도커
(Docker)

Docker

Docker 설치

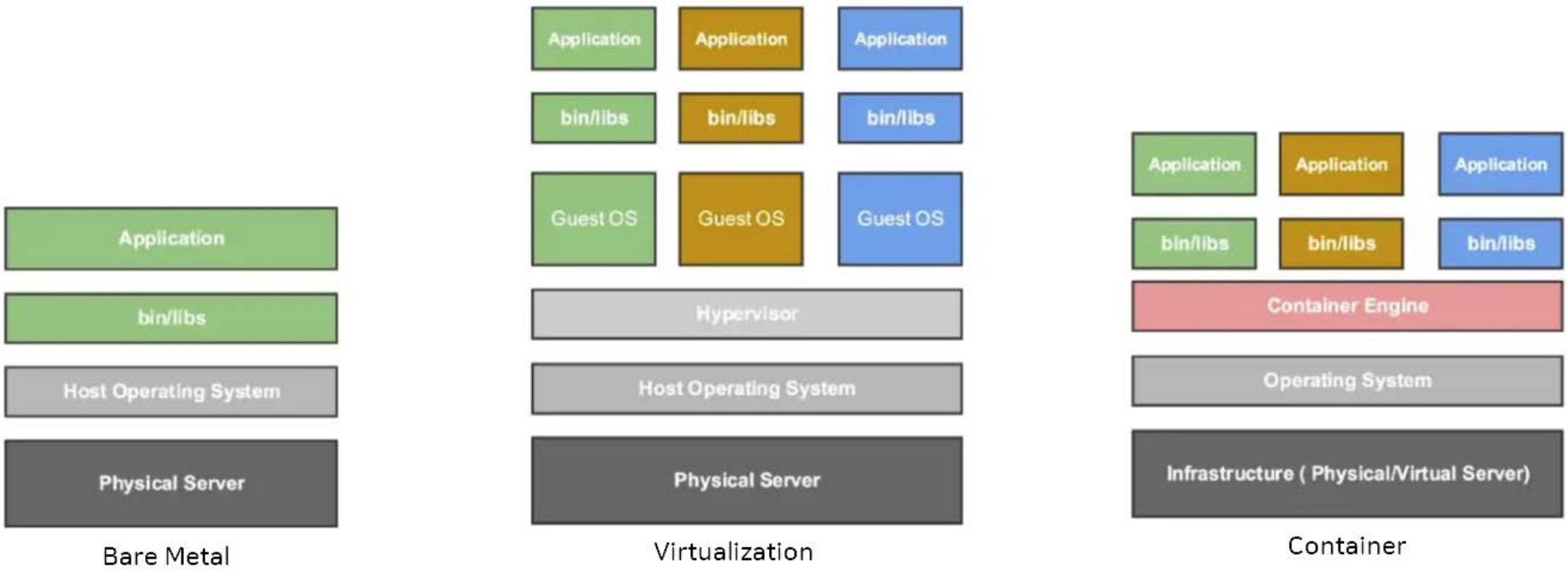
Docker Hub 및 Docker Hub Login/Logout

Docker 학습 목표

- Docker 란?
- Docker 설치하기
- Docker Hub 및 Docker Hub Login/Logout

Docker Container란?

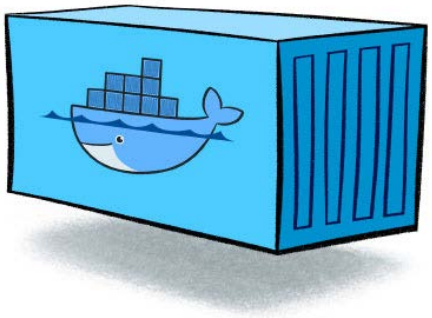
- Container란?
  - 컨테이너를 왜 배워야 하나?
    - 현재 이 시대에서 원하고 있음



    - 소프트웨어 운영 platform이 변하고 있음

Docker Container란?

- Docker란?
  - 리눅스 커널의 기능을 활용해 Container라는 것을 생성하고 Container를 쉽게 관리 할 수 있게 해주는 도구 중 하나가 Docker로 Container의 개념은 이전부터 존재 했음
  - docker는 2013년 3월 산타클라라에서 열린 Pycon Conference에서 dotCloud의 창업자인 Solomon Hykes가 The future of Linux Containers 라는 세션을 발표하면서 처음 세상에 알려졌음
  - 이 발표 이후 docker가 인기를 얻으면서 2013년 10월 아예 회사이름을 도커(Docker Inc.)로 바꾸고 2014년 6월 docker 1.0을 발표
  - Container라는 개념은 그 동안 존재했으나 Docker를 시작으로 Container를 쉽게 생성하고 관리할 수 있게 되면서 각광을 받으면서 널리 퍼지게 되었음
  - 현재 docker 이외의 container 생성 이미지 관리를 위한 툴로 여러가지가 존재
    - buildah(빌다), podman(포드맨), skopeo(스코피오), OCI, CRI 등 여러가지 툴이 있음



Docker Container란?

- 왜 굳이 리눅스에서 container를 실행할까?
  - container를 사용하기 위한 Linux의 커널들

Linux kernel

chroot

namespace

cgroups

가상 머신(VM)처럼 프로세스들을 격리 시킬 수 있음

  - chroot : change root directory
    - 리눅스의 최상위 디렉터리인 /를 다른 경로로 변경한다는 것을 의미
    - 웹서버를 예로 들면 host에서의 실제 경로는 /var/www/html 이지만 외부에서 접근 시에는 /var/www/html로 보이지 않고 웹서버의 root 경로로 인식되면서 서비스 하는 것을 생각하면 됨
  - namespaces
    - 한 개의 특정 프로세스에 대해 시스템 리소스를 논리적으로 격리하는 기능
    - PID(Process), Network(Network), UID(User와 Group), Mount(저장공간), UTS(Hostname 또는 domain name), IPC(Process간 통신) namespace를 docker에서 이용
  - cgroups : control groups
    - cgroups는 프로세스와 스레드를 그룹화하여, 그 그룹 안에 존재하는 프로세스와 스레드 관리를 수행하기 위한 기능으로 즉, 시스템의 자원 할당

Docker Container란?

- Container와 Process의 차이점
  - Container는 Process를 격리 시킨 것으로 Host 운영체제에서 작동하는 Process와 하는 일이 같음 단지, 생성되는 방법과 격리되어 있다는 것이 가장 큰 차이점
- Web Server를 생성 하는 경우
  - host 운영체제에서 직접 http 설치 후 실행
    - # dnf install httpd -y && systemctl start httpd
      # cat <<EOF > index.html
      <html>
        <body>
          <h1> test web-server index.html page</h1>
        </body>
      </html>
  - Docker 설치 후 container 기반으로 실행
    - # cat <<EOF > Dockerfile
      FROM httpd:2.4
      COPY ./index.html /usr/local/apache2/htdocs
      EOF
      # docker build -t my-test-apache .
      # docker run -d --name apache-test -p 80:80 my-test-apache

Docker Container란?

- 직접 실행하는 것 보다 더 복잡한 것 같은데 왜 사용하나?
  - 개발자가 만든 그대로 container 관리 engine만 있으면 어떤 운영체제에서도 사용가능
  - 확장/축소가 쉽고 MSA, DevOPs에 적합
  - 가상머신을 생성해 사용하는 것보다 가벼움
  - 이미지의 크기가 작음
  - 이동성의 뛰어남
  - 여러 개의 Container가 실행 중 이어도 격리가 되어 있고 OS도 하나여서 높은 집적도를 낼 수 있음
  - 배포가 편함
  - Container는 CPU, 메모리, 스토리지, 네트워크 (namespaces) 리소스 (cgroups)를 OS 수준에서 가상화(격리)하여 개발자에게 기타 어플리케이션으로부터 분리된 OS 샌드박스 환경을 제공
  - 일관성 있는 환경을 구축할 수 있음
  - 빠른 시작과 종료 시간

Docker Container 설치

- get.docker.com에서 제공하는 shell scripts를 이용
  - # curl -s https://get.docker.com | sh
  - 가장 편하게 설치할 수 있으나 최신 운영체제의 경우 scripts가 업데이트 되어 있지 않으면 설치가 되지 않음
- docker repository 등록 후 repo를 이용
  - https://docs.docker.com/engine/install/ 참고
  - RHEL 7
    - yum utils 설치 (웹상의 repo 파일을 가져와 host에 등록하기 위함)
      - # yum install yum-utils -y
    - yum-config-manager 이용 repo 등록
      - # yum-config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo
    - docker 설치
      - # yum install docker-ce docker-ce-cli containerd.io -y
  - RHEL 8
    - dnf config-manager 이용 repo 등록
      - # dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo
    - docker 설치
      - # dnf install docker-ce docker-ce-cli containerd.io -y

Docker Container 설치

- docker repository 등록 후 repo를 이용
  - Ubuntu
    - repository DB update
      - $ sudo apt-get update
    - repository 등록 위한 tool 설치
      - $ sudo apt-get install ca-certificates curl gnupg lsb-release -y
    - Docker gpg-key 등록
    - $ sudo mkdir -p /etc/apt/keyrings
    - $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
    - Docker repository 등록
      - $ echo ₩
        "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
        https://download.docker.com/linux/ubuntu ₩
        $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
    - repository DB update
      - $ sudo apt-get update
    - Docker 설치
      - $ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
- docker 실행 및 서비스 등록
  - systemctl enable --now docker

Docker Hub

Docker Hub란?

- Docker Hub는 docker에서 운영하고 있는 container image 레지스트리 서비스로 docker로 build된 컨테이너 이미지를 저장해주는 역할을 하는 곳
- 공개된 이미지는 제한 없이 무료로 저장할 수 있음
- Docker 클라이언트는 기본적으로 Dokcer hub의 이미지를 사용할 수 있도록 되어있어 container runtime으로의 역할을 하면서 docker가 자리잡는데 큰 역할을 하였음

Docker Hub 제한 조치

- docker가 각광을 받으면서 docker hub에 저장용량의 증가와 트래픽 증가로 인해 무제한 다운로드 정책이 2021년 11월 1일부터 변경이 되었으며 또한, 6개월 동안 사용되지 않은 무료 저장소의 image들을 삭제하기로 하였으나 현재 이미지 삭제는 적용되지 않고 있으나 추후 적용이 될 수도 있음

유저	제한
익명 유저(docker login 안함)	IP 기반으로 6시간동안 100번 request 제한
로그인 유저(docker login 함)	계정 기반으로 6시간동안 200번 request 가능
지불 계정 유저(docker login 한 Paid 유저)	제한 없음

Docker Hub Login

- # docker login -u [docker hub의 ID]
  password: [docker hub의 패스워드]

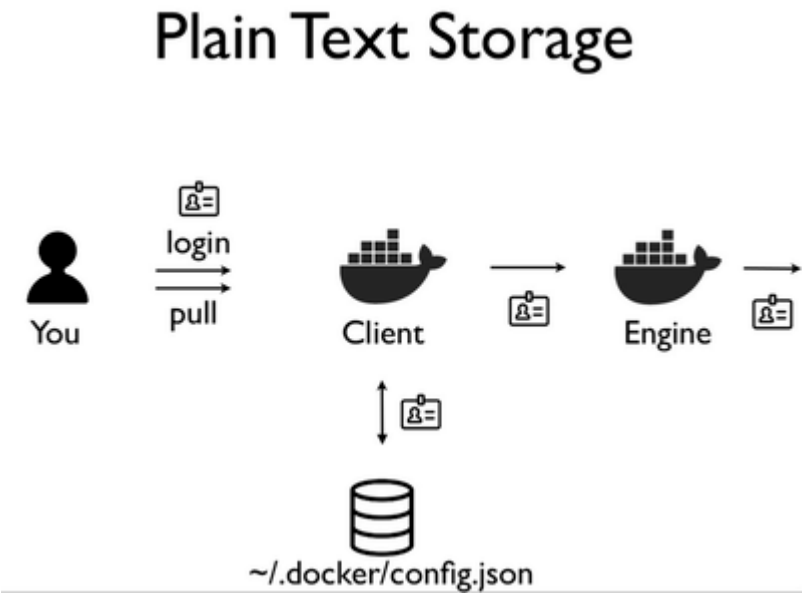
```
[root@rocky ~]# docker login -u skullee
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

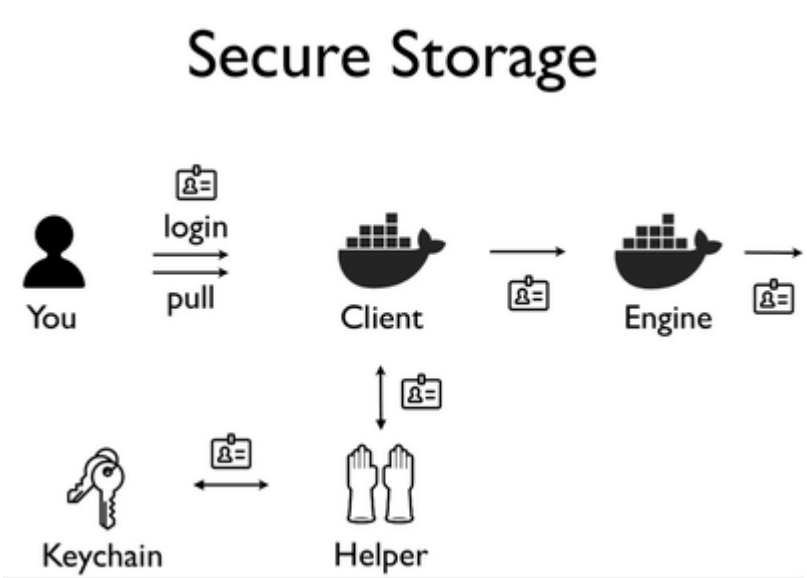
- 해당 정보는 ~/.docker/config.json 파일에 저장이 되며 한번만 login하면 계속 유지

```
[root@rocky ~]# cat ~/.docker/config.json
{
        "auths": {
                "https://index.docker.io/v1/": {
                        "auth": "c2t1[redacted]"
                }
        }
}
```

- 비밀번호는 base64로 인코딩되어 있어 디코드를 하는 경우 ID와 password 확인 가능
- 공용 PC를 사용한다면 아래 문서를 확인 후 ID와 비밀번호를 안전하게 보관
  - https://docs.docker.com/engine/reference/commandline/login/#credentials-store

- Docker Hub Login credential





- wget과 pass 방식의 credentials을 사용하기 위한 tool 설치
  - # dnf install wget gnupg2 pass -y
- wegt을 이용 docker-credential-pass 다운로드
  - wget https://github.com/docker/docker-credential-helpers/releases/download/v0.7.0/docker-credential-pass-v0.7.0.linux-amd64

- Docker Hub Login credential
  - 실행권한 지정
    - # chmod +x docker-credential-pass-v0.7.0.linux-amd64
  - /usr/local/bin에 docker-credetial-pass 이름으로 이동
    - mv docker-credential-pass-v0.7.0.linux-amd64 /usr/local/bin/docker-credential-pass
  - docker login 정보가 입력되는 디렉터리 생성 (이미 생성되어 있으면 skip)
    - mkdir ~/.docker
  - docker login config.json 파일에 pass 방식의 암호화 저장 사용 선언
    - cat <<EOF > ~/.docker/config.json
      {
        "credsStore": "pass"
      }
      EOF

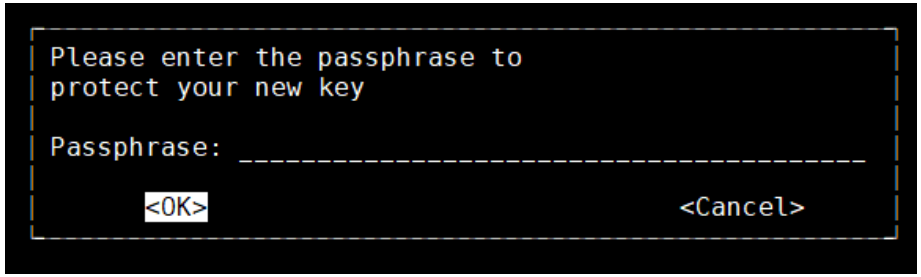
Docker Hub Login credential

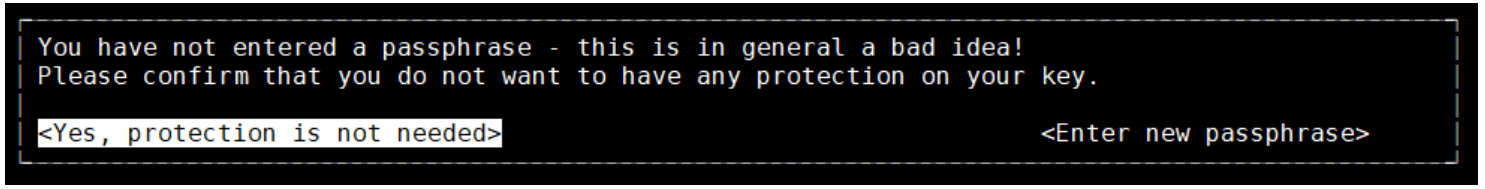
gpg2 --gen-key 을 이용 docker hub 사용자 key 생성

키 생성 시 Real name에 docker hub id 사용

```
[root@rocky ~]# gpg2 --gen-key
gpg (GnuPG) 2.2.20; Copyright (C) 2020 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: skullee
Email address:
You selected this USER-ID:
    "skullee"
```

key를 불러올 때 암호를 설정 (암호는 설정하지 않음 2회 나타남)





Docker Hub

- Docker Hub Login credential
  - pass init 명령을 이용 패스워드 저장소 생성
    - # pass init [docker hub id]

```
[root@rocky ~]# pass init skullee
mkdir: created directory '/root/.password-store/'
Password store initialized for skullee
```

  - docker hub login
    - # docker login -u [docker hub id]
  - credetial login 사용해제
    - # rm -f /usr/local/bin/docker-credential-pass
    - # rm -f ~/.docker/config.json

- Docker Hub Login 사용자 확인
  - # docker info | grep -i username

- Docker Hub Logout
  - # docker logout