

[실습 환경 구성]

- WSL2 설치(Windows Subsystem for Linux)
 - <https://docs.microsoft.com/ko-kr/windows/wsl/install-win10> 참조
 - www.microsoft.co.kr 접속 후 검색에서 WSL2 검색
 - “Windows 10에 Linux용 Windows 하위 시스템 설치 가이드” 선택

1단계: 파워셸에서 다음 명령 실행

dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
실행 후 리붓!!!

2단계: 윈도우 최신 버전 업데이트 확인

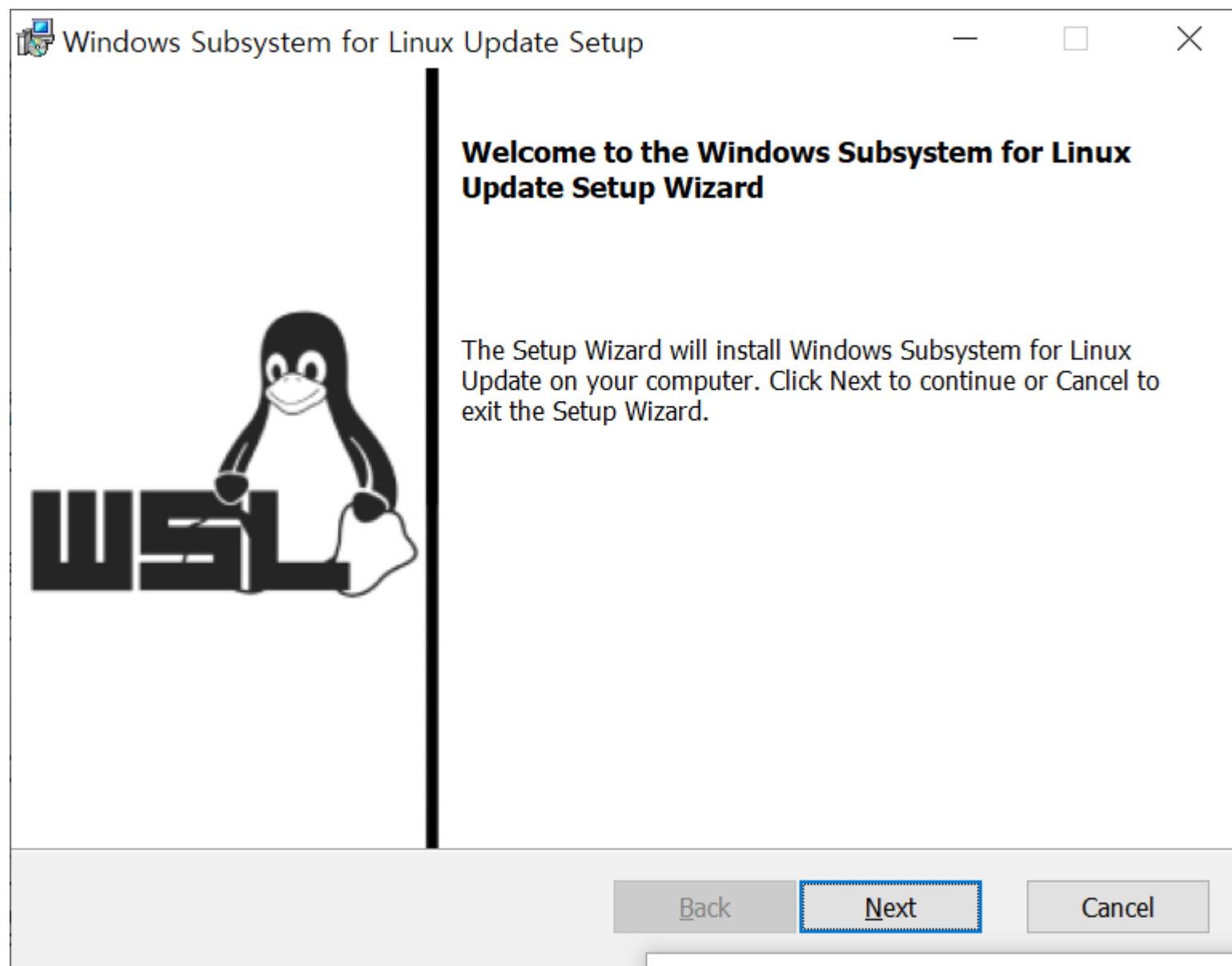
- Win키 + R 선택 후 winver 입력하여 아래 빌드 요건 확인
 - x64 시스템의 경우: 버전 1903 이상, 빌드 18362 이상
 - ARM64 시스템의 경우: 버전 2004 이상, 빌드 19041 이상
- 18362보다 낮은 빌드는 WSL 2를 지원하지 않습니다.

3단계: 파워셸에서 다음 명령 실행

dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

4단계: 설치 프로그램 다운로드 후 설치


https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi



5단계 - WSL 2를 기본 버전으로 설정

PowerShell을 열고 이 명령을 실행하여 새 Linux 배포를 설치할 때 WSL 2를 기본 버전으로 설정합니다.

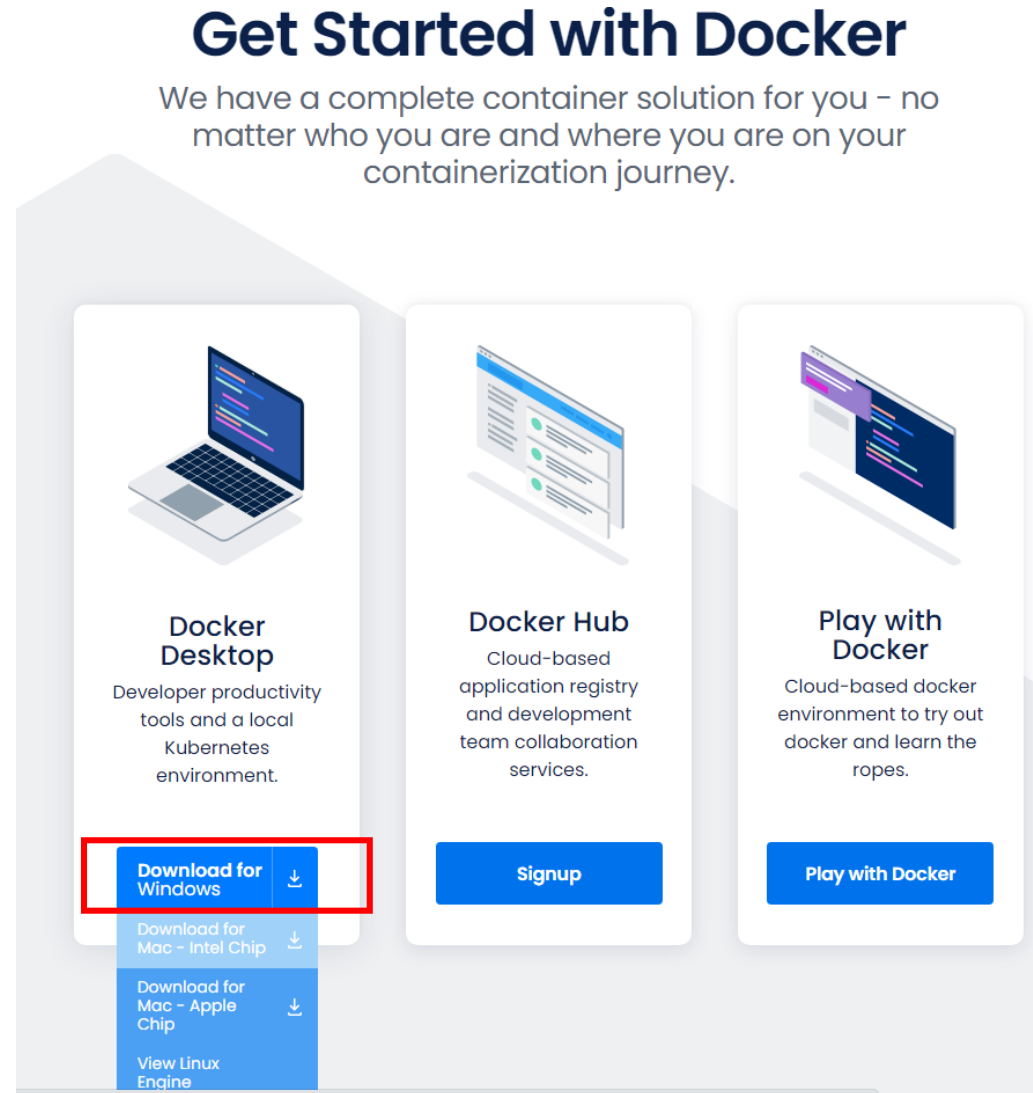
PowerShell

 복사

```
wsl --set-default-version 2
```

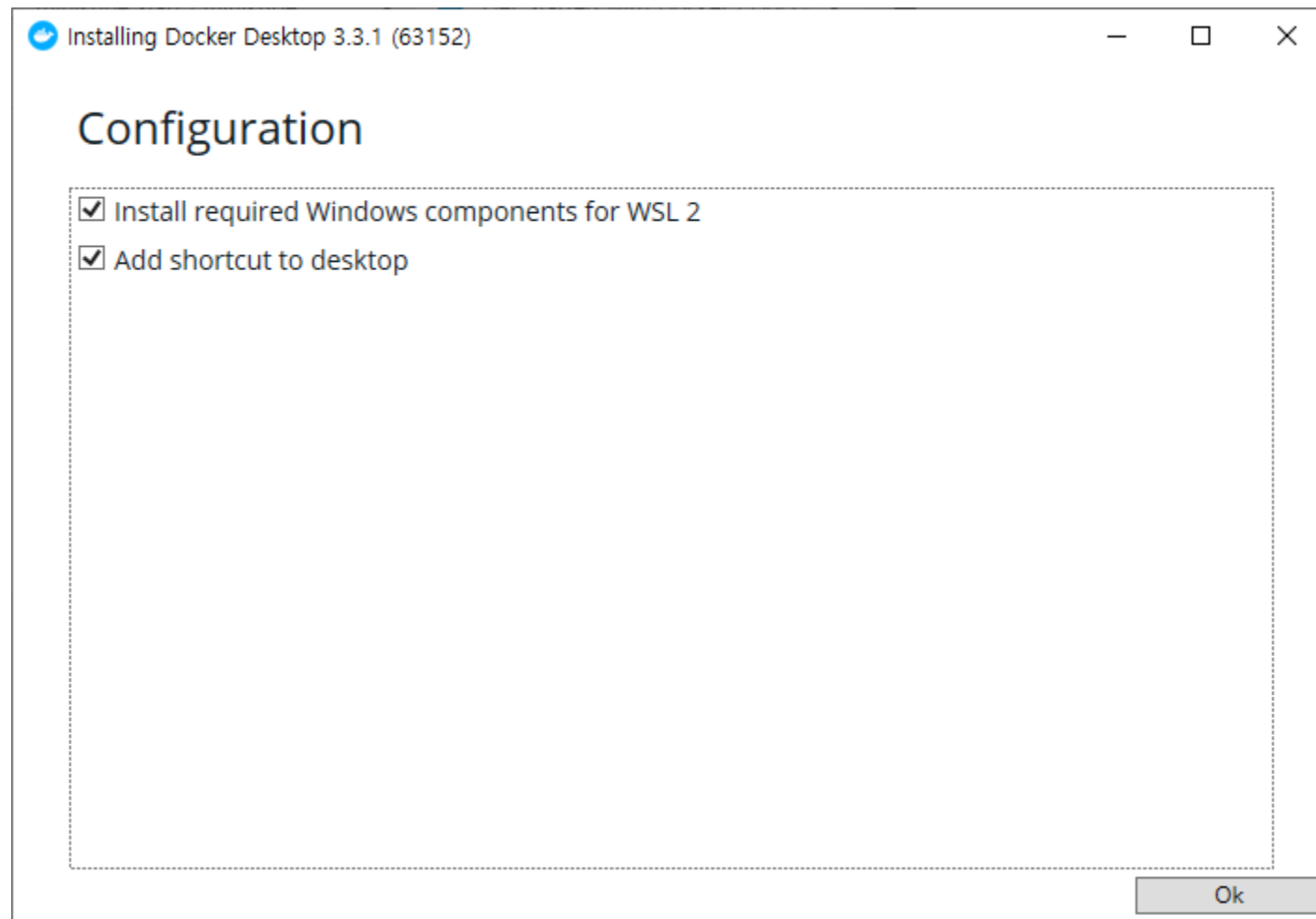
[실습 환경 구성] – Docker 설치

- Docker Desktop 설치
Docker.com 접속 → Get Started → Docker Desktop → Download for Windows 다운로드 후 설치



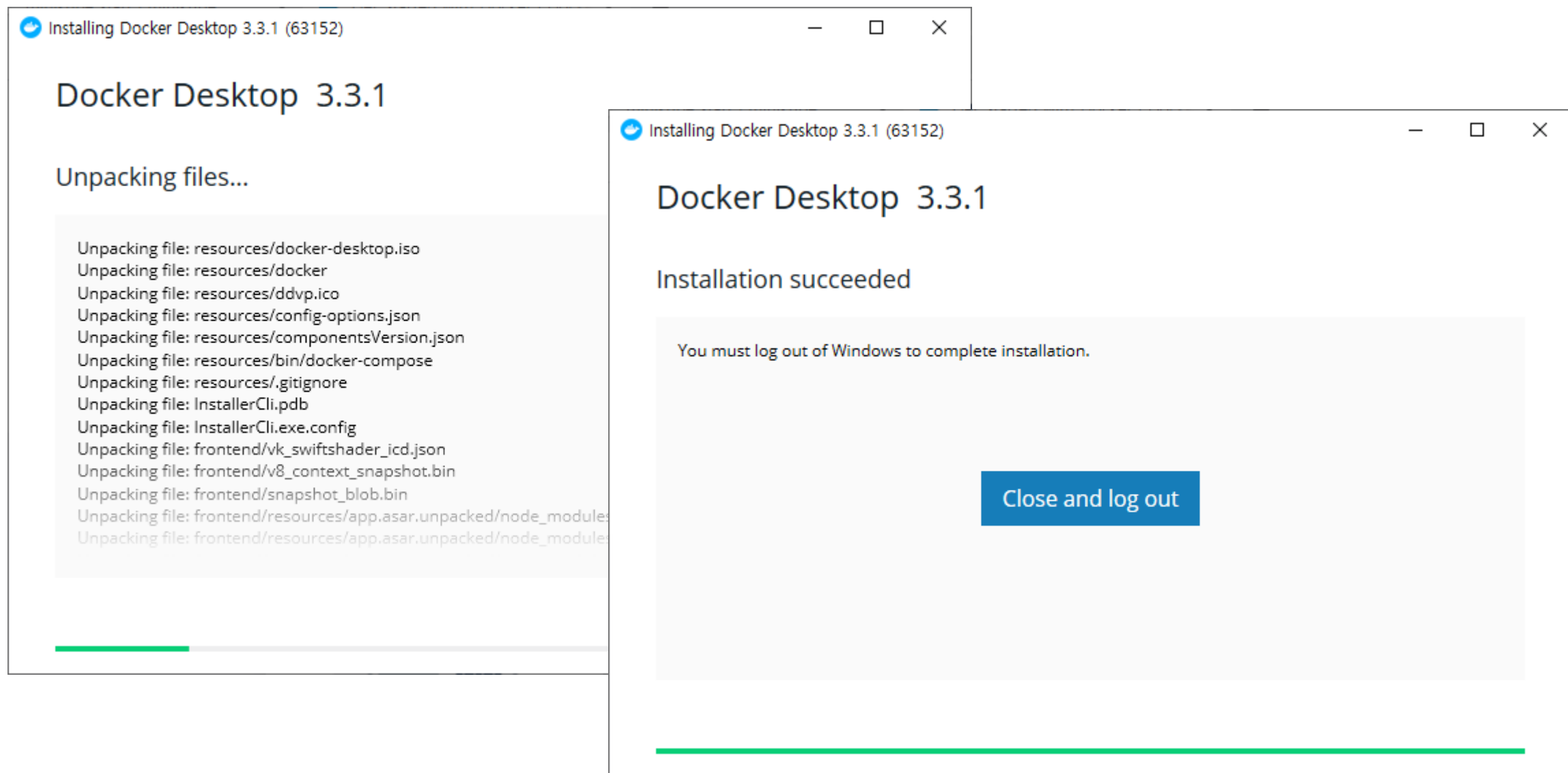
[실습 환경 구성] – Docker 설치

- Docker Desktop 설치
설치 Configuration



[실습 환경 구성] – Docker 설치

- Docker Desktop 설치
설치 진행

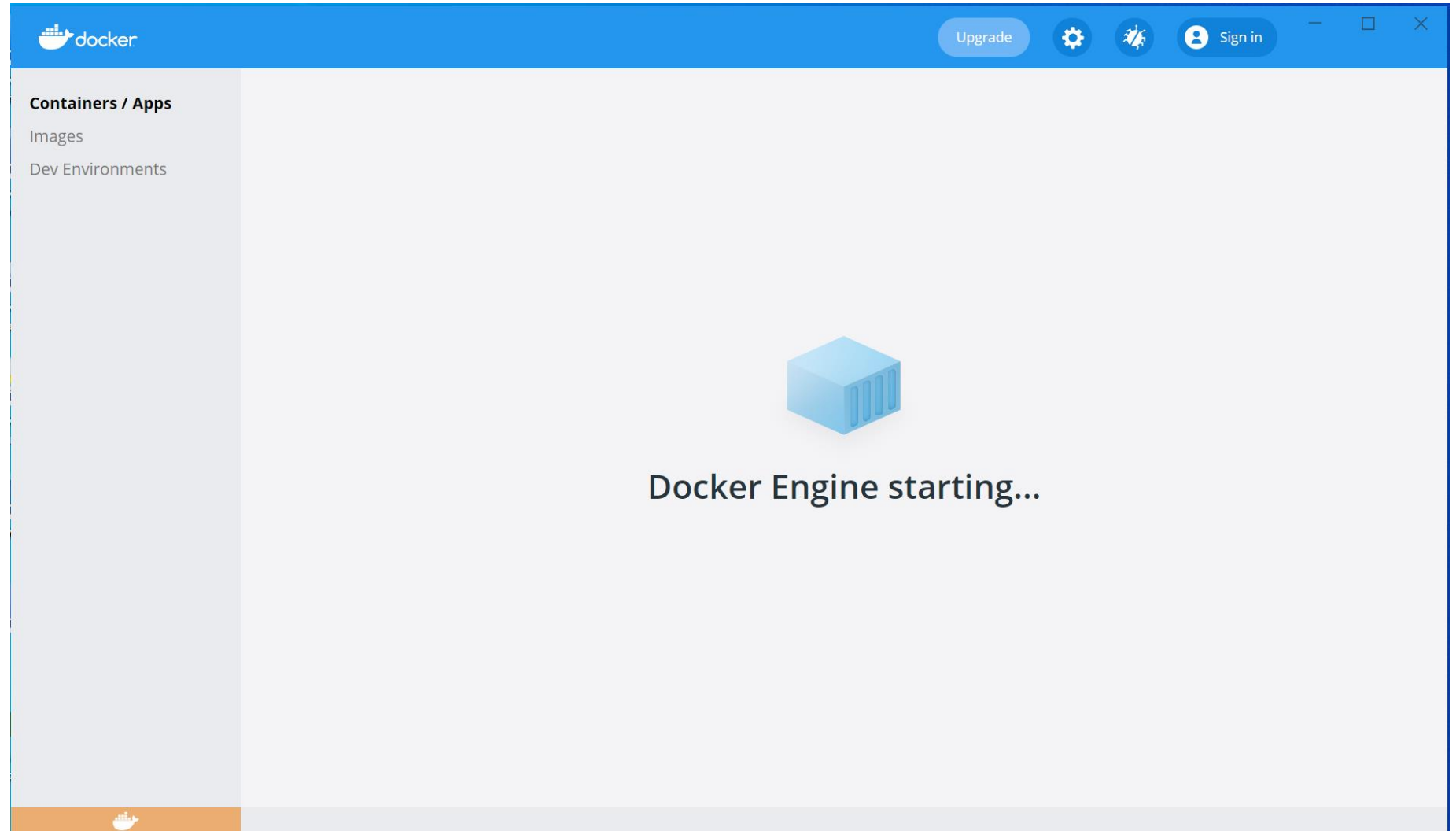


[실습 환경 구성] – Docker 설치

- Docker Desktop 설치

Docker Desktop 설치 완료
후 모습

설치가 완료되면 Docker
Engine이 실행 된다



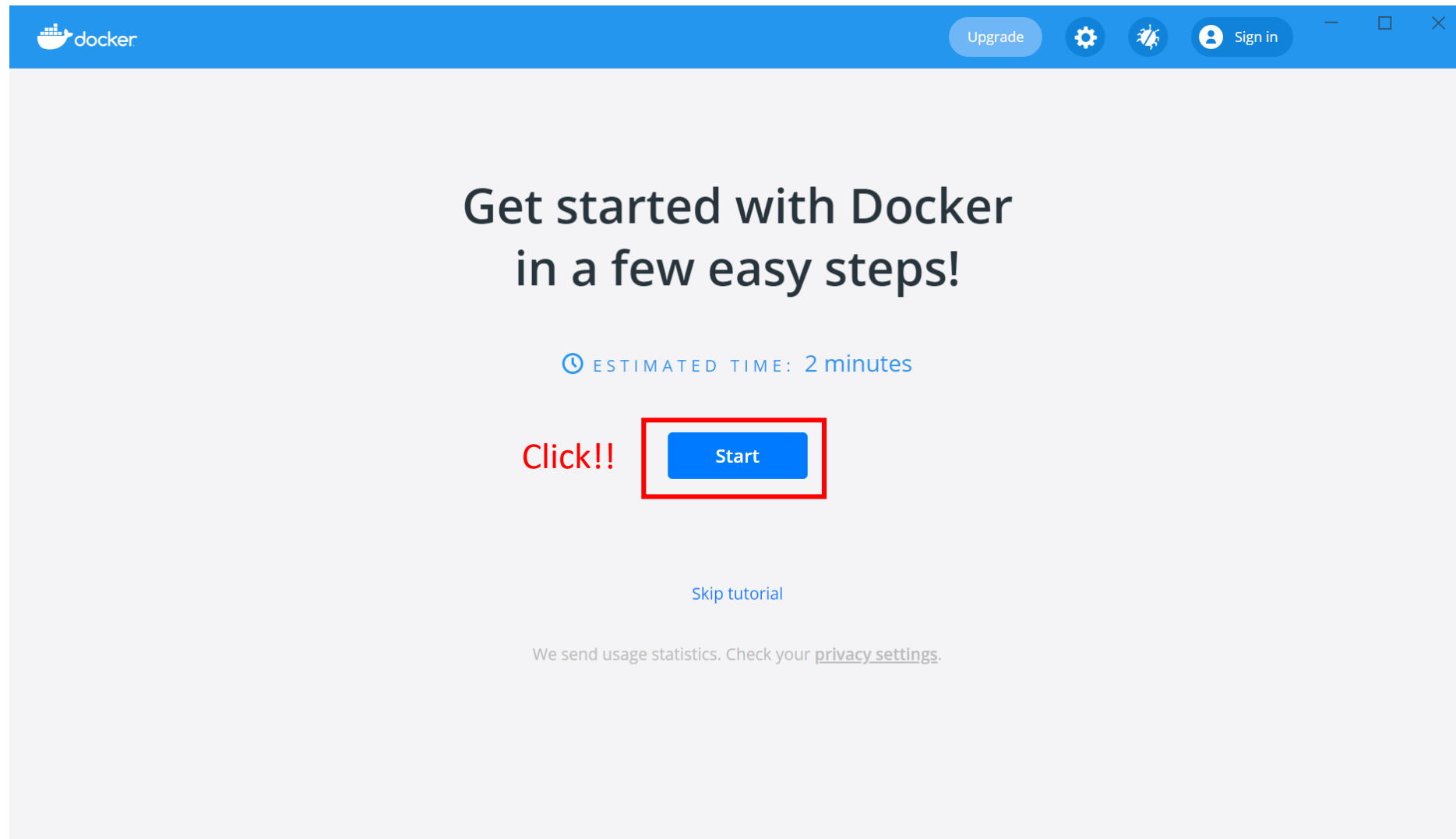
[실습 환경 구성] – Docker tutorial

- Tutorial에 따라 sample 이미지를 받아 작동 시켜 보자

설치 완료되면 “Start”
버튼을 클릭하여
Tutorial을 시작해 보자

Tutorial 진행 단계

1. Github에서 프로젝트 Clone(복제)
2. 프로젝트를 현재 디렉토리로 복사
3. 이미지 빌드
4. 생성된 이미지로 컨테이너에서 실행
5. 생성된 이미지를 Docker Hub에 업로드
6. 실행된 컨테이너 작동 확인



[실습 환경 구성] – Docker tutorial

- 1. Github의 예제 이미지 Clone

[진행 절차]

1. github.com에서 getting-started 이미지를 로컬 리퍼지토리로 Clone(복제)
2. 로컬 리퍼지토리에 Clone된 이미지를 현재 디렉토리로 복사

파란색 박스를 클릭하여
진행해 보자

docker

Upgrade

Sign in

1 Clone

2 Build

3 Run

4 Share

First, clone a repository

The *Getting Started* project is a simple GitHub repository which contains everything you need to build an image and run it as a container.

Clone the repository by running Git in a container.

Click!!

```
docker run --name repo alpine/git clone https://github.com/docker/getting-started.git
docker cp repo:/git/getting-started/ .
```

You can also type the command directly in a command line interface.

Next Step

Skip tutorial

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6
PS C:\Users\controlc>

[실습 환경 구성] – Docker tutorial

- 1. Github의 예제 이미지 Clone

- docker run – name
repo alpine/git clone
[https://github.com/doc
ker/getting-started.git](https://github.com/docker/getting-started.git)
➔ github.com의
getting-started
프로젝트를 로컬
리퍼지토리로 clone

The screenshot shows the Docker website's tutorial for cloning a repository. The sidebar on the left lists four steps: 1. Clone (highlighted), 2. Build, 3. Run, and 4. Share. The main content area is titled 'First, clone a repository' and explains that the 'Getting Started' project is a simple GitHub repository. It provides a code block with the following commands:

```
docker run --name repo alpine/git clone \
https://github.com/docker/getting-started.git
docker cp repo:/git/getting-started/ .
```

Below the code block, it says 'You can also type the command directly in a command line interface.' At the bottom, there is a blue 'Next Step' button, which is highlighted with a red rectangular box. The text 'Click!!' is written in red above the button. In the bottom left corner, there is a link 'Skip tutorial'.

On the right side of the screenshot, a Windows PowerShell terminal window is open, showing the execution of the Docker command and the successful cloning of the repository:

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

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

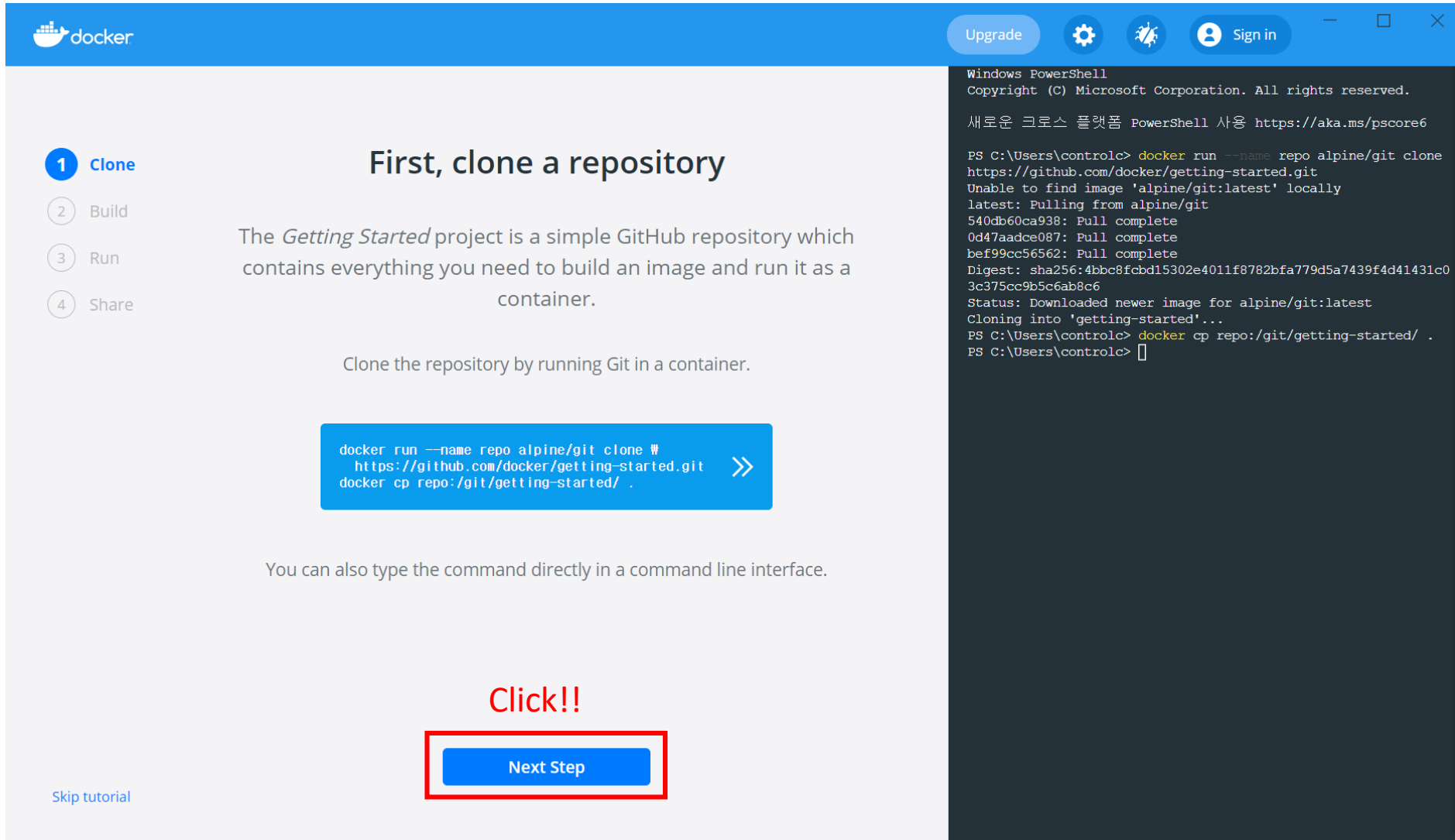
PS C:\Users\controlc> docker run --name repo alpine/git clone
https://github.com/docker/getting-started.git
Unable to find image 'alpine/git:latest' locally
latest: Pulling from alpine/git
540db60ca938: Pull complete
0d47aadce087: Pull complete
bef99cc56562: Pull complete
Digest: sha256:4bbc8fcbd15302e4011f8782bfa779d5a7439f4d41431c0
3c375cc9b5c6ab8c6
Status: Downloaded newer image for alpine/git:latest
Cloning into 'getting-started'...
PS C:\Users\controlc> docker cp repo:/git/getting-started/ .
PS C:\Users\controlc>
```

[실습 환경 구성] – Docker tutorial

- 2. 프로젝트를 현재 디렉토리로 복사

- `docker cp`
 `repo:/git/getting-started/` .
→ 로컬 리포지토리에서
 현재 디렉토리로
 복사

Next Step을 클릭하여
진행해 보자



Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

```
PS C:\Users\controlc> 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
540db60ca938: Pull complete
0d47aadce087: Pull complete
bef99cc56562: Pull complete
Digest: sha256:4bbc8fcbd15302e4011f8782bfa779d5a7439f4d41431c03c375cc9b5c6ab8c6
Status: Downloaded newer image for alpine/git:latest
Cloning into 'getting-started'...
PS C:\Users\controlc> docker cp repo:/git/getting-started/ .
PS C:\Users\controlc> 
```

[실습 환경 구성] – Docker tutorial

• 3. Docker 이미지 만들기

파란색 박스를 클릭하여
이미지 빌드 명령 실행

- cd getting-started
➔ getting-start
디렉토리로 이동
- docker build -t
docker101tutorial
➔ docker101tutorial
명칭으로 현재
디렉토리를 docker
이미지로 만들기

The screenshot shows the Docker Desktop application window. The main area displays the 'Now, build the image' step of a tutorial. A red rectangular box highlights a blue button containing the command: `cd getting-started` followed by `docker build -t docker101tutorial .` and a double arrow `>>`. To the left of this box, the text 'Click!!' is written in red. The sidebar on the left shows four steps: 'Clone' (checked), 'Build' (selected), 'Run', and 'Share'. At the bottom of the sidebar is a 'Skip tutorial' link. At the bottom of the main area is a 'Next Step' button. On the right side of the window, a PowerShell terminal is open, showing the execution of the Docker build command and the output, which includes pulling the 'alpine/git:latest' image and cloning the repository.

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

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

PS C:\Users\controlc> 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
540db60ca938: Pull complete
0d47aadce087: Pull complete
bef99cc56562: Pull complete
Digest: sha256:4bbc8fcbd15302e4011f8782bfa779d5a7439f4d41431c0
3c375cc9b5c6ab8c6
Status: Downloaded newer image for alpine/git:latest
Cloning into 'getting-started'...
PS C:\Users\controlc> docker cp repo:/git/getting-started/ .
PS C:\Users\controlc>
```

[실습 환경 구성] – Docker tutorial

- 3. Docker 이미지 만들기

“Next Step”을 클릭하여
다음 단계로 이동

docker

Upgrade

Sign in

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 .
```

Click!!

Next Step

Skip tutorial

```
위치 줄:1 문자:1
+ cd getting-started
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (C:\Users\cont
ro...getting-started:String) [Set-Location], ItemNotFoun
dException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerSh
ell.Commands.SetLocationCommand

PS C:\Users\controlc\getting-started> docker build -t docker1
01tutorial .
[+] Building 83.3s (25/25) FINISHED
=> [internal] load build definition from Dockerfile      0.3s
=> => transferring dockerfile: 32B                      0.0s
=> [internal] load .dockerignore                        0.1s
=> => transferring context: 34B                          0.0s
=> [internal] load metadata for docker.io/library/pyth  4.3s
=> [internal] load metadata for docker.io/library/node  4.2s
=> [internal] load metadata for docker.io/library/ngin  4.5s
=> [app-base 1/8] FROM docker.io/library/node:12-alpin  0.0s
=> [base 1/4] FROM docker.io/library/python:alpine@sha  0.0s
=> [internal] load build context                        0.2s
=> => transferring context: 7.70kB                       0.1s
=> CACHED [stage-5 1/3] FROM docker.io/library/nginx:a  0.0s
=> CACHED [base 2/4] WORKDIR /app                      0.0s
=> CACHED [base 3/4] COPY requirements.txt .            0.0s
=> CACHED [base 4/4] RUN pip install -r requirements.t  0.0s
=> CACHED [build 1/2] COPY . .                          0.0s
=> CACHED [build 2/2] RUN mkdocs build                  0.0s
=> CACHED [app-base 2/8] RUN apk add --no-cache python  0.0s
=> CACHED [app-base 3/8] WORKDIR /app                   0.0s
=> CACHED [app-base 4/8] COPY app/package.json app/yar  0.0s
=> [app-base 5/8] RUN yarn install                      59.1s
=> [app-base 6/8] COPY app/spec ./spec                 0.2s
=> [app-base 7/8] COPY app/src ./src                   0.3s
=> [app-base 8/8] RUN yarn test                        11.0s
=> [app-zip-creator 1/1] RUN rm -rf node_modules &&    4.8s
=> [stage-5 2/3] COPY --from=app-zip-creator /app.zip  0.2s
=> [stage-5 3/3] COPY --from=build /app/site /usr/shar  0.3s
=> exporting to image                                 0.2s
=> => exporting layers                                    0.2s
=> => writing image sha256:0dc7cc3653ec647a6e15a28f025  0.0s
=> => naming to docker.io/library/docker101tutorial    0.0s

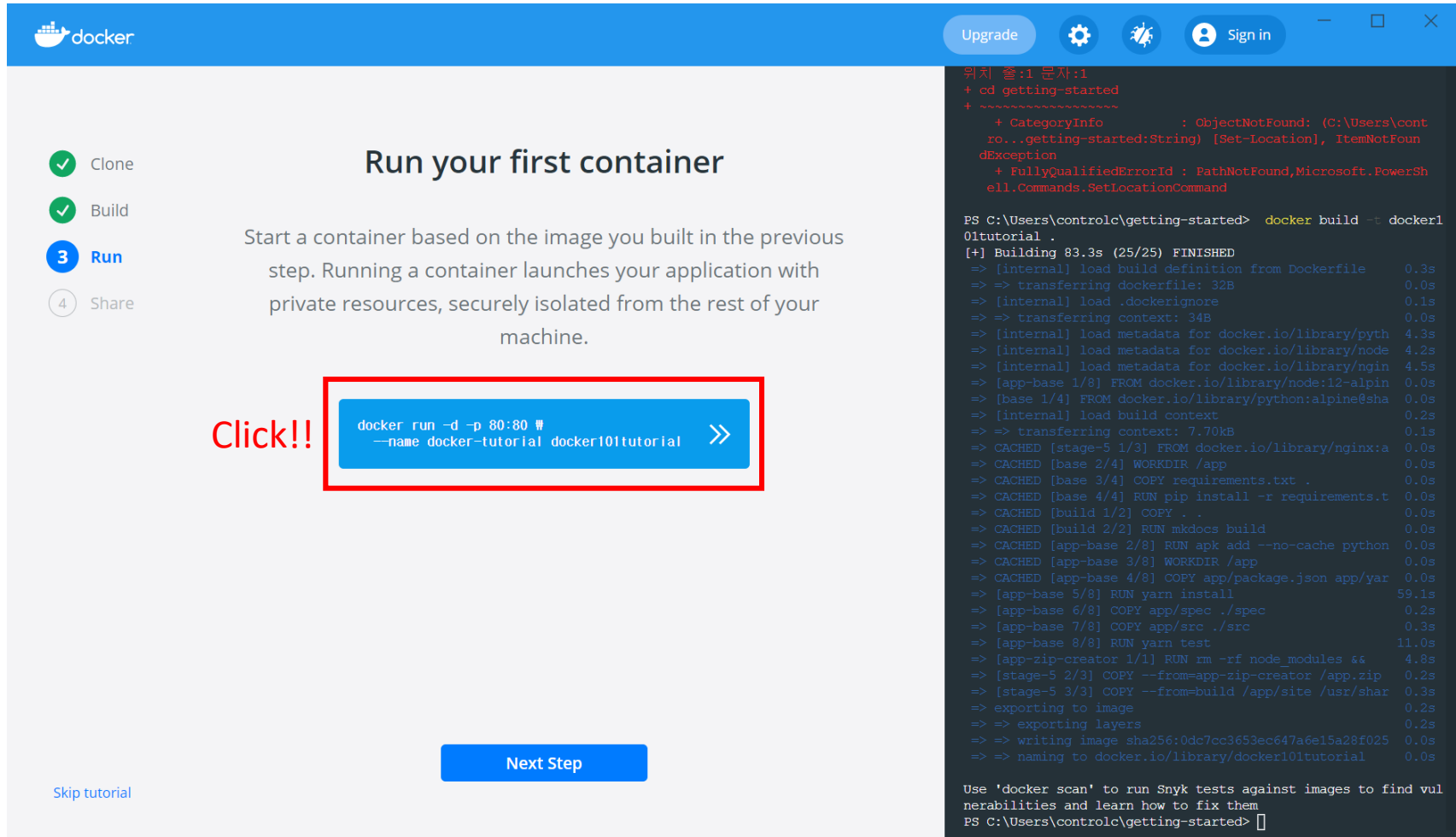
Use 'docker scan' to run Snyk tests against images to find vul
nerabilities and learn how to fix them
PS C:\Users\controlc\getting-started>
```

[실습 환경 구성] – Docker tutorial

• 4. 생성된 이미지를 Docker 컨테이너에서 실행

파란색 박스를 클릭하여
컨테이너 실행

- `docker run -d -p 80:80 --name docker-tutorial docker101tutorial`
➔ `docker-tutorial` 이라는 컨테이너를 생성하고 컨테이너에서 `docker101tutorial` 이미지를 실행
➔ 컨테이너 내부 80포트를 외부 80포트에 연결



The screenshot shows the Docker Desktop interface. On the left, there's a sidebar with 'Clone', 'Build', 'Run' (highlighted with a blue circle and '3'), and 'Share' (highlighted with a blue circle and '4'). The main area is titled 'Run your first container' and contains the text: 'Start a container based on the image you built in the previous step. Running a container launches your application with private resources, securely isolated from the rest of your machine.' Below this text is a blue button with the command `docker run -d -p 80:80 --name docker-tutorial docker101tutorial` and a right arrow. A red box highlights this button, and a red text 'Click!!' points to it. At the bottom left is a 'Skip tutorial' link, and at the bottom center is a 'Next Step' button. On the right side, there's a terminal window showing the command `docker build -t docker101tutorial .` and its output, which includes the build progress and the final image name `docker101tutorial`.

[실습 환경 구성] – Docker tutorial

- 4. 생성된 이미지를 Docker 컨테이너에서 실행

“Next Step”을 클릭하여
다음 단계로 이동

✓ Clone

✓ Build

3 Run

4 Share

Run your first container

Start a container based on the image you built in the previous step. Running a container launches your application with private resources, securely isolated from the rest of your machine.

```
docker run -d -p 80:80 \
--name docker-tutorial docker101tutorial
```

Click!!

Next Step

Skip tutorial

Upgrade

Settings

Sign in

```
ro...getting-started:String) [Set-Location], ItemNotFound
Exception
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerSh
ell.Commands.SetLocationCommand

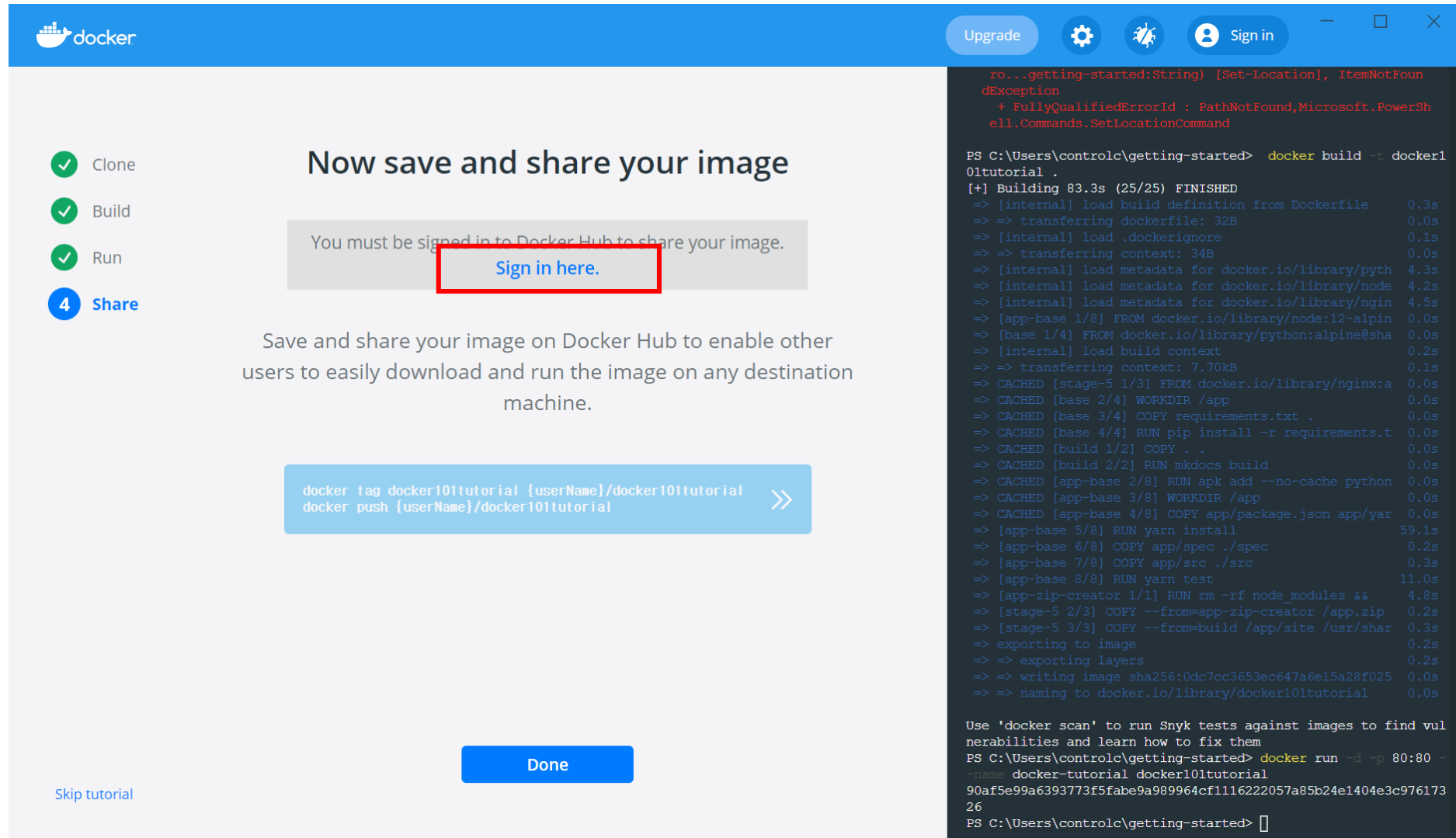
PS C:\Users\controlc\getting-started> docker build -t docker1
01tutorial .
[+] Building 83.3s (25/25) FINISHED
=> [internal] load build definition from Dockerfile 0.3s
=> => transferring dockerfile: 32B 0.0s
=> [internal] load .dockerignore 0.1s
=> => transferring context: 34B 0.0s
=> [internal] load metadata for docker.io/library/python 4.3s
=> [internal] load metadata for docker.io/library/node 4.2s
=> [internal] load metadata for docker.io/library/nginx 4.5s
=> [app-base 1/8] FROM docker.io/library/node:12-alpin 0.0s
=> [base 1/4] FROM docker.io/library/python:alpine@sha 0.0s
=> [internal] load build context 0.2s
=> => transferring context: 7.70kB 0.1s
=> CACHED [stage-5 1/3] FROM docker.io/library/nginx:a 0.0s
=> CACHED [base 2/4] WORKDIR /app 0.0s
=> CACHED [base 3/4] COPY requirements.txt . 0.0s
=> CACHED [base 4/4] RUN pip install -r requirements.t 0.0s
=> CACHED [build 1/2] COPY . . 0.0s
=> CACHED [build 2/2] RUN mkdocs build 0.0s
=> CACHED [app-base 2/8] RUN apk add --no-cache python 0.0s
=> CACHED [app-base 3/8] WORKDIR /app 0.0s
=> CACHED [app-base 4/8] COPY app/package.json app/yar 0.0s
=> [app-base 5/8] RUN yarn install 59.1s
=> [app-base 6/8] COPY app/spec ./spec 0.2s
=> [app-base 7/8] COPY app/src ./src 0.3s
=> [app-base 8/8] RUN yarn test 11.0s
=> [app-zip-creator 1/1] RUN rm -rf node_modules && 4.8s
=> [stage-5 2/3] COPY --from=app-zip-creator /app.zip 0.2s
=> [stage-5 3/3] COPY --from=build /app/site /usr/shar 0.3s
=> exporting to image 0.2s
=> => exporting layers 0.2s
=> => writing image sha256:0dc7cc3653ec647a6e15a28f025 0.0s
=> => naming to docker.io/library/docker101tutorial 0.0s

Use 'docker scan' to run Snyk tests against images to find vul
nerabilities and learn how to fix them
PS C:\Users\controlc\getting-started> docker run -d -p 80:80 -
--name docker-tutorial docker101tutorial
90af5e99a6393773f5fabe9a989964cf1116222057a85b24e1404e3c976173
26
PS C:\Users\controlc\getting-started>
```

[실습 환경 구성] – Docker tutorial

- 5. 생성된 이미지를 Docker Hub에 업로드

“sign in here”를 클릭하여
Docker Hub에 로그인



Now save and share your image

You must be signed in to Docker Hub to share your image.

[Sign in here.](#)

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 {userName}/docker101tutorial
docker push {userName}/docker101tutorial
```

Done

[Skip tutorial](#)

```
ro...getting-started:String) [Set-Location], ItemNotFound
Exception
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerSh
ell.Commands.SetLocationCommand

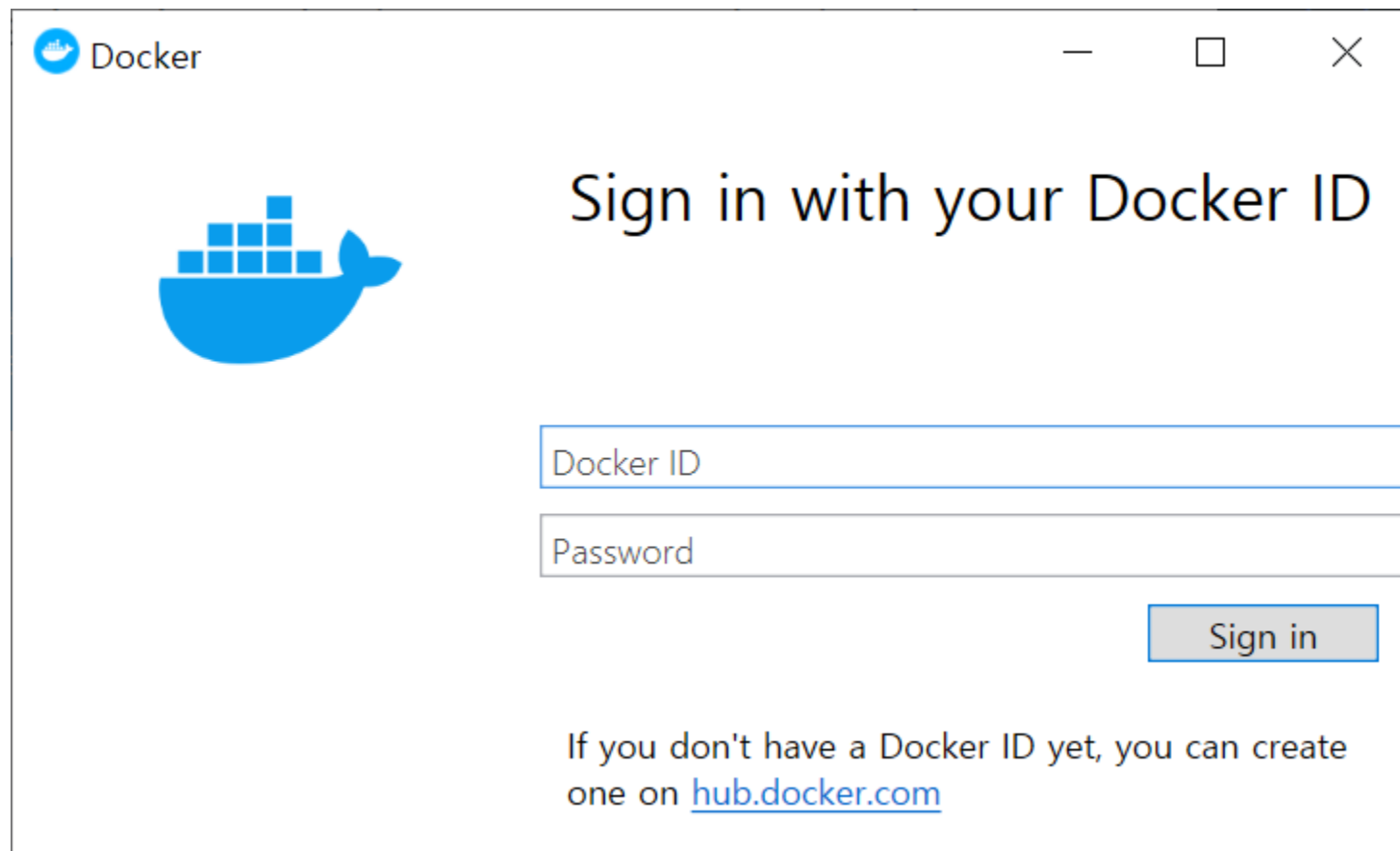
PS C:\Users\controlc\getting-started> docker build -t docker1
01tutorial .
[+] Building 83.3s (25/25) FINISHED
=> [internal] load build definition from Dockerfile 0.3s
=> => transferring dockerfile: 32B 0.0s
=> [internal] load .dockerignore 0.1s
=> => transferring context: 34B 0.0s
=> [internal] load metadata for docker.io/library/python 4.3s
=> [internal] load metadata for docker.io/library/node 4.2s
=> [internal] load metadata for docker.io/library/nginx 4.5s
=> [app-base 1/8] FROM docker.io/library/node:l2-alpin 0.0s
=> [base 1/4] FROM docker.io/library/python:alpine@sha 0.0s
=> [internal] load build context 0.2s
=> => transferring context: 7.70kB 0.1s
=> CACHED [stage-5 1/3] FROM docker.io/library/nginx:a 0.0s
=> CACHED [base 2/4] WORKDIR /app 0.0s
=> CACHED [base 3/4] COPY requirements.txt . 0.0s
=> CACHED [base 4/4] RUN pip install -r requirements.t 0.0s
=> CACHED [build 1/2] COPY . . 0.0s
=> CACHED [build 2/2] RUN makedocs build 0.0s
=> CACHED [app-base 2/8] RUN apk add --no-cache python 0.0s
=> CACHED [app-base 3/8] WORKDIR /app 0.0s
=> CACHED [app-base 4/8] COPY app/package.json app/yar 0.0s
=> [app-base 5/8] RUN yarn install 59.1s
=> [app-base 6/8] COPY app/spec ./spec 0.2s
=> [app-base 7/8] COPY app/src ./src 0.3s
=> [app-base 8/8] RUN yarn test 11.0s
=> [app-zip-creator 1/1] RUN rm -rf node_modules && 4.8s
=> [stage-5 2/3] COPY --from=app-zip-creator /app.zip 0.2s
=> [stage-5 3/3] COPY --from=build /app/site /usr/shar 0.3s
=> exporting to image 0.2s
=> => exporting layers 0.2s
=> => writing image sha256:0dc7cc3653ec647a6e15a28f025 0.0s
=> => naming to docker.io/library/docker101tutorial 0.0s

Use 'docker scan' to run Snyk tests against images to find vul
nerabilities and learn how to fix them
PS C:\Users\controlc\getting-started> docker run -d -p 80:80 -
--name docker-tutorial docker101tutorial
90af5e99a6393773f5fabe9a989964cf1116222057a85b24e1404e3c976173
26
PS C:\Users\controlc\getting-started>
```


[실습 환경 구성] – Docker tutorial

- 5. 생성된 이미지를 Docker Hub에 업로드

- Docker Hub 아이디로 로그인
- 아이디가 없다면 하단의 hub.docker.com 링크를 클릭하여 가입
- 가입 시 free 요금제 선택

A screenshot of a Docker application window titled "Docker". The window displays the Docker logo (a blue whale with a stack of containers) on the left. On the right, the text "Sign in with your Docker ID" is prominently displayed. Below this text are two input fields: "Docker ID" and "Password". A "Sign in" button is located to the right of the password field. At the bottom of the window, there is a message: "If you don't have a Docker ID yet, you can create one on [hub.docker.com](\"https://hub.docker.com\")".

Docker

Sign in with your Docker ID

Docker ID

Password

Sign in

If you don't have a Docker ID yet, you can create one on hub.docker.com

[실습 환경 구성] – Docker tutorial

- 5. 생성된 이미지를 Docker Hub에 업로드

파란색 박스를 클릭하여 명령 실행


- `docker tag`
`docker101tutorial`
[아이디]/`docker101tutorial`
→ `docker101tutorial` 이미지에
“[아이디]/`docker101tutorial`” 이라는 tag를 붙이기
- `docker push`
[아이디]/`docker101tutorial`
→ Docker Hub에
“[아이디]/`docker101tutorial`” tag를 붙인
이미지를 push(업로드)


The screenshot shows the Docker Desktop interface. On the left, a sidebar lists steps: Clone, Build, Run, and Share (highlighted with a blue circle and number 4). The main area is titled 'Now save and share your image' and contains the instruction: 'Save and share your image on Docker Hub to enable other users to easily download and run the image on any destination machine.' Below this, a blue button with a red border contains the commands: `docker tag docker101tutorial noburi/docker101tutorial` and `docker push noburi/docker101tutorial`. A red box highlights this button. Below the button is a link: 'See what you've saved on Hub'. At the bottom left is a 'Skip tutorial' link, and at the bottom center is a 'Done' button. On the right, a terminal window shows the execution of these commands. The output includes: `exporting layers`, `writing image sha256:0dc7cc3653ec647a6e15a28f025`, `naming to docker.io/library/docker101tutorial`, `Error response from daemon: No such image: mytutorial:latest`, `docker tag docker101tutorial noburi/docker101tutorial`, `docker push noburi/docker101tutorial`, and the final push output showing layers being pushed and mounted.


[실습 환경 구성] – Docker tutorial

- 5. 생성된 이미지를 Docker Hub에 업로드

Docker hub에 접속하여
Push(업로드) 된 이미지
확인

 **noburi / docker101tutorial**

This repository does not have a description 

 Last pushed: 2 minutes ago


Docker commands

[Public View](#)



To push a new tag to this repository,

```
docker push noburi/docker101tutorial:tagname
```

Tags and Scans

 VULNERABILITY SCANNING - DISABLED [Enable](#)

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
 latest		2 minutes ago	3 minutes ago

[See all](#)

Recent builds

Link a source provider and run a build to see build results here.

[실습 환경 구성] – Docker tutorial

- 5. 생성된 이미지를 Docker Hub에 업로드

“Done”을 클릭하여
Tutorial 종료

docker tag docker101tutorial noburi/docker101tutorial
docker push noburi/docker101tutorial

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.

See what you've saved on Hub

Click!!

Done

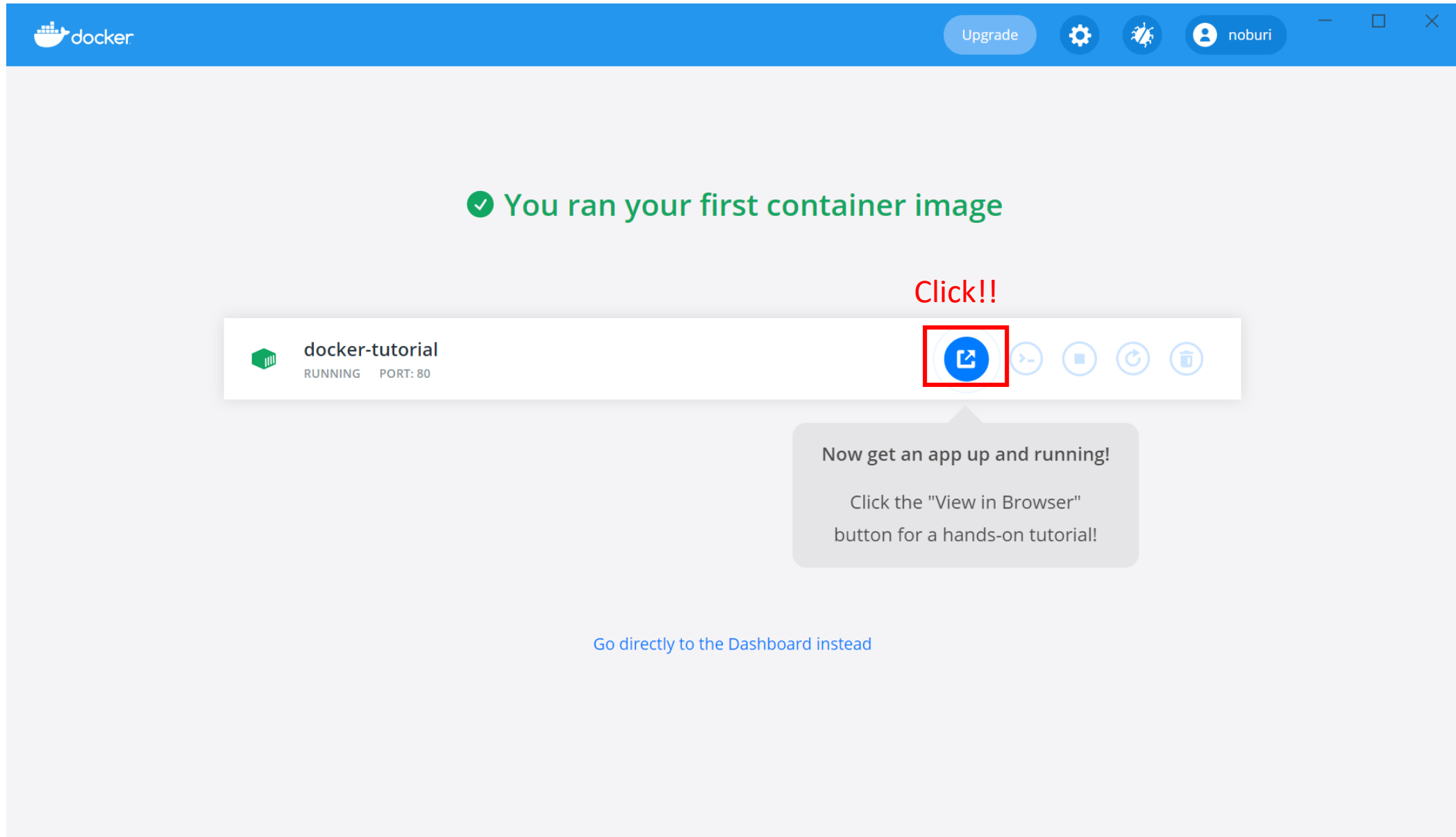
Skip tutorial

```
=> => exporting layers 0.2s
=> => writing image sha256:0dc7cc3653ec647a6e15a28f025 0.0s
=> => naming to docker.io/library/docker101tutorial 0.0s

Use 'docker scan' to run Snyk tests against images to find vul
nerabilities and learn how to fix them
PS C:\Users\controlc\getting-started> docker run -d -p 80:80 -
-name docker-tutorial docker101tutorial
90af5e99a6393773f5fabe9a989964cf1116222057a85b24e1404e3c976173
26
PS C:\Users\controlc\getting-started> docker tag mytutorial no
buri/mytutorial
Error response from daemon: No such image: mytutorial:latest
PS C:\Users\controlc\getting-started> docker tag docker101tuto
rial noburi/docker101tutorial
PS C:\Users\controlc\getting-started> docker push noburi/docke
r101tutorial
Using default tag: latest
The push refers to repository [docker.io/noburi/docker101tutor
ial]
ec341728483a: Preparing
02be27bfb42: Preparing
4689e8eca613: Preparing
3480549413ea: Preparing
3c369314e003: Preparing
4531e200ac8d: Waiting
ed3fe3f2b59f: Waiting
b2d5eeeaba3a: Waiting
denied: requested access to the resource is denied
PS C:\Users\controlc\getting-started> docker tag docker101tuto
rial noburi/docker101tutorial
PS C:\Users\controlc\getting-started> docker push noburi/docke
r101tutorial
Using default tag: latest
The push refers to repository [docker.io/noburi/docker101tutor
ial]
ec341728483a: Pushed
02be27bfb42: Pushed
4689e8eca613: Mounted from library/nginx
3480549413ea: Mounted from library/nginx
3c369314e003: Mounted from library/nginx
4531e200ac8d: Mounted from library/nginx
ed3fe3f2b59f: Mounted from library/nginx
b2d5eeeaba3a: Mounted from library/python
latest: digest: sha256:4d1dc3619a19410736fcd9bffc33fcc48d6dd
2fddc84077aa89220882764fa size: 1990
PS C:\Users\controlc\getting-started>
```


[실습 환경 구성] – Docker tutorial

- 6. 실행된 컨테이너 작동 확인




- 6. 실행된 컨테이너 작동 확인

localhost/tutorial/

 docker Labs

Getting Started

 Search

Getting Started

Getting Started

Our Application

Updating our App

Sharing our App

Persisting our DB

Using Bind Mounts

Multi-Container Apps

Using Docker Compose

Image Building Best Practices

What Next?

Getting Started

The command you just ran

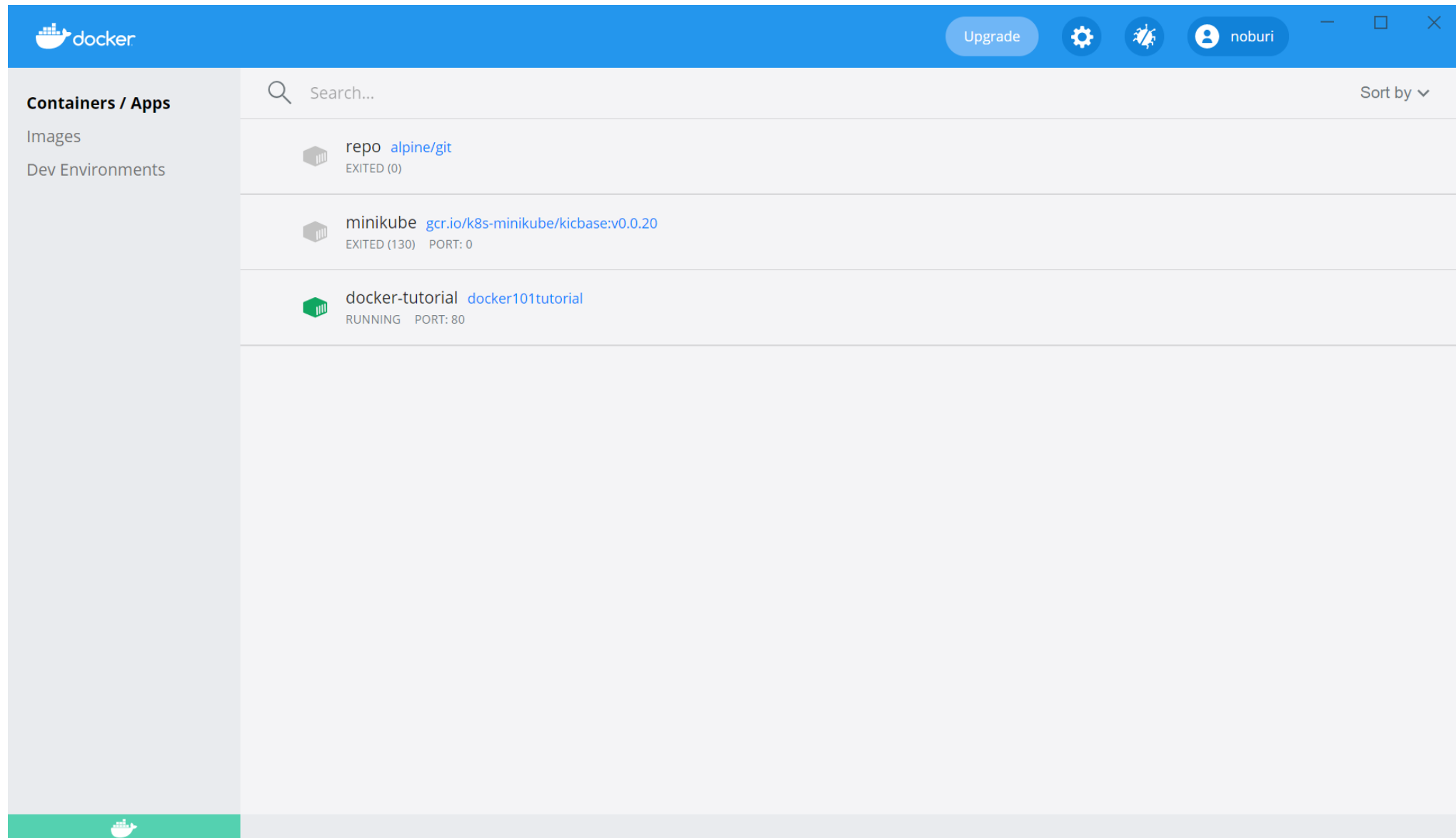
Congratulations! You have started the container for this tutorial! Let's first explain the command that you just ran. In case you forgot, here's the command:

```
docker run -d -p 80:80 docker/getting-started
```

You'll notice a few flags being used. Here's some more info on them:

- `-d` - run the container in detached mode (in the background)
- `-p 80:80` - map port 80 of the host to port 80 in the container
- `docker/getting-started` - the image to use

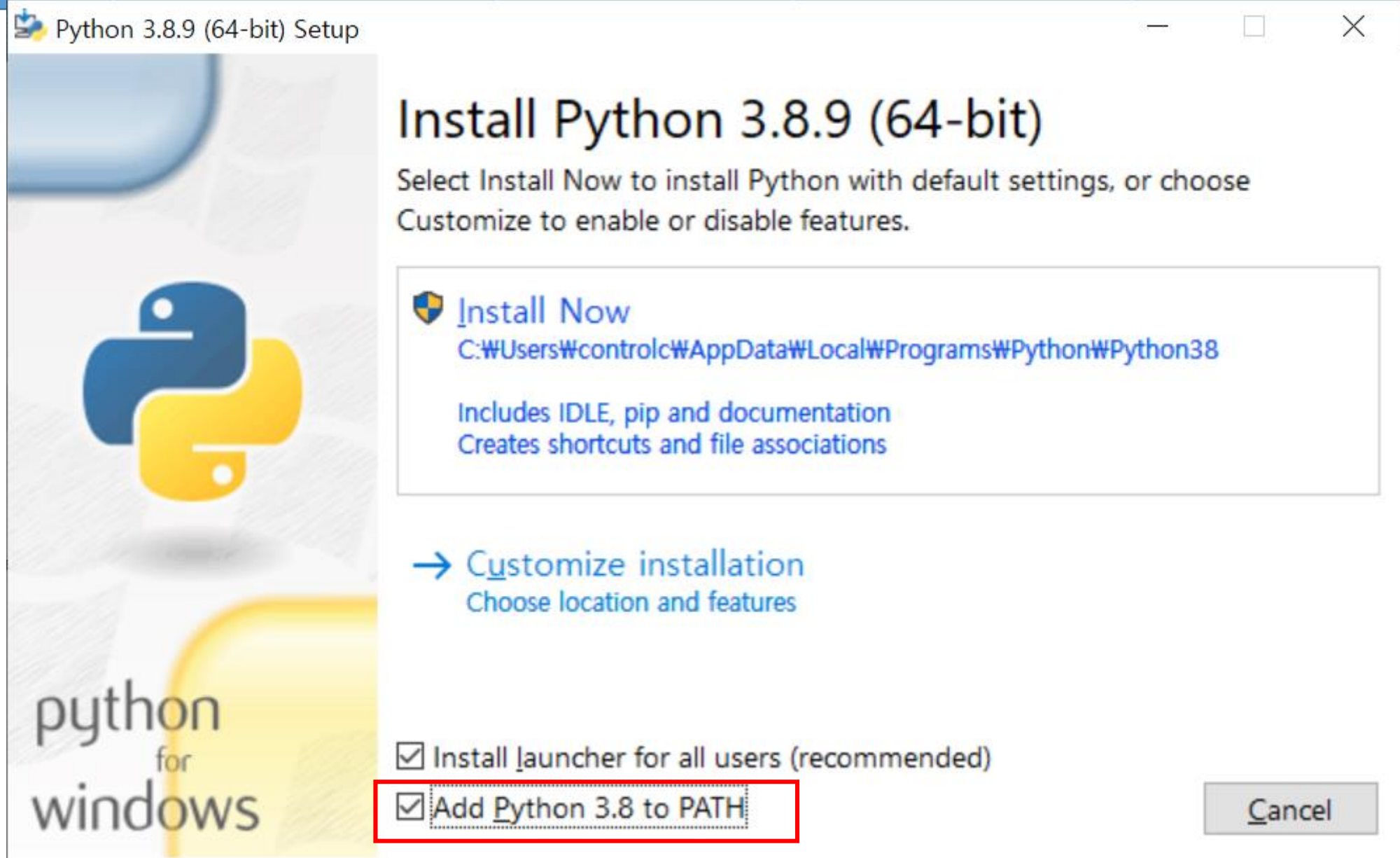
[실습 환경 구성] – Docker tutorial



- 실습을 위한 Python 설치

<https://www.python.org/downloads/>

<https://www.python.org/downloads/release/python-389/>



■ 전체 실습 구성

1. 소스 프로그램 작성

실습용 디렉토리 만들기 ➔ 파이썬 소스 프로그램 작성 ➔ 실행 결과 확인

2. Docker 이미지 만들기

Dockerfile 생성 ➔ Docker 이미지 만들기 ➔ Docker 컨테이너 실행

3. Docker 이미지 수정

파이썬 소스 수정 ➔ Docker 이미지 재생성 ➔ Docker 컨테이너 실행(변경 확인)

4. Docker Hub에 이미지 올리기

Docker Hub 계정 생성 ➔ Repository 생성 ➔ Docker 로그인 ➔ 이미지 올리기

5. Kubernetes 실습

이미지로 컨테이너 생성 ➔ 서비스 접근

Docker Hub에서 이미지 받기 ➔ 컨테이너 생성 ➔ 서비스 열기 ➔ 접속 확인

- 실습용 디렉토리 만들기

아래 명령을 실행

```
# mkdir c:\docker_lab
```

```
# mkdir c:\docker_lab\python-app-v1
```

```
# cd c:\docker_lab\python-app-v1
```

- Python 프로그램 작성
app.py 파일을 아래 내용으로 생성(c:\docker_lab\python-app-v1)

```
import streamlit as st
import socket
import sys

hostname = socket.gethostname()
local_ip = socket.gethostbyname(hostname)

st.title('hello world by ' + local_ip)
x = st.slider('x')
st.write(x, 'squared is', x * x)
```

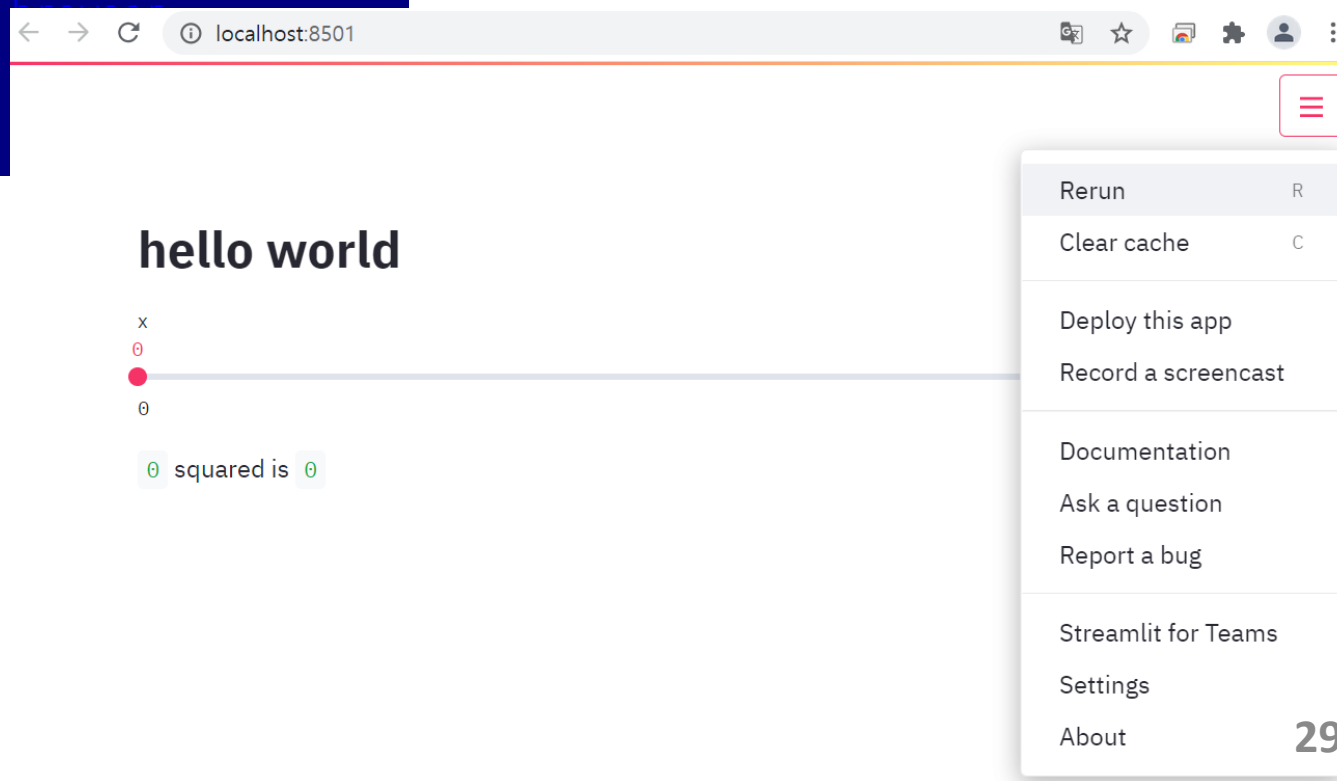
Docker 실습 1

- 실행 결과 확인
 - 파워셸에서 아래 명령 실행
 - `python -m streamlit run app.py`

```
Windows PowerShell
PS C:\#docker_lab\python-app-v1> python -m streamlit run app.py

You can now view your Streamlit app in your browser.

  http://localhost:8501
  http://10.32.4.31:8501
```



- Docker 이미지 만들기
 - Dockerfile 생성(파일명: Dockerfile)

```
# syntax=docker/dockerfile:1

FROM python:3.9-slim-buster

EXPOSE 8501

WORKDIR /app

RUN pip install streamlit

COPY ..

CMD [ "streamlit", "run", "app.py"]
```

- 디렉토리 파일 확인

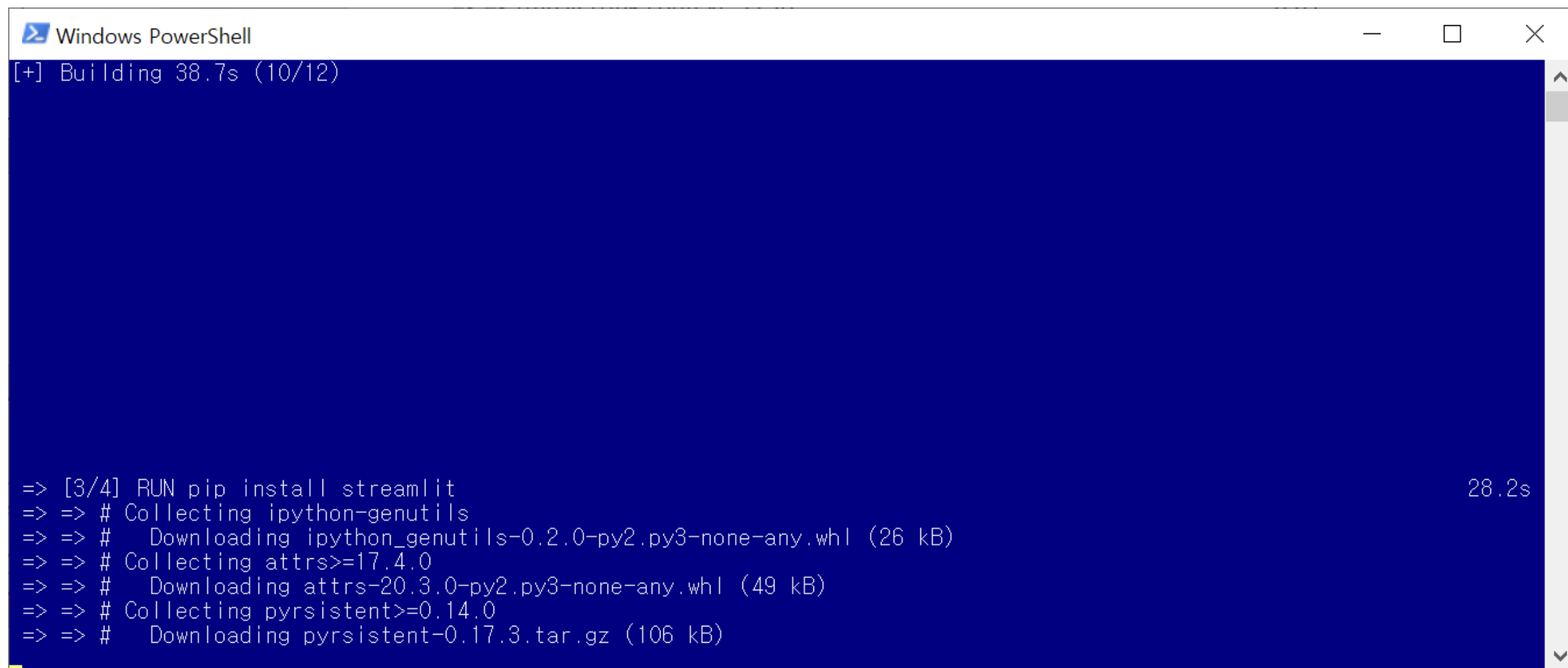
☐ 이름

 app.py

 Dockerfile

■ Docker 이미지 만들기

- `docker build -t python-streamlit:001 .` (파워셸에서 실행)

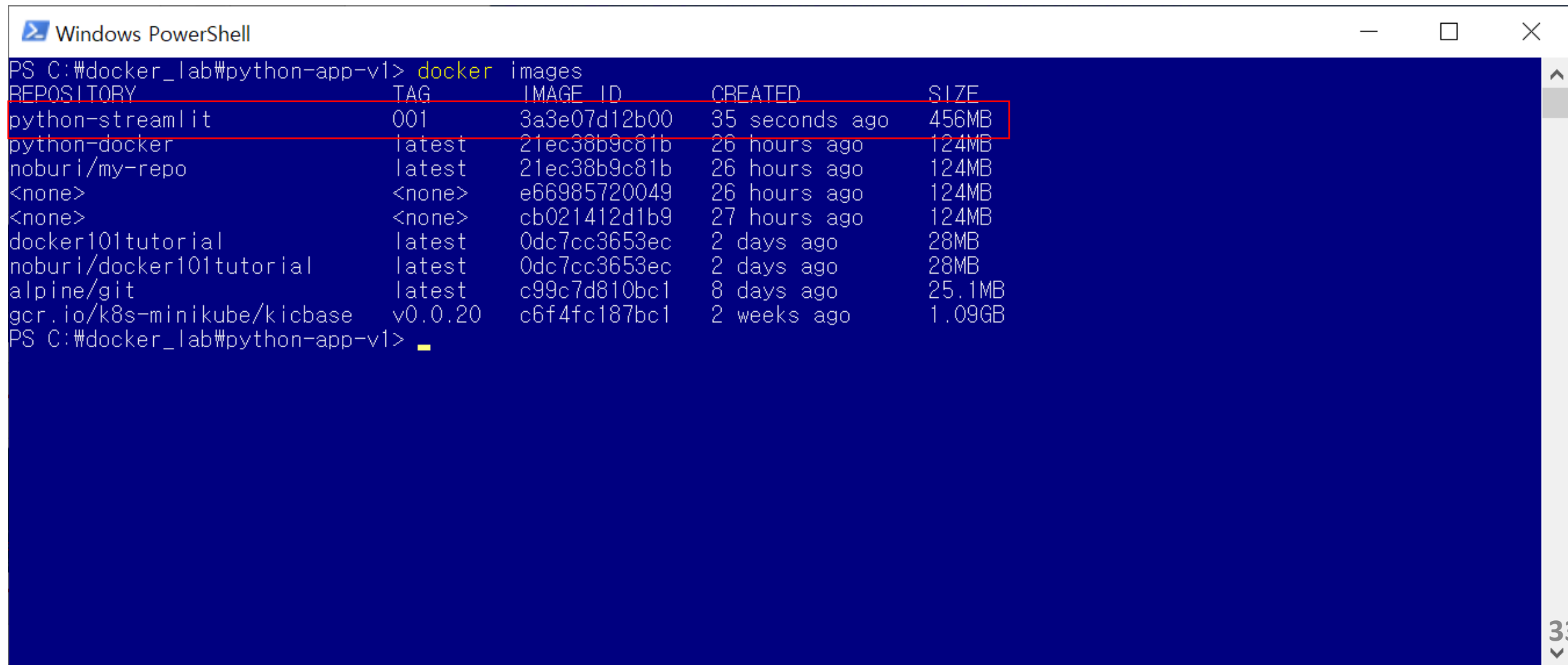


```
Windows PowerShell
[+] Building 38.7s (10/12)

=> [3/4] RUN pip install streamlit 28.2s
=> => # Collecting ipython-genutils
=> => #   Downloading ipython_genutils-0.2.0-py2.py3-none-any.whl (26 kB)
=> => # Collecting attrs>=17.4.0
=> => #   Downloading attrs-20.3.0-py2.py3-none-any.whl (49 kB)
=> => # Collecting pyrsistent>=0.14.0
=> => #   Downloading pyrsistent-0.17.3.tar.gz (106 kB)
```


Docker 실습 2

- Docker 이미지 만들기
 - 생성된 이미지 확인
 - docker images 명령 실행



```
Windows PowerShell
PS C:\#docker_lab\python-app-v1> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
python-streamlit     001                 3a3e07d12b00       35 seconds ago     456MB
python-docker        latest              21ec38b9c81b       26 hours ago       124MB
noburi/my-repo       latest              21ec38b9c81b       26 hours ago       124MB
<none>               <none>              e66985720049       26 hours ago       124MB
<none>               <none>              cb021412d1b9       27 hours ago       124MB
docker101tutorial    latest              0dc7cc3653ec       2 days ago         28MB
noburi/docker101tutorial latest              0dc7cc3653ec       2 days ago         28MB
alpine/git           latest              c99c7d810bc1       8 days ago         25.1MB
gcr.io/k8s-minikube/kicbase v0.0.20            c6f4fc187bc1       2 weeks ago        1.09GB
PS C:\#docker_lab\python-app-v1>
```

- Docker 컨테이너에서 이미지 실행
 - `docker run --name python-streamlit --publish 8501:8501 python-streamlit:001`

```
Windows PowerShell
PS C:\#docker_lab\python-app-v1> docker run --name python-streamlit --publish 8501:8501 python-streamlit:001

You can now view your Streamlit app in your browser.

Network URL: http://172.17.0.3:8501
External URL: http://203.250.61.14:8501
```

hello world

x

0

0

100

0 squared is 0

- Docker 이미지 수정
 - 프로그램 소스파일 수정
 - app.py 파일 수정

```
import streamlit as st
import socket
import sys

hostname = socket.gethostname()
local_ip = socket.gethostbyname(hostname)

st.title('docker lab 1.1 by ' + local_ip)
x = st.slider('x')
st.write(x, 'squared is', x * x)
```

■ 재 빌드 및 배포 (28쪽 부터 참조)

1. 컨테이너 종료

1. docker ps 로 컨테이너 이름 확인 후 종료
2. docker stop [컨테이너명]
3. docker rm [컨테이너명]

2. 이미지 재 빌드

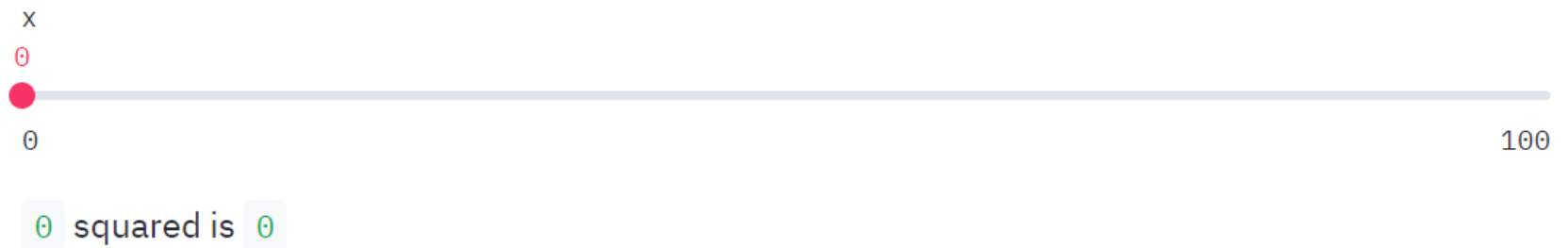
docker build -t python-streamlit:001 .

3. 컨테이너 실행

docker run --name python-streamlit --publish 8501:8501 python-streamlit:001

4. 결과 확인

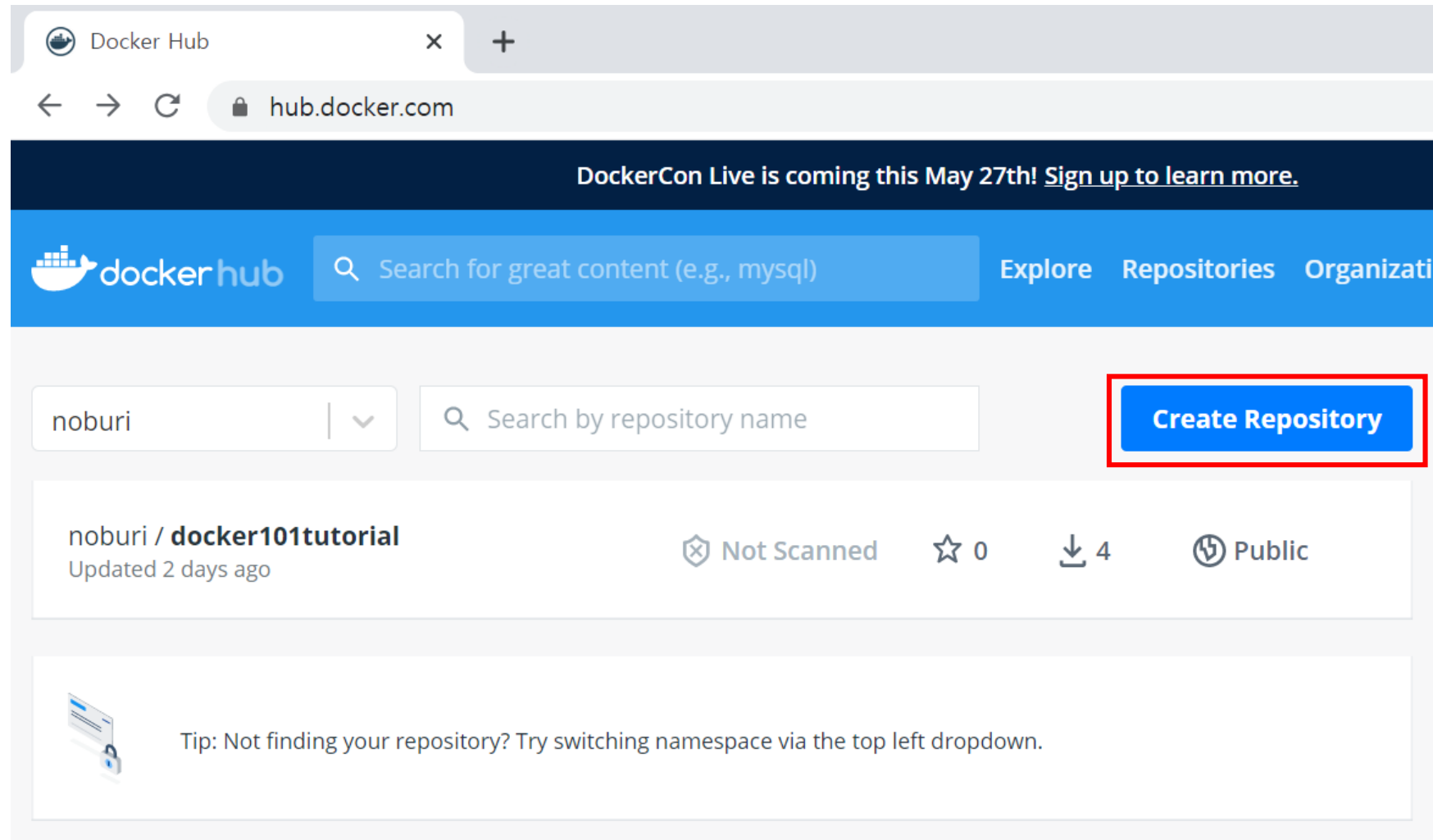
docker lab 1.1



- Docker Hub에 이미지 올리기
 1. Docker Hub 계정 생성
 - 회원 가입 후 로그인
 - Repositories로 이동

■ Docker Hub에 이미지 올리기

2. Repository 생성



■ Docker Hub에 이미지 올리기

2. Repository 생성

Repositories

Create

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

Create Repository

noburi

my-repo

개인 레파지토리

Visibility

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

☐ Public

☒ Private

Public repositories appear in Docker Hub search results

Only you can view private repositories

Pro tip

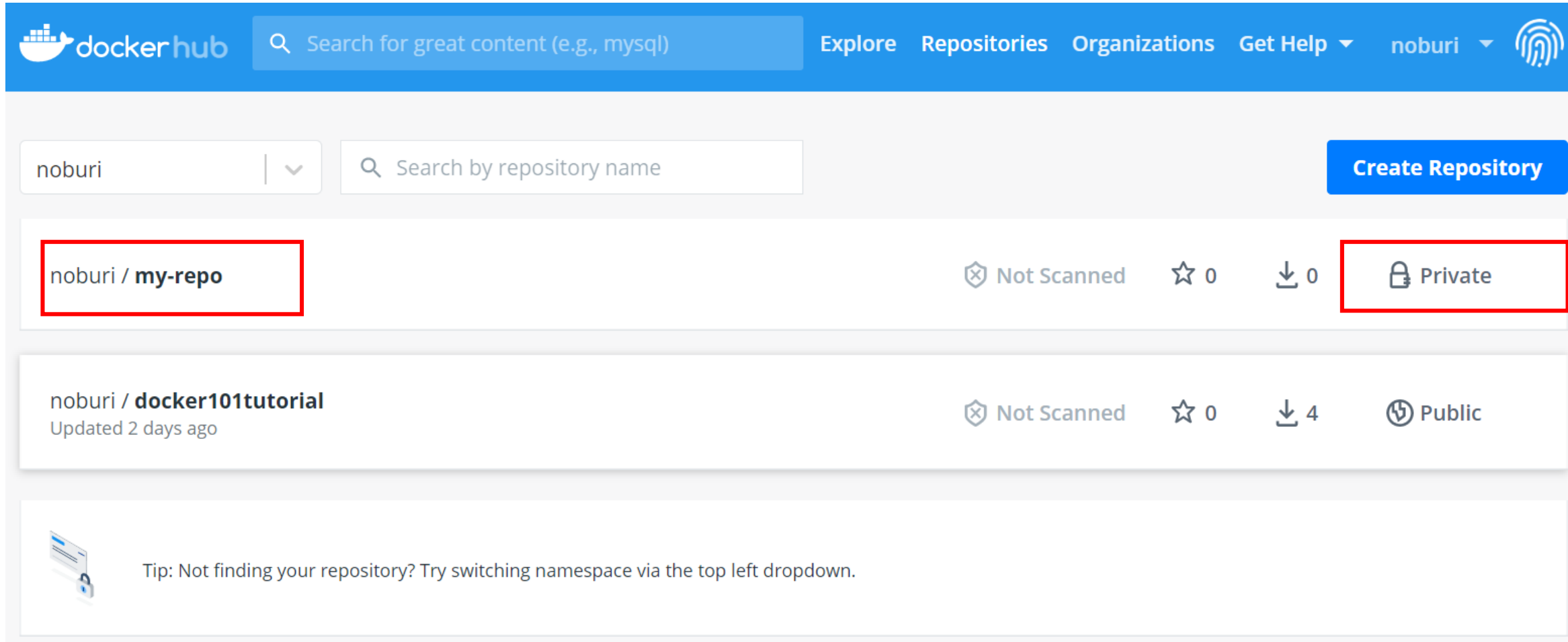
You can push a new image to this repository using the CLI

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

Make sure to change *tagname* with your desired image repository tag.

■ Docker Hub에 이미지 올리기

2. Repository 생성




The screenshot shows the Docker Hub interface. At the top is a blue navigation bar with the Docker Hub logo, a search bar, and links for Explore, Repositories, Organizations, Get Help, and a user profile for 'noburi'. Below the navigation bar is a light gray section with a dropdown menu set to 'noburi' and a search bar labeled 'Search by repository name'. To the right of this is a blue 'Create Repository' button. Below these elements is a list of repositories. The first repository, 'noburi / my-repo', is highlighted with a red box. It shows 'Not Scanned', 0 stars, 0 downloads, and is marked as 'Private' (also highlighted with a red box). The second repository, 'noburi / docker101tutorial', is updated 2 days ago, shows 'Not Scanned', 0 stars, 4 downloads, and is marked as 'Public'. At the bottom, there is a tip: 'Tip: Not finding your repository? Try switching namespace via the top left dropdown.'

dockerhub Search for great content (e.g., mysql) Explore Repositories Organizations Get Help ▾ noburi ▾

noburi ▾ Search by repository name Create Repository

noburi / **my-repo** Not Scanned ☆ 0 ↓ 0 Private

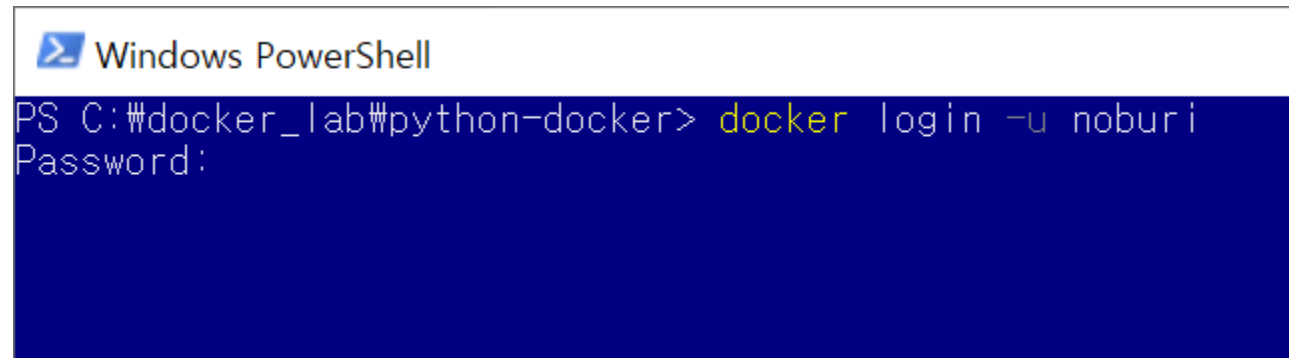
noburi / **docker101tutorial**
Updated 2 days ago Not Scanned ☆ 0 ↓ 4 Public

 Tip: Not finding your repository? Try switching namespace via the top left dropdown.

■ Docker Hub에 이미지 올리기

3. Docker login

```
docker login -u [아이디] -p [비밀번호]
```

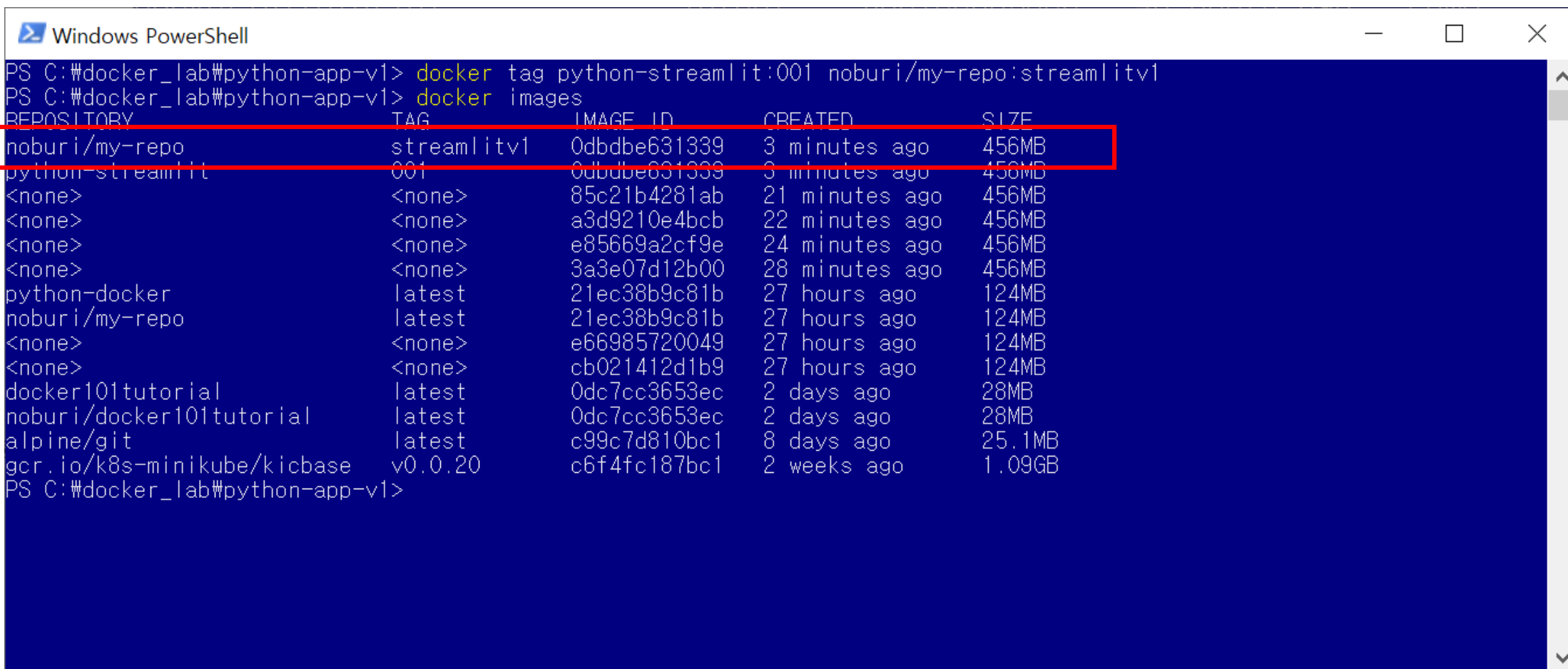


```
Windows PowerShell
PS C:\#docker_lab\python-docker> docker login -u noburi
Password:
```

■ Docker Hub에 이미지 올리기

4. 이미지 올리기

```
docker tag python-streamlit:001 noburi/my-repo:streamlitv1
```



A screenshot of a Windows PowerShell terminal window. The title bar reads "Windows PowerShell". The terminal shows the following commands and output:

```
PS C:\#docker_lab#python-app-v1> docker tag python-streamlit:001 noburi/my-repo:streamlitv1
PS C:\#docker_lab#python-app-v1> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
noburi/my-repo	streamlitv1	0dbdbe631339	3 minutes ago	456MB
python-streamlit	001	0dbdbe631339	3 minutes ago	456MB
<none>	<none>	85c21b4281ab	21 minutes ago	456MB
<none>	<none>	a3d9210e4bcb	22 minutes ago	456MB
<none>	<none>	e85669a2cf9e	24 minutes ago	456MB
<none>	<none>	3a3e07d12b00	28 minutes ago	456MB
python-docker	latest	21ec38b9c81b	27 hours ago	124MB
noburi/my-repo	latest	21ec38b9c81b	27 hours ago	124MB
<none>	<none>	e66985720049	27 hours ago	124MB
<none>	<none>	cb021412d1b9	27 hours ago	124MB
docker101tutorial	latest	0dc7cc3653ec	2 days ago	28MB
noburi/docker101tutorial	latest	0dc7cc3653ec	2 days ago	28MB
alpine/git	latest	c99c7d810bc1	8 days ago	25.1MB
gcr.io/k8s-minikube/kicbase	v0.0.20	c6f4fc187bc1	2 weeks ago	1.09GB

```
PS C:\#docker_lab#python-app-v1>
```

■ Docker Hub에 이미지 올리기

4. 이미지 올리기


`docker push [아이디]/my-repo:streamlitv1`


```
Windows PowerShell
noburi/my-repo      streamlitv1  0dbdbe631339  3 minutes ago  456MB
python-streamlit    001         0dbdbe631339  3 minutes ago  456MB
<none>             <none>      85c21b4281ab  21 minutes ago  456MB
<none>             <none>      a3d9210e4bcb  22 minutes ago  456MB
<none>             <none>      e85669a2cf9e  24 minutes ago  456MB
<none>             <none>      3a3e07d12b00  28 minutes ago  456MB
python-docker       latest      21ec38b9c81b  27 hours ago   124MB
noburi/my-repo      latest      21ec38b9c81b  27 hours ago   124MB
<none>             <none>      e66985720049  27 hours ago   124MB
<none>             <none>      cb021412d1b9  27 hours ago   124MB
docker101tutorial   latest      0dc7cc3653ec  2 days ago     28MB
noburi/docker101tutorial latest      0dc7cc3653ec  2 days ago     28MB
alpine/git          latest      c99c7d810bc1  8 days ago     25.1MB
gcr.io/k8s-minikube/kicbase v0.0.20    c6f4fc187bc1  2 weeks ago    1.09GB
PS C:\#docker_lab\python-app-v1> docker push noburi/my-repo:streamlitv1
The push refers to repository [docker.io/noburi/my-repo]
68fe9c67da78: Pushed
a88a652acab6: Pushing [=====>] 45.38MB/340.9MB
e714743a0607: Pushed
35990bbb67bd: Mounted from library/python
be3883e87d34: Layer already exists
9ec86c039eae: Layer already exists
371ce8b24b31: Layer already exists
7e718b9c0c8c: Layer already exists
```

■ Docker Hub에 이미지 올리기

4. 이미지 올리기(push된 이미지 확인)

 noburi / my-repo

개인 레파지토리 


 Last pushed: a minute ago

Docker commands





To push a new tag to this repository,

```
docker push noburi/my-repo:tagname
```

Tags and Scans

 VULNERABILITY SCANNING - DISABLED [Enable](#)

This repository contains 2 tag(s).

TAG	OS	PULLED	PUSHED
 latest		2 hours ago	3 hours ago
 streamlitv1		a minute ago	a minute ago

[See all](#)

Recent builds

Link a source provider and run a build to see build results here.

Minikube 설치

- <https://minikube.sigs.k8s.io/docs/start/>

- Kubernetes 실습을 위해 설치되는 minikube는 가상화 기반 또는 컨테이너 환경이 필요함
- 운영체제에 따라 우측에 나열된 기반 환경이 추가로 필요
- 이번 실습은 Windows 10 Pro에 Hyper-V를 기반으로 진행

Linux

- [Docker](#) - container-based (preferred)
- [KVM2](#) - VM-based (preferred)
- [VirtualBox](#) - VM
- [None](#) - bare-metal
- [Podman](#) - container (experimental)
- [SSH](#) - remote ssh

macOS

- [Docker](#) - VM + Container (preferred)
- [Hyperkit](#) - VM
- [VirtualBox](#) - VM
- [Parallels](#) - VM
- [VMware](#) - VM
- [SSH](#) - remote ssh

Windows

- [Hyper-V](#) - VM (preferred)
- [Docker](#) - VM + Container (preferred)
- [VirtualBox](#) - VM
- [SSH](#) - remote ssh

Minikube 설치

1 Installation

Linux

macOS

Windows

Windows Package Manager

If the [Windows Package Manager](#) is installed, use the following command to install minikube:

```
winget install minikube
```

Chocolatey

If the [Chocolatey Package Manager](#) is installed, use the following command:

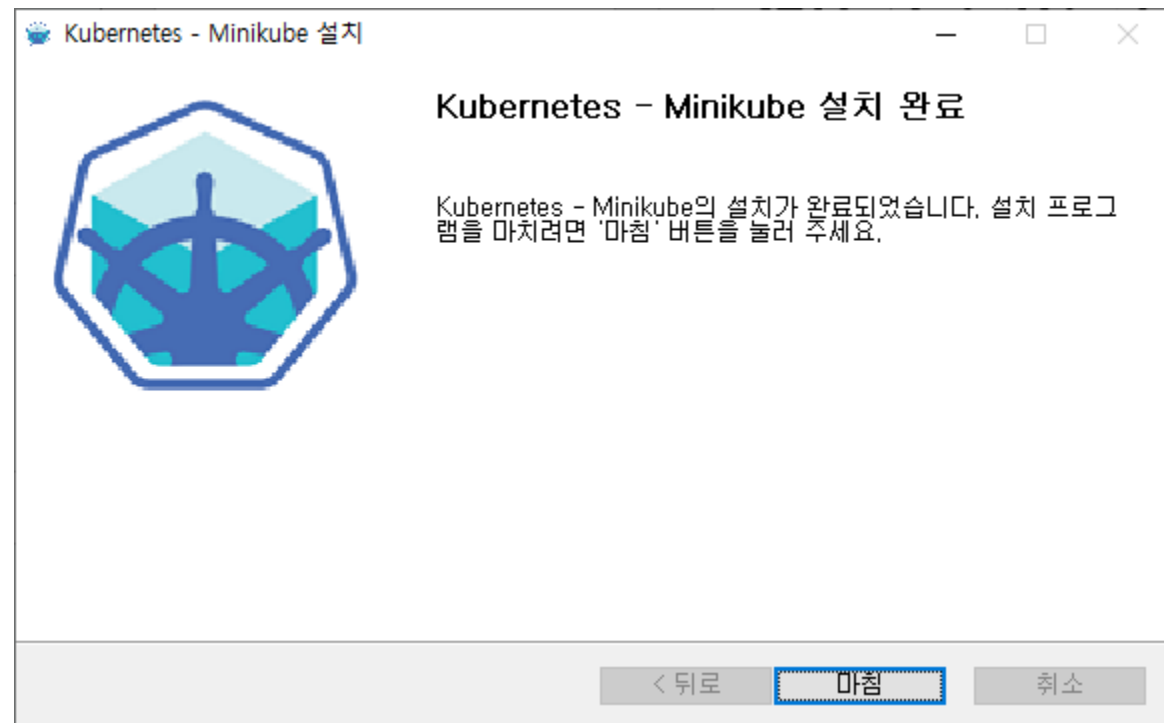
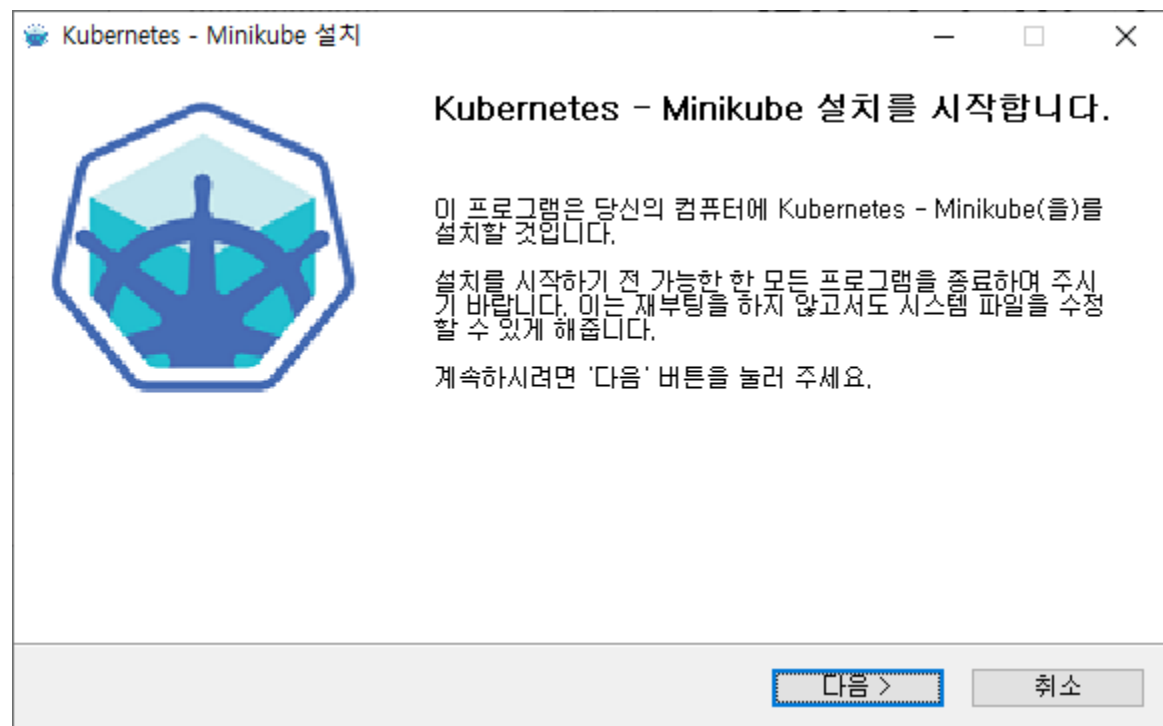
```
choco install minikube
```

Stand-alone Windows Installer

Otherwise, download and run the [Windows installer](#)

If you used a CLI to perform the installation, you will need to close that CLI and open a new one before proceeding.

Minikube 설치



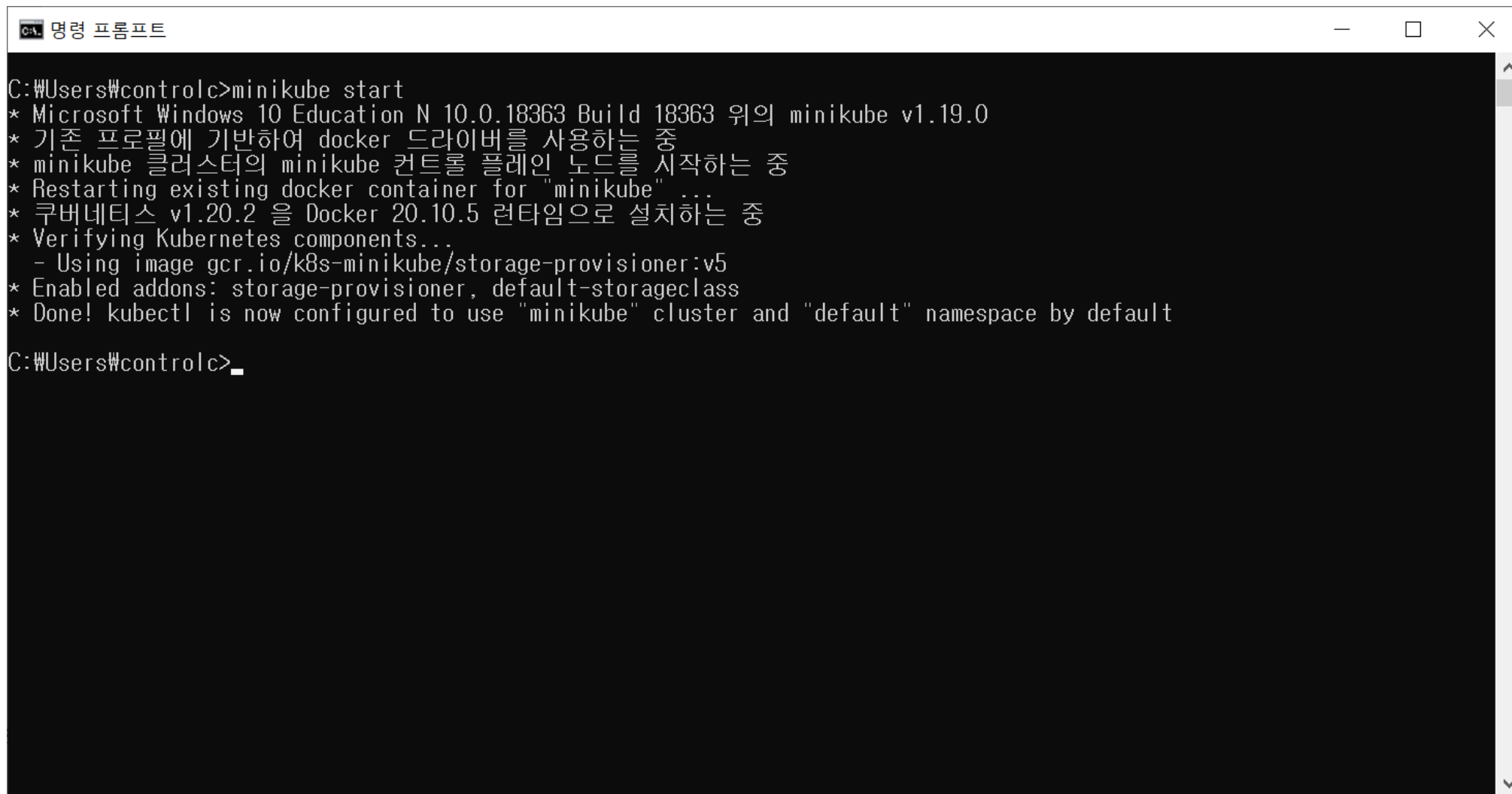
[기초 환경 구성]

- BIOS에서 VT-X/AMD-V enable 필요
- Hyper-V 활성화 (명령창을 관리자 권한으로 실행하고 아래 명령!!!!)
DISM /Online /Enable-Feature /All /FeatureName:Microsoft-Hyper-V
끄기
bcdedit /set hypervisorlaunchtype off
켜기
bcdedit /set hypervisorlaunchtype auto

가상머신 시작 (반드시 관리자권한으로 실행)
minikube start --driver=hyperv

- Windows 10 Ent, Pro, Edu 버전이 아니라면 Hyper-V 불가
→ Virtual Box 설치 필요
virtual box를 사용
minikube start --driver=virtualbox

- Minikube 실행하기
 - # minikube start

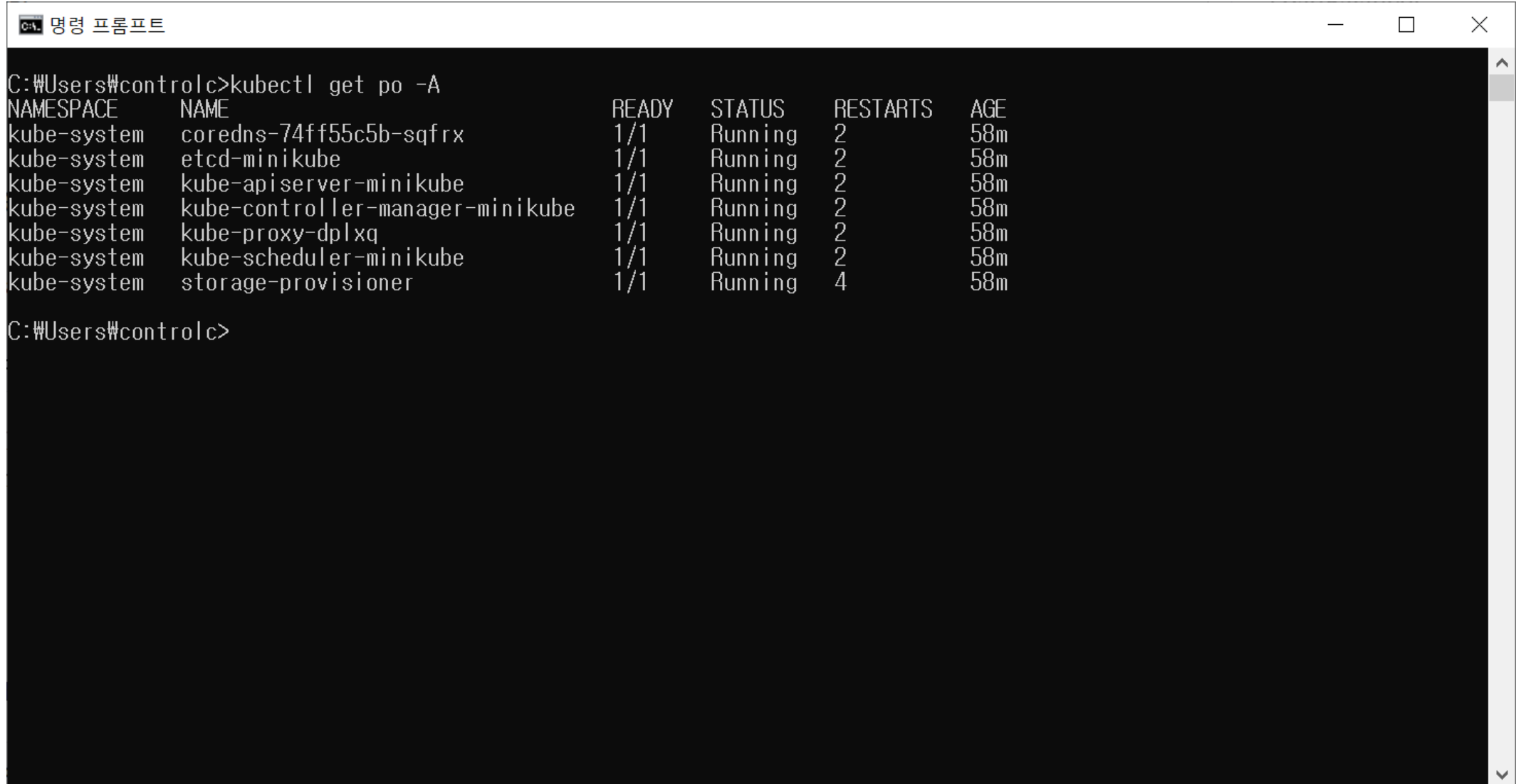


```
C:\> 명령 프롬프트

C:\Users\controlc>minikube start
* Microsoft Windows 10 Education N 10.0.18363 Build 18363 위의 minikube v1.19.0
* 기존 프로파일에 기반하여 docker 드라이버를 사용하는 중
* minikube 클러스터의 minikube 컨트롤 플레인 노드를 시작하는 중
* Restarting existing docker container for "minikube" ...
* 쿠버네티스 v1.20.2 을 Docker 20.10.5 런타임으로 설치하는 중
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

C:\Users\controlc>_
```

- 실행 중인 POD 확인
kubectl get po -A



```
CA 명령 프롬프트
C:\Users\controlc>kubectl get po -A
NAMESPACE      NAME                                READY   STATUS    RESTARTS   AGE
kube-system    coredns-74ff55c5b-sqfrx            1/1     Running   2           58m
kube-system    etcd-minikube                      1/1     Running   2           58m
kube-system    kube-apiserver-minikube            1/1     Running   2           58m
kube-system    kube-controller-manager-minikube   1/1     Running   2           58m
kube-system    kube-proxy-dplxq                   1/1     Running   2           58m
kube-system    kube-scheduler-minikube            1/1     Running   2           58m
kube-system    storage-provisioner                1/1     Running   4           58m

C:\Users\controlc>
```

- WEB UI 확인하기
- # minikube dashboard

Kubernetes Dashboard

127.0.0.1:52147/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/overview?namespace=default

kubernetes

default

검색

Overview

워크로드

크론 잡

데몬 셋

디플로이먼트

잡

파드

레플리카 셋

레플리케이션 컨트롤러

스테이트풀 셋

서비스

인그레스

서비스

컨피그 및 스토리지

컨피그 맵

퍼시스턴트 볼륨 클레임

시크릿

스토리지 클래스

클러스터

클러스터 롤 바인딩

클러스터 롤

서비스

이름	네임스페이스	레이블	클러스터 IP	내부 엔드포인트	외부 엔드포인트	생성 시간
kubernetes	default	<div>component: apiserver</div> <div>provider: kubernetes</div>	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	an hour ago

1 - 1 of 1

컨피그 및 스토리지

컨피그 맵

이름	네임스페이스	레이블	생성 시간
kube-root-ca.crt	default	-	an hour ago

1 - 1 of 1

Secrets

이름	네임스페이스	레이블	타입	생성 시간
default-token-4kl49	default	-	kubernetes.io/service-account-token	an hour ago

- `kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.4`
- `kubectl expose deployment hello-minikube --type=NodePort --port=8080`

- `kubectl get services hello-minikube`
- `minikube service hello-minikube`

```
명령 프롬프트 - minikube service hello-minikube

C:\Users\controlc>kubectl get services hello-minikube
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
hello-minikube      NodePort    10.110.164.235 <none>         8080:30725/TCP   3m16s

C:\Users\controlc>minikube service hello-minikube
-----
| NAMESPACE | NAME          | TARGET PORT | URL                               |
|-----|-----|-----|-----|
| default   | hello-minikube | 8080        | http://192.168.49.2:30725      |
|-----|-----|-----|-----|
* hello-minikube 서비스의 터널을 시작하는 중
| NAMESPACE | NAME          | TARGET PORT | URL                               |
|-----|-----|-----|-----|
| default   | hello-minikube |             | http://127.0.0.1:52301        |
|-----|-----|-----|-----|
* Opening service default/hello-minikube in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

■ minikube service hello-minikube

← → ↻ ⓘ 127.0.0.1:52301

CLIENT VALUES:

client_address=172.17.0.1
command=GET
real path=/
query=nil
request_version=1.1
request_uri=http://127.0.0.1:8080/

SERVER VALUES:

server_version=nginx: 1.10.0 - lua: 10001

HEADERS RECEIVED:

accept=text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
accept-encoding=gzip, deflate, br
accept-language=ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
connection=keep-alive
host=127.0.0.1:52301
sec-ch-ua=" Not A;Brand";v="99", "Chromium";v="90", "Google Chrome";v="90"
sec-ch-ua-mobile=?0
sec-fetch-dest=document
sec-fetch-mode=navigate
sec-fetch-site=none
sec-fetch-user=?1
upgrade-insecure-requests=1
user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36
BODY:
-no body in request-

- `kubectl port-forward service/hello-minikube 7080:8080`

← → ↻ ⓘ 127.0.0.1:7080

CLIENT VALUES:

client_address=127.0.0.1
command=GET
real path=/
query=nil
request_version=1.1
request_uri=http://127.0.0.1:8080/

SERVER VALUES:

server_version=nginx: 1.10.0 - lua: 10001

HEADERS RECEIVED:

accept=text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
accept-encoding=gzip, deflate, br
accept-language=ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
connection=keep-alive
host=127.0.0.1:7080
sec-ch-ua="Not A;Brand";v="99", "Chromium";v="90", "Google Chrome";v="90"
sec-ch-ua-mobile=?0
sec-fetch-dest=document
sec-fetch-mode=navigate
sec-fetch-site=none
sec-fetch-user=?1
upgrade-insecure-requests=1
user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36

BODY:

-no body in request-

Kubernetes 실습

- `kubectl create deployment balanced --image=k8s.gcr.io/echoserver:1.4`
- `kubectl expose deployment balanced --type=LoadBalancer --port=8080`
- `minikube tunnel`

```
명령 프롬프트 - minikube tunnel
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
* Stopping tunnel for service hello-minikube.

C:\Users\controlc>
C:\Users\controlc>
C:\Users\controlc>
C:\Users\controlc>
C:\Users\controlc>kubectl port-forward service/hello-minikube 7080:8080
Forwarding from 127.0.0.1:7080 -> 8080
Forwarding from [::1]:7080 -> 8080
Handling connection for 7080
Handling connection for 7080
Handling connection for 7080
Handling connection for 7080

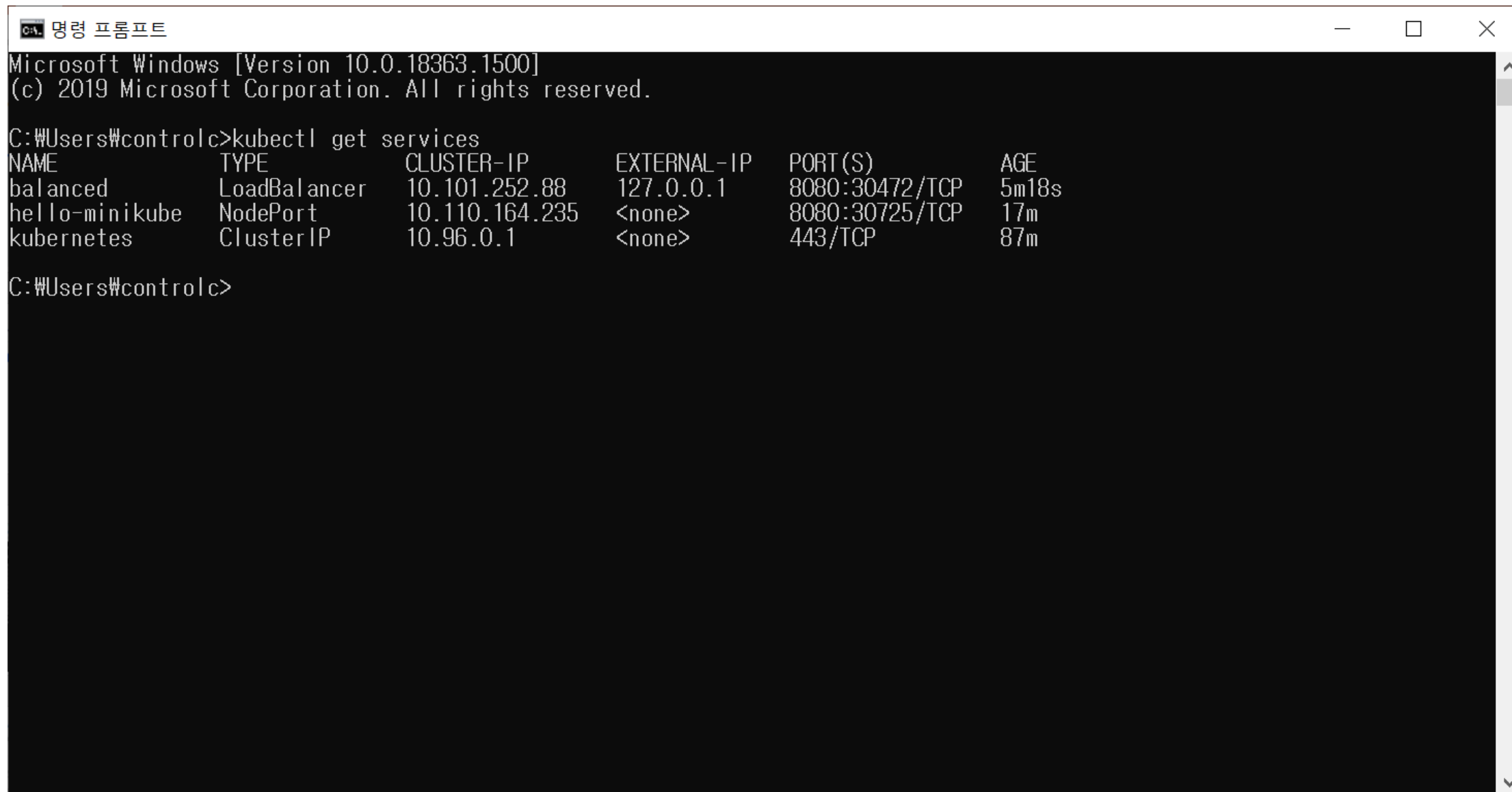
C:\Users\controlc>
C:\Users\controlc>
C:\Users\controlc>kubectl create deployment balanced --image=k8s.gcr.io/echoserver:1.4
deployment.apps/balanced created

C:\Users\controlc>kubectl expose deployment balanced --type=LoadBalancer --port=8080
service/balanced exposed

C:\Users\controlc>kubectl get services balanced
NAME      TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
balanced  LoadBalancer 10.101.252.88  <pending>      8080:30472/TCP   61s

C:\Users\controlc>minikube tunnel
* balanced 서비스의 터널을 시작하는 중
```

- kubectl get services



A screenshot of a Windows Command Prompt window titled "명령 프롬프트". The window shows the output of the command `kubectl get services`. The output is a table with columns: NAME, TYPE, CLUSTER-IP, EXTERNAL-IP, PORT(S), and AGE. The table lists three services: 'balanced' (LoadBalancer), 'hello-minikube' (NodePort), and 'kubernetes' (ClusterIP).

```
Microsoft Windows [Version 10.0.18363.1500]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\controlc>kubectl get services
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
balanced            LoadBalancer  10.101.252.88  127.0.0.1      8080:30472/TCP   5m18s
hello-minikube       NodePort       10.110.164.235 <none>         8080:30725/TCP   17m
kubernetes           ClusterIP      10.96.0.1     <none>         443/TCP          87m

C:\Users\controlc>
```

Kubernetes 실습

- Docker 에서 생성한 이미지로 kubernetes에서 컨테이너 앱을 실행시켜 보자

1. YAML 만들기 ➔ 2. Docker Hub 등록 ➔ 3. 배포(다운로드) ➔ 4. 앱 확인

127.0.0.1:62941

docker lab 1.1

x
0
0

0 squared is 0

- Rerun R
- Clear cache C
- Deploy this app
- Record a screenshot
- Documentation
- Ask a question
- Report a bug
- Streamlit for Teams
- Settings
- About

Kubernetes 실습

- Docker Hub에서 이미지를 가지고 와서 서비스를 구성해 보자
- C:\에 k8s_lab2 디렉토리 생성

명령창을 열고 mkdir k8s_lab2 실행

- YAML에 대한 이해
 - YAML은 쿠버네틱스의 오브젝트를 생성할 때 오브젝트에 대한 기본적인 정보와 의도하는 상태를 기술하는 용도
 - 쿠버네틱스는 API 통신 시 JSON 형식으로 정보를 교환
 - 대부분의 경우 .yaml 파일을 생성하여 kubectl에 제공하는 방식으로 사용

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: py-streamlitv1
spec:
  selector:
    matchLabels:
      run: py-streamlitv1
  template:
    metadata:
      labels:
        run: py-streamlitv1
    spec:
      containers:
        - name: py-streamlitv1
          image: noburi/my-repo:streamlitv1
```

- Docker Hub에서 이미지를 PULL하여 서비스를 구성해 보자
- C:\k8s_lab2 에 YAML 파일 만들기(파일명: my-python-streamlitv1.yml)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: py-streamlitv1
  labels:
    app: py-streamlitv1
spec:
  replicas: 1
  selector:
    matchLabels:
      run: py-streamlitv1
  template:
    metadata:
      labels:
        run: py-streamlitv1
    spec:
      containers:
        - name: py-streamlitv1
          image: noburi/my-repo:streamlitv1
          ports:
            - containerPort: 5000
              name: py-streamlitv1
      imagePullSecrets:
        - name: dockersecret
```

■ kubectl 사용법

kubectl [command] [TYPE] [NAME] [flag]

command: create, get, delete, apply 등

TYPE: 리소스 타입. pod, service, deployment

NAME: 리소스 이름

flag: 부가 옵션

예시)

Kubectl get pods # 파드 리스트 조회

Kubectl get services # 서비스 리스트 조회

Kubectl apply -f py-app1.yml # py-app1.yml에 정의된 내용 적용

Kubectl delete -f py-app1.yml # py-app1.yml에 정의된 내용 제거

Kubectl describe nodes my-node # my-node에 대한 상세 정보 출력

Kubectl delete pod zoo # zoo라는 파드 삭제

Kubernetes 실습

- Docker Hub에서 이미지를 가지고 와서 서비스를 구성해 보자
- 쿠버네틱스 secret 생성(아래 명령 실행 - #은 빼고)

```
# kubectl create secret docker-registry dockersecret --docker-username="[Docker Hub 계정]" \
  --docker-password="[Docker Hub 비밀번호]" --docker-server=https://index.docker.io/v1/
```

- 이제 Docker Hub에서 이미지를 가져와서 구동 시켜 보자

kubectl apply -f my-python-streamlitv1.yml 실행

 Windows PowerShell

```
PS C:\#k8s_lab2> kubectl apply -f .\my-python-streamlitv1.yml
deployment.apps/py-streamlitv1 created
```


Kubernetes 실습

- Docker Hub에서 이미지를 가지고 와서 서비스를 구성해 보자
- 잘 가지고 왔는지 확인

kubectl get all

```
Windows PowerShell
PS C:\k8s_lab2> kubectl get all
NAME                                     READY   STATUS    RESTARTS   AGE
pod/counter-7b76fd9bc6-k27mm            1/1     Running   0           34h
pod/my-python-app1-6b6bc86487-7lbkj     1/1     Running   0           112m
pod/py-streamlitv1-779f7b7745-mssfr     1/1     Running   0           10s
pod/redis-57d787df44-r7nvr             1/1     Running   0           34h

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
service/counter-lb                 LoadBalancer  10.100.164.98  10.32.4.31    30000:30083/TCP  33h
service/counter-np                 NodePort      10.99.81.79    <none>        3000:31000/TCP   34h
service/kubernetes                 ClusterIP     10.96.0.1      <none>        443/TCP          36h
service/redis                      ClusterIP     10.96.76.222   <none>        6379/TCP         34h

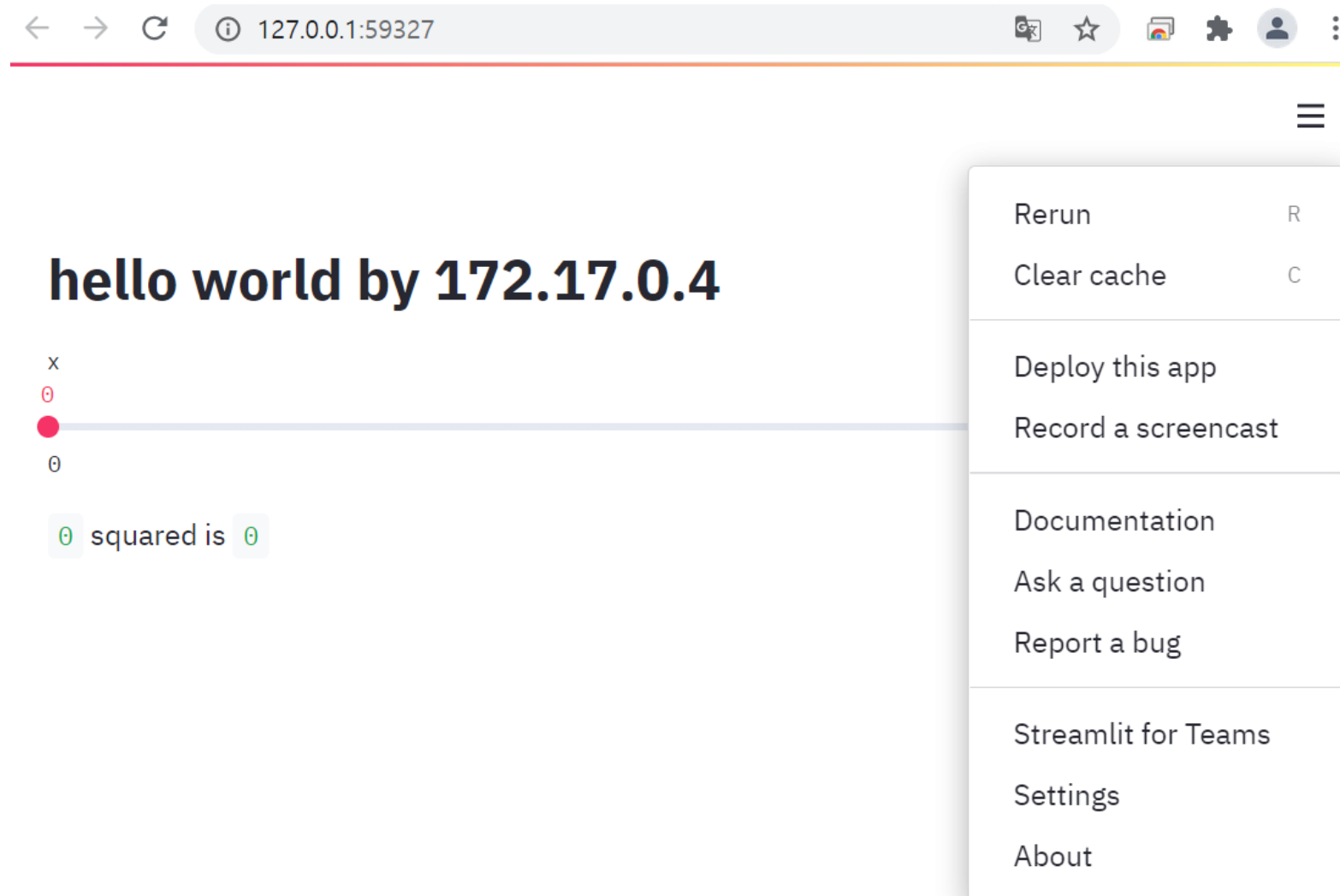
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/counter             1/1     1             1           34h
deployment.apps/my-python-app1      1/1     1             1           112m
deployment.apps/py-streamlitv1      1/1     1             1           10s
deployment.apps/redis               1/1     1             1           34h

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/counter-7b76fd9bc6  1         1         1       34h
replicaset.apps/my-python-app1-6b6bc86487  1         1         1       112m
replicaset.apps/py-streamlitv1-779f7b7745  1         1         1       10s
replicaset.apps/redis-57d787df44  1         1         1       34h
PS C:\k8s_lab2>
```

- 작동하는지 확인해 보자

```
# kubectl expose deployment py-streamlitv1
```

```
# minikube service py-streamlitv1
```



- ReplicaSet 구성 실습 추가
 - my-python-streamlitv1.yml 파일의 replicas 값 수정

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: py-streamlitv1
  labels:
    app: py-streamlitv1
spec:
  replicas: 2
  selector:
    matchLabels:
      run: py-streamlitv1
  template:
    metadata:
      labels:
        run: py-streamlitv1
    spec:
      containers:
        - name: py-streamlitv1
          image: noburi/my-repo:streamlitv1
          ports:
            - containerPort: 5000
              name: py-streamlitv1
      imagePullSecrets:
        - name: dockersecret
```

Kubernetes 실습

- ReplicaSet 구성 실습 추가

kubectl apply -f my-python-streamlitv1.yml 실행

 Windows PowerShell

```
PS C:\#k8s_lab2> kubectl apply -f .\my-python-streamlitv1.yml
deployment.apps/py-streamlitv1 created
```

- pod와 replica 확인

kubectl get pods

kubectl get rs

 Windows PowerShell

```
PS C:\Wk8s_lab2> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
py-streamlitv1-7db5748874-dkdkw	1/1	Running	0	3m29s
py-streamlitv2-84f97d99bd-jdzns	1/1	Running	1	4d18h
py-streamlitv2-84f97d99bd-rrcv2	1/1	Running	1	4d18h

```
PS C:\Wk8s_lab2> kubectl get rs
```

NAME	DESIRED	CURRENT	READY	AGE
py-streamlitv1-7db5748874	1	1	1	3m37s
py-streamlitv2-84f97d99bd	2	2	2	4d18h

```
PS C:\Wk8s_lab2> _
```