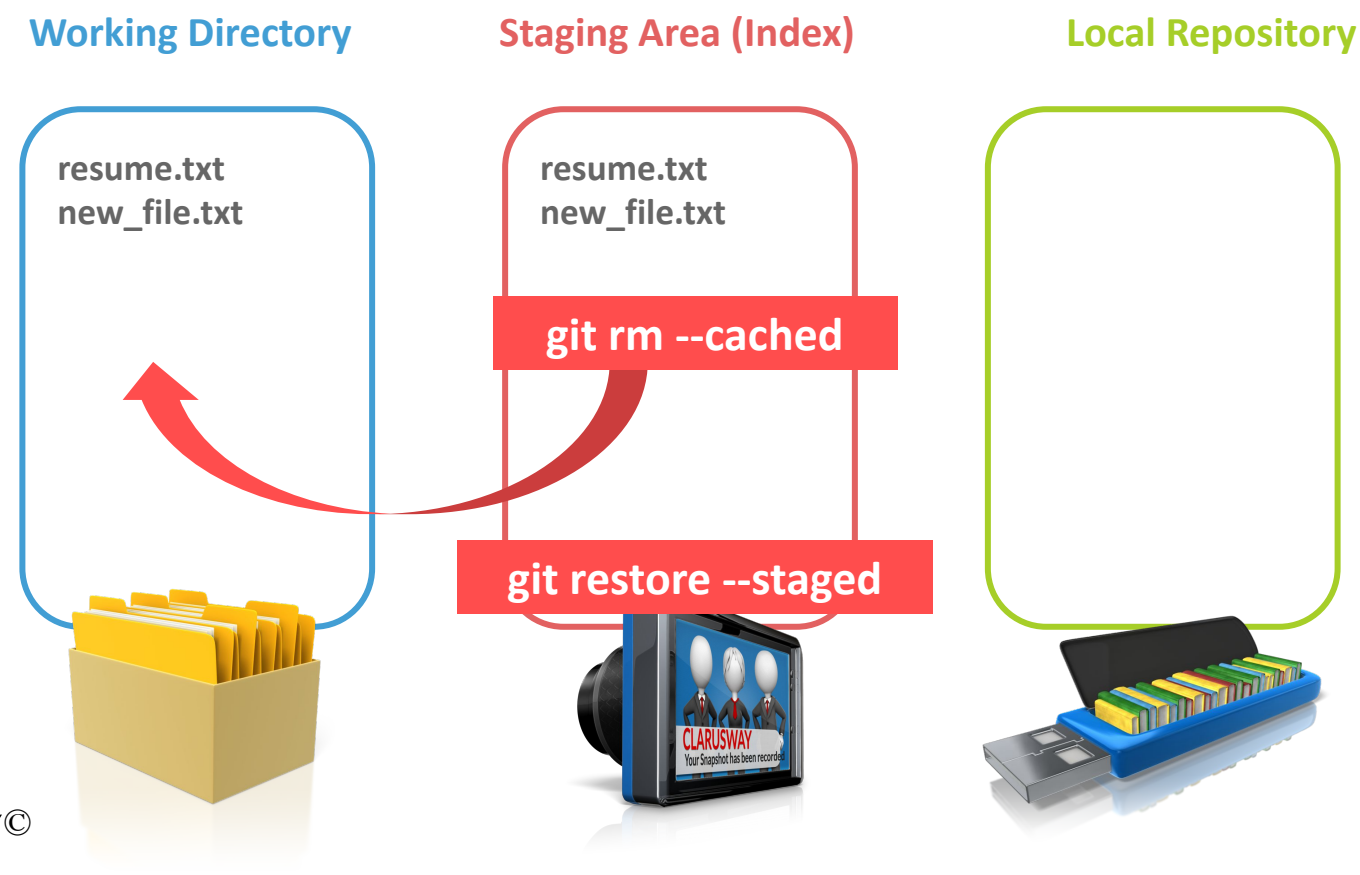
- Add and commit all tracked files

```
git commit -am ”message”
```

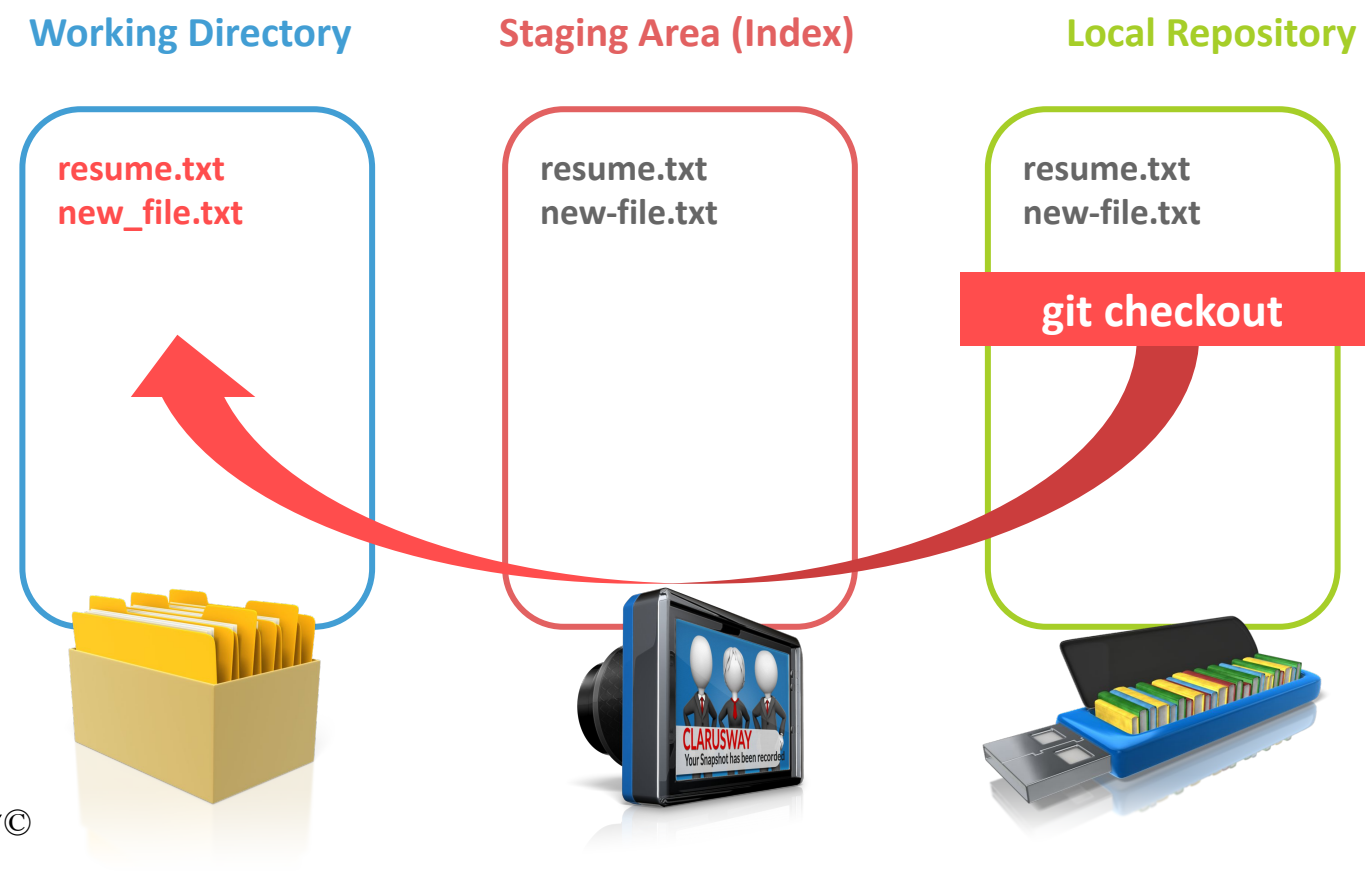
- amend commit message

```
git commit --amend
```

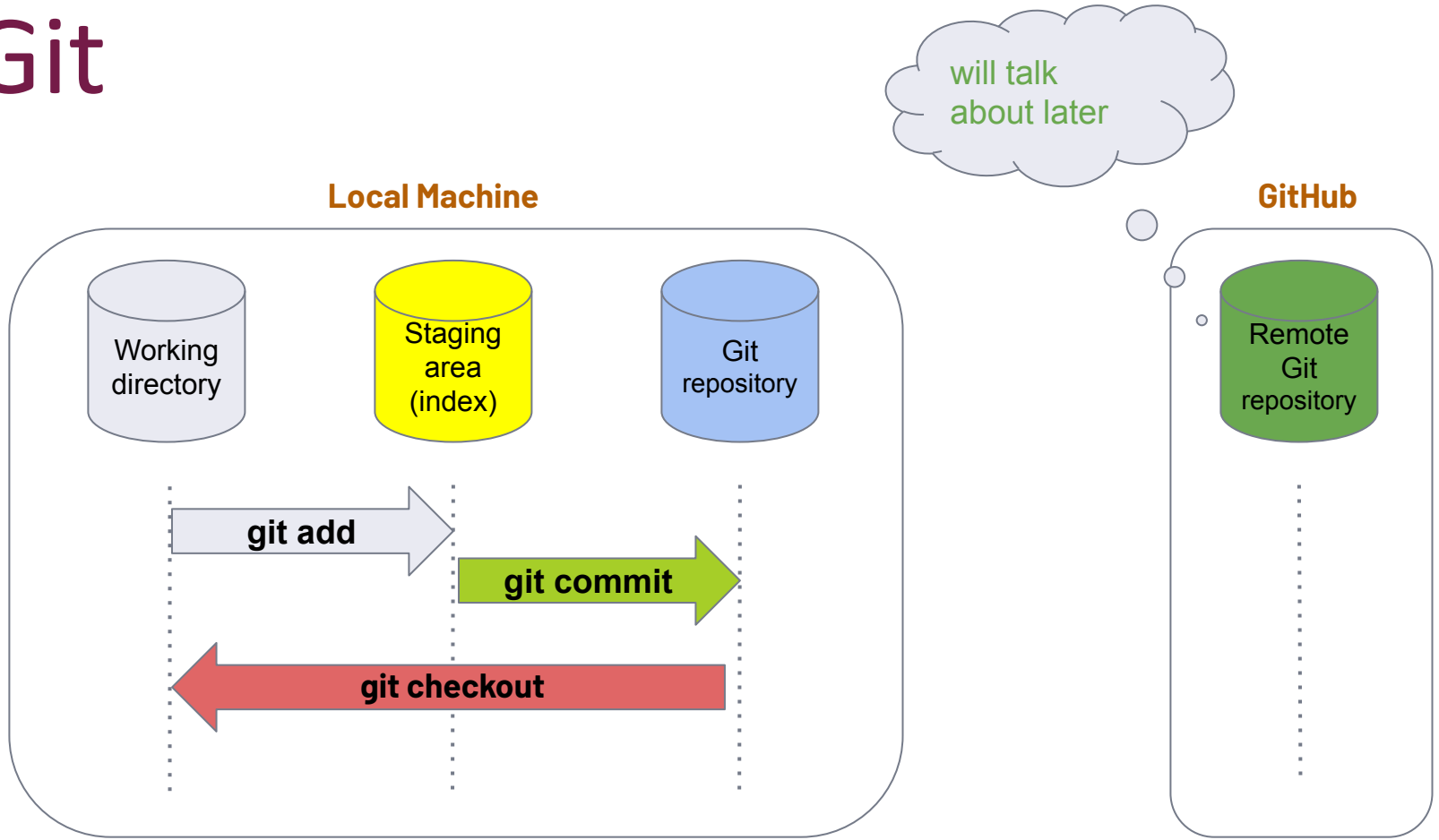
Remove from stage



Checkout from Repo



Git


will talk
about later
Local Machine
GitHub
Working
directory
Staging
area
(index)
Git
repository
Remote
Git
repository
git add
git commit
git checkout

New Project

- Create a repo **git init**
- Create a new file/edit file etc.
- Stage/Track your changes **git add .**
- Commit changes **git commit -m “message”**

Task-1

- Create a new repo under **project-3** folder
- Create a file named **file1.txt**
- Change the file
- Stage the file
- Commit the file to your repo

Students, write your response!

Task-1 Solution

- Create a new repo under **project-3** folder → `git init`
- Create a file named **file1.txt** → `touch file1.txt`
- Change the file → `vim file1.txt`
- Stage the file → `git add .`
- Commit the file to your repo → `git commit -m "message"`

Task-2

- Create a file named **file2.txt**
- Edit **file2.txt**
- Stage
- Delete the file **file1.txt**
- Rename **file2.txt** >> **file3.txt**
- Stage **file3.txt**
- Unstage **file3.txt**
- Stage **file3.txt** again
- Commit the file to your repo
- Change the message of the commit
- Switch back to your first commit in Task-1

Students, write your response!

Task-2 Solution

→ Create a file named **file2.txt**	`touch file2.txt`
→ Edit **file2.txt**	`vim file2.txt`
→ Stage	`git add .`
→ Delete the file **file1.txt**	`rm file1.txt`
→ Rename **file2.txt** >> **file3.txt**	`mv file2.txt file3.txt`
→ Stage **file3.txt**	`git add .`

Task-2 Solution Cntd.

- Unstage **file3.txt**

```
git rm --cached file3.txt
```

- Stage **file3.txt** again

```
git add .
```

- Commit the file to your repo

```
git commit -m "message"
```

- Change the message of the commit

```
git commit --amend -m "message"
```

- Switch back to your first commit in Task-1

```
git log
```

```
git checkout "first commit ID"
```

Summary

Summary

git init

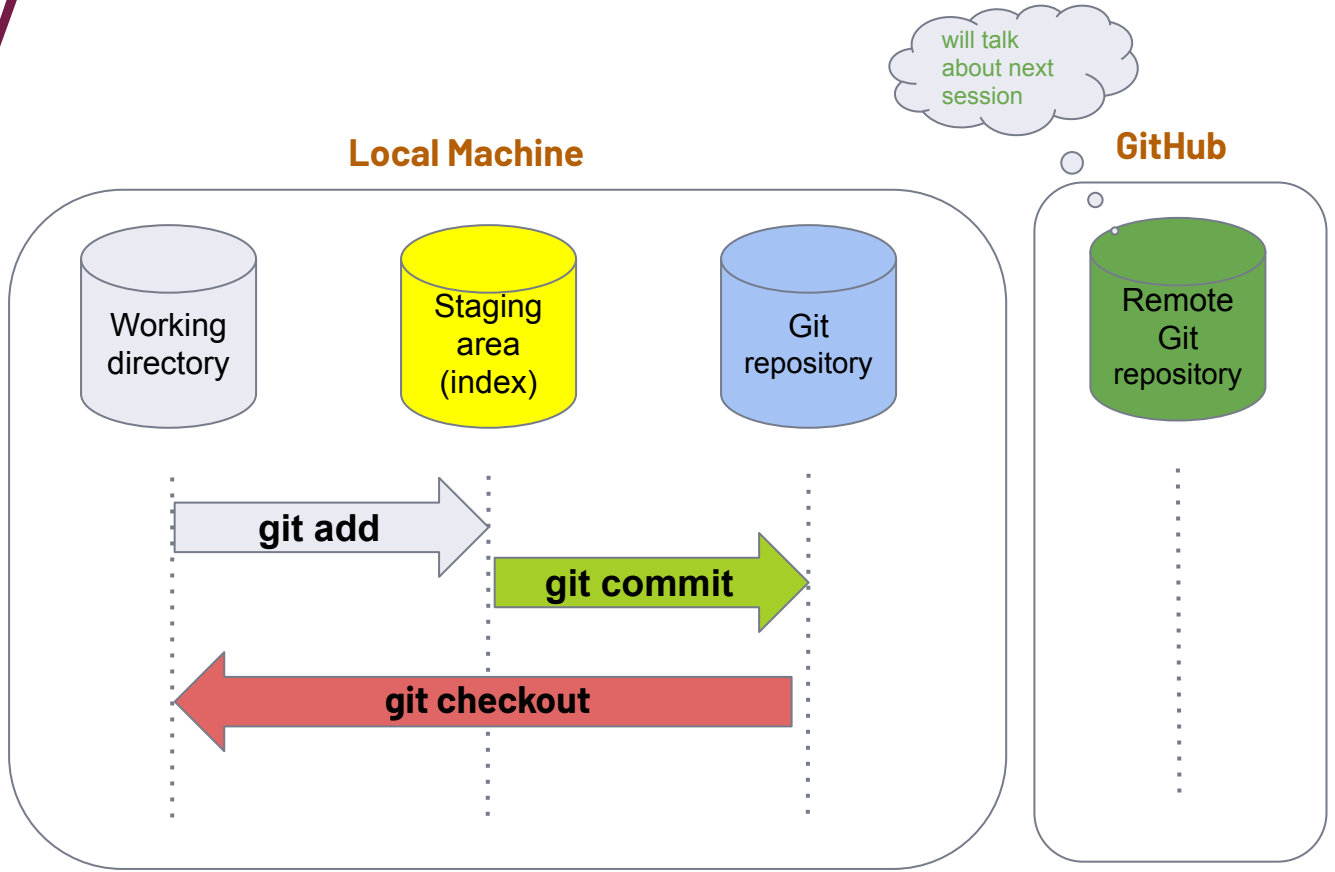
git status

git add .

git commit -m “abc”

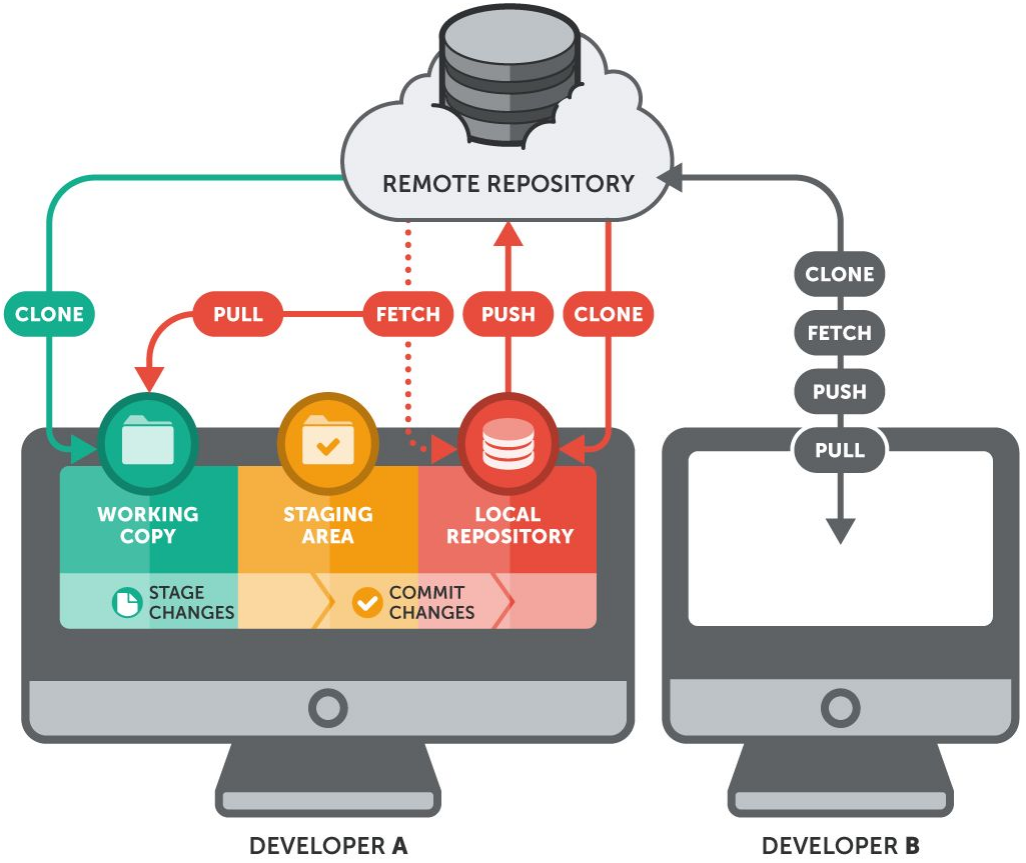
git log

git checkout



Summary

- The overwhelming majority of work happens in the local repository.
- Until this point (except when we called "git clone"), we've worked exclusively with our local Git repository and never left our local computer.
- We were not dependent on any internet or network connection but instead worked completely offline.



THANKS!

Any questions?

You can find me at:

- sumod
- sumod@clarusway.com