

1. Overview

Docker 는 개발자와 시스템 관리자가 컨테이너 기술을 사용하여 어플리케이션을 개발, 배포 및 실행하기 위한 플랫폼이다.

- 유연성 (Flexible) : 복잡한 어플리케이션들도 모두 컨테이너화 할 수 있다.
- 경량화(Lightweight) : 컨테이너는 호스트 커널을 활용하고 공유한다.
- 변화관리 편의성 (InterChangeable) : 업데이트 및 업그레이드를 즉시 배포할 수 있다.
- 포터블 (Portable) : 로컬로 구축하고, 클라우드와 가상화에 배치도 가능하며, 어디서나 실행이 가능하다.
- 확장성 (Scalable) : 컨테이너 복제본을 늘리고 자동으로 배포할 수 있다.
- 스택화(Stackable) : 서비스들에 대한 수직적 또는 수평적 디자인이 매우 용이하다.



이미지 및 컨테이너 개념

컨테이너는 이미지를 실행하여 시작이 된다.

이미지는 코드, 런타임, 라이브러리, 환경 변수 및 구성 파일, 어플리케이션 등을 실행하는 데 필요한 모든 것을 포함하는 실행가능한 패키지이다.

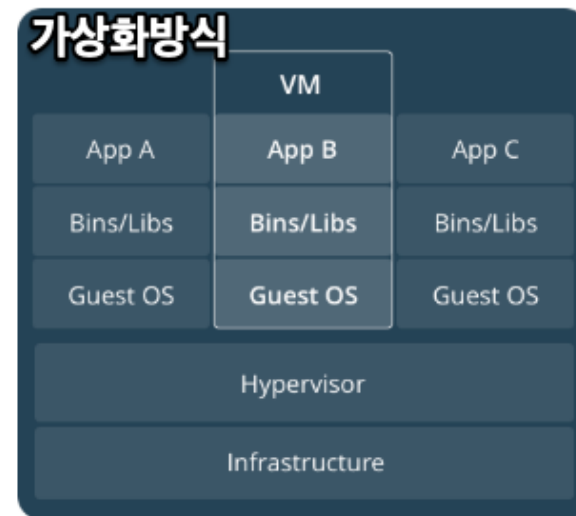
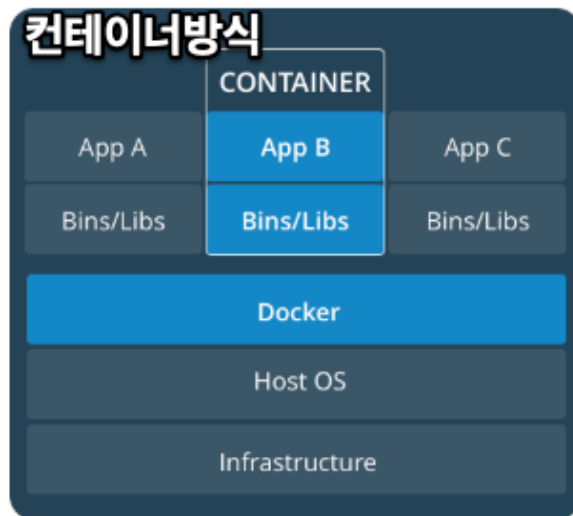
컨테이너는 이미지의 런타임 인스턴스 단위이며, 이미지가 컨테이너로 실행이 될 때 메모리로 로딩이 된다.

컨테이너 및 가상화 머신

컨테이너의 동작 방식은 여러 개의 컨테이너들이 같은 호스트 시스템의 커널을 공유하는 방식이다.

이러한 방식은 가상화 기반의 방식에 비해서 메모리를 경량화 시킬 수 있고, 별도의 프로세스를 실행하는 방식을 사용한다.

이에 비해 많이 비교되는 가상화 방식은 하이퍼바이저를 통해 호스트 리소스에 대한 제어권을 소유하는 방식이다.



도커 구성을 위한 환경

Docker 는 현재 상용 버전과 커뮤니티 버전 두가지를 모두 제공하고 있다.

여기에서는 Docker CE(Community Edition) 기반으로 Mac OS, CentOS, Ubuntu Linux 등에서 설치 방식을 다룬다.

- Docker CE 정보 소개
- 보통 Stable 버전을 사용하는 것이 좋다.
- Stable
- 일반적으로 가장 최신의 릴리즈 버전을 이야기 한다.
- Test
- 일반적으로 가용성이 검증 되기 이전의 버전을 이야기 한다.
- Nightly
- 차기 주요 출시를 위한 진행 중인 최신 빌드 버전을 일컫는다.

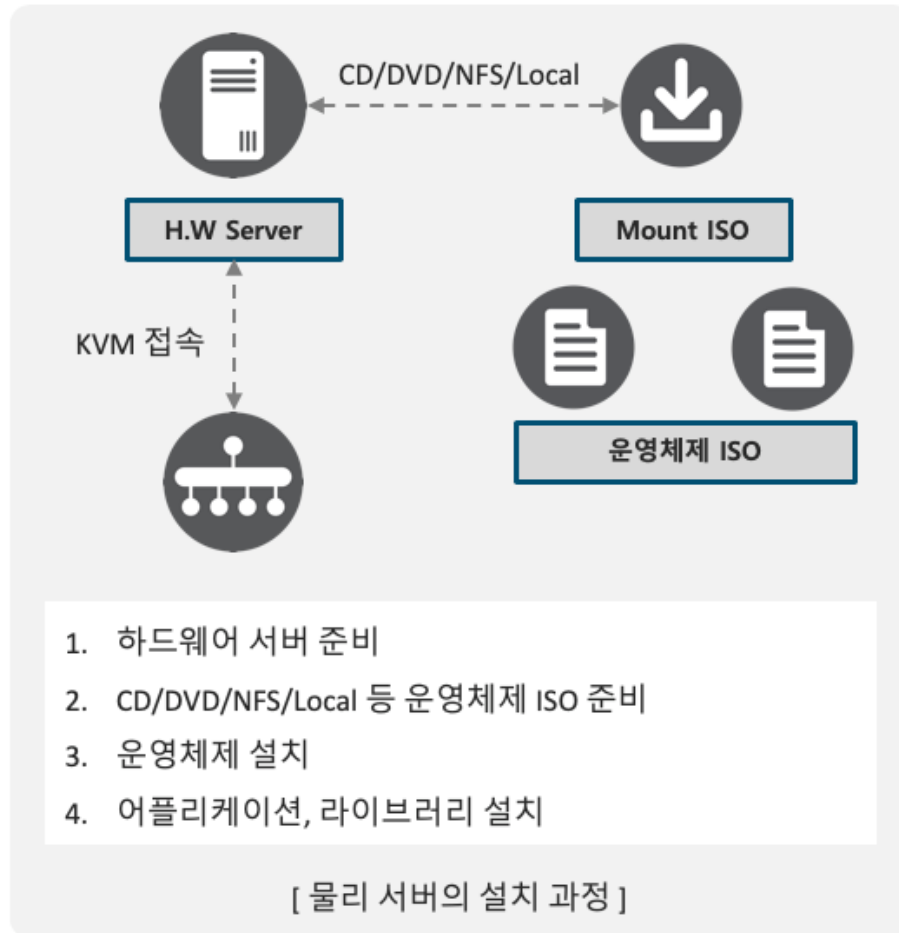
아래와 같이 데스크 톱 버전과 서버 버전을 운영체제별 지원하고 있다.

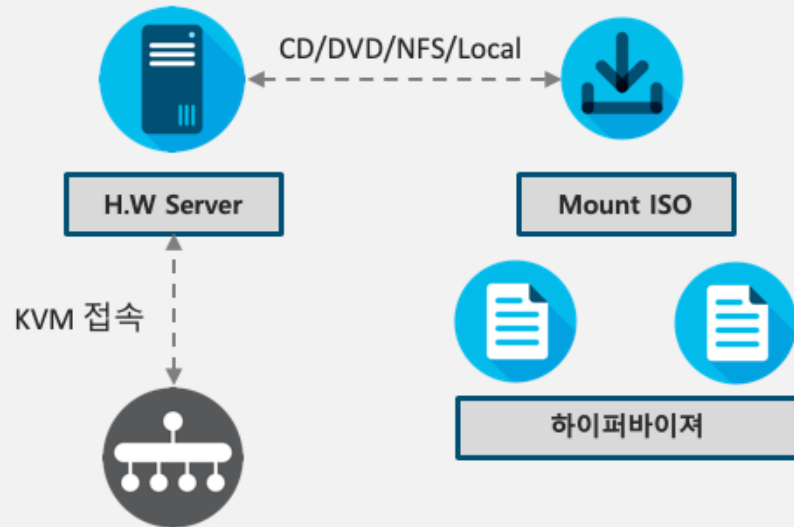
Platform	x86_64				
Docker Desktop for Mac (macOS)	✓				
Docker Desktop for Windows (Microsoft Windows 10)	✓				

Platform	x86_64 / amd64	ARM	ARM64 / AARCH64	IBM Power (ppc64le)	IBM Z (s390x)
CentOS	✓		✓		
Debian	✓	✓	✓		
Fedora	✓		✓		
Ubuntu	✓	✓	✓	✓	✓

버전 넘버링을 통해서 출시일자와 패치를 예측 할 수 있는데, YY.mm.<patch>와 같은 방식으로 구분짓는다.

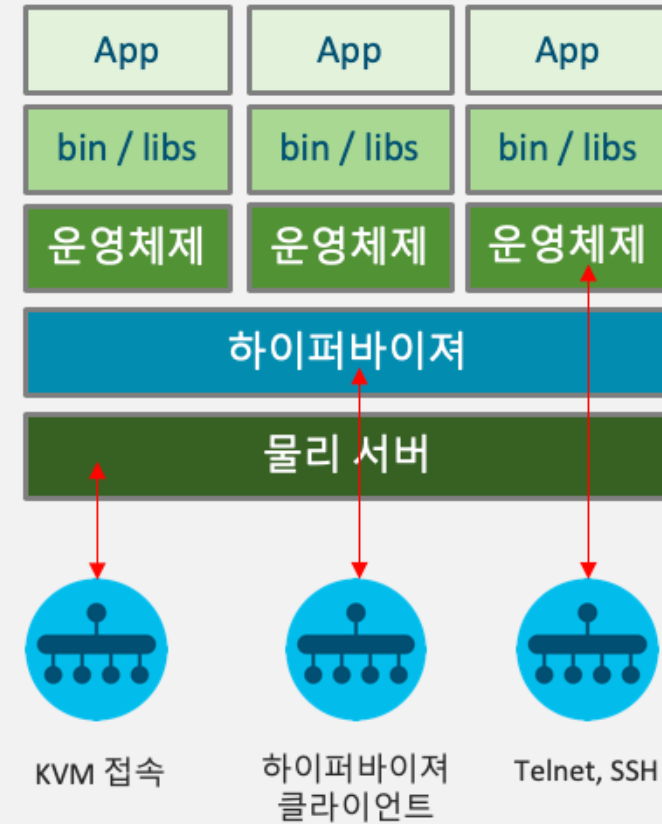
도커의 설치 전 도커의 아키텍처 이해



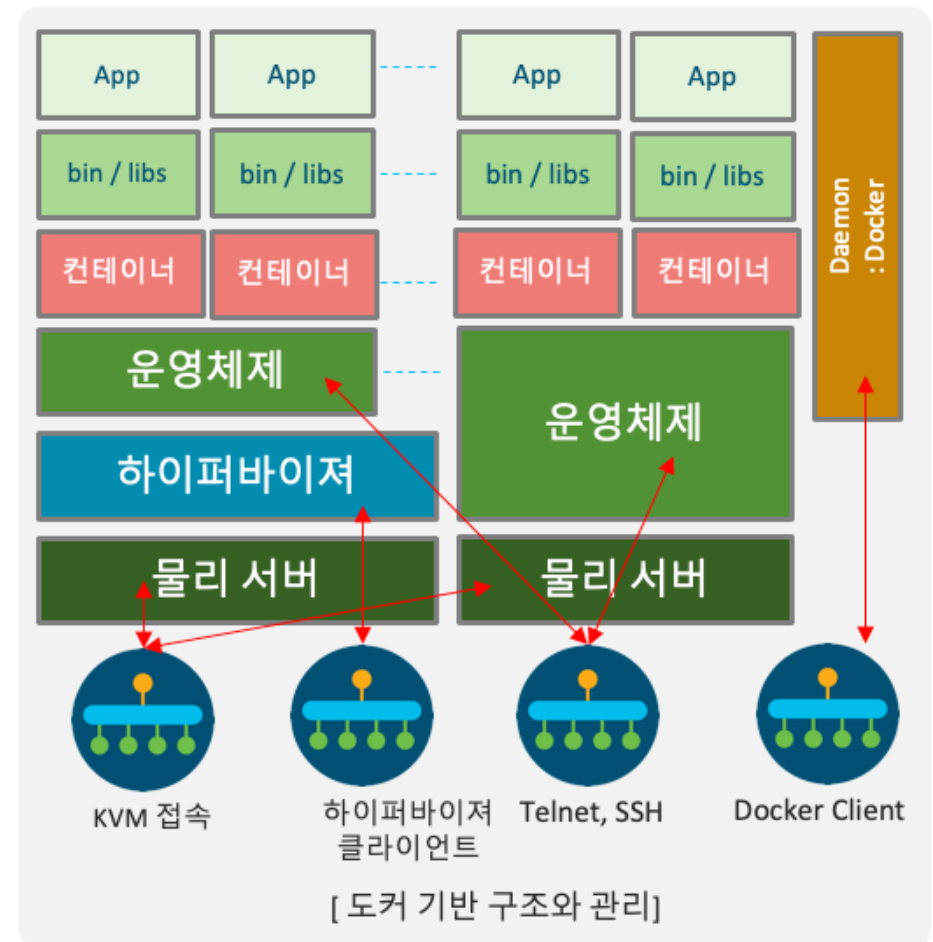
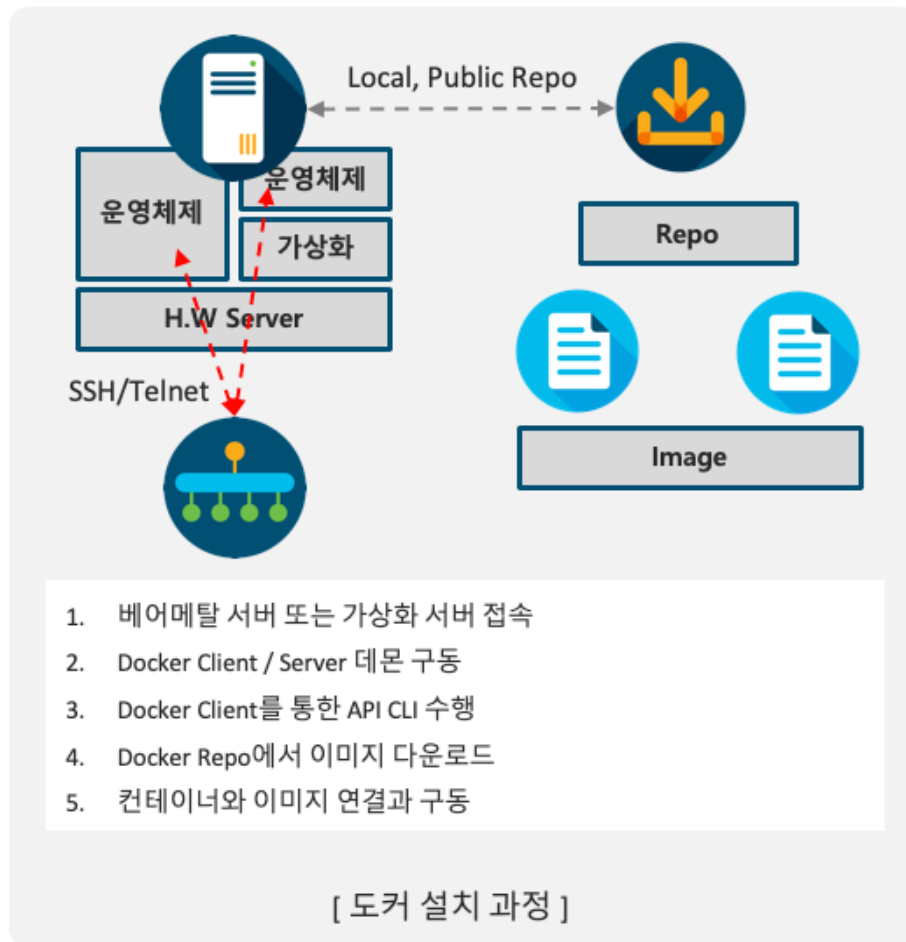


1. 하드웨어 서버 준비
2. CD/DVD/NFS/Local 등 하이퍼바이저 ISO 준비
3. 하이퍼바이저 설치
4. 가상화 클라이언트 설치
5. 가상화 클라이언트를 통한 가상화 서버에 운영체제 설치
6. 운영체제에 어플리케이션, 라이브러리 설치

[가상화서버의 설치 과정]



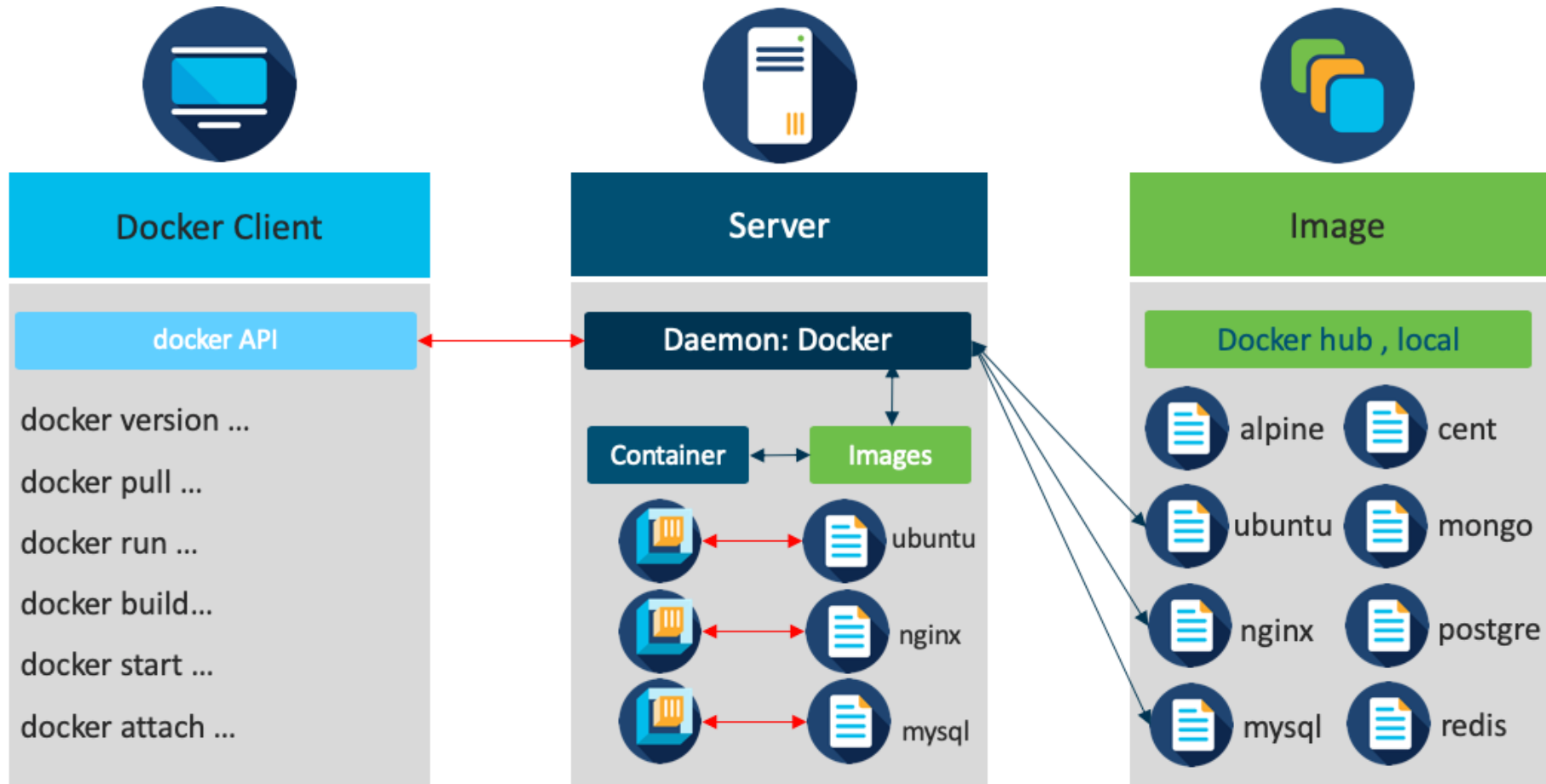
[가상화 서버 기반 구조와 관리]



도커의 설치와 구성 과정을 보면, 앞서 2 개의 플랫폼 모두에서 적용이 가능하기 때문에 설치를 위한 선수 조건은 베어메탈의 설치와 관리, 가상화 서버의 설치와 관리를 포함하고 있다.

다만 가상화나 베어메탈과 다른 점은 서비스 구동을 위한 설치과정에서 도커는 Image 를 사용한다는 것이다.

Docker 를 구성, 운영하는데 있어서 최종적인 아키텍처



물리 서버, 가상화, 클라우드

1. 도커 클라이언트

도커를 구성하고 운영 관리하는 패키지로 주로 관리자가 가장 많이 사용

2. 도커 서버

도커 서버는 따로 규정하지 않고, 도커를 실행하는 데몬이 동작하는 개인 랩탑, 서버, 가상화, 클라우드 모두 사용

3. 이미지

이미지는 로컬 또는 Docker Hub 등과 같은 퍼블릭 Repository 를 사용

도커의 설치

도커의 설치에는 다양한 방법이 있으나, 일반적으로 바이너리를 통해서 직접 설치하거나 저장소를 통해서 각 운영체제에서 제공되는 배포 방법을 통해서 구성할 수 있다.

Ubuntu 에서의 설치 방법

#docker repo 기반 설치를 위한 패키지 설치

```
sudo apt-get install -y curl apt-transport-https ca-certificates software-properties-common
```

#docker reop 추가

```
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 $(lsb_release -cs) stable"
```

```
sudo apt update
```

#docker ce 설치

```
sudo apt install -y docker-ce
```

#특정버전 설치를 위한 확인

```
sudo apt-cache policy docker-ce | sort -r
```

```
sudo apt-get install docker-ce="특정버전"
```

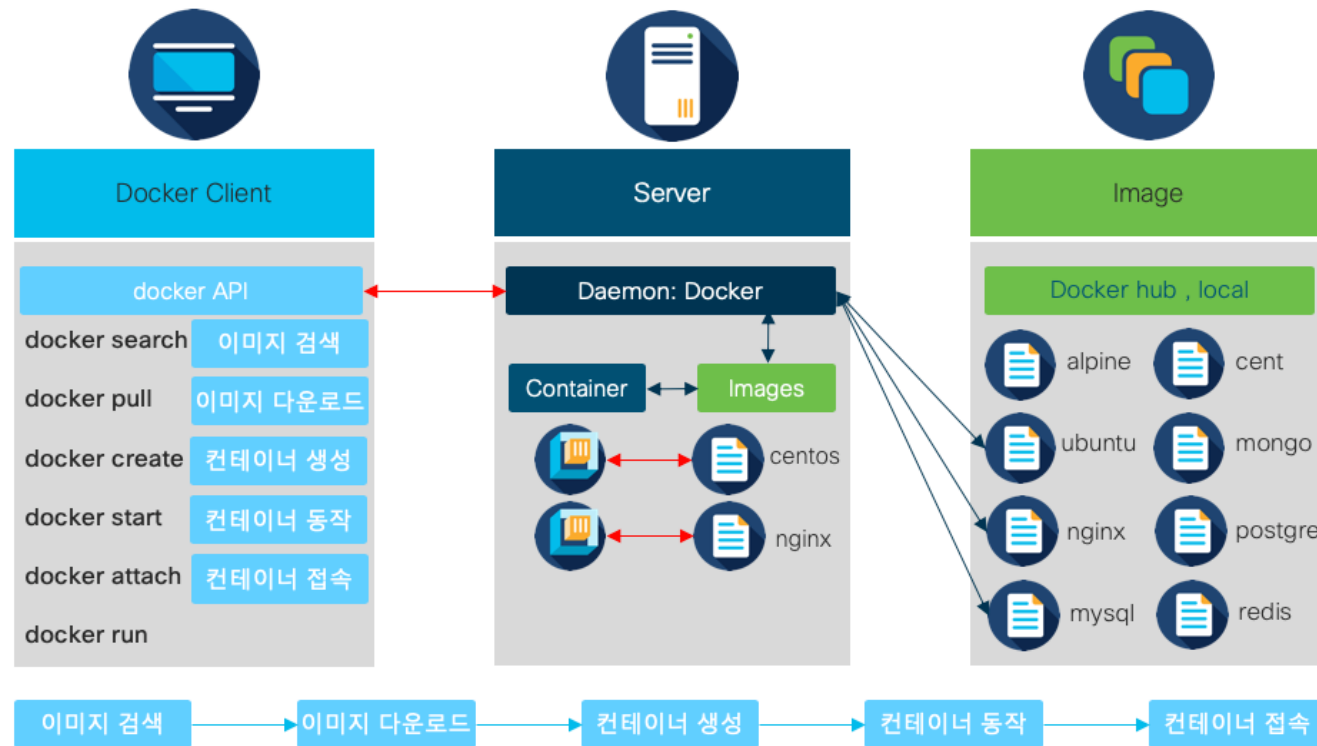
systemctl 을 통해 서비스 구동 및 등록.

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

2. Docker 기본동작

도커의 기본 설치 이후 동작을 위해서는 도커의 기본 동작 흐름만 이해하면, 컨테이너 자체를 만드는 것은 어렵지 않다. 기본 로직을 이해하면 몇가지 옵션들과 함께 간편하게 도커를 구동할 수 있다.



Docker Hub 로 부터 이미지 검색과 다운로드

필요한 이미지 검색

```
docker search centos
```

official image 검색

```
docker search --filter "is-official=true" centos
```

Docker Image Download

```
docker pull centos
```

Docker Images 확인

```
docker images
```

Docker Image 기반으로 Container 생성과 구동

이미지 다운로드 - 이미지 기반 컨테이너 생성 - 컨테이너 동작 - 컨테이너 접속

Image 를 가지고 컨테이너 생성

```
docker create -it centos
```

Image 를 가지고 컨테이너 생성

```
docker create --name centos01 -it centos
```

```
docker ps -a
```

컨테이너 동작

```
docker start centos01
```

```
docker ps -a
```

컨테이너 접속

```
docker attach centos01
```

컨테이너 종료 및 접속 종료

```
[root@44c1e6bac8d0 /]# exit
```

```
exit
```

```
docker ps -a
```

```
docker rm centos01
```

```
docker ps -a
```

docker run 명령을 통한 한꺼번에 명령 수행하기

Docker run 을 사용하면, image 가 없으면 자동으로 다운로드 받고 컨테이너를 생성하고 컨테이너 접속까지 수행 시킬 수 있음.

```
docker run --name centos01 -it centos
```

터미널 종료시 ctrl + p, ctrl+q 를 사용하면, 백그라운드로 동작

Port Forwarding 을 통한 컨테이너 구동

nginx 공식 이미지 검색

```
docker search --filter "is-official=true" nginx
```

nginx 공식 이미지 다운로드

```
docker pull nginx
```

nginx 동작을 8080 포트로 포트포워딩 시킴.

```
docker run --name nginx-test -p 8080:80 nginx
```

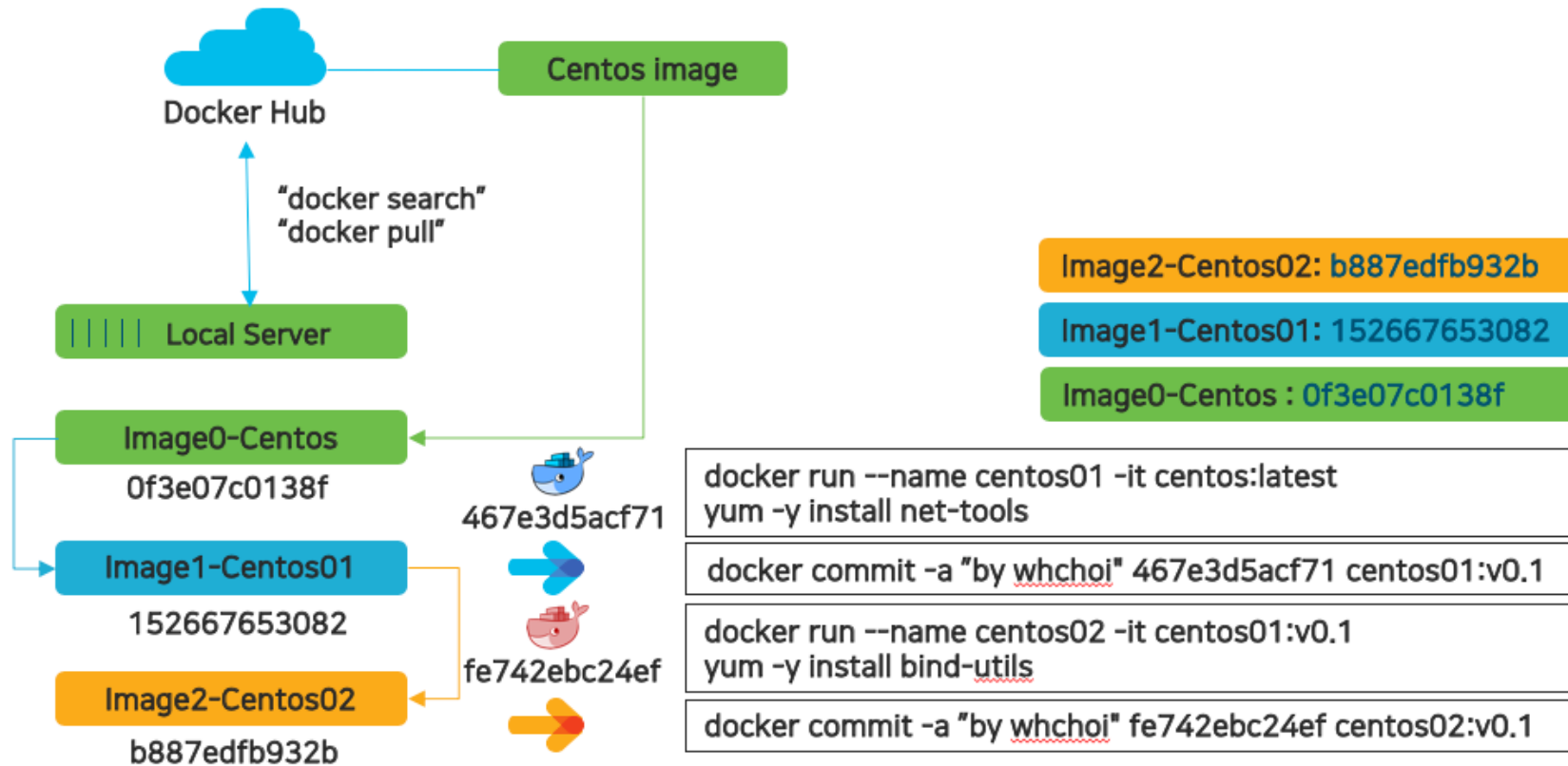
정상적으로 port forwarding 되는지 curl 을 통해 확인.

```
curl http://10.72.78.151:8080
```

docker ps 를 통해 현재 동작 중인 docker 확인 (-a 옵션을 사용하면 정지 중인 도커 까지 확인 할 수 있음)

docker ps

4. 도커 이미지



```
[root@cent154 ~]# docker history centos02:v0.1
```

IMAGE	CREATED	CREATED BY	SIZE
b887edfb932b	2 minutes ago	/bin/bash	33.6MB
152667653082	8 minutes ago	/bin/bash	40.8MB
0f3e07c0138f	3 weeks ago	/bin/sh -c #(nop) CMD ["/bin/bash"]	0B
<missing>	3 weeks ago	/bin/sh -c #(nop) LABEL org.label-schema.sc...	0B
<missing>	3 weeks ago	/bin/sh -c #(nop) ADD file:d6fdacc1972df524a...	220MB

Docker Images

docker registry 에서 필요한 docker image 를 검색할 수 있다.

```
[root@localhost ~]# docker search centos
```

```
[root@localhost ~]# docker search --filter "is-official=true" centos
```

```
[root@localhost ~]# docker pull centos
```

```
[root@localhost ~]# docker images
```

```
[root@localhost ~]# docker run --name centos01 -it centos:latest
```

```
[root@467e3d5acf71 /]# yum -y install net-tools
```

```
[root@localhost ~]# docker ps -a
```

```
[root@localhost ~]# docker commit -a "by whchoi" 467e3d5acf71 centos01:v0.1
```

```
[root@localhost ~]# docker images
```

```
[root@localhost ~]# docker history centos01:v0.1
```

```
[root@localhost ~]# docker run --name centos02 -it centos01:v0.1
```

```
[root@fe742ebc24ef /]# yum -y install bind-utils
```

```
[root@localhost ~]# docker ps -a
```

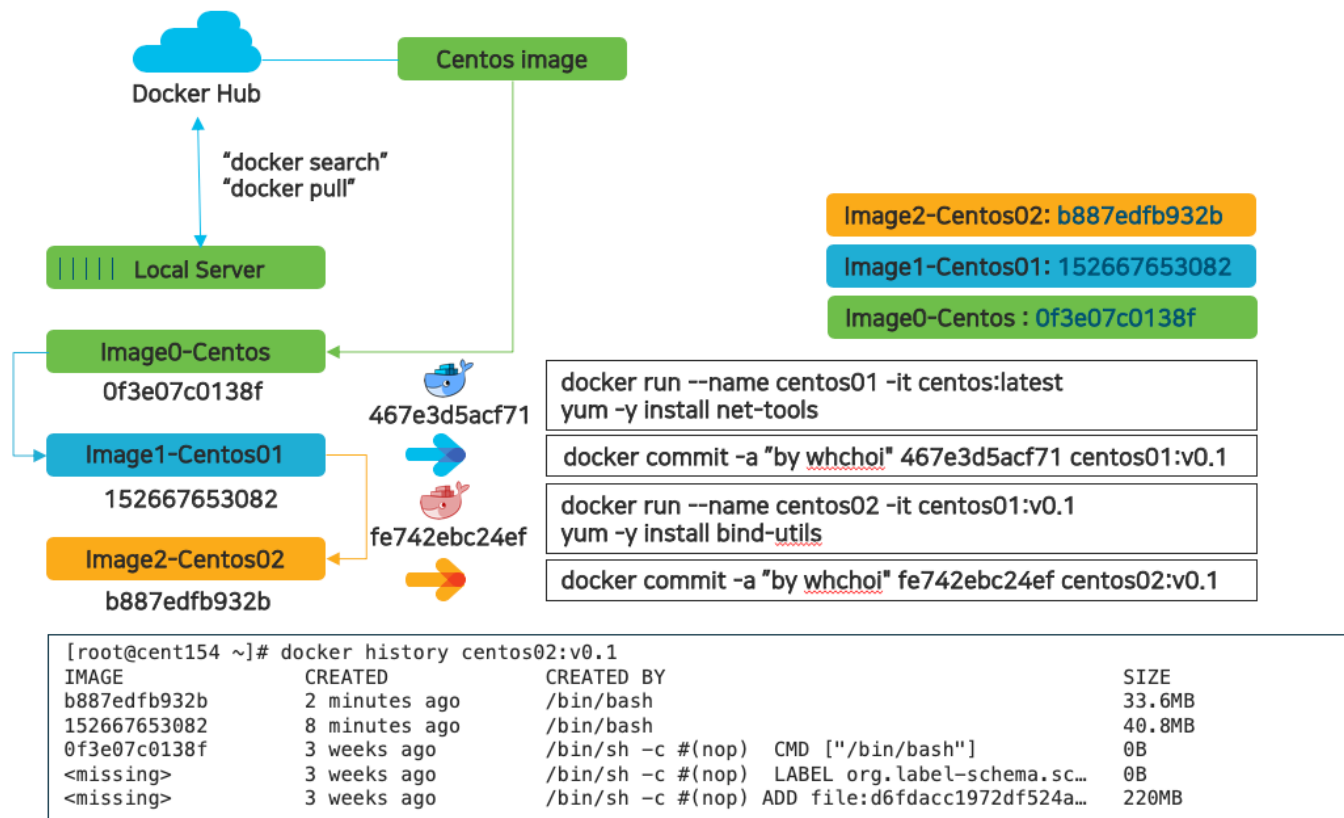
```
[root@localhost ~]# docker commit -a "by whchoi" fe742ebc24ef centos02:v0.1
```

```
[root@localhost ~]# docker history centos02:v0.1
```

```
[root@localhost ~]# docker rmi centos01:v0.1
```

```
[root@localhost ~]# docker rmi centos02:v0.1
```

```
[root@localhost ~]# docker rmi centos01:v0.1
```



docker 이미지 추출/로딩/배포

```
[root@localhost ~]# docker images
```

```
[root@localhost ~]# docker save -o whchoi.tar cisco_centos:v0.1
```

```
[root@localhost ~]# ls -al
```

```
[root@localhost ~]# docker rmi cisco_centos:v0.1
```

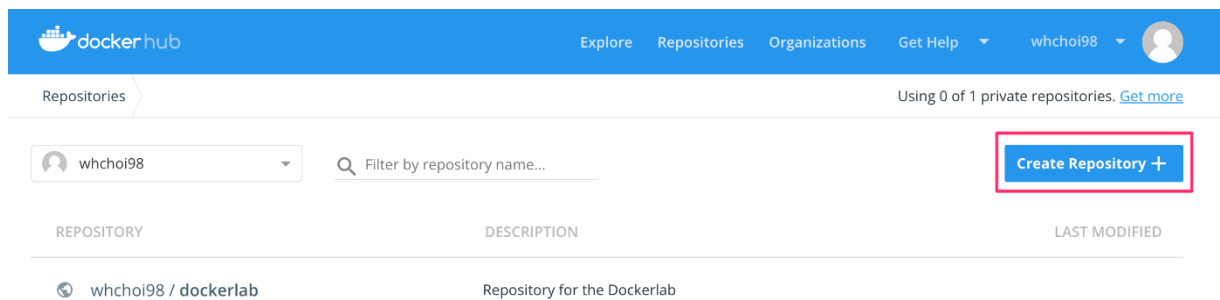
```
[root@localhost ~]# docker images
```


```
[root@localhost ~]# docker load -i whchoi.tar
```

```
[root@localhost ~]# docker images
```

```
[root@localhost ~]# docker run -it --name whchoi-os cisco_centos
```

```
[root@localhost ~]# docker run -it --name whchoi-os cisco_centos:v0.1
```



 dockerhub

ExploreRepositoriesOrganizationsGet Helpwhchoi98

RepositoriesCreate

Using 0 of 1 private repositories. [Get more](#)

Create Repository


whchoi98


dockerlab

Repository for the Dockerlab

Visibility


Using 0 of 1 private repositories. [Get more](#)


☒ Public 
Public repositories appear in Docker Hub search results

☐ Private 
Only you can view private repositories

Build Settings *(optional)*

Autobuild triggers a new build with every **git push** to your source code repository. [Learn More](#).

 Connected

 Disconnected

Cancel

Create

Create & Build

Pro tip

You may push a new image to this repository using the CLI:

```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

Make sure to change *tagname* with your desired image repository tag.

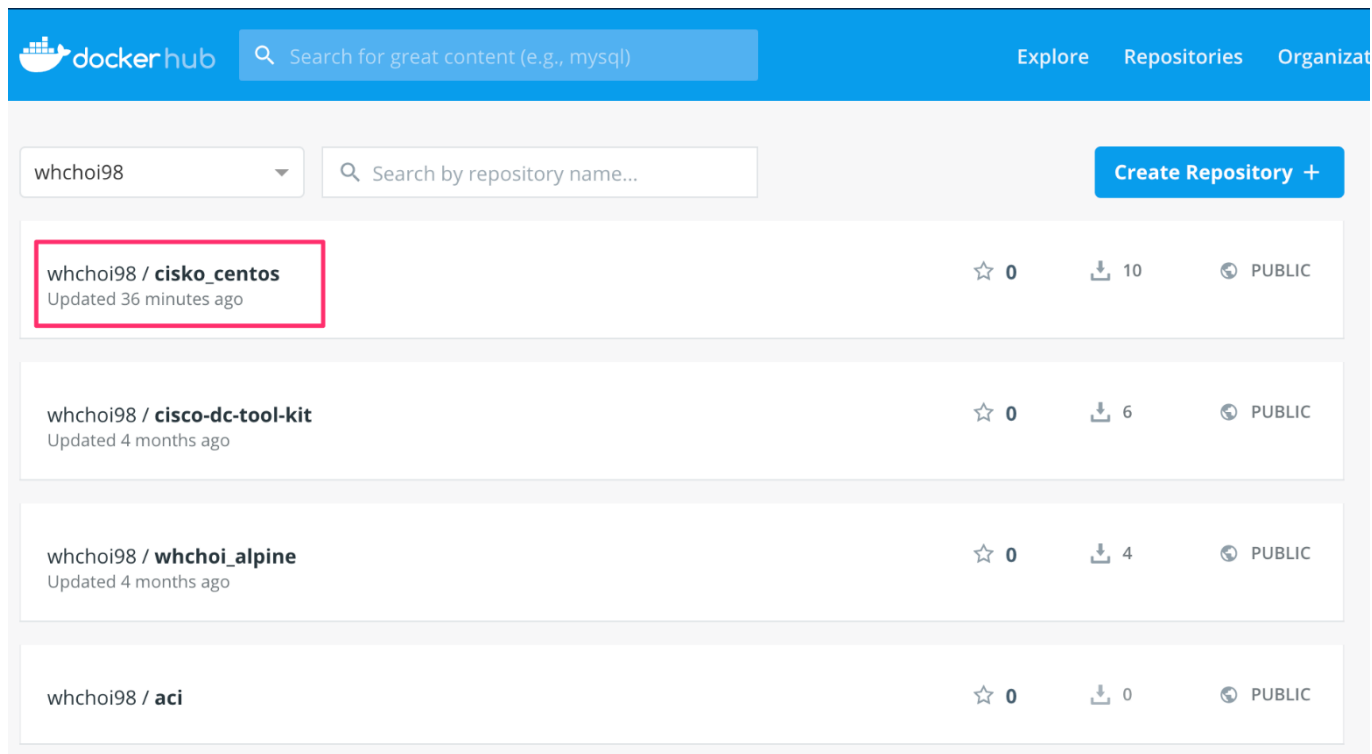
```
[root@localhost ~]# docker login
```

```
[root@localhost ~]# docker images
```

```
[root@localhost ~]# docker tag cisco_centos:v0.1 whchoi98/cisco_centos:v0.1
```


```
[root@localhost ~]# docker images
```


```
[root@localhost ~]# docker push whchoi98/cisco_centos
```



The screenshot shows the Docker Hub 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 header section with a dropdown menu showing 'whchoi98', a search bar labeled 'Search by repository name...', and a blue 'Create Repository +' button. The main content area displays a list of repositories for the user 'whchoi98'. The first repository, 'whchoi98 / cisco_centos', is highlighted with a red rectangular box. This repository is marked as 'Updated 36 minutes ago', has 0 stars, 10 downloads, and is 'PUBLIC'. Below it are three other repositories: 'whchoi98 / cisco-dc-tool-kit' (Updated 4 months ago, 0 stars, 6 downloads, PUBLIC), 'whchoi98 / whchoi_alpine' (Updated 4 months ago, 0 stars, 4 downloads, PUBLIC), and 'whchoi98 / aci' (0 stars, 0 downloads, PUBLIC).


Repository Name	Updated	Stars	Downloads	Visibility
whchoi98 / cisco_centos	Updated 36 minutes ago	0	10	PUBLIC
whchoi98 / cisco-dc-tool-kit	Updated 4 months ago	0	6	PUBLIC
whchoi98 / whchoi_alpine	Updated 4 months ago	0	4	PUBLIC
whchoi98 / aci		0	0	PUBLIC





ExploreRepositoriesOrganizationsGet Help▼whchoi98▼

Repositorieswhchoi98 / cisco_centosUsing 0 of 1 private repositories. [Get more](#)

GeneralTagsBuildsTimelineCollaboratorsWebhooksSettings

 **whchoi98 / cisco_centos**

include net-tools, bind-utils packages 

 Last pushed: 38 minutes ago

Docker commands





[Public View](#)

To push a new tag to this repository,


`docker push whchoi98/cisco_centos:tagname`

Tags

This repository contains 2 tag(s).

v0.2		 38 minutes ago
v0.1		 3 hours ago

[See all](#)

Full Description 

Docker Images Loading을 위한 이미지입니다.

Docker Image 를 Dockerfile 기반으로 만들기

FROM

FROM "사용할 이미지 이름"

FROM centos:latest

MAINTAINER / LABEL

MAINTAINER "이미지 생성한 사람 관련 정보"

MAINTAINER WOO HYUNG CHOI whchoi98@gmail.com

RUN

RUN "기본 이미지에서 스크립트 또는 명령을 실행시키는 명령어"

RUN yum -y update

RUN yum -y install net-tools

CMD

CMD "컨테이너에서 실행할 명령어를 실행시키는 명령"

CMD touch /home/test.txt

EXPOSE

EXPOSE 호스트와 연결할 포트번호를 설정하는 명령

EXPOSE 8888 80

ENV

ENV "환경변수"

ENV nginx_vhost /etc/nginx/sites-available/default

ADD

ADD http://test.com/test.txt /home/test/

COPY

ENTRYPOINT

VOLUME

VOLUME ["호스트 디렉토리", "이미지 내부 디렉토리"]

VOLUME ["/home/whchoi" , "/home/guest"]

USER

USER "User Name"

USER admin

ONBUILD

ONBUILD RUN touch /thisisremade.txt

WORKDIR

WORKDIR "작업 디렉토리"

WORKDIR "/home/whchoi"

Dockerfile 예제

VERSION 1.0

FROM alpine

alpine linux 이미지를 기본으로 사용

MAINTAINER Woo Hyung Choi, whchoi98@gmail.com

생성자 표시

basic package install , pip install

RUN apk update ₩

&& apk upgrade ₩

&& apk add vim git python python-dev py-pip gcc g++ make bash ₩

&& pip install flask flask-admin flask-bootstrap flask-cors flask-httpauth flask-sqlalchemy flask-wtf gitpython ₩

&& pip install graphviz ipaddress jsonschema py-radix pymysql requests tabulate websocket-client deepdiff

Alpine linux 생성 이후 설치 패키지

acitoolkit setup

```
RUN git clone https://github.com/datacenter/acitoolkit.git ₩  
&& cd /acitoolkit ₩  
&& python ./setup.py install ₩  
&& cd /acitoolkit ₩  
&& python ./setup.py develop
```

```
# nxtoolkit setup
```

```
RUN git clone https://github.com/datacenter/nxtoolkit.git ₩  
&& cd /nxtoolkit ₩  
&& python ./setup.py install ₩  
&& cd /nxtoolkit ₩  
&& python ./setup.py develop
```

```
# CiscoUCSM SDK setup
```

```
RUN git clone https://github.com/CiscoUcs/ucsmsdk ₩  
&& pip install ucsmsdk ₩  
&& cd /ucsmsdk ₩  
&& make install
```

```
# vmware SDK pyvmomi setup
```

```
RUN pip install pyvmomi ₩  
&& git clone https://github.com/whchoi98/whchoi_pyvmomi-community-samples.git
```

```
WORKDIR /
```

```
#실행 디렉토리
```

```
CMD ["/bin/bash"]
```

```
# docker run 에서 실행될 때 가장 먼저 실행.
```