# **Docker 기본 개념**

1. **Docker를 왜 배우는 걸까?**

### **✅ 현업에서는 Docker를 왜 이렇게 많이 쓰는 걸까?**

Docker를 쓰는 이유에는 여러가지 장점이 있지만 그 중에서 핵심 장점 딱 1가지만 기억하자.

**이식성 :** 특정 프로그램을 다른 곳으로 쉽게 옮겨서 설치 및 실행할 수 있는 특성

이 핵심 장점을 예시를 통해 이해해보자.

친구는 컴퓨터에 MySQL을 아무 에러 없이 잘 깔았다. 그런데 내 컴퓨터에 MySQL을 깔려고 하니 이상하게 에러가 뜨는 것이다. 분명 친구가 설치한 방식대로 똑같이 했는데 제대로 안 깔릴 때가 있다. 지우고 다시 깔아봐도 계속해서 똑같은 에러가 뜨기도 한다.

내 컴퓨터에만 MySQL이 안 깔리는 이유는 다양하다. 버전을 다른 걸 설치했거나, 운영체제(Window, Mac OS 등)가 다르거나, 내 컴퓨터에 깔려있는 다른 프로그램(ex. 보안 프로그램)과 충돌이 일어났거나 와 같은 다양한 이유로 프로그램이 정상적으로 설치되지 않는다. 그리고 설치 과정이 복잡하다면 새 컴퓨터를 사서 MySQL을 설치할 때마다 번거롭고 귀찮다고 느껴진다.

이걸 깔끔하게 해결하기 위해 나타난 툴이 Docker이다. Docker를 사용하면 명령어 한 줄로 어떤 컴퓨터에든 MySQL을 에러 없이 설치하고 실행할 수 있게 된다.

뿐만 아니라 Docker를 사용하면 아래와 같은 장점이 있다.

* 매번 귀찮은 설치 과정을 일일이 거치지 않아도 된다.
* 항상 일관되게 프로그램을 설치할 수 있다. (버전, 환경 설정, 옵션, 운영 체제 등)
* 각 프로그램이 독립적인 환경에서 실행되기 때문에 프로그램 간에 서로 충돌이 일어나지 않는다.

1. IP와 Port의 개념

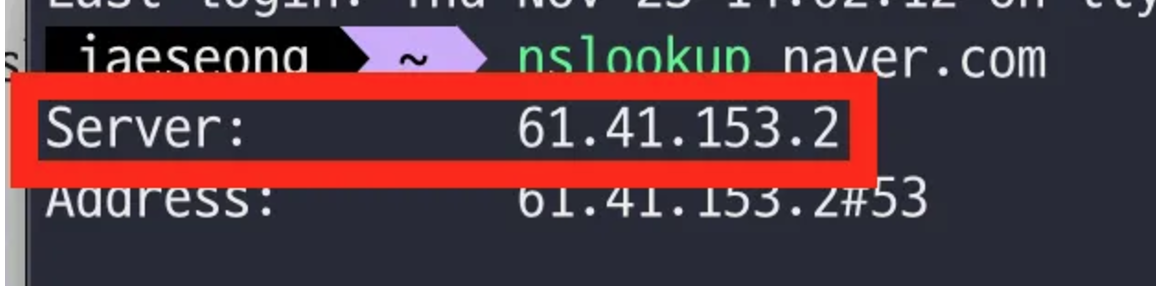
### **✅ IP의 개념**

💡 한 줄 요약 : 네트워크 상에서의 **특정 컴퓨터**를 가리키는 주소

아래와 같은 값이 IP 주소이다.

13.250.15.132

IP는 특정 컴퓨터의 주소를 가리킨다. 예를 들면, naver.com이라는 서비스도 IP 주소를 가지고 있다. 저 IP 주소는 네이버가 운영하고 있는 컴퓨터의 주소이기도 하다.

****

### **✅ Port의 개념**

💡 한 줄 요약 : 한 컴퓨터 내에서 실행되고 있는 **특정 프로그램의 주소**이다.

아래와 같은 값에서 :3000 부분이 포트 번호를 의미한다.

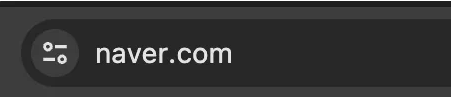
13.250.15.132:3000

한 컴퓨터 내에서 여러 프로그램이 실행되고 있을 것이다. 내가 지금 사용하고 있는 노트북만 보더라도 크롬, 카카오톡, 슬랙, VSCode 등 여러가지 프로그램이 동시에 실행되고 있다. 실제 서버를 운영하는 컴퓨터도 동일하다. 하나의 컴퓨터에서 여러가지의 프로그램이 실행된다.

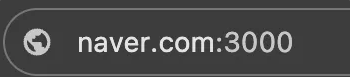
그럼 외부에서 특정 컴퓨터 내부에 있는 Spring Boot라는 서버에 통신을 하고 싶다고 가정하자. 하지만 외부에서 IP 주소만 알아서는 실행되고 있는 여러 프로그램 중 어떤 프로그램과 통신을 해야 할 지 알 수가 없다. 그래서 특정 서버와 통신을 할 때는 IP 주소와 서버가 실행되고 있는 포트 번호까지 알고 있어야 한다.

### **✅ 브라우저 창에 포트 번호를 입력하지 않는 이유?**

위 설명에서 분명 특정 서버와 통신하기 위해서는 IP 주소와 포트 번호를 둘 다 알아야 된다고 했다. 도메인 주소를 통해서 알 수 있는 건 IP 주소 뿐이다. 그럼 포트 번호를 입력해주지도 않았는데 어떻게 정상적으로 통신을 한 걸까?



주소창에 도메인 주소를 입력해서 엔터를 누르면, 브라우저(크롬, 익스플로러 등)는 기본적으로 80번 포트로 통신을 보내게 설정되어 있다. 그래서 포트 번호를 입력해주지 않아도 통신이 잘 됐던 것이었다. 만약 80번 포트로 통신하고 싶지 않고, 3000번 포트로 통신하고 싶다면 아래와 같이 주소창에 입력해야 한다.

****

### 

### **✅ 잘 알려진 포트(well-known port)란 ?**

🧑🏻‍🦱 포트(Port)에는 잘 알려진 포트(well-known port)라는 개념이 있다. 이 개념에 대해 알아보자.

포트 번호는 0 ~ 65,535번까지 사용할 수 있다. 그 중에서 0 ~ 1023번까지의 포트 번호는 주요 통신을 위한 규약에 따라 이미 정해져 있다. 이렇게 규약을 통해 역할이 정해져있는 포트 번호를 보고 잘 알려진 포트(well-known port)라고 부른다.

규약으로 정해져 있는 포트 번호 중 자주 사용되는 포트 번호에 대해서만 알아보자.

* 22번 (SSH, Secure Shell Protocol) : 원격 접속을 위한 포트 번호
  + EC2 인스턴스에 연결할 때 22번 포트를 사용한다.
* 80번 (HTTP) : HTTP로 통신을 할 때 사용
* 443번 (HTTPS) : HTTPS로 통신을 할 때 사용

여기서 착각하면 안 되는 점은 위에서 정해놓은 규약을 꼭 지키지 않아도 된다. 즉, 규약으로 정해져 있는 포트 번호와 다르게 사용해도 된다는 뜻이다. 예를 들어, 특정 서버와 HTTP 통신을 할 때 80번 포트를 쓰지 않고 3000번(익스프레스) 포트나 8080번(스프링부트) 포트를 써도 상관 없다.

1. **Docker란 ? / 컨테이너(Container)란? / 이미지(Image)란?**

### **✅ Docker란 ?**

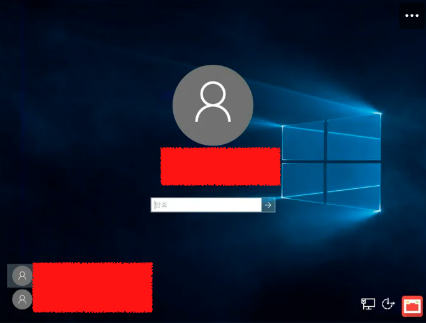
**컨테이너**를 사용하여 각각의 프로그램을 **분리된 환경**에서 실행 및 관리할 수 있는 툴이다.

👨🏻‍🏫 Docker의 정의를 보더라도 한 번에 와닿지 않을 것이다. 당연하다. Docker가 어떤 툴인지는 직접 사용해보고 경험해봐야 느낄 수 있다. 직접 사용해보고 경험해보는 게 Docker가 어떤 툴인지 제일 빠르게 파악하는 방법이다.

### **✅ 컨테이너(Container)란?**

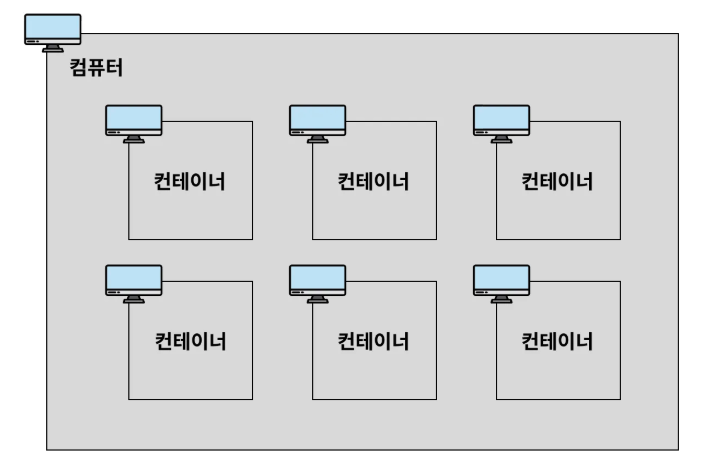
👨🏻‍🏫 Docker에서 **컨테이너(Container)**라는 개념은 아주 중요한 개념이다. 머릿속에 컨테이너(Container)가 어떤 개념인 지 대략적으로 떠올릴 수 있어야 한다. 그래야 Docker를 쉽게 배울 수 있다.

윈도우 환경을 사용해보면 하나의 컴퓨터에 여러 사용자로 나눠서 사용할 수 있게끔 구성되어 있다. 각 사용자의 환경에 들어가보면 독립적으로 구성되어 있어서 필요한 프로그램을 각 사용자 환경에 따로따로 설치해주어야 한다.



컨테이너도 이와 비슷한 개념이다. 하나의 컴퓨터 환경 내에서 **독립적인 컴퓨터 환경**을 구성해서, 각 환경에 프로그램을 별도로 설치할 수 있게 만든 개념이다. 하나의 컴퓨터 환경 내에서 여러개의 **미니 컴퓨터 환경**을 구성할 수 있는 형태이다. 여기서 얘기하는 **미니 컴퓨터**를 보고 Docker에서는 컨테이너(Container)라고 부른다.

(컨테이너(Container)를 이해할 때 머릿속의 이미지로 미니 컴퓨터를 떠올리면 이해하기 편하다.) MSA(작고, 독립적으로 배포 가능한 각각의 기능을 수행하는 서비스로 구성된 프레임워크라고 할 수 있다) 환경을 구성할수 있다.



여기서 ‘**컨테이너**’와 ‘**컨테이너를 포함하고 있는 컴퓨터**’를 구분하기 위해 컨테이너를 포함하고 있는 컴퓨터를 ‘**호스트(host) 컴퓨터**’라고 부른다.

### **✅ 컨테이너(Container)의 독립성**

위의 설명에서 컨테이너는 ‘독립적인 컴퓨터 환경’이라고 얘기했다. 구체적으로 어떤 것들이 독립적으로 관리되는 지 기억해두자.

* **디스크 (저장 공간)** : 각 컨테이너마다 서로 **각자의 저장 공간**을 가지고 있다. 일반적으로 A 컨테이너 내부에서 B 컨테이너 내부에 있는 파일에 접근할 수 없다.
* **네트워크 (IP, Port)** : 각 컨테이너마다 고유의 네트워크를 가지고 있다. 컨테이너는 각자의 IP 주소를 가지고 있다.

### **✅ 이미지(Image)란?**

****

닌텐도와 같은 게임기를 보면 여러가지 칩을 꽂아서 다양한 게임을 즐길 수 있게 되어 있다. Docker에서는 **닌텐도의 칩**과 같은 역할을 하는 개념이 이미지(Image)이다.

Node.js 기반의 Express.js 서버 프로젝트를 이미지로 만들었다고 가정해보자. 이 이미지를 Docker로 실행시키면 Express.js 서버 프로젝트가 컨테이너(Container) 환경에서 실행된다. 복잡한 설치 과정을 거칠 필요 없이 손쉽게 실행된다.

또 다른 예로, MySQL 서버를 이미지로 만들었다면, 이 이미지를 Docker로 실행시키는 순간 MySQL 서버가 컨테이너(Container) 환경에서 실행된다. MySQL을 일일이 설치할 필요없이 MySQL 데이터베이스를 사용할 수 있게 된다.

이미지(Image)는 **프로그램을 실행하는 데 필요한 설치 과정, 설정, 버전 정보 등을 포함**하고 있다. 즉, **프로그램을 실행하는 데 필요한 모든 것을 포함**하고 있다.

1. **Docker 설치 (Windows, Mac OS)**

### **✅ Docker 설치 (윈도우)**

Install Docker Desktop on Windows

Docker Desktop은 컨테이너화된 애플리케이션 및 마이크로서비스를 구축하고 공유할 수 있는 Mac, Linux, Windows 환경용 원클릭 설치 애플리케이션으로, 설치된 머신에서 컨테이너, 애플리케이션, 이미지를 관리할 수 있는 간단한 GUI를 제공합니다.

여기에서는 윈도우 환경에 Docker Desktop을 설치하고 간단한 컨테이너 어플리케이션을 실행하는 방법을 설명합니다.

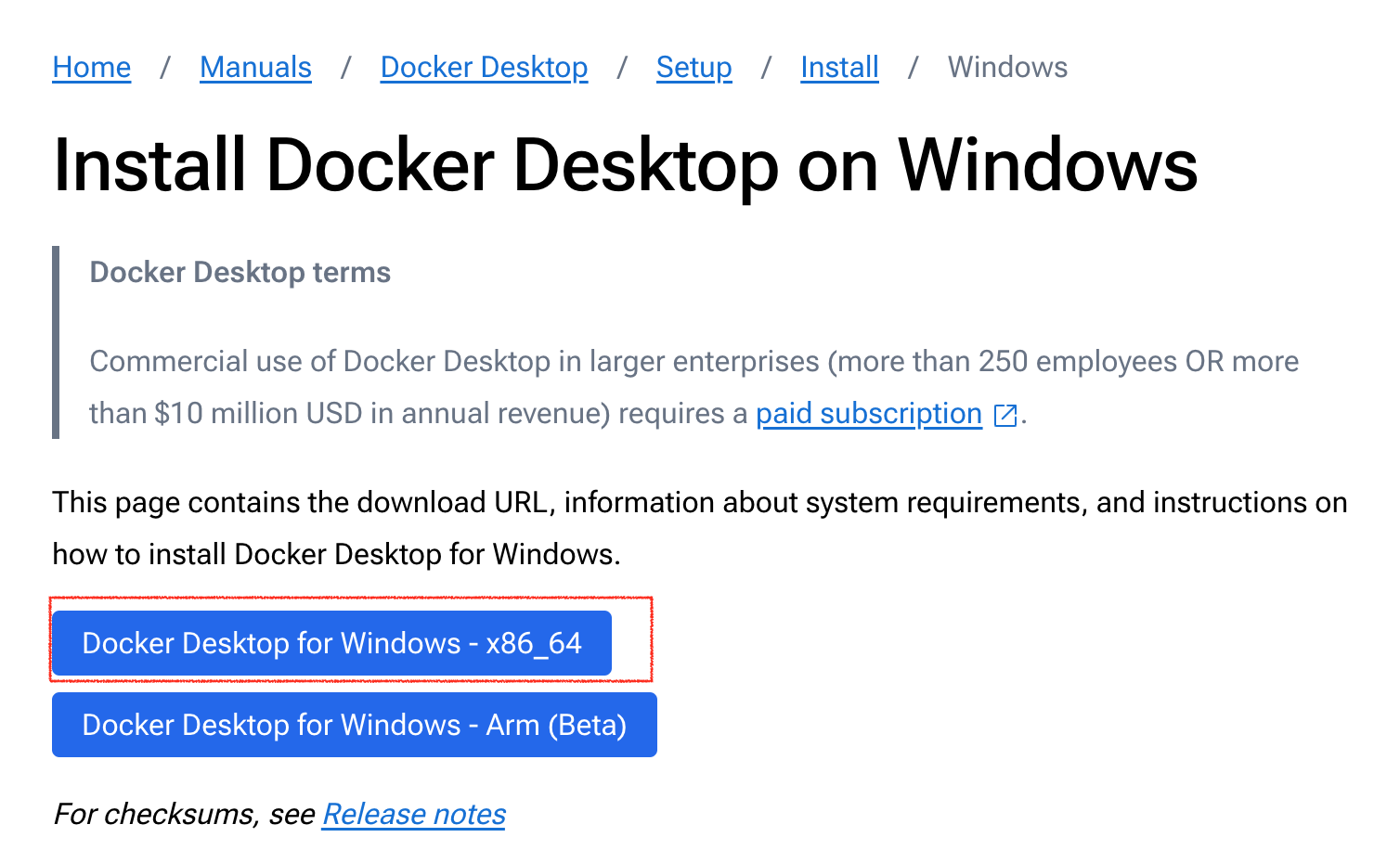
**명령어를 입력할 터미널은** PowerShell**을 활용해라.(관리자권한으로 실행)**

**실습에서 나오는 명령어 중 sudo가 포함된 명령어가 있다. PowerShell에서는 작동하지 않는 명령어이므로 sudo를 빼고 명령어를 입력하면 된다.**

우분투 환경에 도커를 설치하는 방법은 [우분투에 도커 설치](https://myanjini.tistory.com/entry/01-%EB%8F%84%EC%BB%A4-%EC%84%A4%EC%B9%98) 를 참고하세요.

01 설치 파일 다운로드

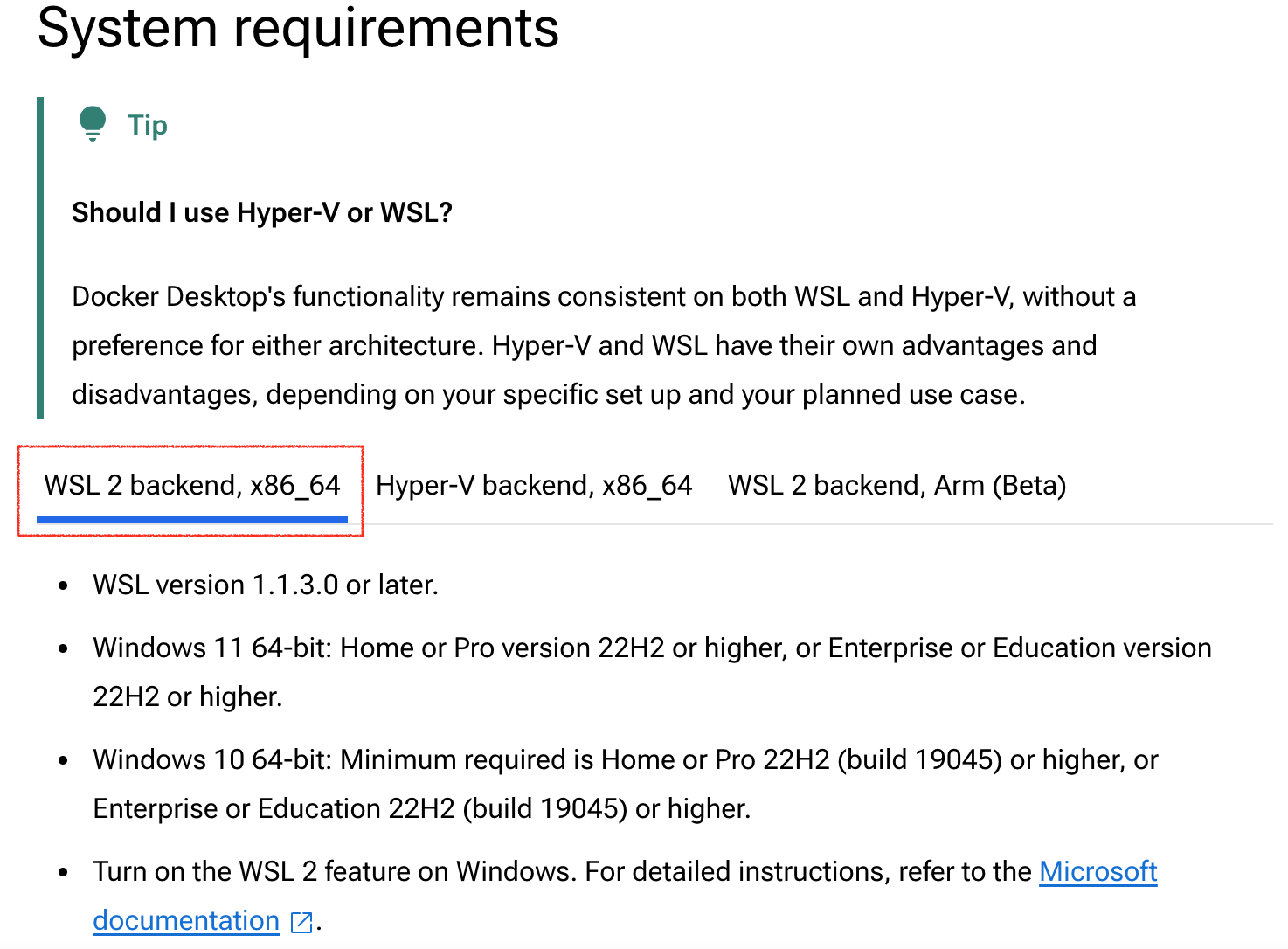
<https://docs.docker.com/desktop/install/windows-install/> 페이지에서 설치 파일을 다운로드 합니다.



02 시스템 요구사항 확인

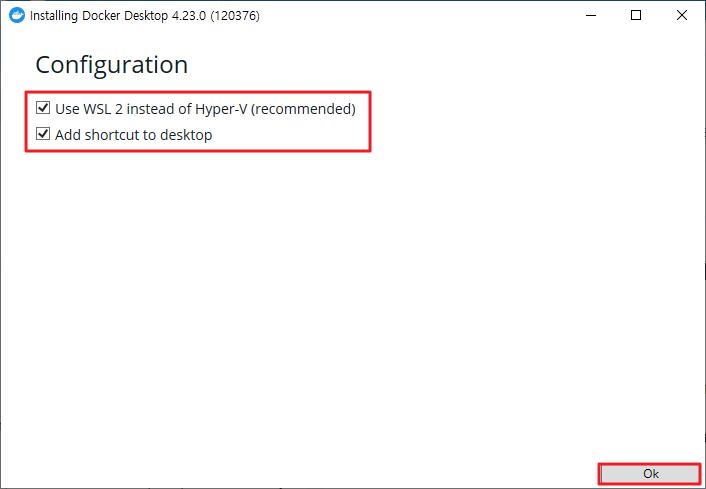
윈도우용 도커 데스크탑은 WSL2 또는 Hyper-V 기반으로 설치할 수 있습니다.

여기에서는 WSL2 기반으로 설치를 진행하므로 최소 시스템 요구사항을 만족하지 못 하는 경우 윈도운 업데이트를 먼저 진행합니다.



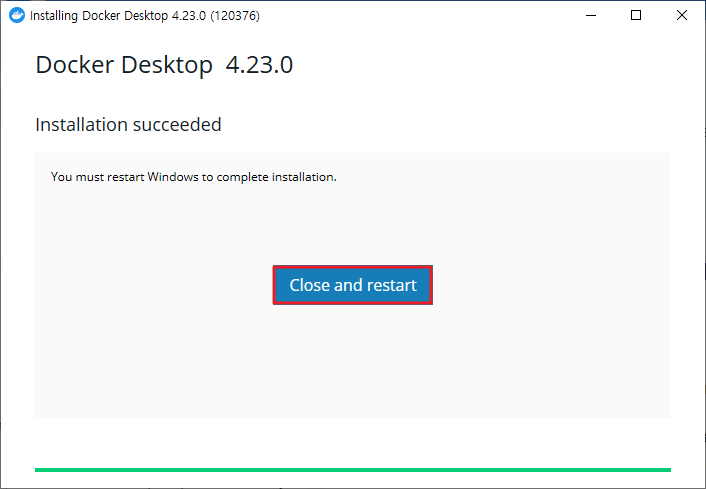
03 WSL2 기반으로 설치

01에서 다운로드 받은 설치 파일을 실행하고, Use WSL2 instead of Hyper-V 를 선택한 상태에서 설치를 진행합니다.



04 설치 완료 후 리부팅

설치가 완료되면 Close and restart 버튼을 클릭해 시스템을 리부팅합니다.

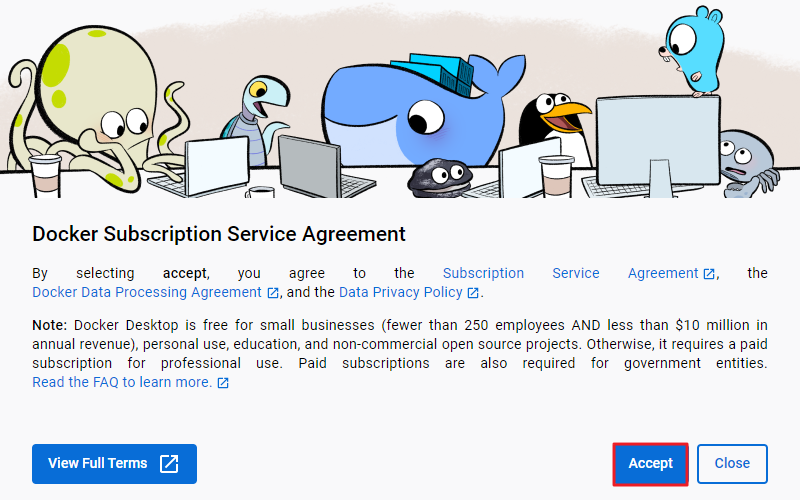


05 Docker Desktop 실행

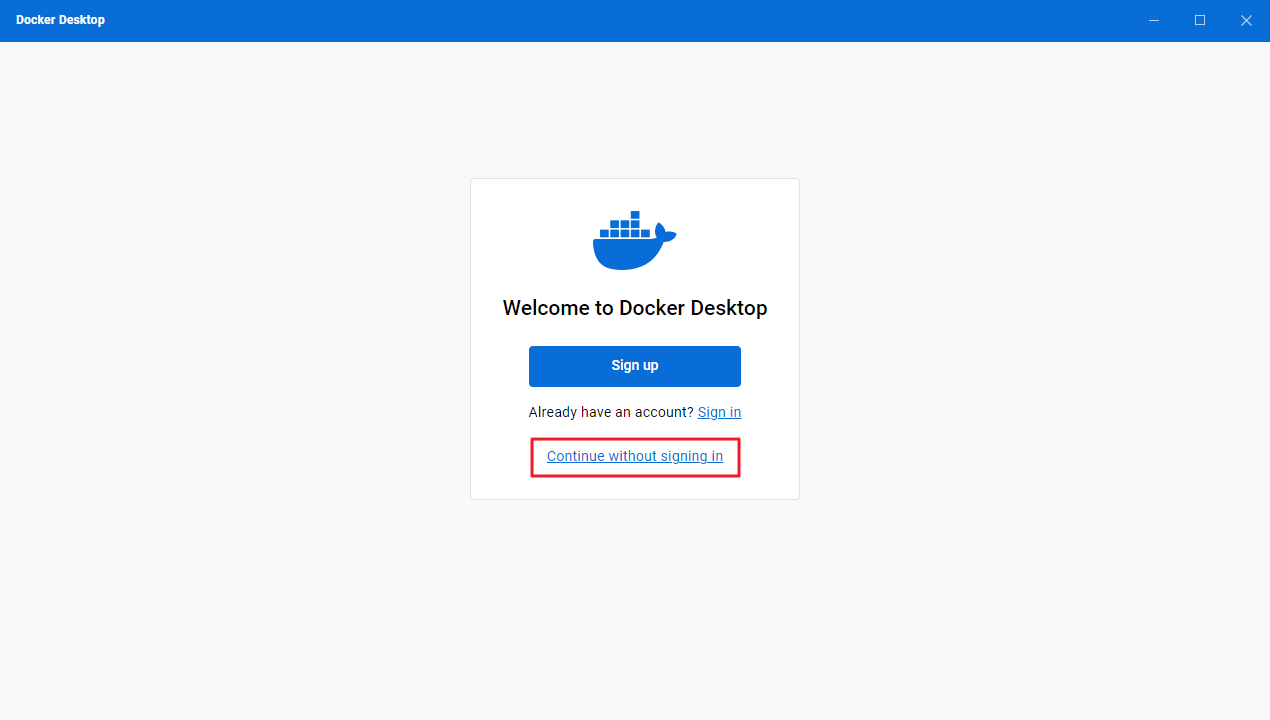
(리부팅 후 자동 시작되지 않는 경우) 바탕화면 또는 시작 메뉴에 Docker Desktop 링크를 클릭해 Docker Desktop을 실행합니다.

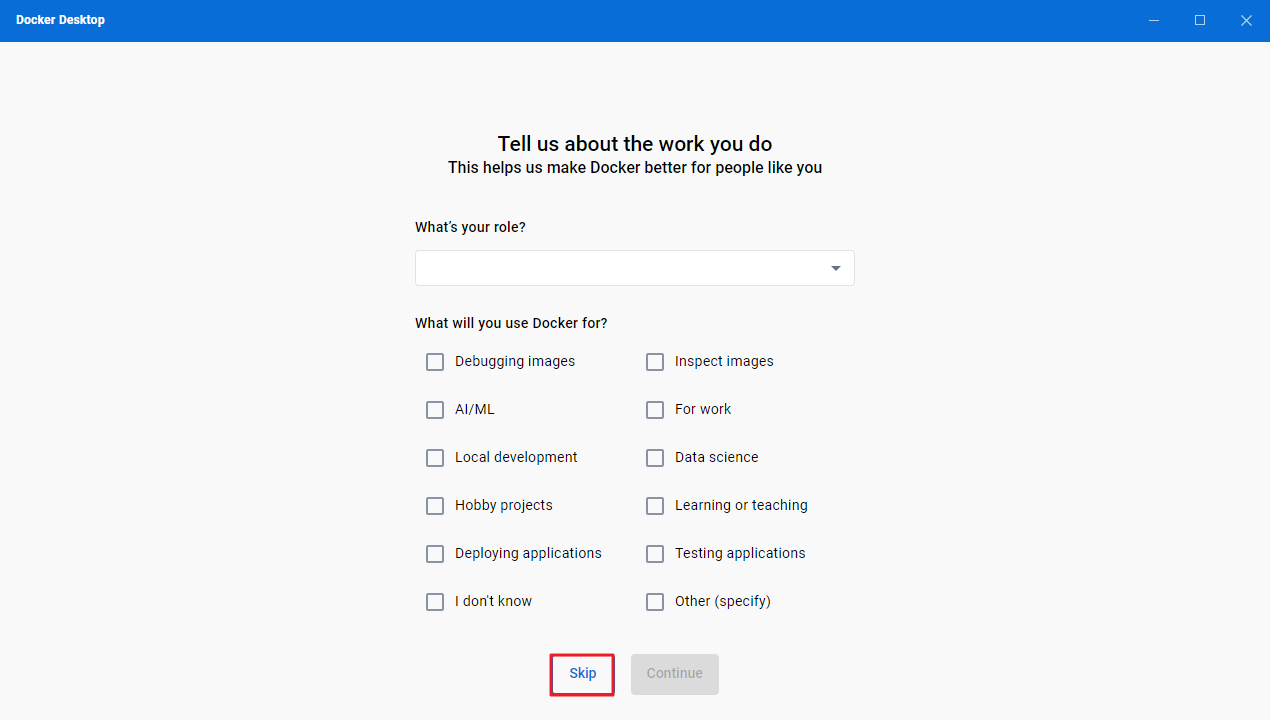


06 서비스 약관 동의



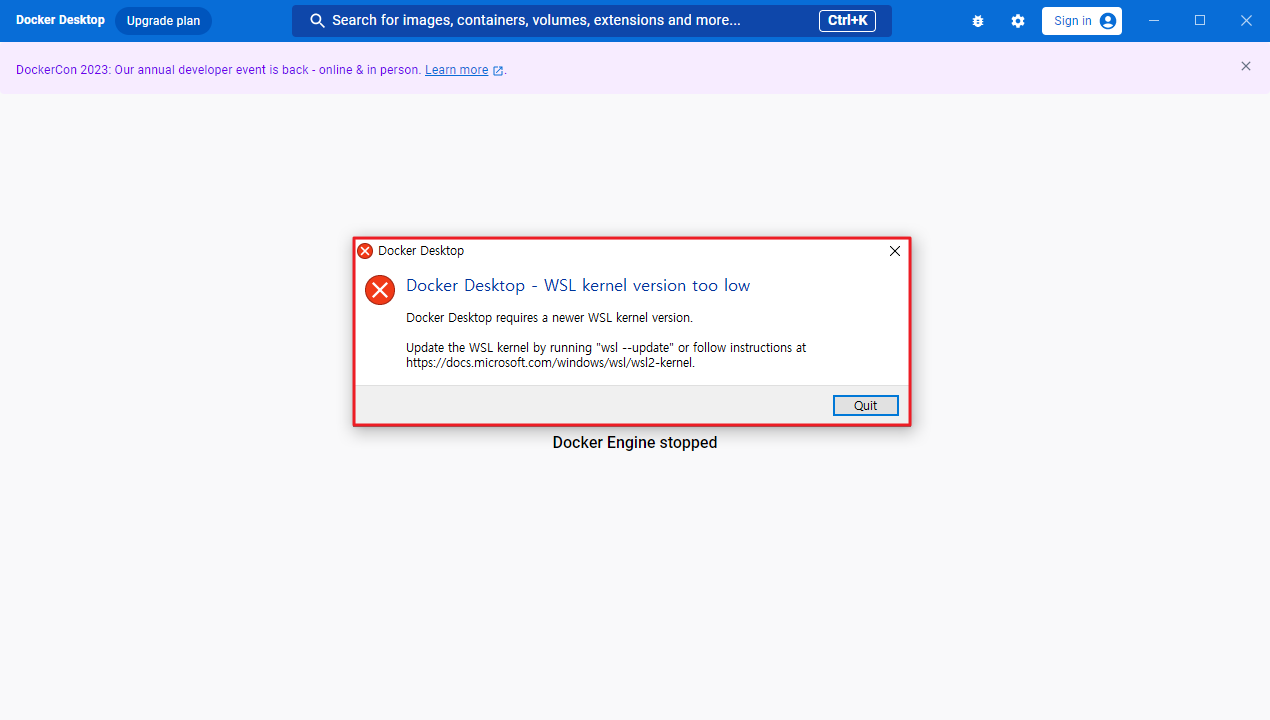
최초 실행 시 WSL 커널 업데이트 안내 창이 떴다가 화면 뒤로 감춰지면서 인증 정보 입력창과 이용 관련 설문 창이 뜰 수 있습니다. 해당 입력창을 스킵하고 다음으로 넘어가도 무관하겠습니다.





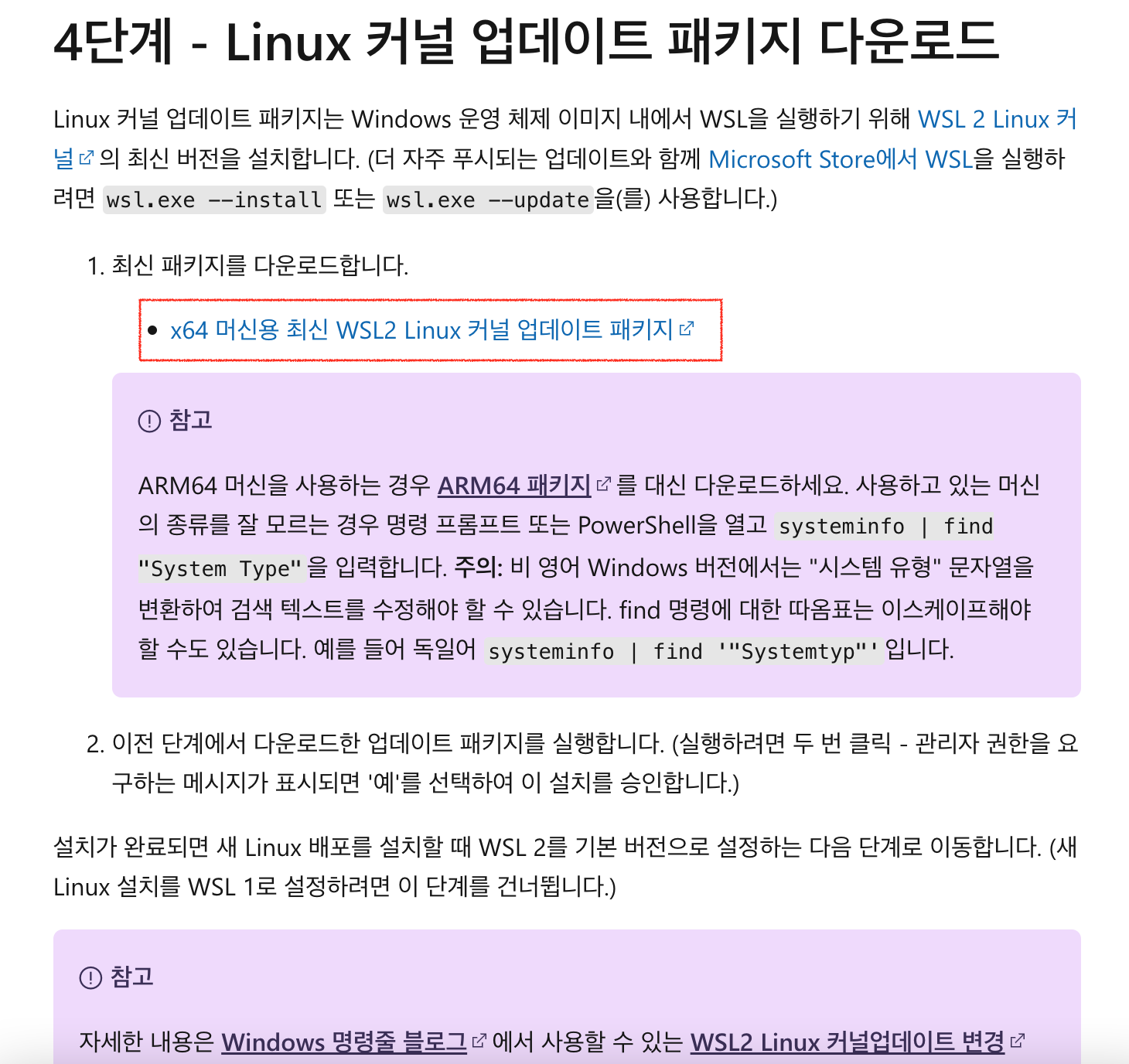
07 WSL 커널 업데이트

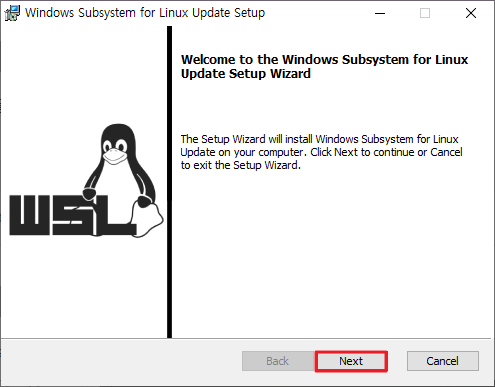
WSL 커널 업데이트 안내 창을 확인하고 <https://learn.microsoft.com/ko-kr/windows/wsl/wsl2-kernel> 사이트로 이동합니다.

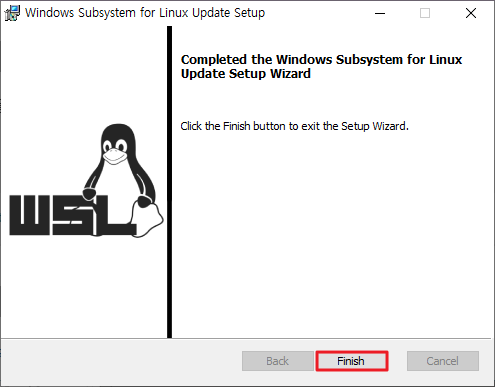


08 WSL2 Linux 커널 업데이트 패키지 다운로드 후 설치

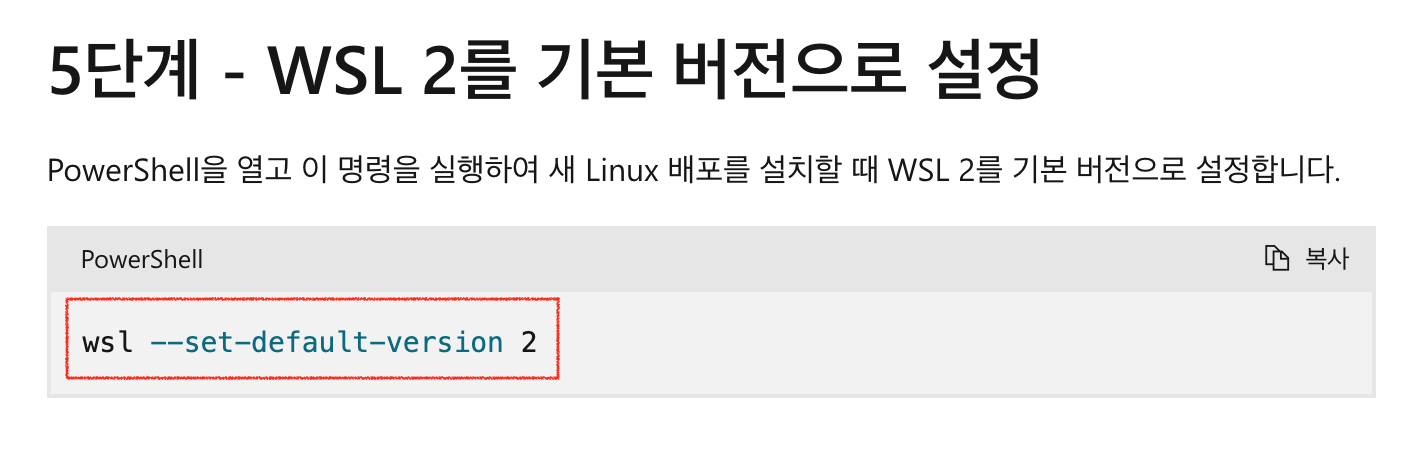
WSL2 Linux 커널 업데이트 패키지 다운로드 링크를 클릭해 설치 파일을 다운로드 후 설치합니다.







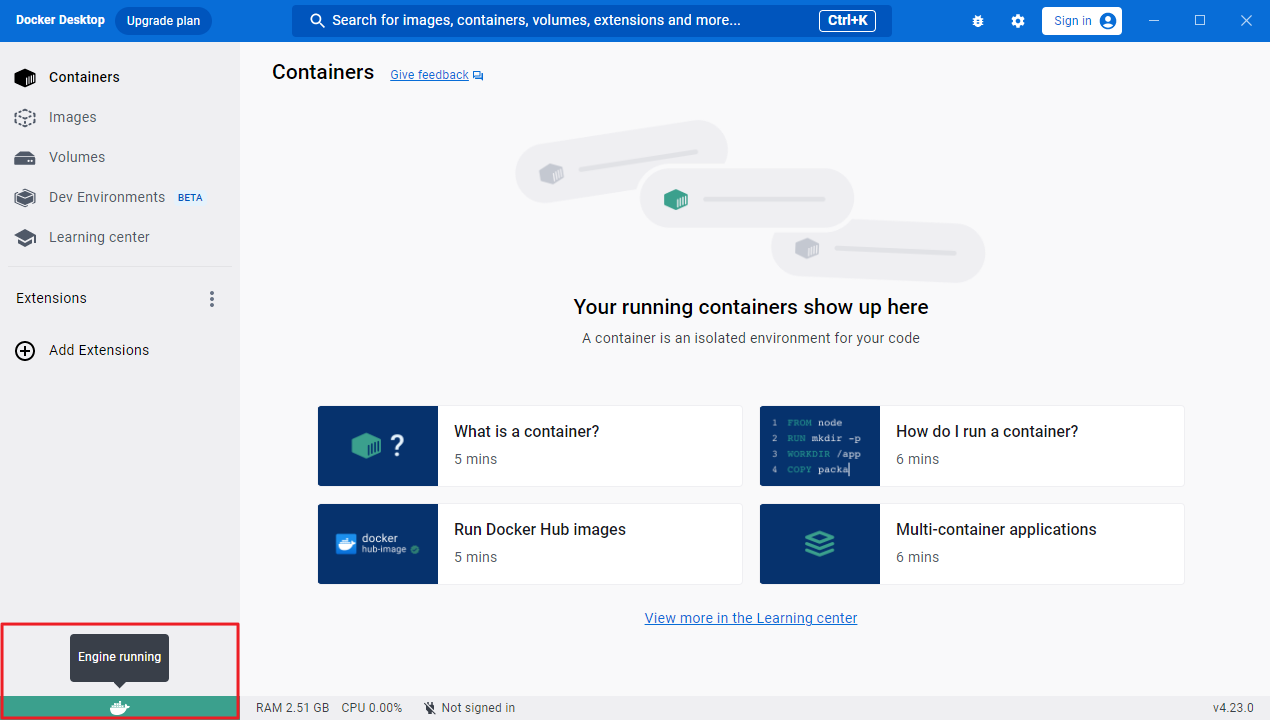
09 PowerShell를 실행하고 WSL2를 기본 버전으로 설정





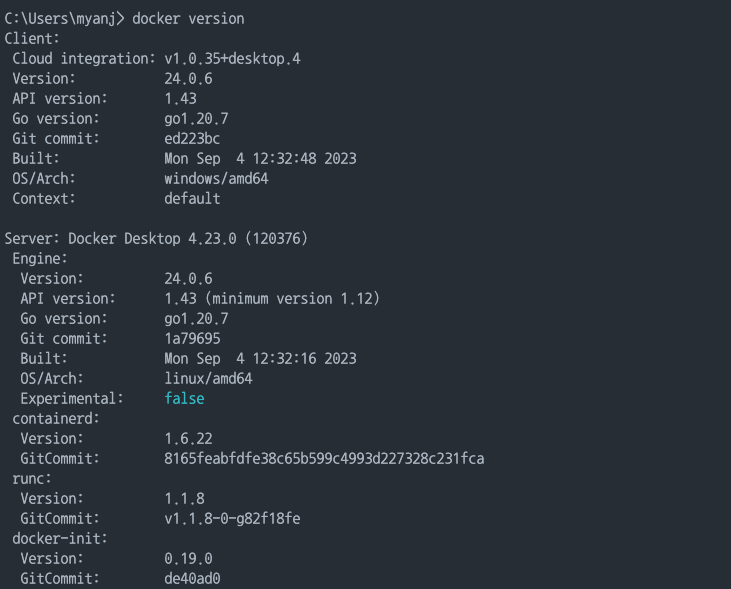
10 Docker Desktop 다시 실행

바탕화면 또는 시작 메뉴에 Docker Desktop 링크를 클릭해 Docker Desktop을 실행합니다. 화면 왼쪽 하단에 아이콘이 초록색으로 바뀌고 마우스를 올리면 Engine running 메시지가 출력되는 것을 확인할 수 있습니다.



11 명령 프롬프트에서 도커 CLI 실행 확인

명령 프롬프트를 실행하고, 도커 CLI 명령어가 실행되는지 확인합니다.



12 도커 명령어 확인

docker --help 명령어로 도커 명령어 사용법과 구성을 확인합니다.

| C:\Users\myanj> docker --help  Usage: docker [OPTIONS] COMMAND  A self-sufficient runtime for containers  Common Commands:  run Create and run a new container from an image  exec Execute a command in a running container  ps List containers  build Build an image from a Dockerfile  pull Download an image from a registry  push Upload an image to a registry  images List images  login Log in to a registry  logout Log out from a registry  search Search Docker Hub for images  version Show the Docker version information  info Display system-wide information  Management Commands:  builder Manage builds  buildx\* Docker Buildx (Docker Inc., v0.11.2-desktop.4)  compose\* Docker Compose (Docker Inc., v2.21.0-desktop.1)  container Manage containers  context Manage contexts  dev\* Docker Dev Environments (Docker Inc., v0.1.0)  extension\* Manages Docker extensions (Docker Inc., v0.2.20)  image Manage images  init\* Creates Docker-related starter files for your project (Docker Inc., v0.1.0-beta.7)  manifest Manage Docker image manifests and manifest lists  network Manage networks  plugin Manage plugins  sbom\* View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc., 0.6.0)  scan\* Docker Scan (Docker Inc., v0.26.0)  scout\* Command line tool for Docker Scout (Docker Inc., 0.24.1)  system Manage Docker  trust Manage trust on Docker images  volume Manage volumes  Swarm Commands:  swarm Manage Swarm  Commands:  attach Attach local standard input, output, and error streams to a running container  commit Create a new image from a container's changes  cp Copy files/folders between a container and the local filesystem  create Create a new container  diff Inspect changes to files or directories on a container's filesystem  events Get real time events from the server  export Export a container's filesystem as a tar archive  history Show the history of an image  import Import the contents from a tarball to create a filesystem image  inspect Return low-level information on Docker objects  kill Kill one or more running containers  load Load an image from a tar archive or STDIN  logs Fetch the logs of a container  pause Pause all processes within one or more containers  port List port mappings or a specific mapping for the container  rename Rename a container  restart Restart one or more containers  rm Remove one or more containers  rmi Remove one or more images  save Save one or more images to a tar archive (streamed to STDOUT by default)  start Start one or more stopped containers  stats Display a live stream of container(s) resource usage statistics  stop Stop one or more running containers  tag Create a tag TARGET\_IMAGE that refers to SOURCE\_IMAGE  top Display the running processes of a container  unpause Unpause all processes within one or more containers  update Update configuration of one or more containers  wait Block until one or more containers stop, then print their exit codes  Global Options:  --config string Location of client config files (default  "C:\\Users\\myanj\\.docker")  -c, --context string Name of the context to use to connect to the  daemon (overrides DOCKER\_HOST env var and  default context set with "docker context use")  -D, --debug Enable debug mode  -H, --host list Daemon socket to connect to  -l, --log-level string Set the logging level ("debug", "info",  "warn", "error", "fatal") (default "info")  --tls Use TLS; implied by --tlsverify  --tlscacert string Trust certs signed only by this CA (default  "C:\\Users\\myanj\\.docker\\ca.pem")  --tlscert string Path to TLS certificate file (default  "C:\\Users\\myanj\\.docker\\cert.pem")  --tlskey string Path to TLS key file (default  "C:\\Users\\myanj\\.docker\\key.pem")  --tlsverify Use TLS and verify the remote  -v, --version Print version information and quit  Run 'docker COMMAND --help' for more information on a command.  For more help on how to use Docker, head to https://docs.docker.com/go/guides/ |
| --- |

13 도커 하위 명령어 사용법 확인

docker *관리명령어* --help 명령어 또는 docker *관리명령어 하위명령어* --help 명령어로 도커 하위 명령어 사용법과 구성을 확인합니다.

| C:\Users\myanj> docker container --help  Usage: docker container COMMAND  Manage containers  Commands:  attach Attach local standard input, output, and error streams to a running container  commit Create a new image from a container's changes  cp Copy files/folders between a container and the local filesystem  create Create a new container  diff Inspect changes to files or directories on a container's filesystem  exec Execute a command in a running container  export Export a container's filesystem as a tar archive  inspect Display detailed information on one or more containers  kill Kill one or more running containers  logs Fetch the logs of a container  ls List containers  pause Pause all processes within one or more containers  port List port mappings or a specific mapping for the container  prune Remove all stopped containers  rename Rename a container  restart Restart one or more containers  rm Remove one or more containers  run Create and run a new container from an image  start Start one or more stopped containers  stats Display a live stream of container(s) resource usage statistics  stop Stop one or more running containers  top Display the running processes of a container  unpause Unpause all processes within one or more containers  update Update configuration of one or more containers  wait Block until one or more containers stop, then print their exit codes  Run 'docker container COMMAND --help' for more information on a command. |
| --- |

14 최신 버전의 우분투 컨테이너에서 Hello World 메시지를 출력

최신 버전의 우분투 컨테이너를 실행하고, 해당 컨테이너 쉘에서 Hello World 메시지를 출력합니다.

| C:\Users\myanj> docker container run ubuntu /bin/echo "Hello World"  Unable to find image 'ubuntu:latest' locally  latest: Pulling from library/ubuntu  445a6a12be2b: Pull complete  Digest: sha256:aabed3296a3d45cede1dc866a24476c4d7e093aa806263c27ddaadbdce3c1054  Status: Downloaded newer image for ubuntu:latest  Hello World  C:\Users\myanj> docker container ls -a  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  a1ac43f66fa4 ubuntu "/bin/echo 'Hello Wo…" 11 seconds ago Exited (0) 10 seconds ago boring\_lalande  C:\Users\myanj> docker image ls  REPOSITORY TAG IMAGE ID CREATED SIZE  ubuntu latest c6b84b685f35 4 weeks ago 77.8MB |
| --- |

**[참고]**

WSL2가 설치되어 있어야 Docker Desktop이 정상 작동함.

[[Docker] Docker Desktop - Unexpected WSL error in Windows11 Home](https://ohorange.tistory.com/56)

### **✅ Docker 설치 (Mac OS)**

이번에는 맥북에 Docker를 설치해 보겠습니다.

1. 설치 파일 다운로드

우선 공식 사이트 또는 Docker hub에서 본인 사양에 맞는 설치 파일을 다운로드한다.

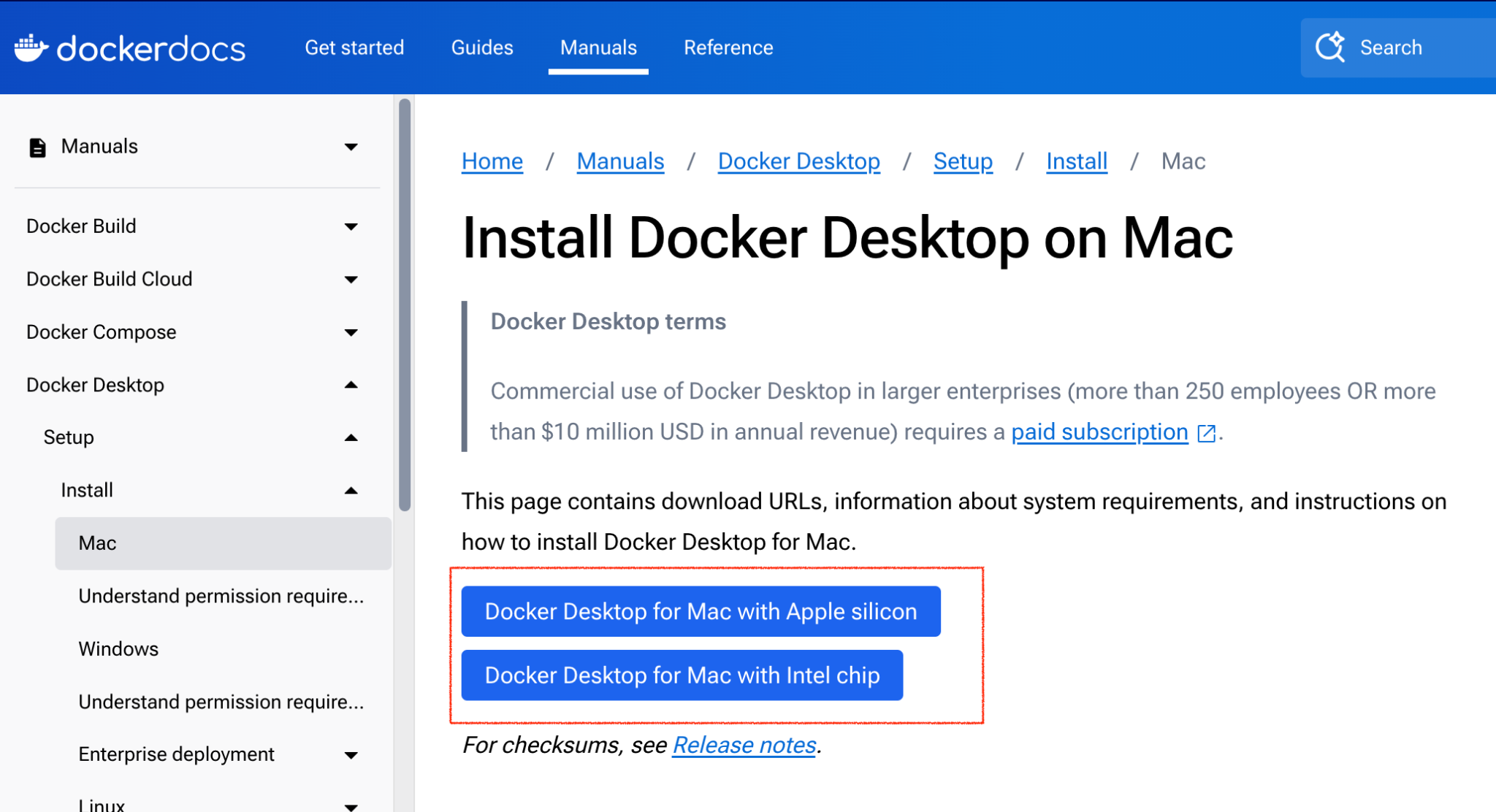
[Install Docker Desktop on Mac](https://docs.docker.com/desktop/setup/install/mac-install/)

[Get Started | Docker](https://www.docker.com/get-started/)

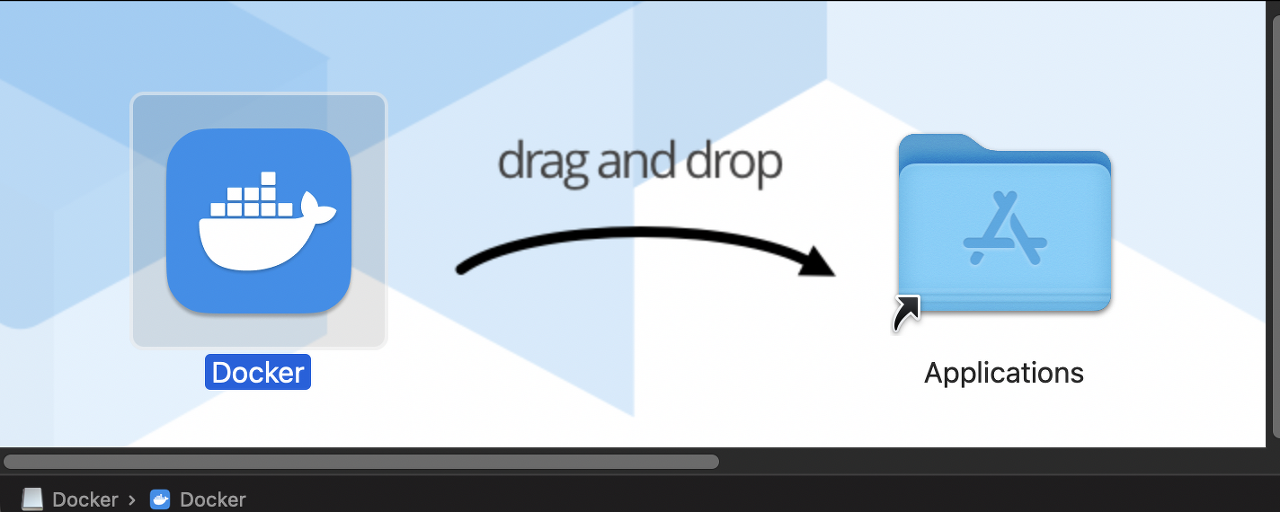
설명을 하기위해 docker hub쪽에서 다운로드 하겠습니다.(위족 링크)

본인에게 맞는 칩셋을 선택하여 설치하도록 합니다.

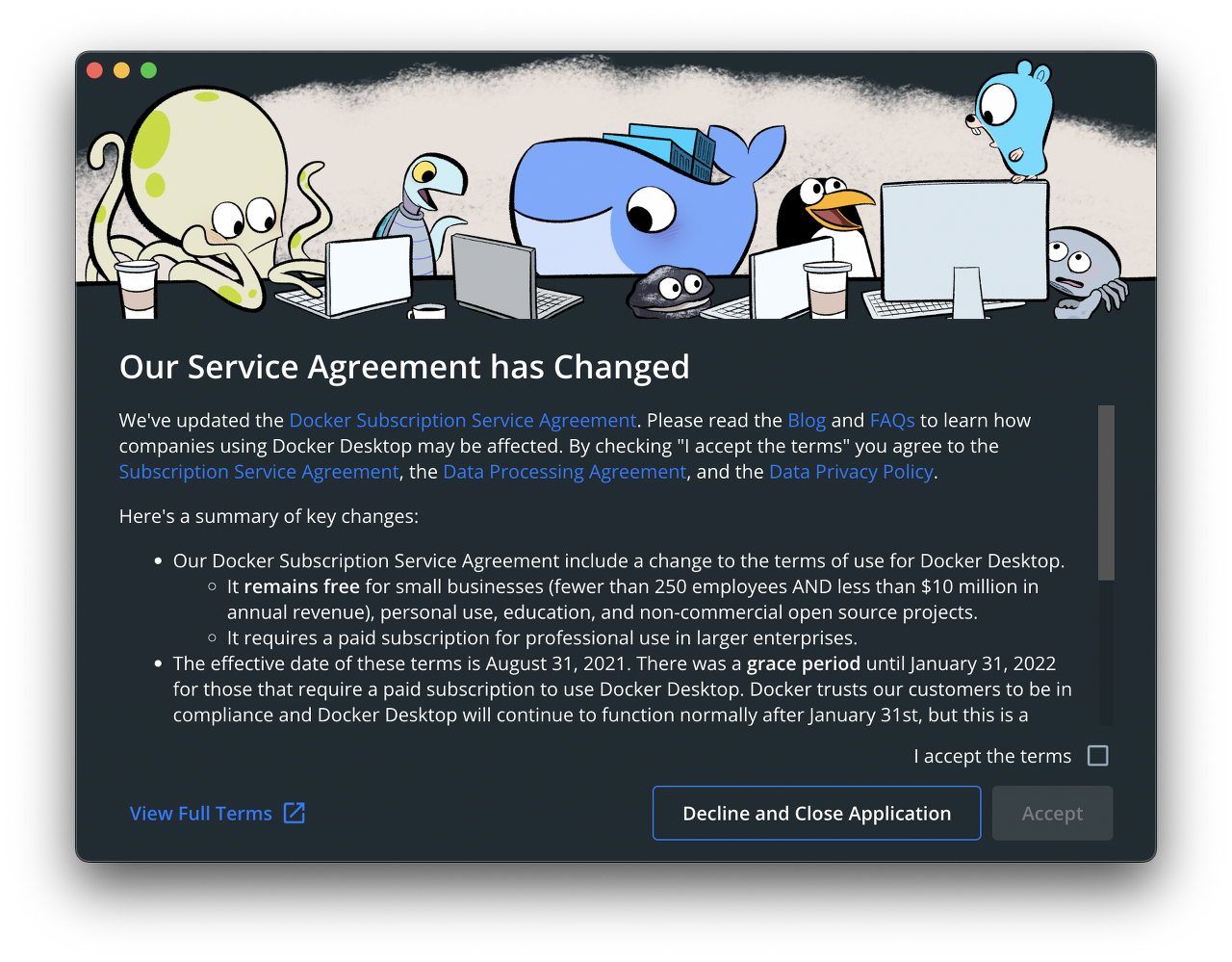
최근에 구매한 M 유저라면 ‘Apple silicon’으로 선택하면 됩니다. 그외에는 “Intel chip”을 선택합니다.



1. 설치하기



다운로드한 .dmg 파일을 실행하여 Docker.app을 응용 프로그램으로 복사해준다.



1. 구동 및 환경 설정 확인

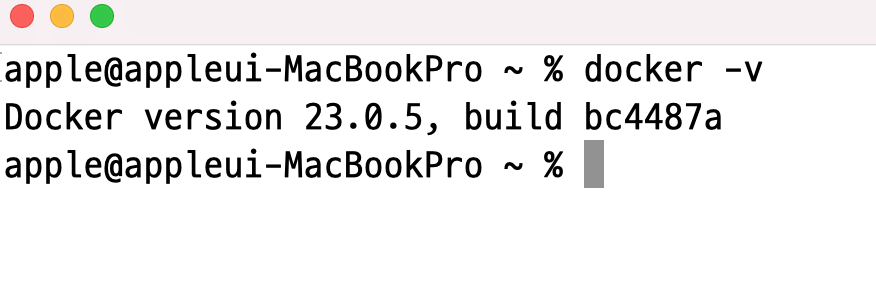
이제 응용프로그램에 보면 Docker App이 설치된 것을 확인할 수 있다.



설치된 Docker App을 실행하면 맨 위 메뉴바에 고래 모양 Docker App이 생기면서 실행 된다.



한번 더 Terminal을 이용하여 확인 및 버전 정보를 확인해 봅니다.



이제 Docker가 잘 설치 되었으니, Docker를 이용하여 여러 애플리케이션들을 사용해 봅시다.

### **✅ 체크 사항**

2023년 7월부터 Docker Compose V1의 업데이트를 중단했다. 따라서 Docker Compose는 V2를 설치할 것을 권장한다. 혹시나 기존에 설치되어 있는 Docker Compose의 버전이 V1이라면 V2로 교체할 것을 권장한다.

**실습)Docker 전체 흐름 느껴보기 (Nginx 설치 및 실행)**

### **✅ Docker를 조작하려면?**

터미널에서 명령어(CLI)를 통해 Docker를 조작한다.

### **✅ 혹시나 Nginx가 뭔지 모르는 분들을 위해**

Nginx란 여러 기능을 가진 서버 중 하나이다.

* **웹 서버 (HTML 웹 페이지를 렌더링 시키는 역할)를 포스트 하는 기능이있는 서버**
* ~~로드 밸런싱~~ (몰라도 괜찮습니다.)
* ~~리버스 프록시~~ (몰라도 괜찮습니다.)

### **✅ 무작정 따라하면서 Docker 사용해보기**

🧑🏻 아래 명령어가 지금 바로 이해 안 된다고 해도 괜찮습니다! 다음 섹션에서 차근차근 하나씩 정리 해드릴거에요😊

먼 Docker Desktop 실행합니다.

1. **Nginx 이미지 다운로드** [nginx - Official Image | Docker Hub](https://hub.docker.com/_/nginx)

| $ docker pull nginx |
| --- |

도커 이미지를 다운 받는 명령어

1. **다운로드 된 이미지 확인하기**

| $ docker image ls |
| --- |

ls : list의 약자

1. **이미지를 컨테이너에 올려 Nginx 서버 실행시키기**

| $ docker run --name webserver -d -p 80:80 nginx |
| --- |

1. **Nginx 서버가 잘 실행되는 지 확인하기**

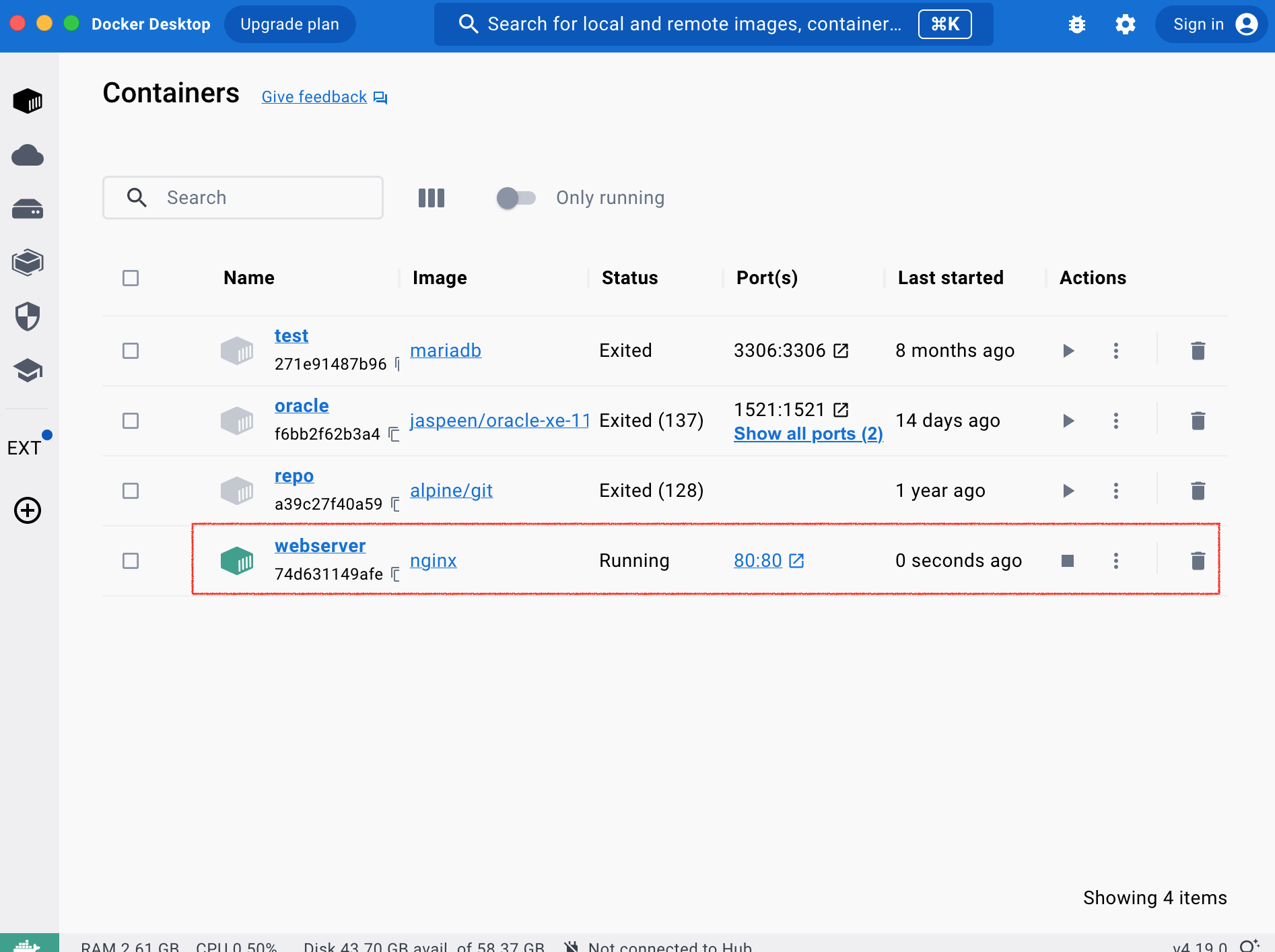
**브라우저 띄우고 lodcalhost:80**

****

1. **실행되고 있는 모든 컨테이너 상태 확인하기**

| $ docker ps |
| --- |

데스크탑에서 확인하기

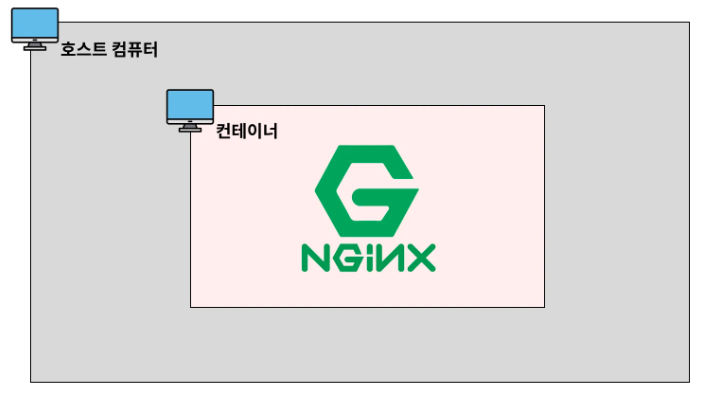


1. **특정 컨테이너 정지**

| $ docker stop webserver |
| --- |

다시 웹을 열고 들어가면 접속이 안되는 것을 알 수 있음.

### **✅ 그림으로 이해하기**

****