

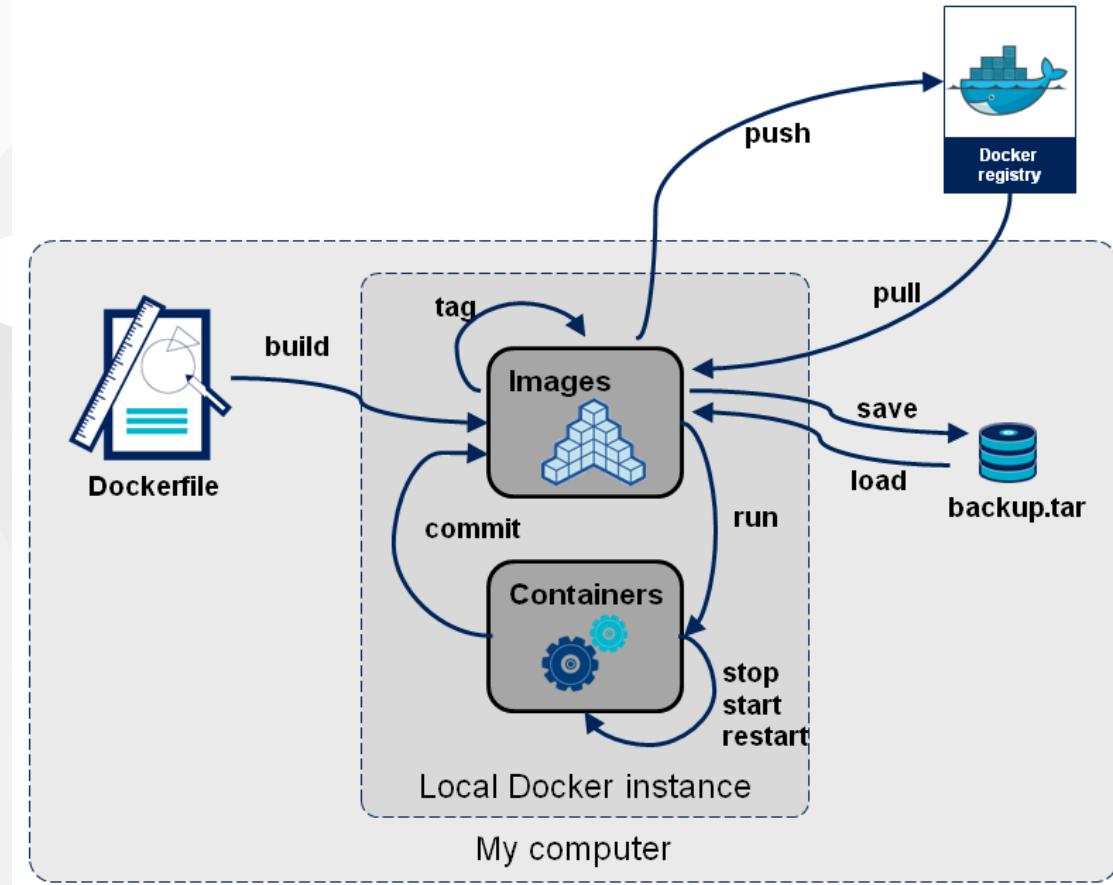
# Contents

- Docker commands - 이미지 다루기
- Docker commands - 컨테이너 실행하기
- Docker commands - 오브젝트 확인하기
- Docker commands(정리)



### Docker commands

도커 명령어를 이용하여 이미지와 컨테이너를 다루는 방법을 알아보겠습니다.



### Docker commands - 이미지 다루기

컨테이너를 생성하기 위해서는 컨테이너 이미지가 필요합니다.  
이미지를 준비하는 방법은 여러가지가 있지만, 대표적으로

- 빌드(Build)해서 만들기
- Registry에서 다운로드(Pull)하기

와 같은 방법이 있습니다.

`docker pull` 명령어를 실행하면, Registry에서 이미지를 다운로드(Pull) 할 수 있습니다.  
명령어 형식은 다음과 같습니다.

```
docker pull [OPTIONS] NAME[:TAG|@DIGEST]
```

Pull할 이미지(IMAGE[:TAG|@DIGEST])를 지정하여 실행합니다.  
이 때, TAG를 지정하지 않으면(생략하면) `:latest` 를 기본으로 사용합니다.

그리고, Registry는 [Docker hub\(default\)](#) 나 다른 Registry들이 사용될 수 있습니다.

- Docker Hub의 이미지 (예시) - `nginx:latest`, `nginx:1.18`, `ubuntu:20.04`
- 다른 Registry의 이미지 (예시) - `registry.k8s.io/busybox`, `registry.k8s.io/liveness`

#### docker Pull

## Docker & Kubernetes - 02. Docker commands

### Docker commands - 이미지 다루기

`docker pull` 명령어의 실행 예시는 다음과 같습니다.

```
ubuntu@ip-10-0-1-14:~$ docker pull nginx:1.18
1.18: Pulling from library/nginx
f7ec5a41d630: Pull complete
0b20d28b5eb3: Pull complete
1576642c9776: Pull complete
c12a848bad84: Pull complete
03f221d9cf00: Pull complete
Digest: sha256:e90ac5331fe095cea01b121a3627174b2e33e06e83720e9a934c7b8ccc9c55a0
Status: Downloaded newer image for nginx:1.18
docker.io/library/nginx:1.18
```

이미지는 레이어들로 이루어져 있고, 각 레이어들이 모두 pull 됩니다.

`docker push` 명령어의 실행 예시는 다음과 같습니다.

```
ubuntu@ip-10-0-1-14:~$ docker push rogallo/todo-app:1.0.0
The push refers to repository [docker.io/rogallo/todo-app]
a07f0156c43d: Pushed
8c40db749504: Pushed
4efca0eb4778: Pushed
edff9ff691d5: Layer already exists
cbe4b9146f86: Layer already exists
a6524c5b12a6: Layer already exists
9a5d14f9f550: Layer already exists
1.0.0: digest: sha256:5cee6f196aa06a6ba00a1b7c40a0b674510cf9f931785d9491daaa31af0d9de1 size: 1787
```

### Docker commands - 컨테이너 실행하기

`docker run` 명령어를 실행하면, 이미지를 이용해서 만들어진 컨테이너에서 프로세스가 실행되게 됩니다.  
이 때 이 컨테이너 프로세스는 자신의 파일시스템, 네트워킹, 프로세스 트리를 가지게 됩니다. (격리된 환경)

컨테이너를 실행하는 명령인 `docker run` 명령은 다음과 같은 형식을 가지고 있습니다.

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

기본적으로 실행 할 이미지(IMAGE[:TAG|@DIGEST])를 지정해야 하고 다음과 같은 옵션들을 추가할 수 있습니다.

- detached(-d) or foreground(-it) running
- container identification(--name)
- network settings(--network)
- runtime constraints on CPU and memory

 [Docker run reference](#)

## Docker & Kubernetes - 02. Docker commands

### Docker commands - 컨테이너 실행하기

실행 시 추가할 수 있는 주요 옵션 들은 다음과 같습니다.

Options:	
-a, --attach list	Attach to STDIN, STDOUT or STDERR
-d, --detach	Run container in background and print container ID
--entrypoint string	Overwrite the default ENTRYPOINT of the image
-e, --env list	Set environment variables
-i, --interactive	Keep STDIN open even if not attached
-p, --publish list	Publish a container's port(s) to the host
--rm	Automatically remove the container when it exits
-t, --tty	Allocate a pseudo-TTY
-v, --volume list	Bind mount a volume

컨테이너 실행은 다음 두 가지 유형이 있습니다.

**Detached (-d)**

**Foreground (-it)**

-백그라운드에서 실행되는 모드    -프로세스의 standard I/O, standard error에 console로 연결

이 두 가지의 차이를 알아보겠습니다.

 [Detached vs foreground](#)

### Docker commands - 컨테이너 실행하기 (Detached)

아래 명령어는 Nginx Container를 Detached 모드로 실행하는 명령어입니다.

```
ubuntu@ip-10-0-1-14:~$ docker run -d --name my-nginx -p 8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
b85a868b505f: Pull complete
f4407ba1f103: Pull complete
4a7307612456: Pull complete
935cecace2a0: Pull complete
8f46223e4234: Pull complete
fe0ef4c895f5: Pull complete
Digest: sha256:10f14ffa93f8dedf1057897b745e5ac72ac5655c299dade0aa434c71557697ea
Status: Downloaded newer image for nginx:latest
bbc0d5c6b94abd21fc831bedd9d28d4f4f08fd25aab474953e6e9f9a56c34434
ubuntu@ip-10-0-1-14:~$
```

마지막 프롬프트가 Host머신의 프롬프트(`ubuntu@ip-10-0-1-14:~$`)임.

위 명령어를 실행할때, Docker에서 아래와 같은 일들이 순차적으로 발생합니다.

1. Nginx이미지가 로컬(Host머신)에 없다면, Docker Registry로부터 이미지를 다운로드(pull) 합니다.
2. 다운로드 받은 이미지로 신규 Container를 생성합니다.
3. 생성된 Container에 read-write filesystem을 마지막 Layer로 할당합니다.
4. Default Network Interface인 브릿지 네트워크를 생성합니다.
5. Container를 Detached 모드로 시작하고 컨테이너의 80번 포트가 Host머신의 8080포트를 통해 노출됩니다.

### Docker commands - 컨테이너 실행하기 (Detached)

다음 명령으로 실행된 Nginx 컨테이너의 응답을 확인해볼 수 있습니다.

```
ubuntu@ip-10-0-1-14:~$ curl http://localhost:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

명령어 : `curl http://localhost:8080`

### Docker commands - 컨테이너 실행하기 (Foreground)

아래 명령어는 Ubuntu Container를 Foreground 모드로 실행하는 명령어입니다.

```
ubuntu@ip-10-0-1-14:~$ docker run -it ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
405f018f9d1d: Pull complete
Digest: sha256:b6b83d3c331794420340093eb706a6f152d9c1fa51b262d9bf34594887c2c7ac
Status: Downloaded newer image for ubuntu:latest
root@4276ef1e8f67:/#
```

마지막 라인의 프롬프트가 컨테이너(Ubuntu)의 프롬프트( `root@4276ef1e8f67:/#` )임.

위 명령어를 실행할때, Docker에서 아래와 같은 일들이 순차적으로 발생합니다.

1. Ubuntu 이미지가 로컬(Host머신)에 없다면, Docker Registry로부터 이미지를 다운로드(pull) 합니다.
2. 다운로드 받은 이미지로 신규 Container를 생성합니다.
3. 생성된 Container에 read-write filesystem을 마지막 Layer로 할당합니다.
4. Default Network Interface인 브릿지 네트워크를 생성합니다.
5. Container를 시작하고 /bin/bash를 실행합니다.

exit를 입력하여 /bin/bash를 종료하면 컨테이너는 중지(stop)되지만 삭제되지 않고, 추후에 삭제하거나 다시 재시작 할 수 있습니다.

## Docker & Kubernetes - 02. Docker commands

### Docker commands - 오브젝트 확인하기

현재 Host 머신에 존재하는 오브젝트를 확인하는 명령어들은 다음과 같은 것들이 있습니다.

- 이미지 목록 확인 : `docker images`
- 컨테이너 목록 : `docker ps`
- 오브젝트의 상세내용 확인 : `docker inspect`

각각의 명령어 사용 예는 다음과 같습니다.

먼저 이미지의 목록을 확인하려면 다음과 같이 조회하면 됩니다.

```
ubuntu@ip-10-0-1-14:~$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
nginx           latest   448a08f1d2f9  12 days ago  142MB
ubuntu          latest   3b418d7b466a  2 weeks ago  77.8MB
nginx           1.18    c2c45d506085  2 years ago  133MB
```

그리고, 컨테이너의 목록은 다음과 같이 조회할 수 있습니다.

```
ubuntu@ip-10-0-1-14:~$ docker ps --all
CONTAINER ID      IMAGE      COMMAND                  CREATED      STATUS      PORTS      NAMES
c27a65cf592d      ubuntu      "/bin/bash"              11 seconds ago   Exited (0) 8 seconds ago
d804da47bffd      nginx      "/docker-entrypoint...."  22 seconds ago   Up 21 seconds   0.0.0.0:8080->80/tcp, :::8080->80/tcp  my-nginx
```

--all 옵션을 사용하여 Exit 상태의 컨테이너까지 모두 조회함.

## Docker & Kubernetes - 02. Docker commands

### Docker commands - 오브젝트 확인하기

생성된 컨테이너의 상세내용을 확인하는 명령어의 실행결과는 다음과 같습니다.

```
ubuntu@ip-10-0-1-14:~$ docker inspect my-nginx
[
  {
    "Id": "d804da47bffd9f26a0f3d47f971d4a245de0e3c8a7eab31821003475ed654",
    "Created": "2023-05-16T04:34:55.45188562Z",
    "Path": "/docker-entrypoint.sh",
    "Args": [
      "nginx",
      "-g",
      "daemon off;"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 16302,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2023-05-16T04:34:55.957204065Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:448a08f1d2f94e8db6db9286fd77a3a4f3712786583720a12f1648abb8cace25",
    ... 생략 ...
  }
]
```

### Docker commands(정리) - 기본 명령어 (이미지 관련)

```
# 자주 사용되는 명령어
$ docker build [OPTIONS] PATH | URL | -
$ docker pull [OPTIONS] NAME[:TAG|@DIGEST]
$ docker push [OPTIONS] NAME[:TAG]
$ docker images [OPTIONS] [REPOSITORY[:TAG]]
$ docker rmi [OPTIONS] IMAGE [IMAGE...]
# 기타 명령어
$ docker image prune [OPTIONS]
$ docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]
$ docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]
$ docker save [OPTIONS] IMAGE [IMAGE...]
$ docker load [OPTIONS]
```

Command	Description
<a href="#">docker build</a>	Build an image from a Dockerfile
<a href="#">docker pull</a>	Pull an image or a repository from a registry
<a href="#">docker push</a>	Push an image or a repository to a registry
<a href="#">docker images</a>	List images
<a href="#">docker rmi</a>	Remove one or more images
<a href="#">docker image prune</a>	Remove unused images
<a href="#">docker tag</a>	Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
<a href="#">docker commit</a>	Create a new image from a container's changes
<a href="#">docker save</a>	Save one or more images to a tar archive (streamed to STDOUT by default)
<a href="#">docker load</a>	Load an image from a tar archive or STDIN

### Docker commands(정리) - 기본 명령어 (컨테이너 관련)

```
# 자주 사용되는 명령어
$ docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
$ docker ps [OPTIONS]
$ docker inspect [OPTIONS] NAME|ID [NAME|ID...]
$ docker rm [OPTIONS] CONTAINER [CONTAINER...]
# 기타 명령어
$ docker attach [OPTIONS] CONTAINER
$ docker exec [OPTIONS] CONTAINER COMMAND [ARG...]
$ docker start [OPTIONS] CONTAINER [CONTAINER...]
$ docker stop [OPTIONS] CONTAINER [CONTAINER...]
```

Command	Description
<a href="#">docker run</a>	Run a command in a new container
<a href="#">docker ps</a>	List containers
<a href="#">docker inspect</a>	Return low-level information on Docker objects
<a href="#">docker rm</a>	Remove one or more containers
<a href="#">docker attach</a>	Attach local standard input, output, and error streams to a running container * detach from a container : CTRL-p CTRL-q key sequence
<a href="#">docker exec</a>	Run a command in a running container
<a href="#">docker start</a>	Start one or more stopped containers
<a href="#">docker stop</a>	Stop one or more running containers

### Docker commands(정리) - 기타 명령어

```
# 자주 사용되는 명령어
$ docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-  
$ docker login [OPTIONS] [SERVER]  
$ docker logout [SERVER]  
$ docker logs [OPTIONS] CONTAINER  
# 기타 명령어
$ docker create [OPTIONS] IMAGE [COMMAND] [ARG...]  
$ docker diff CONTAINER  
$ docker export [OPTIONS] CONTAINER  
$ docker info [OPTIONS]
```

Command	Description
<a href="#">docker cp</a>	Copy files/folders between a container and the local filesystem
<a href="#">docker login</a>	Log in to a Docker registry (e.g. <a href="#">Docker hub</a> )
<a href="#">docker logout</a>	Log out from a Docker registry
<a href="#">docker logs</a>	Fetch the logs of a container
<a href="#">docker create</a>	Create a new container (without starting it)
<a href="#">docker diff</a>	Inspect changes to files or directories on a container's filesystem
<a href="#">docker export</a>	Export a container's filesystem as a tar archive
<a href="#">docker info</a>	Display system-wide information

## 유용한 명령어 사용법 (Tip)

```
# (주의!!!!) 실행중인 모든 컨테이너를 삭제(rm)하고 싶을 때 ( stop and remove )
$ docker rm -f $(docker ps -aq)
```

```
# (주의!!!!) 모든 이미지를 삭제하고 싶을 때
$ docker rmi -f $(docker images -aq)
```

```
# bash shell을 실행해서 컨테이너(e.g. ubuntu)에 연결하고 싶을 때
$ docker exec -it my-ubuntu bash
$ docker exec my-ubuntu bash -c "cat /etc/hosts"
```

```
# 컨테이너의 로그를 보면서 확인해보고 싶을 때 (e.g. 테스트를 할 때)
$ docker logs -f my-nginx
```

```
# 파일을 컨테이너로 복사하고 싶을 때 (또는 반대로)
$ docker cp ./some_file my-ubuntu:/work # some_file을 my-ubuntu 컨테이너의 /work 로 복사
```

**주의** : 첫 번째와 두 번째 명령어(전체 삭제)는 학습환경이나 테스트환경에서만 사용하세요.

### Summary

- 컨테이너 실행하기 (`docker run`)
  - Detached (`docker run -d`)
  - Foreground (`docker run -it`)
- 도커 명령어
  - 이미지 관련 명령어
    - `docker build` , `docker pull` , `docker push` , `docker images` , `docker rmi`
  - 컨테이너 관련 명령어
    - `docker run` , `docker ps` , `docker inspect` , `docker rm`
  - 기타 명령어
    - `docker cp` , `docker login` , `docker logout` , `docker logs`