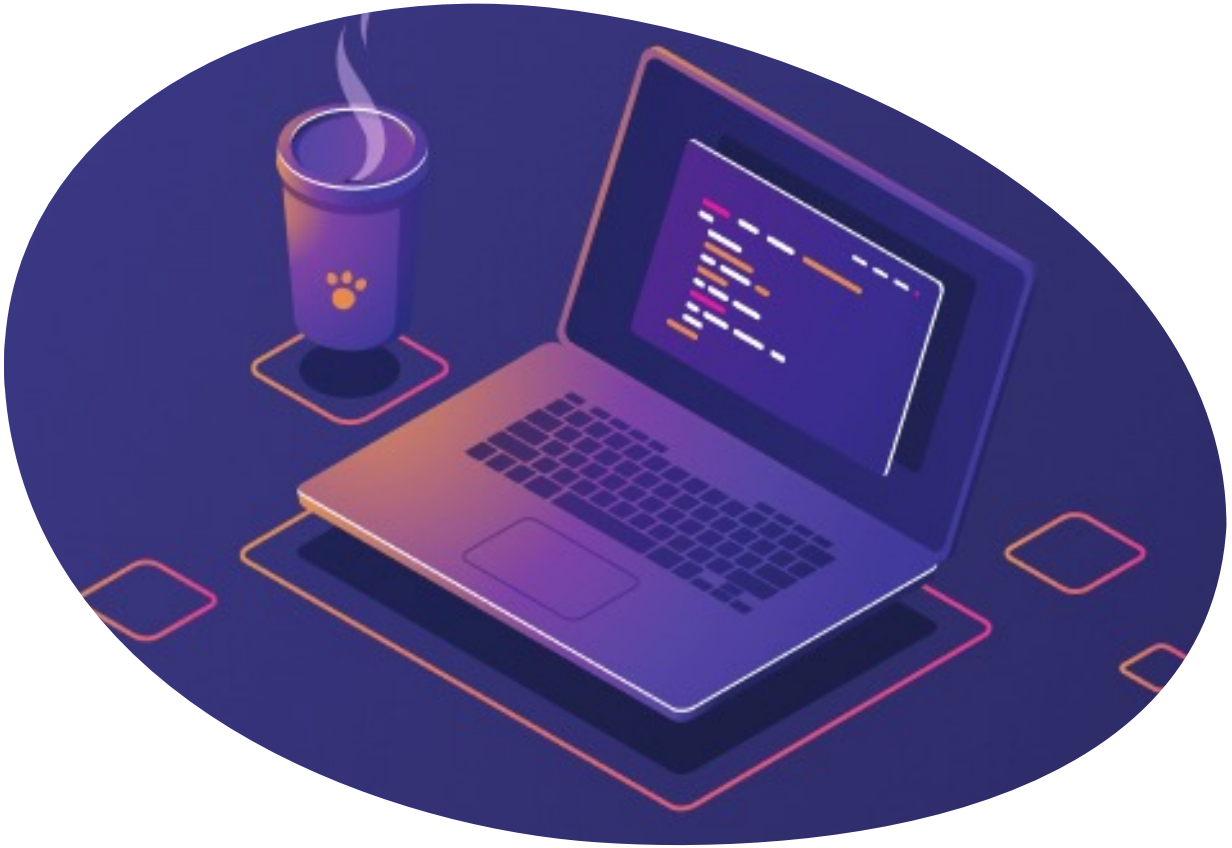


Document

Group

Week1 : SOFTWARE DEVELOPMENT TOOLS AND ENVIRONMENTS

แผนการสอน	
สัปดาห์ที่	หัวข้อ
1	Git and GitHub Configure Git Creating and Cloning Repositories Private Repositories and Token Lab week 1
2	Understanding Git Usage and Workflow Add and Commit Git Log Git Remote and Git Push Fetch and Pull Lab week 2
3	Understanding Branches Understanding HEAD Git Branch Commands Delete and Rename Branches Merging Branches - Theory and Concepts Merging Branches in Practice Git Diff Lab week 3
4	Git with Going back and Undoing Changes -Git Checkout and Detached HEAD -Git Restore, Git Reset, Git Revert Undoing Changes - Exercise and Solution
5	Docker Docker Overview Basic Docker Commands Docker Run Docker Images Environment Variables Command vs Entrypoint Labs - Docker Images Labs - Environment Variables Labs - Command vs Entrypoint
6	Docker Compose Docker Registry Lab: Docker Registry Labs: Docker Compose
7	Docker Engine Docker Storage Docker Networking Labs - Docker Storage Labs - Docker Networking
8	Docker Swarm Docker Service Docker Stacks CI/CD - Docker Integration Lab

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

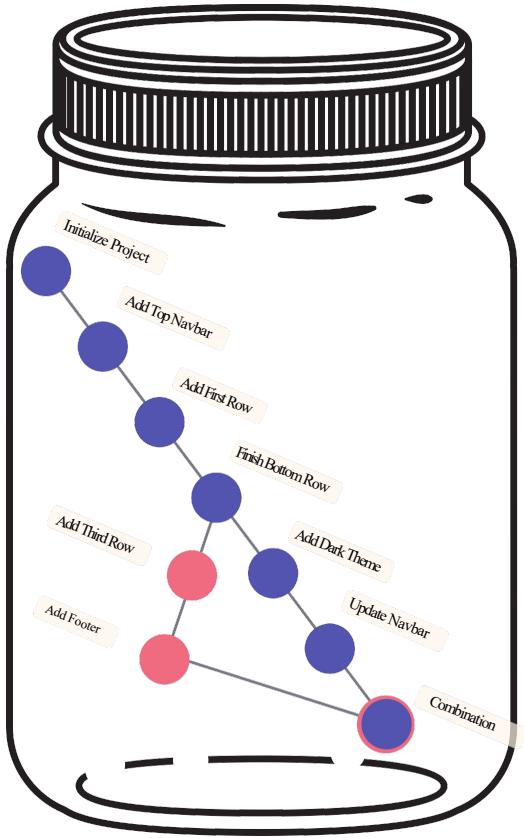
กลางภาค	30
ปลายภาค	30
LAB	20
Mini project	20

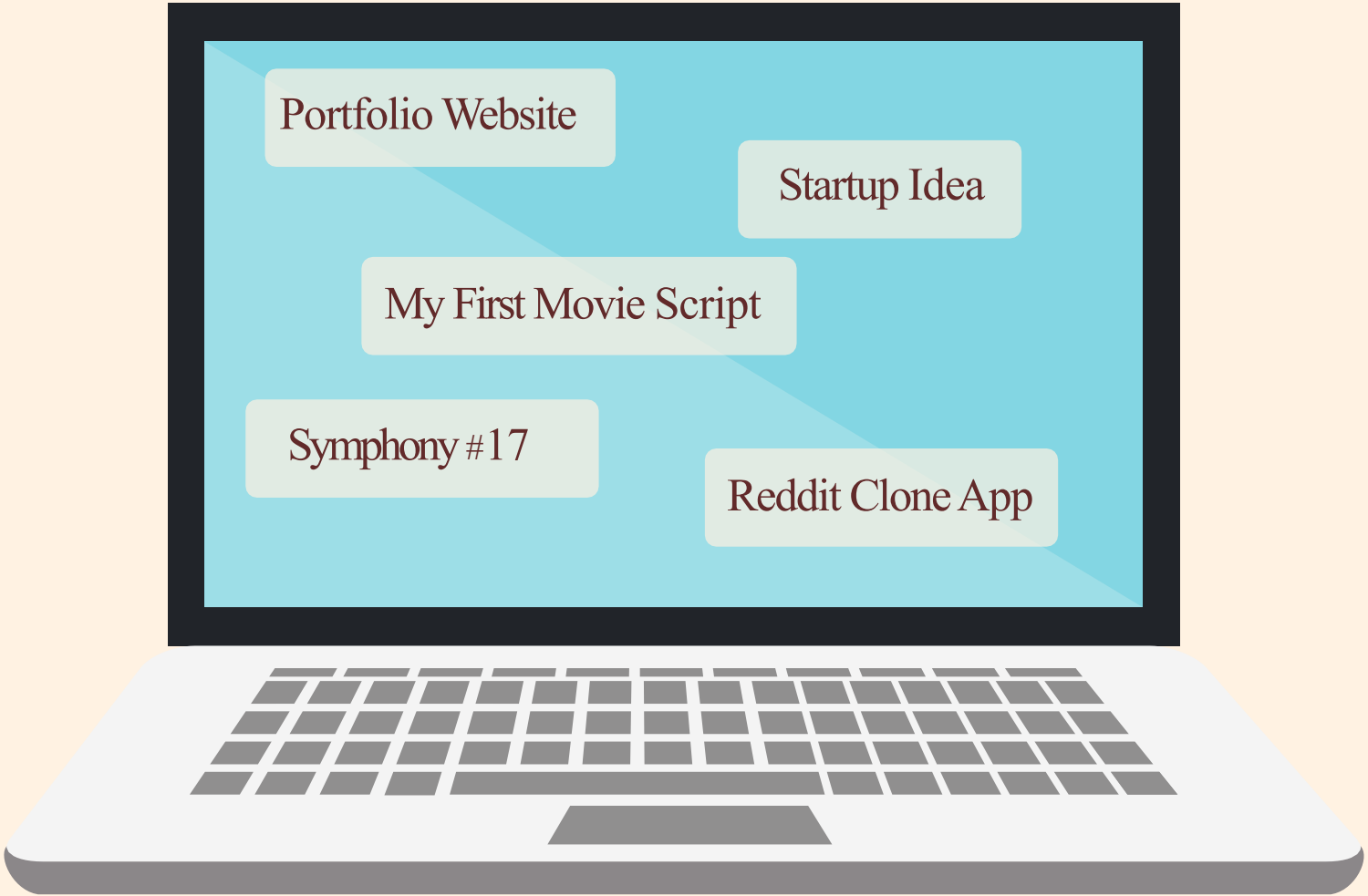
- Git was developed in 2005 by **Linus Torvalds**
- Git is **Version control system** is a system that records changes to a file or set file over time so that you. can restore specific version later
- Git is a **Distributed Version Control System**

Repository

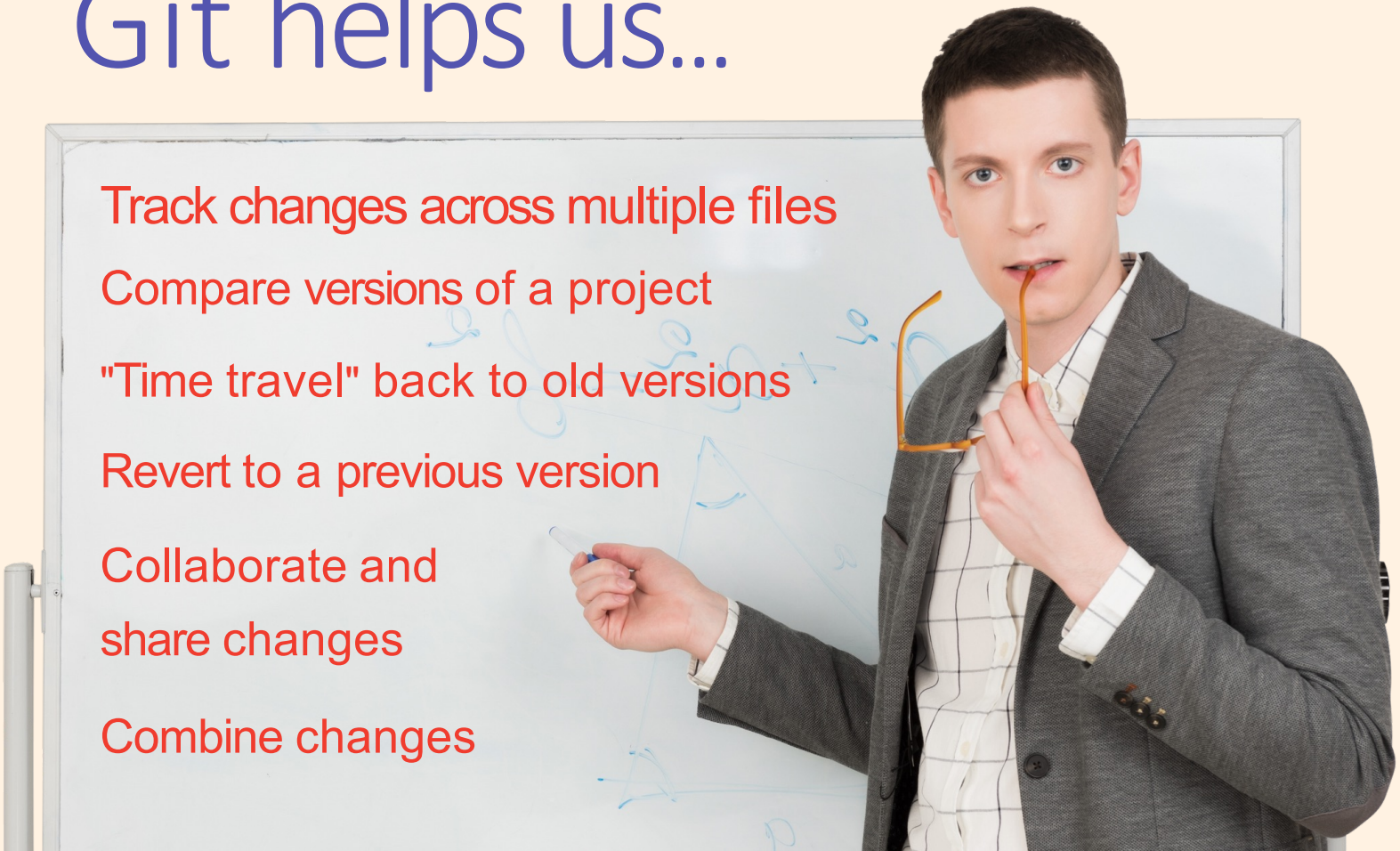
A Git "Repo" is a workspace which tracks and manages files within a folder.

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

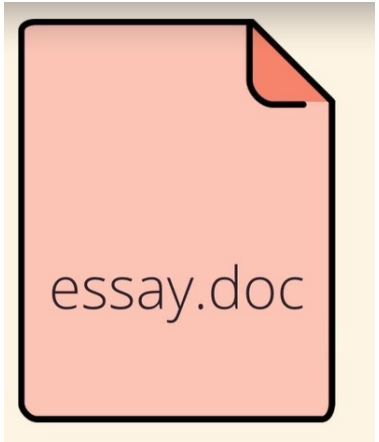
Oh boy, I sure do love playing my video games!

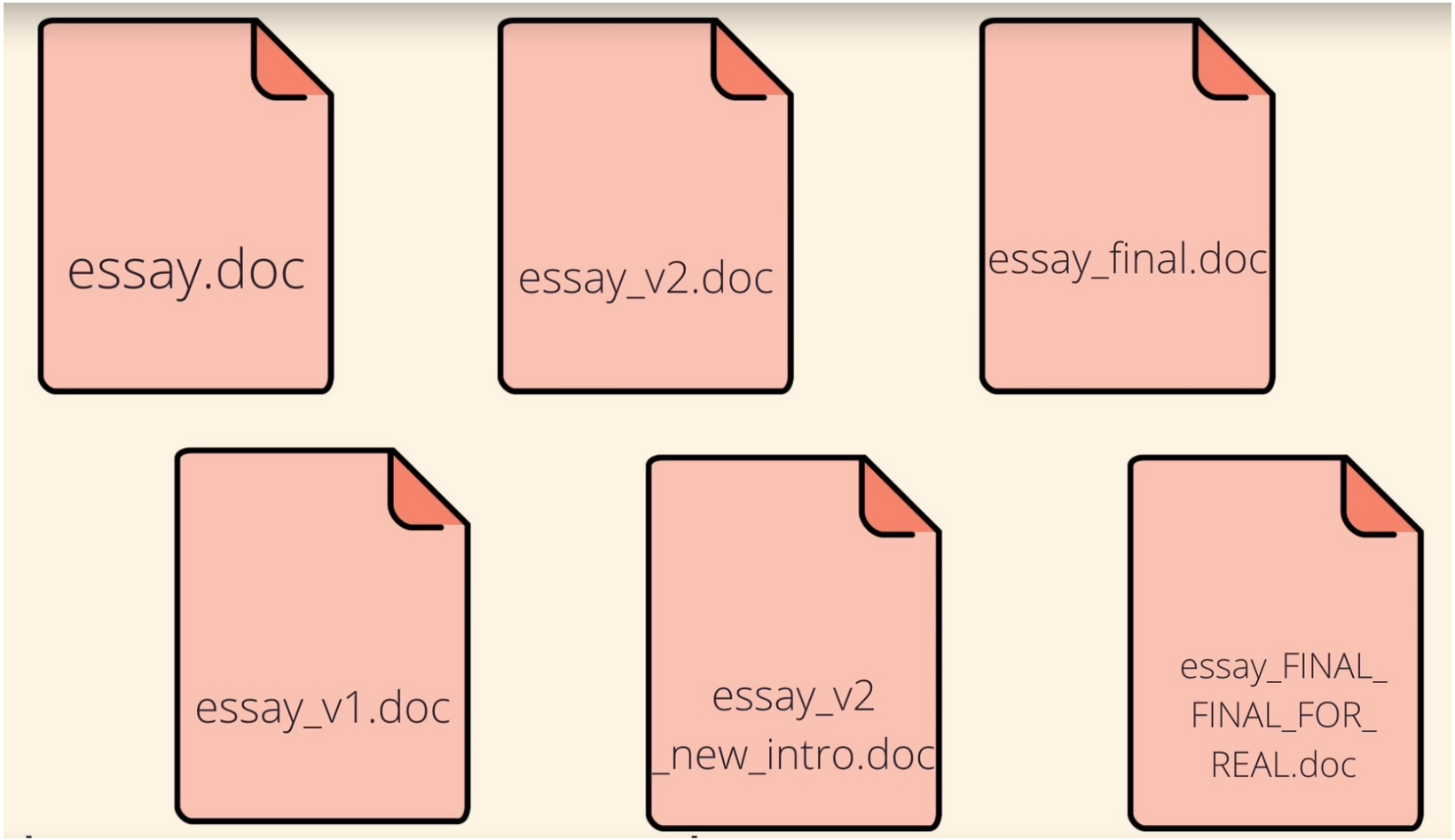
I'm going to save my game now in case I die soon!

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

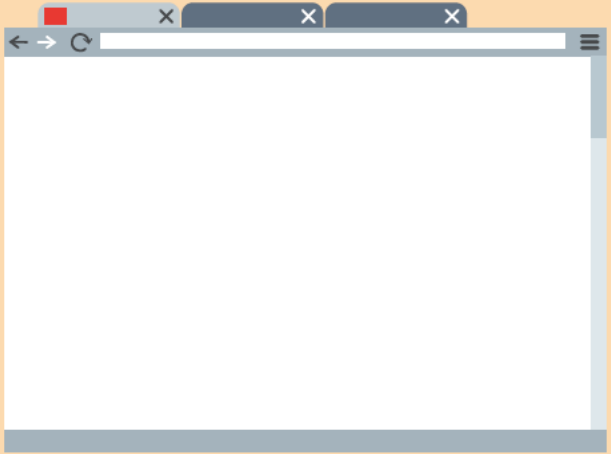
ughhhhh I died!

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


essay.doc

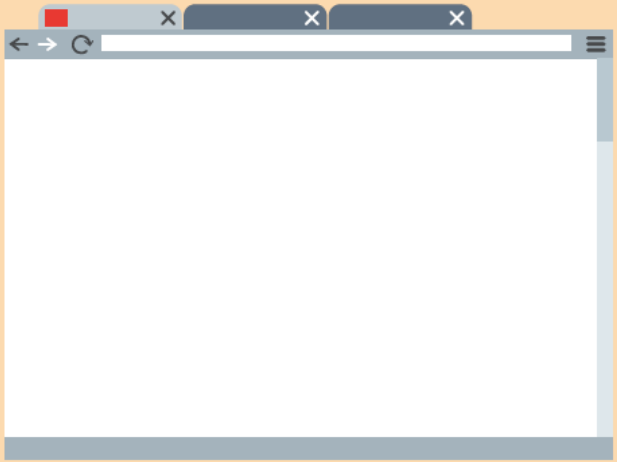

essay.doc
essay_v2.doc
essay_final.doc
essay_v1.doc
essay_v2
_new_intro.doc
essay_FINAL_
FINAL_FOR_
REAL.doc

I Start A New Project!



Add A Checkpoint

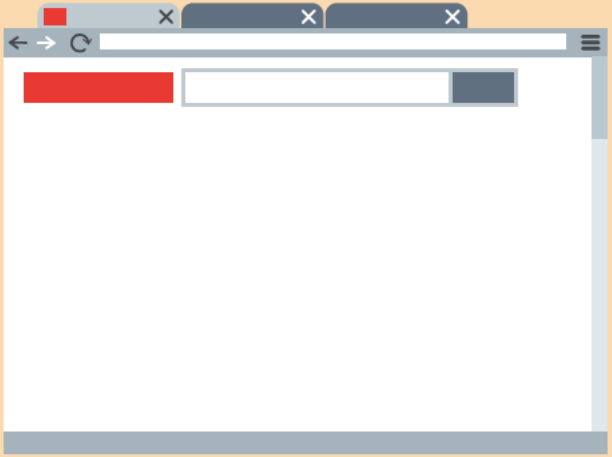
Initialize Project



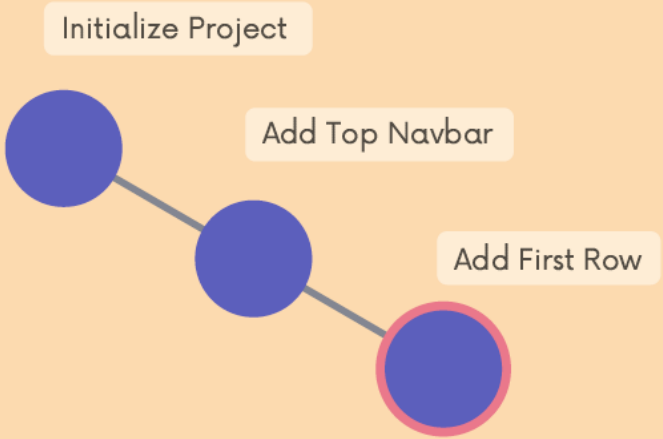
Add A Checkpoint

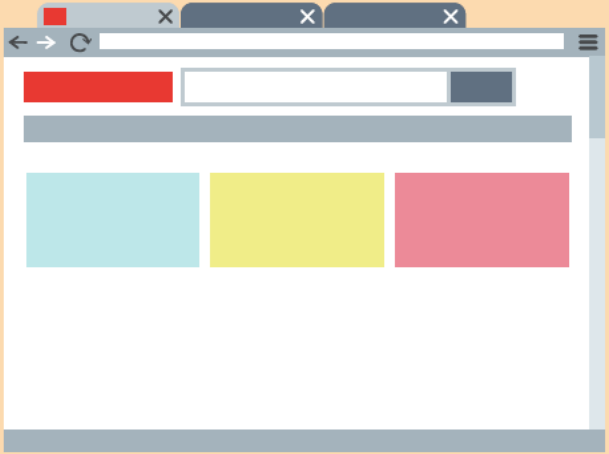
Initialize Project

Add Top Navbar

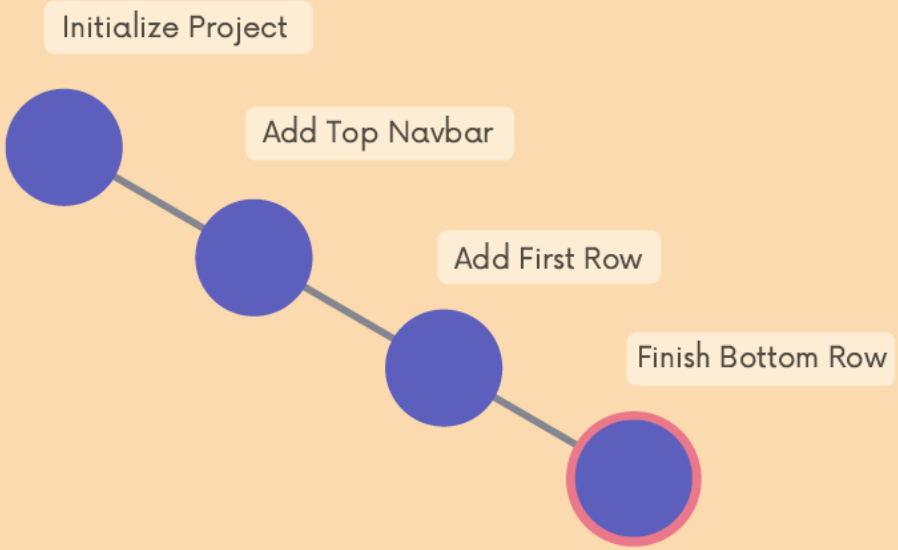


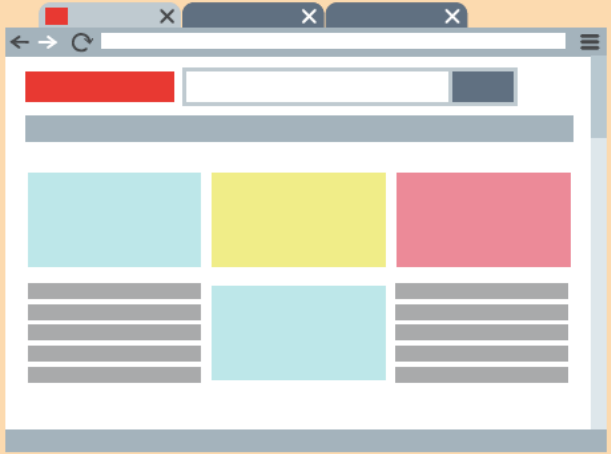
Add A Checkpoint



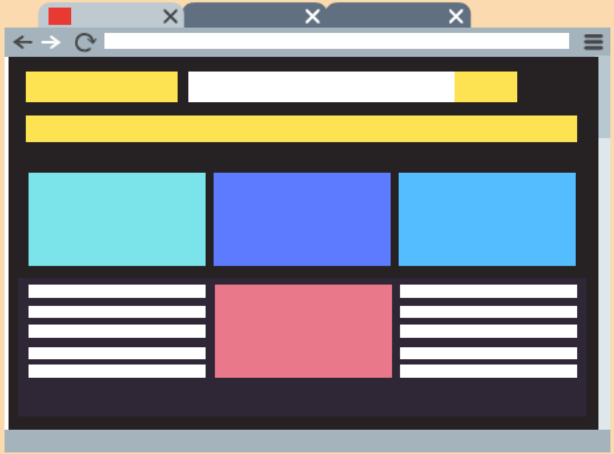


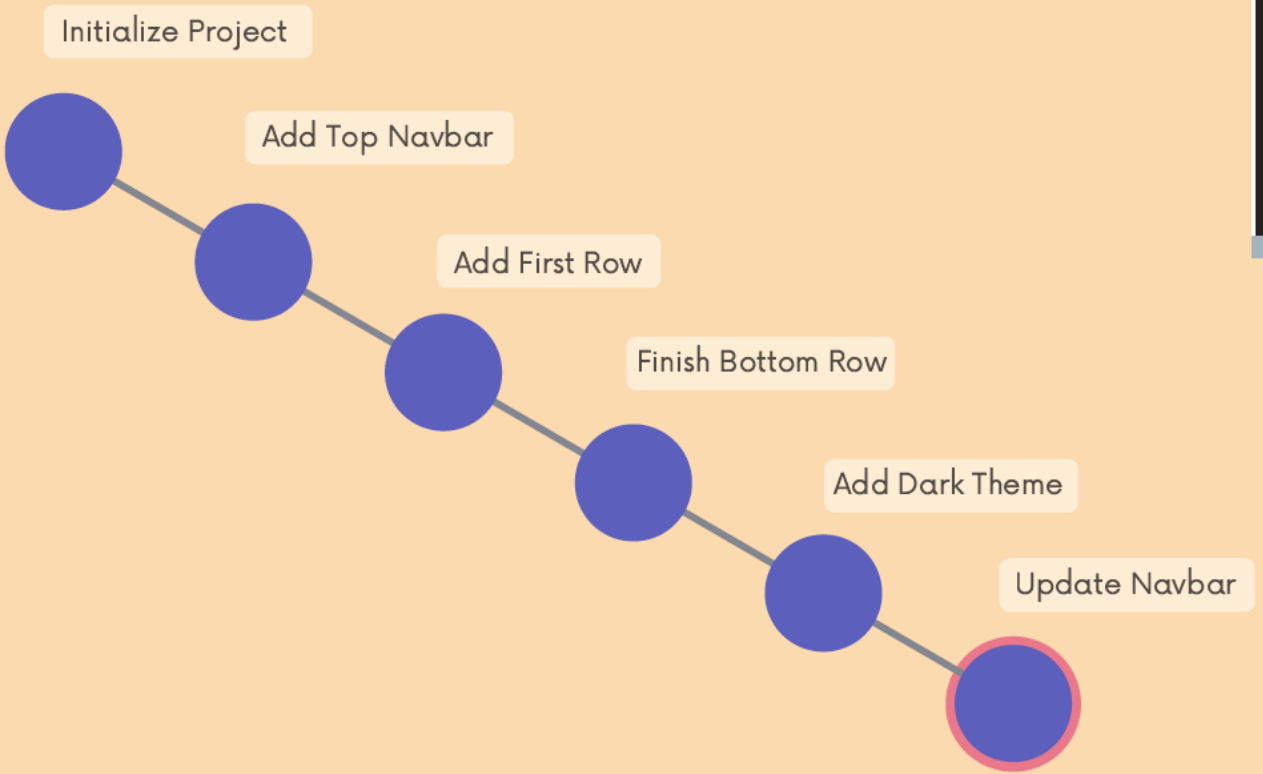
Add A Checkpoint





Add A Checkpoint

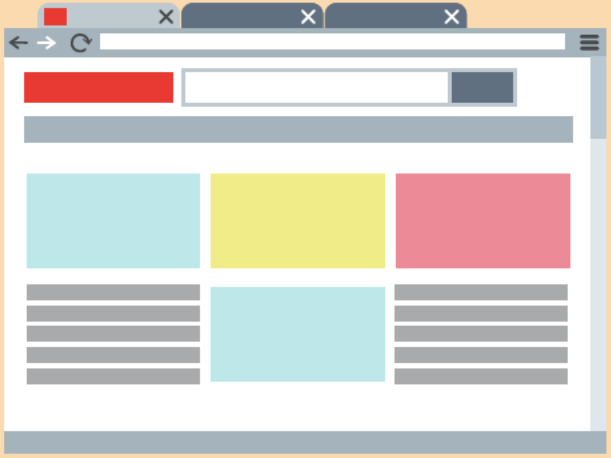


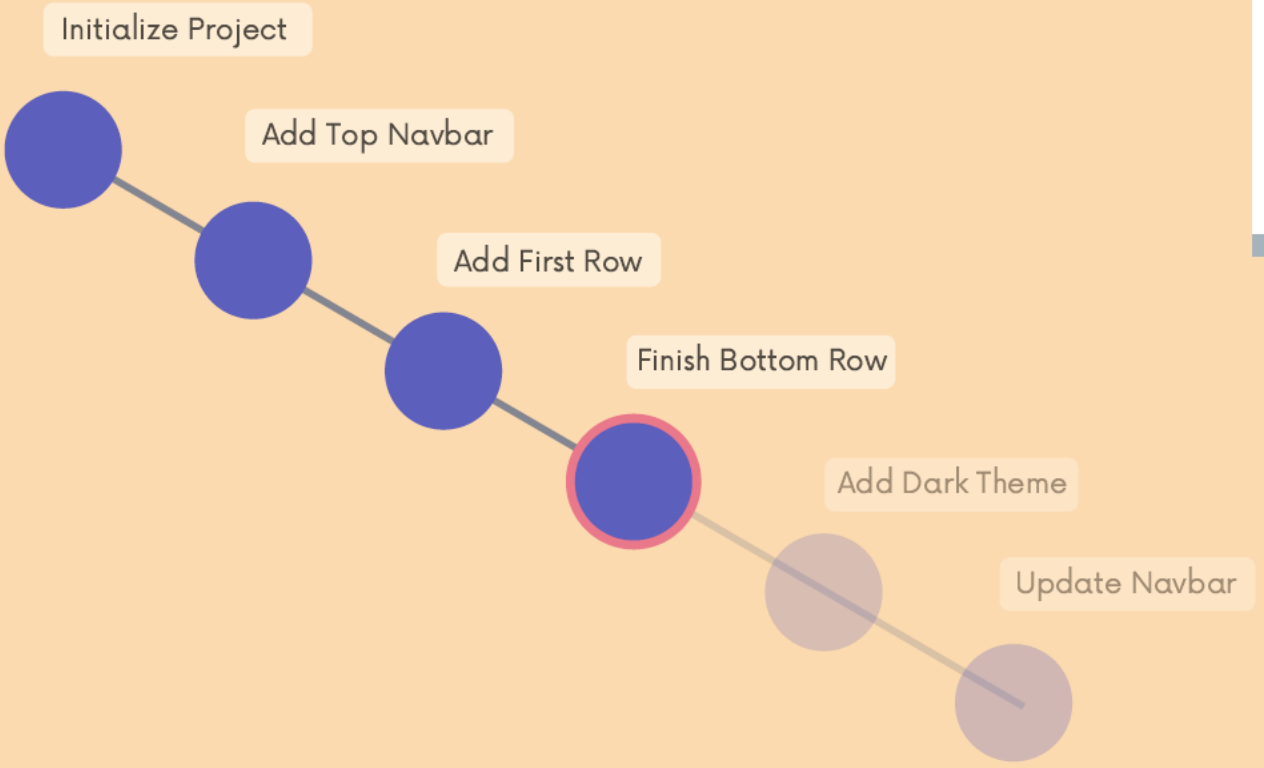


ANGRY BOSS SAYS...

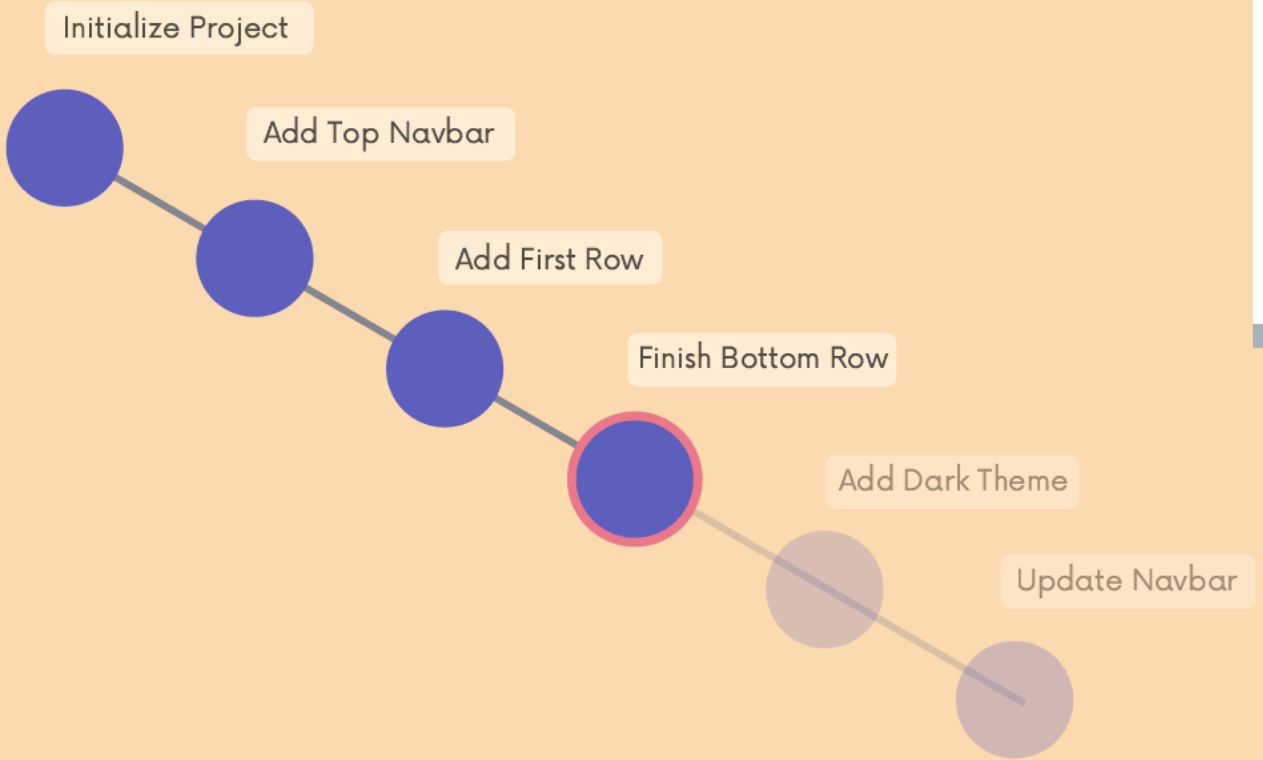
THE COLORS ARE BAD!

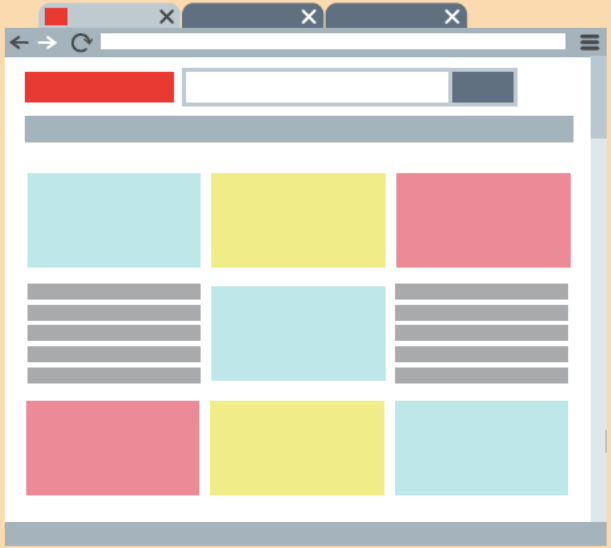
I can go back to prior checkpoints I made!





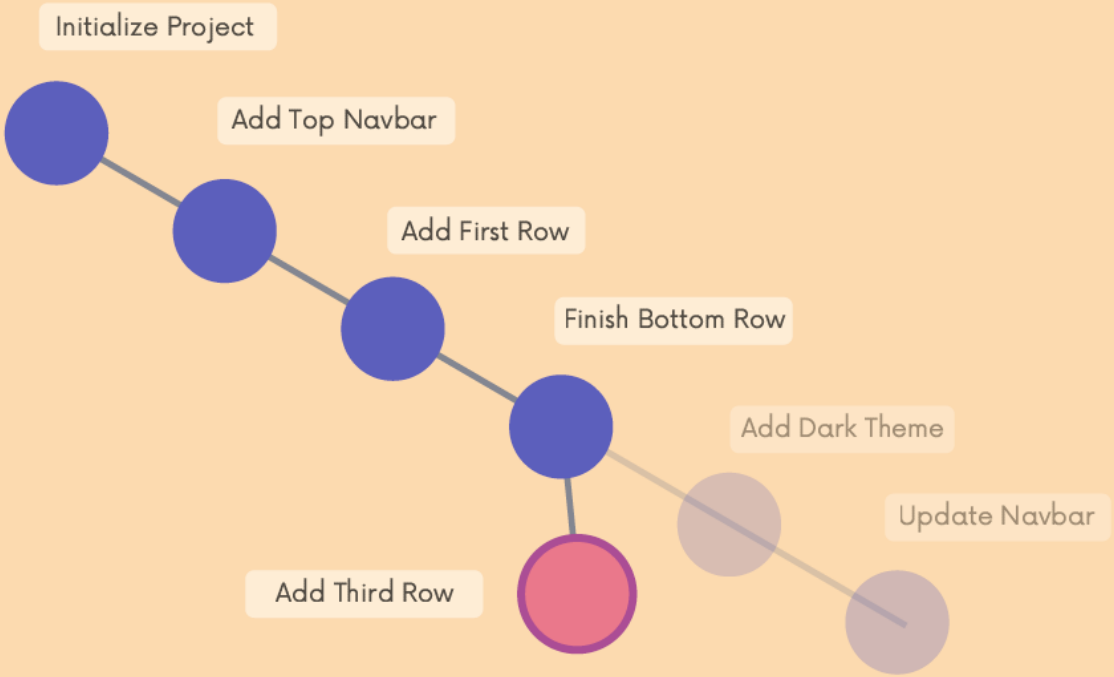
I can even start more work off of an old checkpoint

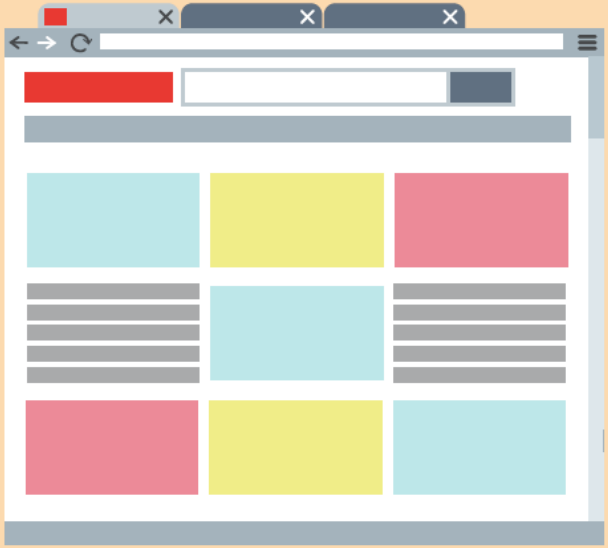




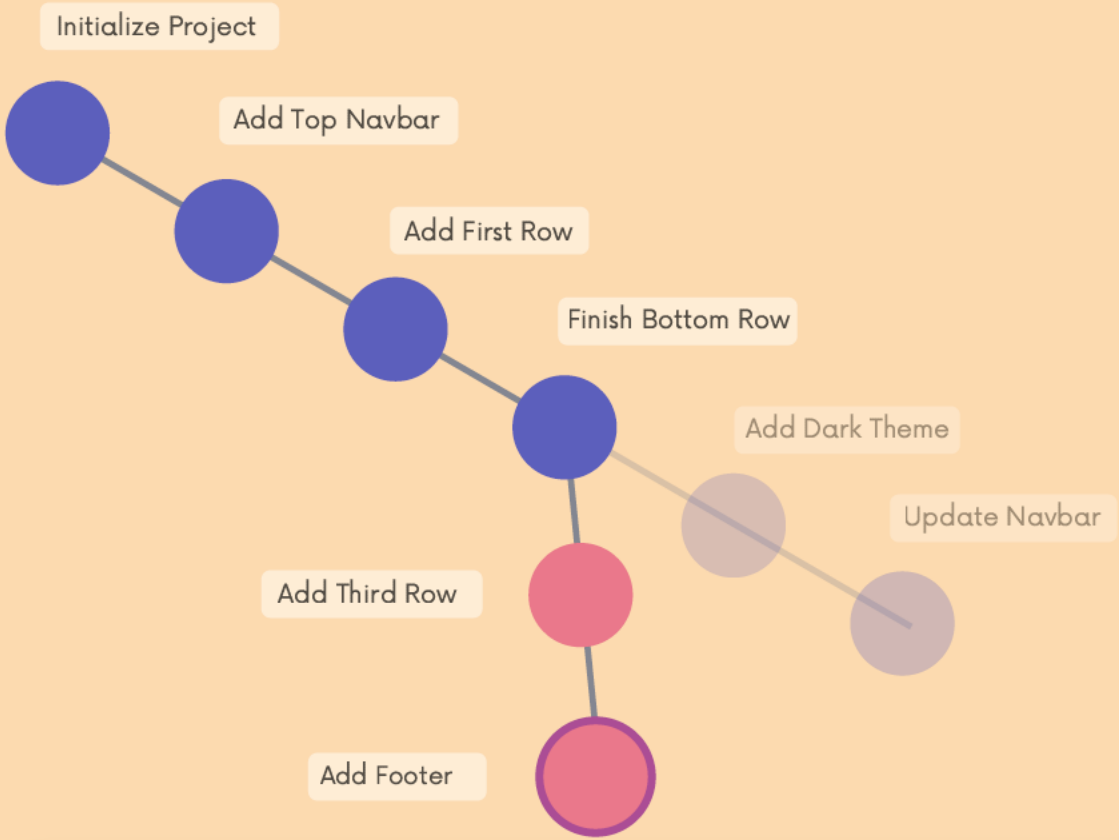
I add more content!

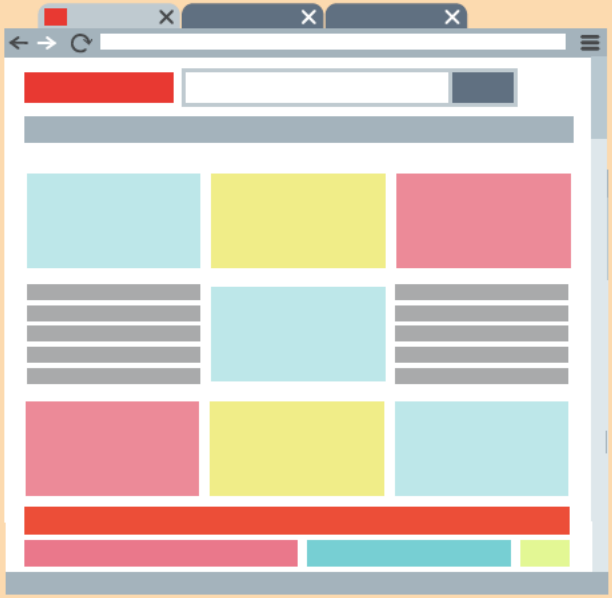
I add a new checkpoint!



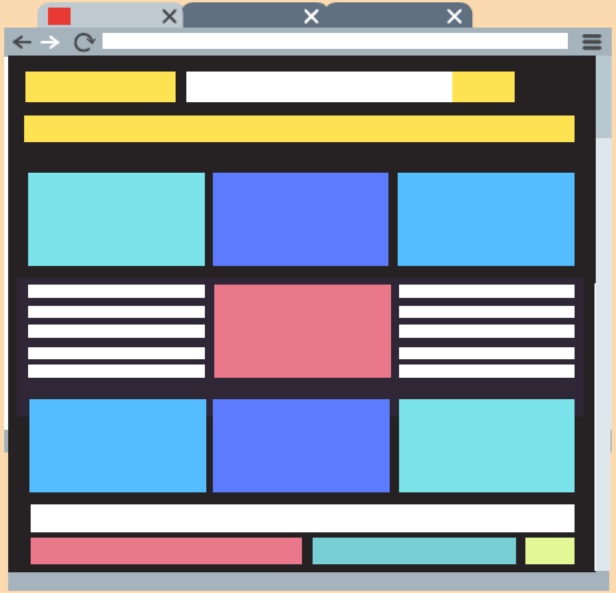


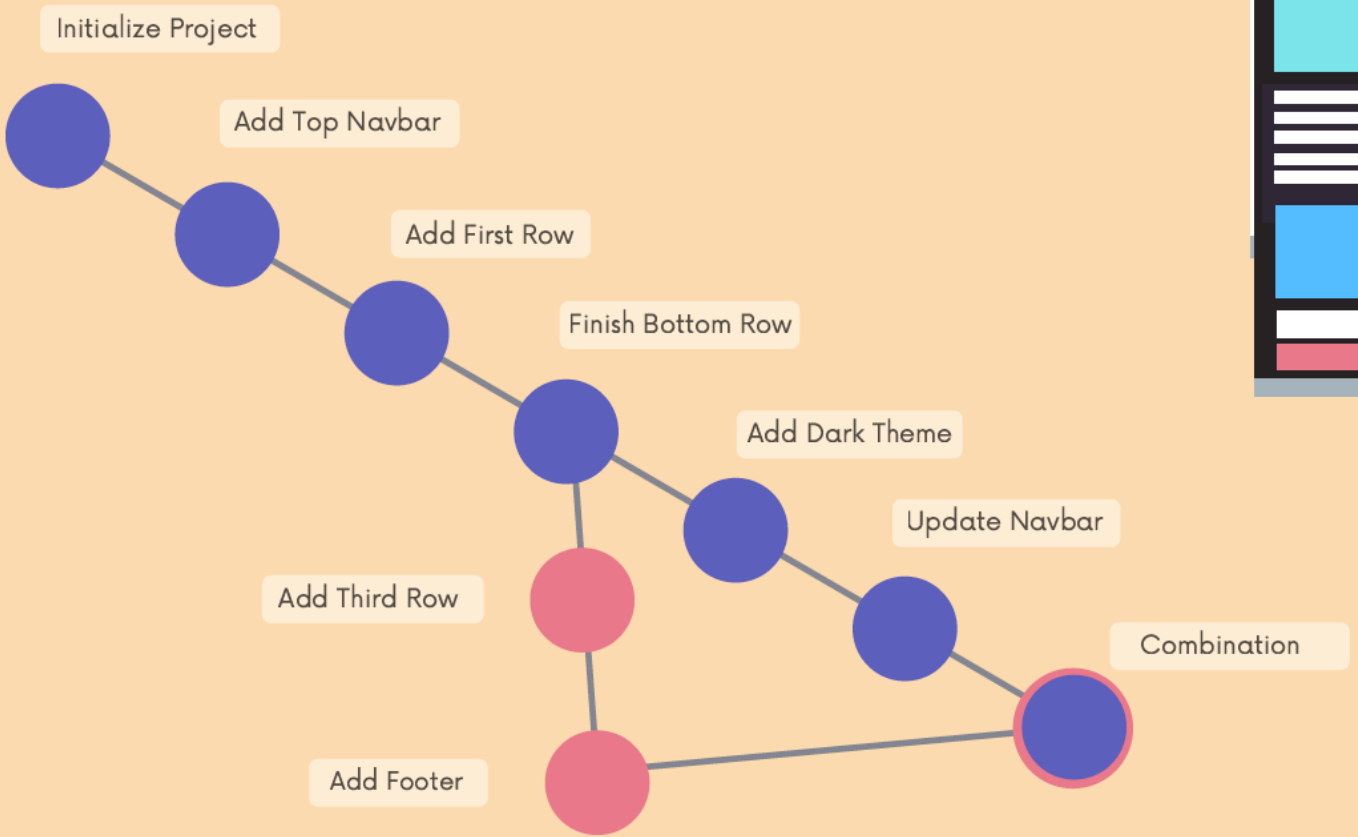
Another checkpoint!





And I can even combine checkpoints!





Before Version Control System

Before Version Control System

Before Version Control System

Version 1.0

running code version 1.0

Before Version Control System

Version 1.0

I have a great idea, send me your code

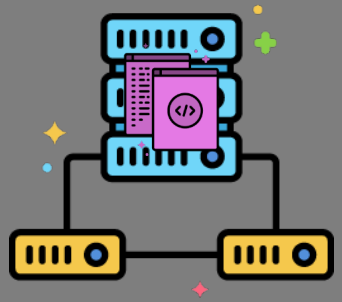
Version 2.0

running code version 1.0

Before Version Control System

Version 1.0

Version 2.0



running code version 2.0

Before Version Control System

Version 1.0

Version 2.0

running code version 2.0

Before Version Control System

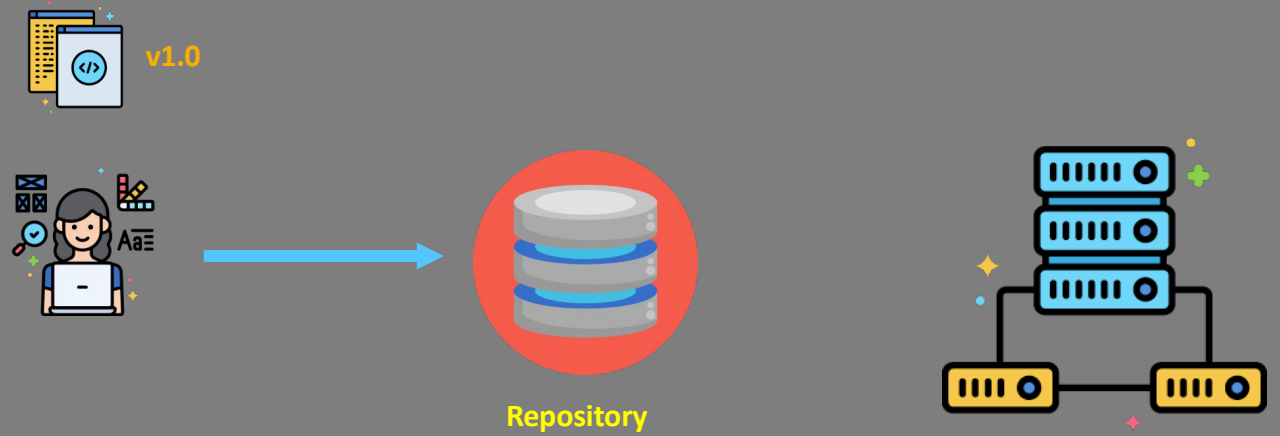
Version 1.0

Version 2.0

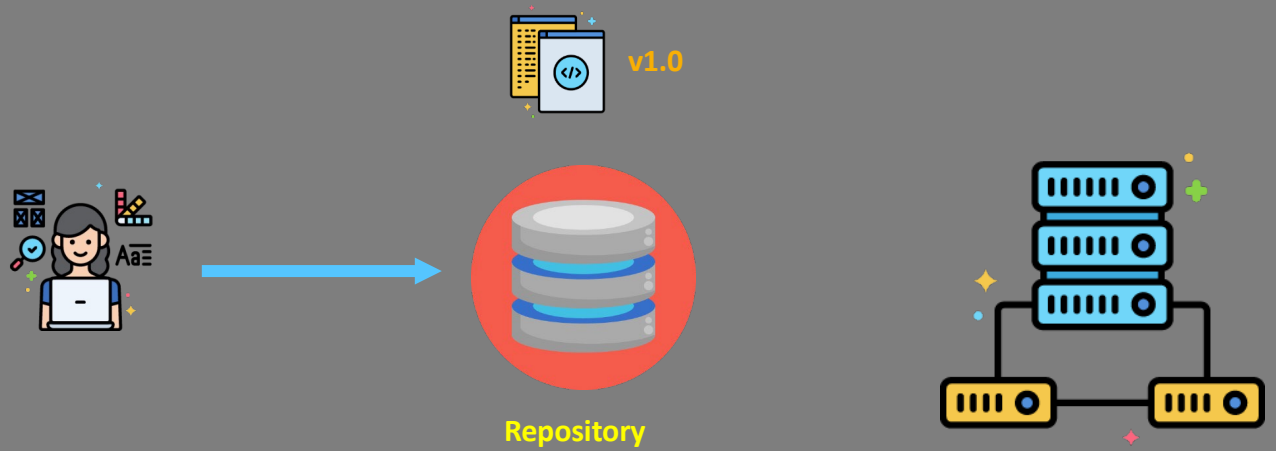
running code version 2.0

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

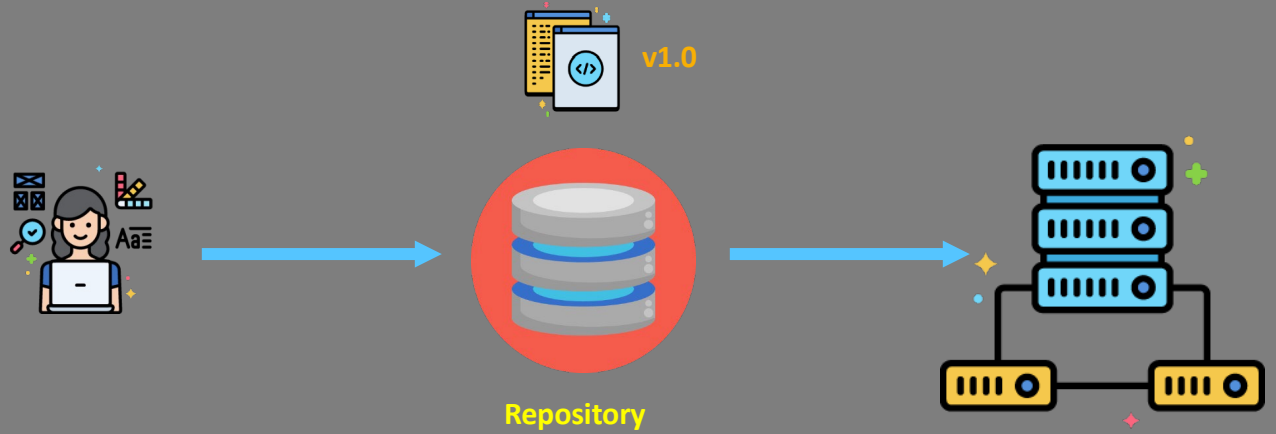
Version Control System



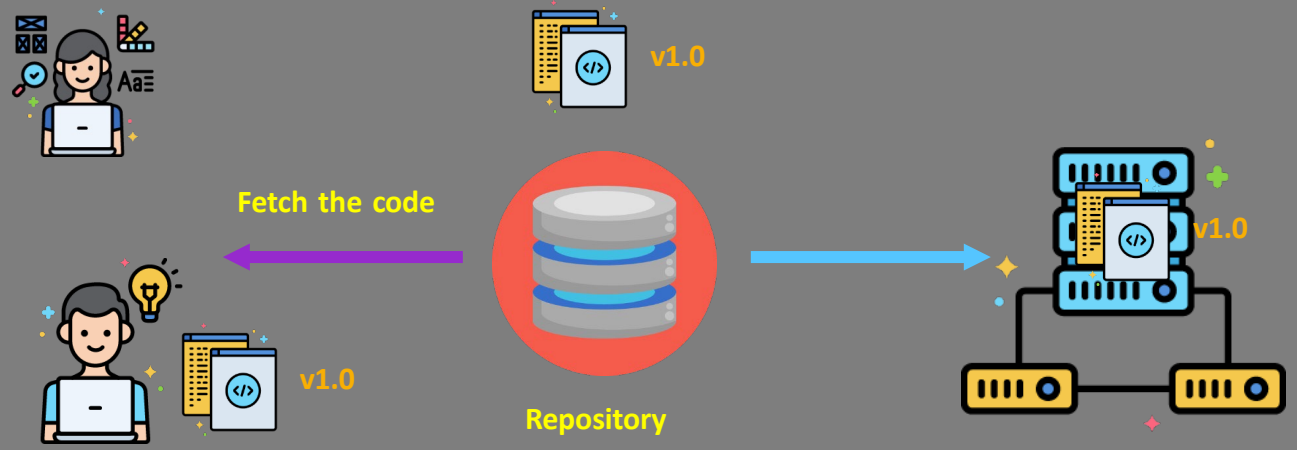
Version Control System



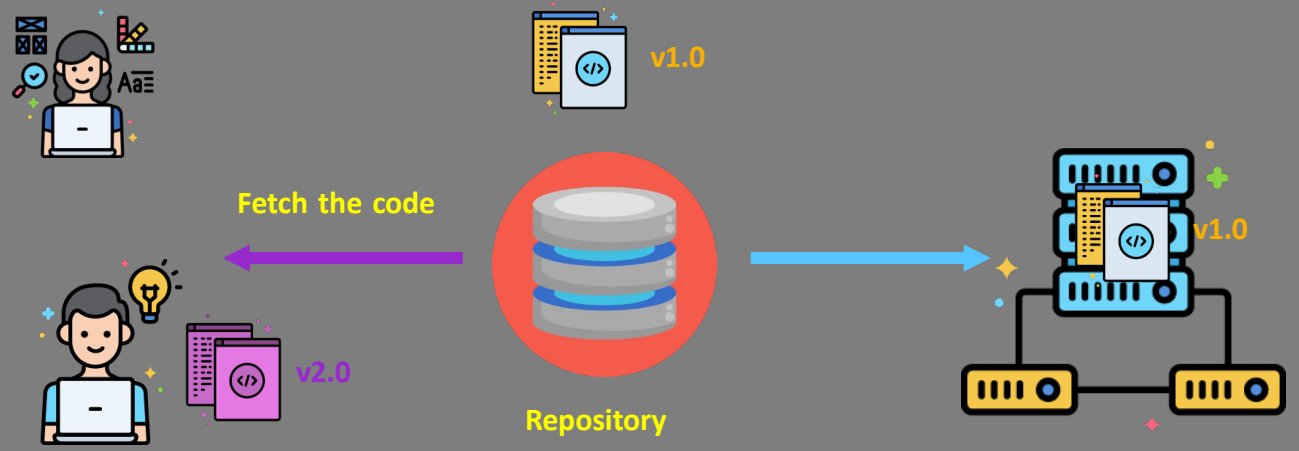
Version Control System



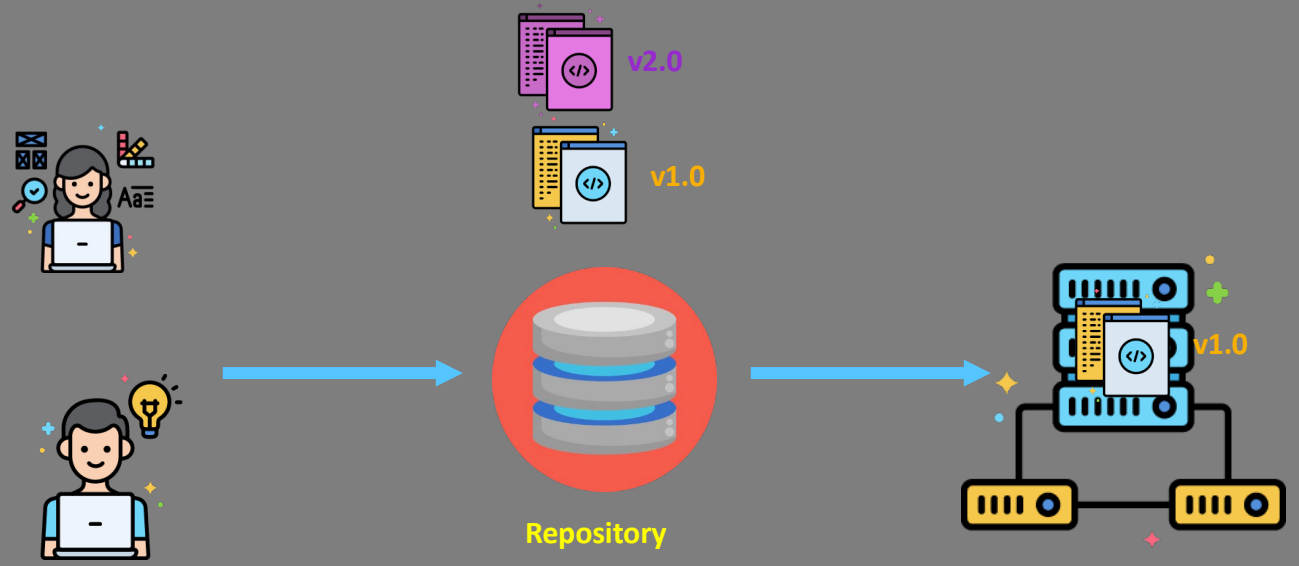
Version Control System



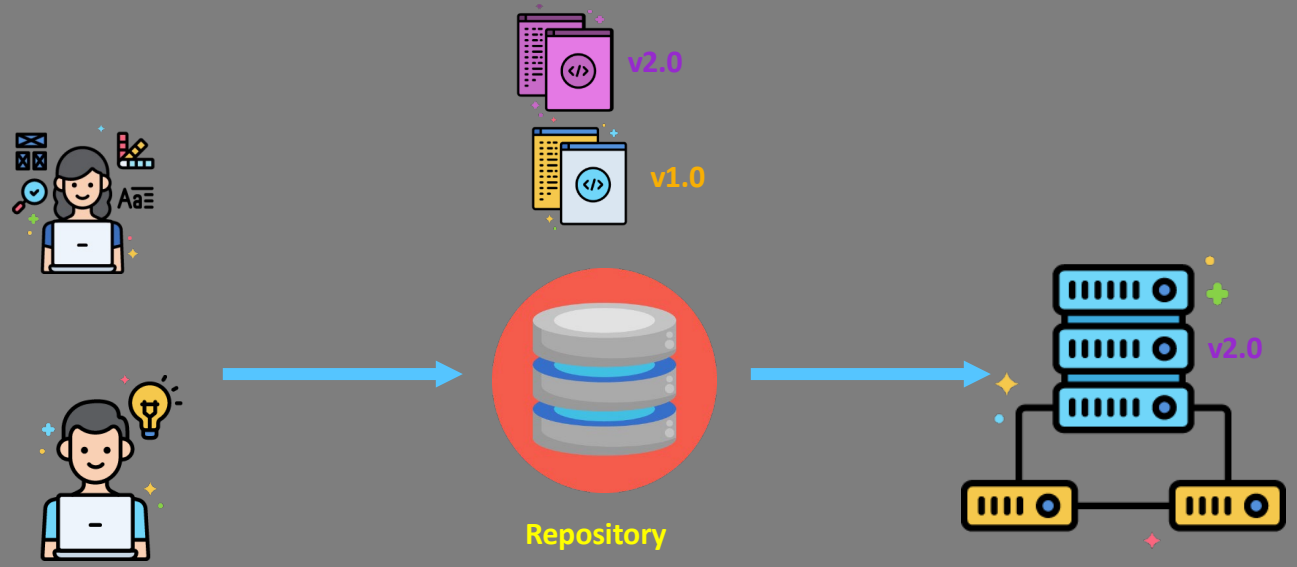
Version Control System



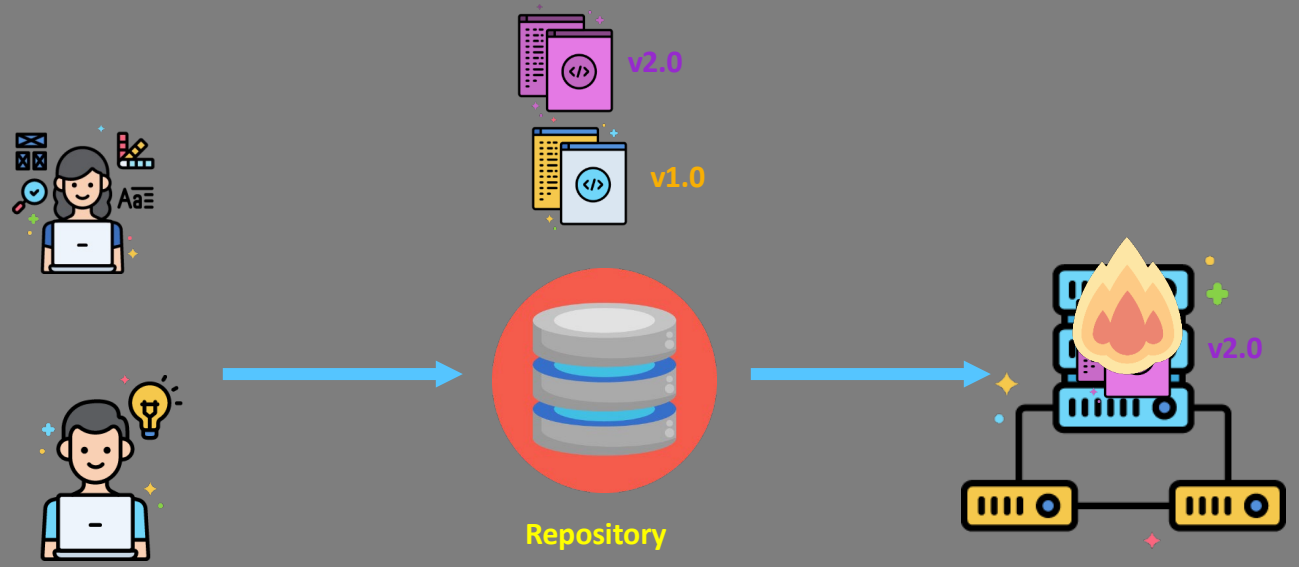
Version Control System



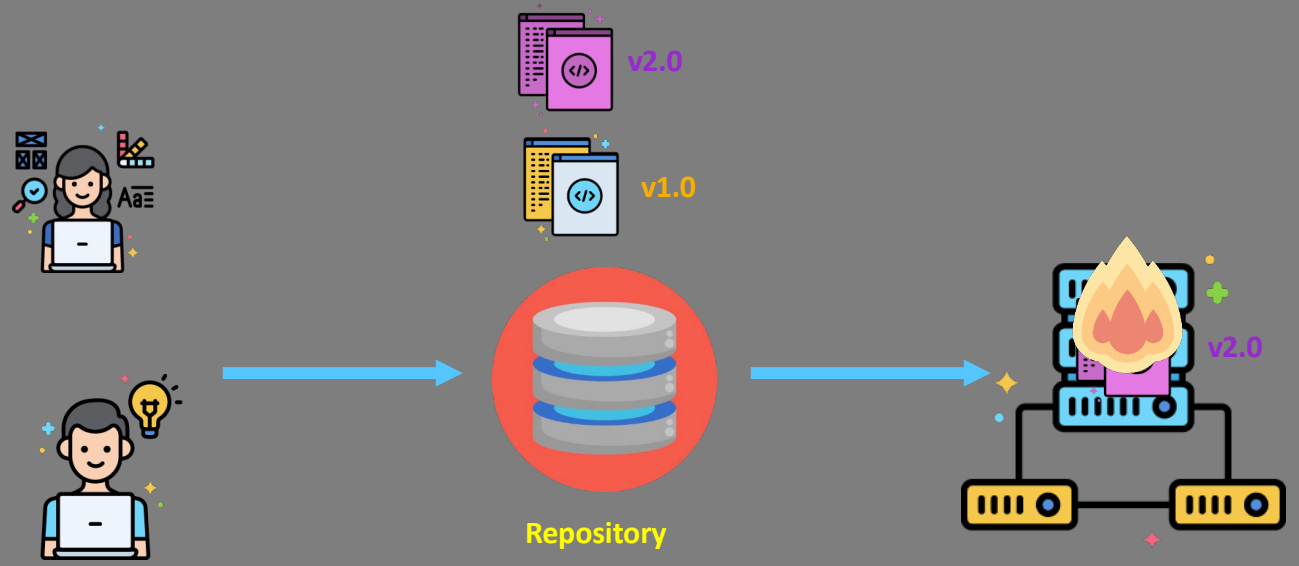
Version Control System



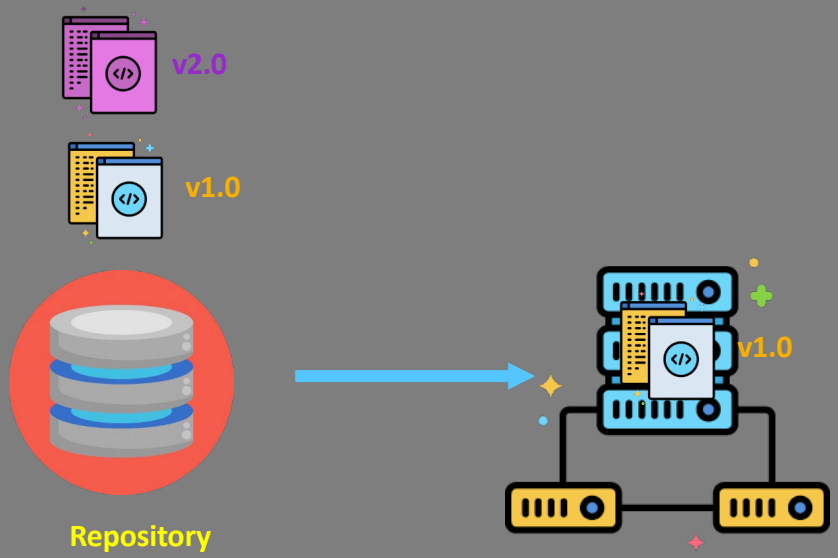
Version Control System



Version Control System



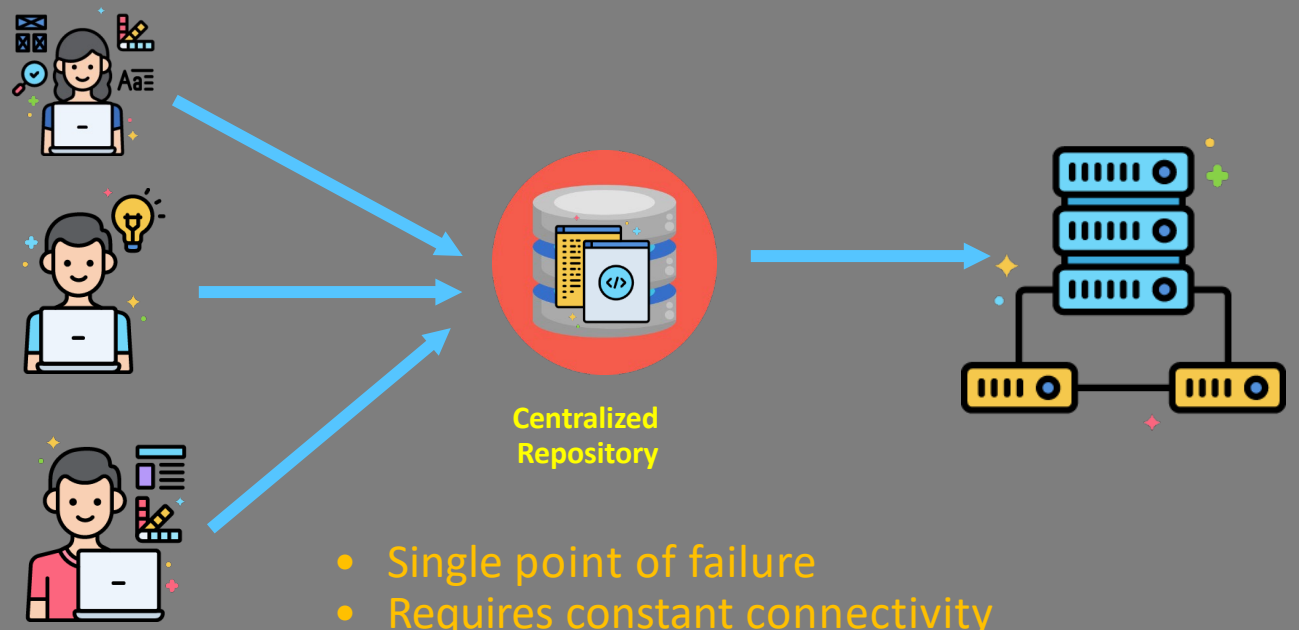
Version Control System - Git



Why Git?

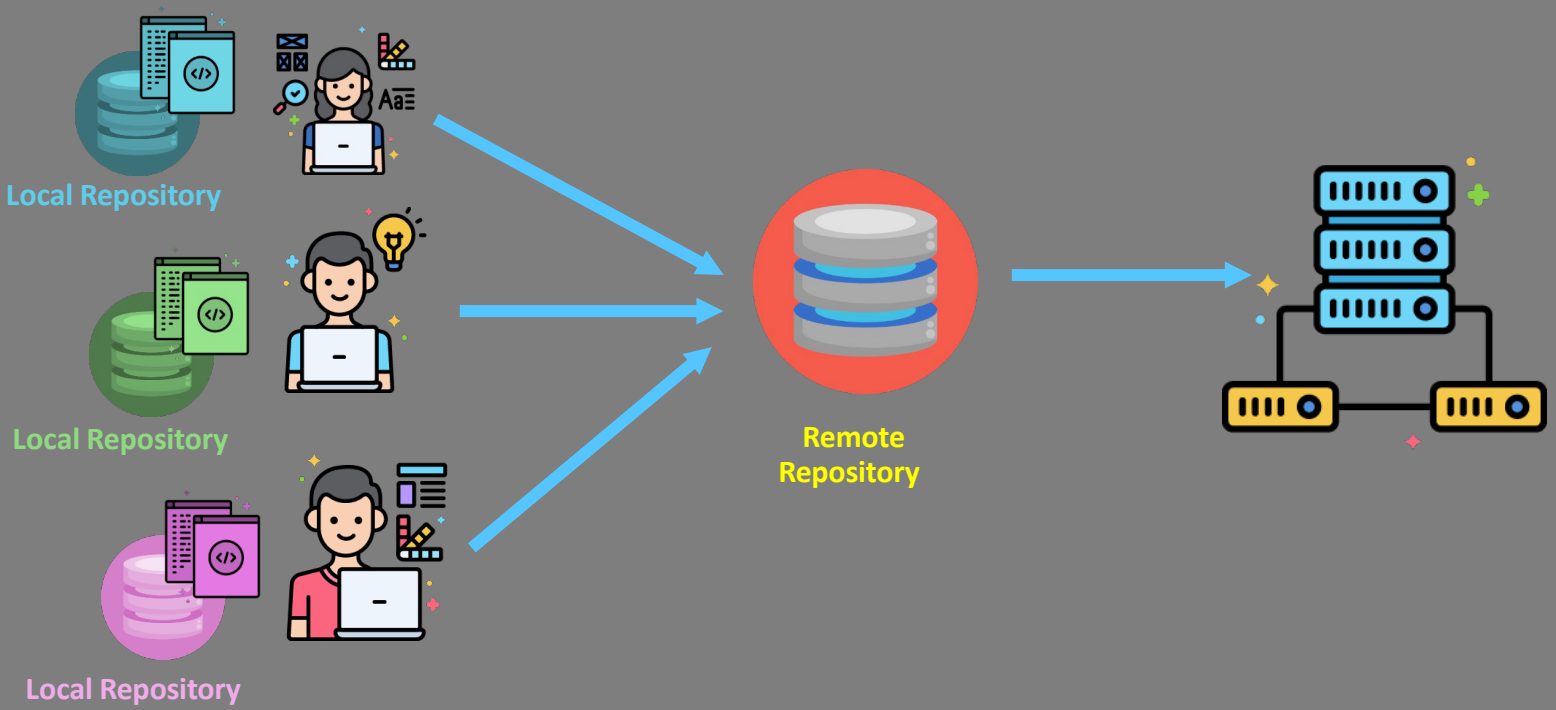
- Distributed

Centralized Version Control System

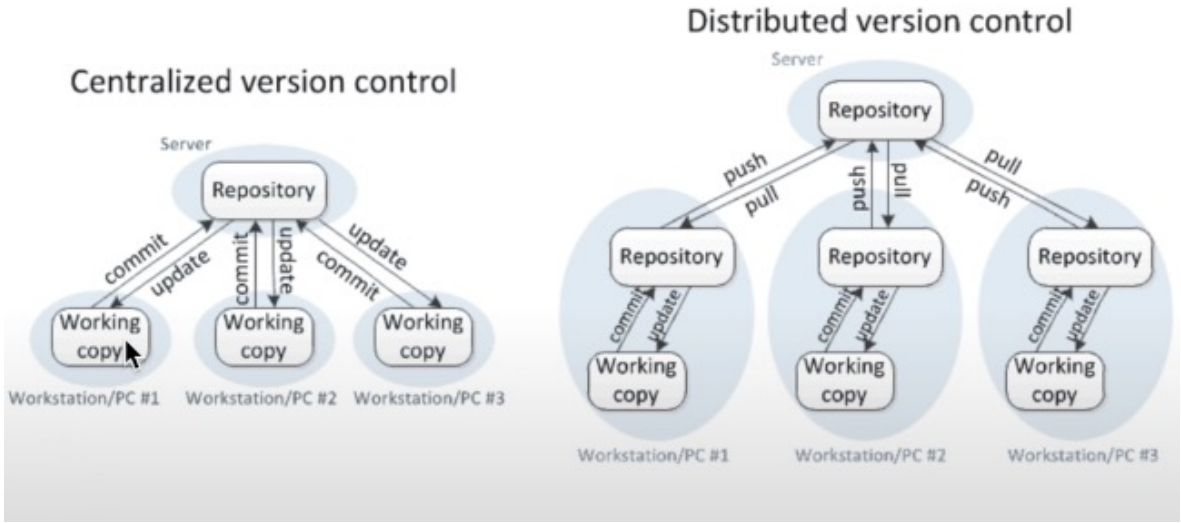


- Single point of failure
- Requires constant connectivity
- E.G – Subversion, Endevor

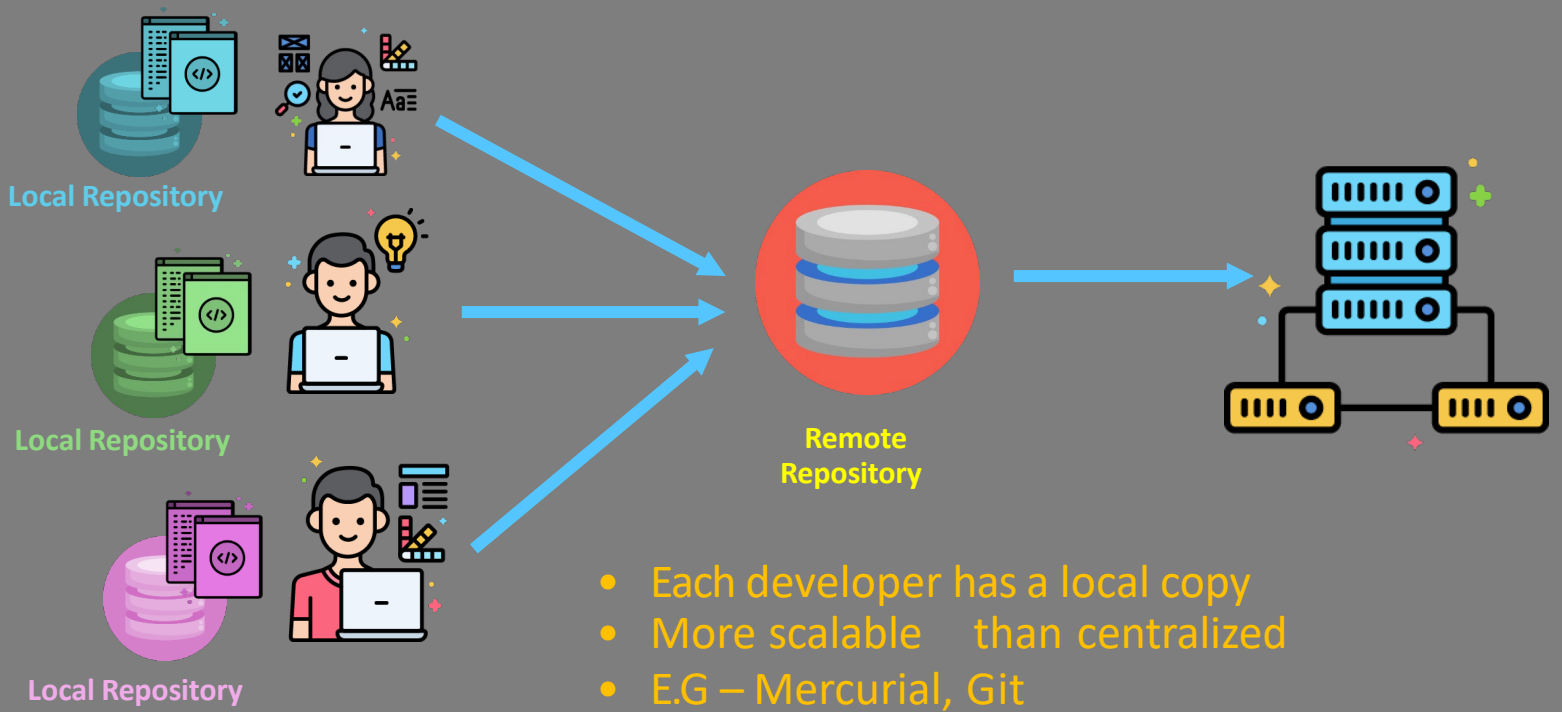
Distributed Version Control System



Centralized vs Distributed
Version Control

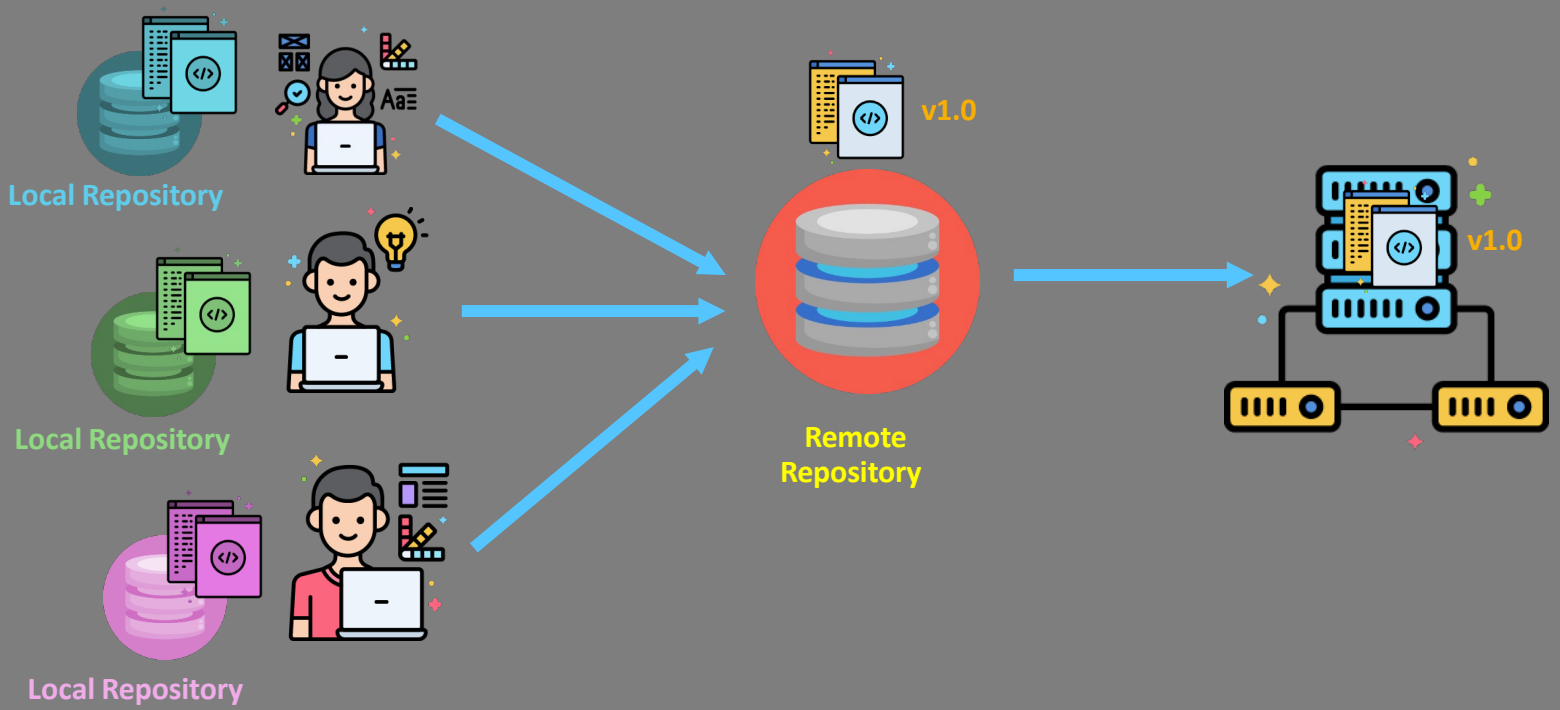


Distributed Version Control System



- Each developer has a local copy
- More scalable than centralized
- E.G – Mercurial, Git

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

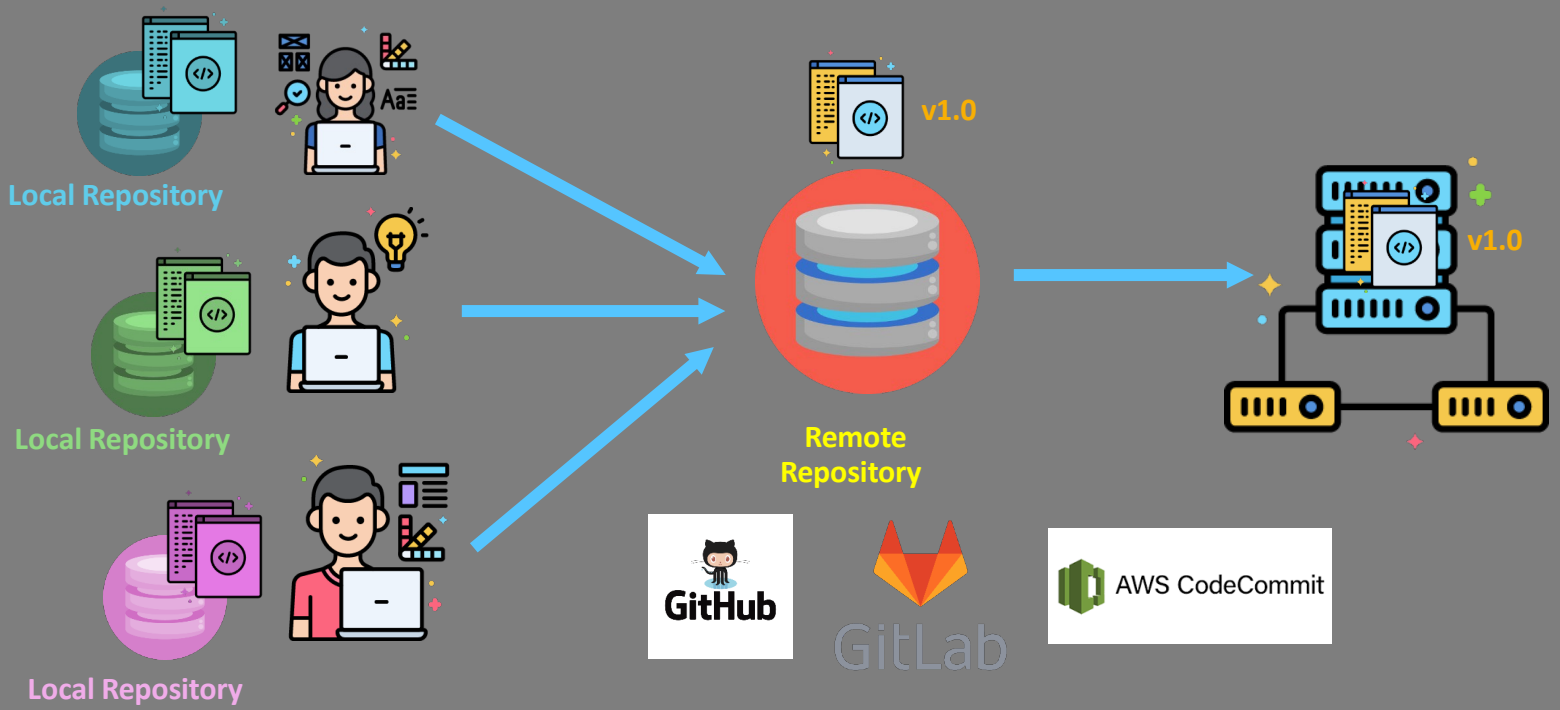
Git vs GitHub

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

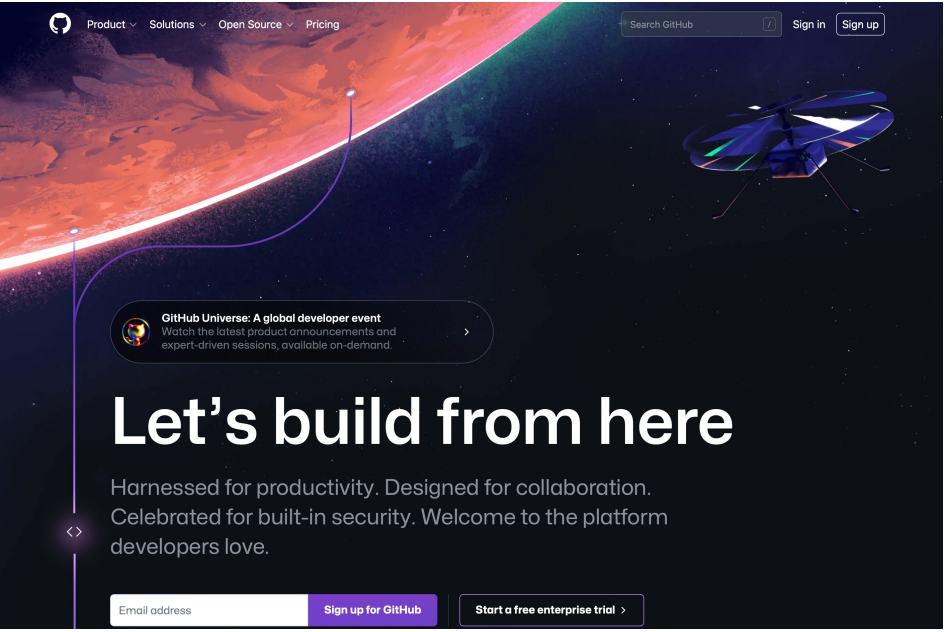
- Git repository hosting services with other features
- Runs on the cloud
- Created in 2008, currently owned by Microsoft
- Not open source, have free and paid tiers

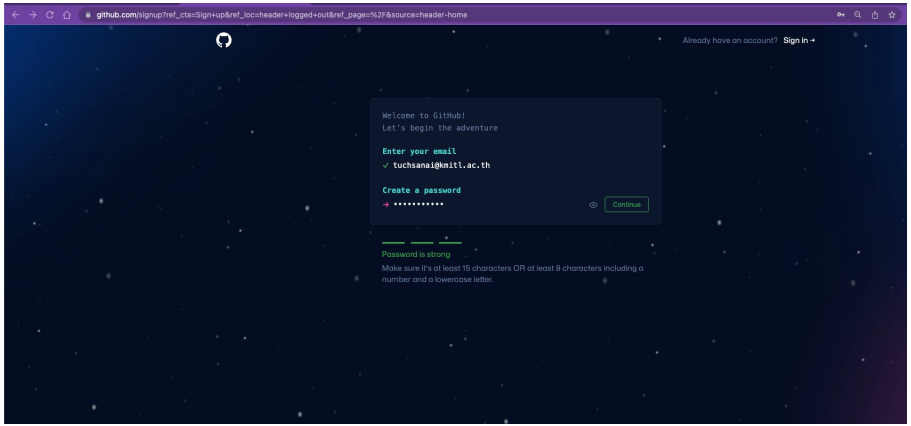
Distributed Version Control System



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

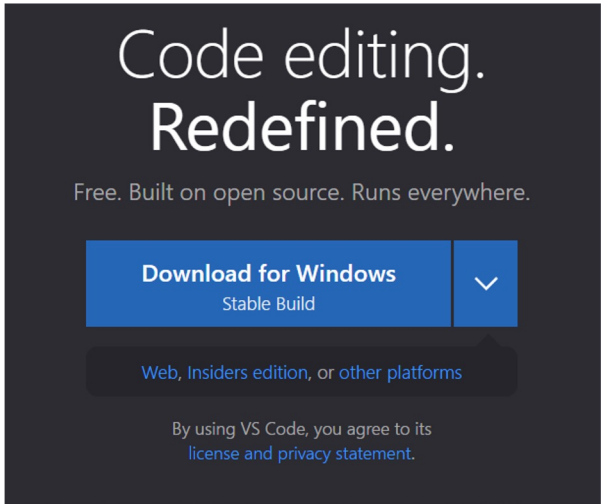
- **MacOS or Linux Users:**
  - Our suggested text editor for this course is VS Code:
    - **https://code.visualstudio.com/**
  - Its created by Microsoft and has direct integrations with GitHub and is one of the most popular text editors today.
  - You can follow along with any text editor you prefer 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.

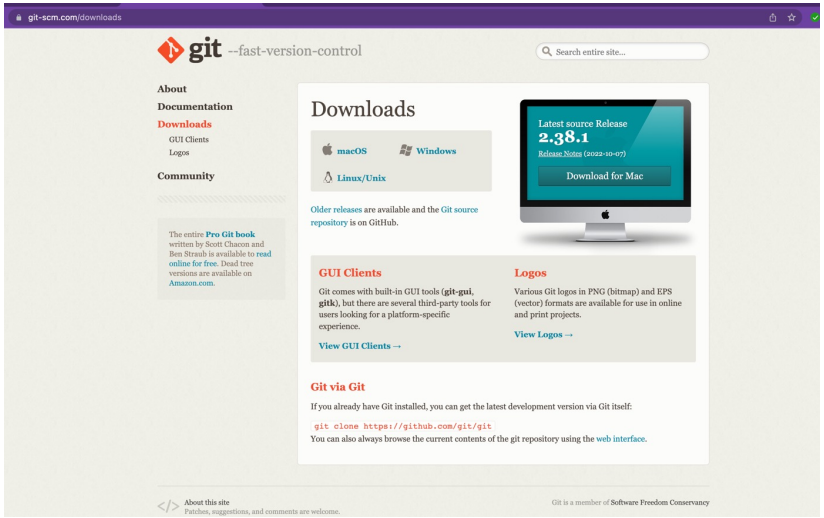
Week 1 - Starting with Git

- **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/**



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

Week 1 - Starting with Git

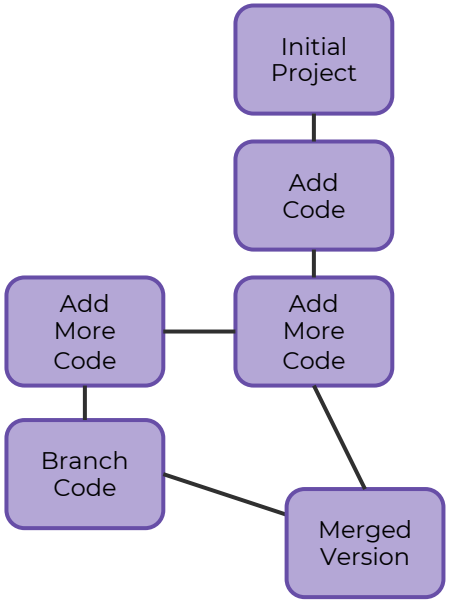
DAY 1
Creating a Git Repository

Day 1 - Starting with Git

- The main place we track changes and manage our files that are using Git is called a **repository**.

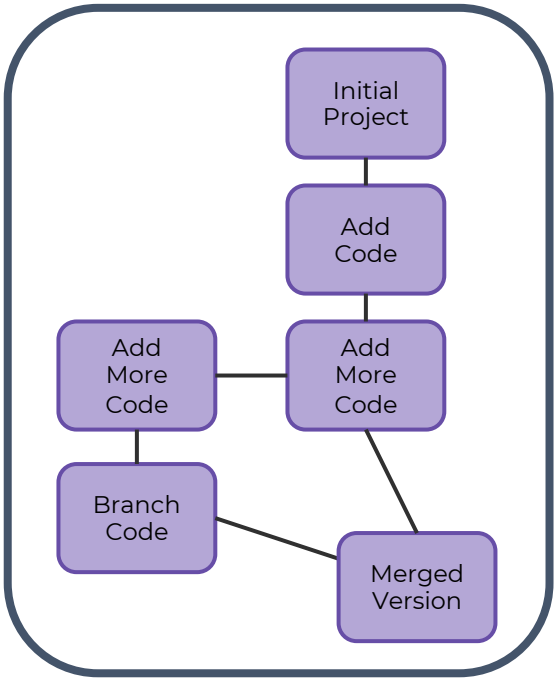
Day 1 - Starting with Git

- The main place we track changes and manage our files that are using Git is called a **repository**.



Day 1 - Starting with Git

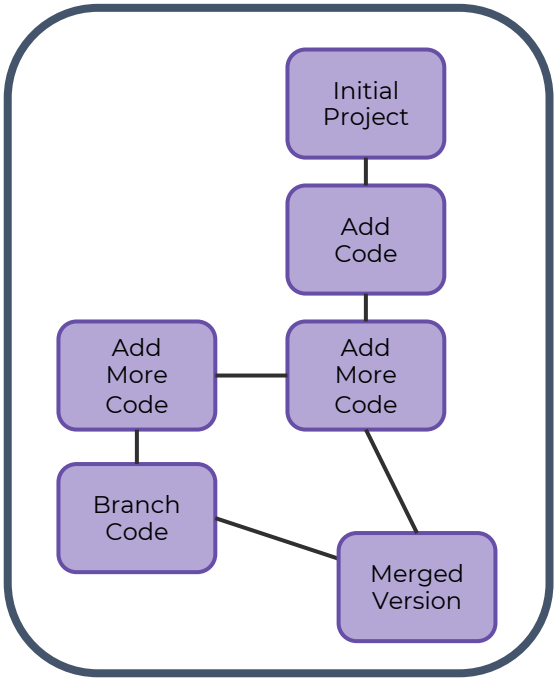
- The main place we track changes and manage our files that are using Git is called a **repository**.



Repository
AKA “Repo”

Day 1 - Starting with Git

- Let's explore a public repository:
  - **https://github.com/tensorflow/tensorflow**



Repository AKA "Repo"

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

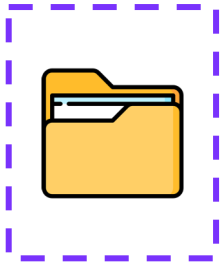
- How can we create a Git Repository?
  - Upon creating a repository with **git init** you will create a hidden .git file.
  - The .git file is a hidden file that manages the versioning of the files inside the Git repository.

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

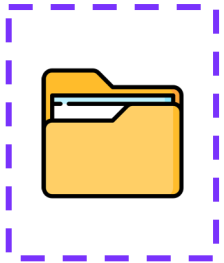
- Git inside a Folder/Directory:
  - Upon creating a Git Repository, all the folders/directories inside the top level Git Repository will also be part of that Repository, meaning all the changes are tracked.



`git init`

Day 1 - Starting with Git

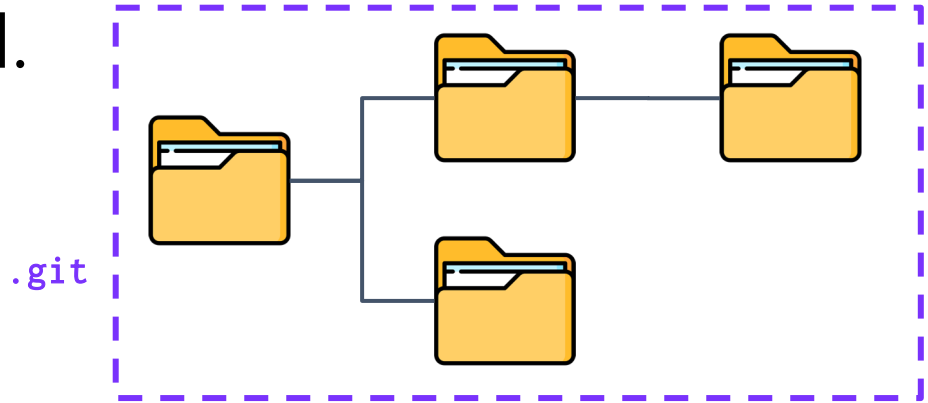
- Git inside a Folder/Directory:
  - Upon creating a Git Repository, all the folders/directories inside the top level Git Repository will also be part of that Repository, meaning all the changes are tracked.



`git init`

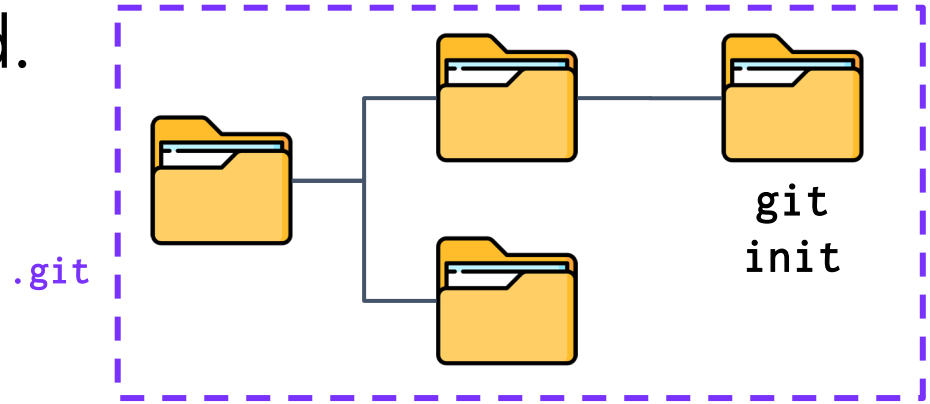
Day 1 - Starting with Git

- 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.



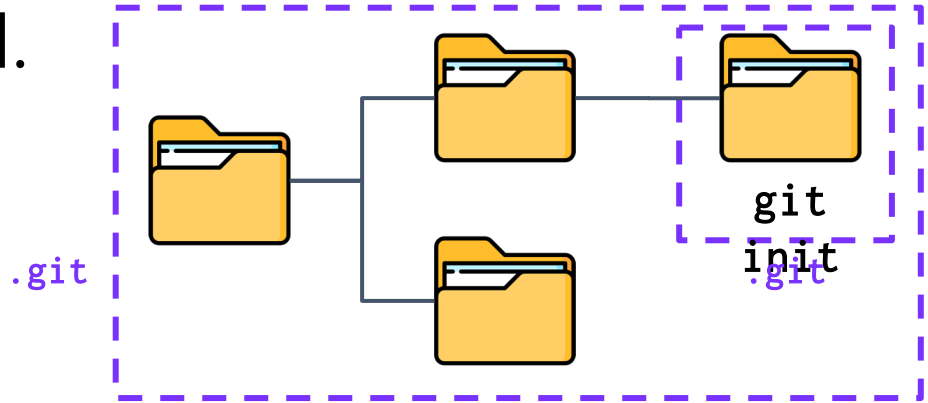
Day 1 - Starting with Git

- 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.



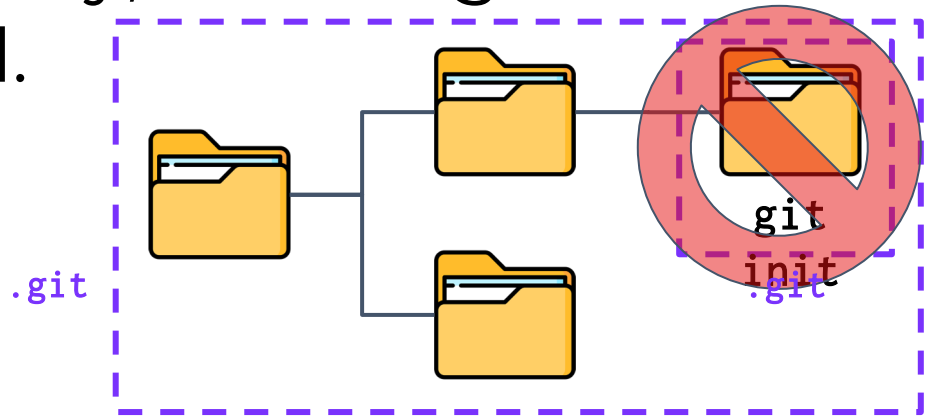
Day 1 - Starting with Git

- 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.



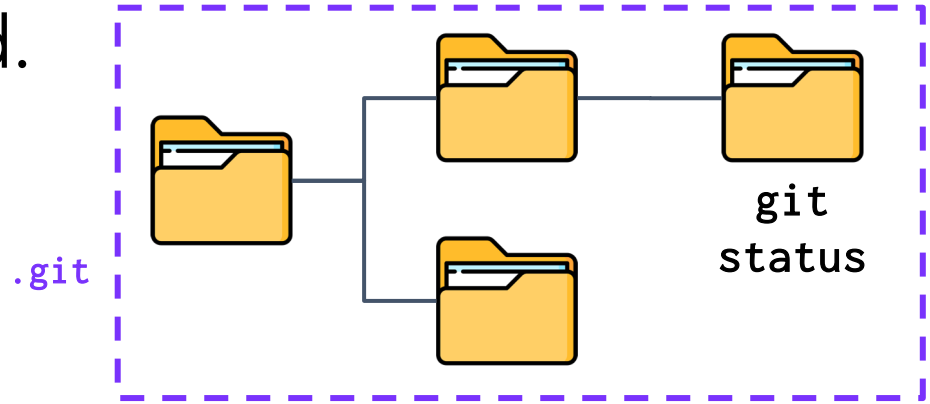
Day 1 - Starting with Git

- 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

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**.
  - 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

- We discovered we can easily clone other public repositories with the **git clone** command and then the HTTPS URL for the public repository.
- Now let’s explore how to deal with private repositories we wish to clone.

Day 1 - Starting with Git

- Option 1: Command Line:
  - Create Personal Access Tokens (PAT) on Github.com
  - When using the **git clone** command, reference the PAT.
- Option 2: GitHub Desktop Tool GUI:
  - Open the Github Desktop Tool
  - Login with GitHub Username and PW
  - Clone Repo via GUI

Day 1 - Starting with Git

- Clone Syntax with PAT:

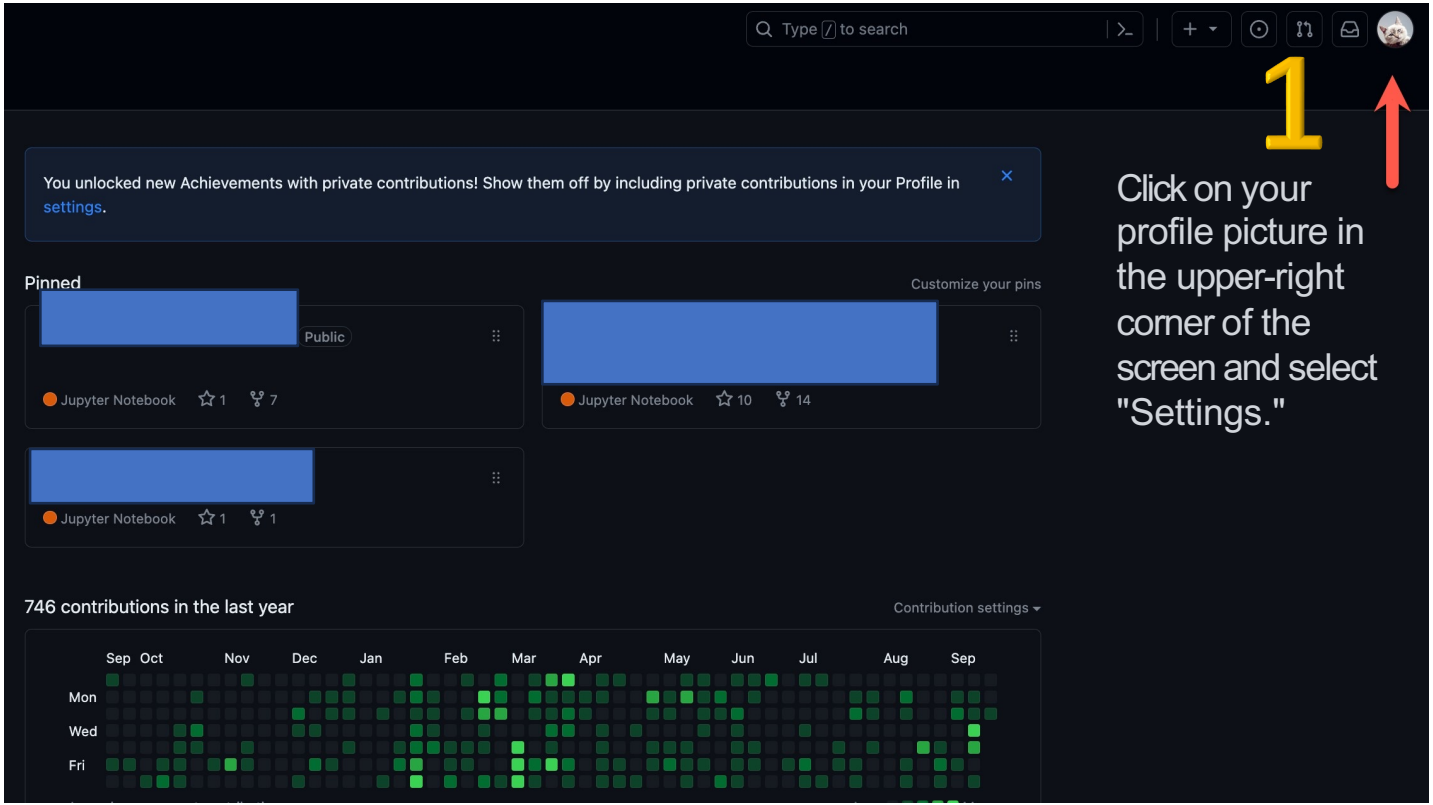
git clone https://token@github.com/account/repo.git

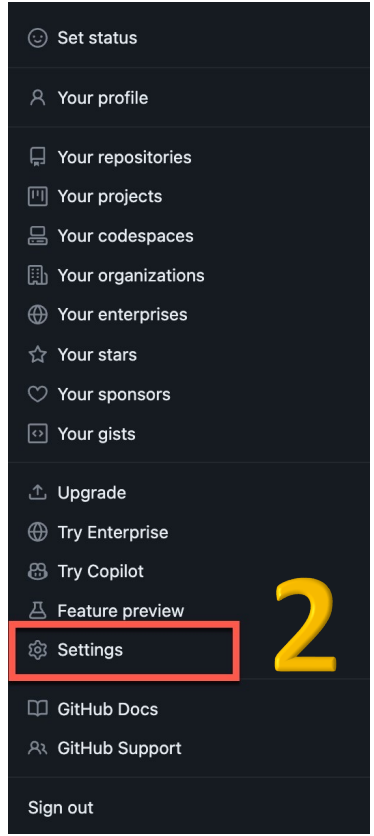
- Previously we used:

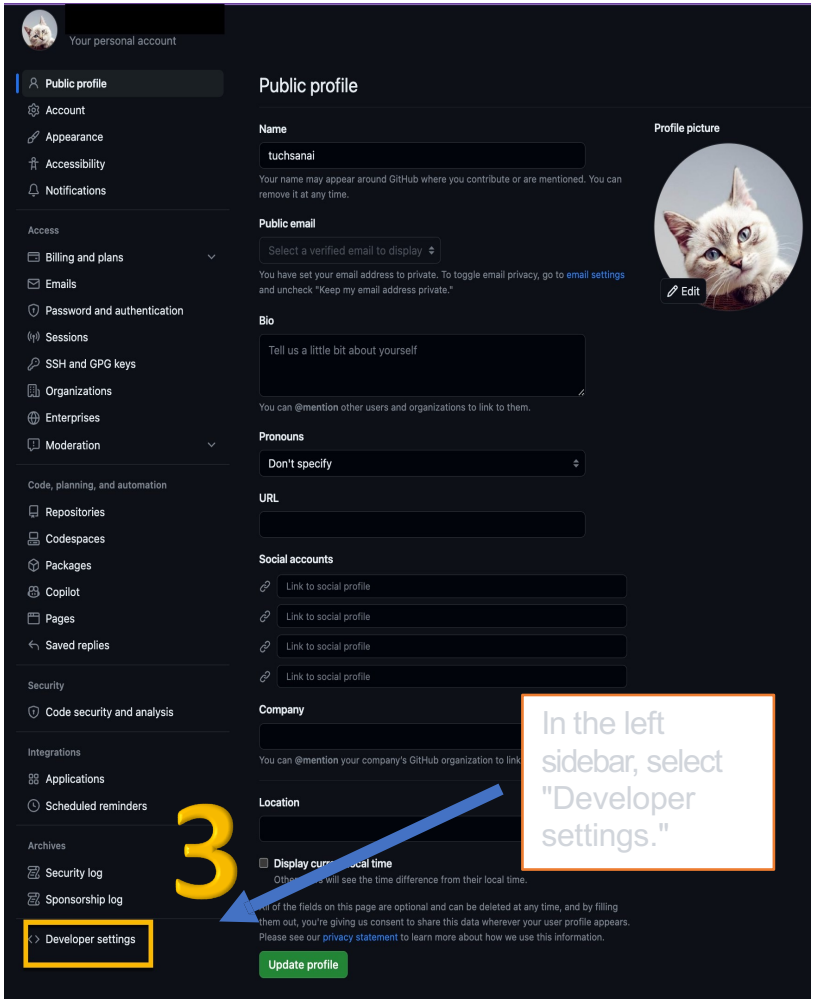
git clone https://github.com/account/repo.git

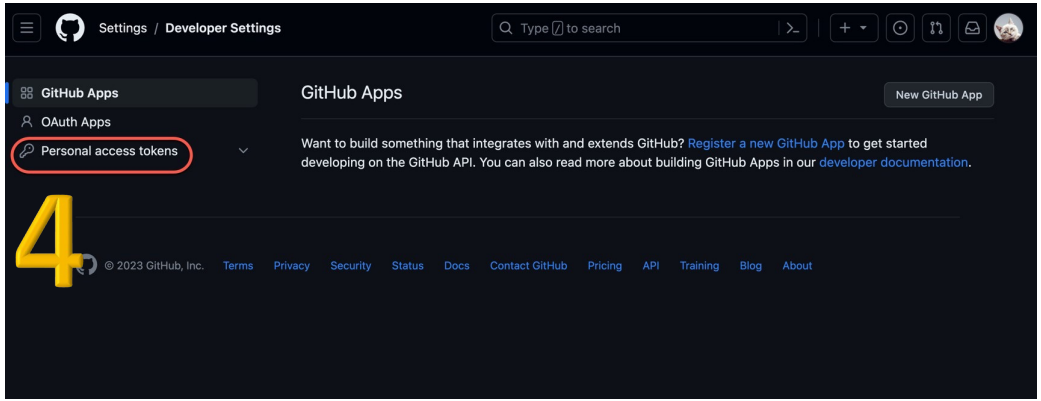
Create a Personal Access Token

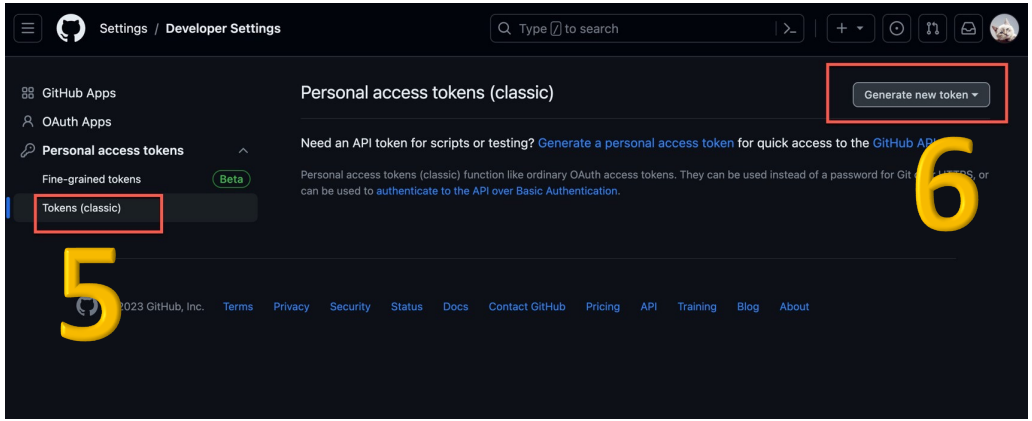
- Log in to your GitHub account that you want to create the PAT for.
- Click on your profile picture in the upper-right corner of the screen and select "Settings."
- In the left sidebar, select "Developer settings."
- Under the "Developer settings" menu, choose "Personal access tokens."
- Click on the "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.).

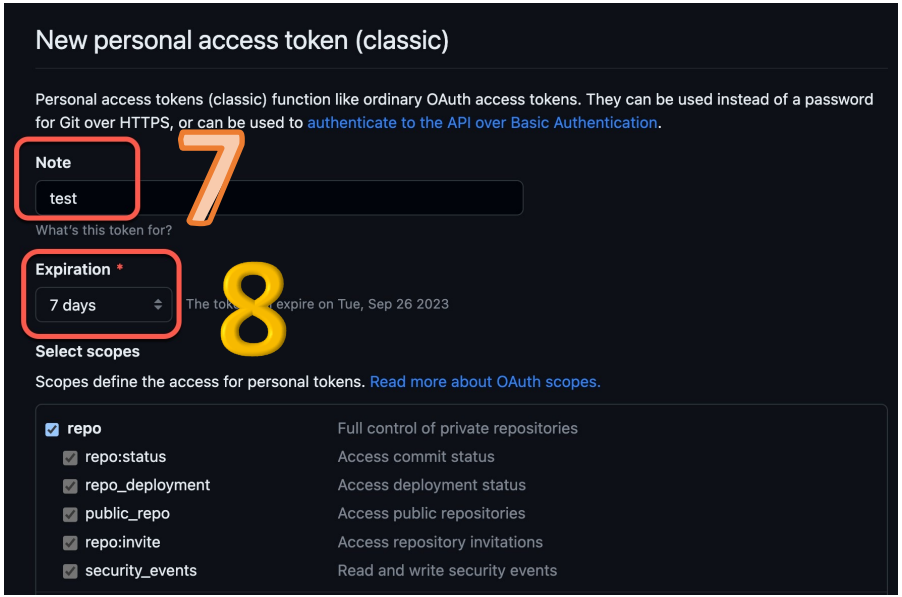

Type / to search
1
You unlocked new Achievements with private contributions! Show them off by including private contributions in your Profile in settings.
Pinned
Customize your pins
Public
Jupyter Notebook 1 7
Jupyter Notebook 10 14
Jupyter Notebook 1 1
746 contributions in the last year
Contribution settings
Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep
Mon
Wed
Fri
Click on your profile picture in the upper-right corner of the screen and select "Settings."

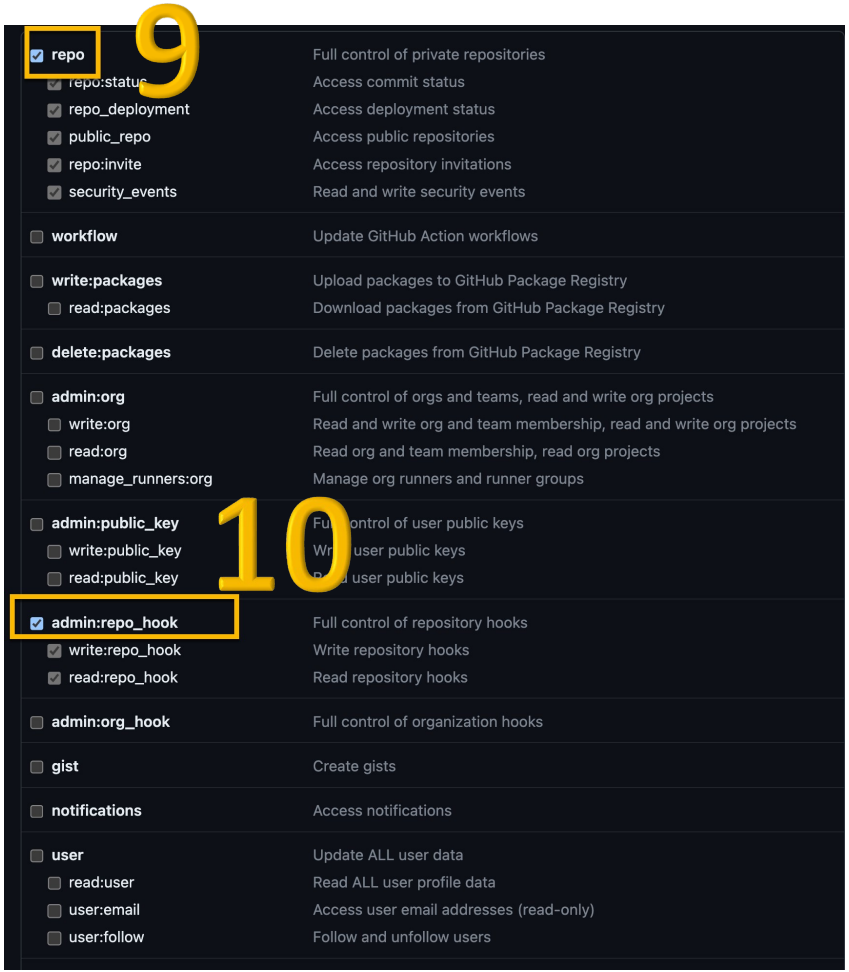

Set status
Your profile
Your repositories
Your projects
Your codespaces
Your organizations
Your enterprises
Your stars
Your sponsors
Your gists
Upgrade
Try Enterprise
Try Copilot
Feature preview
Settings
2
GitHub Docs
GitHub Support
Sign out

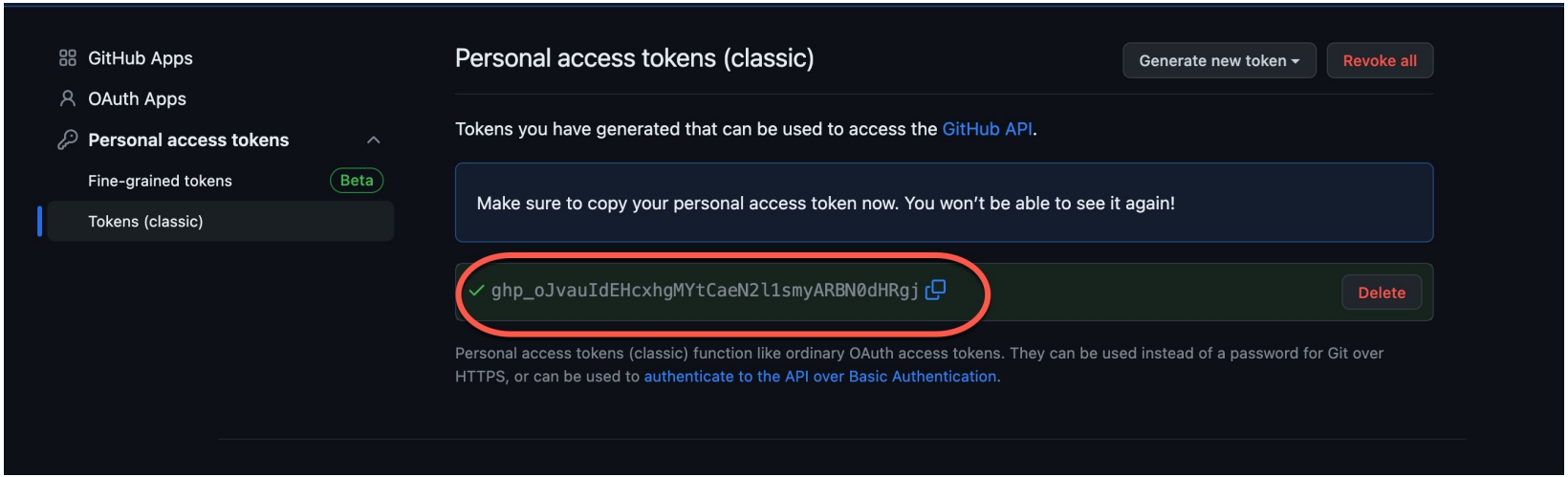

Your personal account
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and plans
Emails
Password and authentication
Sessions
SSH and GPG keys
Organizations
Enterprises
Moderation
Code, planning, and automation
Repositories
Codespaces
Packages
Copilot
Pages
Saved replies
Security
Code security and analysis
Integrations
Applications
Scheduled reminders
Archives
Security log
Sponsorship log
Developer settings
Public profile
Name
tuchsanai
Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.
Public email
Select a verified email to display
You have set your email address to private. To toggle email privacy, go to email settings and uncheck "Keep my email address private."
Bio
Tell us a little bit about yourself
You can @mention other users and organizations to link to them.
Pronouns
Don't specify
URL
Social accounts
Link to social profile
Link to social profile
Link to social profile
Link to social profile
Company
You can @mention your company's GitHub organization to link
Location
Display cur cal time
Oth s will see the time difference from their local time.
ll of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our privacy statement to learn more about how we use this information.
Update profile
Profile picture
Edit
3
In the left sidebar, select "Developer settings."


Settings / Developer Settings
Type / to search
GitHub Apps
OAuth Apps
Personal access tokens
GitHub Apps
New GitHub App
Want to build something that integrates with and extends GitHub? Register a new GitHub App to get started developing on the GitHub API. You can also read more about building GitHub Apps in our developer documentation.
4
© 2023 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About


Settings / Developer Settings
Type / to search
GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Beta
Tokens (classic)
Personal access tokens (classic)
Generate new token
Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API.
Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git
can be used to authenticate to the API over Basic Authentication.
6
5
2023 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About


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?
7
Expiration *
7 days
The tok expire on Tue, Sep 26 2023
8
Select scopes
Scopes define the access for personal tokens. Read more about OAuth scopes.
repo Full control of private repositories
repo:status Access commit status
repo_deployment Access deployment status
public_repo Access public repositories
repo:invite Access repository invitations
security_events Read and write security events


9
repo Full control of private repositories
repo:status Access commit status
repo_deployment Access deployment status
public_repo Access public repositories
repo:invite Access repository invitations
security_events Read and write security events
workflow Update GitHub Action workflows
write:packages Upload packages to GitHub Package Registry
read:packages Download packages from GitHub Package Registry
delete:packages Delete packages from GitHub Package Registry
admin:org Full control of orgs and teams, read and write org projects
write:org Read and write org and team membership, read and write org projects
read:org Read org and team membership, read org projects
manage_runners:org Manage org runners and runner groups
admin:public_key
write:public_key
read:public_key
10
admin:repo_hook Full control of repository hooks
write:repo_hook Write repository hooks
read:repo_hook Read repository hooks
admin:org_hook Full control of organization hooks
gist Create gists
notifications Access notifications
user Update ALL user data
read:user Read ALL user profile data
user:email Access user email addresses (read-only)
user:follow 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!
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
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.