

“**클라우드를 있는 무한 *IT* 시대!**



## 클라우드 컴퓨팅과 AI서비스 (6주차)

융합학과 권오영

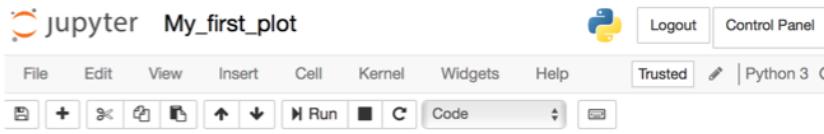
oykwon@koreatech.ac.kr

## 학습내용

- ❖ 주피터 환경
- ❖ 교내 클라우드 환경
- ❖ Markdown

# 주피터 (Jupyter: JULia PYThon R)

# 주피터 노트북



## My first plot

We will use our favorite libraries, **NumPy** and **Matplotlib**, to make a plot of a periodic function. First, our beautiful equation:

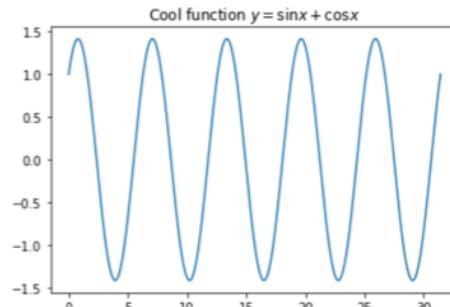
$$y = \sin x + \cos x$$

```
In [1]: import numpy  
from matplotlib import pyplot  
%matplotlib inline
```

The `numpy.linspace()` function creates an array of equally spaced numbers.

```
In [2]: x = numpy.linspace(0, 10*numpy.pi, 10**3)  
y = numpy.sin(x) + numpy.cos(x)
```

```
In [3]: pyplot.plot(x,y)  
pyplot.title('Cool function $ y = \sin{x} + \cos{x} $');
```



Jupyter header and tool bar.

A markdown cell, with title, explanation, and equation.

A code cell, setting things up with needed libraries.

A short explanation.

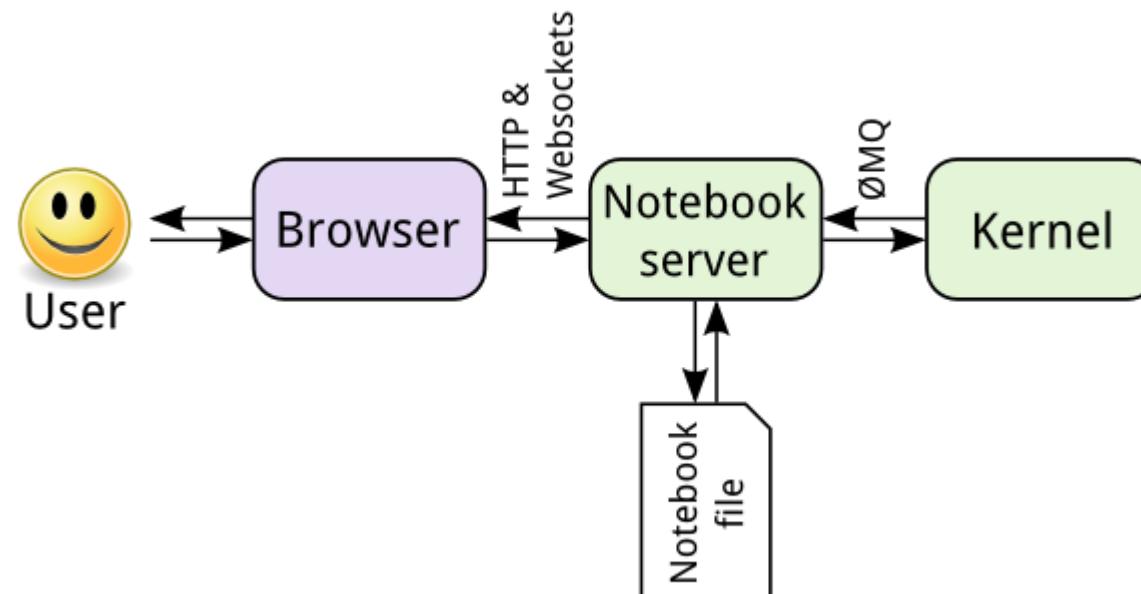
Code cells assigning two array variables, then making a line plot.

❖ 주피터(Jupyter) 프로젝트: 대화형 및 탐색적 컴퓨팅을 위한 오픈 소스 도구를 개발하는 광범위한 협업 프로젝트 (코드와 문서(콘텐츠)의 결합)

- Julia, Python, R 이 중점
- 100 개가 넘는 컴퓨터 언어 지원
- Jupyter Notebook은 2014년 후반부터 데이터 과학을 위해 가장 선호하는 환경으로 채택되면서 인기가 폭발적으로 증가
- Jupyter 노트북은 실행 코드, 방정식, 시각화 및 설명 텍스트 혼합을 지원하는 문서

# 주피터 노트북

- ❖ 노트북의 가장 중요한 구성 요소는 셀로, 노트북의 전체 내용은 셀로만 구성됨
  - 셀은 텍스트 또는 코드의 두 가지 형식 중 하나를 취함
  - 코드 셀은 입력 영역, 표시 영역 및 출력 영역의 세 영역으로 구성



[https://jupyter.readthedocs.io/en/latest/architecture/how\\_jupyter\\_ipython\\_work.html](https://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html)

# 주피터 노트북

## ❖ 입력 영역은 In []: 셀 왼쪽의 프롬프트로 식별됨

- In 프롬프트의 괄호 사이에는 숫자, 별표 또는 공백의 세 항목 중 하나가 있을 수 있고, 숫자는 이 셀이 실행되었음을 나타내고 숫자 값은 실행 순서를 나타냄(예를 들어, 노트북을 연 후 첫 번째 셀을 실행하면 프롬프트가 In [1]: 로 표시 됨)
- 입력 프롬프트 번호로 셀을 참조 할 수 있지만 셀을 다시 실행하거나 학생들이 자신의 노트북 사본을 실행하는 경우 다른 번호를 가질 수 있기 때문에 셀을 참조하는 좋은 방법은 셀을 실행하는 동안 변경되지 않는 셀 바로 위의 텍스트를 참조하는 것임
- 셀을 실행하는 동안 입력 프롬프트에 별표가 포함
- 많은 시간이 지났는데도 여전히 In [\*]: 상태이면 코드가 무한 루프 상태이거나 커널과의 통신이 끊어진 것이므로, 커널을 중단하거나 다시 시작해야 할 수도 있음

# 주피터 노트북

- ❖ 표시 영역은 코드에서 보기 위한 간단한 텍스트(예로 print("hello, world") 또는 그림과 같은 항목들을 위해 예약되어 있고, 출력 영역은 셀이 반환하는 항목을 위해 예약되어 있음)
- ❖ 노트북 셀에서 Tab 키는 자동 완성기능으로 이용되고, SHIFT-TAB은 전체 문서를 가져오고, 메소드 또는 함수 뒤에 물음표를 사용하면 셀이 실행 된 후 메소드나 함수에 대한 도움말이 나타남

In [1]: 1 print?

Docstring:

```
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file: a file-like object (stream); defaults to the current sys.stdout.
sep: string inserted between values, default a space.
end: string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type: builtin_function_or_method
```

# 주피터 노트북

- ❖ 매직(magic)은 Jupyter 내에서만 작동하고 사용자가 언어/커널 관련 기능에 액세스 할 수 있도록 하는 메타 명령임
  - 매직 명령어는 단일라인을 위한 명령어은 %로 시작하고, 전체 셀에 영향을 끼치는 명령은 %%로 시작함
  - Matplotlib는 시각화를 지원하고, 매직 %matplotlib은 결과 그림을 노트북에 표시함
    - ✓ %matplotlib inline 은 노트북에 포함 된 정적 이미지를 만들고,
    - ✓ %matplotlib notebook 은 상호작용(확대, 축소등)이 가능한 이미지를 생성함
  - 매직 %run은 외부 스크립트 (및 기타 노트북)를 실행하고 출력을 캡처하여 노트북에 표시
    - ✓ %run my\_script.py

# 주피터 노트북

## ❖ 매직(magic)

- 매직 %time은 파이썬 코드의 실행 시간을 보고함
  - ✓ %time sum(range(1000))
- 매직 %timeit은 %time과 유사하지만, 코드를 여러 번 실행하고 평균 실행 시간을 보고함
- 매직 %reset은 입출력과 함께 모든 사용자 정의 변수를 삭제
  - ✓ %reset -s 는 소프트 리셋이며 사용자 정의 변수만 제거
- 매직 %debug 는 코드가 예외상황으로 인해 멈추면 사용, 즉 문제가 발생하면 디버그 모드로 전환하여 변수를 탐색하고 문제의 원인을 찾을 수 있음

## ❖ 셀의 전체 내용에서 작동하는 좋은 예

%%HTML로 셀 전체를 HTML로 해석하여 적절하게 렌더링

# 주피터 노트북

- ❖ IPython 커널을 실행하는 동안 다른 언어를 호출하는 매직도 있음
  - 매직 %%R을 사용하여 IPython 노트북 내에서 R 코드를 실행
- ❖ 노트북은 텍스트로 작성되어 버전 관리가 용이함
  - 여러 사람들이 GitHub와 같은 플랫폼을 이용해 함께 버전 관리를 할 수 있음(노트북 개발에 기여하는 과정에서 저자를 추적하고 새로운 기여를 검토하거나 새로운 과제에 대한 요청된 개발을 살펴보는 커뮤니케이션 도구로 활용)
  - 버전 관리 사용의 장점은 공개된 노트북의 렌더링 된 뷰를 제공함(GitHub는 노트북이 구성된 ASCII 텍스트가 아닌 렌더링 된 버전의 노트북을 보여줌)

# 주피터 노트북

- ❖ 파이썬 생태계의 수많은 패키지 중에서 NumPy, Scipy, Matplotlib 및 Pandas가 가장 일반적으로 사용
  - Numpy(<http://www.numpy.org/>)는 Python을 사용한 수치 및 과학 계산을 위한 기본 라이브러리로 숫자 형 배열, 선형 대수 도구, 난수 기능 등을 위한 데이터 구조가 포함되어 있음
  - SciPy(<https://docs.scipy.org/>)는 최적화, 보간, 통계 및 신호 처리와 같은 과학 컴퓨팅을 위한 다양한 기능을 제공
  - Matplotlib(<https://matplotlib.org/>)는 Python의 핵심 plotting 라이브러리며 노트북에서 매직 명령어를 이용하여 %matplotlib notebook 또는 %matplotlib inline 형태로 사용 가능
  - Pandas(<https://pandas.pydata.org/>)는 데이터 분석을 위한 리소스와 레이블이 지정된 테이블 형식의 데이터를 위한 유연한 데이터 구조 제공

# 주피터 노트북 단축키

## ❖ 셀 선택 모드 (Command Mode)

[esc] 또는 [ctrl] + [m]를 눌러 셀이 아래와 같이 파란색이 된 상태(셀 선택 모드)에서 해당 단축키 누름

In [ ]: a = 10

위로 셀 추가	[a]
아래로 셀 추가	[b]
선택 셀 삭제	[d][d] (d를 두번 누름)
선택 셀 잘라내기 (삭제로 써도 무방)	[x]
선택 셀 복사하기	[c]
선택 셀 아래에 붙여넣기	[p]
선택 셀과 아래 셀과 합치기	[shift] + [m]
실행결과 열기/닫기	[o]
Markdown으로 변경	[m]
Code로 변경	[y]
파일 저장	[ctrl] + [s] 또는 [s]
선택 셀의 코드 입력 모드로 돌아가기	[enter]

# 주피터 노트북 단축키

## ❖ 코드 입력 모드 (Edit Mode)

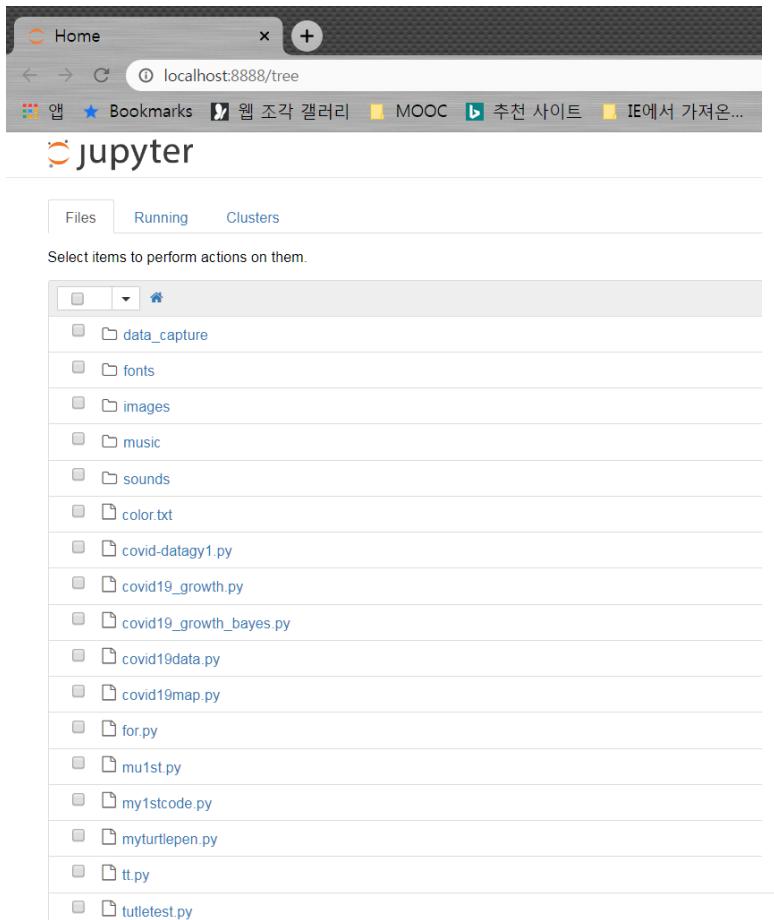
[enter]를 눌러 셀이 아래와 같이 초록색이 된 상태(코드 입력 모드)에서 해당 단축키 누름

In [ ]: a = 10

선택 셀의 코드 전체 선택	[ctrl] + [a]
선택 셀 내 실행 취소	[ctrl] + [z]
선택 셀 내 다시 실행	[ctrl] + [y]
커서 위치 라인 주석처리	[ctrl] + [/]
선택 셀 코드 실행	[ctrl] + [enter]
선택 셀 코드 실행 후 다음 Cell로 이동 (없으면 새로 추가)	[shift] + [enter]
커서 위치에서 셀 둘로 나누기	[shift] + [ctrl] + [-]
파일 저장	[ctrl] + [s]
셀 선택 모드로 돌아가기	[esc] 또는 [ctrl] + [m]

# 주피터 노트북

## ❖ 로컬서버로 작동



### ## How to use Pip from the Jupyter Notebook

If you're using the Jupyter notebook and want to install a package with `pip`, you similarly might be inclined to run pip directly in the shell:

```
In [3]: # DON'T DO THIS  
!pip install numpy
```

```
Requirement already satisfied: numpy in /Users/jakevdp/anaconda/envs/python3.6
```

For various reasons that I'll outline more fully below, this **will not generally work** if you want to use these installed packages from the current notebook, though it may work in the simplest cases.

Here is a short snippet that should generally work:

```
In [4]: # Install a pip package in the current Jupyter kernel  
import sys  
!{sys.executable} -m pip install numpy
```

```
Requirement already satisfied: numpy in /Users/jakevdp/anaconda/lib/python3.6
```

That bit of extra boiler-plate makes certain that you are running the `pip` version associated with the current Python kernel, so that the installed packages can be used in the current notebook. This is related to the fact that, even setting Jupyter notebooks aside, it's better to install packages using

# Covid-19 map 예제

- ❖ Step별로 수행 <https://opensource.com/article/20/4/python-map-covid-19>
- ❖ 패키지 설치

```
In [8]: import pandas as pd
```

```
In [9]: import sys  
!{sys.executable} -m pip install pycountry
```

```
Collecting pycountry  
  Downloading https://files.pythonhosted.org/packages/16/b6/154fe93072051d8ce7bf197690957b6d0ac9a21d51c9a1d05  
bd7c6fdb16f/pycountry-19.8.18.tar.gz (10.0MB)  
Building wheels for collected packages: pycountry  
  Running setup.py bdist_wheel for pycountry  
    Stored in directory: C:\Users\0hyoung\AppData\Local\pip\Cache\wheels\98bf0fa1c6bf8cf2cbdb750d583f84be  
51c2cd8272460b8b36bd3  
Successfully built pycountry  
Installing collected packages: pycountry  
Successfully installed pycountry-19.8.18
```

You are using pip version 7.1.2, however version 20.1.1 is available.  
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

```
In [10]: import pycountry
```

```
In [ ]:
```

# Covid-19 map 예제

The image shows two adjacent Jupyter Notebook windows. Both windows have the title 'Untitled' and are running on 'localhost:8888'. The left window shows the initial setup where the user attempts to import 'plotly' but fails due to a missing module. It then proceeds to install 'plotly' via pip, which successfully installs version 4.7.1. The right window shows the user reading a CSV dataset named 'countries-aggregated.csv' from a GitHub URL. The notebook then lists all countries in the dataset.

In [11]:

```
<ipython-input-11-3e22a21d> in <module>()
    1 import plotly.express as px
  2 
  3 ImportError: No module named 'plotly'
```

In [12]:

```
import sys
!{sys.executable} -m pip install plotly

Collecting plotly
  Downloading https://files.pythonhosted.org/packages/d7/78/eb6cbe96c8379c54819592bb228c58ed7386fcc60a55eca7d
b994321df14/plotly-4.7.1-py2.py3-none-any.whl (11.5MB)
Requirement already satisfied (use --upgrade to upgrade): six in c:\anaconda3\lib\site-packages (from plotly)
Collecting retrying<1.3.3 (from plotly)
  Downloading https://files.pythonhosted.org/packages/44/ef/beae4b4e80902f22e3af07339f079c96969c69b2c7d52a5
7ea9ae61c9d/retrying-1.3.3.tar.gz
Building wheels for collected packages: retrying
  Running setup.py bdist_wheel for retrying
    Stored in directory: C:\Users\0hyoung\AppData\Local\Temp\pip\Cache\wheels\7a9033acc7b709e2a35caa7d4cae442f6fe
6fb2c48f80823d46460
Successfully built retrying
Installing collected packages: retrying, plotly
Successfully installed plotly-4.7.1 retrying-1.3.3

You are using pip version 7.1.2, however version 20.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

In [13]:

```
# ----- Step 1 -----
URL_DATASET = 'https://raw.githubusercontent.com/datasets/covid-19/master/data/countries-aggregated.csv'
df1 = pd.read_csv(URL_DATASET)
print(df1.head) # Uncomment to see what the dataframe is like
```

	Date	Country	Confirmed	Recovered	Deaths
0	2020-01-22	Afghanistan	0	0	0
1	2020-01-22	Albania	0	0	0
2	2020-01-22	Algeria	0	0	0
3	2020-01-22	Andorra	0	0	0
4	2020-01-22	Angola	0	0	0
5	2020-01-22	Antigua and Barbuda	0	0	0
6	2020-01-22	Argentina	0	0	0
7	2020-01-22	Armenia	0	0	0
8	2020-01-22	Australia	0	0	0
9	2020-01-22	Austria	0	0	0
10	2020-01-22	Azerbaijan	0	0	0
11	2020-01-22	Bahamas	0	0	0
12	2020-01-22	Bahrain	0	0	0
13	2020-01-22	Bangladesh	0	0	0
14	2020-01-22	Barbados	0	0	0
15	2020-01-22	Belarus	0	0	0
16	2020-01-22	Belgium	0	0	0
17	2020-01-22	Belize	0	0	0
18	2020-01-22	Benin	0	0	0
19	2020-01-22	Bhutan	0	0	0

In [14]:

```
list_countries = df1['Country'].unique().tolist()
print(list_countries) # Uncomment to see list of countries
```

[Afghanistan, Albania, Algeria, Andorra, Angola, Antigua and Barbuda, Argentina, Armenia, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Belarus, Belgium, Benin, Bhutan, Bolivia, Bosnia and Herzegovina, Botswana, Brazil, Brunei, Bulgaria, Burkina Faso, Burma, Burundi, Cabo Verde, Cambodia, Cameroon, Canada, Central African Republic, Chad, Chile, China, Colombia, Comoros, Congo (Brazzaville), Congo (Kinshasa), Costa Rica, Côte d'Ivoire, Croatia, Cuba, Cyprus, Czechia, Denmark, Diamond Princess, Djibouti, Dominica, Dominican Republic, Ecuador, Egypt, El Salvador, Equatorial Guinea, Eritrea, Estonia, Eswatini, Ethiopia, Fiji, Finland, France, Gabon, Gambia, Georgia, Germany, Ghana, Greece, Guatemala, Guinea, Guinea-Bissau, Guyana, Haiti, Holy See, Honduras, Hungary, Iceland, India, Indonesia, Iran, Iraq, Ireland, Israel, Italy, Jamaica, Japan, Jordan, Kazakhstan, Kenya, Korea, South, Kosovo, Kuwait, Kyrgyzstan, Laos, Latvia, Lebanon, Lesotho, Liberia, Libya, Liechtenstein, Lithuania, Luxembourg, MS Zaandam, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Mauritania, Mauritius, Mexico, Moldova, Monaco, Mongolia, Monte negro, Morocco, Mozambique, Namibia, Nepal, Netherlands, New Zealand, Nicaragua, Niger, Nigeria, North Macedonia, Norway, Oman, Pakistan, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Poland, Portugal, Qatar, Romania, Russia, Rwanda, Saint Kitts and Nevis, Saint Lucia, Saint Vincent and the Grenadines, San Marino, São Tomé and Príncipe, Saudi Arabia, Senegal, Serbia, Seychelles, Sierra Leone, Singapore, Slovakia, Slovenia, Somalia, South Africa, South Sudan, Spain, Sri Lanka, Sudan, Suriname, Sweden, Switzerland, Syria, Taiwan\*, Tajikistan, Tanzania, Thailand, Timor-Leste, Togo, Trinidad and Tobago, Tunisia, Turkey, US, Uganda, Ukraine, United Arab Emirates, United Kingdom, Uruguay, Uzbekistan, Venezuela, Vietnam, West Bank and Gaza, Western Sahara, Yemen, Zambia, Zimbabwe]

In [15]:

```
d_country_code = {} # To hold the country names and their ISO
for country in list_countries:
    try:
        country_data = pycountry.countries.search_fuzzy(country)
        # country_data is a list of objects of class pycountry.db.Country
        # The first item ie at index 0 of list is best fit
        # object of class Country have an alpha_3 attribute
        country_code = country_data[0].alpha_3
        d_country_code.update({country: country_code})
    except:
        print('could not add ISO 3 code for ->', country)
        # If could not find country, make ISO code ''
        d_country_code.update({country: ''})
print(d_country_code) # Uncomment to check dictionary
```

could not add ISO 3 code for -> Burma  
could not add ISO 3 code for -> Congo (Brazzaville)  
could not add ISO 3 code for -> Congo (Kinshasa)  
could not add ISO 3 code for -> Diamond Princess  
could not add ISO 3 code for -> Korea, South  
could not add ISO 3 code for -> Laos  
could not add ISO 3 code for -> MS Zaandam  
could not add ISO 3 code for -> Taiwan\*

# Covid-19 map 예제

The image shows two side-by-side Jupyter Notebook interfaces. Both have tabs at the top labeled 'Home', 'Untitled', and 'Installing Python Package'. The left notebook has a status bar indicating 'localhost:8888/notebooks/Untitled.ipynb?kernel\_name=python3' and 'Last Checkpoint: 34 minutes ago (autosaved)'. The right notebook has a similar status bar with 'Last Checkpoint: 36 minutes ago (unsaved changes)'.

**Left Notebook (Working):**

- Cell 16 contains the following code:

```
# create a new column iso_alpha in the df
# and fill it with appropriate iso_3_code
for k, v in dcountry_code.items():
    df1.loc[(df1.Country == k), 'iso_alpha'] = v
```
- Cell 17 contains:

```
print(df1.head()) # Uncomment to confirm that ISO codes added
```
- The output of Cell 17 is a table:

	Date	Country	Confirmed	Recovered	Deaths
0	2020-01-22	Afghanistan	0	0	0
1	2020-01-22	Albania	0	0	0
2	2020-01-22	Algeria	0	0	0
3	2020-01-22	Andorra	0	0	0
4	2020-01-22	Angola	0	0	0
5	2020-01-22	Antigua and Barbuda	0	0	0
6	2020-01-22	Argentina	0	0	0
7	2020-01-22	Armenia	0	0	0
8	2020-01-22	Australia	0	0	0
9	2020-01-22	Austria	0	0	0
10	2020-01-22	Azerbaijan	0	0	0
11	2020-01-22	Bahamas	0	0	0
12	2020-01-22	Bahrain	0	0	0
13	2020-01-22	Bangladesh	0	0	0
14	2020-01-22	Barbados	0	0	0
15	2020-01-22	Belarus	0	0	0
16	2020-01-22	Belgium	0	0	0
17	2020-01-22	Belize	0	0	0
18	2020-01-22	Benin	0	0	0

**Right Notebook (Error):**

- Cell 18 contains:

```
import plotly.express as px
```
- Cell 19 contains:

```
# ----- Step 3 -----
fig = px.choropleth(data_frame = df1,
                     locations= "iso_alpha",
                     color= "Confirmed", # value in column 'Confirmed' determines color
                     hover_name= "Country",
                     color_continuous_scale= 'RdYlGn', # color scale red, yellow green
                     animation_frame= "Date")
```
- The output of Cell 19 is a choropleth map. A red error message is overlaid on the map, reading '문제 발생' (Problem Occurred).
- An error traceback is shown below the map:

```
ValueError Traceback (most recent call last)
<ipython-input-19-4a8d1849b5e> in <module>()
      7         animation_frame= "Date")
      8
--> 9 fig.show()
```

```
C:\Anaconda3\lib\site-packages\plotly\basedatatypes.py in show(self, *args, **kwargs)
   2822     import plotly.io as pio
   2823
--> 2824     return pio.show(self, *args, **kwargs)
   2825
   2826 def to_json(self, *args, **kwargs):
```

```
C:\Anaconda3\lib\site-packages\plotly\file_renderers.py in show(fig, renderer, validate, **kwargs)
   383     if not nbformat or LooseVersion(nbformat._version_) < LooseVersion("4.2.0"):
   384         raise ValueError(
--> 385             "Mime type rendering requires nbformat>=4.2.0 but it is not installed"
   386         )
   387
```

```
ValueError: Mime type rendering requires nbformat>=4.2.0 but it is not installed
```

# 주피터 노트북

- ❖ 주피터 노트북에 기능을 추가하는 확장이 많이 존재하며 확장을 설치하고 활성화하는 방법은 다음 사이트를 참조

<https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/install.html>

- Jupyter 노트북과 상호 작용하는 많은 도구 중 하나인 Google Colaboratory가 Jupyter 확장 기능을 활용하여 사용자 지정 상호 작용 및 프레젠테이션을 수행  
(코드 셀 왼쪽의 실행/재생 아이콘을 확정하여 사용)

- ▼ TensorFlow execution

Colaboratory allows you to execute TensorFlow code in your browser with a single click. The example below adds two matrices.

$$\begin{bmatrix} 1. & 1. & 1. \\ 1. & 1. & 1. \end{bmatrix} + \begin{bmatrix} 1. & 2. & 3. \\ 4. & 5. & 6. \end{bmatrix} = \begin{bmatrix} 2. & 3. & 4. \\ 5. & 6. & 7. \end{bmatrix}$$

The screenshot shows a code cell in Google Colaboratory. The code imports tensorflow, defines input1 and input2, creates a session, and evaluates the sum of the two matrices. The output is an array with two rows and three columns, containing values 2.0, 3.0, 4.0 in the first row and 5.0, 6.0, 7.0 in the second row, all of type float32.

```
import tensorflow as tf

input1 = tf.ones((2, 3))
input2 = tf.reshape(tf.range(1, 7, dtype=tf.float32), (2, 3))
output = input1 + input2

with tf.Session():
    result = output.eval()
result
```

array([[2., 3., 4.],  
 [5., 6., 7.]], dtype=float32)



## ❖ Jupyter 노트북을 확장한 colab

- 구글이제공하는 Colaboratory(또는 줄여서 'Colab')를 사용하면 브라우저에서 Python을 작성하고 실행할 수 있음  
(<https://colab.research.google.com/notebooks/intro.ipynb>)
- 구성 필요 없음
- GPU 무료 액세스
- 간편한 공유

The screenshot shows the Colab interface in a browser window. The title bar says 'Colaboratory에 오신 것을 환영합니다' (Welcome to Colaboratory). On the left, there's a sidebar with a '목차' (Table of Contents) section containing links like '시작하기', '데이터 과학', '머신러닝', '추가 리소스', and '머신러닝 예시'. Below the sidebar, there's a '시작하기' section with text about Colab being a large-scale environment for running code. It shows a code cell with the following Python code:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

The output of the code is 86400. Below the code cell, there's a note about running cells and a link to 'Colab 메모장 만들기'. Another code cell below it shows:

```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

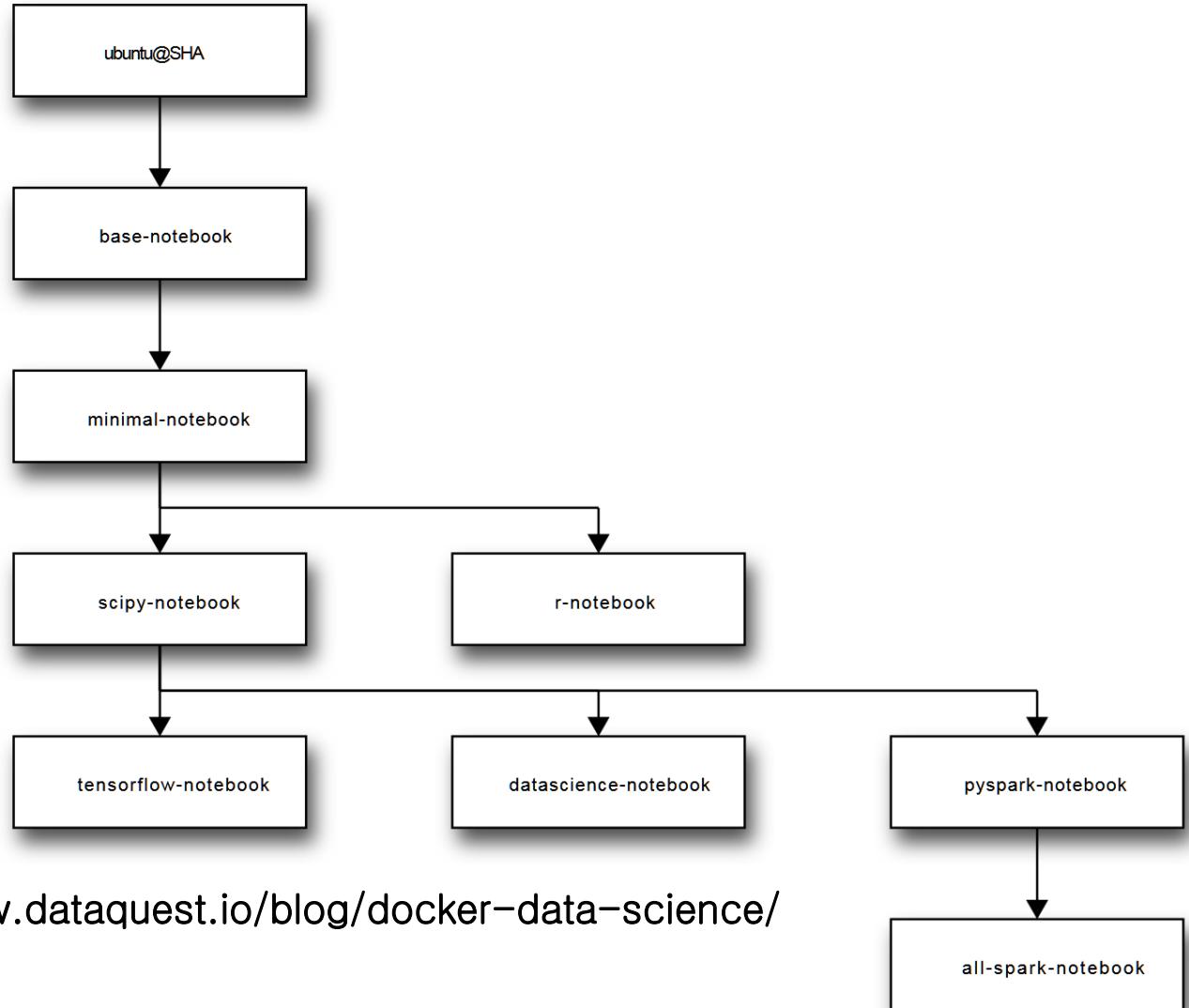
The output of this code is 604800. At the bottom, there's a note about using Colab notebooks and a link to 'Colab 메모장'.

## 리눅스에서 도커를 활용한 주피터 실행

# 주피터 노트북 실행

- ❖ 주피터 노트북을 활용할 수 있는 Docker image 계층

minimal-notebook



출처: <https://www.dataquest.io/blog/docker-data-science/>



# 주피터 노트북 실행

## ❖ Python 실행

The screenshot shows a terminal window titled "RabbitMQ [실행 중] - Oracle VM VirtualBox". The terminal is running on a Linux system with a dark theme. The user has navigated to the directory `~/mydocker` and run a Docker container with the command `docker run -it python:3.7`. The terminal output shows the container pulling the Python image from the library and then executing a simple "Hello, World" print statement.

```
rabbitmq@rabbitmq-VirtualBox:~$ cd mydocker/
rabbitmq@rabbitmq-VirtualBox:~/mydocker$ ls
rabbitmq@rabbitmq-VirtualBox:~/mydocker$ ls
rabbitmq@rabbitmq-VirtualBox:~/mydocker$ docker run -it python:3.7
Unable to find image 'python:3.7' locally
3.7: Pulling from library/python
376057ac6fa1: Pull complete
5a63a0a859d8: Pull complete
496548a8c952: Pull complete
2adae3950d4d: Pull complete
0ed5a9824906: Pull complete
bb94ffe72389: Pull complete
70d0b3967cd8: Pull complete
5efaeecfa72a: Pull complete
db8816f44548: Pull complete
Digest: sha256:00759a433cfec8c420688d4d0f941bc789d84bd248cda8d8ece7c2a93554eac2
Status: Downloaded newer image for python:3.7
Python 3.7.7 (default, May 16 2020, 07:16:36)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello, World')
Hello, World
>>>
```

# 주피터 노트북 실행

## ❖ 주피터 노트북 (minimal\_notebook) 실행

- docker run -p 8888:8888 jupyter/minimal-notebook

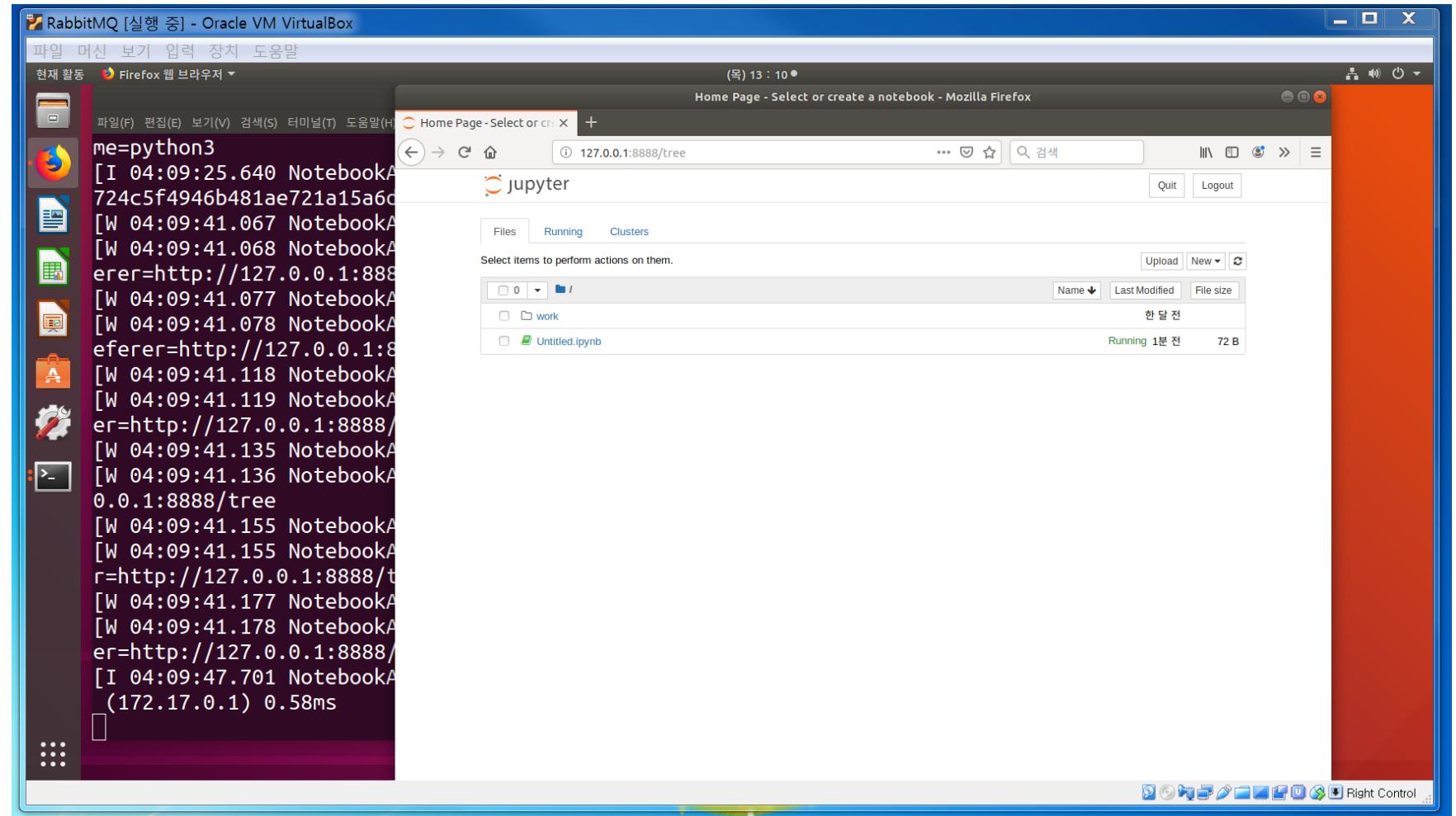
```
rabbitmq@rabbitmq-VirtualBox:~$ docker run -p 8888:8888 jupyter/minimal-notebook
latest: Pulling from jupyter/minimal-notebook
23884877105a: Pull complete
bc38caa0f5b9: Pull complete
2910811b6c42: Pull complete
36505266dc66: Pull complete
c5f758dc05d3: Pull complete
2f0b81506fd: Pull complete
441f40fe64ef: Pull complete
fb99bd169219: Pull complete
f48e613876bdd: Pull complete
e4f5ba882b7c: Pull complete
c21434c77188: Pull complete
19f19202440b: Pull complete
50c6e016f00b: Pull complete
c9fe8d804c33: Pull complete
7dff026c3fcf: Pull complete
c942425f2c13: Pull complete
a72135sec2be: Pull complete
Digest: sha256:ea37fe94e47691bbc7dd240d51d3ec1a99abde200b777c2868d6c278d106d75d
Status: Downloaded newer image for jupyter/minimal-notebook:latest
Executing the command: jupyter notebook
[I 04:02:58.600 NotebookApp] Writing notebook server cookie secret to /home/jovyan
```

```
rabbitmq@rabbitmq-VirtualBox:~$ docker run -p 8888:8888 jupyter/minimal-notebook
Executing the command: jupyter notebook
[I 04:05:06.140 NotebookApp] Writing notebook server cookie secret to /home/jovyan/.local/share/jupyter/runtime/notebook_cookie_secret
[I 04:05:07.256 NotebookApp] JupyterLab extension loaded from /opt/conda/lib/python3.7/site-packages/jupyterlab
[I 04:05:07.256 NotebookApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 04:05:07.266 NotebookApp] Serving notebooks from local directory: /home/jovyan
[I 04:05:07.266 NotebookApp] The Jupyter Notebook is running at:
[I 04:05:07.267 NotebookApp] http://6a1c7a373cee:8888/?token=8b1c20ed99acac9cdb5f07ce3ee14840ee12e9d991236bcc
[I 04:05:07.267 NotebookApp] or http://127.0.0.1:8888/?token=8b1c20ed99acac9cdb5f07ce3ee14840ee12e9d991236bcc
[I 04:05:07.267 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 04:05:07.273 NotebookApp]

To access the notebook, open this file in a browser:
file:///home/jovyan/.local/share/jupyter/runtime/nbserver-6-open.html
Or copy and paste one of these URLs:
http://6a1c7a373cee:8888/?token=8b1c20ed99acac9cdb5f07ce3ee14840ee12e9d991236bcc
or http://127.0.0.1:8888/?token=8b1c20ed99acac9cdb5f07ce3ee14840ee12e9d991236bcc
```

# 주피터 노트북 실행

## ❖ 토근 값으로 로그인 하여



# 주피터 노트북 실행

- ❖ Docker 이미지 디렉토리와 연결 -v 옵션 사용

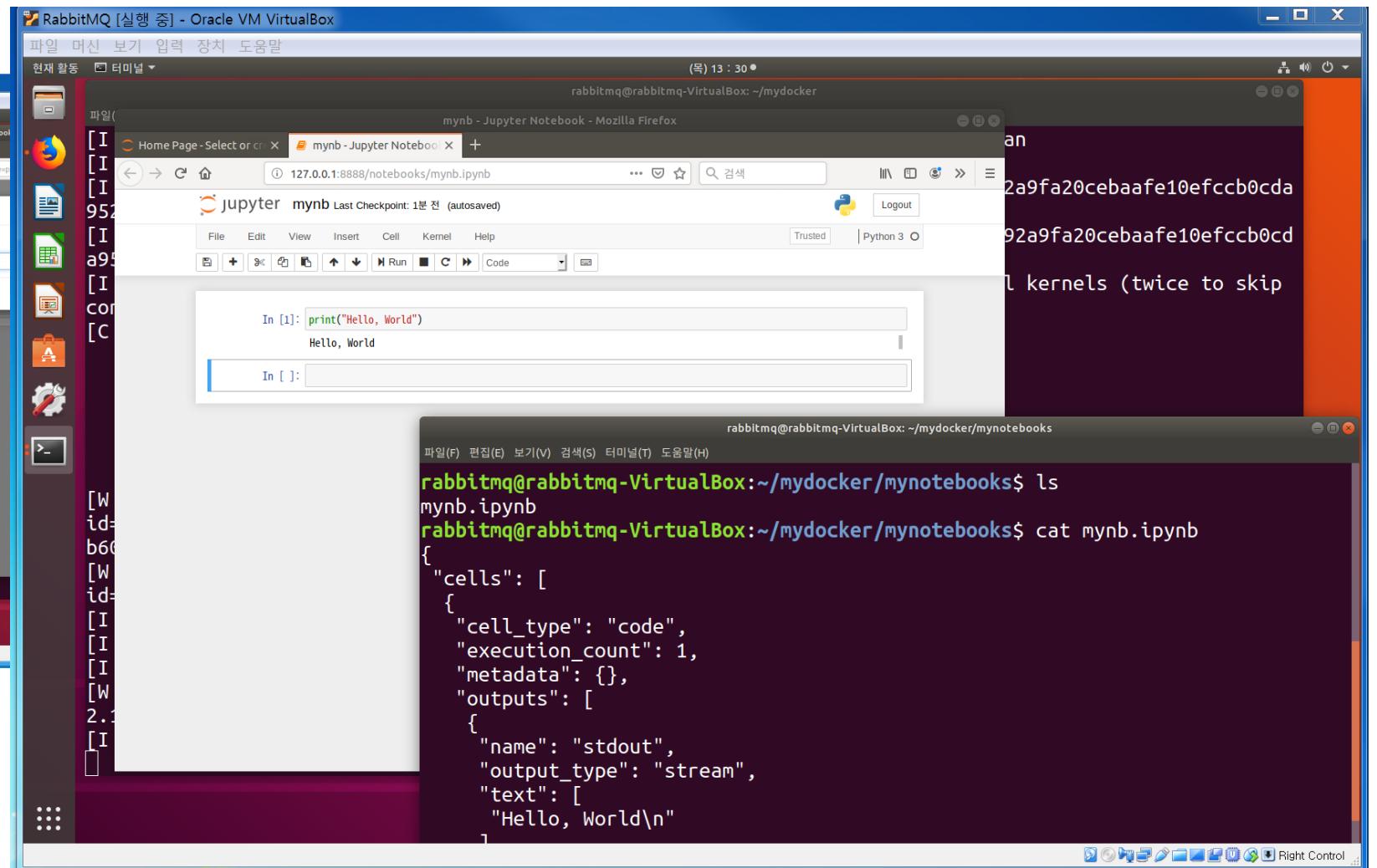
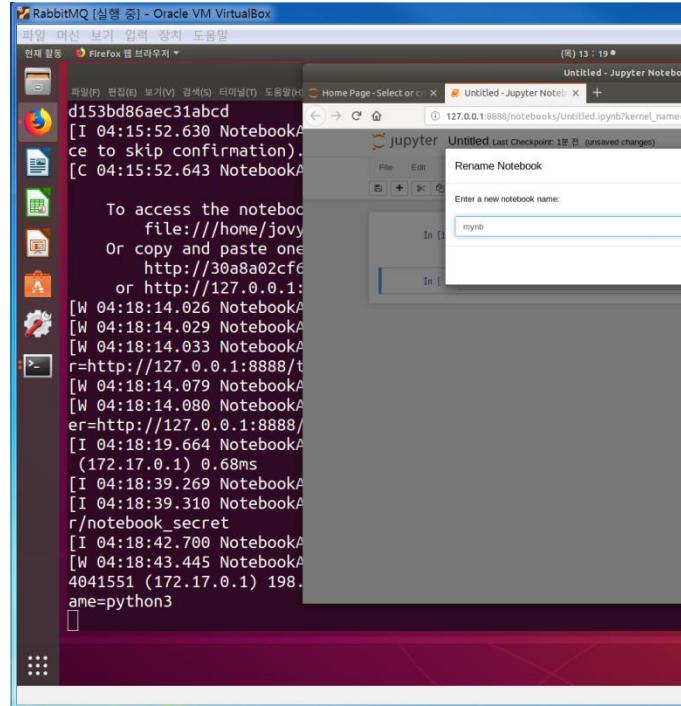
```
docker run -p 8888:8888 -v ~/mydocker/mynotebooks:/home/jovyan jupyter/minimal-notebook
```

```
RabbitMQ [실행 중] - Oracle VM VirtualBox
파일 메신 보기 입력 장치 도움말
현재 활동 터미널
(국) 13 : 26 ●
rabbitmq@rabbitmq-VirtualBox: ~/mydocker
rabbitmq@rabbitmq-VirtualBox:~/mydocker$ docker run -p 8888:8888 -v ~/mydocker/mynotebooks:/home/jovyan jupyter/minimal-notebook
Executing the command: jupyter notebook
[I 04:26:13.420 NotebookApp] JupyterLab extension loaded from /opt/conda/lib/python3.7/site-packages/jupyterlab
[I 04:26:13.421 NotebookApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 04:26:13.426 NotebookApp] Serving notebooks from local directory: /home/jovyan
[I 04:26:13.426 NotebookApp] The Jupyter Notebook is running at:
[I 04:26:13.426 NotebookApp] http://f1057033a70e:8888/?token=be14fa7f1496daa5d92a9fa20cebaafe10efccb0cd95277
[I 04:26:13.427 NotebookApp] or http://127.0.0.1:8888/?token=be14fa7f1496daa5d92a9fa20cebaafe10efccb0cd95277
[I 04:26:13.427 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 04:26:13.444 NotebookApp]

To access the notebook, open this file in a browser:
file:///home/jovyan/.local/share/jupyter/runtime/nbserver-6-open.html
Or copy and paste one of these URLs:
http://f1057033a70e:8888/?token=be14fa7f1496daa5d92a9fa20cebaafe10efccb0cd95277
or http://127.0.0.1:8888/?token=be14fa7f1496daa5d92a9fa20cebaafe10efccb0cd95277
[W 04:26:22.790 NotebookApp] 404 GET /api/kernels/4296e812-8f0d-451b-a4aa-62deddeb6075/channels?session_id=95e8bd2e856d48af86803b7a31c9814d (172.17.0.1): Kernel does not exist: 4296e812-8f0d-451b-a4aa-62deddeb6075
[W 04:26:22.866 NotebookApp] 404 GET /api/kernels/4296e812-8f0d-451b-a4aa-62deddeb6075/channels?session_id=95e8bd2e856d48af86803b7a31c9814d (172.17.0.1) 84.52ms referer=None
```

# 주피터 노트북 실행

## ❖ 저장된 파일 확인

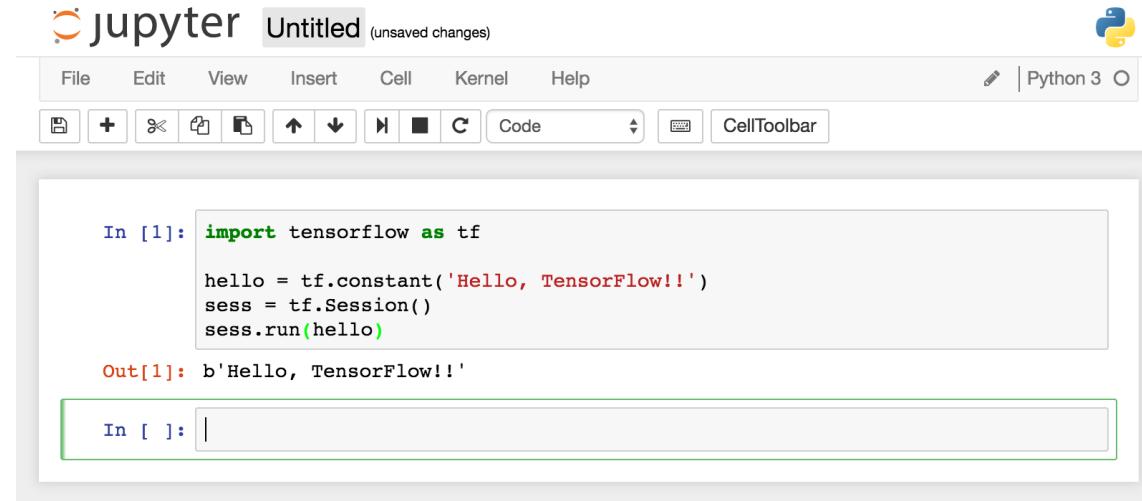


# 컨테이너 실행

## ❖ tensorflow 컨테이너 실행

```
$ docker run -d -p 8888:8888 -p 6006:6006 teamlab/pydata-tensorflow:0.1
```

- tensorflow 이미지는 python, numpy, scipy, pandas, jupyter, scikit-learn, gensim, BeautifulSoup4, Tensorflow가 설치
- 컨테이너가 실행되면 웹 브라우저로 jupyter에 접속하여 인터랙티브한 프로그래밍이 가능



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter Untitled (unsaved changes)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Help, Python 3
- Cell Content (In [1]):**

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!!')
sess = tf.Session()
sess.run(hello)
```
- Output (Out[1]):** b'Hello, TensorFlow!!'
- Input Cell (In [ ]):** An empty cell for the next input.

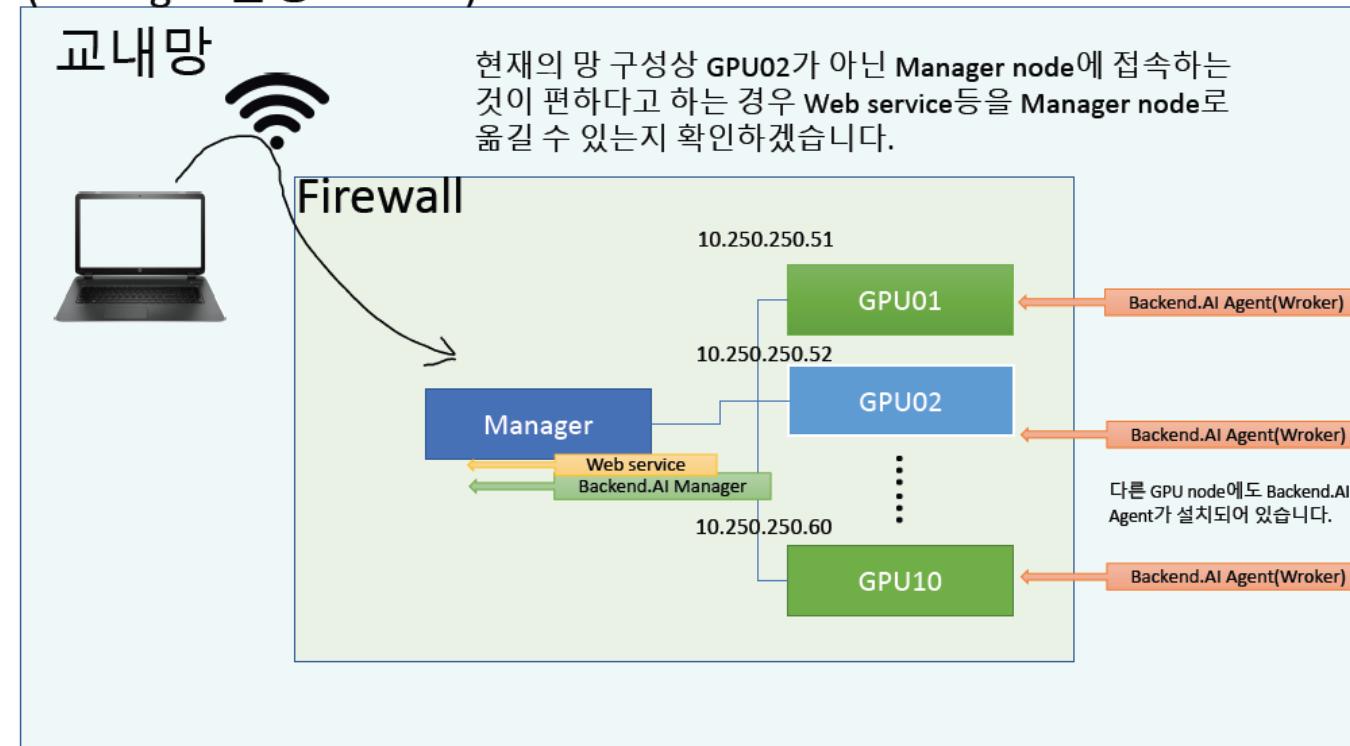
## 교내 GPU 클라우드

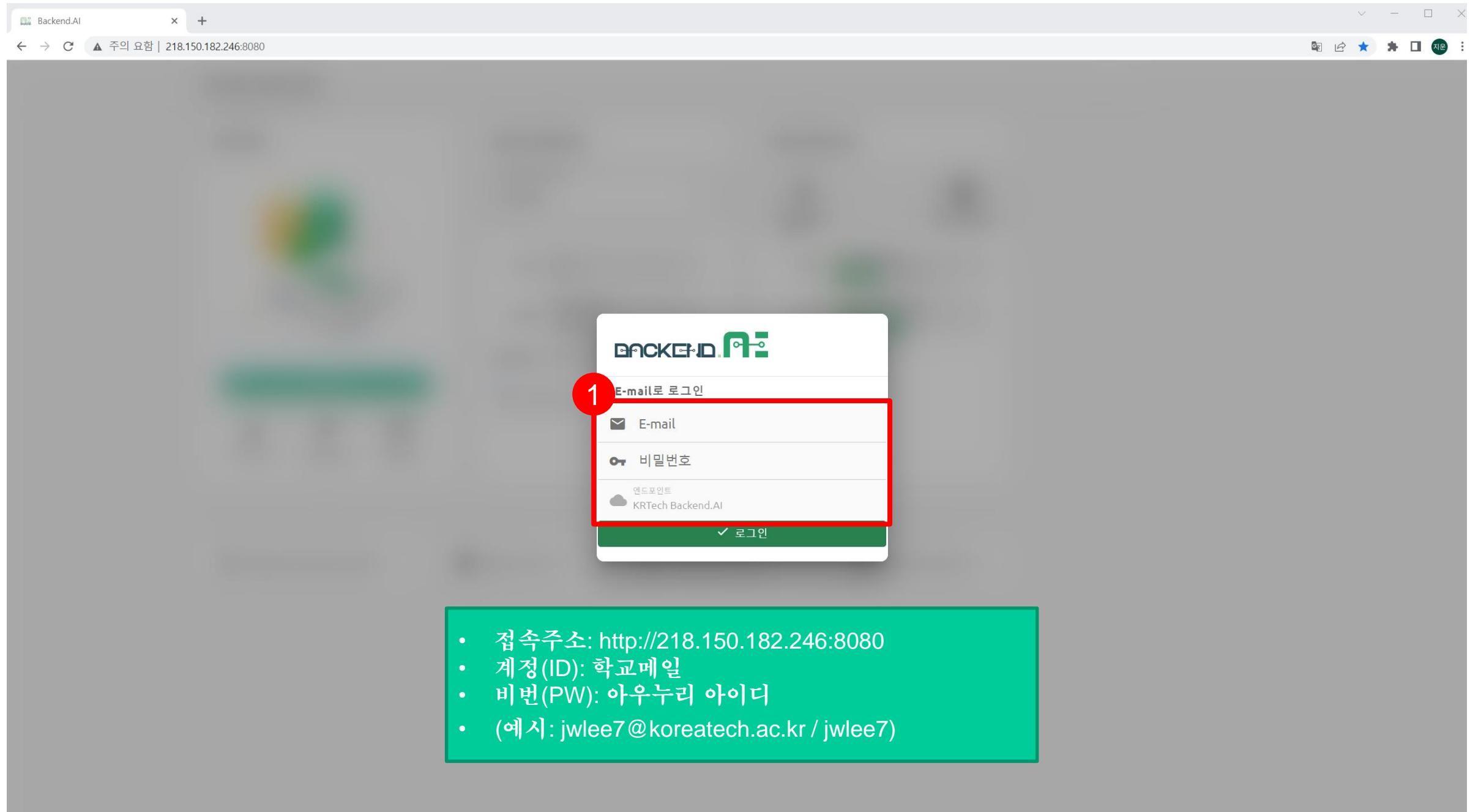
# GPU 클라우드 구성

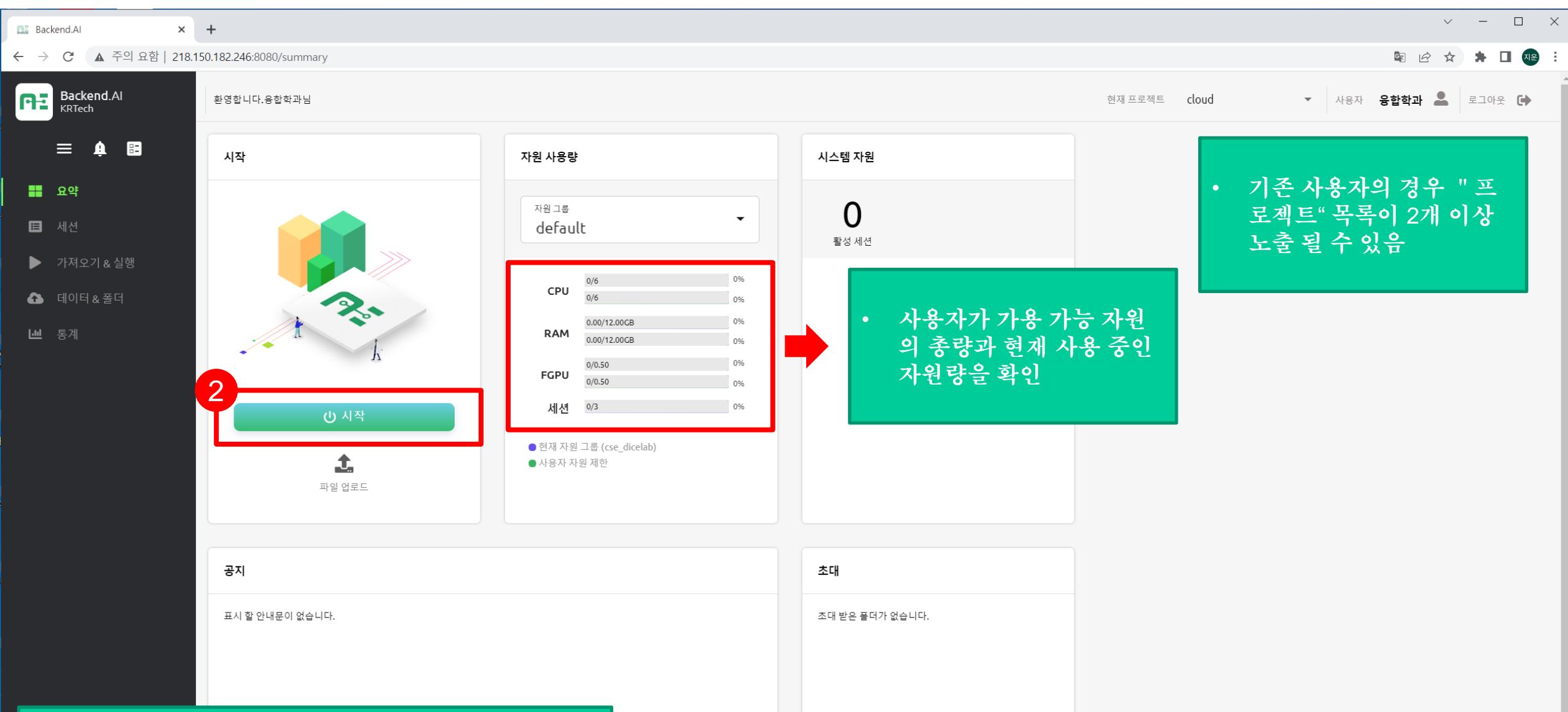
- ❖ 교내 10대의 노드로 구성된 GPU 클러스터 (교육, 실습에 적합)
- ❖ 교내 3대의 고성능 GPU 클러스터 (인공지능 연구에 적합)

## 1-2. 교내에서 web service에 접속이 가능한 망 구성 문의 (Manager 변경 version)

CONFIDENTIAL







- “시작” 버튼을 클릭하면서 자원 할당 및 세션을 생성할 수 있습니다.

- 기존 사용자의 경우 "프로젝트" 목록이 2개 이상 노출 될 수 있음

- 사용자가 가능 자원의 총량과 현재 사용 중인 자원량을 확인

Backend.AI

주의 요함 | 218.150.182.246:8080/summary

Backend.AI KRTech

환영합니다. 융합학과님

현재 프로젝트 cloud

사용자 융합학과 로그아웃

시작

자원 사용량

시스템 자원

0

자원 그룹 cse\_dicelab

CPU 0/6 0/6

RAM 0.00/12.00GB 0.00/12.00GB

FGPU 0/0.50 0/0.50

세션 0/3

새 세션 시작

3

실행 환경\* PyTorch (Krttech)

버전\* 1.7 / Python 3.8 / CUDA11.1

세션 이름 (옵션)

환경 변수 설정 (옵션)

설정

추가된 환경 변수가 없습니다.

검토 및 시작

4

실행환경은 Pytorch 또는 TensorFlow 를 선택하는 것을 권장 합니다.

3

TensorFlow (Krttech)

Language Python Krtech i

Machine Learning / Deep Learning

PyTorch Krtech i

TensorFlow Krtech i

MXNet Krtech i

Lablup Platform Nvidia GPU Cloud Krtech i

Utilities FileBrowser Krtech i

검토 및 시작

• 자원 및 세션 생성 과정으로서 사용을 원하는 실행환경과 버전을 선택 합니다. (실행환경을 선택시 가장 최신 버전이 선택 됩니다.)

Backend.AI

주의 요함 | 218.150.182.246:8080/summary

Backend.AI KRTech

환영합니다. 융합학과님

현재 프로젝트 cloud

사용자 융합학과 로그아웃

시작

자원 사용량

시스템 자원

자원 그룹 cse\_dicelab

CPU 0/6  
0/6

RAM 0.00/12.00GB  
0.00/12.00GB

FGPU 0/0.50  
0/0.50

세션 0/3

마운트할 폴더 대체이름

cloud-ai

5

새 세션 시작

마운트할 폴더 대체이름

cloud-ai

5

새 세션 시작

마운트할 폴더 대체이름

cloud-ai

5

마우트된 폴더

검토 및 시작

6

마우트된 폴더

검토 및 시작

→

5

5

5

6

- 데이터를 저장하기 위한 공간을 설정하는 과정으로서 실행환경에서 필요한 데이터를 업로드, 다운로드 그리고 코드 저장 등을 할 수 있습니다.
- 반드시 체크 후 다음 단계를 클릭 합니다.

The screenshot shows the Backend.AI web interface. On the left, there's a sidebar with icons for Home, Projects, Sessions, Import & Run, Data & Folders, and Metrics. The main area has tabs for 'Start' and 'Self-Service'. In the 'Start' tab, there's a large orange button labeled 'Start' and a file upload section. The 'Self-Service' tab shows resource usage and system status. Two overlapping windows are displayed:

- Window 1 (Left):** A 'New Session Start' dialog. It shows a dropdown for 'Resource Group' set to 'default', and a list of available resources under 'Resource Allocation'. One item, 'cloud ( 1CPU 2GB 64MB 0.1 GPU )', is highlighted with a red box and circled with a red number '7'. Below the list is a table for session details.
- Window 2 (Right):** Another 'New Session Start' dialog, similar to the first. It also shows 'Resource Group' set to 'default', and a list of resources. The same 'cloud' resource is highlighted with a red box and circled with a red number '7'. Below the list is a table for session details.

A large red arrow points from the left window to the right window, indicating the progression of the process. At the bottom of each window is a green button labeled 'Scan and Start' with a red box and circled with a red number '8'.

- 자원의 할당량을 선택하는 과정으로서 “cloud” 프리셋이 기본 값입니다. 기본 값으로 노출이 되지 않는 경우 클릭 후 “cloud”를 선택합니다.

Backend.AI

주의 요함 | 218.150.182.246:8080/summary

Backend.AI KRTech

환영합니다. 융합학과님

현재 프로젝트 cloud

사용자 융합학과 로그아웃

요약 세션 가져오기 & 실행 데이터 & 폴더 통계

시작

자원 사용량

시스템 자원

0

새 세션 시작

총 자원 할당량

CPU	메모리	공유메모리	GPU
4	8	1	0.5
Core	GB	GB	슬롯

x1 단일 노드  
컨테이너 배치

마운트된 폴더

cloud-ai

추가될 환경 변수 (옵션)

추가된 환경 변수가 없습니다.

9 시작

• 생성정보(자원할당 포함)을 최종적으로 확인 하는 단계로서 최종 확인 후 "시작"을 클릭 합니다.  
• 클릭 후 자원할당 및 세션 생성이 이루어 집니다.

The screenshot shows the Backend.AI web interface. On the left sidebar, there are several icons: '요약' (Summary), '세션' (Session), '가져오기 & 실행' (Get & Run), '데이터 & 폴더' (Data & Folder), and '통계' (Statistics). The main area has three sections: '시작' (Start), '자원 사용량' (Resource Usage), and '시스템 자원' (System Resources). A central modal window titled '새 세션 시작' (New Session Start) is open, showing resource allocation details. The modal includes a summary table:

CPU	메모리	공유메모리	GPU
4	8	1	0.5
Core	GB	GB	슬롯

Below the table, it says 'x1 단일 노드' (x1 Single Node) and '컨테이너 배치' (Container Deployment). The modal also lists mounted folders ('마운트된 폴더') and environment variables ('추가될 환경 변수 (옵션)'). At the bottom of the modal, there is a red box highlighting the '시작' (Start) button, which is labeled with the number '9'. A green box at the bottom left contains instructions for confirming resource allocation and starting the session.

Backend.AI

주의 요함 | 218.150.182.246:8080/summary

Backend.AI KRTech

환영합니다. 융합학과님

현재 프로젝트 cloud

사용자 융합학과 로그아웃

요약 세션 가져오기 & 실행 데이터 & 풀더 통계

시작

자원 사용량 시스템 자원

자원 그룹 cse\_dicelab

0 활성 세션

App

CPU 4/6 4/6

RAM 8.00/12.00Gi 8.00/12.00Gi

FGPU 0.5/0.50 0.5/0.50

세션 1/3

Utilities Console Development

10

Visual Studio Code JupyterLab Jupyter Notebook

Machine Learning Tools

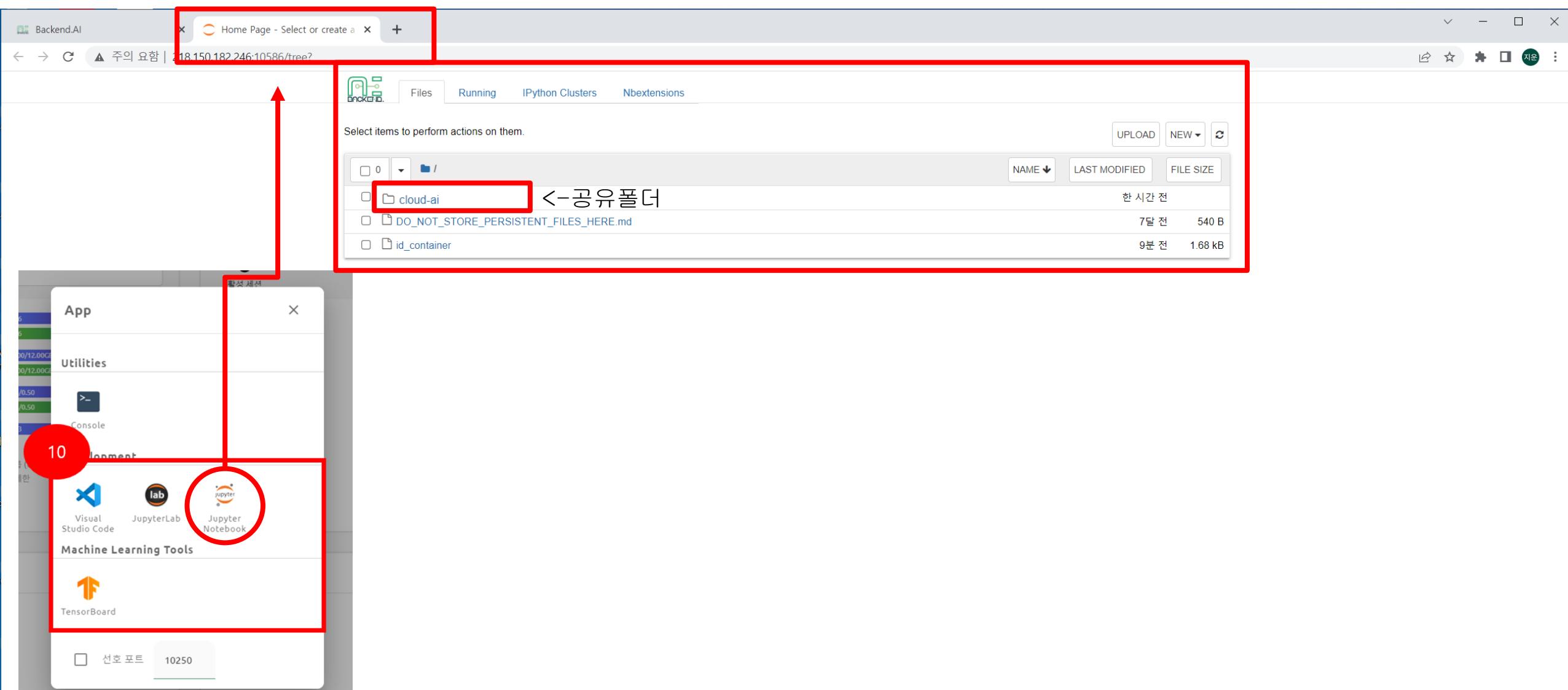
TensorBoard

선호 포트 10250

공지

표시 할 안내문이 없습니다.

- 자원 할당 및 세션 생성이 완료되면 개발 환경을 선택할 수 있습니다.
- 개발환경은 필요로 하는 도구를 선택합니다.



- (예시) 개발도구 중 Jupyter Notebook 을 선택 하면 새창 또는 새탭으로 개발도구 실행 됩니다.
- 데이터 업로드, 다운로드 및 코드 저장에는 공유 폴더 안에 넣어주시기 바랍니다.

Backend.AI

주의 요함 | 218.150.182.246:8080/data

Backend.AI KRTech 환영합니다. 융합학과님 현재 프로젝트 ML2022-02 사용자 융합학과 로그아웃

저장소 상태

0 1 10  
생성됨 공유받음 생성가능 갯수

데이터 & 폴더 1

통계

폴더 자동 마운트 폴더 + 새 폴더

# 이름 ID 위치 종류 권한 소유자 여부 제어

1 cloud-ai 1f9ec45cad0c4443 815d168f006a204a primary:nfs [User] [RW] [i] [d] 2

- 데이터의 업로드, 다운로드 및 다른 환경에서 제작한 코드를 자체적으로 제공하는 파일 브라우저 인터페이스로 할 수 있습니다.

#	이름	ID	위치	종류	권한	소유자 여부	제어
1	machinelearning2022_02	5e752d4ac1e249ac 816943dcf3033d54	primary:nfs	...	R W		2 클릭

The screenshot shows the Backend.AI application interface. At the top, there's a navigation bar with tabs like 'Backend.AI' and 'Machine Learning'. Below it is a search bar and a user profile section. The main area is a file browser titled 'cloud-ai' with a sidebar on the left containing project and user information. On the right, there's a large workspace and a terminal window. A red arrow points from the 'Edit' icon in the table above to the 'Edit' icon in the terminal window. Another red circle labeled '3' highlights the '파일브라우저 실행' (File Browser Execute) button in the terminal window.

- 내장 인터페이스가 실행되며, “파일브라우저실행” 을 클릭하면 브라우저가 실행 됩니다.
- 파일브라우저를 생성하면 자동으로 자원할당 및 세션이 생성 됩니다.



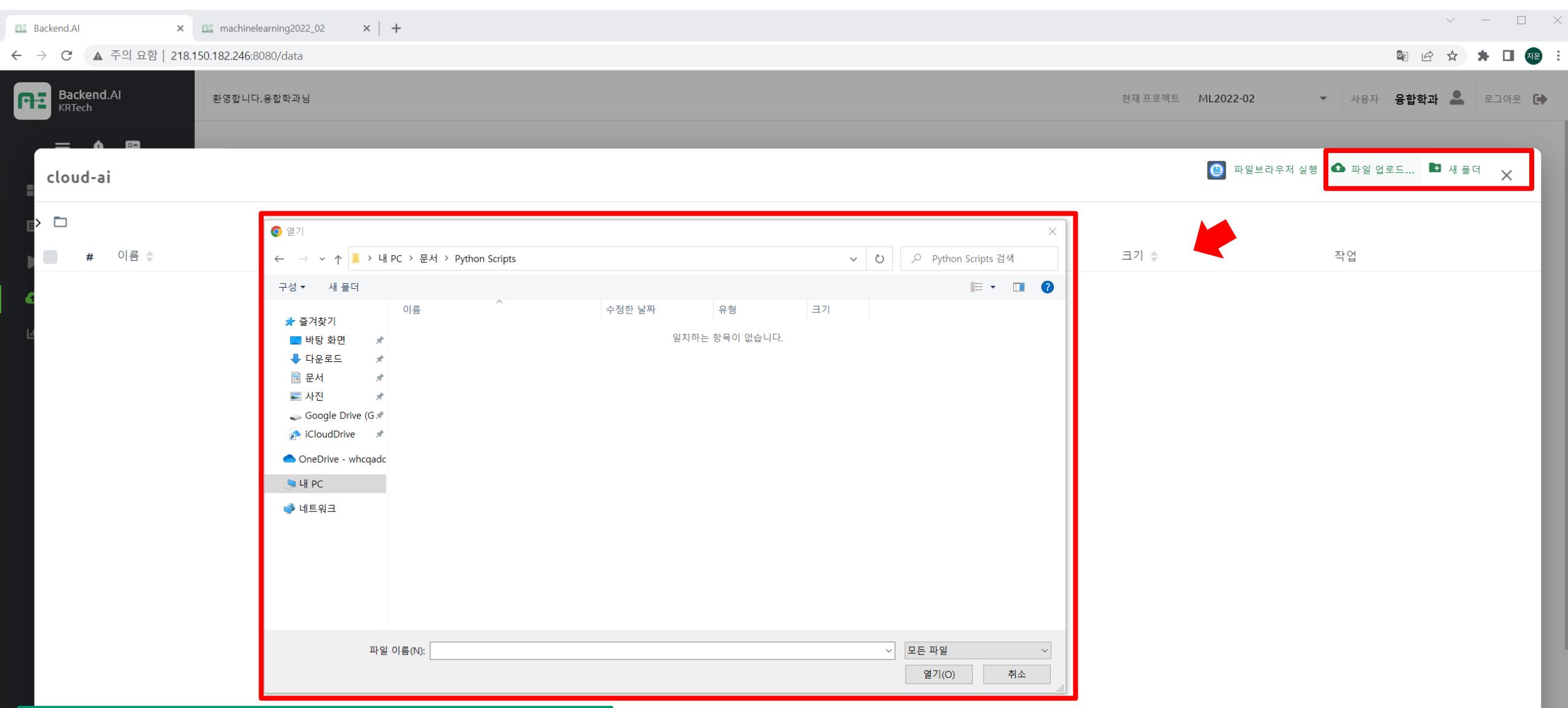
업로드, 다운로드 등 도구 제공

폴더 및 파일 생성



It feels lonely here...

- 파일브라우저가 실행된 모습으로서 폴더 및 파일 생성, 업로드 및 다운로드 등의 도구를 제공 합니다.



- 파일브라우저를 실행하지 않고, 파일 업로드를 선택하면 윈도우 탐색기 인터페이스가 실행 됩니다.

The screenshot shows the Backend.AI web interface for managing machine learning projects. The main window displays a dashboard with resource usage (CPU 83%, RAM 70%), session management (Sessions tab selected), and a list of active sessions (e.g., K1LfuwGb-jsSDK, clfPdYNm-session). A red box highlights the 'Session' tab in the sidebar and the session list. Another red box highlights the 'Utilities' section in the top right, which includes Console and FileBrowser tools, and a port forwarding dialog (선호 포트 10250).

On the right side, there is a detailed view of machine resources and their usage over time. Two machines are listed:

- cse\_dicelabRG (1 core, 0.50GB RAM, 0.00GPU): Last used 2022. 9. 6. 오후 8:45:33 (00:01:02)
- cse\_dicelabRG (4 cores, 8.00GB RAM, 0.50GPU): Last used 2022. 9. 6. 오후 8:43:18 (00:03:17)

A large green button labeled '도구 실행' (Tool Execution) is positioned between the utility section and the machine details. Red arrows point from this button to the 'Console' and 'FileBrowser' icons in the utilities list, indicating they are part of the execution tools.

A red box highlights the 'Session' tab in the sidebar and the session list. Another red box highlights the 'Utilities' section in the top right, which includes Console and FileBrowser tools, and a port forwarding dialog (선호 포트 10250).

On the right side, there is a detailed view of machine resources and their usage over time. Two machines are listed:

- cse\_dicelabRG (1 코어 0.50GB 0.00GPU machine) CPU RAM I/O R: 0.0MB/W: 0.2 2022. 9. 6. 오후 8:45:33 (00:01:02)
- cse\_dicelabRG (4 코어 8.00GB 0.50GPU machine) CPU RAM GPU I/O R: 0.0MB/W: 0.2 2022. 9. 6. 오후 8:43:18 (00:03:17)

A large green button labeled '도구 실행' (Tool Execution) is positioned between the utility section and the machine details. Red arrows point from this button to the 'Console' and 'FileBrowser' icons in the utilities list, indicating they are part of the execution tools.

A large green button labeled '세션 종료' (Session Termination) is located below the machine details, with a red arrow pointing towards it from the 'FileBrowser' icon in the utilities list.

**Bottom Left Callout:**

- 사용자가 할당 및 생성한 세션 정보를 확인할 수 있고, 개발도구 실행 및 종료를 할 수 있습니다.

# 활용

A screenshot of a web browser window. The address bar shows the URL `172.18.69.106:10447`. The main content area displays a terminal session. The user has run the command `python`, which outputs the Python version information:

```
work@10aaef4ed065:~$ who
work@10aaef4ed065:~$ python
Python 3.6.9 (default, Nov  7 2019, 10:44:02)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

## ❖ Linux 터미널

A screenshot of a web browser window. The address bar shows the URL `172.18.69.106:10643/terminals/1`. The main content area displays a terminal session. The user has run the command `ls`, which lists the contents of the current directory:

```
work@10aaef4ed065:~$ ls
id_container
work@10aaef4ed065:~$
```



## Analyses about the COVID-19 virus (예제)

- ❖ 사이트 정보 <https://github.com/twiecki/covid19>
- ❖ Docker, 주피터 등을 활용할 수 있는 좋은 예제
- ❖ 교내 클라우드에서 수행
  - covid-19 맵 응용을 교내클라우드에서 수행해보기  
<https://opensource.com/article/20/4/python-map-covid-19>
  - Colab 코드를 교내 클라우드에서 수행해보기
  - **Algorithmic Trading Strategy Using Python**  
(<https://www.youtube.com/watch?v=SEQbb8w7VTw>)
  - 골프자세교정AI만들기  
(<https://www.youtube.com/watch?v=PeuE0nLQqzU>)
  - **A Machine Learning Stock Trading Strategy Using Python**  
(<https://www.youtube.com/watch?v=Whj0u6T6nBg>)

# 마크업 언어(MARKUP LANGUAGES)

# 마크업 언어(markup language)

- ❖ 마크업 언어(markup 言語, markup language)는 태그(mark) 등을 이용하여 데이터의 구조를 명기하는 언어의 한 가지이다. (위키정의)
- ❖ Markdown 언어
  - <http://nolboo.github.io/blog/2014/04/15/how-to-use-markdown/>
  - 온라인상에 메시지를 전달하기 위한 readme 파일을 작성하는데 많이 이용된다. 예로 github 사이트의 readme.md 파일
  - 2004년 John Gruber가 제안 (<http://daringfireball.net/projects/markdown/>)
  - Html보다 간단하고, md파일을 html파일로 변경하는 툴이 다수 존재한다.
  - <http://daringfireball.net/projects/markdown/syntax>
    - ✓ 제안자가 문법을 자세히 설명해 놓은 곳
  - <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>
    - ✓ Cheatsheet의 내용만 살펴보아도 쉽게 문서를 작성할 수 있다.

# Markdown 문서

<http://markdown-here.com/livedemo.html>

The screenshot shows a web browser window with the URL <http://markdown-here.com/livedemo.html> in the address bar. The page title is "Markdown Here".

The interface consists of two main sections:

- Try it out...**: This section contains a text input area with placeholder text: "This is just a sample. You can play around with your own text right here." Below the input area, there's a "Markdown" section with a horizontal line separator. It contains the following text:

...is really just ordinary text, *plain and simple*. How is it good for you?

  - You just **type naturally**, and the result looks good.
  - You **don't have to worry** about clicking formatting buttons.
  - Or fiddling with indentation. (Two spaces is all you need.)

To see what else you can do with Markdown (including **tables**, **images**, **numbered lists**, and more) take a look at the [\[Cheatsheet\]](#)[1]. And then try it out by typing in this box!

[1]: <https://github.com/adam-p/markdown-here/wiki/Markdown-Here-Cheatsheet>
- ...and see what happens!**: This section shows the resulting HTML output generated by the Markdown parser. The text is displayed in a larger font and includes the following HTML elements:

This is just a sample. You can play around with your own text right here.

**Markdown**

...is really just ordinary text, *plain and simple*. How is it good for you?

  - You just **type naturally**, and the result looks good.
  - You **don't have to worry** about clicking formatting buttons.
    - Or fiddling with indentation. (Two spaces is all you need.)

To see what else you can do with Markdown (including **tables**, **images**, **numbered lists**, and more) take a look at the [\[Cheatsheet\]](#). And then try it out by typing in this box!



## 학습내용

- ❖ 주피터 (PC 부터 클라우드까지 일관된 인터페이스)
- ❖ 교내 클라우드 환경 (파이썬, 주피터, 인공지능 라이브러리 지원)
- ❖ Markdown (Markup 언어)
- ❖ Note-taking APP
  - Obsidian (<https://obsidian.md/>)
    - ✓ <https://github.com/ieshreya/Obsidian-Cheat-Sheet>
  - Logseq (<https://logseq.com/>)
    - ✓ <https://cheatography.com/bgrolleman/cheat-sheets/logseq/>