

# Temmy Alex

Programming Enthusiast

Focus in Web Development

Currently work in KoinWorks as Backend Engineer



**Temmy Alex**

8 years experience as Web  
Developer

## Education Background



**UNIVERSITAS  
BUDI LUHUR**

**2011-2015**

**Bachelor Degree**

Information System

# Basic Docker

- ☐ Introduction
- ☐ Architecture
- ☐ Install
- ☐ Docker Registry
- ☐ Docker Image
- ☐ Docker Container

# Basic Docker



## Introduction



Architecture



Install



Docker Registry



Docker Image



Docker Container

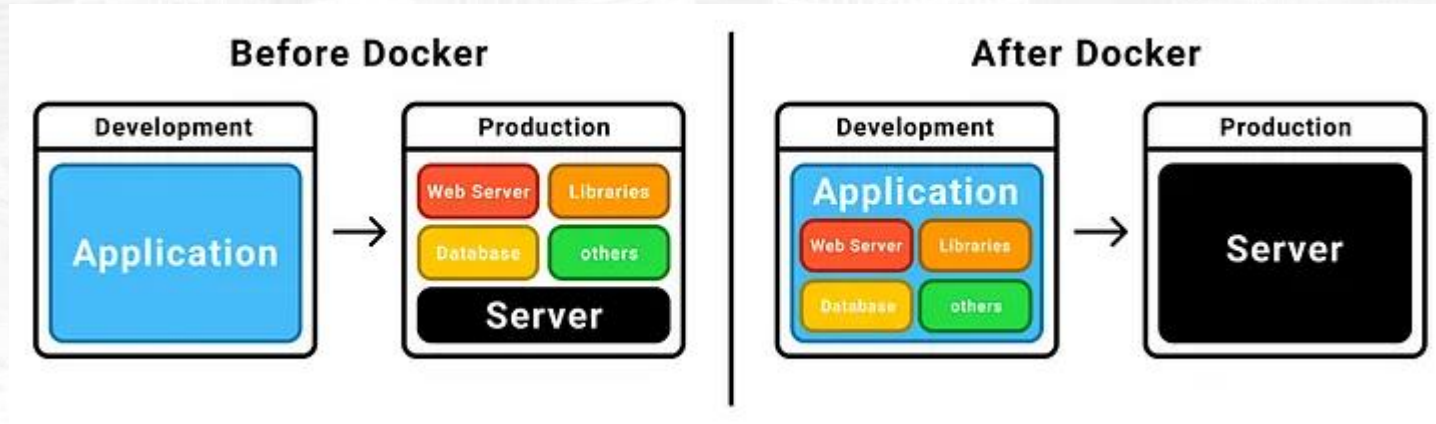
# Apa itu Docker?

Docker merupakan salah satu container manager yang saat ini paling banyak digunakan untuk keperluan deployment yang biasanya digunakan melalui VPS atau Cloud Server dan tentunya Docker ini merupakan teknologi Free dan Open Source sehingga dapat digunakan secara gratis dan tidak terbatas

Untuk referensi mengenai Docker dapat diakses <https://www.docker.com/>

# Mengapa harus menggunakan Docker?

Sebelum ada Docker untuk proses deployment menggunakan Virtual Machine yang cukup memakan banyak resource namun dengan adanya teknologi Docker proses deployment dapat menghemat resource jauh lebih banyak dengan konfigurasi yang lebih sederhana serta dapat digunakan pada beberapa platform cloud server



# Basic Docker

☐ Introduction



**Architecture**

☐ Install

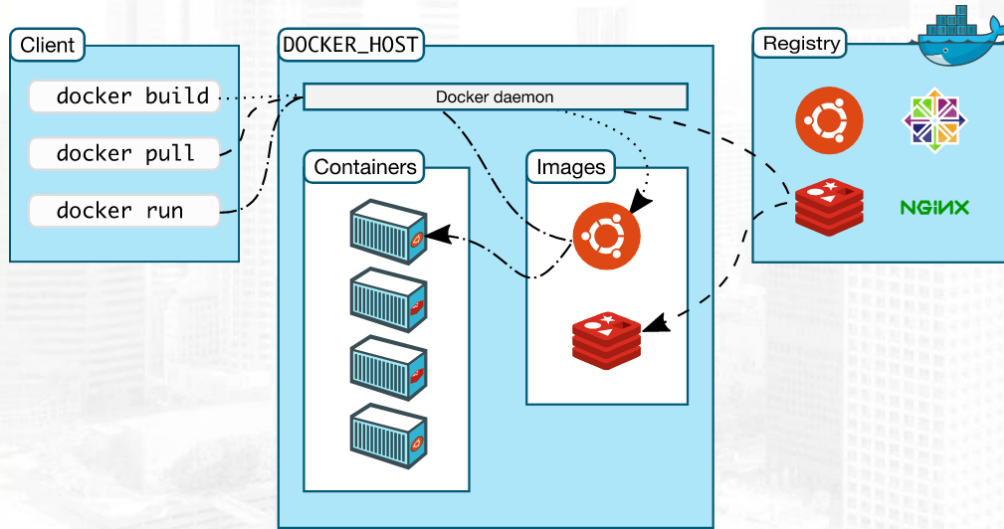
☐ Docker Registry

☐ Docker Image

☐ Docker Container




# Arsitektur



1. Docker Client digunakan untuk menjalankan perintah docker
2. Docker Daemon tempat dimana aplikasi berjalan pada host machine docker
3. Docker Registry tempat penyimpanan docker image
4. Docker Images kumpulan file untuk membuat container
5. Docker container tempat untuk menjalankan aplikasi

# Basic Docker

- ☐ Introduction
- ☐ Architecture
-  ☒ **Install**
- ☐ Docker Registry
- ☐ Docker Image
- ☐ Docker Container



# Install Docker in Windows 10

1. Download installer docker di website resmi docker <https://www.docker.com/get-started/> kemudian pilih **Download for Windows**

## Get Started with Docker

We have a complete container solution for you – no matter who you are and where you are on your containerization journey.



### Docker Desktop

Developer productivity tools and a local Kubernetes environment.

Download for Windows



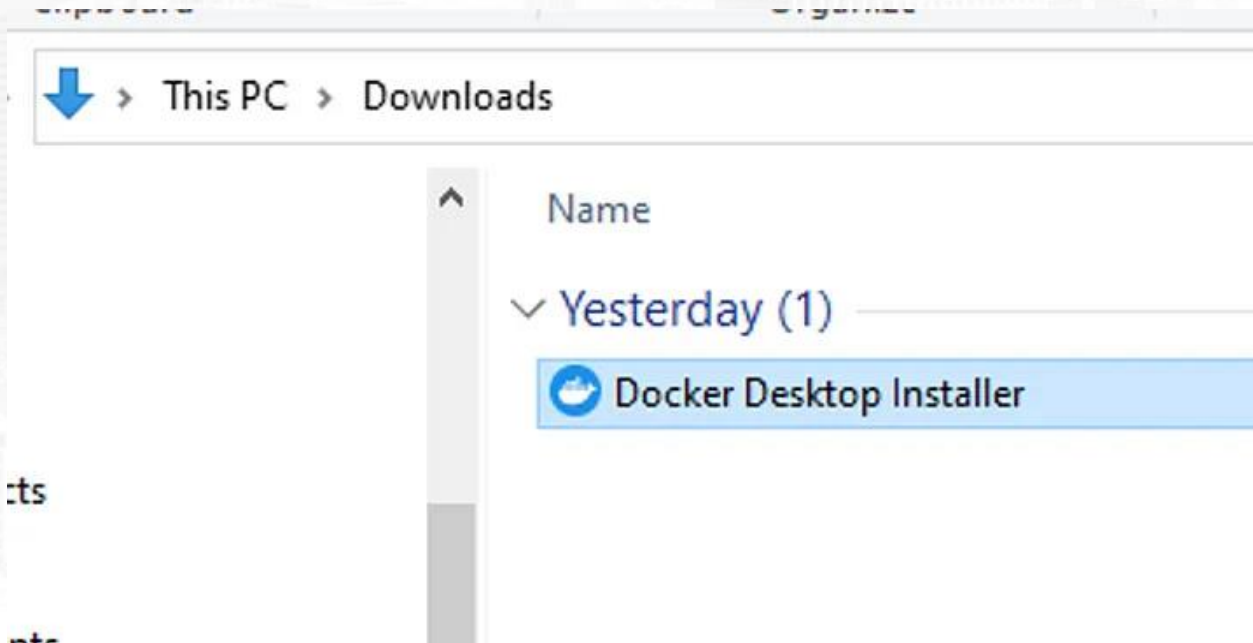
### Docker Hub

Cloud-based application registry and development team collaboration services.

Signup

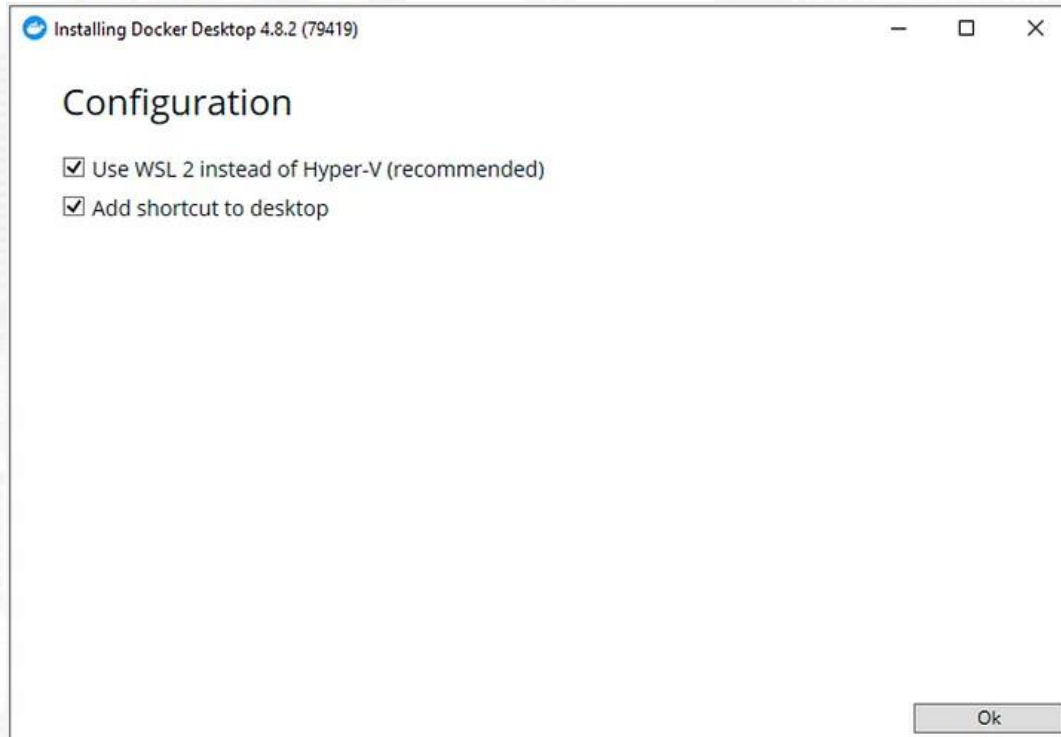
# Install Docker in Windows 10

2. Kemudian klik 2x untuk menjalankan installer yang sudah terdownload



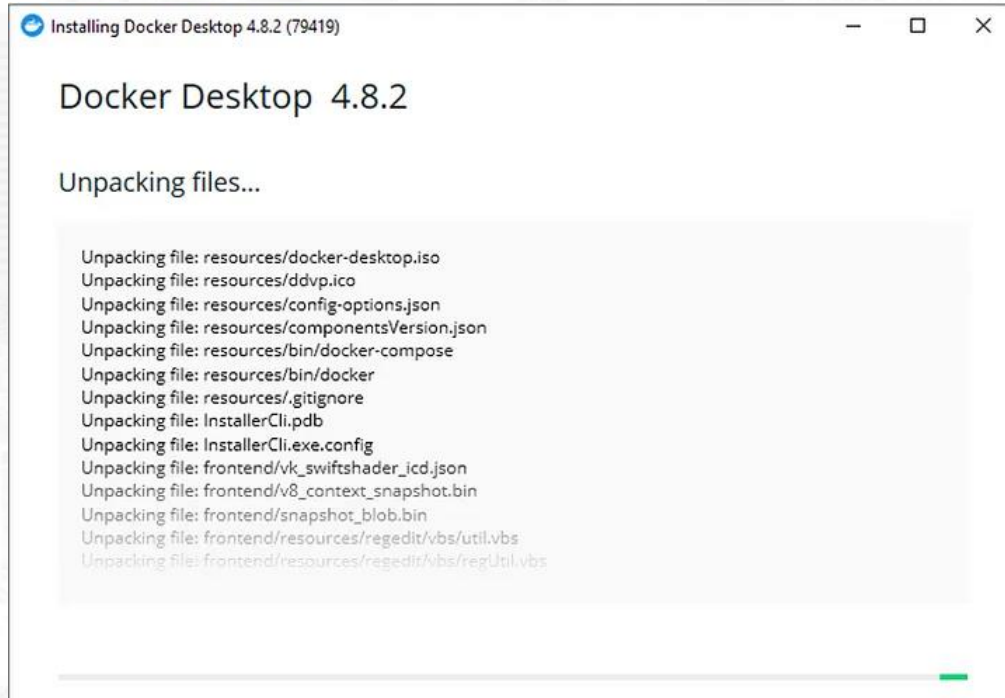
# Install Docker in Windows 10

3. Untuk menjalankan Linux on Windows silahkan centang pilihan **Use WSL 2...**



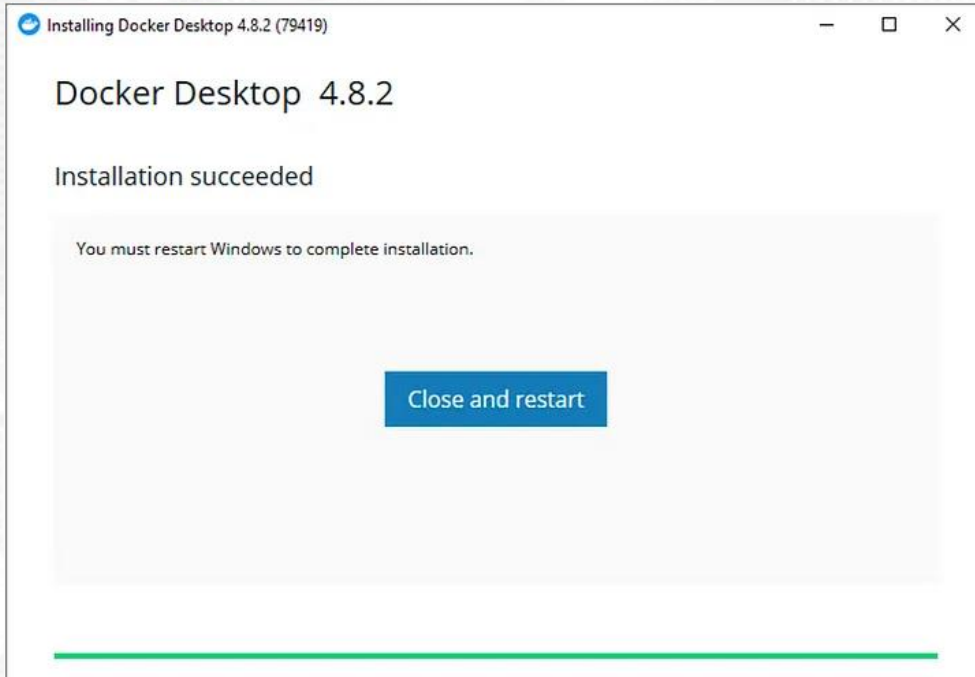
# Install Docker in Windows 10

4. Klik OK dan tunggu hingga proses selesai



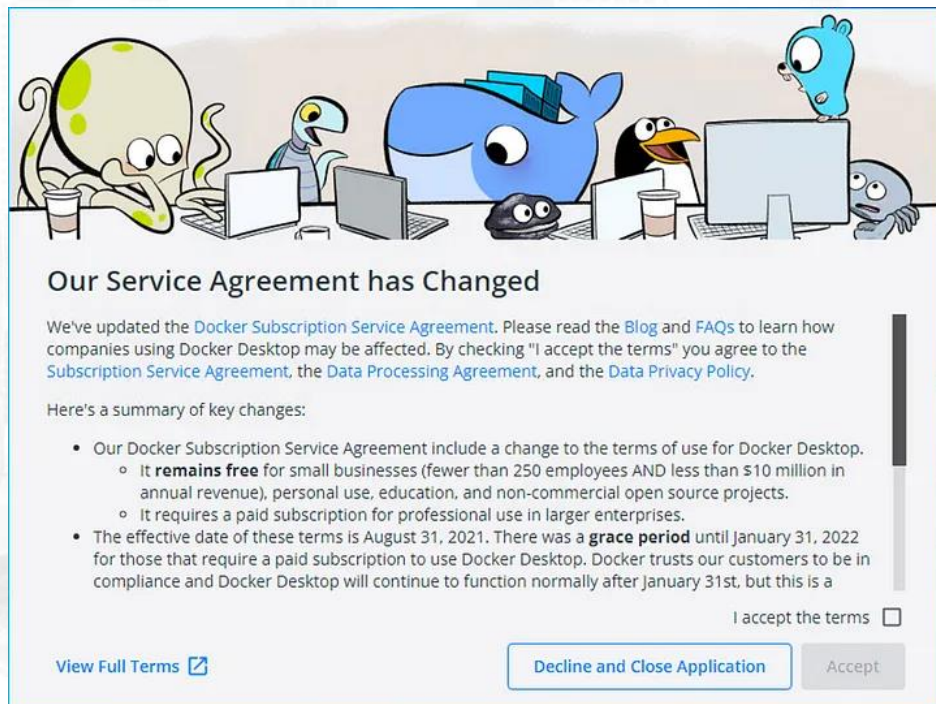
# Install Docker in Windows 10

5. Setelah proses selesai pilih **Close and Restart**, proses restart merupakan proses mandatory untuk melanjutkan proses instalasi



# Install Docker in Windows 10

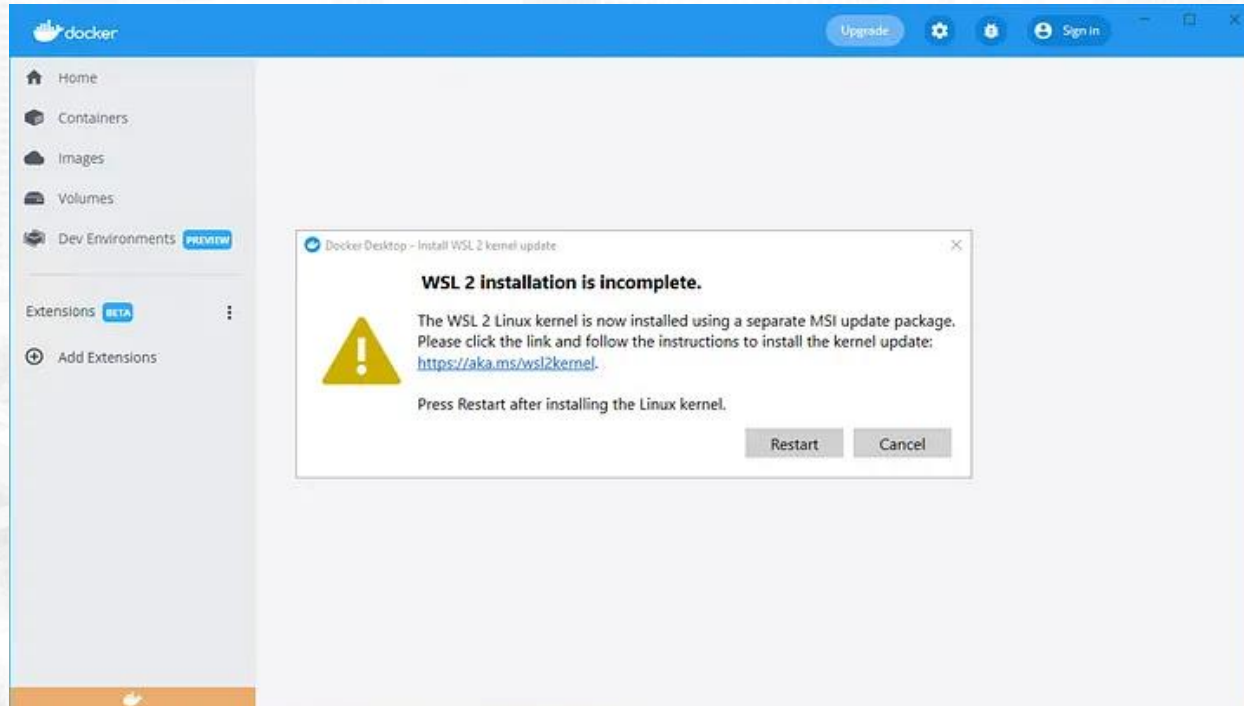
6. Setelah proses restart selesai akan muncul menu agreement klik **centang** dan Button **Accept**





# Install Docker in Windows 10

7. Setelah proses agreement selesai maka akan muncul jendela docker bahwa proses WSL belum selesai, jangan tekan tombol Restart dan klik link yang terdapat pada dialog window



# Install Docker in Windows 10

8. Kemudian selanjutnya download update pada package linux kernel

1. Download the latest package:

- [WSL2 Linux kernel update package for x64 machines](#) 

## ① Note

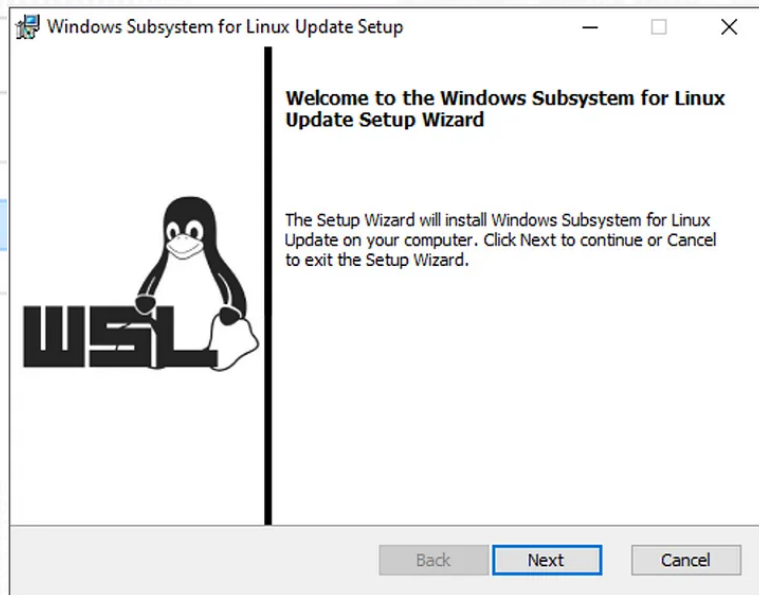
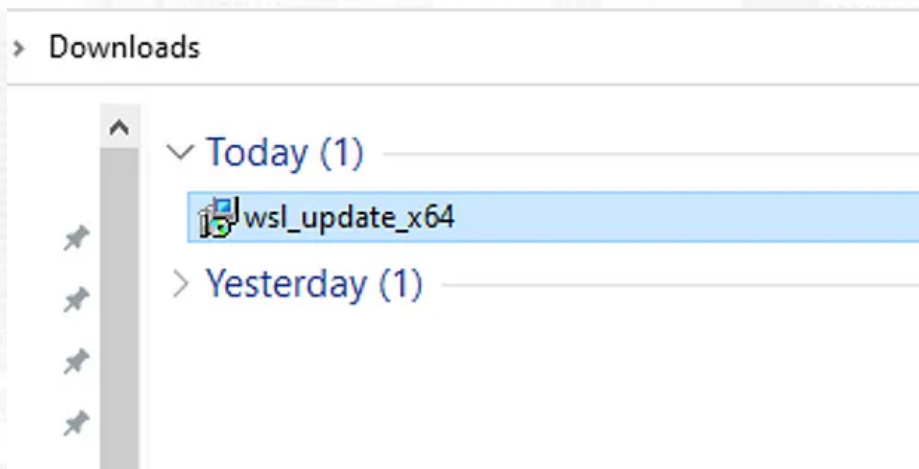
If you're using an ARM64 machine, please download the [ARM64 package](#) instead. If you're not sure what kind of machine you have, open Command Prompt or PowerShell and enter: `systeminfo | find "System Type"`. **Caveat:** On non-English Windows versions, you might have to modify the search text, translating the "System Type" string. You may also need to escape the quotations for the find command. For example, in German `systeminfo | find '"Systemtyp"'`.

2. Run the update package downloaded in the previous step. (Double-click to run - you will be prompted for elevated permissions, select 'yes' to approve this installation.)

Once the installation is complete, move on to the next step - setting WSL 2 as your default version when installing new Linux distributions. (Skip this step if you want your new Linux installs to be set to WSL 1).

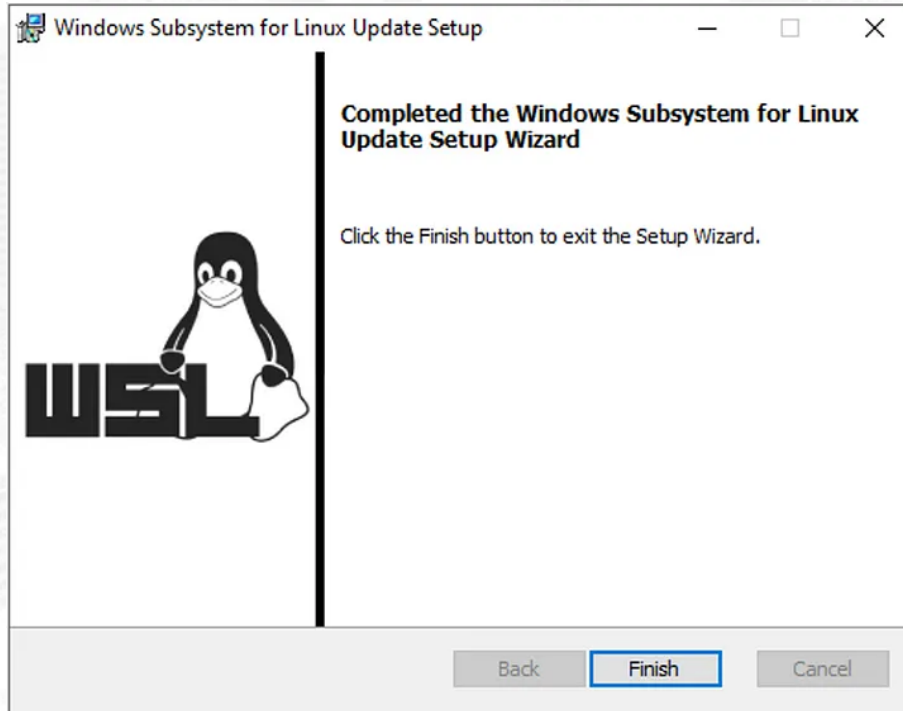
# Install Docker in Windows 10

9. Kemudian jalankan installer update WSL dan klik next



# Install Docker in Windows 10

10. Kemudian klik **Finish** dan **Restart**



# Install Docker in Windows 10

10. Jalankan perintah `docker run hello-world` pada terminal Shell Windows

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\valentin> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:80f31da1ac7b312ba29d65080fddf797dd76acfb870e677f390d5acba9741b17
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

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

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

# Install

- Docker support untuk berbagai macam OS
- Untuk cara installnya dapat disesuaikan dengan OS yang digunakan
- Untuk install Docker Desktop pada OS MacOS  
<https://docs.docker.com/desktop/install/mac-install/>
- Untuk install Docker pada OS Linux (Ubuntu)  
<https://docs.docker.com/engine/install/ubuntu/>



# Basic Docker

☐ Introduction

☐ Architecture

☐ Install

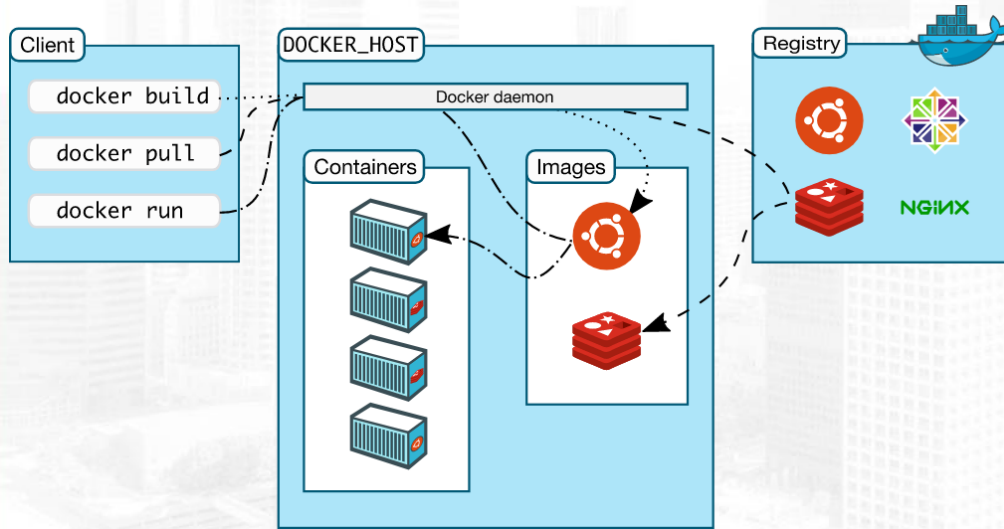


**Docker Registry**

☐ Docker Image

☐ Docker Container

# Docker Registry



Docker Registry digunakan untuk menyimpan Docker Image yang telah dibuat dan dapat digunakan pada Docker Daemon jika Docker Image itu terkoneksi ke Docker Registry

# Docker Registry

Dibawah ini terdapat beberapa Docker Registry yang dapat digunakan

- Docker Hub <https://hub.docker.com/>
- Google Cloud <https://cloud.google.com/container-registry>
- Amazon Cloud <https://aws.amazon.com/ecr/>

# Basic Docker

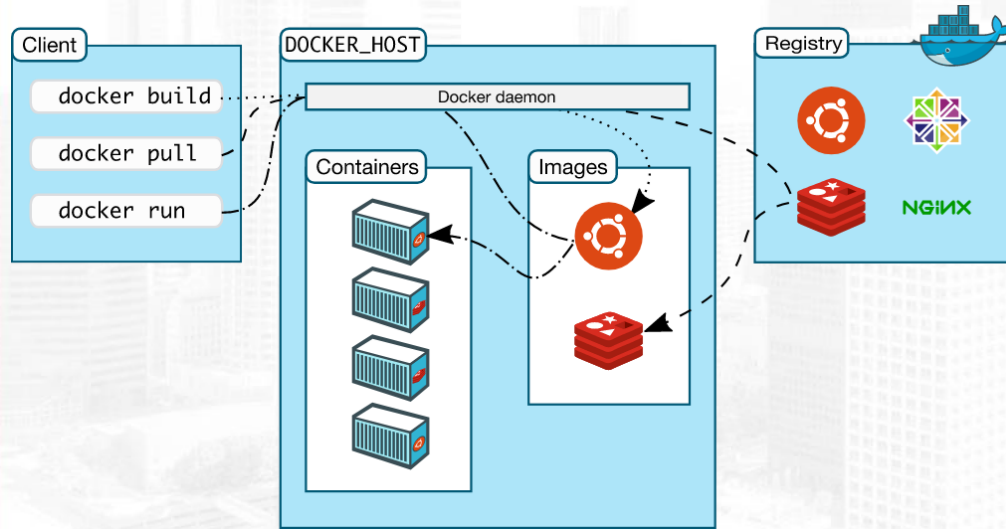
- ☐ Introduction
- ☐ Architecture
- ☐ Install
- ☐ Docker Registry



## **Docker Image**

- ☐ Docker Container

# Docker Image



Docker Image merupakan installer yang di dalam nya terdapat aplikasi dan dependency yang dapat digunakan

# Docker Image - List

Untuk melihat list image pada docker yang digunakan, kita dapat menjalankan perintah

`docker image ls`

Atau bisa juga digunakan untuk melihat image yang sudah terinstall dengan menggunakan perintah

`docker images imagename` (Contoh : `docker images golang`)

```
→ ~ docker images golang
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
golang	1.18-alpine3.17	f37f1bcb329f	2 months ago	330MB
golang	1.17-alpine3.15	675faa0f5534	6 months ago	314MB
golang	latest	65375c930b21	10 months ago	964MB



# Docker Image - Download

Untuk download docker image dapat menggunakan perintah

```
docker image pull imagename:tag
```

Untuk mendownload docker image yang akan kita gunakan dapat di akses di <https://hub.docker.com/> atau <https://hub.docker.com/search?q=>

Contoh

```
→ ~ docker image pull php
Using default tag: latest
latest: Pulling from library/php
```

# Docker Image - Remove

Untuk menghapus docker image dapat menggunakan perintah

```
docker rm imagename:tag
```

Contoh

```
→ ~ docker image rm php
Untagged: php:latest
Untagged: php@sha256:708663eb3ba7e5553f23dd3816e7c7a640cb6d9af1f20c26a1495a3693567b3f
Deleted: sha256:43fca8d539d459aa1301eca41ddb1a9281ee4cc96ed54ed9611c70298d3684ee
Deleted: sha256:b75d3bfd0f732f0912635eebdfc8d61f579f9715548a5957b500e27bf57bc85c
```

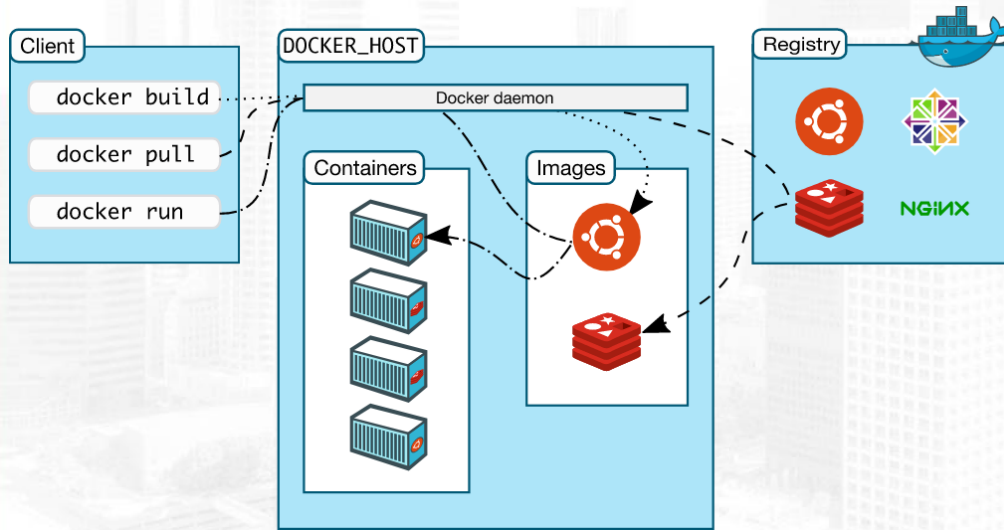
# Basic Docker

- ☐ Introduction
- ☐ Architecture
- ☐ Install
- ☐ Docker Registry
- ☐ Docker Image



**Docker Container**

# Docker Container



- Docker Container merupakan hasil installer yang dapat digunakan
- Sedangkan Docker Image yang dipelajari sebelumnya merupakan installernya
- 1 Docker Image dapat digunakan untuk beberapa Docker Container selama nama Docker Containernya berbeda

# Docker Container - List

Untuk melihat list container pada docker yang digunakan, kita dapat menjalankan perintah

```
docker container ls
```

```
→ ~ docker container ls
```

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

# Docker Container - Create

Untuk membuat container pada docker yang digunakan, kita dapat menjalankan perintah

```
docker container create --name containername imagename:tag
```

```
→ ~ docker container create --name testingphp php
Unable to find image 'php:latest' locally
latest: Pulling from library/php
bb263680fed1: Downloading [=====] 15.69MB/31.41MB
```



# Docker Container - Running

Untuk membuat container pada docker yang digunakan, kita dapat menjalankan perintah

`docker container start containerid / containername`

```

$ docker container start testingphp
testingphp
$ ~
  
```

# Docker Container - Stop

Untuk membuat container pada docker yang digunakan, kita dapat menjalankan perintah

`docker container stop containerid / containername`

```
→ ~ docker container stop testingphp  
testingphp  
→ ~ █
```

# Docker Container - Remove

Untuk membuat container pada docker yang digunakan, kita dapat menjalankan perintah

`docker container rm containerid / containername`

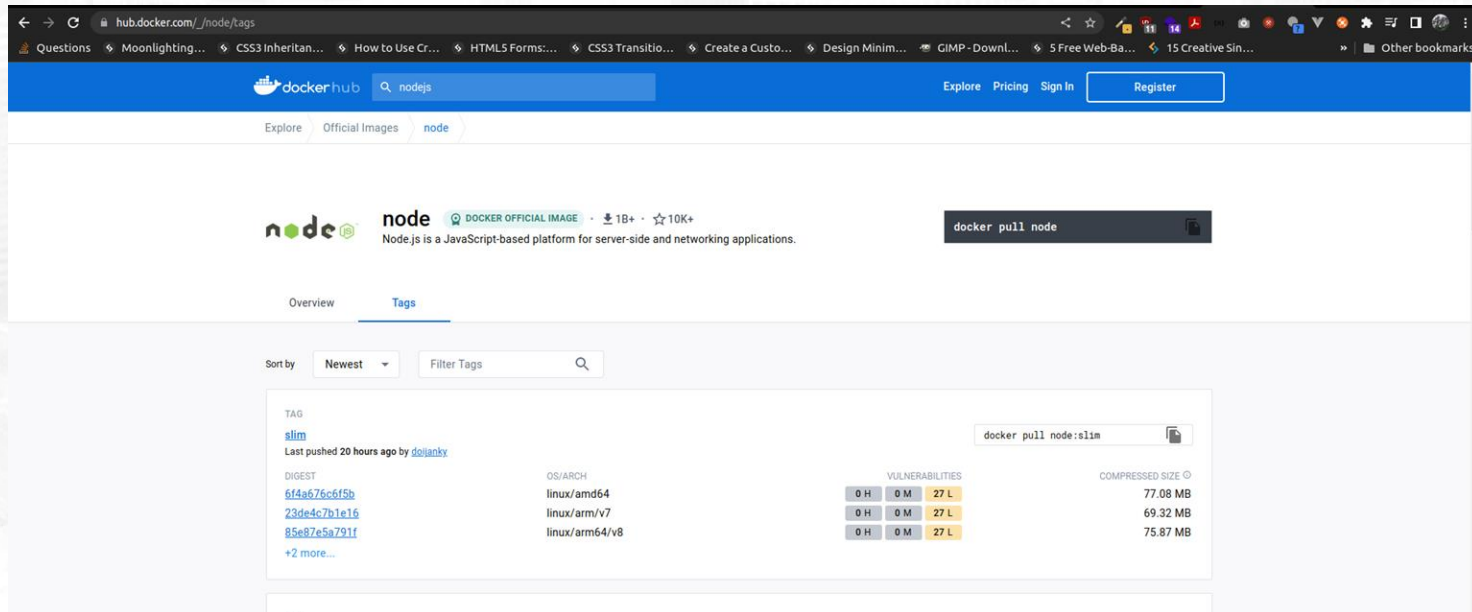
```
→ ~ docker container rm testingphp  
testingphp  
→ ~ █
```

# Study Case

1. Buatlah container dengan nama learning-docker
2. Kemudian buatlah image menggunakan nodejs dengan versi terbaru
3. Kemudian jalankan container tersebut

# Study Case Explanation

1. Untuk mencari image nodejs terbaru dapat diakses di [docker hub](https://hub.docker.com/_/node)



The screenshot shows the Docker Hub interface for the 'node' image. The page includes a search bar, navigation links, and a 'Tags' section. The 'slim' tag is highlighted, showing it was pushed 20 hours ago by 'dolanv'. Below the tag, there is a table of details for the 'linux/amd64' architecture, including the digest, vulnerabilities, and compressed size.

TAG	OS/ARCH	VULNERABILITIES	COMPRESSED SIZE
<a href="#">slim</a> Last pushed 20 hours ago by <a href="#">dolanv</a>	linux/amd64	0 H 0 M 27 L	77.08 MB
<a href="#">6f4a676c6f5b</a>	linux/arm/v7	0 H 0 M 27 L	69.32 MB
<a href="#">23de4c7b1e16</a>	linux/arm64/v8	0 H 0 M 27 L	75.87 MB
<a href="#">85e87e5a791f</a>			
<a href="#">+2 more...</a>			

# Study Case Explanation

2. Kemudian jalankan perintah untuk membuat container dengan nama learning-docker dan pull versi node terbaru

```
→ ~ docker container create --name learning-docker node
Unable to find image 'node:latest' locally
latest: Pulling from library/node
32fb02163b6b: Pull complete
167c7feebee8: Pull complete
d6dfff1f6f3d: Pull complete
e9cdcd4942eb: Pull complete
ca3bce705f6c: Pull complete
4f4cf292bc62: Pull complete
8347f8b4b86b: Pull complete
c5f20f1b0856: Pull complete
d220dfa3e187: Pull complete
Digest: sha256:83841d113e09345a28b146e431f15b062341c5449218e501ba45ef8f9cff6049
Status: Downloaded newer image for node:latest
43f305ae87f0f99baf8692f2a180310f5be170fb0a73915d6b94018034a646a9
```



# Study Case Explanation

2. Dan jalankan docker container start untuk mengaktifkan docker container

## Activities

```
→ ~ docker container stats learning-docker
```

# Basic Docker



**Introduction**



**Architecture**



**Install**



**Docker Registry**



**Docker Image**



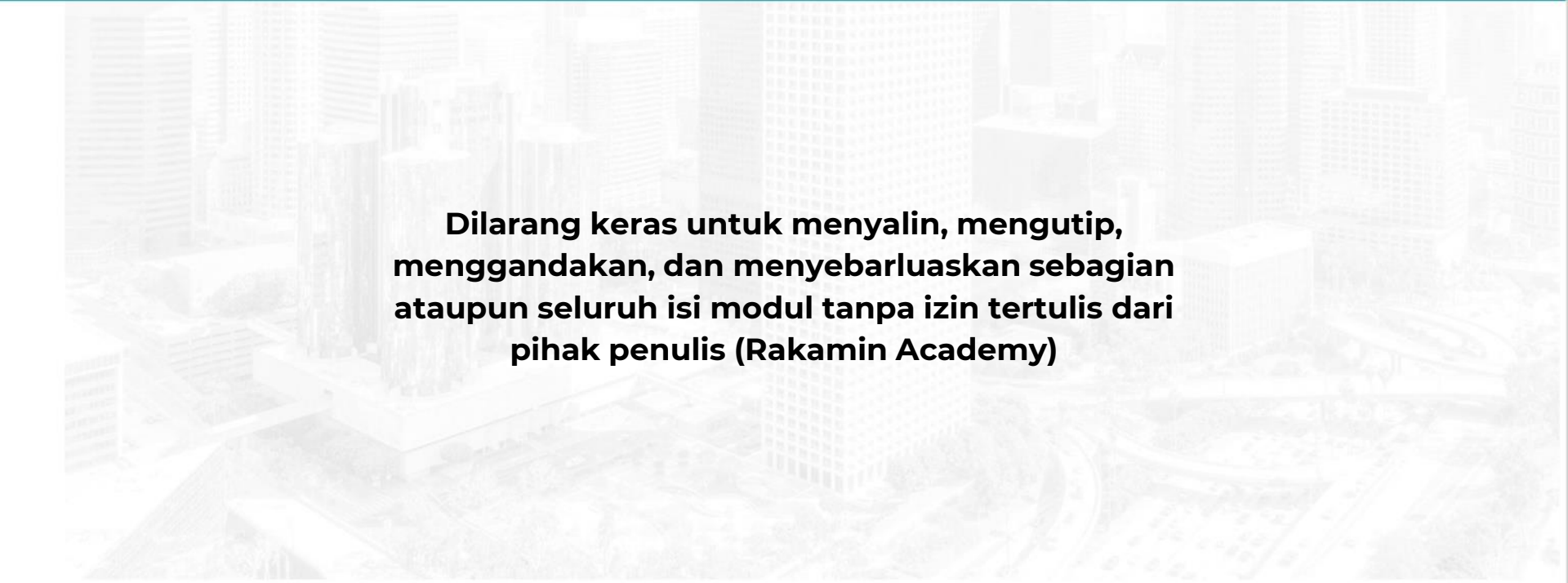
**Docker Container**

# Reference material

1. [Install Docker](#)
2. [Basic Using Docker](#)
3. [Docker Commands](#)
4. [Docker Manipulation Image](#)
5. [Programmer Zaman Now](#)

**Terima kasih!**

# Copyright Rakamin Academy

A faded, light-colored background image of a city skyline with various skyscrapers and buildings.

**Dilarang keras untuk menyalin, mengutip,  
menggandakan, dan menyebarkan sebagian  
ataupun seluruh isi modul tanpa izin tertulis dari  
pihak penulis (Rakamin Academy)**

# Temmy Alex

Programming Enthusiast

Focus in Web Development

Currently work in KoinWorks as Backend Engineer



**Temmy Alex**

8 years experience as Web Developer

## Education Background



**UNIVERSITAS  
BUDI LUHUR**

**2011-2015**

**Bachelor Degree**

Information System



# Dockerfile

- ☐ Introduction
- ☐ Format
- ☐ Instruction / Commands

# Dockerfile

- Mengetahui tentang apa itu Dockerfile
- Mengetahui tentang format-format yang digunakan pada Dockerfile
- Mengetahui tentang apa saja instruksi atau perintah yang digunakan pada Dockerfile

# Objektif Sesi

- Student memahami tentang apa itu Dockerfile
- Student memahami tentang format-format yang digunakan pada Dockerfile
- Student memahami tentang intruksi atau perintah yang dapat digunakan pada Dockerfile

# Dockerfile



## Introduction



Format



Instruction / Commands

# Apa itu Dockerfile?

Dockerfile merupakan file text yang berisi perintah yang dapat digunakan untuk membuat Docker Image dengan menyimpan perintah pada Dockerfile perintah tersebut dapat digunakan ulang untuk membuat Docker Image

# Docker Build

Untuk membuat Docker Image melalui Dockerfile kita dapat menjalankan perintah

Menggunakan tag name untuk container pada docker build dengan menggunakan -t

```
docker build -t rakamin/api-todo
```

Atau build current directory (posisi saat didalam root project)

```
docker build .
```



# Dockerfile

☐ Introduction



**Format**

☐ Instruction / Commands

# Dockerfile Format

Tidak ada extension khusus dalam membuat Dockerfile cukup membuat nama file dengan nama Dockerfile kemudian tambahkan beberapa instruksi yang akan digunakan dan Dockerfile merupakan nama yang direkomendasikan untuk membuat Dockerfile

# Dockerfile Format

Untuk penulisan perintah terdapat konvensi yang dianjurkan yaitu dengan menggunakan huruf besar seperti pada contoh dibawah ini beberapa perintah yang sering digunakan dalam Dockerfile seperti RUN, FROM, WORKDIR, CMD atau ENTRYPOINT

# Dockerfile

☐ Introduction

☐ Format



**Instruction / Commands**

# Dockerfile - Comment Instruction

Untuk melakukan komentar pada perintah Dockerfile kita dapat menggunakan tanda # yang sama halnya memiliki fungsi komentar pada bahasa pemrograman sehingga perintah tersebut akan diabaikan atau tidak dijalankan dan dapat berfungsi sebagai keterangan

Contoh : # FROM ubuntu

# Dockerfile - FROM Instruction

FROM merupakan perintah yang digunakan untuk membuat base baik itu sistem operasi maupun bahasa pemrograman

Untuk format penulisan dengan versi

```
FROM image:version
```

Untuk format penulisan tanpa versi

```
FROM image
```

Dan untuk membuat instruksi bisa menggunakan contoh-contoh docker image yang sudah ada kemudian tinggal disesuaikan dengan kebutuhan



# Dockerfile - FROM Instruction

Menggunakan Versi

```
Dockerfile X
Dockerfile > ...
1 FROM node:18-alpine
```

Tanpa Menggunakan Versi

```
Dockerfile X
Dockerfile > ...
1 FROM node
```

# Dockerfile - Build Instruction

Perintah build digunakan untuk membangun sebuah Docker Image untuk penggunaannya dapat menggunakan perintah berikut

```
docker build -t imagename destinationpath
```

```
● → docker-example docker build -t rakamin/docker-example .  
Sending build context to Docker daemon 2.56kB  
Step 1/1 : FROM node:18-alpine  
18-alpine: Pulling from library/node  
Digest: sha256:0d2712ac2b2c1149391173de670406f6e3dbdb1b2ba44e8530647e623e0e1b17  
Status: Downloaded newer image for node:18-alpine  
---> 9423415aa47a  
Successfully built 9423415aa47a  
Successfully tagged rakamin/docker-example:latest
```

# Dockerfile - Build Instruction

Perintah build digunakan untuk membangun sebuah Docker Image untuk penggunaannya dapat menggunakan perintah berikut

```
docker build -t imagename destinationpath
```


```
● → docker-example docker build -t rakamin/docker-example .  
Sending build context to Docker daemon 2.56kB  
Step 1/1 : FROM node:18-alpine  
18-alpine: Pulling from library/node  
Digest: sha256:0d2712ac2b2c1149391173de670406f6e3dbdb1b2ba44e8530647e623e0e1b17  
Status: Downloaded newer image for node:18-alpine  
---> 9423415aa47a  
Successfully built 9423415aa47a  
Successfully tagged rakamin/docker-example:latest
```


# Dockerfile - RUN Instruction

Perintah RUN digunakan untuk melakukan eksekusi perintah yang terdapat dalam docker image dan perintah tersebut akan di jalan ketika proses docker build

Terdapat 2 argumen yang digunakan pada perintah RUN

RUN[ “ executable ” , “ argument ” ]

 Dockerfile X

 Dockerfile > ...

```
1 FROM node:18-alpine
2
3 RUN mkdir docker-example
4 RUN echo "Docker Example" > docker-example.txt
5 RUN cat docker-example.txt
```

# Dockerfile - RUN Instruction

Untuk implementasi nya pada Dockerfile maka bisa jalankan lagi perintah build nya

```
• → docker-example docker build .  
Sending build context to Docker daemon 2.56kB  
Step 1/4 : FROM node:18-alpine  
---> 9423415aa47a  
Step 2/4 : RUN mkdir docker-example  
---> Running in da7274e09f55  
Removing intermediate container da7274e09f55  
---> 3990fbb86c50  
Step 3/4 : RUN echo "Docker Example" > docker-example.txt  
---> Running in cde95e8d92a6  
Removing intermediate container cde95e8d92a6  
---> 928c5a5f2c58  
Step 4/4 : RUN cat docker-example.txt  
---> Running in 47b062ef2c2a  
Docker Example  
Removing intermediate container 47b062ef2c2a  
---> 1c9d671b732c  
Successfully built 1c9d671b732c
```

# Dockerfile - COPY Instruction

Perintah COPY digunakan untuk menambahkan file / folder dari local source ke dalam folder Docker Image

Untuk format penulisan perintah COPY

COPY source destination

Contoh : COPY package\*.json ./ (Copy 1 file package.json termasuk package-lock.json ke directory docker image)

Kemudian jalankan perintah docker build untuk melakukan update perintah pada Dockerfile



# Dockerfile - COPY Instruction

```
Dockerfile X
Dockerfile > ...
1 FROM node:18-alpine
2
3 COPY package*.json ./
4
```

```
• → docker-example docker build .
Sending build context to Docker daemon 4.608kB
Step 1/2 : FROM node:18-alpine
---> 9423415aa47a
Step 2/2 : COPY package*.json ./
---> 13b48960fa04
Successfully built 13b48960fa04
```

# Dockerfile - CMD Instruction

CMD merupakan perintah digunakan untuk menjalankan instruksi pada Docker Container

Dan hanya dapat digunakan untuk 1 CMD di Dockerfile / Tidak bisa menjalankan lebih dari 1 CMD dalam 1 Dockerfile

Dockerfile X

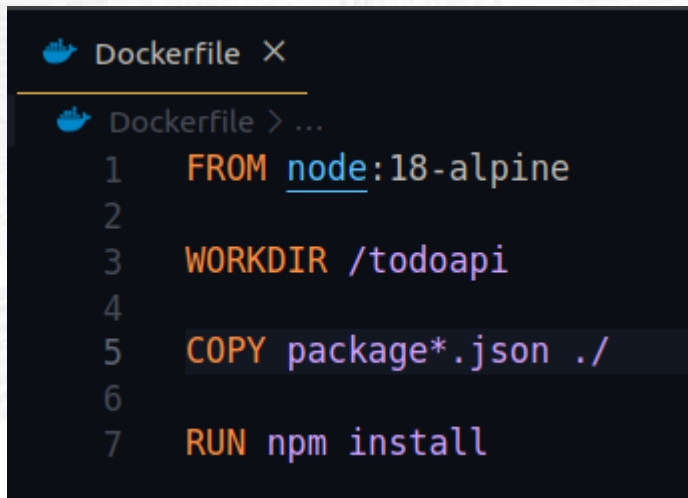
Dockerfile > ...

```
1 FROM alpine
2
3 RUN mkdir docker-example
4 RUN echo "Docker Example" > docker-example.txt
5
6 CMD cat "docker-example.txt"
```

# Dockerfile - Workdir Instruction

WORKDIR adalah instruksi yang digunakan untuk menentukan direktori / folder yang digunakan menjalankan instruksi RUN, CMD, ENTRYPOINT dll

Jika WORKDIR tidak ditentukan pada Dockerfile maka secara otomatis akan dibuat melalui Docker Compiler



```

Dockerfile X
Dockerfile > ...
1 FROM node:18-alpine
2
3 WORKDIR /todoapi
4
5 COPY package*.json ./
6
7 RUN npm install
  
```

# Study Case

Buatlah sebuah Dockerfile yang didalamnya terdapat perintah untuk

1. Menentukan direktori yang akan digunakan docker
2. Kemudian untuk sistem operasi yang digunakan adalah nodejs terbaru
3. Kemudian pindahkan seluruh package.json pada project local kalian dan lakukan instalasi menggunakan perintah npm

# Study Case Explained

4. Buatlah file bernama Dockerfile dan isikan perintah seperti pada gambar dibawah

```

Dockerfile X
Dockerfile > ...
1  FROM node:18.14.2-alpine3.17
2
3  WORKDIR /run-nodejs-dockerfile/src/app
4
5  COPY package*.json ./
6
7  RUN npm install
8
9  COPY . .
10
11 EXPOSE 8080
12
13 CMD [ "node", "app.js" ]

```

# Study Case Explained

1. Kemudian jalankan perintah docker build .

```
● → run-nodejs-dockerfile git:(master) docker build .  
Sending build context to Docker daemon 2.485MB  
Step 1/7 : FROM node:18.14.2-alpine3.17  
--> 9423415aa47a  
Step 2/7 : WORKDIR /run-nodejs-dockerfile/src/app  
--> Using cache  
--> 940f7d6d03d7  
Step 3/7 : COPY package*.json ./  
--> Using cache  
--> fde234a57c95  
Step 4/7 : RUN npm install  
--> Using cache  
--> e4d7ba4b3a94  
Step 5/7 : COPY . .  
--> ac49cdc7678a  
Step 6/7 : EXPOSE 8080  
--> Running in 90935c05a278  
Removing intermediate container 90935c05a278  
--> 38116c18833c  
Step 7/7 : CMD [ "node", "app.js" ]  
--> Running in 0f23402d29fa  
Removing intermediate container 0f23402d29fa  
--> 612438693321  
Successfully built 612438693321
```



# Dockerfile



**Introduction**



**Format**



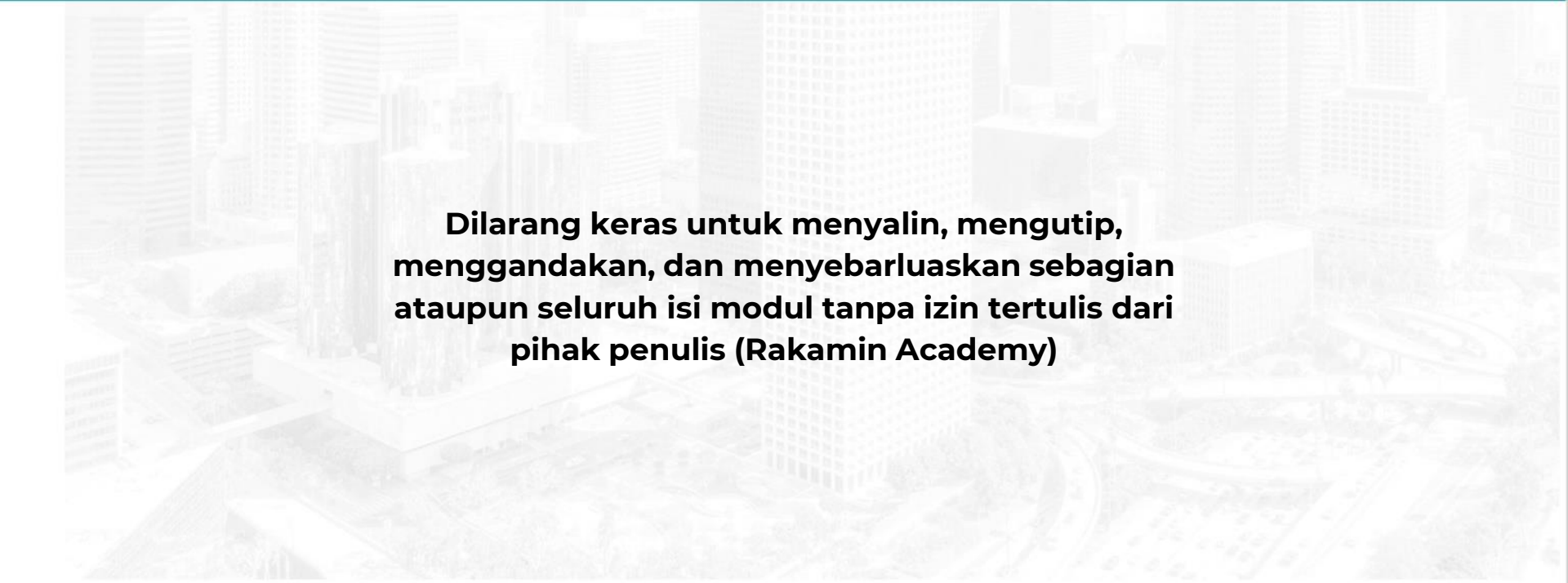
**Instruction / Commands**

# Reference material

1. [Install Docker](#)
2. [Basic Using Docker](#)
3. [Docker Commands](#)
4. [Docker Manipulation Image](#)
5. [Programmer Zaman Now](#)

**Terima kasih!**

# Copyright Rakamin Academy

A faded, light-colored background image of a city skyline with various skyscrapers and buildings.

**Dilarang keras untuk menyalin, mengutip,  
menggandakan, dan menyebarkan sebagian  
ataupun seluruh isi modul tanpa izin tertulis dari  
pihak penulis (Rakamin Academy)**

# Temmy Alex

Programming Enthusiast

Focus in Web Development

Currently work in KoinWorks as Backend Engineer



**Temmy Alex**

8 years experience as Web  
Developer

## Education Background



**UNIVERSITAS  
BUDI LUHUR**

**2011-2015**

**Bachelor Degree**

Information System

# Docker Compose

- ☐ Introduction
- ☐ Configuration
- ☐ Implementation



# Docker Compose

- Mengetahui tentang apa itu docker compose
- Mengetahui cara konfigurasi docker compose
- Mengetahui cara menggunakan docker compose

# Objektif Sesi

- Student memahami tentang docker compose
- Student memahami cara konfigurasi docker compose
- Student memahami cara menggunakan docker compose

# Docker Compose



## Introduction



Configuration



Implementation

# Apa itu Docker Compose?

Docker Compose merupakan tool yang digunakan untuk menjalankan multiple Docker Container sekaligus dan untuk konfigurasi nya dapat di setting menggunakan format yaml / yml

# Mengapa harus menggunakan Docker Compose?

Docker Compose cocok digunakan untuk membuat development environment yang berbeda untuk setiap project

Dan dapat digunakan untuk proses deployment aplikasi

# Docker Compose

☐ Introduction



**Configuration**



Implementation



# Install Docker Compose

Untuk menggunakannya kita dapat menjalankan perintah

`docker compose`

Untuk mengecek versi docker compose kita dapat menjalankan perintah

`docker compose version`

```
→ ~ docker compose version  
Docker Compose version v2.12.0
```

# Introduction YAML

1. YAML file digunakan untuk menyimpan konfigurasi docker compose
2. Memiliki format seperti JSON namun tidak menggunakan kurung kurawal
3. Memiliki attribute dan value
4. Referensi <https://yaml.org/>

# Format YAML

data.yaml X

data.yaml
1 name: User
2 email: user@mail.com
3
4

data.json

data.json > email
1 {
2 "name": "User",
3 "email": "user@mail.com"
4 }

# Format YAML - Nested Object

data.yaml

1 name: User  
2 email: user@mail.com  
3 hobbies:  
4 - Coding  
5 - Music

data.json

data.json > [ ] hobbies  
1 {  
2 "name": "User",  
3 "email": "user@mail.com",  
4 "hobbies": [  
5 "Coding",  
6 "Music"  
7 ]  
8 }

# Configuration Using YAML

1. Untuk setting konfigurasi docker compose, file yang digunakan memiliki ekstension `.yaml / .yml`
2. YAML memiliki struktur yang hampir sama dengan JSON dan lebih sederhana
3. Dan disimpan dalam bentuk `docker-compose.yaml`
4. Biasanya file docker compose berada dalam root folder project

# Docker Compose

☐ Introduction

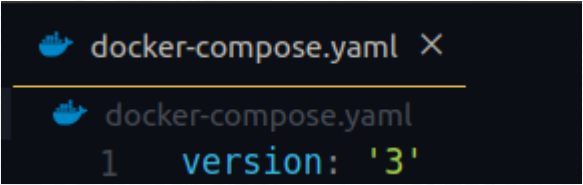
☐ Configuration



**Implementation**

# Versi Docker Compose

1. Versi pada file perlu di set ketika awal menggunakan docker compose
2. Salah satu guideline untuk [versi 3](#)



```
🐳 docker-compose.yml ✕
```

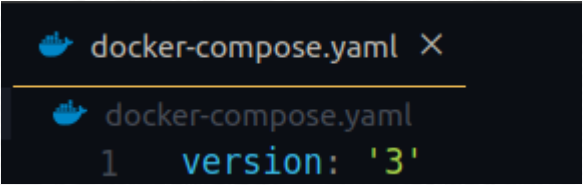
```
🐳 docker-compose.yml
```

```
1 version: '3'
```



# Versi Docker Compose

1. Versi pada file perlu di set ketika awal menggunakan docker compose
2. Salah satu guideline yang dapat digunakan yaitu menggunakan [versi 3](#)



```
🐳 docker-compose.yml ✕
```

```
🐳 docker-compose.yml
1  version: '3'
```

# Service Docker Compose

```

docker-compose.yml
1  version: '3'
2  services:
3    postgres:
4      container_name: todoapi-db
5      image: postgres
6      ports:
7        - 6548:5432
8      environment:
9        POSTGRES_USER: postgres
10       POSTGRES_PASSWORD: root
11       POSTGRES_DB: todoapi-db
  
```

1. File dibuat menggunakan format yaml dan docker-compose.yml merupakan nama yang direkomendasikan untuk digunakan
2. Kemudian tentukan nama services, nama services ini bebas sesuai dengan kebutuhan (Contoh : postgres)
3. Container name, yang dapat digunakan untuk set docker compose
4. Ports merupakan perintah yang digunakan untuk menjalankan ports yang digunakan pada container docker compose
5. Environment merupakan set variable yang digunakan pada docker compose

# Service Docker Compose

Untuk menggunakan docker compose kita bisa menjalankan perintah

`docker-compose up`

```

o → todoapi docker-compose up
Creating network "todoapi_default" with the default driver
Creating todoapi-db ... done
Attaching to todoapi-db
todoapi-db | The files belonging to this database system will be owned by user "postgres".
todoapi-db | This user must also own the server process.
todoapi-db |
todoapi-db | The database cluster will be initialized with locale "en_US.utf8".
todoapi-db | The default database encoding has accordingly been set to "UTF8".
todoapi-db | The default text search configuration will be set to "english".
todoapi-db |
todoapi-db | Data page checksums are disabled.
todoapi-db |
todoapi-db | fixing permissions on existing directory /var/lib/postgresql/data ... ok
todoapi-db | creating subdirectories ... ok
todoapi-db | selecting dynamic shared memory implementation ... posix
todoapi-db | selecting default max_connections ... 100
todoapi-db | selecting default shared_buffers ... 128MB
todoapi-db | selecting default time zone ... Etc/UTC
todoapi-db | creating configuration files ... ok

```

# Service Docker Compose

Untuk menjalankan docker compose secara background cukup tambahkan `-d` kita bisa menjalankan perintah

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

Dan untuk menghentikan service bisa jalankan perintah

```
docker-compose down
```

# Study Case

1. Buatlah service docker compose yang menjalankan service nodejs menggunakan Dockerfile dan docker-compose.yml
2. Kemudian jalankan docker compose tersebut pada port 8080

# Study Case Explained

1. Buatlah project api sederhana menggunakan nodejs untuk referensi bisa menggunakan repository berikut
2. Kemudian buatlah file bernama Dockerfile dan isikan instruksi [berikut](#)
3. Kemudian buatlah file bernama docker-compose.yaml dengan kode [berikut](#)



# Study Case Explained

4. Jalankan perintah docker build . untuk menjalankan Dockerfile

```
● → run-nodejs-dockerfile git:(master) docker build .  
Sending build context to Docker daemon 2.485MB  
Step 1/7 : FROM node:18.14.2-alpine3.17  
--> 9423415aa47a  
Step 2/7 : WORKDIR /run-nodejs-dockerfile/src/app  
--> Using cache  
--> 940f7d6d03d7  
Step 3/7 : COPY package*.json ./  
--> Using cache  
--> fde234a57c95  
Step 4/7 : RUN npm install  
--> Using cache  
--> e4d7ba4b3a94  
Step 5/7 : COPY . .  
--> ac49cdc7678a  
Step 6/7 : EXPOSE 8080  
--> Running in 90935c05a278  
Removing intermediate container 90935c05a278  
--> 38116c18833c  
Step 7/7 : CMD [ "node", "app.js" ]  
--> Running in 0f23402d29fa  
Removing intermediate container 0f23402d29fa  
--> 612438693321  
Successfully built 612438693321
```



# Study Case Explained

4. Jalankan perintah `docker compose up` untuk menjalankan api melalui docker-

```

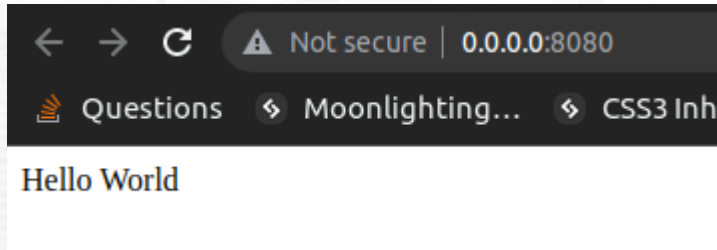
CO
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
added 57 packages, and audited 58 packages in 743ms

7 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
Removing intermediate container 92f5de471994
--> 21a3c956dce7
Step 5/7 : COPY . .
--> 9e0fda9eb273
Step 6/7 : EXPOSE 8080
--> Running in 593bca0116d6
Removing intermediate container 593bca0116d6
--> 1b33c117e1f8
Step 7/7 : CMD [ "node", "app.js" ]
--> Running in e2e446059942
Removing intermediate container e2e446059942
--> 28dc3d5fa966
Successfully built 28dc3d5fa966
Successfully tagged run-nodejs-dockerfile_todo-api-example:latest
WARNING: Image for service todo-api-example was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docker-compose up --build`.
Recreating run-nodejs-dockerfile_todo-api-example-1 ... done
Attaching to run-nodejs-dockerfile_todo-api-example_1
todo-api-example_1 | Running on http://0.0.0.0:8080
  
```

# Study Case Explained

5. Lakukan pengecekan api yang sudah dijalankan oleh docker melalui browser / postman



Api berhasil ditampilkan

# Docker Compose



**Introduction**



**Configuration**



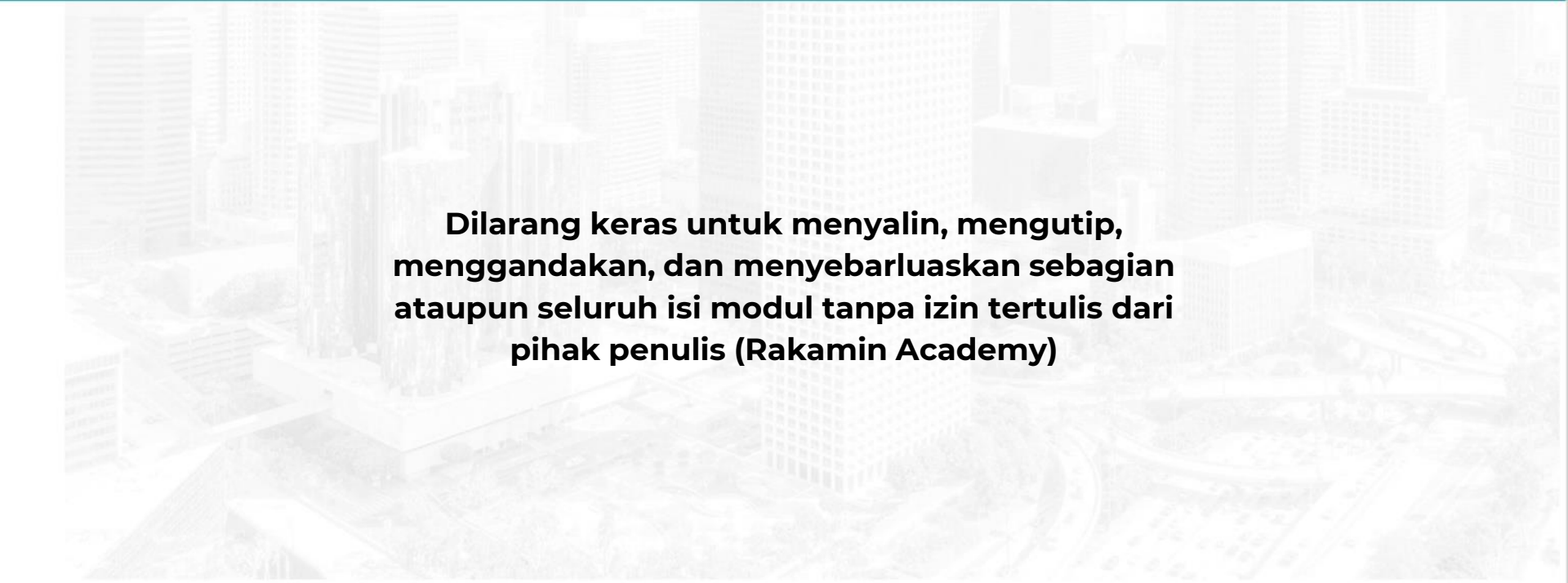
**Implementation**

# Reference material

1. [Install Docker](#)
2. [Basic Using Docker](#)
3. [Docker Commands](#)
4. [Docker Manipulation Image](#)
5. [Programmer Zaman Now](#)

**Terima kasih!**

# Copyright Rakamin Academy

A faded, light-colored background image of a city skyline with various skyscrapers and buildings.

**Dilarang keras untuk menyalin, mengutip,  
menggandakan, dan menyebarkan sebagian  
ataupun seluruh isi modul tanpa izin tertulis dari  
pihak penulis (Rakamin Academy)**