컨테이너 기반 가상화

2023. 08.

도커(Docker)란? 컨테이너 기반 가상화 도구

도커(Docker) 컨테이너는 일종의 소프트웨어를 소프트웨어의 실행에 필요한 모든 것을 포함하는 완전한 파일 시스템 안에 감싼다. 여기에는 코드, 런타임, 시스템 도구, 시스템 라이브러리 등 서버에 설치되는 무엇이든 아우른다. 이는 실행 중인 환경에 관계 없이 언제나 동일하게 실행될 것을 보증한다.

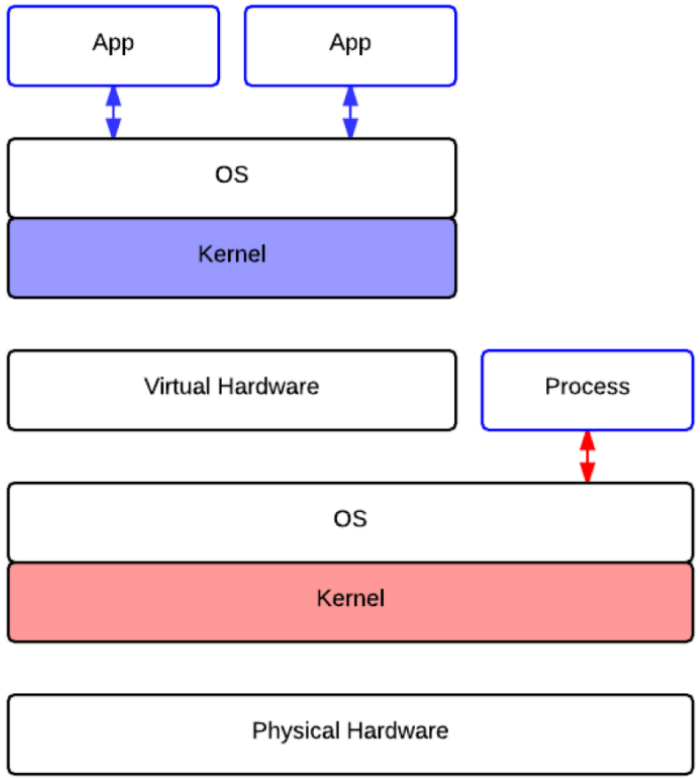
가상머신(VM)과 클라우드

가상의 컴퓨터. 가상으로 하드웨어를 생성하고 OS 를 설치하여 실제 컴퓨터와 격리된 가상의 환경 구축하여 사용 가능.

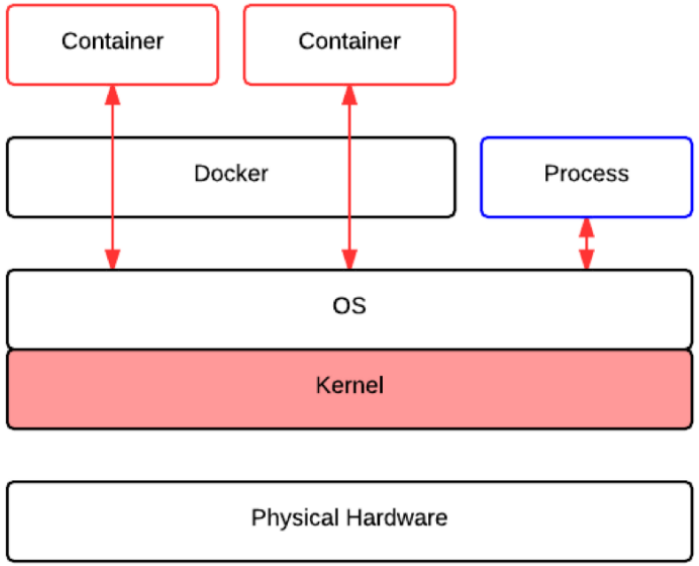
시스템 지원을 VM 으로 나눠서 사용할 수 있음.

VirtualBox, VMWare, Hyper V, Parallels, Xen 등이 있음

가상머신(VM) vs Docker 아키텍쳐 비교



Virtual Machine



Docker

컨테이너(Container)란?

하드웨어 가상화 없이 격리된 환경에서 실행되는 프로세스

VM(Virtual Machine)은 호스트OS 위에 게스트OS를 가상화 하여 사용하는 방식이다. 게스트 OS는 호스트OS의 자원을 할당받아 하이퍼바이저를 이용해 가상화하게된다. 하이퍼바이저는 호스트 시스템에서 다수의 게스트 OS를 구동할 수 있게 해주는 소프트웨어

VM은 호스트OS의 리소스 분할(+오버헤드)과 OS위에 또 OS를 설치하기 때문에 속도저하의 단점이 있음

컨테이너 기술은 호스트OS를 공유하며, 여러개의 컨테이너들이 서로 영향을 미치지 않고 독립적으로 실행되어 가볍다.
가장 큰 차이점은 프로세스를 격리 하는 것이며, VM은 가상머신들 사이에 오버헤드가 발생할때 상당한 시간이 걸린다.
반면, 컨테이너 기술은 독립적으로, 호스트OS의 자원을 공유하며 각각 필요한 자원들을 할당받아 실행되기 때문에 오버헤드가 적다.

도커 사용의 장점은?

Docker를 사용하면 코드를 더 빨리 전달하고, 애플리케이션 운영을 표준화하고, 코드를 원활하게 이동하고, 리소스 사용률을 높여 비용을 절감할 수 있습니다. Docker를 사용하면 어디서나 안정적으로 실행할 수 있는 단일 객체를 확보하게 됩니다.

도커의 이미지란?

특정 프로세스를 실행하기 위한 환경

이미지는 파일들의 집합

프로세스가 실행되는 환경도 결국 파일들의 집합

1) 따라서 도커 이미지의 용량은 보통 수백MB ~ 수GB가 넘는다. 하지만 가상머신의 이미지에 비하면 굉장히 적은 용량이다.
2) 이미지는 상태 값을 가지지 않고 변하지 않는다. (Immutable)
3) 하나의 이미지는 여러 컨테이너를 생성할 수 있고, 컨테이너가 삭제되더라도 이미지는 변하지 않고 그대로 남아 있음.

도커의 이미지란? (계속)

4) 도커 이미지들은 github과 유사한 서비스인 DockerHub를 통해 버전 관리 및 배포 (push&pull)가 가능하다.

5) 다양한 API가 제공되어 원하는 만큼 자동화가 가능하다.

6) 도커는 Dockerfile이라는 파일로 이미지를 만든다. Dockerfile에는 소스와 함께 의존성 패키지 등 사용했던 설정 파일을 버전 관리하기 쉽도록 명시되어진다.

컨테이너 생성

컨테이너의 생성은 기본적으로 '도커파일'이라는 것을 이용한다.
가상화 할 프로그램들을 '도커파일'이란 DSL(Domain Specific Language)형태로 작성한다.

컨테이너 생성과정은 다음과 같다.

도커파일 -> (build) = 도커이미지 생성
도커이미지 -> (run) = 도커컨테이너 생성

도커 파일은 소스와 함께 버전관리도 된다.
도커 파일로 생성한 이미지들은 용량이 매우 크기 때문에 '도커허브'라는 곳에서 무료로 관리 할 수 있어 누구나 쉽게 공개 이미지들을 다운로드 받을 수 있다.

도커 설치 (Linux)

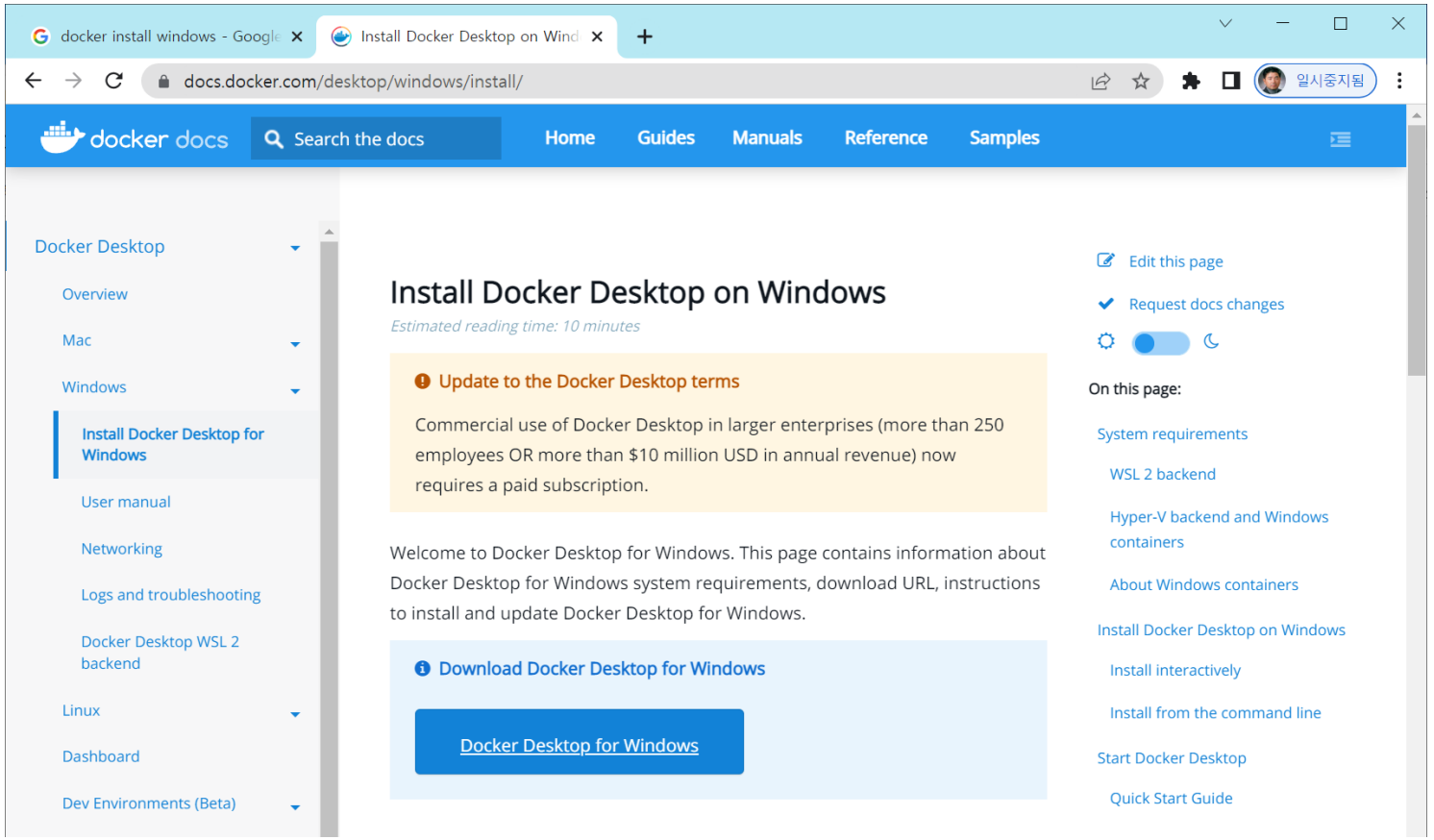
Linux

$ curl -s https://get.docker.com/ | sudo sh

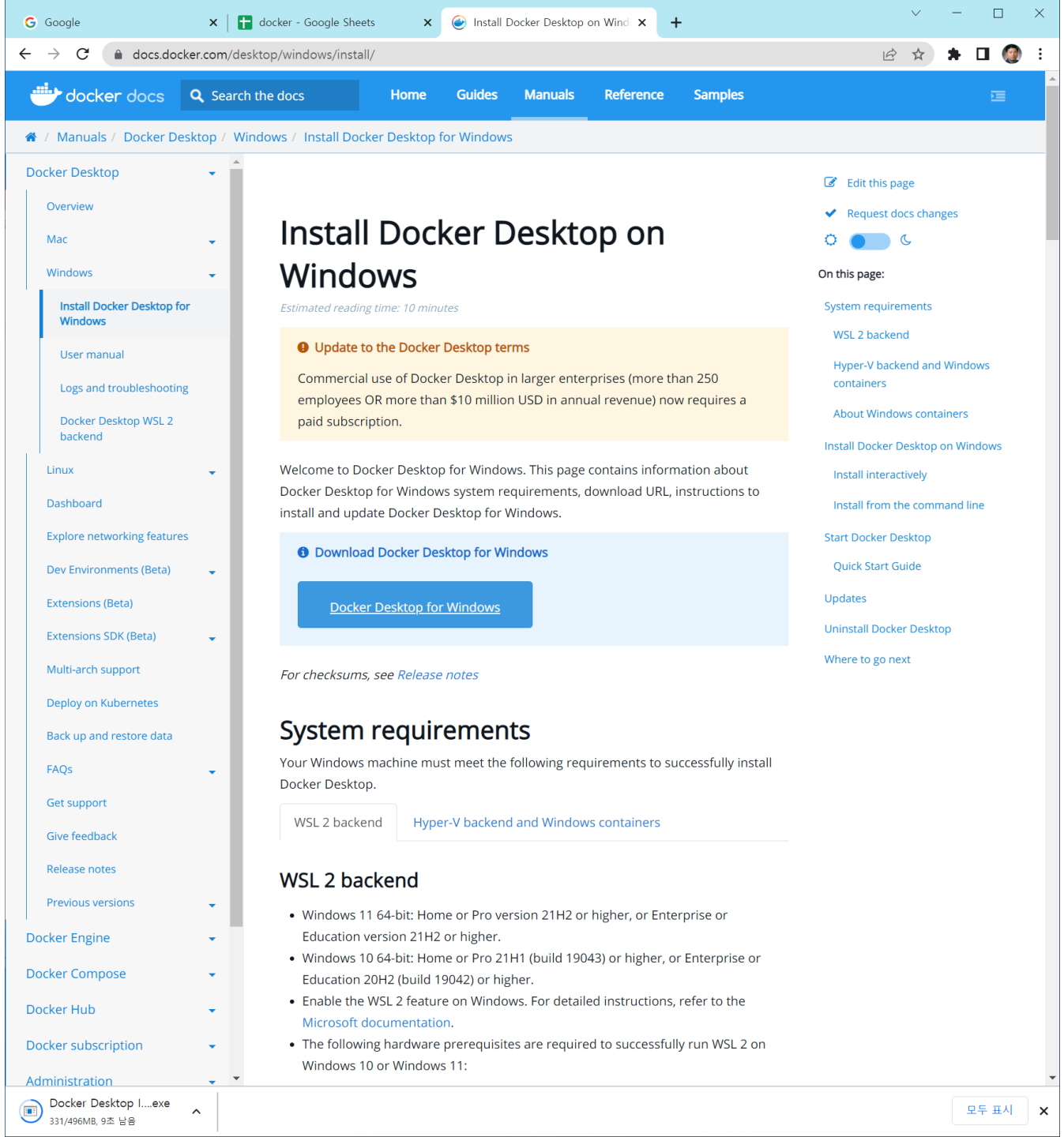
명령어를 입력하고 패스워드를 넣으면 리눅스 배포판(ubuntu/centos/..)에 따라 자동으로 최신버전 의 도커를 설치합니다.

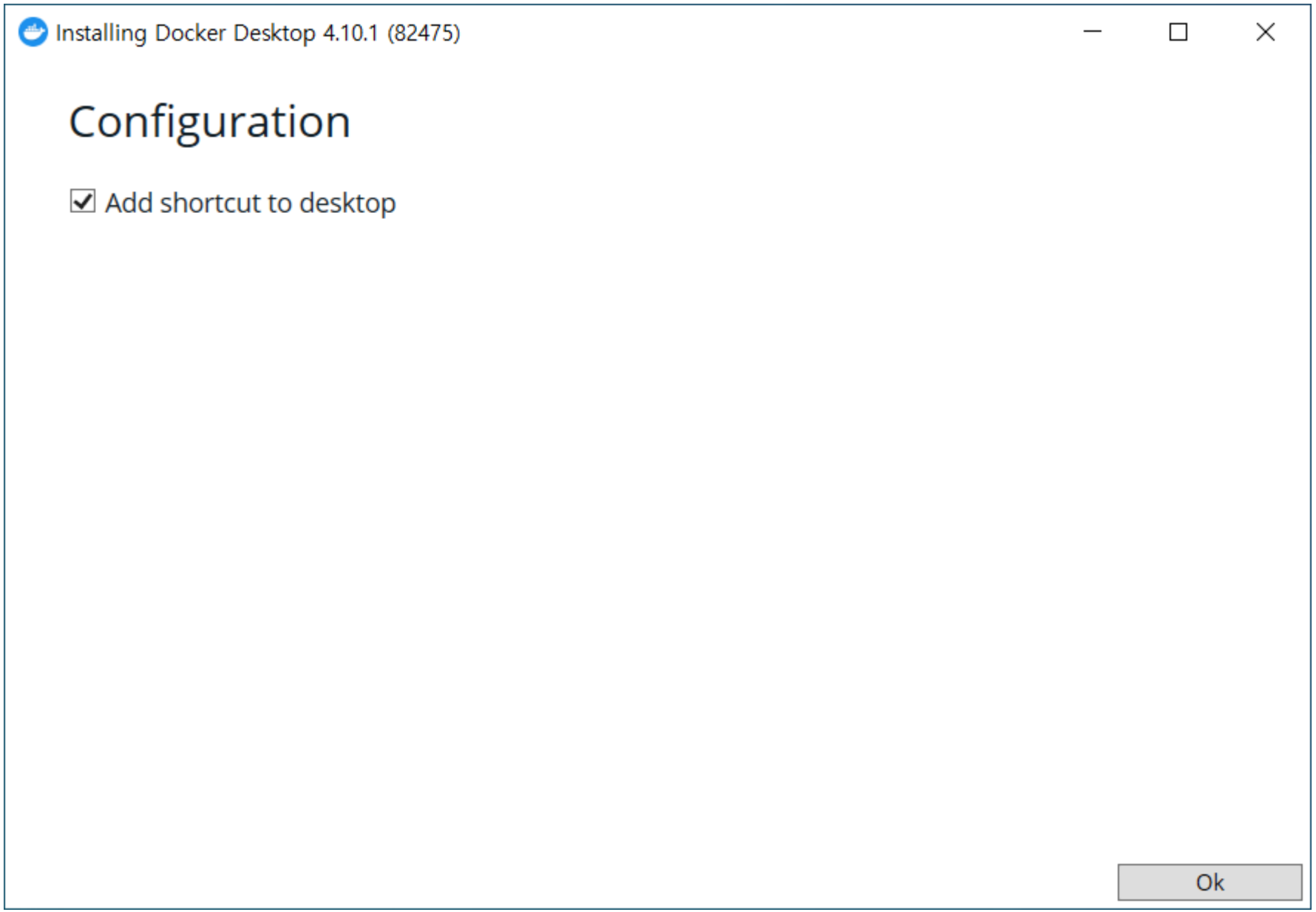
https://docs.docker.com/desktop/linux/install/

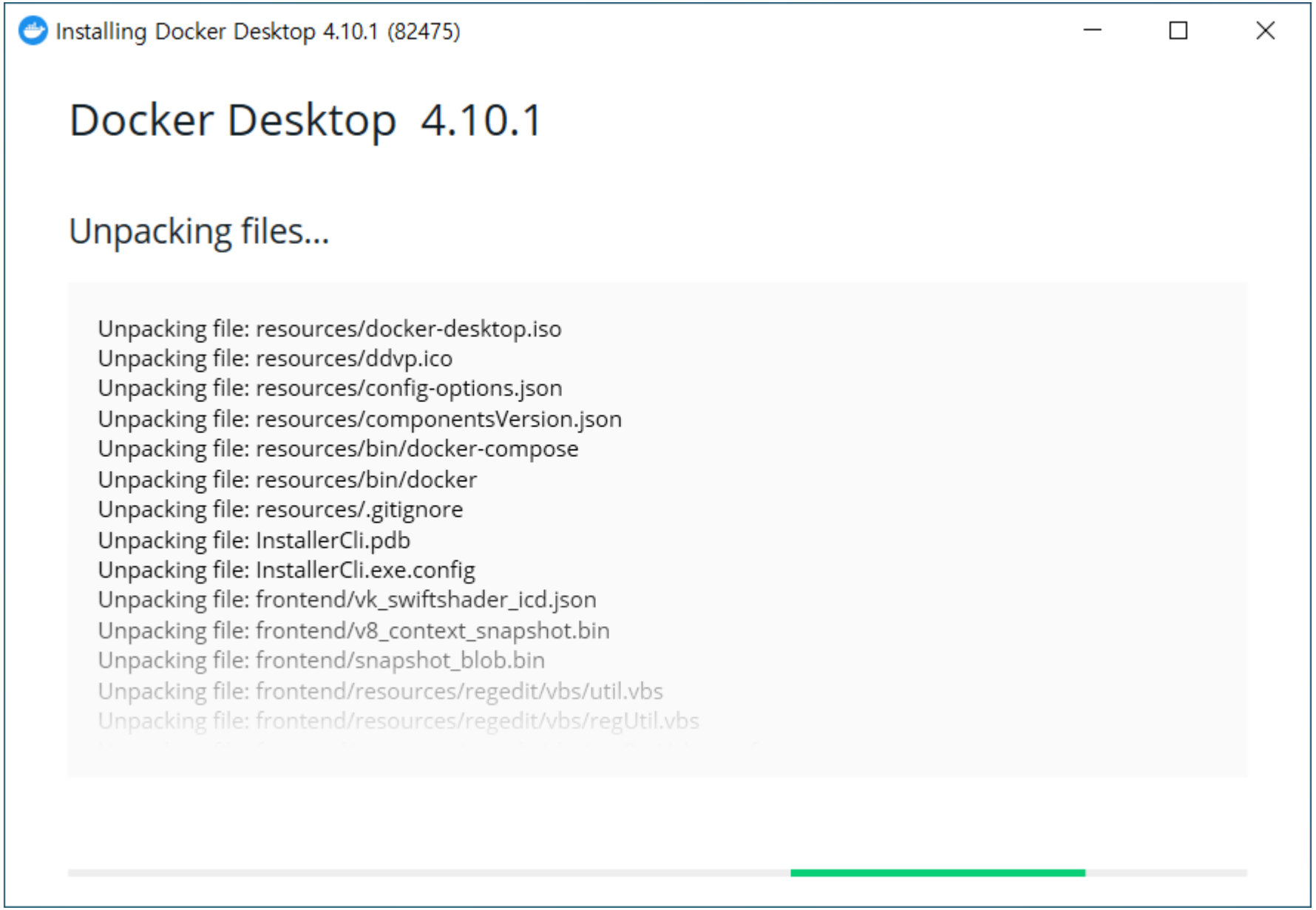
도커 설치 (Docker for Windows)

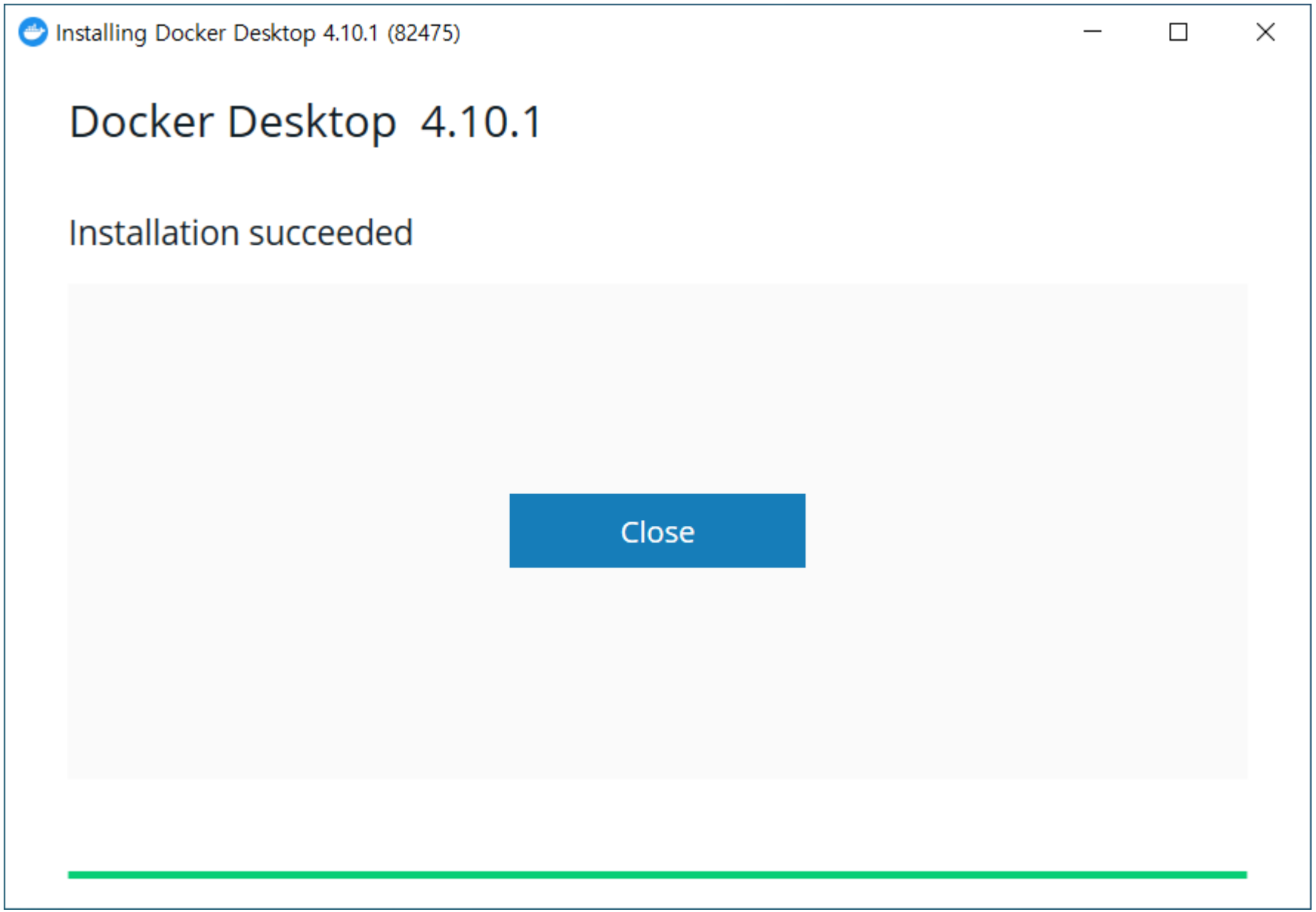


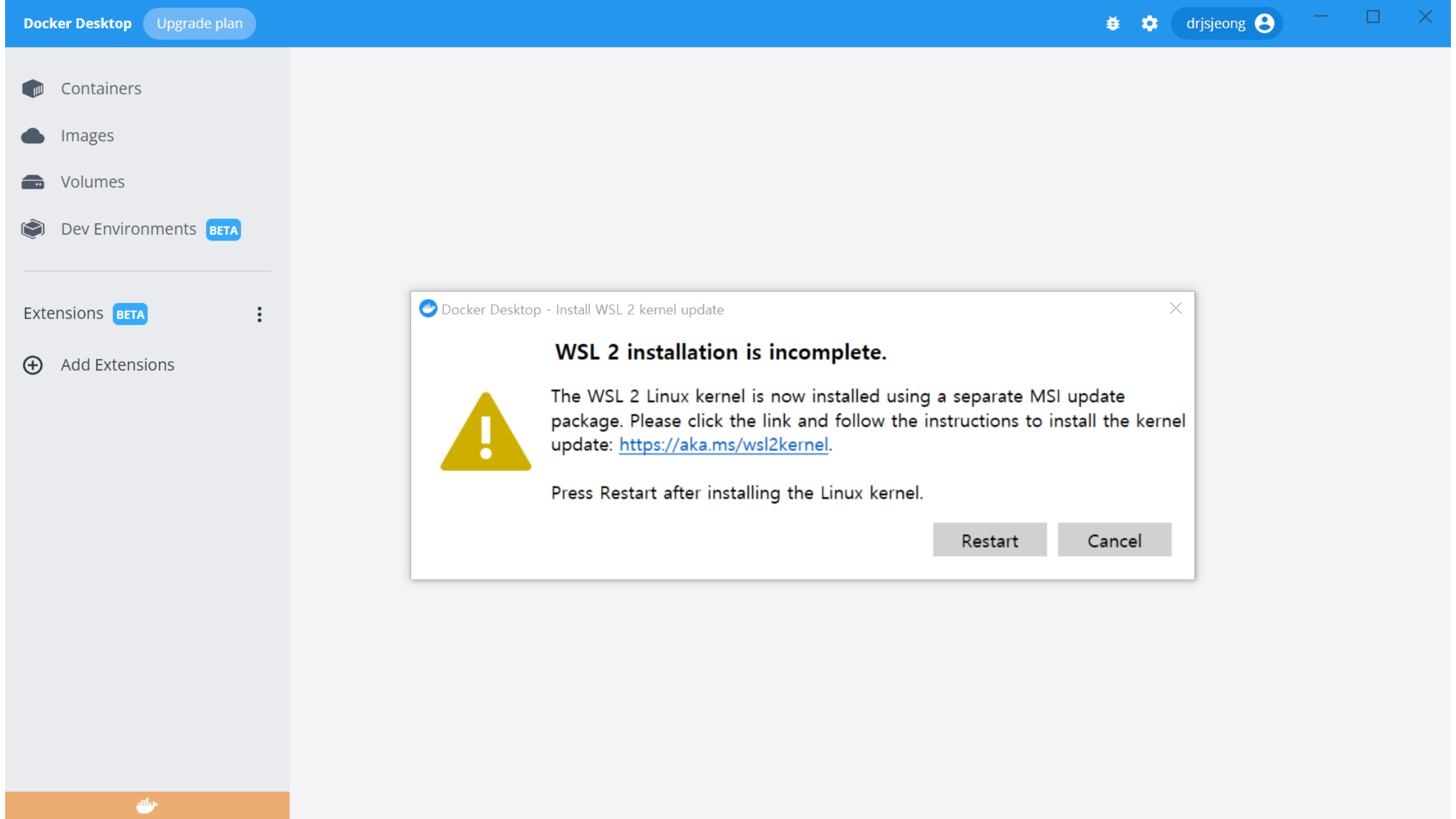
https://docs.docker.com/desktop/windows/install/


Google
docker - Google Sheets
Install Docker Desktop on Wind
docs.docker.com/desktop/windows/install/
docker docs
Search the docs
Home
Guides
Manuals
Reference
Samples
Manuals / Docker Desktop / Windows / Install Docker Desktop for Windows
Docker Desktop
Overview
Mac
Windows
Install Docker Desktop for Windows
User manual
Logs and troubleshooting
Docker Desktop WSL 2 backend
Linux
Dashboard
Explore networking features
Dev Environments (Beta)
Extensions (Beta)
Extensions SDK (Beta)
Multi-arch support
Deploy on Kubernetes
Back up and restore data
FAQs
Get support
Give feedback
Release notes
Previous versions
Docker Engine
Docker Compose
Docker Hub
Docker subscription
Administration
Edit this page
Request docs changes
On this page:
System requirements
WSL 2 backend
Hyper-V backend and Windows containers
About Windows containers
Install Docker Desktop on Windows
Install interactively
Install from the command line
Start Docker Desktop
Quick Start Guide
Updates
Uninstall Docker Desktop
Where to go next
Install Docker Desktop on Windows
Estimated reading time: 10 minutes
Update to the Docker Desktop terms
Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) now requires a paid subscription.
Welcome to Docker Desktop for Windows. This page contains information about Docker Desktop for Windows system requirements, download URL, instructions to install and update Docker Desktop for Windows.
Download Docker Desktop for Windows
Docker Desktop for Windows
For checksums, see Release notes
System requirements
Your Windows machine must meet the following requirements to successfully install Docker Desktop.
WSL 2 backend
Hyper-V backend and Windows containers
WSL 2 backend
Windows 11 64-bit: Home or Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher.
Windows 10 64-bit: Home or Pro 21H1 (build 19043) or higher, or Enterprise or Education 20H2 (build 19042) or higher.
Enable the WSL 2 feature on Windows. For detailed instructions, refer to the Microsoft documentation.
The following hardware prerequisites are required to successfully run WSL 2 on Windows 10 or Windows 11:
Docker Desktop I....exe
331/496MB, 9초 남음
모두 표시

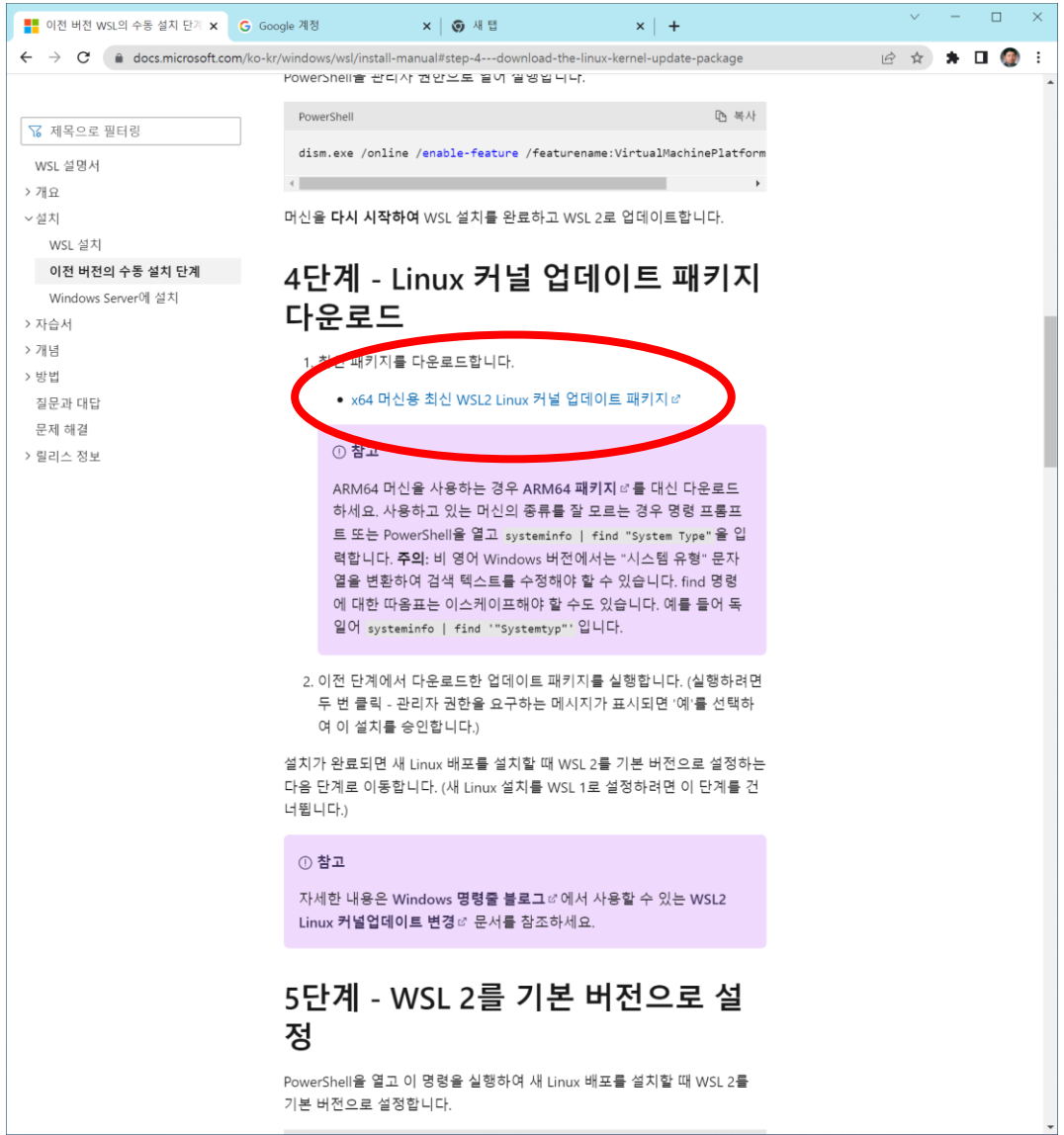

Installing Docker Desktop 4.10.1 (82475)
Configuration
Add shortcut to desktop
Ok

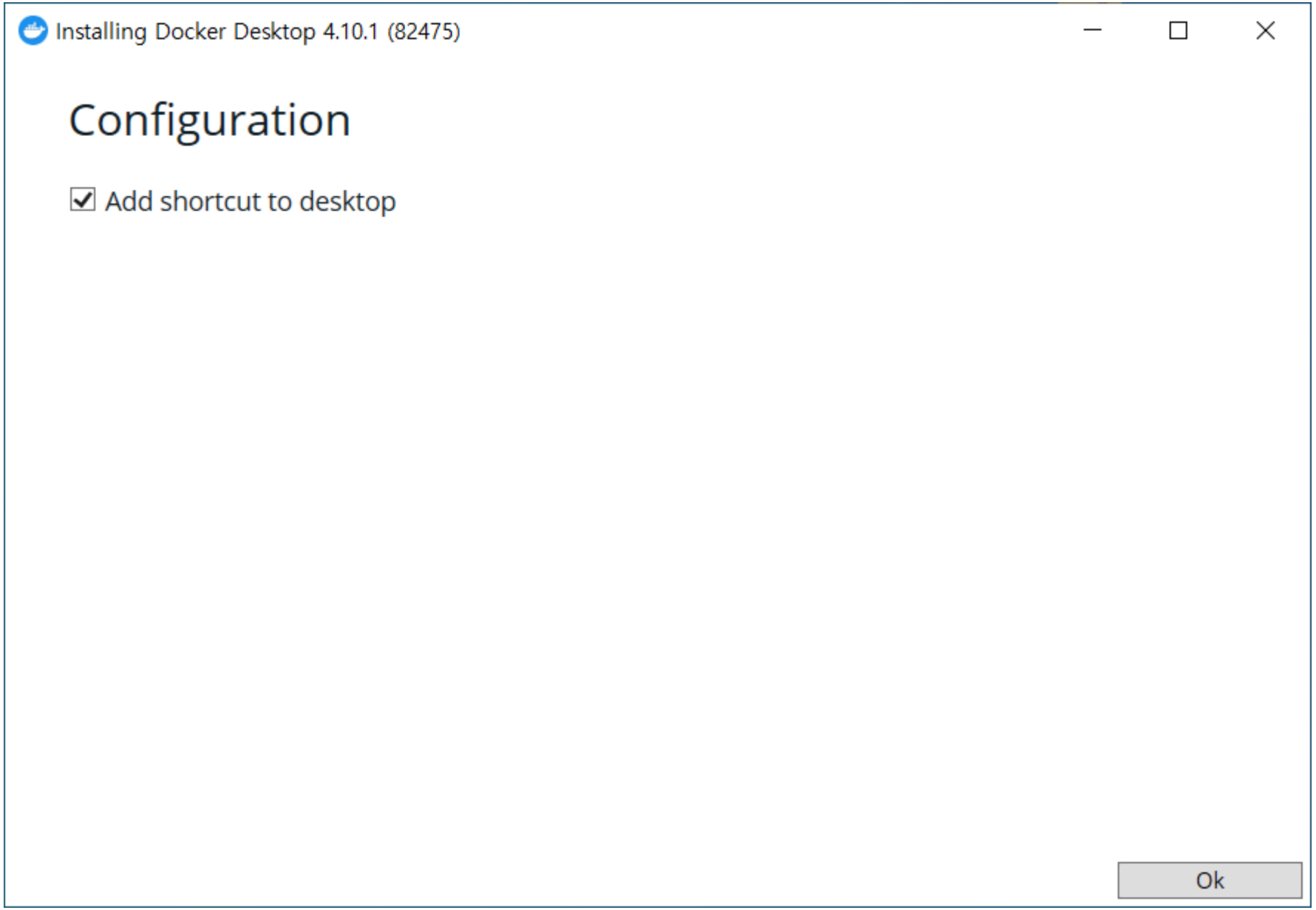

Installing Docker Desktop 4.10.1 (82475)
Docker Desktop 4.10.1
Unpacking files...
Unpacking file: resources/docker-desktop.iso
Unpacking file: resources/ddvp.ico
Unpacking file: resources/config-options.json
Unpacking file: resources/componentsVersion.json
Unpacking file: resources/bin/docker-compose
Unpacking file: resources/bin/docker
Unpacking file: resources/.gitignore
Unpacking file: InstallerCli.pdb
Unpacking file: InstallerCli.exe.config
Unpacking file: frontend/vk_swiftshader_icd.json
Unpacking file: frontend/v8_context_snapshot.bin
Unpacking file: frontend/snapshot_blob.bin
Unpacking file: frontend/resources/regedit/vbs/util.vbs
Unpacking file: frontend/resources/regedit/vbs/regUtil.vbs

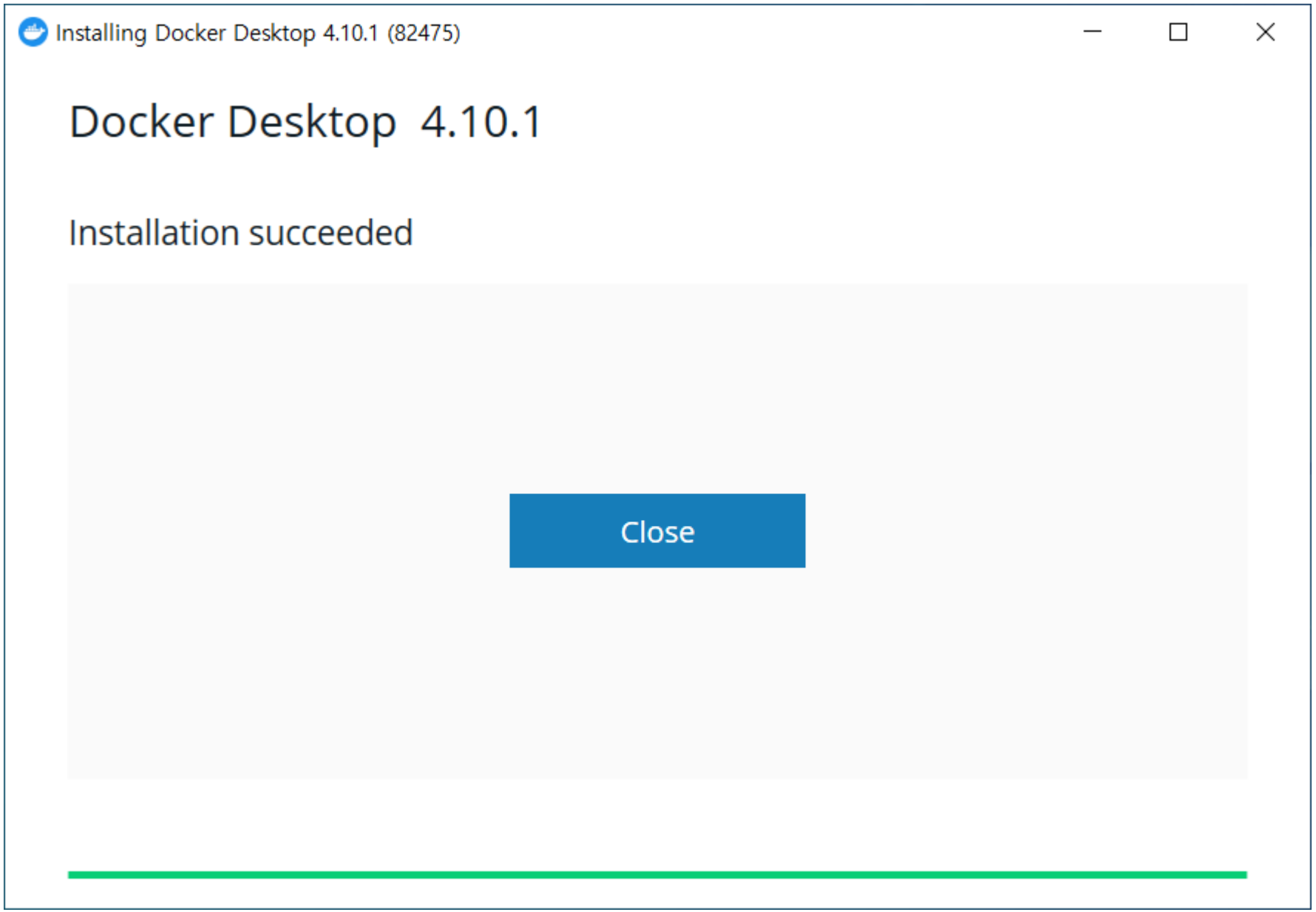

Installing Docker Desktop 4.10.1 (82475)
Docker Desktop 4.10.1
Installation succeeded
Close


Docker Desktop
Upgrade plan
drjsjeong
Containers
Images
Volumes
Dev Environments
BETA
Extensions
BETA
Add Extensions
Docker Desktop - Install WSL 2 kernel update
WSL 2 installation is incomplete.
The WSL 2 Linux kernel is now installed using a separate MSI update package. Please click the link and follow the instructions to install the kernel update: https://aka.ms/wsl2kernel.
Press Restart after installing the Linux kernel.
Restart
Cancel

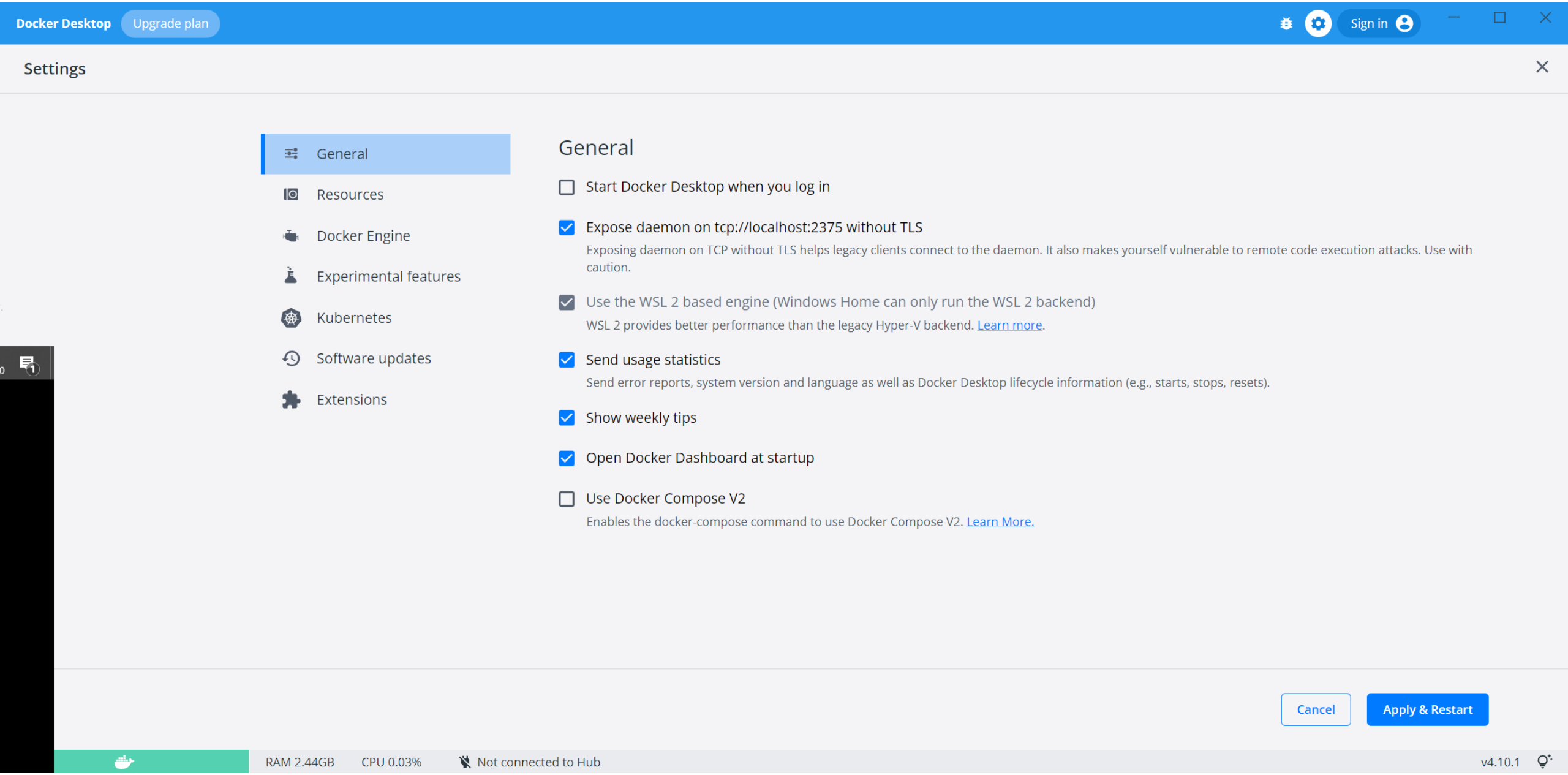
https://docs.microsoft.com/ko-kr/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package




Installing Docker Desktop 4.10.1 (82475)
Configuration
Add shortcut to desktop
Ok


Installing Docker Desktop 4.10.1 (82475)
Docker Desktop 4.10.1
Installation succeeded
Close

Docker for Windows는 Windows container를 지원하지만 여기서 다루지 않습니다.


Docker Desktop
Upgrade plan
Sign in
Settings
General
Resources
Docker Engine
Experimental features
Kubernetes
Software updates
Extensions
General
Start Docker Desktop when you log in
Expose daemon on tcp://localhost:2375 without TLS
Exposing daemon on TCP without TLS helps legacy clients connect to the daemon. It also makes yourself vulnerable to remote code execution attacks. Use with caution.
Use the WSL 2 based engine (Windows Home can only run the WSL 2 backend)
WSL 2 provides better performance than the legacy Hyper-V backend. Learn more.
Send usage statistics
Send error reports, system version and language as well as Docker Desktop lifecycle information (e.g., starts, stops, resets).
Show weekly tips
Open Docker Dashboard at startup
Use Docker Compose V2
Enables the docker-compose command to use Docker Compose V2. Learn More.
Cancel
Apply & Restart
RAM 2.44GB
CPU 0.03%
Not connected to Hub
v4.10.1

도커 설치 확인

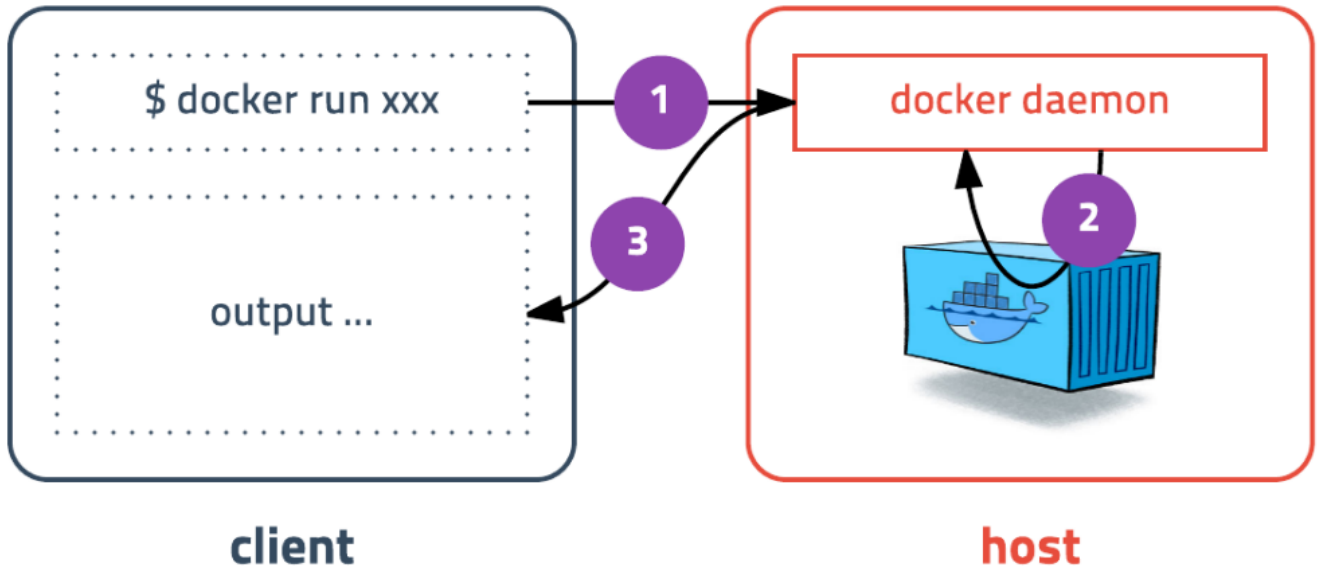
```
$ docker version

Client:
 Version:    17.12.0-ce
 API version:    1.35
 Go version:    go1.9.2
 Git commit:    c97c6d6
 Built:    Wed Dec 27 20:03:51 2017
 OS/Arch:    darwin/amd64

Server:
 Engine:
  Version:    17.12.0-ce
  API version:    1.35 (minimum version 1.12)
  Go version:    go1.9.2
  Git commit:    c97c6d6
  Built:    Wed Dec 27 20:12:29 2017
  OS/Arch:    linux/amd64
  Experimental:    true
```

도커 컨테이너의 실행

client(command) - server(daemon)



도커 컨테이너 실행

run command

```
docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]
```

옵션	설명
-d	detached mode 흔히 말하는 백그라운드 모드
-p	호스트와 컨테이너의 포트를 연결 (포워딩)
-v	호스트와 컨테이너의 디렉토리를 연결 (마운트)
-e	컨테이너 내에서 사용할 환경변수 설정
--name	컨테이너 이름 설정
--rm	프로세스 종료시 컨테이너 자동 제거
-it	-i와 -t를 동시에 사용한 것으로 터미널 입력을 위한 옵션
--network	네트워크 연결

https://docs.docker.com/engine/reference/commandline/docker/

도커 컨테이너 실행

ubuntu 16.04

```
docker run ubuntu:16.04
```

`run명령어`를 사용하면 사용할 이미지가 저장되어 있는지 확인하고 없다면 다운로드(`pull`)를 한 후 컨테이너를 생성(`create`)하고 시작(`start`) 합니다.

컨테이너는 정상적으로 실행됐지만 뭘 하라고 명령어를 전달하지 않았기 때문에 컨테이너는 생성되자마자 종료됩니다. 컨테이너는 프로세스이기 때문에 실행중인 프로세스가 없으면 컨테이너는 종료됩니다.

도커 컨테이너 실행

/bin/sh

```
docker run --rm -it ubuntu:16.04 /bin/sh
```

컨테이너 내부에 들어가기 위해 `sh`을 실행하고 키보드 입력을 위해 `-it`(== -i -t) 옵션을 줍니다.
추가적으로 프로세스가 종료되면 컨테이너가 자동으로 삭제되도록 `--rm` 옵션도 추가하였습니다.

`--rm`옵션이 없다면 컨테이너 종료되더라도 삭제되지 않고 남아있습니다.

도커 컨테이너 실행

CentOS

```
docker run --rm -it centos:7 /bin/sh
```

도커는 다양한 리눅스 배포판을 실행할 수 있습니다. 공통점은 모두 동일한 커널kernel을 사용한다는 점입니다.

Ubuntu 또는 CentOS에 포함된 다양한 기본기능이 필요 없는 경우, Alpine이라는 초소형(약 5MB) 이미지를 사용할 수도 있습니다.

도커 컨테이너 실행

Web Application

간단한 웹 애플리케이션을 컨테이너로 생성해봅니다.

```
docker run -d -p 4567:4567 subicura/docker-workshop-app:1
```

`detached mode(백그라운드 모드)`로 실행하기 위해 `-d` 옵션을 추가하고 `-p` 옵션을 추가하여 컨테이너의 포트를 호스트의 포트로 연결하였습니다.

브라우저를 열고 `localhost:4567`에 접속하면 컨테이너 아이디를 확인 할 수 있습니다.

`curl` 명령어로 확인해볼까요?

```
# mac
curl localhost:4567
# windows
docker run --rm byrnedo/alpine-curl docker.for.win.localhost:4567
```

도커 컨테이너 실행

Redis

redis는 메모리기반의 다양한 기능을 가진 스토리지입니다.

```
docker run --name=redis -d -p 1234:6379 redis
```

`telnet` 프로그램으로 테스트해봅니다.

```
# mac
docker run --rm -it mikesplain/telnet docker.for.mac.localhost 1234
# windows
docker run --rm -it mikesplain/telnet docker.for.win.localhost 1234

set hello world
+OK
get hello
$5
world
quit
```

도커 컨테이너 실행

MySQL

가장 유명한 데이터베이스 중 하나입니다. docker hub mysql을 검색해서 어떤 옵션(환경변수)이 있는지 확인해 봅니다.

```
docker run -d -p 3306:3306 \
  -e MYSQL_ALLOW_EMPTY_PASSWORD=true \
  --name mysql \
  mysql:5.7
```

mysql에 접속하여 database를 만듭니다.

```
docker exec -it mysql mysql
create database wp CHARACTER SET utf8;
grant all privileges on wp.* to wp@'%' identified by 'wp';
flush privileges;
quit
```

도커 컨테이너 목록확인

ps

실행중인 컨테이너 목록을 확인하는 명령어는 다음과 같습니다.

```
docker ps
```

중지된 컨테이너도 확인하려면 `-a` 옵션을 붙입니다.

```
docker ps -a
```

도커 컨테이너 중지하기

stop

실행중인 컨테이너를 중지하는 명령어는 다음과 같습니다.

```
docker stop [OPTIONS] CONTAINER [CONTAINER...]
```

실행중인 컨테이너를 하나 또는 여러개 (띄어쓰기로 구분) 중지할 수 있습니다.

도커 컨테이너 제거하기

rm

종료된 컨테이너를 완전히 제거하는 명령어는 다음과 같습니다.

```
docker rm [OPTIONS] CONTAINER [CONTAINER...]
```

종료 명령어도 옵션은 특별한게 없습니다. 종료된 컨테이너를 하나 또는 여러개 삭제할 수 있습니다.

`mysql`과 `wordpress`를 제외하고 모두 삭제해주세요.

도커 컨테이너 로그보기

logs

컨테이너가 정상적으로 동작하는지 확인하는 좋은 방법은 로그를 확인하는 것 입니다. 로그를 확인하는 방법은 다음과 같습니다.

```
docker logs [OPTIONS] CONTAINER
```

기본 옵션과 `-f`, `--tail` 옵션을 살펴봅니다.

도커 컨테이너 이미지 다운로드하기

pull

이미지를 다운로드하는 명령어는 다음과 같습니다.

```
docker pull [OPTIONS] NAME[:TAG|@DIGEST]
```

ubuntu:14.04를 다운받아보겠습니다.

```
docker pull ubuntu:14.04
```

run명령어를 입력하면 이미지가 없을 때 자동으로 다운받으니 pull명령어를 언제 쓰는지 궁금할 수 있는데 pull은 최신버전으로 다시 다운 받습니다. 같은 태그지만 이미지가 업데이트 된 경우는 pull 명령어를 통해 새로 다운받을 수 있습니다.

도커 컨테이너 이미지 삭제하기

rmi

이미지를 삭제하는 방법은 다음과 같습니다.

```
docker rmi [OPTIONS] IMAGE [IMAGE...]
```

images명령어를 통해 얻은 이미지 목록에서 이미지 ID를 입력하면 삭제가 됩니다. 단, 컨테이너가 실행중인 이미지는 삭제되지 않습니다. 컨테이너는 이미지들의 레이어를 기반으로 실행중이므로 당연히 삭제할 수 없습니다.

네트워크 만들기

network create

도커 컨테이너끼리 통신을 할 수 있는 가상 네트워크를 만듭니다.

```
docker network create [OPTIONS] NETWORK
```

`app-network` 라는 이름으로 wordpress와 mysql이 통신할 네트워크를 만듭니다.

```
docker network create app-network
```

네트워크 만들기

network connect

기존에 생성된 컨테이너에 네트워크를 추가합니다.

```
docker network connect [OPTIONS] NETWORK CONTAINER
```

만들어 놓은 mysql에 네트워크를 추가합니다.

```
docker network connect app-network mysql
```

네트워크 만들기

run with network

워드프레스를 app-network에 속하게 생성하고 mysql을 IP가 아닌 `mysql`로 바로 접근합니다.

```
docker run -d -p 8080:80 \
  --network=app-network \
  -e WORDPRESS_DB_HOST=mysql \
  -e WORDPRESS_DB_NAME=wp \
  -e WORDPRESS_DB_USER=wp \
  -e WORDPRESS_DB_PASSWORD=wp \
  wordpress
```

같은 네트워크에 속해 있으면 상대 컨테이너의 이름을 DNS로 조회하여 바로 접근 할 수 있습니다. 하나의 컨테이너는 여러개의 network에 속할 수 있으며 `Docker Swarm` 같은 클러스터에서 편리하게 사용할 수 있습니다.

볼륨 마운트

volume mount (-v)

mysql을 삭제후에 다시 실행합니다.

```
docker stop mysql
docker rm mysql
docker run -d -p 3306:3306 \
  -e MYSQL_ALLOW_EMPTY_PASSWORD=true \
  --network=app-network \
  --name mysql \
  mysql:5.7
```

워드프레스를 접속하면 데이터베이스 오류가 발생합니다. 이전 데이터가 전부 초기화 되었습니다!
😨

```
localhost:8080
```

도커 Compose 설치 확인

Docker for Mac / Windows는 기본으로 같이 설치됩니다.

```
$ docker-compose version

docker-compose version 1.18.0, build 8dd22a9
docker-py version: 2.6.1
CPython version: 2.7.12
OpenSSL version: OpenSSL 1.0.2j  26 Sep 2016
```

Linux는 다음 명령어로 설치합니다.

```
sudo curl -L https://github.com/docker/compose/releases/download/1.18.0/docker-compose-`uname -s`-`u
sudo chmod +x /usr/local/bin/docker-compose
```

도커 Compose

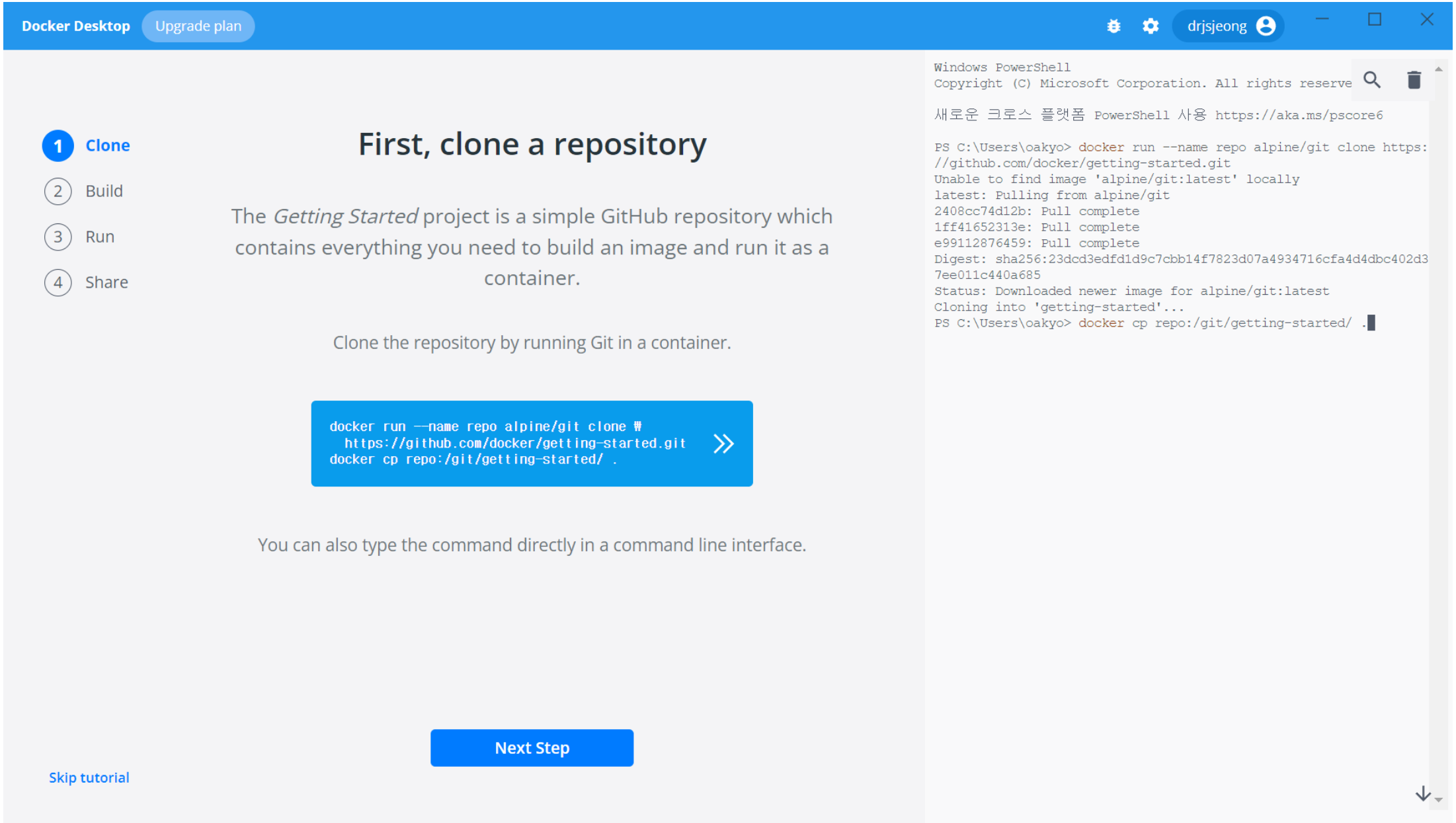
실제 운영환경에선 명령어를 입력하는 대신 대부분 Docker Compose를 사용합니다.

Docker Compose는 docker의 거의 모든 기능을 사용할 수 있습니다.

실습 파일

https://github.com/JSJeong-me/docker

First, clone a repository



Docker Desktop Upgrade plan drjsjeong

- Clone
- 2 Build
- 3 Run
- 4 Share

Now, build the image

A Docker image is a private file system just for your container.
It provides all the files and code your container needs.

```
cd getting-started
docker build -t docker101tutorial .
```

Next Step

Skip tutorial

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserve

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Users\oakyo> docker run --name repo alpine/git clone https:
//github.com/docker/getting-started.git
Unable to find image 'alpine/git:latest' locally
latest: Pulling from alpine/git
2408cc74d12b: Pull complete
1ff41652313e: Pull complete
e99112876459: Pull complete
Digest: sha256:23dcd3edfd1d9c7cbb14f7823d07a4934716cfa4d4dbc402d3
7ee011c440a685
Status: Downloaded newer image for alpine/git:latest
Cloning into 'getting-started'...
PS C:\Users\oakyo> docker cp repo:/git/getting-started/ .cd getti
ng-started
"docker cp" requires exactly 2 arguments.
See 'docker cp --help'.

Usage:  docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-
        docker cp [OPTIONS] SRC_PATH|- CONTAINER:DEST_PATH

Copy files/folders between a container and the local filesystem
PS C:\Users\oakyo> docker build -t docker101tutorial .
[+] Building 0.1s (2/2) FINISHED
 => [internal] load build definition from Dockerfile        0.1s
 => => transferring dockerfile: 2B                          0.0s
 => [internal] load .dockerignore                           0.1s
 => => transferring context: 2B                             0.0s
failed to solve with frontend dockerfile.v0: failed to read docke
rfile: open /var/lib/docker/tmp/buildkit-mount1874297261/Dockerfi
le: no such file or directory
PS C:\Users\oakyo>
```

Docker Desktop Upgrade plan drjsjeong

- Clone
- Build
- Run
- 4 Share

Now save and share your image

Save and share your image on Docker Hub to enable other users to easily download and run the image on any destination machine.

```
docker tag docker101tutorial drjsjeong/docker101tutorial
docker push drjsjeong/docker101tutorial
```

See what you've saved on Hub

Done

Skip tutorial

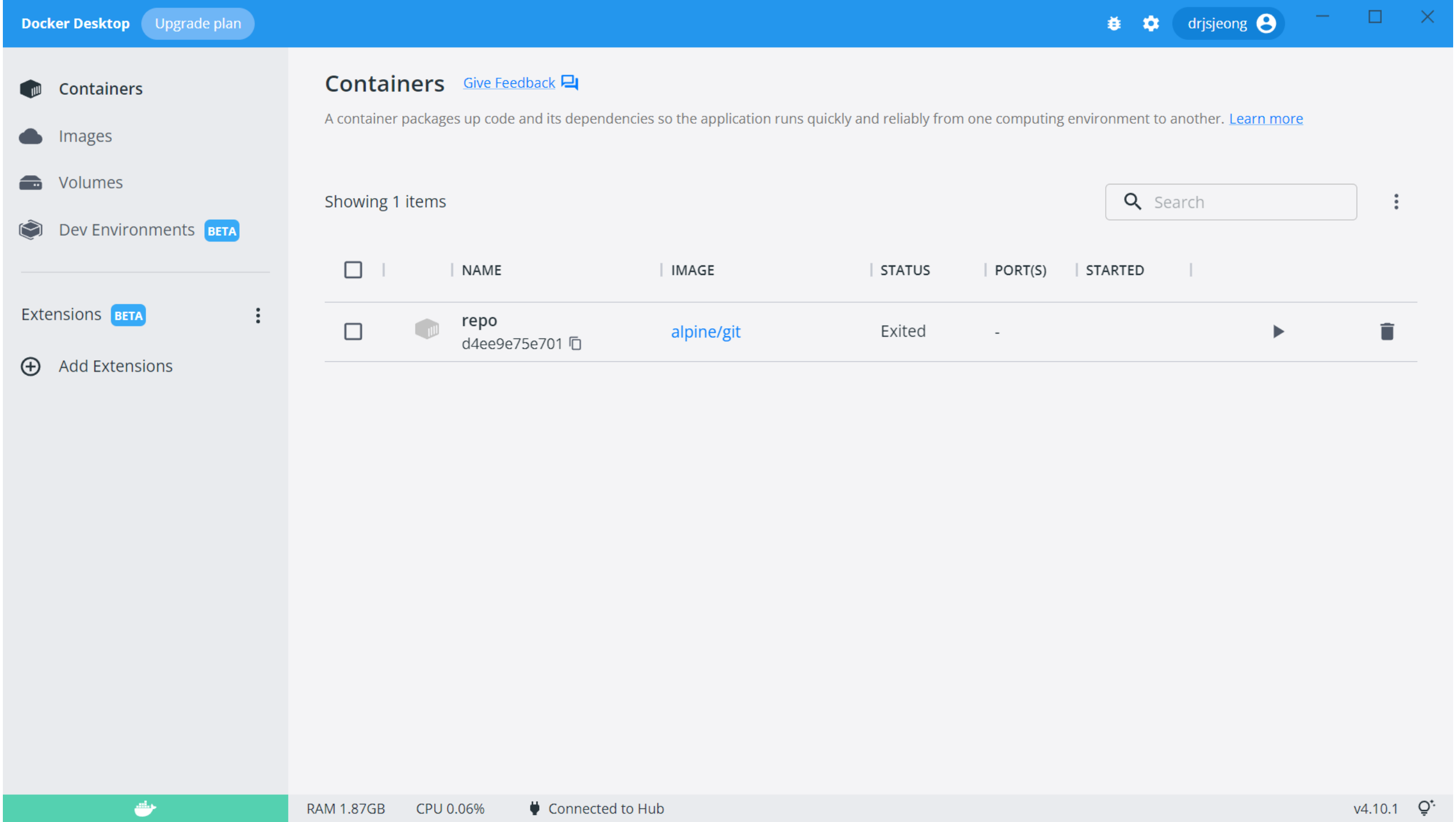
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserve

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Users\oakyo> docker run --name repo alpine/git clone https:
//github.com/docker/getting-started.git
Unable to find image 'alpine/git:latest' locally
latest: Pulling from alpine/git
2408cc74d12b: Pull complete
1ff41652313e: Pull complete
e99112876459: Pull complete
Digest: sha256:23dcd3edfd1d9c7cbb14f7823d07a4934716cfa4d4dbc402d3
7ee011c440a685
Status: Downloaded newer image for alpine/git:latest
Cloning into 'getting-started'...
PS C:\Users\oakyo> docker cp repo:/git/getting-started/ .cd getti
ng-started
"docker cp" requires exactly 2 arguments.
See 'docker cp --help'.

Usage:  docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-
        docker cp [OPTIONS] SRC_PATH|- CONTAINER:DEST_PATH

Copy files/folders between a container and the local filesystem
PS C:\Users\oakyo> docker build -t docker101tutorial .
[+] Building 0.1s (2/2) FINISHED
 => [internal] load build definition from Dockerfile        0.1s
 => => transferring dockerfile: 2B                          0.0s
 => [internal] load .dockerignore                           0.1s
 => => transferring context: 2B                             0.0s
failed to solve with frontend dockerfile.v0: failed to read docke
rfile: open /var/lib/docker/tmp/buildkit-mount1874297261/Dockerfi
le: no such file or directory
PS C:\Users\oakyo> docker run -d -p 80:80 --name docker-tutorial
docker101tutorial
Unable to find image 'docker101tutorial:latest' locally
docker: Error response from daemon: pull access denied for docker
101tutorial, repository does not exist or may require 'docker log
in': denied: requested access to the resource is denied.
See 'docker run --help'.
PS C:\Users\oakyo>
```


Docker Desktop
Upgrade plan
drjsjeong
Containers
Images
Volumes
Dev Environments
BETA
Extensions
BETA
Add Extensions
Containers
Give Feedback
A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more
Showing 1 items
Search
NAME
IMAGE
STATUS
PORT(S)
STARTED
repo
d4ee9e75e701
alpine/git
Exited
-
RAM 1.87GB
CPU 0.06%
Connected to Hub
v4.10.1

정 준 수 / Ph.D （ jsjeong@hansung.ac.kr ）

- 前) 삼성전자 연구원
- 前) 삼성의료원 (삼성생명과학연구소)
- 前) 삼성SDS (정보기술연구소)
- 現) (사)한국인공지능협회, AI, 머신러닝 강의
- 現) 한국소프트웨어산업협회, AI, 머신러닝 강의
- 現) 서울디지털재단, AI 자문위원
- 現) 한성대학교 교수(겸)
- 전문분야: Computer Vision, 머신러닝(ML), RPA
- https://github.com/JSJeong-me/