

Microsoft VS Code

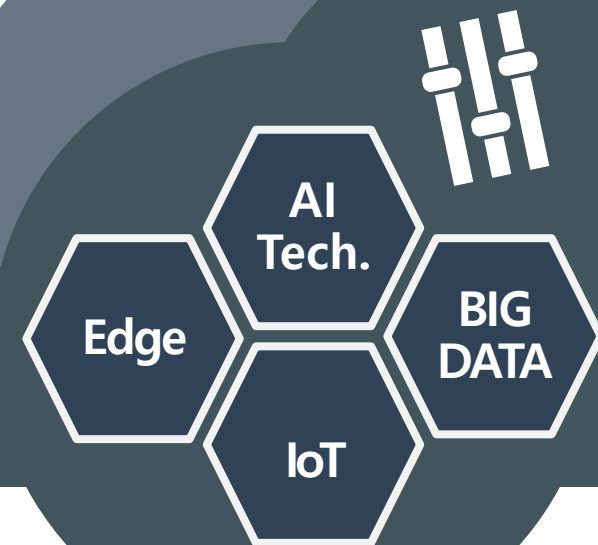
# Visual Studio Code 활용



2021년 06월 16일

(주)비알프레임

# 목차



- 1 Visual Studio Code
- 2 주요 기능
- 3 Python 활용
- 4 Jupyter Notebook 활용
- 5 Remote 서버(VM) 활용
- 6 Docker 활용

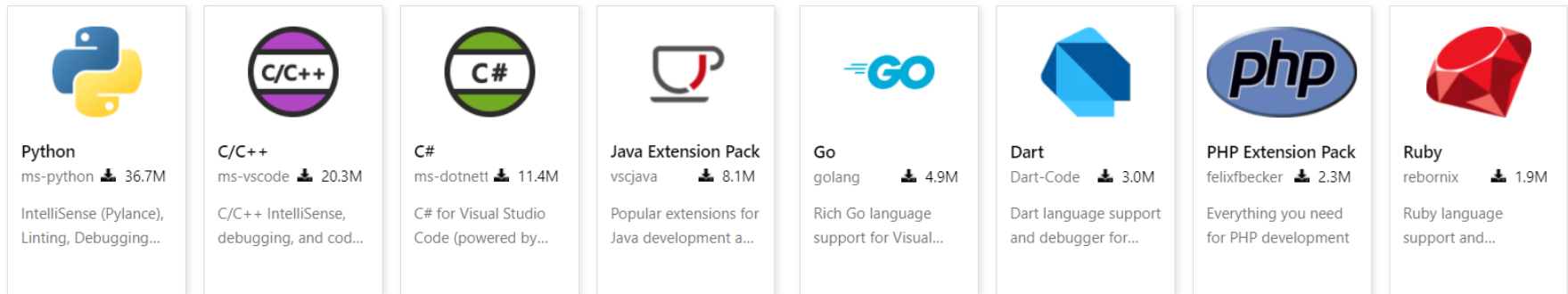
# 01. Visual Studio Code

- 개요

- Microsoft가 Windows, Mac OS, Linux용으로 개발한 소스 코드 편집기
- 편집기의 테마, 단축키, 설정 등을 Customize 가능
- 2015년 Preview 공개, 2016년 정식 버전 배포

- 기능

- 디버깅 지원과 Git 제어, 구문 강조 기능 등이 포함
- 다양한 프로그래밍 언어 지원  
(Python, C/C++, C#, Java, Go, Dart, PHP, Ruby 등)

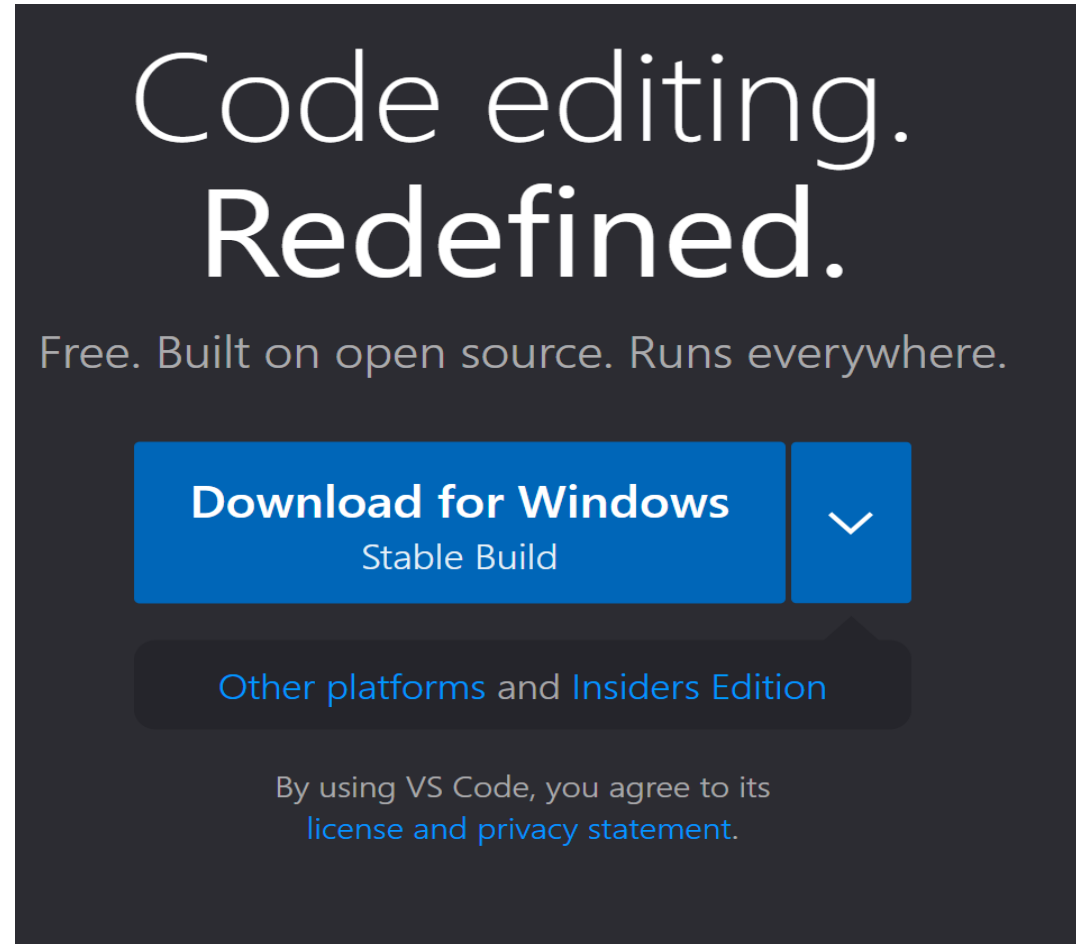


- 각 언어와 함께 사용할 수 있는 편리한 기능 제공
- 메뉴를 통해 접근할 수 없는 기능까지 Command Palette를 통해 사용 가능

# 01. Visual Studio Code

- 설치

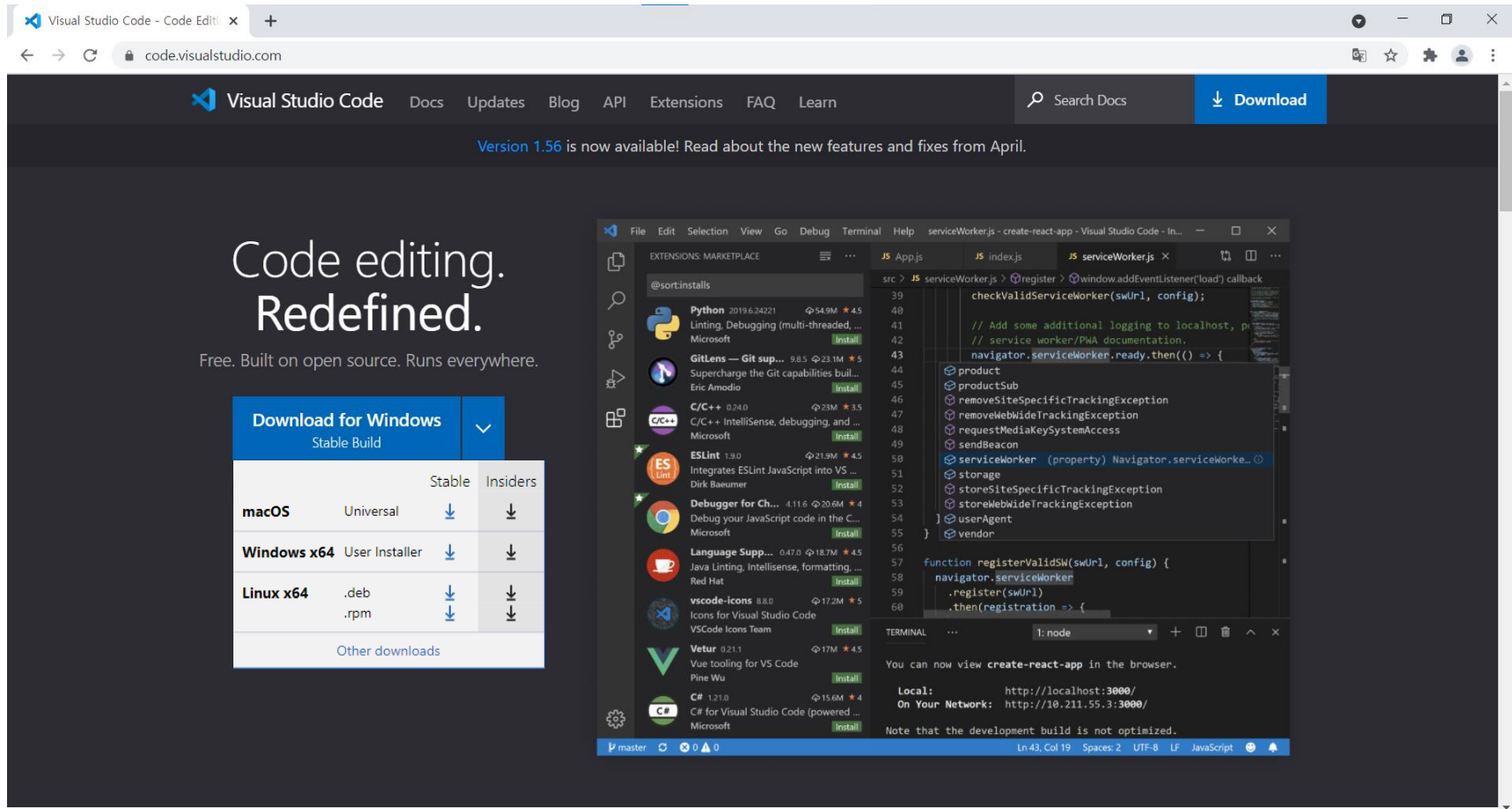
<https://code.visualstudio.com/>



# 01. Visual Studio Code

- Visual Studio Code 설치

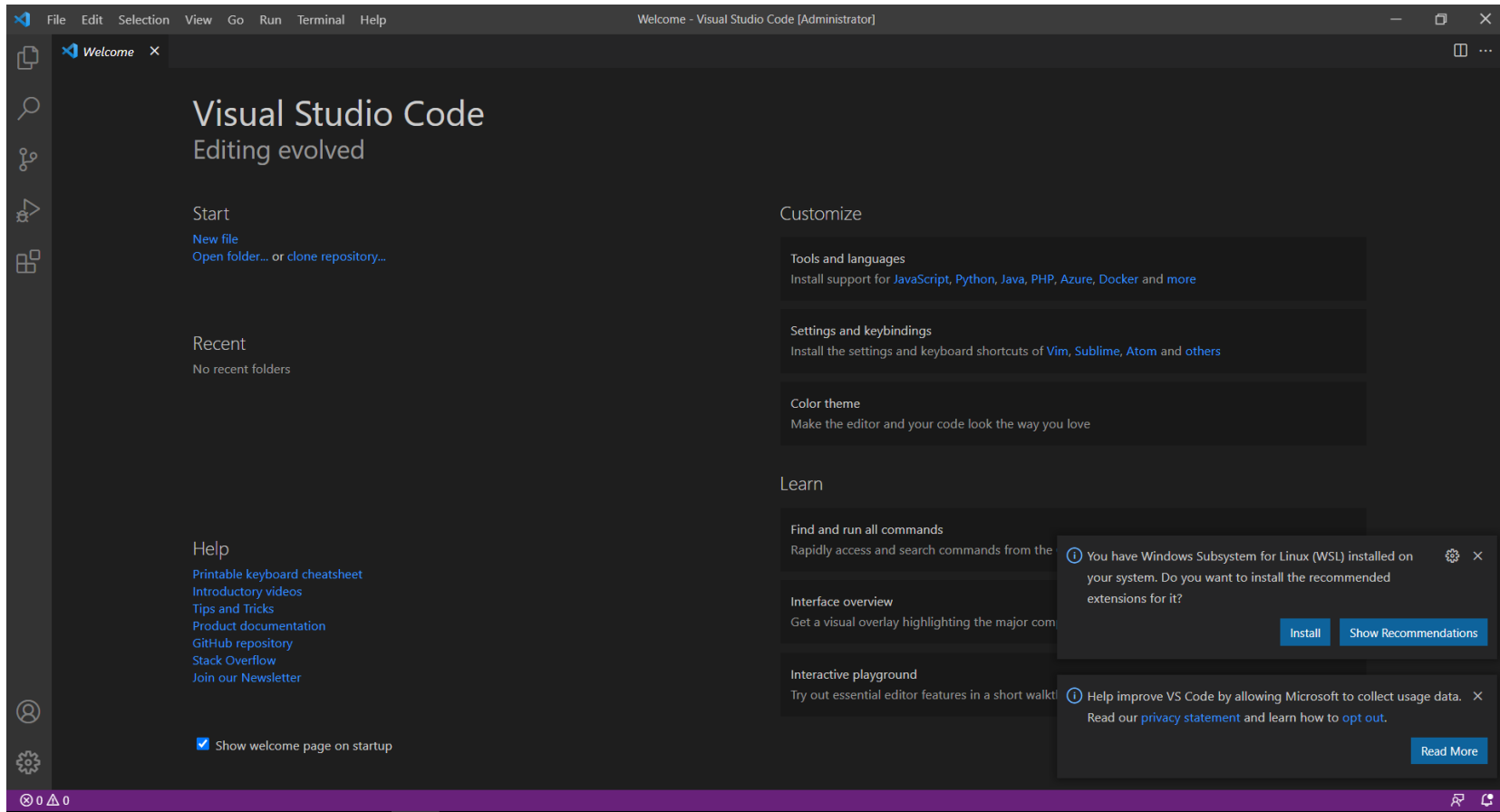
- <https://code.visualstudio.com/> 에서 각 OS에 맞게 다운로드하여 설치



# 01. Visual Studio Code

- Visual Studio Code 설치

- 설치 완료 후 실행 첫 화면



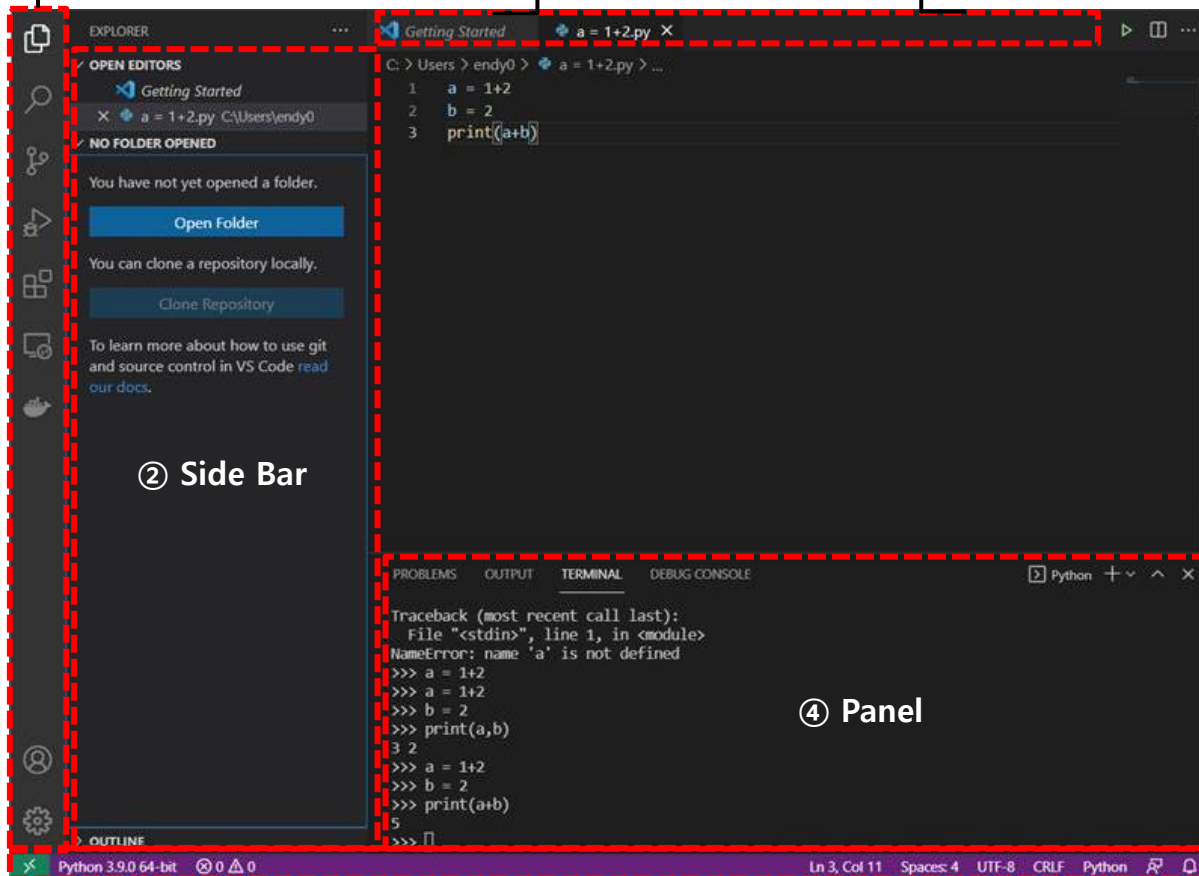
# 01. Visual Studio Code

- Visual Studio Code 실행화면

- 화면 설명

① Activity Bar

③ Editor Groups



① 탐색기, 검색, Git, Debug, Extension 등의 버튼 제공 영역

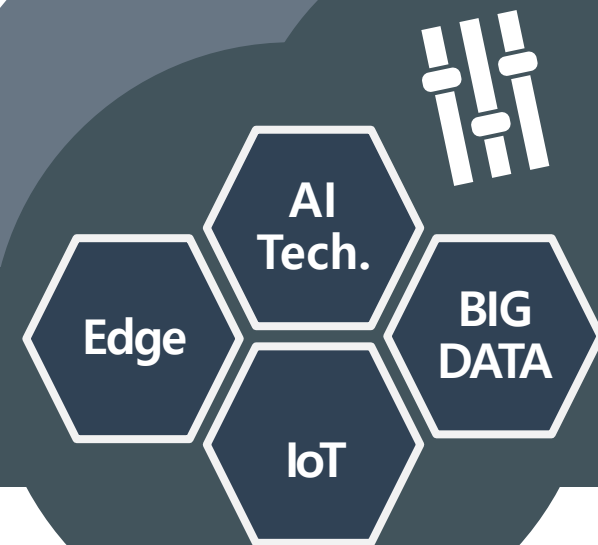
② Activity Bar에서 선택한 기능을 설정 또는 사용하는 영역

③ 파일 편집 영역

④ Output, Debugging info, Error 등을 보거나 terminal을 사용할 수 있는 영역

⑤ 열려 있는 프로젝트 및 편집한 파일에 대한 정보(언어, 인코딩, 공백 등)제공

# 목차



- 1 Visual Studio Code
- 2 주요 기능
- 3 Python 활용
- 4 Jupyter Notebook 활용
- 5 Remote 서버(VM) 활용
- 6 Docker 활용



## 02. 주요 기능

- **Command Palette**

Visual Studio Code의 모든 기능을 키보드 입력으로 쉽게 찾아 실행 가능

- **Integrated Terminal**

Powershell, Command Prompt, Bash 등 통합적으로 Shell 이용 가능

- **Extensions**

확장 프로그램 설치를 통해 언어, 디버거 및 도구를 추가하여 Work Flow 지원 가능

- **Intellisense**

parameter 정보, 변수, method 등을 포함한 다양한 코드 자동완성 기능

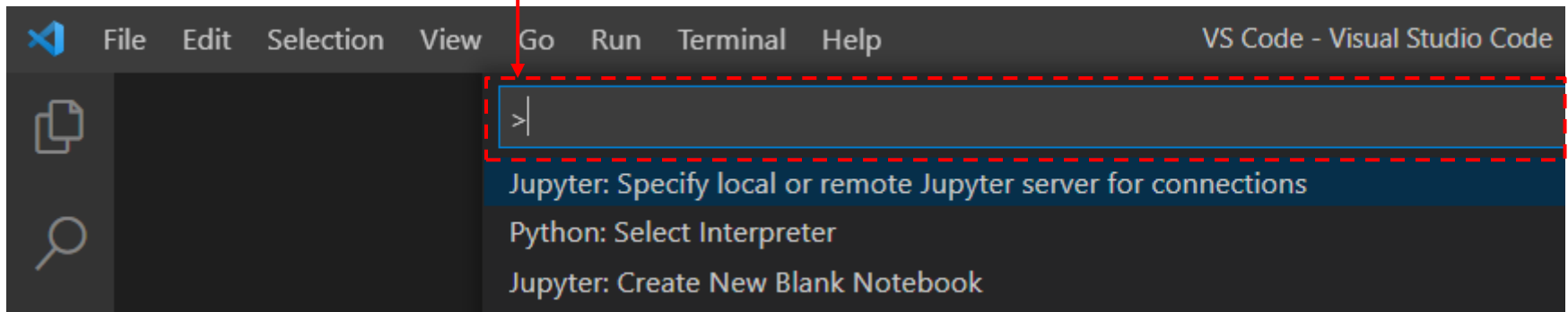
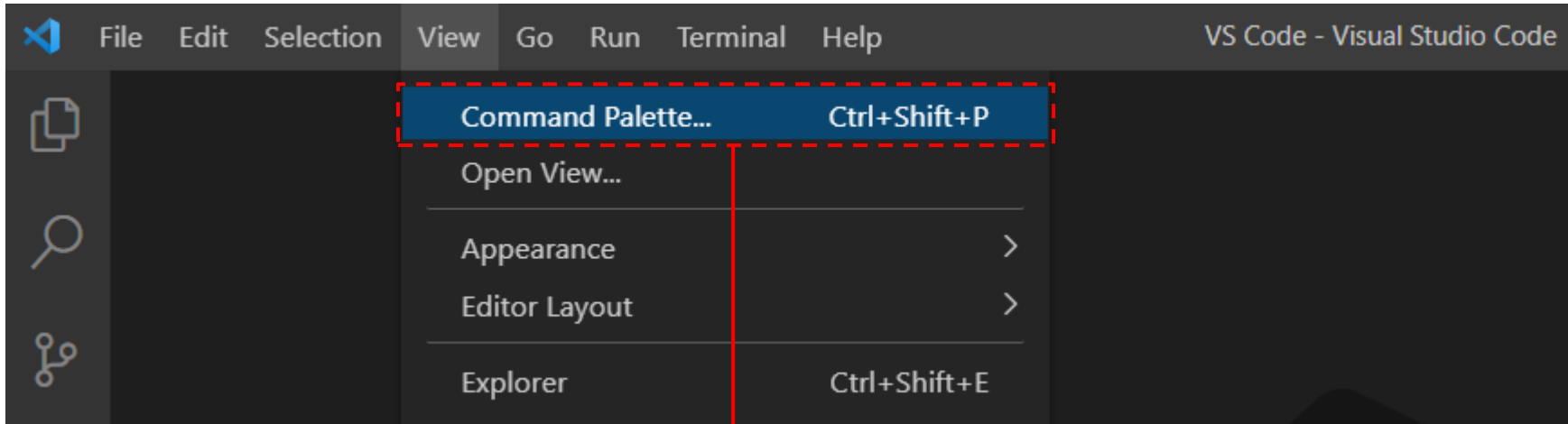
- **Debugging**

Terminal 도구를 이용한 간소화 된 디버깅 지원

## 02. 주요 기능

### • Command Palette

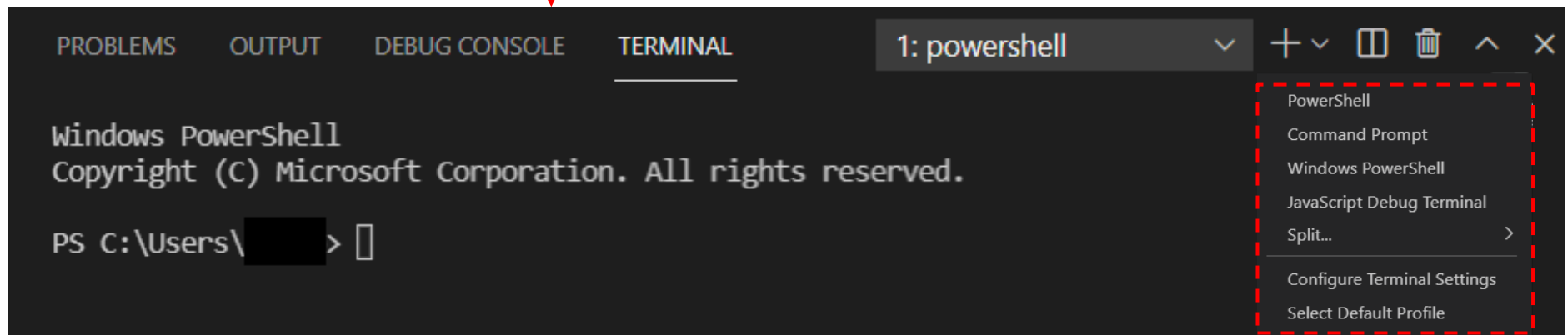
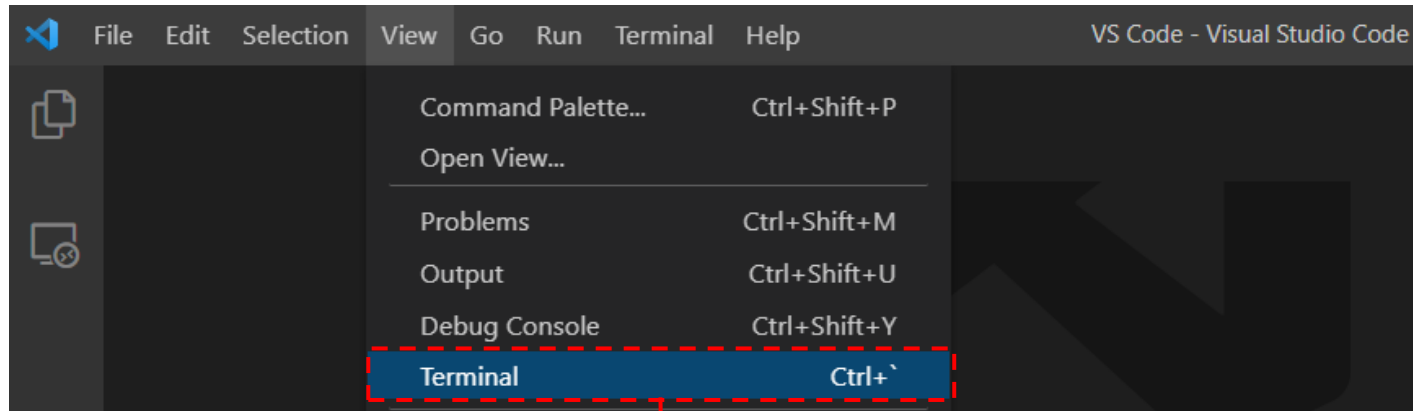
- Visual Studio Code의 모든 기능을 키보드 입력으로 쉽게 찾아 실행 가능
- 단축키 : Ctrl + Shift + P, F1



## 02. 주요 기능

- **Integrated Terminal**

- Powershell, Command Prompt, Bash 등 통합적으로 Shell 이용 가능
- 단축키 : Ctrl + `



## 02. 주요 기능

- **Intellisense**

- parameter 정보, 변수, method 등을 포함한 다양한 코드 자동완성 기능

```
4  var server = express();
5  server.use(bodyParser.json);
6
7  server.g
8      get (property) Application.get: ((name: string)... ⓘ
9      getMaxListeners
10     arguments
11     engine
12     length
13     merge
14     purge
15     settings
16     toString
17     defaultConfiguration
18
```

## 02. 주요 기능

- Extensions

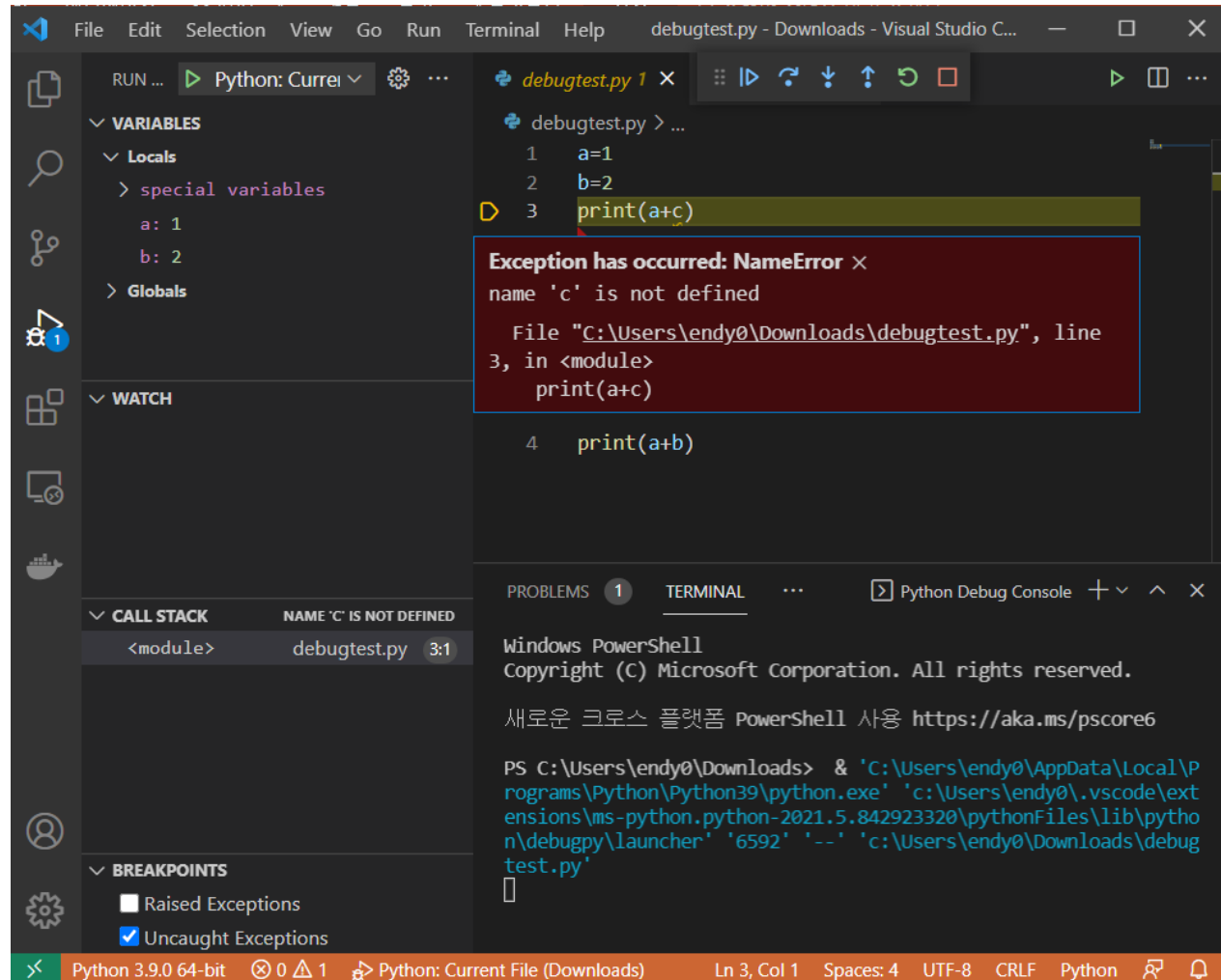
- 확장 프로그램 설치를 통해 언어, 디버거 및 도구를 추가하여 Work Flow 지원 가능



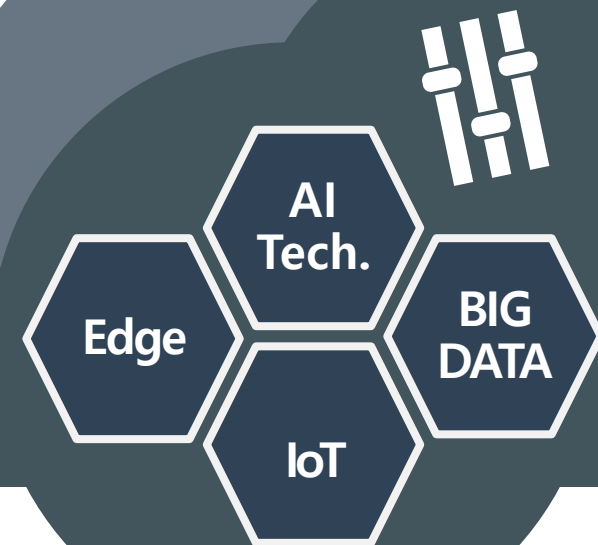
## 02. 주요 기능

- Debugging

- Terminal 도구를 이용한 간소화 된 디버깅 지원



# 목차



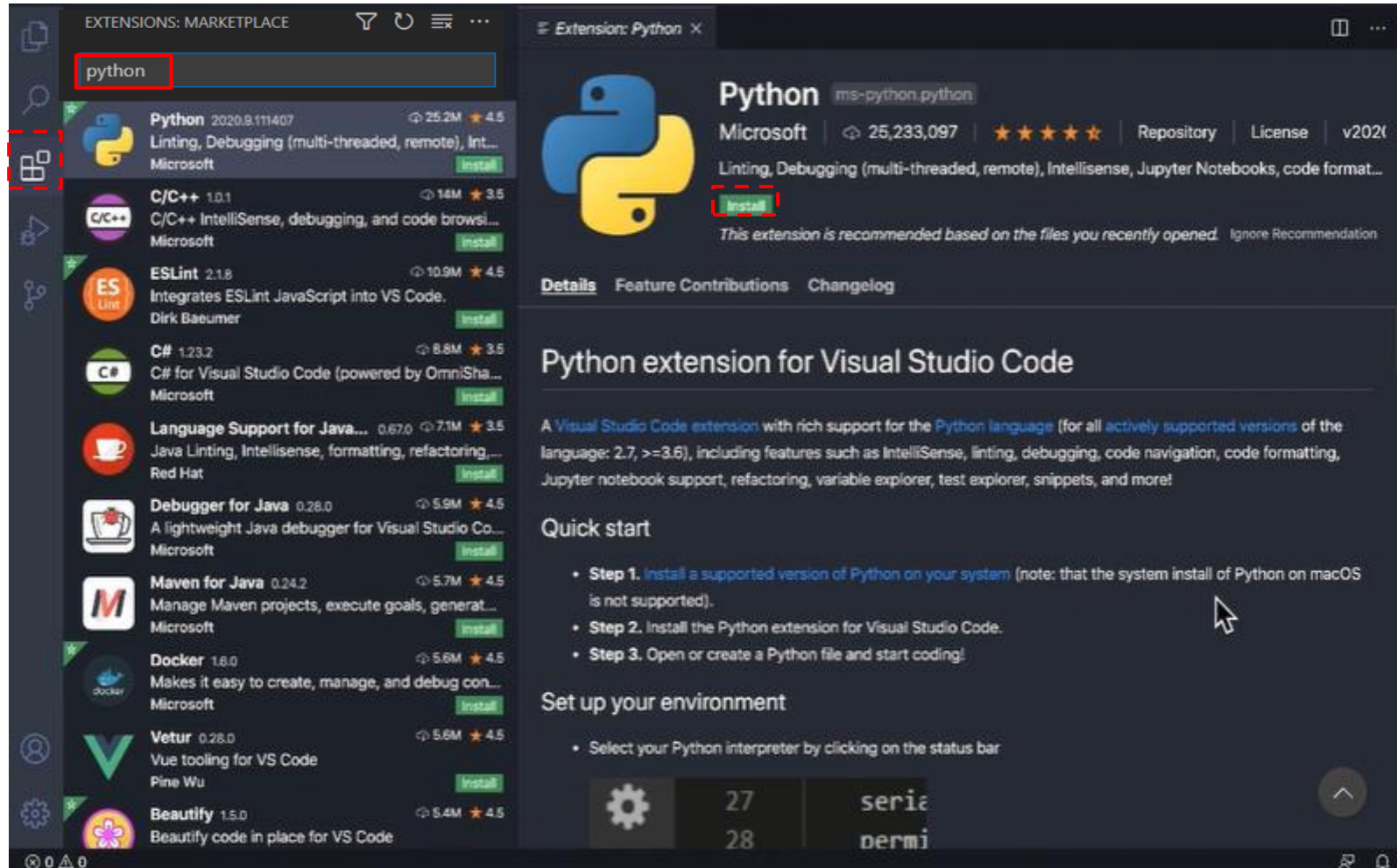
- 1 Visual Studio Code
- 2 주요 기능
- 3 Python 활용
- 4 Jupyter Notebook 활용
- 5 Remote 서버(VM) 활용
- 6 Docker 활용



# 03. Python 활용

- 확장 프로그램 설치

- Extensions – Python 검색 – Install

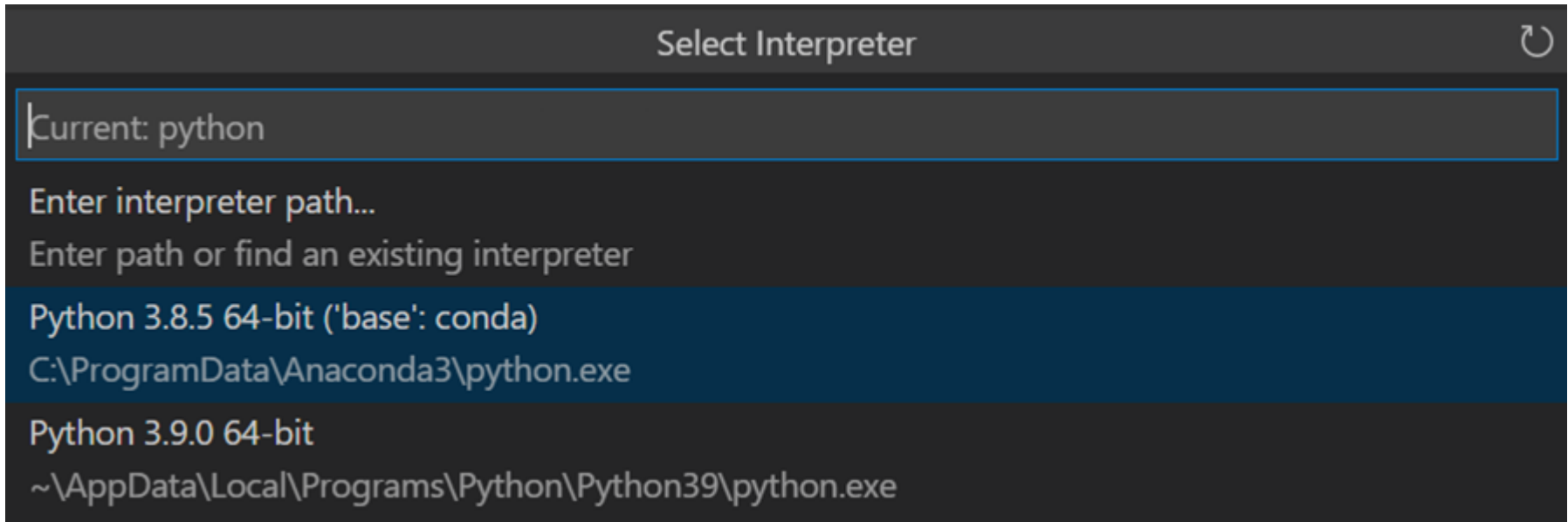
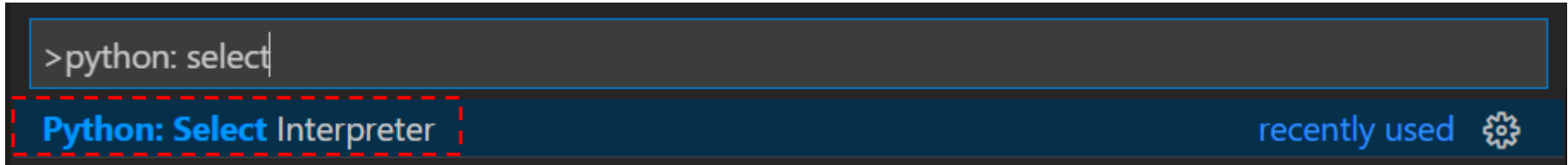




## 03. Python 활용

- Python 파일(.py) 생성

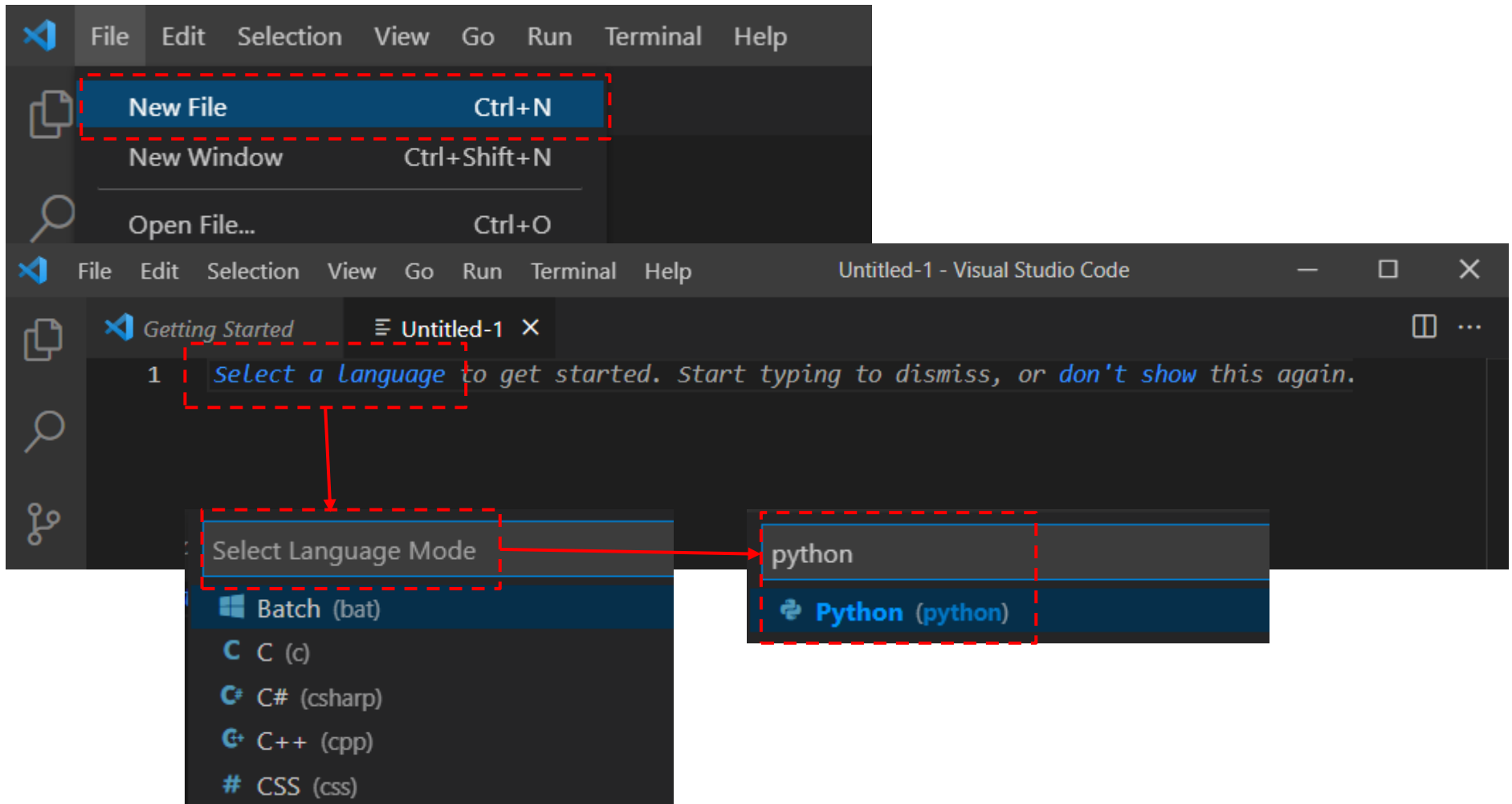
1. Command Palette에서 Python: Select Interpreter 클릭 – Interpreter 선택 – Status bar에 표시됨



# 03. Python 활용

- Python 파일(.py) 생성

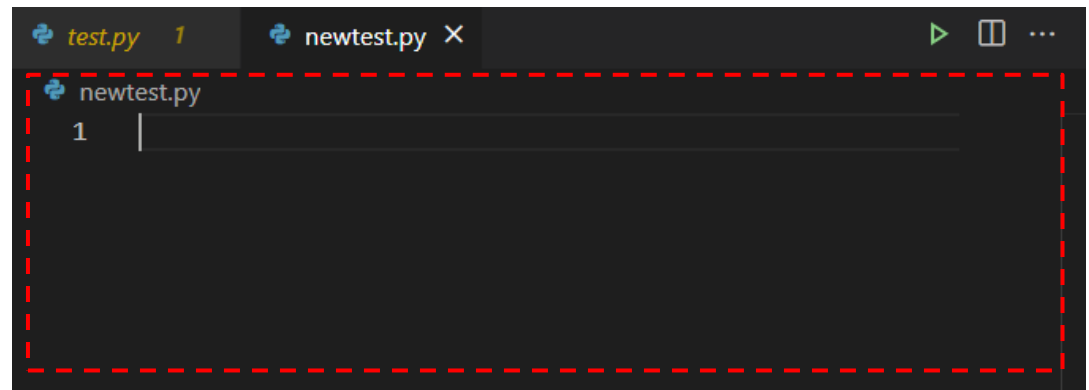
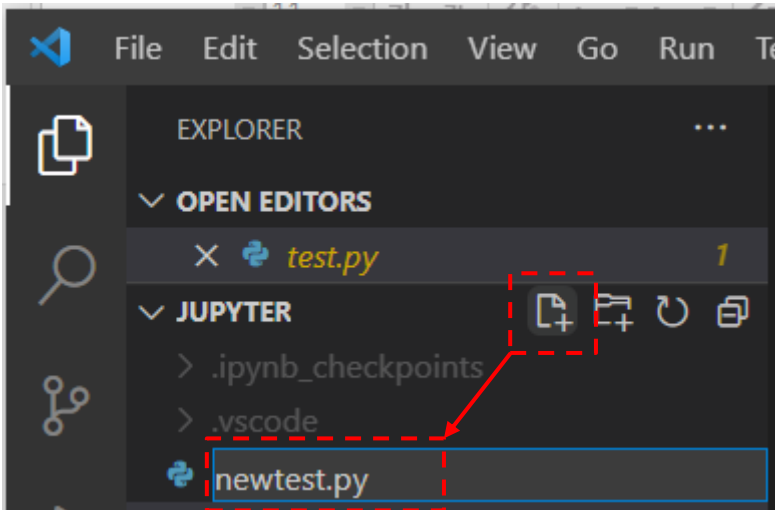
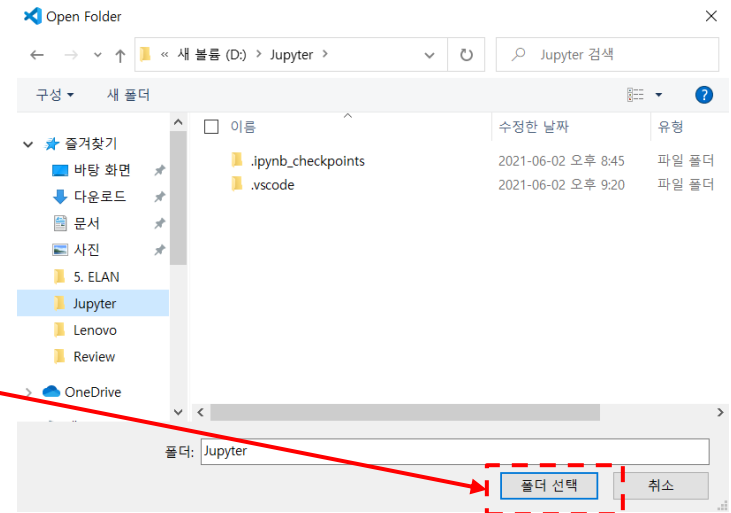
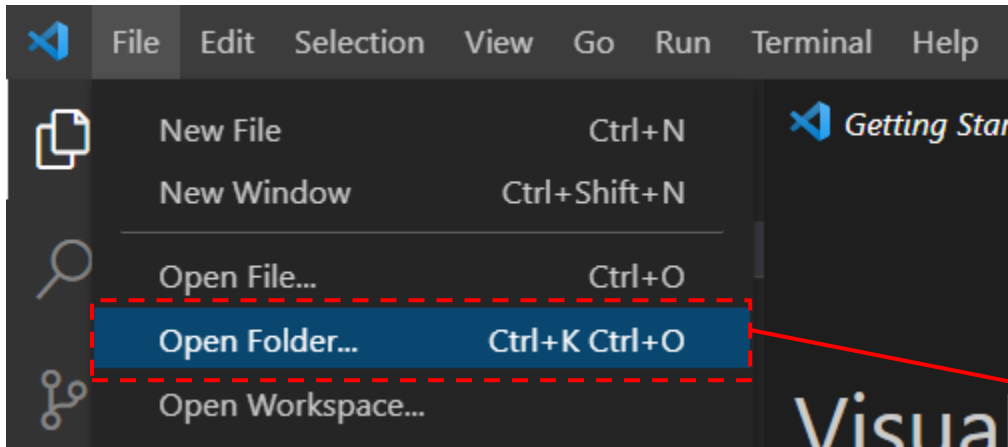
2-1. File – New file – Select a Language – Python 입력 후 선택



# 03. Python 활용

- Python 파일(.py) 생성

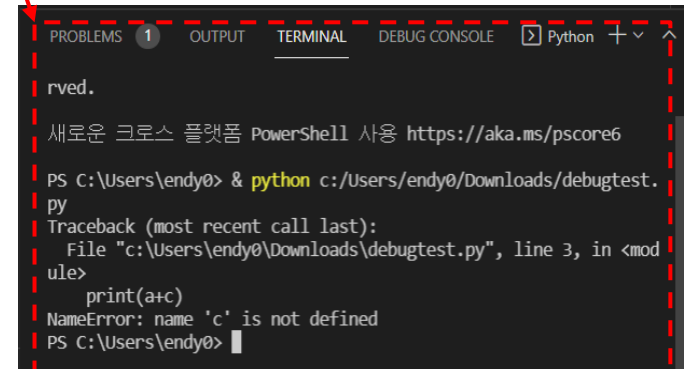
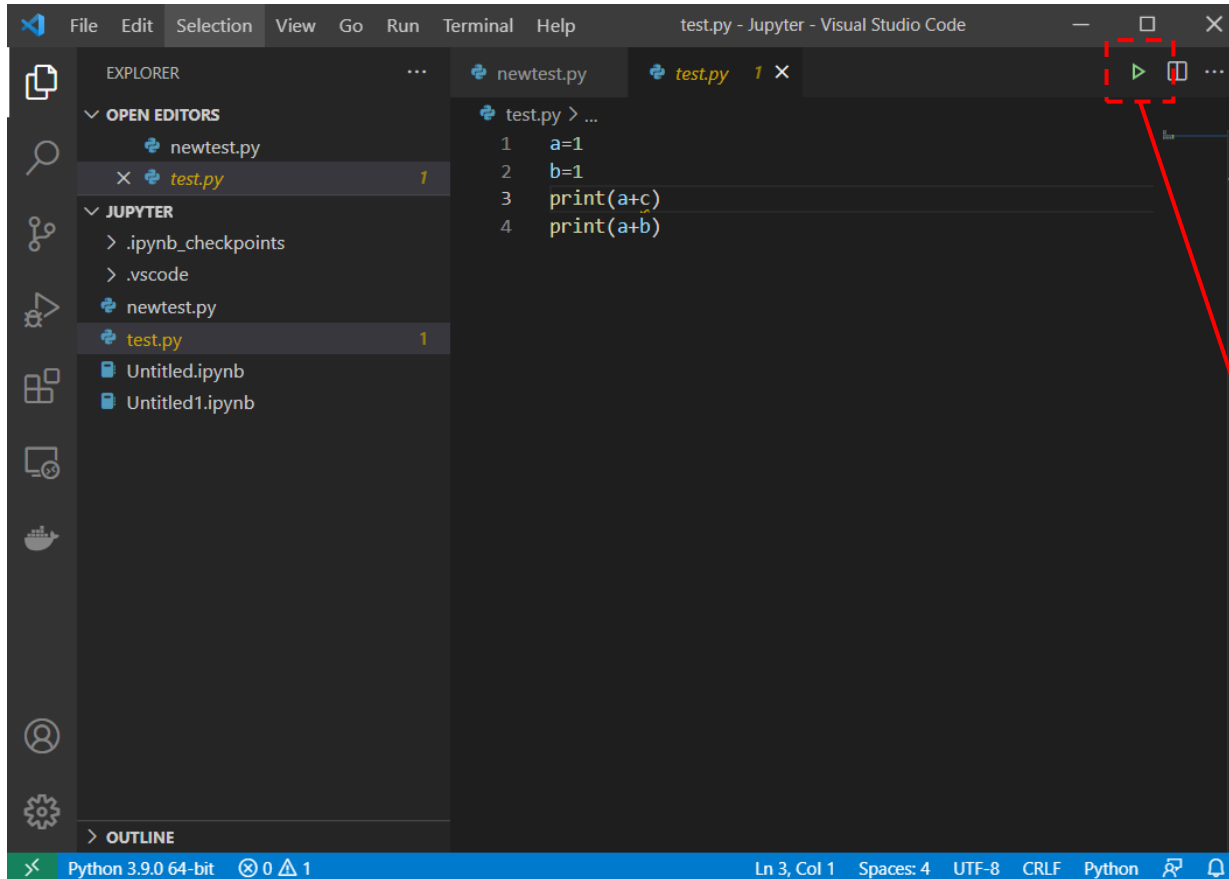
2-2. File – Open Folder – 폴더 선택 – New File 아이콘 클릭 – 파일명 및 확장자(.py) 지정



# 03. Python 활용

- Python 파일(.py) 실행

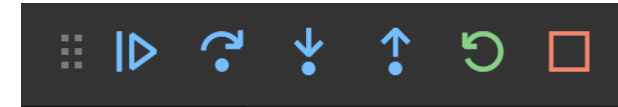
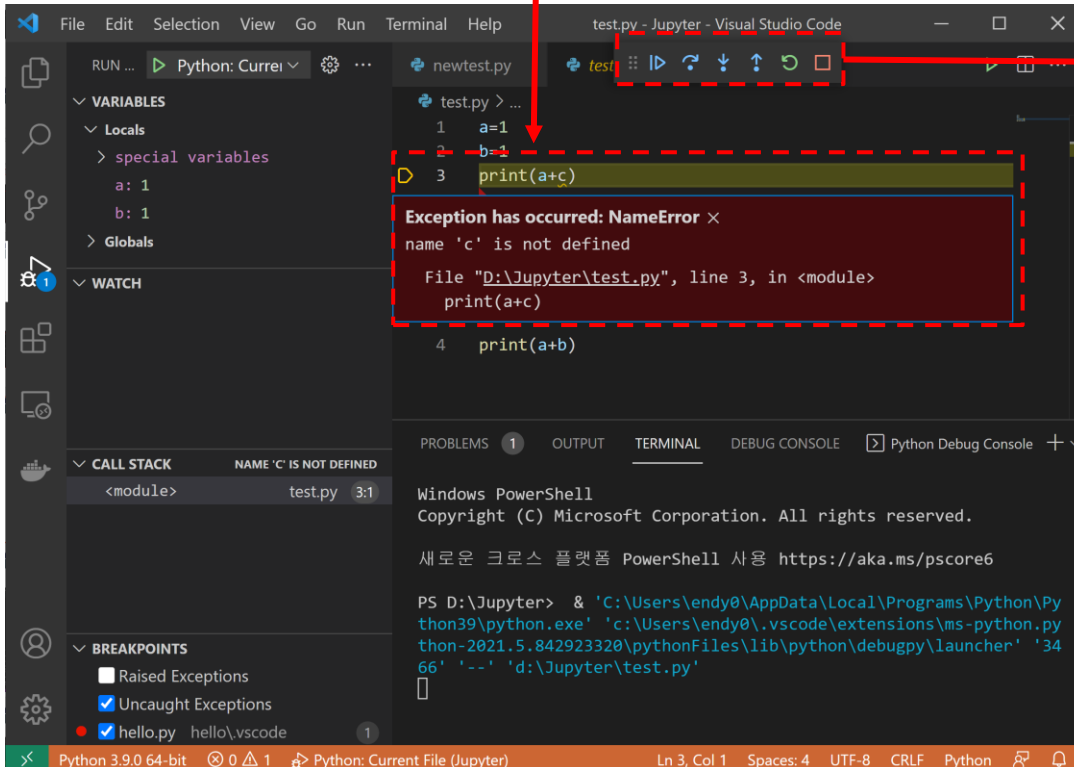
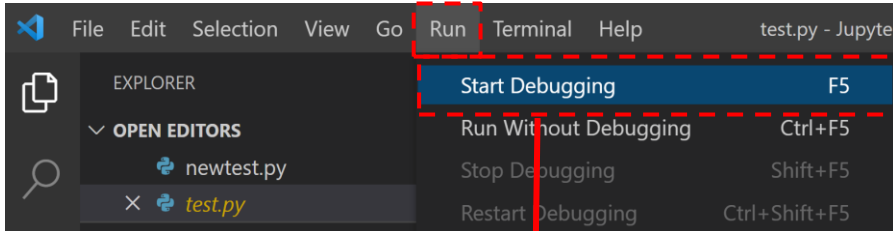
- Run 아이콘 클릭 – Terminal 창에 실행됨



# 03. Python 활용

## • Python 파일(.py) 디버깅

### 1. Run – Start Debugging – 디버깅 결과 확인



- Continue (F5) : 다음 breakpoint로 이동
- Step Over (F10) : 다음 line으로 이동 (함수 내부x)
- Step Into (F11) : 다음 line으로 이동 (함수 내부o)
- Step Out (Shift + F11)  
: 현재 함수의 나머지 부분 실행 후 리턴 완료에서 멈춤
- Restart (Ctrl + Shift + F5) : 디버깅 재시작
- Stop (Shift + F5) : 디버깅 중지

# 03. Python 활용

- Python 파일(.py) 디버깅

2. Breakpoint 지정 : breakpoint 지정할 line에서 F9키 또는 line number의 좌측 클릭 – 디버깅 실행 시 Side bar에서 확인 가능

The image consists of two side-by-side screenshots of the Visual Studio Code interface, demonstrating Python debugging.

**Left Screenshot:** Shows the editor with a file named `test.py` containing the following code:

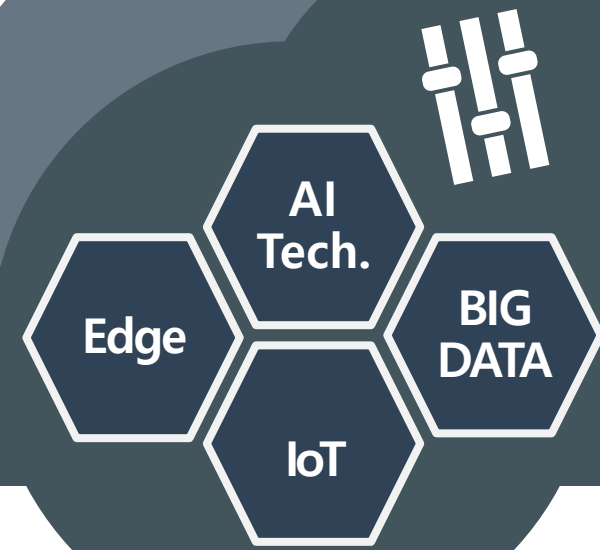
```
1 a=1
2 b=1
3 print(a+b)
4 print(a+c)
```

A red dashed box highlights a breakpoint (a red dot) set on line 3. The left sidebar shows the **VARIABLES** panel with `test.py > ...` and `1 a=1`, `2 b=1`.

**Right Screenshot:** Shows the same code with the debugger running. The code is highlighted in green, indicating it is being executed. A red dashed box highlights the breakpoint on line 3. The left sidebar shows the **VARIABLES** panel with `test.py > ...` and `1 a=1`, `2 b=1`. The **WATCH** panel is empty. The **CALL STACK** panel shows `<module>` and `test.py 3:1`. The **BREAKPOINTS** panel shows a breakpoint set on `hello.py` at `hello\vscode` with `1` next to it.

The bottom status bar shows `Python 3.9.0 64-bit`, `Python: Current File (Jupyter)`, `Ln 3, Col 1`, `Spaces: 4`, `UTF-8`, `CRLF`, `Python`.

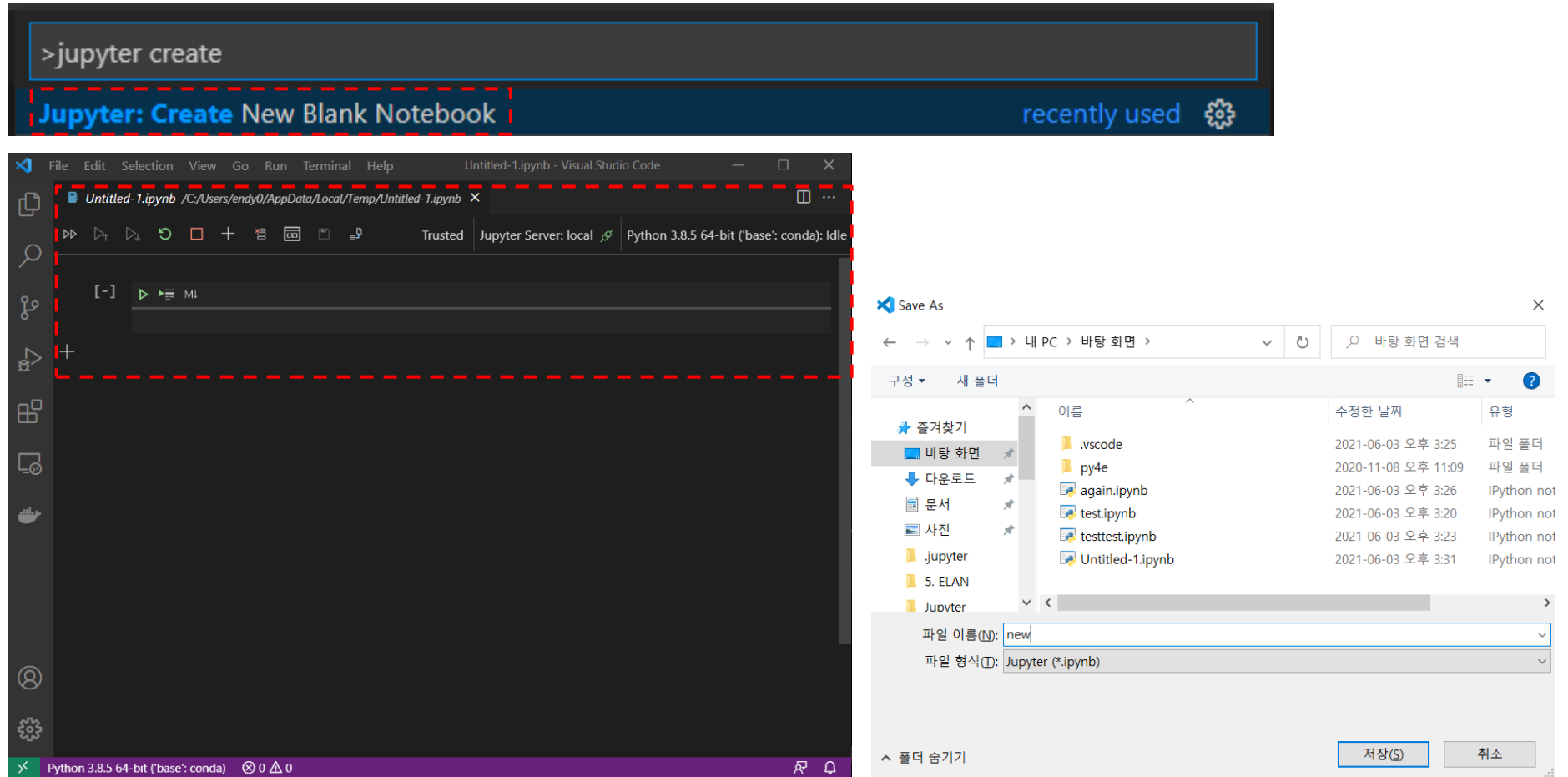
# 목차



- 1 Visual Studio Code
- 2 주요 기능
- 3 Python 활용
- 4 Jupyter Notebook 활용
- 5 Remote 서버(VM) 활용
- 6 Docker 활용

# 04. Jupyter Notebook 활용

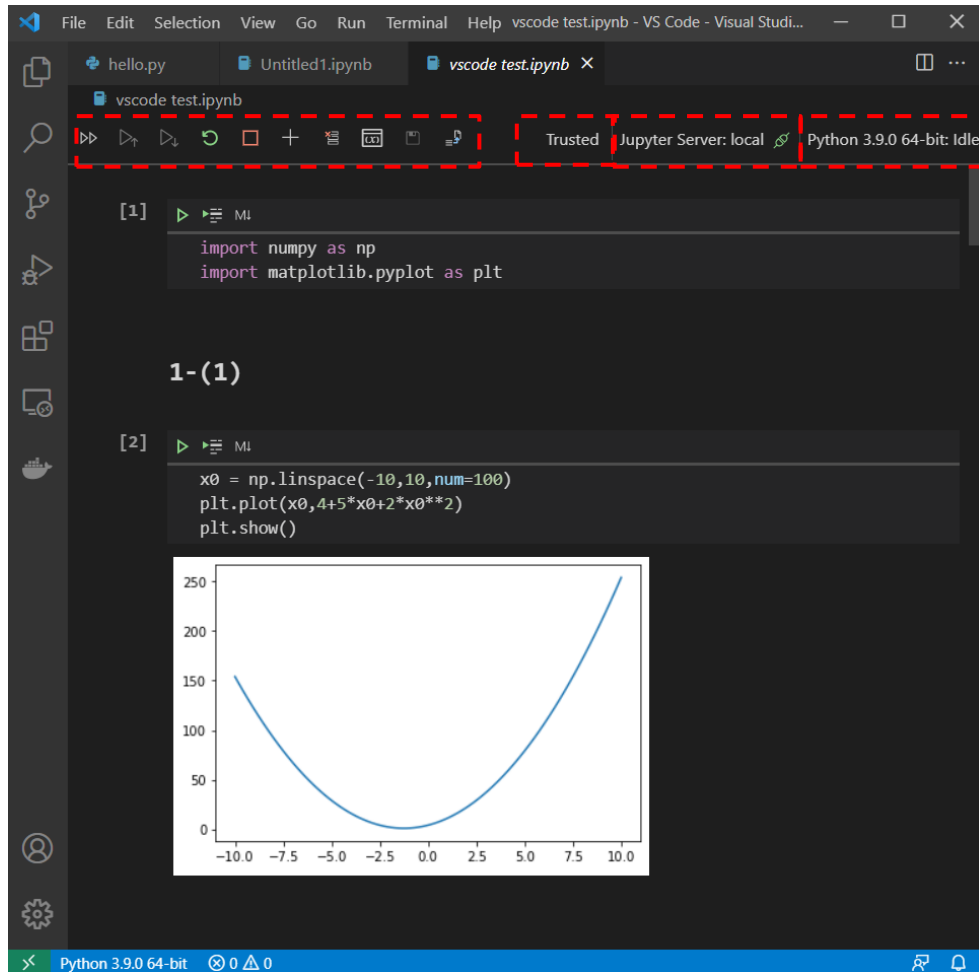
- Python Extension 설치 시 Jupyter Notebook도 함께 자동 install 됨
- Jupyter Notebook 파일(.ipynb) 생성
  - Command Palette에서 Jupyter: Create New Blank Notebook 클릭 – 파일 생성 – 작업 후 저장 시 경로 및 파일명 변경





# 04. Jupyter Notebook 활용

- Jupyter Notebook Editor 화면



- Run all cells : 전체 셀 실행
- Run cells above : 위 셀 실행
- Run cell and below : 아래 셀 실행
- Restart Jupyter kernel : 커널 재시작
- Interrupt Jupyter kernel : 커널 일시정지
- Insert cell : 셀 삽입
- Clear All Output : 모든 출력 삭제
- Show variables active in jupyter kernel : 변수 탐색기
- Save notebook : 저장하기
- Export As : 내보내기 (.py, pdf, html 가능)

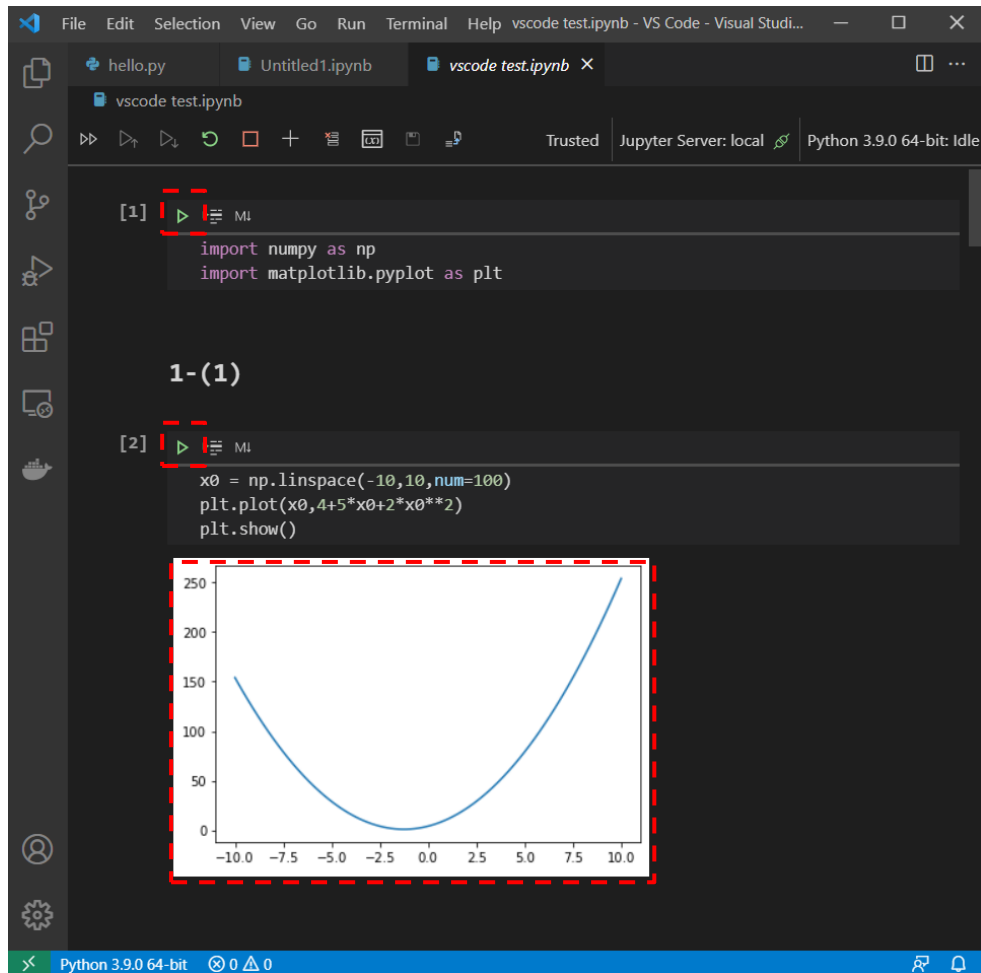


- Not Trusted / Trusted
- Jupyter Server 상태
- Jupyter Kernel 상태

# 04. Jupyter Notebook 활용

- Jupyter Notebook 실행

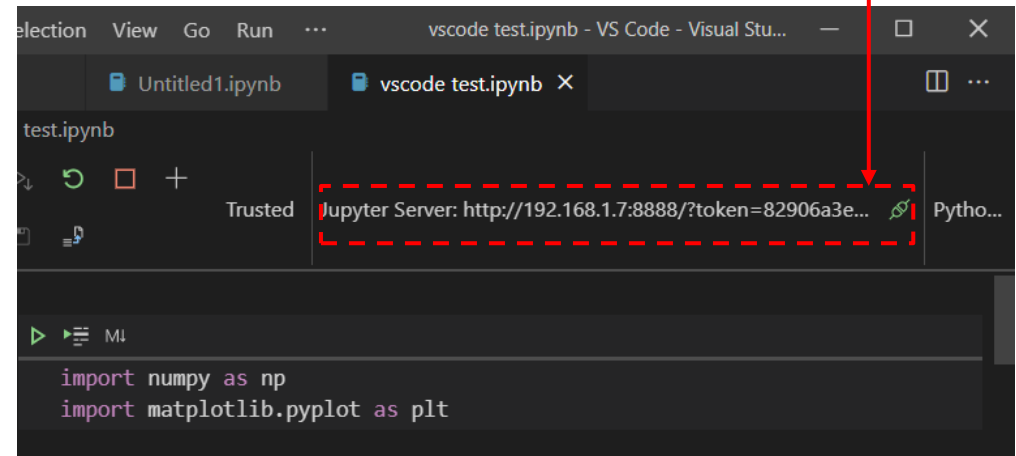
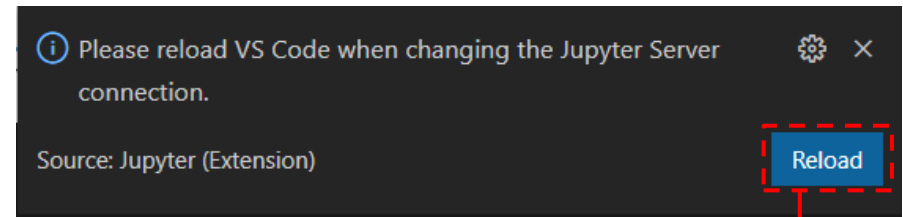
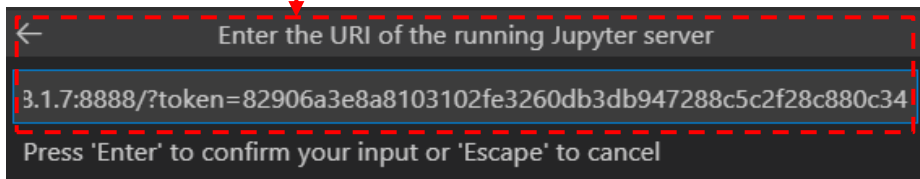
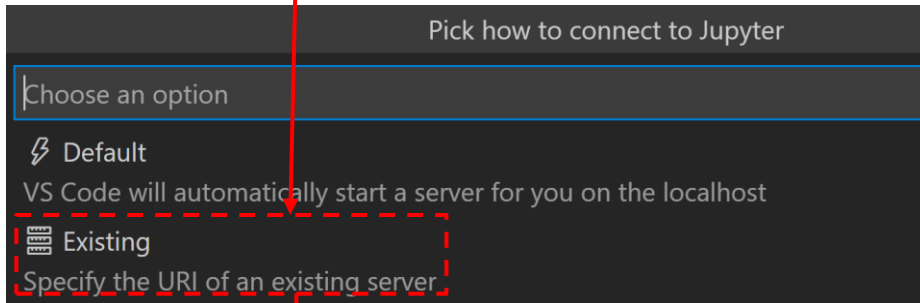
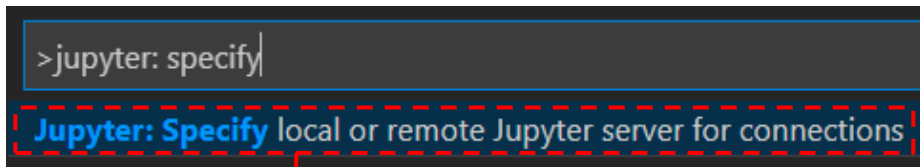
- Run 아이콘 클릭 – 코드 cell 실행 – 출력은 cell 바로 아래에 나타남



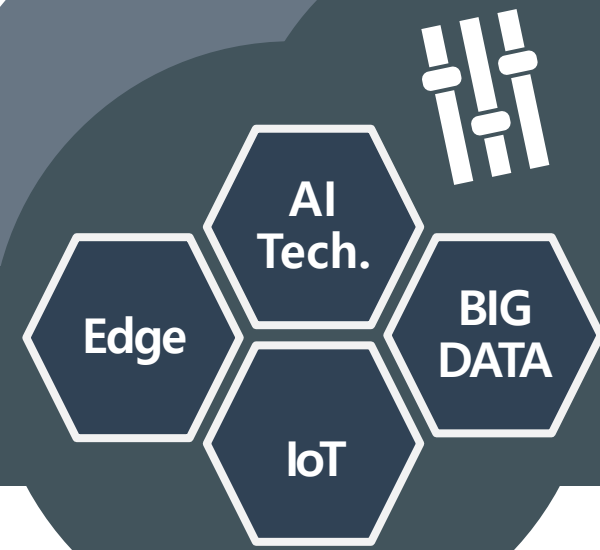
# 04. Jupyter Notebook 활용

## • Remote Server 연결

- Command Palette에서 jupyter: specify local or remote Jupyter server for connections 클릭
  - Existing 선택 (Default는 Local) – Jupyter Server URI 입력 – Reload VS Code – Server 상태 변경 확인
- 연결되면 코드 cell은 local이 아닌 remote server에서 실행됨 (파일 공유는 안됨)



# 목차

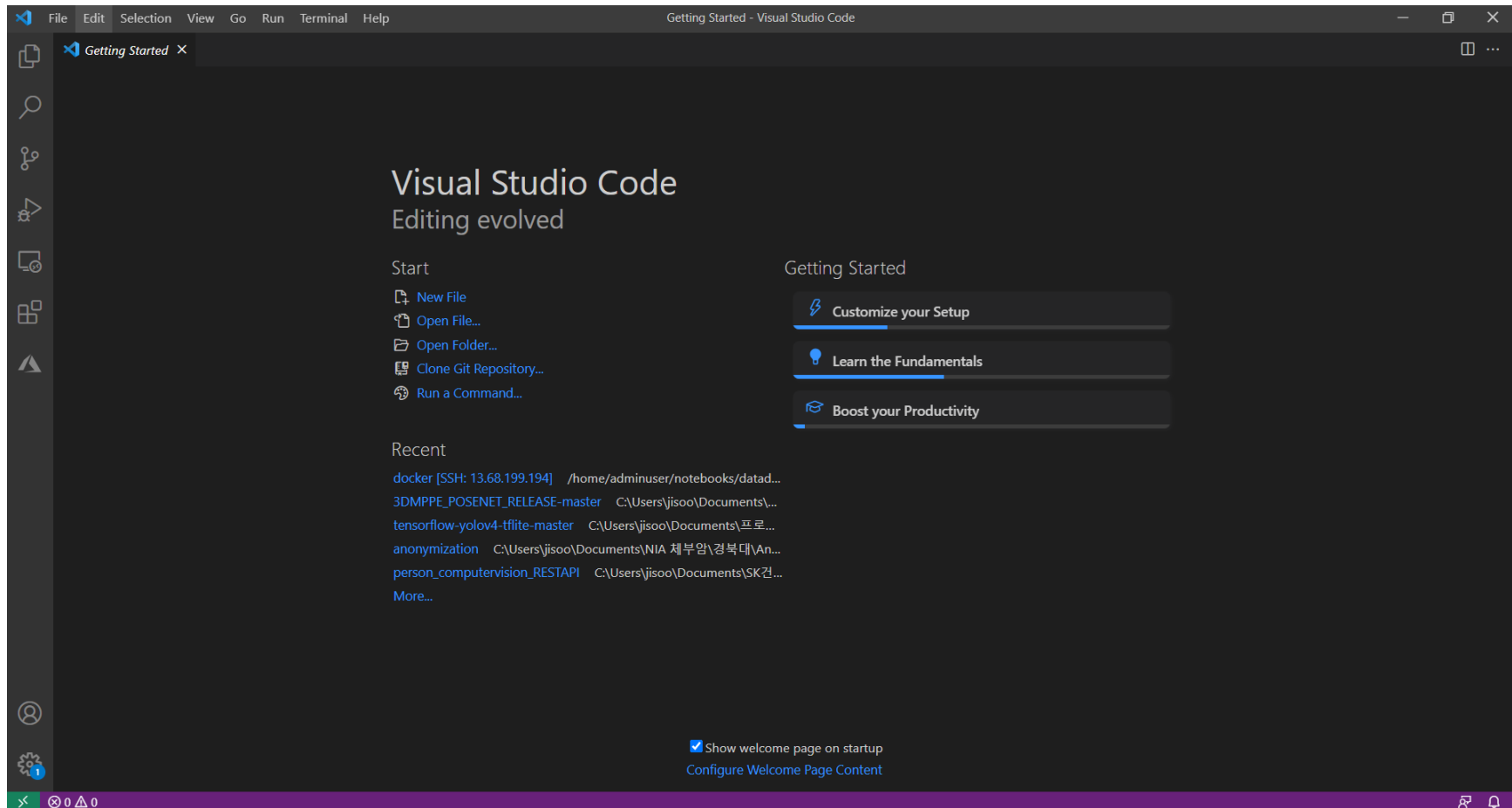


- 1 Visual Studio Code
- 2 주요 기능
- 3 Python 활용
- 4 Jupyter Notebook 활용
- 5 Remote 서버(VM) 활용
- 6 Docker 활용

# 05. Remote 서버(VM) 활용

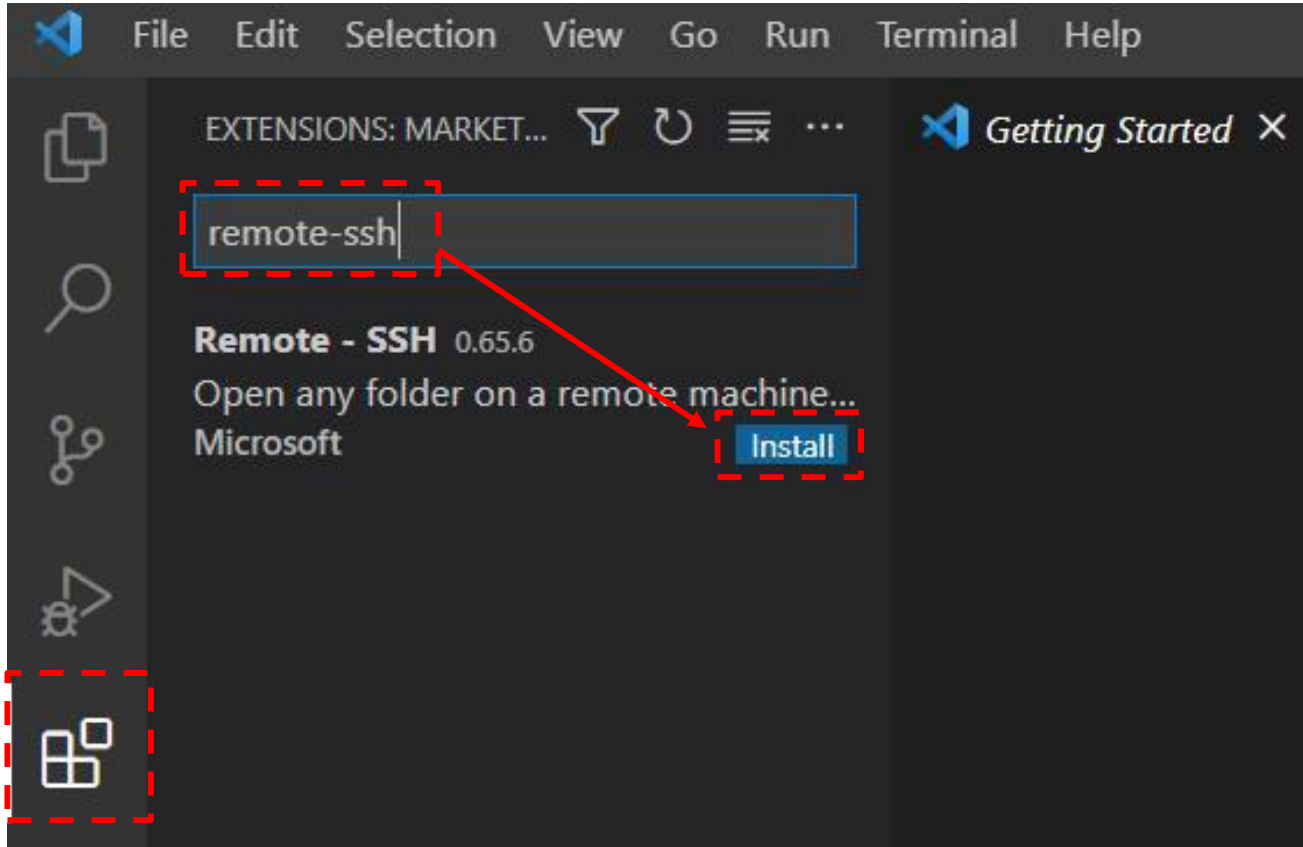
- Visual Studio Code에서 Linux VM에 remote 연결

## 1. Visual Studio Code 실행



## 05. Remote 서버(VM) 활용

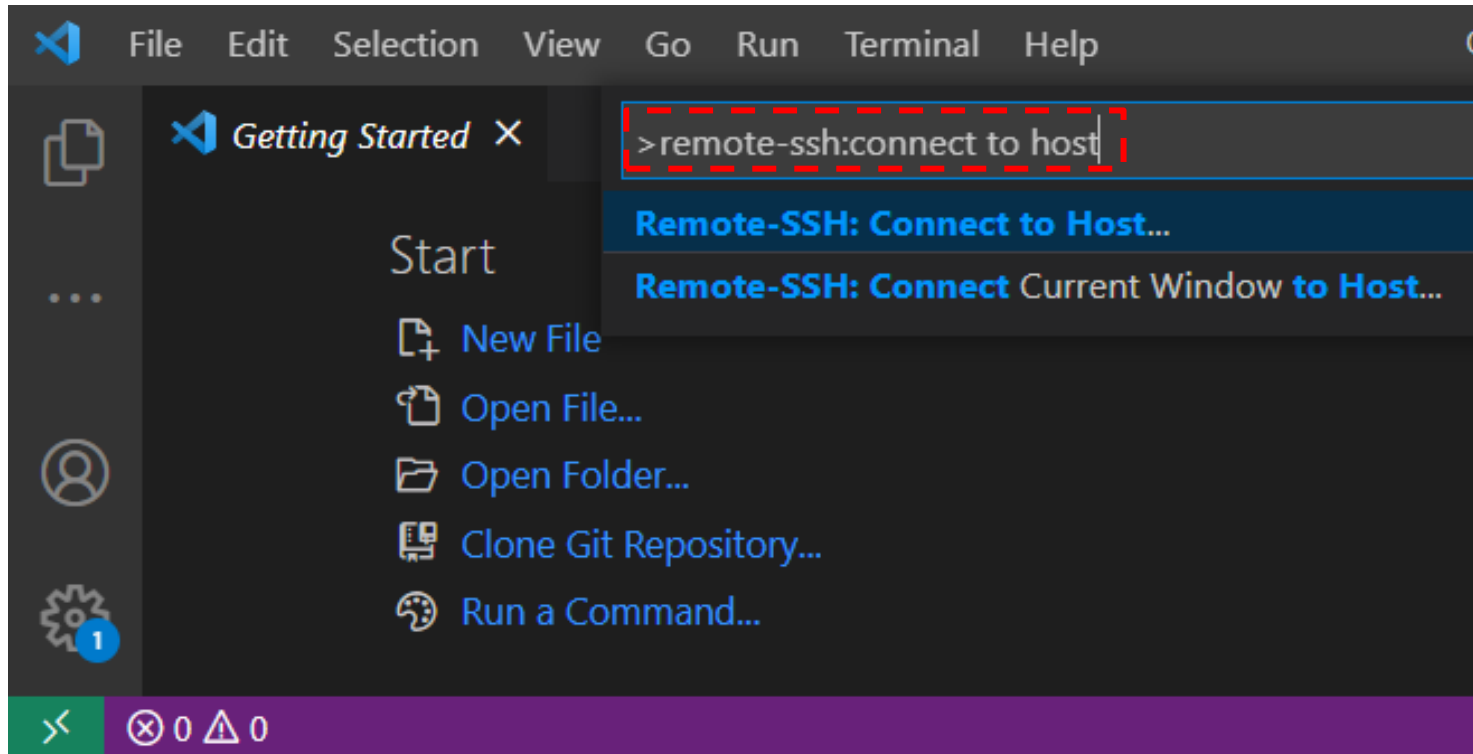
- Visual Studio Code에서 Linux VM에 remote 연결
  - 2. 좌측의 Extensions에서 'remote-ssh' 검색 후 install



## 05. Remote 서버(VM) 활용

- Visual Studio Code에서 Linux VM에 remote 연결

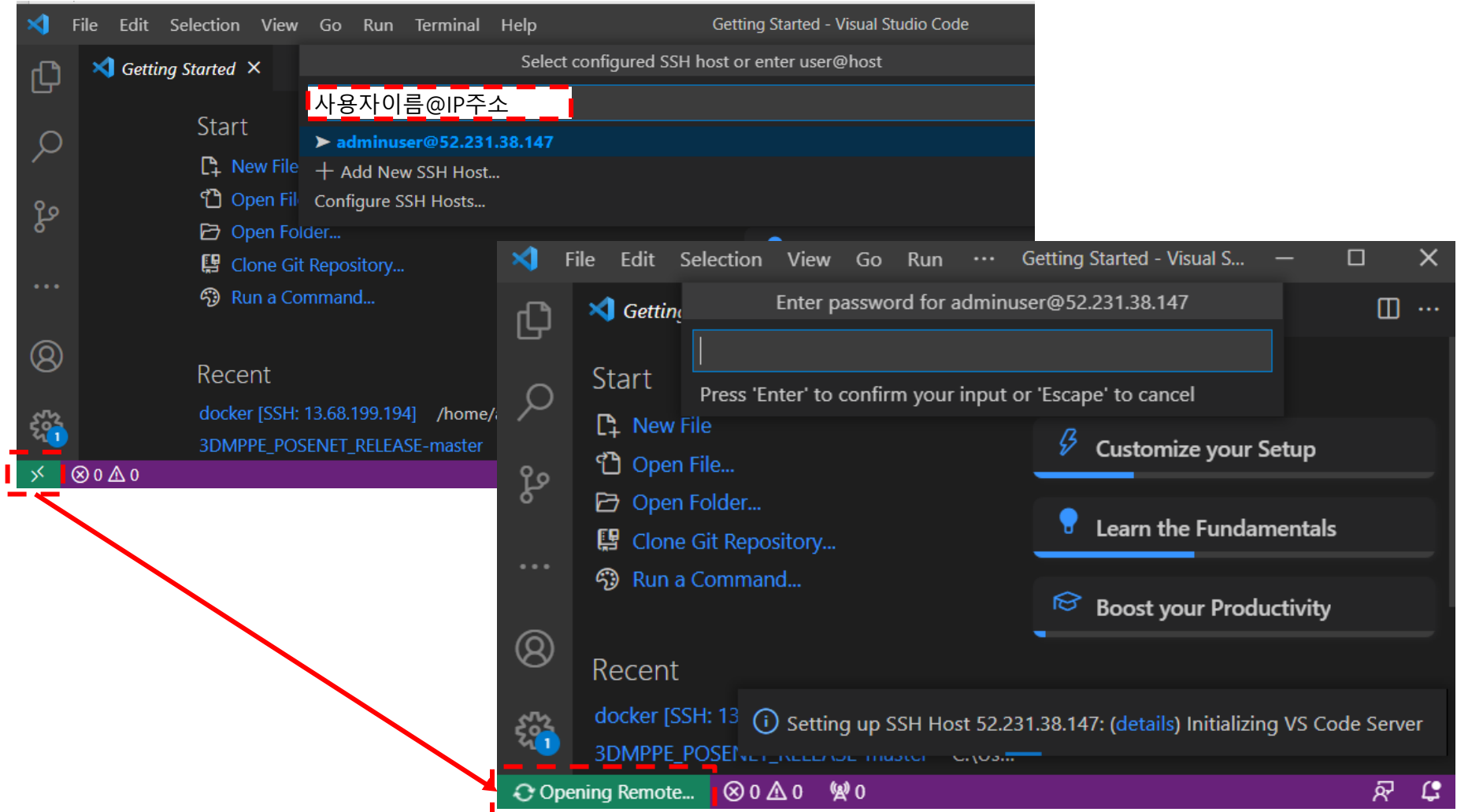
3. Command Palette에 'Remote-SSH:Connect to Host' 입력 후 Enter



## 05. Remote 서버(VM) 활용

- Visual Studio Code에서 Linux VM에 remote 연결

4. VM생성 시 입력한 '사용자이름'@'VM IP주소' 입력 후 Enter → VM에 remote 중 확인

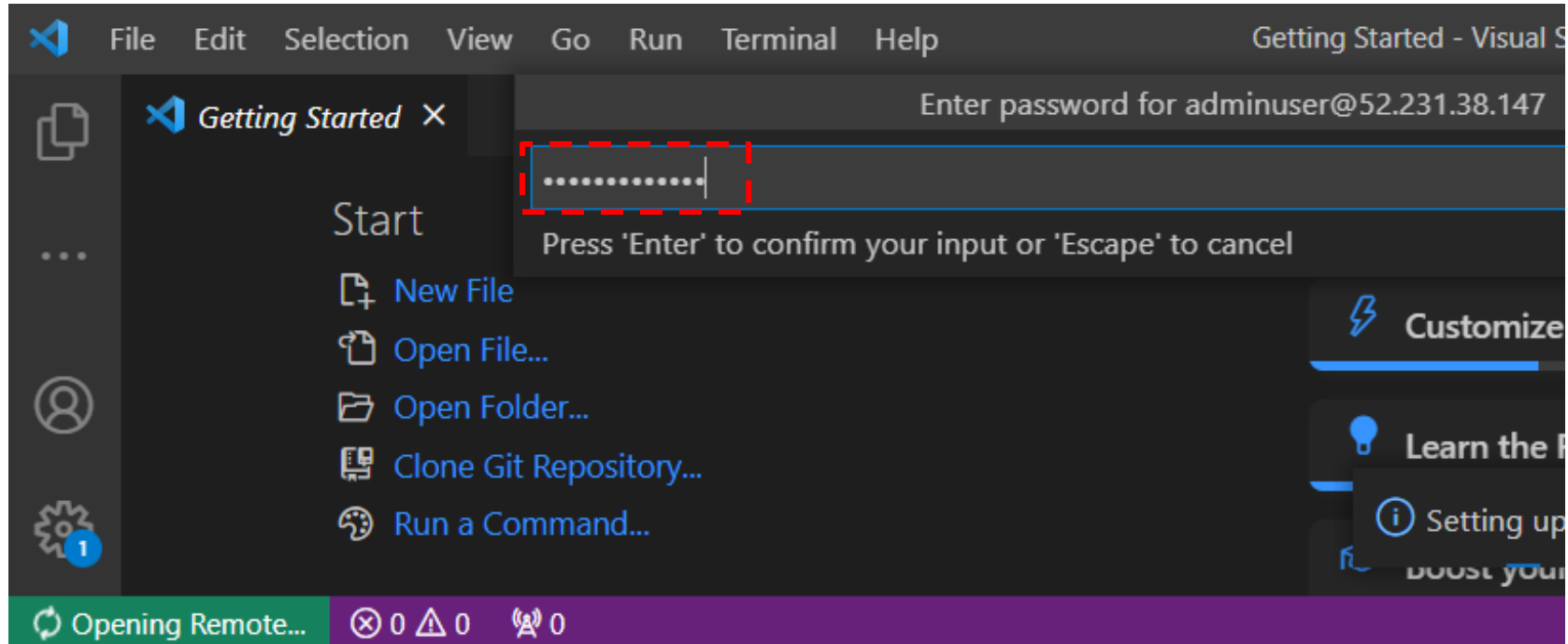




## 05. Remote 서버(VM) 활용

- Visual Studio Code에서 Linux VM에 remote 연결

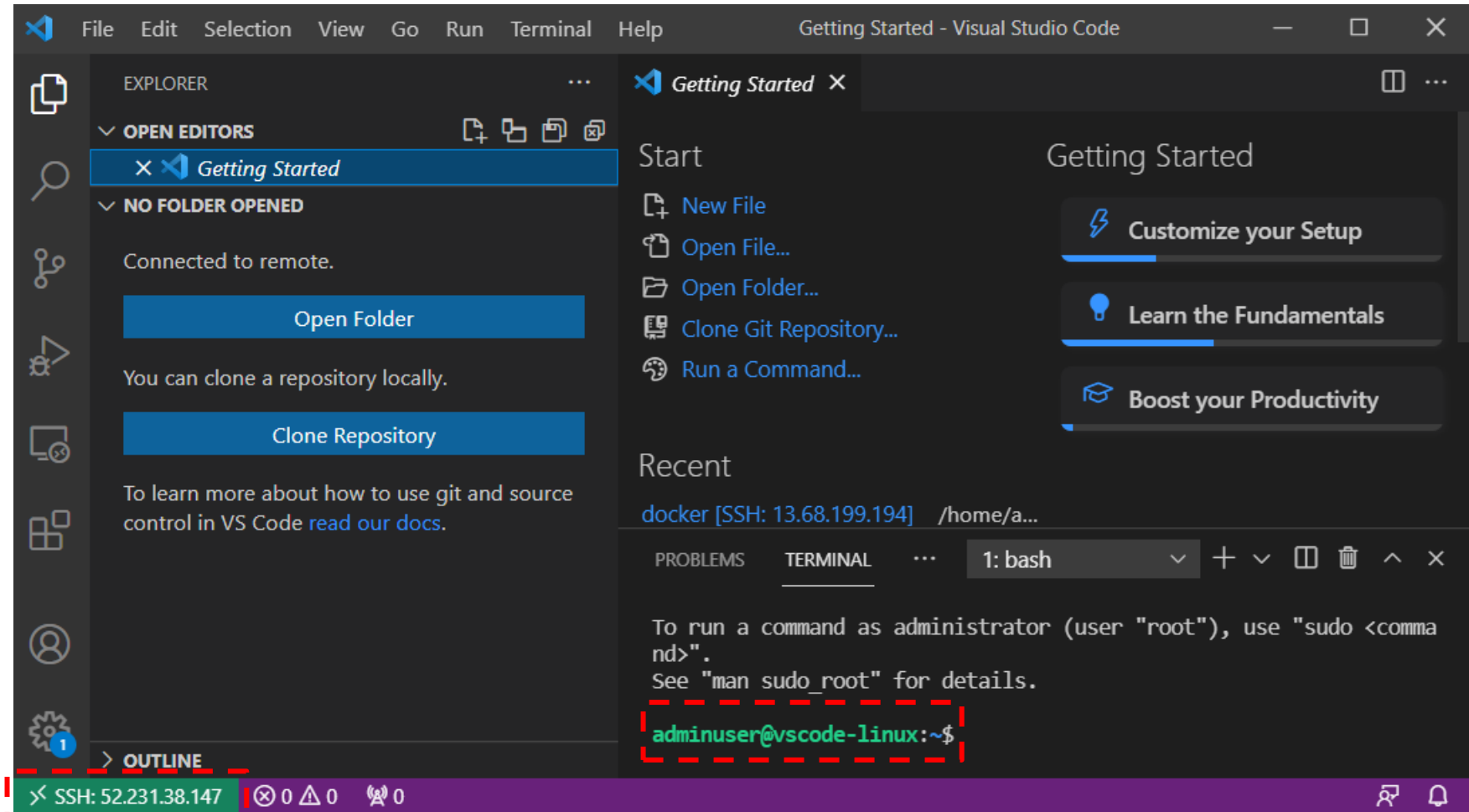
5. VM생성 시 입력한 암호 입력



# 05. Remote 서버(VM) 활용

- Visual Studio Code에서 Linux VM에 remote 연결

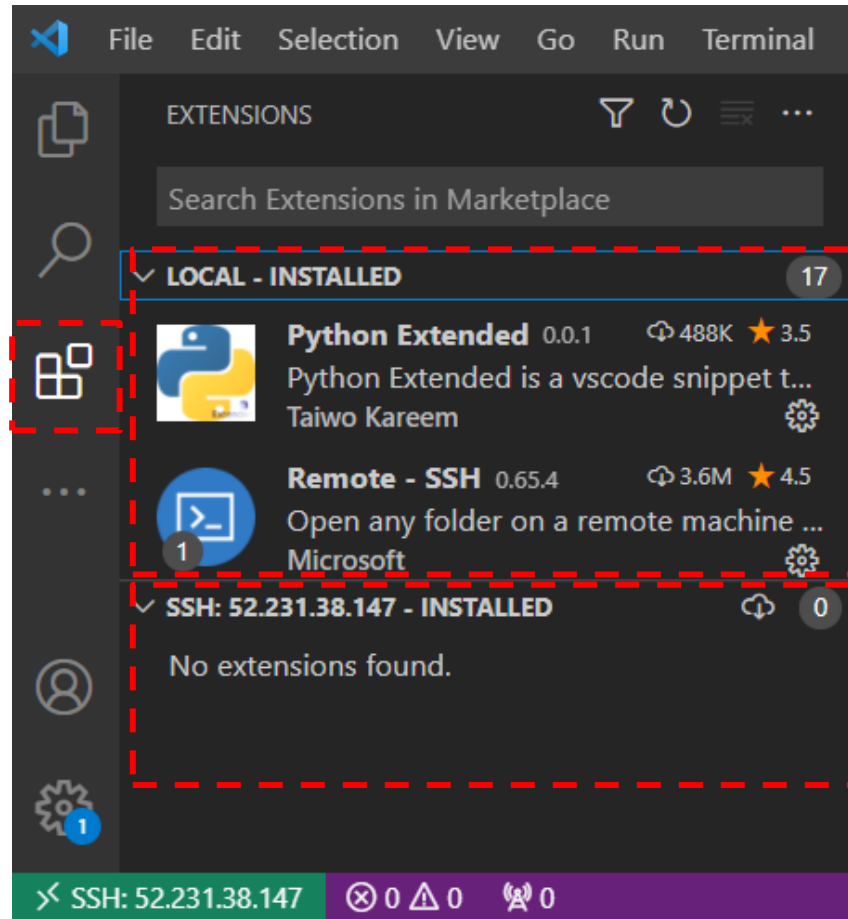
6. Visual Studio Code에서 Linux VM으로 접속 확인



## 05. Remote 서버(VM) 활용

- 연결한 Linux VM에서 Python설치 및 활용

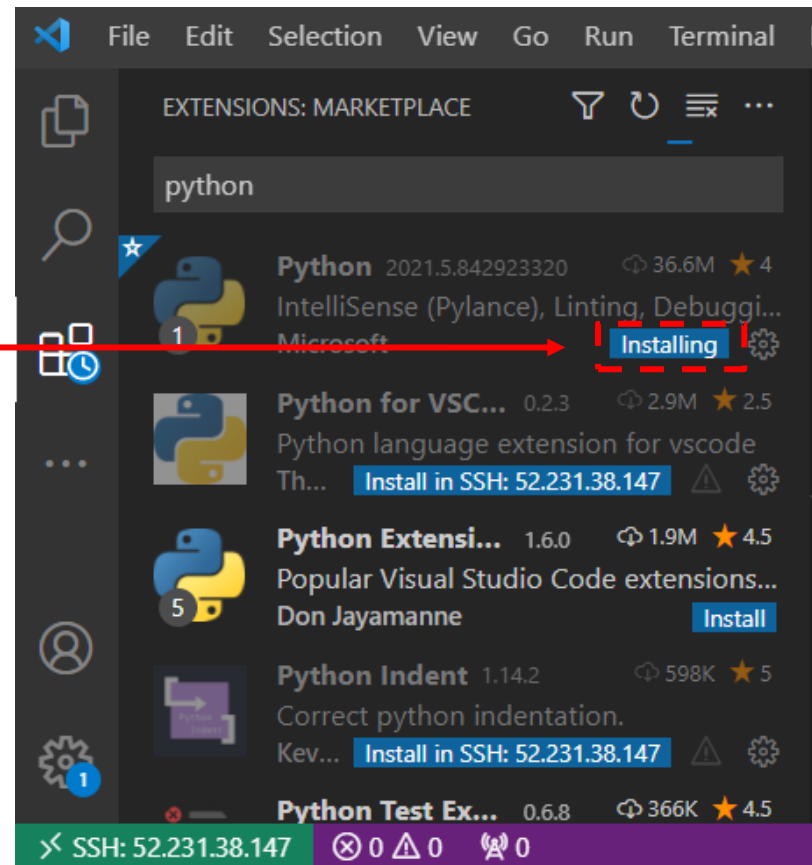
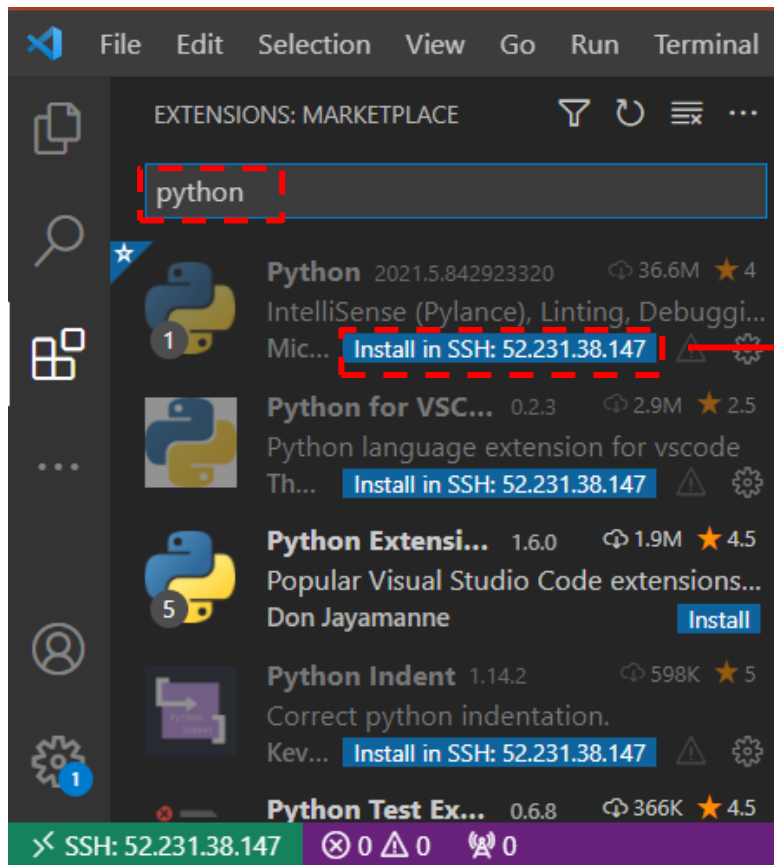
1. VM Extensions를 클릭하면 로컬에 install된 Extensions와 VM에 install된 Extensions를 볼 수 있음



## 05. Remote 서버(VM) 활용

- 연결한 Linux VM에서 Python설치 및 활용

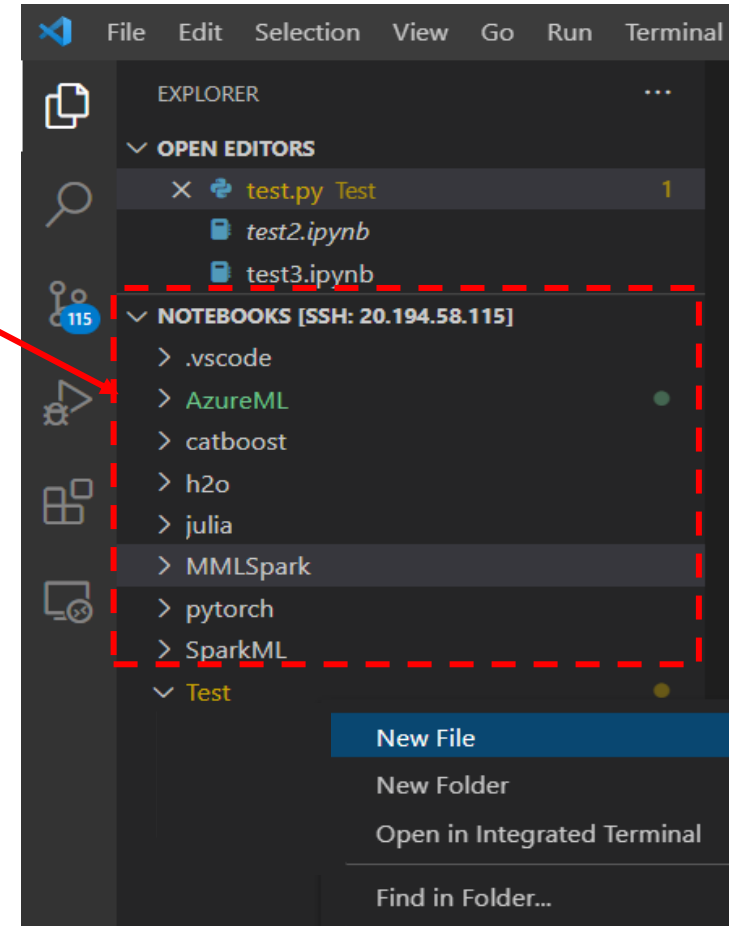
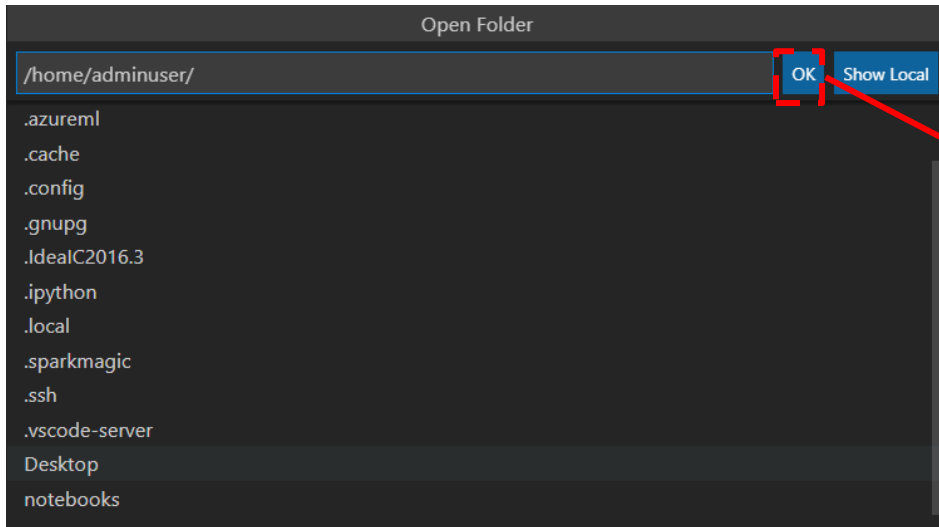
2. Extensions 검색창에 python을 검색하여 install



## 05. Remote 서버(VM) 활용

- 연결한 Linux VM에서 Python설치 및 활용

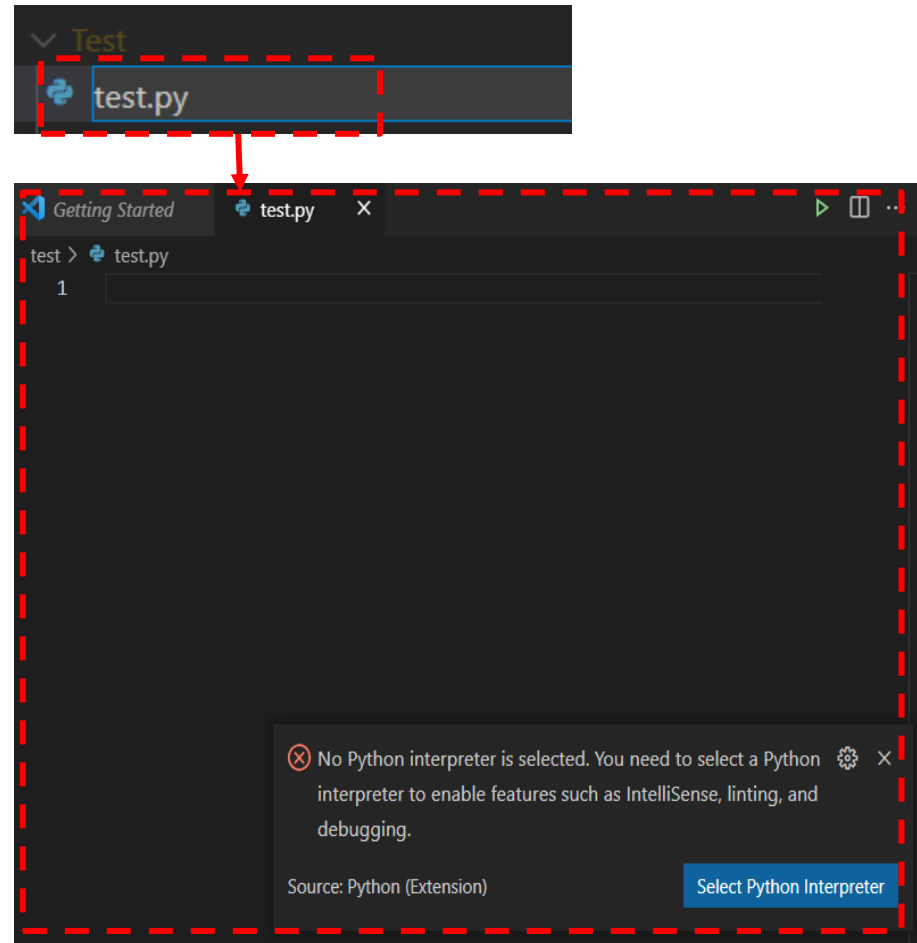
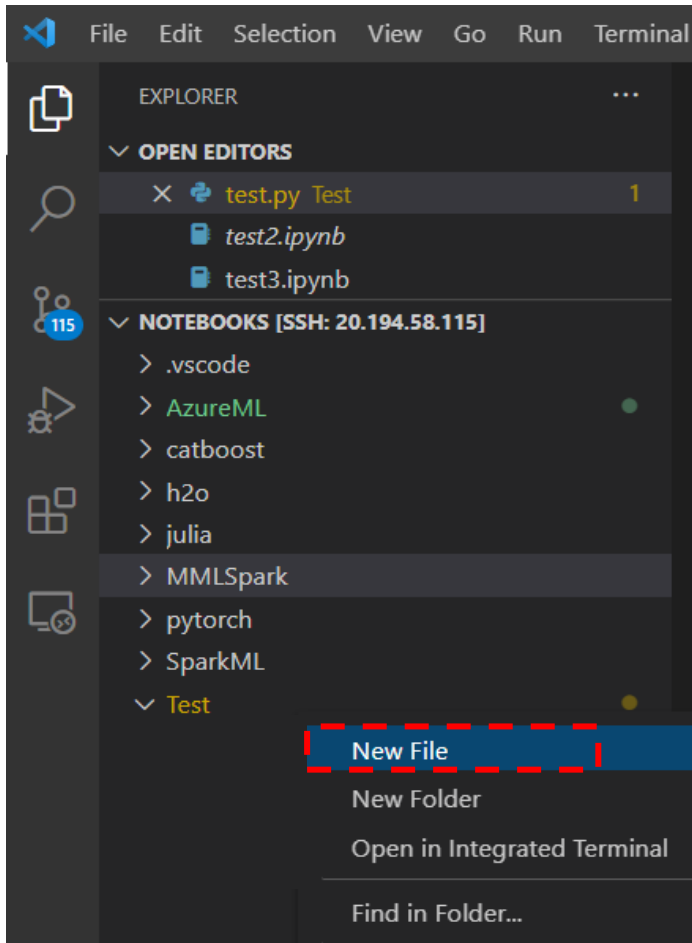
3. Open Folder – 경로 선택 - 가져온 폴더 확인



## 05. Remote 서버(VM) 활용

- 연결한 Linux VM에서 Python설치 및 활용

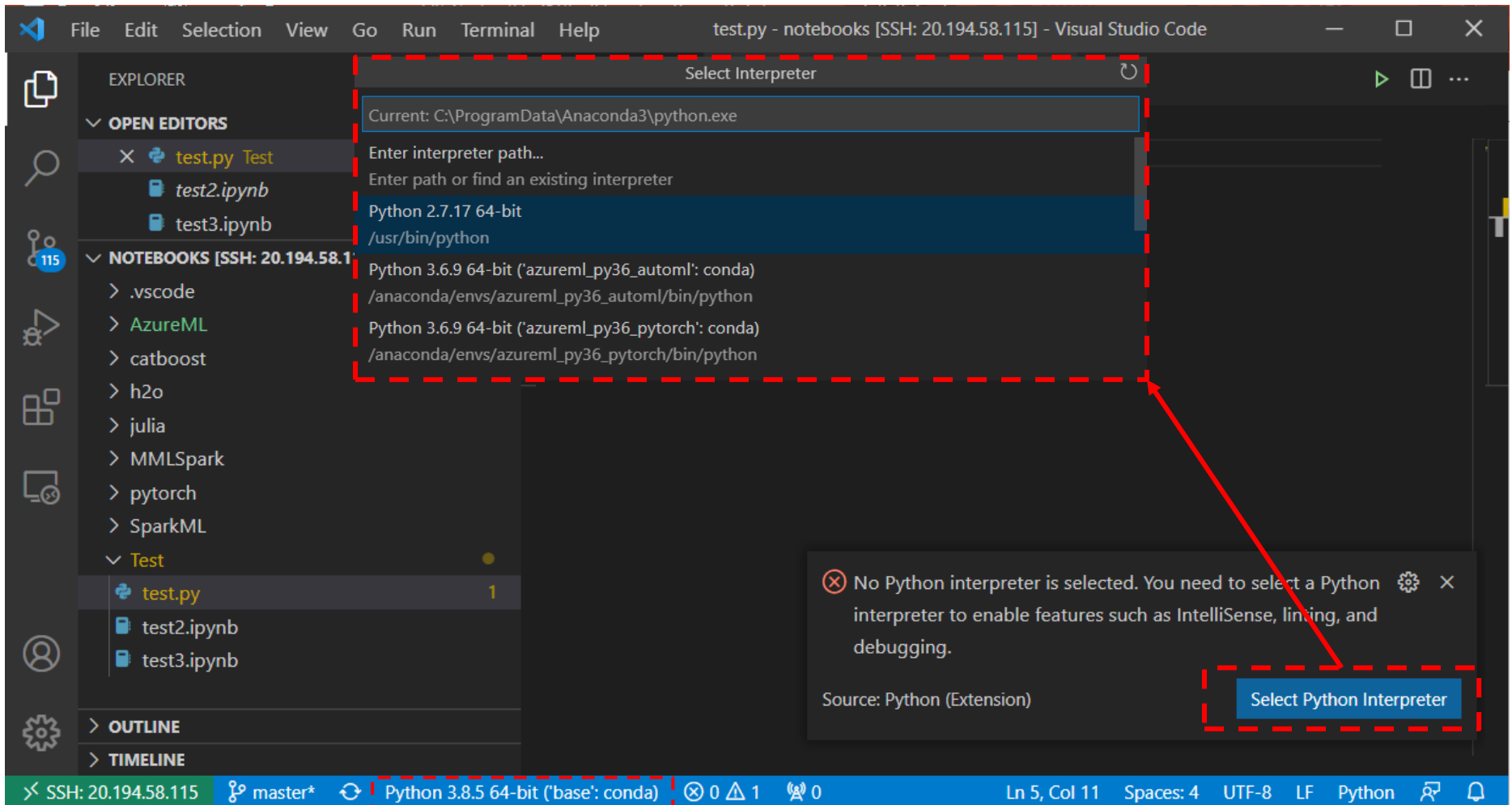
4. 원하는 폴더에서 우클릭 – New File – python 파일(.py) 생성



# 05. Remote 서버(VM) 활용

- 연결한 Linux VM에서 Python설치 및 활용

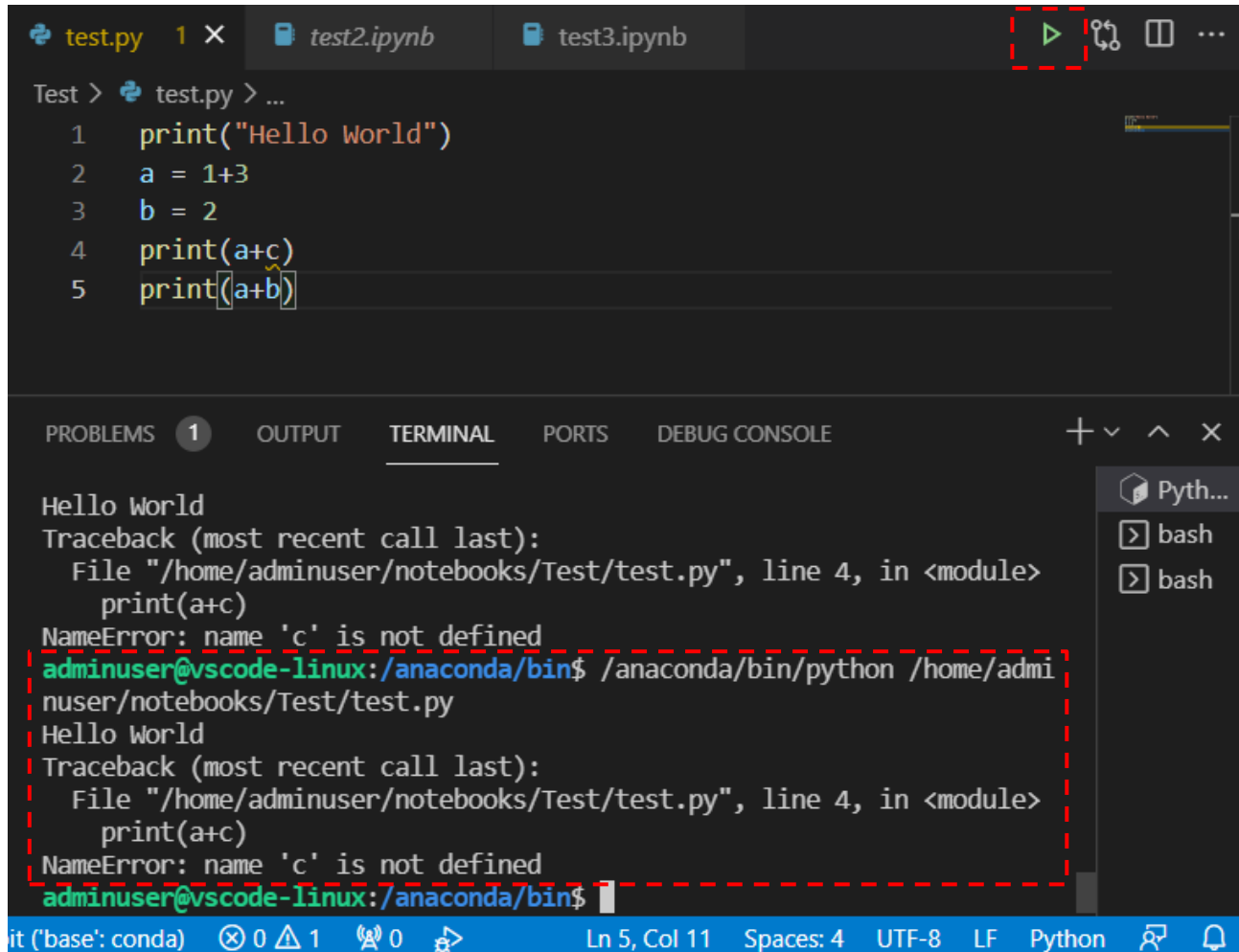
5. Select Python Interpreter – Interpreter 선택 – Status bar에 나타남



## 05. Remote 서버(VM) 활용

- 연결한 Linux VM에서 Python 설치 및 활용

6. 실행 – Terminal에서 결과 확인



The screenshot shows the VS Code interface with a file explorer at the top containing 'test.py', 'test2.ipynb', and 'test3.ipynb'. The editor window displays a Python script in 'test.py' with the following code:

```
1 print("Hello World")
2 a = 1+3
3 b = 2
4 print(a+c)
5 print(a+b)
```

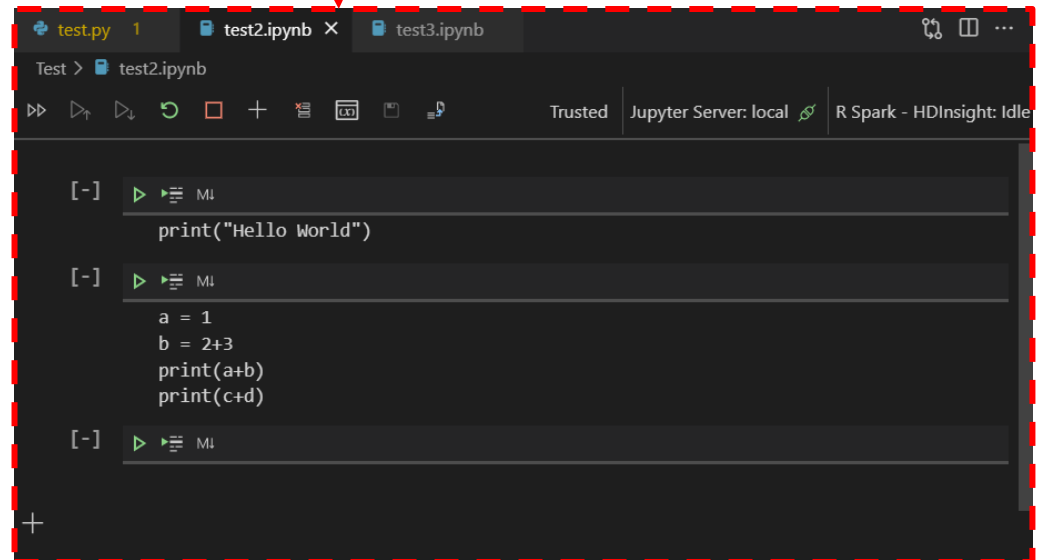
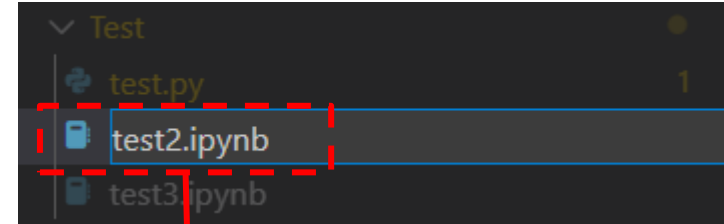
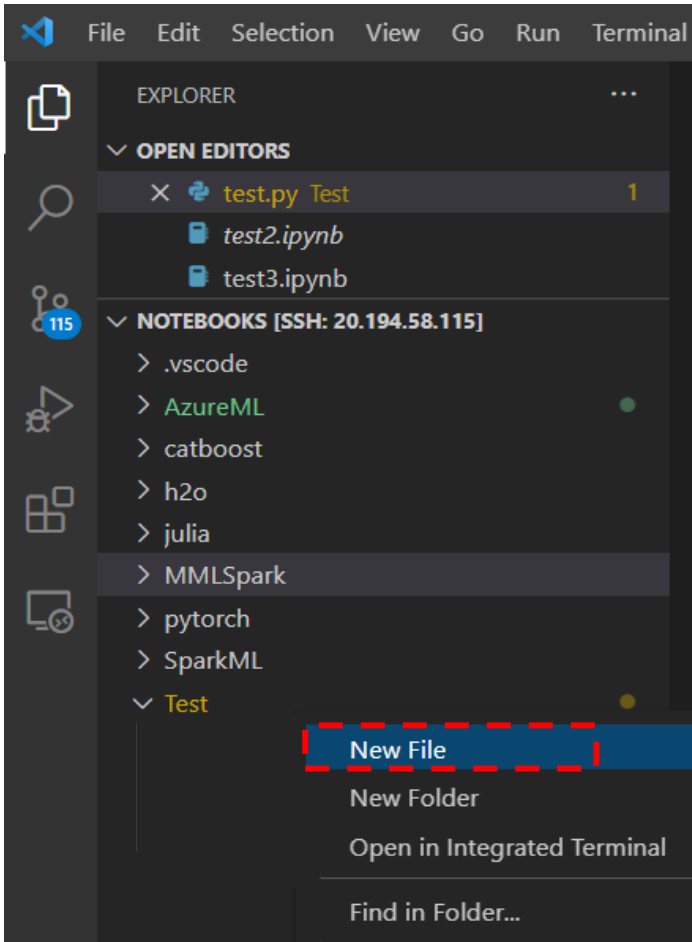
The terminal panel at the bottom shows the output of running the script. It displays 'Hello World' followed by a traceback error: 'NameError: name 'c' is not defined'. The error message is highlighted with a red dashed box. Below the error, the terminal shows the command to run the script using the Anaconda Python interpreter: `/anaconda/bin/python /home/adminuser/notebooks/Test/test.py`. The output of the command is 'Hello World' followed by the same error message. The terminal prompt is `adminuser@vscode-linux: /anaconda/bin$`.



# 05. Remote 서버(VM) 활용

- 연결한 Linux VM에서 Jupyter Notebook 활용

1. 우클릭 – New File – Jupyter Notebook 파일(.ipynb) 생성



# 05. Remote 서버(VM) 활용

- Remote VM config - SSH Key

Windows cmd 실행

> **ssh-keygen -t rsa -b 4096**

```
C:\Users\adminuser>ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\adminuser/.ssh/id_rsa):
Created directory 'C:\Users\adminuser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\adminuser/.ssh/id_rsa.
Your public key has been saved in C:\Users\adminuser/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:8jFU+z2K0n/yrKnXrnANlcIM2hLDWcZJnBo6V6mYvt4 adminuser@BRDSVM00-gpu-wi
The key's randomart image is:
+---[RSA 4096]-----+
|      .O**+      |
|    +BB=      .  |
|  ++=O + O      |
| =.+ . . +      |
| ..oS   O O      |
| .O + . + .      |
| .O + O..      |
| .. . +OOO      |
| .. E .o=B+      |
+---[SHA256]-----+
```

- 서버(~/.ssh/authorized\_key)에 ssh public key(/.ssh/id\_rsa.pub) Copy

> **scp <유저홈디렉터리패스>/.ssh/id\_rsa.pub <서버계정명@서버주소>:~/tmp.pub**

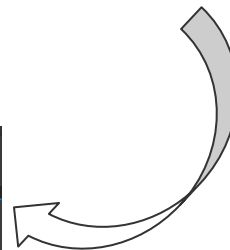
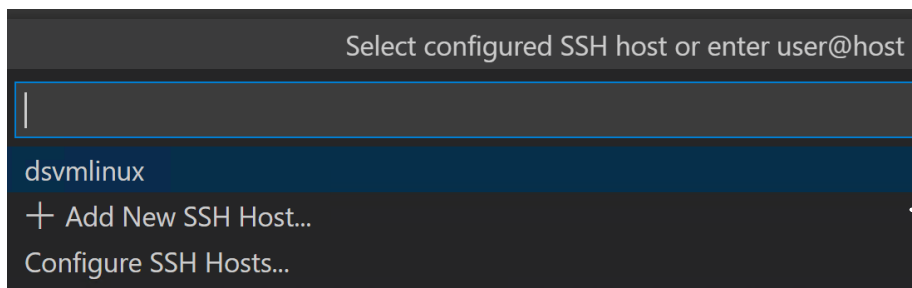
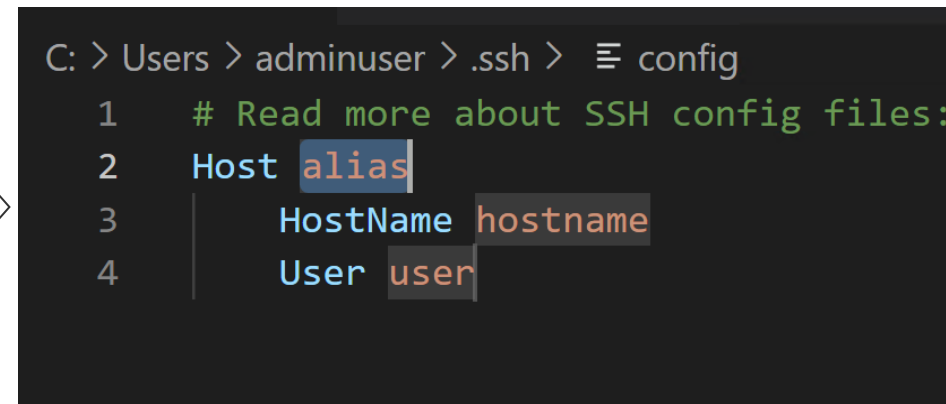
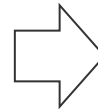
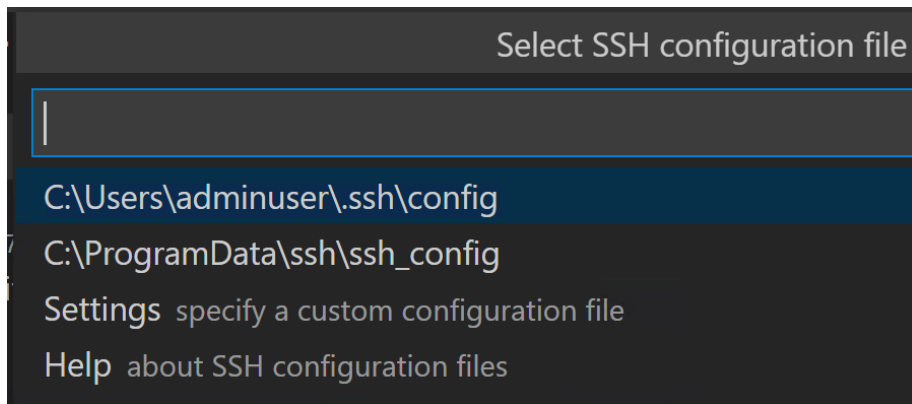
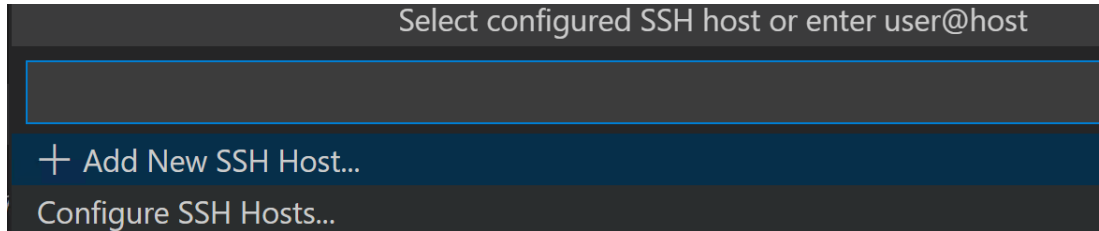
- Linux에서 아래 명령어 실행

>**mkdir -p ~/.ssh**

>**chmod 700 ~/.ssh && cat ~/tmp.pub >> ~/.ssh/authorized\_keys && chmod 600 ~/.ssh/authorized\_keys && rm -f ~/tmp.pub**

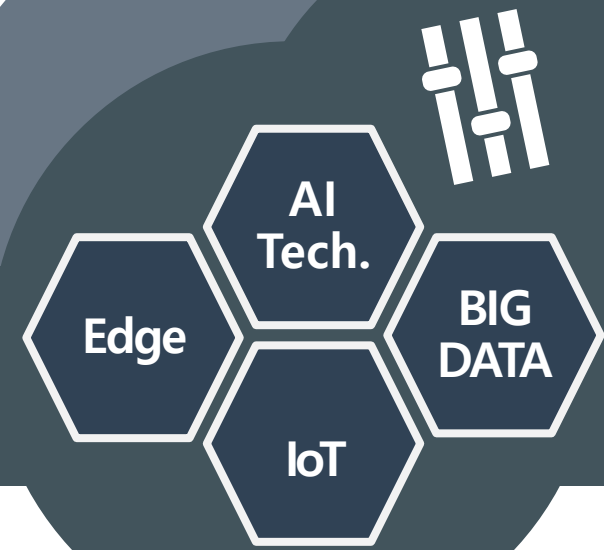
# 05. Remote 서버(VM) 활용

- Remote VM config - SSH Key



Config VM으로 접속

# 목차

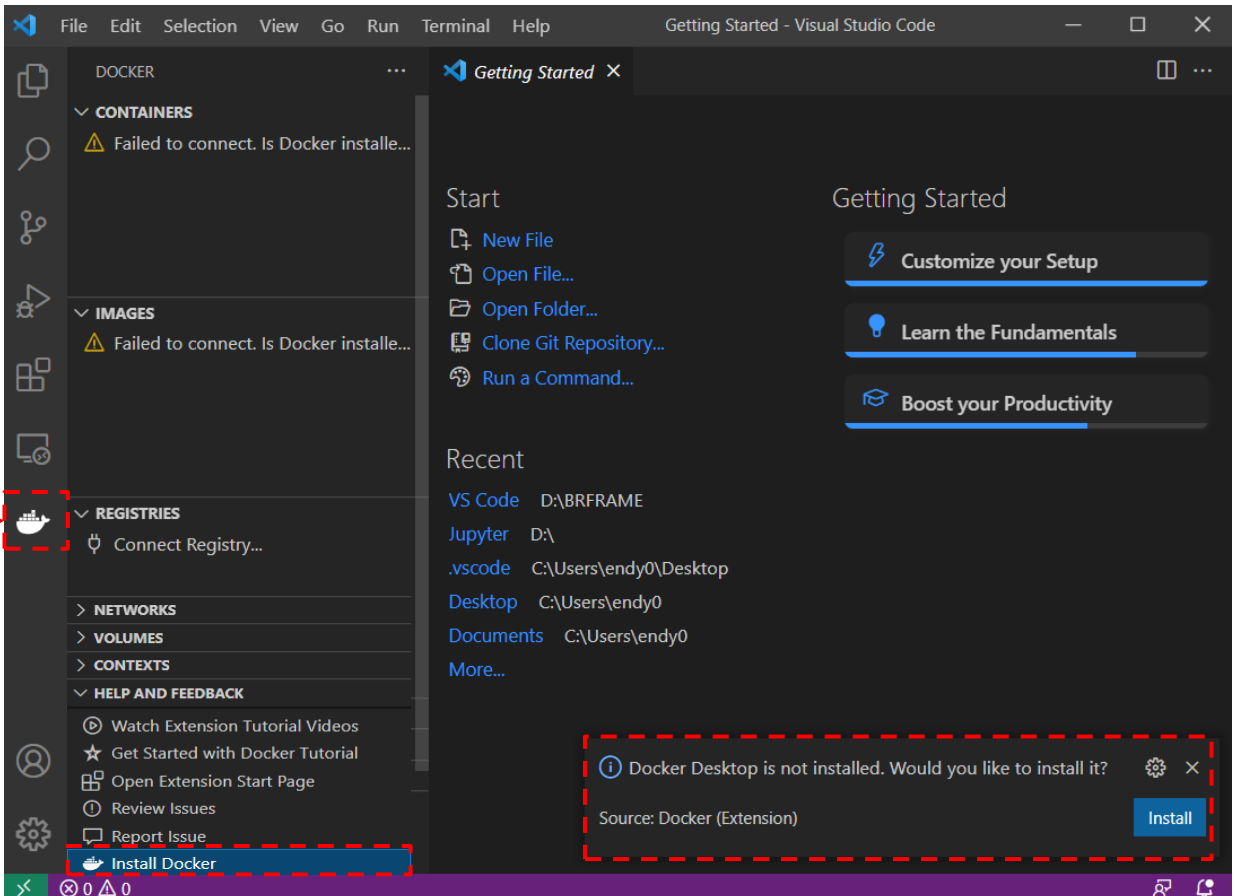
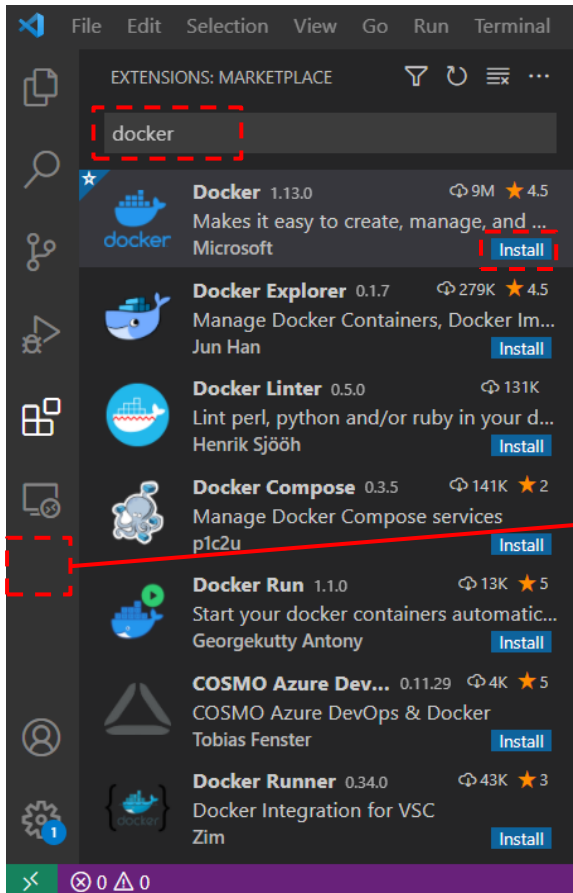


- 1 Visual Studio Code
- 2 주요 기능
- 3 Python 활용
- 4 Jupyter Notebook 활용
- 5 Remote 서버(VM) 활용
- 6 Docker 활용

# 06. Docker 활용

## • Docker 설치

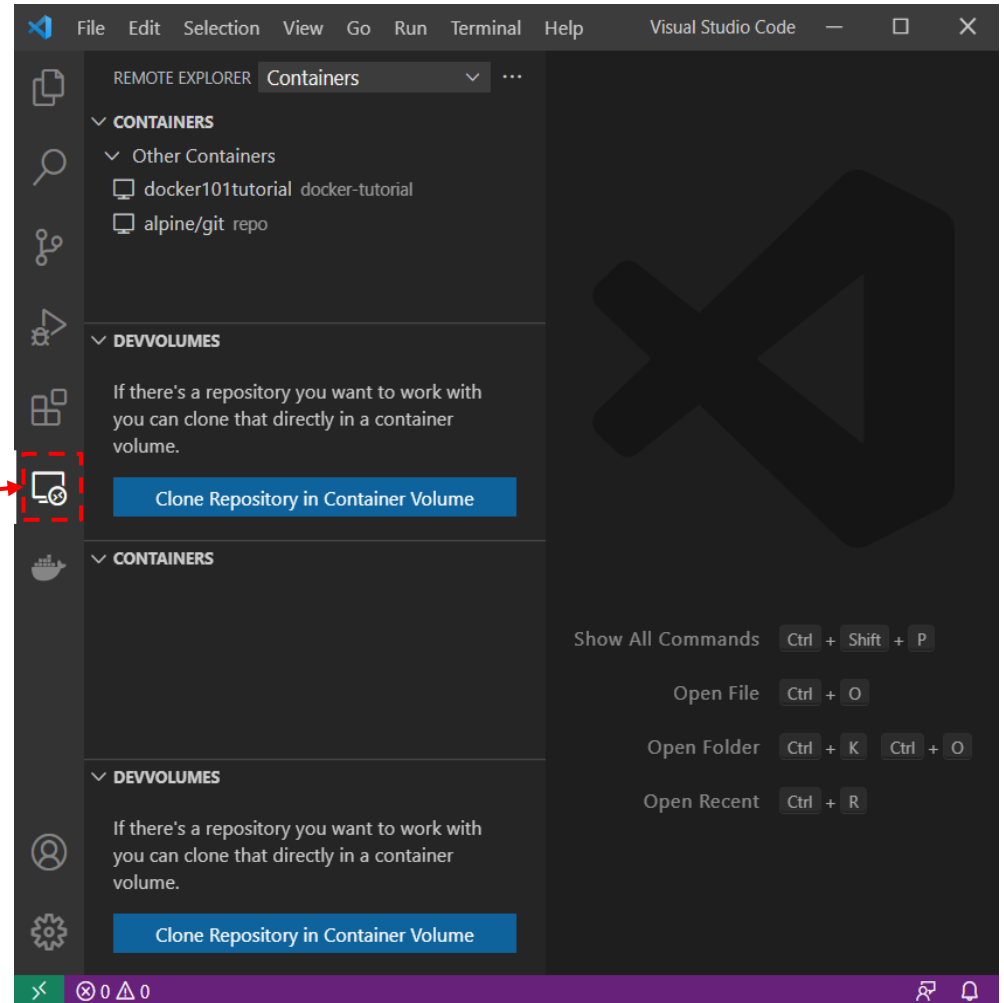
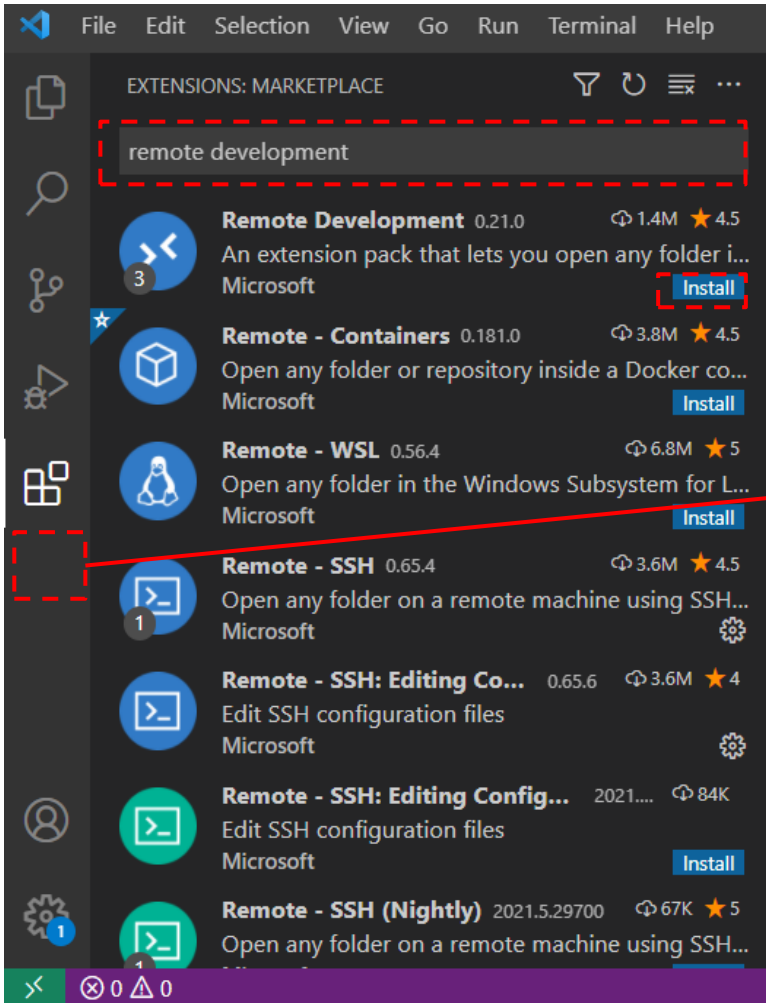
1. Extensions 검색창에 docker를 검색하여 install → Active bar에 Docker 아이콘 생성 확인
2. Docker 아이콘 클릭 → 팝업 또는 Help And Feedback에서 Desktop에도 install



# 06. Docker 활용

- Docker 연결

1. Extensions – Remote Development 확장 프로그램 install – Active bar에 Remote Explorer 아이콘 생성 확인



# 06. Docker 활용

- Docker 연결

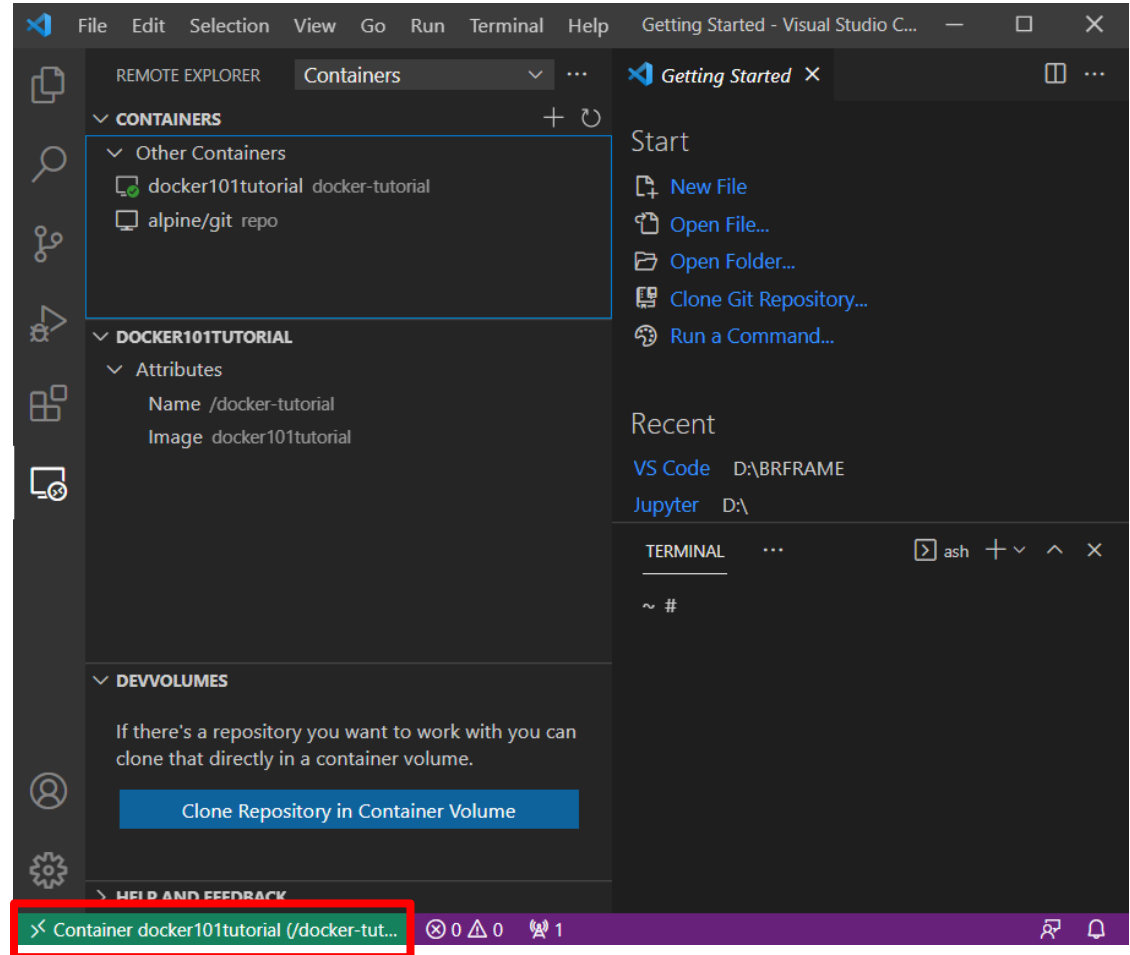
2. Command Palette에서 Remote-Containers: Attach to Running Container 클릭
  - 실행중인 컨테이너 list가 나타남 - 클릭하여 연결 - Status bar에서 연결 확인

```
>remote containers attach
```

Remote-Containers: Attach to Running Container...

Select the container to attach VS Code

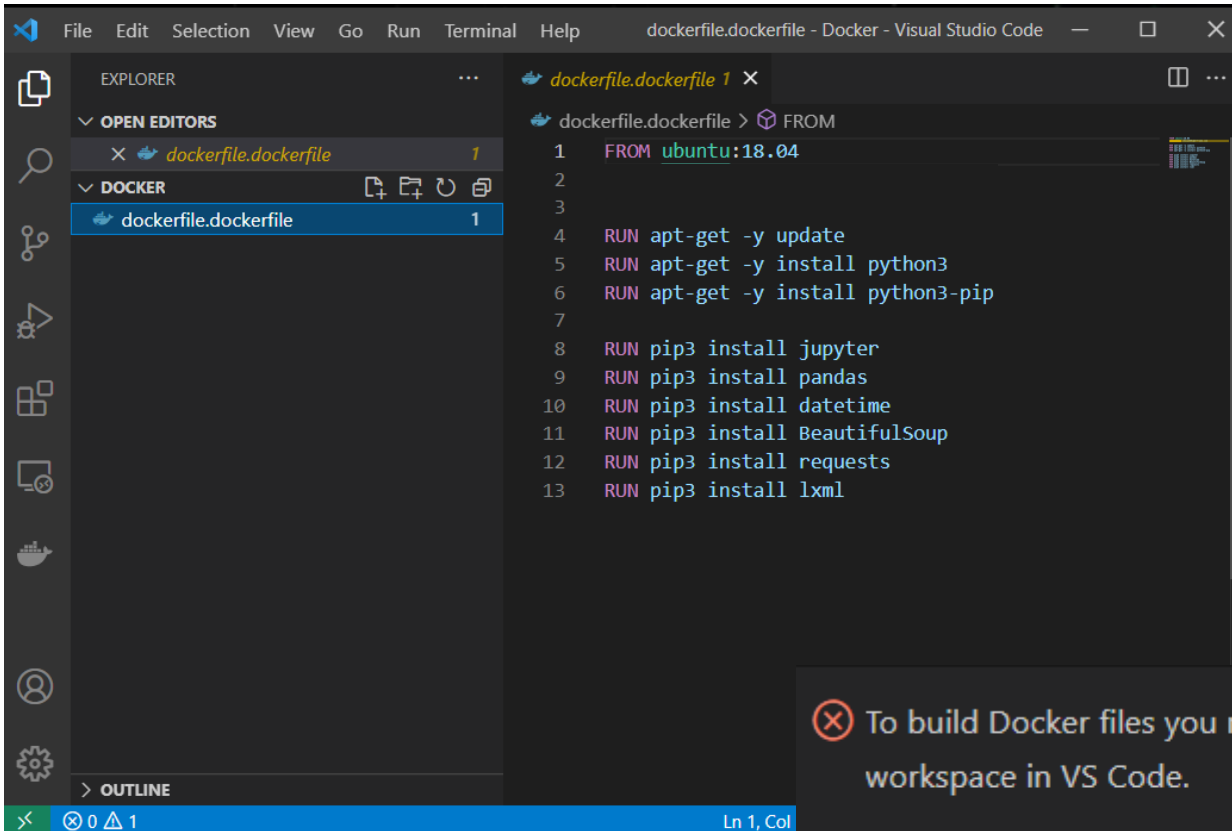
/docker-tutorial docker101tutorial 22cbdb85e25d00e0fe3e55



# 06. Docker 활용

- Container 생성

1. dockerfile 열기



```
dockerfile.dockerfile > FROM
1 FROM ubuntu:18.04
2
3
4 RUN apt-get -y update
5 RUN apt-get -y install python3
6 RUN apt-get -y install python3-pip
7
8 RUN pip3 install jupyter
9 RUN pip3 install pandas
10 RUN pip3 install datetime
11 RUN pip3 install BeautifulSoup
12 RUN pip3 install requests
13 RUN pip3 install lxml
```

⊗ To build Docker files you must first open a folder or workspace in VS Code.

Source: Docker (Extension)

Open Folder

※ 반드시 Open Folder를 통해 경로가 지정되어야 함  
(곧바로 Open File 시 위와 같은 error 팝업창 나타남)



# 06. Docker 활용

- Container 생성

- 2. 우클릭 – Build Image – Image Tag 지정 – Building 시작

The screenshot illustrates the process of building a Docker image in Visual Studio Code. The interface shows the Docker extension sidebar on the left with sections for Containers, Images, Registries, Networks, and Volumes. The main editor displays a Dockerfile with the following content:

```
1 FROM ubuntu:18.04
2 MAINTAINER
3
4 RUN
5 RUN
6 RUN
7
8 RUN
9 RUN
10 RUN
11 RUN
12 RUN
13 RUN
```

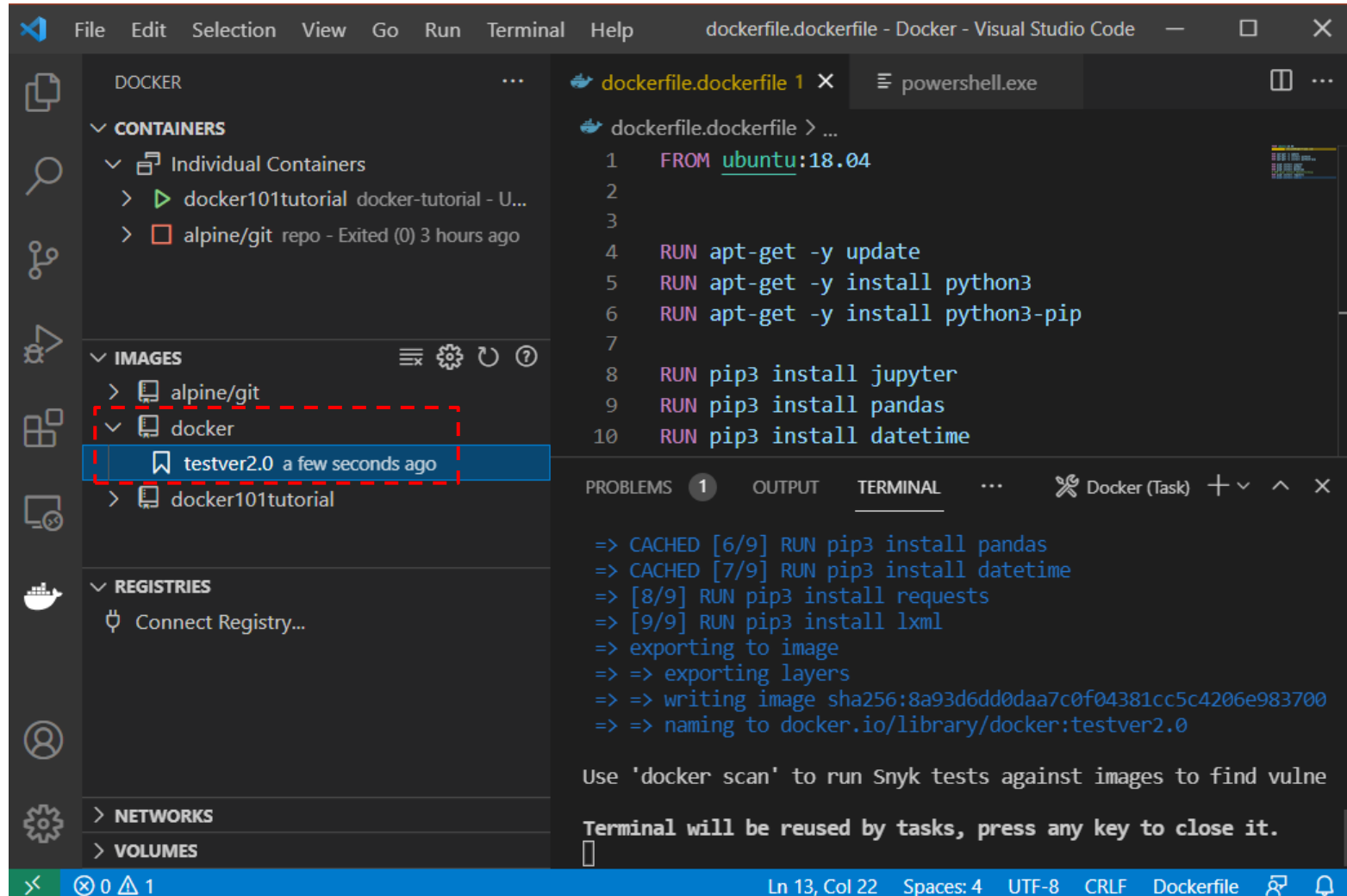
A right-click context menu is open over the Dockerfile, with the 'Build Image...' option highlighted. Below the editor, a modal dialog prompts the user to tag the image as 'docker:testver1.0', with the instruction 'Tag image as... (Press 'Enter' to confirm or 'Escape' to cancel)'. To the right, the Docker task output window shows the build progress:

```
PROBLEMS 1 OUTPUT TERMINAL ... Docker (Task) + ^ X
[+] Building 7.7s (4/14)
=> [internal] load metadata for docker.io/library/ubuntu 3.7s
=> [auth] library/ubuntu:pull token for registry-1.docker.io 0.0s
=> [ 1/10] FROM docker.io/library/ubuntu:18.04@sha256:67b730ece0d34429b455c08124f1.41kB / 1.41kB 3.9s2
=> => resolve docker.io/library/ubuntu:18.04@sha256:67b730ece0d34429b455c08124f1.41kB / 1.41kB 0.0s
=> => sha256:ceed028aae0eac7db9dd33bd89c14d5 943B / 943B 0.0s5
=> => sha256:81bcf752ac3dc8a12d54908ecdf 3.32kB / 3.32kB 0.0s
=> => sha256:4bbfd2c87b7524455f144a03b 26.70MB / 26.70MB 2.0s
=> => sha256:d2e110be24e168b42c1a2ddbc4a476a 857B / 857B 0.6s0
=> => sha256:889a7173dcfeb409f9d88054a97ab24 189B / 189B 0.8s
=> => extracting sha256:4bbfd2c87b7524455f144a03bf387c88 1.8s
```

# 06. Docker 활용

- Container 생성

## 3. Image Build 완료



The screenshot shows the Visual Studio Code interface with the Docker extension. The left sidebar displays the Docker environment, including Containers, Images, and Registries. The 'Images' section is expanded, showing a list of images: 'alpine/git', 'docker', 'testver2.0 a few seconds ago' (highlighted with a red dashed box), and 'docker101tutorial'. The main editor area shows the 'dockerfile.dockerfile' file with the following content:

```
1 FROM ubuntu:18.04
2
3
4 RUN apt-get -y update
5 RUN apt-get -y install python3
6 RUN apt-get -y install python3-pip
7
8 RUN pip3 install jupyter
9 RUN pip3 install pandas
10 RUN pip3 install datetime
```

The bottom panel shows the 'TERMINAL' output, indicating the build process is complete:

```
=> CACHED [6/9] RUN pip3 install pandas
=> CACHED [7/9] RUN pip3 install datetime
=> [8/9] RUN pip3 install requests
=> [9/9] RUN pip3 install lxml
=> exporting to image
=> => exporting layers
=> => writing image sha256:8a93d6dd0daa7c0f04381cc5c4206e983700
=> => naming to docker.io/library/docker:testver2.0

Use 'docker scan' to run Snyk tests against images to find vulne

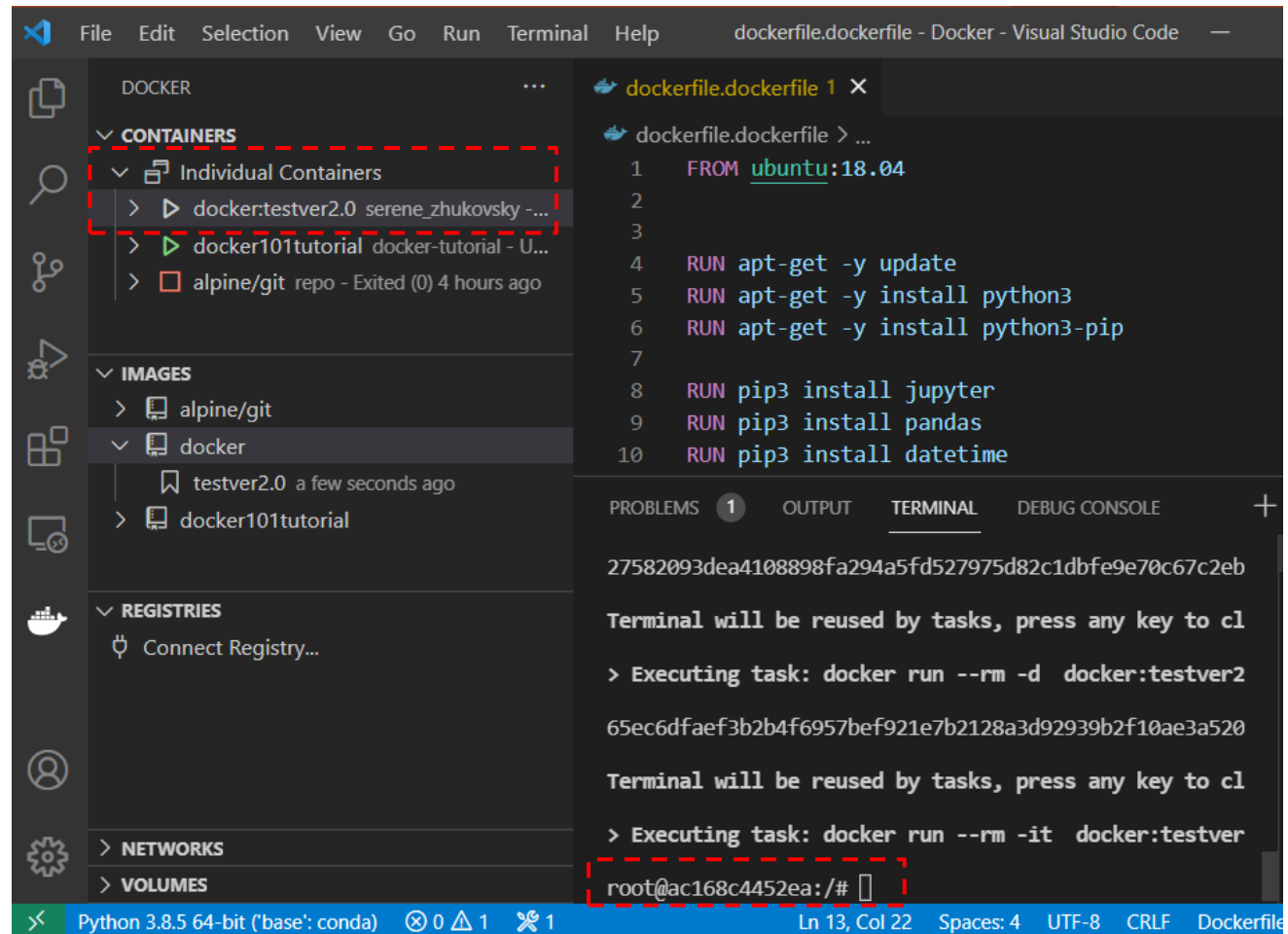
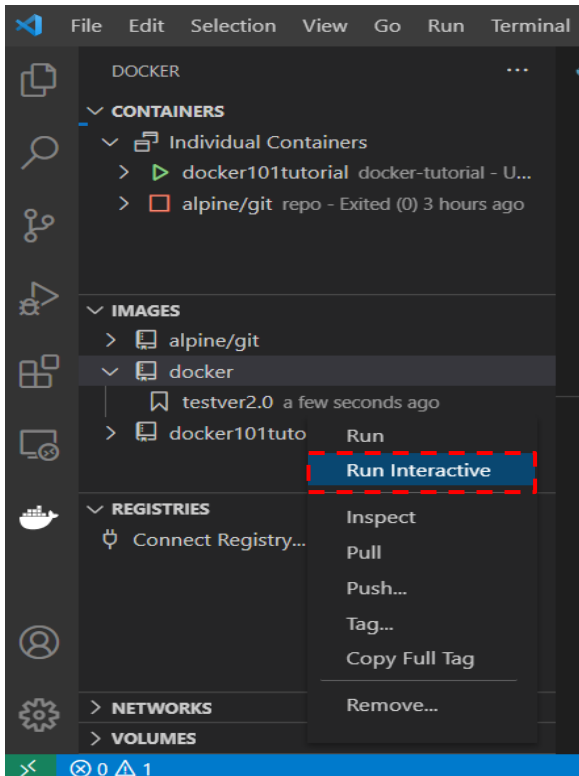
Terminal will be reused by tasks, press any key to close it.
```

The status bar at the bottom indicates the current line and column: 'Ln 13, Col 22'.

# 06. Docker 활용

- Container 생성

## 4. Image 우클릭 – Run Interactive – Container 생성 완료





# ***End of document***

## **Website**

<http://www.brframe.com/>

## **Contact Point**

**E-mail**    [admin@brframe.com](mailto:admin@brframe.com)