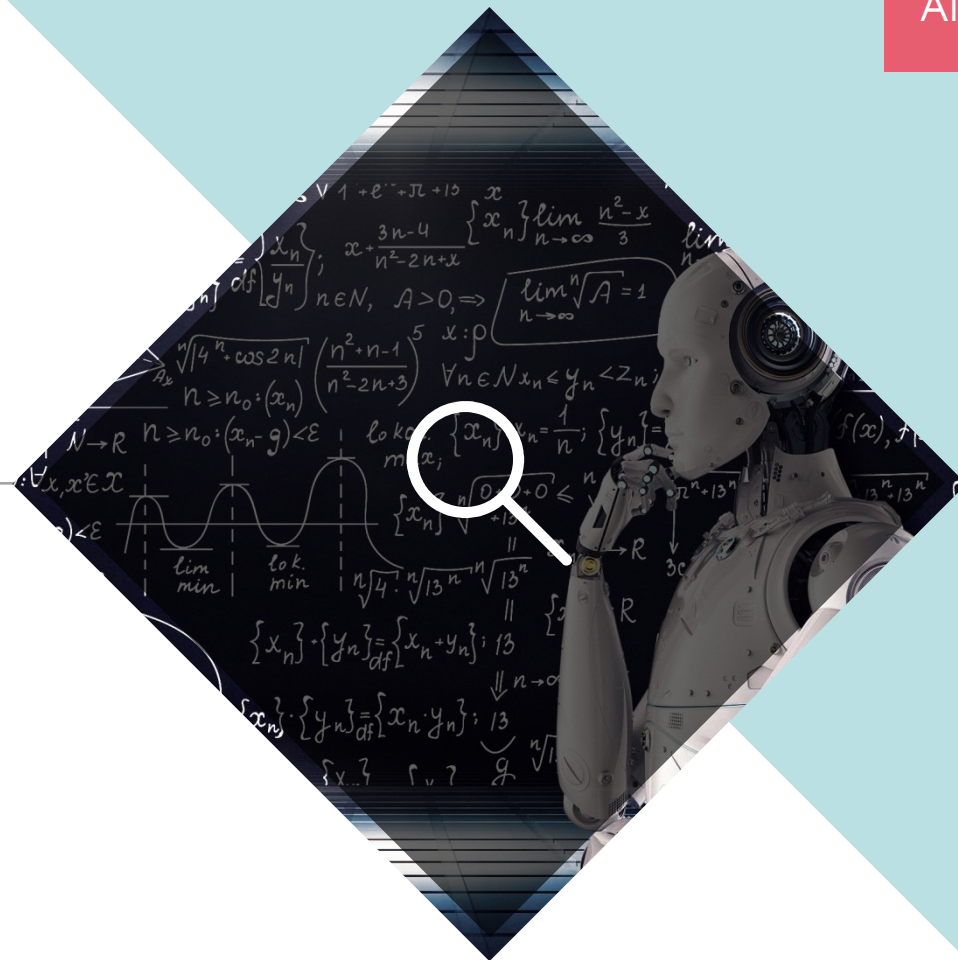
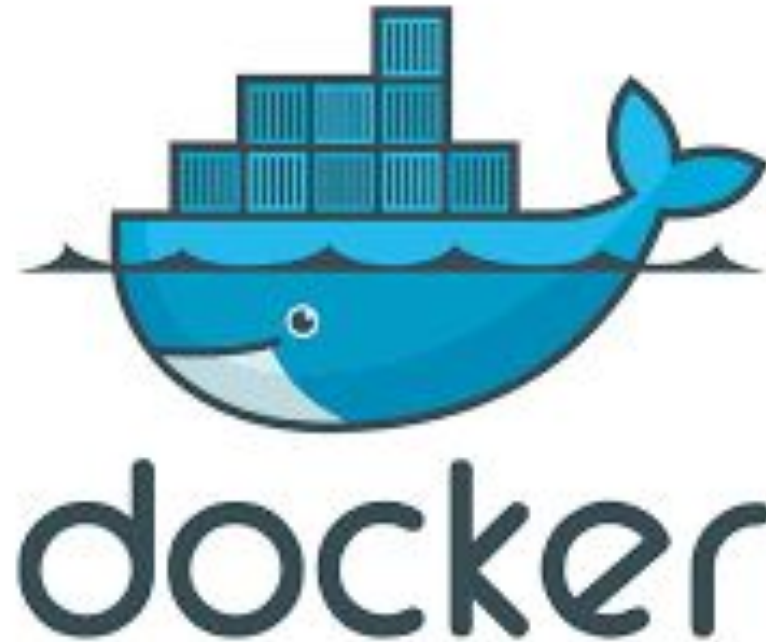


Docker 실습



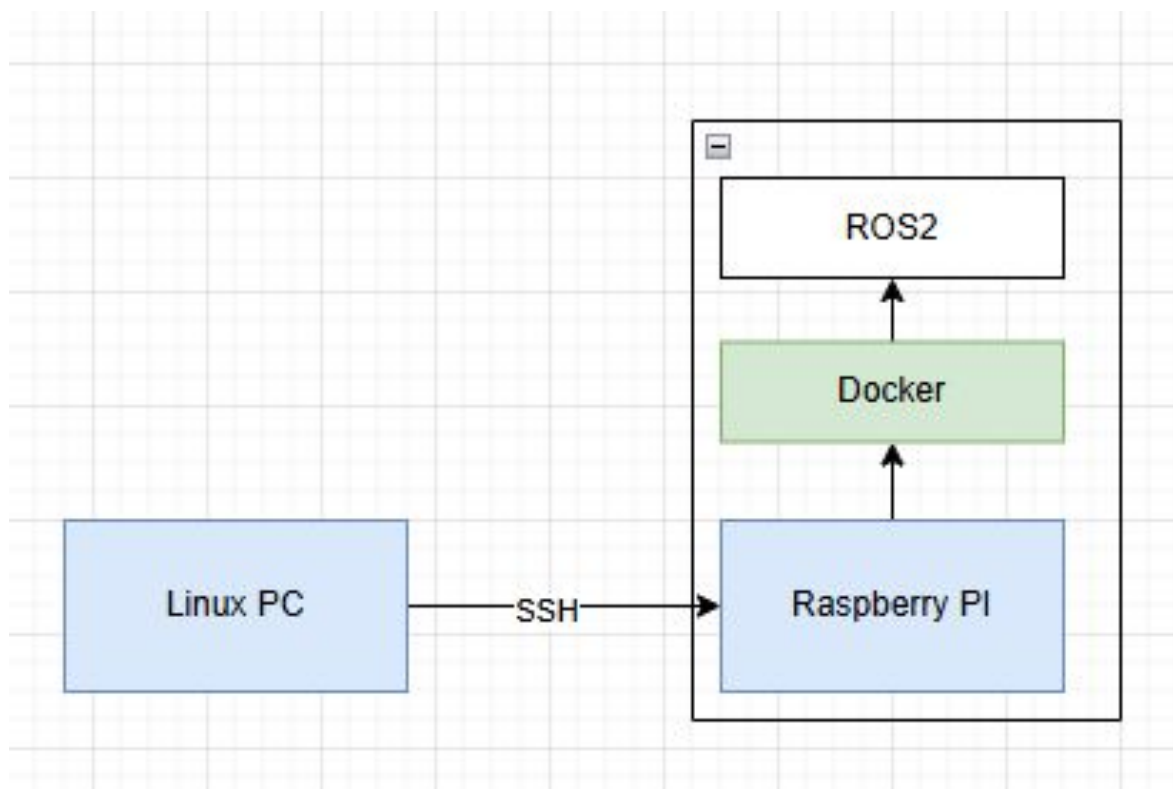
Contents

- I. 사전 준비 및 환경 확인
- II. Docker 설치
 - Docker 설치
 - Docker 권한 설정
- III. Docker 이미지 실습
 - Dockerfile 작성
 - 이미지 빌드
 - 이미지 실행
- IV. Docker 버전 관리
 - Docker Commit



I. 사전 준비 및 환경 확인

리눅스 PC에서 SSH로 원격 접속을 한 후,
라즈베리파이에 Docker를 설치하고 ROS2 를 세팅해보겠습니다.



I. 사전 준비 및 환경 확인

라즈베리 파이 SSH 설정

sudo raspi-config -> Interface Options -> SSH -> enable

Raspberry Pi Software Configuration Tool (raspi-config)

- | | |
|----------------------------|---|
| 1 System Options | Configure system settings |
| 2 Display Options | Configure display settings |
| 3 Interface Options | Configure connections to peripherals |
| 4 Performance Options | Configure performance settings |
| 5 Localisation Options | Configure language and regional settings |
| 6 Advanced Options | Configure advanced settings |
| 8 Update | Update this tool to the latest version |
| 9 About raspi-config | Information about this configuration tool |

Raspberry Pi Software Configuration Tool (raspi-config)

- | | |
|----------------|--|
| I1 SSH | Enable/disable remote command line access using SSH |
| I2 RPi Connect | Enable/disable Raspberry Pi Connect |
| I3 VNC | Enable/disable graphical remote desktop access |
| I4 SPI | Enable/disable automatic loading of SPI kernel module |
| I5 I2C | Enable/disable automatic loading of I2C kernel module |
| I6 Serial Port | Enable/disable shell messages on the serial connection |
| I7 1-Wire | Enable/disable one-wire interface |
| I8 Remote GPIO | Enable/disable remote access to GPIO pins |

<Select>

<Back>

Would you like the SSH server to be enabled?

Caution: Default and weak passwords are a security risk when SSH is enabled!

<Yes>

<No>

I. 사전 준비 및 환경 확인

라즈베리 파이 SSH 설정

ifconfig 로 라즈베리파이의 IP 주소를 알아내기

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.2 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::9f7c:252d:f43c:f523 prefixlen 64 scopeid 0x20<link>
    ether e4:5f:01:9d:09:2f txqueuelen 1000 (Ethernet)
    RX packets 227259 bytes 299958092 (286.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 67503 bytes 5060610 (4.8 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
piduke@raspberrypi:~ $ |
```

I. 사전 준비 및 환경 확인

리눅스 터미널에서 SSH로 라즈베리파이에 접속

`ssh {your id}@{ip주소}`

- 이후 유저의 비밀번호 입력

예시 : `ssh piduke@192.168.0.2`

```
PS C:\Users\hcmx> ssh piduke@192.168.0.2
piduke@192.168.0.2's password:
```

```
Linux raspberrypi 6.12.20+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.12.20-1+rpt1~bpo12+1 (2025-03-19) aarch64
```

```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

```
Last login: Thu May 1 23:37:52 2025 from 192.168.0.11
```

```
piduke@raspberrypi:~ $
```

II. Docker 설치

[APT 업데이트]

```
sudo apt update && sudo apt upgrade -y
```

[폴더 생성 & 폴더 내부로 이동]

```
mkdir docker-tutorial && cd docker-tutorial
```

[도커 설치]

```
curl -fsSL https://get.docker.com -o get-docker.sh
```

```
sudo sh get-docker.sh
```

[도커 그룹 생성 & 내 계정에 그룹 추가]

```
sudo usermod -aG docker $USER
```

```
newgrp docker
```

```
sudo reboot
```

II. Docker 설치

Docker 소켓 권한 구조

- `/var/run/docker.sock`
 - 소유자: `root`
 - 그룹: `docker`
 - 권한: `srw-rw----`

주요 장점

- `sudo` 없이 Docker 실행
 - 그룹 멤버는 소켓 접근 가능 → `docker run ...`
- 권한 분리·관리 용이
 - 필요 사용자만 `docker` 그룹에 추가/제거
- 보안 강화
 - 소켓 접근 권한 = 호스트 제어 권한
 - 최소 권한 원칙 준수

II. Docker 설치

현재 로그인한 User가 Docker 그룹에 속해있는지 확인하는 방법

`groups | grep docker`

```
piduke@raspberrypi:~/docker-tutorial/sample $ groups | grep docker
piduke adm dialout cdrom sudo audio video plugdev games users input render netdev lpadmin docker gpio i2c spi
piduke@raspberrypi:~/docker-tutorial/sample $
```

Docker 그룹에 대한
정보

```
grep docker /etc/group
piduke@raspberrypi:~/docker-tutorial/sample $ grep docker /etc/group
docker:x:991:piduke
```

II. Docker 설치

`docker --version`

```
piduke@raspberrypi:~ $ docker --version
Docker version 28.1.1, build 4eba377
```

`docker run hello-world`

```
piduke@raspberrypi:~ $ docker run hello-world
```

Hello from Docker!

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

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(arm64v8)
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:

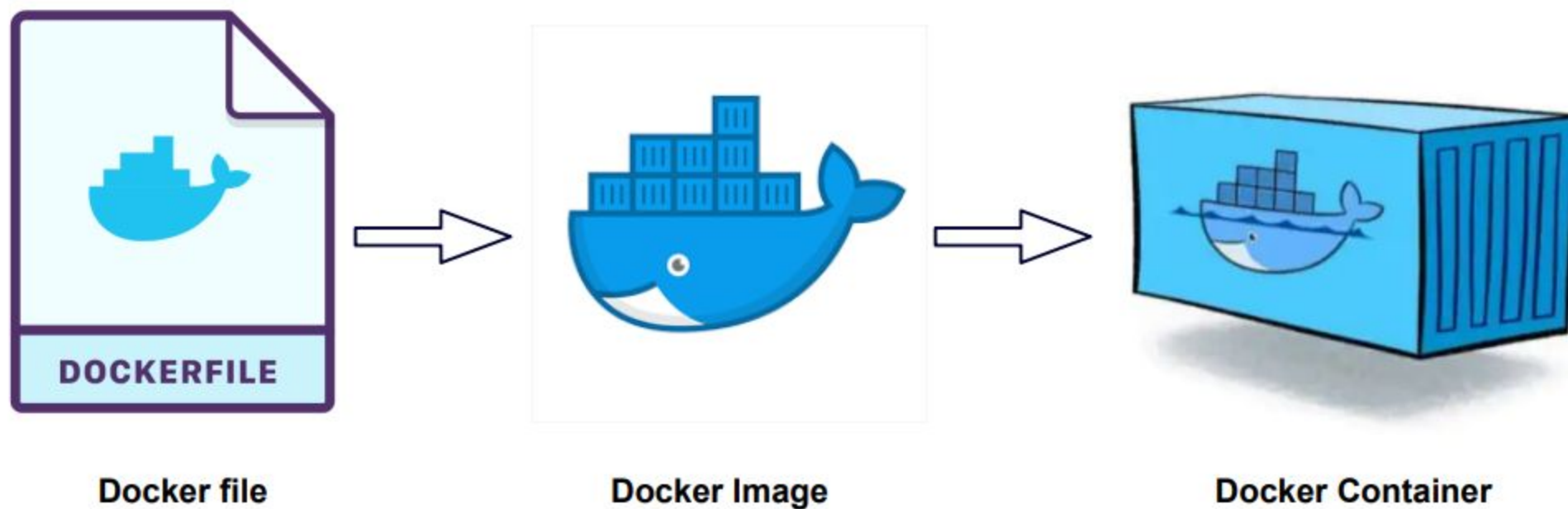
<https://hub.docker.com/>

For more examples and ideas, visit:

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

```
piduke@raspberrypi:~ $
```

III. Docker 이미지 실습



III. Docker 이미지 실습

1. Dockerfile을 작성한다.

어떤 OS 기반인지, 어떤 패키지를 설치할지, 어떤 앱을 실행할지를 정의한 “설계도” 작성.

2. 이미지를 빌드한다.

Dockerfile을 바탕으로 실제 Docker 이미지 생성.

`docker build` 명령어 사용.

1. 이미지를 실행한다 → 컨테이너가 생성된다.

`docker run` 명령어로 이미지를 실행하면, 컨테이너(실행 인스턴스)가 만들어진다.

III. Docker 이미지 실습

Dockerfile 작성하기

vi 에디터 사용법을 익혀두면 터미널 환경에서 파일 편집 작업이 수월해지기 때문에 연습해봅시다.

폴더를 만들고 경로로 이동

```
mkdir -p img-sample
```

```
cd img-sample
```

vi 에디터로 Dockerfile을 만들기

```
vi Dockerfile
```

* 꼭 파일 이름이 Dockerfile 이어야하나요 ?

도커는 기본적으로 현재 디렉토리의 **Dockerfile**을 자동으로 인식해서 빌드합니다.

만약 도커파일의 이름을 다른 것으로 만들었을 경우엔,

docker build 명령어에서 **-f** 옵션으로 파일명을 명시해줘야 합니다.

III. Docker 이미지 실습

vi 에디터의 3가지 주요 모드

| 모드 | 전환 방법 | 설명 |
|----------------------|-----------|--|
| 명령 모드 (Command Mode) | Esc | 기본 모드. 복사, 붙여넣기, 삭제, 저장, 종료 등 명령 입력 |
| 입력 모드 (Insert Mode) | i, a, o 등 | 텍스트 입력 가능. |
| 라인 모드 (Ex Mode) | : | 저장, 종료 같은 명령 입력 (:w, :q, :wq, :x, :q! 등) |

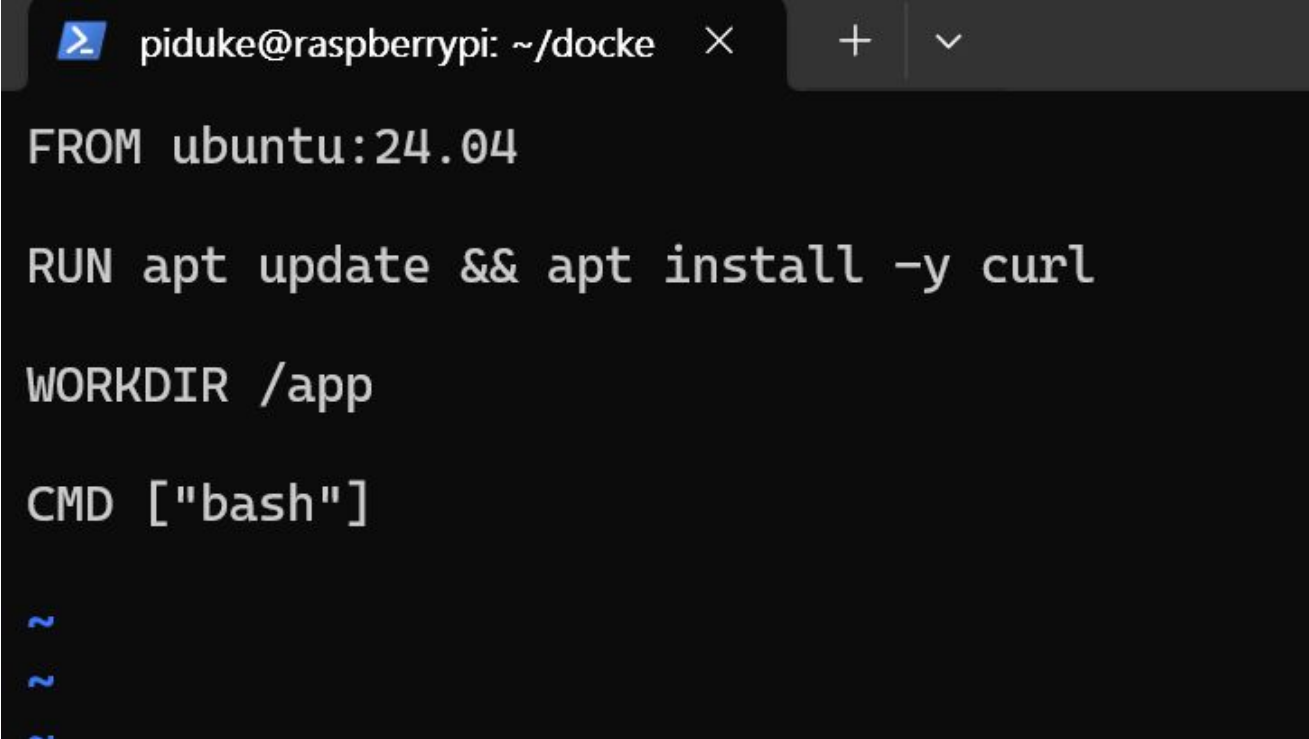
III. Docker 이미지 실습

도커 파일 예시

====

```
FROM ubuntu:24.04
RUN apt update && apt install -y curl
WORKDIR /app
CMD ["bash"]
```

===

A screenshot of a terminal window with a dark background. The window title bar shows 'piduke@raspberrypi: ~/docke' and standard window controls. The terminal content displays a Dockerfile with the following instructions: 'FROM ubuntu:24.04', 'RUN apt update && apt install -y curl', 'WORKDIR /app', and 'CMD ["bash"]'. There are some blue squiggly lines at the bottom of the terminal, likely indicating syntax highlighting or autocomplete suggestions.

```
> piduke@raspberrypi: ~/docke × + v
FROM ubuntu:24.04
RUN apt update && apt install -y curl
WORKDIR /app
CMD ["bash"]
~
~
~
```

입력모드로 내용을 작성한 이후,
ESC키 -> :w -> :q 으로 문서 작성 종료

III. Docker 이미지 실습

작성한 Dockerfile을 이용해서 Docker 이미지를 빌드 해봅시다.

#빌드 명령어 (Dockerfile이 있는 곳에서 실행)

docker build -t ubuntu:24.04 .

- **-t ubuntu:24.04** 생성될 이미지 이름 (**ubuntu**) : Tag 정보 (**24.04**)
- **.** 현재 디렉토리의 Dockerfile을 사용한다는 의미

```
piduke@raspberrypi:~/docker-tutorial/img-sample $ docker build -t ubuntu:24.04 .
[+] Building 45.0s (7/7) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.1s
=> => transferring dockerfile: 123B                             0.0s
=> [internal] load metadata for docker.io/library/ubuntu:24.04 2.6s
=> [internal] load .dockerignore                                0.1s
=> => transferring context: 2B                                    0.0s
=> [1/3] FROM docker.io/library/ubuntu:24.04@sha256:1e622c5f073b4f6bfad6632f2616c7f59ef256e96fe78bf6a595d1dc4376ac02 6.9s
=> => resolve docker.io/library/ubuntu:24.04@sha256:1e622c5f073b4f6bfad6632f2616c7f59ef256e96fe78bf6a595d1dc4376ac02 0.1s
=> => sha256:1e622c5f073b4f6bfad6632f2616c7f59ef256e96fe78bf6a595d1dc4376ac02 6.69kB / 6.69kB 0.0s
=> => sha256:97b5e4984a1c353da6a9406c84554aab0422735a1335427a9e883ac477bd71df 424B / 424B 0.0s
=> => sha256:7fc8925289a890695754108847a52df143c50fb950d185b28ec19be502d09071 2.31kB / 2.31kB 0.0s
=> => sha256:49b96e96358d7aed127d4f4cd2294d77d497c683123bbad89fa80a83d8ef64aa 28.85MB / 28.85MB 3.1s
=> => extracting sha256:49b96e96358d7aed127d4f4cd2294d77d497c683123bbad89fa80a83d8ef64aa 3.3s
=> [2/3] RUN apt update && apt install -y curl                 33.2s
=> [3/3] WORKDIR /app                                           0.2s
=> exporting to image                                           1.6s
=> => exporting layers                                          1.6s
=> => writing image sha256:ccc23bdd3bb29e3ae384fba9a98057bd78a8a95e7d6fcdbf5341d90de590faf8 0.0s
=> => naming to docker.io/library/ubuntu:24.04                 0.0s
piduke@raspberrypi:~/docker-tutorial/img-sample $ |
```


III. Docker 이미지 실습

#빌드된 이미지를 확인하는 방법

docker images

```
piduke@raspberrypi:~/docker-tutorial/img-sample $ docker images
REPOSITORY      TAG              IMAGE ID         CREATED          SIZE
ubuntu          24.04           ccc23bdd3bb2    36 minutes ago  159MB
hello-world     latest          f1f77a0f96b7    3 months ago   5.2kB
ros             humble          f3589c9910d6    2 years ago     716MB
piduke@raspberrypi:~/docker-tutorial/img-sample $
```

| 컬럼 | 의미 |
|------------|-----------------------------------|
| REPOSITORY | 이미지 이름 (docker build -t 시 설정한 이름) |
| TAG | 버전 태그 (latest, v1.0 등) |
| IMAGE ID | 이미지 고유 ID |
| CREATED | 생성 시점 |
| SIZE | 이미지 용량 |

III. Docker 이미지 실습

컨테이너의 실행

문법 : `docker run [OPTIONS] <이미지이름>:<태그>`

`docker run -itd <이미지이름>:<태그>`

run 명령어를 통해서 실행된 컨테이너를 확인하는 방법

`docker ps -a`

```
piduke@raspberrypi:~/docker-tutorial/img-sample $ docker run -itd ubuntu:24.04
a34a147698b8bbe065823b2cfb68e3a6543799983bf513700294dc491031a983
```

```
piduke@raspberrypi:~/docker-tutorial/img-sample $ docker ps -a
```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|--------------|---------|---------------|--------------|-------|--------------------|
| a34a147698b8 | ubuntu:24.04 | "bash" | 5 seconds ago | Up 4 seconds | | unruffled_meninsky |

```
piduke@raspberrypi:~/docker-tutorial/img-sample $
```

III. Docker 이미지 실습

컨테이너의 실행 옵션 설명

`docker run -itd <이미지이름>:<태그>` 에서 각각의 옵션은 다음을 의미합니다:

- **-i, --interactive**

컨테이너의 표준 입력(STDIN)을 열어 둡니다.
터미널에서 입력을 받을 수 있도록 유지해 줍니다.

- **-t, --tty**

가상 터미널(tty)을 할당합니다.
셸을 사용할 때 컬러 출력, 이모지, 커서 이동 등 터미널 제어 문자를 제대로 처리할 수 있게 해 줍니다.

- **-d, --detach**

컨테이너를 백그라운드에서 실행합니다.
명령을 실행한 터미널에서는 바로 프롬프트가 돌아오고, 컨테이너는 뒤에서 계속 실행됩니다.

III. Docker 이미지 실습

컨테이너의 삭제

`docker rm {컨테이너의 ID}`

```
piduke@raspberrypi:~ $ docker rm 2d4b9a67a5fe
```

```
2d4b9a67a5fe
```

```
piduke@raspberrypi:~ $
```

```
piduke@raspberrypi:~ $
```

```
piduke@raspberrypi:~ $ docker ps -a
```

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

```
piduke@raspberrypi:~ $
```

III. Docker 이미지 실습

다시 컨테이너를 생성해보고, 터미널 접속해보자.

`docker run -itd <이미지이름>:<태그>`

`docker exec -it <컨테이너_ID_or_NAME> /bin/bash`

```
piduke@raspberrypi:~/docker-tutorial/img-sample $  
piduke@raspberrypi:~/docker-tutorial/img-sample $ docker exec -it a34a147698b8 /bin/bash  
root@a34a147698b8:/app#  
root@a34a147698b8:/app#  
root@a34a147698b8:/app#
```

root 사용자로, bash 쉘 터미널로 접속이 된 것을 확인 할 수 있다.

터미널 종료는 `exit` 를 입력하거나, `Ctrl + D` 키를 누르면 종료 할 수 있다.

III. Docker 이미지 실습

exit 입력 후 컨테이너 상태 확인

터미널은 종료되지만, 컨테이너가 동작 중 인것을 확인할 수 있다.

```
root@a34a147698b8:/app# exit
exit
piduke@raspberrypi:~/docker-tutorial/img-sample $ docker ps -a
```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|--------------|---------|--------------|-------------|-------|--------------------|
| a34a147698b8 | ubuntu:24.04 | "bash" | 21 hours ago | Up 21 hours | | unruffled_meninsky |

```
piduke@raspberrypi:~/docker-tutorial/img-sample $
```

IV. Docker 의 버전관리

Docker commit 을 하는 방법

`docker commit [OPTIONS] <컨테이너_ID_or_NAME> <저장할_이미지명>:<버전태그>`

다시 컨테이너에 쉘로 붙어서 테스트를 해보자.

현재 실행중인 컨테이너 확인

`docker ps -a`

`docker exec -it <컨테이너 ID> /bin/bash`

현재 경로에서 `touch` 명령어로 비어있는 `test.txt` 파일을 생성해주고 확인한다.

`touch test.txt`

`ls -l`

```
piduke@raspberrypi:~/docker-tutorial/img-sample $ docker exec -it a34a147698b8 /bin/bash
root@a34a147698b8:/app# touch test.txt
root@a34a147698b8:/app# ls -l
total 0
-rw-r--r-- 1 root root 0 May  6 04:24 test.txt
root@a34a147698b8:/app#
```


IV. Docker 의 버전관리

도커 컨테이너에서 나간 후, commit 생성하기

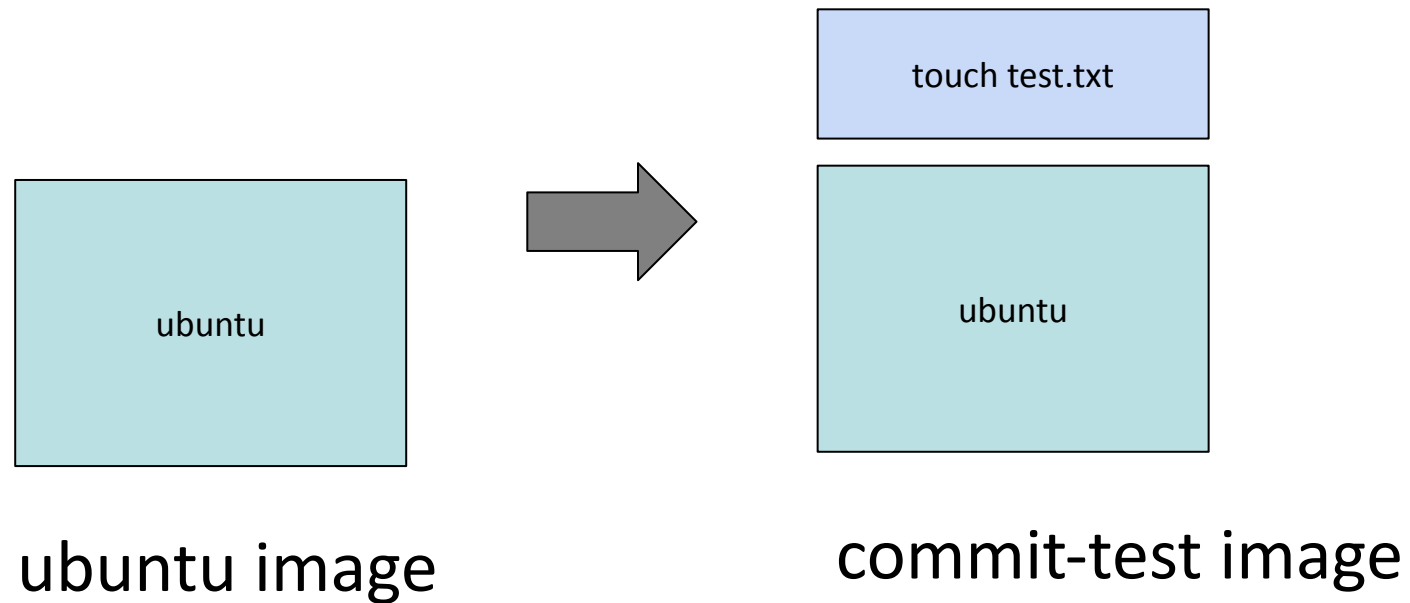
exit

docker commit [OPTIONS] <컨테이너_ID_or_NAME> <저장할_이미지명>:<버전태그>

현재 컨테이너의 상태와 동일한 Docker Image가 생성 된 것을 확인 할 수 있다.

```
root@a34a147698b8:/app# exit
exit
piduke@raspberrypi:~/docker-tutorial/img-sample $ docker commit a34a147698b8 commit-test/test:0.0.1
sha256:bd5c7318f2598d5edf8b0dcdca28153f5afa83aca07cbbbae4346cfd3b809ee1
piduke@raspberrypi:~/docker-tutorial/img-sample $ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
commit-test/test    0.0.1              bd5c7318f259       43 seconds ago     159MB
ubuntu              24.04              ccc23bdd3bb2       2 days ago         159MB
hello-world         latest             f1f77a0f96b7       3 months ago       5.2kB
ros                 humble             f3589c9910d6       2 years ago        716MB
piduke@raspberrypi:~/docker-tutorial/img-sample $ |
```


IV. Docker 의 버전관리



IV. Docker 의 버전관리

Docker Commit 장점

- 즉시 스냅샷
 - 컨테이너의 현재 상태를 즉시 이미지로 저장 -> 중간 결과 보존
- 빠른 디버깅 & 테스트
 - 변경 전후 이미지를 비교하며 문제 원인 탐색
- 버전 태깅 & 공유
 - 태그로 상태 식별, 팀원과 공유
- Dockerfile 학습 보조
 - 레이어 생성 과정을 실험하며 Docker 구조 이해
- 프로토타이핑 최적화
 - PoC 단계에서 수동 설정 후 커밋 -> 빠른 반복 작업

주의: 최종 배포·협업 시에는 Dockerfile을 작성하여 빌드를 하는 것을 권장.

QnA



THANK YOU