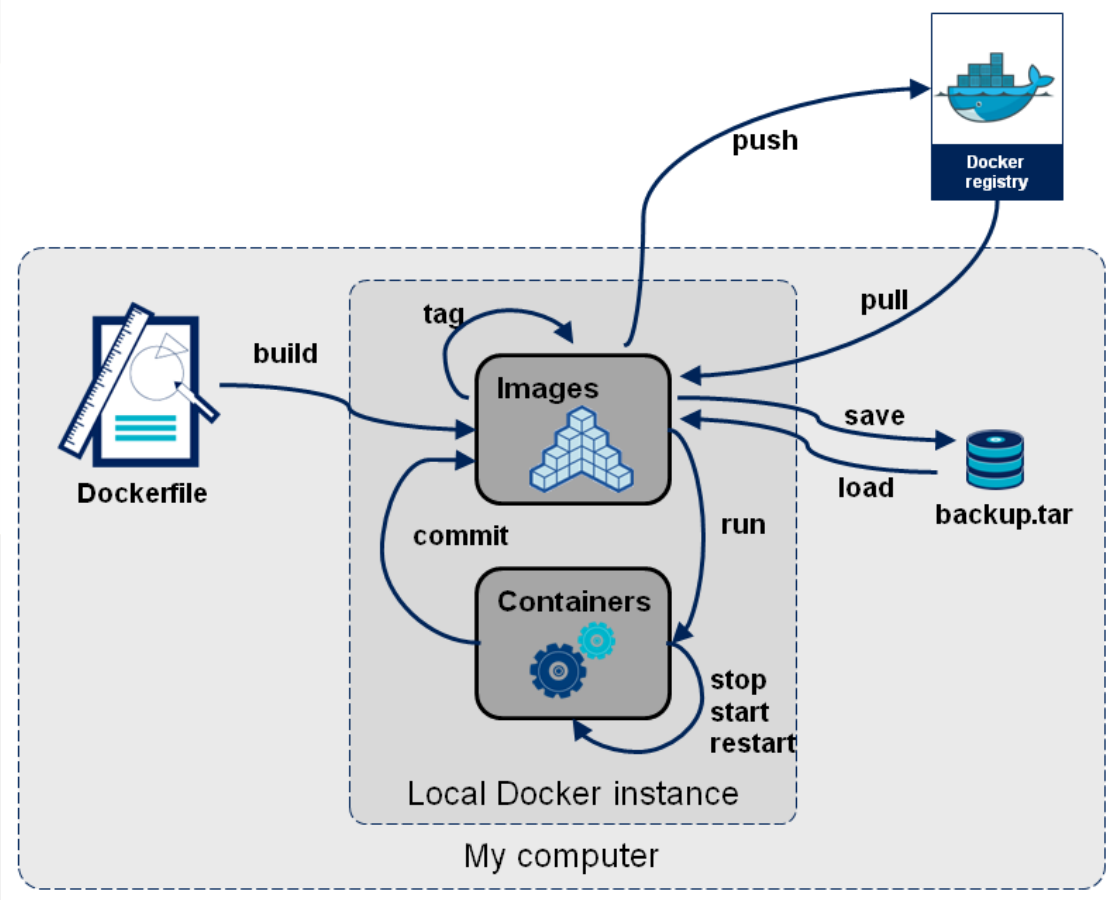
Docker commands

도커 명령어를 이용하여 컨테이너 이미지와 컨테이너를 다루는 방법을 알아보겠습니다.



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 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
```

위의 옵션들 외에도 많은 옵션들이 있습니다.

실행은 다음 두 가지 유형이 있습니다.

Detached (-d)	Foreground (-it) - Default
백그라운드에서 실행되는 모드	프로세스의 standard I/O, standard error에 console 로 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>
```

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. 이를 통해 실행중인 Container는 로컬파일 시스템의 파일과 디렉터리를 생성하거나 수정할 수 있습니다.
4. Default Network Interface인 브릿지 네트워크를 생성합니다.
5. Container를 시작하고 /bin/bash를 실행합니다.
6. exit를 입력하여 /bin/bash를 종료하면 컨테이너는 중지되지만 삭제되지 않고, 추후에 삭제하거나 다시 재시작 할 수 있습니다.

Docker commands - 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 tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]
$ docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]
$ docker save [OPTIONS] IMAGE [IMAGE...]
$ docker load [OPTIONS]
```

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

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

Docker commands - 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
docker cp	Copy files/folders between a container and the local filesystem
docker login	Log in to a Docker registry (e.g. Docker hub , ReDii)
docker logout	Log out from a Docker registry
docker logs	Fetch the logs of a container
docker create	Create a new container (**without starting it**)
docker diff	Inspect changes to files or directories on a container's filesystem
docker export	Export a container's filesystem as a **tar archive**
docker info	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

# 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
```

Hands-on : 03_Docker_Commands

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`

문의처 : 정상업 / rogallo.jung@samsung.com