

Document



Group line

Week1 : SOFTWARE DEVELOPMENT TOOLS AND ENVIRONMENTS

แผนกราสสอน	
หัวข้อ	รายละเอียด
1	<p>Git and GitHub Configure Git Creating and Cloning Repositories Private Repositories and Token</p> <p>Lab week 1</p>
2	<p>Understanding Git Usage and Workflow Add and Commit Git Log Git Remote and Git Push Fetch and Pull</p> <p>Lab week 2</p>
3	<p>Understanding Branches Understanding HEAD Git Branch Commands Delete and Rename Branches Merging Branches - Theory and Concepts Merging Branches in Practice Git Diff</p> <p>Lab week 3</p>
4	<p>Git with Going back and Undoing Changes -Git Checkout and Detached HEAD -Git Restore, Git Reset, Git Revert Undoing Changes - Exercise and Solution</p>

5	<p>Docker</p> <p>Docker Overview Basic Docker Commands Docker Run</p> <p>Docker Images Environment Variables Command vs Entrypoint</p>
6	<p>Labs - Docker Images Labs - Environment Variables Labs - Command vs Entrypoint</p> <p>Docker Compose Docker Registry</p> <p>Lab: Docker Registry Labs: Docker Compose</p>
7	<p>Docker Engine Docker Storage Docker Networking</p> <p>Labs - Docker Storage Labs - Docker Networking</p>
8	<p>Docker Swarm Docker Service Docker Stacks CI/CD - Docker Integration</p> <p>Lab</p>

<b>5</b>	<p>Docker</p> <p>Docker Overview Basic Docker Commands Docker Run</p> <p>Docker Images Environment Variables Command vs Entrypoint</p> <p>Labs - Docker Images Labs - Environment Variables Labs - Command vs Entrypoint</p>	<b>6</b>	
<b>7</b>	<p>Docker Engine Docker Storage Docker Networking</p> <p>Labs - Docker Storage Labs - Docker Networking</p>	<b>8</b>	<p>Docker Swarm Docker Service Docker Stacks CI/CD - Docker Integration</p> <p>Lab</p>
		<b>9</b>	<p>Kubernetes 1</p> <p>Container Orchestration Kubernetes Architecture PODs</p>
		<b>10</b>	<p>Kubernetes 2</p> <p>Basics of Networking in Kubernetes ReplicaSets and Deployments</p>
		<b>11</b>	<p>Micro service</p>
		<b>12</b>	<p>Jenkins</p>
		<b>13</b>	<p>Mini Project Pitching</p>
		<b>14</b>	<p>Mini Project Pitching</p>
		<b>15</b>	<p>Mini Project Pitching</p>
		<b>16</b>	<p>Mini Project Pitching</p>

การให้คะแนน	
กล่างภาค	30
ปลายภาค	35
LAB	15
Mini project	20

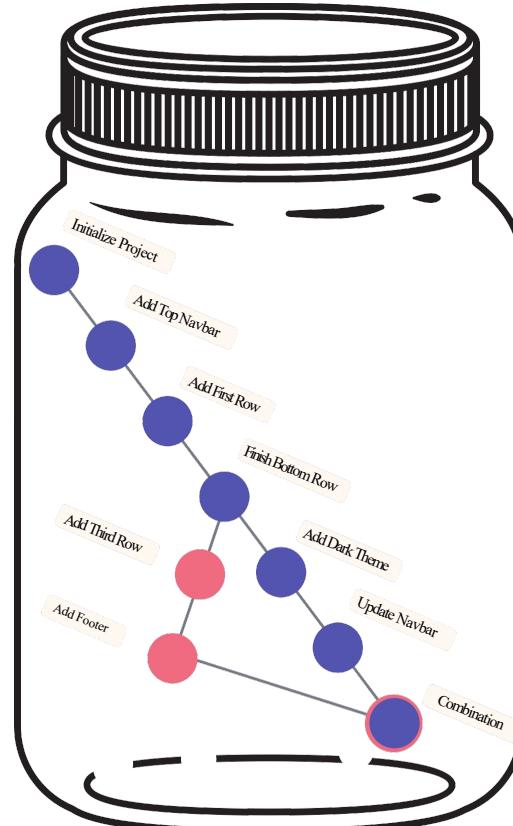
- Git was developed in 2005 by **Linus Torvalds**
- Git is **Version control system** is a system that records changes to a file or set file over time so that you. can restore specific version later
- Git is a **Distributed Version Control System**

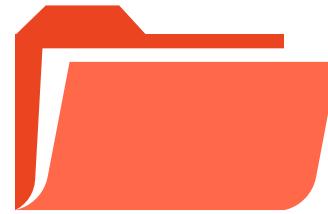


# Repository

A Git "Repo" is a workspace which tracks and manages files within a folder.

Anytime we want to use Git with a project, app, etc we need to create a new git repository. We can have as many repos on our machine as needed, all with separate histories and contents















Portfolio Website

Startup Idea

My First Movie Script

Symphony #17

Reddit Clone App

# Git helps us...

Track changes across multiple files

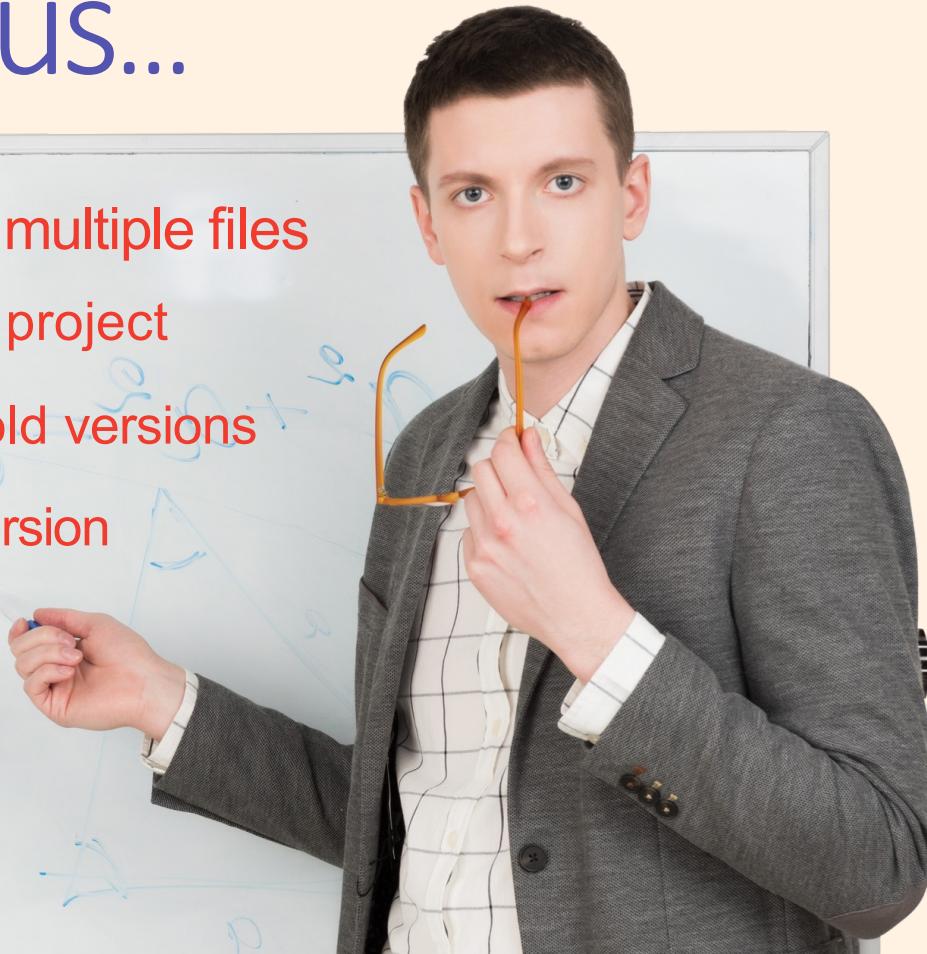
Compare versions of a project

"Time travel" back to old versions

Revert to a previous version

Collaborate and  
share changes

Combine changes



# Git – What and Why

---

Oh boy, I sure do  
love playing my  
video games!



I'm going to save  
my game now in  
case I die soon!



Oh jeez, this is  
going to be a  
difficult fight!





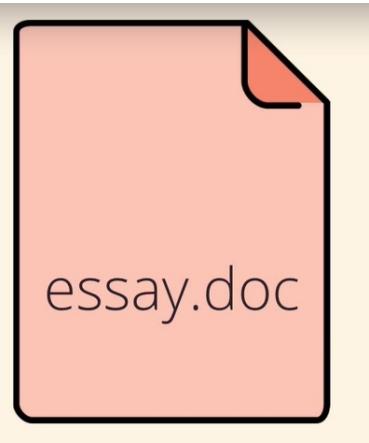
ughhhh I died!





Thank heavens I  
saved my game! I  
can just revert!





essay.doc

essay\_v2.doc

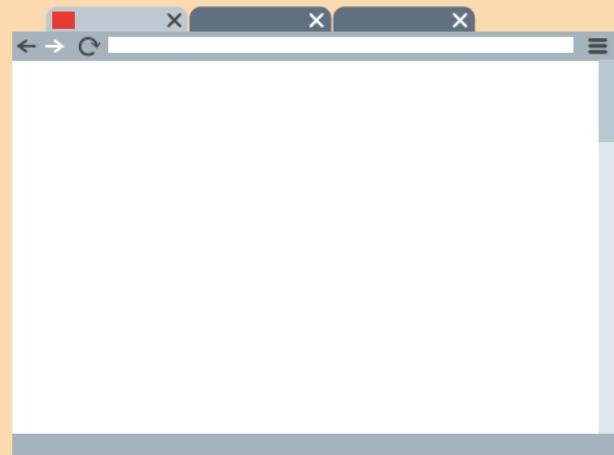
essay\_final.doc

essay\_v1.doc

essay\_v2  
new\_intro.doc

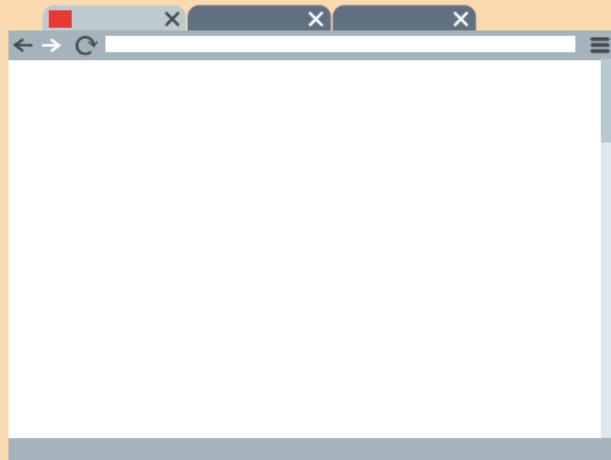
essay\_FINAL\_  
FINAL\_FOR\_  
REAL.doc

# I Start A New Project!



# Add A Checkpoint

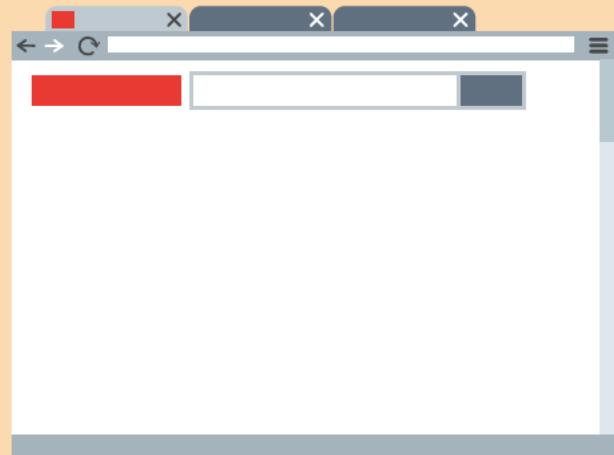
Initialize Project



# Add A Checkpoint

Initialize Project

Add Top Navbar

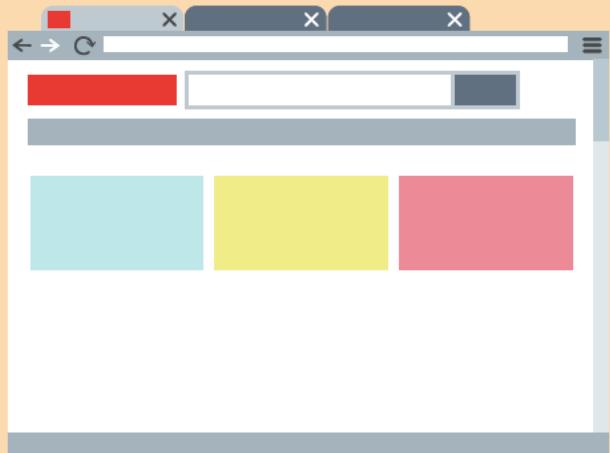


# Add A Checkpoint

Initialize Project

Add Top Navbar

Add First Row



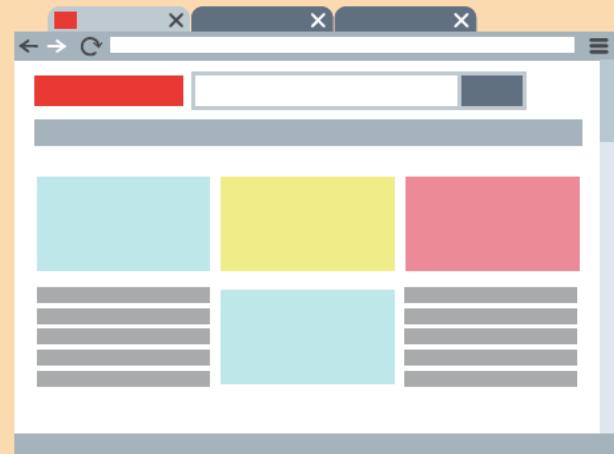
# Add A Checkpoint

Initialize Project

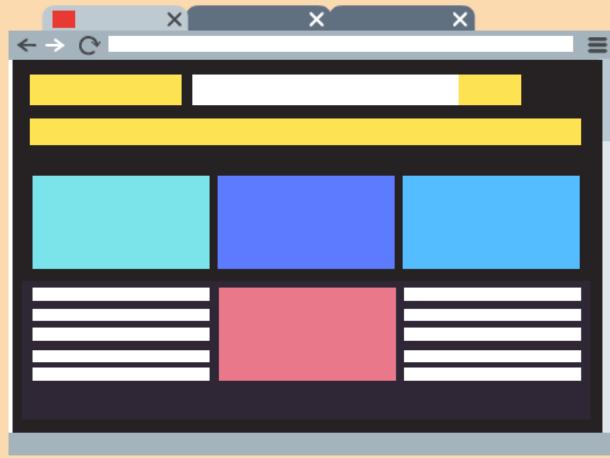
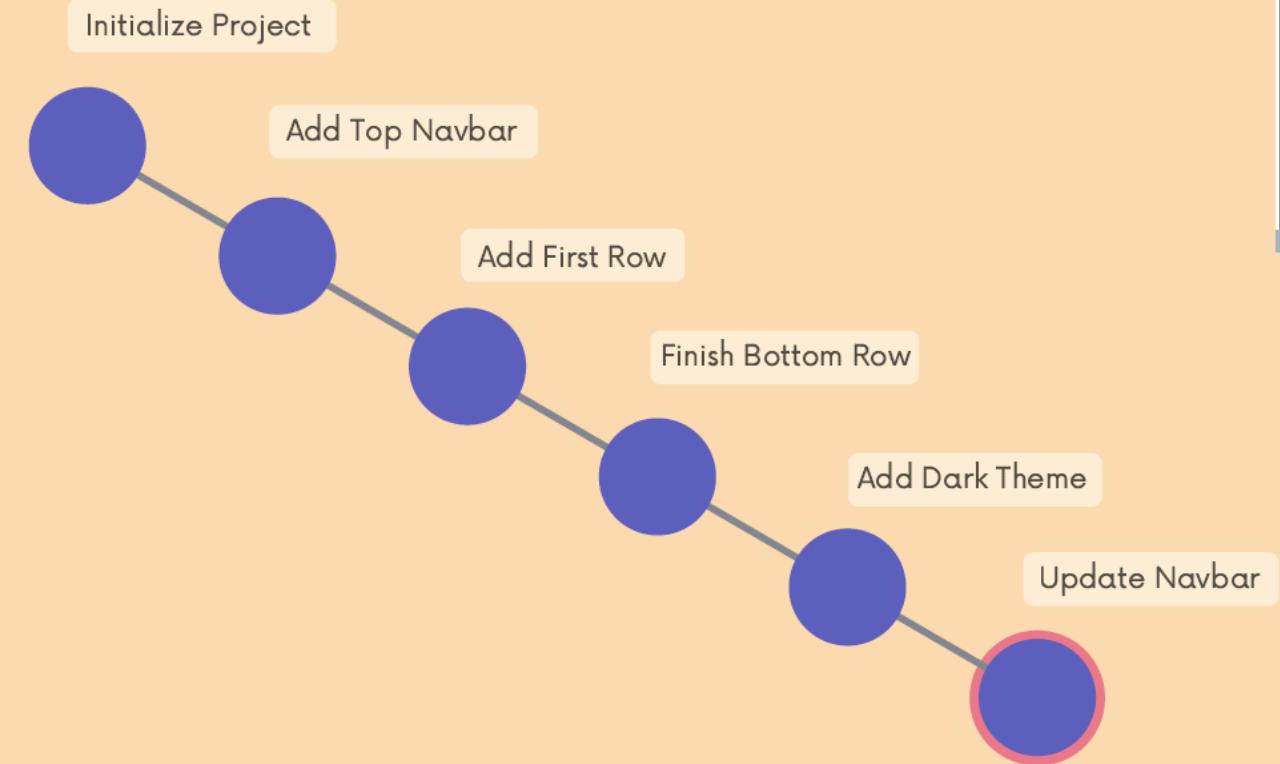
Add Top Navbar

Add First Row

Finish Bottom Row



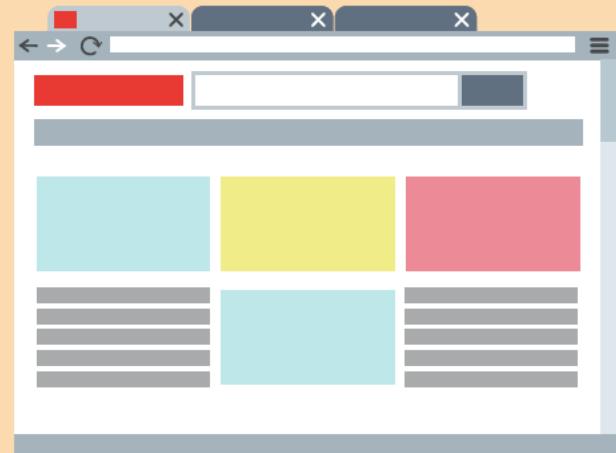
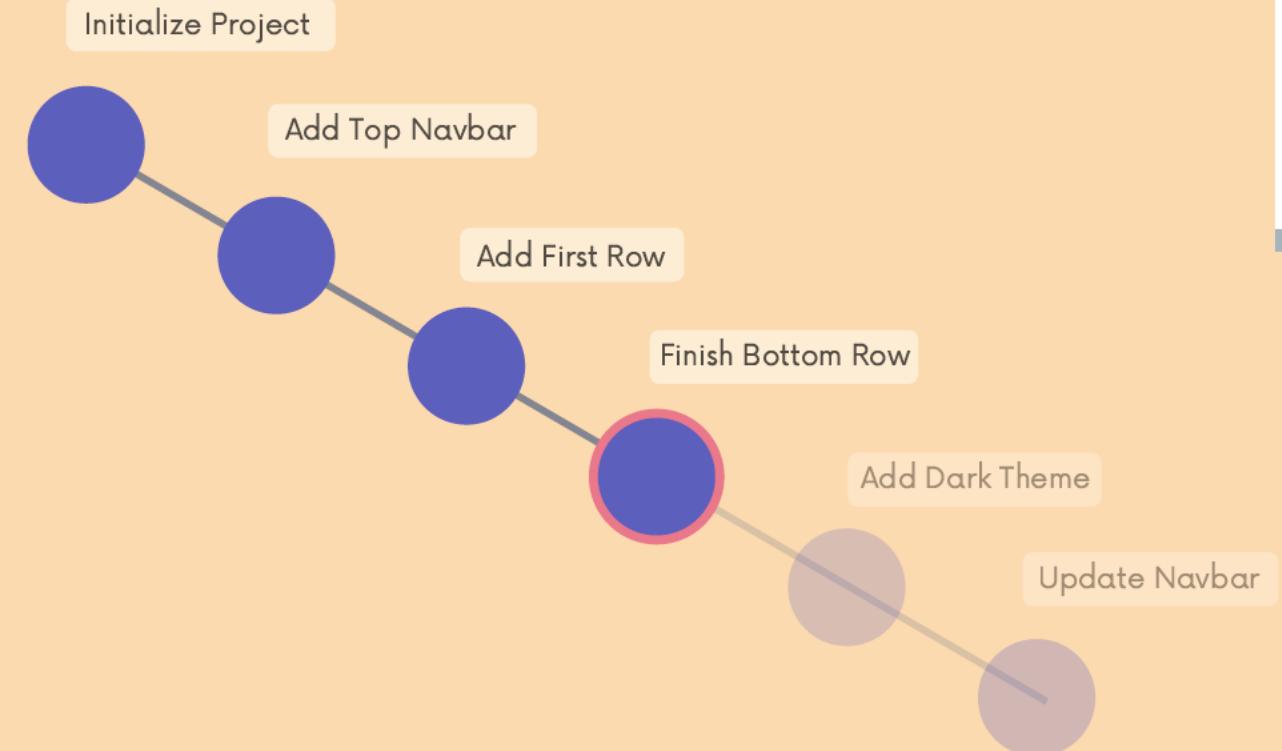
# Add A Checkpoint



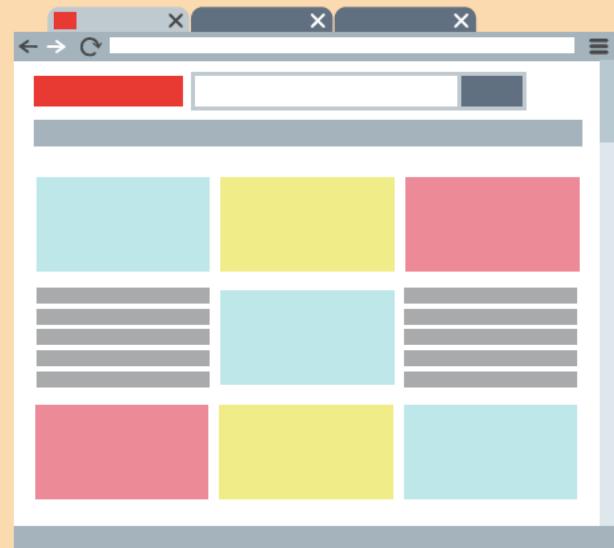
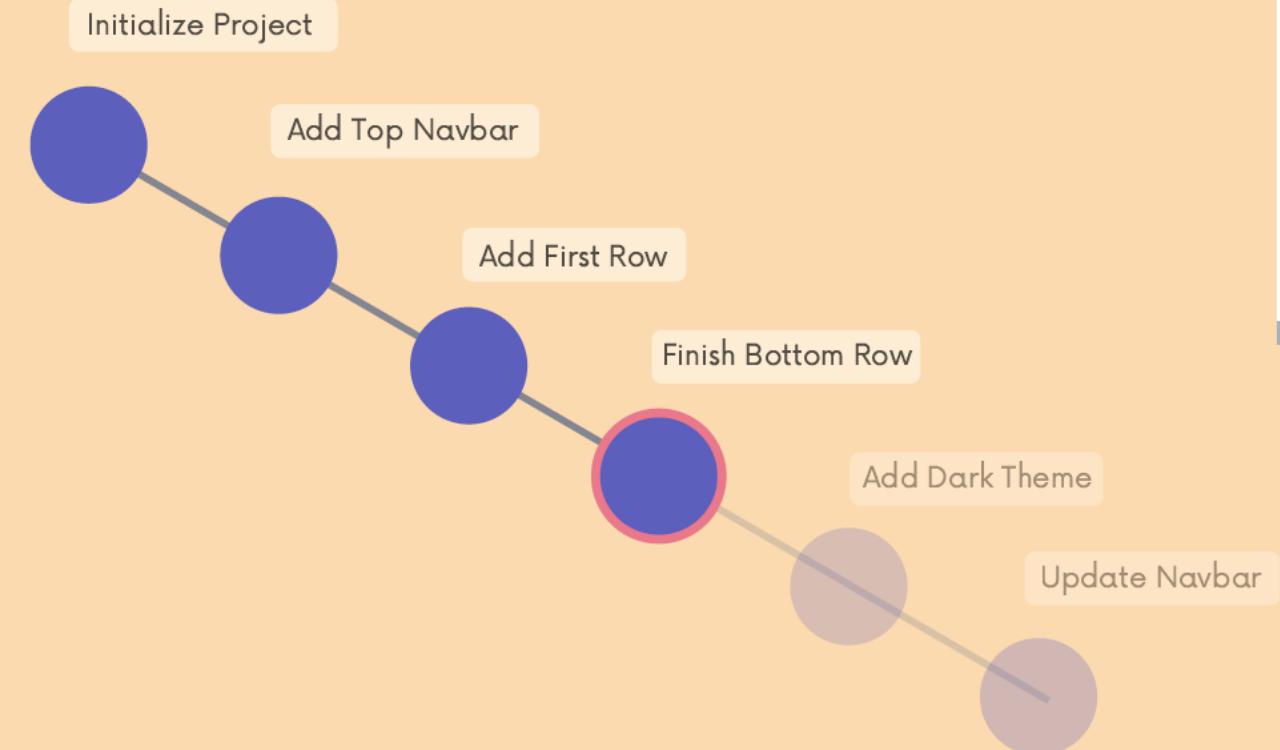
A woman with long brown hair, wearing a brown blazer over a light-colored shirt and shorts, is shouting and pointing her right index finger towards the right side of the frame. She is holding a small blue object in her left hand.

**ANGRY BOSS SAYS...  
THE COLORS ARE BAD!**

# I can go back to prior checkpoints I made!

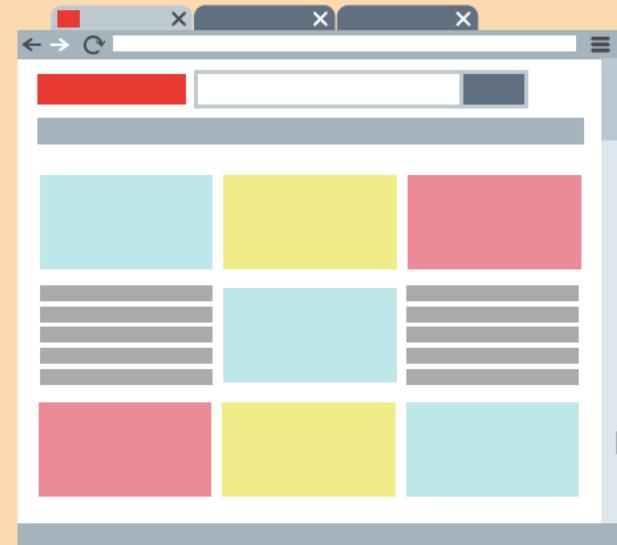
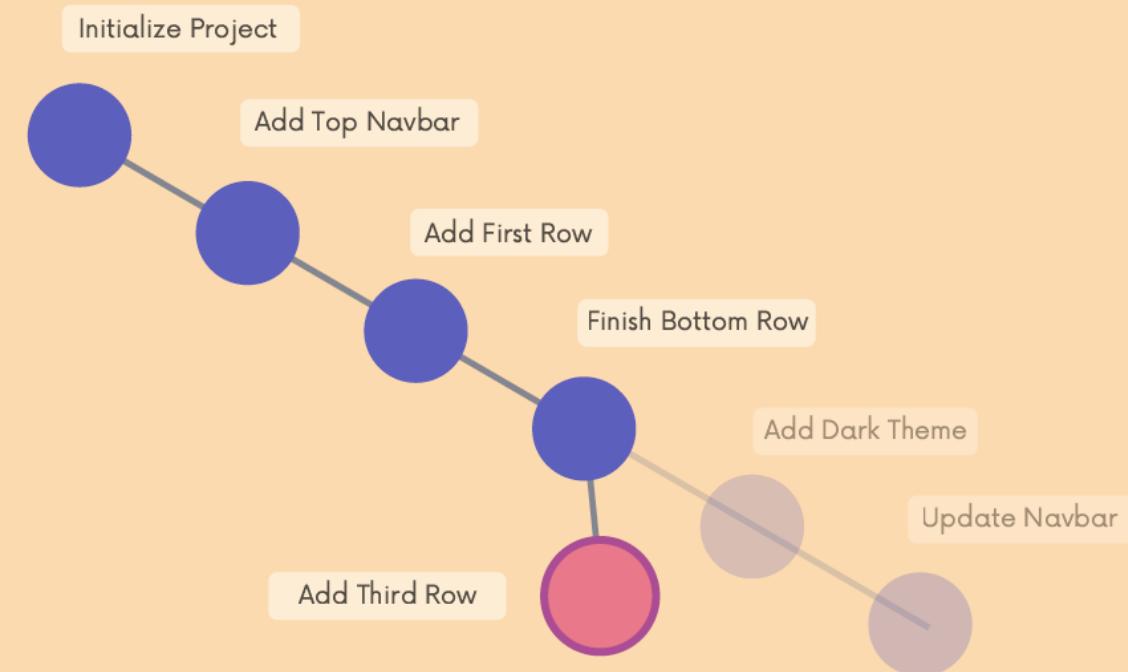


# I can even start more work off of an old checkpoint

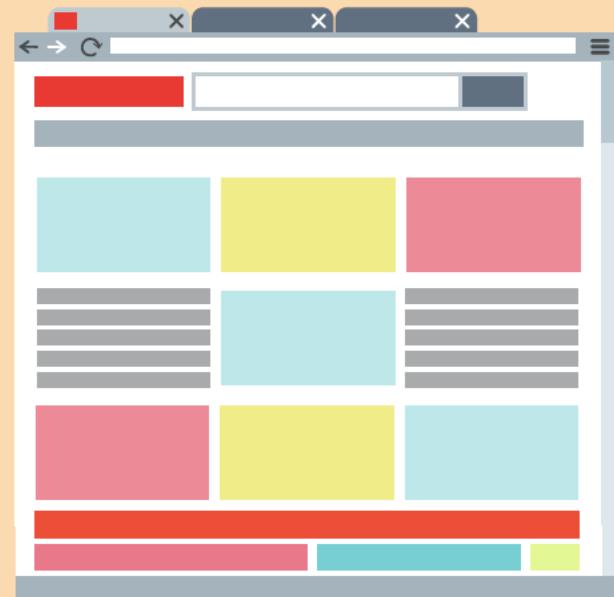
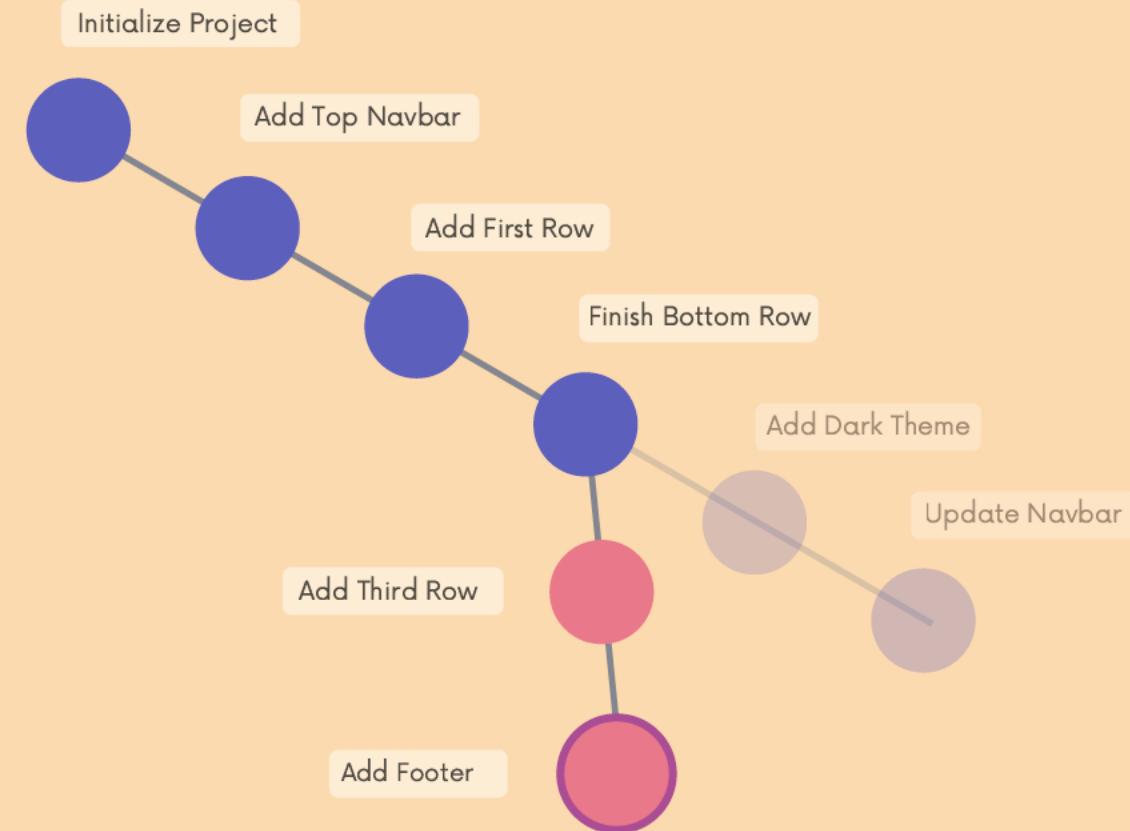


I add more content!

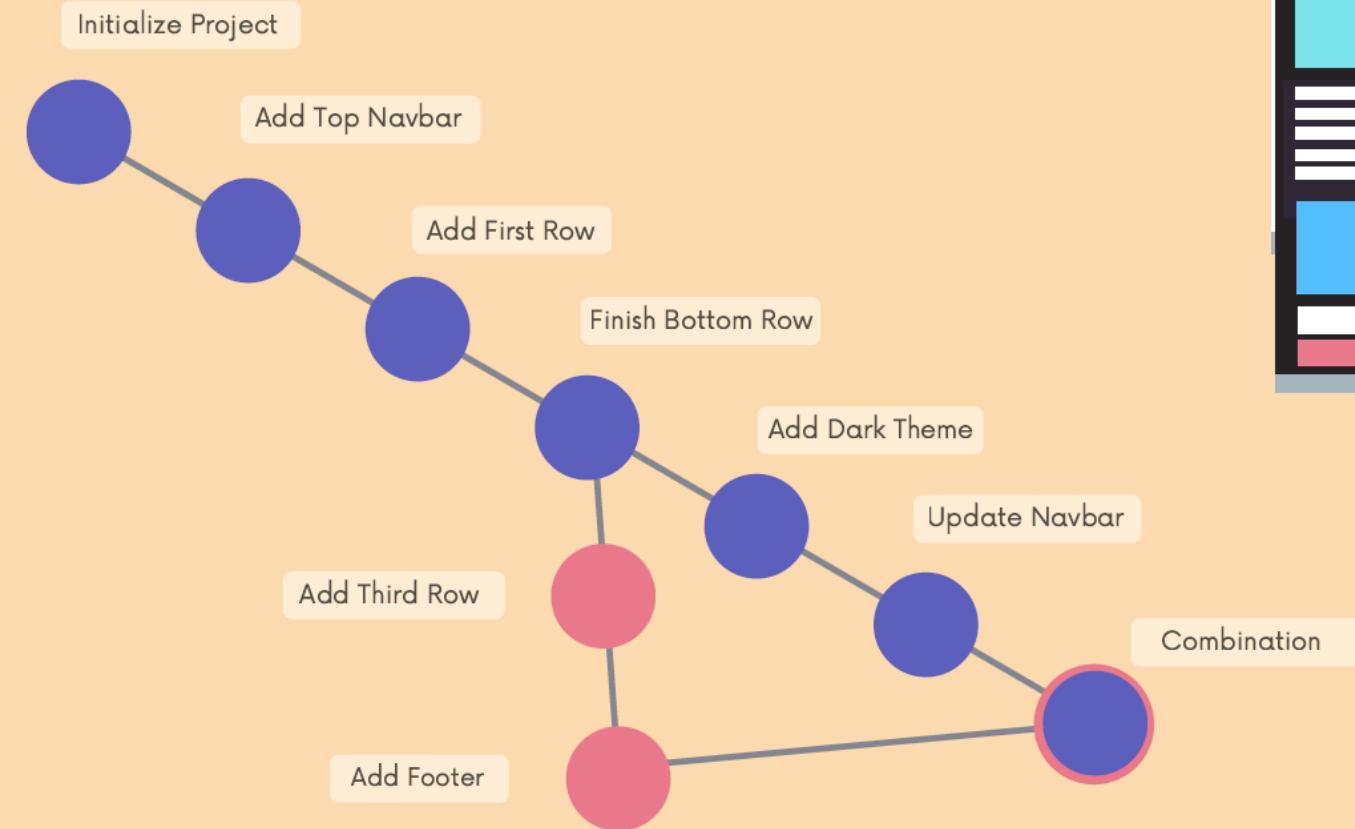
# I add a new checkpoint!



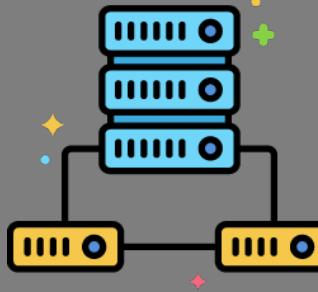
# Another checkpoint!



# And I can even combine checkpoints!



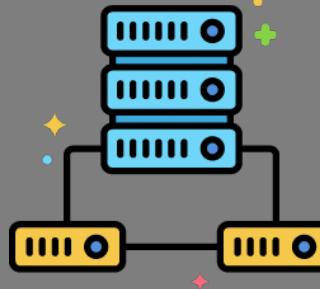
# Before Version Control System



# Before Version Control System



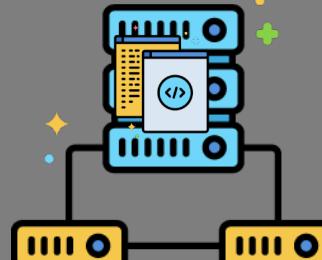
Version 1.0



# Before Version Control System



Version 1.0



running code version 1.0

# Before Version Control System

I have a great idea, send me your code



# Before Version Control System



# Before Version Control System

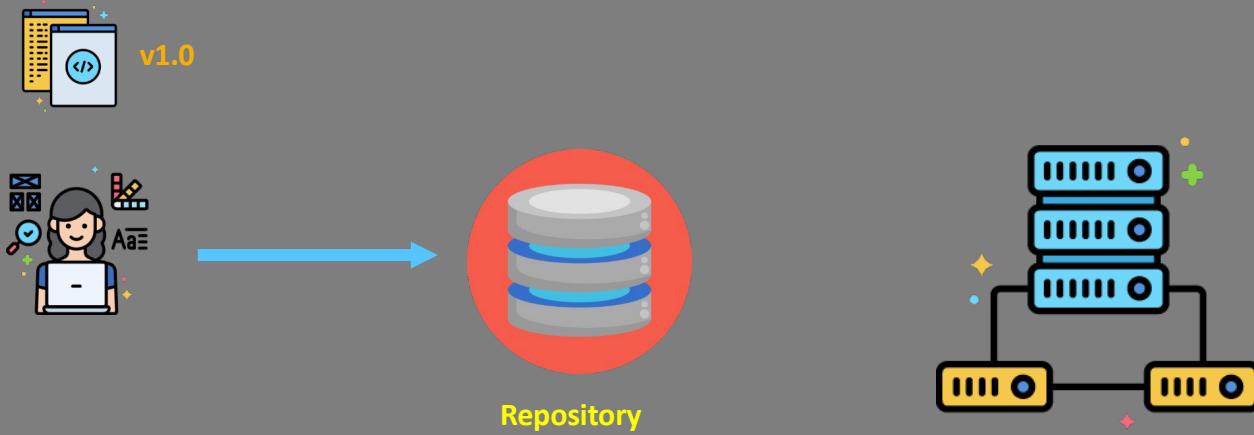


# Before Version Control System

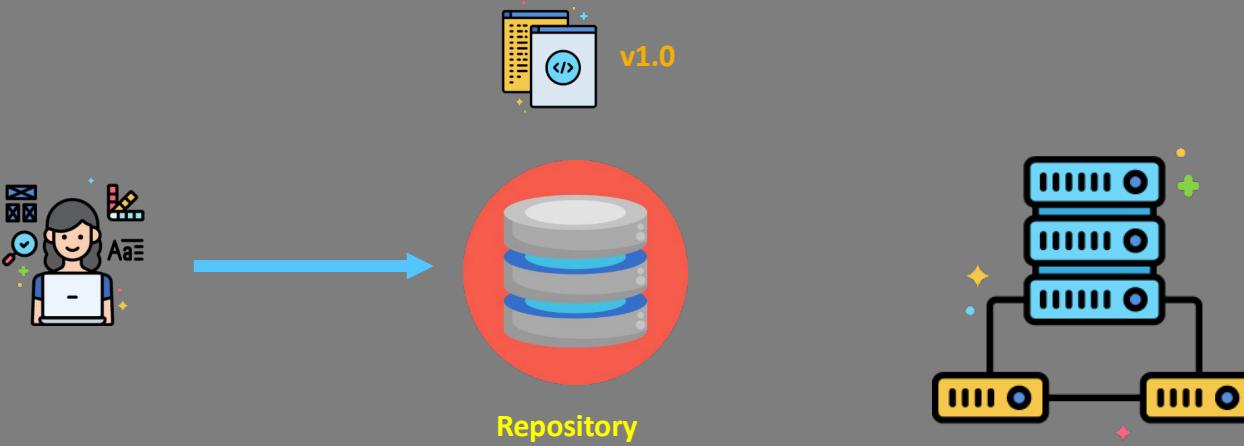


- Rollback is time consuming
- No audit tracking
- Not scalable for large teams

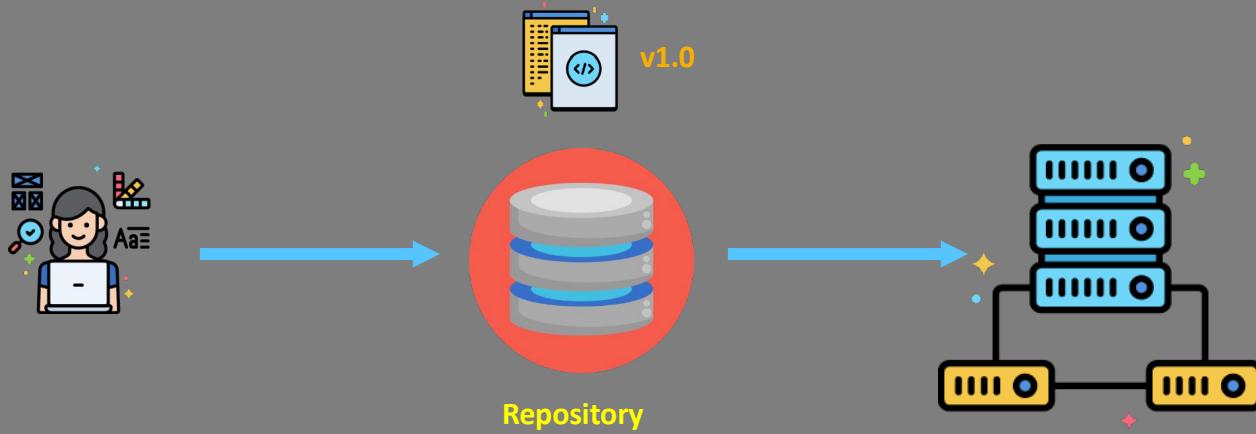
# Version Control System



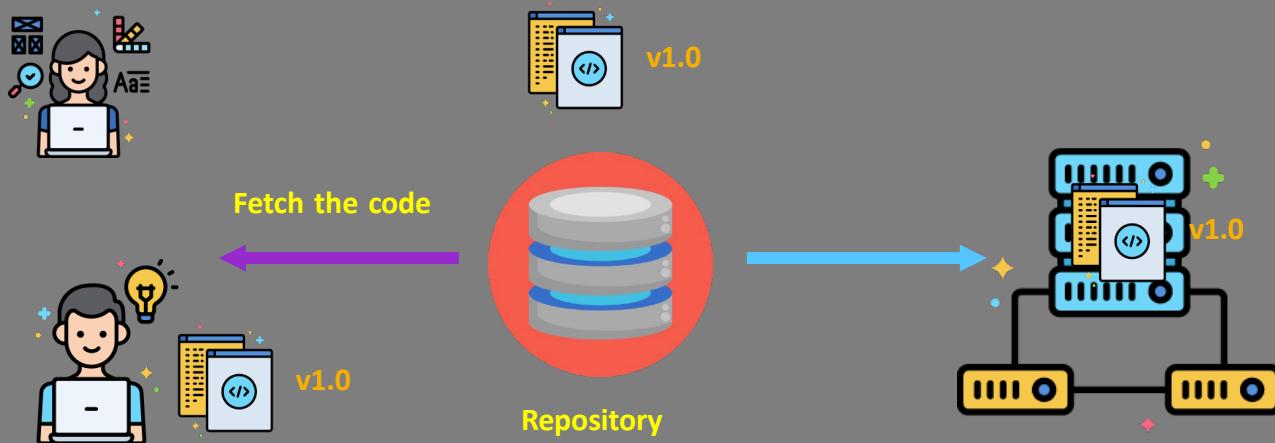
# Version Control System



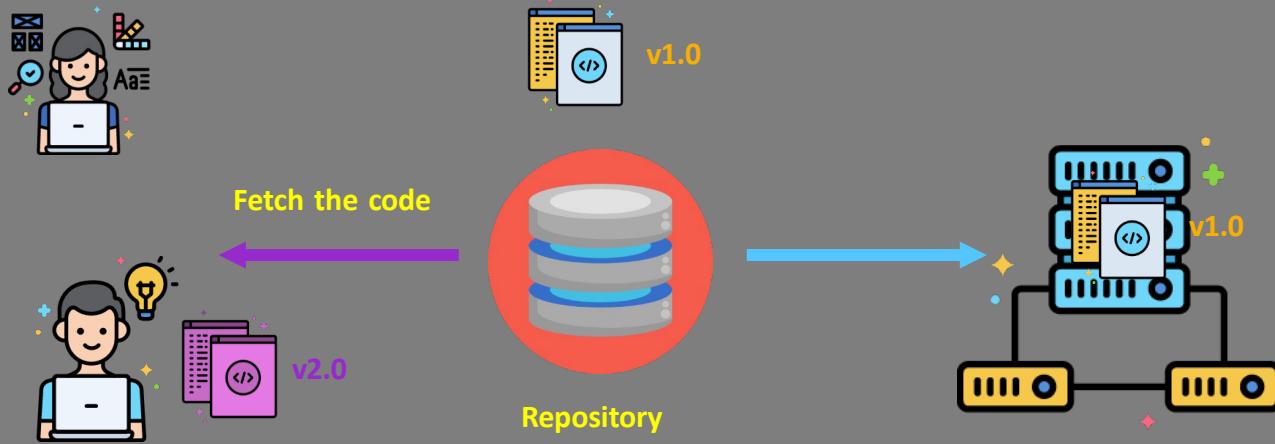
# Version Control System



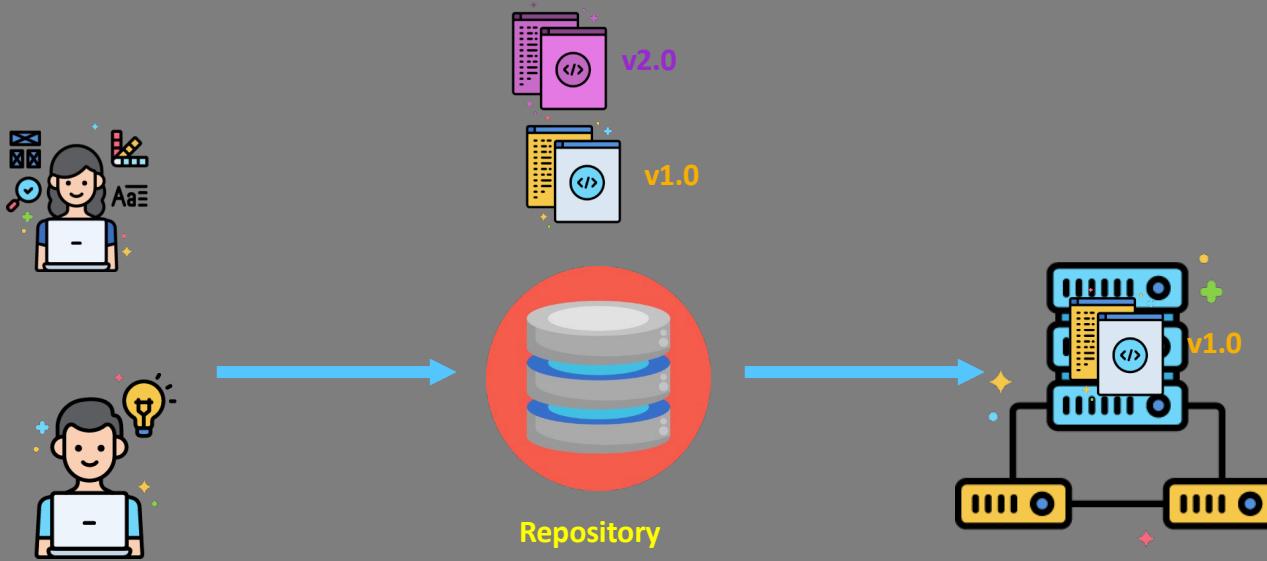
# Version Control System



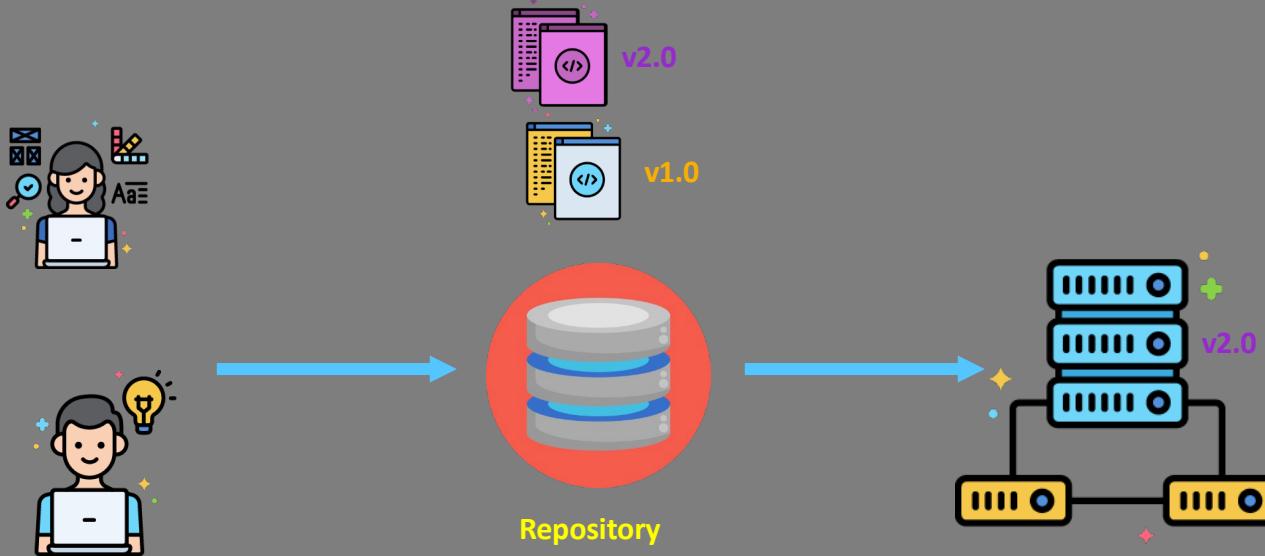
# Version Control System



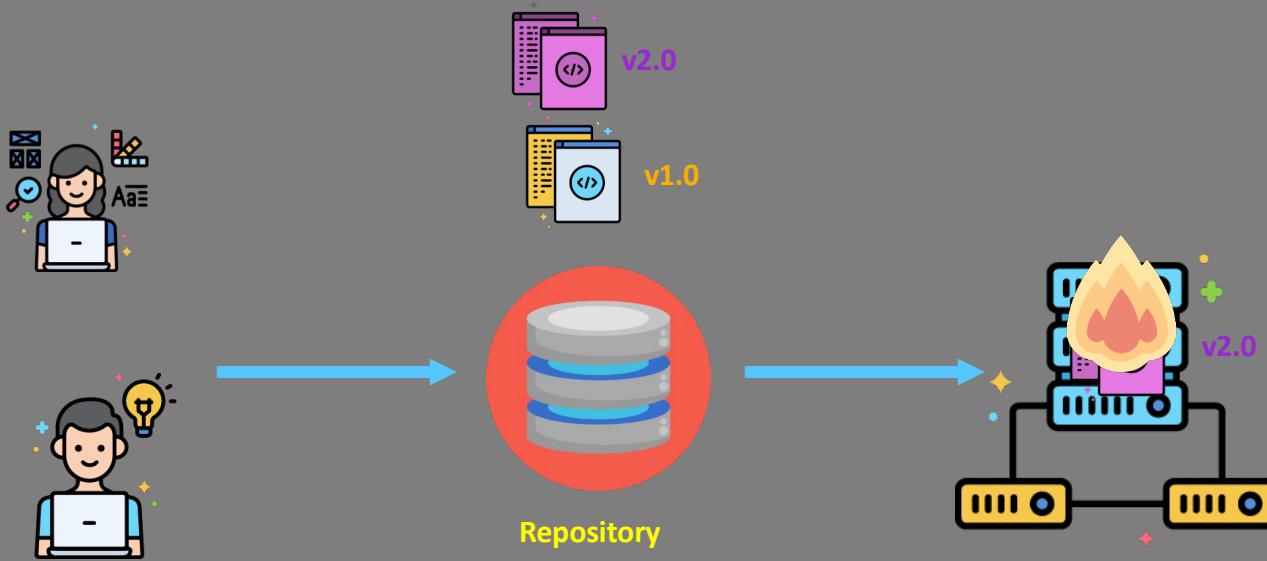
# Version Control System



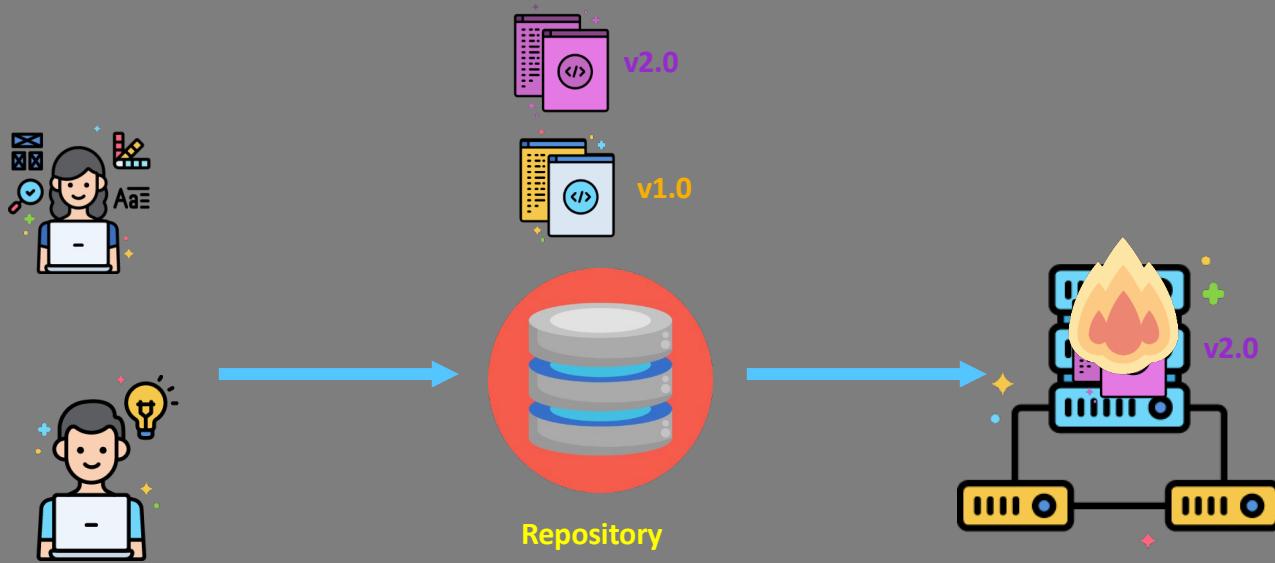
# Version Control System



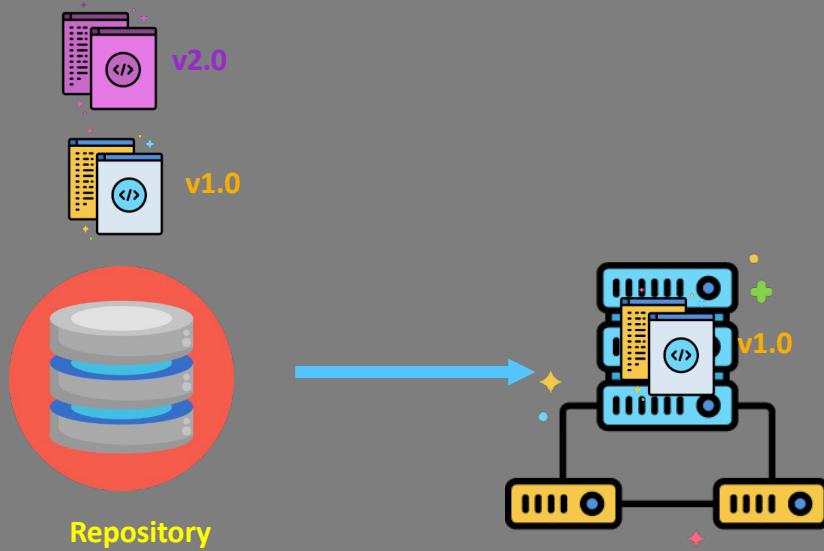
# Version Control System



# Version Control System



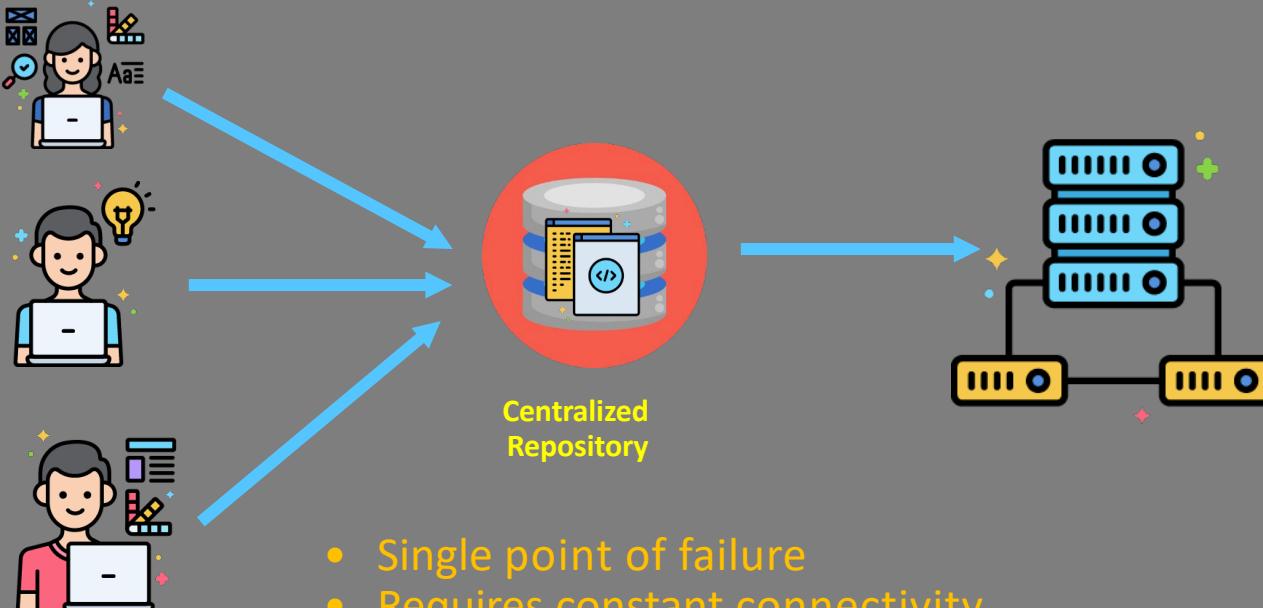
# Version Control System - Git



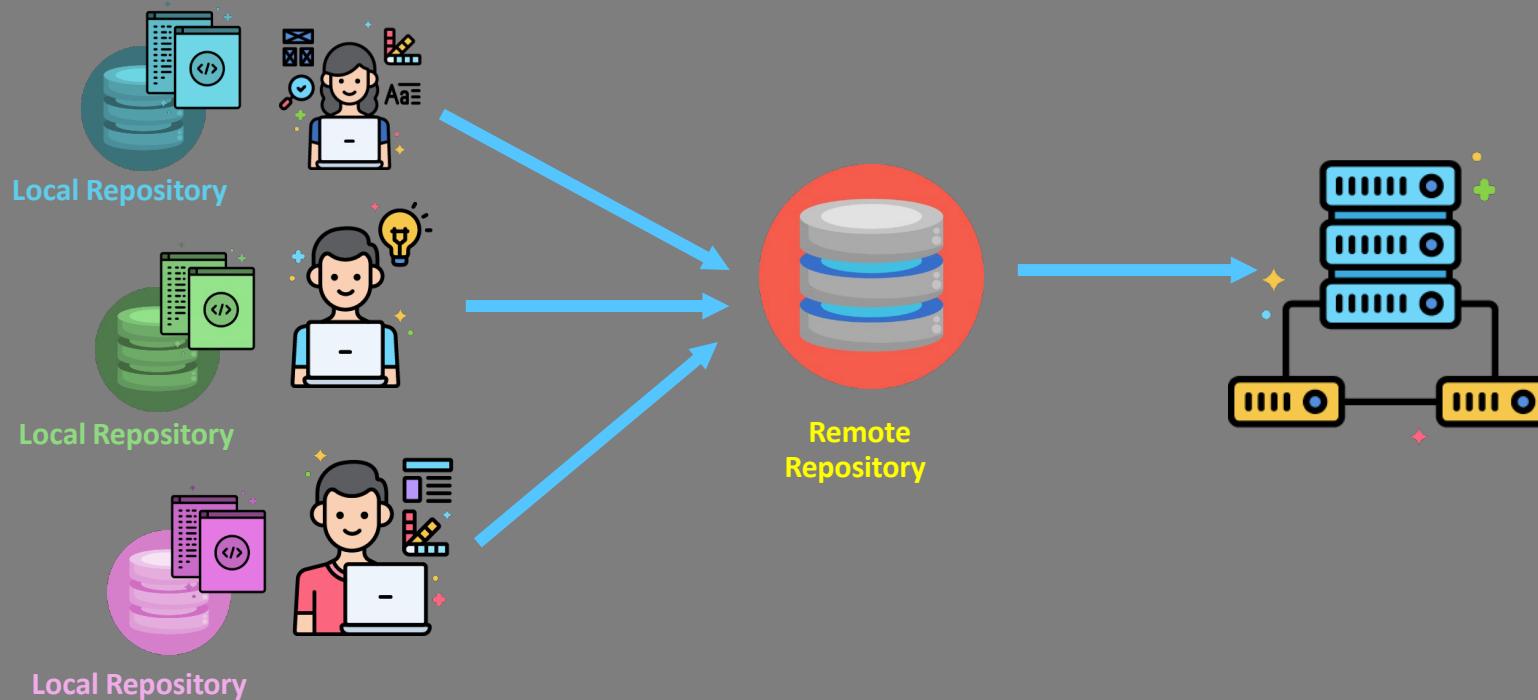
## Why Git?

- Distributed

# Centralized Version Control System

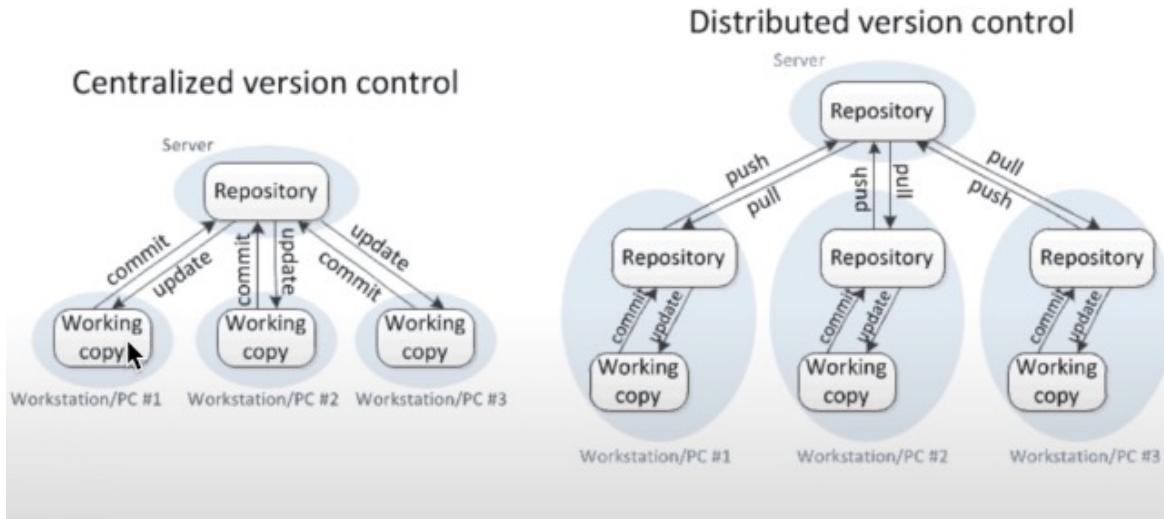


# Distributed Version Control System

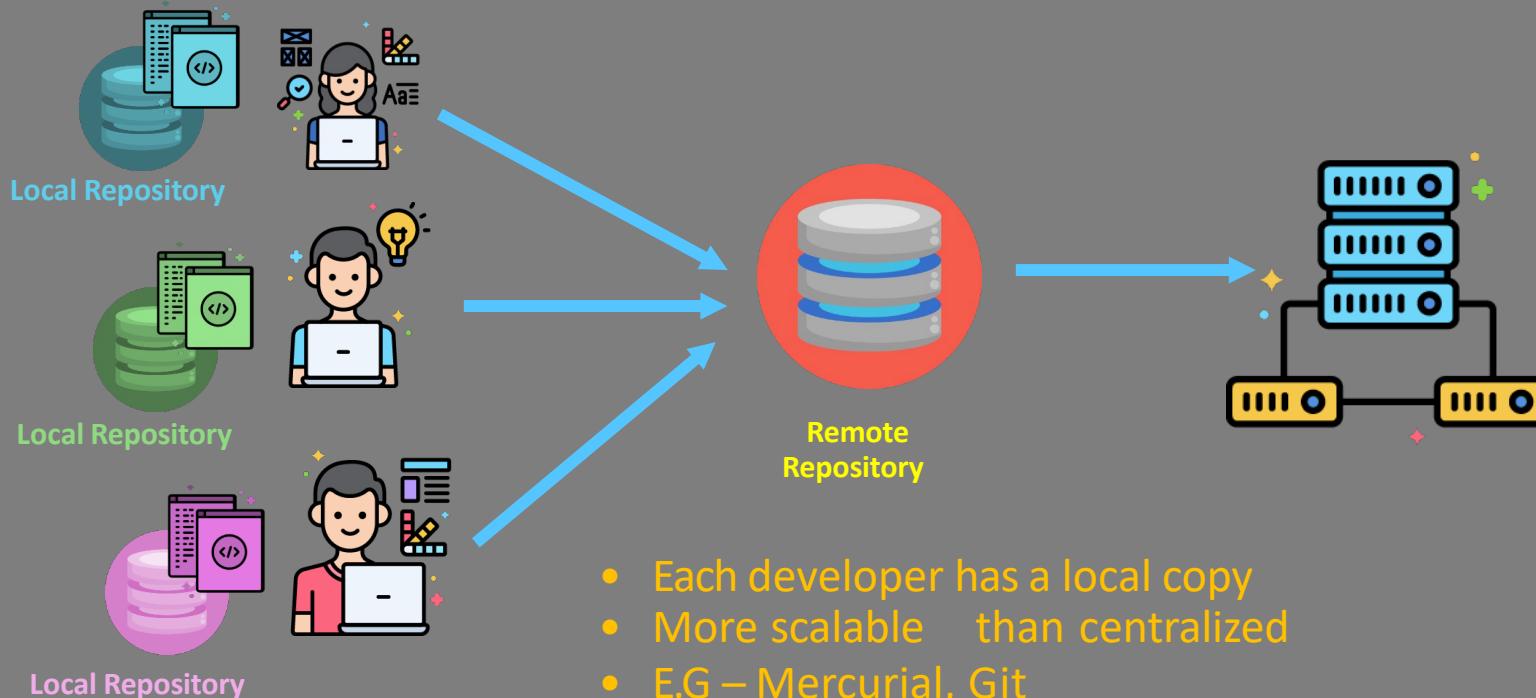


# Centralized vs Distributed

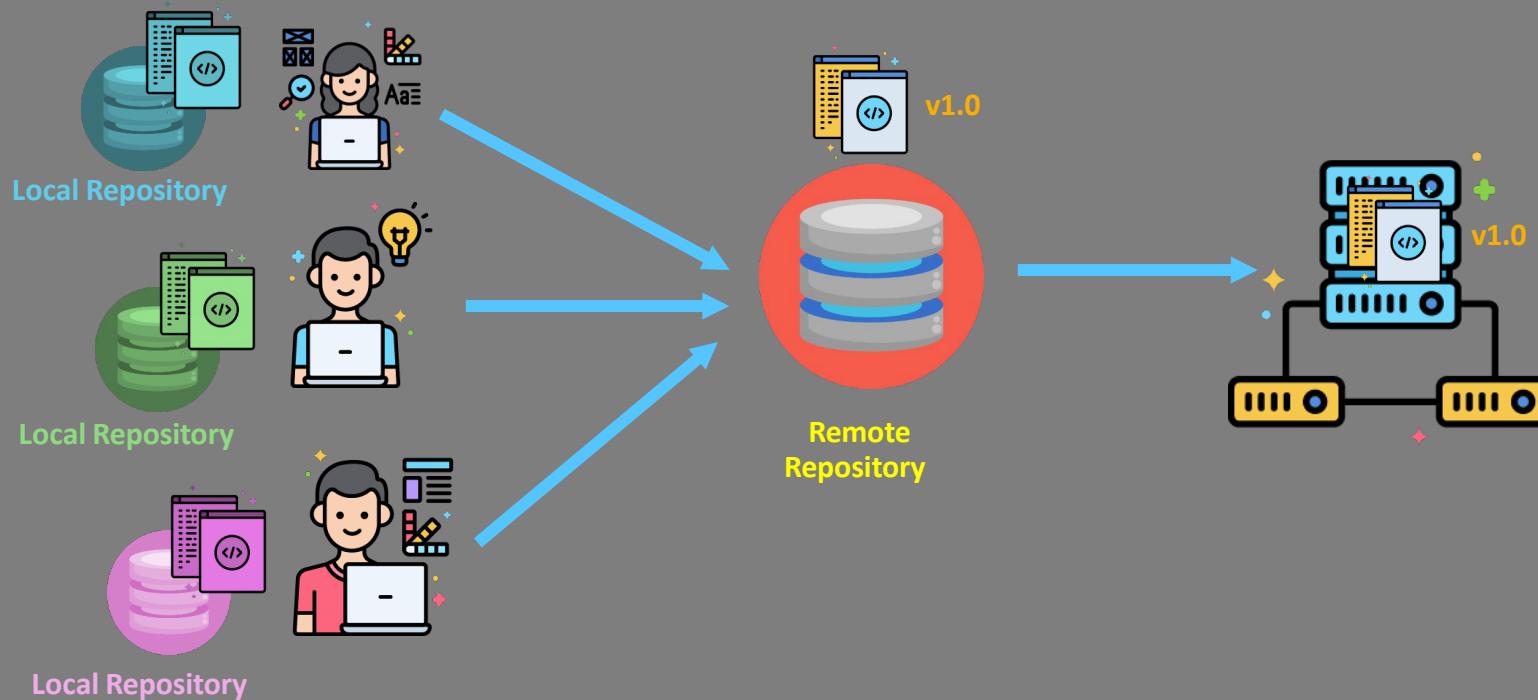
## Version Control



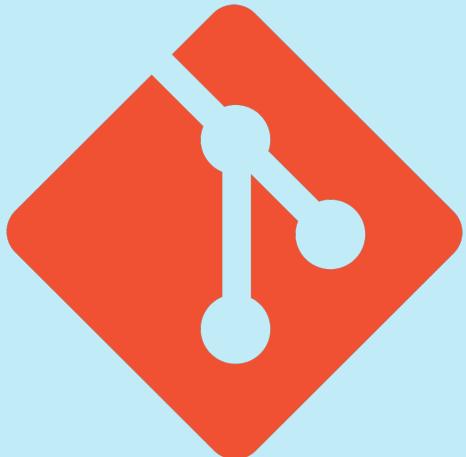
# Distributed Version Control System



# Distributed Version Control System



# Version Control System



## Why Git?

- Distributed
- Performant
- Detailed audit tracking
- Open source
  - Free!
  - Implemented with Kubernetes GitOps, integration with Jenkins and other DevOps tools
  - GitHub, GitLab, Code Commit are all based on Git

# Git vs GitHub

---

# Git Vs. GitHub

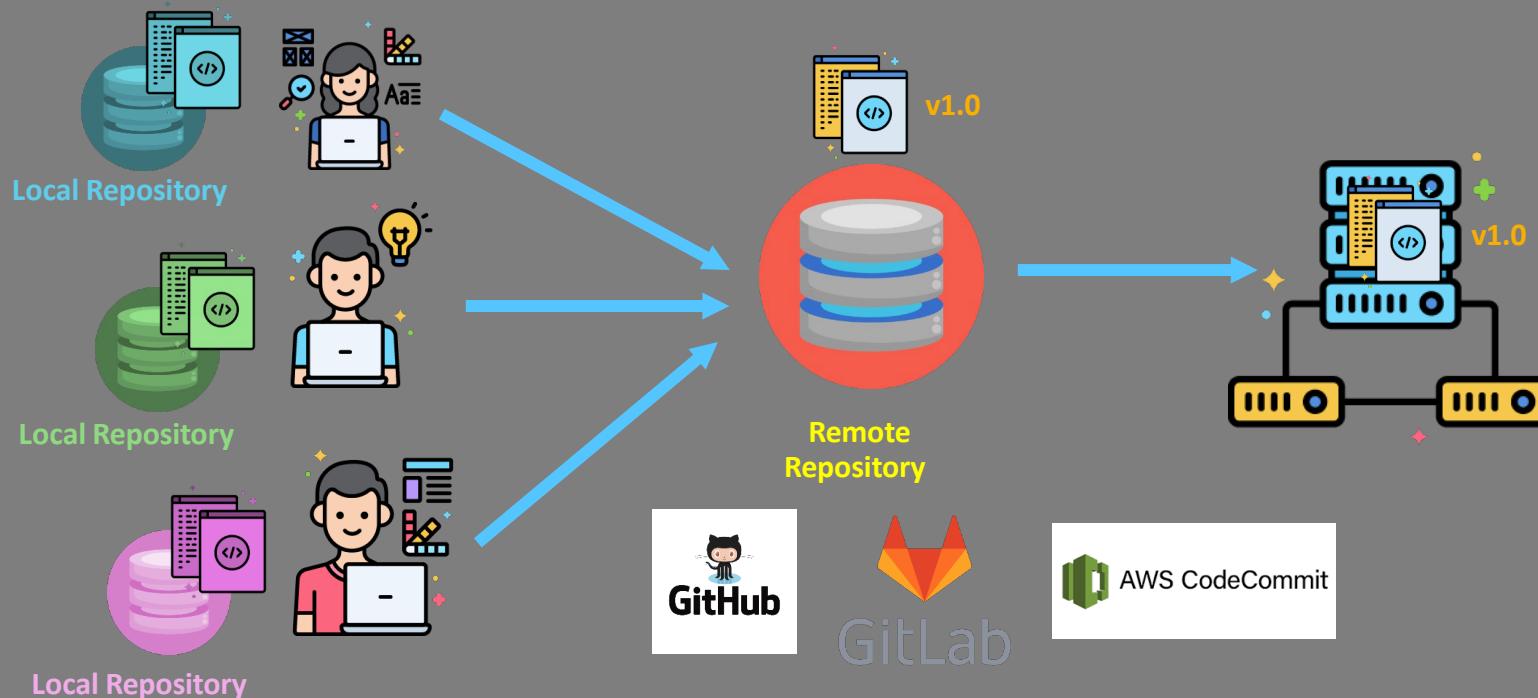


- Version Control System
- Installed locally on the system
- Created in 2005, by Linus Torvalds
- Open source, and used in multiple cloud repository services



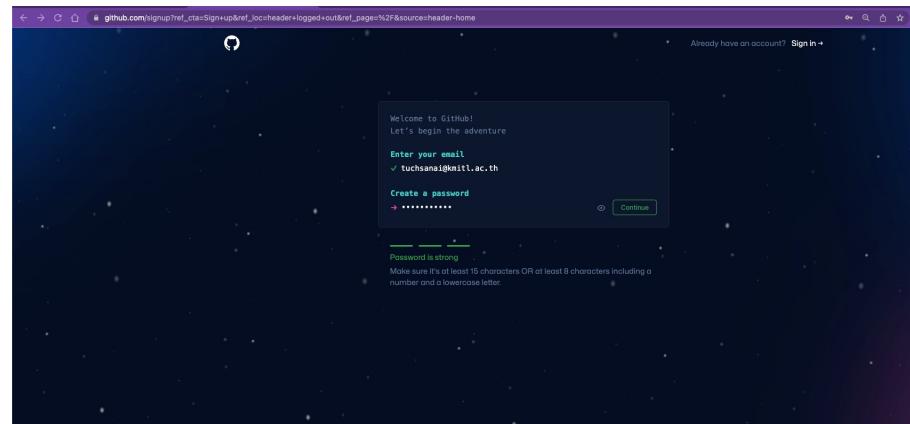
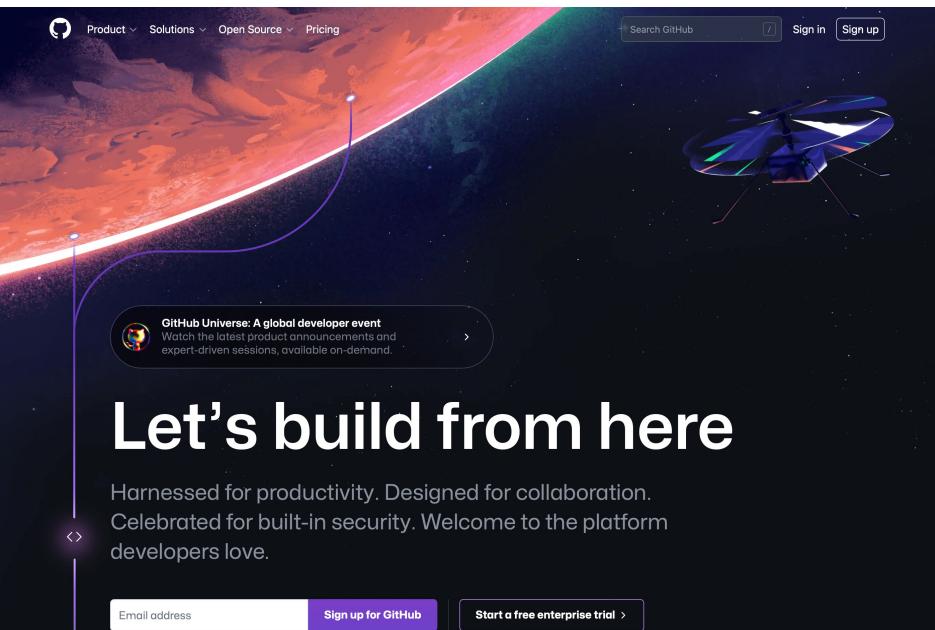
- Git repository hosting services with other features
- Runs on the cloud
- Created in 2008, currently owned by Microsoft
- Not open source, have free and paid tiers

# Distributed Version Control System



# Week 1 - Starting with Git

**Sign Up :** <https://github.com>



<https://github.com>

# Installing Git

# **Week 1 - Starting with Git**

- Let's install git on to your computer!
  - The installation process will be slightly different depending on your Operating System.

# **Week 1 - Starting with Git**

- MacOS or Linux Users:**

- Congrats! You already have Git installed on your machine since it comes pre-installed as part of your OS.
  - To confirm this, open up a terminal and type:
    - `git --version`
    - `>> git version 2.25.1 (Apple Git-128)`

# **Week 1 - Starting with Git**

- **MacOS or Linux Users:**
  - If you wish to update or re-install git, you can do this by simply selecting the MacOS or Linux links on the official git website:
    - **<https://git-scm.com/downloads>**

# Week 1 - Starting with Git

- **MacOS or Linux Users:**

- Now we'll be editing text files for this course, which means we need a text editor.
- If you're in this course, we'll assume you've used a text editor before, and often people have very strong opinions on a preferred text editor!

# Week 1 - Starting with Git

- **MacOS or Linux Users:**

- Our suggested text editor for this course is VS Code:
  - <https://code.visualstudio.com/>
- It's created by Microsoft and has direct integrations with GitHub and is one of the most popular text editors today.
- You can follow along with any text editor you prefer however.



# Week 1 - Starting with Git

- **Windows Users:**

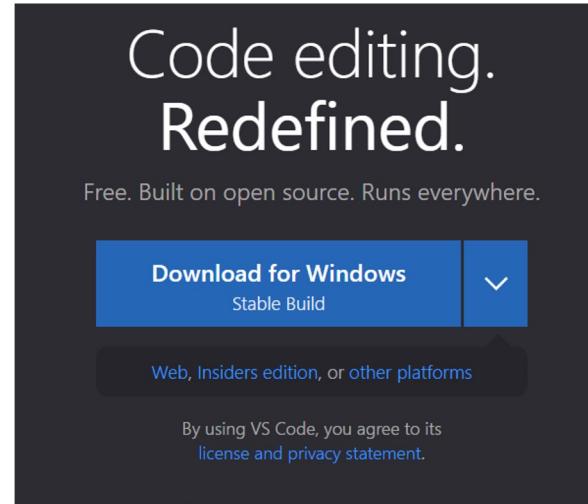
- Our *HIGHLY* recommend text editor for this course is VS Code:
  - **<https://code.visualstudio.com/>**
- Why *HIGHLY* recommended?
  - Windows + VS Code + GitHub
  - Upon installing git you will be asked to select a default editor, you'll need VS Code installed to select it as default.



# Week 1 - Starting with Git

- **Windows Users:**

- Go to:
  - <https://code.visualstudio.com/>
- Download with Default Settings:





# Week 1 - Starting with Git

- **Windows Users:**

- Next we'll download git, go to:
  - **<https://git-scm.com/>**

The screenshot shows the 'Downloads' section of the git-scm.com website. At the top, there's a navigation bar with links for 'About', 'Documentation', 'Downloads', 'GUI Clients', and 'Logos'. Below this, there's a sidebar with links for 'Community', 'Pro Git book', 'GUI Clients', and 'Logos'. The main content area features a large image of a Mac desktop with a Git interface. It displays the text 'Latest source Release 2.38.1' and 'Release Notes (0022-10-07)'. Below this, there are sections for 'macOS', 'Windows', and 'Linux/Unix'. A note states that older releases are available on GitHub. To the right, there are sections for 'GUI Clients' (with a link to 'View GUI Clients') and 'Logos' (with a link to 'View Logos'). At the bottom, there's a section for 'Git via Git' with a note about getting the latest development version via Git itself.

**DAY 1**

**Configure Git**



# Week 1 - Starting with Git

- So far we've:
  - Installed Git
  - Created a GitHub Account Profile
  - Installed GitHub Desktop and VS Code
- What left for Day 1:
  - Configure Git Locally
  - Create a Repository
  - Explore VS Code Integrations
  - Exercise and Solution

## Week 1 - Starting with Git

- Take careful note of the user name and email address you register with at GitHub, ideally it will be the same username and email you configure git with locally.
- We can technically use any username/email we want, but your history of “commits” (changes to code) will be saved in the public log of changes in the repository.

# Week 1 - Starting with Git

- In this lecture we will set-up a name and email address on our local installation of Git.
- If you only ever used Git locally by yourself then this username and email would just be stored on your local historical logs.
- However if you end up working with others and using GitHub, this information will be useful to identify who did what.

# Week 1 - Starting with Git

- You can check the current configuration with the commands:
  - **git config user.name**
  - **git config user.email**
- The configuration commands will be:
  - **git config --global user.name “user”**
  - **git config --global user.email “email”**
- If switch with another github account
  - **git config --global user.name “user”**
  - **git config --global user.email “email”**
  - **git config --global credential.username “user”**

Show global Git configuration?

```
git config --list or git config -l
```

or look at your `~/.gitconfig` file. The local configuration will be in your repository's `.git/config` file.

```
git config --list --show-origin
```

# Week 1 - Starting with Git

- Let's head over to our command line interface to set-up our Git configuration:
  - Git Bash
  - Terminal
  - Command Prompt

# DAY 1

# Creating a Git Repository

# Day 1 - Starting with Git

- How can we create a Git Repository?
  - **git init**
    - This command initializes a Git Repository on your local machine.
    - You only need to run this command once per project.
  - **git status**
    - This command will report back the status of your Git repository.

# Day 1 - Starting with Git

- How can we create a Git Repository?
  - Upon creating a repository with **git init** you will create a hidden .git file.
  - The .git file is a hidden file that manages the versioning of the files inside the Git repository.

# Day 1 - Starting with Git

- Git inside a Folder/Directory:
  - Upon creating a Git Repository, all the folders/directories inside the top level Git Repository will also be part of that Repository, meaning all the changes are tracked.

# Day 1 - Starting with Git

- Git inside a Folder/Directory:
  - Upon creating a Git Repository, all the folders/directories inside the top level Git Repository will also be part of that Repository, meaning all the changes are tracked.



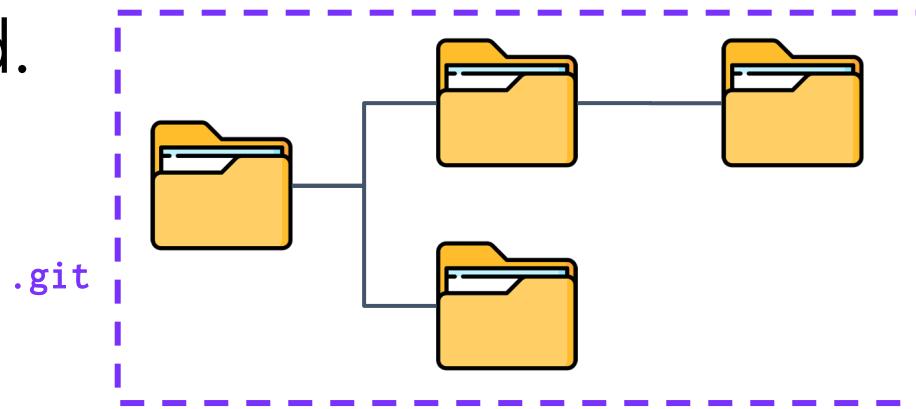
# Day 1 - Starting with Git

- Git inside a Folder/Directory:
  - Upon creating a Git Repository, all the folders/directories inside the top level Git Repository will also be part of that Repository, meaning all the changes are tracked.



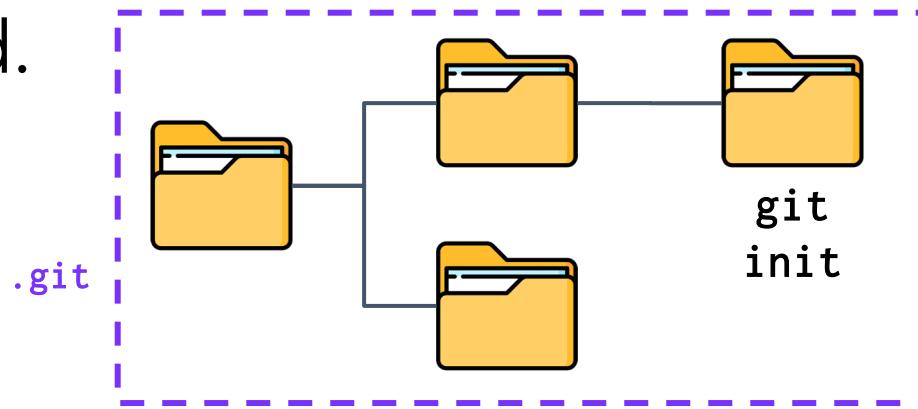
# Day 1 - Starting with Git

- Git inside a Folder/Directory:
  - Upon creating a Git Repository, all the folders/directories inside the top level Git Repository will also be part of that Repository, meaning all the changes are tracked.



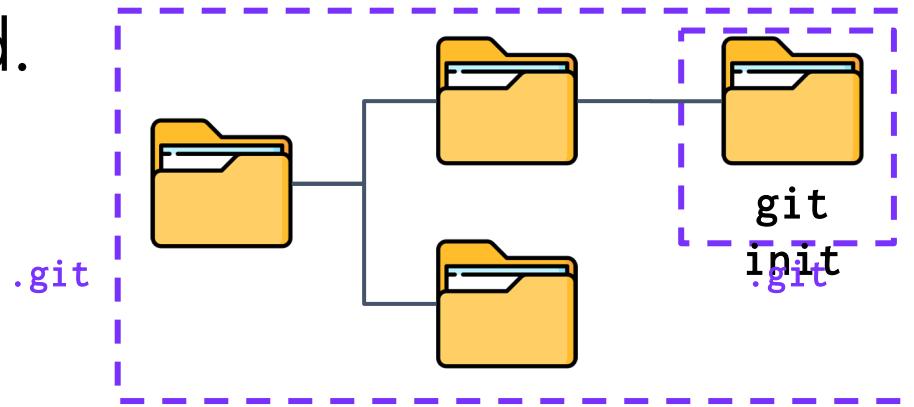
# Day 1 - Starting with Git

- Git inside a Folder/Directory:
  - Upon creating a Git Repository, all the folders/directories inside the top level Git Repository will also be part of that Repository, meaning all the changes are tracked.



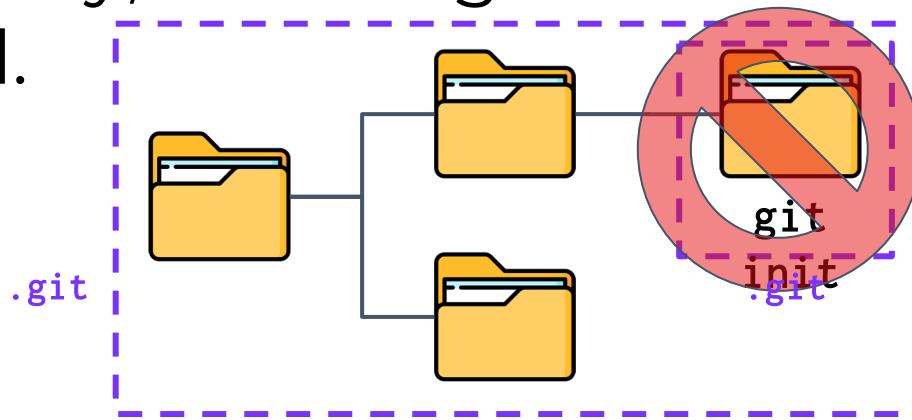
# Day 1 - Starting with Git

- Git inside a Folder/Directory:
  - Upon creating a Git Repository, all the folders/directories inside the top level Git Repository will also be part of that Repository, meaning all the changes are tracked.



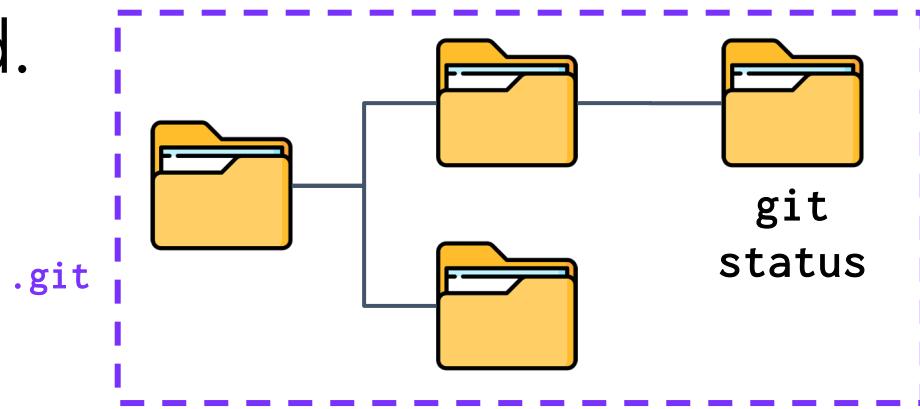
# Day 1 - Starting with Git

- Git inside a Folder/Directory:
  - Upon creating a Git Repository, all the folders/directories inside the top level Git Repository will also be part of that Repository, meaning all the changes are tracked.



# Day 1 - Starting with Git

- Git inside a Folder/Directory:
  - Upon creating a Git Repository, all the folders/directories inside the top level Git Repository will also be part of that Repository, meaning all the changes are tracked.



# Ignoring Files

We can tell Git which files and directories to ignore in a given repository, using a `.gitignore` file. This is useful for files you know you NEVER want to commit, including:

- Secrets, API keys, credentials, etc. Operating
- System files (`.DS_Store` on Mac) Log files
- Dependencies & packages



# .gitignore

Create a file called .gitignore in the root of a repository. Inside the file, we can write patterns to tell Git which files & folders to ignore:

- `.DS_Store` will ignore files named `.DS_Store`
- `folderName/` will ignore an entire directory
- `*.log` will ignore any files with the `.log` extension

<https://www.toptal.com/developers/gitignore>



› **gitignore.io**

สร้างไฟล์ .gitignore ที่มีประโยชน์สำหรับโปรเจกต์ของคุณ

ซอฟต์แวร์ | ผลงานเอกสาร คำอธิบาย คอมมานต์ไลน์ คู่มือ!



# Day 1 - Starting with Git

- How can we create a Git Repository?
  - We can also use the Graphical Interface with GitHub Desktop or we can even create a new repository online at [www.github.com](http://www.github.com).
  - Then we can **git clone** this repository to our local machine.

# Day 1 - Starting with Git

- Let's create our first local Git repository at the command line.
- Then we'll create a repository on GitHub and use **git clone** to clone it to our local computer.
  - We'll need to set-up some tokens in order to clone private repositories.

**DAY 1**

# **Private Repositories and Tokens**

# Day 1 - Starting with Git

- Clone Syntax with PAT:

```
git clone https://username:YOUR_TOKEN@github.com/username/repo.git
```

- Previously we used:

```
git clone https://github.com/account/repo.git
```

# Create a Personal Access Token

- Log in to your GitHub account that you want to create the PAT for.
- Click on your profile picture in the upper-right corner of the screen and select "Settings."
- In the left sidebar, select "Developer settings."
- Under the "Developer settings" menu, choose "Personal access tokens."
- Click on the "Generate token" button.
- You'll be taken to the "Create a new personal access token" page. Please provide the following information:
  - "Note": Enter a name for your token to specify its purpose (e.g., "GitHub API access").
  - "Expiration": You can optionally set an expiration date for the PAT (not required).
  - "Select scopes": You need to choose the scopes you want this PAT to have access to (e.g., read repository data, write repository data, read user data, etc.).

You unlocked new Achievements with private contributions! Show them off by including private contributions in your Profile in settings.

Pinned

Customize your pins

746 contributions in the last year

Contribution settings ▾

Mon    Wed    Fri

Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep

A large yellow number 1 is displayed in the upper right corner, with a red arrow pointing upwards towards the user profile picture icon in the top right of the header.

1

Click on your profile picture in the upper-right corner of the screen and select "Settings."

Set status

Your profile

Your repositories

Your projects

Your codespaces

Your organizations

Your enterprises

Your stars

Your sponsors

Your gists

Upgrade

Try Enterprise

Try Copilot

Feature preview

Settings

GitHub Docs

GitHub Support

Sign out

A large yellow number 2 is displayed in the bottom right corner, with a red rectangular box highlighting the "Settings" link in the sidebar menu.

2

**3** In the left sidebar, select "Developer settings".

This screenshot shows the GitHub Public profile settings page. The sidebar on the left contains various account management options like Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, Emails, Password and authentication, Sessions, SSH and GPG keys, Organizations, Enterprises, and Moderation. Under Code, planning, and automation, there are links for Repositories, Codespaces, Packages, Copilot, Pages, Saved replies, and Security. The main area displays the user's public profile information, including their name (tuchsanai), profile picture (a white cat), and bio ('Tell us a little bit about yourself'). It also includes sections for pronouns (Don't specify) and social accounts (Link to social profile). A note at the bottom states that all fields are optional and can be deleted at any time.

**Update profile**

**4**

This screenshot shows the GitHub Developer Settings page under the GitHub Apps section. The "Personal access tokens" option is highlighted with a red box. The page includes a large yellow number 4 and a note encouraging users to build GitHub Apps.

**5**

This screenshot shows the GitHub Developer Settings page under the Personal access tokens section. The "Tokens (classic)" option is highlighted with a red box. A button labeled "Generate new token" is visible on the right. The page includes a large yellow number 5 and a note explaining that personal access tokens function like ordinary OAuth access tokens.

**6**

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

**Note**

test

What's this token for?

**Expiration \***

7 days      The token will expire on Tue, Sep 26 2023

**Select scopes**

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories
<input checked="" type="checkbox"/> <b>repo:status</b>	Access commit status
<input checked="" type="checkbox"/> <b>repo_deployment</b>	Access deployment status
<input checked="" type="checkbox"/> <b>public_repo</b>	Access public repositories
<input checked="" type="checkbox"/> <b>repo:invite</b>	Access repository invitations
<input checked="" type="checkbox"/> <b>security_events</b>	Read and write security events

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories Access commit status Access deployment status Access public repositories Access repository invitations Read and write security events
<input type="checkbox"/> <b>workflow</b>	Update GitHub Action workflows
<input type="checkbox"/> <b>write:packages</b>	Upload packages to GitHub Package Registry Download packages from GitHub Package Registry
<input type="checkbox"/> <b>delete:packages</b>	Delete packages from GitHub Package Registry
<input type="checkbox"/> <b>admin:org</b>	Full control of orgs and teams, read and write org projects Read and write org and team membership, read and write org projects Read org and team membership, read org projects Manage org runners and runner groups
<input type="checkbox"/> <b>admin:public_key</b>	Full control of user public keys Write user public keys Read user public keys
<input checked="" type="checkbox"/> <b>admin:repo_hook</b>	Full control of repository hooks Write repository hooks Read repository hooks
<input type="checkbox"/> <b>admin:org_hook</b>	Full control of organization hooks
<input type="checkbox"/> <b>gist</b>	Create gists
<input type="checkbox"/> <b>notifications</b>	Access notifications
<input type="checkbox"/> <b>user</b>	Update ALL user data Read ALL user profile data Access user email addresses (read-only) Follow and unfollow users

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens Beta

Tokens (classic)

## Personal access tokens (classic)

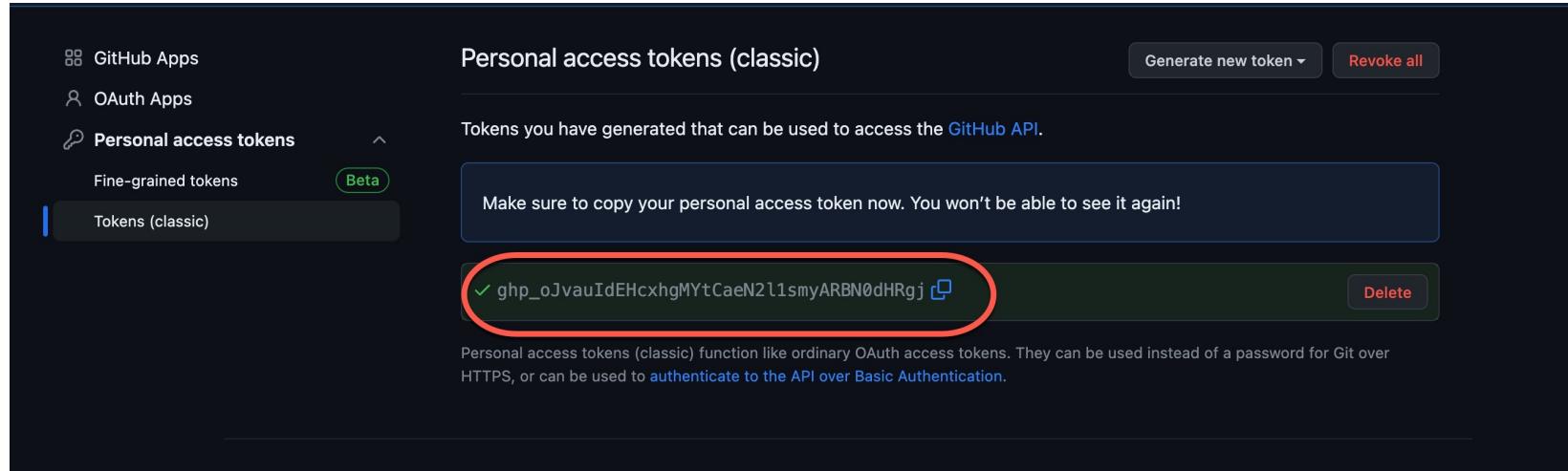
Generate new token ▾ Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp\_oJvauIdEHcxhgMYtCaeN2l1smyARBN0dHRgj [Copy](#) Delete

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).



# DAY 1

# Summary and Exercise



# Day 1 - Starting with Git

- **Exercise Tasks:**
  - Create a new Private Repository on GitHub.
  - Initialize your repository with README, license and gitignore.
  - Clone your Repository using the Command Line and a PAT.