Research & Development Team

DevOps For Dev

www.pnpsw.com

sommai.k@gmail.com

081-754-4663

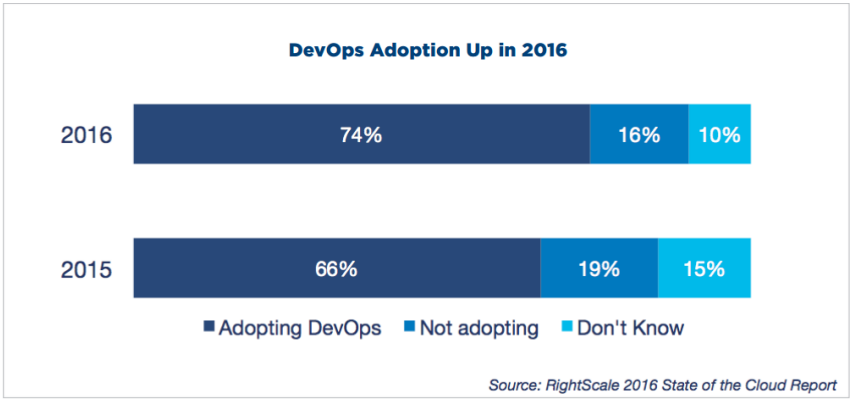
Line Id : sommaik

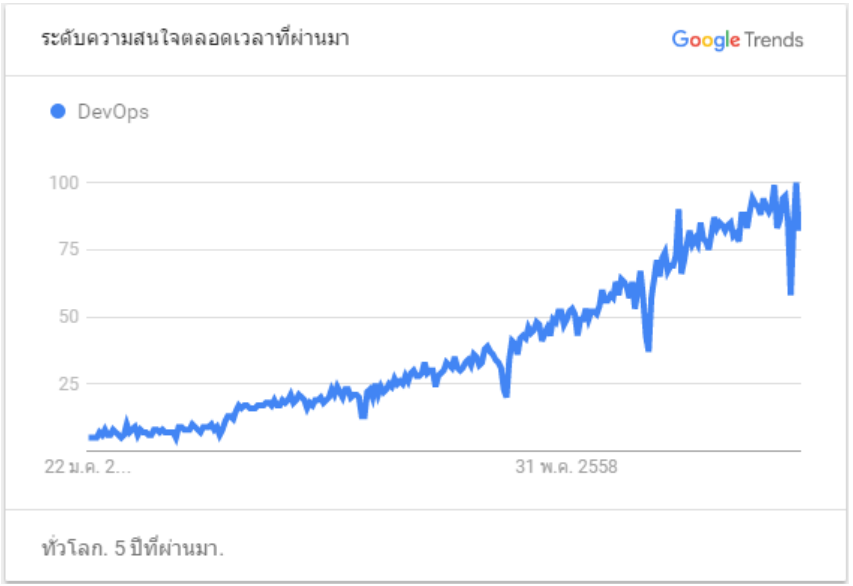
Definitions and History

- 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



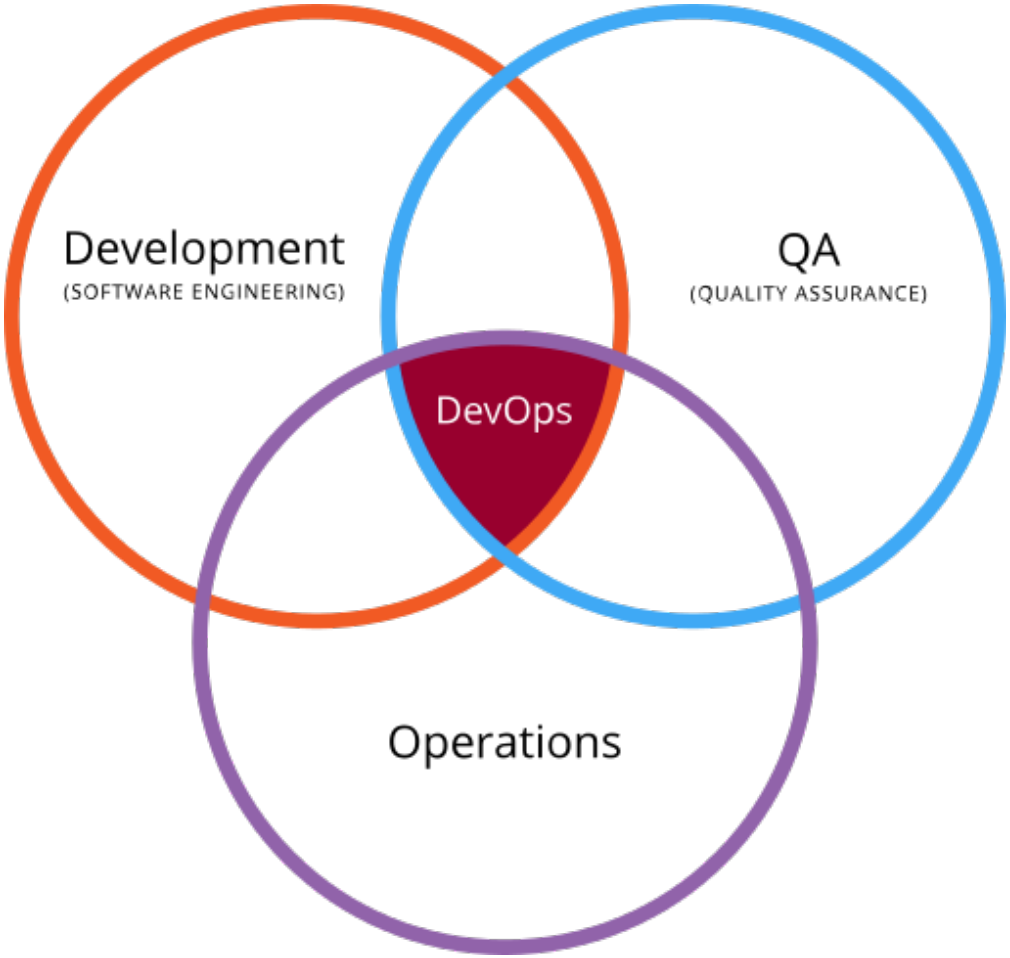


Cultural change

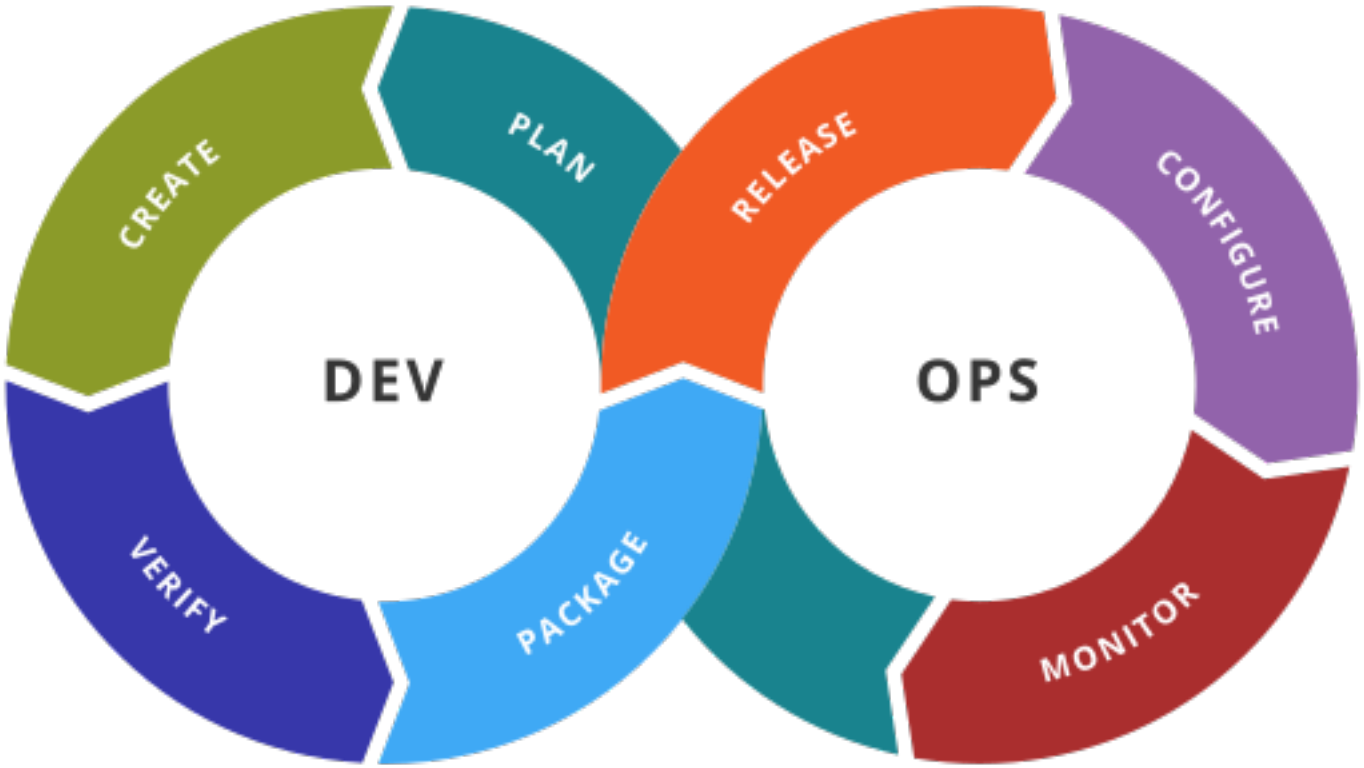
DevOps is more than just a tool or a process change.

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

Overview



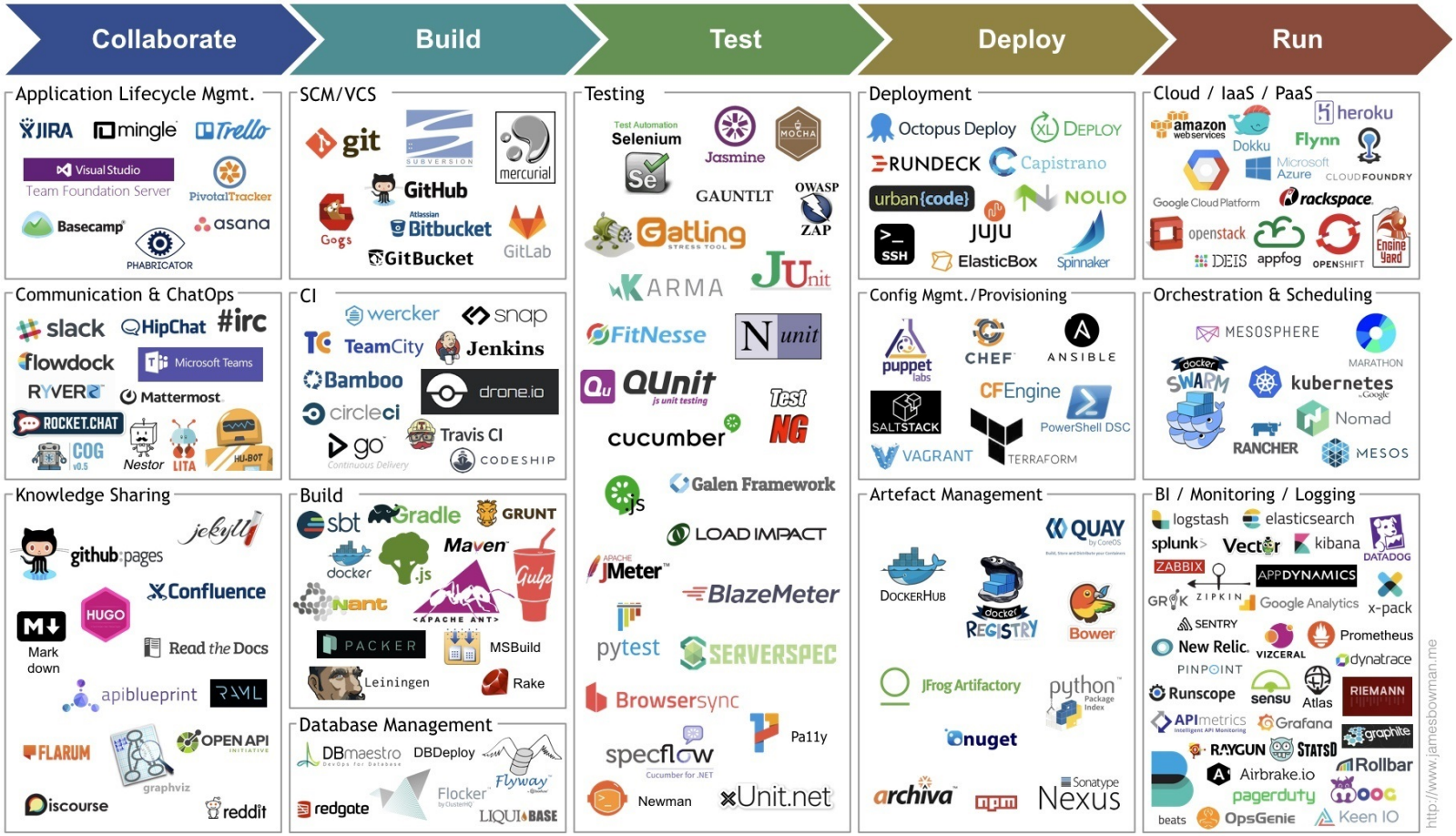
Workflow



7 Ways to Get Started with DevOps

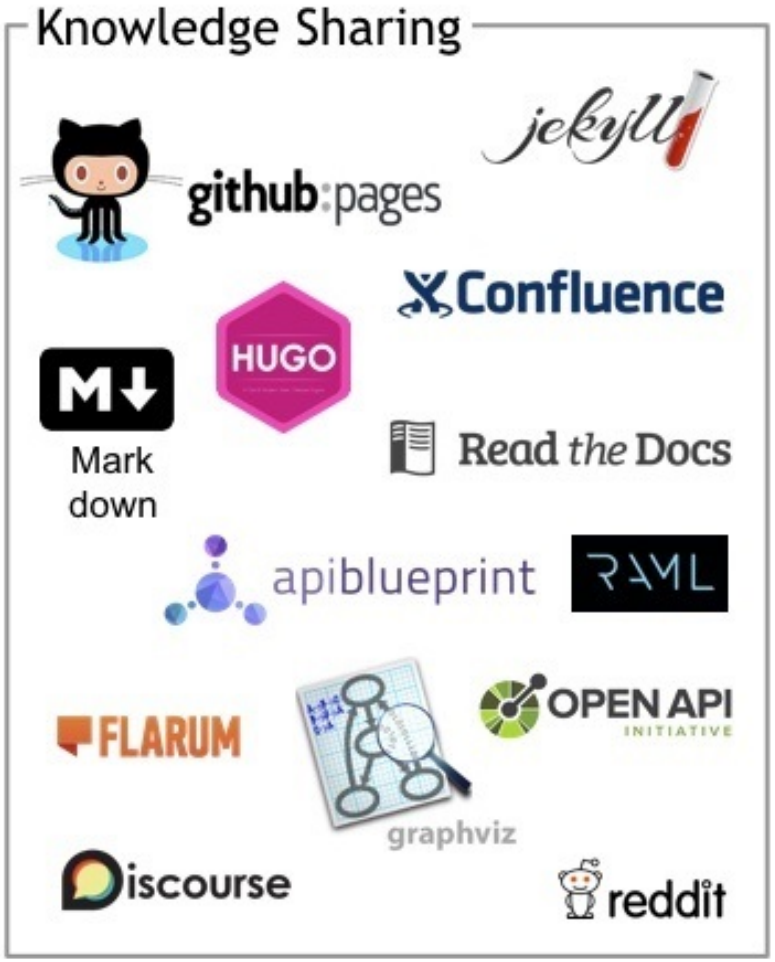
- Invite Your Operations Team Into Your Development Process
- Visualize the Work Together
- Automate Your Test/Build Process
- Create a Deployment Plan
- Identify Fragile Systems
- Smooth Out Wait States
- Link Your Work to Your Value

DevOps Ecosystem



Collaborate

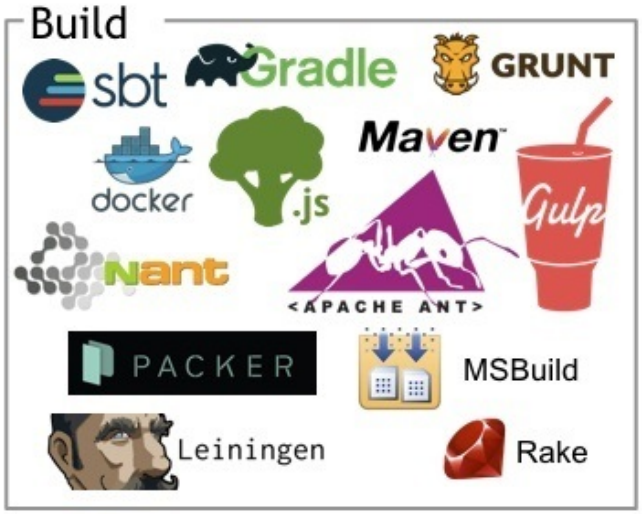
Collaborate



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
TestNG
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
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
Wooc
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

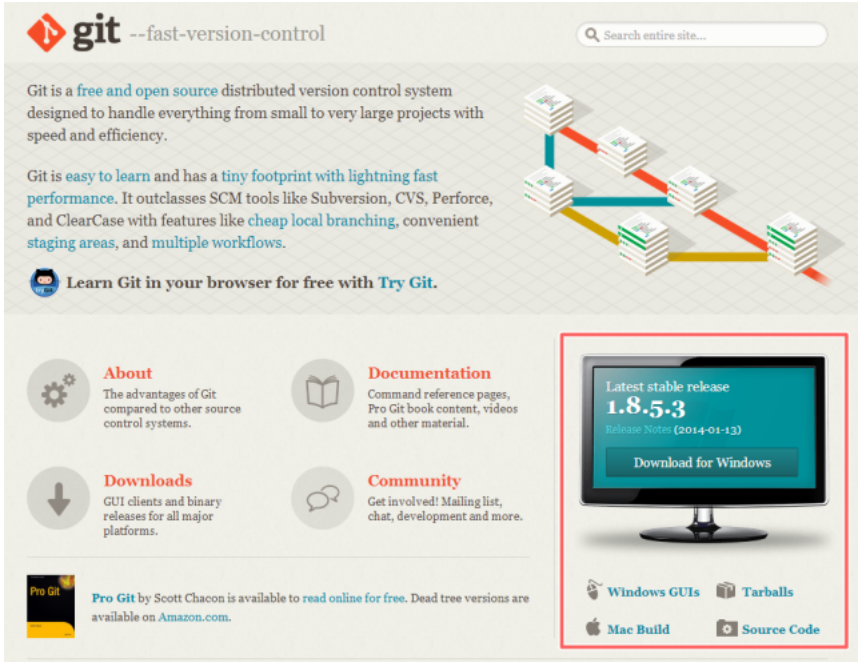
INSTALLATION

การติดตั้ง

GIT FOR WINDOWS

การติดตั้ง GIT สำหรับ Windows

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

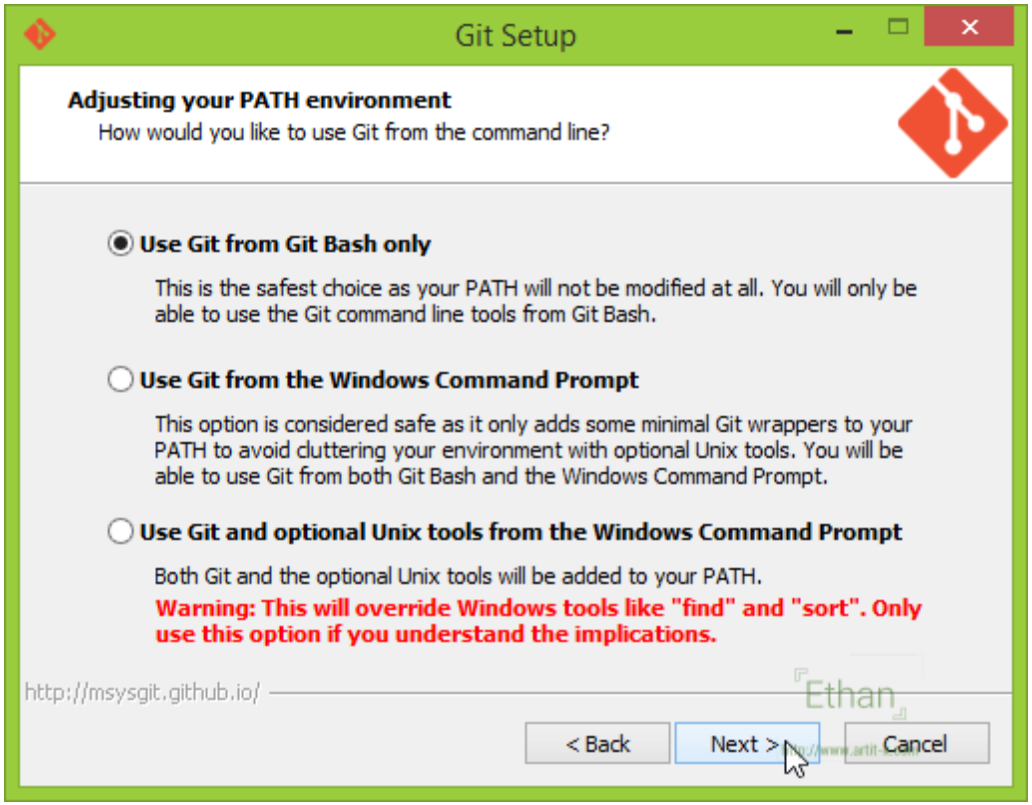


การติดตั้ง GIT สำหรับ Windows #2

- ติดตั้งโดยใช้สิทธิ Administrator ในการติดตั้ง

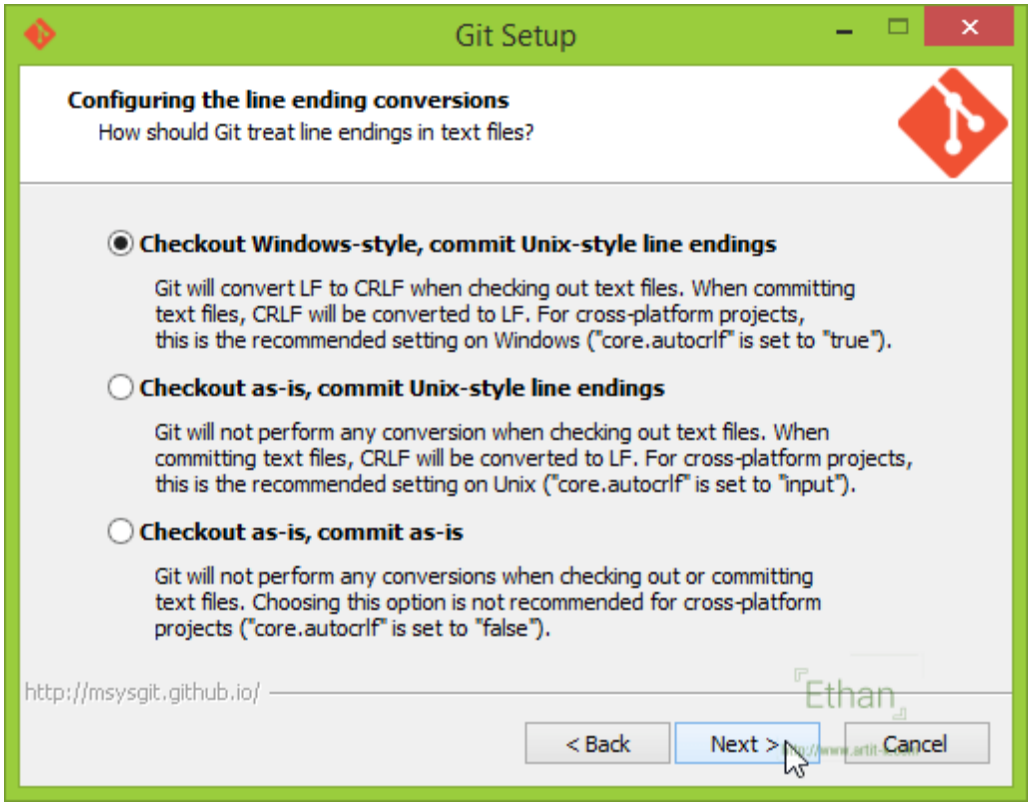
การติดตั้ง GIT สำหรับ Windows #3

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



การติดตั้ง GIT สำหรับ Windows #4

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

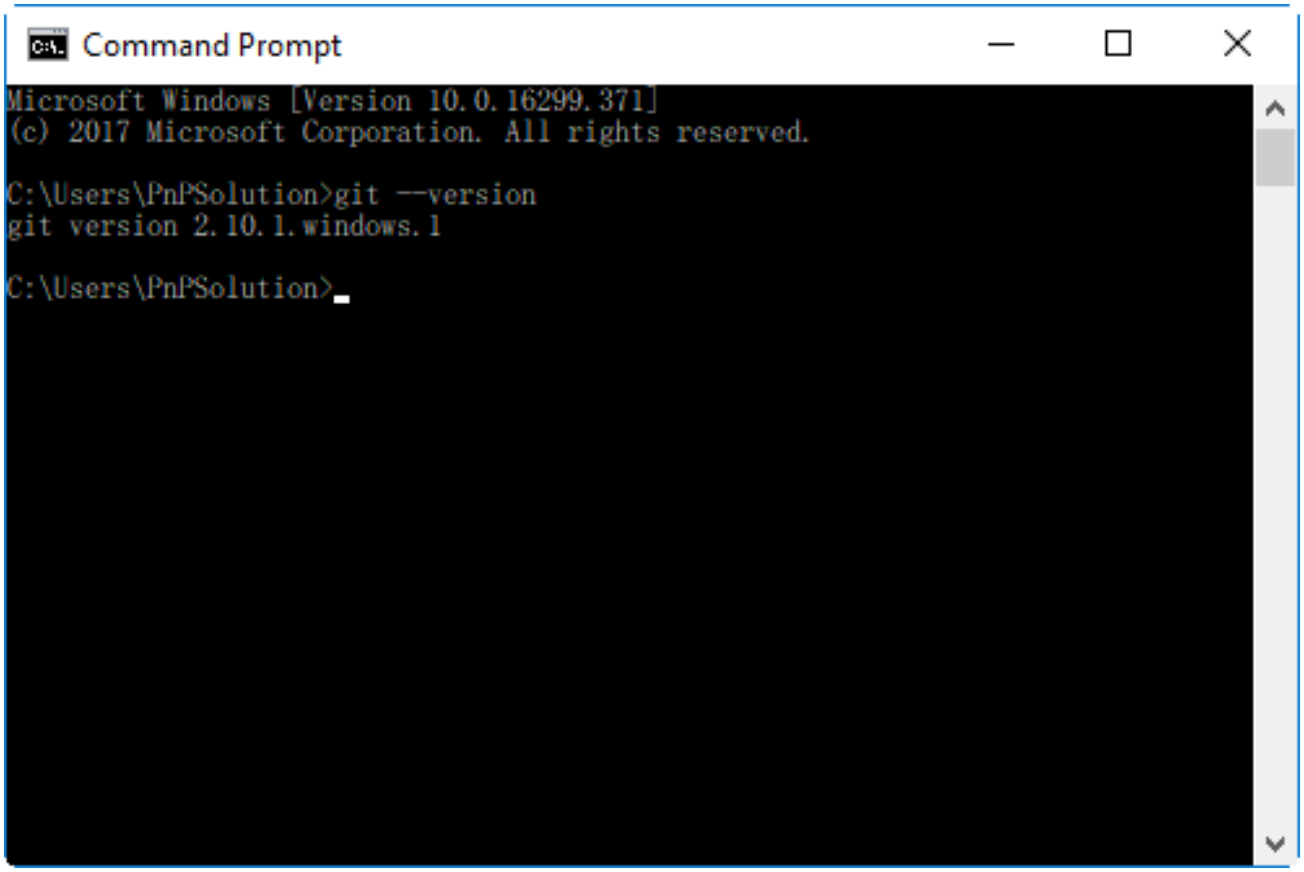


การติดตั้ง GIT สำหรับ Windows #5

- กดปุ่ม next ไปจนถึงหน้าสุดท้าย

ทดสอบหลังการติดตั้ง Git

- เปิดโปรแกรม cmd แล้วพิมพ์คำสั่ง git --version

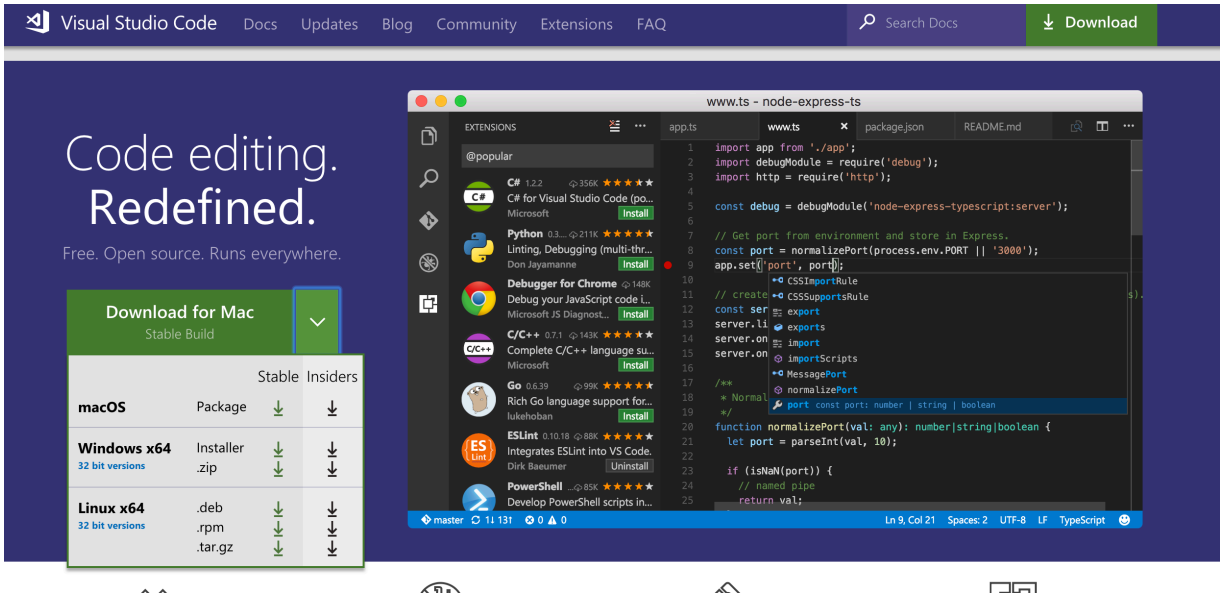


การติดตั้ง

VISUAL STUDIO CODE

การติดตั้ง VSC

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

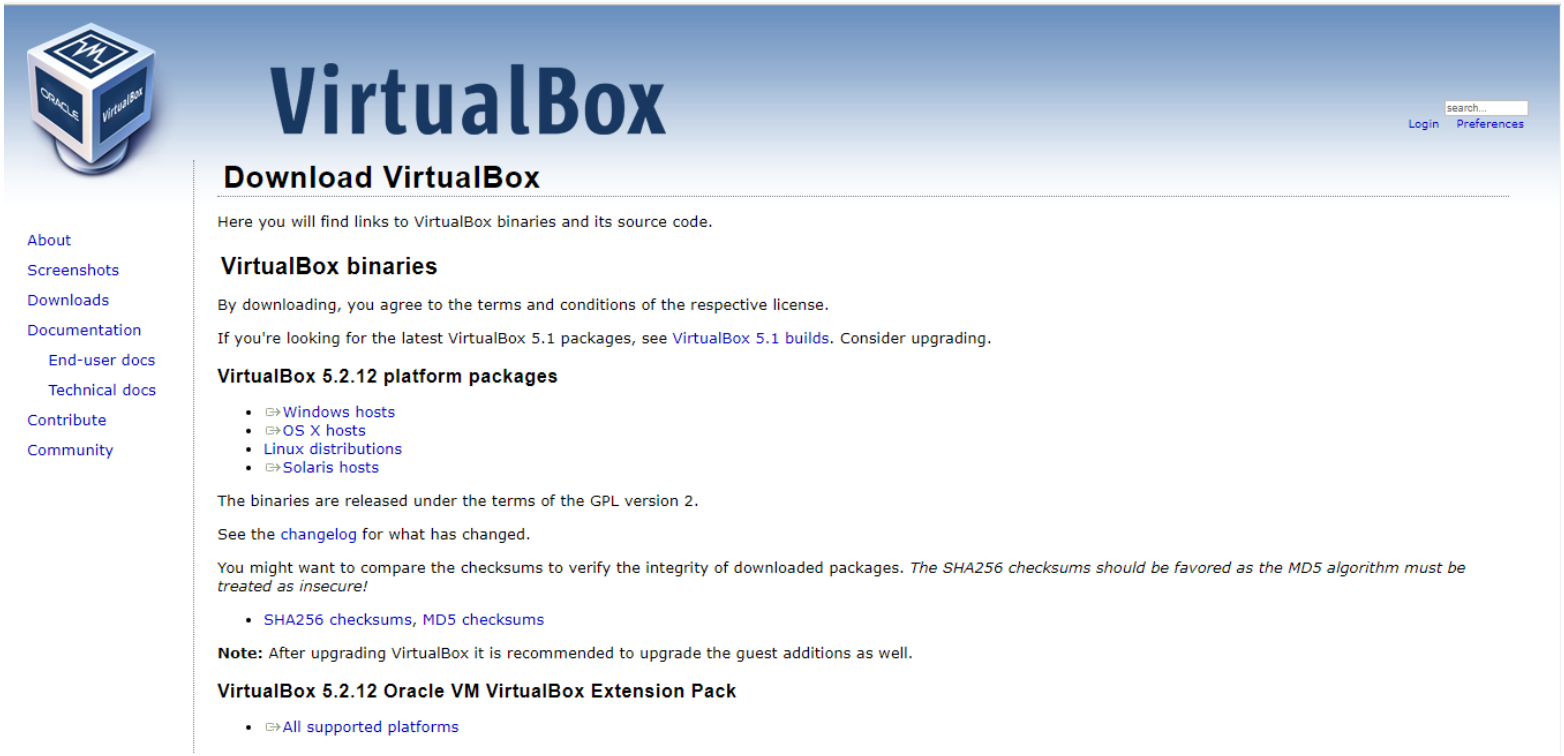


การติดตั้ง

VIRTUAL BOX

การติดตั้ง Virtual Box

- https://www.virtualbox.org/wiki/Downloads



VirtualBox

search...
Login Preferences

About
Screenshots
Downloads
Documentation
End-user docs
Technical docs
Contribute
Community

Download VirtualBox

Here you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

If you're looking for the latest VirtualBox 5.1 packages, see VirtualBox 5.1 builds. Consider upgrading.

VirtualBox 5.2.12 platform packages

- Windows hosts
- OS X hosts
- Linux distributions
- Solaris hosts

The binaries are released under the terms of the GPL version 2.

See the changelog for what has changed.

You might want to compare the checksums to verify the integrity of downloaded packages. *The SHA256 checksums should be favored as the MD5 algorithm must be treated as insecure!*

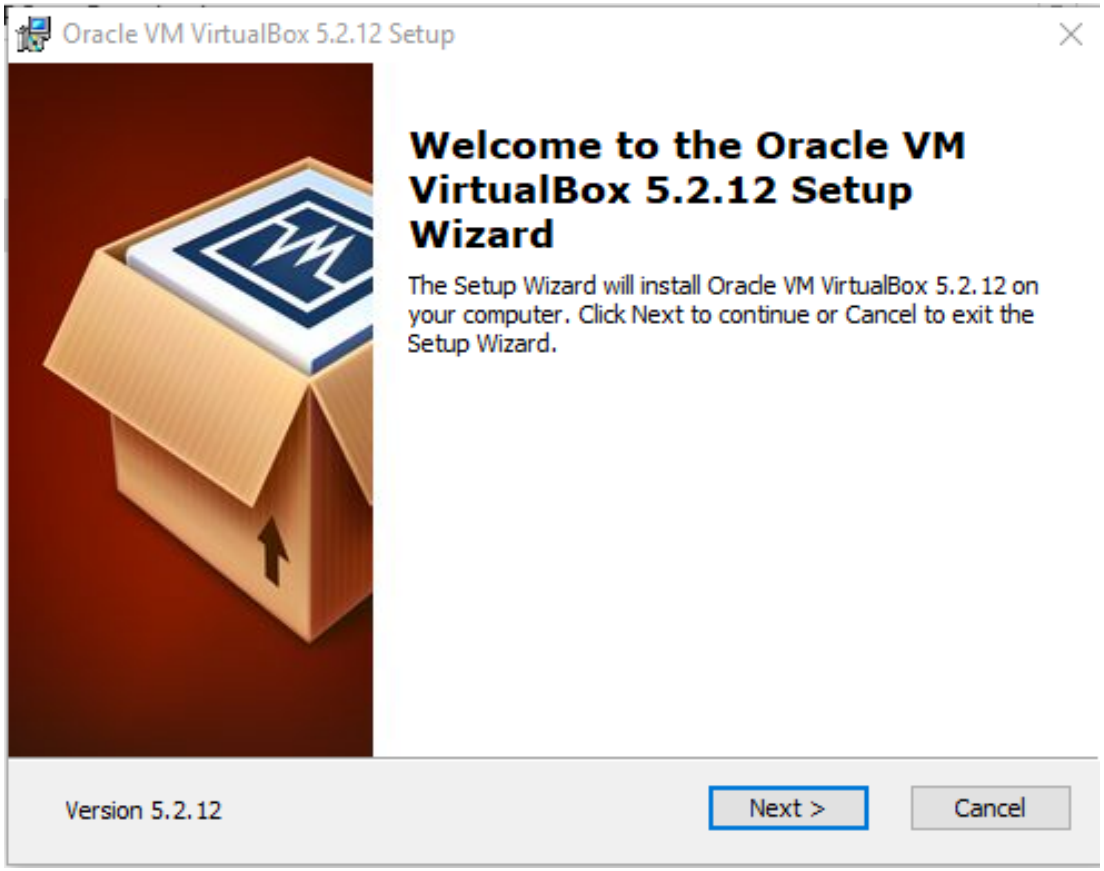
- SHA256 checksums, MD5 checksums

Note: After upgrading VirtualBox it is recommended to upgrade the guest additions as well.

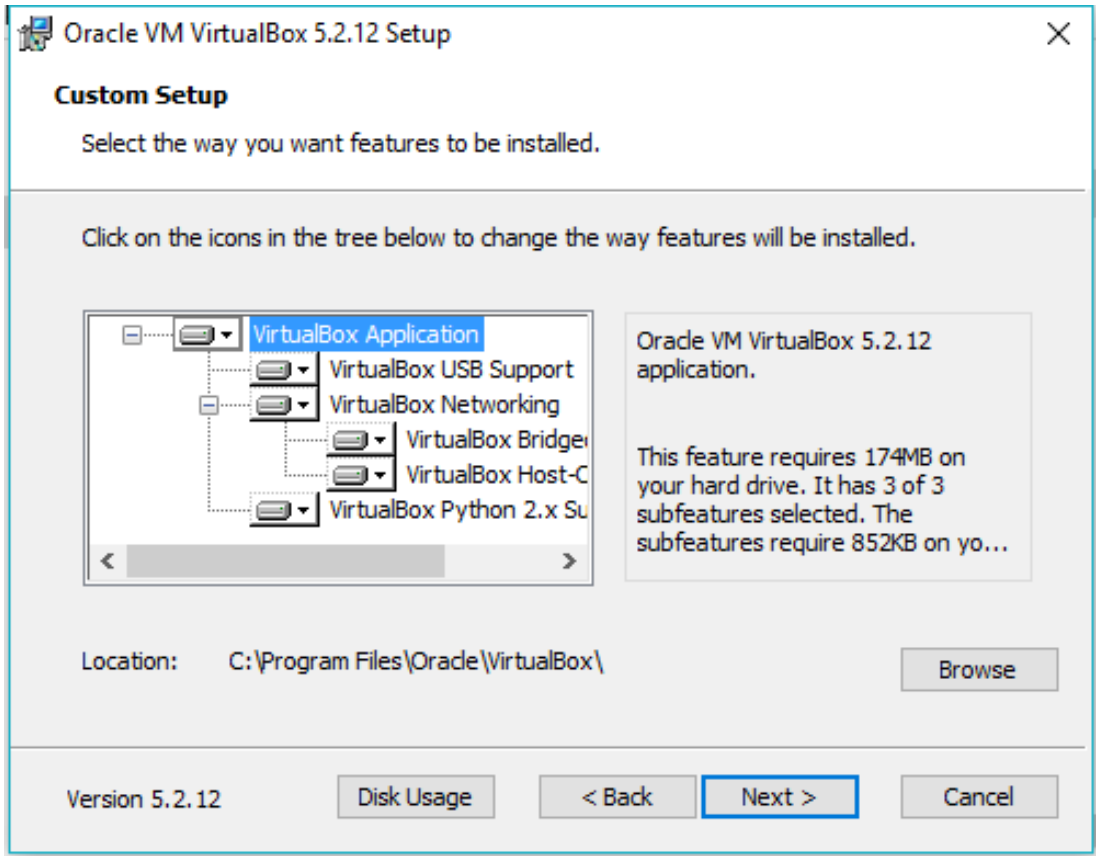
VirtualBox 5.2.12 Oracle VM VirtualBox Extension Pack

- All supported platforms

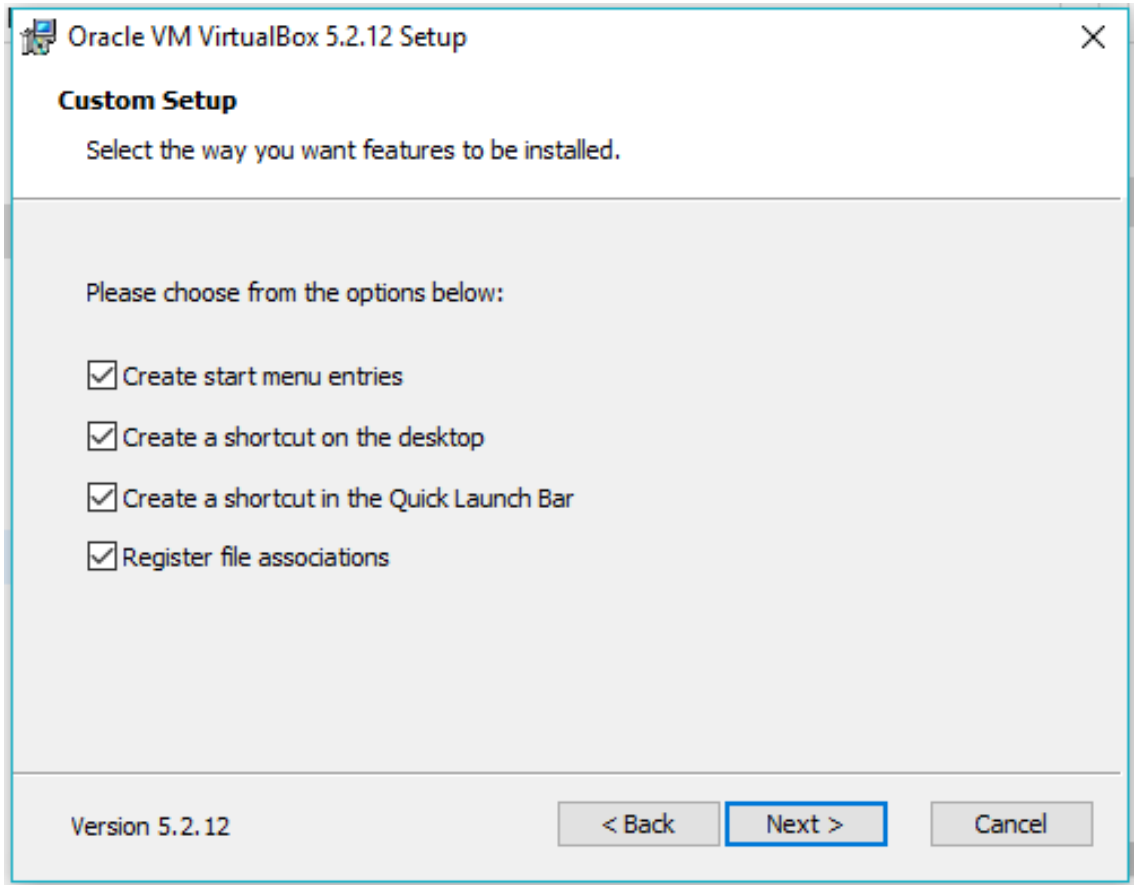
การติดตั้ง Virtual Box



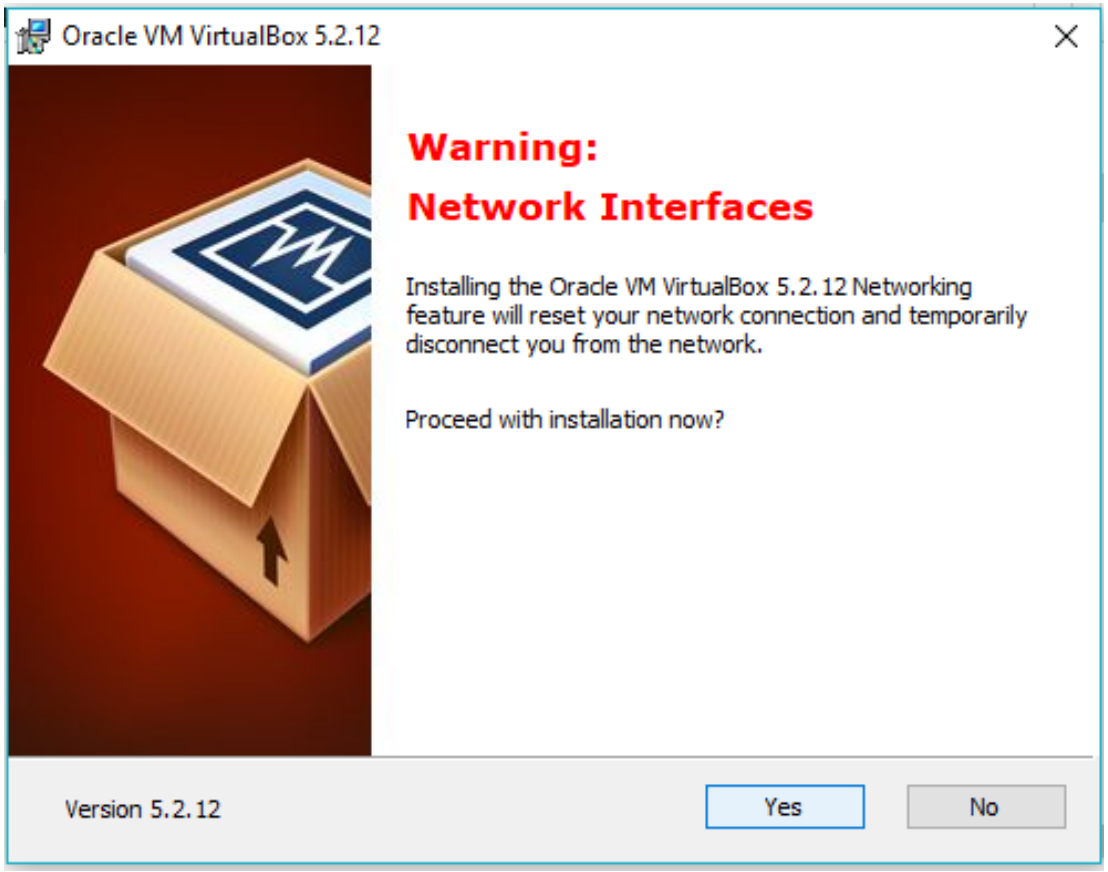
การติดตั้ง Virtual Box



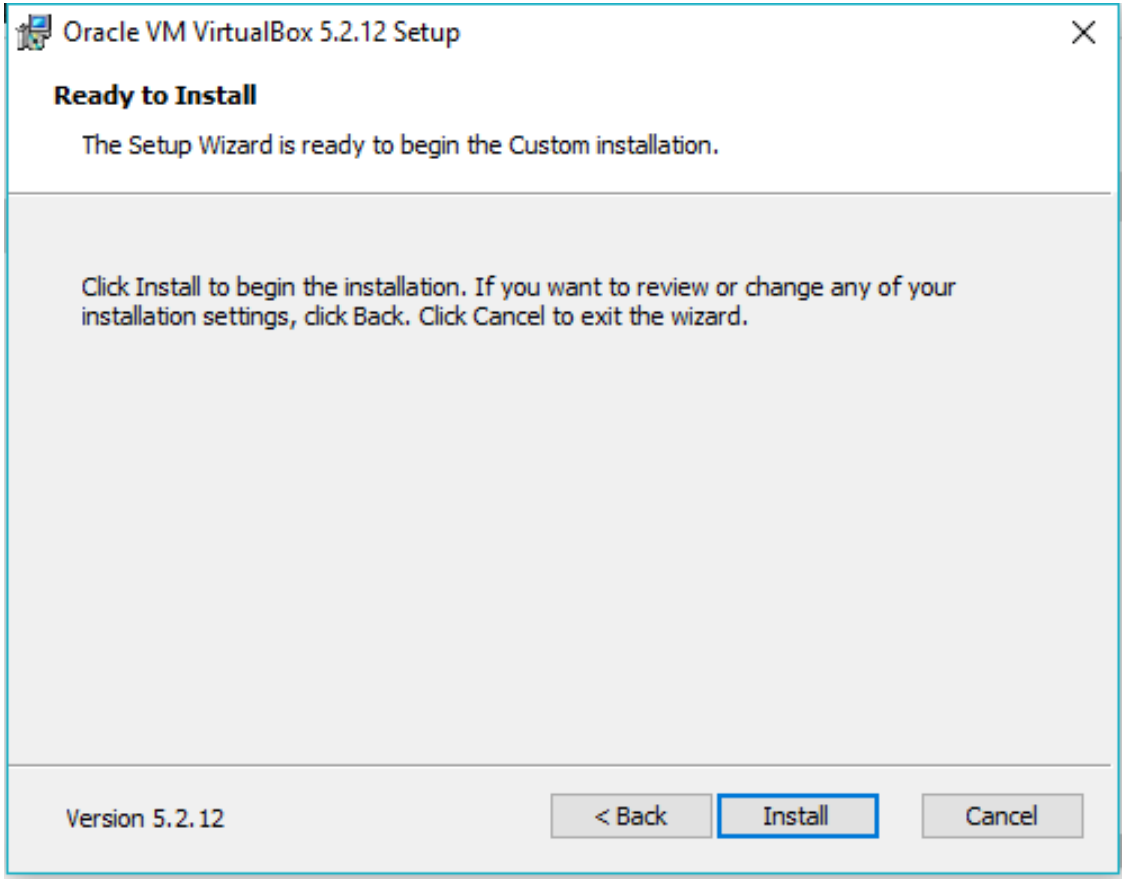
การติดตั้ง Virtual Box



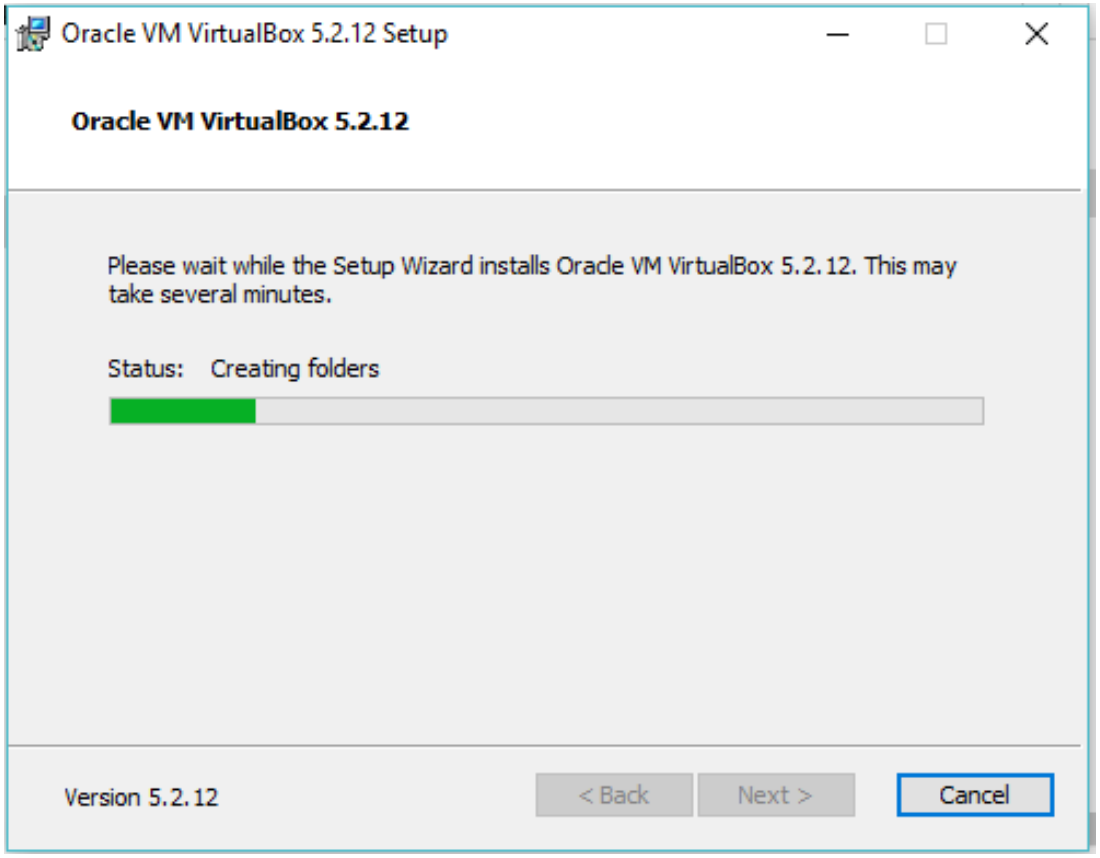
การติดตั้ง Virtual Box



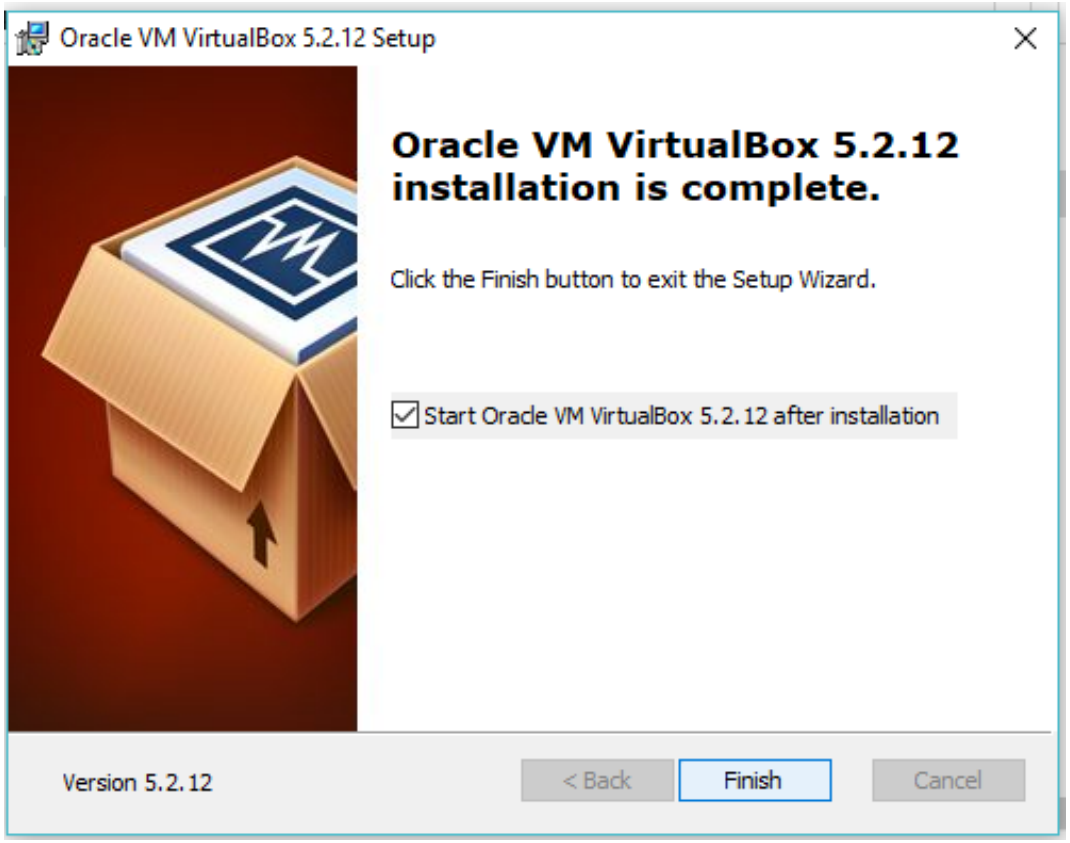
การติดตั้ง Virtual Box



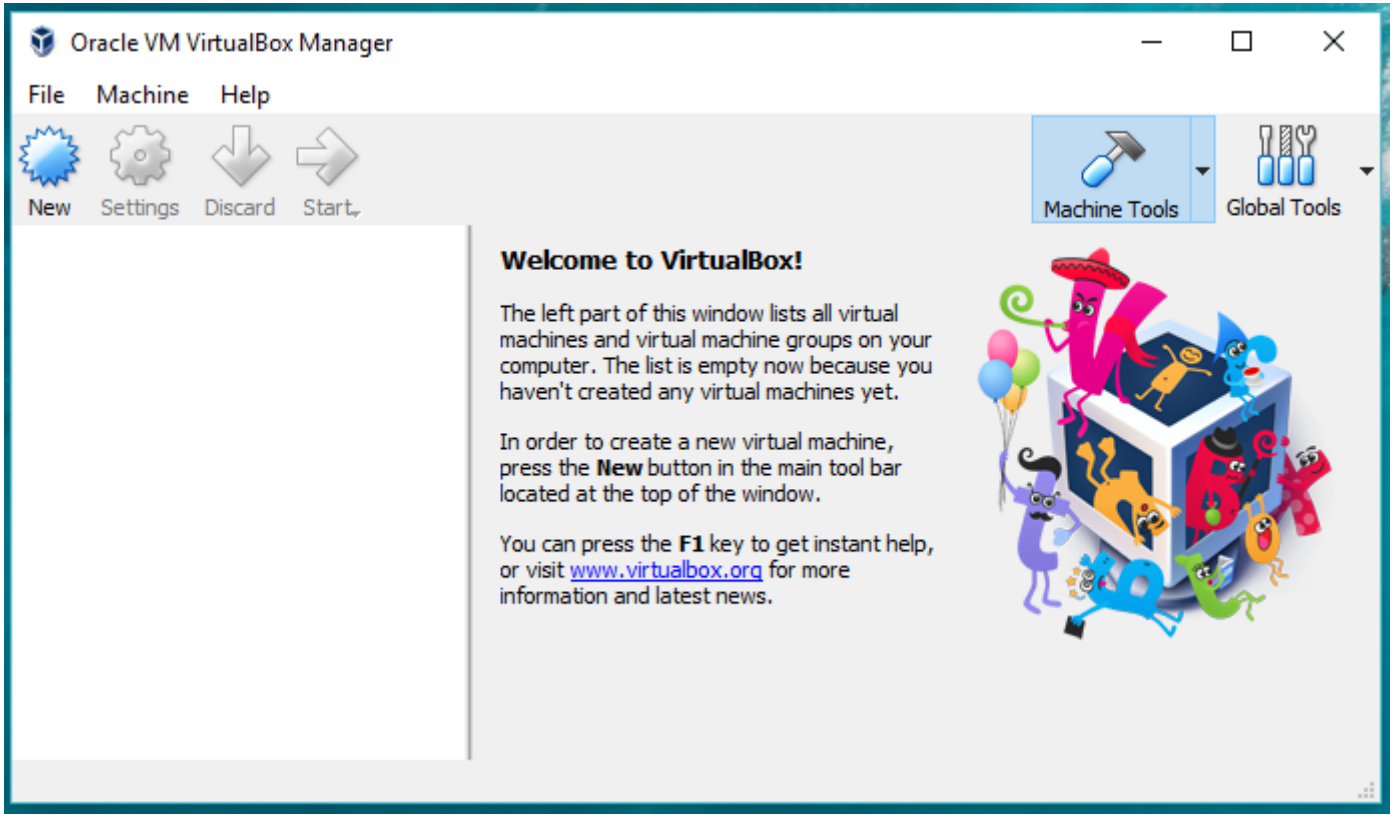
การติดตั้ง Virtual Box



การติดตั้ง Virtual Box



การติดตั้ง Virtual Box

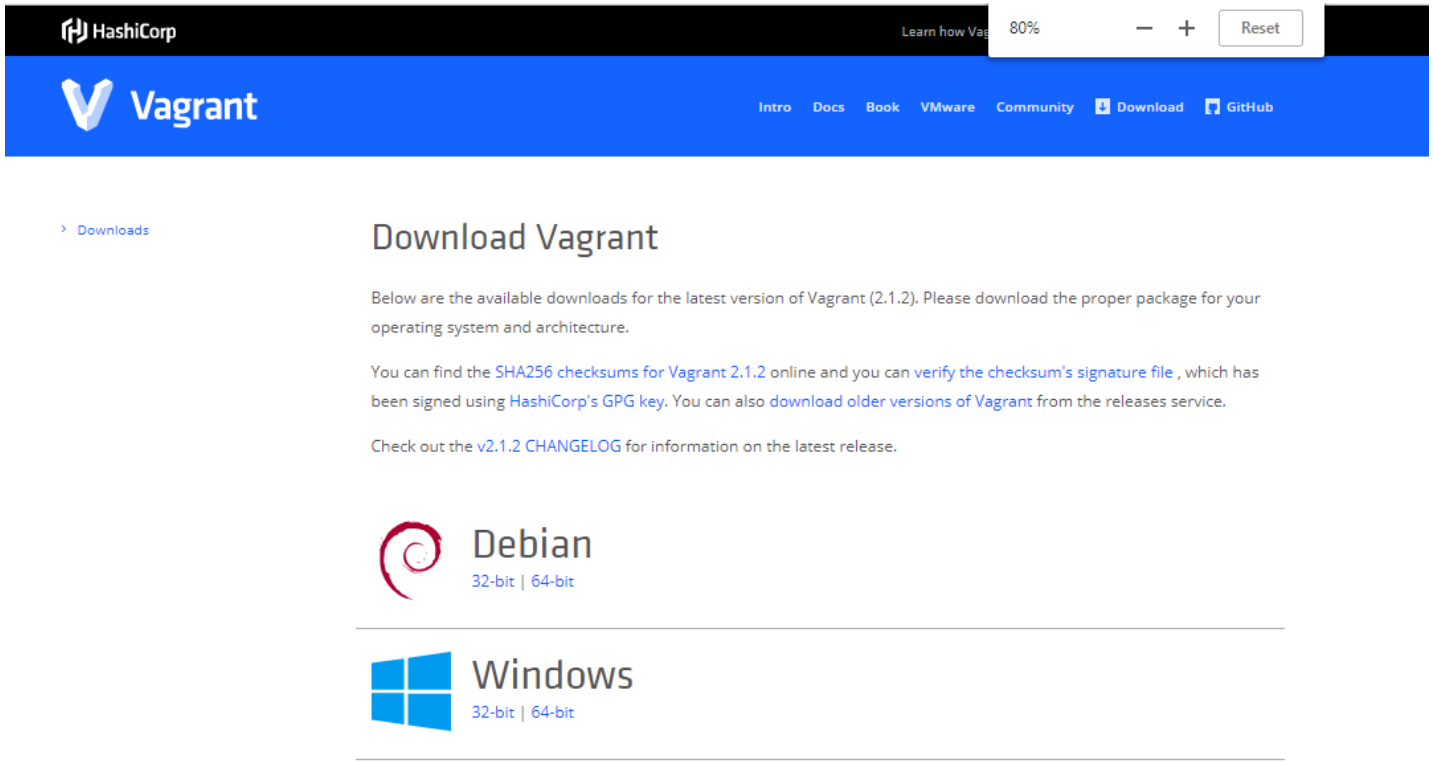


การติดตั้ง

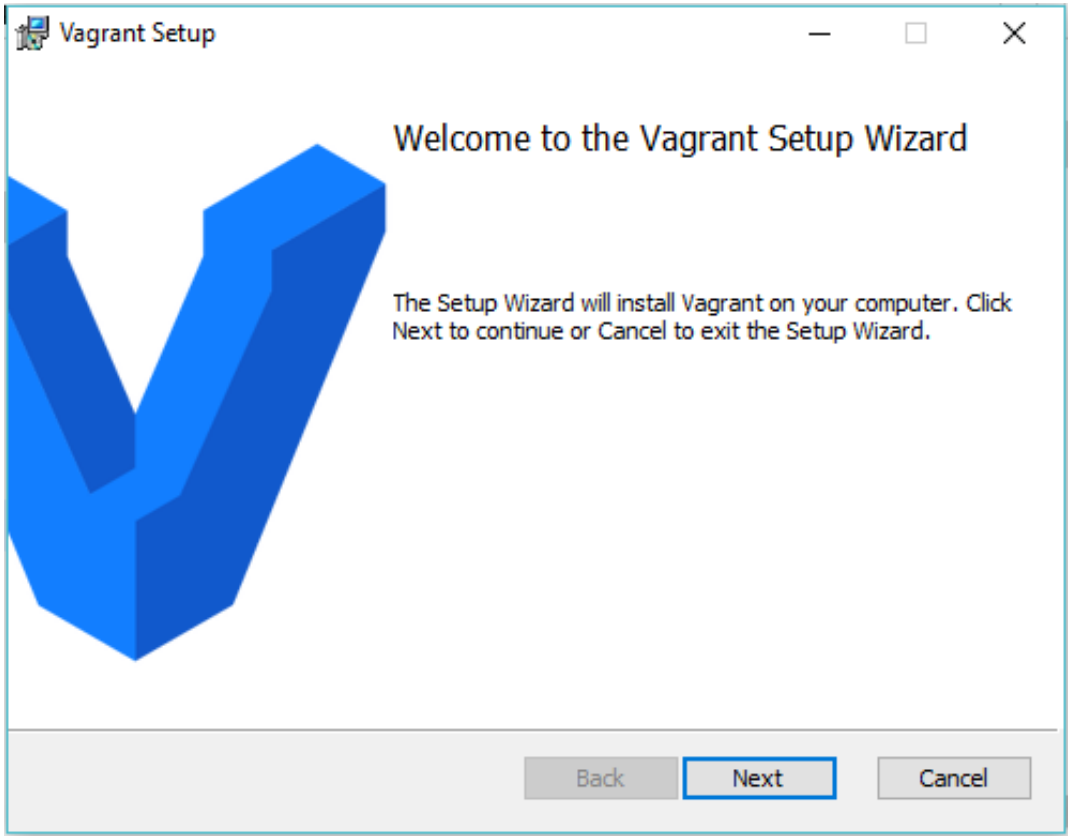
VAGRANT

การติดตั้ง Vagrant

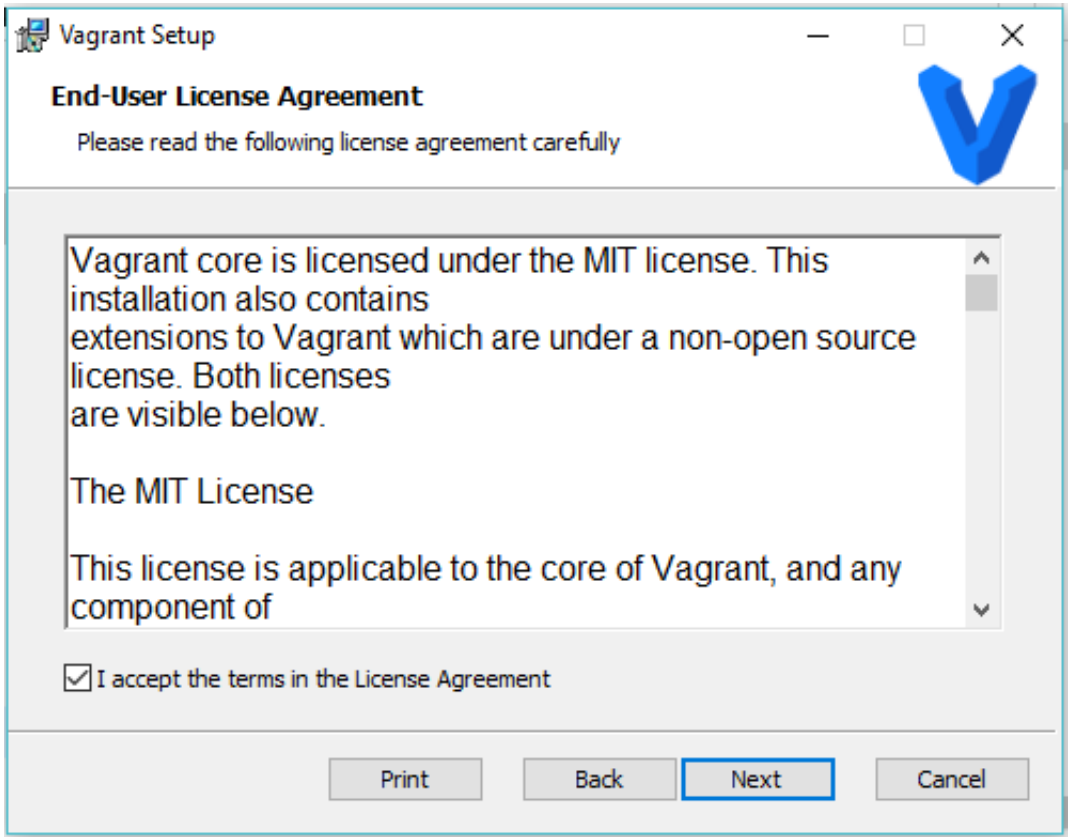
- https://www.vagrantup.com/downloads.html



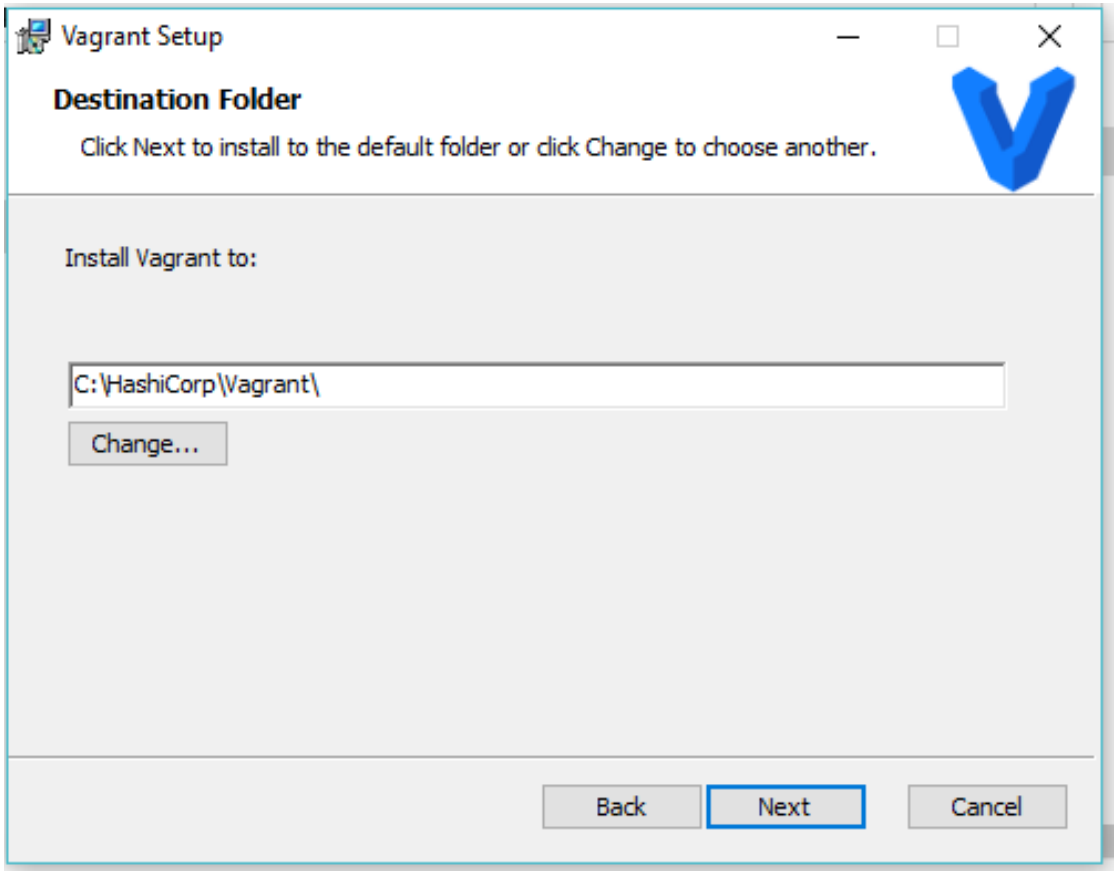
การติดตั้ง Vagrant



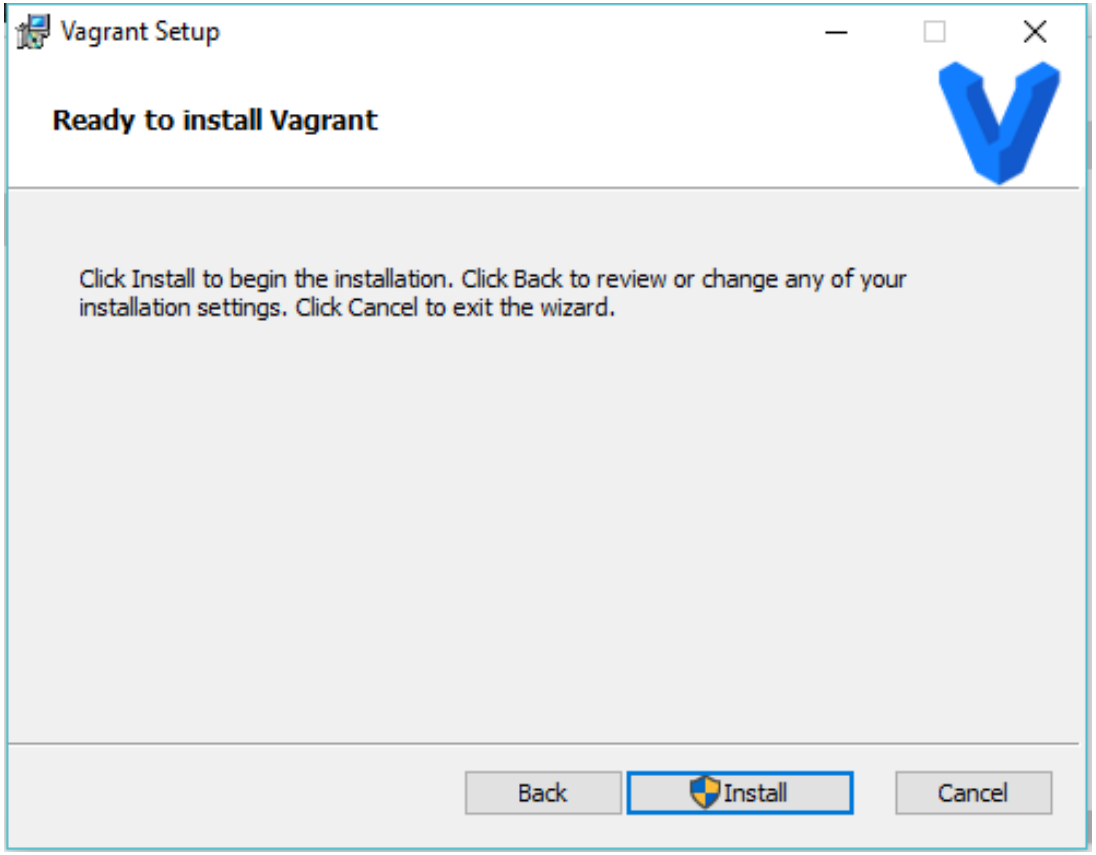
การติดตั้ง Vagrant



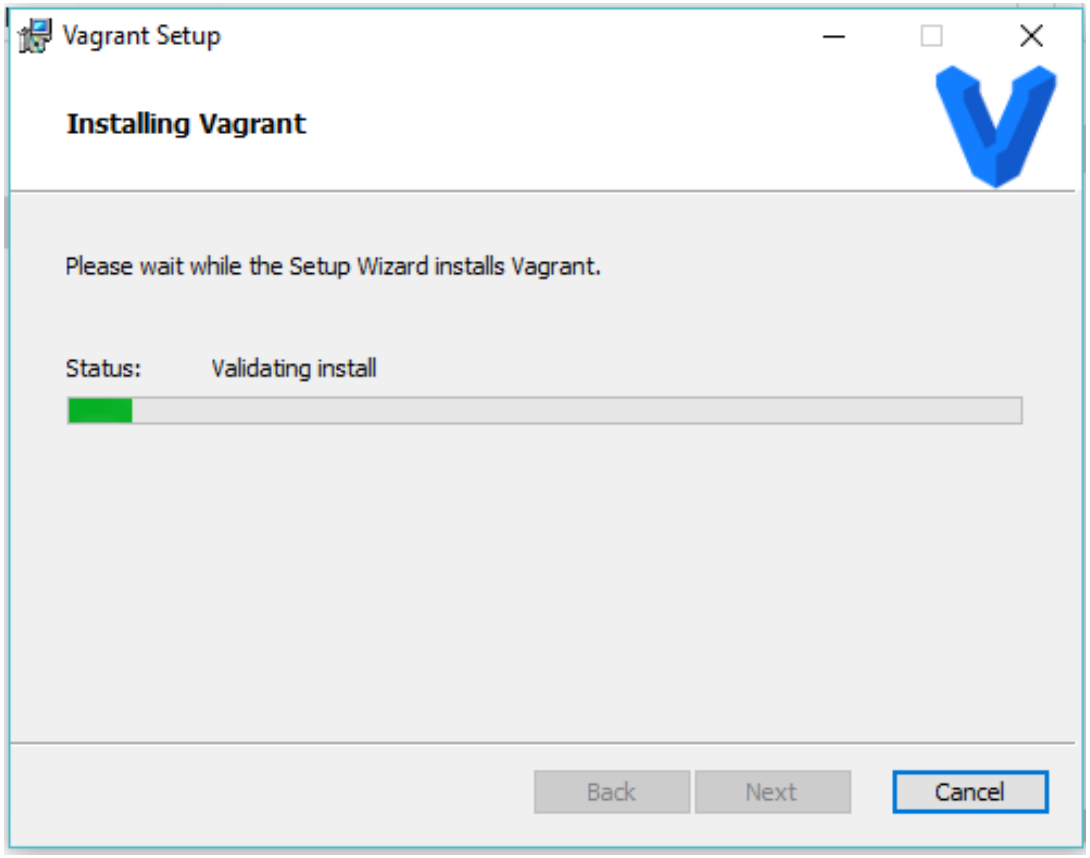
การติดตั้ง Vagrant



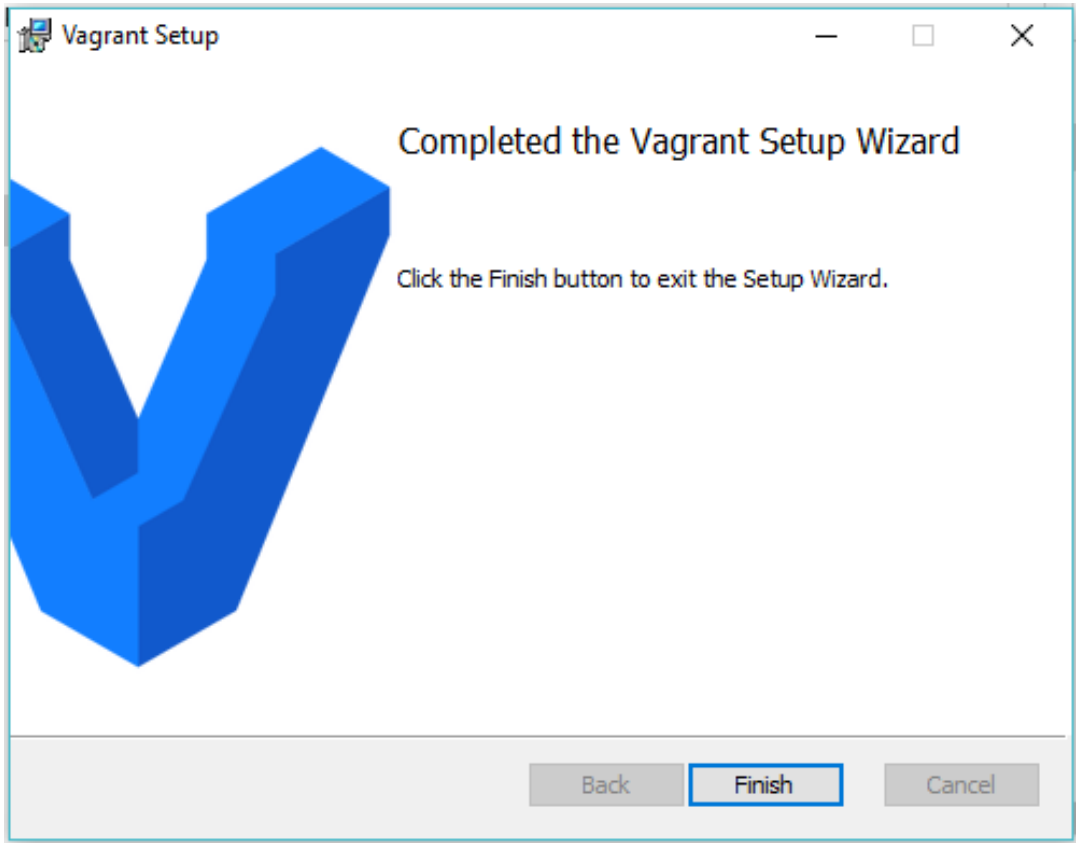
การติดตั้ง Vagrant



การติดตั้ง Vagrant



การติดตั้ง Vagrant

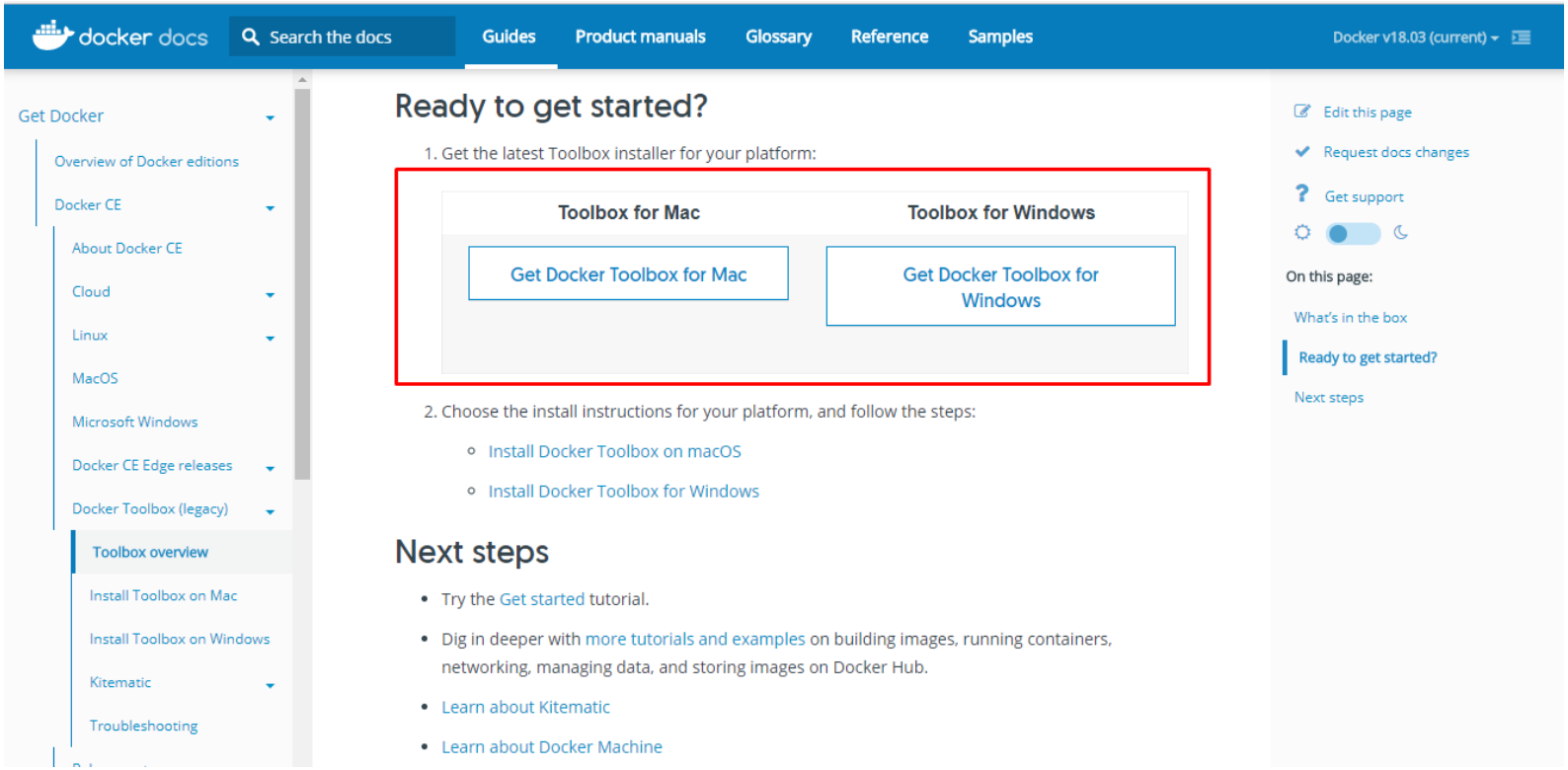


การติดตั้ง

DOCKER TOOLBOX

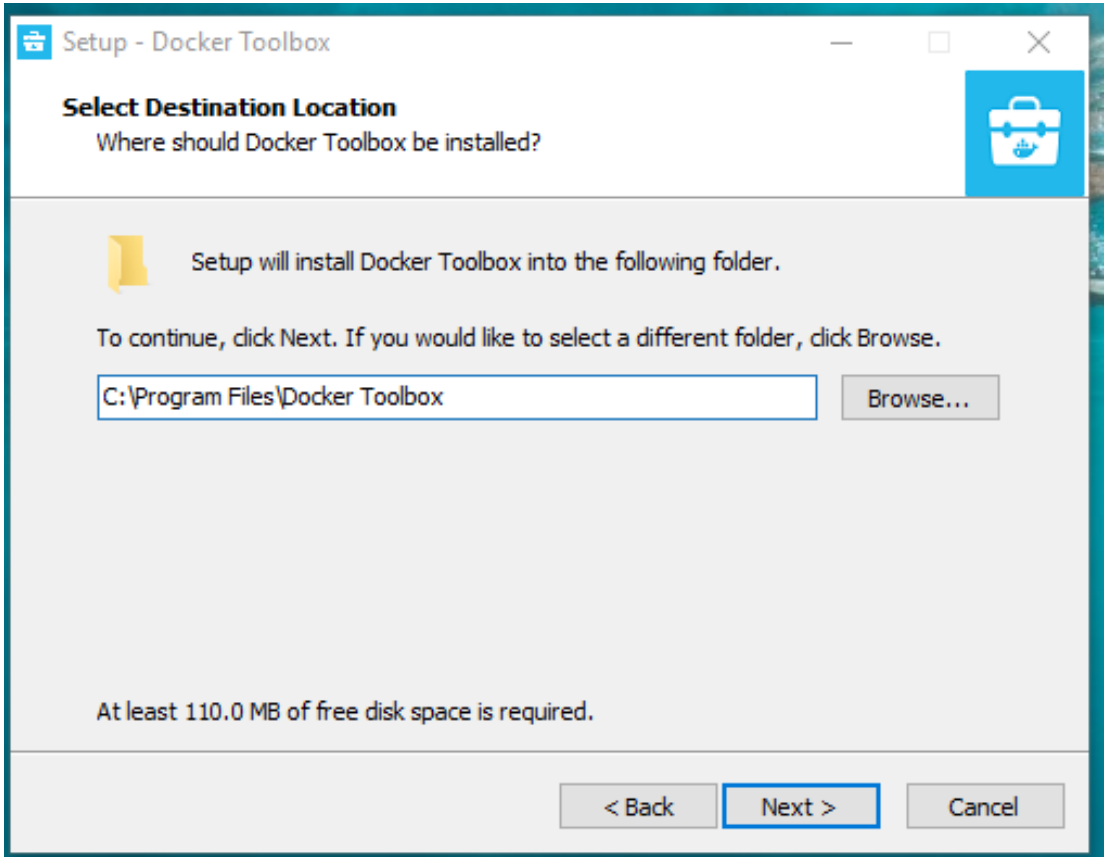
การติดตั้ง Docker Toolbox

- https://docs.docker.com/toolbox/overview/#ready-to-get-started



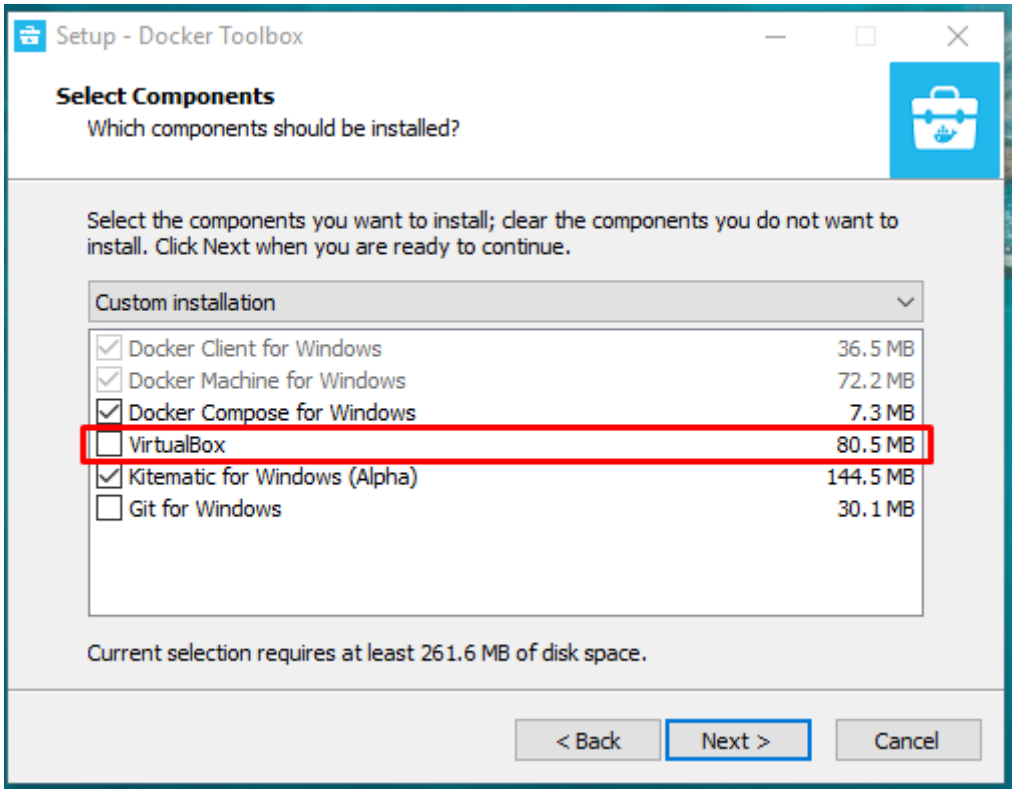
การติดตั้ง Docker Toolbox

การติดตั้ง Docker Toolbox

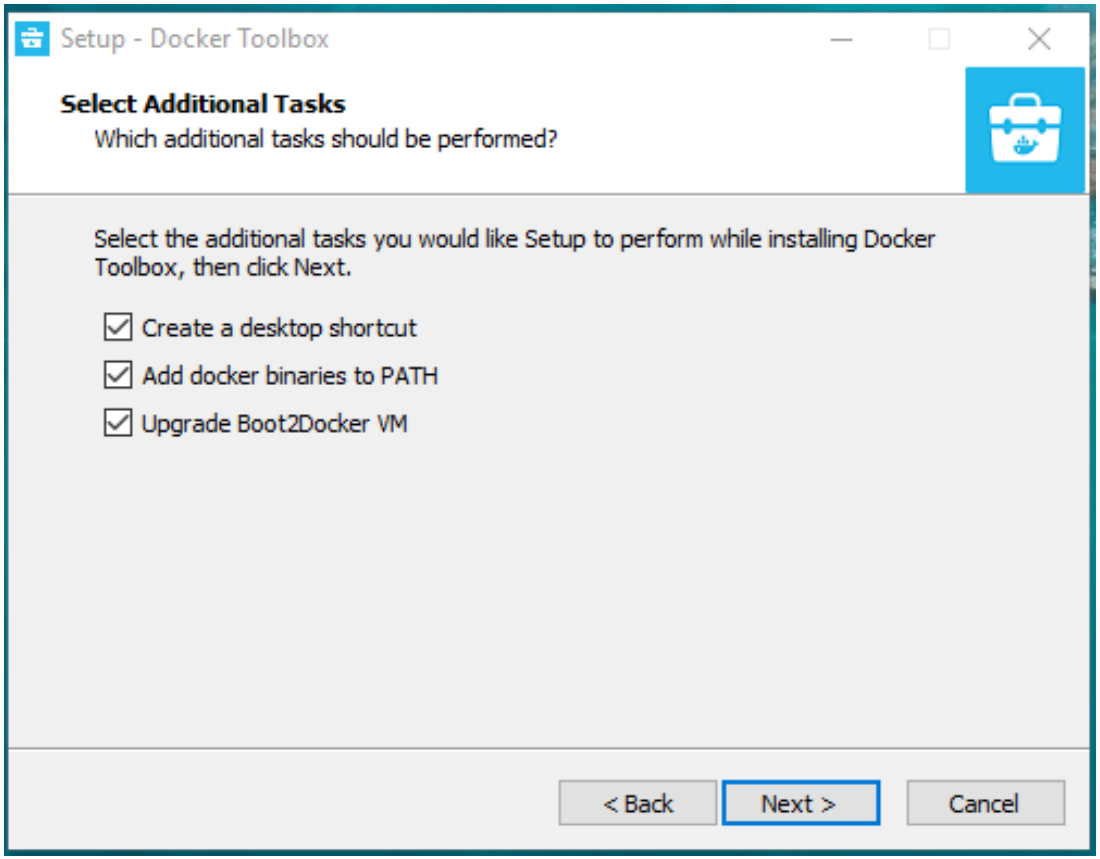


การติดตั้ง Docker Toolbox

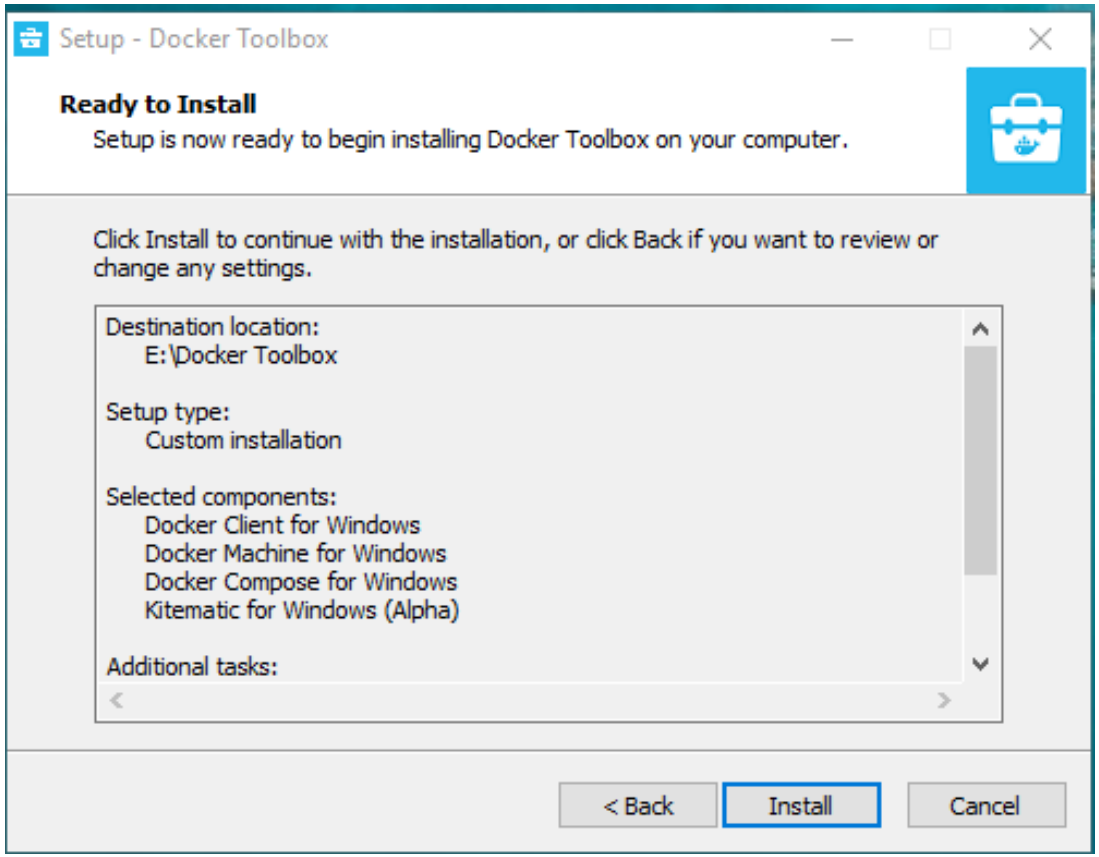
- เอาเครื่องหมายถูกหน้า VirtualBox ออก



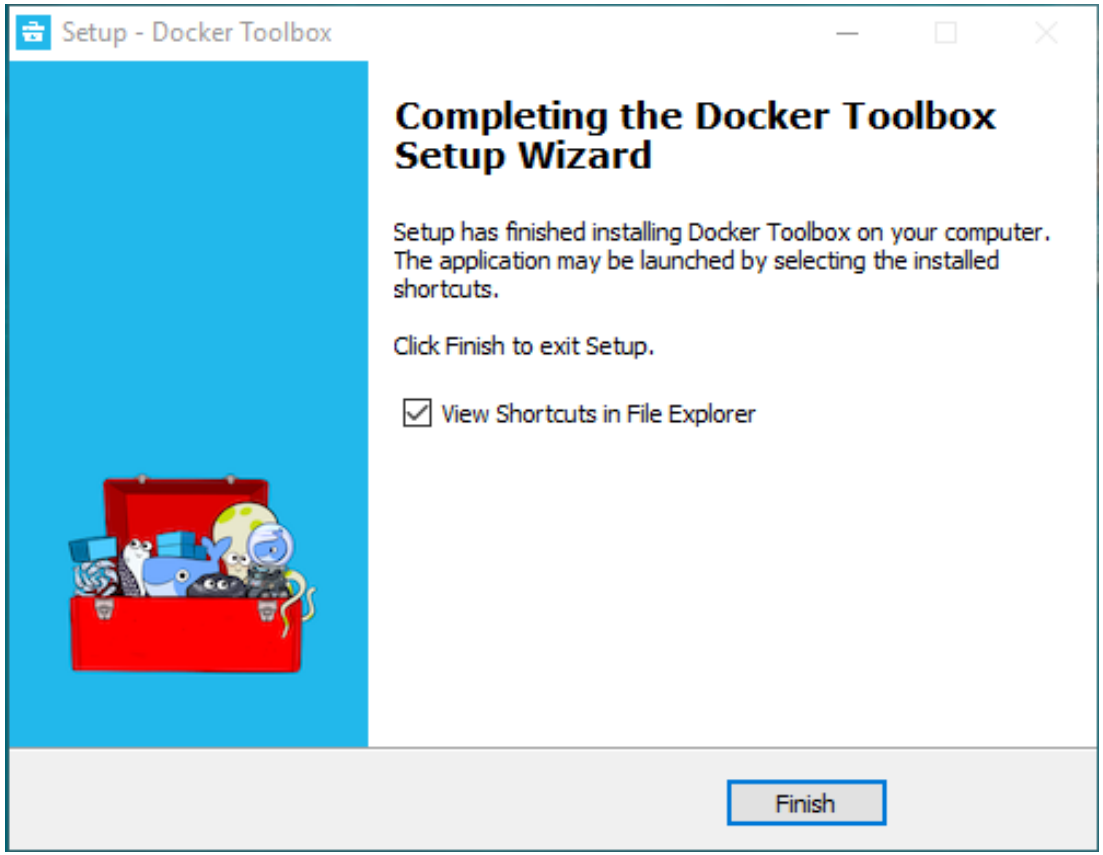
การติดตั้ง Docker Toolbox



การติดตั้ง Docker Toolbox



การติดตั้ง Docker Toolbox



ติดตั้ง

CHOCOLATEY

`https://chocolatey.org`

ติดตั้ง

MINIKUBE, KUBECTL

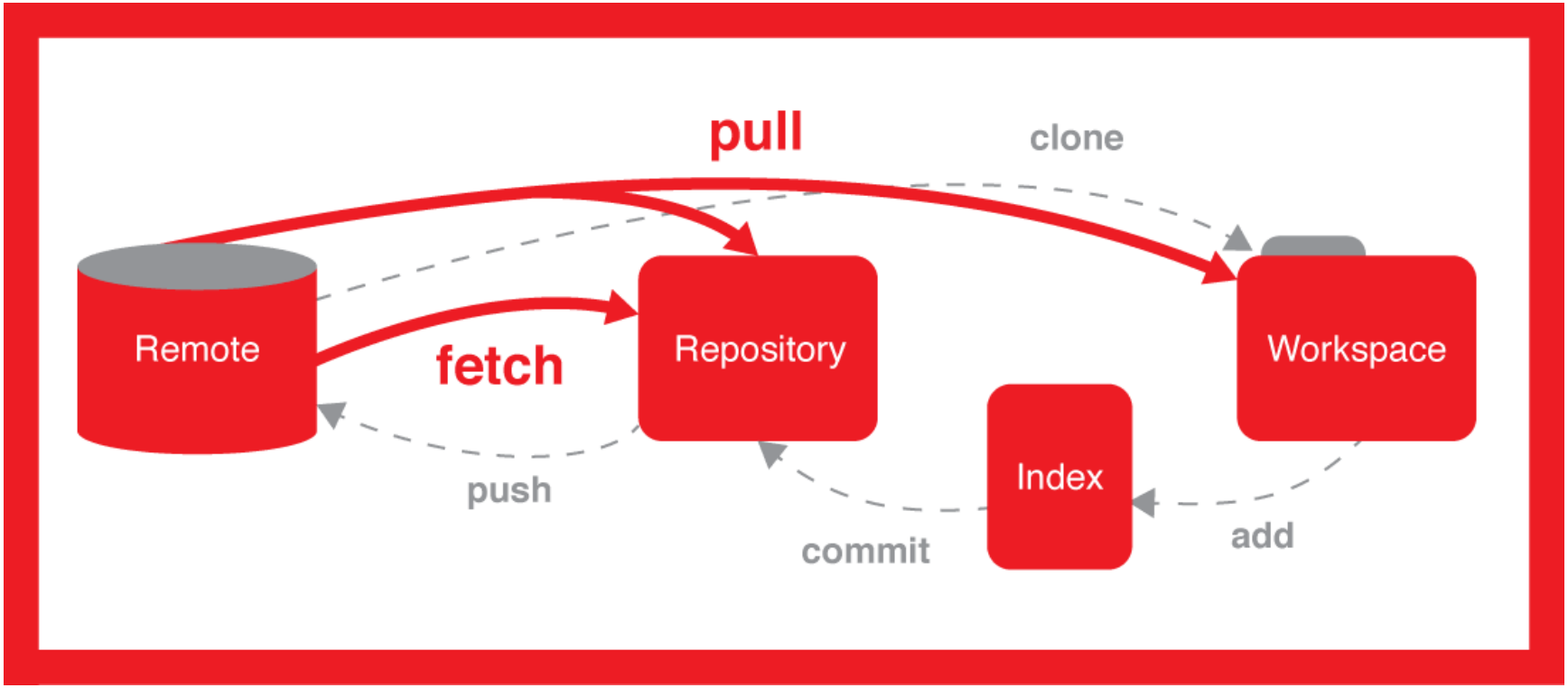
Install command

```
choco install minikube

choco install kubernetes-cli
```

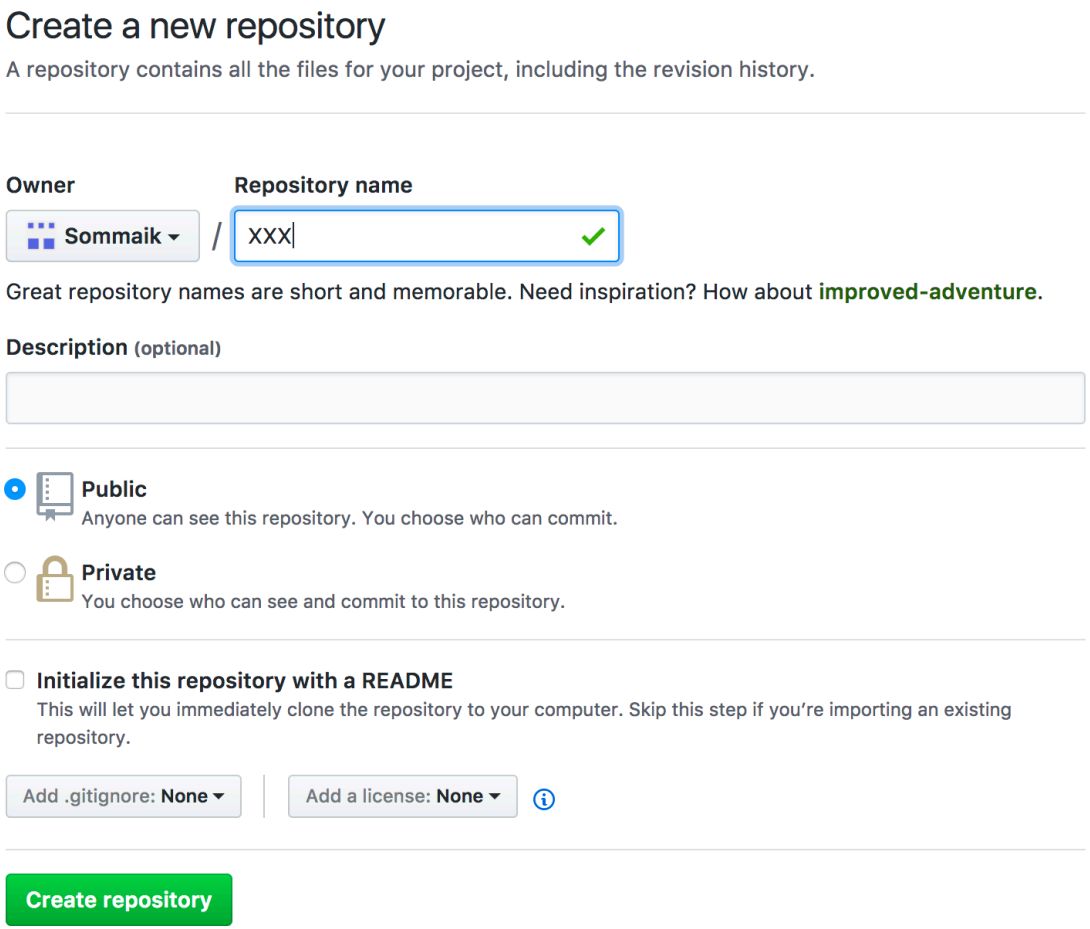
Software version control with

GIT

Overview



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

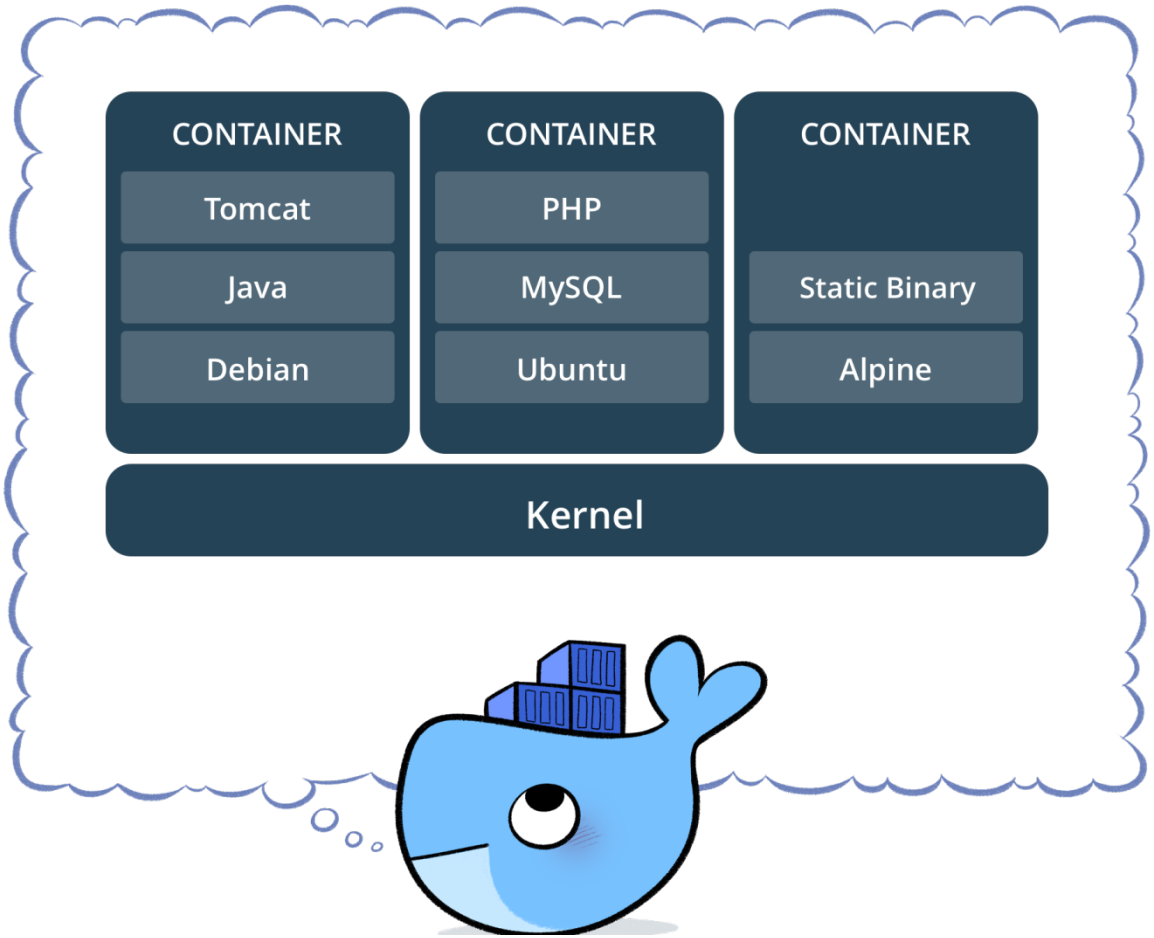
Downloads bookmark history and incorporates changes

- git pull

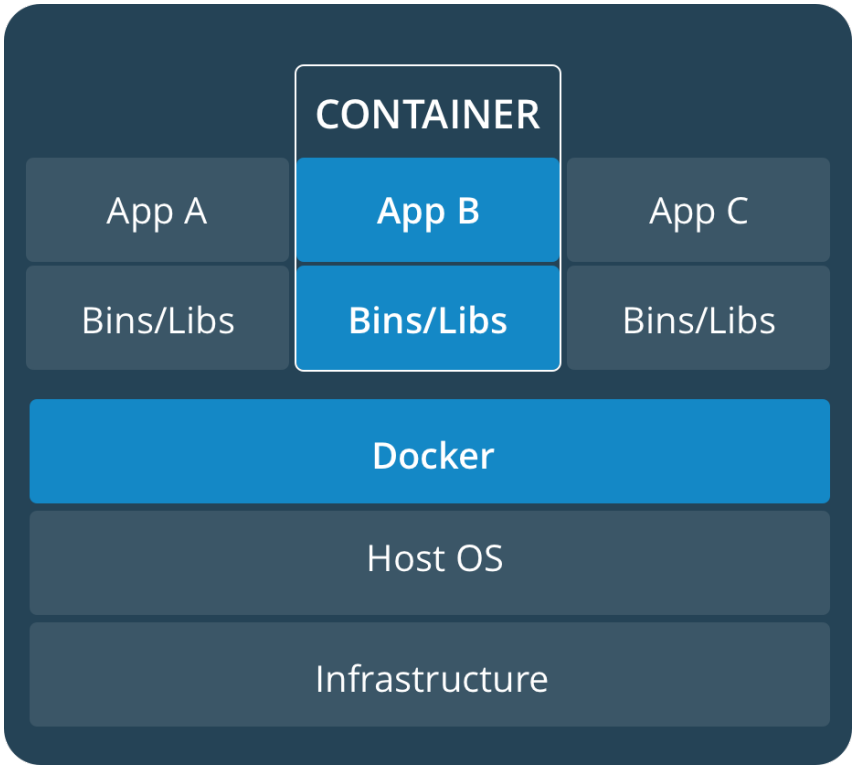
Containers virtualize with

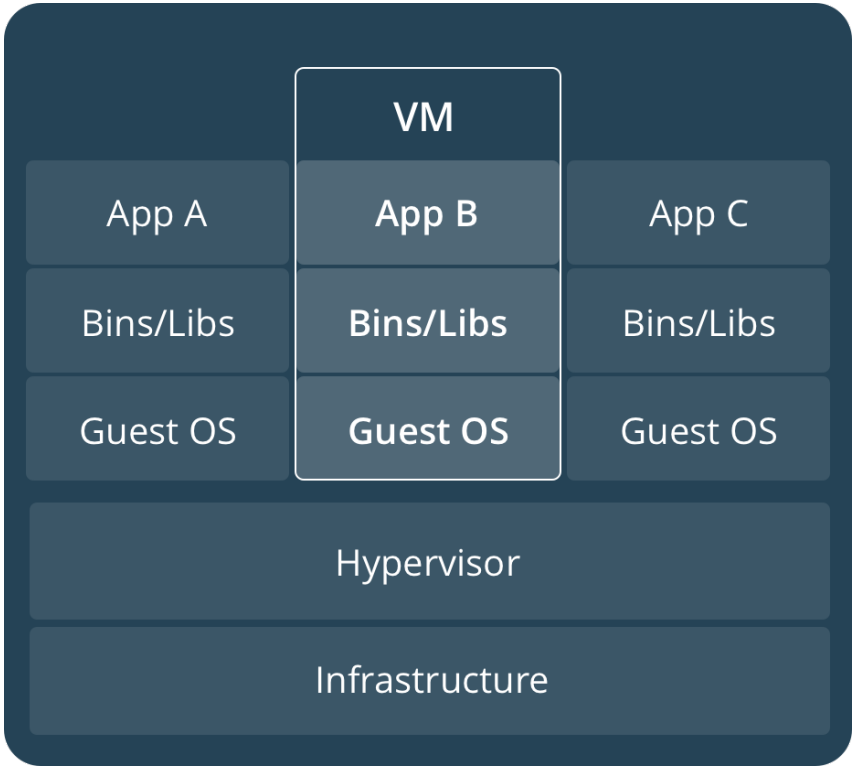
DOCKER

About Container

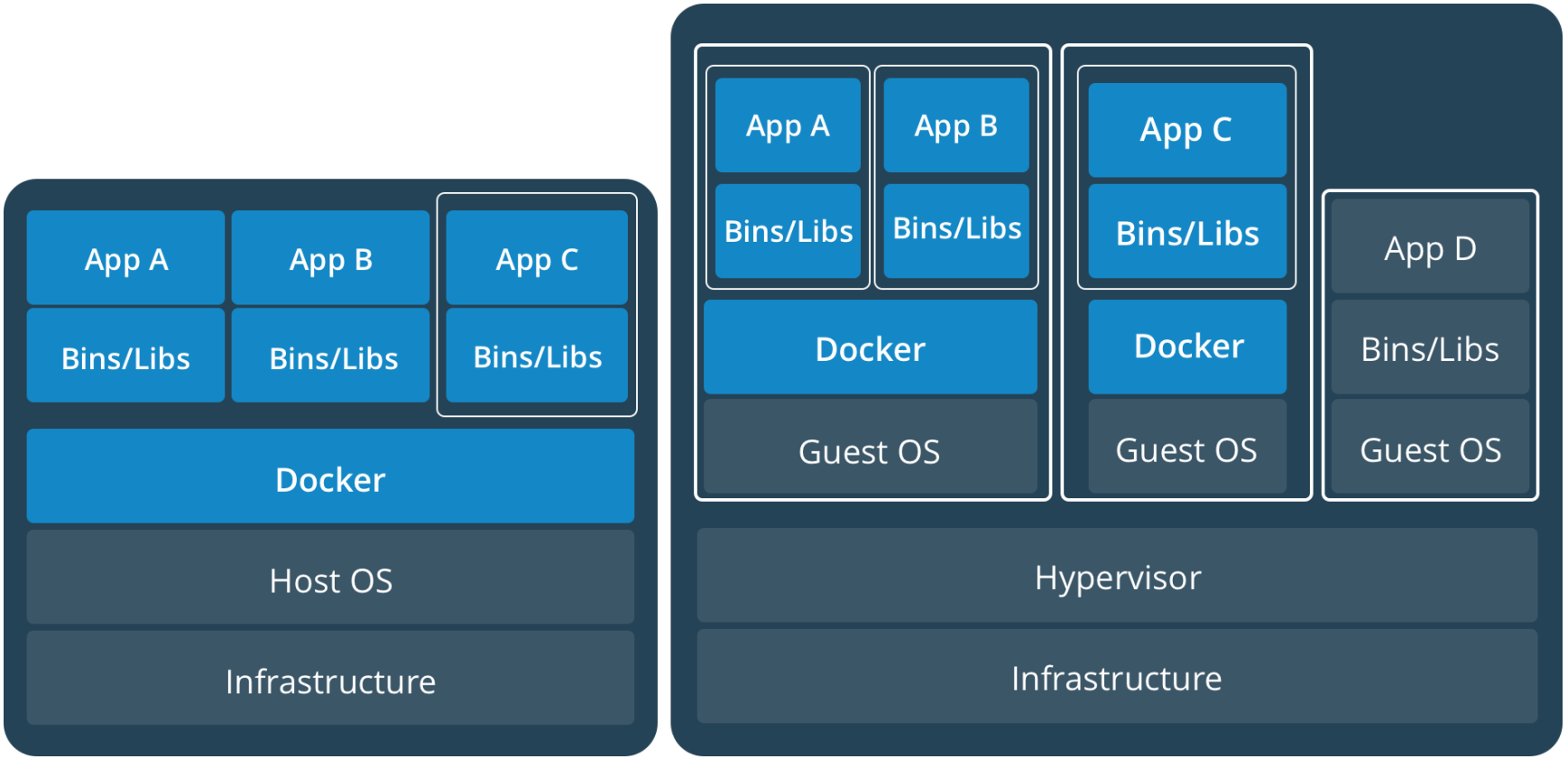


Comparing Containers and Virtual Machines

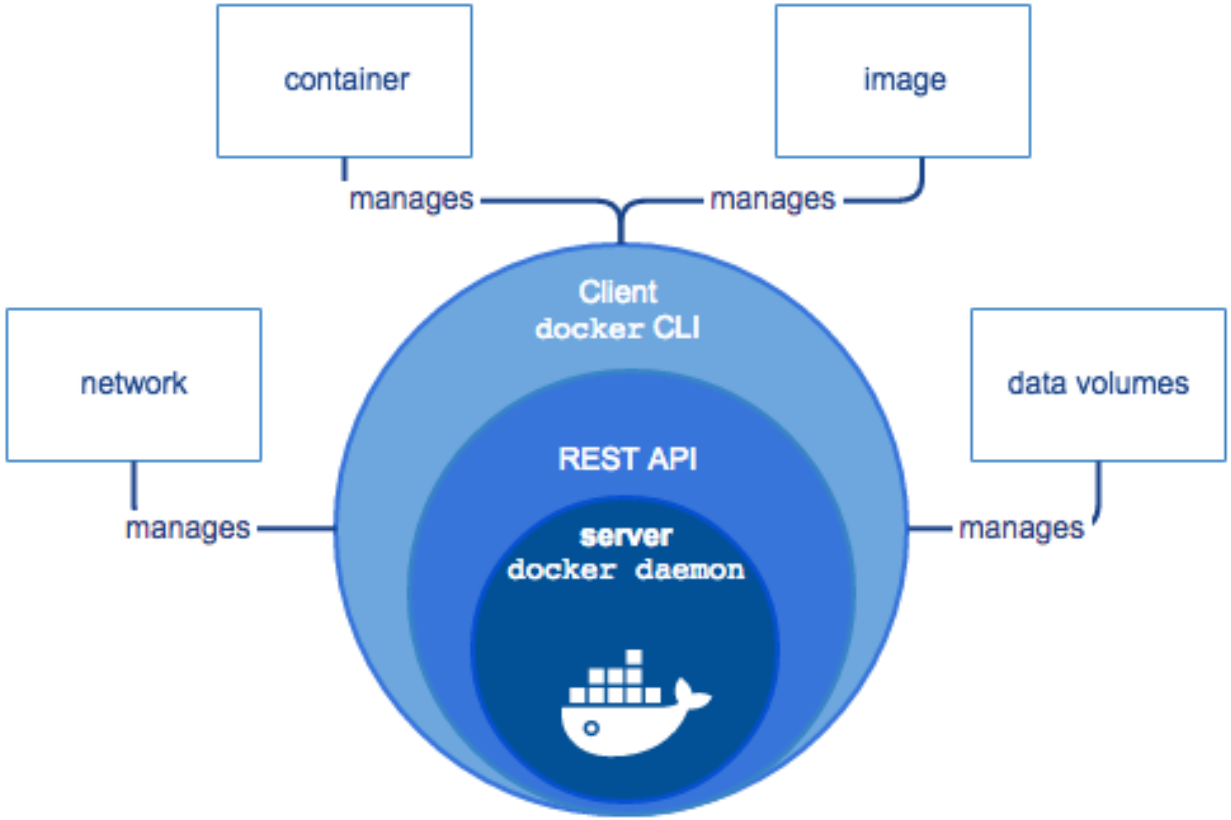




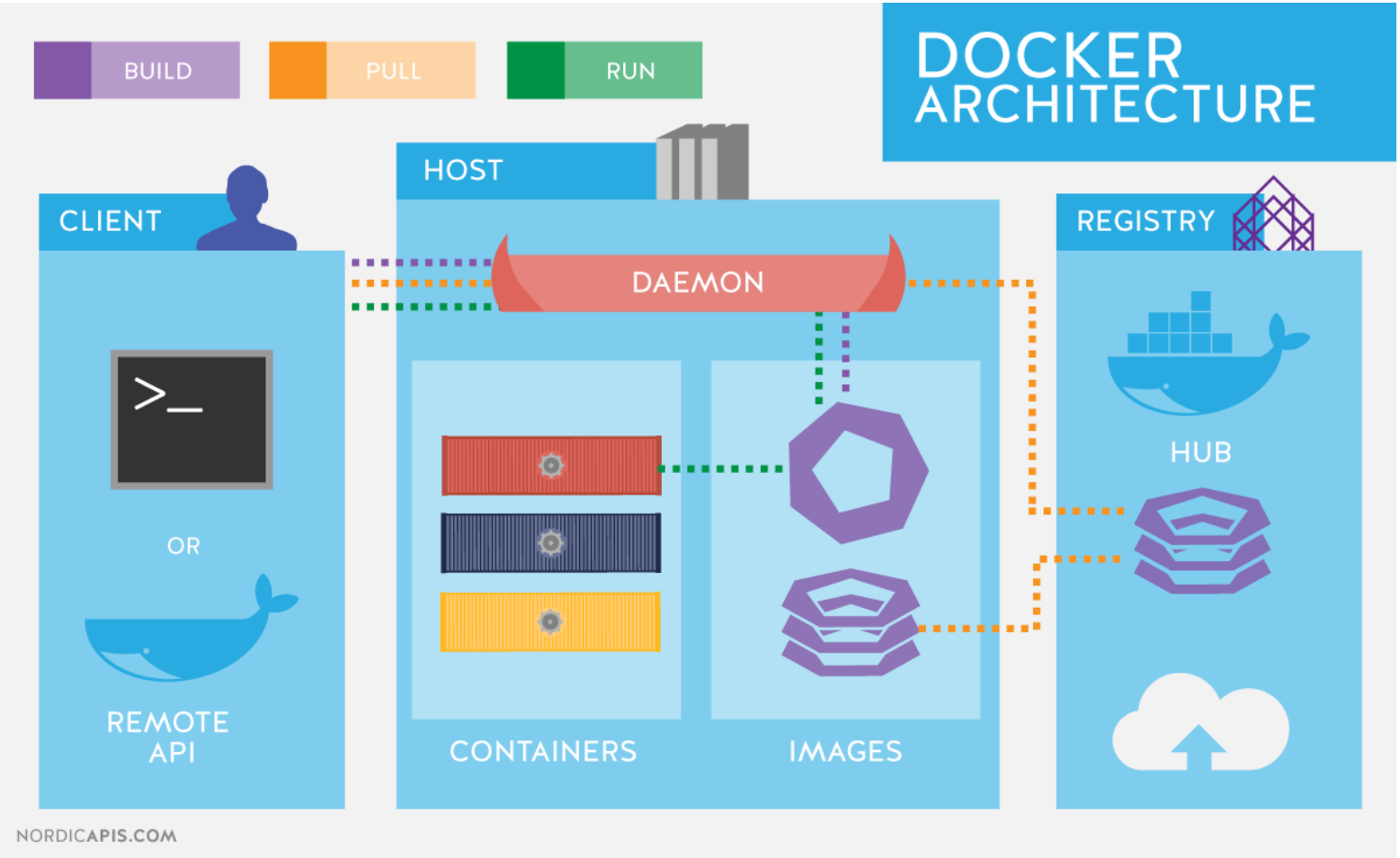
Containers and Virtual Machines Together



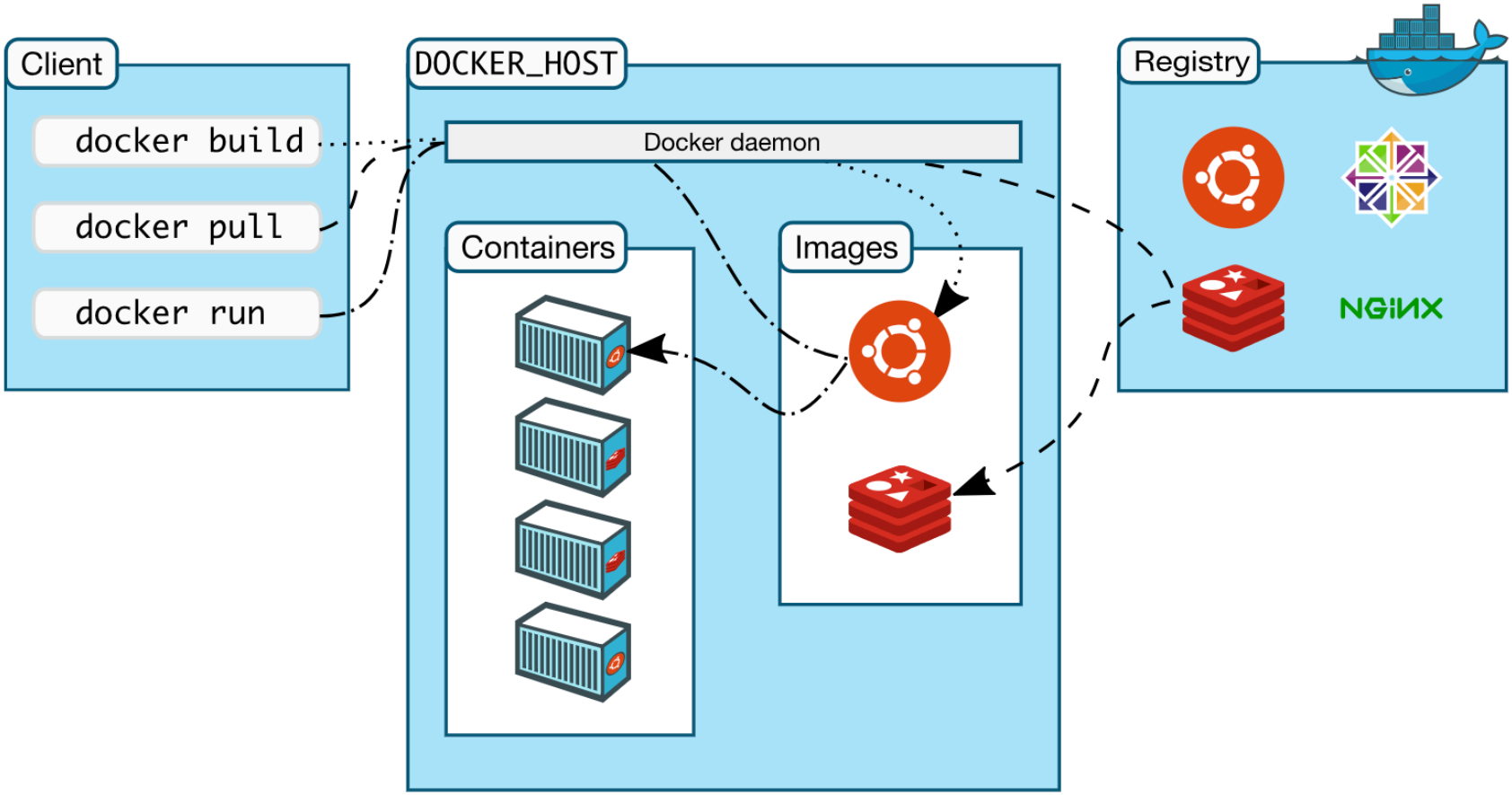
Docker engine



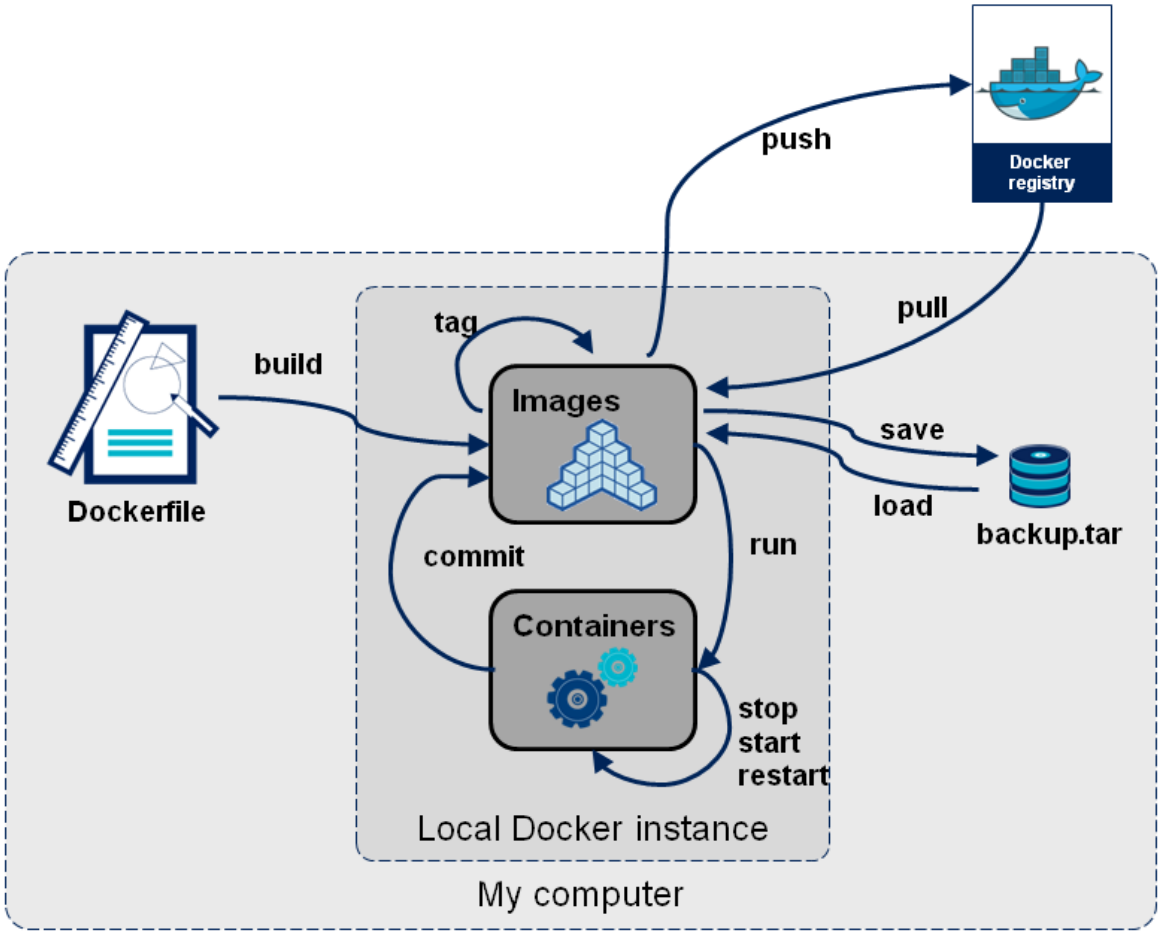
Docker Architecture



Docker Architecture#2



Docker Architecture#3



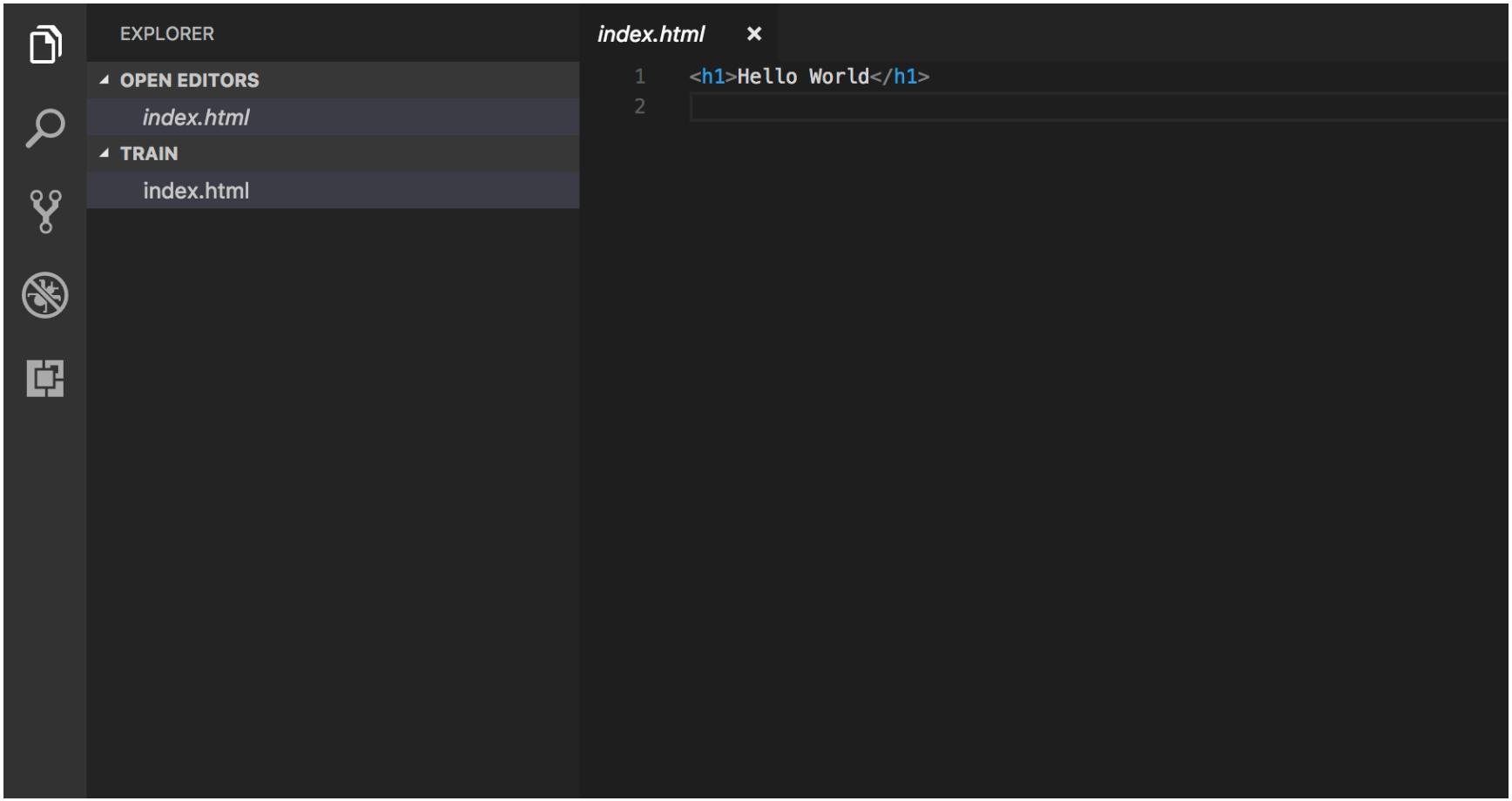
Hello World Docker

- เปิด cmd แล้วรันคำสั่งดังนี้

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

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

สร้าง file index.html



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 Configuration with

VAGRANT

Introduction

- Vagrant - the command line utility for managing the lifecycle of virtual machines
- Vagrant is a tool for building and managing virtual machine environments in a single workflow
- With an easy-to-use workflow and focus on automation, Vagrant lowers development environment setup time, increases production parity, and makes the "works on my machine" excuse a relic of the past.

Why Vagrant

- Vagrant provides easy to configure, reproducible, and portable work environments built on top of industry-standard technology and controlled by a single consistent workflow to help maximize the productivity and flexibility of you and your team.
- Vagrant gives you a disposable environment and consistent workflow for developing and testing infrastructure management scripts.
- Vagrant is designed for everyone as the easiest and fastest way to create a virtualized environment!

Project Setup

- Run command
  - mkdir vagrant_getting_started
  - cd vagrant_getting_started
  - vagrant init

- *** vagrant_getting_started คือชื่อ folder ที่เอาไว้เก็บ project

Boxes

- ติดตั้ง image (box)
  - vagrant box add ubuntu/xenial64

*** สามารถหา image(box) ได้ที่ website https://app.vagrantup.com/boxes/search

Using a box

- Open Vagrantfile add following command

```
Vagrant.configure("2") do |config|
    config.vm.box = “ubuntu/xenial64"
end
```

*** config คือ ชื่อตัวแปรเอาไว้อ้างอิงใน script สามารถตั้งชื่ออะไรก็ได้

*** เลข 2 หมายถึง version configure ของ vagrant ที่จะใช้

Up And SSH

- Run command for start vm
  - vagrant up <NAME>

- Run command for remote to vm
  - vagrant ssh <NAME>

*** <NAME> ใช้ในกรณีที่มีการ manage มากกว่า 1 vm

Synced Folders

- Vagrant will automatically sync your files to and from the guest machine.
- /vagrant is synceddirectory
- Run command
  - vagrant up
  - vagrant ssh
  - ls /vagrant

Output

Vagrantfile

Provisioning

- Provisioners in Vagrant allow you to automatically install software, alter configurations, and more on the machine as part of thevagrant upprocess.

```
Vagrant.configure("2") do |config|
  # ... other configuration

  config.vm.provision "shell", inline: "echo hello"
end
```

Shell Provisioner

SCRIPTING

```
$script = <<-SCRIPT
echo I am provisioning...
date > /etc/vagrant_provisioned_at
SCRIPT

Vagrant.configure("2") do |config|
  config.vm.provision "shell", inline: $script
end
```

Multi-Machine

- Vagrant is able to define and control multiple guest machines per Vagrantfile.
- This is known as a "multi-machine" environment.

Define machine

```
Vagrant.configure("2") do |config|
  config.vm.provision "shell", inline: "echo Hello"

  config.vm.define "web" do |web|
    web.vm.box = "apache"
  end

  config.vm.define "db" do |db|
    db.vm.box = "mysql"
  end
end
```

Networking

- In order to access the Vagrant environment created, Vagrant exposes some high-level networking options for things such as forwarded ports, connecting to a public network, or creating a private network.

Defining a Forwarded Port

FORWARD PORT

```
Vagrant.configure("2") do |config|
  config.vm.network "forwarded_port", guest: 80, host: 8080
end
```

FORWARD PORT PROTOCOL

```
Vagrant.configure("2") do |config|
  config.vm.network "forwarded_port", guest: 2003, host: 12003, protocol: "tcp"
  config.vm.network "forwarded_port", guest: 2003, host: 12003, protocol: "udp"
end
```

Private networks

DHCP

```
Vagrant.configure("2") do |config|
  config.vm.network "private_network", type: "dhcp"
end
```

STATIC IP

```
Vagrant.configure("2") do |config|
  config.vm.network "private_network", ip: "192.168.50.4"
end
```

Public networks

DHCP

```
Vagrant.configure("2") do |config|
  config.vm.network "public_network"
end
```

Using the DHCP Assigned Default Route

```
Vagrant.configure("2") do |config|
  config.vm.network "public_network",
    use_dhcp_assigned_default_route: true
end
```

STATIC IP

```
config.vm.network "public_network", ip: "192.168.0.17"
```

Command-Line Interface

box manages boxes: installation, removal, etc.

destroy stops and deletes all traces of the vagrant machine

global-status outputs status Vagrant environments for this user

halt stops the vagrant machine

help shows the help for a subcommand

init initializes a new Vagrant environment by creating a Vagrantfile

login log in to HashiCorp's Vagrant Cloud

package packages a running vagrant environment into a box

plugin manages plugins: install, uninstall, update, etc.

port displays information about guest port mappings

powershell connects to machine via powershell remoting

provision provisions the vagrant machine

Command-Line Interface

push deploys code in this environment to a configured destination

rdp connects to machine via RDP

reload restarts vagrant machine

resume resume a suspended vagrant machine

snapshot manages snapshots: saving, restoring, etc.

ssh connects to machine via SSH

ssh-config outputs OpenSSH valid configuration

status outputs status of the vagrant machine

suspend suspends the machine

up starts and provisions the vagrant environment

validate validates the Vagrantfile

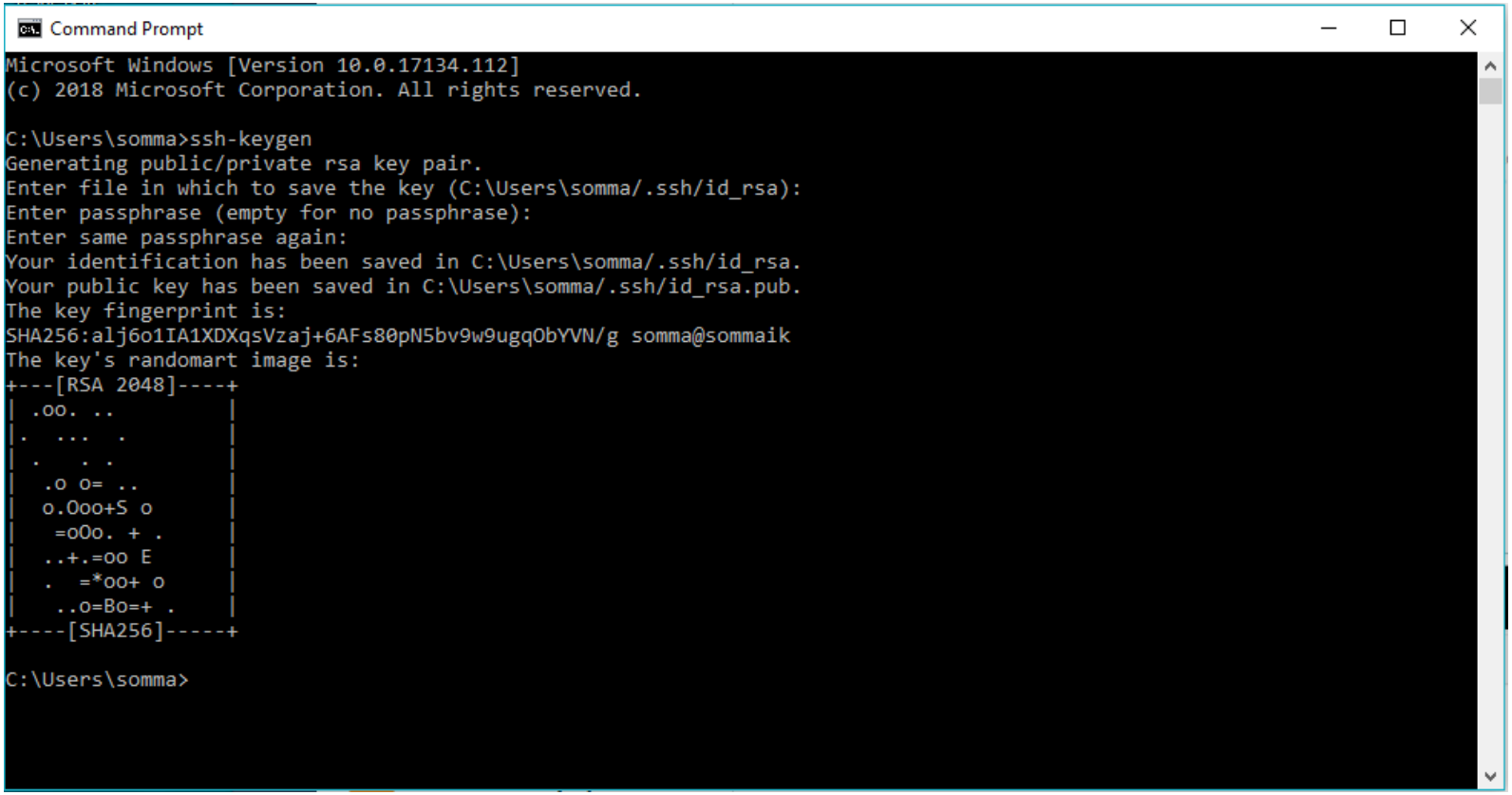
version prints current and latest Vagrant version

การใช้ vagrant with

SSH KEY

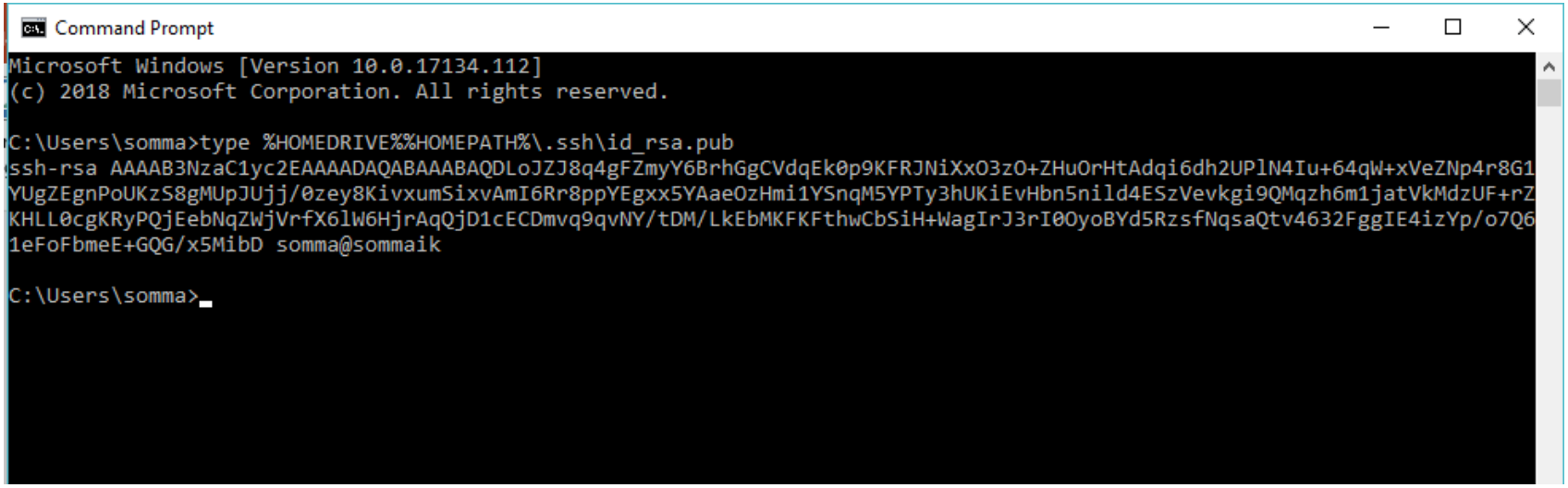
การสร้าง SSH KEY

- พิมพ์ ssh-keygen



การสร้าง SSH KEY

- ตรวจสอบโดยการพิมพ์คำสั่งนี้ type %HOMEDRIVE%%HOMEPATH% \.ssh\id_rsa.pub



การ Run Vagrant

- สร้างโปรเจคที่ต้องการ แล้ว run คำสั่ง vagrant init จะได้ Vagrantfile

```
C:\WINDOWS\system32>e:

E:\>cd Nooti3w

E:\Nooti3w>cd workspace_train

E:\Nooti3w\workspace_train>mkdir vagrant_getting_started

E:\Nooti3w\workspace_train>cd vagrant_getting_started

E:\Nooti3w\workspace_train\vagrant_getting_started>vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.

E:\Nooti3w\workspace_train\vagrant_getting_started>dir
 Volume in drive E has no label.
 Volume Serial Number is 6A49-8E1C

 Directory of E:\Nooti3w\workspace_train\vagrant_getting_started

06/27/2018  04:48 PM    <DIR>          .
06/27/2018  04:48 PM    <DIR>          ..
06/27/2018  04:48 PM             3,081 Vagrantfile
               1 File(s)          3,081 bytes
               2 Dir(s)  302,528,217,088 bytes free

E:\Nooti3w\workspace_train\vagrant_getting_started>
```

การ Run Vagrant

- Run คำสั่ง vagrant box add ubuntu/xenial64

```
E:\Nooti3w\workspace_train\vagrant_getting_started>vagrant box add ubuntu/xenial64
==> box: Loading metadata for box 'ubuntu/xenial64'
    box: URL: https://vagrantcloud.com/ubuntu/xenial64
==> box: Adding box 'ubuntu/xenial64' (v20180622.0.0) for provider: virtualbox
The box you're attempting to add already exists. Remove it before
adding it again or add it with the `--force` flag.

Name: ubuntu/xenial64
Provider: virtualbox
Version: 20180622.0.0
```

- จ

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/xenial64"
end
```

โปรเจค

เรียกใช้ ssh key ใน Vagrantfile

แทรกเข้าในใน Vagrantfile

```
config.ssh.insert_key = false
config.ssh.private_key_path = ["~/.ssh/id_rsa", "~/.vagrant.d/insecure_private_key"]
config.vm.provision "file", source: "~/.ssh/id_rsa.pub", destination: "~/.ssh/authorized_keys"
config.vm.provision "file", source: "~/.ssh/id_rsa", destination: "~/.ssh/id_rsa"
config.vm.provision "file", source: "~/.ssh/id_rsa.pub", destination: "~/.ssh/id_rsa.pub"
config.vm.provision "shell", inline: <<-EOC
  sudo sed -i -e "\\#PasswordAuthentication yes# s#PasswordAuthentication yes#PasswordAuthentication no#g" /etc/ssh/sshd_config
  sudo service ssh restart
EOC
```

การ Run Vagrant

- Run คำสั่ง vagrant up

```
E:\Nooti3w\workspace_train\vagrant_getting_started>vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'ubuntu/xenial64' is up to date...
==> default: Clearing any previously set forwarded ports...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default: Warning: Connection aborted. Retrying...
    default: Warning: Connection reset. Retrying...
    default: Warning: Connection aborted. Retrying...
    default: Warning: Connection reset. Retrying...
    default: Warning: Connection aborted. Retrying...
    default: Warning: Connection reset. Retrying...
    default: Warning: Remote connection disconnect. Retrying...
    default: Warning: Connection aborted. Retrying...
    default: Warning: Connection reset. Retrying...
    default: Warning: Remote connection disconnect. Retrying...
    default: Warning: Connection aborted. Retrying...
    default: Warning: Connection reset. Retrying...
    default: Warning: Connection aborted. Retrying...
    default: Warning: Remote connection disconnect. Retrying...
    default: Warning: Connection aborted. Retrying...
    default: Warning: Connection reset. Retrying...
    default: Warning: Connection aborted. Retrying...
    default: Warning: Remote connection disconnect. Retrying...
    default: Warning: Connection aborted. Retrying...
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
```

การ Run Vagrant

- Run คำสั่ง vagrant box list

```
E:\Nooti3w\workspace_train\vagrant_getting_started>vagrant box list
ubuntu/xenial64 (virtualbox, 20180622.0.0)
```

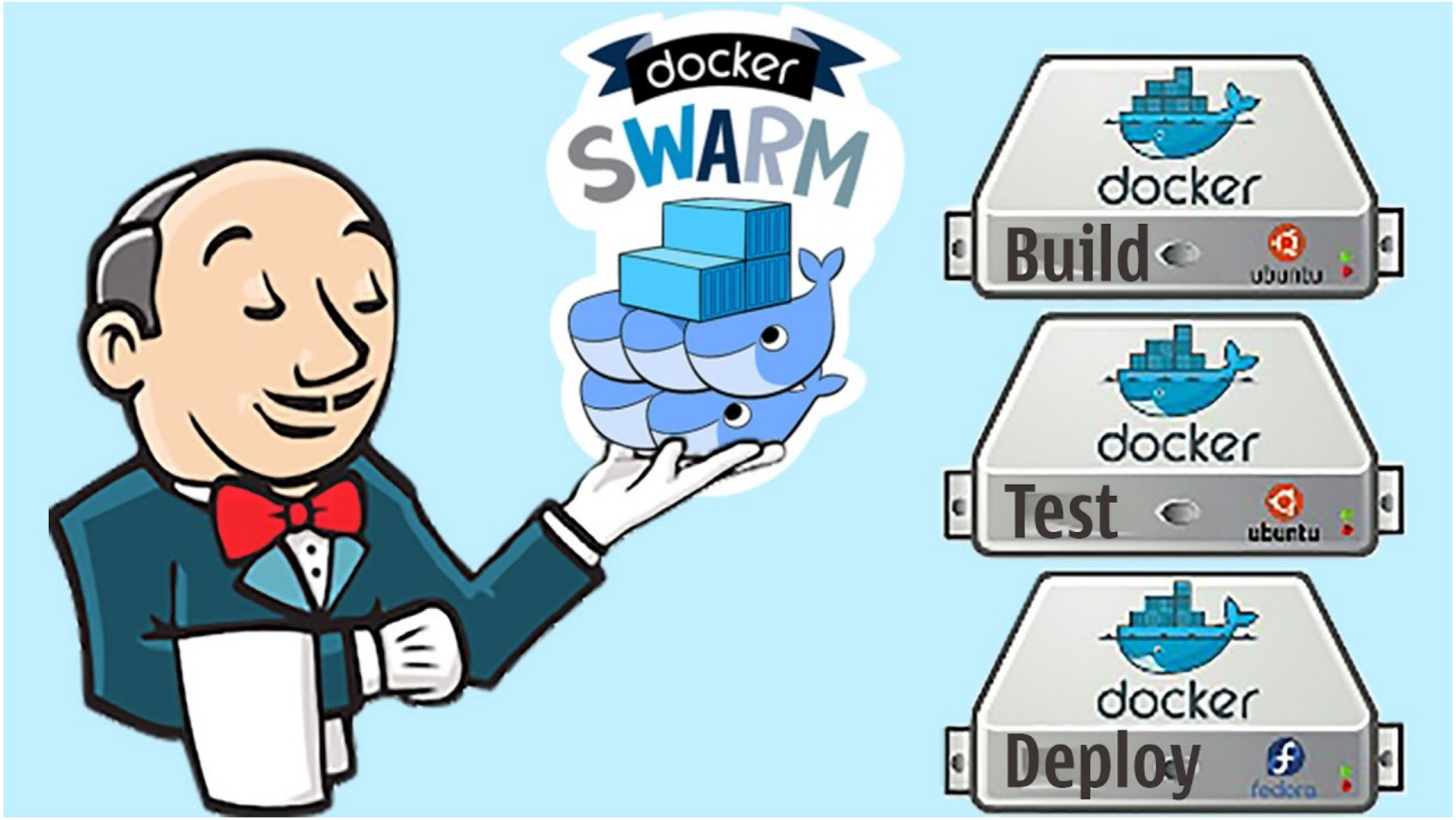
- Ru

```
E:\Nooti3w\workspace_train\vagrant_getting_started>vagrant ssh
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-128-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.


vagrant@ubuntu-xenial:~$
```

ORCRESTRATION WITH

DOCKER SWARM

Jenkins With docker swarm



Docker Swarm command

- docker swarm init : สำหรับเริ่มต้น docker swarm mode (manager)
- docker swarm join : สำหรับให้เครื่อง worker join เข้า manager
- docker swarm join-token : สำหรับสร้าง key เพื่อให้เข้ามา join
- docker service create : สร้าง service เพื่อให้บริการ
- docker service inspect : ตรวจสอบ service
- docker service ls : ดู service ทั้งหมด
- docker service rm : ลบ service
- docker service scale : เพิ่ม / ลด จำนวน node ที่ รัน service
- docker service ps : ดูสถานะ service
- docker service update : ปรับปรุง service
- docker node ls : ดู node ทั้งหมด

เริ่มสร้าง docker swarm (manager)

- runคำสั่ง ใน terminal ดังนี้

docker swarm init --advertise-addr=192.168.33.12

```
[vagrant@mgr1:~$ docker swarm init --advertise-addr=192.168.33.12
Swarm initialized: current node (izupnbin1ra7cv2a9ur4k95ve) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3dip03ya9iixhhsk1aegfs3ptd30pwvuzpk7413iookg6nca99-egmkmkx8hvhh8fok2d8zfppmx 192.168.33.12:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

หลังจากที่รันคำสั่งเสร็จจะได้ผลลัพธ์เป็น เลข token เพื่อเอาไป run ที่เครื่อง worker ต่อไป

Join เข้ากลุ่ม swarm

- หลังจากที่ได้สร้างเครื่อง manager ไปแล้วก็ต้องนำคำสั่งที่ได้จาก console ของเครื่อง manager มารันในเครื่อง worker ดังตัวอย่าง

```
[vagrant@node1:~$ docker swarm join --token SWMTKN-1-3dip03ya9iixhhsk1aegfs3ptd30pwvuzpk7413iookg6nca99-egmkmkx8hvhh8fok2d8zfppmx 192.168.33.12:2377
This node joined a swarm as a worker.
```

ตรวจสอบจำนวน node ทั้งหมด

- runคำสั่ง ใน terminal ดังนี้

docker node list

```
vagrant@mgr1:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
izupnbin1ra7cv2a9ur4k95ve *   mgr1                Ready               Active              Leader              18.03.1-ce
mdi3i3cjpknx8d58ujrfc9ujz     node1               Ready               Active                                  18.03.1-ce
```

สร้าง service บนเครื่อง manager

- runคำสั่ง ใน terminal ดังนี้

docker service create <<option>> nginx

```
99nj4lwci1spqifd7a2iwn7qm
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
```

** option ที่ควรมี

--replicas 2

--name web_server

--constraint "node.role != manager"

--publish 8080:80

replicas คือ จำนวน node ที่ต้องการสร้าง

name ชื่อที่เอาไว้อ้างอิง

constraint เงื่อนไขในการเลือกเครื่อง

ดู service ทั้งหมด

- runคำสั่ง ใน terminal ดังนี้

docker service ls

```
verify: Service converged
[vagrant@mgr1:~$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
99nj4lwci1sp        competent_chatterjee   replicated          1/1                 nginx:latest
```

เพิ่มลดจำนวน node

- runคำสั่ง ใน terminal ดังนี้

docker service scale <service_name>=2

```
[vagrant@mgr1:~$ docker service scale competent_chatterjee=2
competent_chatterjee scaled to 2
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
```

ตรวจสอบ service

- runคำสั่ง ใน terminal ดังนี้

docker service ps <service_name>

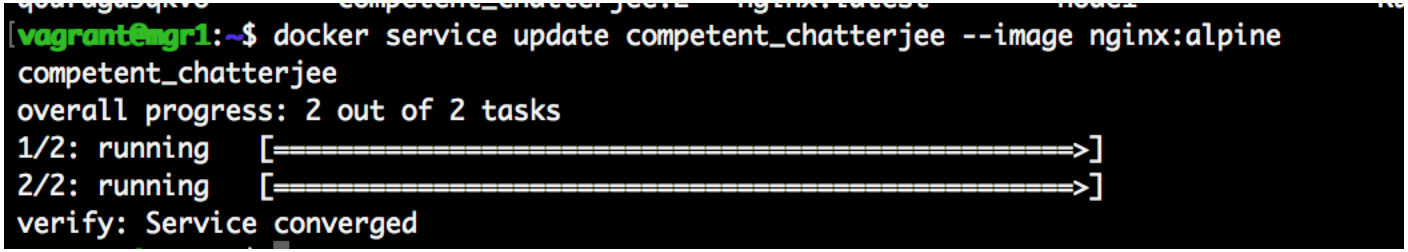
```
vagrant@mgr1:~$ docker service ps competent_chatterjee
ID                  NAME                    IMAGE               NODE                DESIRED STATE       CURRENT STATE            ERROR               PORTS
txebpk4l9vsb        competent_chatterjee.1  nginx:latest        mgr1                Running             Running 6 minutes ago
q0u7dgd9qkvo        competent_chatterjee.2  nginx:latest        node1               Running             Running 37 seconds ago
```

จะแสดงข้อมูลของ service ว่าทำงานอยู่บน node ไหนตามจำนวน replicas ที่ได้ตั้งเอาไว้

update service

- runคำสั่ง ใน terminal ดังนี้

docker service update <service_name> --image nginx:alpine



au service

- runคำสั่ง ใน terminal ดังนี้

docker service rm <service_name>

```
[vagrant@mgr1:~$ docker service rm competent_chatterjee
competent_chatterjee
```

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.

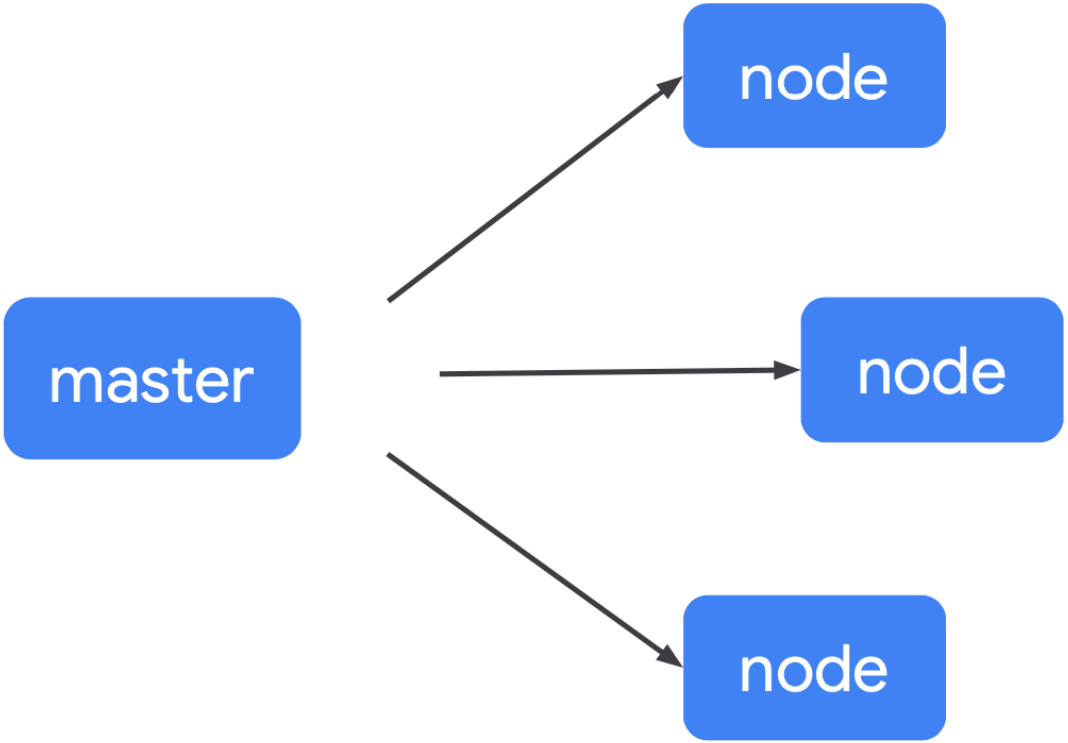
Run K8s Anywhere

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.

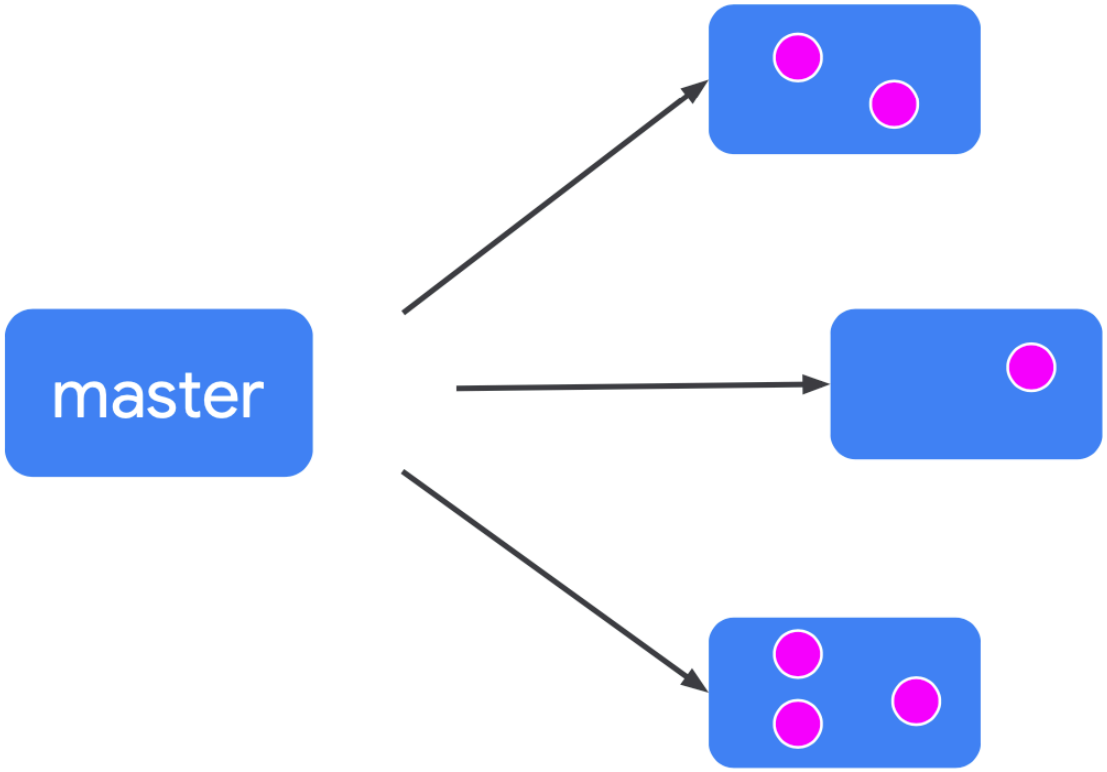
Kubernetes runs Applications in a Cluster.

Applications = Pods

Cluster



Pods in a Cluster



```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: hello-world
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
      - name: hello-world-server
        image: gcr.io/megangcp/helloworld:v0.0.1
        ports:
        - containerPort: 8080
```

Pod

Docker Image

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

Kubernetes with

MINIKUBE

Start

- minikube start --driver=virtualbox --container-runtime=cri-o

** driver เอาไว้บอกว่าต้องการให้สร้าง cluster ด้วย vm ไหน

** container-runtime เอาไว้บอกว่าต้องการให้ใช้ runtime อะไร มี docker, containerd, cri-o

*** option อื่นๆ

--memory=5120

--cpus=4

--kubernetes-version=v1.11.0

minikube command list

- minikube delete ลบ vm ออก
- minikube status ดูสถานะ
- minikube version ดู version
- minikube service ดูข้อมูลของ service
- minikube dashboard ดู dashboard
- minikube ip ดู ip
- minikube logs ดู logs
- minikube addons list ดู addons ที่ติดตั้ง
- minikube tunnel

Manage Kubernetes with

KUBECTL

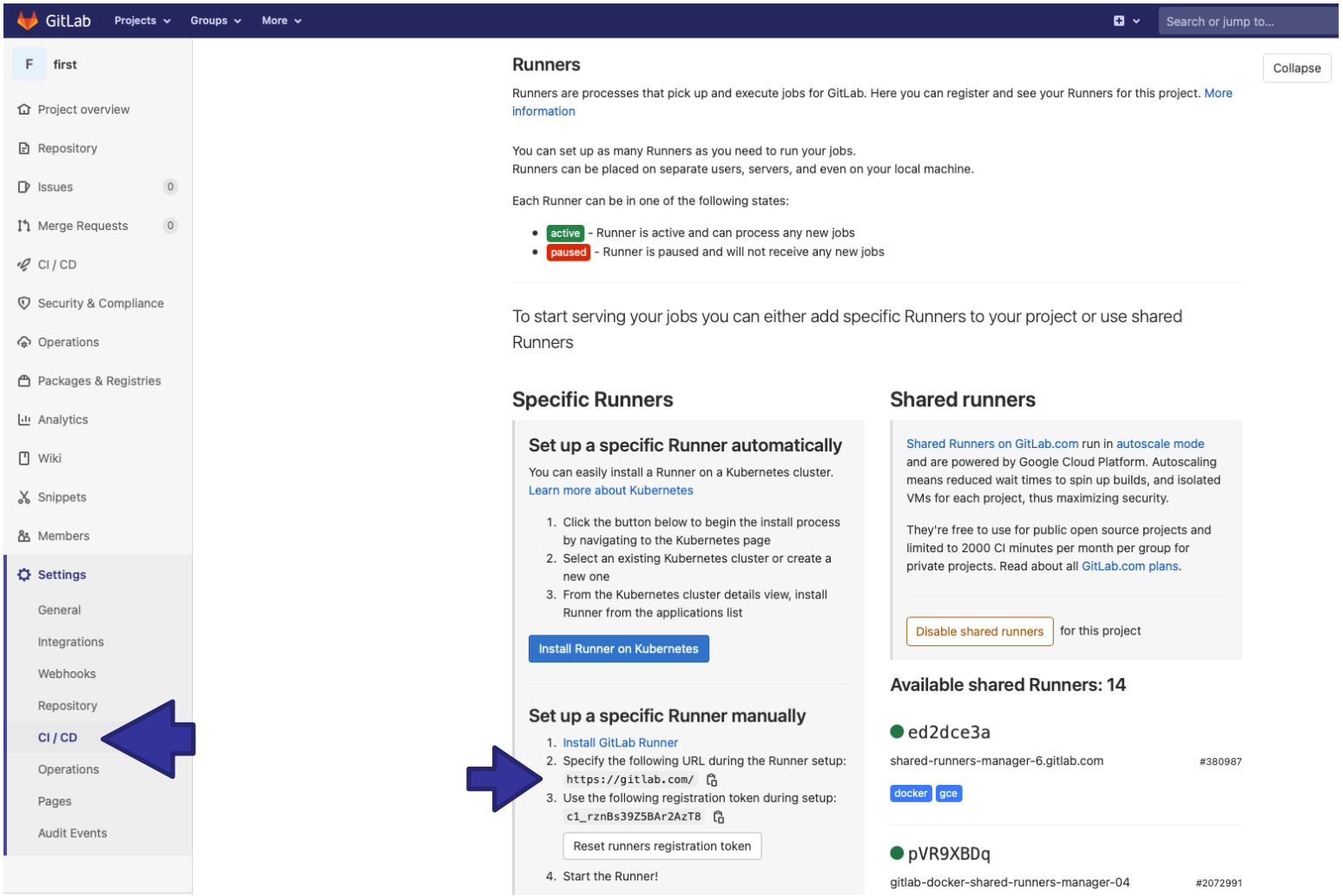
kubectl command list

- 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

CI/CD with

GIT RUNNER

Setup Git runner



.gitlab-ci.yml

.gitlab-ci.yml 829 Bytes

Edit | Web IDE | Replace | Delete

```yaml
image: docker:stable

variables:
  CONTAINER_IMAGE: registry.pnpsw.com/thaicom/oap-rm-app
  APP_NAME: oap_rm-app

stages:
  - build
  - deploy

before_script:
  - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $CI_REGISTRY

build:
  stage: build
  services:
    - docker:dind
  tags:
    - pnp-agent
  script:
    - docker build --cache-from $CONTAINER_IMAGE:latest --tag $CONTAINER_IMAGE:$CI_PIPELINE_ID --tag $CONTAINER_IMAGE:latest .
    - docker push $CONTAINER_IMAGE:$CI_PIPELINE_ID
    - docker push $CONTAINER_IMAGE:latest
  only:
    - tags
    - web

depoy:
  stage: deploy
  tags:
    - pnp-ssh-agent
  variables:
    GIT_STRATEGY: none
  script:
    - docker service update --with-registry-auth --image ${CONTAINER_IMAGE}:latest ${APP_NAME}
  environment:
    name: production
  only:
    - tags
    - web
```

Any questions?