

1인1프로그래밍언어 마스터제

# 집중프로그래밍언어(코딩클리닉) 교육

유재명

전남대학교 SW중심대학사업단

# 1인1프로그래밍언어 마스터제 소개

- 목적

- ✓실전적 SW개발역량 강화를 위하여 SW전공(소프트웨어공학과, 컴퓨터 정보통신공학과, AI융합학부)대상으로 코딩클리닉 교육

- 운영

- ✓학기중 프로그래밍언어(C/C++, **JAVA**, Python) 교과목 운영

- 교육일정

- 일시: 2025. 10. 16(목) 15:00 ~ 17:00
  - 장소: 창조관 101호

- 교육자료

- URL: <https://github.com/yoojaemyeong/java>

# 깃허브 자료실 (<https://github.com/yoojaemyeong/java>)

📖 README

## 1인1프로그래밍언어 마스터제

### 코딩클리닉

- 교육일시: 2025.10.16.(목) 15:00 ~ 17:00
- 교육내용: 자바
- SW개발역량 강화를 위하여

### 목차

- [1. 프로그램가이드](#)
- [2. 자바 개요](#)
- [3. try-catch-finally](#)
- [4. 최적화-overloading 주의사항](#)
- [5. 최적화-상수대신 열거타입사용](#)

 yoojaemyeong 강의자료	
📄 01.키보드입력방법 및 활용가능 매직셀.ipynb	강의자료
📄 02.파일생성.ipynb	강의자료
📄 03.java-overview.ipynb	강의자료
📄 04.overloading.ipynb	강의자료
📄 05.try-catch-finally.ipynb	강의자료
📄 06.개발노하우.ipynb	강의자료
📄 07.매직셀명령어정리.ipynb	강의자료
📄 1인1프로그래밍언어 마스터제.pdf	프로그램개요 및 자바환경구성
📄 Dockerfile	강의자료-Dockerfile

# 자바 프로그램 실습 환경 만들기

1. 운영체제 - 아키텍처 소개
2. 도커 설치(마이크로 서비스)
  - <https://www.docker.com>
3. 컨테이너 기반 자바 개발환경 만들기
  - 개인 작업폴더:

```
mkdir C:\Users\[사용자홈]\Documents\java_up
```

- 아키텍처

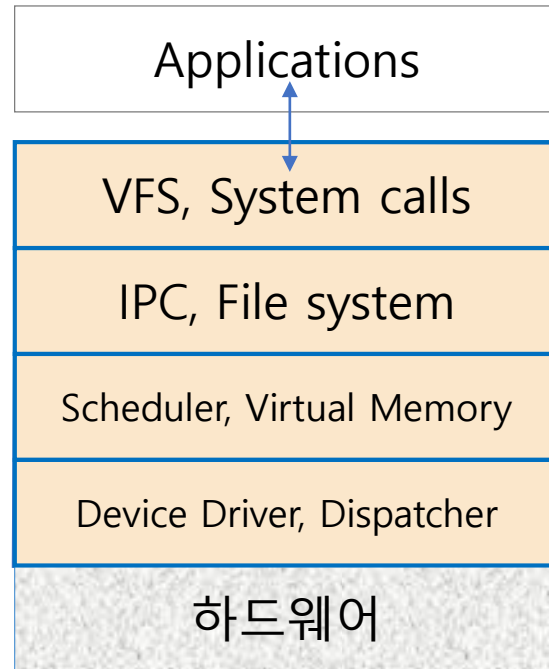
- Monolithic Service : 하나의 큰 목적이 있는 서비스 또는 애플리케이션에 여러 기능이 **통합**되어 있는 구조
- Micro Service : 시스템 전체가 하나의 목적을 지향하는 바는 같으나, 개별 기능 각각을 개발해 연결하는 데서 차이가 있음. 곧 보안, 인증 등과 관련된 기능이 **독립**된 서비스를 구성하고 있으며 다른 서비스들도 독립적으로 동작할 수 있는 구조

- Micro Service: 컨테이너 인프라 환경

- 컨테이너를 중심으로 구성된 인프라 환경
- 컨테이너란 하나의 운영체제안에서 다른 프로세스에 영향을 받지 않고 독립적으로 실행되는 프로세스 격리 환경을 활용하는 커널 생성 기술
  - **Cgroup(Control Group)**: CPU, 메모리, 네트워크 대역폭과 같은 자원을 제한하고 격리하는 기능
  - **Namespace** : 시스템 리소스를 프로세스 전용 자원처럼 분리하는 기능
    - ❖ 시스템 리소스로 Volume Mount, PID, Network, IPC, UTS(Unix Time Sharing), USER

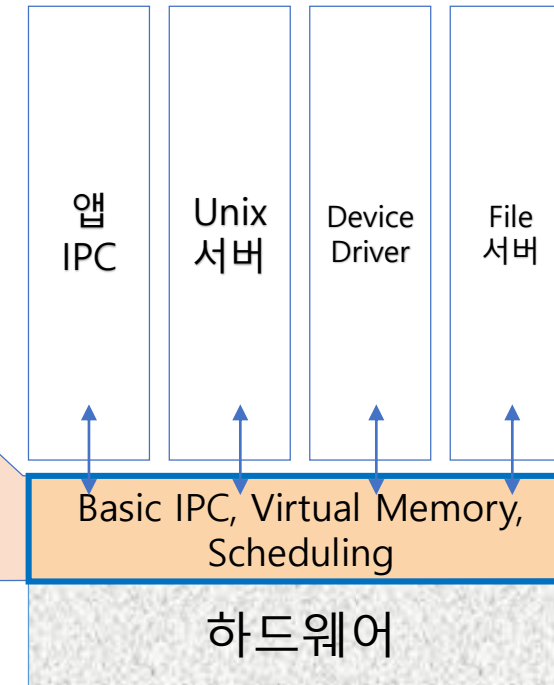
# 1. 모놀리틱 vs. 마이크로서비스

## • 모놀리틱 커널



- 소프트웨어가 하나의 결합된 코드로 구성
- 초기 단계에서 설계가 용이, 단순하나 규모가 커지고, 수정이 많을 경우, 연관성이 복잡해지고, 추가 인프라 증설이 어려움

## 마이크로 커널

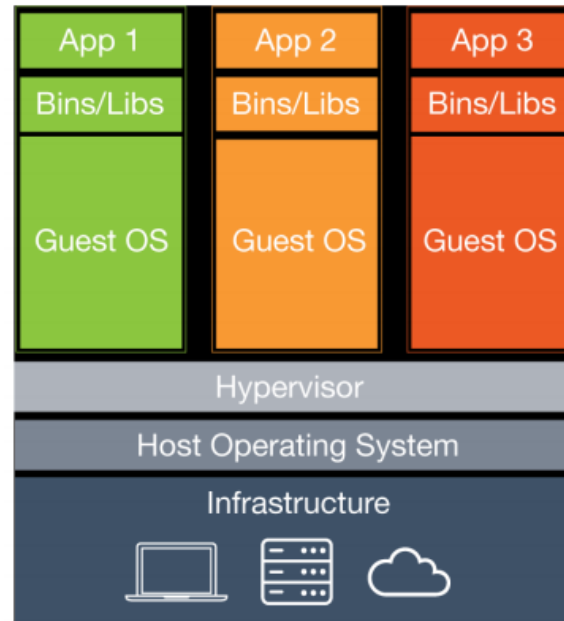


- 개별 기능에 따라 각각 개발
- 각 모듈별 초기 개발이 어려우나, 추후 재사용이 쉽고, 추가 서비스 생성 또는 확장성 좋음

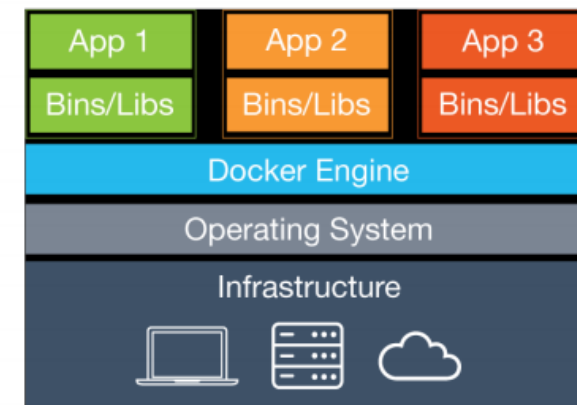
# 1. 주요 클라우드 플랫폼

- 주요 클라우드 플랫폼

- AWS(Amazon Web Services)
  - 주요 서비스: EC2, S3, Lambda, RDS, EKS 등
- Microsoft Azure
  - 주요 서비스: VM, Blob Storage, AKS, Azure ML 등
- Google Cloud Platform
  - 주요 서비스: Compute Engine, GKE, BigQuery, Vertex AI 등
- Oracle Cloud Infrastructure
  - 주요 서비스: OCI compute, Autonomous DB 등
- IBM Cloud
  - 주요 서비스: IBM Watson, Code Engine, Cloud Functions 등



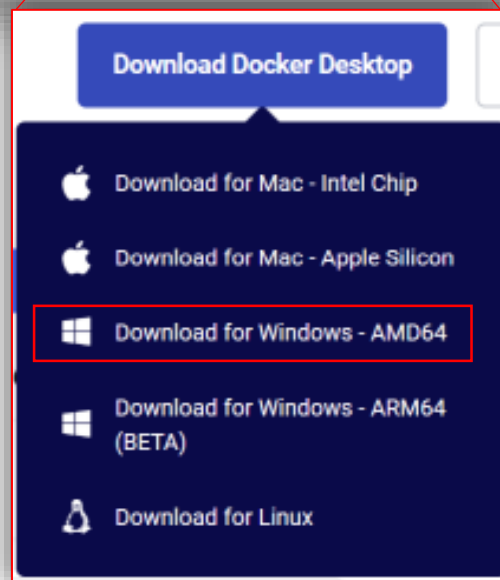
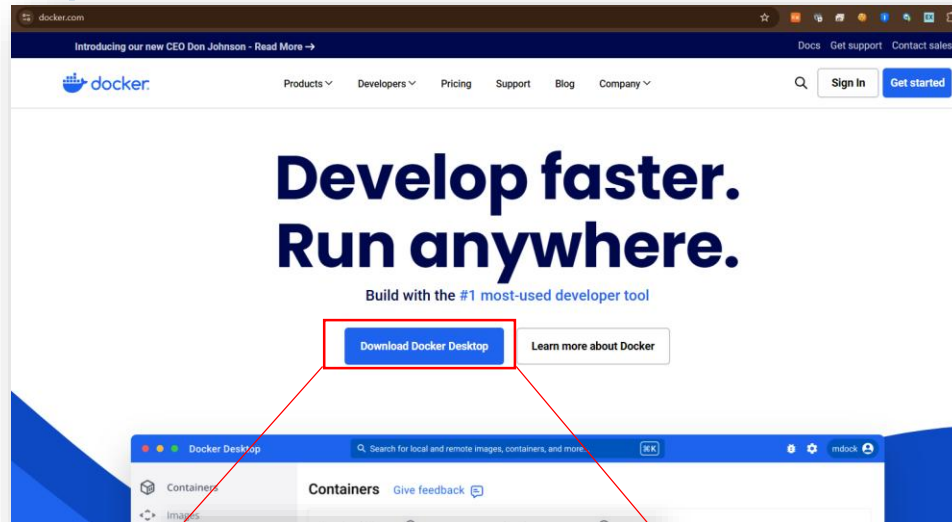
Hypervisor based Virtualization




Container Virtualization

## 2. Docker Desktop 다운로드

- <https://www.docker.com/>



다운로드 및 설치

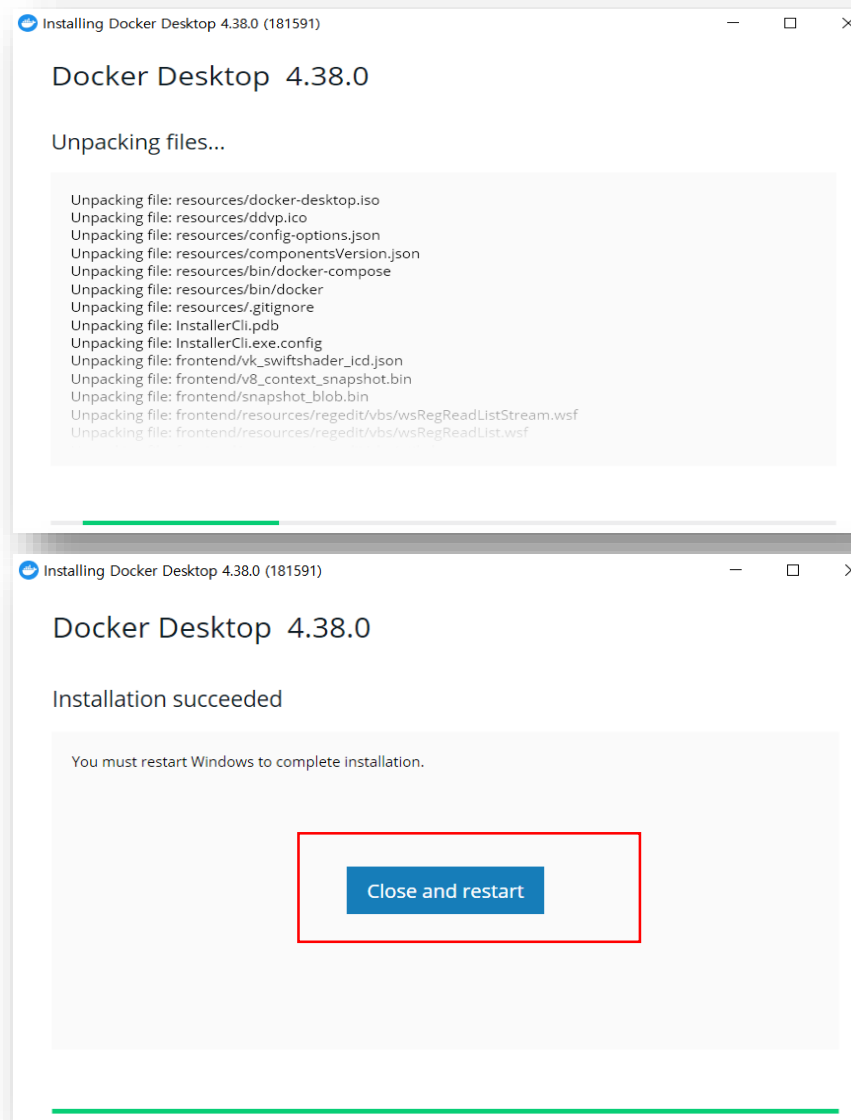
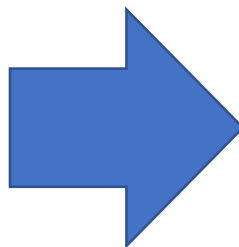
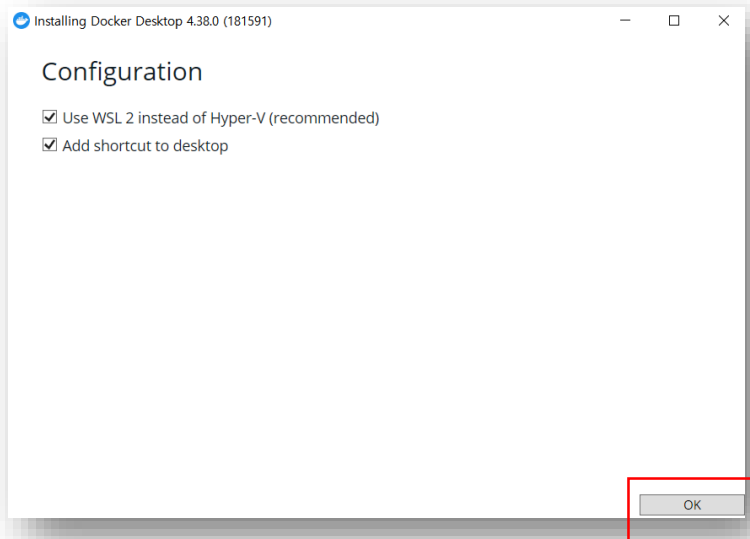
 Download for Windows - AMD64



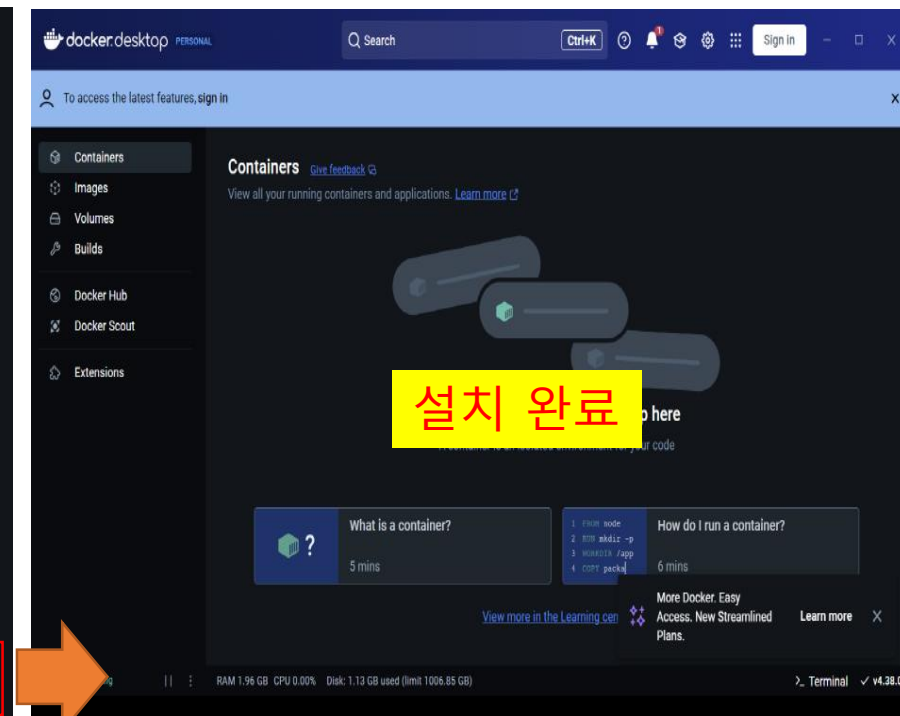
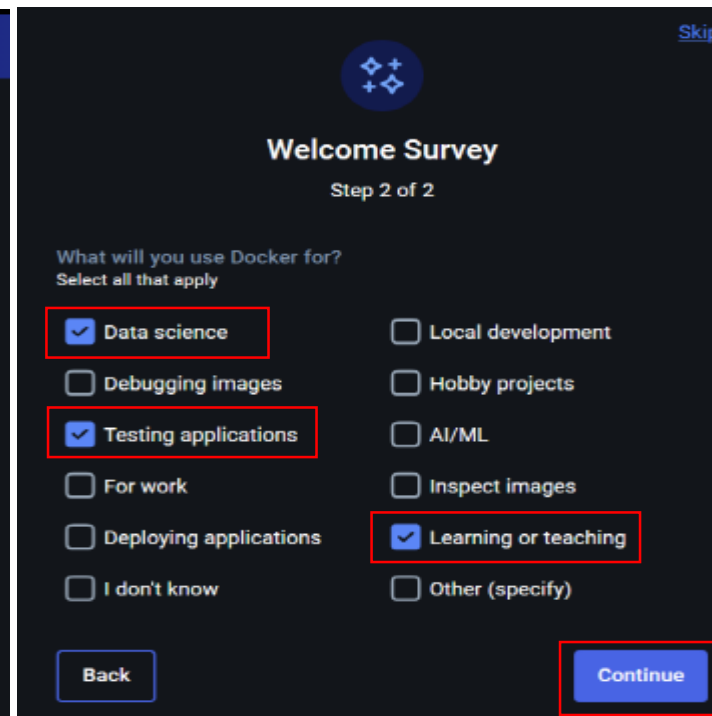
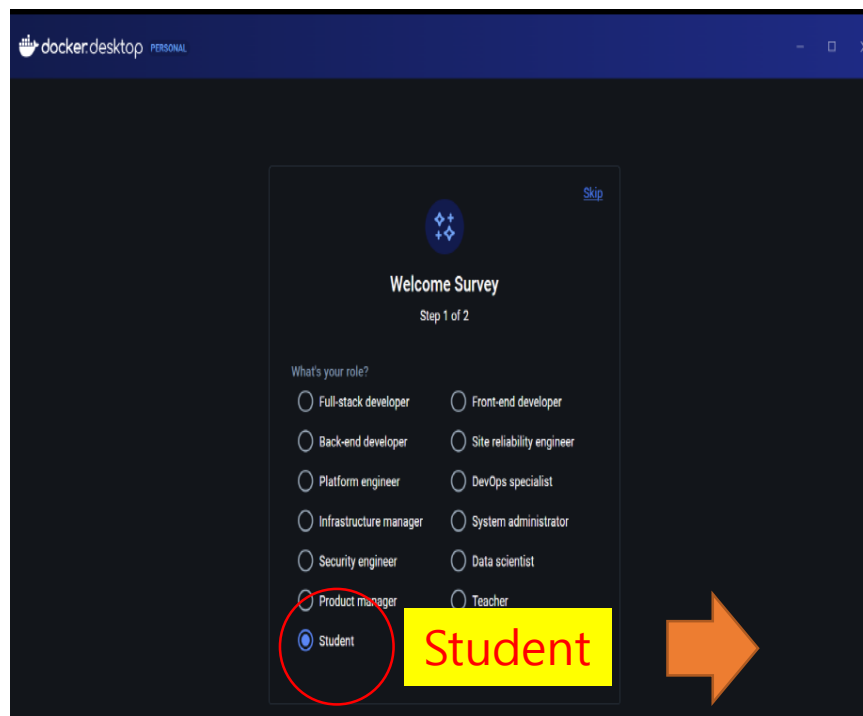
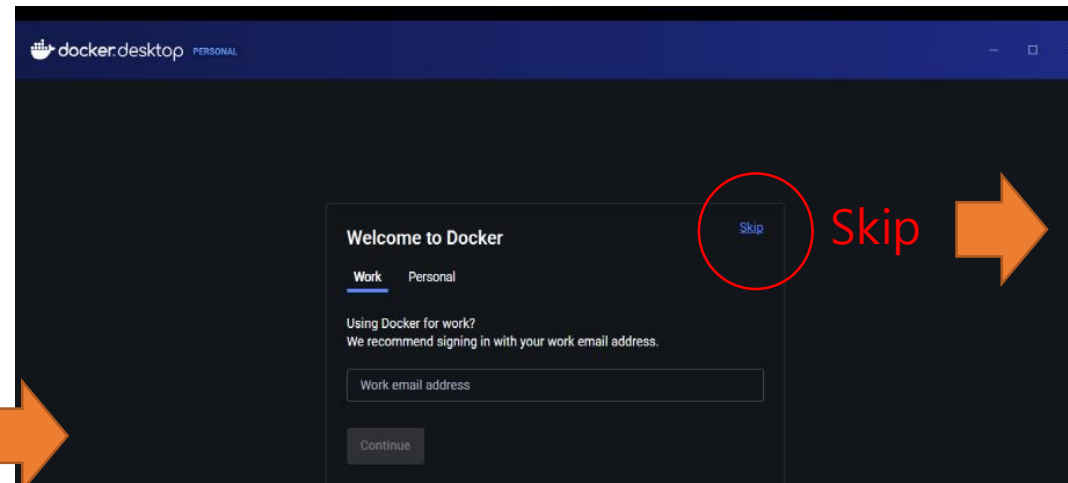
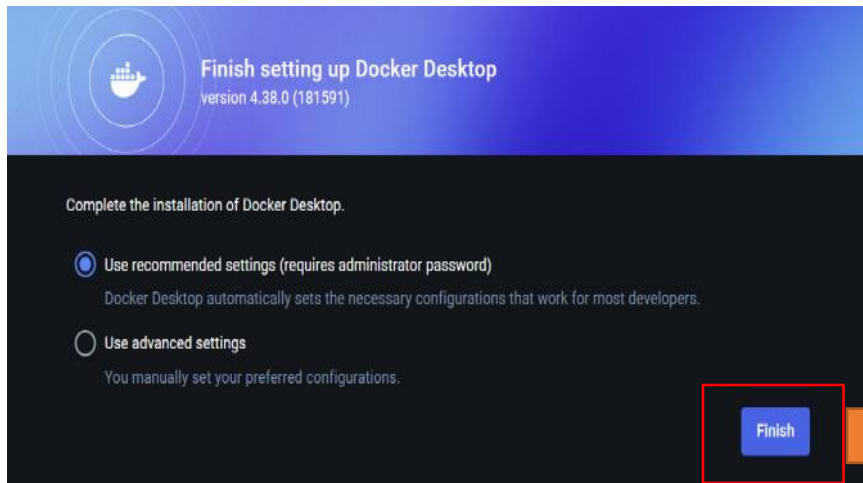
## 2. Docker Desktop 설치(1)

다운로드 및 **설치**

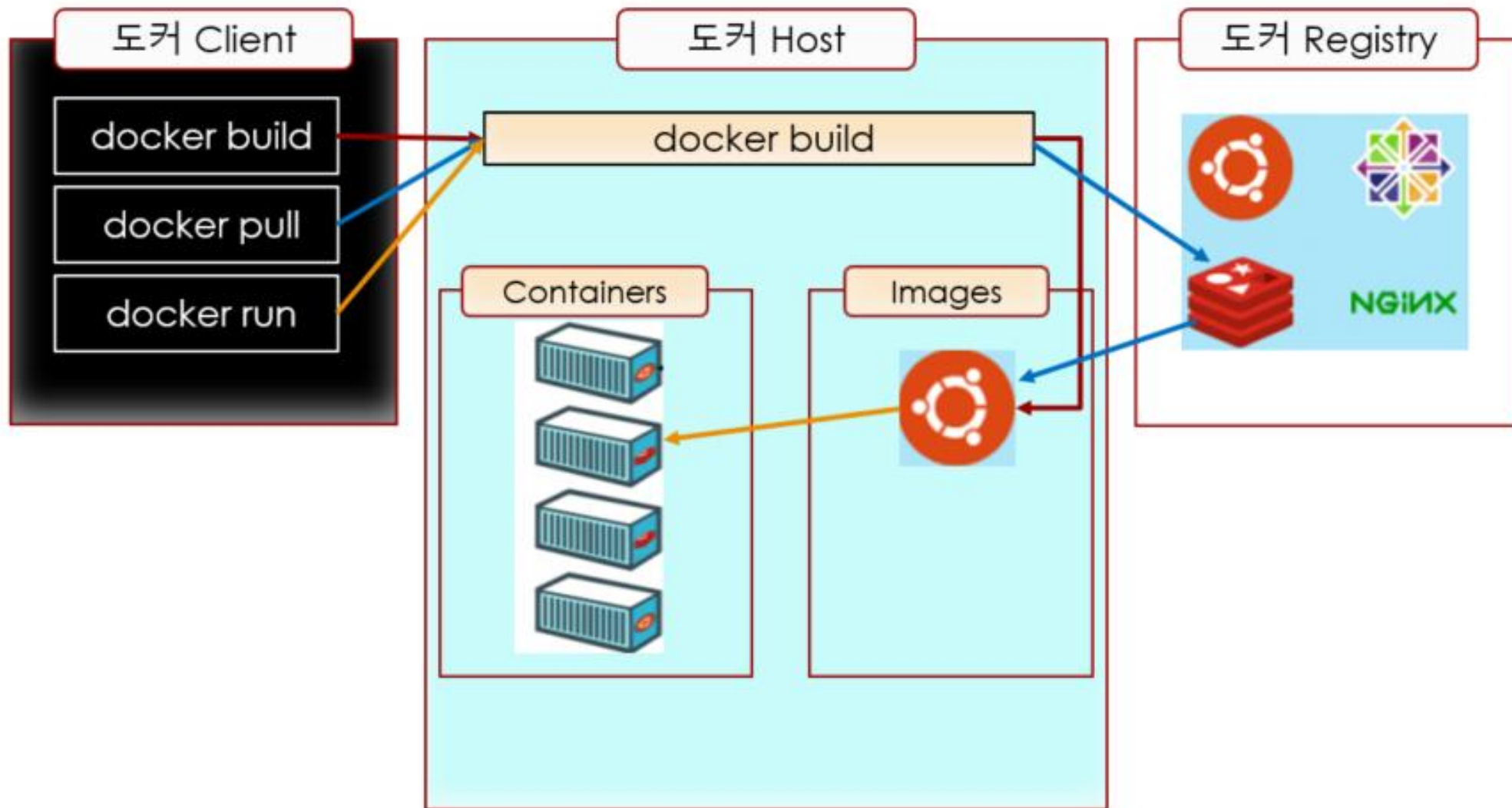
Download for Windows - AMD64



## 2. Docker Desktop 설치(2)



# Docker Architecture



## 2 자바 컨테이너 생성 절차

### • 진행순서

1. Docker Desktop 다운로드

<https://www.docker.com>

2. Docker 기본 설치

3. mkdir Java && cd Java

4. notepad Dockerfile <= 배포한 Dockerfile 활용

5. 도커이미지 생성

docker build -t jdk21-ijava .

6. 컨테이너 생성

docker run -itd --name javaup -e TZ=Asia/Seoul -v "%cd%:/home/jovyan" -p 9999:8888 jdk21-ijava

### Dockerfile

```
# Jupyter
FROM jupyter/datascience-notebook

# 루트로 전환해 OS 패키지 설치
USER root
RUN apt-get update && \
    apt-get install -y --no-install-recommends openjdk-21-
jdk wget unzip && \
    rm -rf /var/lib/apt/lists/*

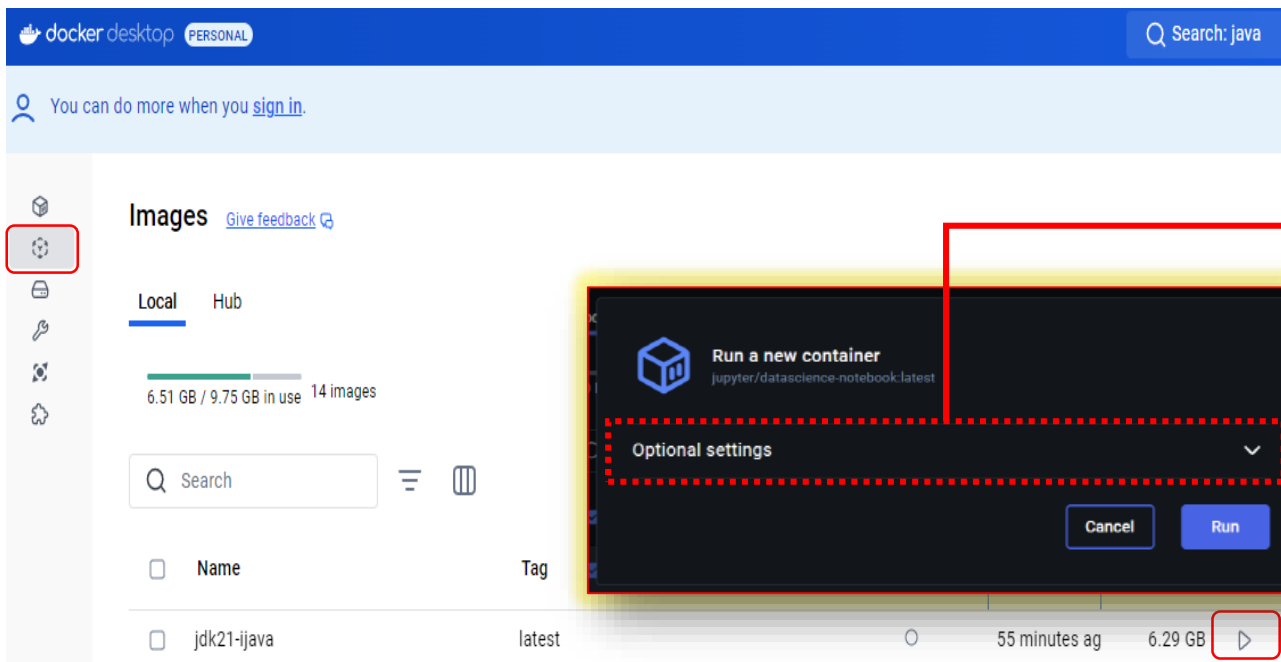
# JDK 경로 설정
ENV JAVA_HOME=/usr/lib/jvm/java-21-openjdk-amd64
ENV PATH=$JAVA_HOME/bin:$PATH

# 다시 jovyan 사용자로
USER ${NB_UID}

# IJava(자바 커널, 1.3.0) 설치
RUN wget -q
https://github.com/SpencerPark/IJava/releases/download/v1.
3.0/ijava-1.3.0.zip && \
    unzip -q ijava-1.3.0.zip -d /tmp/ijava && \
    python3 /tmp/ijava/install.py --sys-prefix && \
    rm -rf /tmp/ijava ijava-1.3.0.zip

# 기본 작업 폴더
WORKDIR /home/jovyan
```

### 3. jdk21-ijava 이미지 이용하여 컨테이너 구성



**Run a new container**  
jdk21-ijava:latest

**Optional settings**

Container name  
**javaup**

A random name is generated if you do not provide one.

**Ports**  
Enter "0" to assign randomly generated host ports.  
**9999** :8888/tcp

**Volumes**

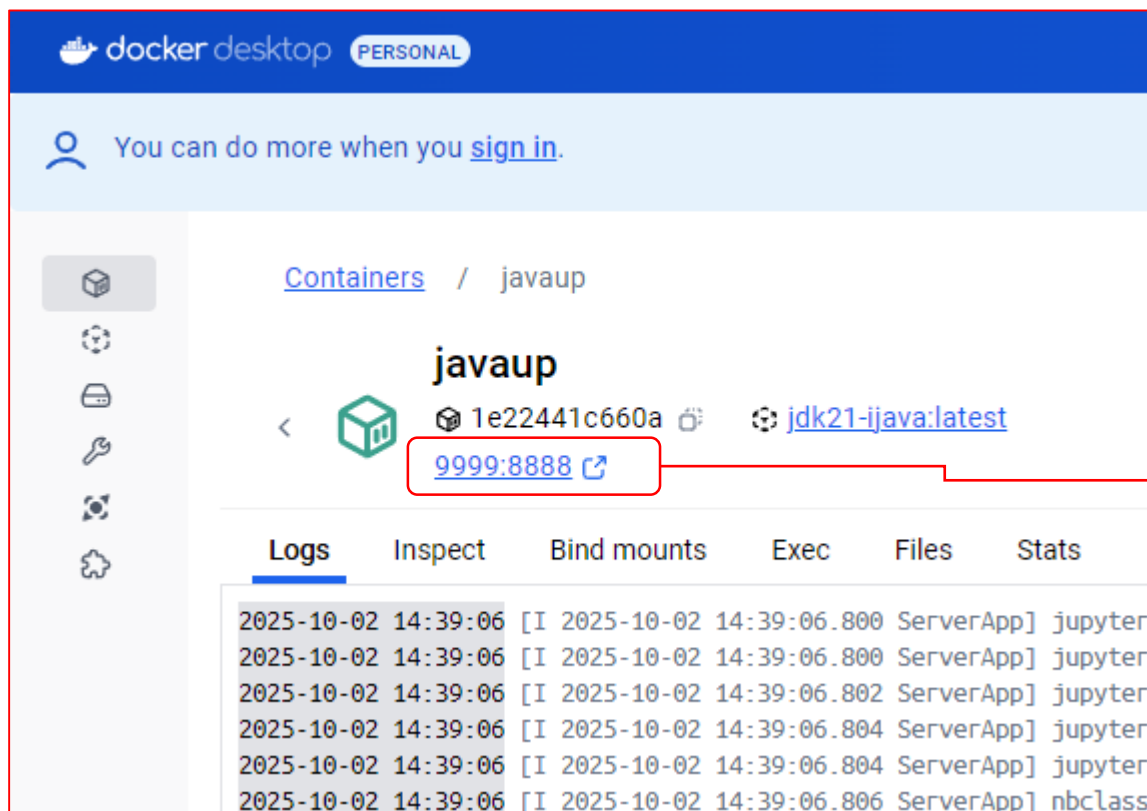
Host path	Container path
<b>c:\Users\WDocuments\Wjava_up</b>	<b>/home/jovyan/work</b>

**Environment variables**

Variable	Value
TZ	Asia/Seoul

**Run**

### 3. Jupyter/datascience-notebook 이미지 이용하여 컨테이너 구성






docker desktop PERSONAL

You can do more when you [sign in](#).

Containers / javaup

**javaup**

<  1e22441c660a  [jdk21-ijava:latest](#)

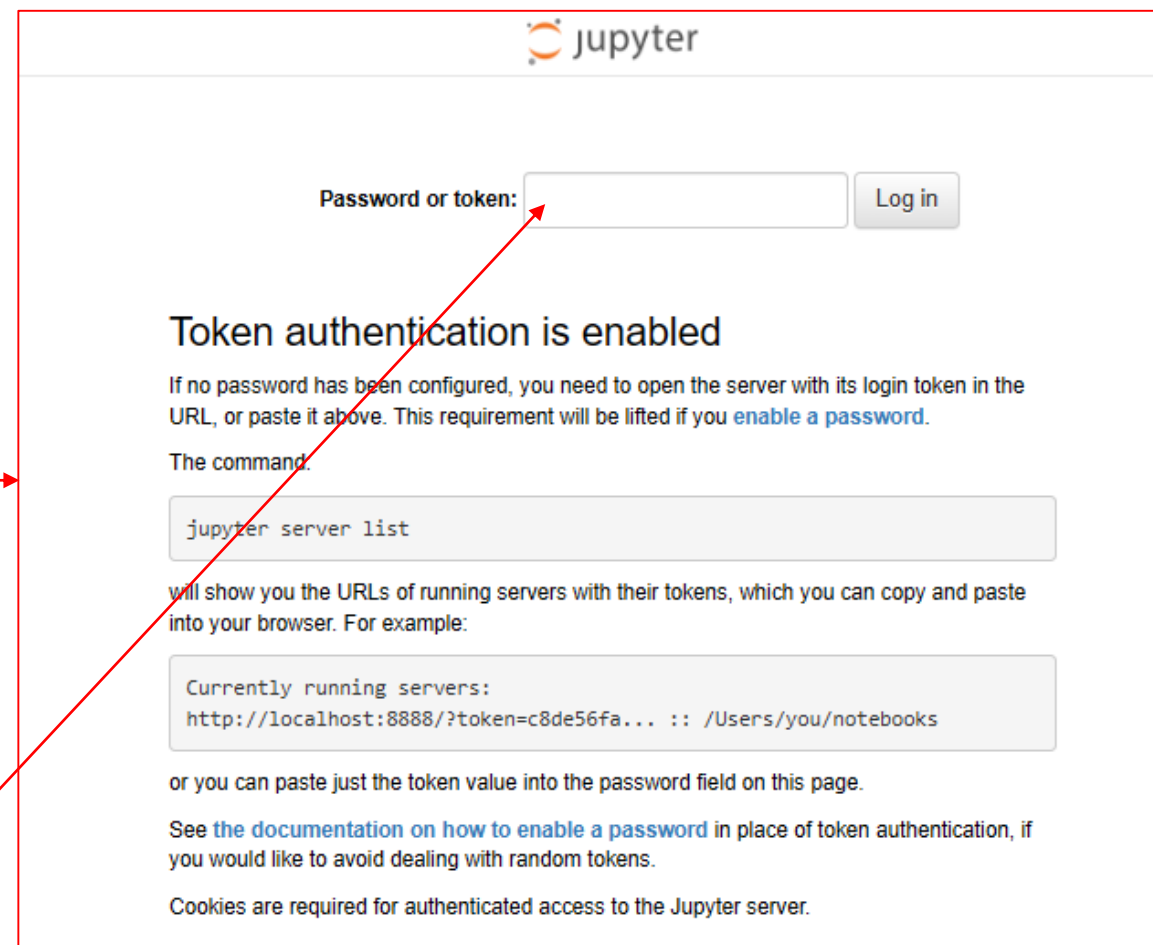
[9999:8888](#) 

Logs Inspect Bind mounts Exec Files Stats

2025-10-02 14:39:06 [I 2025-10-02 14:39:06.800 ServerApp] jupyter  
2025-10-02 14:39:06 [I 2025-10-02 14:39:06.800 ServerApp] jupyter  
2025-10-02 14:39:06 [I 2025-10-02 14:39:06.802 ServerApp] jupyter  
2025-10-02 14:39:06 [I 2025-10-02 14:39:06.804 ServerApp] jupyter  
2025-10-02 14:39:06 [I 2025-10-02 14:39:06.804 ServerApp] jupyter  
2025-10-02 14:39:06 [I 2025-10-02 14:39:06.806 ServerApp] nbclass

or copy and paste one of these URLs:

2025-10-02 14:39:07 http://1e22441c660a:8888/lab?token=77dfd2d840971be3c60cdb214987dea95f93bde05fb440fa  
2025-10-02 14:39:07 http://127.0.0.1:8888/lab?token=77dfd2d840971be3c60cdb214987dea95f93bde05fb440fa



jupyter

Password or token:  Log in

**Token authentication is enabled**

If no password has been configured, you need to open the server with its login token in the URL, or paste it above. This requirement will be lifted if you [enable a password](#).

The command:

```
jupyter server list
```

will show you the URLs of running servers with their tokens, which you can copy and paste into your browser. For example:

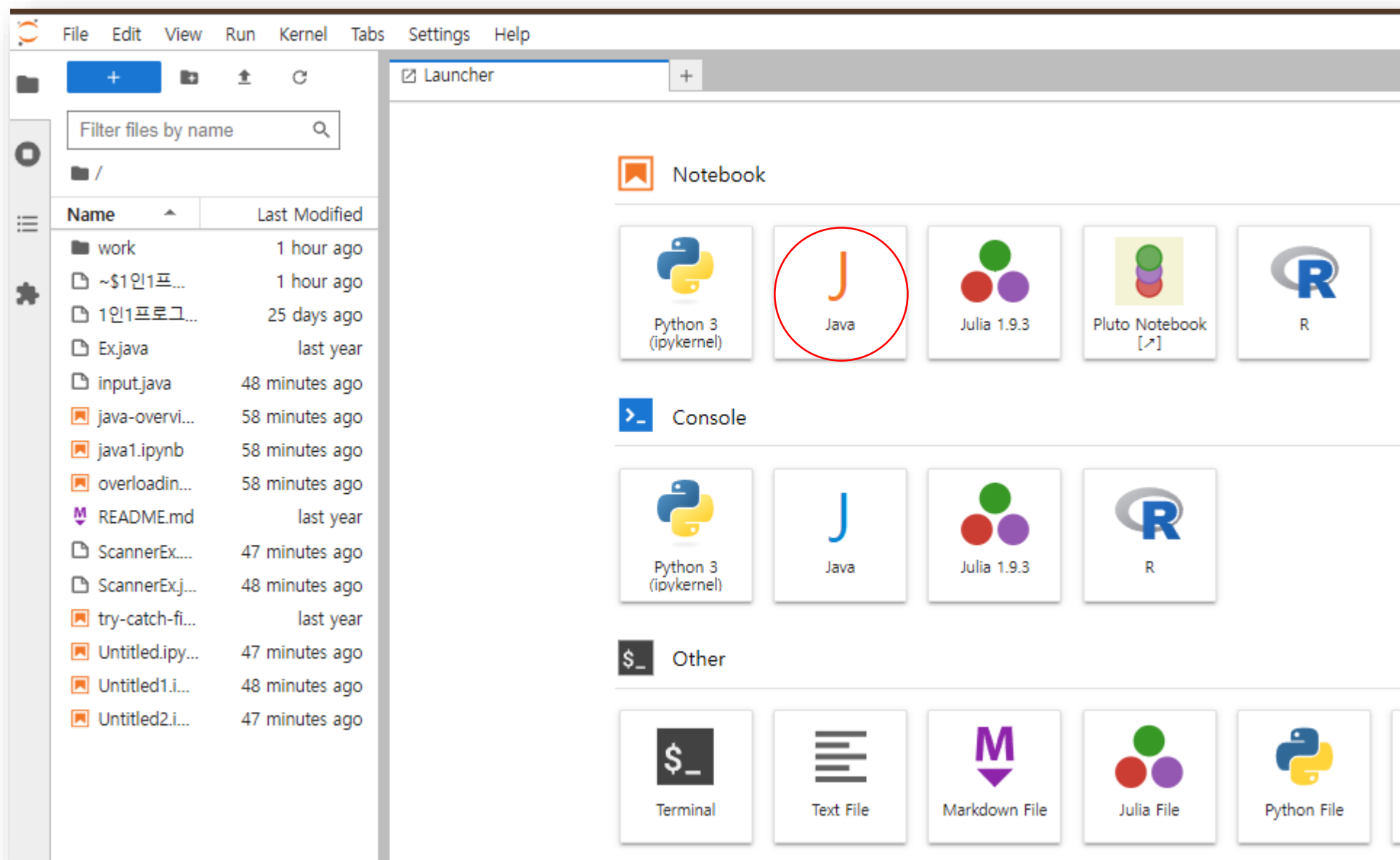
```
Currently running servers:  
http://localhost:8888/?token=c8de56fa... :: /Users/you/notebooks
```

or you can paste just the token value into the password field on this page.

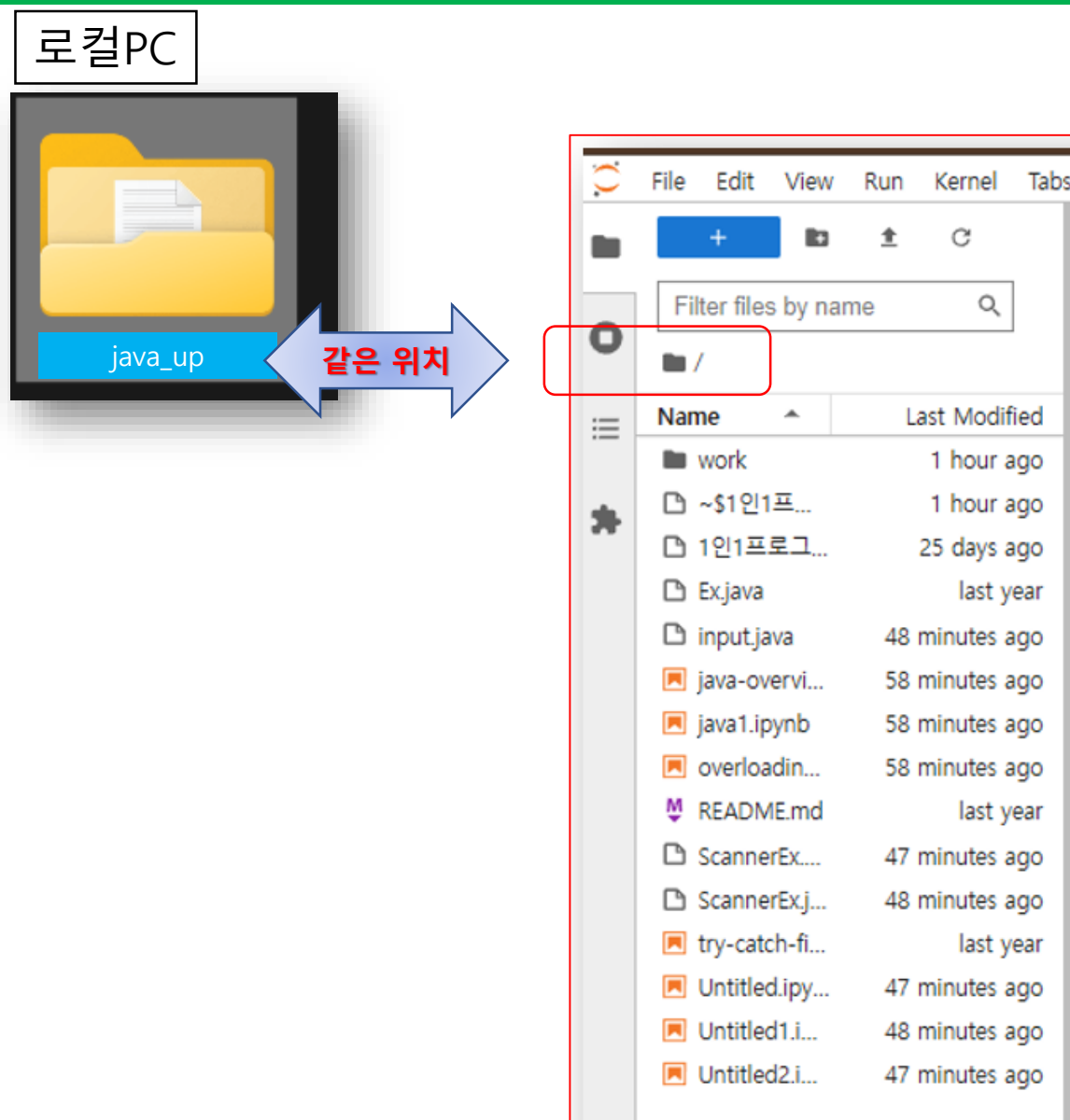
See [the documentation on how to enable a password](#) in place of token authentication, if you would like to avoid dealing with random tokens.

Cookies are required for authenticated access to the Jupyter server.

### 3. Jupyter/datascience-notebook 이미지 이용하여 컨테이너 구성

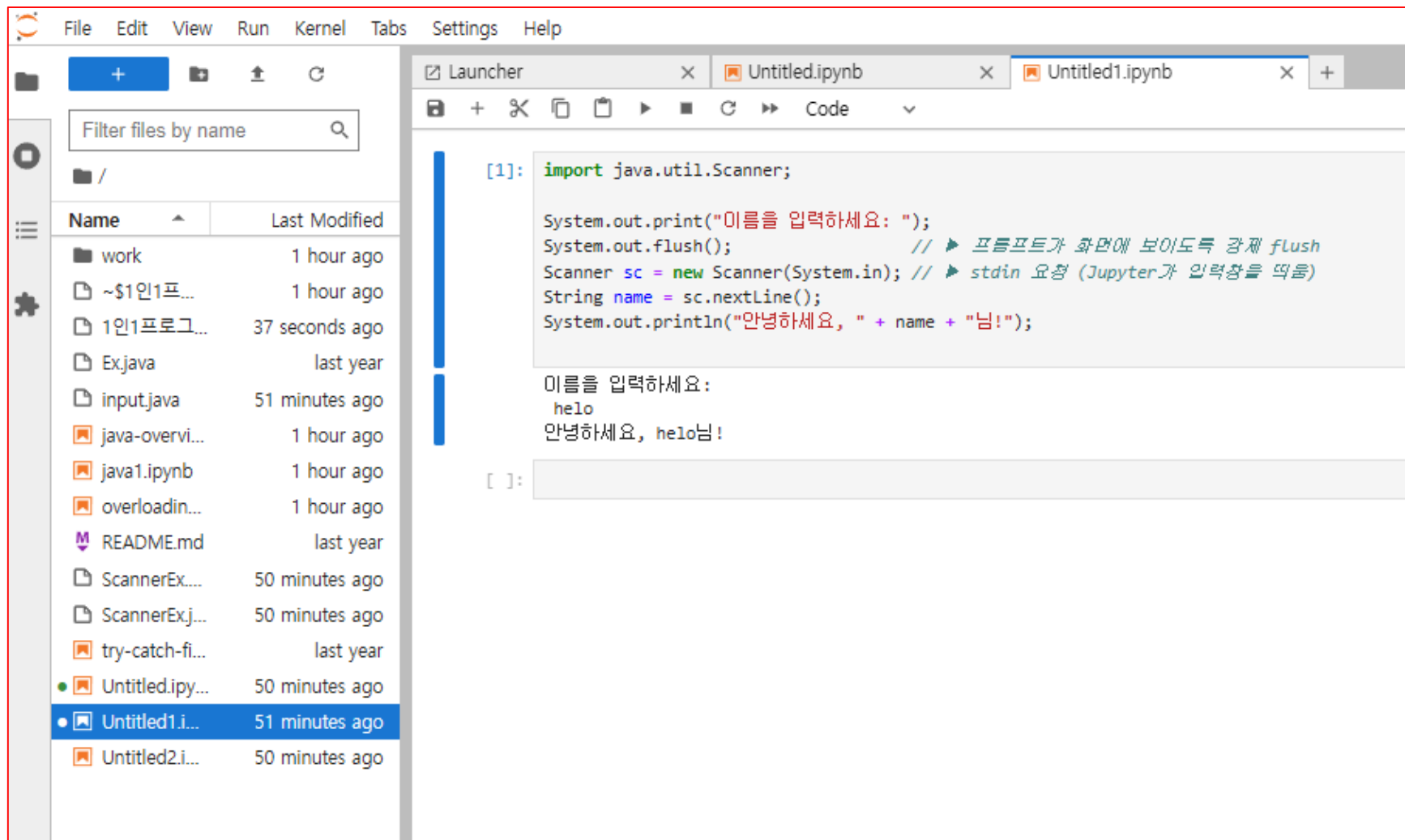


## 2. Jupyter/datascience-notebook 이미지 이용하여 컨테이너 구성





# 실행화면



The screenshot displays the JupyterLab environment. On the left, the file explorer shows a directory structure with files like 'work', 'Ex.java', 'input.java', 'java-overvi...', 'java1.ipynb', 'overloadin...', 'README.md', 'ScannerEx....', 'ScannerEx.j...', 'try-catch-fi...', 'Untitled.ipy...', 'Untitled1.i...', and 'Untitled2.i...'. The file 'Untitled1.i...' is selected. On the right, the code editor shows the following Java code:

```
[1]: import java.util.Scanner;

System.out.print("이름을 입력하세요: ");
System.out.flush(); // ▶ 프롬프트가 화면에 보이도록 강제 flush
Scanner sc = new Scanner(System.in); // ▶ stdin 요청 (Jupyter가 입력창을 띄움)
String name = sc.nextLine();
System.out.println("안녕하세요, " + name + "님!");
```

Below the code, the output of the program is shown:

```
이름을 입력하세요:
helo
안녕하세요, helo님!
```