

# Deep Learning on AWS

---

AWS 환경에서 딥 러닝 시작하기

안성하

동서대학교 메카트로닉스융합공학부

[seongha.ahn4@gmail.com](mailto:seongha.ahn4@gmail.com)

- **목적**

이 설명서는 Linux와 서버에 대한 기초 지식을 가지고 있는 개인 사용자를 대상으로 합니다. AWS에서 제공하는 제품을 이용하여 딥 러닝 모델을 학습시키기 위한 서버를 구성하고, 서버 내에서 기본적인 개발환경을 구축하는 방법을 설명합니다.

- **이 설명서에서 다루는 내용**

- 사용할 AWS 제품(EC2)을 이해하기 위한 용어와 간단한 개념
- GPU 연산을 지원하는, 개인적인 용도의 딥 러닝용 서버를 구축하고 설정하기 위해 필수적인 설명

- **이 설명서에서 다루지 않는 내용**

- AWS 제품에 대한 자세한 설명 → References 참조
- SSH 접속을 제외한, 서버 내 작업을 위한 Linux 명령어 설명
- EC2와 함께 사용할 수 있는 AWS 제품의 활용

# Agenda

3

- AWS 개요
  - 프리 티어 활용하여 제품 무료로 이용하기
- AWS 환경에서 딥 러닝 개발하기
  - EC2
  - EBS 볼륨을 이용하여 인스턴스 백업하기
  - Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기
- Supplementary
  - Demonstration
- References

## AWS란?

- Amazon Web Services (AWS)
- 시장 점유율 1위 클라우드 플랫폼
- 162개의 서비스를 통해 IT 인프라 제공

## 주요 제품

- 스토리지 (S3)
- 가상서버 (EC2)
- 데이터베이스 (DynamoDB, Aurora, etc.)

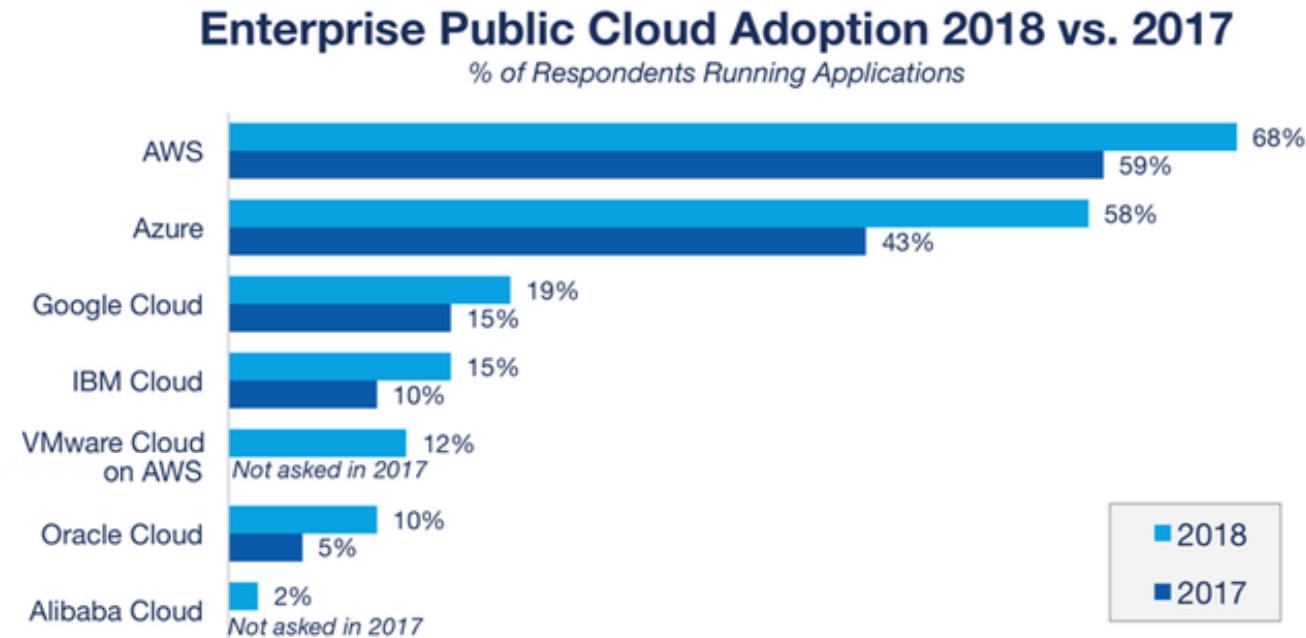


Figure 1 : 클라우드 컴퓨팅 기업별 점유율 [1](#)

## 프리 티어<sup>2</sup>를 활용하여 제품 무료로 이용하기

- 프리 티어 : 60가지 이상의 제품을 무료로 사용해 볼 수 있음
  - 프리 티어 유형에 따라 사용 기간 달라짐

### **프리 티어 활용 시 주의!**

- 모든 제품이 무료로 제공되는 것이 아니므로,  
사용하려는 제품이 프리 티어로 사용 가능한지 잘 확인할 것
- 기간과 무관하게 제품별 사용 시간, 용량, 옵션에 한도 있으므로 서비스 이용 시 주의

# AWS 환경에서 딥 러닝 개발하기

6

## AWS 환경에서 딥 러닝 시작하기

- AWS 제품을 이용한 딥 러닝 개발 환경 구축
  - GPU 컴퓨팅을 사용할 수 없는 환경에서 딥 러닝 개발 시 활용 가능
  - AWS EC2(가상서버) 제품을 중심으로 사용

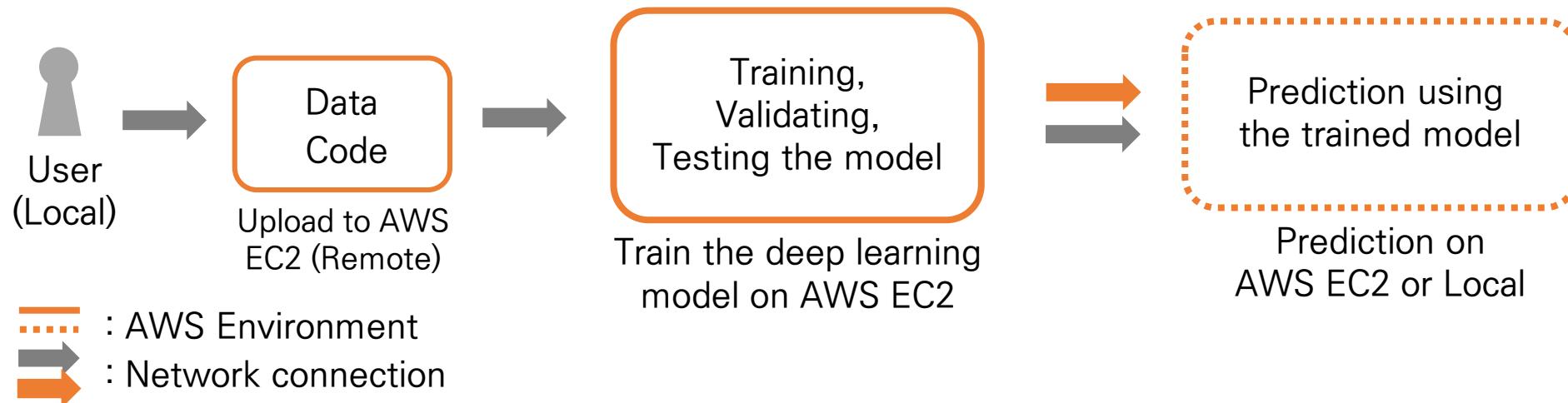


Figure 2 : AWS EC2 제품을 이용한 딥 러닝 개발 workflow

# AWS 환경에서 딥 러닝 개발하기

7

## EC2<sup>3</sup>

- Amazon Elastic Compute Cloud (EC2)
  - 원하는 소프트웨어 구성(AMI\*)의 가상서버(인스턴스) 구축을 통해 클라우드 컴퓨팅 환경 제공
  - GPU, CPU, 메모리 등의 사양을 선택하여 인스턴스 구성
  - 프리 티어 활용 시, GPU 사용 불가하여 딥 러닝용 인스턴스 구축에 적합하지 않음 : 스팟 인스턴스 사용 고려
- EC2 구매옵션<sup>4</sup> : ‘스팟 인스턴스<sup>5</sup>’ 활용하여 저렴하게 GPU 사용하기
  - 스팟 인스턴스 : 남는 자원을 경매 형식으로 판매. On-demand 사용 대비 최대 90% 할인된 가격으로 구매할 수 있으므로 저렴하게 GPU 인스턴스를 이용할 수 있음
  - 단기적으로 수요가 많을 때 활용
  - 수시로 가격 변동, AWS의 자원 현황에 따라 인스턴스가 종료되는 경우<sup>6</sup> 발생 가능성 존재 : 백업 필요

\*AMI : Amazon Machine Image, 소프트웨어 구성이 기재된 템플릿. 다양한 운영체제, 사용 목적에 따라 AMI 선택할 수 있음

# AWS 환경에서 딥 러닝 개발하기

8

## EBS 볼륨을 이용하여 인스턴스 백업하기

- Amazon Elastic Block Store(EBS)<sup>7</sup>
  - AWS의 스토리지 제품 중 하나<sup>8</sup>
  - 인스턴스 생성 시 필수적으로 일정 이상의 스토리지를 요구 : EBS 볼륨을 생성
  - 인스턴스에서 하드디스크 역할을 수행
- EBS 볼륨을 이용하여 스팟 인스턴스 종료에 대비하기
  - 볼륨을 캡처하는 ‘스냅샷<sup>9</sup>’ 기능을 제공
  - 스냅샷으로부터 ‘AMI’를 생성하여, 캡처한 볼륨과 동일한 볼륨을 갖춘 인스턴스 시작
  - 원본 인스턴스 종료 후 볼륨이 삭제되지 않도록 설정 필요
  - 스팟 인스턴스가 모델 학습 중 종료된 경우, 볼륨에 남아있는 모델로 이어서 학습 가능\*

\*TensorFlow, Keras의 모델 저장과 불러오기에 관한 내용은 [Supplementary G](#),  
인스턴스 종료 시 학습 진행상황 확인은 [Supplementary H](#) 참조

# AWS 환경에서 딥 러닝 개발하기

9

Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 0. 개요
  - EC2 스팟 인스턴스 사용하여 저렴하게 GPU 인스턴스 구축
  - Deep Learning AMI 사용을 통해 별도의 설정작업 없이 개발환경 사용
  - 인스턴스 직접 접속없이 개발환경 사용을 위한 Jupyter notebook 원격 접속 설정
  - 인스턴스 종료 후 데이터를 백업하여 사용하기

# AWS 환경에서 딥 러닝 개발하기

10

## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

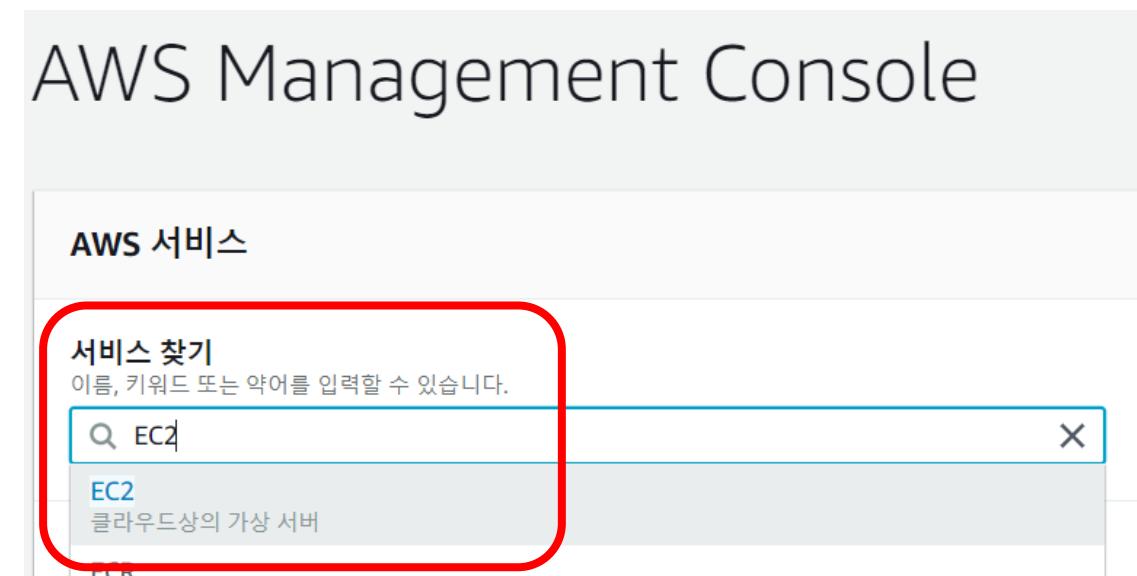
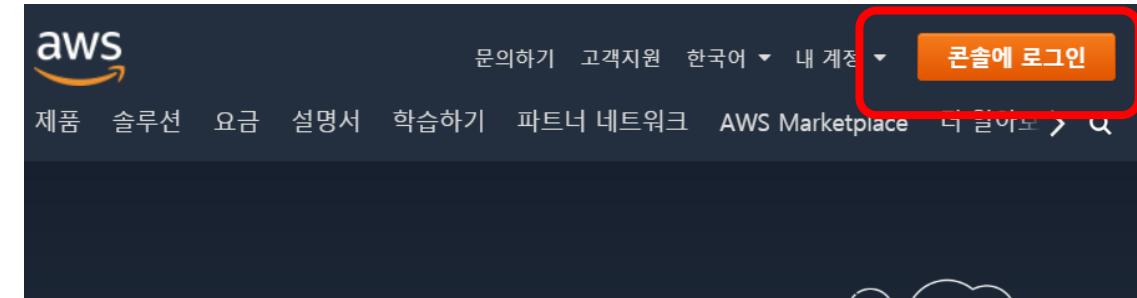
- 1. 모델 Training, Testing, Prediction을 수행할 수 있는 GPU 서버 구축\*

(1) AWS 계정 생성, 결제 방법 (결제 가능한 카드 등록) 설정

(2) 페이지 우측 상단 ‘콘솔에 로그인’ 클릭

(3) AWS Management Console의 서비스 찾기에서

‘EC2’ 입력 후 Enter



# AWS 환경에서 딥 러닝 개발하기

11

## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 1. 모델 Training, Testing, Prediction을 수행할 수 있는 GPU 서버 구축

### (4) 우측 상단의 ‘리전’ 선택 후, ‘AMI’ 항목으로 이동

리전이란 서비스를 제공할 AWS의 서버가 위치한

지역을 의미. 미국 지역의 리전 사용 시 서울 리전 대비

가격이 저렴함. (튜토리얼에서는 ‘버지니아 북부’ 리전 선택)

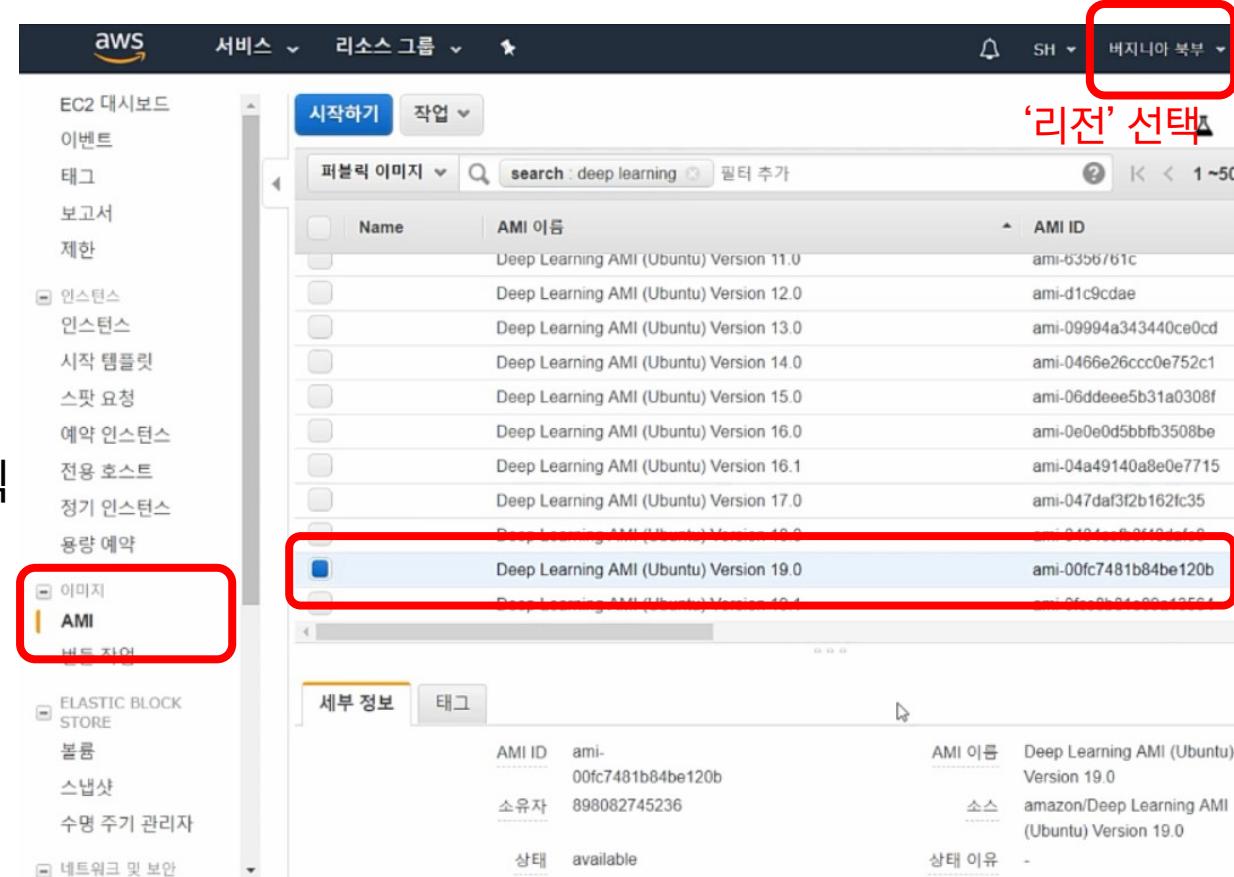
### (5) ‘퍼블릭 이미지’–‘deep learning’ 검색, ‘Deep Learning

AMI (Ubuntu) Version 19.0<sup>10</sup> 선택 후 ‘시작하기’ 버튼 클릭

리눅스에서 GPU 연산을 위해서는 NVIDIA 드라이버와

CUDA, cuDNN 등 인터페이스를 설치하는 과정이 필요.

‘Deep Learning AMI’는 이러한 과정없이 GPU 연산을 사용할 수 있도록 사전에 설치되어 있음



# AWS 환경에서 딥 러닝 개발하기

12

## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 1. 모델 Training, Testing, Prediction을 수행할 수 있는 GPU 서버 구축

### (6) 인스턴스 유형 선택 : 'GPU instances' 사용

'필터링 기준 : GPU instances'로 검색하여 필요한 스펙에 해당하는 유형을 선택.  
(튜토리얼에서는 'p2.xlarge' 사용)

단계 2: 인스턴스 유형 선택

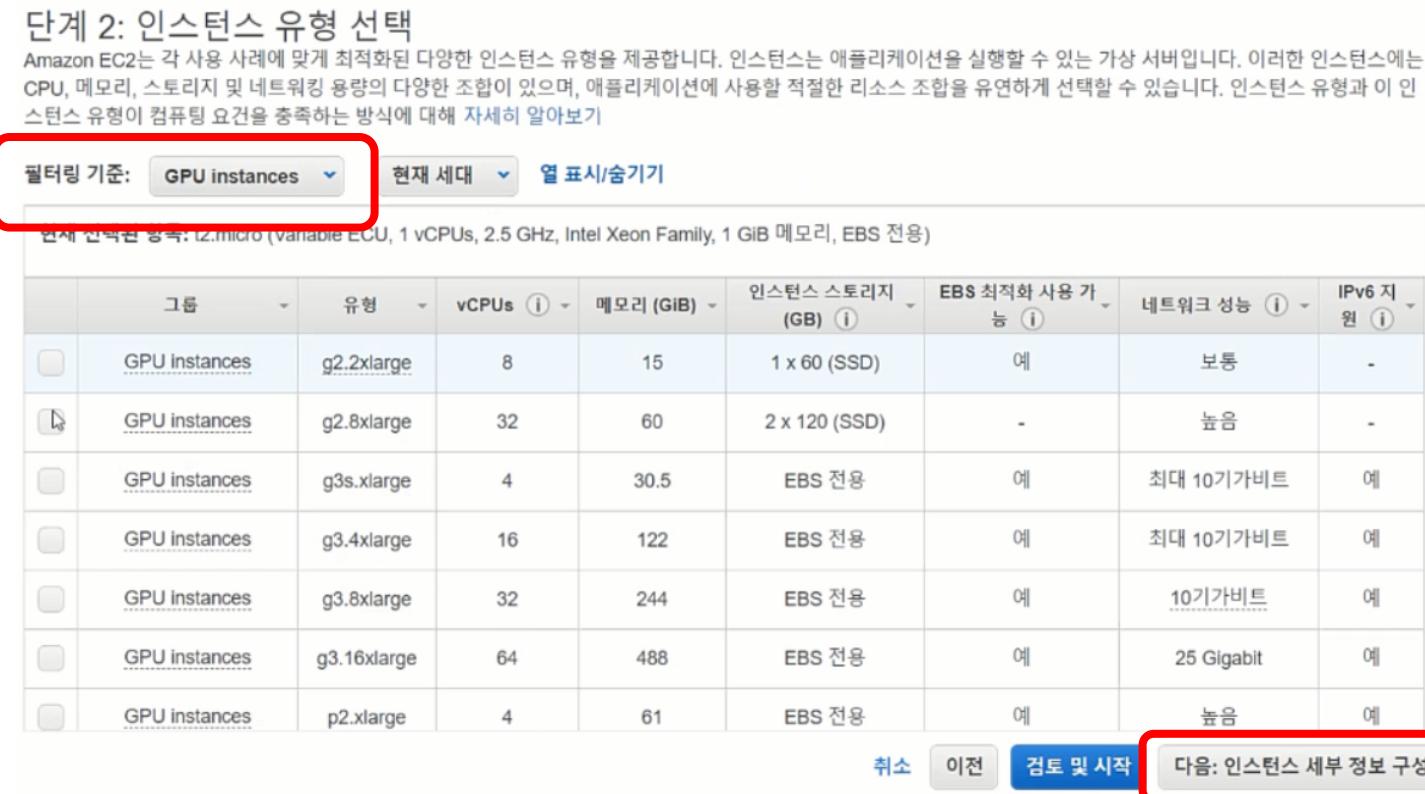
Amazon EC2는 각 사용 사례에 맞게 최적화된 다양한 인스턴스 유형을 제공합니다. 인스턴스는 애플리케이션을 실행할 수 있는 가상 서버입니다. 이러한 인스턴스에는 CPU, 메모리, 스토리지 및 네트워킹 용량의 다양한 조합이 있으며, 애플리케이션에 사용할 적절한 리소스 조합을 유연하게 선택할 수 있습니다. 인스턴스 유형과 이 인스턴스 유형이 컴퓨팅 요구를 충족하는 방식에 대해 자세히 알아보기

필터링 기준: GPU instances 현재 세대 열 표시/숨기기

현재 선택된 항목: t2.micro (variable ECU, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB 메모리, EBS 전용)

그룹	유형	vCPUs	메모리 (GiB)	인스턴스 스토리지 (GB)	EBS 최적화 사용 가능	네트워크 성능	IPv6 지원	
	GPU instances	g2.2xlarge	8	15	1 x 60 (SSD)	예	보통	-
	GPU instances	g2.8xlarge	32	60	2 x 120 (SSD)	-	높음	-
	GPU instances	g3s.xlarge	4	30.5	EBS 전용	예	최대 10기가비트	예
	GPU instances	g3.4xlarge	16	122	EBS 전용	예	최대 10기가비트	예
	GPU instances	g3.8xlarge	32	244	EBS 전용	예	10기가비트	예
	GPU instances	g3.16xlarge	64	488	EBS 전용	예	25 Gigabit	예
	GPU instances	p2.xlarge	4	61	EBS 전용	예	높음	예

취소 이전 다음: 인스턴스 세부 정보 구성



# AWS 환경에서 딥 러닝 개발하기

13

## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 1. 모델 Training, Testing, Prediction을 수행할 수 있는 GPU 서버 구축

### (8) 인스턴스 세부 정보 구성 : 스팟 인스턴스 요청

‘구매옵션’ 항목에서 ‘스팟 인스턴스 요청’ 박스에 체크

### (9) 하단 오른쪽 ‘다음: 스토리지 추가’ 버튼 클릭

단계 3: 인스턴스 세부 정보 구성

요구 사항에 적합하게 인스턴스를 구성합니다. 동일한 AMI의 여러 인스턴스를 시작하고 스팟 인스턴스를 요청하여 보다 저렴한 요금을 활용하며 역할을 할당하는 등 다양한 기능을 사용할 수 있습니다.

인스턴스 개수	1	Auto Scaling 그룹 시작	
구매 옵션	<input checked="" type="checkbox"/> 스팟 인스턴스 요청		
현재 가격	가용 영역	현재 가격	
	us-east-1a	\$0.3602	
	us-east-1b	\$0.3649	
	us-east-1c	\$0.379	
	us-east-1d	\$0.3602	
	us-east-1e	\$0.3582	
	us-east-1f	\$0.900	
최고 가격	\$ 예: 0.045 = 4.5센트/시간(선택 사항)		
연속 요청	<input type="checkbox"/> 연속 요청		
시작 그룹	(선택 사항)		
요청 유효 시작 시간:	항상 편집		
요청 유효 종료 시간:	항상 편집		
취소	이전	검토 및 시작	다음: 스토리지 추가

# AWS 환경에서 딥 러닝 개발하기

14

## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 1. 모델 Training, Testing, Prediction을 수행할 수 있는 GPU 서버 구축

### (10) 스토리지 추가 : '종료 시 삭제' 박스 체크 해제

인스턴스 생성 시 필수적으로 생성되는 EBS 볼륨이  
인스턴스 종료 후, 남아있는 볼륨을 이용하여  
작업환경과 종료된 인스턴스의 데이터를 유지하여  
사용 가능. (볼륨을 이용한 데이터 유지와 사용은  
[5](#)에서 설명)

#### 단계 4: 스토리지 추가

인스턴스가 다음 스토리지 디바이스 설정으로 시작됩니다. 추가 EBS 볼륨 및 인스턴스 스토어 볼륨을 인스턴스에 연결하거나 루트 볼륨의 설정을 편집할 수 있습니다. 인스턴스를 시작한 후 추가 EBS 볼륨을 연결할 수도 있지만, 인스턴스 스토어 볼륨은 연결할 수 없습니다. Amazon EC2의 스토리지 옵션에 대해 자세히 알아보기.



### (11) '다음: 태그 추가', '다음:보안그룹 구성' 버튼 클릭

프리 티어 사용 가능 고객은 최대 30GB의 EBS 범용(SSD) 또는 마그네틱 스토리지를 사용할 수 있습니다. 프리 티어 자격 및 사용량 제한에 대해 자세히 알아보기

취소 이전 검토 및 시작 다음: 태그 추가

# AWS 환경에서 딥 러닝 개발하기

15

## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 1. 모델 Training, Testing, Prediction을 수행할 수 있는 GPU 서버 구축

### (12) 보안 그룹 구성 : 포트 22, 8888 개방

새 보안 그룹을 생성하여 원하는 이름과 설명을 입력하고, ‘규칙 추가’ 버튼을 클릭하여 규칙을 추가  
추가된 규칙을 ‘사용자 지정 TCP’, ‘TCP’, ‘8888’,  
‘〈인스턴스로 접속하려는 사용자의 퍼블릭 IP〉’를 차례로 입력.  
(튜토리얼에서는 모든 IP에서 접근할 수 있도록 설정)

#### 단계 6: 보안 그룹 구성

보안 그룹은 인스턴스에 대한 트래픽을 제어하는 방화벽 규칙 세트입니다. 이 페이지에서는 특정 트래픽을 인스턴스에 도달하도록 허용할 규칙을 주를 들면 웹 서버를 설정하여 인터넷 트래픽을 인스턴스에 도달하도록 허용하려는 경우 HTTP 및 HTTPS 트래픽에 대한 무제한 액세스를 허용하는 보안 그룹을 생성하거나 아래에 나와 있는 기존 보안 그룹 중에서 선택할 수 있습니다. Amazon EC2 보안 그룹에 대해 자세히 알아보기

보안 그룹 할당:  새 보안 그룹 생성  
 기존 보안 그룹 선택

보안 그룹 이름: test1  
설명: test

유형	프로토콜	포트 범위	소스
SSH	TCP	22	사용자 지정 0.0.0.0/0
사용자 지정 T	TCP	8888	사용자 지정 0.0.0.0/0

규칙 추가

**경고**  
소스가 0.0.0.0/0인 규칙은 모든 IP 주소에서 인스턴스에 액세스하도록 허용합니다. 알려진 IP 주소의 액세스만 허용하도록 보안 그룹을 합니다.

### (13) ‘검토 및 시작’ 버튼 클릭

취소 이전 검토 및 시작

# AWS 환경에서 딥 러닝 개발하기

16

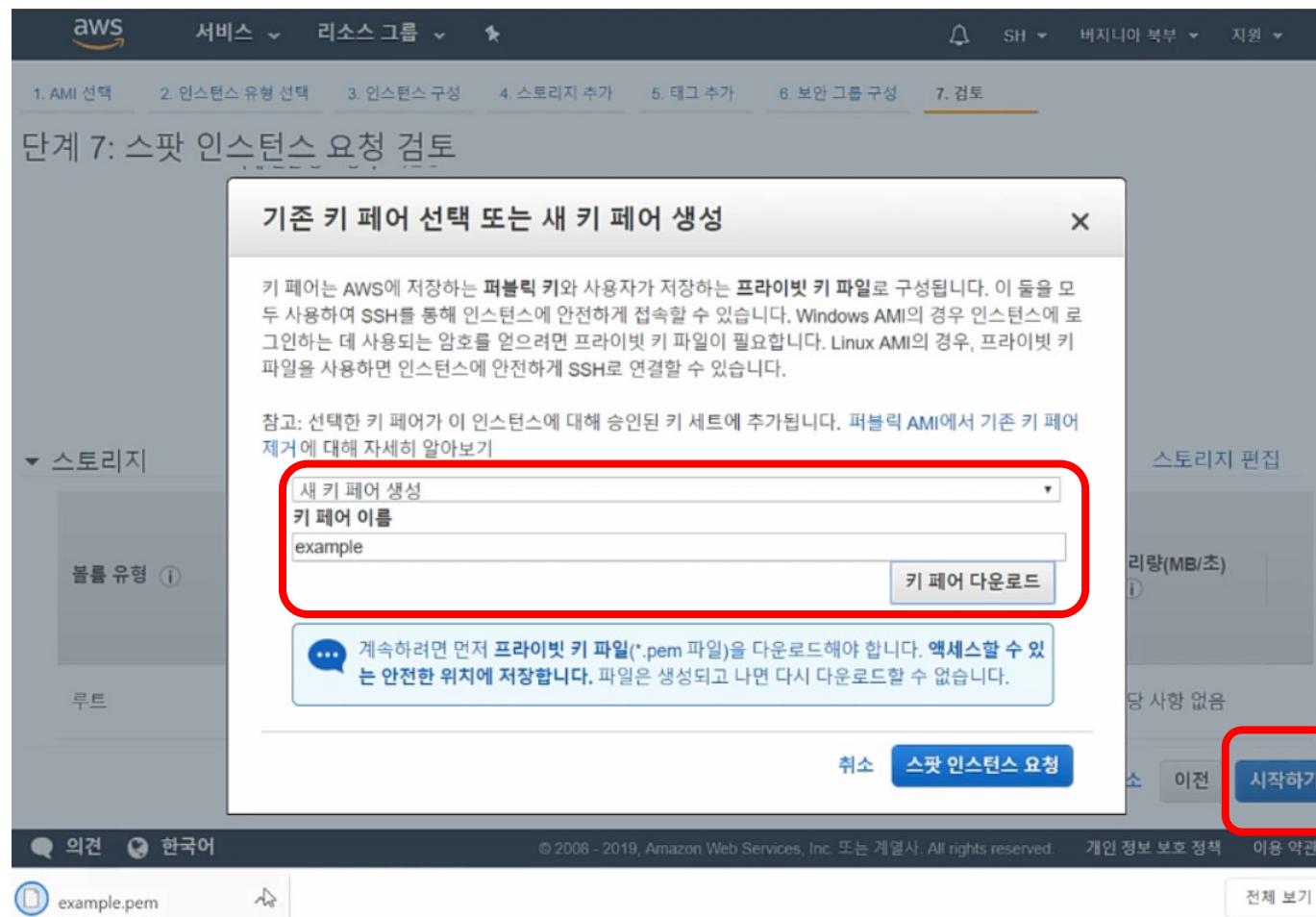
## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 1. 모델 Training, Testing, Prediction을 수행할 수 있는 GPU 서버 구축

### (14) 스팟 인스턴스 요청 검토: 키 페어 설정

앞에서 설정한 내용 확인 후, ‘시작하기’ 버튼 클릭.  
‘기존 키 페어 선택 또는 새 키 페어 생성’ 창에서 ‘새 키 페어 생성’ 선택, 키 페어 이름 입력 후 ‘키 페어 다운로드’ 버튼 클릭. 이후 SSH접속을 위해 키 페어가 저장된 위치를 확인.

### (15) ‘스팟 인스턴스 요청’ 버튼 클릭



# AWS 환경에서 딥 러닝 개발하기

17

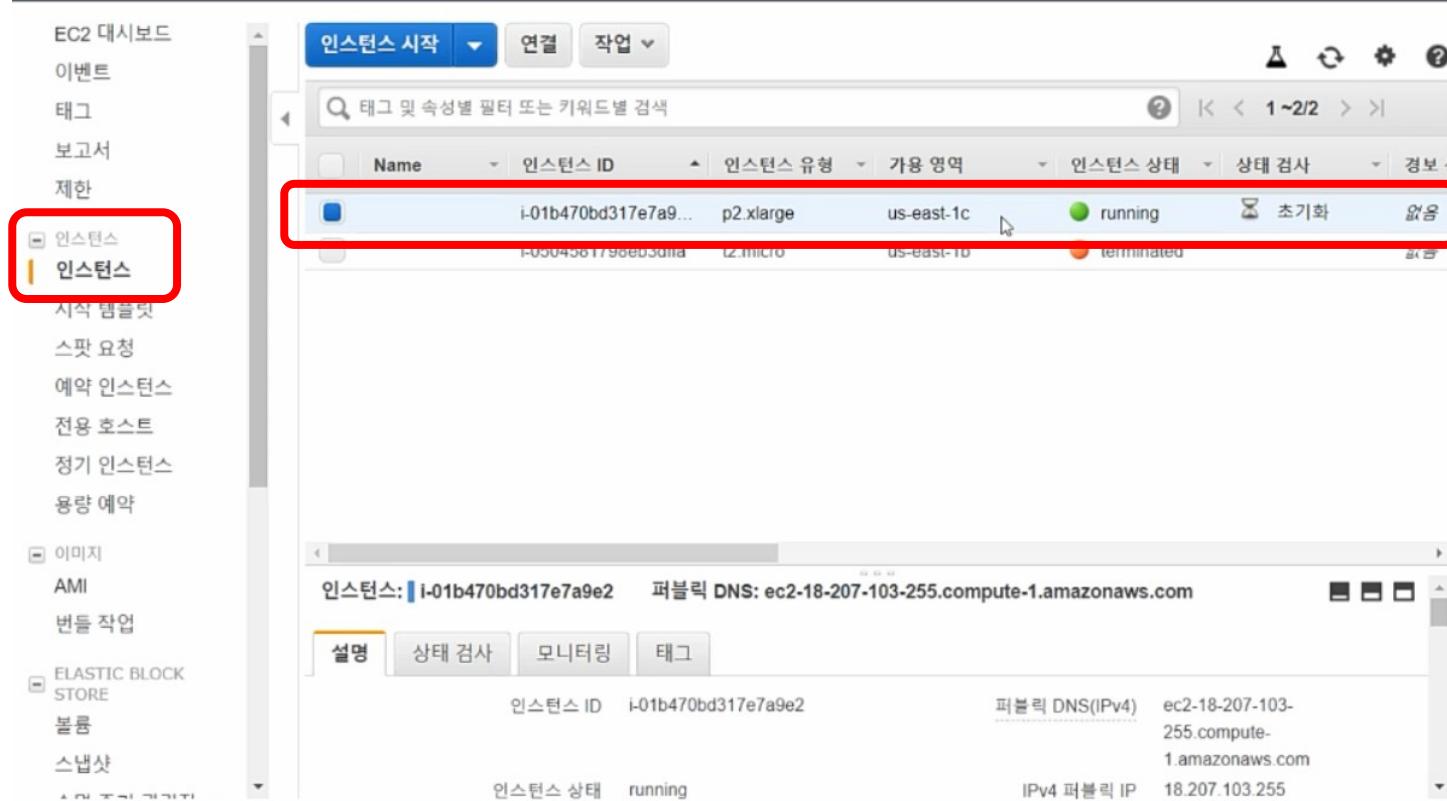
## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 1. 모델 Training, Testing, Prediction을 수행할 수 있는 GPU 서버 구축

### (16) 생성된 인스턴스 확인

왼쪽의 메뉴바에서 ‘인스턴스’를 클릭하여 이동, 인스턴스가 생성된 것을 확인. 인스턴스 선택 시 하단의 설명 탭에서 퍼블릭 DNS, IP, 프라이빗 DNS, IP등의 자세한 정보를 확인할 수 있음.

- + 인스턴스 생성 이후 비용이 발생하므로, ‘결제 대시보드’를 수시로 확인할 것을 권장



# AWS 환경에서 딥 러닝 개발하기

18

Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 2. 인스턴스에 접속하기 : ① Window에서 서버 접속\*

## (1) PuTTY 프로그램 설치

[putty.org](http://putty.org) 에서 PuTTY 다운로드 후 설치.

## (2) PuTTY와 함께 설치되는 PuTTYgen으로 키페어 변환

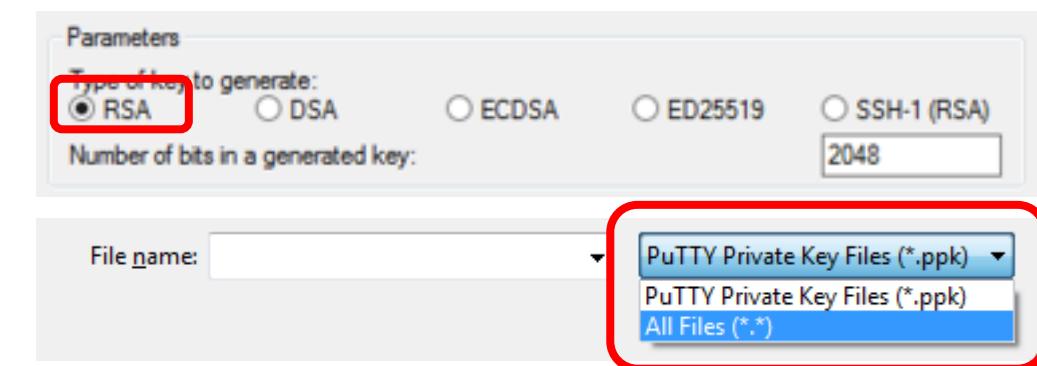
Parameters 항목에서 RSA 선택,

Load 클릭, All Files 표시 옵션 선택.

다운로드한 키 페어 (.pem)파일 선택 후 열기.

Save private key 버튼 클릭, 키 페어와 동일한 이름 사용.

추가된 .ppk 파일 저장 위치 확인.



# AWS 환경에서 딥 러닝 개발하기

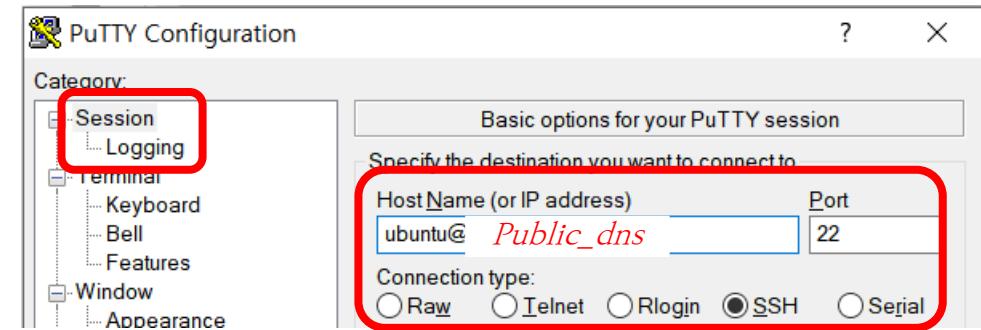
19

## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 2. 인스턴스에 접속하기 : ① Window에서 서버 접속

### (3) PuTTY 실행, Session 설정

Host Name에 `ubuntu@public_dns` 입력.

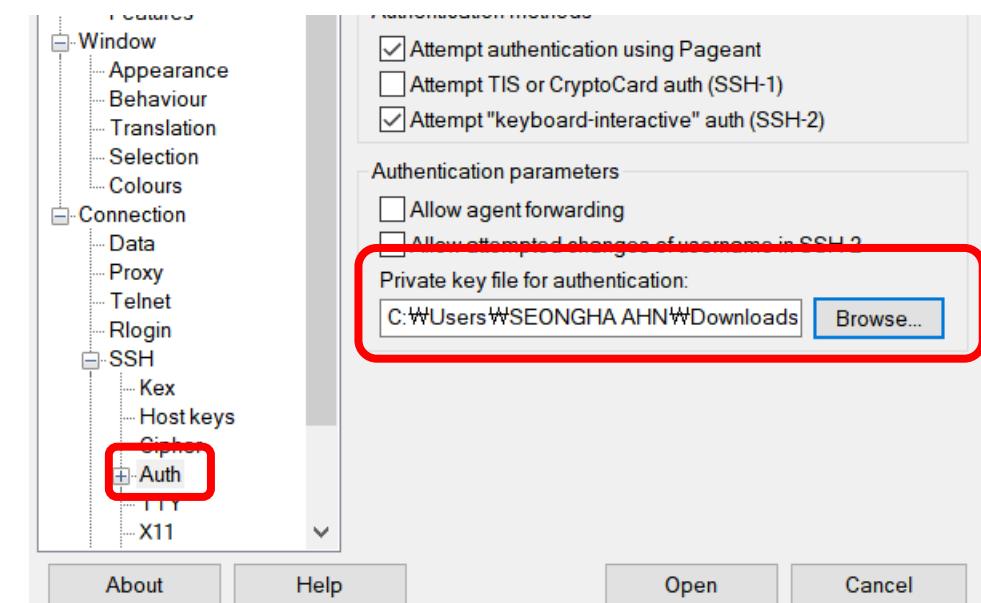


### (4) Connection/SSH/Auth 설정

'Authentication parameters'의

'Private key file for authentication' 항목에서 'Browse' 클릭,

2에서 저장한 .ppk 포맷 키페어 파일 선택.



### (5) 오른쪽 하단 'Open' 버튼 클릭하여 접속

# AWS 환경에서 딥 러닝 개발하기

20

Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 2. 인스턴스에 접속하기 : ② Linux에서 서버 접속\*  
(1) 키 페어 권한 변경 (키 페어가 저장된 디렉토리로 이동 후 실행)

```
chmod 400 keypair_name.pem
```

- (2) 터미널에서 SSH 접속

```
ssh -i /path/my-key-pair.pem ubuntu@Public_dns
```

# AWS 환경에서 딥 러닝 개발하기

21

## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 2. 인스턴스에 접속하기 : 생성한 인스턴스 접속 결과 확인

```
ubuntu@ip-172-31-1-125: ~
Using username "ubuntu".
Authenticating with public key "imported-openssh-key"
=====
[| ( | / Deep Learning AMI (Ubuntu) Version 19.0
[| \|_|_|
=====

Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-1072-aws x86_64v)

Please use one of the following commands to start the required environment with
the framework of your choice:
for MXNet(+Keras2) with Python3 (CUDA 9.0 and Intel MKL-DNN)
    source activate mxnet_p36
for MXNet(+Keras2) with Python2 (CUDA 9.0 and Intel MKL-DNN)
    source activate mxnet_p27
for MXNet(+Amazon Elastic Inference) with Python3
    source activate amazonei_mxnet_p36
for MXNet(+Amazon Elastic Inference) with Python2
    source activate amazonei_mxnet_p27
for TensorFlow(+Keras2) with Python3 (CUDA 9.0 and Intel MKL-DNN)
    source activate tensorflow_p36
for TensorFlow(+Keras2) with Python2 (CUDA 9.0 and Intel MKL-DNN)
    source activate tensorflow_p27
for Tensorflow(+Amazon Elastic Inference) with Python2
    source activate amazonei_tensorflow_p27
```

```
sh@sh-930QAA:~$ chmod 400 example.pem
sh@sh-930QAA:~$ ssh -i example.pem ubuntu@ec2-18-207-103-255.compute-1.amazonaws.com
=====
[| ( | / Deep Learning AMI (Ubuntu) Version 19.0
[| \|_|_|
=====

Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-1072-aws x86_64v)

Please use one of the following commands to start the required environment with
the framework of your choice:
for MXNet(+Keras2) with Python3 (CUDA 9.0 and Intel MKL-DNN)
    source activate mxnet_p36
for MXNet(+Keras2) with Python2 (CUDA 9.0 and Intel MKL-DNN)
    source activate mxnet_p27
for MXNet(+Amazon Elastic Inference) with Python3
    source activate amazonei_mxnet_p36
for MXNet(+Amazon Elastic Inference) with Python2
    source activate amazonei_mxnet_p27
for TensorFlow(+Keras2) with Python3 (CUDA 9.0 and Intel MKL-DNN)
    source activate tensorflow_p36
```

Figure 3 : PuTTY를 통해 접속 시 결과(좌)와 Linux 터미널에서 접속 시 결과 (우)

# AWS 환경에서 딥 러닝 개발하기

22

Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 3. Jupyter notebook 원격 접속 설정\*

(1) config 파일 생성

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

(2) password 설정

```
python3
>> from notebook.auth import passwd
>> passwd()
# Enter password: <jupyter notebook에서 사용할 패스워드 입력>
# Verify password: <패스워드 재입력>
# 'sha1:<암호화된 패스워드>'
# 출력된 암호화된 패스워드 복사
```

# AWS 환경에서 딥 러닝 개발하기

23

## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 3. Jupyter notebook 원격 접속 설정
  - (3) config 파일 열기

```
# jupyter_notebook_config.py 열기  
sudo vim /home/ubuntu/.jupyter/jupyter_notebook_config.py
```

- (4) Config 파일 수정

```
# 열린 jupyter_notebook_config.py에 아래내용 수정하여 파일 가장 아래에 붙여넣기, 수정된 파일 저장  
c = get_config()  
c.NotebookApp.password = u'sha1:<암호화된 패스워드>'  
c.NotebookApp.ip = '<프라이빗 IP>' # 인스턴스 설명탭-프라이빗 IP 참조  
c.NotebookApp.notebook_dir = '<초기 디렉토리(튜토리얼에서는 '/home/ubuntu' 입력)>'
```

# AWS 환경에서 딥 러닝 개발하기

24

## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 4. Jupyter notebook 접속과 MNIST 손글씨 숫자 Classification 예제 실행\*

(1) Jupyter notebook 실행 후 브라우저에서  $\langle /Pv4\ Public\_ip \rangle:8888$  으로 이동, 비밀번호 입력 후 접속  
ssh 접속 session 종료 후에도 Jupyter notebook 사용하고 싶은 경우, Jupyter notebook 실행 시 아래 명령어로 실행.  
실행중인 인스턴스의 background에서 실행되므로 ssh접속 없이 Jupyter notebook 사용 가능.

```
# ssh 접속 중에만 사용하는 경우 명령어 : jupyter notebook  
nohup jupyter notebook &
```

- (2) /tutorials/tensorflow 이동 후 ‘3\_mnist\_from\_scratch.ipynb’ 클릭



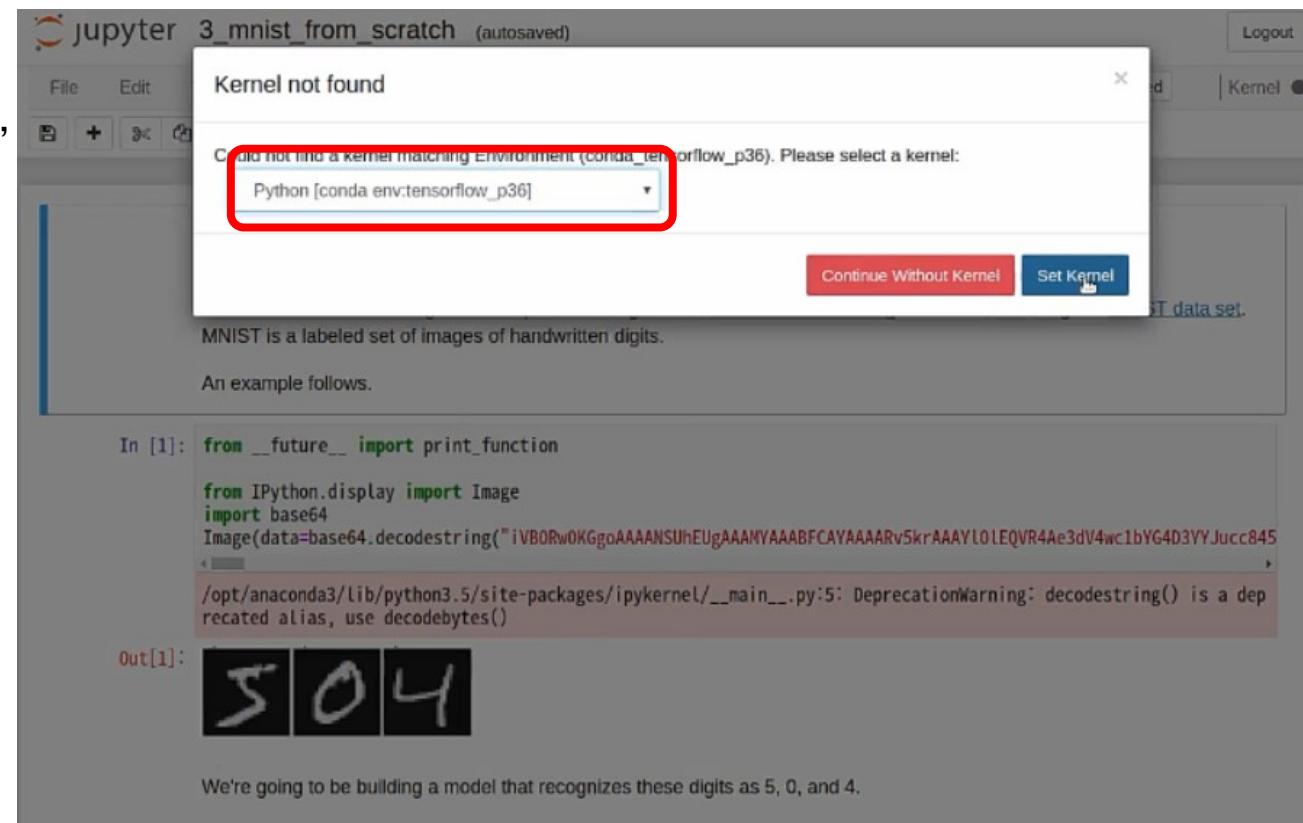
# AWS 환경에서 딥 러닝 개발하기

25

Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 4. Jupyter notebook 접속과 MNIST 손글씨 숫자 Classification 예제 실행  
(3) 커널 설정 : ‘Python [conda env:tensorflow\_p36]’ 선택, ‘Set Kernel’ 버튼 클릭

인스턴스 구성 시 사용했던 ‘Deep Learning AMI’에는 프레임워크 사용을 위한 가상환경이 구성되어 있으므로, 목적에 따라 선택. MNIST 예제 실행 시에는 ‘Python [conda env: tensorflow\_p36]’를 사용.



# AWS 환경에서 딥 러닝 개발하기

26

Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 4. Jupyter notebook 접속과 MNIST 손글씨 숫자 Classification 예제 실행

## (4) 코드 실행

설명에 따라 순서대로 코드 실행, MNIST 데이터셋을 이용한 손글씨 숫자 Classification을 위한 모델 학습과 테스트를 진행.

The screenshot shows a Jupyter Notebook interface with the title 'jupyter 3\_mnist\_from\_scratch (unsaved changes)'. The toolbar at the top includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and a 'Not Trusted' button. A red box highlights the 'Run' button in the toolbar. The code cell contains Python code for printing loss and error metrics periodically during training:

```
# Print out the loss periodically.
if step % 100 == 0:
    error, _ = error_rate(predictions, batch_labels)
    print('Step %d of %d' % (step, steps))
    print('Mini-batch loss: %.5f Error: %.5f' % (l, error, lr))
    print('Validation error: %.1f%%' % error_rate(
        validation_prediction.eval(), validation_labels)[0])
```

The output pane below the code cell displays the results of the training steps:

```
Step 0 of 916
Mini-batch loss: 7.71261 Error: 91.66667 Learning rate: 0.01000
Validation error: 88.9%
Step 100 of 916
Mini-batch loss: 3.28754 Error: 8.33333 Learning rate: 0.01000
Validation error: 6.0%
Step 200 of 916
Mini-batch loss: 3.34397 Error: 11.66667 Learning rate: 0.01000
Validation error: 3.8%
Step 300 of 916
Mini-batch loss: 3.16391 Error: 5.00000 Learning rate: 0.01000
Validation error: 3.2%
Step 400 of 916
Mini-batch loss: 3.10554 Error: 5.00000 Learning rate: 0.01000
Validation error: 2.6%
Step 500 of 916
Mini-batch loss: 3.01965 Error: 1.66667 Learning rate: 0.01000
Validation error: 2.2%
Step 600 of 916
Mini-batch loss: 3.09692 Error: 5.00000 Learning rate: 0.01000
Validation error: 2.2%
Step 700 of 916
Mini-batch loss: 3.13634 Error: 5.00000 Learning rate: 0.01000
Validation error: 2.0%
```

# AWS 환경에서 딥 러닝 개발하기

27

## Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

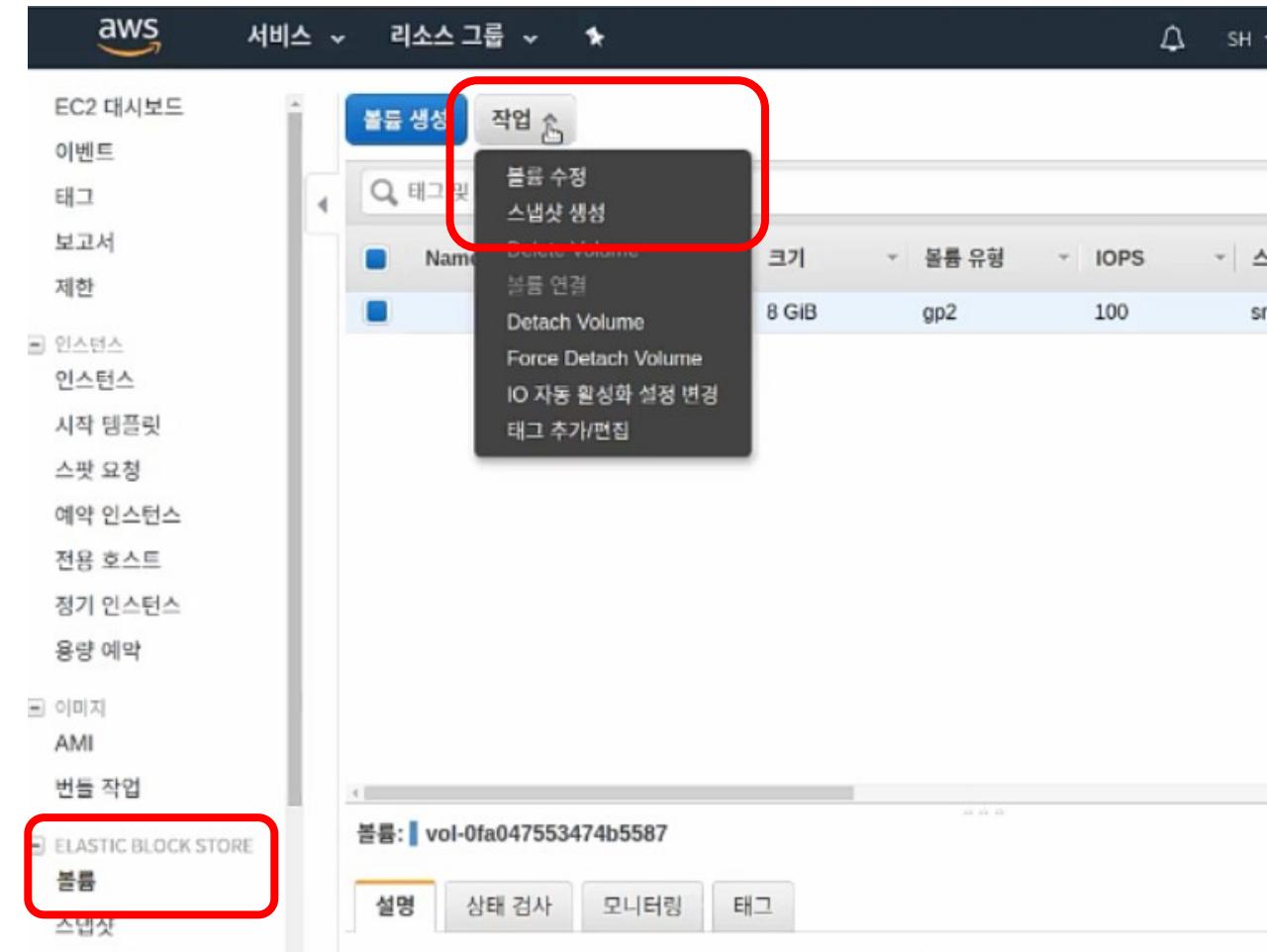
- 5. 스팟 인스턴스 종료 후 내부 데이터를 유지하여 재사용 - 볼륨에서부터 이미지 생성하기\*

### (1) 인스턴스 종료 후 남아있는 EBS 볼륨 확인

왼쪽 메뉴바에서 ‘볼륨’ 클릭.

스팟 인스턴스 생성 시, ‘종료 시 볼륨 삭제’ 박스 제크

해제했으므로 인스턴스 종료 후에도 EBS 볼륨은 유지됨.



### (2) 볼륨 선택 후 ‘작업’ – ‘스냅샷 생성’ 클릭

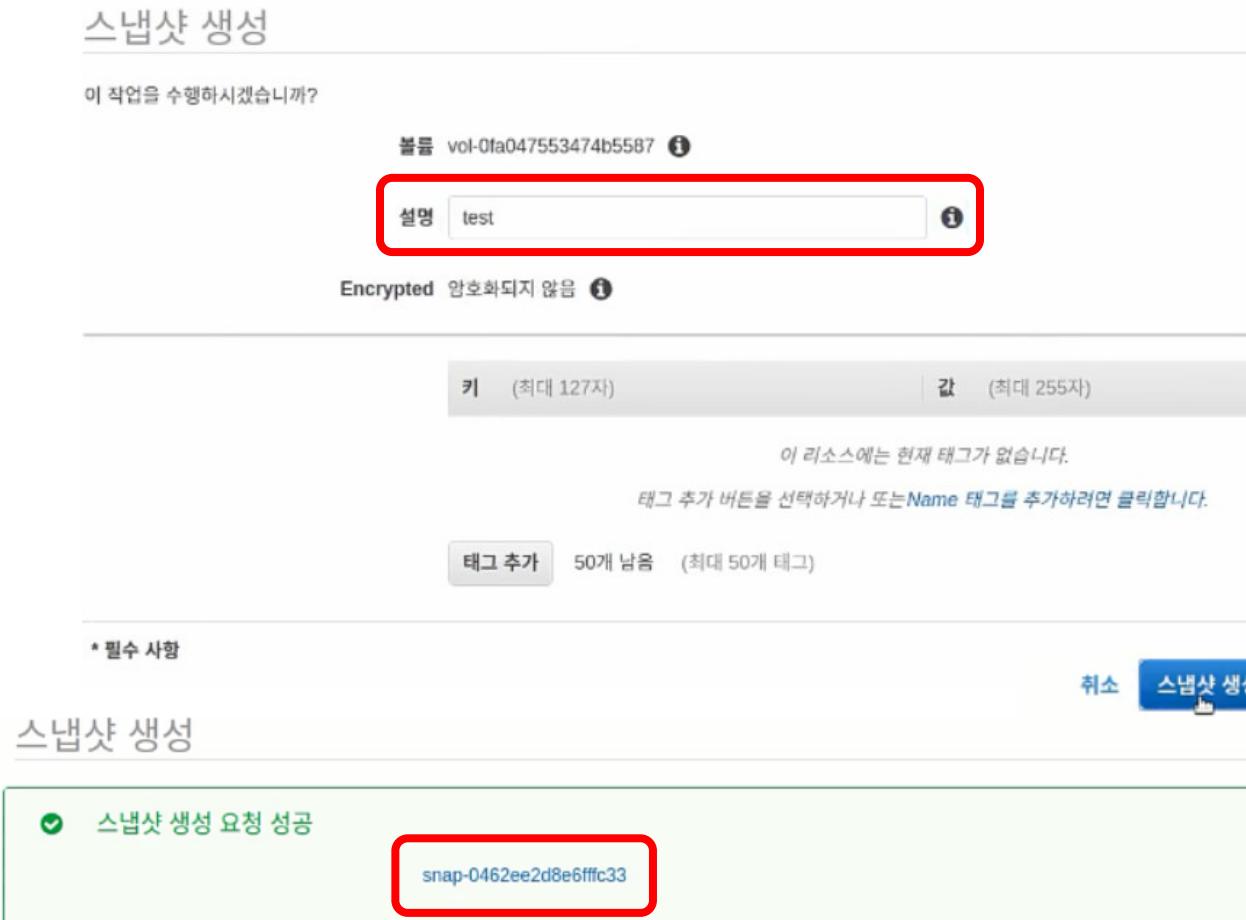
# AWS 환경에서 딥 러닝 개발하기

28

Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 5. 스팟 인스턴스 종료 후 내부 데이터를 유지하여 재사용

(3) 스냅샷 생성: 설명 입력 후 ‘스냅샷 생성’ 버튼 클릭



(4) 스냅샷 생성 요청 성공 확인

스냅샷 ID 클릭.

# AWS 환경에서 딥 러닝 개발하기

29

Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

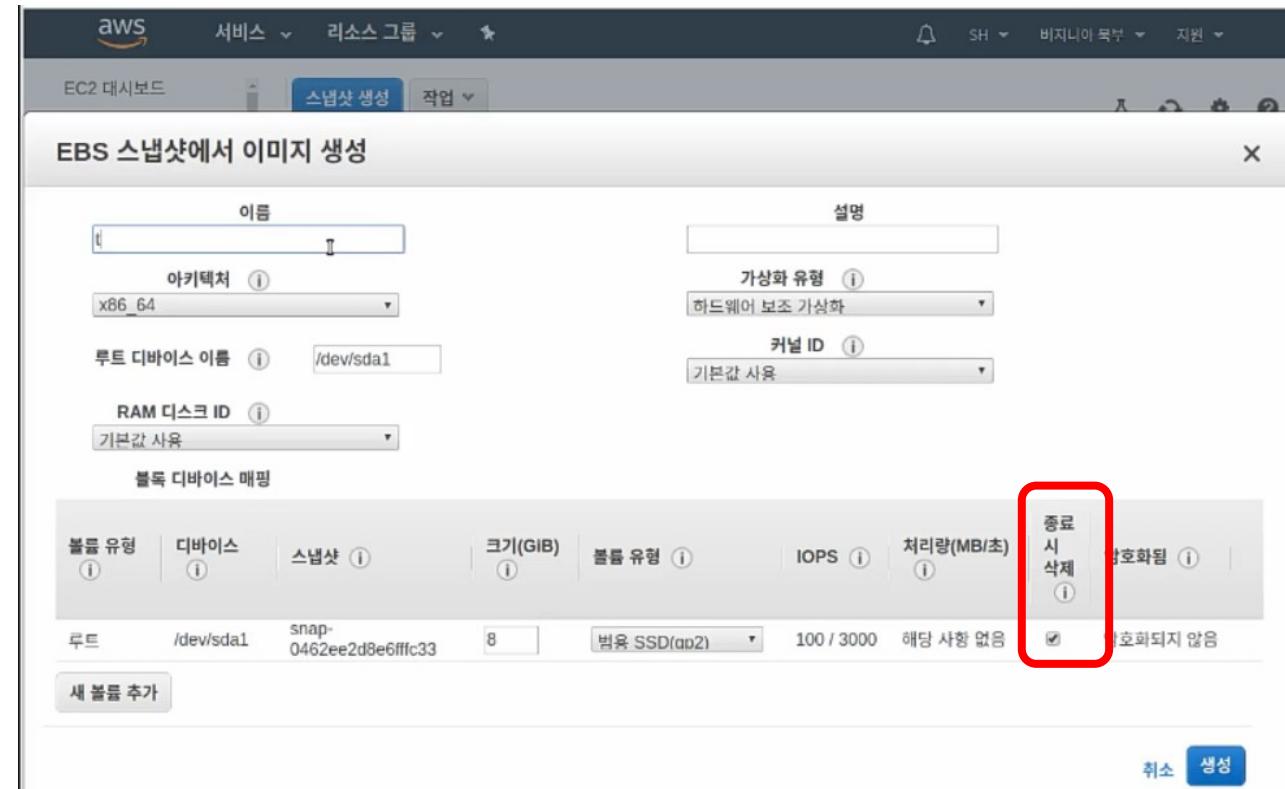
- 5. 스팟 인스턴스 종료 후 내부 데이터를 유지하여 재사용

(5) 스냅샷 선택 후 ‘작업’ – ‘이미지 생성’ 클릭

(6) ‘EBS 스냅샷에서 이미지 생성’ 창에서 이름, 설명

입력 후 ‘종료 시 삭제’ 박스 체크 해제

(7) ‘생성’버튼 클릭, 이미지 ID 클릭



# AWS 환경에서 딥 러닝 개발하기

30

Tutorial : EC2 스팟 인스턴스를 사용한 딥 러닝 개발 환경 구축하기

- 5. 스팟 인스턴스 종료 후 내부 데이터를 유지하여 재사용

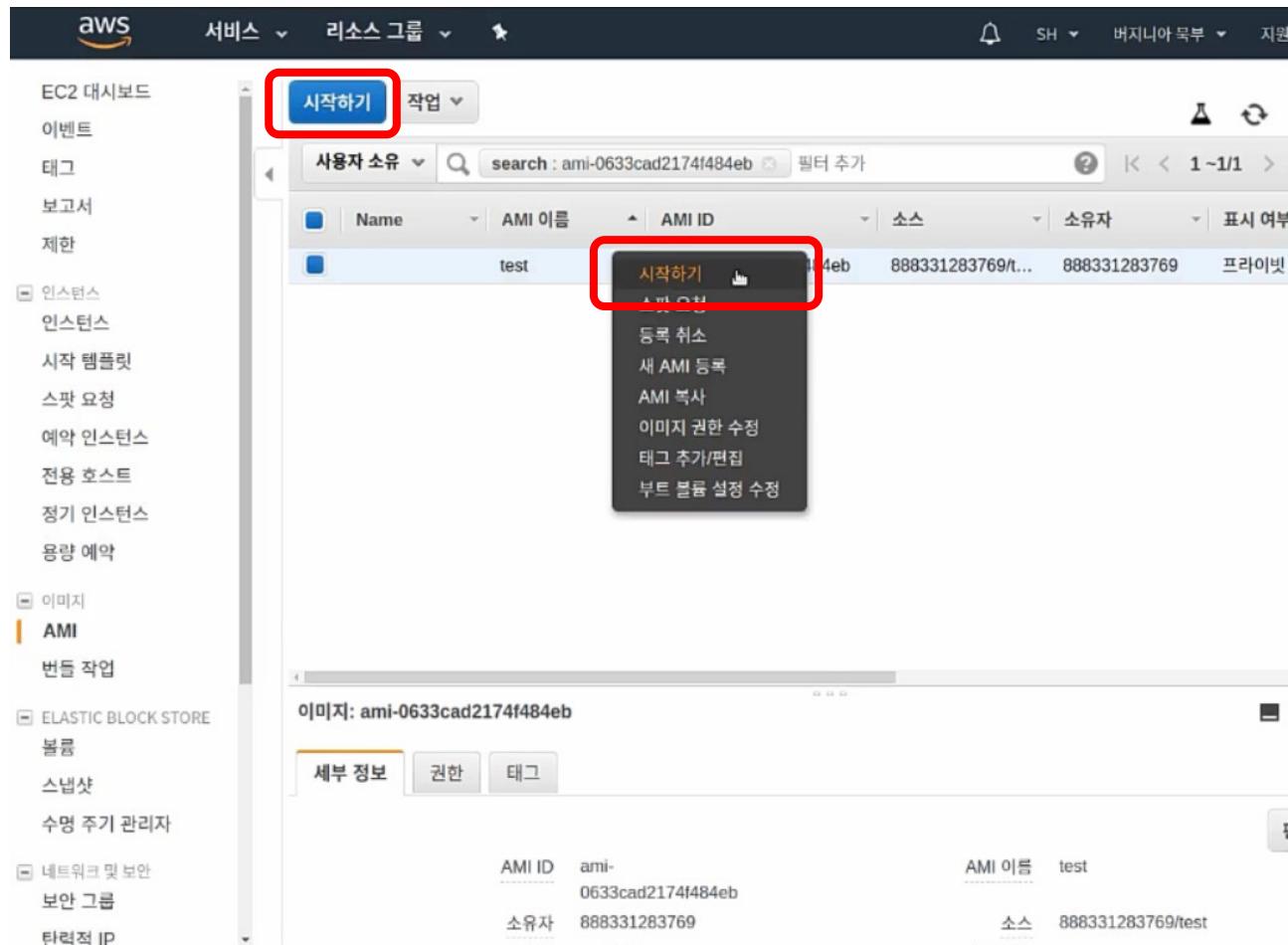
(8) 이미지 생성됨 확인

(9) 이미지 선택 후 ‘시작하기’ 버튼 클릭

(10) 생성된 이미지로 인스턴스 생성을 위한 설정 시작

(이하 1과 동일한 과정을 거쳐 인스턴스 생성)

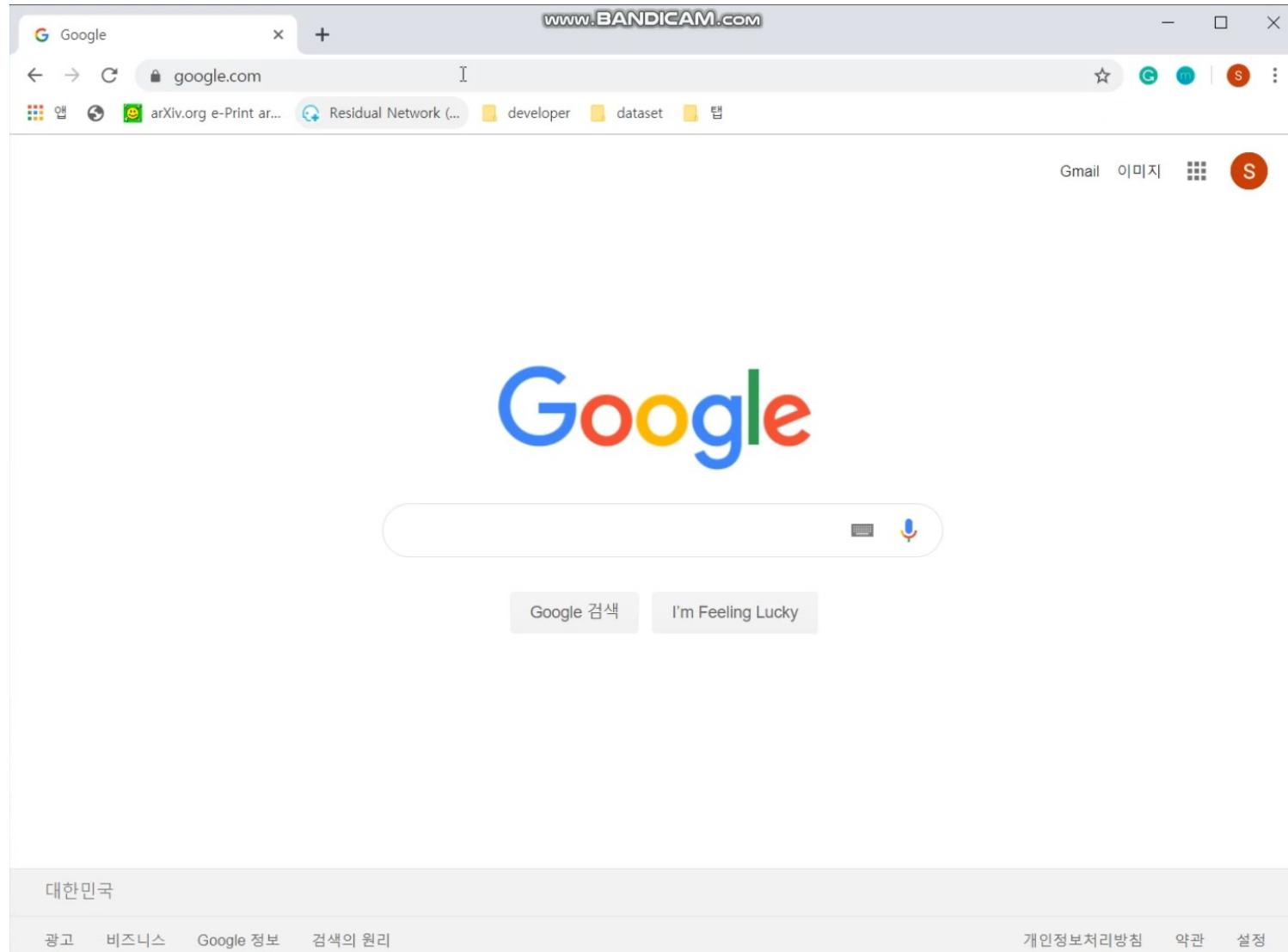
이상의 과정을 거쳐 인스턴스를 시작하면, 이전  
인스턴스의 데이터가 그대로 남아있는 인스턴스를  
사용할 수 있음.



# Supplementary

31

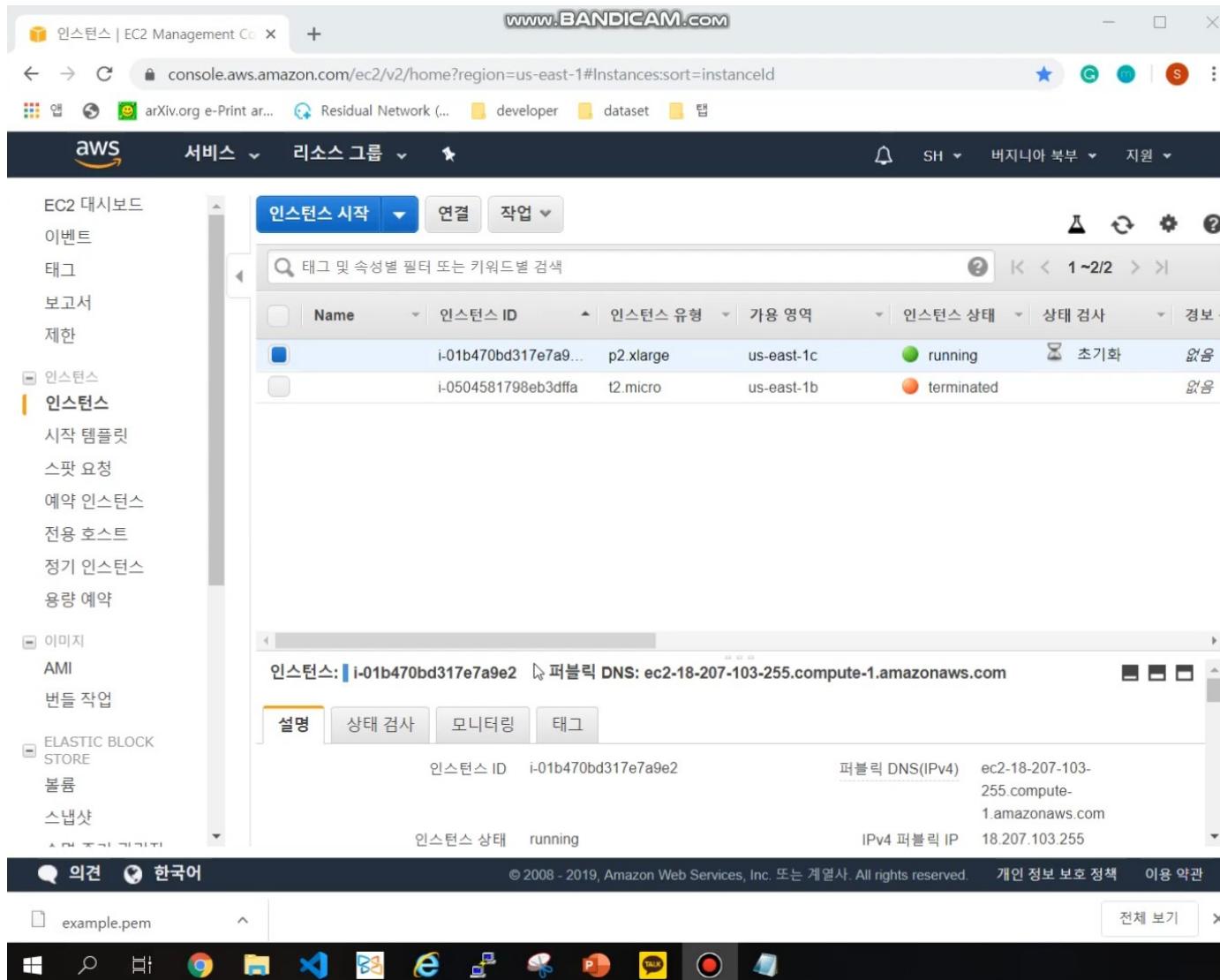
## A. EC2 인스턴스 생성 Demonstration ([Google Drive link](#))



# Supplementary

32

## B. Window에서 EC2 인스턴스 접속 Demonstration ([Google Drive link](#))



The screenshot shows the AWS EC2 Management Console interface. On the left, a sidebar menu is open under the 'Instances' section, showing options like 'Instances', 'Start Template', 'Spot Request', 'Scheduled Instances', etc. The main content area displays a list of instances. There are two entries:

Name	Instance ID	Type	Region	Status	Launch Time	Actions
	i-01b470bd317e7a9e2	p2.xlarge	us-east-1c	running	초기화	없음
	i-0504581798eb3dff	t2.micro	us-east-1b	terminated		없음

Below the instance list, a detailed view is shown for the first instance (i-01b470bd317e7a9e2). The public DNS is listed as ec2-18-207-103-255.compute-1.amazonaws.com. The instance is currently running.

At the bottom of the window, there is a footer bar with links for 'Feedback', 'Korean', and copyright information from 2008-2019 Amazon Web Services, Inc. The taskbar at the very bottom of the screen shows various application icons.

[해당 항목으로 돌아가기](#)

# Supplementary

33

## C. Linux에서 EC2 인스턴스 접속 Demonstration ([Google Drive link](#))

The screenshot shows the AWS Management Console homepage. The top navigation bar includes the AWS logo, service dropdown, resource group dropdown, notifications, and region selection (Virginia North). The main header reads "AWS Management Console".

**AWS 서비스** (AWS Services) section:

- 서비스 찾기**: A search bar with placeholder text "예: 관계형 데이터베이스 서비스, 데이터베이스, RDS".
- 최근 방문한 서비스**: A list with icons and names: EC2, 결제, IAM, S3, Amazon SageMaker.
- 전체 서비스**: A button to view all services.

**이동 중에도 리소스 액세스** (Access resources on the go) section:

AWS Console 모바일 앱을 사용하여 관리 콘솔에 액세스합니다. [자세히 알아보기](#)

**AWS 탐색** (AWS Explore) section:

- Amazon Redshift**: 퀼리를 사용자 데이터 레이크로 확장할 수 있는 빠르고 간단하며 비용 효과적인 데이터 웨어하우스입니다. [자세히 알아보기](#)
- AWS Fargate를 사용하여 서비스 컨테이너 실행**: AWS Fargate는 서버 또는 클러스터를 관리할 필요 없이 컨테이너를 실행 및 확장합니다. [자세히 알아보기](#)
- Amazon S3를 통해 확장 가능하고 내구성이 뛰어나며 안전한 백업 및 복원 수행**

[해당 항목으로 돌아가기](#)

# Supplementary

34

## D. Jupyter notebook 원격 접속 설정 Demonstration ([Google Drive link](#))

The screenshot shows the AWS EC2 Management Console interface. On the left, a sidebar menu includes 'EC2 대시보드', '이벤트', '태그', '보고서', '제한', '인스턴스' (selected), '시작 템플릿', '스팟 요청', '예약 인스턴스', '전용 호스트', '정기 인스턴스', '용량 예약', '이미지' (AMI), '번들 작업', 'ELASTIC BLOCK STORE' (EBS), '네트워크 및 보안' (Network & Security), and '탄력적 IP' (Elastic IP). The main content area displays a list of instances with the following details:

Name	인스턴스 ID	인스턴스 유형	가용 영역	인스턴스 상태	상태 검사	경보 상태
	i-01b470bd317e7a9e2	p2.xlarge	us-east-1c	running	2/2 검사 통과	없음
	i-0504581798eb3dffaa	t2.micro	us-east-1b	terminated		없음

Below the list, a detailed view for the running instance (i-01b470bd317e7a9e2) is shown. It includes the instance ID, public DNS (ec2-18-207-103-255.compute-1.amazonaws.com), and its configuration:

설명	상태 검사	모니터링	태그
인스턴스 ID: i-01b470bd317e7a9e2	인스턴스 상태: running	퍼블릭 DNS(IPv4): ec2-18-207-103-255.compute-1.amazonaws.com	
	인스턴스 유형: p2.xlarge	IPv4 퍼블릭 IP: 18.207.103.255	
		IPv6 IP: -	

[해당 항목으로 돌아가기](#)

# Supplementary

35

## E. Jupyter notebook 접속 Demonstration ([Google Drive link](#))

The screenshot shows a terminal window on an AWS EC2 instance. The user is configuring a Jupyter notebook by running the command `jupyter notebook --generate-config`. They then open a text editor (vim) to edit the configuration file `jupyter_notebook_config.py`. In the editor, they change the password to a SHA1 hash and set the IP address to the instance's private IP (172.31.1.125). The terminal also shows the user navigating through the AWS EC2 Management console, viewing instance details like state and network settings.

```
# password 를 python3
from notebook.auth import passwd
passwd()
# Enter password:jupyter notebook에서 사용할 패스워드 입력
# Verify password: 패스워드 재입력

'sha1:4db09cdcfb83:4073595f409cb5b7256e0f44b77923fc23c84381'
# 암호화된 패스워드 복사

# jupyter_notebook_config.py 열기
sudo vim /home/ubuntu/.jupyter/jupyter_notebook_config.py

# jupyter_notebook_config.py에 아래내용 붙여넣기
# c = get_config()
# c.NotebookApp.password = u'sha1:<암호화된 패스워드>'
# c.NotebookApp.ip = '<프라이빗 IP>'
# c.NotebookApp.notebook_dir = '<시작 디렉토리>'

c = get_config()
c.NotebookApp.password =
u'sha1:4db09cdcfb83:4073595f409cb5b7256e0f44b77923fc23c84381'
c.NotebookApp.ip = '172.31.1.125'
c.NotebookApp.notebook_dir = '/home/ubuntu'

# jupyter_notebook_config.py 저장
# 브라우저에서 'IPv4 퍼블릭IP:8888'로 이동 후 패스워드 입력
# 주피터 노트북 사용 준비 완료
```

Current Activity: 터미널  
AWS EC2 Management | 안경잡이개발자 :: AWS EC2 | +

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

```
zict          0.1.3
You are using pip version 10.0.1, however version 19.2.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
ubuntu@ip-172-31-1-125:~$ jupyter notebook --generate-config
Overwrite /home/ubuntu/.jupyter/jupyter_notebook_config.py with default config? [y/N]y
Writing default config to: /home/ubuntu/.jupyter/jupyter_notebook_config.py
ubuntu@ip-172-31-1-125:~$ python3
Python 3.6.5 |Anaconda, Inc.| (default, Apr 29 2018, 16:14:56)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from notebook.auth import passwd
>>> passwd()
Enter password:
Verify password:
'sha1:4db09cdcfb83:4073595f409cb5b7256e0f44b77923fc23c84381'
>>>
ubuntu@ip-172-31-1-125:~$ sudo vim /home/ubuntu/.jupyter/jupyter_notebook_config.py
ubuntu@ip-172-31-1-125:~$
```

AMI  
번들 작업  
ELASTIC BLOCK STORE  
볼륨  
스냅샷  
수명 주기 관리자  
네트워크 및 보안  
보안 그룹  
탄력적 IP

인스턴스 상태: running  
인스턴스 유형: p2.xlarge  
탄력적 IP  
가용 영역: us-east-1c  
보안 그룹: test1. 인바운드 규칙 보기  
아웃바운드 규칙 보기

(화) 15 : 47

\*이름 없는 문서 1

저장(S) ...

일반 텍스트 템 너비: 8 22행, 1열

프라이빗 DNS: ip-172-31-1-125.ec2.internal  
프라이빗 IP: 172.31.1.125  
보조 프라이빗 IP:

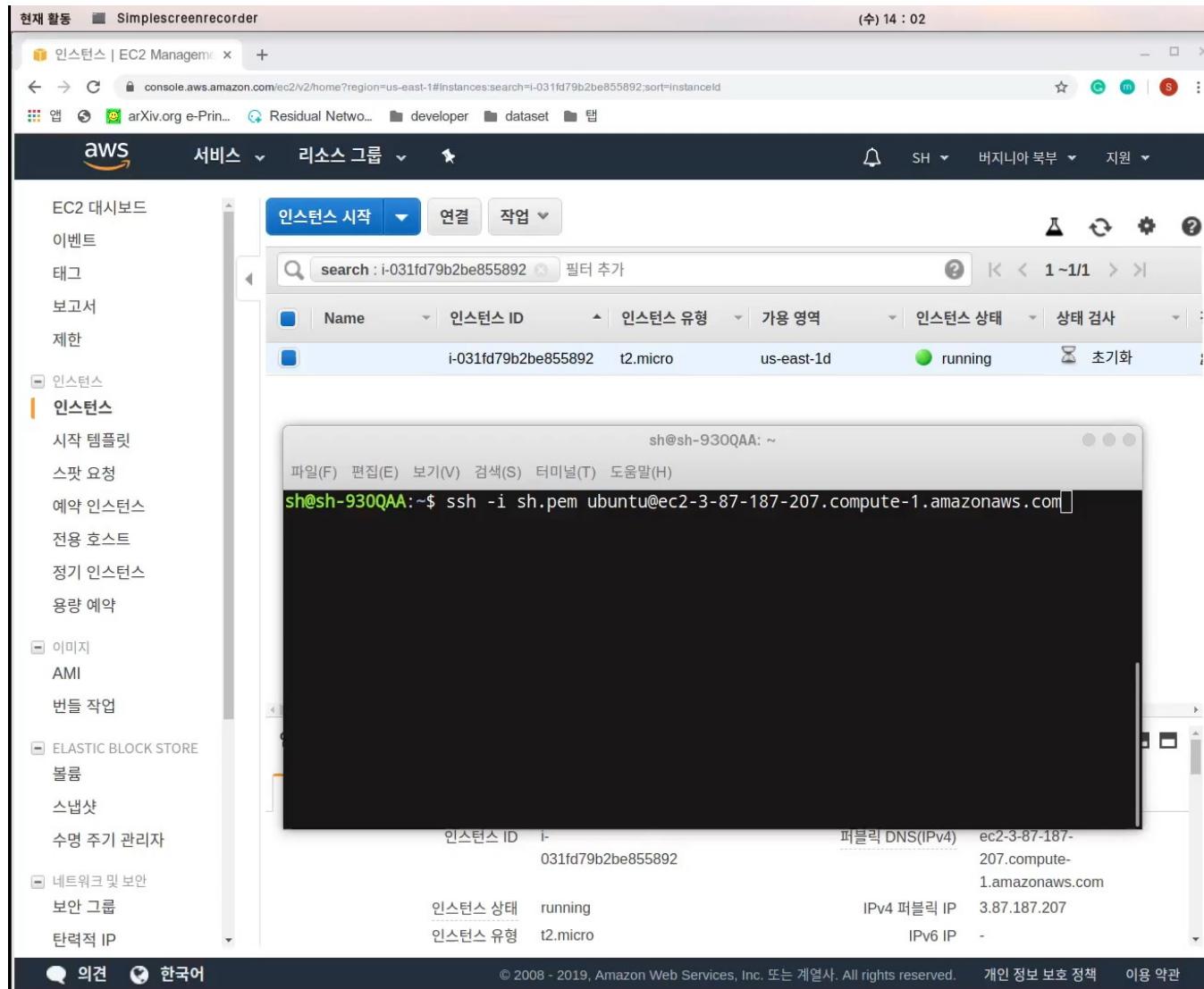
© 2008 - 2019, Amazon Web Services, Inc. 또는 계열사. All rights reserved. 개인 정보 보호 정책 이용 약관

[해당 항목으로 돌아가기](#)

# Supplementary

36

## F. EBS 볼륨 - 스냅샷 생성 - 이미지 생성 - 인스턴스 시작 Demonstration ([Google Drive link](#))



# Supplementary

36

## G. Tensorflow, Keras 학습 중 모델 저장과 저장한 모델 불러오기

- Tensorflow : tf.train.Saver<sup>11</sup> API
  - TensorFlow를 이용한 MNIST 데이터셋 손글씨 숫자 Classification에서 모델 저장하고 불러오기 ([.ipynb](#))
- Keras : tf.keras.callbacks.ModelCheckpoint <sup>12</sup>
  - Keras를 이용한 MNIST 데이터셋 손글씨 숫자 Classification에서 모델 저장하고 불러오기 ([.ipynb](#))

## H. 인스턴스가 종료된 경우 학습 진행상황 체크를 위한 nohup 명령 사용

‘mnist.py’라는 코드를 실행할 경우, 아래 명령어 사용하여 실행

```
# in terminal  
nohup python mnist.py &
```

터미널에 출력되는 내용이 ‘nohup.out’ 파일에 저장되므로, 인스턴스가 예기치 않게 종료되었을 경우 ‘nohup.out’ 파일을 확인하여 종료 직전까지의 학습 진행상황을 확인할 수 있음

# Reference

1. RightScale 2018 State of the Cloud Report
2. 프리 티어에 관한 자세한 내용은 AWS 설명서 내 ‘프리 티어 사용’ 항목 참조  
(Link : <http://bitly.kr/freetier> )
3. EC2에 관한 자세한 내용은 AWS 설명서 내 ‘EC2’ 항목 참조  
(Link : <http://bitly.kr/awsec2> )
4. EC2의 구매 옵션에 관한 자세한 내용은 AWS 설명서 내 ‘인스턴스 구매옵션’ 항목 참조  
(Link : <http://bitly.kr/instancesoption> )
5. 스팟 인스턴스 개념, 사용방법, 요금은 AWS 설명서 내 ‘스팟 인스턴스’ 항목 참조  
(Link : <http://bitly.kr/spotinstances> , 실시간 요금 확인 Link : <https://aws.amazon.com/ko/ec2/spot/pricing/>)
6. 스팟 인스턴스가 종료되는 경우에 관한 자세한 내용은 AWS 설명서 내 ‘스팟 인스턴스 중단’ 항목 참조  
(Link : <http://bitly.kr/instancesshutdown> )
7. EBS에 관한 자세한 내용은 AWS 설명서 내 ‘Elastic Block Store’ 항목 참조  
(Link : <http://bitly.kr/awsebs> )

# Reference

38

8. AWS 스토리지 제품군에 관한 자세한 내용은 아래 링크 참조  
(Link : <http://bitly.kr/awsstorage> )
9. 스냅샷에 관한 자세한 내용은 AWS 설명서 내 ‘EBS 스냅샷’ 항목 참조  
( Link : <http://bitly.kr/awssnapshot> )
10. Deep Learning AMI에 관한 자세한 내용은 AWS 설명서 내 ‘Deep Learning AMI’ 항목 참조  
( Link : <http://bitly.kr/awsdlami> )
11. tf.train.Saver API 사용법, 메소드 등 자세한 내용은 API documentation 내 ‘Saver’ 항목 참조  
(Link : <http://bitly.kr/tfsaver> )
12. Keras model.save 사용법은 Keras documentation 내 ‘FAQ’ 참조  
(Link : <https://keras.io/getting-started/faq/#how-can-i-save-a-keras-model>)

\*이 자료는 (주)성우하이텍 기업 연수(기간:2019.08.19 – 2019.08.30)에 참가한 동서대학교 안성하가 제작하였습니다.  
자료 내 모든 저작권은 안성하에게 있으며, 저자 명시를 조건으로 자유롭게 수정, 배포하실 수 있습니다.  
Supplementary의 동영상과 코드는 <http://bitly.kr/spmntryfordlonaws> 에서도 확인하실 수 있습니다.