

# 도커 환경 구성 및 설정

# 도커 환경 구성 및 설정

## 우분투 환경 구축

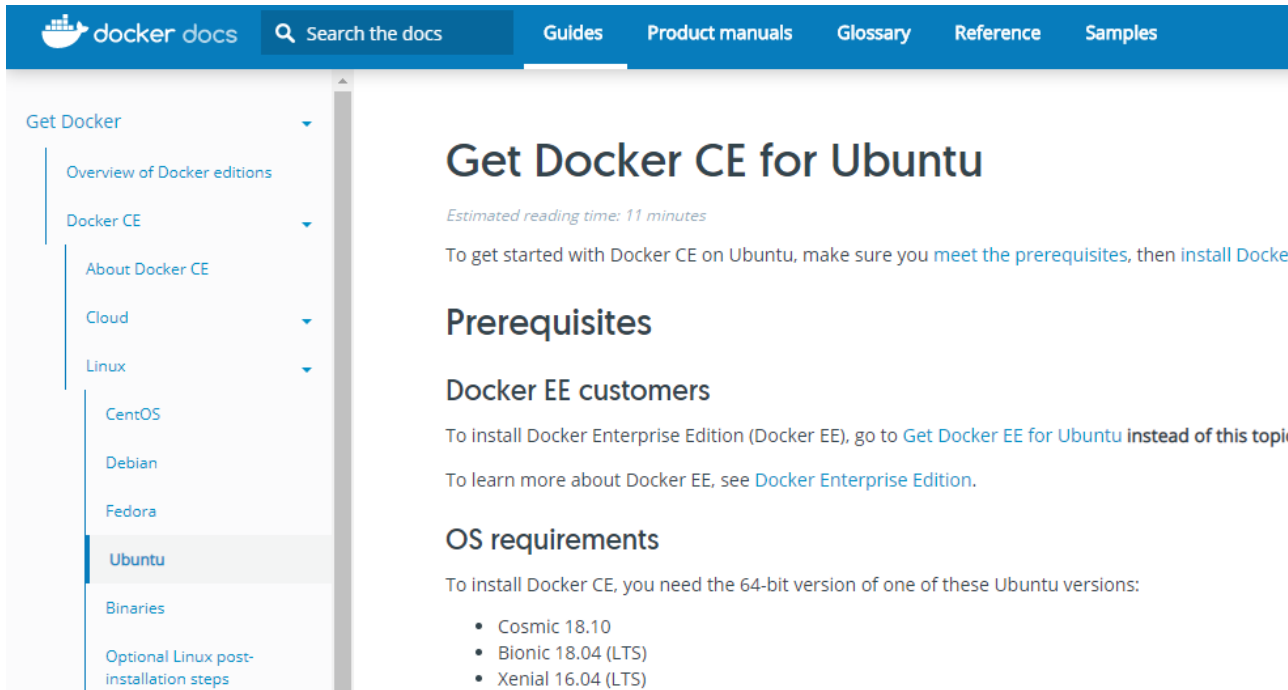
- Ubuntu를 사용
- VirtualBox에서 Ubuntu 설치 후 사용
- 시간 관계상 이미 설치된 파일을 사용
- 설치하는 다음 영상을 참고 ( <https://youtu.be/gj1sU2Qs> )



# 도커 환경 구성 및 설정

## 🚀 Docker CE 설치

- <https://docs.docker.com/install/linux/docker-ce/ubuntu/>
- 필요에 따라 다양한 방법으로 설치
- 대부분은 Docker의 저장소를 설정하여 쉽게 설치
- 또 다른 방법으로 DEB 패키지를 다운로드하여 수동으로 설치
- 테스트 및 개발 환경에서 일부 사용자는 자동화된 스크립트를 사용



The screenshot shows the Docker documentation website. The top navigation bar includes the Docker logo, 'docker docs', a search bar, and links to 'Guides', 'Product manuals', 'Glossary', 'Reference', and 'Samples'. The left sidebar shows a tree view with 'Get Docker' expanded, containing 'Overview of Docker editions', 'Docker CE' (expanded), 'Cloud', and 'Linux'. Under 'Docker CE', there are links for 'About Docker CE', 'CentOS', 'Debian', 'Fedora', 'Ubuntu' (highlighted), 'Binaries', and 'Optional Linux post-installation steps'. The main content area is titled 'Get Docker CE for Ubuntu' with an estimated reading time of 11 minutes. It includes a paragraph about prerequisites, a section for Docker EE customers, and a section for OS requirements listing Ubuntu versions: Cosmic 18.10, Bionic 18.04 (LTS), and Xenial 16.04 (LTS).

docker docs Search the docs Guides Product manuals Glossary Reference Samples

### Get Docker

- Overview of Docker editions
- Docker CE
  - About Docker CE
  - Cloud
  - Linux
    - CentOS
    - Debian
    - Fedora
    - Ubuntu**
    - Binaries
    - Optional Linux post-installation steps

## Get Docker CE for Ubuntu

Estimated reading time: 11 minutes

To get started with Docker CE on Ubuntu, make sure you [meet the prerequisites](#), then [install Docker](#).

### Prerequisites

#### Docker EE customers

To install Docker Enterprise Edition (Docker EE), go to [Get Docker EE for Ubuntu](#) instead of this topic.

To learn more about Docker EE, see [Docker Enterprise Edition](#).

### OS requirements

To install Docker CE, you need the 64-bit version of one of these Ubuntu versions:

- Cosmic 18.10
- Bionic 18.04 (LTS)
- Xenial 16.04 (LTS)

# 도커 환경 구성 및 설정

## Docker CE 설치

- 대부분은 [Docker의 저장소를 설정](#)하여 apt-get으로 설치
- `sudo apt install docker.io`

```
$ sudo apt-get update
$ sudo apt-get install \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg-agent \
  software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo apt-key fingerprint 0EBFCD88
$ sudo add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) \
  stable"
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

# 도커 환경 구성 및 설정

## 도커 명령어 확인

- 터미널에 “docker”를 쓰고 탭을 눌러 도커 관련 커맨드 확인

```
server1@server1-VirtualBox: ~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
server1@server1-VirtualBox:~$ docker  
docker          docker-init    docker-proxy   dockerd  
server1@server1-VirtualBox:~$
```

# 도커 환경 구성 및 설정

## Docker CE와 Docker EE

- 2017년 까지는 도커는 단일 제품으로 개발됐으나 1.13.1 버전 이후로는 엔터프라이즈 에디션과 커뮤니티 에디션 두 가지 버전으로 나누어 공개
- 현재는 버전 번호 대신의 연도 번호를 붙여서 버전 이름을 명시(예 17.03)
- EE는 유료로 제공하는 프로그램으로 도커사의 기술 지원을 받을 수 있으며 다양한 OS(MS 서버2016, 오라클 리눅스, RHEL)를 설치 가능

엔터프라이즈에서 제공하는 세 가지 버전(출처: 도커/쿠버네티스를 활용한 컨테이너 개발 실전 입문)

기능	Basic	Standard	Advanced
보안성 높은 네트워크 및 스토리지를 갖춘 컨테이너 엔진	○	○	○
인증된 환경, 이미지, 플러그인 제공	○	○	○
도커 이미지 관리(프라이빗 레지스트리 등)		○	○
통합된 컨테이너를 이용한 애플리케이션 관리		○	○
확장 RBAC와 LDAP/AD 지원		○	○
기밀 정보 관리 및 이미지 서명		○	○
안전한 멀티 테넌시			○
규칙 기반 도커 이미지 태그 부여 및 배포			○
이미지 레지스트리 미러링 지원			○
도커 이미지 보안 스캔			○

# 도커 컨테이너 기본 활용



# 도커 컨테이너 기본 활용

## 간단한 HTTP 톰캣 서버 구현하기

- 간단한 애플리케이션을 구현하면서 도커의 사용방법을 하나씩 배워보자.
- 가장 먼저 할 일은 적절한 이미지를 찾는 일

## 이미지 검색하기

- 도커에서는 search라는 명령어로 내가 원하는 이미지가 이미 레지스트리에 존재하는 지 확인 가능
- `sudo docker search "tomcat"`

```
server1@server1-VirtualBox:~$ sudo docker search tomcat
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
tomcat	Apache Tomcat is an open source implementati...	2396	[OK]	
tomee	Apache TomEE is an all-Apache Java EE certif...	66	[OK]	
dordoka/tomcat	Ubuntu 14.04, Oracle JDK 8 and Tomcat 8 base...	53		[OK]
davidcaste/alpine-tomcat	Apache Tomcat 7/8 using Oracle Java 7/8 with...	34		[OK]
bitnami/tomcat	Bitnami Tomcat Docker Image	28		[OK]
cloudesire/tomcat	Tomcat server, 6/7/8	14		[OK]
meirwa/spring-boot-tomcat-mysql-app	a sample spring-boot app using tomcat and My...	12		[OK]
tutum/tomcat	Base docker image to run a Tomcat applicatio...	11		
aallam/tomcat-mysql	Debian, Oracle JDK, Tomcat & MySQL	11		[OK]
jeanblanchard/tomcat	Minimal Docker image with Apache Tomcat	8		
arm32v7/tomcat	Apache Tomcat is an open source implementati...	6		
rightctrl/tomcat	CentOS , Oracle Java, tomcat application ssl...	4		[OK]
amd64/tomcat	Apache Tomcat is an open source implementati...	2		
arm64v8/tomcat	Apache Tomcat is an open source implementati...	2		
99taxi/tomcat7	Tomcat7	1		[OK]
i386/tomcat	Apache Tomcat is an open source implementati...	1		
ppc64le/tomcat	Apache Tomcat is an open source implementati...	1		
camptocamp/tomcat-logback	Docker image for tomcat with logback integra...	1		[OK]
s390x/tomcat	Apache Tomcat is an open source implementati...	0		
1and1internet/debian-9-java-8-tomcat-8.5	Our tomcat 8.5 image	0		[OK]



# 도커 컨테이너 기본 활용

## 도커 레지스트리!

- 도커 레지스트리에는 사용자가 사용할 수 있도록 데이터베이스를 통해 Image를 제공해주고 있음
- 누구나 이미지를 만들어 푸시할 수 있으며 푸시된 이미지는 다른 사람들에게 공유 가능

## Basic taxonomy in Docker

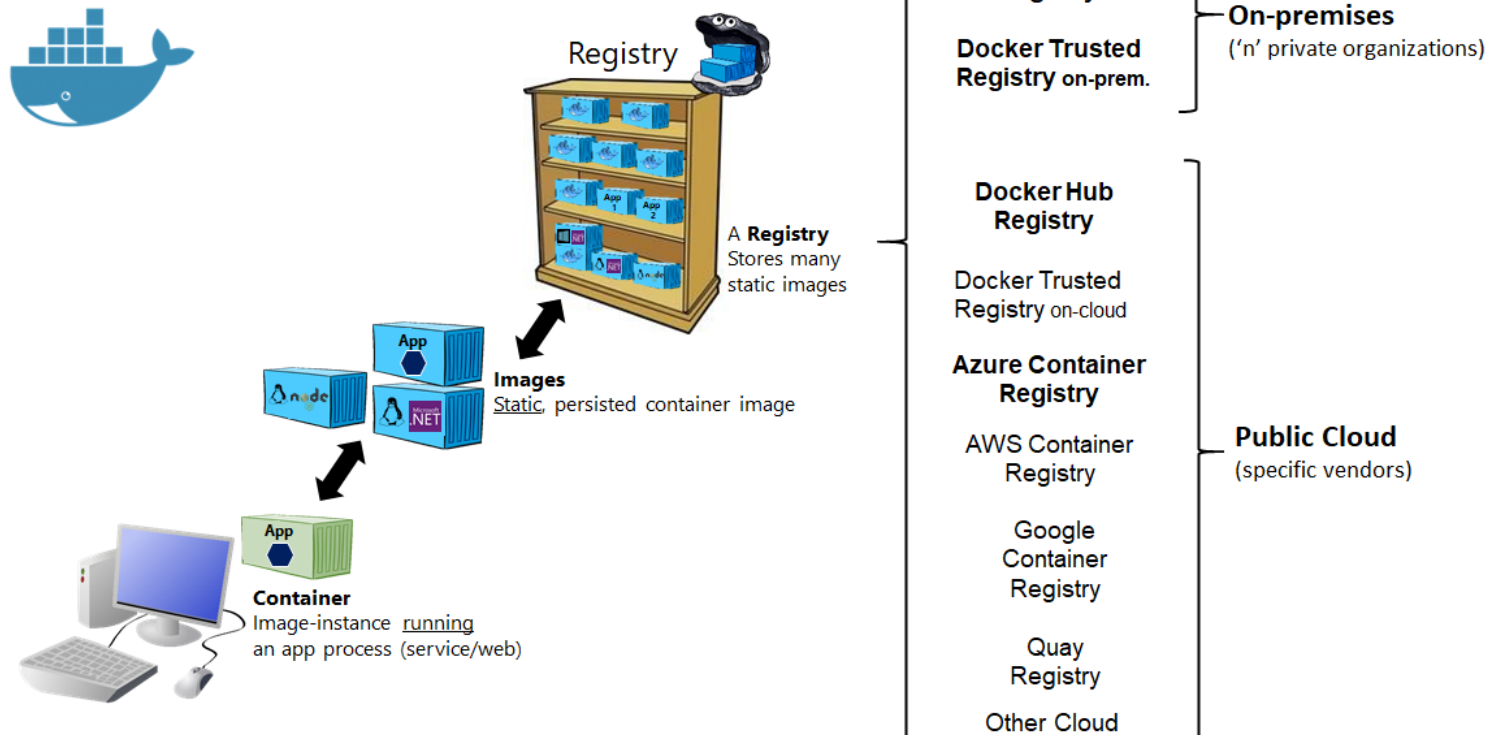
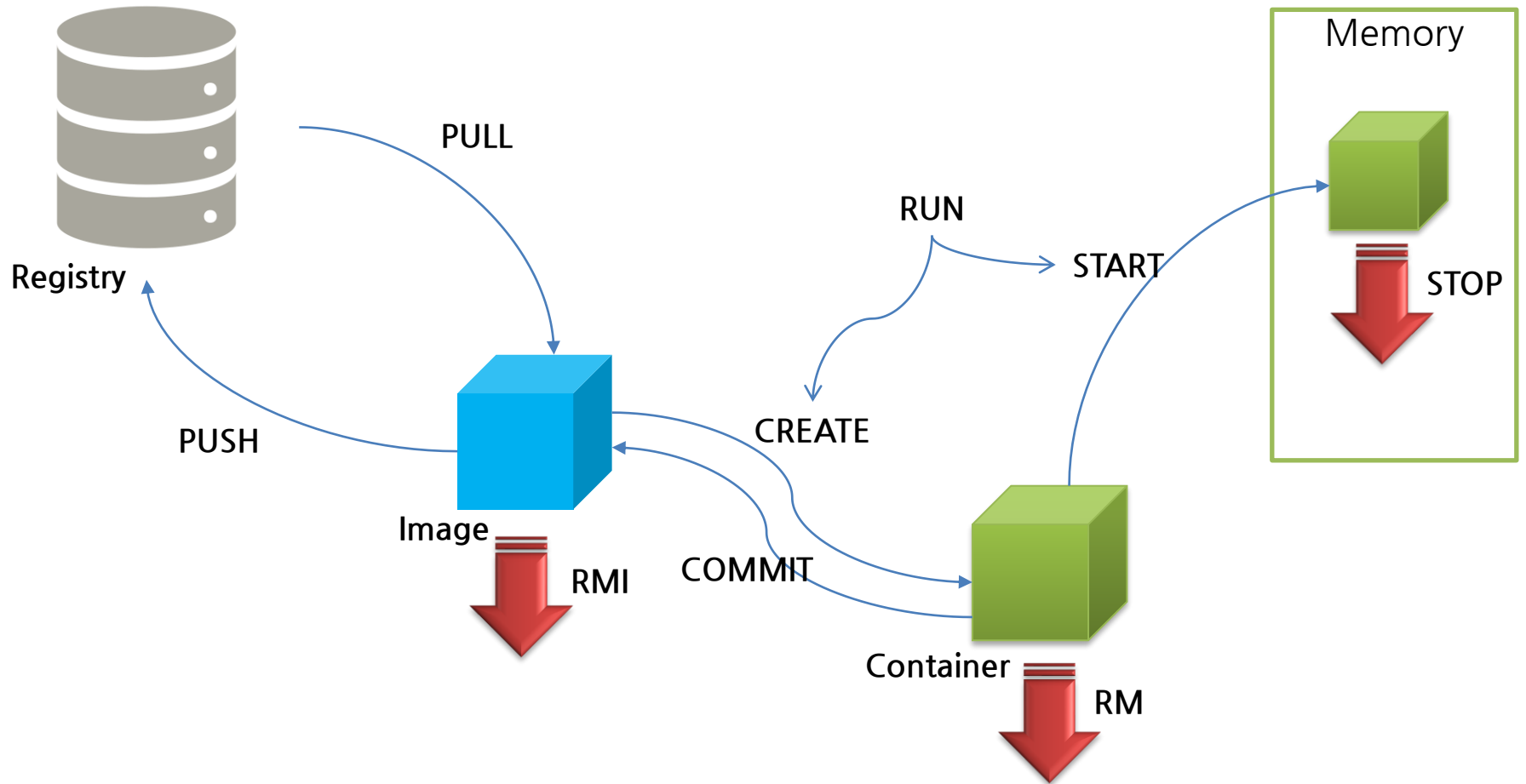


그림 출처: <https://github.com/dotnet-architecture/eShopModernizing/wiki/03.-Publishing-your-Windows-Container-images-into-a-Docker-Registry>

# 도커 컨테이너 기본 활용

## 도커의 라이프 사이클



# 도커 컨테이너 기본 활용

## 도커 레지스트리로부터 이미지를 다운로드 받기

- `sudo docker pull <이미지 이름>`

```
server1@server1-VirtualBox:~$ sudo docker pull tomcat
Using default tag: latest
latest: Pulling from library/tomcat
c5e155d5a1d1: Pull complete
221d80d00ae9: Pull complete
4250b3117dca: Pull complete
d1370422ab93: Pull complete
deb6b03222ca: Pull complete
9cdea8d70cc3: Pull complete
968505be14db: Pull complete
04b5c270ac81: Pull complete
301d76fcab1f: Pull complete
57ca7a0b9e79: Pull complete
3c1d6826d7a3: Pull complete
Digest: sha256:7cdf9dca1472da80e7384403c57b0632753a3a5cdf4f310fc39462e08af8ef39
Status: Downloaded newer image for tomcat:latest
```

# 도커 컨테이너 기본 활용

## 도커 레지스트리로부터 다운로드 받은 이미지 확인하기

- `sudo docker images`

```
server1@server1-VirtualBox:~$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
tomcat	latest	3639174793ba	2 days ago	463MB

## 로컬에서 도커 이미지 삭제하기

- `sudo docker rmi tomcat`

```
server1@server1-VirtualBox:~$ sudo docker rmi tomcat
```

```
Untagged: tomcat:latest
```

```
Untagged: tomcat@sha256:7cdf9dca1472da80e7384403c57b0632753a3a5cdf4f310fc39462e08af8ef39
```

```
Deleted: sha256:3639174793bafbf0f22aeaf6096842f563de3154fe7c756d04966eca1fe4e55d
```

```
Deleted: sha256:ad683ea82dd8cc2ea298e08626031a01b3bf94a35714572774405f04daa0fea8
```

```
Deleted: sha256:c43fe4301f09c132eefa0707afd2995ade93975a3a7cd11c7ba61b616b5c59e2
```

```
Deleted: sha256:4abee72d794e2aadaa1fd461440964ec8306780c9e22b7b363e953730613e145
```

```
Deleted: sha256:5f01074ca9f442239d769dff326dc5004d8bda0365a4767fff8af8e5f58ee41
```

```
Deleted: sha256:fc82c5f00299a2e87402e3090f53e161801b9c6aa6c86121c36e7bb277c68cd0
```

```
Deleted: sha256:067619b0ac688892f0e9f086135168006bdc58bc40d902c570772250ea264b3c
```

```
Deleted: sha256:6b6450559d31f9c4e78c7ca449410807188a4d5f88fecb4db7c0f5efbbd9bc2c
```

```
Deleted: sha256:fa14a85b824e4d22bebf8a3787f80f7b13a87e63636efc630416350d3cb09c8d
```

```
Deleted: sha256:3030c8853205275070075b0ba780c0f57317206041b30232a00da3f0d40b80f3
```

# 도커 컨테이너 기본 활용

## 원가 느끼셨나요?

- 도커는 레이어 단위로 이미지를 관리하여 새로 업로드된 이미지에 대해서 모든 데이터를 저장하는 것이 아니라 변경 사항을 저장하여 효율적으로 데이터를 관리

다운로드할 때 여러 해시를 다운로드 받는 모습

```
server1@server1-VirtualBox:~$ sudo docker pull tomcat
Using default tag: latest
latest: Pulling from library/tomcat
c5e155d5a1d1: Pull complete
221d80d00ae9: Pull complete
4250b3117dca: Pull complete
d1370422ab93: Pull complete
deb6b03222ca: Pull complete
9cdea8d70cc3: Pull complete
968505be14db: Pull complete
04b5c270ac81: Pull complete
301d76fcab1f: Pull complete
57ca7a0b9e79: Pull complete
3c1d6826d7a3: Pull complete
Digest: sha256:7cdf9dca1472da80e7384403c57b0632753a3a5cdf4f310fc39462e08af8ef39
Status: Downloaded newer image for tomcat:latest
```

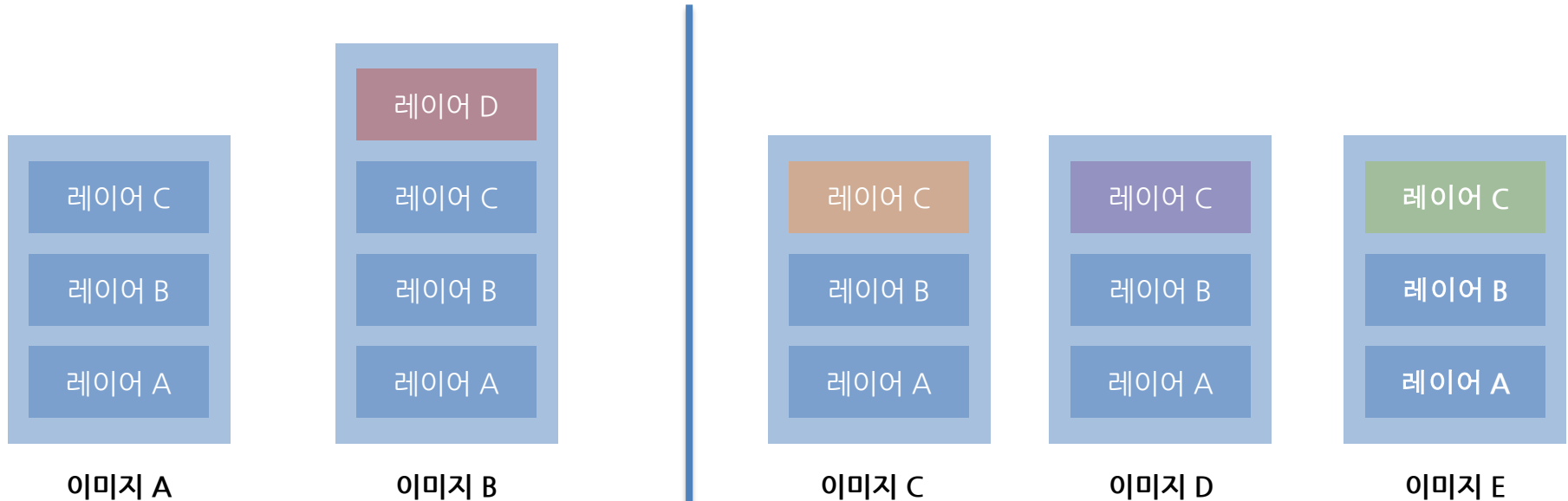
삭제할 때 여러 해시를 지우는 모습

```
server1@server1-VirtualBox:~$ sudo docker rmi tomcat
Untagged: tomcat:latest
Untagged: tomcat@sha256:7cdf9dca1472da80e7384403c57b0632753a3a5cdf4f310fc39462e08af8ef39
Deleted: sha256:3639174793bafb0f22aeaf6096842f563de3154fe7c756d04966eca1fe4e55d
Deleted: sha256:ad683ea82dd8cc2ea298e08626031a01b3bf94a35714572774405f04daa0fea8
Deleted: sha256:c43fe4301f09c132eefa0707afd2995ade93975a3a7cd11c7ba61b616b5c59e2
Deleted: sha256:4abee72d794e2aedaa1fd461440964ec8306780c9e22b7b363e953730613e145
Deleted: sha256:5f01074ca9f442239d769dffdc326dc5004d8bda0365a4767fff8af8e5f58ee41
Deleted: sha256:fc82c5f00299a2e87402e3090f53e161801b9c6aa6c86121c36e7bb277c68cd0
Deleted: sha256:067619b0ac688892f0e9f086135168006bdc58bc40d902c570772250ea264b3c
Deleted: sha256:6b6450559d31f9c4e78c7ca449410807188a4d5f88fecb4db7c0f5efbbd9bc2c
Deleted: sha256:fa14a85b824e4d22bebf8a3787f80f7b13a87e63636efc630416350d3cb09c8d
Deleted: sha256:a9a9c8853295275070975beba78ec0f573172e6e41b30232a00d8af0d49b8ef3
Deleted: sha256:ddf0293e8e23246803d265b158ffbb9453d925fe392b43515984815853e9121b
Deleted: sha256:f94641f1fe1f5c42c325652bf55f0513c881c86b620b912b15460e0bca07cc12
```

# 도커 컨테이너 기본 활용

## 이미지 추가와 삭제에 따른 스토리지 용량 확보

- 도커의 이미지를 생성할 때 만드는 manifest 파일의 명시된 내용을 사용하여 어떤 이미지를 쓰는지 알림
- 도커 이미지는 다양한 도커를 사용할 때 중복된 이미지를 생성하거나 삭제할 때 중복된 이미지에 대해서는 삭제하거나 생성하지 않음



이미지 A를 지운다 하더라도 이미지 B에서 레이어 A, B, C를 사용하고 있기 때문에 지워지지 않음

이미 존재하는 레이어 A, B는 새로 다운로드 받을 필요가 없음

# 도커 컨테이너 기본 활용

## 도커 이미지의 manifest 파일 확인

### ● sudo docker inspect tomcat

```
server1@server1-VirtualBox:~$ sudo docker inspect tomcat
```

```
[
  {
    "Id": "sha256:3639174793bafbf0f22aeaf6096842f563de3154fe7c756d04966eca1fe4e55d",
    "RepoTags": [
      "tomcat:latest"
    ],
    "RepoDigests": [
      "tomcat@sha256:7cdf9dca1472da80e7384403c57b0632753a3a5cdf4f310fc39462e08af8ef39"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2019-05-16T00:50:37.435017822Z",
    "Container": "e8a49f7e34b074702cea44f60c86402821e3f02afa74af10992ac2fd25ec7e7d",
    "ContainerConfig": {
      "Hostname": "e8a49f7e34b0",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "ExposedPorts": {
        "8080/tcp": {}
      },
      "Layers": [
        "sha256:f94641f1fe1f5c42c325652bf55f0513c881c86b620b912b15460e0bca07cc12",
        "sha256:ec62f19bb3aa1dcfacc9864be06f0af635c18021893d42598da1564beed97448",
        "sha256:2c719774c1e1c4b82c5b23bd40a7fc139aa5f0efddf7a969f72f8170c71dd058",
        "sha256:1b958b53b256f515bff1c037e7b10d77842ce7fc4512a0c02f0a15cd9344eb1d",
        "sha256:7d63f8777ebf22319f74d93672b47dd565ea1e11f03b5da8a7ea59c49ffc3879",
        "sha256:fe60061c6c4e7ff82d643b63e946798b2f160b4b903b1fd7ba24cedda4062efe",
        "sha256:d38f3d5a39fbfc13ec7c4b24a216c0d5d445228ac2695d5f43153b48c8236a49",
        "sha256:2b6c38ff31378735644fa7c81ed6313f2efd81ce1bfad56ce33c4bc4313f1e30",
        "sha256:f0e1731fd2867da96ee6371dc9fbc76a0a525beccc1135d5863877f61d408fc97",
        "sha256:c8bcc49b9925867b34533e64768dea2703313fd6a763bf1d664bb9b648d736e0",
        "sha256:f24d8b358bb1b7681a60f7bf0f2a4c0b08d696ad723c51d9e279692b96759682"
      ]
    }
  ]
}
```



# 도커 컨테이너 기본 활용

## 이미지를 실행하는 명령어

- `sudo docker run <이미지 이름>` (-d 옵션을 사용하면 백그라운드에서 동작)
- 운영체제 및 애플리케이션 관련 데이터의 로그와 함께 어떤 포트가 열렸는지 정보를 알려줌

```
server1@server1-VirtualBox:~$ sudo docker run tomcat
```

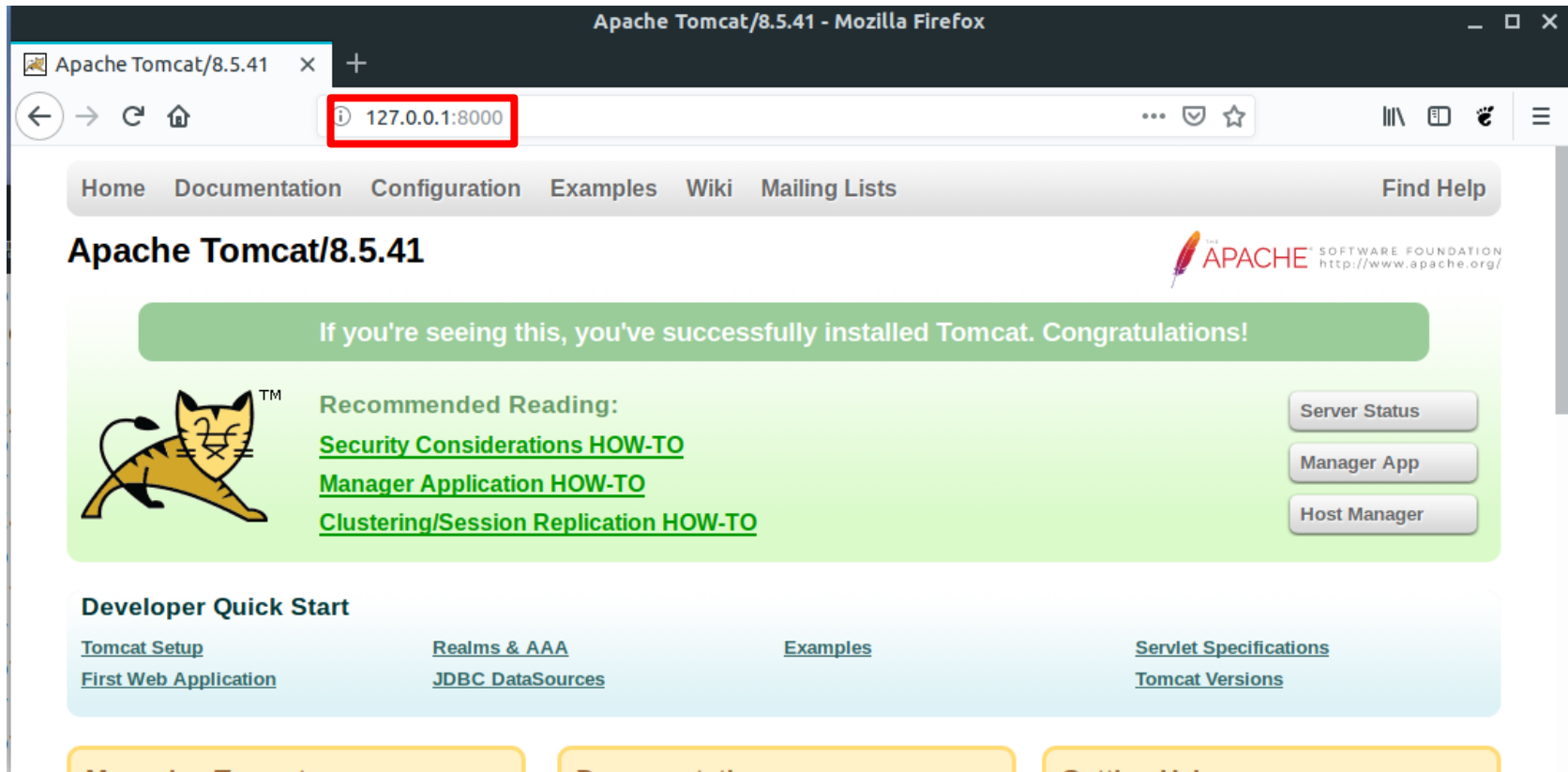
```
[sudo] server1의 암호:
```

```
18-May-2019 13:37:45.014 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version: Apache Tomcat/8.5.41
18-May-2019 13:37:45.017 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server built: May 4 2019 09:17:16 UTC
18-May-2019 13:37:45.017 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server number: 8.5.41.0
18-May-2019 13:37:45.017 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Name: Linux
18-May-2019 13:37:45.018 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Version: 4.18.0-20-generic
18-May-2019 13:37:45.018 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Architecture: amd64
18-May-2019 13:37:45.018 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Java Home: /usr/lib/jvm/java-8-openjdk-amd64/jre
18-May-2019 13:37:45.019 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Version: 1.8.0_212-8u212-b01-1~deb9u1-b01
18-May-2019 13:37:45.020 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Vendor: Oracle Corporation
18-May-2019 13:37:45.020 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_BASE: /usr/local/tomcat
18-May-2019 13:37:45.020 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_HOME: /usr/local/tomcat
18-May-2019 13:37:45.021 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.config.file=/usr/local/tomcat/conf/logging.properties
18-May-2019 13:37:46.409 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
18-May-2019 13:37:46.428 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["ajp-nio-8009"]
18-May-2019 13:37:46.436 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 1165 ms
```

# 도커 컨테이너 기본 활용

## ▶ 포트포워딩 옵션 추가하여 이미지를 실행

- 옵션에 추가로 포트포워딩하도록 하여야 접속 가능
- `sudo docker run -t -p 8000:8080 tomcat` (왼쪽에 호스트 포트, 오른쪽이 게스트 포트)
- 톰캣 테스트 페이지에 접속된 화면



# 도커 컨테이너 기본 활용

## ▶▶ 실행 중인 컨테이너 확인

- `sudo docker container ls`

```
server1@server1-VirtualBox:~$ sudo docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
a9fa30758ef0	tomcat	"catalina.sh run"	18 minutes ago	Up 18 minutes	0.0.0.0:8000->8080/tcp	sharp_merkle

## ▶▶ 원하는 컨테이너 중지

- `sudo docker stop a9fa30758ef0`

## ▶▶ 원하는 컨테이너 실행

- `sudo docker start a9fa30758ef0`

## ▶▶ 모든 컨테이너 확인(실행/비실행)

- `sudo docker container ls -a`

```
server1@server1-VirtualBox:~$ sudo docker container ls -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
c04629886aae	tomcat	"catalina.sh run"	About a minute ago	Exited (143) 6 seconds ago		gallant_hopper
496b1e240311	tomcat	"-t -p 8000:8080"	2 minutes ago	Created	8080/tcp	friendly_chatterjee
a9fa30758ef0	tomcat	"catalina.sh run"	25 minutes ago	Exited (143) 5 minutes ago		sharp_merkle
84dfe77d6124	tomcat	"catalina.sh run"	28 minutes ago	Exited (130) 28 minutes ago		blissful_shaw
3bc0343a7e87	tomcat	"catalina.sh run"	31 minutes ago	Exited (130) 29 minutes ago		lucid_williamson

# 도커 컨테이너 기본 활용

## 모든 컨테이너 삭제(실행/비실행)

- `sudo docker container ls -a -q` (컨테이너 아이디만 조회)

```
server1@server1-VirtualBox:~$ sudo docker container ls -a -q
c04629886aae
496b1e240311
a9fa30758ef0
84dfe77d6124
3bc0343a7e87
```

- `sudo docker rm `sudo docker container ls -a -q`` (모두 삭제)

```
server1@server1-VirtualBox:~$ sudo docker rm `sudo docker container ls -a -q`
c04629886aae
496b1e240311
a9fa30758ef0
84dfe77d6124
3bc0343a7e87
```

# 도커 컨테이너 기본 활용

## 도커 가상환경으로 들어가기

- `sudo docker run -t -p 8000:8080 tomcat` # 톰캣 이미지 다시 실행
- `sudo docker ps` # 실행 중인 컨테이너 ID 확인
- `sudo docker exec -it e2f1bc0f8a88 /bin/bash` # 획득한 ID를 사용하여 배시셸로 접근

```
server1@server1-VirtualBox:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
e2f1bc0f8a88        tomcat             "catalina.sh run"   48 seconds ago      Up 46 seconds      0.0.0.0:8000->8080/tcp   happy_poitras
server1@server1-VirtualBox:~$ sudo docker exec -it e2f1bc0f8a88 /bin/bash
root@e2f1bc0f8a88:/usr/local/tomcat# uname -a
Linux e2f1bc0f8a88 4.18.0-20-generic #21~18.04.1-Ubuntu SMP Wed May 8 08:43:37 UTC 2019 x86_64 GNU/Linux
root@e2f1bc0f8a88:/usr/local/tomcat#
```

# 도커 컨테이너 기본 활용

## ▶ 컨테이너의 스토리지의 변경사항 저장

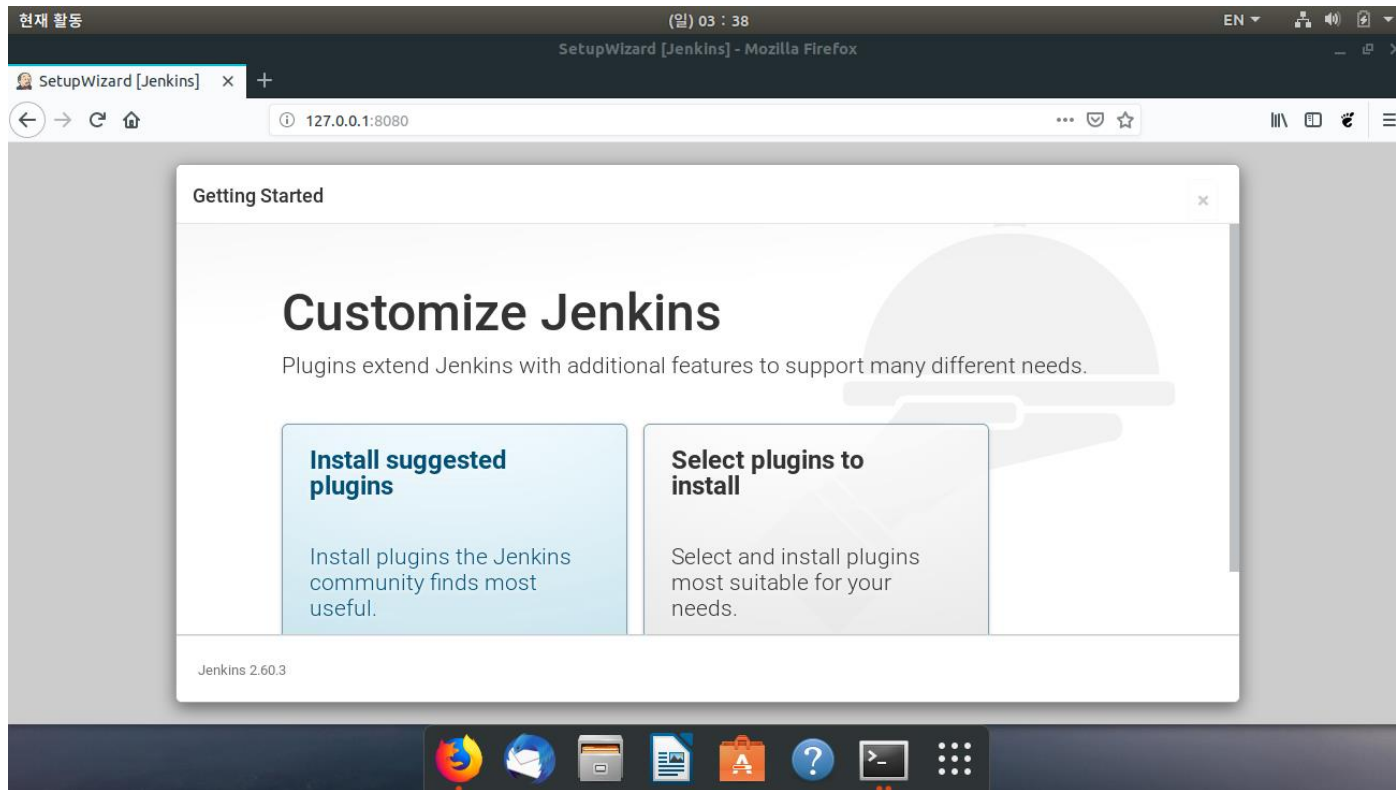
- 도커 가상환경 컨테이너의 스토리지의 변경사항은 컨테이너를 삭제하기 전에는 유지됨
- 재실행 후에도 확인되는 test.txt 파일

```
server1@server1-VirtualBox:~$ sudo docker exec -it e2f1bc0f8a88 /bin/bash
root@e2f1bc0f8a88:/usr/local/tomcat# echo test1234 > test.txt
root@e2f1bc0f8a88:/usr/local/tomcat# exit
exit
server1@server1-VirtualBox:~$ sudo docker stop e2f1bc0f8a88
^[[Ae2f1bc0f8a88
server1@server1-VirtualBox:~$ sudo docker start e2f1bc0f8a88
e2f1bc0f8a88
server1@server1-VirtualBox:~$ sudo docker exec -it e2f1bc0f8a88 /bin/bash
root@e2f1bc0f8a88:/usr/local/tomcat# cat test.txt
test1234
root@e2f1bc0f8a88:/usr/local/tomcat#
```

# 도커 컨테이너 기본 활용

## 연습문제

- 기존에 설치된 모든 컨테이너와 이미지 정지 및 삭제
- 도커 기능을 사용해 Jenkins 검색
- Jenkins를 사용하여 설치
- Jenkins 포트에 접속하여 웹 서비스 열기

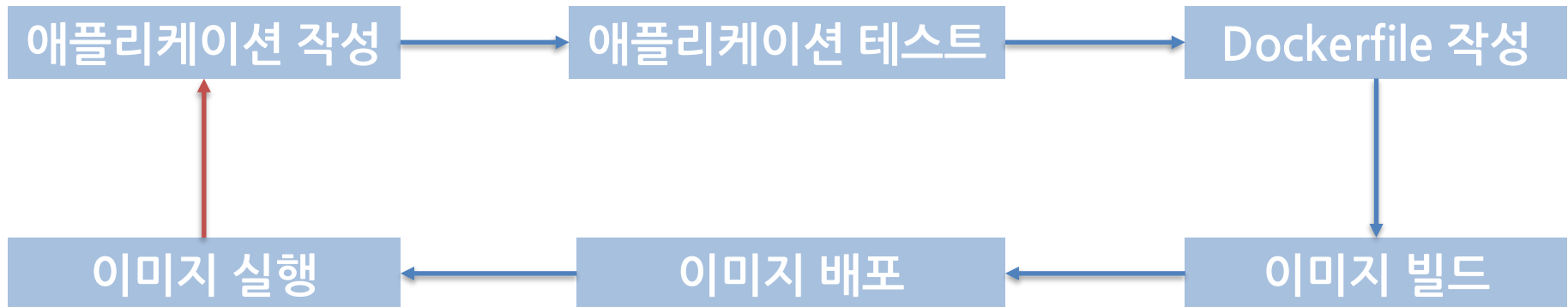




# 도커 컨테이너 기본 활용

## ▶ 이미지 생성과 푸시

- 누군가 만들어 놓은 이미지를 활용하는 것도 매우 좋은 일!
- 그러나 경우에 따라 우리가 필요한 이미지가 없는 경우도 존재
- 우리가 필요한 도커를 만들고 레지스트리에 등록하는 방법에 대해 학습



# 도커 컨테이너 기본 활용

## ▶ 파이썬 에코 서버 스크립트 작성

- 간단한 python 스크립트를 작성하고 잘 실행되는지 테스트

### Server

```
server1@server1-VirtualBox:~$ gedit test_server.py
server1@server1-VirtualBox:~$ python3 test_server.py
Connect by ('127.0.0.1', 36134)
```

```
#!/ test_server.py
import socket
```

```
with socket.socket() as s:
    s.bind(("0.0.0.0", 12345))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print("Connect by", addr)
        while True:
            data = conn.recv(1024)
            if not data: break
            conn.sendall(data)
```

### Client

```
server1@server1-VirtualBox:~$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license"
ormation.
>>> import socket
>>> s = socket.socket()
>>> s.connect(("127.0.0.1",12345))
>>> s.send("I am your father....".encode())
20
>>> s.recv(1024)
b'I am your father....'
>>> □
```

# 도커 컨테이너 기본 활용

## ▶ 이미지 빌드를 위한 Dockerfile 생성

- 도커는 Dockerfile에서 지침을 읽어 자동으로 이미지 빌드
- 이미지를 어셈블하기 위해 사용자가 명령 행에서 호출할 수 있는 모든 명령을 포함하는 텍스트 파일
- 이미지를 작성하기 원하는 디렉토리를 별도로 생성하고 그 안에서 작업
- **주의: 루트 디렉토리를 빌드에 사용하는 경우에는 전체 파일 시스템을 도커 이미지로 만들려고 시도**
- `$ mkdir my_first_project`
- `$ cd my_first_project`
- `gedit Dockerfile`

```
server1@server1-VirtualBox:~$ mkdir my_first_project
server1@server1-VirtualBox:~$ cd my_first_project/
server1@server1-VirtualBox:~/my_first_project$ gedit Dockerfile &
[1] 13190
```



```
FROM python:3.7

RUN mkdir /echo
COPY test_server.py /echo

CMD ["python", "/echo/test_server.py"]
```

# 도커 컨테이너 기본 활용


## Dockerfile에 정의할 수 있는 인스트럭션

인스트럭션	설명	정의 방법
FROM	우분투와 같은 운영체제 이름이나 필요한 애플리케이션 이름을 지정	FROM ubuntu:14.04
RUN	컨테이너가 빌드될 때 실행할 명령어	RUN ["mv", "1234", "5678"]
EXPOSE	컨테이너 외부에 노출할 포트 지정	EXPOSE 80
ENV	환경 변수 이름 지정	ENV abc /tmp echo \${abc}
CMD	컨테이너가 시작될 때 실행할 명령어	CMD ["echo", "1234"]
ENTRYPOINT	컨테이너가 실행될 때 기본 명령어 지정 (CMD와 달리 옵션을 넘길 수 있음)	ENTRYPOINT ["python"] ... \$ sudo docker run -t 이미지 -c "print('1234')"
WORKDIR	작업할 디렉토리를 세팅	WORKDIR /project
USER	도커 실행 시 사용할 유저 이름 또는 uid를 지정	USER user 또는 USER user:group
VOLUME	호스트의 디렉토리를 도커에 연결 커밋 없이 사용 가능하며 주로 로그 수집이나 데이터 저장용으로 사용	VOLUME ["/tmp"]
COPY, ADD	파일이나 디렉토리를 복사할 때 사용하는 명령어 명령어를 대신 사용하면 URL도 가능하며 압축된 파일은 압축을 해제하면서 옮김	COPY file /copy/to/path ADD http://example.com/big.tar.xz /usr/src/things/ ADD rootfs.tar.xz /
ARG		
SHELL	원하는 타입의 셸을 사용	SHELL ["/bin/sh", "-c"] SHELL ["cmd", "/S", "/C"]

# 도커 컨테이너 기본 활용

## 📁 Dockerfile 레퍼런스

- Dockerfile에 대한 기타 자세한 내용은 <https://docs.docker.com/engine/reference/builder/>를 참고

 docker docs

[Guides](#)

[Product manuals](#)

[Glossary](#)

[Reference](#)

[Samples](#)

File formats

Dockerfile reference

Compose file reference

Command-Line Interfaces (CLIs)

Application Programming Interfaces (APIs)

Drivers and specifications

Compliance control references

Estimated reading time: 74 minutes

## Dockerfile reference

Docker can build images automatically by reading the instructions from a `Dockerfile`. A `Dockerfile` is a text document that contains all the commands a user could call on the command line to assemble an image. Using `docker build` users can create an automated build that executes several command-line instructions in succession.

This page describes the commands you can use in a `Dockerfile`. When you are done reading this page, refer to the `Dockerfile` [Best Practices](#) for a tip-oriented guide.

## Usage

The `docker build` command builds an image from a `Dockerfile` and a `context`. The build's context is the set of files at a specified location `PATH` or `URL`. The `PATH` is a directory on your local filesystem. The `URL` is a Git repository location.

A context is processed recursively. So, a `PATH` includes any subdirectories and the `URL` includes the repository and its submodules. This example shows a build command that uses the current directory as context:

```
$ docker build .
Sending build context to Docker daemon 6.51 MB
...
```

# 도커 컨테이너 기본 활용

## ▶ 이미지 빌드하기

- `sudo docker build -t gasbugs/echo_test:latest .`
- “현재 디렉토리를 `gasbugs/echo_test`로 만들어라”는 뜻, `latest`는 최신 버전이라는 뜻(번호를 사용 가능)
- 레지스트리의 `gasbugs` 네임스페이스에 `echo_test`라는 이미지로 생성 (이미지 명의 충돌 회피)
- 이미지 명 없이(`-t` 옵션 삭제) 이미지를 만들 수 있으나 해시값을 사용하기에는 불편
- `tomcat`과 같이 네임스페이스가 없는 경우는 일반적으로 도커에서 지원하는 이미지

```
server1@server1-VirtualBox:~/my_first_project$ sudo docker build -t gasbugs/echo_test:la
Sending build context to Docker daemon 3.072kB
Step 1/4 : FROM python:3.7
3.7: Pulling from library/python
c5e155d5a1d1: Already exists
Status: Downloaded newer image for python:3.7
----> a4cc999cf2aa
Step 2/4 : RUN mkdir /echo
----> Running in 2755712faf1d
Removing intermediate container 2755712faf1d
----> 6cf79f7e114e
Step 3/4 : COPY test_server.py /echo
----> 3311b3a008ae
Step 4/4 : CMD ['python3', '/echo/test_server.py']
----> Running in 32657d203885
Removing intermediate container 32657d203885
----> 3240c5532acc
Successfully built 3240c5532acc
Successfully tagged gasbugs/echo_test:latest
```

# 도커 컨테이너 기본 활용

## ▶ 도커 이미지가 제대로 만들어졌는지 실행 확인

- `sudo docker run -t -p 12345:12345 gasbugs/echo_test:latest`

```
server1@server1-VirtualBox:~/my_first_project$ sudo docker run -t -p 12345:12345 gasbugs/echo_test:la
```

```
Connect by ('172.17.0.1', 49892)
```

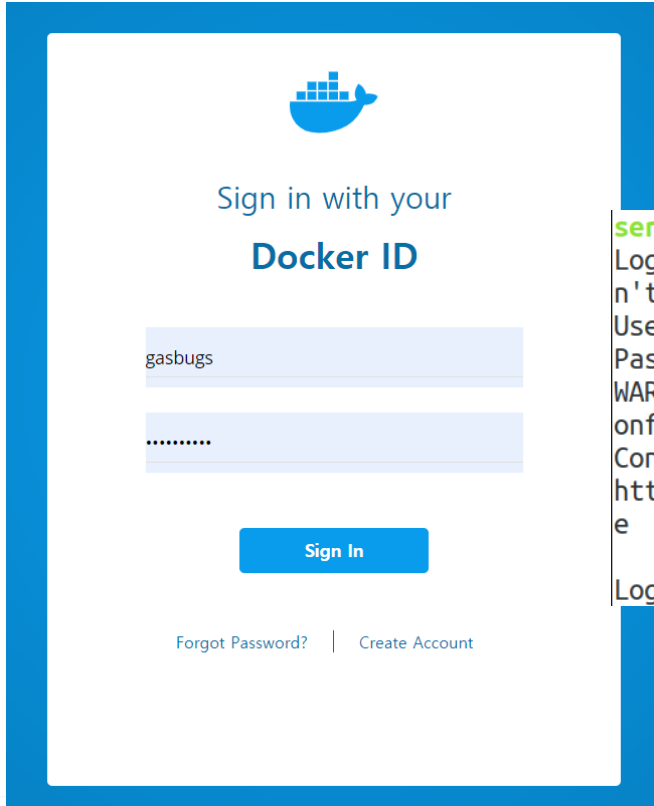
```
>>> s = socket.socket()
>>> s.connect(("127.0.0.1",12345))
>>> s.send("I am your father....".encode())
20
>>> s.recv(1024)
b'I am your father....'
>>>
```



# 도커 컨테이너 기본 활용

## 도커 회원 가입하여 내 레지스트리 만들기

- 도커 레지스트리에 푸시를 하기 위해 회원가입이 필요
- <https://www.docker.com/>
- 회원가입 후 `sudo docker login`을 사용하여 로그인



로그인시 timeout 에러가 출력된다면 다음 명령으로 프로토콜 관련 파일을 설치  
`sudo apt install gnupg2 pass`

```
server1@server1-VirtualBox:~/test_server$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you do
n't have a Docker ID, head over to https://hub.docker.com to create one.
Username: gasbugs
Password:
WARNING! Your password will be stored unencrypted in /home/server1/.docker/c
onfig.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-stor
e
Login Succeeded
```

# 도커 컨테이너 기본 활용

## ▶ 내가 만든 도커 이미지를 푸시하기

- `sudo docker image push gasbugs/echo_test`

```
server1@server1-VirtualBox:~/my_first_project$ sudo docker image push gasbugs/echo_test
The push refers to repository [docker.io/gasbugs/echo_test]
9b19f98029e2: Pushed
d42866664a6e: Pushed
2633623f6cf4: Mounted from library/python
5194c23c2bc2: Mounted from library/python
69bbfe9f27d4: Mounted from library/python
2492a3be066b: Mounted from library/python
910d7fd9e23e: Mounted from library/python
4230ff7f2288: Mounted from library/python
2c719774c1e1: Mounted from library/python
ec62f19bb3aa: Mounted from library/python
f94641f1fe1f: Mounted from library/python
latest: digest: sha256:4b210c5c737f3e8c775b450bfba2d8dc984c2034691fe2869ae160daeef5c587 size: 2632
```

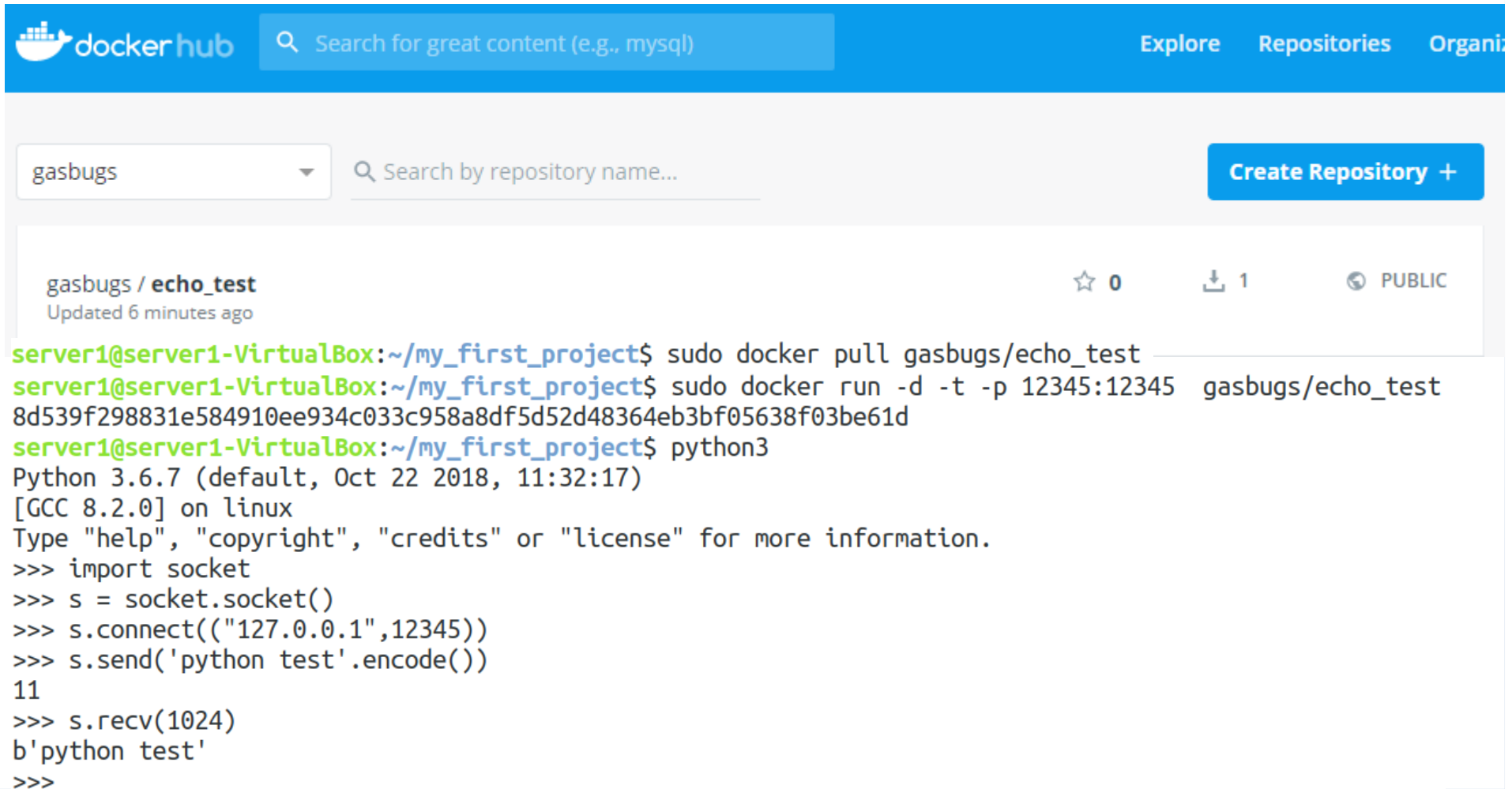
## ▶ 만약 내가 만든 도커 이미지가 업로드 되지 않는다면!?

- 이미지의 네임스페이스가 자신의 아이디와 일치해야 한다.
- 일치하지 않는 경우는 간단히 tag 명을 수정하면 된다.
- 태그 명을 수정하는 명령어:
  - `sudo docker tag foo/sample_image:latest username/sample_image:latest`

# 도커 컨테이너 기본 활용

## 푸시한 도커 이미지 확인하기

- <https://hub.docker.com/> 으로 접속하여 로그인 후 내 리파지토리를 확인



The screenshot shows the Docker Hub web interface. At the top is a blue navigation bar with the Docker Hub logo, a search bar, and links for 'Explore', 'Repositories', and 'Organizations'. Below the navigation bar is a white header area with a dropdown menu showing 'gasbugs' and a search bar labeled 'Search by repository name...'. To the right of the search bar is a blue button labeled 'Create Repository +'. Below the header is a card for the repository 'gasbugs / echo\_test', which was updated 6 minutes ago. The card shows 0 stars, 1 download, and is public. Below the repository card is a terminal window showing the following commands and output:

```
server1@server1-VirtualBox:~/my_first_project$ sudo docker pull gasbugs/echo_test
server1@server1-VirtualBox:~/my_first_project$ sudo docker run -d -t -p 12345:12345 gasbugs/echo_test
8d539f298831e584910ee934c033c958a8df5d52d48364eb3bf05638f03be61d
server1@server1-VirtualBox:~/my_first_project$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import socket
>>> s = socket.socket()
>>> s.connect(("127.0.0.1",12345))
>>> s.send('python test'.encode())
11
>>> s.recv(1024)
b'python test'
>>>
```

# 도커 컨테이너 기본 활용

## ▶ Private 레지스트리에 내 이미지 업로드하기

- \$ sudo docker image tag gasbugs/echo\_test:latest localhost:5000/gasbugs/echo\_test:latest
- \$ sudo docker image push localhost:5000/gasbugs/echo\_test:latest
- \$ sudo docker image pull localhost:5000/gasbugs/echo\_test:latest

```
server1@server1-VirtualBox:~/my_first_project$ sudo docker image tag gasbugs/echo_test:latest localhost:5000/ga
server1@server1-VirtualBox:~/my_first_project$ sudo docker image push localhost:5000/gasbugs/echo_test:latest
The push refers to repository [localhost:5000/gasbugs/echo_test]
2277eeefb8ec: Pushed
c35d34fb9c1d: Pushed
799a7872c8c7: Pushed
715450468940: Pushed
c9d608035aef: Pushed
bb9c02680a15: Pushed
a637c551a0da: Pushed
2c8d31157b81: Pushed
7b76d801397d: Pushed
f32868cde90b: Pushed
0db06dff9d9a: Pushed
latest: digest: sha256:2a54c6118f8eed967864c0c85ad95bf75e77ba27dba87f67f0f9ffce6a5e1300 size: 2632
server1@server1-VirtualBox:~/my_first_project$ sudo docker image pull localhost:5000/gasbugs/echo_test:latest
latest: Pulling from gasbugs/echo_test
Digest: sha256:2a54c6118f8eed967864c0c85ad95bf75e77ba27dba87f67f0f9ffce6a5e1300
Status: Image is up to date for localhost:5000/gasbugs/echo_test:latest
```

# 도커 컨테이너 기본 활용

## 다양한 도커 명령어

- 너무 많아서 다 짚고 넘어가기가 어려움
- `sudo docker --help`
- <https://docs.docker.com/engine/reference/run/>

```
server1@server1-VirtualBox:~$ sudo docker docker --help
```

```
Usage: docker [OPTIONS] COMMAND
```


A self-sufficient runtime for containers

### Options:

--config string	Location of client config files (default "/etc/docker")
-D, --debug	Enable debug mode
-H, --host list	Daemon socket(s) to connect to
-l, --log-level string	Set the logging level ("debug" "info" "warn" "error")
--tls	Use TLS; implied by --tlsverify
--tlscacert string	Trust certs signed only by this CA (default "/etc/docker/ca.pem")
--tlscert string	Path to TLS certificate file (default "/home/server1/.docker/tls.crt")
--tlskey string	Path to TLS key file (default "/home/server1/.docker/tls.key")
--tlsverify	Use TLS and verify the remote
-v, --version	Print version information and quit

### Management Commands:

builder	Manage builds
config	Manage Docker configs
container	Manage containers
engine	Manage the docker engine
image	Manage images
network	Manage networks
node	Manage Swarm nodes
plugin	Manage plugins
secret	Manage Docker secrets
service	Manage services
stack	Manage Docker stacks
swarm	Manage Swarm

 docker docs

Search the docs

GuidesProduct manualsGlossaryReferenceSamples

File formats

Command-Line Interfaces (CLIs)

Docker CLI (docker)

Docker run reference

Use the Docker command line

docker (base command)

docker attach

docker build

docker builder \*

docker checkpoint \*

docker commit

docker config \*

Estimated reading time: 61 minutes

Docker run reference

Docker runs processes in isolated containers. A container is a process which runs on a host. The host may be local or remote. When an operator executes `docker run`, the container process that runs is isolated in that it has its own file system, its own networking, and its own isolated process tree separate from the host.

This page details how to use the `docker run` command to define the container's resources at runtime.

General form

The basic `docker run` command takes this form:

```
$ docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]
```

# 도커 컨테이너 기본 활용

## 자주 사용하는 도커 명령어

도커 명령어	설명
<code>sudo docker container run &lt;이미지 명&gt;</code>	컨테이너 생성 및 실행 -i: 컨테이너의 표준 입력 연결 -t: 터미널 기능 활성화 --rm: 컨테이너 종료시 파기 -v: 디렉토리나 파일 공유
<code>sudo docker container ls &lt;컨테이너 명&gt;</code>	도커 컨테이너 목록 보기 -q: 컨테이너 ID만 보기 --filter "필터명=값": 필터 적용(다음장에서 확인) -a: 모든 컨테이너 보기
<code>sudo docker container stop &lt;컨테이너 명&gt;</code>	컨테이너 정지
<code>sudo docker container restart &lt;컨테이너 명&gt;</code>	컨테이너 재시작
<code>sudo docker container rm &lt;컨테이너 명&gt;</code>	컨테이너 삭제
<code>sudo docker container logs &lt;컨테이너 명&gt;</code>	표준 출력 연결하기
<code>sudo docker container exec &lt;컨테이너 명&gt; &lt;컨테이너에서 실행할 명령&gt;</code>	실행중인 컨테이너에 명령 실행하기 예: <code>sudo docker exec -it e2f1bc0f8a88 /bin/bash</code> # 획득한 ID를 사용하여 배시셀로 접근
<code>sudo docker container cp &lt;source&gt; &lt;dest&gt;</code>	파일 복사하기 예: <code>sudo docker container cp echo:/tmp/a.txt .</code>

# 도커 컨테이너 기본 활용

## 필터의 종류 설명

- 필터는 도커 명령어 ls나 ps 등에서 사용할 수 있다.
- <https://docs.docker.com/engine/reference/commandline/ps/>

Filter	Description
id	Container's ID
name	Container's name
label	An arbitrary string representing either a key or a key-value pair. Expressed as <code>&lt;key&gt;</code> or <code>&lt;key&gt;=&lt;value&gt;</code>
exited	An integer representing the container's exit code. Only useful with <code>--all</code> .
status	One of <code>created</code> , <code>restarting</code> , <code>running</code> , <code>removing</code> , <code>paused</code> , <code>exited</code> , or <code>dead</code>
ancestor	Filters containers which share a given image as an ancestor. Expressed as <code>&lt;image-name&gt;[:&lt;tag&gt;]</code> , <code>&lt;image id&gt;</code> , or <code>&lt;image@digest&gt;</code>
before or since	Filters containers created before or after a given container ID or name

volume	Filters running containers which have mounted a given volume or bind mount.
network	Filters running containers connected to a given network.
publish or expose	Filters containers which publish or expose a given port. Expressed as <code>&lt;port&gt;[/&lt;proto&gt;]</code> or <code>&lt;startport-endport&gt;[/&lt;proto&gt;]</code>
health	Filters containers based on their healthcheck status. One of <code>starting</code> , <code>healthy</code> , <code>unhealthy</code> or <code>none</code> .
isolation	Windows daemon only. One of <code>default</code> , <code>process</code> , or <code>hyperv</code> .
is-task	Filters containers that are a "task" for a service. Boolean option ( <code>true</code> or <code>false</code> )



# 도커 컨테이너 기본 활용

## ▶ 운영 환경 명령어

도커 명령어	설명
sudo docker container prune	중지 된 모든 컨테이너를 제거
sudo docker image prune	태그가 없는 모든 이미지 파기
sudo docker system prune	사용하지 않는 모든 데이터 삭제(이미지, 컨테이너, 볼륨, 네트워크 등)
sudo docker container stats	컨테이너 사용 현황 출력

```
현재 활동  터미널 (일) 04 : 31
server1@server1-VirtualBox: ~
CONTAINER ID        NAME                      CPU %       MEM USAGE / LIMIT   MEM %       NET I/O       BLOCK I/O    PIDS
6b4b5035f95d       condescending_dubinsky   83.56%     364.8MiB / 3.852GiB  9.25%      240kB / 4.8kB   0B / 6.2MB   34
12c1f198801b       festive_diffie           0.00%     5.75MiB / 3.852GiB  0.15%     3.05kB / 0B     0B / 0B      1
```

sudo docker container stats 실행