1. Apache2 Installation

```
$ cd ~
$ mkdir demo
$ cd demo
$ nano Dockerfile   <---반드시 대문자로 시작해야
   FROM ubuntu:20.04

   LABEL Author=Instructor
   LABEL Email=javaexpert@nate.com

   ENV DEBIAN_FRONTEND=noninteractive          #Ubuntu 20.04에서 사용자의 입력을 설치과정중에 하지 않기 위해

   RUN apt-get update && apt-get -y install apache2  # 주의 할점  : -y의 위치

   EXPOSE 80

   CMD ["/usr/sbin/apachectl", "-DFOREGROUND"]

$ sudo docker build -t mywebserver .

$ sudo docker run -d -p 8000:80 mywebserver
```

-Web Browser 에서 확인할 것

--http://{{IP}}:8000

2. Php 7.4 Installation

1)모든 Process 종료하기

```
$ sudo docker rm -f `sudo docker ps -a -q`
```

2)Dockerfile 수정

```
$ nano Dockerfile
   FROM ubuntu:20.04

   LABEL Author=Instructor
   LABEL Email=javaexpert@nate.com

   ENV DEBIAN_FRONTEND=noninteractive

   RUN apt-get update && apt-get -y install apache2
   RUN apt-get -y install software-properties-common
   RUN add-apt-repository ppa:ondrej/php
   RUN apt-get update && apt-get -y install php7.4

   EXPOSE 80

   CMD ["/usr/sbin/apachectl", "-DFOREGROUND"]
```

3)재 빌드

```
$ sudo docker build -t mywebserver .
```

4)Docker Image Run

```
$ sudo docker run -d -p 8000:80 -v /home/ubuntu/demo/html:/var/www/html mywebserver
```

5)index.php 파일 생성

```
$ cd html
$ sudo nano index.php
   <?php
      phpinfo();
   ?>
```

6)Web Browser 에서 확인할 것
-http://{{IP}}:8000

3. MySQL Installation
1)모든 Docker Process 삭제
$ sudo docker ps -a
$ sudo docker rm -f `sudo docker ps -a -q`

2)MySQL 실행하기
$ sudo docker run -d -p 3306:3306 --name db -e MYSQL_ROOT_PASSWORD=password mysql:5.7

$ sudo docker inspect db
"Networks" > "IPAddress" 확인할 것

$ sudo docker exec -it db bash
/# mysql -h 172.17.0.2 -u root --port 3306 -p
Enter password:password
mysql>exit
/# exit

4. PHP7.4 + MySQL
1)Dockerfile 수정하기
```
FROM ubuntu:20.04

LABEL Author=Instructor
LABEL Email=javaexpert@nate.com

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get -y install apache2
RUN apt-get -y install software-properties-common
RUN add-apt-repository ppa:ondrej/php
RUN apt-get update && apt-get -y install php7.4
RUN apt-get -y install php7.4-mysql          <---추가

EXPOSE 80

CMD ["/usr/sbin/apachectl", "-DFOREGROUND"]
```

3)재 빌드
$ sudo docker build -t mywebserver .

4)MySQL Run
$ sudo docker rm -f `sudo docker ps -a -q`
$ sudo docker run -d --name db -p 3306:3306 -v /home/ubuntu/dbdata:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=password mysql:5.7

$ sudo docker exec -it db bash
/# mysql -h localhost -u root -p

mysql> CREATE DATABASE Employee CHARACTER SET utf8;
Query OK, 1 row affected (0.00 sec)

mysql> CREATE USER scott IDENTIFIED BY 'tiger';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'scott'@'%';
Query OK, 0 rows affected (0.00 sec)

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql>exit
/# exit


5)Web Server Run
$ sudo docker run -d -p 8000:80 -v /home/ubuntu/demo/html:/var/www/html mywebserver


6)index.php 수정
$ cd html
$ sudo nano index.php

<?php
   $conn = mysqli_connect(
         '172.17.0.2',       <--------localhost가 아니라 반드시 IP를 적을 것
         'scott',
         'tiger',
         'Employee',
         '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()"]);
?>


7)Web Browser에서 확인
-http://{{IP}}:8000
   5.7.36



5. Git에 Docker Project 올리기
1)github에 login
   https://github.com/gitinstructor


2)Private Repository 생성
-왜 Private?
   --Database 인증정보처럼 민감한 데이터가 있기 때문...
-Repository name : docker-demo
-Private
-Create repository


3)현재 Cloud의 VM에는 Git이 기본적으로 설치되어 있음.
$ git --version으로 확인


4)방금 생성한 Repository를 Cloud VM에서 git clone 할 것
$ cd ~
$ git clone https://github.com/gitinstructor/docker-demo.git
Cloning into 'docker-demo...
Username for 'https://github.com': gitinstructor    <-- Repository를 Private로 생성했기 때문
Password for 'https://gitinstructor@github.com':    <---Personal Access Token 사용할 것
warning: You appear to have cloned an empty repository.


5)앞에서 생성한 Dockerfile 과 index.php를 clone한 docker-demo에 같이 복사해서 넣을 것
-일반적으로 Dockerfile과 index.php같은 소스코드를 같이 넣는다.
```

```
$ cd docker-demo/
$ cp ~/demo/Dockerfile .
$ cp ~/demo/html/index.php .
$ ls
Dockerfile  index.php
```

6)Github에 Push하기

```
$ git add .
$ git commit -m "Project Initialization"
[master (root-commit) f49a009] Project Initialization
 Committer: ubuntu <ubuntu@localhost.localdomain>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:
    git commit --amend --reset-author

 2 files changed, 32 insertions(+)
 create mode 100644 Dockerfile
 create mode 100644 index.php

$ git push
Username for 'https://github.com': gitinstructor
Password for 'https://gitinstructor@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 743 bytes | 743.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/gitinstructor/docker-demo.git
 * [new branch]      master -> master
```

7)Github에서 Refresh해서 확인할 것

6. DockerHub와 GitHub 연동하기

1)DockerHub에 Login

2)DockerHub에 Private Repository 생성하기

-이름 : docker-demo
-Visibility : Private
-Build Settings : Connected 클릭
-Select organization : git 계정 선택
-Select repository : docker-demo
-Create & Build button click
-그럼 DockerHub에서 Build가 진행됨.
-시간이 꽤 흘러야 함.
-화면 우측의 Recent builds에 목록에 보면 github계정/repository가 있고 그 아래에 Buid in 'master'의 아이콘이 노란색으로 현재 빌드중임을 알려준다.
-github계정/repository 링크를 클릭하면 In Progress가 현재 빌드 진행중임을, 그리고 빌드가 마치고 성공했으면 SUCCESS 초록색 글자가 나타난다.
-Build하는데 약 5분 정도 걸림.
-그 아래에 Build Log도 보이고 Dockerfile도 볼 수 있고 Readme도 볼 수 있다.
-이 Build Log는 우리가 수동으로 빌드했을 때의 로그와 동일하다.
-다시 General에 가면 노란색 버튼이 초록색 체크 아이콘으로 변경된 것을 볼 수 있다.
-그래서 이제까지 우리가 Cloud에서 생성했던 모든 Docker 이미지를 삭제해도 된다는 의미이다.

3)Cloud에 있는 모든 Docker Image 삭제

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

4)Github docker-demo repository의 README 파일 생성하기
-docker-demo repository에서 [Add a README] 클릭

Docker를 사용하는 진짜 이유 : CI/CD
Installation
<pre> <--일반적으로 pre tag로 작성
cd /home
sudo git clone https://github.com/gitinstructor/docker-demo <---github repository 경로
cd docker-demo
</pre>

Run
<pre>
# Login for Private Docker Repository
sudo docker login
sudo docker pull pythonexpert/docker-demo
sudo docker run -d --name db -p 3306:3306 -v /home/ubuntu/dbdata:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=password mysql:5.7
sudo docker run -dp 8000:80 -v /home/docker-demo/Project:/var/www/html pythonexpert/docker-demo
</pre>

-README 파일 COMMIT
-이렇게 하면 local machine에서 코드를 수정하면 그 변경된 내용이 commit만 하면 자동으로 dockerhub에서 build를 한다.

5)소스쪽에서 수정하고 다시 github에 push 하기

```
$ cd ~/docker-demo
$ mkdir Project
$ mv index.php ./Project/index.php
$ ls ./Project
index.php


$ git add .
$ git commit -m "index.php path changed"
[master 066cc46] index.php path changed
 Committer: ubuntu <ubuntu@localhost.localdomain>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 0 insertions(+), 0 deletions(-)
 rename index.php => Project/index.php (100%)


$ git push
Username for 'https://github.com': gitinstructor
Password for 'https://gitinstructor@github.com':
To https://github.com/gitinstructor/docker-demo.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/gitinstructor/docker-demo.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
```

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

-오류발생...그 이유는 README 파일을 수정했기 때문, 그러면 pull을 하면 됨.

$ git pull
Username for 'https://github.com': gitinstructor
Password for 'https://gitinstructor@github.com':

-그러면 README 파일을 불어옴
-바로 Ctrl + X로 빠져 나옴.
-다시 push 하면 반영됨.

$ git push
Username for 'https://github.com': gitinstructor
Password for 'https://gitinstructor@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 596 bytes | 298.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/gitinstructor/docker-demo.git
ed26cc5..f6369c0 master -> master

6)Github에서 Refresh 하면 docker-demo Repository에 Project 폴더가 생겼고, 그 폴더 안에 index.php 파일이 있는 것을 확인할 수 있다.
7)GitHub에서 변경된 것을 DockerHub가 감지하면 바로 다시 Build 한다.
8)만일 성공적으로 빌드가 되면, 화면 Refresh했을 때, 방금 SUCCESS 빌드한 빌드 상세 페이지의 README 파일이 위에서 새로 생성한 README 파일로 변경된 것을 확인할 수 있다.

7. DockerHub의 READMD 파일대로 수행해 보기
1)현재 모든 Docker Image혹은 Docker Container가 모두 없다는 것을 확인한다 .
$ sudo docker ps -a
$ sudo docker images
$ sudo docker volume ls
$ sudo docker volume rm volumeName

2)DockerHub의 README 파일의 내용대로 그대로 해 본다.

3)Web Browser에서 확인하면
-http://{{IP}}:8000
5.7.36

8. Jenkins를 이용해서 Docker Project Build 하기
1)모든 Docker Process 삭제
$ sudo docker ps -a
$ sudo docker rm -f `sudo docker ps -a -q`
$ sudo docker rmi -f `sudo docker images`

- 현재 모든 Docker Image혹은 Docker Container가 모두 없다는 것을 확인한다.

$ sudo docker ps -a
$ sudo docker images
$ sudo docker volume ls

2)Jenkins Container Image Pull
$ sudo docker pull jenkins/jenkins

$ sudo docker run -dp 8080:8080 -v /home/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -u root jenkins/jenkins

3)Web Browser에서
-http://{EIP}:8080
-Administrator password :
-Admin 패스워드는 $ sudo docker logs jenkins's pid 에서 비밀번호 확인할 것
-그리고 Continue 버튼 클릭...
-Install suggested plugins 버튼 클릭
-혹 설치 에러가 나도 [continue] 링크 클릭
-관리자 계정 생성하기
-아이디/패스워드 : admin/P@$$W0rd
-이름/Email : admin/javaexpert@nate.com
-Save and Continue
-Instance Configuration 창에서 Jenkins URL을 확인하고 [Save and Finish] 클릭
-Jenkins is ready! > [Start using Jenkins] click

4)Jenkins page에서
-왼쪽 프레임에서 [새로운 item] 클릭
-Enter an item name : Example
-Freesstyle project 선택 > OK
-General > Build > Add build step > Execute shell
-Command : DockerHub의 README에서
sudo docker pull pythonexpert/docker-demo
sudo docker run -d --name db -p 3306:3306 -v /home/ubuntu/dbdata:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=password mysql:5.7
sudo docker run -dp 8000:80 -v /home/docker-demo/Project:/var/www/html pythonexpert/docker-demo
-저장
-다시 젠킨스 페이지에서 좌측 프레임의 [Build Now] 클릭
-Build History의 해당 Build No. (예:#1) 클릭 > [Console Output] 클릭 하여 상세 내용 볼 수 있음.
-sudo : not found 에러 발생
-다시 Example Item click > [구성]
-sudo를 빼고 저장 후 [Build Now] 클릭
-이번에는 docker not found Error 발생
-jenkins Container 안에는 docker가 설치안되어 있으면 에러 발생함.

$ sudo docker exec -it jenkins' pid bash
/# docker version
bash: docker: command not found
apt-get update
apt-get install apt-transport-https ca-certificates curl gnupg lsb-release
curl -fsSLO https://get.docker.com/builds/Linux/x86_64/docker-17.04.0-ce.tgz
tar xvfz docker-17.04.0-ce.tgz
mv docker/docker /usr/local/bin
rm -f docker-17.04.0-ce.tgz
rm -rf docker
docker version

docker login
Username : pythonexpert
Password :
Login Succeeded

-Docker Hub의 docker-demo에 있는 READMD의 Installation 파트 실행
cd /home
git clone https://github.com/gitinstructor/docker-demo
Username ... : gitinstructor
Password ... :
...
Unpacking objects : 100% (12/12), done.
exit

-다시 Jenkins Page에서 Example Item click > [구성]
-sudo를 빼고 저장 후 [Build Now] 클릭

-[Console Output] 클릭 하여 자세한 Console 내용 확인
-그럼 수동으로 docker image run 하는 과정을 똑같이 Jenkins가 한다.

5)Web Browser에서 확인하면
-http://{{IP}}:8000
5.7.34

6)만일 접속 에러가 발생할 경우
-$ sudo docker inspect mysql's pid --> 혹시 MySQL IPAddress가 변경되었을 수 있다. (예:172.17.0.3)
-그럴 때는 index.php의 MySQL 접속주소를 172.17.0.3로 변경
$ cd /home/docker-demo/Project
$ sudo nano index.php
--접속 주소 172.17.0.3으로 변경

-index.php 변경됐으니까 다시 git에 업그레이드
$ sudo git add .
$ sudo git commit -m "index.php source changed."
$ sudo git push

-docker hub에서도 다시 build
-Jenkins Container 를 제외한 나머지 Container를 Process 에서 내리기
-Jenkins 에서 다시 Build Now

7)다음과 같은 에러 발생 시
-Failed to connect to MySQL: Access denied for user 'scott'@'172.17.0.4' (using password: YES)

9. Jenkins를 이용해서 원격에서 서버 프로그램 빌드하기
1)Jenkins Page에서 현재 작동중인 Build 중지하기

2)Jenkins Container를 제외한 나머지 Container를 Process에서 내리기

3)Jenkins 페이지에서 Example > 구성 > Build > Execute shell 수정
-docker-demo 로 이동하고
-pull 하며
-PHP Container의 이름을 myphp로 명명하고,
-Container를 올리기 전에 myphp가 있으면 프로세스에서 삭제한다. 있든 없든 true로 무조건 실행한다 .

```
cd /home/docker-demo
git pull
docker rm -f myphp || true
docker pull pythonexpert/docker-demo
docker run -d --name db -p 3306:3306 -v /home/ubuntu/dbdata:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=password mysql:5.7
docker run -dp 8000:80 -v /home/docker-demo/Project:/var/www/html --name myphp pythonexpert/docker-demo
```

4)Build를 원격으로 유발(Build Triggers)하기
-빌드 유발(Build Triggers) > 빌드를 원격으로 유발(예: 스크립트, Trigger builds remotely(e.g., from scripts)) 체크
-Authentication Token : 원래 어려운 Token을 넣어야 하나, 수업 상 간단하게 mybuild_token 으로 명명
-Save

5)GitHub 정보를 Jenkins Container안에 저장하기
-현재 Jenkins의 Build Script는 cd /home/docker-demo 후에 git pull을 해야 하는데, 현재 Private Repository이기 때문에 수정해야 한다.
-Jenkins Container의 bash로 들어간다.
$ sudo docker exec -it jenkins' pid bash
/#cd /home/docker-demo

-GitHub에서 Access Token 발행하기
--GitHub에서 본인의 계정의 Settings > Developer settings > Personal access tokens > Generate new token
--New personal access token

--Note : docker-demo를 위한 Access Token
--repo check
--admin:repo_hook check
--Generate token 버튼 클릭
-생성된 Access token을 복사하여 메모장에 붙여넣기

-Jenkins Container 안에서
/#git config --global credential.helper "cache --timeout=7200" --> 2시간만 id와 패스워드 저장하게끔...
/# git pull
/# Username for 'https://github.com' : gitinstructor
/# Password for 'https://gitinstructor@github.com' : Access Token 값 붙여넣기
/# exit

5)Jenkins Page의 구성(Configure)에 보면 위에서 설정한 Build Triggers(빌드 유발)의 [Authentication Token] 아래에 설명이 다음과 같이 나오는 것을 확인할 수 있다.
-Use the following URL to trigger build remotely(다음 URL을 사용하여 원격 빌드 유발): JENKINS_URL/job/Example/build?token=TOKEN_NAME or /buildWithParameters?token=TOKEN_NAME

6)따라서 현재 Build History에 빌드가 진행이 없는 것을 확인하고 위의 설명대로 URL에 Authentication Token을 넣어서 입력하기로 한다.
-http://{{EIP}}:8080/job/Example/build?token=mybuild_token