redis 설치하기

docker install : https://docs.docker.com/engine/install/ubuntu/

docker-compose install : https://docs.docker.com/compose/install/

Redis - Official Image | Docker Hub
Redis is an open source key-value store that functions as a data structure server.
https://hub.docker.com/_/redis

Redis 설치방법 1

Docker Hub에 등록된 이미지를 다운받아, 그 이미지를 바탕으로 Container를 생성

1. 최신버전 가져오기

pull 명령을 통해 docker hub에 있는 이미지 다운로드

```
sudo docker pull redis
```

※ 버전을 지정해서 이미지를 가져오고 싶은 경우
docker pull redis:5.0.3

```
ubuntu@ip-172-31-42-9:~$ sudo docker pull reids
Using default tag: latest
Error response from daemon: pull access denied for reids, repository does not exist or may require 'docker login':
 denied: requested access to the resource is denied
ubuntu@ip-172-31-42-9:~$ sudo docker pull redis
Using default tag: latest
latest: Pulling from library/redis
bd897bb914af: Pull complete
24daa36e9c8d: Pull complete
fae61c888a3a: Pull complete
e498372c1890: Pull complete
61691b3c8cfe: Pull complete
ffd564714c18: Pull complete
Digest: sha256:0c0484b1d1ff36faace984fe9d8e0fe58892ecc34a4859b97171045b9cd343e1
Status: Downloaded newer image for redis:latest
docker.io/library/redis:latest
```

```
sudo docker images
```

2. Redis 도커 네트워크 생성

```
sudo docker network create redis-net
```

redis-cli도 같이 구동해서 통신해야하기 때문에 2개의 컨테이너를 실행할 것이고, 그 연결을 위해서 docker network 구성을 먼저함

```
ubuntu@ip-172-31-42-9:~$ sudo docker network create redis-net
e200768620e9c206a57aaac9df5e17972277fd2bde5de93a0bafe5428df20c31
ubuntu@ip-172-31-42-9:~$ sudo docker network ls
NETWORK ID     NAME            DRIVER    SCOPE
5bb557027ada   bridge          bridge    local
692a7ce6a807   hadoop3_bridge  bridge    local
a34a59ac0fcf   host            host      local
8209f0541dbd   none            null      local
e200768620e9   redis-net       bridge    local
```

```
sudo docker network ls
```

3. Redis server 실행하기

```
sudo docker run --name myredis --network redis-net -d -p 6379:6379 redis
```

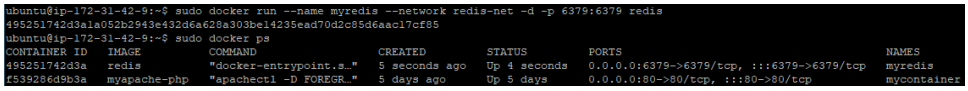
도커 컨테이너 실행 : docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

-d : detach, Container 생성 후 Background로 실행 (이 옵션 없을 시, Redis 같은 경우에는 Console에 Redis log가 지속적으로 표시됩니다.)

-p : publish, 해당 Container의 외부 Port 지정 (10000:6379) (외부:내부) 형식으로 지정

—network : 컨테이너를 실행할 때 —network 옵션을 명시하지 않으면 기본적으로 bride라는 이름의 디폴트 네트워크에 붙게됨

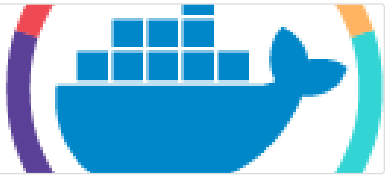
-requirepass redis123: 비밀번호 설정



docker run

Run a command in a new container The docker run command first creates a writeable container layer over the specified image, and then starts it using the specified command. That is, docker run is equivalent to the API /containers/create then/containers/(id)/start. A stopped container can be restarted with all its

https://docs.docker.com/engine/reference/commandline/run/



도커 run 명령어에 관한 옵션

4. redis-cli 접속

데이터의 추가, 조회, 삭제를 위한 interface

a. Docker의 redis-cli로 접속

```
sudo docker run -it --network redis-net --rm redis redis-cli -h myredis
```

—network 옵션으로 컨테이너를 실행하면서 바로 연결할 네트워크를 지정

—rm 옵션은 실행할때 컨테이너 id가 존재하면 삭제 후 run

docker network를 설정했기 때문에 myredis라는 도커컨테이너 이름을 사용해서 redis server에 접속

b. Redis-cli로 직접 접속하기 (연결된 6379 포트 사용)

```
#redis tool 설치
sudo apt install redis-tools

sudo redis-cli -p 6379
```

5. redis test

redis 명령어 요약

구분	SET	GET	POP	REM	INCR	집합연산
Strings	SET	GET	-	DEL	INCR	-
Lists	LPUSH	LRANGE	LPOP	LREM	-	-
Sets	SADD	SMEMBERS	SPOP	SREM	-	SUNION

구분	SET	GET	POP	REM	INCR	집합연산
ZSets	ZADD	ZRANGE	ZPOPMIN	ZREM	ZINCRBY	ZUNION
Hashes	HSET	HGET	-	HDEL	HINCRBY	-
Streams	XADD	XREAD	-	XDEL	-	-

```
myredis:6379> keys *
(empty array)
myredis:6379> set test "I like coffee"
OK
myredis:6379> keys *
1) "test"
myredis:6379> get test
"I like coffee"
```

6. 컨테이너 shell로 redis 접속

```
sudo docker exec -it myredis /bin/bash
```

docker exec {options} [container명/container ID] [명령어]

redis의 경우 container에 bin/bash가 없어 bin/sh로 실행하면 내부접속이 가능

■ 영구 저장소로 실행

```
sudo docker run --name myredis -v /home/ubuntu/redis:/data --network redis-net -d redis redis-server --save 60 1 --loglevel warning
```

--save 60 1 : 1회 쓰기 작업이 수행된 경우 60초마다 DB 스냅샷을 저장 (데이터 세트에 M개 이상의 변경사항이 있는 경우 N초마다 데이터 세트를 디스크에 자동으로 저장하도록 함)

--loglevel warning : 많은 로그가 발생함으로 log 레벨 옵션을 줌

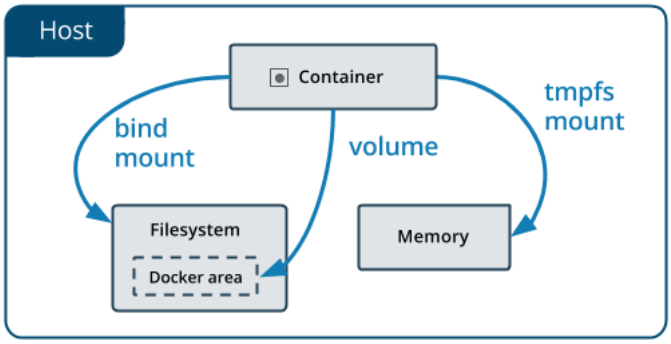
```
ubuntu@ip-172-31-42-9:/var/log/redis$ sudo docker exec -it myredis /bin/bash
root@20b2e3242a05:/data# ls -al
total 12
drwxr-xr-x 2 redis redis 4096 Oct  8 12:25 .
drwxr-xr-x 1 root  root  4096 Oct  8 12:24 ..
-rw-r--r-- 1 redis redis  122 Oct  8 12:25 dump.rdb
```

dump.rdb가 생성된 것을 알 수 있음

--volumes-from some-volume-container 혹은 -v /docker/host/dir:/data을 사용하면 데이터는 VOLUME /data에 저장된다.

```
ubuntu@ip-172-31-42-9:/var/log/redis$ cd /home/ubuntu/redis/
ubuntu@ip-172-31-42-9:~/redis$ ls -al
total 12
drwxr-xr-x 2 systemd-coredump root             4096 Oct  8 13:01 .
drwxr-xr-x 8 ubuntu           ubuntu           4096 Oct  8 11:35 ..
-rw-r--r-- 1 systemd-coredump systemd-coredump  123 Oct  8 13:01 dump.rdb
```

※ 영구저장소 또는 볼륨저장소는 사용자가 실행 중인 컨테이너의 파일시스템에 영구계층을 추가할 수 있는 방법을 제공



Volumes 은 도커 컨테이너에 의해 생성, 사용되는 호스트 파일 시스템에 저장. 따라서, 파일을 많이 생성하더라도 컨테이너 이미지의 사이즈가 커지지 않는다.

▼ volumes

- 볼륨은 컨테이너가 초기화될때 만들어 진다. 컨테이너의 기본 이미지가 특정 마운트 지점에 데이터를 가지고 있으면, 그 데이터는 볼륨 초기화때 새로운 볼륨에 복사된다.
- 여러개의 컨테이너가 공유해서 사용할 수 있다.
- 데이터 볼륨의 데이터를 직접 변경할 수 있다.
- 데이터 볼륨에서 작업한 내역은 이미지를 업데이트할 때 포함되지 않는다.
- 컨테이너를 지워도 데이터 볼륨에 작업한 데이터는 남아 있는다.

단순히 볼륨을 생성하는 것뿐만 아니라 호스트에 있는 파일이나 디렉토리를 컨테이너에 마운트할수도 있다.

방법은 간단하다.

- v [호스트 경로]:[컨테이너 경로]

이런 식으로 옵션을 주면 컨테이너 내부에서 [컨테이너경로]로 접근하면 호스트의 [호스트경로]에 있는 데이터에 접근할 수 있게 된다. 이때 경로는 디렉토리를 명시할수도 있고, 파일을 명시할수도 있다. 이때 [컨테이너경로]에 기존 데이터가 있더라도, 기존데이터는 무시되고 –v 옵션에서 지정한 [호스트경로]에 있던 데이터가 컨테이너 내부의 [컨테이너경로] 에 있게 된다.

데이터 볼륨은 컨테이너를 백업, 복구할때도 유용하게 사용할 수 있다. 아래 같은 명령을 통해서 /dbdata디렉토리를 통째로 백업해 올 수 있다.

```
docker run --volumes-from dbdata -v $(pwd):/backup ubuntu tar cvf /backup/backup.tar /dbdata
```

이렇게 백업한 데이터는 아래처럼 간단하게 복구해서 사용할 수 있다.

```
docker run --volumes-from dbdata2 -v $(pwd):/backup ubuntu cd /dbdata && tar xvf /backup/backup.tar
```

Bind mounts는 볼륨과 다르게 호스트 파일 시스템 어디에서나 데이터를 저장할 수 있고, 도커가 아닌 프로세스도 이 파일을 수정할 수 있다.

■ 도커용 redis.conf 파일을 지정

/home/ubuntu/redis/redis.conf 파일 생성

```
# daemonize yes로 하면 구동되지 않음
daemonize no

# bind 127.0.0.1
protected-mode no
```

```
#redis 서버는 기본적으로 6379
port 6001
logfile "redis.log"

# Working dir를 지정
dir /usr/local/etc/redis
```

▼ redis.conf 설정

redis.conf 설정

Aa 설정값	설명
daemonize [yes||no] default no	Redis는 기본적으로 daemon으로 실행하지 않습니다. 만약 daemon으로 실행하고 싶다면 'yes'를 사용하시면 됩니다. 만약 daemon으로 실행시 '/var/run/redis.pid' 파일에 pid를 디폴트로 기록할 것입니다.(파일경로 변경가능 - pidfile 설정)
port [number] default 6379	Connection 요청을 받을 포트를 지정하는 설정입니다.
bind [ip] default all interface	Redis를 bind 할 특정 interface를 지정하는 설정입니다. 만약 명시하지 않으면 모든 interface로부터 들어오는 요청들을 listen합니다.
unixsocket [path], unixsocketperm [number]unixsocket /tmp/redis.sock / unixsocketperm 755	소켓으로 Redis Server를 붙게할 수 있는 설정입니다.
timeout [second]	클라이언트의 idle 시간이 해당 초만큼 지속되면 connection을 닫습니다. 0으로 지정할시 항상 connection을 열어둡니다.
loglevel [level - debug, verbose,notice,warning]	로그레벨지정
logfile [file path]	로그파일이 쓰여질 파일 경로를 지정합니다.
slaveof -> replicaof [master ip] [master port] 최근 버젼은 slaveof에서 replicaof로 변경됨.(어느 버전부터 바뀐지는 확인필요)	Master-Slave 관계를 구성하기 위하여 Master의 replication Node가될 slave Node 설정파일에 Master Node의 ip와port 정보를 기입하는 설정이다. 즉, Master의 데이터가 slave에 replicate되게 된다.
masterauth [master-password]	만약 Master Node 설정파일에 Slave Node가 붙기 위한 암호요구를 하였다면(require pass [password]), Slave Node의 설정파일은 Master Node가 지정한 패스워드로 해당 설정을 해줘야한다.
slave-server-stale-data [yes||no] default yes-> replica-server-stale-data [yes||no] default yes	만약 slave가 master와의 connection이 끊겼거나 replication 중일 때, 취할수 있는 행동 2가지가 있다. 1. [yes] - 유효하지 않은 데이터로 클라이언트 요청에 답한다. 2. [no] - 몇몇 명령을 제외하고 요청에 대해 error로 응답한다.
repl-ping-slave-perid [seconds] default 10	Slave가 지정된 시간마다 ping을 날린다.
repl-timeout [seconds] default 60	타임아웃 시간을 설정한다. 기본적으로 ping 주기보다 큰 시간을 주어야한다.
require pass [password]	클라이언트에게 패스워드를 요구한다.(Slave가 Master에 접근할때의 보안설정이기도하다)
maxclient [size]	한번에 연결될 수 있는 클라이언트의 연결수이다.
maxmemory [bytes]	Redis Server의 최대 메모리 사용양을 설정한다. 만약 memory에 한계가 도달했다면 Redis는 선택된 eviction policy(제거정책)에 따라 key들을 제거한다. 만약 제거에 실패한다면 read-only에 대한 명령만 수행할 수있게 된다.
maxmemory-policy [policy]	max memory 도달시 취할 정책을 설정한다 1) volatile-ru : expire가 설정된 key들 중 LRU 알고리즘을 이용해 선택된 key를 제거한다. 2) alleys-lru : 모든 keys에 대해 LRU 알고리즘을 적용해 제거한다. 3) volatile-random : expire가 설정된 key들 중 임의의 key를 제거한다. 4) alleys-random : 모든 keys중 랜덤하게 제거한다. 5) volatile-ttl : expire time이 가장 적게 남은 key를 제거한다. 6) noeviction : 어떠한 key도 제거하지 않는다. read-only작업만 허용하고 write와 그 밖에 명령은 error를 내뿜는다.
appendonly [yes||no]	Redis는 기본적으로 비동기로 disk에 dataset의 dump를 남긴다. 이 방법은 최근 데이터가 손실되어도 큰 문제가 없다면 사용해도 좋지만, 하나라도 데이터의 유실을 원치 않는 상황이라면 yes로 설정하여 요청받는 모든 write를 disk에 쓰도록하면 된다.(default file appendonly.aof)
appendfilename [file name]	append only 파일 이름을 설정한다.

※ redis.conf 파일 지정해서 실행(volume 지정)하면 redis 컨테이너용 Dockerfile이 필요없음

```
sudo docker run -v /home/ubuntu/redis:/usr/local/etc/redis --name myredis2 -d -p 6001:6001 redis redis-server /usr/local/etc/redis/red
```

-v : volume, 외부 볼륨 연동 /home/ubuntu/redis:/usr/local/etc/redis (외부 OS 디렉토리:컨테이너 내부 디렉토리) - 상호 데이터가 공유됨

■ Dockerfile로 이미지를 만들어 실행

- Dockerfile은 이미지를 바탕으로 몇가지 명령을 실행할 수 있는 로직을 담은 파일
- Dockerfile을 통해 더 유연하게 이미지에 대한 설정을 변경할 수 있고, 그 설정에 이미지를 담아 배포할 수 있음
- Dockerfile을 작성하면 유지보수에 유리

/home/ubuntu/redis/Dockerfile 파일 생성

```
#기반이 되는 이미지 레이어
FROM ubuntu:18.04

#환경변수 지정
ENV WDIR /usr/local/etc/redis

# config 파일을 복사할 디렉토리를 미리 만들어 에러 방지
RUN mkdir $WDIR

RUN apt-get update && apt-get install -y --no-install-recommends apt-utils
RUN apt-get update
RUN apt-get install -y redis-server
RUN apt-get clean

# 볼륨을 지정. 호스트 디렉토리를 이 볼륨에 마운트 할 수 있음
VOLUME $WDIR

# Docker 이미지 내부에서 CMD에서 설정한 실행파일이 실행될 디렉터리
WORKDIR $WDIR

#컨테이너 외부에서 사용할 포트를 지정
#컨테이너간 연결에 사용되며, host os와 연결을 할 뿐 외부와 노출이 되지 않기 때문에 포트를 노출하기 위해서는 run -p host_port:container_port로 호스트 포트와 연
EXPOSE 6001

COPY redis.conf /usr/local/etc/redis/redis.conf

# 컨테이너 안에서 실행하고자 하는 프로세스를 띄우는 명령
CMD [ "redis-server", "/usr/local/etc/redis/redis.conf" ]
```

Dockerfile을 사용하여 빌드 - 이미지 생성

```
sudo docker build --tag bagidevelopment/redis:6.2.6 .
```

docker 이미지 실행

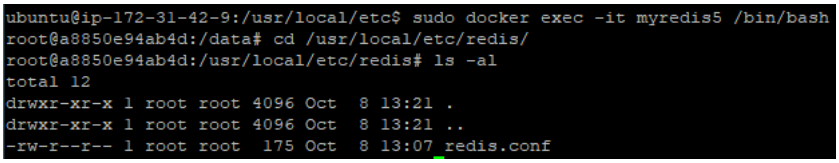
```
sudo docker run --name myredis5 --network redis-net -d -p 6001:6001 bagidevelopment/redis:6.2.6
```

redis-cli 실행

```
sudo docker run -it --network redis-net --rm redis redis-cli -h myredis5 -p 6001:6001
```

shell로 redis server 접근

```
$ sudo docker exec -it myredis5 /bin/bash
$ cd /usr/local/etc/redis/
$ ls -al
```



■ Docker compose 설정파일을 사용해 실행

- container들의 image 정보들을 담고 있는 설정파일
- docker-compose 파일은 YAML형식으로 지원버전과 함께 서비스, 네트워크, 볼륨 등을 정의
- 한번에 여러가지 소프트웨어에 대한 이미지 정보를 담아 빌드할 수 있기 때문에 사용자는 Docker-compose로 포함된 소프트웨어에 대한 Container를 한번에 만들 수 있음

redis-docker-compose.yml 작성

```
version: '3.8'
services:
    myredis6:
      image: bagidevelopment/redis:6.2.6
      command: redis-server /usr/local/etc/redis/redis.conf
      container_name: myredis6
      hostname: myredis6
      labels:
        - "name=redis"
        - "mode=standalone"
      ports:
        - 6001:6001
      networks:
        - default
      volumes:
        - .:/usr/local/etc/redis
networks:
    default:
      external:
        name : redis-net
```

LABEL key=value

label을 지정하면 docker ps -f (filter) label=mode=standalone 처럼 standalone/sentinel/cluster 종류별로 조회가능

hostname을 지정하면 쉘로 접속했을 때 hostname을 표시가능

docker-compose 버전

```
sudo docker-compose -v
```

docker compose 설정파일을 이용해 Redis 실행

```
sudo docker-compose -f redis-docker-compose.yml up -d
```

redis-cli

```
sudo docker run -it --network redis-net --rm redis redis-cli -h myredis6 -p 6001:6001
```

shell로 redis server 접근

```
sudo docker exec -it myredis6 /bin/bash
```

[docker-compose] 설정 가이드

는 docker-compose.yml 이라는 Compose 정의 파일에 시스템 안에서 가동하는 여러 서버들의 구성을 모아서 정의 함이정의 파일은 YAML 형식으로 기술 # 버전을 지정(현재 메이저 최신 버전) version: "3" # 서비스 정의 services: webserver: image: ubuntu ports: - "80:80" networks: - webnet redis: image: redis networks: - webnet # 네트

https://freedeveloper.tistory.com/182

참고 docker compose 참고 설정

그외 참고자료

Docker기반 Redis 구축하기 - (3) Dockerfile | Carrey`s 기술블로그

이전 예제에서는 Docker Hub에 등록된 이미지를 다운받아서 사용하였고, 이미지를 바탕으로 Container를 생성 할 수 있었습니다. Docker Hub에는 각 소프트웨어에 대한 공식이미지도 있지만, Docker Hub 사용자들이 공식이미지를 바탕으로 커스터마이징하여 배포한 이미지도 등록되어 있습니다. 어떻게 이미지를 빌드하고 배포 할 수 있었을까요?

https://jaehun2841.github.io/2018/12/01/2018-12-01-docker-3/#run-container-%EB%82%B4%EB%B6%80%EC%97%90%EC%84%9C-%EB%AA%85%EB%A0%B9-%EC%8B%A4%ED%96%89

가장 빨리만나는 Docker: 클라우드 플랫폼 어디서나 빠르게 배포하고 실행할 수 있는 리눅스 기반 경량화 컨테이너

저작권 안내 책 또는 웹사이트의 내용을 복제하여 다른 곳에 게시하는 것을 금지합니다. 책 또는 웹사이트의 내용을 발췌, 요약하여 발표 자료, 블로그 포스팅 등으로 만드는 것을 금지합니다. 가장 빨리 만나는 Docker 출간 및 원고 공개 가장 빨리 만나는 Docker(이하 '책')의 저작권은 이재홍에게 있습니다. 책의 출판권 및 배타적발행권과 전자책의 배타적전송권은 (주)도서출판 길벗

http://pyrasis.com/book/DockerForTheReallyImpatient