

● LIVE

อบรมออนไลน์



ปั้นระบบอัตโนมัติอย่างมือโปรดด้วย

Jenkins และ GitHub Actions ร่วมกับ n8n



สอนสดผ่าน Zoom
รับจำนวนจำกัด

515
วัน
ชั่วโมงเต็ม

หมายสำคัญเริ่มต้น
เริ่มจาก 0 อย่างเข้าใจ



Samit Koyom
สถาบันไอทีเนียส

● LIVE

อบรมออนไลน์

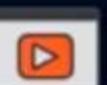
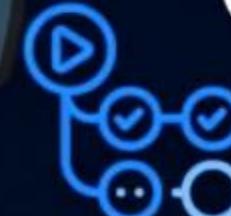


วัน
515
ชั่วโมงเต็ม

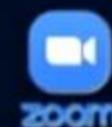
หมายสำหรับผู้เริ่มต้น
เริ่มจาก 0 อย่างเข้าใจ

ปั้นระบบอัตโนมัติอย่างมือโปรดด้วย

Jenkins และ GitHub Actions ร่วมกับ n8n



มีวิดีโอบันทึกการอบรม
ย้อนหลังให้ทุกวัน



สอนสดผ่าน Zoom
รับจำนำแຈักด้



Samit Koyom
สถาบันไอทีนานาชาติ



วิทยากร



อ.สา米ตร โภยม (ปาน)

ปริญญาโท คณะเทคโนโลยีและสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ

▶ Frontend

Angular, React, Vue, Next, Nuxt, Bootstrap, Tailwind CSS

▶ Backend

PHP, Python, Java, Kotlin, Go, Rust, NodeJS, NestJS, .NET

▶ Database

MySQL, PostgreSQL, MS SQL Server, MongoDB, Supabase

▶ Mobile

Java, Kotlin, Objective C, Swift, KMP, Cordova, Flutter ,
React Native, Expo

▶ DevOps

Git, GitHub, Gitlab, Docker, Kubernetes, Jenkins, CI/CD



แบบทดสอบก่อนอบรม



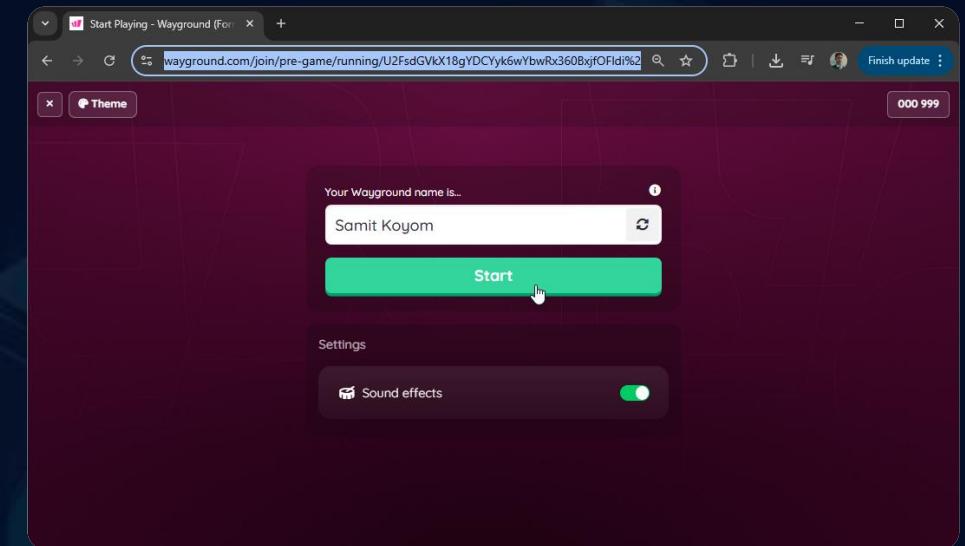
Pretest ทำแบบทดสอบก่อนเรียน

STEP 1: เข้าทำแบบทดสอบที่ลิงค์ ป้อนรหัสเข้าห้องสอบ

wayground.com/join



STEP 2: ป้อนชื่อ

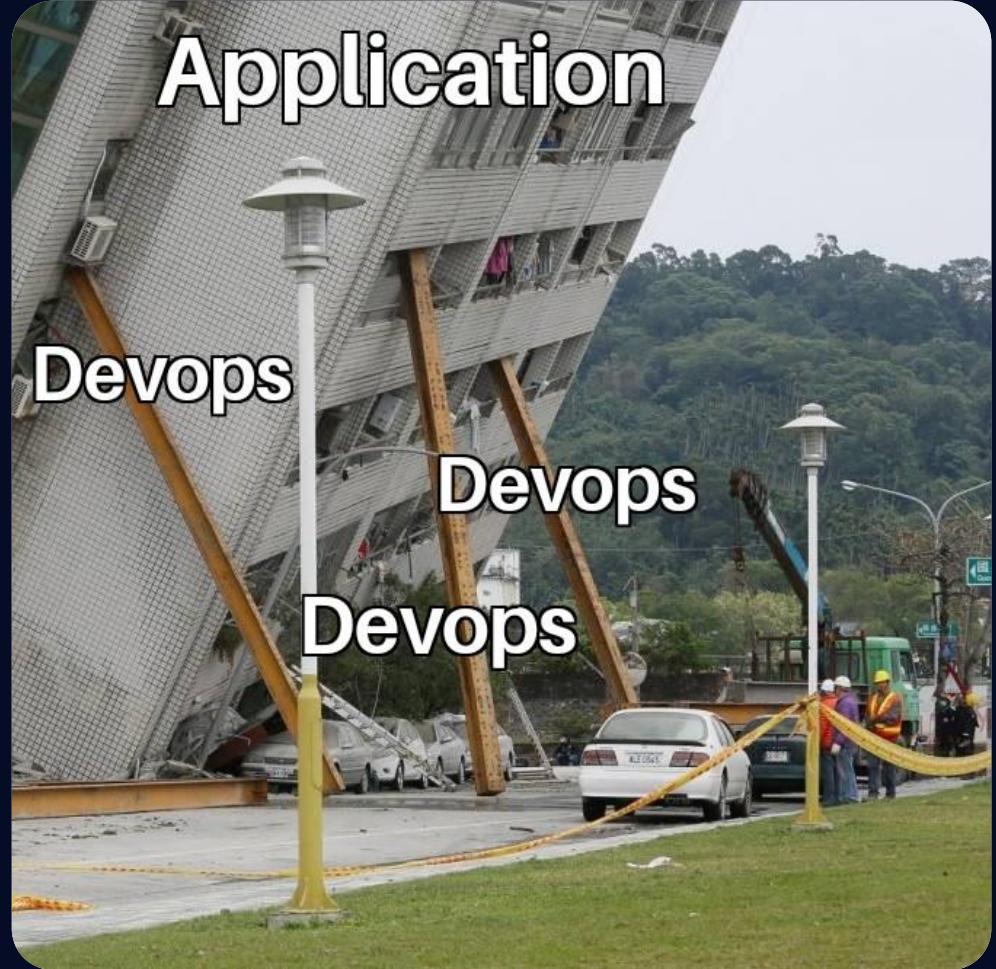


STEP 3: รอผู้สอน Start ข้อสอบ



สถาบันไอทีจีเนียส

www.itgenius.co.th



“

Full Stack ให้ดี ต้องมี Dev
จะให้เทพ ต้องมี Ops ให้ครบสาย

ระบบเสร็จ งาน Deploy ต้องไม่ตาย
สร้าง Pipeline สุดอลัง พังตอนรัน

งาน DevOps เข้าเน้น "Flow" ไม่เน้น "Fix"
ไม่จุกจิก แค่ Config งานนวนเช้า

แค่ Push โค้ด เดียวมัน Test มัน Build เอง ได้ยาวๆ
จะ Fail now หรือ never เดียวเจอกัน

”



GIT CLONE TO PRODUCTION

มาหากกากา!

#ผมว่าไม่គด





ดาวน์โหลดเอกสารประกอบการอบรม

bit.ly/devops_jenkins



สถาบันไอทีเกี้ยนส์

www.itgenius.co.th

● LIVE

อบรมออนไลน์

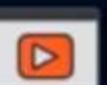
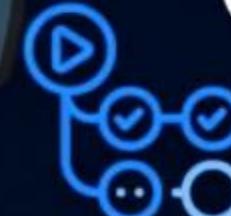


วัน
515
ชั่วโมงเต็ม

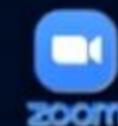
หมายสำหรับผู้เริ่มต้น
เริ่มจาก 0 อย่างเข้าใจ

ปั้นระบบอัตโนมัติอย่างมือโปรดด้วย

Jenkins และ GitHub Actions ร่วมกับ n8n



มีวิดีโอบันทึกการอบรม
ย้อนหลังให้ทุกวัน



สอนสดผ่าน Zoom
รับจำนำแจำกัด



Samit Koyom
สถาบันไอทีนานาชาติ

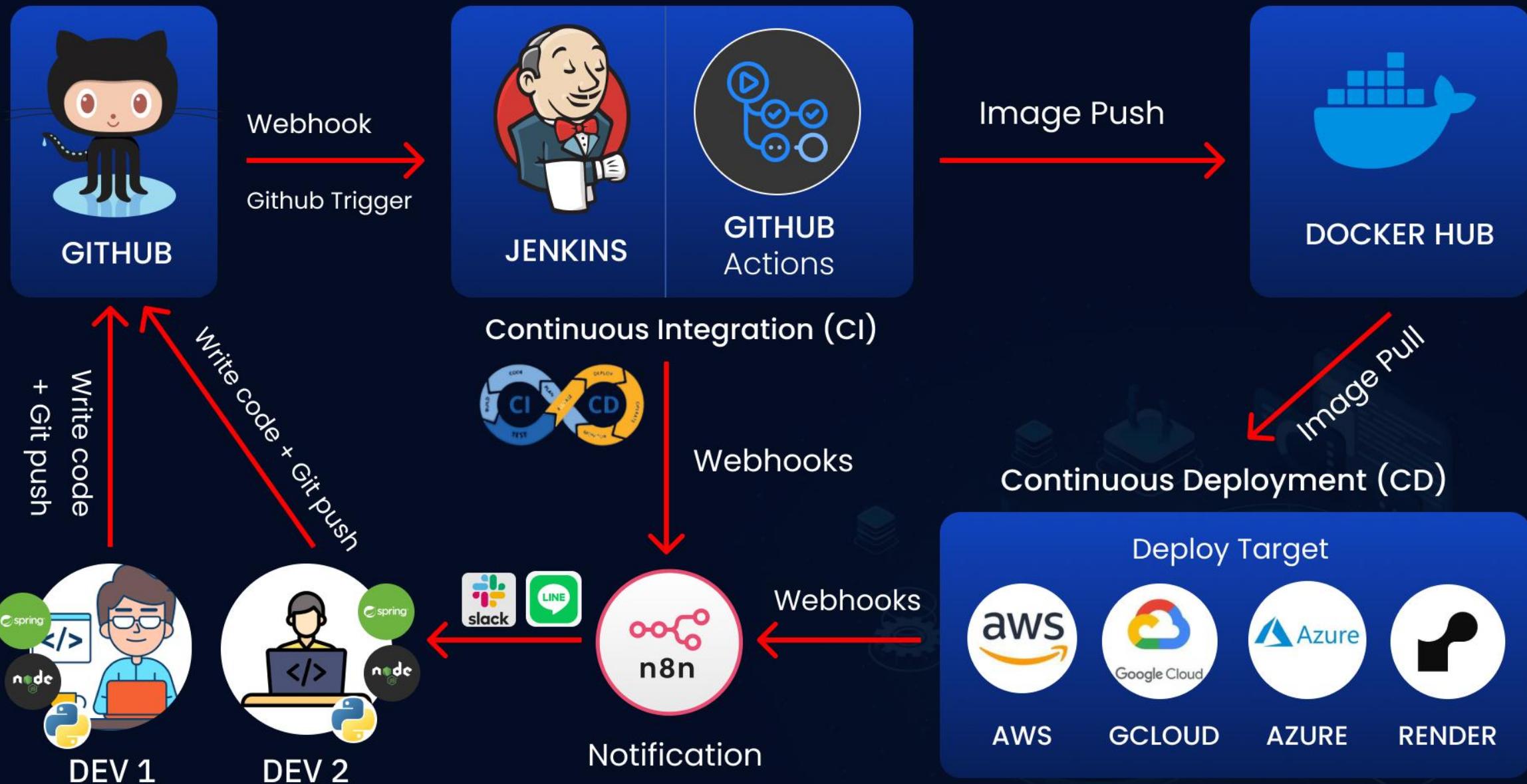


Course Content



- | | | | |
|----|---------------------------------|----|-------------------------------------|
| 01 | Introduction | 06 | Jenkins workflow |
| 02 | Installation & Tools | 07 | Basic GitHub Actions |
| 03 | Basic Git | 08 | GitHub Action workflow |
| 04 | Basic Docker | 09 | N8N Notification |
| 05 | Basic Jenkins | 10 | Integration & Real-World |

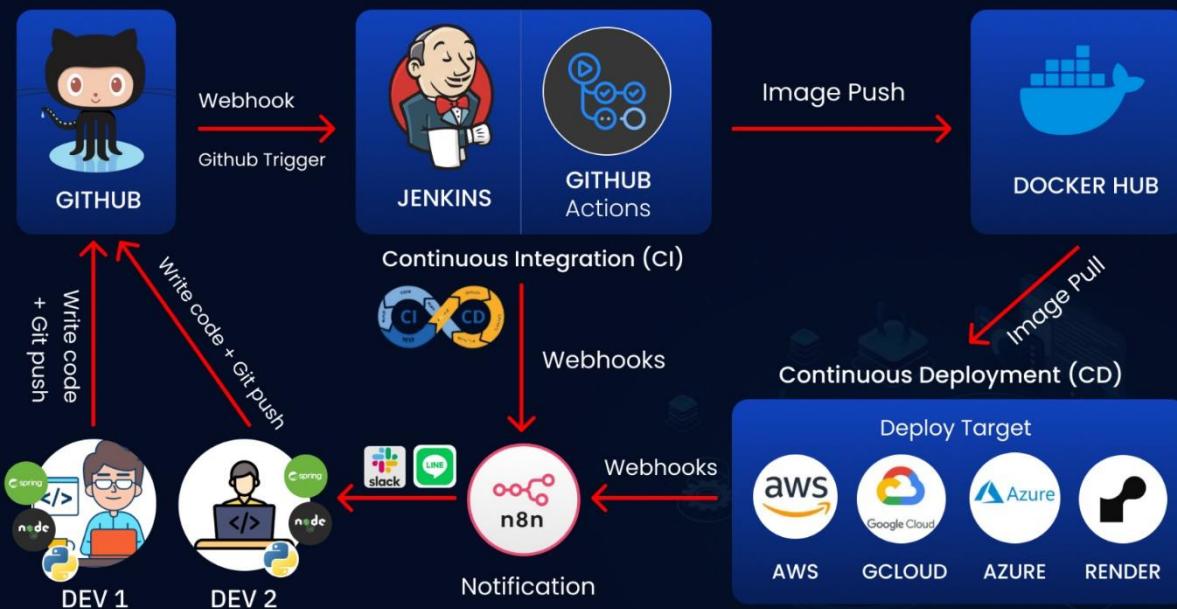




การพร้อมการทำ CI/CD

ใช้ Docker เป็นหัวใจหลักและเน้นการ Deploy ผ่าน Container

ในหลักสูตรนี้มีการใช้ทั้ง Jenkins และ Github Actions เพื่อให้เห็นความแตกต่าง และนำไปใช้ในการตัดสินใจ เรื่องทั้งมีการใช้เครื่องมือ Automation อย่าง N8N เพื่อประยุกต์ทำระบบแจ้งเตือนที่จัดทำขึ้น



Flow ของ Jenkins Pipeline

1. เริ่ม Pipeline ให้ agent ได้รับ
2. Checkout: ดึงซอฟต์แวร์มาสู่ Jenkins (checkout scm)
3. Install & Test
4. Build Docker Image
5. Push Docker Image
6. Cleanup Docker
7. Deploy Local & Real Server
8. Post Success (เฉพาะขั้น Deploy Local)
9. Post Failure (ทั้ง Pipeline)
10. Deploy ไปเซิร์ฟเวอร์โดยผ่าน SSH พร้อม Webhook แจ้งผล

The screenshot shows the Jenkins Pipeline Overview for job #21:

- Graph:** A timeline of steps: Start → Checkout SCM → Checkout → Install & Test → Build Docker Image → Push Docker Image → Cleanup Docker → Deploy Local → End.
- Step Details:**
 - Checkout SCM: 1.5s
 - Checkout: 1.1s
 - Install & Test: 17s
 - Build Docker Image: 14s
 - Push Docker Image: 21s
 - Cleanup Docker: 1.3s
 - Deploy Local: 10s
- Logs:** Shows log entries for each step, including command execution and timestamp.

Flow ของ Github Action Pipeline

1. Trigger เริ่มงาน เรียกใช้งานแบบ Auto Push
2. กำหนดค่ารวมของ Workflow
3. Job: test (Install & Test)
4. Job: build_and_push (Build & Push Docker)
5. Job: smoke_test (Run & Curl)
6. Deploy ไปเซิร์ฟเวอร์ระยะไกลผ่าน SSH

The screenshot shows the GitHub Actions interface for the 'CI/CD - Express Docker App' workflow. It displays a summary of the most recent run, which was triggered manually 4 minutes ago, completed successfully in 2m 56s, and produced 1 artifact. The main workflow file, 'main.yml', is shown with a matrix step for 'Install & Test (Node 22.x)' and a subsequent 'Build & Push Docker' step. Both steps are marked as completed. Below this, there is a 'Docker Build summary' section with a download link for the build record archive and a table showing the details of the Docker build.

ID	Name	Status	Cached	Duration
29C058	express-docker-app (production)	completed	0%	1m5s

Push Image to Docker Hub

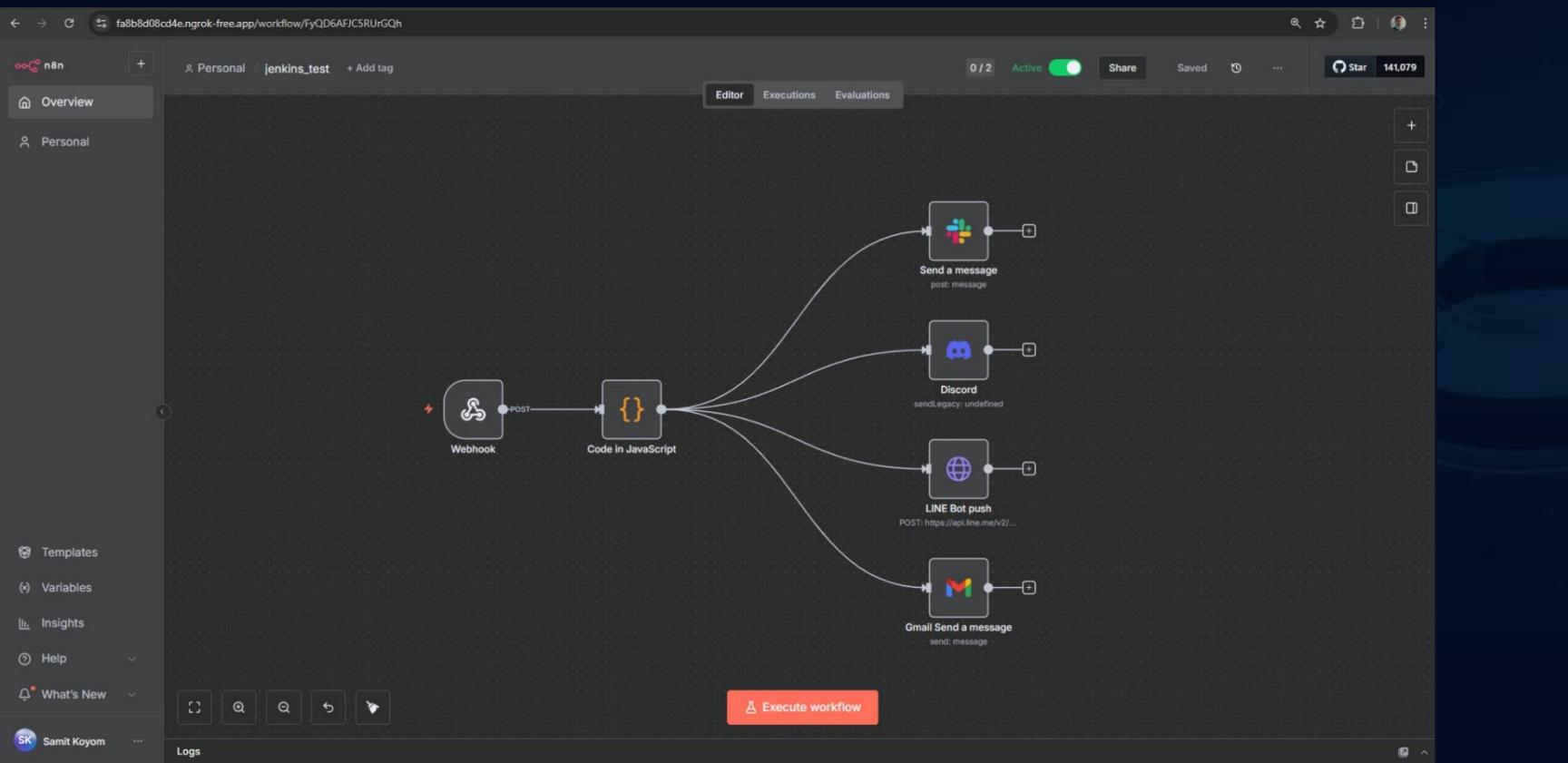
The screenshot shows the Docker Hub repository page for 'iamsamitdev/express-docker-app'. The repository was last pushed 2 minutes ago and has a size of 133.9 MB. The 'Tags' tab is selected, showing two tags: 'sha-5a86497' and 'latest'. The 'sha-5a86497' tag was pushed 2 minutes ago and has a digest of f70c93007659, OS/ARCH of linux/amd64, and a compressed size of 55.35 MB. The 'latest' tag was also pushed 2 minutes ago and has the same digest, OS/ARCH, and compressed size. On the right side, there are 'Docker commands' for pushing a new tag to the repository, and a 'Public view' button.

TAG	Digest	OS/ARCH	Last pull	Compressed size
sha-5a86497	f70c93007659	linux/amd64	less than 1 day	55.35 MB
latest	f70c93007659	linux/amd64	less than 1 day	55.35 MB



Webhook to n8n for Notification

- แจ้งเตือนผ่าน Slack
- แจ้งเตือนผ่าน Discord
- แจ้งเตือนผ่าน Line
- แจ้งเตือนผ่าน Email
- และอื่นๆ ตามต้องการ



Workshop การเขียน Pipeline
เพื่อ Deploy กัน 3 ภาษา 3 framework ยอดนิยม



Express Docker Application



โปรเจกต์ Express.js + TypeScript REST API ที่ทำงานใน Docker Container พร้อมด้วย CI/CD Pipeline ผ่าน Jenkins, Github Actions และการแจ้งเตือนผ่าน N8N

Flask Docker Application



โปรเจกต์ Flask API แบบ RESTful ที่ทำงานใน Docker Container พร้อมด้วย CI/CD Pipeline ผ่าน Jenkins, Github Actions และการแจ้งเตือนผ่าน N8N

Spring Boot Docker Application

โปรเจกต์ Spring Boot REST API ที่ทำงานใน Docker Container พร้อมด้วย CI/CD Pipeline ผ่าน Jenkins, GitHub Actions และการแจ้งเตือนผ่าน N8N



ตัวอย่าง Workshop Node.js Express Pipeline



Features

- ✓ **Express.js REST API** - Fast, minimal Node.js web framework
- ◆ **TypeScript Support** - Type-safe JavaScript development
- ◆ **Docker Containerization** - Lightweight Alpine-based container
- ◆ **Jest Testing** - Comprehensive test suite with Supertest
- ◆ **CI/CD Pipeline** - Automated Jenkins pipeline with deployment
- ◆ **GitHub Actions** - Modern CI/CD with GitHub-native automation
- ◆ **Test Coverage** - Code coverage reports and analysis
- ◆ **Docker Hub Integration** - Automated image publishing
- ◆ **Server Deployment** - Automated deployment to remote servers
- ◆ **Semantic Versioning** - Build number based tagging
- ◆ **Hot Reload** - Development with ts-node

Project Structure

```
express-docker-app/
├── src/
│   └── app.ts          # Main Express application (TypeScript)
├── tests/
│   └── app.test.ts     # Jest test suite with Supertest
├── dist/
│   └── app.js          # Compiled JavaScript output
├── node_modules/
├── .github/
│   └── workflows/
│       └── main.yml    # GitHub Actions workflow
├── Dockerfile          # Docker build configuration
├── Jenkinsfile         # Jenkins CI/CD pipeline
├── jest.config.js      # Jest testing configuration
├── package.json         # Node.js project configuration
├── package-lock.json    # Dependency lock file
├── tsconfig.json        # TypeScript configuration
└── README.md           # Project documentation
```

ตัวอย่าง Workshop Python Flask Pipeline



Features

- ✓ **Flask REST API** - Simple Hello World API endpoint
- ◆ **Docker Support** - Containerized application for easy deployment
- ◆ **Unit Testing** - Comprehensive test suite with pytest
- ◆ **CI/CD Pipeline** - Automated Jenkins pipeline
- ◆ **GitHub Actions** - Modern CI/CD with GitHub-native automation
- ◆ **Test Reports** - JUnit XML test result reporting
- ◆ **Docker Hub Integration** - Automated image publishing
- ◆ **N8N Notifications** - Webhook-based notifications
- ◆ **Semantic Versioning** - Build number based tagging

Project Structure

```
flask-docker-app/
├── tests/              # Test directory
│   ├── __init__.py      # Test package initializer
│   ├── conftest.py       # Pytest configuration
│   └── test_app.py       # Application tests
├── __pycache__/         # Python bytecode cache
├── .github/             # GitHub Actions workflow
│   └── workflows/
│       └── main.yml      # GitHub Actions workflow
├── app.py               # Main Flask application
├── Dockerfile            # Docker build configuration
├── Jenkinsfile          # Jenkins CI/CD pipeline
├── pytest.ini            # Pytest configuration
├── requirements.txt      # Python dependencies
├── Readme.md             # Project documentation
└── .gitignore            # Git ignore rules
```

Prerequisites

- Python 3.13+
- Docker & Docker Compose
- Ci/CD

ตัวอย่าง Workshop Java SpringBoot Pipeline



Features

- ✓ **Spring Boot REST API** - RESTful API endpoints with comprehensive functionality
- ◆ **Multi-stage Docker Build** - Optimized Docker image with build and runtime stages
- ◆ **Java 21** - Latest LTS Java version support
- ◆ **Comprehensive Testing** - Unit tests, integration tests, and test coverage
- ◆ **CI/CD Pipeline** - Automated Jenkins pipeline with Maven
- ◆ **GitHub Actions** - Modern CI/CD with GitHub-native automation
- ◆ **JaCoCo Coverage** - Code coverage reports and analysis
- ◆ **Docker Hub Integration** - Automated image publishing
- ◆ **N8N Notifications** - Webhook-based notifications
- ◆ **Semantic Versioning** - Build number based tagging
- ◆ **Health Check** - Built-in health monitoring endpoint

Project Structure

```
springboot-docker-app/
├── src/
│   └── main/
│       ├── java/
│       │   └── com/example/demo/
│       │       └── DemoApplication.java      # Main Spring Boot application
│       ├── controller/
│       │   └── HelloController.java        # REST API controller
│       └── model/
│           └── GreetingRequest.java      # Request/Response model
├── resources/
│   ├── application.properties        # Application configuration
│   ├── static/
│   └── templates/                   # Template files
└── test/
```

โปรเจกต์ทำงานบน Docker Container ทั้งหมด

```

Dockerfile X
express-docker-app > Dockerfile > ...
You, 45 minutes ago | 1 author (You)
1 # ใช้ Official Node.js image (เวอร์ชัน Long Term Support) เป็น base image
2 FROM node:22-alpine
3
4 # กำหนด Working Directory ภายใน Container
5 WORKDIR /app
6
7 # Copy ไฟล์ package.json และ package-lock.json เข้าไปเก็บ
8 # เพื่อใช้สำหรับ Docker cache layer ที่อาจมีผลต่อ install dependencies ใหม่ๆทุกครั้งที่มีการ pull
9 COPY package*.json .
10
11 # ติดตั้ง Dependencies
12 RUN npm install
13
14 # Copy ไฟล์ทั้งหมดในโปรเจกต์เข้าไปใน container
15 COPY .
16
17 # กำหนด Port ที่ Container จะทำงาน
18 EXPOSE 3000
19
20 # ค่าสีสำหรับ Express Application
21 CMD ["npm", "start:ts"]

Dockerfile X
flask-docker-app > Dockerfile > ...
You, 47 minutes ago | 1 author (You)
1 # ใช้ Official Python image เป็น base image
2 FROM python:3.13-slim
3
4 # กำหนด Working Directory ภายใน Container
5 WORKDIR /app
6
7 # Copy ไฟล์ requirements.txt เข้าไปเก็บ เพื่อใช้ cache layer ของ Docker
8 COPY requirements.txt .
9
10 # ติดตั้ง Dependencies ที่ระบุไว้
11 RUN pip install --no-cache-dir -r requirements.txt
12
13 # Copy ไฟล์ทั้งหมดในโปรเจกต์เข้าไปใน container
14 COPY .
15
16 # กำหนด Port ที่ Container จะทำงาน
17 EXPOSE 5000
18
19 # ค่าสีสำหรับ Flask Application
20 CMD ["python", "app.py"]

Dockerfile X
springboot-docker-app > Dockerfile > ...
You, 52 minutes ago | 1 author (You)
1 # STAGE 1: Build Stage - ใช้ Maven Image เพื่อ Build โปรเจกต์
2 FROM maven:3.8.5-openjdk-21 AS build
3 WORKDIR /app
4 COPY pom.xml .
5 RUN mvn dependency:go-offline
6 COPY src ./src
7 RUN mvn package -DskipTests
8
9 # STAGE 2: Run Stage - ใช้ JRE Image ขนาดเล็กเพื่อ Run แยก
10 FROM eclipse-temurin:21-jre-focal
11 WORKDIR /app
12 COPY --from=build /app/target/*.jar app.jar
13 EXPOSE 8080
14 ENTRYPOINT ["java", "-jar", "app.jar"]

```

เขียน Jenkinsfile ทำ Pipeline แยกแต่ละโปรเจกต์

```

Jenkinsfile X
express-docker-app > Jenkinsfile
You, 47 minutes ago | 1 author (You)
1 pipeline {
2   agent any
3
4   environment {
5     DOCKER_HUB_CREDENTIALS = credentials ('dockerhub-cred')
6     DOCKER_REPO = "your-dockerhub-username/express-docker-app"
7     APP_NAME = "express-docker-app"
8     DEPLOY_SERVER = "user@your-server-ip"
9   }
10
11   stages {
12     stage('Checkout') {
13       steps {
14         git branch: 'main', url: 'https://github.com/your-repo/express-docker-app.git'
15       }
16     }
17
18     stage('Install & Test') {
19       steps {
20         sh ...
21         npm install
22         npm test
23       }
24     }
25
26     stage('Build Docker Image') {
27       steps {
28         script {
29           dockerImage = docker.build (
30             "${DOCKER_REPO}#${BUILD_NUMBER}"
31           )
32         }
33       }
34     }
35
36     stage('Push Docker Image') {
37       steps {
38         script {
39           docker.withRegistry (
40             'https://index.docker.io/v1/',
41             "${DOCKER_HUB_CREDENTIALS}"
42           ) {
43             dockerImage.push()
44             dockerImage.push ("latest")
45           }
46         }
47       }
48     }
49
50     stage('Deploy to Server') {
51       steps {
52         sshagent (
53           ['deploy-server-cred']
54         ) {
55           sh """
56             You, 48 minutes
57           """
58         }
59       }
60     }
61   }
62 }

```

เขียน Github Actions ทำ Pipeline แยกแต่ละโปรเจกต์

```

main.yml X
express-docker-app > _github > workflows > main.yml
You, 51 minutes ago | 1 author (You)
1 # ชื่อของ Workflow ที่จะแสดงในหน้า Actions ของ GitHub
2 name: CI/CD - Express Docker App
3
4 # กำหนด Trigger: ให้ Workflow นี้ทำงานทุกครั้งที่มีการ push ไปยัง branch 'main'
5 on:
6   push:
7     branches: [ "main" ]
8
9 # กำหนด Jobs ที่จะให้ทำงาน
10 jobs:
11   build-and-push:
12     # กำหนดสภาพแวดล้อมที่จะรัน Job นี้ (ใช้ Ubuntu ของ GitHub)
13     runs-on: ubuntu-latest
14
15     # กำหนดขั้นตอนการทำงาน (Steps)
16     steps:
17       # 1. Checkout Code: ลีดโค้ดจาก Repository ลงมาที่ Runner
18       - name: Checkout repository
19         uses: actions/checkout@v4
20
21       # 2. Login to Docker Hub: ล็อกอินเข้า Docker Hub โดยใช้ Secrets
22       - name: Log in to Docker Hub
23         uses: docker/login-action@v3
24         with:
25           username: ${{ secrets.DOCKERHUB_USERNAME }}
26           password: ${{ secrets.DOCKERHUB_TOKEN }}
27
28       # 3. Build and Push Docker Image: สร้างและ Push Image
29       - name: Build and push Docker image
30         uses: docker/build-push-action@v5
31         with:
32           context: .
33           push: true
34           tags: ${{ secrets.DOCKERHUB_USERNAME }}/
35           my-express-api:latest, ${{ secrets.DOCKERHUB_USERNAME }}}/my-express-api:${{ secrets.github.sha }}
36
37       # 4. Notify N8N on Success: แจ้งเตือน N8N เมื่อสำเร็จ
38       - name: Notify N8N on Success
39         if: success() # ทำงานเมื่อ Step ก่อนหน้าทั้งหมดสำเร็จ
40         run:
41           curl -X POST -H "Content-Type: application/json" \
42             -d '{
43               "status": "SUCCESS",
44               "project": "${{ github.repository }}"
45             }',
46             You, 51 minutes

```

● LIVE

อบรมออนไลน์



ปั้นระบบอัตโนมัติอย่างมือโปรดด้วย

Jenkins และ GitHub Actions ร่วมกับ n8n



มีวิดีโอบันทึกการอบรม
ย้อนหลังให้ทุกวัน

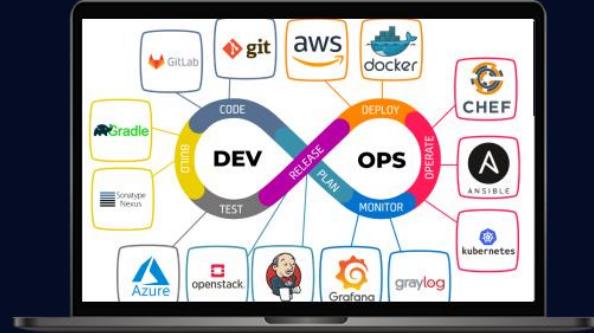


สอนสดผ่าน Zoom
รับจำนวนจำกัด



Samit Koyom
สถาบันไอทีจีเนียส

Day 1



1. พื้นฐานและการติดตั้งเครื่องมือ DevOps CI/CD
2. พื้นฐาน Git



Jenkins
และ GitHub Actions
ร่วมกับ n8n

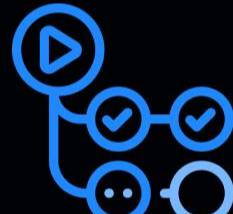
ปั้นระบบอัตโนมัติอย่างมือโปรดด้วย

Jenkins

และ GitHub

Actions

ร่วมกับ **n8n**



มีวิดีโอบันทึกการอบรม
ย้อนหลังให้ทุกวัน



สถาบันไอทีเนียส

วัน
5 15
ซั่วโมงเต็ม



การเตรียม
เครื่องมือ



Samit Koyom
สถาบันไอทีเนียส

โปรแกรม (Tool and Editor) ที่ใช้อบรม

1. Visual Studio Code
2. Java JDK 21.x
3. Node.js 22.x
4. Python 3.1x
5. Docker Desktop
6. Git

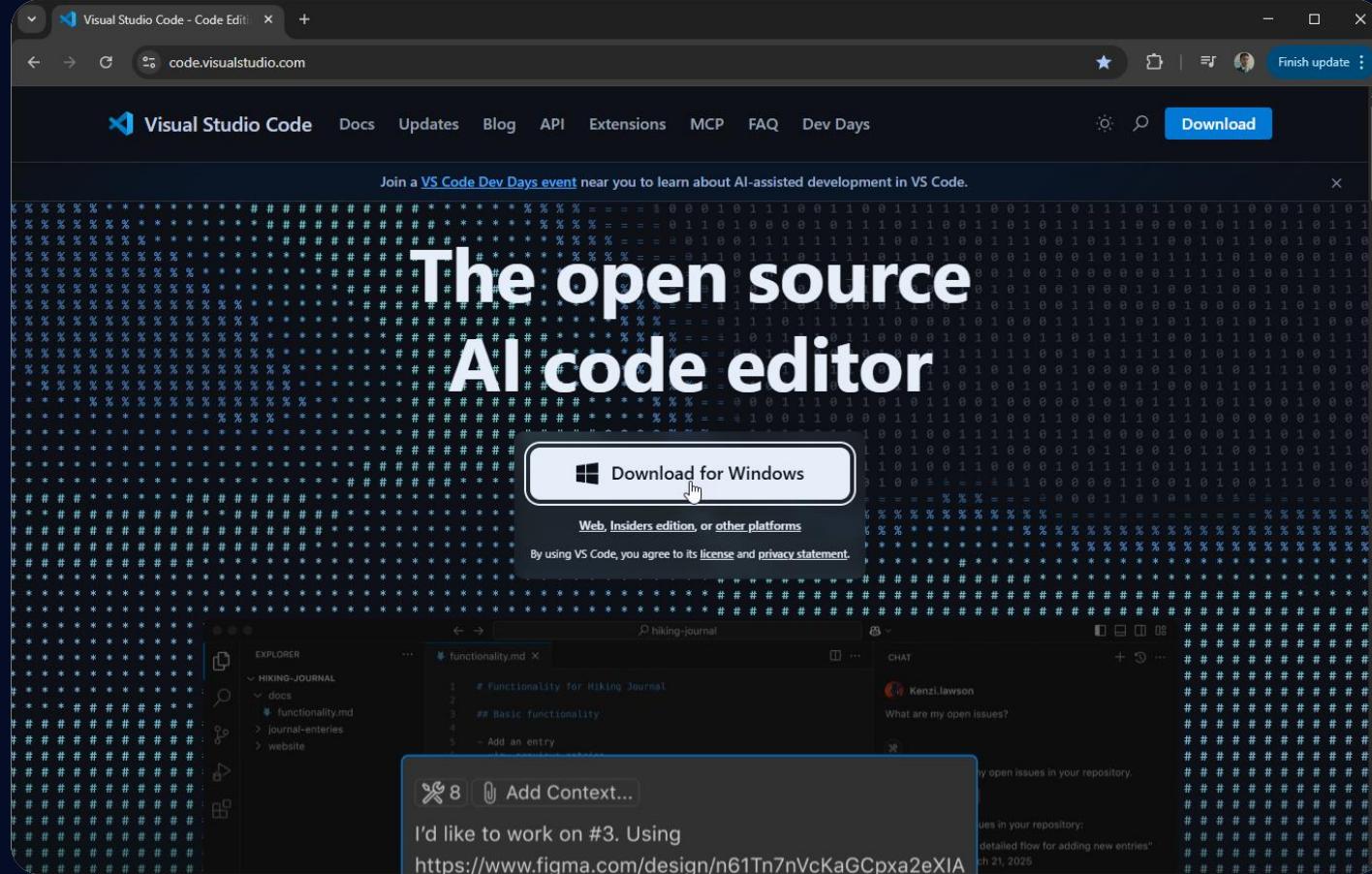




1. ติดตั้ง Visual Studio Code



ຕັດຕັ້ງ Visual Studio Code ວຽກເສຍເວັບໄຊທີ່ຈຳເປັນ



ເຂົ້າໄປດາວໂຫລດ Visual Studio Code ໄດ້ <https://code.visualstudio.com>



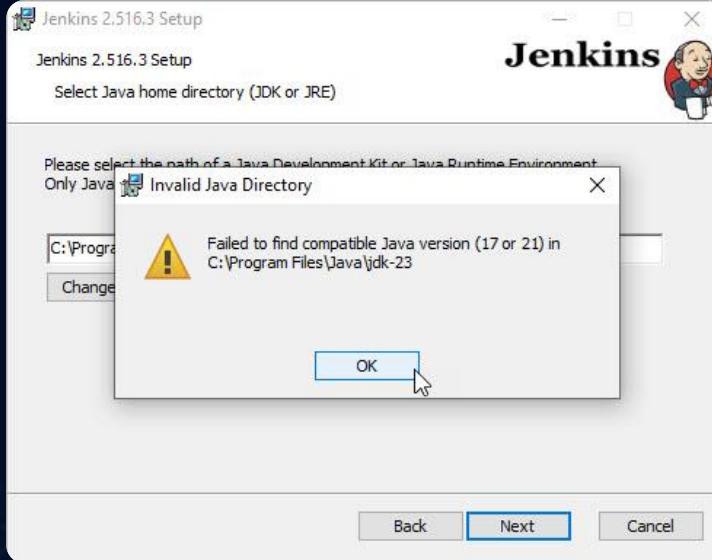
การติดตั้งส่วนเสริม (Extension) ของ Visual Studio Code



รายชื่อ Extensions ที่แนะนำสำหรับ VS Code

- 1. Material Icon Theme** by Philipp Kief
- 2. Python** by Microsoft
- 3. Language Support for Java(TM)** by Red Hat by Red Hat
- 4. Docker** by Microsoft
- 5. Jenkins** by p1c2u
- 6. GitHub Actions** by Github
- 7. One Dark Pro** by binaryify





2. ติดตั้ง Java JDK 21.x



หมายเหตุ ในคอร์สนี้ Jenkins ต้องการ Java JDK **17** หรือ **21** เท่านั้น (เก่าหรือใหม่กว่านี้ไม่รองรับ)



The screenshot shows the Oracle Java Downloads page at oracle.com/java/technologies/downloads/#jdk21-windows. The 'Windows' tab is selected, highlighted with a red box. A large orange button labeled 'Windows เลือกดังนี้' is positioned to the right of the download links. The table lists three download options:

Product/file description	File size	Download
x64 Compressed Archive	186.05 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip (sha256)
x64 Installer	164.42 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe (sha256)
x64 MSI Installer	163.16 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi (sha256)

Below the table, there is a 'Documentation Download' button and a 'Release information' section with links to 'Online Documentation' and 'Installation Instructions'. A URL link is also present at the bottom of the section.

ดาวน์โหลด Java JDK 21 ได้ที่ <https://www.oracle.com/java/technologies/downloads/#jdk21-windows>



The screenshot shows the Oracle Java Downloads page for macOS. The URL in the address bar is oracle.com/java/technologies/downloads/#jdk21-mac. The page title is "Java SE Development Kit 21.0.8 downloads". It states that JDK 21 binaries are free to use in production and redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions \(NFTC\)](#). It also mentions that JDK 21 will receive updates under the NFTC until September 2026, after which updates will be licensed under the [Java SE OTN License \(OTN\)](#).

The "macOS" tab is selected, highlighted with a red box. Other tabs available are "Linux" and "Win". An orange box highlights the "MacOS เลือกดังนี้" section.

Product/file description	File size	Download
ARM64 Compressed Archive	181.46 MB	https://download.oracle.com/java/21/latest/jdk-21_macos-aarch64_bin.tar.gz (sha256)
ARM64 DMG Installer	180.79 MB	https://download.oracle.com/java/21/latest/jdk-21_macos-aarch64_bin.dmg (sha256)
x64 Compressed Archive	183.65 MB	https://download.oracle.com/java/21/latest/jdk-21_macos-x64_bin.tar.gz (sha256)
x64 DMG Installer	183.00 MB	https://download.oracle.com/java/21/latest/jdk-21_macos-x64_bin.dmg (sha256)

Buttons at the bottom include "Documentation Download" and "Release information".

Annotations on the right side highlight the download links for "MacOS CPU Arm" and "MacOS CPU Intel" with orange boxes.

ดาวน์โหลด Java JDK 21 ได้ที่ <https://www.oracle.com/java/technologies/downloads/#jdk21-mac>



The screenshot shows the Oracle Java Downloads page for JDK 21 on a Linux system. The 'Linux' tab is selected, highlighted with a red border. The page lists five download options:

Product/file description	File size	Download
ARM64 Compressed Archive	186.07 MB	https://download.oracle.com/java/21/latest/jdk-21_linux-aarch64_bin.tar.gz (sha256)
ARM64 RPM Package	185.76 MB	https://download.oracle.com/java/21/latest/jdk-21_linux-aarch64_bin.rpm (sha256) (OL 9 GPG Key)
x64 Compressed Archive	187.89 MB	https://download.oracle.com/java/21/latest/jdk-21_linux-x64_bin.tar.gz (sha256)
x64 Debian Package	159.73 MB	https://download.oracle.com/java/21/latest/jdk-21_linux-x64_bin.deb (sha256)
x64 RPM Package	187.55 MB	https://download.oracle.com/java/21/latest/jdk-21_linux-x64_bin.rpm (sha256) (OL 9 GPG Key)

Three download links are highlighted with orange boxes and red borders: the ARM64 RPM Package, the x64 Compressed Archive, and the x64 RPM Package.

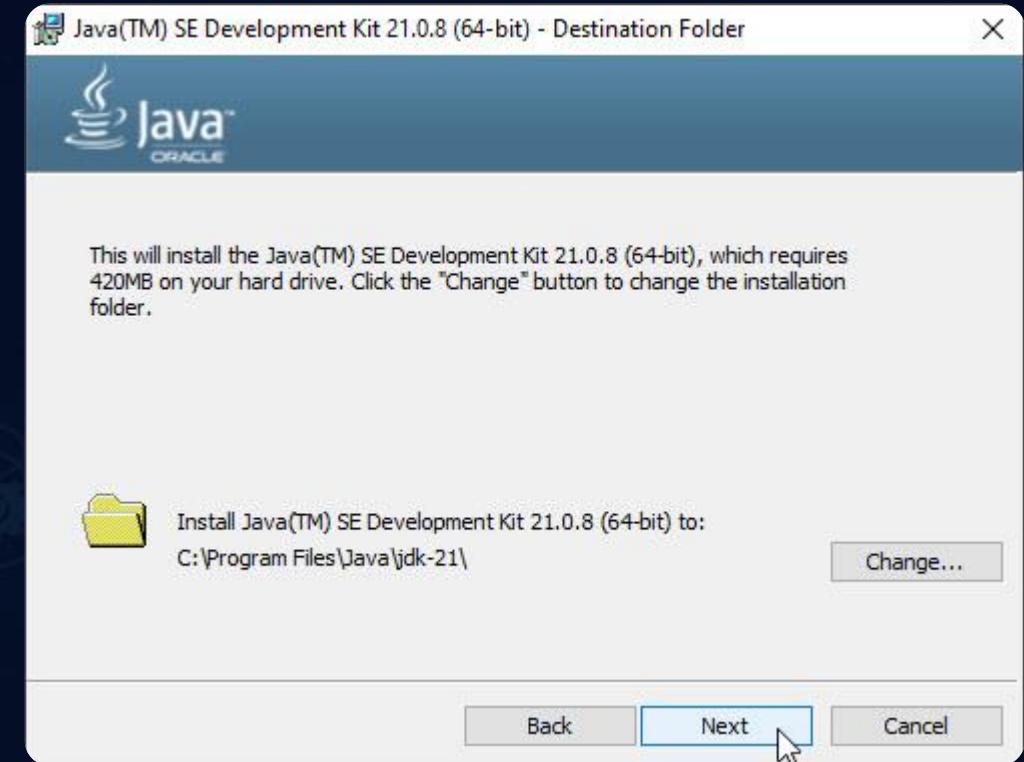
ดาวน์โหลด Java JDK 21 ได้ที่ <https://www.oracle.com/java/technologies/downloads/#jdk21-linux>



หลังดาวน์โหลดไฟล์ทำการคลิกเริ่มติดตั้งกัน



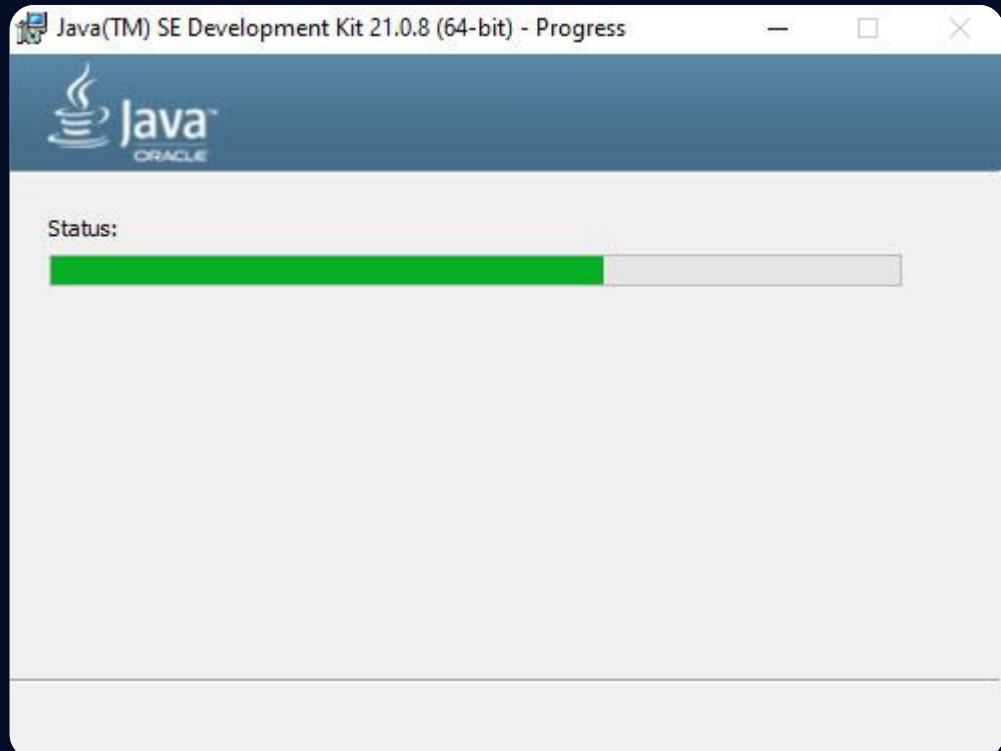
คลิก Next



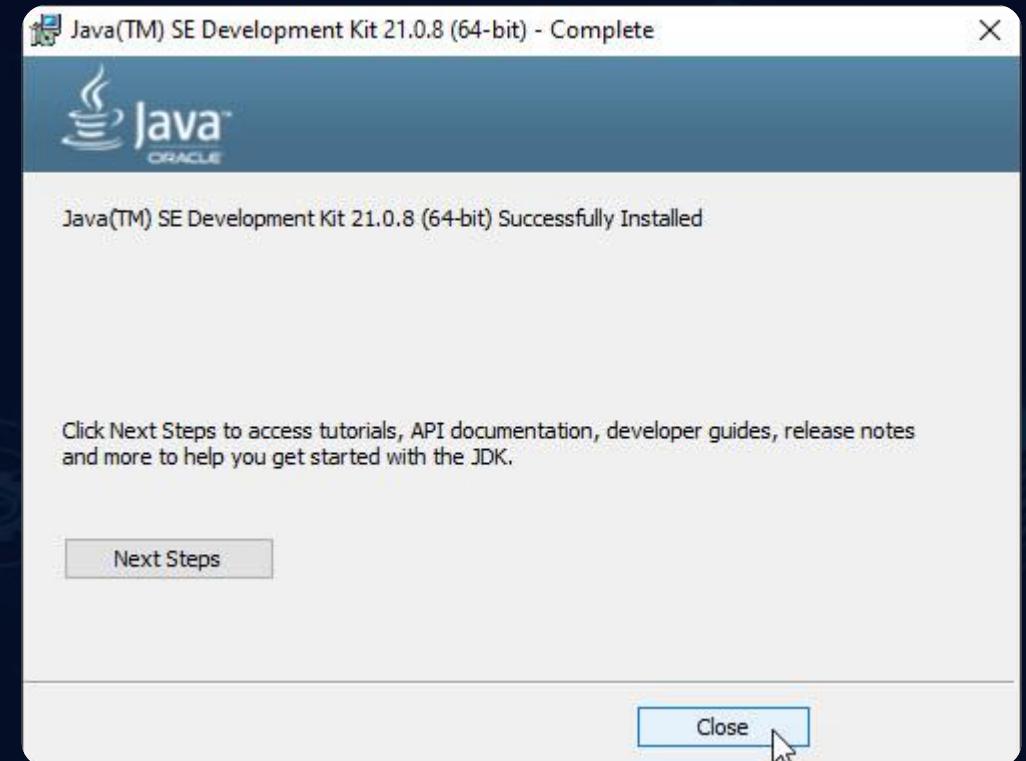
บน Windows จะติดตั้งไว้ตาม path นี้ คลิก Next



รอการติดตั้งจบแล้วเสร็จ...



รอการติดตั้ง



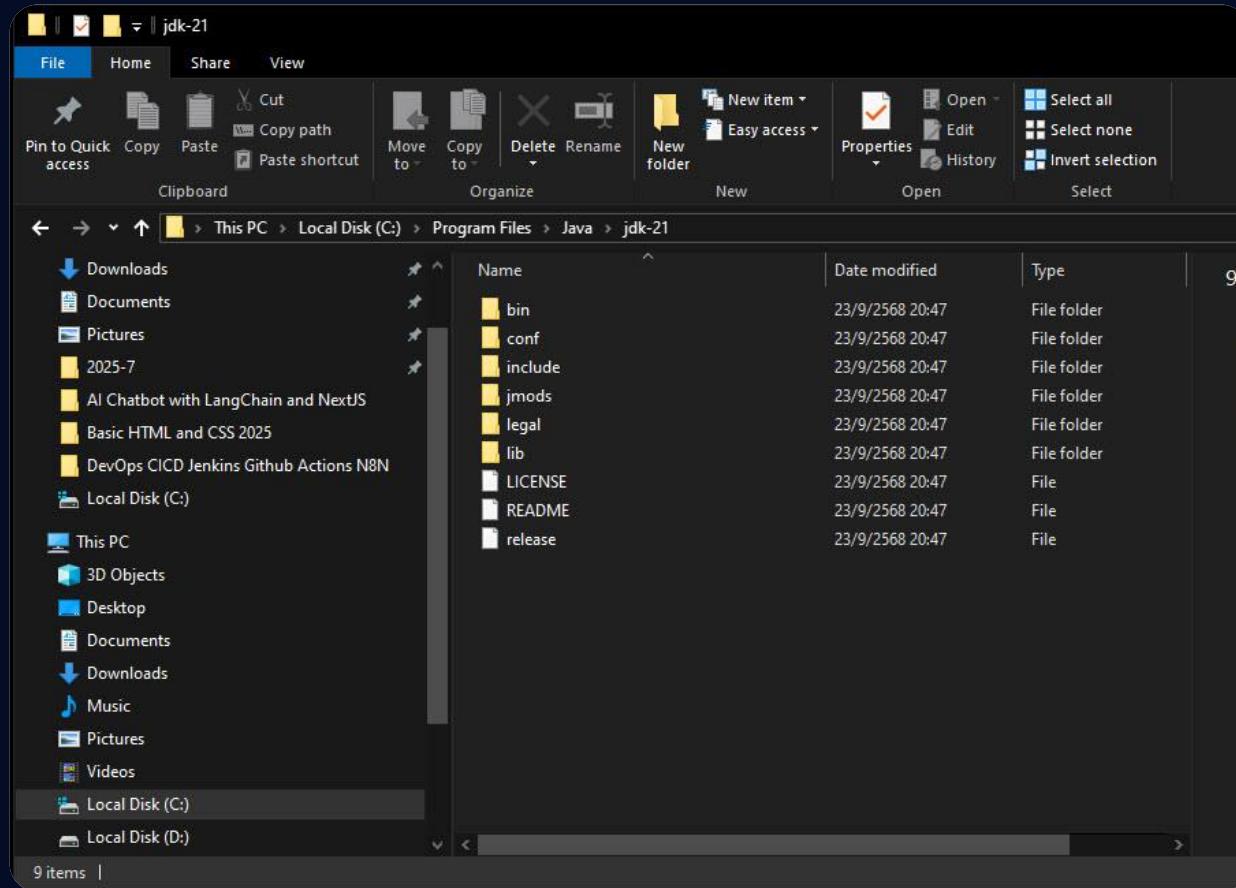
ติดตั้งเสร็จแล้ว คลิก Close



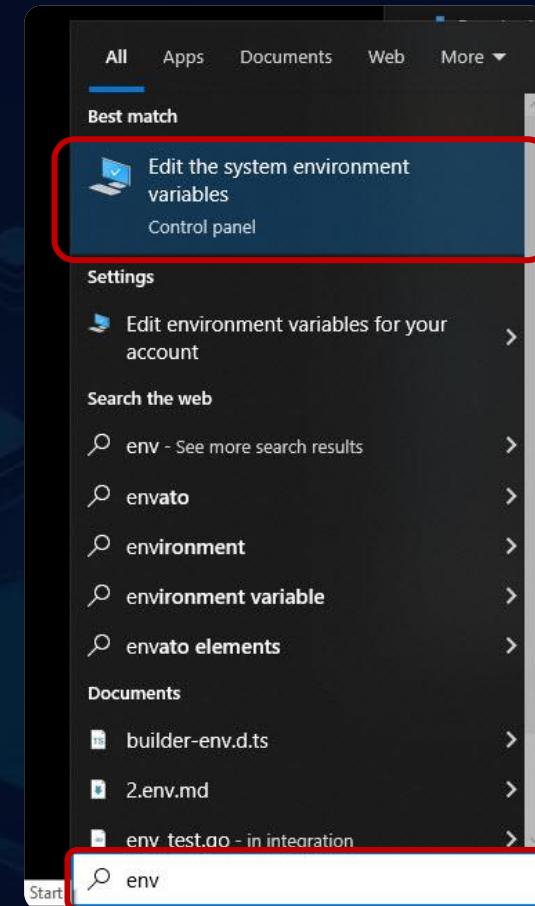
สถาบันไอทีเกี้ยนส์

www.itgenius.co.th

หลังติดตั้ง Java JDK เสร็จเรามากำหนด path ให้ Windows เรียกใช้งานได้



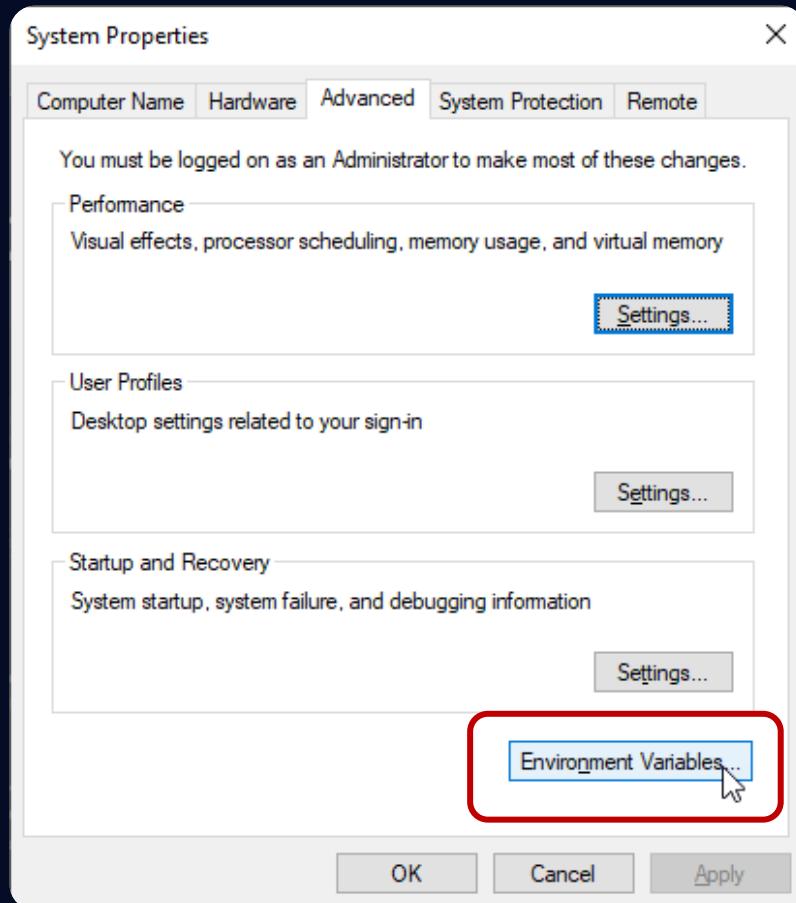
Windows เข้าไปเช็ค path ที่ C:\Program Files\Java\jdk-21
(คัดลอก path นี้ไว้)



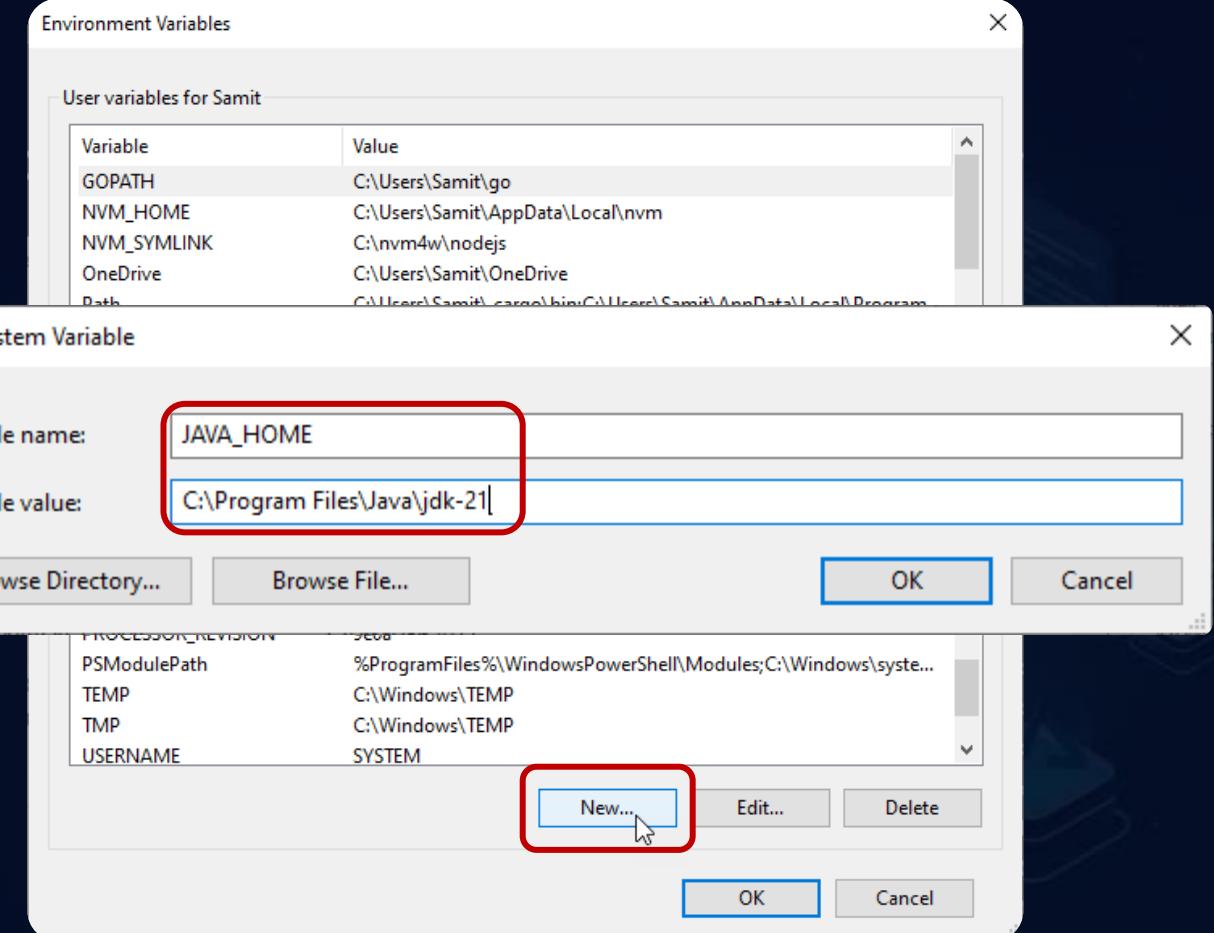
คลิก start ของ windows ค้นหา “env”
แล้วคลิกเข้าไปตั้งค่าดังรูป



หลังติดตั้ง Java JDK เสร็จเรามากำหนด path ให้ Windows เรียกใช้งานได้



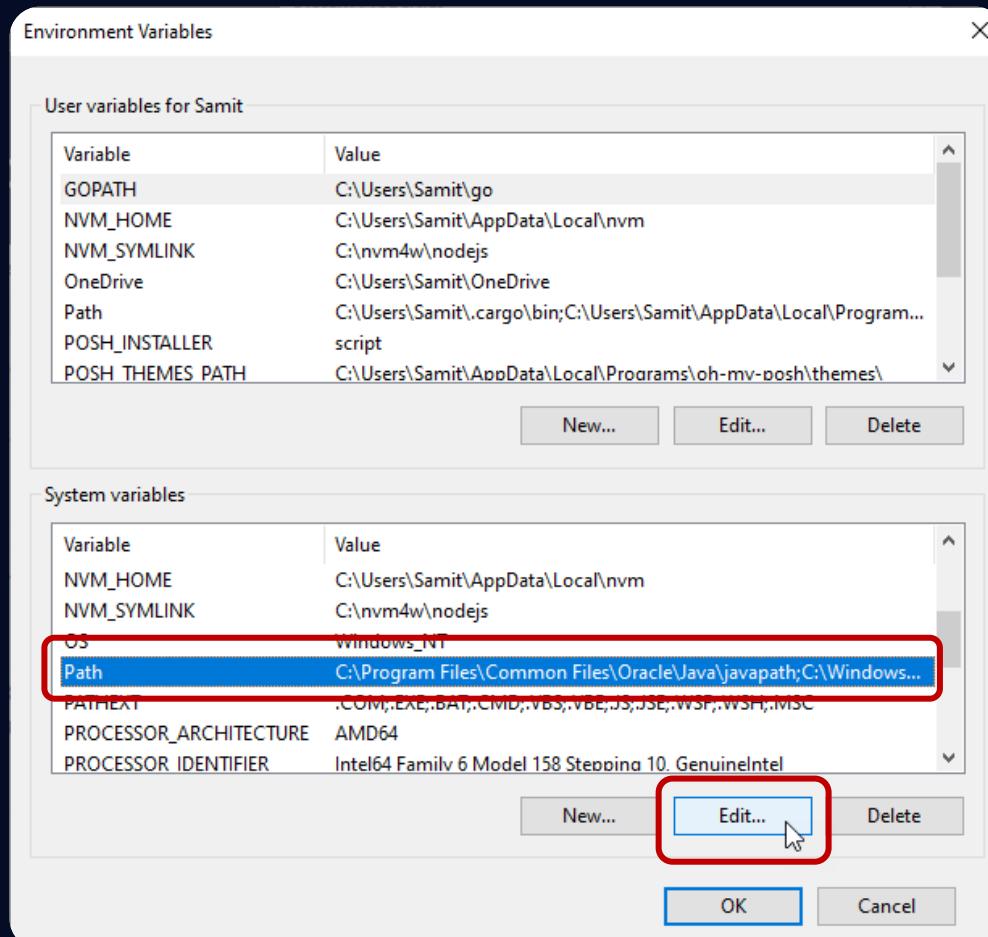
คลิกที่ Environment Variables...



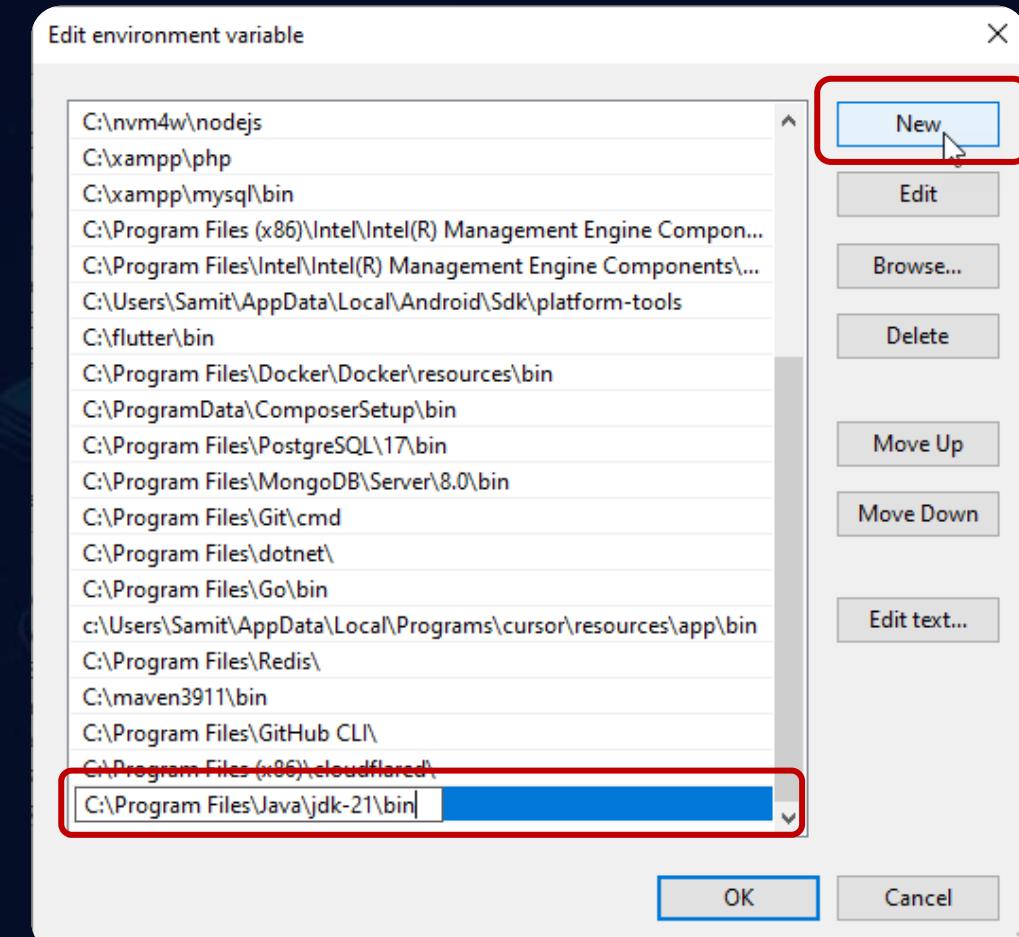
คลิกที่ปุ่ม New... ดังภาพ และกำหนดค่าดังนี้
Variable name: **JAVA_HOME**
Variable value: **C:\Program Files\Java\jdk-21**



หลังติดตั้ง Java JDK เสร็จเรามากำหนด path ให้ Windows เรียกใช้งานได้



คลิกที่ Path ดังภาพ และคลิกปุ่ม Edit...



คลิกที่ปุ่ม New... ดังภาพ และกำหนดค่าดังนี้
C:\Program Files\Java\jdk-21\bin



หลังกำหนดค่าเรียบร้อยลงแล้ว ให้เปิด Command Prompt / Terminal ขึ้นมา ตรวจสอบด้วยคำสั่งดังนี้ หากได้ผลลัพธ์ดังภาพถือว่าเรียบร้อย

java -version

set JAVA_HOME

```
Command Prompt
C:\Users\Samit>java -version
java version "21.0.8" 2025-07-15 LTS
Java(TM) SE Runtime Environment (build 21.0.8+12-LTS-250)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.8+12-LTS-250, mixed mode, sharing)
```

```
Command Prompt
C:\Users\Samit>set JAVA_HOME
JAVA_HOME=C:\Program Files\Java\jdk-21

C:\Users\Samit>
```



ส

คอร์สนี้ Jenkins ต้องการ Java JDK **17** หรือ **21** เก่าที่สุด (เก่าหรือใหม่กว่านี้ไม่รองรับ) th



3. ติดตั้ง Node JS



Download Node.JS V.22.x

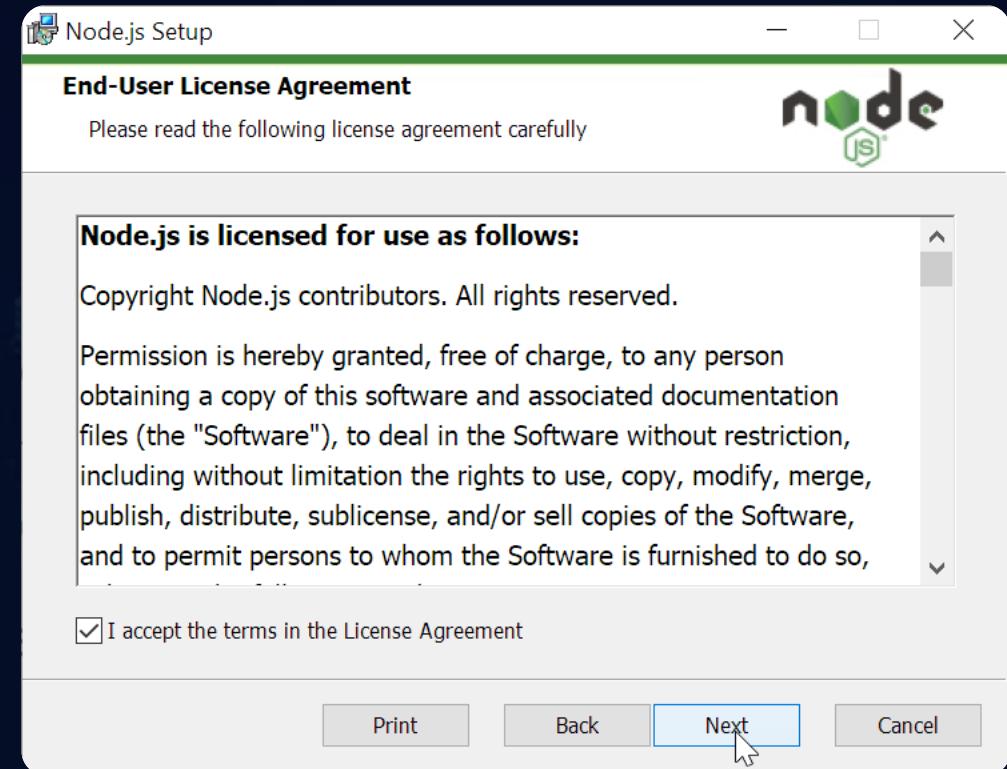
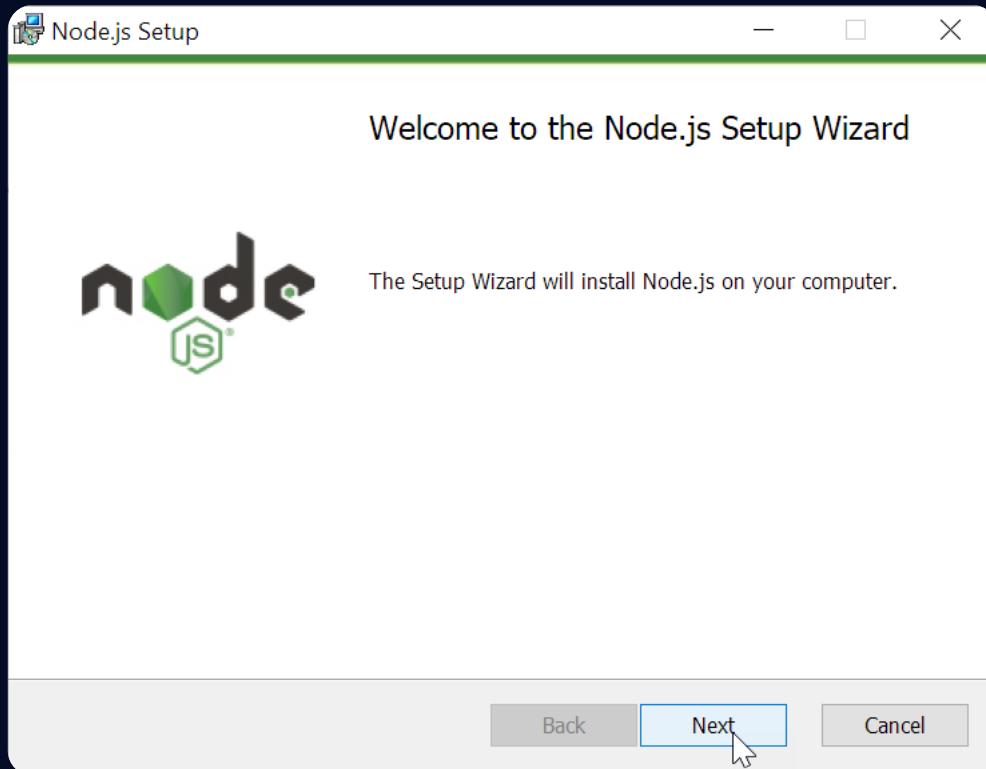
<https://nodejs.org/en/>

The screenshot shows the official Node.js download page at <https://nodejs.org/en/>. The top navigation bar includes links for Learn, About, Download (which is highlighted in green), Blog, Docs, Contribute, Certification, and a search bar. A banner at the top states "New security releases to be made available Wednesday, May 14, 2025". The main section is titled "Download Node.js®" and shows the selection "Get Node.js® v22.15.0 (LTS) for Windows using fpm with npm". Below this, a code block displays the following PowerShell commands:

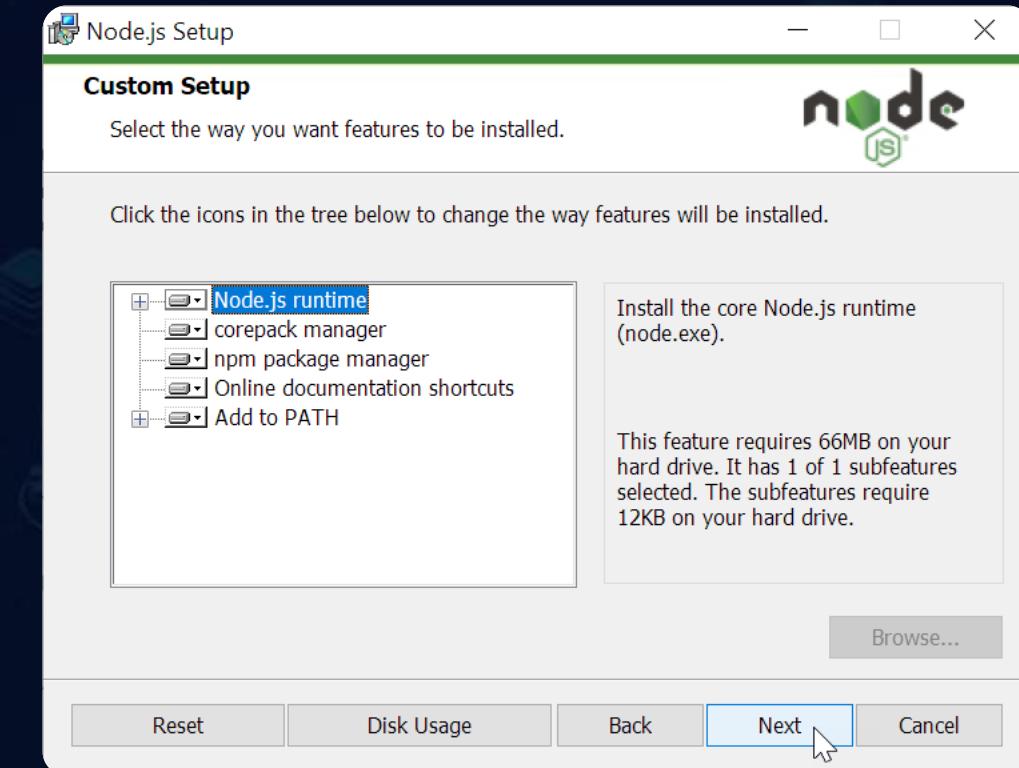
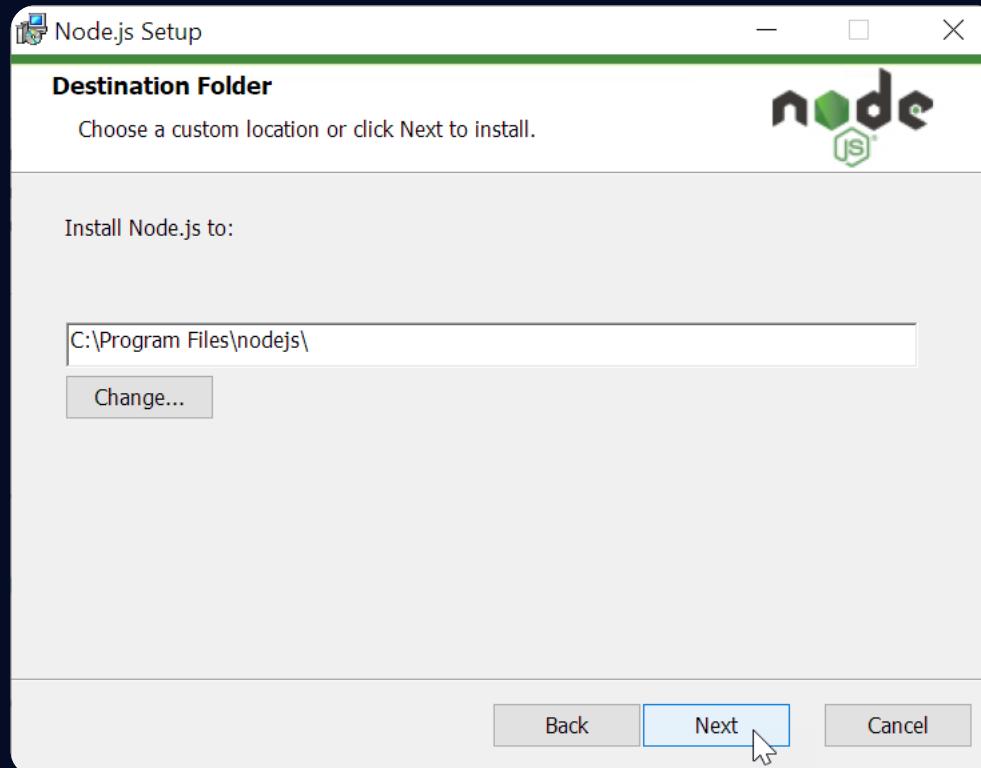
```
1 # Download and install fpm:  
2 winget install Schniz.fpm  
3  
4 # Download and install Node.js:  
5 fpm install 22  
6  
7 # Verify the Node.js version:  
8 node -v # Should print "v22.15.0".  
9  
10 # Verify npm version:  
11 npm -v # Should print "10.9.2".
```

A "Copy to clipboard" button is located below the code block. Further down, it says "Or get a prebuilt Node.js® for Windows running a x64 architecture." with two download buttons: "Windows Installer (.msi)" and "Standalone Binary (.zip)". At the bottom, there's a link to the changelog and blog post for this version.

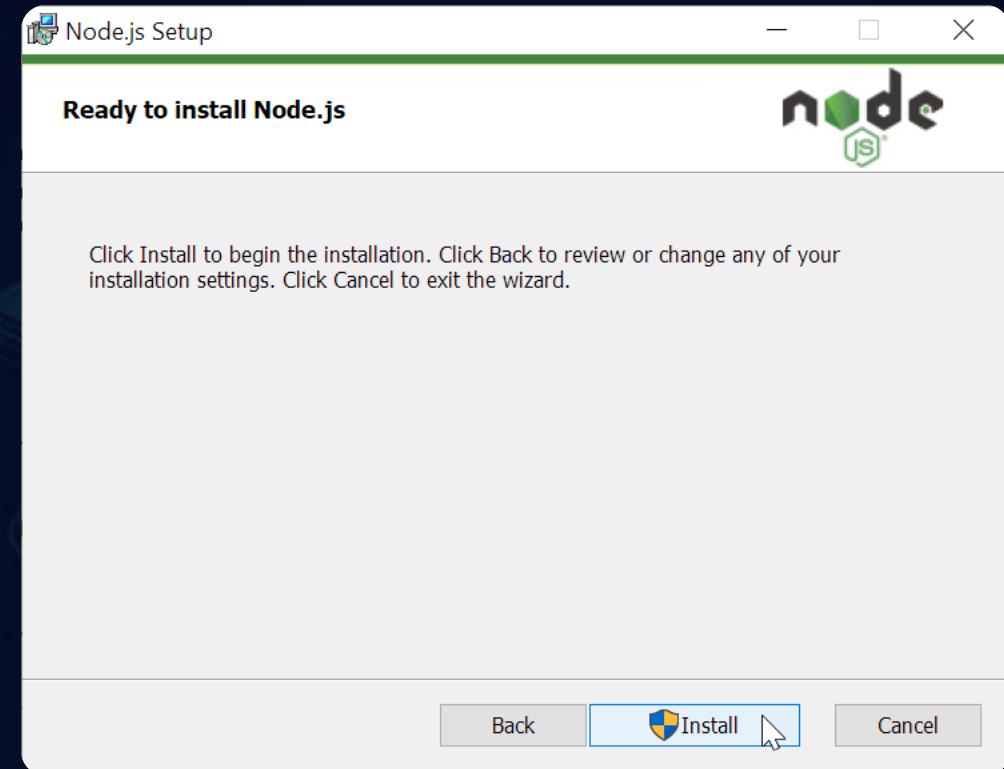
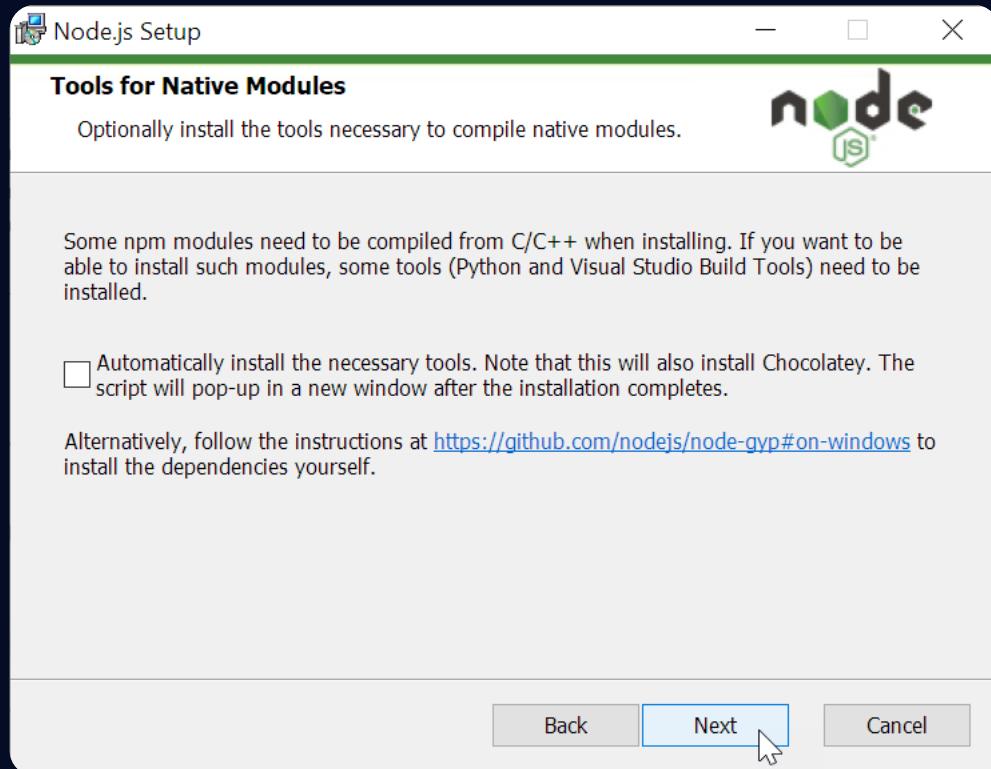
ติดตั้ง Node.js V.22.x



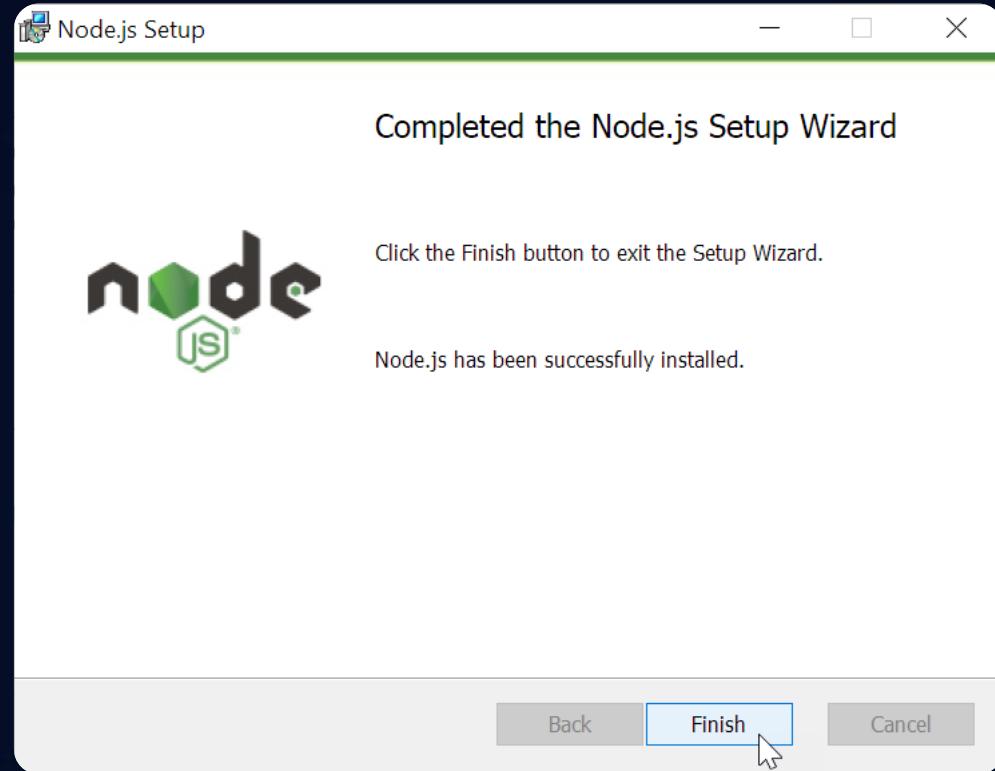
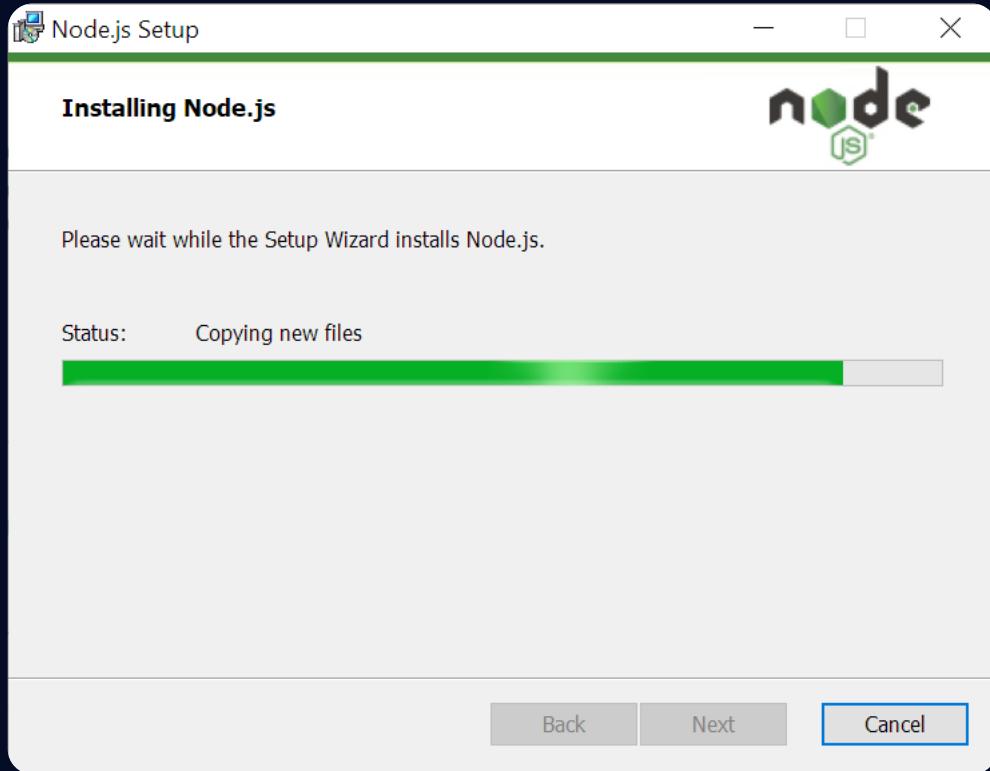
ติดตั้ง Node.js V.22.x



ติดตั้ง Node.js V.22.x



ติดตั้ง Node.js V.22.x



ກດສອບអລັງຕິດຕັ້ງເສັ່ງ

```
node -v
```

```
C:\Users\Samit>node -v  
v22.14.0
```

```
C:\Users\Samit>
```

```
npx -v
```

```
C:\Users\Samit>npx -v  
10.9.2
```

```
C:\Users\Samit>
```

```
npm -v
```

```
C:\Users\Samit>npm -v  
10.9.2
```

```
C:\Users\Samit>
```

ໝາຍເຫດ ກາວອບມອງຮັບຕັ້ງແຕ່ Node.JS 20 ຂຶ້ນໄປ ສາມາຄໃ້ Node.JS 21 , 22, 23 ມີເຊື້ອ 24 ກີດໄດ້





4. ติดตั้ง Python



The screenshot shows the Python Releases for Windows page. The 'Downloads' tab is selected, highlighted with a red box. A dropdown menu is open under 'Windows', also highlighted with a red box. The menu items include 'All releases', 'Source code', and 'Windows'. The 'Windows' item is the active choice. To the right, there's a large orange callout box with the text:

Windows เลือกดังนี้

ก่อนดาวน์โหลดติดตั้งลงเปิด cmd/terminal มาเช็คด้วยคำสั่ง

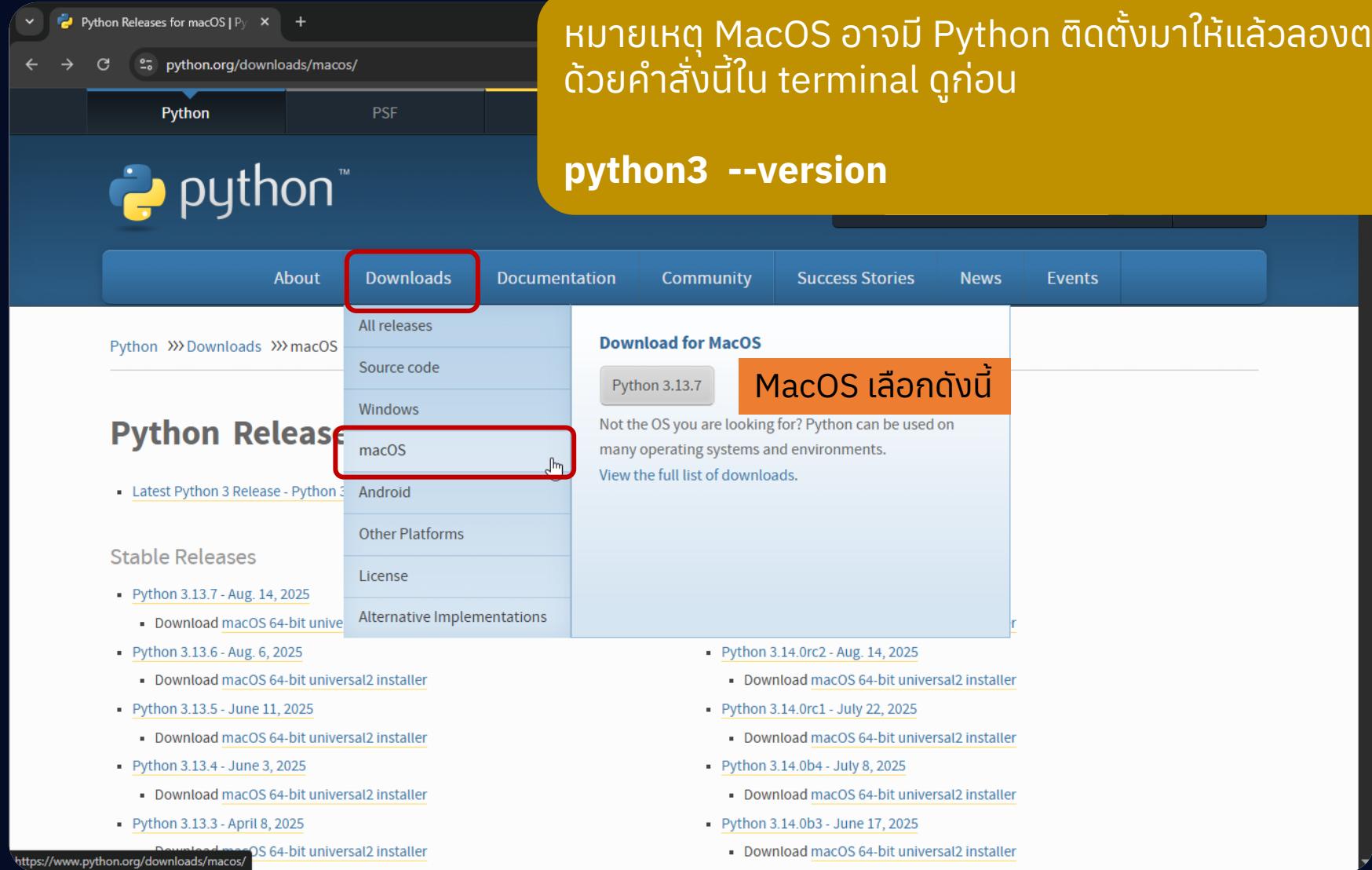
python --version

ถ้าพบ version python 3.x ขึ้นไปก็ถือว่าใช้ได้ ถ้าต่ำกว่านี้ก่อนอุปกรณ์ต้องติดตั้งใหม่ได้เลย

<https://www.python.org/ftp/python/3.13.7/python-3.13.7-amd64.exe>

ดาวน์โหลด Python 3.x ได้ที่ <https://www.python.org/downloads/windows/>

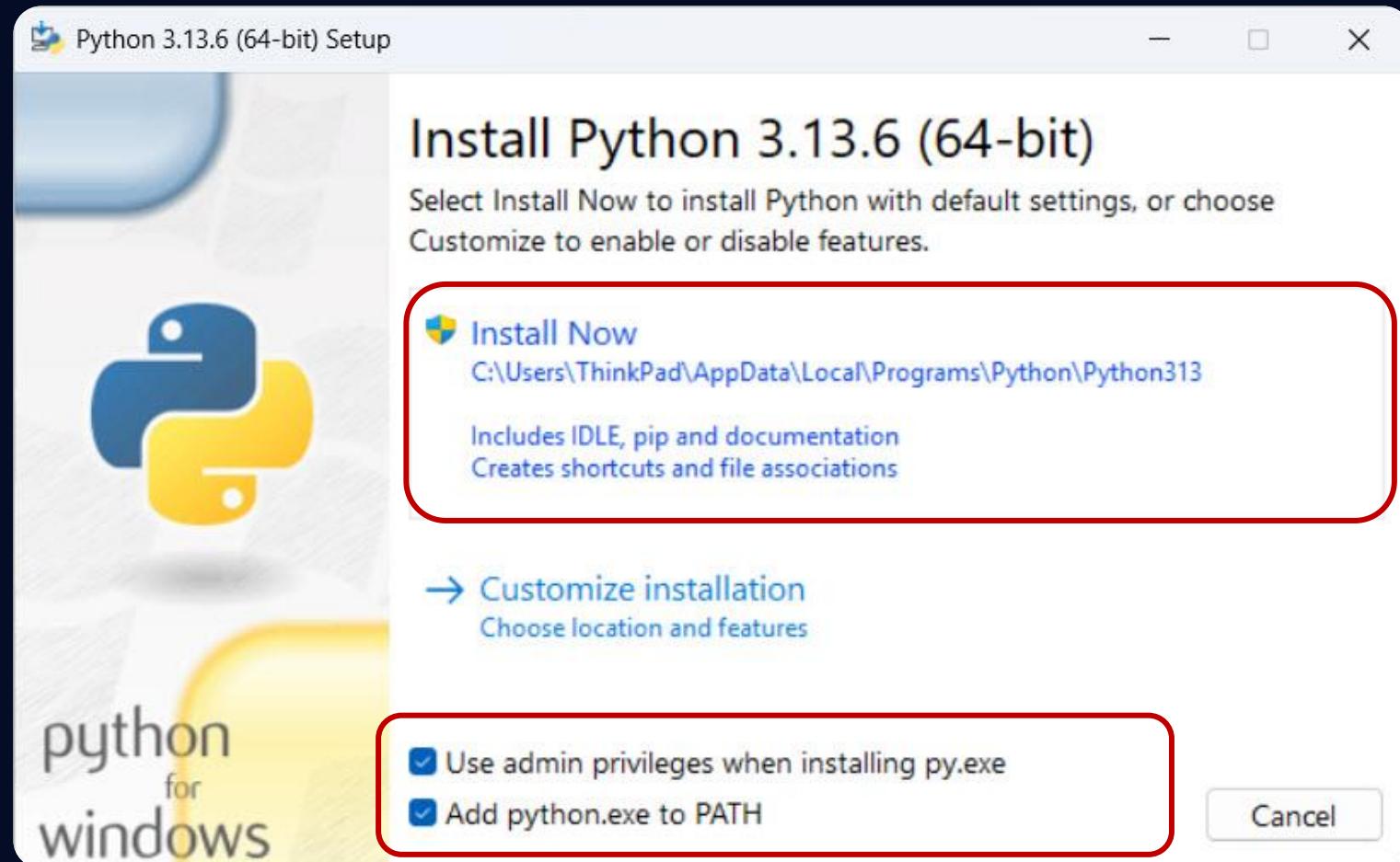




ดาวน์โหลด Python 3.x ได้ที่ <https://www.python.org/downloads/macos/>

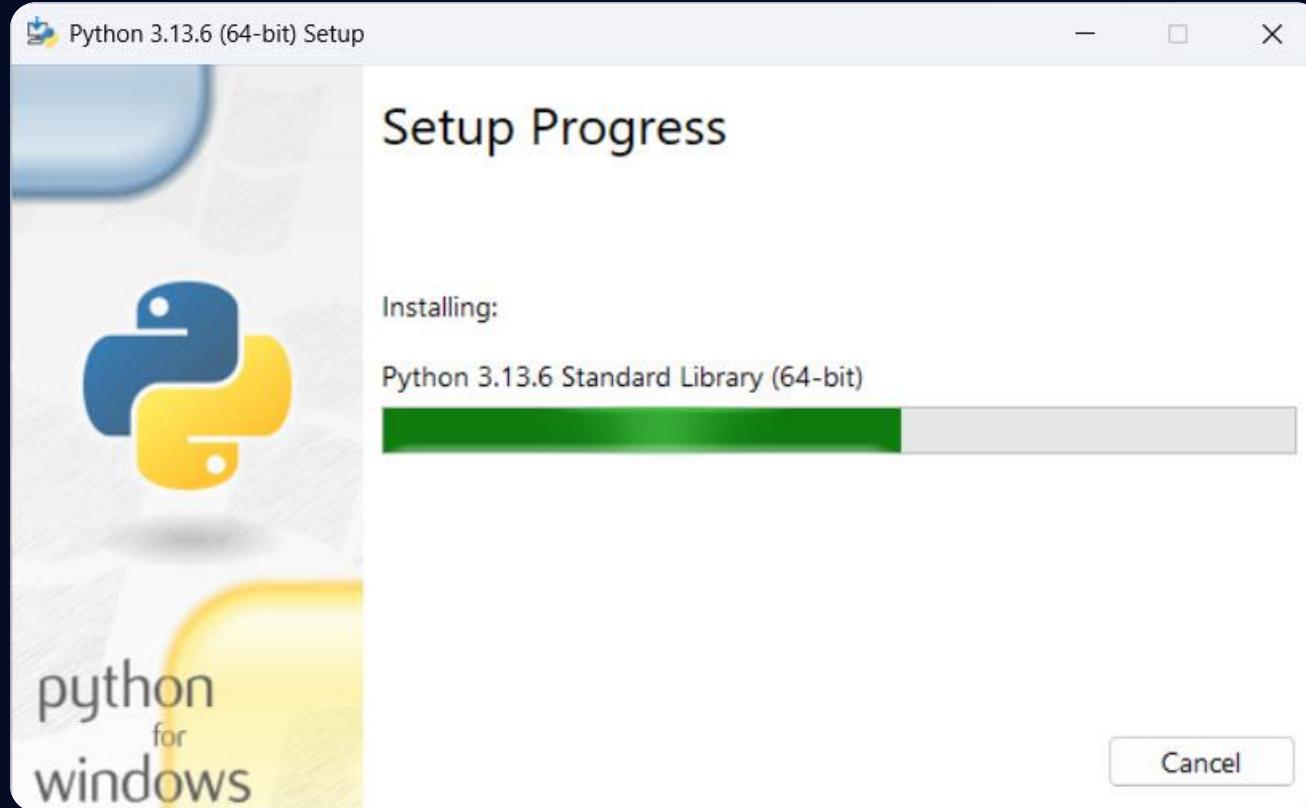


หลังดาวน์โหลดไฟล์มาแล้วทำการติดตั้ง เลือกดังภาพ



อย่าลืม! เช็คบ็อกเลือก 2 รายการด้านล่างก่อนเริ่มทำการติดตั้งนะครับ

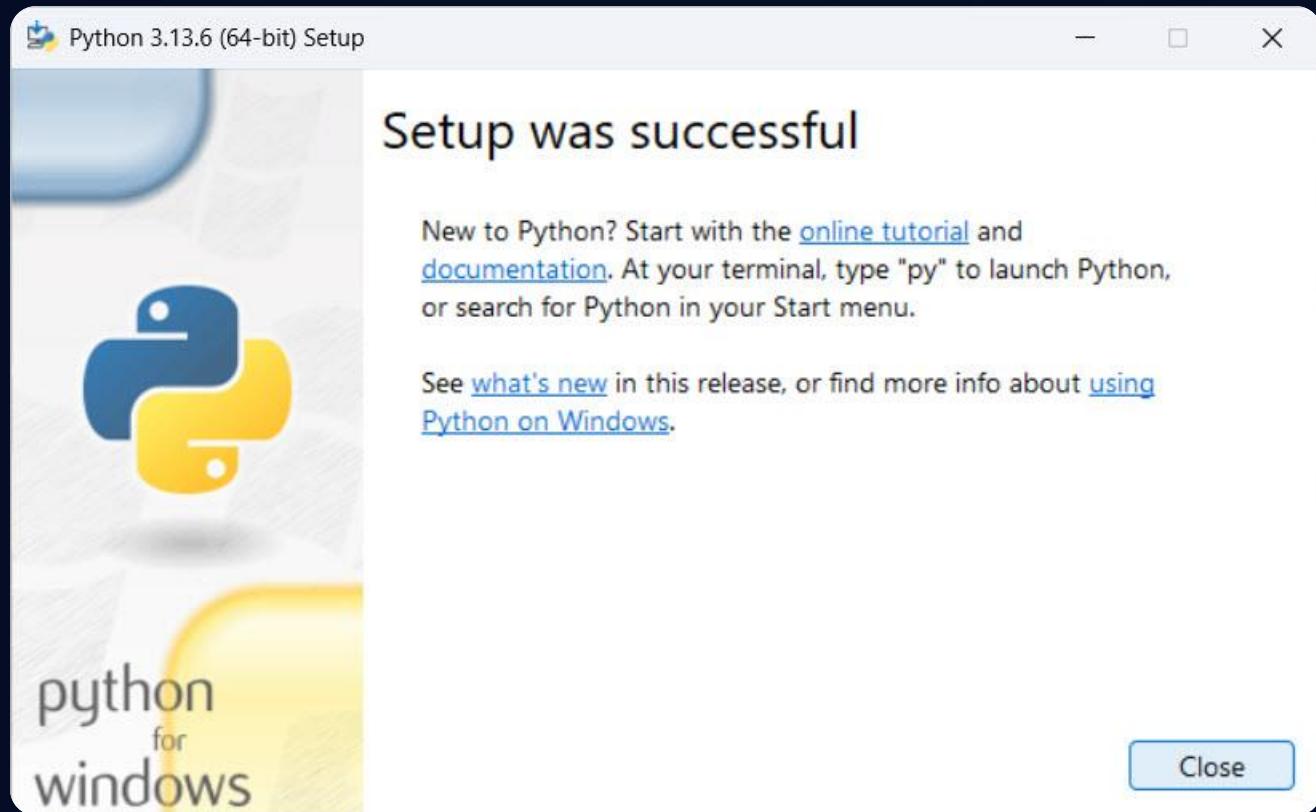
รอการติดตั้ง



รอติดตั้งแล้วเสร็จ...



ติดตั้งเสร็จเรียบร้อย



เมื่อถึงหน้านี้แสดงว่าติดตั้งเรียบร้อยแล้วคลิกปุ่ม Close



ກດສອບჩັດຕິທີ່ເສື້ຈ

Windows

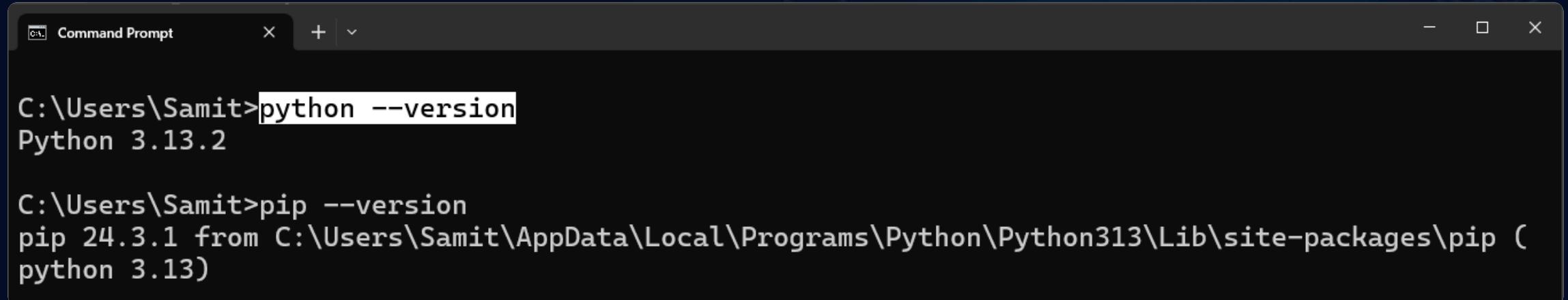
python --version

pip --version

MacOS

python3 --version

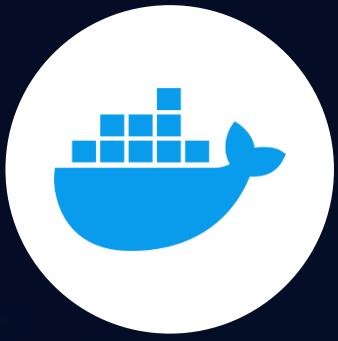
pip3 --version



```
C:\Users\Samit>python --version
Python 3.13.2

C:\Users\Samit>pip --version
pip 24.3.1 from C:\Users\Samit\AppData\Local\Programs\Python\Python313\Lib\site-packages\pip (python 3.13)
```





5. ติดตั้ง Docker Desktop



System Requirements Windows



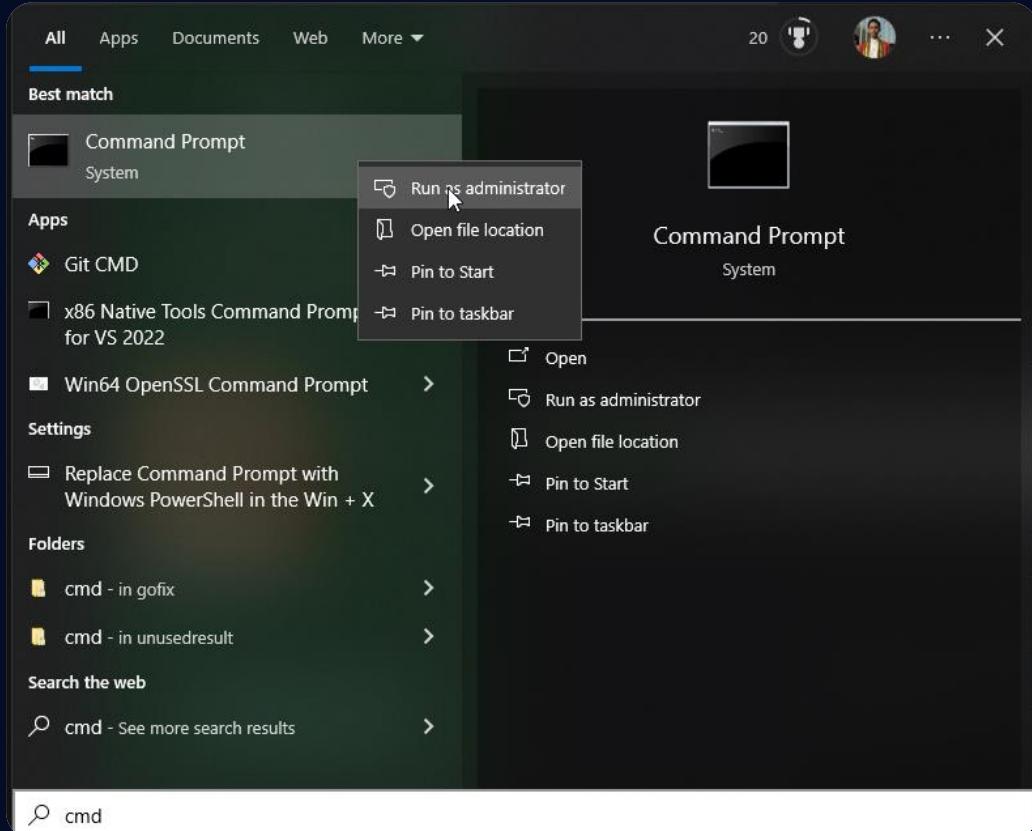
Docker Engine 20.10.9
Docker Desktop 4.10

- Windows 10 64-bit: Pro 2004 (build 19041) or higher, or Enterprise or Education 1909 (build 18363) or higher.
- For Windows 10 Home, see System requirements for WSL 2 backend.
- Hyper-V and Containers Windows features must be enabled.
- The following hardware prerequisites are required to successfully run Client Hyper-V on Windows 10:
- 64 bit processor with Second Level Address Translation (SLAT)
- 4GB system RAM
- BIOS-level hardware virtualization support must be enabled in the BIOS settings

<https://docs.docker.com/desktop/windows/install/>



สำหรับ Windows 10/11 แนะนำให้ติดตั้ง wsl ก่อน



คลิก start พิมพ์ cmd คลิกเปิดด้วย Run as Administrator

```
Administrator: C:\Windows\System32\cmd.exe - wsl --install
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

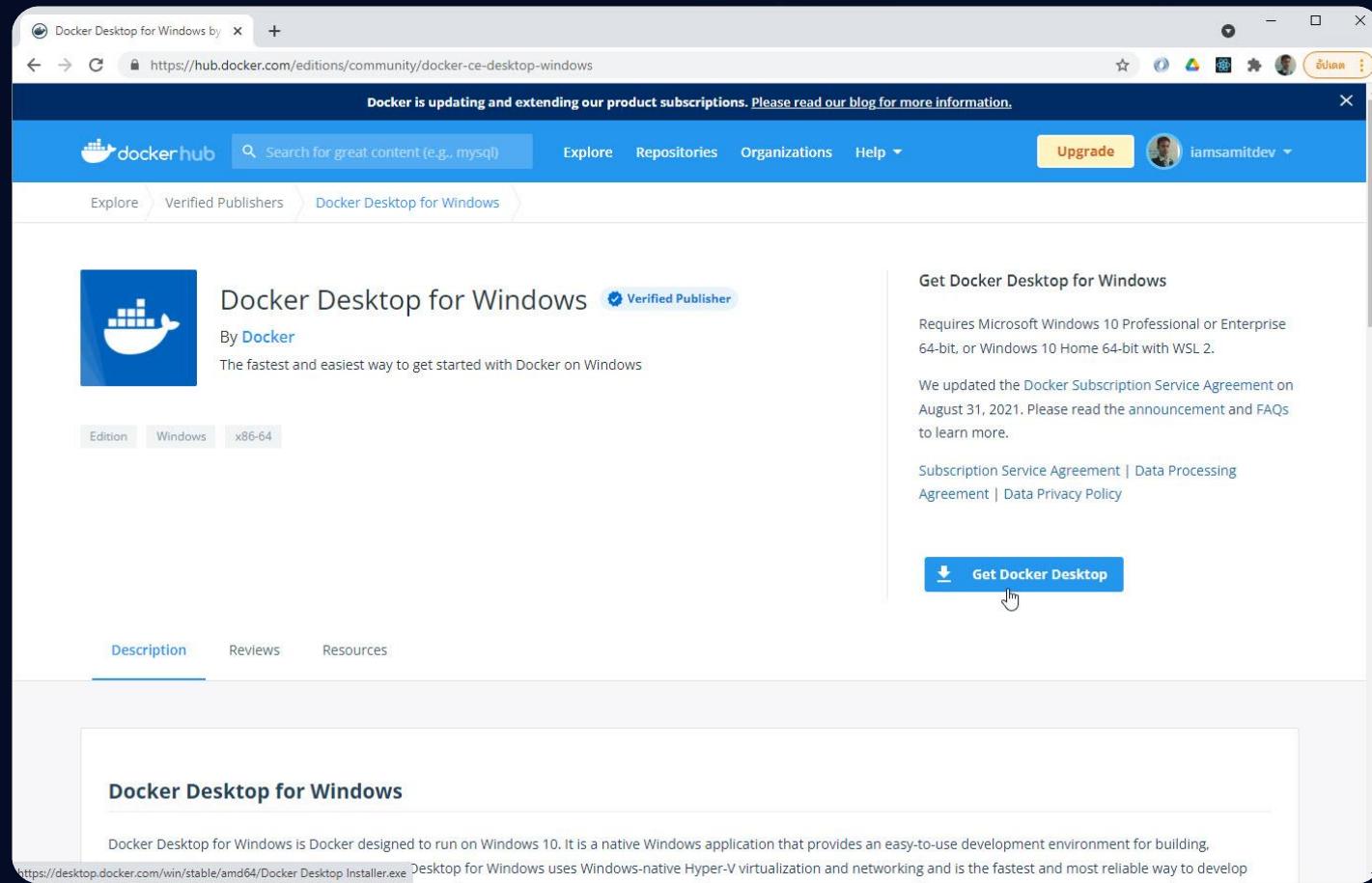
C:\WINDOWS\system32>wsl --install
Installing: Virtual Machine Platform
Virtual Machine Platform has been installed.
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
Downloading: WSL Kernel
[=====] 27.2%
```

พิมพ์คำสั่งติดตั้ง wsl ด้วย **wsl --install**



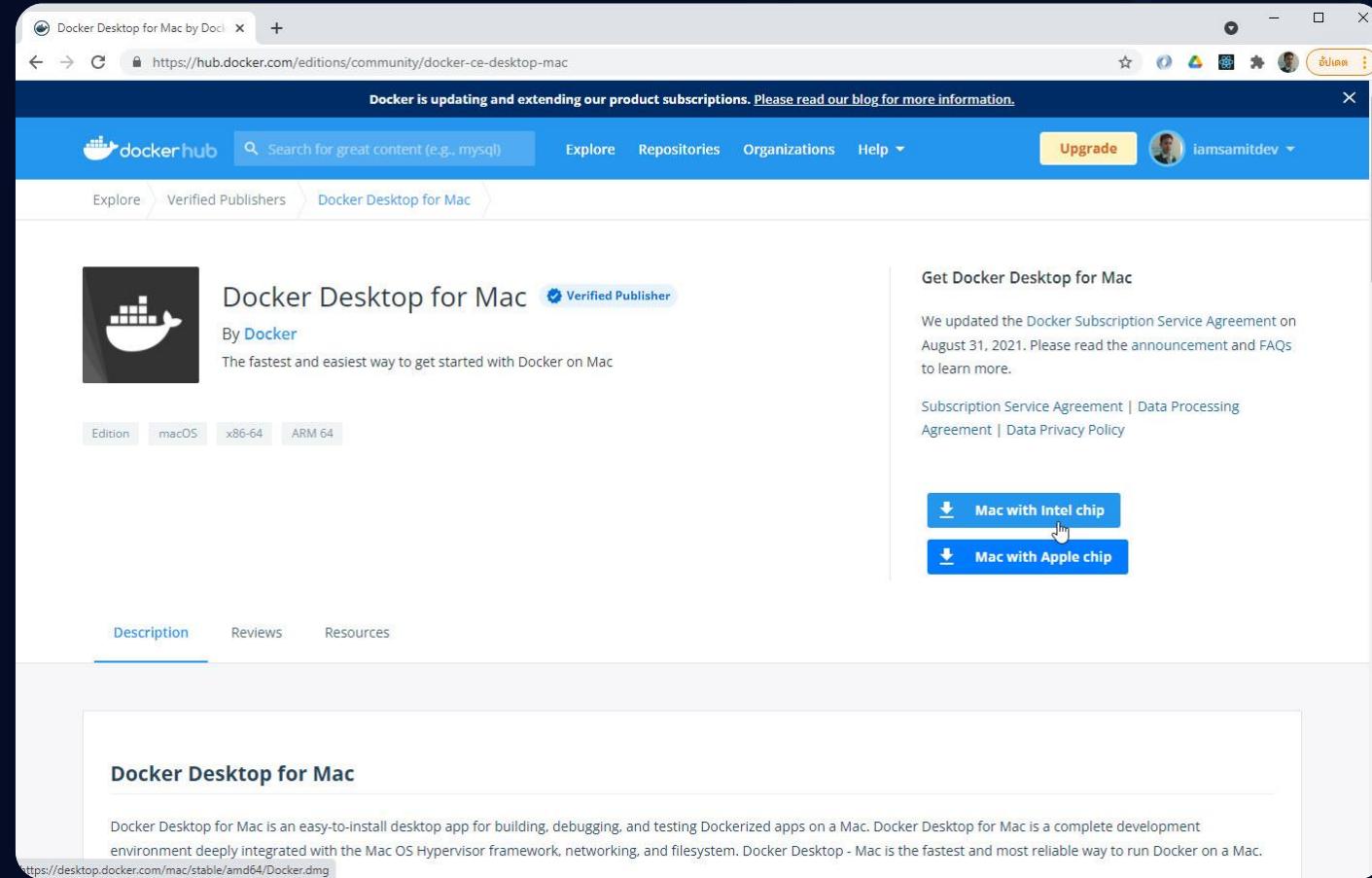
Download Docker Desktop for Windows

<https://hub.docker.com/editions/community/docker-ce-desktop-windows>

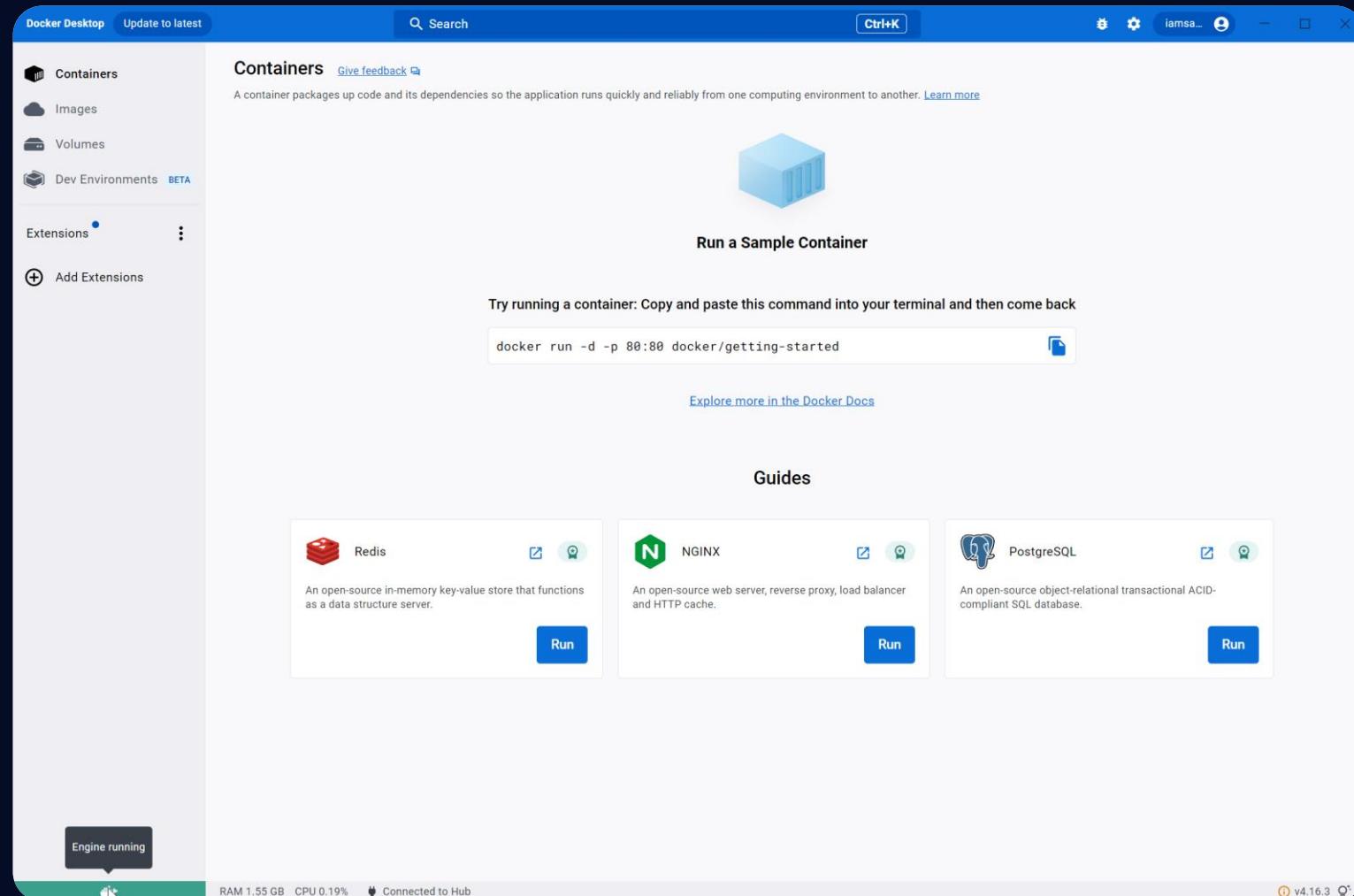


Download Docker Desktop for MacOS

<https://hub.docker.com/editions/community/docker-ce-desktop-mac>



ตัวอย่าง Docker Desktop หลังจากติดตั้งเรียบร้อยแล้ว





6. ติดตั้ง Git

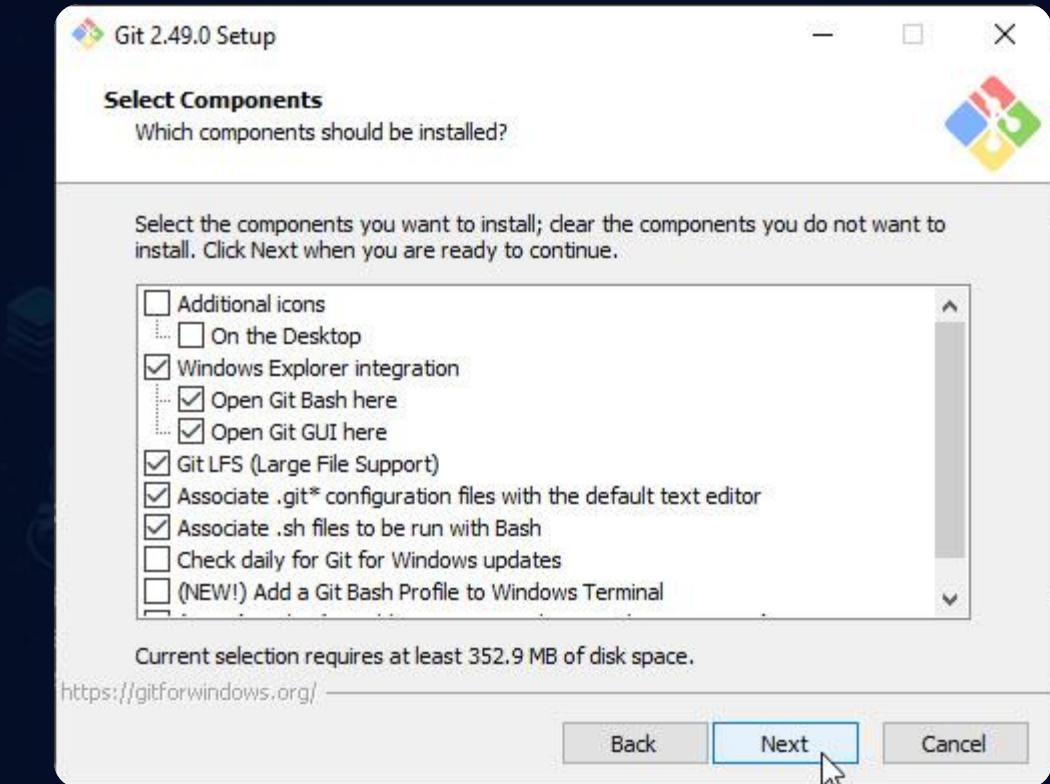
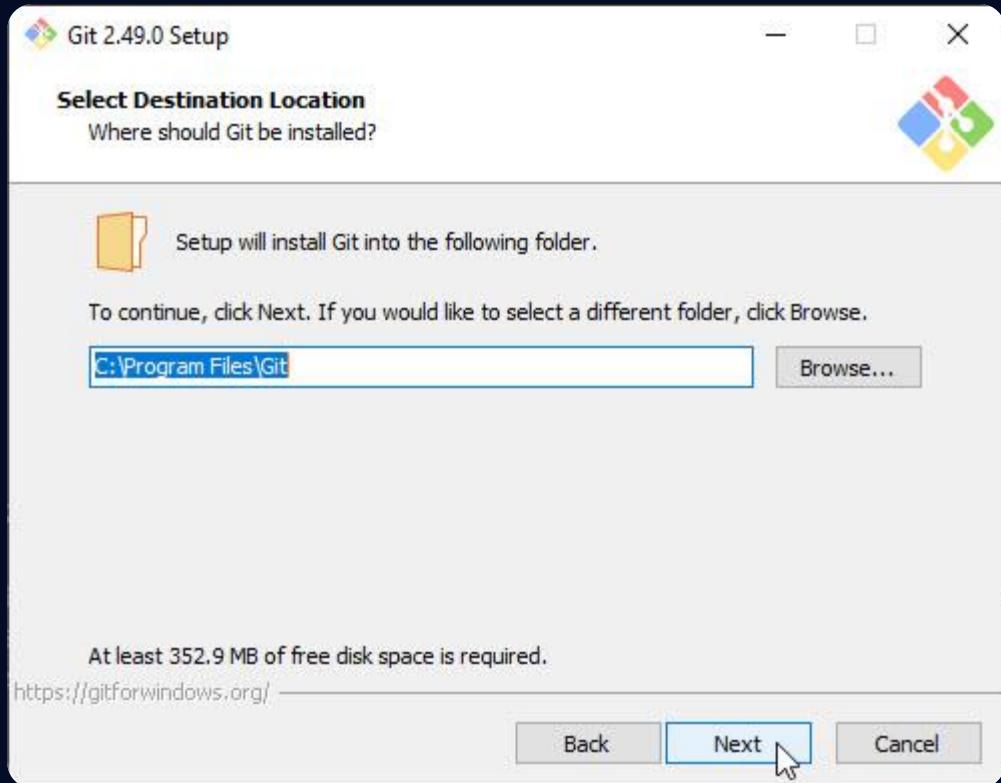


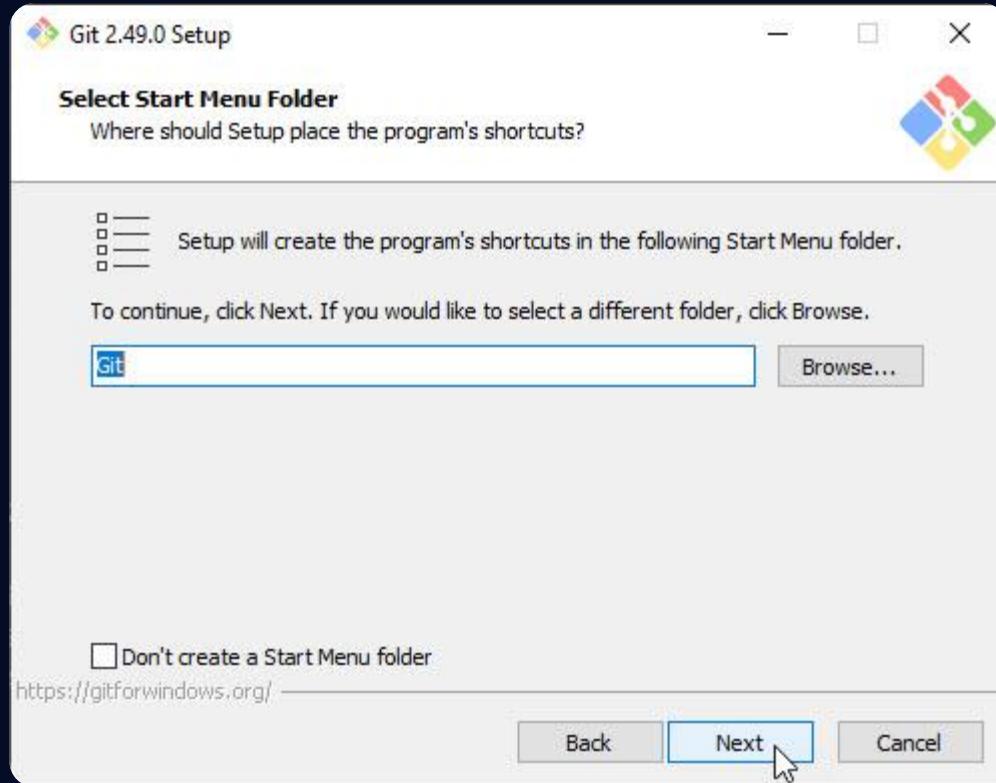
ดาวน์โหลดไฟล์ติดตั้ง Git ได้ที่ <https://git-scm.com/>

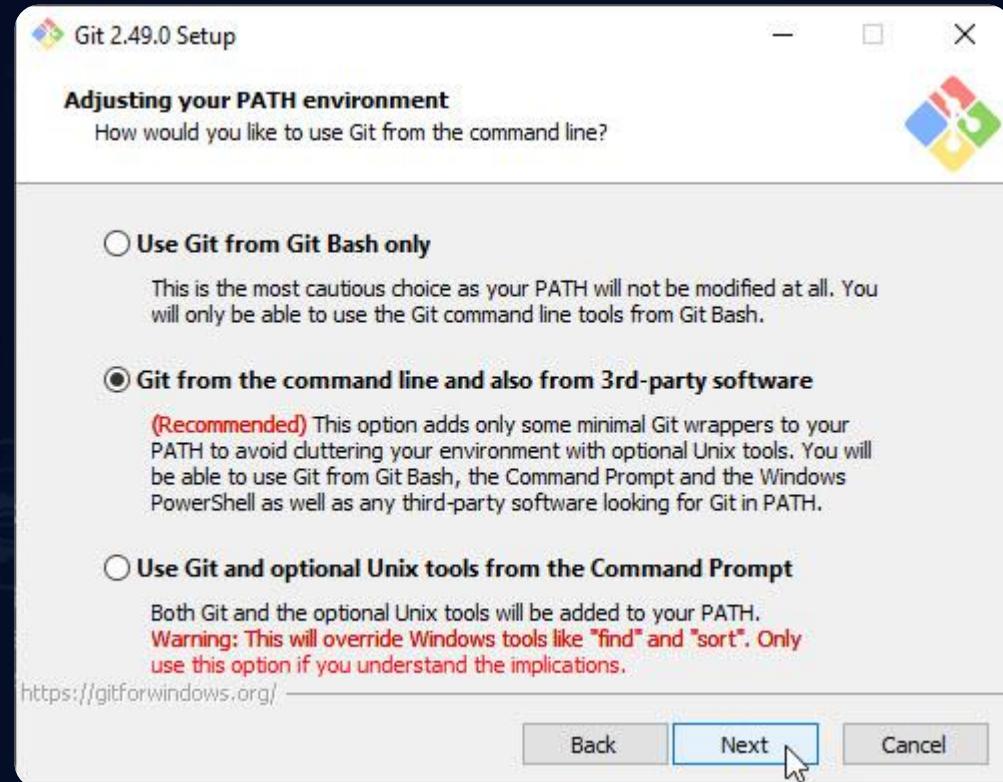
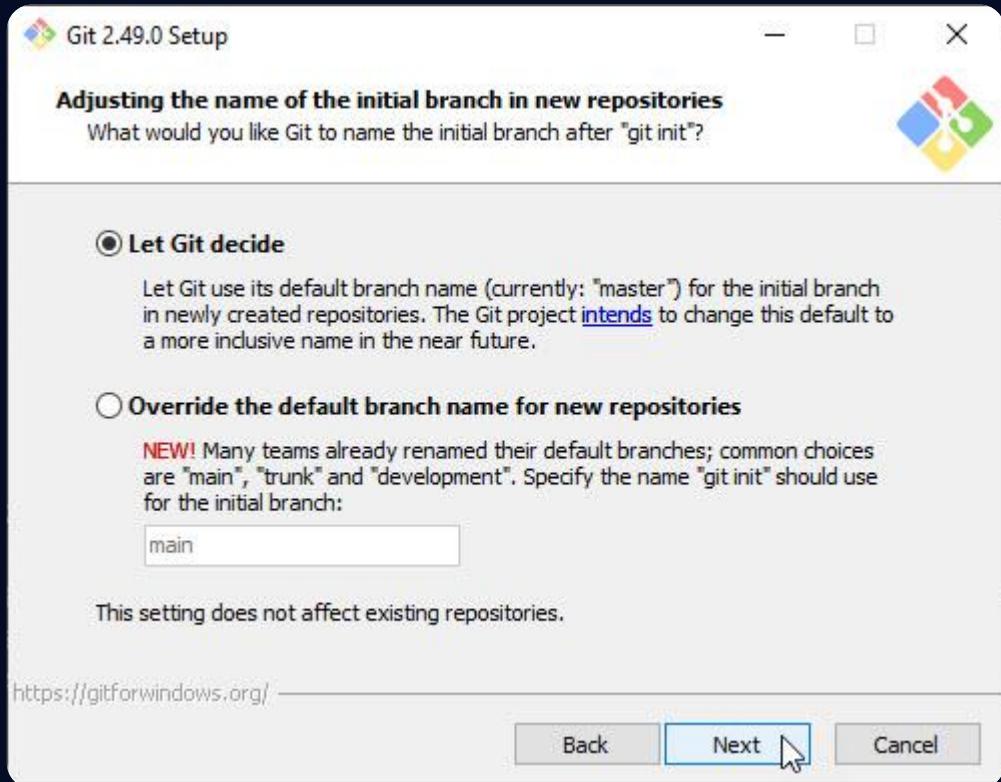
The screenshot shows the official website for Git (<https://git-scm.com/>). The page features a dark background with a central diagram of five computer stacks connected by red and blue lines, representing a distributed version control system. At the top, there's a search bar and a 'Relaunch to update' button. The main text highlights Git as a free and open-source distributed version control system designed for efficiency and speed. Below this, another paragraph describes Git's ease of learning, fast performance, and unique features like cheap local branching. On the left side, there are four main navigation links: 'About' (with a gear icon), 'Documentation' (with a book icon), 'Downloads' (with a download arrow icon), and 'Community' (with a speech bubble icon). A small image of the 'Pro Git' book is also visible. On the right side, there's a large monitor icon displaying the latest source release version '2.49.0' and a 'Download for Windows' button. Below the monitor, there are links for 'Windows GUIs', 'Tarballs', 'Mac Build', and 'Source Code'.

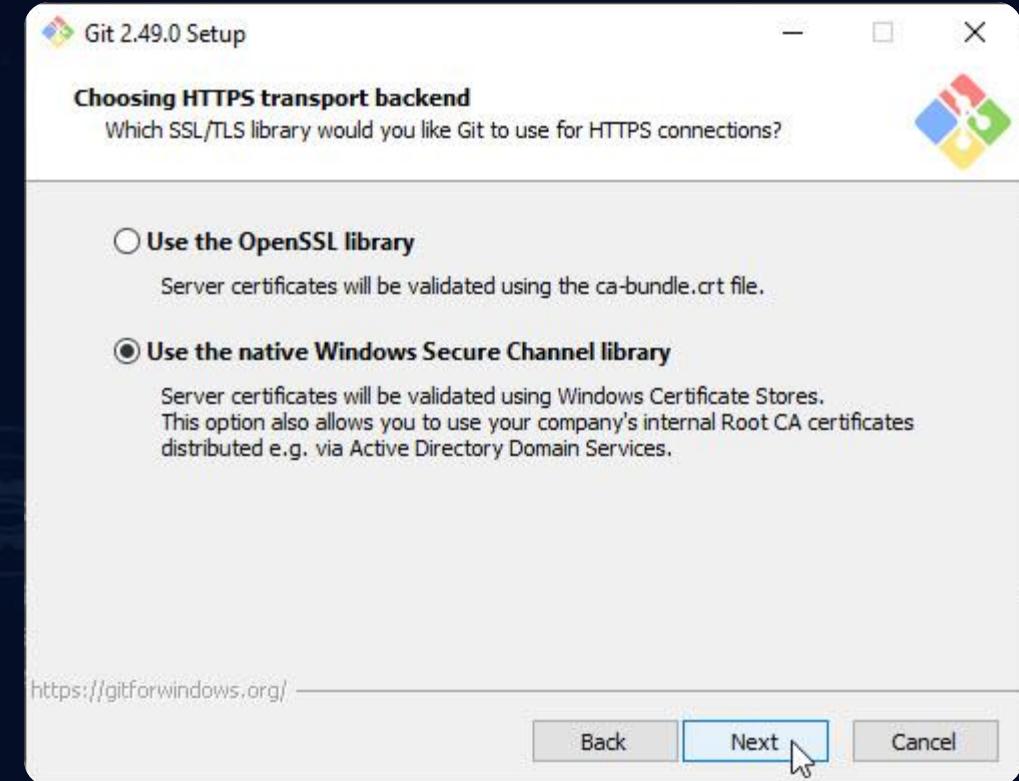
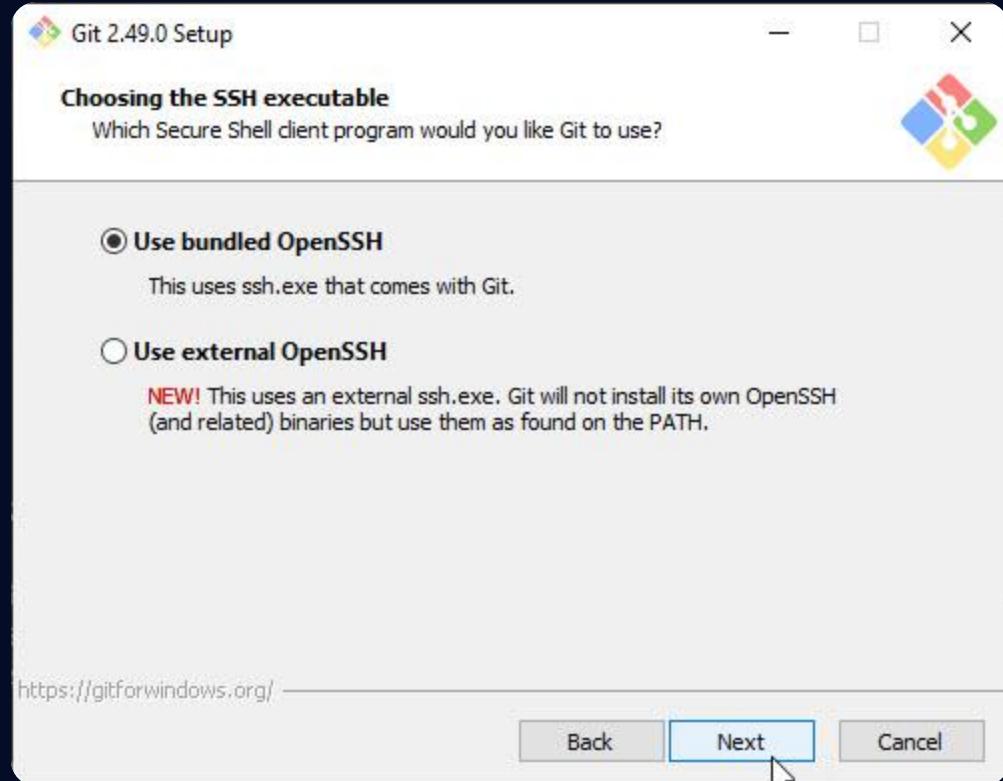


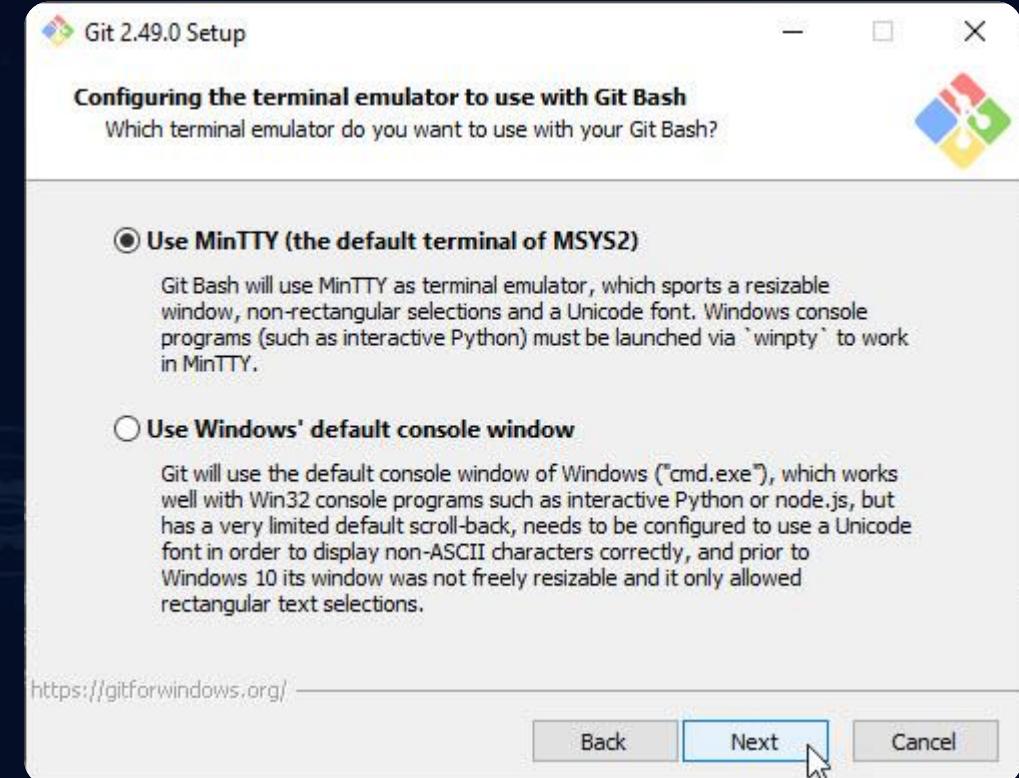
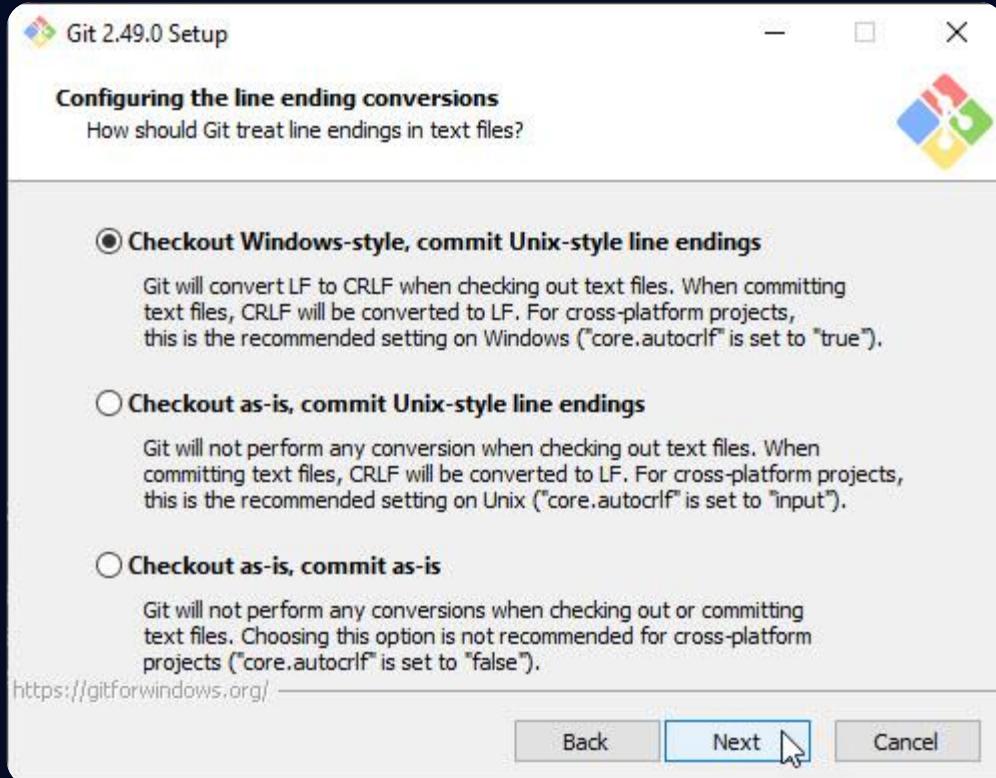


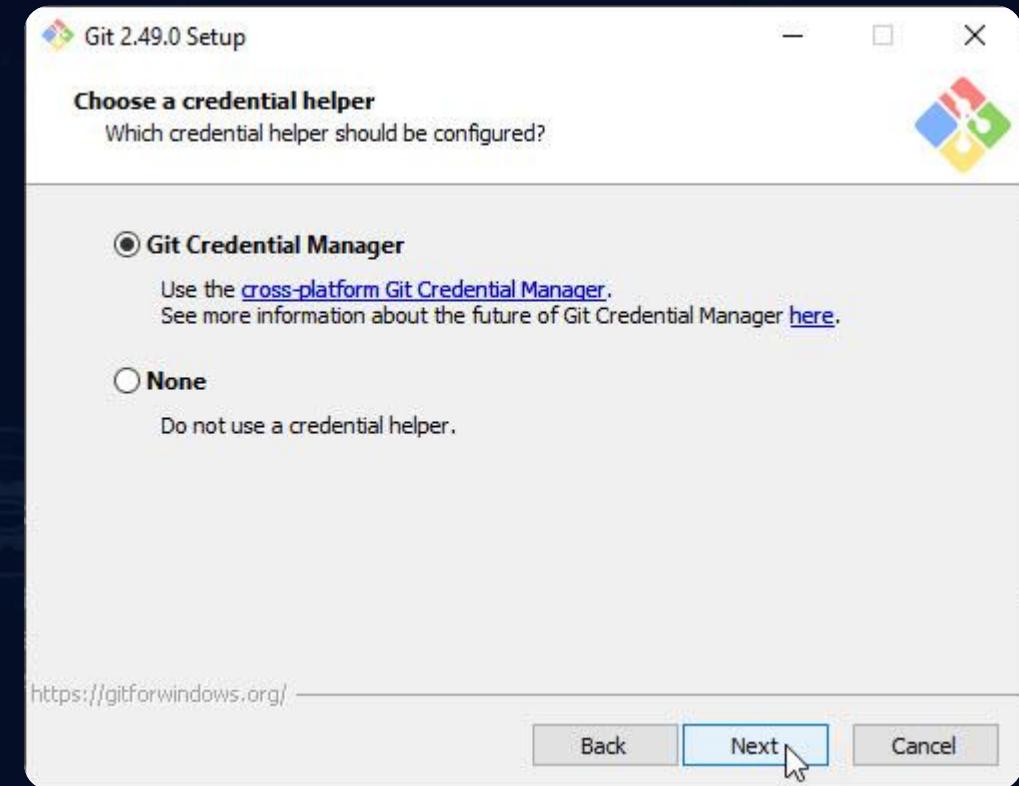
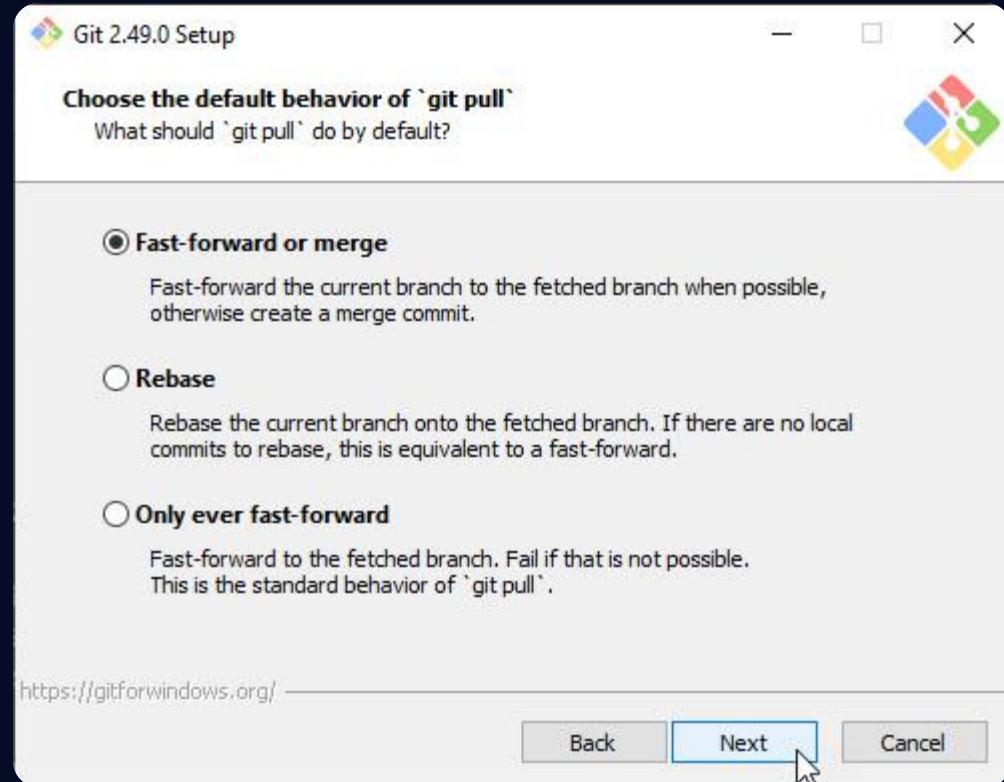


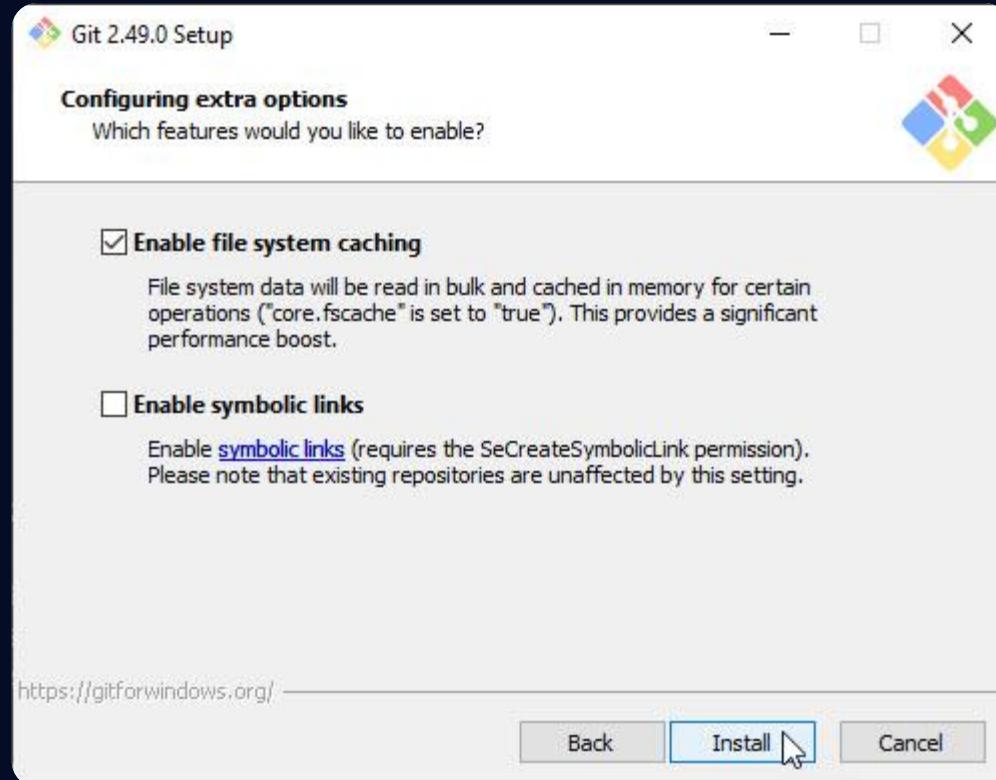








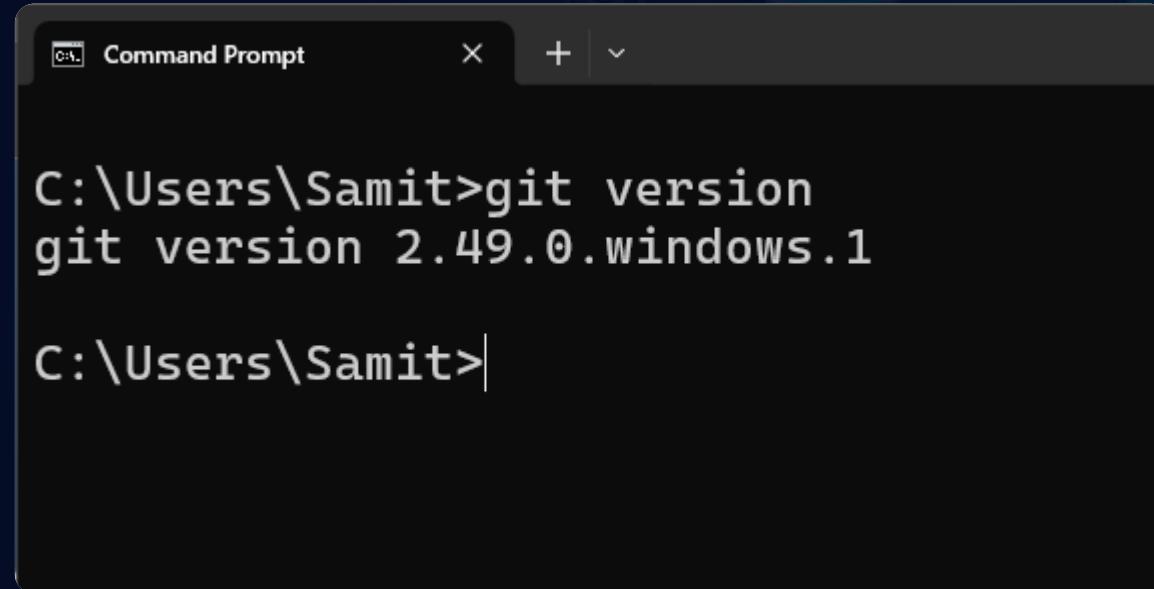




หลังติดตั้งสำเร็จกดสอบด้วยคำสั่ง

git version

หากพบเวอร์ชันดังภาพ ถือว่าติดตั้งเรียบร้อยพร้อมใช้งาน



```
Command Prompt
C:\Users\Samit>git version
git version 2.49.0.windows.1
C:\Users\Samit>
```



การตรวจสอบความเรียบร้อยของเครื่องมือที่ติดตั้งบน Windows / Mac OS / Linux

เปิด Command Prompt บน Windows หรือ Terminal บน Mac ขึ้นมาป้อนคำสั่งดังนี้

Visual Studio Code

```
code --version
```

Node JS

```
node -v  
npm -v  
npx -v
```

Java

```
java -version
```

Python (windows)

```
python --version  
pip --version
```

Python (macos)

```
python3 --version  
pip3 --version
```

Docker

```
docker --version
```

Git

```
git version
```





พื้นฐานการใช้งาน Git

- เริ่มต้นใช้งาน Git ในโปรเจกต์
- การตั้งค่า , การ clone , คำสั่งพื้นฐาน



Git First time setup

กำหนดข้อมูลผู้ใช้

```
git config --global user.name "John Doe"  
git config --global user.email johndoe@example.com
```

คำสั่งเช็คข้อมูลที่กำหนดไว้

```
git config --list  
git config --global --list
```



Set Default branch “main”

ตั้งค่าให้ branch หลักชื่อ “main”

```
git config --global init.defaultBranch main
```

ตรวจสอบหลังตั้งค่า

```
git config --list
```



Set Default branch “main”

ตรวจสอบหลังตั้งค่า

```
pwsh in Samit
Samit ➤ 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=schannel
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppa
init.defaultbranch=master
user.name=Samit Koyom
user.email=samit90daytalk@hotmail.com
core.page=cat
```

git config --list --show-origin

```
Samit ➤ git config --list --show-origin
file:C:/Program Files/Git/etc/gitconfig diff.astextplain.textconv=astextplain
file:C:/Program Files/Git/etc/gitconfig filter.lfs.clean=git-lfs clean -- %f
file:C:/Program Files/Git/etc/gitconfig filter.lfs.smudge=git-lfs smudge -- %f
file:C:/Program Files/Git/etc/gitconfig filter.lfs.process=git-lfs filter-process
file:C:/Program Files/Git/etc/gitconfig filter.lfs.required=true
file:C:/Program Files/Git/etc/gitconfig http.sslbackend=schannel
file:C:/Program Files/Git/etc/gitconfig core.autocrlf=true
file:C:/Program Files/Git/etc/gitconfig core.fscache=true
file:C:/Program Files/Git/etc/gitconfig core.symlinks=false
file:C:/Program Files/Git/etc/gitconfig pull.rebase=false
file:C:/Program Files/Git/etc/gitconfig credential.helper=manager
file:C:/Program Files/Git/etc/gitconfig credential.https://dev.azure.com.usehttppath=true
file:C:/Program Files/Git/etc/gitconfig init.defaultbranch=master
file:C:/Users/Samit/.gitconfig user.name=Samit Koyom
file:C:/Users/Samit/.gitconfig user.email=samit90daytalk@hotmail.com
file:C:/Users/Samit/.gitconfig core.page=cat
file:C:/Users/Samit/.gitconfig init.defaultbranch=main
```

Git มีการตั้งค่าได้ 3 ระดับ ซึ่งจะถูกอ่านเรียงต่อกันตามลำดับนี้ครับ:

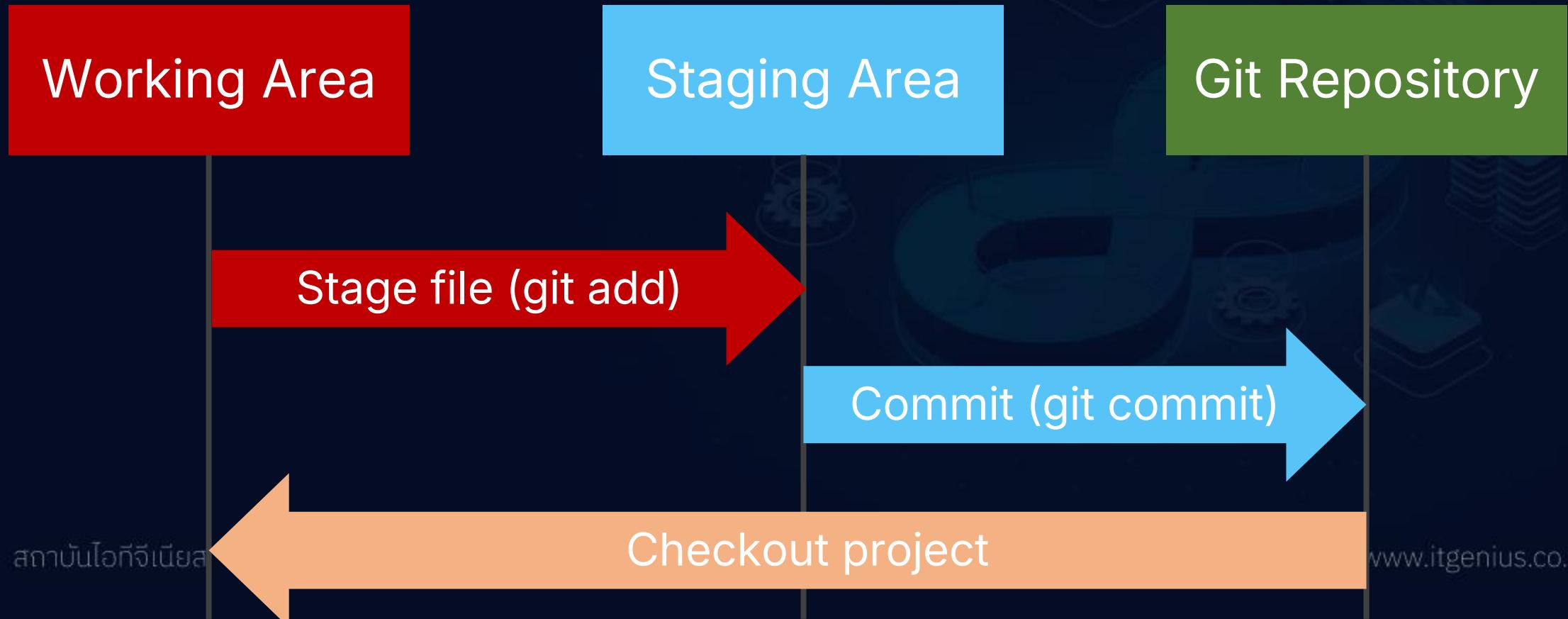
1. **System:** การตั้งค่าสำหรับทุก User บนเครื่องคอมพิวเตอร์นี้
2. **Global:** การตั้งค่าสำหรับ User ของคุณคนเดียว (ไฟล์ `~/.gitconfig`)
3. **Local:** การตั้งค่าสำหรับโปรเจกต์นั้นๆ โปรเจกต์เดียว (ไฟล์ `.git/config` ในโฟลเดอร์โปรเจกต์)

คำสั่ง `git config --list` ก็คุณใช้ จะแสดงค่าจาก ทุกระดับ ที่มีอยู่ ทำให้คุณเห็นค่าซึ่กันได้หากมีการตั้งค่าซึ่กันไว้ในไฟล์ต่างระดับกัน

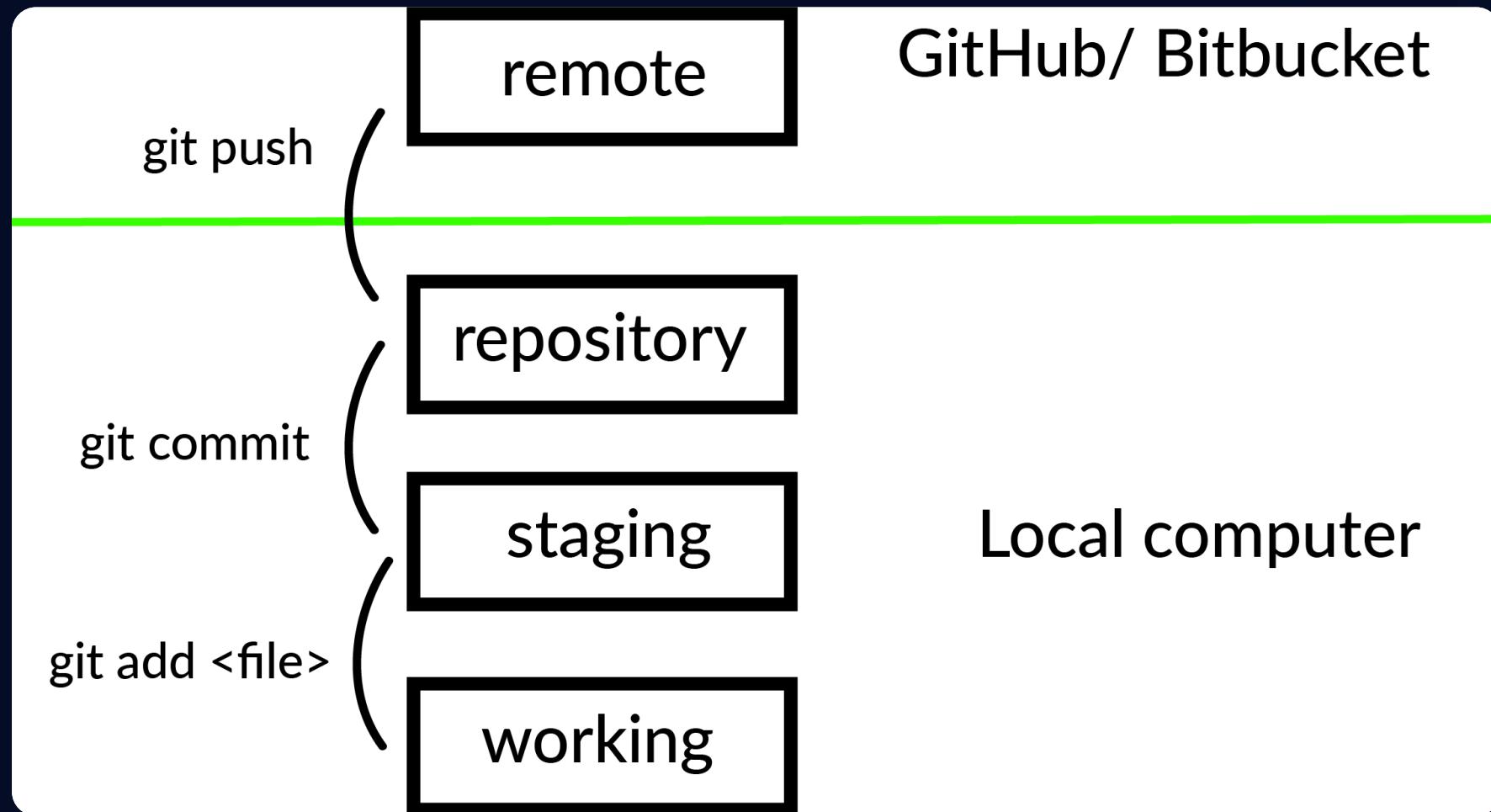


Git Workflow

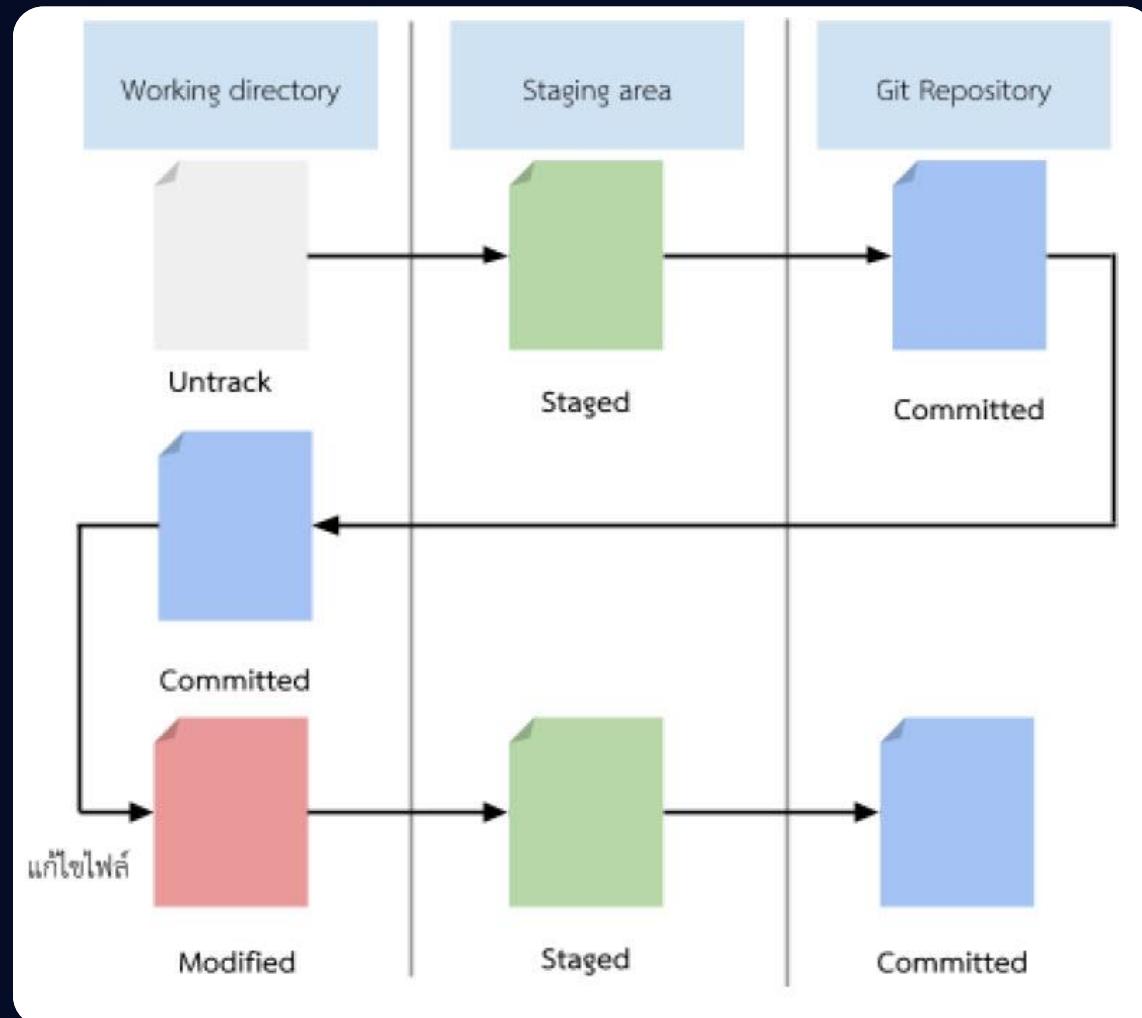
- สร้างไฟล์ใหม่
- เปลี่ยนแปลง แก้ไขไฟล์
- ลบไฟล์
- นำไฟล์ที่เปลี่ยนแปลงเข้า Staging Area
- บันทึกการเปลี่ยนแปลง ไปเก็บไว้ใน Repository อย่างถาวร



Git Work Flow



Git File Life Cycle



Example Workflow Exercise

ดูประวัติการ commit

git log

git log --oneline

เริ่มสร้างระบบ Git ในโปรเจกต์

git init

1

ดูสิ่งที่เปลี่ยนแปลง

git diff master
git diff HEAD

เพิ่ม/แก้ไขไฟล์

2

เรียกดูสถานะไฟล์

git status

3

คำสั่ง track และเพิ่มเข้า Staging area

git add

4

บันทึกประวัติเข้า Repository

git commit

5



คำสั่งพื้นฐาน Git ที่ใช้บ่อยในโปรเจกต์จริง





1. การตั้งค่าเบื้องต้น (Initial Setup)

ใช้ครั้งแรกในเครื่องหรือโปรเจกต์ใหม่

bash

Copy code

```
git config --global user.name "Samit Koyom"      # ตั้งชื่อผู้ใช้  
git config --global user.email "your@email.com" # ตั้งอีเมล  
git config --global core.editor "code --wait"    # ตั้ง VS Code เป็น editor  
git config --list                                # ตรวจสอบการตั้งค่า
```



📁 2. เริ่มต้นโปรเจกต์

bash

```
git init  
git clone <url>
```

เริ่มต้นโปรเจกต์ใหม่
คัดลอก repo จาก remote (GitHub, GitLab)

 Copy code





3. ตรวจสอบสถานะและการเปลี่ยนแปลง

bash



Copy code

```
git status          # ตรวจสอบสถานะไฟล์
git diff           # ดูรายละเอียดการเปลี่ยนแปลง
git log --oneline --graph    # ดูประวัติ commit แบบย่อและเป็นกราฟ
git show <commit_id>      # ดูรายละเอียด commit ได้
```



+ 4. จัดการไฟล์ใน staging area

bash

 Copy code

```
git add .          # เพิ่มไฟล์ทั้งหมด  
git add <file>    # เพิ่มไฟล์เฉพาะ  
git restore --staged <file> # เอาไฟล์ออกจาก staging
```



💬 5. การ commit

bash

Copy code

```
git commit -m "ข้อความ commit"          # commit พร้อมข้อความ  
git commit -am "commit โดยไม่ต้อง add" # เพิ่มและ commit ไฟล์ที่เคย track แล้ว
```





6. การเชื่อมต่อ กับ remote repository

bash

Copy code

```
git remote add origin <url>      # เชื่อมกับ remote  
git remote -v                      # แสดง remote ทั้งหมด  
git push -u origin main            # push ครึ่งแรก กำหนด branch  
git push                          # push ครึ่งต่อไป  
git pull                           # ดึงข้อมูลล่าสุดจาก remote
```



7. การจัดการ Branch

bash

 Copy code

```
git branch          # แสดง branch ทั้งหมด  
git branch <name>      # สร้าง branch ใหม่  
git switch <name>      # สลับไป branch อื่น (หรือใช้ git checkout)  
git merge <name>       # รวม branch เข้ากับ branch ปัจจุบัน  
git branch -d <name>    # ลบ branch ที่ merge แล้ว  
git push origin --delete <name> # ลบ branch บน remote
```





8. การย้อนกลับและแก้ไข

bash



Copy code

```
git restore <file>          # คืนไฟล์กลับเป็นเหมือนล่าสุด  
git reset --hard HEAD        # ย้อนกลับทุกไฟล์เป็น commit ล่าสุด  
git reset --hard <commit_id> # ย้อนกลับไป commit ที่ต้องการ  
git revert <commit_id>       # สร้าง commit ใหม่เพื่อยกเลิกการเปลี่ยนแปลง
```





9. การดูและเลือก commit

bash

Copy code

```
git log --oneline          # แสดง commit ย่อ  
git checkout <commit_id>    # ไปยัง commit นั้น (detached HEAD)  
git switch -c newbranch <commit_id> # สร้าง branch ใหม่จาก commit เก่า
```



💡 10. การ Sync โปรเจกต์

bash

 Copy code

```
git fetch                                # ดึงข้อมูลจาก remote โดยไม่ merge  
git pull origin main                      # ดึงและรวมจาก remote main  
git push origin main                      # อัปเดตชิ้น remote main
```





11. คำสั่งอื่นที่มีประโยชน์

bash

 Copy code

```
git stash                      # เก็บการเปลี่ยนแปลงชั่วคราว  
git stash pop                 # ดึงกลับมาใช้งาน  
git tag <version>            # สร้าง tag  
git push origin --tags        # ส่ง tag ขึ้น remote
```



📌 ตัวอย่าง Workflow พื้นฐาน

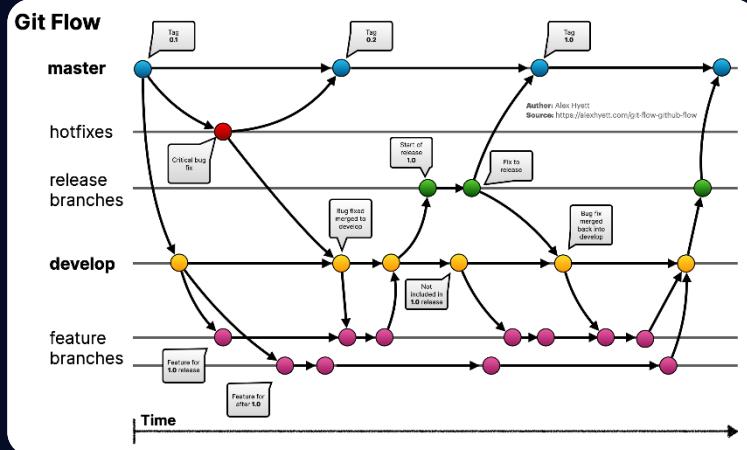
bash

 Copy code

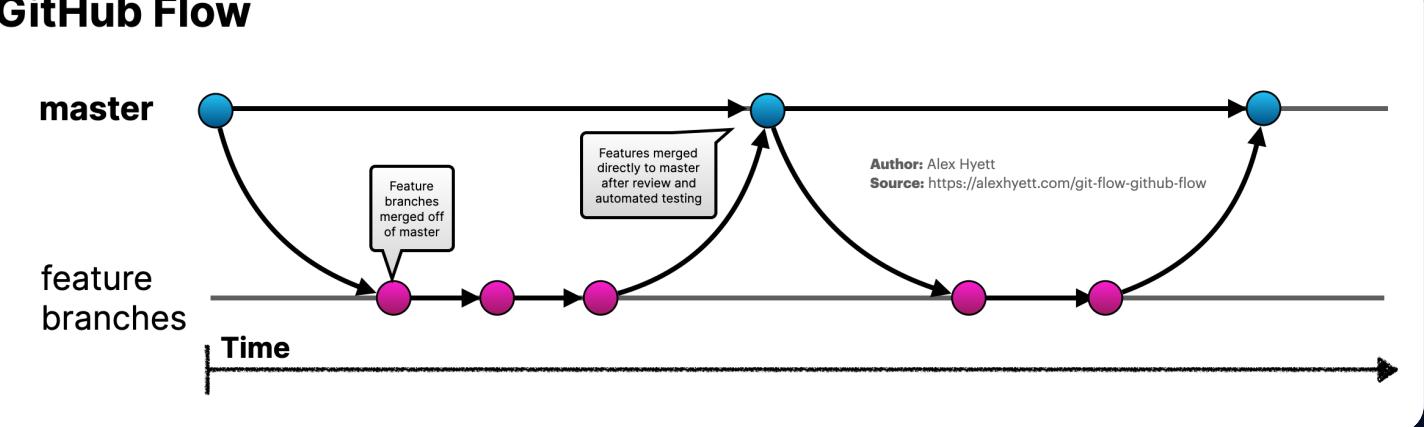
```
git init  
git add .  
git commit -m "init project"  
git branch -M main  
git remote add origin https://github.com/iamsamitdev/project.git  
git push -u origin main
```



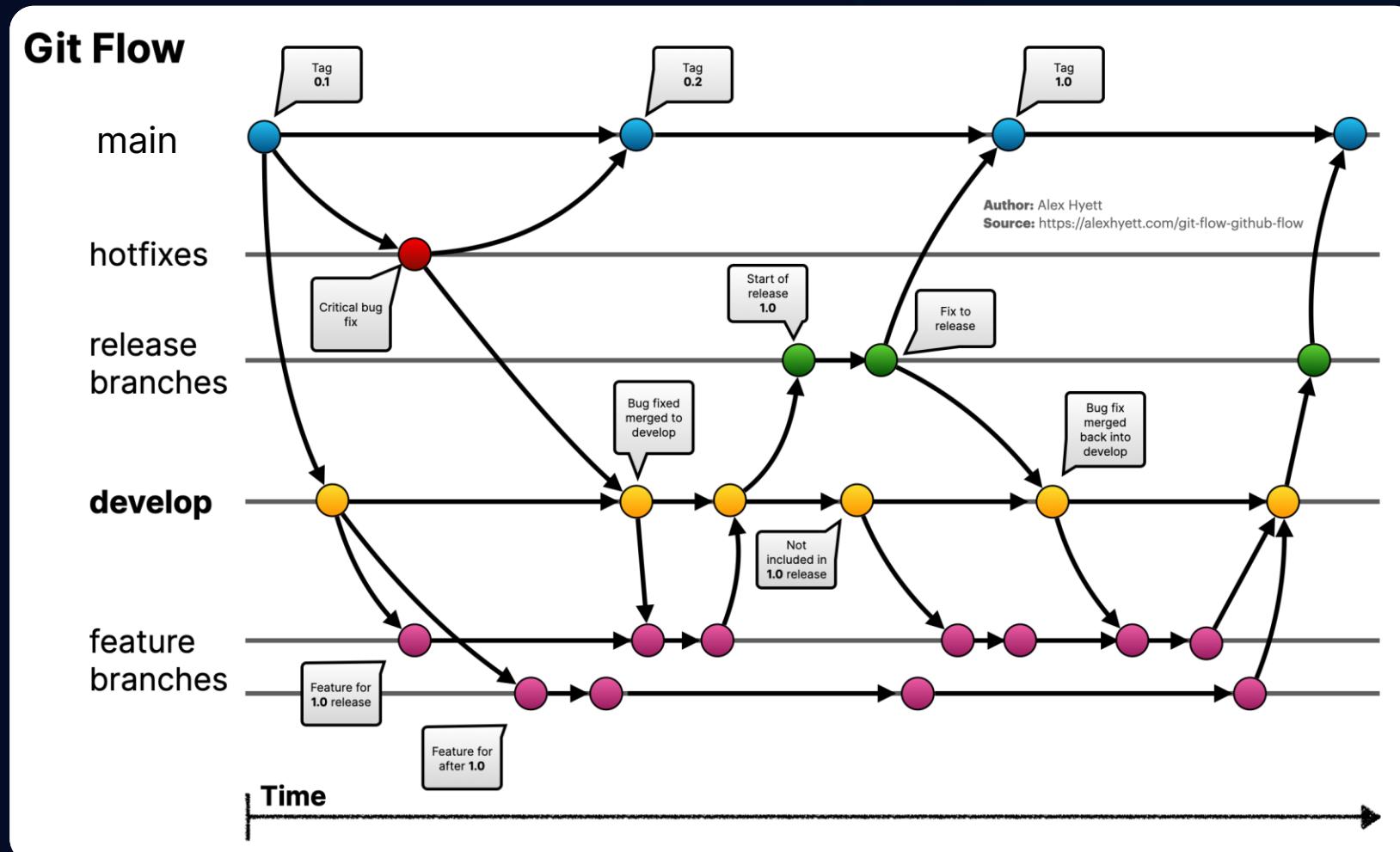
Git Flow vs GitHub Flow



GitHub Flow

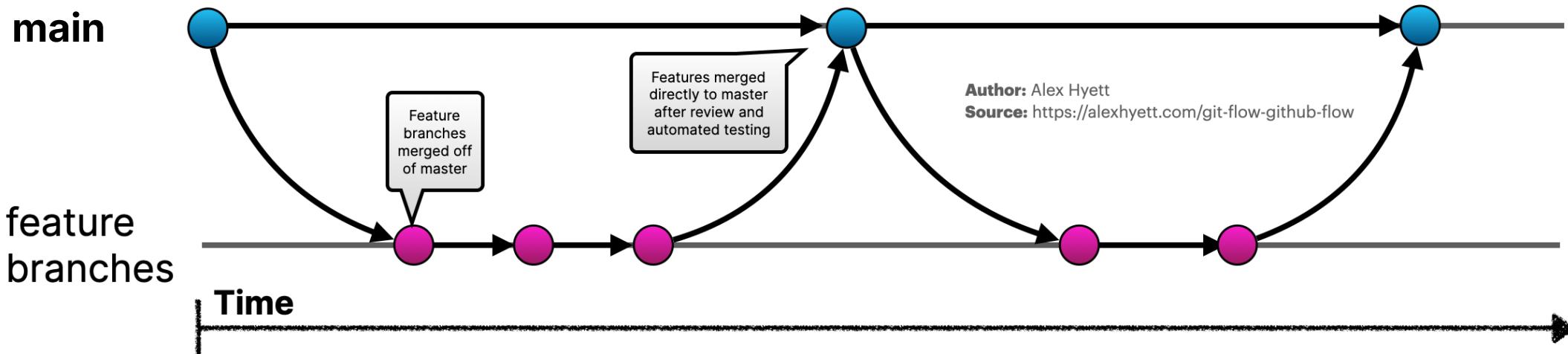


Git Flow



GitHub Flow

GitHub Flow



• LIVE

อบรมออนไลน์



ปั้นระบบอัตโนมัติอย่างมือโปรดด้วย

Jenkins และ GitHub Actions ร่วมกับ n8n



มีวิดีโอบันทึกการอบรม
ย้อนหลังให้ทุกวัน



สอนสดผ่าน Zoom
รับจำนวนจำกัด

วันที่

2



Samit Koyom
สถาบันไอทีจีเนียส



Day 2



2. พื้นฐาน Docker



Jenkins
และ GitHub
Actions
ร่วมกับ n8n



สถาบันไอทีจีเนียส

www.itgenius.co.th



พื้นฐาน Docker

- รู้จัก Docker และแนวคิด Container
- รู้จัก Docker Images และ Containers
- การสร้าง Custom Images (Dockerfile)
- รู้จัก Docker Compose และการจัดการ

WHAT IS DOCKER?



รู้จัก Virtual Machine



Lenovo ThinkServer RD340
70AB001UUX 1U Rack Server -
1 x Intel Xeon E5-2407 v2 2.40
GHz Ram 16GB

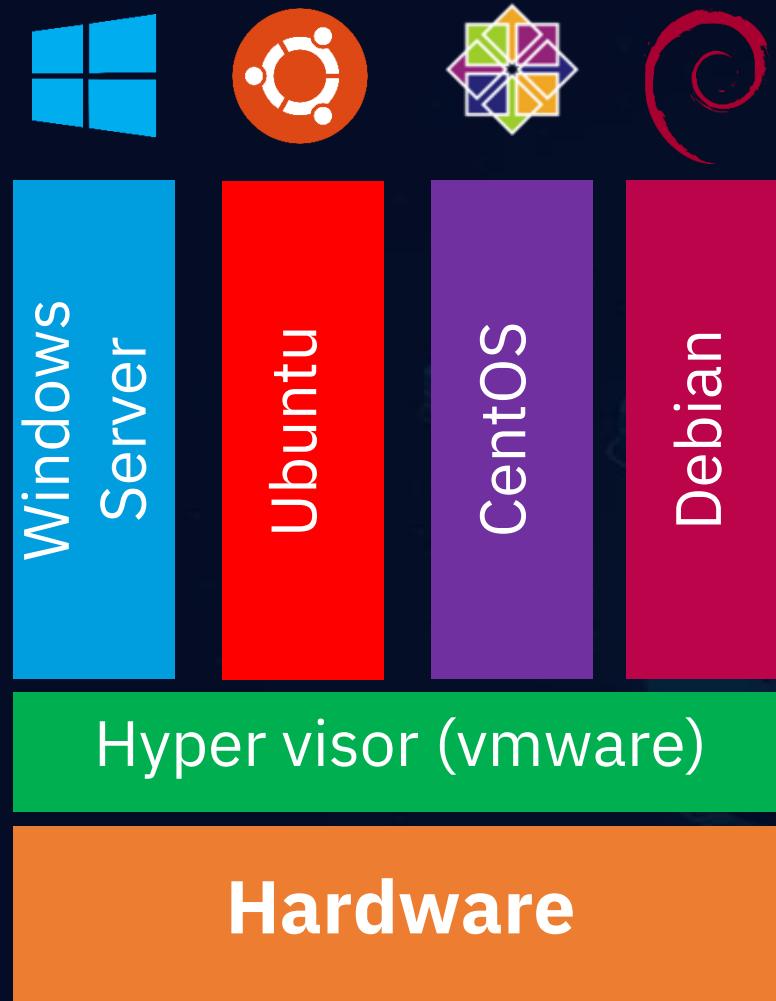
VMware ESXi

Windows Server
2 CPU
4 GB Ram

Linux Ubuntu
1 CPU
2 GB Ram



รู้จัก Virtual Machine





คือ **Software Container** ถูกออกแบบมาเพื่อสร้างสภาพแวดล้อมให้ Application ของเราทำงานร่วมกันได้บน OS เดียวกัน ช่วยให้การ Setup และจัดการ Application ต่าง ๆ ทำได้ง่ายขึ้น



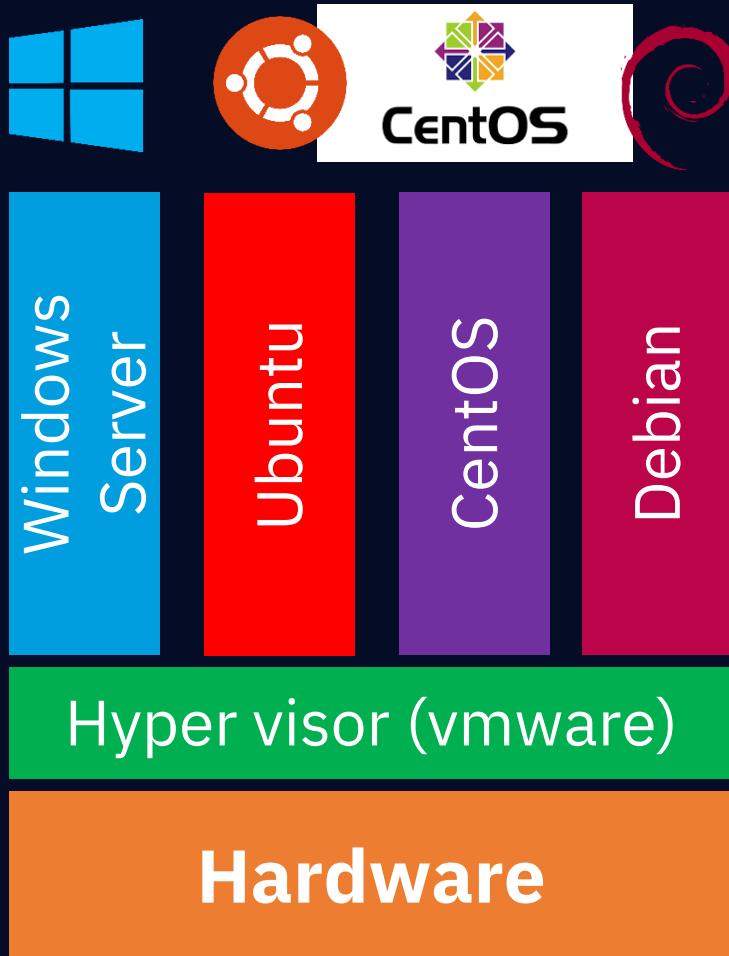
Container คืออะไร



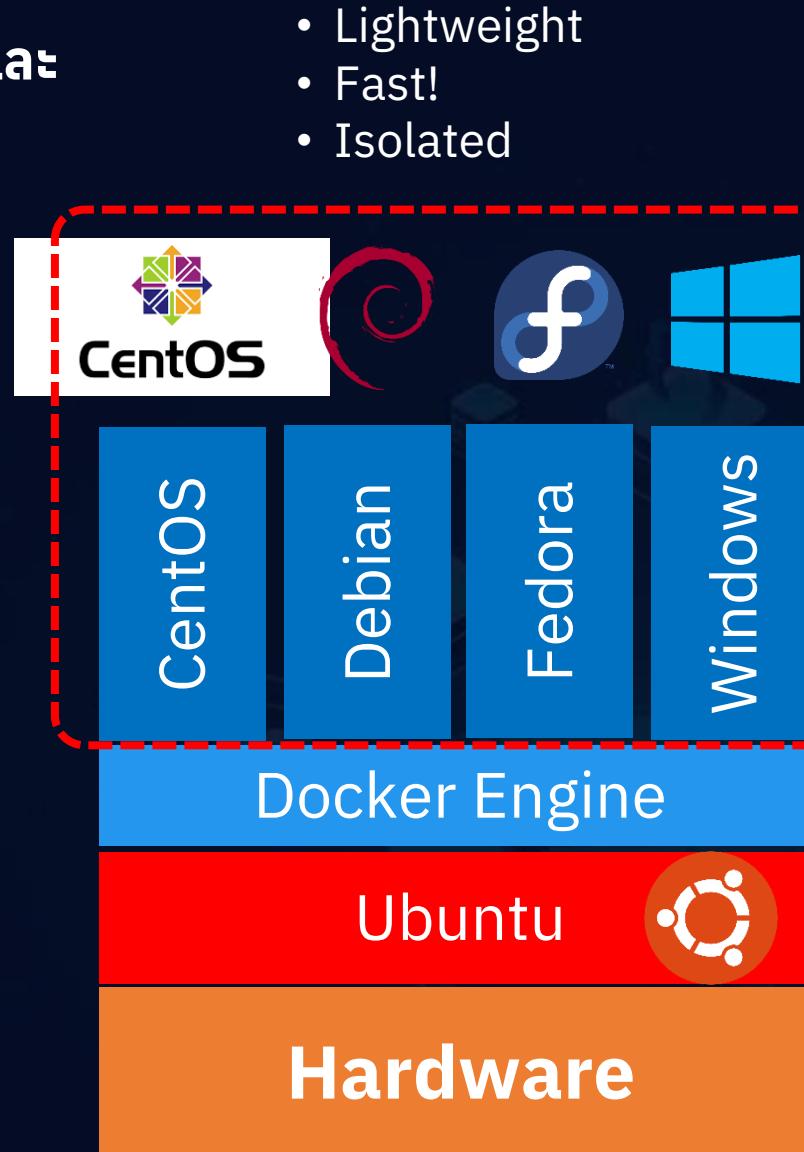
เปรียบเสมือนการ Pack รวม App/Bins/Libs ต่างๆ เป็นก้อนเดียวกัน แล้วทำการกำหนดสภาพแวดล้อมการทำงานไว้ภายในตัวมันเอง

เมื่อต้องการนำไปใช้งานที่อื่นหรือบน Production จริง ก็นำไปกังก้อน Container นี้เลย ทำให้สภาพแวดล้อม เมื่อกันทุกประการ

ความแตกต่างระหว่าง Virtual Machine และ Docker Container



VM
สถาบันไอทีเจเนียส



Container

- Lightweight
- Fast!
- Isolated

Container
Micro Computer

- OS
- CPU
- Mem
- Network

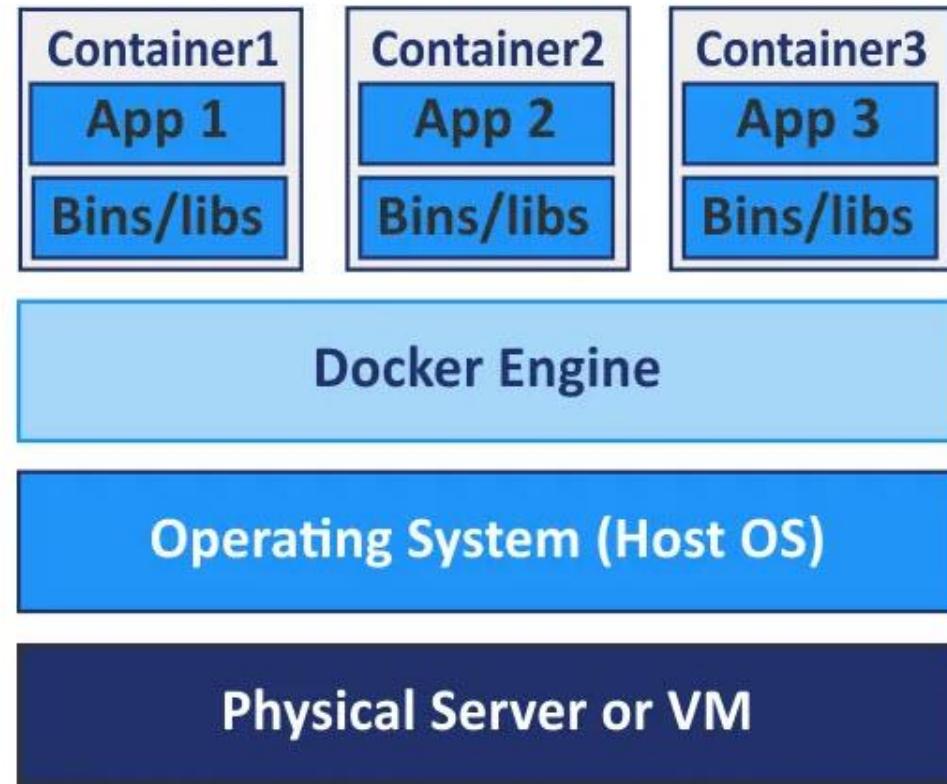


ความแตกต่างระหว่าง Virtual Machine และ Docker Container

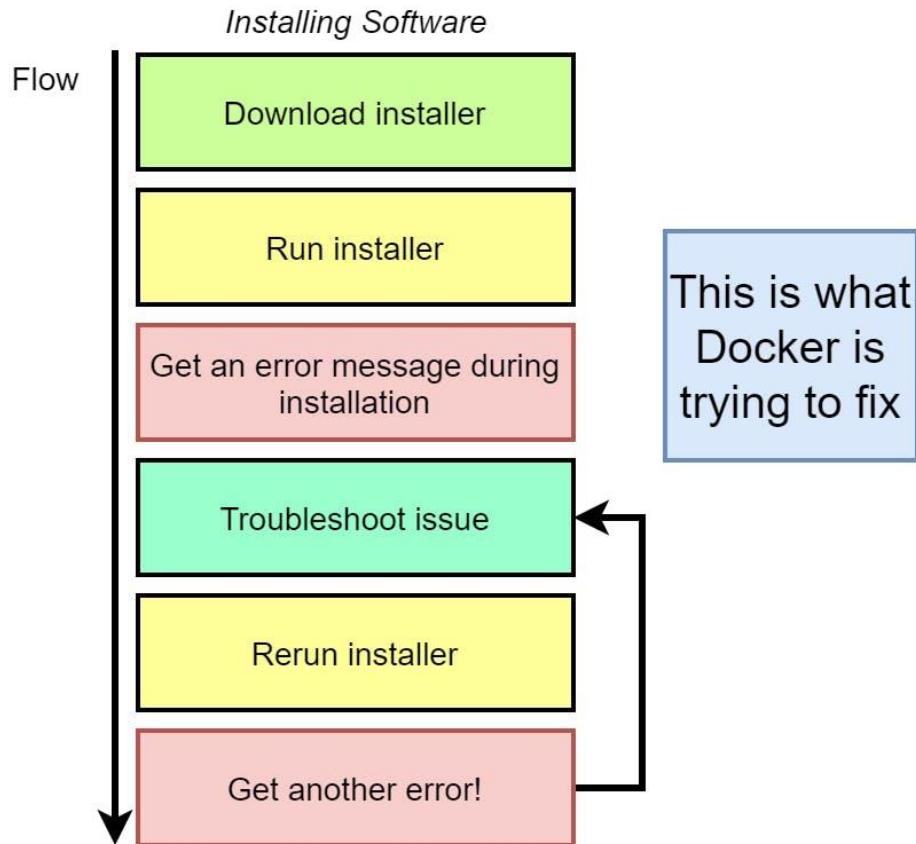
Virtual Machines



Containers

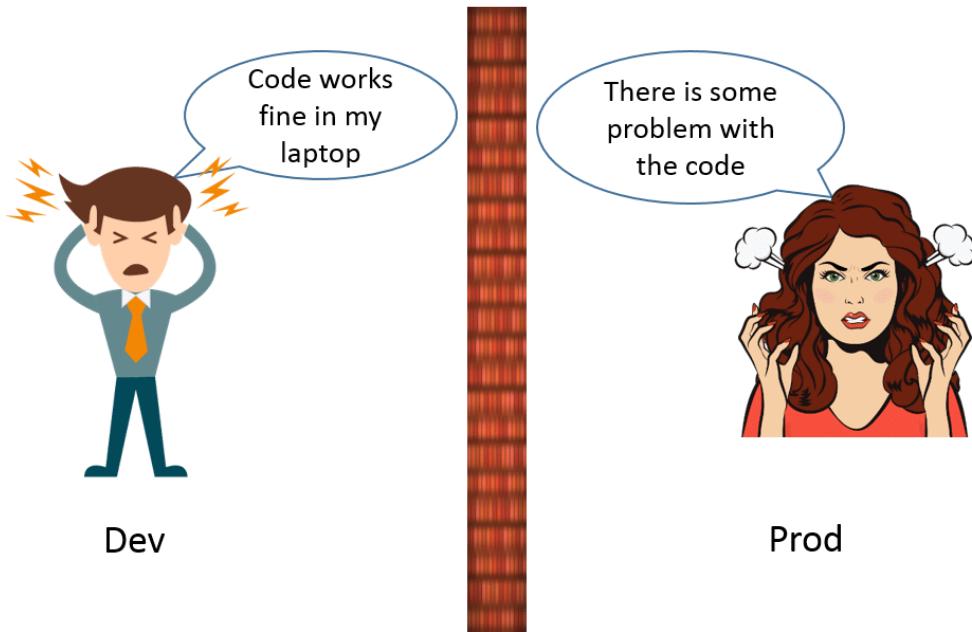


ทำไมต้องใช้ Docker และ Containers



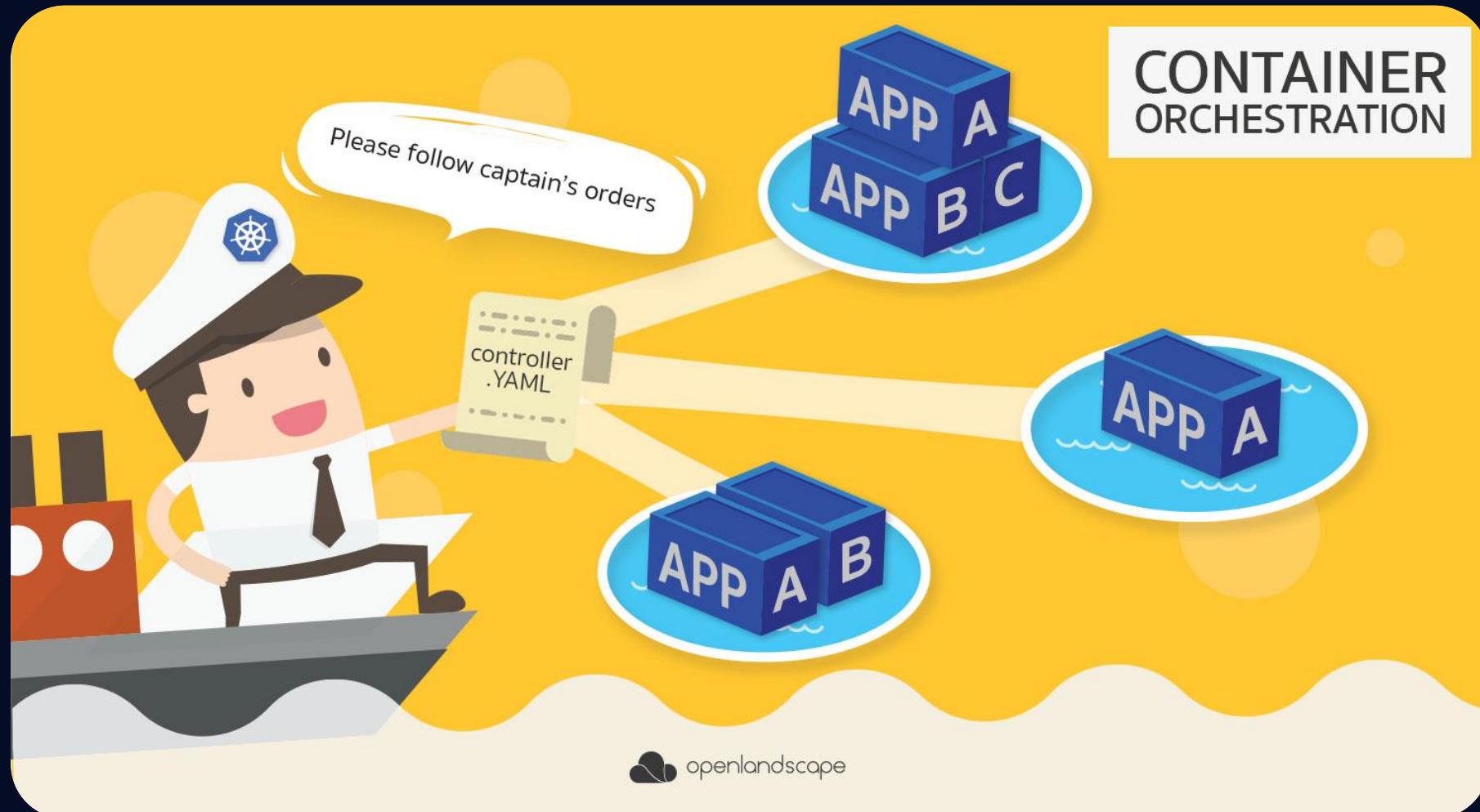
ทำไมต้องใช้ Docker และ Containers

In Dev there can be a software that is upgraded and in Prod the old version of that software might be present

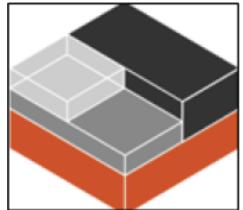


KEEP
CALM
IT
WORKS ON
ALL MACHINES

Kubernetes คุณางสู่การทำระบบที่ไม่มีวันล่ม



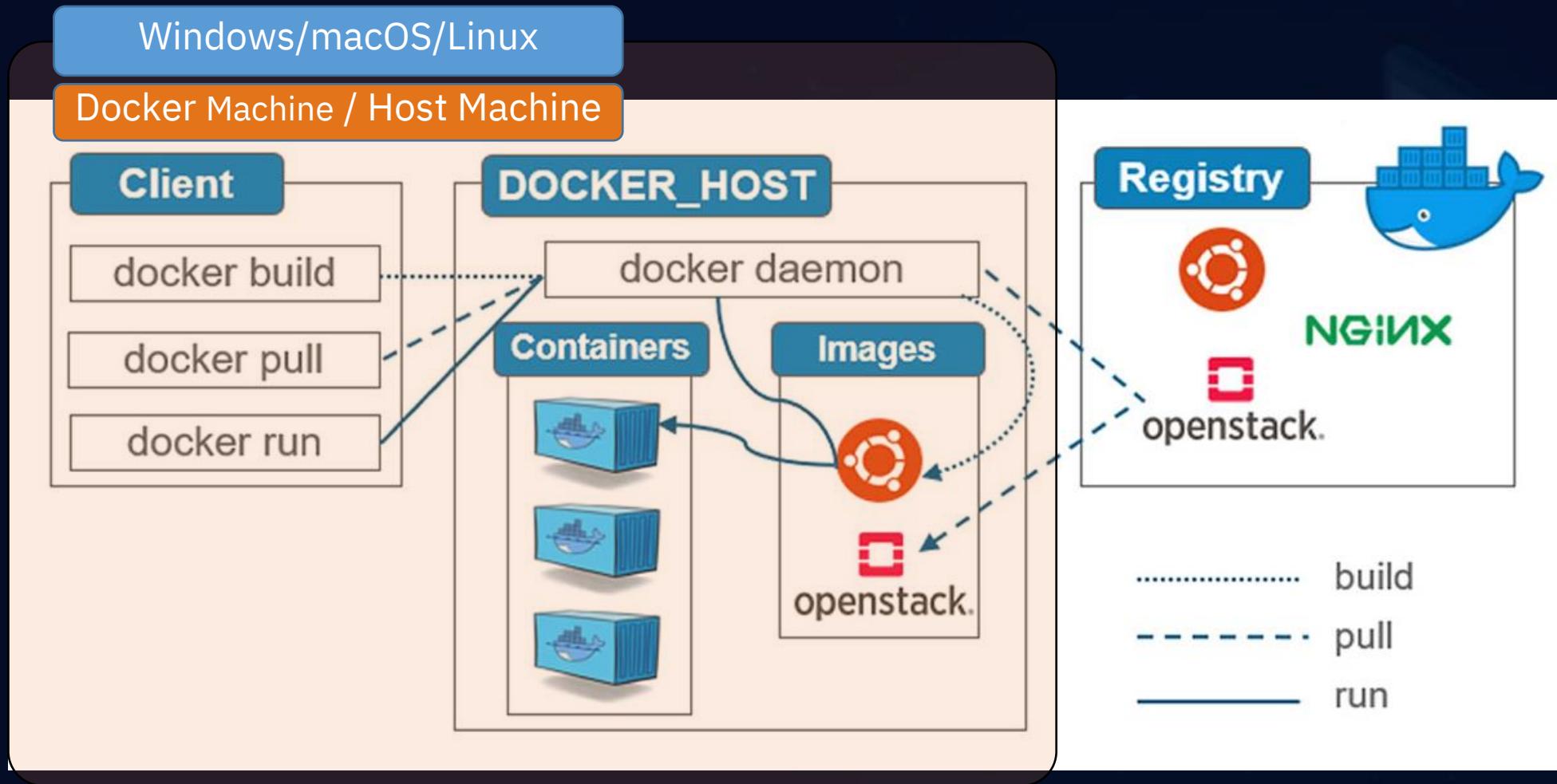
Other Software Container



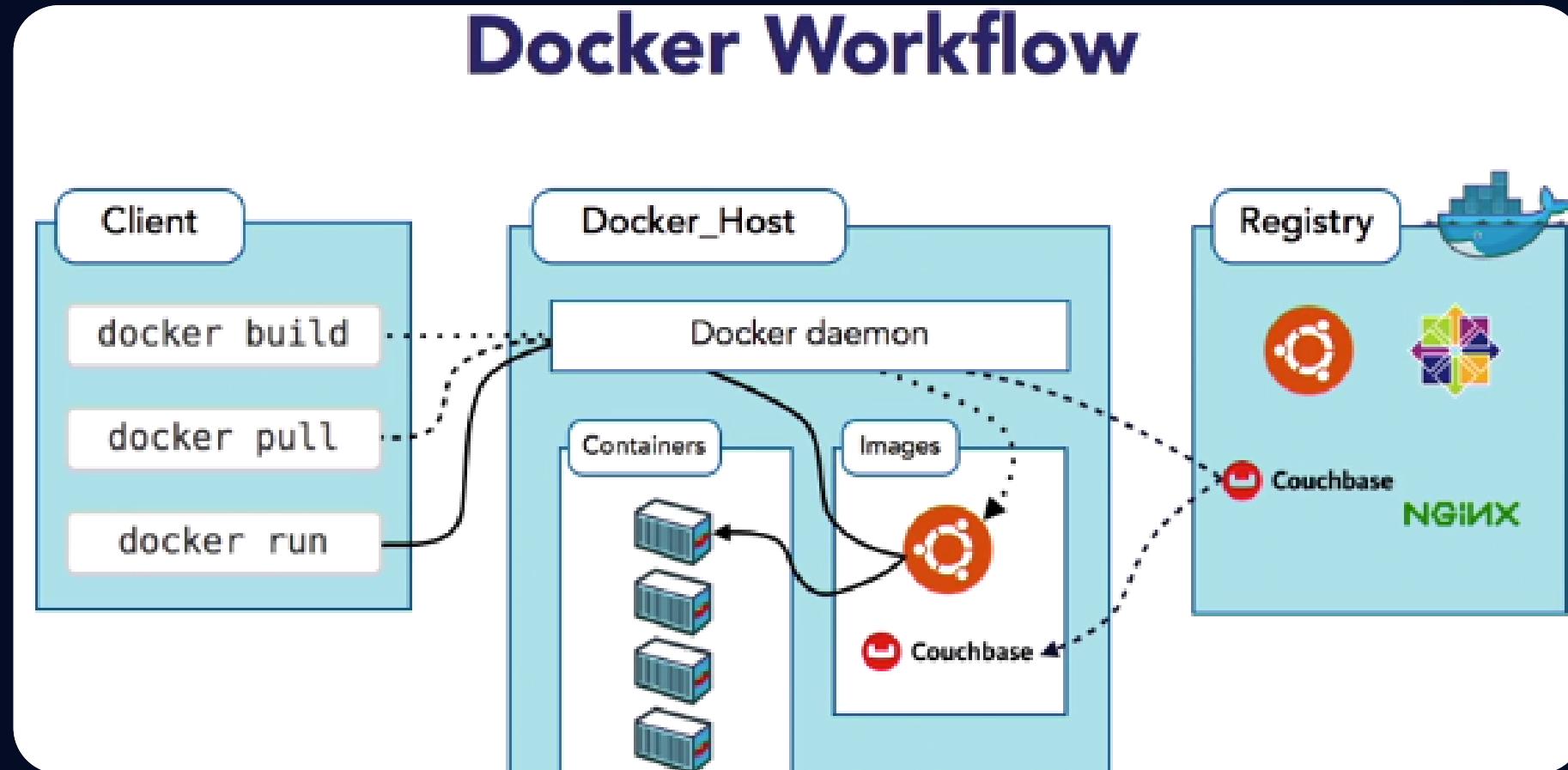
LXC (Linux)



โครงสร้างและสถาปัตยกรรมของ Docker



Workflow ในการทำงานกับ Docker



Workflow ในการทำงานกับ Docker

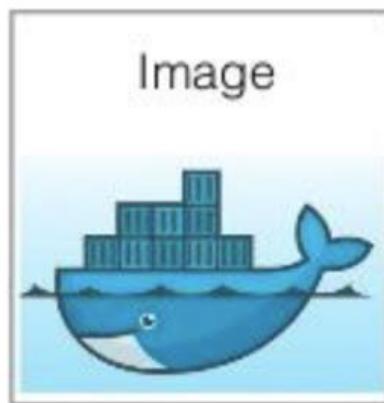


Workflow ในการทำงานกับ Docker



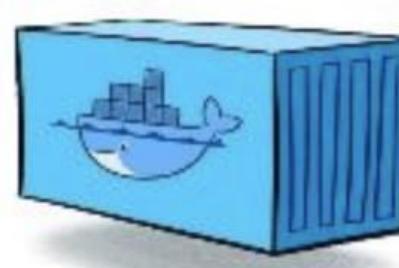
Dockerfile

build



Docker Image

run



Docker Container



Hello World



```
docker run hello-world
```

```
$ docker run hello-world
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

<https://hub.docker.com/>

For more examples and ideas, visit

<https://docs.docker.com/get-started/>

```
$ docker images hello-world
```

REPOSITORY	TAG	IMAGE ID	SIZE
hello-world	latest	1b44b5a3e06a	10.07kB



```
$ docker run hello-world
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

<https://hub.docker.com/>

For more examples and ideas, visit

<https://docs.docker.com/get-started/>

```
$ docker images hello-world
REPOSITORY      TAG          IMAGE ID          SIZE
hello-world    latest        1b44b5a3e06a   10.07kB
```



คำสั่งพื้นฐาน Docker ที่ใช้บ่อยในโปรเจ็คต์จริง



Docker Basic Command



ลบรายการ image กึ่งหมด

```
$ docker image prune -a
```

แสดง version ของ docker

```
$ docker version
```

แสดงสถานะของ docker

```
$ docker stats
```

รายการ Images กึ่งหมด รายการ Container กึ่งหมด

```
$ docker images
```

```
$ docker image ls
```

รายการ Container ที่กำกันอยู่

```
$ docker ps
```

```
$ docker ps -a
```

ดึงรายการ image จาก registry

```
$ docker pull <imagename:tag>
```

ลบรายการ image ที่ไม่ต้องการ

```
$ docker rmi <imagename:tag>
```

Container Management



สร้าง Container

```
$ docker create <imagename|id>
```

สร้างและรัน Container (create + start)

```
$ docker run <imagename|id>
```

Start/Stop Container

```
$ docker [start|stop] <container_id>
```

Stop All Container

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

ลบรายการ Container (ที่ stop แล้ว)

```
$ docker rm <container_id>
```

ลบรายการ Container (ที่ start อยู่)

```
$ docker rm -f <container_id>
```

ลบรายการ Container ก็งหมดที่ stop อยู่

```
$ docker container prune
```



สถาบันไอทีเนยส์



1. การตรวจสอบและตั้งค่าระบบ (System Info & Setup)

คำสั่ง

```
docker --version
```

คำอธิบาย

ตรวจสอบเวอร์ชันของ Docker

```
docker info
```

แสดงข้อมูลระบบ Docker เช่น จำนวน container, image

```
docker login
```

เข้าสู่ระบบ Docker Hub

```
docker logout
```

ออกจากระบบ Docker Hub





2. การจัดการ Images (Image Management)

คำสั่ง

คำอธิบาย

`docker images`

แสดงรายการ image ทั้งหมดที่มีอยู่ในเครื่อง

`docker pull <image>`

ดาวน์โหลด image จาก Docker Hub

`docker rmi <image>`

ลบ image ออกจากเครื่อง

`docker tag <source> <target>`

เปลี่ยนชื่อหรือแท็ก image

`docker build -t <name> .`

สร้าง image จาก Dockerfile



คำแนะนำ: ก่อน deploy จริง ควรใช้ `docker image prune` เพื่อล้าง image ที่ไม่ได้ใช้





3. การจัดการ Containers (Container Lifecycle)

คำสั่ง

```
docker ps
```

```
docker ps -a
```

```
docker run <image>
```

```
docker run -d -p 8080:80 --name web nginx
```

```
docker start <container>
```

```
docker stop <container>
```

```
docker restart <container>
```

```
docker rm <container>
```

```
docker kill <container>
```

คำอธิบาย

แสดง container ที่กำลังทำงาน

แสดง container ทั้งหมด (รวมที่หยุดแล้ว)

สร้างและรัน container ในมี

รันแบบ background และกำหนดชื่อ พร้อม map port

เริ่ม container ที่หยุดไว้

หยุด container

รีสตาร์ท container

ลบ container ที่หยุดแล้ว

บังคับหยุด container





4. ตรวจสอบและเข้าใช้งาน Container (Inspection & Debug)

คำสั่ง

```
docker logs <container>
```

คำอธิบาย

ดู log ของ container

```
docker logs -f <container>
```

ดู log แบบ realtime

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

เข้า shell ภายใน container

```
docker inspect <container>
```

ดูรายละเอียดเต็มของ container (เช่น IP, volume)

```
docker top <container>
```

แสดง process ที่รันอยู่ใน container





5. การจัดการ Network และ Volume

คำสั่ง

```
docker network ls
```

คำอธิบาย

แสดง network ทั้งหมด

```
docker network inspect <name>
```

ดูรายละเอียด network

```
docker network create <name>
```

สร้าง network ใหม่

```
docker volume ls
```

แสดง volume ทั้งหมด

```
docker volume inspect <name>
```

ดูรายละเอียด volume

```
docker volume rm <name>
```

ลบ volume





6. การทำความสะอาดระบบ (Cleanup)

คำสั่ง

docker system prune

คำอธิบาย

ลบทุกอย่างที่ไม่ใช้งาน (container, image, volume, network)

docker image prune

ลบเฉพาะ image ที่ไม่ได้ใช้งาน

docker container prune

ลบ container ที่หยุดทำงาน

docker volume prune

ลบ volume ที่ไม่ได้ใช้งาน



คำเตือน: ควรตรวจสอบด้วย `docker ps -a` ก่อน prune เพื่อป้องกันการลบผิด



7. การใช้งาน Docker Compose (นโยบายบริการพร้อมกัน)

คำสั่ง

`docker-compose up`

คำอธิบาย

สร้างและรัน service จาก `docker-compose.yml`

`docker-compose up -d`

รันแบบ background

`docker-compose down`

หยุดและลบ service ทั้งหมด

`docker-compose ps`

แสดง service ที่กำลังทำงาน

`docker-compose logs -f`

ดู log ทั้งหมดแบบ realtime





ตัวอย่างไฟล์ docker-compose.yml

yaml

Copy code

```
version: '3'  
services:  
  web:  
    image: nginx  
    ports:  
      - "8080:80"  
  db:  
    image: postgres  
  environment:  
    POSTGRES_PASSWORD: example
```



⚡ 8. ตัวอย่าง Workflow พื้นฐาน (มือใหม่ควรรู้)

bash

 Copy code

```
# ดึง image
docker pull nginx

# สร้างและรัน container พร้อม port
docker run -d -p 8080:80 --name mynginx nginx

# ตรวจสอบสถานะ
docker ps

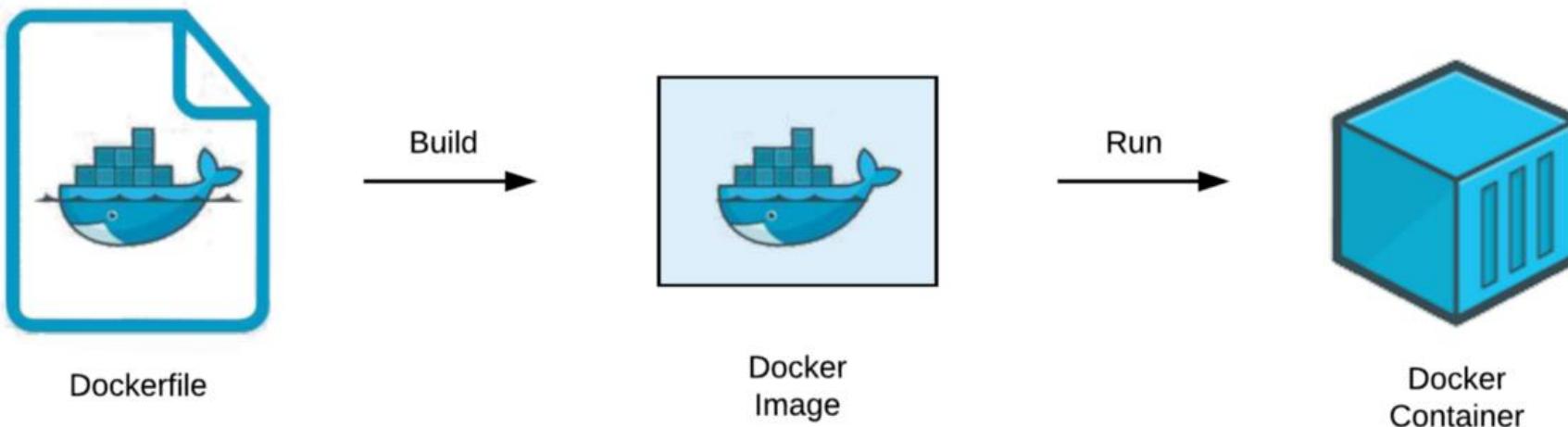
# ดู Log
docker logs mynginx

# เข้าไปใน container
docker exec -it mynginx bash

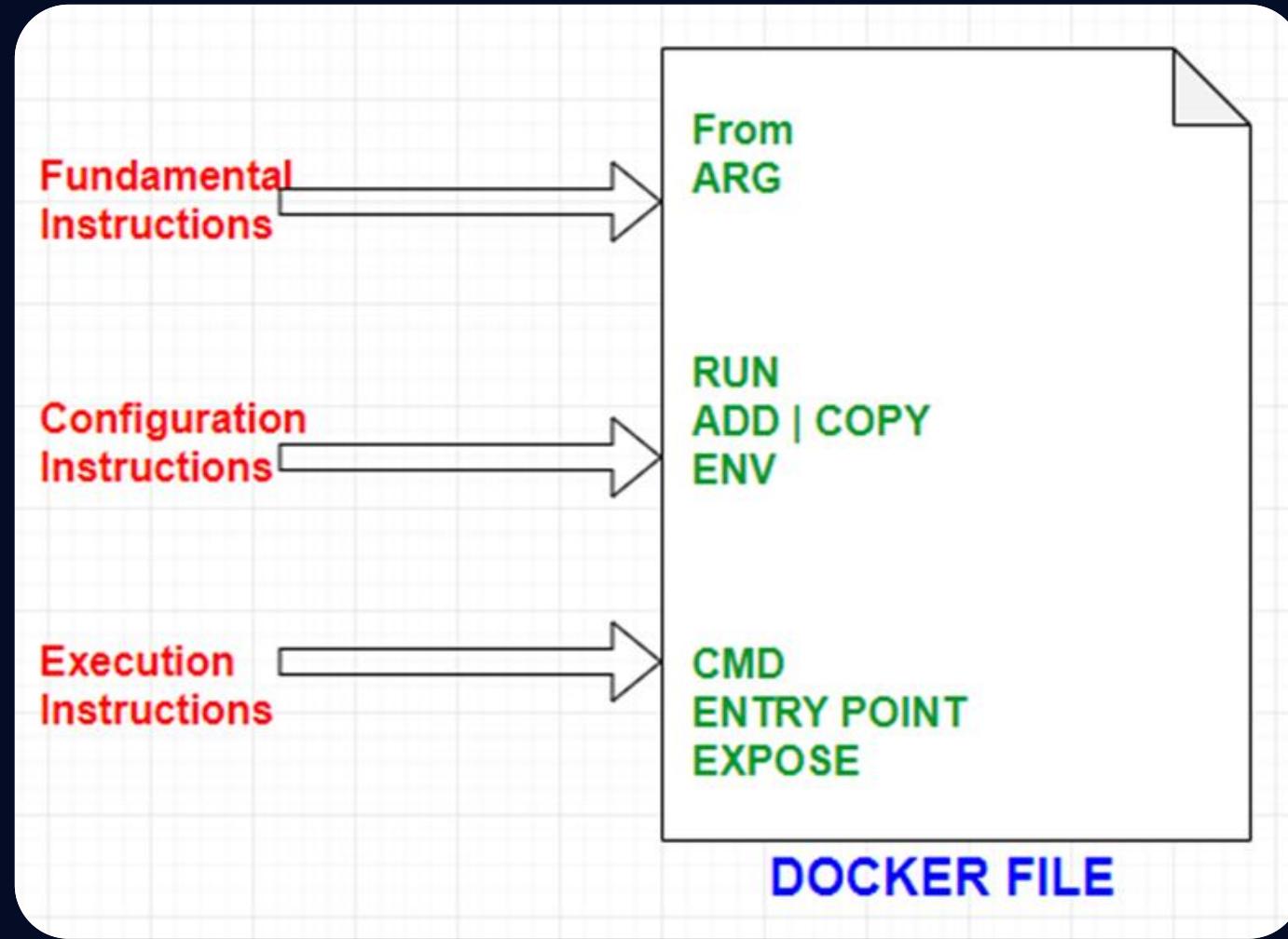
# หยุดและลบ container
docker stop mynginx
docker rm mynginx
```



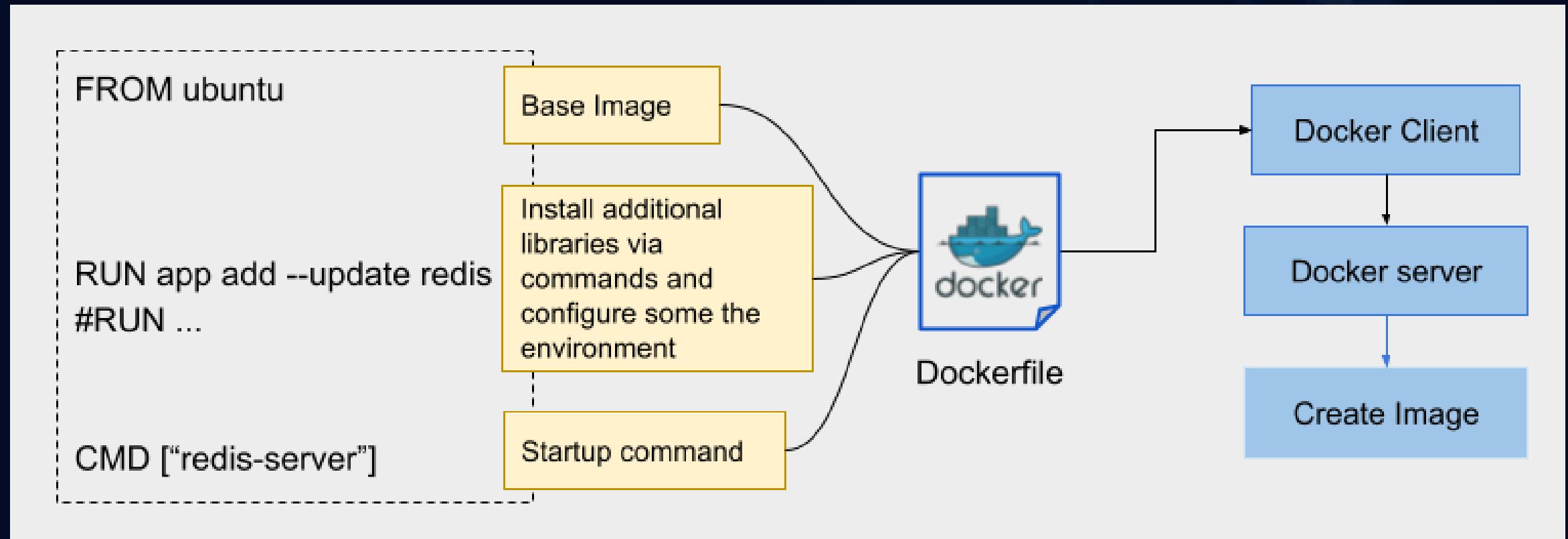
Docker File



Docker File Structure



Docker File Structure



Docker File Structure

```
FROM python:3
```

```
RUN pip3 install redis  
RUN pip3 install numpy  
RUN pip3 install pandas
```

dependencies

```
ADD executable.py /executable.py  
RUN ["chmod", "+x", "/executable.py"]
```

executable

```
ADD ProcessFileAdapterInput.py /ProcessFileAdapterInput.py  
RUN ["chmod", "+x", "/ProcessFileAdapterInput.py"]  
ADD ProcessFileAdapterOutput.py /ProcessFileAdapterOutput.py  
RUN ["chmod", "+x", "/ProcessFileAdapterOutput.py"]
```

adapter

```
ADD ProcessRedisConsumer.py /ProcessRedisConsumer.py  
RUN ["chmod", "+x", "/ProcessRedisConsumer.py"]  
ADD ProcessRedisPublisher.py /ProcessRedisPublisher.py  
RUN ["chmod", "+x", "/ProcessRedisPublisher.py"]
```

Redis communication

```
ADD ProcessRuntimeManagement.py /ProcessRuntimeManagement.py  
RUN ["chmod", "+x", "/ProcessRuntimeManagement.py"]
```

manager

```
ENTRYPOINT ["python3", "/ProcessRuntimeManagement.py"]
```



คำสั่งหลัก ๆ ของ Dockerfile

คำสั่ง	คำอธิบาย
FROM	กำหนด Base Image
LABEL	กำหนด Metadata เช่น ชื่อ version หรือเจ้าของ Image
ENV	กำหนด Environment Variable ภายใน Container
RUN	สำหรับติดตั้ง Packages ให้ Container
COPY	สำหรับคัดลอกไฟล์และโฟลเดอร์ไปยัง Container
ADD	สำหรับคัดลอกไฟล์และโฟลเดอร์ไปยัง Container โดยสามารถแตกไฟล์ .tar และคัดลอกจาก Host ภายนอกได้
CMD	สำหรับรับคำสั่งที่ต้องการขณะรัน Container
WORKDIR	กำหนด Working Directory ของ Container
ARG	กำหนด Variable ขณะสร้าง Image
ENTRYPOINT	สำหรับรับคำสั่งที่ต้องการขณะรัน Container
EXPOSE	กำหนด Port ที่เปิดให้ Container อีนติดต่อเข้ามา
VOLUME	สร้าง Folder เก็บข้อมูลแบบถาวรให้ Container



Docker File Structure

```
nodejs
  Dockerfile
  index.js
  package.json
```

Dockerfile

```
1 # Specify a base image
2 FROM node:alpine
3 WORKDIR /usr/app
4
5 # Install some dependencies
6 COPY ./package.json ./
7 RUN npm install
8 COPY ./ ./
```

9

```
10 # Default command
11 CMD ["npm", "start"]
12 |
```



Docker File Structure

Dockerfile x

```
1 # Specify a base image
2 FROM node:alpine
3 WORKDIR /usr/app
4
5 # Install some dependencies
6 COPY ./package.json .
7 RUN npm install
8 COPY ./ .
9
10 # Default command
11 CMD ["npm", "start"]
```

The diagram illustrates the build process of a Docker image. On the left, a dark gray rounded rectangle contains the Dockerfile code. An upward-pointing arrow points from the bottom of this box to the first layer of the resulting image on the right. The right side shows a vertical stack of seven rectangular layers, each representing a different stage of the build. The top layer is labeled 'writeable container layer' and includes 'docker run nodejsapp'. The other six layers are 'image layers' corresponding to the commands in the Dockerfile: 'CMD ["npm", "start"]', 'COPY ./ .', 'RUN npm install', 'COPY ./package.json .', 'WORKDIR /usr/app', and 'FROM node:alpine'.

cf650ef85086	writeable container layer: docker run nodejsapp
995a21532fce	image layer: CMD ["npm", "start"]
ecf7275feff3	image layer: COPY ./ .
334d93a151ee	image layer: RUN npm install
86c81d89b023	image layer: COPY ./package.json .
7184cc184ef8	image layer: WORKDIR /usr/app
530c750a346e	base image: FROM node:alpine



• LIVE

อบรมออนไลน์



ปั้นระบบอัตโนมัติอย่างมือโปรดด้วย

Jenkins และ GitHub Actions ร่วมกับ n8n



มีวิดีโอบันทึกการอบรม
ย้อนหลังให้ทุกวัน



สอนสดผ่าน Zoom
รับจำนวนจำกัด

วันที่

3



Samit Koyom
สถาบันไอทีจีเนียส

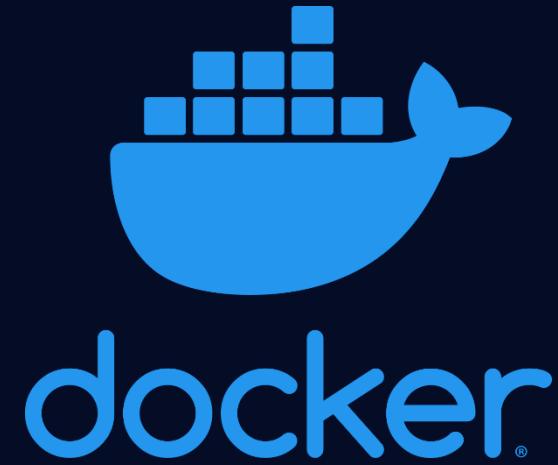
Day 3



2. พื้นฐาน Docker (ต่อ)
3. เริ่มต้นกับ Jenkins Server



Jenkins
และ GitHub Actions
ร่วมกับ n8n



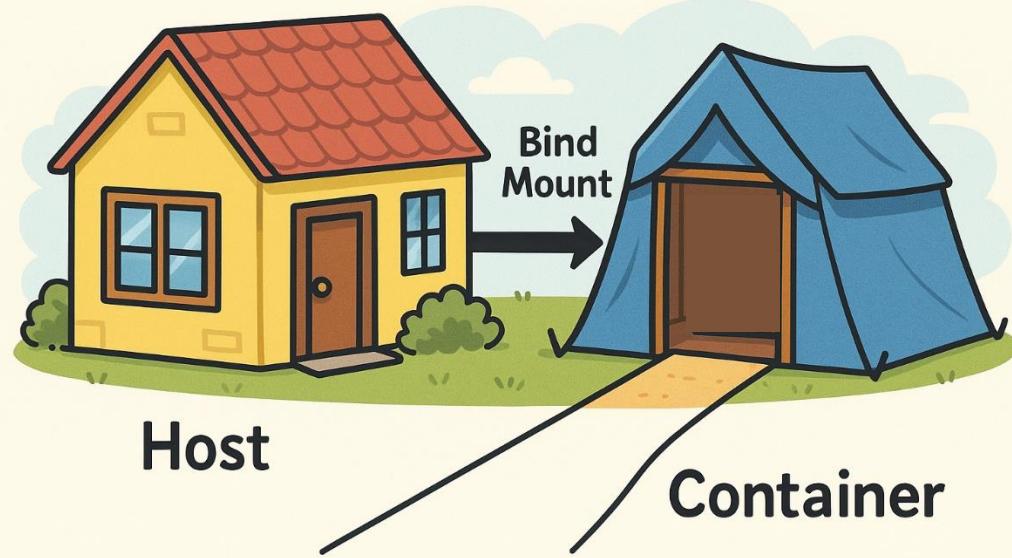
Bind Mount Volume



Bind Mount Volume

```
// MySQL  
docker run --name mysql --network wordpress -v db_data:/var/lib/mysql -e  
MYSQL_ROOT_PASSWORD=1111 -d mysql:5.7
```

Bind Mount is a "Door", Not a "Copy"



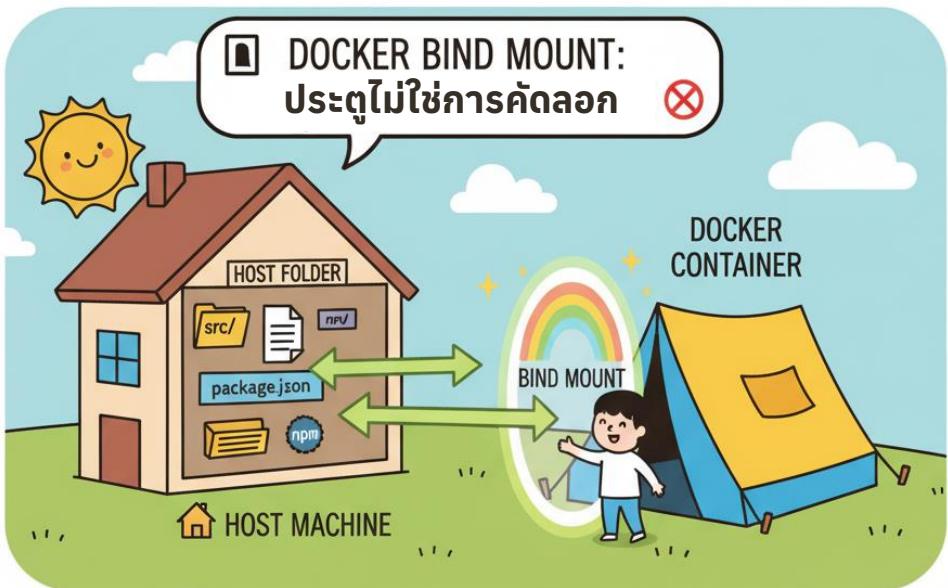
การทำงานของ Bind Mount

Bind mount ทำหน้าที่เหมือนการ "เชื่อม" หรือ "สะก้อน" ไฟล์เดอร์หรือไฟล์จากเครื่องคอมพิวเตอร์ของคุณ (Host) เข้าไปใน Docker Container โดยตรง มันไม่ใช่การคัดลอกข้อมูล แต่เป็นการทำให้กั้งสองที่ (Host และ Container) มองเห็นและใช้งานไฟล์ชุดเดียวกัน

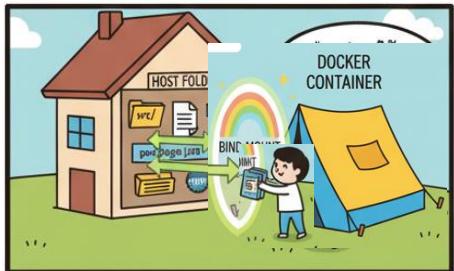
สิ่งที่เกิดขึ้นเมื่อ lob ข้อมูลฝัง Host

- เมื่อคุณ lob ไฟล์หรือไฟล์เดอร์ในฝั่ง Host ที่ทำการ bind mount ไว้...
- การเปลี่ยนแปลงนั้นจะสะก้อนเข้าไปใน Container กันที
- ดังนั้น ข้อมูลในฝั่ง Container ก็จะถูกกลบหายไปด้วย

-v db_data:/var/lib/mysql



ทำงาน (CONTAINER RUNNING)



ทำงาน (CONTAINER RUNNING)

หยุด (CONTAINER STOPPED)



เติมกู้ครื้อ! แต่ข้อมูลในบ้านยังอยู่ 😊

Bind Mount คือ "ประตู" ไม่ใช่ "การคัดลอก"

- เครื่องคอมพิวเตอร์ของคุณ (Host) คือ บ้านของคุณ 🏠
- โฟลเดอร์ใน Host คือ ห้องเก็บของในบ้านของคุณ
- Docker Container คือ เต็นท์ที่คุณugasชั่วคราวในสนามหญ้า 🌳
- Bind Mount คือ การที่คุณสร้างประตูวิเศษจากในเต็นท์ (Container) ให้เปิดไปเจาะห้องเก็บของ (Host folder) ได้โดยตรง

ตอนทำงาน (Container is running): เมื่อคุณเข้าไปในเต็นท์ (Container) และเปิดประตูวิเศษนี้ คุณจะเห็นของในห้องเก็บของ (Host folder) และถ้าคุณหยิบของออกหรือใส่ของเพิ่มในห้องเก็บของผ่านประตูนี้ ของในห้องจริงๆ ก็จะเปลี่ยนแปลงตามไปด้วย (นี่คือเหตุผลที่ลับไฟล์ใน Host และใน Container ถูกห้ามตาม)

ตอนลบ Container (docker rm): การลบ Container คือ เมื่อจบภารรือเต็นท์ทิ้ง เต็นท์และประตูวิเศษจะหายไป แต่ตัวห้องเก็บของและของข้างในที่อยู่ในบ้านของคุณยังอยู่ ครบถ้วนเหมือนเดิม ไม่ได้หายไปไหน



Docker Compose

docker compose version





docker-compose.yml

Docker Compose คือเครื่องมือที่ช่วยให้เราสามารถจัดการแอปพลิเคชันที่ประกอบด้วยหลายๆ Docker container ได้ง่ายขึ้นผ่านไฟล์ๆ เดียว

พูดง่ายๆ คือ แทนที่จะต้องมาบันทึกคำสั่ง docker run ยาวๆ พร้อม config ต่างๆ กีละ container, Docker Compose ให้เราเขียนคำสั่งกึ่งกำหนดในไฟล์ที่ชื่อว่า docker-compose.yml และสั่งรันกึ่งกำหนดได้ด้วยคำสั่งเดียว





ทำไมเราถึงต้องใช้ Docker Compose?

ลองนึกภาพว่าคุณกำลังพัฒนาเว็บแอปพลิเคชัน ซึ่งโดยทั่วไปมักจะประกอบด้วยส่วนต่างๆ อย่างน้อย 3 ส่วน:

1. **Web Server:** ตัวแอปพลิเคชันหลักของคุณ (เช่น Node.js, Python, PHP)
2. **Database:** ที่สำหรับเก็บข้อมูล (เช่น MongoDB, PostgreSQL)
3. **Cache:** ระบบเก็บข้อมูลชั่วคราวเพื่อความเร็ว (เช่น Redis)

ถ้าไม่มี Docker Compose คุณจะต้อง:

- สร้าง network เพื่อให้ container คุยกันได้
- สั่งรัน container ของ Database ด้วยคำสั่ง `docker run...`
- สั่งรัน container ของ Cache ด้วยคำสั่ง `docker run...`
- สั่งรัน container ของ Web Server ด้วยคำสั่ง `docker run...` ที่ยาวมากๆ เพราะต้องระบุ port, volume, environment variables และ network เพื่อให้เชื่อมต่อกับอีก 2 container แรกได้

ซึ่งกังวลนี้ขับข้อนและผิดพลาดได้ง่าย

แต่เมื่อใช้ Docker Compose คุณแค่กำหนดค่ากังวลนี้ไว้ในไฟล์ `docker-compose.yml` ไฟล์เดียวแล้วสั่ง `docker-compose up` ทุกอย่างก็จะทำงานพร้อมกันและเชื่อมต่อกันโดยอัตโนมัติ

DOCKER HOST

CONTAINERS

CONTAINER



WORDPRESS

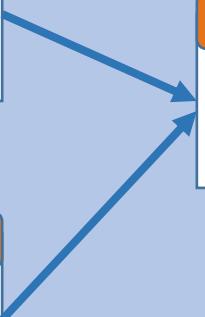
CONTAINER



CONTAINER



phpMyAdmin



```
// CREATE NETWORK
docker network create wordpress
// CREATE VOLUME
docker volume create db_data

// MYSQL
docker run --name mysql --network wordpress -v db_data:/var/lib/mysql -e
MYSQL_ROOT_PASSWORD=1111 -d mysql:5.7

// WORDPRESS
docker run -d --name wordpress --network wordpress -p 888:80 -e WORDPRESS_DB_HOST=mysql
-e WORDPRESS_DB_USER=root -e WORDPRESS_DB_PASSWORD=1111 wordpress

// PHPMYADMIN
docker run --name pma -d --network wordpress -p 8888:80 -e PMA_ARBITRARY=1
phpmyadmin/phpmyadmin
```

DOCKER HOST

CONTAINERS

CONTAINER



Express

→



CONTAINER



React

```
// CREATE NETWORK
docker network create shoppers

// MongoDB
docker run -d --name mymongodb --network shoppers mymongodb:1.0

// Node.js - API
docker run -d --name mynodejsapp -p 3000:3000 -e DATABASE_USER=admin -e
DATABASE_PASSWORD=1111 -e DATABASE_HOST=mymongodb --network shoppers mynodejsapp:1.1

// NGINX - React
docker run -d --name myreactapp_p --network shoppers -p 3001:80 myreactapp:1.1
```





หัวใจหลักคือไฟล์ `docker-compose.yml`

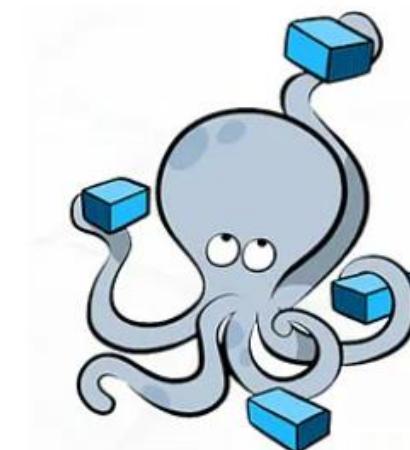
ไฟล์นี้เป็นไฟล์ที่เขียนด้วยรูปแบบ **YAML** ซึ่งอ่านง่ายเหมือนการเขียนรายการ สิ่งที่เรากำหนดในไฟล์นี้คือ:

- `services` : กำหนดว่าแอปของเรามี container อะไรบ้าง (เช่น `nodejs` , `mongodb`)
- `image` หรือ `build` : บอกว่าจะใช้ image สำเร็จรูปจากที่ไหน หรือจะให้สร้าง image ใหม่จาก `Dockerfile`
- `ports` : กำหนดการเชื่อมต่อ port ระหว่างเครื่องเรากับ container
- `volumes` : กำหนดการเชื่อมต่อโฟลเดอร์ (mount volume) เพื่อให้ข้อมูลไม่หาย
- `networks` : กำหนด network ให้ container คุยกันได้
- `environment` : กำหนดค่า environment variables ต่างๆ

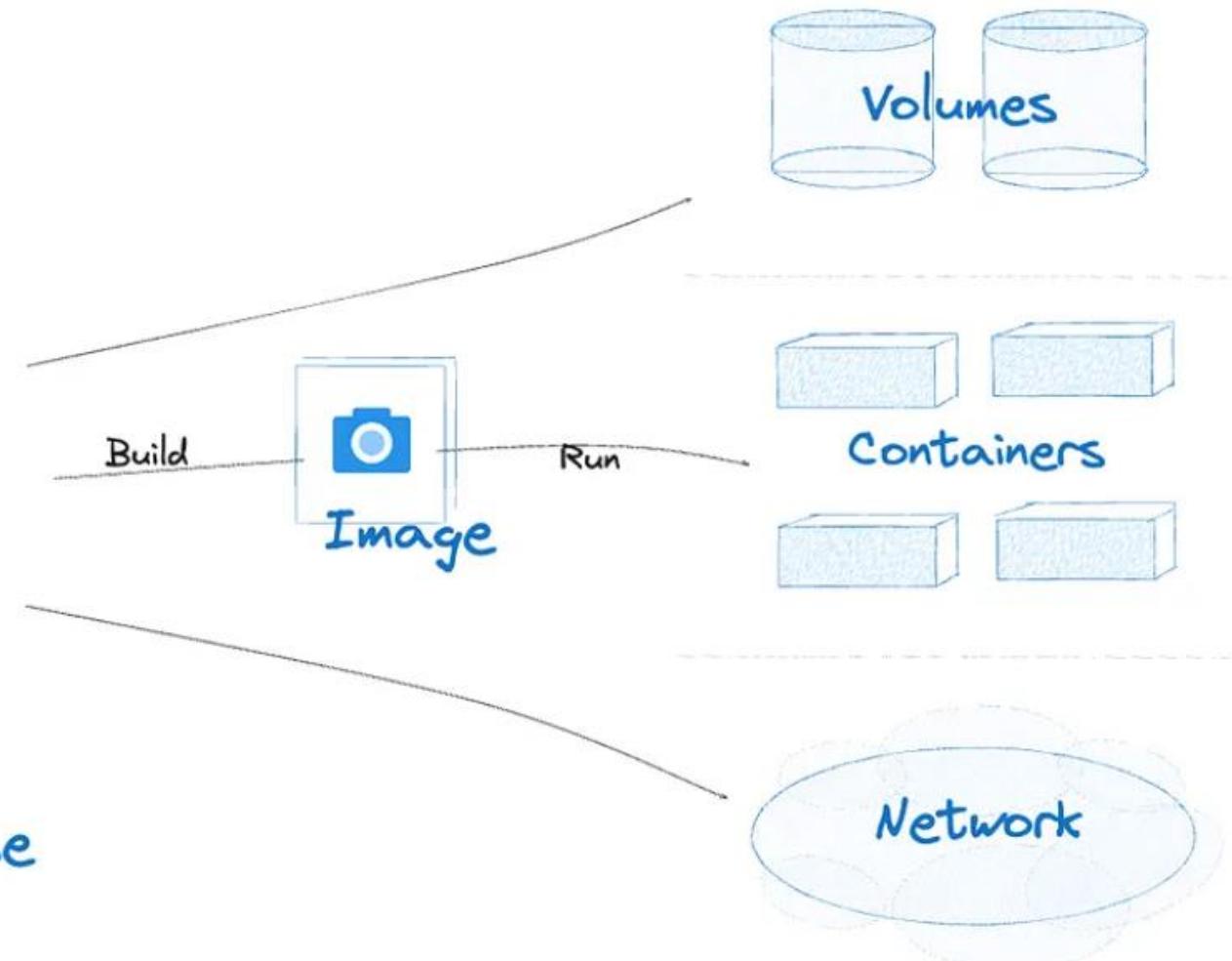
```
docker-compose.yaml
```

```
services:  
  app:  
    image: node:18-alpine  
    command: sh -c "yarn install && yarn run dev"  
    ports:  
      - 127.0.0.1:3000:3000  
    working_dir: /app  
    volumes:  
      - ./app  
    environment:  
      MYSQL_HOST: mysql  
      MYSQL_USER: root  
      MYSQL_PASSWORD: secret  
      MYSQL_DB: todos  
  
  mysql:  
    image: mysql:8.0  
    volumes:  
      - todo-mysql-data:/var/lib/mysql  
    environment:  
      MYSQL_ROOT_PASSWORD: secret  
      MYSQL_DATABASE: todos  
  
  volumes:  
    todo-mysql-data:
```

Config YAML

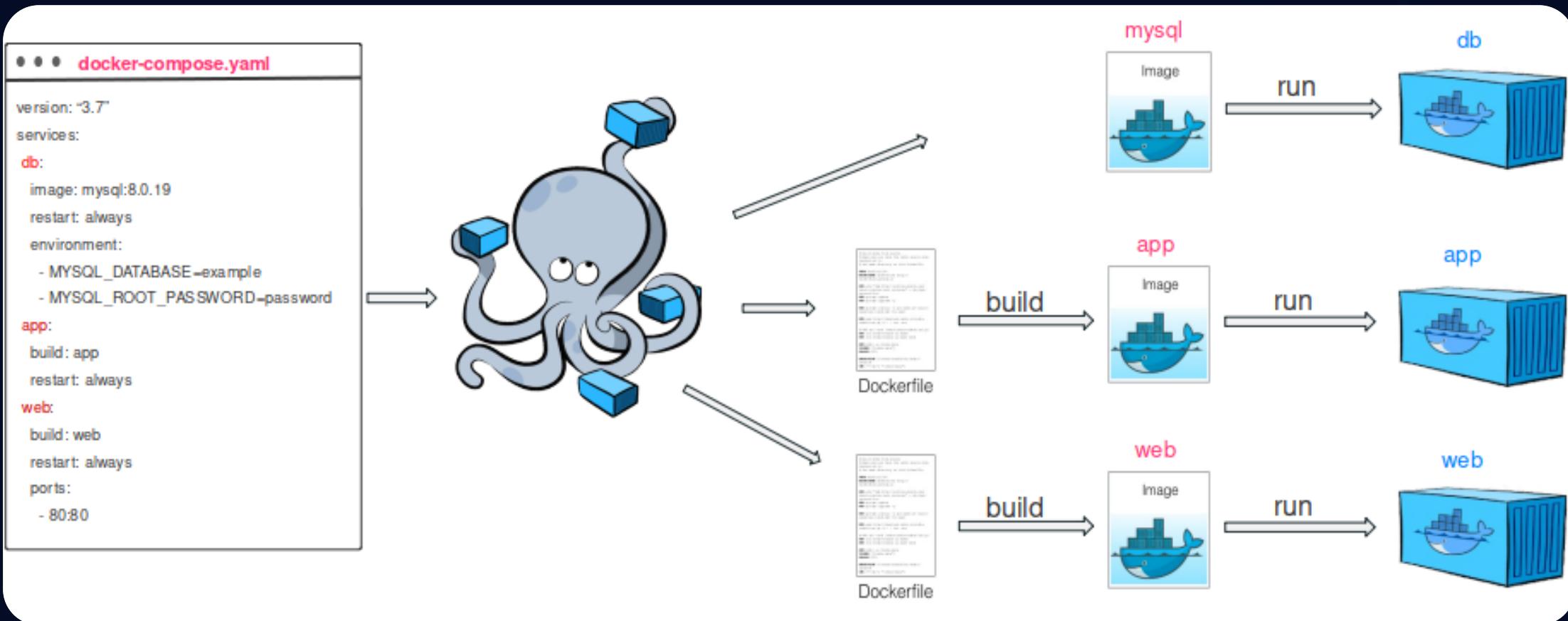


Docker Compose



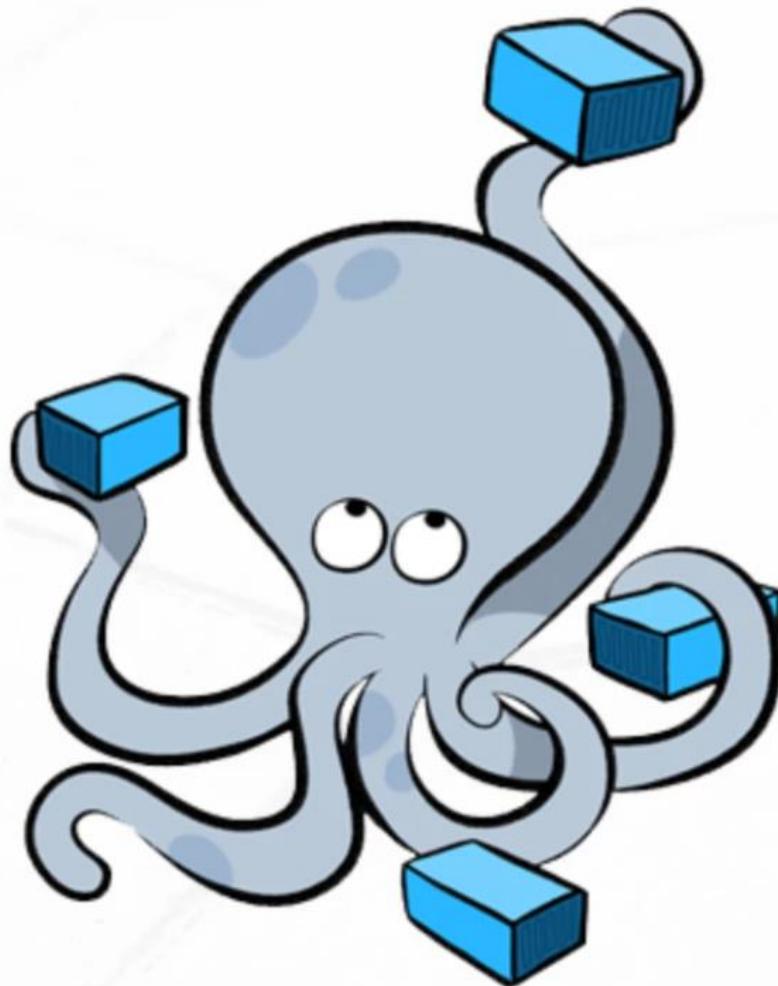
สถาบันไอทีจีเนียส

www.itgenius.co.th



Docker Compose file structure





```
php:  
  build: php  
  ports:  
    - "80:80"  
    - "443:443"  
  volumes:  
    - ./php/www:/var/www/html  
  links:  
    - db
```

```
$ docker-compose up
```



ตัวอย่างคำสั่งที่ใช้บ่อย

- `docker-compose up` : สร้างและ启動 (run) container ทั้งหมด
- `docker-compose up -d` : รัน container ทั้งหมดแบบ background (detach mode)
- `docker-compose down` : หยุดและลบ container ทั้งหมด
- `docker-compose build` : สร้าง image ใหม่ตามที่ระบุใน Dockerfile
- `docker-compose ps` : ดูสถานะของ container ทั้งหมดที่รันอยู่

โดยสรุป **Docker Compose** เป็นเครื่องมือที่จำเป็นมากสำหรับการพัฒนาแอปพลิเคชันยุคใหม่ที่ใช้สถาปัตยกรรมแบบ Microservices เพราะมันช่วยลดความซับซ้อนในการจัดการ container หลายๆ ตัว ทำให้เราโฟกัสกับการเขียนโค้ดได้มากขึ้นครับ

```
version: '3'

services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: 1111
```

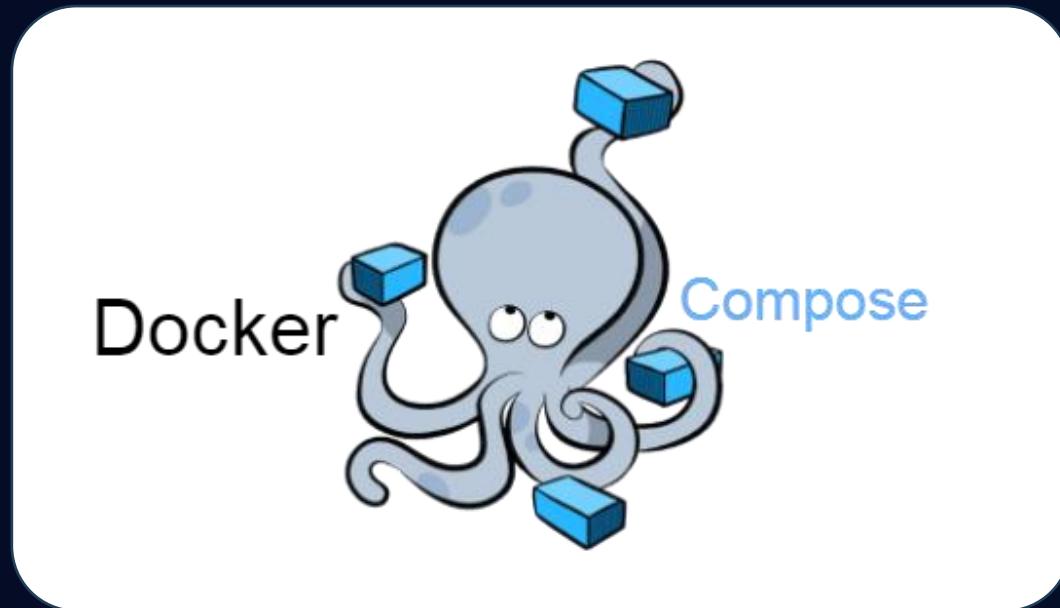
```
wordpress:
  depends_on:
    - db
  image: wordpress:latest
  ports:
    - 888:80
  restart: always
  environment:
    WORDPRESS_DB_HOST: db:3306
    WORDPRESS_DB_USER: root
    WORDPRESS_DB_PASSWORD: 1111

phpmyadmin:
  depends_on:
    - db
  image: phpmyadmin/phpmyadmin
  restart: always
  ports:
    - 8888:80
  volumes:
    db_data:

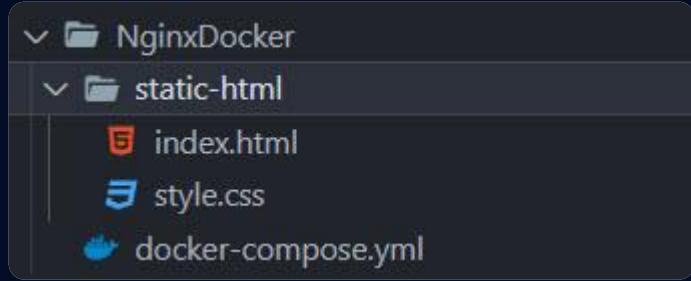
networks:
  wordpress:
```



Create docker-compose.yml for NGINX Web Server



โครงสร้างโปรเจกต์



```
index.html
1 <html lang="en">
2 <head>
3   <meta charset="UTF-8">
4   <meta http-equiv="X-UA-Compatible" content="IE=edge">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Nginx Docker</title>
7   <link rel="stylesheet" href="style.css">
8 </head>
9 <body>
10  <h1>Hello Nginx with Docker</h1>
11 </body>
12 </html>
```

```
style.css
1 body{
2   background-color: salmon;
3 }
```



Create file docker-compose.yml

```
version: '3.9'

services:
  nginx:
    container_name: nginx
    restart: always
    image: nginx:stable-alpine
    volumes:
      - ./static-html:/usr/share/nginx/html
    ports:
      - "82:80"
    networks:
      - web_network

networks:
  web_network:
    name: nginx
    driver: bridge
```



เช็คความถูกต้องของไฟล์ docker-compose.yml

```
$ docker compose
```

```
$ docker compose config -q
```



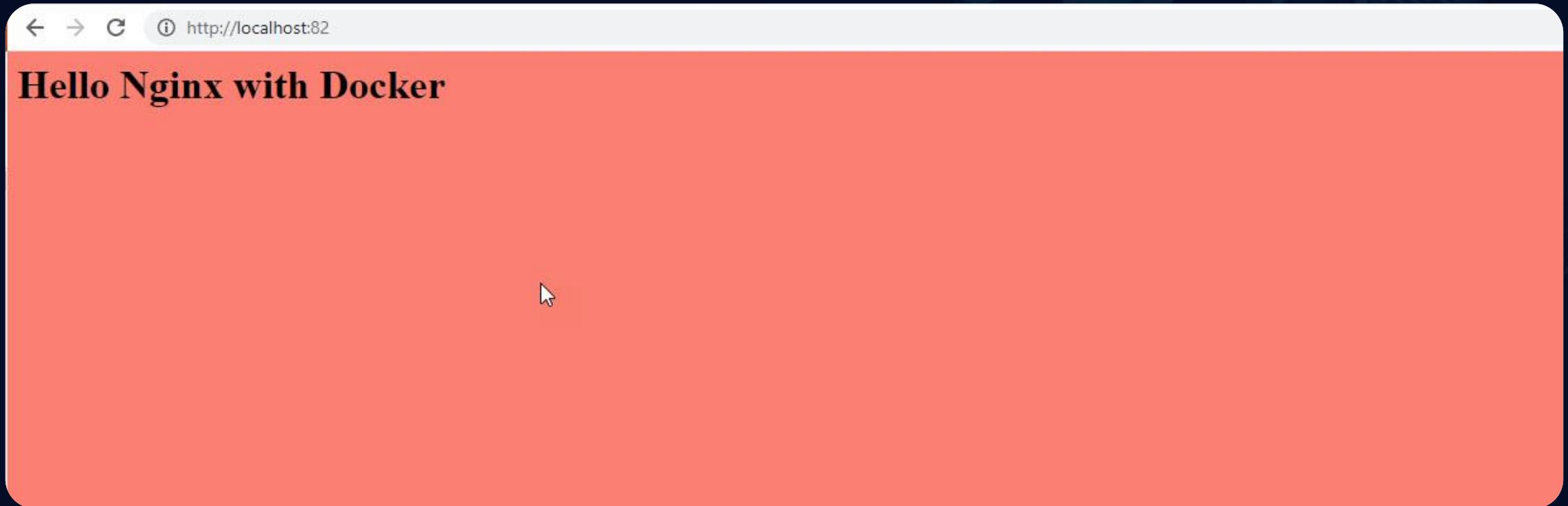
การรัน docker-compose.yml เพื่อสร้าง images และ containers

```
$ docker compose up -d
```

```
$ docker-compose up -d
Creating network "nginx" with driver "bridge"
Pulling nginx (nginx:stable-alpine)...
stable-alpine: Pulling from library/nginx
4e9f2cdf4387: Already exists
6cac039d45af: Pull complete
7bbb5e46b36d: Pull complete
39f6d17c9d38: Pull complete
a5f9fd374d3b: Pull complete
7d84628ff318: Pull complete
Digest: sha256:2012644549052fa07c43b0d19f320c871a25e105d0b23e33645e4f1bcf8fc97
Status: Downloaded newer image for nginx:stable-alpine
Creating nginx ... done
```



ทดสอบเรียกใช้งาน



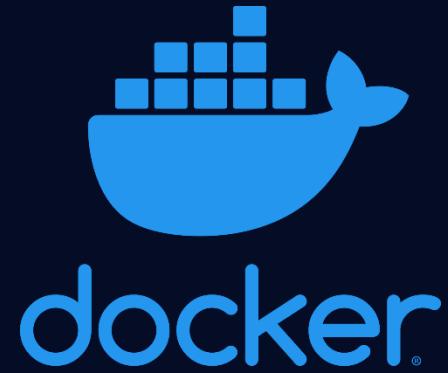
ลบ container และ image ออก

```
$ docker-compose down --rmi all
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
$ docker-compose down --rmi all
Stopping httpd ... done
Removing httpd ... done
Removing network httpd
Removing image httpd:2.4.41-alpine
```





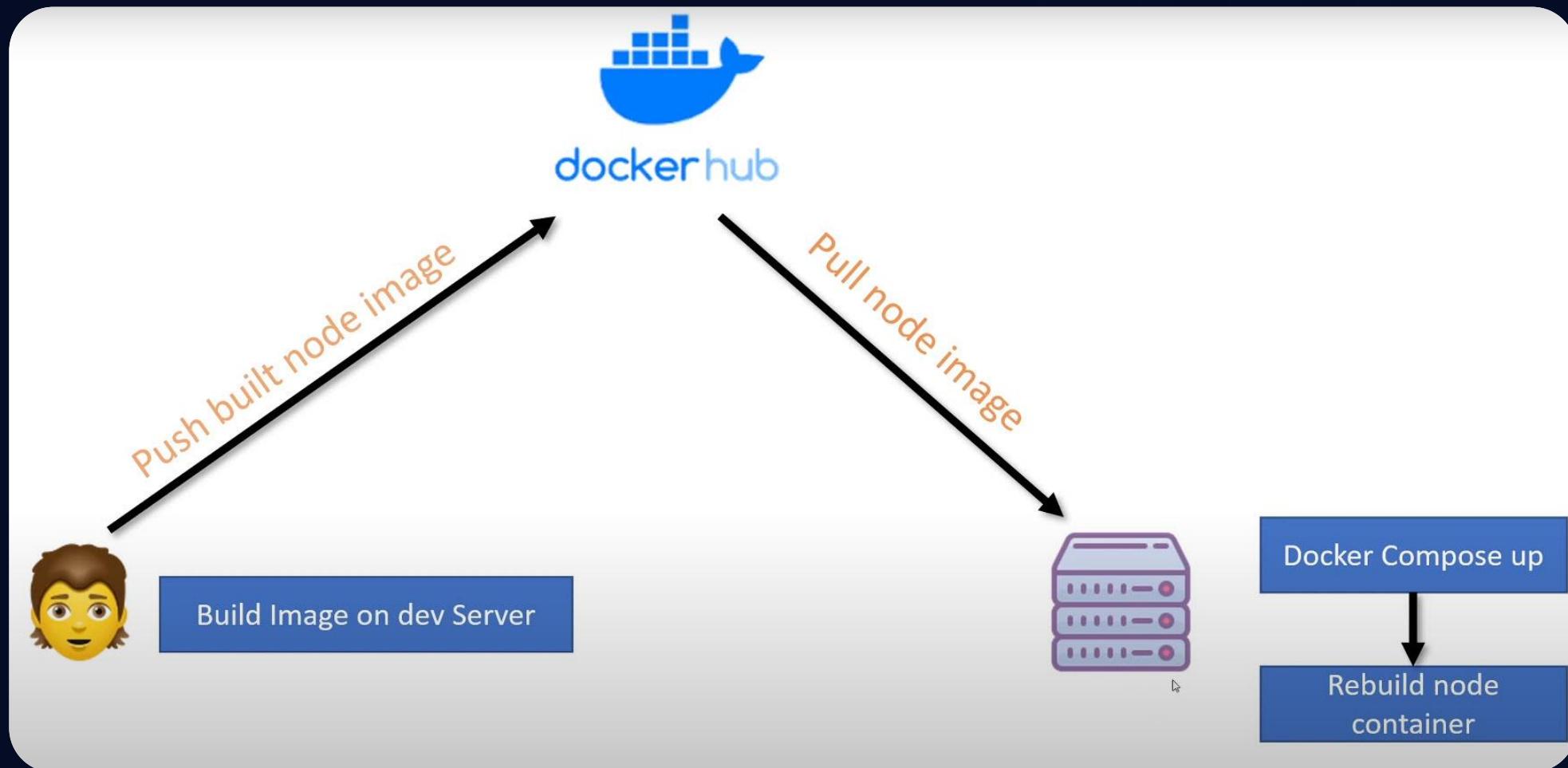
Docker Hub



สถาบันไอทีจีเนียส

www.itgenius.co.th

docker hub

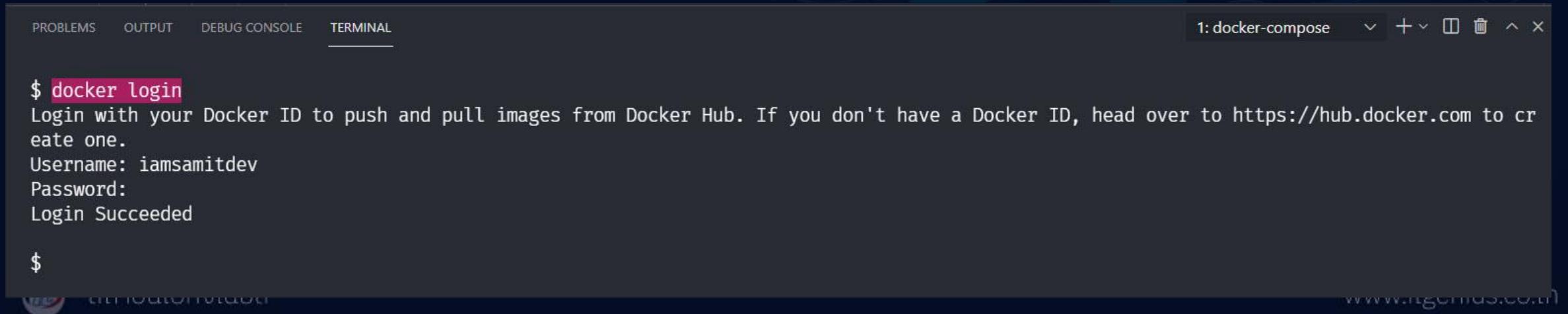


ການ Logout ອອກຈາກ docker hub

```
$ docker logout
```

ການ Login ເຂົ້າ docker hub

```
$ docker login
```



The screenshot shows a terminal window in a dark-themed IDE interface. The window title is "1: docker-compose". The terminal tabs at the top include PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL (which is underlined). The terminal content is as follows:

```
$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: iamsamitdev
Password:
Login Succeeded
$
```

At the bottom left, there is a small circular profile picture, and at the bottom right, the URL "www.w3schools.com" is visible.

Create Repository on docker hub

The screenshot shows the Docker Hub interface. At the top, there is a navigation bar with links for Explore, Repositories, Organizations, Help, and an Upgrade button. A user profile for 'iamsamitdev' is visible on the right. Below the navigation bar, a search bar allows users to search for content like 'mysql'. The main area shows a list of repositories under the user's account, including 'iamsamitdev/hello-world'. A prominent 'Create Repository' button is highlighted with a cursor. A modal window titled 'Create Repository' is open, showing the input field where 'iamsamitdev' is selected as the owner and 'mern_nodejs' is entered as the repository name. To the right of the modal, a 'Pro tip' section provides CLI instructions for pushing images to the repository. The bottom of the modal has 'Cancel' and 'Create' buttons.

dockerhub

Search for great content (e.g., mysql)

Explore Repositories Organizations Help ▾

Upgrade

iamsamitdev

Search by repository name

Create Repository

iamsamitdev / hello-world

Create Repository

iamsamitdev | mern_nodejs

Description

Pro tip

You can push a new image to this repository using the CLI

```
docker tag local-image:tagname new-repo:tagname  
docker push new-repo:tagname
```

Visibility

Using 1 of 1 private repositories. [Get more](#)

Public Appears in Docker Hub search results

Private Only visible to you

Cancel Create



Create Repository on docker hub

Advanced Image Management

View all your images and tags in this repository, clean up unused content, recover untagged images. Available for Pro and Team accounts.

[View preview](#)

iamsamitdev / mern_nodejs

This repository does not have a description [Edit](#)

Last pushed: never

Docker commands

To push a new tag to this repository,

```
docker push iamsamitdev/mern_nodejs:tagname
```

[Public View](#)

Tags and Scans

This repository is empty. When it's not empty, you'll see a list of the most recent tags here.

VULNERABILITY SCANNING - DISABLED

[Enable](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

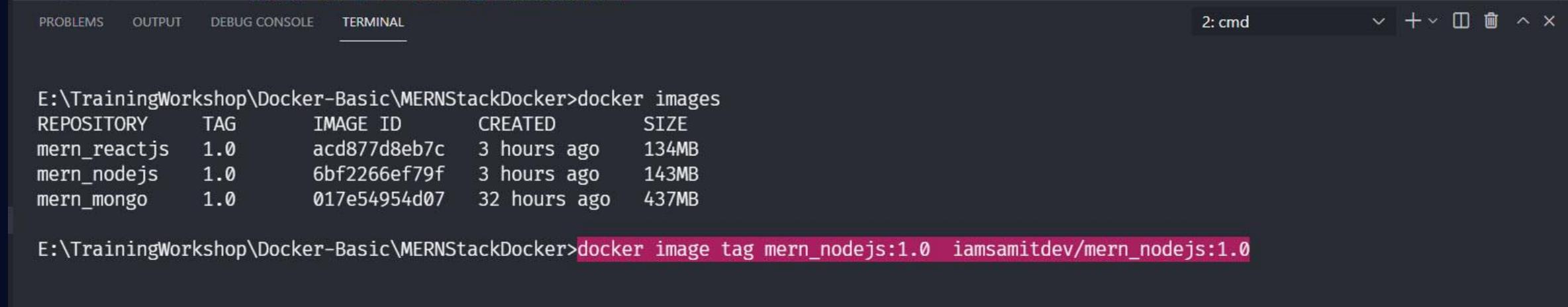
Available on Pro and Team plans.

[Upgrade to Pro](#) [Learn more](#)



Tag image to repository

```
$ docker image tag mern_nodejs:1.0 iamsamitdev/mern_nodejs:1.0
```



A screenshot of a terminal window in a dark-themed IDE. The window title is "2: cmd". The terminal tabs include PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL (which is selected). The terminal content shows the following commands and their output:

```
E:\TrainingWorkshop\ Docker-Basic\ MERNStackDocker> docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
mern_reactjs    1.0      acd877d8eb7c  3 hours ago  134MB
mern_nodejs     1.0      6bf2266ef79f  3 hours ago  143MB
mern_mongo      1.0      017e54954d07  32 hours ago 437MB

E:\TrainingWorkshop\ Docker-Basic\ MERNStackDocker> docker image tag mern_nodejs:1.0 iamsamitdev/mern_nodejs:1.0
```



Tag image to repository

```
E:\TrainingWorkshop\Docker-Basic\MERNStackDocker>docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
mern_reactjs        1.0      acd877d8eb7c   3 hours ago  134MB
iamsamitdev/mern_nodejs  1.0      6bf2266ef79f   3 hours ago  143MB
mern_nodejs         1.0      6bf2266ef79f   3 hours ago  143MB
mern_mongo          1.0      017e54954d07   33 hours ago 437MB
```

```
E:\TrainingWorkshop\Docker-Basic\MERNStackDocker>
```



Push Image to docker hub

```
$ docker image push iamsamitdev/mern_nodejs:1.0
```

The screenshot shows a terminal window in VS Code with the following content:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: cmd ▾ + ×
```

```
E:\TrainingWorkshop\ Docker-Basic\ MERNStackDocker> docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
mern_reactjs        1.0      acd877d8eb7c  3 hours ago  134MB
mern_nodejs         1.0      6bf2266ef79f  3 hours ago  143MB
iamsamitdev/mern_nodejs  1.0      6bf2266ef79f  3 hours ago  143MB
mern_mongo          1.0      017e54954d07  33 hours ago 437MB

E:\TrainingWorkshop\ Docker-Basic\ MERNStackDocker> docker image push iamsamitdev/mern_nodejs:1.0
```



Push Image to docker hub

 iamsamitdev / mern_nodejs

This repository does not have a description 

 Last pushed: a minute ago

Docker commands

To push a new tag to this repository,

```
docker push iamsamitdev/mern_nodejs:tagname
```

Tags and Scans

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
1.0		a minute ago	a minute ago

 VULNERABILITY SCANNING - DISABLED [Enable](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available on Pro and Team plans.

[Upgrade to Pro](#) [Learn more](#)

[See all](#)



Create repository for reactjs

Create Repository

iamsamitdev mern_reactjs

Description

Visibility

Using 1 of 1 private repositories. [Get more](#)

Public

Private
Appears in Docker Hub search results
Only visible to you

Pro tip

You can push a new image to this repository using the CLI

```
docker tag local-image:tagname new-repo:tagname  
docker push new-repo:tagname
```

Make sure to change *tagname* with your desired image repository tag.

[Cancel](#) [Create](#)



Create repository for reactjs

 iamsamitdev / mern_reactjs

This repository does not have a description 

 Last pushed: never

Docker commands

To push a new tag to this repository,

```
docker push iamsamitdev/mern_reactjs:tagname
```

Tags and Scans

This repository is empty. When it's not empty, you'll see a list of the most recent tags here.

 VULNERABILITY SCANNING - DISABLED
[Enable](#)

Automated Builds

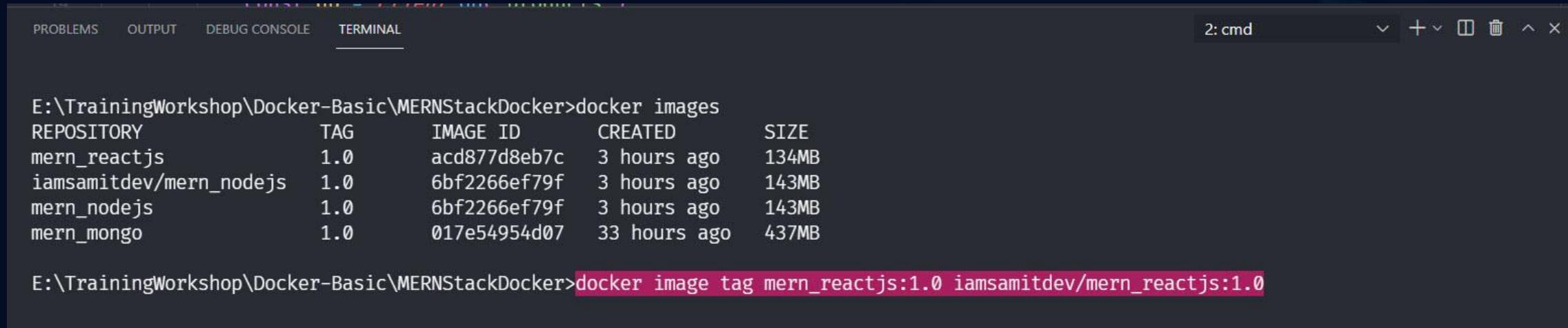
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available on Pro and Team plans.

[Upgrade to Pro](#) [Learn more](#)



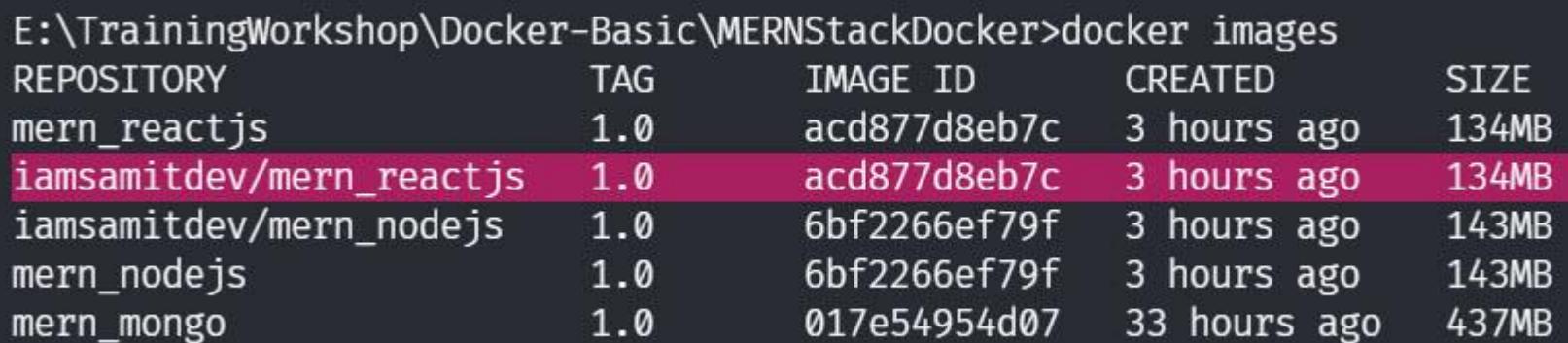
Tag image to repository



E:\TrainingWorkshop\ Docker-Basic\ MERNStackDocker> docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mern_reactjs	1.0	acd877d8eb7c	3 hours ago	134MB
iamsamitdev/mern_nodejs	1.0	6bf2266ef79f	3 hours ago	143MB
mern_nodejs	1.0	6bf2266ef79f	3 hours ago	143MB
mern_mongo	1.0	017e54954d07	33 hours ago	437MB

E:\TrainingWorkshop\ Docker-Basic\ MERNStackDocker> docker image tag mern_reactjs:1.0 iamsamitdev/mern_reactjs:1.0



E:\TrainingWorkshop\ Docker-Basic\ MERNStackDocker> docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mern_reactjs	1.0	acd877d8eb7c	3 hours ago	134MB
iamsamitdev/mern_reactjs	1.0	acd877d8eb7c	3 hours ago	134MB
iamsamitdev/mern_nodejs	1.0	6bf2266ef79f	3 hours ago	143MB
mern_nodejs	1.0	6bf2266ef79f	3 hours ago	143MB
mern_mongo	1.0	017e54954d07	33 hours ago	437MB



Push Image to docker hub

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2: cmd

```
E:\TrainingWorkshop\Docker-Basic\MERNStackDocker>docker image push iamsamitdev/mern_reactjs:1.0
```



Push Image to docker hub

 iamsamitdev / **mern_reactjs**

This repository does not have a description 

 Last pushed: a few seconds ago

Docker commands [Public View](#)

To push a new tag to this repository,

```
docker push iamsamitdev/mern_reactjs:tagname
```

Tags and Scans

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
1.0		a few seconds ago	a few second...

 VULNERABILITY SCANNING - DISABLED [Enable](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available on Pro and Team plans.

[Upgrade to Pro](#) [Learn more](#)



Edit docker-compose.yml change image from registry

```
docker-compose.yml M ×
11      - web_network
12      restart: always
13
14  nodejs:
15    depends_on:
16      - mongodb
17    # build: nodejs/
18    # image: mern_nodejs:1.0
19    image: iamsamitdev/mern_nodejs:1.0
20    container_name: mern_nodejs
21    volumes:
22      - /usr/app/node_modules
23      - ./nodejs:/usr/app
24    ports:
25      - 3000:3000
26    environment:
27      - DATABASE_USER=admin
28      - DATABASE_PASSWORD=1234
29      - DATABASE_HOST=mongodb
30    networks:
31      - web_network
32    restart: always
33
34  reactjs:
35    depends_on:
36      - mongodb
37    # build: frontend/
38    # image: mern_reactjs:1.0
39    image: iamsamitdev/mern_reactjs:1.0
40    container_name: mern_reactjs
41    volumes:
42      - /usr/app/node_modules
43      - ./frontend:/usr/app
44    networks:
45      - web_network
46    restart: always
47    ports:
```



Push to git

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    2: cmd    ▾ + ▾ □ □ ▲ ▲ ▾  
no changes added to commit (use "git add" and/or "git commit -a")  
E:\TrainingWorkshop\ Docker-Basic\ MERNStackDocker>git add .  
E:\TrainingWorkshop\ Docker-Basic\ MERNStackDocker>git commit -m "add image file from registry"  
[main 89e9fb4] add image file from registry  
 1 file changed, 4 insertions(+), 2 deletions(-)  
E:\TrainingWorkshop\ Docker-Basic\ MERNStackDocker>git push --all  
Enumerating objects: 5, done.  
Counting objects: 100% (5/5), done.  
Delta compression using up to 12 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 360 bytes | 360.00 KiB/s, done.  
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.  
To https://github.com/iamsamitdev/MERNDockerDeploy.git  
 575fd48..89e9fb4 main -> main
```



Git Pull in production

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

root@ubuntu-docker-lab:~/app# ls
docker-compose.yml  frontend  mongodb  nodejs
root@ubuntu-docker-lab:~/app# git pull
Username for 'https://github.com': iamsamitdev
Password for 'https://iamsamitdev@github.com':
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0
Unpacking objects: 100% (3/3), 340 bytes | 170.00 KiB/s, done.
From https://github.com/iamsamitdev/MERNDockerDeploy
      575fd48..89e9fb4  main      -> origin/main
Updating 575fd48..89e9fb4
Fast-forward
  docker-compose.yml | 6 +-----
   1 file changed, 4 insertions(+), 2 deletions(-)
root@ubuntu-docker-lab:~/app# |
```



Then docker-compose up

```
root@ubuntu-docker-lab:~/app# clear
root@ubuntu-docker-lab:~/app# docker-compose up -d
Building nodejs
Sending build context to Docker daemon 69.12kB
Step 1/6 : FROM node:lts-alpine
--> fe39f43f1d22
Step 2/6 : WORKDIR /usr/app
--> Using cache
--> f7f38754e9fa
Step 3/6 : COPY ./package.json .
--> Using cache
--> 80fc9e622380
Step 4/6 : RUN npm install
--> Using cache
--> 5db98d3ff573
Step 5/6 : COPY . .
--> Using cache
--> 37f04c4b7762
Step 6/6 : CMD [ "npm", "start" ]
--> Using cache
--> a7f1f9131faa
```



Then docker-compose up

```
----> Using cache
----> f58ff6eaee12
Step 7/10 : CMD ["npm", "start"]
----> Using cache
----> 314dbbdef07b
Step 8/10 : FROM nginx
----> 87a94228f133
Step 9/10 : EXPOSE 80
----> Using cache
----> 55a5451dde77
Step 10/10 : COPY --from=builder /usr/app/build /usr/share/nginx/html
----> Using cache
----> 95da88af78d6
Successfully built 95da88af78d6
Successfully tagged iamsamitdev/mern_reactjs:1.0
WARNING: Image for service reactjs was built because it did not already exist. To rebuild this image, run `docker-compose up --build`.
mern_mongo is up-to-date
Recreating mern_nodejs ... done
Recreating mern_reactjs ... done
root@ubuntu-docker-lab:~/app#
```



Docker images

PROBLEMS

OUTPUT

DEBUG CONSOLE

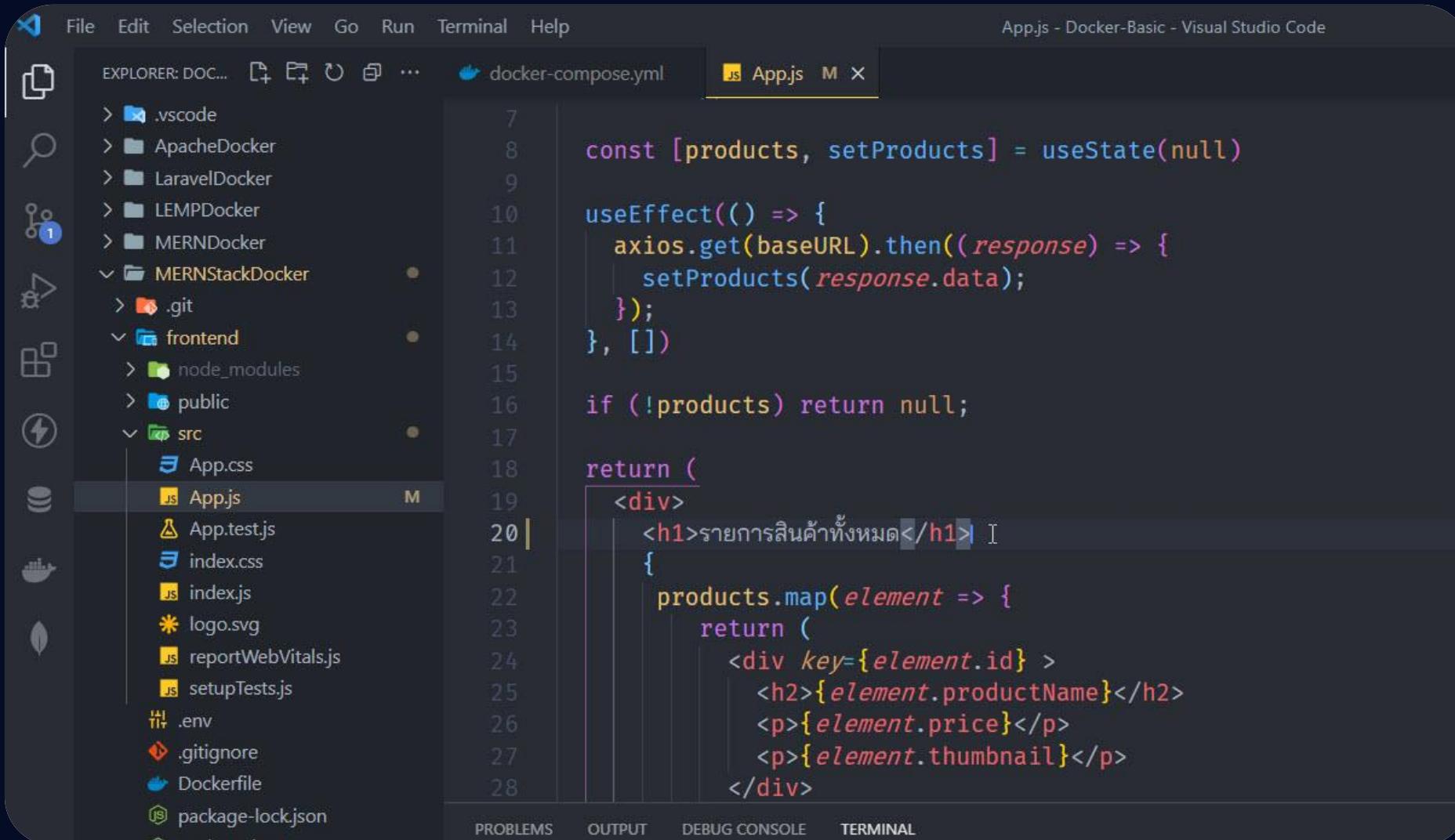
TERMINAL

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
iamsamitdev/mern_reactjs	1.0	95da88af78d6	3 hours ago	134MB
mern_reactjs	1.0	95da88af78d6	3 hours ago	134MB
iamsamitdev/mern_nodejs	1.0	a7f1f9131faa	5 hours ago	143MB
mern_nodejs	1.0	a7f1f9131faa	5 hours ago	143MB
mern_mongo	1.0	d29222708e58	5 hours ago	437MB
node	lts-alpine	fe39f43f1d22	6 days ago	118MB
node	alpine	0f877e6f48f8	6 days ago	110MB
nginx	latest	87a94228f133	7 days ago	133MB
mongo	4.4.9	b6e8852ad84e	2 weeks ago	437MB

```
root@ubuntu-docker-lab:~/app# |
```



Change something in local



The screenshot shows a dark-themed instance of Visual Studio Code. The title bar reads "App.js - Docker-Basic - Visual Studio Code". The left sidebar contains a file tree with several projects like ".vscode", "ApacheDocker", "LaravelDocker", "LEMPDocker", "MERNDocker", and "MERNStackDocker". Under "MERNStackDocker", there's a ".git" folder and a "frontend" directory containing "node_modules", "public", and "src". Inside "src", files like "App.css", "App.test.js", "index.css", "index.js", "logo.svg", "reportWebVitals.js", "setupTests.js", ".env", ".gitignore", "Dockerfile", and "package-lock.json" are listed. The "App.js" file is currently selected in the tree and is open in the main editor area. The code in "App.js" is a functional component that uses useState and useEffect hooks to fetch products from a baseURL and map them to a list of items. A cursor is visible in the code, highlighting the text "รายการสินค้าทั้งหมด". The bottom navigation bar includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL.

```
const [products, setProducts] = useState(null)

useEffect(() => {
  axios.get(baseURL).then((response) => {
    setProducts(response.data);
  });
}, []);

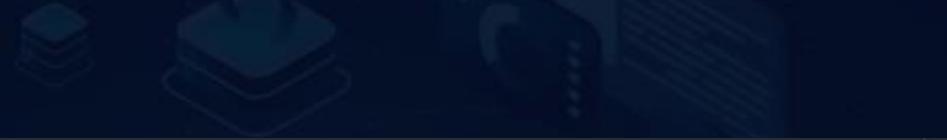
if (!products) return null;

return (
  <div>
    <h1>รายการสินค้าทั้งหมด</h1>
    {
      products.map(element => {
        return (
          <div key={element.id} >
            <h2>{element.productName}</h2>
            <p>{element.price}</p>
            <p>{element.thumbnail}</p>
          </div>
        )
      })
    }
  </div>
)
```



Rebuild images iamsamitdev/mern_nodejs

```
$ docker-compose build reactjs
```



A screenshot of a terminal window in a dark-themed IDE. The terminal tab is active, showing the command `docker-compose build reactjs`. The output shows the build process for the `reactjs` service:

```
E:\TrainingWorkshop\ Docker-Basic\ MERNStackDocker> docker-compose build reactjs
Building reactjs
[+] Building 16.3s (16/16) FINISHED
=> [internal] load build definition from Dockerfile                               0.2s
=> => transferring dockerfile: 32B                                              0.0s
=> [internal] load .dockerignore                                               0.3s
=> => transferring context: 2B                                                 0.0s
=> [internal] load metadata for docker.io/library/nginx:latest                3.5s
=> [internal] load metadata for docker.io/library/node:alpine                 3.6s
=> [auth] library/nginx:pull token for registry-1.docker.io                   0.0s
```



Docker images

```
$ docker images
```

```
E:\TrainingWorkshop\ Docker-Basic\ MERNStackDocker> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
iamsamitdev/mern_reactjs	1.0	7ec928c93ce3	22 seconds ago	134MB
mern_reactjs	1.0	acd877d8eb7c	4 hours ago	134MB
iamsamitdev/mern_reactjs	<none>	acd877d8eb7c	4 hours ago	134MB
mern_nodejs	1.0	6bf2266ef79f	4 hours ago	143MB
iamsamitdev/mern_nodejs	1.0	6bf2266ef79f	4 hours ago	143MB
mern_mongo	1.0	017e54954d07	33 hours ago	437MB



docker-compose push to hub

```
$ docker-compose push reactjs
```

```
E:\TrainingWorkshop\Docker-Basic\MERNStackDocker>docker-compose push reactjs
Pushing reactjs (iamsamitdev/mern_reactjs:1.0)...
The push refers to repository [docker.io/iamsamitdev/mern_reactjs]
577fd8dadbc6: Pushed
9959a332cf6e: Layer already exists
f7e00b807643: Layer already exists
f8e880dfc4ef: Layer already exists
788e89a4d186: Layer already exists
43f4e41372e4: Layer already exists
e81bff2725db: Layer already exists
1.0: digest: sha256:07254fc41596211e8b68d0b71f13cf0f0ef94176a870eb497154935693dca63a size: 1780
```

```
E:\TrainingWorkshop\Docker-Basic\MERNStackDocker>
```



Then docker-compose pull on production server

```
$ docker-compose pull reactjs
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
root@ubuntu-docker-lab:~/app# ls
docker-compose.yml  frontend  mongodb  nodejs
root@ubuntu-docker-lab:~/app# docker-compose pull
Pulling mongodb ... done
Pulling nodejs   ... done
Pulling reactjs ... done
WARNING: Some service image(s) must be built from source by running:
  docker-compose build mongodb
root@ubuntu-docker-lab:~/app# |
```



Then docker-compose up -d on production server

```
$ docker-compose up -d --no-deps reactjs
```

```
root@ubuntu-docker-lab:~/app# docker-compose up -d
mern_mongo is up-to-date
Recreating mern_nodejs ... done
Recreating mern_reactjs ... done
root@ubuntu-docker-lab:~/app#
```

```
root@ubuntu-docker-lab:~/app# docker images
```

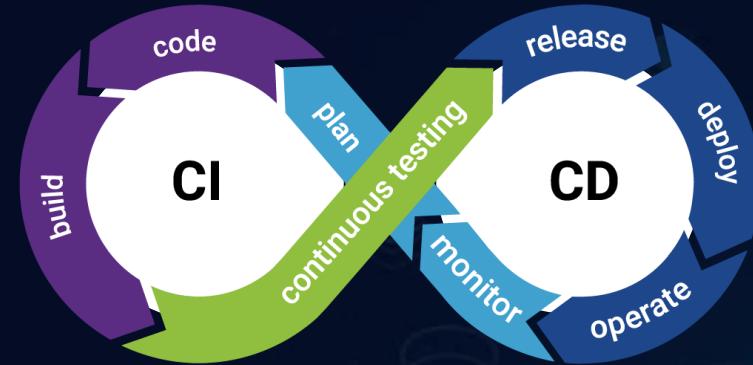
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
iamsamitdev/mern_reactjs	1.0	7ec928c93ce3	11 minutes ago	134MB
mern_reactjs	1.0	95da88af78d6	4 hours ago	134MB
iamsamitdev/mern_nodejs	1.0	6bf2266ef79f	4 hours ago	143MB
mern_nodejs	1.0	a7f1f9131faa	5 hours ago	143MB
mern_mongo	1.0	d29222708e58	5 hours ago	437MB
node	lts-alpine	fe39f43f1d22	6 days ago	118MB
node	alpine	0f877e6f48f8	6 days ago	110MB
nginx	latest	87a94228f133	7 days ago	133MB
mongo	4.4.9	b6e8852ad84e	2 weeks ago	437MB

Then docker-compose ps

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

root@ubuntu-docker-lab:~/app# docker-compose ps
      Name           Command       State        Ports
-----  
mern_mongo   docker-entrypoint.sh mongod   Up          27017/tcp
mern_nodejs   docker-entrypoint.sh npm start Up          0.0.0.0:3000->3000/tcp,:::3000->3000/tcp
mern_reactjs  /docker-entrypoint.sh ngin ... Up          0.0.0.0:80->80/tcp,:::80->80/tcp
root@ubuntu-docker-lab:~/app#
```





Basic CI/CD



What is CI/CD ?

CI = Continuous Integration (การบูรณาการอย่างต่อเนื่อง)

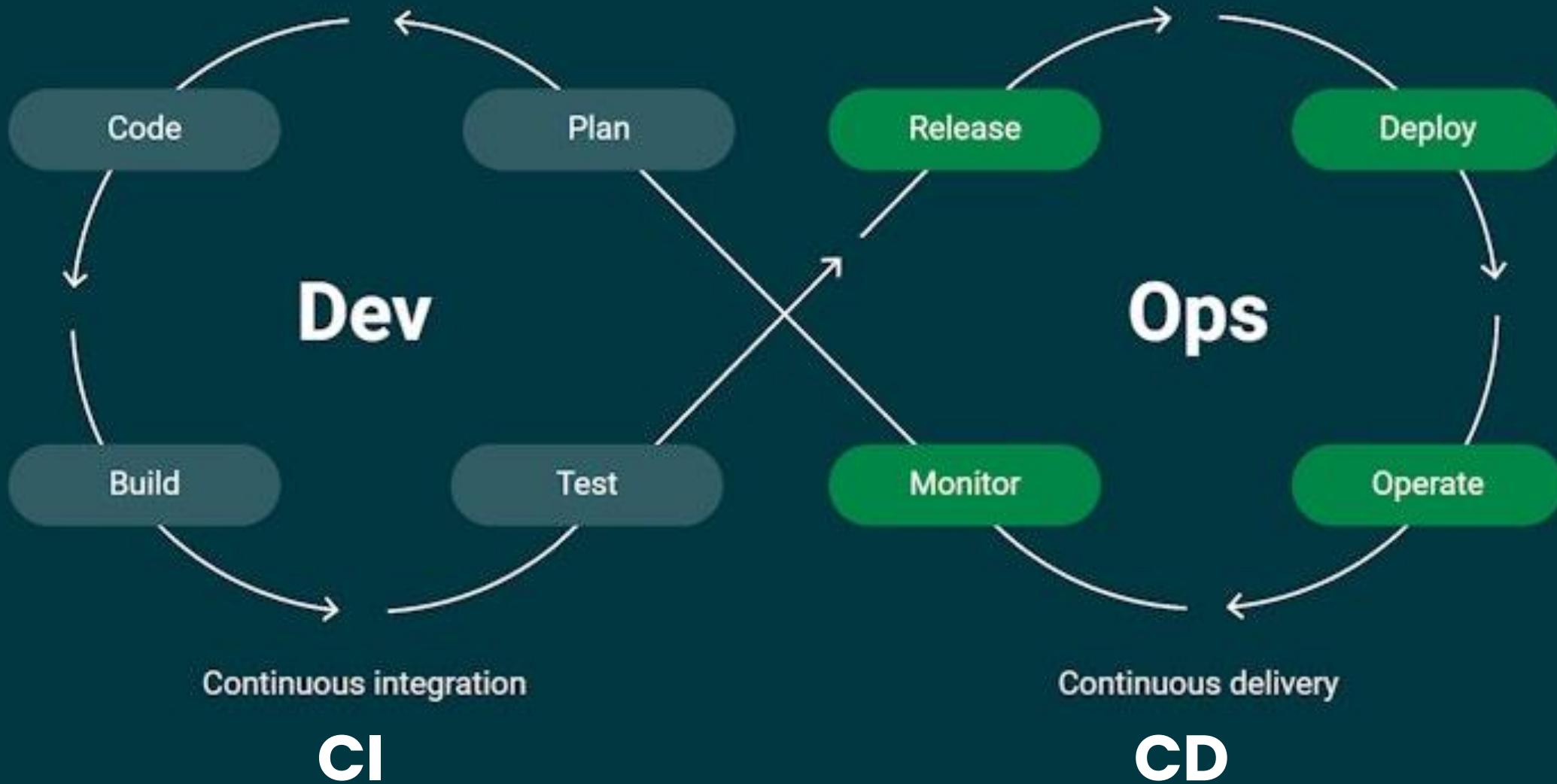
หมายถึง แนวปฏิบัติที่ให้ผู้พัฒนาทำการ **ผสม (merge / commit)** โค้ดเข้ากับสาขาหลัก (main branch / master / trunk) ป่อย ๆ และมีการ “build + ทดสอบอัตโนมัติ” ทุกครั้งที่มีการเปลี่ยนแปลง เพื่อให้ข้อผิดพลาดถูกจับได้เร็ว ลดการชนกันของโค้ด

CD = Continuous Delivery / Continuous Deployment (การจัดส่ง / การปรับใช้อย่างต่อเนื่อง)

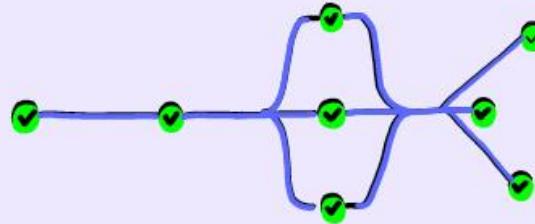
- **Continuous Delivery:** หลังจากผ่านการทดสอบใน CI แล้ว โค้ดสามารถถูก deploy ได้เสมอในสภาพ ready-to-deploy โดยอาจมีจุดที่ต้องมีคานอบบุมัติค่อนปล่อยขึ้น production
- **Continuous Deployment:** ขยายจาก Delivery โดยไม่มีจุดหยุดกลาง – ถ้าโค้ดผ่านทุกการทดสอบแล้ว จะปล่อยขึ้น production โดยอัตโนมัติเลย



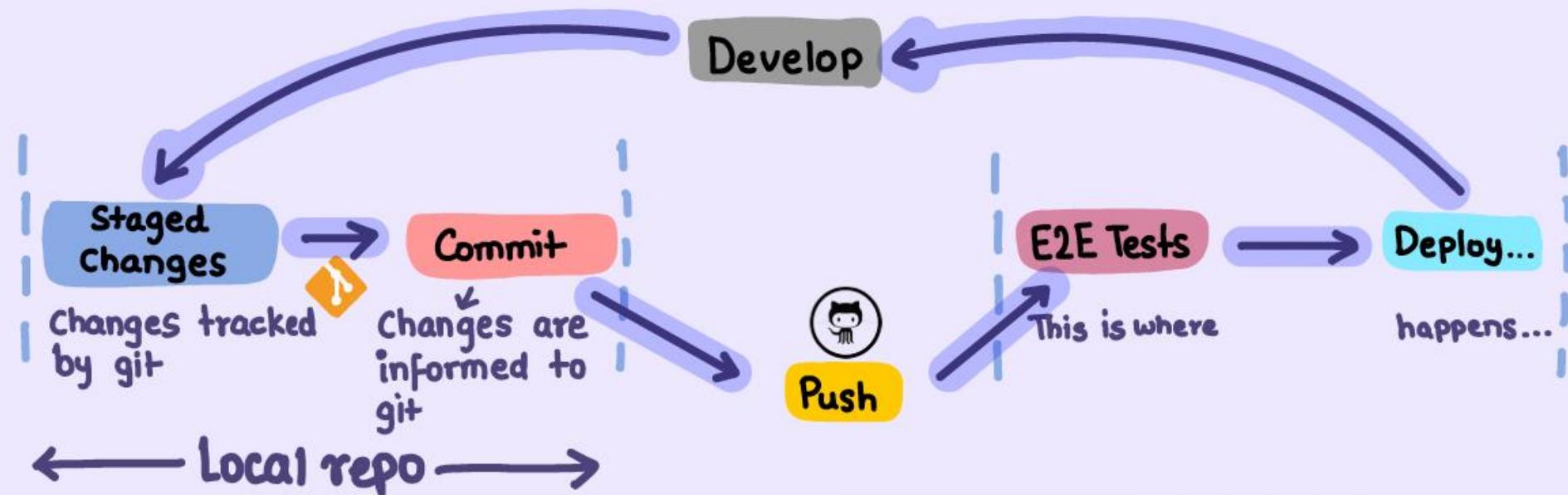
What is CI/CD ?



CI/CD Pipelines



1 Software development Lifecycle



2

← Continuous Integration → CI/CD ← Continuous delivery →

* Build and test if code in commit works well with other components

\$ git clone repo

\$ > -

Checkout
repo

Build
\$ make build

unit test
\$ make install

integration
test

Test on

* If everything is okay so far deploy & package

deploy

package

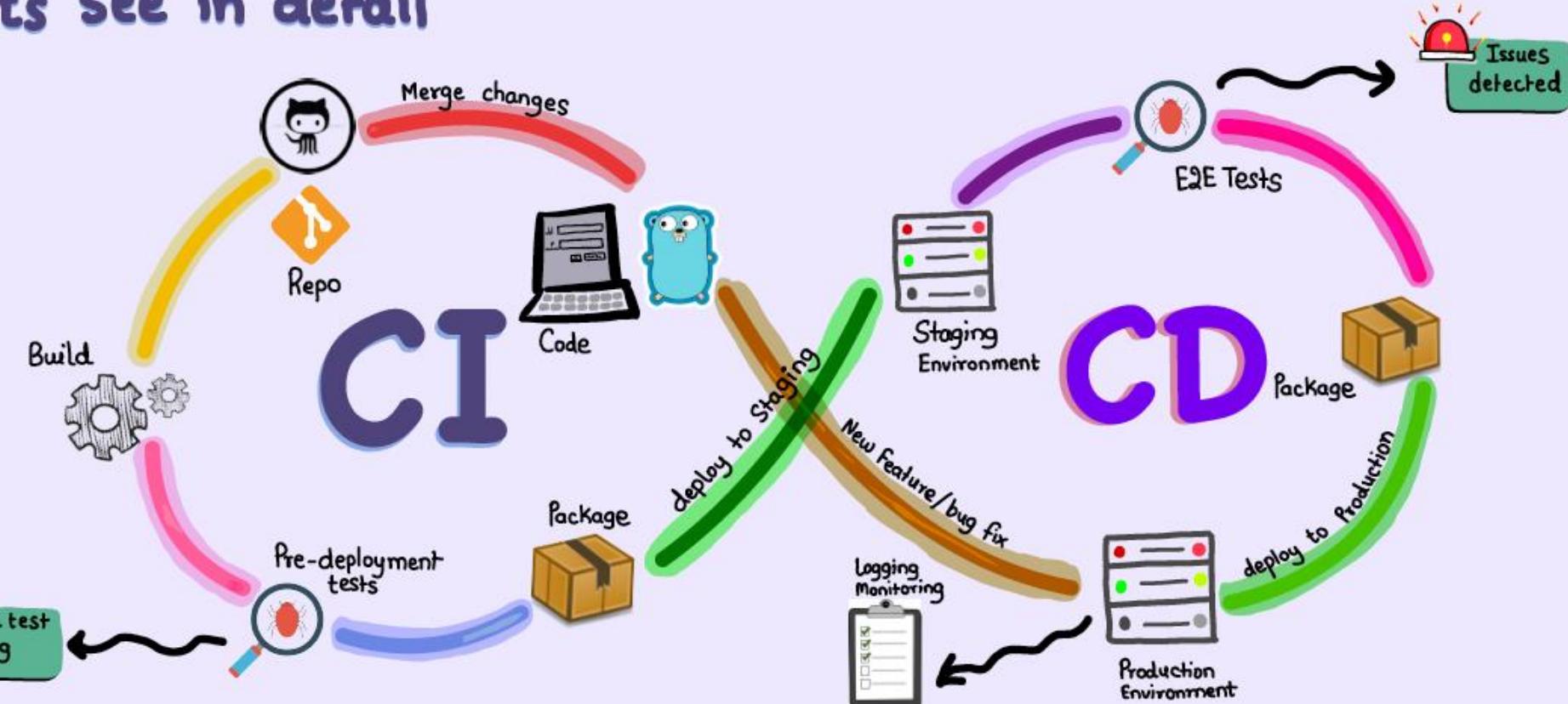
* This makes packaging & deployment match the speed of development



* Every step is triggered one after the other or in parallel, thus called as pipelines



3 Let's see in detail



SecurityZines.com In Collaboration with  ByteByteGo

Designed With
Love By @Sec_70



สถาบันไอทีจีเนียส

www.itgenius.co.th

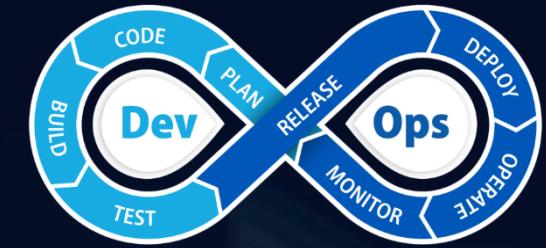
ข้อดี / ประโยชน์ของ CI/CD



- ตรวจจับปัญหาเร็ว (early detection) – เมื่อโค้ดใหม่ถูกรวมเข้ามา จะมีการทดสอบอัตโนมัติกันที่
- ลดความเสี่ยงจากการ deploy – เพราะ pipeline ถูกทดสอบและมาตรฐาน
- ความเร็วในการส่งมอบซอฟต์แวร์ (faster release cycles)
- ความคงที่และความน่าเชื่อถือในการ deploy
- feedback loop ดีขึ้น – นักพัฒนาได้รับผลตอบกลับเร็ว → ปรับปรุงได้กันเวลา
- ลดงาน manual ซ้ำ ๆ – อัตโนมัติตั้งแต่ build → test → deploy

หลักการ / แนวทางปฏิบัติที่ดี (Best Practices)

1. Commit ป้อย ๆ — อย่าเก็บโค้ดหลายวันแล้วรวมครั้งใหญ่
2. Pipeline รันเร็วที่สุด — ลดเวลา build / test ให้น้อยที่สุด
3. แยก test ชนิดต่าง ๆ — unit tests, integration tests, smoke tests
4. Infrastructure as Code (IaC) — การจัดการโครงสร้างพื้นฐานผ่านโค้ด
5. Version everything — โค้ด, config, script, database migrations — ควบคุม version control กั้งหมัด
6. ให้คุณภาพอยู่ใน pipeline — static analysis, security scan ใน pipeline
7. Rollback / Roll-forward strategy — ถ้าปล่อยแล้วมีปัญหา ให้กลับคืนสถานะได้อย่างปลอดภัย
8. แยก environments ชัดเจน — dev / staging / production
9. ใช้ pipeline as code — เช่น YAML / DSL สำหรับ config pipeline
10. ปรับปรุง pipeline อย่างต่อเนื่อง (Continuous Improvement)





Basic Jenkins



What is Jenkins ?

Jenkins คือ เซิร์ฟเวอร์อัตโนมัติ (Automation Server) และ^{เป็น}โอเพนซอร์สที่ได้รับความนิยมมากที่สุดในโลก



ถ้าจะให้อธิบายง่ายๆ Jenkins เปรียบเสมือน "พ่อบ้าน" หรือ "ผู้ควบคุมสายพานการผลิต" สำหรับวงการพัฒนาซอฟต์แวร์ หน้าที่หลักของมันคือการทำงานช้าๆ ทีบ่่าเบื่อและผิดพลาดง่ายให้เป็น "อัตโนมัติ" ก็งหมด เพื่อให้โปรแกรมเมอร์มีเวลาไปโฟกัสกับการเขียนโค้ดมากขึ้น

Jenkins ทำอะไรได้บ้าง?

ลองนึกภาพขั้นตอนการพัฒนาซอฟต์แวร์แบบปกติ:

1. เขียนโคด (Code): โปรแกรมเมอร์เขียนโค้ดบนเครื่องของตัวเอง
2. ส่งโคด (Commit & Push): นำโคดที่เขียนเสร็จแล้วไปเก็บไว้ที่ส่วนกลาง (เช่น GitHub, GitLab)
3. ทดสอบ (Test): ทดสอบว่าโคดที่เขียนไปไม่ทำให้ล่วงอื่นๆ พัง
4. สร้างโปรแกรม (Build): แปลงโคดให้กลายเป็นโปรแกรมที่พร้อมใช้งาน
5. นำไปใช้งาน (Deploy): นำโปรแกรมที่สร้างเสร็จไปติดตั้งบน Server เพื่อให้ผู้ใช้งานเข้ามาใช้ได้

Jenkins จะเข้ามายกท่าน้ำที่ในข้อ 2-5 ทั้งหมดโดยอัตโนมัติ เราเรียกระบวนการนี้ว่า **CI/CD (Continuous Integration / Continuous Delivery)**



ตัวอย่างการทำงานของ Jenkins



1. **จับตากู้โค้ด:** Jenkins จะคอย "เฝ้ามอง" แหล่งเก็บโค้ด (เช่น GitHub) ตลอดเวลา
2. **เริ่มทำงานอัตโนมัติ:** กันที่ที่โปรแกรมเมอร์ส่งโค้ดใหม่เข้ามา Jenkins จะรู้ตัวและเริ่มทำงานกันทันที
3. **Build & Test:** มันจะดึงโค้ดล่าสุดมาทำการ Build (สร้าง) และ Test (ทดสอบ) ตามที่เราตั้งค่าไว้
4. **แจ้งผล:**
 - **ถ้าทดสอบไม่ผ่าน:** Jenkins จะรีบส่งอีเมลหรือแจ้งเตือนผ่าน Slack ไปบอกทีมกันว่า "โค้ดล่าสุดที่ส่งมาทำให้โปรแกรมพังนะ!" ทำให้ทีมแก้ไขปัญหาได้อย่างรวดเร็ว
 - **ถ้าทดสอบผ่าน:** Jenkins จะทำงานขั้นต่อไป คือการนำโปรแกรม (Deploy) ขึ้น Server ให้โดยอัตโนมัติ
5. **ทำซ้ำ:** Jenkins จะรอการส่งโค้ดครั้งต่อไปและทำซ้ำแบบนี้ไปเรื่อยๆ

ทำไมถึงต้องใช้ Jenkins ?



- ลดความผิดพลาดของมนุษย์ (Human Error): การทำงานแบบอัตโนมัติมีความแม่นยำสูงกว่ามนุษย์
- ทำงานได้เร็วขึ้น: ช่วยให้เราสามารถปล่อยซอฟต์แวร์เวอร์ชันใหม่ๆ ให้ผู้ใช้งานได้เร็วและบ่อยขึ้น
- ตรวจสอบปัญหาได้ไว: เมื่อโค้ดมีปัญหา ก็จะรู้ตัวแทนทันที ไม่ต้องรอไปเจอตอนกা�耶
- ปรับแต่งได้หลากหลาย: Jenkins มีสิ่งที่เรียกว่า "Plugins" เป็นพันๆ ตัว ทำให้เราสามารถปรับแต่งให้มันทำงานเชื่อมต่อกับเครื่องมืออื่นๆ ได้แทนทุกชนิด

โดยสรุป Jenkins คือหัวใจสำคัญของกระบวนการพัฒนาซอฟต์แวร์สมัยใหม่ (DevOps) ที่ช่วยให้การสร้าง, ทดสอบ, และส่งมอบซอฟต์แวร์เป็นไปอย่างรวดเร็ว มีประสิทธิภาพ และเชื่อถือได้ครับ

ข้อดีของ Jenkins

- เป็นเครื่องมือโอเพนซอร์ส (open source) ที่มีชุมชนผู้ใช้งานขนาดใหญ่ ค่อยช่วยเหลือ
- ติดตั้งง่าย
- มีปลั๊กอินมากกว่า 1,000+ ตัวเพื่อช่วยให้งานของคุณง่ายขึ้น และถ้าปลั๊กอินที่ต้องการยังไม่มี คนสามารถเขียนขึ้นมาเองและแบ่งปันให้กับชุมชนได้
- ใช้งานได้ฟรี ไม่มีค่าใช้จ่าย
- สร้างขึ้นด้วยภาษา Java จึงทำให้สามารถนำไปใช้งานได้บนทุกแพลตฟอร์มหลักๆ





Setup Jenkins

หลักๆ ทำได้ 2 วิธี

1. Bare Metal Installation (ติดตั้งบน OS โดยตรง)
2. Containerized Deployment (ติดตั้งผ่าน Container)



1. Bare Metal Installation (ติดตั้งบน OS โดยตรง)

เป็นการติดตั้ง Jenkins ลงบนระบบปฏิบัติการของ Server (เช่น Ubuntu, CentOS, Windows) โดยตรง เหมือนกับการติดตั้งโปรแกรมทั่วไป วิธีนี้ Jenkins จะใช้ทรัพยากรของเครื่อง Server ร่วมกับโปรแกรมอื่นๆ ทั้งหมด

ข้อดี

- ประสิทธิภาพสูงสุด:** Jenkins สามารถเข้าถึงทรัพยากร Hardware (CPU, RAM, Disk) ได้โดยตรงโดยไม่มีขั้นของ Virtualization มาคั่น ทำให้ได้ประสิทธิภาพสูงสุดเท่าที่เครื่องจะทำได้
- เข้าถึง Hardware ง่าย:** เหมาะสำหรับงานที่ต้องใช้อุปกรณ์ต่อพ่วงพิเศษ หรือต้องการการตั้งค่าระบบในระดับลึกที่ Container ทำได้ลำบาก
- เรียนรู้ง่ายสำหรับผู้เริ่มต้น:** หากคุณเคยกับการติดตั้ง Server แบบเดิม วิธีนี้จะตรงไปตรงมาและเข้าใจง่ายในช่วงแรก

ข้อเสีย

- จัดการยาก (Dependency Hell):** Jenkins และ Plugins ต่างๆ อาจต้องการ Java หรือ Library เวอร์ชันที่แตกต่างกัน ซึ่งอาจขัดแย้งกับโปรแกรมอื่นที่รันอยู่บน Server เดียวกัน การอัปเดต OS หรือ Library อาจทำให้ Jenkins พังได้
- ยั่วยวนระบบลำบาก:** การย้าย Jenkins ไปยัง Server ใหม่เป็นเรื่องใหญ่ ต้องติดตั้งและตั้งค่าใหม่ทั้งหมด ซึ่งใช้เวลาและอาจเกิดข้อผิดพลาดได้ง่าย
- สื้นเปลืองทรัพยากร:** หาก Server มีทรัพยากรเหลือ แต่ไม่สามารถติดตั้งโปรแกรมอื่นได้ เพราะ Library ขัดกัน ทรัพยากรส่วนนั้นก็จะถูกปล่อยกึ่งไว้โดยเปล่าประโยชน์
- ความปลอดภัย:** หาก Jenkins มีช่องโหว่ อาจส่งผลกระทบโดยตรงต่อระบบปฏิบัติการหลักของ Server ได้





2. Containerized Deployment (ติดตั้งผ่าน Container)

เป็นการ "แพ็ค" Jenkins พร้อมกับทุกสิ่งที่จำเป็นต้องใช้ (เช่น Java, Library) ไว้ในกล่องที่เรียกว่า **Container** (นิยมใช้ Docker) และนำกล่องนี้ไปวางรันบน Server อีกทีหนึ่ง ตัว Container จะถูกแยกสภาพแวดล้อมออกจาก OS หลักของ Server อย่างสื่นเชิง

ข้อดี

- ติดตั้งและอัปเดตง่ายมาก: แค่ใช้คำสั่ง `docker pull` เพื่อดึงเวอร์ชันใหม่ และ `docker run` เพื่อเปิดใช้งาน ไม่ต้องกังวลเรื่อง Dependency ใดๆ เพราะทุกอย่างอยู่ใน Container แล้ว
- ย้ายไปไหนก็ได้ (**Portability**): คุณสามารถนำ Container ของ Jenkins กีตั้งค่าเรียบร้อยแล้วไปรันบนเครื่องไหนก็ได้ที่มี Docker ติดตั้งอยู่ ไม่ว่าจะเป็นเครื่องเพื่อน, Cloud หรือ Server ใหม่ ทำให้การย้ายระบบ ทำได้สะดวกและรวดเร็วมาก
- สภาพแวดล้อมเหมือนกันทุกที่: หมวดปัญหา "เครื่องคอมพิวเตอร์ใดๆ แต่พอขึ้น Server แล้วพัง" เพราะ Container จะสร้างสภาพแวดล้อมที่เหมือนกัน 100% ไม่ว่าจะรันที่ไหนก็ตาม
- ปลอดภัยกว่า: เนื่องจาก Jenkins ถูกแยกสภาพแวดล้อมไว้ใน Container ถึงแม้ว่ามีช่องโหว่เกิดขึ้น ก็จะจำกัดความเสียหายอยู่แค่ภายใน Container นั้น ไม่ส่งผลกระทบต่อ OS หลักของ Server
- จัดการทรัพยากรได้ดี: สามารถรัน Container หลายๆ ตัวบน Server เดียวกันได้โดยที่แต่ละตัวไม่รบกวนกัน ทำให้ใช้ทรัพยากรเครื่องได้คุ้มค่ากว่า

ข้อเสีย

- ประสิทธิภาพลดลงเล็กน้อย: มี Overhead จากขั้นของ Container ทำให้ประสิทธิภาพด้อยกว่า Bare Metal เล็กน้อย (แต่ในการใช้งานจริงແກบไม่เห็นความแตกต่าง)
- ขับข้อนี้เล็กน้อย: ต้องมีความรู้พื้นฐานเกี่ยวกับ Docker หรือ Container เพิ่มเติม เช่น การจัดการ Volume สำหรับเก็บข้อมูล, การตั้งค่า Network ภายใน Container
- การเข้าถึง Hardware: การเข้าถึงอุปกรณ์ Hardware พิเศษโดยตรงจากใน Container จะมีความซับซ้อนมากกว่า



ขั้นตอนการติดตั้ง Jenkins

Bare Metal Installation
(ติดตั้งบน OS โดยตรง)



ขั้นตอนการติดตั้ง Jenkins

(ต้องการ Java 17 หรือ 21)



java -version

หมายเหตุ ในคอร์สนี้ Jenkins ต้องการ Java JDK **17** หรือ **21** เท่านั้น (เก่าหรือใหม่กว่านี้ไม่รองรับ)



สถาบันไอทีเจเนียส

www.itgenius.co.th

ตรวจสอบ Java Version

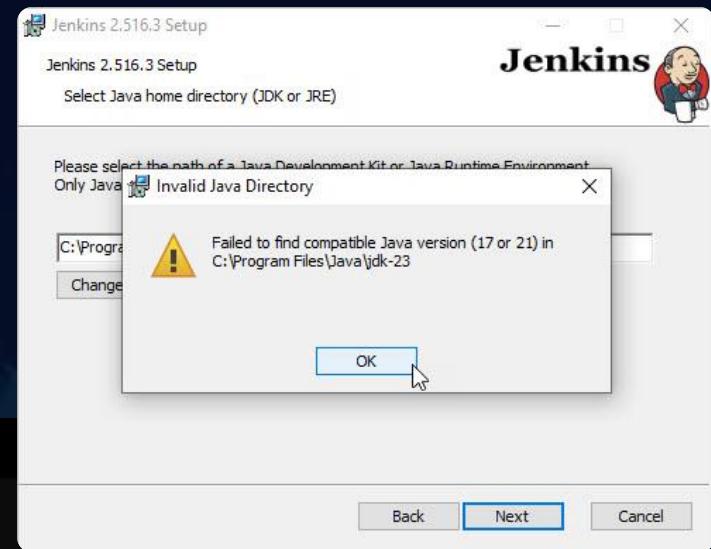
Command Prompt

```
C:\Users\Samit>java -version
java version "21.0.8" 2025-07-15 LTS
Java(TM) SE Runtime Environment (build 21.0.8+12-LTS-250)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.8+12-LTS-250, mixed mode, sharing)

C:\Users\Samit>set JAVA_HOME
JAVA_HOME=C:\Program Files\Java\jdk-21

C:\Users\Samit>where java
C:\Program Files\Common Files\Oracle\Java\javapath\java.exe
C:\Program Files\Java\jdk-21\bin\java.exe

C:\Users\Samit>
```



ดาวน์โหลด Jenkins [ลิงค์ <https://www.jenkins.io/download/>]

The screenshot shows the Jenkins download page with two main sections: "Download Jenkins 2.516.3 LTS for:" and "Download Jenkins 2.531 for:". Both sections provide links for various platforms, including Generic Java package (.war), Docker, Kubernetes, Ubuntu/Debian, Red Hat Enterprise Linux and derivatives, Fedora, Windows, openSUSE, Arch Linux (Third party), FreeBSD (Third party), and Gentoo (Third party). The "Windows" link in the first section is being clicked, as indicated by a mouse cursor icon.

1. Before downloading, please take a moment to review the [Hardware and Software requirements](#) section of the User Handbook.
2. Select one of the packages below and follow the download instructions.
3. Once a Jenkins package has been downloaded, proceed to the [Installing Jenkins](#) section of the User Handbook.
4. You may also want to verify the package you downloaded. [Learn more about verifying Jenkins downloads.](#)

Download Jenkins 2.516.3 LTS for:

- Generic Java package (.war)
SHA-256: 81b3abcc0f24cea48e74effe152f69dc5f0d880edc0c2737c61446b3c5992c00
- Docker
- Kubernetes
- Ubuntu/Debian
- Red Hat Enterprise Linux and derivatives
- Fedora
- Windows
- openSUSE
- FreeBSD
- Gentoo

Download Jenkins 2.531 for:

- Generic Java package (.war)
SHA-256: ccaa59b34ff7fb13bcd442d57b39eee130f7ee582792269fde262fb18b9212f41
- Docker
- Ubuntu/Debian
- Red Hat Enterprise Linux and derivatives
- Fedora
- Windows
- openSUSE
- Arch Linux Third party
- FreeBSD Third party
- Gentoo Third party



สถาบันฯ

<https://www.jenkins.io/download/thank-you-downloading-windows-installer-stable>

itgenius.co.th

Thank you for downloading Windows Stable installer

Download hasn't started? [Click this link](#)



Changing boot configuration

By default, your Jenkins runs at <https://localhost:8080/>. This can be changed by editing `jenkins.xml`, which is located in your installation directory. This file is also the place to change other boot configuration parameters, such as JVM options, HTTPS setup, etc.

Starting/stopping the service

Jenkins is installed as a Windows service, and it is configured to start automatically upon boot. To start/stop them manually, use the service manager from the control panel, or the `sc` command line tool.

Inheriting your existing Jenkins installation

If you'd like your new installation to take over your existing Jenkins data, copy your old data directory into the new `JENKINS_HOME` directory.

See Also

- [Running Jenkins behind Internet Information Server \(IIS\)](#)
- [Running Jenkins behind nginx](#)
- [Running Jenkins behind Apache](#)

Downloads

File Home Share View Application Tools Manage

Pin to Quick access Copy Paste Cut Copy path Move to Copy to Delete Rename New folder New item Open Select all Select none Invert selection Properties Easy access History

Clipboard Organize New Open Select

Search Downloads

jenkins.msi
Windows Installer Package

Date modified: 8/10/2568 17:20
Size: 89.3 MB
Date created: 23/9/2568 18:07

Quick access

Desktop Downloads Documents Pictures 2025-7 Basic Docker for Developer 2026 Devops Jenkins Github Action N8N คู่เพื่อน โนบลูน This PC 3D Objects Desktop Documents Downloads Music Pictures Videos Local Disk (C:) Local Disk (D:) Local Disk (H:) Local Disk (I:)

Today (1)

jenkins.msi

Earlier this week (8)

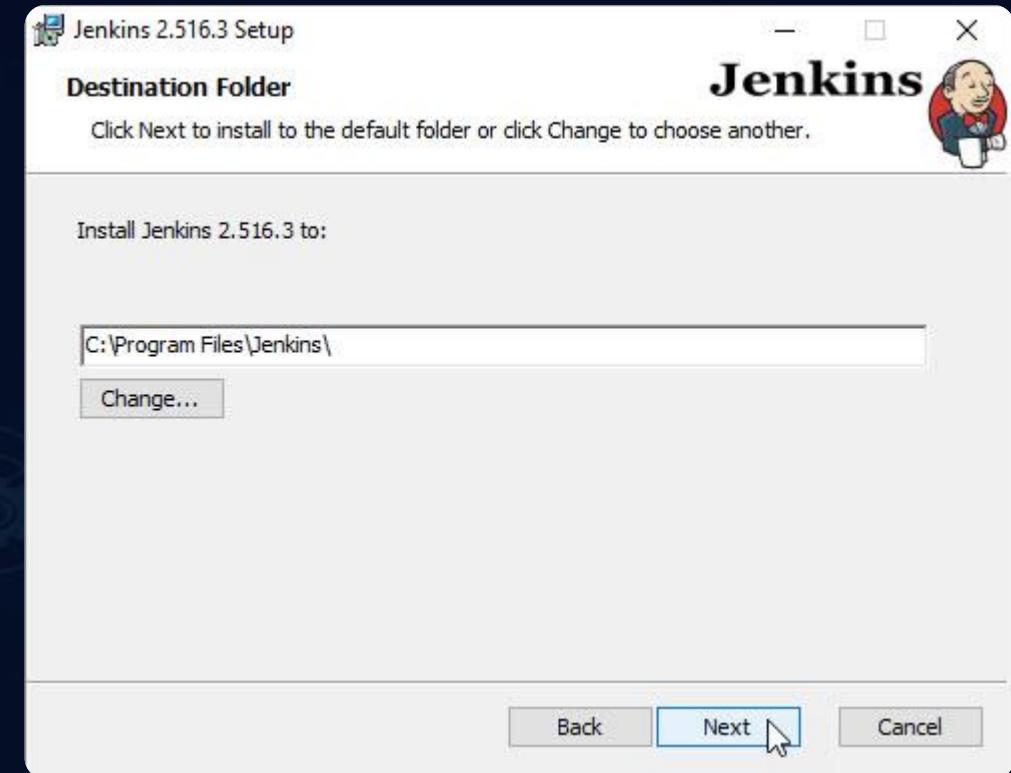
Day1_GitCheatSheetThai.md Day1_Note.md PPT_Meme.jpg ChatGPT Image Oct 6, 2025, 06_29_49 P... ChatGPT Image Oct 6, 2025, 06_28_30 P... r_1978859_GTKuw.jpg 1Ygk03a.png AWS Billing

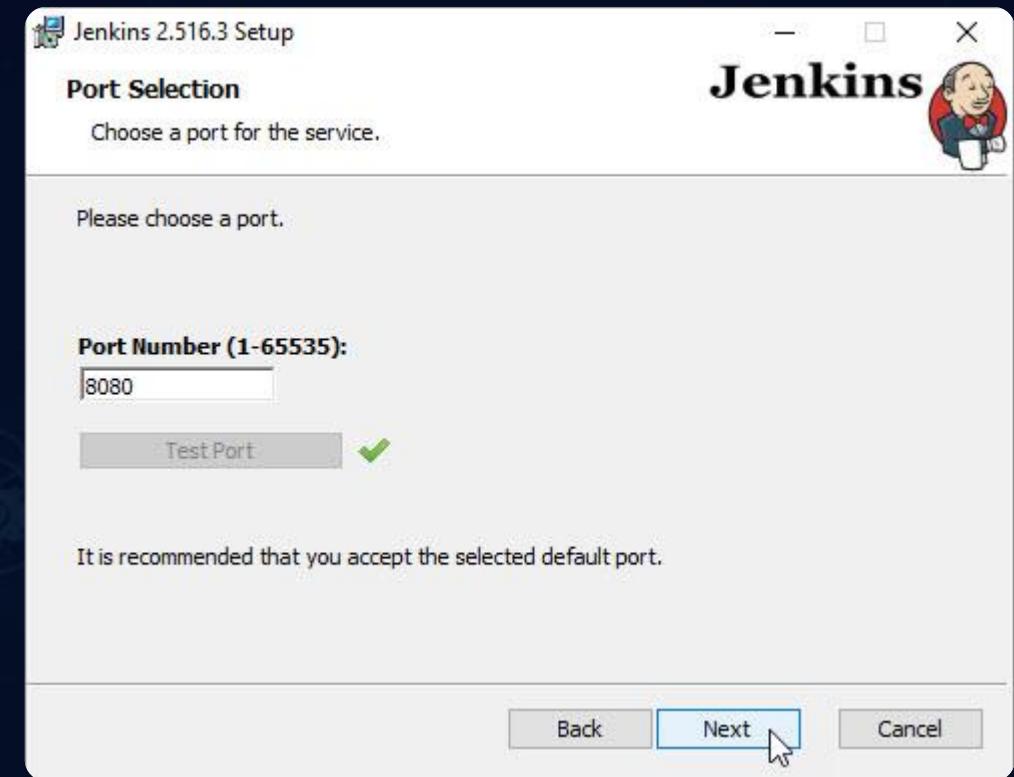
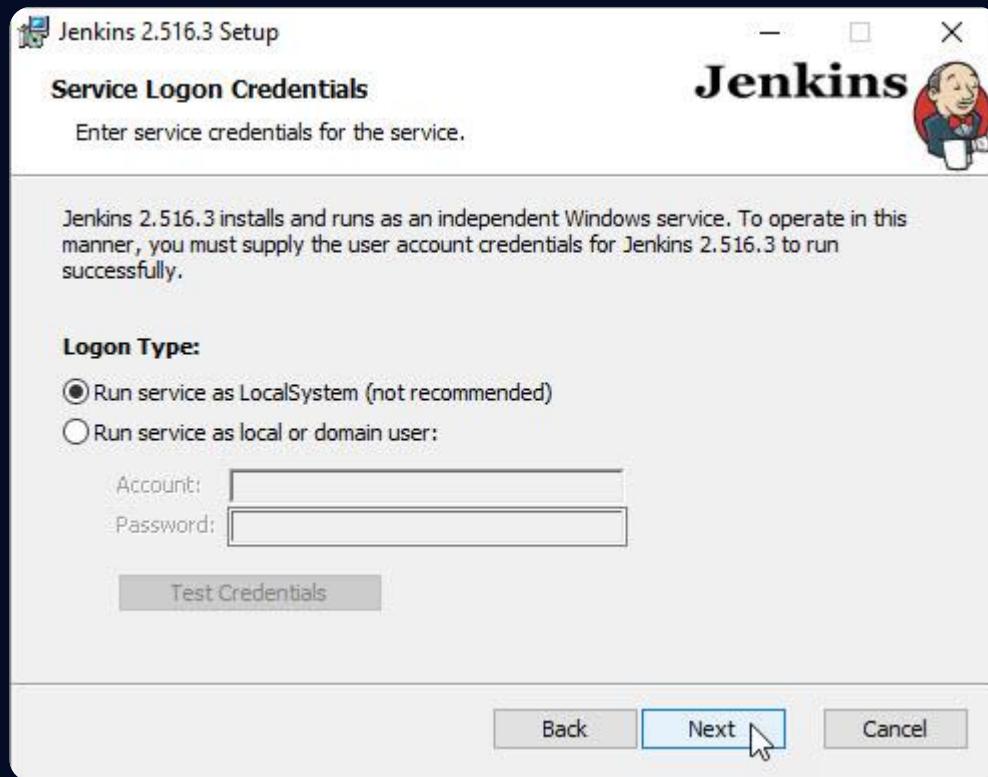
Last week (26)

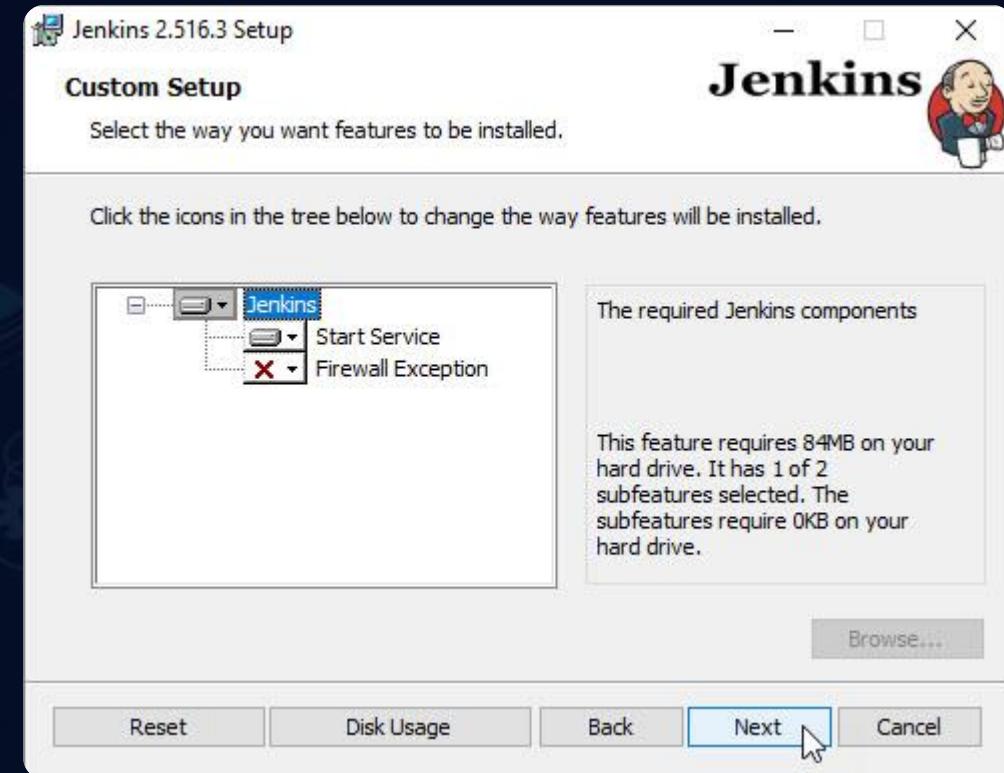
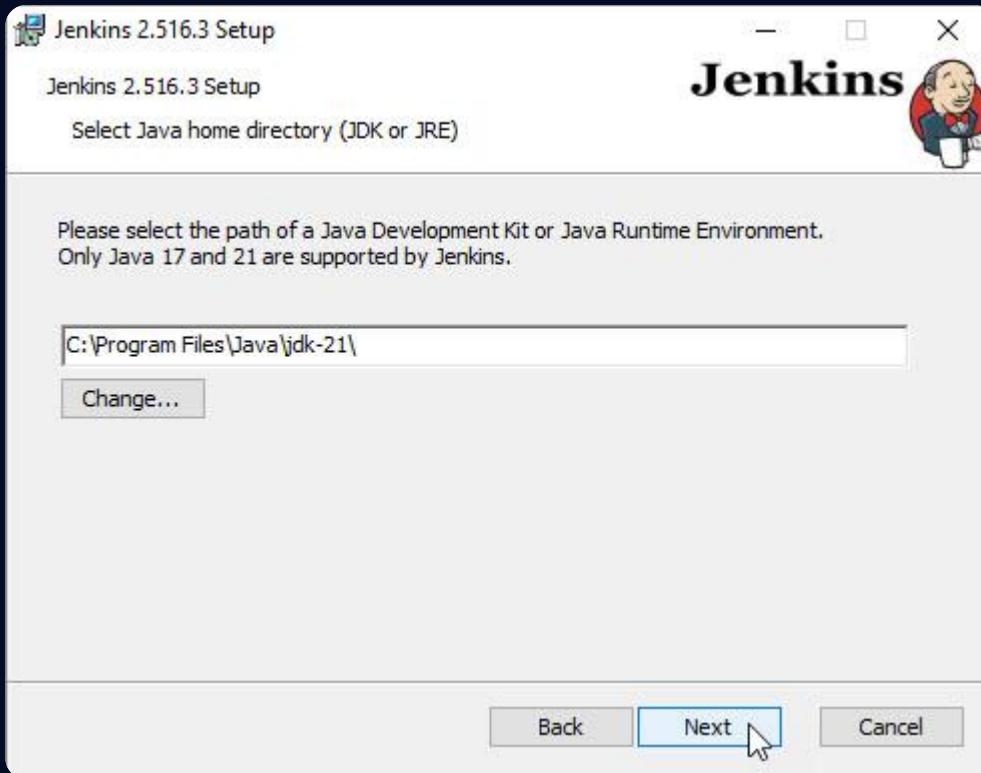
Adobe Express - file (1).png What_is_Ci_CD.png ChatGPT Image Oct 2, 2025, 07_55_17 P... SCB_09_2025.pdf KBANK_09_2025.pdf 1744856874818.png itgecoth_db(1).sql Data Dictionary - ตาราง Course.pdf course_data_dict.md

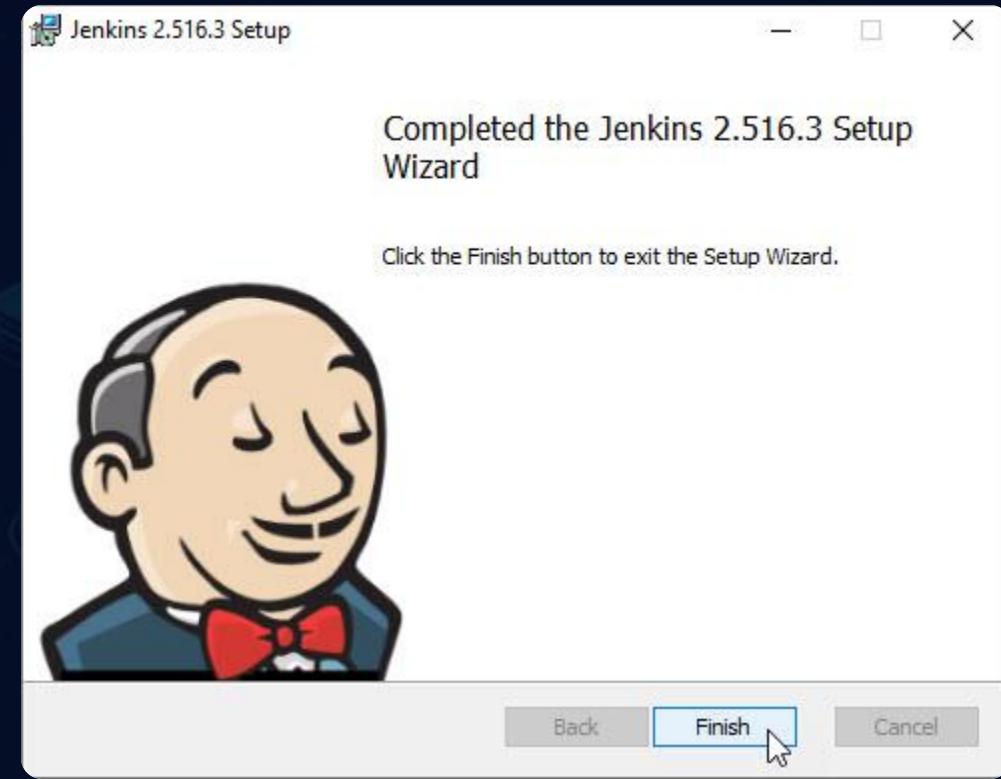
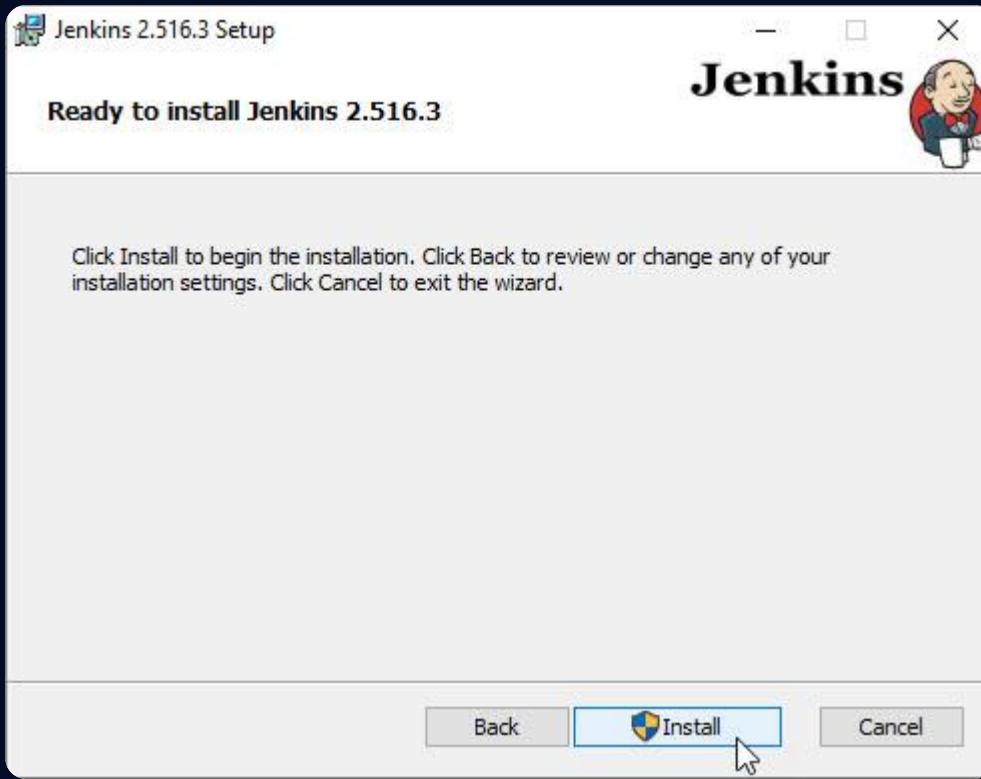
Claude-Setup.exe Invoice_2186.pdf Invoice_2223.pdf Invoice_2326.pdf Invoice_2326.pdf 1_lz.webp DockerCheatsheet.pdf carbon.pdf GitCheatSheet.pdf

736 items 1 item selected 89.3 MB









Task Manager

File Options View

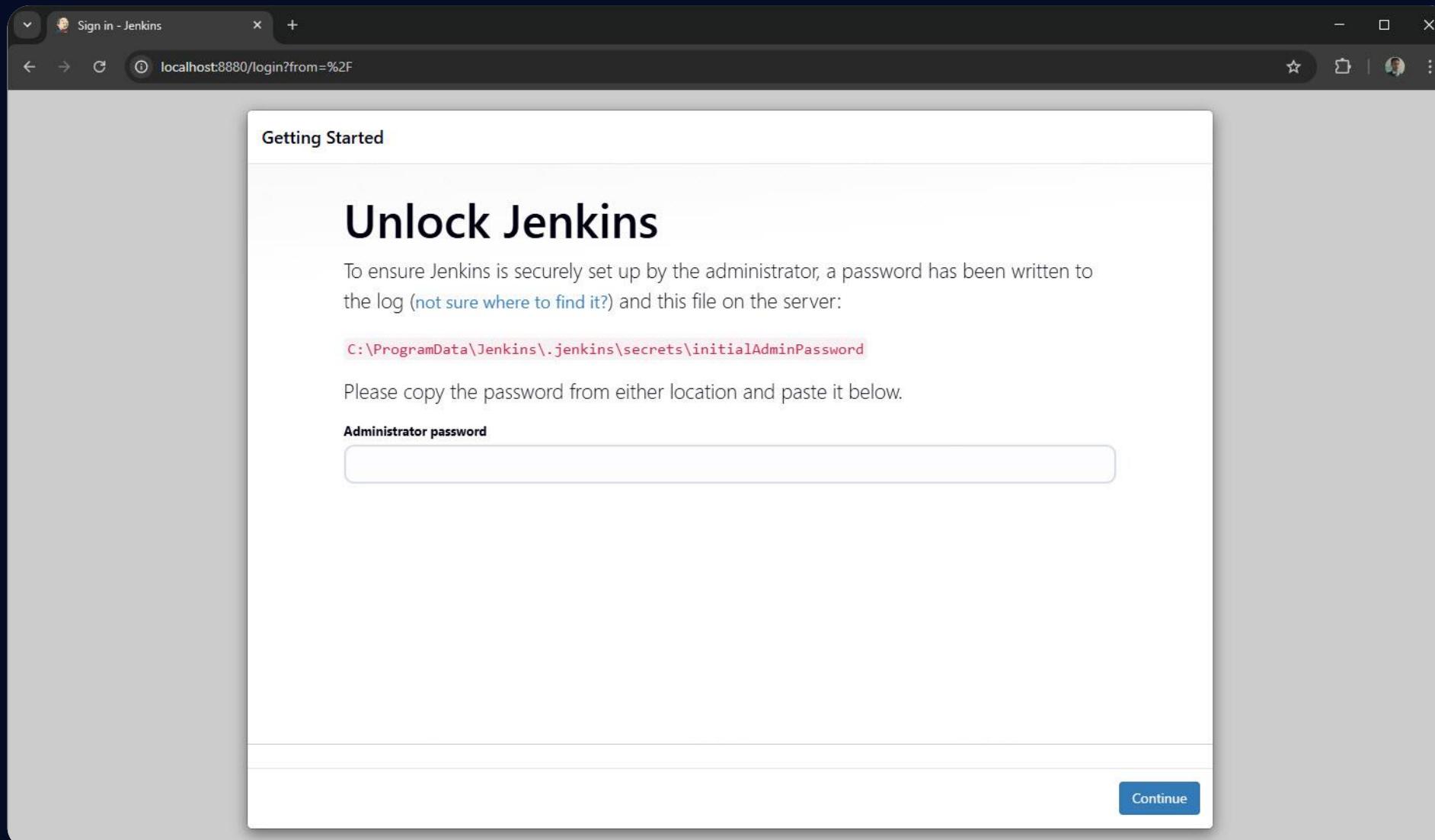
Processes Performance App history Startup Users Details Services

Name	PID	Description	Status	Group
Jenkins	45812	Jenkins	Running	
jhi_service	6712	Intel(R) Dynamic Application Loader Host Interface Service	Running	
KeyIso	644	CNG Key Isolation	Running	
KtmRm		KtmRm for Distributed Transaction Coordinator	Stopped	NetworkService...
LanmanServer	4788	Server	Running	netsvcs
LanmanWorkstation	4356	Workstation	Running	NetworkService
lfsvc	13296	Geolocation Service	Running	netsvcs
LicenseManager	6836	Windows License Manager Service	Running	LocalService
lltdsvc		Link-Layer Topology Discovery Mapper	Stopped	LocalService
Imhosts	62608	TCP/IP NetBIOS Helper	Running	LocalServiceN...
LMS	18332	Intel(R) Management and Security Application Local Mana...	Running	
LSM	1188	Local Session Manager	Running	DcomLaunch
LxpSvc		Language Experience Service	Stopped	netsvcs
LxssManager		LxssManager	Stopped	netsvcs
LxssManagerUser		LxssManagerUser	Stopped	LxssManagerU...
LxssManagerUser_194004		LxssManagerUser_194004	Stopped	LxssManagerU...
MapsBroker		Downloaded Maps Manager	Stopped	NetworkService
McpManagementService		McpManagementService	Stopped	McpManage...
MDCoreSvc	57548	Microsoft Defender Core Service	Running	
MessagingService		MessagingService	Stopped	UnistackSvcGr...
MessagingService_194004		MessagingService_194004	Stopped	UnistackSvcGr...
MicrosoftEdgeElevationSer...		Microsoft Edge Elevation Service (MicrosoftEdgeElevationS...	Stopped	
MixedRealityOpenXRService		Windows Mixed Reality OpenXR Service	Stopped	LocalSystemN...
MongoDB		MongoDB Server (MongoDB)	Stopped	
mpssvc	2316	Windows Defender Firewall	Running	LocalServiceN...
MSDTC		Distributed Transaction Coordinator	Stopped	
MSiSCSI		Microsoft iSCSI Initiator Service	Stopped	netsvcs
msiserver	34700	Windows Installer	Running	
MsKeyboardFilter		Microsoft Keyboard Filter	Stopped	netsvcs
NaturalAuthentication		Natural Authentication	Stopped	netsvcs
NcaSvc		Network Connectivity Assistant	Stopped	NetSvcs
NcbService	1468	Network Connection Broker	Running	LocalSystemN...
NcdAutoSetup		Network Connected Devices Auto-Setup	Stopped	LocalServiceN...
Netlogon		Netlogon	Stopped	
Netman	15492	Network Connections	Running	LocalSystemN...

[Fewer details](#) | [Open Services](#)



เปิดเข้าใช้งาน Jenkins ที่ <http://localhost:8080>



Local Disk (C:)

File Home Share View

Panes: Details pane

Layout: Details

Sort by: Current view

Show/hide: Hide selected items

Options: Item check boxes, File name extensions, Hidden items

Navigation pane: This PC, Local Disk (C:), Local Disk (D:), Local Disk (H:), Local Disk (I:)

Search Local Disk (C:)

Quick access:

- Desktop
- Downloads
- Documents
- Pictures
- 2025-7
- Basic Docker for Developer 2026
- Devops Jenkins Github Action N8N
- คู่เพียง
- โนบเนนจาด

This PC:

- 3D Objects
- Desktop
- Documents
- Downloads
- Music
- Pictures
- Videos
- Local Disk (C:)
- Local Disk (D:)
- Local Disk (H:)
- Local Disk (I:)

Table of contents:

Name	Date modified	Type	Size
คลังศิลปะรวมไฟล์ 2025	24/9/2568 22:02	File folder	
app@tmp	24/9/2568 4:30	File folder	
app	24/9/2568 4:24	File folder	
ProgramData	23/9/2568 20:53	File folder	
maven3911	23/9/2568 17:25	File folder	
nvm4w	14/9/2568 18:44	File folder	
ITGeniusProject	1/9/2568 22:12	File folder	
DockerWebDev	16/8/2568 2:35	File folder	
TC	1/8/2568 1:47	File folder	
BackupDBITGenius	14/7/2568 21:22	File folder	
\$WinREAgent	8/6/2568 15:39	File folder	
Microsoft Shared	25/5/2568 20:56	File folder	
Windows	12/5/2568 13:04	File folder	
flutter	7/4/2568 9:25	File folder	
AWSKey	1/4/2568 15:31	File folder	
โครงการที่ปรับปรุง	31/3/2568 15:08	File folder	
คลังศิลปะรวมไฟล์ 2023	31/3/2568 13:50	File folder	
ProjectFiles	31/3/2568 13:50	File folder	
คลังศิลปะ 2024	31/3/2568 13:42	File folder	
คลังศิลปะรวมไฟล์ 2024	31/3/2568 13:42	File folder	
BandicamRecord	31/3/2568 3:20	File folder	
xampp	30/3/2568 17:04	File folder	
OneDriveTemp	30/3/2568 16:50	File folder	
Users	30/3/2568 15:07	File folder	
PerfLogs	7/12/2562 16:14	File folder	

ProgramData: File folder, Date modified: 23/9/2568 20:53

31 items | 1 item selected

File Home Share View

Panes Navigation pane Details pane Layout Current view Show/hide Options

Extra large icons Large icons Medium icons Details Group by Add columns Sort by Size all columns to fit Item check boxes File name extensions Hidden items Hide selected items

Navigation pane Details pane Panes Layout Current view Show/hide Options

Search secrets

Name	Date modified	Type	Size
hudson.console.AnnotatedLargeText.con...	24/9/2568 3:43	CONSOLEANNOT...	1 KB
hudson.console.ConsoleNote.MAC	23/9/2568 21:11	MAC File	1 KB
hudson.model.Job.serverCookie	24/9/2568 3:32	SERVERCOOKIE File	1 KB
hudson.util.Secret	23/9/2568 21:05	SECRET File	1 KB
initialAdminPassword	27/9/2568 3:02	File	1 KB
jenkins.model.Jenkins.crumbSalt	23/9/2568 20:53	CRUMBSALT File	1 KB
master.key	23/9/2568 20:53	KEY File	1 KB
org.jenkinsci.main.modules.instance_ide...	23/9/2568 21:01	KEY File	1 KB
org.jenkinsci.plugins.workflow.log.Cons...	23/9/2568 21:12	CONSOLEANNOT...	1 KB
org.springframework.security.web.authe...	26/9/2568 23:46	MAC File	1 KB

initialAdminPassword
File Date modified: 27/9/2568 3:02
Size: 32 bytes Date created: 27/9/2568 3:01

initialAdminPassword - Notepad

File Edit Format View Help

52b49468908d4435b5243f5dfc5098ae

10 items 1 item selected 32 bytes Ln 1, Col 1 160% Windows (CRLF) UTF-8

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword

Please copy the password from either location and paste it below.

Administrator password

.....

Continue

Sign in - Jenkins

localhost:8080/login?from=%2F

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

.....

Continue

o.th

Setup Wizard - Jenkins

localhost:8880

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.



Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.516.3

localhost:8880/#

Setup Wizard - Jenkins

localhost:8880

Getting Started

Getting Started

<input checked="" type="checkbox"/> Folders	<input checked="" type="checkbox"/> OWASP Markup Formatter	<input checked="" type="checkbox"/> Build Timeout	<input checked="" type="checkbox"/> Credentials Binding	** commons-lang3 v3.x Jenkins API ** Ionicons API Folders OWASP Markup Formatter ** ASM API ** JSON Path API ** Structs ** Pipeline: Step API ** commons-text API ** Token Macro Build Timeout ** bouncycastle API ** Credentials ** Plain Credentials ** Variant ** SSH Credentials Credentials Binding ** SCM API ** Pipeline: API
<input type="radio"/> Timestamper	<input type="radio"/> Workspace Cleanup	<input type="radio"/> Ant	<input type="radio"/> Gradle	
<input type="radio"/> Pipeline	<input type="radio"/> GitHub Branch Source	<input type="radio"/> Pipeline: GitHub Groovy Libraries	<input type="radio"/> Pipeline Graph View	
<input type="radio"/> Git	<input type="radio"/> SSH Build Agents	<input type="radio"/> Matrix Authorization Strategy	<input type="radio"/> LDAP	
<input type="radio"/> Email Extension	<input type="radio"/> Mailer	<input type="radio"/> Dark Theme		** - required dependency

Jenkins 2.516.3

Setup Wizard - Jenkins

localhost:8880

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins 2.516.3

Skip and continue as admin

Save and Continue

Setup Wizard - Jenkins

localhost:8880

Getting Started

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.516.3

Not now

Save and Finish

Setup Wizard - Jenkins

localhost:8880

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

Jenkins 2.516.3

Dashboard - Jenkins

localhost:8880

New Item

Build History

Build Queue

No builds in the queue.

Build Executor Status

0/2

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job +

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds ?

REST API Jenkins 2.516.3

The screenshot shows the Jenkins dashboard at localhost:8880. At the top left, there's a sidebar with links for 'New Item' and 'Build History'. Below this is a section for 'Build Queue' which says 'No builds in the queue.' To the right, a large 'Welcome to Jenkins!' header is followed by a descriptive paragraph about setting up distributed builds or starting a software project. A prominent 'Start building your software project' button is centered. Below it is a 'Create a job' button with a '+' sign. Further down, there's a 'Set up a distributed build' section with three items: 'Set up an agent' (with a monitor icon), 'Configure a cloud' (with a cloud icon), and a link 'Learn more about distributed builds' with a question mark icon. At the bottom right, there are links for 'REST API' and 'Jenkins 2.516.3'.

Sign in - Jenkins

localhost:8880/login?from=%2F



Sign in to Jenkins

Username

Password

Keep me signed in

Sign in

File Home Share View Account - Jenkins + localhost:8880/user/admin/account/ Pin to Quick access Copy Paste Clipboard

Jenkins / Samit Koyom / Account

Samit Koyom samitkoyom@gmail.com

Extended Email Job Watching

No configuration available

Notification URL

Notification URL ? Default

Time zone

Select a time zone to use rather than the system default.

Asia/Bangkok

Current time on server in Indochina Time: Sep 23, 2025, 9:05:52 PM; in proposed display zone: Sep 23, 2025, 9:05:52 PM

Save Apply

Search secrets

Samit Koyom Theme Light My Views Account Appearance Preferences Security Experiments Credentials Sign out

4 items | DevOps CICD Jenkins... README.md - Devop... secrets GitHub Actions docu... Account - Jenkins - G... CI/CD with Jenkins, G... Continuous Integrati... initialAdminPassword... ENG 21:06

Appearance - Jenkins

localhost:8880/user/admin/appearance/

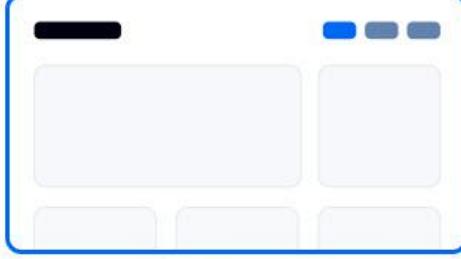
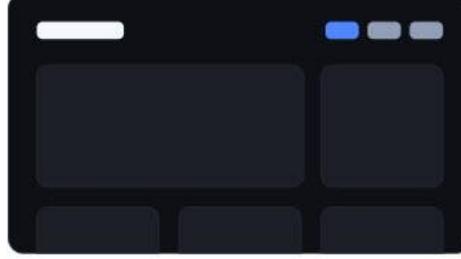
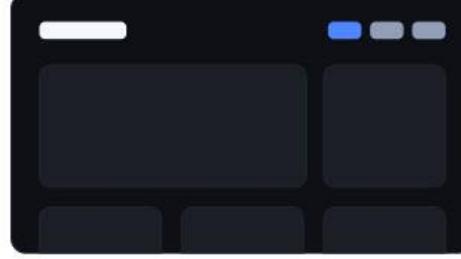
Jenkins / Samit Koyom / Appearance

Status Builds My Views Account Appearance Preferences Security Experiments Credentials

Appearance

Theme

Select a theme to change how Jenkins appears.

Light  Dark  Dark (System) 

Jenkins 2.516.3



ขั้นตอนการติดตั้ง Jenkins

Containerized Deployment
(ติดตั้งผ่าน Container)

jenkins/jenkins Tags | Docker Hub +

hub.docker.com/r/jenkins/jenkins/tags

Search Docker Hub CtrlK

Explore My Hub

Finish update



jenkins/jenkins Sponsored OSS

By [Jenkins](#) • Updated about 22 hours ago

The leading open source automation server

IMAGE

SECURITY

INTEGRATION & DELIVERY

MONITORING & OBSERVABILITY

★ 4.2K ↓ 1B+

Overview Tags

Sort by Newest ▾

Filter tags

TAG

[jdk21](#)

Last pushed about 22 hours by [jenkinsinfraadmin](#)

docker pull jenkins/jenkins:jk21

Digest

[a5d77018d0e5](#)

OS/ARCH

linux/amd64

Compressed size

278.35 MB

docker-compose.yml X

jenkins > docker-compose.yml > {} services > {} jenkins > [] networks > 0

docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-agnostic applications (compose-spec.json)

```
1 # Define Network
2 networks:
3   jenkins_network:
4     name: jenkins_network
5     driver: bridge
6
7 # Define Services
8 >Run All Services
9 services:
10   jenkins:
11     image: jenkins/jenkins:jdk21
12     container_name: jenkins
13     volumes:
14       - ./jenkins_home:/var/jenkins_home # เพื่อให้ Jenkins สามารถเก็บข้อมูลไว้ใน host
15       - /var/run/docker.sock:/var/run/docker.sock # เพื่อให้ Jenkins สามารถใช้งาน Docker daemon ที่รันบน host
16     environment:
17       - JENKINS_OPTS=--httpPort=8800 # กำหนด Port สำหรับ Jenkins UI
18     ports:
19       - "8800:8800" # สำหรับ Jenkins UI
20     restart: always
21     networks:
22       - jenkins_network|
```

```
jenkins ➤ docker compose config
```

```
name: jenkins
services:
  jenkins:
    container_name: jenkins
    environment:
      JENKINS_OPTS: --httpPort=8800
    image: jenkins/jenkins:jdk21
    networks:
      jenkins_network: null
    ports:
      - mode: ingress
        target: 8800
        published: "8800"
        protocol: tcp
    restart: always
    volumes:
      - type: bind
        source: C:\TrainingWorkshop\Devops_Jenkins_Github Actions_N8N\jenkins\jenkins
        target: /var/jenkins_home
        bind:
          create_host_path: true
      - type: bind
        source: /var/run/docker.sock
        target: /var/run/docker.sock
        bind:
          create_host_path: true
  networks:
    jenkins_network:
      name: jenkins_network
      driver: bridge
```



```
jenkins ➤ docker compose up -d
[+] Running 13/13
✓ jenkins Pulled                                70.8s
✓ cae3b572364a Pull complete                   13.9s
✓ 1d29e73f5855 Pull complete                   15.9s
✓ 1d29e73f5855 Pull complete                   15.9s
✓ 8b4c33723b69 Pull complete                   16.6s
✓ 1dae4729c030 Pull complete                   67.0s
✓ 9f98d0843ab2 Pull complete                   16.7s
✓ 3f5883b511f6 Pull complete                   2.5s
✓ 45393cee554d Pull complete                   2.4s
✓ 7e1b13b62f4e Pull complete                   2.1s
✓ ea33278004bf Pull complete                   16.0s
✓ a4905a8c1a34 Pull complete                   2.3s
✓ 13f268808647 Pull complete                   1.2s
●   ✓ f0720e32929a Pull complete               66.9s
[+] Running 3/3
✓ Network jenkins_network Created              0.1s
✓ Volume "jenkins_data" Created                0.0s
✳ ✓ Container jenkins Started                 0.8s
Samit ➤ ⏎ jenkins ➤ ✓                                in pwsh at 18:18:09
```



docker exec -it jenkins /bin/bash

```
PROBLEMS    PORTS    OUTPUT    DEBUG CONSOLE    GITLENS    TERMINAL
jenkins ➔ docker exec -it jenkins /bin/bash
jenkins@c8c9ab1c292d:$ git --version
git version 2.47.3
jenkins@c8c9ab1c292d:$ java -version
openjdk version "21.0.8" 2025-07-15 LTS
OpenJDK Runtime Environment Temurin-21.0.8+9 (build 21.0.8+9-LTS)
OpenJDK 64-Bit Server VM Temurin-21.0.8+9 (build 21.0.8+9-LTS, mixed mode)
jenkins@c8c9ab1c292d:$ |
```

Ask Gordon BETA

Containers

Images

Volumes

Builds

Docker Hub

Docker Scout

Extensions

Images Give feedback

View and manage your local and Docker Hub images. [Learn more](#)

[Local](#) [Hub repositories](#)

6.79 GB / 4.93 GB in use 13 images

Last refresh: 20 hours ago

 jenkins[X](#)

<input type="checkbox"/>	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	jenkins/jenkins	jdk21	3ac87059be85	22 hours ago	818.68 MB	▶ ⋮ trash

Showing 1 item

Ask Gordon BETA

Containers

Images

Volumes

Builds

Docker Hub

Docker Scout

Extensions

Containers [Give feedback](#)View all your running containers and applications. [Learn more](#)

Container CPU usage

1.44% / 1200% (12 CPUs available)

Container memory usage

1.29GB / 15.22GB

[Show charts](#)

jenkins



Only show running containers

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	Actions
<input type="checkbox"/>	jenkins	-	-	http://localhost:8800	
<input type="checkbox"/>	jenkins	c8c9ab1c292d	jenkins/jenkins:jdk21	8800:80	

Showing 2 items

Sign in - Jenkins

localhost:8800/login?from=%2F

Finish update :

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue

EXPLORER: JENKINS



jenkins_home

.cache

java

jobs

plugins

secrets

initialAdminPassword

jenkins.model.Jenkins.crumbSalt

master.key

updates

userContent

users

war

.lastStarted

config.xml

copy_reference_file.log

hudson.model.UpdateCenter.xml

jenkins.telemetry.Correlator.xml

nodeMonitors.xml

secret.key

secret.key.not-so-secret

docker-compose.yml

initialAdminPassword X

jenkins_home > secrets > initialAdminPassword

1 13fec88278704813a270cb
2

jenkins.model.Jenkins.crumbSalt X

jenkins_home > secrets > jenkins.model.Jenkins

1 ?M?EM?X?????fs\93
2 ?G[ivM

master.key X

jenkins_home > secrets > master.key

1 aa6304c4687c9ef7fc049f
08e155c88c4b8bacb9fa77
225a5768e8b983d43d946c

1. `initialAdminPassword`

- หน้าที่:** ไฟล์นี้จะเก็บรหัสผ่านเริ่มต้นที่ Jenkins สร้างขึ้นในครั้งแรกที่ติดตั้ง Jenkins บนระบบของคุณ รหัสผ่านนี้ถูกใช้ในการตั้งค่าเริ่มต้นของ Jenkins ซึ่งจำเป็นต้องใช้เมื่อคุณเข้าสู่ Jenkins ผ่านเว็บอินเตอร์เฟช เป็นครั้งแรก
- ความสำคัญ:** เมื่อคุณเปิดใช้งาน Jenkins ครั้งแรก ระบบจะขอให้คุณกรอกรหัสผ่านจากไฟล์นี้เพื่อปลดล็อก Jenkins และดำเนินการติดตั้งเพิ่มเติม เช่น การติดตั้งปลักอินและการสร้างบัญชีผู้ใช้งาน

2. `jenkins.model.Jenkins.crumbSalt`

- หน้าที่:** ไฟล์นี้เก็บค่า "crumb salt" ซึ่งเป็นข้อมูลที่ใช้ในการป้องกันการโจมตีแบบ Cross-Site Request Forgery (CSRF) ใน Jenkins โดยการสร้าง "crumb" หรือโทเคนที่เชื่อมโยงกับ session ของผู้ใช้งานแต่ละคน
- ความสำคัญ:** การป้องกัน CSRF เป็นมาตรการรักษาความปลอดภัยที่สำคัญในการป้องกันการโจมตีทางเว็บ ทำให้ค่าของทั้งหมดที่ส่งไปยัง Jenkins ต้องมาจากแหล่งที่เชื่อถือได้

3. `master.key`

- หน้าที่:** ไฟล์นี้เก็บคีย์หลักที่ใช้ในการเข้ารหัสข้อมูลลับต่าง ๆ ภายใน Jenkins เช่น รหัสผ่านและข้อมูลอื่น ๆ ที่ต้องการการป้องกันเพิ่มเติม คีย์นี้ถูกใช้โดย Jenkins เพื่อเข้ารหัสและถอดรหัสข้อมูลที่เก็บอยู่ในระบบ
- ความสำคัญ:** เป็นส่วนสำคัญของการรักษาความปลอดภัยข้อมูลใน Jenkins หากคีย์นี้สูญหายหรือถูกเปลี่ยนแปลง ข้อมูลลับที่ถูกเข้ารหัสโดย Jenkins จะไม่สามารถถูกถอดรหัสได้



Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

.....

Continue

AutoSave Off Devops_Jenkins_Github_Action_N8N.pptx • Saved to this PC

Search

File Home Insert Draw Design Transitions Animations Slide Show Record Review View Help ACROBAT

Cut Copy Format Painter Paste New New Slide with Copilot Reset Layout Section Convert to SmartArt Align Text Styles Shape Effects Select Find and Replace Replace Fonts Dictate Add-ins Design Suggestions Copilot

Clipboard Slide 16

Setup Wizard - Jenkins

localhost:8800

Finish update

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Select plugins to install

Install plugins the Jenkins community finds most useful.

Select and install plugins most suitable for your needs.

Jenkins 2.531

Slide 234 of 258 English (United States) Accessibility: Investigate Notes Display Settings 101% 18:30

232 ★

233 ★

234 ★

235 ★

236 ★

237 ★

238 ★

239 ★

Jenkins Configuration (Ctrl)

Initial Admin Password Screenshot - Google Contributors to docker/README Setup Wizard - Jenkins Devops_Jenkins_ GitHub Actions Containers - Docker Calculator

Setup Wizard - Jenkins

localhost:8800

Finish update

Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
✓ Timestamper	✓ Workspace Cleanup	✓ Ant	⌚ Gradle
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline Graph View
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ LDAP
⌚ Email Extension	⌚ Mailer	⌚ Dark Theme	

Timestamper

- ** Caffeine API
- ** Script Security
- ** JavaBeans Activation Framework (JAF) API
- ** JAXB
- ** SnakeYAML API
- ** Jakarta Activation API
- ** Jakarta XML Binding API
- ** JSON Api
- ** Jackson 2 API
- ** Pipeline: Supporting APIs
- ** Plugin Utilities API
- ** Font Awesome API
- ** Bootstrap 5 API
- ** JQuery3 API
- ** ECharts API
- ** Display URL API
- ** Checks API
- ** JUnit
- ** Matrix Project
- ** Resource Disposer

Workspace Cleanup

Ant

** - required dependency

Jenkins 2.531

Setup Wizard - Jenkins

localhost:8800

Getting Started

Create First Admin User

Username

Password

 123456
Confirm password 123456
Full name

E-mail address

Jenkins 2.531

Skip and continue as admin

Save and Continue

co.th



Getting Started

Instance Configuration

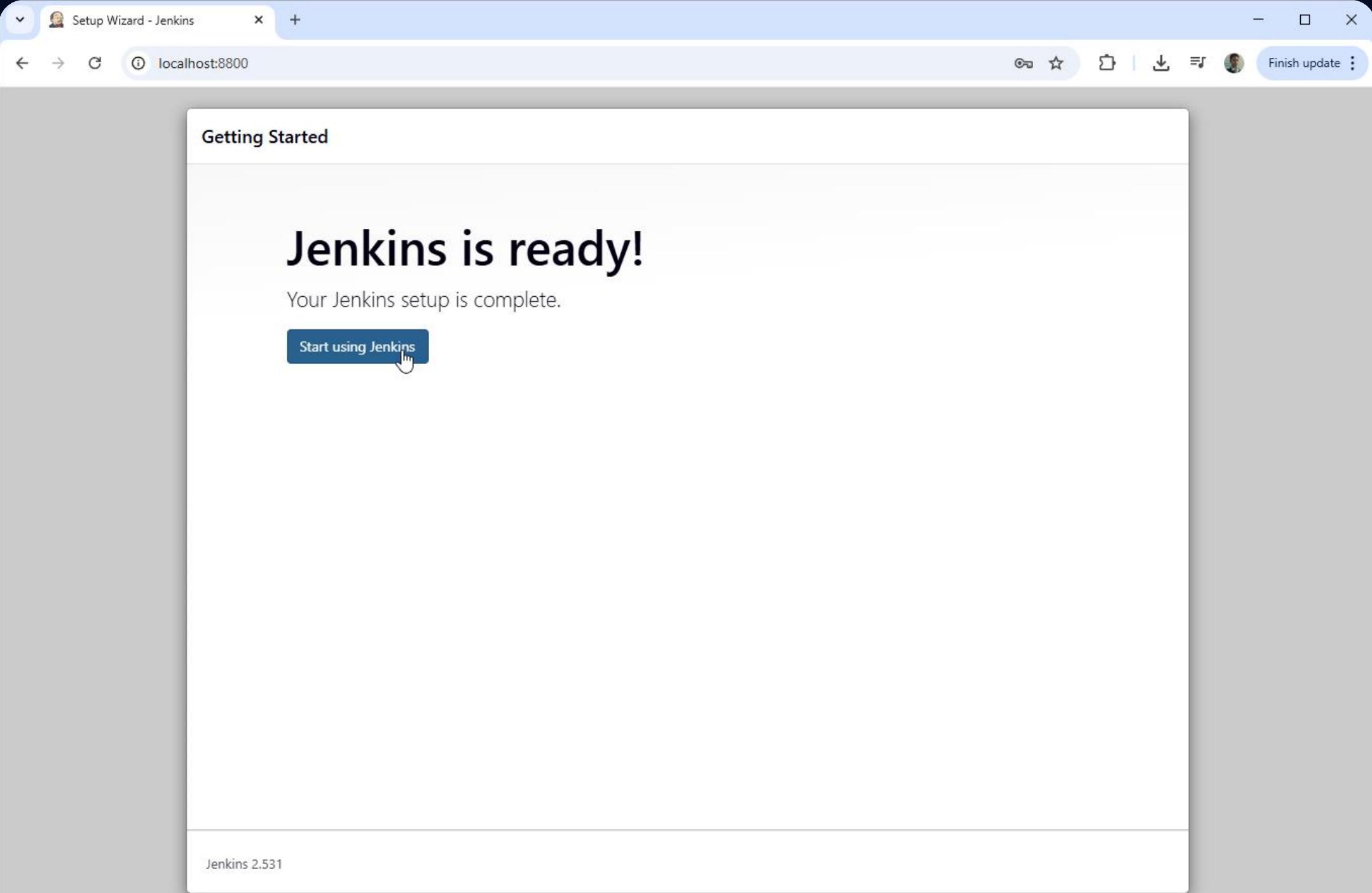
Jenkins URL:

http://localhost:8800/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.





Dashboard - Jenkins

localhost:8800

Finish update :

Jenkins

+ New Item

Build History

Build Queue

No builds in the queue.

Build Executor Status

0/2

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job +

Set up a distributed build

Set up an agent

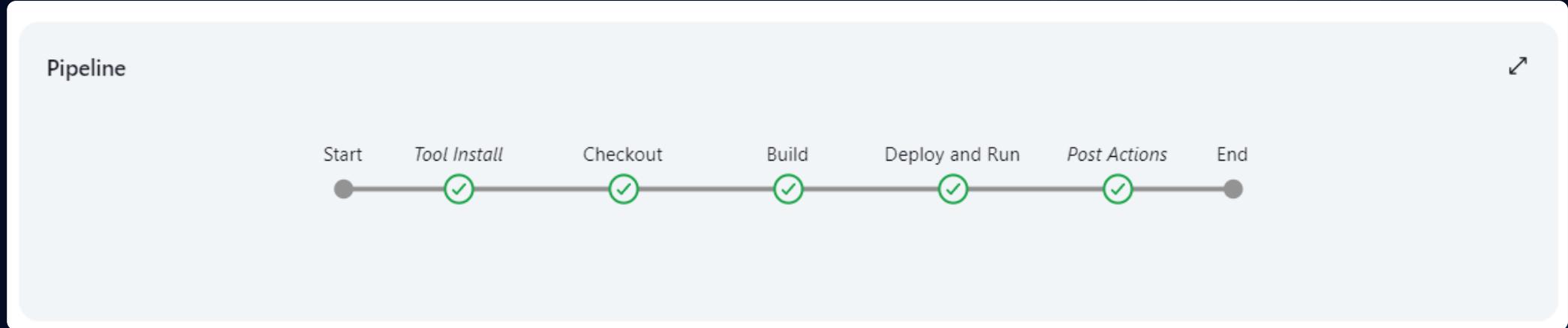
Configure a cloud

Learn more about distributed builds ?

REST API Jenkins 2.531

localhost:8800/view/all/newJob

What is Jenkins Pipeline?



Jenkins Pipeline (หรือเรียกสั้นๆ ว่า "Pipeline" โดยใช้ตัวพิมพ์ใหญ่ "P") เป็นชุดของปลั๊กอินที่สนับสนุนการนำเสนอกระบวนการและการผนวกรวม pipeline สำหรับ continuous delivery เข้าไปใน Jenkins

Continuous delivery (CD) pipeline เป็นการแสดงกระบวนการอัตโนมัติที่ช่วยนำซอฟต์แวร์จากระบบควบคุมเวอร์ชันไปจนถึงมือผู้ใช้และลูกค้าของคุณ ทุกครั้งที่มีการเปลี่ยนแปลงซอฟต์แวร์ (ที่ถูก commit ในระบบควบคุมเวอร์ชัน) จะต้องผ่านกระบวนการที่ซับซ้อนเพื่อให้สามารถนำออกใช้งานได้ กระบวนการนี้รวมถึงการสร้างซอฟต์แวร์ในลักษณะที่เชื่อถือได้และสามารถทำซ้ำได้ รวมถึงการนำซอฟต์แวร์ที่สร้างเสร็จแล้ว (เรียกว่า "build") ผ่านหลายขั้นตอนของการทดสอบและการปรับใช้



Declarative vs Scripted Pipeline syntax

การเขียน Pipeline ใน Jenkins สามารถทำได้หลัก ๆ 2 แบบ คือ Declarative Pipeline และ Scripted Pipeline โดยแต่ละแบบมีลักษณะและการใช้งานที่แตกต่างกัน ดังนี้

- 1. Declarative Pipeline**
- 2. Scripted Pipeline**



Declarative Pipeline

- ลักษณะ: เป็นรูปแบบที่ถูกออกแบบมาให้ใช้งานมากที่สุด เนื่องจากมีโครงสร้างที่ชัดเจนและใช้งานง่าย Declarative Pipeline ถูกออกแบบมาให้ผู้ใช้งานเข้าใจได้ง่ายขึ้นและลดความซับซ้อนในการเขียน Pipeline
- การใช้งาน: มีการใช้บล็อกที่กำหนดไว้อย่างชัดเจน เช่น pipeline, agent, stages, steps ทำให้โครงสร้างของ Pipeline มีความเป็นระเบียบและง่ายต่อการจัดการ

Declarative Pipeline

```
groovy

pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                echo 'Building...'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing...'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying...'
            }
        }
    }
}
```



Scripted Pipeline

- ลักษณะ: เป็นรูปแบบ Pipeline ที่ใช้ Groovy Script แบบเต็มรูปแบบ ซึ่งให้ความยืดหยุ่นสูงกว่า แต่มีความซับซ้อนมากขึ้น Scripted Pipeline เมนูสำหรับผู้ที่มีประสบการณ์ในการเขียนโค้ดและต้องการควบคุม Pipeline ได้อย่างละเอียด
- การใช้งาน: ใช้รูปแบบการเขียนโค้ด Groovy โดยไม่มีการบังคับให้ใช้ล็อกที่กำหนดไว้ สามารถเขียนโค้ดได้อย่างอิสระมากขึ้นและสามารถกำหนดการควบคุมฟลว์ของ Pipeline ได้อย่างซับซ้อน

Scripted Pipeline

```
groovy

node {
    stage('Build') {
        echo 'Building...'
    }
    stage('Test') {
        echo 'Testing...'
    }
    stage('Deploy') {
        echo 'Deploying...'
    }
}
```

คัดลอกโค้ด



Declarative vs Scripted Pipeline syntax

groovy

```
pipeline {  
    agent any  
    stages {  
        stage('Build') {  
            steps {  
                echo 'Building...'  
            }  
        }  
        stage('Test') {  
            steps {  
                echo 'Testing...'  
            }  
        }  
        stage('Deploy') {  
            steps {  
                echo 'Deploying...'  
            }  
        }  
    }  
}
```

คัดลอกโค้ด

groovy

```
node {  
    stage('Build') {  
        echo 'Building...'  
    }  
    stage('Test') {  
        echo 'Testing...'  
    }  
    stage('Deploy') {  
        echo 'Deploying...'  
    }  
}
```



Example Declarative Pipeline

Dashboard > declarative-pipeline > Configuration

Pipeline

Configure

General Advanced Project Options Pipeline

Definition

Pipeline script

Script ?

```
1 pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Hello') {  
6             steps {  
7                 echo 'Hello World'  
8             }  
9         }  
10    }  
11}  
12
```

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply



Example Scripted Pipeline

The screenshot shows a Jenkins configuration page for a 'scripted-pipeline' job. The URL in the browser bar is `localhost:8080/job/scripted-pipeline/configure`. The page has a sidebar on the left with 'Configure' sections: General, Advanced Project Options (which is selected and highlighted in blue), and Pipeline. The main content area is titled 'Advanced Project Options' and contains a 'Pipeline' section. Under 'Definition', there is a 'Pipeline script' editor. The script content is as follows:

```
1 node {
2   stage('hello') {
3     echo 'Hello World'
4   }
5 }
```

At the bottom of the editor are 'Save' and 'Apply' buttons.



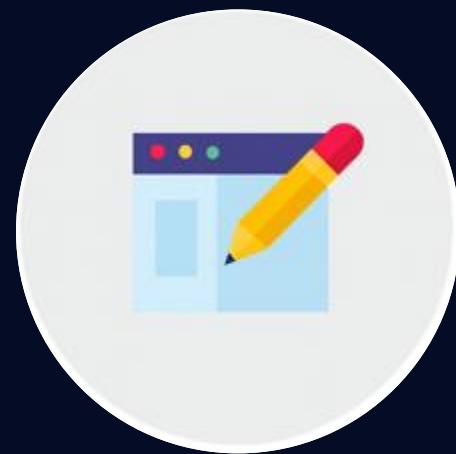
สถาบันไอทีจีเนียส

www.itgenius.co.th | 02-570-8449

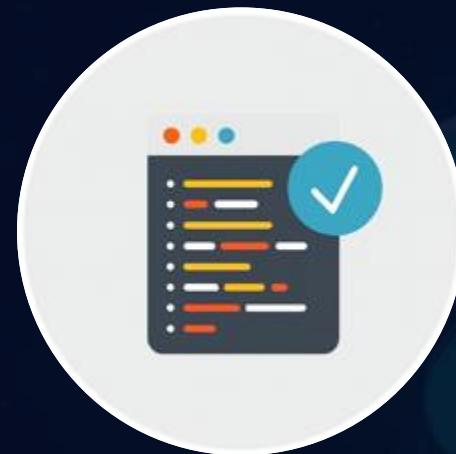
www.itgenius.co.th

IT Genius Engineering | สถาบันไอทีจีเนียส

Jenkins Configuration



Freestyle



Scripted Pipeline



Declarative Pipeline

Jenkins Pipeline with Github

