



**Document Create Date:**

**Version:** 1.0

**Status:** Initial Version

**Document Author:** นายสุรวัชร มั่นกตัญญู

**Document Reviewer:** นายกิตติเมศศักดิ์ วังศรี

## คำนำ

เอกสารฉบับนี้มีวัตถุประสงค์เพื่อให้ความรู้และข้อมูลเกี่ยวกับการใช้งาน GitLab CI/CD ซึ่งเป็นเครื่องมือที่ช่วยในเรื่องของการพัฒนาซอฟต์แวร์ร่วมกัน และการทดสอบโค้ดแบบอัตโนมัติตามสคริปต์ที่เราได้เขียนไว้ในไฟล์ .gitlab-ci.yml

ในเอกสารนี้จะครอบคลุมถึงการติดตั้งและการตั้งค่าเบื้องต้นของ GitLab CI/CD และ GitLab runner การนำเข้าข้อมูล การสร้างโปรเจค รวมถึงการตั้งค่าต่างๆ

ผู้จัดทำหวังว่าเอกสารฉบับนี้จะเป็นประโยชน์สำหรับนักพัฒนาระบบ และผู้ที่สนใจ และช่วยให้ท่านนำความรู้ที่ได้รับไปประยุกต์ใช้ในการทำงานได้ อย่างมีประสิทธิภาพ

นายศุภรักษ์ มั่นกตัญญู

## สารบัญ

หัวข้อ	หน้า
1.GitLab CI/CD คือ .....	1
ความแตกต่างของ CI และ CD มีอะไรบ้าง? .....	2
2.ทำไมเราถึงต้องใช้ GitLab CI/CD .....	3
3.ยกตัวอย่าง usecase .....	4
เครื่องมือสำหรับการทำ CI/CD ที่คนส่วนใหญ่นิยมมีอะไรบ้าง? .....	4
4.Demo .....	5
4.1 ติดตั้ง git .....	5
4.2 สมัคร gitlab .....	6
4.3 ทำการ config username and email gitlab.....	7
4.4 สร้าง project ใน gitlab.....	8
4.5 ทำการ set up ssh key ใน gitlab.....	10
4.6 สร้าง Project runner .....	13
4.7 ทำการ install gitlab-runner.....	16
4.8 ทำการ register .....	18
การ Upload file .....	22
.gitlab-ci.yml .....	24
.gitlab-ci.yml คืออะไร .....	24
.gitlab-ci.yml ทำงานยังไง? .....	24
ความแตกต่างของ Stages .....	24
ตัวอย่าง Code.....	25
การตรวจสอบสถานะ Pipeline .....	26

## GitLab CI/CD

### 1. GitLab CI/CD คือ

GitLab CI/CD คือ แนวทางการพัฒนาซอฟต์แวร์ที่เน้นการทำงานร่วมกัน และส่งมอบงานให้ได้ไวมากยิ่งขึ้น GitLab CI/CD เป็นเครื่องมือที่ช่วยให้สามารถ:

รันงาน build และ test โค้ดได้อัตโนมัติ เช่น ทุกครั้งที่มีการ push โค้ดใหม่ GitLab CI/CD จะรันงาน build และ test โค้ดให้ ไม่ต้องพิมพ์คำสั่ง build แค่ push แล้วรอดูผลลัพธ์ นอกจากนี้ยังสามารถใช้ Deploy โค้ดไปบน production ได้ ช่วยลดเวลาและความยุ่งยากในการ deploy

Continuous Integration (CI): คือกระบวนการรวบรวมโค้ดที่ได้รับการพัฒนาจากสมาชิกแต่ละคนในทีมให้เป็นชิ้นเดียว จากนั้นทำการทดสอบด้วย Test Script เพื่อตรวจสอบว่าโค้ดแต่ละส่วนเข้ากัน ไม่เกิดข้อผิดพลาดก่อนจะรวมโค้ดไปยัง Branch Main หรือ Master

CI จึงเข้ามามีบทบาทในการช่วย Developer ให้ Merge code ที่แก้ไขส่งกลับไป Pool กลาง เช่น Git Repository เวลาที่ Developer มีการเปลี่ยนแปลงอะไรเกี่ยวกับ App ก็จะมี CI ช่วย Merge code รวมไปถึงการทำ Automate test, Unit test และ Integration test เพื่อให้มั่นใจว่าสิ่งที่ Developer เปลี่ยนแปลงไปจะไม่กระทบกับการทำงานของ Application ทดสอบทุกอย่างตั้งแต่ Class ไปจนถึง Function ใน Module ที่แตกต่างกันแล้วนำมาประกอบเป็น Application ถ้า Automate test แล้วเจอปัญหาเรื่อง Conflict ระหว่าง Code เดิมกับของใหม่ CI จะช่วยให้ง่ายต่อการหา และแก้ไข Bug

หลังจากขั้นตอนการทำ CI แล้ว เราจะดำเนินการ Continuous Delivery และ Continuous Deployment (CD) เป็นกระบวนการส่งออกโค้ดหรือระบบ ที่พัฒนาขึ้นไปยังระบบจริง หรือที่เรียกว่า Production โดยการทำงานของ CD จะเป็นการทำงานแบบอัตโนมัติ

ข้อแตกต่างของ Continuous Delivery และ Continuous Deployment

- Continuous Deployment เป็นกระบวนการที่ทำครบทุกขั้นตอน ตั้งแต่รวมโค้ดของแต่ละคนในสมาชิกลงไปจนถึงส่งออกโค้ดขึ้นระบบจริงแบบอัตโนมัติทั้งหมด

Continuous Deployment ขั้นตอนนี้เป็นขั้นตอนสุดท้ายของ CI/CD Pipeline ซึ่งเป็นส่วนที่ขยายมาจาก Continuous Delivery ซึ่งจะทำหน้าที่จัดการโดยอัตโนมัติในส่วนของการนำ Code จาก Repository ไป Deploy บน Production Continuous Deployment จะเหมาะกับการใช้งานที่มีการ Deploy เป็นจำนวนมาก และมีขั้นตอน Test automation ที่ออกแบบมาเป็นอย่างดีโดยในทางปฏิบัติแล้ว Continuous Deployment หมายถึงการที่ Developer สามารถ Dev app ขึ้นมาแล้วไป Go Live บน Cloud ได้โดยใช้เวลาเพียงไม่กี่นาที

- Continuous Delivery มีขั้นตอนการทำงานที่คล้ายกับ Continuous Deployment แต่ไม่รวมการส่งออกโค้ดขึ้นระบบจริง ซึ่งการทำงานในส่วนนี้จะเป็นการทำงานแบบไม่อัตโนมัติ (Manual Process) หรือรูปแบบอื่นตามการทำงานของทีม

CD ที่เป็น Continuous Delivery จะทำหน้าที่ส่ง Code ที่มีการตรวจสอบแล้วไปยัง Repository ดังนั้นถ้าต้องการให้ระบบ Continuous Delivery มีประสิทธิภาพมากที่สุด จะต้องทำระบบ CI ให้อยู่ให้ Pipeline เดียวกันกับ CD เป้าหมายของการทำ CD คือการที่พร้อมที่จะนำ Code ไป Deploy ที่ Production ได้ตลอดเวลาในองครักษ์ แต่ Continuous Delivery จะยังเป็น Step Manual อยู่ ไม่ได้เป็น Automation ทั้งหมด

#### ความแตกต่างของ CI และ CD มีอะไรบ้าง?

- CI จะเป็นระบบ Automate test สำหรับ Developer ถ้าแก้ Code แล้วผ่านระบบ CI ไปได้หมายถึง Code จะ Merge ไปยัง Repository สำเร็จ CI เข้ามาตอบโจทย์ในกรณีที่ Developer มีการ Dev แยกส่วนแล้วต้องนำ Code มารวมกัน ทำให้อาจเกิดปัญหาเรื่อง Conflict กันได้
- CD Continuous Delivery จะหมายถึงการที่ Developer สามารถ Upload Code เพื่อไปทดสอบหา Bug ที่ Repository เช่น GitHub ได้โดยอัตโนมัติ ซึ่งทำให้ทีม Operation สามารถนำไป Deploy ได้เลย ช่วยลดปัญหาเรื่องของการสื่อสารระหว่าง Developer และทีม Business สุดท้ายแล้วจุดประสงค์ของการทำ Continuous Delivery คือการทำให้มั่นใจว่าเราจะใช้แรง หรือความพยายามที่น้อยที่สุดในการ Deploy Code ใหม่ๆ
- CD Continuous Deployment หมายถึง การปล่อย Code จาก Repository ไปยัง Production โดยอัตโนมัติ ซึ่งจะถูกใช้งานโดย user การทำแบบนี้ช่วยลดการทำงานของทีม Operation ในการทำระบบแบบ Manual ซึ่งอาจทำให้การ Deploy ล่าช้าได้

## 2.ทำไมเราถึงต้องใช้ GitLab CI/CD

ทำไมเราถึงต้องใช้ GitLab CI/CD เพราะ CI/CD จะใช้ระบบอัตโนมัติและช่วยมอนิเตอร์ตลอดช่วงเวลาพัฒนา ตั้งแต่การนำ Code ของเหล่า Developer มารวมกัน ไปจนถึงการทดสอบการใช้งานเพื่อที่จะ Deploy ลง Production ขั้นตอนทั้งหมดนี้เรียกว่า CI/CD Pipeline ช่วยให้มีทีม Development และทีม Operation ทำงานด้วยกันได้อย่างต่อเนื่องแบบ Agile คนกลางที่จะมาทำ CI/CD จะเรียกตำแหน่งนี้ว่า DevOps หรือ Site reliability Engineer (SRE)

เป็นการ:

- ลดความยุ่งยากของการ build/test และ deploy
- ลดปัญหาเรื่องการแก้ไข Code – เนื่องจากมีระบบที่คอยนำ Code มารวมกันแล้วทดสอบให้
- ลดปัญหาจากการทำงานแยกส่วนกัน
- ลดระยะเวลาในการส่งมอบงาน – ระบบเป็นผู้ทดสอบให้ ทำให้ไม่ต้องเสียเวลาให้คนมาตรวจสอบแบบ Manual
- เพิ่มความน่าเชื่อถือในการทดสอบ – เนื่องจากเวลาแก้ไข Feature จะแก้ไขแบบเล็ก ๆ ทำให้เวลาเปลี่ยนแปลงระบบ จะทดสอบด้วยเวลาไม่นาน และมีความแม่นยำมากกว่าการเริ่มทำใหม่ตั้งแต่ต้น
- ดูแลง่าย Update ง่าย – การทำ CI/CD ช่วยทำให้สามารถตรวจสอบหาปัญหาได้ในทุก ๆ ขั้นตอน ไม่ว่าจะในช่วงแรกของการ Build หรือการ Update จาก App เดิมที่มีอยู่ โดย Bug ที่เกิดขึ้นจะถูกตรวจสอบได้ง่ายและตรงจุด ทำให้สามารถ Maintenance และ Update ได้ง่าย

### 3.ยกตัวอย่าง usecase

ใช้ gitlab-runner ทำ CI/CD แทน Jenkins เนื่องจากมีหลาย Project ที่ใช้ Jenkins ทำ CI/CD แล้ว มักจะพบกับปัญหาที่ Jenkins มักชอบ Crash ไปโดยไม่บอกไม่กล่าว อีกทั้งปัจจุบันก็ได้ใช้ Gitlab เป็นที่จัดเก็บ Code เป็นส่วนใหญ่อยู่แล้ว

อีกทั้งตัว Gitlab เองก็มีความสามารถในการทำ CI/CD มาให้พร้อมอยู่แล้วนั่นก็คือตัว gitlab-runner

### เครื่องมือสำหรับการทำ CI/CD ที่คนส่วนใหญ่นิยมมีอะไรบ้าง?

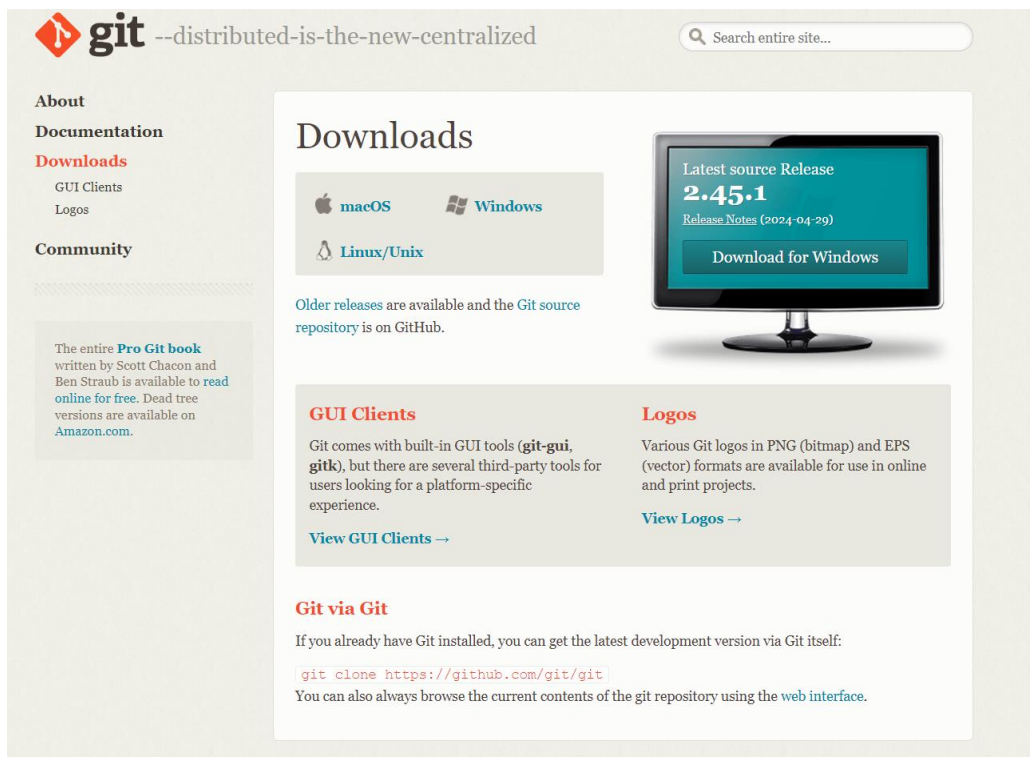
CI/CD tool ที่ช่วยให้ทีมงานสามารถทำ Automate ในเรื่องของการ Develop, Deploy และ Test จะมีอยู่หลากหลาย บางตัวจะเชี่ยวชาญเป็นพิเศษในการทำ CI บางตัวก็ถนัด CD เป็นพิเศษสำหรับเครื่องสำหรับทำ CI/CD ยอดนิยมที่ใช้กันเยอะมาก ๆ คือ Jenkins ครั้นแต่ก็มีเครื่องมืออื่น ๆ ให้ใช้งานได้จากหลากหลายช่องทาง ไม่ว่าจะเป็นจากทาง Public Cloud เจ้าต่าง ๆ หรือจะเป็น GitLab, CircleCI, Travis CI, Atlassian Bamboo หรืออื่น ๆ นอกจากนี้ก็ยังมีบางส่วนที่เป็นเครื่องมือที่ไม่ได้ครอบคลุมระบบ CI/CD ทั้งหมด แต่เป็นเครื่องมือสำหรับ Config การทำ Automation เช่น Ansible, Chef และ Puppet หรือจะเป็นเครื่องมือที่ใช้สำหรับทำ Container เช่น Docker และ rkt และตัวจัดการ Container เช่น Kubernetes

## 4.Demo

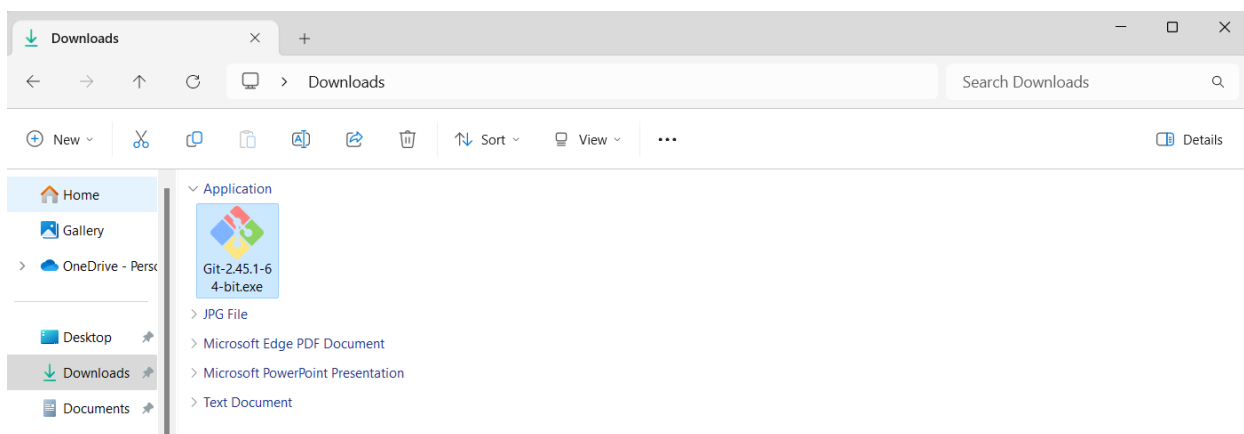
### 4.1 ติดตั้ง git

จาก <https://git-scm.com/downloads>

เลือก version และ ระบบปฏิบัติการ ให้เหมาะสมกับเครื่องของตัวเอง

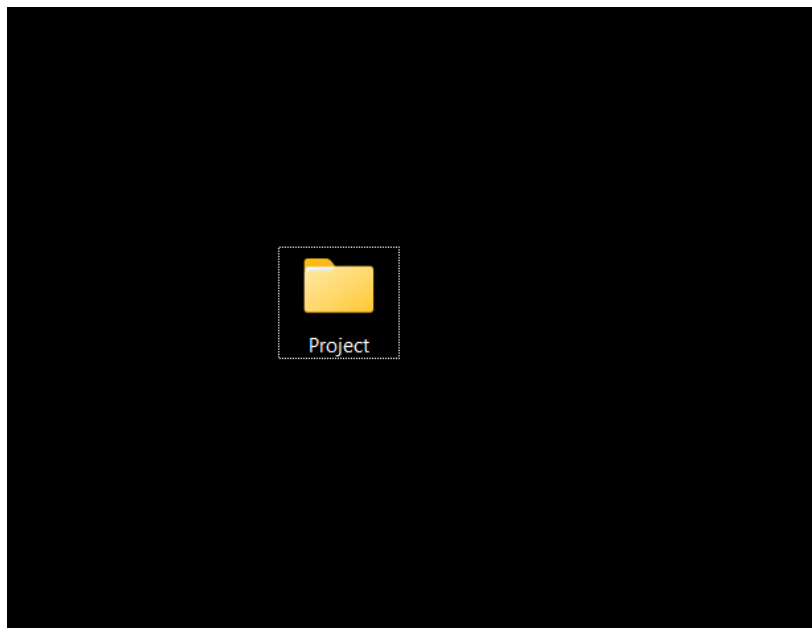


ทำการ Install file git.exe

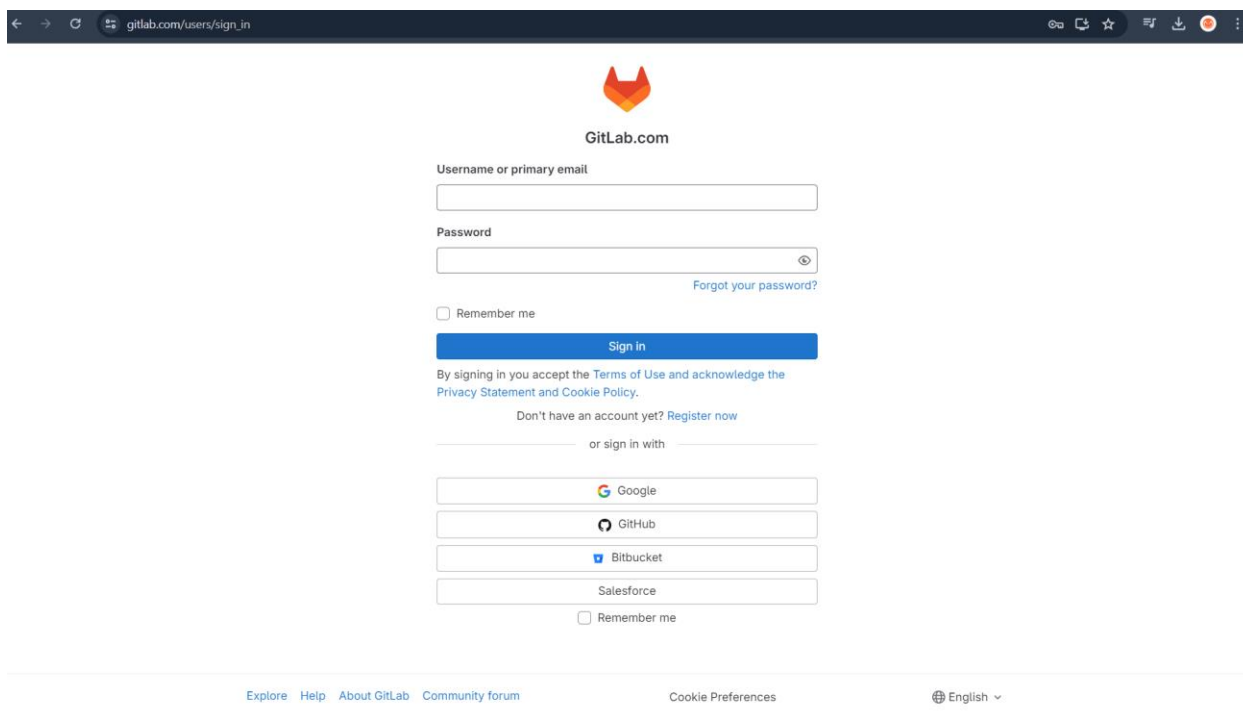




สร้าง folder project



#### 4.2 สมัคร gitlab [https://gitlab.com/users/sign\\_in](https://gitlab.com/users/sign_in)

A screenshot of the GitLab.com sign-in page. The browser address bar shows "gitlab.com/users/sign\_in". The page features the GitLab logo (a red fox head) and the text "GitLab.com". Below this, there are input fields for "Username or primary email" and "Password". A "Remember me" checkbox is present. A blue "Sign in" button is located below the password field. To the right of the password field is a link "Forgot your password?". Below the "Sign in" button, there is a line of text: "By signing in you accept the Terms of Use and acknowledge the Privacy Statement and Cookie Policy." followed by a link "Don't have an account yet? Register now". Below this, there is a section "or sign in with" with four buttons: "Google", "GitHub", "Bitbucket", and "Salesforce". A "Remember me" checkbox is also present at the bottom of this section. The footer contains links for "Explore", "Help", "About GitLab", and "Community forum", along with "Cookie Preferences" and a language selector set to "English".

### 4.3 ทำการ config username and email gitlab

ผ่าน Administrator Command Prompt ด้วยคำสั่ง

- git config --global user.name "....."
- git config --global user.email "....."

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.22631.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Users\supar\OneDrive\Desktop\Project\GitLab\keycloak2\demokeycloak

C:\Users\supar\OneDrive\Desktop\Project\GitLab\keycloak2\demokeycloak>git config --global user.name "msprarmmers"

C:\Users\supar\OneDrive\Desktop\Project\GitLab\keycloak2\demokeycloak>git config --global user.email "msprarmmers@gmail.com"
```

สามารถเช็ค username and email ที่ config ไปได้ ด้วยคำสั่ง

git config --list

```
Administrator: Command Prompt

C:\Users\supar\OneDrive\Desktop\Project\GitLab\keycloak2\demokeycloak>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=msprarmmers
user.email=msprarmmers@gmail.com

C:\Users\supar\OneDrive\Desktop\Project\GitLab\keycloak2\demokeycloak>
```

ติดตั้ง file .git ด้วยคำสั่ง

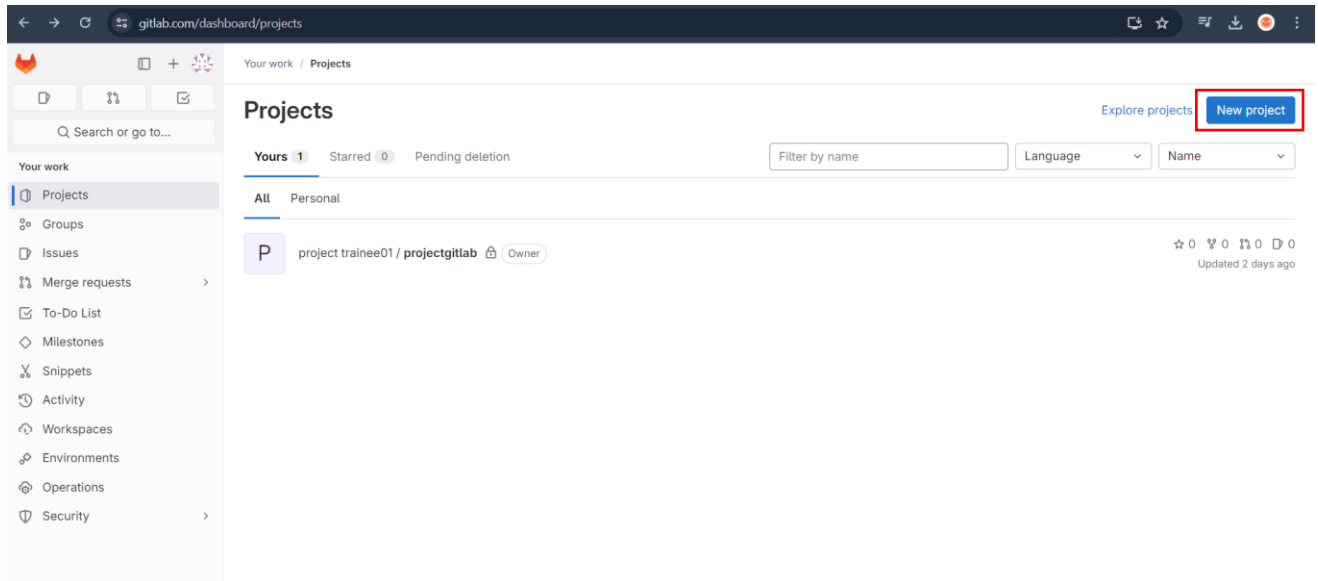
git init

```
C:\Users\supar\OneDrive\Desktop\Project\GitLab\keycloak2\demokeycloak>git init
Initialized empty Git repository in C:/Users/supar/OneDrive/Desktop/Project/GitLab/keycloak2/demokeycloak/.git/

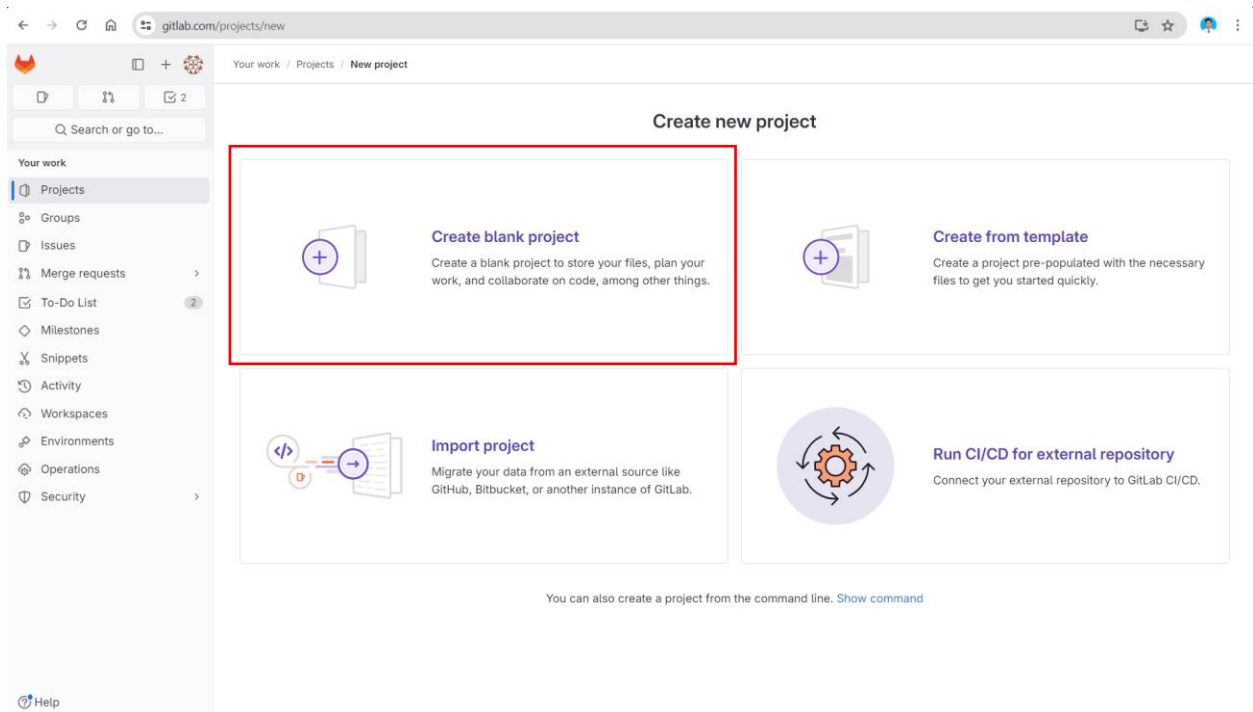
C:\Users\supar\OneDrive\Desktop\Project\GitLab\keycloak2\demokeycloak>
```

Name	Date modified	Type	Size
.git	5/23/2024 10:42 AM	File folder	

## 4.4 สร้าง project ใน gitlab



เลือก Create blank project



## ตั้งชื่อโปรเจกต์ เสร็จแล้วกด Create project

gitlab.com/projects/new#blank\_project

Your work / Projects / New project / Create blank project

### Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

**Project name**

My awesome project

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

**Project URL**

https://gitlab.com/trainee01

**Project slug**

my-awesome-project

**Project deployment target (optional)**

Select the deployment target

**Visibility Level**

☒ Private  
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

☐ Internal  
The project can be accessed by any logged in user except external users.

☐ Public  
The project can be accessed without any authentication.

**Project Configuration**

☒ Initialize repository with a README  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

☐ Enable Static Application Security Testing (SAST)  
Analyze your source code for known security vulnerabilities. [Learn more.](#)

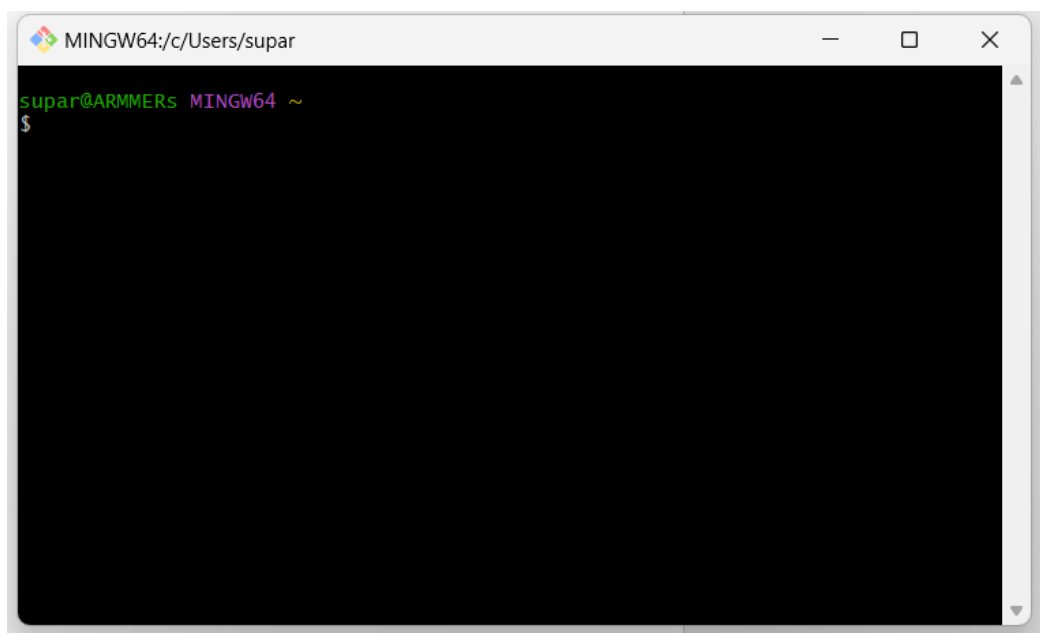
**Create project** Cancel

gitlab.com

<

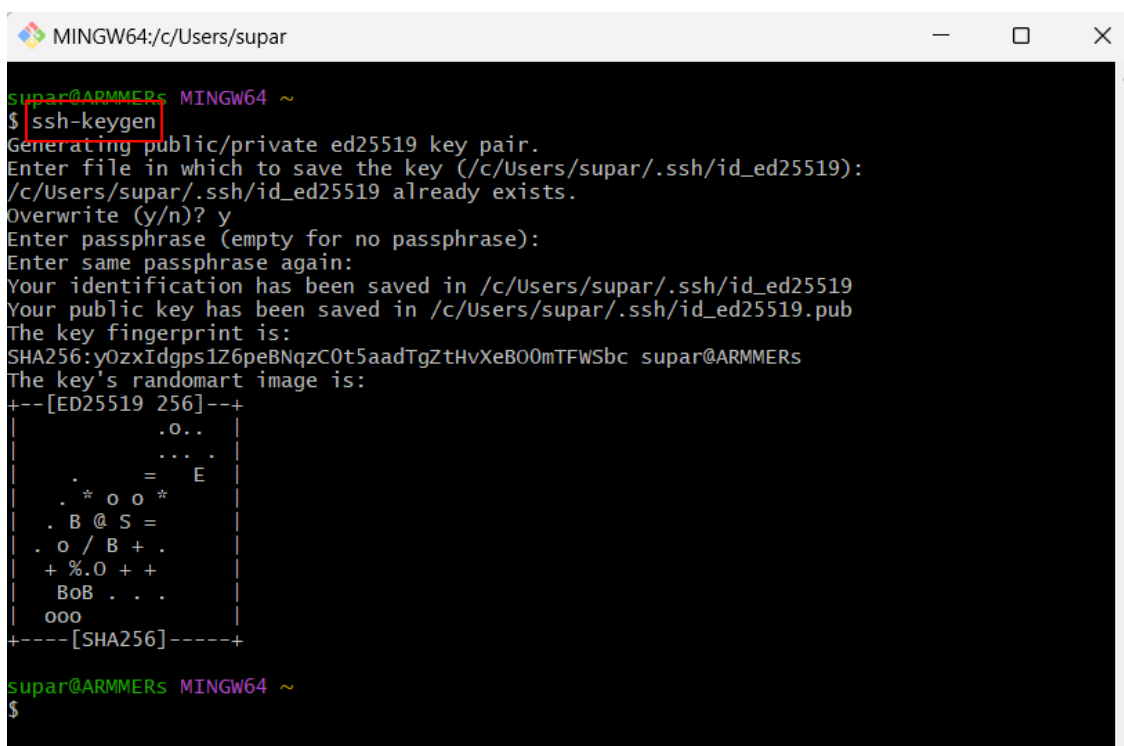
#### 4.5 ทำการ set up ssh key ใน gitlab

เปิด Git Bash app

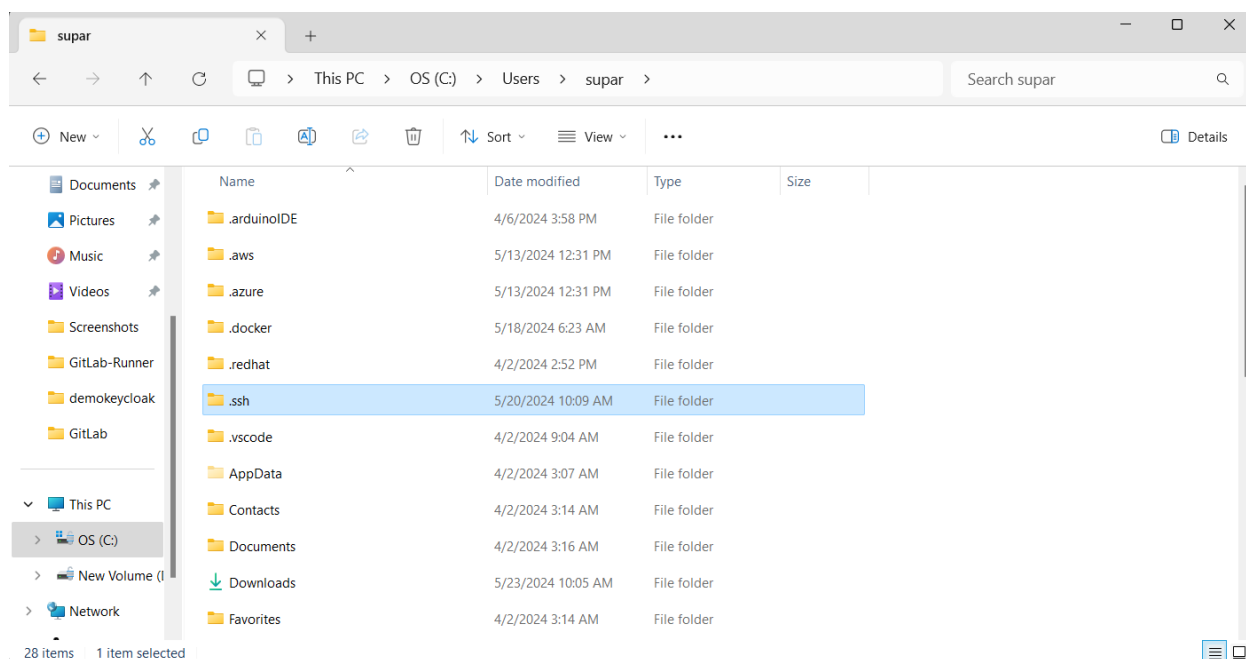


จากนั้นใช้คำสั่ง

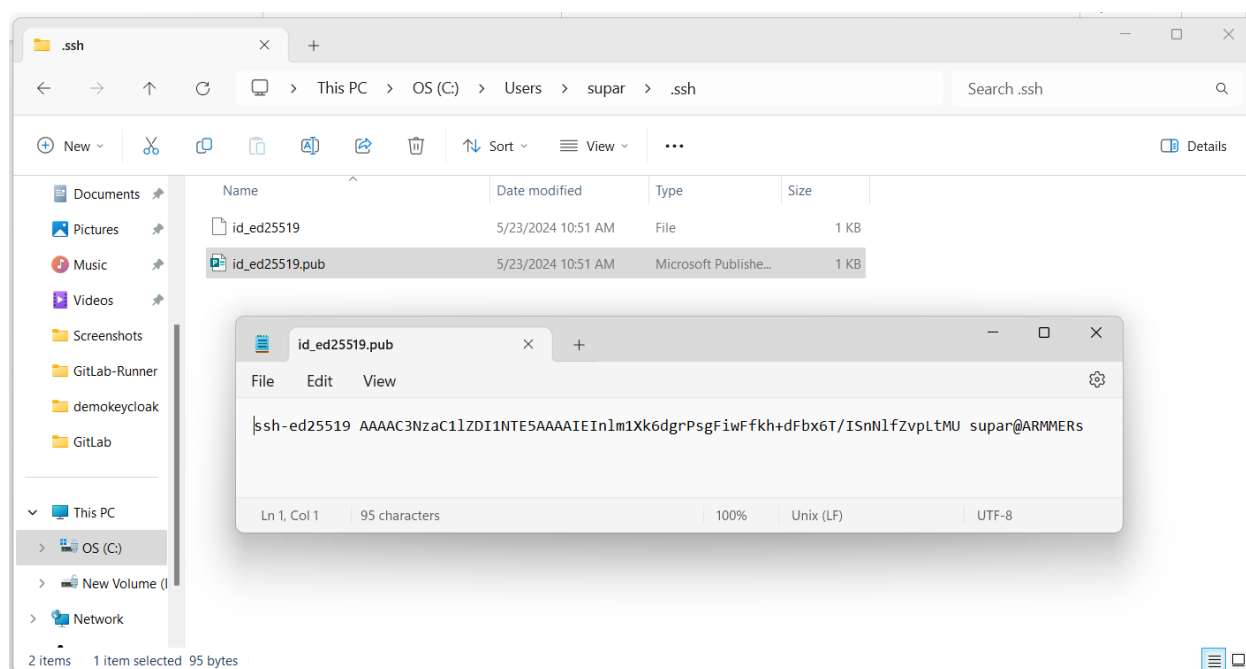
ssh-keygen



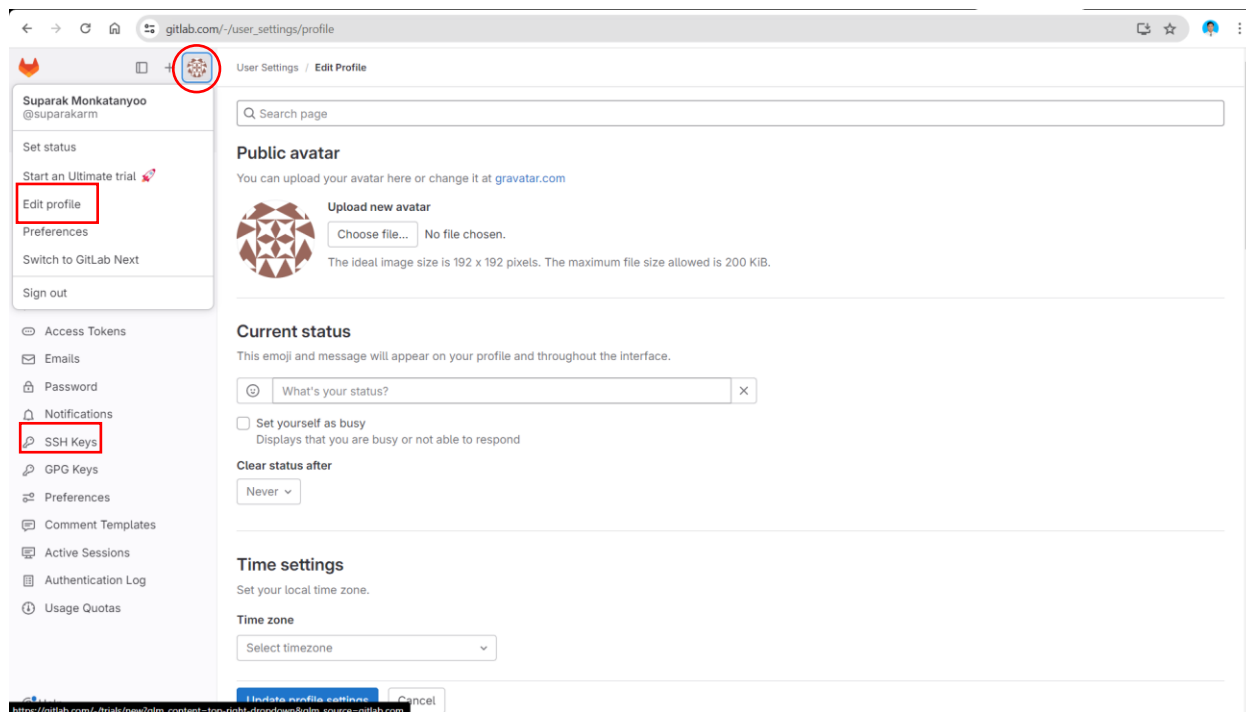
Folder .ssh จะถูกสร้างไว้ที่ C:



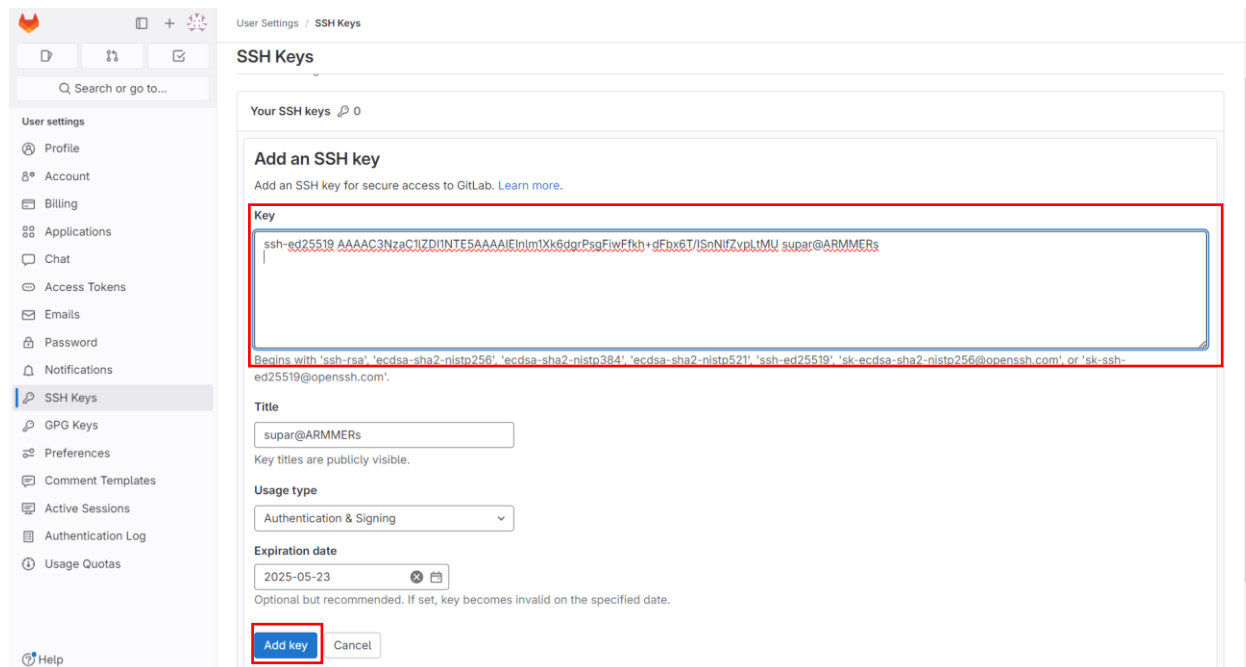
เปิด file id\_ed25519.pub ด้วย notepad จากนั้น copy code นำไปวางใน gitlab



ไปที่ Profile -> Edit profile -> SSH Keys

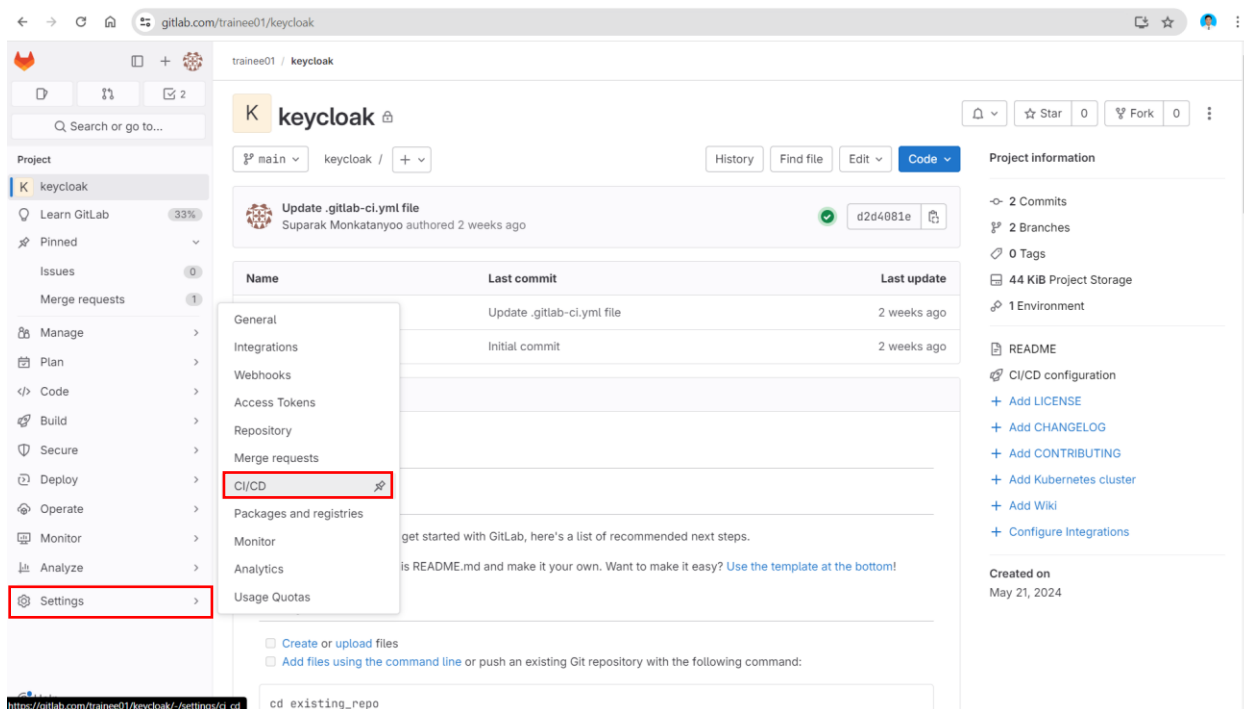


ใส่ code ที่ copy ลงในช่อง key เสร็จแล้ว Add key

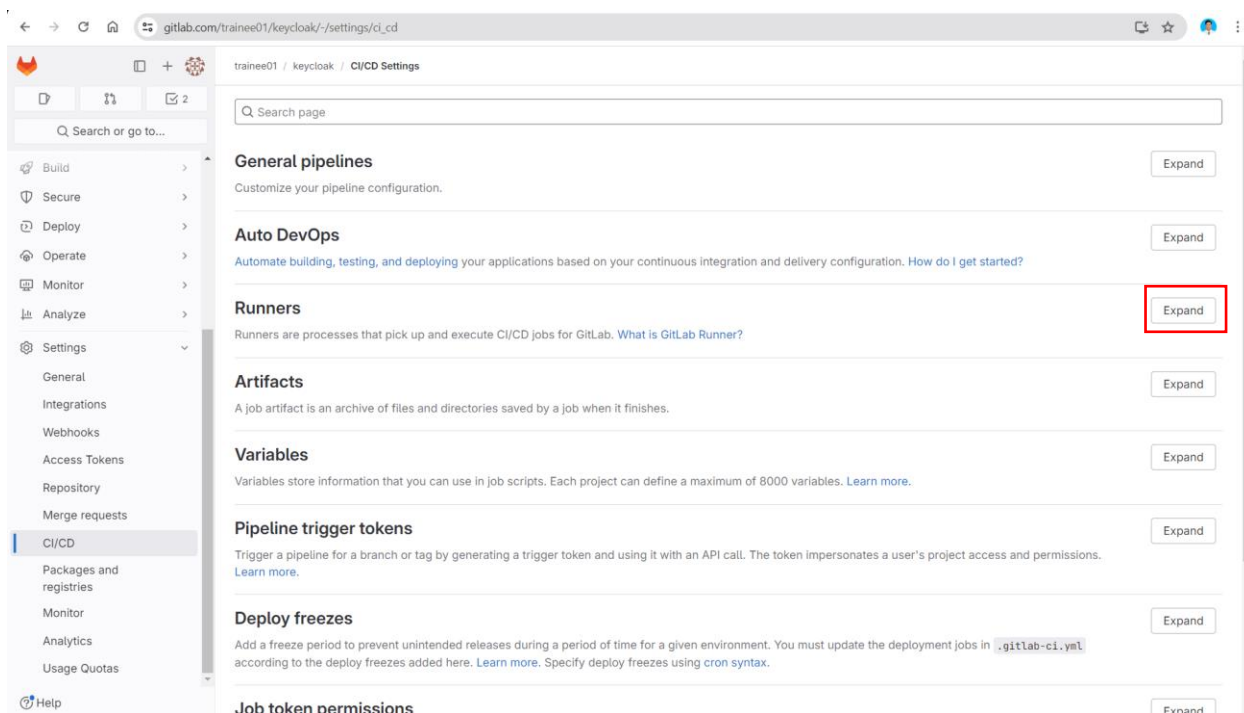


## 4.6 สร้าง Project runner

ไปที่ Settings -> CI/CD

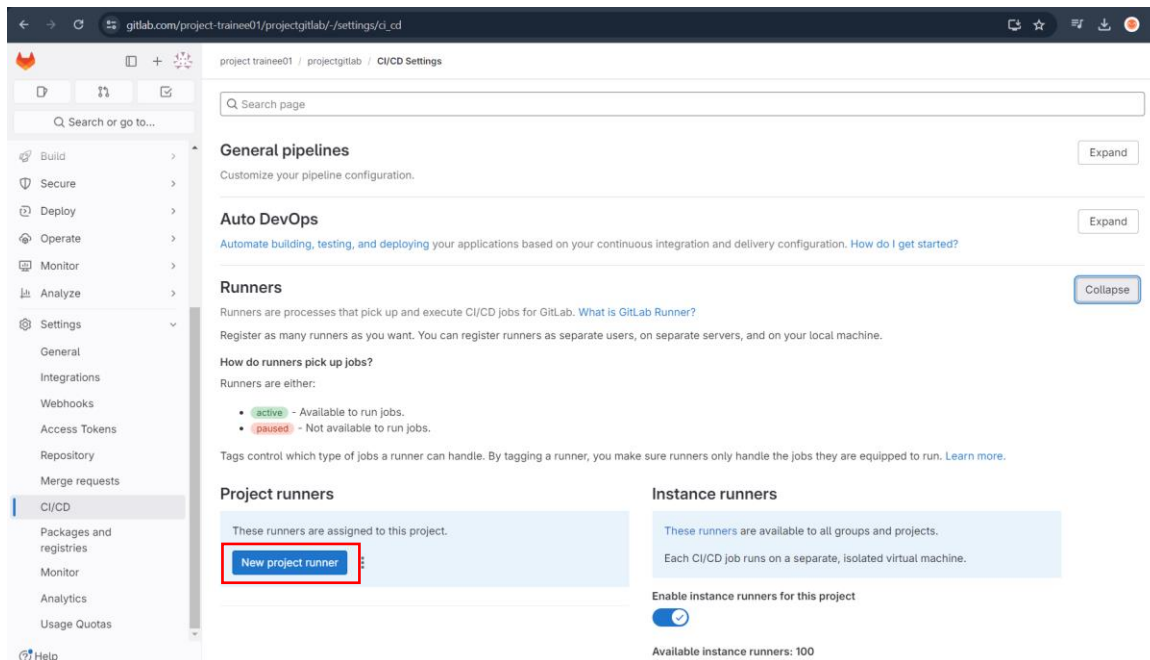


กด Expand ที่ Runners



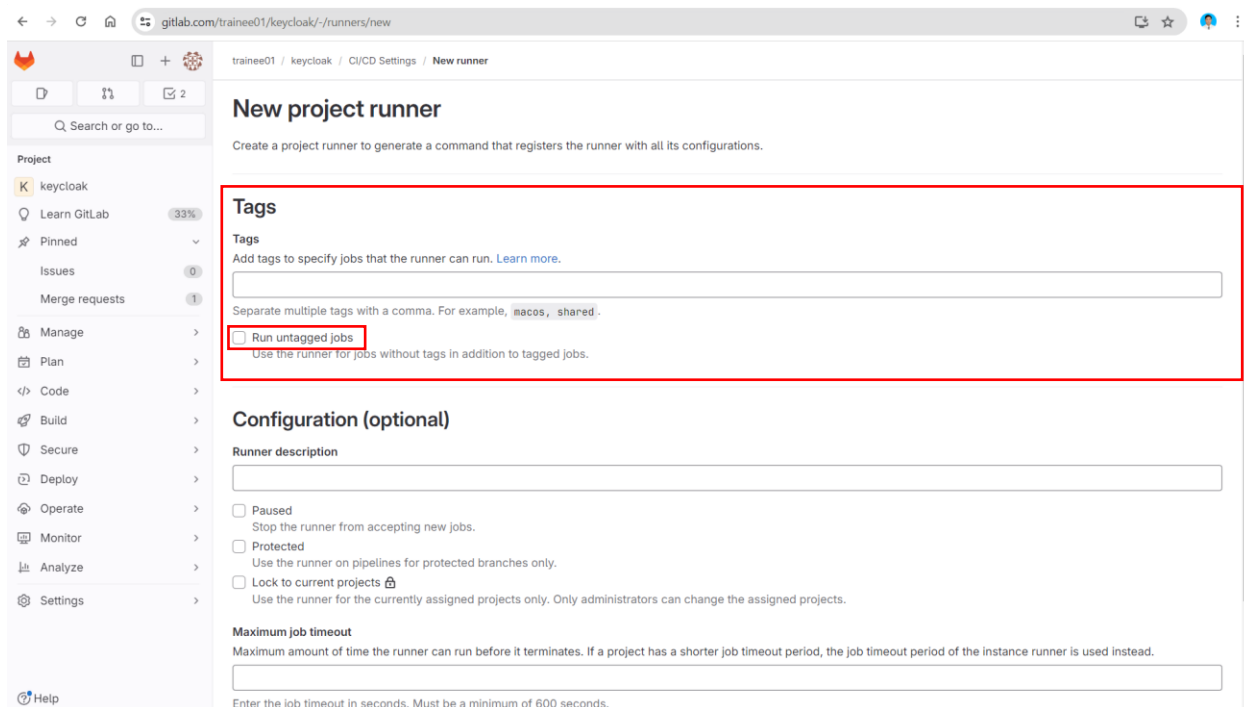


เลือก New project runner เพื่อสร้างโปรเจกต์ runner

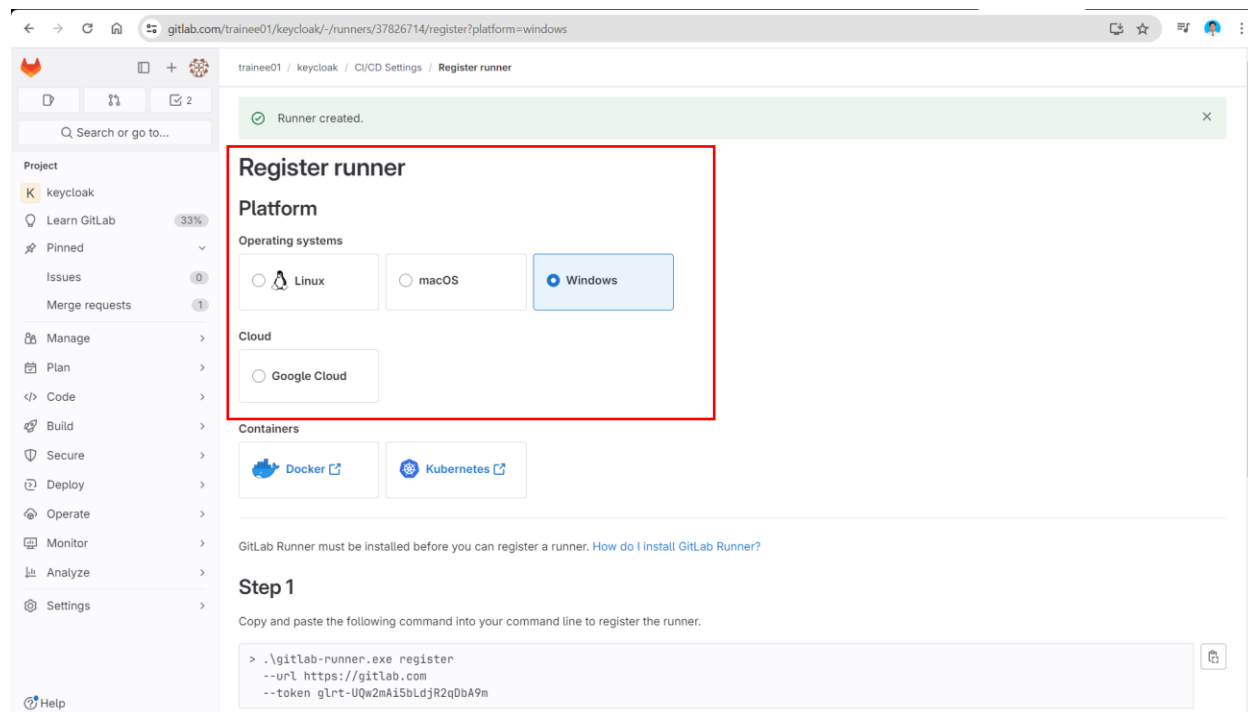


ใส่ Tags ให้ Project runner หรือถ้าไม่ต้องการใส่ Tags ให้เลือกที่ Run untagged jobs

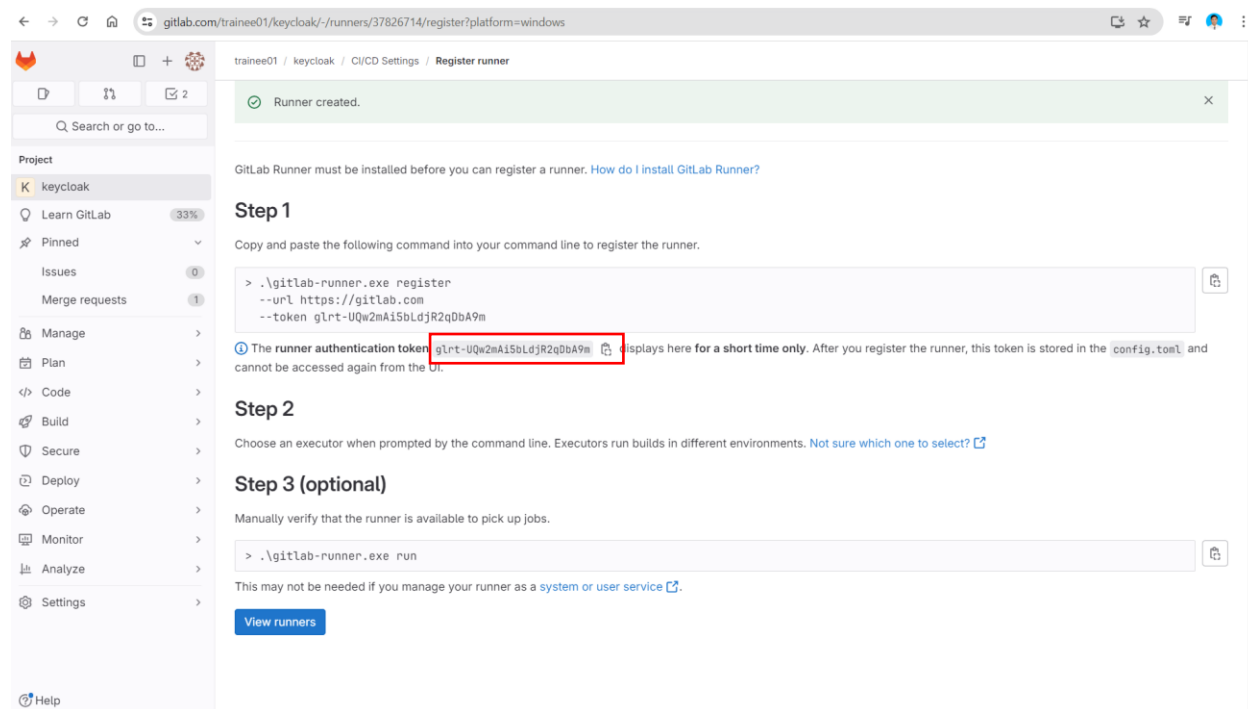
เสร็จแล้วกด Create project



เลือก systems ของตัวเอง



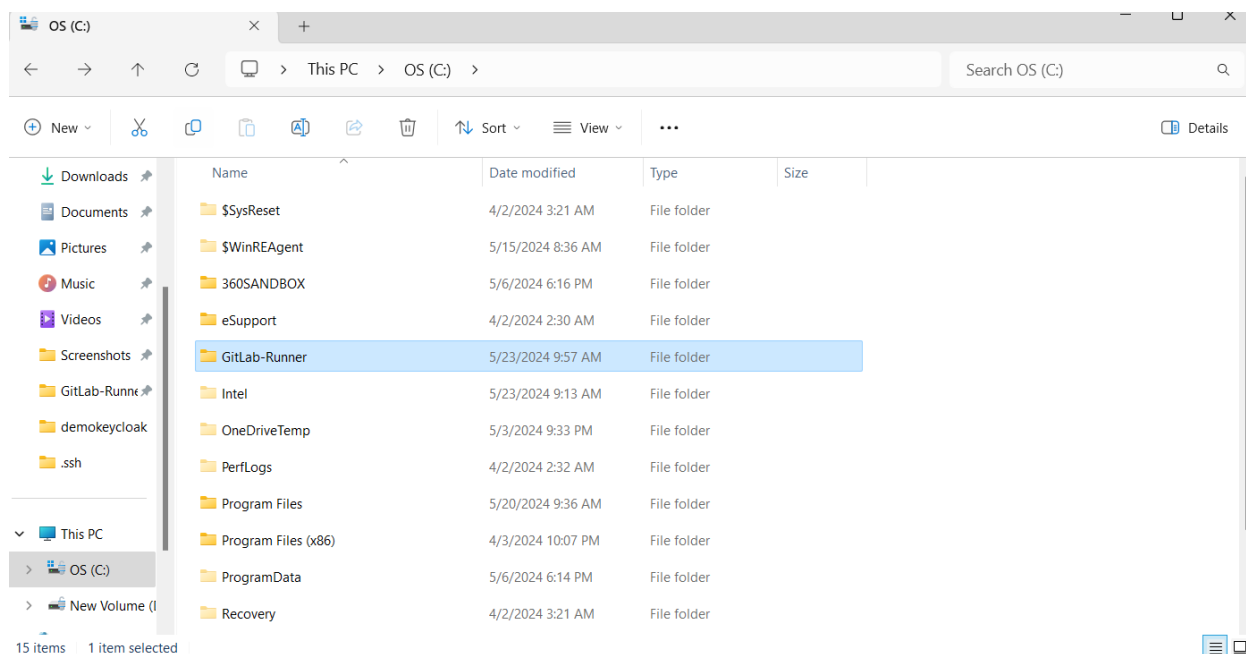
Copy Token เก็บไว้สำหรับ Register เสร็จแล้วกด View runners



4.7 ทำการ install gitlab-runner จาก <https://docs.gitlab.com/runner/install/windows.html>

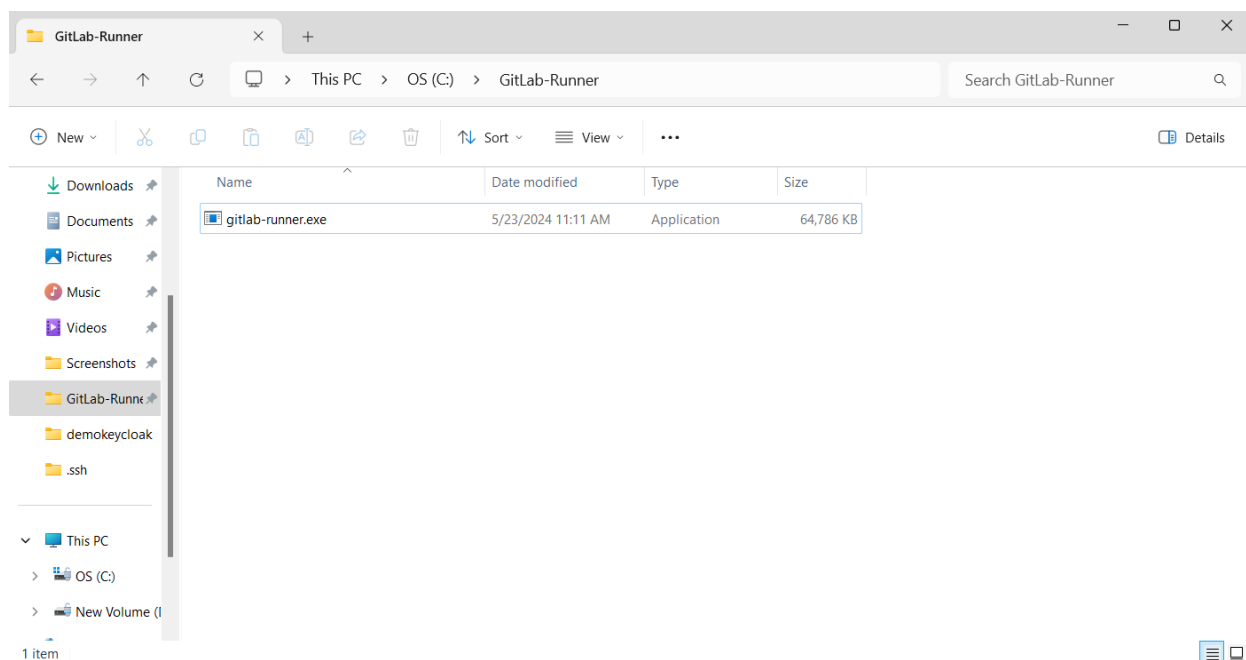
Download binary

สร้าง folder GitLab-Runner ใน C:



นำ file binary ที่โหลดไว้ ไปใส่ใน folder GitLab-Runner ที่สร้างไว้

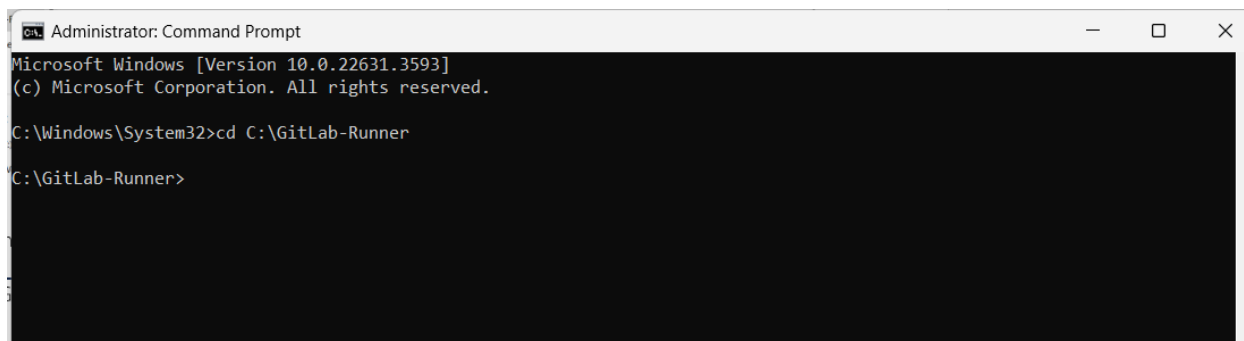
ทำการเปลี่ยนชื่อ file เป็น gitlab-runner.exe



ทำการ install gitlab-runner ผ่าน Administrator Command Prompt ด้วยคำสั่ง

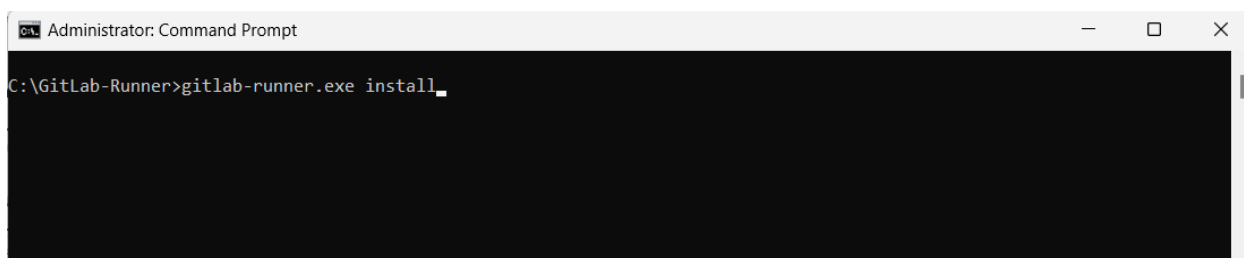
```
/cd C:\GitLab-Runner
```

```
/gitlab-runner.exe install
```



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.22631.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\GitLab-Runner
C:\GitLab-Runner>
```

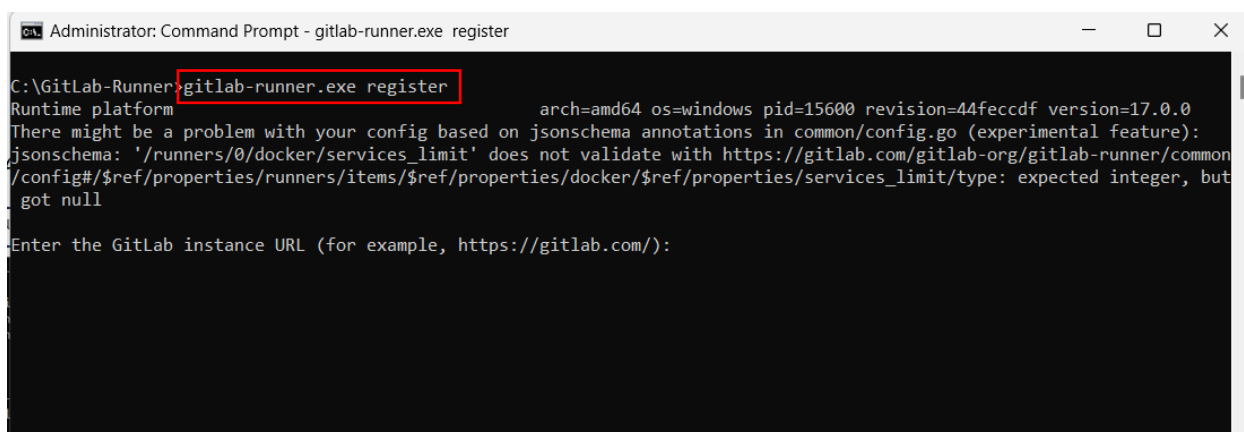


```
Administrator: Command Prompt

C:\GitLab-Runner>gitlab-runner.exe install_
```

#### 4.8 ทำการ register ด้วยคำสั่ง gitlab-runner.exe register

```
gitlab-runner.exe register
```



```
Administrator: Command Prompt - gitlab-runner.exe register

C:\GitLab-Runner>gitlab-runner.exe register
Runtime platform arch=amd64 os=windows pid=15600 revision=44fecddf version=17.0.0
There might be a problem with your config based on jsonschema annotations in common/config.go (experimental feature):
jsonschema: '/runners/0/docker/services_limit' does not validate with https://gitlab.com/gitlab-org/gitlab-runner/common
/config#/$ref/properties/runners/items/$ref/properties/docker/$ref/properties/services_limit/type: expected integer, but
got null

Enter the GitLab instance URL (for example, https://gitlab.com/):
```

ใส่ URL ของ gitlab

```
Administrator: Command Prompt - gitlab-runner.exe register

C:\GitLab-Runner>gitlab-runner.exe register
Runtime platform arch=amd64 os=windows pid=15600 revision=44fecddf version=17.0.0
There might be a problem with your config based on jsonschema annotations in common/config.go (experimental feature):
jsonschema: '/runners/0/docker/services_limit' does not validate with https://gitlab.com/gitlab-org/gitlab-runner/common
/config#/$ref/properties/runners/items/$ref/properties/docker/$ref/properties/services_limit/type: expected integer, but
got null

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.com
```

นำ Token ที่ copy ใ้มาใส่

```
Administrator: Command Prompt - gitlab-runner.exe register

C:\GitLab-Runner>gitlab-runner.exe register
Runtime platform arch=amd64 os=windows pid=15600 revision=44fecddf version=17.0.0
There might be a problem with your config based on jsonschema annotations in common/config.go (experimental feature):
jsonschema: '/runners/0/docker/services_limit' does not validate with https://gitlab.com/gitlab-org/gitlab-runner/common
/config#/$ref/properties/runners/items/$ref/properties/docker/$ref/properties/services_limit/type: expected integer, but
got null

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.com
Enter the registration token:
glrt-UQw2mAi5bLdjR2qDbA9m
```

ตั้งชื่อให้ runner

```
Administrator: Command Prompt - gitlab-runner.exe register

C:\GitLab-Runner>gitlab-runner.exe register
Runtime platform arch=amd64 os=windows pid=15600 revision=44fecddf version=17.0.0
There might be a problem with your config based on jsonschema annotations in common/config.go (experimental feature):
jsonschema: '/runners/0/docker/services_limit' does not validate with https://gitlab.com/gitlab-org/gitlab-runner/common
/config#/$ref/properties/runners/items/$ref/properties/docker/$ref/properties/services_limit/type: expected integer, but
got null

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.com
Enter the registration token:
glrt-UQw2mAi5bLdjR2qDbA9m
Verifying runner... is valid runner=UQw2mAi5b
Enter a name for the runner. This is stored only in the local config.toml file:
[ARMMEs]: project1
```

เลือกตัวดำเนินการที่ใช้ เช่น docker-windows

```
Administrator: Command Prompt - gitlab-runner.exe register

C:\GitLab-Runner>gitlab-runner.exe register
Runtime platform arch=amd64 os=windows pid=15600 revision=44fecddf version=17.0.0
There might be a problem with your config based on jsonschema annotations in common/config.go (experimental feature):
jsonschema: '/runners/0/docker/services_limit' does not validate with https://gitlab.com/gitlab-org/gitlab-runner/common
/config#/$ref/properties/runners/items/$ref/properties/docker/$ref/properties/services_limit/type: expected integer, but
got null

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.com
Enter the registration token:
glrt-UQw2mAi5bLdjR2qDbA9m
Verifying runner... is valid runner=UQw2mAi5b
Enter a name for the runner. This is stored only in the local config.toml file:
[ARMMEs]: project1
Enter an executor: parallels, virtualbox, docker, docker+machine, kubernetes, custom, shell, ssh, instance, docker-windo
ws, docker-autoscaler:
docker-windows
```

ใส่ mcr.microsoft.com/windows/servercore:1809

```
Administrator: Command Prompt - gitlab-runner.exe register

C:\GitLab-Runner>gitlab-runner.exe register
Runtime platform arch=amd64 os=windows pid=15600 revision=44fecddf version=17.0.0
There might be a problem with your config based on jsonschema annotations in common/config.go (experimental feature):
jsonschema: '/runners/0/docker/services_limit' does not validate with https://gitlab.com/gitlab-org/gitlab-runner/common
/config#/$ref/properties/runners/items/$ref/properties/docker/$ref/properties/services_limit/type: expected integer, but
got null

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.com
Enter the registration token:
glrt-UQw2mAi5bLdjR2qDbA9m
Verifying runner... is valid runner=UQw2mAi5b
Enter a name for the runner. This is stored only in the local config.toml file:
[ARMMEs]: project1
Enter an executor: parallels, virtualbox, docker, docker+machine, kubernetes, custom, shell, ssh, instance, docker-windo
ws, docker-autoscaler:
docker-windows
Enter the default Docker image (for example, mcr.microsoft.com/windows/servercore:1809):
mcr.microsoft.com/windows/servercore:1809
```

ใช้คำสั่ง gitlab-runner.exe start เพื่อเปิดการทำงานของ gitlab-runner

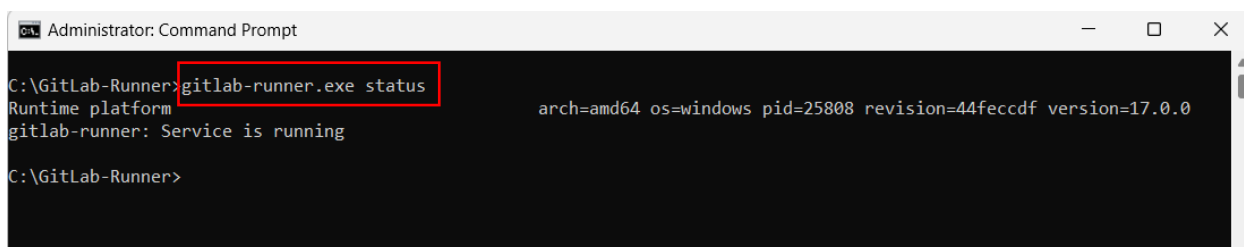
gitlab-runner.exe start

```
:\GitLab-Runner>gitlab-runner.exe start
untime platform arch=amd64 os=windows pid=12280 revision=44fecddf version=17.0.0

:\GitLab-Runner>
```

หากต้องการรู้ว่า runner ได้ทำงานอยู่หรือไม่ สามารถใช้คำสั่ง

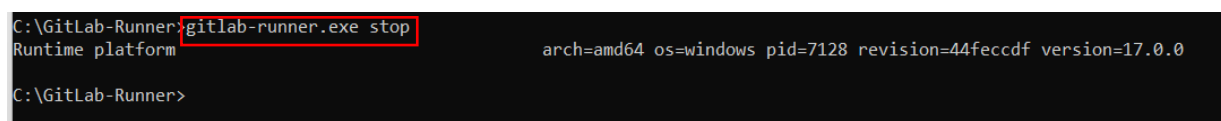
```
gitlab-runner.exe status
```



```
Administrator: Command Prompt
C:\GitLab-Runner>gitlab-runner.exe status
Runtime platform arch=amd64 os=windows pid=25808 revision=44feccdf version=17.0.0
gitlab-runner: Service is running
C:\GitLab-Runner>
```

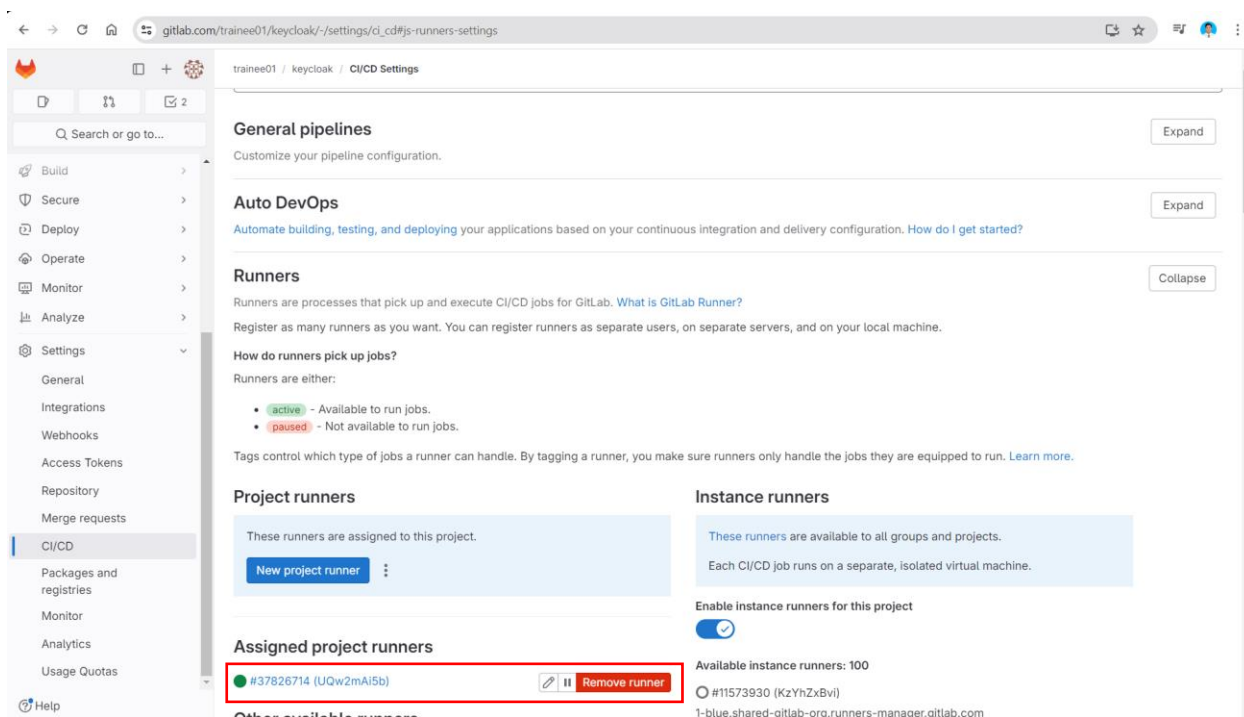
หากต้องการหยุดการทำงานของ runner สามารถใช้คำสั่ง

```
gitlab-runner.exe stop
```



```
C:\GitLab-Runner>gitlab-runner.exe stop
Runtime platform arch=amd64 os=windows pid=7128 revision=44feccdf version=17.0.0
C:\GitLab-Runner>
```

เมื่อ gitlab-runner พร้อมใช้งานแล้วจะขึ้นสีเขียว ดังรูป

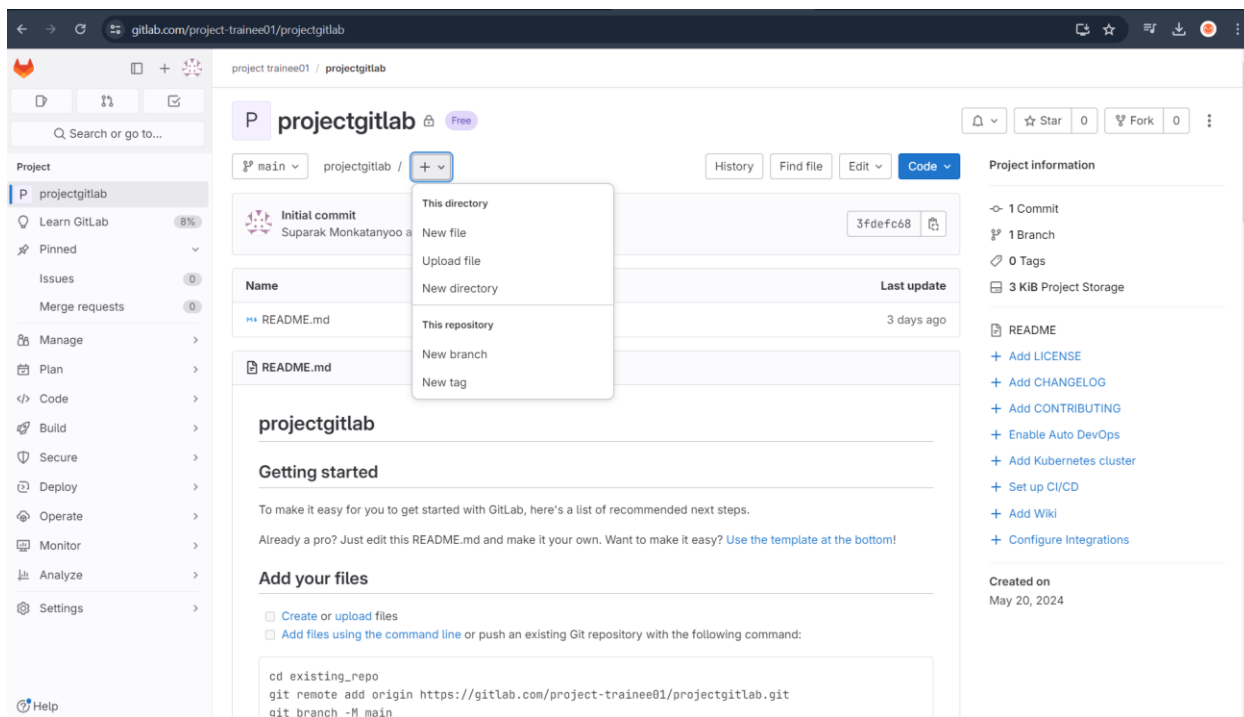




## การ Upload file

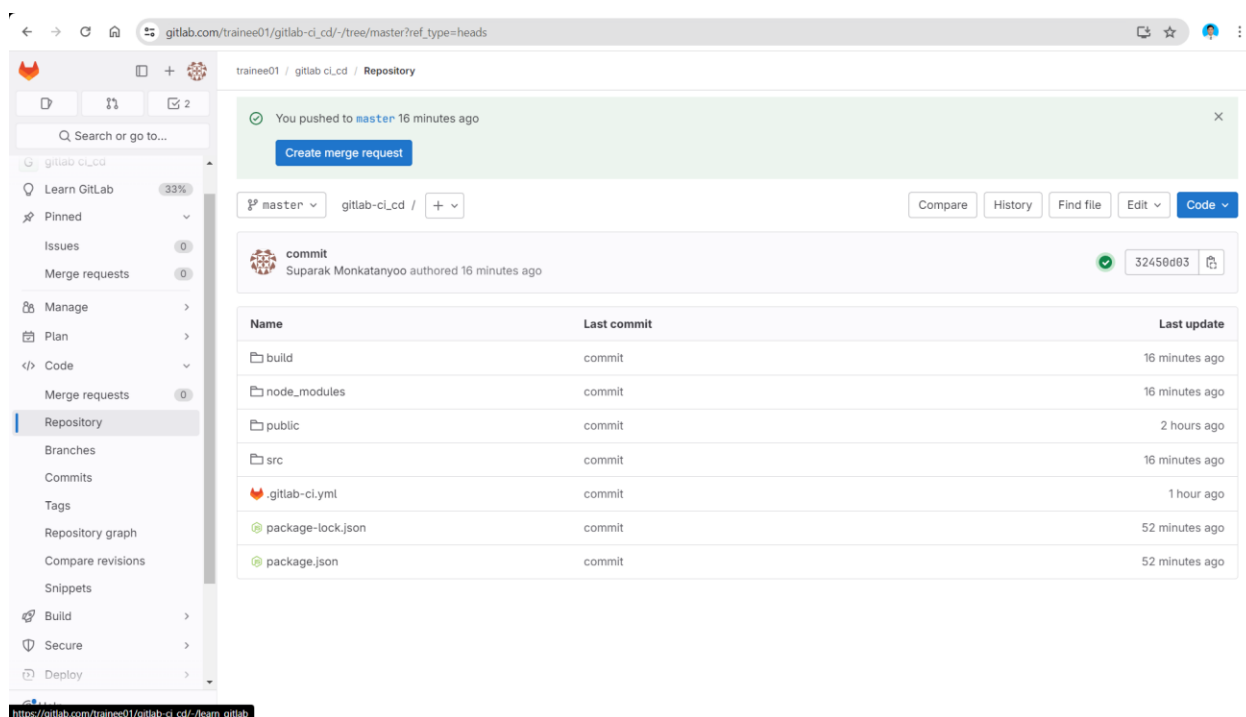
Upload file project ไปยัง gitlab มี 2 วิธี คือ

### 1.ผ่านตัว gitlab โดยตรง



## 2.ผ่าน Administrator Command Prompt โดยใช้คำสั่ง

- git init = เพื่อติดตั้ง file .git สำหรับใช้คำสั่งต่างๆ
- git status = เช็คสถานะของ file ใน folder ว่ามีการเปลี่ยนแปลงหรือแก้ไขอะไรหรือไม่
- git add . = git add คือ คำสั่งสำหรับเพิ่ม file หรือ folder project เราลงไปในส่วนที่เตรียมไว้สำหรับการ commit (Staging Area)
- git commit -m “...” = เป็นคำสั่งทำการยืนยันการเปลี่ยนแปลงจากการ add. หรือทำการ update HEAD ที่ local repository
- git push -u “URL” master = เป็นการส่ง commit ที่อยู่ใน local ไปยัง URL ที่เราต้องการ



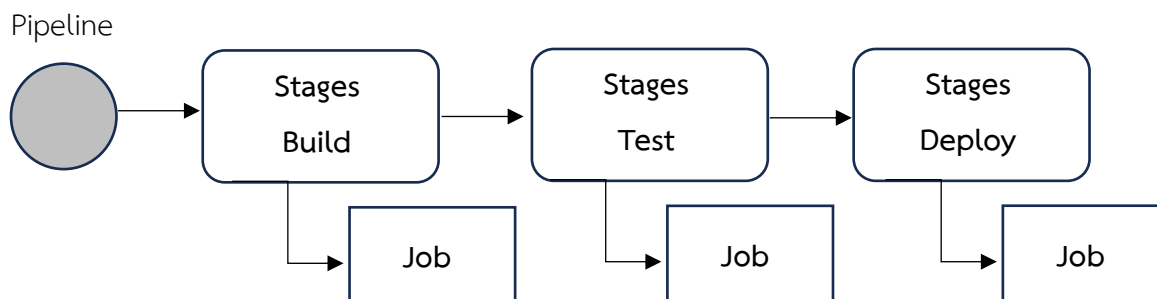
## .gitlab-ci.yml

### .gitlab-ci.yml คืออะไร

.gitlab-ci.yml เป็นไฟล์ YAML ที่กำหนดขั้นตอนที่ GitLab CI จะใช้ในการสร้าง ทดสอบ และส่งมอบของเราได้ดีขึ้น มันเป็นเครื่องมือที่มีประสิทธิภาพที่สามารถช่วยให้เราพัฒนาและปรับใช้กระบวนการโดยอัตโนมัติ และสามารถช่วยปรับปรุงคุณภาพและความน่าเชื่อถือของโค้ดเราได้

### .gitlab-ci.yml ทำงานยังไง?

เมื่อมีการ Push commit ใหม่ขึ้นไปใน Gitlab repo และมีไฟล์ .gitlab-ci.yml อยู่ มันจะทำการอ่านไฟล์ .gitlab-ci.yml ให้ทันที และดำเนินการตามขั้นตอนที่กำหนดไว้ในนั้น ขั้นตอนต่างๆ จะถูกจัดระเบียบเป็นขั้นๆ และแต่ละขั้นสามารถบรรจุได้หลายงาน งานจะดำเนินการพร้อมกัน และแต่ละงานสามารถมีชุดขั้นตอนของตัวเองได้ด้วย ซึ่งลำดับขั้นใน GitLab มี Pipeline > Stages > Jobs



### ความแตกต่างของ Stages

- **build:** ขั้นตอนนี้รับผิดชอบในการสร้าง project ของคุณ ซึ่งอาจเกี่ยวข้องกับการ compile code รันการทดสอบ หรือสร้างอิมเมจ Docker
- **test:** ขั้นตอนนี้รับผิดชอบในการ test project ของคุณ ซึ่งอาจเกี่ยวข้องกับการทดสอบหน่วยการทำงาน การทดสอบการรวมระบบ หรือการทดสอบตั้งแต่ต้นทางถึงปลายทาง
- **deploy:** ขั้นตอนนี้ส่งมอบงานของเราไปยังที่ต่างๆ เช่น Hosting, K8S หรือ Google cloud

Job คือ หน่วยที่เล็กที่สุดในไฟล์ .gitlab-ci.yml แต่ละงานสามารถมีชุดขั้นตอนของตัวเองได้ และแต่ละขั้นตอนสามารถเป็นคำสั่ง shell script หรืออิมเมจ Docker แบบกำหนดเองได้

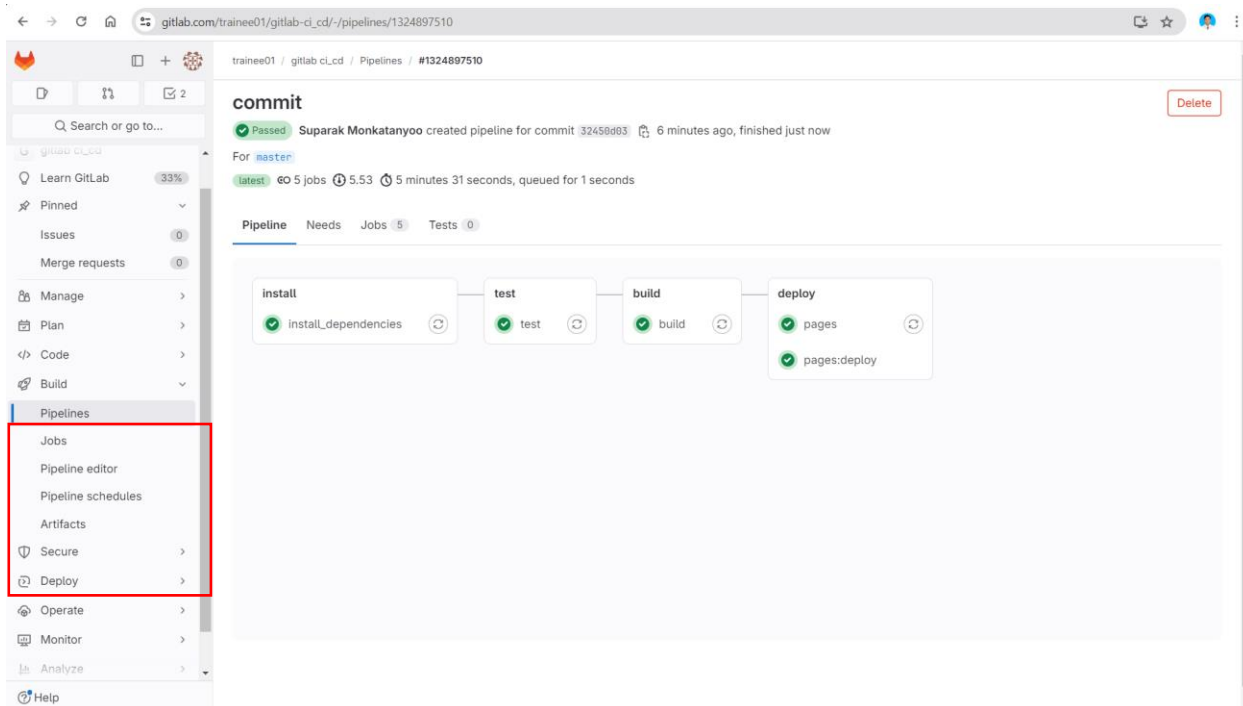
## ตัวอย่าง Code

```
image: node:14
stages:
  - install
  - test
  - build
  - deploy
cache:
  key: ${CI_COMMIT_REF_SLUG}
  paths:
    - node_modules/
install_dependencies:
  stage: install
  script:
    - npm install
  artifacts:
    paths:
      - node_modules/
test:
  stage: test
  script:
    - npm install
    - npm test
  artifacts:
    when: always
    reports:
      junit: report.xml
build:
  stage: build
  script:
    - npm install
    - npm run build
  artifacts:
    paths:
      - build/
    expire_in: 1 week
pages:
  stage: deploy
  script:
    - mv build public
  artifacts:
    paths:
      - public
only:
  - master
```

อ่านเพิ่มเติมเกี่ยวกับ .gitlab-ci.yml ได้ที่ <https://codinggun.com/gitlab/gitlab-ci/>

### การตรวจสอบสถานะ Pipeline

ไปที่ Build > Pipeline จากเมนูด้านซ้าย คุณสามารถตรวจสอบสถานะของ pipeline ได้ในหน้านี้ ที่นี้ คุณสามารถตรวจสอบ commit ID, branch, ผู้ใช้ที่กด pipeline, stage และสถานะต่างๆ



เช็ค syntax .yaml <https://www.yamllint.com/>