Research & Development Team

PnP Solution Co., Ltd.

DevOps Gitlab Jenkins and K8s

www.pnpsw.com

sommai.k@gmail.com

081-754-4663

Lineid : sommaik

Medium: @sommaikrangpanich

สมหมาย กรังพานิช

ตำแหน่งปัจจุบัน

- กรรมการสมาคมอุตสาหกรรมซอฟต์แวร์ไทย ATSI
- กรรมการผู้จัดการบริษัท พี เอ็น พี โซลูชั่น จำกัด

ด้านความเชี่ยวชาญ

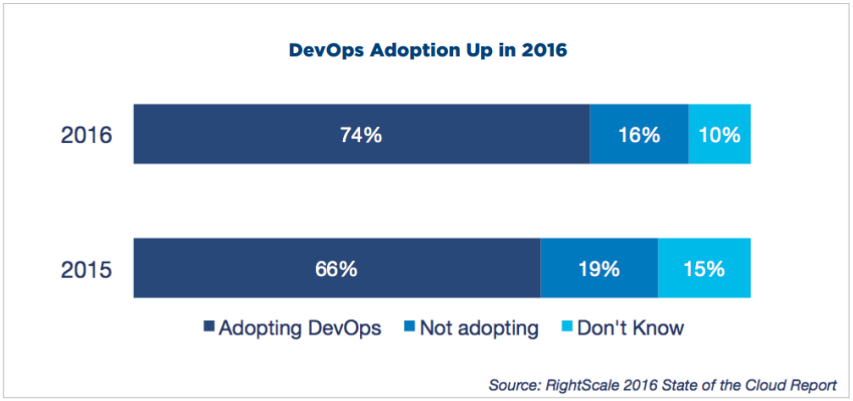
- ผู้เชี่ยวชาญด้านการออกแบบและพัฒนา Software ด้วย Framework ดังนี้
  - Java, Springboot, Go, Node.js
  - Full Stack, Angular, Vue, React, Svelte
  - Apollo GraphQL
  - Docker, Docker Swarm, K8s
  - Flutter, Dart
  - MongoDB, Oracle, MySQL, SQL Server, DB2, PostgreSQL
- ผู้เชี่ยวชาญด้านการออกแบบและประยุกต์ใช้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

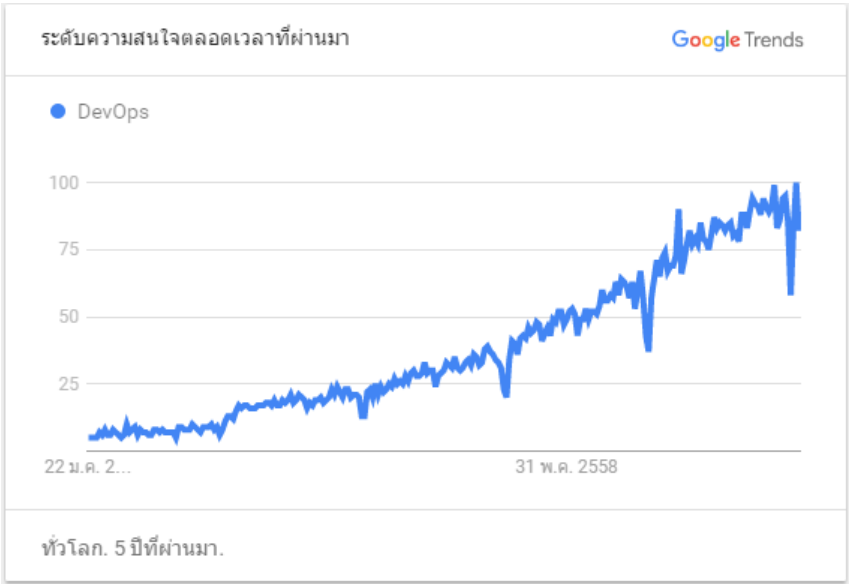
Definitions and History

- 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 เป็นเรื่องทั่วไปที่ทำกันในชีวิตประจำวัน การนำเสนอเรื่องนี้เป็นแรงบันดาลใจให้ Patrick Debois จัดงาน DevOpsDay ขึ้นมาในปีเดียวกัน คำว่า DevOps ที่ย่อมาจาก Development และ Operations จึงถูกสร้างขึ้นตั้งแต่นั้นเป็นต้นมา

- ปัจจุบัน DevOps เป็นแนวคิดที่มีประสิทธิภาพและแพร่หลายออกไปทั่วโลก จากผลสำรวจองค์กรกว่า 1,000 แห่งจากรายงานRightScale 2016 State of the Cloud Report: DevOps Trendsพบว่าในปี 2016 มีองค์กรนำ DevOps ไปปรับใช้แล้วถึง 74% ซึ่งเพิ่มขึ้นจากปีที่แล้วถึง 8% และจำนวนการ Search คำว่า DevOps ใน google ก็ยังเพิ่มขึ้นเรื่อยๆ ด้วย

ระดับความสนใจ



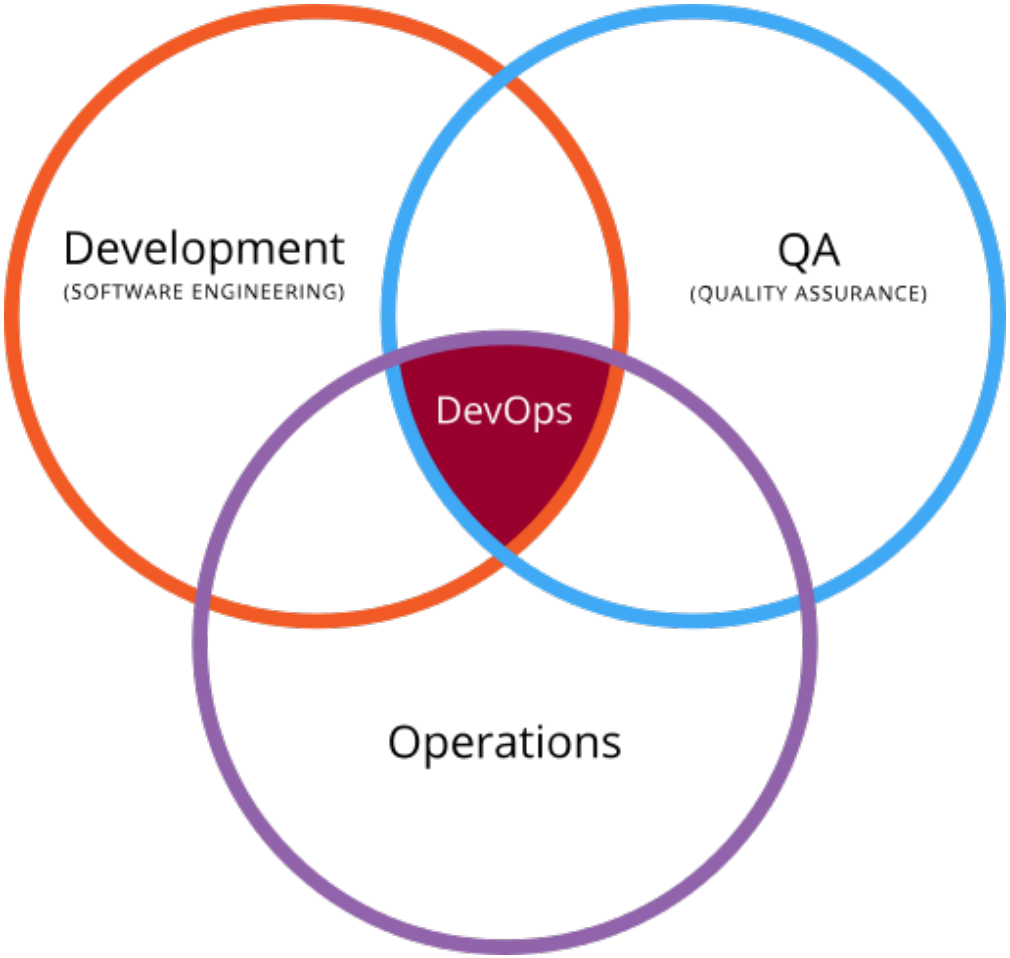


Cultural change

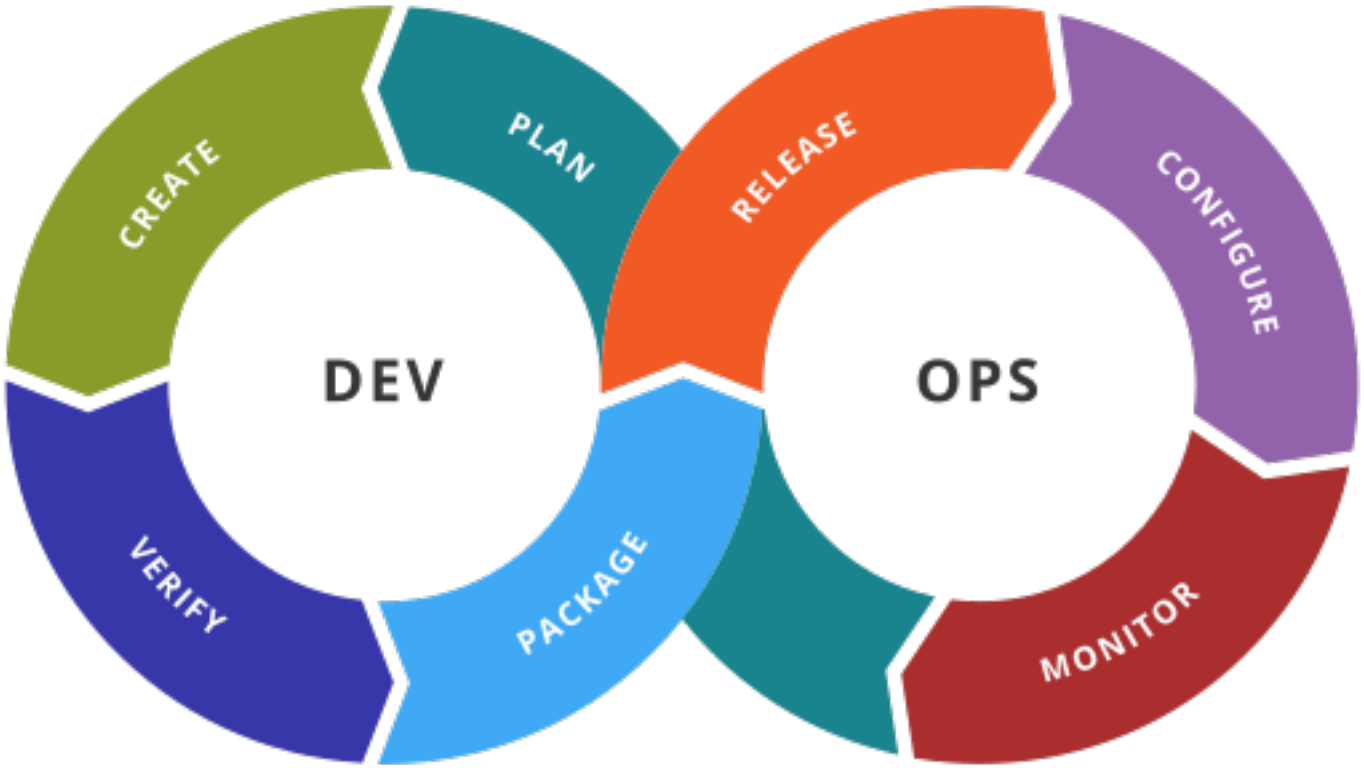
DevOps is more than just a tool or a process change.

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

Overview



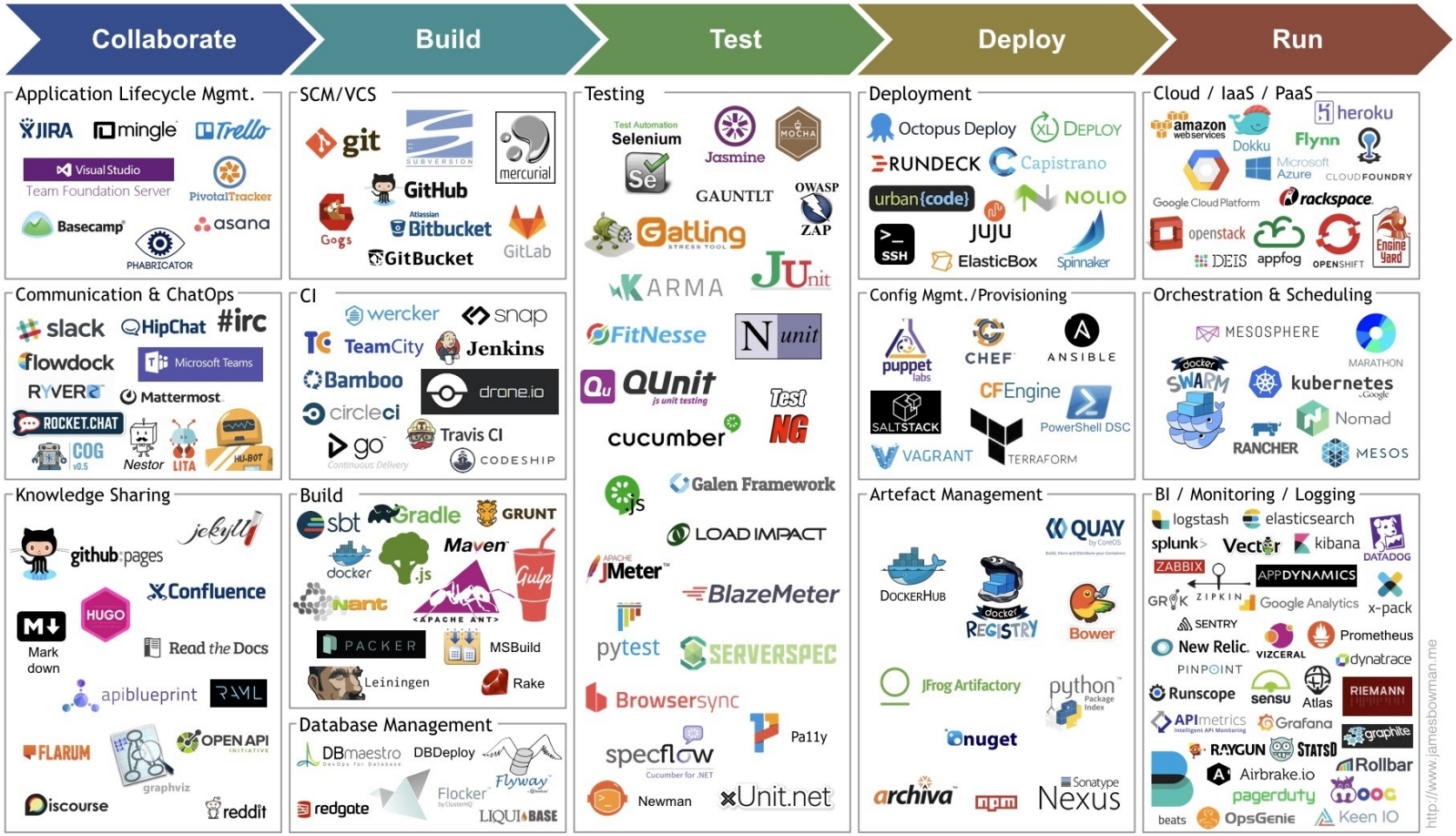
Workflow



7 Ways to Get Started with DevOps

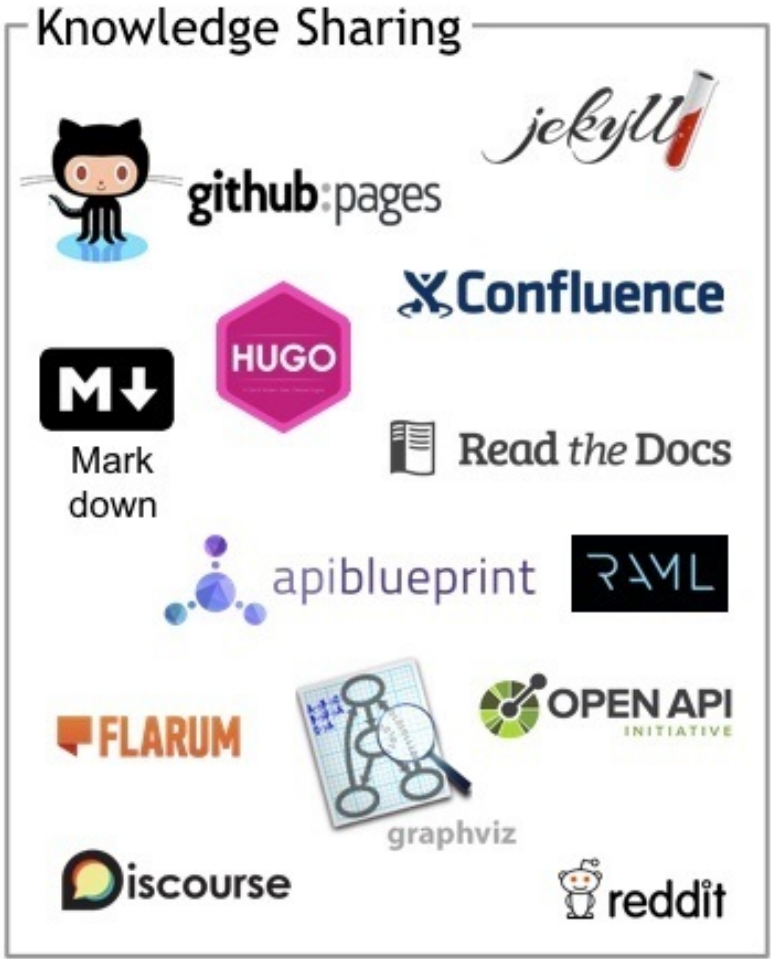
- Invite Your Operations Team Into Your Development Process
- Visualize the Work Together
- Automate Your Test/Build Process
- Create a Deployment Plan
- Identify Fragile Systems
- Smooth Out Wait States
- Link Your Work to Your Value

DevOps Ecosystem

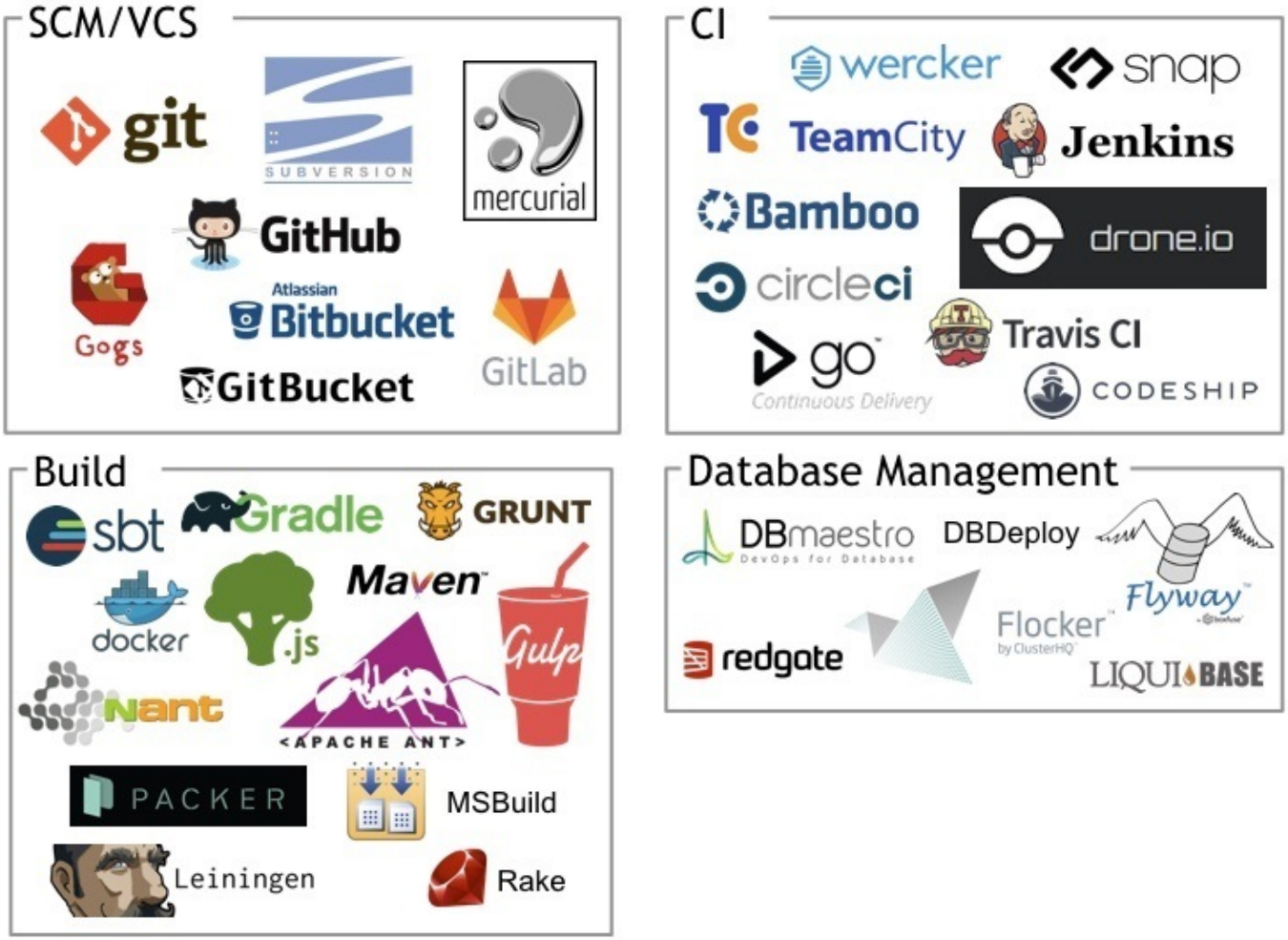


Collaborate

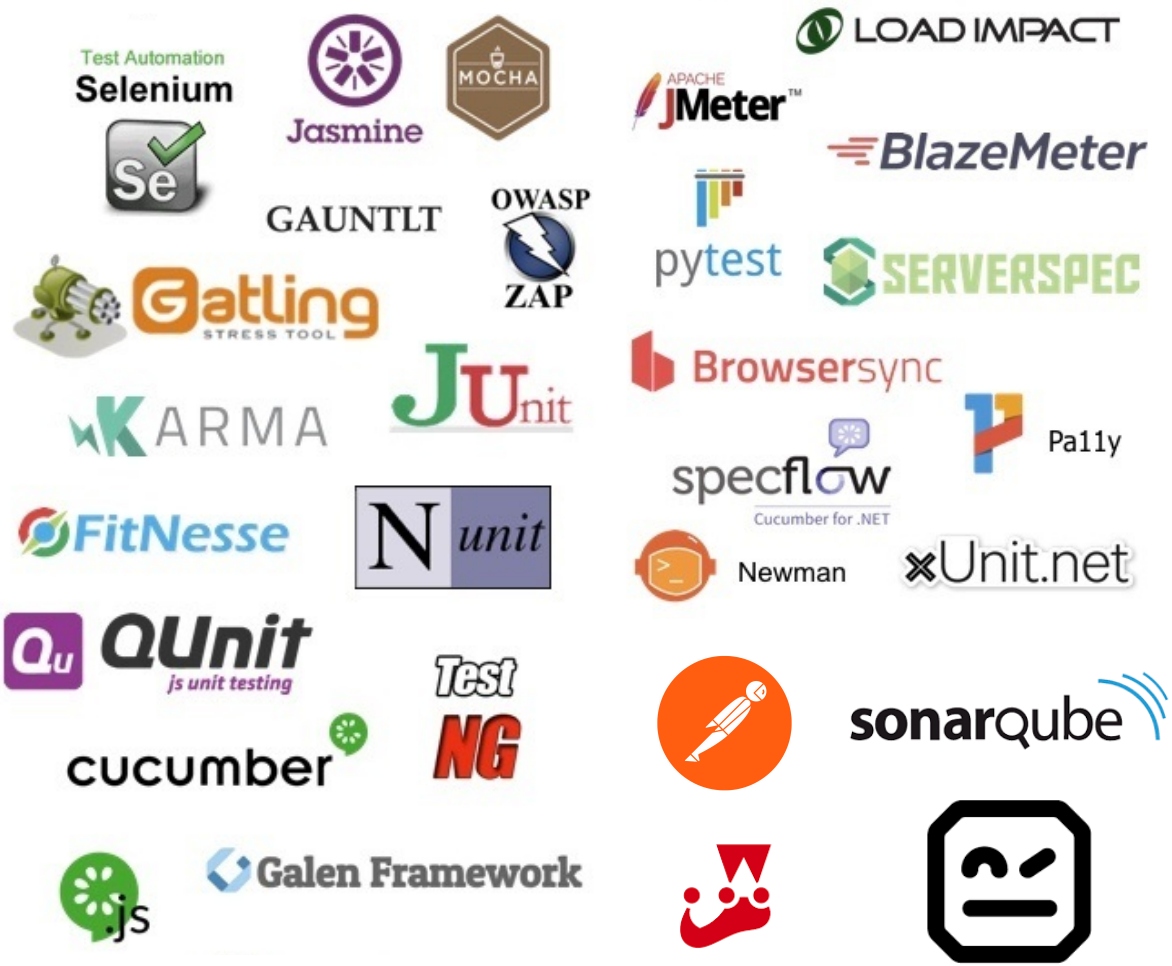
Collaborate



Build


SCM/VCS
git
SUBVERSION
mercurial
GitHub
Gogs
Atlassian
Bitbucket
GitLab
GitBucket
CI
wercker
snap
TeamCity
Jenkins
Bamboo
drone.io
circleci
Travis CI
go
Continuous Delivery
CODESHIP
Build
sbt
Gradle
GRUNT
Maven
docker
.js
Gulp
Nant
<APACHE ANT>
PACKER
MSBuild
Leiningen
Rake
Database Management
DBmaestro
DevOps for Database
DBDeploy
Flyway
redgate
Flocker
by ClusterHQ
LIQUIBASE

Test


Test Automation
Selenium
Jasmine
MOCHA
LOAD IMPACT
APACHE
JMeter™
BlazeMeter
GAUNTLT
OWASP
ZAP
pytest
SERVERSPEC
Gatling
STRESS TOOL
Browsersync
JUnit
KARMA
Pa11y
specflow
Cucumber for .NET
FitNesse
N unit
Newman
xUnit.net
QUnit
js unit testing
Test
NG
sonarqube
cucumber
.js
Galen Framework

Deploy


Deployment
Octopus Deploy
XL DEPLOY
RUNDECK
Capistrano
urban{code}
NOLIO
JUJU
>_ SSH
ElasticBox
Spinnaker
Config Mgmt. / Provisioning
puppet labs
CHEF
ANSIBLE
CFEngine
SALTSTACK
PowerShell DSC
VAGRANT
TERRAFORM


Artefact Management
QUAY by CoreOS
Build, Store and Distribute your Containers
DOCKERHUB
docker REGISTRY
Bower
JFrog Artifactory
python Package Index
nuget
archiva
npm
Sonatype Nexus

Run


Cloud / IaaS / PaaS
amazon web services
Dokku
heroku
Flynn
Microsoft Azure
CLOUDFOUNDRY
Google Cloud Platform
rackspace
openstack
DEIS
appfog
OPENSHIFT
Engine Yard
Orchestration & Scheduling
MESOSPHERE
MARATHON
docker SWARM
kubernetes
Google
Nomad
RANCHER
MESOS
BI / Monitoring / Logging
logstash
elasticsearch
splunk>
Vector
kibana
DATADOG
ZABBIX
APPDYNAMICS
GROK
ZIPKIN
Google Analytics
x-pack
SENTRY
New Relic.
VIZCERAL
Prometheus
dynatrace
PINPOINT
Runscope
sensu
Atlas
RIEMANN
APImetrics
Intelligent API Monitoring
Grafana
graphite
RAYGUN
STATSD
Rollbar
Airbrake.io
pagerduty
Mooc
beats
OpsGenie
Keen IO

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

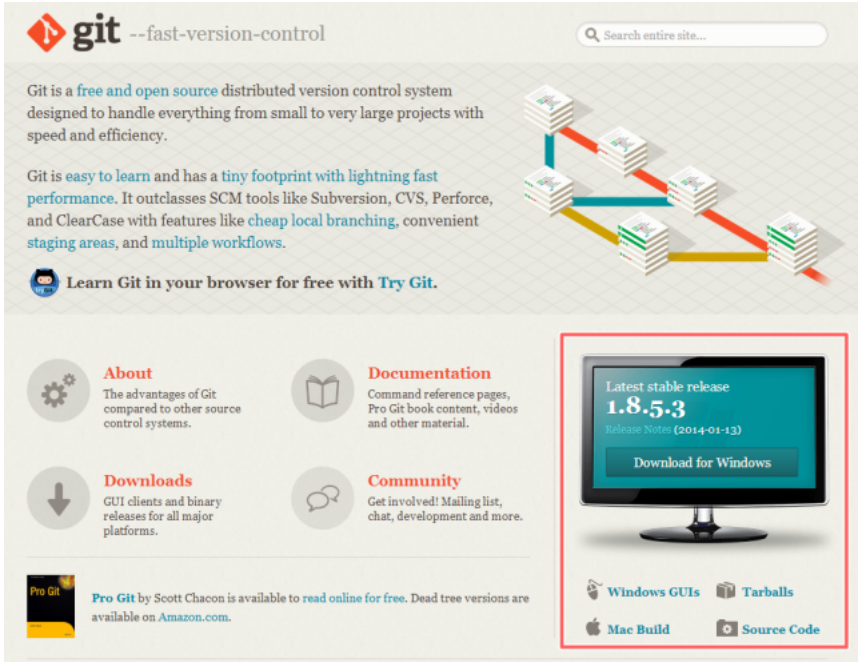
INSTALLATION

การติดตั้ง

GIT FOR WINDOWS

การติดตั้ง GIT สำหรับ Windows

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

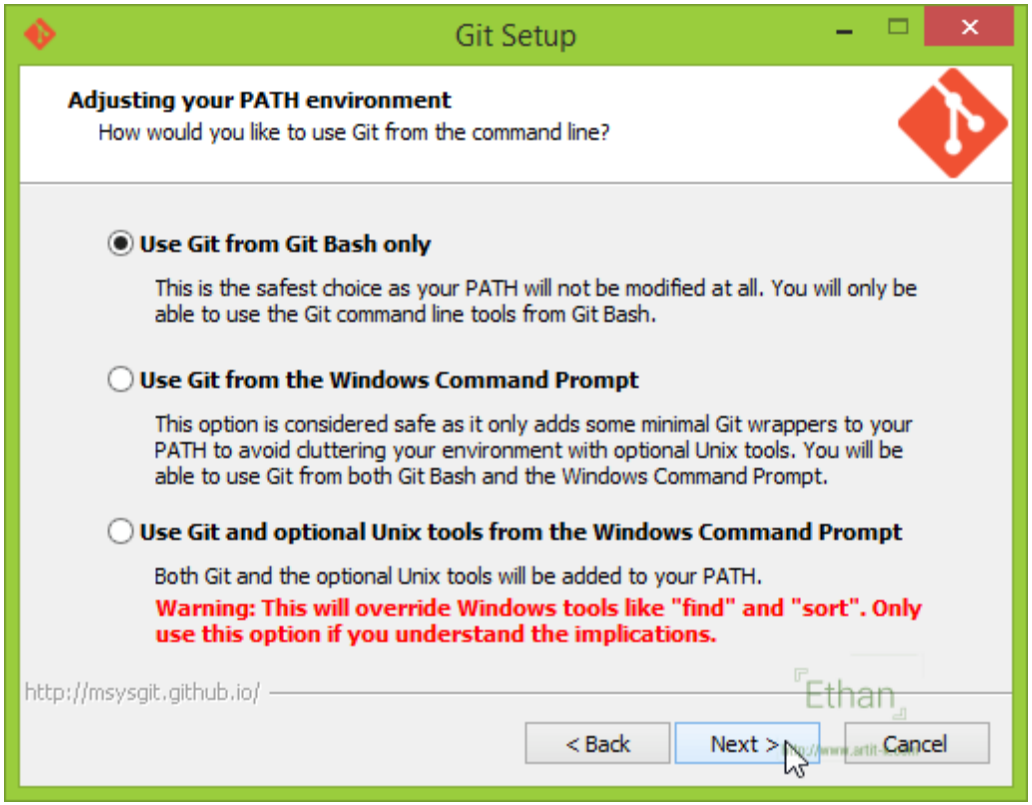


การติดตั้ง GIT สำหรับ Windows #2

- ติดตั้งโดยใช้สิทธิ Administrator ในการติดตั้ง

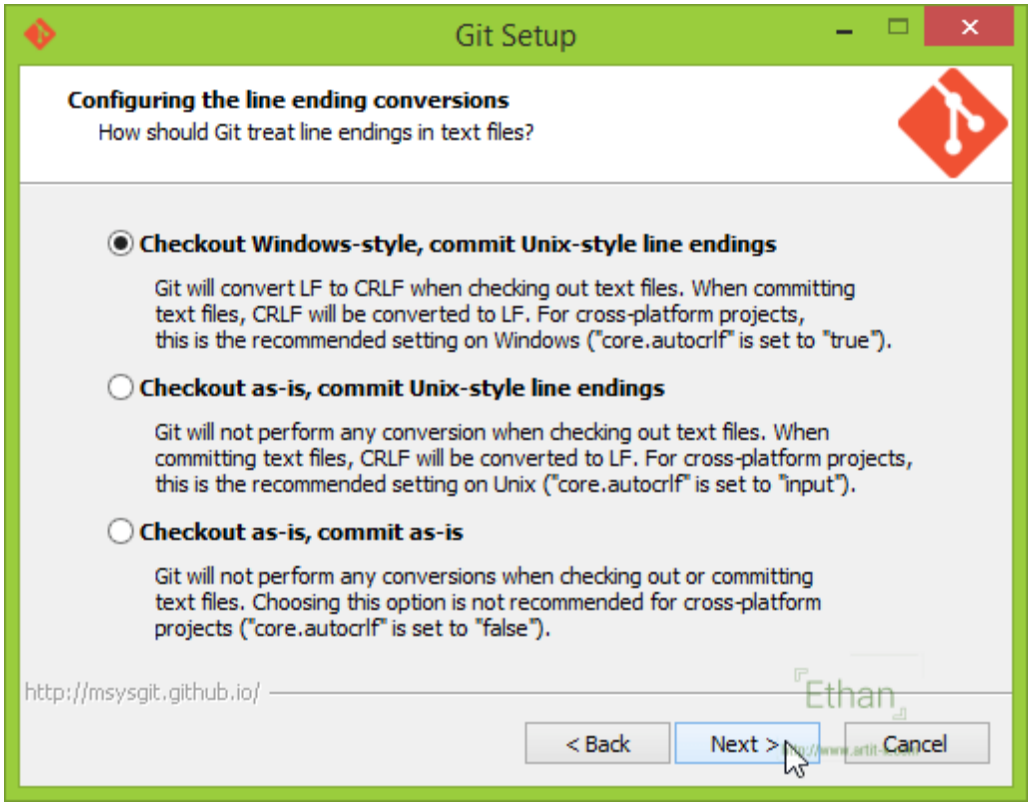
การติดตั้ง GIT สำหรับ Windows #3

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



การติดตั้ง GIT สำหรับ Windows #4

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

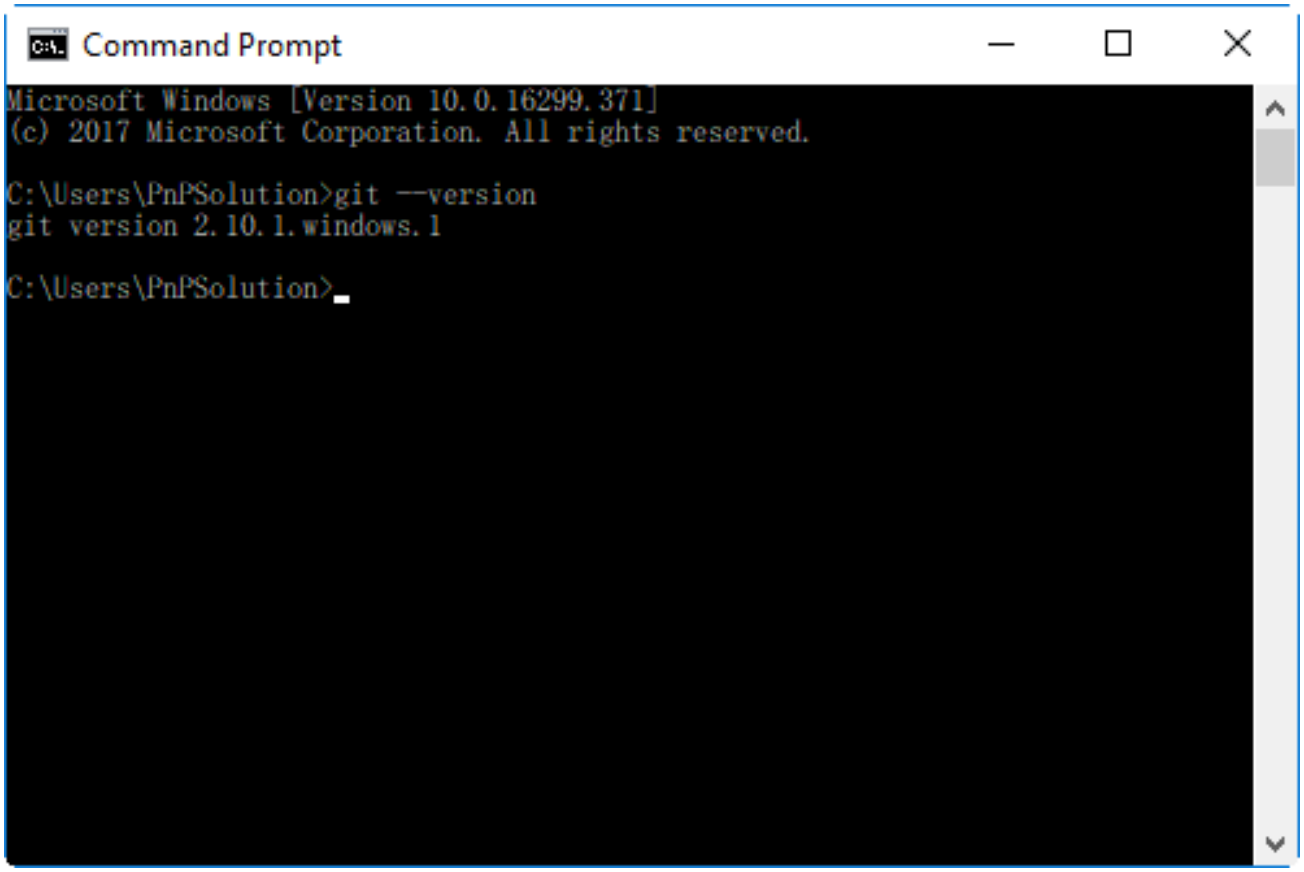


การติดตั้ง GIT สำหรับ Windows #5

- กดปุ่ม next ไปจนถึงหน้าสุดท้าย

ทดสอบหลังการติดตั้ง Git

- เปิดโปรแกรม cmd แล้วพิมพ์คำสั่ง git --version

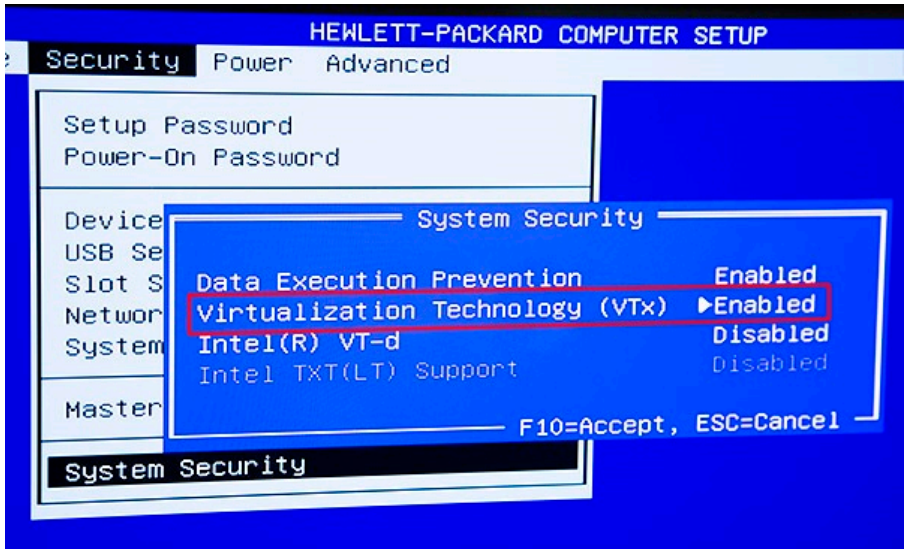


การติดตั้ง

DOCKER DESKTOP

การติดตั้ง Docker สำหรับ Windows

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



การติดตั้ง Docker สำหรับ Windows #2

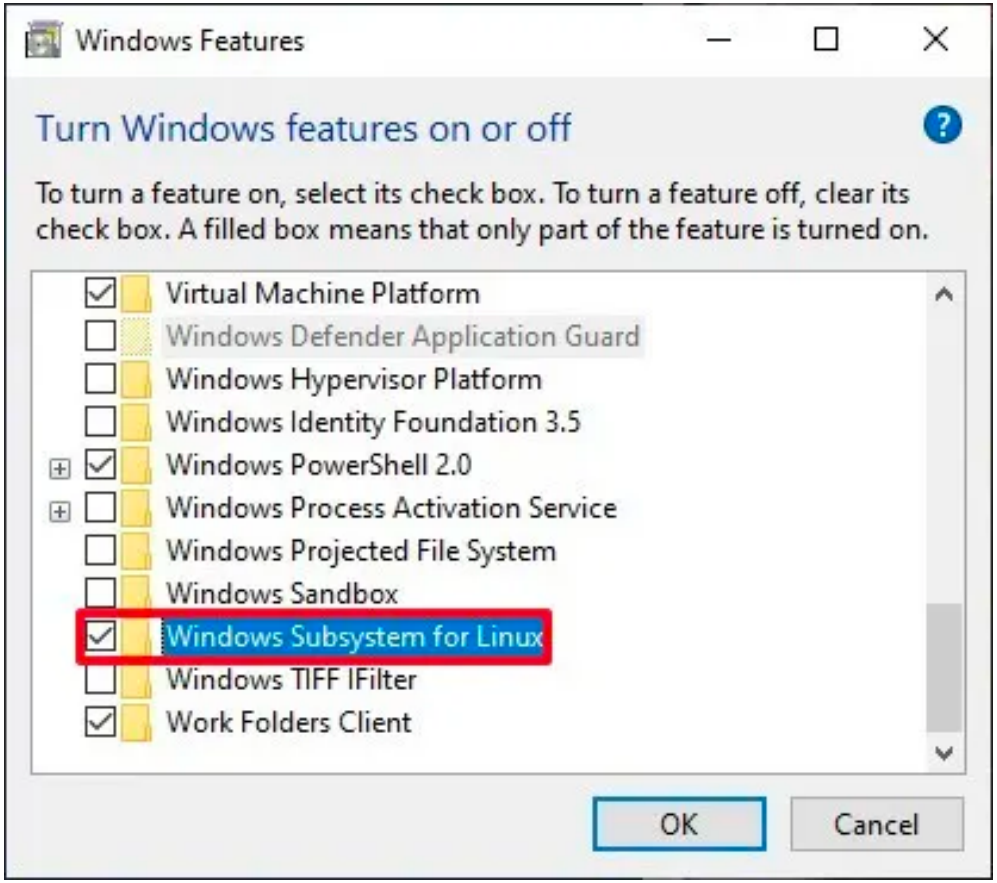
- เข้า link https://www.docker.com/products/docker-desktop/
- กด next ไปเรื่อยๆ จนสิ้นสุดการติดตั้ง

การติดตั้ง Docker สำหรับ Windows #2

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

การติดตั้ง Docker สำหรับ Windows #3

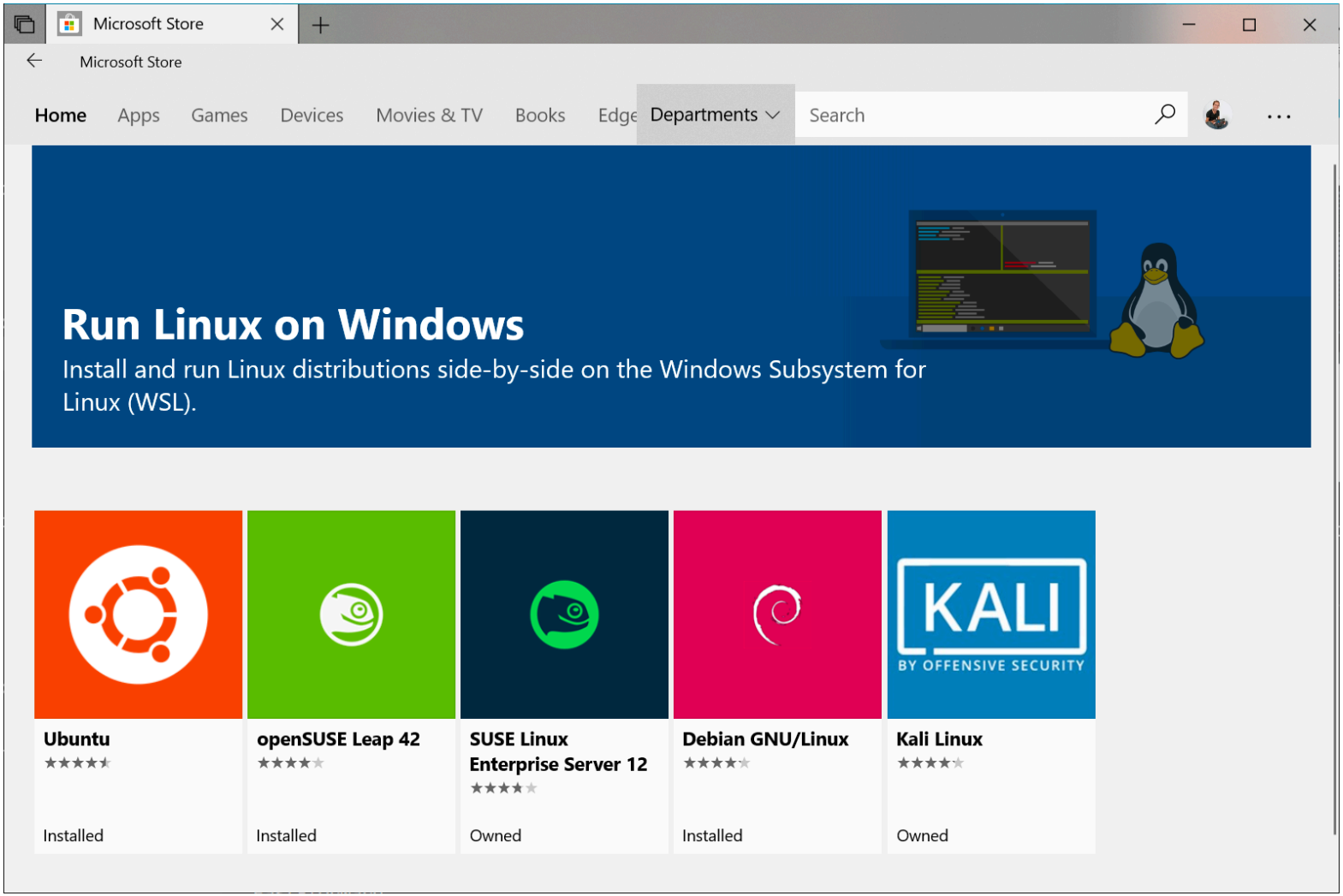
- เปิดใช้ WSL2



การติดตั้ง Docker สำหรับ Windows #4

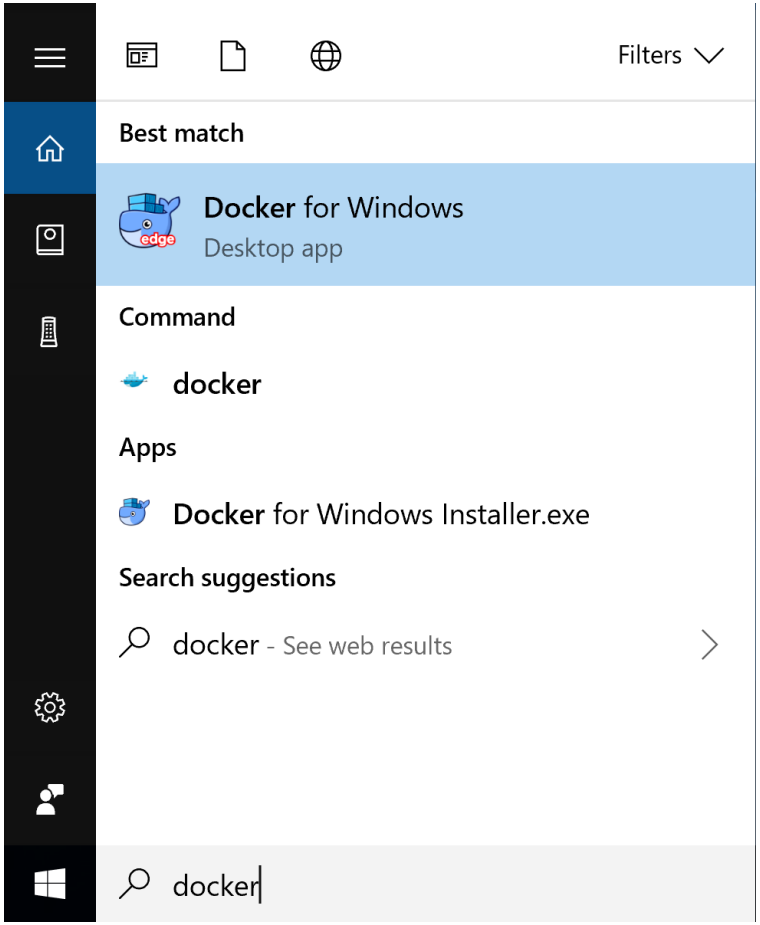
- run คำสั่งดังต่อไปนี้
  - dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
  - dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
- download and install
  - https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi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

ติดตั้ง Linux distributition



ทดสอบหลังการติดตั้ง #1

- หลังจากติดตั้งเสร็จแล้วให้ทำการเปิด docker ดังนี้



ทดสอบหลังการติดตั้ง #2

- จะมี docker run อยู่ที่ taskbar ดังภาพ

- ทดสอบการติดตั้งโดยการเปิด cmd แล้วพิมพคำสั่งดังนี้
- docker ps

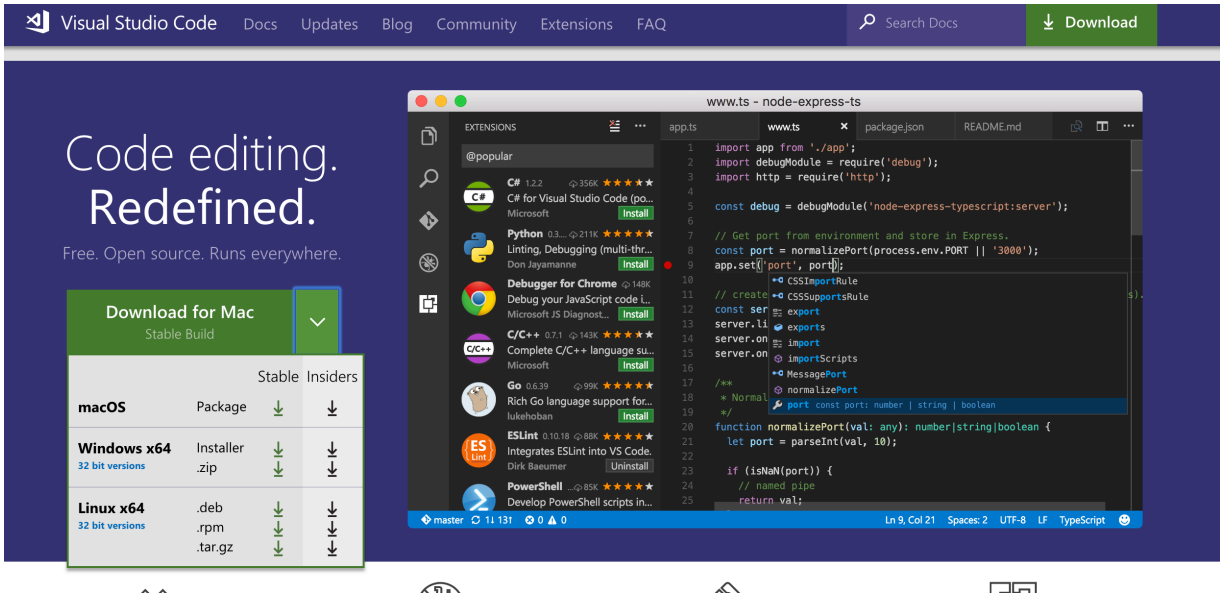
```
Sommais-MacBook-Pro:~ sommaik$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

การติดตั้ง

VISUAL STUDIO CODE

การติดตั้ง VSC

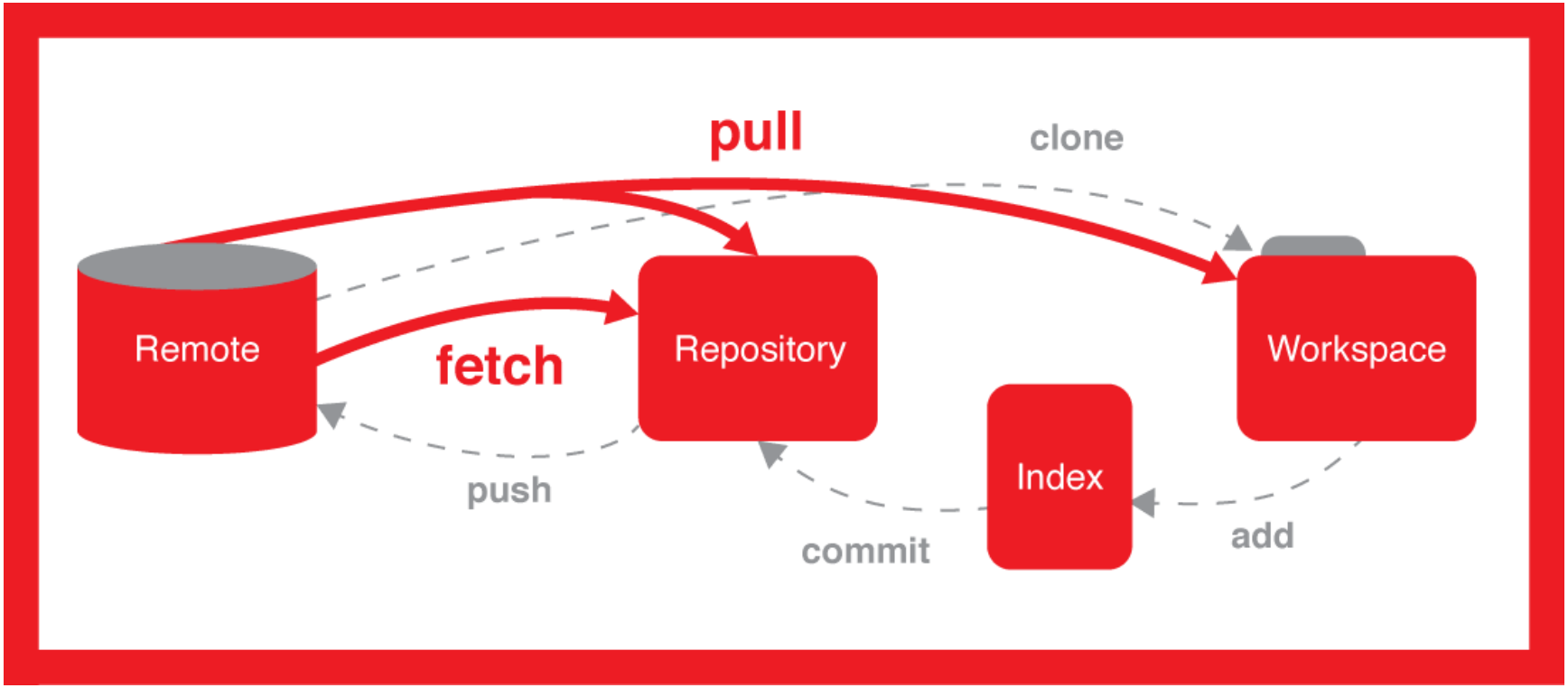
- เข้าไปที่ website https://code.visualstudio.com/
- เลือก download สำหรับ windows (stable)



Software version control with

GIT

Overview

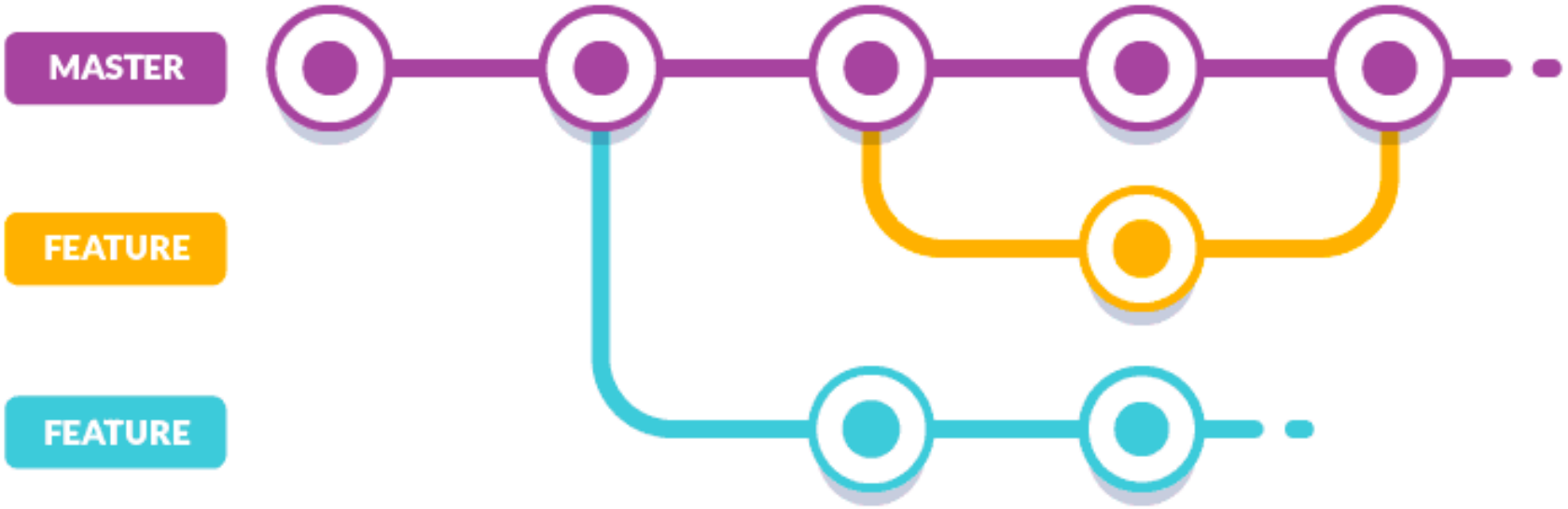


Git workflow type

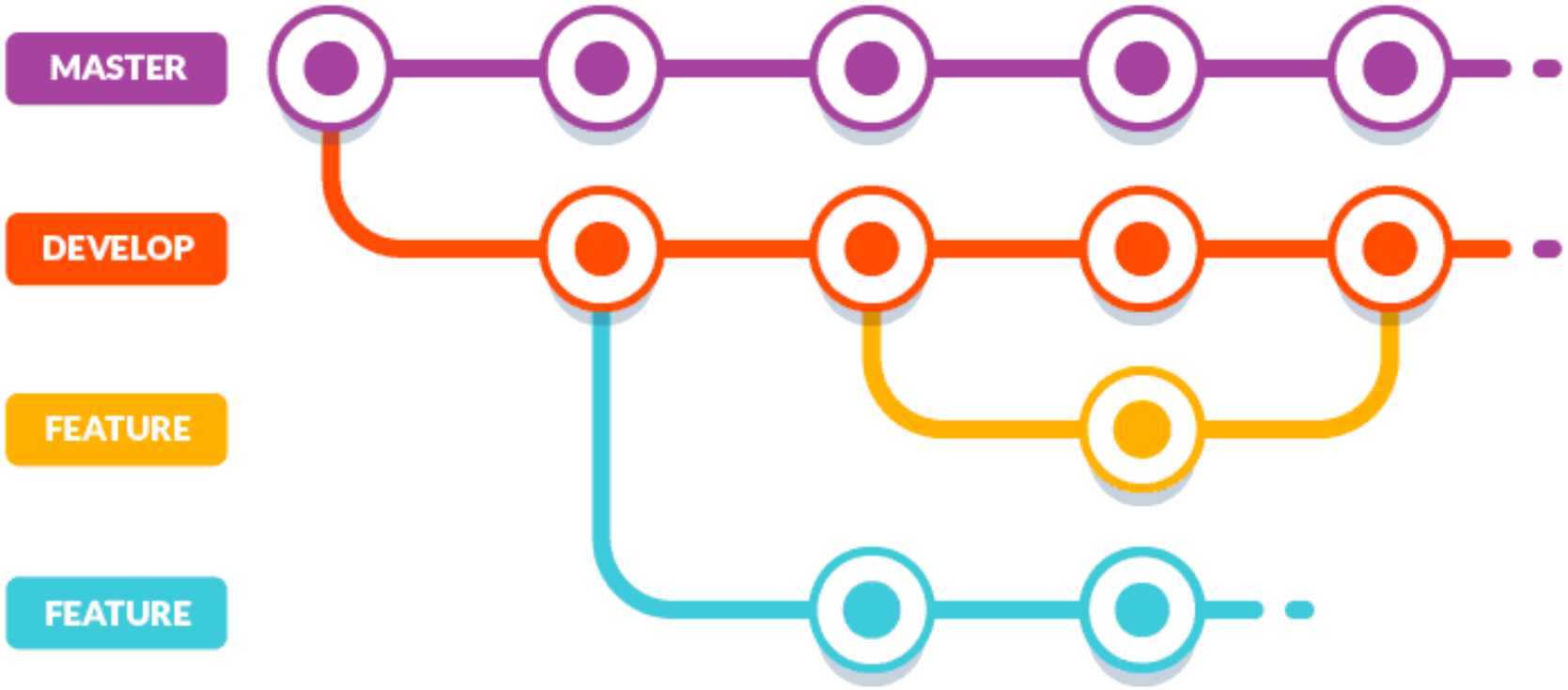
- Basic เหมาะกับ Project ขนาดเล็ก
- Feature Branch เหมาะกับ Project ขนาดกลาง
- Gitflow เหมาะกับ Project ขนาดใหญ่

Basic

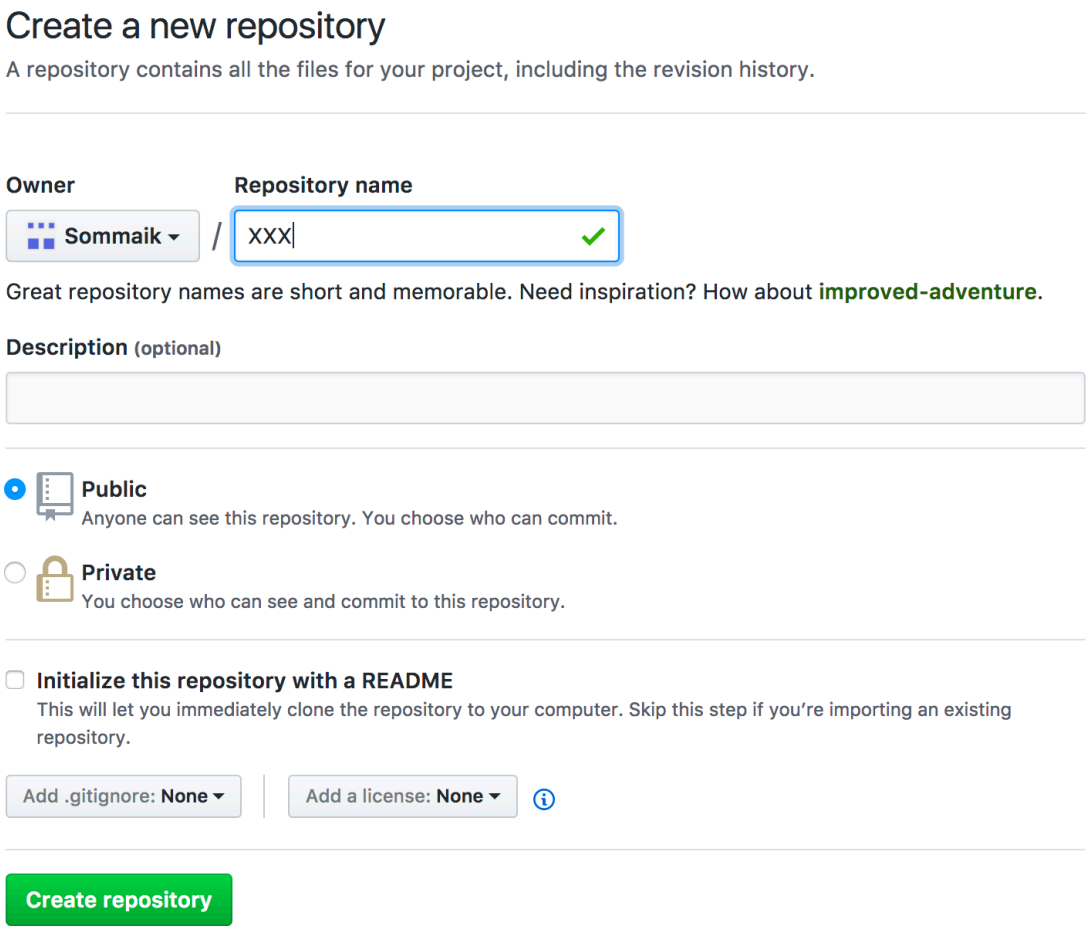
Feature Branch



Gitflow



Create a new Git Repository (Remote)



CONFIGURE TOOLING

Sets the name you want atached to your commit transactions

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

Sets the email you want atached 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 commited

- 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 paterns

- *.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

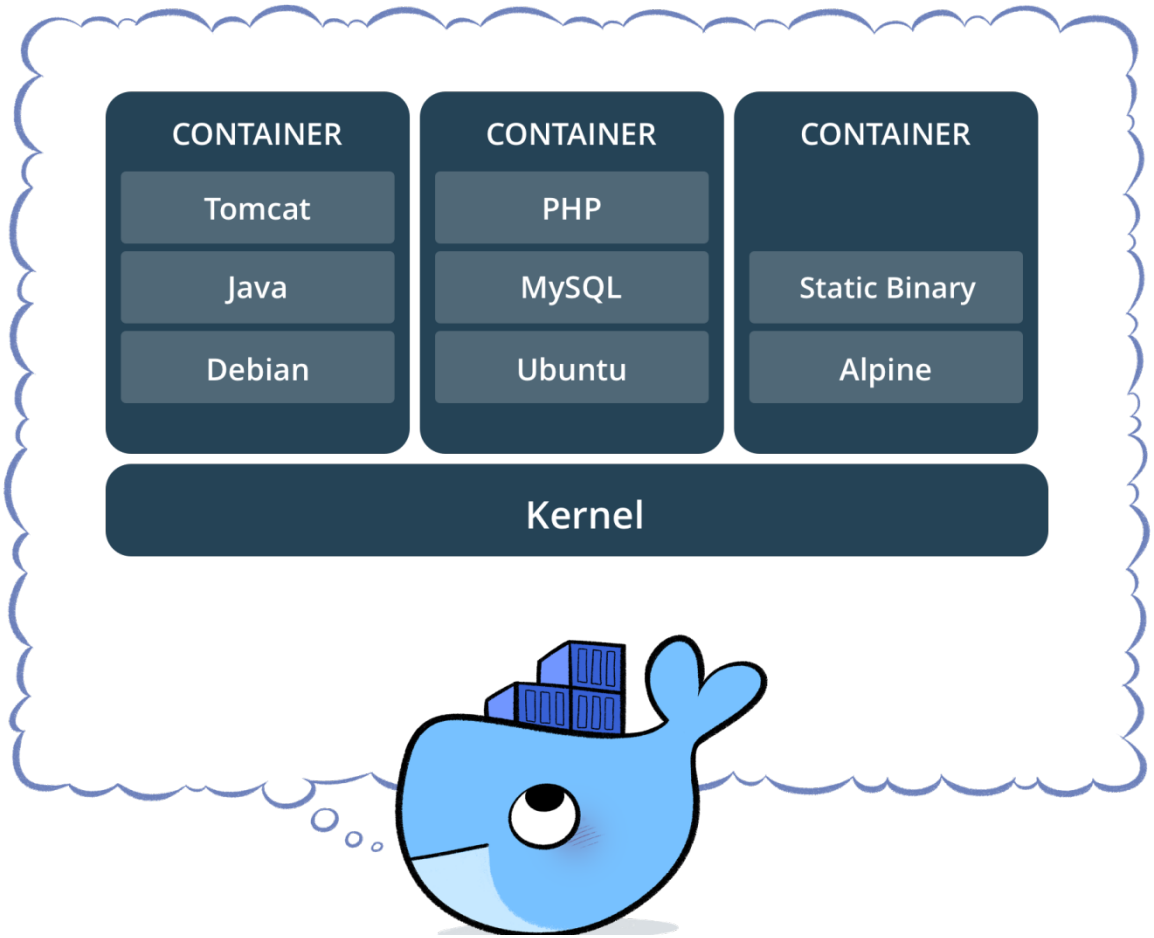
Downloads bookmark history and incorporates changes

- git pull

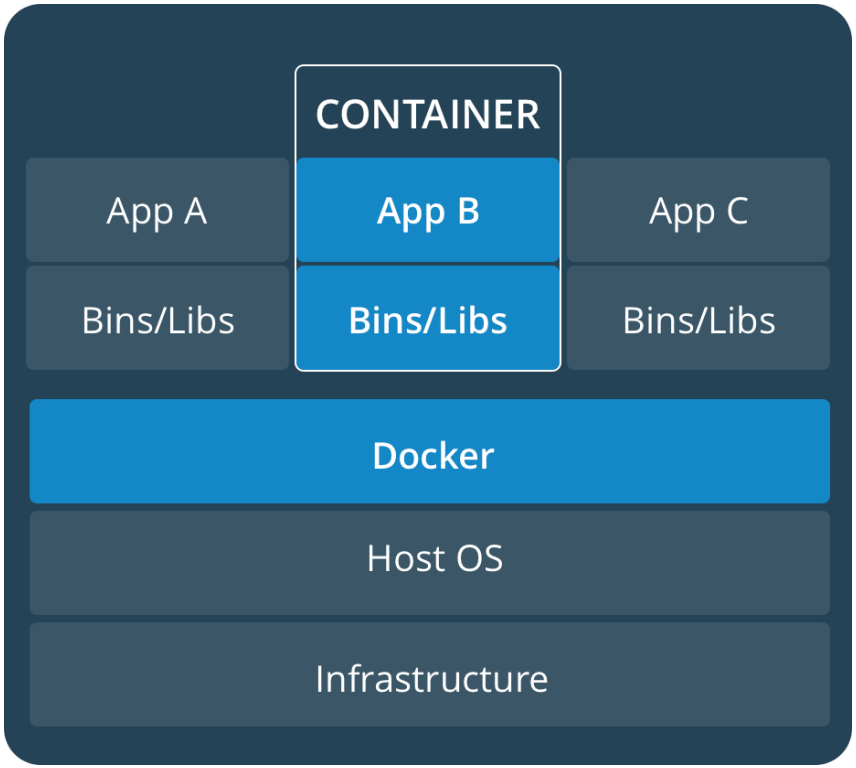
Containersvirtualize with

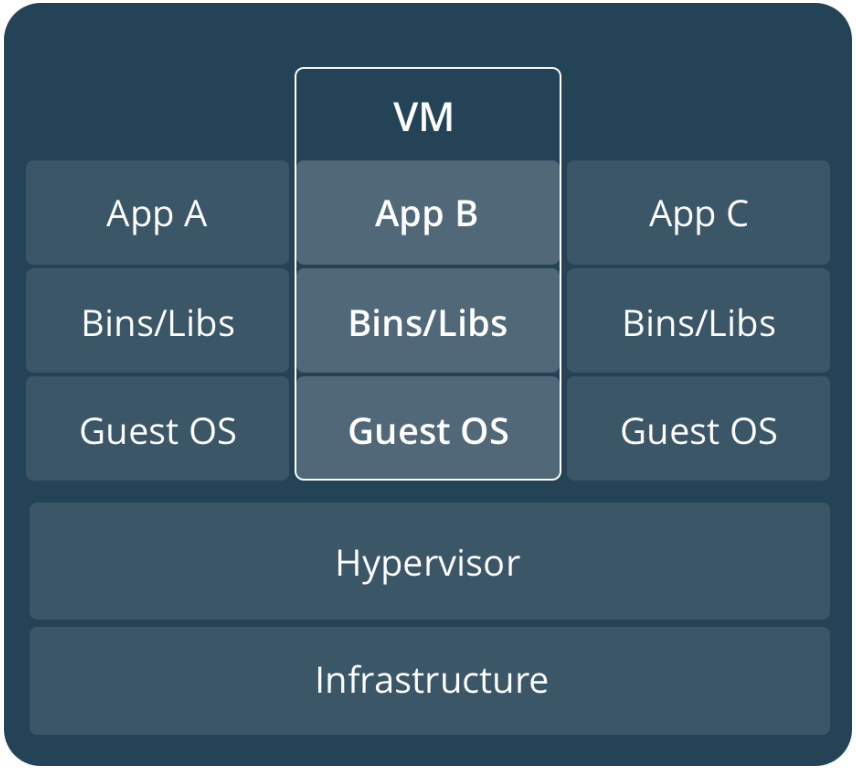
DOCKER

About Container

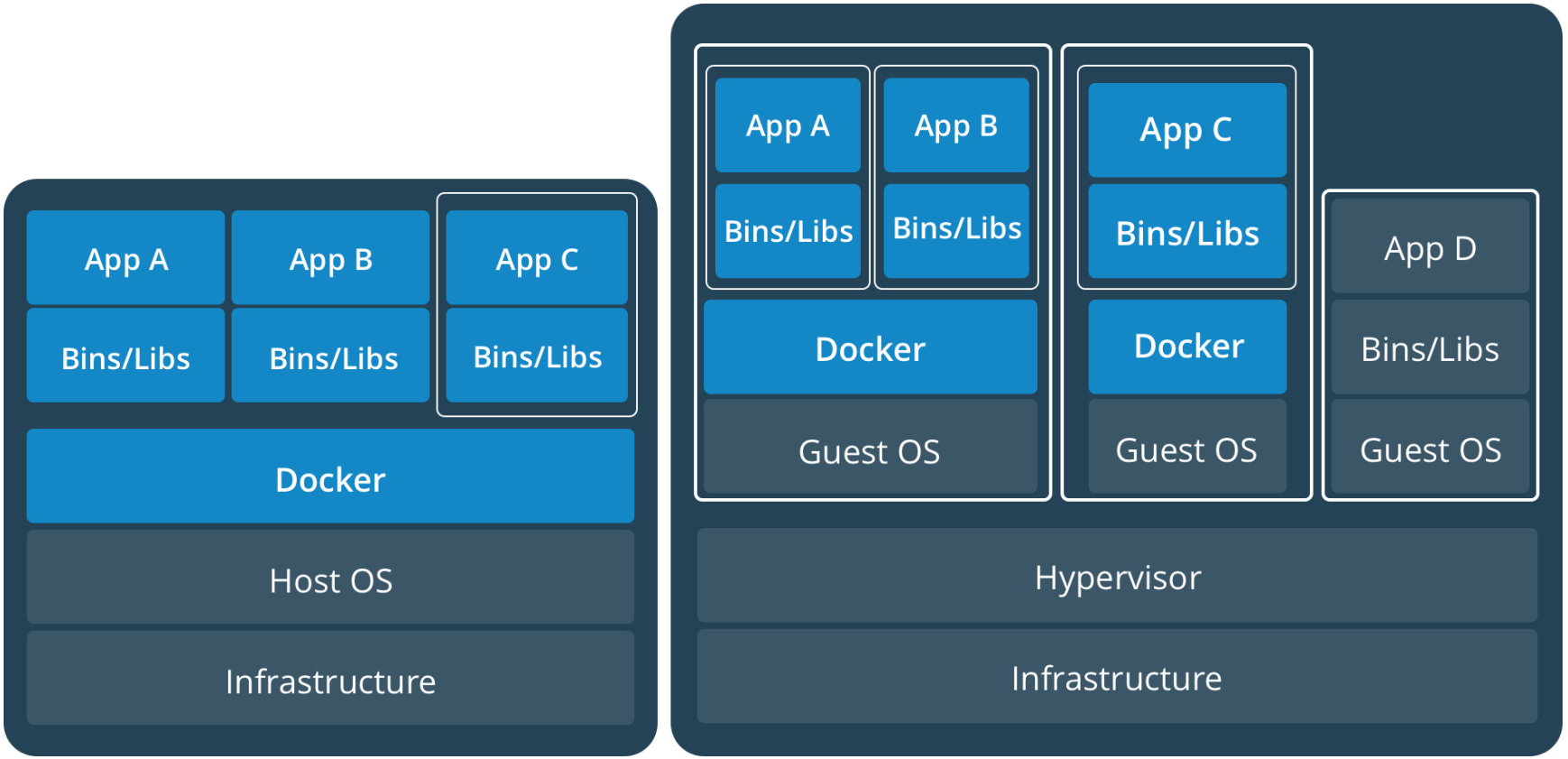


Comparing Containers and Virtual Machines

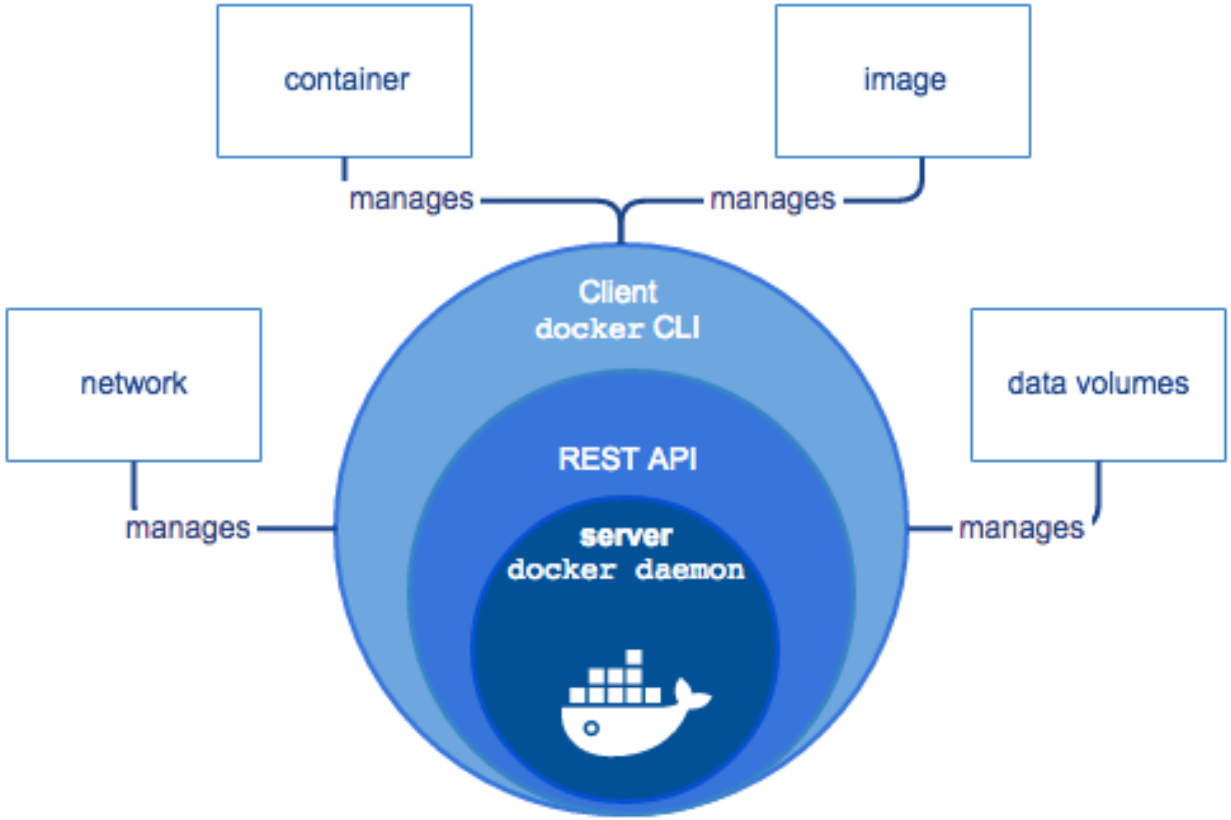




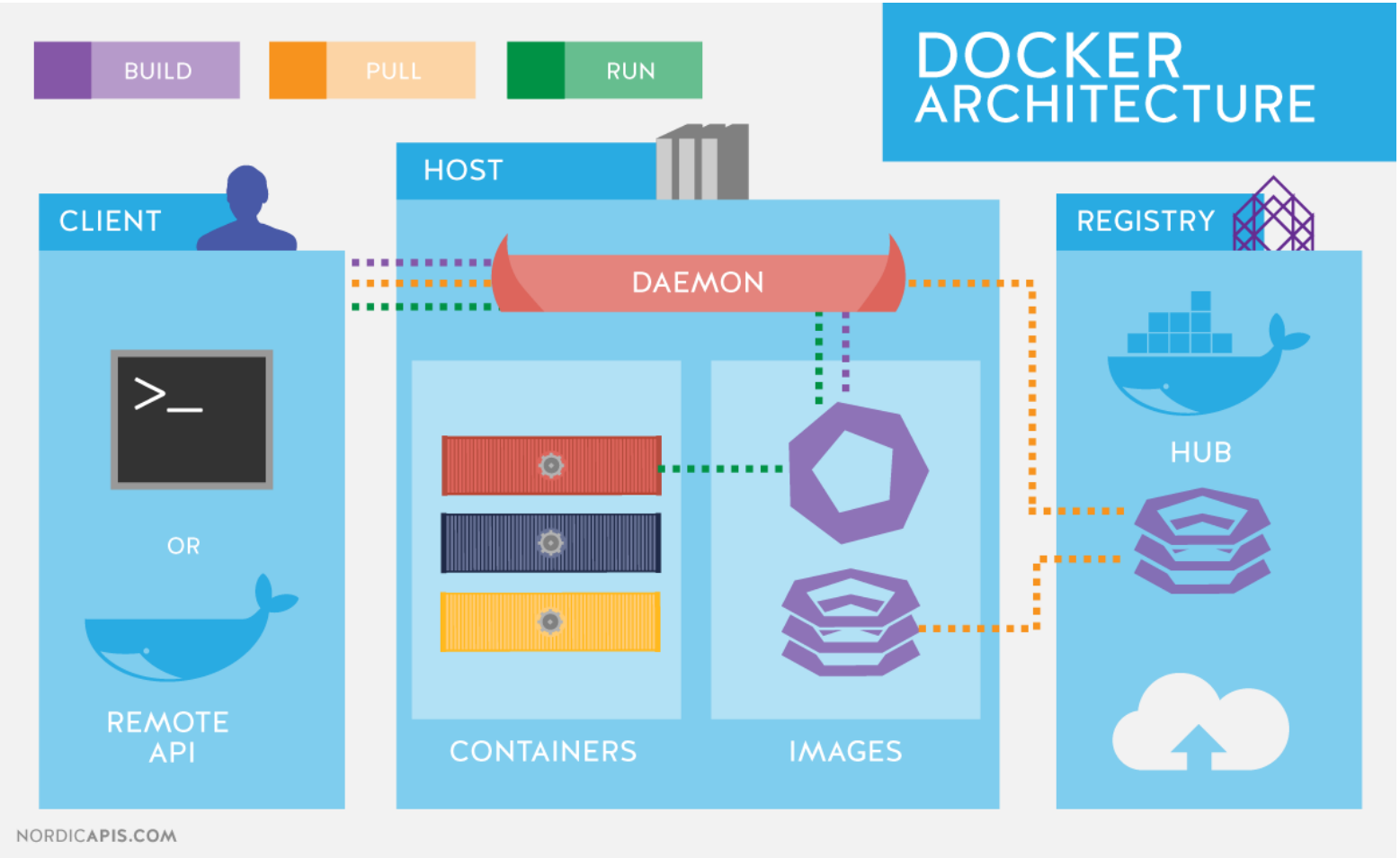
Containers and Virtual Machines Together



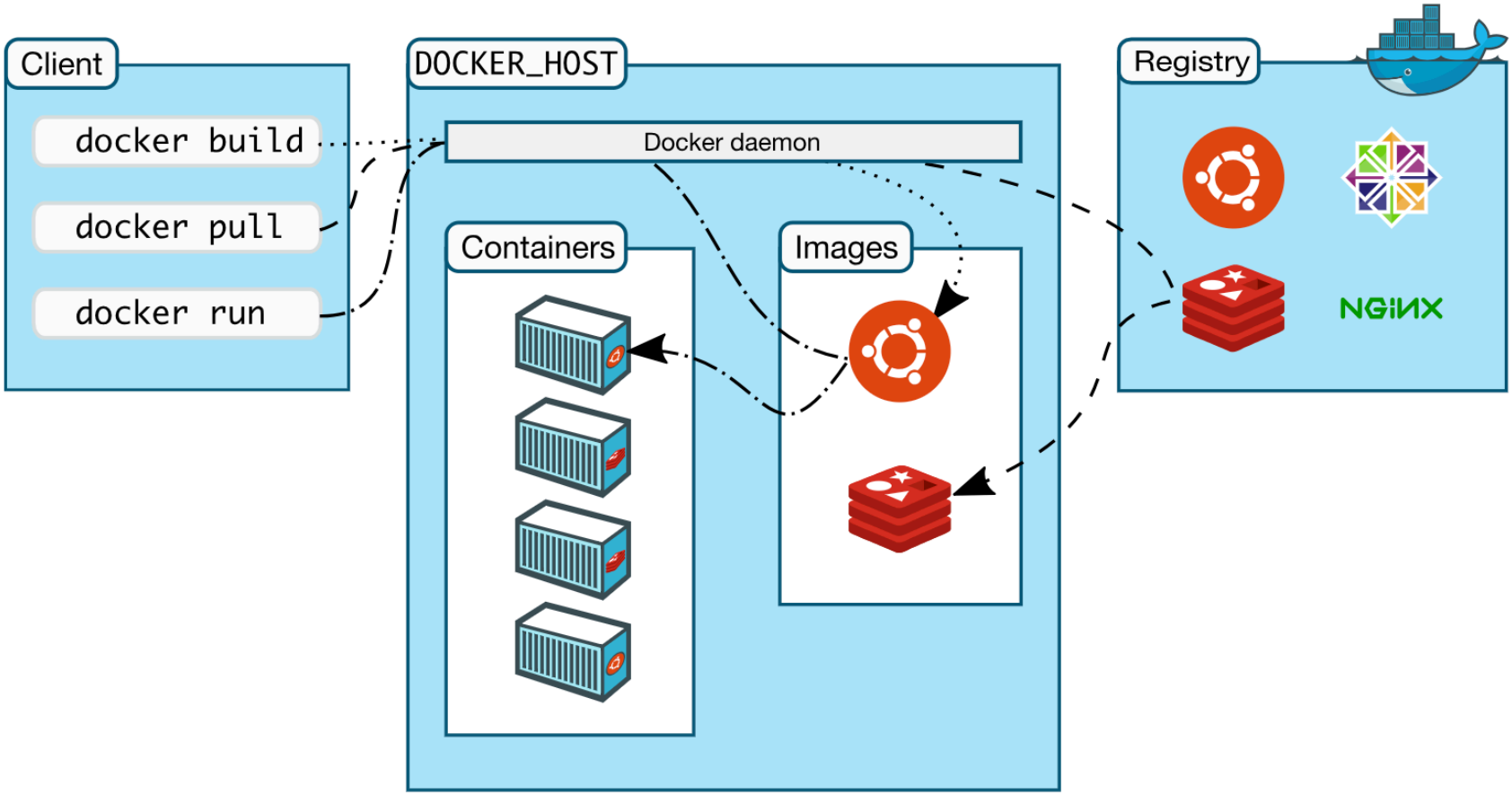
Docker engine



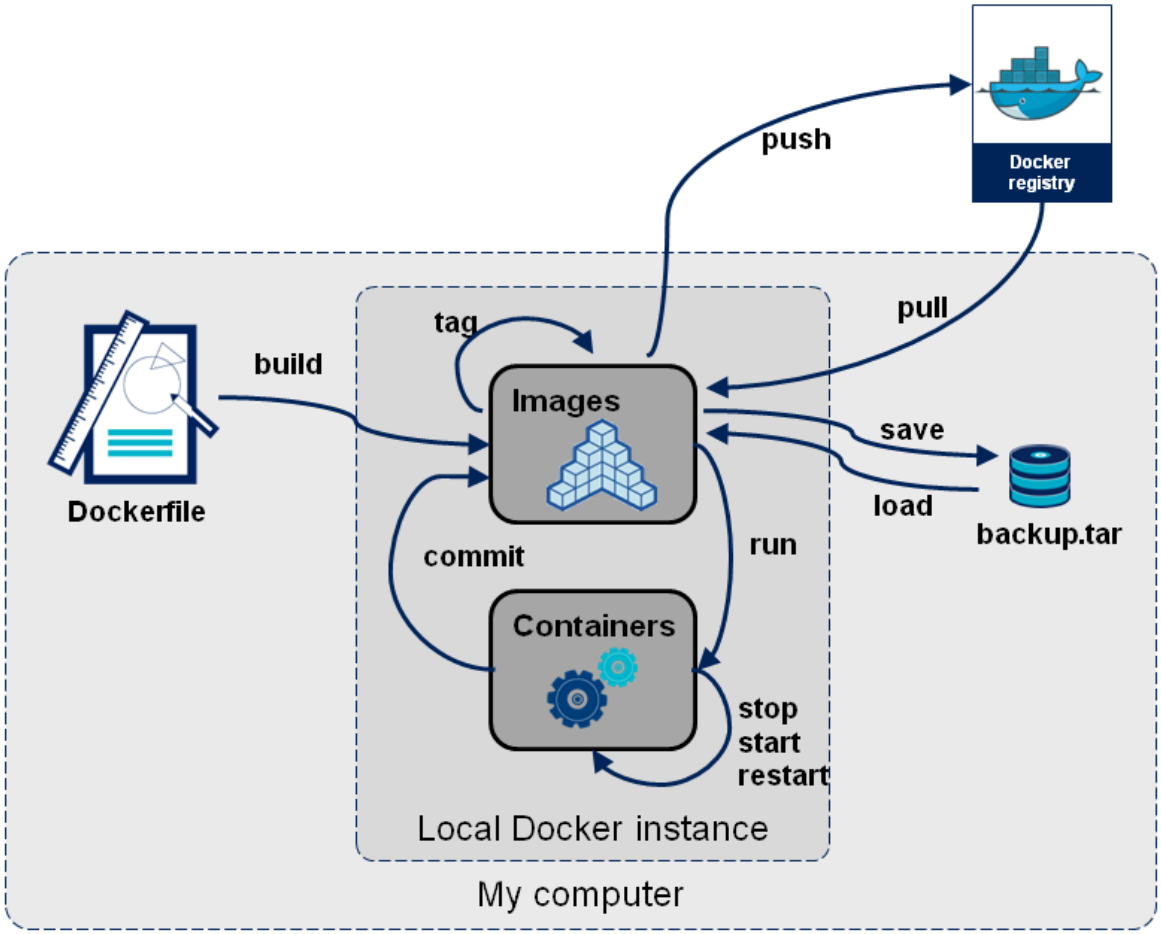
Docker Architecture



Docker Architecture#2



Docker Architecture#3



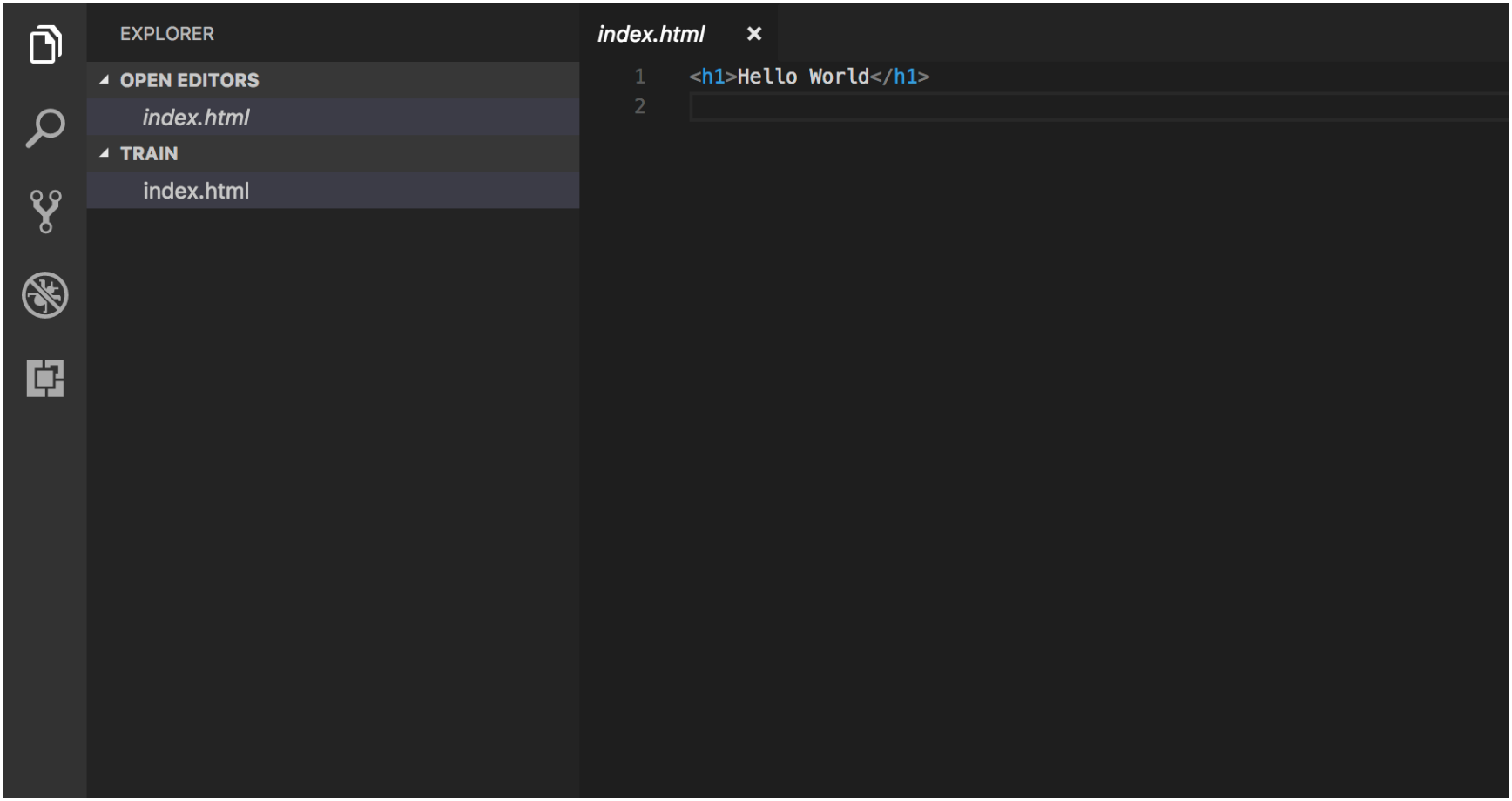
Hello World Docker

- เปิด cmd แล้วรันคำสั่งดังนี้

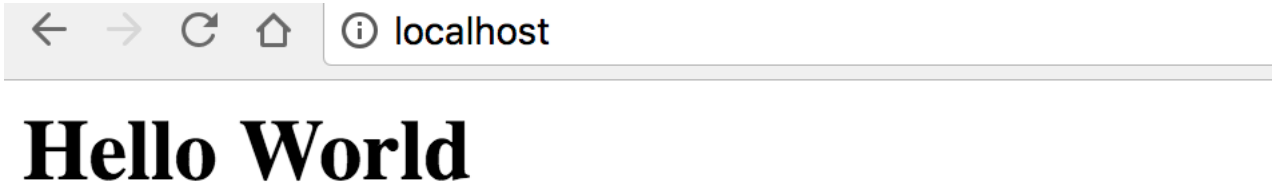
```
docker run --name some-nginx \
-p 80:80 \
-d nginx
```

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

สร้าง file index.html



เข้า url http://localhost



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

```yaml
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"
```

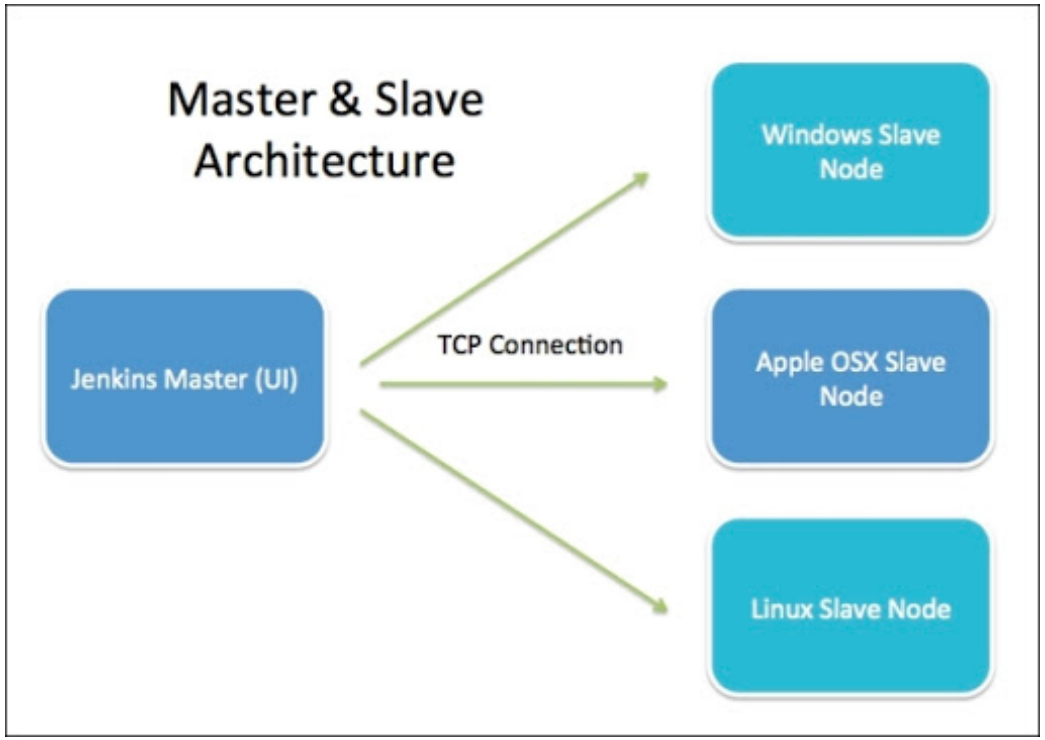
Docker Compose

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

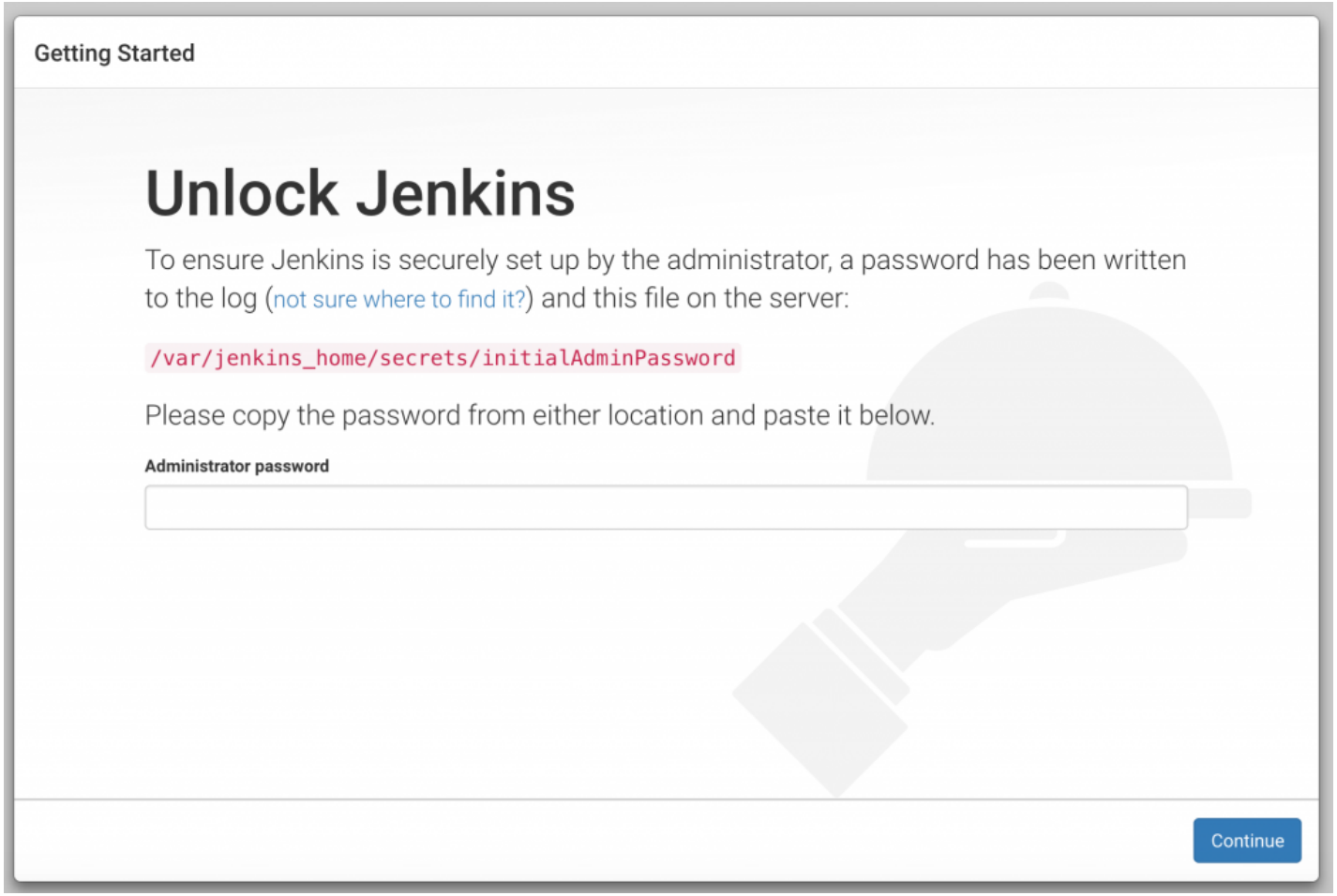
Automate Build and Deploy with

JENKINS

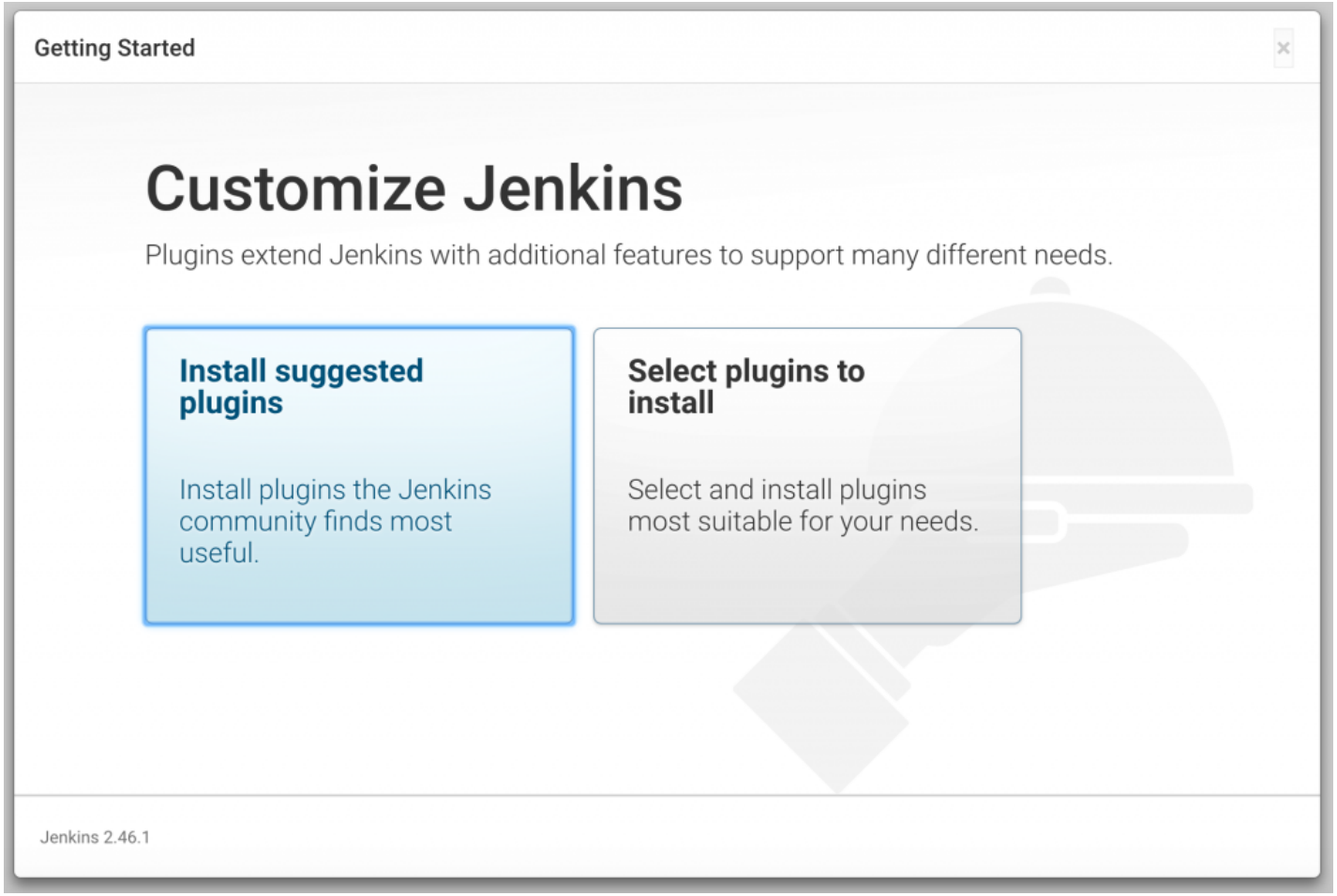
Master & Slave Architecture



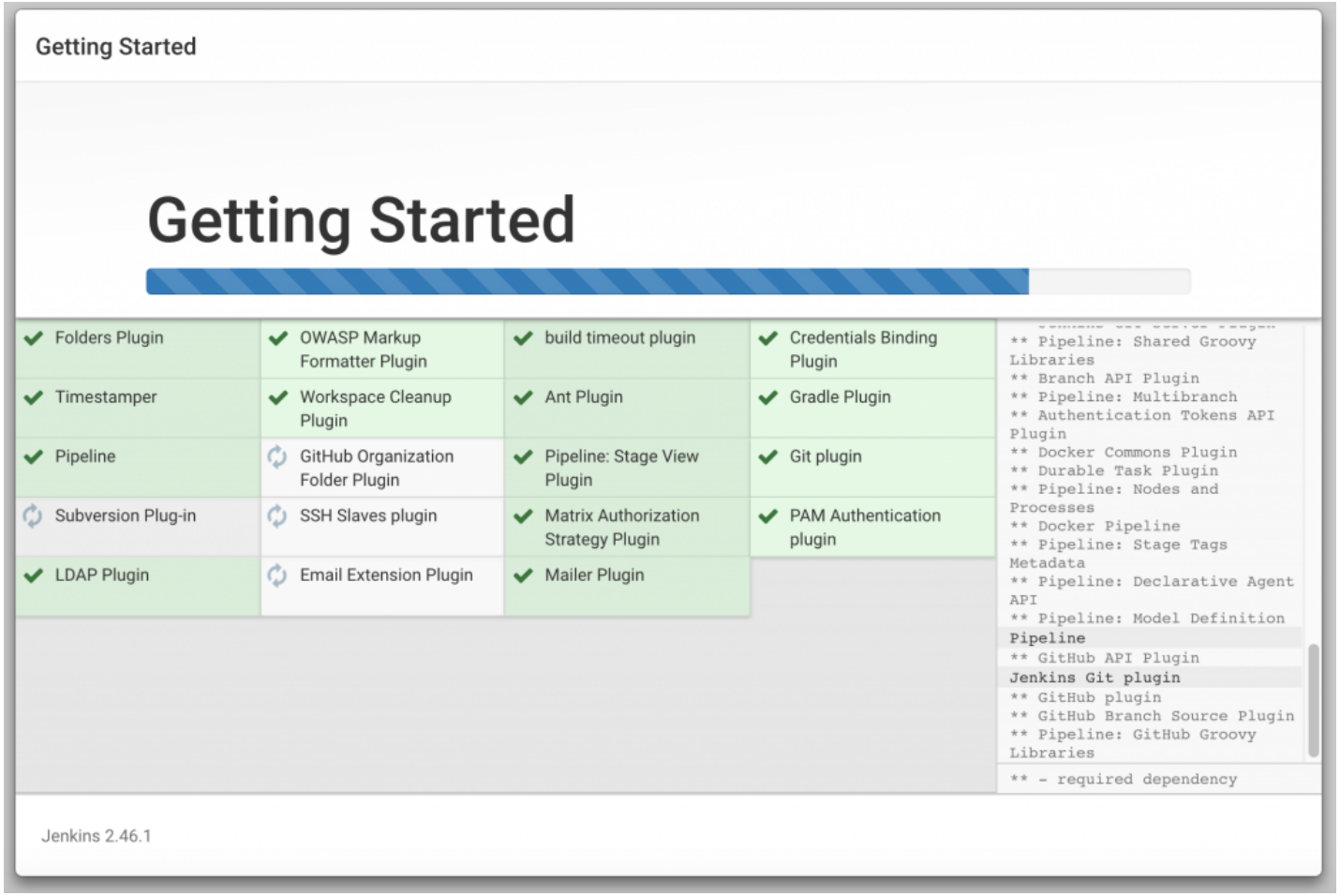
Unlock Jenkins



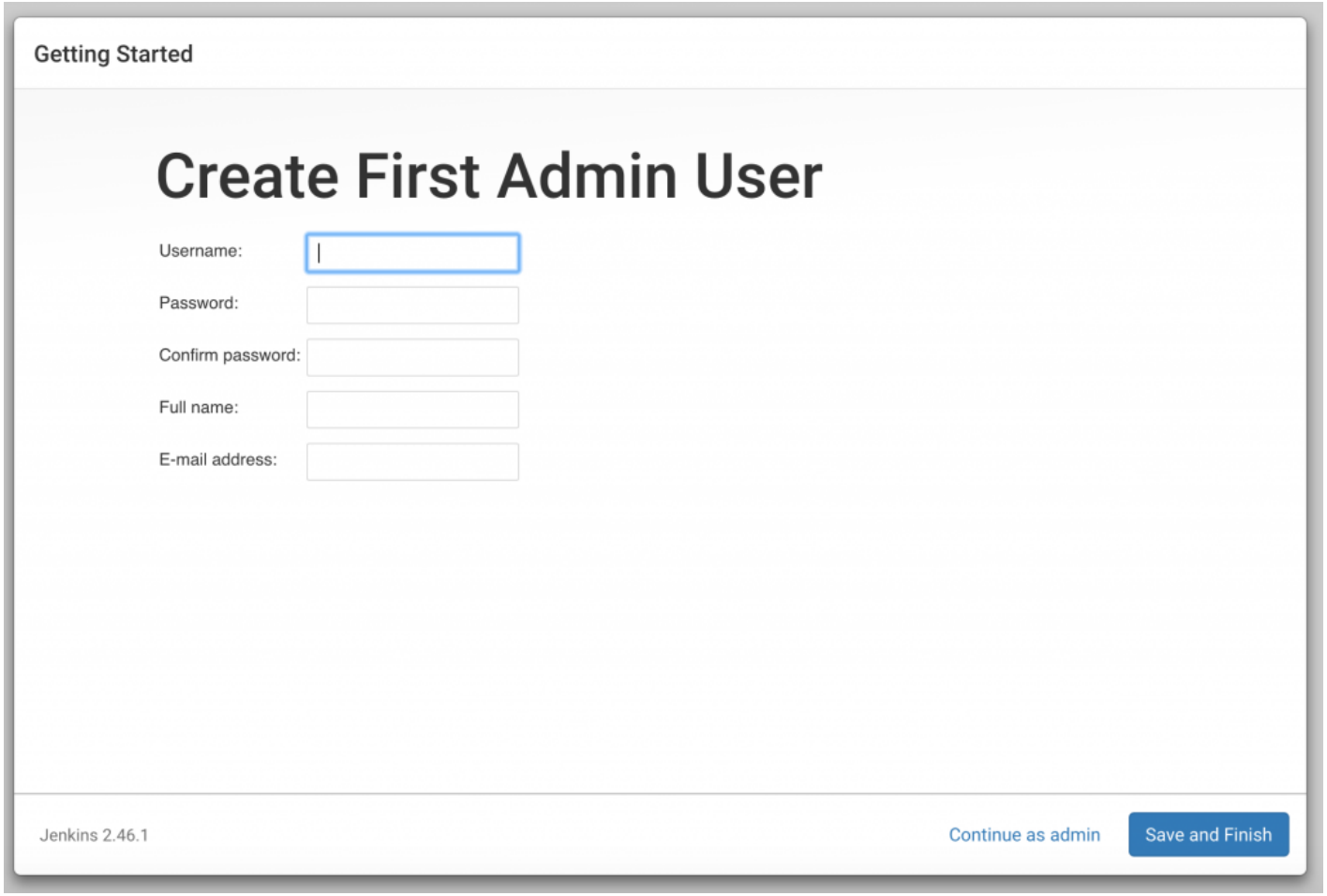
Install Plugins



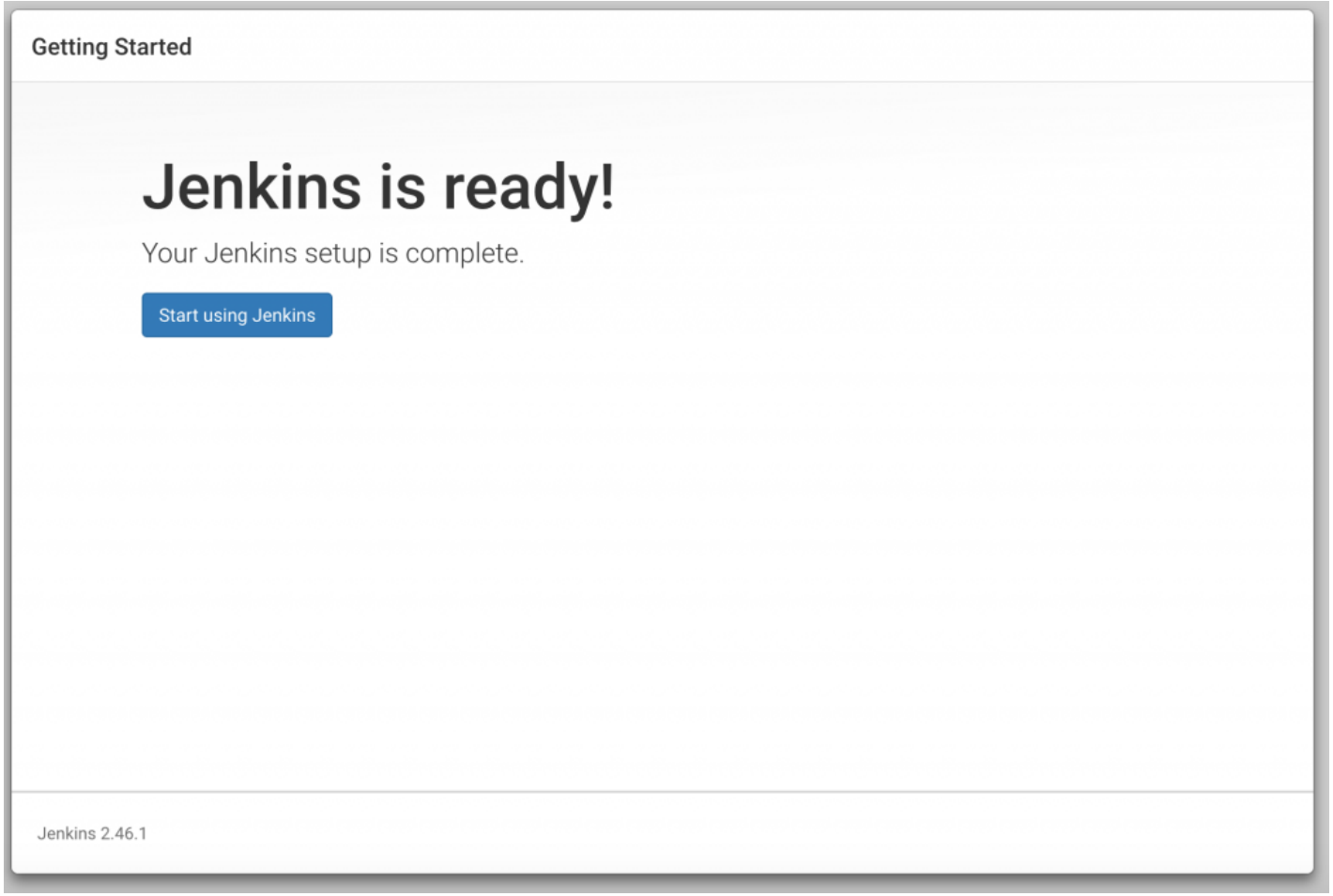
Loading Plugins



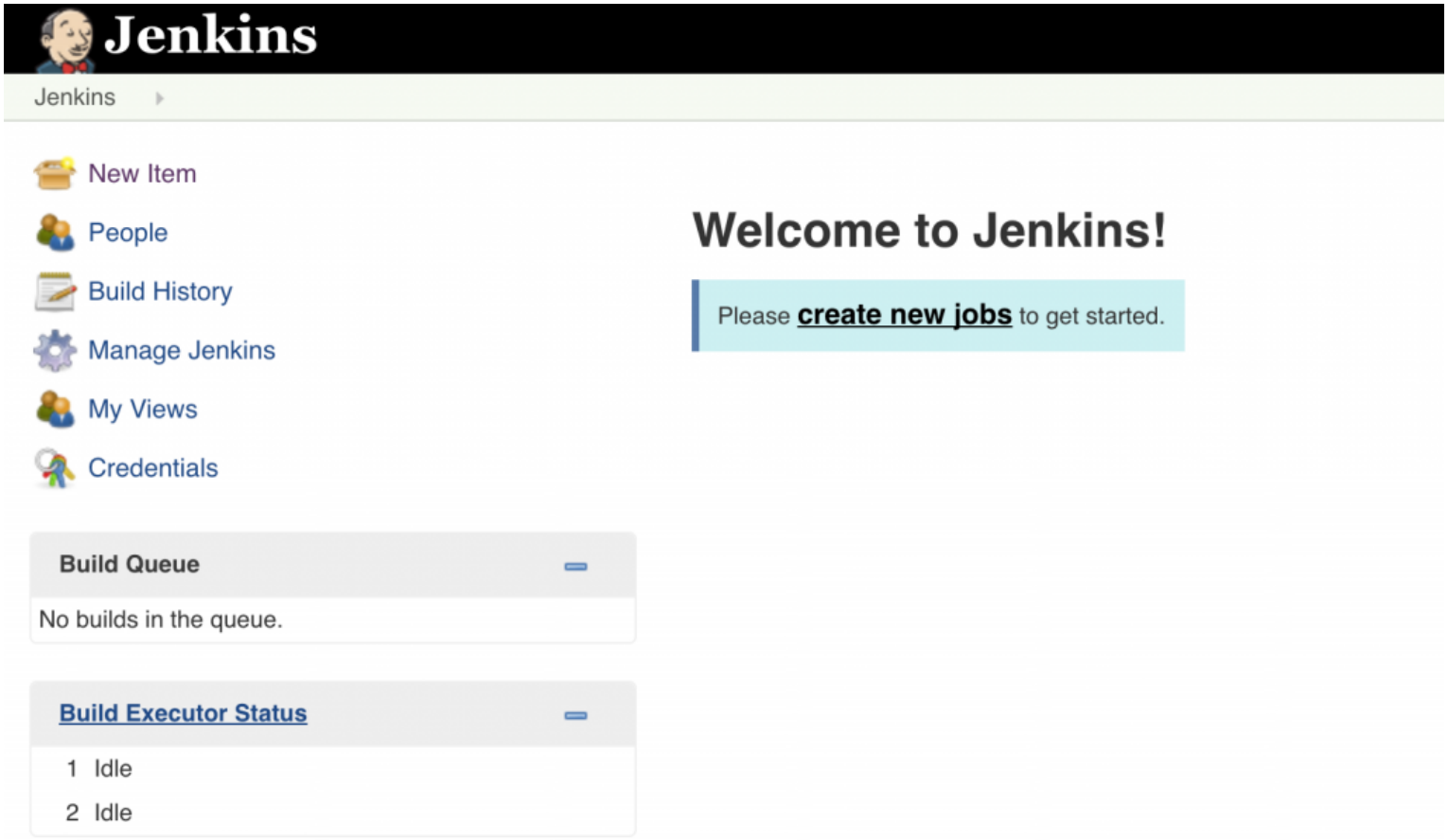
Create Admin User



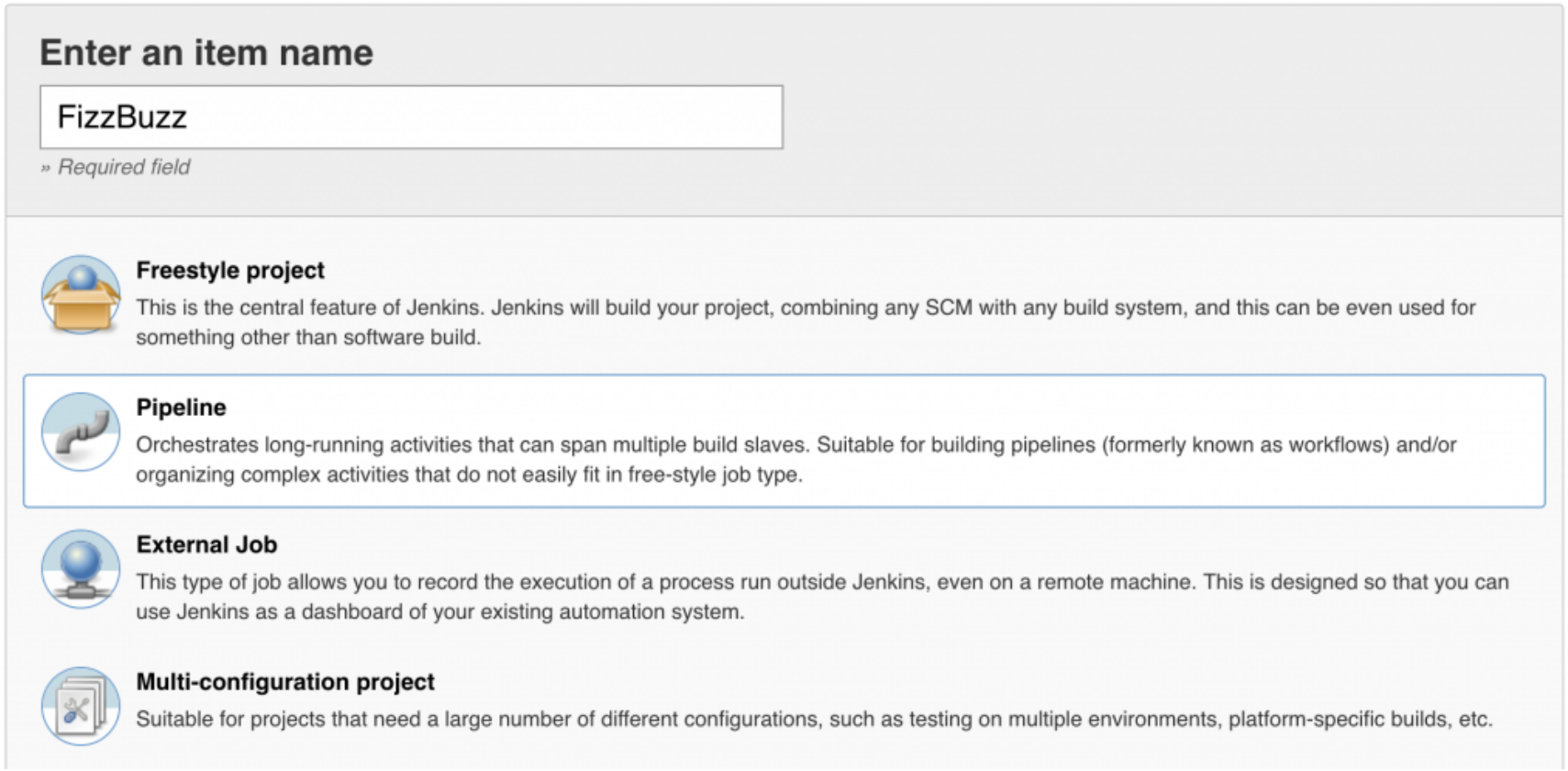
Setup Complete



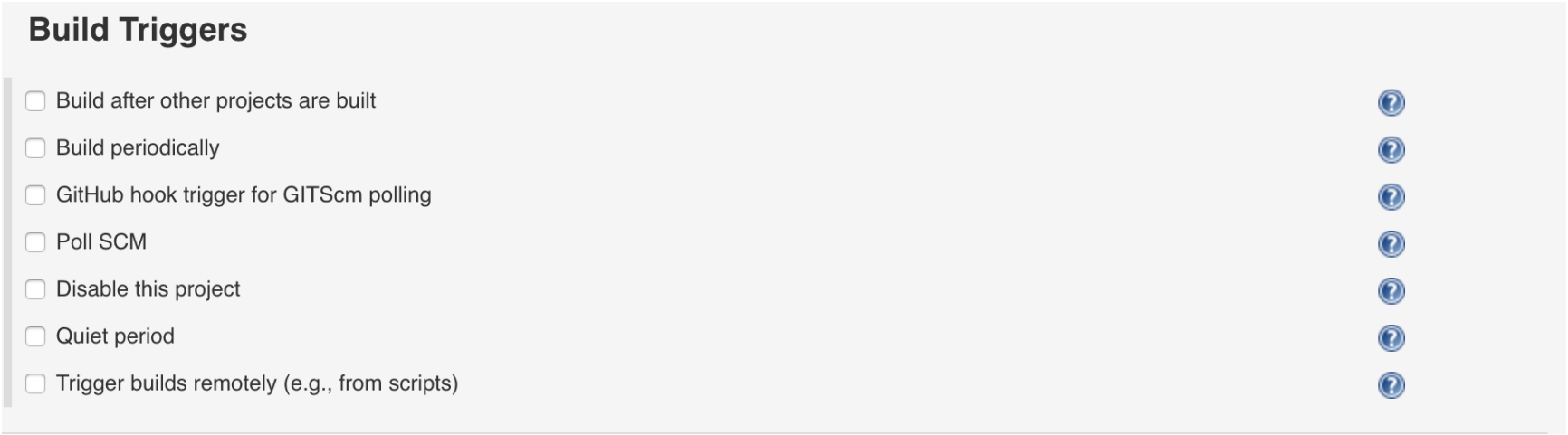
Create New Jobs



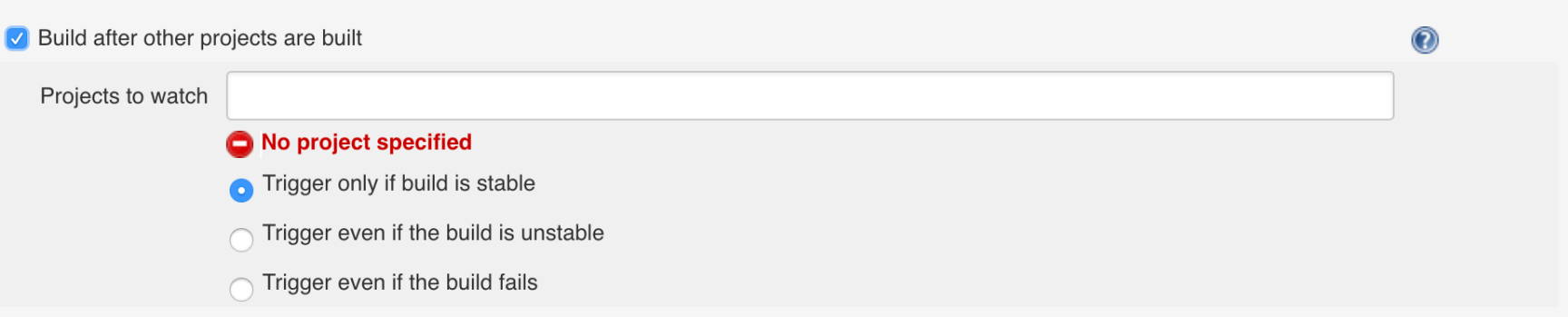
Choose Pipeline



Jenkins Config Build Trigger



Build after other projects are built



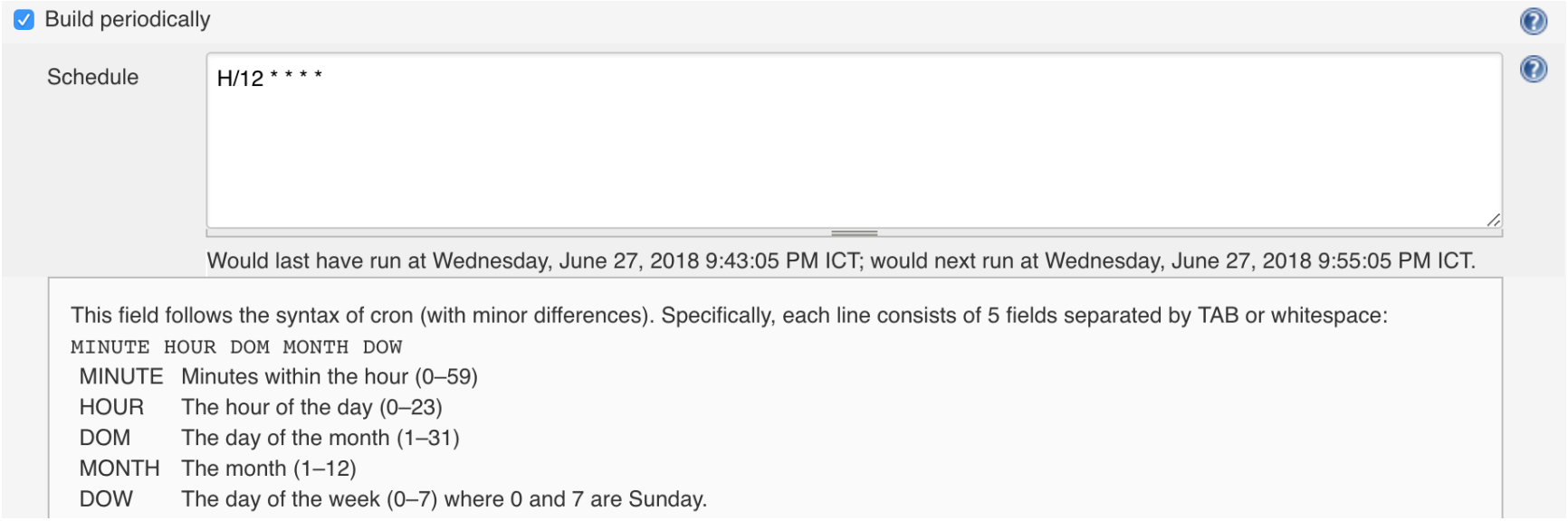
สั่งให้ทำงานอัตโนมัติ หลังจากที่ project กำหนดทำงานเสร็จ โดยมี 3 option คือ

ทำงานเมื่อ build สำเร็จ ไม่มี warning

ทำงานเมื่อ build สำเร็จ แบบมี warning

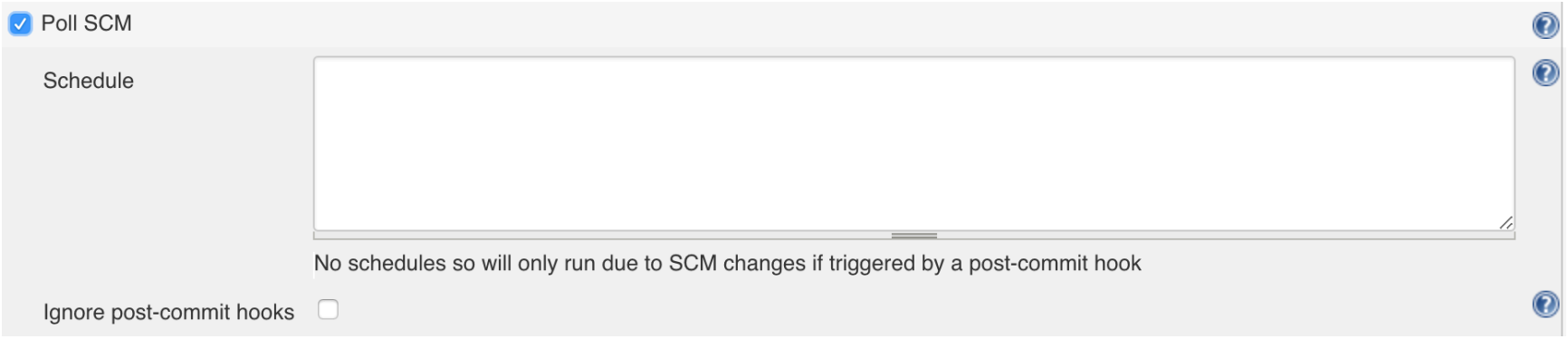
ทำงานไม่สำเร็จ

Build periodically



สั่งให้ทำงานอัตโนมัติตามช่วงเวลาที่กำหนด โดยสามารถกำหนดเวลาได้อยู่ในรูปแบบ crontab format

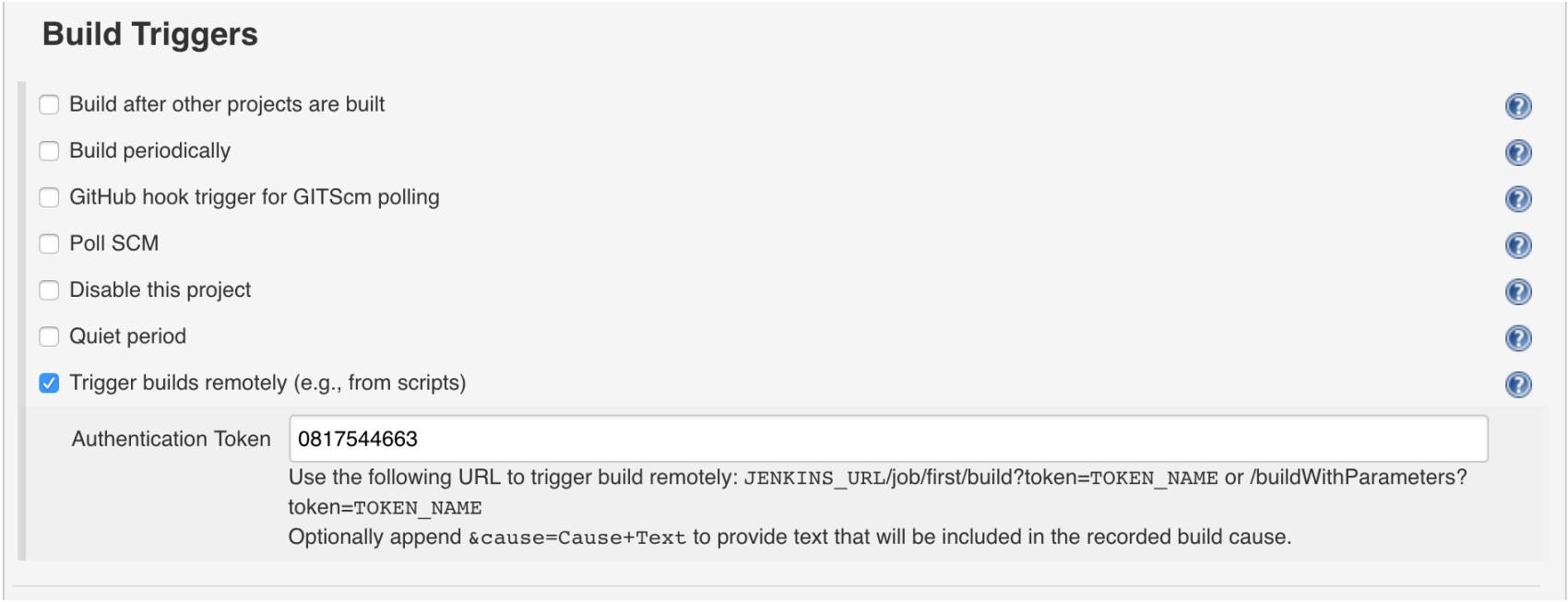
Poll SCM



สั่งให้ทำงานอัตโนมัติตามช่วงเวลาที่กำหนด โดยสามารถกำหนดเวลาได้อยู่ในรูปแบบ crontab format

โดยมีลักษณะการทำงานคือจะไปทำการตรวจสอบดูก่อนว่า source code มีการเปลี่ยนแปลงหรือไม่ ถ้าไม่มีเปลี่ยนแปลงก็จะไม่ทำการ build job

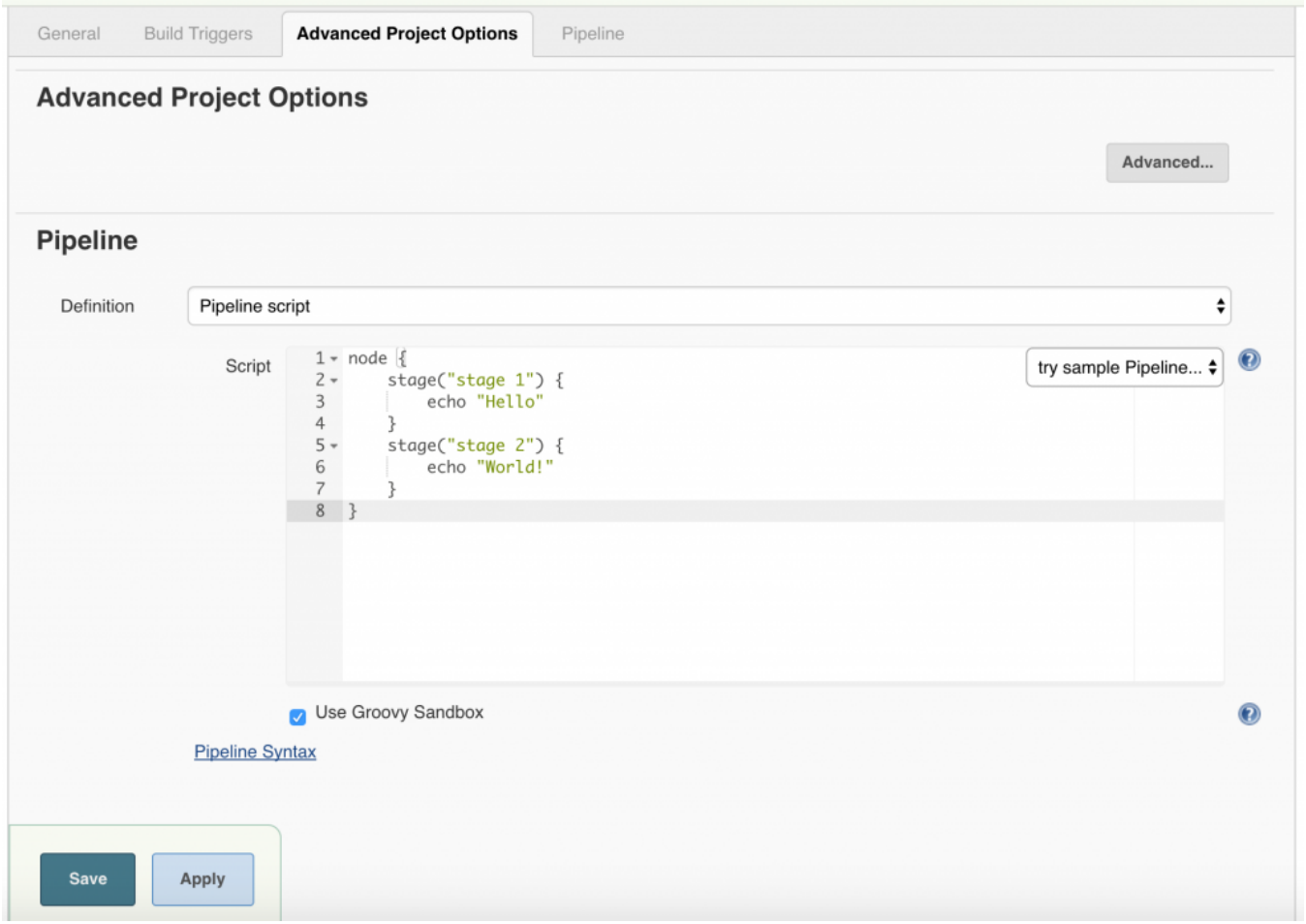
Trigger builds remotely



เป็นการตั้งค่าให้สามารถสั่งให้ job ทำงานโดยการเรียกผ่าน url เช่น ใส่ค่า Authenication 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

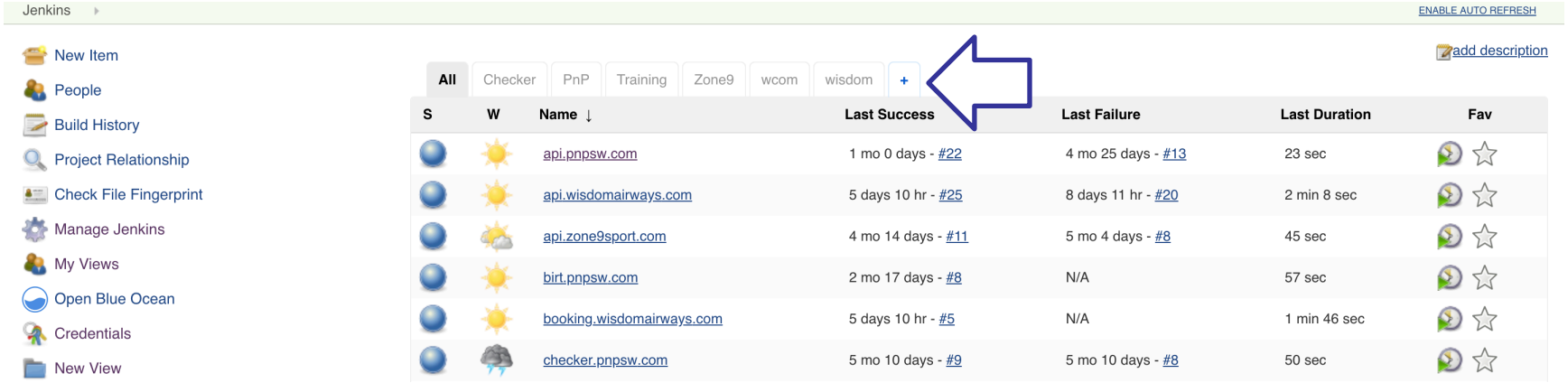
Pipeline script



Pipeline template

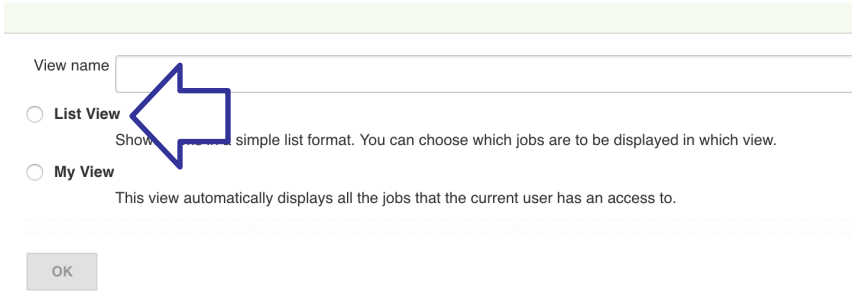
```
pipeline {
    agent any
    environment {
        APP_NAME = "test app name"
    }
    stages {
        stage('Build Image'){
            steps {
                sh "echo ${env.APP_NAME}"
            }
        }
    }
}
```

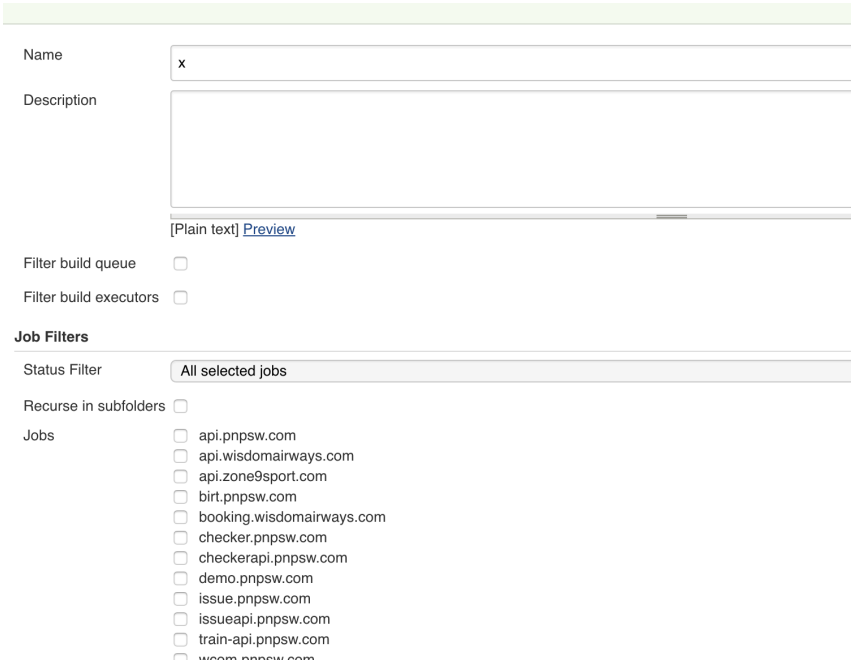
Jenkins Job List



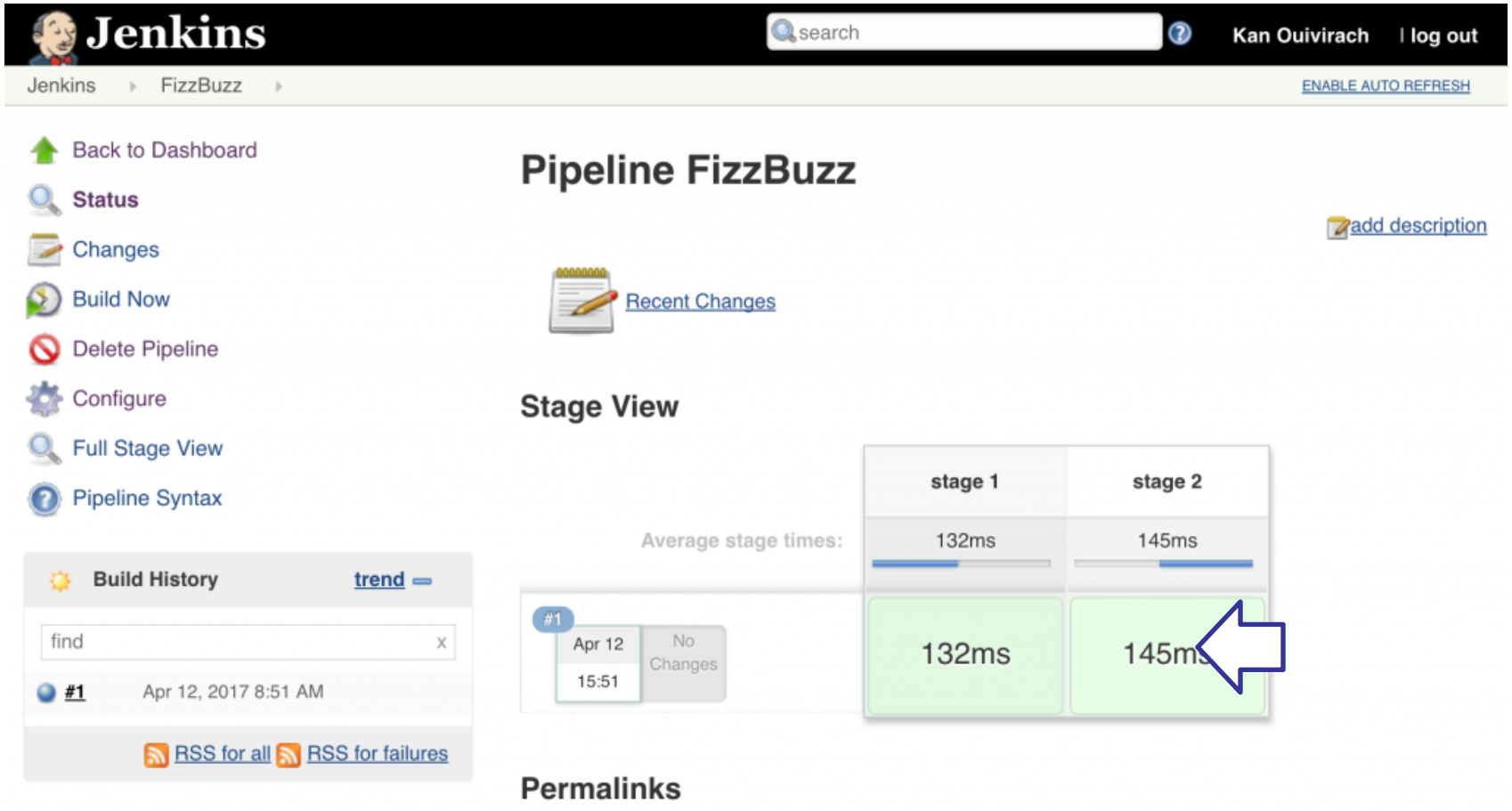
กดเครื่องหมาย + เพื่อทำการเพิ่ม tab เพื่อการกรองข้อมูลของ job

Set List View

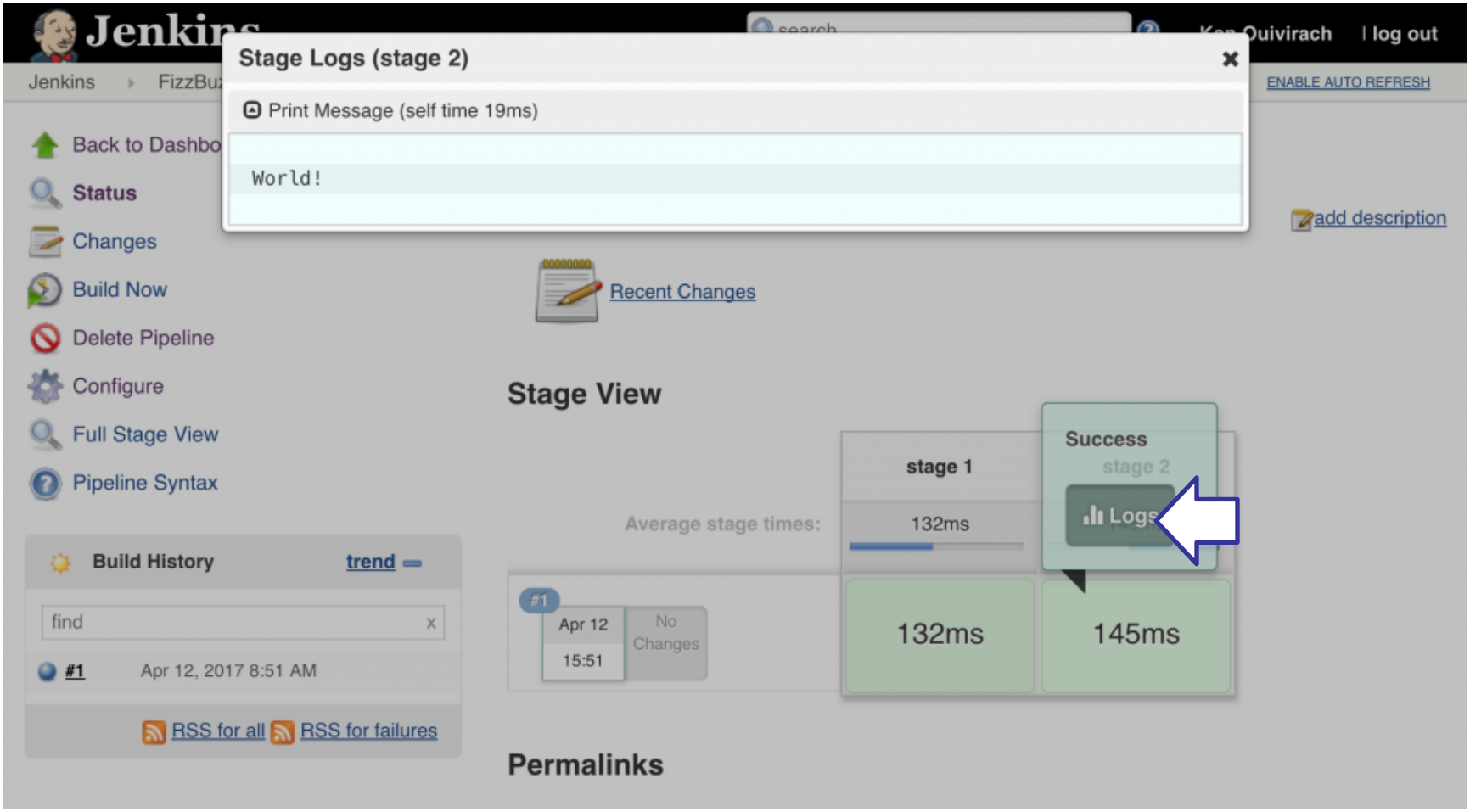




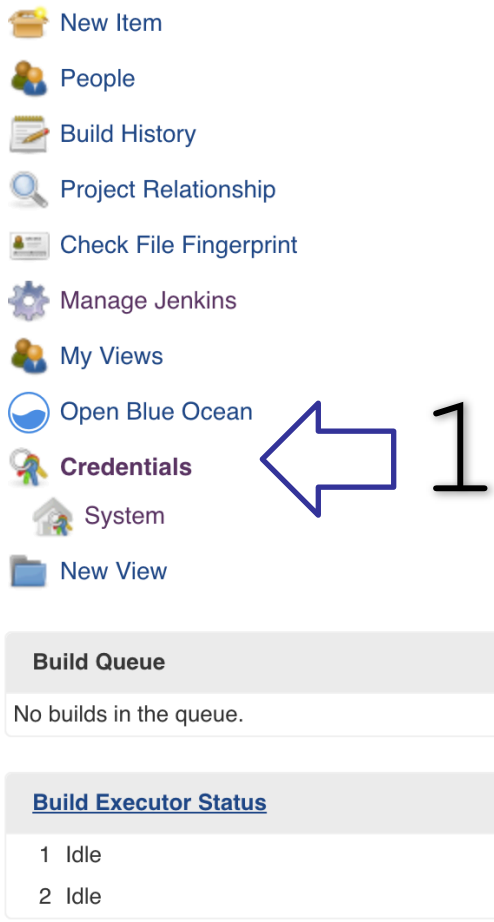
Logs


Jenkins
search
Kan Ouivirach
log out
Jenkins
FizzBuzz
ENABLE AUTO REFRESH
Back to Dashboard
Status
Changes
Build Now
Delete Pipeline
Configure
Full Stage View
Pipeline Syntax
Pipeline FizzBuzz
add description
Recent Changes
Stage View
stage 1
stage 2
Average stage times:
132ms
145ms
#1
Apr 12
15:51
No Changes
132ms
145ms
Build History
trend
find
#1
Apr 12, 2017 8:51 AM
RSS for all
RSS for failures
Permalinks

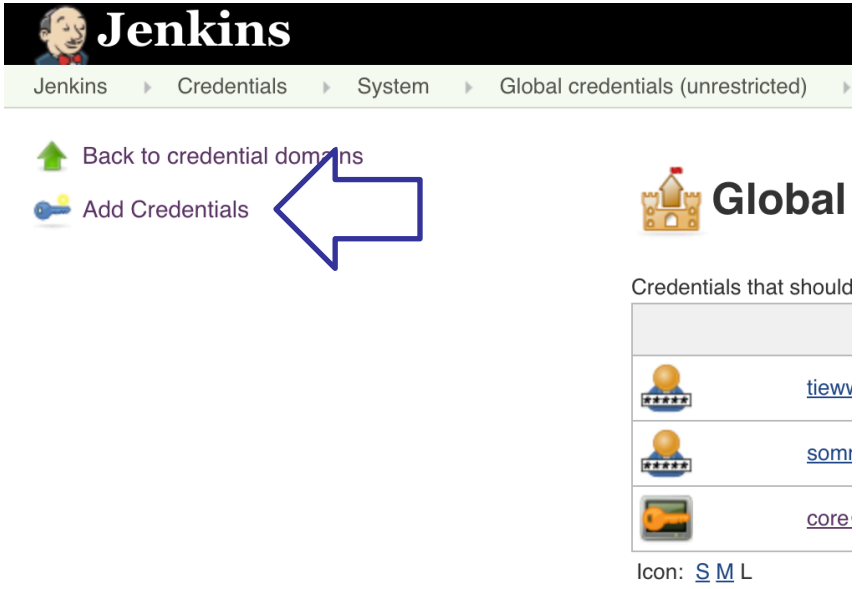
Logs

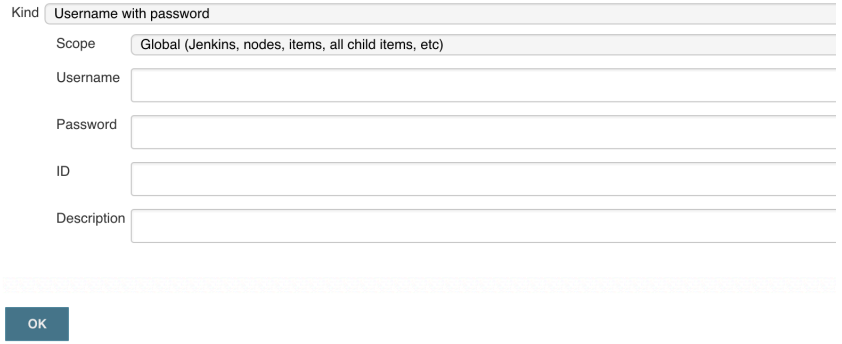

Jenkins
search
Kan Quivirach
log out
Jenkins
FizzBuz
ENABLE AUTO REFRESH
Stage Logs (stage 2)
Print Message (self time 19ms)
World!
Back to Dashbo
Status
Changes
Build Now
Delete Pipeline
Configure
Full Stage View
Pipeline Syntax
add description
Recent Changes
Stage View
Success
stage 1
stage 2
Average stage times:
132ms
Logs
Build History
trend
find
#1
Apr 12
15:51
No Changes
132ms
145ms
#1
Apr 12, 2017 8:51 AM
RSS for all
RSS for failures
Permalinks

Add Credentials #1

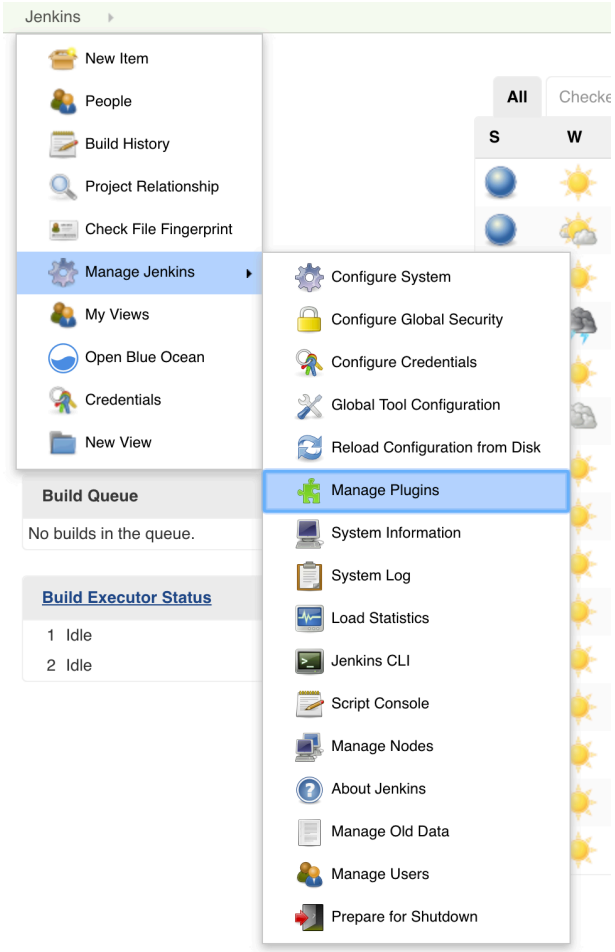


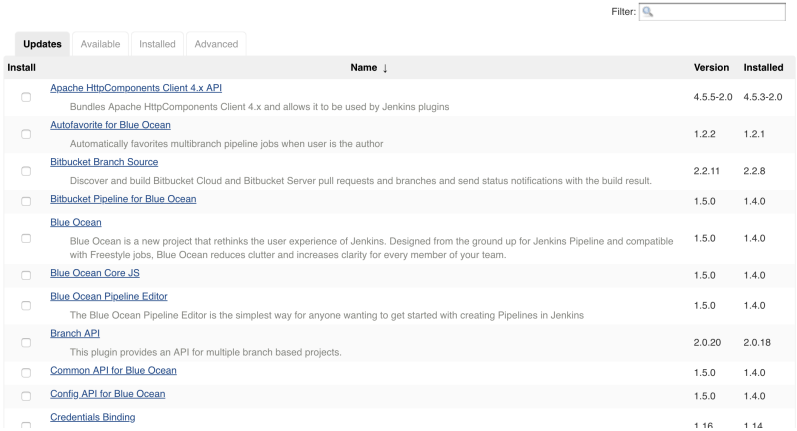
Add Credentials #2



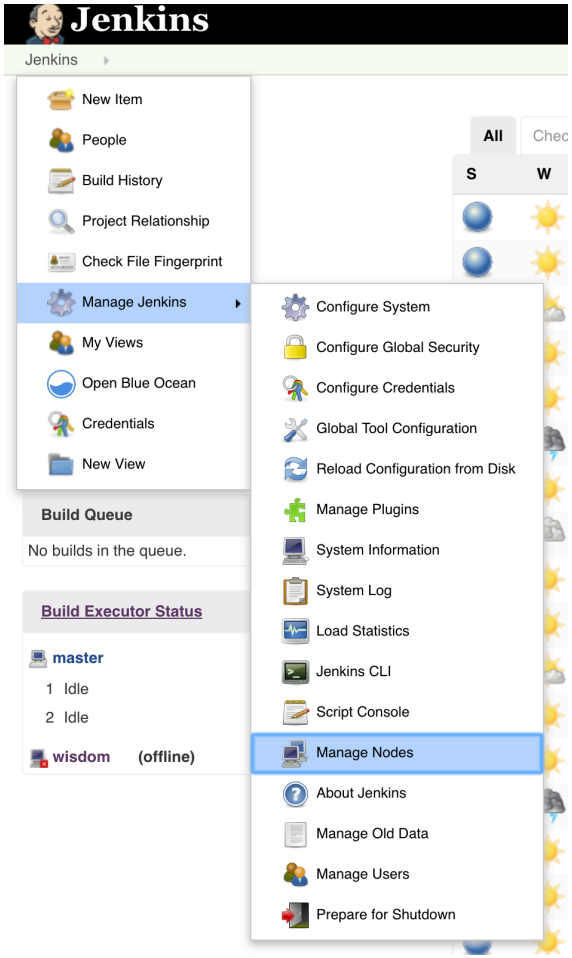


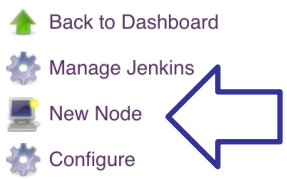
Install Plugins

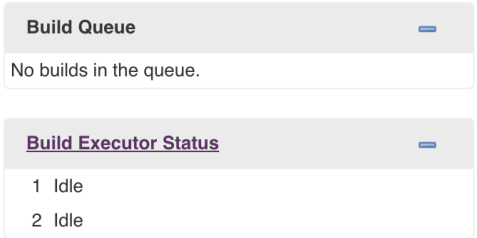


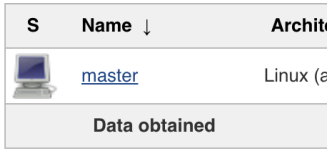


Add new node

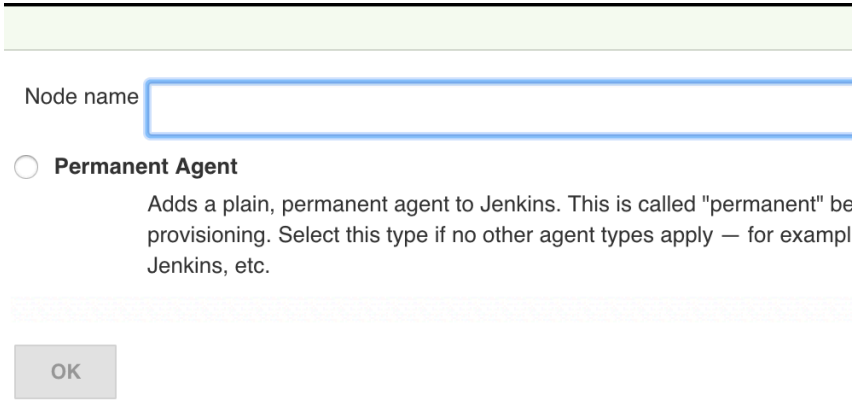


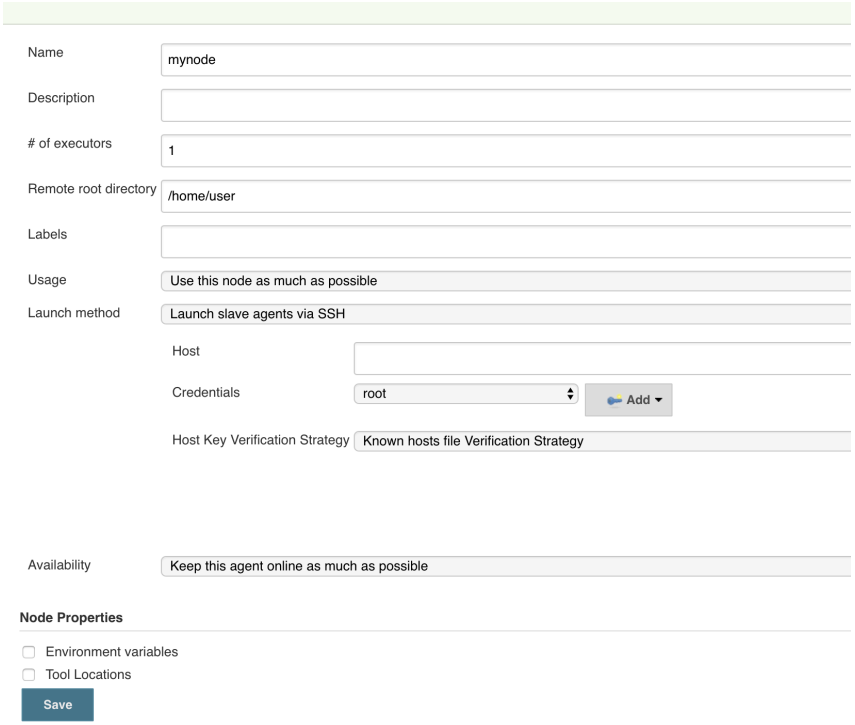






Add new node





Add new node

- Remote root directory : ใส่ชื่อ path ของเครื่องปลายทาง
- Labels : ชื่อจะถูกเรียกใช้ใน pipeline
- Usage : Only Build job with label expression matching this node
- Launch Method : Launch slave agents via SSH
  - Host : ชื่อเครื่องหรือ ip
  - Credentials : ค่า Credentials ที่ตั้งไว้
  - 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.

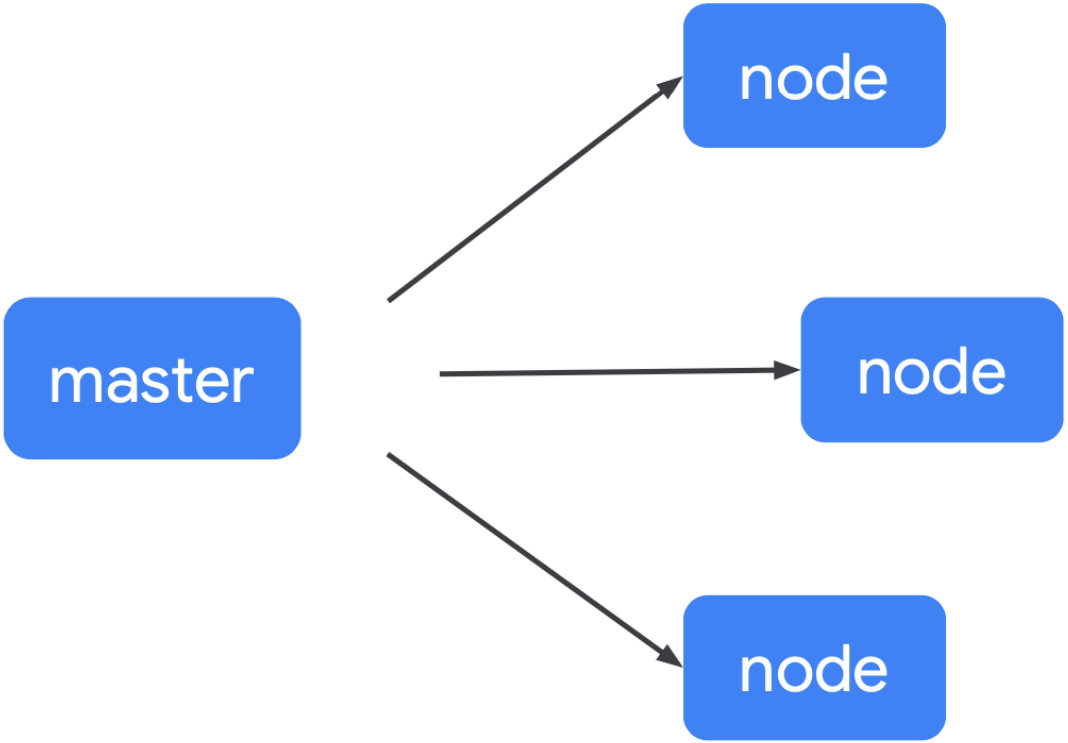
Run K8s Anywhere

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.

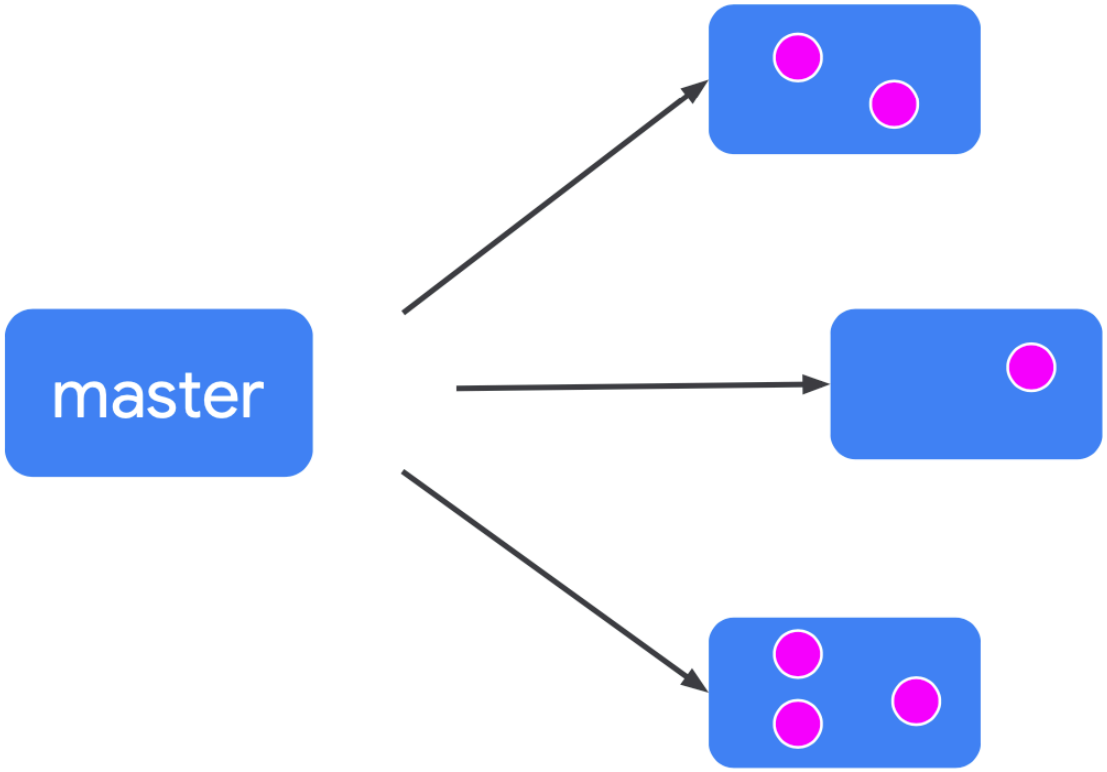
Kubernetes runs Applications in a Cluster.

Applications = Pods

Cluster



Pods in a Cluster



Deployment

```yaml
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

```yaml
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

```yaml
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

Linux | Windows | macOS

1 Download the installer for Windows

Download MicroK8s for Windows

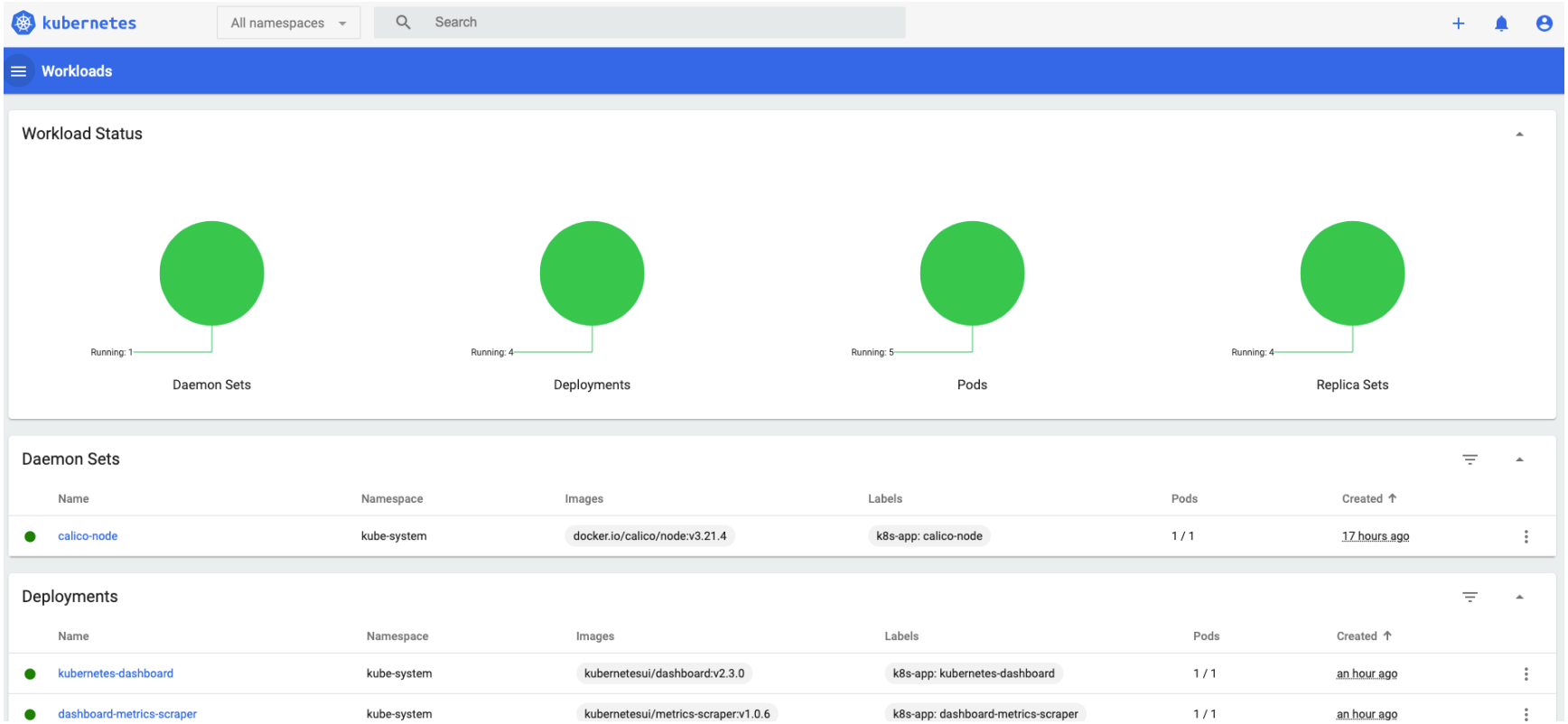
2 Run the Installer

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



Manage Kubernetes with

KUBECTL

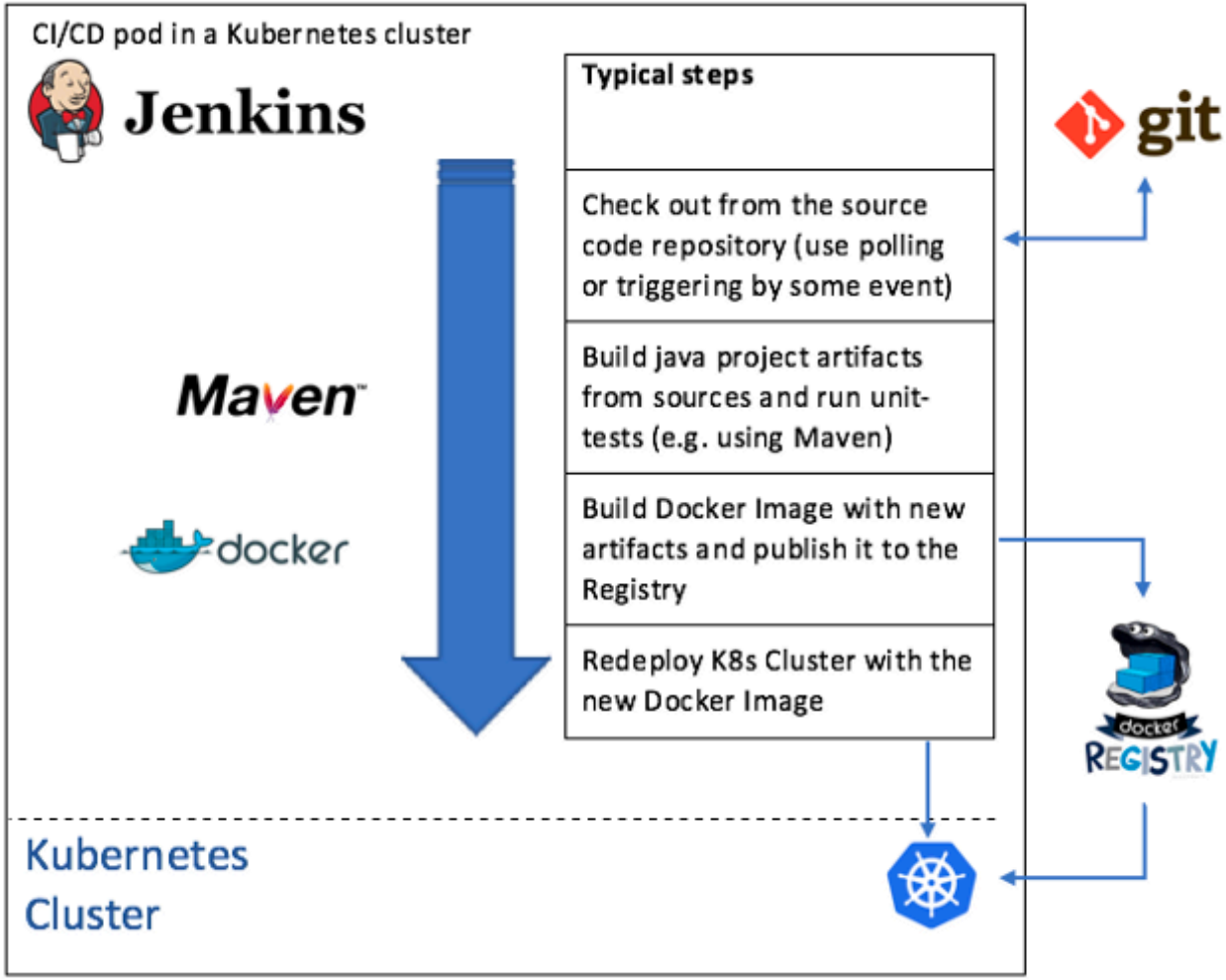
kubectl command list

- 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?