

Contents

- 이미지 검색하기
- 이미지 다운로드(Pull)하기
- 컨테이너 실행하기
- 이미지 업로드(Push)하기

이미지 검색하기

자주 사용되는 도커 명령어를 알아보겠습니다.

도커 이미지를 검색하는 명령어는 `docker search` 입니다.

도커허브에서 Ubuntu 이미지를 찾아볼까요?

```
ubuntu@ip-10-0-1-14:~$ docker search ubuntu
NAME                           DESCRIPTION                                     STARS   OFFICIAL  AUTOMATED
ubuntu                          Ubuntu is a Debian-based Linux operating sys... 15508   [OK]
websphere-liberty                WebSphere Liberty multi-architecture images ... 291     [OK]
ubuntu-upstart                   DEPRECATED, as is Upstart (find other proces... 112     [OK]
neurodebian                      NeuroDebian provides neuroscience research s... 98      [OK]
ubuntu/nginx                     Nginx, a high-performance reverse proxy & we... 75      -
open-liberty                      Open Liberty multi-architecture images based... 56      [OK]
ubuntu/apache2                   Apache, a secure & extensible open-source HT... 53      -
ubuntu-debootstrap               DEPRECATED; use "ubuntu" instead                 50      [OK]
ubuntu/squid                     Squid is a caching proxy for the Web. Long-t... 49      -
ubuntu/bind9                     BIND 9 is a very flexible, full-featured DNS... 44      -
ubuntu/mysql                     MySQL open source fast, stable, multi-thread... 41      -
ubuntu/prometheus                Prometheus is a systems and service monitori... 35      -
ubuntu/postgres                  PostgreSQL is an open source object-relation... 23      -
ubuntu/kafka                      Apache Kafka, a distributed event streaming ... 21      -
ubuntu/redis                      Redis, an open source key-value store. Long-... 16      -
ubuntu/prometheus-alertmanager  Alertmanager handles client alerts from Prom... 8       -
ubuntu/dotnet-deps               Chiselled Ubuntu for self-contained .NET & A... 6       -
...생략...
```

명령어 : `docker search ubuntu`

Docker & Kubernetes - [Hands-on] 02. Docker commands

<https://hub.docker.com/>에서도 한번 검색을 해보세요.

The screenshot shows the Docker Hub search interface with the query 'ubuntu' entered in the search bar. The results page displays 1-25 of 124,052 results for 'ubuntu'. The results are listed in a grid format, each showing a thumbnail, the repository name, the official status (DOCKER OFFICIAL IMAGE), the last update time, download count, star count, and a brief description. The repositories listed are:

- ubuntu** (DOCKER OFFICIAL IMAGE) - Updated 18 days ago. Description: Ubuntu is a Debian-based Linux operating system based on free software. Tags: Linux, IBM Z, 386, x86-64, ARM, ARM 64, PowerPC 64 LE, riscv64. Downloads: 1B+, Stars: 10K+.
- websphere-liberty** (DOCKER OFFICIAL IMAGE) - Updated 18 days ago. Description: WebSphere Liberty multi-architecture images based on Ubuntu 18.04. Tags: Linux, PowerPC 64 LE, IBM Z, x86-64, 386. Downloads: 10M+, Stars: 286.
- open-liberty** (DOCKER OFFICIAL IMAGE) - Updated 18 days ago. Description: Open Liberty multi-architecture images based on Ubuntu 18.04. Tags: Linux, x86-64, PowerPC 64 LE, IBM Z, 386. Downloads: 10M+, Stars: 53.
- ubuntu-debootstrap** (DOCKER OFFICIAL IMAGE) - Updated 6 years ago. Description: DEPRECATED; use "ubuntu" instead. Tags: None. Downloads: 5M+, Stars: 46.
- neurodebian** (DOCKER OFFICIAL IMAGE) - Updated 2 days ago. Description: NeuroDebian provides neuroscience research software for Debian, Ubuntu, and other derivatives. Tags: Linux, x86-64. Downloads: 5M+, Stars: 91.

On the left side of the search results, there are sidebar filters for 'Products' (Images, Plugins), 'Trusted Content' (Docker Official Image, Verified Publisher, Open Source Program), 'Operating Systems' (Linux, Windows), and 'Architectures' (ARM, ARM 64, IBM POWER, IBM Z, PowerPC 64 LE, x86, x86-64).

두 가지 결과가 어떤지 비교도 해보시구요.

이미지 다운로드(Pull)하기

이제 ubuntu 이미지를 다운로드(pull) 해 보겠습니다.

```
ubuntu@ip-10-0-1-14:~$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
405f018f9d1d: Pull complete
Digest: sha256:b6b83d3c331794420340093eb706a6f152d9c1fa51b262d9bf34594887c2c7ac
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

명령어 : `docker pull ubuntu`

tag를 특정해서(18.04) 다운로드도 해보구요.

```
ubuntu@ip-10-0-1-14:~$ docker pull ubuntu:18.04
18.04: Pulling from library/ubuntu
09db6f815738: Pull complete
Digest: sha256:478caf1bec1af54a58435ec681c8755883b7eb843a8630091890130b15a79af
Status: Downloaded newer image for ubuntu:18.04
docker.io/library/ubuntu:18.04
```

명령어 : `docker pull ubuntu:18.04`

Docker & Kubernetes - [Hands-on] 02. Docker commands

받아온 이미지를 확인해볼까요?

```
ubuntu@ip-10-0-1-14:~$ docker images ubuntu
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
ubuntu          latest   27941809078c  2 weeks ago  77.8MB
ubuntu          18.04   ad080923604a  2 weeks ago  63.1MB
```

명령어 : `docker images ubuntu`

tag를 명시하지 않은 경우는 default tag인 `latest`를 받아오네요.

Docker & Kubernetes - [Hands-on] 02. Docker commands

컨테이너 실행하기

이제 실행(run)을 해보겠습니다.

```
ubuntu@ip-10-0-1-14:~$ docker run --interactive --tty ubuntu /bin/bash
root@060b1a36d1e5:/#
```

명령어 : `docker run --interactive --tty ubuntu /bin/bash`
--interactive --tty 는 [-it]로 줄여서 쓸 수도 있습니다.

- `--interactive --tty (-it)`로 실행했기 때문에 ubuntu의 bash shell에 콘솔로 연결되었습니다. (프롬프트 확인!)

실행된 ubuntu의 OS 정보를 확인 해볼까요?

```
root@060b1a36d1e5:/# cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.1 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.1 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
```

명령어 : `cat /etc/os-release`

Docker & Kubernetes - [Hands-on] 02. Docker commands

Ubuntu 22.04 LTS로 실행된 것을 확인할 수 있습니다. (실행한 시기에 따라 달라질 수 있습니다.)

이제 `exit` 명령어로 컨테이너를 빠져나오겠습니다.

```
root@060b1a36d1e5:/# exit  
exit  
ubuntu@ip-10-0-1-14:~$
```

명령어 : `exit`

- 참고 : `exit` 는 컨테이너를 stop합니다. stop하지 않고 detach만 하기 위해서는 `ctrl-p` + `ctrl-q` 를 이용하면 됩니다.

Docker & Kubernetes - [Hands-on] 02. Docker commands

이번에는 `ubuntu:18.04` 를 실행해봅시다.

```
ubuntu@ip-10-0-1-14:~$ docker run --interactive --tty ubuntu:18.04 /bin/bash
root@31d0f5ae7f56:/#
```

명령어 : `docker run --interactive --tty ubuntu:18.04 /bin/bash`

좀전과는 다르게 `tag(18.04)` 를 명시해서 실행했습니다.

`cat /etc/os-release` 의 결과는 어떻게 나올까요?

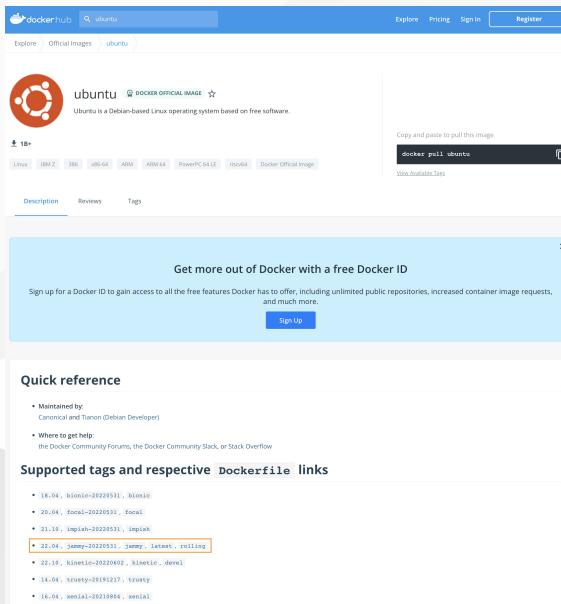
```
root@31d0f5ae7f56:/# cat /etc/os-release
NAME="Ubuntu"
VERSION="18.04.6 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04.6 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
```

명령어 : `cat /etc/os-release`

둘의 차이를 찾으셨나요? ↗_↗ (힌트 : VERSION)

Docker & Kubernetes - [Hands-on] 02. Docker commands

https://hub.docker.com/_/ubuntu 를 보시면, 어떤 tag가 latest인지 알 수 있습니다.



이제 `exit` 명령어로 컨테이너에서 나와주세요.

```
root@31d0f5ae7f56:/# exit  
exit  
ubuntu@ip-10-0-1-14:~$
```

명령어 : `exit`

Docker & Kubernetes - [Hands-on] 02. Docker commands

이번엔 다른 방법(`--detach`)으로 실행해 보겠습니다. (다른 이미지를 사용합니다.)

```
ubuntu@ip-10-0-1-14:~$ docker run --detach --name my-nginx --publish 8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
b85a868b505f: Pull complete
f4407ba1f103: Pull complete
4a7307612456: Pull complete
935cecace2a0: Pull complete
8f46223e4234: Pull complete
fe0ef4c895f5: Pull complete
Digest: sha256:10f14ffa93f8dedf1057897b745e5ac72ac5655c299dade0aa434c71557697ea
Status: Downloaded newer image for nginx:latest
f87853d90ac2305aa55945ea7babf3888ea5b13024046aead8968da2315b135b
ubuntu@ip-10-0-1-14:~$
```

명령어 : `docker run --detach --name my-nginx --publish 8080:80 nginx`

이전에 `--interactive` 옵션을 적용했을때와는 달리, 프롬프트가 그대로 있네요.

이제 `docker ps --all` 명령어로 컨테이너 목록을 조회해보세요.

```
ubuntu@ip-10-0-1-14:~$ docker ps --all
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
f87853d90ac2        nginx              "/docker-entrypoint...."   About a minute ago   Up About a minute   0.0.0.0:8080->80/tcp, :::8080->80/tcp   my-nginx
31d0f5ae7f56        ubuntu:18.04       "/bin/bash"            14 minutes ago     Exited (0) 3 minutes ago
060b1a36d1e5        ubuntu              "/bin/bash"            25 minutes ago     Exited (0) 25 minutes ago
wonderful_bassi
determined_mahavira
```

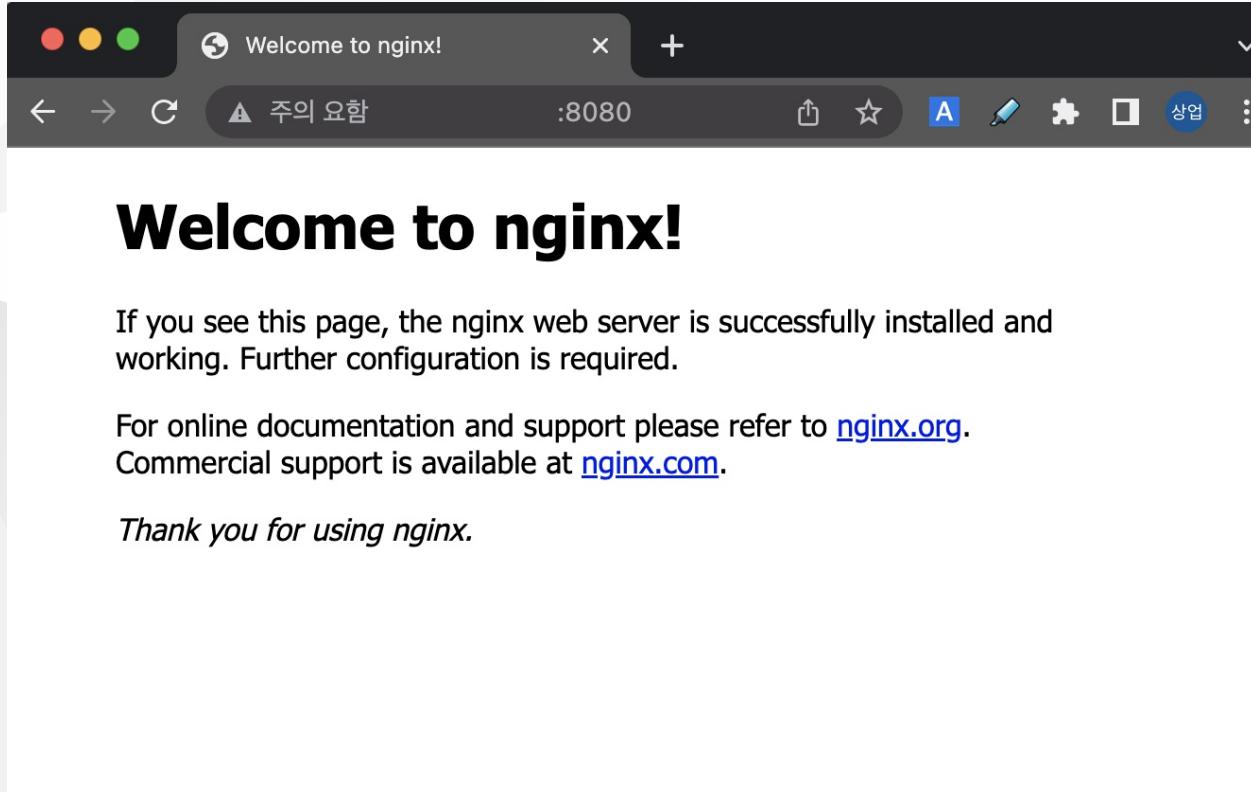
이전에 실행했던 ubuntu와 nginx가 보일거예요.

ubuntu는 Exited 상태이고, nginx는 Running 상태입니다.

Docker & Kubernetes - [Hands-on] 02. Docker commands

nginx가 정말 Running 상태인지 8080번 포트로 접속해서 확인도 해보세요.

- AWS EC2인 경우 인스턴스의 Public IPv4 address로 접속하면 됩니다. (e.g. <http://IP:8080/>)
- Security group의 Inbound rule에 8080번 포트에 대한 규칙이 있어야 합니다.



Docker & Kubernetes - [Hands-on] 02. Docker commands

이번엔 `docker stop` 명령어로 nginx 컨테이너를 멈춰봅시다.

```
ubuntu@ip-10-0-1-14:~$ docker stop $(docker ps --filter "name=my-nginx" --quiet)  
f87853d90ac2
```

명령어 : `docker stop $(docker ps --filter "name=my-nginx" --quiet)`

`docker ps --all`로 상태도 확인해보시고, 8080번 포트로 접속이 되는지 확인도 해보세요.

```
ubuntu@ip-10-0-1-14:~$ docker ps --all  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
f87853d90ac2 nginx "/docker-entrypoint..." 13 minutes ago Exited (0) About a minute ago my-nginx  
31d0f5ae7f56 ubuntu:18.04 "/bin/bash" 26 minutes ago Exited (0) 15 minutes ago wonderful_bassi  
060b1a36d1e5 ubuntu "/bin/bash" 37 minutes ago Exited (0) 37 minutes ago determined_mahavira
```

명령어 : `docker ps --all`

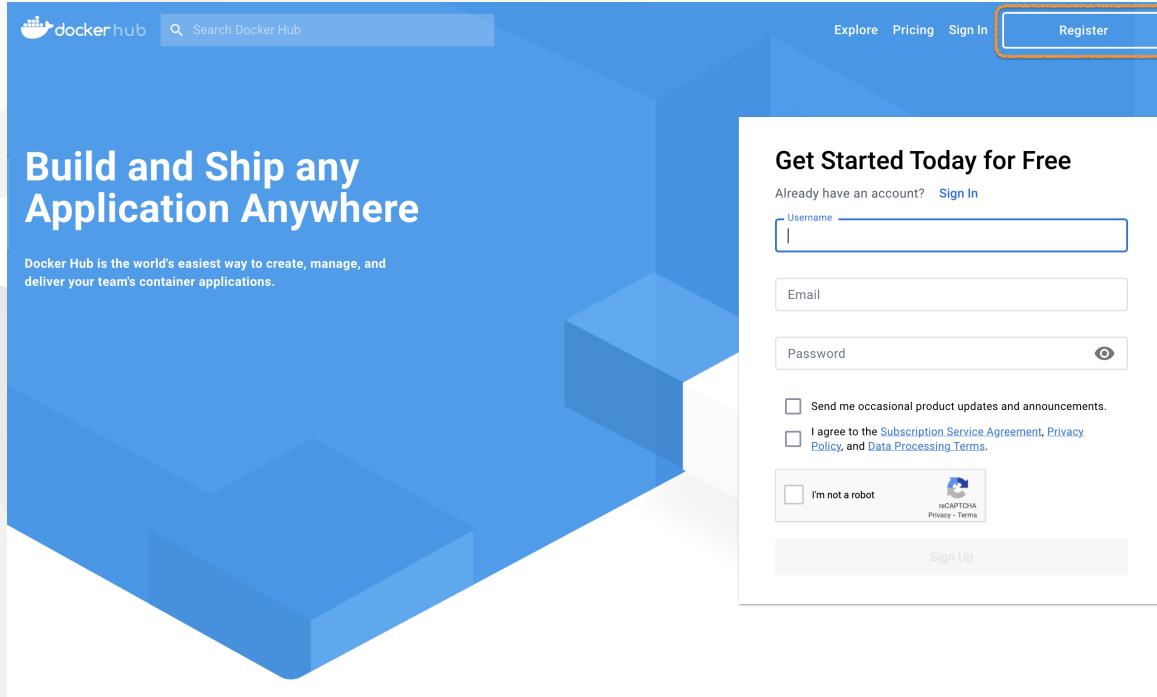
- Exited 상태인 컨테이너는 `--all (-a)` 옵션을 적용해야 조회가 됩니다.

`docker start` 와 `docker restart` 는 직접 명령어를 만들어서 한번 해보세요.

이미지 업로드(Push) 하기

이제 도커 레지스트리에 대해 알아보고, 우리가 만든 애플리케이션을 등록해 보겠습니다.

먼저 <https://hub.docker.com/> 에 가입(Register)을 합니다.



Register를 클릭하고, Docker account를 하나 만듭니다. (이미 있으면 있는 Account를 사용해도 됩니다.)

Docker & Kubernetes - [Hands-on] 02. Docker commands

이제 실습을 위해서 Repository를 하나 생성합니다.

로그인 후 **Create Repository** 버튼을 클릭해서 시작하면 됩니다.

이름은 todo-app 으로 하고, Visibility는 Private으로 합니다.

The screenshot shows the Docker Hub interface for creating a new repository. In the top navigation bar, there are links for 'Explore', 'Repositories', 'Organizations', 'Help', and a user profile for 'rogallo'. Below the navigation, it says 'Using 0 of 1 private repositories. [Get more](#)'. The main section is titled 'Create repository' and has a form where 'repository name' is set to 'todo-app'. Under 'Visibility', the 'Private' option is selected, while 'Public' is unselected. A note says 'Only visible to you'. At the bottom right of the form are 'Cancel' and 'Create' buttons. A 'Pro tip' box contains the following text and CLI commands:
You can 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.

무료 계정인 경우 Private repository는 하나만 만들 수 있습니다.

이미 사용중인 Private repository가 있으면, Public으로 만들어도 됩니다.

Docker & Kubernetes - [Hands-on] 02. Docker commands

이제 여러분의 Docker repository가 하나 생겼습니다.
앞으로 이 곳에 여러분의 컨테이너 이미지를 저장하고 사용하면 됩니다.

The screenshot shows the Docker Hub interface for a private repository named 'rogallo/todo-app'. The top navigation bar includes 'Explore', 'Repositories', 'Organizations', 'Help', 'Upgrade', and a user profile for 'rogallo'. The repository path 'rogallo/todo-app' is visible in the breadcrumb menu. The main content area displays the repository details:

- Name:** rogallo / todo-app (highlighted with an orange border)
- Description:** Private repository for todo-app (with edit icon)
- Last pushed:** a few seconds ago
- Docker commands:** docker push rogallo/todo-app:tagname (highlighted with an orange border)
- Tags:** This repository is empty. When it's not empty, you'll see a list of the most recent tags here. (with VULNERABILITY SCANNING - DISABLED link)
- Automated Builds:** Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating. Available with Pro, Team and Business subscriptions. (with Upgrade and Learn more buttons)
- README:** Repository description is empty. Click [here](#) to edit.

[USER-NAME]/[REPOSITORY-NAME] 이 여러분의 Repository입니다. (e.g. rogallo/todo-app)

Docker & Kubernetes - [Hands-on] 02. Docker commands

샘플 애플리케이션 이미지를 만들어 볼까요?
먼저 소스코드를 Github에서 clone 합니다.

```
ubuntu@ip-10-0-1-14:~$ git clone https://github.com/JungSangup/todo_list_manager.git app
Cloning into 'app'...
remote: Enumerating objects: 52, done.
remote: Counting objects: 100% (52/52), done.
remote: Compressing objects: 100% (49/49), done.
remote: Total 52 (delta 2), reused 52 (delta 2), pack-reused 0
Receiving objects: 100% (52/52), 1.67 MiB | 4.59 MiB/s, done.
Resolving deltas: 100% (2/2), done.
```

명령어 : `git clone https://github.com/JungSangup/todo_list_manager.git app`

그리고, 소스코드가 있는 경로로 이동합니다.

```
ubuntu@ip-10-0-1-14:~$ cd app
ubuntu@ip-10-0-1-14:~/app$
```

명령어 : `cd app`

샘플 소스코드에는 두 개의 Tag가 있습니다.

```
ubuntu@ip-10-0-1-14:~$ git tag
v1.0.0
v2.0.0
```

명령어 : `git tag`

Docker & Kubernetes - [Hands-on] 02. Docker commands

먼저 v1.0.0 이미지를 만듭니다. (`docker build` 명령어를 이용합니다.)
v1.0.0 tag로 checkout을 하구요,

```
ubuntu@ip-10-0-1-14:~$ git checkout v1.0.0
HEAD is now at c7a54f7 .
```

명령어 : `git checkout v1.0.0`

이제 Dockerfile을 이용해서 빌드를 합니다.

```
ubuntu@ip-10-0-1-14:~/app$ docker build -t rogallo/todo-app:1.0.0 .
Sending build context to Docker daemon 6.488MB
Step 1/5 : FROM node:10-alpine
10-alpine: Pulling from library/node
ddad3d7c1e96: Already exists
de915e575d22: Already exists
7150aa69525b: Already exists
d7aa47be044e: Already exists
Digest: sha256:dc98dac24efd4254f75976c40bce46944697a110d06ce7fa47e7268470cf2e28
Status: Downloaded newer image for node:10-alpine
--> aa67ba258e18
Step 2/5 : WORKDIR /app
--> Running in aa555177ae12
Removing intermediate container aa555177ae12
--> 4b535d14ddc1
Step 3/5 : COPY .
--> 6794fc99530e
Step 4/5 : RUN yarn install --production
--> Running in 0c148b3bb386
yarn install v1.22.5
[1/4] Resolving packages...
[2/4] Fetching packages...
```

Docker & Kubernetes - [Hands-on] 02. Docker commands

```
info fsevents@1.2.9: The platform "linux" is incompatible with this module.  
info "fsevents@1.2.9" is an optional dependency and failed compatibility check. Excluding it from installation.  
[3/4] Linking dependencies...  
[4/4] Building fresh packages...  
Done in 8.56s.  
Removing intermediate container 0c148b3bb386  
--> 77980b3ee3d7  
Step 5/5 : CMD ["node", "/app/src/index.js"]  
--> Running in 96d732e03656  
Removing intermediate container 96d732e03656  
--> bed2b2f466e4  
Successfully built bed2b2f466e4  
Successfully tagged rogallo/todo-app:1.0.0
```

명령어 : `docker build --tag [USER-NAME]/todo-app:1.0.0 .`
[USER-NAME]에는 여러분의 정보로 채워넣어 주세요.

Docker & Kubernetes - [Hands-on] 02. Docker commands

이제 v2.0.0 이미지를 만듭니다.

v2.0.0 tag로 checkout을 하구요,

```
ubuntu@ip-10-0-1-14:~$ git checkout v2.0.0
Previous HEAD position was c7a54f7 .
HEAD is now at d1c1aaaf Update index.html
```

명령어 : `git checkout v2.0.0`

Dockerfile을 이용해서 빌드를 합니다.

```
ubuntu@ip-10-0-1-14:~/app$ docker build -t rogallo/todo-app:2.0.0 .
Sending build context to Docker daemon 6.489MB
...생략...
```

명령어 : `docker build --tag [USER-NAME]/todo-app:2.0.0 .`

[USER-NAME]에는 여러분의 정보로 채워넣어 주세요.

그리고, 만들어진 이미지를 확인합니다.

```
ubuntu@ip-10-0-1-14:~/app$ docker images rogallo/todo-app
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
rogallo/todo-app  2.0.0   51829023f79f  45 seconds ago  172MB
rogallo/todo-app  1.0.0   bed2b2f466e4  14 minutes ago  172MB
```

명령어 : `docker images rogallo/todo-app`

[USER-NAME]에는 여러분의 정보로 채워넣어 주세요.

Docker & Kubernetes - [Hands-on] 02. Docker commands

이제 우리가 만든 이미지를 우리의 Docker hub repository에 업로드(push)해 보겠습니다.
먼저 로그인을 하구요,

```
ubuntu@ip-10-0-1-14:~/app$ docker login -u rogallo
Password:
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

명령어 : `docker login -u [USER-NAME]`

[USER-NAME]에는 여러분의 정보로 채워넣어 주세요.

아래 명령어로 docker hub의 우리 repository에 업로드(push) 해볼까요?

```
ubuntu@ip-10-0-1-14:~/app$ docker push rogallo/todo-app:1.0.0
The push refers to repository [docker.io/rogallo/todo-app]
fa33f1a79c07: Pushed
4fa6ccb17690: Pushed
7f5055c91ad5: Pushed
edff9ff691d5: Mounted from rogallo/101-todo-app
cbe4b9146f86: Mounted from rogallo/101-todo-app
a6524c5b12a6: Mounted from rogallo/101-todo-app
9a5d14f9f550: Mounted from rogallo/101-todo-app
1.0.0: digest: sha256:146bff86564f7937dad94f018a5e801ad6cf7e0fc03be810e02ead6376fa3b05 size: 1787
```

명령어 : `docker push [USER-NAME]/todo-app:1.0.0`

[USER-NAME]에는 여러분의 정보로 채워넣어 주세요.

Docker & Kubernetes - [Hands-on] 02. Docker commands

똑 같은 방법으로 두 번째 이미지도 push합니다.

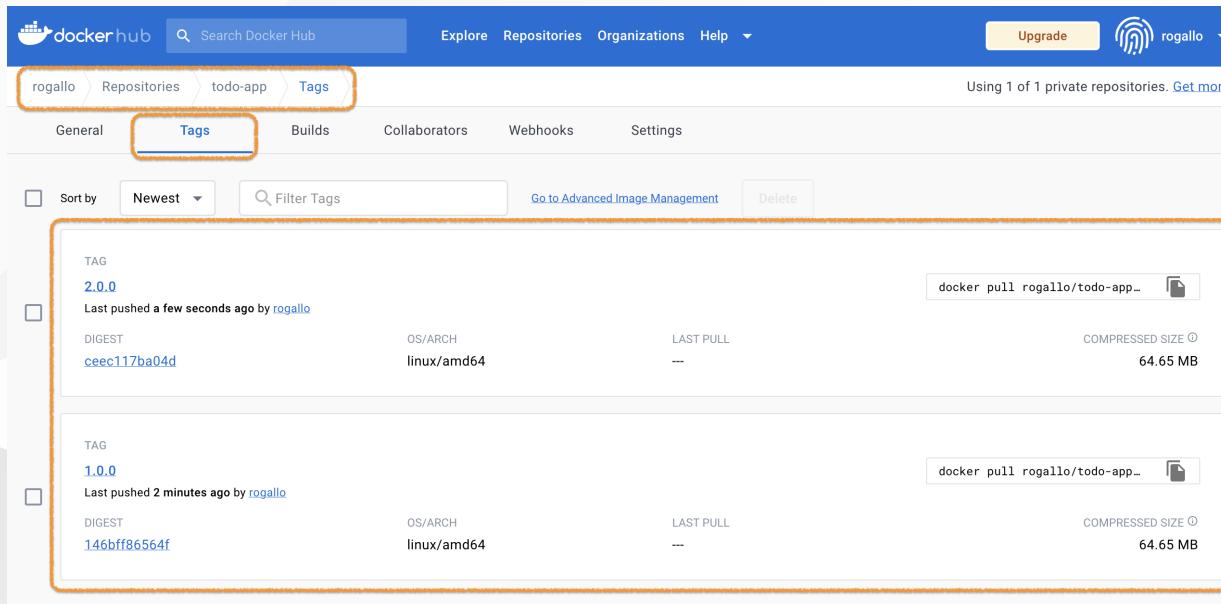
```
ubuntu@ip-10-0-1-14:~/app$ docker push rogallo/todo-app:2.0.0
The push refers to repository [docker.io/rogallo/todo-app]
6dbd0bd5a818: Pushed
8c45740b3656: Pushed
7f5055c91ad5: Layer already exists
edff9ff691d5: Layer already exists
cbe4b9146f86: Layer already exists
a6524c5b12a6: Layer already exists
9a5d14f9f550: Layer already exists
2.0.0: digest: sha256:ceec117ba04d60ac07cbfd601bd282e07de436683a9a4cdae81ff641e0351119 size: 1787
```

명령어 : `docker push [USER-NAME]/todo-app:2.0.0`

[USER-NAME]에는 여러분의 정보로 채워넣어 주세요.

Docker & Kubernetes - [Hands-on] 02. Docker commands

<https://hub.docker.com/> 에 방금 push한 이미지가 잘 올라가 있나요?



The screenshot shows the Docker Hub interface for a private repository named 'todo-app'. The 'Tags' tab is selected, highlighted with a blue border. Two tags are listed: '2.0.0' and '1.0.0'. Both tags were pushed by the user 'rogallo'. The '2.0.0' tag was pushed 'a few seconds ago' and has a digest of 'ceec117ba04d'. The '1.0.0' tag was pushed '2 minutes ago' and has a digest of '146bff86564f'. Both images are tagged as 'linux/amd64' and have a compressed size of 64.65 MB. Each tag entry includes a 'docker pull' command and a copy icon.

축하합니다. (o>◡<) ,

이제 여러분들의 저장공간도 생겼고, 언제 어디서든 방금 올려두신 이미지를 이용해서 여러분의 샘플 애플리케이션을 실행해보실 수 있게 됐습니다.

이번 실습은 여기까지입니다.

위의 두 개 이미지는 뒤의 과정에서 계속 필요하니, 잘 준비해주세요.