MACHINE LEARNING OPERATIONS

WEEK 1

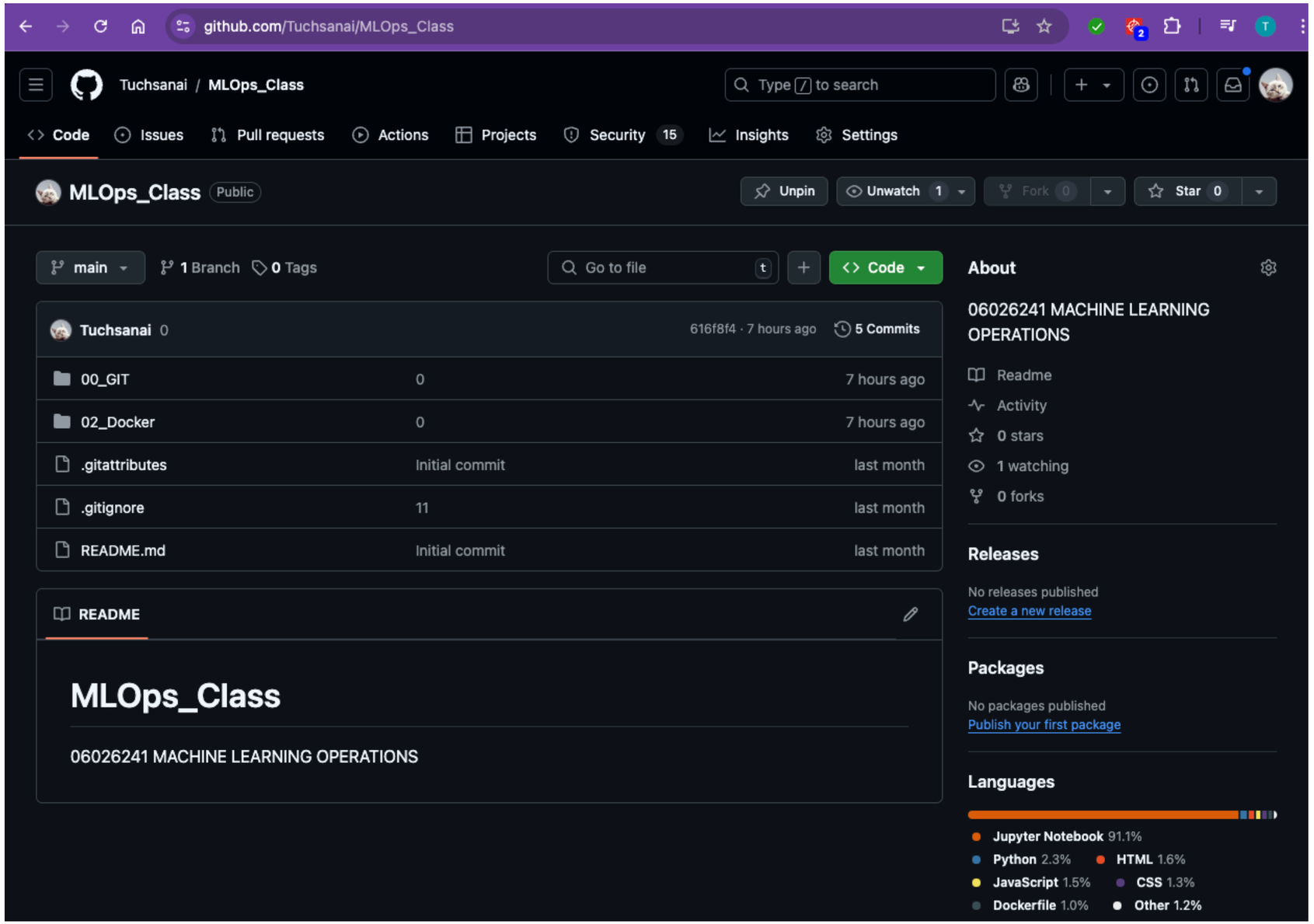
UNLIMITED

Presented by **Asst. Prof. Dr. Tuchsanai Ploysuwan**

MACHINE LEARNING OPERATIONS

Document

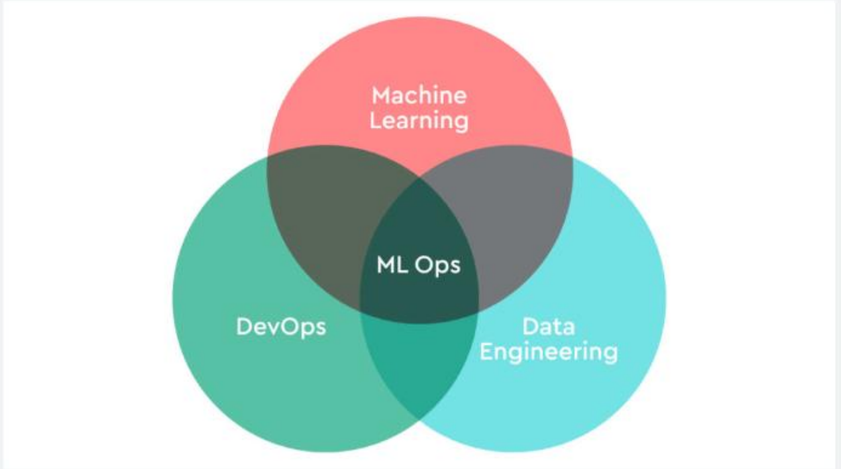
Group line


github.com/Tuchsanai/MLOps_Class
Tuchsanai / MLOps_Class
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security 15
Insights
Settings
MLOps_Class
Public
Unpin
Unwatch 1
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
t
Code
Tuchsanai 0
616f8f4 · 7 hours ago
5 Commits
00_GIT 0 7 hours ago
02_Docker 0 7 hours ago
.gitattributes Initial commit last month
.gitignore 11 last month
README.md Initial commit last month
README
MLOps_Class
06026241 MACHINE LEARNING OPERATIONS
About
06026241 MACHINE LEARNING OPERATIONS
Readme
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
Jupyter Notebook 91.1%
Python 2.3%
HTML 1.6%
JavaScript 1.5%
CSS 1.3%
Dockerfile 1.0%
Other 1.2%

การให้คะแนน	
Midterm	35
Final	30
Homework and Exercise LAB	15
Mini project	20

MLOps is the practice of deploying machine learning models into production

"MLOps refers to the practice and discipline within machine learning that aims to unify and streamline the machine learning system development (Dev) and machine learning system operation (Ops). It involves collaboration between data scientists, ML engineers, and IT professionals to automate and optimize the end-to-end lifecycle of machine learning applications."



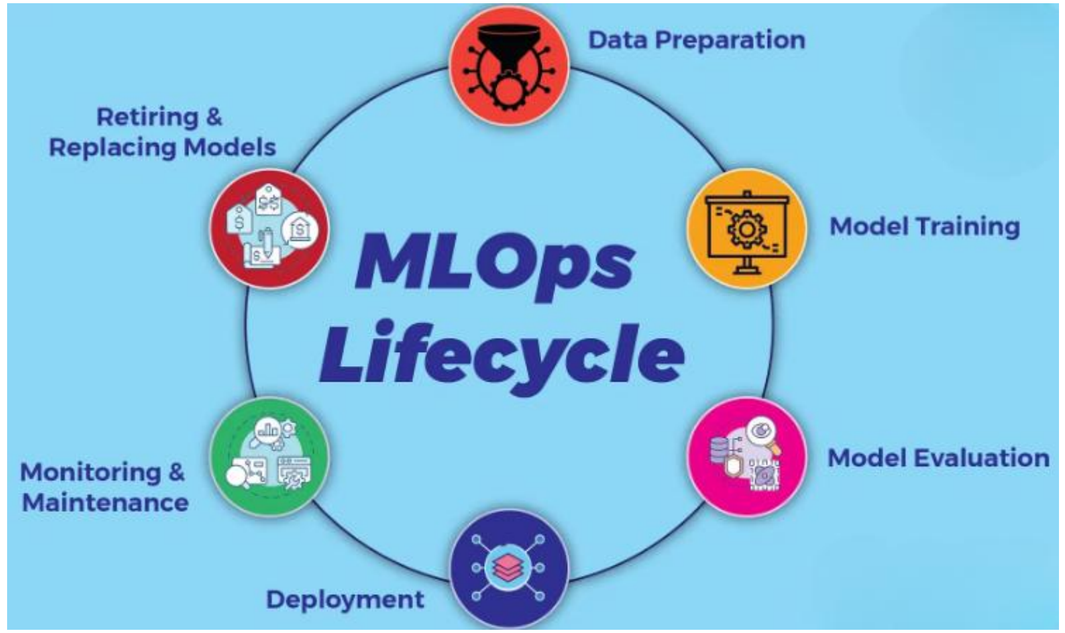
MLOps is to machine learning and data engineering what DevOps is to traditional software development.

💡 แนวคิดหลักของ MLOps

จากแผนภาพ Venn Diagram ด้านบนซ้าย MLOps คือจุดที่เกิดจากการรวมกันของสามสาขาวิชาหลัก:

- **Machine Learning (ML):** เกี่ยวข้องกับการสร้าง, ฝึกฝน, และประเมินผลโมเดลเรียนรู้ของเครื่อง
- **Data Engineering (DE):** เกี่ยวข้องกับการจัดการข้อมูล, การเตรียมข้อมูล, และการสร้างไปป์ไลน์ข้อมูลที่เชื่อถือได้
- **DevOps (Development Operations):** เกี่ยวข้องกับแนวปฏิบัติเพื่อลดวงจรชีวิตของการพัฒนาระบบและการส่งมอบซอฟต์แวร์คุณภาพสูงอย่างต่อเนื่อง (Continuous Integration/Continuous Delivery - CI/CD)

กล่าวโดยสรุป: **MLOps คือสิ่งที่ทำกับ Machine Learning และ Data Engineering คล้ายกับที่ DevOps ทำกับ Software Development ทั่วไป** โดยมีเป้าหมายเพื่อนำโมเดล ML ไปใช้ในการผลิตจริงได้อย่างมีประสิทธิภาพ, เชื่อถือได้, และปรับขนาดได้



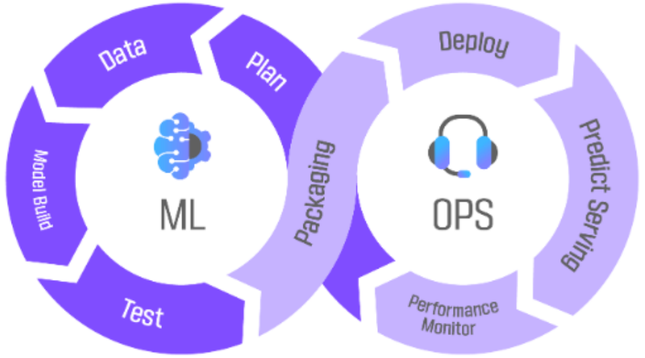
🔄 วงจรชีวิต MLOps (MLOps Lifecycle)

จากแผนภาพวงกลมด้านขวา **MLOps Lifecycle** แสดงให้เห็นขั้นตอนหลักที่ทำงานร่วมกันและเป็นวงจรต่อเนื่อง:

1. **Data Preparation (การเตรียมข้อมูล):** การรวบรวม, ทำความสะอาด, และจัดรูปแบบข้อมูลให้อยู่ในสภาพที่พร้อมสำหรับการฝึกโมเดล
2. **Model Training (การฝึกโมเดล):** การสร้าง, ปรับจูน, และฝึกโมเดล ML โดยใช้ข้อมูลที่เตรียมไว้
3. **Model Evaluation (การประเมินโมเดล):** การวัดประสิทธิภาพของโมเดลที่ฝึกฝนเพื่อตรวจสอบว่าตรงตามเกณฑ์ที่ต้องการหรือไม่
4. **Deployment (การนำไปใช้):** การทำให้โมเดลที่ผ่านการประเมินพร้อมใช้งานจริงในสภาพแวดล้อมการผลิต (Production) เช่น การติดตั้งเป็น API หรือฝังในแอปพลิเคชัน
5. **Monitoring & Maintenance (การเฝ้าระวังและการบำรุงรักษา):** การติดตามประสิทธิภาพของโมเดลที่ทำงานจริง (เช่น ความแม่นยำ, ความคลาดเคลื่อนของข้อมูล หรือ **Data Drift**, ความคลาดเคลื่อนของแนวคิด หรือ **Concept Drift**) และดำเนินการแก้ไขเมื่อจำเป็น
6. **Retiring & Replacing Models (การปลดระวางและเปลี่ยนโมเดล):** การตัดสินใจนำโมเดลเก่าออกและแทนที่ด้วยโมเดลใหม่ที่ดีกว่าที่ผ่านวงจรการฝึกฝนซ้ำ

What is an MLOps platform?

The MLOps platform provides a collaborative environment for software engineers and data scientists. It enables real-time collaboration and iterative data exploration to facilitate experiment tracking, model management, feature engineering, and more.



องค์ประกอบของแพลตฟอร์ม MLOps

จากภาพด้านล่างซ้าย แสดงให้เห็นถึงการผนวกรวมกันของ **ML (Machine Learning) Pipeline** และ **OPS (Operations) Pipeline**:

ส่วน ML:

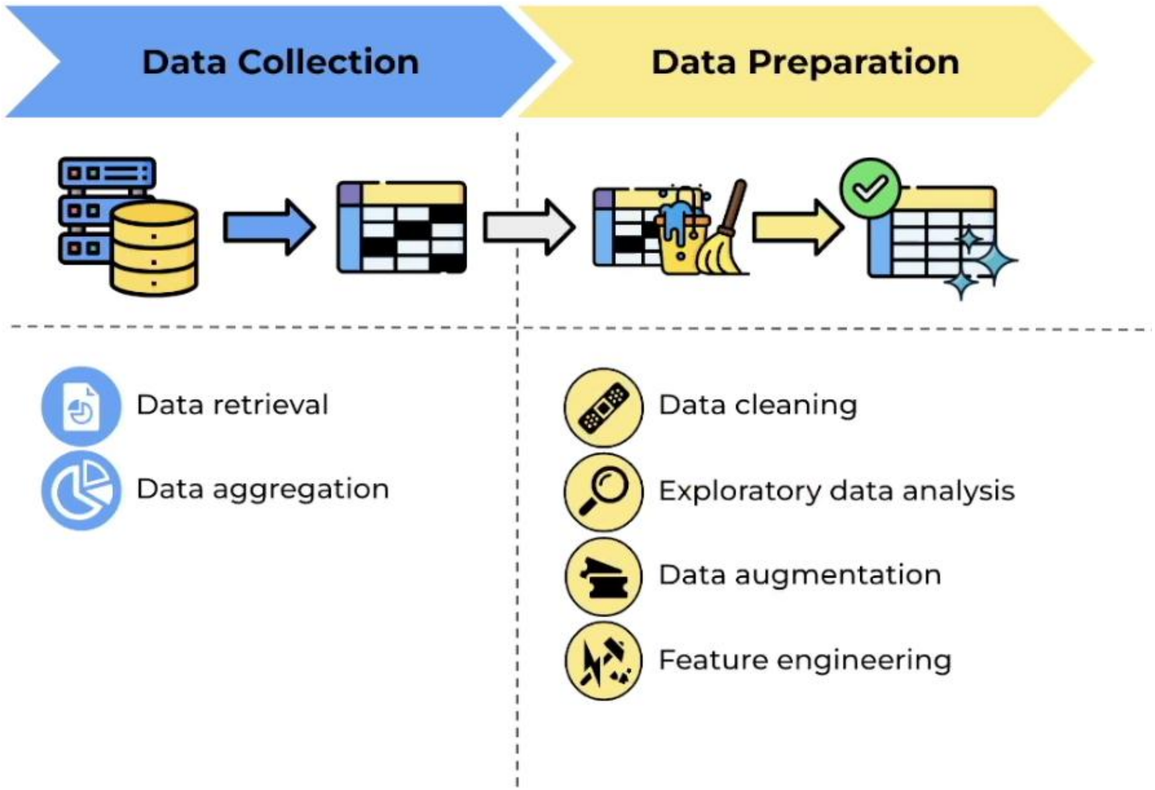
- **Data (ข้อมูล):** เริ่มต้นด้วยข้อมูล
- **Plan (การวางแผน):** การออกแบบการทดลองและกลยุทธ์
- **Test (การทดสอบ):** การทดสอบโมเดลและโค้ด

ส่วน OPS:

- **Packaging (การจัดแพ็คเกจ):** การบรรจุโค้ด, โมเดล, และทรัพยากรที่เกี่ยวข้องทั้งหมดให้อยู่ในรูปแบบที่สามารถนำไปใช้ได้ง่าย (เช่น Containerization)
- **Deploy (การปรับใช้):** การติดตั้งโมเดลและส่วนประกอบที่เกี่ยวข้องในสภาพแวดล้อมจริง
- **Product Serving (การให้บริการผลิตภัณฑ์):** การนำโมเดลที่ติดตั้งไปใช้ตอบสนองคำขอของผู้ใช้หรือระบบอื่น ๆ
- **Performance Monitor (การเฝ้าระวังประสิทธิภาพ):** การติดตามการทำงานของโมเดลในขณะให้บริการจริง (คล้ายกับ Monitoring ใน Lifecycle)

MLOps จึงเป็น **แพลตฟอร์มที่สนับสนุนสภาพแวดล้อมการทำงานร่วมกัน** สำหรับวิศวกรซอฟต์แวร์และนักวิทยาศาสตร์ข้อมูล เพื่อให้การทดลอง, การจัดการโมเดล, และการส่งมอบฟีเจอร์เป็นไปอย่างต่อเนื่องและสามารถติดตามผลได้

ML Product Lifecycle



1. 💾 Data Collection (การรวบรวมข้อมูล)

ขั้นตอนนี้คือการหาและนำข้อมูลดิบมาใช้สำหรับการฝึกฝนโมเดล

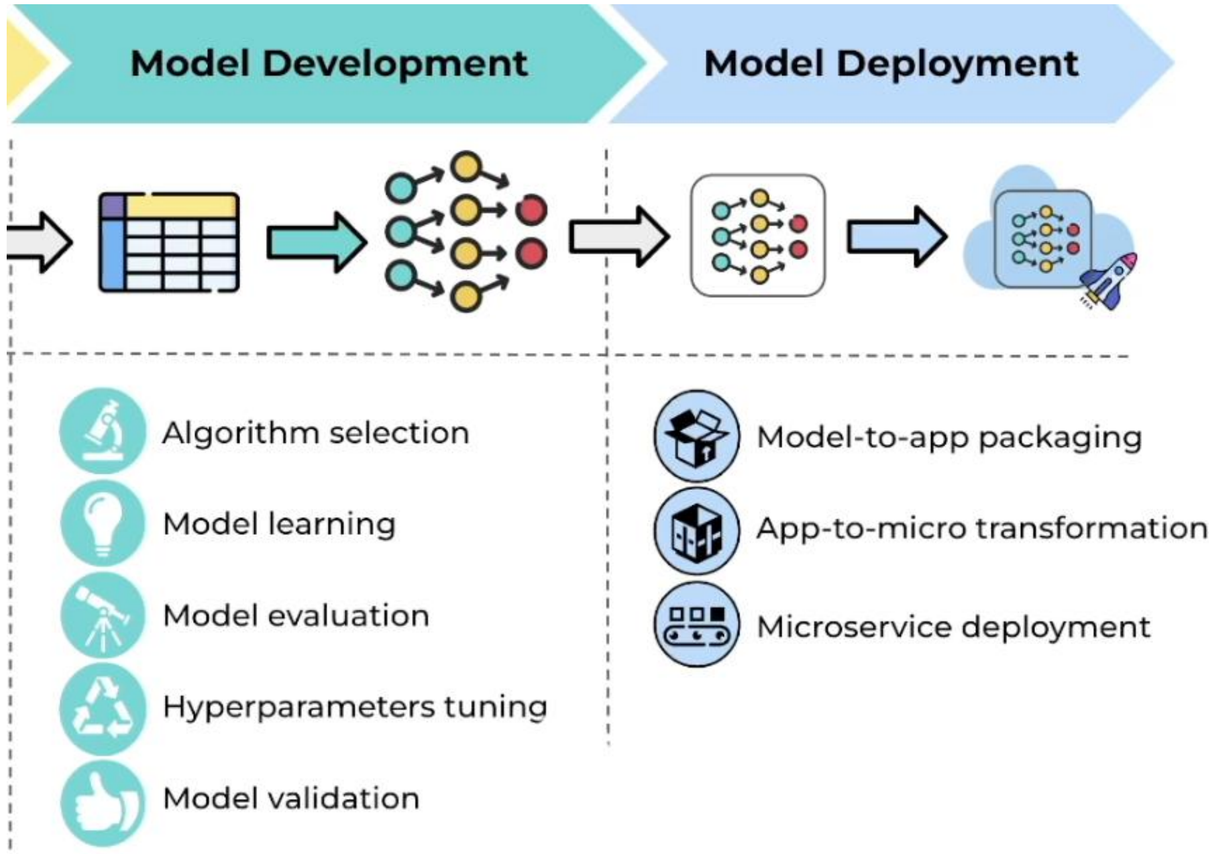
- **Data retrieval (การดึงข้อมูล):** ดึงข้อมูลที่จำเป็นจากแหล่งต่าง ๆ เช่น ฐานข้อมูล, API, หรือไฟล์
- **Data aggregation (การรวมข้อมูล):** นำข้อมูลที่ได้จากหลายแหล่งมารวมกันและจัดระเบียบ

---

2. 🗂️ Data Preparation (การเตรียมข้อมูล)

ขั้นตอนนี้เป็นขั้นตอนที่สำคัญมากในการทำความสะอาด จัดรูปแบบ และแปลงข้อมูลให้อยู่ในสภาพที่พร้อมสำหรับการนำไปใช้ฝึกโมเดล

- **Data cleaning (การทำความสะอาดข้อมูล):** จัดการกับข้อมูลที่ไม่สมบูรณ์, ข้อมูลที่ผิดพลาด, หรือข้อมูลที่ซ้ำซ้อน
- **Exploratory data analysis (การวิเคราะห์ข้อมูลเชิงสำรวจ):** ทำความเข้าใจลักษณะของข้อมูล เช่น การกระจายตัวของข้อมูล, ความสัมพันธ์ระหว่างตัวแปร
- **Data augmentation (การเพิ่มข้อมูล):** สร้างข้อมูลเพิ่มเติมจากข้อมูลที่มีอยู่ (มักใช้กับรูปภาพหรือเสียง) เพื่อเพิ่มขนาดชุดข้อมูล
- **Feature engineering (วิศวกรรมฟีเจอร์):** การสร้างตัวแปร (ฟีเจอร์) ใหม่ ๆ หรือการแปลงฟีเจอร์เดิมเพื่อเพิ่มประสิทธิภาพให้กับโมเดล
- **Feature selection (การเลือกฟีเจอร์):** การเลือกชุดของฟีเจอร์ย่อยที่เหมาะสมที่สุดสำหรับการสร้างโมเดล



3. 🧠 Model Development (การพัฒนาโมเดล)

ขั้นตอนนี้คือการเลือกอัลกอริทึม การฝึกฝนโมเดล และการปรับปรุงประสิทธิภาพของโมเดล

- **Algorithm selection (การเลือกอัลกอริทึม):** เลือกระเบียบวิธี (เช่น Linear Regression, Decision Tree, Neural Networks) ที่เหมาะสมกับประเภทของปัญหาและข้อมูล
- **Model learning (การฝึกฝนโมเดล):** การใช้ชุดข้อมูลที่เตรียมไว้เพื่อฝึกฝนอัลกอริทึมให้เรียนรู้รูปแบบจากข้อมูล
- **Model evaluation (การประเมินโมเดล):** การวัดประสิทธิภาพของโมเดลโดยใช้ชุดข้อมูลทดสอบ (Test Set) ด้วยเมตริกที่เหมาะสม (เช่น Accuracy, Precision, Recall, F1-Score)
- **Hyperparameters tuning (การปรับจูนไฮเปอร์พารามิเตอร์):** การปรับค่าพารามิเตอร์ภายนอกของโมเดล (ที่ไม่ถูกเรียนรู้จากการฝึกฝน) เพื่อให้โมเดลมีประสิทธิภาพสูงสุด
- **Model validation (การตรวจสอบความถูกต้องของโมเดล):** การใช้ชุดข้อมูลการตรวจสอบ (Validation Set) เพื่อยืนยันว่าโมเดลทำงานได้ดีและไม่ได้เกิด Overfitting

---

4. 🚀 Model Deployment (การนำโมเดลไปใช้งาน)

ขั้นตอนนี้คือการนำโมเดลที่พัฒนาเสร็จแล้วไปใช้งานในสภาพแวดล้อมจริง เพื่อให้ผู้ใช้สามารถโต้ตอบกับโมเดลได้

- **Model-to-app packaging (การรวมโมเดลเข้ากับแอปพลิเคชัน):** การจัดแพ็กเกจโมเดลให้อยู่ในรูปแบบที่สามารถเรียกใช้ได้จากแอปพลิเคชันหรือระบบอื่น ๆ
- **App-to-micro transformation (การแปลงแอปพลิเคชันเป็นไมโครเซอร์วิส):** การแยกส่วนฟังก์ชันการทำงานของโมเดลออกมาเป็นบริการย่อย (Microservice)
- **Microservice deployment (การนำไมโครเซอร์วิสไปใช้งาน):** การติดตั้งและเรียกใช้งานบริการย่อยของโมเดลบนเซิร์ฟเวอร์หรือ Cloud Platform

5. 📉 Monitoring & Maintenance (การเฝ้าระวังและการบำรุงรักษา)

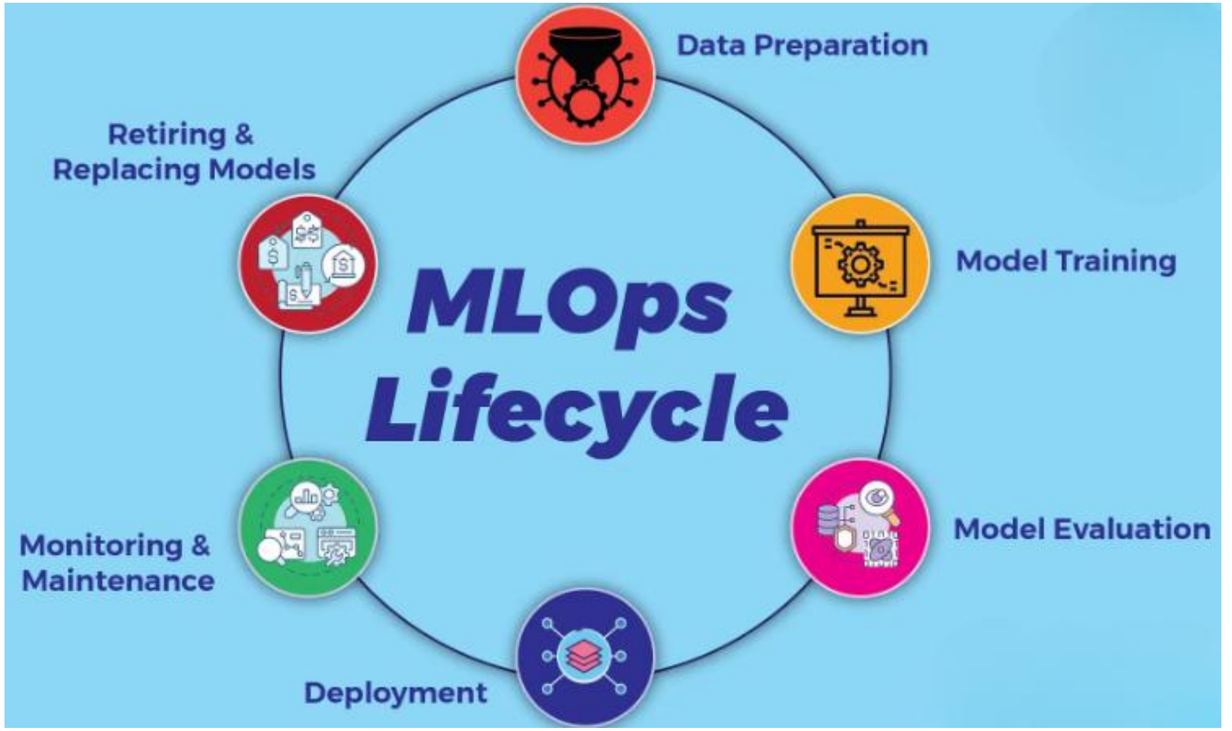
หลังจากโมเดลถูกนำไปใช้งานจริงแล้ว จำเป็นต้องติดตามประสิทธิภาพอย่างต่อเนื่อง

- **วัตถุประสงค์:** ตรวจสอบว่าโมเดลยังคงทำงานได้ถูกต้องและมีประสิทธิภาพตามที่คาดหวัง
- **กิจกรรมหลัก:**
  - **Model Performance Monitoring:** ตรวจสอบ Metrics ทางสถิติและความแม่นยำของโมเดล
  - **Data Drift & Concept Drift Detection:** ตรวจจับความคลาดเคลื่อนของข้อมูลขาเข้า หรือการเปลี่ยนแปลงความสัมพันธ์ระหว่างข้อมูลและผลลัพธ์
  - **System Monitoring:** ตรวจสอบความพร้อมใช้งาน (Uptime), ความหน่วง (Latency), และการใช้ทรัพยากรของบริการ (CPU/RAM)
- **MLOps Focus:** กำหนด **Thresholds** (เกณฑ์) สำหรับ Metrics หากประสิทธิภาพของโมเดลต่ำกว่าเกณฑ์ที่กำหนดไว้ ระบบจะต้องแจ้งเตือน (Alert) และที่สำคัญที่สุดคือ **Trigger** ให้เกิดการฝึกซ้ำ (Retraining) อัตโนมัติ

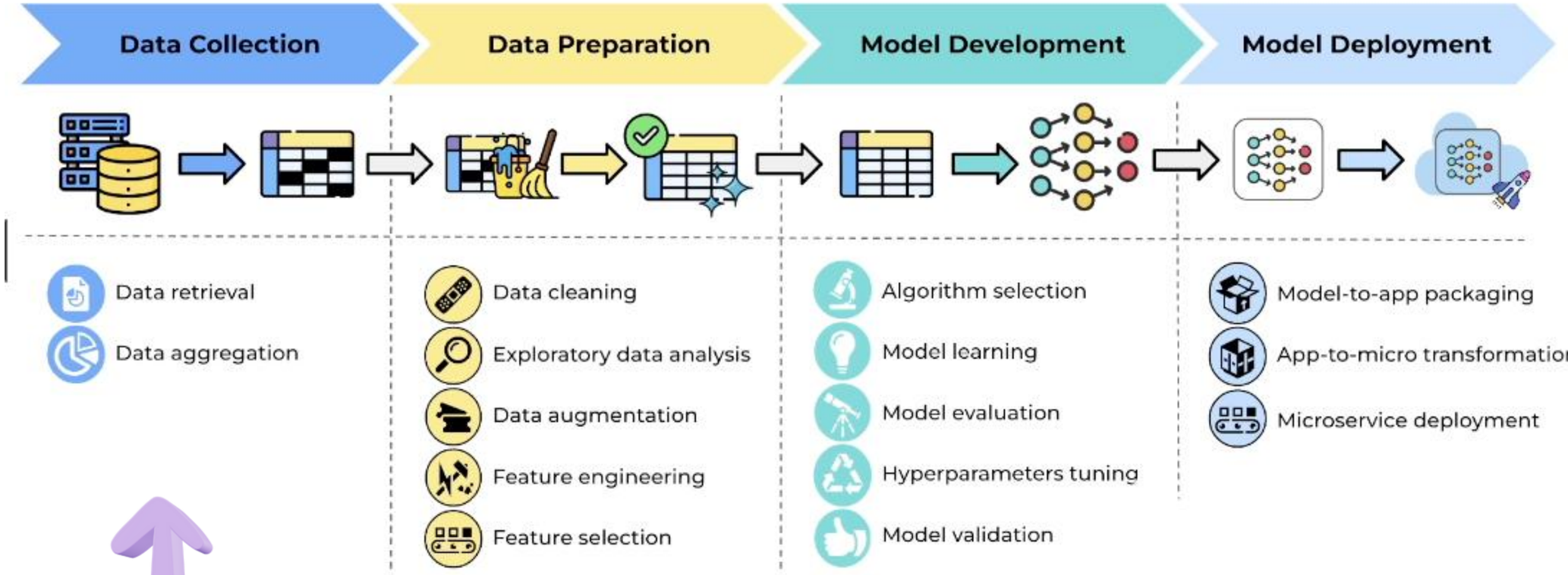
6. ♻️ Retiring & Replacing Models (การปลดระวางและเปลี่ยนโมเดล)

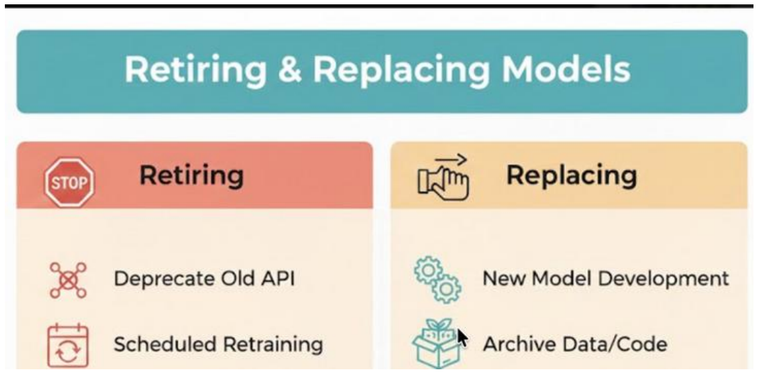
ขั้นตอนนี้เป็นจุดเชื่อมต่อกลับไปสู่จุดเริ่มต้นของวงจร

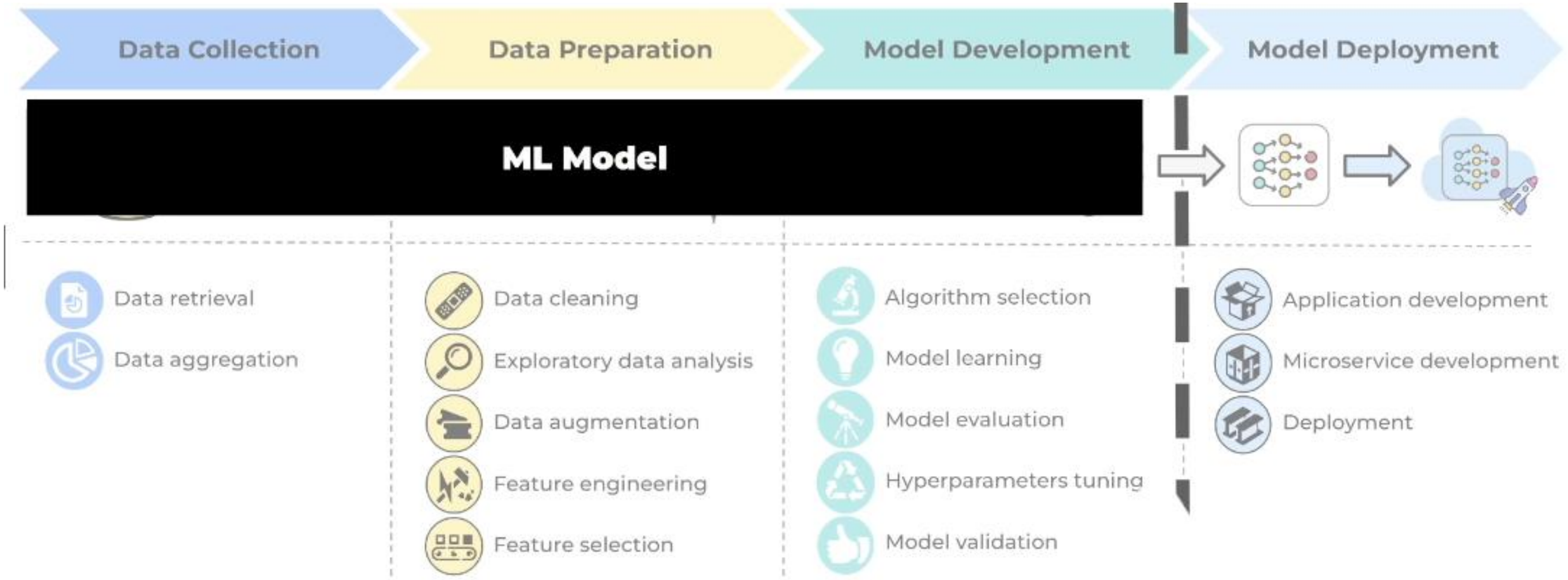
- **วัตถุประสงค์:** จัดการวงจรชีวิตของโมเดลเก่า และนำโมเดลใหม่ที่ฝึกซ้ำมาแทนที่
- **กิจกรรมหลัก:**
  - **Decision to Retrain:** การตัดสินใจฝึกโมเดลใหม่มักเกิดจากการแจ้งเตือนจากขั้นตอน Monitoring
  - **Model Retirement:** การนำโมเดลเวอร์ชันเก่าออกจาก Production อย่างเป็นระเบียบ
  - **Pipeline Triggering:** การรันไปป์ไลน์ทั้งหมด (ตั้งแต่ Data Preparation ใหม่) เพื่อสร้างโมเดลเวอร์ชันใหม่ที่เรียนรู้จากข้อมูลล่าสุด
- **MLOps Focus:** ความสามารถในการทำซ้ำ (Repeatability) และความเป็นอัตโนมัติ (Automation) คือกุญแจสำคัญ เพราะการอัปเดตโมเดลควรเกิดขึ้นบ่อยครั้งและเป็นไปตามกลไกที่กำหนด



Machine Learning Product Lifecycle



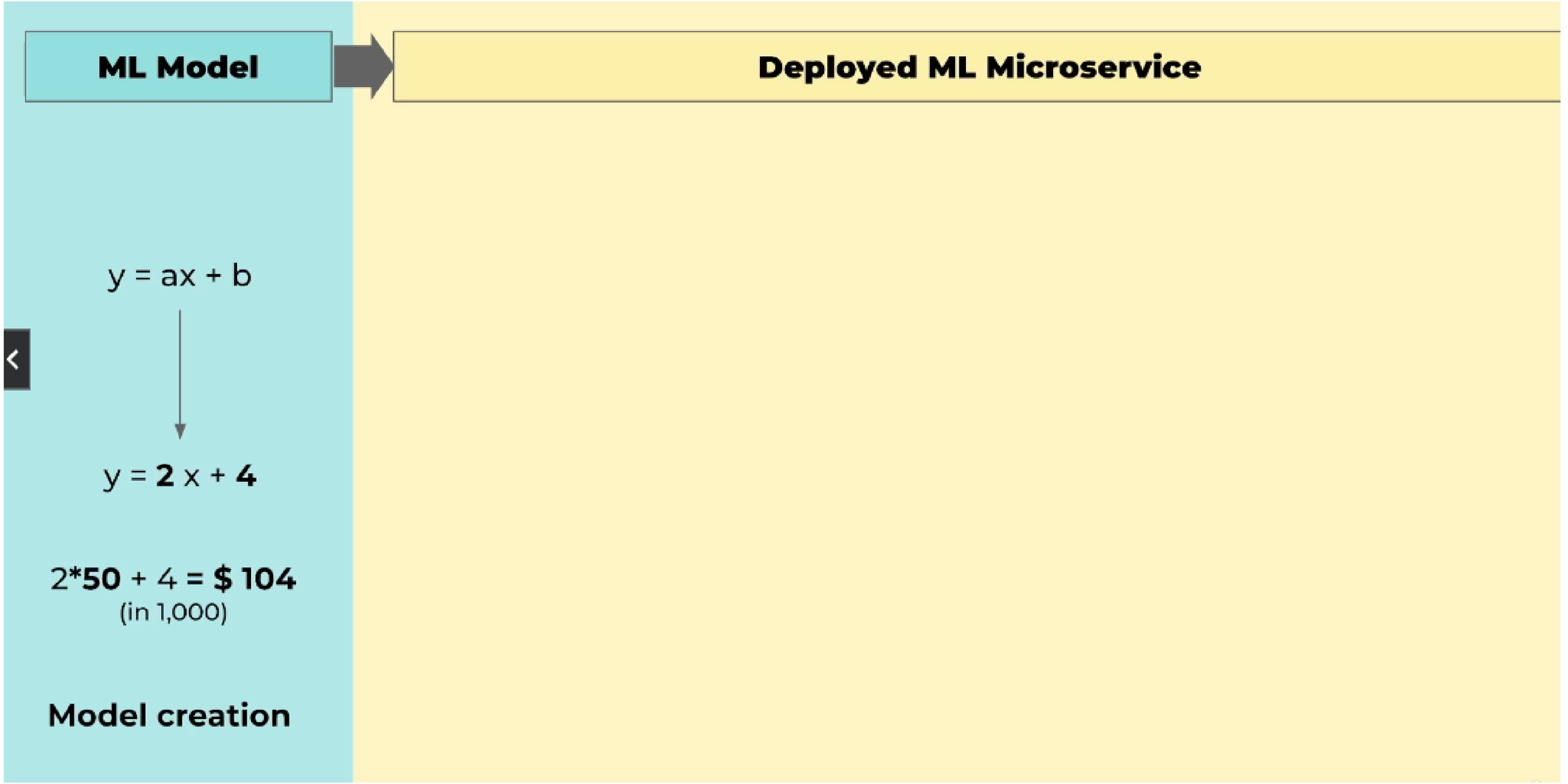


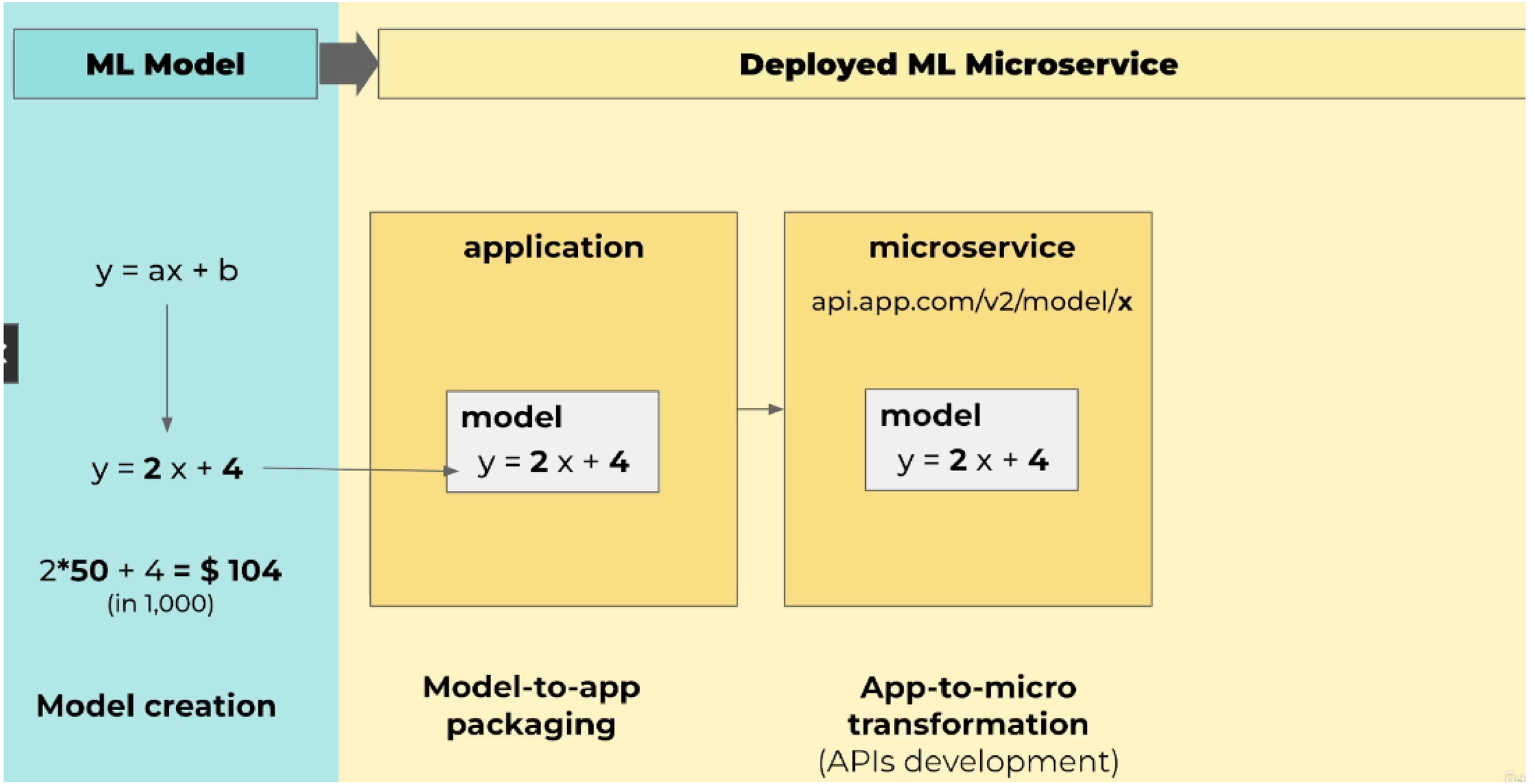

Data Collection
Data Preparation
Model Development
Model Deployment
ML Model
Data retrieval
Data aggregation
Data cleaning
Exploratory data analysis
Data augmentation
Feature engineering
Feature selection
Algorithm selection
Model learning
Model evaluation
Hyperparameters tuning
Model validation
Application development
Microservice development
Deployment

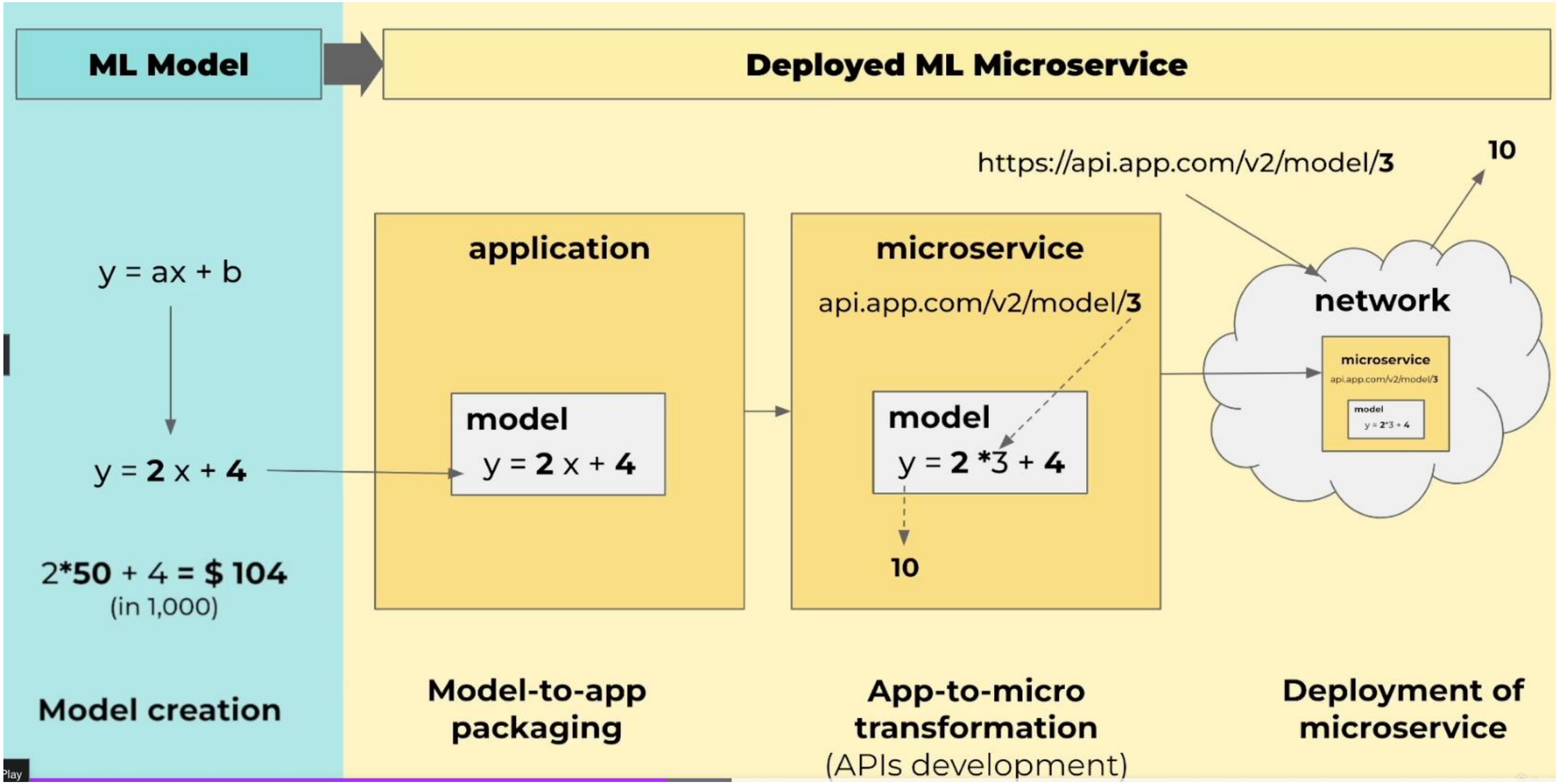
Data Collection → Data Preparation → Model Development → Model Deployment

ML Model

Deployed Microservice


ML Model
Deployed ML Microservice
y = ax + b
y = 2 x + 4
2*50 + 4 = $ 104
(in 1,000)
Model creation

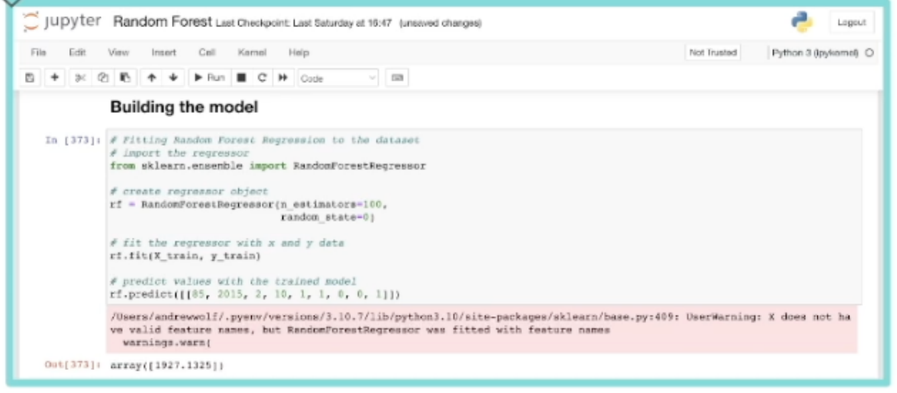

ML Model
Deployed ML Microservice
y = ax + b
y = 2 x + 4
2*50 + 4 = $ 104
(in 1,000)
Model creation
application
model
y = 2 x + 4
Model-to-app packaging
microservice
api.app.com/v2/model/x
model
y = 2 x + 4
App-to-micro transformation
(APIs development)


ML Model
Deployed ML Microservice
https://api.app.com/v2/model/3
10
y = ax + b
y = 2 x + 4
2*50 + 4 = $ 104
(in 1,000)
application
model
y = 2 x + 4
microservice
api.app.com/v2/model/3
model
y = 2 *3 + 4
10
network
microservice
api.app.com/v2/model/3
model
y = 2*3 + 4
Model creation
Model-to-app packaging
App-to-micro transformation
(APIs development)
Deployment of microservice

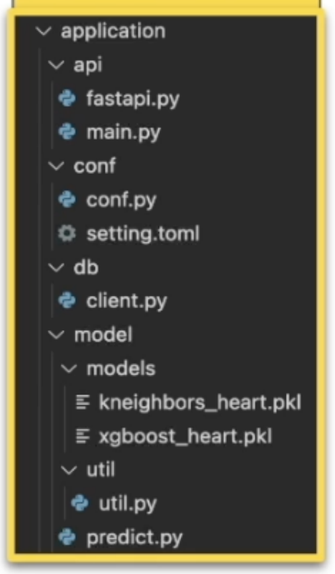
ML Model

Deployed ML Microservice

1 Model creation in Jupyter Notebook



2 Model-to-app packaging



3 App-to-micro transformation via APIs

```
api.app.com/v2/model/84,23,53,43/prediction
```

4 Deploying ML microservice

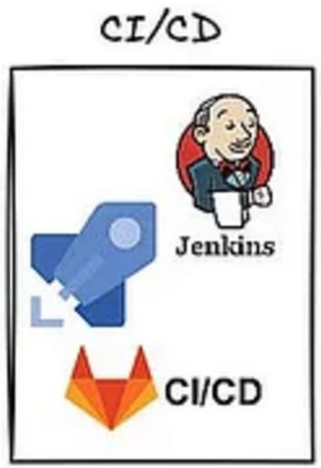
```
version: '3.10'
services:
    app:
        build: app/.
        restart: always
        environment:
        - DOCKER_HOME=/home
        - LOAD_EX=n
        - EXECUTOR=Local
        ports:
        - "5050:5050"
        volumes:
        - ./app:/home
        networks:
        - main_net
```

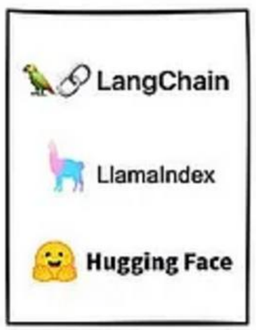
Popular tools for MLOps Process

Version Control

GitHub

MLOPS





LLMOps specifics

Week	Topic	Description
1	Introduction to Git	Learn the basics of Git, including setup, repositories, and basic commands.
2	Branching and Collaboration	Understand branching, merge conflicts, and pull requests for collaborative workflows.
3	Advanced Git	Explore advanced Git features like stashing, rebasing, undoing changes, and managing `.gitignore`.
4	Git Workflows in MLOps	Implement Git workflows such as feature branching and release management.
5	Introduction to Docker	Learn Docker basics, including containerization and writing Dockerfiles.
6	Docker Compose and Networking	Build multi-container applications and understand Docker networking concepts.
7	Docker in MLOps	Containerize machine learning workflows for training and inference.
8	Docker Orchestration	Manage and orchestrate containerized ML workflows effectively.
9	Introduction to MLflow	Learn the components of MLflow and log metrics and parameters in experiments.
10	MLflow Tracking	Track experiments, log custom metrics, and compare ML models.
11	MLflow Models	Package and deploy ML models using MLflow.
12	MLflow Model Registry	Manage the model lifecycle and use MLflow Registry for production workflows.
13	CI/CD in MLOps	Automate ML workflows with Git and integrate MLflow.
14	Monitoring and Scaling	Monitor deployed models, detect data drift, and ensure scalable deployments.
15	MLOps Frameworks	Explore advanced MLOps tools and integrate them with MLflow.
16	Mini Project	Apply learned concepts to create a small MLOps project involving Git, Docker, and MLflow.

- 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**

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!

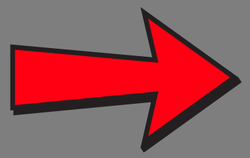
Thank heavens I saved my game! I can just revert!

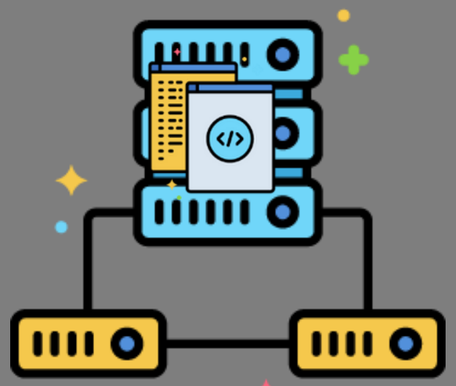
Before Version Control System

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

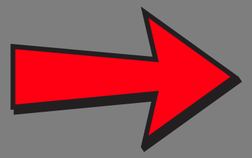
Version 2.0

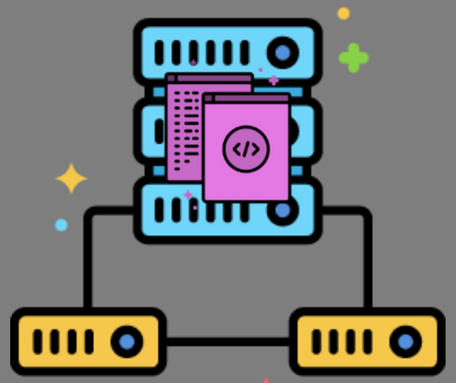
running code version 1.0

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

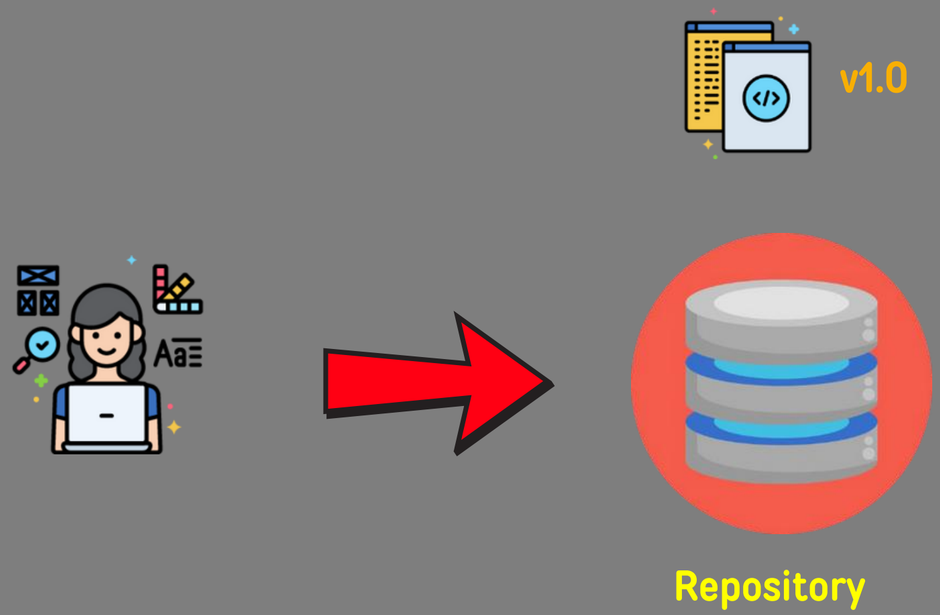
Version 1.0

Version 2.0

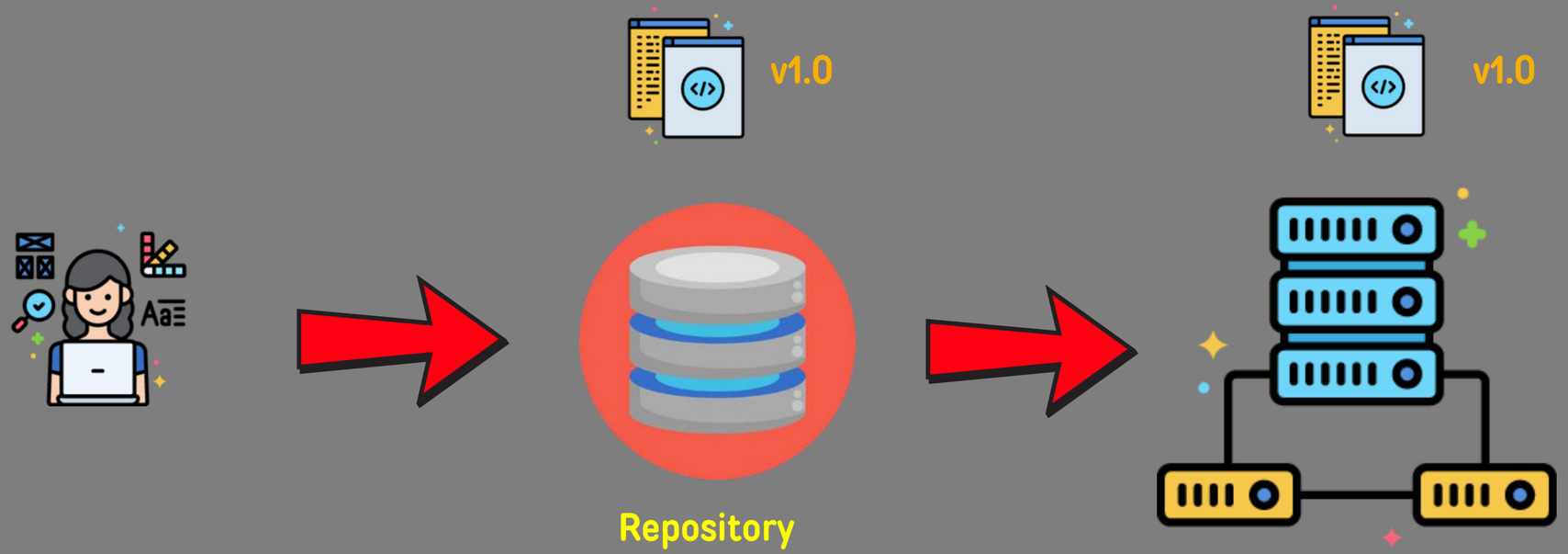
running code version 2.0

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

Version Control System



Version Control System



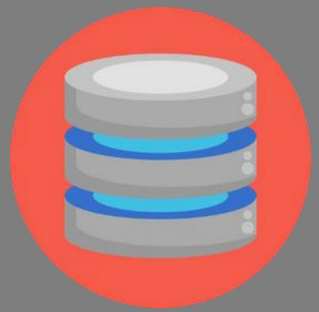
Version Control System

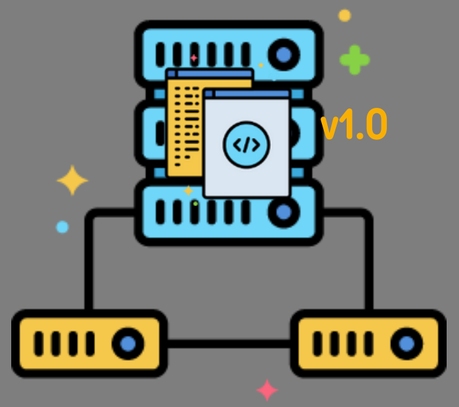
Version Control System

v2.0

Repository

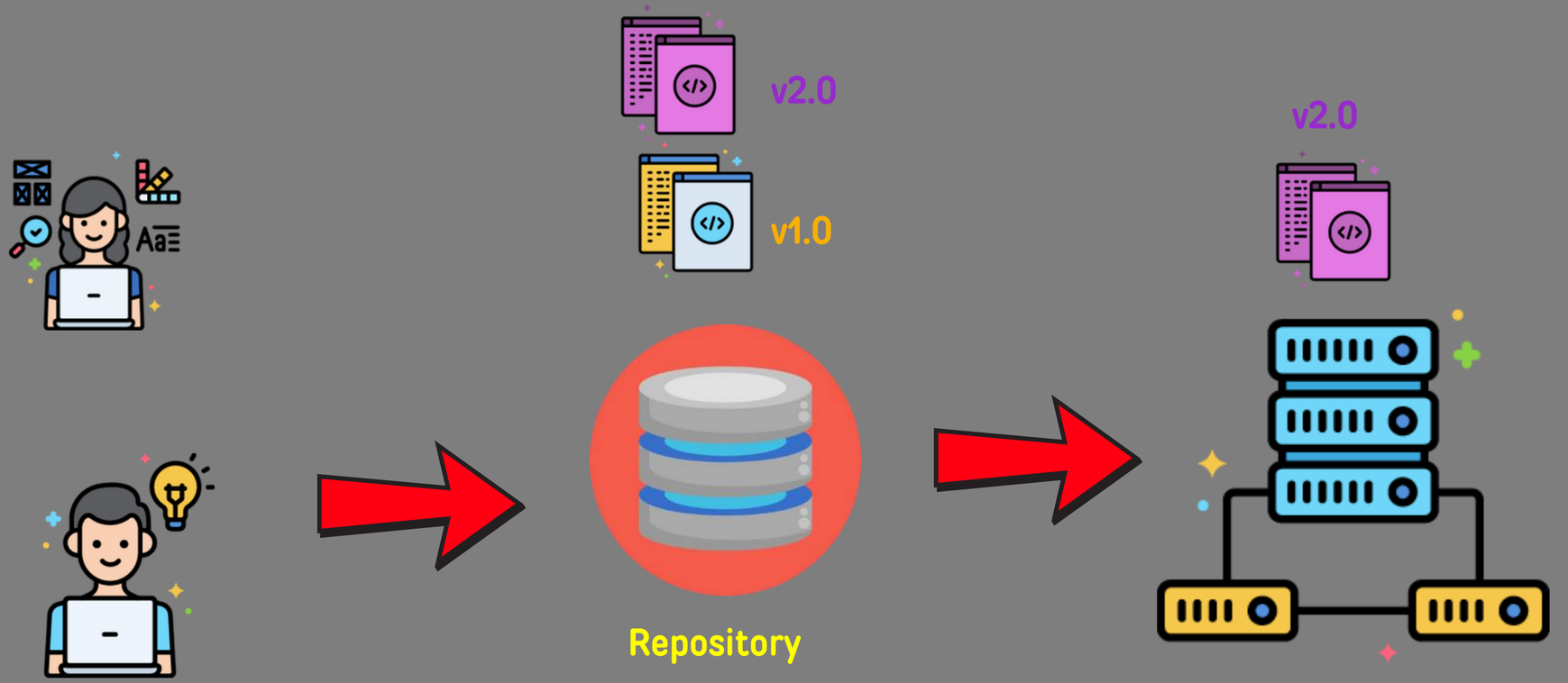
v1.0



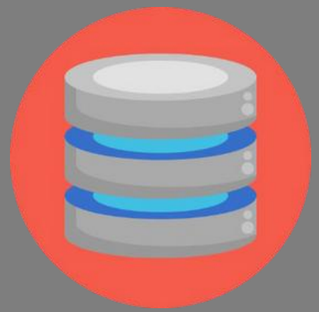


v1.0

Version Control System



Version Control System

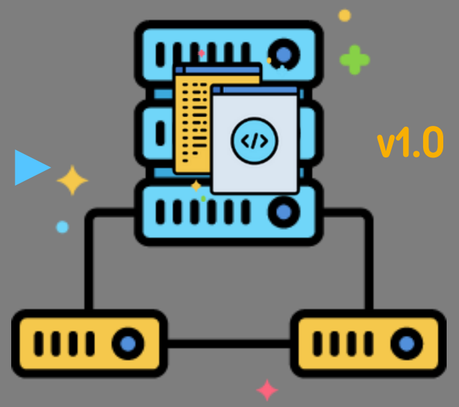


Repository

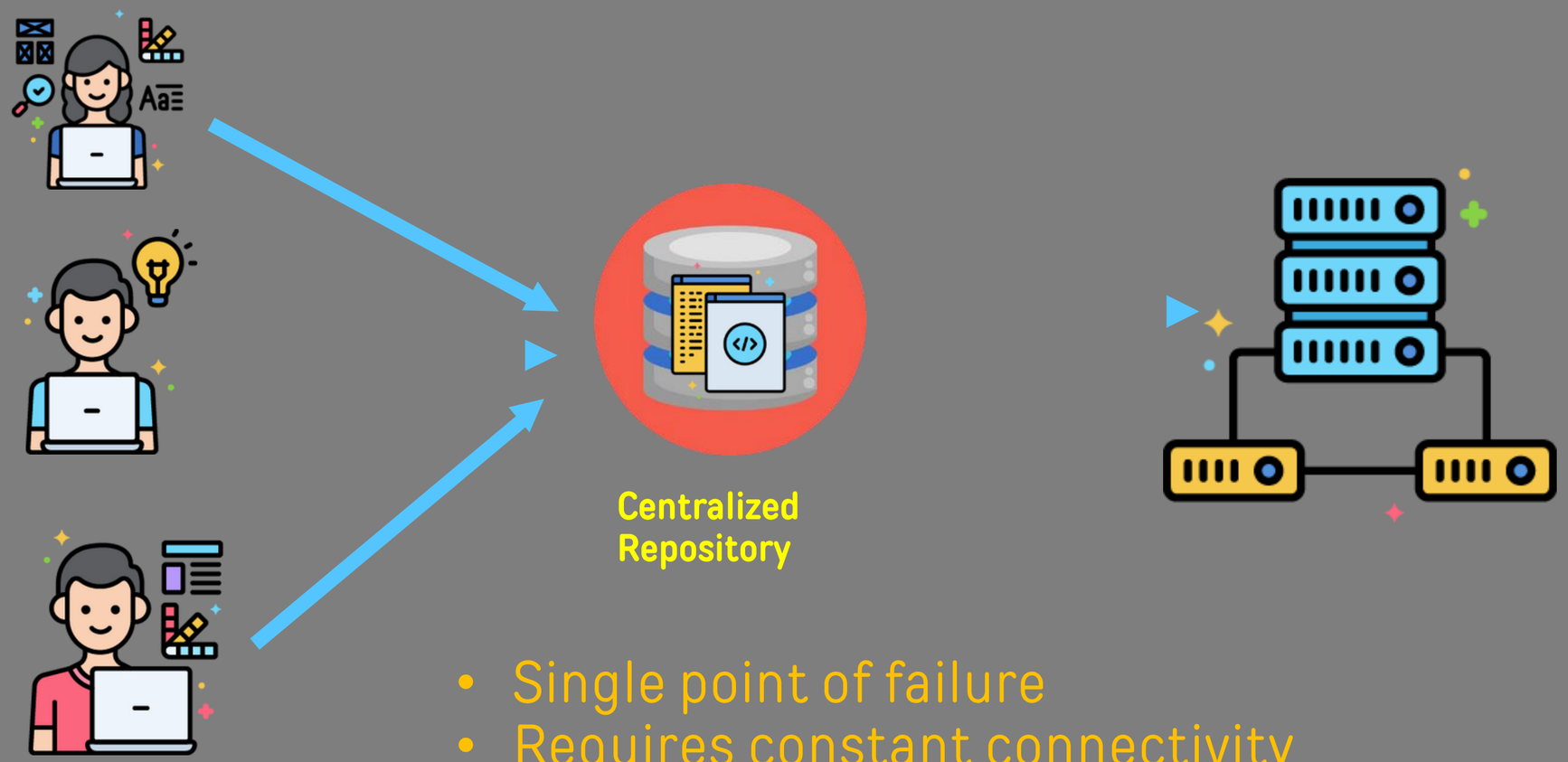
Version Control System

Why Git?

- Distributed

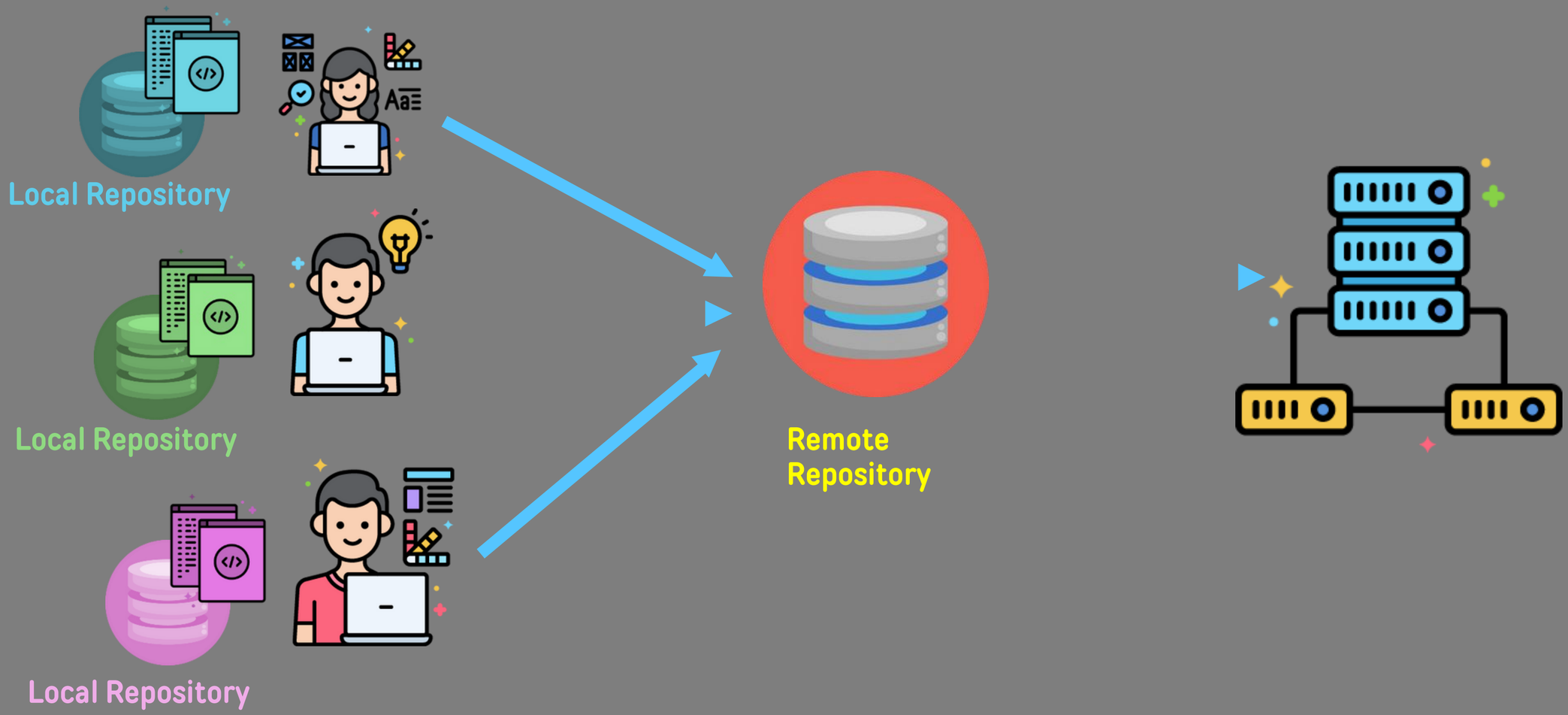


Centralized Version Control System



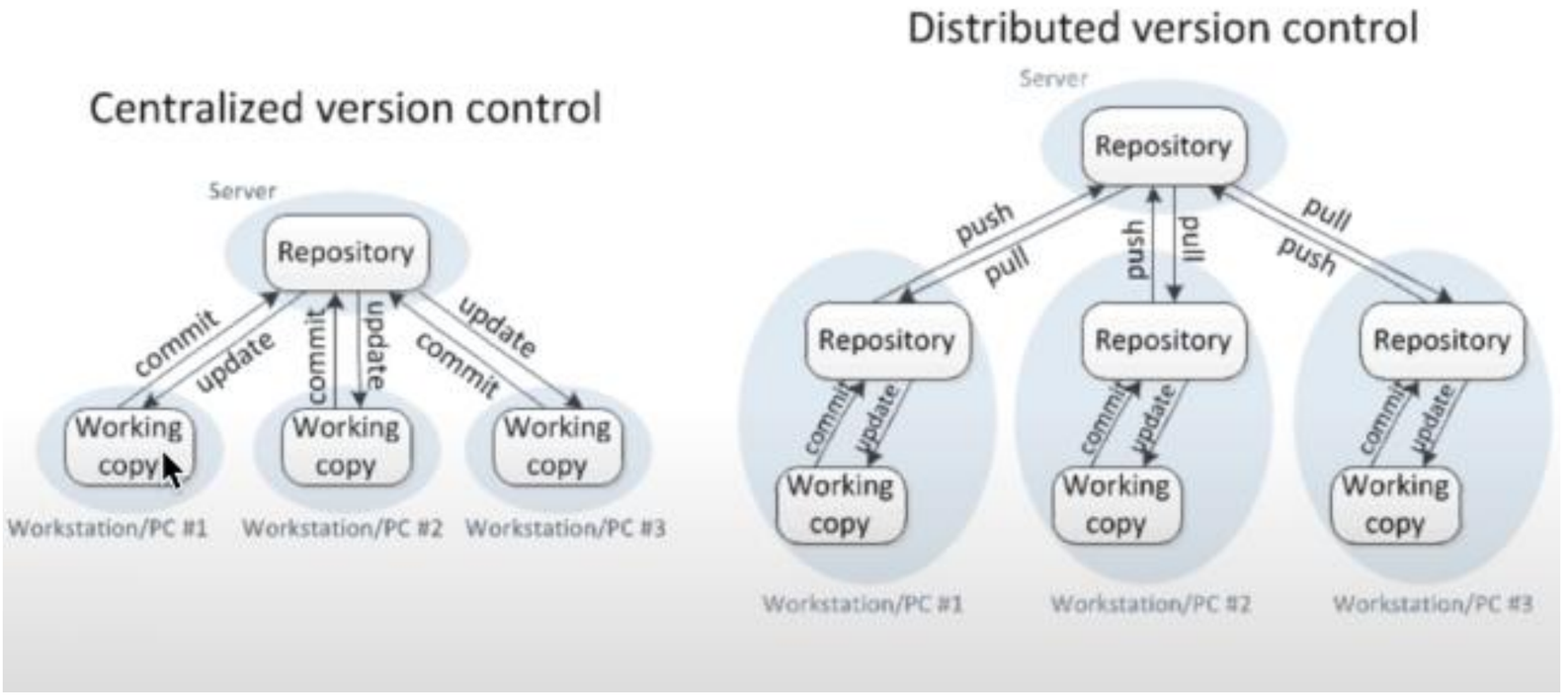
- Single point of failure
- Requires constant connectivity
- E.G – Subversion, Endevor

Distributed Version Control System

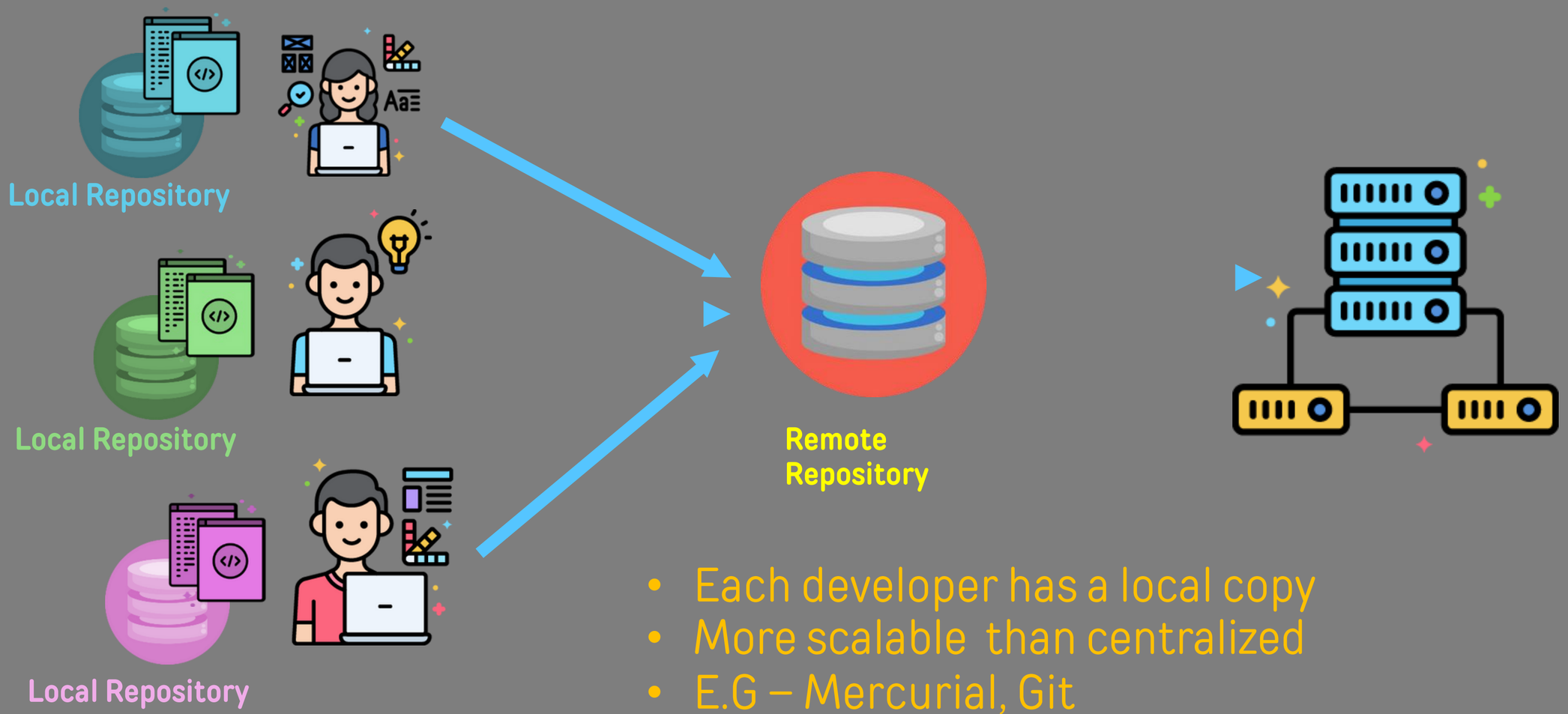


Centralized vs Distributed

Version Control

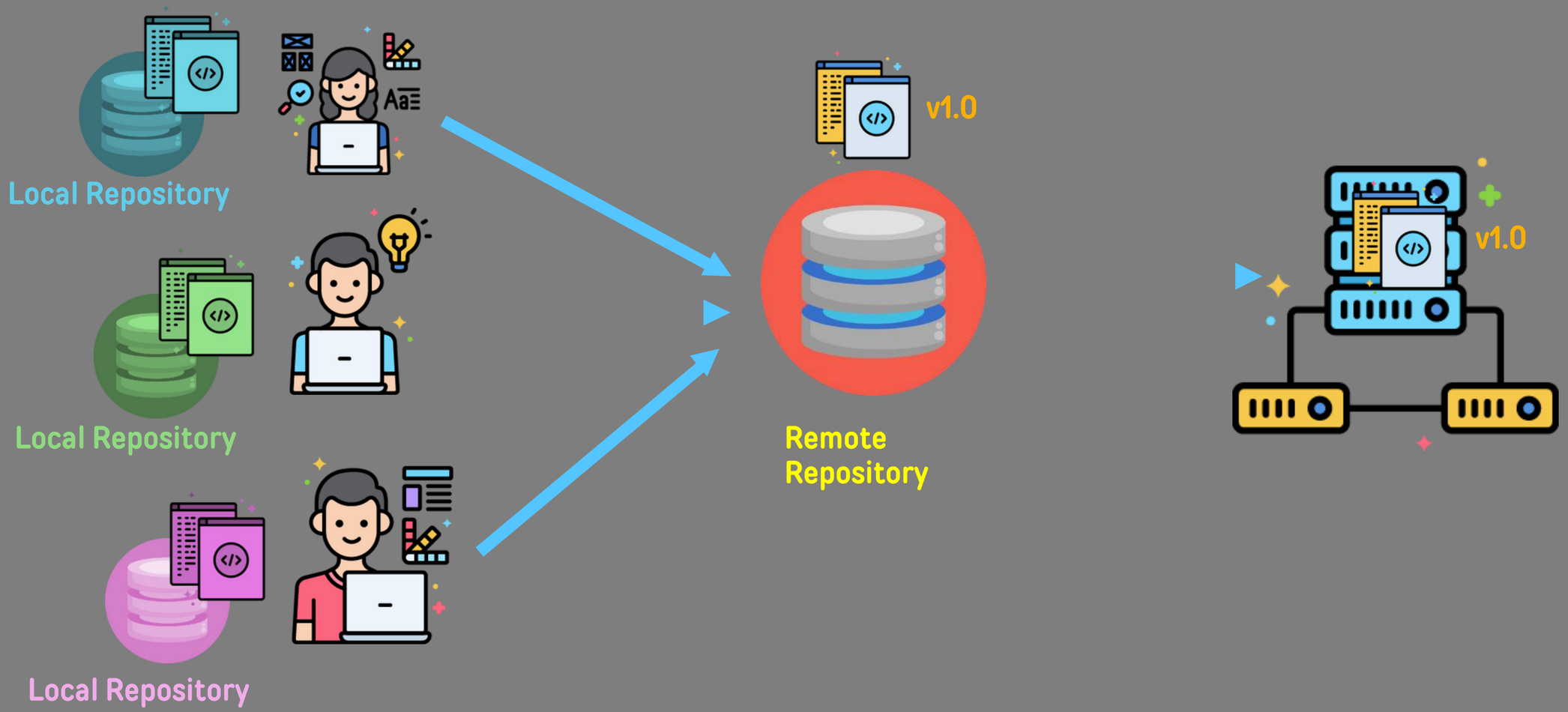


Distributed Version Control System



- 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

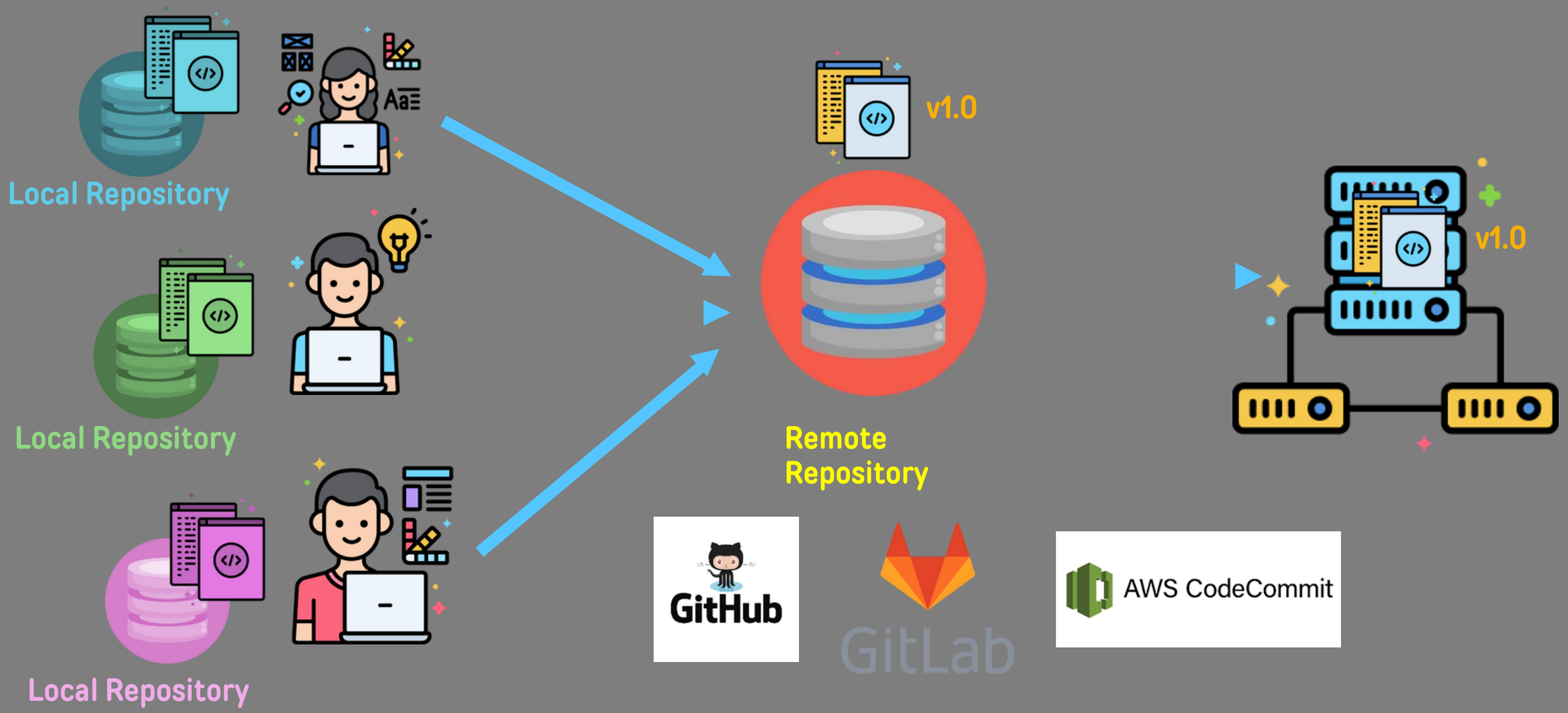
Git vs GitHub

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

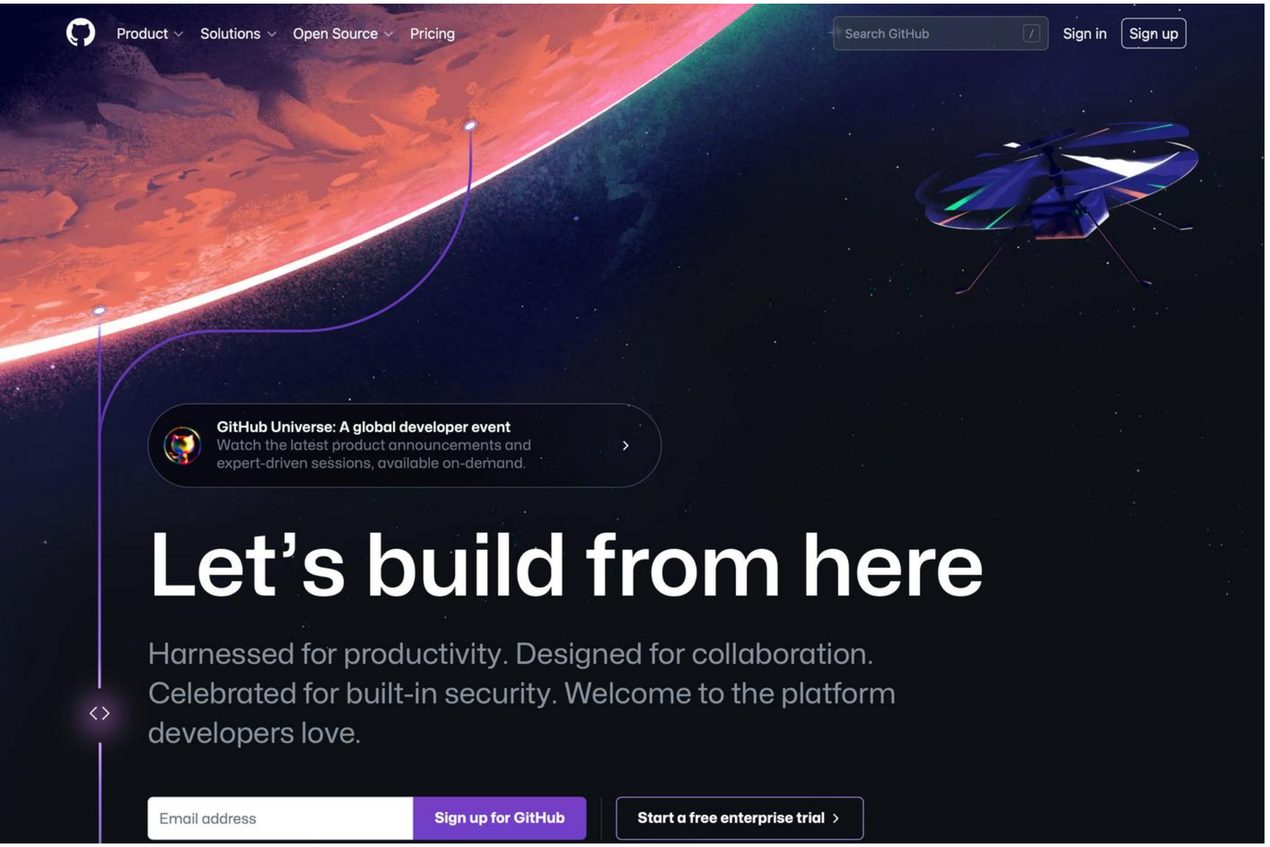
- Git repository hosting services with other features
- Runs on the cloud
- Created in 2008, currently owned by Microsoft
- Not open source, have free and paid tiers

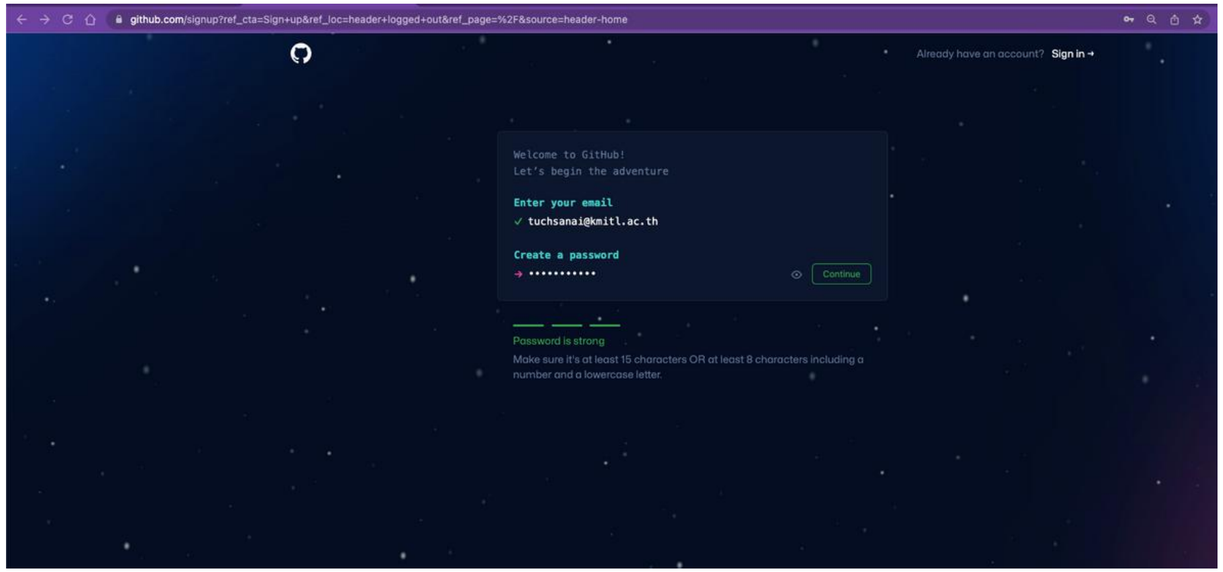
Distributed Version Control System



Week 1 - Starting with Git

Sign Up : https://github.com





https://github.com

Installing Git

Week 1 - Starting with Git

- **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

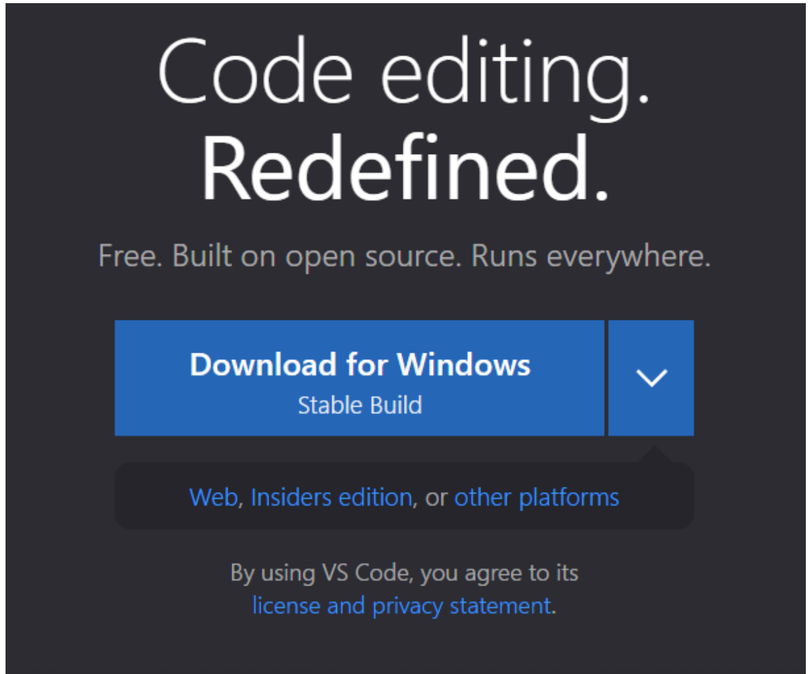
- **MacOS or Linux Users:**
  - Our suggested text editor for this course is VS Code:
    - **https://code.visualstudio.com/**
  - Its created by Microsoft and has direct integrations with GitHub and is one of the most popular text editors today.
  - You can follow along with any text editor you prefer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

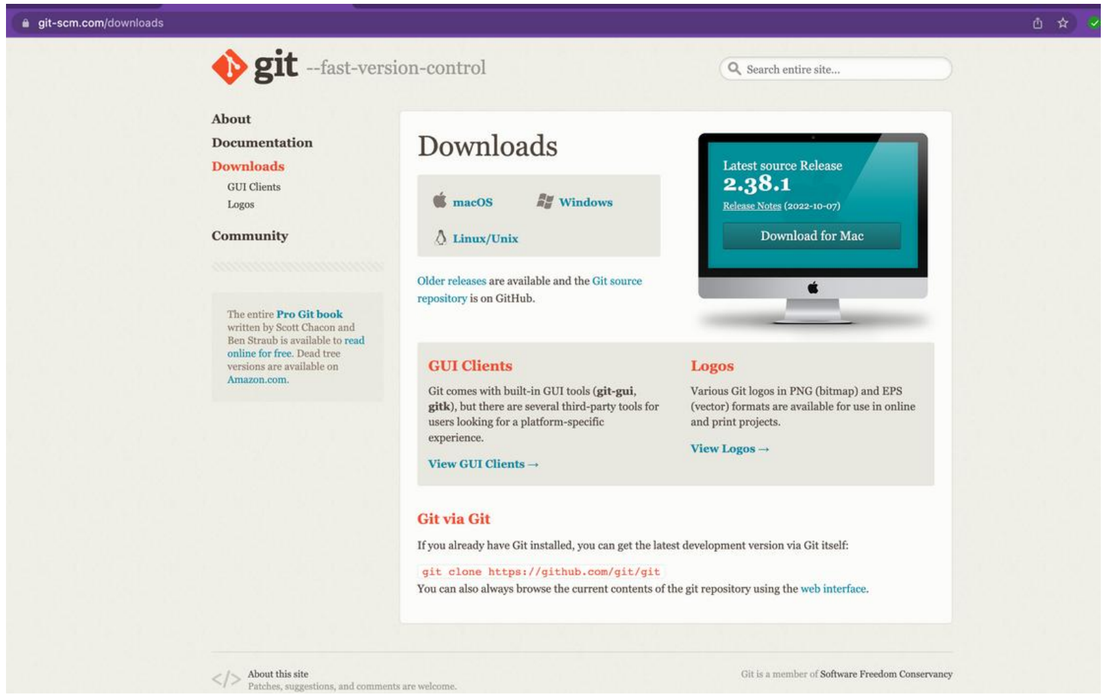
Week 1 - Starting with Git

- **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

- You can check the current configuration with the commands:

- 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”**

- -- ลบ config เก่า --
- git config --global --unset user.name
- git config --global --unset user.email
- git config --global --unset credential.username

- -- ลบ origin คนเก่า ถ้า commit ไม่ได้ --
- git remote remove origin

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

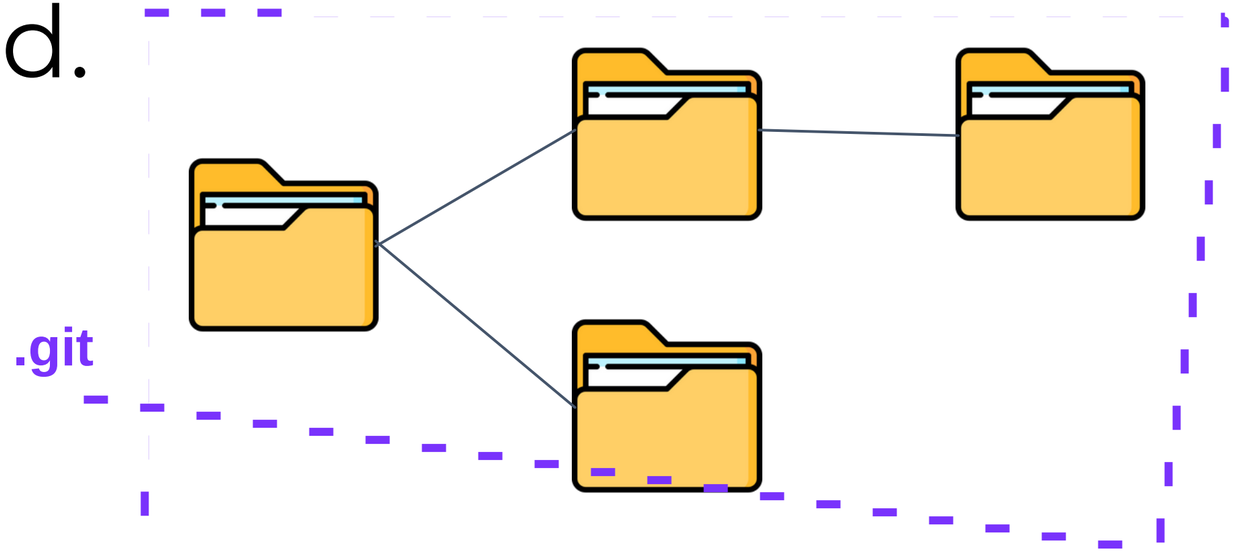
- Git inside a Folder/Directory:
  - Upon creating a Git Repository, all the folders/directories inside the top level Git Repository will also be part of that Repository, meaning all the changes are tracked.

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.

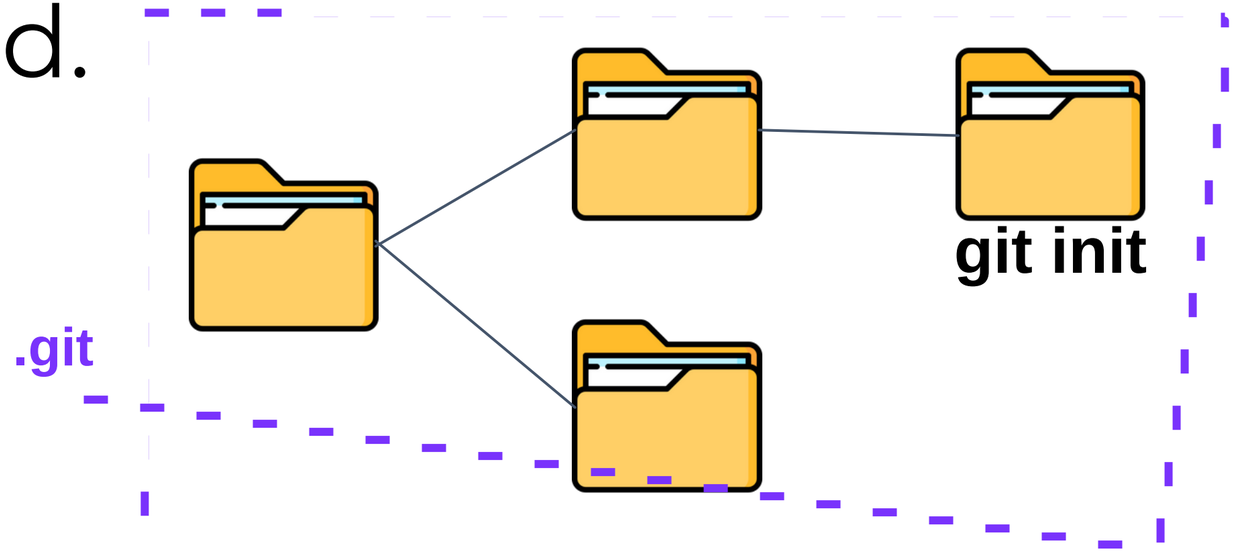
Day 1 - Starting with Git

- 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.



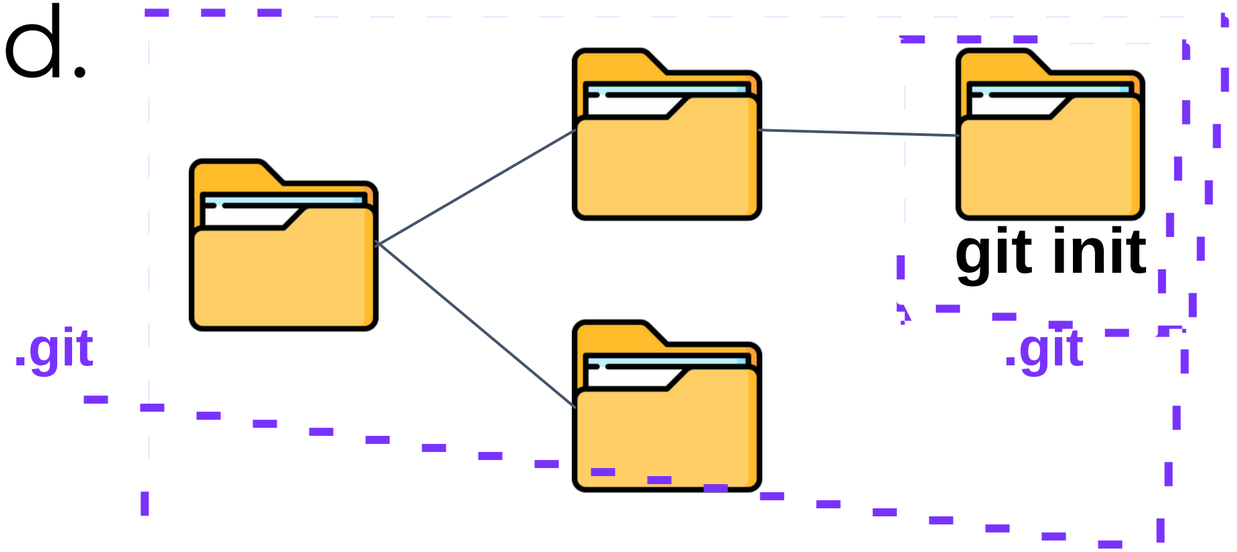
Day 1 - Starting with Git

- 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.



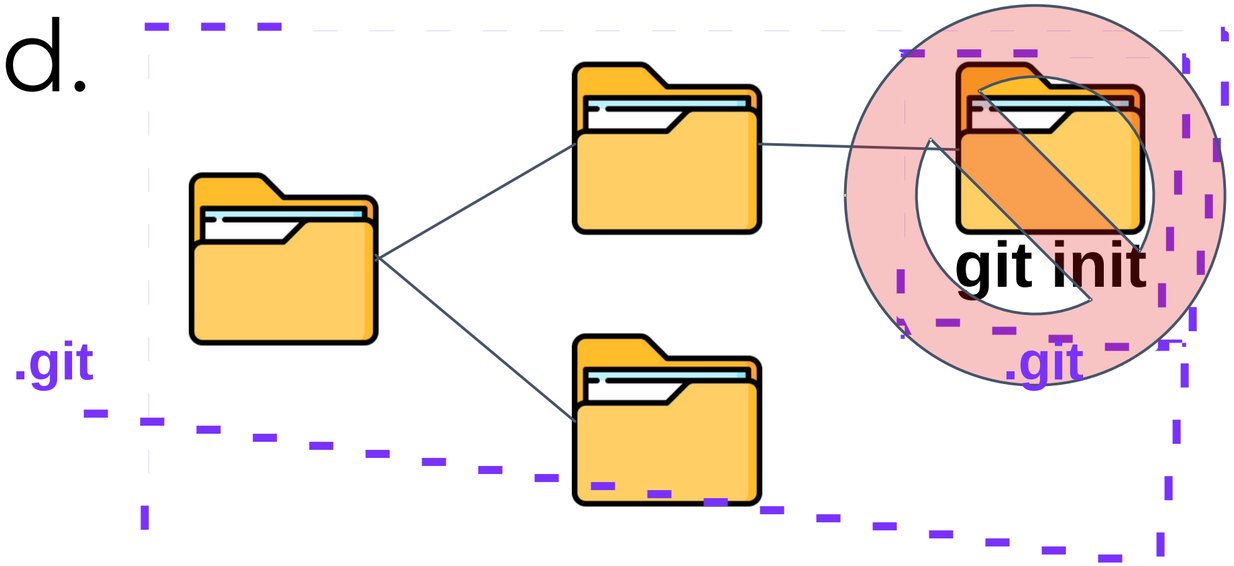
Day 1 - Starting with Git

- 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.



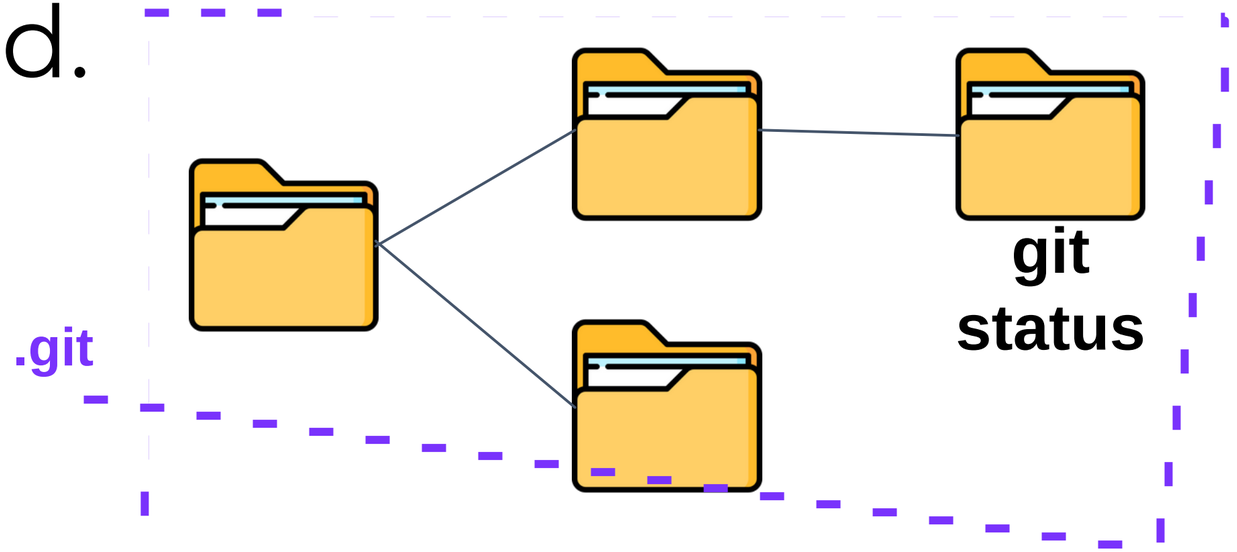
Day 1 - Starting with Git

- 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

DAY 1
Private Repositories and Tokens

Day 1 - Starting with Git

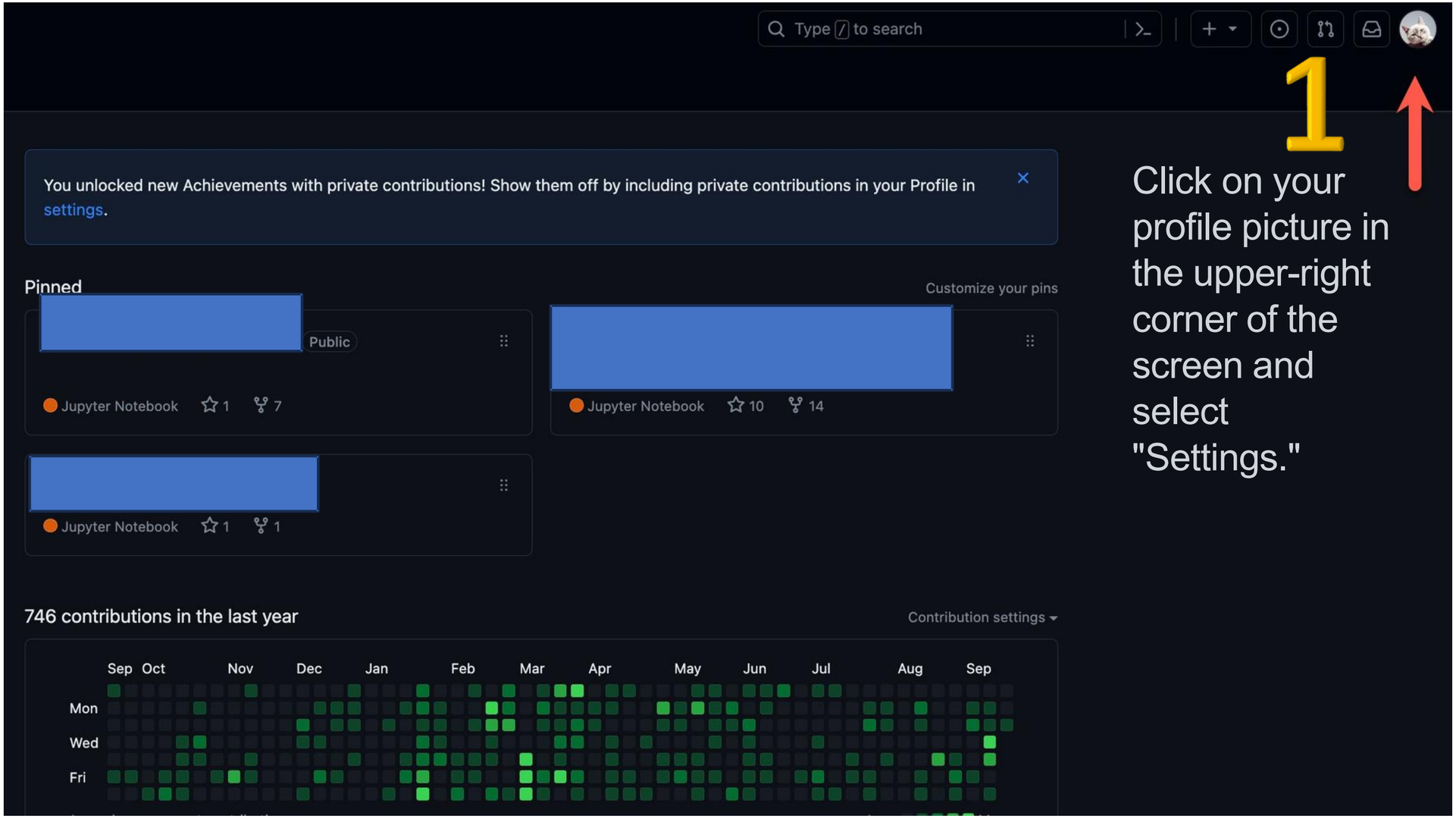
- Clone Syntax with PAT:

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

- Previously we used:

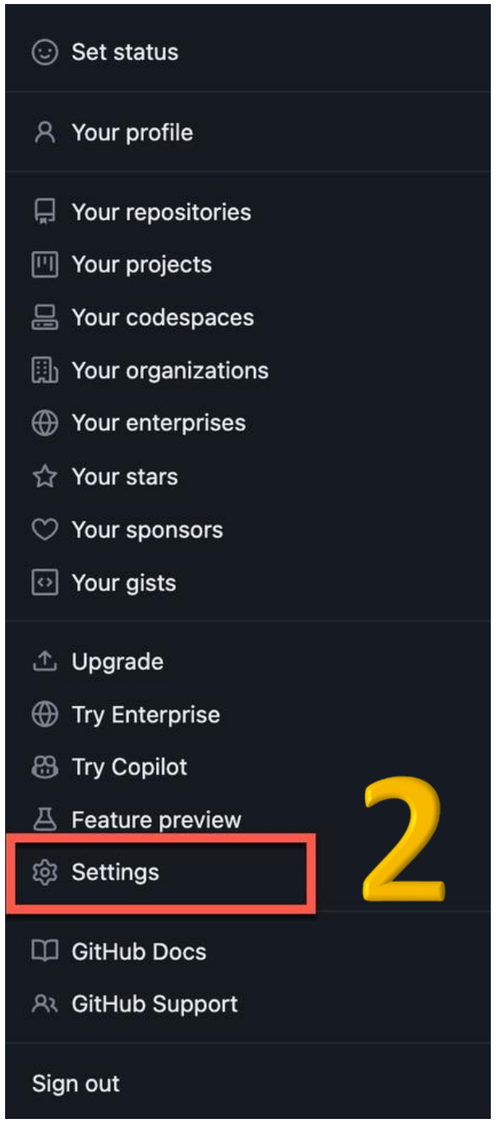
git clone https://github.com/account/repo.git

Create a Personal Access Token

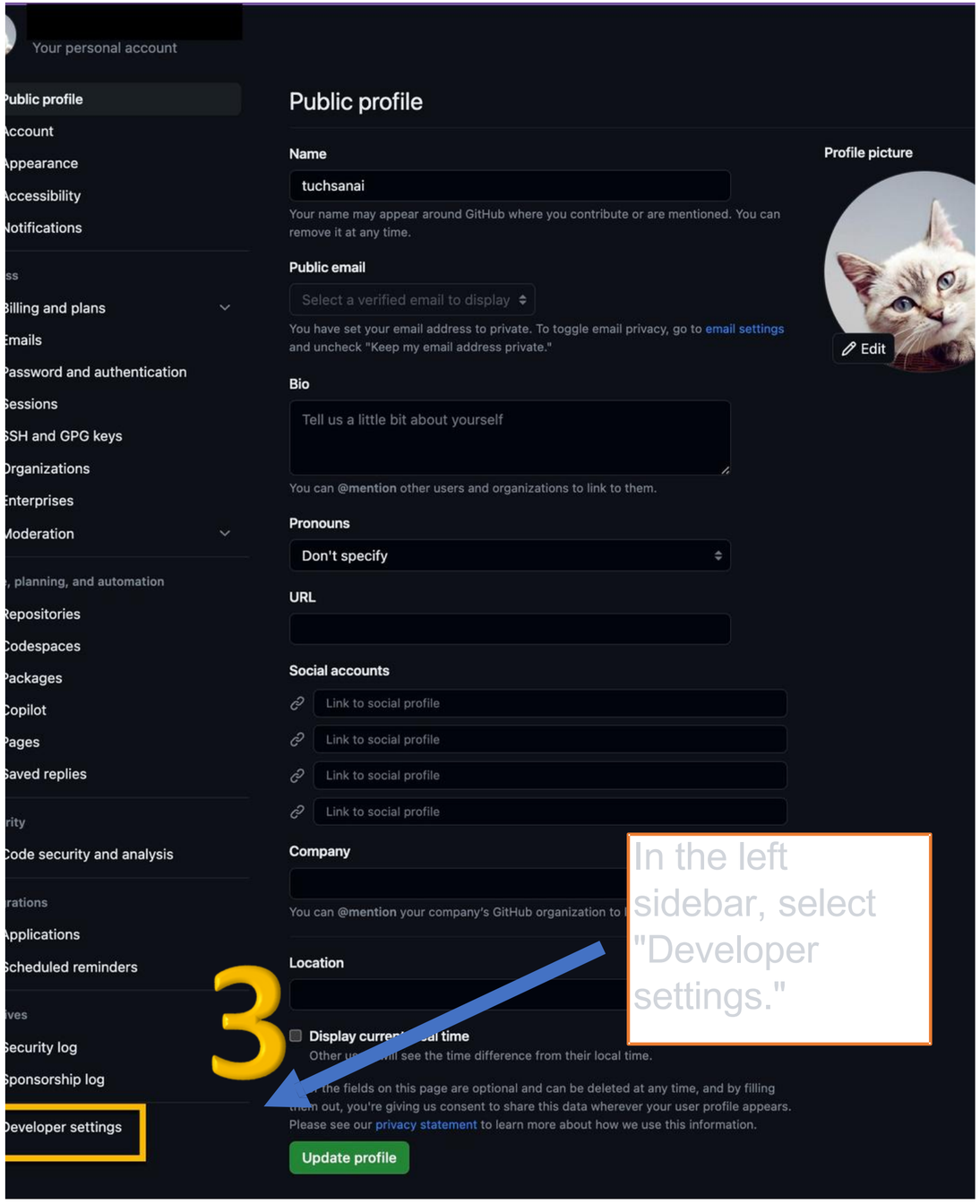


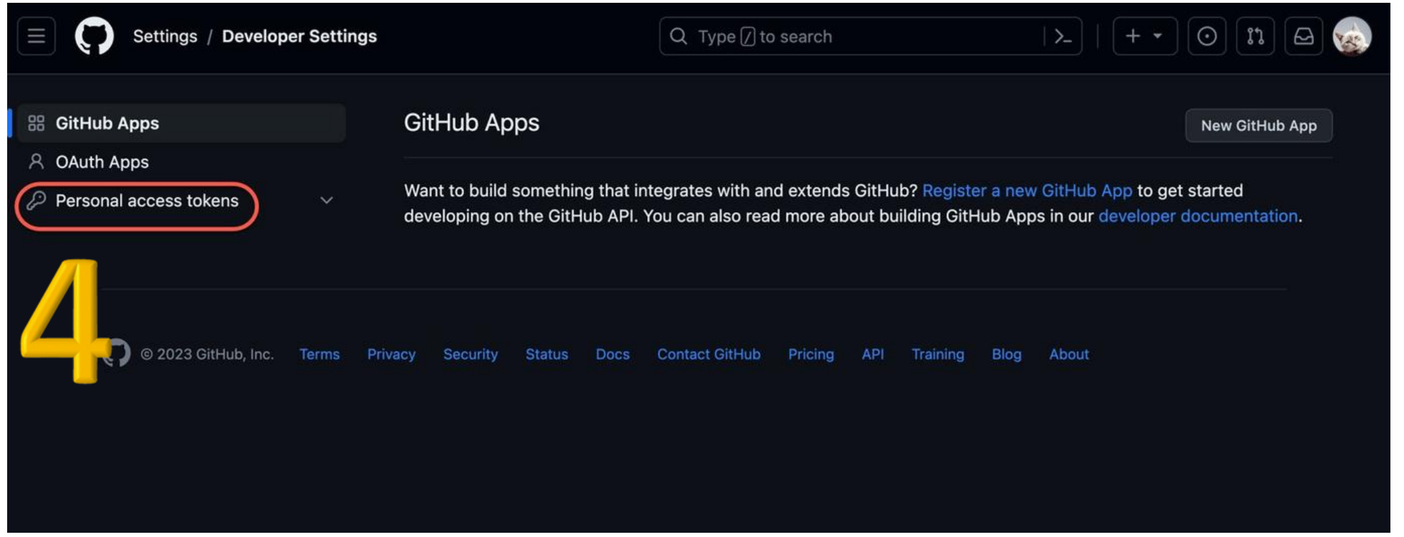
1

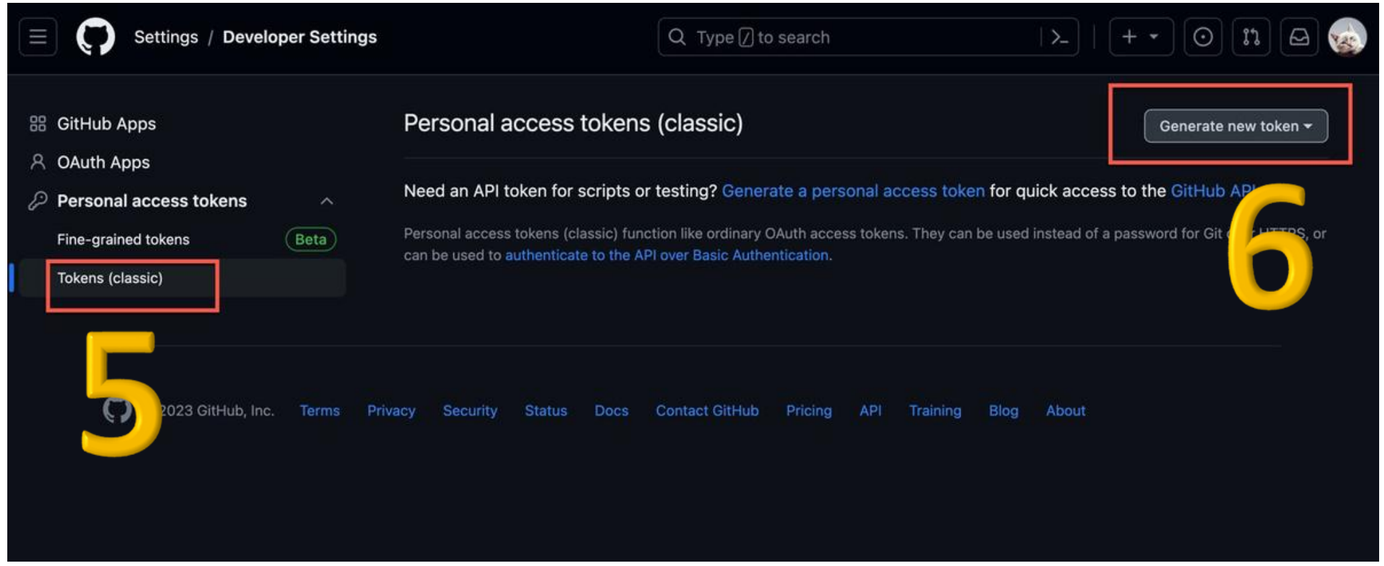
Click on your profile picture in the upper-right corner of the screen and select "Settings."



2


Your personal account
Public profile
Account
Appearance
Accessibility
Notifications
Billing and plans
Emails
Password and authentication
Sessions
SSH and GPG keys
Organizations
Enterprises
Moderation
Repositories
Codespaces
Packages
Copilot
Pages
Saved replies
Code security and analysis
Applications
Scheduled reminders
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
Company
Location
Profile picture
Edit
Update profile
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.
5
6
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

7

What's this token for?

Expiration *

7 days

The tok… expire on Tue, Sep 26 2023

8

Select scopes

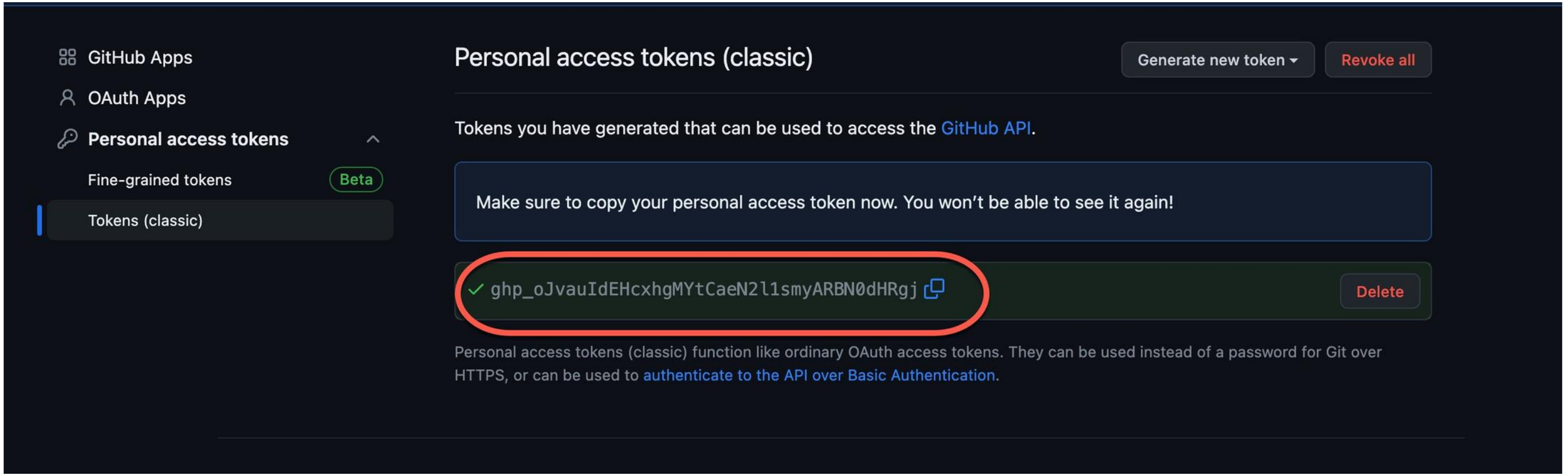
Scopes define the access for personal tokens. Read more about OAuth scopes.

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	Ful… ontrol of user public keys
write:public_key	Wr… user public keys
read:public_key	R… d user public keys
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

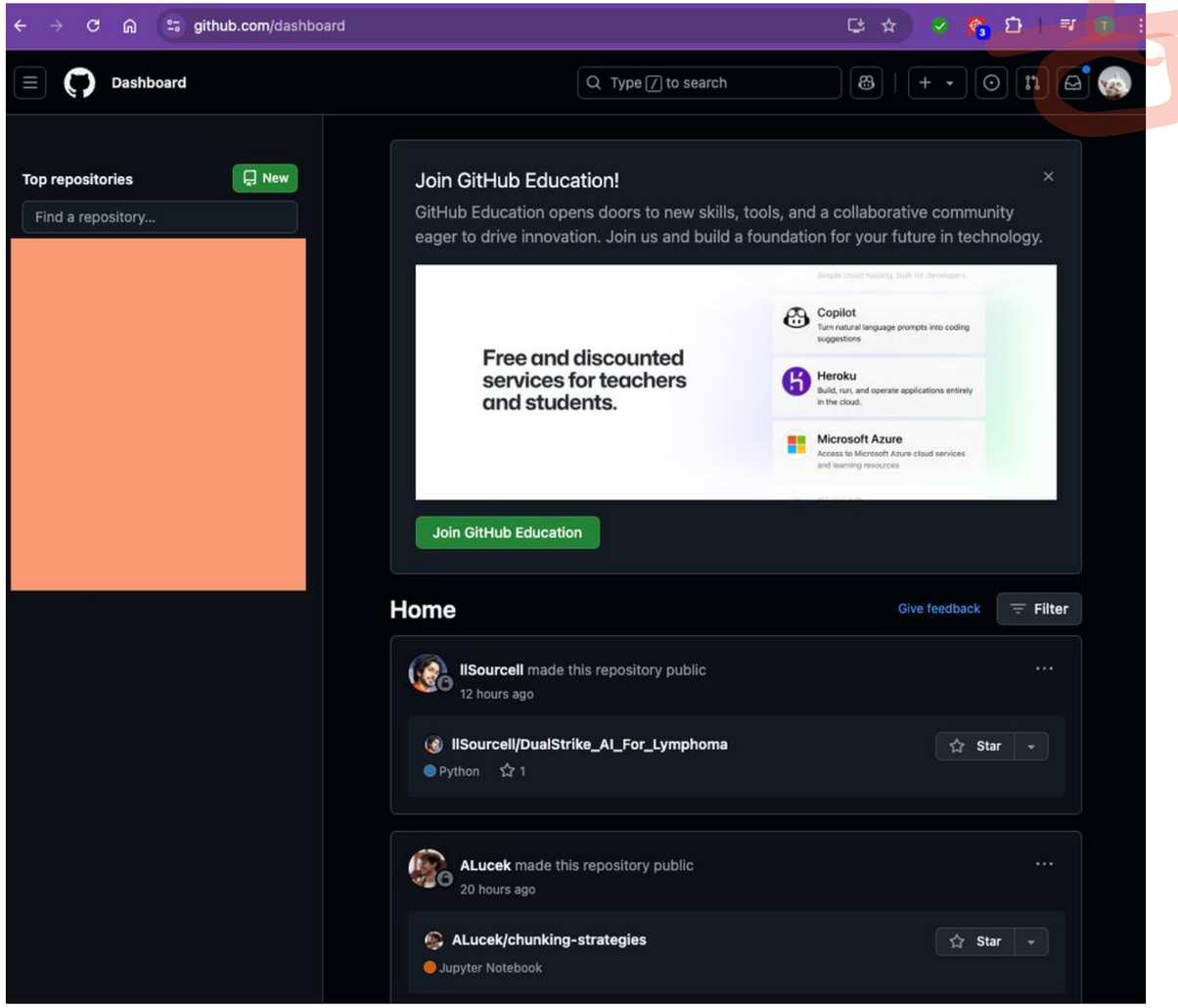
10

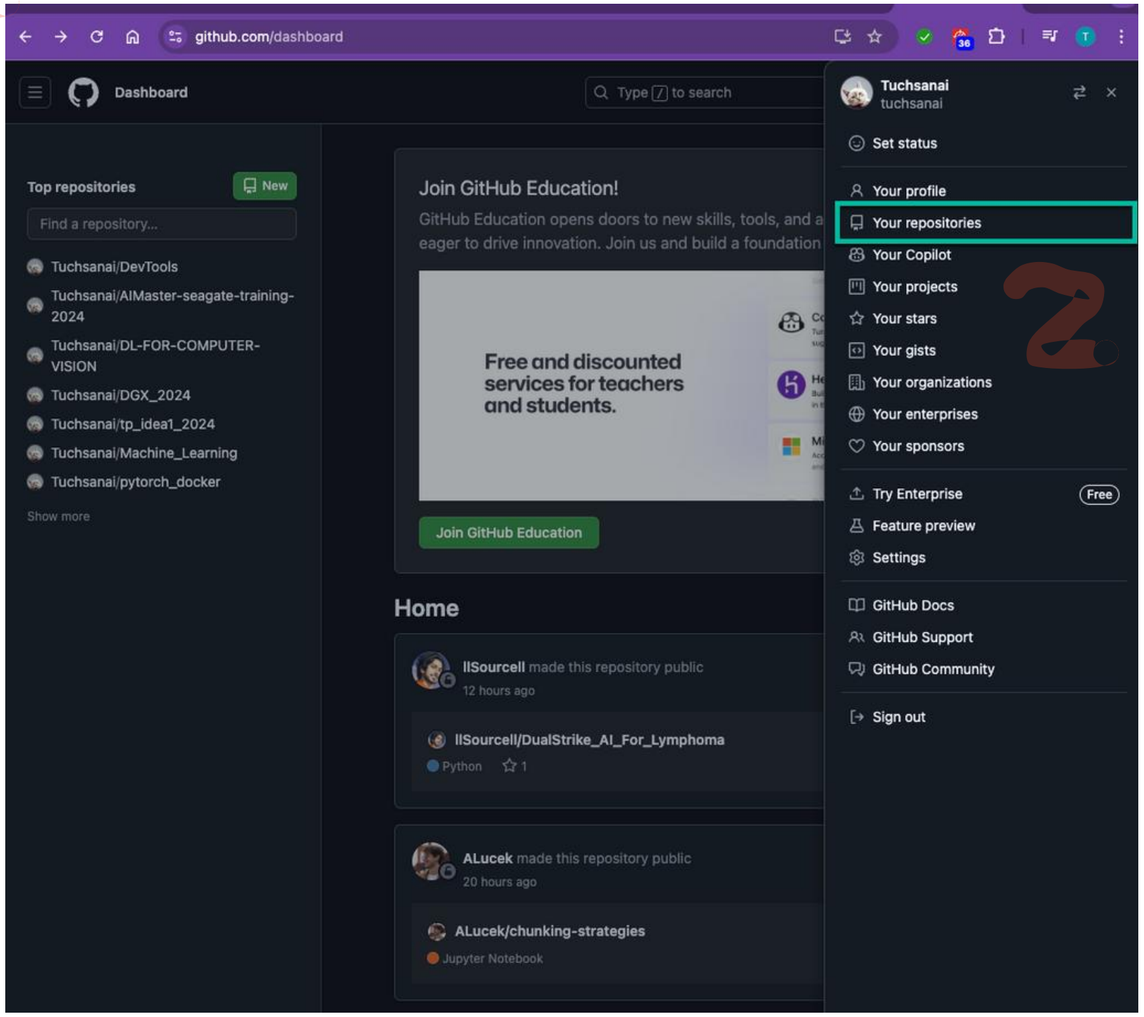

GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Beta
Tokens (classic)
Personal access tokens (classic)
Generate new token
Revoke all
Tokens you have generated that can be used to access the GitHub API.
Make sure to copy your personal access token now. You won't be able to see it again!
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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
Exercise

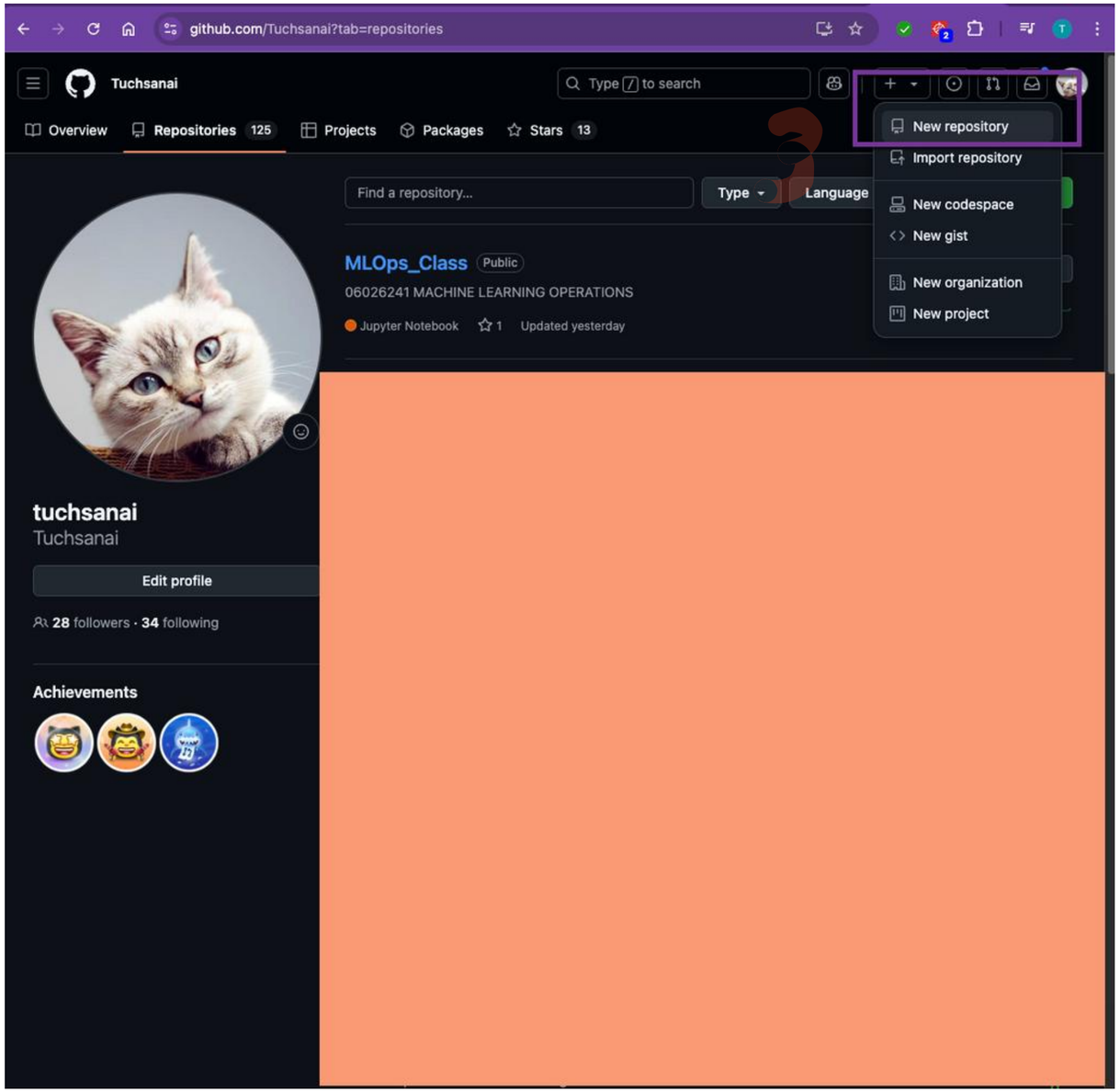
Day 1 - Starting with Git

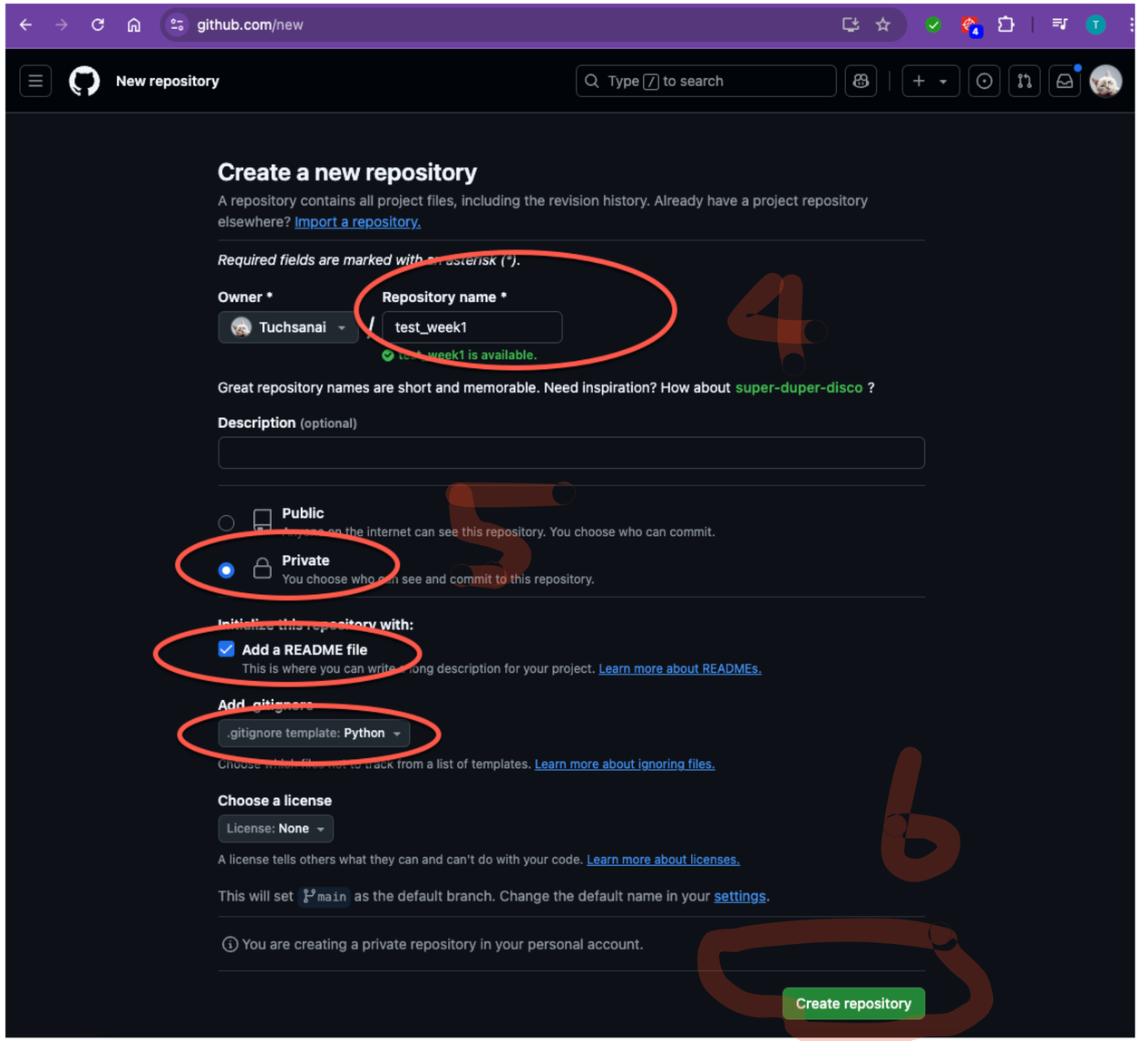
- **Exercise Tasks:**
  - Create a new Private Repository on GitHub.
  - Initialize your repository with README, license and gitignore.
  - create new test.txt with random data
  - Clone your Repository using the Command Line .


github.com/dashboard
Dashboard
Type / to search
Top repositories
New
Find a repository...
Join GitHub Education!
GitHub Education opens doors to new skills, tools, and a collaborative community eager to drive innovation. Join us and build a foundation for your future in technology.
Free and discounted services for teachers and students.
Copilot
Turn natural language prompts into coding suggestions
Heroku
Build, run, and operate applications entirely in the cloud.
Microsoft Azure
Access to Microsoft Azure cloud services and learning resources
Join GitHub Education
Home
Give feedback
Filter
llSourcell made this repository public
12 hours ago
llSourcell/DualStrike_AI_For_Lymphoma
Python
1
Star
ALucek made this repository public
20 hours ago
ALucek/chunking-strategies
Jupyter Notebook
Star

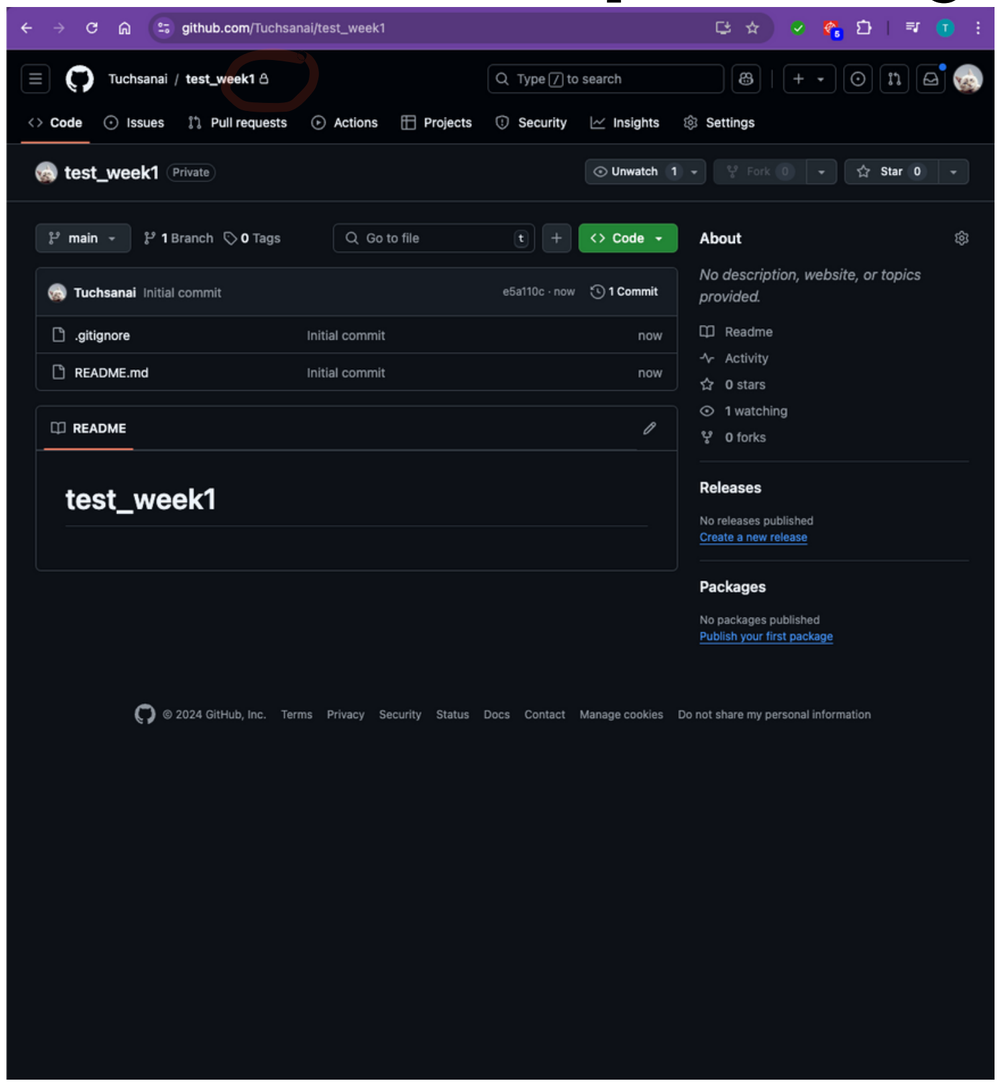

github.com/dashboard
Dashboard
Type / to search
Top repositories
New
Find a repository...
Tuchsanai/DevTools
Tuchsanai/AIMaster-seagate-training-2024
Tuchsanai/DL-FOR-COMPUTER-VISION
Tuchsanai/DGX_2024
Tuchsanai/tp_idea1_2024
Tuchsanai/Machine_Learning
Tuchsanai/pytorch_docker
Show more
Join GitHub Education!
Free and discounted services for teachers and students.
Join GitHub Education
Home
llSourcell made this repository public
12 hours ago
llSourcell/DualStrike_AI_For_Lymphoma
Python
1
ALucek made this repository public
20 hours ago
ALucek/chunking-strategies
Jupyter Notebook
Tuchsanai
tuchsanai
Set status
Your profile
Your repositories
Your Copilot
Your projects
Your stars
Your gists
Your organizations
Your enterprises
Your sponsors
Try Enterprise
Free
Feature preview
Settings
GitHub Docs
GitHub Support
GitHub Community
Sign out

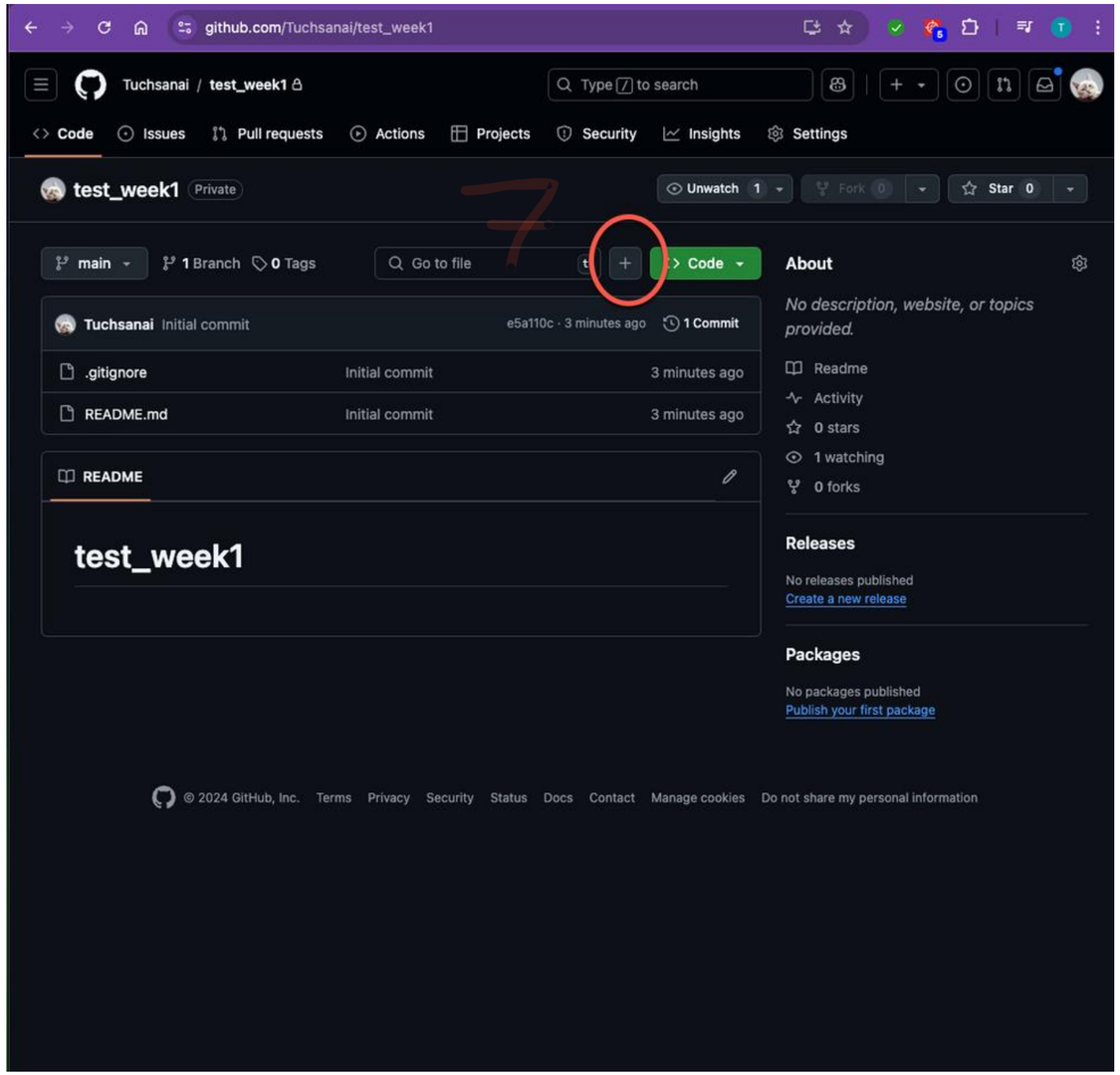
Click New repository



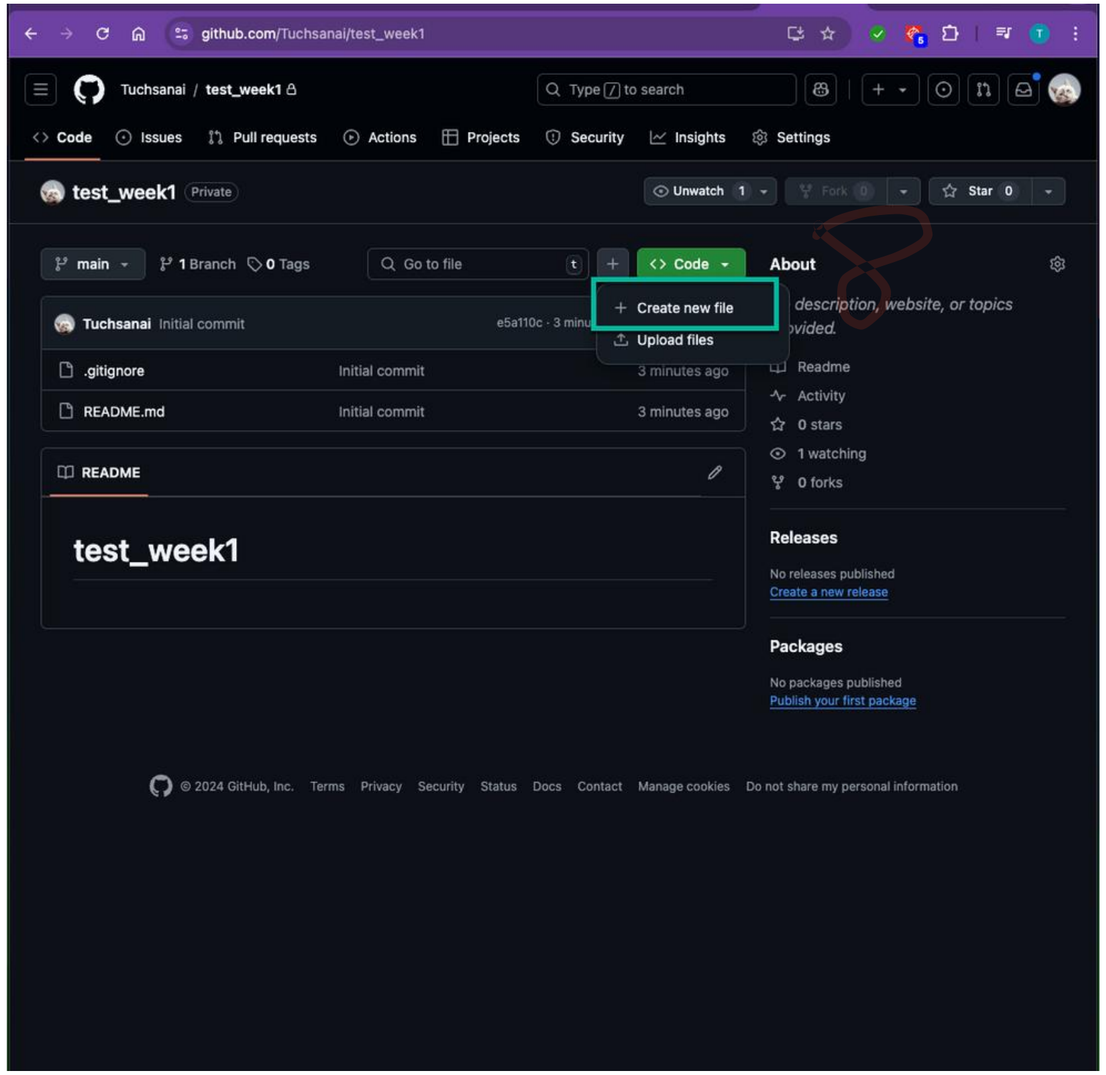


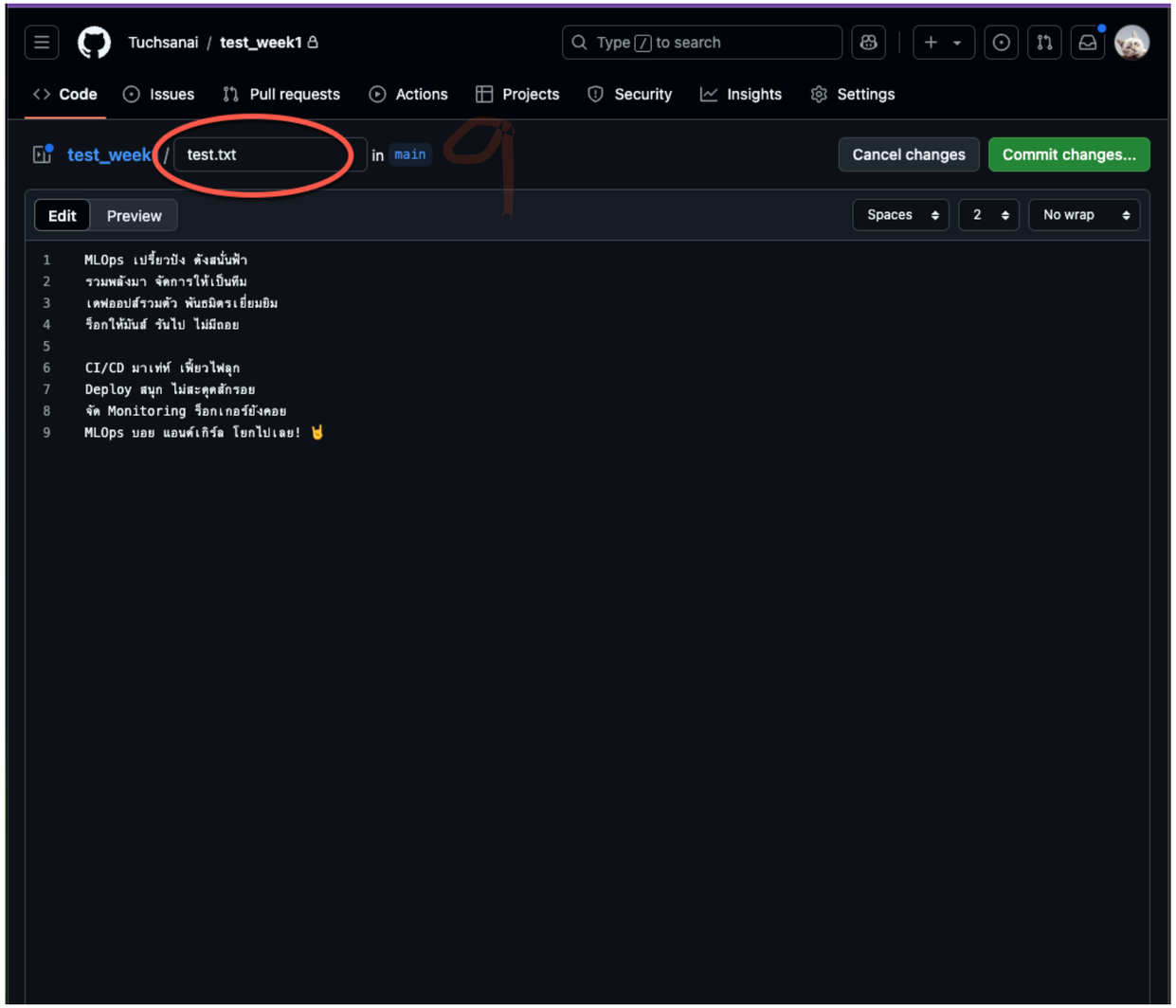
Privated repository

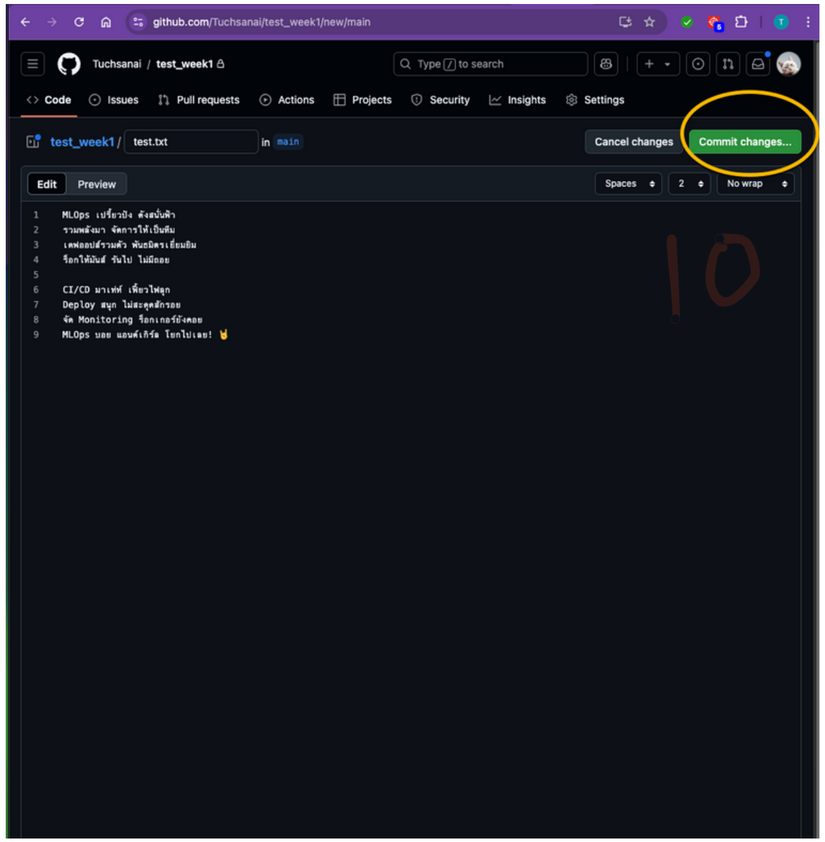


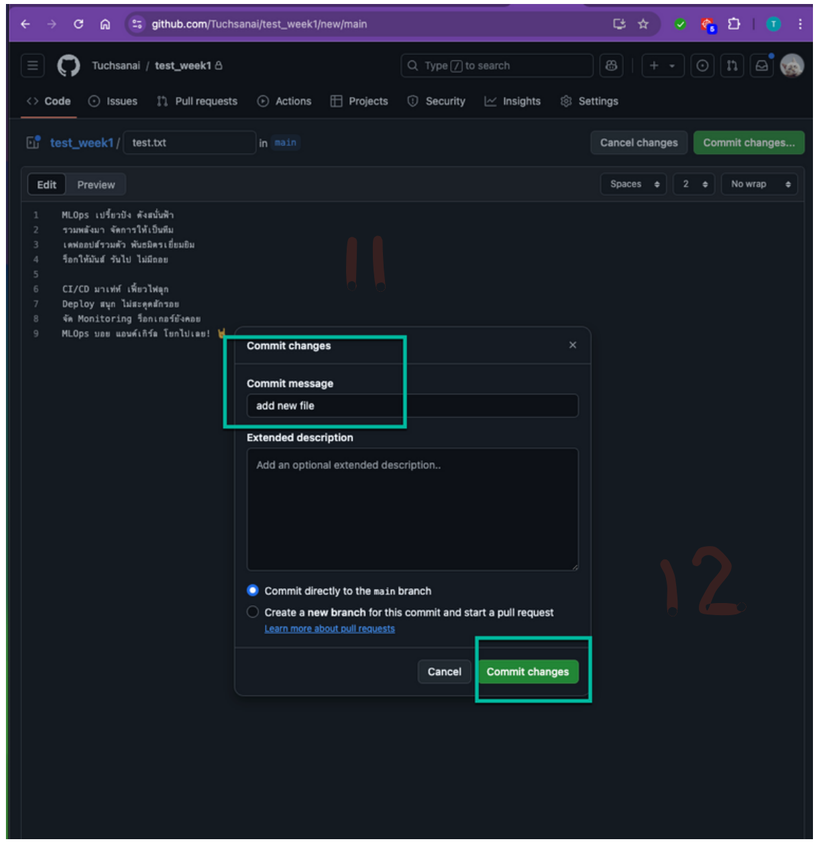


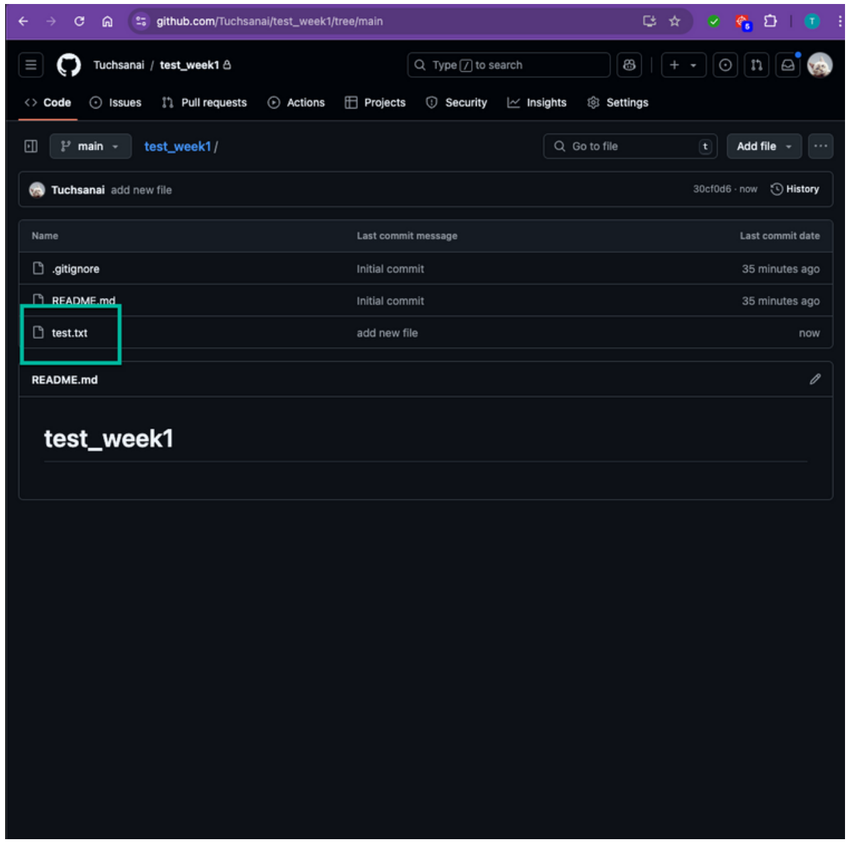
สร้าง file : test.txt











- Clone private respository:

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