

그로킹 강화학습 실습을 위한 환경 구축(윈도우)

동의대학교 강화학습 수업

윤주상 교수님

NVIDIA 앱 설치

- [게이머와 크리에이터를 위한 NVIDIA 앱 다운로드 | NVIDIA](#)

NVIDIA 앱

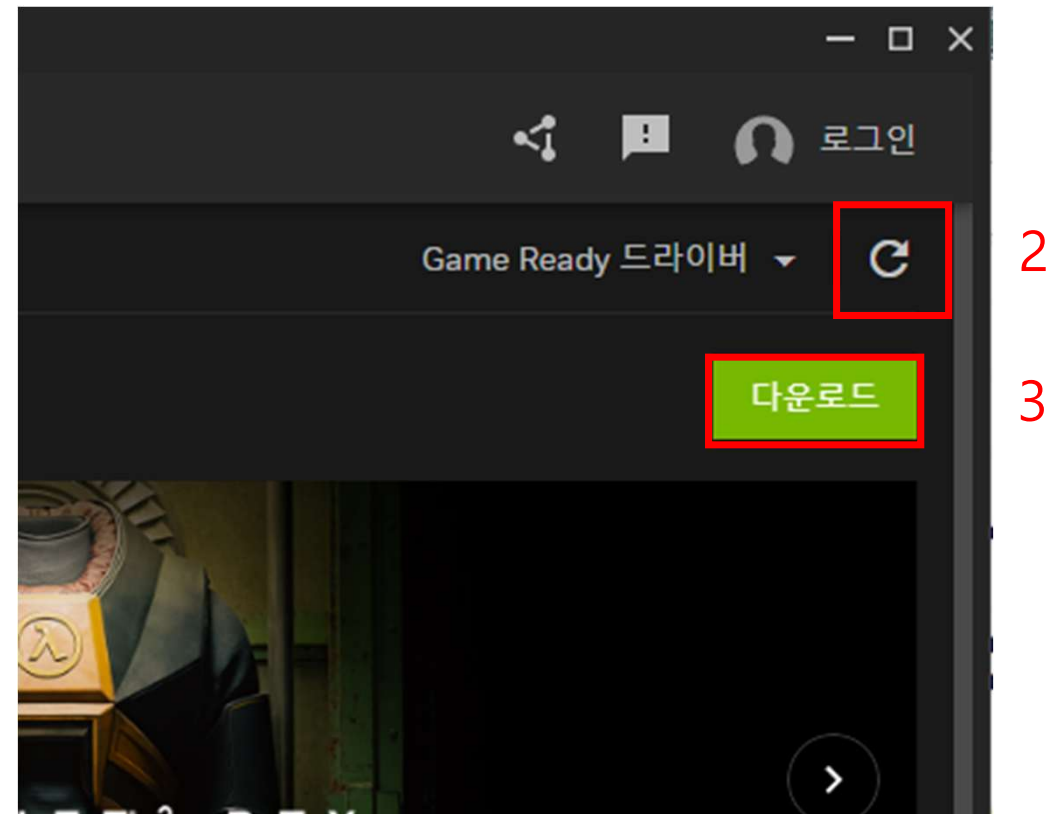
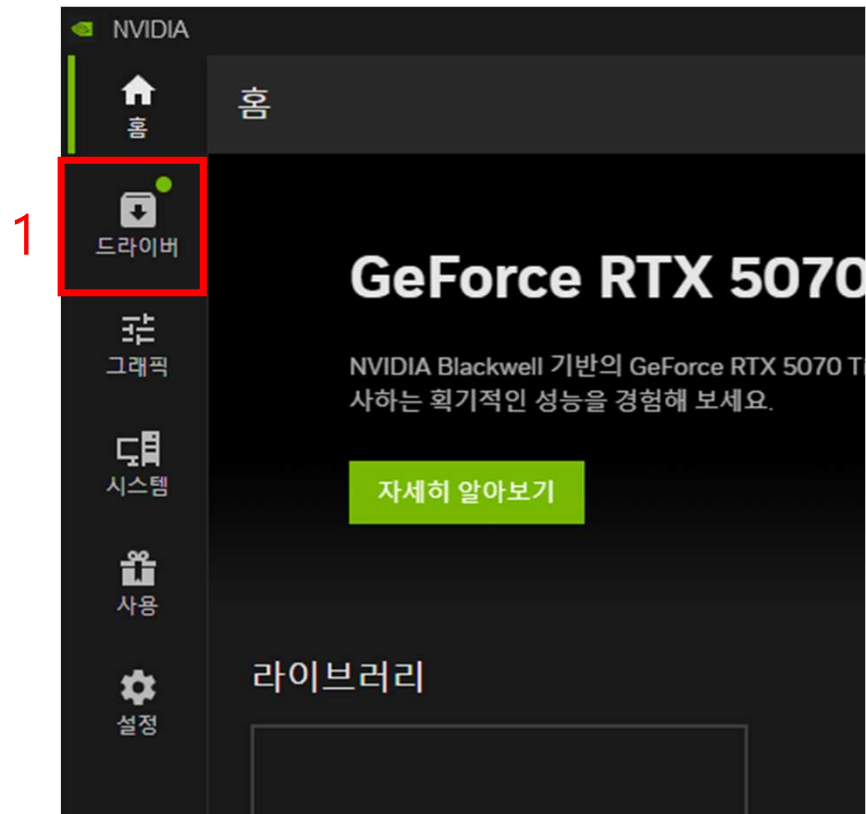
NVIDIA 앱은 PC 게이머와 크리에이터를 위한 필수 애플리케이션입니다. 최신 NVIDIA 드라이버 및 기술로 PC를 최신 상태로 유지하세요. 새로운 통합 GPU 제어 센터로 게임 및 애플리케이션을 최적화하고, 인게임 오버레이를 통해 강력한 녹화 도구로 좋아하는 순간을 캡처하고, 최신 NVIDIA 도구 및 소프트웨어를 살펴보세요.

지금 다운로드 받기

자세히 알아보기 >

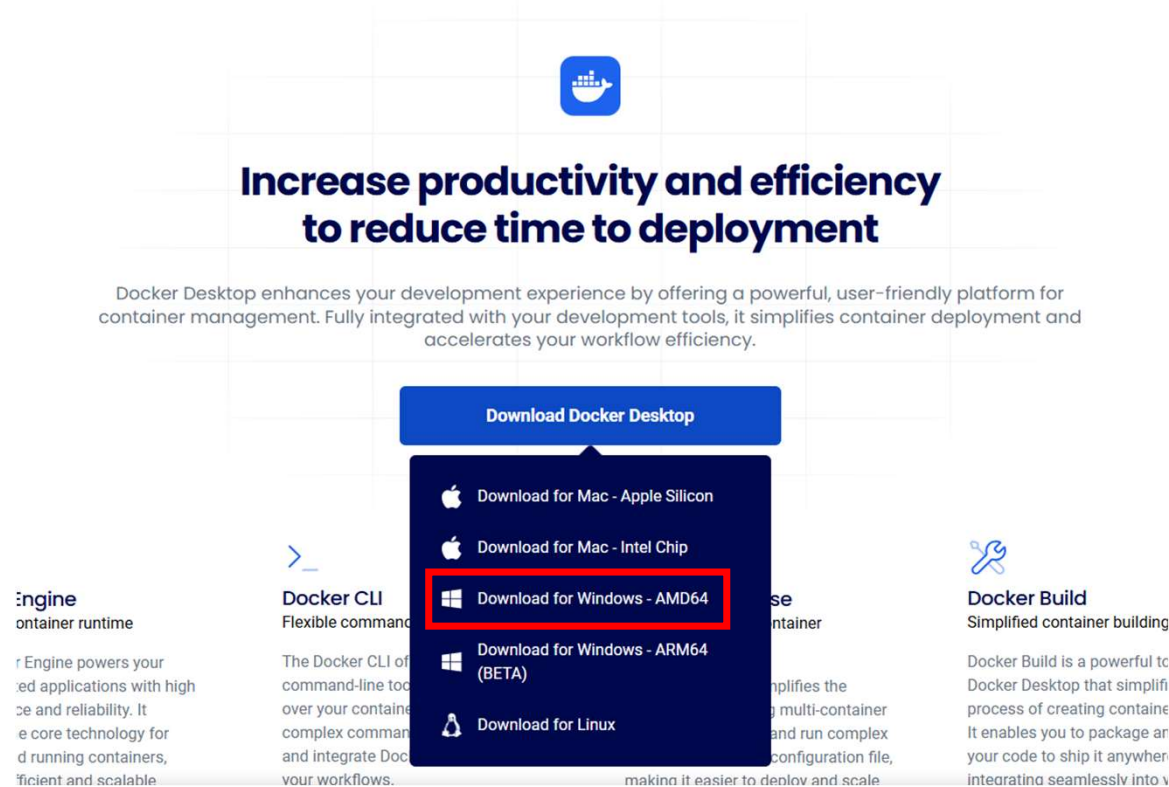
최신 GPU 드라이버 설치

- 드라이버 탭으로 이동, 최신드라이버 확인후 다운로드



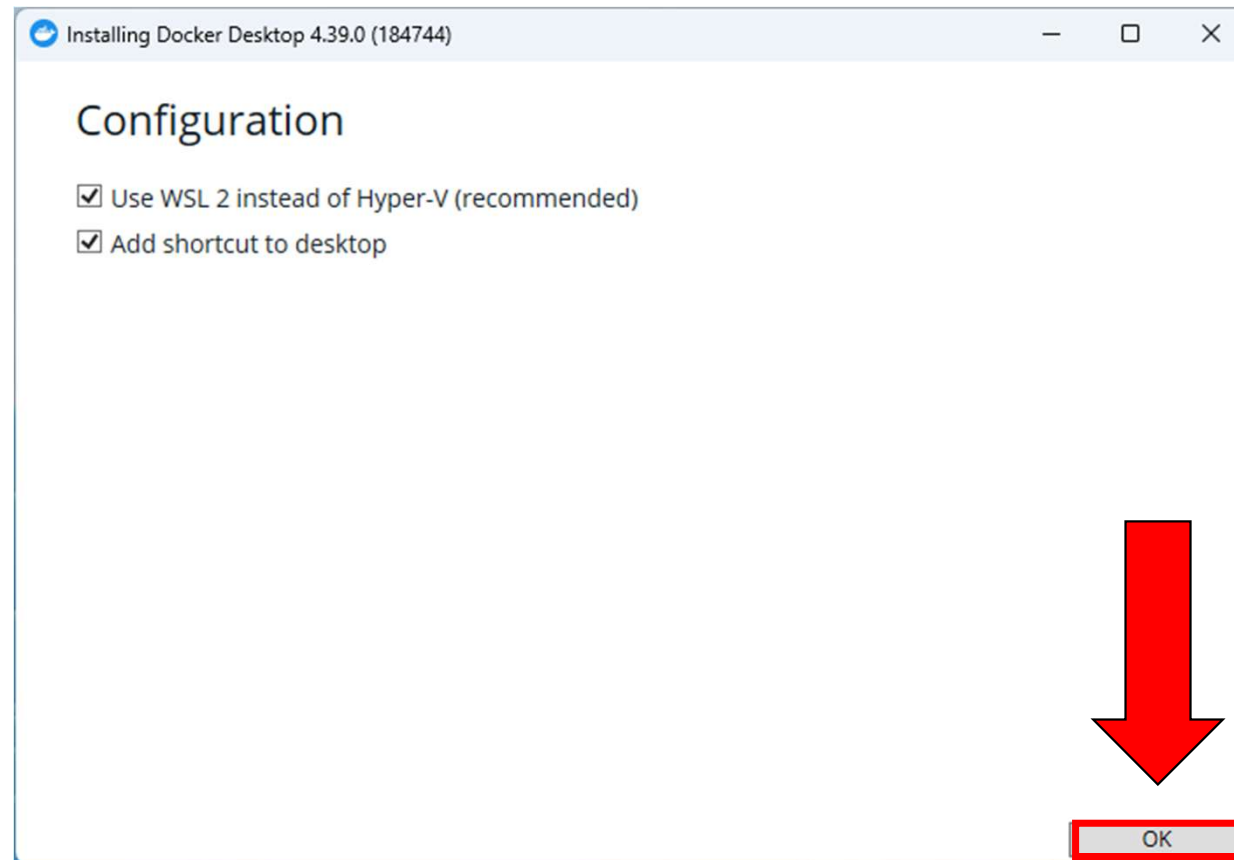
Docker Desktop 설치

- [Docker Desktop: The #1 Containerization Tool for Developers](#)
[Docker](#)



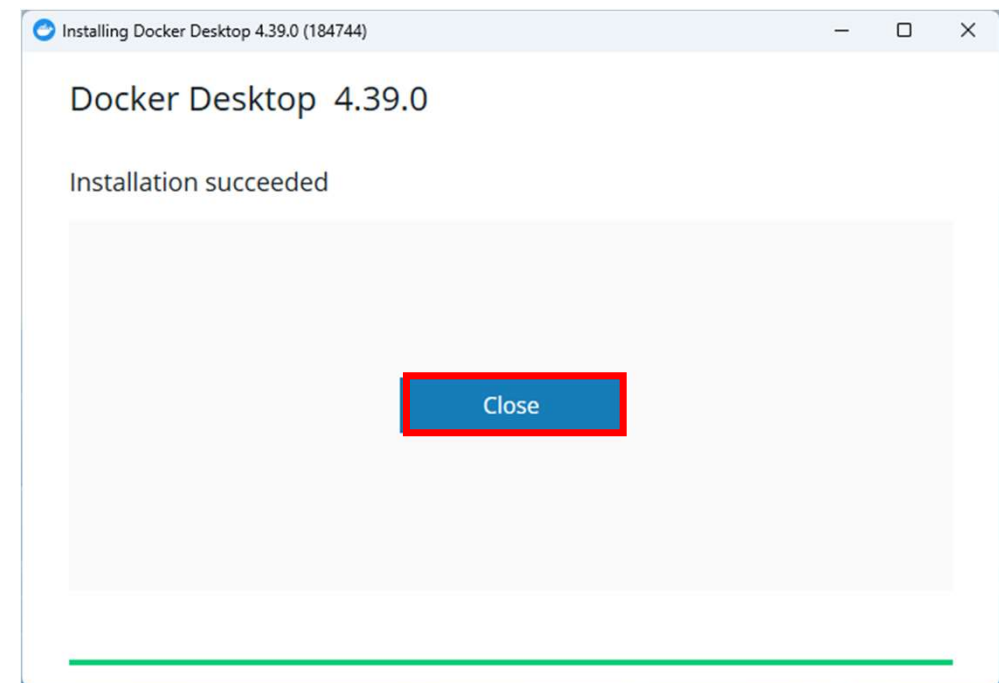
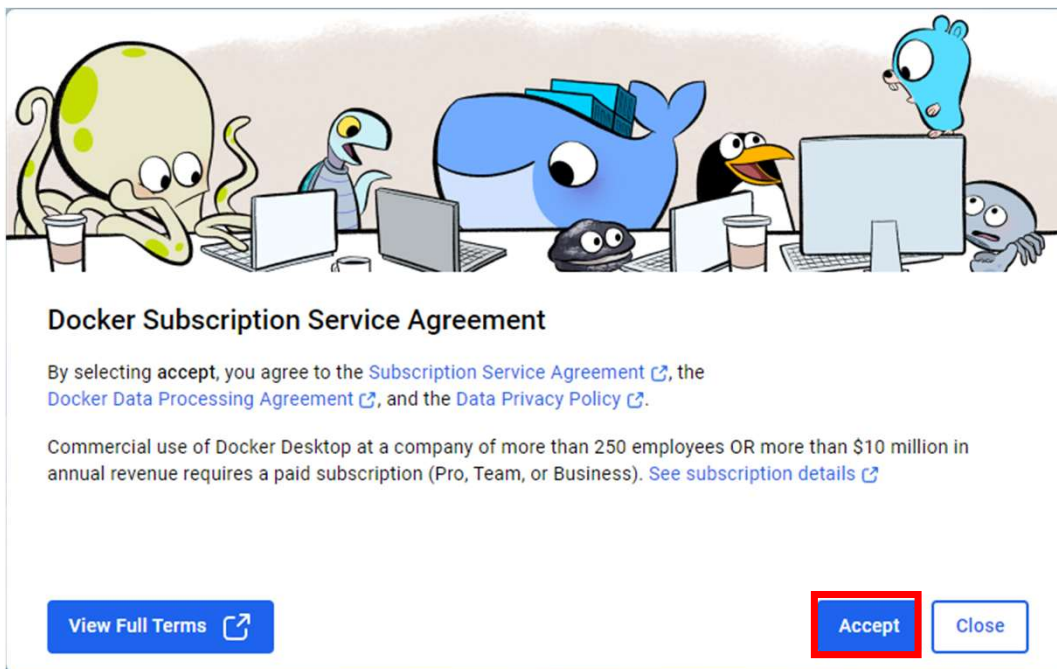
Docker Desktop 설치

- OK 하기



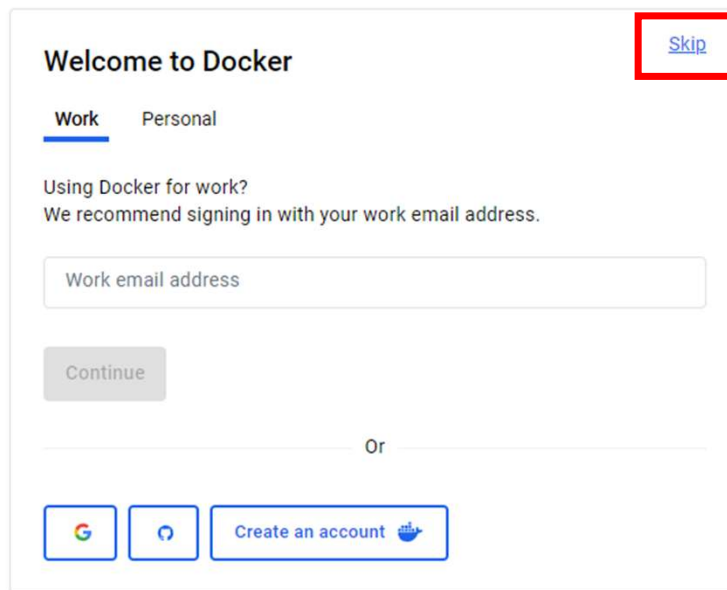
Docker Desktop 설치

- Accept 하기, 설치 완료되면 Close 하기



Docker Desktop 설치 후 설정

- Skip 두번 하기



Welcome to Docker




Work Personal

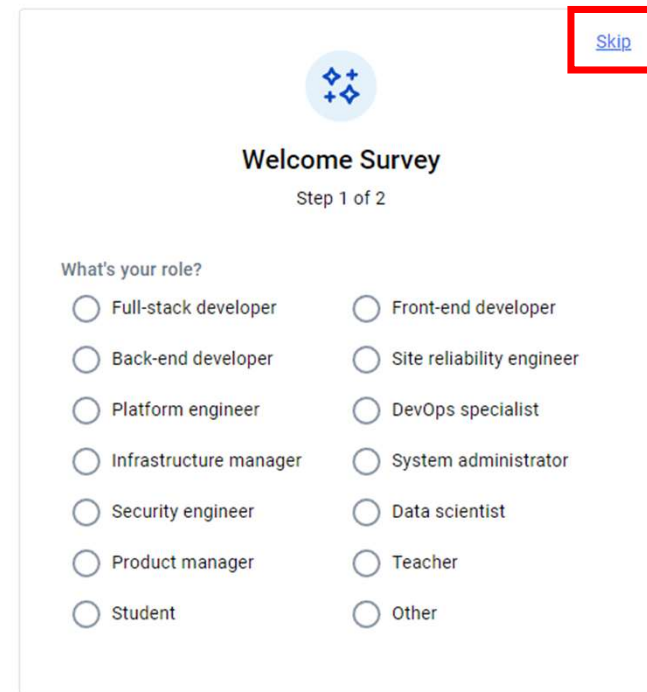
Using Docker for work?
We recommend signing in with your work email address.


Work email address

Continue

Or

  [Create an account](#) 





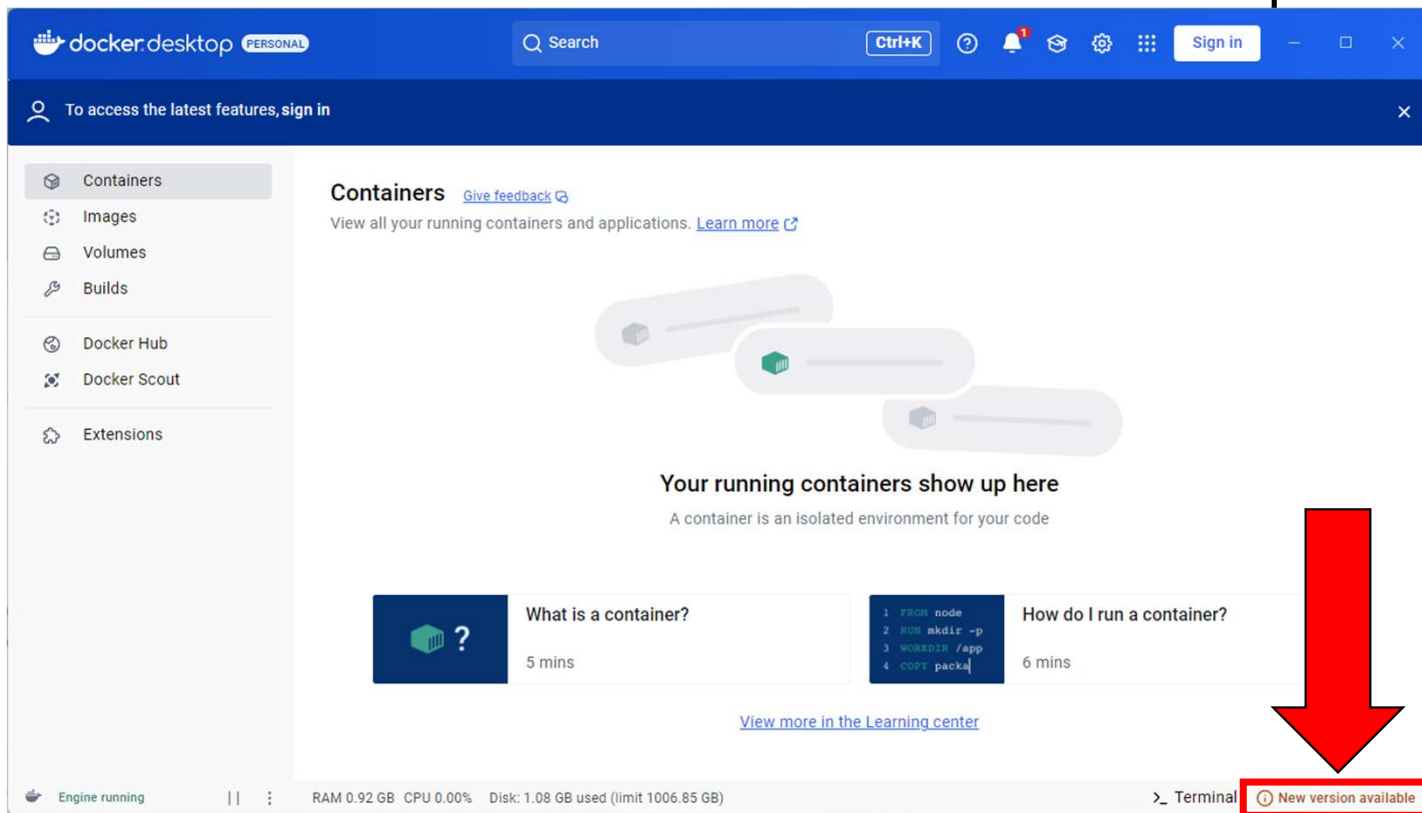
Welcome Survey
Step 1 of 2

What's your role?

<input type="radio"/> Full-stack developer	<input type="radio"/> Front-end developer
<input type="radio"/> Back-end developer	<input type="radio"/> Site reliability engineer
<input type="radio"/> Platform engineer	<input type="radio"/> DevOps specialist
<input type="radio"/> Infrastructure manager	<input type="radio"/> System administrator
<input type="radio"/> Security engineer	<input type="radio"/> Data scientist
<input type="radio"/> Product manager	<input type="radio"/> Teacher
<input type="radio"/> Student	<input type="radio"/> Other

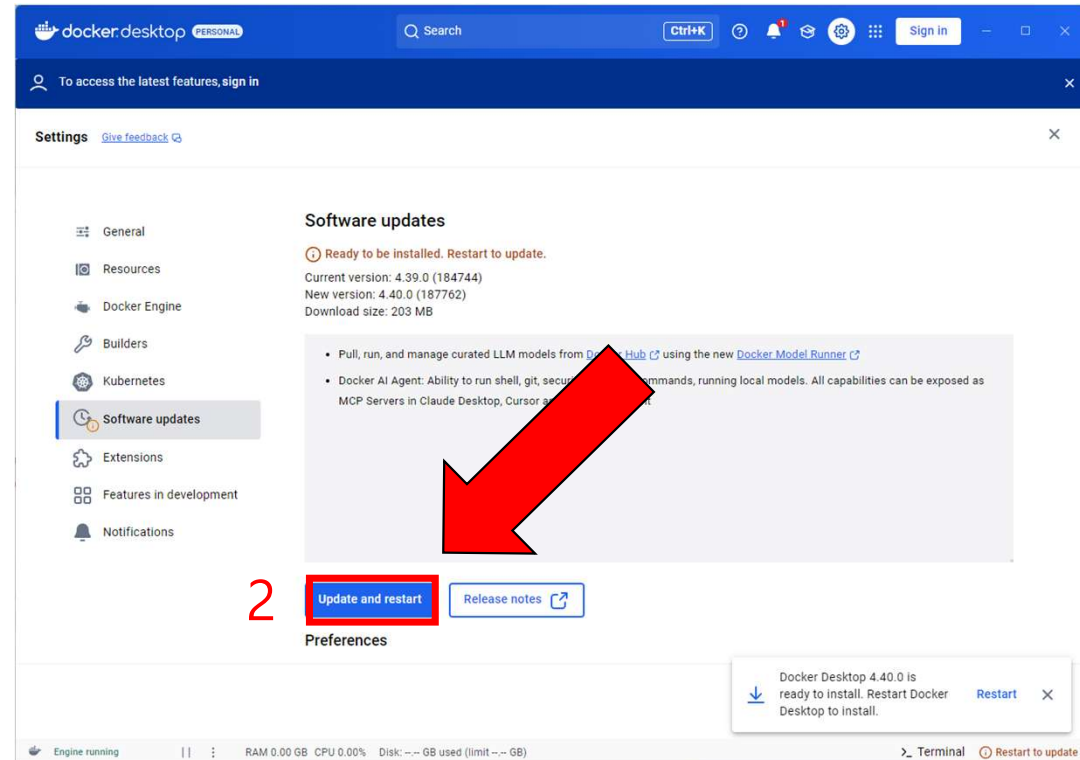
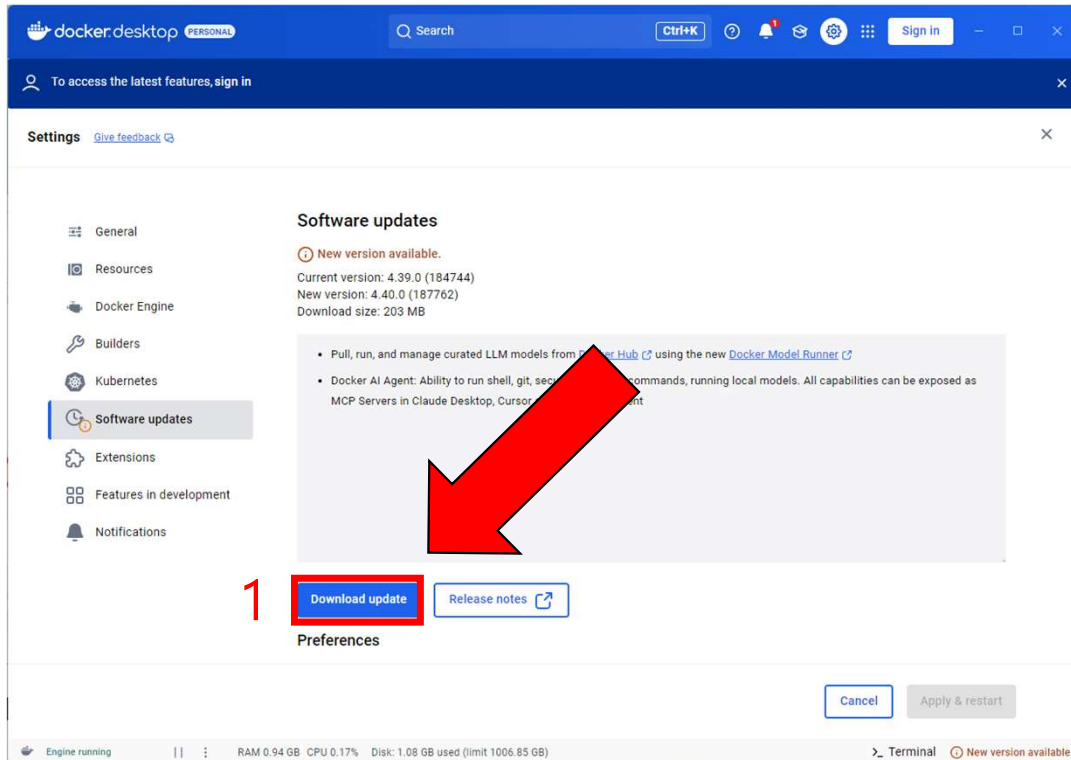
Docker Desktop 업데이트하기

- 우측 하단의 New version available로 Software updates 들어가
기



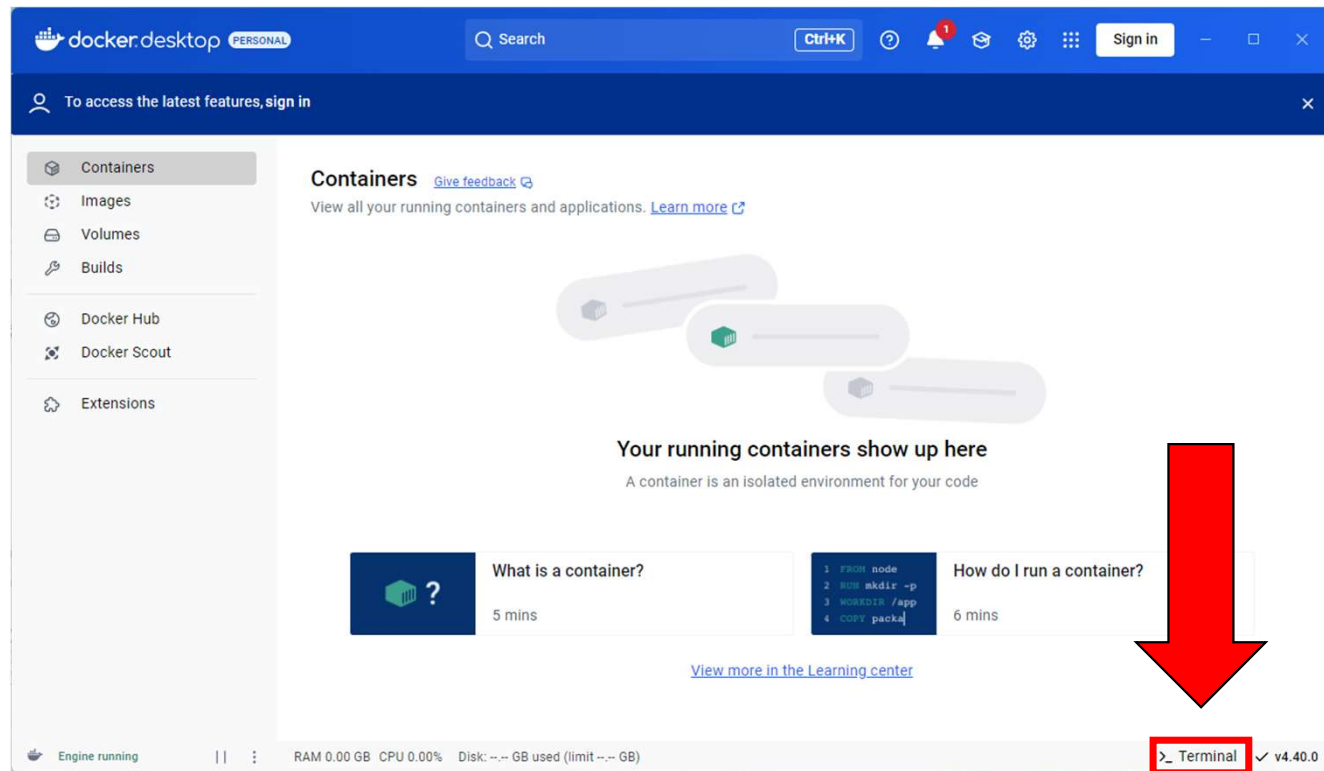
Docker Desktop 업데이트하기

- Download update 하고 Update and restart 하기



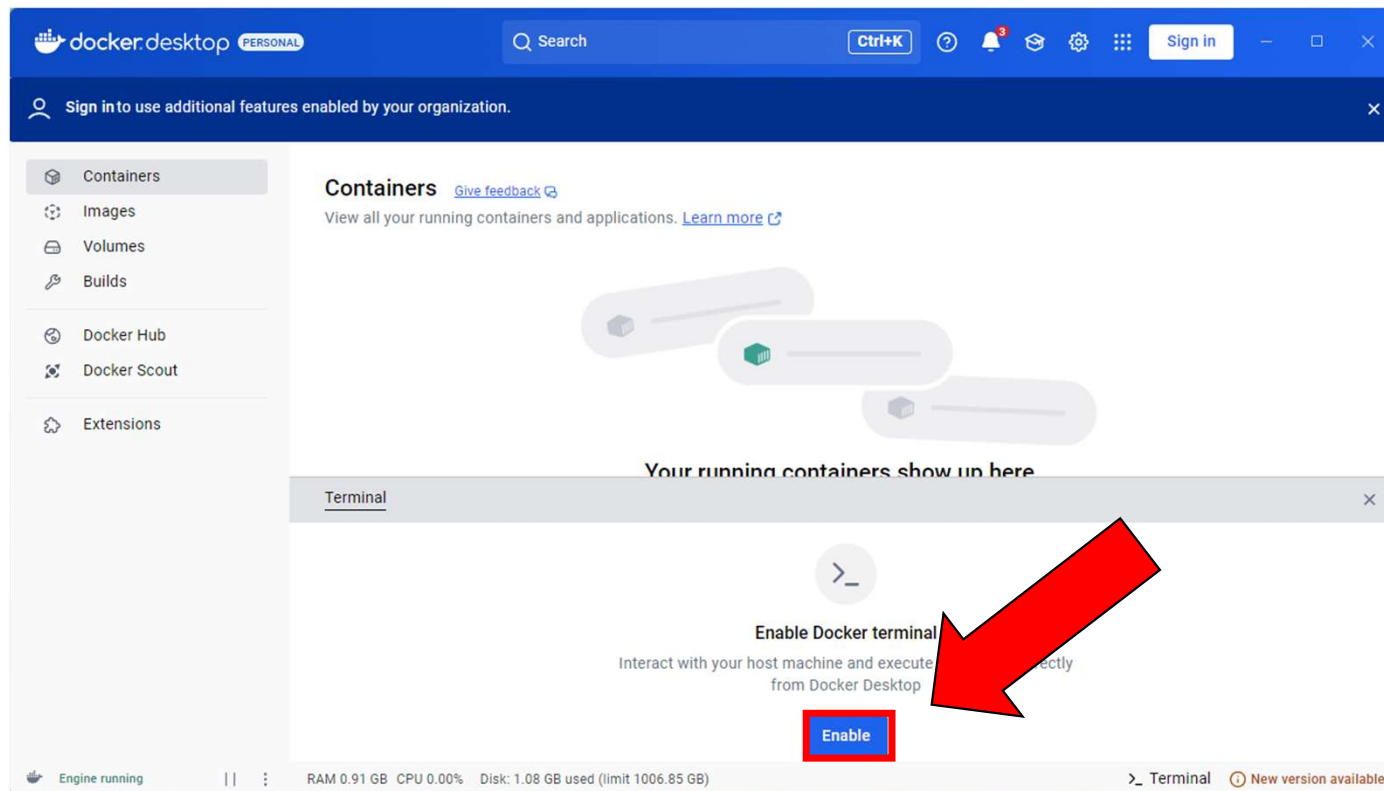
Docker Desktop 터미널 실행

- 우측 하단 터미널 실행하기



Docker Desktop 터미널 실행

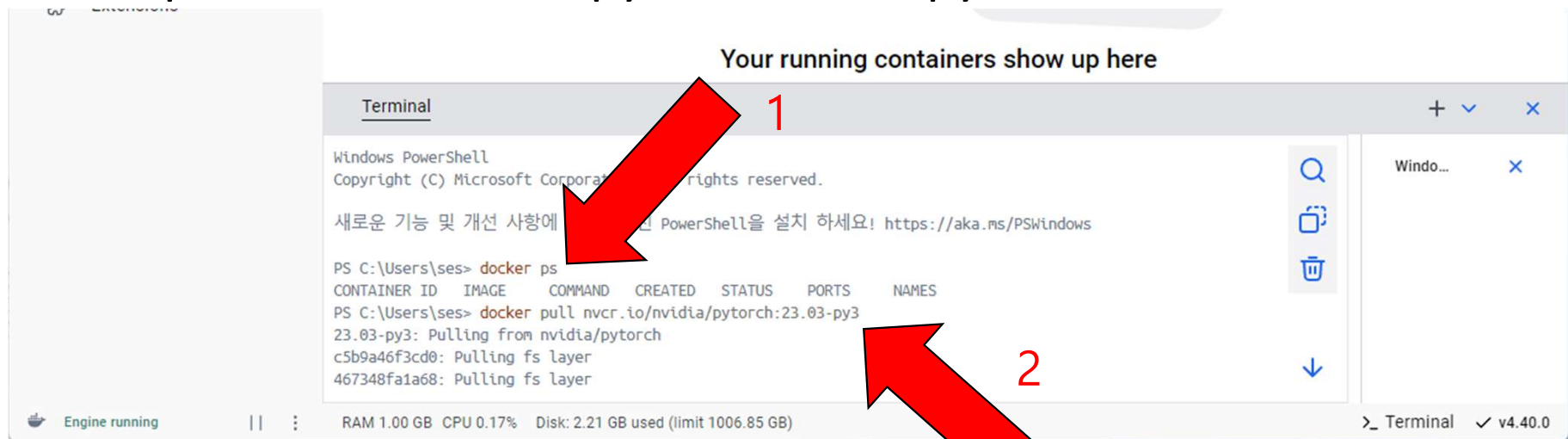
- 터미널 Enable 하기



컨테이너 실행에 필요한 이미지 Pull

- [PyTorch | NVIDIA NGC](#)

- 1 • docker ps
 - 현재 실행중인 컨테이너 확인
- 2 • docker pull nvcr.io/nvidia/pytorch:23.03-py3



Pull 받은 이미지로 컨테이너 실행

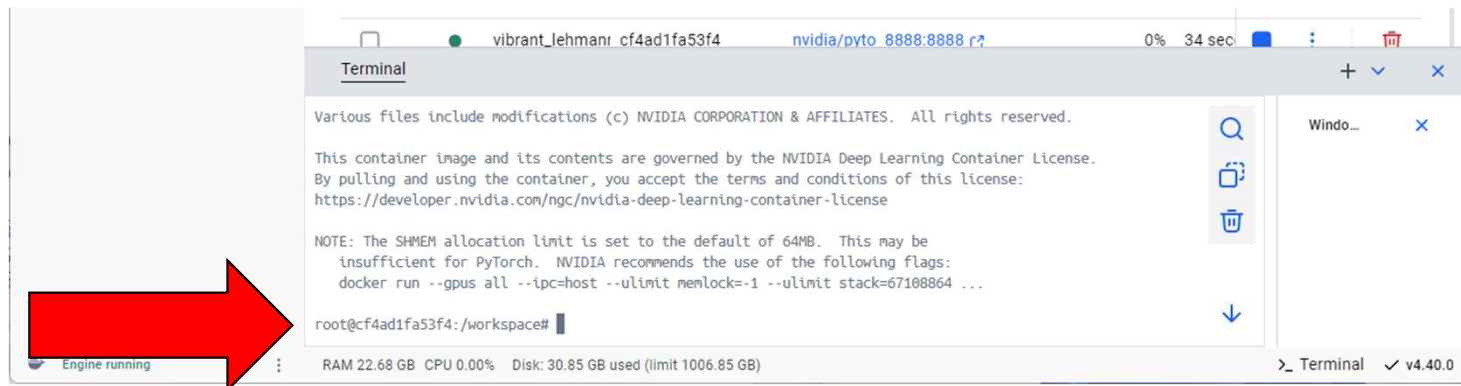
- `docker run --gpus all -it -p 8888:8888 nvcr.io/nvidia/pytorch:23.03-py3`

실행전



```
Terminal
8d0b4e634534: Pull complete
96073cba6da3: Pull complete
36b952fae64f: Pull complete
b659afa6f4a5: Pull complete
010efb4a2b3b: Pull complete
1893b47c020c: Pull complete
33b7a2cbbcf6: Pull complete
Digest: sha256:6ffffaca1c540d9871f97ac2459268bef566f7f73768768c47cacfaee810abe67
Status: Downloaded newer image for nvcr.io/nvidia/pytorch:23.03-py3
nvcr.io/nvidia/pytorch:23.03-py3
PS C:\Users\ses> docker run --gpus all -it -p 8888:8888 nvcr.io/nvidia/pytorch:23.03-py3
```

실행후



```
Terminal
Various files include modifications (c) NVIDIA CORPORATION & AFFILIATES. All rights reserved.

This container image and its contents are governed by the NVIDIA Deep Learning Container License.
By pulling and using the container, you accept the terms and conditions of this license:
https://developer.nvidia.com/ngc/nvidia-deep-learning-container-license

NOTE: The SHMEM allocation limit is set to the default of 64MB. This may be
insufficient for PyTorch. NVIDIA recommends the use of the following flags:
docker run --gpus all --ipc=host --ulimit memlock=-1 --ulimit stack=67108864 ...

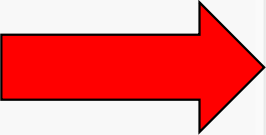
root@cf4ad1fa53f4:/workspace#
```

컨테이너 실행 명령어 설명

- `docker run --gpus all -it -p 8888:8888 nvcr.io/nvidia/pytorch:23.03-py3`
- `--gpus all`
 - gpu를 사용한다는 옵션
- `-p 8888:8888`
 - 컨테이너 내부의 8888포트와 호스트의 8888포트를 연결
- `-it`
 - 컨테이너 실행 후 종료하지 않고 터미널의 입력을 컨테이너 내부에 직접 입력
- `docker run nvcr.io/nvidia/pytorch:23.03-py3`
 - `nvcr.io/nvidia/pytorch:23.03-py3` 이미지를 사용해서 컨테이너 실행
 - 이미지를 pull 하지 않고 바로 run을 해도 자동으로 이미지를 pull함

Github에서 코드 clone

- git clone <https://github.com/mimoralea/gdrl.git>
 - 설명 : <https://github.com/mimoralea/gdrl.git> 저장소의 코드들을 로컬로 그대로 복제



```
docker run --gpus all --ipc=host --ulimit memlock=-1 --ulimit stack=67108864 ...  
root@cf4ad1fa53f4:/workspace# git clone https://github.com/mimoralea/gdrl.git  
Cloning into 'gdrl'...  
remote: Enumerating objects: 3653, done.  
remote: Counting objects: 100% (18/18), done.  
remote: Compressing objects: 100% (11/11), done.  
remote: Total 3653 (delta 7), reused 7 (delta 7), pack-reused 3635 (from 2)  
Receiving objects: 100% (3653/3653), 669.36 MiB | 17.58 MiB/s, done.  
Resolving deltas: 100% (1308/1308), done.  
root@cf4ad1fa53f4:/workspace#
```

Engine running || : RAM 22.62 GB CPU 0.17% Disk: 31.63 GB used (limit 1006.85 GB) >_ Terminal v4.40.0

Clone 되었는지 확인 및 경로 이동

1 • ls

- 현재 directory의 폴더 및 파일 목록 출력

3 • cd gdrl/notebooks/

- cd(change directory) gdrl폴더의 notebooks 폴더(앞의 두 글자 정도 치고 Tab하면 경로 자동완성)

The screenshot shows a Docker container interface with a terminal window. The terminal output shows the cloning of a repository into a folder named 'gdrl'. The container's file explorer shows the contents of the 'gdrl' directory, which includes 'samples', 'gdrl', and 'tutorials' folders. Red arrows and numbers 1, 2, and 3 are overlaid on the image to indicate the steps described in the text.

Name	Container ID	Image	Port(s)	CPU (%)	Last st	Actions
vibrant_lehmanr	cf4ad1fa53f4	nvidia/pyto	8888:8888 r?	0%	6 minu	

```
Terminal
Cloning into 'gdrl'...
remote: Enumerating objects: 3653, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 3653 (delta 7), reused 7 (delta 7), pack-reused 3635 (from 2)
Receiving objects: 100% (3653/3653), 669.36 MiB | 17.58 MiB/s, done.
Resolving deltas: 100% (1308/1308), done.
root@cf4ad1fa53f4:/workspace# ls
NVIDIA_Deep_Learning_Container_License.pdf  README.md  samples  gdrl  tutorials
root@cf4ad1fa53f4:/workspace# cd gdrl/notebooks/
root@cf4ad1fa53f4:/workspace/gdrl/notebooks#
```

1 → (points to the 'ls' command in the terminal)

2 → (points to the 'gdrl' folder in the file explorer)

3 → (points to the 'cd gdrl/notebooks/' command in the terminal)

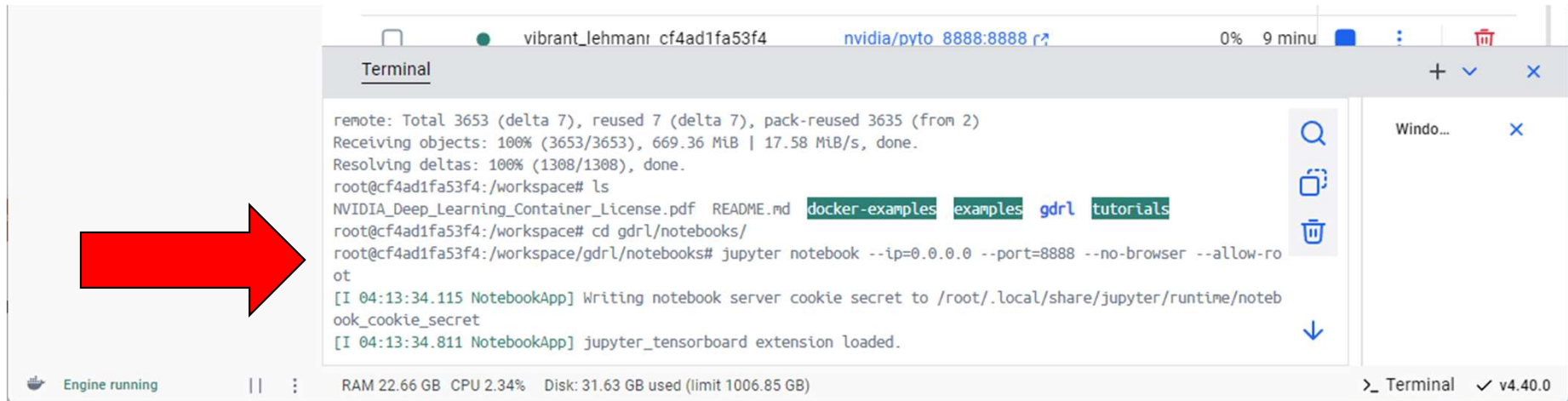
2 gdrl 폴더가 존재하는지 확인

3

Engine running || RAM 22.64 GB CPU 0.17% Disk: 31.63 GB used (limit 1006.85 GB) >_ Terminal ✓ v4.40.0

컨테이너 내부에서 주피터 노트북 실행

- `jupyter notebook --ip=0.0.0.0 --port=8888 --no-browser --allow-root`



The screenshot shows a Docker container terminal with the following output:

```
remote: Total 3653 (delta 7), reused 7 (delta 7), pack-reused 3635 (from 2)
Receiving objects: 100% (3653/3653), 669.36 MiB | 17.58 MiB/s, done.
Resolving deltas: 100% (1308/1308), done.
root@cf4ad1fa53f4:/workspace# ls
NVIDIA_Deep_Learning_Container_License.pdf  README.md  docker-examples  examples  gdrl  tutorials
root@cf4ad1fa53f4:/workspace# cd gdrl/notebooks/
root@cf4ad1fa53f4:/workspace/gdrl/notebooks# jupyter notebook --ip=0.0.0.0 --port=8888 --no-browser --allow-root
[I 04:13:34.115 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret
[I 04:13:34.811 NotebookApp] jupyter_tensorboard extension loaded.
```

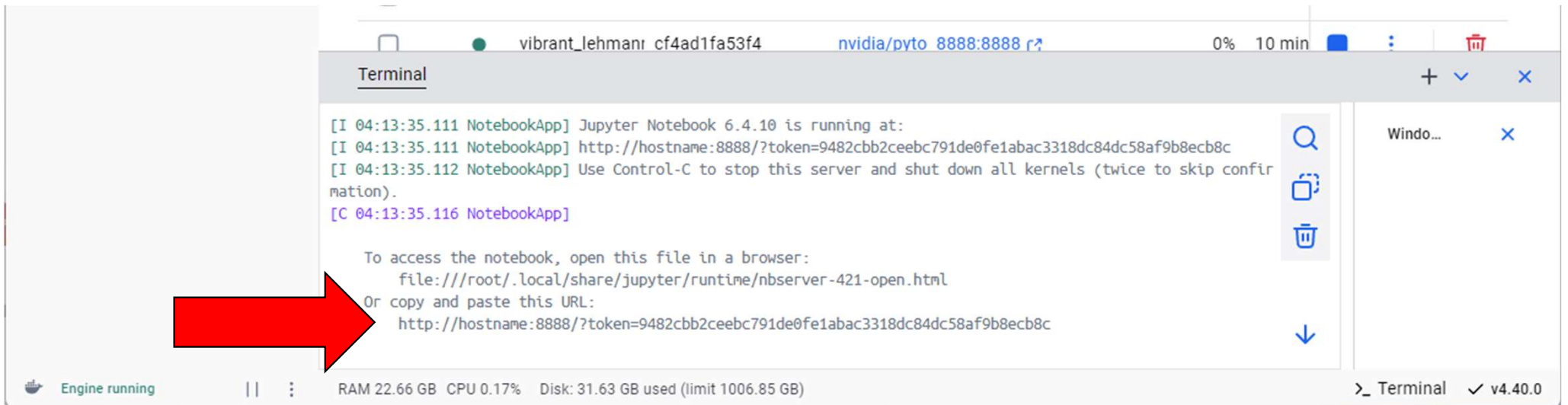
The terminal window is titled "Terminal" and shows the container ID "vibrant_lehmanr cf4ad1fa53f4" and the command "nvidia/pyto 8888:8888 r?". The status bar at the bottom indicates "Engine running", "RAM 22.66 GB", "CPU 2.34%", "Disk: 31.63 GB used (limit 1006.85 GB)", and "v4.40.0". A red arrow points to the terminal window.

주피터 노트북 실행 명령어 설명

- `jupyter notebook --ip=0.0.0.0 --port=8888 --no-browser --allow-root`
- `--ip=0.0.0.0`
 - 외부네트워크에서도 서버에 접근하도록 하는 옵션
 - 안하면 localhost만 가능
- `--port=8888`
 - jupyter notebook 서버가 사용할 포트(8888번) 지정
- `--no-browser`
 - jupyter notebook 실행시 브라우저를 자동으로 열지 않도록 설정
 - 원격 혹은 컨테이너로 실행할때 사용
- `--allow-root`
 - 루트 사용자 권한으로 jupyter notebook을 실행하도록 허용
 - 컨테이너 환경에서 필요

주피터 노트북 접속하기

- 제공되는 url로 주피터 노트북 접속하기



The screenshot shows a terminal window titled "Terminal" with the following output:

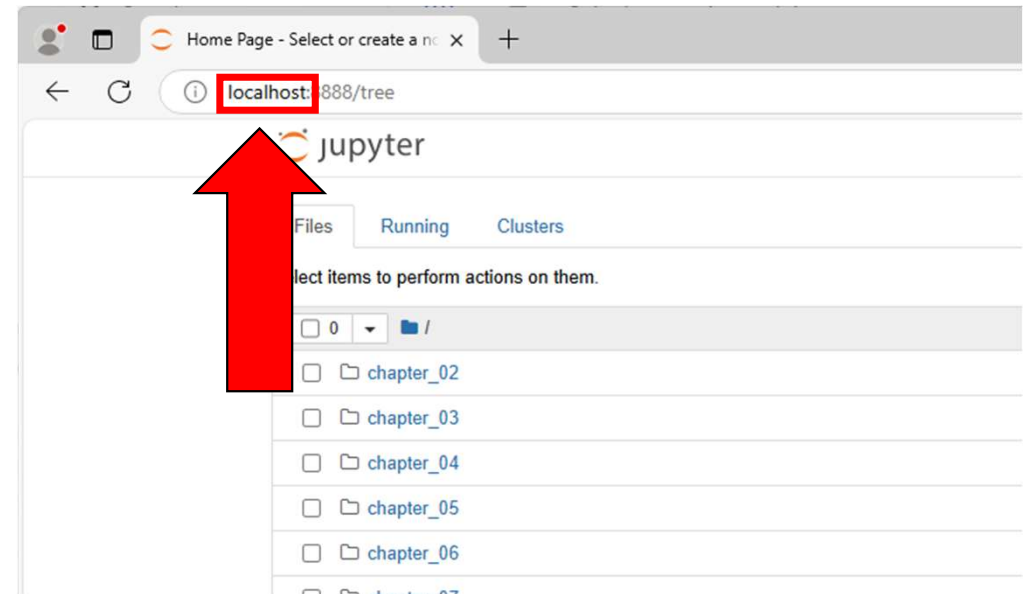
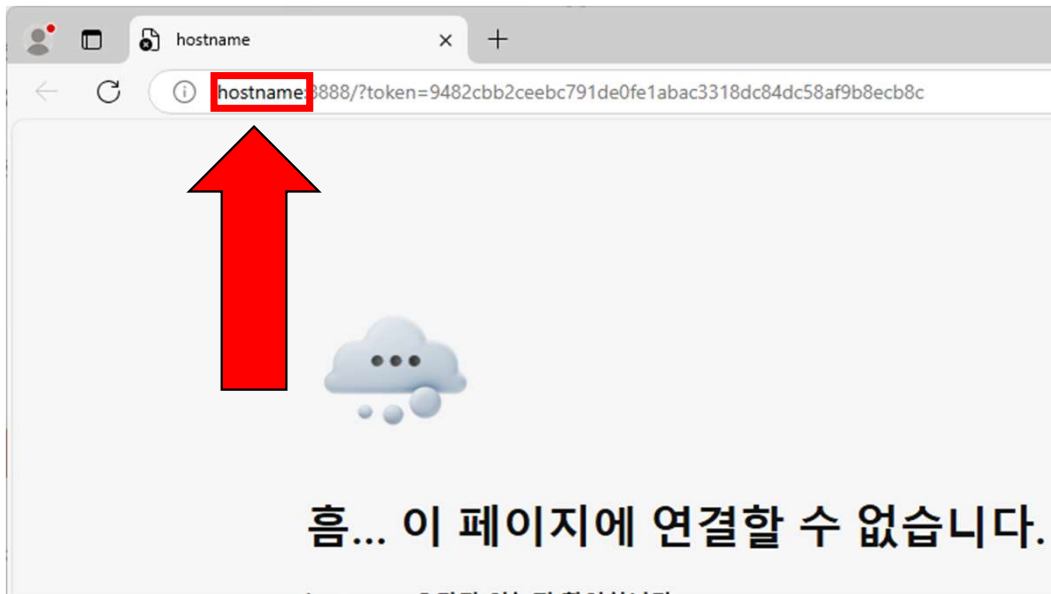
```
[I 04:13:35.111 NotebookApp] Jupyter Notebook 6.4.10 is running at:  
[I 04:13:35.111 NotebookApp] http://hostname:8888/?token=9482cbb2ceebc791de0fe1abac3318dc84dc58af9b8ecb8c  
[I 04:13:35.112 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confir  
mation).  
[C 04:13:35.116 NotebookApp]  
  
To access the notebook, open this file in a browser:  
file:///root/.local/share/jupyter/runtime/nbserver-421-open.html  
Or copy and paste this URL:  
http://hostname:8888/?token=9482cbb2ceebc791de0fe1abac3318dc84dc58af9b8ecb8c
```

A red arrow points to the URL: `http://hostname:8888/?token=9482cbb2ceebc791de0fe1abac3318dc84dc58af9b8ecb8c`.

The terminal window also shows the status bar at the bottom: "Engine running", "RAM 22.66 GB", "CPU 0.17%", "Disk: 31.63 GB used (limit 1006.85 GB)", and "Terminal v4.40.0".

주피터 노트북 접속하기

- 접속 후 url 앞쪽의 hostname을 localhost로 바꾸기



컨테이너 종료 하기

- 1 • 실행중인 주피터 노트북 Ctrl+C로 종료하기
- 2 • 주피터 노트북이 종료되면 exit 입력 후 엔터

```
1 active kernel
Jupyter Notebook 6.4.10 is running at:
http://hostname:8888/?token=9482cbb2ceeabc791de0fe1abac3318dc84dc58af9b8ecb8c
Shutdown this notebook server (y/[n])? y
[C 04:41:49.758 NotebookApp] Shutdown confirmed
[I 04:41:49.760 NotebookApp] Shutting down 1 kernel
[I 04:41:49.761 NotebookApp] Kernel shutdown: f35f4be8-48b3-418d-a71b-7be0e1baa618
[I 04:41:49.976 NotebookApp] Shutting down 0 terminals
root@cf4ad1fa53f4:/workspace/gdrl/notebooks# exit
exit
PS C:\Users\ses>
```

1 종료되었는지 확인

컨테이너 다시 실행하기

- 1 • `docker ps -a`
 - 실행중이거나 정지된 모든 컨테이너 목록 출력
- 2 • 정지된 컨테이너 id 확인하기 (CONTAINER ID)
- 3 • `docker start <CONTAINER ID>`
- 4 • `docker exec -it <CONTAINER ID> /bin/bash`



컨테이너 다시 실행하는 명령어 설명

- `docker start <CONTAINER ID>`
 - `<CONTAINER ID>`에 해당하는 컨테이너 시작
- `docker exec -it <CONTAINER ID> /bin/bash`
 - `exec` : `<CONTAINER ID>`에 해당하는 컨테이너내에서 명령을 실행하기 위한 Docker 명령어
 - `-it` : 컨테이너의 표준입력과 가상 터미널을 활성화 → 사용자가 컨테이너에 직접 입력 할수있도록함
 - `/bin/bash` : 컨테이너 내부의 Bash 셸을 실행

주피터 노트북 다시 접속하기

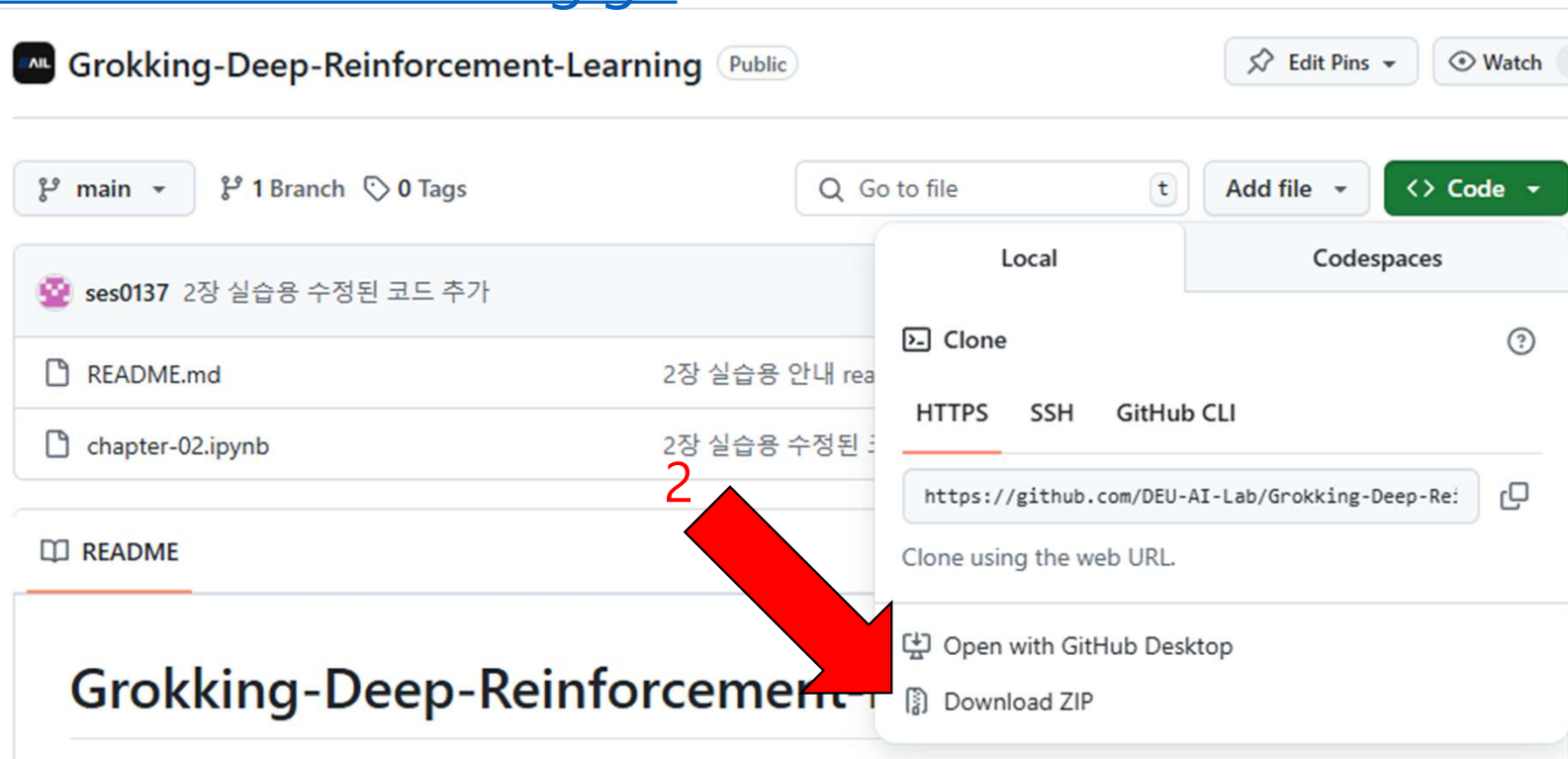
- 컨테이너 종료 상태인지 확인하고
 - 이 파워포인트의 22장 부터 "컨테이너 다시 실행하기 " 을 따라한후
 - 이 파워포인트의 16장 부터 "Clone 되었는지 확인 및 경로이동" 따라하기

책과 다른점

- gym 라이브러리가 gymnasium으로 바뀌어서 설치 해야 하는 라이브러리도 바뀌고 코드도 일부 수정됨
- 해당 라이브러리와 코드는 github를 통해서 제공될 예정

2장 실습용 코드 다운받기

- <https://github.com/DEU-AI-Lab/Grokking-Deep-Reinforcement-Learning.git> → 코드 다운받기



2장 실습용 코드 업로드하기

- 다운로드 받은 2장 실습용 코드 주피터 노트북에 업로드하기

The screenshot shows the JupyterLab interface. At the top, there's a 'jupyter' logo and 'Quit' and 'Logout' buttons. Below that, there are tabs for 'Files', 'Running', and 'Clusters'. A message says 'Select items to perform actions on them.' Below this, there's a file browser showing a directory structure. A red arrow labeled '1' points to the 'Upload' button in the top right. A red arrow labeled '2' points to the file 'chapter-02.ipynb' in the file browser. A red arrow labeled '3' points to the 'Upload' button in the file selection dialog.

1

2

3

jupyter

Quit Logout

Files Running Clusters

Select items to perform actions on them.

Upload New

Name Last Modified File size

11분 전

chapter_02

다운로드 > Grokking-Deep-Reinforcement-Learning

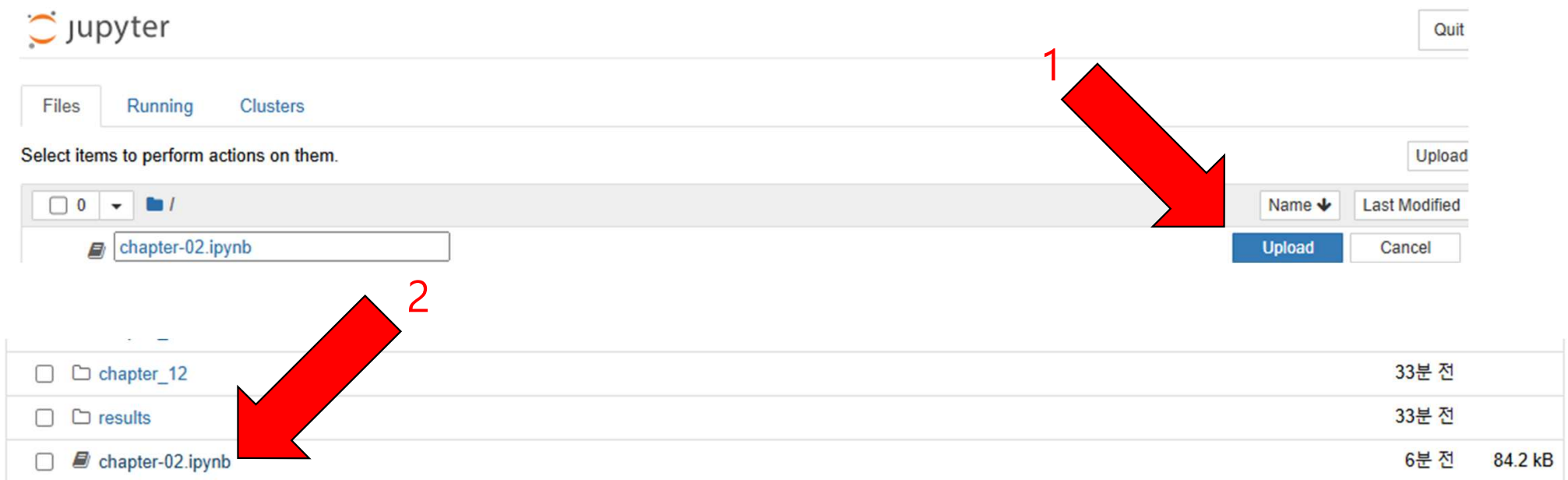
이름	수정한 날짜	유형	크기
chapter-02.ipynb	2025-04-10 10:53	IPYNB 파일	83KB
README.md	2025-04-10 10:25	Markdown 원본 ...	1KB
.idea	2025-04-10 10:59	파일 폴더	
.git	2025-04-10 10:58	파일 폴더	

모든 파일 (*.*)

!바일에서 업로드 열기(O) 취소

2장 실습용 코드 업로드하기

- 다운로드 받은 2장 실습용 코드 주피터 노트북에 업로드하기



The image shows the JupyterLab interface with the 'Files' tab selected. A red arrow labeled '1' points to the 'Upload' button in the top right corner. Below the file browser, a red arrow labeled '2' points to the 'chapter-02.ipynb' file in the file list.

jupyter

Quit

Files Running Clusters

Select items to perform actions on them.

0 /

chapter-02.ipynb

Upload

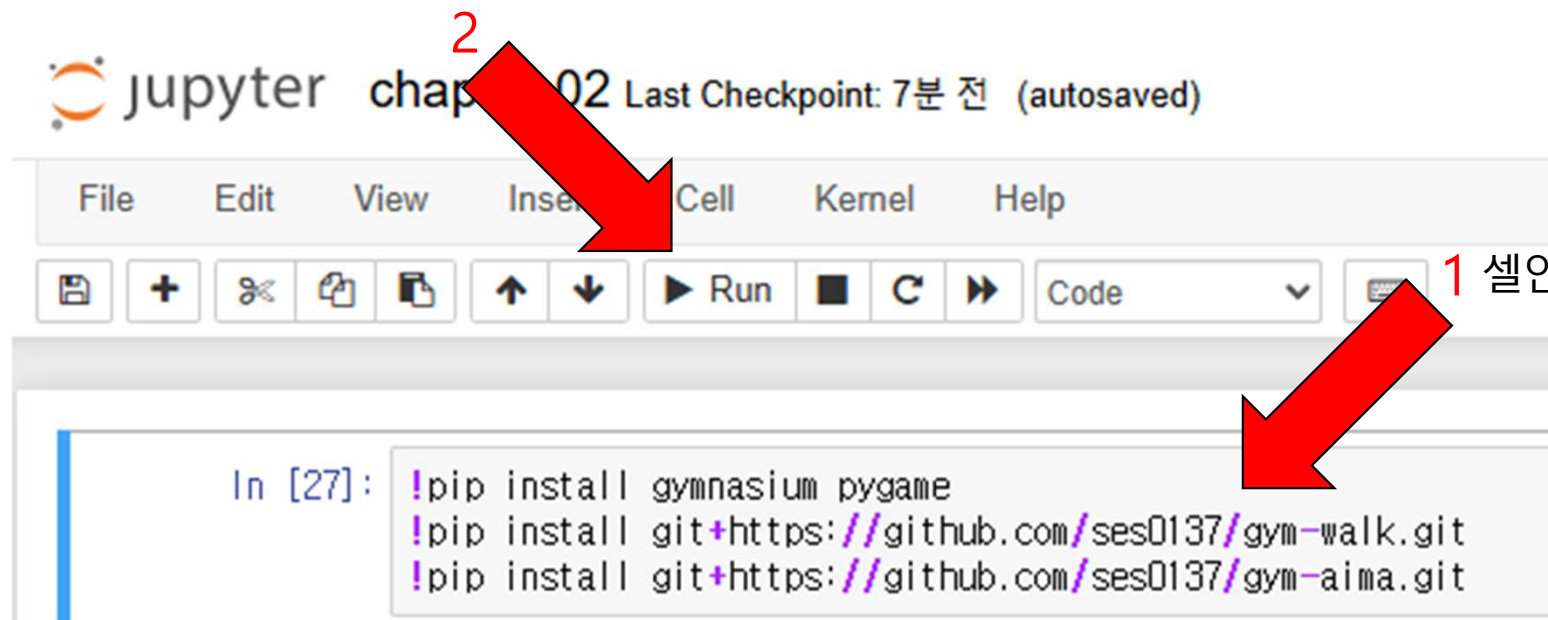
Name Last Modified

Upload Cancel

<input type="checkbox"/>	chapter_12	33분 전	
<input type="checkbox"/>	results	33분 전	
<input type="checkbox"/>	chapter-02.ipynb	6분 전	84.2 kB

2장 라이브러리 설치하기

- 2장 실습에 필요한 라이브러리 설치하기



1 셀안에 커서 먼저 두기