



Research & Development Team

DevOps Gitlab Jenkins and K8s

www.pnpsw.com

sommai.k@gmail.com

081-754-4663

Lineid : sommaik

Medium: @sommaikrangpanich

สมหมาย กรังพาณิช

ตำแหน่งปัจจุบัน

- กรรมการสมาคมอุตสาหกรรมซอฟต์แวร์ไทย ATSI
- กรรมการผู้จัดการบริษัท พี เอ็น พี โซลูชั่น จำกัด

ด้านความเชี่ยวชาญ

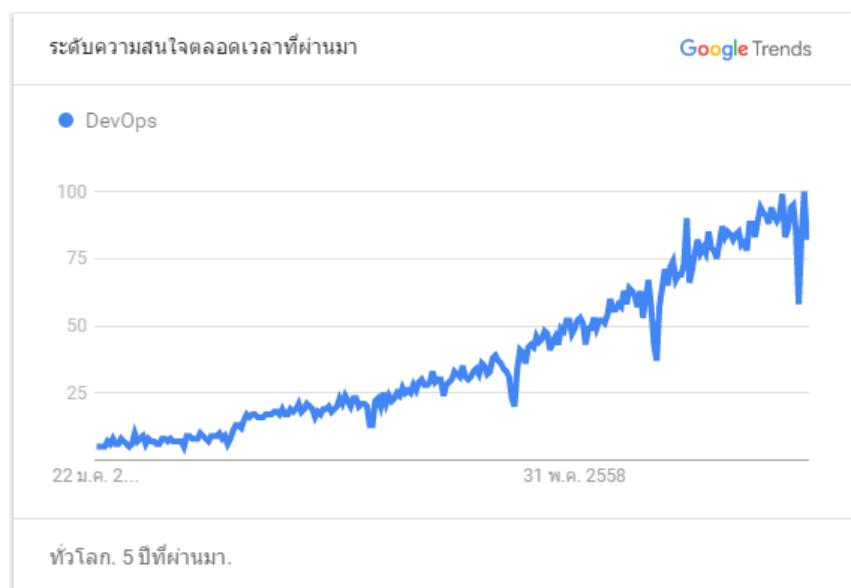
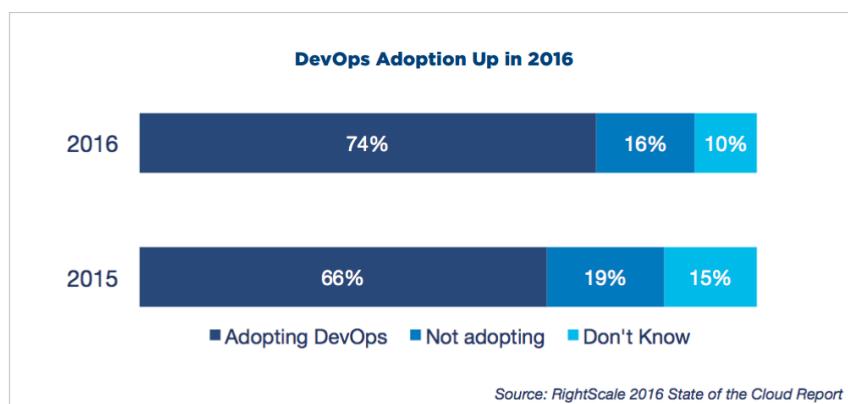
- ผู้เชี่ยวชาญด้านการออกแบบและพัฒนา Software ด้วย Framework ดังนี้
 - Java, Springboot, Go, Node.js
 - Full Stack, Angular, Vue, React, Svelte
 - Apollo GraphQL
 - Docker, Docker Swarm, K8s
 - Flutter, Dart
 - MongoDB, Oracle, MySQL, SQL Server, DB2, PostgreSQL
- ผู้เชี่ยวชาญด้านการออกแบบและประยุกต์ใช้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



Definitions and History

- ในปี ค.ศ. 2009 ที่ Velocity Conference John Allspaw และ Paul Hammond ได้นำเสนอเรื่อง 10 Deploys per Day: Dev and Ops Cooperation at Flickr ซึ่งกล่าวถึงวิธีการสร้างเป้าหมายร่วมกันระหว่างแผนก Development และแผนก Operation และวิธีการที่ทำให้การ Deployment เป็นเรื่องทั่วไปที่กำกันในเซิร์ฟตประจวบัน การนำเสนอเรื่องนี้เป็นแรงบันดาลใจให้ Patrick Debois จัดงาน DevOpsDay ขึ้นมาในปีเดียวกัน คำว่า DevOps ที่ย่อมาจาก Development และ Operations จึงถูกสร้างขึ้นตั้งแต่นั้นเป็นต้นมา
- ปัจจุบัน DevOps เป็นแนวคิดที่มีประสิทธิภาพและแพร่หลายออกไปทั่วโลก จากผลสำรวจของค์กรกว่า 1,000 แห่งจากรายงาน RightScale 2016 State of the Cloud Report: DevOps Trends พบว่าในปี 2016 มีองค์กรนำ DevOps ไปปรับใช้แล้วถึง 74% ซึ่งเพิ่มขึ้นจากปีที่แล้วถึง 8% และจำนวนการ Search คำว่า DevOps ใน google ก็ยังเพิ่มขึ้นเรื่อยๆ ด้วย

ระดับความสนใจ

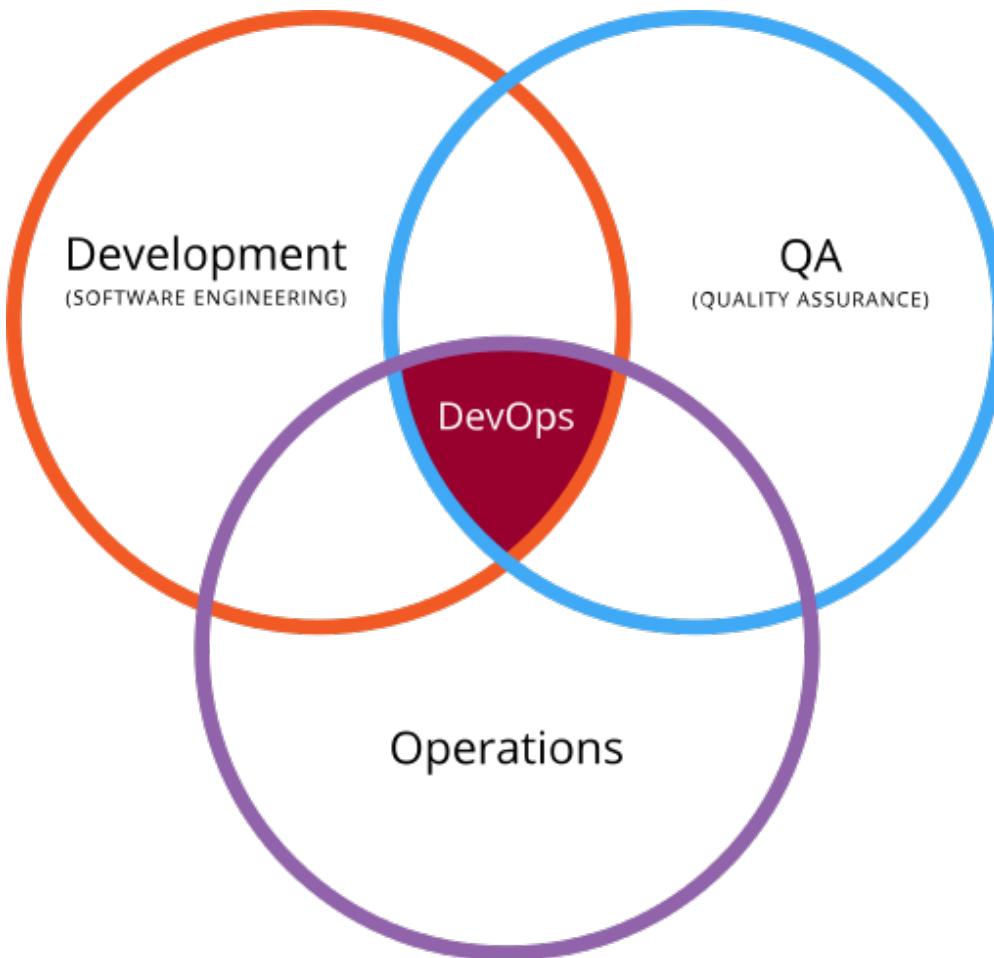


Cultural change

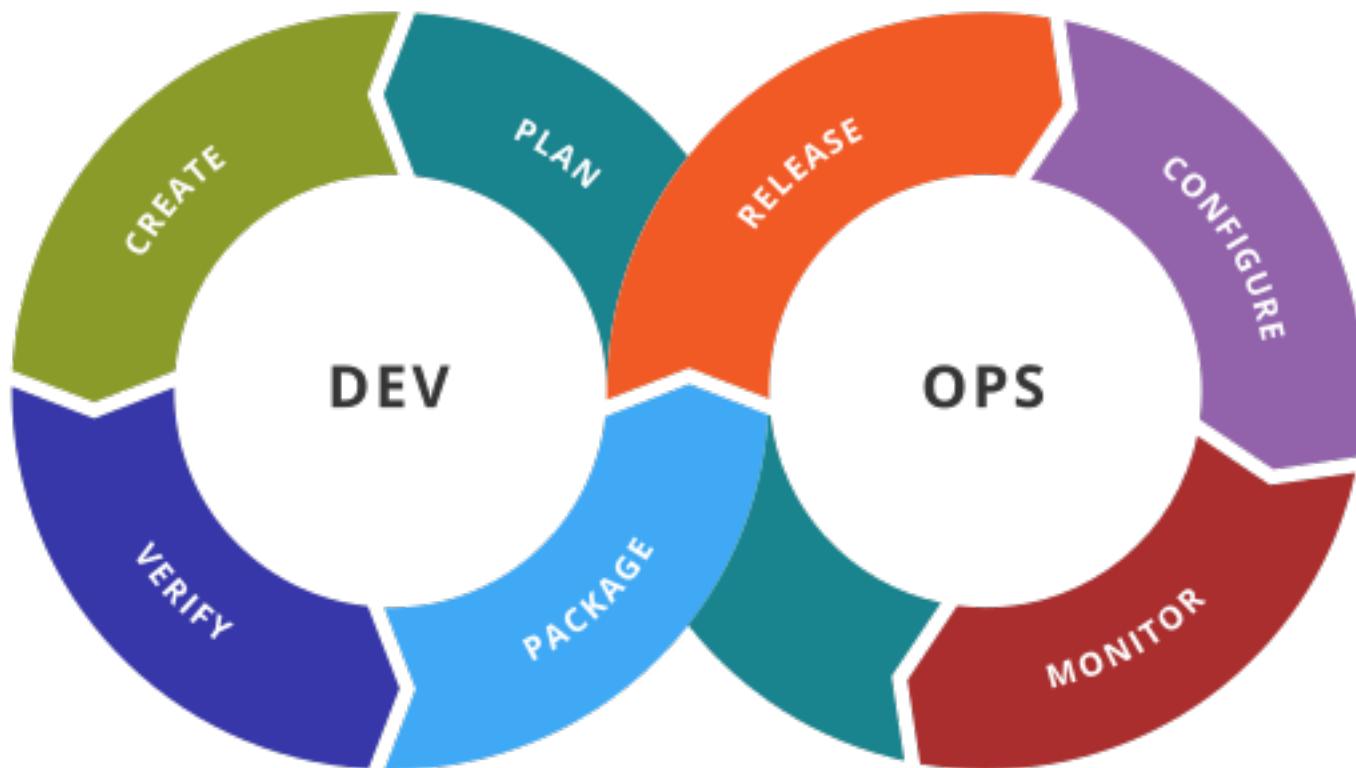
DevOps is more than just a tool or a process change.

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

Overview



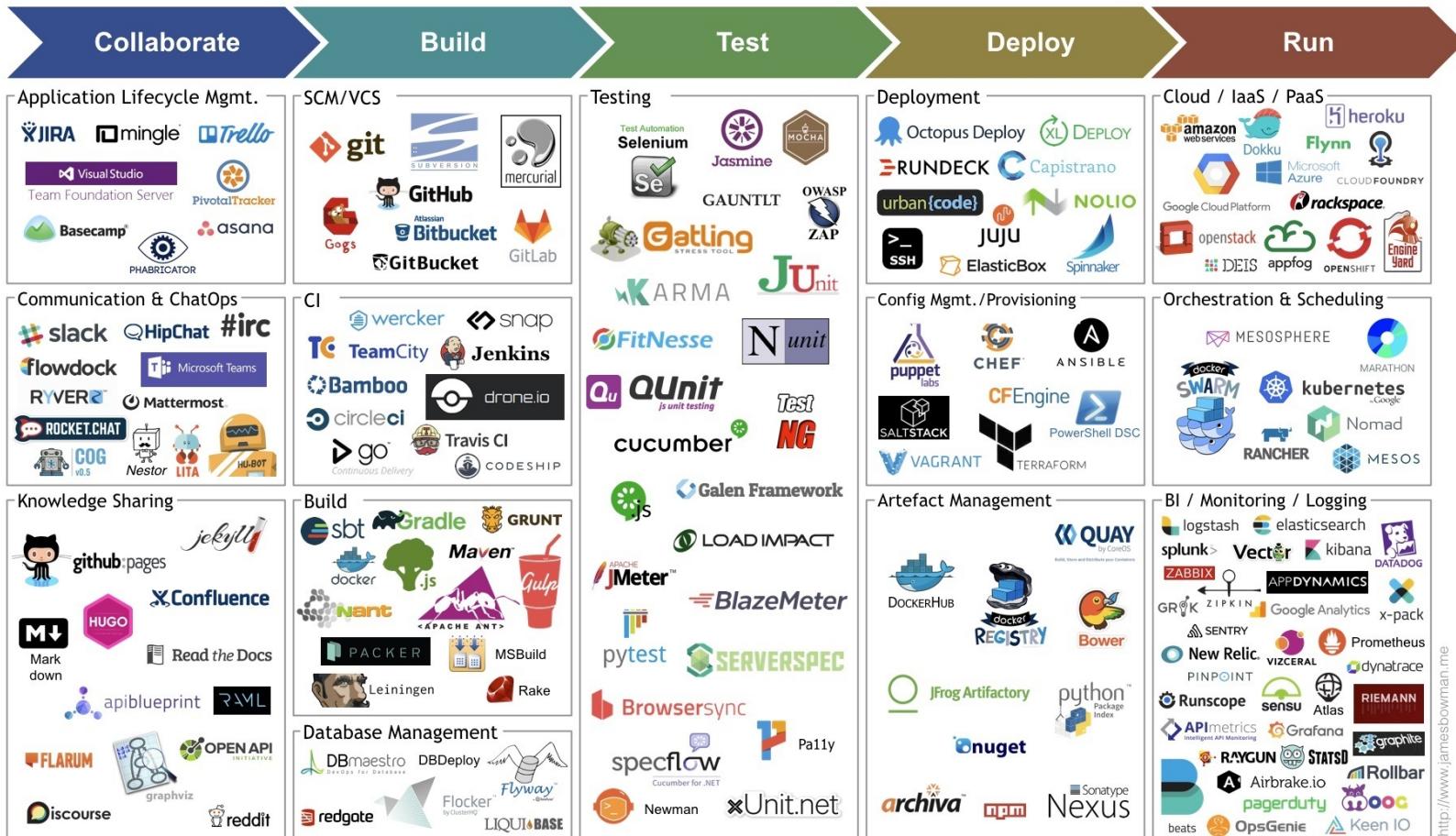
Workflow



7 Ways to Get Started with DevOps

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

DevOps Ecosystem



<http://www.jamesbowman.me>

Collaborate

Application Lifecycle Mgmt.

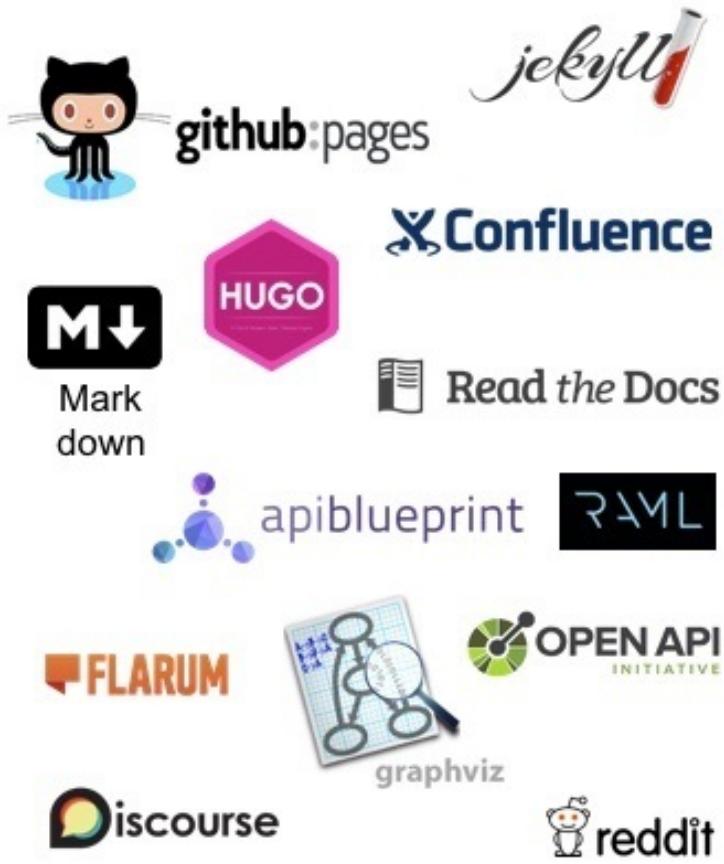


Communication & ChatOps



Collaborate

Knowledge Sharing



Build

SCM/VCS



CI



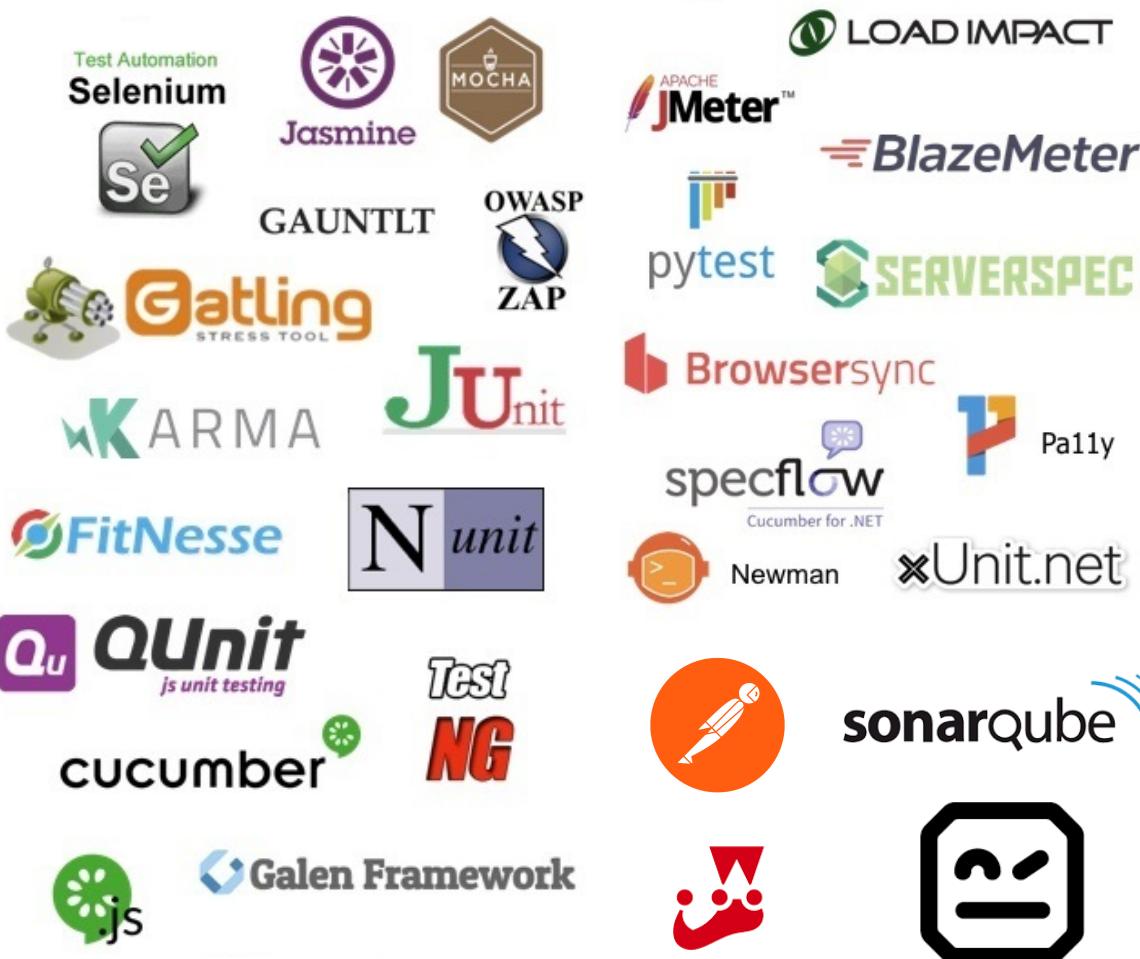
Build



Database Management



Test



Deploy

Deployment



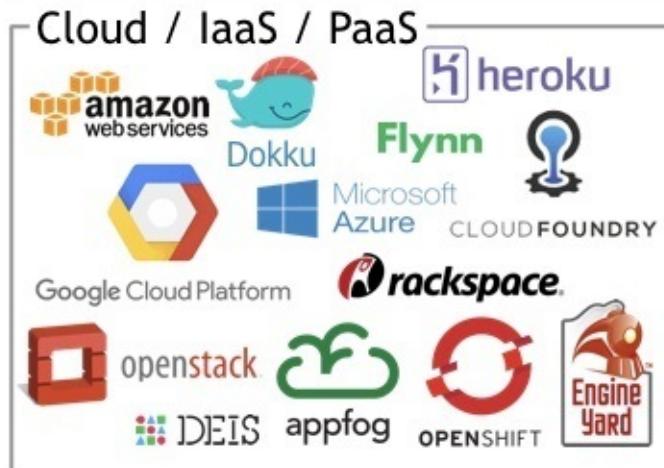
Artifact Management



Config Mgmt./Provisioning



Run



CI/CD

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

Continuous Integration

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

Continuous Delivery

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

Continuous Deployment

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

Software

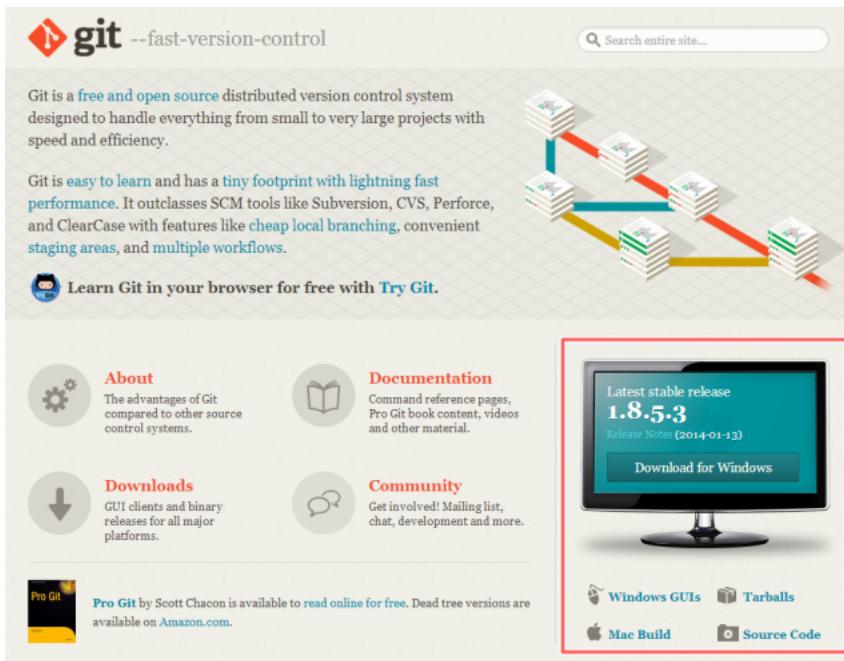
INSTALLATION

การติดตั้ง

GIT FOR WINDOWS

การติดตั้ง GIT สำหรับ Windows

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



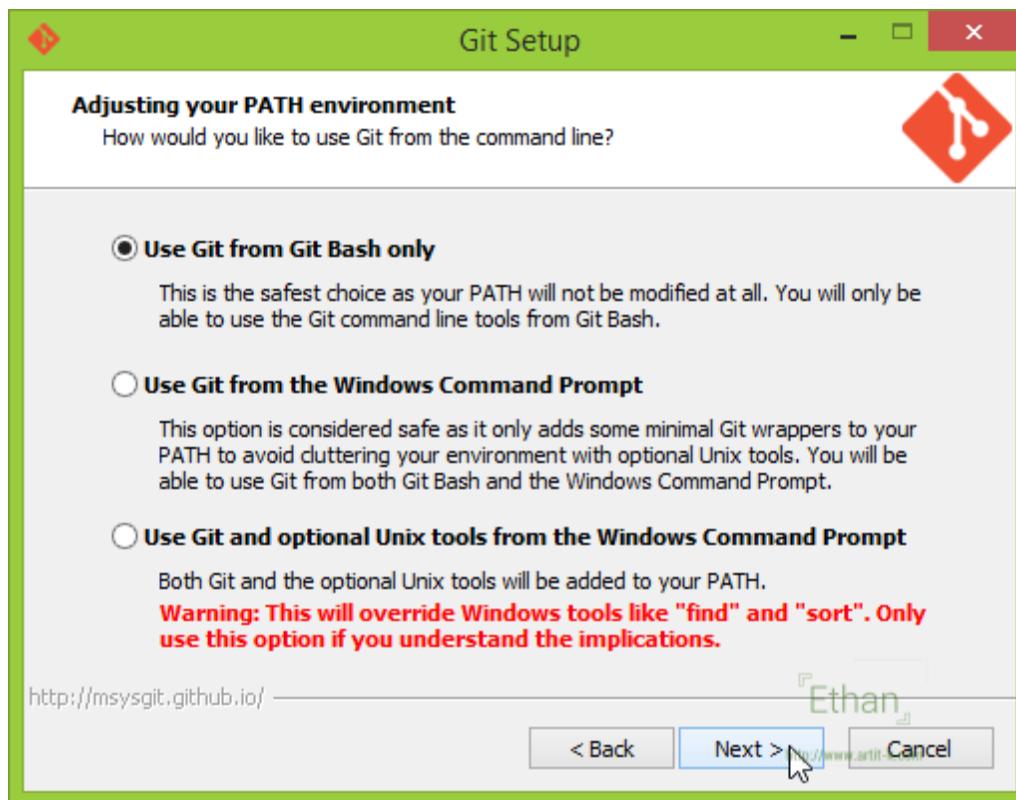
การติดตั้ง GIT สำหรับ Windows #2

- ติดตั้งโดยใช้สิทธิ Administrator ในการติดตั้ง



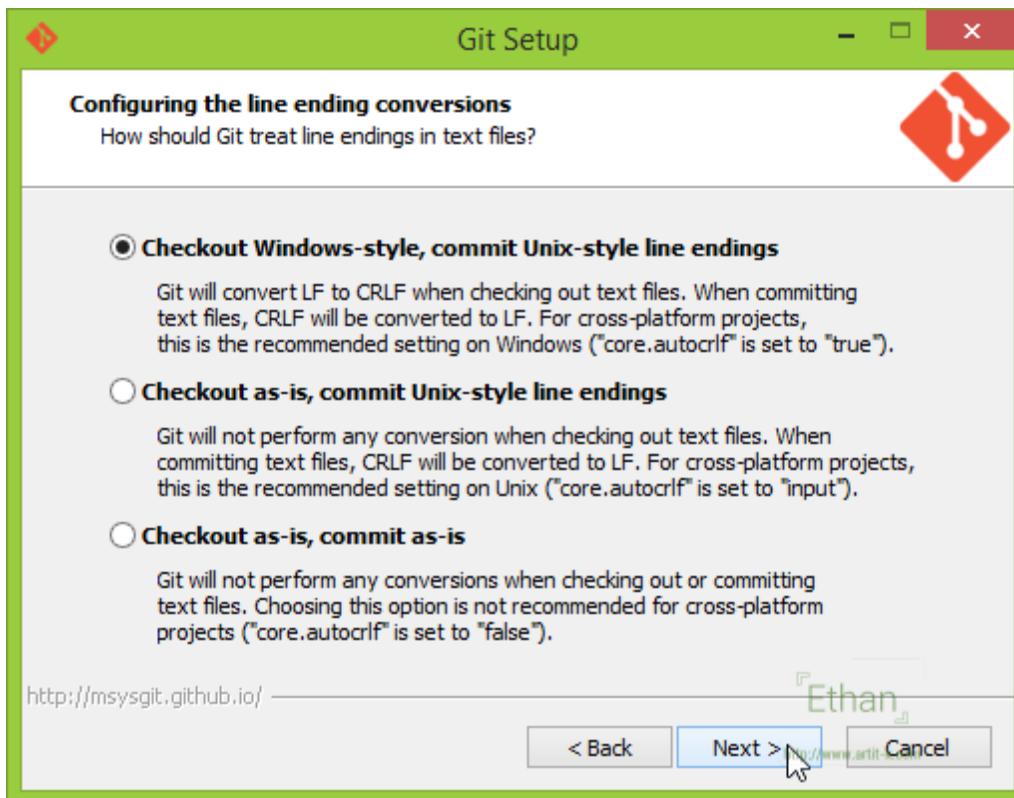
การติดตั้ง GIT สำหรับ Windows #3

- เลือกติดตั้งแบบ **Use Git from the Windows Command Prompt**



การติดตั้ง GIT สำหรับ Windows #4

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



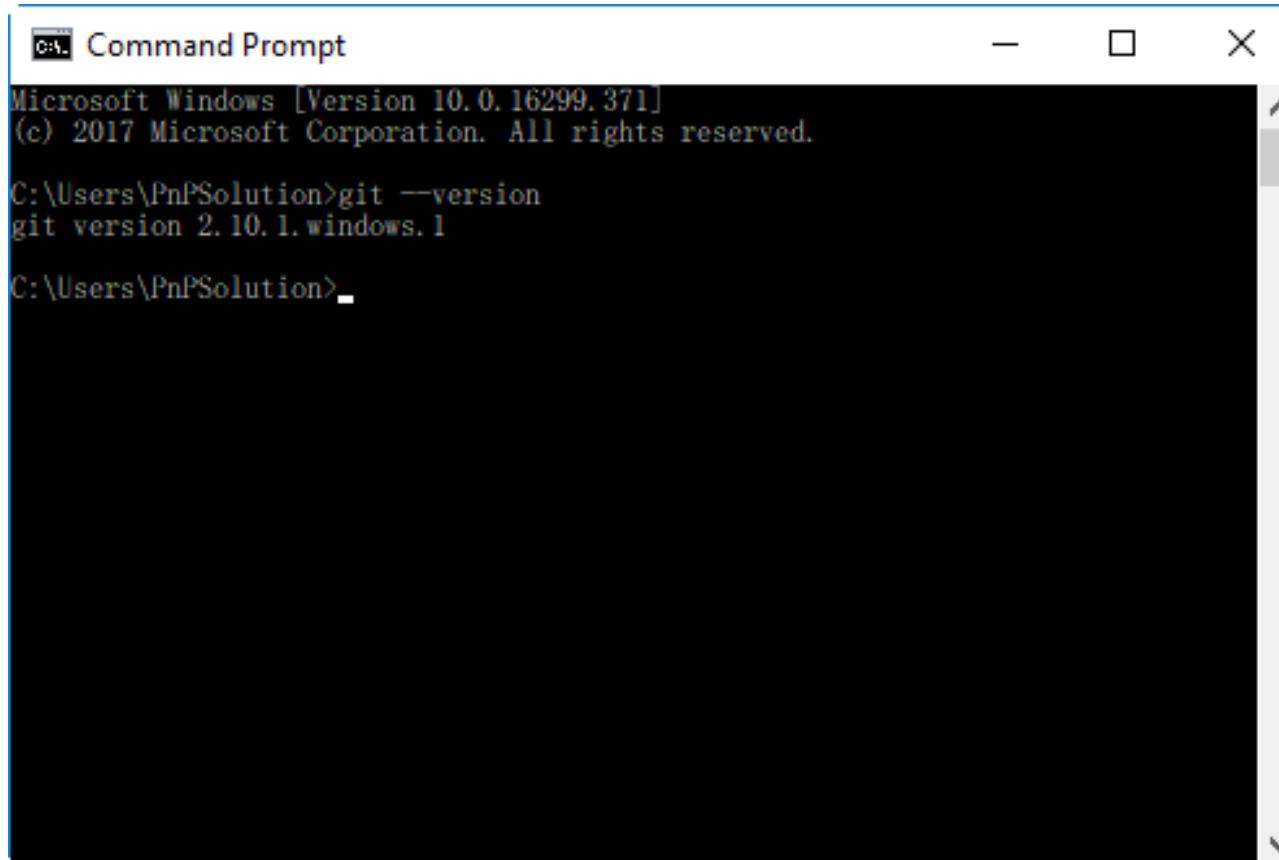
การติดตั้ง GIT สำหรับ Windows #5

- กดปุ่ม next ไปจนถึงหน้าสุดท้าย



ทดสอบหลังการติดตั้ง Git

- เปิดโปรแกรม cmd และพิมพ์คำสั่ง git --version



```
Microsoft Windows [Version 10.0.16299.371]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\PnPSSolution>git --version
git version 2.10.1.windows.1

C:\Users\PnPSSolution>
```

การติดตั้ง

DOCKER DESKTOP

การติดตั้ง Docker สำหรับ Windows

- ความต้องการขั้นพื้นฐานสำหรับการติดตั้ง Docker for windows
 - Windows ต้องเป็น version 64bit เก่ากว่า
 - Windows 10 pro ขึ้นไป
 - Cpu ต้องมีความสามารถ Hyper-v สามารถเปิดได้ที่ BIOS



การติดตั้ง Docker สำหรับ Windows #2

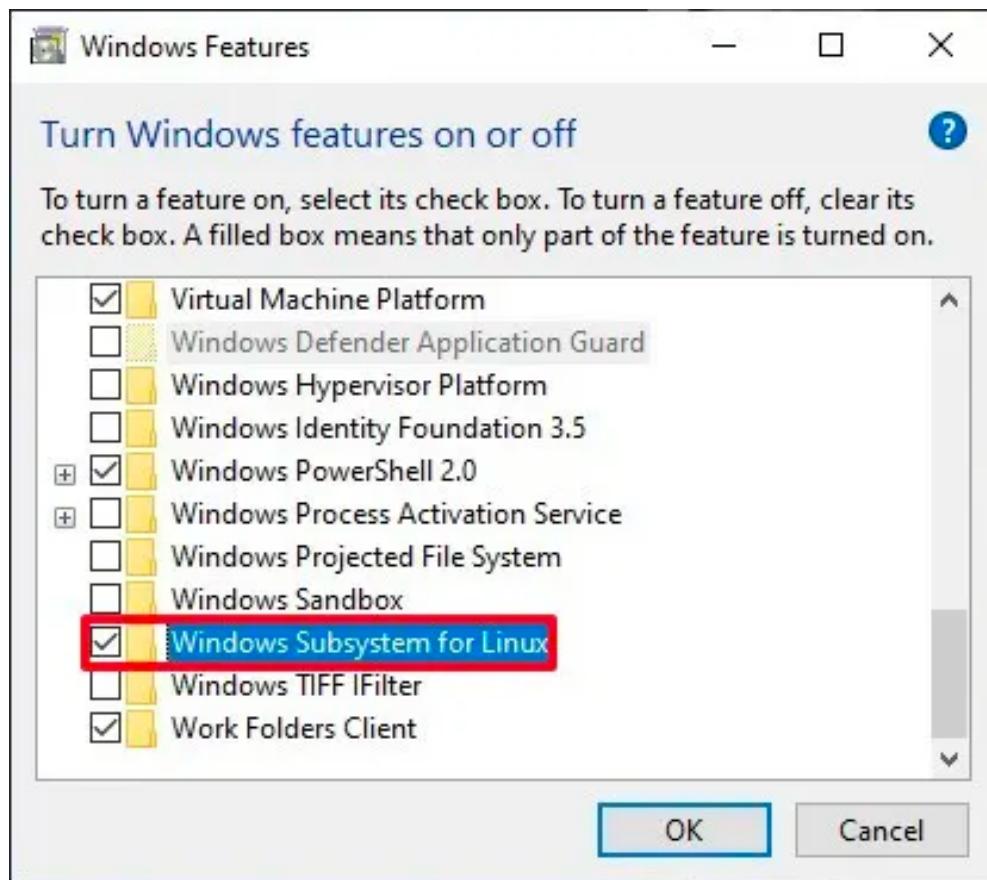
- เข้า link <https://www.docker.com/products/docker-desktop/>
- กด next ไปเรื่อยๆ จนสิ้นสุดการติดตั้ง

การติดตั้ง Docker สำหรับ Windows #2

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

การติดตั้ง Docker สำหรับ Windows #3

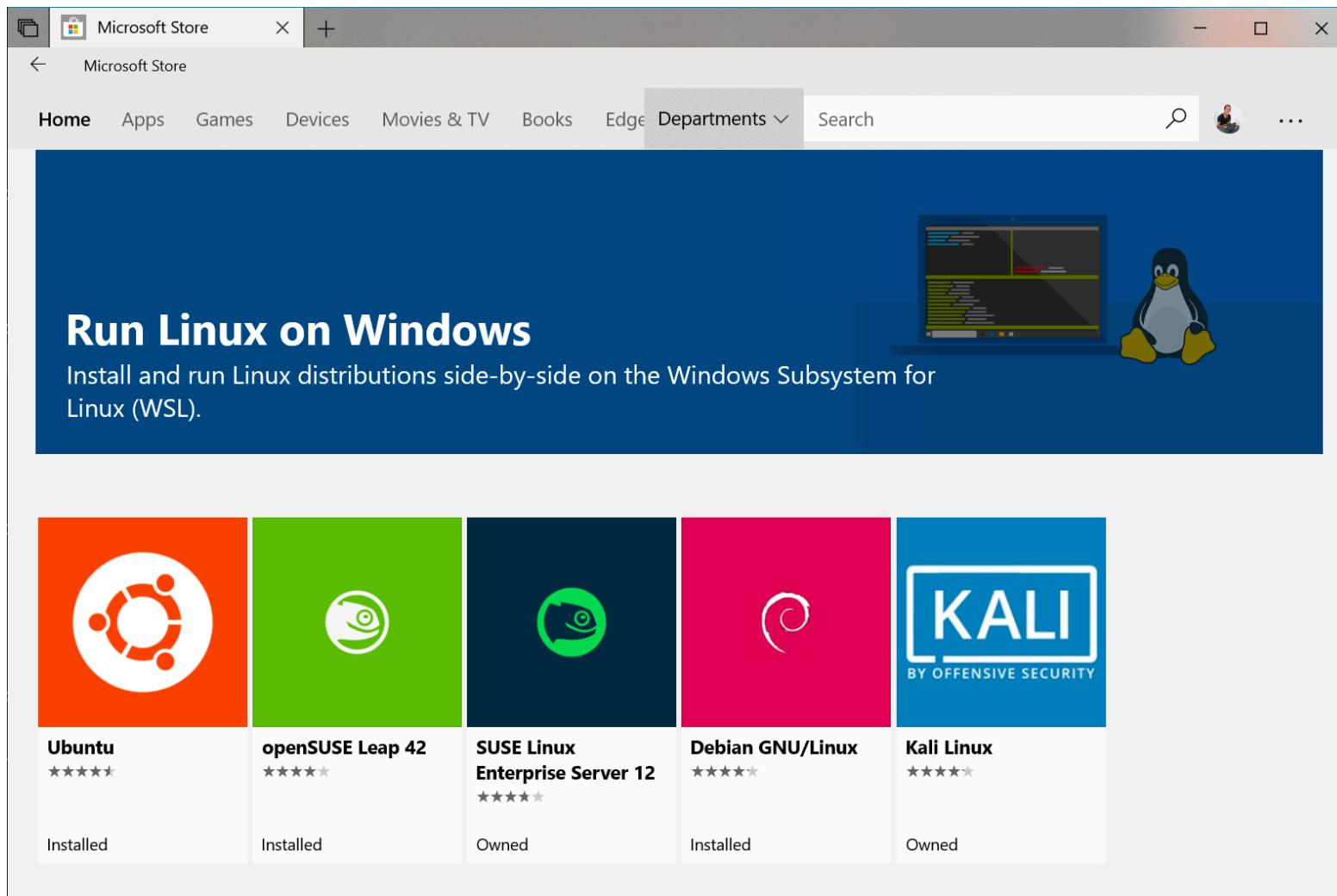
- เปิดใช้ WSL2



การติดตั้ง Docker สำหรับ Windows #4

- run คำสั่งดังต่อไปนี้
 - dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
 - dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
- download and install
 - https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi
- run คำสั่ง wsl --set-default-version 2

ຕាប់តាំង Linux distribution

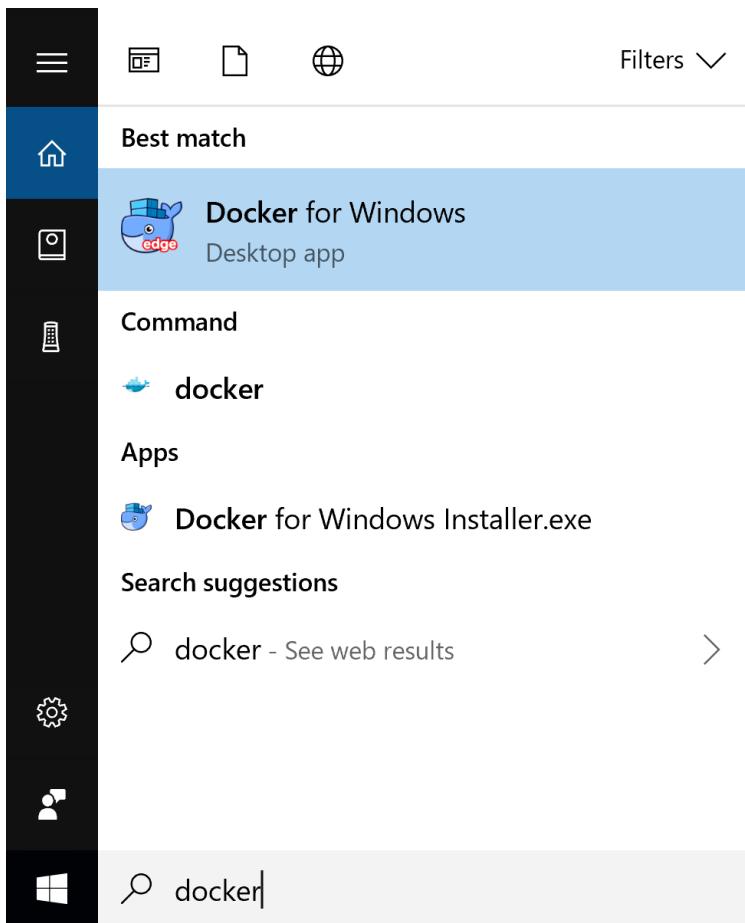


The screenshot shows the Microsoft Store interface with the search bar set to "Microsoft Store". The main content area displays information about running Linux on Windows using the Windows Subsystem for Linux (WSL). It features a large image of a Linux terminal window and the Tux penguin. Below this, five Linux distributions are listed with their logos, names, star ratings, and installation status:

Distribution	Star Rating	Status
Ubuntu	★★★★★	Installed
openSUSE Leap 42	★★★★★	Installed
SUSE Linux Enterprise Server 12	★★★★★	Owned
Debian GNU/Linux	★★★★★	Installed
Kali Linux	★★★★★	Owned

ทดสอบหลังการติดตั้ง #1

- หลังจากติดตั้งเสร็จแล้วให้ทำการเปิด docker ดังนี้



ทดสอบหลังการติดตั้ง #2

- จะมี docker run อยู่ที่ taskbar ดังภาพ



- ทดสอบการติดตั้งโดยการเปิด cmd และพิมพ์คำสั่งดังนี้
- docker ps

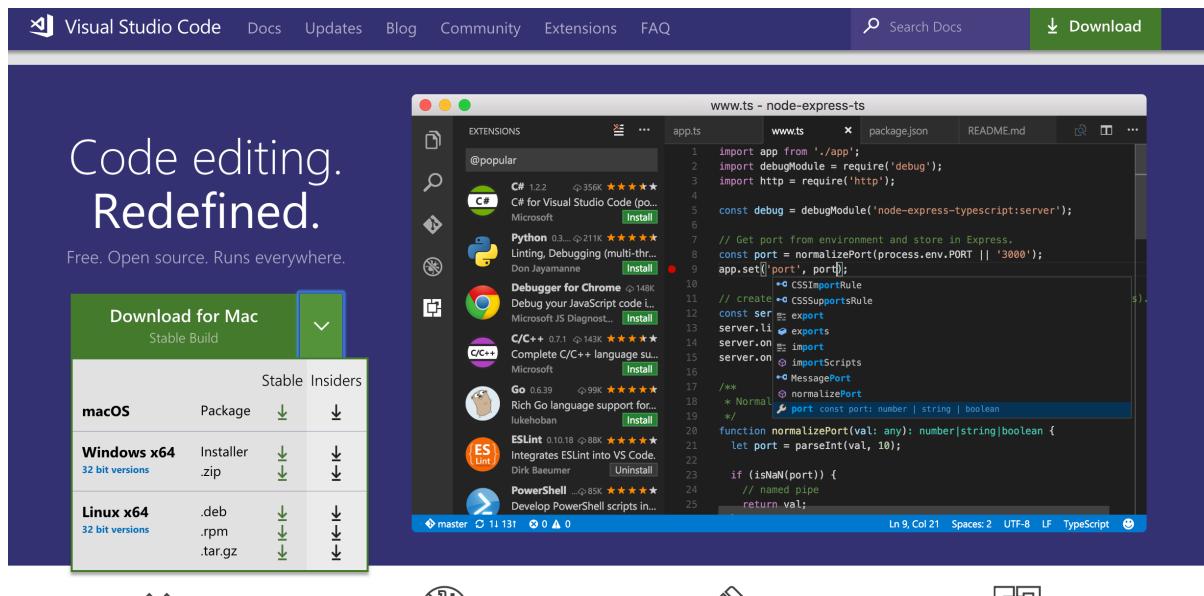
Sommaik-MacBook-Pro:~ sommaik\$ docker ps						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES

การติดตั้ง

VISUAL STUDIO CODE

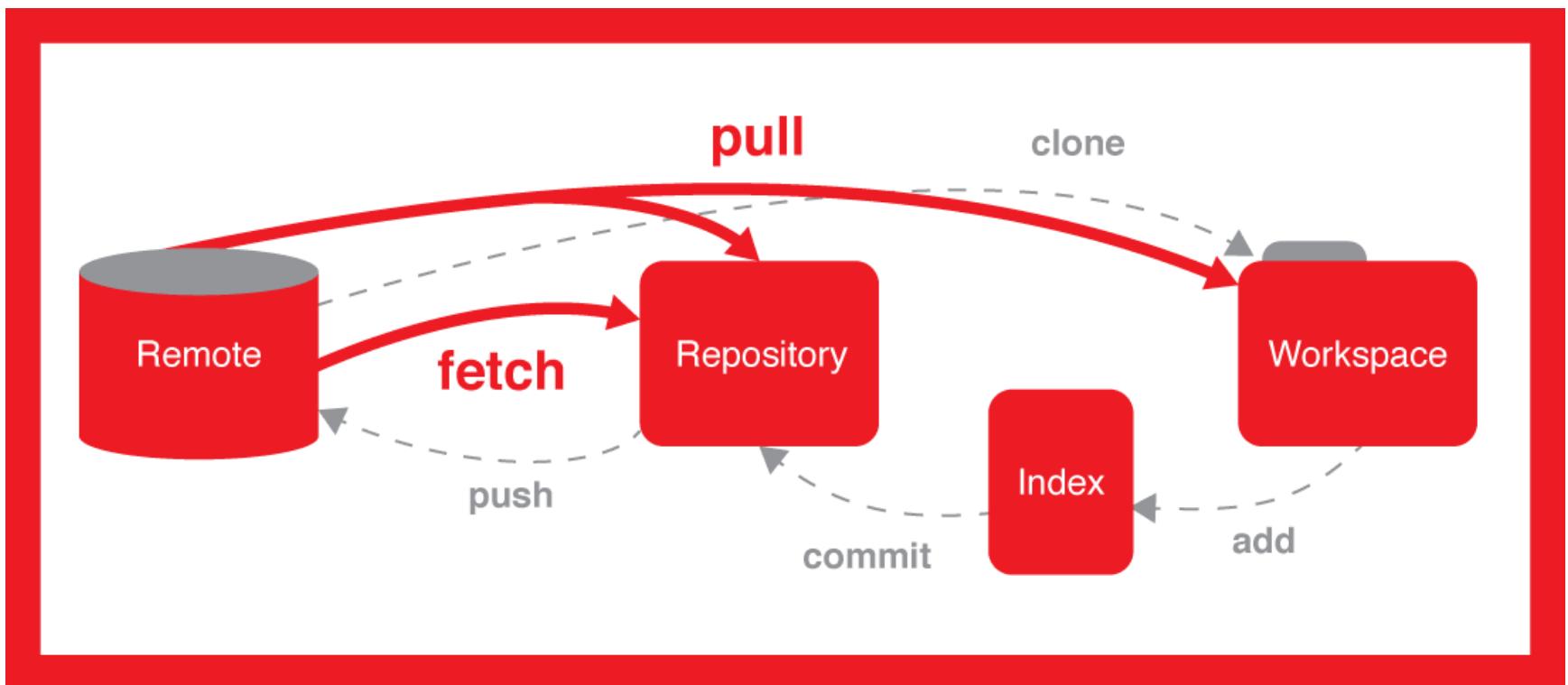
การติดตั้ง VSC

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



Software version control with
GIT

Overview



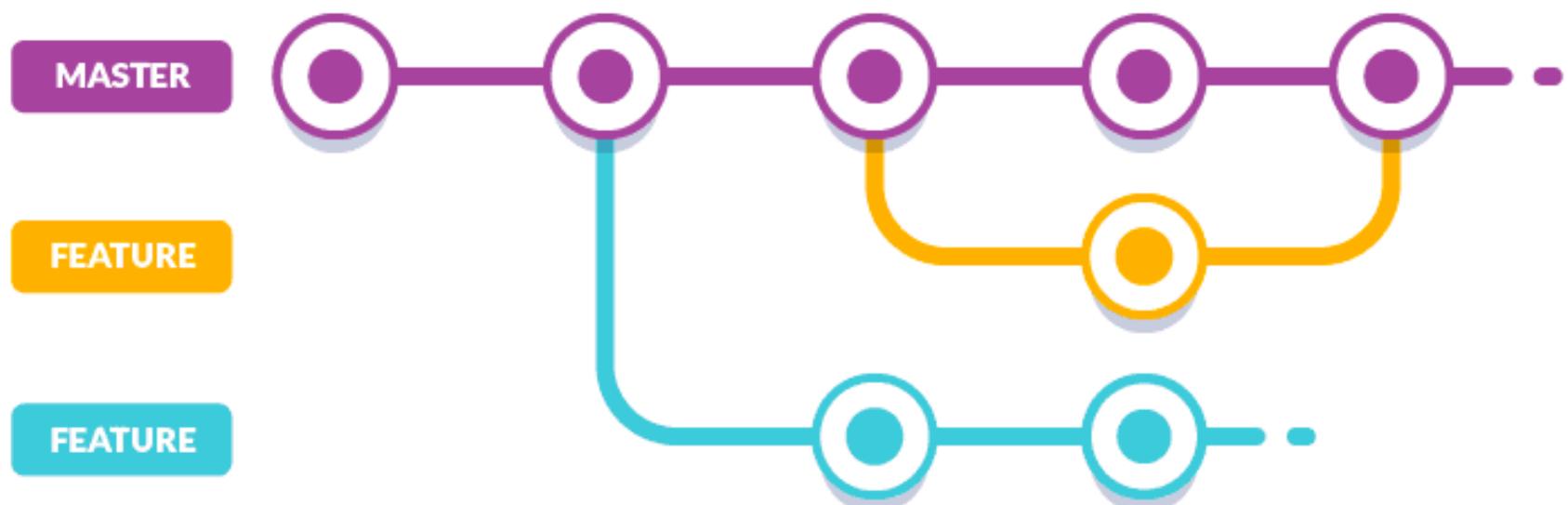
Git workflow type

- Basic ໃໝ່ກັບ Project ຂາດເລືກ
- Feature Branch ໃໝ່ກັບ Project ຂາດກລາງ
- Gitflow ໃໝ່ກັບ Project ຂາດໃຫຍ່

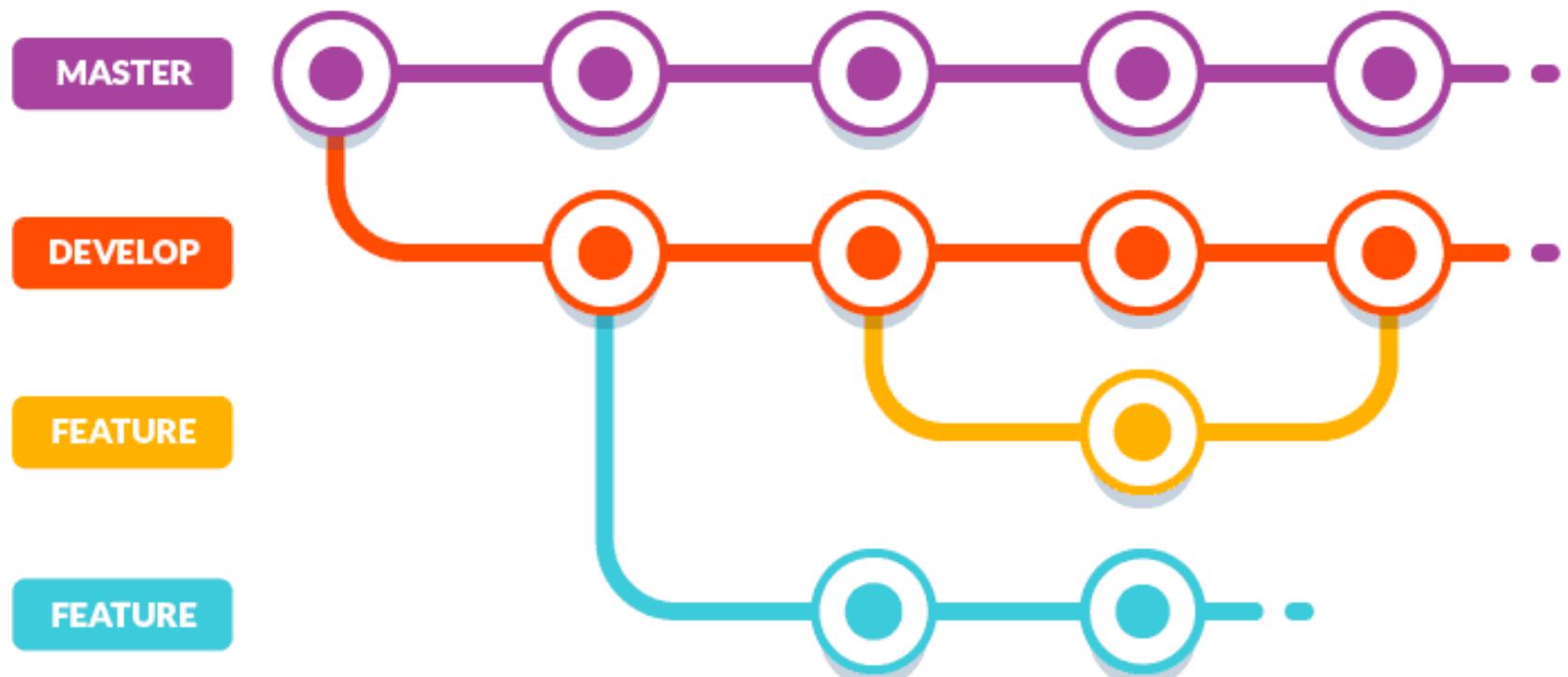
Basic



Feature Branch



Gitflow



Create a new Git Repository (Remote)

Create a new repository

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

Owner Repository name

 Sommaik 

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

Description (optional)

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

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

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

Add .gitignore: **None** Add a license: **None** 

Create repository

CONFIGURE TOOLING

Sets the name you want attached to your commit transactions

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

Sets the email you want attached to your commit transactions

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

Enables helpful colorization of command line output

- git config --global color.ui auto

CREATE REPOSITORIES

Creates a new local repository with the specified name

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

Downloads a project and its entire version history

- git clone [url]
- git clone <https://github.com/Sommaik/XXX.git>

MAKE CHANGES

Lists all new or modified files to be committed

- git status

Shows file differences not yet staged

- git diff

Snapshots the file in preparation for versioning

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

Shows file differences between staging and the last file version

- git diff --staged

MAKE CHANGES

Unstages the file, but preserve its contents

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

Records file snapshots permanently in version history

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

GROUP CHANGES

Lists all local branches in the current repository

- git branch

Creates a new branch

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

Switches to the specified branch and updates the working directory

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

GROUP CHANGES

Combines the specified branch's history into the current branch

- git merge [branch]
- git merge XXX

Deletes the specified branch

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

REFACTOR FILENAMES

Deletes the file from the working directory and stages the deletion

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

Removes the file from version control but preserves the file locally

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

Changes the file name and prepares it for commit

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

SUPPRESS TRACKING

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

- *.log
- build/
- temp-*

Lists all ignored files in this project

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

SAVE FRAGMENTS

Temporarily stores all modified tracked files

- git stash

Restores the most recently stashed files

- git stash pop

Lists all stashed changesets

- git stash list

Discards the most recently stashed changeset

- git stash drop

REVIEW HISTORY

Lists version history for the current branch

- git log

Lists version history for a file, including renames

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

Shows content differences between two branches

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

Outputs metadata and content changes of the specified commit

- git show [commit]
- git show XXX

REDO COMMITS

Undoes all commits after [commit], preserving changes locally

- git reset [commit]
- git reset XXX

Discards all history and changes back to the specified commit

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

SYNCHRONIZE CHANGES

Downloads all history from the repository bookmark

- git fetch [bookmark]
- git fetch origin

Combines bookmark's branch into current local branch

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

SYNCHRONIZE CHANGES

Uploads all local branch commits to Git

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

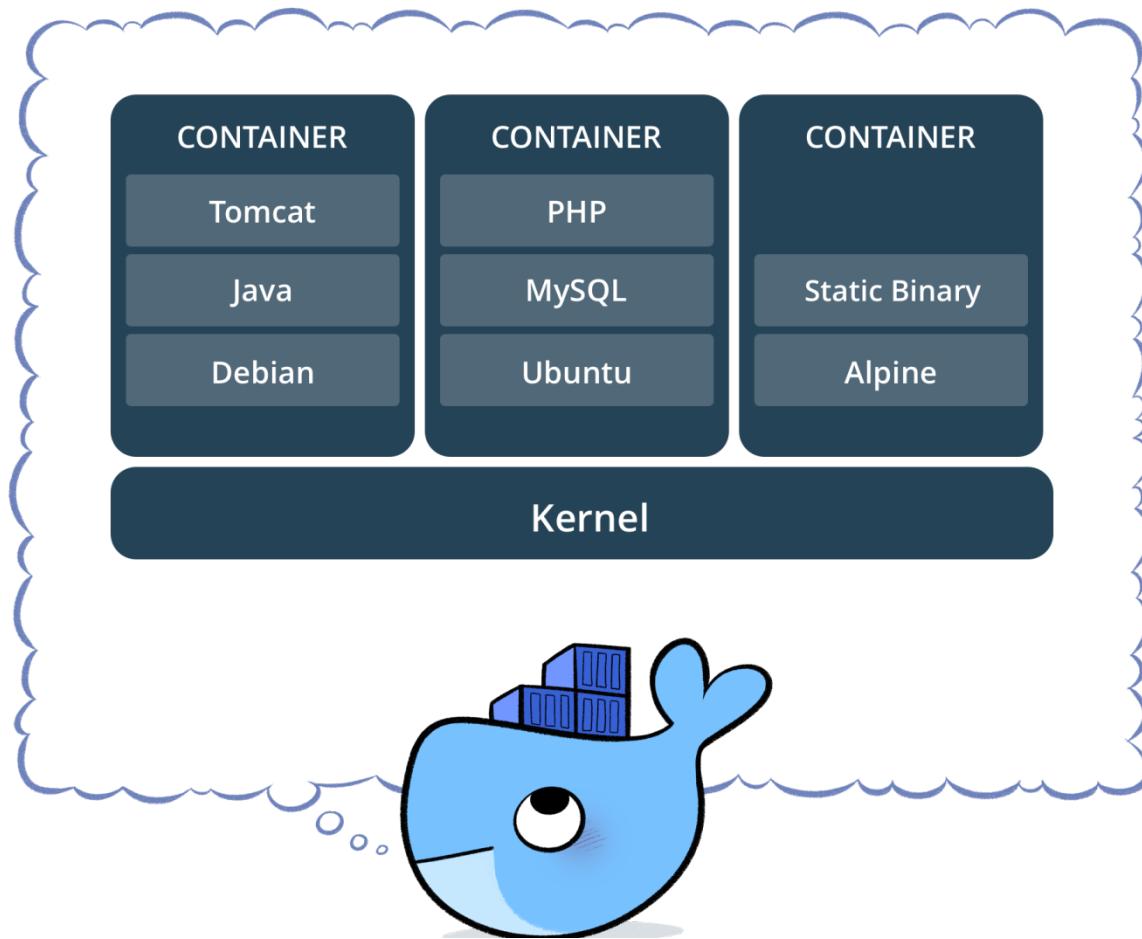
Downloads bookmark history and incorporates changes

- git pull

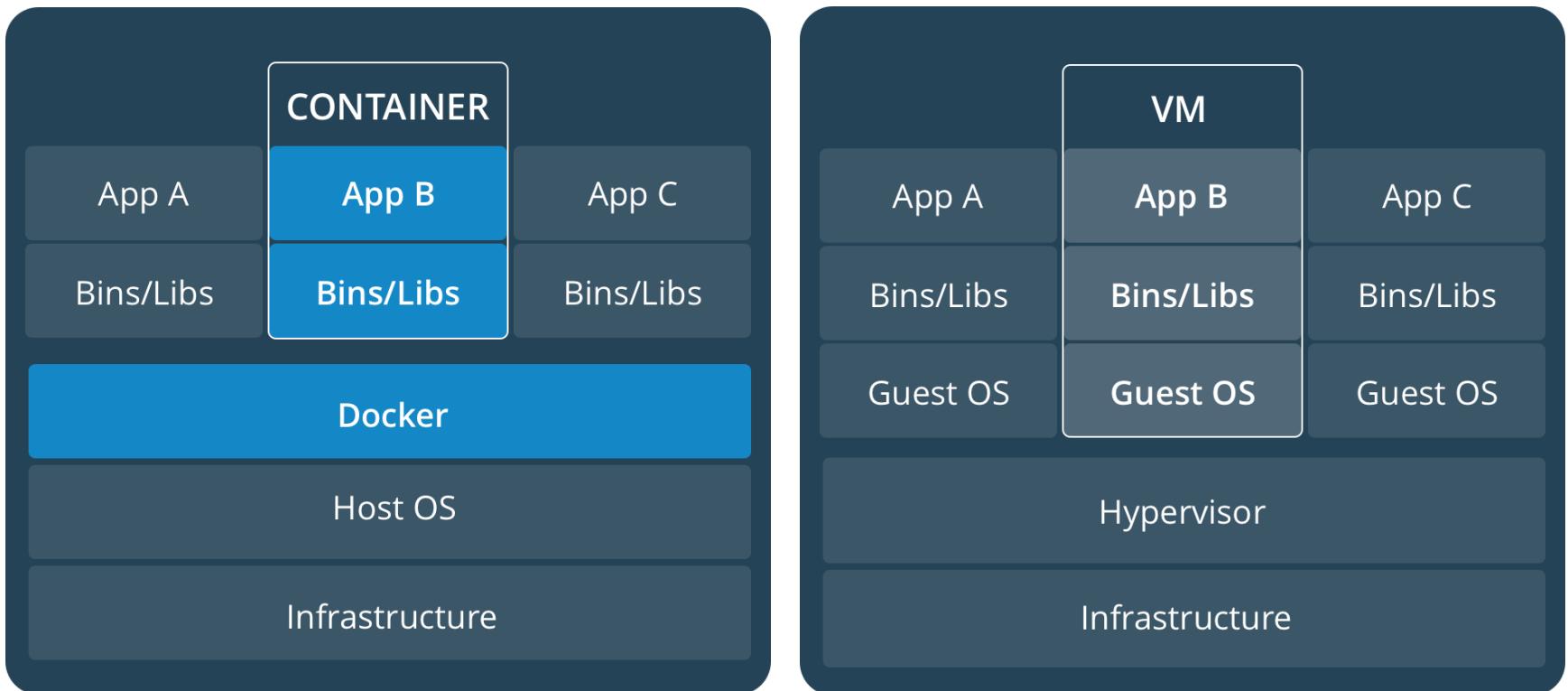
Containers virtualize with

DOCKER

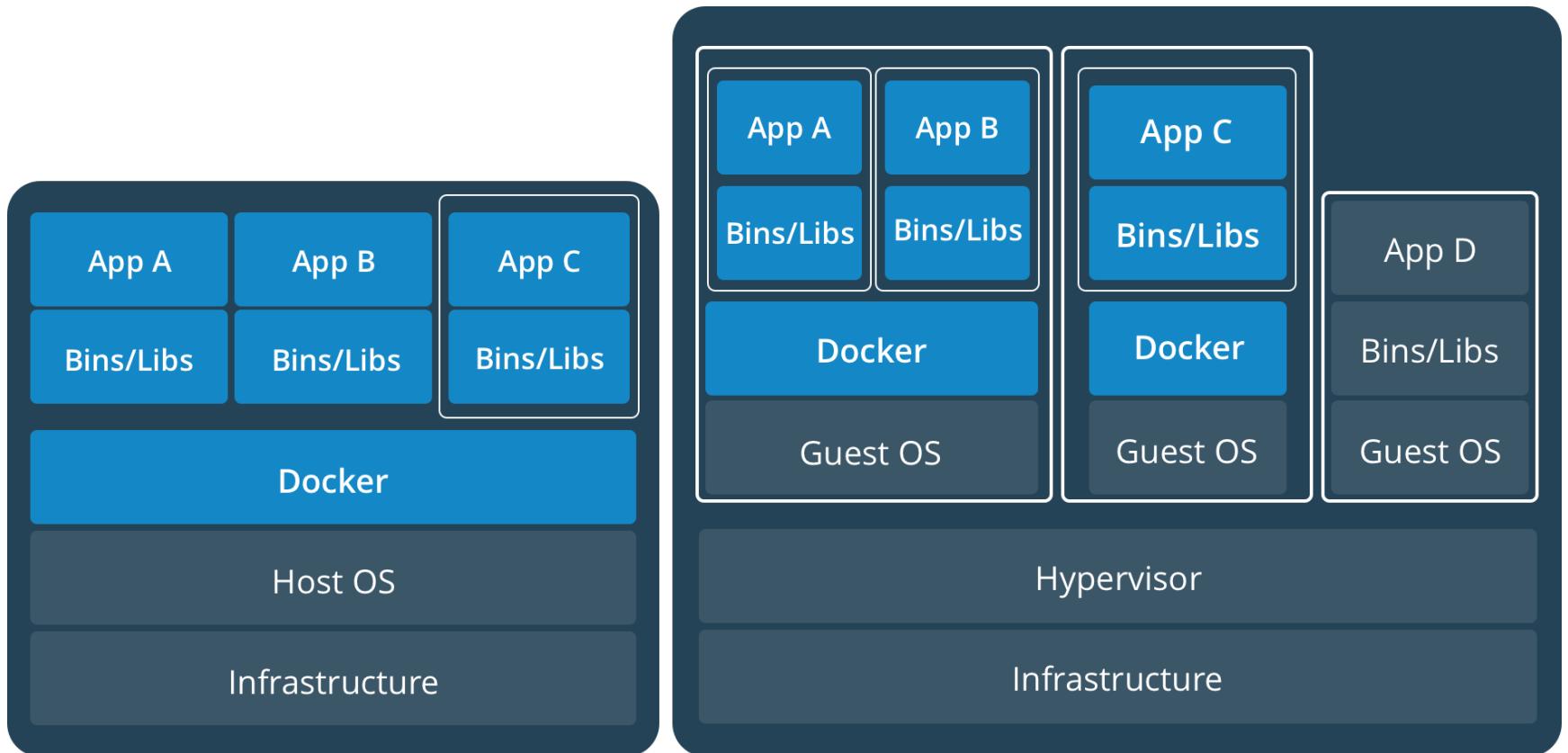
About Container



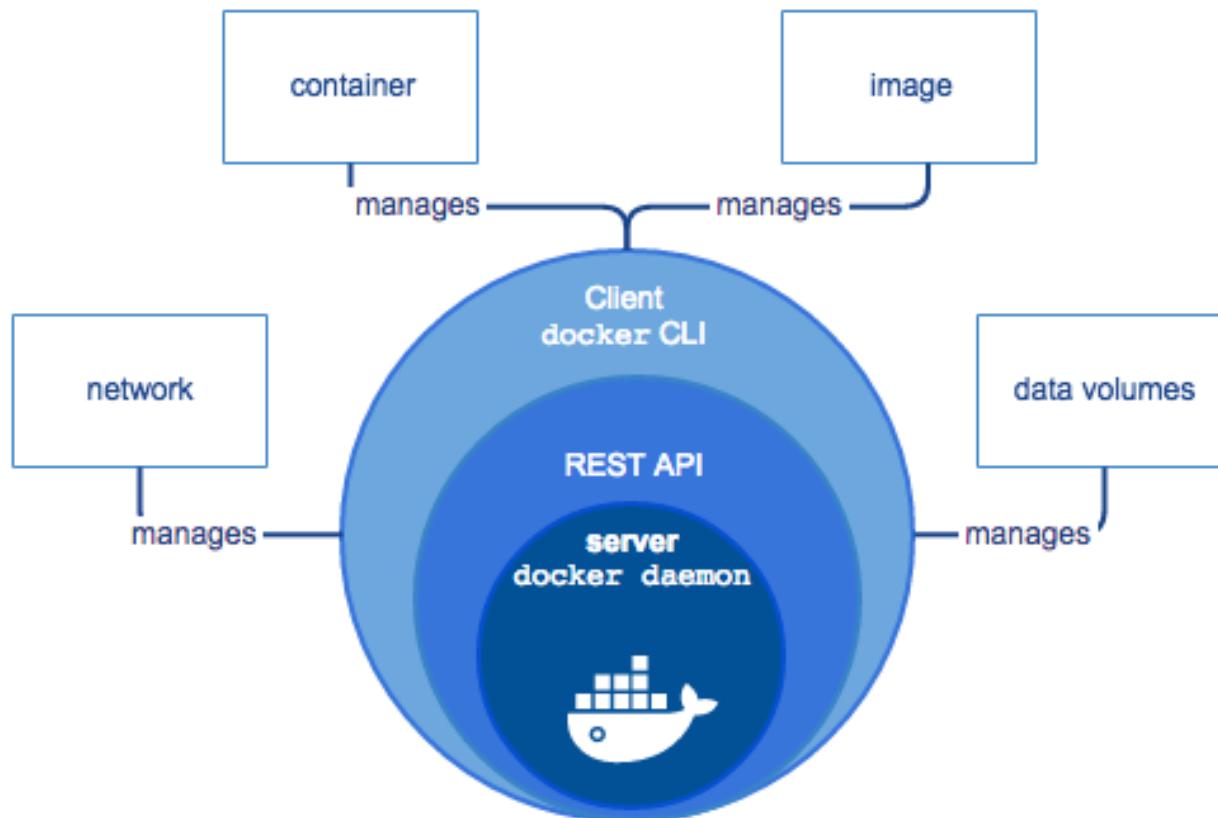
Comparing Containers and Virtual Machines



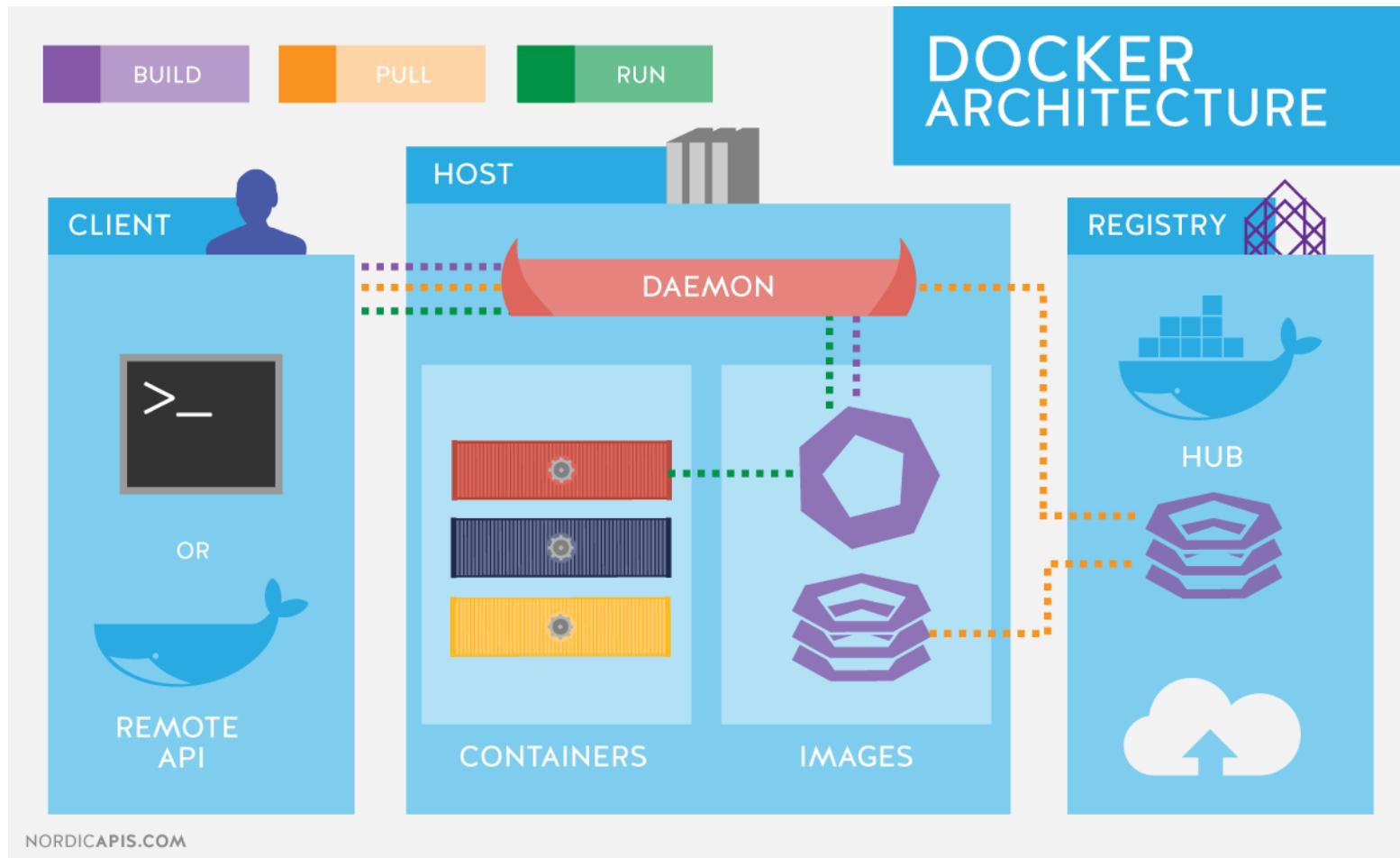
Containers and Virtual Machines Together



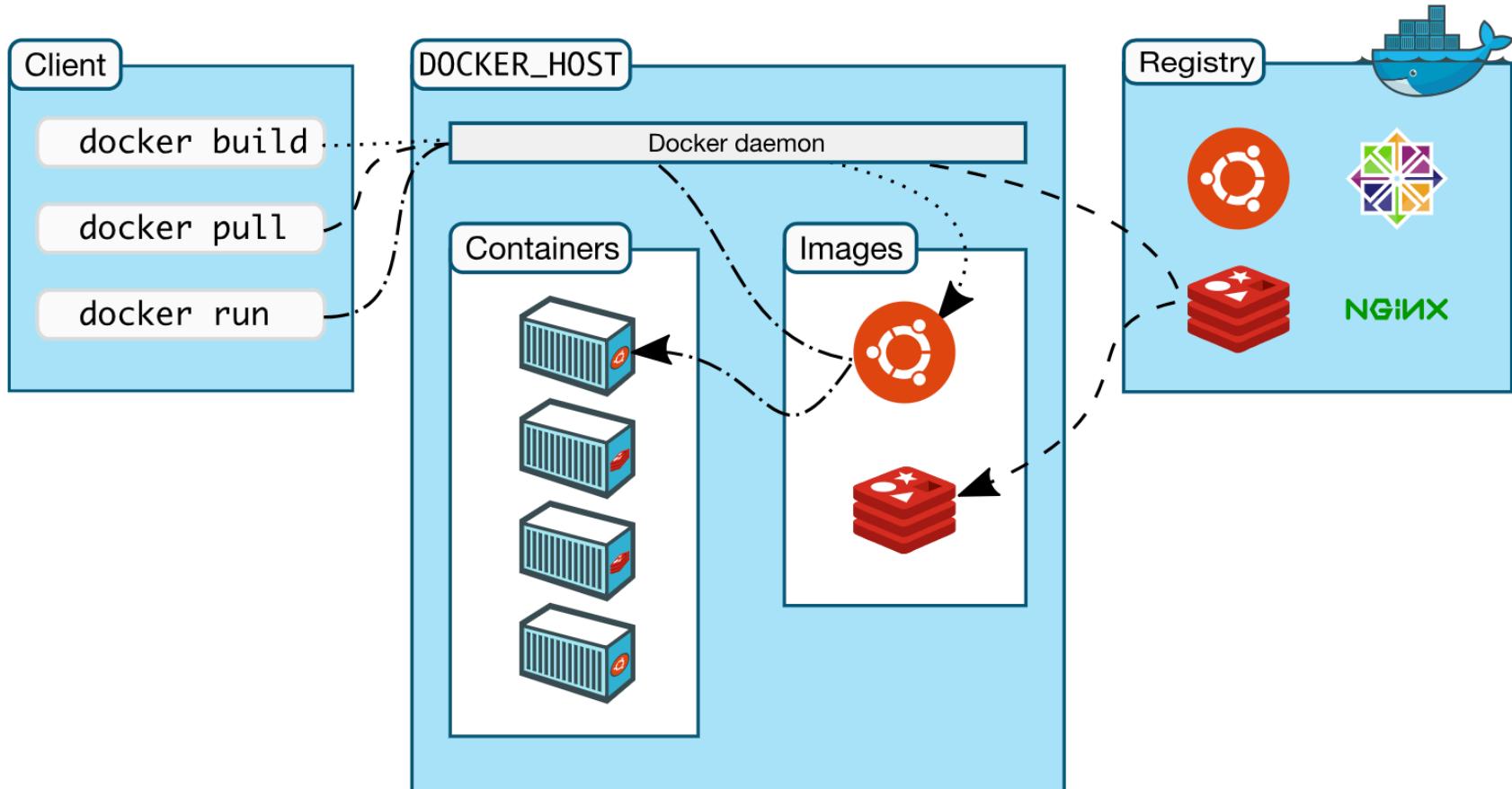
Docker engine



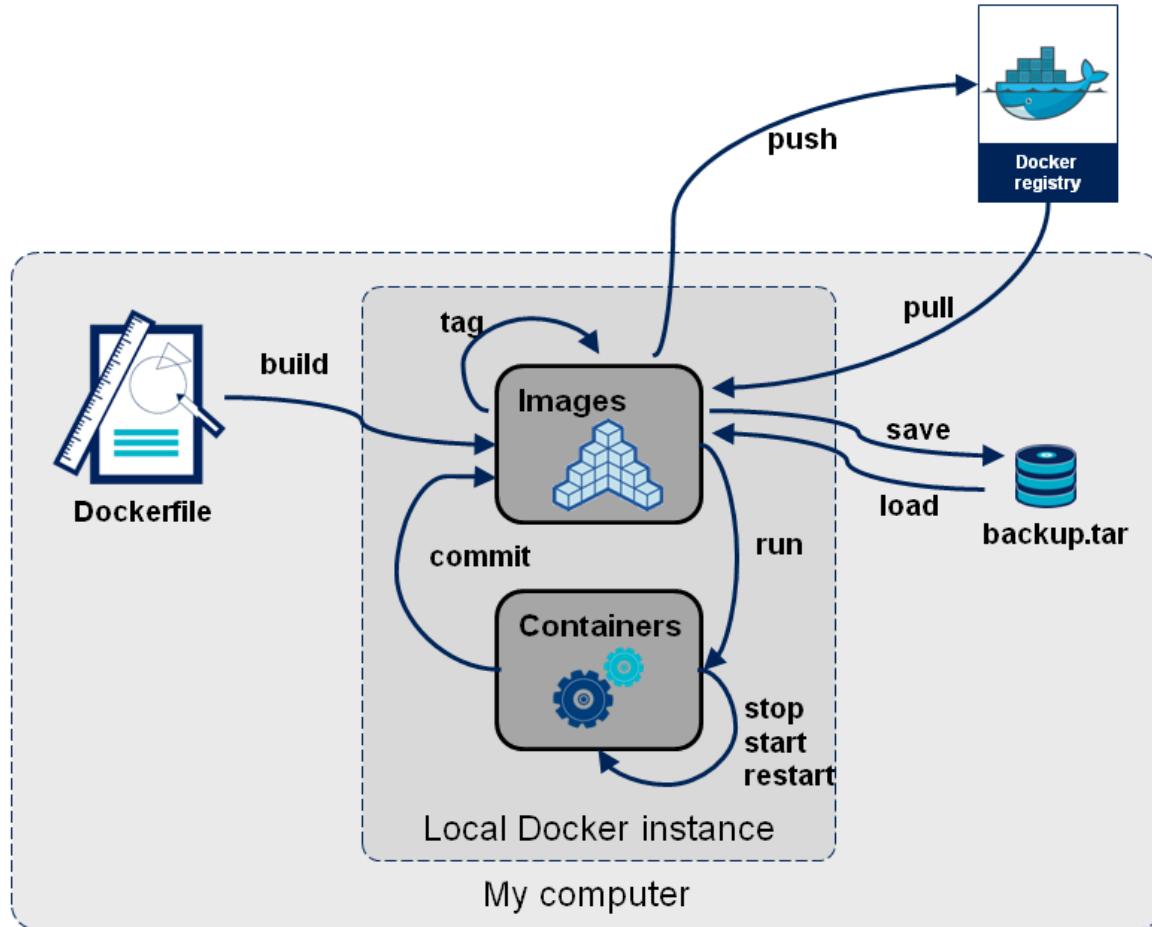
Docker Architecture



Docker Architecture#2



Docker Architecture#3



Hello World Docker

- เปิด cmd แล้วรันคำสั่งดังนี้

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

*** link สำหรับหา image <https://hub.docker.com>

สร้าง file index.html

The screenshot shows a dark-themed code editor interface. On the left is a vertical toolbar with icons for file operations (New, Open, Save, Find, Replace, Cut, Copy, Paste, Undo, Redo) and a search function. To the right of the toolbar is the 'EXPLORER' panel, which lists two entries under 'OPEN EDITORS': 'index.html' and 'TRAIN'. Below these are two entries under 'TRAIN': 'index.html' and another 'index.html'. The main workspace is titled 'index.html' and contains the following code:

```
1 <h1>Hello World</h1>
2
```

เข้า url `http://localhost`



Hello World

Docker Command

Login

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

Logout

- docker logout

List all image

- docker images
- docker image ls

Docker Command

Search image

- docker search <image name>

Pull image

- docker pull <image name>

Create container from image

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

Docker Command

Start Container

- docker start <container_id> or <container_name>

Stop Container

- docker stop <container_id> or <container_name>

Stop all container

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

List all container

- docker ps <options>

Docker Command

Pause Container

- docker pause <container_id> or <container_name>

Unpause Container

- docker unpause <container_id> or <container_name>

Exec Container

- docker exec -it <container_id> bash

Inspect Container

- docker inspect <container_id>

Docker Command

Logs container

- docker logs

Commit Container

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

Push Image

- docker push <account>/<image name>

Tag

- docker tag ubuntu ubuntu-x

Docker Command

Export container

- docker export <container_id> > <to_path>

Import container

- docker import - <from_path>

Save Image

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

Load Image

- docker load < <from path>

Docker Command

Remove container

- docker rm <container_id>

Remove all stop container

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

Remove Image

- docker rmi <image_id>

Docker Network

- docker network ls
- docker network create <network_name> default bridge
- docker network create --subnet 10.0.0.1/24 <network_name>
- docker network inspect <network_name> or <container_id>
- docker network create my-net (create images networks)
- docker run --network <network_name> <image_name>
- docker run -it --name <container_name> --net--alias alias2 --network <network_name> <image_name>

Docker parameter

Run in the background

- -d

Create name to container is running

- --name

Port mapping

- -p (local_port:container_port)

Container host name

- -h

Docker parameter

Environment

- -e

Map volume paths

- -v

Keep STDIN open even if not attached

- -i

Allocate a pseudo-TTY

- -t

Dockerfile

FROM

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

RUN

- RUN <command>
- RUN echo “Hello World”

EXPOSE

- EXPOSE <port>
- EXPOSE 8080

COPY

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

Dockerfile

ENV

- ENV <key> <value>

CMD

- CMD command param1 param2

WORKDIR

- WORKDIR /path/to/workdir

VOLUME

- VOLUME /path

Dockerfile

Build

- docker build <option> <path>
 - -t tag name

Example

- docker build -t first .

Compose file

file name

- docker-compose.yaml

Example

```
version: '3'
services:
  jenkins:
    container_name: jenkins
    image: jenkins
    volumes:
      - ./jenkins:/var/jenkins_home
    ports:
      - 8080:8080
      - 5000:5000
  ubuntu:
    container_name: ubuntu14
    image: "ubuntu:14.04"
```

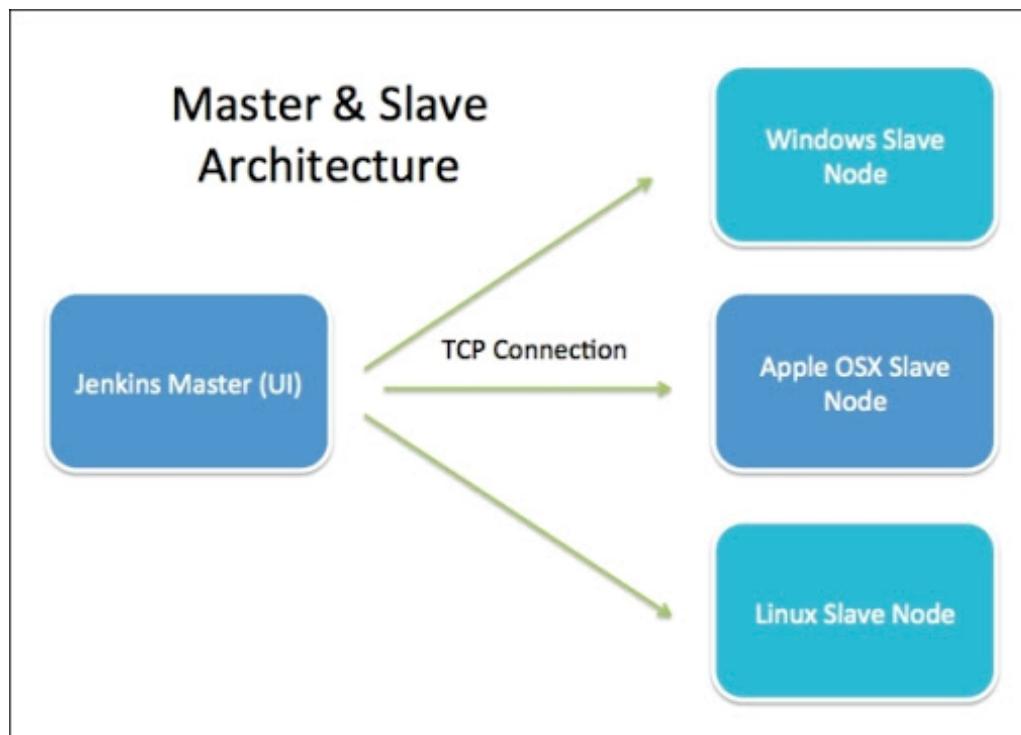
Docker Compose

- docker-compose up -d –build
- docker-compose up --force-recreate
- docker-compose ps
- docker-compose scale web=5
- docker-compose stop
- docker-compose rm

Automate Build and Deploy with

JENKINS

Master & Slave Architecture



Unlock Jenkins

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

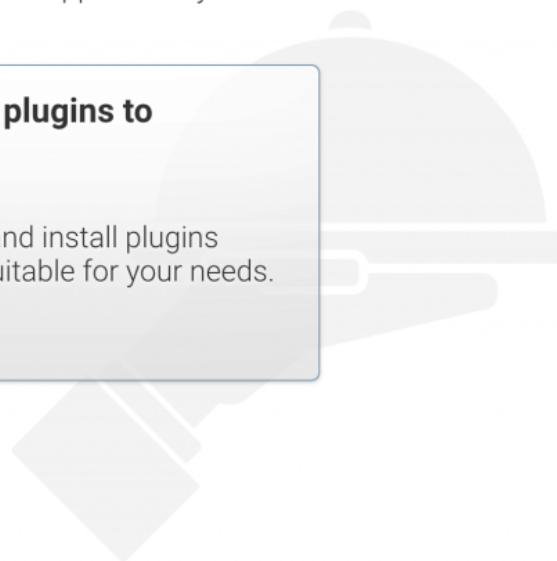
Continue

Install Plugins

Getting Started X

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 2.46.1

Loading Plugins

Getting Started

Getting Started

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	✓ build timeout plugin	✓ Credentials Binding Plugin	** Pipeline: Shared Groovy Libraries ** Branch API Plugin ** Pipeline: Multibranch ** Authentication Tokens API Plugin ** Docker Commons Plugin ** Durable Task Plugin ** Pipeline: Nodes and Processes ** Docker Pipeline ** Pipeline: Stage Tags Metadata ** Pipeline: Declarative Agent API ** Pipeline: Model Definition Pipeline ** GitHub API Plugin Jenkins Git plugin ** GitHub plugin ** GitHub Branch Source Plugin ** Pipeline: GitHub Groovy Libraries ** - required dependency
✓ Timestamper	✓ Workspace Cleanup Plugin	✓ Ant Plugin	✓ Gradle Plugin	
✓ Pipeline	⌚ GitHub Organization Folder Plugin	✓ Pipeline: Stage View Plugin	✓ Git plugin	
⌚ Subversion Plug-in	⌚ SSH Slaves plugin	✓ Matrix Authorization Strategy Plugin	✓ PAM Authentication plugin	
✓ LDAP Plugin	⌚ Email Extension Plugin	✓ Mailer Plugin		

Jenkins 2.46.1

©2003 PnP Solution Co., Ltd.

89

Create Admin User

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.46.1

Continue as admin Save and Finish

Setup Complete

The screenshot shows the Jenkins setup complete page. At the top left, there is a "Getting Started" link. The main content area features a large heading "Jenkins is ready!" followed by the subtext "Your Jenkins setup is complete." Below this is a blue button labeled "Start using Jenkins". At the bottom left of the main content area, the Jenkins version "Jenkins 2.46.1" is displayed. The entire page has a light gray background with a dark gray header bar at the top.

Getting Started

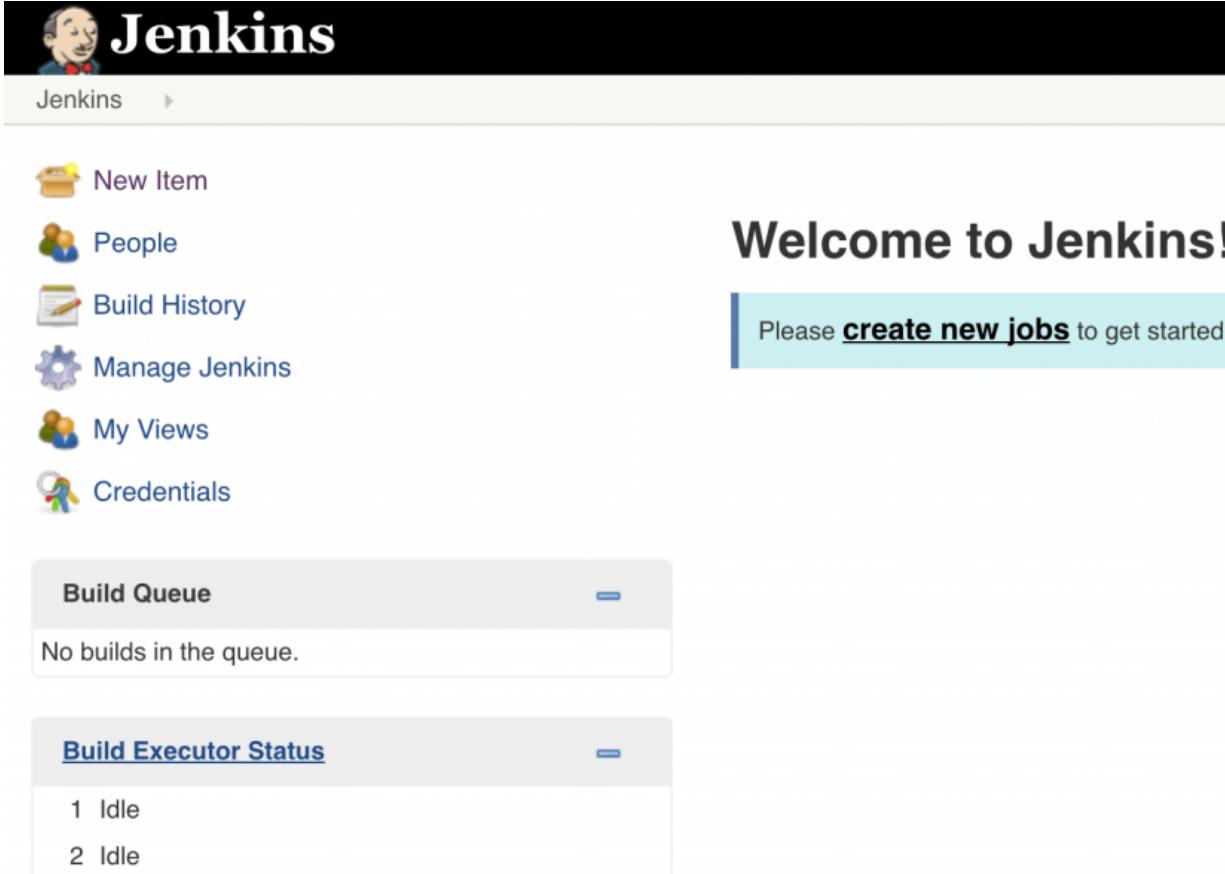
Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

Jenkins 2.46.1

Create New Jobs



The screenshot shows the Jenkins dashboard. At the top left is the Jenkins logo and the word "Jenkins". Below it is a navigation menu with links: "New Item", "People", "Build History", "Manage Jenkins", "My Views", and "Credentials". A large "Welcome to Jenkins!" message is centered, followed by a call-to-action: "Please [create new jobs](#) to get started." Below this are two collapsed sections: "Build Queue" (showing "No builds in the queue.") and "Build Executor Status" (showing "1 Idle" and "2 Idle").

New Item

People

Build History

Manage Jenkins

My Views

Credentials

Welcome to Jenkins!

Please [create new jobs](#) to get started.

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Choose Pipeline

Enter an item name

» Required field



Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.



Pipeline

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



External Job

This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.



Multi-configuration project

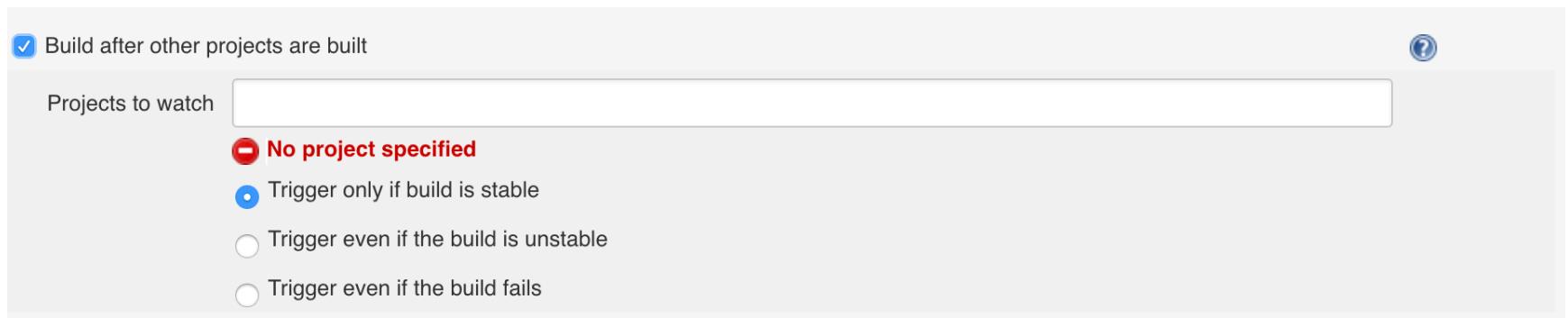
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Jenkins Config Build Trigger

Build Triggers

- Build after other projects are built 
- Build periodically 
- GitHub hook trigger for GITScm polling 
- Poll SCM 
- Disable this project 
- Quiet period 
- Trigger builds remotely (e.g., from scripts) 

Build after other projects are built



The screenshot shows a configuration panel for a build step. At the top left is a checked checkbox labeled "Build after other projects are built". To its right is a blue circular icon with a question mark. Below this is a section titled "Projects to watch" containing a text input field with the placeholder "No project specified". Underneath the input field are three radio button options: "Trigger only if build is stable" (selected), "Trigger even if the build is unstable", and "Trigger even if the build fails".

สั่งให้ทำงานอัตโนมัติ หลังจากที่ project กำหนดทำงานเสร็จ โดยมี 3 option คือ
ทำงานเมื่อ build สำเร็จ ไม่มี warning
ทำงานเมื่อ build สำเร็จ แบบมี warning
ทำงานไม่สำเร็จ

Build periodically

Build periodically ?

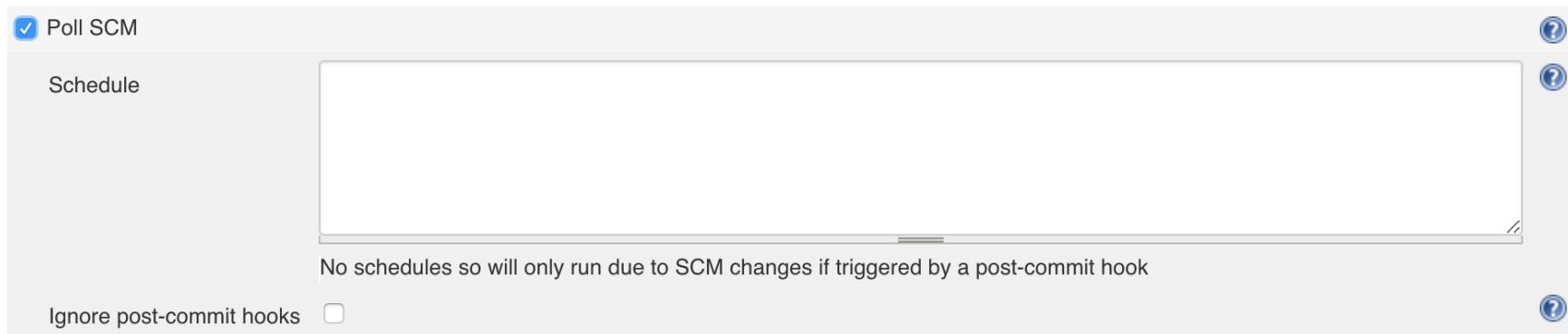
Schedule ?

Would last have run at Wednesday, June 27, 2018 9:43:05 PM ICT; would next run at Wednesday, June 27, 2018 9:55:05 PM ICT.

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:
MINUTE HOUR DOM MONTH DOW
MINUTE Minutes within the hour (0–59)
HOUR The hour of the day (0–23)
DOM The day of the month (1–31)
MONTH The month (1–12)
DOW The day of the week (0–7) where 0 and 7 are Sunday.

ສ້າງໃຫ້ກໍາງານວັດໂນມັຕິຕາມໜ່ວຍເວລາທີ່ກໍານົດ ໂດຍສາມາດກໍາບັນດາເວລາໄດ້ວູ້
ໃນຮູບແບບ crontab format

Poll SCM



สั่งให้ทำงานอัตโนมัติตามช่วงเวลาที่กำหนด โดยสามารถกำหนดเวลาได้อยู่ในรูปแบบ crontab format

โดยมีลักษณะการทำงานคือจะไปทำการตรวจสอบดูก่อนว่า source code มีการเปลี่ยนแปลงหรือไม่ ถ้าไม่มีเปลี่ยนแปลงก็จะไม่ทำการ build job

Trigger builds remotely

Build Triggers

- Build after other projects are built 
- Build periodically 
- GitHub hook trigger for GITScm polling 
- Poll SCM 
- Disable this project 
- Quiet period 
- Trigger builds remotely (e.g., from scripts) 

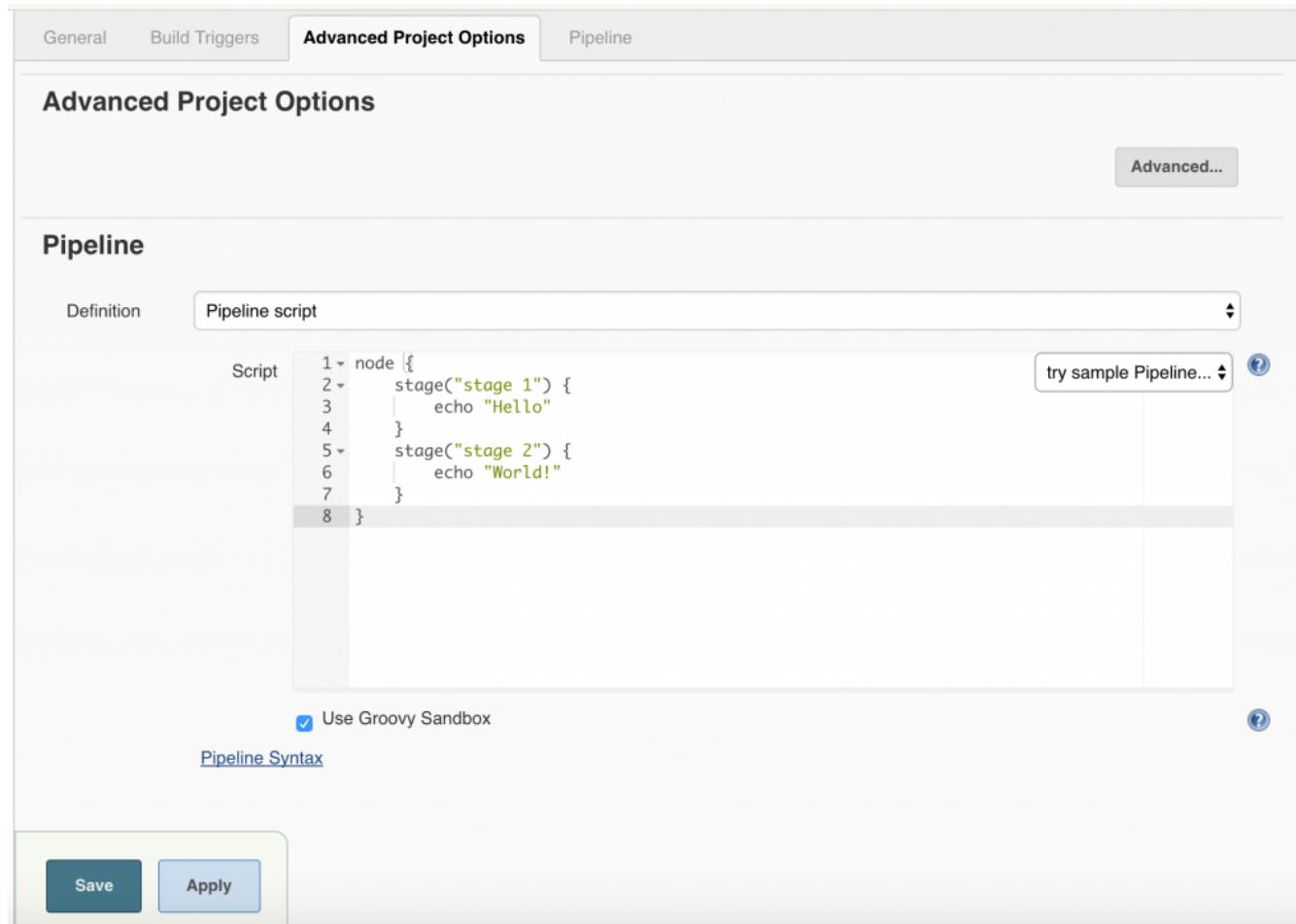
Authentication Token

Use the following URL to trigger build remotely: `JENKINS_URL/job/first/build?token=TOKEN_NAME` or `/buildWithParameters?token=TOKEN_NAME`
Optionally append `&cause=Cause+Text` to provide text that will be included in the recorded build cause.

เป็นการตั้งค่าให้สามารถสั่งให้ job ทำงานโดยการเรียกผ่าน url เช่น ใส่ค่า Authentication Token เป็น **123** Jenkins run ที่เครื่อง 127.0.0.1 ก็จะเรียกผ่าน url ดังนี้

`http://127.0.0.1:8080/job/{job_name}/build?token=123`

Pipeline script



The screenshot shows the 'Advanced Project Options' configuration page for a Jenkins project. The 'Pipeline' tab is selected. In the 'Definition' section, the 'Pipeline script' is set to 'Pipeline script'. The script content is:

```
1 ~ node //{
2 ~   stage("stage 1") {
3 ~     echo "Hello"
4 ~   }
5 ~   stage("stage 2") {
6 ~     echo "World!"
7 ~   }
8 }
```

Below the script, there is a checkbox labeled 'Use Groovy Sandbox' which is checked. A link 'Pipeline Syntax' is also present. At the bottom, there are 'Save' and 'Apply' buttons.

Pipeline template

```
pipeline {
    agent any
    environment {
        APP_NAME = "test app name"
    }
    stages {
        stage('Build Image'){
            steps {
                sh "echo ${env.APP_NAME}"
            }
        }
    }
}
```

Jenkins Job List

Jenkins [ENABLE AUTO REFRESH](#)

New Item 

People 

Build History 

Project Relationship 

Check File Fingerprint 

Manage Jenkins 

My Views 

Open Blue Ocean 

Credentials 

New View 

All Checker PnP Training Zone9 wcom wisdom + 

[add description](#)

S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav
		api.pnpsw.com	1 mo 0 days - #22	4 mo 25 days - #13	23 sec	 
		api.wisdomairways.com	5 days 10 hr - #25	8 days 11 hr - #20	2 min 8 sec	 
		api.zone9sport.com	4 mo 14 days - #11	5 mo 4 days - #8	45 sec	 
		birt.pnpsw.com	2 mo 17 days - #8	N/A	57 sec	 
		booking.wisdomairways.com	5 days 10 hr - #5	N/A	1 min 46 sec	 
		checker.pnpsw.com	5 mo 10 days - #9	5 mo 10 days - #8	50 sec	 

กดเครื่องหมาย + เพื่อกำการเพิ่ม tab เพื่อการกรองข้อมูลของ job

Set List View

View name

List View 
Shows the simple list format. You can choose which jobs are to be displayed in which view.

My View
This view automatically displays all the jobs that the current user has an access to.

Name

Description

[Plain text] [Preview](#)

Filter build queue

Filter build executors

Job Filters

Status Filter

Recurse in subfolders

Jobs

- api.pnpsw.com
- api.wisdomairways.com
- api.zone9sport.com
- birt.pnpsw.com
- booking.wisdomairways.com
- checker.pnpsw.com
- checkerapi.pnpsw.com
- demo.pnpsw.com
- issue.pnpsw.com
- issueapi.pnpsw.com
- train-api.pnpsw.com
- www.pnpsw.com

Logs

Jenkins Kan Ouivirach | log out

ENABLE AUTO REFRESH

Back to Dashboard Status Changes Build Now Delete Pipeline Configure Full Stage View Pipeline Syntax

Recent Changes add description

Pipeline FizzBuzz

Stage View

Average stage times:

stage 1	stage 2
132ms	145ms

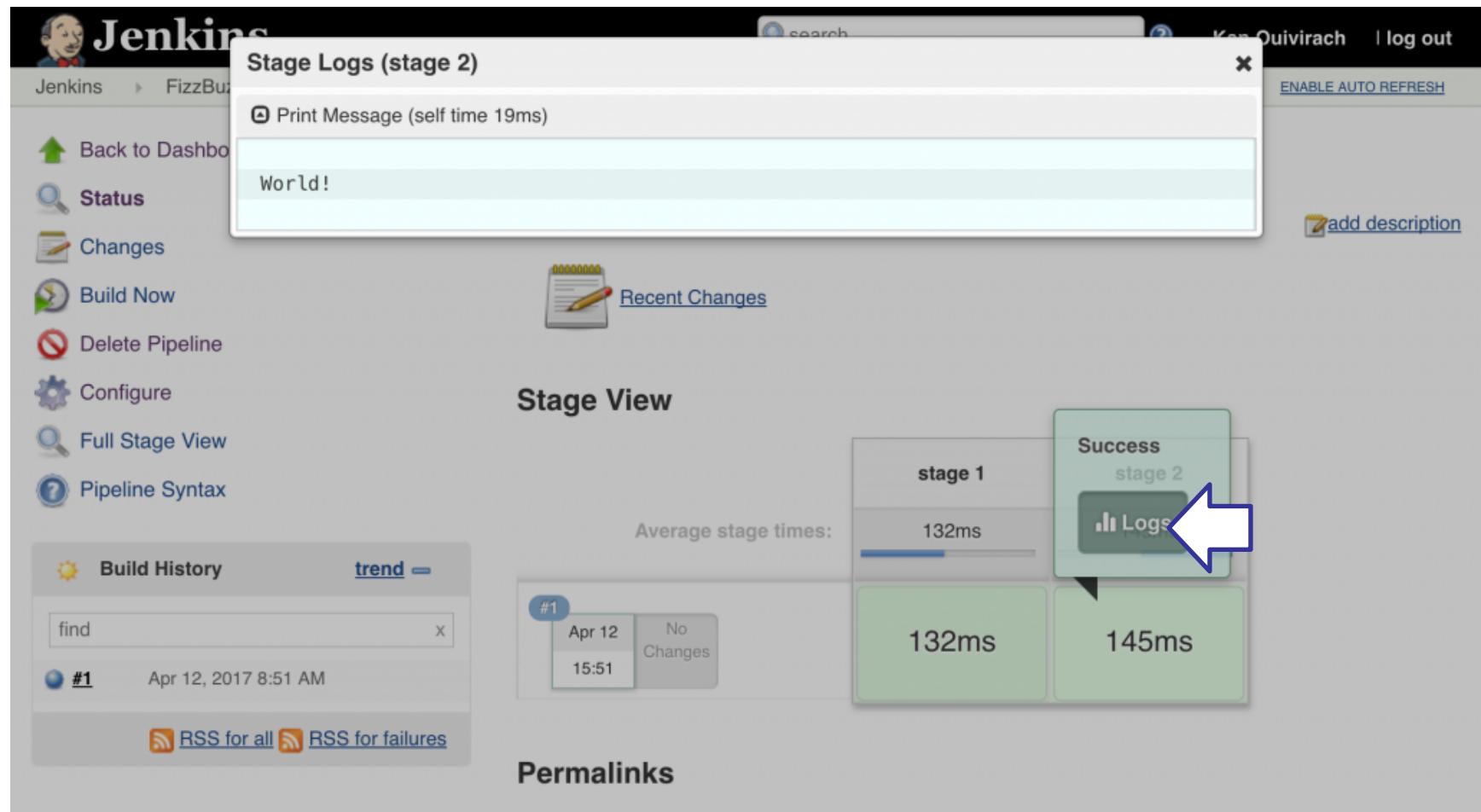
#1 Apr 12, 2017 8:51 AM 15:51 No Changes 132ms 145ms

RSS for all RSS for failures

Permalinks



Logs



The screenshot shows the Jenkins interface for a pipeline named "FizzBuzz".

Stage Logs (stage 2)

- Print Message (self time 19ms)
World!

Recent Changes

Stage View

Average stage times:

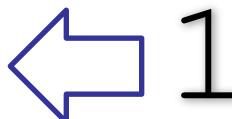
stage 1	stage 2
132ms	Success Logs
132ms	145ms

Permalinks

A blue arrow points to the "Logs" button in the Stage View section of the pipeline summary.

Add Credentials #1

-  New Item
-  People
-  Build History
-  Project Relationship
-  Check File Fingerprint
-  Manage Jenkins
-  My Views
-  Open Blue Ocean
-  **Credentials**
-  System
-  New View

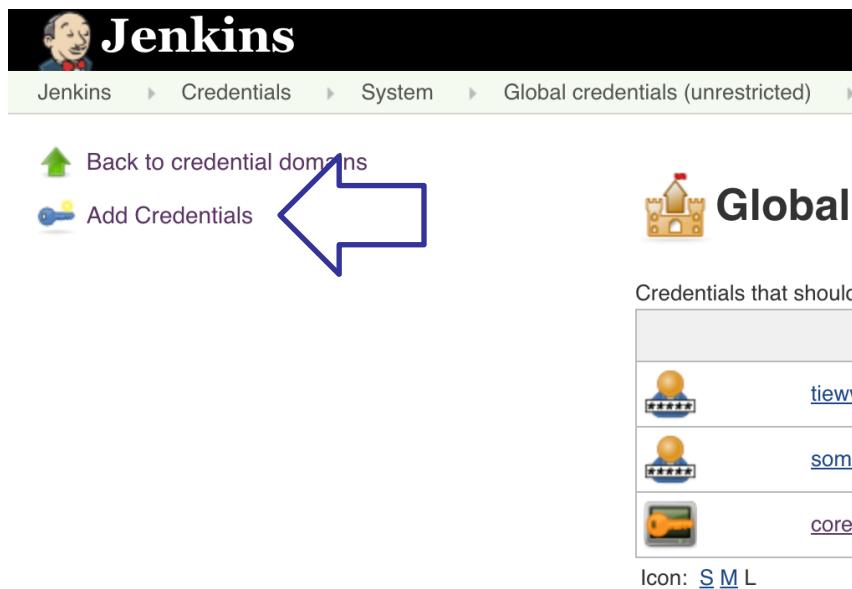


Stores scoped to Jenkins

P	Store ↓	Domains
	Jenkins	
		 (global)

1 2

Add Credentials #2



The screenshot shows the Jenkins Global credentials configuration page. The navigation bar at the top includes links for Jenkins, Credentials, System, and Global credentials (unrestricted). On the left, there are links for Back to credential domains and Add Credentials, with a large blue arrow pointing from the 'Add Credentials' link towards the central form. The main area features a 'Global' icon and a table titled 'Credentials that should'. The table lists three entries: 'tiewy' (with a user icon), 'somi' (with a user icon), and 'core' (with a key icon). Below the table is a link 'Icon: S M L'. The URL in the browser's address bar is `/jenkins/crumbIssuer/api/json`.

Kind Username with password

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

Username

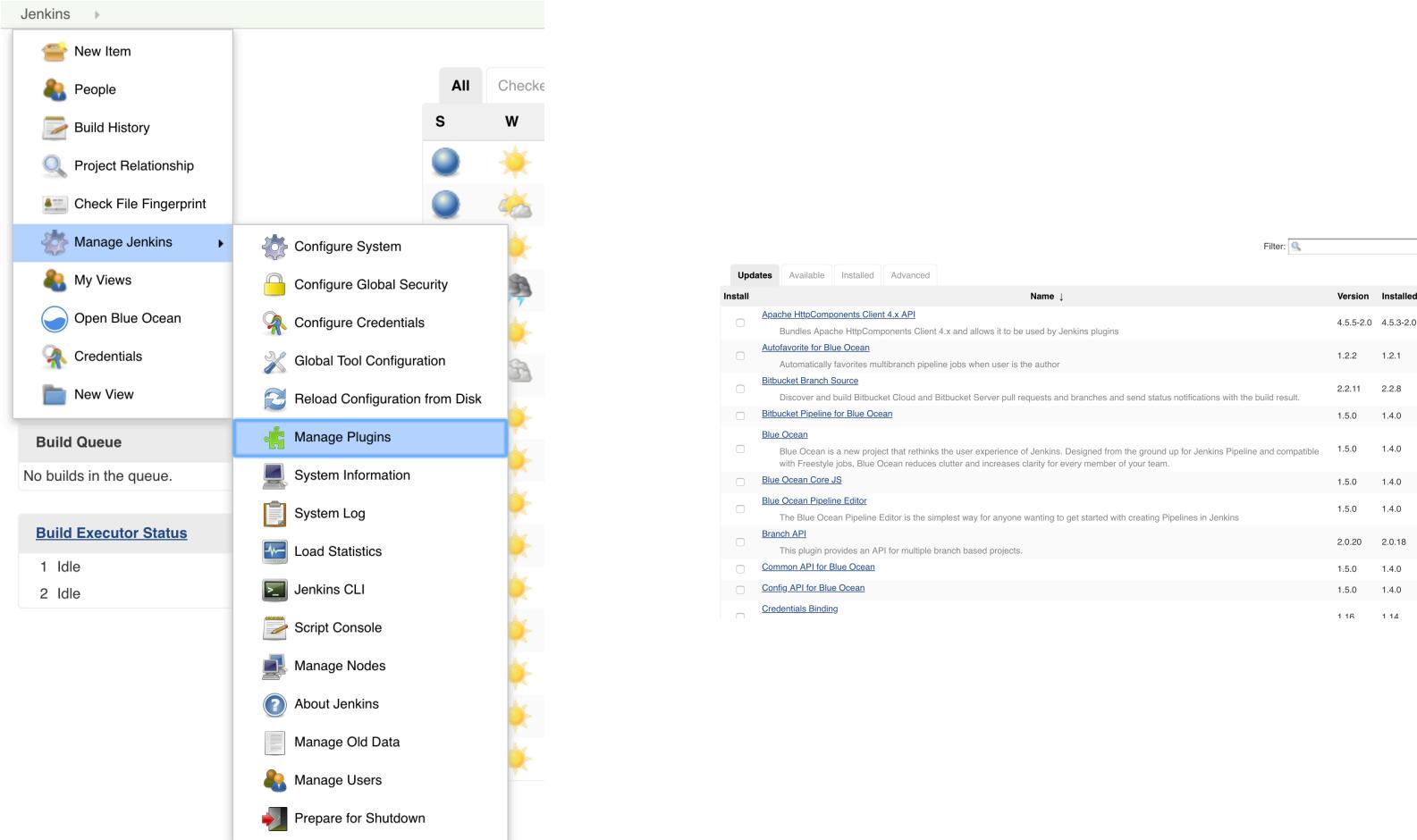
Password

ID

Description

OK

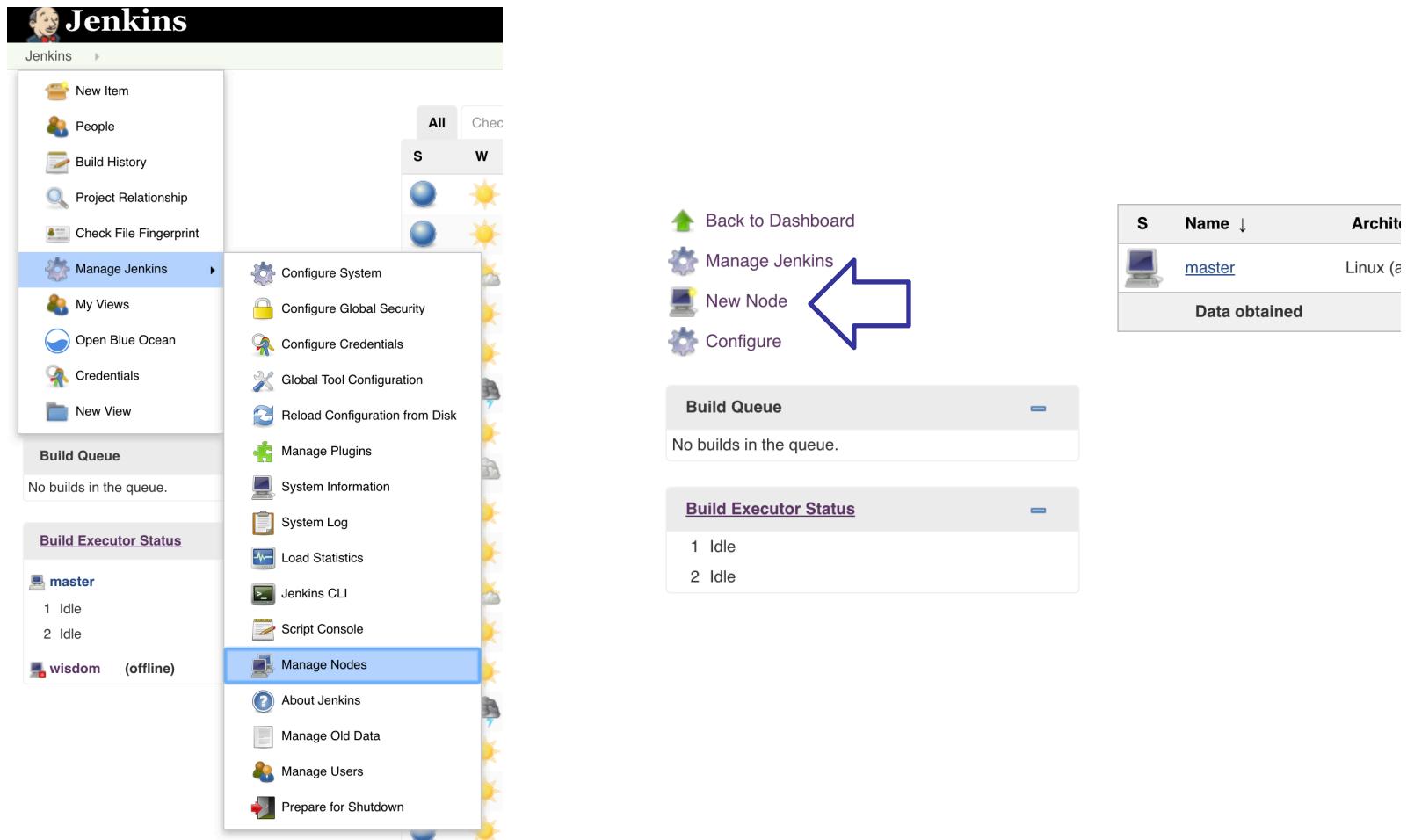
Install Plugins



The screenshot shows the Jenkins dashboard with the 'Manage Jenkins' menu open. Under the 'Manage Jenkins' dropdown, 'Manage Plugins' is selected, highlighted with a blue background. The right side of the screen displays the Jenkins plugin manager interface, specifically the 'Updates' tab. The table lists various Jenkins plugins with their names, descriptions, versions, and installation status.

Name	Version	Installed
Apache HttpClient 4.x API	4.5.5-2.0	4.5.3-2.0
Autofavorite for Blue Ocean	1.2.2	1.2.1
Bitbucket Branch Source	2.2.11	2.2.8
Bitbucket Pipeline for Blue Ocean	1.5.0	1.4.0
Blue Ocean	1.5.0	1.4.0
Blue Ocean Core JS	1.5.0	1.4.0
Blue Ocean Pipeline Editor	1.5.0	1.4.0
Branch API	2.0.20	2.0.18
Common API for Blue Ocean	1.5.0	1.4.0
Config API for Blue Ocean	1.5.0	1.4.0
Credentials Binding		

Add new node



The screenshot shows the Jenkins dashboard with the following interface elements:

- Left Sidebar:** Contains links like New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins (selected), My Views, Open Blue Ocean, Credentials, and New View.
- Central Dashboard:** Shows build status indicators (All, S, W) and a list of nodes:
 - master**: 1 Idle, 2 Idle.
 - wisdom**: (offline)
- Manage Jenkins Submenu:** Shows options like Configure System, Configure Global Security, Configure Credentials, Global Tool Configuration, Reload Configuration from Disk, Manage Plugins, System Information, System Log, Load Statistics, Jenkins CLI, Script Console, and Manage Nodes (selected).
- Manage Jenkins Main Area:** Shows options like Back to Dashboard, Manage Jenkins (selected), New Node (highlighted with a blue arrow), and Configure.
- Build Queue:** No builds in the queue.
- Build Executor Status:** 1 Idle, 2 Idle.
- Table on the right:** Shows a list of nodes with columns S, Name, and Architecture.

S	Name ↓	Architecture
	master	Linux (a)
Data obtained		

Add new node

Node name

Permanent Agent
Adds a plain, permanent agent to Jenkins. This is called "permanent" provisioning. Select this type if no other agent types apply — for example Jenkins, etc.

Name	<input type="text" value="mynode"/>
Description	<input type="text"/>
# of executors	<input type="text" value="1"/>
Remote root directory	<input type="text" value="/home/user"/>
Labels	<input type="text"/>
Usage	<input type="text" value="Use this node as much as possible"/>
Launch method	<input type="text" value="Launch slave agents via SSH"/>
Host	<input type="text"/>
Credentials	<input type="text" value="root"/> <input type="button" value="Add"/>
Host Key Verification Strategy <input type="text" value="Known hosts file Verification Strategy"/>	

Availability

Node Properties

Environment variables
 Tool Locations

Add new node

- Remote root directory : ໃສ່ເຊື້ອ path ຂອງເຄີ່ອງປລາຍກາງ
- Labels : ຜົວຈະຄູກເຮັດໃຫ້ໃນ pipeline
- Usage : Only Build job with label expression matching this node
- Launch Method : Launch slave agents via SSH
 - Host : ເຊື້ອເຄີ່ອງຫົວໜ້ວ ip
 - Credentials : ຄໍາ Credentials ທີ່ຕັ້ງໄວ້
 - Host Key Verification Strategy : None verifying Verification Strategy

Use node in pipeline

Declarative pipeline

```
pipeline {  
    agent {label 'slave'}  
    stages {  
        ...  
    }  
}
```

Scripted pipeline

```
node (label: 'slave') {  
    ...  
}
```

Kubernetes

A SHORT INTRODUCTION



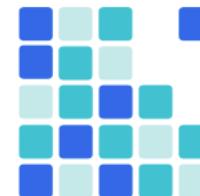
Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon [15 years of experience of running production workloads at Google](#), combined with best-of-breed ideas and practices from the community.



Planet Scale

Designed on the same principles that allows Google to run billions of containers a week, Kubernetes can scale without increasing your ops team.



Never Outgrow

Whether testing locally or running a global enterprise, Kubernetes flexibility grows with you to deliver your applications consistently and easily no matter how complex your need is.



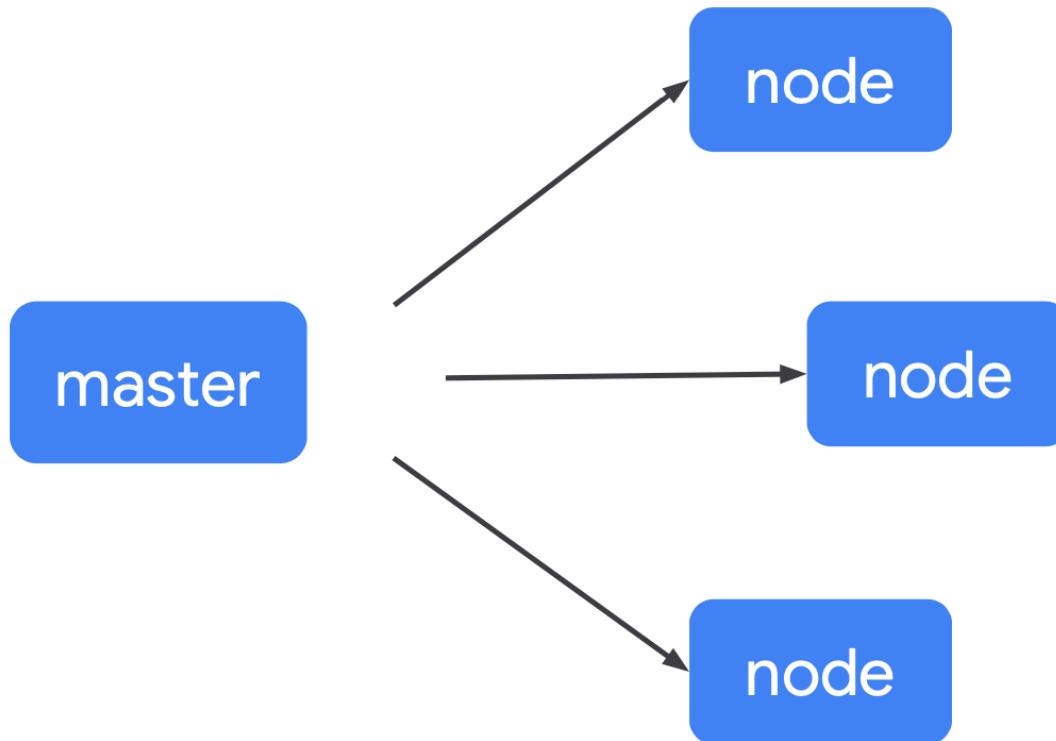
Run K8s Anywhere

Kubernetes is open source giving you the freedom to take advantage of on-premises, hybrid, or public cloud infrastructure, letting you effortlessly move workloads to where it matters to you.

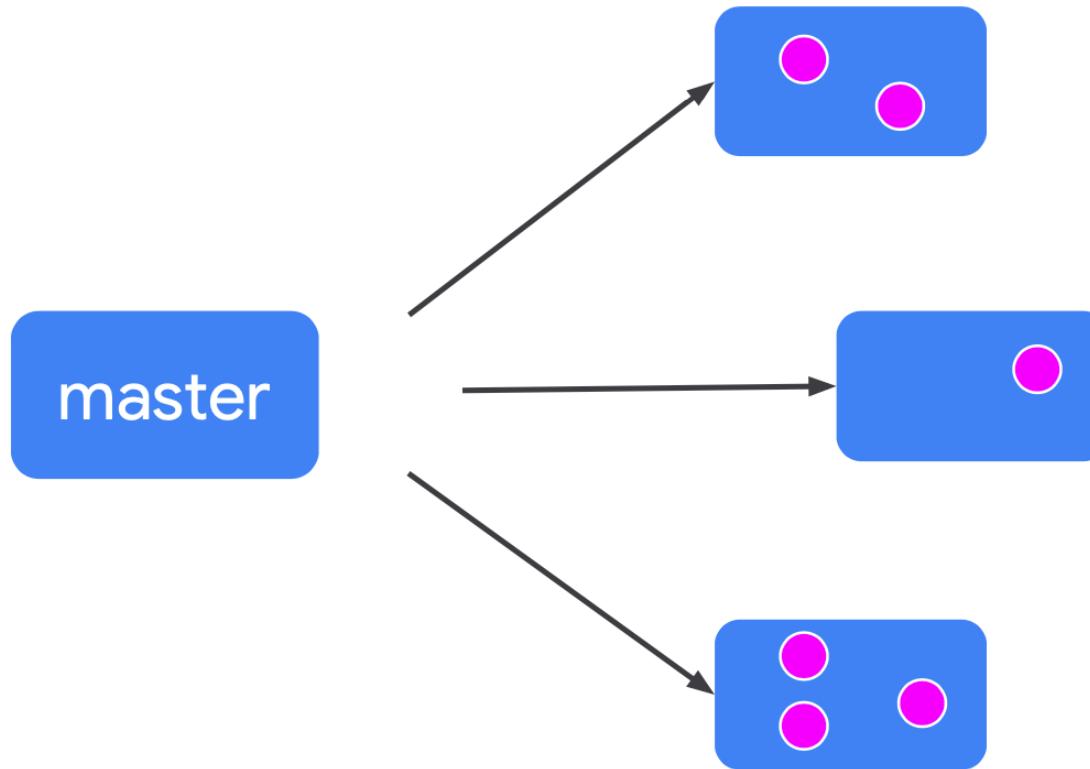
Kubernetes runs
Applications in a
Cluster.

Applications = Pods

Cluster



Pods in a Cluster



Deployment

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

Service

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

Ingress

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: http-ingress
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: helloworld
                port:
                  number: 8080
```

Kubernetes with

MICROK8S

Install MicroK8s

<https://microk8s.io/#install-microk8s>

Install MicroK8s



- 1 Download the installer for Windows

[Download MicroK8s for Windows](#)

- 2 Run the Installer

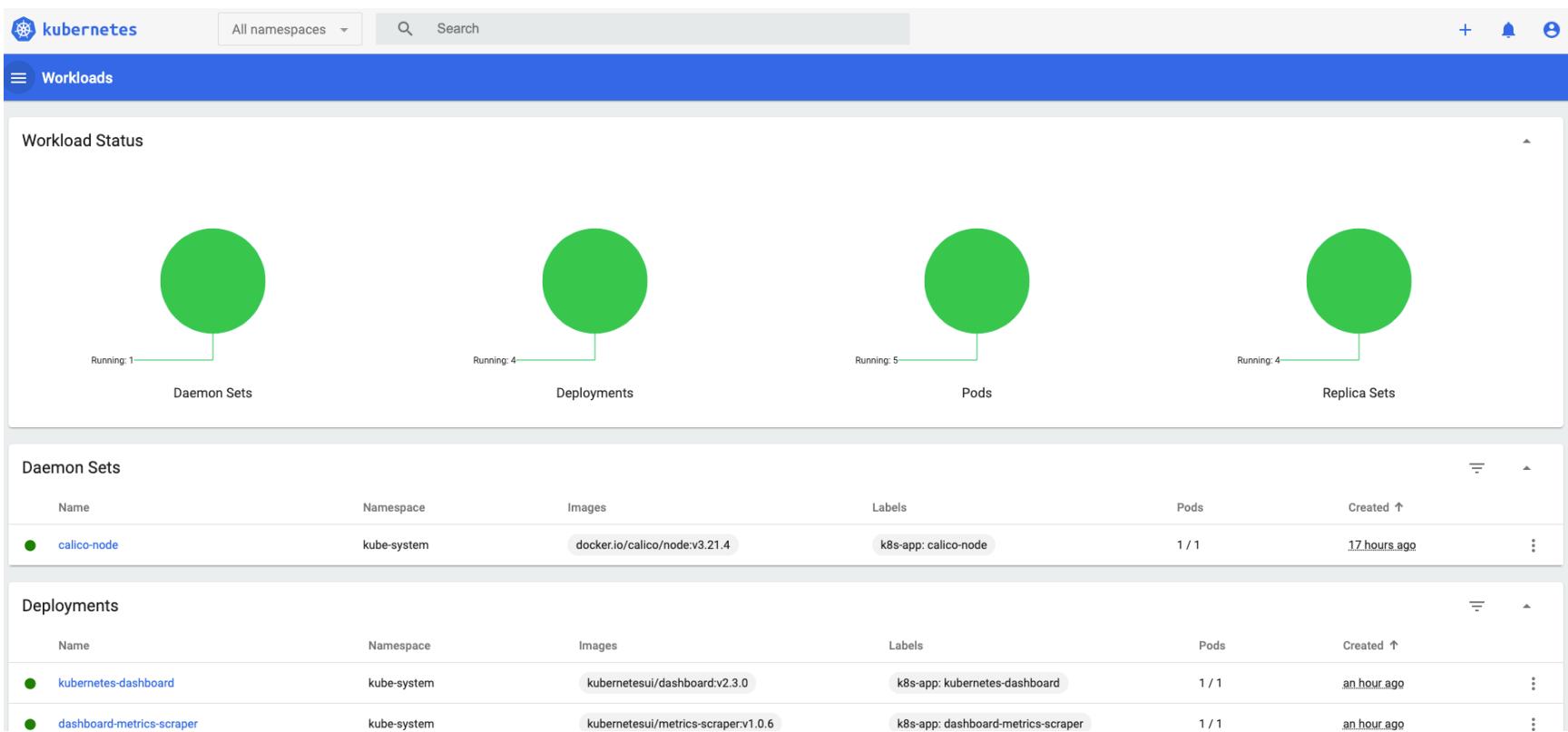


MicroK8s Command

- microk8s status --wait-ready
- microk8s enable dashboard ingress
- microk8s dashboard-proxy
- microk8s start
- microk8s stop
- microk8s add-node
- microk8s leave
- microk8s remove-node
- microk8s kubectl get all --all-namespaces

Problem Solve (Insecure SSL)

- chrome://flags/#allow-insecure-localhost



The screenshot shows the Kubernetes dashboard interface. At the top, there's a navigation bar with a 'kubernetes' logo, dropdown menus for 'All namespaces' and 'Search', and icons for '+' (Create), a bell (Notifications), and a user profile.

The main header is 'Workloads'. Below it, the 'Workload Status' section displays four large green circles representing different workload types:

- Daemon Sets: Running: 1
- Deployments: Running: 4
- Pods: Running: 5
- Replica Sets: Running: 4

Below the status summary, there are three tabs: 'Daemon Sets', 'Deployments', and 'Pods'. The 'Daemon Sets' tab is currently active, showing a single entry:

Name	Namespace	Images	Labels	Pods	Created
calico-node	kube-system	docker.io/calico/node:v3.21.4	k8s-app: calico-node	1 / 1	17 hours ago

The 'Deployments' tab shows two entries:

Name	Namespace	Images	Labels	Pods	Created
kubernetes-dashboard	kube-system	kubernetesui/dashboard:v2.3.0	k8s-app: kubernetes-dashboard	1 / 1	an hour ago
dashboard-metrics-scraper	kube-system	kubernetesui/metrics-scraper:v1.0.6	k8s-app: dashboard-metrics-scraper	1 / 1	an hour ago

Manage Kubernetes with

KUBECTL

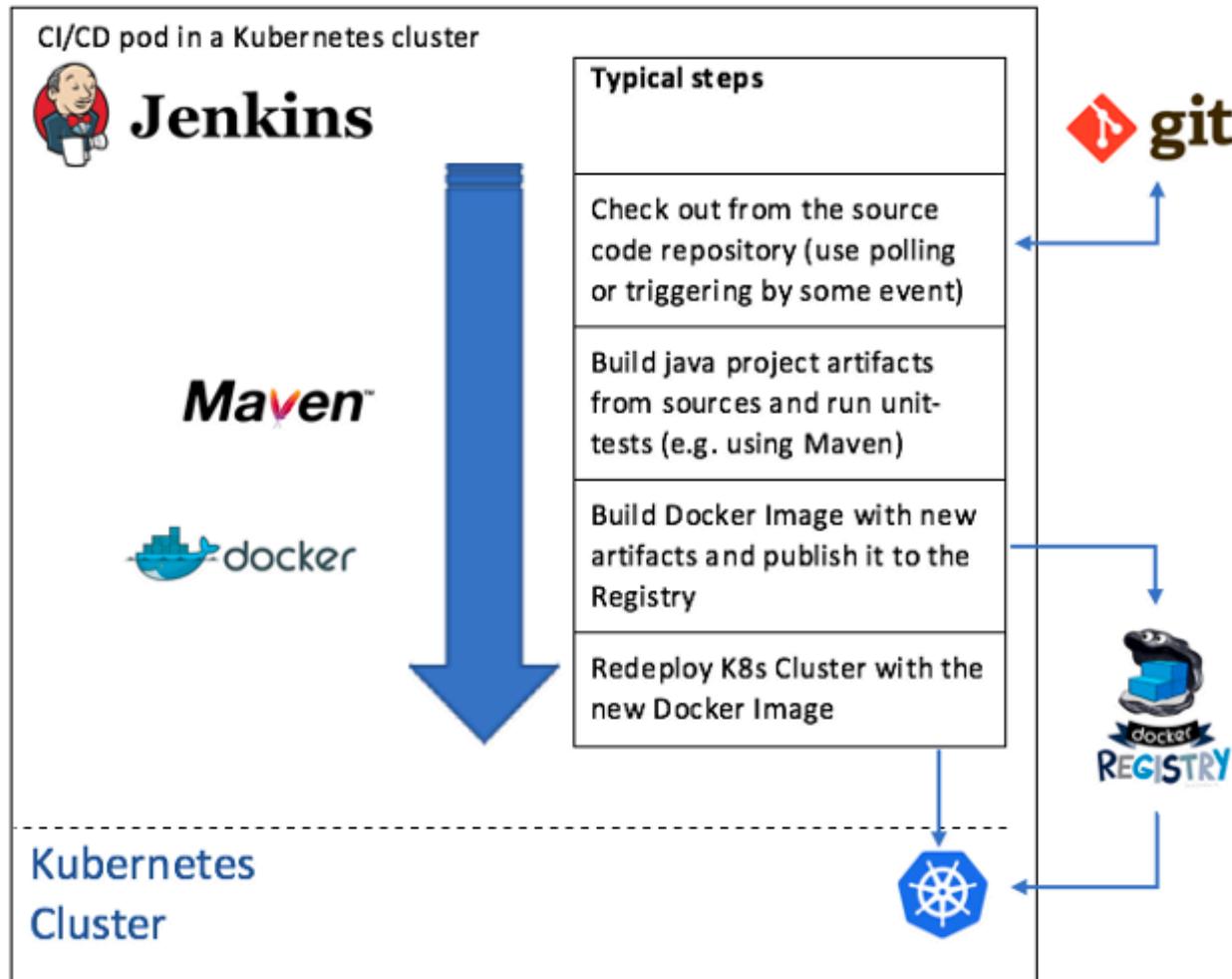
kubectl command list

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

Jenkins with

K8S

Jenkins with k8s



Any questions?

