Introduction To Big Data Analytics INSY 8413

ADVENTIST UNIVERSITY OF CENTRAL AFRICA

Instructor:

- Eric Maniraguha | eric.maniraguha@auca.ac.rw | LinkedIn Profile

6h00 pm – 8h50 pm

- Monday A-G104
- Tuesday E-G108
- Wednesday A-G104
- Thursday E-G108

June 2025

GitHub: Where your code meets its social network!

GitHub

Reference reading

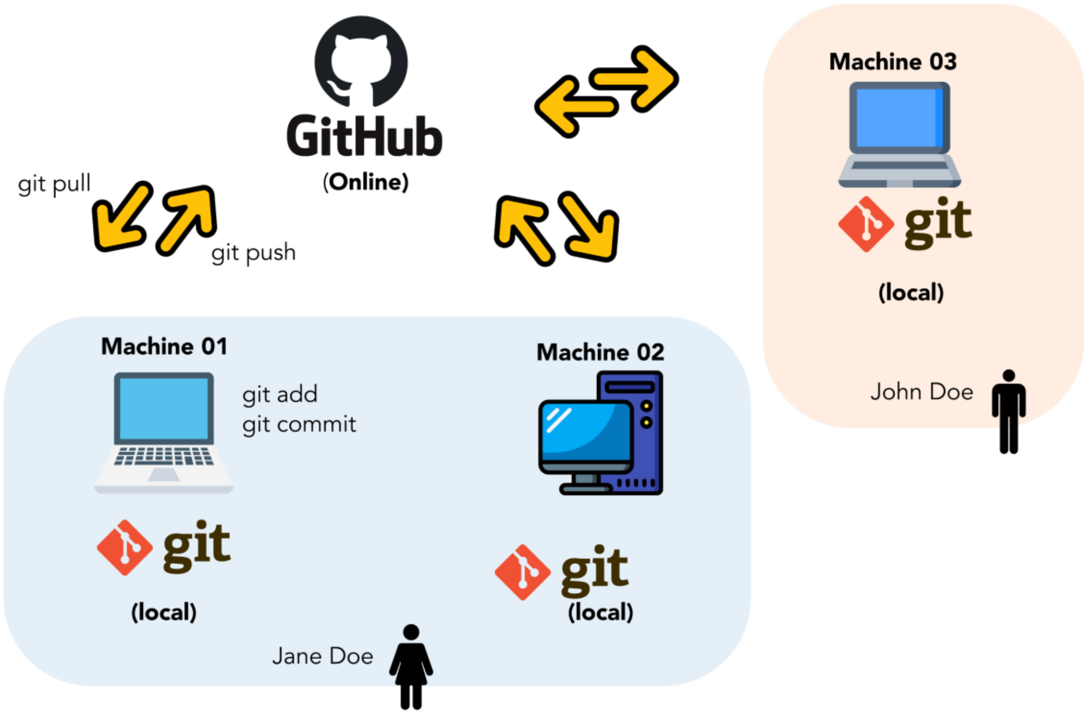
- YouTube Tutorial: Git Tutorial for Beginners: Learn Git in 1 Hour
- GitHub and Git Tutorial for Beginners
- Introduction to GitHub
- Intro to GitHub for version control
- DevOps Tools Mapping
- How to Design an Attractive GitHub Profile Readme...
- YouTube Tutorial: How to create Professional Github Readme Profile (Step By Step)
- Getting Started with R Markdown
- YouTube Tutorial: Git Merge Conflicts | How Merge Conflicts Happen | How to resolve Merge Conflicts | Merge Conflict
- Git & GitHub

Lecture 03 – Introduction to GitHub

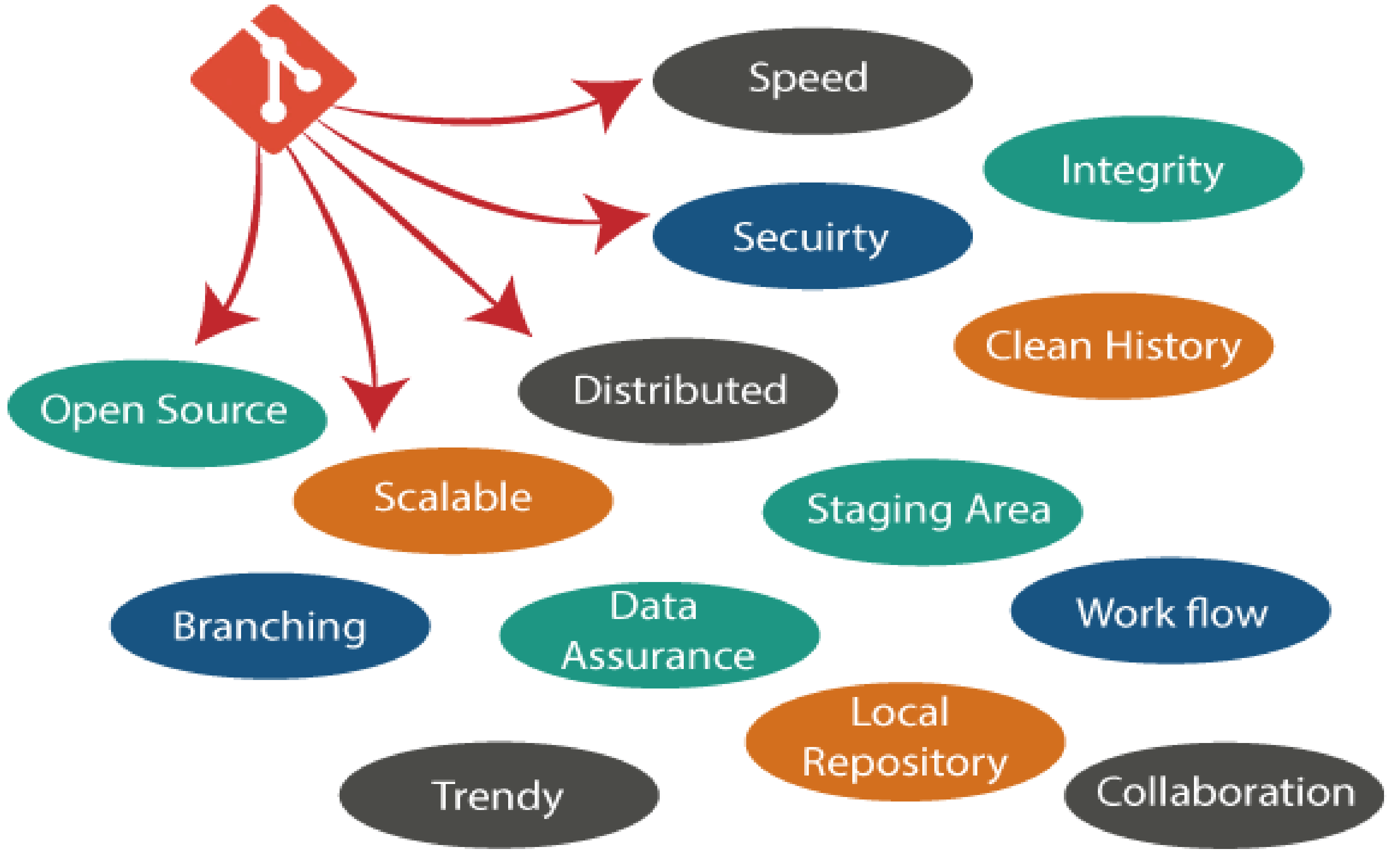
Introduction

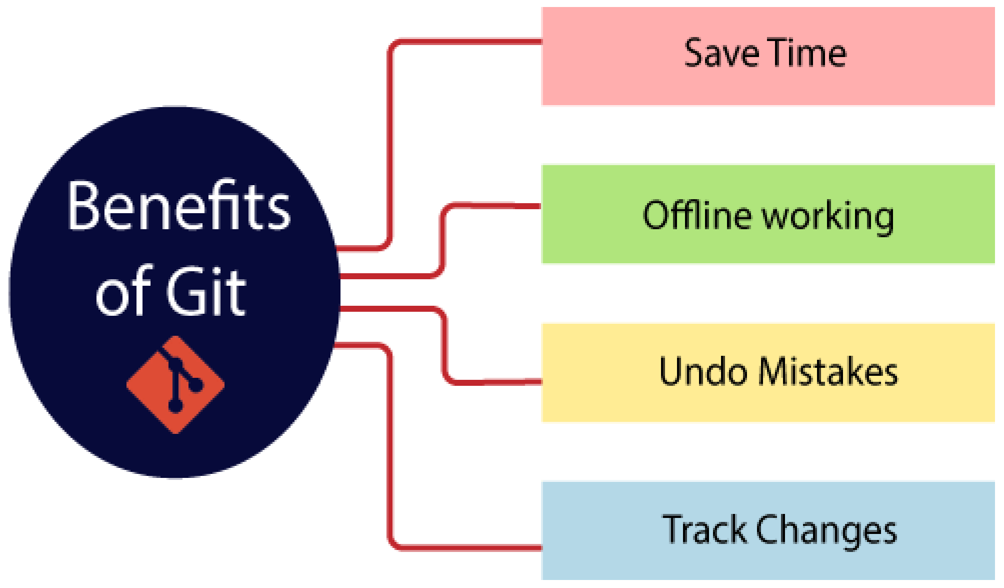
Course Overview

This course is designed to introduce students to GitHub, **a powerful platform for version control** and **collaboration**. Students will learn the basics of using GitHub, including **creating repositories, committing changes, branching, merging, and collaborating with others**. By the end of the course, students will be comfortable using GitHub for their projects.



Why Git?


Speed
Integrity
Secuirty
Clean History
Open Source
Distributed
Scalable
Staging Area
Branching
Data Assurance
Work flow
Local Repository
Trendy
Collaboration


Benefits of Git
Save Time
Offline working
Undo Mistakes
Track Changes

What You'll Learn

Version Control Basics
- Understand Git and how it powers GitHub.
- Track changes, revert code, and create meaningful commit histories.

Repositories & Branching
- Create and manage repositories.
- Use branches to work on new features without affecting the main project.

Collaboration
- Collaborate on projects using pull requests and reviews.
- Resolve conflicts effectively.

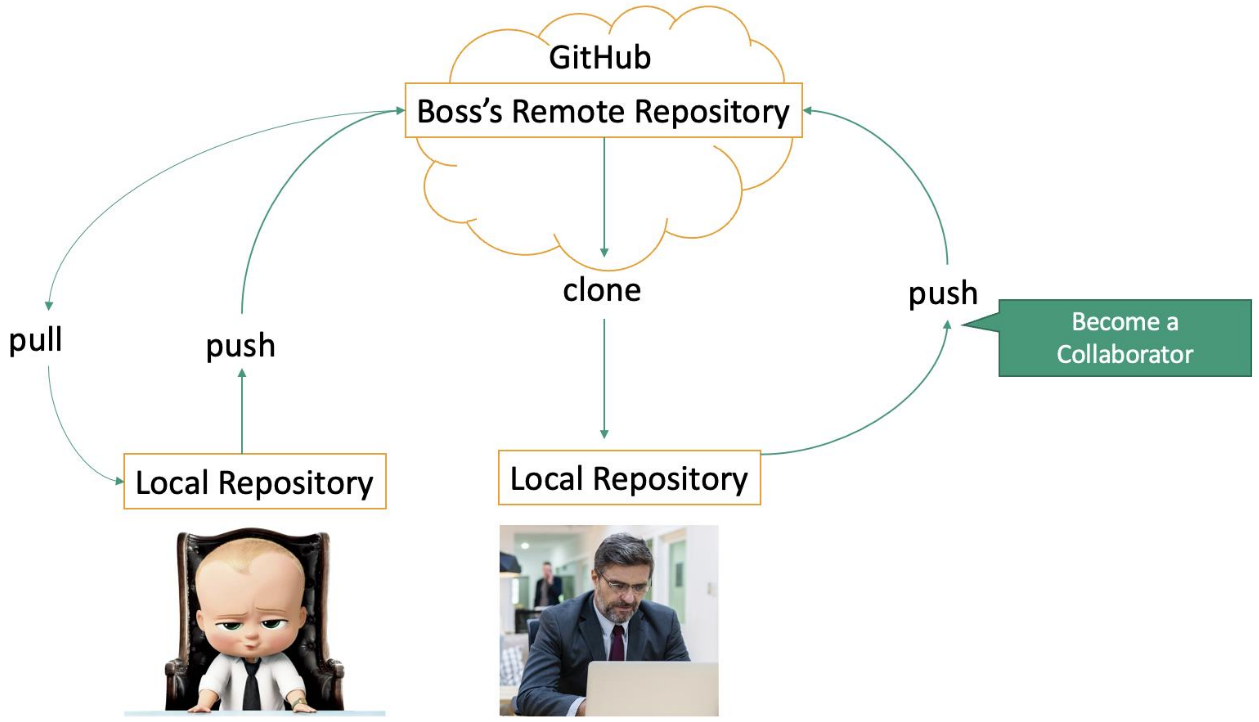
Project Management
- Leverage GitHub tools like Issues, Labels, and Projects to organize tasks.
- Use Actions for automated workflows.

Advanced Techniques
- Learn how to fork repositories and contribute to open source.
- Implement team workflows and CI/CD pipelines.

Best Practices
- Write effective README files.
- Maintain clean code and meaningful commit messages.

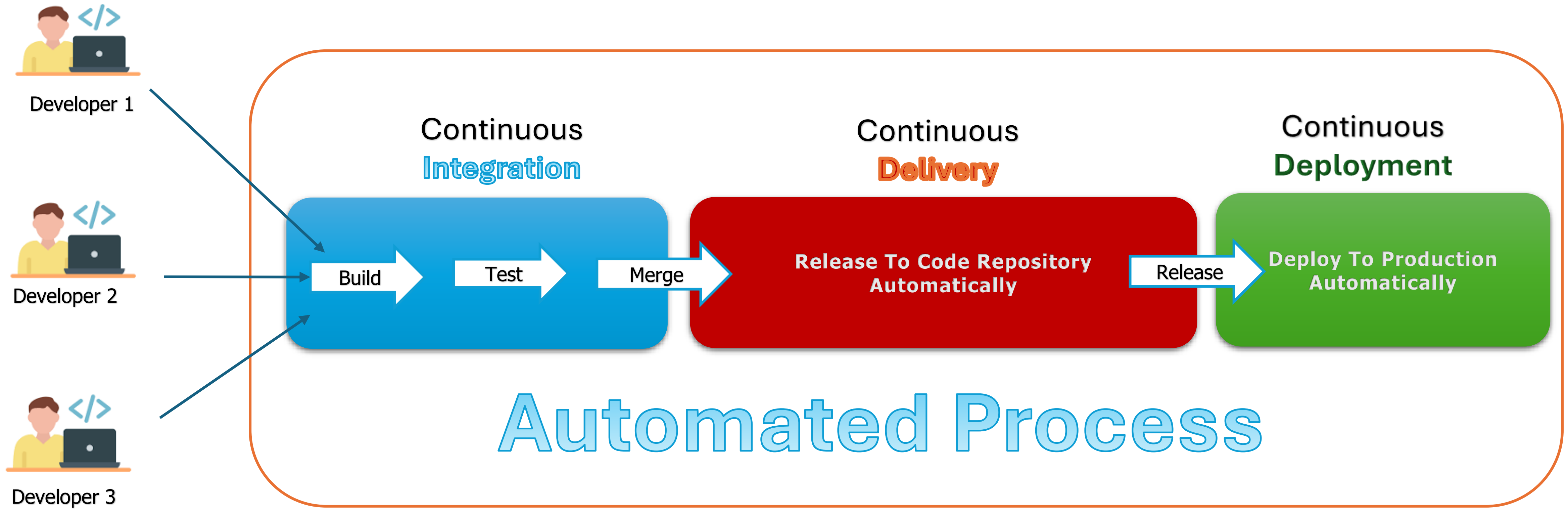
WHO SHOULD ATTEND?

- Developers and data scientists seeking collaboration skills.
- Students working on team projects.
- Professionals aiming to enhance productivity and code management.



Requirements:
- Basic knowledge of programming.
- A GitHub account.

Continuous Integration and Continuous Delivery (or Deployment)



CI/CD

Basic GitHub Terms

Repository (Repo):
A storage location for your project files, similar to a folder, where all the code, resources, and version history for your project are kept.

Branch:
A version of the repository used to develop new features without affecting the main (default) codebase.

Commit:
A snapshot of your changes in the repository, like saving progress with a message describing what was changed.

Push:
Sending your committed changes from your local repository to the remote repository on GitHub.

Pull:
Fetching and merging changes from the remote repository to your local repository.

Pull Request (PR):
A request to merge changes from one branch to another, often used for code review and collaboration.

Fork:
A personal copy of someone else's repository, allowing you to experiment or contribute to their project without affecting the original repository.

Clone:
Creating a copy of a GitHub repository on your local computer.

Basic GitHub Terms

Repository (Repo):
A storage location for your project files, similar to a folder, where all the code, resources, and version history for your project are kept.

Branch:
A version of the repository used to develop new features without affecting the main (default) codebase.

Commit:
A snapshot of your changes in the repository, like saving progress with a message describing what was changed.

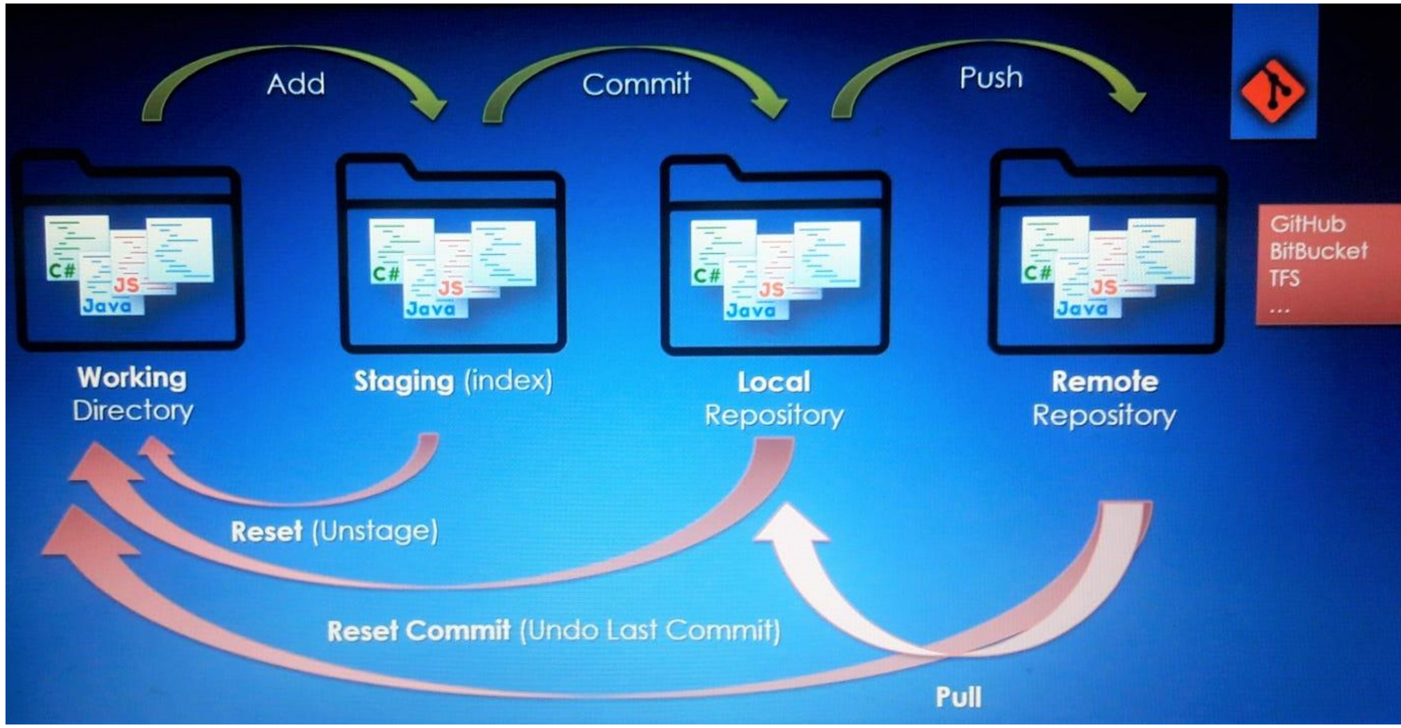
Push:
Sending your committed changes from your local repository to the remote repository on GitHub.

Pull:
Fetching and merging changes from the remote repository to your local repository.

Pull Request (PR):
A request to merge changes from one branch to another, often used for code review and collaboration.

Fork:
A personal copy of someone else's repository, allowing you to experiment or contribute to their project without affecting the original repository.

Clone:
Creating a copy of a GitHub repository on your local computer.



Source Image: https://medium.com/@derya.cortuk/git-terminology-101-366db34f1f4f

Git and Command Line Terms

Terminal or Command Line (CLI):
An interface for entering text commands to interact with your computer or GitHub.

CLI (Command Line Interface):
A text-based interface used to execute Git commands like *git add, git commit, and git push*.

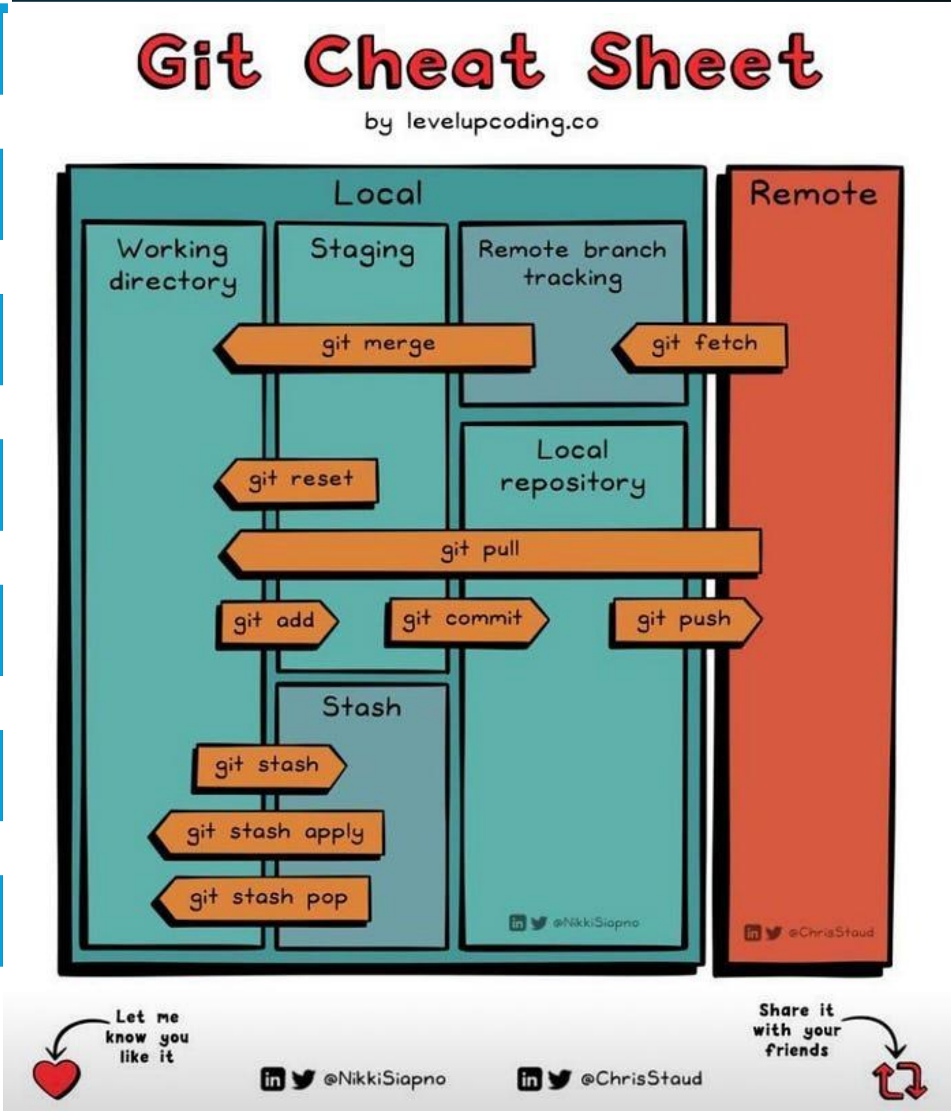
Directory:
A folder in your file system where your project files are stored.

Working Directory:
The directory on your computer where your repository files are currently located.

Staging Area:
A temporary area where changes are stored before committing them to the repository.

Merge:
Combining changes from different branches into a single branch.

HEAD:
Refers to the current branch you are working on in your repository.



Source Image: https://medium.com/@kittipat_1413/git-and-command-line-101-essential-skills-for-software-engineers-7a35b489bf36

GitHub Features

Feature	Description	Use Case
Repository (Repo)	Storage for project files, code, and version history.	Personal projects, team collaboration, or open-source contributions.
Branches	Parallel versions of a repository for development or testing.	Work on features independently without affecting the main branch.
Pull Requests (PRs)	Propose changes from one branch to another for review.	Collaboration, code review, and merging changes.
Issues	Tracks bugs, feature requests, and tasks.	Bug reporting, task management, and feature planning.
GitHub Actions	Automates workflows like CI/CD, testing, and deployments.	Automatically run tests, build code, or deploy applications.
GitHub Pages	Hosts static websites directly from a repository.	Create project documentation, personal portfolios, or simple websites.
Wiki	Documentation space for project guides and instructions.	Maintain detailed project explanations and collaboration instructions.
Projects	Kanban-style boards to visually track tasks and progress.	Organize tasks for teams using Issues and Pull Requests.
Labels	Tags to categorize Issues and Pull Requests.	Mark tasks as bug, enhancement, or help wanted for better organization.
Milestones	Group related Issues and Pull Requests around specific goals or deadlines.	Track progress towards completing a feature or project phase.
Discussions	Open-ended forum-like conversations in a repository.	Community-driven brainstorming or general queries.
Forking	Copies someone else’s repository into your account.	Experiment with changes or contribute to open-source projects.
Notifications	Alerts for repository activities like Issues and Pull Requests.	Stay updated on changes or discussions in your repositories.
GitHub Marketplace	Tools and integrations to enhance workflows.	CI/CD tools, security scans, and project management plugins.
Security Features	Tools like Dependabot and code scanning to ensure code safety.	Identify and fix vulnerabilities in the codebase.
Gists	Simple sharing of code snippets or text.	Share small code samples, notes, or configurations.
Collaborators	Add people to your repository with controlled access permissions.	Invite team members to contribute directly to private projects.
Insights	Analytics about code activity, contributors, and project history.	Monitor code frequency, contributor activity, and Pull Request timelines.
Webhooks	Send real-time updates to external applications.	Trigger notifications, deployments, or actions when changes occur.
GitHub Codespaces	Cloud-based, pre-configured development environments.	Start coding immediately without needing local setup.

Version Control Terms

Version Control Terms

Version Control System (VCS):
A system that tracks changes to files over time, allowing you to recall specific versions later.

Distributed VCS:
A system like Git that allows multiple developers to work on the same codebase independently.

Remote Repository:
A repository hosted on GitHub or another platform, accessible to collaborators online.

Local Repository:
A repository stored on your computer.

Conflict:
A situation where changes from different branches or contributors overlap and need manual resolution.

Command	**Description**
git fetch	Downloads **remote changes** (branches, tags, etc.) into your local repo **without applying them** to your working branch. It updates the remote tracking branches like origin/main.
git pull	Does **git fetch + git merge** automatically. It both **downloads and applies** the changes to your current branch.

Unlocking Git: Understanding Commands Through Diagrams

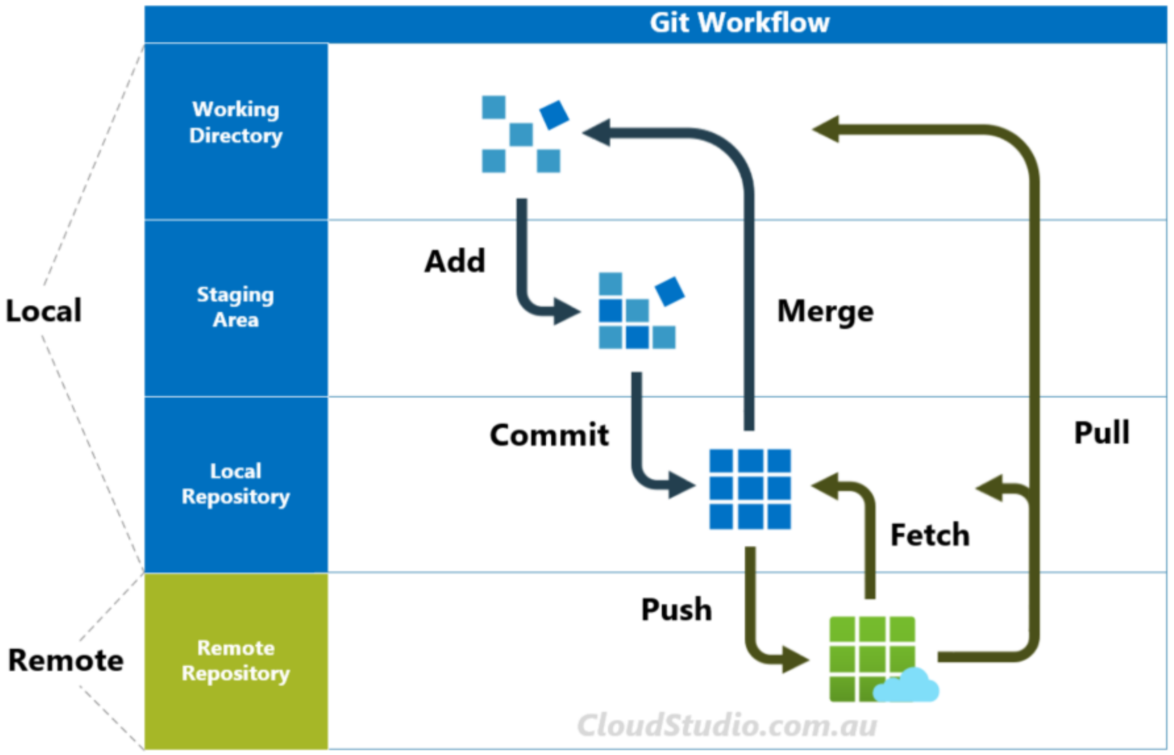


Image Source: https://cloudstudio.com.au/2021/06/26/git-command/

1

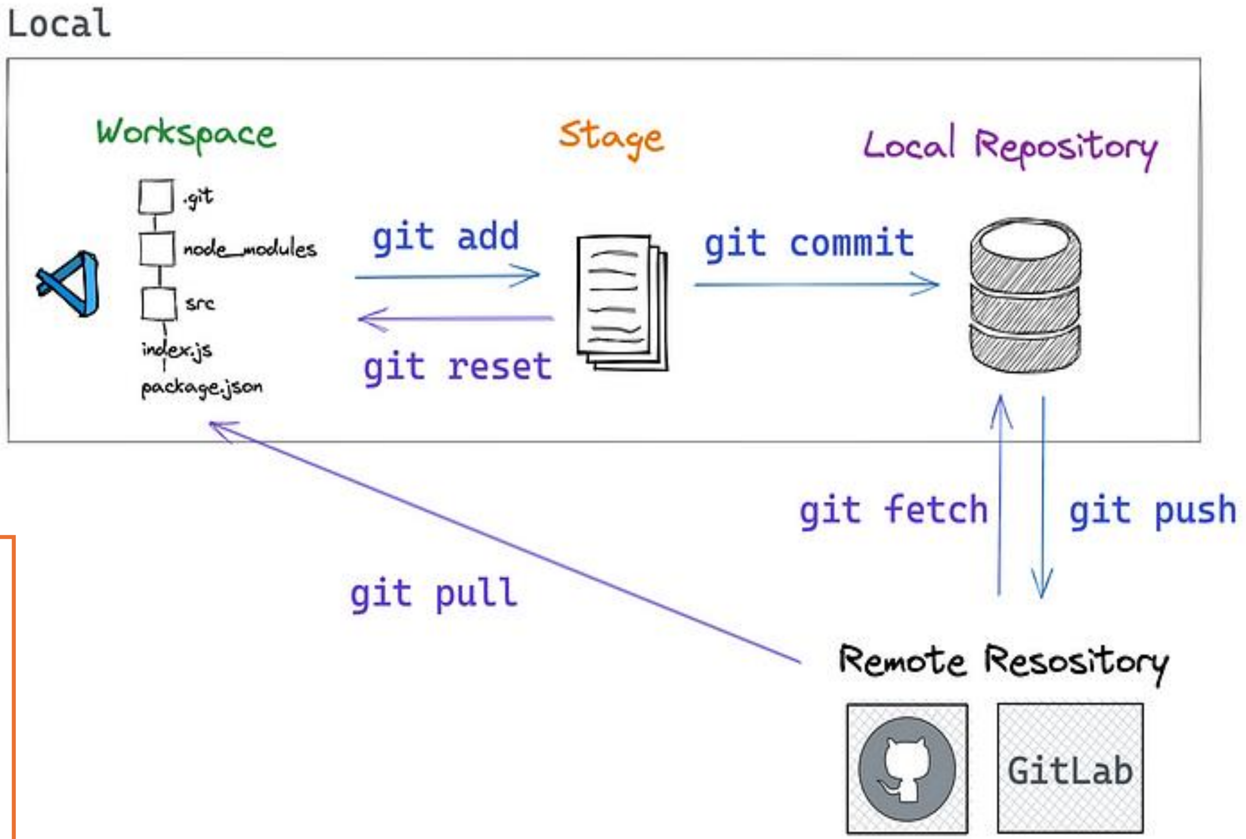


Image Source: https://programming.earthonline.us/you-can-master-git-git-commands-with-these-diagrams-40a0b2f5cc42

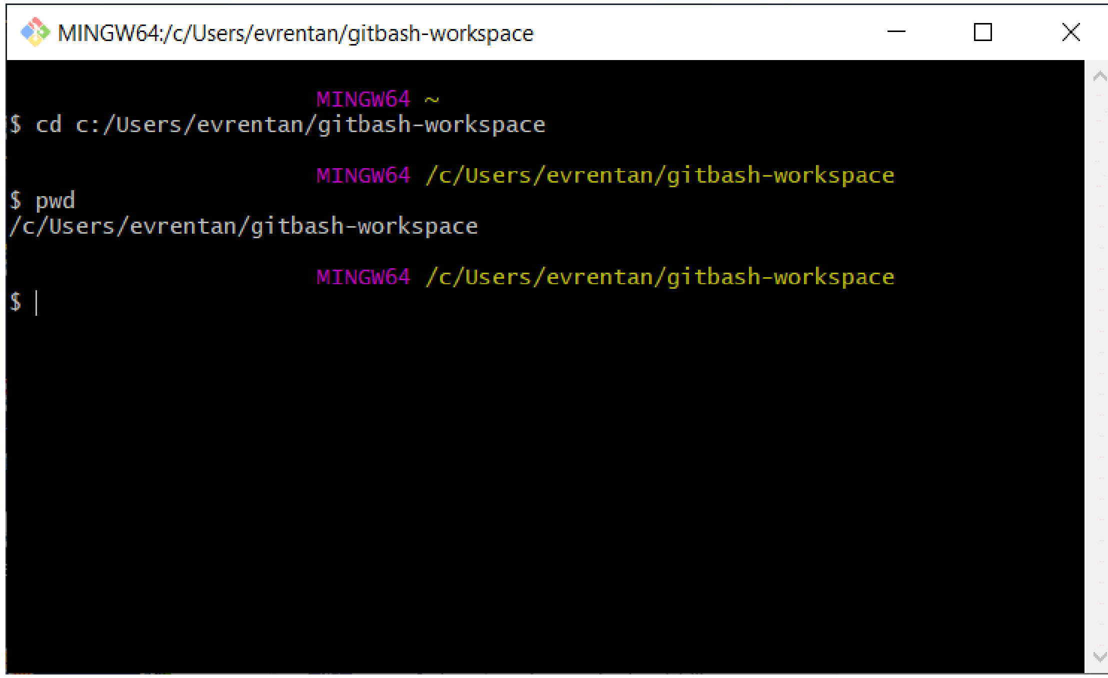
2

How to get started

Please register on the GitHub website.

- On your computer, you need to install **Git** first. The process will depend on your operating system: please follow the instructions below by clicking the relevant button.

Install Git Bash on Windows


MINGW64:/c/Users/evrentan/gitbash-workspace

```
                        MINGW64 ~
$ cd c:/Users/evrentan/gitbash-workspace

                        MINGW64 /c/Users/evrentan/gitbash-workspace
$ pwd
/c/Users/evrentan/gitbash-workspace

                        MINGW64 /c/Users/evrentan/gitbash-workspace
$ |
```

Basic Commands

The basic commands include initializing the Git repository, saving changes, checking logs, pushing the changes to the remote server, and merging.

Git Command Table

Command	**Description**
git init	Create a Git repository in a local directory.
git clone <url>	Copy the entire repository from a remote server to a local directory. Also works for local repos.
git add <file.txt>	Add a single file or multiple files and folders to the staging area.
git commit -m "Message"	Create a snapshot of changes and save it in the repository.
git config	Set user-specific configurations like email, username, and file format.
git status	Show the list of changed files or files that have yet to be staged and committed.
git push <remote-name> <branch-name>	Send local commits to the remote branch of the repository.
git checkout -b <branch-name>	Create a new branch and switch to it.
git remote -v	View all remote repositories.
git remote add <remote-name> <url>	Add a remote server to the local repository.
git branch -d <branch-name>	Delete a branch.
git pull	Merge commits from a remote repository into a local directory.
git merge <branch-name>	After resolving merge conflicts, blend the selected branch into the current branch.
git log	Show a detailed list of commits for the current branch.

Mastering Markdown for GitHub

What is Markdown?

- A lightweight maarkup language to format plain text.
- Used for documentation, README files, and wikis on GitHub.

Key Markdown Syntax

Element	Syntax	Example Output
Headings	*# H1, ## H2, ..., ###### H6*	# Heading 1 → **Heading 1**
Bold	***text***	**bold** → **bold**
Italic	**text**	*italic* → italic
Lists	*- Item (unordered), 1. Item (ordered)*	- Item or 1. Item → • Item1. Item
Links	*[text](URL)*	[GitHub](https://github.com) → GitHub
Images	*![alt](URL)*	![Logo](https://github.githubassets.com/images/modules/logos_page/GitHub-Mark.png) →
Code	*\`inline\` or triple backticks*	\`code\` → code \`\`\`code block\`\`\`
Tables	*Use pipes \`*	and dashes---\`

Why Learn Markdown?

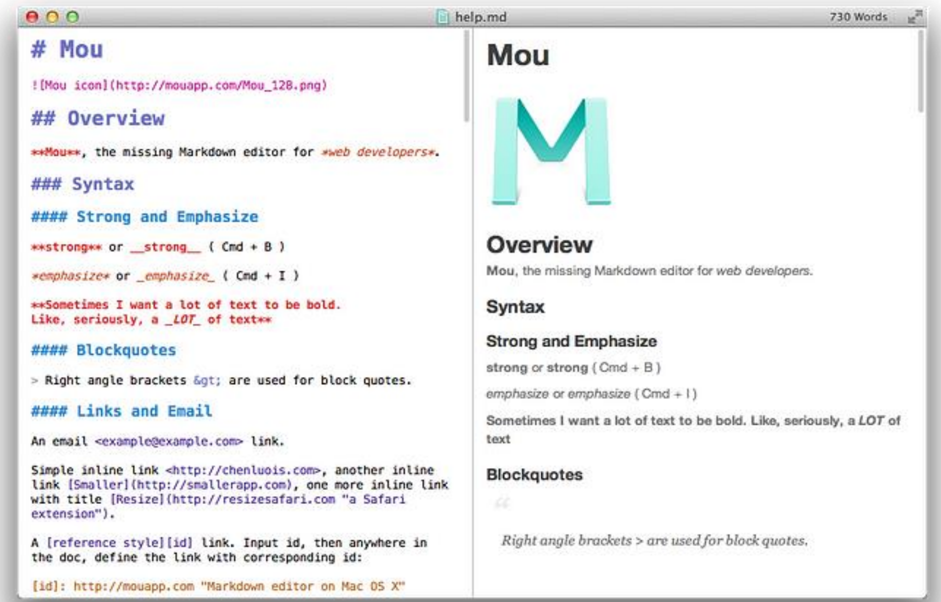
Efficiency: Quick and simple syntax.

Versatility: Powers README files, GitHub Pages, and wikis.

Collaboration: Clear formatting for team projects.

Practice Tip

Use GitHub's **Preview** feature while editing Markdown to see live formatting.



help.md 730 Words

```
# Mou

![Mou icon](http://mouapp.com/Mou_128.png)

## Overview

**Mou**, the missing Markdown editor for *web developers*.

### Syntax

#### Strong and Emphasize

**strong** or __strong__ ( Cmd + B )

*emphasize* or _emphasize_ ( Cmd + I )

**Sometimes I want a lot of text to be bold.
Like, seriously, a _LOT_ of text**

#### Blockquotes

> Right angle brackets &gt; are used for block quotes.

#### Links and Email

An email <example@example.com> link.

Simple inline link <http://chenluois.com>, another inline
link [Smaller](http://smallerapp.com), one more inline link
with title [Resize](http://resizesafari.com "a Safari
extension").

A [reference style][id] link. Input id, then anywhere in
the doc, define the link with corresponding id:

[id]: http://mouapp.com "Markdown editor on Mac OS X"
```

Mou

Overview

Mou, the missing Markdown editor for *web developers*.

Syntax

Strong and Emphasize

strong or **strong** (Cmd + B)

emphasize or *emphasize* (Cmd + I)

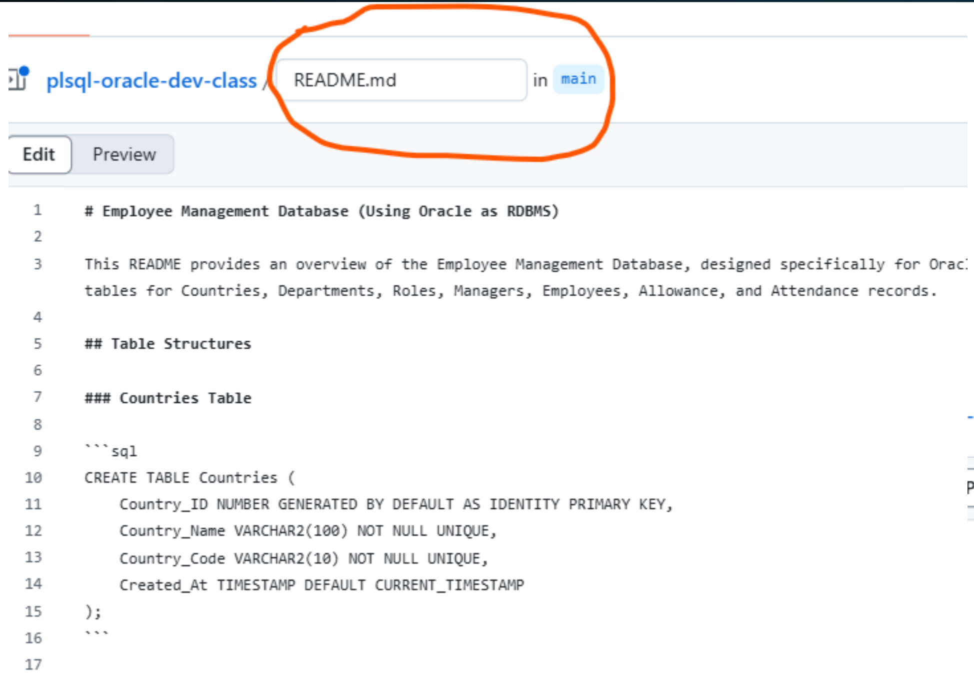
Sometimes I want a lot of text to be bold. Like, seriously, a *LOT* of text

Blockquotes

> *Right angle brackets > are used for block quotes.*

Vs code **Google Docs to Markdown** **GitHub Page** **Slack**

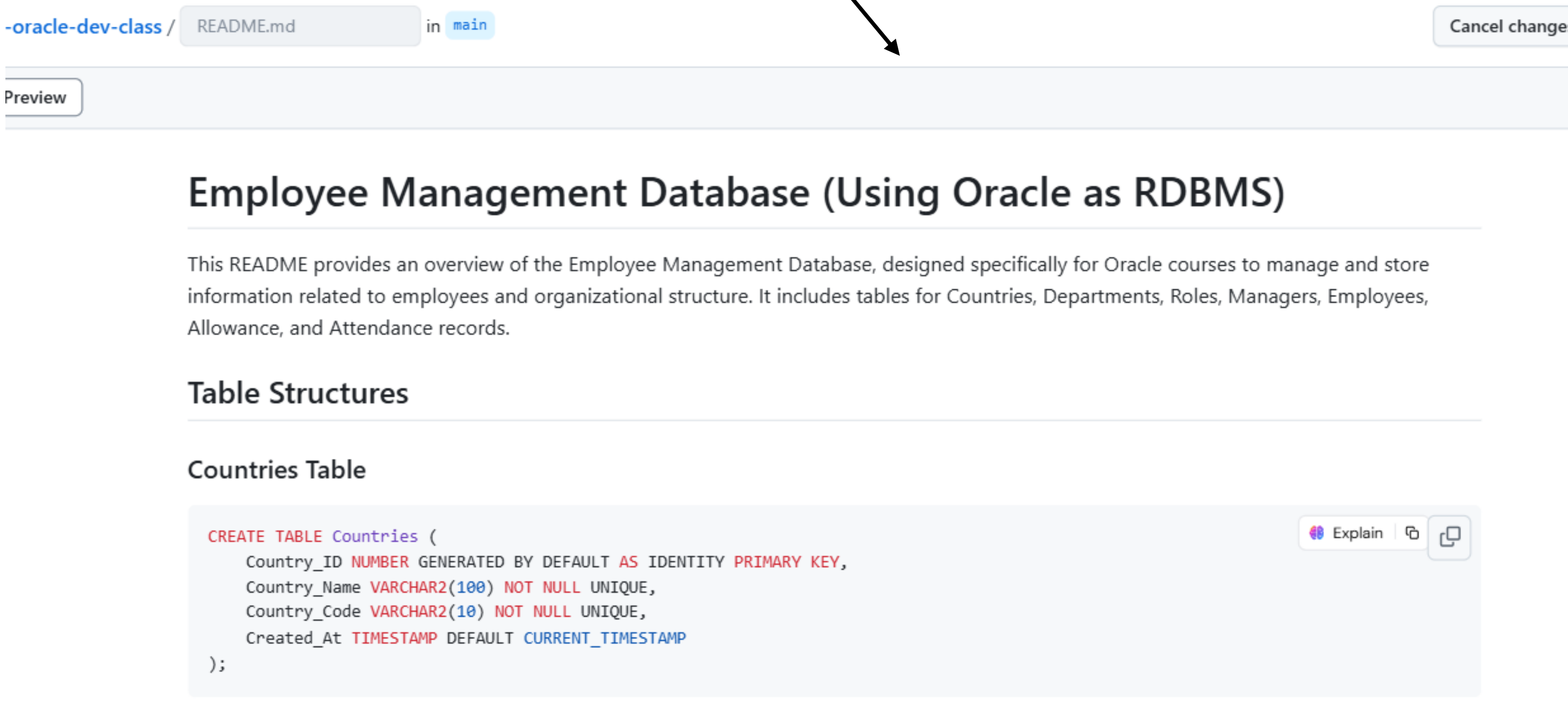
Example Markdown for GitHub



~~~
# Employee Management Database (Using Oracle as RDBMS)

This README provides an overview of the Employee Management Database, designed specifically for Orac
tables for Countries, Departments, Roles, Managers, Employees, Allowance, and Attendance records.

## Table Structures

### Countries Table

```sql
CREATE TABLE Countries (
    Country_ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    Country_Name VARCHAR2(100) NOT NULL UNIQUE,
    Country_Code VARCHAR2(10) NOT NULL UNIQUE,
    Created_At TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```
~~~

Markdown text

Results after Markdown text



Employee Management Database (Using Oracle as RDBMS)

This README provides an overview of the Employee Management Database, designed specifically for Oracle courses to manage and store information related to employees and organizational structure. It includes tables for Countries, Departments, Roles, Managers, Employees, Allowance, and Attendance records.

Table Structures

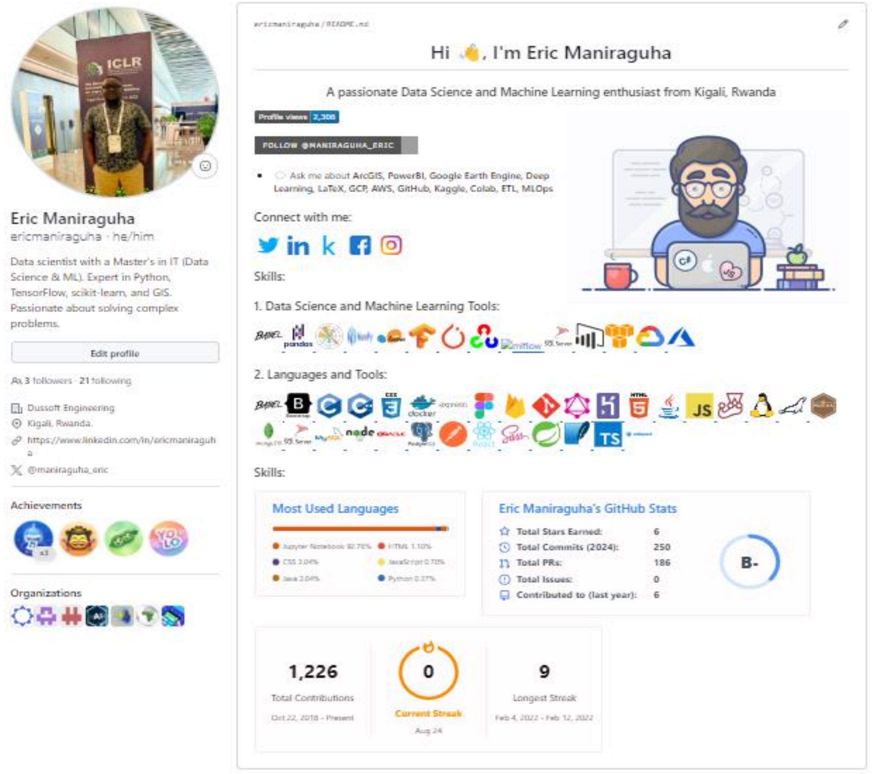
Countries Table

```sql
CREATE TABLE Countries (
    Country_ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    Country_Name VARCHAR2(100) NOT NULL UNIQUE,
    Country_Code VARCHAR2(10) NOT NULL UNIQUE,
    Created_At TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

What is a GitHub User Profile?

A GitHub user profile is a public page that represents a user's identity on GitHub. It shows information about the user, their repositories, contributions, and activities.

Profile Picture and Bio	**Profile Picture**: A photo or avatar to represent the user (optional).
	Bio: A short description about the user, such as their role, interests, or expertise.
Pinned Repositories	Highlight important repositories the user wants to showcase, such as personal projects or open-source contributions.
Contribution Graph	A heatmap showing the user's contributions (commits, pull requests, issues) over the past year.
Repositories	A list of all public repositories owned by the user, including the name, description, and programming languages used.
Followers and Following	**Followers**: People who are subscribed to the user's updates.
	Following: Other GitHub users the profile owner is following.
Activity Feed	Displays recent activities such as commits, pull requests, and discussions.
GitHub README Profile (Optional)	A custom section created by adding a README.md file in a repository named after the user's username.
	This can include skills, achievements, current projects, or links to portfolios.
Organizations	Lists any organizations the user is a part of on GitHub.
Stars	A collection of repositories the user has liked or "starred."
Gists	A list of code snippets or text files the user has shared publicly or privately.


Eric Maniraguha
ericmaniraguha · he/him

Data scientist with a Master's in IT (Data Science & ML). Expert in Python, TensorFlow, scikit-learn, and GIS. Passionate about solving complex problems.

Edit profile

3 followers · 21 following

Dussoft Engineering
Kigali, Rwanda.
https://www.linkedin.com/in/ericmaniraguha
@maniraguha_eric

Achievements

Organizations

ericmaniraguha / README.md

Hi 👋, I'm Eric Maniraguha

A passionate Data Science and Machine Learning enthusiast from Kigali, Rwanda

Profile views 2,306

FOLLOW @MANIRAGUHA_ERIC

- Ask me about ArcGIS, PowerBI, Google Earth Engine, Deep Learning, LaTeX, GCP, AWS, GitHub, Kaggle, Colab, ETL, MLOps

Connect with me:

Skills:

1. Data Science and Machine Learning Tools:

2. Languages and Tools:

Skills:

Most Used Languages

- Jupyter Notebook 92.78%
- HTML 1.10%
- CSS 3.04%
- JavaScript 0.70%
- Java 2.04%
- Python 0.37%

Eric Maniraguha's GitHub Stats

Total Stars Earned:	6
Total Commits (2024):	250
Total PRs:	186
Total Issues:	0
Contributed to (last year):	6

B-

1,226
Total Contributions
Oct 22, 2018 - Present

0
Current Streak
Aug 24

9
Longest Streak
Feb 4, 2022 - Feb 12, 2022

Source Image: Profile readme

How to Enhance Your GitHub Profile

Use a Profile README:

Create a repository named after your username (e.g., username/username) and add a README.md file with personalized content.

Customize Your Bio:

Include links to your portfolio, LinkedIn, or website. Use emojis and markdown for a visually appealing bio.

Pin Repositories Strategically:

Highlight projects that showcase your skills and achievements.

Stay Active:

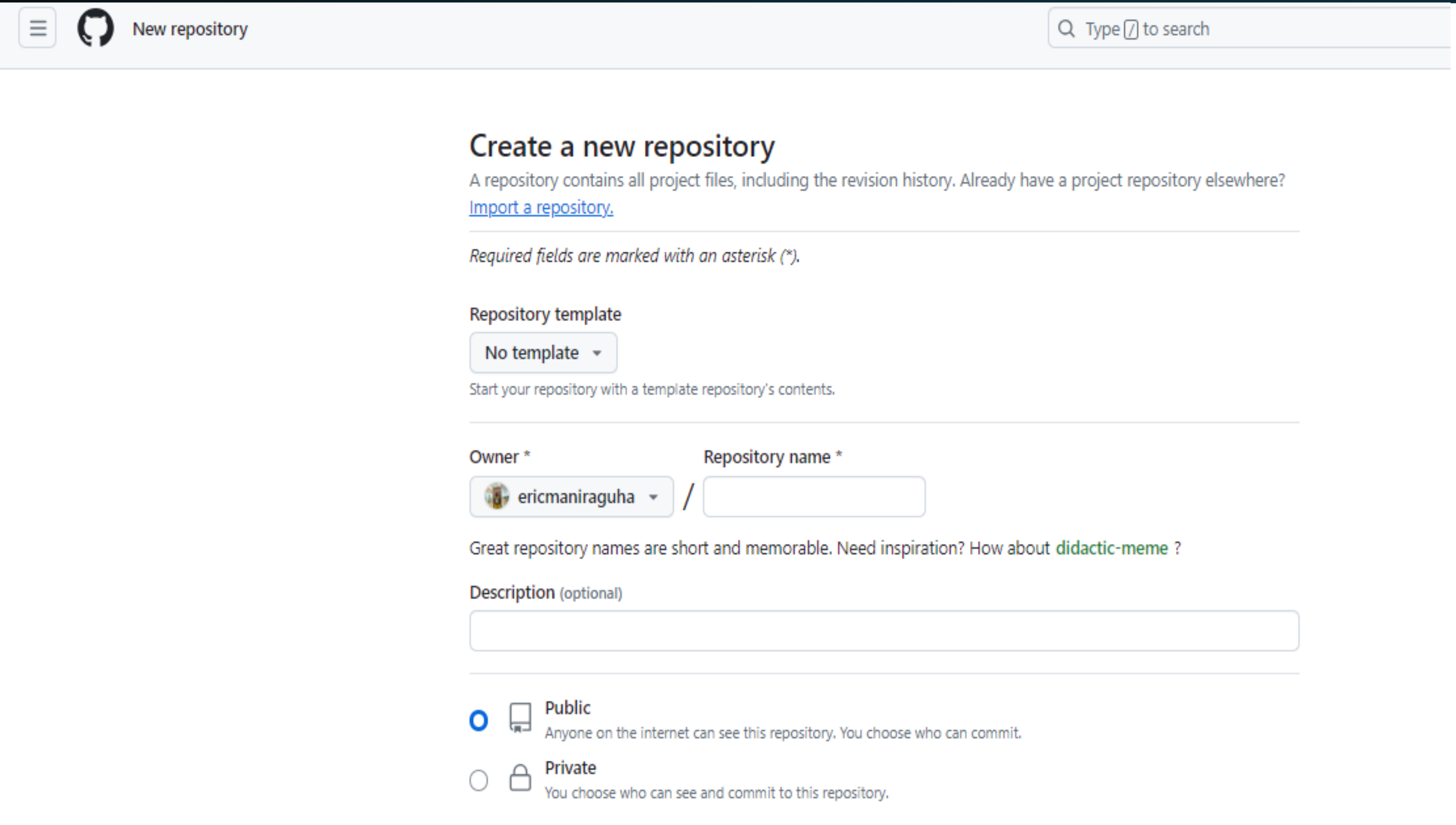
Regularly contribute to repositories, create new projects, or maintain existing ones.

Engage with the Community:

Participate in discussions, report issues, or contribute to other repositories.

A GitHub profile acts as a professional portfolio that demonstrates your skills and contributions, making it valuable for recruiters, collaborators, and the developer community.

GitHub User Profile



New repository

Type / to search

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Repository template

No template

Start your repository with a template repository's contents.

Owner * ericmaniraguha / Repository name *

Great repository names are short and memorable. Need inspiration? How about didactic-meme ?

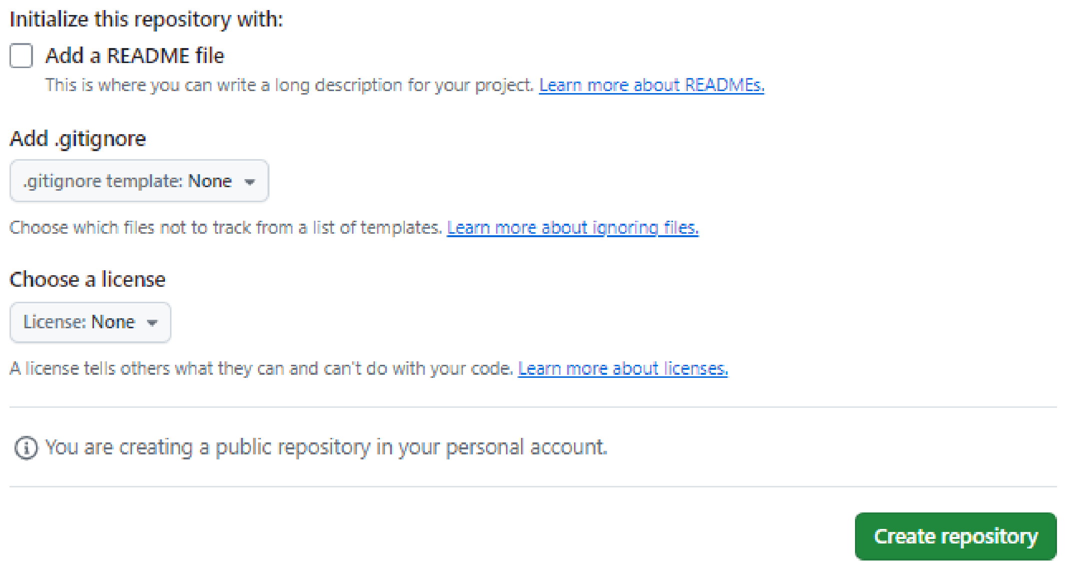
Description (optional)

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Adding a profile README

In the upper-right corner of any page, select + , then click **New repository**.



Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

You are creating a public repository in your personal account.

Create repository

Removing a profile README

The profile README will be removed from your GitHub profile if any of the following apply:

- The README file is removed or made empty.
- The repository is made private.
- The repository name no longer matches your username due to a change in either or both names.

The method you choose depends upon your needs, but if you're unsure, we recommend making your repository private. For steps on how to make your repository private, see " Setting repository visibility."

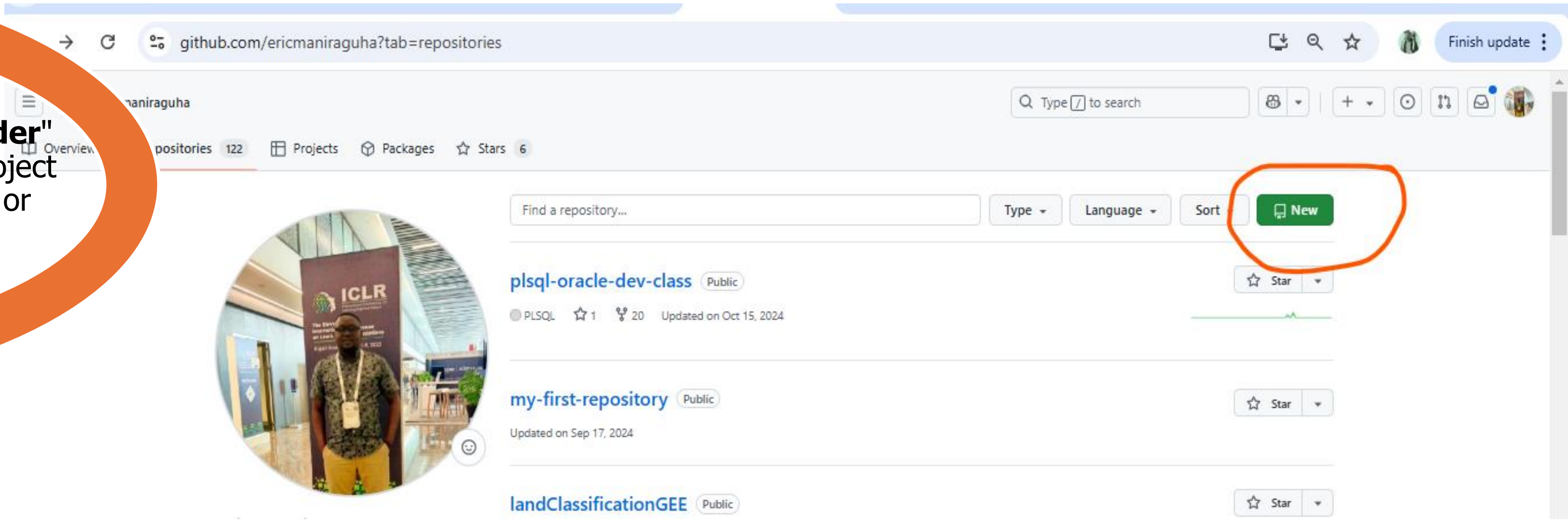
Step I. Create your own repository

Think of a repository (or repo) as **the "main folder"** for a project, where everything related to that project is stored. **A repo** can **contain multiple folders** or individual files, all organized under one project.

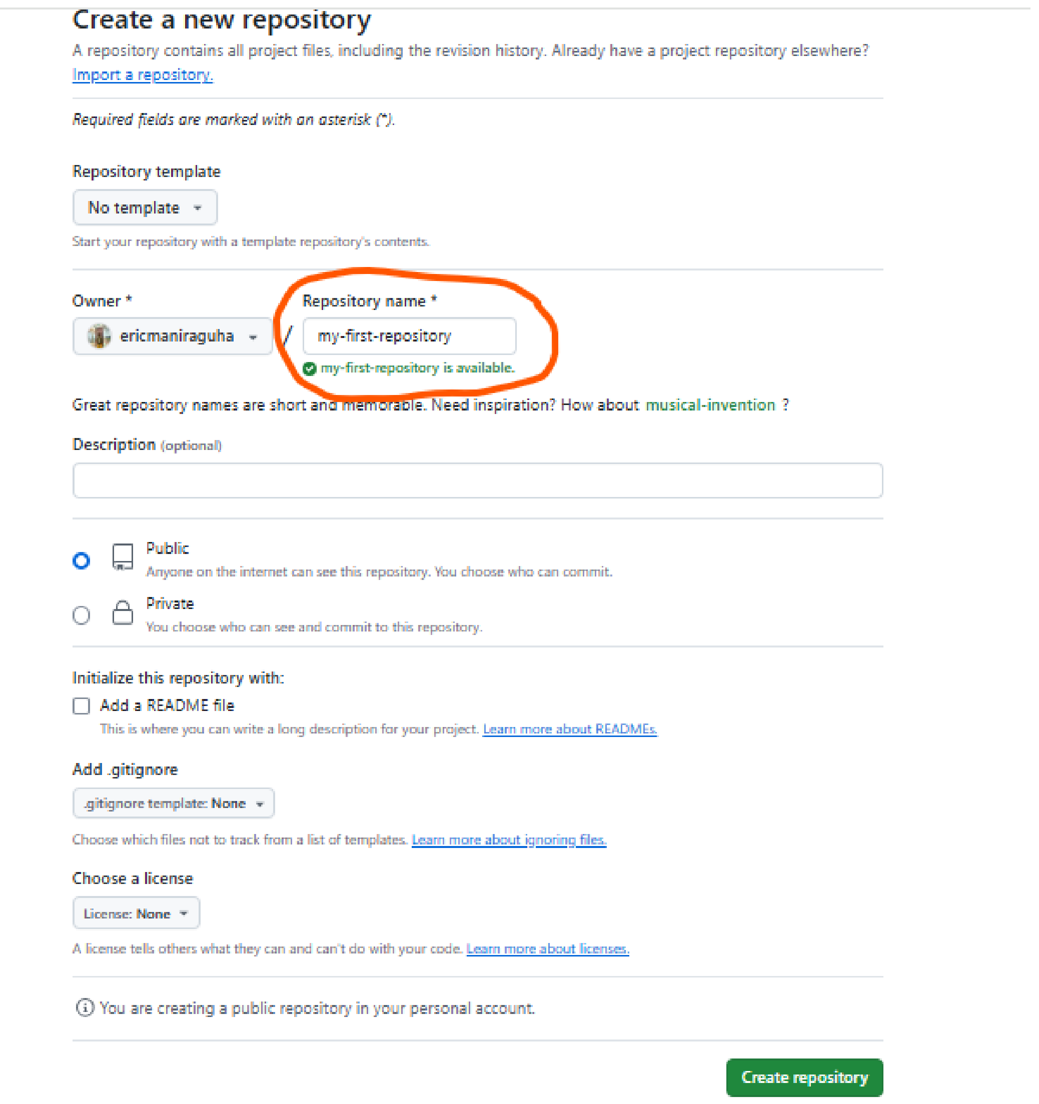
You'll maintain both a local copy (on your computer) and an online copy (on GitHub) of all the files in your repository.

The typical GitHub workflow is captured by the phrase **"commit-pull-push.**

1. Create your own repository and project folder structure


github.com/ericmaniraguha?tab=repositories
Finish update
aniraguha
Type / to search
Overview
positories 122
Projects
Packages
Stars 6
Find a repository...
Type
Language
Sort
New
plsql-oracle-dev-class Public
PLSQL 1 20 Updated on Oct 15, 2024
Star
my-first-repository Public
Updated on Sep 17, 2024
Star
landClassificationGEE Public
Star

Step II. Create your own repository



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Repository template

No template

Start your repository with a template repository's contents.

Owner * ericmaniraguha / Repository name * my-first-repository

my-first-repository is available.

Great repository names are short and memorable. Need inspiration? How about musical-invention ?

Description (optional)

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

You are creating a public repository in your personal account.

Create repository

Here's a concise version for a PowerPoint slide:

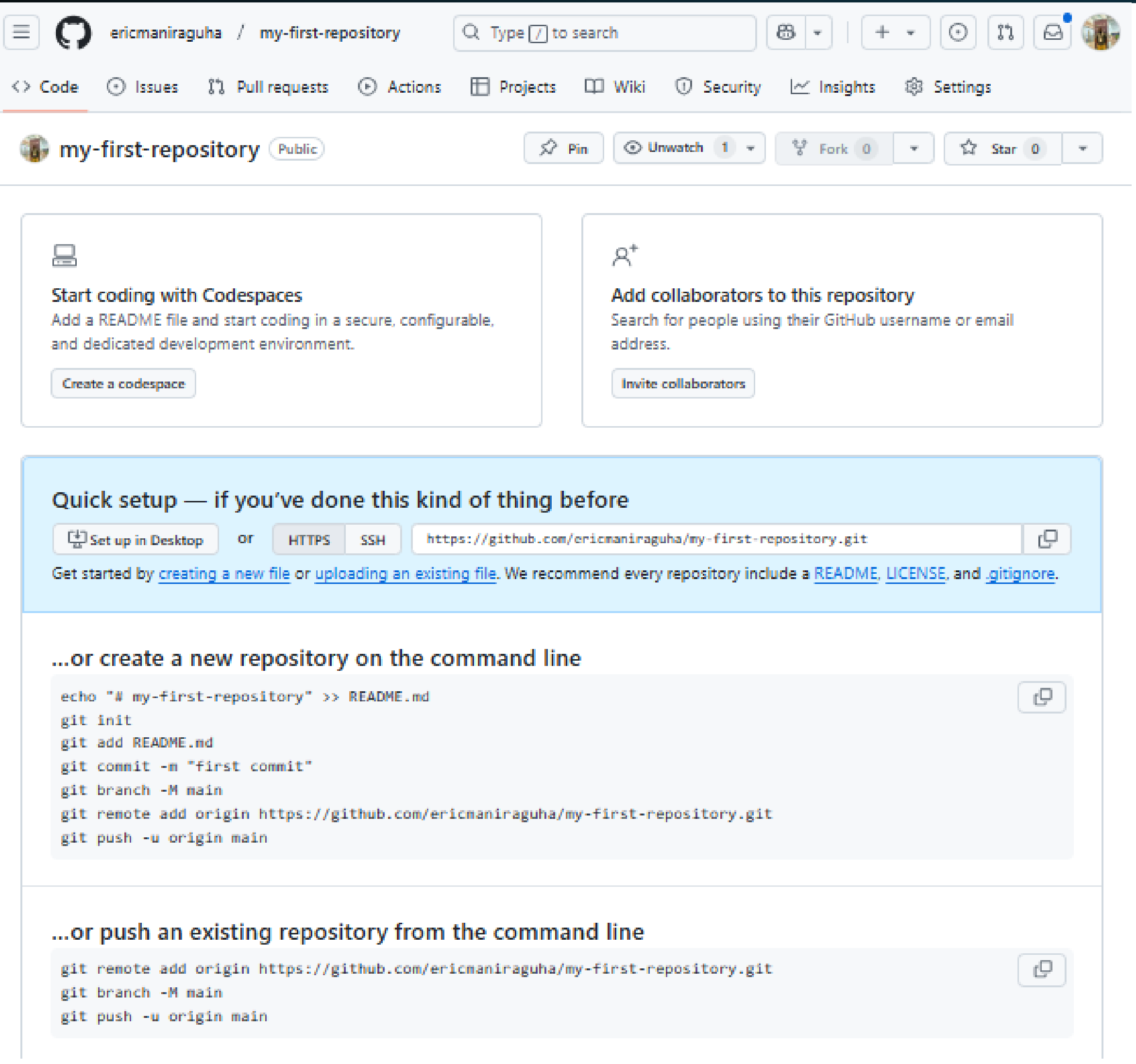
Initialize Repo:
- Create a *README.md* for project details, purpose, and licensing.

Markdown Format:
- Use ***.md*** for structured project documentation.

***.gitignore* File:**
- Exclude unnecessary files from the repo.

Language Template:
- Select the relevant template (e.g., R) in the ***.gitignore***

Created Repository



Start coding with Codespaces
Add a README file and start coding in a secure, configurable, and dedicated development environment.
Create a codespace

Add collaborators to this repository
Search for people using their GitHub username or email address.
Invite collaborators

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/ericmaniraguha/my-first-repository.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# my-first-repository" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/ericmaniraguha/my-first-repository.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/ericmaniraguha/my-first-repository.git
git branch -M main
git push -u origin main
```

After creating **my-first-repository,** I can push my code from local repository to remote repository

Repository is created

Explanation:

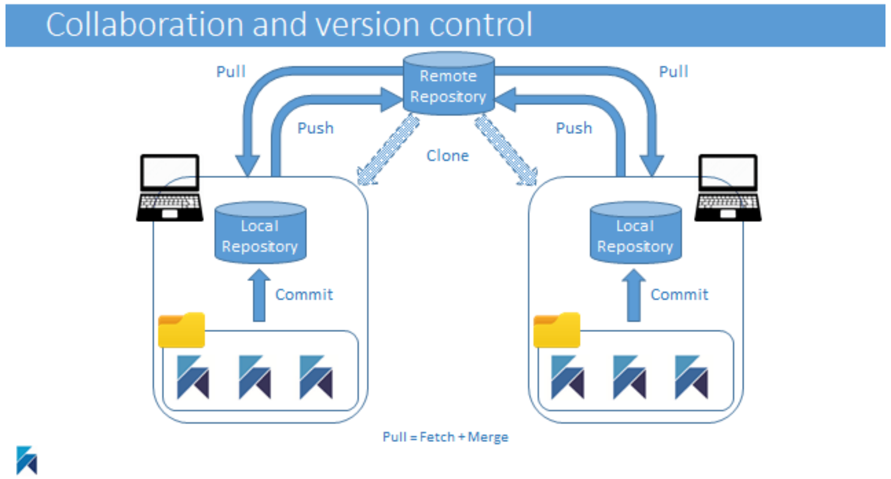
- **echo "# my-first-repository" >> README.md**: This creates a new README.md file with a simple title inside it.
- **git init:** Initializes a new Git repository in your current directory.
- **git add README.md:** Stages the README.md file for committing.
- **git commit -m "first commit":** Commits the staged file with the message "first commit."
- **git branch -M main:** Renames the current branch to main.
- **git remote add origin <url>:** Links the local repository to a remote GitHub repository.
- **git push -u origin main:** Pushes the code from the local main branch to the remote main branch and sets up tracking between the two.

Use case: This method is useful when you're starting from scratch and want to create a new Git repository, initialize it, and then push it to GitHub.

Explanation:

- **git remote add origin <url>:** Links the local repository to the remote GitHub repository.
- **git branch -M main:** Renames the branch to main (if it hasn't been done already).
- **git push -u origin main:** Pushes the local branch to the remote repository and sets up the tracking.

Use case: This method assumes that you've already initialized a Git repository, committed your changes, and just want to push an existing repository to GitHub.

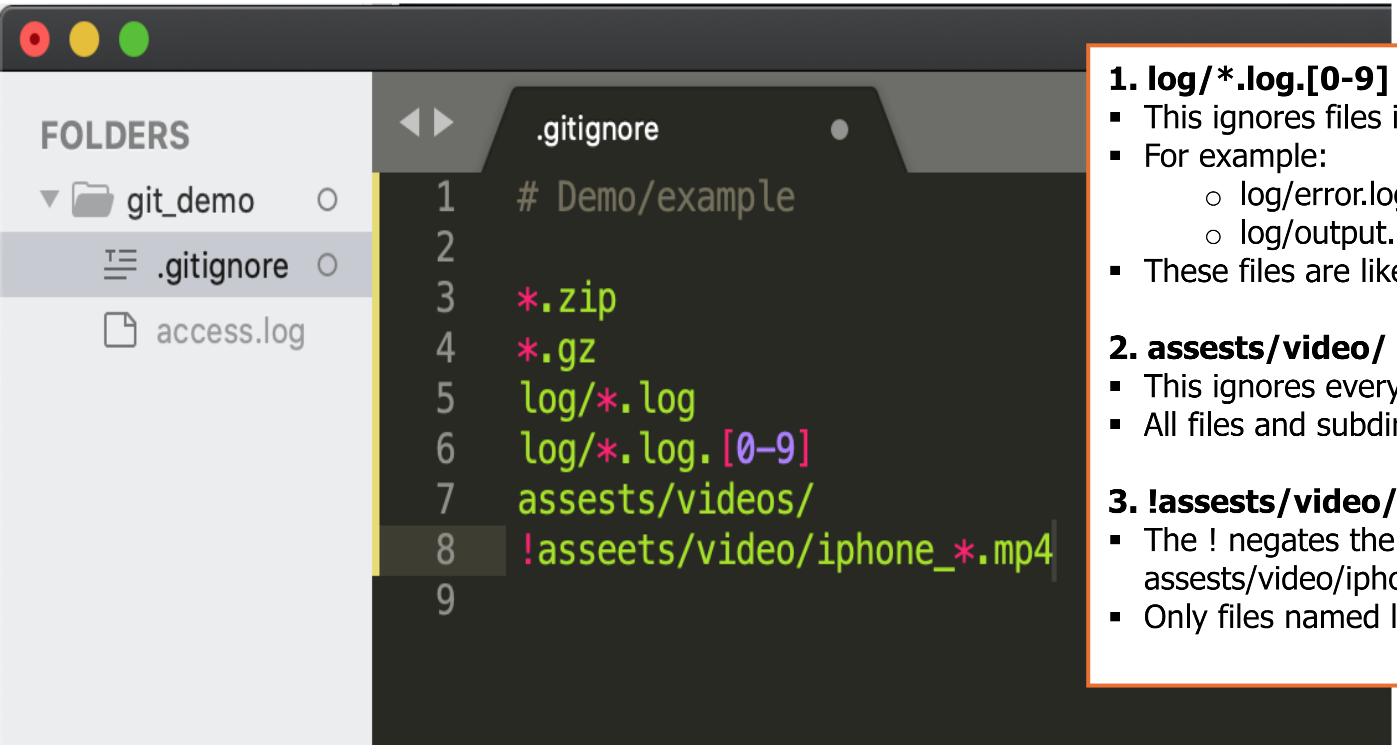


Why ".gitignore"?

The ***.gitignore*** file is essential because it tells Git which files or directories to ignore in a project.

This helps to:

- **Keep Repositories Clean**: Exclude unnecessary files (e.g., temporary files, build artifacts) from being tracked by Git.
- **Protect Sensitive Information**: Prevent sensitive data (e.g., API keys, passwords) from being accidentally committed.
- **Optimize Collaboration**: Ensure that only relevant files are shared among team members, reducing clutter and potential conflicts.
- **Maintain Efficient Version Control**: Focus on tracking and managing only the files that are important to the project.



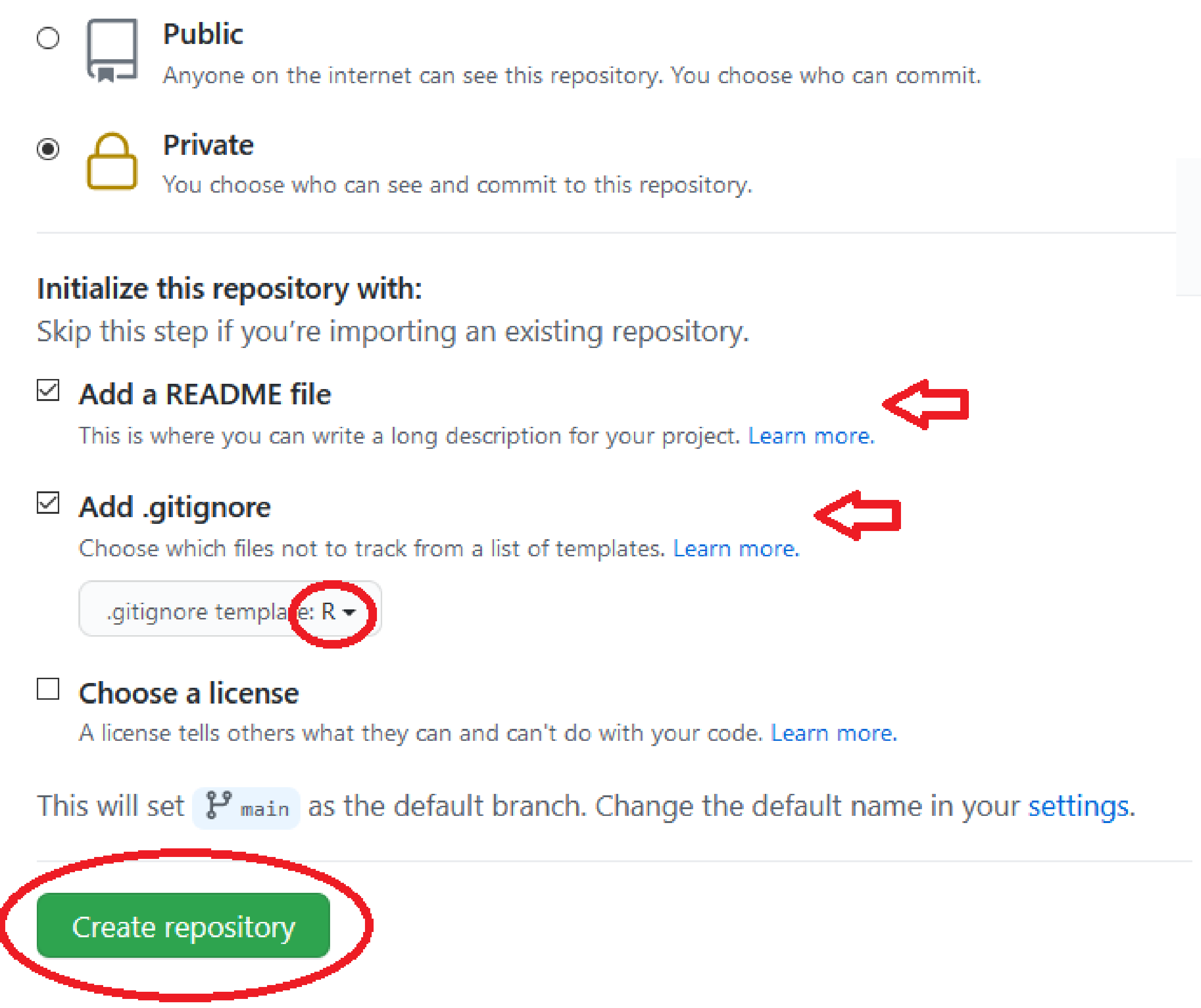
```
# Demo/example

*.zip
*.gz
log/*.log
log/*.log.[0-9]
assests/videos/
!asseets/video/iphone_*.mp4

```

1. log/*.log.[0-9]
- This ignores files in the log/ directory that have the format *.log.[0-9].
- For example:
  - log/error.log.1
  - log/output.log.2
- These files are likely log rotation files (e.g., created when logs are rotated to manage size).

2. assests/video/
- This ignores everything inside the assests/video/ directory.
- All files and subdirectories within assests/video/ will be excluded from Git.

3. !assests/video/iphone_*.mp4
- The ! negates the ignore rule, meaning it will **include** files matching the pattern assests/video/iphone_*.mp4, even though the assests/video/ directory is ignored.
- Only files named like iphone_video.mp4, iphone_clip.mp4, etc., will be tracked by Git.

How to create a git repository?



○ **Public**
Anyone on the internet can see this repository. You choose who can commit.

◉ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☑ **Add a README file**
This is where you can write a long description for your project. Learn more.

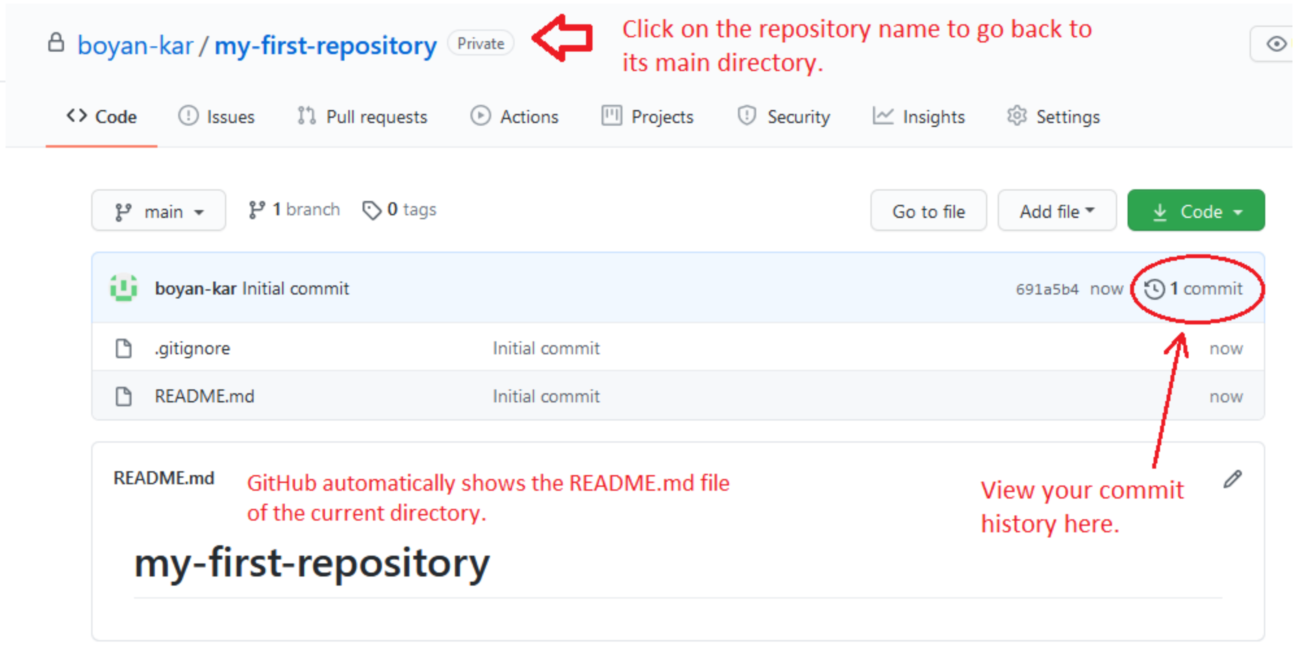
☑ **Add .gitignore**
Choose which files not to track from a list of templates. Learn more.

.gitignore template: R

☐ **Choose a license**
A license tells others what they can and can't do with your code. Learn more.

This will set `main` as the default branch. Change the default name in your settings.

Here is how the repository should look:



boyan-kar / my-first-repository Private

Click on the repository name to go back to its main directory.

boyan-kar Initial commit 691a5b4 now 1 commit

.gitignore	Initial commit	now
README.md	Initial commit	now

README.md

GitHub automatically shows the README.md file of the current directory.

View your commit history here.

my-first-repository

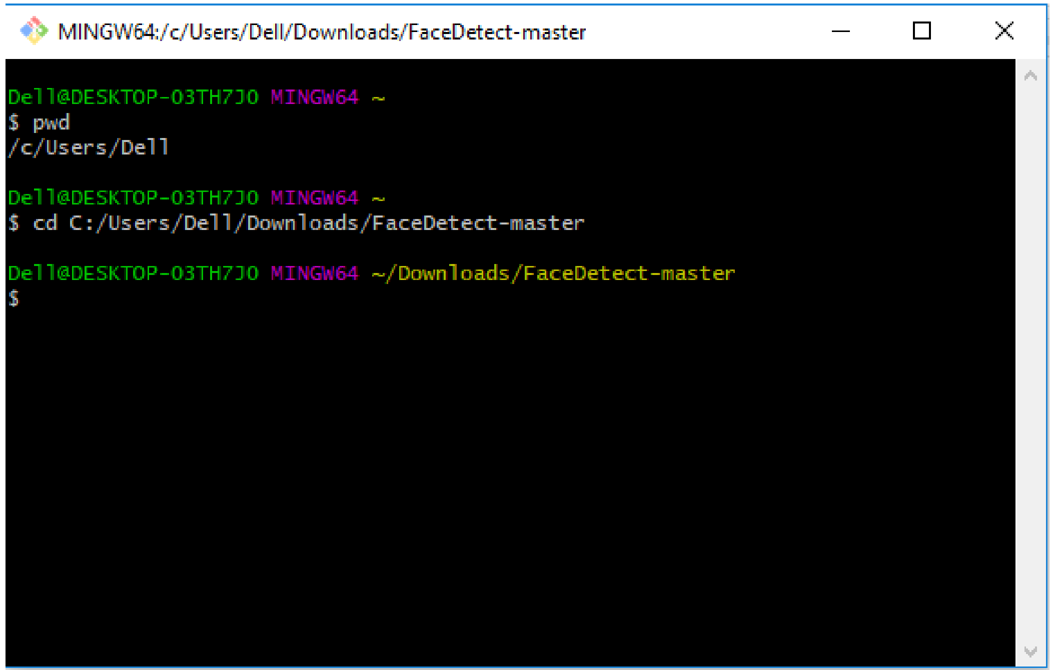
GIT Push and Pull Tutorial

Note: Create a new repository like the previous slide as the 1st step

2. Open your Git Bash

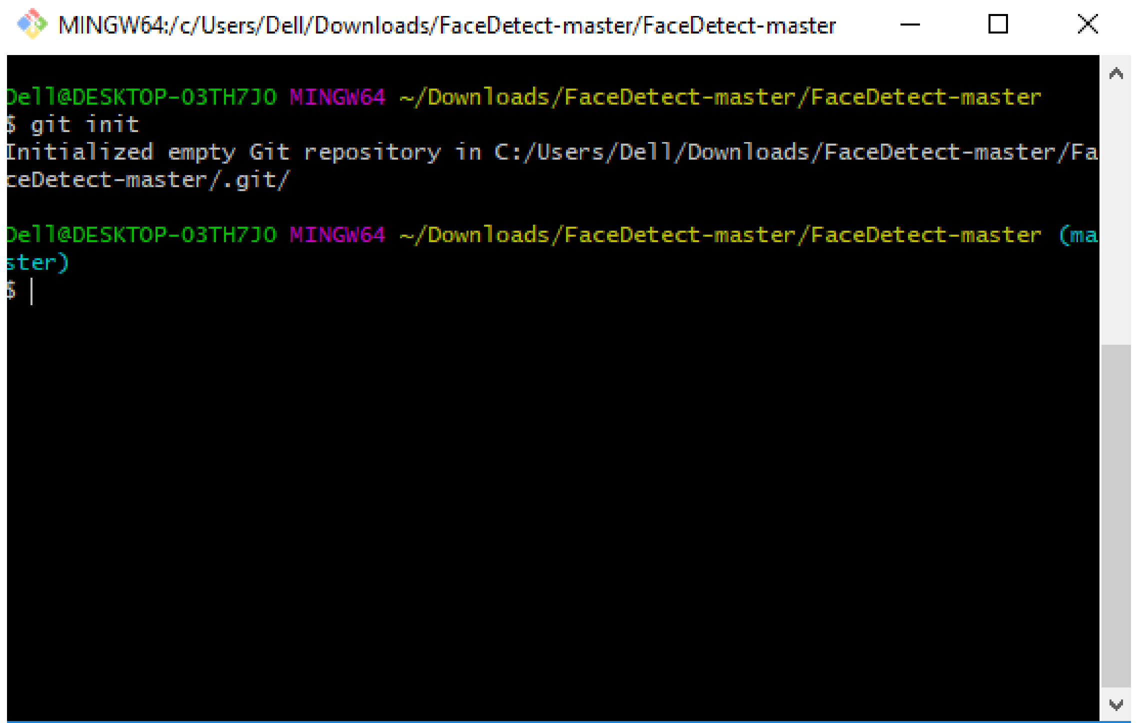
- Git Bash can be downloaded in here, and it is a shell used to interface with the operating system which follows the UNIX command.

3. Create your local project in your desktop directed towards a current working directory



```
MINGW64:/c/Users/Dell/Downloads/FaceDetect-master

Dell@DESKTOP-03TH7J0 MINGW64 ~
$ pwd
/c/Users/Dell

Dell@DESKTOP-03TH7J0 MINGW64 ~
$ cd C:/Users/Dell/Downloads/FaceDetect-master

Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master
$
```

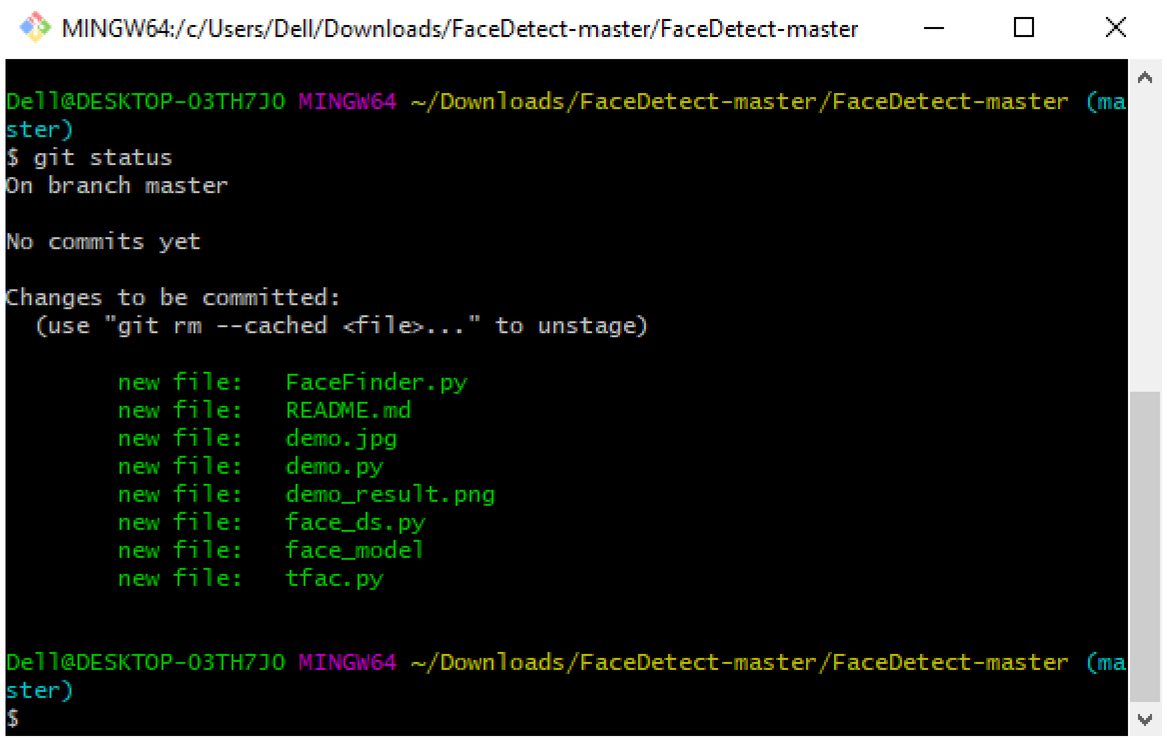
4. Initialize the git repository



```
MINGW64:/c/Users/Dell/Downloads/FaceDetect-master/FaceDetect-master

Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master
$ git init
Initialized empty Git repository in C:/Users/Dell/Downloads/FaceDetect-master/Fa
ceDetect-master/.git/

Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (ma
ster)
$ |
```

GIT Push and Pull Tutorial

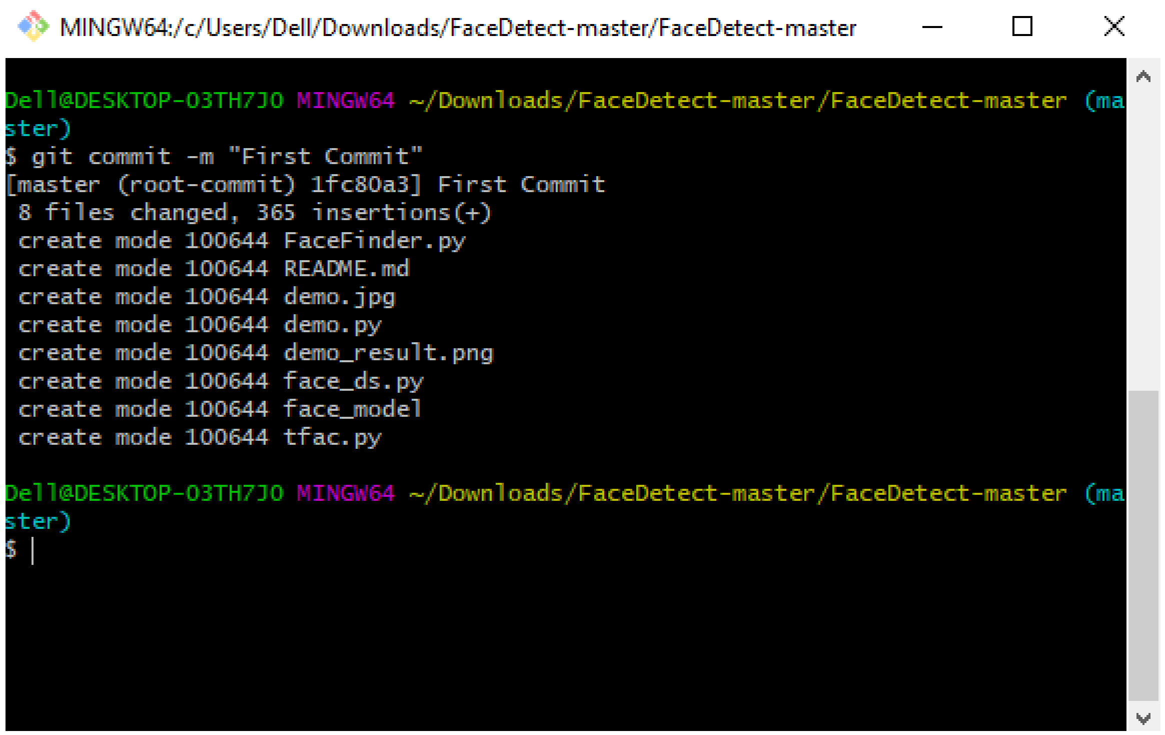
5. Add the file to the new local repository



MINGW64:/c/Users/Dell/Downloads/FaceDetect-master/FaceDetect-master

```
Dell@DESKTOP-03TH7JO MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (ma
ster)
$ git status
On branch master

No commits yet

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

        new file:   FaceFinder.py
        new file:   README.md
        new file:   demo.jpg
        new file:   demo.py
        new file:   demo_result.png
        new file:   face_ds.py
        new file:   face_model
        new file:   tfac.py


Dell@DESKTOP-03TH7JO MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (ma
ster)
$
```

6. Commit the files staged in your local repository by writing a commit message



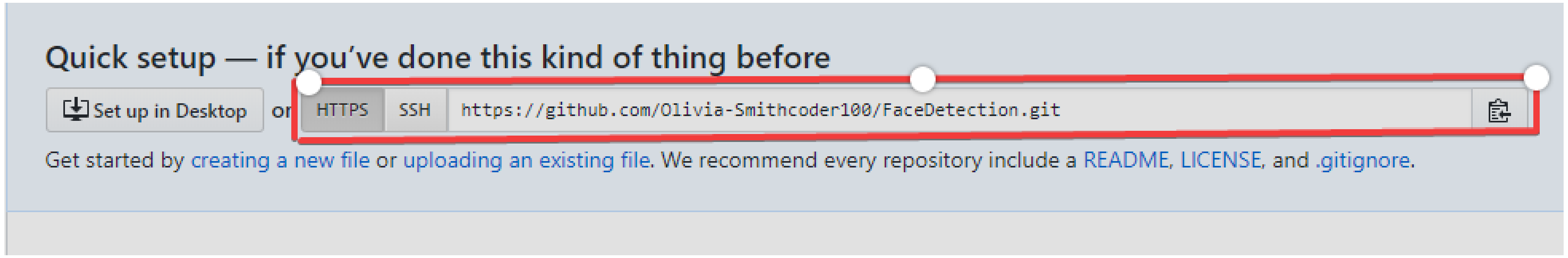
MINGW64:/c/Users/Dell/Downloads/FaceDetect-master/FaceDetect-master

```
Dell@DESKTOP-03TH7JO MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (ma
ster)
$ git commit -m "First Commit"
[master (root-commit) 1fc80a3] First Commit
 8 files changed, 365 insertions(+)
 create mode 100644 FaceFinder.py
 create mode 100644 README.md
 create mode 100644 demo.jpg
 create mode 100644 demo.py
 create mode 100644 demo_result.png
 create mode 100644 face_ds.py
 create mode 100644 face_model
 create mode 100644 tfac.py

Dell@DESKTOP-03TH7JO MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (ma
ster)
$ |
```

GIT Push and Pull Tutorial

7. Copy your remote repository's URL from GitHub

- The HTTPS or URL is copied from the given GitHub account, which is the place of the remote repository.


Quick setup — if you've done this kind of thing before

https://github.com/Olivia-Smithcoder100/FaceDetection.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

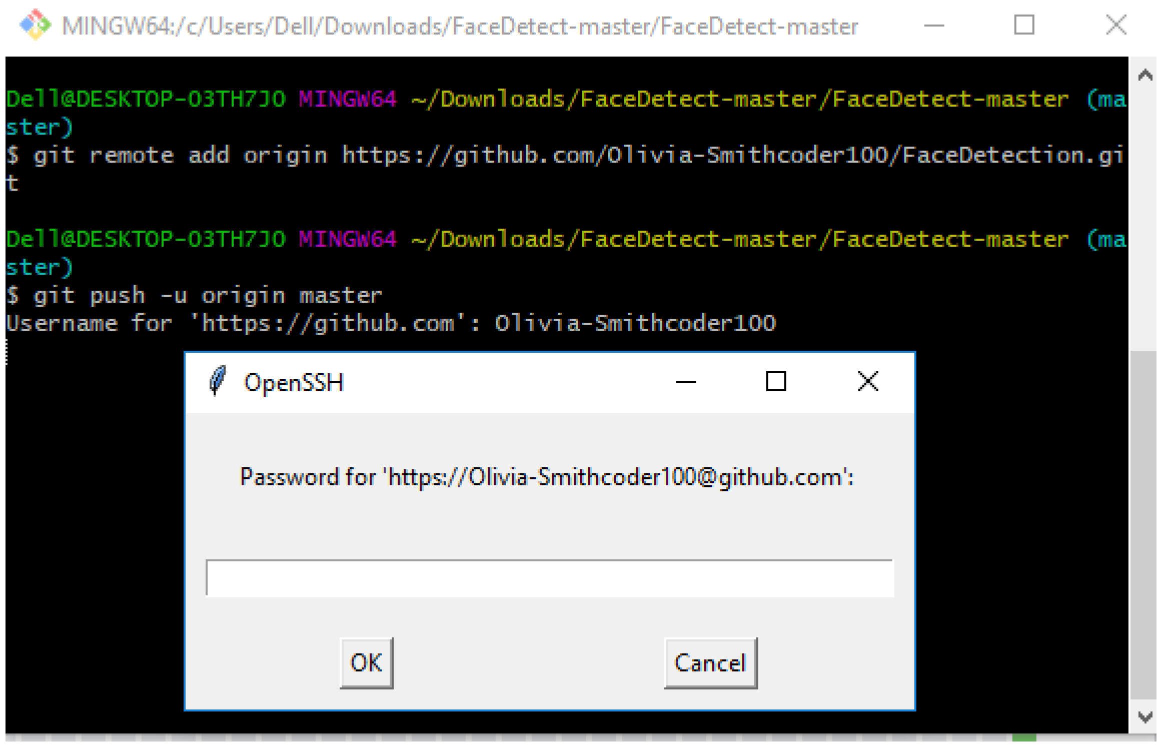
8. Add the URL copied, which is your remote repository to where your local content from your repository is pushed

```
git remote add origin 'your_url_name'
```

GIT Push and Pull Tutorial

9. Push the code in your local repository to GitHub

- *git push -u origin master* is used for pushing local content to GitHub.
- In the code, the origin is your default remote repository name and '-u' flag is upstream, which is equivalent to '-***set-upstream***.' and the master is the branch, name.upstream is the repository that we have cloned the project.
- Fill in your GitHub username and password.



```
MINGW64:/c/Users/Dell/Downloads/FaceDetect-master/FaceDetect-master

Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (ma
ster)
$ git remote add origin https://github.com/Olivia-Smithcoder100/FaceDetection.gi
t

Dell@DESKTOP-03TH7J0 MINGW64 ~/Downloads/FaceDetect-master/FaceDetect-master (ma
ster)
$ git push -u origin master
Username for 'https://github.com': Olivia-Smithcoder100
```

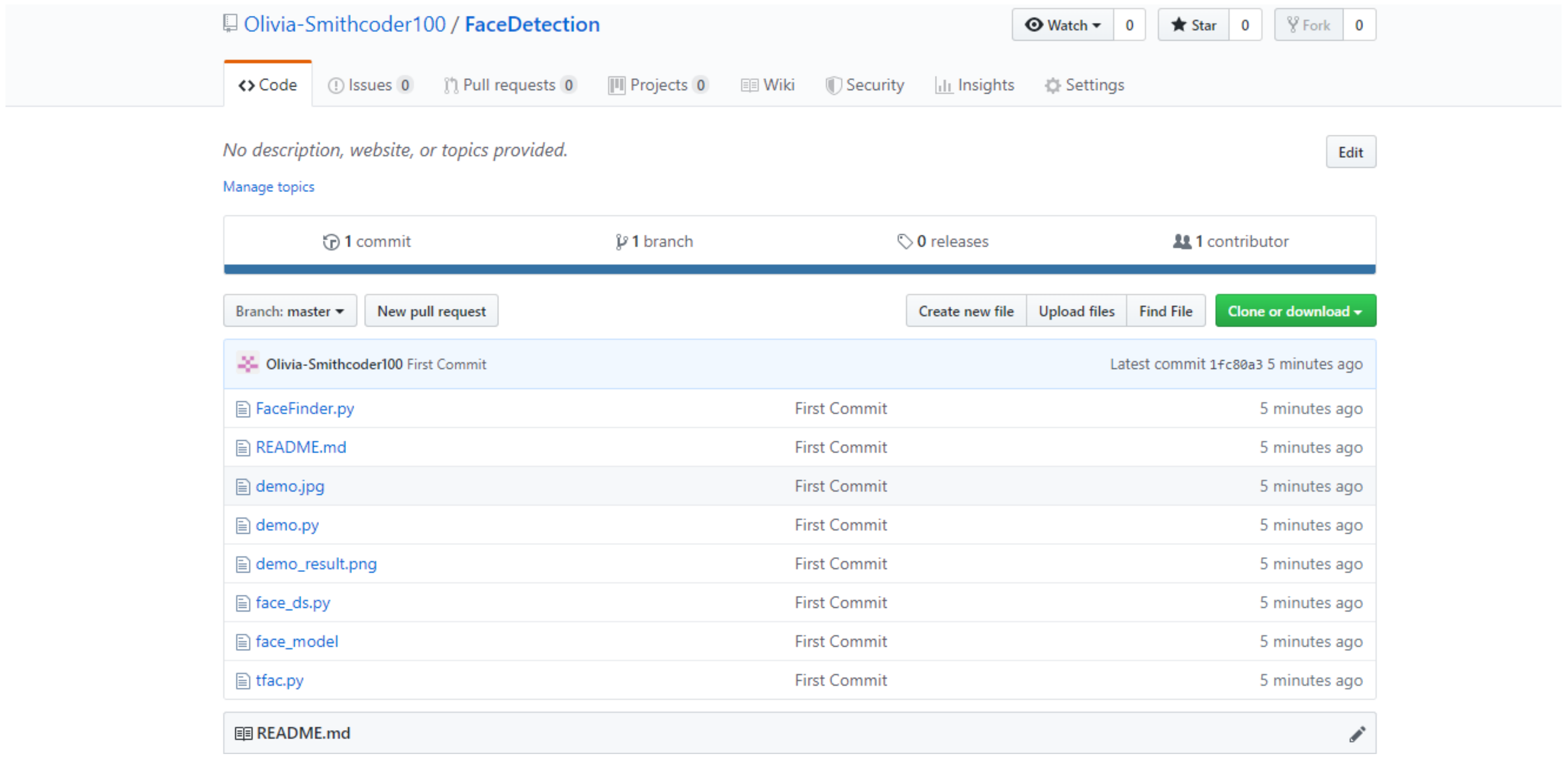
OpenSSH

Password for 'https://Olivia-Smithcoder100@github.com':

Git Push and Pull Tutorial

10. View your files in your repository hosted on GitHub

- *You can finally see the file hosted on GitHub.*



Olivia-Smithcoder100 / FaceDetection
Watch 0 | Star 0 | Fork 0
Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Security | Insights | Settings
Edit
Manage topics
Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

No description, website, or topics provided.

1 commit | 1 branch | 0 releases | 1 contributor

Olivia-Smithcoder100 First Commit — Latest commit 1fc80a3 5 minutes ago

FaceFinder.py	First Commit	5 minutes ago
README.md	First Commit	5 minutes ago
demo.jpg	First Commit	5 minutes ago
demo.py	First Commit	5 minutes ago
demo_result.png	First Commit	5 minutes ago
face_ds.py	First Commit	5 minutes ago
face_model	First Commit	5 minutes ago
tfac.py	First Commit	5 minutes ago

README.md

Challenge Yourself!

Elevate Your GitHub Expertise: Master Branching, Collaboration, and Conflict Resolution

Create Branches:

Learn how to create and manage branches to work on different features or fixes simultaneously.

Manage Team Projects:

Understand how to set up and manage team projects for collaborative development.

Form Groups:

Learn how to organize and collaborate within groups on GitHub.

Resolve Conflicts:

Master conflict resolution techniques to handle and merge conflicting changes smoothly.

Enhance Code Integration: Improve your skills in integrating code from various sources to maintain a cohesive and high-quality project.

Career Opportunities with GitHub Skills: Unlocking Roles in Development, Data, and More

Software Developer: GitHub is crucial for version control and collaboration in software development. Developers use GitHub to manage code, track changes, and collaborate with teams.

DevOps Engineer: DevOps roles involve managing development, testing, and deployment pipelines. GitHub plays a central role in source control, continuous integration (CI), and deployment automation.

Open Source Contributor: Many companies and projects operate in open source, where GitHub is widely used for collaboration. Contributing to open-source projects can be a great way to build your portfolio and network.

Data Scientist/Analyst: GitHub is used to store code, data, and models in data science projects. It's also valuable for collaborating with other data professionals on analysis projects.

Full-Stack Developer: Both frontend and backend developers use GitHub to manage their codebases, ensuring the project is well-organized and contributions from different team members are easily tracked.

Machine Learning Engineer: Version control is critical when developing machine learning models, and GitHub helps manage model code, experiments, and collaborations.

Project Manager (Tech): GitHub can be used for project management, allowing you to track tasks, bugs, and feature requests using GitHub issues and project boards.

Technical Writer: Writing documentation for open-source or internal projects hosted on GitHub is a valuable skill. Documentation helps other developers understand how to use and contribute to projects.

Cloud Engineer: GitHub integrates with cloud platforms like AWS, Azure, and Google Cloud for managing infrastructure as code and deploying applications.

Quality Assurance (QA) Engineer: QA engineers use GitHub to manage testing scripts, track bugs, and collaborate on test automation.

Conclusion

Mastering GitHub is a vital step in building strong version control practices. By learning to manage project versions, track changes, and streamline workflows, you've gained **essential skills for organizing and maintaining any project efficiently**.

Creating and maintaining a professional GitHub profile is equally important. It serves as a **portfolio to showcase your work, demonstrate your expertise, and build your credibility** in academic, professional, or open-source communities.

GitHub empowers seamless collaboration and effective project management, fostering teamwork and improving productivity in group settings.

These **skills and practices provide a solid foundation for your academic and professional growth**, equipping you to tackle future challenges with confidence and contribute meaningfully to any team or project.

Thank you!

Stay Connected!