

# CLOUD WITH AWS

PART I

ABOUT  
CLOUD SERVICE

# ABOUT CLOUD SERVICE

---

## ◆ 클라우드 서비스란?

- 컴퓨터 사용한 정보 처리를 자신이 보유한 PC가 아닌, **인터넷 '너머'에 존재하는 클라우드 사업자의 컴퓨터에서 처리하는 서비스**
- **공유 구성이 가능한 컴퓨팅 리소스(네트워크, 서버, 스토리지, 애플리케이션 서비스) 통합** 통해 어디서나 간편하게, 요청에 따라 네트워크 통해 접근하는 것 가능하게 하는 모델

# ABOUT CLOUD SERVICE

---

## ◆ 클라우드 서비스 등장 배경

### ▪ 1980년대

- 메인 프레임 이라고 하는 **대형 범용 컴퓨터** 시대
- 모든 데이터와 애플리케이션을 **메인 프레임에 모아 처리**
- 단말기는 입력과 출력 표시 기능만 담당

### ▪ 1990년대

- **분산형 클라이언트 서버 모델** 주류
- 클라이언트 단말기에 처리 기능을 탑재하여 **집중 처리 = = = >> 분산 처리**

# ABOUT CLOUD SERVICE

---

## ◆ 클라우드 서비스 등장 배경

- 2000년대

- 사내 시스템이 네트워크 환경 위에 구축됨 처리가 서버에 집중됨.

- 2010년대

- 필요한 때 필요한 만큼 사용하는 클라우드 컴퓨팅 모델로 발전.
  - 서버를 보유하는 것이 아니라, 전 세계에 분산된 서버의 리소스들을 서비스로써 이용하는 모델

# ABOUT CLOUD SERVICE

---

## ◆ 클라우드 서비스 등장 배경

- 온-프레미스 소프트웨어(On-Premises software)

- 소프트웨어를 서버에 직접 설치해 사용하는 방식 → 클라우드 환경과 대조됨
- 클라우드 컴퓨팅 기술이 나오기 전까지 기업 인프라 구축의 일반적인 방식
- 서버와 기타 H/W 인프라 구축비용 및 유지보수 비용, S/W 라이선스 지불 비용 증가
- 구축 수개월 이상 걸리고 비용 또한 많이 들어감

# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 유형

### ▪ IaaS (Infrastructure as a Service)

- CPU나 H/W 등 컴퓨팅 리소스를 네트워크 통해 서비스로 제공하는 모델
- 클라우드 사업자가 보유하는 물리적 서버의 CPU, 메모리, 스토리지 등 하드웨어 자원을 소프트웨어적으로 나누어 사용자에게 제공 → 자유롭게 스케일 업/다운 가능
- 물리적 인프라 구축 및 유지보수와 관련된 **비용과 복잡성을 상당 부분 없애줌**
- 예: *Compute Engine, Cloud Storage*

# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 유형

### ▪ PaaS (Platform as a Service )

- H/W, S/W, 인프라가 포함된 **완전한 클라우드 플랫폼 제공** 클라우드 컴퓨팅 형태
- **애플리케이션 개발 및 실행 환경과 서비스 개발, 배포, 관리 가능한 개발 툴 제공**
- 개발 환경 처음부터 구축하는 것은 많은 시간 소요
- Java, PHP, Ruby 등 프로그래밍 언어 지원하는 실행 환경/ DB 미리 준비되어 제공
- 예: **Cloud Run, App Engine.**

# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 유형

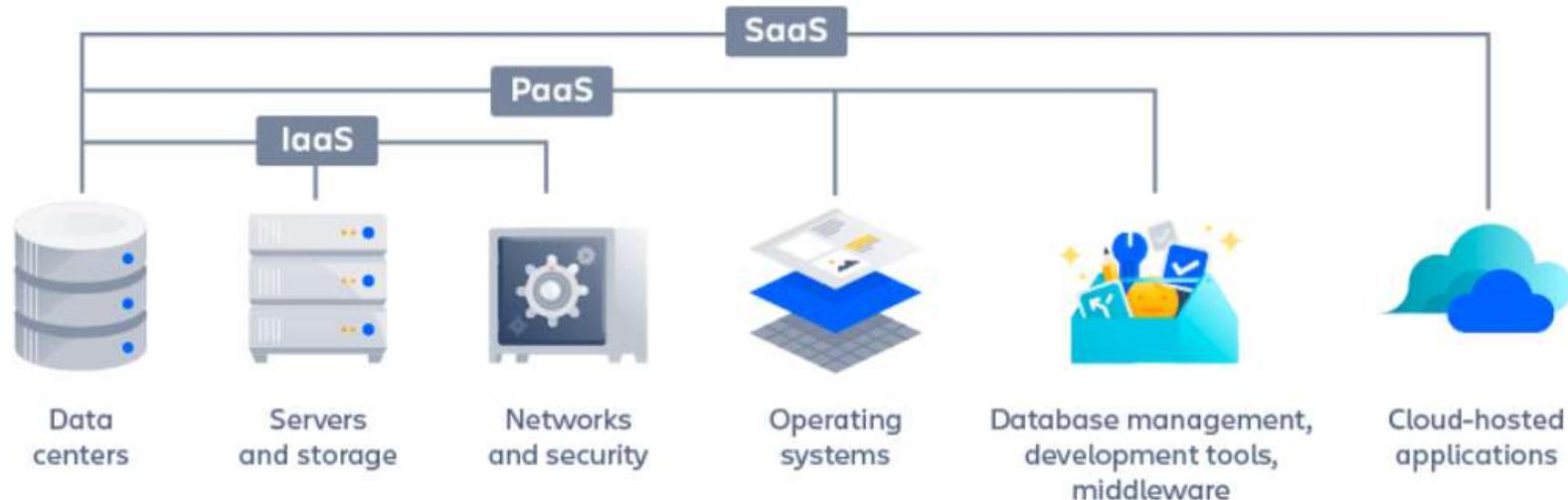
### ▪ SaaS (Software as a Service)

- 고객이 이미 구축된 소프트웨어에 사용료를 지불하면서 이용하는 형태
- 클라우드 애플리케이션과 기본 IT 인프라 및 플랫폼을 **인터넷 브라우저 통해 최종 사용자에게 제공**하는 클라우드 컴퓨팅 형태
- 서비스 성능은 **인터넷 연결 속도**에 따라 달라짐 → 고속 네트워크 하드웨어 투자
- 고객이 **인터넷 통해 앱만 연결하면 제공업체가 다른 모든 작업 수행**
- 예: **Google Workspace, iCloud, Google Drive, 네이버MYBOX**

# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 유형

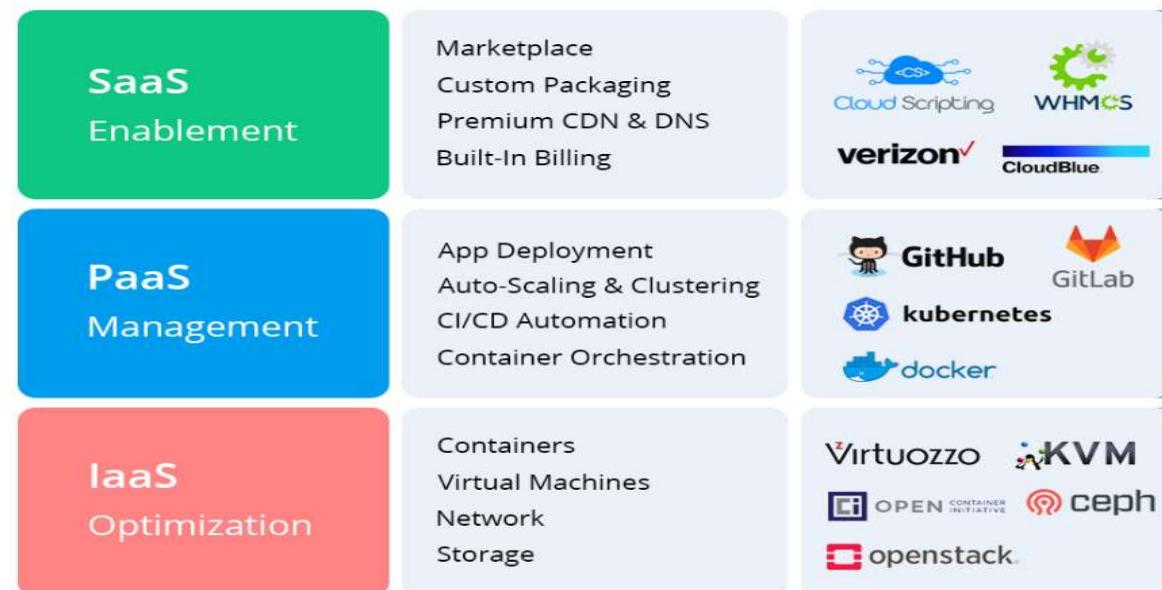
### ▪ SaaS / PaaS / IaaS 제공 서비스 비교



# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 유형

### ▪ SaaS / PaaS / IaaS 제공 서비스 비교



# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 유형

### ▪ 차이점



# ABOUT CLOUD SERVICE

## ◆ 클라우드 컴퓨팅 배포 모델

### ▪ 퍼블릭 클라우드(Public Cloud)

- 일반 대중, 개인이나 기업에서 사용하기 위해 프로비저닝된 클라우드 인프라
- 일반적으로 쓰는 클라우드 형태
- 수요가 유동적인 기업, Gmail, Office 365 같은 공용 어플리케션, 에어비앤비, 넷플릭스 등 세계적으로 서비스 빠르게 확장 할 때 적합한 모델

- 대규모 클라우드 제공업체(AWS, Microsoft, Google) 중 한 곳에 의해 호스팅
- 인터넷 통해 IT 리소스와 서비스(IaaS, PaaS, SaaS) 제공

# ABOUT CLOUD SERVICE

## ◆ 클라우드 컴퓨팅 배포 모델

### ▪ 퍼블릭 클라우드(Public Cloud)

#### 장점

- 물리적 서버 구매, 설치 필요 없음 → 비용 절약
- 필요할 때 필요한 만큼 사용 가능함 → 단순성
- 서버 운영 및 유지보수 (O&M) 부담 없음 → 총 소유비용 (TCO) 절감

# ABOUT CLOUD SERVICE

## ◆ 클라우드 컴퓨팅 배포 모델

### ▪ 퍼블릭 클라우드(Public Cloud)

#### 단점

- 불특정 다수의 액세스에 대한 무단 액세스 또는 크랙 위험 발생 쉬움
- 서비스 내용이 미리 정해져 있음 → 없는 서비스 사용 위한 다른 추가 작업 수행 필요
- 회사 사정으로 서비스 갑자기 중단될 가능성 있음
- 중요한 데이터 클라우드에 남아 있으면 손상 발생 가능

# ABOUT CLOUD SERVICE

## ◆ 클라우드 컴퓨팅 배포 종류

### ▪ 프라이빗 클라우드(Private Cloud)

- 단일 조직 전용 클라우드 환경을 제공하는 클라우드 컴퓨팅 모델
  - 해당 조직의 방화벽 뒤에서 사내 IT팀이 내부적으로 운영
  - 주로 기업 이용하기 때문에 기업용 클라우드 유형으로 분류
- 
- 온프레미스 프라이빗 클라우드 → 자사 전용 클라우드 환경 구축 및 운영 형태
  - 호스티드 프라이빗 클라우드 → 클라우드 제공업체가 사용자별 환경 제공 형태

# ABOUT CLOUD SERVICE

## ◆ 클라우드 컴퓨팅 배포 종류

### ▪ 프라이빗 클라우드(Private Cloud)

#### 장점

- 조직 인트라넷 또는 사업자 데이터 센터에 구성되며 데이터는 모두 방화벽으로 보호
- 최고 수준 보안 적용된 클라우드 솔루션

#### 단점

- 서버와 기타 H/W의 인프라 구축 비용 및 유지보수 비용, SW 라이선스 **지불 비용 증가**

# ABOUT CLOUD SERVICE

---

## ◆ 클라우드 컴퓨팅 배포 종류

### ▪ 커뮤니티 클라우드(Community Cloud)

- 공통 목적 가진 특정 기업들이 클라우드 시스템 형성하여 데이터 센터에서 공동 운영 형태
- 단일 조직이 이용하는 Private Cloud를 몇몇 조직이 같이 공유하는 클라우드
- 그룹사의 Family들끼지 Public Cloud 장점 수용 / Pirvate Colud 장점 수용
- 퍼블릭 클라우드와 프라이빗 클라우드의 중간적인 형태.

# ABOUT CLOUD SERVICE

## ◆ 클라우드 컴퓨팅 배포 종류

### ▪ 하이브리드 클라우드( Hybrid Cloud)

- 네트워크 연결 통해 하나 이상의 **퍼블릭 클라우드 및 프라이빗 클라우드 환경 결합**
- 서로 다른 클라우드 환경 간 데이터와 애플리케이션 공유할 수 있는 클라우드 컴퓨팅 환경
- 기업의 중요한 데이터는 **보안이 보장된 프라이빗 클라우드에 저장**
- 제한없이 클라우드의 **인프라를 필요로 하는 기술들은 퍼블릭 클라우드**
- VPN과 Express Connect (P2P 전용 연결)라는 두 가지 방법으로 연결

# ABOUT CLOUD SERVICE

## ◆ 클라우드 컴퓨팅 배포 종류

### ▪ 하이브리드 클라우드(Hybrid Cloud)

#### 장점

- 프라이빗 클라우드와 퍼블릭 클라우드의 장점을 모두 가짐
- 확장성이 뛰어나고 저장공간이 무제한이며 결제 모델이 유연해 경제적
- 클라우드 리소스 더 유연하게 사용, 더 강력하게 통제, 보안 철저

#### 단점

- 사내 하드웨어와 필요한 추가 소프트웨어 및 도구에 **계속 투자하고 유지보수**
- IT팀과 비즈니스 사용자 모두에게 새로운 **기술 전문성이 필요**
- 클라우드 **환경도 복잡**

# ABOUT CLOUD SERVICE

---

## ◆ 클라우드 컴퓨팅 배포 종류

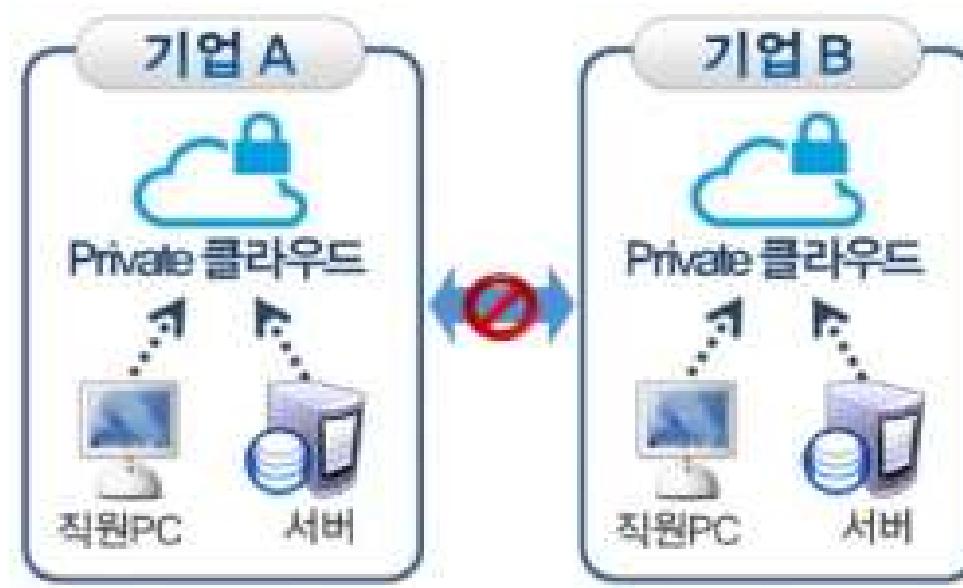
- 멀티 클라우드(Multi Cloud)

- **다수의 퍼블릭 클라우드 의미**
- 여러 클라우드 공급자의 퍼블릭 클라우드 통합
- **하이브리드 클라우드 ← → 멀티클라우드 : 개념은 다름!!!**

# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 적용

- 보안성 ▲
- 서비스 확장성 ▼



# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 적용

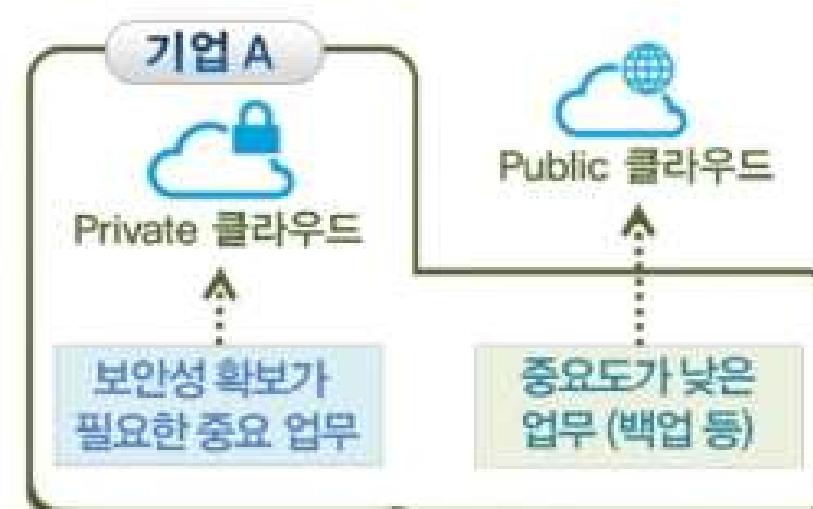
- 보안성 ▼
- 서비스 확장성 ▲



# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 적용

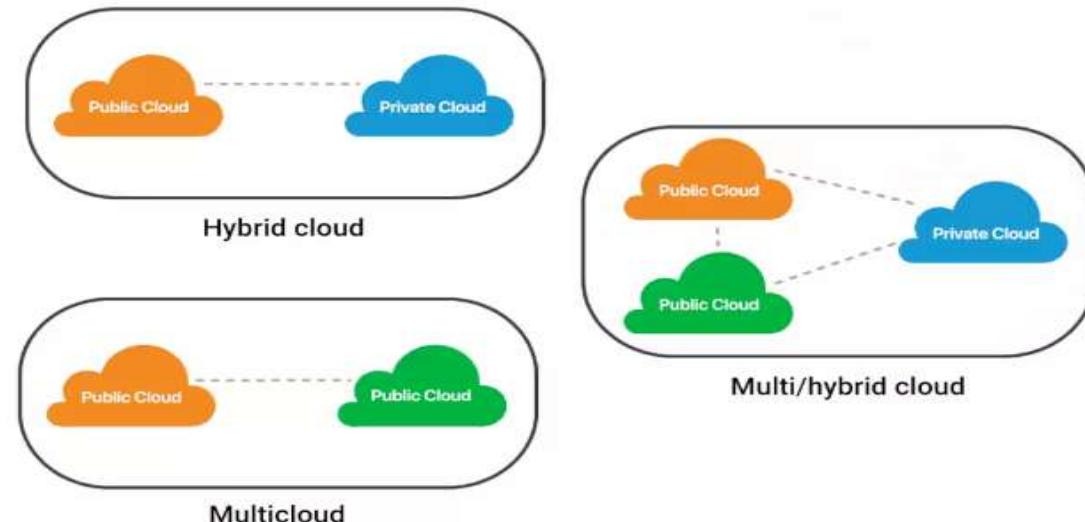
- Private Cloud : 보안성 ▲
- Public Cloud : 서비스 확장성 ▲



# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 종류

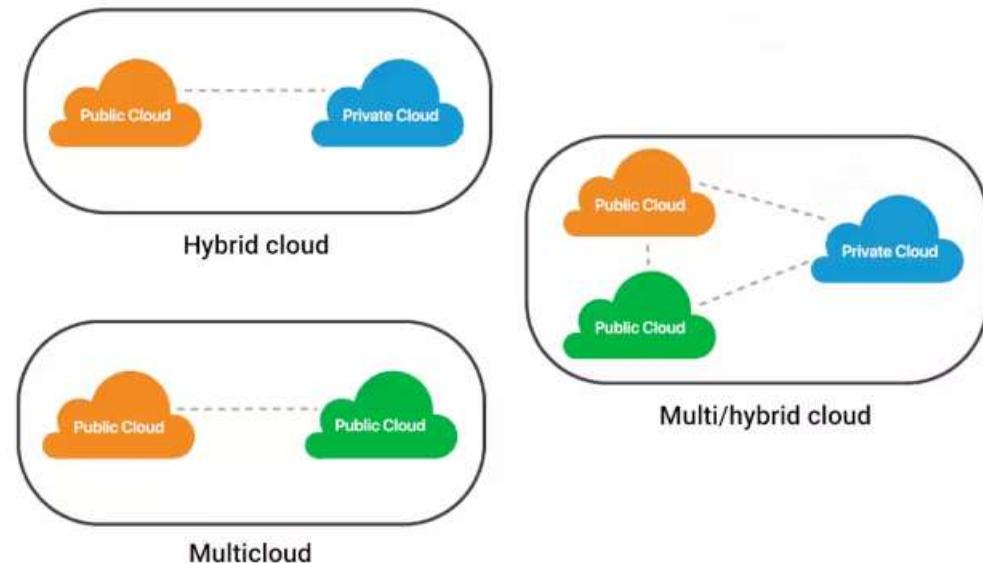
- 멀티 클라우드(Multi Cloud)



# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 기업

- 멀티 클라우드(Multi Cloud)



# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 조사

- Naver 클라우드 구성 → 검색, 포털
- KT 클라우드 구성 → 통신
- 금융 기업 클라우드 구성 → 금융

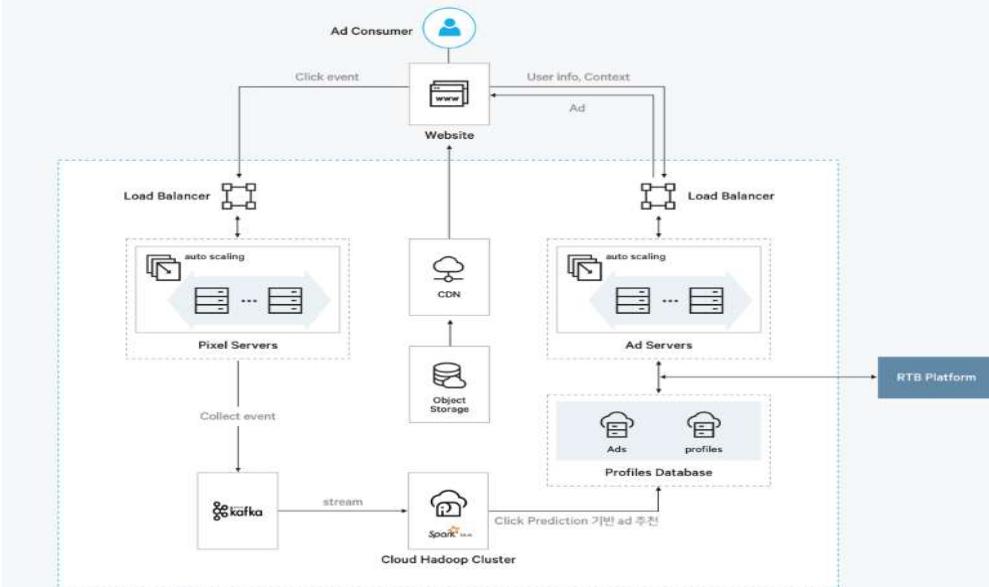


# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 조사

### ▪ Naver 클라우드 구성

<https://www.ncloud.com/>

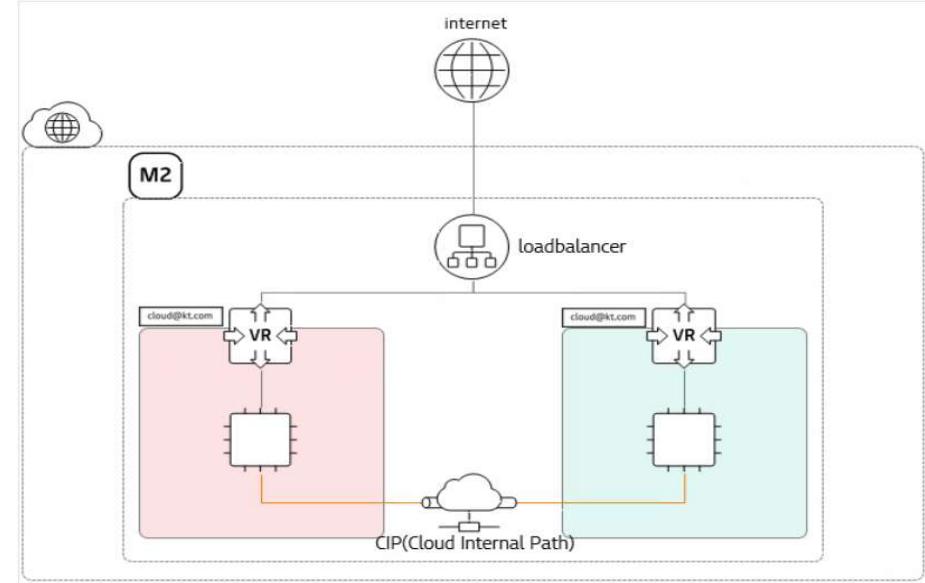


# ABOUT CLOUD SERVICE

## ◆ 클라우드 서비스 조사

- KT 클라우드 구성

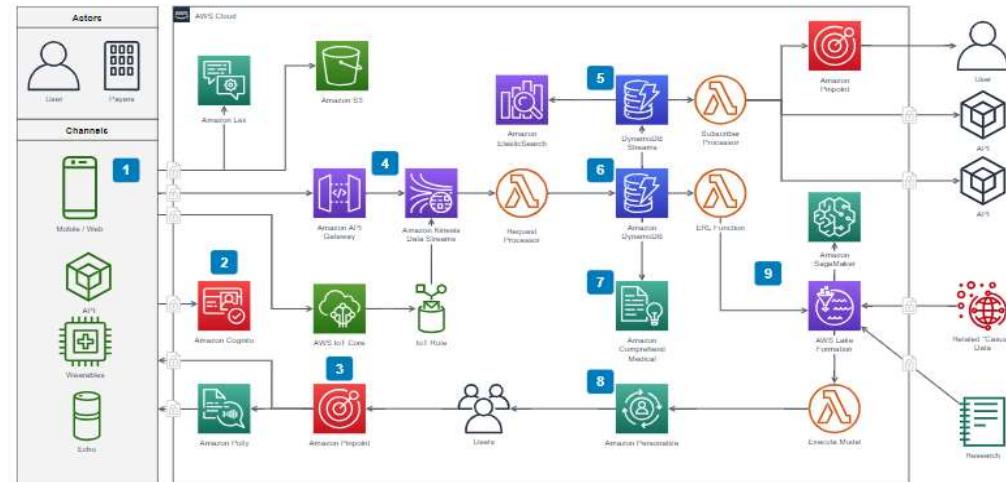
<https://tech.ktcloud.com/>



# ABOUT CLOUD SERVICE

## ◆ 클라우드 시스템 아키텍쳐 구성도 도구

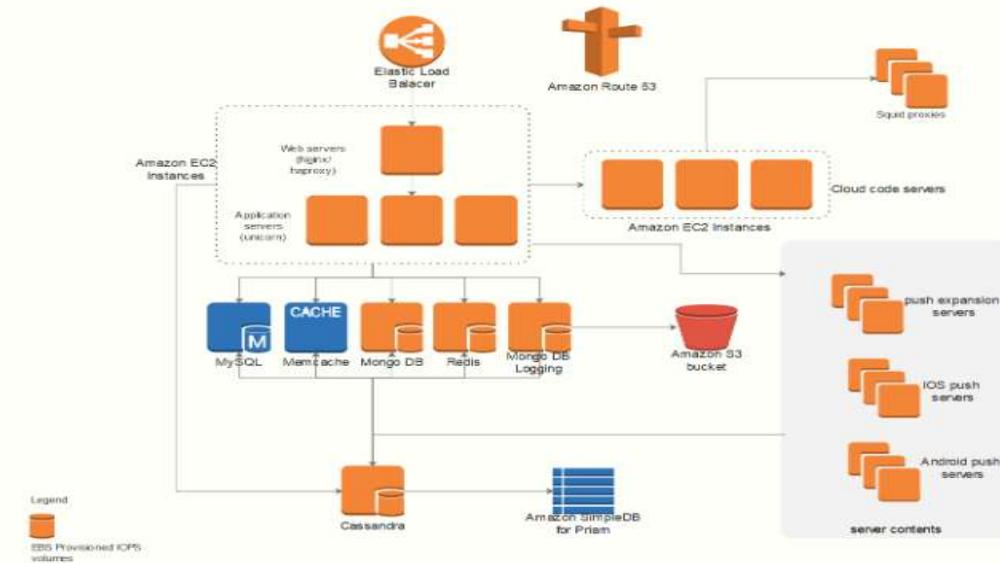
- [draw.io](https://www.drawio.com/) : <https://www.drawio.com/>



# ABOUT CLOUD SERVICE

## ◆ 클라우드 시스템 아키텍쳐 구성도 도구

- [이드로우 맥스\(EdrawMax\)](#)

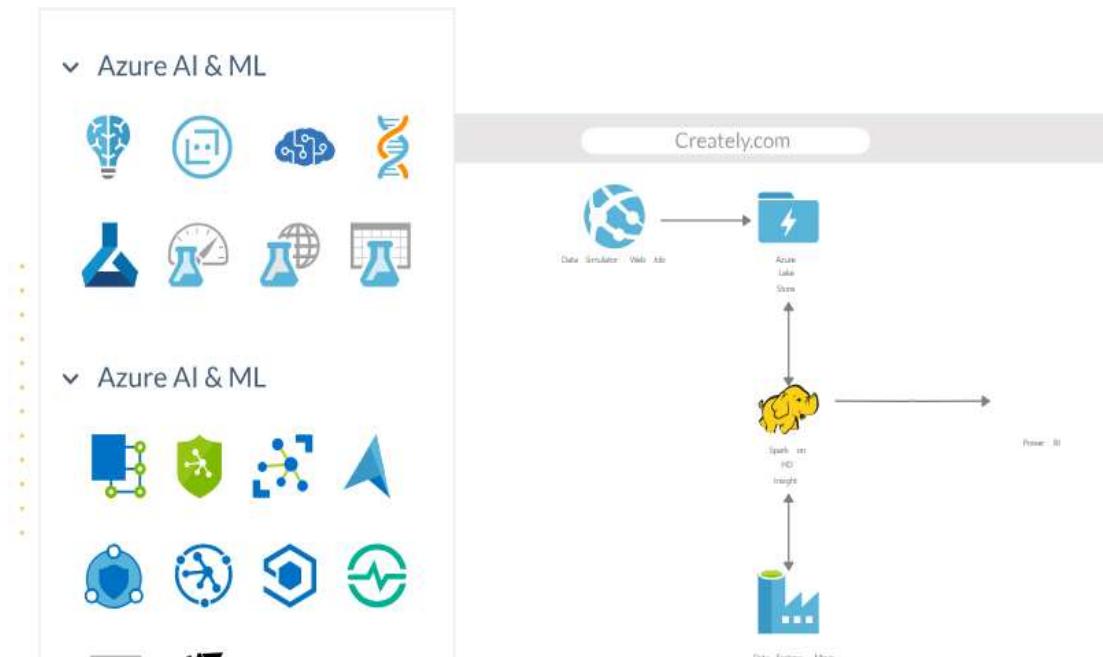


# ABOUT CLOUD SERVICE

## ◆ 클라우드 시스템 아키텍쳐 구성도 도구

- **creately**

<https://creately.com/ko/home/>





**PART I**

**ABOUT AWS**

# ABOUT AWS

---

## ◆ Amazon

- 전자상거래, AWS 이름의 클라우드 컴퓨팅 서비스 제공하고 있는 미국기업
- 1994년에 인터넷에서 책을 파는 아이디어로 제프 베조스가 워싱턴에서 설립
- 전 세계 최초로 전자상거래 서비스를 만든 기업들 중 하나



# ABOUT AWS

---

## ◆ AWS(Amazon Web Services) 란?

- 웹사이트 통해 제공 서비스로 일상에서 이용하는 많은 인터넷 서비스 → **Web Services**
- **컴퓨팅 자원(Computing resources)** 빌려 줌으로써 실제 컴퓨터 없이도 컴퓨터 자원 이용할 수 있게 해 주는 서비스
- 24시간 원격 접속 가능한 서버 통해 컴퓨팅 자원 활용

# ABOUT AWS

---

## ◆ AWS 용도 및 특징

- 실시간 데이터(raw data) 처리
  - 배치 데이터(batch data) 처리
- 
- 사용자 친화적 → 누구나 간단하게 구축 가능
  - 저렴한 비용 (GCP, Azure보다 상대적)
  - 민첩성, 즉각적 탄력성
  - 개방성과 유연성
  - 뛰어난 보안
  - 확장성

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Compting Service

서비스/리소스 이름	특징
<b>EC2 (Elastice Compute Cloud)</b>	<ul style="list-style-type: none"><li>AWS에서 가장 기본적 널리 쓰이는 인프라</li><li>물리 환경의 컴퓨터처럼 컴퓨팅 리소스 제공하는 서비스</li><li>컴퓨팅 파워의 규모 자유자재로 변경 가능</li><li>가상머신으로 제공 → <b>인스턴스(Instance)</b>라 함</li><li>안전성 위해 <b>리전(Reion)</b>과 <b>가용 영역(Availability Zone)</b>에 걸쳐 <b>배포</b></li></ul>
<b>Auto Scaling</b>	<ul style="list-style-type: none"><li>트래픽 따라 EC2 인스턴스들 자동 확장/ 제거 서비스</li><li>특정 트래픽 초과 시 자동 EC2 인스턴스 생성</li><li>트래픽 감소 시 생된된 EC2 인스턴스 삭제</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Compting Service

서비스/리소스 이름	특징
Lambda(Serverless Computing)	<ul style="list-style-type: none"><li>모든 유형의 애플리케이션이나 백엔드 서비스에 대한 코드를 별도의 관리 없이 실행하는 서비스</li><li>사용자는 서버 걱정없이 코드만으로 서비스 실행</li><li><b>serverless 아키텍쳐 구현에 사용</b></li><li>서버 및 운영 체제 유지 보수, 용량 프로비저닝 및 자동 확장, 코드 모니터링 및 로깅과 같은 컴퓨팅 리소스의 <b>모든 관리를 자체적으로 수행</b></li><li>Lambda 사용하는 언어 중 하나로 코드 제공만 하면 됨</li></ul>
Lightsail	<ul style="list-style-type: none"><li>주어진 리소스 옵션(Ubuntu, Node, Lamp stack, Nginx, WordPress, Django.. etc) 중 하나 택하여 단일 가상 서버를 쉽게 설정</li><li>프로젝트를 빠르게 시작하는 데 필요한 가상머신, SSD기반 스토리지, 데이터 전송, DNS관리, 정적IP가 포함</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Compting Service

서비스/리소스 이름	특징
<b>WorkSpaces</b>	<ul style="list-style-type: none"><li>데스크톱 가상화 서비스로 사내 PC 가상화로 구성</li><li>문서 및 데이터 개인 PC 보관하지 않고 서버 보관 관리</li><li>하드웨어 인벤토리, OS 버전 및 패치, 가상 데스크톱 인프라(VDI)를 관리하는 복잡성 제거, PC 제공 전략 간소화</li></ul>
<b>ECS (EC2 Container Service)</b>	<ul style="list-style-type: none"><li>클라우드 서버인 EC2를 Docker 컨테이너로 관리 가능하도록 나온 서비스</li><li>고도로 안전하고, 안정적이고, 확장 가능한 방식</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Compting Service

서비스/리소스 이름	특징
EB (Elastice Beanstalk)	<ul style="list-style-type: none"><li>Java, .NET, PHP, Node.js, Python, Ruby, Go, Docker를 사용하여 Apache, Nginx, Passenger, IIS와 같은 친숙한 서버에서 개발된 웹 애플리케이션 및 서비스를 간편하게 배포 조정할 수 있는 서비스</li><li>간단한 서비스 배포용으로 사용</li><li>코드 업로드 하면 용량 프로비저닝, 로드 밸런싱, Auto Scaling부터 시작하여 애플리케이션 상태 모니터링에 이르기까지 자동 처리</li><li>Heroku와 비슷한 PaaS 같은 서비스. 정확히 PaaS는 아니지만 앱을 배포하는 점에 있어서 더 수월한 서비스</li><li>Lightsail과 비슷한 범주</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Networking Service

서비스/리소스 이름	특징
<b>VPC</b> (Virtual Private Cloud)	<ul style="list-style-type: none"><li>클라우드 가상 네트워크 구축 서비스</li><li>IP 주소 범위 선택, 서브넷 생성, 라우팅 테이블 및 네트워크 게이트웨이 구성 등 가상 네트워킹 환경 완벽하게 제어할 수 있고, VPC에서 IPv4와 IPv6를 모두 사용하여 리소스와 애플리케이션에 안전하고 쉽게 액세스할 수 있음</li></ul>
<b>Route53</b>	<ul style="list-style-type: none"><li>Domain Name System (DNS) 도메인 관리/설정 서비스 (이름만 색다르지 보통 웹서버의 dns 구성과 같다.)</li><li>도메인 이름 구매/관리, 도메인에 대한 DNS 설정 자동으로 구성</li><li>사용자를 AWS외부 인프라로 전달하는 서비스 가능</li><li>EC2 인스턴스, Elastic 로드 밸런서, S3 저장소 등 AWS 서비스 인프라에 효과적으로 연결</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Networking Service

서비스/리소스 이름	특징
<b>Direct Connect</b>	<ul style="list-style-type: none"><li>기존 On-Premise의 인프라와 AWS간 연결을 쉽게 설정할 수 있는 클라우드 서비스 솔루션</li><li>전용선을 구성하여 낮은 지연 시간으로 데이터 및 정보를 공유할 수 있게 하는 서비스 제공</li><li>AWS-On-Premise를 연결하는 전용 네트워크 선 서비스</li></ul>
<b>ELB(Elastic Load Balancing)</b>	<ul style="list-style-type: none"><li>접속량이 많을 경우 L4 서비스(load balancing) 트래픽을 분산해주는 역할을 하여 고가용성 서비스 구축 서비스</li><li>들어오는 애플리케이션 트래픽을 Amazon EC2 인스턴스, 컨테이너, IP 주소, Lambda 함수와 같은 여러 대상에 자동 분산</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Networking Service

서비스/리소스 이름	특징
CloudFront	<ul style="list-style-type: none"><li>데이터, 동영상, 애플리케이션 및 API를 전 세계 고객에게 안전하게 전송하는 고속 글로벌 콘텐츠 전송 네트워크(CDN) 서비스</li><li>AWS의 CDN(Content Delivery Network, 콘텐츠 전송 네트워크)</li><li><b>S3, EC2, Elastic Load Balancing, Route 53 등과 같은 AWS 서비스와 통합 운영</b></li><li>리전에 상관없이 엣지 로케이션 기준으로 가장 가까운 곳에서 파일 캐시를 가져오기 때문에 속도도 빠르며 비용도 EC2 혹은 S3로 서비스를 제공하는 것 보다 더 저렴</li></ul>
Transit Gateway	<ul style="list-style-type: none"><li>VPC 및 계정 연결 손쉽게 확장</li><li>VPC와 온프레미스 네트워크 단일 게이트웨이에 연결 지원 서비스</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Storage Service

서비스/리소스 이름	특징
S3 (Simple Storage Service)	<ul style="list-style-type: none"><li>정적 파일 스토리지 서비스 (사진, 비디오, 문서 등 또는 frontend 코드, Lambda 함수 코드 해당)</li><li>사용자는 URL을 통해 파일을 사용</li><li>다른 유저들의 액세스를 컨트롤할 수 있는 기능 제공</li><li>HTTP 프로토콜과 연동되어 정적 사이트 호스팅</li><li>CloudFront 구성하면 S3에 저장된 정적 파일이 CDN을 통해 더 효율적으로 빠르게 보급되는 장점</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Storage Service

서비스/리소스 이름	특징
EBS (Elastic Block Store)	<ul style="list-style-type: none"><li>EC2 인스턴스에 장착하여 사용할 수 있는 가상 저장 장치 (HDD 나 SSD 처럼 가상 컴퓨터에 장착한 저장장치)</li><li>EC2 인스턴스 제공 기본 용량보다 더 사용해야 할 때, 운영체제 중단시키지 않고 자유롭게 늘리고 싶을 때, 영구적인 데이터 보관 필요할 때, RAID 등의 고급 기능 필요할 때 사용</li><li>EC2에 설치된 OS에서 그냥 일반적인 하드디스크 또는 SSD처럼 인식되어 원하는 크기로 만들 수 있고, 성능 (IOPS) 또한 원하는 수치로 설정할 수 있으며 사용자가 삭제하기 전까지 데이터가 안전하게 유지</li><li>EBS를 연결하여 EBS에 파일을 저장한다면 EC2 인스턴스 와 관계 없이 영구적으로 보관이 가능</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Storage Service

서비스/리소스 이름	특징
Glacier	<ul style="list-style-type: none"><li>데이터 보관, 백업 및 아카이브를 위한 스토리지 서비스</li><li>저장에만 특화되어있는 저렴한 스토리지 서비스</li><li>거의 무제한으로 데이터를 저장 가능</li><li>저장하고 꺼내는데 3시간-5시간 걸린다는 특징</li><li>S3에서 -&gt; Glacier로 백업을 자동 생성하도록 설정 가능.</li><li>스토리지 비용이 저렴하며 데이터에 밀리초 만에 액세스 해야 할 필요가 없다면 이 제품 사용</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Storage Service

서비스/리소스 이름	특징
<b>Storage Gateway</b>	<ul style="list-style-type: none"><li>On-Premise에 있는 데이터를 클라우드로 저장 보관하기 위한 게이트웨이 연결 서비스</li></ul>
<b>Snowball</b>	<ul style="list-style-type: none"><li>Import/Export 서비스를 통해 대량의 데이터를 AWS로 이전할 때 네트워크로 전송하지 않고 디스크나 스토리지에 저장하여 물리적으로 전달하고 이를 업로드 하여 주는 서비스</li><li>대량의 데이터를 AWS로 업로드 할 때 유용</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Database Service

서비스/리소스 이름	특징
RDS(Relational DB Service)	<ul style="list-style-type: none"><li>관계형 데이터베이스 이용할 수 있는 서비스</li><li>DB 설정, 패치, 백업 등 시간 소모적인 관리 작업 처리</li><li>RDBMS 클라우드 서비스 : Amazon Aurora, MySQL, MariaDB, PostgreSQL, Oracle, SQL Server등을 지원</li></ul>
DynamoDB	<ul style="list-style-type: none"><li>어떠한 규모에서도 10ms 미만 성능 제공하는 <b>key-value 형태의 NoSQL 데이터베이스 서비스</b></li><li>데이터 규모와 관계없이 데이터 저장 및 검색, 어떤 수준의 요청 트래픽이라도 처리할 수 있는 데이터베이스 테이블을 생성할 수 있음</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Database Service

서비스/리소스 이름	특징
ElasticCache	<ul style="list-style-type: none"><li>Database Caching 서비스.</li><li>Memcached, Redis 호환 지원</li></ul>
DocumentDB	<ul style="list-style-type: none"><li><a href="#">MongoDB</a>와 호환되는 데이터베이스를 쉽게 설정, 운영 및 조정할 수 있는 데이터베이스 서비스</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Management Tools

서비스/리소스 이름	특징
<b>CloudWatch</b>	<ul style="list-style-type: none"><li>- AWS 서비스들을 모니터링, 알람 받는 설정들 할 수 있는 서비스</li><li>- 특정 금액 초과할 경우 알람을 받거나 EC2의 CPU 사용률등의 알람</li></ul>
<b>CloudFormation</b>	<ul style="list-style-type: none"><li>- AWS 서비스 생성 및 배포 자동화 템플릿 서비스</li><li>- 다양한 서비스들을 이용하여 아키텍쳐 구현시 미리 만들어놓은 템플릿(JSON)을 이용하여 생성, 직접 템플릿 작성하여 관리 가능</li></ul>
<b>IAM</b>	<ul style="list-style-type: none"><li>- Identity and Access Management 약자</li><li>- AWS 리소스 액세스 안전하게 제어할 수 있는 서비스</li><li>- 사용자 및 애플리케이션별 제어 권한 설정</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Analytics Platform

서비스/리소스 이름	특징
Kinesis	<ul style="list-style-type: none"><li>- 대량의 데이터를 저장 분류할 수 있는 서비스</li><li>- 다양한 규모의 스트리밍 데이터를 비용 효율적 처리할 수 있는 기능</li></ul>
Redshift	<ul style="list-style-type: none"><li>- 효율적으로 비용 및 데이터 분석할 수 있는 빠르고 확장 가능한 데이터 웨어하우스</li><li>- 기계학습, 다량 병렬 쿼리 실행, 고성능 디스크의 열 기반 스토리지를 사용하여 다른 데이터 웨어하우스 보다 10배 빠른 성능 제공</li><li>- 몇백 GB부터 페타바이트 규모 이상의 데이터 세트에 최적화</li></ul>
EMR	<ul style="list-style-type: none"><li>- Elastic MapReduce</li><li>- 저장된 대량 데이터를 분류하고 분석하여 필요한 정보 추출 서비스</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - Application Service

서비스/리소스 이름	특징
<b>CloudSearch</b>	<ul style="list-style-type: none"><li>- 검색서비스로 손 쉽게 중요 정보를 모바일로 전달</li><li>- SWF: 워크플로우 서비스</li><li>- SQS: 큐서비스 활용한 대량 데이터 제공 서비스</li></ul>
<b>SES (Simple Email Services)</b>	<ul style="list-style-type: none"><li>- 외부로 대량의 메일을 발송하는 서비스</li></ul>
<b>Elastice Transcoder</b>	<ul style="list-style-type: none"><li>- 동영상을 인코딩 할 수 있는 서비스</li></ul>

# ABOUT AWS

---

## ◆ AWS 제공 서비스/리소스 - AWS 가격정책

정책	특징
<b>On-Demand</b>	- 기본적 사용하는 과금 방식, 사용한 시간 만큼 비용 지불하는 형태
<b>Reserved</b>	- 일정 기간 인스턴스 사용 약속하고, 그에 대한 할인 적용받는 방식
<b>Spot</b>	<ul style="list-style-type: none"><li>- 입찰 방식의 사용방법, 사용자가 입찰 가격을 제시해놓으면 아마존에서 남는 인스턴스들에 대해서 Spot 가격 책정</li><li>- 가격이 입찰가격 내로 들어오면 인스턴스 기동되는 방식</li><li>- 입찰 가격이 넘어가면 자동으로 Spot Instance는 다시 종료</li><li>- 항상 실행시키는 업무가 아닌 특정 작업 배치 돌릴 서버용도 사용 적합</li></ul>

# ABOUT AWS

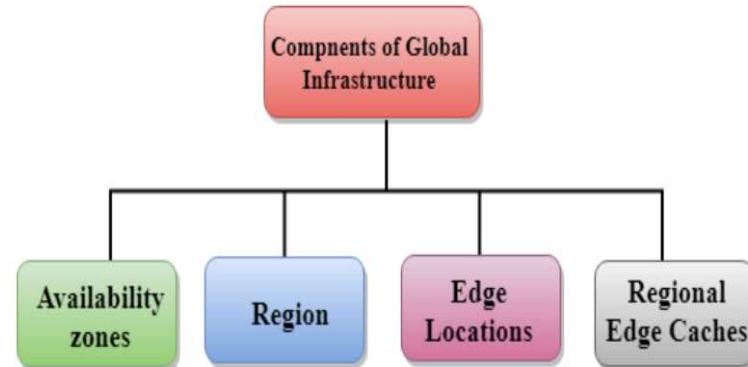
## ◆ AWS 글로벌 인프라

- 글로벌 서비스

- 한국 미국 영국 상관없이 전세계에서 공통적으로 사용하는 인프라

- 구성

- 리전(Region)
- 가용영역(AZ, Availability Zones)
- 엣지 로케이션(Edge Location)
- 리전 엣지 캐시(Regional Edge Cache)





**PART I**

**USING AWS**

# USING AWS

## ◆ AWS 계정

### ▪ Free Tier Account

- 일부 AWS 서비스 특정 사용량 한도 내에서 무료 사용 가능 계정
- 가입일로부터 **12개월** 후에 만료
- 무료 사용 만료되거나, 애플리케이션 사용량 범위 초과할 경우,  
    ➔ 종량제 표준 요금 지불
- 간단한 아키텍처를 구성할 수 있으므로, AWS를 직접 다뤄보는데 큰 도움

# USING AWS

## ◆ AWS 회원가입

### ▪ 루트 사용자(Root User)



새로운 AWS 계정으로 프리 티어 제품  
을 살펴보세요.

자세히 알아보려면 [aws.amazon.com/free](http://aws.amazon.com/free)를 방  
문하세요.



### AWS에 가입

루트 사용자 이메일 주소  
계정 복구 및 일부 관리 기능에 사용

AWS 계정 이름  
계정의 이름을 선택합니다. 이름은 가입 후 계정 설정에  
서 변경할 수 있습니다.

[이메일 주소 확인](#)

또는

[기존 AWS 계정에 로그인](#)

# USING AWS

## ◆ AWS 회원가입

### ▪ 루트 사용자(Root User)



새로운 AWS 계정으로 프리 티어 제품  
을 살펴보세요.

자세히 알아보려면 [aws.amazon.com/free](http://aws.amazon.com/free)를 방  
문하세요.



### AWS에 가입

#### 사용자 확인

보안 유지 – AWS가 하는 일입니다.

확인 코드가 포함된 이메일을  
[anece00@naver.com](mailto:anece00@naver.com)(으)로 보냈습니다. (귀하  
의 계정이 아닙니까?)

이메일을 확인하려면 아래에 입력하세요.

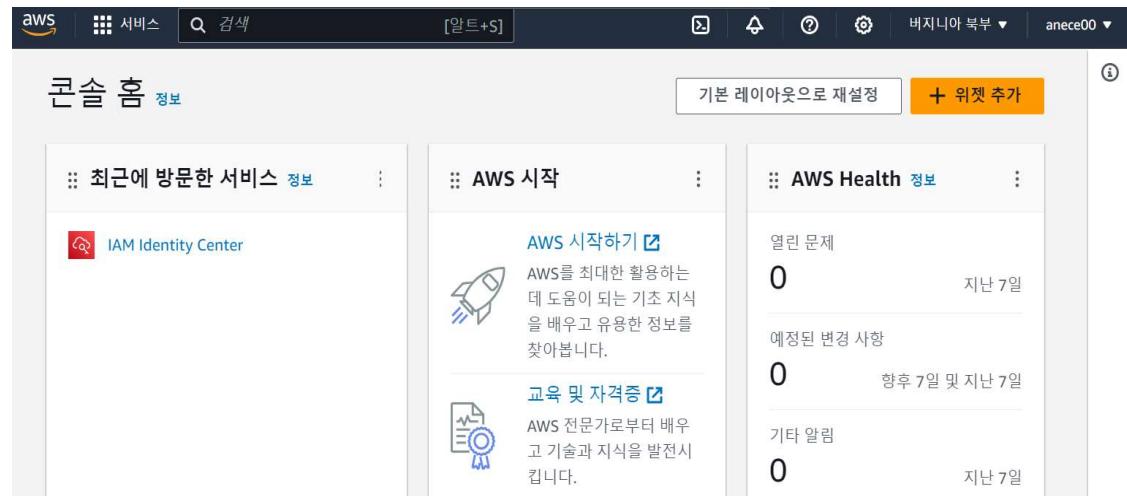
확인 코드

확인

# USING AWS

## ◆ AWS Management Console

- AWS 리소스 관리 위한 **다양한 서비스 콘솔 모음을 구성하고 참조하는 웹 애플리케이션**
- 로그인하면 콘솔 홈 페이지 표시



**PART II**

**EC2**  
**Elastic Compute Cloud**

# EC2:Elastic Compute Cloud

## ◆ EC2

- 온-디맨드 확장 가능 컴퓨팅 용량 제공 서비스로 인스턴스(Instance)라 함
- 원하는 수의 **가상 서버 구축**하고 **보안 및 네트워킹 구성**하며 **스토리지 관리** 가능
- 애플리케이션을 더욱 빠르게 개발하고 배포 가능해짐
- 유형 : 서버용, 머신러닝용, 게임용 등 타입별로 부여
- 비용 : 온디맨드, 리저브드, 스팟

# EC2:Elastic Compute Cloud

## ◆ EC2

- 온-디맨드 확장 가능 컴퓨팅 용량 제공 서비스로 인스턴스(Instance)라 함
- 원하는 수의 **가상 서버 구축**하고 **보안 및 네트워킹 구성**하며 **스토리지 관리** 가능
- 애플리케이션을 더욱 빠르게 개발하고 배포 가능해짐
- 유형 : 서버용, 머신러닝용, 게임용 등 타입별로 부여
- 비용 : 온디맨드, 리저브드, 스팟

# EC2:Elastic Compute Cloud

## ◆ EC2 용도 및 제한 조건

- <https://aws.amazon.com/ko/ec2/features/>

◀ 페이지 콘텐츠

글로벌 인프라

비용 및 용량 최적화

스토리지

네트워킹

운영 체제 및 소프트웨어

유지 관리

용도 및 제한 조건

### 용도 및 제한 조건

이 서비스 사용에는 [Amazon Web Services 고객 계약](#)이 적용됩니다.



#### 인스턴스 유형 알아보기

인스턴스 유형 페이지로 이동

[자세히 알아보기 »](#)

#### 무료 계정에 가입

AWS 프리 티어에 즉시 액세스할 수 있습

니다.

[가입 »](#)

#### 콘솔에서 구축 시작

AWS 콘솔에서 Amazon EC2 구축을 시작

하십시오.

[시작하기 »](#)

# EC2:Elastic Compute Cloud

## ◆ EC2 인스턴스 유형

- 표기법



- 인스턴스 패밀리 : M, T, C, X, IO, I, D, G, P, F

T	간헐적 높은 성능, 소규모/마이크로	R	메모리 최적화	P	범용 CPU
M	범용, 중소형 DB, 앱	I	SSD기반 초고속 IO	F	FPGA 이용한 높은 컴퓨팅 앱
C	가장 높음 수준 컴퓨팅 성능	D	HDD기반 높은 디스크 처리량	G	그래픽 집약, GPU 이용
X	메모리 최적화, 대용량 메모리				

- 인스턴스 세대 : 패밀리에 따라 1 ~ 5 세대
- 인스턴스 크기 : 패밀리 및 세대에 따라 다양한 크기 제공

# EC2:Elastic Compute Cloud

## ◆ EC2 인스턴스 요소

- **AMI (Amazon Machine Image)**

- 서버에 필요한 구성 요소(OS와 추가 SW 포함)를 패키징하는 인스턴스용 사전 구성 템플릿
- EC2 복사본으로 여러 인스턴스 실행 가능
- OS : Amazon Linux2, CentOS, Red Hat Enterprise Linux, Window Server, Ubuntu

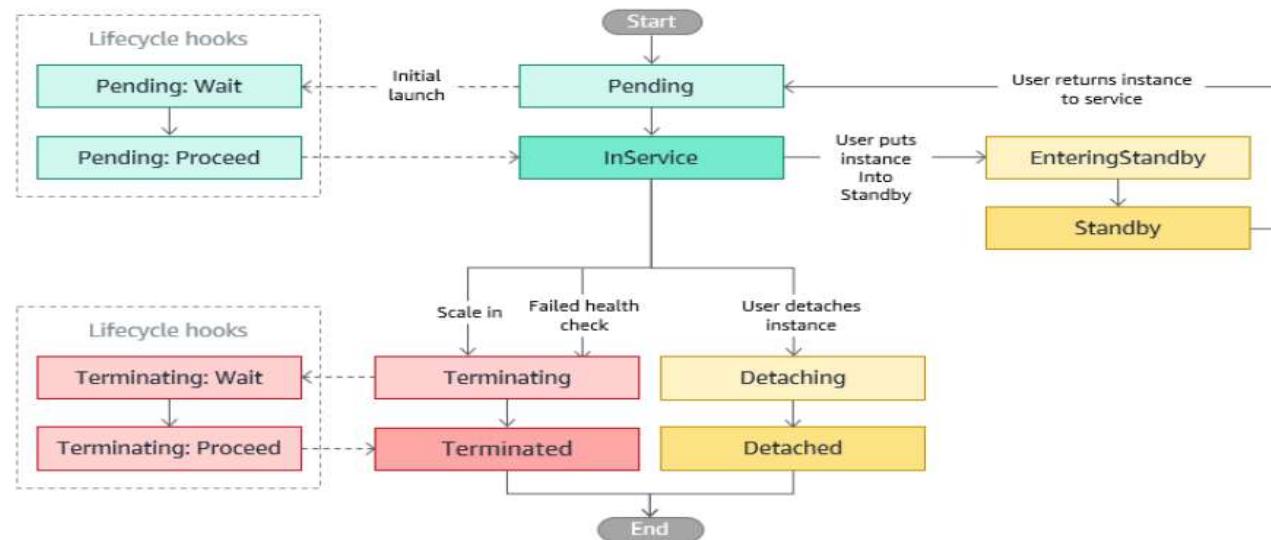
- **Key Pair**

- 인스턴스에 대한 보안 로그인 정보
- 퍼블릭 키를 저장하고 사용자는 프라이빗 키를 안전한 장소에 저장

# EC2: Elastic Compute Cloud

## ◆ EC2 인스턴스 생명주기

- 인스턴스 시작되는 즉시 인스턴스에 대한 요금 청구, 서비스되지 않는 시간도 포함



# EC2:Elastic Compute Cloud

## ◆ EC2 인스턴스 생명주기

- 인스턴스 시작되는 즉시 인스턴스에 대한 요금 청구, 서비스되지 않는 시간도 포함



# EC2:Elastic Compute Cloud

## ◆ EC2 인스턴스 생명주기

- 상태별 인스턴스 요금

인스턴스 상태	설명	요금
pending	인스턴스가 running 상태로 전환할 준비	미청구
<b>running</b>	<b>인스턴스 사용 중</b>	<b>청구</b>
stopping	인스턴스 중지 중	미청구
<b>stopping</b>	<b>인스턴스 최대절전 모드로 전환 중</b>	<b>최대절전 시 청구</b>
stopped	인스턴스 중지 상태 ➔ 재시작 가능 상태	미청구
shutting-down	인스턴스가 종료 중	미청구
terminated	인스턴스가 영구적으로 삭제됨	미청구

# EC2:Elastic Compute Cloud

## ◆ EC2 인스턴스 시작

### ▪ 상태별 인스턴스 요금

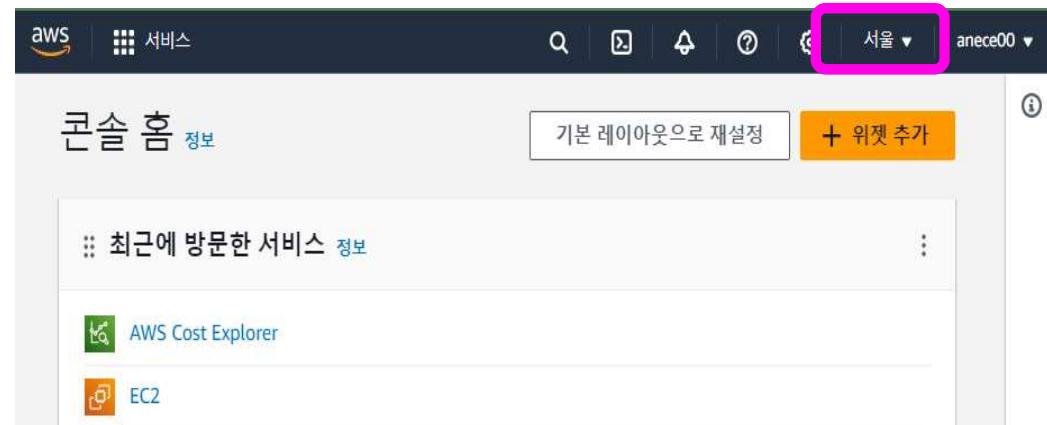
The screenshot shows the AWS EC2 Dashboard. On the left, a sidebar menu includes 'EC2 대시보드', 'EC2 글로벌 보기', '이벤트', '인스턴스' (selected), '인스턴스', '인스턴스 유형', '시작 템플릿', '스팟 요청', 'Savings Plans', '예약 인스턴스', '전용 호스트', '용량 예약', '이미지' (AMI, AMI 카탈로그), and 'Elastic Block Store' (볼륨, 스냅샷, 수명 주기 관리자). The main content area is titled '리소스' and displays resource counts: 0 instances (running), 0 launch templates, 0 volumes, 0 snapshots, 0 instances, 0 key pairs, 0 Auto Scaling groups, 0 load balancers, 1 security groups, 0 reserved instances, 0 dedicated hosts, and 0 static IPs. Below this is the '인스턴스 시작' (Launch Instance) section, which contains a large orange '인스턴스 시작' button and a '서버 마이그레이션' link. A '서비스 상태' sidebar shows 'AWS Health 대시보드' and '리전: 아시아 태평양 (서울)'. The top navigation bar includes tabs for '서비스' and '검색', and icons for '알트+S', search, refresh, help, settings, and account.

# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ 지역 설정

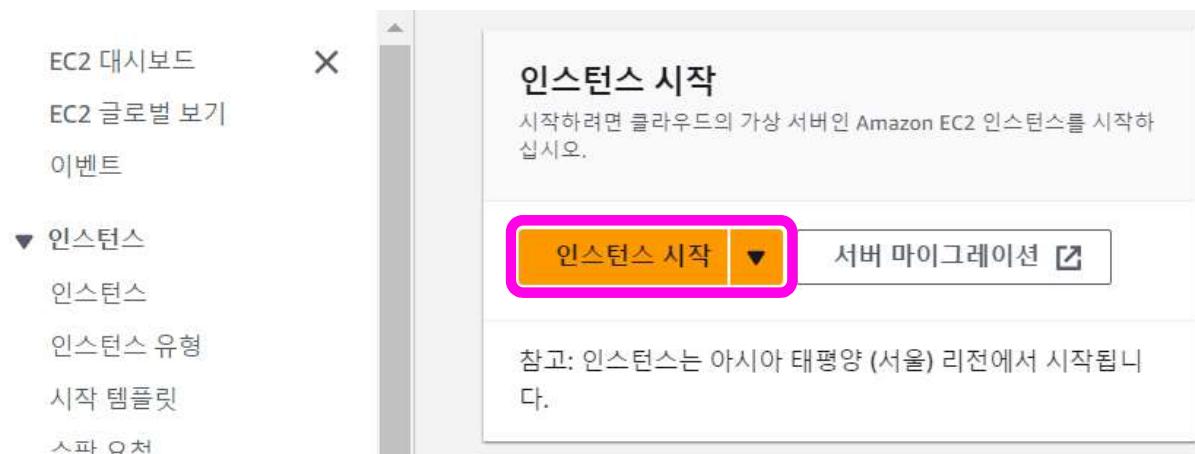
- 생성한 지역 국한 인스턴스 사용
- 지역에 따른 네트워크 속도
- 지역에 따른 지원 서비스



# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ 인스턴스 생성

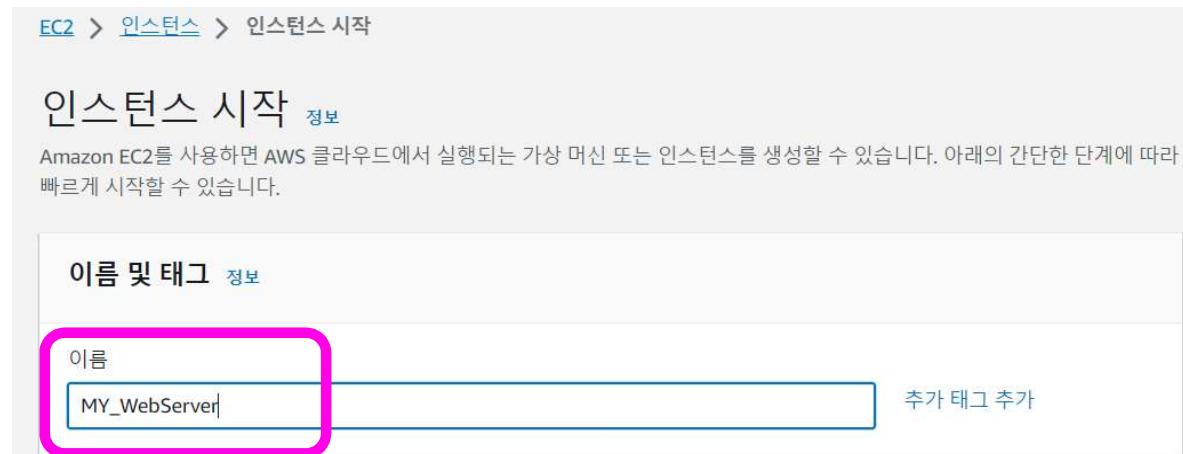


# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ 인스턴스 생성

- 이름 및 태그 설정



# EC2: Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ 인스턴스 생성

#### ➤ AMI 선택

Quick Start

Amazon Machine Image(AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type  
ami-086cae3329a3f7d75 (64비트(x86)) / ami-0f8536fc6ad2ba267 (64비트(Arm))  
가상화: hvm ENA 활성화됨: true 루트 디바이스 유형: ebs

프리 티어 사용 가능

설명  
Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-09-19

아키텍처  
64비트(x86)

AMI ID  
ami-086cae3329a3f7d75

확인된 공급 업체

# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ 유형 및 키 페어 선택

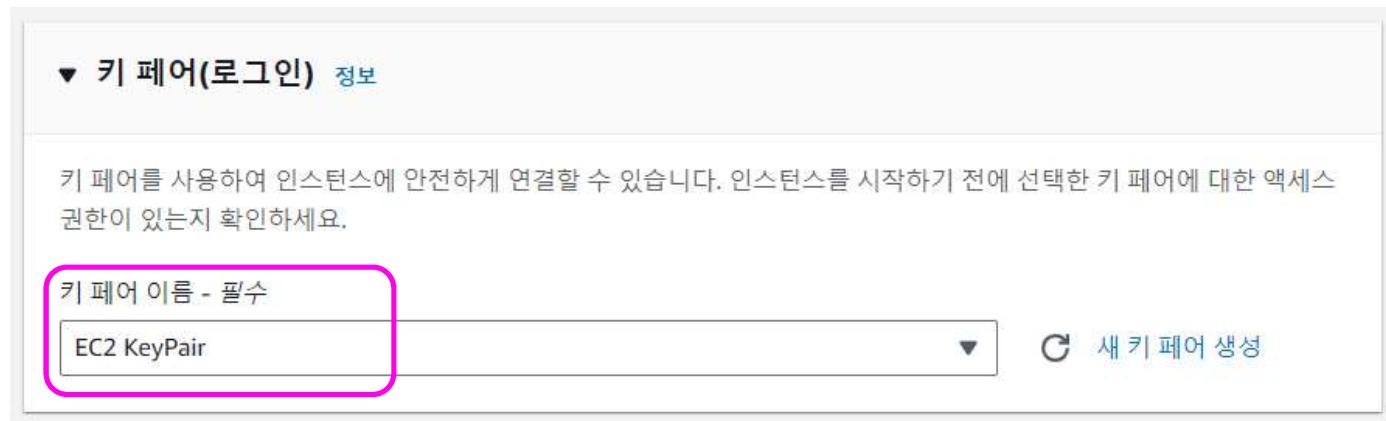
- EC2 접속 사용되는 암호화된 파일
  - 보안 위해 ID/PW 권장하지 않음
  - 키 페어 방식 접속 권장
- 
- 생성된 키페어 파일 다운로드
  - 한번만 다운로드 가능



# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

- 유형 및 키 페어 선택



# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ 네트워크 설정

- VPC(Virtual Private Cloud) 네트워크
- 세부적 설정 가능
- 기본 설정만으로도 운영 가능

- 보안그룹(SG) 생성
- 자신의 IP주소 설정 무방
- 코드 서버/도메인 할당 → HTTP(S)허용

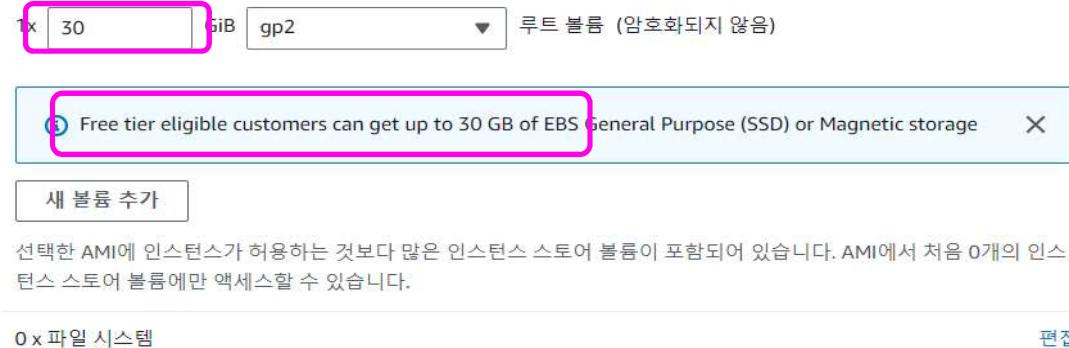


# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ 스토리지 구성 설정

- 디스크 용량 의미
- Tree Tier 최대 30GB



# EC2: Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ 클라이언트 접속

인스턴스 (1/3) 정보							작업 ▾
<input type="text"/> 인스턴스를 속성 또는 (case-sensitive) 태그로 찾기							인스턴스 상태 ▾
Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	경보 상태	가용 영역	
<input type="checkbox"/> MyWindow	i-098020deddd9aae6e	<input type="radio"/> 중지됨	<input type="radio"/>	t2.micro	-	경보 없음	+ ap-northeast-1
<input checked="" type="checkbox"/> UbuntuCom	i-0c17822967995e99f	<input checked="" type="radio"/> 실행 중	<input type="radio"/>	t2.micro	<input checked="" type="radio"/> 2/2개 검사 통과	경보 없음	+ ap-northeast-1

# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

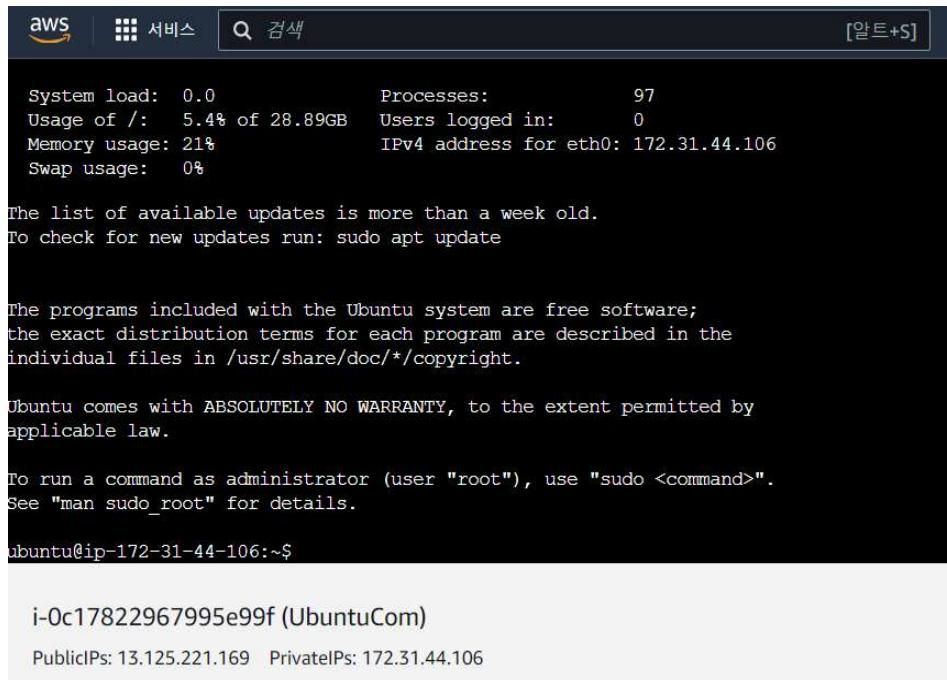
### ▪ EC2 기반 클라이언트 접속



# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

- EC2 기반 클라이언트 접속



The screenshot shows a terminal window within an AWS Lambda function. The terminal output is as follows:

```
aws | 서비스 | 검색 | [알트+S]
```

```
System load: 0.0          Processes:      97
Usage of /: 5.4% of 28.89GB  Users logged in:  0
Memory usage: 21%
Swap usage:  0%           IPv4 address for eth0: 172.31.44.106

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

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-172-31-44-106:~$
```

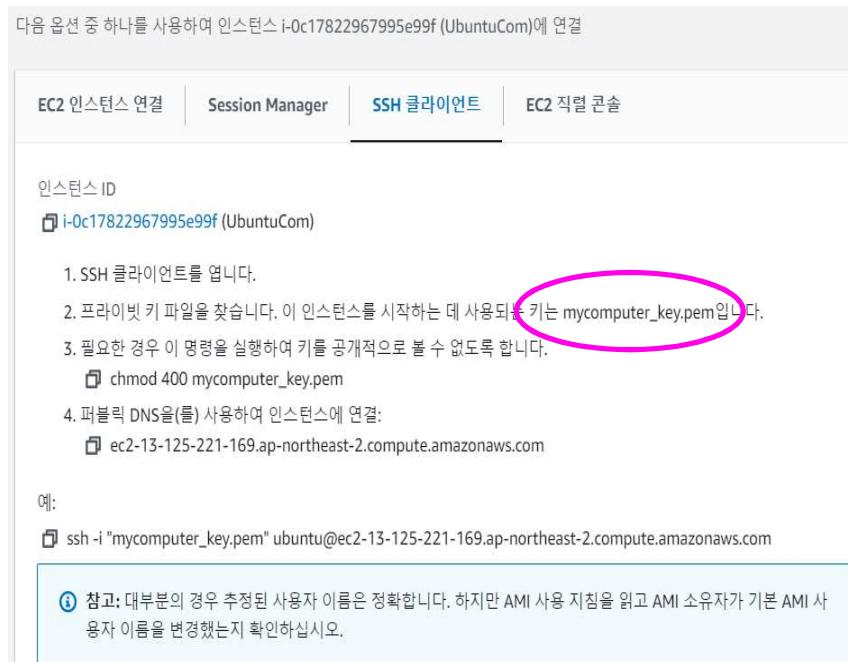
i-0c17822967995e99f (UbuntuCom)  
Public IPs: 13.125.221.169 Private IPs: 172.31.44.106

# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ SSH통신 기반 클라이언트 접속

- ① 전용 프로그램 준비
- ② PC상에서 keypair 암호화 해제
- ③ PC상에서 접속 프로그램 실행
- ④ PC상에서 접속 프로그램으로 접속

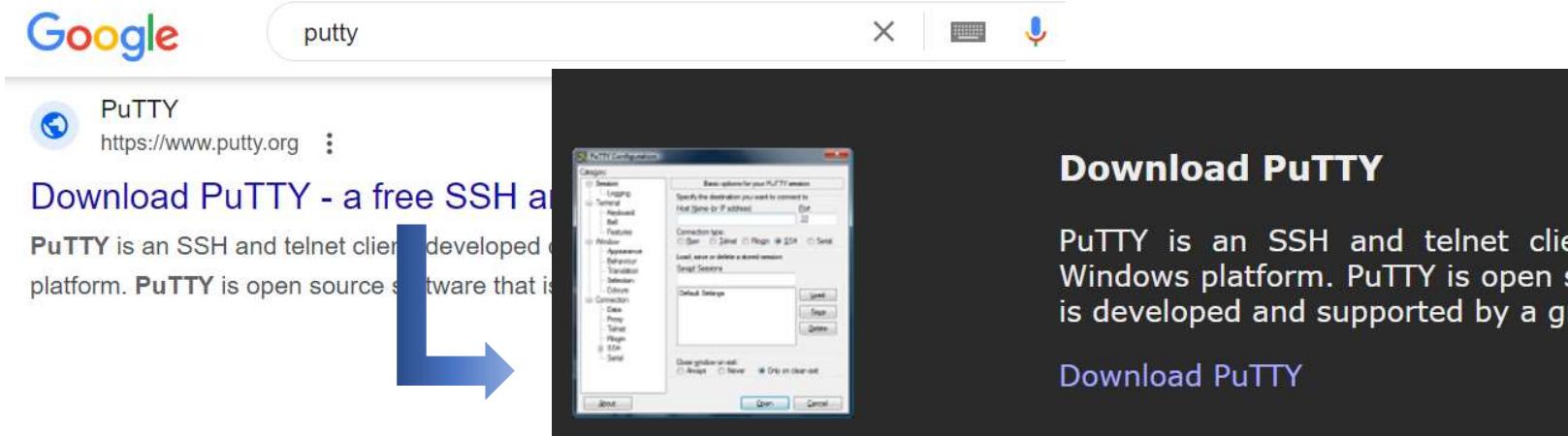


# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

- SSH통신 기반 클라이언트 접속

- ① 전용 프로그램 준비



# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ SSH통신 기반 클라이언트 접속

#### ① 전용 프로그램 준비

- 암호화 키 변환 → puttygen.exe
- SSH 통신 프로그램 → putty.exe

#### Alternative binary files

The installer packages above will provide versions of all of these (e prefer).

(Not sure whether you want the 32-bit or the 64-bit version? Read

##### putty.exe (the SSH and Telnet client itself)

64-bit x86:	<a href="#">putty.exe</a>	<a href="#">(signature)</a>
64-bit Arm:	<a href="#">putty.exe</a>	<a href="#">(signature)</a>
32-bit x86:	<a href="#">putty.exe</a>	<a href="#">(signature)</a>

##### pscp.exe (an SCP client, i.e. command-line secure file copy)

64-bit x86:	<a href="#">pscp.exe</a>	<a href="#">(signature)</a>
64-bit Arm:	<a href="#">pscp.exe</a>	<a href="#">(signature)</a>

##### puttygen.exe (a RSA and DSA key generation utility)

64-bit x86:	<a href="#">puttygen.exe</a>	<a href="#">(signature)</a>
64-bit Arm:	<a href="#">puttygen.exe</a>	<a href="#">(signature)</a>
32-bit x86:	<a href="#">puttygen.exe</a>	<a href="#">(signature)</a>

# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

- SSH통신 기반 클라이언트 접속

- ② PC상에서 keypair 암호화 해제
  - puttygen.exe 프로그램 실행

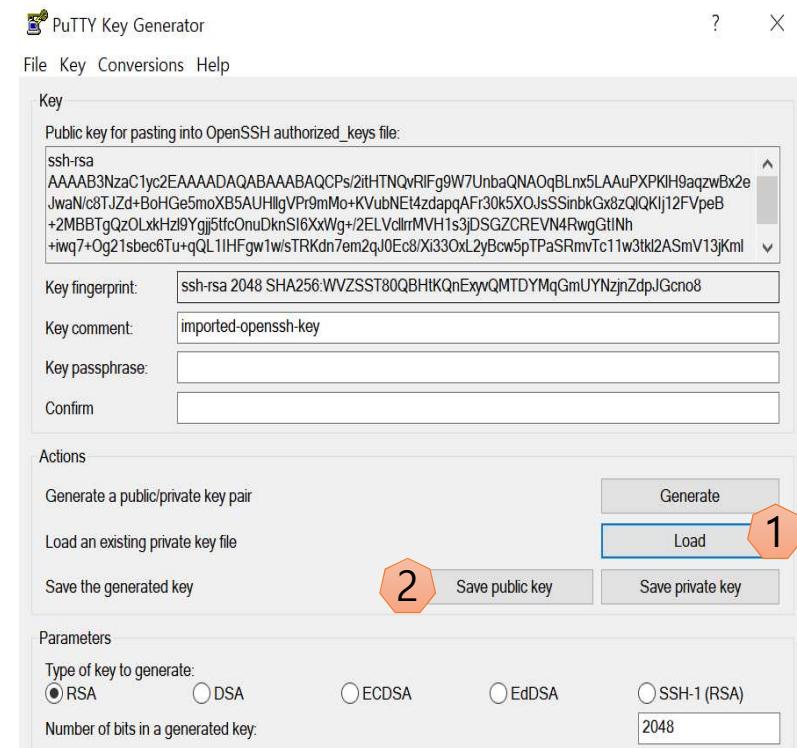


# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ SSH통신 기반 클라이언트 접속

- ② PC상에서 keypair 암호화 해제
  - puttygen.exe 프로그램 실행
  - ✓ Load버튼 → KeyPair 파일 선택
  - ✓ Save public key 버튼 클릭

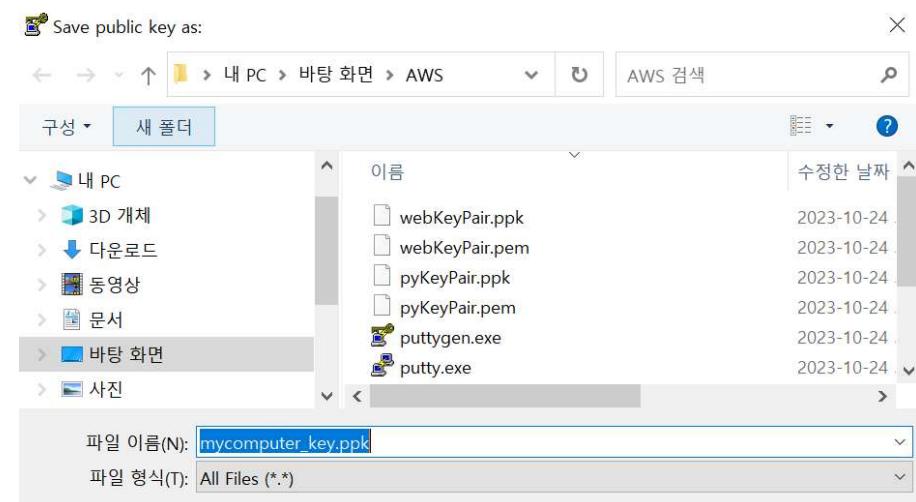


# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ SSH통신 기반 클라이언트 접속

- ② PC상에서 keypair 암호화 해제
  - puttygen.exe 프로그램 실행
  - ✓ **mycomputer\_key.ppk** 저장



# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

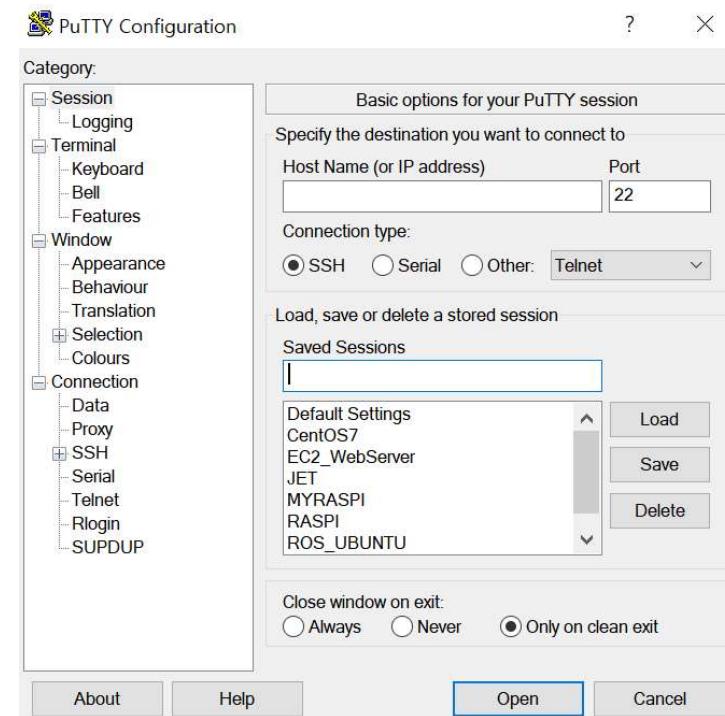
### ▪ SSH통신 기반 클라이언트 접속

③ PC상 접속 프로그램 실행

✓ putty.exe 프로그램 실행

✓ 접속 원격 서버 지정

✓ keypair 지정



# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

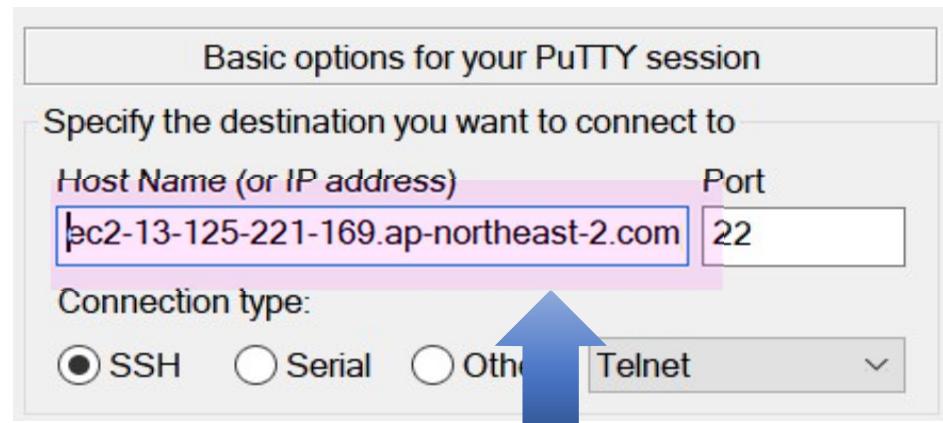
### ▪ SSH통신 기반 클라이언트 접속

#### ③ PC상 접속 프로그램 실행

✓ putty.exe 프로그램 실행

✓ 접속 원격 서버 지정

✓ keypair 지정



4. 퍼블릭 DNS을(를) 사용하여 인스턴스에 연결:

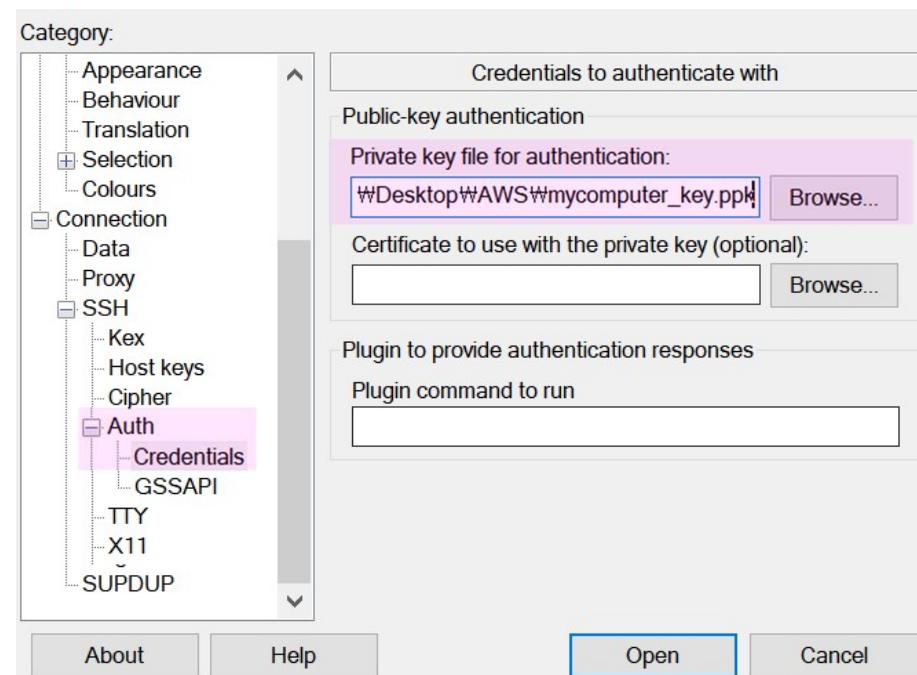
▫ ec2-13-125-221-169.ap-northeast-2.compute.amazonaws.com

# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ SSH통신 기반 클라이언트 접속

- ③ PC상 접속 프로그램 실행
  - ✓ putty.exe 프로그램 실행
  - ✓ 접속 원격 서버 지정
  - ✓ **keypair** 지정

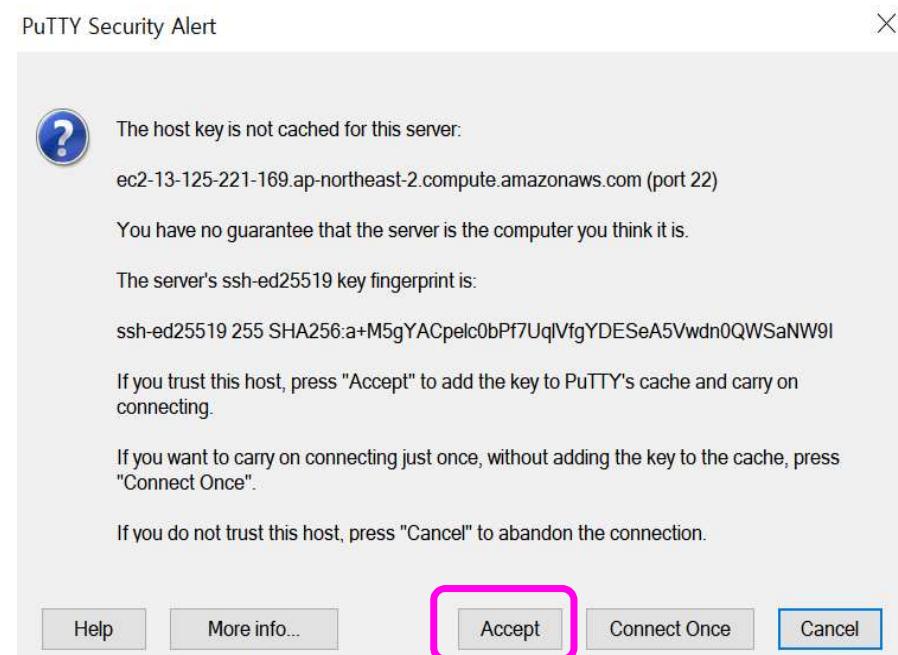


# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

### ▪ SSH통신 기반 클라이언트 접속

④ 접속 진행

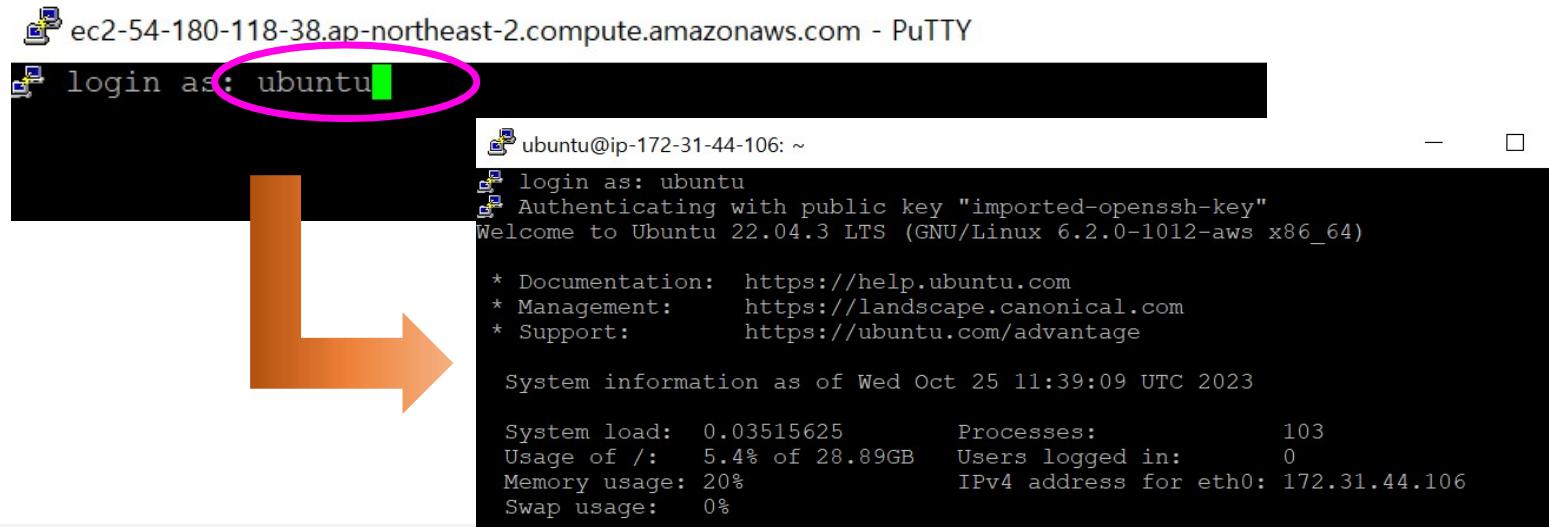


# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

- SSH통신 기반 클라이언트 접속

- ⑤ 접속 완료



```
ec2-54-180-118-38.ap-northeast-2.compute.amazonaws.com - PuTTY
login as: ubuntu

ubuntu@ip-172-31-44-106: ~
login as: ubuntu
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1012-aws x86_64)

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

System information as of Wed Oct 25 11:39:09 UTC 2023

System load: 0.03515625      Processes:          103
Usage of /: 5.4% of 28.89GB   Users logged in:    0
Memory usage: 20%           IPv4 address for eth0: 172.31.44.106
Swap usage: 0%
```

# EC2:Elastic Compute Cloud

## ◆ Linux OS EC2 인스턴스 생성

- SSH통신 기반 클라이언트 접속

- ⑤ 접속 완료

```
ubuntu@ip-172-31-44-106: ~
```

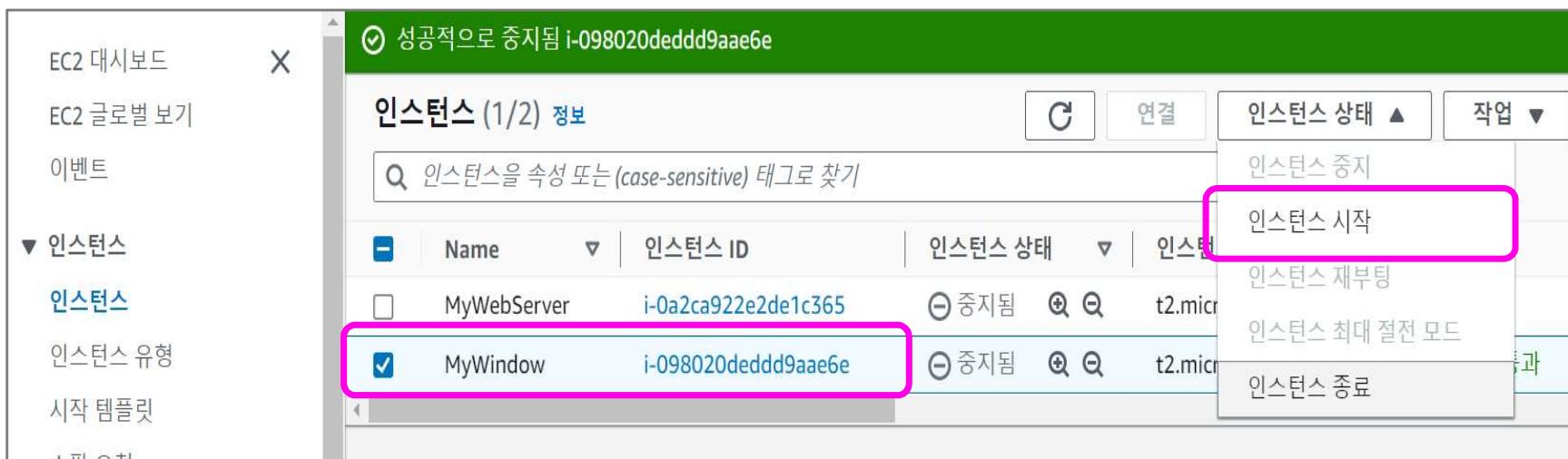
```
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
Last login: Wed Oct 25 11:37:28 2023 from 223.39.250.224  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.
```

```
ubuntu@ip-172-31-44-106:~$
```

# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

### ▪ Window EC2 실행



# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

### ▪ Window EC2 실행

Name	인스턴스 ID	인스턴스 상태
<input type="checkbox"/> MyWebServer	i-0a2ca922e2de1c365	중지됨
<input checked="" type="checkbox"/> MyWindow	i-098020deddd9aae6e	대기 중



Name	인스턴스 ID	인스턴스 상태
<input type="checkbox"/> MyWebServer	i-0a2ca922e2de1c365	중지됨
<input checked="" type="checkbox"/> MyWindow	i-098020deddd9aae6e	실행 중

# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

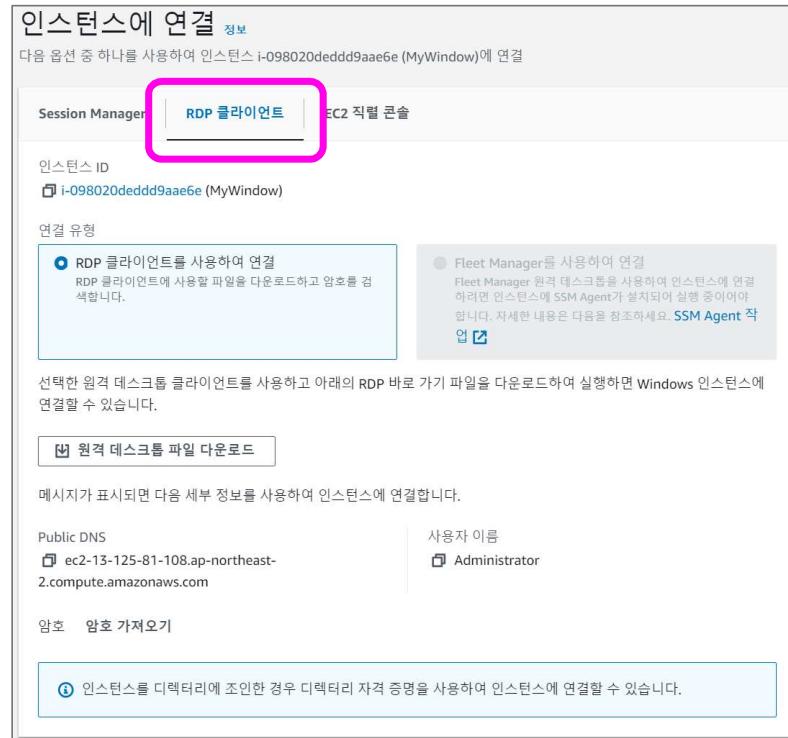
### ▪ Window EC2 실행

인스턴스 (1/2) 정보					
<input type="text"/> 인스턴스를 속성 또는 (case-sensitive) 태그로 찾기					
Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	경보 상태
<input type="checkbox"/> MyWebServer	i-0a2ca922e2de1c365	<input type="radio"/> 중지됨	<input type="radio"/> t2.micro	-	경보 없음
<input checked="" type="checkbox"/> MyWindow	i-098020deddd9aae6e	<input checked="" type="radio"/> 실행 중	<input checked="" type="radio"/> t2.micro	<input checked="" type="radio"/> 2/2개 검사 통과	경보 없음

# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

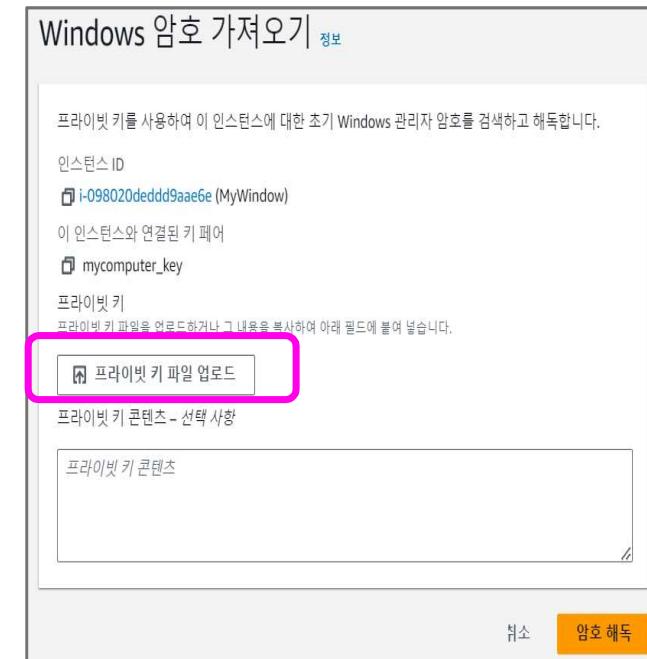
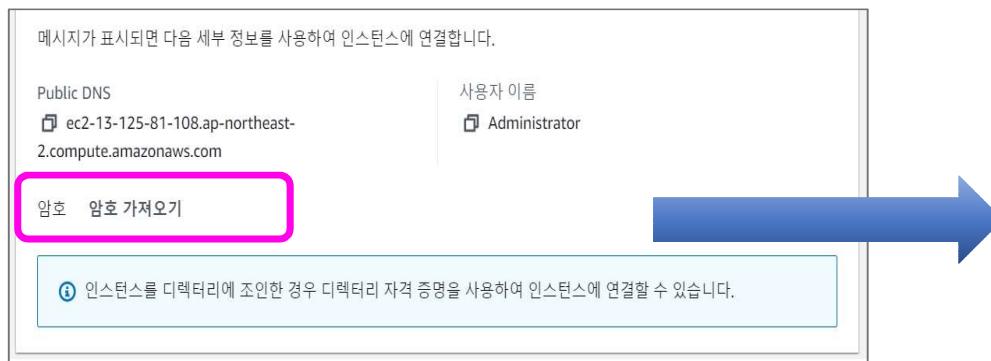
- **Window EC2 실행 → RDP 클라이언트**



# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

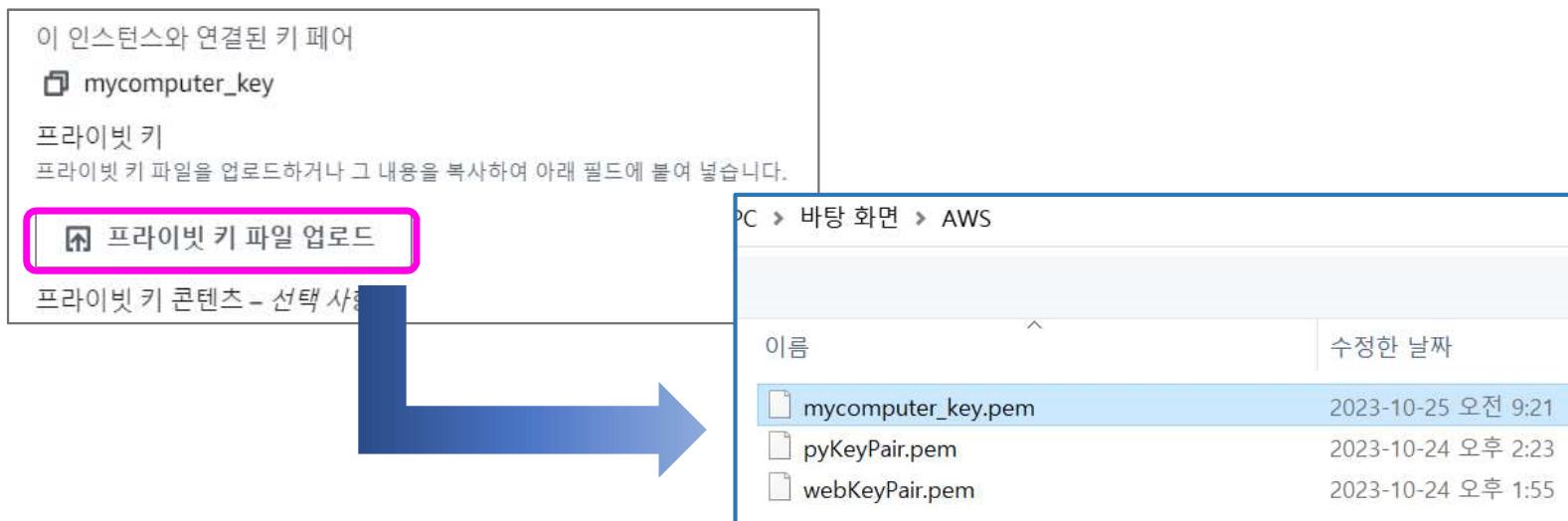
### ▪ Window EC2 실행 → RDP 클라이언트 설정



# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

- **Window EC2 실행 → RDP 클라이언트 설정**



# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

- Window EC2 실행 → RDP 클라이언트 설정

프라이빗 키 파일 업로드

mycomputer\_key.pem  
1.678KB

프라이빗 키 콘텐츠 - 선택 사항

```
-----BEGIN RSA PRIVATE KEY-----  
MIIEpaIBAAKCAQEAj7P9orR0zULOZRYPVu1J22kDQDqgS58eSwALj1zypR/Wqs8A  
cdnicGjf3PEyWXfgaBxnuZqFweQFB5ZYFT6/ZjKPilbmzRLeM3WqagBa99JOVzib  
Ekop25BsFM0JUCI9dhVaXgftjAQU4EMzI8ZB85fwII4+bX3Dp7g5J0iO18VoPv9  
hC1XJZa6zFR9bN4w0hmQkRFTeEclBrSDYfosKu/joNtbG3nOk7vqkC95BxYMNcP7  
EOSnZ+3ptqidBHPP14t9zsS9sgXMOaUz2kkZr03NdcN7ZJdgEpldd4yppfsae3Gy  
cNY5ixtqrCeilgUVOK7d2f1TDWhnlLnc01t4EQIDAQABaoBAEGTvrQIK6Eb49FZ  
fsiJYoHo6x3dYxb/VuCc9amHngcyf1DVQ/uwlX/8q+P0gailZvS7ua3eURnJB26N
```

Public DNS  
ec2-13-125-81-108.ap-northeast-2.compute.amazonaws.com

사용자 이름  
Administrator

암호  
CpGZrSVv(Zw(VktXFu1myaM=cmg1p7qR

취소 암호 해독

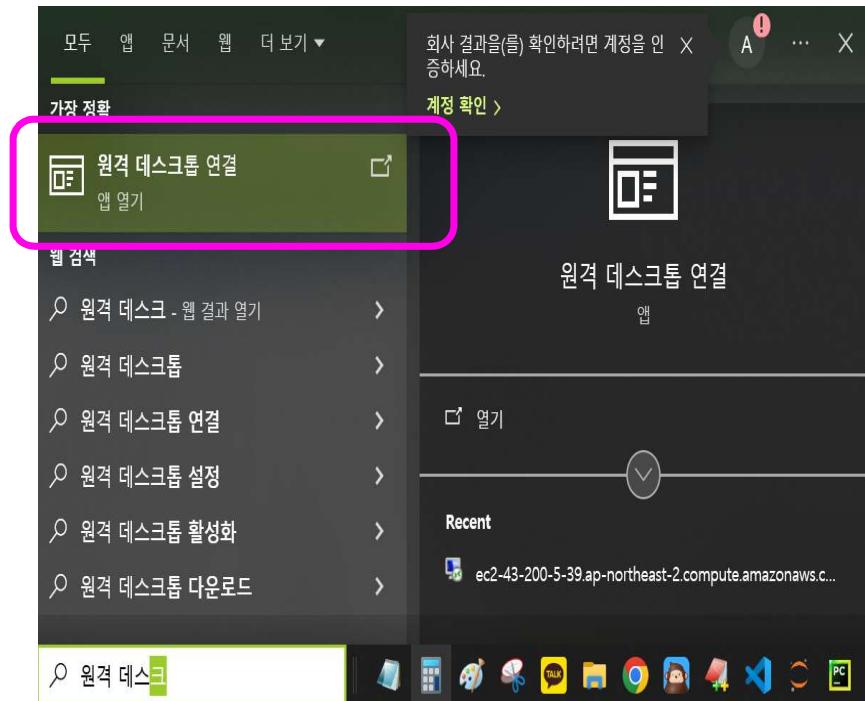


# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

### ▪ 클라이언트 Window PC

- '원격 데스크톱 연결' 입력

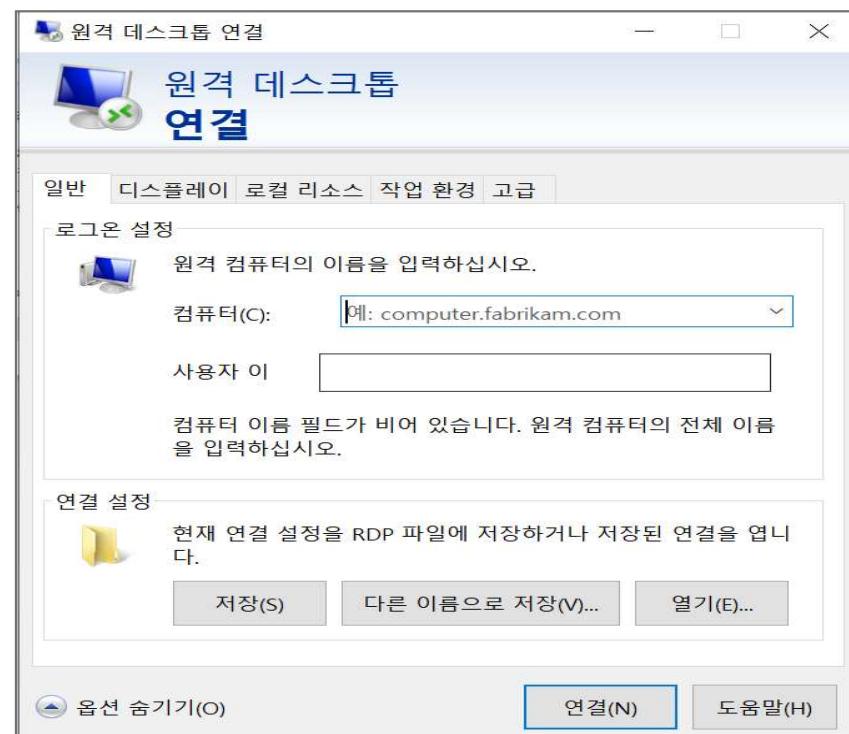


# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

### ▪ 클라이언트 Window PC

- '원격 데스크톱 연결' 실행



# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

### ▪ 클라이언트 Window PC

- 원격 컴퓨터 즉, AWS EC2 정보 입력

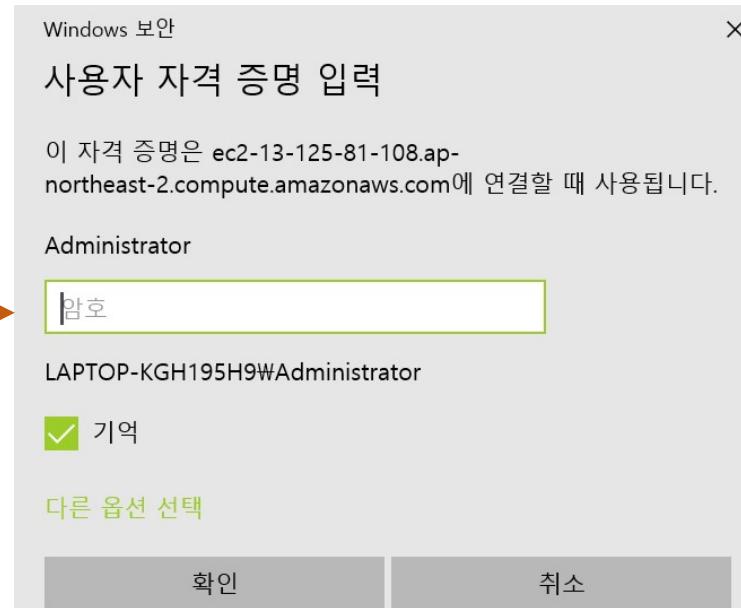


# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

### ▪ 클라이언트 Window PC

- 원격 컴퓨터 즉, AWS EC2 정보 입력

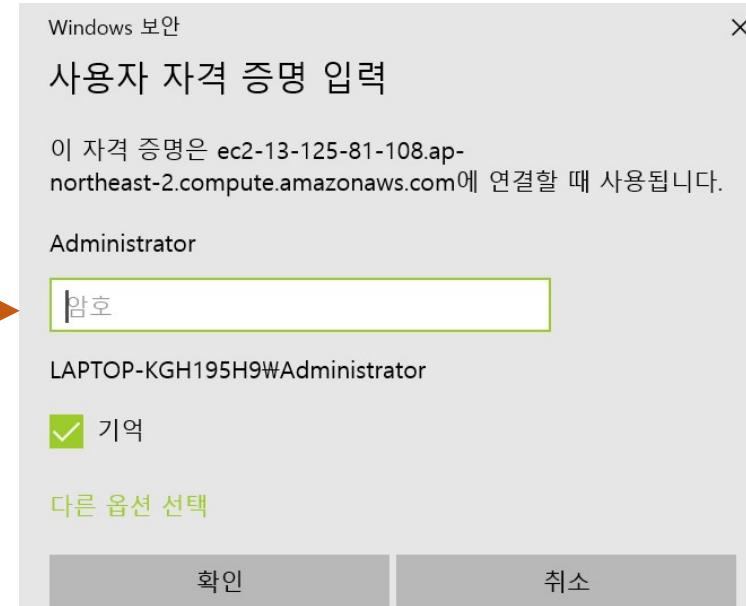


# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

### ▪ 클라이언트 Window PC

- 원격 컴퓨터 즉, AWS EC2 정보 입력

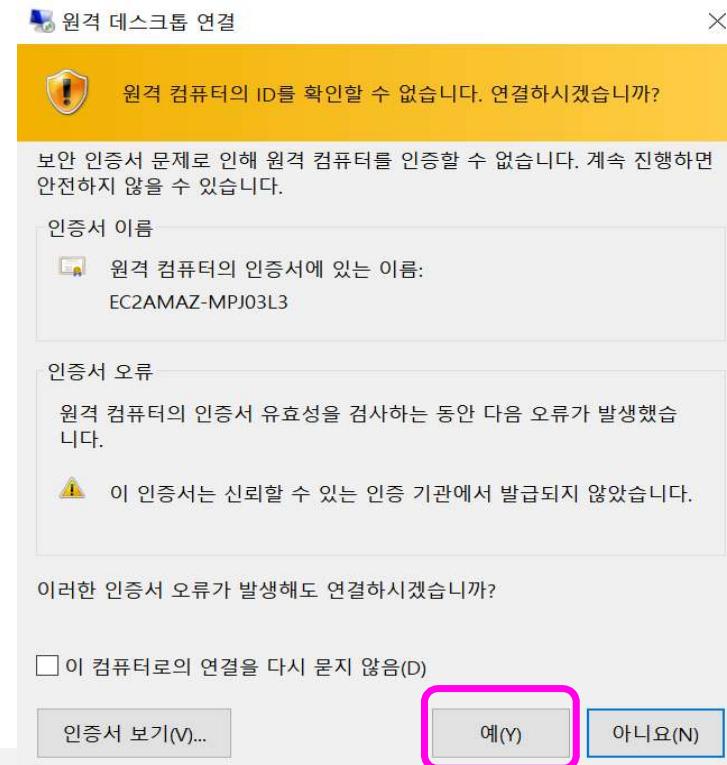


# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

### ▪ 클라이언트 Window PC

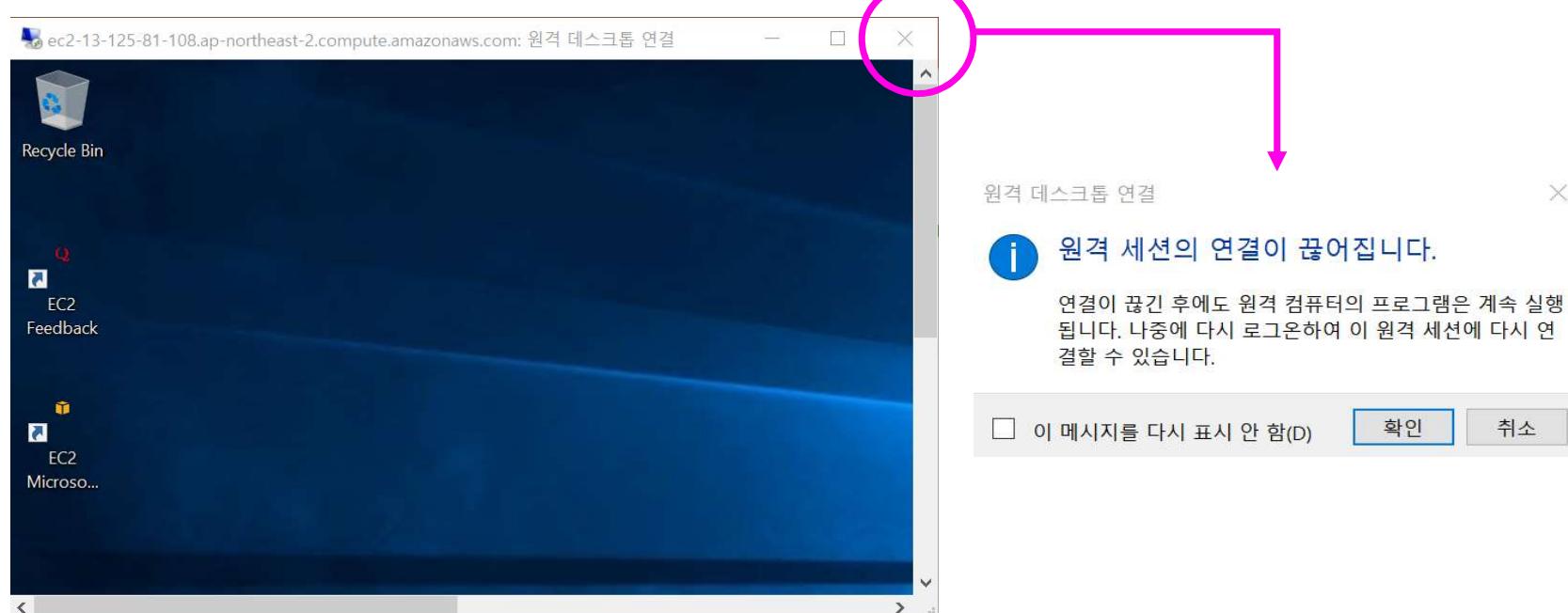
- 원격 컴퓨터 즉, AWS EC2 정보 입력



# EC2:Elastic Compute Cloud

## ◆ Window OS EC2 접속

- EC2 ← → 클라이언트 Window PC 연결



# PART II

# RDS

## Relational Database Service

# RDS:Relational Database Service

## ◆ RDS

- AWS에서 제공하는 관계형데이터베이스 서비스
- 데이터베이스 배포와 관리는 가장 복잡하고 많은 시간과 비용이 소요되는 IT 작업 처리
- 다양한 관계형데이터베이스시스템 지원

Microsoft SQL, Oracle, MySQL PostgreSQL, **Aurora**, MariaDB 등

# RDS:Relational Database Service

---

## ◆ RDS 비용 지불

- EC2인스턴스 위에서 돌아가는 서비스, [EC2인스턴스 시간당 요금 지불](#)
- [반드시 예약 인스턴스 구매](#)
- Multi-AZ 기능 손쉽게 사용 가능

# RDS:Relational Database Service

## ◆ RDS 데이터 시스템

### ▪ Data Warehousing

- 특히 비즈니스 인텔리전스에서 주로 사용
- **리포트 작성, 데이터 분석 시**에 사용 (Production Database -> Data Warehousing)
- **엄청난 양의 빅데이터를 불러올 때 사용되는 시스템**
- 서로 다양한 소스로부터 데이터가 합쳐진 데이터
- 도움이 될만한 데이터들을 저장하고 있다가 필요에 의해서 데이터를 추출

# RDS:Relational Database Service

## ◆ RDS 데이터 처리 프로세스

### ▪ 온라인 트랜잭션 처리(OLTP : OnLine Transaction Processing)

- 실시간 트랜잭션 프로세스
- 트랜잭션에서 발생되는 INSERT, UPDATE, DELETE의 과정을 무결성을 보장하여 처리하고 그 결과를 SELECT 하는 과정
- 무수히 많이 발생되는 각각의 작업요청을 오류없이 처리하고, 그 결과값을 실시간 처리하는 방법

# RDS:Relational Database Service

## ◆ RDS 데이터 처리 프로세스

### ▪ 온라인 분석 처리(OLAP:OnLine Analytical Processing)

- 집계된 데이터를 분석하는 것
- 데이터웨어하우스(DW), 쉽게 말해 DB에 저장되어 있는 데이터 분석하고, 데이터 분석을 통해 사용자에게 유의미한 정보를 제공해주는 처리방법
- 기존에 저장되어 있는 데이터를 사용자의 요구와 목적에 맞게 분석하여 정보 제공

# RDS:Relational Database Service

## ◆ RDS 데이터 처리 프로세스

구분	OLTP	OLAP
목적	비즈니스 활동 지원	비즈니스 활동에 대한 평가, 분석
주 트랜잭션 형태	SELECT, INSERT, UPDATE, DELETE	SELECT
속도	수초 이내	수초 이상 수분 이내
데이터 표현 시간	실시간	과거
관리단위	테이블	분석된 정보
최적화 방법	트랜잭션 효율화, 무결성의 극대화	조회 속도, 정보의 가치, 편의성
데이터 특성	트랜잭션 중심	정보 중심
예시	회원정보 수정, 상품수정, 댓글 작성 /수정	1년간 주요 인기 트랜드, 한달간 항목별 수입/지출

# RDS:Relational Database Service

## ◆ RDS 백업

### ▪ 스냅샷(SnapShot)

- 사진 촬영하듯이 특정 시간(시점)에 데이터 저장 장치(스토리지)의 파일 시스템을 포착해 별도의 파일이나 이미지로 저장, 보관하는 기술
- 과거 시점의 시스템 상태를 마치 사진을 찍어서 그 순간을 남긴다는 개념

# RDS:Relational Database Service

---

## ◆ RDS 백업

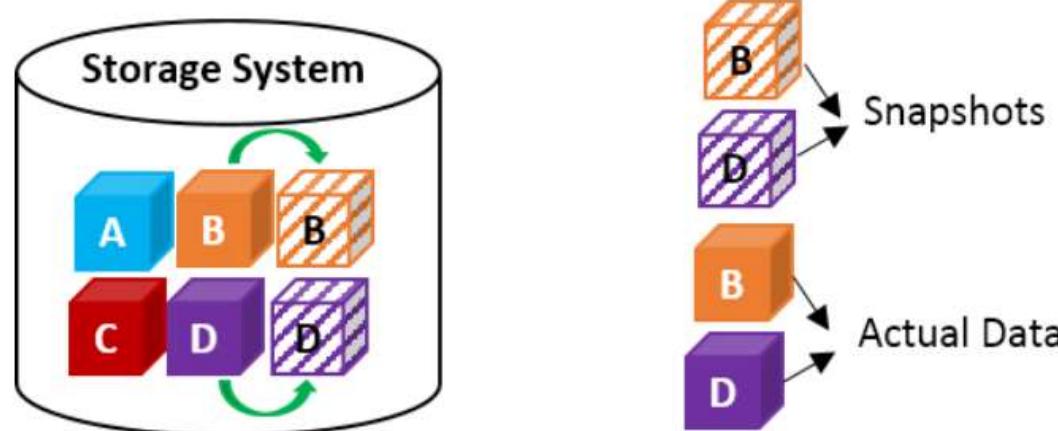
- 스냅샷(SnapShot) - **Point-in-time Copy 스냅샷**

- 특정 시점에서 데이터를 복사해두는 스냅샷 생성
- 원본 데이터(original data)를 별도로 준비된 스냅샷 공간(snapshot pool)에 복제
- 나중에 원본 데이터에 문제가 발생하게 되면 복원 작업 진행

# RDS:Relational Database Service

## ◆ RDS 백업

- 스냅샷(SnapShot) - Point-in-time Copy 스냅샷



# RDS:Relational Database Service

## ◆ RDS 백업

- 스냅샷(SnapShot) - Split-mirror(clone) 스냅샷

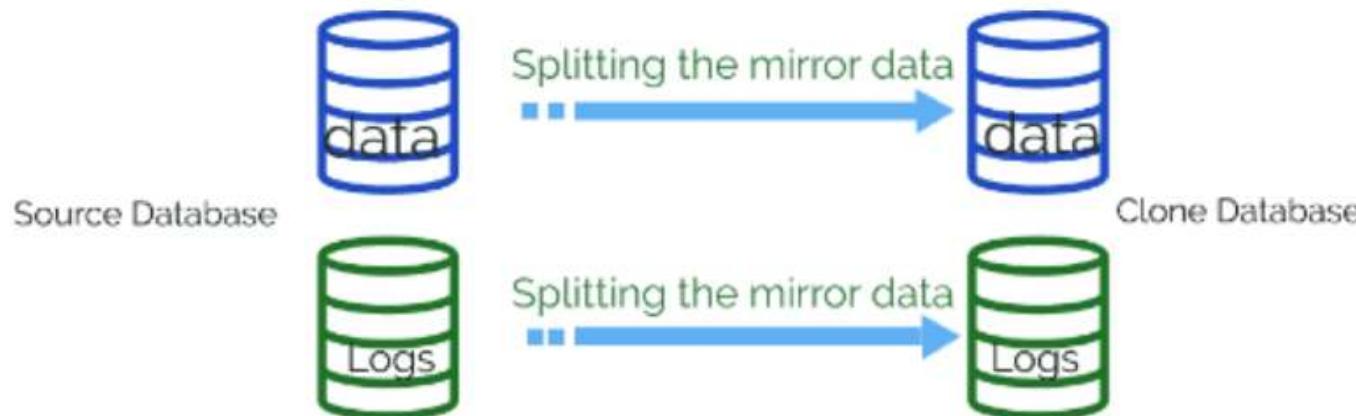
- 원본 데이터를 통째로 복제하는 방식으로, 가장 단순한 형태의 스냅샷
- 장점 : 저장 방식과 구조가 단순, 원본에 문제 발생 시 통째 복원 가능
- 단점 : 원본 데이터와 같은 양의 스토리지 공간 요구 → 공간 효율 떨어짐

데이터 양이 커질수록 스냅샷 생성 속도 느려짐 → 가용량과 성능 부담

# RDS:Relational Database Service

## ◆ RDS 백업

- 스냅샷(SnapShot) - Split-mirror(clone) 스냅샷



# RDS:Relational Database Service

## ◆ RDS 백업

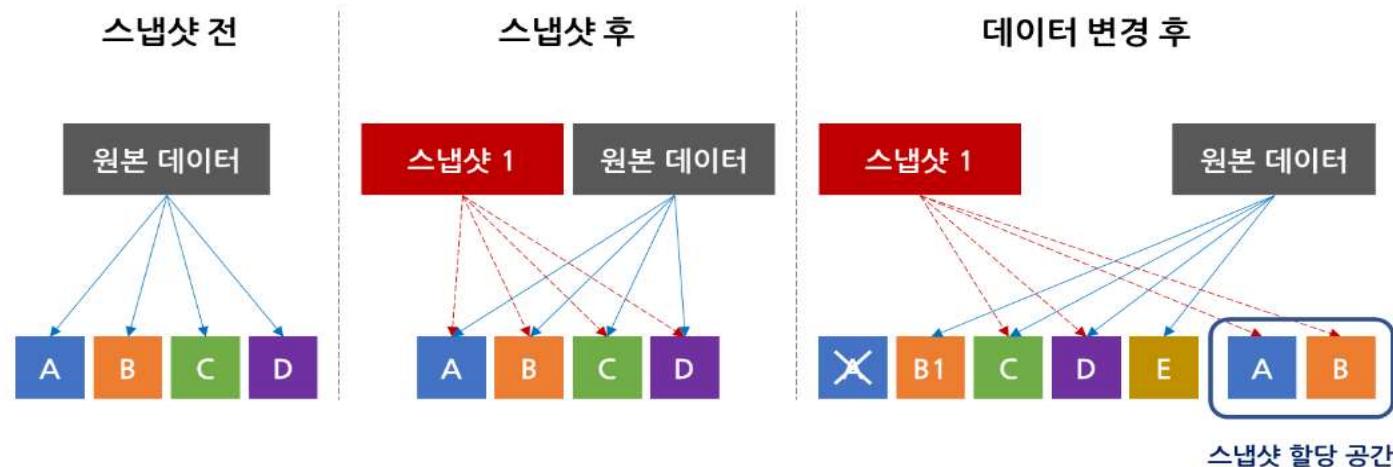
- 스냅샷(SnapShot) - **Copy-on-write(cow) 스냅샷**

- 가장 널리 쓰이는 스냅샷 방식 중 하나
- 쓰기 작업(write)이 있을 시 데이터를 복제(copy) → 데이터 변경되는 시점 스냅샷
- 미리 확보해 놓은 스냅샷 공간에 메타데이터만 만들어 놓기 때문에 거의 바로 생성
- 수정 경우, 원본 데이터 스냅샷 공간에 복제 후 수정된 데이터 기존 위치에 덮어씀

# RDS:Relational Database Service

## ◆ RDS 백업

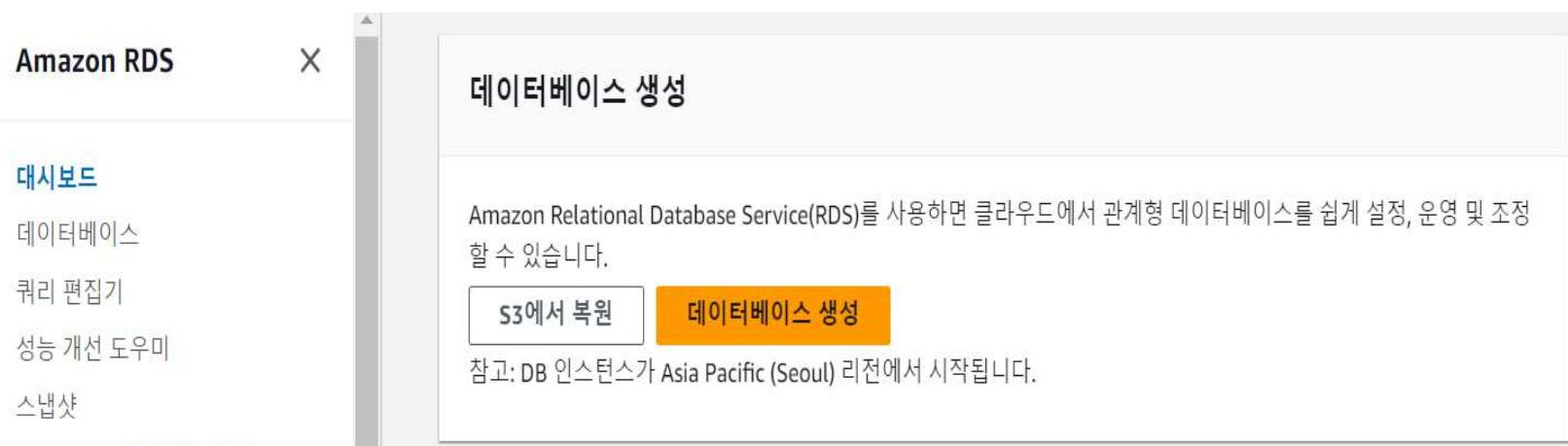
- 스냅샷(SnapShot) - Copy-on-write(cow) 스냅샷



# RDS:Relational Database Service

## ◆ RDS 백업

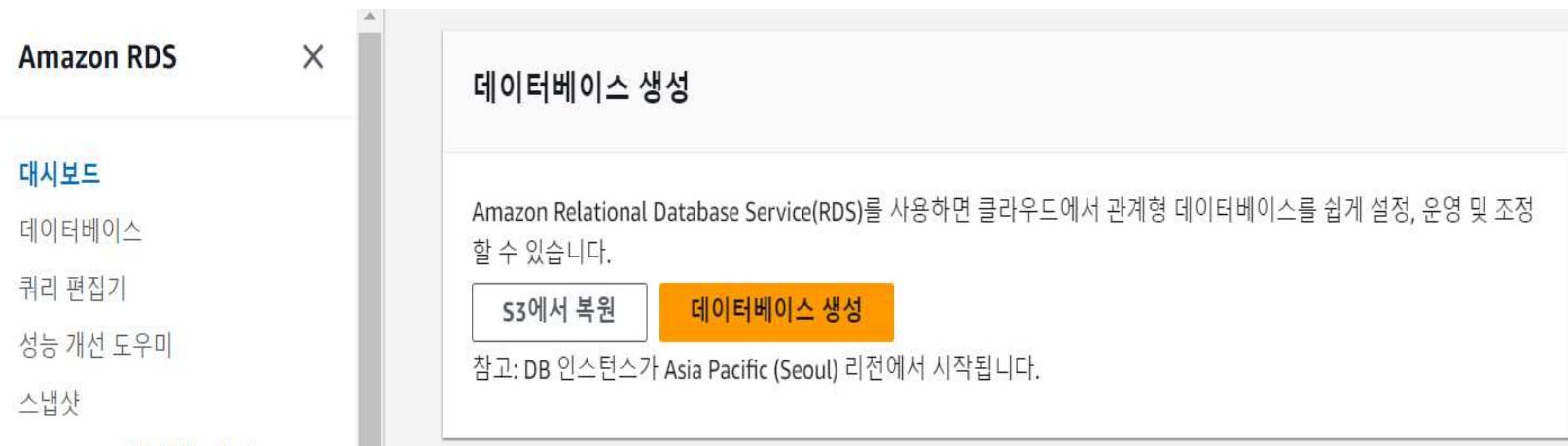
- 리전(Region)과 가용영역(AZ – Availability Zone)



# RDS:Relational Database Service

## ◆ RDS Instance 생성

### ▪ DB 인스턴스 생성



# RDS:Relational Database Service

## ◆ RDS Instance 생성

### ▪ DB 인스턴스 생성

#### 데이터베이스 생성

##### 데이터베이스 생성 방식 선택 정보

###### 표준 생성

가용성, 보안, 백업 및 유지 관리에 대한 옵션을 포함하여 모든 구성 옵션을 설정합니다.

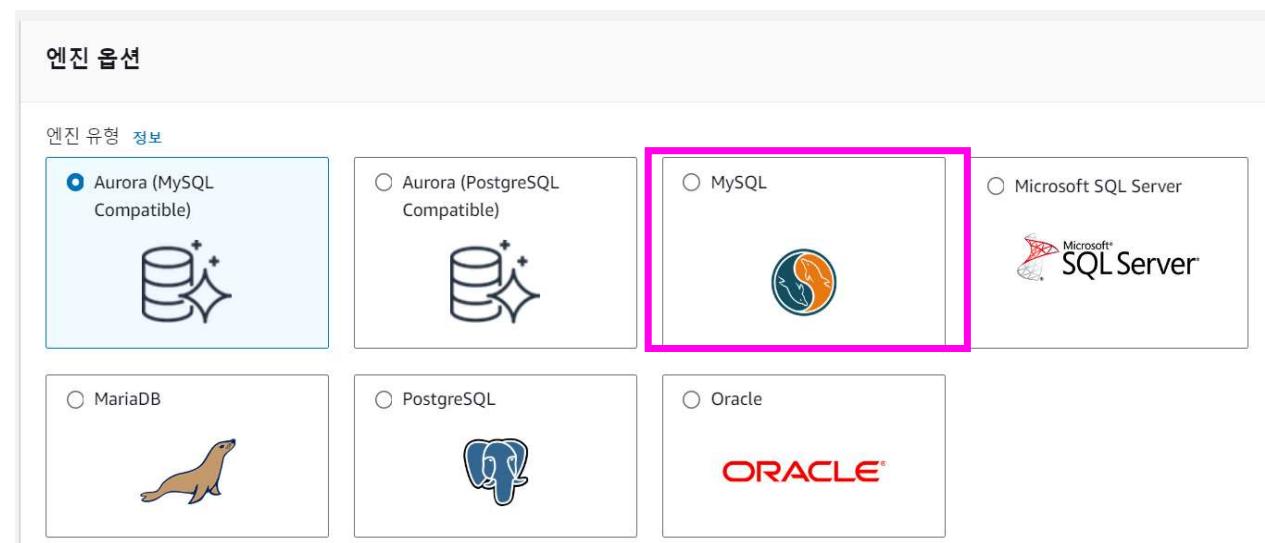
###### 손쉬운 생성

권장 모범 사례 구성을 사용합니다. 일부 구성 옵션은 데이터베이스를 생성한 후 변경할 수 있습니다.

# RDS:Relational Database Service

## ◆ RDS Instance 생성

### ▪ DB 인스턴스 생성



# RDS:Relational Database Service

## ◆ RDS Instance 생성

### ▪ DB 인스턴스 생성

에디션  
MySQL Community

**알려진 문제/제한 사항**  
알려진 문제/제한 사항 [\[?\] 정보](#)을 검토하여 특정 데이터베이스 버전과 발생할 수 있는 호환성 문제를 확인하세요.

▼ 필터 숨기기

다중 AZ DB 클러스터를 지원하는 버전 표시 [정보](#)  
기본 DB 인스턴스 1개와 읽기 가능한 대기 DB 인스턴스 2개로 다중 AZ DB 클러스터를 생성합니다. 다중 AZ DB 클러스터는 최대 2 배 빠른 트랜잭션 커밋 지연 시간과 일반적으로 35초 미만의 자동 장애 조치를 제공합니다.

Amazon RDS 최적화된 쓰기를 지원하는 버전 표시 [정보](#)  
Amazon RDS 최적화된 쓰기는 추가 비용 없이 쓰기 처리량(throughput)을 최대 2배 늘립니다.

엔진 버전  
MySQL 8.0.33

# RDS:Relational Database Service

## ◆ RDS Instance 생성

### ▪ DB 인스턴스 생성

**템플릿**  
해당 사용 사례를 충족하는 샘플 템플릿을 선택하세요.

- 프로덕션**  
고가용성 및 빠르고 일관된 성능  
을 위해 기본값을 사용하세요.
- 개발/테스트**  
이 인스턴스는 프로덕션 환경 외  
부에서 개발 용도로 마련되었습니다.
- 프리 티어**  
RDS 프리 티어를 사용하여 새로  
운 애플리케이션을 개발하거나,  
기존 애플리케이션을 테스트하  
거나 Amazon RDS에서 실무 경  
험을 쌓을 수 있습니다. [정보](#)

# RDS:Relational Database Service

## ◆ RDS Instance 생성

### ▪ DB 인스턴스 생성

▼ 자격 증명 설정

마스터 사용자 이름 [정보](#)  
DB 인스턴스의 마스터 사용자에 로그인 ID를 입력하세요.

1~16자의 영숫자. 첫 번째 문자는 글자여야 합니다.

AWS Secrets Manager에서 마스터 보안 인증 정보 관리  
Secrets Manager에서 마스터 사용자 보안 인증을 관리합니다. RDS는 사용자 대신 암호를 생성하고 수명 주기 동안 이를 관리할 수 있습니다.

i Secrets Manager에서 마스터 사용자 보안 인증 정보를 관리하는 경우 일부 RDS 기능은 지원되지 않습니다.  
[자세히 알아보기](#)

암호 자동 생성  
Amazon RDS에서 사용자를 대신하여 암호를 생성하거나 사용자가 직접 암호를 지정할 수 있습니다.

마스터 암호 [정보](#)  
\*\*\*\*\*  
제약 조건: 8자 이상의 인쇄 가능한 ASCII 문자. 다음은 포함할 수 없습니다. /(슬래시), '(작은따옴표)', "(큰따옴표)" 및 "@(엣 기호).

마스터 암호 확인 [정보](#)  
\*\*\*\*\*

# RDS:Relational Database Service

## ◆ RDS Instance 생성

### ▪ DB 인스턴스 생성

The screenshot shows a step in the Amazon RDS instance creation process. At the top, there is a header with an information icon and the text "Amazon RDS 최적화된 쓰기 - 신규 정보". Below this is a toggle switch labeled "Amazon RDS 최적화된 쓰기를 지원하는 인스턴스 클래스 표시".

Below the header, there is a section titled "DB 인스턴스 클래스 정보" with three options:

- 스탠다드 클래스(m 클래스 포함)
- 메모리 최적화 클래스(r 및 x 클래스 포함)
- 버스터블 클래스(t 클래스 포함)

A dropdown menu is open, showing the selected option: "db.t3.micro". Inside the dropdown, it says "2 vCPUs 1 GiB RAM 네트워크: 2,085Mbps".

At the bottom of the dropdown, there is another toggle switch labeled "이전 세대 클래스 포함".

# RDS:Relational Database Service

## ◆ RDS Instance 생성

### ▪ DB 인스턴스 생성

#### ▼ 스토리지 자동 조정

##### 스토리지 자동 조정 정보

애플리케이션의 필요에 따라 데이터베이스 스토리지의 동적 조정 지원을 제공합니다.

##### 스토리지 자동 조정 활성화

이 기능을 활성화하면 지정한 임계값 초과 후 스토리지를 늘릴 수 있습니다.

##### 최대 스토리지 임계값 정보

데이터베이스를 지정된 임계값으로 자동 조정하면 요금이 부과됩니다.

1000

GiB

최솟값은 22GiB이고, 최댓값은 6,144GiB입니다.

# RDS:Relational Database Service

## ◆ RDS Instance 생성

### ▪ DB 인스턴스 생성

연결 정보

C

컴퓨팅 리소스

이 데이터베이스의 컴퓨팅 리소스에 대한 연결을 설정할지를 선택합니다. 연결을 설정하면 컴퓨팅 리소스가 이 데이터베이스에 연결할 수 있도록 연결 설정이 자동으로 변경됩니다.

EC2 컴퓨팅 리소스에 연결 안 함  
이 데이터베이스의 컴퓨팅 리소스에 대한 연결을 설정하지 않습니다. 나중에 컴퓨팅 리소스에 대한 연결을 수동으로 설정할 수 있습니다.

EC2 컴퓨팅 리소스에 연결  
이 데이터베이스의 EC2 컴퓨팅 리소스에 대한 연결을 설정합니다.

네트워크 유형 정보

듀얼 스택 모드를 사용하려면 IPv6 CIDR 블록을 지정한 VPC의 서브넷과 연결해야 합니다.

IPv4  
리소스는 IPv4 주소 지정 프로토콜을 통해서만 통신할 수 있습니다.

듀얼 스택 모드  
리소스는 IPv4, IPv6 또는 둘 모두를 통해 통신할 수 있습니다.

Virtual Private Cloud(VPC) 정보

VPC를 선택합니다. VPC는 이 DB 인스턴스의 가상 네트워킹 환경을 정의합니다.

Default VPC (vpc-03723ef11a8327c6d)  
4 서브넷, 4 가용 영역

▼

해당 DB 서브넷 그룹이 있는 VPC만 나열됩니다.

# RDS:Relational Database Service

## ◆ RDS Instance 생성

### ▪ DB 인스턴스 생성

DB 서브넷 그룹 [정보](#)  
DB 서브넷 그룹을 선택합니다. DB 서브넷 그룹은 선택한 VPC에서 DB 인스턴스가 어떤 서브넷과 IP 범위를 사용할 수 있는지를 정의합니다.



퍼블릭 액세스 [정보](#)

예

RDS는 데이터베이스에 퍼블릭 IP 주소를 할당합니다. VPC 외부의 Amazon EC2 인스턴스 및 다른 리소스가 데이터베이스에 연결할 수 있습니다. VPC 내부의 리소스도 데이터베이스에 연결할 수 있습니다. 데이터베이스에 연결할 수 있는 리소스를 지정하는 VPC 보안 그룹을 하나 이상 선택합니다.

아니요

RDS는 퍼블릭 IP 주소를 데이터베이스에 할당하지 않습니다. VPC 내부의 Amazon EC2 인스턴스 및 다른 리소스만 데이터베이스에 연결할 수 있습니다. 데이터베이스에 연결할 수 있는 리소스를 지정하는 VPC 보안 그룹을 하나 이상 선택합니다.

VPC 보안 그룹(방화벽) [정보](#)

데이터베이스에 대한 액세스를 허용할 VPC 보안 그룹을 하나 이상 선택합니다. 보안 그룹 규칙이 적절한 수신 트래픽을 허용하는지 확인합니다.

기존 항목 선택

기존 VPC 보안 그룹 선택

새로 생성

새 VPC 보안 그룹 생성

# RDS:Relational Database Service

## ◆ RDS Instance 생성

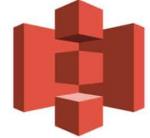
### ▪ DB 인스턴스 생성



**PART II**

**S3**  
**Simple Storage Service**

# S3:Simple Storage Service



## ◆ S3란

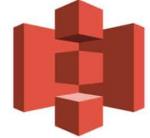
- AWS의 메인 저장소이며 오래된 서비스
- 매우 안전/가변적 저장 공간으로 파일 단위 저장 즉, 파일 서버 기능 서비스
- 구글 클라우드와 비슷
- 사진, 동영상 등 비정형 데이터 다룰 수 있으며 다운로드/업로드 가능
- 최소 요금 없음. 사용한 만큼만 비용 지불, S3 버킷 위치에 따라 결정
- 정적 리소스 저장용 / 정적 웹 페이지 및 콘텐츠 호스팅 / 정적 리소스 서버 활용
- 버킷(Bucket) 영역 생성하고 데이터를 키-값 형식 객체(Object)로 저장

# S3:Simple Storage Service

## ◆ S3 특징

- 저장할 수 있는 파일 수 제한 없음, 최소 1바이트에서 최대 5TB의 데이터 저장 가능
- 파일 인증 부여하여 무단 액세스 불가 가능
- HTTP와 BitTorrent 프로토콜, REST, SOAP 인터페이스 제공
- 중복 데이터 저장으로 데이터 손실 발생할 경우 자동 복원
- 버전관리 기능 통해서 사용자 실수 복원 가능
- 정보 중요도 따라 보호 수준 차등, 비용 절감 가능

# S3:Simple Storage Service



## ◆ S3 통신

### ▪ 데이터(파일 객체) 통신

- http와 https 둘 다 지원 → 가급적 https 액세스
- https 사용할 경우 URL 제약

예) my.bucket.s3.amazonaws.com 이라는 URL은 인증서 예러

<https://s3-us-west-2.amazonaws.com/my.bucket> 버킷 이름 구체적 지정

- CloudFront 서비스 사용해 커스텀 도메인 연결 가능

# S3:Simple Storage Service



## ◆ S3 비용

### ▪ 요금 정책

- 무료 용량 없음
- 객체/데이터/파일 업로드 시 GB당 비용 지불
- 저장 데이터 크기, 액세스 요청 횟수, 데이터 반출(네트워크 아웃)용량 등 계산
- 다른 AWS 서비스로 데이터 전송 시 비용 발생

# S3:Simple Storage Service

## ◆ S3 구성 요소

### ▪ 버킷(Bucket)

- 데이터를 저장하는 최상위 폴더
- 버킷 안에 데이터 종류별 폴더 생성 가능
- 고유한 이름 지정 필요
- 버킷에 포함된 모든 객체 대해 일괄적으로 인증, 접속 제한 설정 가능
- **버킷 생성이 S3 시작!!**

# S3:Simple Storage Service

## ◆ S3 구성 요소

- 객체(Object)

- 데이터 저장 단위로 파일
- 구성 : 키와 값
  - 키 - 파일명, test.txt
  - 값 - 내용, Good Luck!!!
- 버전 아이디 부여로 객체 복원/식별
- 메타데이터 즉, 데이터에 대한 설명 및 정보 담고 있는 데이터

# S3:Simple Storage Service

---

## ◆ S3 버킷 유형

### ▪ 일반 S3버킷

- 가장 보편적인 버킷 ➔ 기본값
- 데이터 손실 100% 가까이 막아줌
- 사용 : 클라우드, 모바일 게임 앱, 빅데이터 분석

# S3:Simple Storage Service

## ◆ S3 버킷 유형

- 드문 접근 (Infrequent Access Bucket: IA Bucket) S3 버킷
  - 객체에 자주 접근하지 않지만 객체가 많아 신속한 접근 요구될 때 적합
  - 일반 S3버킷에 비해 비용 저렴 → 객체 접근 시 추가 비용 발생
  - 다중 가용 영역(AZ) 사용으로 신속 접근 가능

# S3:Simple Storage Service

## ◆ S3 버킷 유형

- 단일 존 S3 버킷

- 하나의 가용 영역(AZ)에만 의존하는 버킷
- 일반 S3버킷, 드문접근버킷에 비해 **내구성, 가용성 떨어짐**
- 버킷 접근이 찾지 않는 객체 관리
- 값싼 비용 버킷

# S3:Simple Storage Service

## ◆ S3 버킷 유형

### ▪ 글레시어 S3 버킷

- 몇 년에 한번 객체 접근 하는 경우의 버킷
- 비용이 굉장히 저렴
- 객체 접근에 오랜 시간 소요 → 약 3 ~ 5시간
- 거의 사용하지 않는 객체 보관에 좋음

# S3:Simple Storage Service

## ◆ S3 버킷 유형

- 지능적 티어링 S3 버킷

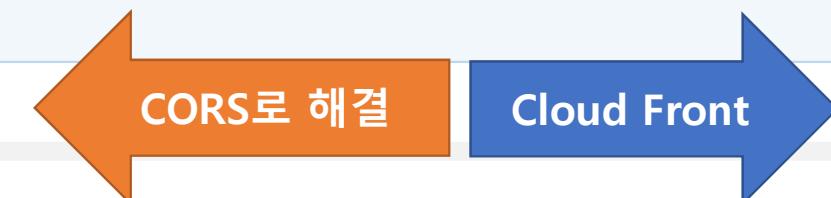
- 객체 접근 주기 분석하여 3가지 티어 제공 버킷
  - Frequent Tier → 접근 자주 발생하는 경우
  - Lower-Cost Tier → 접근이 한 달 이상 하지 않는 경우
  - Very Low-Cost Tier → 몇 년에 한번 접근 하는 경우

# S3:Simple Storage Service

## ◆ S3 리소스 공유

- 교차 출처 리소스 공유(CORS:Cross Origin Resource Sharing)

- 웹 브라우저 보안상 이유로 도입 (다른 도메인 데이터 공유 불가)
  - 현재 사용자가 접속한 웹 애플리케이션이 다른 출처의 리소스를 불러올 때,
  - **Access-Control-Allow-Origin** 헤더를 보내주지 않으면 브라우저가
  - 그 리소스를 거부하는 보안 정책



# S3:Simple Storage Service

## ◆ S3 버킷 제어 정책

- 기본 정책

- 버킷 생성자만 버킷에 대한 모든 권한 가짐 & 비공개

- 정책 변경

- 모든 객체에 대한 접근 구너한 설정 가능
- 객체, 폴더별 권한 설정 → 접근 제어 리스트 변경

# S3:Simple Storage Service

## ◆ S3 관련 용어

RSS

Reduced Redundancy Storage 약자

S3 객체에 비해서 데이터 손실 확률이 높은 형태 저장 방식

대신 가격 저렴 / 복원 가능한 데이터 저장 적합

물리적 HDD 대비 400배 가량 안전하다는 것이 아마존 주장

Glacier

빙하라는 뜻으로 매우 저렴한 가격으로 데이터 저장 가능

# S3:Simple Storage Service

## ◆ S3 생성

- 버킷 생성



# S3:Simple Storage Service

## ◆ S3 생성

### ▪ 버킷 생성

- **underscores( \_ )**  
doc\_example\_bucket
- **uppercase letters**  
DocExampleBucket
- **hyphen( - )**  
doc-example-bucket

일반 구성

버킷 이름  
 버킷 이름은 글로벌 네임스페이스 내에서 고유해야 하며 버킷 이름 지정 규칙을 따라야 합니다. [버킷 이름 지정 규칙 보기](#)

AWS 리전

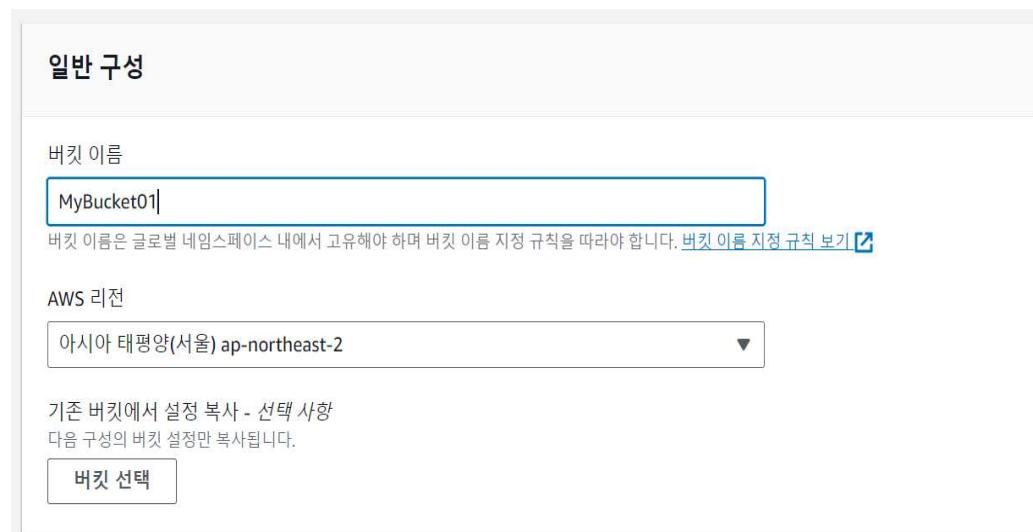
기존 버킷에서 설정 복사 - 선택 사항  
다음 구성의 버킷 설정만 복사됩니다.

# S3:Simple Storage Service

## ◆ S3 생성

### ▪ 버킷 생성

- **underscores( \_ )**  
doc\_example\_bucket
- **uppercase letters**  
DocExampleBucket
- **hyphen( - )**  
doc-example-bucket



PART III

**DEPLOY  
FLASK WEB SERVER**

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy)

- 개발환경구축

### EC2 Instance 환경에 WEB SERVER 구동 환경 구축

- 언어 : python
- 가상환경 SW : anaconda
- 패키지 리트스 : requirements.txt 파일

# DEPLOY FLASK WEB SERVER

---

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ① Local 컴퓨터의 개발환경 패키지 리스트 추출

```
pip list --format=freeze > requirement_pip_list.txt
```

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

### ▪ 개발환경구축 - ① Local 컴퓨터의 개발환경 패키지 리스트 추출

가상환경

```
(AI_WEB) C:\Users\anece\SO_PROJECT\00_WEBAI>pip list --format=freeze > requirement_aiweb.txt
```

```
(AI_WEB) C:\Users\anece\SO_PROJECT\00_WEBAI>dir  
C 드라이브의 볼륨에는 이름이 없습니다.  
볼륨 일련 번호: 2000-FDFF
```

C:\Users\anece\SO\_PROJECT\00\_WEBAI 디렉터리

```
2023-10-25 오후 02:54 <DIR> .  
2023-10-25 오후 02:54 <DIR> ..  
2023-10-19 오전 06:46 <DIR> migrations  
2023-10-18 오후 03:32 <DIR> myproject  
2023-10-25 오후 02:54 3,866 requirement_aiweb.txt  
2023-10-20 오후 02:26 <DIR> TotalApp  
2023-10-19 오전 05:04 <DIR> __pycache__  
1개 파일 3,866 바이트  
10개 디렉터리 42,049,024,000 바이트 남음
```

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ② 개발환경 패키지 리스트 Git에 업로드

The screenshot shows a GitHub repository interface. At the top, it says "Files" and "Private". On the right, there's a "Unwatch" button with a count of 1. Below that, there are buttons for "main", "1 branch", "0 tags", "Go to file", "Add file", and "Code". The main area lists three files: "anece00 aiweb\_package\_list ...", "README.md", and "requirement\_aiweb.txt". The first file was committed by "d0ac02d" 2 minutes ago with 2 commits. The "README.md" file was committed 7 minutes ago. The "requirement\_aiweb.txt" file was committed 2 minutes ago.

File	Commit	Time Ago
anece00 aiweb_package_list ...	d0ac02d	2 minutes ago
README.md		7 minutes ago
requirement_aiweb.txt		2 minutes ago

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

### (1) python 버전 확인

```
~$ python3 --version
```

### (2) pip 설치

```
~$ sudo apt install python3-pip
```

### (3) Git 설치

```
~$ sudo apt install git
```

# DEPLOY FLASK WEB SERVER

---

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

(4) MariaDB 관련 설치

(4-1) 라이브러리 설치

```
~$ sudo apt-get install libmariadb3 libmariadb-dev
```

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

### (4-2) DB 초기화 및 권한설정

```
~$ sudo mysql_secure_installation
```

```
In order to log into MariaDB to secure it, we'll need the current  
password for the root user. If you've just installed MariaDB, and  
you haven't set the root password yet, the password will be blank,  
so you should just press enter here.
```

```
Enter current password for root (enter for none):
```

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

### (4-2) DB 초기화 및 권한설정 ➔ root 비밀번호 설정 여부 ➔ 설정

```
OK, successfully used password, moving on...
Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.
Set root password? [Y/n]
```

```
New password: mariadb!
Re-enter new password: mariadb!
Password updated successfully!
Reloading privilege tables..
... Success!
```

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

### (4-2) DB 초기화 및 권한설정 ➔ root 원격 접속 여부

```
... Success!
```

```
Normally, root should only be allowed to connect from 'localhost'. This  
ensures that someone cannot guess at the root password from the network.
```

```
Disallow root login remotely? [Y/n]
```

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

### (4-2) DB 초기화 및 권한설정 ➔ test 데이터베이스 삭제 여부

```
By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.
```

```
Remove test database and access to it? [Y/n]
```

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

### (4-2) DB 초기화 및 권한설정 ➔ 지금까지 설정한 권한 정보 적용 여부

```
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n]
```

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

### (4-2) DB 초기화 및 권한설정 ➔ 설정 완료

```
... Success!
```

```
Cleaning up...
```

```
All done! If you've completed all of the above steps, your MariaDB  
installation should now be secure.
```

```
Thanks for using MariaDB!
```

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

### (5) anaconda 설치

#### (5-1) Linux용 anaconda 설치 파일 URL 추출

#### (5-2) Linux용 anaconda 설치 파일 다운로드

```
~$ wget https://repo.anaconda.com/archive/Anaconda3-  
2023.09-0-Linux-x86_64.sh
```



# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

### (5) anaconda 설치

#### (5-3) Linux용 anaconda 설치

~\$ bash [Anaconda3-2019.10-Linux-x86\\_64.sh](#)

```
installation finished.  
Do you wish to update your shell profile to automatically initialize conda?  
This will activate conda on startup and change the command prompt when activated.  
If you'd prefer that conda's base environment not be activated on startup,  
run the following command when conda is activated:
```

```
conda config --set auto_activate_base false
```

```
You can undo this by running `conda init --reverse $SHELL`? [yes|no]  
[no] >>> yes
```

```
--> For changes to take effect, close and re-open your current shell. <--
```

```
Thank you for installing Anaconda3!  
ubuntu@ip-172-31-32-222:~$
```

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

### (5) anaconda 설치

#### (5-4) Linux용 anaconda 관련 환경변수 설정

```
~$ nano ~/.bashrc # 파일 열기 → Ctrl+X 종료
```

```
else
    export PATH="/home/ubuntu/anaconda3/bin:$PATH"
fi
unset __conda_setup
# <<< conda initialize <<<
```

```
~$ source ~/.bashrc # 추가된 환경변수 적용
```

# DEPLOY FLASK WEB SERVER

---

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

(5) anaconda 설치

(5-5) conda 버전 확인

~\$ conda --version

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 아나콘다 & 가상환경 준비

### (6) anaconda 가상환경 생성

(6-1) 이름 : AIWEB, 파이썬 버전 : 3.9

```
~$ conda create -n AIWEB python=3.9
```

# DEPLOY FLASK WEB SERVER

## ◆ 배포(Deploy) 진행

- 개발환경구축 - ③ Linux OS EC2에서 패키지 설치

### (6) anaconda 가상환경 생성

#### (6-1) 가상환경 이동

```
~$ conda activate AIWEB
```

#### (6-2) 패키지 설치 리스트 준비

```
(AIWEB)~$ git clone
```