Fastcampus Data Science Extension SCHOOL

API server with Flask

Index

- Jupyter with EC2
- Flask

EC2

- https://aws.amazon.com/ec2/

AWS EC2

- Amazon Elastic Computer Cloud
- rent virtual computers
- Google Compute Engine, Azure Virtual Machines

Install AWS CLI for windows

- 64bit: https://s3.amazonaws.com/aws-cli/AWSCLI64.msi
- 32bit: https://s3.amazonaws.com/aws-cli/AWSCLI32.msi

windows10 built-in SSH

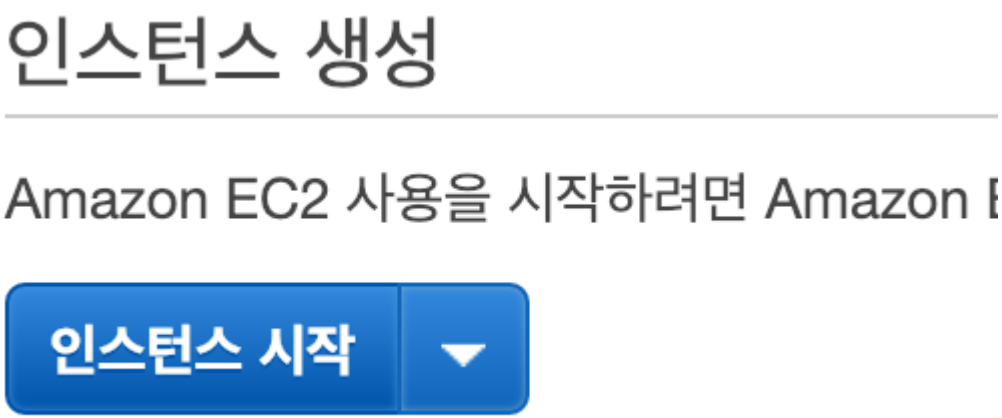
- https://www.howtogeek.com/336775/how-to-enable-and-use-windows-10s-built-in-ssh-commands/

MacOS, Linux

```
$ pip install awscli
```

or

```
$ pip3 install awscli
```


인스턴스 생성
Amazon EC2 사용을 시작하려면 Amazon E
인스턴스 시작

미국 동부 (오하이오)

미국 서부 (캘리포니아 북부)

미국 서부 (오레곤)

아시아 태평양 (뭄바이)

아시아 태평양 (서울)

아시아 태평양 (싱가포르)

아시아 태평양 (시드니)

아시아 태평양 (도쿄)

캐나다 (중부)

EU (프랑크푸르트)

EU (아일랜드)

EU (런던)

EU (파리)

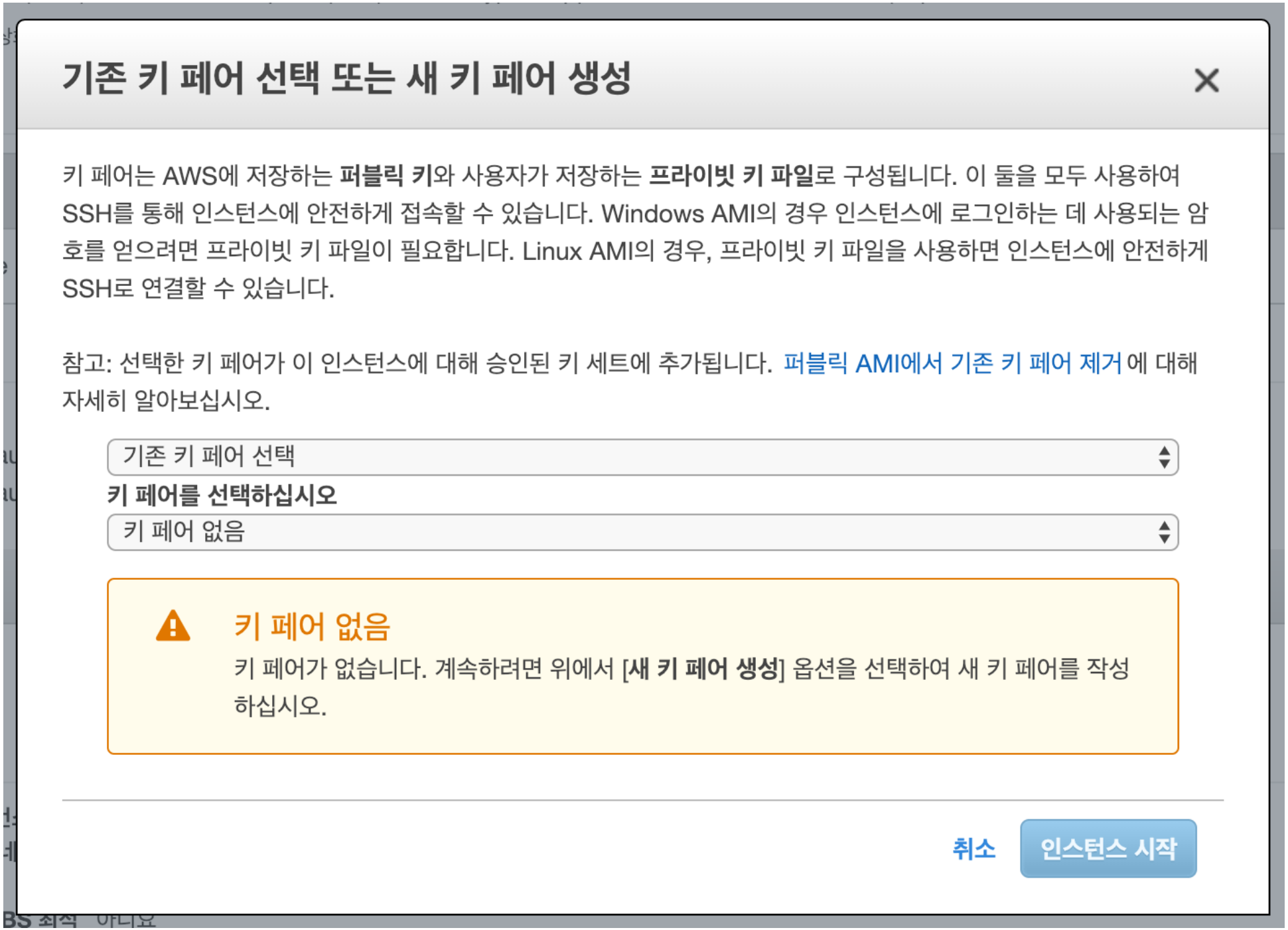
Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-06c43a7df16e8213c

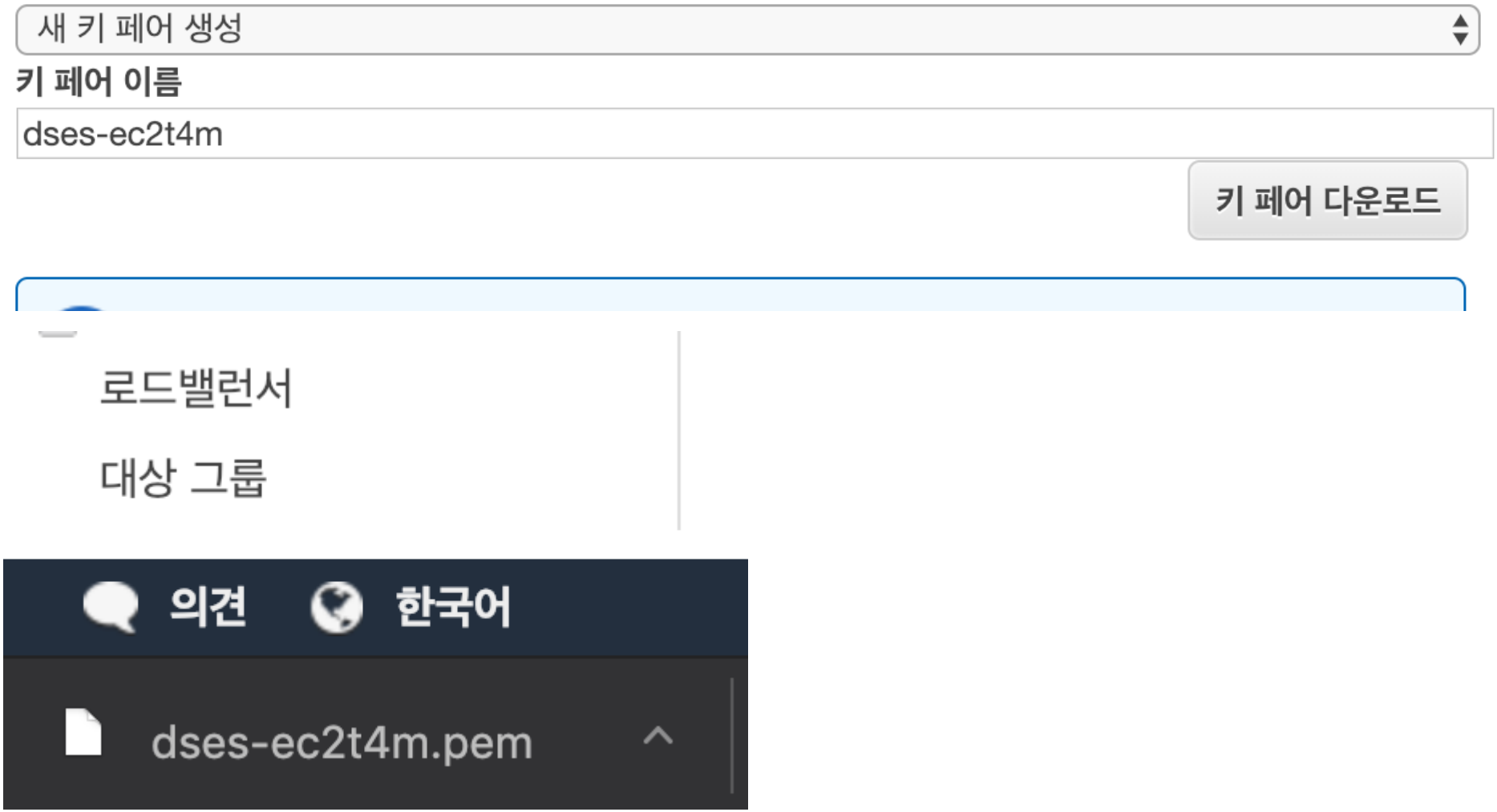
프리 티어 사용 가능

Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

루트 디바이스 유형: ebs　　가상화 유형: hvm

■	General purpose	t2.micro 프리 티어 사용 가능	1	1


기존 키 페어 선택 또는 새 키 페어 생성
키 페어는 AWS에 저장하는 퍼블릭 키와 사용자가 저장하는 프라이빗 키 파일로 구성됩니다. 이 둘을 모두 사용하여 SSH를 통해 인스턴스에 안전하게 접속할 수 있습니다. Windows AMI의 경우 인스턴스에 로그인하는 데 사용되는 암호를 얻으려면 프라이빗 키 파일이 필요합니다. Linux AMI의 경우, 프라이빗 키 파일을 사용하면 인스턴스에 안전하게 SSH로 연결할 수 있습니다.
참고: 선택한 키 페어가 이 인스턴스에 대해 승인된 키 세트에 추가됩니다. 퍼블릭 AMI에서 기존 키 페어 제거 에 대해 자세히 알아보십시오.
기존 키 페어 선택
키 페어를 선택하십시오
키 페어 없음
키 페어 없음
키 페어가 없습니다. 계속하려면 위에서 [새 키 페어 생성] 옵션을 선택하여 새 키 페어를 작성하십시오.
취소
인스턴스 시작


새 키 페어 생성
키 페어 이름
dses-ec2t4m
키 페어 다운로드
로드밸런서
대상 그룹
의견
한국어
dses-ec2t4m.pem

시작 상태

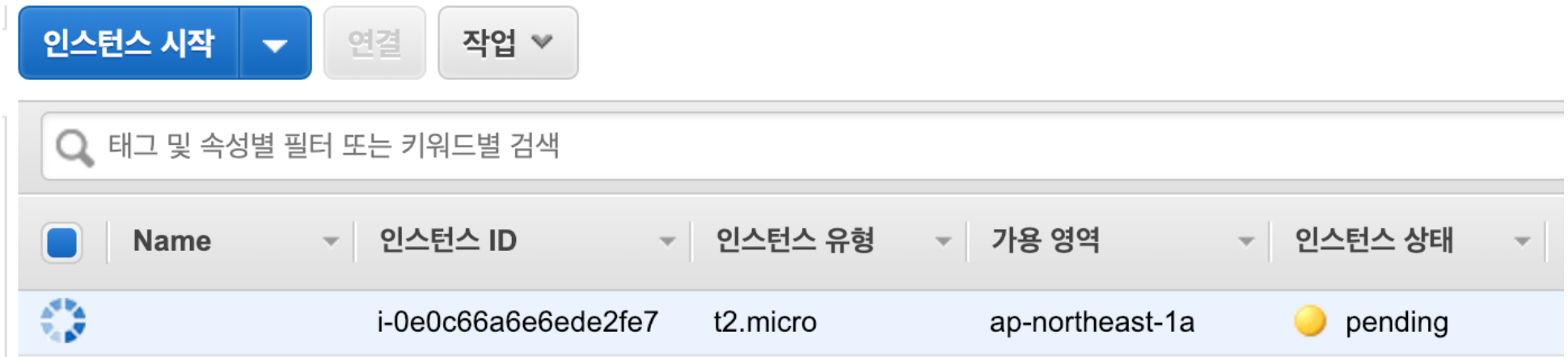
지금 인스턴스를 시작 중입니다.

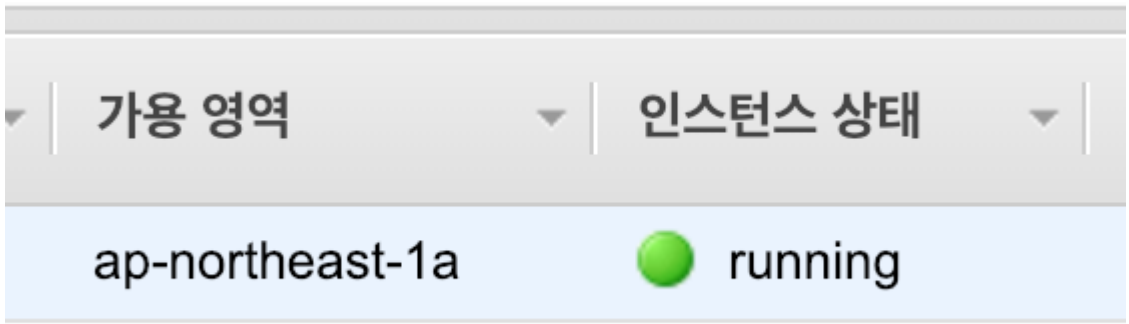
다음 인스턴스 시작 개시: i-0e0c66a6e6ede2fe7 시작 로그 보기

예상 요금 알림 받기

결제 알림 생성 AWS 결제 예상 요금이 사용자가 정의한 금액을 초과하는 경우(예를 들면 프리 티어를

인스턴스에 연결하는 방법



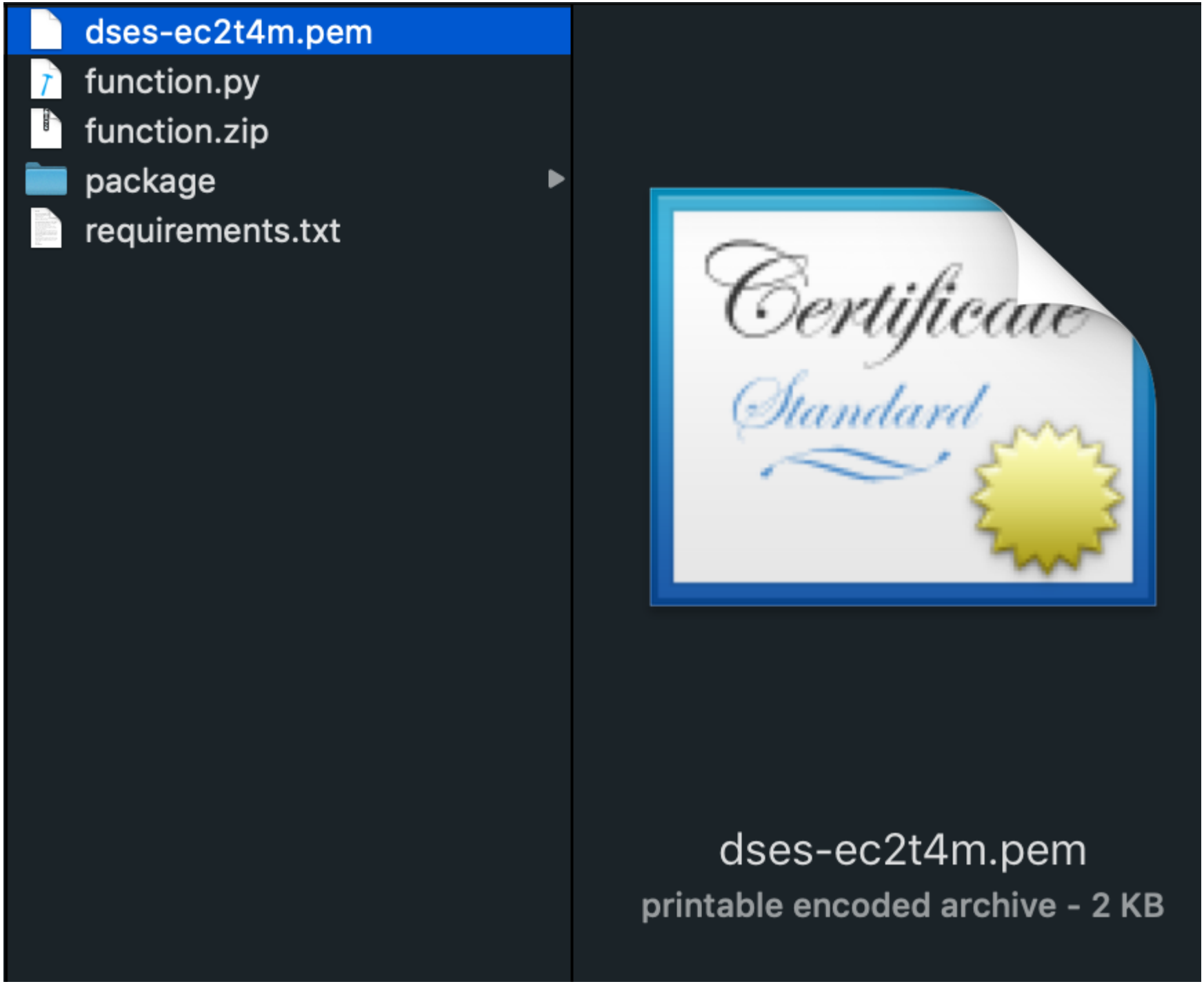


인스턴스: i-0e0c66a6e6ede2fe7 **퍼블릭 DNS:** ec2-13-115-82-39.ap-northeast-1.compute.amazonaws.com

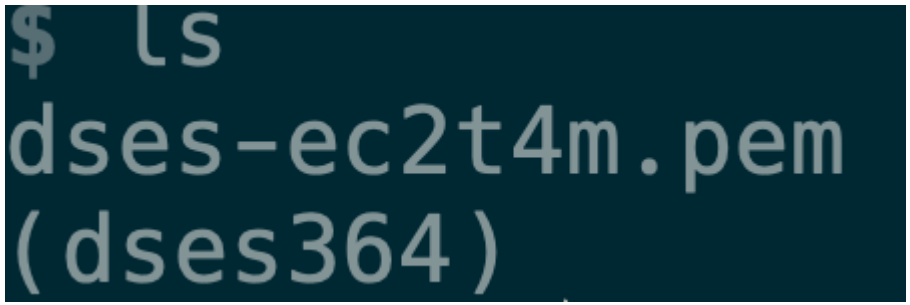
설명 | 상태 검사 | 모니터링 | 태그

인스턴스 ID	i-0e0c66a6e6ede2fe7	퍼블릭 DNS(IPv4)	ec2-13-115-82-39.ap-northeast-1.compute.amazonaws.com
인스턴스 상태	running	IPv4 퍼블릭 IP	13.115.82.39
인스턴스 유형	t2.micro	IPv6 IP	-
탄력적 IP		프라이빗 DNS	ip-172-31-36-135.ap-northeast-1.compute.internal
가용 영역	ap-northeast-1a	프라이빗 IP	172.31.36.135
보안 그룹	launch-wizard-1. 인바운드 규칙 보기. view outbound rules	보조 프라이빗 IP	
예약된 이벤트	예약된 이벤트 없음	VPC ID	vpc-8e3e1be9
AMI ID	ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20180912 (ami-06c43a7df16e8213c)	서브넷 ID	subnet-884bacc0


pute.amazonaws.com
클립보드에 복사
퍼블릭 DNS(IPv4)
ec2-
1.compute.amazonaws.com
IPv4 퍼블릭 IP
13.115.82.39
IPv6 IP
-


dses-ec2t4m.pem
function.py
function.zip
package
requirements.txt
Certificate
Standard
dses-ec2t4m.pem
printable encoded archive - 2 KB


인스턴스에 연결
다음에 연결:
독립 실행형 SSH 클라이언트
현재 웹 브라우저에서 Java SSH 클라이언트에 직접(Java 필요)
인스턴스 액세스 방법:
1. SSH 클라이언트를 개방하십시오. (PuTTY를 사용하여 연결 방법 알아보기)
2. 프라이빗 키 파일(dses-ec2t4m.pem)을 찾습니다. 마법사가 인스턴스를 시작하는 데 사용되는 키를 자동으로 검색합니다.
3. SSH가 작동하려면 키가 공개적으로 표시되지 않아야 합니다. 필요할 경우 이 명령을 사용합니다.
chmod 400 dses-ec2t4m.pem
4. 퍼블릭 DNS을(를) 사용하여 인스턴스에 연결:
ec2-13-115-82-39.ap-northeast-1.compute.amazonaws.com
예:
ssh -i "dses-ec2t4m.pem" ubuntu@ec2-13-115-82-39.ap-northeast-1.compute.amazonaws.com
대부분의 경우 위의 사용자 이름이 맞지만, AMI 사용 지침을 숙지하여 AMI 소유자가 기본 AMI 사용자 이름을 변경하지 않도록 하십시오.
인스턴스에 연결하는 데 도움이 필요한 경우 연결 설명서을(를) 참조하십시오.
닫기




```
$ chmod 400 dses-ec2t4m.pem
```


```
$ ssh -i "dses-ec2t4m.pem" ubuntu@<your-public-domain>
```

```
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-1067-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-36-135:~$ clear
ubuntu@ip-172-31-36-135:~$
```

```
Resolving repo.continuum.io (repo.continuum.io)... 104.16
Connecting to repo.continuum.io (repo.continuum.io)|104.1
HTTP request sent, awaiting response... 200 OK
Length: 523283080 (499M) [application/x-sh]
Saving to: ‘Anaconda3-4.4.0-Linux-x86_64.sh’

Anaconda3-4.4.0-Linux-x86_64.  51%[=======================
```

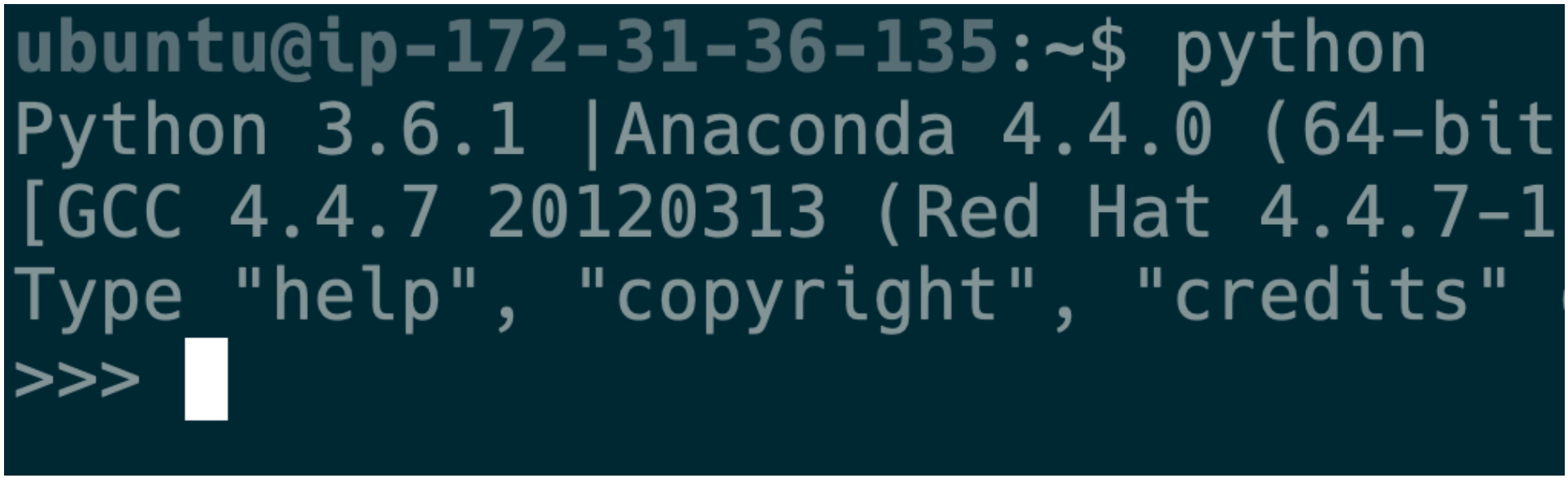
```
$ wget https://repo.continuum.io/archive/Anaconda3-4.4.0-Linux-x86_64.sh
```

```
$ bash Anaconda3-4.4.0-Linux-x86_64.sh
```

```
$ vi ~/.bashrc
```

```
export PATH="/home/ec2-user/anaconda3/bin:$PATH"
```
<= add

```
ubuntu@ip-172-31-36-135:~$ vi .bashrc
ubuntu@ip-172-31-36-135:~$ source ~/.bashrc
ubuntu@ip-172-31-36-135:~$ which python
/home/ubuntu/anaconda3/bin/python
ubuntu@ip-172-31-36-135:~$
```


ubuntu@ip-172-31-36-135:~$ python
Python 3.6.1 |Anaconda 4.4.0 (64-bit
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1
Type "help", "copyright", "credits"
>>>

```
$ ipython
```

```
In [1]: from IPython.lib import passwd

In [2]: passwd()
Enter password:
Verify password:
```

copy **SHAstring!!!**

```
$ jupyter notebook --generate-config
```

```
$ mkdir certs && cd certs
```

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -
keyout mycert.pem -out mycert.pem
```

```
Generating a 1024 bit RSA private key
.++++++
.....................++++++
writing new private key to 'mycert.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:KR
State or Province Name (full name) [Some-State]:Seoul
Locality Name (eg, city) []:Seoul
Organization Name (eg, company) [Internet Widgits Pty Ltd]:NONE
Organizational Unit Name (eg, section) []:NONE
Common Name (e.g. server FQDN or YOUR name) []:dses
Email Address []:email
ubuntu@ip-172-31-36-135:~/certs$
```

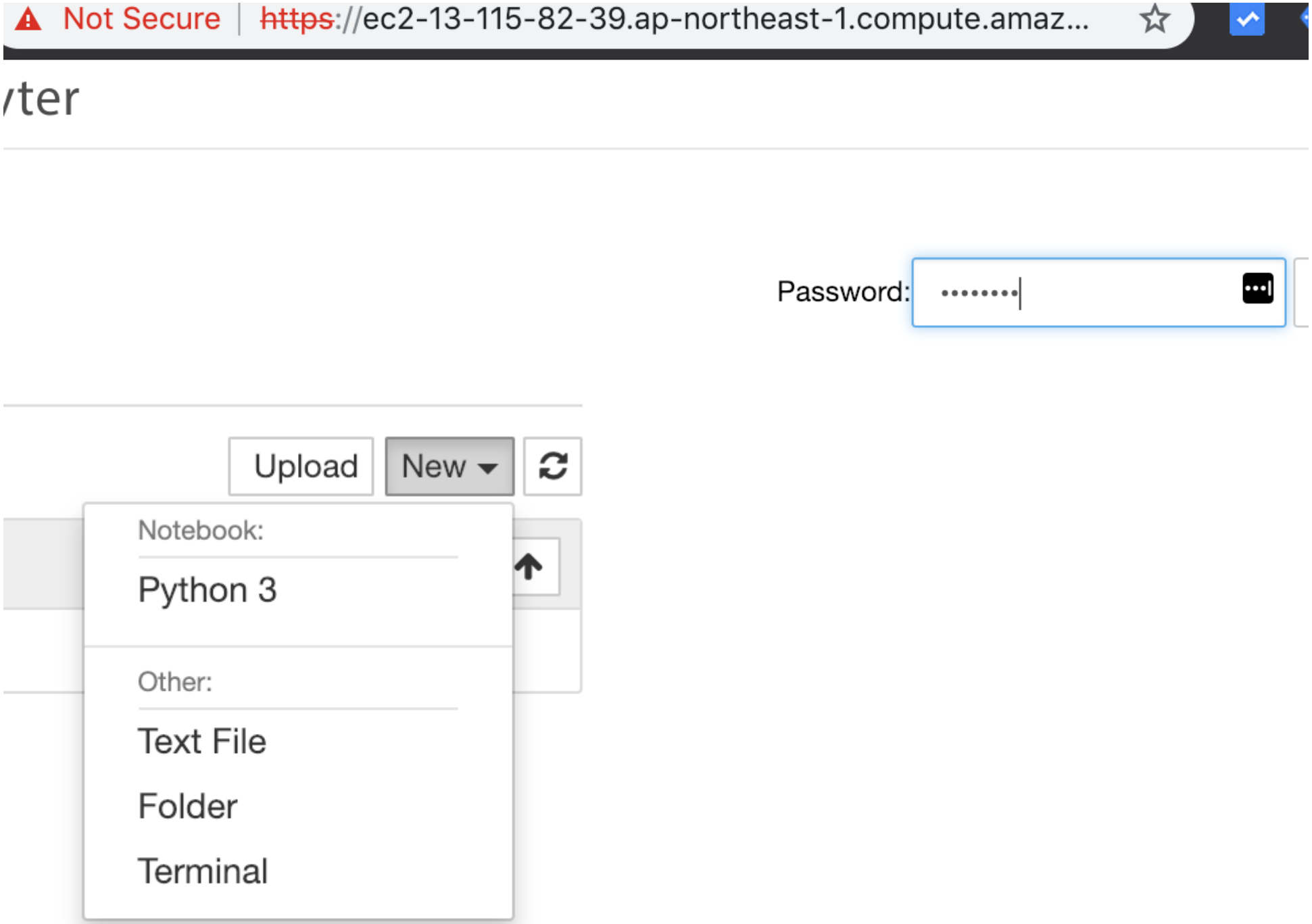
```
$ vi .jupyter/jupyter_notebook_config.py
```

```
c = get_config()

# Kernel config
c.IPKernelApp.pylab = 'inline'

# Notebook config
c.NotebookApp.certfile = u'/home/ubuntu/certs/mycert.pem' #locat
c.NotebookApp.ip = '*'
c.NotebookApp.open_browser = False
c.NotebookApp.password = u'sha1:...'

c.NotebookApp.port = 8888
```

```
:wq
```

```
$ mkdir dev/pynbs && cd dev/pynbs
$ jupyter notebook
```

```
https://<your-public-domain>:8888/
```


Not Secure
https://ec2-13-115-82-39.ap-northeast-1.compute.amaz...
ter
Password:
Upload
New
Notebook:
Python 3
Other:
Text File
Folder
Terminal

Flask

Web Framework

- 웹서비스를 제공하기 위해 필요한 기능들을 모아둔 클래스와 라이브러리의 모임

Web Frameworks built with python

- Full-stack
  - Django
  - Pyramid
  - Web2py
- Microframework
  - **Flask**
  - Bottle
- Async
  - Tornado
  - Sanic

Flask - start app

```
$ pip install flask
```

```
from flask import Flask


app = Flask(__name__)

@app.route('/')
def index():
        return 'hello world!'

if __name__ == '__main__':
        app.run(host='0.0.0.0', port=8080, debug=True)
```

```
$ python server.py
```

c9.io

```
from flask import Flask
import os

app = Flask(__name__)

@app.route('/')
def index():
    return 'hello world!'

app.run(host=os.getenv('IP', '0.0.0.0'),port=int(os.getenv('PORT
```

Flask - route

```
from flask import Flask


app = Flask(__name__)

@app.route('/')
def index():
        return 'hello'

@app.route('/about')
def about():
        return 'about'

if __name__ == '__main__':
        app.run(host='0.0.0.0', port=8080, debug=True)
```

Flask - render

```
from flask import Flask, render_template


app = Flask(__name__)

@app.route('/')
def index(name=None):
        return render_template('index.html', name=name)

@app.route('/about')
def about(name=None):
        return render_template('about.html', name=name)

if __name__ == '__main__':
        app.run(host='0.0.0.0', port=8080, debug=True)
```

Flask - render

```
/
  server.py
  /templates
    index.html
    about.html
```

Flask with BeautifulSoup

```
from bs import BeautifulSoup

def index():
    ...
    .(some code).
    ...
```

route with querystring, path

```
@app.route('/user/<string:name>')
def user(name=None):
    return render_template('user.html', msg=name)
```

```
@app.route('/users')
def users():
    querystring = request.args
    return render_template('users.html', rows=querystring)
```

form with sqlite

```
@app.route('/movies', methods=['GET', 'POST'])
def movies():
    if request.method == 'GET':
        conn = lite.connect('./data/data.db')
        conn.row_factory = lite.Row

        cur = conn.cursor()
        cur.execute("SELECT * FROM Movies;")

        rows = cur.fetchall()
        conn.close()
        return render_template('movies.html', rows=rows)

...
```

```
...

    elif request.method == 'POST':
        try:
            input_name = request.form["movie-name"]
            input_year = request.form["movie-year"]
            input_studio = request.form["movie-studio"]
            with lite.connect('./data/data.db') as conn:
                cur = conn.cursor()
                cur.execute("""
                        INSERT INTO Movies(name, year, studio)
                        VALUES(?,?,?);
                        """,
                        (input_name, input_year, input_studio))
                conn.commit()
                msg = "Success"
        except:
            conn.rollback()
            msg = "Failed to Save"
        finally:
            return render_template('movies.html', msg=msg)
```

simple api server with flask, mlab

```
mongo_uri = "mongodb://strongadmin:admin1234@ds135844.mlab.com:3

@app.route('/api/v1/item')
def api():
    client = MongoClient(mongo_uri)
    db = client.mydbinstance
    items = db.bigbang
    try:
        query = {}
        projection= {
                "_id":0,
                "title":1,
                "item":1,
                }
        result = list(items.find(query, projection))
    except:
        result = "failed"
    finally:
        return jsonify({"items":result})
```

EC2 with flask

```
$ git clone ~~
```

```
if __name__=='__main__':
    app.run(host='0.0.0.0', port=80, debug=False)
```

```
$ python server.py
```