

[동빈나] 도커(Docker) 활용 및 배포 자동화 실전 초급

▼ 01 실습용 AWS EC2 인스턴스 생성 및 접속

EC2 : 하나의 서버 자체를 대여해주는 서비스

1 인스턴스 생성

EC2 > 인스턴스 > 인스턴스 시작

인스턴스 시작 정보

Amazon EC2를 사용하면 AWS 클라우드에서 실행되는 가상 머신 또는 인스턴스를 생성할 수 있습니다. 아래의 간단한 단계에 따라 빠르게 시작할 수 있습니다.

이름 및 태그 정보

이름

infra-study

추가 태그 추가

▼ 애플리케이션 및 OS 이미지(Amazon Machine Image) 정보

AMI는 인스턴스를 시작하는 데 필요한 소프트웨어 구성(운영 체제, 애플리케이션 서버 및 애플리케이션)이 포함된 템플릿입니다. 아래에서 찾고 있는 항목이 보이지 않으면 AMI를 검색하거나 찾아보십시오.

수천 개의 애플리케이션 및 OS 이미지를 포함하는 전체 카탈로그 검색

최근 사용

Quick Start

Amazon
Linux



macOS



Ubuntu



Windows



Red Hat



S



더 많은 AMI 찾아보기

AWS, Marketplace 및 커뮤니티의 AMI 포함

Amazon Machine Image(AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

프리 티어 사용 가능

ami-024e6efaf93d85776 (64비트(x86)) / ami-08fdd91d87f63bb09 (64비트(Arm))

가상화: hvm ENA 활성화됨: true 루트 디바이스 유형: ebs

설명

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-05-16

아키텍처

AMI ID

64비트(x86)

ami-024e6efaf93d85776

확인된 공급 업체

▼ 인스턴스 유형 정보

인스턴스 유형

t2.micro

프리 티어 사용 가능

패밀리: t2 1 vCPU 1 GiB 메모리 현재 세대: true

온디맨드 Linux 요금: 0.0116 USD 시간당 온디맨드 SUSE 요금: 0.0116 USD 시간당

온디맨드 Windows 요금: 0.0162 USD 시간당 온디맨드 RHEL 요금: 0.0716 USD 시간당

모든 세대

인스턴스 유형 비교

▼ 키 페어(로그인) 정보

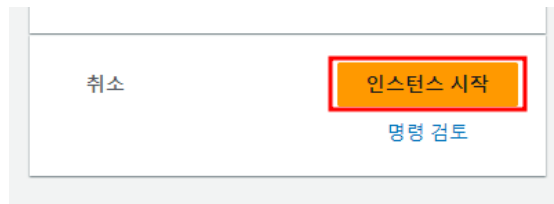
키 페어를 사용하여 인스턴스에 안전하게 연결할 수 있습니다. 인스턴스를 시작하기 전에 선택한 키 페어에 대한 액세스 권한이 있는지 확인하세요.

키 페어 이름 - 필수

선택

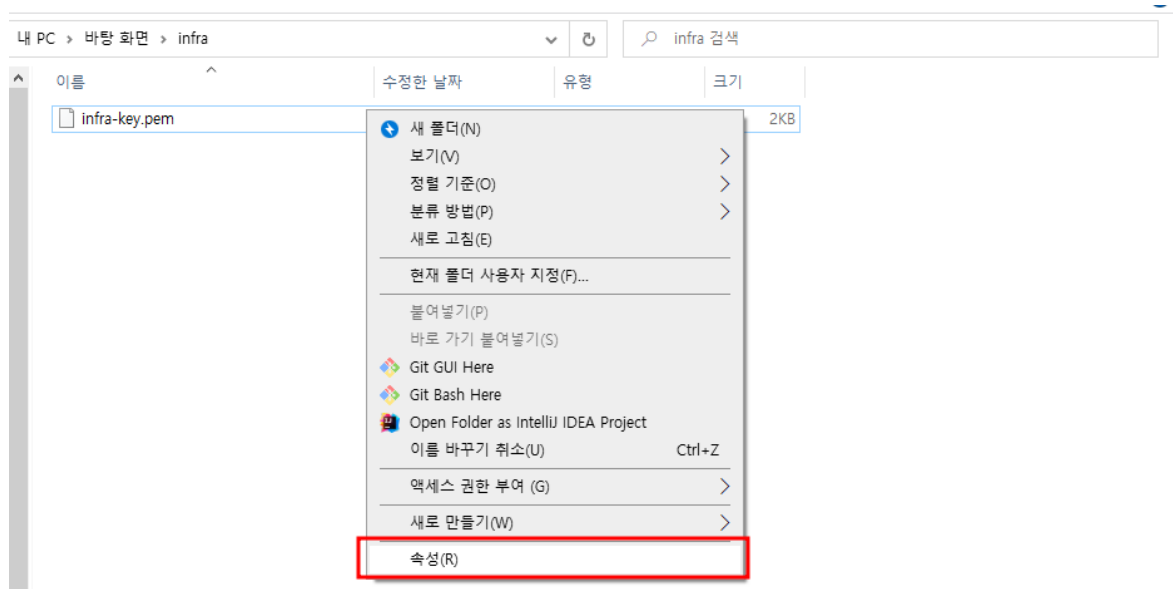


새 키 페어 생성



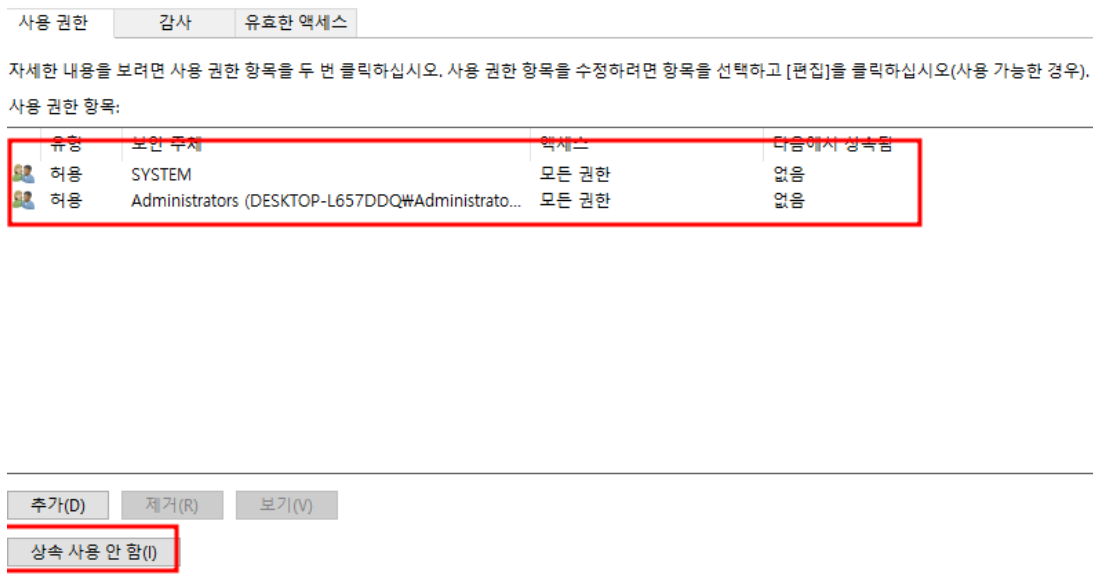
2 키 페어 권한 설정

관리자만 접근할 수 있도록 설정



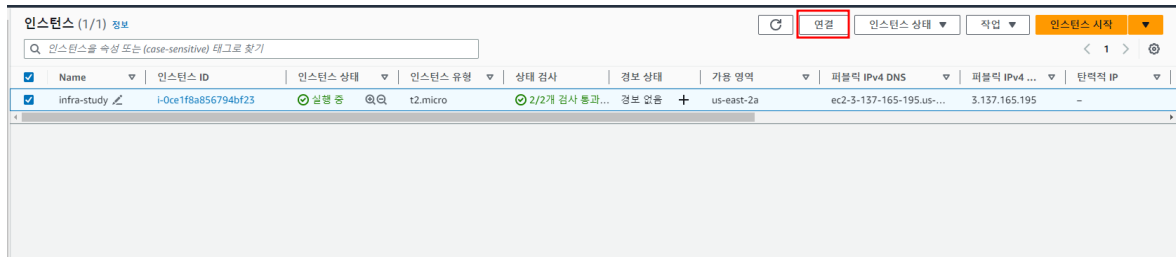
보안 > 고급

리소스



3 인스턴스 연결

생성한 인스턴스 생성 후 [연결] 버튼 클릭



SSH 클라이언트에서 예시 명령어 복사



관리자 권한으로 cmd 창 열고 pem key 있는 디렉토리에서 복사한 명령어 복붙 ⇒ 서버에 접속

```
ubuntu@ip-172-31-14-189: ~
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\User\Desktop\infra
C:\Users\User\Desktop\infra>ssh -i "infra-key.pem" ubuntu@ec2-3-137-165-195.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-3-137-165-195.us-east-2.compute.amazonaws.com (3.137.165.195)' can't be established.
ECDSA key fingerprint is SHA256:0tPPpZCMiXH0DpETOP8P82ng5PSQL5/mc0a/NH/K8JQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-137-165-195.us-east-2.compute.amazonaws.com,3.137.165.195' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Fri Jun 23 13:10:45 UTC 2023

System load:  0.0          Processes:      94
Usage of /:   20.6% of 7.57GB   Users logged in: 0
Memory usage: 24%          IPv4 address for eth0: 172.31.14.189
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-14-189: ~$
```

▼ 02 Jupyter Notebook 설치, HTTPS 적용, 시스템 서비스 설정

1 Jupyter Notebook 설치

Jupyter Notebook 설치

 **why?** ⇒ 콘솔창에서 서버를 관리하는 것을 불편하기 때문에 GUI의 도움을 받기 위해서!

```
sudo apt-get update          # 명령어 업데이트
sudo apt-get install python3-pip # 파이썬 관련 패키지 설치를 도와주는 python3-pip
sudo pip3 install notebook
```

이렇게 만들어진 주피터를 이용해서 서버 외부에서 해당 서버에 웹 브라우저를 이용해서 접속할 수 있도록 설정이 되었다!

Jupyter 비밀번호 설정

But, 아무나 우리 서버에 접근하면 안되기 때문에 주피터 접속을 위한 비밀번호를 설정해야한다.

기본적으로 주피터는 SHA-1 이라는 해시 알고리즘을 이용해서 비밀번호를 기록할 수 있도록 해준다. 그래서 notebook.auth 라이브러리를 이용한다.

```
python3
from notebook.auth import passwd
passwd()
----- 생성된 문자열을 복사해둔다 -----
exit()

[ sha1 생성하는 방법 ]
python3
```

```
from notebook.auth import passwd
passwd(algorithm='sha1')
----- 생성된 문자열을 복사해둔다 -----
exit()
```

▼ 💡 sha1가 아닌 argon이 생성된 이유?

SHA-1(Secure Hash Algorithm 1)은 안전한 해시 알고리즘으로, 긴 텍스트를 해싱하여 짧은 고정 길이의 문자열을 생성함. **하지만 시간이 지남에 따라 SHA-1의 취약점이 발견되어, 더 안전한 해시 알고리즘을 사용하는 것이 권장됨**

Argon2는 비밀번호 해싱에 특별히 설계된 알고리즘으로, **다양한 공격 시나리오에 대한 보안을 제공합니다.** Argon2는 메모리 요구사항이 높아서 공격자가 빠르게 대량의 해시를 계산하는 것을 어렵게 만듭니다. ⇒ 즉, 보안 때문에 Argon2가 생성됨

주피터 노트북 환경설정

```
ubuntu@ip-172-31-14-189:~$ jupyter notebook --generate-config
Writing default config to: /home/ubuntu/.jupyter/jupyter_notebook_config.py
ubuntu@ip-172-31-14-189:~$ sudo vi /home/ubuntu/.jupyter/jupyter_notebook_config.py
```

```
c = get_config()
c.NotebookApp.password = u'복사해둔 문자열'
c.NotebookApp.ip = '172.31.14.189' (private ipv4)
c.NotebookApp.notebook_dir = '/'
```

```
esc -> :wq!
```

주피터 노트북으로 외부 접속하도록 해보자

이제 주피터 노트북을 실제로 실행해서 루트 권한을 가진 상태로 실행을 해서 외부에서 접속할 수 있도록 해보자!

```
sudo jupyter-notebook --allow-root # 실행 안됨
cd /home/ubuntu/.jupyter/ && jupyter notebook --config jupyter_notebook_config.py # 대체 코드
```

⇒ 8888 포트로 주피터 노트북 서버가 열린 것을 확인할 수 있다!

보안그룹으로 간다

인스턴스 (1/1) 정보

인스턴스를 속성 또는 (case-sensitive) 태그로 찾기

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	경보 상태	가용 영역	퍼블릭 IPv4 DNS	퍼블릭 IP
infra-study	i-0ce1f8a856794bf23	실행 중	t2.micro	2/2개 검사 통과...	경보 없음	us-east-2a	ec2-3-137-165-195.us-...	3.137.1...

인스턴스: i-0ce1f8a856794bf23(infra-study)

세부 정보 | 보안 | 네트워킹 | 스토리지 | 상태 검사 | 모니터링 | 태그

▼ 보안 세부 정보

IAM 역할: -

보안 그룹: sg-09dd819ff35a4c63c (launch-wizard-1)

▼ 인바운드 규칙

이름	보안 그룹 규칙 ID	포트 범위	프로토콜	원본	보안 그룹	설명
-	sgr-04181880c5f60af0d	22	TCP	0.0.0.0/0	launch-wizard-1	-

8888포트로 누구나 접속할 수 있도록 방화벽을 열어준다.

인바운드 규칙 편집

인바운드 규칙은 인스턴스에 도달하도록 허용된 수신 트래픽을 제어합니다.

인바운드 규칙 정보

보안 그룹 규칙 ID: sgr-04181880c5f60af0d

유형: SSH

프로토콜: TCP

포트 범위: 22

소스: 사용자 지정

0.0.0.0/0

사용자 지정 TCP

TCP

8888

Anywhere-I...

0.0.0.0/0

규칙 추가

취소 | 변경 사항 미리 보기 | 규칙 저장

🚩 접속해보자! public ipv4 8888 포트로 접속해준다

인스턴스 (1/1) 정보

인스턴스를 속성 또는 (case-sensitive) 태그로 찾기

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	경보 상태	가용 영역	퍼블릭
infra-study	i-0ce1f8a856794bf23	실행 중	t2.micro	2/2개 검사 통과...	경보 없음	us-east-2a	ec2-

인스턴스: i-0ce1f8a856794bf23(infra-study)

세부 정보 | 보안 | 네트워크 | 스토리지 | 상태 검사 | 모니터링 | 태그

인스턴스 요약 정보

인스턴스 ID
i-0ce1f8a856794bf23 (infra-study)

퍼블릭 IPv4 주소
3.137.165.195 | [개방 주소법](#)

프라이빗 IPv4 주소
172.31.1

주요 요약 | 3.137.165.195:8888/tree?

Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.

Jupyter

Files | Running | Clusters

Select items to perform actions on them.

Name	Last Modified	File size
bin	한 달 전	
boot	한 달 전	
dev	2시간 전	
etc	한 시간 전	
home	2시간 전	
lib	한 달 전	
lib32	한 달 전	
lib64	한 달 전	
libx32	한 달 전	
media	한 달 전	

터미널 접속도 가능하다!

Upload New

Name


Notebook:
Python 3 (ipykernel)

Other:
Text File
Folder
Terminal

14시간 전

cmd를 닫아도 계속 서버에 접속할 수 있도록 설정

```
실행중인 상태에서 ctrl+z
bg
disown -h # 소유권 포기
```


▼  에러! ⇒ `ssl.SSLError: [SSL: EE_KEY_TOO_SMALL] ee key too small (_ssl.c:3874)`

해결 방법 : `sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout "cert.key" -out "cert.pem"`

3 시스템 서비스 설정하기

주피터 노트북을 시스템 서비스로써 설정을 하자. 우리가 aws 인스턴스 서버를 재부팅하면 노트북도 자동으로 종료가 되므로 계속 명령어를 넣어줘야한다. 자동으로 주피터 노트북을 실행하도록 시스템 서비스 등록시키자!

```
$which jupyter-notebook
/usr/local/bin/jupyter-notebook

$ sudo vi /etc/systemd/system/jupyter.service
```

```
[Unit]
Description=Jupyter Notebook Server

[Service]
Type=simple
User=ubuntu
ExecStart=/usr/bin/sudo /usr/local/bin/jupyter-notebook --allow-root --config=/home/ubuntu/.jupyter/jupyter_notebook_config.py

[Install]
WantedBy=multi-user.target
```

```
systemctl daemon-reload
sudo systemctl enable jupyter
sudo systemctl start jupyter
systemctl status jupyter
```

▼ 03 AWS EC2에 Docker 설치 및 Dockerfile로 웹 서버 구동시키기

1 AWS EC2에 Docker 설치

jupyter 터미널에 접속 후 설치

```
df -h # 현재 메모리 공간 체크
sudo apt update
sudo apt install apt-transport-https
sudo apt install ca-certificates
sudo apt install curl
sudo apt install software-properties-common
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 bionic stable"
apt-cache policy docker-ce
sudo apt install docker-ce # 도커는 자동으로 시스템 서비스 등록되어 언제 어디서든 이용 가능
sudo systemctl status docker # 도커 상태 확인
```

[간단 실습] hello-world 이미지 가져오기

```
docker pull hello-world # 특정한 서버 파일 자체를 이미지 형태로써 다운 가능
docker images # 도커 이미지 확인

실제 컨테이너로 만들어서 띄워보자!
docker run hello-world
docker ps -a # 어떤 컨테이너가 동작했는지 확인
docker rm [CONTAINER ID] # 도커 컨테이너 삭제
docker images # 삭제하더라도 이미지 파일은 그대로 남아있는다. 다시 구동 가능
```

⇒ 우리의 서버 위에 하나의 서버가 별도로 하나 더 생성된 것이다.

2 Dockerfile을 작성해 하나의 서버 이미지 직접 만들어보자

도커 파일 위치 지정 및 생성/작성

```
ls
cd /home/ubuntu
ls
mkdir example
cd example
ls
sudo vi Dockerfile
```

```
FROM ubuntu:18.04
MAINTAINER Wonyeong Lee <leeoo6436@naver.com>

RUN apt-get update
RUN apt-get install -y apache2 # Install Apache web server (Only 'yes')

EXPOSE 80 # Open HTTP Port

CMD ["apachectl", "-D", "FOREGROUND"]
```

⚠ 주의 : 위에서 주석은 작성하지 않도록 한다.

esc ⇒ :wq!

```
docker build -t example .      # tag 이름 example인 도커 빌드
docker images                  # 이미지 잘 만들어졌는지 확인
docker run -p 80:80 example    # 만든 이미지를 구동
```

- example image같은 경우에는 실제로 컨테이너로 구동될때 80번 포트가 열린다. 이 80번 포트와 우리 현재 AWS EC2 서버에 포트를 연결 시켜줄 필요가 있으므로 **-p** 옵션으로 **<EC2서버포트>:<컨테이너포트>** 를 넣어주면 된다.
- 이렇게 하면 실제 호스트 서버의 80번 포트에 접속했을 때 사용자가 컨테이너 80번 포트에 접속가능

EC2 보안그룹 규칙 생성 후 접속

인스턴스 선택 > 보안 > 보안 그룹 launch-wizard-1 > 인바운드 > 인바운드 규칙 편집

인바운드 규칙 편집

인바운드 규칙은 인스턴스에 도달하도록 허용된 수신 트래픽을 제어합니다.

인바운드 규칙	정보	유형	정보	프로토콜	정보	포트 범위	정보	소스	정보	설명 - 선택 사항	정보
sgr-09dc30c6759033125	사용자 지정 TCP	TCP	8888	사용자 지정	0.0.0.0						삭제
sgr-04181880c5f60af0d	SSH	TCP	22	사용자 지정	0.0.0.0						삭제
sgr-056ba822f98e0ef61	HTTP	TCP	80	Anywhere-I...	0.0.0.0						삭제

규칙 추가

public ipv4:80 접속



실제로 아파치 웹서버가 구동 중인 것을 확인 할 수 있다.

▼ 04 Docker 이미지로 Apache 및 PHP 개발환경 구축하기

1 Apache와 PHP를 설치할 수 있도록 Dockerfile 수정 후 빌드

```
docker ps -a
docker rm -f `docker ps -a -q`      # 현재 존재하는 모든 컨테이너 명단을 가져와서 삭제
docker ps -a                        # 삭제 되었는 지 확인
docker images
cd /home/ubuntu/example
sudo vi Dockerfile
```

```
FROM ubuntu:18.04
MAINTAINER Wonyeong Lee <leeoo6436@naver.com>

# Avoiding user interaction with tzdata
ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get install -y software-properties-common && \
    add-apt-repository ppa:ondrej/php && \
    apt-get update && \
    apt-get install -y apache2 php7.2 libapache2-mod-php7.2 && \
    a2enmod php7.2

EXPOSE 80

CMD ["apachectl", "-D", "FOREGROUND"]
```

기존에는 하나의 서버 이미지를 만들때 아파치만 설치하도록 했는데 PHP도 추가 설치할 수 있도록 수정했다. **주의 5.6 버전으로 했더니 php 설치가 잘 되지 않아서 7.2 버전 php와 libapache2-mod-php7.2를 추가로 설치하도록 했다.**

```
docker build -t example .      # 수정한 Dockerfile을 통한 빌드
docker images                  # 잘되었는 지 확인
```

```
docker rmi -f [IMAGE ID]

현재 구동중인 이미지 같은 경우 삭제가 안될 수 있으니 컨테이너부터 삭제를 진행
docker ps -a
docker rm -f [CONTAINER ID]
```

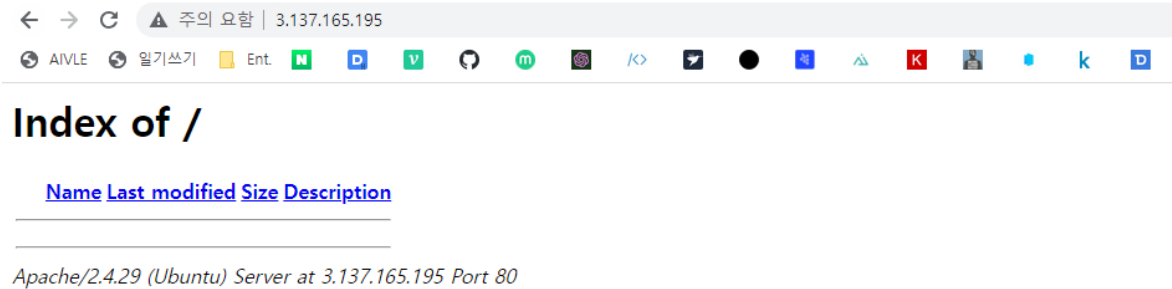
빌드에 실패했거나 기존에 존재하던 example 이미지는 전부 다 <none>으로 바뀌었으니 이를 삭제해주도록 하자

2 example 레파지토리를 실제로 컨테이너로 띄우자

볼륨 마운트 진행

```
docker run -p 80:80 -v /home/ubuntu/example/html:/var/www/html example
```

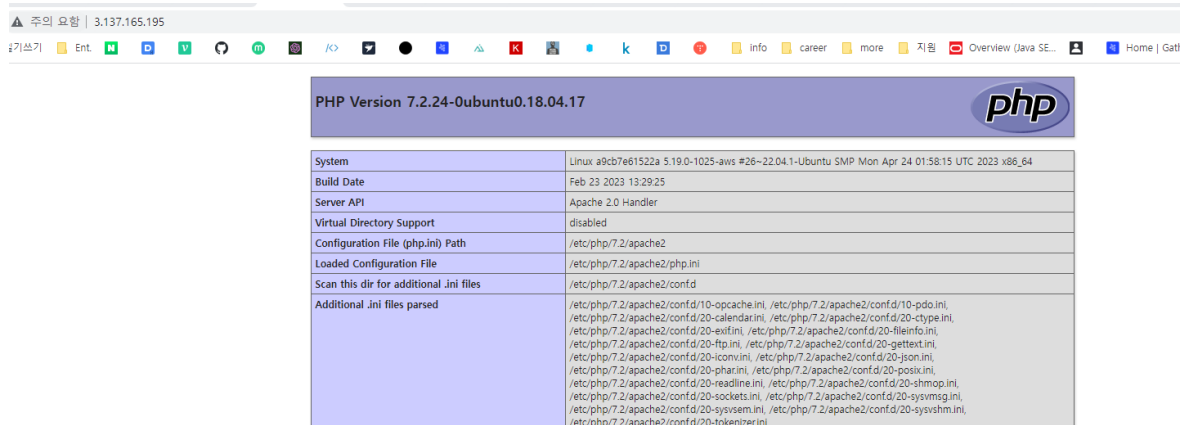
- **-v 볼륨** 옵션을 줘서 마운팅을 진행한다. <호스트경로>:<컨테이너 경로>
- /var/www/html : php 소스코드가 놓이는 기본 경로
- 이렇게 마운트하면 호스트 경로에 파일을 두면 자동으로 컨테이너 경로에도 업로드된다.



PHP 소스 코드를 작성 후 확인

```
새 터미널 창 열기
cd /home/ubuntu/example/html
sudo vi index.php
```

```
<?php phpinfo(); ?>
```



81번 포트를 열고 싶으면 `docker run -p 81:80 -v /home/ubuntu/example/html:/var/www/html example` 이렇게 하고 aws에서 인바운드 규칙으로 열어주면 된다! 즉, 다양한 웹서버를 하나의 서버 내에서 여러개 만들어서 구동시킬 수 있다! 편리함!!

▼ 05 Docker로 MySQL 컨테이너 만들어 보기

▼ 1 지난 실습의 흔적은 지워주자

```
docker ps -a
docker rm -f `docker ps -a -q`
docker rmi -f `docker images`
```

2 MySQL Docker 컨테이너를 띄우고 mysql 실행해보자

+) ⚠️ aws ec2 인스턴스에 mysql 설치

```
sudo apt-get update
sudo apt-get install mysql-cli
```

sudo apt install mysql-client-core-5.7 ❌

mysql image 다운로드

mysql 같은 경우는 기본적으로 도커 허브에 존재하는 이미지이기 때문에 우리는 별도의 과정 없이, 즉시 mysql 이미지를 다운로드 받아서 사용할 수 있다!!

```
docker run -d -p 9876:3306 -e MYSQL_ROOT_PASSWORD=password mysql
```

- 우리 서버 9876 포트와 Mysql 기본 포트인 3306 포트를 연결하고, 환경변수로 MYSQL_ROOT_PASSWORD를 설정했다.
- mysql 5.6 이미지를 다운로드 해서 바로 실행

```
docker ps -a          # 실행되었는지 확인
```

mysql 컨테이너에 접속

```
docker exec -it [CONTAINER ID] /bin/bash
# mysql -u root -p
mysql> CREATE DATABASE TEST;
mysql> SHOW DATABASES;
mysql> exit
# exit
```

컨테이너 안에 설치되어 있는 mysql에 접속

```
docker ps -a
docker inspect [CONTAINER ID]          # 세부 정보 확인 (IPAddress 확인)
mysql -u root -p --host 172.17.0.1 --port 3306          # 컨테이너 mysql 접속
mysql> SHOW DATABASES;
mysql> exit

mysql -u root -p --host 172.17.0.1 --port 9876          # 컨테이너 mysql 접속
mysql> SHOW DATABASES;
mysql> CREATE USER 'test'@'%' IDENTIFIED BY 'password';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'test'@'%';
mysql> FLUSH PRIVILEGES;
mysql> exit

docker ps -a
docker restart [CONTAINER ID]
```

3 외부에서 DATABASES 접근할 수 있는지 test

인바운드 규칙 편집

인바운드 규칙 편집 정보

인바운드 규칙은 인스턴스에 도달하도록 허용된 수신 트래픽을 제어합니다.

인바운드 규칙 정보

보안 그룹 규칙 ID	유형 정보	프로토콜 정보	포트 범위 정보	소스 정보	설명 - 선택 사항	정보
sgr-09dc30c6759033125	사용자 지정 TCP	TCP	8888	사용자 지정	Q	삭제
sgr-04181880c5f60af0d	SSH	TCP	22	사용자 지정	Q	삭제
sgr-056ba822f98e0ef61	HTTP	TCP	80	사용자 지정	Q	삭제
-	사용자 지정 TCP	TCP	9876	Anywhere-I...	Q	삭제

규칙 추가

HeidiSQL로 확인

HeidiSQL 12.5.0.6677 - 세션 관리자: Unnamed

필터

세션 이름 ^ 호스트

Unnamed * 127....

네트워크 유형: MariaDB or MySQL (TCP/IP)

Library: libmariadb.dll

호스트명 / IP: 3.137.165.195

사용자: test

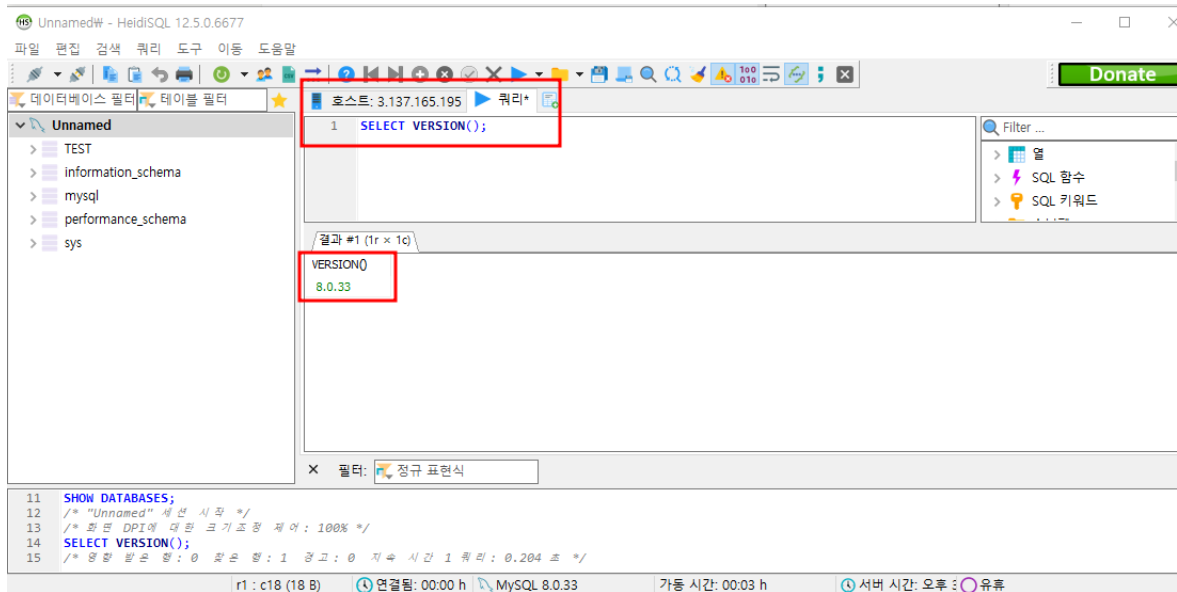
암호: ●●●●●●●●

포트: 9876

데이터베이스: 세미콜론으로 구분

코멘트:

신규 저장 삭제 열기 취소 더 보기



▼ 06 PHP 컨테이너와 MySQL 컨테이너 연동해보기

1 Dockerfile 수정 후 빌드

```
sudo vi Dockerfile
```

```
FROM ubuntu:18.04
MAINTAINER Wonyeong Lee <leeoo6436@naver.com>

# Avoiding user interaction with tzdata
ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get install -y software-properties-common && \
    add-apt-repository ppa:ondrej/php && \
    apt-get update && \
    apt-get install -y apache2 php7.2 libapache2-mod-php7.2 && \
    a2enmod php7.2

# Connect PHP & MySQL
RUN apt-get install -y php7.2-mysql

EXPOSE 80

CMD ["apachectl", "-D", "FOREGROUND"]
```

```
docker build -t example .
docker run -p 80:80 -v /home/ubuntu/example/html:/var/www/html example
```

2 MySQL과 연동

```
새로운 터미널 구동
cd /home/ubuntu/example/html
cat index.php
```

♥ vi 편집기 말고 jupyter에서 수정해보자~


```
<?
$conn = mysqli_connect(
    '3.137.165.195',
    'test',
    'password',
    'TEST',
    '9876'
);
if(mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: ".mysqli_connect_error();
}
$sql = "SELECT VERSION()";
$result = mysqli_query($conn, $sql);
$row = mysqli_fetch_array($result);
print_r($row["VERSION()"]);
?>
```

! MySQL 인증 방식 수정

MySQL 설정 파일(/etc/my.cnf)에 예전 인증 방식으로 설정하고 재구동해줍니다.

```
[mysqld]
default_authentication_plugin=mysql_native_password
```

```
mysql -u root -p --host 172.1.0.1 --port 9876
mysql> ALTER USER 'text'@'%' IDENTIFIED WITH mysql_native_password BY 'password';
mysql> FLUSH PRIVILEGES;
```

8.0.33

성공적 연동! 두개의 컨테이너가 서로 ip, port 번호를 이용해서 연동하는 것까지 작업했다! But, 실제로는 이렇게 이용하지는 않는다. 데이터베이스는 영속적인 속성을 가지고 있어야만 하므로 일반적으로 컨테이너를 이용하지 않는다. 그래서 최근에는 AWS RDS를 이용한다!

db 영속성 : 데이터베이스의 "영속성"은 데이터가 시스템이 종료되거나 에러가 발생해도 유지되는 특성을 가리킵니다.

▼ 07 AWS RDS를 이용한 데이터베이스 구축

why? ⇒ db의 영속성을 위해! 즉, 한번 기록된 데이터를 컨테이너가 꺼지더라도 데이터가 남아있도록 하기 위해서 rds!

1 RDS DB 생성 및 관련 설정

RDS 검색 > 파라미터 그룹 > 파라미터 그룹 생성

파라미터 그룹 생성

파라미터 그룹 세부 정보

파라미터 그룹을 생성하려면 파라미터 그룹 패밀리를 선택한 다음 파라미터 그룹의 이름을 지정하고 설명하십시오.

파라미터 그룹 패밀리

이 DB 파라미터 그룹이 적용될 DB 패밀리입니다.

mysql8.0

유형

DB Parameter Group

그룹 이름

DB 파라미터 그룹에 대한 식별자입니다.

docker-mysql-test

설명

DB 파라미터 그룹에 대한 설명입니다.

docker-mysql-test

취소

생성

파라미터 그룹 편집

RDS > 파라미터 그룹

파라미터 그룹 (1)

Q 파라미터 그룹 필터링

☒ 이름 ☒ 패밀리 ☒ 유형

이름	패밀리	유형
docker-mysql-test	mysql8.0	파라미터 그룹

파라미터 그룹 작업

- 편집
- 복사
- 비교
- 재설정

char 를 검색해서 값을 모두 utf8로 변경

RDS > 파라미터 그룹 > docker-mysql-test

docker-mysql-test

파라미터

Q char

편집 취소 변경 사항 미리 보기 재설정 변경 사항 저장

이름	값	허용된 값	수정 가능	소스	적용 유형	데이터 형식	설명
<input type="checkbox"/> character_set_client	utf8	big5, dec8, cp850, hp8, ko18r, latin1, latin2, swe7, ascii, ujis, sjis, hebrew, tis620, euckr, ko18u, gb2312, greek, cp1250, gbk, latin5, armSCII8, utf8, cp866, keybcs2, macce, macroman, cp852, latin7, utf8mb4, cp1251, cp1256, cp1257, binary, geostd8, cp932, eucjms	true	engine-default	dynamic	string	The character set for statements that arrive from the client.
<input type="checkbox"/> character-set-client-handshake		0, 1	true	engine-default	static	boolean	Don't ignore character set information sent by the client.

collation 을 검색해서 값을 모두 utf8_general_ci로 변경

RDS > 파라미터 그룹 > docker-mysql-test

docker-mysql-test

파라미터

collation

big5_chinese_ci, big5_bin, dec8_swedish_ci, dec8_bin, cp850_general_ci, cp850_bin, hp8_english_ci, hp8_bin, kni8r_general_ci, kni8r_bin.

RDS > 데이터베이스 생성 > MySQL 선택 > 프리티어 확인

설정

DB 인스턴스 식별자 정보
DB 인스턴스 이름을 입력하세요. 이름은 현재 AWS 리전에서 AWS 계정이 소유하는 모든 DB 인스턴스에 대해 고유해야 합니다.

docker-mysql-test

DB 인스턴스 식별자는 대소문자를 구분하지 않지만 'mydbinstance'와 같이 모두 소문자로 저장됩니다. 제약: 1~60자의 영숫자 또는 하이픈으로 구성되어야 합니다. 첫 번째 문자는 글자여야 합니다. 하이픈 2개가 연속될 수 없습니다. 하이픈으로 끝날 수 없습니다.

▼ 자격 증명 설정

마스터 사용자 이름 정보
DB 인스턴스의 마스터 사용자에 로그인 ID를 입력하세요.

user

1~16자의 영숫자. 첫 번째 문자는 글자여야 합니다.

☐ AWS Secrets Manager에서 마스터 보안 인증 정보 관리
Secrets Manager에서 마스터 사용자 보안 인증을 관리합니다. RDS는 사용자 대신 암호를 생성하고 수명 주기 동안 이를 관리할 수 있습니다.

Secrets Manager에서 마스터 사용자 보안 인증 정보를 관리하는 경우 일부 RDS 기능은 지원되지 않습니다. 자세히 알아보기

☐ 암호 자동 생성
Amazon RDS에서 사용자를 대신하여 암호를 생성하거나 사용자가 직접 암호를 지정할 수 있습니다.

마스터 암호 정보

제약 조건: 8자 이상의 인쇄 가능한 ASCII 문자. 다음은 포함할 수 없습니다. /(슬래시), (작은따옴표), (큰따옴표) 및 @(앳 기호).

마스터 암호 확인 정보

퍼블릭 액세스 가능성 > 예

▼ 추가 구성

데이터베이스 옵션, 암호화 켜짐, 백업 켜짐, 역추적 꺼짐, 유지 관리, CloudWatch Logs, 삭제 방지 꺼짐.

데이터베이스 옵션

초기 데이터베이스 이름 [정보](#)

TEST

데이터베이스 이름을 지정하지 않으면 Amazon RDS에서 데이터베이스를 생성하지 않습니다.

DB 파라미터 그룹 [정보](#)

docker-mysql-test ▼

옵션 그룹 [정보](#)

default:mysql-8-0 ▼

백업

생성된 데이터베이스 보안그룹

RDS > 데이터베이스 > docker-mysql-test

docker-mysql-test

요약

DB 식별자

docker-mysql-test

CPU

6.82%

상태

사용 가능

역할

인스턴스

현재 활동

0 연결

엔진

MySQL Community

연결 & 보안

모니터링

로그 및 이벤트

구성

유지 관리 및 백업

태그

연결 & 보안

엔드포인트 및 포트

엔드포인트

docker-mysql-test.cdizsrd2r4yy.us-east-2.rds.amazonaws.com

네트워킹

가용 영역

us-east-2a

VPC

보안

VPC 보안 그룹

default (sg-0410f7dcfd47169bd)

활성

EC2 > 보안 그룹 > sg-0410f7dcfd47169bd - default > 인바운드 규칙 편집

인바운드 규칙 편집

인바운드 규칙은 인스턴스에 도달하도록 허용된 수신 트래픽을 제어합니다.

인바운드 규칙 정보

보안 그룹 규칙 ID

sg-044ea1854603d8e21

유형 정보

모든 트래픽

프로토콜 정보

전체

포트 범위 정보

전체

소스 정보

사용자 지정

설명 - 선택 사항 정보

삭제

-

사용자 지정 TCP

TCP

3306

Anywhere-I...

sg-0410f7dcfd47169bd

0.0.0.0/0

Q

Q

삭제

삭제

규칙 추가

취소

변경 사항 미리 보기

규칙 저장

RDS > 데이터베이스 > docker-mysql-test

docker-mysql-test

요약

DB 식별자

docker-mysql-test

CPU

2.52%

상태

사용 가능

역할

인스턴스

현재 활동

0 연결

엔진

MySQL Community

연결 & 보안

모니터링

로그 및 이벤트

구성

유지 관리 및 백업

태그

연결 & 보안

엔드포인트 및 포트

엔드포인트

docker-mysql-test.cdziisd2r4yy.us-east-2.rds.amazonaws.com

포트

3306

네트워킹

가용 영역

us-east-2a

VPC

vpc-02e1102171936e479

서브넷 그룹

보안

VPC 보안 그룹

default (sg-0410f7dcfd47169bd)

활성

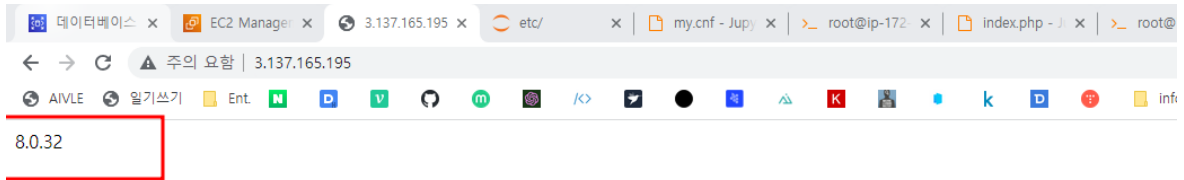
퍼블릭 액세스 가능

예

2 php에 rds 연동

/home/ubuntu/example/html/index.php에 엔드포인트, user, port 변경

```
<?php
$conn = mysqli_connect(
    'docker-mysql-test.cdziisd2r4yy.us-east-2.rds.amazonaws.com',
    'user',
    'password',
    'TEST',
    '3306'
);
if(mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: ".mysqli_connect_error();
}
$sql = "SELECT VERSION()";
$result = mysqli_query($conn, $sql);
$row = mysqli_fetch_array($result);
print_r($row["VERSION()"]);
?>
```



3 이전에 작업한 MySQL 컨테이너 삭제

why? ⇒ 이제는 AWS RDS를 사용할 것이기 때문이다!

```
docker ps -a
docker rm -f [CONTAINER ID]
```

▼ 08 GitHub에 Docker 프로젝트 올리기

1 Github 레파지토리 생성

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

lwy210

Repository name *

Docker-Practice

✓ Docker-Practice is available.

Great repository names are short and memorable. Need inspiration? How about [solid-engine](#) ?

Description (optional)

☐ Public

Anyone on the internet can see this repository. You choose who can commit.

☒ Private

You choose who can see and commit to this repository.

```
cd /home/ubuntu
git clone https://github.com/lwy210/Docker-Practice.git
```

오류 발생 > 해결방법 : <https://shortcuts.tistory.com/12>

*) 토큰을 password에 입력하자

2 Dockerfile, php 파일 생성 후 깃에 push

```
cd Docker-Practice
sudo vi Dockerfile
```

```
FROM ubuntu:18.04
MAINTAINER Wonyeong lee <leeeoo6436@naver.com>
```

```
# Avoiding user interaction with tzdata
ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get install -y software-properties-common && \
    add-apt-repository ppa:ondrej/php && \
    apt-get update && \
    apt-get install -y apache2 php7.2 libapache2-mod-php7.2 && \
    a2enmod php7.2

# Connect PHP & MySQL
RUN apt-get install -y php7.2-mysql

EXPOSE 80

CMD ["apachectl", "-D", "FOREGROUND"]
```

index.php

```
<?
    $conn = mysqli_connect(
        '3.137.165.195',
        'test',
        'password',
        'TEST',
        '9876'
    );
    if(mysqli_connect_errno()) {
        echo "Failed to connect to MySQL: ".mysqli_connect_error();
    }
    $sql = "SELECT VERSION()";
    $result = mysqli_query($conn, $sql);
    $row = mysqli_fetch_array($result);
    print_r($row["VERSION()"]);
?>
```

깃에 반영

```
git add .
git commit -m '초기 프로젝트 구성'
git push -u origin main
```

Docker-Practice Private

main 1 branch 0 tags

Go to file Add file <> Code

File	Commit Message	Commit Hash	Time
Dockerfile	초기 프로젝트 구성	ef230b0	now
README.md	Initial commit		18 minutes ago
index.php	초기 프로젝트 구성		now

▼ 09 DockerHub와 GitHub 연동하기

why? ⇒ 둘을 연동하면 DockerHub에서 Dockerfile 빌드를 자동으로 수행해준다. 우리가 소스코드를 수정해 깃허브에 업로드만 하면 DockerHub에서 감지해서 다시 빌드를 수행하므로 매우 쉽게 이미지를 컨테이너에 띄울 수 있게 된다.

1 DockerHub

<https://hub.docker.com/>

회원 가입 후 레파지토리 생성

Create repository

Namespace: lwy210

Repository Name *: **docker-practice**

Description

Visibility

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

☐ Public
Appears in Docker Hub search results

☒ **Private**
Only visible to you

[Cancel](#) [Create](#)

Pro tip

You can push a new image to this repository

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

Make sure to change tagname with your repository name

Personal	Pro	Team	Business
Includes the Docker essentials and is ideal for individual developers, education, open source communities, and small businesses.	Extends the Docker capabilities and includes pro tools for individual developers who want to accelerate their productivity.	Ideal for teams and includes capabilities for enhanced collaboration, productivity and security.	Ideal for medium and large businesses who need centralized management and advanced security capabilities.
\$0	\$5 /month	\$9 /user/month max 100 users minimum 5 seats	\$24 /user/month only available with an annual subscription
<ul style="list-style-type: none">Docker DesktopUnlimited public repositoriesDocker Engine + KubernetesLimited image pulls per dayUnlimited scoped tokens	<ul style="list-style-type: none">Everything in Personal, plus:Docker DesktopUnlimited private repositories5,000 image pulls per day5 concurrent builds300 vulnerability scans	<ul style="list-style-type: none">Everything in Pro, plus:Docker DesktopUnlimited teams15 concurrent buildsUnlimited vulnerability scansAdd users in bulkAudit logs	<ul style="list-style-type: none">Everything in Team, plus:Hardened Docker DesktopEnhanced Container IsolationSettings managementCentralized managementImage Access ManagementRegistry Access ManagementSingle Sign-On (SSO)SCIM user provisioning

흑..😭😭😭😭😭

▼ 10 Jenkins를 이용해 Docker 프로젝트 빌드해보기

Jenkins를 이용하면 배포 자동화가 가능해진다. 즉, 말 그대로 진짜 소스코드만 올려서 깃허브에 푸시하기만 해도 알아서 서버에 배포까지 완전히 다 맞춰주는 작업이 가능하다. 젠킨스를 이용하면 이게 더 쉬워진다!

1 pull Jenkins 후 컨테이너 구동

<https://hub.docker.com/r/jenkins/jenkins>

```
docker pull jenkins/jenkins:lts-jdk11
```

- 젠킨스 또한 컨테이너로 이용할 것이다.
- 젠킨스 컨테이너에서 새롭게 php 컨테이너를 띄우는 방식으로 개발할 것이다. (Docker in Docker 구조)

젠킨스 컨테이너 구동


```
docker run -d -p 8080:8080 -v /home/jenkins:/var/jenkins_home
-v /var/run/docker.sock:/var/run/docker.sock -u root jenkins/jenkins

docker ps -a
```

⚠️ `docker pull jenkins` 가 아닌 `docker pull jenkins/jenkins:lts-jdk11` 를 해서 확인해보니 Image 레파지토리가 `jenkins/jenkins`이므로 이에 맞춰서 구동

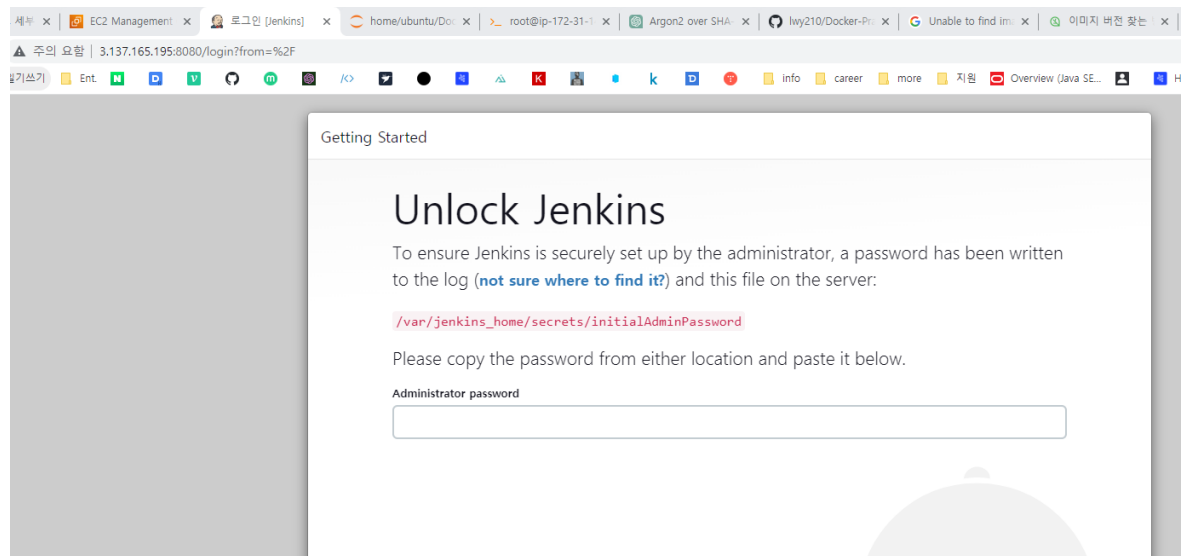
EC2 인바운드 규칙으로 8080 포트 열기

인바운드 규칙 편집 정보

인바운드 규칙은 인스턴스에 도달하도록 허용된 수신 트래픽을 제어합니다.

보안 그룹 규칙 ID	유형	프로토콜	포트 범위	소스	실명 - 선택 사항	작업
sgr-09dc30c6759033125	사용자 지정 TCP	TCP	8888	사용자 지정	0.0.0.0/0	삭제
sgr-04181880c5f60af0d	SSH	TCP	22	사용자 지정	0.0.0.0/0	삭제
sgr-056ba822f98e0ef61	HTTP	TCP	80	사용자 지정	0.0.0.0/0	삭제
sgr-0febe462bdce5b585	사용자 지정 TCP	TCP	9876	사용자 지정	0.0.0.0/0	삭제
-	사용자 지정 TCP	TCP	8080	Anywhere-1...	0.0.0.0/0	삭제

규칙 추가



비밀번호 얻기


```
docker logs [CONTAINER ID]
```



2 Jenkins를 이용해서 php docker 프로젝트를 빌드해 배포수행


Create new job


Enter an item name


* Required field


**Freestyle project**
이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(항상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

**Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**
다양한 환경에서의 테스트, 플러그인 특성 빌드, 기타 종종 저함 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.

**Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



**Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

Build Steps

- Execute Windows batch command
- Execute shell**
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets

Build Steps

 **Execute shell** 

Command

See [the list of available environment variables](#)