

# DevOps with Gitlab, Jenkins and Kubernetes

**Mr. Sommai Krangpanich**

[www.pnpsw.com](http://www.pnpsw.com)

081-754-4663

[sommai.k@gmail.com](mailto:sommai.k@gmail.com)

Line Id : sommai.k

Blog : <https://medium.com/@sommaikrangpanich>



# สมหมาย กรังพาณิช

## ตำแหน่งปัจจุบัน

- กรรมการสมาคมอุตสาหกรรมซอฟต์แวร์ไทย ATSI
- กรรมการผู้จัดการบริษัท พี เอ็น พี โซลูชั่น จำกัด

## ด้านความเชี่ยวชาญ

- ผู้เชี่ยวชาญด้านการออกแบบและพัฒนา Software ด้วย Framework ดังนี้
  - Java, Springboot, Go, Node.js
  - Full Stack, Angular, Vue, React, Svelte
  - Apollo GraphQL
  - Docker, Docker Swarm, K8s
  - Flutter, Dart
  - MongoDB, Oracle, MySQL, SQL Server, DB2, PostgreSQL
- ผู้เชี่ยวชาญด้านการออกแบบและประยุกต์ใช้ DevOps ในการพัฒนา Software
- ผู้เชี่ยวชาญด้านการออกแบบและประยุกต์ใช้ Cloud Native ในการพัฒนา Software
- ผู้เชี่ยวชาญด้านการออกแบบและพัฒนาระบบงานในรูปแบบ Microservice

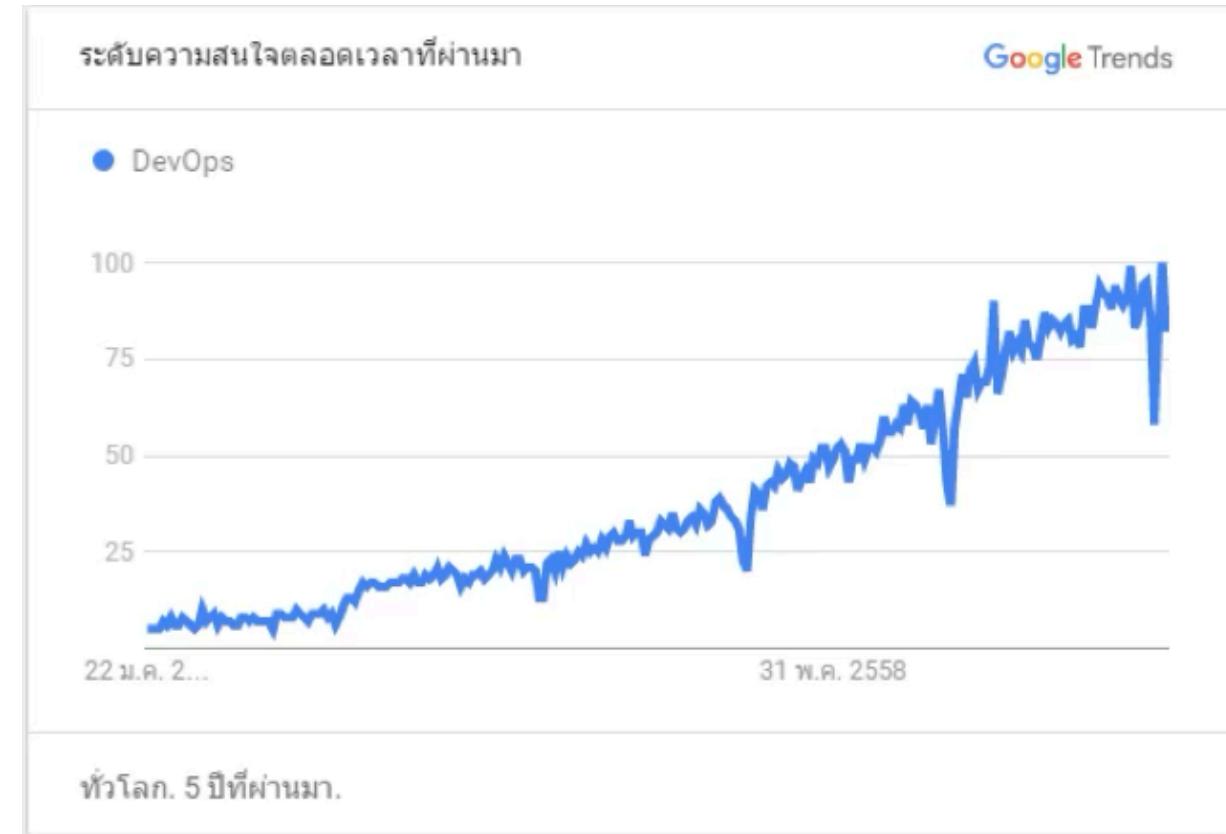
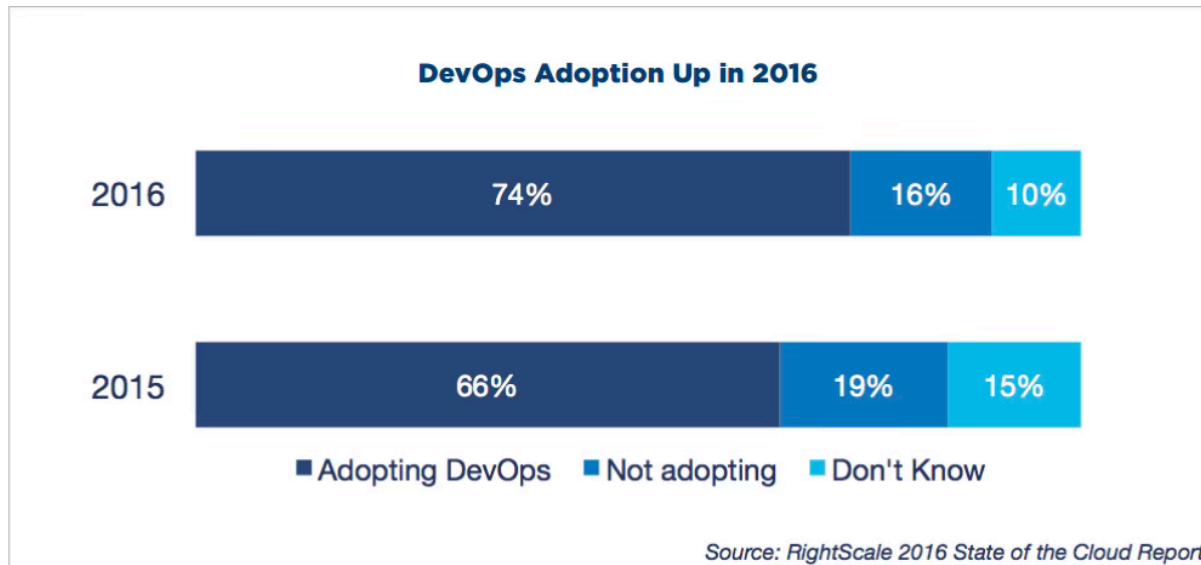
# Introduction to **DEVOPS**

# Definitions and History

ในปี ค.ศ. 2009 ที่ Velocity Conference John Allspaw และ Paul Hammond ได้นำเสนอเรื่อง 10 Deploys per Day: Dev and Ops Cooperation at Flickr ซึ่งกล่าวถึงวิธีการสร้างเป้าหมายร่วมกันระหว่างแผนก Development และแผนก Operation และวิธีการที่ทำให้การ Deployment เป็นเรื่องทั่วไปที่ทำกันในชั่วต่ำประจำวัน การนำเสนอเรื่องนี้เป็นแรงบันดาลใจให้ Patrick Debois จัดงาน DevOpsDay ขึ้นมาในปีเดียวกัน คำว่า DevOps ที่ย่อมาจาก Development และ Operations จึงถูกสร้างขึ้นตั้งแต่นั้นเป็นต้นมา

ปัจจุบัน DevOps เป็นแนวคิดที่มีประสิทธิภาพและแพร่หลายออกไปทั่วโลก จากผลสำรวจของค์กรกว่า 1,000 แห่งจากรายงาน RightScale 2016 State of the Cloud Report: DevOps Trends พบร่วมกับ Google ระบุว่าในปี 2016 มีองค์กรนำ DevOps ไปปรับใช้แล้วถึง 74% ซึ่งเพิ่มขึ้นจากปีที่แล้วถึง 8% และจำนวนการ Search คำว่า DevOps ใน google ก็ยังเพิ่มขึ้นเรื่อยๆ ด้วย

# ระดับความน่าสนใจ

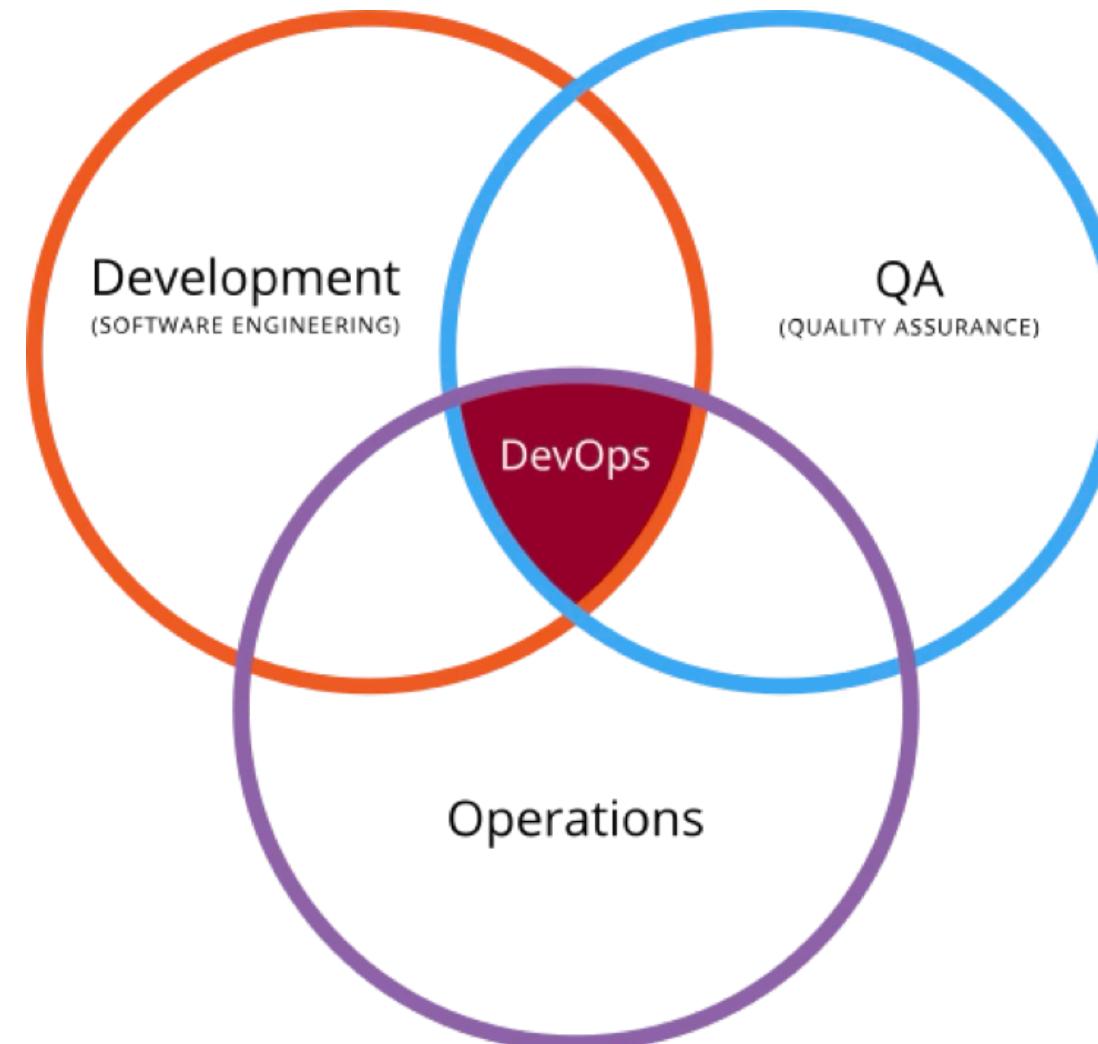


# Cultural change

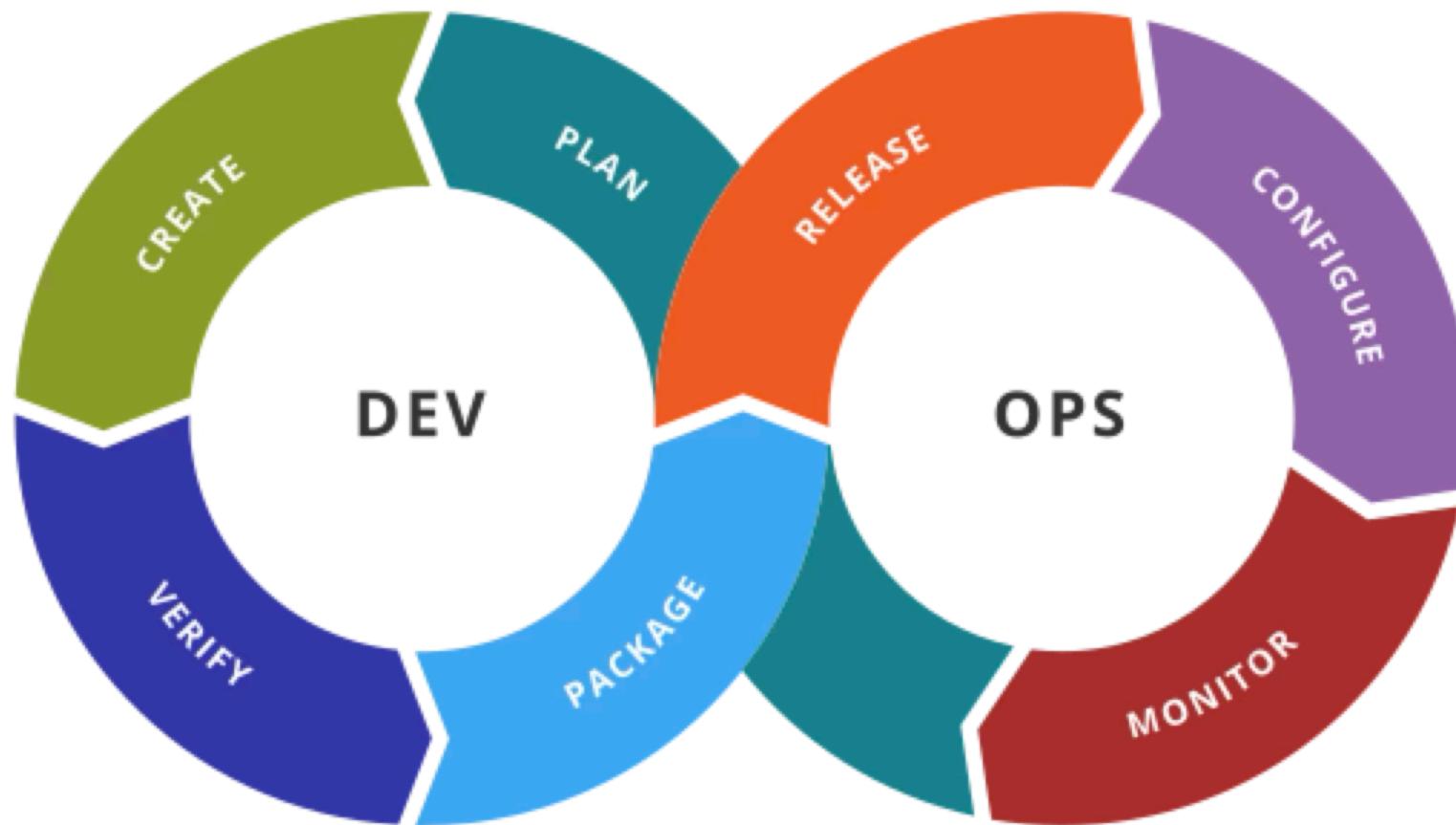
DevOps is more than just a tool or a process change.

- Operations – seeks organizational stability
- Developers – seek change
- Testers – seek risk reduction

# Overview



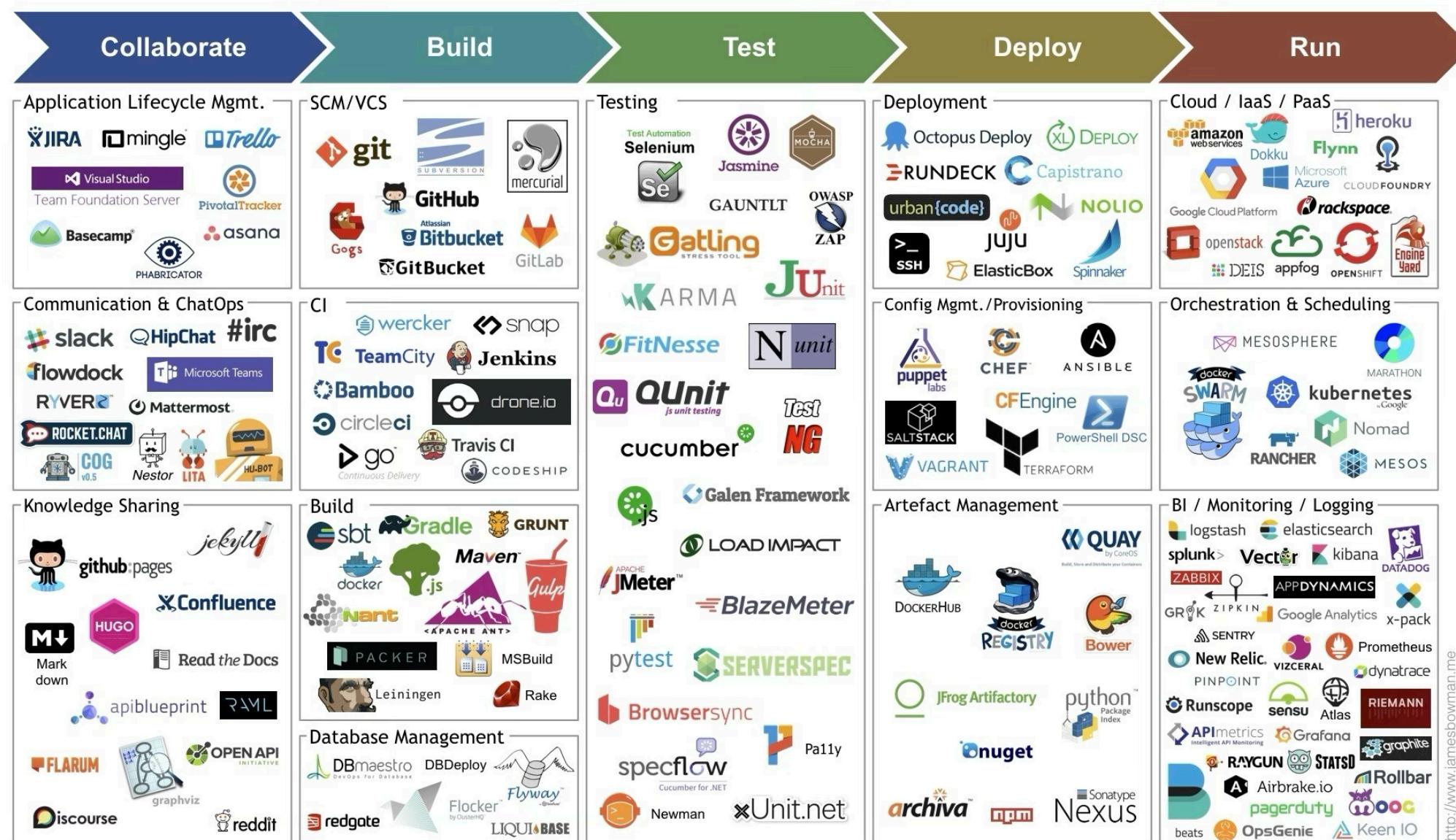
# Workflow



# 7 Ways to Get Started with DevOps

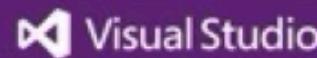
- Invite Your Operations Team Into Your Development Process
- Visualize the Work Together
- Automate Your Test/Build Process
- Create a Deployment Plan
- Identify Fragile Systems
- Smooth Out Wait States
- Link Your Work to Your Value

# DevOps Ecosystem


<http://www.jamesbowman.me>

# Collaborate

## Application Lifecycle Mgmt.



Team Foundation Server



Basecamp®



PHABRICATOR



PivotalTracker



## Communication & ChatOps



slack



HipChat

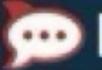
#irc



flowdock



Microsoft Teams



ROCKET.CHAT



COG  
v0.5



Mattermost.



Nestor



LITA

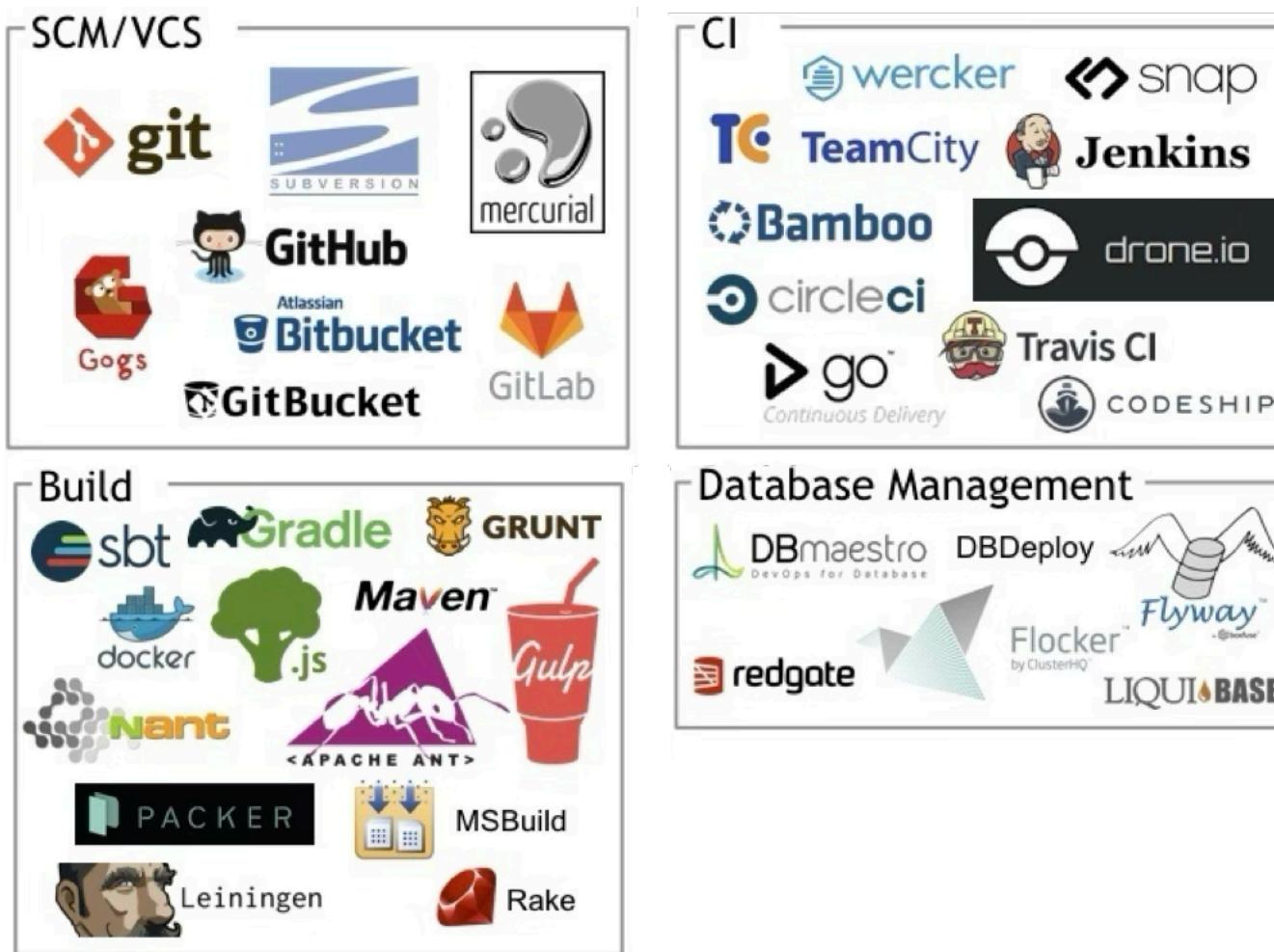


HU-BOT

# Collaborate #2



# Build



# Test



# Deploy

## Deployment



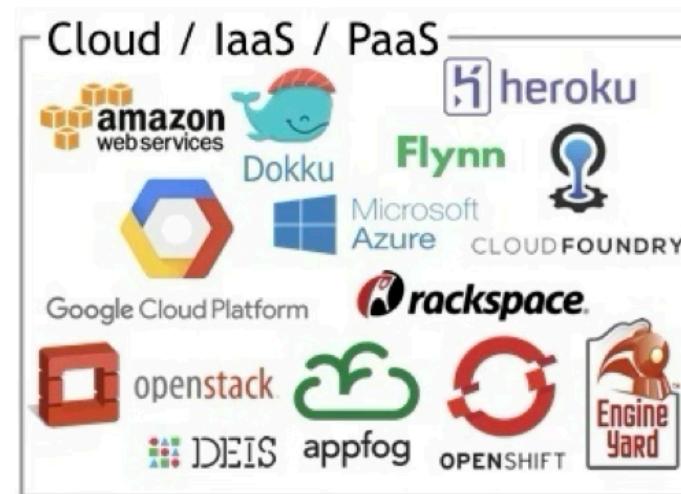
## Config Mgmt./Provisioning



## Artefact Management



# Run



# Everything as code

# Everything as code

Everything as code is a software development practice that seeks to apply the same principles of version control, testing, and deployment to enhance maintainability and scalability of all aspects of the development lifecycle, including networking infrastructure, documentation, and configuration.

This practice adds the ability to automate more, leading to faster, more consistent, and more reliable development cycles. By using code for as many use cases as possible, developers can achieve a higher level of quality, reduce the risk of errors, and increase the speed at which they can deploy new features and updates.

# Everything as code

- Infrastructure as Code (IaC)
- Diagram as Code
- Presentation Slide as Code

# What is Infrastructure as Code (IaC)

Infrastructure as code (IaC) is the ability to provision and support your computing infrastructure using code instead of manual processes and settings. Any application environment requires many infrastructure components like operating systems, database connections, and storage. Developers have to regularly set up, update, and maintain the infrastructure to develop, test, and deploy applications.

Manual infrastructure management is time-consuming and prone to error—especially when you manage applications at scale. Infrastructure as code lets you define your infrastructure's desired state without including all the steps to get to that state. It automates infrastructure management so developers can focus on building and improving applications instead of managing environments. Organizations use infrastructure as code to control costs, reduce risks, and respond with speed to new business opportunities.

# What is diagram as code

Diagram-as-code is a way to quickly draw beautiful diagrams using a simple DSL instead of dragging boxes and arrows with your mouse.

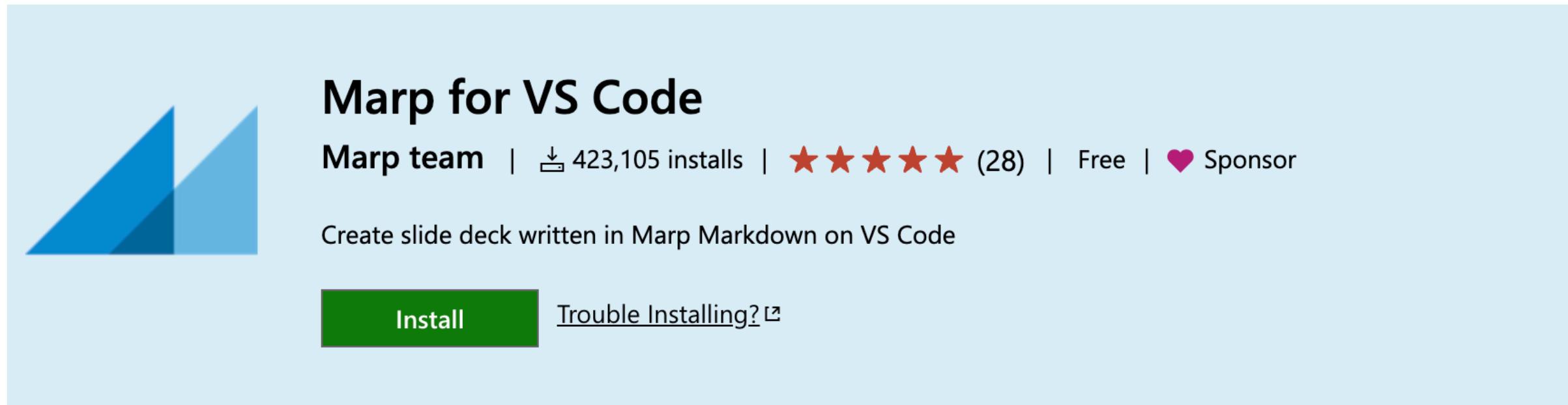
## Types of diagrams

- Flow Charts: Visualize process flows, user flows, logic flows
- Entity Relationship Diagrams: Visualize data models
- Cloud Architecture Diagrams: Visualize cloud infrastructure and data flow
- Sequence Diagrams: Visualize system flows and interactions

# เอกสารประกอบการบรรยายนี้

สร้างด้วย Slide as code โดยใช้เครื่องมือดังนี้

- visual studio code
- markdown
- extension: Marp for VS Code



The screenshot shows the Microsoft Store page for the "Marp for VS Code" extension. The page features a large blue triangle graphic on the left. The title "Marp for VS Code" is prominently displayed in bold black text. Below the title, it says "Marp team | 423,105 installs | ★★★★★ (28) | Free | Sponsor". A descriptive text below reads "Create slide deck written in Marp Markdown on VS Code". At the bottom, there are two buttons: a green "Install" button and a link "Trouble Installing?".

# CI/CD

# CI/CD

- Continuous Integration (CI)
- Continuous Deployment (CD)
- Continuous Delivery (CD)

# Continuous Integration (CI)

- Continuous Integration uses automation tools that empower development teams to build and test code after each merge as seamlessly as possible

# Continuous Delivery (CD)

- Focused on keeping the code ready to deploy at any given time.
- It's not about making bugged-code available for the production environment.
- Rather, all the sets of features are ready to go, and the latest build is ready to be delivered at any time given.

# Continuous Deployment

- Production deployment of every change done to the code.
- The changes are ideally automated, without human intervention.
- The approach works well in a corporate environment, where the user and the tester are ultimately one person.

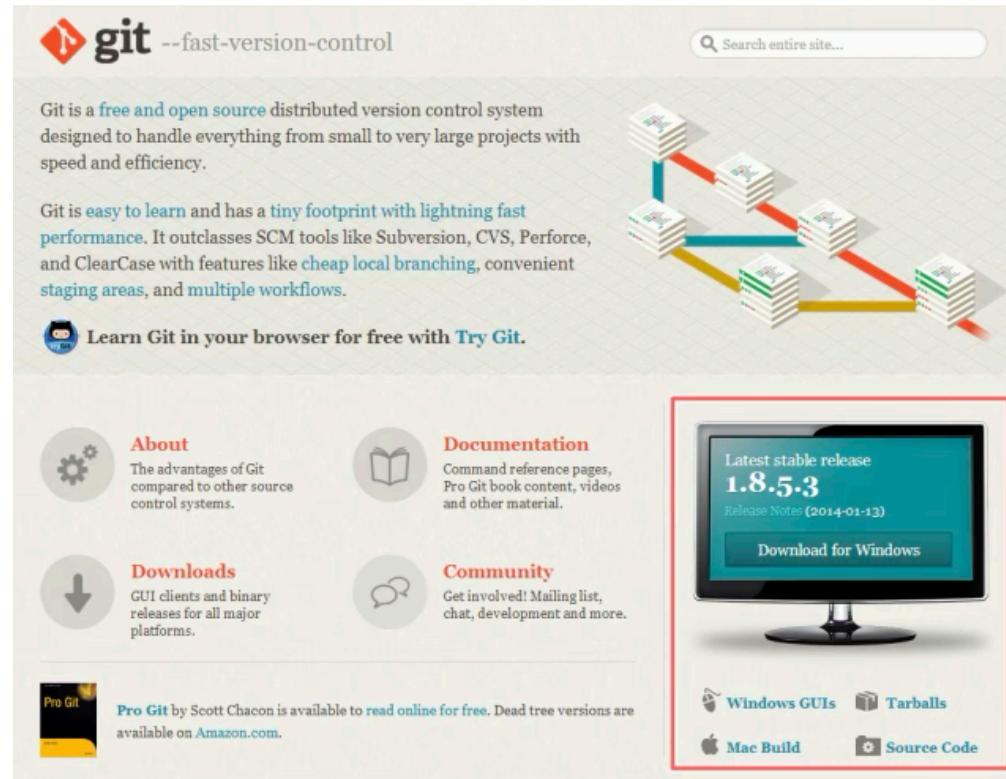
# Software Installation

การติดตั้ง

# GIT FOR WINDOWS

# การติดตั้ง git สำหรับ windows

- เข้า website <https://git-scm.com/downloads>
- เลือก Download Git เพื่อติดตั้งบน Windows



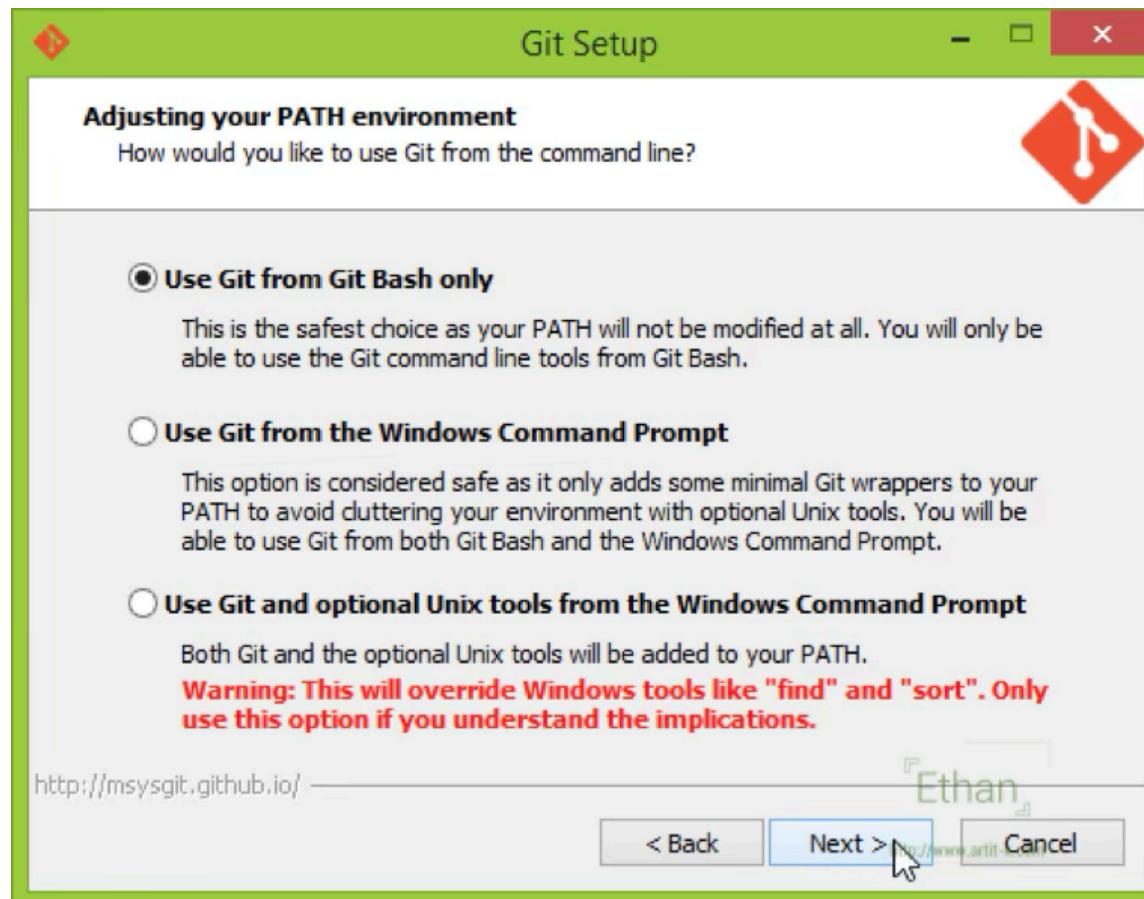
# การติดตั้ง GIT สำหรับ Windows #2

ติดตั้งโดยใช้สิทธิ์ Administrator ในการติดตั้ง



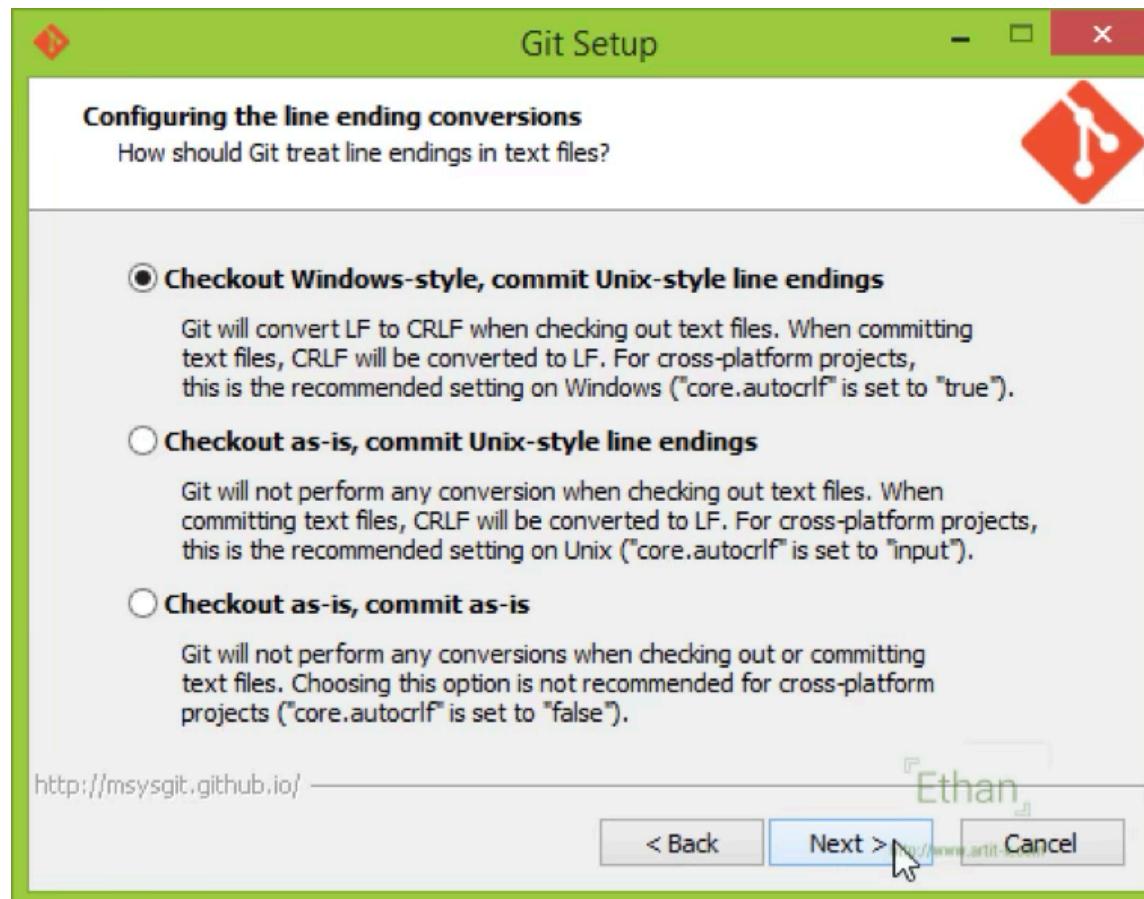
# การติดตั้ง GIT สำหรับ Windows #3

เลือกติดตั้งแบบ Use Git from the Windows Command Prompt



# การติดตั้ง GIT สำหรับ Windows #4

เลือกเป็น Checkout Windows-style, commit Unix-style line endings



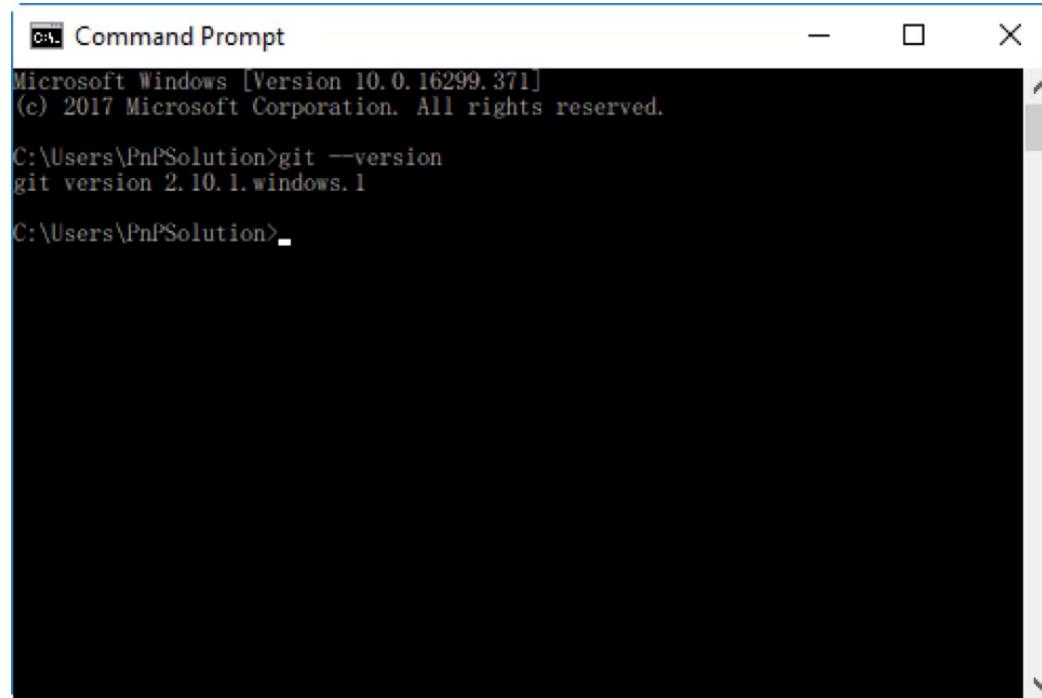
# การติดตั้ง GIT สำหรับ Windows #5

กดปุ่ม next ไปจนถึงหน้าสุดท้าย



# ทดสอบหลังการติดตั้ง Git

เปิดโปรแกรม cmd และพิมพ์คำสั่ง `git --version`



A screenshot of a Microsoft Windows Command Prompt window titled "Command Prompt". The window shows the following text:  
Microsoft Windows [Version 10.0.16299.371]  
(c) 2017 Microsoft Corporation. All rights reserved.  
C:\Users\PnP\Documents>git --version  
git version 2.10.1.windows.1  
C:\Users\PnP\Documents>

ผลลัพธ์จะได้เป็นเลข version ของ git

การติดตั้ง

# DOCKER DESKTOP

# การติดตั้ง Docker สำหรับ Windows

ความต้องการขั้นพื้นฐานสำหรับการติดตั้ง Docker for windows

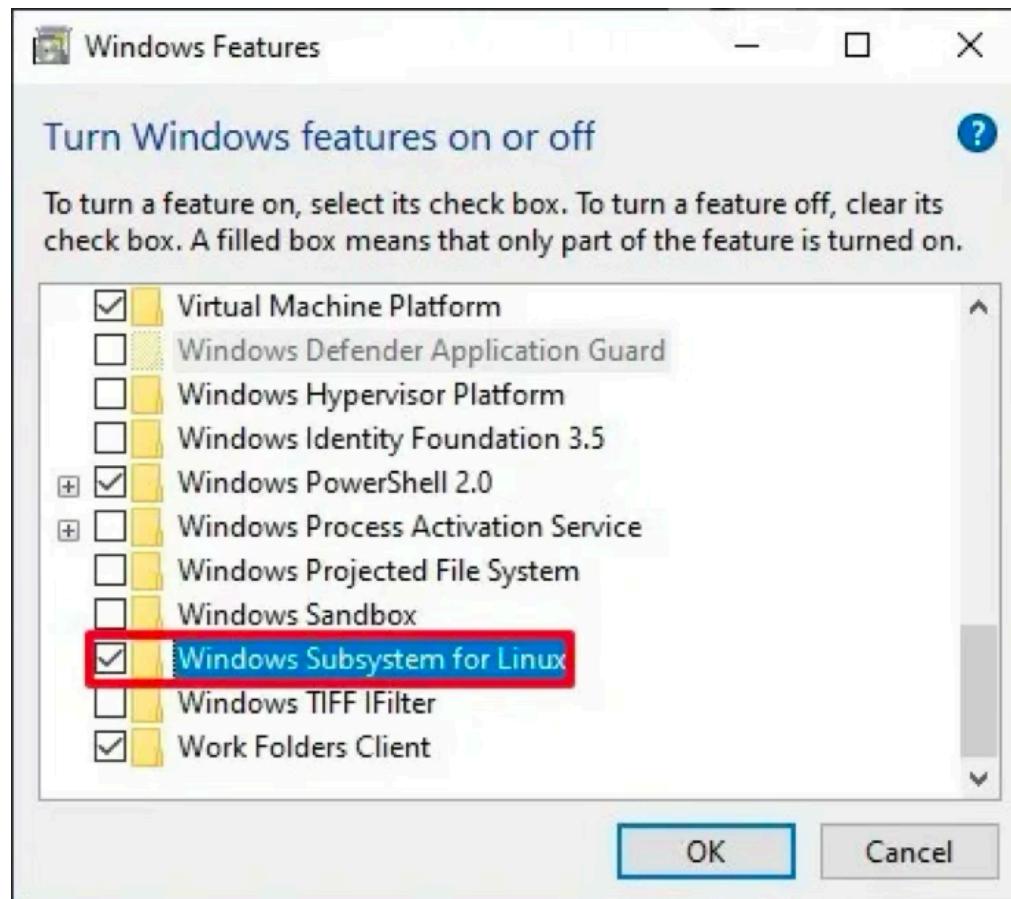
- Windows ต้องเป็น version 64bit เท่านั้น
- Windows 10 pro ขึ้นไป
- CPU ต้องมีความสามารถ VTx เพื่อรองรับการทำงานของ Hyper-v โดยสามารถเข้าไปเปิดได้ที่ BIOS ของเครื่อง

# ตัวอย่างการเปิดใช้งาน VTx ใน BIOS



# การติดตั้ง Docker สำหรับ Windows #2

## เปิดใช้ WSL2



# การติดตั้ง Docker สำหรับ Windows #3

- run คำสั่งดังต่อไปนี้ ใน Power Shell แบบ Administrator

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart  
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

- download and install patch จาก link

```
https://wslstorestorage.blob.core.windows.net/wslblob/wsl\_update\_x64.msi
```

- run คำสั่งดังนี้เพื่อตั้งค่า wsl เป็น version 2

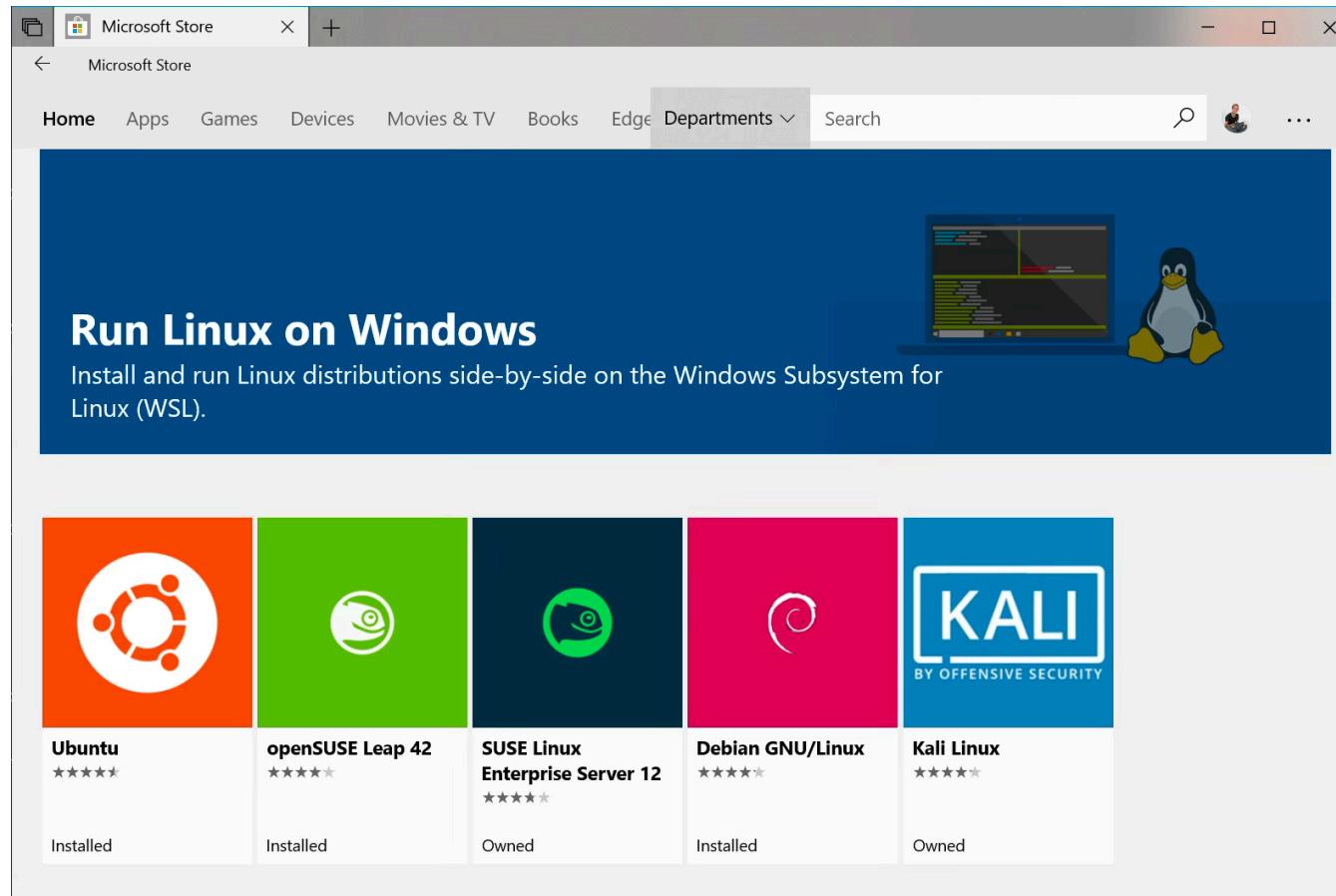
```
wsl --set-default-version 2  
wsl --update --web-download
```

# การติดตั้ง Docker สำหรับ Windows #4

- เข้า link <https://www.docker.com/products/docker-desktop/>
- กด next ไปเรื่อยๆ จนสิ้นสุดการติดตั้ง

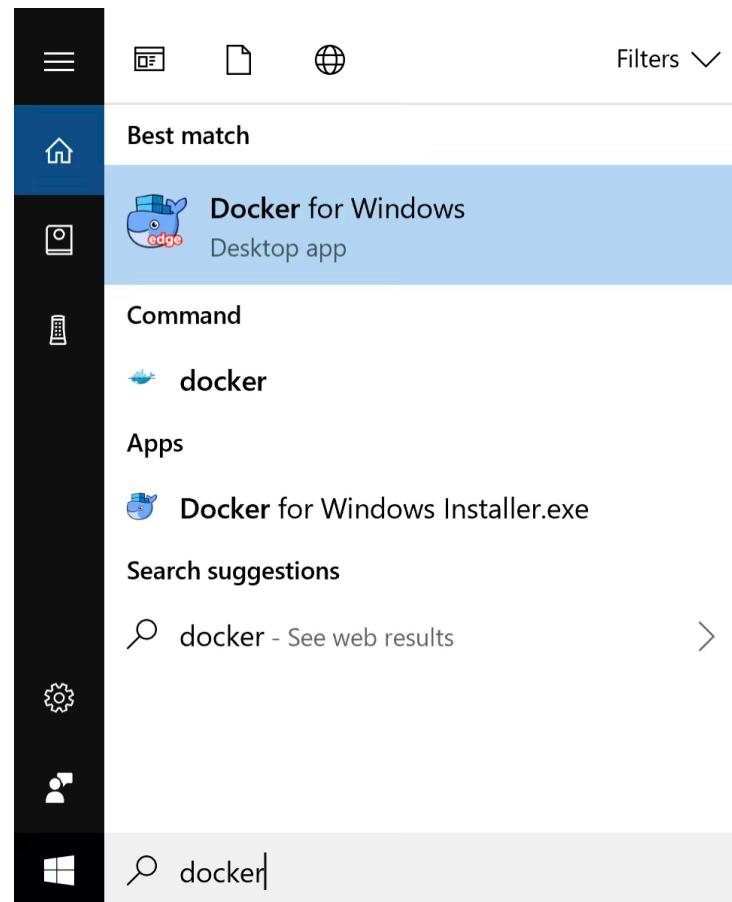
# การติดตั้ง Docker สำหรับ Windows #5

## ติดตั้ง linux distro สำหรับ wsl



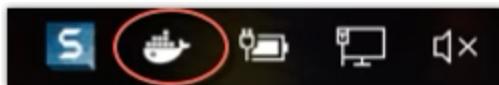
# ทดสอบหลังการติดตั้ง #1

หลังจากติดตั้งเสร็จแล้วให้ทำการเปิด docker desktop ดังนี้



# ทดสอบหลังการติดตั้ง #2

จะมี docker run อยู่ที่ taskbar ดังภาพ



ทดสอบการติดตั้งโดยการเปิด cmd และพิมพ์คำสั่งดังนี้

```
docker ps
```

ผลที่ได้จากการ run คำสั่ง

```
[Sommais-MacBook-Pro:~ sommaik$ docker ps
```

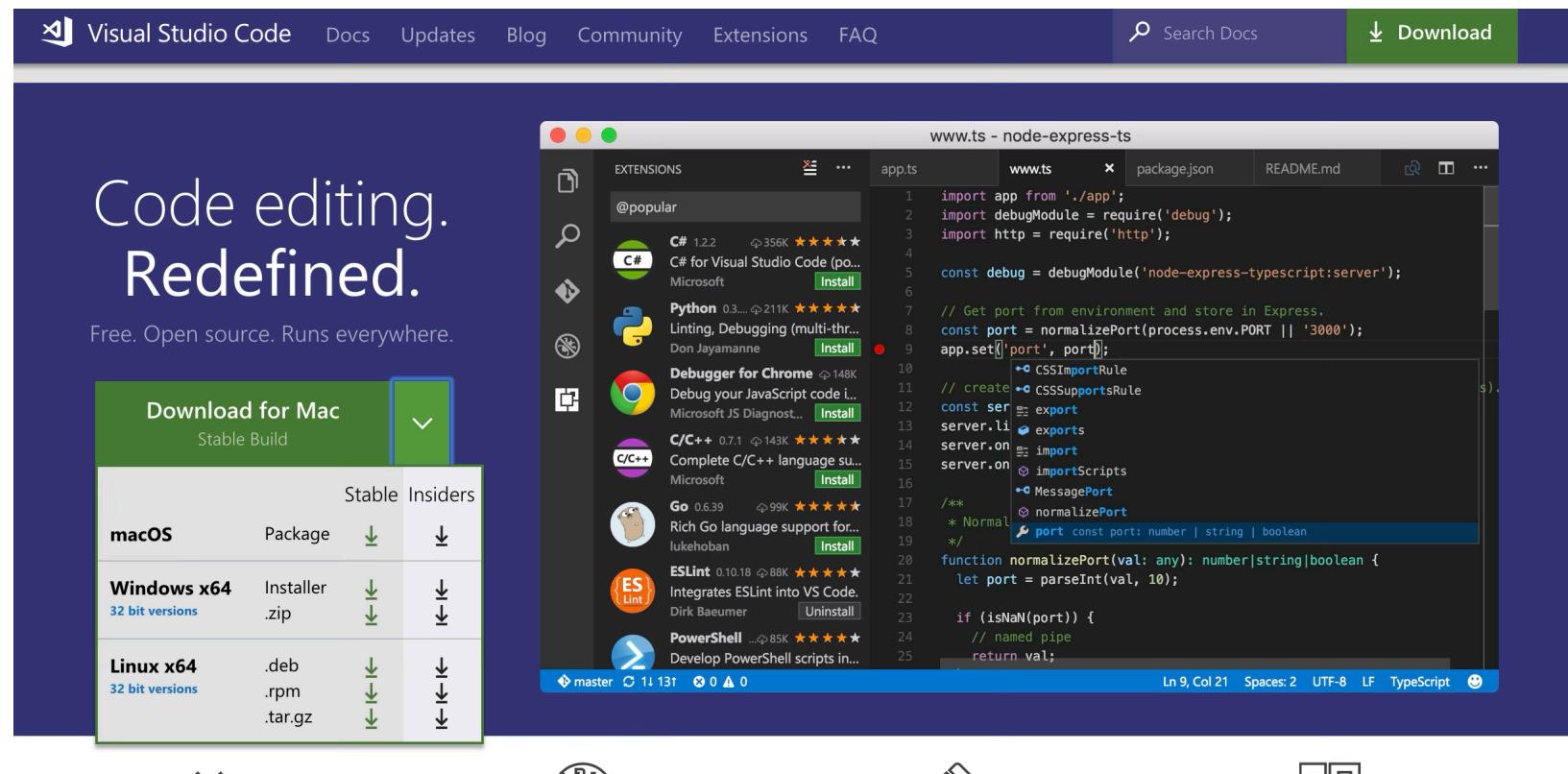
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

การติดตั้ง

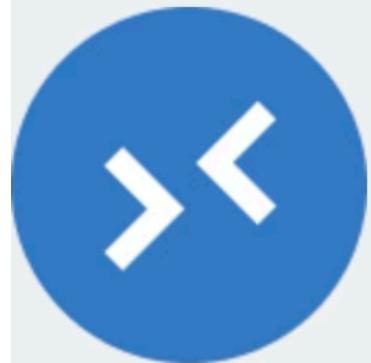
# VISUAL STUDIO CODE

# การติดตั้ง Visual Studio Code (VSC)

1. เข้าไปที่ website <https://code.visualstudio.com/>
2. เลือก download สำหรับ windows (stable)



# Install extension ดังนี้



## Remote Development

[Preview](#)

Microsoft  [microsoft.com](https://microsoft.com) |  4,294,545 installs |  (107) | Free

An extension pack that lets you open any folder in a container, on a remote machine, or in WSL and take advantage of VS Code's full feature set.

[Install](#)[Trouble Installing?](#)

การติดตั้ง

**MINIKUBE**

# Step 1

- เข้า website <https://minikube.sigs.k8s.io/docs/start/>
- เลือก Download Version ที่ต้องการ

## 1 Installation

Click on the buttons that describe your target platform. For other architectures, see [the release page](#) for a complete list of minikube binaries.

Operating system

[Linux](#) [macOS](#) [Windows](#)

Architecture

[x86-64](#)

Release type

[Stable](#) [Beta](#)

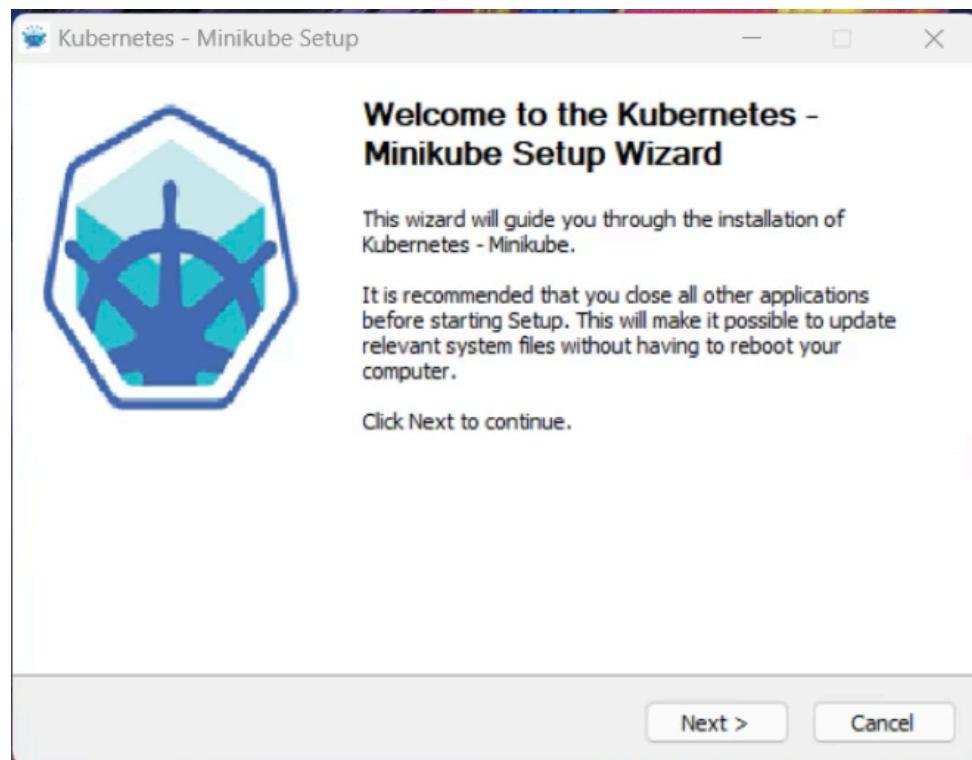
Installer type

[.exe download](#) [Windows Package Manager](#) [Chocolatey](#)

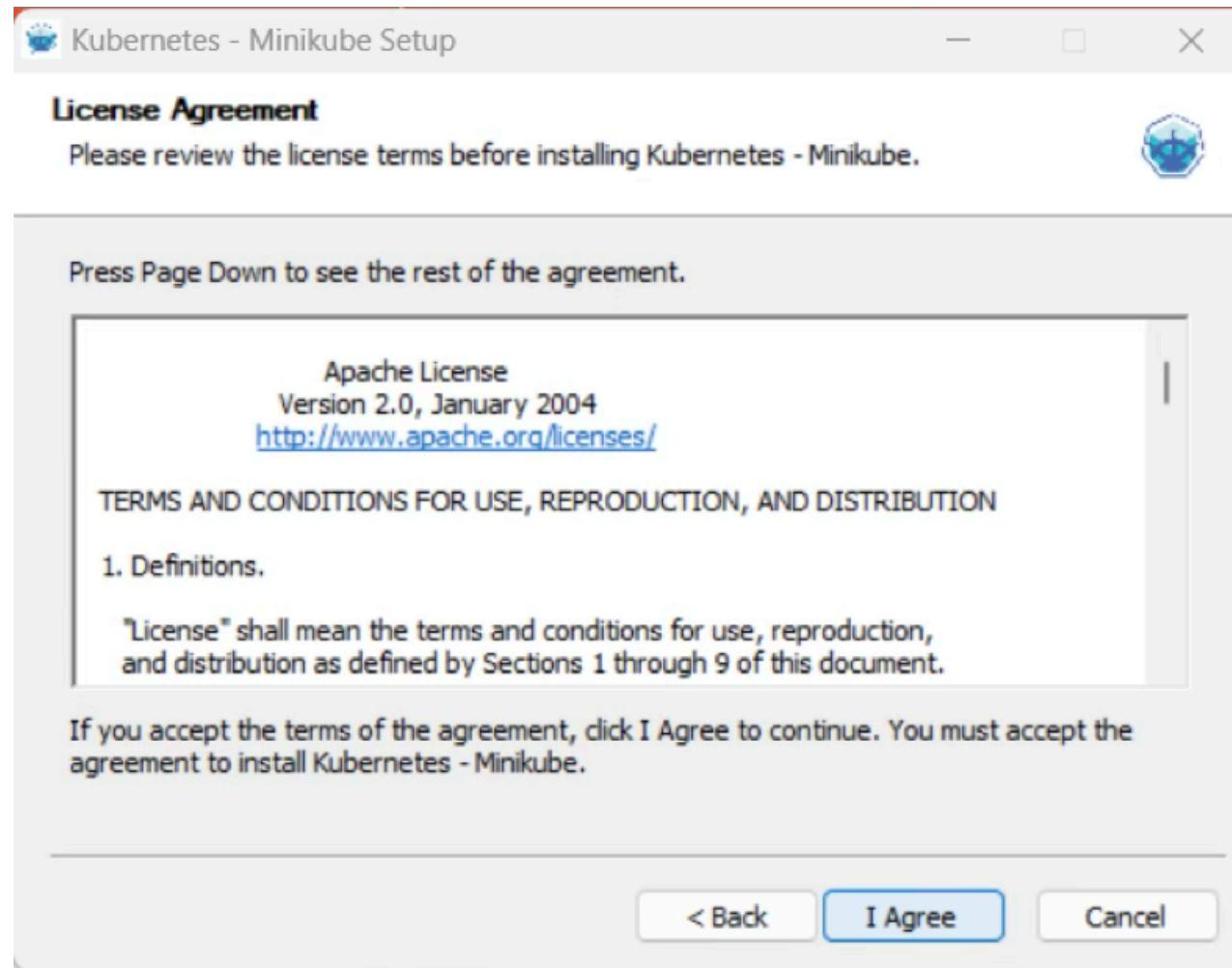
To install the latest minikube **stable** release on **x86-64 Windows** using **.exe download**:

# Step 2

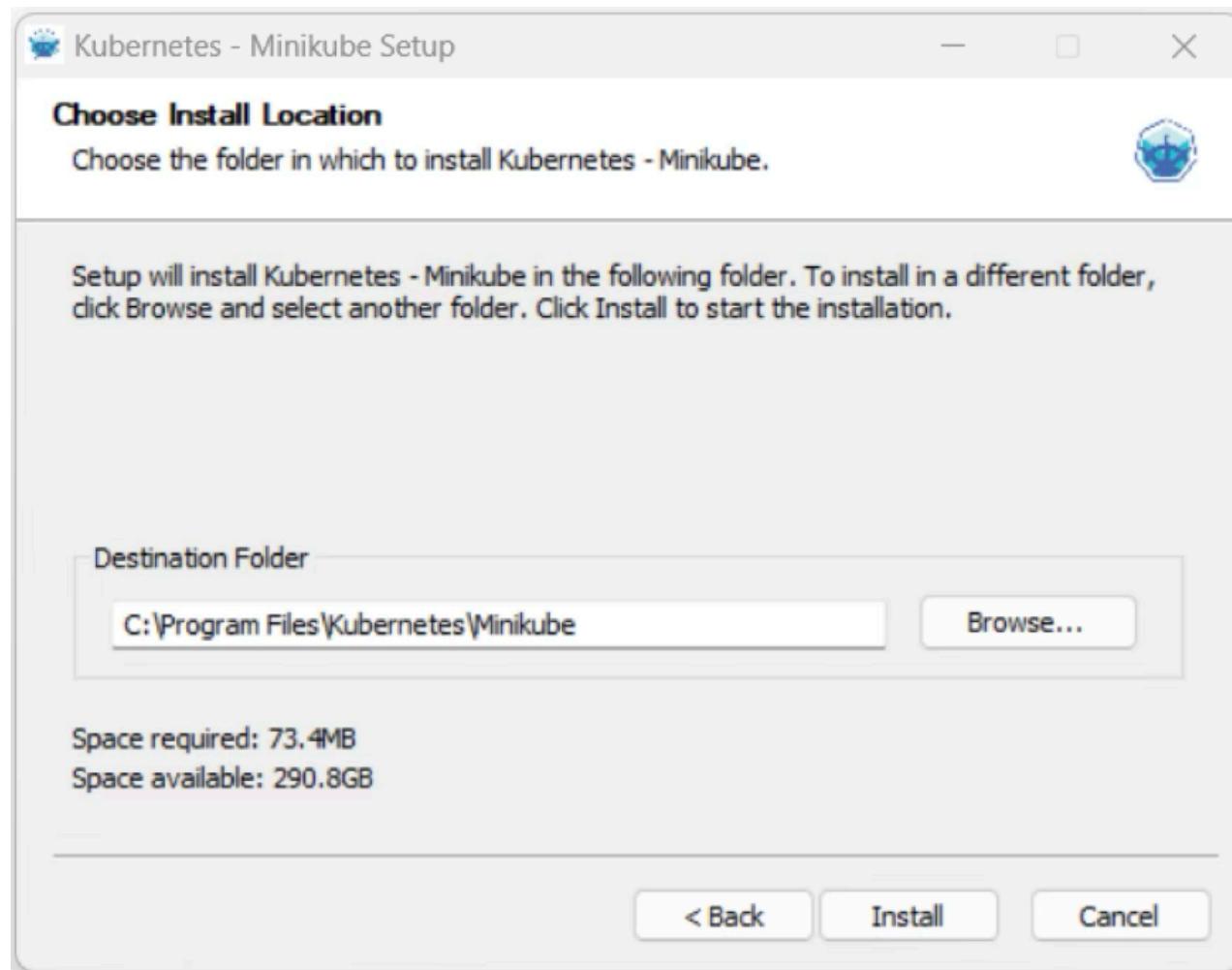
- ติดตั้งโดยใช้สิทธิ์ Administrator ในการติดตั้ง



# Step 3



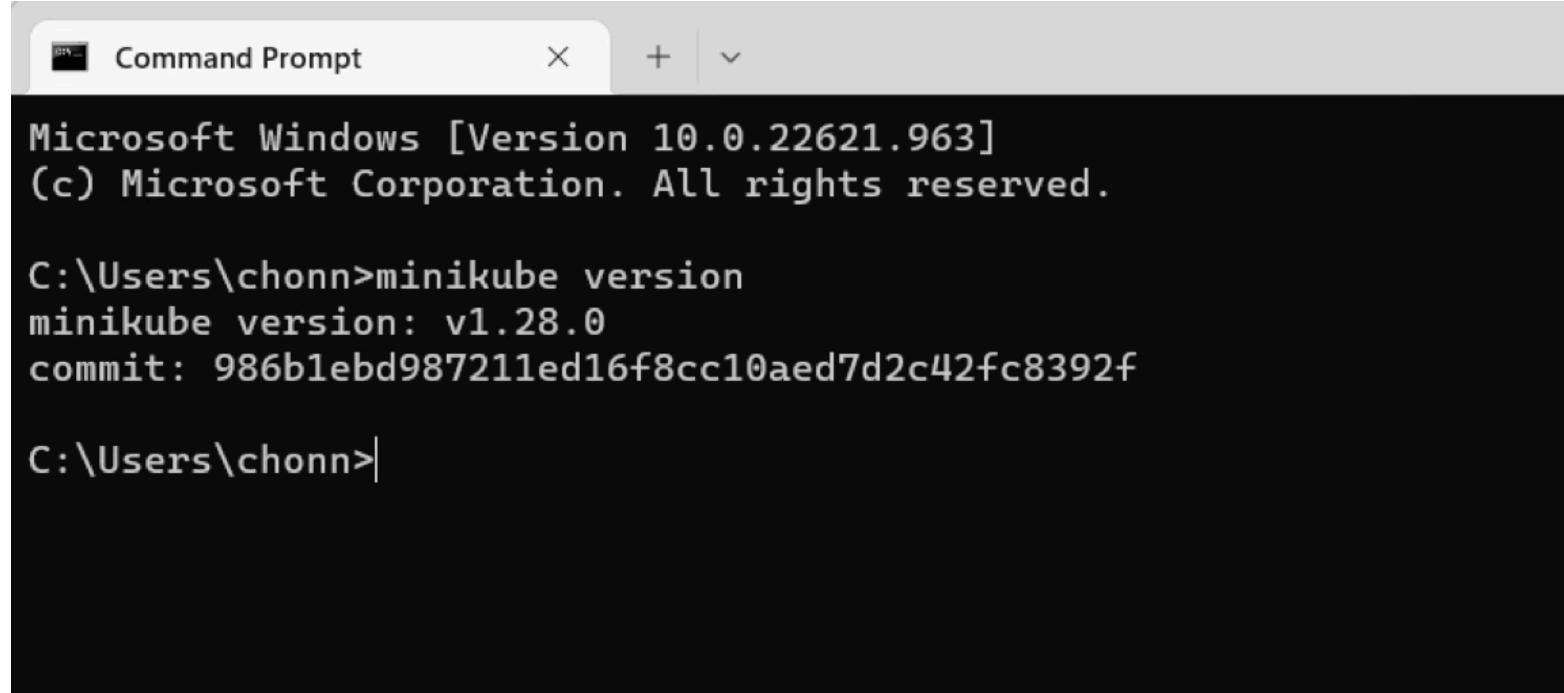
# Step 4



# Step 5 ทดสอบหลังการติดตั้ง MINIKUBE

- เปิดโปรแกรม cmd และพิมพ์คำสั่ง

```
minikube version
```



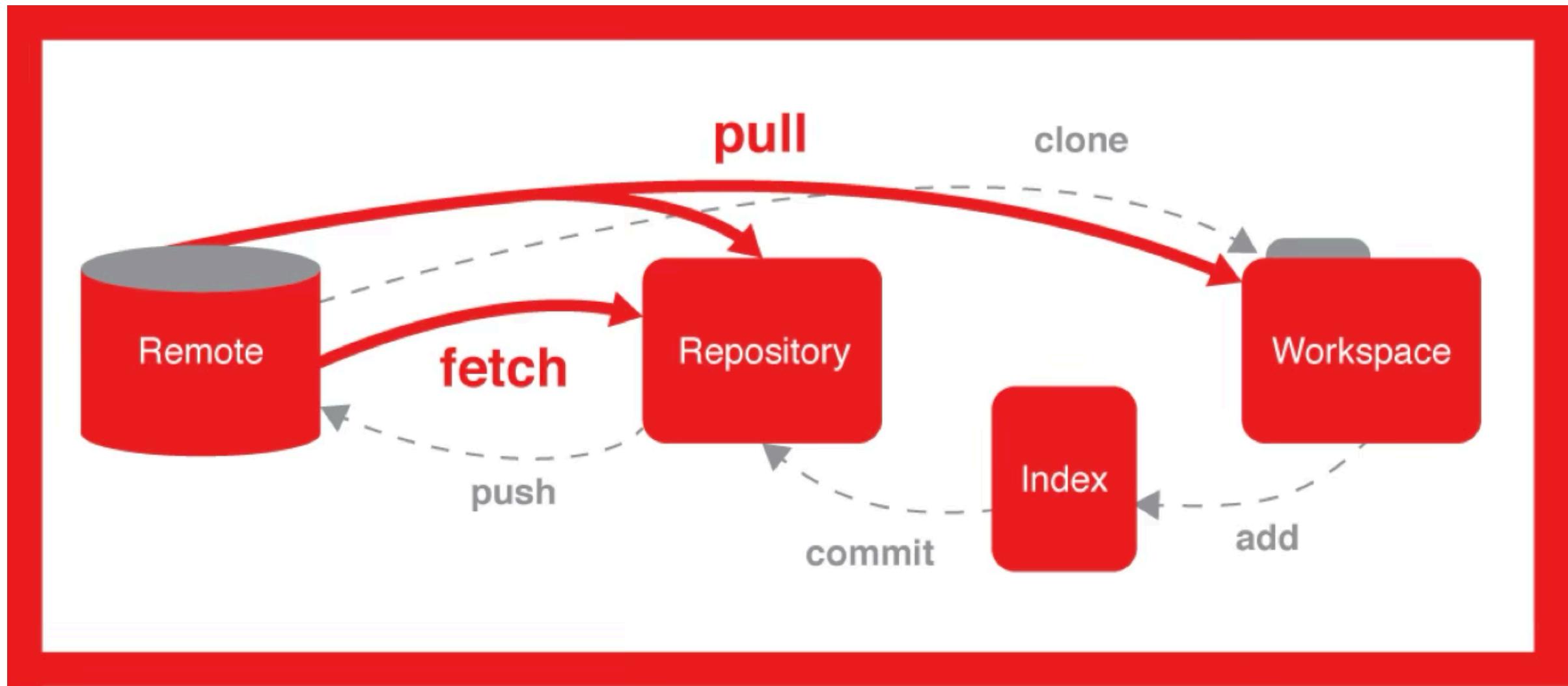
Microsoft Windows [Version 10.0.22621.963]  
(c) Microsoft Corporation. All rights reserved.

```
C:\Users\chonn>minikube version
minikube version: v1.28.0
commit: 986b1ebd987211ed16f8cc10aed7d2c42fc8392f

C:\Users\chonn>
```

# Software version control with **GIT**

# Overview



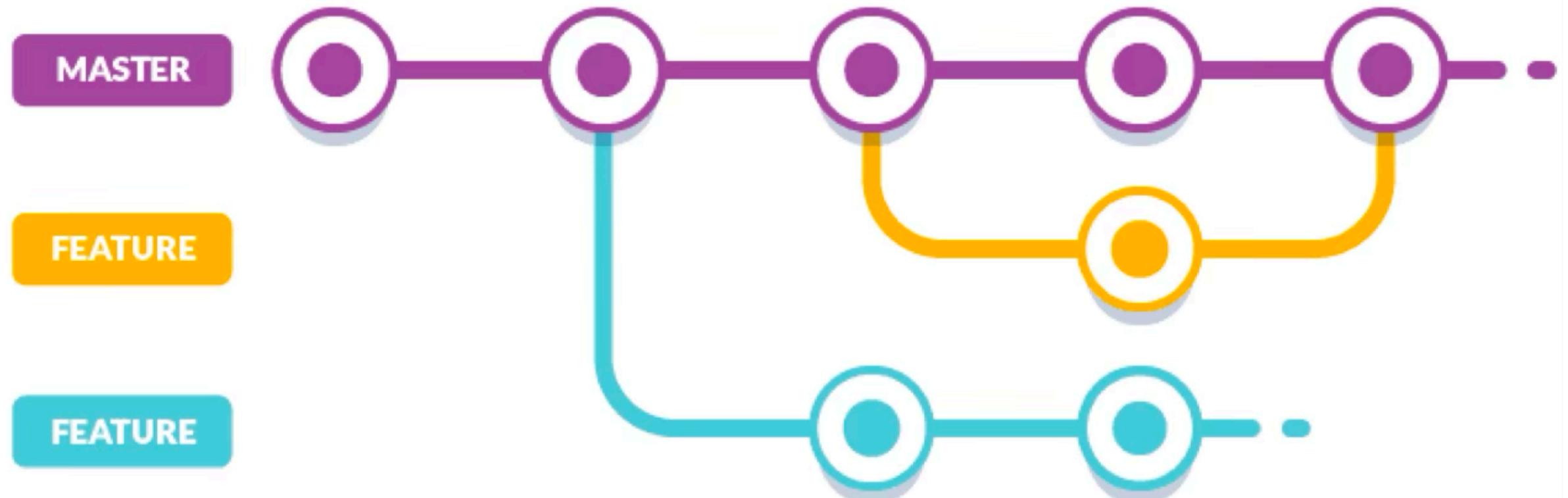
# Git workflow type

- Basic เหมาะกับ Project ขนาดเล็ก
- Feature Branch เหมาะกับ Project ขนาดกลาง
- Gitflow เหมาะกับ Project ขนาดใหญ่

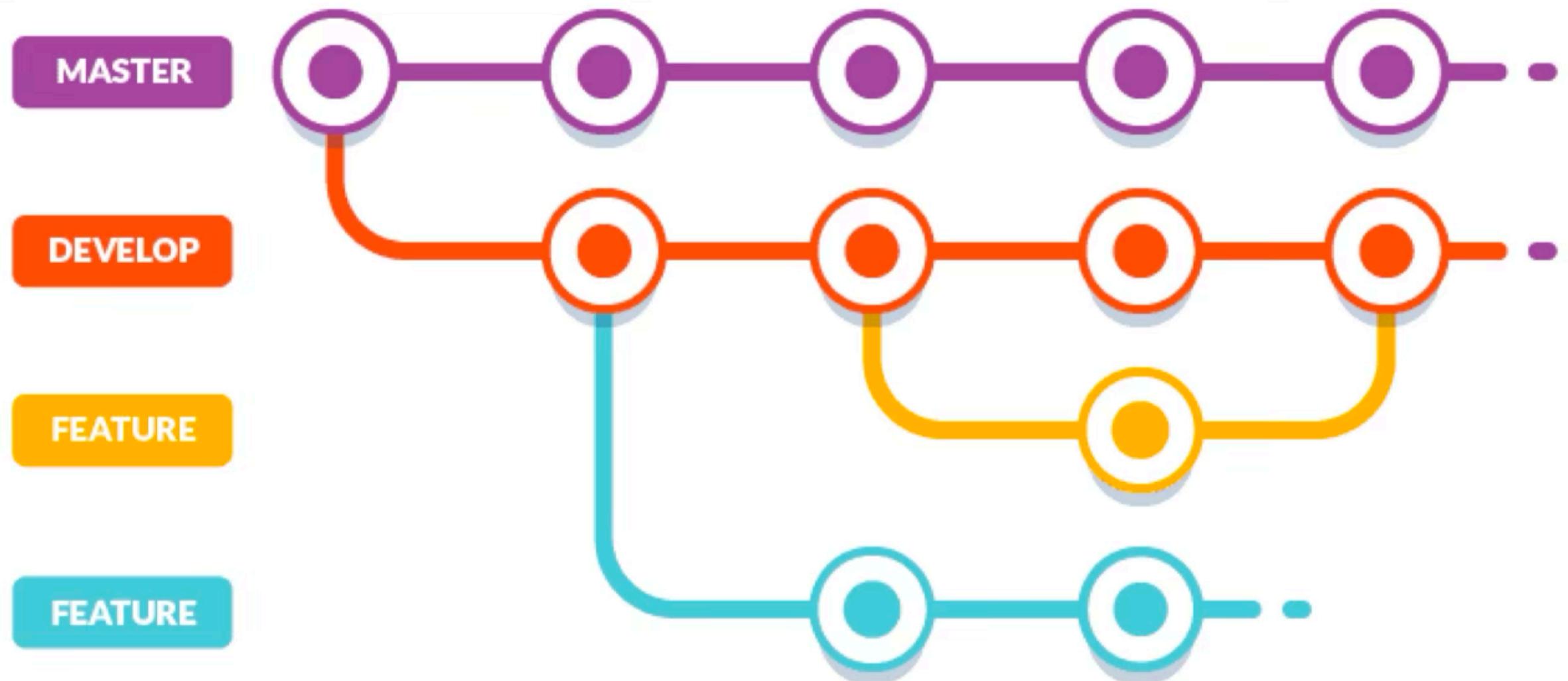
# Basic



# Feature Branch



# Gitflow



# Git Workshop

# Create a new Git Repository (Remote)

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner      Repository name



/ XXX ✓

Great repository names are short and memorable. Need inspiration? How about [improved-adventure](#).

Description (optional)



Public  
Anyone can see this repository. You choose who can commit.



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

Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



**Create repository**

# CONFIGURE TOOLING (First Time)

Sets the name you want attached to your commit transactions

```
git config --global user.name "[name]"  
git config --global user.name "XXX"
```

Sets the email you want attached to your commit transactions

```
git config --global user.email "[email address]"  
git config --global user.email "XXXX@hotmail.com"
```

Enables helpful colorization of command line output

```
git config --global color.ui auto
```

# CREATE REPOSITORIES

Creates a new local repository with the specified name

```
git init [project-name]  
git init XXX
```

Downloads a project and its entire version history

```
git clone [url]  
git clone https://github.com/Sommaik/XXX.git
```

# MAKE CHANGES

Lists all new or modified files to be committed

```
git status
```

Shows file differences not yet staged

```
git diff
```

Snapshots the file in preparation for versioning

```
git add [file]  
git add readme.txt  
git add .
```

# MAKE CHANGES #2

Shows file differences between staging and the last file version

```
git diff --staged
```

Unstages the file, but preserve its contents

```
git reset [file]  
git reset readme.txt
```

Records file snapshots permanently in version history

```
git commit -m "[descriptive message]"  
git commit -m "Initial Project"
```

# GROUP CHANGES #3

Lists all local branches in the current repository

```
git branch
```

Creates a new branch

```
git branch [branch-name]  
git branch XXX
```

Switches to the specified branch and updates the working directory

```
git checkout [branch-name]  
git checkout XXX
```

# GROUP CHANGES #4

Combines the specified branch's history into the current branch

```
git merge [branch]  
git merge XXX
```

Deletes the specified branch

```
git branch -d [branch-name]  
git branch -d XXX
```

# REFACTOR FILENAMES

Deletes the file from the working directory and stages the deletion

```
git rm [file]  
git rm XXX.txt
```

Removes the file from version control but preserves the file locally

```
git rm --cached [file]  
git rm --cached XXX.txt
```

Changes the file name and prepares it for commit

```
git mv [file-original] [file-renamed]  
git mv XXX.txt YYY.txt
```

# SUPPRESS TRACKING

A text file named **.gitignore** suppresses accidental versioning of files and paths matching the specified patterns

```
*.log  
build/  
temp-*
```

Lists all ignored files in this project

```
git ls-files --other --ignored --exclude-standard
```

# SAVE FRAGMENTS

Temporarily stores all modified tracked files

```
git stash
```

Restores the most recently stashed files

```
git stash pop
```

Lists all stashed changesets

```
git stash list
```

Discards the most recently stashed changeset

```
git stash drop
```

# REVIEW HISTORY

Lists version history for the current branch

```
git log
```

Lists version history for a file, including renames

```
git log --follow [file]  
git log --follow XXX.txt
```

Shows content differences between two branches

```
git diff [first-branch]...[second-branch]
```

Outputs metadata and content changes of the specified commit

```
git show [commit]  
git show XXX
```

# REDO COMMITS

Undoes all commits after [commit], preserving changes locally

```
git reset [commit]  
git reset XXX
```

Discards all history and changes back to the specified commit

```
git reset --hard [commit]  
git reset --hard XXX
```

# SYNCHRONIZE CHANGES

Downloads all history from the repository bookmark

```
git fetch [bookmark]  
git fetch origin
```

Combines bookmark's branch into current local branch

```
git merge [bookmark]/[branch]  
git merge origin/master2
```

# SYNCHRONIZE CHANGES #2

Uploads all local branch commits to Git

```
git push [alias] [branch]  
git push origin master
```

Downloads bookmark history and incorporates changes

```
git pull
```

How to push source to

# GITLAB SERVER REPOSITORY

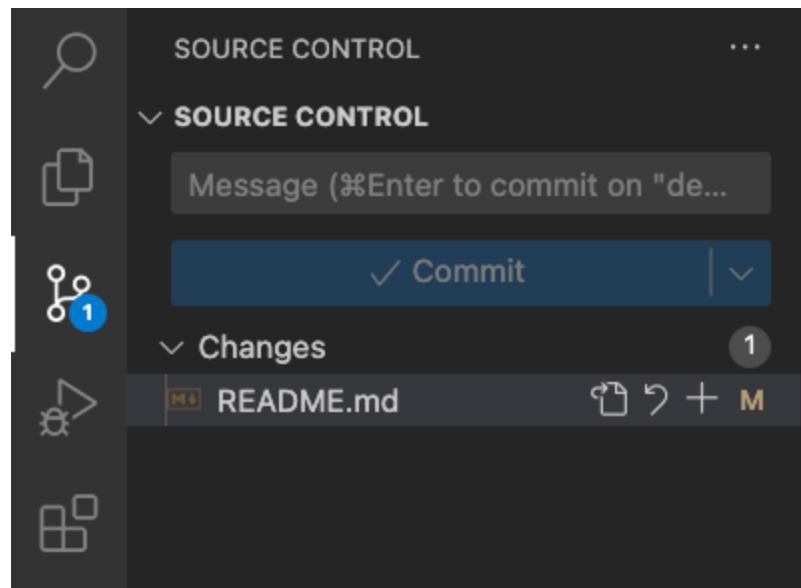
# Step 1/5

- สร้าง file ชื่อ README.md

```
# H1
## H2
### H3
#### H4
##### H5
###### H6
```

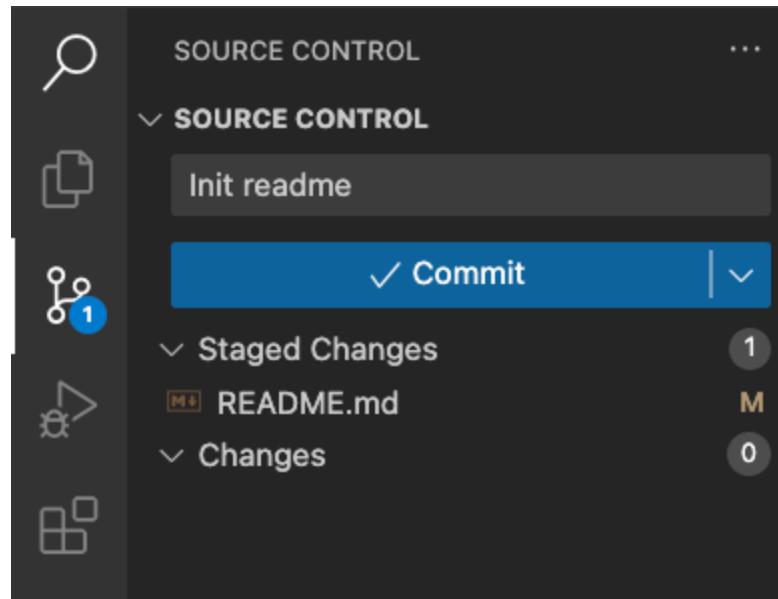
# Step 2/5

- กดเครื่องหมาย + ด้านหลัง file README.md
- ระบบจะเปลี่ยนจาก Changes เป็น Staged Changes



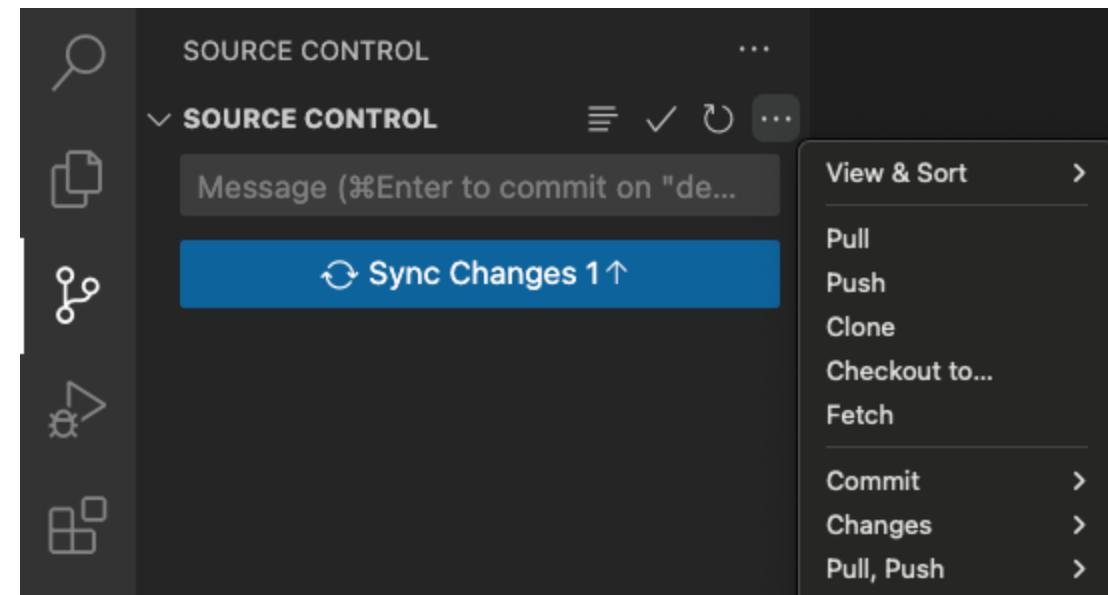
# Step 3/5

- ใส่ข้อความในช่อง Message และกดปุ่ม commit



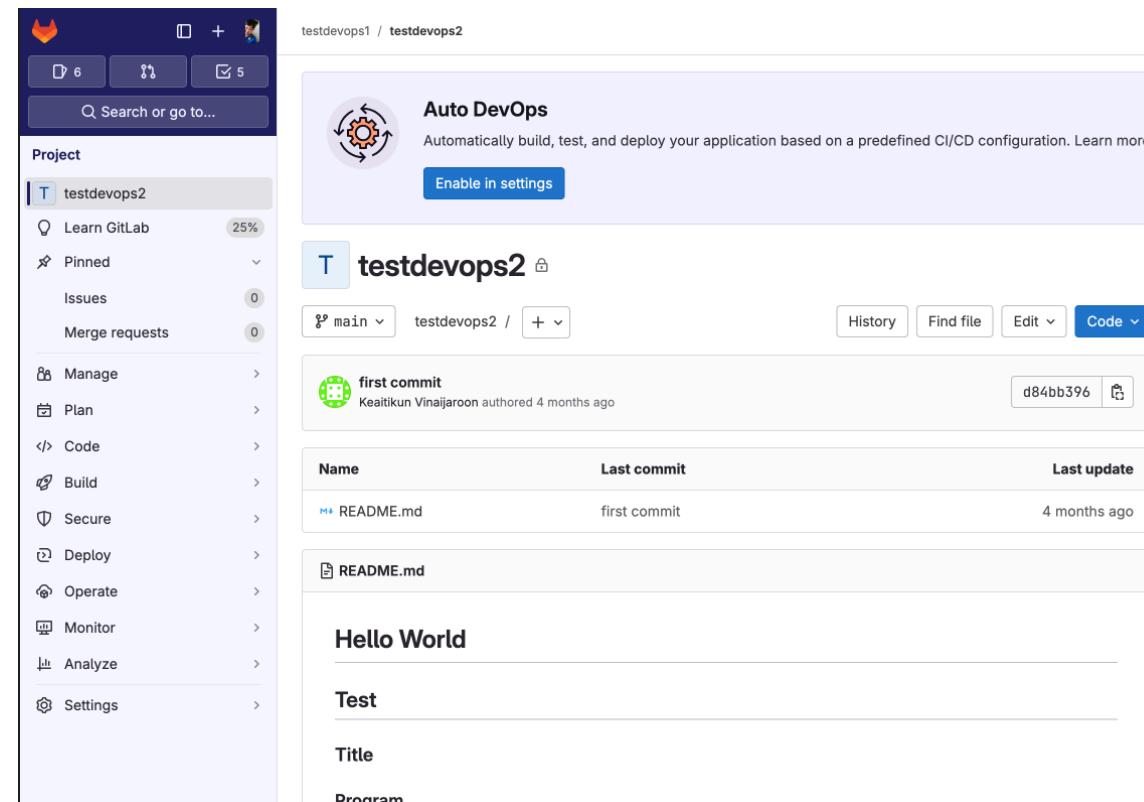
# Step 4/5

- กดปุ่ม Sync Changes
- หรือไปกดที่ menu Push (ในกรณีที่เป็นการใช้งานครั้งแรก ปุ่ม sync changes อาจจะไม่แสดงขึ้นมา)



# Step 5/5

- สามารถไปดูผลการ push code ได้ที่หน้า project บน Gitlab Server ของเรา
- โดยให้ตรวจสอบว่ามี code ของเรา update ขึ้นไปหรือไม่



The screenshot shows a GitLab project interface for 'testdevops2'. The left sidebar lists various project management sections like Project, Manage, Plan, Code, Build, Secure, Deploy, Operate, Monitor, Analyze, and Settings. The main area displays a commit history for 'first commit' by Keaitkun Vinajaroong, dated 4 months ago. The commit hash is d84bb396. Below the commit, there's a table showing file details: README.md has the last commit at 4 months ago. On the right, there's a code editor window titled 'Hello World' containing the text 'Hello World'.

How to use

# MARKDOWN

# Step 1/8

- update file README.md

Orderd list

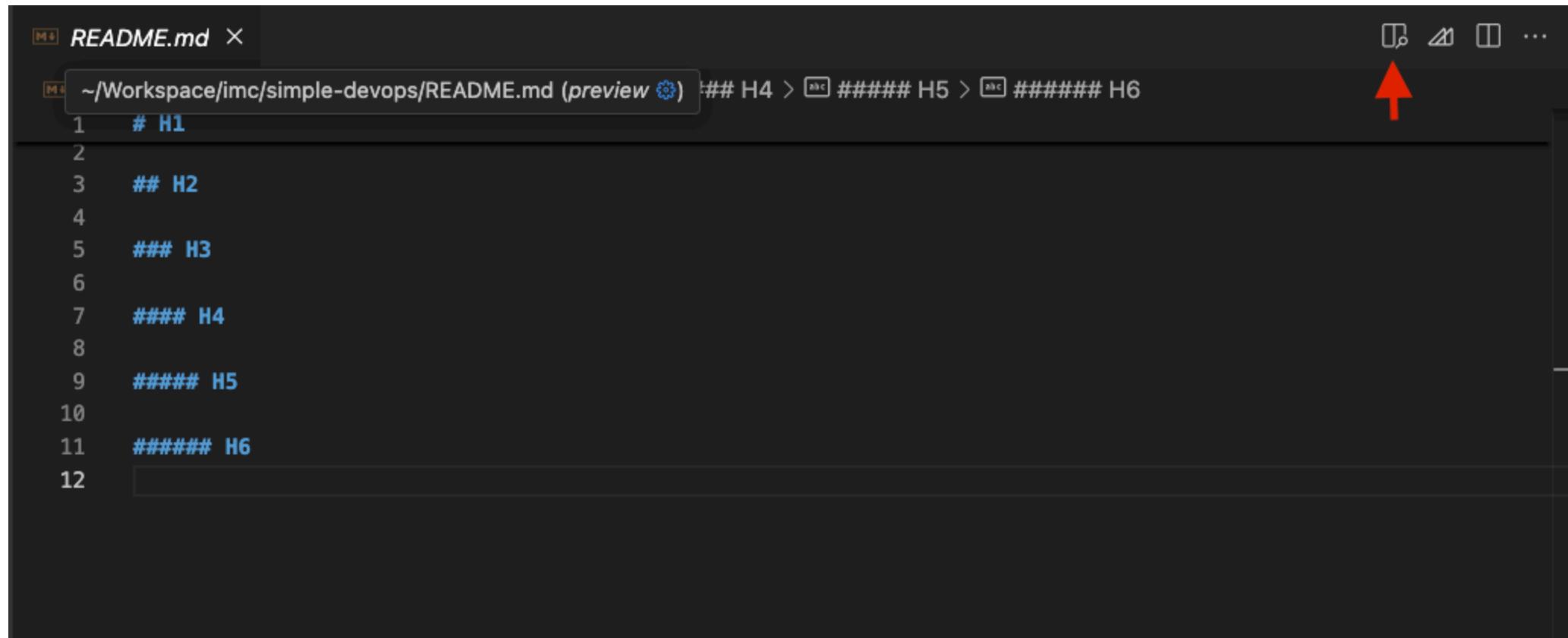
1. One
1. Two
1. Three

Unorderd list

- Line 1
- Line 2
- Line 3

# Step 2/8

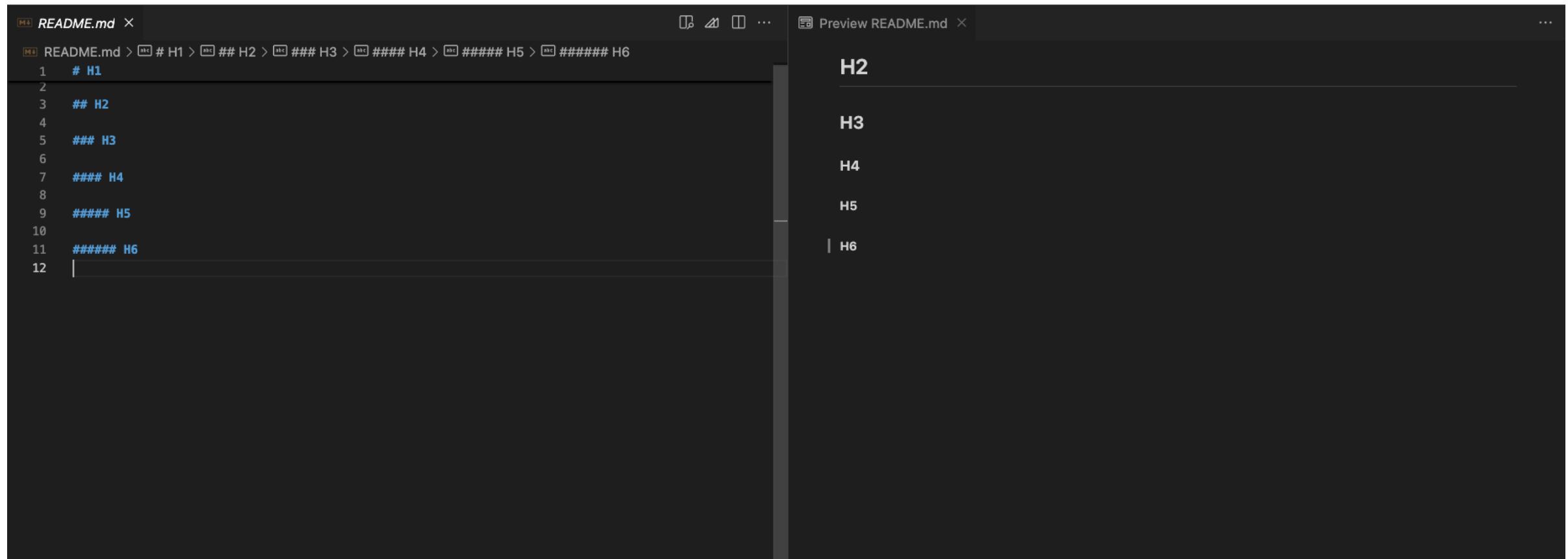
- กดปุ่มตามลูกศรเพื่อเป็นการแสดงผล live preview ของ markdown



```
1 # H1
2
3 ## H2
4
5 ### H3
6
7 #### H4
8
9 ##### H5
10
11 ##### H6
```

# Step 3/8

- เมื่อเราแก้ไข code ทางด้านซ้าย vs-code จะทำการแสดงผลการทำงานที่ด้านขวา



The screenshot shows the Visual Studio Code interface. On the left, the 'README.md' editor tab is open, displaying the following content:

```
1 # H1
2
3 ## H2
4
5 ### H3
6
7 #### H4
8
9 ##### H5
10
11 ##### H6
```

On the right, the 'Preview README.md' tab is open, showing the rendered output of the Markdown file:

## H2

### H3

#### H4

##### H5

###### H6

# Step 4/8

- เพิ่มเติม code เข้าไปที่ README.md

Image

! [](<https://placehold.co/600x400>)

Table

Code	Name	Age	
---	---	---	
001	Jame	24	

# Step 5/8

- เพิ่ม code สำหรับการสร้าง code block

```
```javascript
let a = 0;
````
```

```
```python
print("Hello World")
````
```

# Step 6/8

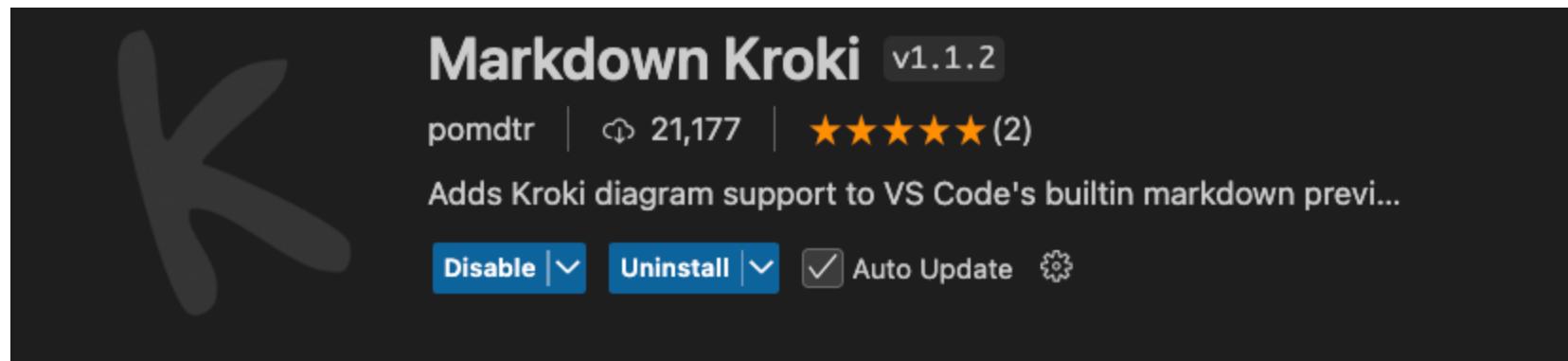
- เพิ่ม code สำหรับ สร้าง diagram เข้าไปที่ README.md

```
```mermaid
sequenceDiagram
    Alice->>John: Hello John, how are you?
    John-->>Alice: Great!
    Alice->>John: See you later!
````
```

Reference : [Mermaid.js.org](https://mermaid.js.org)

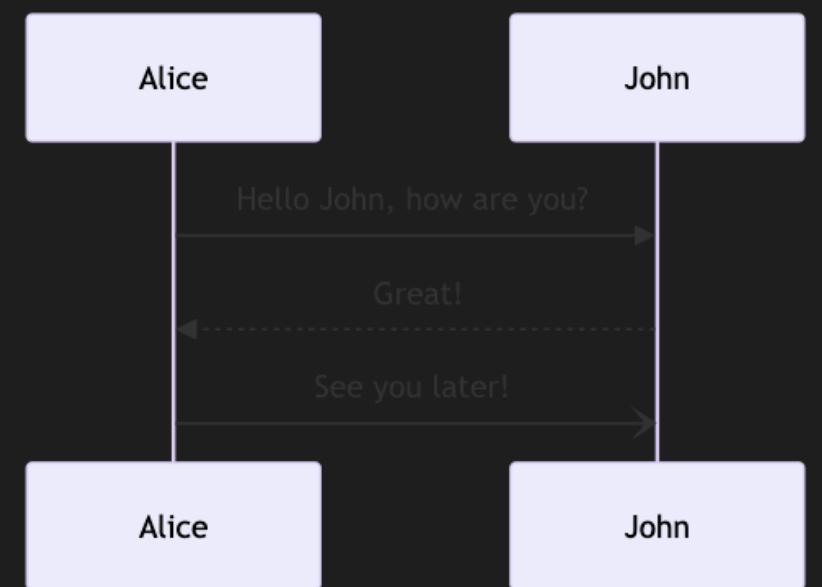
# Step 7/8

- ติดตั้ง Extensions : Markdown Kroki เพื่อช่วยการ preview diagram



# Step 8/8

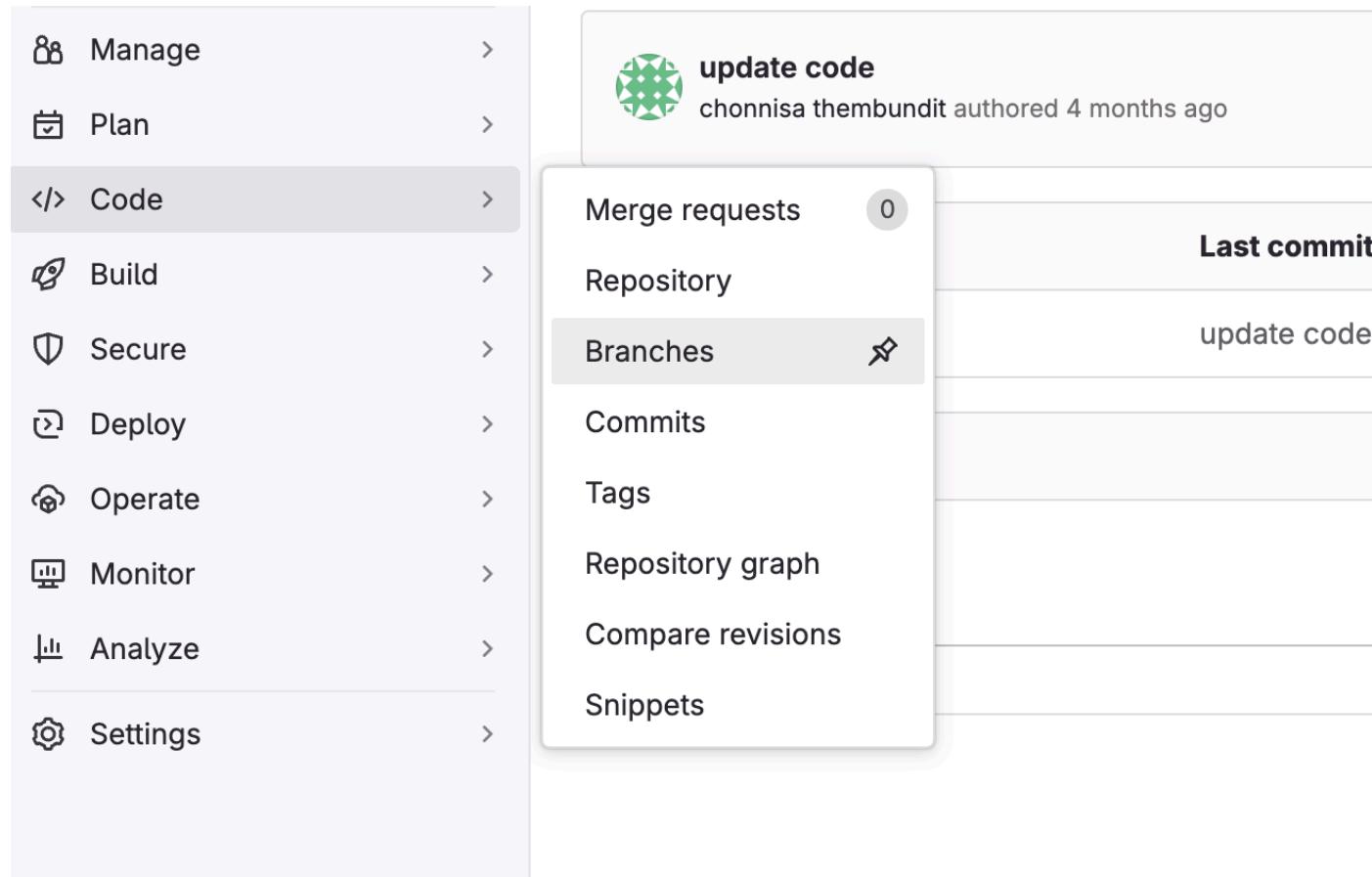
```
```mermaid
sequenceDiagram
    Alice->>John: Hello John, how are you?
    John-->>Alice: Great!
    Alice->>John: See you later!
````
```



## How to create new **BRANCH**

# Step 1/4

- เข้าเมนูเพื่อเข้าสู่หน้าจอ Branch



The screenshot shows a software interface with a sidebar menu on the left and a detailed repository view on the right.

**Sidebar Menu:**

- Manage
- Plan
- Code** (selected)
- Build
- Secure
- Deploy
- Operate
- Monitor
- Analyze
- Settings

**Repository View (Details):**

- update code (by chonnisa thembundit, authored 4 months ago)
- Merge requests: 0
- Last commit: update code
- Branches (highlighted)
- Commits
- Tags
- Repository graph
- Compare revisions
- Snippets

# Step 2/4

- នរបៀប New Branch

pnp / swp\_01 / Branches

Overview   Active   Stale   All

Filter by branch name  Q View branch rules **New branch** :

**Stale branches**

| Branch | Last Commit  | Actions |
|--------|--|---------|
| main   | default protected<br>1cdddf78 · update code · 4 months ago |         |
| dev    | e6b0d99d · add pull with secret · 4 months ago             | 0/12    |

# Step 3/4

- ช่อง Branch name คือชื่อของ branch กี่ต้องการสร้าง
- ช่อง Create from คือชื่อของ branch ต้นทางที่เราต้องการตั้งต้นมา

SOMMAI KRANGPANICH / npru-express / Branches / **New branch**

## New branch

Branch name

Create from

Existing branch name, tag, or commit SHA

Create branchCancel

## Step 4/4

- ใช้คำสั่งดังนี้ใน terminal ของ vs-code เพื่อให้ vs-code ทำการ sync ข้อมูลเข้า branch ใหม่ที่เราได้สร้างขึ้นมาใหม่

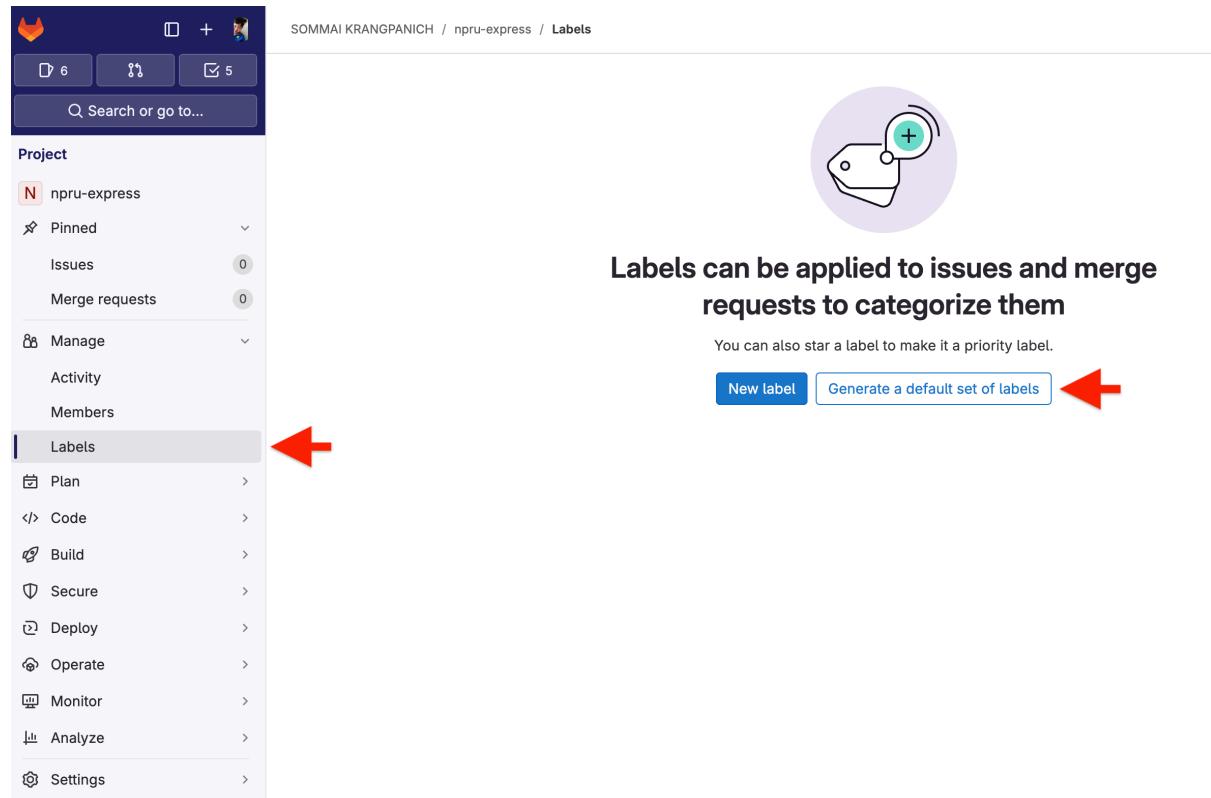
```
git fetch  
git switch <branch name>
```

หมายเหตุ ให้เปลี่ยน `<branch name>` เป็นชื่อ branch ก็ต้องการ sync code ด้วย

Software life cycle management process with  
**GITLAB ISSUE**

# Step 1/8

- ເຂົ້າແມ່ນ Manage / Labels
- ກດປຸນ Generate a default set of labels



SOMMAI KRANGPANICH / npnu-express / Labels

Project

npru-express

Pinned

Issues 0

Merge requests 0

Manage

Activity

Members

Labels

Plan

Code

Build

Secure

Deploy

Operate

Monitor

Analyze

Settings

Labels can be applied to issues and merge requests to categorize them

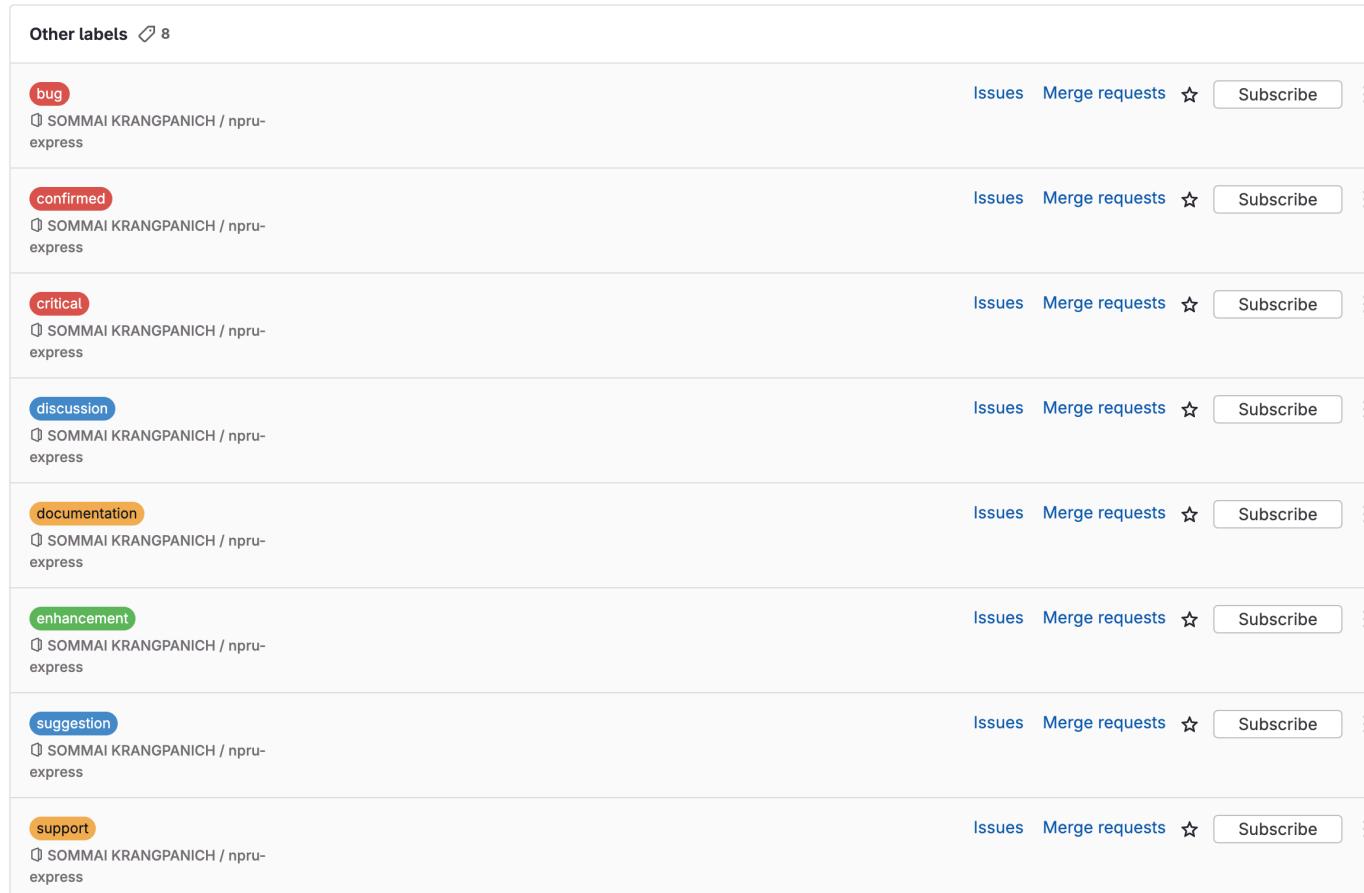
You can also star a label to make it a priority label.

New label

Generate a default set of labels

# Step 2/8

- អត្ថបទ generate



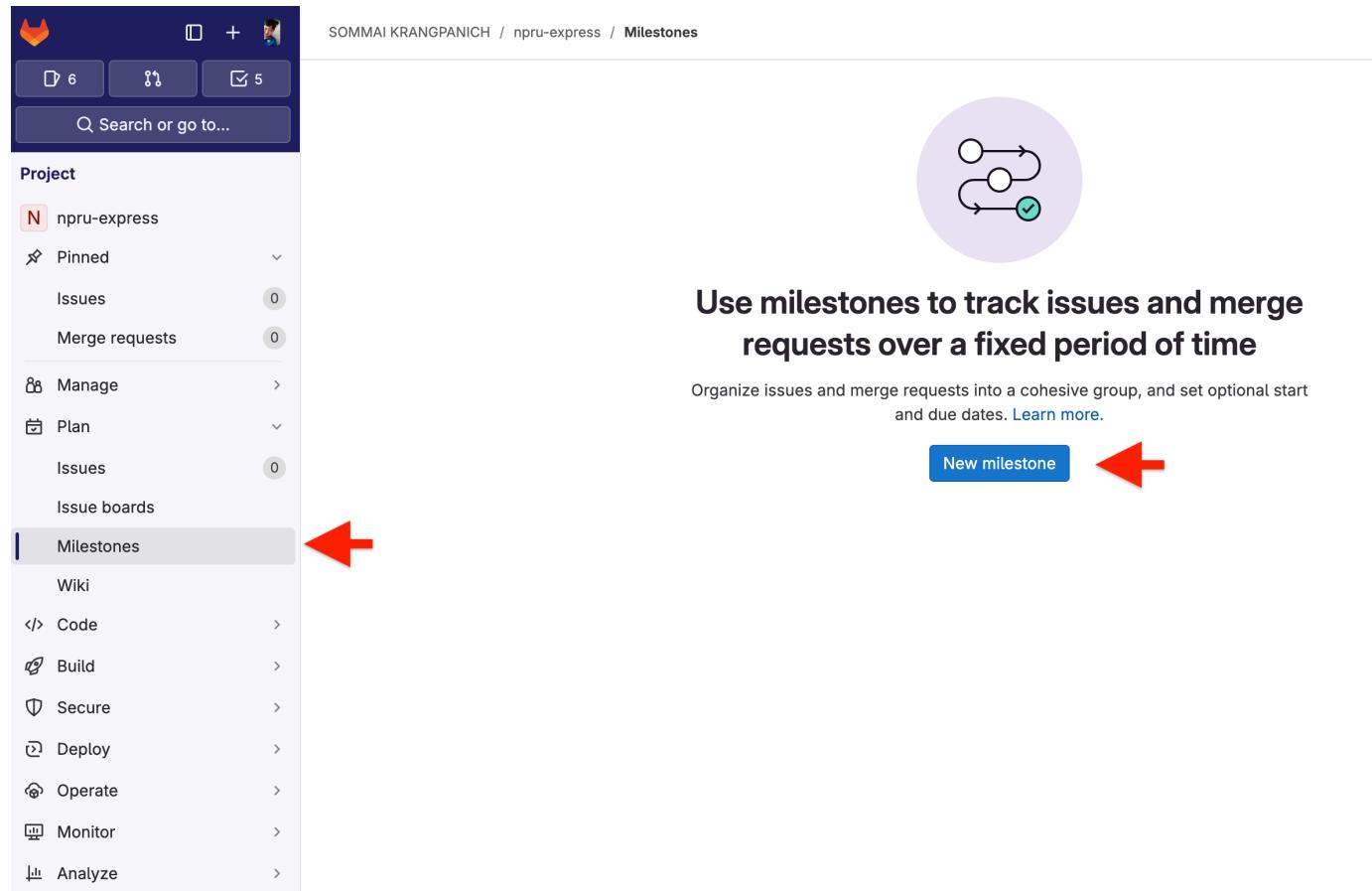
The screenshot shows a list of issues from a GitHub repository. Each issue is categorized by a label:

- bug**: Issues (1), Merge requests (0)
- confirmed**: Issues (1), Merge requests (0)
- critical**: Issues (1), Merge requests (0)
- discussion**: Issues (1), Merge requests (0)
- documentation**: Issues (1), Merge requests (0)
- enhancement**: Issues (1), Merge requests (0)
- suggestion**: Issues (1), Merge requests (0)
- support**: Issues (1), Merge requests (0)

Each item includes a "Subscribe" button and a more options menu (three dots). The repository owner is SOMMAI KRANGPANICH / npru-express.

# Step 3/8

- Create new milestone



The screenshot shows a project management interface with the following elements:

- Header:** SOMMAI KRANGPANICH / npnu-express / Milestones
- Top Bar:** Includes a profile icon, a search bar, and buttons for issues (6), merge requests (5), and a plus sign.
- Project Sidebar:** Lists the project "npru-express" as pinned, with 0 issues and 0 merge requests. It also includes links for "Manage", "Plan", "Issues", and "Issue boards".
- Milestones Section:** The "Milestones" link in the sidebar is highlighted with a red arrow. A large purple circular icon with a checkmark and a curved line is displayed above the text.
- Text:** "Use milestones to track issues and merge requests over a fixed period of time". Below it, a subtext reads: "Organize issues and merge requests into a cohesive group, and set optional start and due dates. [Learn more.](#)"
- Buttons:** A blue "New milestone" button with a red arrow pointing to it.

# Step 3/8

**Title****Start Date** Select start date[Clear start date](#)**Due Date** Select due date[Clear due date](#)**Description**[Preview](#) | [\*\*B\*\*](#) [\*I\*](#) [U](#) | [!\[\]\(264a34b3045d551f9054d3bfef4444c0\_img.jpg\)](#) [!\[\]\(cdb03f5b194b09d81018eca64b1cfb03\_img.jpg\)](#) [!\[\]\(0d025d0c6bbcd9395834d6b45b255bec\_img.jpg\)](#) [!\[\]\(fbf091bd799fcf7bdfe303f264e535dd\_img.jpg\)](#) [!\[\]\(a2c322d0f0179d4960a4bd7d39c7d3ea\_img.jpg\)](#) [!\[\]\(cddfba62a577d025c492344f32556648\_img.jpg\)](#) | [!\[\]\(d83d10f475838b384f0e4a559807c4a7\_img.jpg\)](#) [!\[\]\(01a7277dbea5253f63a90b4165b7e3b2\_img.jpg\)](#)

Write milestone description...

[Switch to rich text editing](#)[Create milestone](#)[Cancel](#)

# Step 4/8

- รายละเอียดของหน้าจอก Milestone

| Field       | Description    | Example                     |
|-------------|----------------|-----------------------------|
| Title       | ชื่อ           | Sprint #01                  |
| Start Date  | วันที่เริ่มต้น | 2023-01-01                  |
| Due Date    | วันที่สิ้นสุด  | 2023-01-31                  |
| Description | รายละเอียด     | รายชื่อ feature ที่จะส่งมอบ |

# Step 5/8

- ผลลัพธ์ของการสร้าง Milestone

Open Milestone Nov 1, 2024–Nov 30, 2024

[Close milestone](#) [⋮](#)

## Sprint #01

Issues 0 Merge requests 0 Participants 0 Labels 0

Unstarted Issues (open and unassigned) 0 Ongoing Issues (open and assigned) 0 Completed Issues (closed) 0

0% complete [»](#)

Start date **Nov 1, 2024** [Edit](#)

Due date **Nov 30, 2024 (26 days remaining)** [Edit](#)

Issues 0 New issue  
Open: 0 Closed: 0

Time tracking  
No estimate or time spent

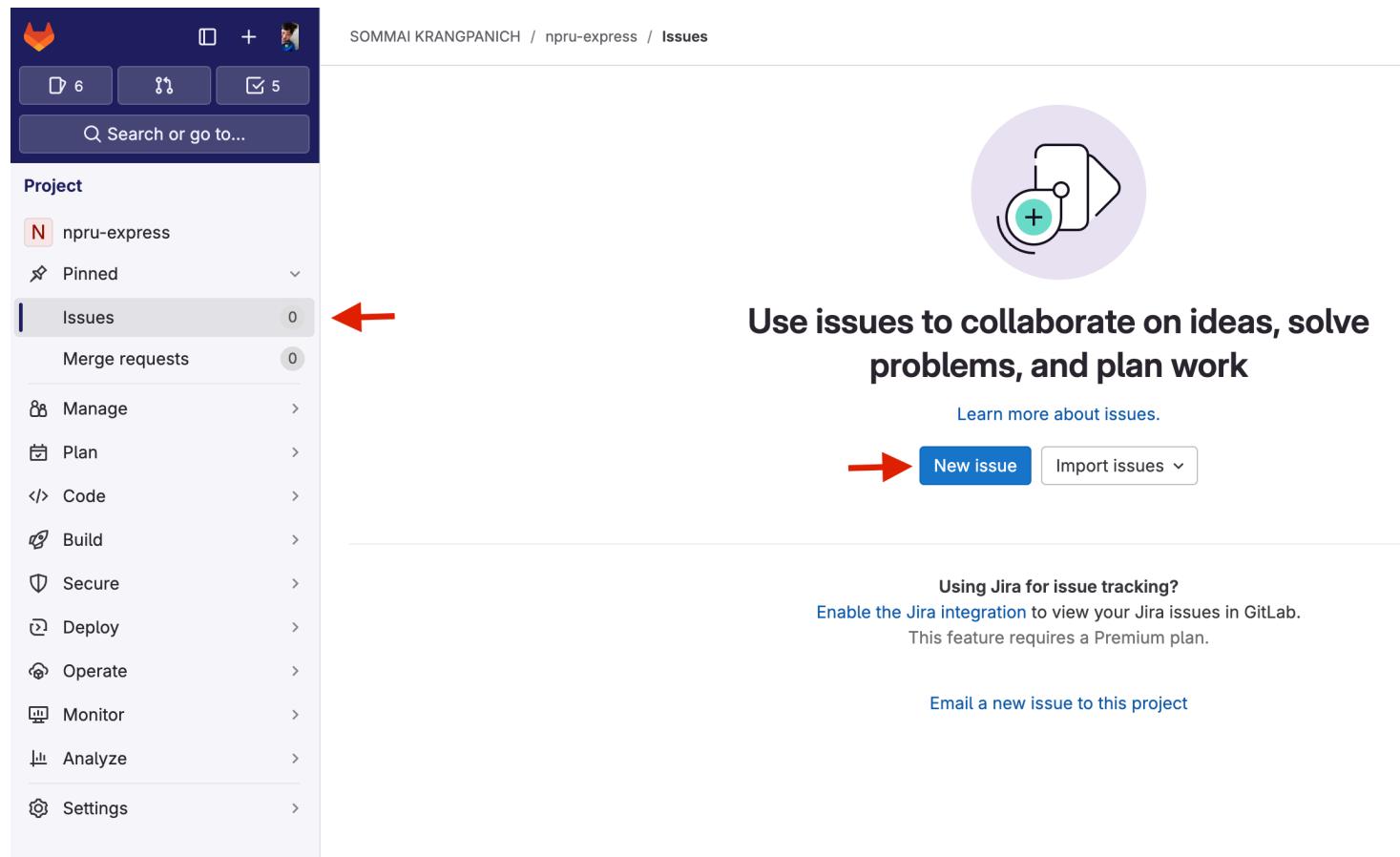
Merge requests 0  
Open: 0 Closed: 0 Merged: 0

Releases  
None

Reference: sommai.k/npru-expr... [🔗](#)

# Step 6/8

- ເຂົ້າເມນຸ plan / issue



The screenshot shows the GitLab interface for the project 'npru-express'. The left sidebar lists various project sections: Issues (selected), Merge requests, Manage, Plan, Code, Build, Secure, Deploy, Operate, Monitor, Analyze, and Settings. The main area displays the 'Issues' page for the user 'SOMMAI KRANGPANICH'. It shows 6 open issues and 5 closed issues. A large callout in the center of the page reads: 'Use issues to collaborate on ideas, solve problems, and plan work'. Below this, there's a link to 'Learn more about issues.' and two buttons: 'New issue' (highlighted with a red arrow) and 'Import issues'. A note at the bottom mentions 'Using Jira for issue tracking?' and instructions to enable the Jira integration.

SOMMAI KRANGPANICH / npru-express / Issues

Project

npru-express

Pinned

Issues 0

Merge requests 0

Manage

Plan

Code

Build

Secure

Deploy

Operate

Monitor

Analyze

Settings

6 5

Search or go to...

Use issues to collaborate on ideas, solve problems, and plan work

New issue Import issues

Using Jira for issue tracking?  
Enable the [Jira integration](#) to view your Jira issues in GitLab.  
This feature requires a Premium plan.

Email a new issue to this project

# Step 7/8

- Create new issue

Title (required)

Type 

Description

Preview |                                                                           <img alt="dropdown icon" data-bbox="

# Step 8/8

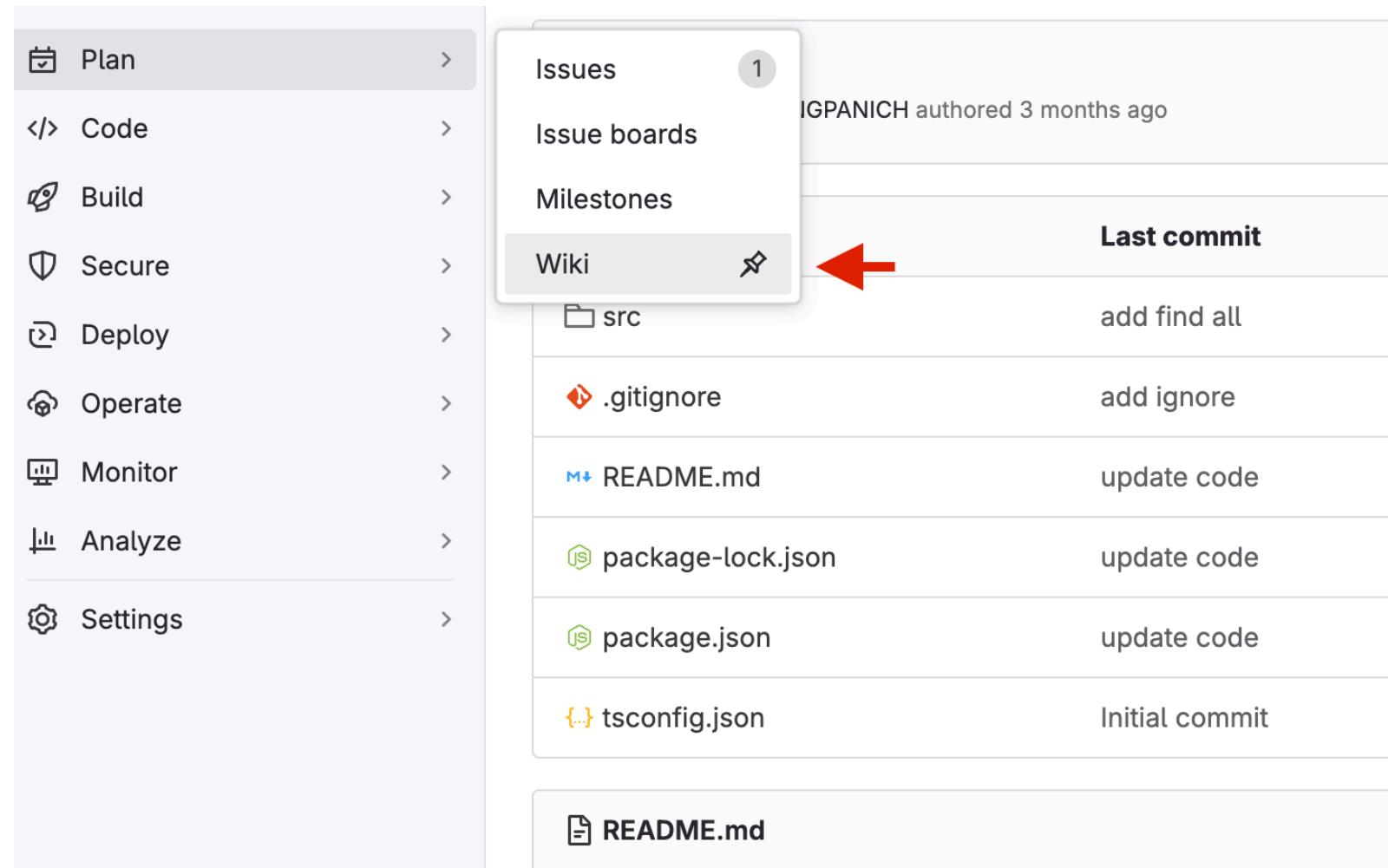
- รายละเอียดของหน้าจอ Issue

| Field       | Description   | Example                     |
|-------------|---------------|-----------------------------|
| Title       | ชื่อ          | สร้างหน้าจอ Login           |
| Type        | ประเภท        | Issue                       |
| Description | รายละเอียด    | รายชื่อ feature ที่จะส่งมอบ |
| Assignee    | ผู้รับผิดชอบ  |                             |
| Due Date    | วันที่สิ้นสุด |                             |
| Milestone   | กรอบเวลา      |                             |
| Labels      | tag ประเภท    | bug                         |

Knowledge sharing process with

**GITLAB WIKI**

# Step 1/3



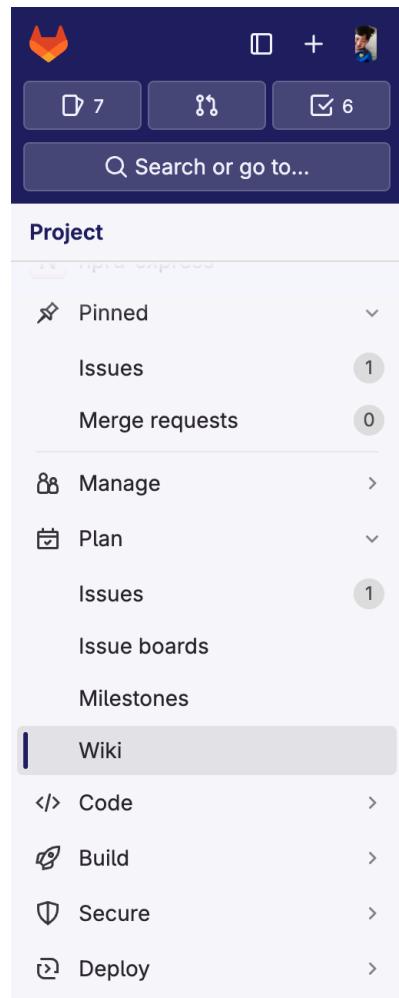
The screenshot shows a software interface for a research and development team. On the left, a vertical sidebar lists various project management and development tasks:

- Plan
- Code
- Build
- Secure
- Deploy
- Operate
- Monitor
- Analyze
- Settings

The "Wiki" item in the sidebar is highlighted with a red arrow pointing to it. The main panel displays a list of recent commits:

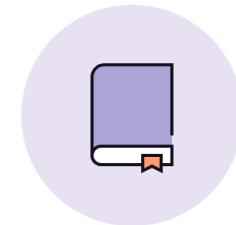
| File              | Message        |
|-------------------|----------------|
| src               | add find all   |
| .gitignore        | add ignore     |
| README.md         | update code    |
| package-lock.json | update code    |
| package.json      | update code    |
| tsconfig.json     | Initial commit |
| README.md         |                |

# Step 2/3



The screenshot shows the GitLab interface. At the top, there's a header with a user icon, a search bar, and project navigation. The main area has a sidebar on the left with sections like Project, Manage, Plan, Issues, and Milestones. A specific section for Wiki is highlighted with a blue background.

SOMMAI KRANGPANICH / npnu-express / Wiki



## Get started with wikis

Use GitLab Wiki to collaborate on documentation in a project or group. You can store wiki pages written in markup formats like Markdown or AsciiDoc in a separate Git repository, and access the wiki through Git, the GitLab web interface, or the API. Have a Confluence wiki already? Use that instead.



Create your first page

Enable the Confluence Wiki integration

# Step 3/3

**Title**

home

**Path**

home

 Generate page path from title**Format**

Markdown

**Content**Preview | **B** *I* ~~S~~ |                                        

Write your content or drag files here...

Switch to rich text editing

To link to a (new) page, simply type [Link Title](page-slug). More examples are in the [documentation](#).**Commit message**

Create home

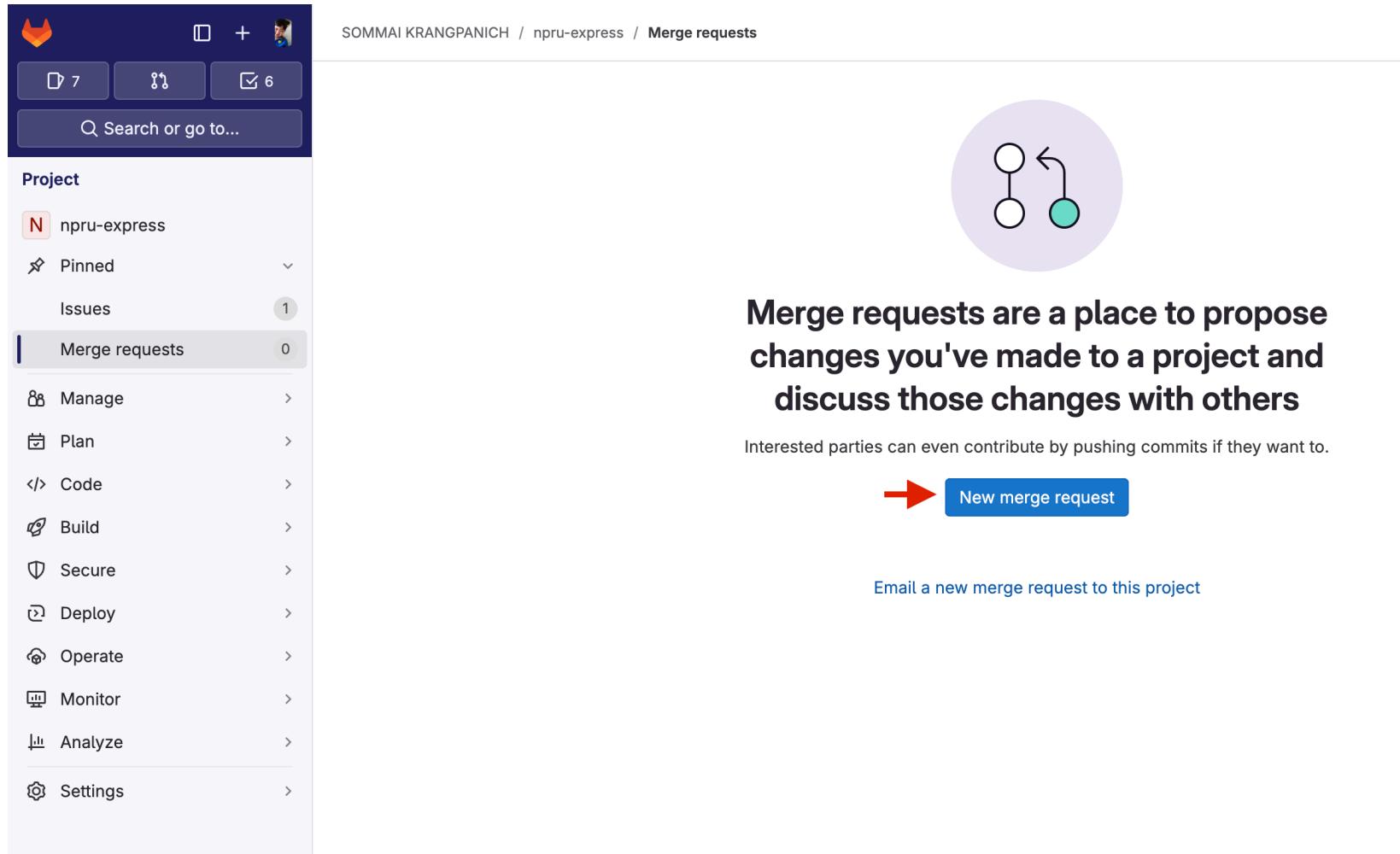
**Create page**

Cancel

How to use

# MERGE REQUEST PROCESS

# Step 1/5



The screenshot shows a user interface for managing a GitLab project. The top navigation bar includes a profile icon, a search bar, and project-related buttons. The main area displays a sidebar with project navigation links like 'Project', 'Merge requests', 'Issues', 'Pinned', and various operational tabs such as 'Manage', 'Plan', 'Code', 'Build', 'Secure', 'Deploy', 'Operate', 'Monitor', 'Analyze', and 'Settings'. The 'Merge requests' link is currently selected, indicated by a blue border. The main content area shows a large purple circular icon containing two nodes connected by a curved arrow, symbolizing collaboration or a workflow. Below the icon, a bold text box states: 'Merge requests are a place to propose changes you've made to a project and discuss those changes with others'. A smaller text below it says: 'Interested parties can even contribute by pushing commits if they want to.' A prominent blue button with a red arrow pointing right is labeled 'New merge request'.

SOMMAI KRANGPANICH / npru-express / Merge requests

Project

npru-express

Pinned

Issues 1

Merge requests 0

Manage >

Plan >

Code >

Build >

Secure >

Deploy >

Operate >

Monitor >

Analyze >

Settings >

Merge requests are a place to propose changes you've made to a project and discuss those changes with others

Interested parties can even contribute by pushing commits if they want to.

New merge request

Email a new merge request to this project

# Step 2/5

## New merge request

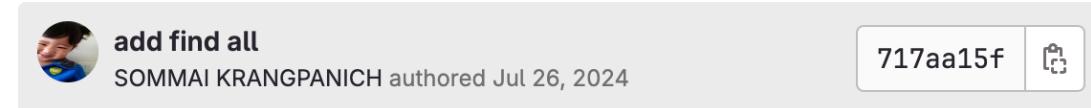
### Source branch

sommai.k/npru-express

▼

dev

▼



[Compare branches and continue](#)

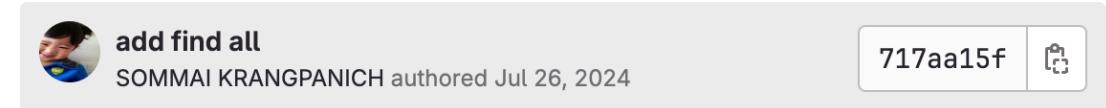
### Target branch

sommai.k/npru-express

▼

main

▼



# Step 3/5

## New merge request

From dev into main [Change branches](#)

### Title (required)

Draft: Dev

**Mark as draft**

Drafts cannot be merged until marked ready.

### Description

Preview | **B** *I* ~~S~~ |        |   

↗

Describe the goal of the changes and what reviewers should be aware of.

Switch to rich text editing

↗

Add [description templates](#) to help your contributors to communicate effectively!

# Step 4/5

**Assignee** [Assign to me](#)**Reviewer** **Milestone** **Labels** **Merge options**

- Delete source branch when merge request is accepted.
- Squash commits when merge request is accepted. 

[Create merge request](#)[Cancel](#)

# Step 5/5



8✓ [Approve](#) Approval is optional [?](#)

 Ready to merge!

Delete source branch  Squash commits [?](#)  Edit commit message

1 commit and 1 merge commit will be added to main (squashes 1 commit).

[Merge](#)

## Activity

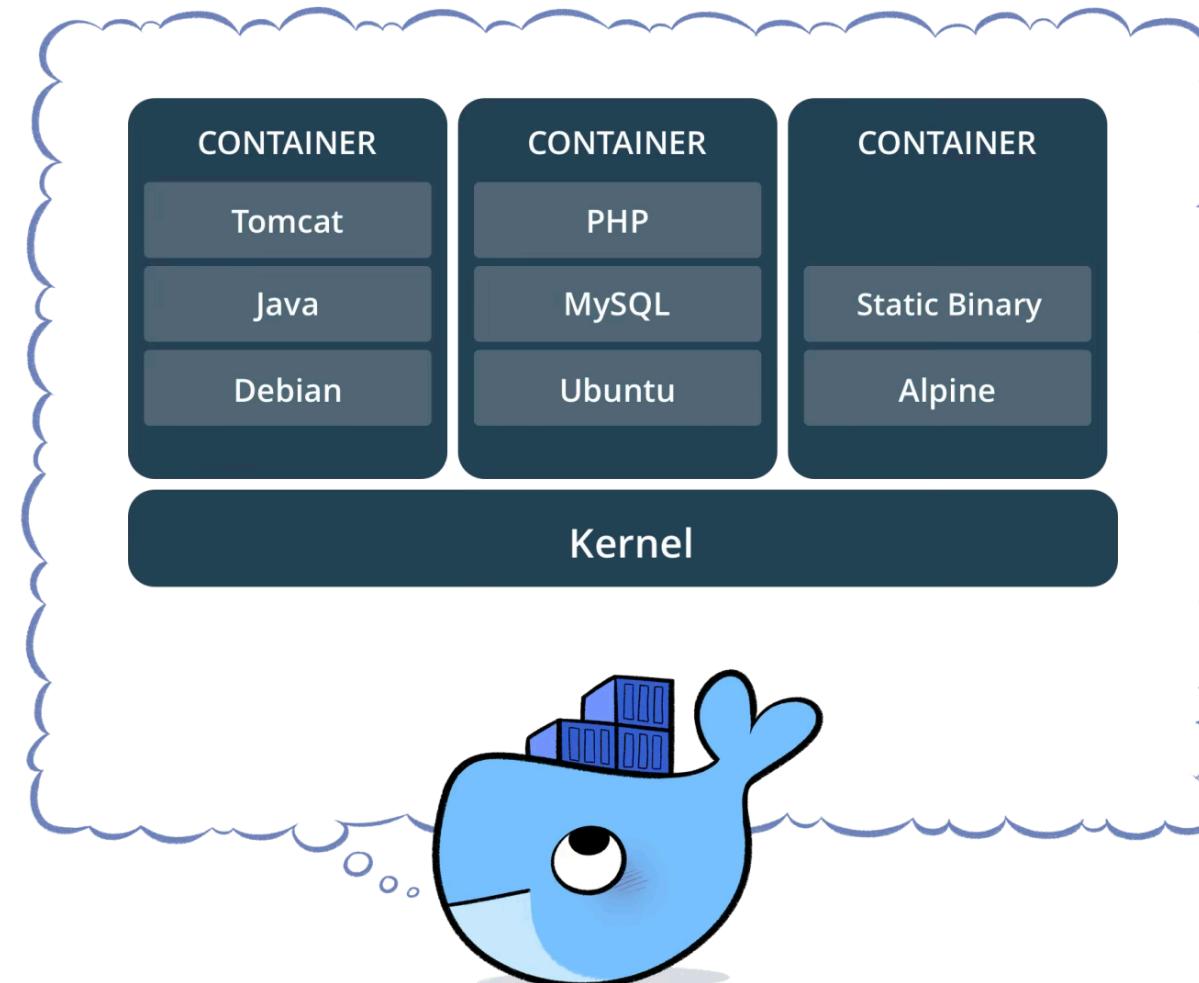
All activity [▼](#) [↑](#)

- SOMMAI KRANGPANICH changed milestone to [%Sprint #01](#) 2 minutes ago
- SOMMAI KRANGPANICH added [confirmed](#) label 2 minutes ago
- SOMMAI KRANGPANICH requested review from [@sommai.k](#) 2 minutes ago
- SOMMAI KRANGPANICH assigned to [@sommai.k](#) 2 minutes ago
- SOMMAI KRANGPANICH added 1 commit just now
  - [87fc69df](#) - update readme

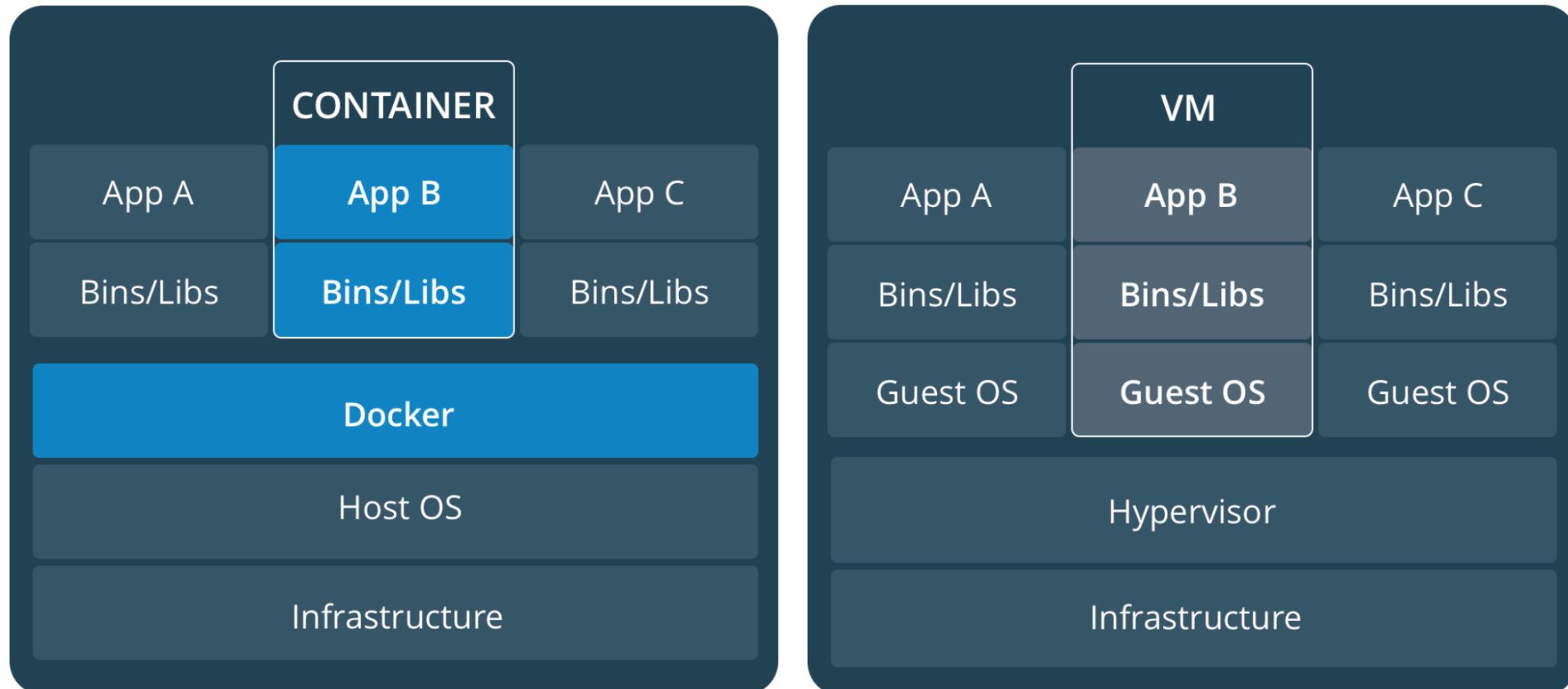
[Compare with previous version](#)

Containers virtualize with  
**DOCKER**

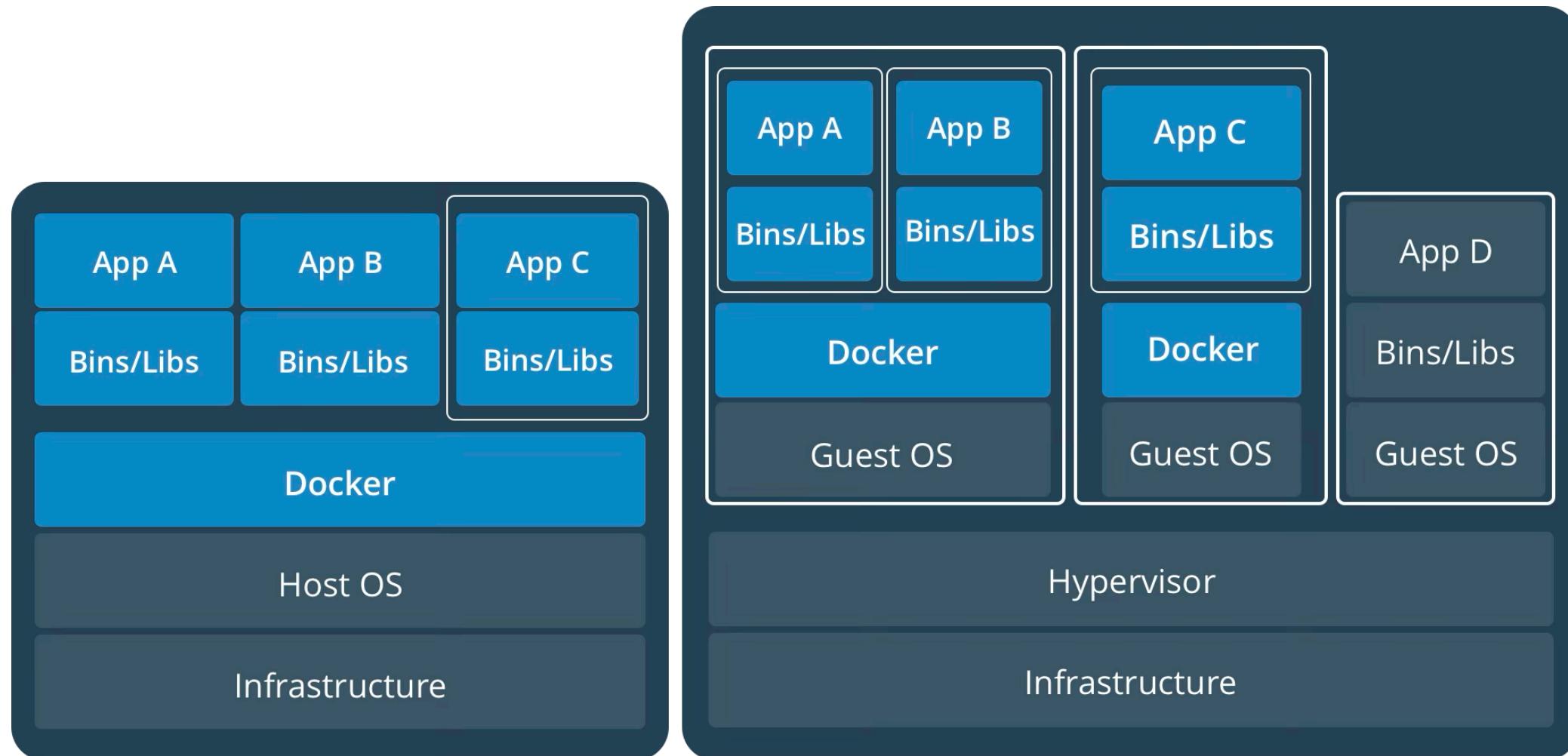
# About Container



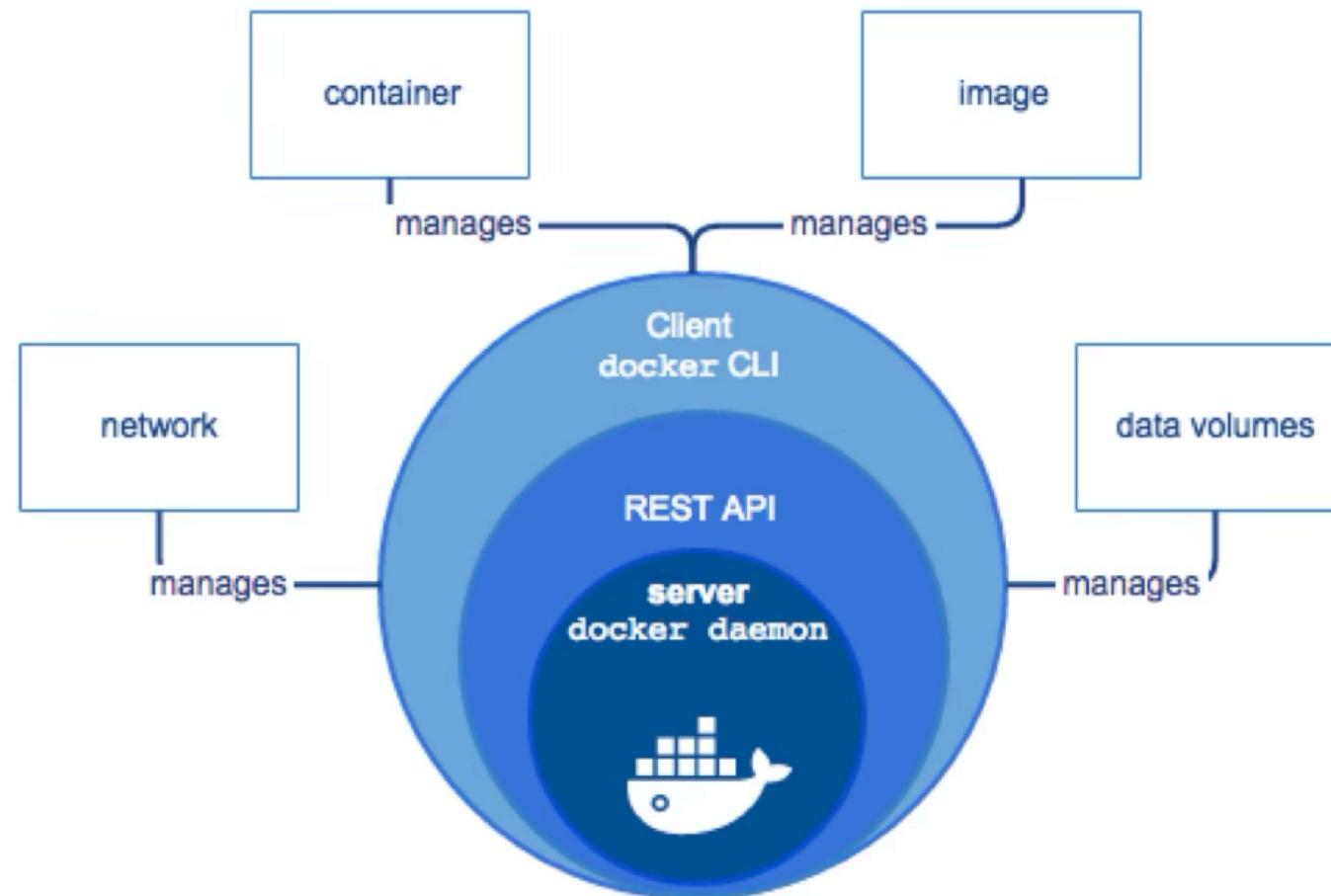
# Comparing Containers and Virtual Machines



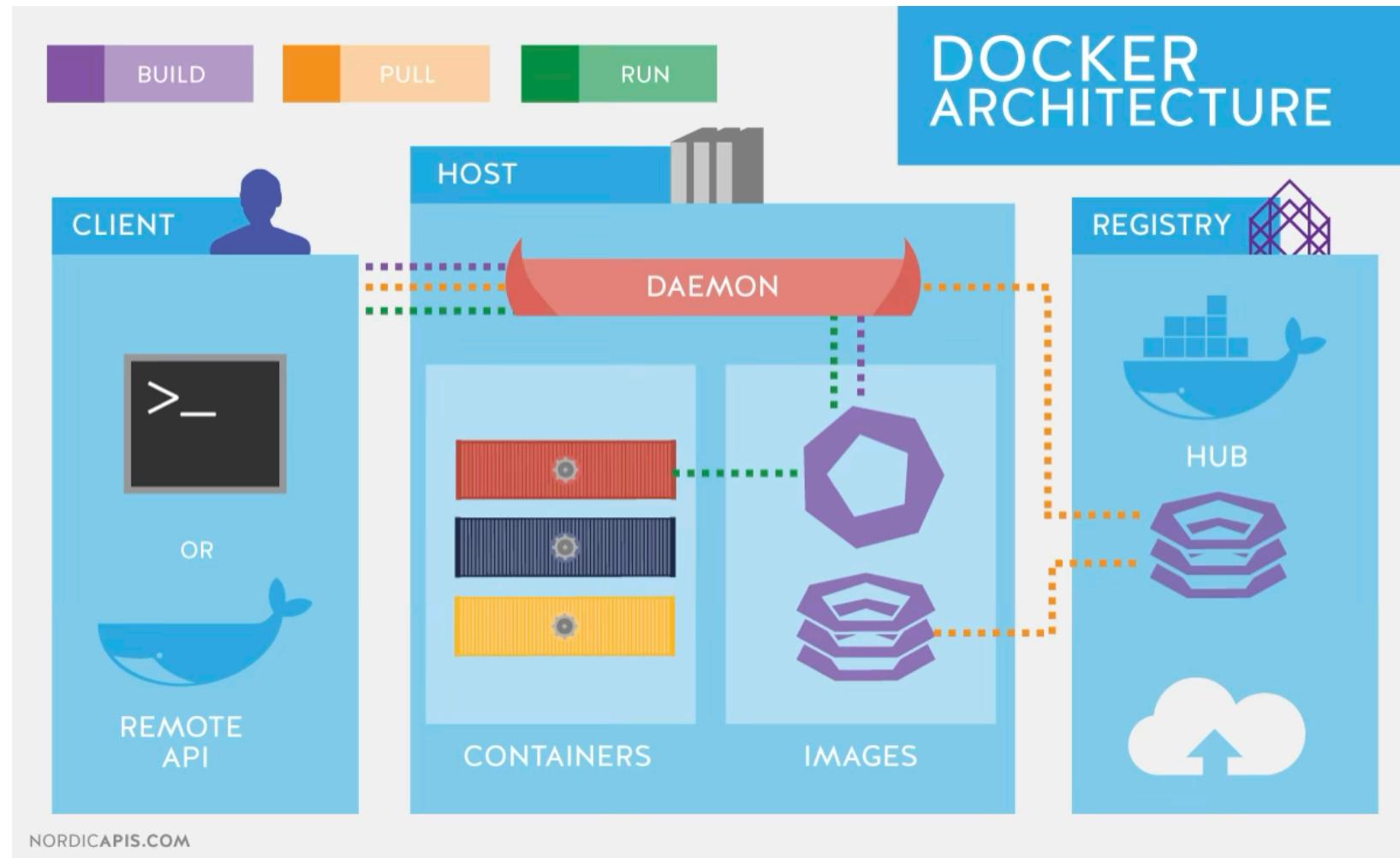
# Containers and Virtual Machines Together



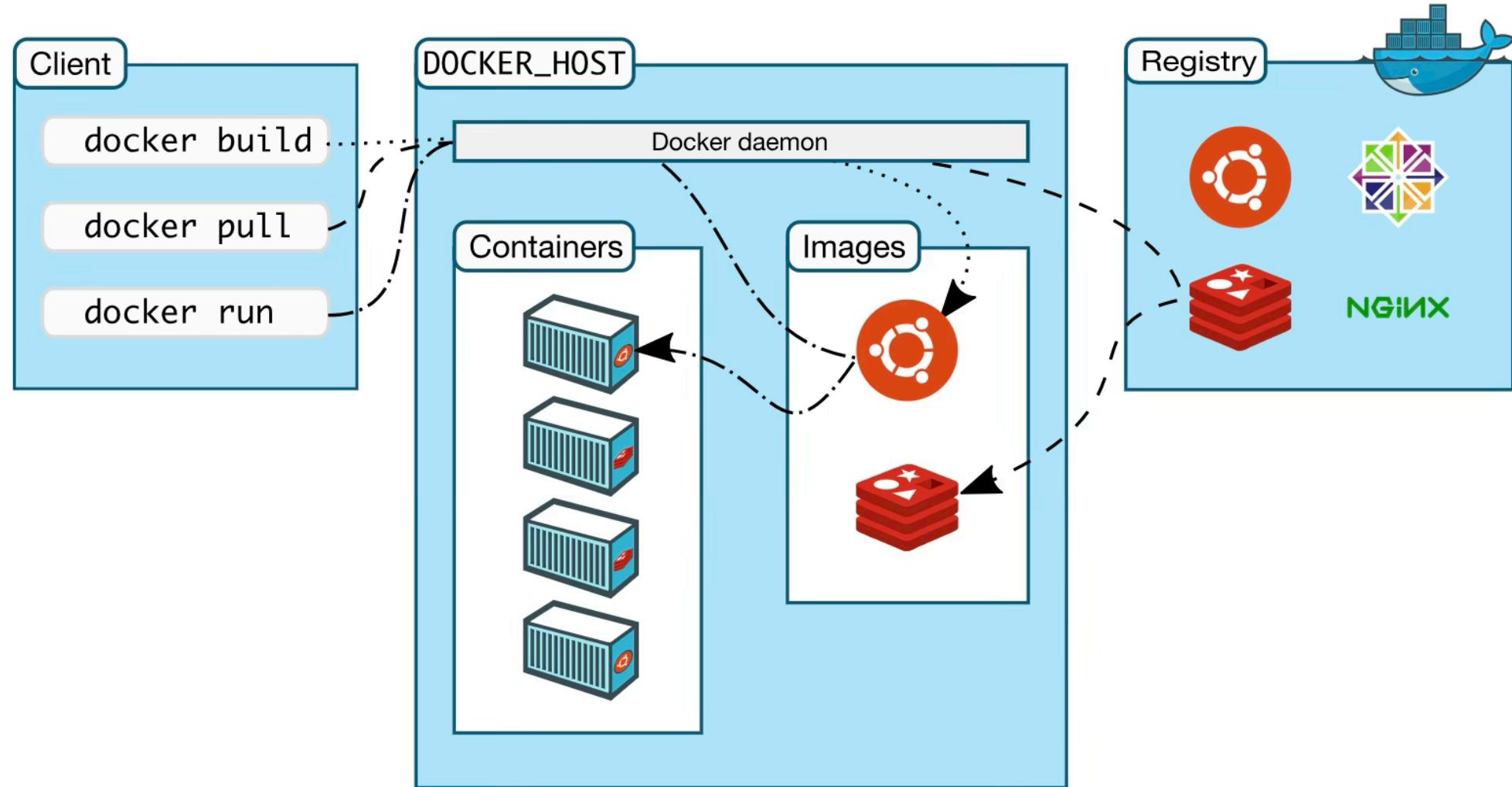
# Docker engine



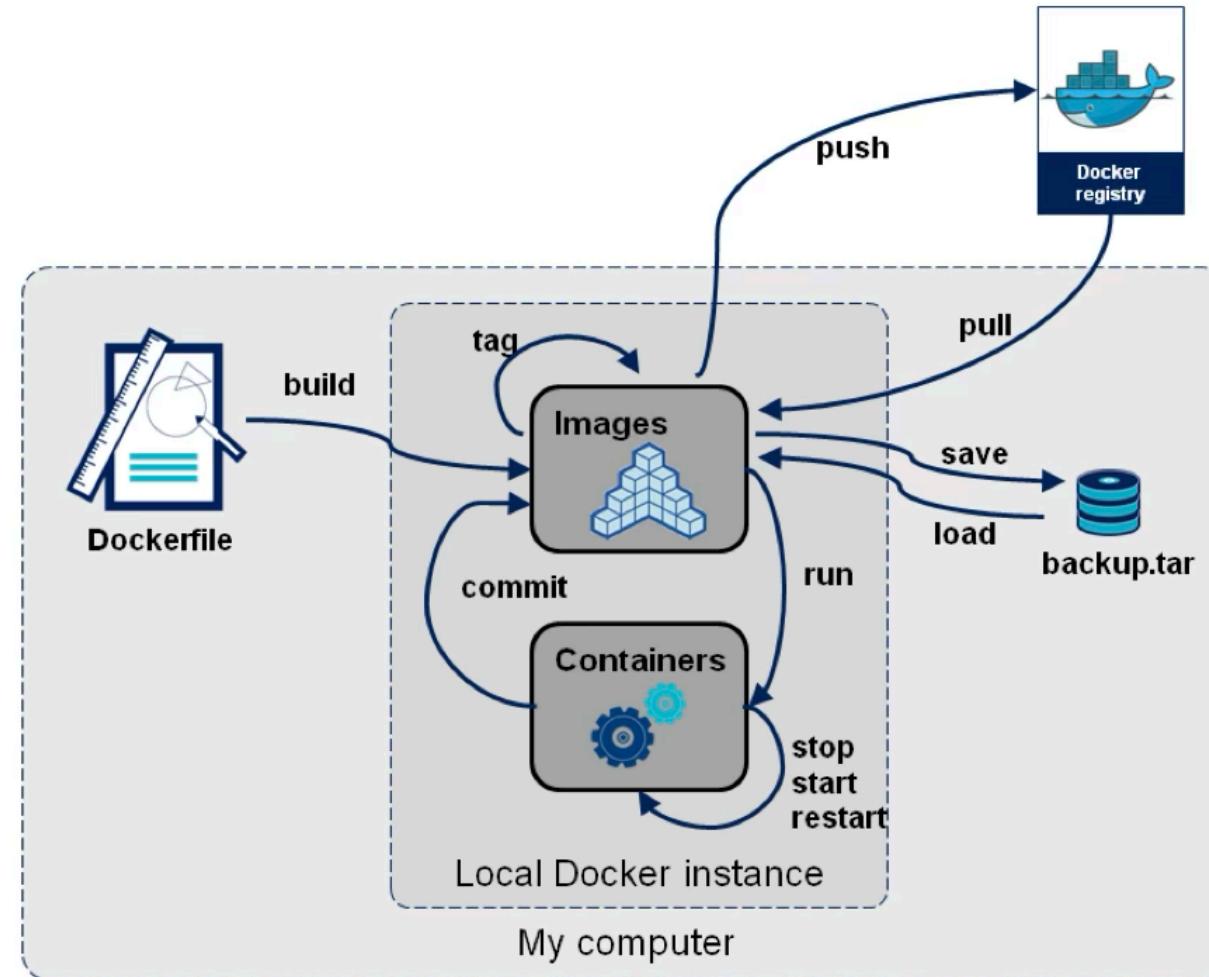
# Docker Architecture



# Docker Architecture #2



# Docker Architecture #3



# Docker Workshop

# Hello World Docker

ເປີດ cmd ແລ້ວຮັນຄຳສັ່ງດັ່ງນີ້

```
docker run --name some-nginx -p 9080:80 -d nginx
```

| link ສໍາຫຼັບຫາ image <https://hub.docker.com>

ເປີດ browser ເຂົ້າ url ດັ່ງນີ້ <http://localhost:9080>

# สร้าง file index.html

- สร้าง folder ชื่อ html
- สร้าง file index.html และใส่ code ดังนี้

```
<h1>Hello World</h1>
```

- เปิด cmd แล้ว run คำสั่งดังนี้

```
docker run --name my-nginx -p 9081:80 -d -v <your project path>:/usr/share/nginx/html nginx
```

เปิด browser เข้า url ดังนี้ <http://localhost:9081>

# Docker Command

## Login

```
docker login
docker login -u <user_name>
docker login -u <user_name> -p <password>
```

## Logout

```
docker logout
```

## List all image

```
docker images
docker image ls
```

# Docker Command #2

## Search image

```
docker search <image name>
```

## Pull image

```
docker pull <image name>
```

## Create container from image

```
docker create <options> <image name> --name -v -p  
docker create --name ubuntu14 -v /user/sommaik:/home ubuntu:14.04
```

# Docker Command #3

## Start Container

```
docker start <container_id> or <container_name>
```

## Stop Container

```
docker stop <container_id> or <container_name>
```

## Stop all container

```
docker stop $(docker ps -a -q)
```

## List all container

```
docker ps <options>
```

# Docker Command #4

## Pause Container

```
docker pause <container_id> or <container_name>
```

## Unpause Container

```
docker unpause <container_id> or <container_name>
```

## Exec Container

```
docker exec -it <container_id> bash
```

## Inspect Container

```
docker inspect <container_id>
```

# Docker Command #5

## Logs container

```
docker logs <options>
```

## Commit Container

```
docker commit <container_id> <new_image_name>
docker commit 2x5t aloha
```

## Push Image

```
docker push <account>/<image name>
```

## Tag

```
docker tag ubuntu ubuntu-x
```

# Docker Command #6

## Export container

```
docker export <container_id> > <to_path>
```

## Import container

```
docker import - <from_path>
```

## Save Image

```
docker save <image name> > <to path>
docker save <image name>:<tag> > <to path>
```

## Load Image

```
docker load < <from path>
```

# Docker Command #7

Remove container

```
docker rm <container_id>
```

Remove all stop container

```
docker rm $(docker ps -a -q)
```

Remove Image

```
docker rmi <image_id>
```

# Docker Network

## List all network

```
docker network ls
```

## Create new network

```
docker network create <network_name> default bridge
docker network create --subnet 10.0.0.1/24 <network_name>
docker network create my-net (create images networks)
```

## Inspect network

```
docker network inspect <network_name> or <container_id>
```

## Use network

```
docker run --network <network_name> <image_name>
docker run -it --name <container_name> --net--alias alias2 --network <network_name> <image_name>
```

# Docker run command parameter

Parameter	Description
-d	Run in the background
--name	Create name to container is running
-p	Port mapping (local_port:container_port)
-h	Container host name
-e	Environment
-v	Map volume paths
-i	Keep STDIN open even if not attached
-t	Allocate a pseudo-TTY
--network	Use network

How to create Dockerfile for

# BUILD CUSTOM DOCKER IMAGE

# Dockerfile

FROM

```
FROM <image>[:<tag>]  
FROM ubuntu:14.04
```

RUN

```
RUN <command>  
RUN echo "Hello World"
```

EXPOSE

```
EXPOSE <port>  
EXPOSE 8080
```

COPY

```
COPY <local_path> <container_path>  
COPY ./tomcat/context.xml /usr/local/tomcat/conf
```

# Dockerfile #2

## ENV

```
ENV <key> <value>
```

## CMD

```
CMD command param1 param2
```

## WORKDIR

```
WORKDIR /path/to/workdir
```

## VOLUME

```
VOLUME /path
```

# Dockerfile #3

## Build

```
docker build <option> <path> -t <tagname> .
```

## Example

```
docker build -t first .
```

# Works shop การสร้าง docker image

- สร้าง file ชื่อ Dockerfile

```
FROM nginx:alpine
COPY ./html /usr/share/nginx/html
```

- run คำสั่งเพื่อ build image

```
docker build -t my-first-image .
```

- ตรวจสอบผลการ build

```
docker image ls my-first-image
```

Container orchestration with  
**DOCKER COMPOSE**

# Docker compose

file name

```
docker-compose.yaml
```

Example

```
services:  
  hello:  
    image: nginx:alpine  
    ports:  
      - 9085:80
```

# docker compose cli

Start docker compose

```
docker compose up -d --build
```

Start docker compose with create new

```
docker compose up -d --force-recreate
```

List all container

```
docker compose ps
```

Stop all container

```
docker compose stop
```

# docker compose cli #2

Delete all container

```
docker compose rm
```

Stop and delete all container

```
docker compose down
```

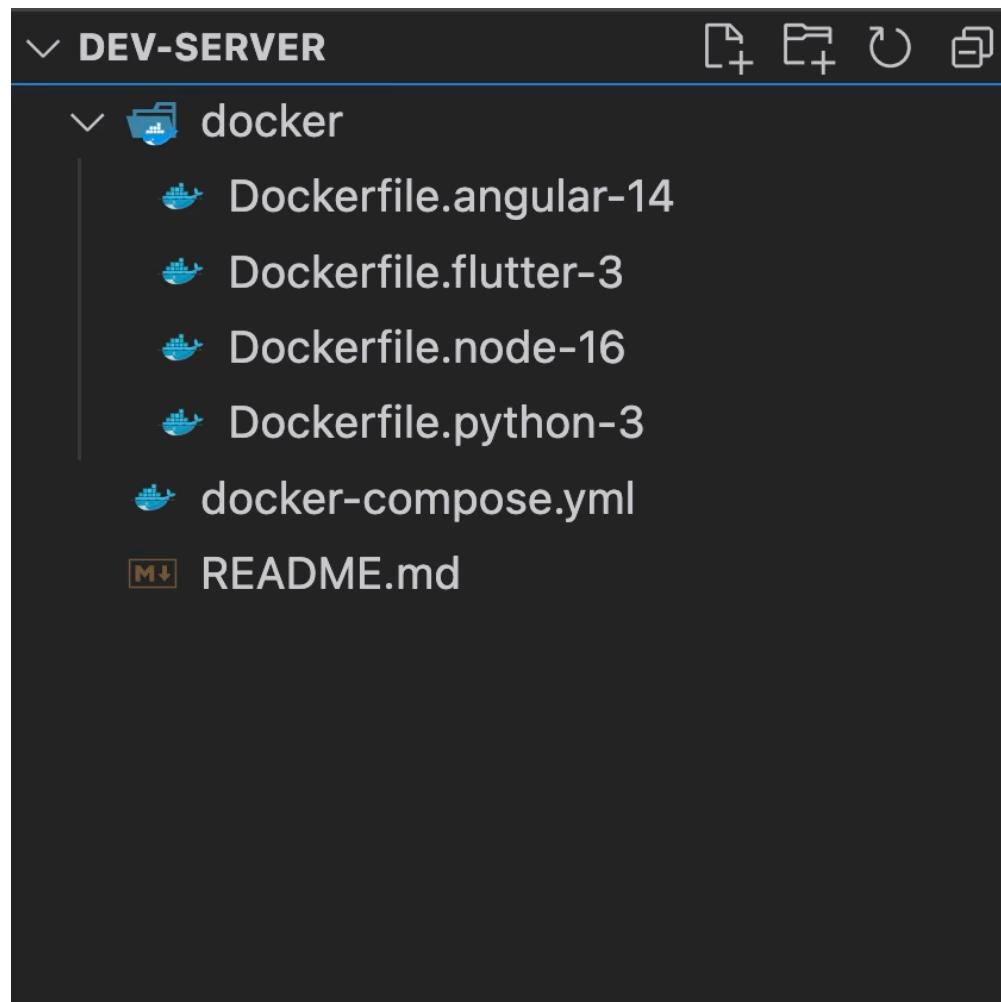
List all compose

```
docker compose list
```

How to

# REMOTE DEVELOPMENT

# Step 1 Create docker folder and file



## Step 2 Create Docker.node-18 file

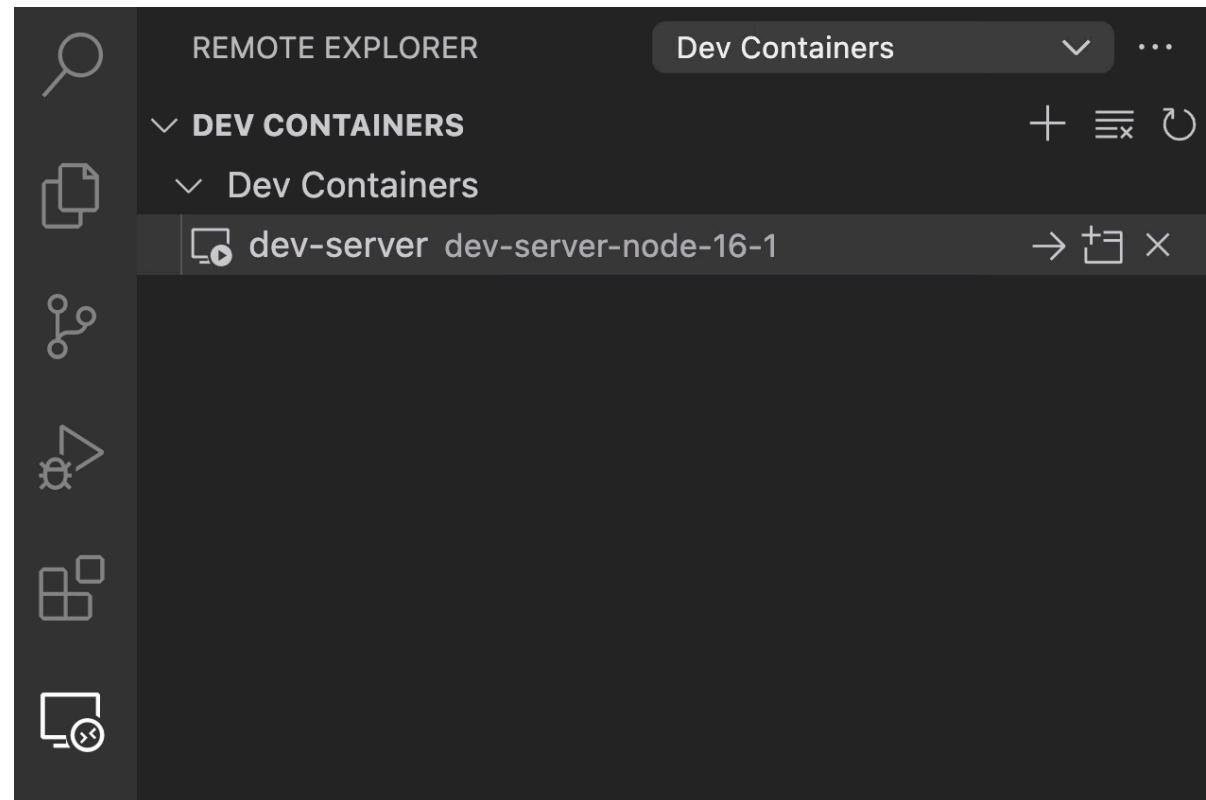
```
FROM node:18-alpine
RUN apk add git
```

# Step 3 Create docker-compose.yml

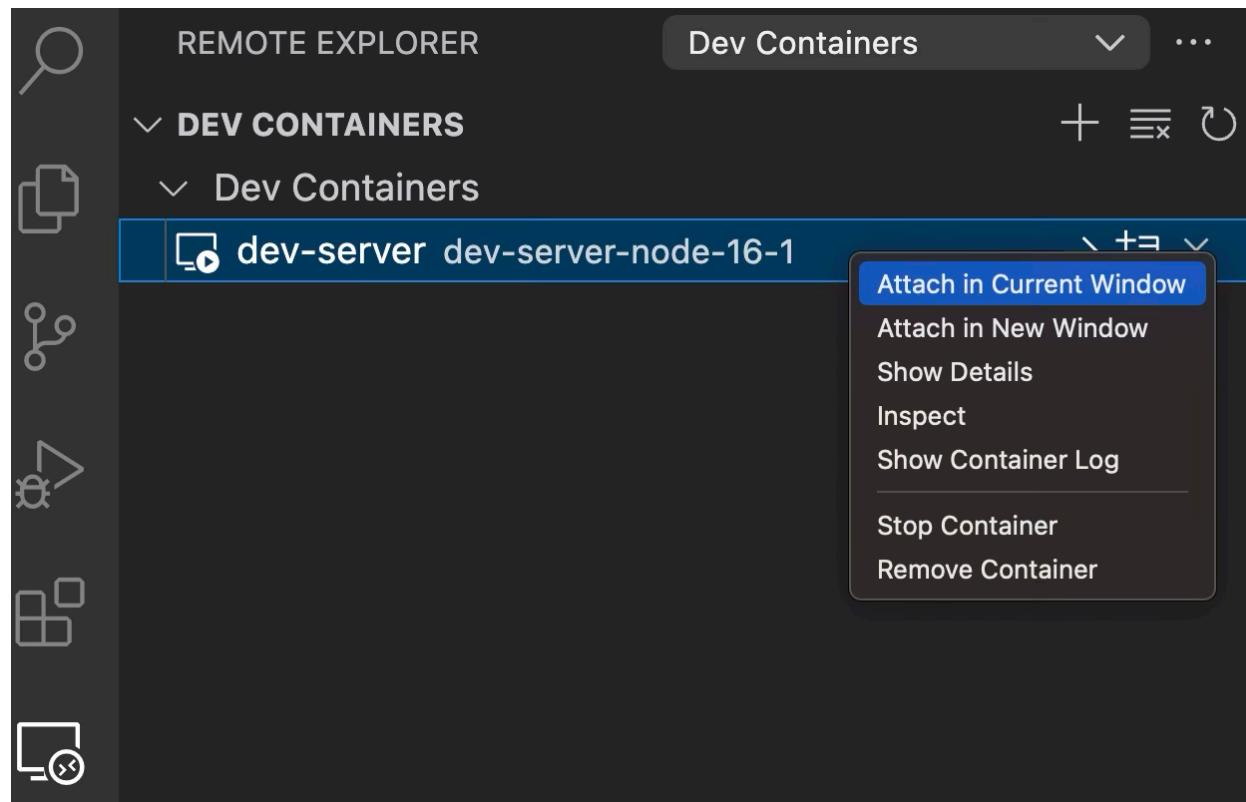
```
volumes:  
  dev_data:  
services:  
  node-18:  
    build:  
      context: .  
      dockerfile: docker/Dockerfile.node-18  
    restart: on-failure  
    command: ["sleep", "infinity"]  
    volumes:  
      - dev_data:/home/dev/source
```

# Step 4 Connect to container

```
docker compose up -d node-18
```



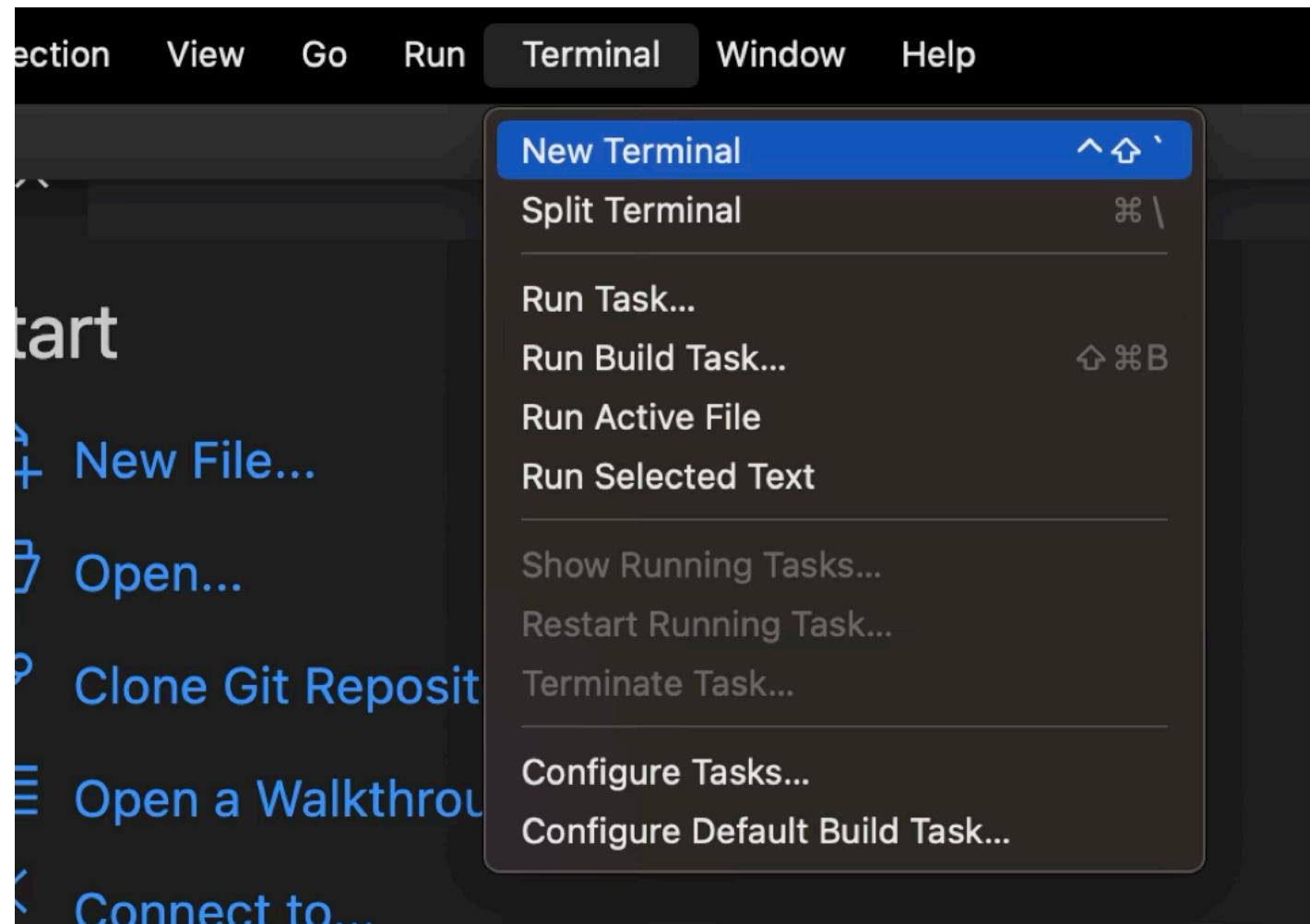
# Step 5 Attach in current window



# Step 6 Connect to container success



# Step 7 Open terminal



# Step 8 Tunnel

[PROBLEMS](#)[OUTPUT](#)[DEBUG CONSOLE](#)[PORTS](#)[TERMINAL](#)

No forwarded ports. Forward a port to access your running services locally.

[Forward a Port](#)

How to debug

# CUSTOM DOCKER IMAGE

# Step to debug

- run คำสั่งเพื่อหาชื่อ container กี่ต้องการจะ debug

```
docker ps -a
```

- run คำสั่งเพื่อเข้าไปทำงานภายใน container

```
docker exec -it <container-name> bash
```

หมายเหตุ เปลี่ยน `<container-name>` เป็นชื่อ container กี่ต้องการเข้าไป debug

- หลังจากนั้นเราจะสามารถใช้คำสั่งของ container นั้น ๆ ได้
- ในการลนีที่ใช้งานเสร็จแล้วให้ออกโดยใช้คำสั่ง `exit`

How to push custom docker image to  
**DOCKER REGISTRY**

# Step

- login เข้าไปที่ registry server

```
docker login <registry-server>
```

- ตั้งชื่อ image ให้ตรงกับ container registry pattern ดังนี้ `<registry-server>/<owner>/<project-name>:<tag>`
- ในกรณีชื่อไม่ตรงตาม pattern ให้ใช้คำสั่งนี้ในการสร้างให้ชื่อตรงกัน

```
docker tag <old-name> <new-name>
```

- ใช้คำสั่งนี้ในการนำ image ขึ้นไปเก็บบน registry

```
docker push <image-name>:<tag>
```

หมายเหตุ ต้องเปลี่ยน `<image-name>` เป็นชื่อ image ตัวที่เราต้องการนำขึ้นไป ส่วน `<tag>` ถ้าไม่ระบุจะระบบจะค่า default เป็น `latest`

Introduction to containerization

# WHAT IS CONTAINERIZATION

# What is containerization

- Each container that you run is repeatable; the standardization from having dependencies included means that you get the same behavior wherever you run it.
- Containers decouple applications from underlying host infrastructure. This makes deployment easier in different cloud or OS environments.
- Each node in a Kubernetes cluster runs the containers that form the Pods assigned to that node. Containers in a Pod are co-located and co-scheduled to run on the same node

Introduction to containerization

# WHAT IS KUBERNETES

# What is Kubernetes

- Kubernetes is an open-source system that automates container deployment tasks. It was originally developed at Google but is now maintained as part of the Cloud Native Computing Foundation (CNCF).
- Kubernetes has risen to prominence because it solves many of the challenges around using containers in production. It makes it easy to launch limitless container replicas, distribute them across multiple physical hosts, and set up networking so users can reach your service.
- Most developers begin their container journey with Docker. While this is a comprehensive tool, it's relatively low-level and relies on CLI commands that interact with one container at a time. Kubernetes provides much higher-level abstractions for defining applications and their infrastructure using declarative schemas you can collaborate on.

Introduction to containerization

# KUBERNETES FEATURES

# Kubernetes features

Kubernetes has a comprehensive feature set that includes a full spectrum of capabilities for running containers and associated infrastructure:

- **Automated rollouts, scaling, and rollbacks** – Kubernetes automatically creates the specified number of replicas, distributes them onto suitable hardware, and takes action to reschedule your containers if a node goes down. You can instantly scale the number of replicas on-demand or in response to changing conditions such as CPU usage.
- **Service discovery, load balancing, and network ingress** – Kubernetes provides a complete networking solution that covers internal service discovery and public container exposure.
- **Stateless and stateful applications** – While Kubernetes initially focused on stateless containers, it's now also got built-in objects to represent stateful apps too. You can run any kind of application in Kubernetes

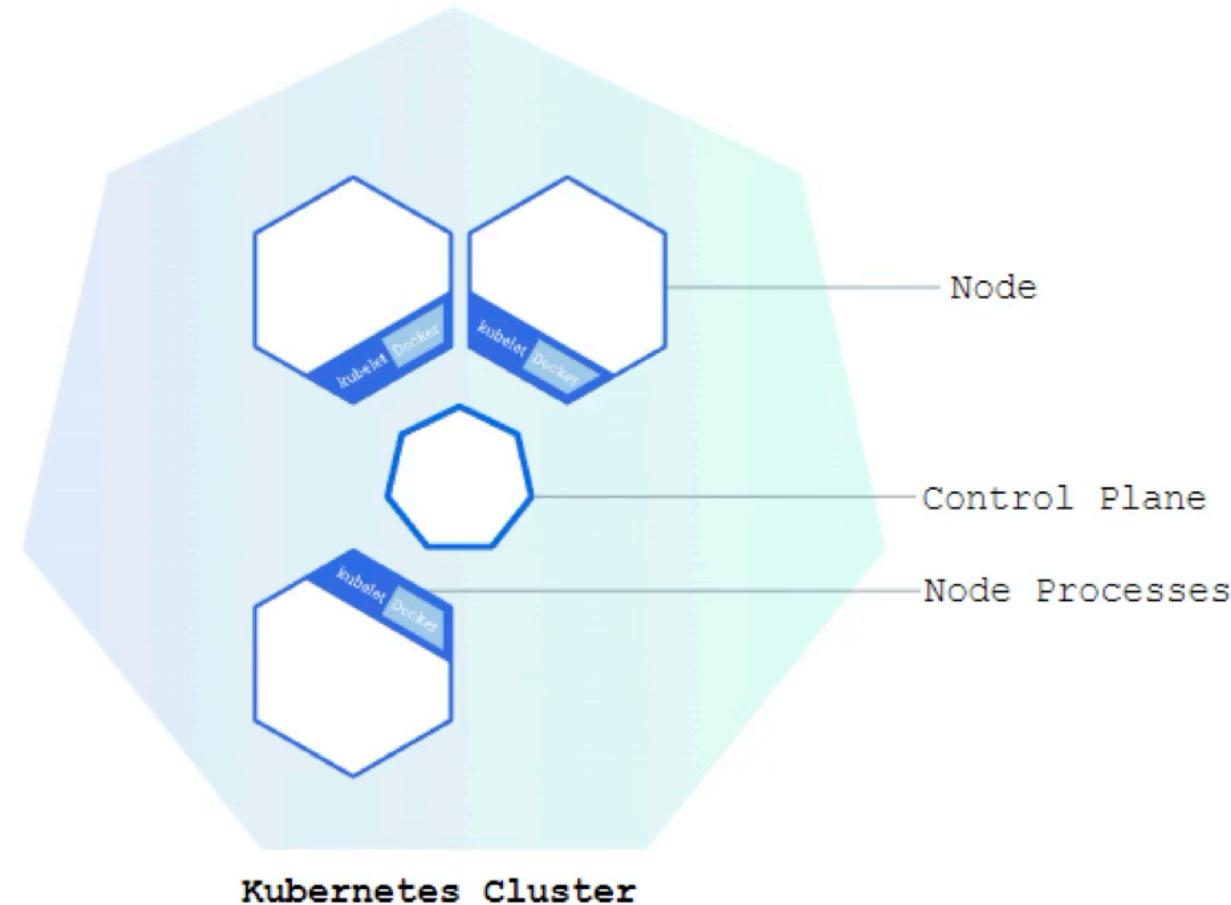
## Kubernetes features #2

- **Storage management** – Persistent storage is abstracted by a consistent interface that works across providers, whether in the cloud, on a network share, or on a local filesystem.
- **Declarative state** – Kubernetes uses object manifests in YAML files to define the state you want to create in your cluster. Applying a manifest instructs Kubernetes to automatically transition the cluster to the target state. You don't have to manually script the changes you want to see.
- **Works across environments** – Kubernetes can be used in the cloud, at the edge, or on your developer workstation. Many different distributions are available to match different use cases. Major cloud providers like AWS and Google Cloud offer managed Kubernetes services, while single-node distributions such as Minikube and K3s are great for local use.
- **Highly extensible** – Kubernetes packs in a lot of functionality, but you can add even more using extensions. You can create custom object types, controllers, and operators to support your own workloads.

Introduction to containerization

# ARCHITECTURE OF KUBERNETES CLUSTER

# Architecture of Kubernetes cluster



Introduction to

# KUBERNETES 101

# Nodes

- Nodes represent the physical machines that form your Kubernetes cluster. They run the containers you create. Kubernetes tracks the status of your nodes and exposes each one as an object. You used Kubectl to retrieve a list of nodes in the example above.
- While your fresh cluster has only one node, Kubernetes advertises support for up to 5,000 nodes. It's theoretically possible to scale even further.

# Pods

- Pods are the fundamental compute unit in Kubernetes. A Pod is analogous to a container but with some key differences. Pods can contain multiple containers, each of which share a context. The entire Pod will always be scheduled onto the same node. The containers within a Pod are tightly coupled so you should create a new Pod for each distinct part of your application, such as its API and database.
- In simple situations, Pods will usually map one-to-one with the containers your application runs. In more advanced cases, Pods can be enhanced with init containers and ephemeral containers to customize startup behavior and provide detailed debugging.

# Deployments

- Deployments wrap ReplicaSets with support for declarative updates and rollbacks. They're a higher level of abstraction that's easier to control.
- A Deployment object lets you specify the desired state of a set of Pods. This includes the number of replicas to run. Modifying the Deployment will automatically detect the required changes and scale the ReplicaSet as required. You can pause the rollout or revert to an earlier revision, features that aren't available with plain ReplicaSets.

# Jobs

- A Kubernetes Job is an object that creates a set of Pods and waits for them to terminate. It will retry any failed Pods until a specified number have exited successfully. The Job's then marked as complete.
- Jobs provide a mechanism for running ad-hoc tasks inside your cluster. Kubernetes also provides CronJobs that wrap Jobs with cron-like scheduling support. These let you automatically run a job on a regular cadence to accommodate batch activities, backups, and any other scheduled tasks your application requires.

# Services

- Services are used to expose Pods to the network. They allow defined access to Pods either within your cluster or externally.
- Ingresses are closely related objects. These are used to set up HTTP routes to services via a load balancer. Ingresses also support HTTPS traffic secured by TLS certificates.

**Read more** Kubernetes Ingress with NGINX Ingress Controller Example

# Labels

- Labels are key/value pairs that are attached to objects, such as pods.
- Labels are intended to be used to specify identifying attributes of objects that are meaningful and relevant to users, but do not directly imply semantics to the core system.
- Labels can be used to organize and to select subsets of objects.
- Labels can be attached to objects at creation time and subsequently added and modified at any time.
- Each object can have a set of key/value labels defined.
- Each Key must be unique for a given object.

# Selectors

- Label selector, the client/user can identify a set of objects.  
The label selector is the core grouping primitive in Kubernetes.

## example

```
kubectl get pods -l environment=production,tier=frontend
```

# Namespaces

- Namespaces isolate different groups of resources. They avoid name collisions by scoping the visibility of your resources.
- Creating two objects with the same name is forbidden within the same namespace. If you're in the default namespace, you can't create two Pods that are both called database, for example. Namespaces resolve this by providing logical separation of resources. Two namespaces called app-1 and app-2 could each contain a Pod called database without causing a conflict.
- Namespaces are flexible and can be used in many different ways. It's a good idea to create a namespace for each workload in your cluster. You can also use namespaces to divide resources between users and teams by applying role-based access control.

# Kubelet

- Kubelet is a worker process that runs on each of your nodes. It maintains communication with the Kubernetes control plane to receive its instructions.
- Kubelet is responsible for pulling container images and starting containers in response to scheduling requests.

# Kube-proxy

- Proxy is another component found on individual nodes. It configures the host's networking system so traffic can reach the services in your cluster.

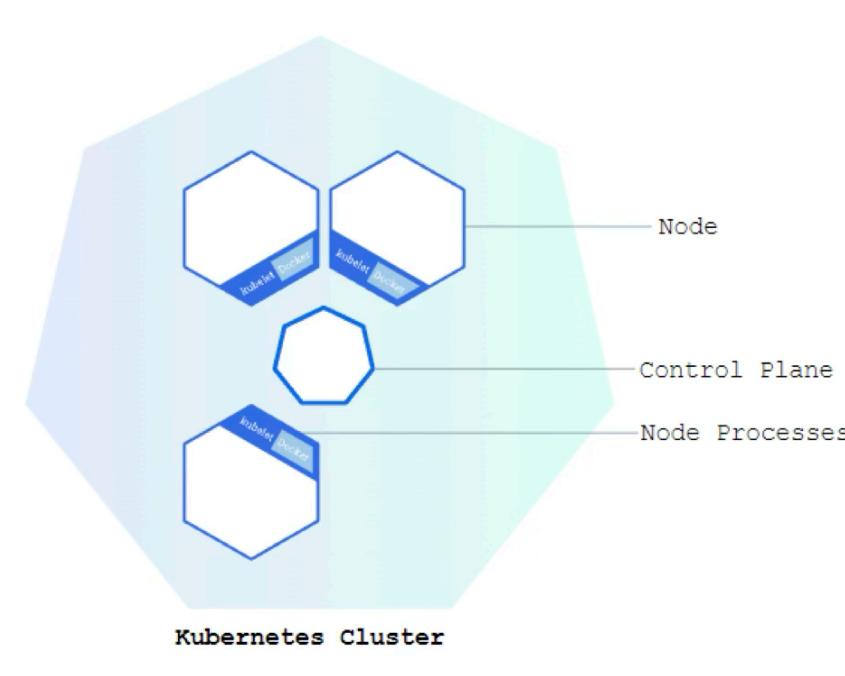
How to run

# THE HELLO WORLD APPLICATION

# Using Minikube to Create a Cluster

A Kubernetes cluster consists of two types of resources:

- The Control Plane coordinates the cluster
- Nodes are the workers that run applications



# Create a minikube cluster

- Start minikube

```
minikube start
```

```
C:\Windows\System32>minikube start
* minikube v1.28.0 on Microsoft Windows 11 Home Single Language 10.0.22621 Build 22621
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Updating the running docker "minikube" container ...
* Preparing Kubernetes v1.25.3 on Docker 20.10.20 ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

# Deployment step

- Create deployment

```
kubectl create deployment hello-node --image=nginx  
kubectl get deployment
```

- Create loadbalancer server

```
minikube tunnel
```

- Create service

```
kubectl expose deployment hello-node --type=LoadBalancer --port=8080 --target-port=80  
kubectl get svc
```

- Tunnel

```
kubectl port-forward service/hello-node 8080:8080 --address=0.0.0.0
```

# Scaling the Hello World application

```
kubectl scale --replicas=2 deployment hello-node  
kubectl get deployment  
kubectl get pod
```

advance topic in

# KUBERNETES

# kubectl command list

```
kubectl api-resources
kubectl apply -f ./test.yaml
kubectl create deployment nginx --image=nginx
kubectl expose deployment nginx --type=LoadBalancer --port=80
kubectl get services
kubectl get pods
kubectl get deployments
kubectl get nodes
kubectl set image deployment/frontend www=image:v2
kubectl expose rc nginx --port=80 --target-port=8000
kubectl autoscale deployment foo --min=2 --max=10
kubectl get hpa
kubectl scale --replicas=3 -f ./test.yaml
kubectl delete pod,service baz foo
```

# Deployment

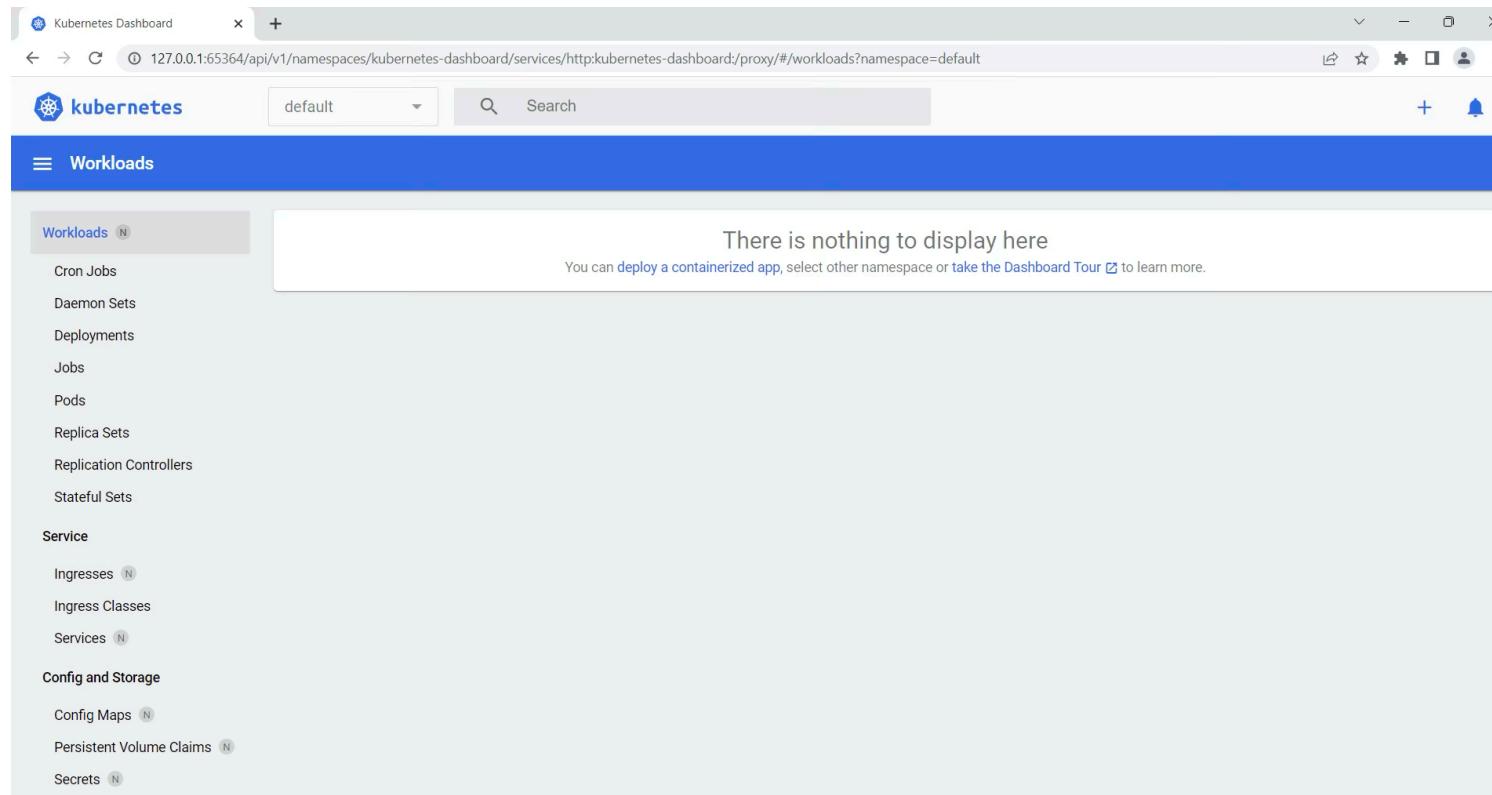
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
  spec:
    containers:
      - name: hello-world-server
        image: nginx:alpine
        ports:
          - containerPort: 80
```

# Service

```
apiVersion: v1
kind: Service
metadata:
  name: helloworld
  labels:
    app: hello-world
spec:
  selector:
    app: hello-world
  ports:
  - name: http
    protocol: TCP
    port: 8080
    targetPort: 80
  type: ClusterIP
```

# Kubernetes dashboard

```
minikube addons enable metrics-server
minikube dashboard
```



The screenshot shows a web browser window titled "Kubernetes Dashboard" with the URL "127.0.0.1:65364/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/workloads?namespace=default". The dashboard has a blue header bar with the "kubernetes" logo and a search bar. Below the header is a navigation menu with "Workloads" selected. The main content area displays a message: "There is nothing to display here. You can deploy a containerized app, select other namespace or take the Dashboard Tour [to learn more.](#)". On the left side, there is a sidebar with categories: "Workloads" (selected), "Cron Jobs", "Daemon Sets", "Deployments", "Jobs", "Pods", "Replica Sets", "Replication Controllers", "Stateful Sets", "Service" (with sub-options: "Ingresses", "Ingress Classes", "Services"), and "Config and Storage" (with sub-options: "Config Maps", "Persistent Volume Claims", "Secrets").

# ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-config
data:
  root-password: "1234"
  database-name: "simple_db"
  user: "usr"
  password: "usr"
```

# Deployment with ConfigMap

env:

- name: MYSQL\_ROOT\_PASSWORD  
valueFrom:  
  configMapKeyRef:  
    name: mysql-config  
    key: root-password

# Application Secret

- Create new secret from file

```
kubectl create secret generic regcred  
--from-file=.dockerconfigjson=<path/to/.docker/config.json>  
--type=kubernetes.io/dockerconfigjson
```

- Create new secret from parameter

```
kubectl create secret docker-registry regcred  
--docker-server=<your-registry-server>  
--docker-username=<your-name>  
--docker-password=<your-pword>  
--docker-email=<your-email>
```

- Get all secret

```
kubectl get secret
```

# Dealing with application secret

```
spec:  
  containers:  
    - name: hello-world-server  
      image: nginx:alpine  
      ports:  
        - containerPort: 80  
      resources:  
        requests:  
          memory: "10Mi"  
          cpu: "50m"  
        limits:  
          memory: "128Mi"  
          cpu: "500m"  
      imagePullSecrets:  
        - name: regcred
```

հմայւեգ  $\text{cpu } 1000\text{m} = 1 \text{ core}$

# Workshop #1

## การสร้าง Jenkins user

# login เข้า jenkins server



## Sign in to Jenkins

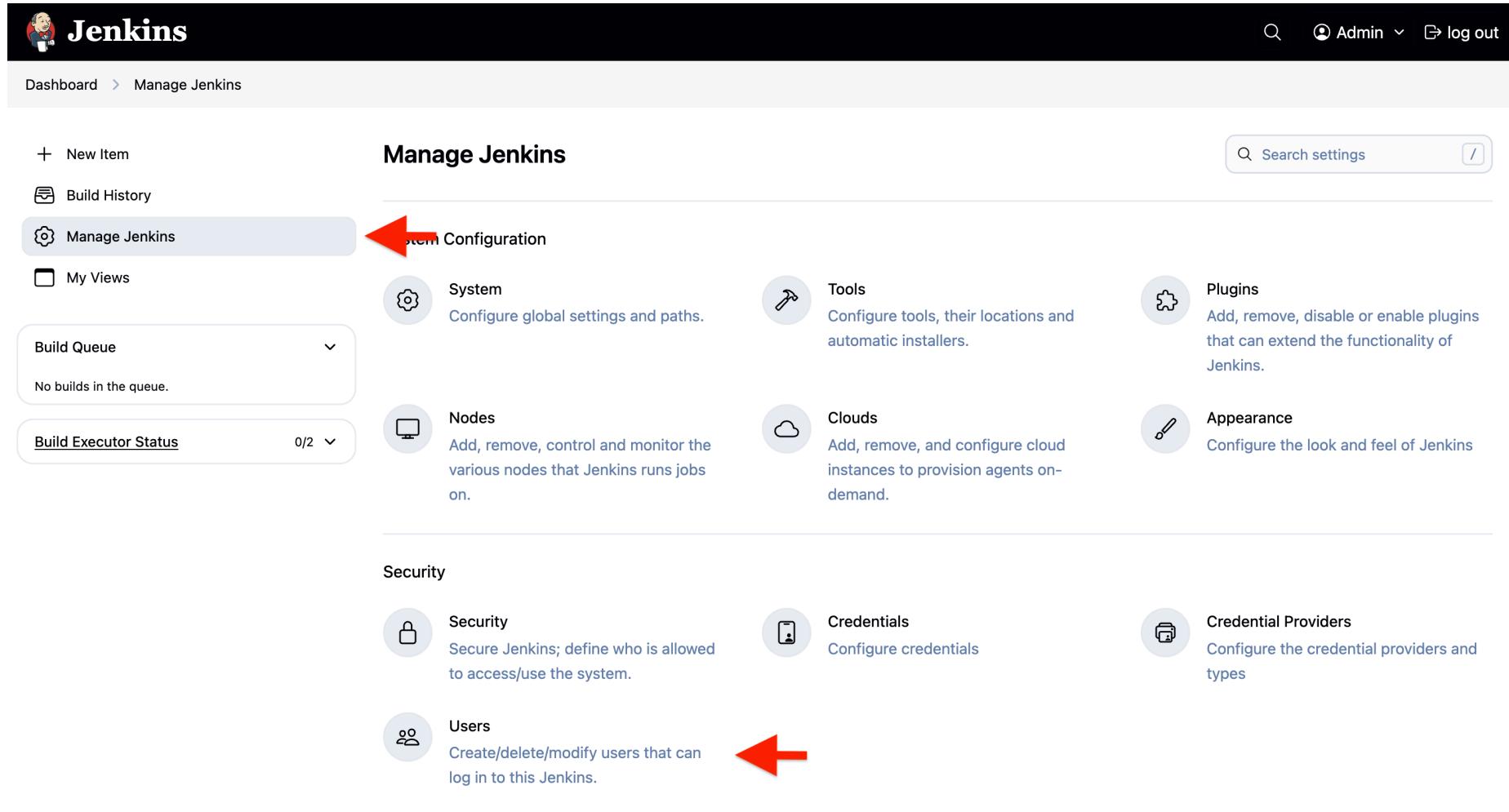
Username

Password

Keep me signed in

Sign in

# สร้าง user ใน Jenkins #1



The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'Build History', 'Manage Jenkins' (which is highlighted with a red arrow), and 'My Views'. Below that are two collapsed sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (0/2). The main content area is titled 'Manage Jenkins' and contains several configuration sections:

- Item Configuration**: A placeholder section.
- System**: Configure global settings and paths.
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Tools**: Configure tools, their locations and automatic installers.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Appearance**: Configure the look and feel of Jenkins.

---

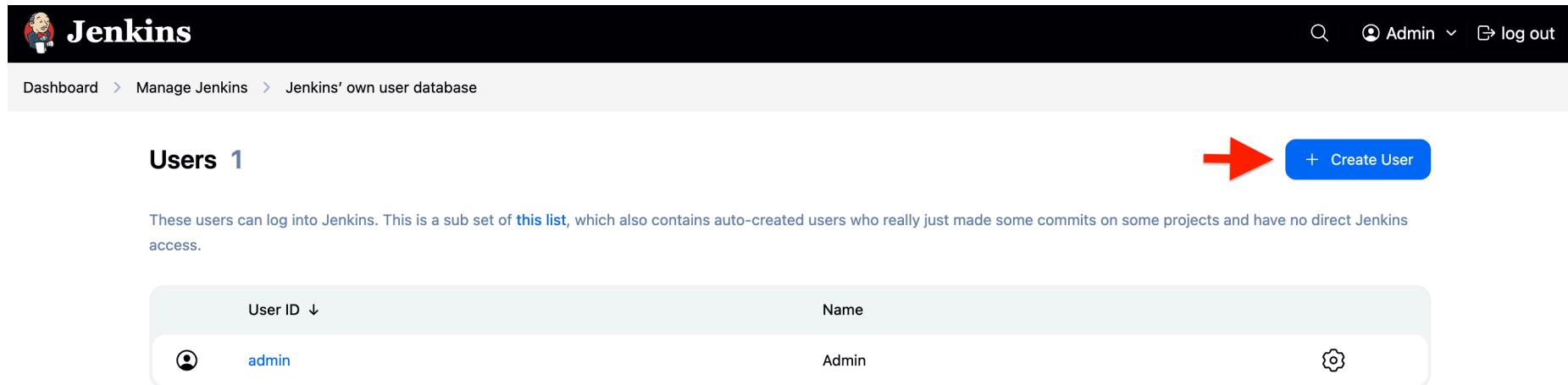
**Security**

- Security**: Secure Jenkins; define who is allowed to access/use the system.
- Credentials**: Configure credentials.
- Credential Providers**: Configure the credential providers and types.

---

**Users**: Create/delete/modify users that can log in to this Jenkins. (This section is highlighted with a red arrow).

# สร้าง user ใหม่ #2



The screenshot shows the Jenkins user management interface. At the top, there's a navigation bar with the Jenkins logo, a search icon, 'Admin' dropdown, and 'log out'. Below it, a breadcrumb trail shows 'Dashboard > Manage Jenkins > Jenkins' own user database'. The main area is titled 'Users 1'. It lists a single user: 'User ID' (admin), 'Name' (Admin), and a gear icon for settings. To the right of the list is a blue button with a white plus sign and the text '+ Create User', which is highlighted by a large red arrow.

User ID ↓	Name	
admin	Admin	

# สร้าง user ใหม่ #3

## Create User

Username

Password

Confirm password

Full name

E-mail address



Create User

# สร้าง user ใหม่ #4

- logout ออกจากระบบ
- ทดสอบ login ใหม่ด้วย user ที่เพิ่งสร้างขึ้นมา

# Workshop #2

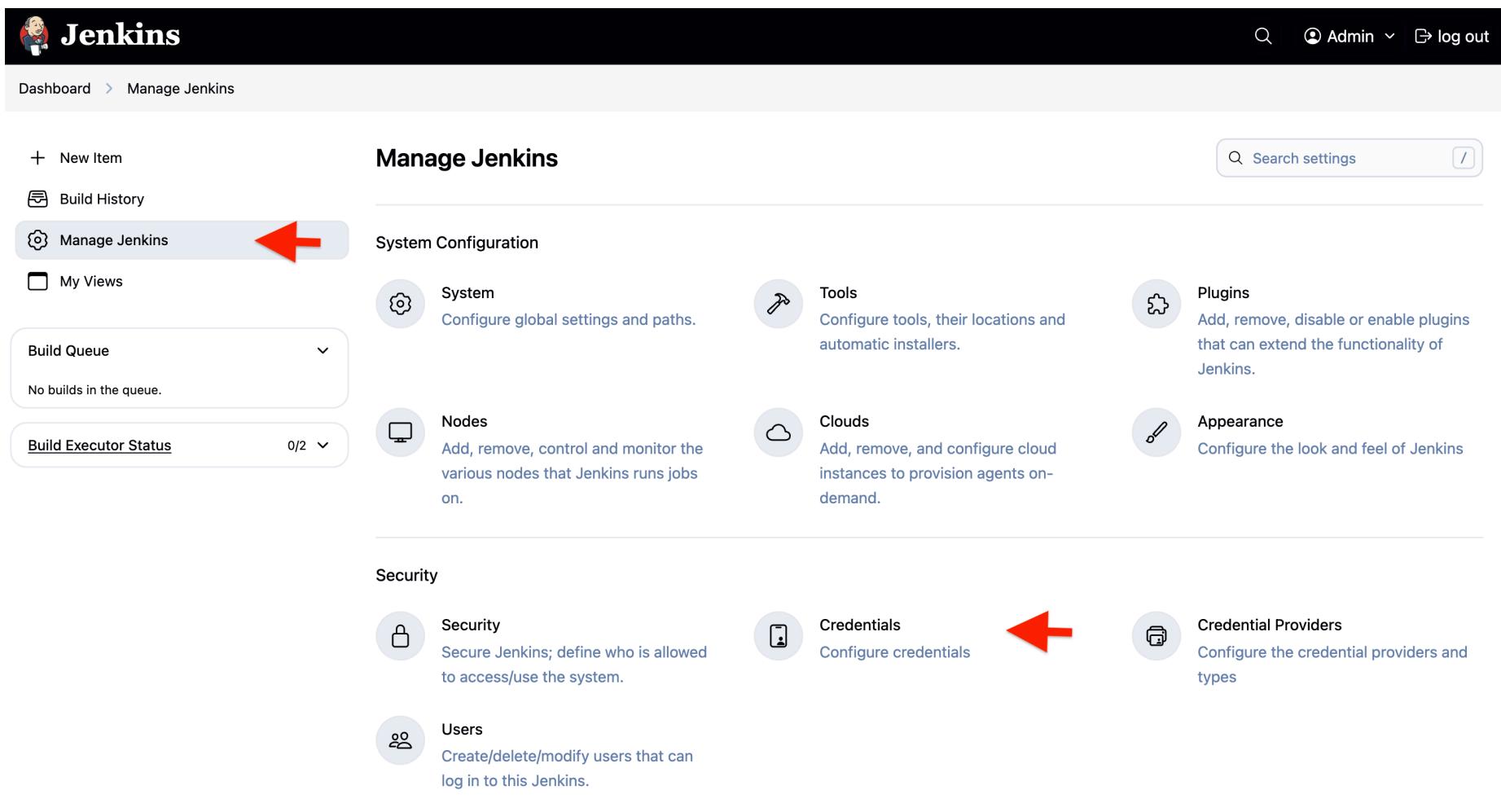
## การสร้าง Jenkins credential

# About Jenkins credential

Credential เอาไว้เก็บข้อมูลที่จำเป็นต้องใช้ใน script เพื่อเชื่อมต่อกับระบบอื่น ๆ โดยแบ่งเป็น ประเภทดังนี้

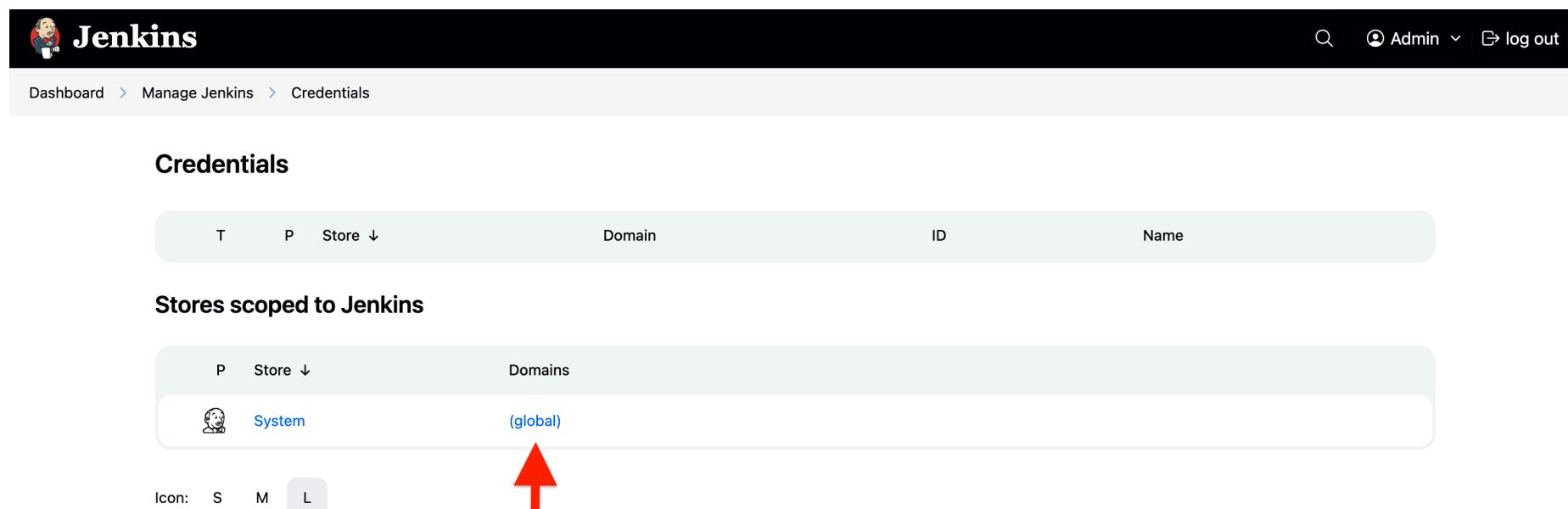
- Username with password
- GitHub App
- SSH Username with private key
- Secret file
- Secret text
- Certificate

# សរោង Credential ឱ្យកៅន Username with password



The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'Build History', 'Manage Jenkins' (which is highlighted with a red arrow), 'My Views', 'Build Queue' (showing 'No builds in the queue.'), and 'Build Executor Status' (showing '0/2'). The main area is titled 'Manage Jenkins' and has two main sections: 'System Configuration' and 'Security'. In 'System Configuration', there are links for 'System', 'Tools', 'Plugins', 'Nodes', 'Clouds', and 'Appearance'. In 'Security', there are links for 'Security', 'Credentials' (which is highlighted with a red arrow), 'Users', and 'Credential Providers'. A search bar at the top right says 'Search settings'.

# เลือก global

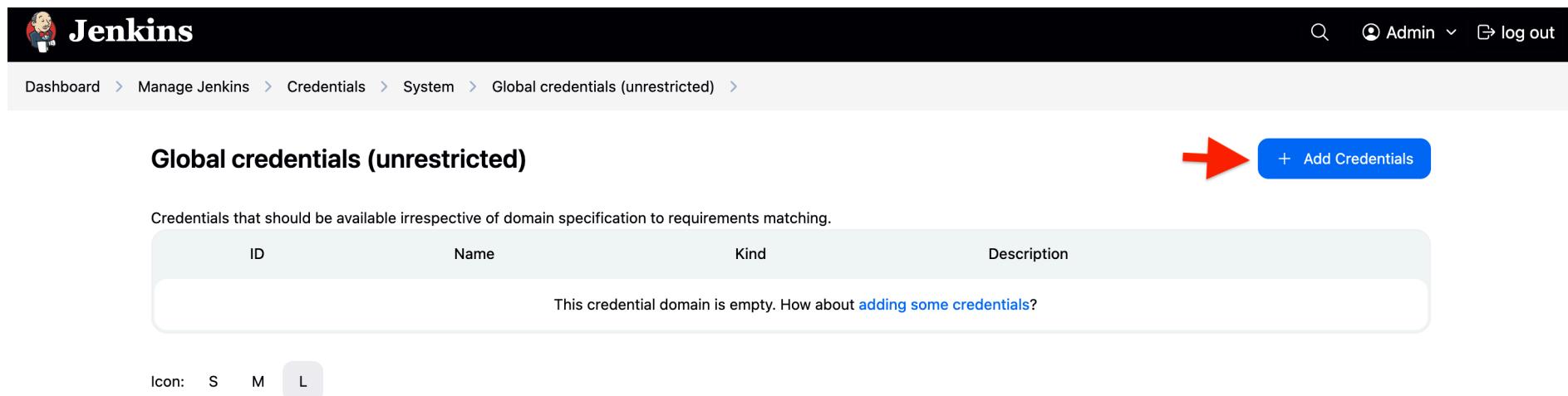


The screenshot shows the Jenkins 'Credentials' management interface. At the top, there is a navigation bar with the Jenkins logo, a search icon, 'Admin' user information, and a 'log out' link. Below the navigation bar, the breadcrumb navigation shows 'Dashboard > Manage Jenkins > Credentials'. The main title is 'Credentials'. A table header row includes columns for 'Domain', 'ID', and 'Name'. Below this, a section titled 'Stores scoped to Jenkins' contains a table with columns for 'Domain' and 'Name'. One entry is visible: 'System' under 'Domain' and '(global)' under 'Name'. A red arrow points upwards from the bottom of this table towards the 'Domains' column header. At the bottom of the table, there are icons for 'Icon:', 'S', 'M', and 'L'.

Domain	ID	Name
System	(global)	

Icon: S M L

# ເລືອກ add credential



The screenshot shows the Jenkins Global credentials (unrestricted) page. The top navigation bar includes the Jenkins logo, a search icon, 'Admin' dropdown, and 'log out'. The breadcrumb path is: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted). The main title is 'Global credentials (unrestricted)'. A red arrow points to the blue 'Add Credentials' button. Below the table, a message says 'This credential domain is empty. How about [adding some credentials?](#)'.

ID	Name	Kind	Description
This credential domain is empty. How about <a href="#">adding some credentials?</a>			

Icon: S M L

# ក្រោកខាងមុខໃន form

## New credentials

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

 Treat username as secret ?

Password ?

ID ?

Description ?

Create

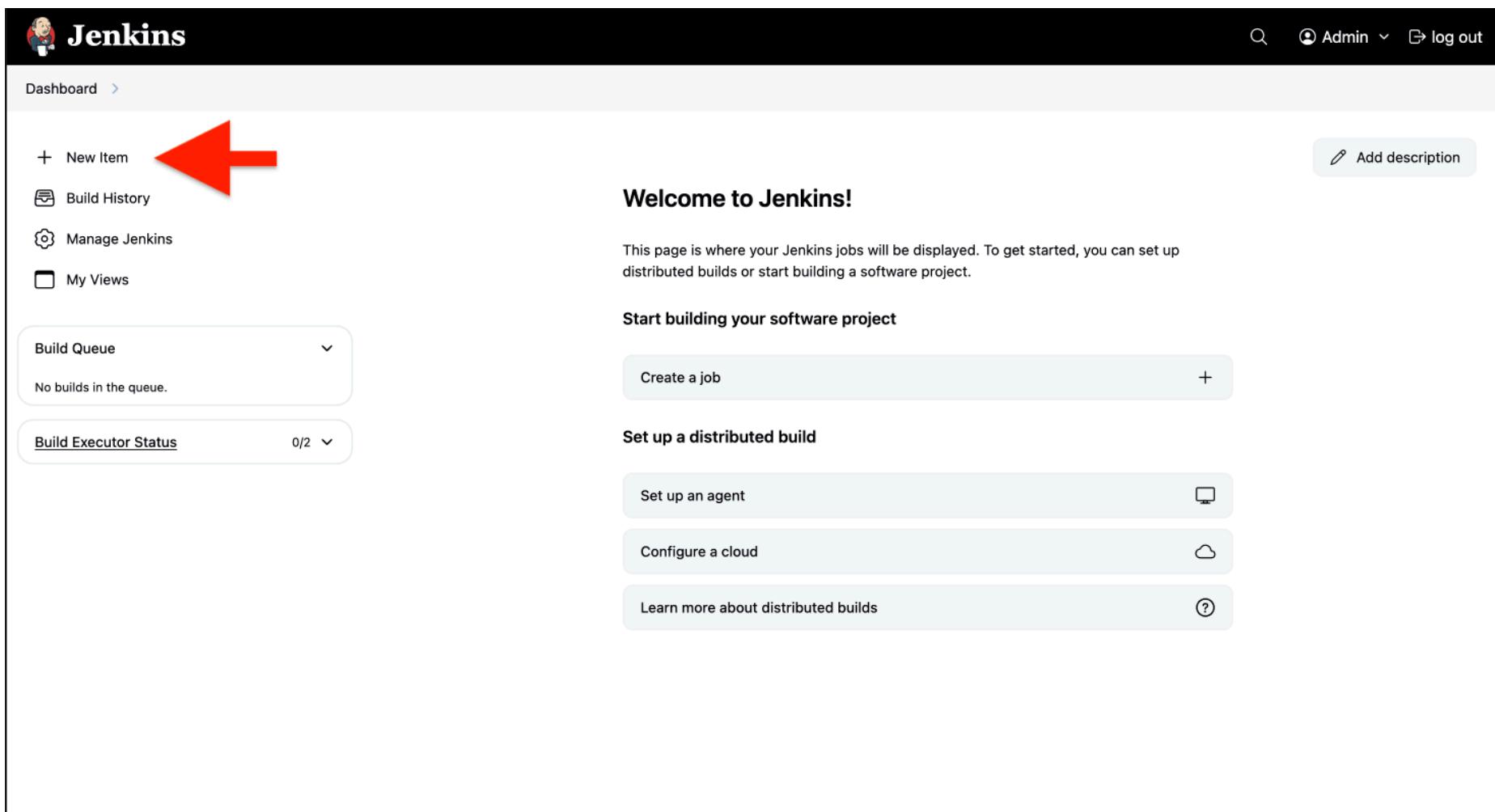
# กรอกข้อมูล credential สำหรับ gitlab

- Kind = Username with password
- Scope = Global
- Username = email ที่ใช้ login เข้า gitlab
- Password = รหัสผ่านสำหรับ gitlab
- ID = ชื่อตัวแปรสำหรับการอ้างถึงใน script Jenkinsfile
- Description = คำอธิบายรายการ

# Workshop #3

การสร้าง Jenkins Job / Item

# เลือก New Item



The screenshot shows the Jenkins dashboard. On the left, there is a sidebar with the following items:

- + New Item (highlighted with a large red arrow)
- Build History
- Manage Jenkins
- My Views

Below the sidebar, there are two sections:

- Start building your software project**: Contains a "Create a job" button with a "+" icon.
- Set up a distributed build**: Contains three buttons: "Set up an agent" (with a monitor icon), "Configure a cloud" (with a cloud icon), and "Learn more about distributed builds" (with a question mark icon).

At the top right of the dashboard, there is a search bar, a user dropdown menu showing "Admin", and a "log out" link.

# ក្រោកខាងមុខ Item

## New Item

Enter an item name



Select an item type



### Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



### Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



### Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



### Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



### Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK



# ក្រោកខាងមុន Item

## Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

### Definition

Pipeline script

#### Script ?

1

try sample Pipeline... ▼

Use Groovy Sandbox ?

#### Pipeline Syntax

Save

Apply

# ຮະບຸຂ້ອມູລດັ່ງນີ້

- Definition = Pipeline script
- Script

```
node: {  
    stage("stage 1") {  
        echo "Hello"  
    }  
    stage("stage 2") {  
        echo "World!"  
    }  
}
```

# ສົ່ງໃຫ້ Job ກໍາງານແບບ manual

Status      admin-job      Add description

</> Changes

▷ Build Now 

Configure

Delete Pipeline

Full Stage View

Stages

Rename

Pipeline Syntax

**Stage View**

No data available. This Pipeline has not yet run.

**Permalinks**

Builds      ...

No builds

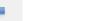
# អាជីវកម្មនៃរូបរាង stage view

Status      admin-job      Add description

</> Changes      ▷ Build Now      ⚙ Configure      🗑 Delete Pipeline      🔎 Full Stage View      📂 Stages      🖊 Rename      ⓘ Pipeline Syntax

### Stage View

Average stage times:  
(full run time: ~959ms)

stage 1	stage 2
203ms	78ms
	
#1 May 14 10:55	No Changes
203ms	78ms

### Permalinks

Builds      ⋮      

No builds

Today

✓ #1 3:55 AM

# Workshop #4

## Jenkins CI/CD with Gitlab

# สร้าง Jenkinsfile

```
pipeline {  
    agent any  
    environment {  
        APP_NAME = "test app name"  
    }  
    stages {  
        stage("Stage 1") {  
            steps {  
                sh "echo ${APP_NAME}"  
            }  
        }  
        stage("Stage 2") {  
            steps {  
                sh "echo Hello"  
            }  
        }  
    }  
}
```

# push Jenkinsfile ទៅ gitlab

```
git add Jenkinsfile
git commit -m "add Jenkinsfile"
git push
```

# สร้าง Item / Job ใหม่ใน Jenkins

- Name = <user-name>-ci-job
- Type = Pipeline
- Definition = Pipeline script from SCM
  - SCM = Git
  - Repositories
    - Repository URL = <gitlab repository url>
    - Credentials = <Credential id ที่สร้างจากขั้นตอนก่อนหน้านี้>
  - Branch Specifier = \*/main
  - Script Path = Jenkinsfile

# ຕັວອຢ່າງ

Definition  
Pipeline script from SCM

SCM ?  
Git

Repositories ?

Repository URL ?  
`https://gitlab.com/sommai.k/devops-easset.git`

Credentials ?  
`sommai.k@gmail.com/*****`

+ Add

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?  
`*/mair`

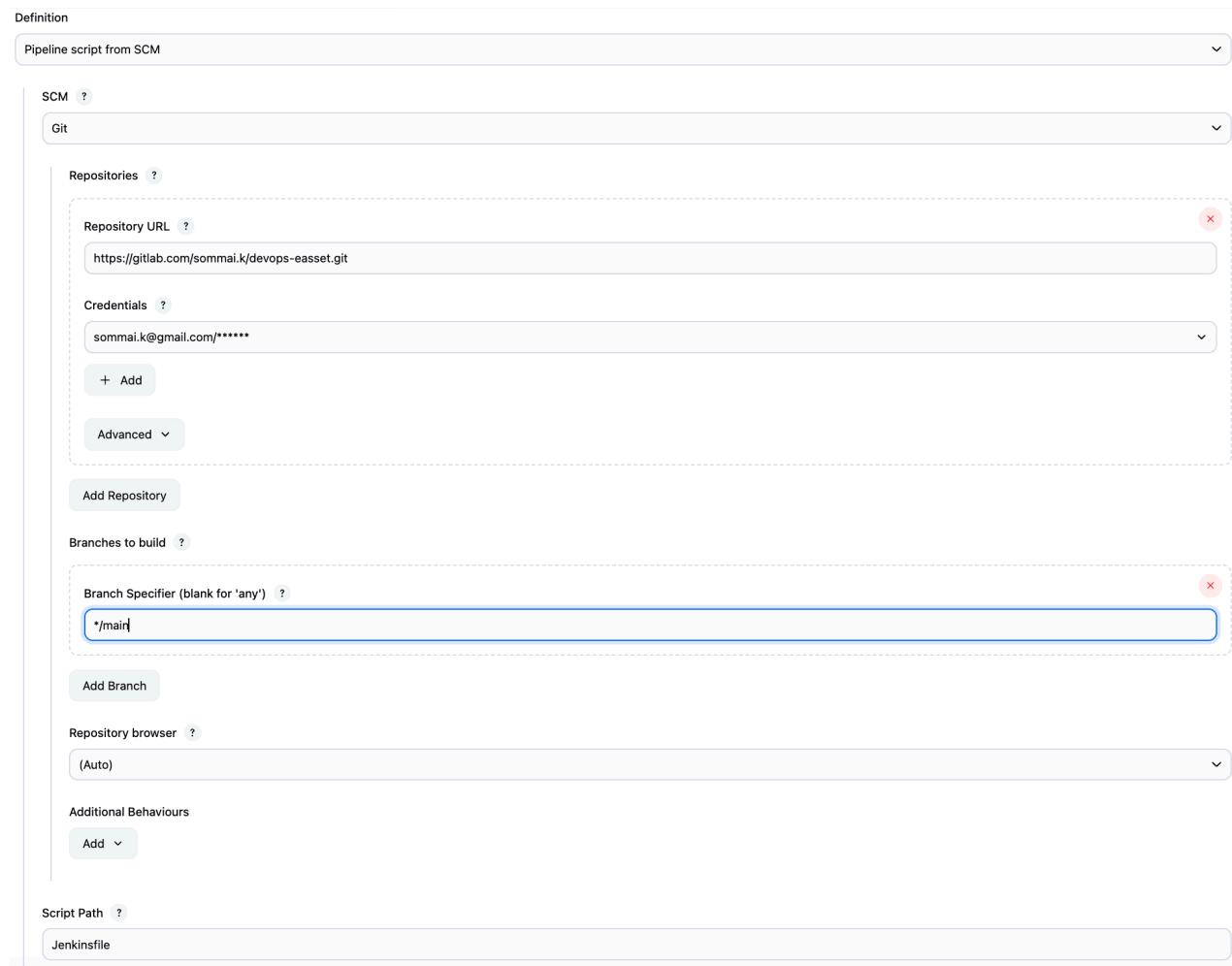
Add Branch

Repository browser ?  
(Auto)

Additional Behaviours

Add ▾

Script Path ?  
Jenkinsfile



# ສົ່ງໃຫ້ Job ກໍາງານແບບ manual

Status      admin-job      Add description

</> Changes

▷ Build Now 

Configure

Delete Pipeline

Full Stage View

Stages

Rename

Pipeline Syntax

**Stage View**

No data available. This Pipeline has not yet run.

**Permalinks**

Builds      ...

No builds

# ผลลัพธ์จากการทำงาน

Dashboard > admin-ci-job >

Status

</> Changes

▷ Build Now

⚙ Configure

trash Delete Pipeline

🔍 Full Stage View

☰ Stages

-pencil Rename

ⓘ Pipeline Syntax

✖ **admin-ci-job**

## Stage View

Average stage times:  
(full run time: ~14s)

#3  
May 14  
13:07  
No Changes

**Declarative:**  
**Checkout SCM**

6s

Stage 1

503ms

Stage 2

366ms

6s

503ms

366ms

# Workshop #5

สร้าง custom image

# สร้าง file ชื่อ Dockerfile ที่ root ของ Project

```
FROM nginx:alpine
COPY ./html /usr/share/nginx/html
```

# Build custom image ใน local

- run คำสั่งเพื่อ build docker image

```
docker build -t <registry-server>/hello .
```

- ทดสอบผลการ build ด้วยคำสั่ง

```
docker image ls <registry-server>/hello
```

- ทดสอบนำ image ขึ้น registry ด้วยคำสั่งดังนี้

```
docker push <registry-server>/hello
```

# Push Dockerfile ขึ้น gitlab

- push code ขึ้น gitlab ด้วยคำสั่งดังนี้

```
git add Dockerfile
git commit -m "Add Dockerfile"
git push
```

# Workshop #6

## Jenkins CI/CD with docker

# แก้ไข file Jenkinsfile

```
pipeline {  
    agent { label 'build-server' }  
    environment {  
        APP_NAME = "test app name"  
        IMAGE_NAME = "<registry-server>/hello"  
    }  
    stages {  
        stage("Build Image") {  
            steps {  
                sh "echo ${APP_NAME}"  
                sh "docker version"  
                sh "docker build -t ${IMAGE_NAME} ."  
            }  
        }  
    }  
}
```

# push Jenkinsfile ទៅ gitlab

```
git add Jenkinsfile
git commit -m "add Jenkinsfile"
git push
```

# ສົ່ງໃຫ້ Job ກໍາງານແບບ manual

Status      admin-job      Add description

</> Changes

▷ Build Now 

Configure

Delete Pipeline

Full Stage View

Stages

Rename

Pipeline Syntax

**Stage View**

No data available. This Pipeline has not yet run.

**Permalinks**

Builds      ...

No builds

# ผลลัพธ์จากการทำงาน

Dashboard > admin-ci-job >

 Status

</> Changes

 Build Now

 Configure

 Delete Pipeline

 Full Stage View

 Stages

 Rename

 Pipeline Syntax

 admin-ci-job

## Stage View

Average stage times:  
(full run time: ~13s)

Declarative:	Checkout SCM	Build Image
6s	2s	
4s	5s	

#6 May 14 13:51 1 commit

# Workshop #7

## Jenkins CI/CD with docker registry

# แก้ไข file Jenkinsfile

- เพิ่ม stage เข้าไปที่ file Jenkinsfile ภายใต้ stages

```
stage("Delivery") {  
    steps {  
        withCredentials(  
            [usernamePassword(  
                credentialsId: '<credential-id จากขั้นตอนการสร้าง credential id>',  
                passwordVariable: 'gitlabPassword',  
                usernameVariable: 'gitlabUser'  
            )])  
        {  
            sh "docker login registry.gitlab.com -u ${gitlabUser} -p ${gitlabPassword}"  
            sh "docker push ${IMAGE_NAME}"  
        }  
    }  
}
```

# push Jenkinsfile ទៅ gitlab

```
git add Jenkinsfile
git commit -m "add Jenkinsfile"
git push
```

# ສົ່ງໃຫ້ Job ກໍາງານແບບ manual

Status      admin-job      Add description

</> Changes

▷ Build Now 

Configure

Delete Pipeline

Full Stage View

Stages

Rename

Pipeline Syntax

**Stage View**

No data available. This Pipeline has not yet run.

**Permalinks**

Builds      ...

No builds

# ผลลัพธ์จากการทำงาน

 Status

</> Changes

▷ Build Now

 Configure

 Delete Pipeline

 Full Stage View

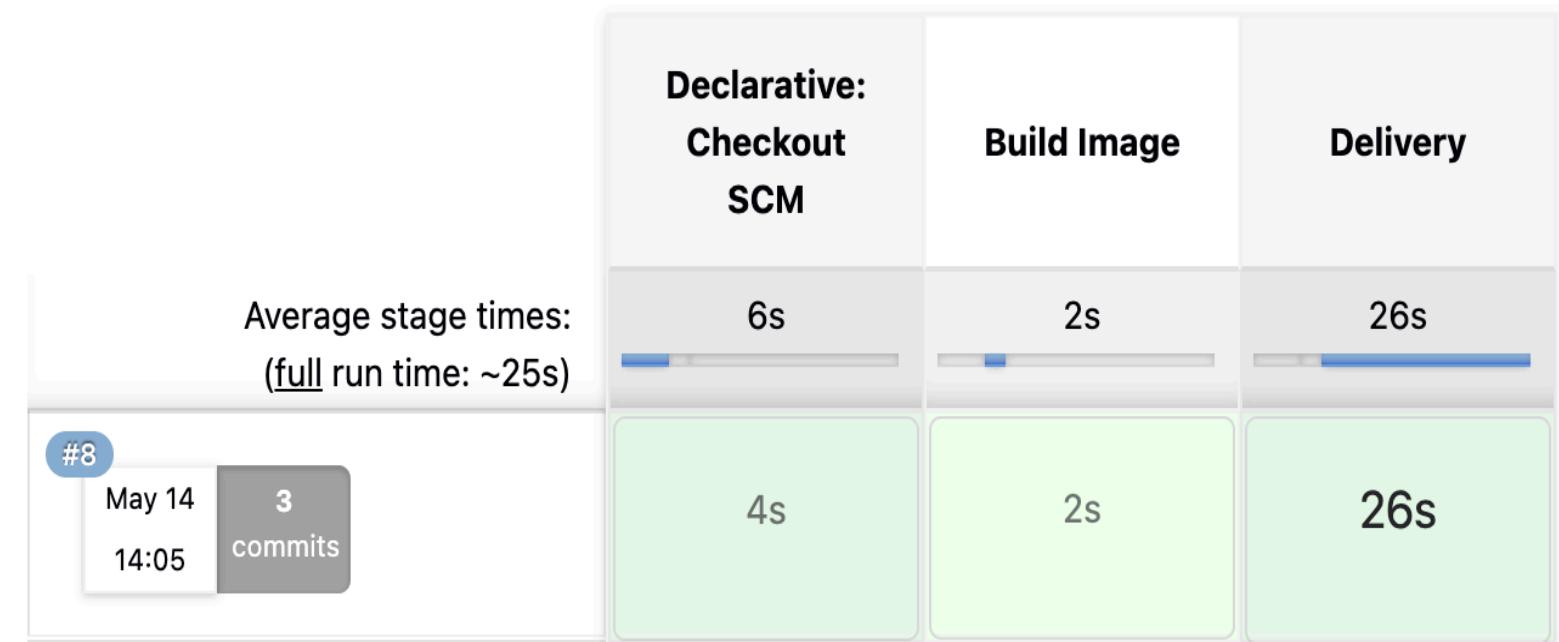
 Stages

 Rename

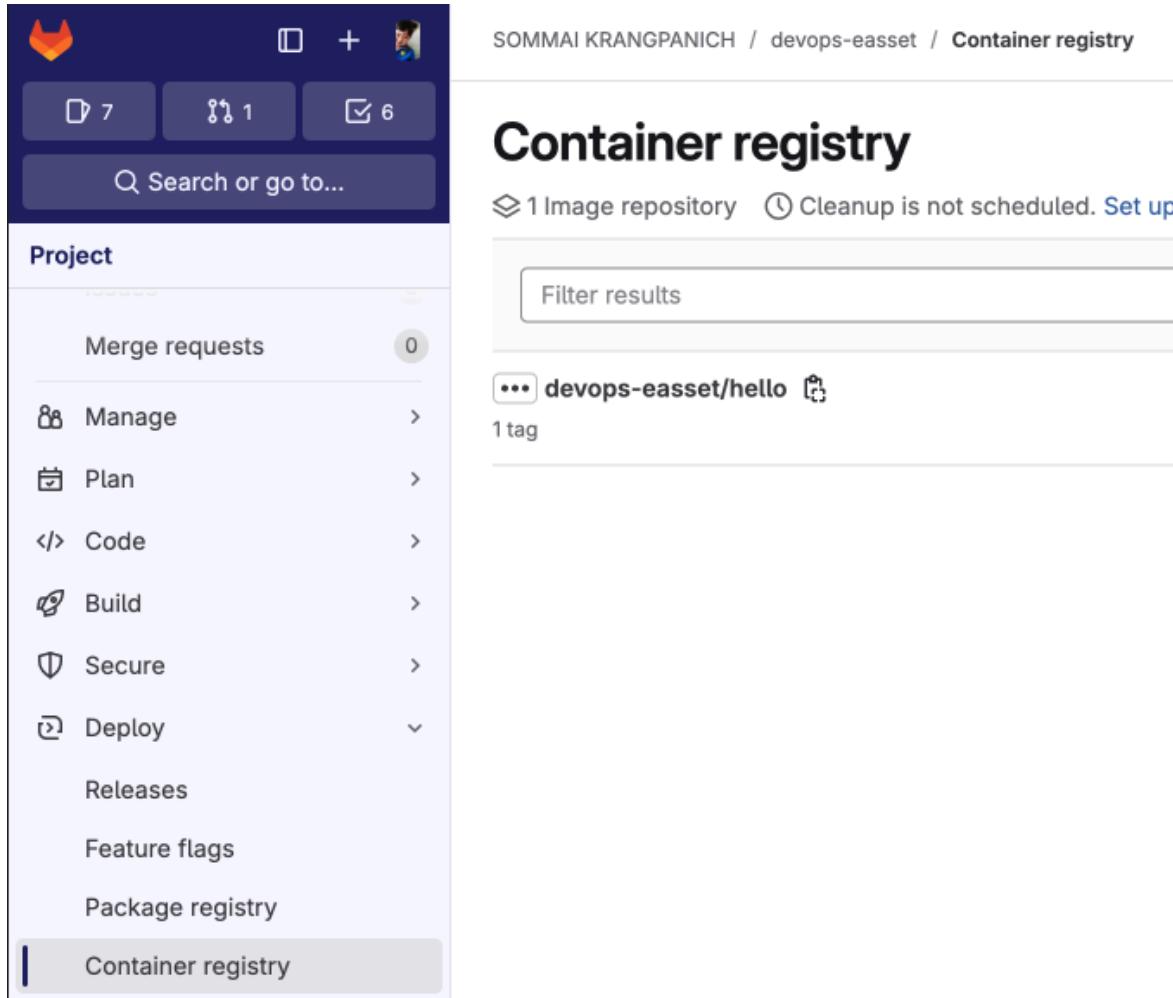
 Pipeline Syntax

 admin-ci-job

## Stage View



# ຕຽວຈັບສອບ docker image ໃນ docker-registry



The screenshot shows a user interface for managing a CI/CD pipeline. On the left, there is a sidebar with various project management and development tools listed:

- Merge requests (0)
- Manage
- Plan
- Code
- Build
- Secure
- Deploy
- Releases
- Feature flags
- Package registry
- Container registry (highlighted with a blue bar at the bottom)

The main area is titled "Container registry" and shows the following information:

SOMMAI KRANGPANICH / devops-easset / Container registry

## Container registry

1 Image repository    Cleanup is not scheduled. Set up cleanup

Filter results

... devops-easset/hello

1 tag

# Workshop #8

## Revision and Cleanup

# แก้ไข file Jenkinsfile

- แก้ไข steps ภายใต้ stage Delivery ดังนี้

```
{  
    sh "docker login registry.gitlab.com -u ${gitlabUser} -p ${gitlabPassword}"  
    sh "docker tag ${IMAGE_NAME} ${IMAGE_NAME}:1.0.${BUILD_NUMBER}"  
    sh "docker push ${IMAGE_NAME}"  
    sh "docker push ${IMAGE_NAME}:1.0.${BUILD_NUMBER}"  
    sh "docker rmi ${IMAGE_NAME}:1.0.${BUILD_NUMBER}"  
    sh "docker rmi ${IMAGE_NAME}"  
}
```

# push Jenkinsfile ទៅ gitlab

```
git add Jenkinsfile
git commit -m "add Jenkinsfile"
git push
```

# ສົ່ງໃຫ້ Job ກໍາງານແບບ manual

Status      admin-job      Add description

</> Changes

▷ Build Now 

Configure

Delete Pipeline

Full Stage View

Stages

Rename

Pipeline Syntax

**Stage View**

No data available. This Pipeline has not yet run.

**Permalinks**

Builds      ...

No builds

# ผลลัพธ์จากการทำงาน

 Status

</> Changes

▷ Build Now

 Configure

 Delete Pipeline

 Full Stage View

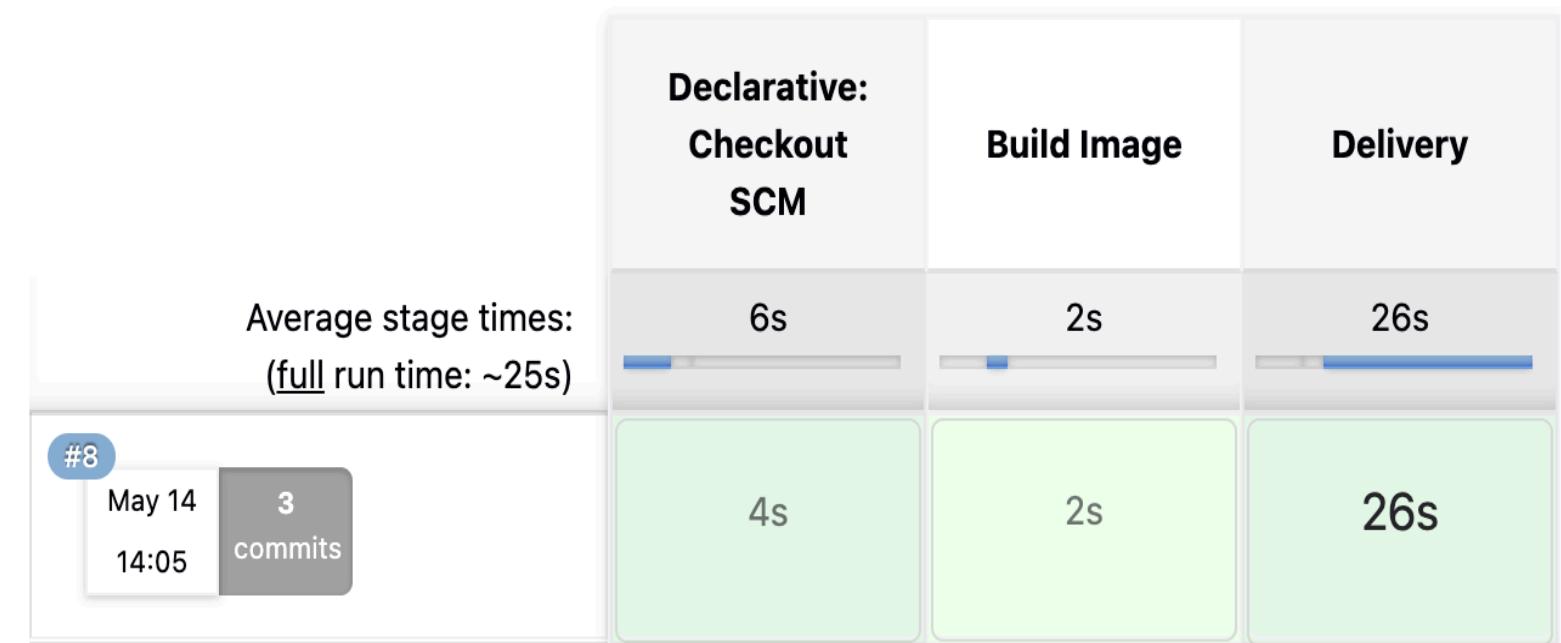
 Stages

 Rename

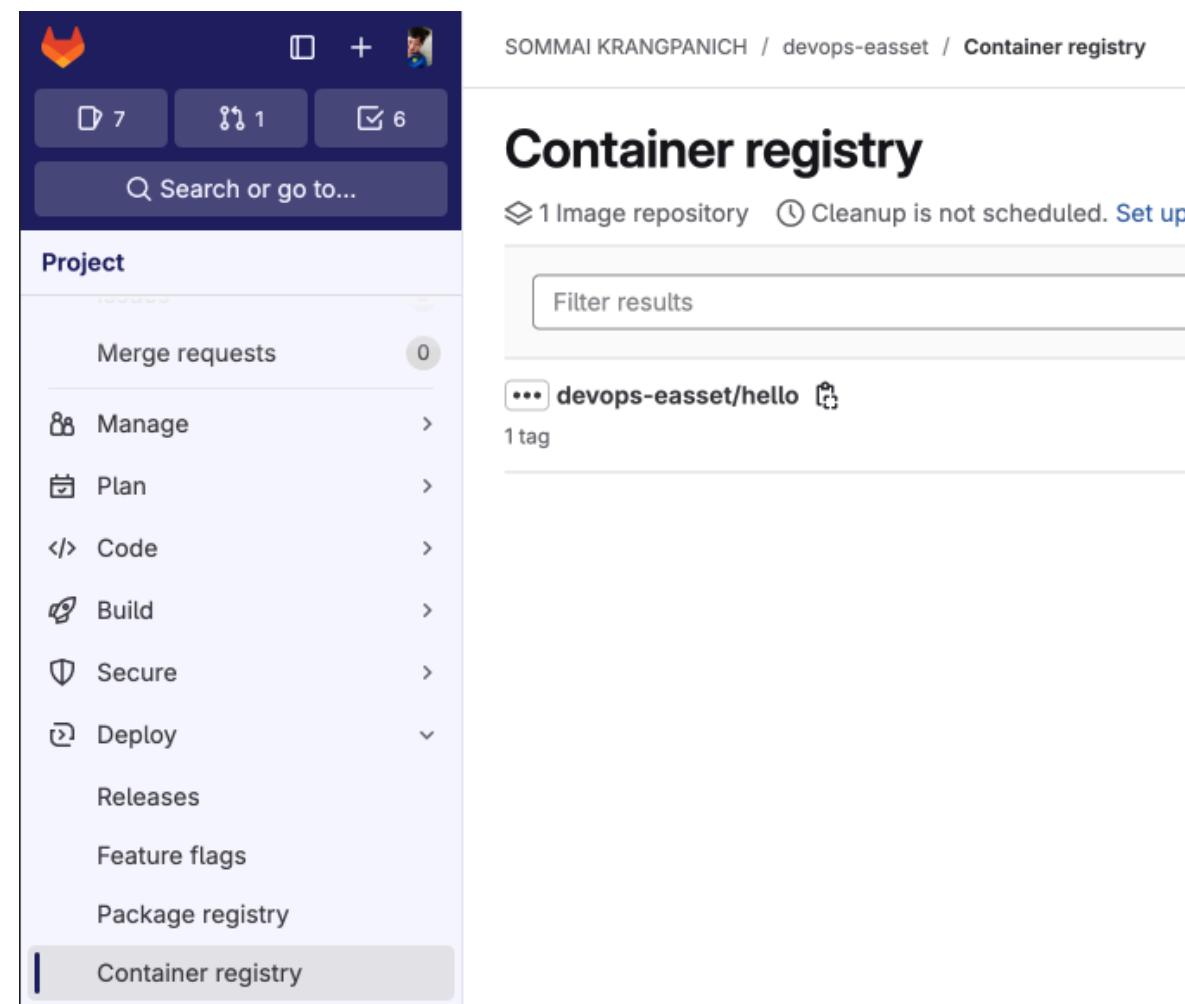
 Pipeline Syntax

 admin-ci-job

## Stage View



# ຕຽວຈັບສອບ docker image ໃນ docker-registry

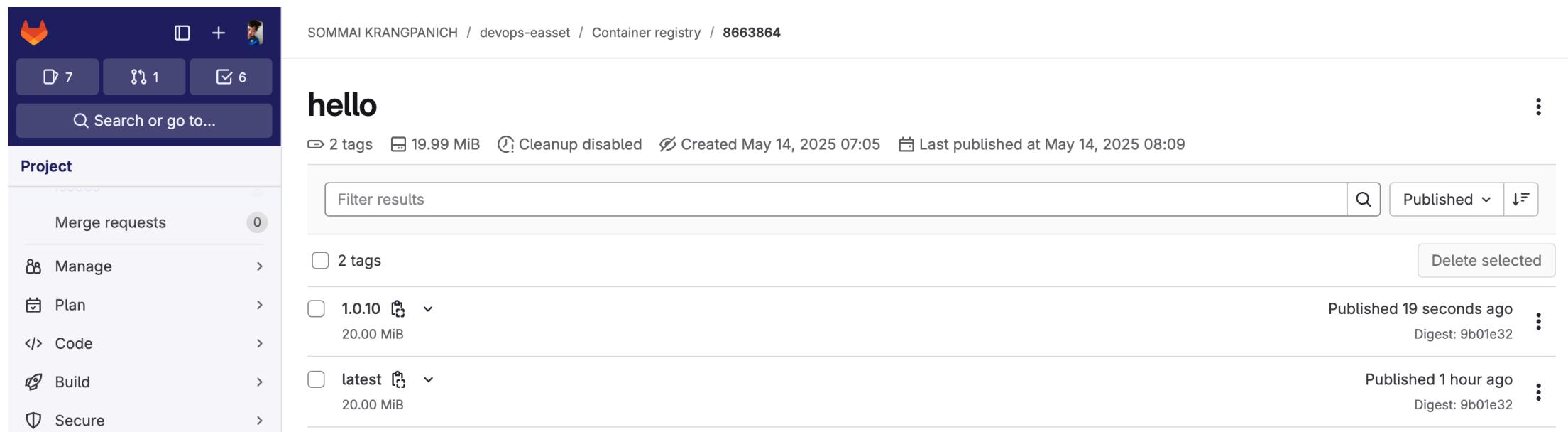


The screenshot shows a software interface for managing a CI/CD pipeline. On the left, there is a sidebar with various project management and development tools listed:

- Merge requests
- Manage
- Plan
- Code
- Build
- Secure
- Deploy
- Releases
- Feature flags
- Package registry
- Container registry

The "Container registry" option is highlighted with a blue bar at the bottom of the sidebar. To the right of the sidebar, the main content area displays the "Container registry" section. The top navigation bar shows the user's name (SOMMAI KRANGPANICH), the repository name (devops-easset), and the current page (Container registry). A message indicates there is 1 image repository and that cleanup is not scheduled, with a link to "Set up cleanup". Below this, there is a "Filter results" input field and a list of repositories. The first repository listed is "devops-easset/hello", which has 1 tag.

# ຕຽວຈສອບ docker image version ຫຼື docker-registry



SOMMAI KRANGPANICH / devops-easset / Container registry / 8663864

**hello**

2 tags 19.99 MiB Cleanup disabled Created May 14, 2025 07:05 Last published at May 14, 2025 08:09

Project

- Merge requests 0
- Manage >
- Plan >
- Code >
- Build >
- Secure >

Filter results   Published

Tag	Size	Published	Digest
2 tags			
1.0.10	20.00 MiB	Published 19 seconds ago	Digest: 9b01e32
latest	20.00 MiB	Published 1 hour ago	Digest: 9b01e32

# Workshop #9

## Kubernetes deployment yml

# create file hello-deploy.yml ລາຍໃຕ້ folder k8s

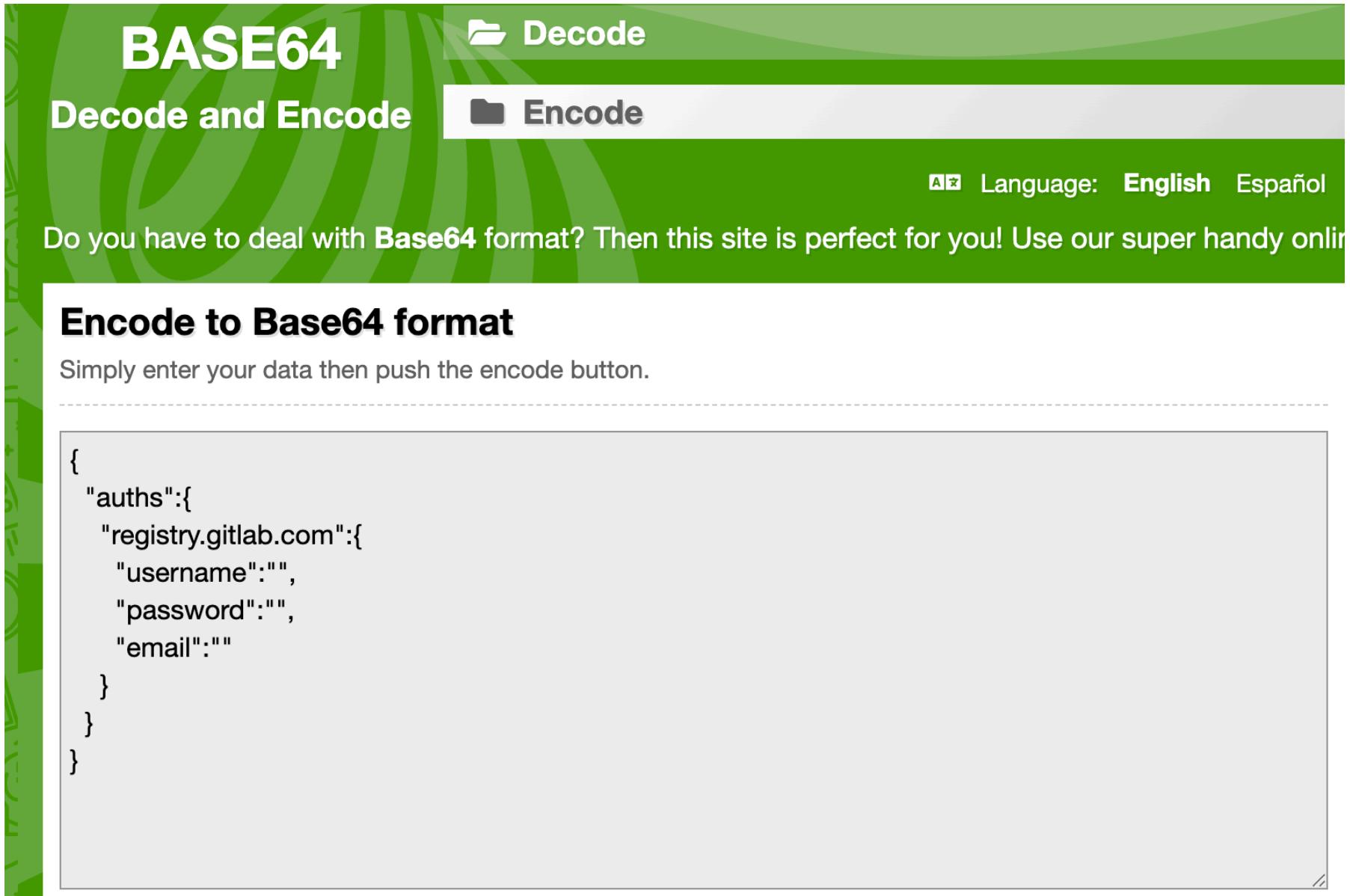
- ສ້າງ namespace

```
apiVersion: v1
kind: Namespace
metadata:
  name: <insert-namespace-name-here>
```

# สร้าง secret

- สำหรับการ pull private image
- เข้า website <https://www.base64encode.org/>
- ใส่ข้อมูลเพื่อเข้ารหัส

```
{  
  "auths": {  
    "registry.gitlab.com": {  
      "username": "",  
      "password": "",  
      "email": ""  
    }  
  }  
}
```



The screenshot shows a web application titled "BASE64" with a green header. The header includes navigation links for "Decode" and "Encode", language selection ("English" and "Español"), and a search bar. Below the header, a message encourages users to use the site for dealing with Base64 format. The main content area is titled "Encode to Base64 format" and contains a text input field where a JSON object is being typed.

```
{  
  "auths":{  
    "registry.gitlab.com":{  
      "username": "",  
      "password": "",  
      "email": ""  
    }  
  }  
}
```

# แก้ไข hello-deploy.yml

- เพิ่มส่วนการสร้าง secret

```
---
```

```
apiVersion: v1
kind: Secret
metadata:
  namespace: <name-space>
  name: regcred
type: kubernetes.io/dockerconfigjson
data:
  .dockerconfigjson: >-
    <base64>
```

- <name-space> <base64> มาจากขั้นตอนด้านบน

## แก้ไข hello-deploy.yml

- เพิ่มส่วนการสร้าง deployment

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: <name-space>
  name: hello
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
    spec:
      containers:
        - name: hello
          image: <your-image-name>
        ports:
          - containerPort: 80
  imagePullSecrets:
    - name: regcred
```

หมายเหตุ อย่าลืมแก้ <name-space> <your-image-name>

# แก้ไข hello-deploy.yml

- เพิ่มส่วนการสร้าง service

```
---  
apiVersion: v1  
kind: Service  
metadata:  
  namespace: <name-space>  
  name: hello  
spec:  
  selector:  
    app: hello  
  ports:  
    - port: 8080  
      targetPort: 80  
  type: ClusterIP
```

หมายเหตุ อย่าลืมแก้ <name-space>

## แก้ไข hello-deploy.yml

- เพิ่มส่วนการสร้าง network ingress

```
---
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: <name-space>
  name: http-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
    nginx.ingress.kubernetes.io/use-regex: "true"
spec:
  rules:
    - http:
        paths:
          - path: /<your-path>(/|$(.*))
            pathType: ImplementationSpecific
            backend:
              service:
                name: hello
                port:
                  number: 8080
```

หมายเหตุ อย่าลืมแก้ <name-space> <your-path>

# push code ទៅ gitlab

```
git add k8s/hello-deploy.yml  
git commit -m "add hello deploy"  
git push
```

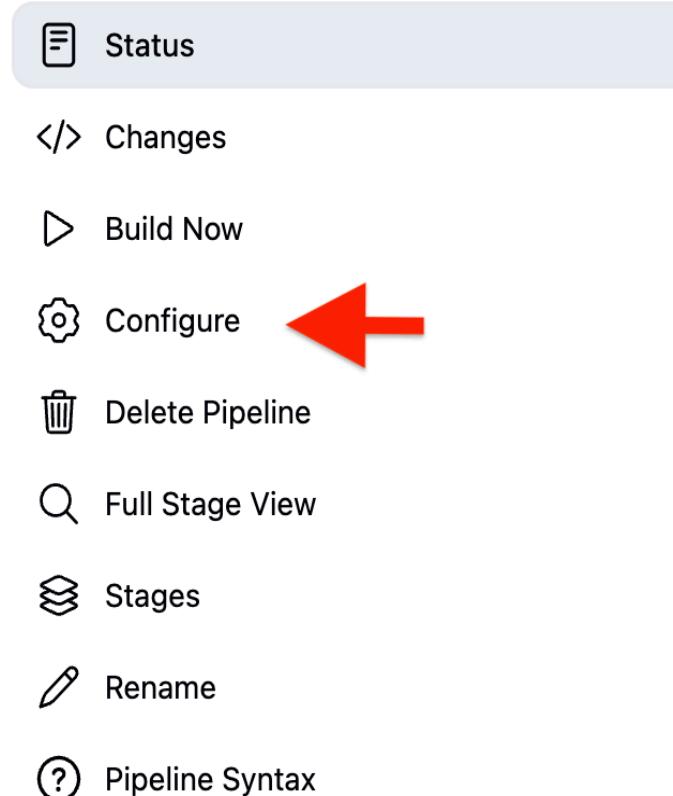
# Workshop #10

**Jenkins automate CI/CD with gitlab docker and kubernetes**

# แก้ไข Job ใน Jenkins

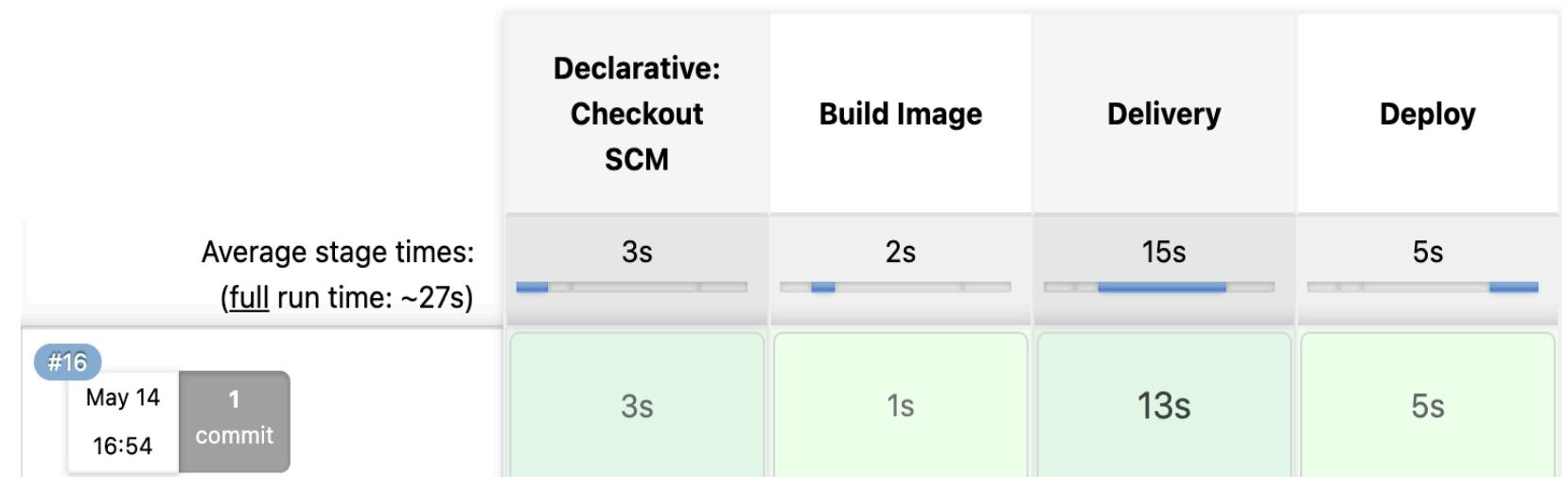
- เลือก configure

Dashboard > admin-ci-job >



## admin-ci-job

### Stage View



# แก้ไข Job ใน Jenkins

- เลือก Poll SCM

## Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- Build after other projects are built [?](#)
- Build periodically [?](#)
- GitHub hook trigger for GITScm polling [?](#)
- Poll SCM [?](#) 
- Trigger builds remotely (e.g., from scripts) [?](#)

# แก้ไข Job ใน Jenkins

- จะบุข้อมูลดังนี้ H/5 \* \* \* \*

## Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

Build after other projects are built ?

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Schedule ?

H/5 \* \* \* \*

Would last have run at Wednesday, May 14, 2025, 10:12:00 AM Coordinated Universal Time; would next run at Wednesday, May 14, 2025, 10:17:00 AM Coordinated Universal Time.

Ignore post-commit hooks ?

Trigger builds remotely (e.g., from scripts) ?

# แก้ไข Jenkinsfile

- เพิ่ม stage Deploy ต่อจาก Delivery

```
stage("Deploy") {  
    agent { label 'k8s-server' }  
    steps {  
        script {  
            try {  
                sh "kubectl set image deployment/hello -n <name space> hello=${IMAGE_NAME}:1.0.${BUILD_NUMBER}"  
                sh "echo update service"  
            } catch (e){  
                sh "kubectl apply -f k8s/hello-deploy.yml"  
                sh "echo create service"  
            }  
        }  
    }  
}
```

# push code ขึ้น gitlab

```
git add Jenkinsfile  
git commit -m "add deploy step"  
git push
```

# หลักการทำงานของ Jenkins

- หลังจากที่ push code ขึ้น gitlab และ Jenkins จะทำงานดังนี้
  - ทุก ๆ 5 นาที Jenkins จะไปทำการตรวจสอบ branch ของ gitlab ว่ามี code เปลี่ยนแปลง หรือไม่
  - ถ้า code มีการเปลี่ยนแปลง Jenkins จะทำการ pull code ลงมาใน workspace
  - Jenkins จะทำการ run script กีล์ส step จาก file Jenkinsfile

# Any questions ?

