

# Docker Tutorial





# INDEX

1 Docker 란?

---

2 Docker 설치법

---

3 Docker 간단한 사용법

---

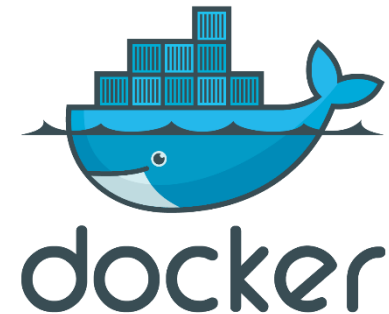
4 Docker 예제 – AWS

---

# 1.1 Docker 란?

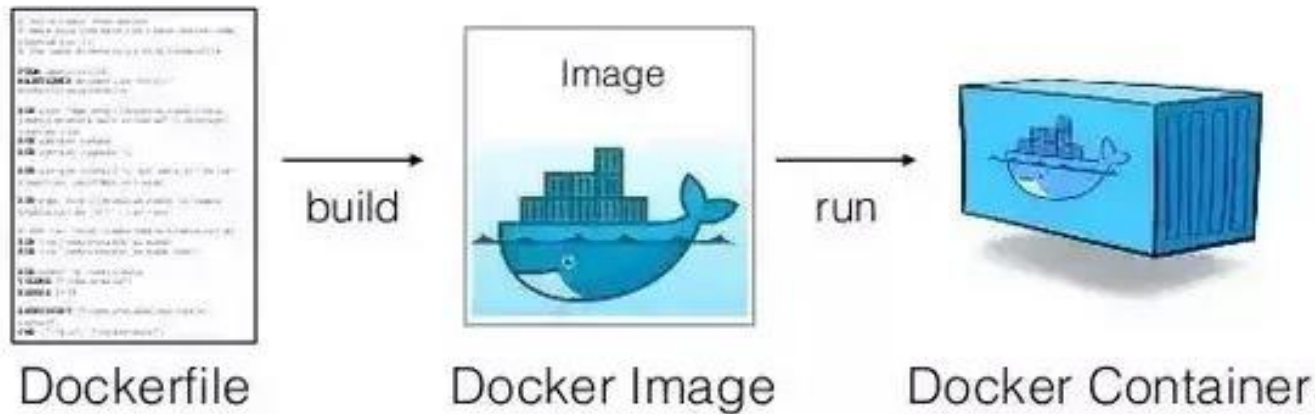
▶ 리눅스 컨테이너 기반으로 하여 특정한 서비스를 패키징하고 배포하는데 유용한 오픈소스 가상화 플랫폼

- 컨테이너(Container): 서버에서의 컨테이너는 다양한 프로그램, 실행환경을 컨테이너로 추상화 하고 동일한 인터페이스를 제공하여 프로그램의 배포 및 관리를 단순하게 해준다.
- 리눅스 커널의 여러 기술을 활용한다.
- 하드웨어 가상화 기술보다 가볍다.
- 이미지 단위로 프로세스 실행 환경을 구성한다.



# 1.1 Docker 란?

## Docker의 구동방식 Overview



**Docker File** 을 만들어서 "나는 어떠한 소프트웨어를 컨테이너에 담아서 구동시킬 것이다."라는 것만 정확히 명시하고 빌드(Build)를 한다.

**Docker Image**가 그에 맞게 생성이 된다.

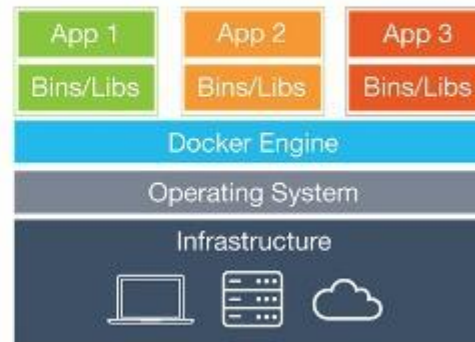
Docker Image를 구동 시키면 순식간에 **Docker Container** 위에서 실행된다.  
(웹 서버도 간단히 실행된다.)

# 1.2 Docker 사용하는 이유

## VM vs Docker 비교하였을 때



Virtual Machines



Containers

Vmware나 VirtualBox 같은 Virtual Machines는 호스트OS 위에 게스트 OS 전체를 가상화 하여 사용하는 방법과 달리,  
Docker는 별도의 Guest OS가 사용되지 않는다.

# 1.3 Docker의 장점

---

- PaaS와 같은 제한이 없다.
- 클라우드 이미지보다 관리하기 쉽다.
- 다른 프로세스와 격리되어 가상머신처럼 사용하지만 성능저하가 (거의) 없다.
- 복잡한 기술(namespace, cgroups, network....)을 몰라도 사용할 수 있다.
- 이미지 빌드 기록이 남는다.
- 코드와 설정으로 관리한다. => 재현 및 수정이 가능하다.
- 오픈소스가 특정 회사 기술에 종속적이지 않는다.

# 1.3 Docker의 장점

---

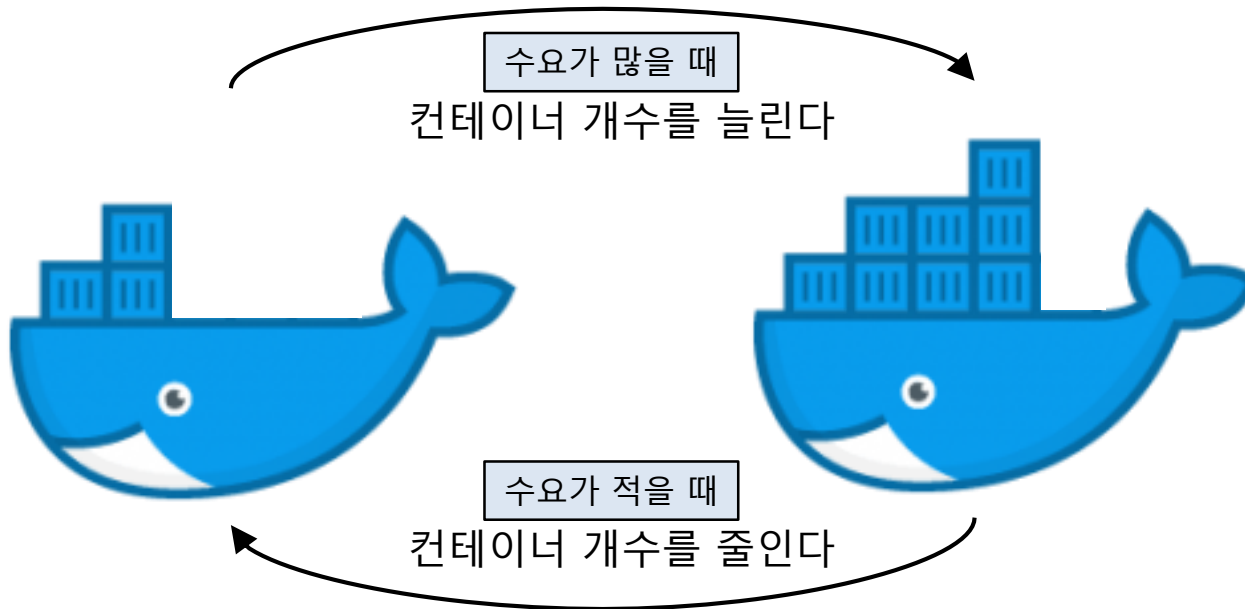
## ▶ 쉬운 환경 구축

- 도커 파일을 만들어 쉽게 배포하고 다운받아 이미지를 실행시키기만 하면 어디서든 같은 환경을 쉽고 빠르게 구축할 수 있다.
- 개발자 데스크톱부터 테스트 시스템, 로컬 장비 뿐만 아니라 AWS, azure 등의 많은 클라우드 서비스에서 docker를 지원하므로 원격 클라우드에도 쉽게 환경을 구축할 수 있다.

# 1.3 Docker의 장점

## ▶ 탄력적인 운용

- 앱의 수요에 따라 컨테이너의 개수를 조정해 탄력으로 운용할 수 있다.





# 1.3 Docker의 장점

---

## ▶ 안전성

- 컨테이너는 완전히 독립된 상태로, 특정 컨테이너를 수정/삭제해도 다른 컨테이너에는 아무 영향을 끼치지 않아 다른 프로세스는 안정적 상태를 유지한다.

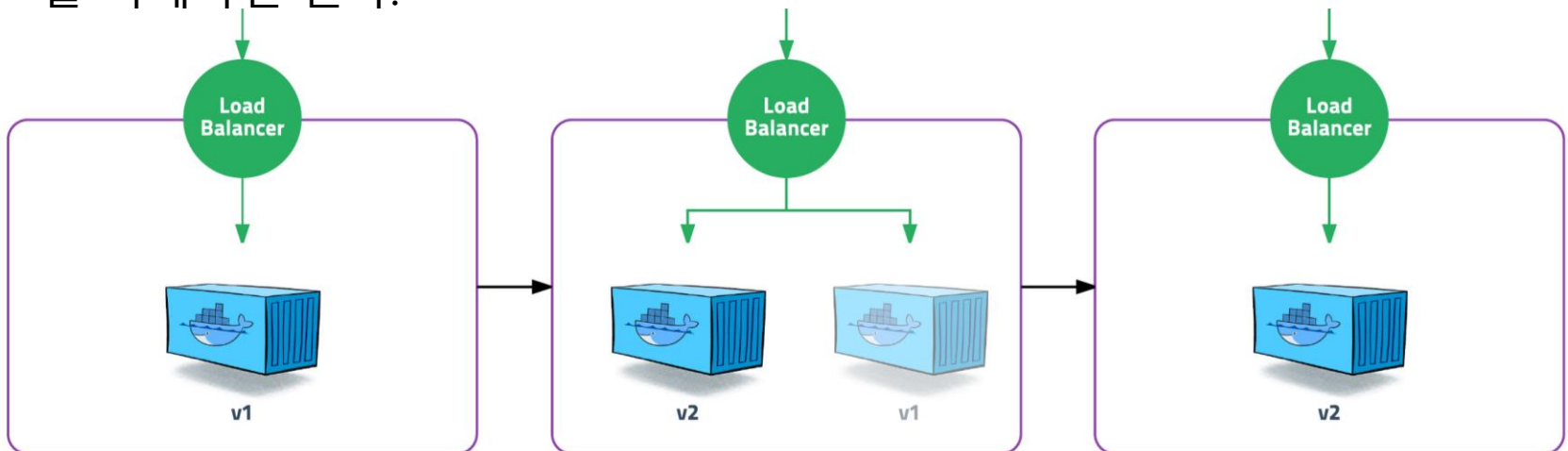
## ▶ 가벼움

- 게스트 OS를 사용하는 가상 머신과 달리 도커 컨테이너는 호스트 OS의 자원을 사용하므로 게스트OS와 호스트OS간의 기능 중복이 없다.
- 도커 컨테이너는 호스트 OS를 활용하고 애플리케이션이 수행할 때 필요로 하는 것들만 실행시키므로 메모리 소모가 크지 않다.

# 1.3 Docker의 장점

## ▶ 업그레이드/다운그레이드가 쉽다

- 아예 새로운 컨테이너를 실행
- 환경에 대한 변화가 생기면 이미지 파일에서 해당 부분만 수정해서 새로운 컨테이너를 실행시키고 기존의 컨테이너는 중지시키면 된다.
- 다운그레이드의 경우 원래 이미지를 실행 시킨 후 업그레이드 된 컨테이너를 삭제하면 된다.

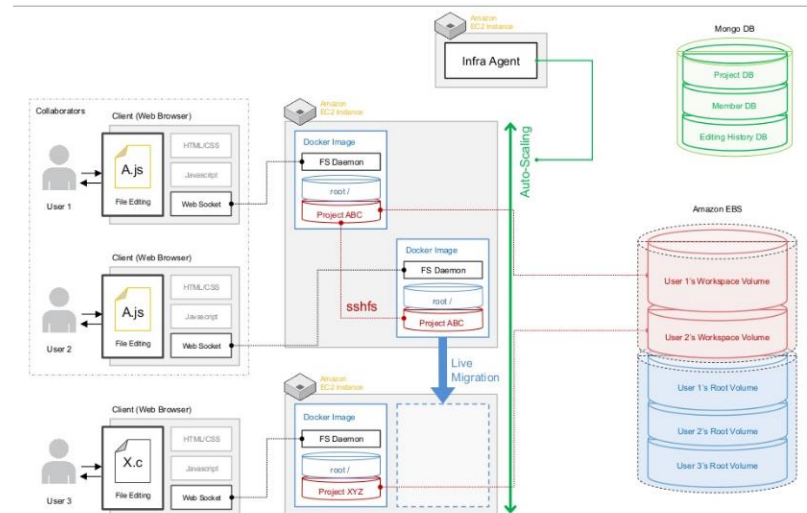
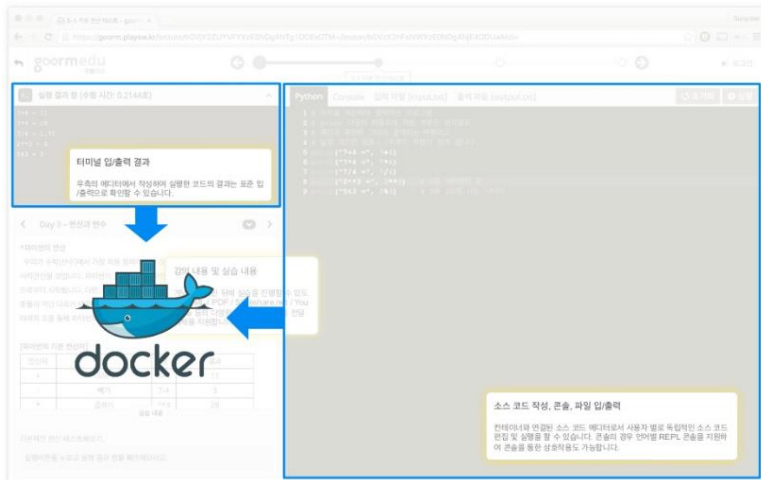


# 1.4 이용사례

AWS Container Day 2015.12.14 codigm 발표 내용

## ▶ 클라우드 코딩 서비스 구름

- 환경의 제약을 없애고 Docker container안에서 코딩을 할 수 있도록 한 교육용 코딩 서비스
- Docker와 AWS를 사용해 하나의 서버만으로 여러 사용자에게 개인별 가상 머신을 제공, 다양한 프로그래밍 언어로 코딩을 해 온라인으로 실행할 수 있게 했다.



## 1.4 이용사례

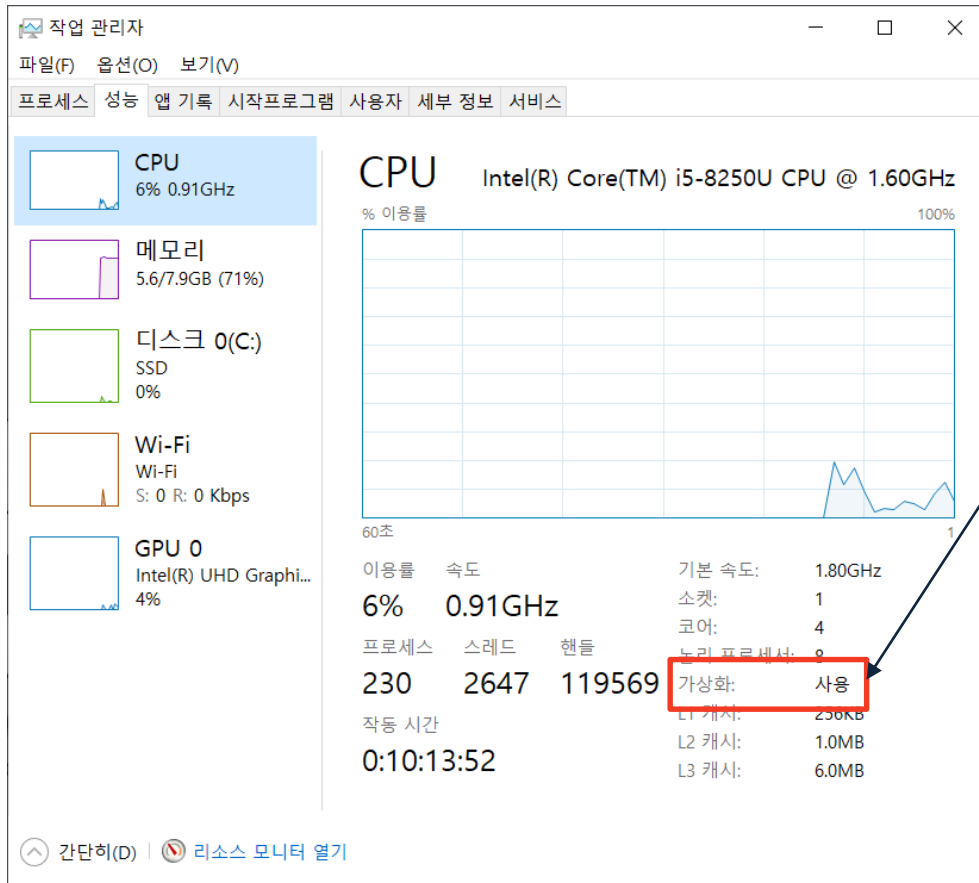
▶ 수많은 기업에서 서비스 배포에 도커 사용



# 2.1 Windows 10 Pro에서 docker설치

## 1. 작업관리자에서 가상화 사용 중인지 확인합니다.

※window 버전 확인 : 설정 – 시스템 – 정보 에 있는 Windows 사양-에디션 부분에서 확인

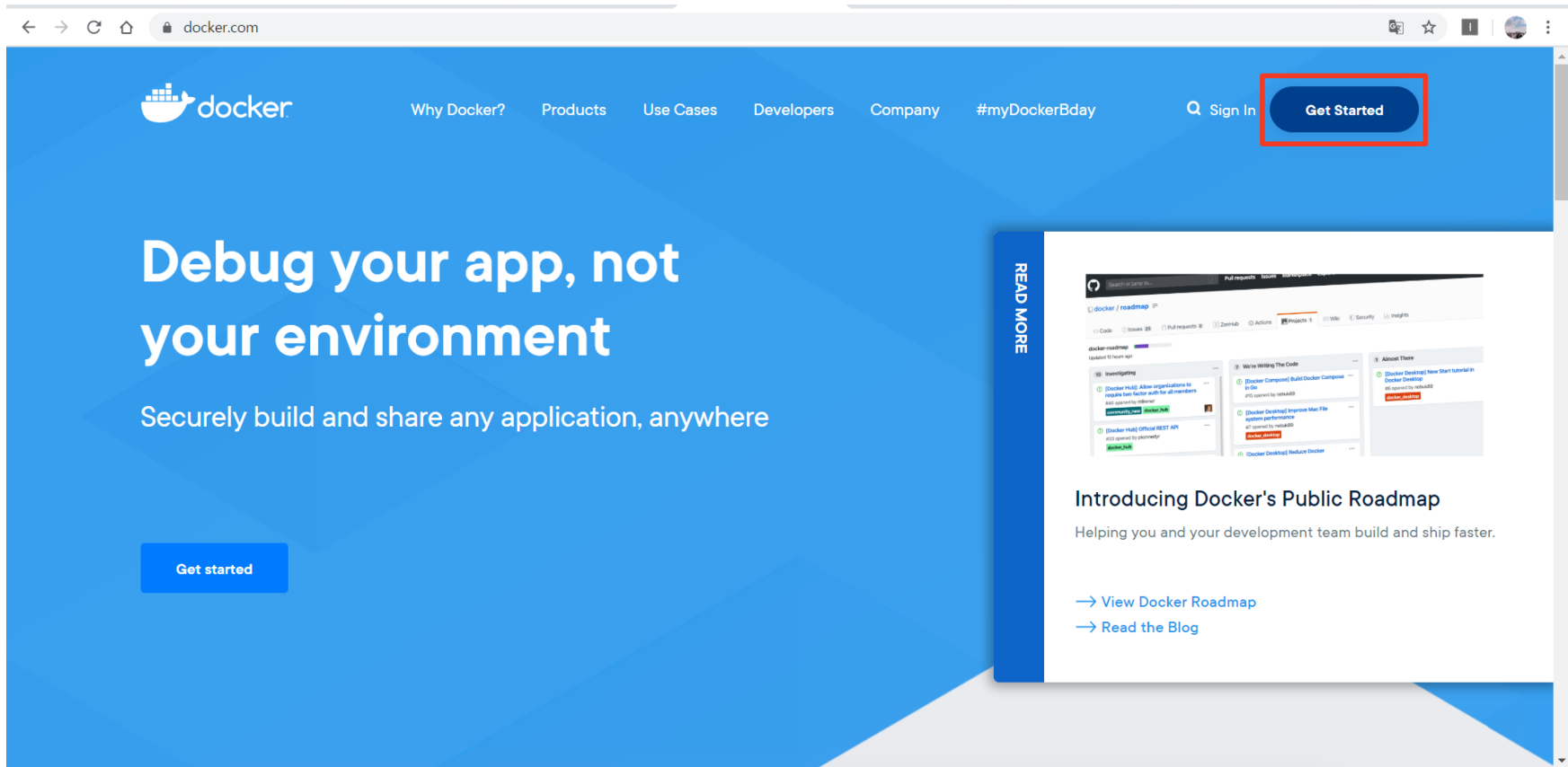


가상화가 사용 안함으로 되어 있을 경우  
가상화기능을 활성화 시켜야 함

※BIOS 세팅 모드- 시큐리티/부팅(어드밴스) –  
Virtualization Technology를 활성화(Enable) - 재부  
팅

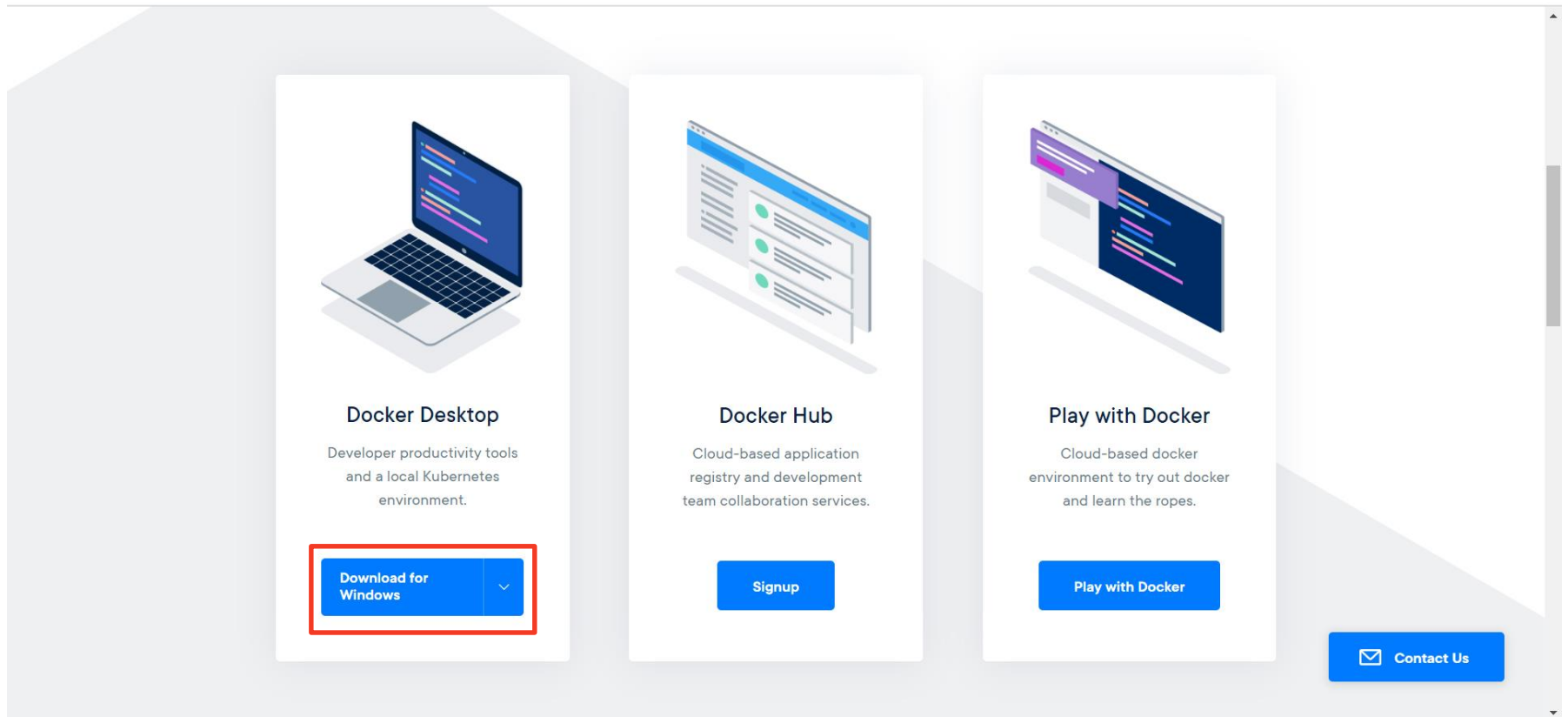
# 2.1 Windows 10 Pro에서 docker설치

2. docker.com에 접속해 Get started 버튼을 누릅니다



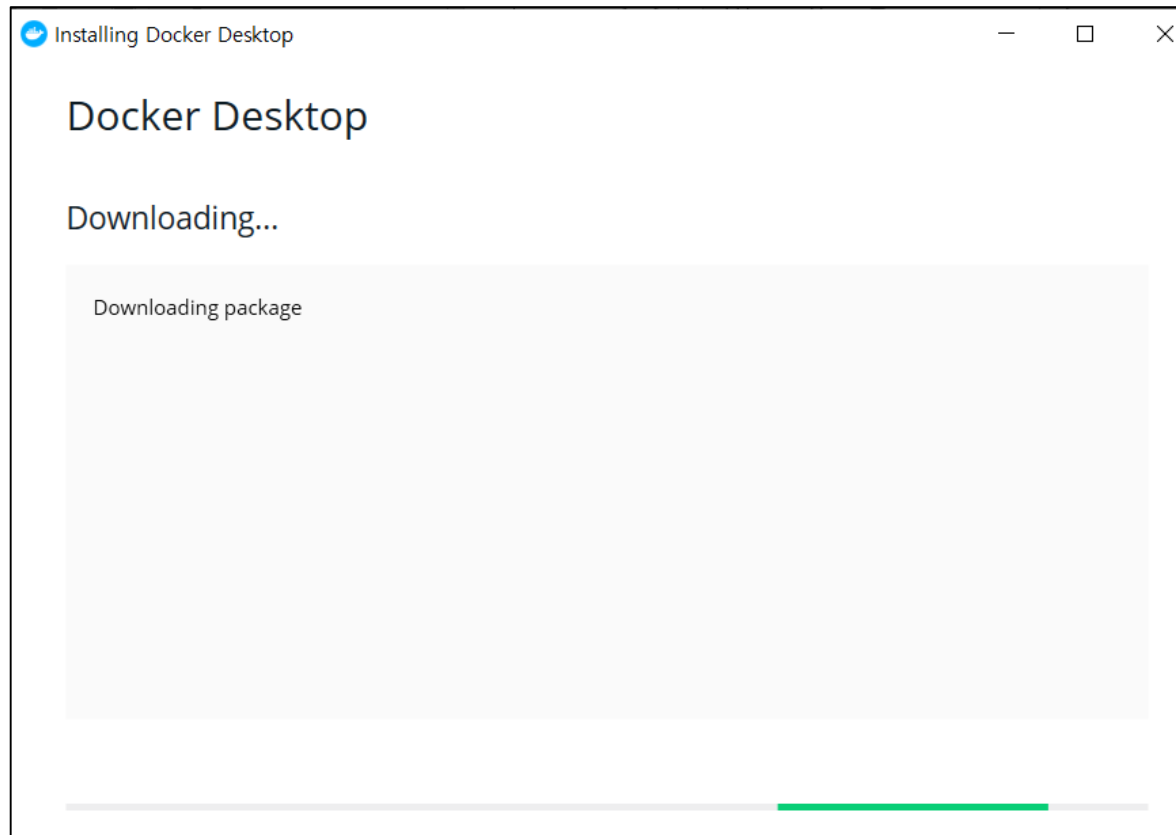
## 2.1 Windows 10 Pro에서 docker설치

### 3. Docker Desktop 설치파일을 다운받습니다.



## 2.1 Windows 10 Pro에서 docker설치

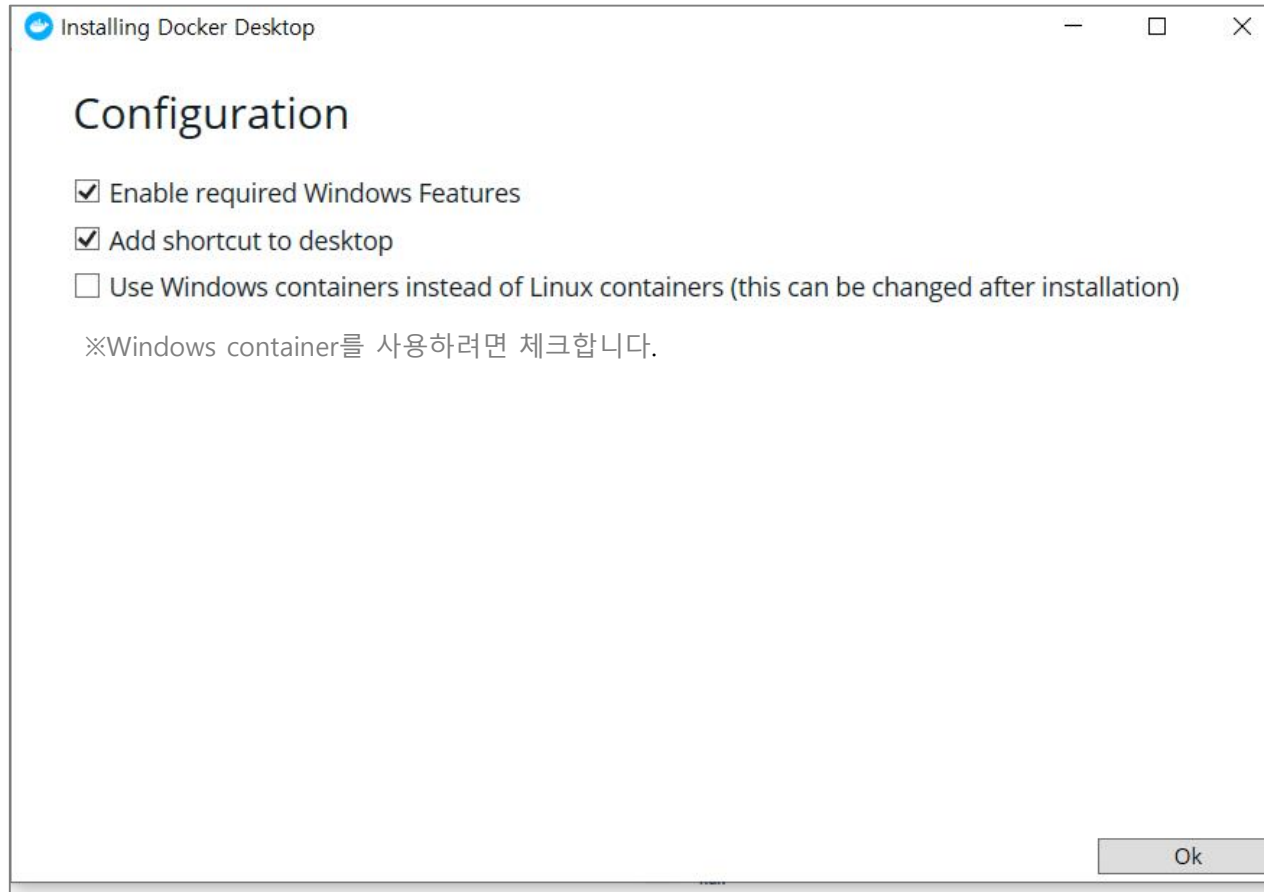
4. 다운받은 Docker Desktop Installer를 실행합니다.





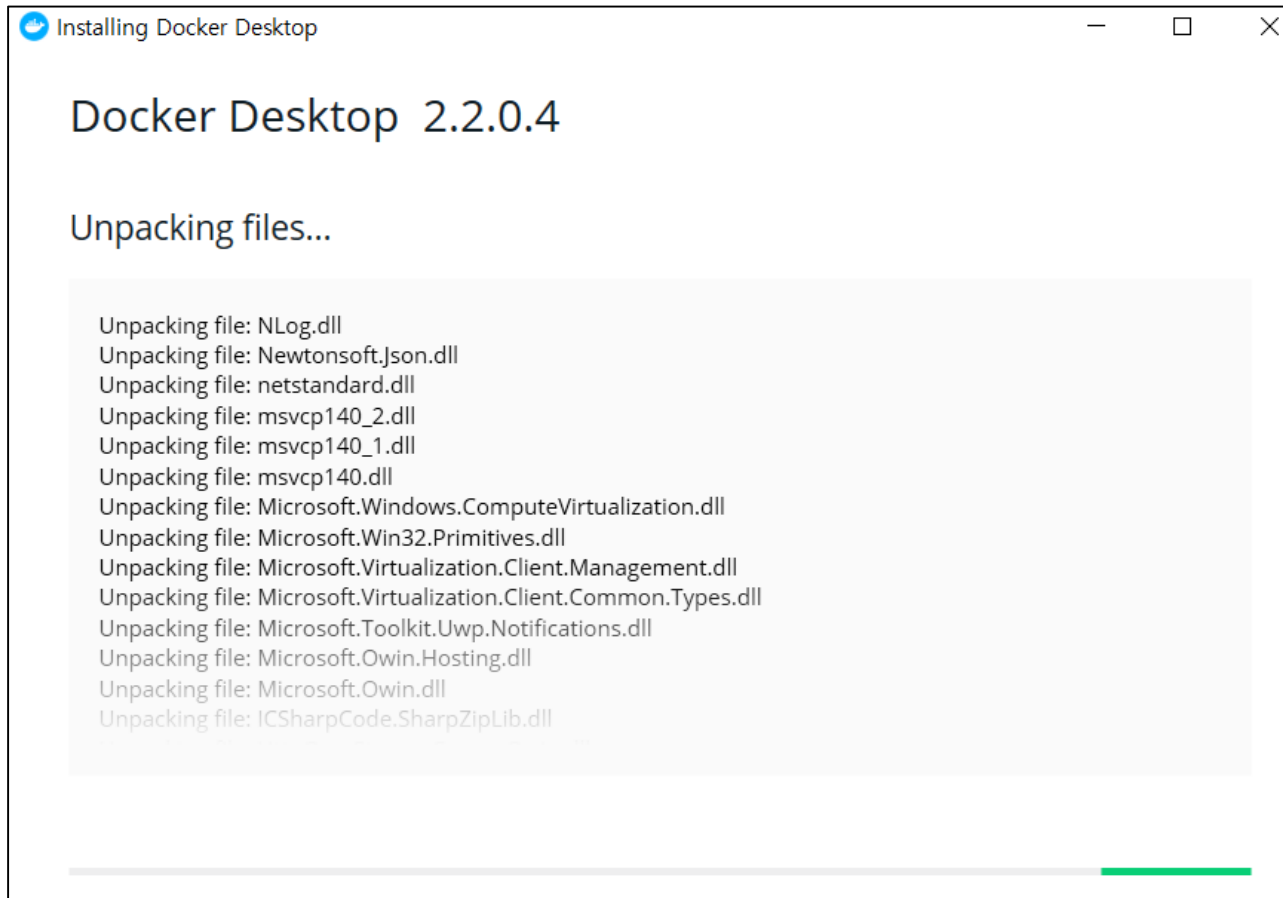
## 2.1 Windows 10 Pro에서 docker설치

5. 아래와 같이 설정한 후 Ok를 누릅니다.



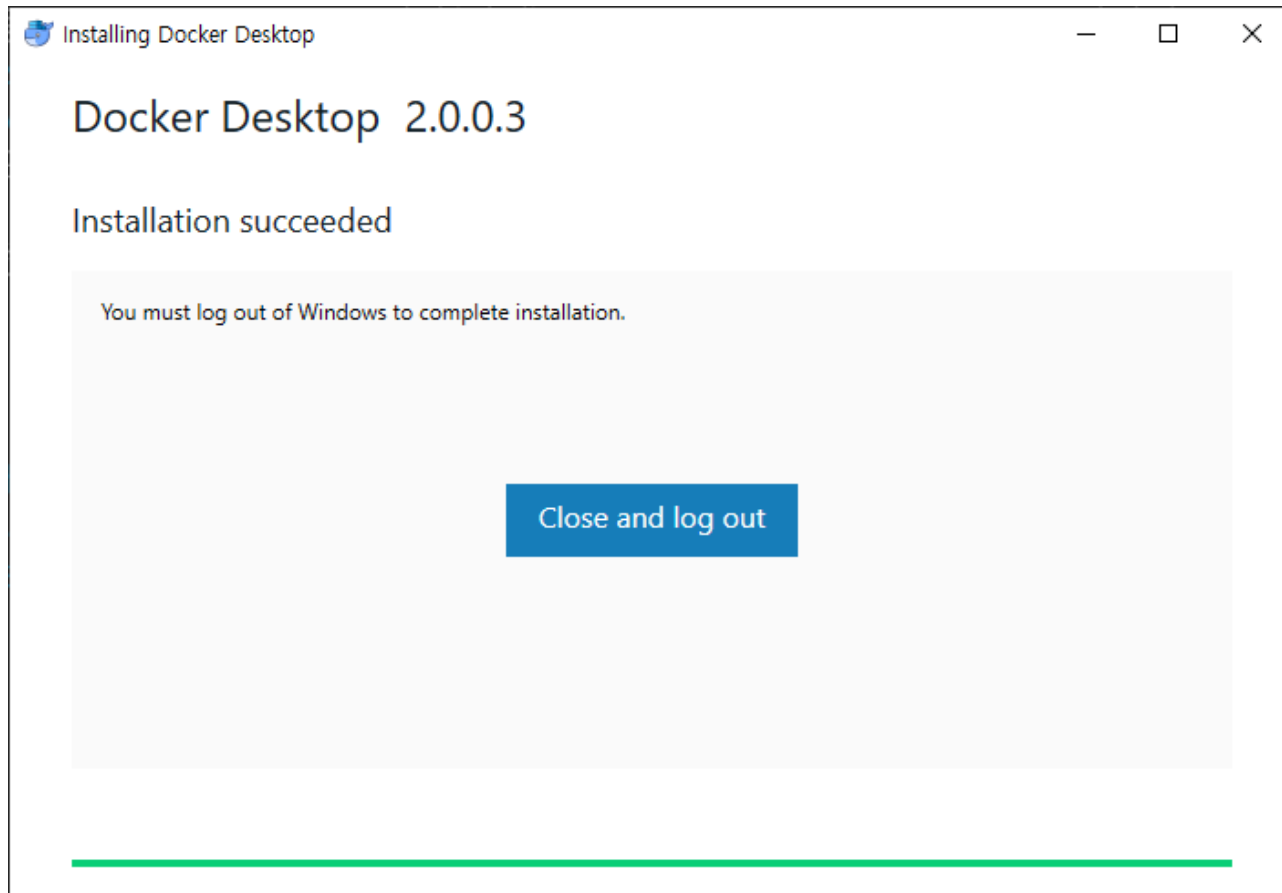
# 2.1 Windows 10 Pro에서 docker설치

## 6. 기다립니다



## 2.1 Windows 10 Pro에서 docker설치

### 7. Docker Desktop 설치완료



## 2.1 Windows 10 Pro에서 docker설치

8. Docker 홈페이지의 계정으로 로그인합니다.



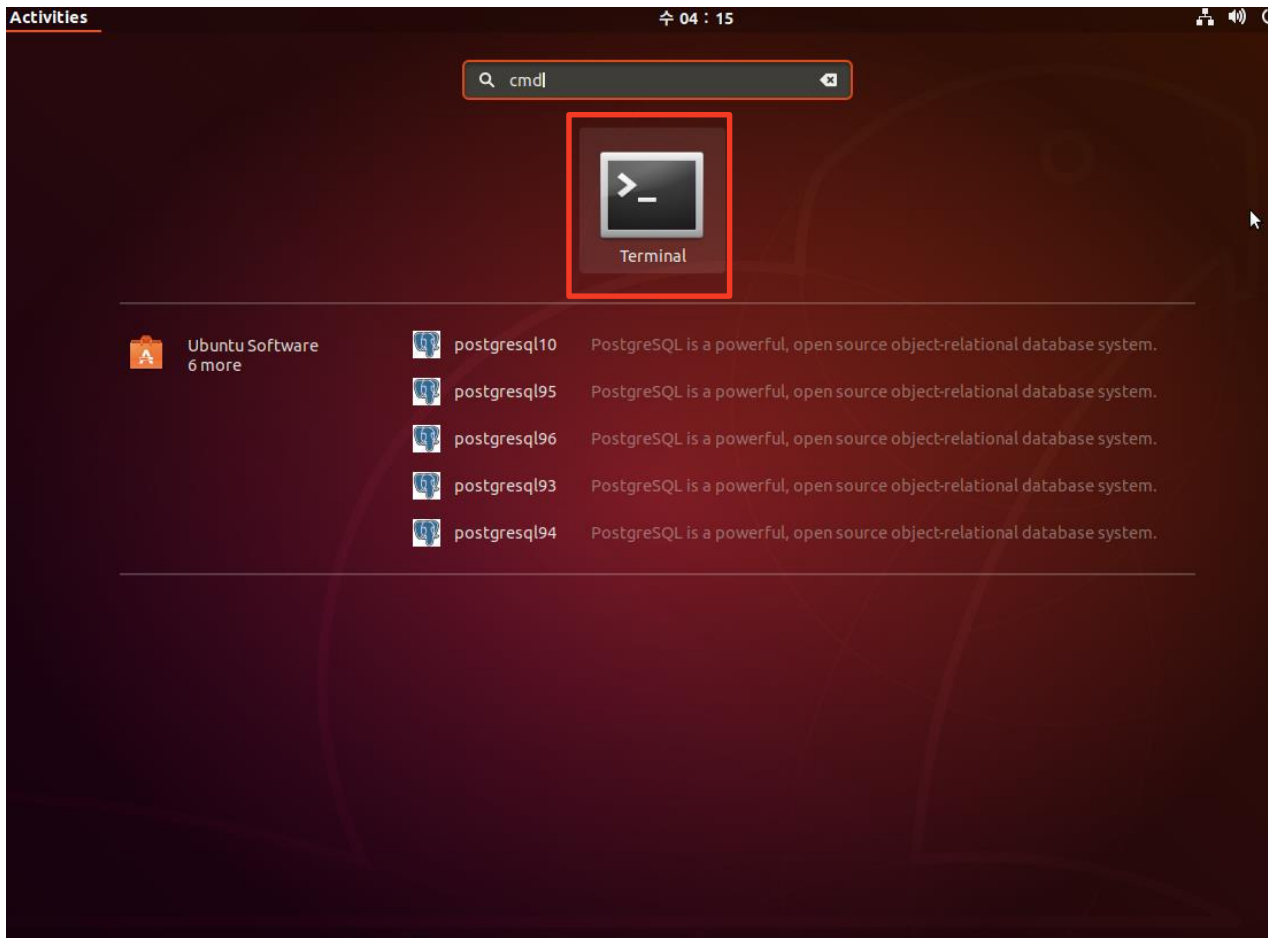
※계정이 없으면 클릭하여 만들어주세요



작업표시줄에 아이콘 생김

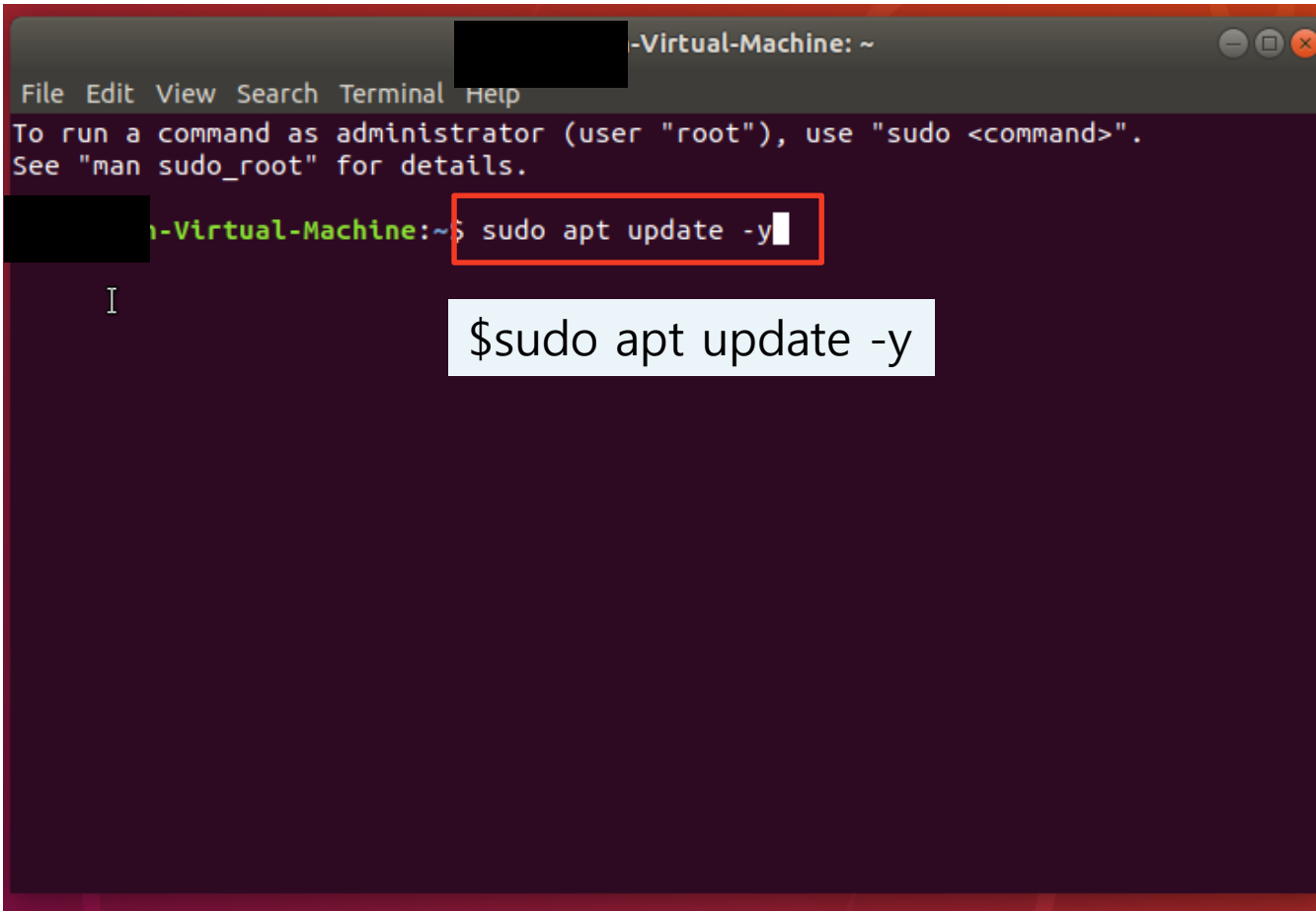
## 2.2 Ubuntu 18.04에서 docker설치

### 1. Ubuntu의 Terminal에 접속한다.



## 2.2 Ubuntu 18.04에서 docker설치

### 2. 패키지목록을 업데이트 한다.



A terminal window titled "-Virtual-Machine: ~" with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal displays the message: "To run a command as administrator (user "root"), use "sudo <command>". See "man sudo\_root" for details." Below this, the command `sudo apt update -y` is being entered at the prompt `-Virtual-Machine:~$`. The command is highlighted with a red box. A separate white box below the terminal shows the command `$sudo apt update -y`.

## 2.2 Ubuntu 18.04에서 docker설치

### 3. Docker CE에 필요한 4가지 패키지를 설치한다.

```
kwon@kwon-Virtual-Machine: ~  
File Edit View Search Terminal Help  
curl software-properties-common  
패키지 목록을 읽는 중입니다... 완료  
의존성 트리를 만드는 중입니다  
상태 정보를 읽는 중입니다... 완료  
apt-get https 패키지를 찾을 수 없습니다  
kwon@kwon-Virtual-Machine:~$ sudo apt install -y apt-transport-https ca-certificates  
curl software-properties-common  
패키지 목록을 읽는 중입니다... 완료  
의존성 트리를 만드는 중입니다
```

\$sudo apt install -y apt-transport-https ca-certificates curl software-properties-common

```
curl is already the newest version (7.58.0-2ubuntu3.8).  
software-properties-common is already the newest version (0.96.24.32.12).  
apt-transport-https is already the newest version (1.6.12).  
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:  
app-install-data apt-clone archdetect-deb btrfs-tools cryptsetup-bin  
device-tree-compiler dmeventd dmraid dpkg-repack gir1.2-timezonemap-1.0  
gir1.2-xkl-1.0 grub-pc-bin kpartx kpartx-boot libdebian-installer4  
libdevmapper-event1.02.1 libdmraid1.0.0.rc16 libido3-0.1-0 liblvm2app2.2  
liblvm2cmd2.02 libreadline5 libtimezonemap-data libtimezonemap1 lvm2 python3-icu  
python3-pam rdate u-boot-tools  
Use 'sudo apt autoremove' to remove them.  
0개 업그레이드, 0개 새로 설치, 0개 제거 및 298개 업그레이드 안 함.  
kwon@kwon-Virtual-Machine:~$
```

## 2.2 Ubuntu 18.04에서 docker설치

### 4. 도커 패키지 저장소를 apt에 등록한다.

```
kwon@kwon-Virtual-Machine: ~  
File Edit View Search Terminal Help  
kwon@kwon-Virtual-Machine:~$ curl -feSL https://download.docker.com/linux/ubuntu/gpg  
| sudo apt-key add -  
% Total % Received % Xferd Average Speed Time Time Time Current  
$curl -feSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
OK  
kwon@kwon-Virtual-Machine:~$ sudo add-apt-repository "deb [arch=amd64] https://downl  
oad.docker.com/linux/ubuntu bionic stable"  
받기:1 https://download.docker.com/linux/ubuntu bionic InRelease [64.4 kB]  
받기:2 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages [11.0 k  
$sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu  
bionic stable"  
기존:5 http://archive.ubuntu.com/ubuntu bionic-updates InRelease  
기존:6 http://archive.ubuntu.com/ubuntu bionic-backports InRelease  
내려받기 75.5 k바이트, 소요시간 2초 (36.7 k바이트/초)  
패키지 목록을 읽는 중입니다... 완료  
kwon@kwon-Virtual-Machine:~$
```



## 2.2 Ubuntu 18.04에서 docker설치

### 5. Apt패키지 목록을 업데이트한다.

```
kwon@kwon-Virtual-Machine: ~  
File Edit View Search Terminal Help  
100 3817 100 3817 0 0 54528 0 --:--:-- --:--:-- --:--:-- 54528  
OK  
kwon@kwon-Virtual-Machine:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"  
받기:1 https://download.docker.com/linux/ubuntu bionic InRelease [64.4 kB]  
받기:2 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages [11.0 kB]  
기존:3 http://security.ubuntu.com/ubuntu bionic-security InRelease  
기존:4 http://archive.ubuntu.com/ubuntu bionic InRelease  
기존:5 http://archive.ubuntu.com/ubuntu bionic-updates InRelease  
기존:6 http://archive.ubuntu.com/ubuntu bionic-backports InRelease  
내려받기 75.5 k바이트, 소요시간 2초 (36.7 k바이트/초)  
패키지 목록을 읽는 중입니다... 완료  
kwon@kwon-Virtual-Machine:~$ sudo apt update -y  
기존:1 http://archive.ubuntu.com/ubuntu bionic InRelease  
기존:2 http://archive.ubuntu.com/ubuntu bionic-updates InRelease  
$sudo apt update -y 받기:3 https://download.docker.com/linux/ubuntu bionic InRelease  
기존:4 http://archive.ubuntu.com/ubuntu bionic-backports InRelease  
기존:5 http://security.ubuntu.com/ubuntu bionic-security InRelease  
패키지 목록을 읽는 중입니다... 완료  
의존성 트리를 만드는 중입니다  
상태 정보를 읽는 중입니다... 완료  
298 packages can be upgraded. Run 'apt list --upgradable' to see them.  
kwon@kwon-Virtual-Machine:~$
```

## 2.2 Ubuntu 18.04에서 docker설치

### 6. Docker CE를 설치한다.

```
-Virtual-Machine: ~
File Edit View Search Terminal Help
-Virtual-Machine:~$ sudo apt install -y docker-ce
패키지 모루을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
$ sudo apt install -y docker-ce
더 이상 필요하지 않습니다:
app-install-data apt-clone archdetect-deb btrfs-tools cryptsetup-bin
device-tree-compiler dmeventd dmraid dpkg-repack gir1.2-timezonemap-1.0
gir1.2-xkl-1.0 grub-pc-bin kpartx kpartx-boot libdebian-installer4
libdevmapper-event1.02.1 libdmraid1.0.0.rc16 libido3-0.1-0 liblvm2app2.2
liblvm2cmd2.02 libreadline5 libtimezonemap-data libtimezonemap1 lvm2 python3-icu
python3-pam rdate u-boot-tools
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
aufs-tools cgroupfs-mount containerd.io docker-ce-cli git git-man liberror-perl
pigz
제안하는 패키지:
git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
gitweb git-cvs git-mediawiki git-svn
다음 새 패키지를 설치할 것입니다:
aufs-tools cgroupfs-mount containerd.io docker-ce docker-ce-cli git git-man
liberror-perl pigz
0개 업그레이드, 9개 새로 설치, 0개 제거 및 298개 업그레이드 안 함.
90.5 M바이트 아카이브를 받아야 합니다.
이 작업 후 419 M바이트의 디스크 공간을 더 사용하게 됩니다.
```

## 2.2 Ubuntu 18.04에서 docker설치

### 7. Docker CE를 시작한다.

```
on-Virtual-Machine: ~
File Edit View Search Terminal Help
Preparing to unpack .../8-git_1%3a2.17.1-1ubuntu0.5_amd64.deb ...
Unpacking git (1:2.17.1-1ubuntu0.5) ...
aufs-tools (1:4.9+20170918-1ubuntu1) 설정하는 중입니다 ...
git-man (1:2.17.1-1ubuntu0.5) 설정하는 중입니다 ...
containerd.io (1.2.13-1) 설정하는 중입니다 ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service →/li
b/systemd/system/containerd.service.
liberror-perl (0.17025-1) 설정하는 중입니다 ...
cgroupfs-mount (1.4) 설정하는 중입니다 ...
docker-ce-cli (5:19.03.8~3-0~ubuntu-bionic) 설정하는 중입니다 ...
pigz (2.4-1) 설정하는 중입니다 ...
git (1:2.17.1-1ubuntu0.5) 설정하는 중입니다 ...
docker-ce (5:19.03.8~3-0~ubuntu-bionic) 설정하는 중입니다 ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service →/lib/sy
stemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket →/lib/system
d/system/docker.socket.
Processing triggers for libc-bin (2.27-3ubuntu1) ...
$sudo systemctl start docker
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
on-Virtual-Machine:~$ sudo systemctl start docker
Virtual Machine: $
```

## 2.2 Ubuntu 18.04에서 docker설치

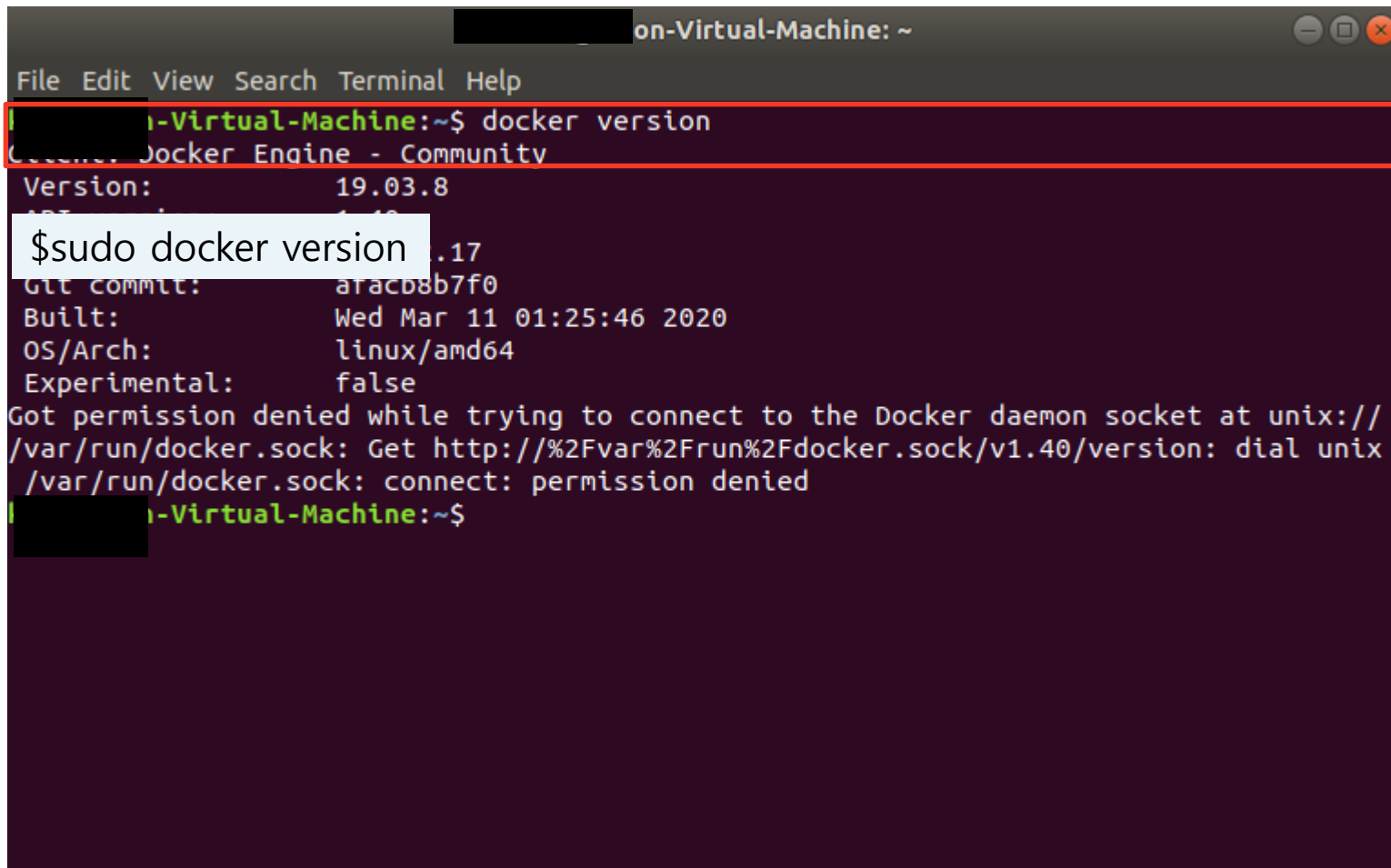
### 8. Docker 상태를 확인한다.

```
File Edit View Search Terminal Help
kwon-Virtual-Machine: ~
kwon-Virtual-Machine:~$ sudo systemctl status docker
systemd.service - Docker Application Container Engine
Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enable)
       Active: active (running) since 2020-04-03 21:58:17 KST; 3min 20s ago
       Main PID: 3568 (dockerd)
          Tasks: 12
         CGroup: /system.slice/docker.service
                 └─3568 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.

4월 03 21:58:16 kwon-Virtual-Machine dockerd[3568]: time="2020-04-03T21:58:16.98287
4월 03 21:58:16 kwon-Virtual-Machine dockerd[3568]: time="2020-04-03T21:58:16.98287
4월 03 21:58:16 kwon-Virtual-Machine dockerd[3568]: time="2020-04-03T21:58:16.98287
4월 03 21:58:16 kwon-Virtual-Machine dockerd[3568]: time="2020-04-03T21:58:16.98301
4월 03 21:58:17 kwon-Virtual-Machine dockerd[3568]: time="2020-04-03T21:58:17.07036
4월 03 21:58:17 kwon-Virtual-Machine dockerd[3568]: time="2020-04-03T21:58:17.13023
4월 03 21:58:17 kwon-Virtual-Machine dockerd[3568]: time="2020-04-03T21:58:17.15335
4월 03 21:58:17 kwon-Virtual-Machine dockerd[3568]: time="2020-04-03T21:58:17.15343
4월 03 21:58:17 kwon-Virtual-Machine dockerd[3568]: time="2020-04-03T21:58:17.18065
4월 03 21:58:17 kwon-Virtual-Machine systemd[1]: Started Docker Application Contain
lines 1-19/19 (END)
```

## 2.2 Ubuntu 18.04에서 docker설치

### 8. Docker 버전을 확인한다.

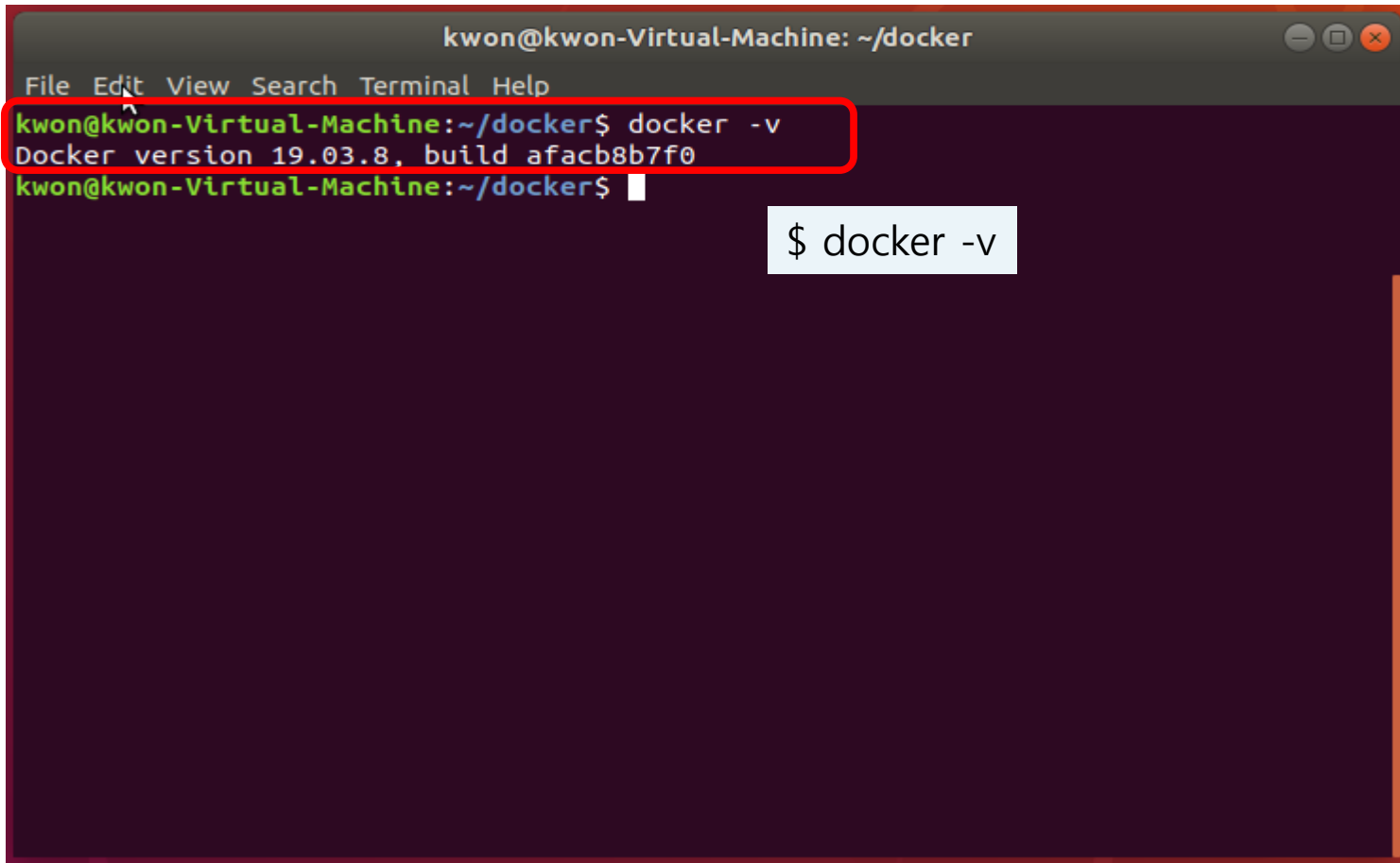


A terminal window titled "on-Virtual-Machine: ~" with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command `docker version` being executed. The output displays Docker Engine - Community version 19.03.8, build afac68b7f0, built on Wed Mar 11 01:25:46 2020, OS/Arch: linux/amd64, and Experimental: false. Below this, an error message is shown: "Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get http://%2Fvar%2Frun%2Fdocker.sock/v1.40/version: dial unix /var/run/docker.sock: connect: permission denied". The prompt returns to `on-Virtual-Machine:~$`. A red rectangle highlights the command and the first part of the output. A white box with the text `$sudo docker version` is overlaid on the terminal.

```
on-Virtual-Machine: ~
File Edit View Search Terminal Help
on-Virtual-Machine:~$ docker version
Docker Engine - Community
Version:      19.03.8
Build:        afac68b7f0
Built:        Wed Mar 11 01:25:46 2020
OS/Arch:      linux/amd64
Experimental: false
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get http://%2Fvar%2Frun%2Fdocker.sock/v1.40/version: dial unix /var/run/docker.sock: connect: permission denied
on-Virtual-Machine:~$
```

# 3.1 자주 사용하는 Docker 명령어

## Docker version 확인



A terminal window titled 'kwon@kwon-Virtual-Machine: ~/docker' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'docker -v' being executed, with the output 'Docker version 19.03.8, build afacb8b7f0'. The command and output are highlighted with a red box. A separate white box with the text '\$ docker -v' is also present.

```
kwon@kwon-Virtual-Machine: ~/docker
File Edit View Search Terminal Help
kwon@kwon-Virtual-Machine:~/docker$ docker -v
Docker version 19.03.8, build afacb8b7f0
kwon@kwon-Virtual-Machine:~/docker$
```

\$ docker -v

# 3.1 자주 사용하는 Docker 명령어

## Docker 이미지 가져오기

```
kwon@kwon-Virtual-Machine: ~/docker
File Edit View Search Terminal Help
kwon@kwon-Virtual-Machine:~/docker$ docker pull centos:7
7: Pulling from library/centos
ab5ef0e58194: Pull complete
Digest: sha256:4a701376d03f6b39b8c2a8f4a8e499441b0d567f9ab9d58e4991de4472fb813c
Status: Downloaded newer image for centos:7
docker.io/library/centos:7

$docker pull [이미지 이름]:[태그]
```

- 위 명령어에서 [태그] 생략 시 latest로 붙는다.

Ex) ubuntu:latest

# 3.1 자주 사용하는 Docker 명령어

## Docker 이미지 확인하기

```
kwon@kwon-Virtual-Machine: ~/docker
File Edit View Search Terminal Help

kwon@kwon-Virtual-Machine:~/docker$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos               7                  5e35e350aded       4 months ago       203MB
kwon@kwon-Virtua
```

\$docker images



# 3.1 자주 사용하는 Docker 명령어

## Docker 컨테이너 생성하기(이미지로 컨테이너 생성하기)

```
kwon@kwon-Virtual-Machine: ~/docker
File Edit View Search Terminal Help
kwon@kwon-Virtual-Machine:~/docker$ docker create -i -t ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
5bed26d33875: Pull complete
f11b29a9c730: Pull complete
930bda195c84: Pull complete
78bf9a5ad49e: Pull complete
Digest: sha256:bec5a2727be7fff3d308193cfde3491f8fba1a2ba392b7546b43a051853a341d
Status: Downloaded newer image for ubuntu:latest
ec53ffe7d0f6d09399d2ea86aaf771cf833f209925d38252001ea85f2e3fea81
kwon@kwon-Virtual-Machine:~/docker$ docker create -i -t centos:7
8b23d5fb6b7bd40e17cfd5ce5f03f5c95cebd5e7607d6e59dc28a606f2fe4ded
kwon@kwon-Virtual-Machine:~/docker$
```

\$docker create [옵션] [이미지이름]:[태그]

- 옵션

옵션	기능
-i	상호 입출력
-t	tty를 활성화 하여 bash셸 사용

- 이미지가 PC에 없다면 자동으로 다운로드 한다.

# 3.1 자주 사용하는 Docker 명령어

## Docker 컨테이너 목록 확인

```
kwon@kwon-Virtual-Machine: ~/docker
File Edit View Search Terminal Help
kwon@kwon-Virtual-Machine:~/docker$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS            PORTS              NAMES
8b23d5fb6b7b       centos:7            "/bin/bash"        14 minutes ago
Created                       epic_cartwright
```

- 옵션

옵션	설명
-a	정지된 컨테이너 까지 보기

## 3.1 자주 사용하는 Docker 명령어

### Docker 컨테이너 목록 확인

- 표시항목

항목	설명
CONTAINER ID	컨테이너 고유 ID
IMAGE	컨테이너에 사용된 이미지 이름
COMMAND	컨테이너가 시작될 때 실행될 명령어 기본은 /bin/bash
CREATED	컨테이너가 생성되고 난 뒤 흐른 시간
STATUS	컨테이너의 상태 ex) Up(실행 중), Exited(종료), Pause(일시중지)
PORTS	컨테이너가 개방한 포트와 호스트에 연결한 포트
NAMES	컨테이너의 고유한 이름

# 3.1 자주 사용하는 Docker 명령어

## Docker 컨테이너 실행하기

```
kwon@kwon-Virtual-Machine: ~/docker
File Edit View Search Terminal Help
94ac41aafd4f centos:7 "/bin/bash"
Created
kwon@kwon-Virtual-Machine:~/docker$ docker start 1778279889a6
1778279889a6
```

- 만들어진 Docker 컨테이너를 실행하는 명령어.

# 3.1 자주 사용하는 Docker 명령어

## Docker 컨테이너 들어가기

```
kwon@kwon-Virtual-Machine: ~/docker
File Edit View Search Terminal Help
kwon@kwon-Virtual-Machine:~/docker$ docker s $docker attach [컨테이너 ID]
1778279889a6
kwon@kwon-Virtual-Machine:~/docker$ docker attach 1778279889a6
[root@1778279889a6 /]#
```

- Docker의 내부셸로 들어가는 명령어.

# 3.1 자주 사용하는 Docker 명령어

## Docker 컨테이너 내부 쉘에서 나가기

```
kwon@kwon-Virtual-Machine: ~/docker
File Edit View Search Terminal Help
1778279889a6      centos:7          "/bin/bash"       14 seconds ago
Created
94ac41aafd4f     centos:7          "/bin/bash"       3 minutes ago
Created
adoring_hermann
kwon@kwon-Virtual-Machine:~/docker$ docker start 1778279889a6
1778279889a6
kwon@kwon-Virtual-Machine:~/docker$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS             PORTS              NAMES
1778279889a6       centos:7           "/bin/bash"        About a minute ago
Up 11 seconds
unruffled_banzai
kwon@kwon-Virtual-Machine:~/docker$ docker attach centos:7
Error: No such container: centos:7
kwon@kwon-Virtual-Machine:~/docker$ docker attach 1778279889a6
[root@1778279889a6 /]# ps
PID TTY          TIME CMD
  1 pts/0        00:00:00 bash
 14 pts/0        00:00:00 ps
[root@1778279889a6 /]#
[root@1778279889a6 /]#
[root@1778279889a6 /]#
[root@1778279889a6 /]# exit
exit
kwon@kwon-Virtual-Machine:~/docker$
```

\$exit 입력 or [Ctrl]+[D]

# 3.1 자주 사용하는 Docker 명령어

## Docker 컨테이너 만들고 실행하기

```
root@81008b5a93a3: /
File Edit View Search Terminal Help
kwon@kwon-Virtual-Machine:~/docker$ docker s $docker run [옵션] [이미지 이름]:[태그]
1778279889a6
kwon@kwon-Virtual-Machine:~/docker$ docker attach 1778279889a6
[root@1778279889a6 /]# exit
exit
kwon@kwon-Virtual-Machine:~/docker$ docker run -i -t ubuntu:14.04
Unable to find image 'ubuntu:14.04' locally
14.04: Pulling from library/ubuntu
2e6e20c8e2e6: Pull complete
30bb187ac3fc: Pull complete
b7a5bcc4a58a: Pull complete
Digest: sha256:ffc76f71dd8be8c9e222d420dc96901a07b61616689a44c7b3ef6a10b7213de4
Status: Downloaded newer image for ubuntu:14.04
root@81008b5a93a3:/#
```

- Docker 의 컨테이너 생성 -> 실행 -> 들어가기 까지 한번에 해주는 명령어.

# 3.1 자주 사용하는 Docker 명령어

## Docker 컨테이너 만들고 실행하기

### - 옵션

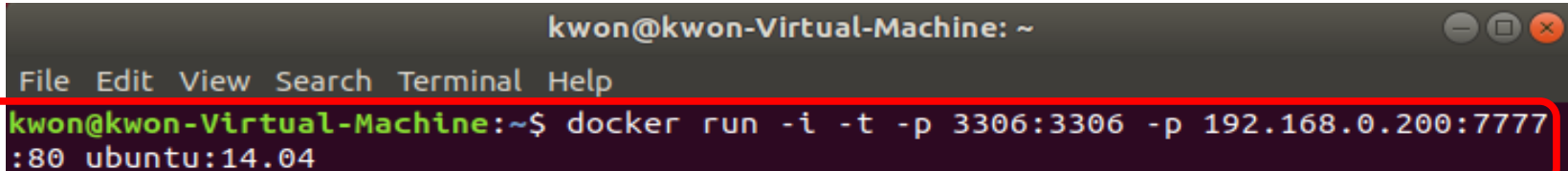
옵션	설명
-d	백그라운드 컨테이너로 실행
-p	포트 포워딩 지정 ex) -p [호스트 포트]:[컨테이너 포트] 특정 IP를 사용할시 ex) -p [IP주소]:[호스트 포트]:[컨테이너 포트]
--name	컨테이너 이름



# 3.1 자주 사용하는 Docker 명령어

## Docker 컨테이너 만들고 실행하기

- Docker 컨테이너 외부에 노출하고 싶을 경우

A terminal window titled 'kwon@kwon-Virtual-Machine: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command 'docker run -i -t -p 3306:3306 -p 192.168.0.200:7777 :80 ubuntu:14.04' is entered and highlighted with a red rectangle.

```
kwon@kwon-Virtual-Machine: ~  
File Edit View Search Terminal Help  
kwon@kwon-Virtual-Machine:~$ docker run -i -t -p 3306:3306 -p 192.168.0.200:7777 :80 ubuntu:14.04
```

```
$docker run -i -t -p [포트] -p [IP주소] [이미지이름]
```

- 이후 다른 컨테이너에서 apache 설치 후 로컬 호스트의 IP로 웹 브라우저를 통해서 접근하면 Apache가 보인다.

```
# 타 컨테이너에서 아파치 설치방법  
$ apt-get update #apt-get 업데이트  
$ apt-get install apache2 -y #아파치2 설치  
$ service apache2 restart #apache 실행
```

## 3.1 자주 사용하는 Docker 명령어

### Docker 컨테이너 정지하기

\$docker container stop [컨테이너 ID or 컨테이너 명]

File Edit View Search Terminal Help

kwon@kwon-Virtual-Machine:~/docker\$ docker container stop adoring\_hermann

^C

kwon@kwon-Virtual-Machine:~/docker\$ docker container ps

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	

kwon@kwon-Virtual-Machine:~/docker\$

# 3.1 자주 사용하는 Docker 명령어

## Docker 컨테이너 재시작하기

```
kwon@kwon-Virtual-Machine: ~/docker
File Edit View Search Terminal Help
kwon@kwon-Virtual-Machine:~/docker$ docker container stop adoring_hermann
^C
$docker container restart [컨테이너 ID or 컨테이너 명] CREATED
STATUS          PORTS          NAMES
kwon@kwon-Virtual-Machine:~/docker$ docker container restart adoring_hermann
adoring_hermann
kwon@kwon-Virtual-Machine:~/docker$
```

# 3.1 자주 사용하는 Docker 명령어

## Docker 컨테이너 삭제

```
kwon@kwon-Virtual-Machine: ~/docker
File Edit View Search Terminal Help
kwon@kwon-Virtual-Machine:~/docker$ docker container stop adoring_hermann
adoring_hermann
kwon@kwon-Virtual-Machine:~/docker$ docker container rm adoring_hermann
adoring_hermann
kwon@kwon-Virtual-Machine:~/docker$
```

\$docker container rm [옵션] [컨테이너 ID or 컨테이너 명]

- 옵션

옵션	설명
-f	현재 실행중인 컨테이너 삭제

# 3.1 자주 사용하는 Docker 명령어

Docker 컨테이너 삭제(실행중이 아닌 모든 컨테이너 삭제)

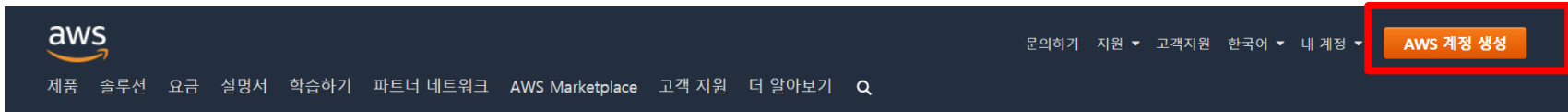
```
kwon@kwon-Virtual-Machine: ~/docker
File Edit View Search Terminal Help
kwon@kwon-Virtual-Machine:~/docker$ docker container stop adoring_hermann
adoring_hermann
kwon@kwon-Virtual-Machine:~/docker$ docker container rm adoring_hermann
adoring_hermann
kwon@kwon-Virtual-Machine:~/docker$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
81008b5a93a3d67cf299102d0d637c67795e6343f424241376e3ee7d7ae65f93
8b23d5fb6b7bd40e17cfd5ce5f03f5c95cebd5e7607d6e59dc28a606f2fe4ded
71cf833f209925d38252001ea85f2e3fea81
47cf3acc2c92e91b503411084cafb046ba9f
$docker container prune
Total reclaimed space: 23B
```

# 4 AWS EC2에서 도커 사용

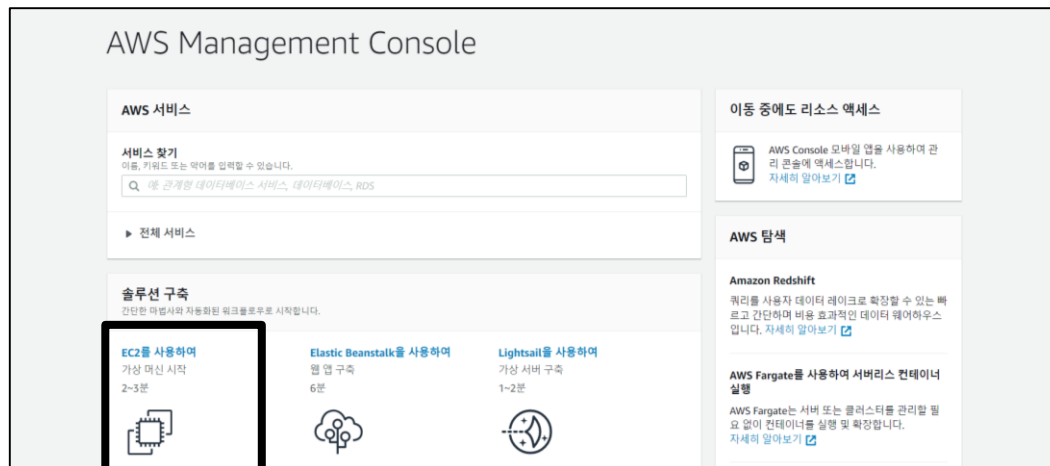
\*AWS의 EC2의 인스턴스에서 docker를 사용해 apache 웹 서버를 구축합니다.

## 1. EC2 인스턴스 생성

- 1) AWS 사이트에 접속합니다. <https://aws.amazon.com/ko/>
- 2) 상단 오른쪽에 주황색 버튼을 눌러 계정생성 또는 로그인을 합니다.



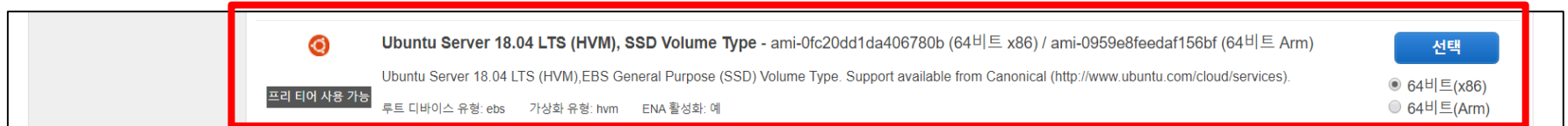
- 3) 솔루션 구축 – 'EC2를 사용하여 가상 머신 시작'을 클릭합니다.



# 4.1 AWS EC2에서 도커 사용

## 1. EC2인스턴스 생성

4) AMI를 선택하는 창에서 Ubuntu 18.04 버전을 선택합니다.



5) 프리티어 사용 가능이라고 적혀 있는 것을 선택 후 검토시작 버튼을 클릭합니다.

\*그 외의 인스턴스 유형은 무료계정에서는 사용이 불가능 합니다.

단계 2: 인스턴스 유형 선택

Amazon EC2는 각 사용 사례에 맞게 최적화된 다양한 인스턴스 유형을 제공합니다. 인스턴스는 애플리케이션을 실행할 수 있는 가상 서버입니다. 이러한 인스턴스에는 CPU, 메모리, 스토리지 및 네트워킹 용량의 다양한 조합이 있으며, 애플리케이션에 사용할 적절한 리소스 조합을 유연하게 선택할 수 있습니다. 인스턴스 유형과 이 인스턴스 유형이 컴퓨팅 요건을 충족하는 방식에 대해 [자세히 알아보기](#).

필터링 기준: 모든 인스턴스 유형 현재 세대 열 표시/숨기기

현재 선택된 항목: t2.micro (Variable ECU, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB 메모리, EBS 전용)

	그룹	유형	vCPUs	메모리 (GiB)	인스턴스 스토리지 (GB)	EBS 최적화 사용 가능	네트워크 성능	IPv6 지원
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS 전용	-	낮음에서 중간	예
<input checked="" type="checkbox"/>	General purpose	t2.micro 프리 티어 사용 가능	1	1	EBS 전용	-	낮음에서 중간	예
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS 전용	-	낮음에서 중간	예
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS 전용	-	낮음에서 중간	예

# 4.1 AWS EC2에서 도커 사용

## 1. EC2 인스턴스 생성

6) EC2 인스턴스에 apache 웹서버를 열었을 때 80번 포트가 열려 있어야 해당 웹 서버에 접근할 수 있으므로 80번 포트를 열어 주기 위해 보안 그룹 편집을 합니다.

The screenshot shows the AWS Management Console interface for creating an EC2 instance. The top navigation bar includes the AWS logo, service categories, and user information. The main content area displays the '단계 7: 인스턴스 시작 검토' (Step 7: Review Instance) page. It includes a progress bar with steps 1 through 7, where step 7 is currently active. The page provides a summary of the instance configuration, including the AMI (Ubuntu Server 18.04 LTS), instance type (t2.micro), and the security group (launch-wizard-7). A red box highlights the '보안 그룹' (Security Groups) section, and a red arrow points to the '보안 그룹 편집' (Edit Security Group) link. The bottom of the page features buttons for '취소' (Cancel), '이전' (Previous), and '시작하기' (Launch).

aws 서비스 리소스 그룹

1. AMI 선택 2. 인스턴스 유형 선택 3. 인스턴스 구성 4. 스토리지 추가 5. 태그 추가 6. 보안 그룹 구성 7. 검토

단계 7: 인스턴스 시작 검토

인스턴스 시작 세부 정보를 검토하십시오. 이전으로 돌아가서 각 섹션에 대한 변경 내용을 편집할 수 있습니다. 키 페어를 인스턴스에 할당하고 시작 프로세스를 완료하려면 [시작]을 클릭합니다.

AMI 세부 정보 AMI 편집

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0fc20dd1da406780b

프리 티어 사용 가능 Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

인스턴스 유형 인스턴스 유형 편집

인스턴스 유형	ECU	vCPUs	메모리 (GiB)	인스턴스 스토리지 (GB)	EBS 최적화 사용 가능	네트워크 성능
t2.micro	Variable	1	1	EBS 전용	-	Low to Moderate

보안 그룹 보안 그룹 편집

보안 그룹 이름 launch-wizard-7

설명 launch-wizard-7 created 2020-04-14T22:53:40.272+09:00

유형	프로토콜	포트 범위	소스	설명
----	------	-------	----	----

이 보안 그룹에 규칙이 없습니다.

인스턴스 세부 정보 인스턴스 세부 정보 편집

취소 이전 시작하기

의견 한국어 © 2008 - 2020, Amazon Web Services, Inc. 또는 계열사. All rights reserved. 개인 정보 보호 정책 이용 약관



# 4.1 AWS EC2에서 도커 사용

## 1. EC2 인스턴스 생성

7) 아래 화면과 같이 HTTP 인바운드 규칙을 추가 해 준 후 이전 페이지로 돌아가 '시작하기' 버튼을 누릅니다.

인바운드 규칙 정보

유형 정보	프로토콜 정보	포트 범위 정보	소스 정보	설명 - 선택 사항 정보
HTTP	TCP	80	내 IP	
SSH	TCP	22	사용자 ...	

규칙 추가

참고: 기존 규칙을 편집하면 편집된 규칙이 삭제되고 새 세부 정보가 포함된 새 규칙이 생성됩니다. 그러면 새 규칙이 생성될 때까지 해당 규칙에 의존하는 트래픽이 매우 짧은 시간 동안 삭제됩니다.

- http : 내 ip로 80번 포트에 접근할 수 있게 함

# 4.1 AWS EC2에서 도커 사용

## 1. EC2 인스턴스 생성

8) 새 키 페어 생성을 누르고 키 페어 이름을 입력하신 후 키 페어를 다운받습니다.

\*다운받은 키 페어는 편한 위치(ex.바탕화면, 개인 폴더)등에 옮겨 두면 됩니다.

9) 인스턴스 시작 버튼을 누릅니다.

\*키 페어로 SSH 프로토콜을 통해 EC2 인스턴스로 만들어진 서버에 접속할 수 있습니다.

\*SSH프로토콜이란 원격으로 연결을 가능하게 하는 네트워크 접속 도구입니다.

\*키 페어를 잃어버리면 서버에 접속이 불가능 해집니다. 인스턴스 사용 중에는 해당 키페어를 절대 삭제하면 안됩니다.

# 4.1 AWS EC2에서 도커 사용

## 1. EC2 인스턴스 생성

10) 인스턴스 생성이 완료되었습니다.

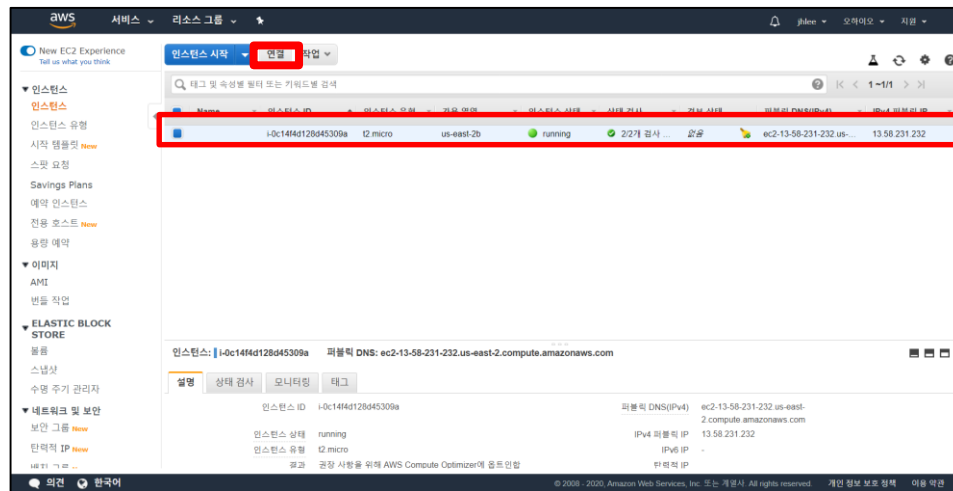
11) 인스턴스 보기를 눌러 인스턴스 관리 페이지로 넘어갑니다.

The screenshot shows the AWS Management Console interface. At the top, there's a navigation bar with the AWS logo, service categories, and user information. The main content area is titled '시작 상태' (Start Status). It features a green success message: '지금 인스턴스를 시작 중입니다.' (Starting instance now), followed by the instance ID 'i-0c14f4d128d45309a' and a link to '시작 로그 보기' (View start logs). Below this is a blue information box titled '예상 요금 알림 받기' (Receive estimated cost notifications), explaining that users will receive email alerts if their usage exceeds a set budget. The section '인스턴스에 연결하는 방법' (How to connect to the instance) provides instructions on monitoring the instance's state and connecting via SSH or RDP. It lists helpful resources like 'Linux 인스턴스에 연결하는 방법' and 'Amazon EC2: 사용 설명서'. A list of actions to perform while the instance starts is provided, including checking status, EBS volume attachment, and security group management. At the bottom right, a red-bordered button labeled '인스턴스 보기' (View instance) is highlighted.

# 4.1 AWS EC2에서 도커 사용

## 2. 인스턴스에 연결하기

- 1) 방금 만든 인스턴스가 확인이 됩니다.
- 2) 인스턴스 연결을 위해 연결 버튼을 누릅니다.



\*인스턴스를 생성했다는 것은 아까전에 우리가 선택한 Ubuntu OS를 가진 클라우드 서버가 생성됐음을 의미합니다.  
(한 마디로 우리가 사용할 수 있는 컴퓨터가 하나 켜진 것입니다.)

# 4.1 AWS EC2에서 도커 사용

## 2. 인스턴스에 연결하기

※Putty, xshell, mobaxterm등의 터미널 에뮬레이터를 사용할 줄 알면 편한 툴을 이용하시면 됩니다.

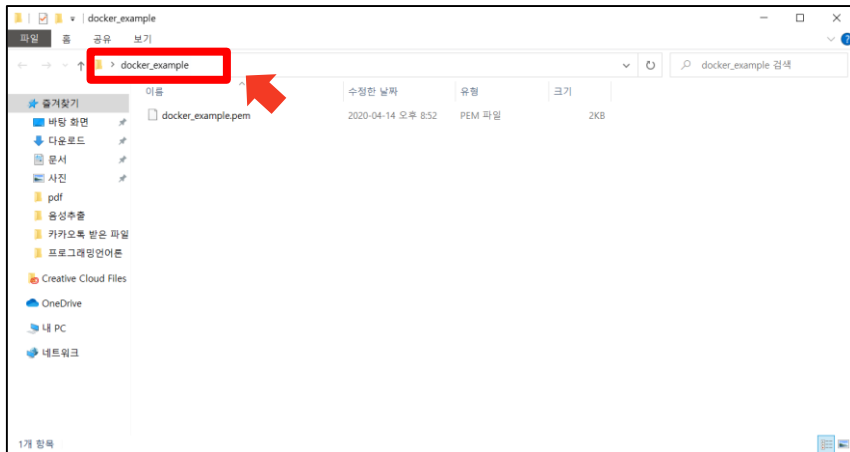
3) 인스턴스 관리창은 그대로 두고 명령 프롬프트를 관리자 권한으로 실행합니다.



# 4.1 AWS EC2에서 도커 사용

## 2. 인스턴스에 연결하기

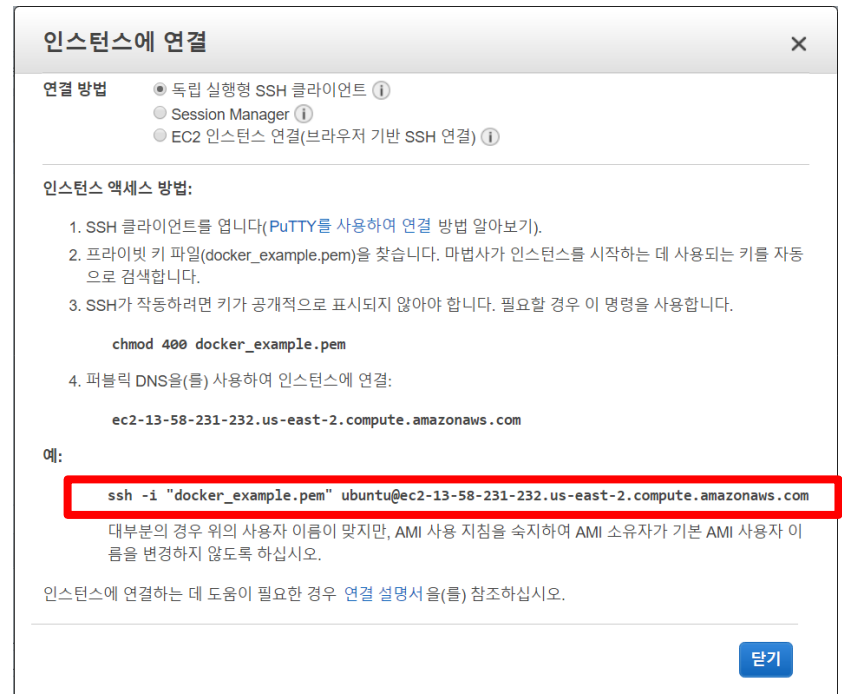
※Putty, xshell, mobaxterm등의 터미널 에뮬레이터를 사용할 줄 알면 편한 툴을 이용하시면 됩니다.



5) 인스턴스 관리창에서 연결을 눌러서 뜬 화면입니다. 빨간박스로 표시한 부분을 복사해 명령 프롬프트에 붙여넣어 줍니다.

4) 키페어를 저장해둔 폴더의 주소를 복사한 뒤 명령 프롬프트에 "cd 복사한 주소"를 입력합니다.

\*cd는 입력한 주소로 이동하는 명령어입니다.



# 4.1 AWS EC2에서 도커 사용

## 2. 인스턴스에 연결하기

```
root@ip-172-31-29-96: /home/ubuntu
Microsoft Windows [Version 10.0.19587.1000]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32> 4) cd C:\Users\JH\Desktop\docker_example

C:\Users\JH\Desktop\docker_example> 5) ssh -i "docker_example.pem" ubuntu@ec2-13-58-231-232.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1057-aws x86_64)

 * Documentation:  https://help.ubuntu.com
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1057-aws x86_64)

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

System information as of Tue Apr 14 14:02:34 UTC 2020

System load:  0.0               Processes:            94
Usage of /:   23.0% of 7.69GB   Users logged in:     0
Memory usage: 25%              IP address for eth0:  172.31.29.96
Swap usage:   0%               IP address for docker0: 172.17.0.1

77 packages can be updated.
50 updates are security updates.
```

\*인스턴스에 연결 성공, 이 다음부터 입력하는 것은 Ubuntu 명령어입니다.

# 4.1 AWS EC2에서 도커 사용

## 2. 인스턴스에 연결하기

6) `sudo passwd root`를 입력해 root의 비밀번호를 설정합니다.

7) `su root`을 입력한 후 6에서 설정한 비밀번호로 root로 로그인 합니다.

\*다른 계정의 경우 파일의 권한에 따라 접근할 수 없을 수도 있고 명령에 제약이 있을 수 있습니다.

```
ubuntu@ip-172-31-29-96:~$ sudo passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
ubuntu@ip-172-31-29-96:~$ su root
Password:
root@ip-172-31-29-96:/home/ubuntu#
```



# 4.1 AWS EC2에서 도커 사용

## 3. 도커 설치

1) 본 튜토리얼 21-28p의 설명을 따라 우분투에 도커를 설치합니다.

\*아래의 코드를 순서대로 입력

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
sudo apt update
sudo apt install docker-ce
apt-cache policy docker-ce
sudo systemctl status docker //정상적으로 작동하는지 확인
```

\*후에 docker 로그인에 안되어 있다고 뜨면 창에 docker login을 입력하여 로그인을 합니다.

# 4.1 AWS EC2에서 도커 사용

## 4. Apache 이미지 파일 생성 및 실행

1) 명령 프롬프트에 아래의 코드를 입력합니다.

```
cd home/ubuntu      // home/ubuntu로 이동
mkdir example        // 현재 경로에서 example이란 이름 가진 폴더 생성
cd example           //example 폴더로 이동
vi Dockerfile         //example 폴더안에 Dockerfile이란 이름을 가진 파일이 있으면 그 파일을 문서편집
                     기(메모장 같은것)로 열고 없으면 새로 만든다
```

2) Apache container를 위한 Dockerfile 작성

```
root@ip-172-31-29-96: /home/ubuntu/example
FROM ubuntu:18.04
MAINTAINER cbnu <email@cbnu.ac.kr>

RUN apt-get update
RUN apt-get install -y apache2

EXPOSE 80
CMD ["apachectl", "-D", "FOREGROUND"]
```

- 우분투 18.04 이미지 기반
- Dockerfile 작성자 정보
- 실행시킬 명령어 입력(apt-get 업데이트후 apache2 설치)
- 80번 포트를 사용한다고 표기
- 컨테이너가 계속 돌아가게 하기 위한 설정
- 작성 완료하면 <ESC버튼>+ :wq 입력해 저장 후 종료

# 4.1 AWS EC2에서 도커 사용

## 4. Apache 이미지 파일 생성 및 실행

### 3) 이미지 파일 생성(빌드)

```
docker build -t apache_example . // -t 옵션은 이미지 파일에 태그(이름)를 붙여준다.  
// 이 이미지의 이름은 apache_example이다.  
// 맨 마지막 .은 현재 디렉토리를 의미한다. 현재 디렉토리의 도커파일을  
// 빌드하도록 경로를 지정해 준 것이다.
```

```
root@ip-172-31-29-96:/home/ubuntu/example# docker build -t apache_example .  
Sending build context to Docker daemon 2.048kB  
Step 1/6 : FROM ubuntu:18.04  
18.04: Pulling from library/ubuntu  
5bed26d33875: Pull complete  
f11b29a9c730: Pull complete  
930bda195c84: Pull complete  
78bf9a5ad49e: Pull complete  
Digest: sha256:bec5a2727be7fff3d308193cfde3491f8fba1a2ba392b7546b43a051853a341d  
Status: Downloaded newer image for ubuntu:18.04  
--> 4e5021d210f6  
Step 2/6 : MAINTAINER cbnu <email@cbnu.ac.kr>  
--> Running in c29cd18e8789  
Removing intermediate container c29cd18e8789  
--> f50f1b178c0e  
Step 3/6 : RUN apt-get update  
--> Running in d2573c3748ee  
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
```

성공 ▶

```
Successfully built b03b5ab07037  
Successfully tagged apache_example:latest  
root@ip-172-31-29-96:/home/ubuntu/example# clear
```

# 4.1 AWS EC2에서 도커 사용

## 4. Apache 이미지 파일 생성 및 컨테이너 실행

4) 아래의 코드를 입력해 컨테이너를 실행합니다.

```
docker run -d -p 80:80 --name apache apache_example
// -d 옵션 : 컨테이너를 백그라운드로 실행
// -p 옵션(80:80) : 호스트의 80번 포트와 컨테이너의 80번 포트를 연결하고 외부에 노출
// --name apache : 컨테이너 이름을 apache로 지정
// docker run apache_example을 하면 단순히 apache_example 이미지가 실행되고 종료.
```

5) docker ps 명령어로 컨테이너가 잘 실행되고 있는지 확인합니다.

```
ubuntu 18.04 4e5021d21016 3 weeks ago 64.2MB
root@ip-172-31-29-96:/home/ubuntu/example# docker run -d -p 80:80 --name apache apache_example
82234665441c3ed09f2f5600f0a2d29239091585a6980bf6f218ef1328bbb60c
root@ip-172-31-29-96:/home/ubuntu/example# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
82234665441c	apache_example	"apachectl -D FOREGR..."	4 seconds ago	Up 3 seconds	0.0.0.0:80->80/tcp	apache

```
root@ip-172-31-29-96:/home/ubuntu/example#
```

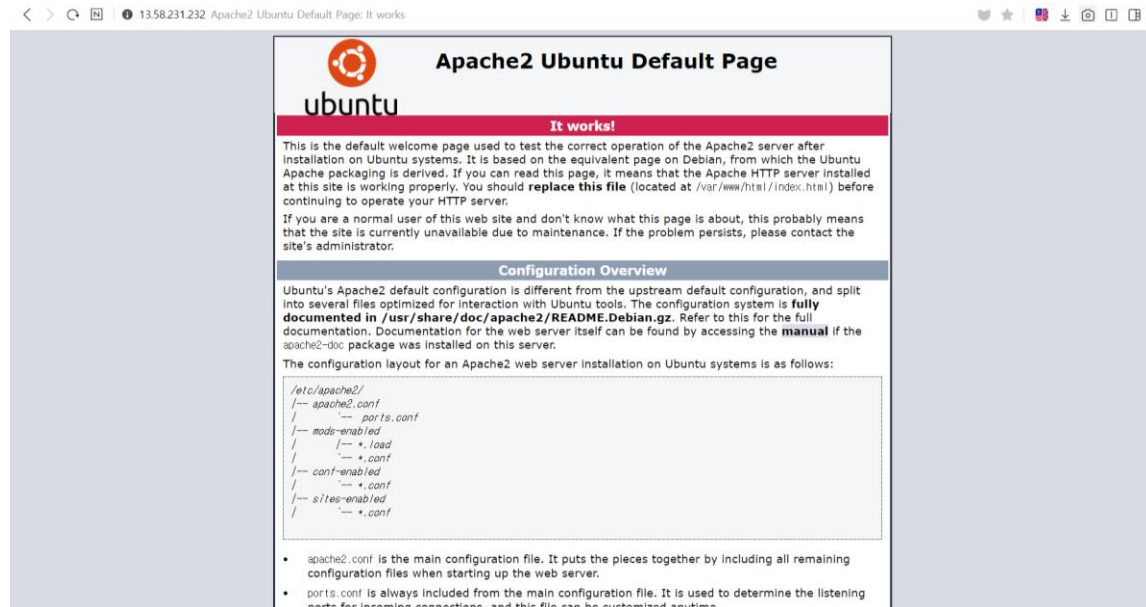
# 4.1 AWS EC2에서 도커 사용

## 5. 웹서버 접속

1) 인스턴스의 아이피:80을 웹 브라우저의 주소창에 입력합니다.

\*인스턴스 관리페이지 하단의 설명란 ipv4퍼블릭 ip 옆에 적힌 ip를 사용하면 됩니다.

2) 아래와 같은 화면이 뜨면 apache웹서버가 성공적으로 열린 것입니다.



주의사항 : AWS는 1년간 무료입니다. 인스턴스 사용량에 따라 요금이 부과 될 수 있으니 꼭 인스턴스를 종료해주시기 바랍니다.



**충북대학교**  
**SW중심대학사업단**