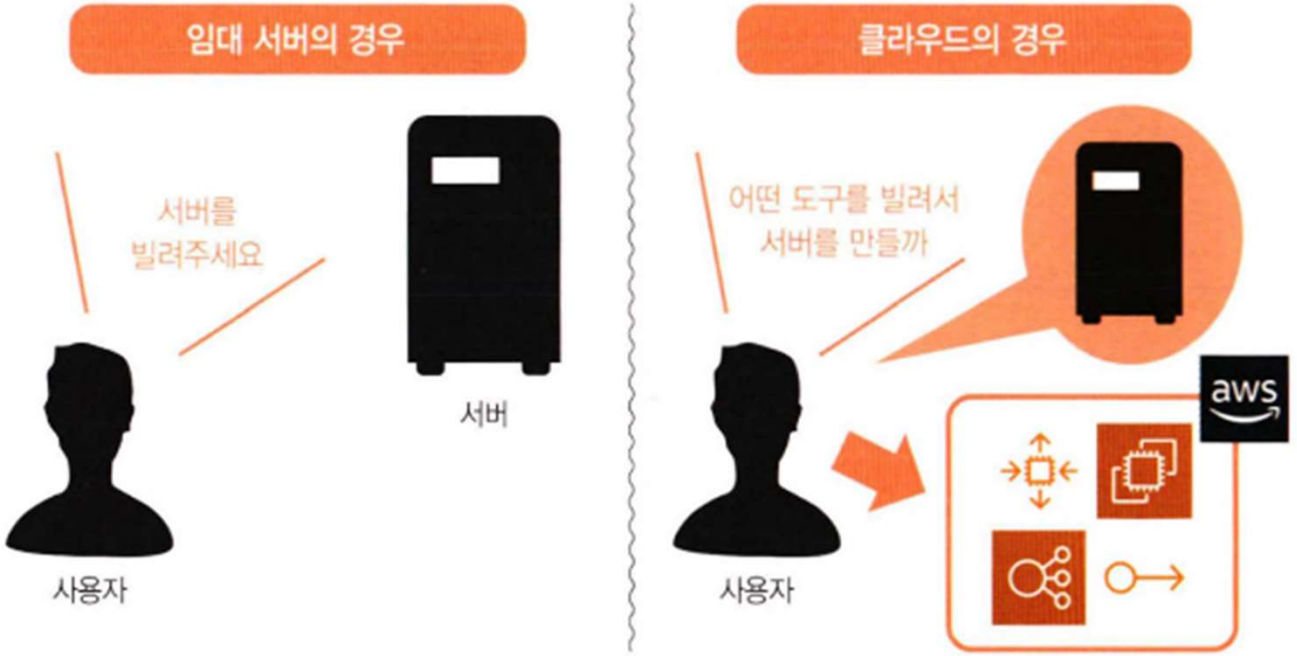
EC2

EC2

❖ Amazon EC2

✓ EC2

- Amazon Elastic Compute Cloud(Amazon EC2)는 컴퓨팅 용량을 제공하는 서비스
- EC2는 Managed 서비스가 아니므로 서버 및 네트워크 운영은 AWS가 담당하지만 운영 체제를 포함하여 필요한 소프트웨어는 사용자가 직접 설치하고 운영
- 임대 서버 와 Cloud 차이



EC2

- Amazon EC2
  - ✓ 클릭 한 번으로 최적의 서버 생성
    - 서버를 구축하려면 물리적인 기계를 준비하고 OS 및 소프트웨어를 설치하고 네트워크와 보안을 설정해야 하는데 이렇게 하려면 서버를 구축할 수 있는 지식을 가진 기술자가 필요
    - EC2는 서버를 직접 만든다고는 하지만 관리 콘솔에서 클릭 한 번으로 생성할 수 있기 때문에 서버에 대한 기술적 지식은 그다지 필요하지 않음
    - 다양한 서버 시스템의 조합(인스턴스 유형)과 OS 및 소프트웨어의 조합이 준비되어 있기 때문에 이를 선택하기만 하면 되고 물리적인 기계를 준비할 필요가 없어 초기 투자 비용도 줄일 수 있음
    - EC2는 Managed 서비스가 아니므로 AWS에 의해서 강제로 업데이트되지 않기 때문에 자유도가 높은 반면 관리하기는 번거로운 서비스
    - 구체적인 매뉴얼보다 어떻게 구성할까? 어느 정도의 성능이 필요할까? 와 같이 설계적인 관점이 필요

EC2

❖ Amazon EC2

✓ 클릭 한 번으로 최적의 서버 생성

● EC2에 서버를 생성하는 장점

누구든지 바로 사용할 수 있다

- 클릭 한 번으로 생성할 수 있다
  ➡ 관리 콘솔
- 준비된 것을 선택할 수 있다
  ➡ AMI, 인스턴스 유형
- 나중에 변경하기 쉬우므로 일단 시작할 수 있다
  ➡ 관리 콘솔, 인스턴스 유형

여러 가지를 선택할 수 있다

- CPU와 메모리 사양이 다양하게 준비되어 있다
  ➡ 인스턴스 유형
- OS나 소프트웨어의 종류가 다양하게 준비되어 있다
  ➡ AMI
- 연계하고 싶은 기능도 충실하게 되어 있다
  ➡ 다른 AWS 서비스

백업을 취득하기 쉽다 ➡ 가상화 기술
어디서든지 접속 ➡ 클라우드
물리적으로 다른 여러 장소에 설치할 수 있다 ➡ 리전과 가용 영역

EC2

- Amazon EC2
  - ✓ 생성 과 삭제가 편리
    - EC2는 관리 콘솔에 로그인하고 구성을 선택하는 것만으로도 서버를 생성할 수 있음
    - 누구든지 서버를 쉽게 생성할 수 있으므로 테스트용 서버를 만들고 싶을 때 나 서버 지식이 없는 사람이 서버를 만들고 싶을 때 도 유용
    - 간편하게 사용할 수 있으며 환경 구축에 필요한 소프트웨어를 설치하는 수고를 줄일 수 있으므로 시간도 단축
    - 바로 생성할 수 있고 바로 삭제할 수 있기 때문에 불필요한 리소스를 유지할 필요가 없음
    - 서버에 문제가 발생한 경우에도 즉시 복구할 수 있는데 복제 및 스케일 아웃, 스케일 다운 기능도 있고 이벤트 사이트 등 일시적으로 접속이 증가하는 경우에 적합
    - 바로 생성할 수 있고 바로 삭제할 수 있다 라는 것은 한마디로 불확실성이 많은 경우 에 매우 유용
    - 접속량이 얼마나 될지 모르고 서버 사양이 얼마나 필요할지 모르고 테스트를 생성하거나 일시적으로 사용할 경우 등 초기에 사양을 확정하기가 어려운 경우에 매우 많은 도움이 됨

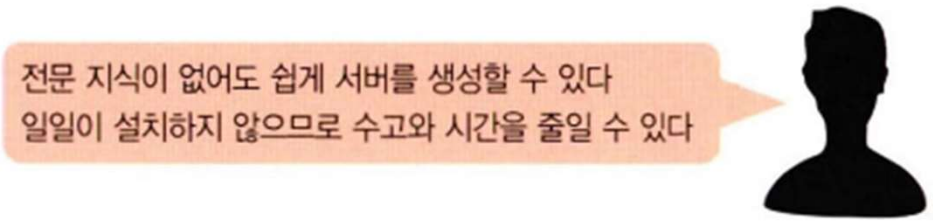
EC2

❖ Amazon EC2

✓ 생성 과 삭제가 편리

● 장점

고장난 경우	같은 서버를 복제하기 쉽기 때문에 바로 복구할 수 있다
부하가 증가한 경우	같은 구성의 서버를 복제하여 부하 분산하기 쉽다
접속이 줄었을 경우	서버 사양을 스케일 다운하여 비용을 줄이기 쉽다
일시적으로 사용할 경우	개발 테스트나 이벤트 사이트 등 일시적으로 사용할 경우에 간편하다



EC2

- Amazon EC2
  - ✓ 단점: 간단하고 선택의 폭이 넓은 점이 매력인 EC2이지만 적합하지 않는 경우도 있는데 단순한 서버 1대로 구성되어 그다지 변화가 없는 시스템의 경우라면 AWS의 장점을 살릴 수 없으므로(확장성. 다기능) EC2를 사용하는 이점이 없고 EC2는 스스로 관리해야 한다는 점을 전제로 하기 때문에 운영하는데 수고로울 수 있으므로 단순히 구축 만 하는 시스템에는 적합하지 않음

EC2

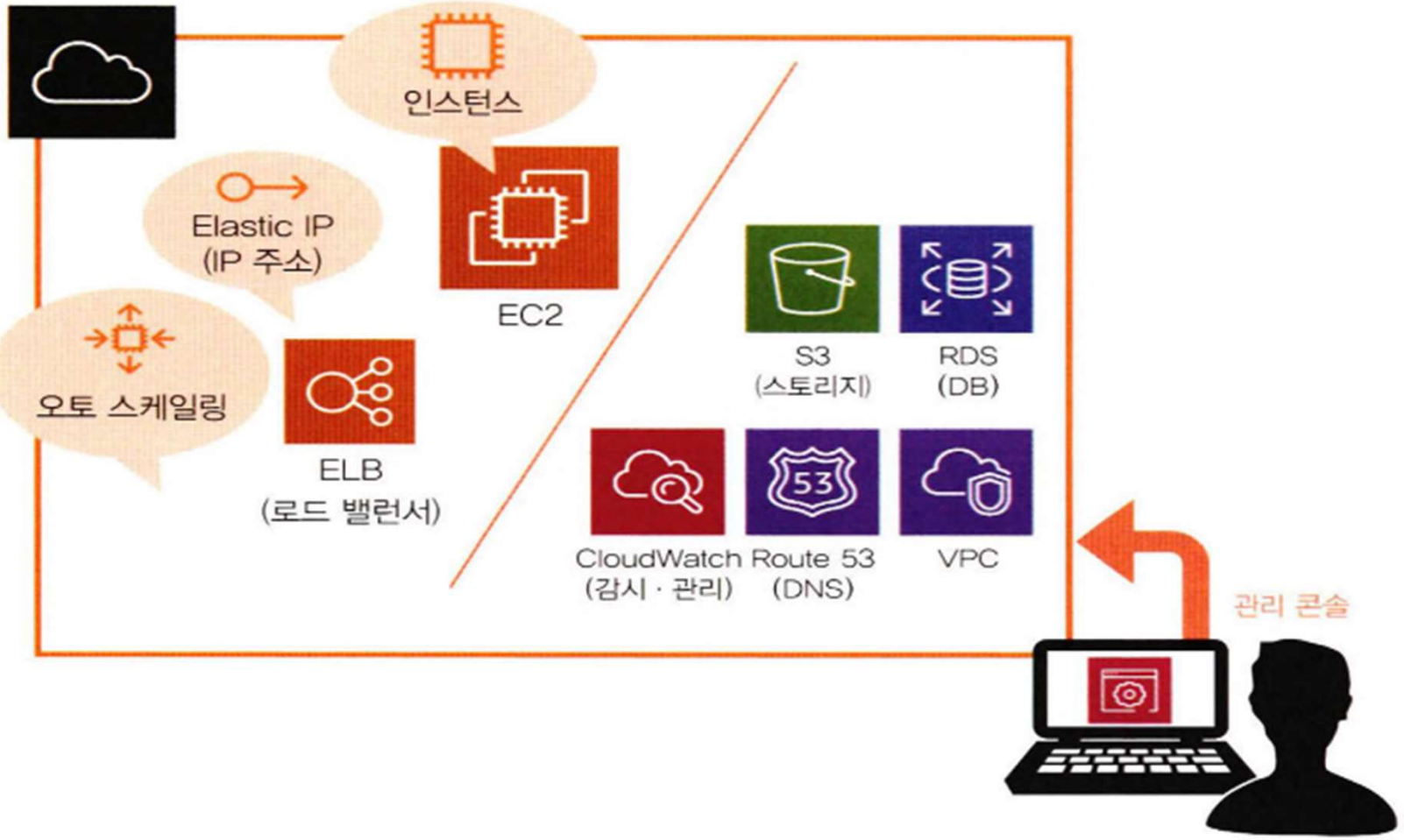
- EC2 서비스의 주요 구성
  - ✓ 인스턴스: AWS 클라우드에 생성한 가상 서버를 의미하는데 AMI로 몇 번이고 같은 구성의 서버를 생성할 수 있기 때문에 생성한 서버를 이와 같이 부름
  - ✓ AMI: 가상 이미지를 의미하는데 인스턴스를 생성하는 기준이 되는 금형과 같은 것으로 OS 만 설치된 간단한 유형부터 소프트웨어도 설정된 유형까지 다양한 AMI가 있음
  - ✓ Key Pair: 인스턴스에 접속할 때 인증을 위해 사용하는 키
  - ✓ EBS: AWS 클라우드에서 사용할 수 있는 스토리지로 인스턴스의 스토리지로 사용
  - ✓ 보안 그룹: 가상 방화벽으로 1개 이상의 인스턴스 트래픽을 제한
  - ✓ Elastic IP: 정적(고정) IPv4 주소

EC2

- EC2 인스턴스 요금
  - ✓ 인스턴스의 이용 요금은 다음 4가지 항목의 비용을 합한 것: 인스턴스 사용량 + EBS 요금 + 통신 요금 + 그 외 옵션
    - 인스턴스 사용량(가동 시간 X 단가): 인스턴스가 가동한 시간(초) 단위로 과금되며 정지하고 있는 동안은 과금되지 않는데 단위는 인스턴스 유형에 따라 다르고 고기능을 사용할 경우 비싸짐
    - EBS(스토리지) 요금(용량 X 단가): 인스턴스가 사용하는 EBS 요금으로 보유한 용량 단위로 과금되며 스토리지 성능(SSD인지, HDD인지, IOPS 담보를 할지 여부 등)에 따라 단가가 달라지는데 보유한 용량 단위이며 저장 용량 단위가 아니므로 주의해야 하며 인스턴스와는 달리 정지하고 있는 동안에도 요금이 부과
    - 통신 요금(아웃바운드 통신 요금): 인스턴스의 통신 요금으로 인터넷에서 인스턴스로 들어오는 인바운드 통신료는 무료이며 인스턴스에서 인터넷으로 나가는 통신 아웃바운드만 요금이 부과되며 요금은 리전에 따라서 다름
    - 그 외 옵션: Elastic IP 서비스 등 옵션을 사용할 경우 해당 요금이 추가

EC2

❖ EC2 기능 과 다른 서비스의 연계



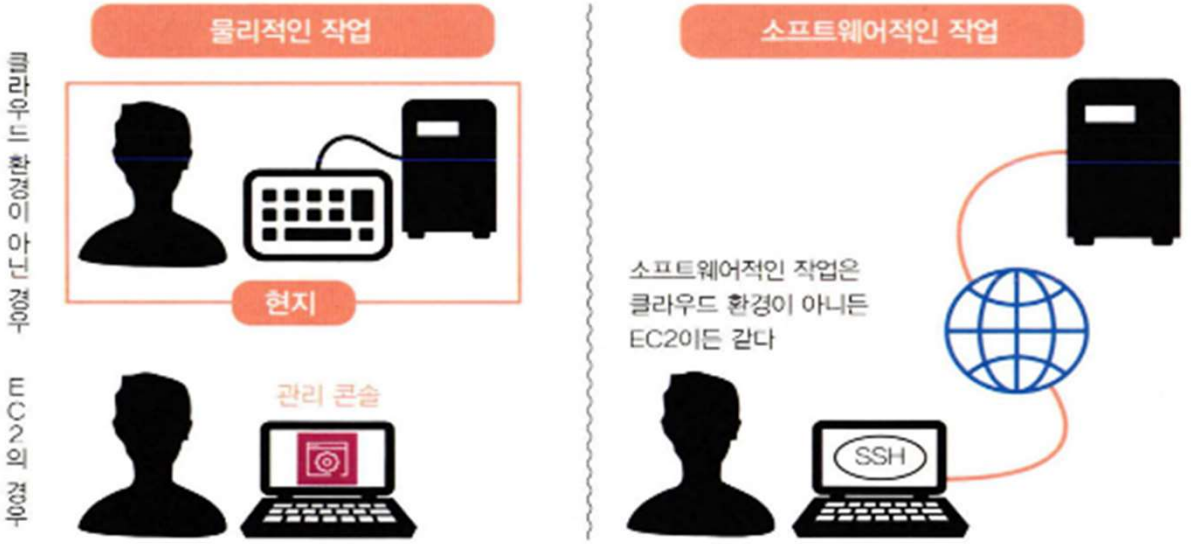
EC2

- EC2 기능 과 다른 서비스의 연계
  - ✓ EC2 와 많이 조합하는 AWS 서비스
    - Amazon S3: 인터넷용 스토리지 서비스
    - Amazon RDS: 관계형 데이터베이스 서비스
    - Amazon CloudWatch: 모니터링 및 관리 서비스
    - Amazon Route 53: DNS 서비스
    - Amazon VPC: 가상 사설 클라우드

EC2 인스턴스 생성

❖ EC2 운영

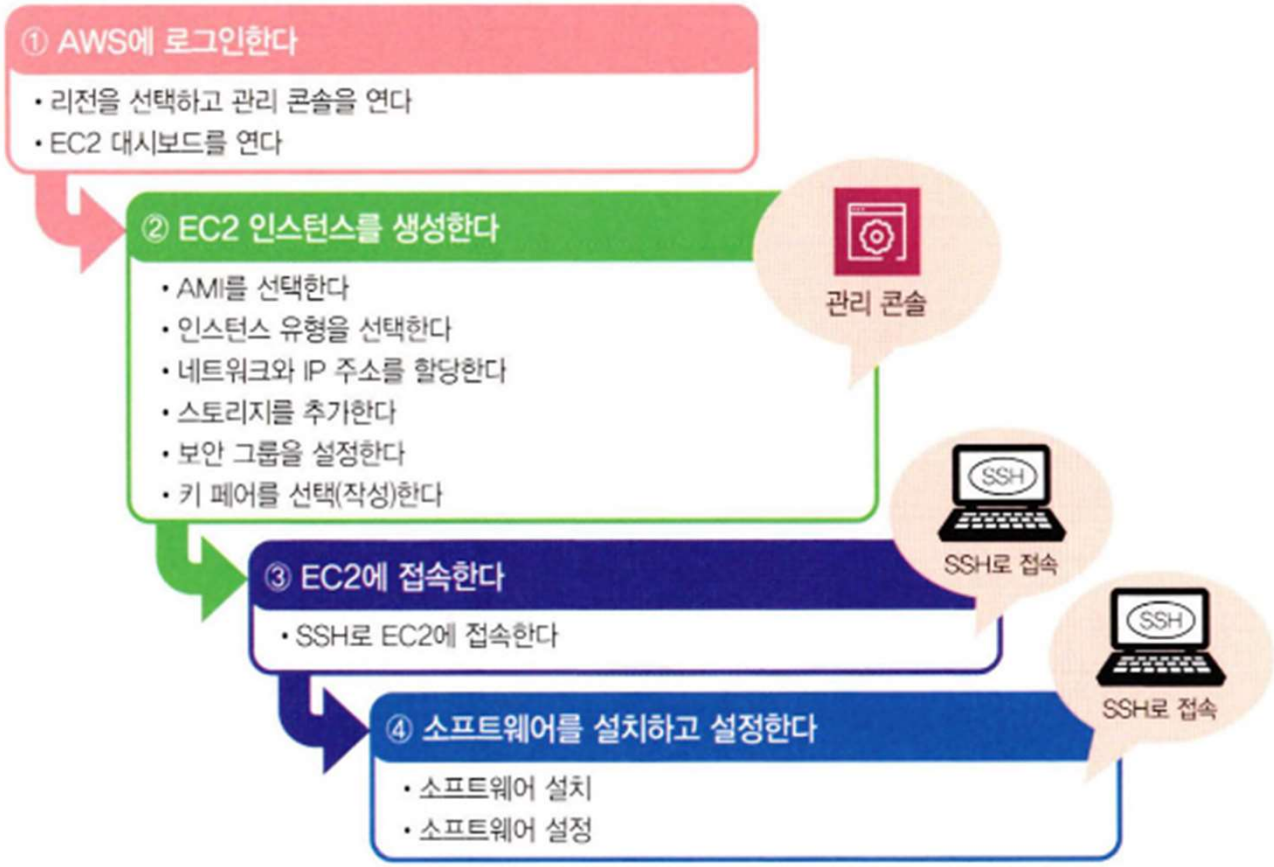
- ✓ 서버 관리자의 업무는 서버를 설치하거나 서버의 하드웨어 및 주변 환경을 구축하는 물리적인 작업 그리고 서버 OS에 로그인하여 작업하는 소프트웨어적 인 작업이 있음
- ✓ 물리적인 작업은 클라우드 환경이 아니라면 현지에 가서 직접 케이블을 연결하여 작업하는 경우가 많지만 EC2는 관리 콘솔에서 작업할 수 있음
- ✓ 소프트웨어적인 작업은 클라우드가 아닌 환경이든 EC2이든 관계없이 SSH를 이용해 원격으로 로그인하여 작업
- ✓ SSH는 서버를 원격에서 작업하기 위한 프로토콜로 EC2도 SSH로 접속하도록 설정하는 것은 동일
- ✓ 서버를 구성할 때 물리적인 작업 과 소프트웨어적인 작업



EC2 인스턴스 생성

❖ EC2 사용 절차

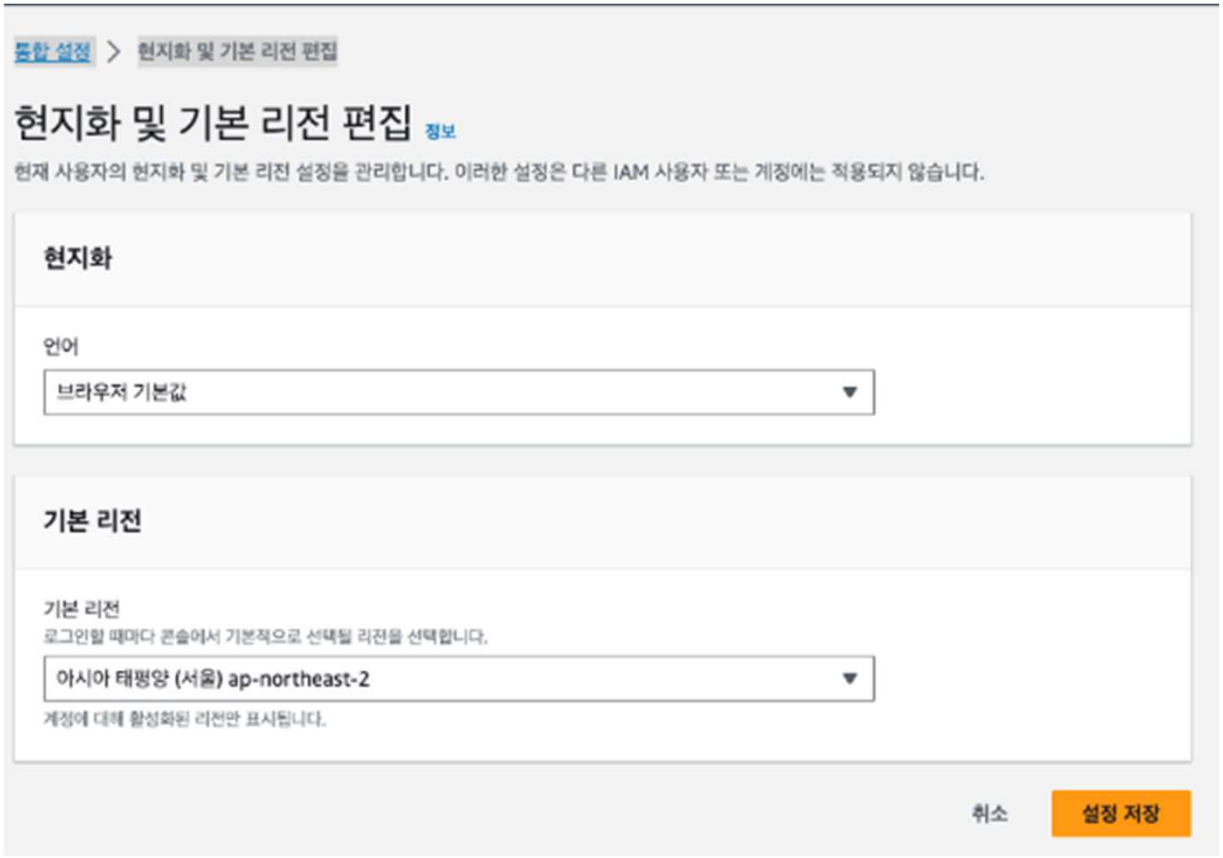
- ✓ EC2를 사용하려면 관리 콘솔에서 EC2 대시보드를 열고 인스턴스를 생성
- ✓ 인스턴스가 생성되면 SSH에 접속
- ✓ 인스턴스를 만들려면 컴퓨터 사양과 OS의 종류, 소프트웨어 뿐 만 아니라 네트워크와 IP 주소, 보안도 설정해야 하는데 이런 작업은 일반적인 서버와 동일



EC2 인스턴스 생성

❖ EC2 사용 절차

✓ 기본 리전 설정 및 편집: 통합 설정 > 현지화 및 기본 리전 편집



EC2 인스턴스 생성

❖ EC2 사용 절차

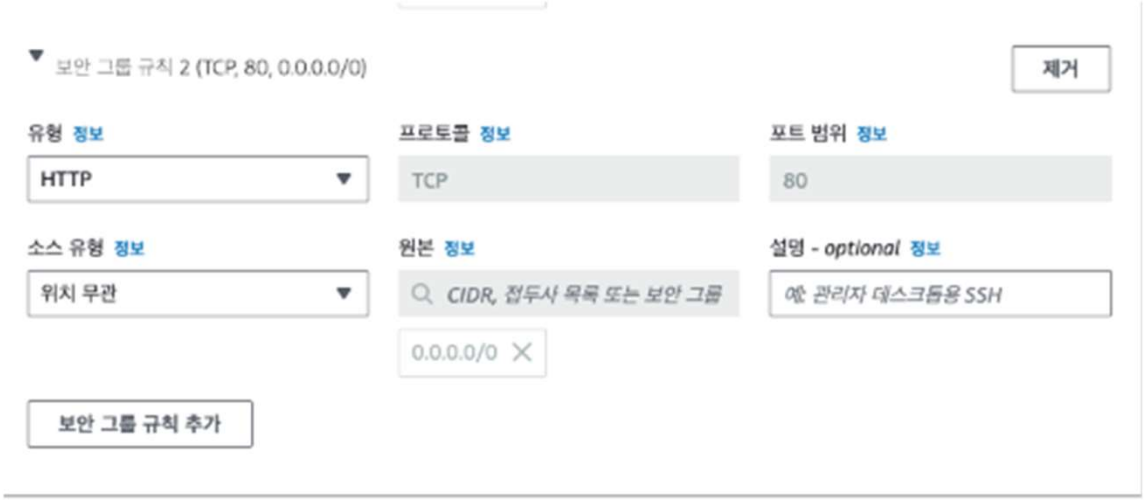
✓ EC2 인스턴스 생성

EC2 인스턴스 생성

- 인스턴스 설정 항목
  - ✓ 일반적인 서버를 생성할 때 검토해야 할 내용은 인스턴스 생성 때도 마찬가지로 정해야 하는데 CPU는 XX, 메모리는 OO, OS는 센트OS 와 같이 일반적으로는 하나씩 검토하지만 AWS의 경우는 다양한 인스턴스 유형과 여러 AMI를 제공하고 있음
  - ✓ 인스턴스 설정 항목
    - 이름: 인스턴스를 구별하기 위한 고유 이름
    - 태그(EC2 인스턴스의 레이블): 인스턴스에 임의의 태그를 부여할 수 있는데 Name 태그를 사용하여 인스턴스 이름에 부여할 수 있으므로 사용하면 편리
    - AMI: EC2 인스턴스의 가상 이미지이며 소프트웨어 구성을 작성한 템플릿으로 사용할 OS, 소프트웨어 등 어떤 인스턴스를 만들 것인지 정하고 이에 해당하는 AMI를 선택
    - 인스턴스 유형: EC2 인스턴스의 장비 사양으로 CPU, 메모리, 장비 유형을 결정
    - 키 페어
      - 키 페어를 가지고 EC2 인스턴스에 접속할 수 있음
      - 한 번 다운로드한 후에는 두 번 다시 다운로드할 수 없기 때문에 반드시 안전한 곳에 보관 해두는 것을 권장
      - 기존에 이미 키 페어를 생성했다면 드롭다운 메뉴에서 존재하는 키 페어를 사용할 수 있음
      - 페어 유형은 RSA를 선택하고 보통은 pem으로 선택하고 Putty라는 프로그램을 통해 인스턴스에 접속해야 하는 경우 ppk 파일을 다운

EC2 인스턴스 생성

- ❖ 인스턴스 설정 항목
  - ✓ 인스턴스 설정 항목
    - ● 네트워크
      - ◆ EC2 인스턴스를 배치할 네트워크로 AWS의 VPC(AWS 계정 전용 가상 네트워크)에서 선택하는데 VPC가 없으면 새로 생성하거나 기본 VPC를 사용
      - ◆ 하나의 VPC 안에서 인스턴스는 공유되는 컴퓨팅 파워를 사용
      - ◆ 서브넷 역시 디폴트 값을 사용
      - ◆ 퍼블릭 IP 자동 할당은 활성화되어 있으며 이를 통해 인스턴스의 퍼블릭 아이피 주소를 획득
      - ◆ 네트워크 설정 옆의 편집 버튼을 눌러서 나오는 화면에서 [보안 그룹 규칙 추가]를 누르고 HTTP를 추가



EC2 인스턴스 생성

- ❖ 인스턴스 설정 항목
  - ✓ 인스턴스 설정 항목
    - 스토리지의 용량 과 종류: 서버 장비의 스토리지로 OS가 설치된 장소에 있는데 기본적으로 EBS를 선택하고 사용할 디스크 용량과 스토리지의 종류를 선택하는데 용도에 따라서는 인스턴스 스토어(전원이 끊기면 데이터가 사라지지만 읽기 성능이 빨라지는 것으로 일부 인스턴스 유형에 대응}를 선택해도 되는데 S3와 같은 외부 스토리지는 선택할 수 없음

EC2 인스턴스 생성

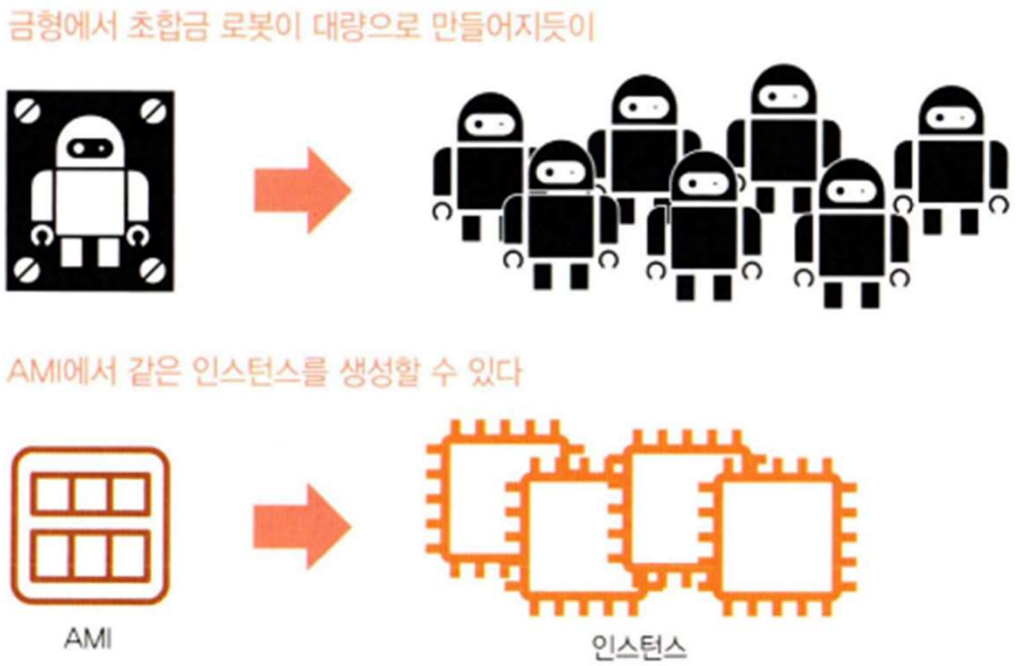
❖ Amazon EC2 인스턴스 생성

✓ 인스턴스 유형 및 OS를 선택

- EC2는 선택의 폭이 넓어 매력적
- EC2로 서버를 생성할 때는 인스턴스 유형을 선택하여 구성할 수 있는데 어떤 OS를 사용할 것인가 소프트웨어를 설치할 것인가 어떤 AMI를 선택할 것 인지에 따라 결정
- 인스턴스 유형과 AMI 모두 다양하게 준비되어 있어 자신만의 소프트웨어를 설치 및 설정할 수도 있고 미리 구성된 상태의 인스턴스를 사용할 수도 있음
- 서버를 생성할 때 선택의 폭이 넓음

EC2 인스턴스 생성

- ❖ Amazon EC2 인스턴스 생성
  - ✓ AMI
    - ● AMI 와 인스턴스
      - ◆ AMI(Amazon Machine Image)란 소프트웨어 구성을 기록한 템플릿으로 인스턴스(가상 서버)를 생성하기 위한 금형과 같은 것으로 금형을 한 번 만들어 두면 얼마든지 같은 설정의 서버를 생성하는 것이 가능



EC2 인스턴스 생성

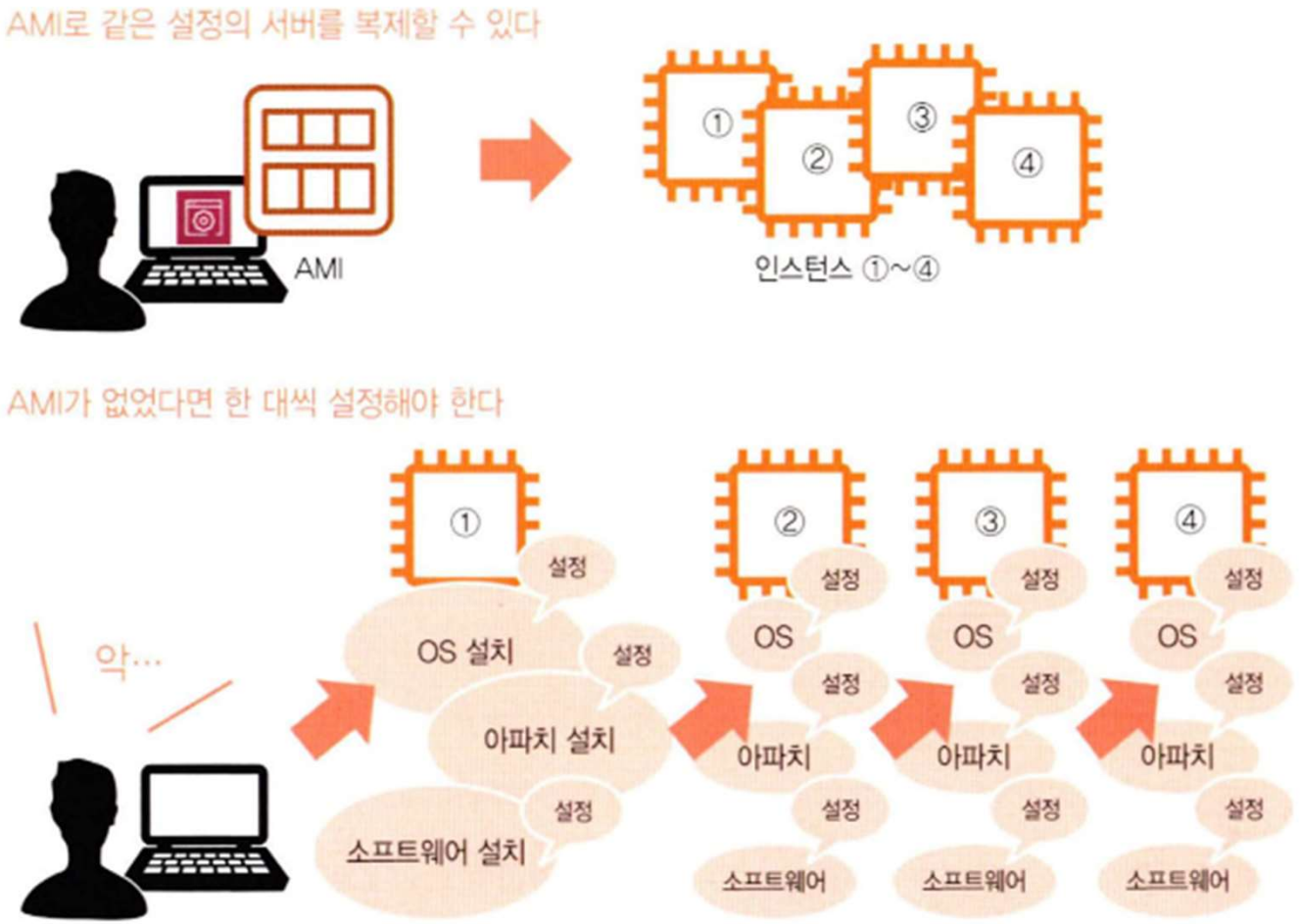
- ❖ Amazon EC2 인스턴스 생성
  - ✓ AMI
    - ● AMI 와 인스턴스
      - ◆ 같은 설정의 서버를 얼마든지 생성할 수 있다면 같은 서버를 여러 대 구축해야 할 경우에 편리한데 같은 서버를 생성하거나 삭제하기가 쉽다는 의미로 소프트웨어 설정까지 동일한 상황이라면 매우 유용
      - ◆ 동일한 서버가 여러 개 필요할 때 서버마다 서버 OS를 설치하고 아파치를 설치하고 소프트웨어를 설치하고 각각을 설정하고 와 같은 작업을 반복해야 하므로 시간과 노력이 많이 들지만 AMI를 사용하면 단 몇 분만에 동일한 서버를 만들 수 있음
      - ◆ AMI는 서버 디스크에 설치된 내용 전체가 들어 있는데 AMI에서 인스턴스를 생성하면 전부 복사되기 때문에 어떤 AMI라도 반드시 OS는 설치되어 있음

EC2 인스턴스 생성

❖ Amazon EC2 인스턴스 생성

✓ AMI

● AMI 와 인스턴스



EC2 인스턴스 생성

- ❖ Amazon EC2 인스턴스 생성
  - ✓ AMI
    - ● 제공되는 OS

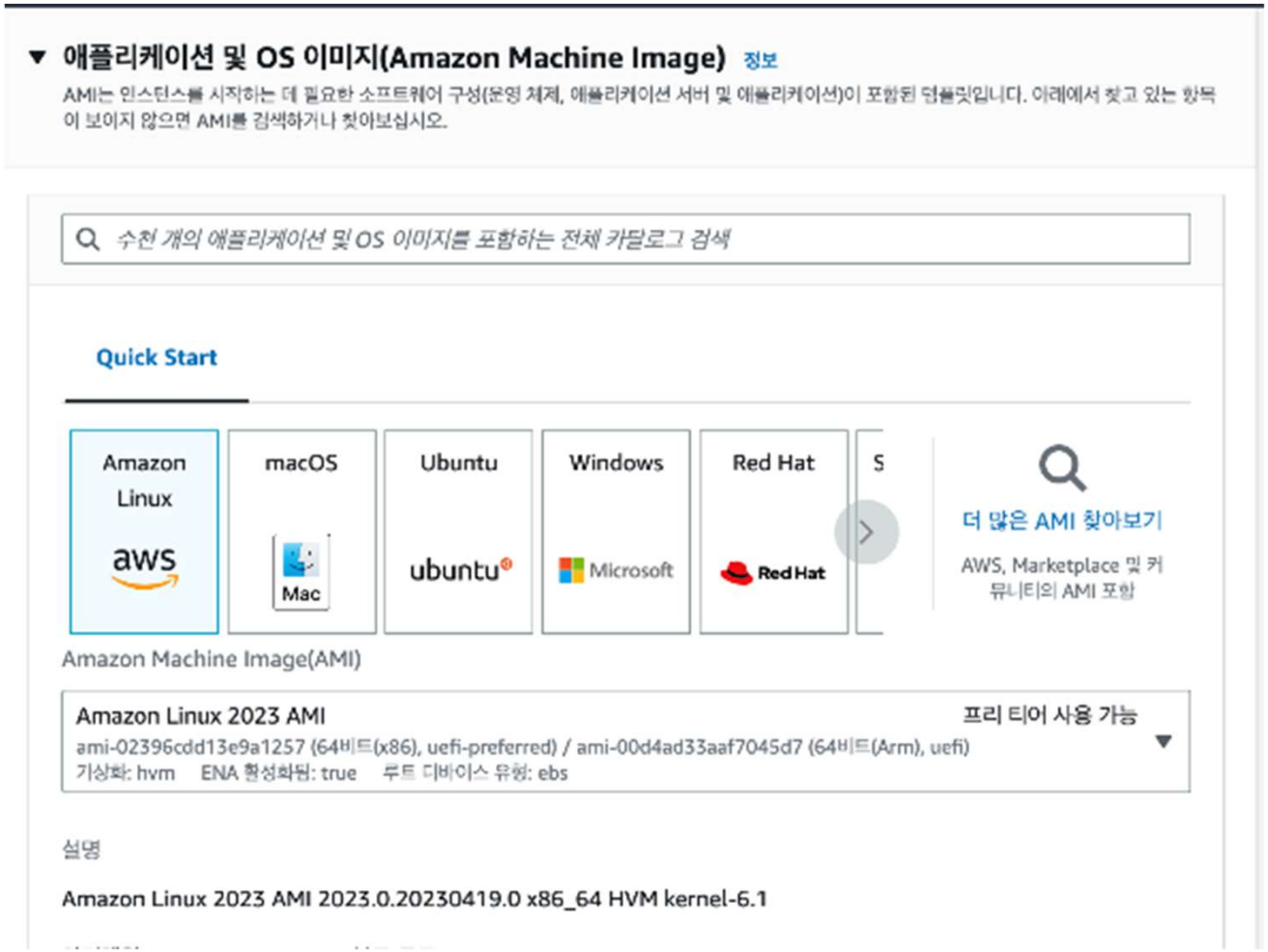
AMI	내용
Amazon Linux	아마존이 제공하는 레드햇(Red Hat) 기반 리눅스
CentOS	센트OS(CentOS) 리눅스
Red Hat Enterprise Linux	레드햇 리눅스
Debian GNU/Linux	데비안(Debian) 리눅스
Ubuntu Server	우분투(Ubuntu) 리눅스
SUSE Linux Enterprise Server	수세(SUSE) 리눅스
Microsoft Windows Server	윈도 서버(Windows Server)
LAMP Certified by Bitnami	리눅스와 아파치, MySQL PHP 서버
Tomcat Certified By Bitnami	자바 서블릿을 움직이는 톰캣(Tomcat) 서버
WordPress powered by AMIMOTO	블로그 시스템 워드프레스(WordPress)
Movable Type	블로그 시스템 무버블 타입(Movable Type)
NGINX Open Source Certified by Bitnami	웹 서버인 엔진엑스(Nginx) 서버
Redmne Certified by Bitnami	진행 현황 관리 도구인 레드마인(Redmine) 서버
NextCloud Powered by IVCISA	파일 공유 도구인 NextCloud 서버

EC2 인스턴스 생성

- ❖ Amazon EC2 인스턴스 생성
  - ✓ AMI
    - AMI 요금
      - ◆ AMI는 경우에 따라서 무료와 유료가 있는데 마이크로소프트가 제공하는 윈도 서버와 같이 원래 유료인 소프트웨어는 AMI에서도 유료이고 AMI를 개인이 생성할 경우에는 이미지의 용량에 따라서 요금이 부과
    - AMI는 개인이 만들 수 있기 때문에 같은 구성의 서버를 복제하거나 구축한 서버의 백업용으로 사용하면 편리하고 생성한 AMI는 마켓 플레이스라는 장소에서 배포할 수 있음
    - AMI는 EC2 인스턴스에서 생성하며 기본이 되는 인스턴스를 개인의 취향에 맞게 필요한 소프트웨어를 설정한 후 AMI로 내보냄
    - 마켓 플레이스란 AWS에서 AMI를 배포할 수 있는 장소로 유료, 무료에 관계없이 사용할 수 있는데 현재 수천 개의 AMI가 공개되어 있고 인스턴스를 생성할 때 사용할 수 있고 개인이나 기업 또는 커뮤니티에서 만든 AMI가 있는데 AMI가 반드시 안전하다는 보증은 없기 때문에 배포자를 잘 확인하고 사용

EC2 인스턴스 생성

- Amazon EC2 인스턴스 생성
  - ✓ AMI 선택
    - 운영체제 이미지를 선택



EC2 인스턴스 생성

- ❖ Amazon EC2 인스턴스 생성
  - ✓ 인스턴스 유형
    - 인스턴스 유형이란 머신의 용도
    - CPU, 메모리, 스토리지, 네트워크 용량 등이 용도에 맞게 조합되어 있음
    - 용도는 다섯 가지로 구분되어 있고 각 여러 인스턴스 유형이 존재
    - 종류
      - ◆ 범용: M1과 M3로 시작하는 인스턴스 유형으로 vCPU, 메모리, 네트워크, 저장 공간 등이 평균적인 사양으로 제공
      - ◆ 컴퓨팅 최적화: C로 시작하는 인스턴스 유형으로 다른 인스턴스 패밀리에 비해 메모리 대비 vCPU 비율이 높음
      - ◆ GPU 인스턴스: G로 시작하는 인스턴스 유형으로 고성능의 NVIDIA GPU가 장착되어 있어서 CUDA, OpenCL 등을 실행할 때 사용
      - ◆ 메모리 최적화: M2와 CR1로 시작하는 인스턴스 유형으로 다른 인스턴스 패밀리에 비해 메모리 용량이 훨씬 큼
      - ◆ 스토리지 최적화: H와 I로 시작하는 인스턴스 유형으로 다른 인스턴스 패밀리보다 스토리지 용량이 훨씬 크거나 초고속 I/O를 제공
      - ◆ 마이크로 인스턴스: 가격이 가장 싼 인스턴스로 낮은 vCPU 성능과 적은 메모리를 제공하는데 프리티어(무료 사용 계정) 에서는 이 인스턴스 유형을 무료로 사용할 수 있음

EC2 인스턴스 생성

❖ Amazon EC2 인스턴스 생성

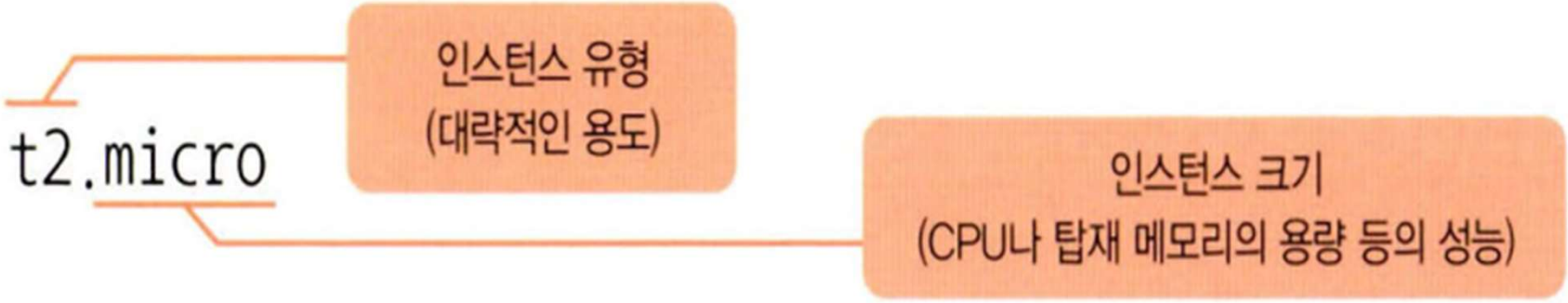
✓ 인스턴스 유형

EC2 인스턴스 생성

❖ Amazon EC2 인스턴스 생성

✓ 인스턴스 유형 과 크기

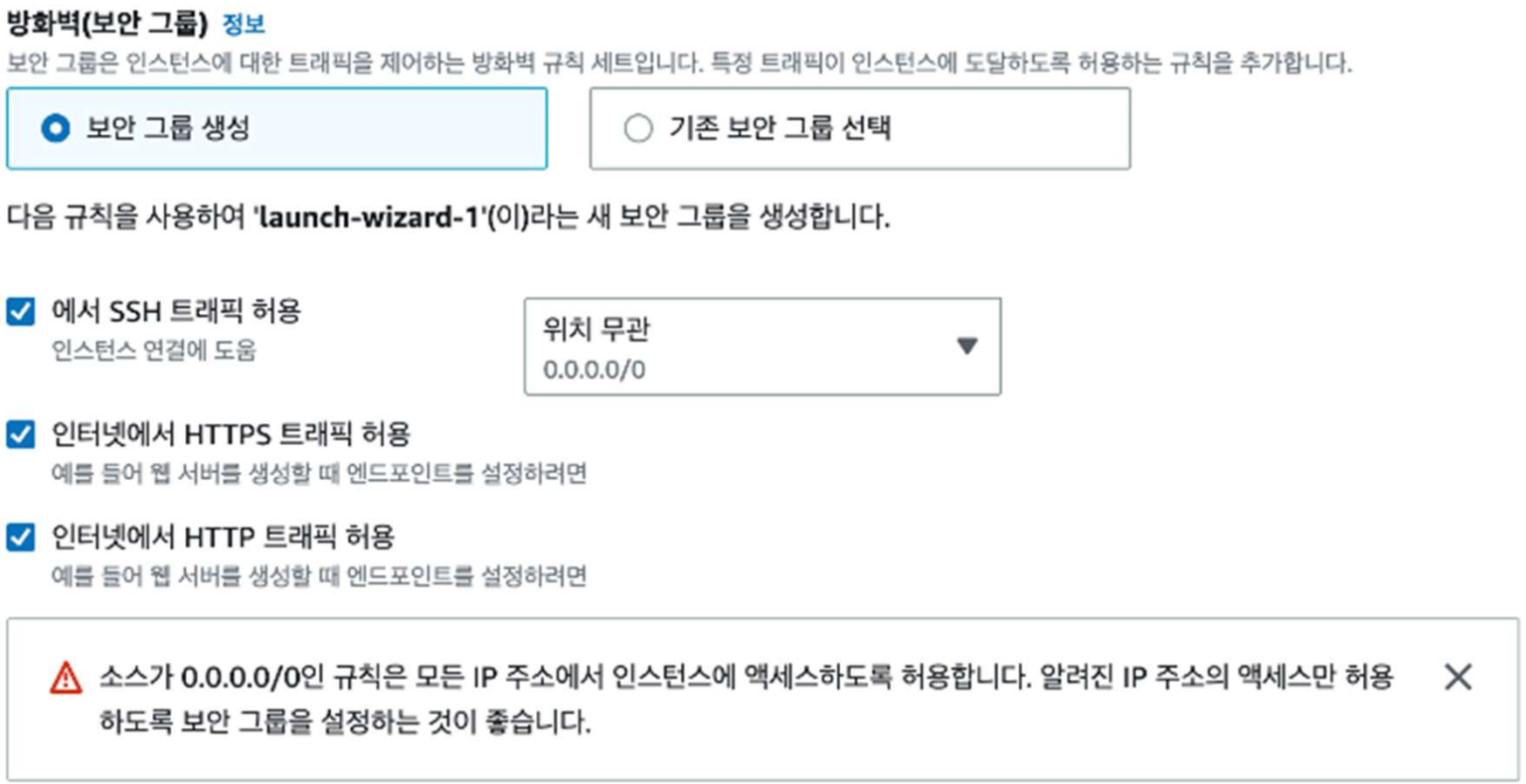
- 인스턴스의 유형과 크기를 선택하는데 이 두 가지의 조합이 성능을 의미
- T2라고 하는 일시적으로 성능을 올려주는 버스트 기능을 지원하는 범용 인스턴스 유형이 있는데 T2에는 nano, micro, small, medium, large, xlarge, 2xlarge, 4xlarge 8종류의 크기가 있고 규모에 맞춰서 선택하는 것도 가능하기 때문에 인스턴스를 선택하는 경우는 t2.micro처럼 유형과 크기를 붙여서 표기
- 유형 과 크기에 따라 단가가 달라지는데 단가 X 사용 시간 이 기본 요금



EC2 인스턴스 생성

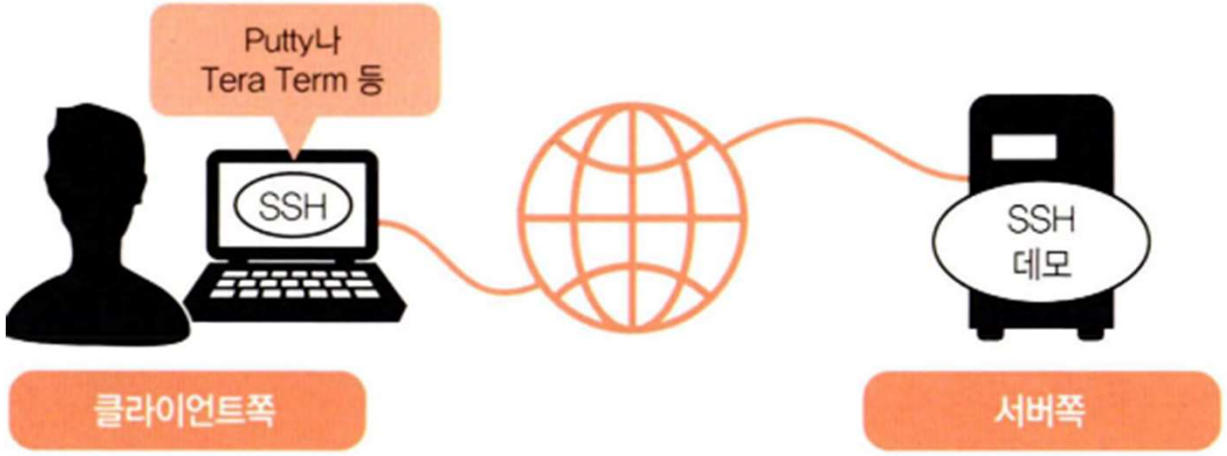
❖ Amazon EC2 인스턴스 생성

✓ 보안 그룹 생성



EC2 인스턴스 생성

- ❖ Amazon EC2 인스턴스 생성
  - ✓ 공개키 암호 방식을 이용한 접근 관리
    - ● SSH로 접속
      - ◆ 서버 성능을 올리거나 내리거나 백업하는 등 서버의 전반적인 작업은 관리 콘솔에서 수행하지만 서버에 설치한 소프트웨어를 조작하려면 SSH 방식을 사용해 원격 접속으로 조작하는 것이 일반적
      - ◆ 서버에서 SSH를 사용하기 위한 프로그램(데몬)을 기동하고 클라이언트에는 조작하기 위한 소프트웨어를 설치해야 하는데 서버에 SSH를 사용하기 위한 프로그램을 설치하지 않아도 OS에 이미 설치되어 동작하고 있으며 클라이언트는 Putty 나 Tera Term과 같은 소프트웨어를 주로 사용

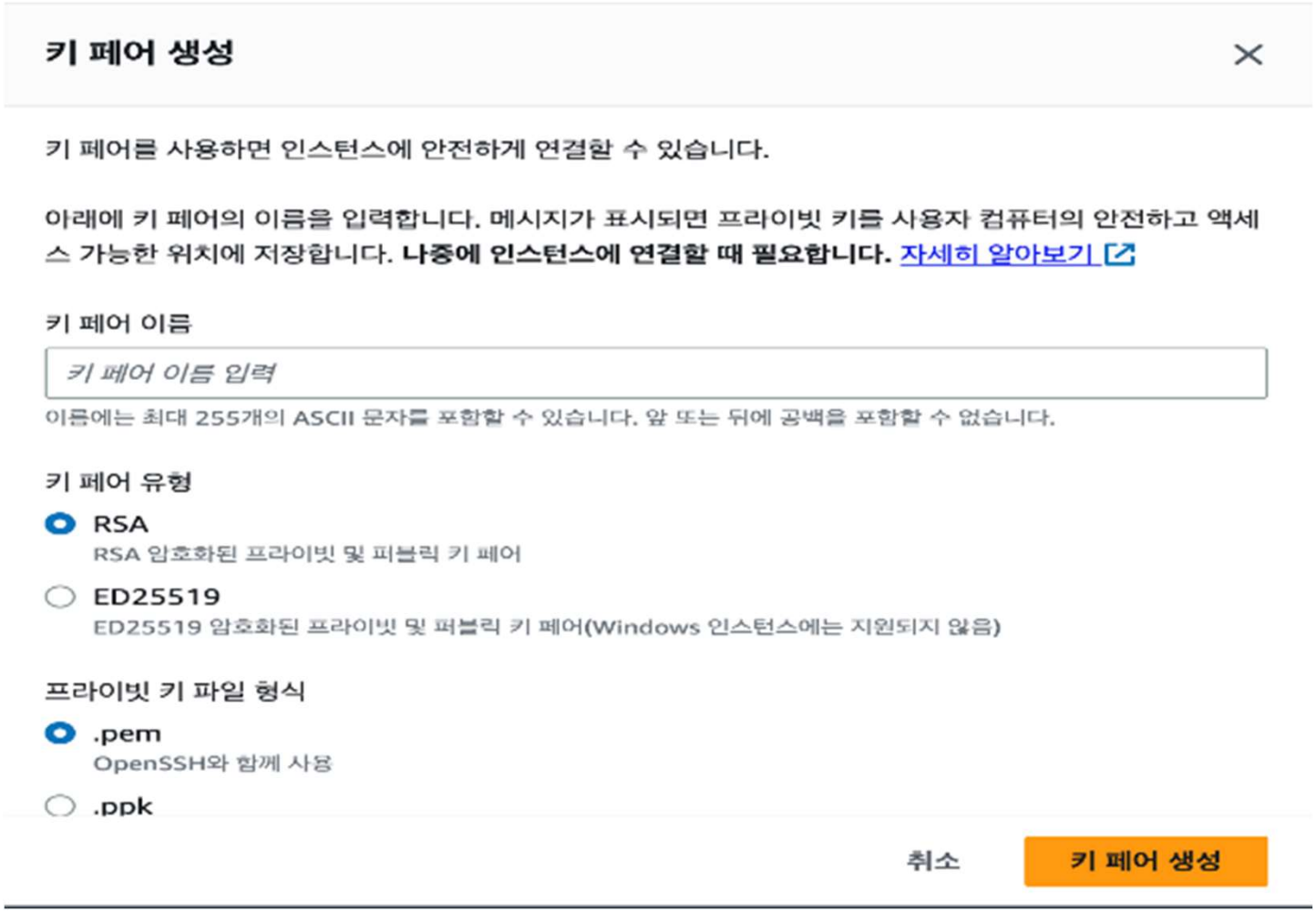


EC2 인스턴스 생성

- ❖ Amazon EC2 인스턴스 생성
  - ✓ 공개키 암호 방식을 이용한 접근 관리
    - ● 키 페어
      - ◆ 키 페어는 로그인할 때 인증으로 사용하며 공개키와 비밀키로 된 한 쌍
      - ◆ 공개키 방식이라고 불리며 자물쇠를 잠그는 키와 자물쇠를 여는 키를 공개키 와 비밀키로 조합하여 사용하는 방법
      - ◆ 자신 외에도 공개된 키를 공개키 그리고 자신만 알고 있는 키를 비밀키(개인키) 라고 하며 두 가지 키는 한 세트이며 AWS는 이 두 키를 파일 하나로 취급
      - ◆ SSH로 서버(인스턴스)에 접속할 때 인스턴스쪽에서 키 페어에 포함되어 있는 공개키를 지정하고 클라이언트쪽 소프트웨어에는 내려받은 키 페어 파일을 비밀키로 설정하여 사용
      - ◆ 키 페어는 생성했을 때만 내려받기가 가능하며 재발행이 안되므로 잊어버렸을 경우에는 서버를 재구축해야 함
      - ◆ 다른 리전에서는 사용할 수 없지만 같은 리전의 서비스라면 공통의 키 페어를 사용할 수 있으며 추가 비용이 들지 않음

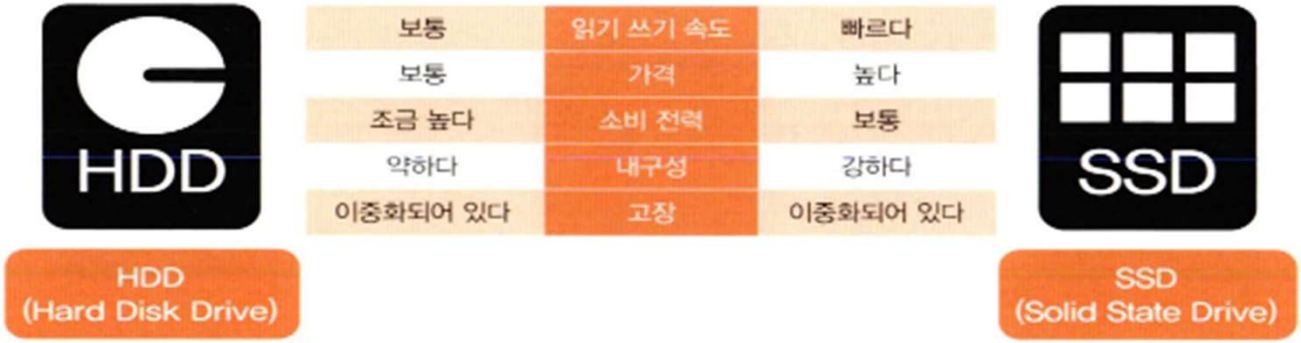
EC2 인스턴스 생성

- Amazon EC2 인스턴스 생성
  - ✓ Key-Pair 생성



EC2 인스턴스 생성

- ❖ Amazon EC2 인스턴스 생성
  - ✓ EBS
    - ● 개요
      - ◆ Amazon EBS(Amazon Elastic Block Store)는 영구적인 블록 스토리지 볼륨으로 EC2 인스턴스와 조합하여 사용
      - ◆ 스토리지는 데이터를 기록하는 장소로 대표적인 스토리지는 HDD 와 SSD
      - ◆ 블록 스토리지 볼륨은 데이터를 바이트 블록 단위로 디스크에 저장하는 일반적인 방식이고 같은 AWS 서비스인 Amazon S3는 오브젝트 스토리지라고 하는 방식을 채용
      - ◆ EBS는 HDD와 SSD를 선택할 수 있는데 HDD는 대용량을 지원하며 SSD에 비해 가격이 싸지만 성능이 낮고 SSD는 가격이 조금 비싸지만 IOPS(Input Output Per Second - 1초간 처리할 수 있는 입출력 수)가 빠르므로 고성능이라면 SSD로 저비용이라면 HDD가 적합

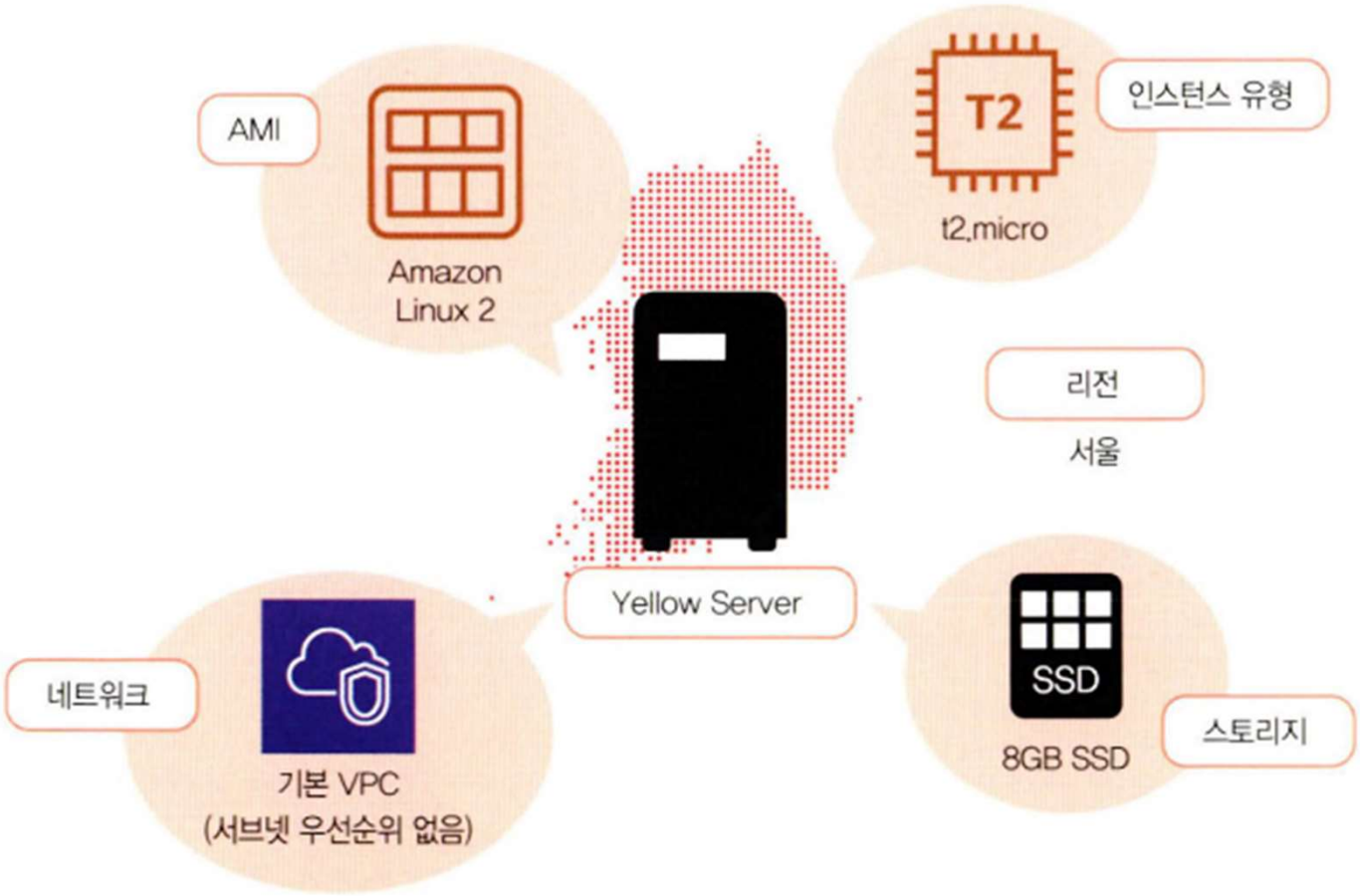


EC2 인스턴스 생성

- Amazon EC2 인스턴스 생성
  - ✓ EBS
    - 기능
      - 탄력적 볼륨: 볼륨 크기를 간단히 조정할 수 있는 기능
      - 스냅샷: 어떤 시점의 데이터를 통째로 저장하는 기능
      - 데이터 라이프 사이클 매니저: 일정에 따라서 스냅샷을 생성, 삭제하는 기능
      - 최적화 인스턴스: 특정 인스턴스 유형을 최적화 인스턴스로써 읽기 쓰기를 고속화 하는 기능
      - 암호화: 데이터 볼륨. 부팅 볼륨 및 스냅샷을 암호화하는 기능으로 KMS(AWS Key Management Service. 키를 생성/관리할 수 있는 기능)를 사용할 수 있음
    - 요금은 단가 X 저장 시간 으로 산출할 수 있는데 저장한 용량 단위의 과금(기가 바이트 단위)이므로 사용 여부와 관계없이 저장하고 있는 용량만큼 비용이 부과
    - 서버(인스턴스)를 정지해도 요금이 부과되며 단가는 디스크의 종류에 따라 다름

EC2 인스턴스 생성

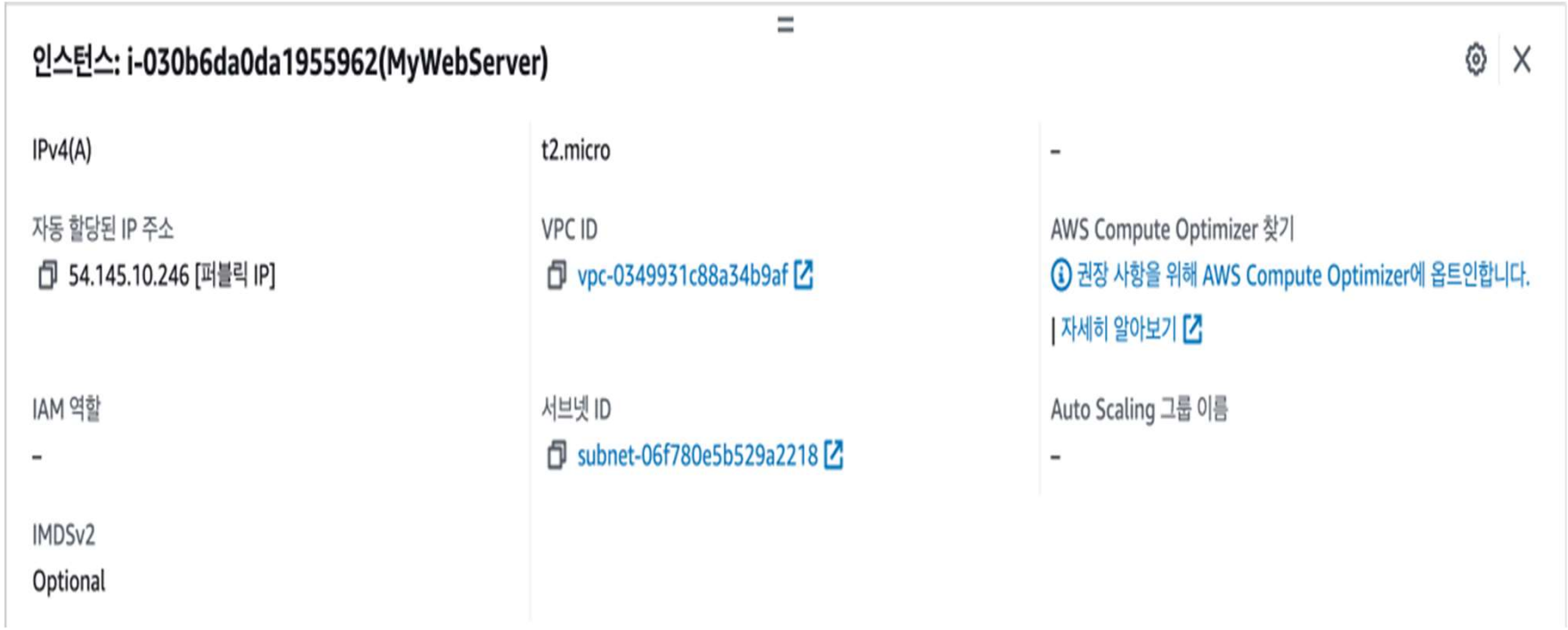
❖ Amazon EC2 인스턴스 생성



EC2 인스턴스 생성

❖ Amazon EC2 인스턴스 생성

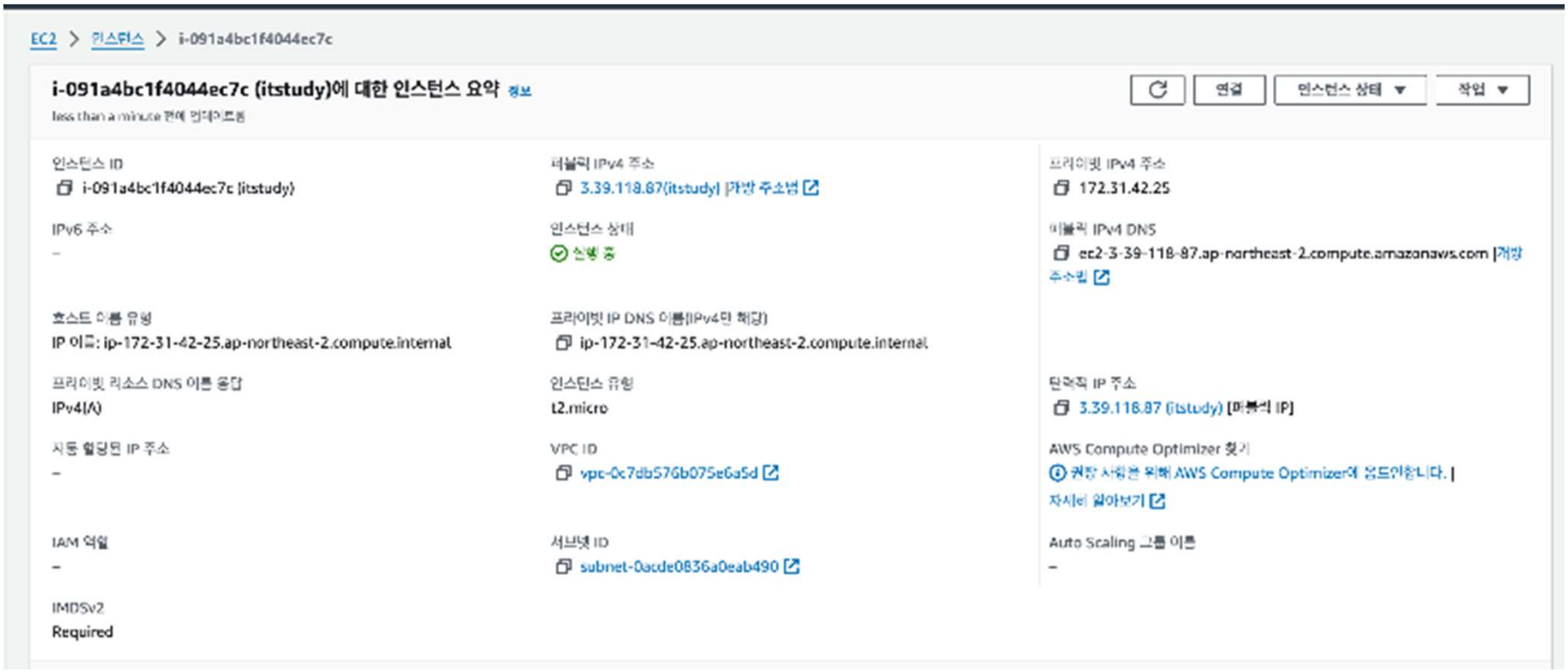
✓ IP 확인



EC2 인스턴스 생성

❖ 인스턴스 접속

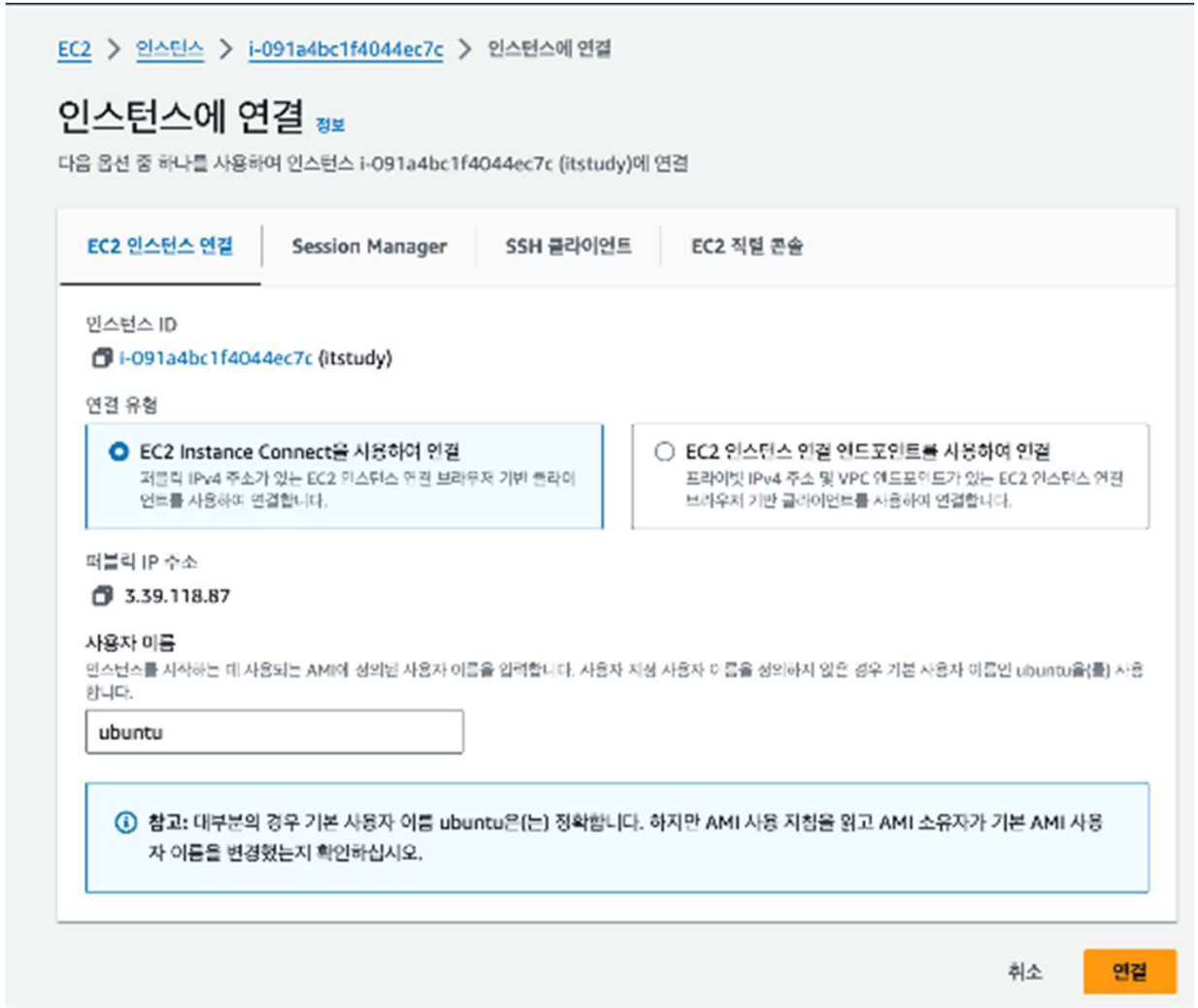
✓ AWS 사이트에서 연결



EC2 인스턴스 생성

❖ 인스턴스 접속

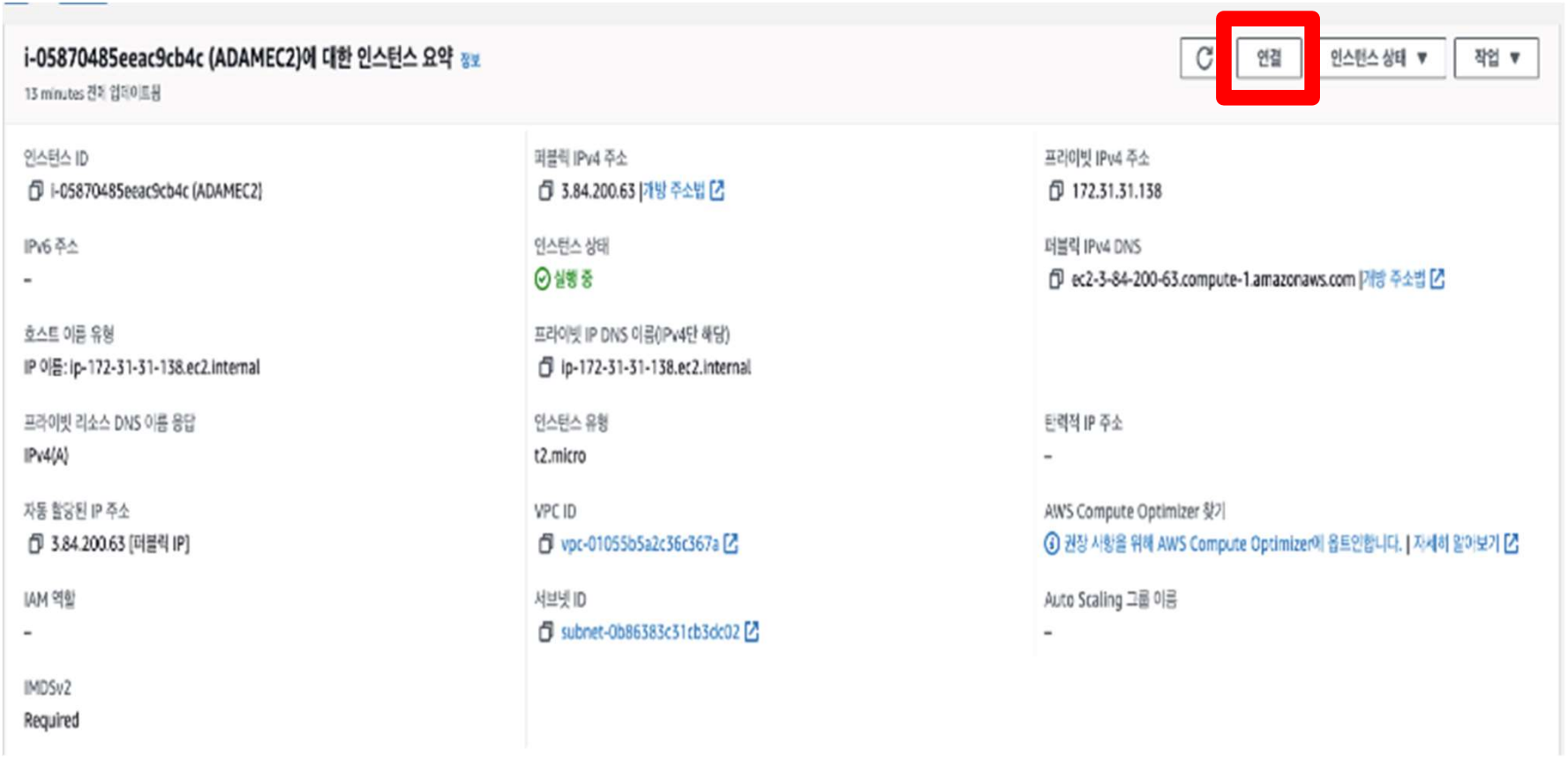
✓ AWS 사이트에서 연결



EC2 인스턴스 생성

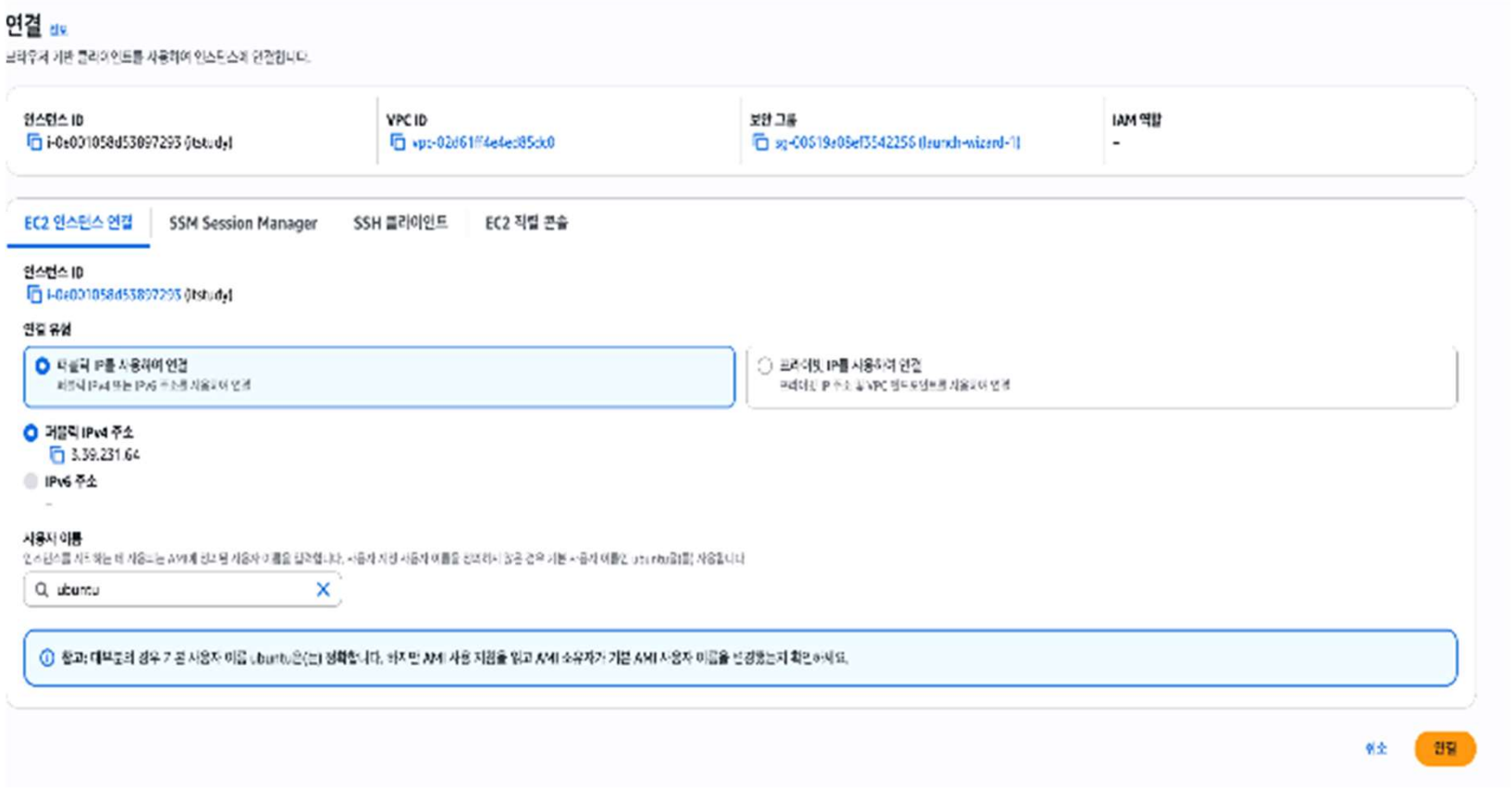
❖ 인스턴스 접속

✓ AWS 사이트에서 연결



EC2 인스턴스 생성

- ❖ 인스턴스 접속
  - ✓ AWS 사이트에서 연결



EC2 인스턴스 생성

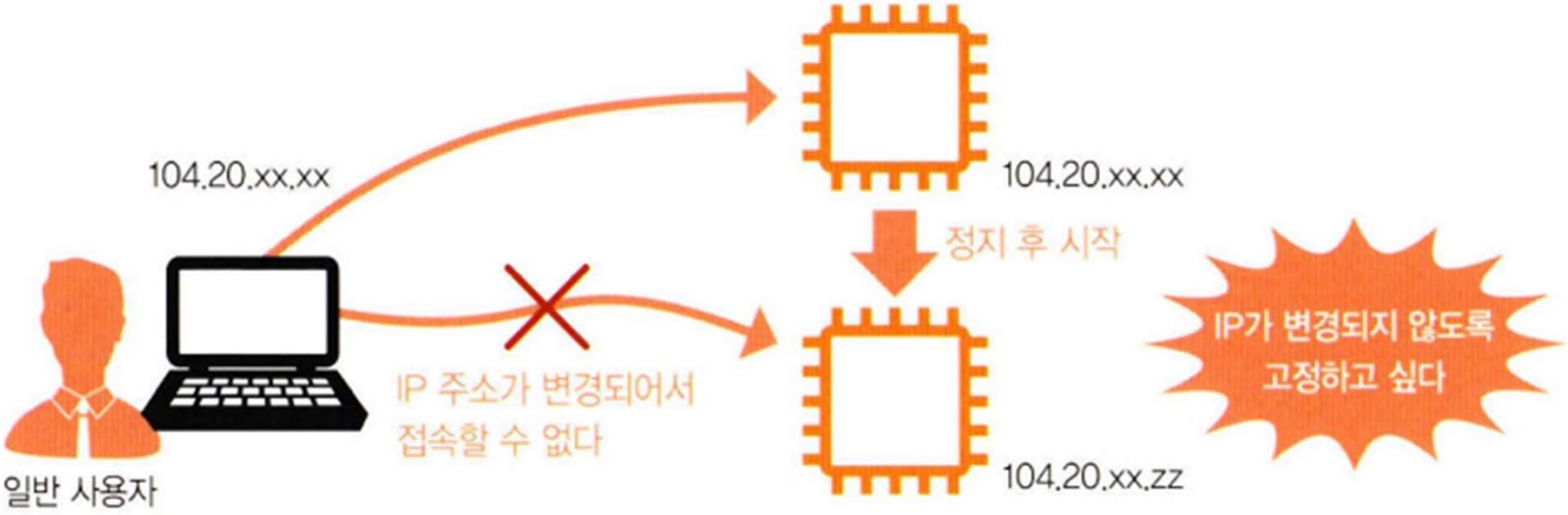
- ❖ 인스턴스 접속
  - ✓ AWS 사이트에서 연결

```
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1014-aws x86_64)

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

  System information as of Mon Apr  8 06:36:09 UTC 2024

  System load:  0.080078125       Processes:               99
  Usage of /:   20.7% of 7.57GB   Users logged in:         1
  Memory usage: 21%               IPv4 address for eth0: 172.31.31.138
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


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

Last login: Mon Apr  8 06:32:50 2024 from 1.223.55.43
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-31-138:~$
```

EC2 인스턴스 생성

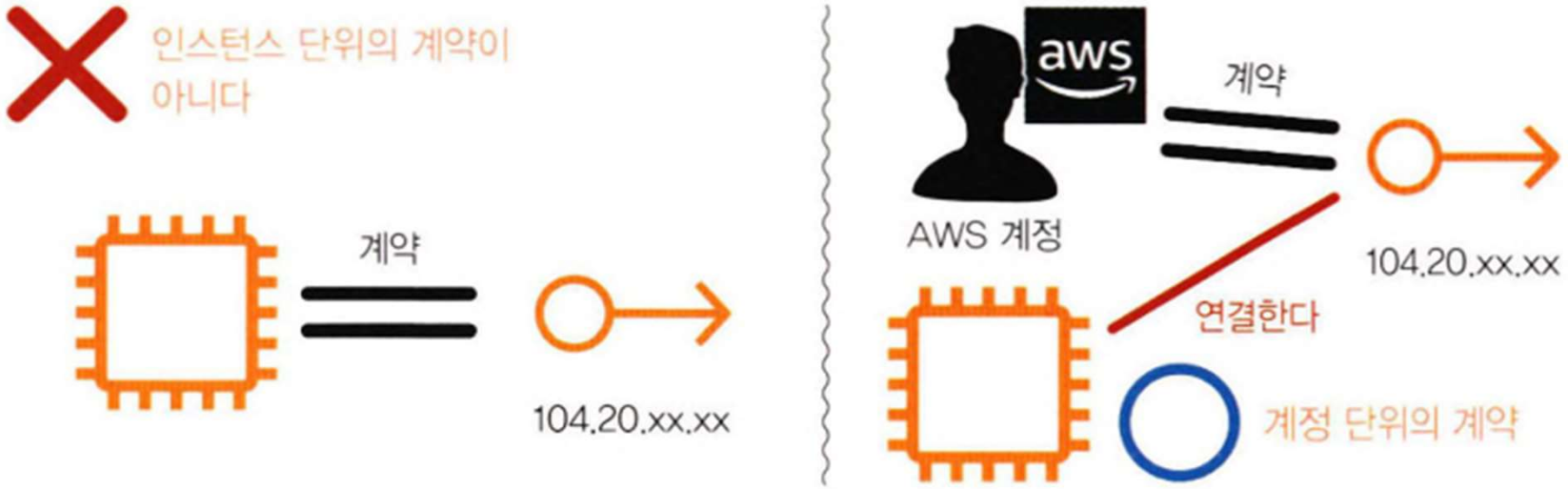
❖ Elastic IP

- ✓ Elastic IP 주소는 AWS가 제공하는 정적인 공인 IPv4 주소
- ✓ EC2 인스턴스는 정지 후 다시 시작하면 공인 IP 주소가 변경되는데 이는 서버로서 사용하는 데 문제가 되기 때문에 고정 IP 주소를 인스턴스와 연결해야 하고 이때 고정 IP로 사용되는 것이 Elastic IP 주소
- ✓ EC2 인스턴스를 생성하면 탄력적 IP를 대부분 필수적으로 생성



EC2 인스턴스 생성

- ❖ Elastic IP
  - ✓ Elastic IP 확보 와 부여
    - Elastic IP 주소는 AWS 계정에 연결되어 있는데 인스턴스 단위가 아니므로 IP 주소를 부여한 인스턴스를 삭제해도 확보한 IP 주소는 그대로 AWS 계정에서 소유하는 것이 가능
    - 보유하고 있는 IP 주소는 다른 인스턴스나 네트워크에 부여할 수 있음
    - 인스턴스에 이미 할당되어 있는 공인 IP 대신 Elastic IP 주소를 할당하면 AWS 공인 IPv4 주소 풀(인스턴스에 할당하기 위해 확보한 IP 주소)로 되돌아 감
    - 이미 할당되어 있는 IP 주소는 Elastic IP 주소로 사용할 수 없고 Elastic IP 주소는 리전에 속하므로 다른 리전이 보유하고 있는 Elastic IP 주소 역시 사용할 수 없음



EC2 인스턴스 생성

- Elastic IP
  - ✓ Elastic IP 요금
    - Elastic IP 주소의 요금은 기본적으로 인스턴스에 부여된 IP 한 개는 무료이고 추가로 IP를 연결하면 IP는 시간에 비례하여 요금이 부과되고 소유한 IP가 중지된 인스턴스나 분리된 네트워크에 연결된 경우에도 시간 당 요금이 부과되므로 사용하지 않는 IP 주소는 해지해야 함

EC2 인스턴스 생성

- ❖ Elastic IP
  - ✓ Elastic IP 발급 후 EC2 인스턴스에 연결
    - ● EC2 서비스의 왼쪽에서 탄력적 IP 클릭

EC2 인스턴스 생성

- ❖ Elastic IP
  - ✓ Elastic IP 발급 후 EC2 인스턴스에 연결
    - ● 오른쪽 상단에서 [탄력적 IP 주소 할당] 클릭

EC2 인스턴스 생성

- ❖ Elastic IP
  - ✓ Elastic IP 발급 후 EC2 인스턴스에 연결
    - ● 하단에서 [할당] 클릭

EC2 인스턴스 생성

❖ 인스턴스 접속

✓ 외부 컴퓨터에서 접속 – 키페어 이용해서 접속

- ssh -i {키페어 파일 경로} 계정@퍼블릭 IPv4
- 계정은 운영체제 별로 다른데 ubuntu에서는 ubuntu

EC2 인스턴스 생성

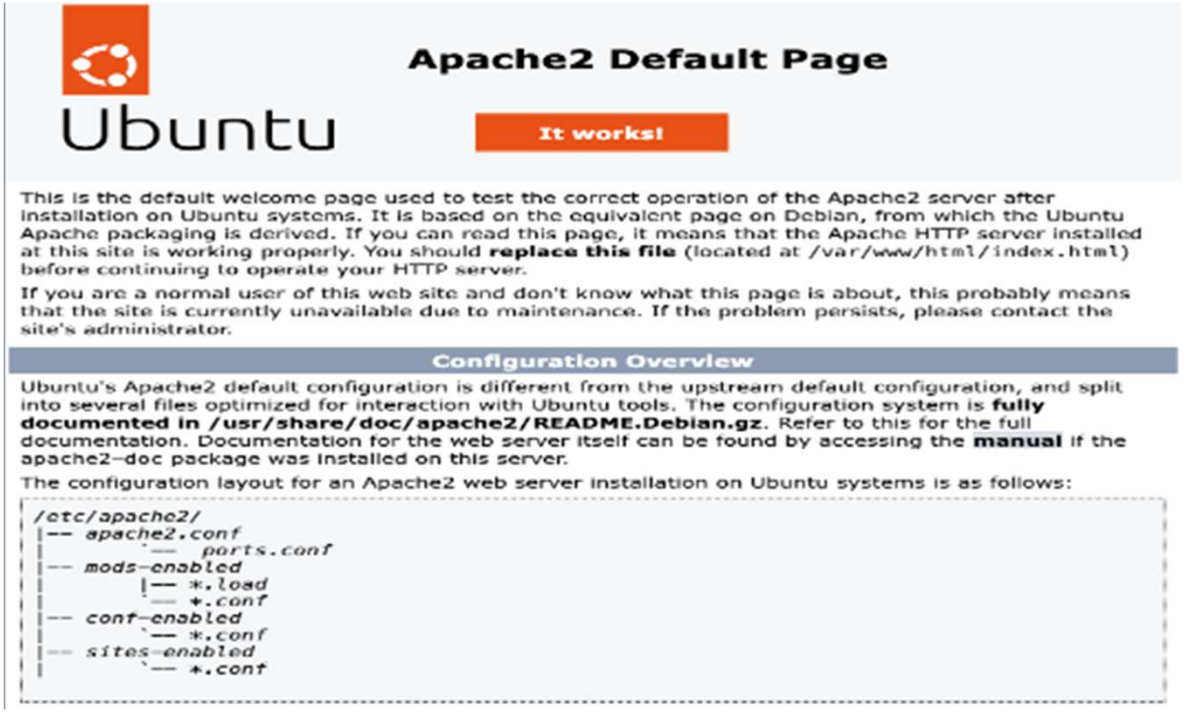
- ❖ 인스턴스 접속
  - ✓ 외부 컴퓨터에서 접속 – 키페어 없이 접속
    - 신규 유저 생성: sudo adduser [유저명]
    - 비밀번호 설정(ubuntu 계정의 경우): sudo passwd ubuntu
    - 유저 권한 변경
      - sudo chmod u+w /etc/sudoers
      - sudo vi /etc/sudoers
      - 파일 하단에 코드 추가: 유저명 ALL=(ALL:ALL) ALL
    - ssh 접속 방법 추가
      - sudo vim /etc/ssh/sshd_config
      - 코드 수정

        PasswordAuthentication yes

        KbdInteractiveAuthentication yes
    - ssh 재시작
      - sudo service ssh restart
    - 접속: ssh [유저명]@IP

EC2 인스턴스 생성

- ❖ 웹 서버 생성 및 애플리케이션 배포
  - ✓ 인스턴스에 접속해서 작업
    - 패키지 업데이트: sudo apt update
    - Apache Web Server 생성: sudo apt install apache2 -y
    - 웹 서버 서비스 시작: sudo service apache2 start
    - 확인: ps aux | grep apache2
    - 브라우저에서 http://공인IP 로 확인



EC2 인스턴스 생성

❖ 웹 서버 생성 및 애플리케이션 배포

✓ 인스턴스에 접속해서 작업

- cd /var/www/html
- ls
- sudo rm index.html
- sudo vi index.html

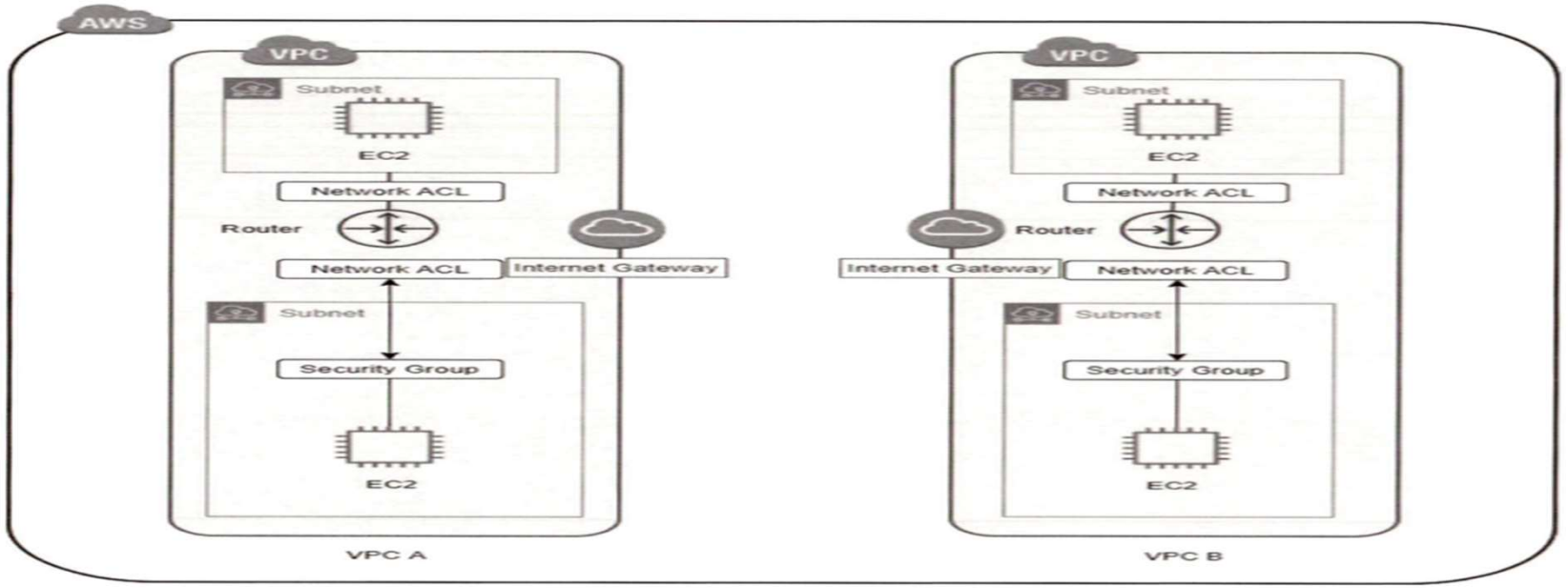
```
<html>
<body>
<hl>AWS EC2</hl>
</body>
</html>
```

EC2 인스턴스 생성

- 웹 서버 생성 및 애플리케이션 배포
  - ✓ 브라우저에서 http://공인IP 로 확인

애플리케이션 배포

- 보안 그룹
  - ✓ 보안 그룹은 가상 방화벽 역할을 수행
  - ✓ 보안 그룹은 인스턴스 앞에서 트래픽을 제어하는 가상 방화벽이고 네트워크 ACL은 서브 네트워크 앞에서 트래픽을 제어하는 역할
  - ✓ 인스턴스에 접근하기 전에 앞문의 성격인 네트워크 ACL 과 중문의 성격인 보안 그룹이 있음
  - ✓ Amazon VPC에서는 인스턴스를 시작할 때 최대 5개의 보안 그룹에 인스턴스를 할당할 수 있음(하나의 인스턴스에 5개까지 보안 그룹을 연결할 수 있는 것이고 VPC 내에 보안 그룹은 2500개까지 가능)
  - ✓ 보안 그룹은 서브 네트워크가 아닌 인스턴스에서 작동하기에 VPC에 있는 서브 네트워크의 각 인스턴스를 서로 다른 보안 그룹으로 구성할 수 있음



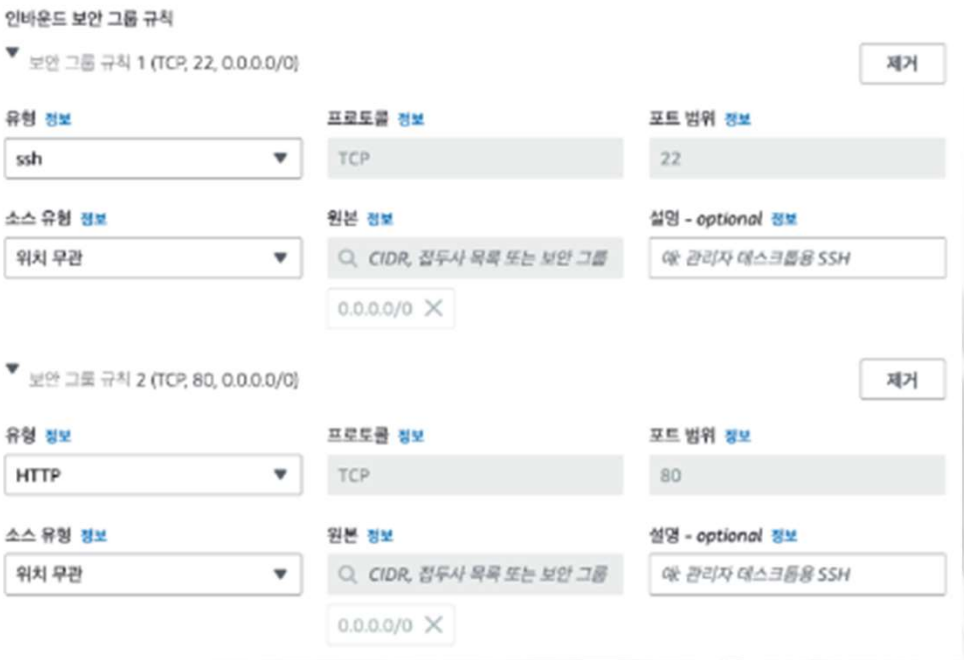
애플리케이션 배포

❖ 보안 그룹

✓ 인터넷에서 일부 사용자가 인스턴스에 접근(액세스)하려고 할 때 인스턴스에 방화벽 역할을 수행할 보안 그룹(Security Group)을 만들고 보안 그룹에 규칙을 지정하면 이 보안 규칙에는 인바운드 트래픽(외부에서 인스턴스로 보내는 트래픽)에서 어떤 트래픽을 허용할 지 설정할 수 있고 아웃바운드 트래픽(인스턴스에서 외부로 나가는 트래픽)에서는 어떤 트래픽을 허용할 지 설정할 수 있음

✓ 보안 그룹을 설정할 때는 허용할 IP 범위와 포트(port)를 설정할 수 있음

✓ 설정

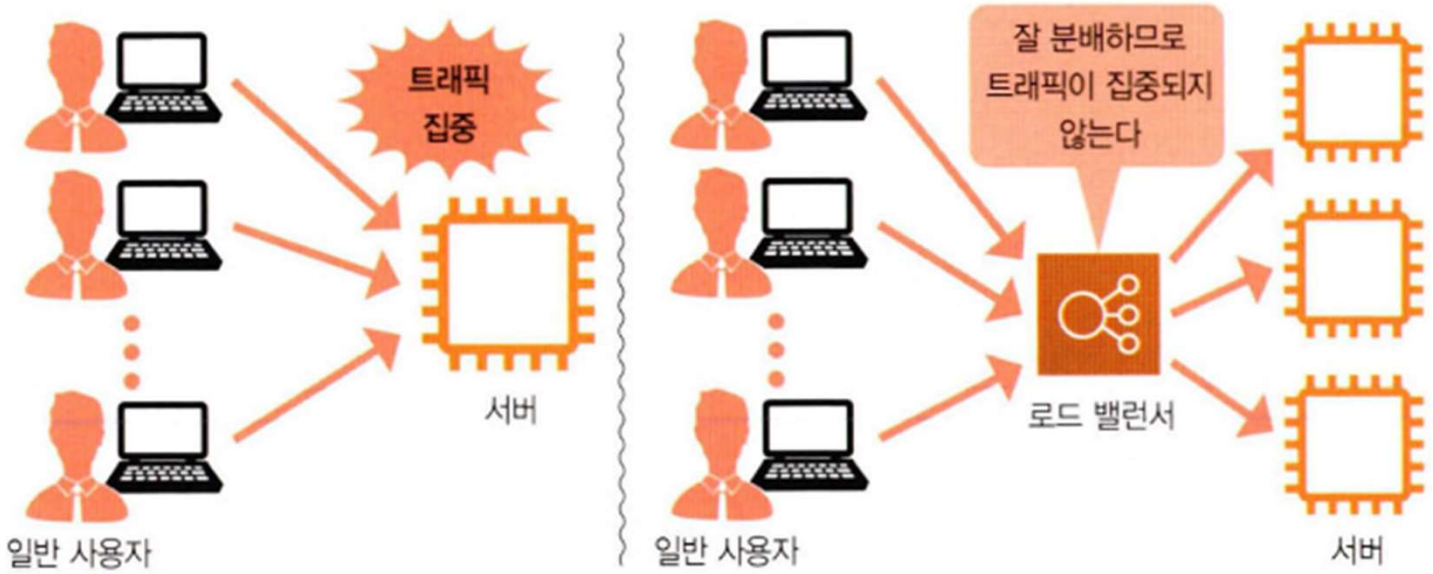


애플리케이션 배포

❖ Load Balancing

✓ ELB

- AWS가 제공하는 Load Balancer
- Load Balancer는 집중되는 접속(트래픽)을 여러 대나 네트워크에 분배하는 장비
- 한 대에 집중되는 부하를 분산시키기 때문에 부하 분산 장치 라고도 함



애플리케이션 배포

- ❖ Load Balancing
  - ✓ ELB 종류
    - ● ALB(Application Load Balancer)
      - ◆ HTTP 및 HTTPS에 가장 적합한 Load Balancer로 OSI 모형의 애플리케이션 계층(구체적인 통신을 제공하는 계층)에서 동작
      - ◆ 요청되는 명령어의 내용을 보고 판단하기 때문에 대상의 URL 디렉터리 단위로 분배하는 것이 가능하며 인스턴스와 Load Balancer 사이의 통신은 암호화가 가능하다는 것도 특징 중 하나이지만 분배 대상으로 정적 IP 주소를 설정하고 그 IP를 가진 호스트(기기)로 전송할 수 없음
      - ◆ 지원 프로토콜: HTTP. HTTPS
    - ● NLB(Network Load Balancer)
      - ◆ OSI 모형의 전송 계층(전송된 데이터의 제어를 담당하는 계층)에서 동작하는데 패킷이라고 불리는 단편 데이터 밖에 볼 수 없기 때문에 ALB만큼 상세하게 분배할 수 없지만 대신 분배 대상의 정적 IP 주소를 설정할 수 있고 서버에 접속한 클라이언트의 IP 주소를 그대로 서버에 전송하도록 설정할 수도 있음
      - ◆ 지원 프로토콜: TCP, TLS
    - ● CLB(Classic Load Balancer)
      - ◆ 오래된 유형의 Load Balancer로 지원하는 프로토콜이 많다는 것이 장점이지만 앞으로 구축할 시스템에는 사용하지 않는 것을 권장
      - ◆ 지원 프로토콜: TCP, SSL/TLS, HTTP, HTTPS

애플리케이션 배포

- ❖ Load Balancing
  - ✓ ELB 종류
    - ● GLB(Gateway Load Balancer)
      - ◆ AWS 환경으로 들어오고 나가는 네트워크 트래픽을 차세대 방화벽(NGFW), 침입 탐지 및 방지 시스템(IDS/IPS), 심층 패킷 검사(DPI) 시스템 같은 서드파티 가상 네트워크 가전(Virtual Appliance) 장비들로 쉽고 안전하게 라우팅하고 이 장비들을 관리할 수 있게 해주는 부하 분산 서비스
      - ◆ 기본의 Application Load Balancer(ALB)나 Network Load Balancer(NLB)는 사용자의 요청을 애플리케이션 서버(EC2)로 분산하는 데 특화되어 있는데 기업에서 보안을 강화하기 위해 Palo Alto, Check Point, Fortinet 같은 전문 방화벽 장비(Appliance)를 AWS 환경에 도입하려고 할 때는 발생하는 문제
        - ▪ 복잡한 라우팅 구조: 트래픽이 반드시 방화벽을 거쳐 가도록 라우팅 테이블을 이리저리 꼬아서 설정해야 함
        - ▪ 고가용성(HA)의 한계: 특정 방화벽 장비가 죽으면 네트워크 전체가 마비되거나, 이를 복구하기 위한 스크립트를 짜야 함
        - ▪ 스케일링의 어려움: 트래픽이 몰릴 때 방화벽 장비의 개수를 자동으로 늘리거나 줄이기가 매우 까다로움
      - ◆ GWLB는 이 문제를 완전히 해결해 주는 트래픽의 게이트웨이 역할을 하면서 동시에 방화벽 장비들을 뒤에 묶어(Target Group) 자동으로 부하를 분산하고, 장비가 고장 나면 알아서 정상적인 장비로 트래픽을 돌려줌

애플리케이션 배포

- ❖ Load Balancing
  - ✓ ELB 종류
    - ● GLB(Gateway Load Balancer)
      - ◆ 장점
        - ▪ 투명성(Bump-in-the-wire): 원본 패킷의 출발지 IP, 목적지 IP, 포트 등이 전혀 변하지 않고 그대로 유지되므로, 보안 장비가 정확하게 로그를 분석하고 차단할 수 있음
        - ▪ 자동 고가용성(Health Check): 뒤에 연결된 가상 방화벽 장비들의 상태를 계속 체크하여, 비정상적인 장비가 발견되면 즉시 트래픽 할당을 중단
        - ▪ 쉬운 확장성: 트래픽 양에 따라 방화벽 인스턴스를 Auto Scaling 그룹으로 묶어 자동으로 늘리고 줄일 수 있어 비용 측면에서 효율적
        - ▪ VPC 간 분리(중앙집중형 보안): 보안 장비가 있는 VPC와 실제 서비스가 돌아가는 VPC를 분리하여, 여러 VPC의 트래픽을 하나의 중앙 보안 VPC(GWLB)에서 통합 관리할 수 있음

애플리케이션 배포

- ❖ Load Balancing
  - ✓ Target Group
    - ● 개요
      - ◆ AWS에서 대상 그룹(Target Group)은 로드 밸런서가 받은 트래픽을 최종적으로 어디로 보낼지 지정하는 목적지 안내 대장(또는 그룹)
      - ◆ 로드 밸런서(ALB, NLB, GLB)를 구성할 때 필수적으로 함께 만들어야 하는 개념으로 트래픽을 처리할 실제 서버나 자원들을 하나의 바구니에 담아두고 관리하는 역할을 수행
    - ● 역할
      - ◆ 트래픽 라우팅(목적지 지정): 로드 밸런서에 80번 포트로 들어오는 요청은 A 대상 그룹으로 보내고, 443번 포트로 들어오는 요청은 B 대상 그룹으로 보내라 와 같은 규칙(Listener Rule)을 지정하면 대상 그룹은 그 요청을 받아 실제 내부에 등록된 타겟들에게 골고루 나눠줌
      - ◆ 상태 검사 (Health Check): 대상 그룹의 가장 중요한 임무 중 하나로 그룹에 속한 서버들이 정상적으로 작동하고 있는지 주기적으로 Health Check를 수행
      - ◆ 유연한 컴포넌트 관리(EC2 교체의 편리함): 서버를 교체하거나 추가할 때 로드 밸런서를 건드릴 필요가 없이 대상 그룹 안에서 EC2 인스턴스를 빼거나 넣기만 하면 로드 밸런서가 알아서 변경된 서버들로 트래픽을 보냄(Auto Scaling 그룹과 연동할 때도 이 대상 그룹을 지정)

애플리케이션 배포

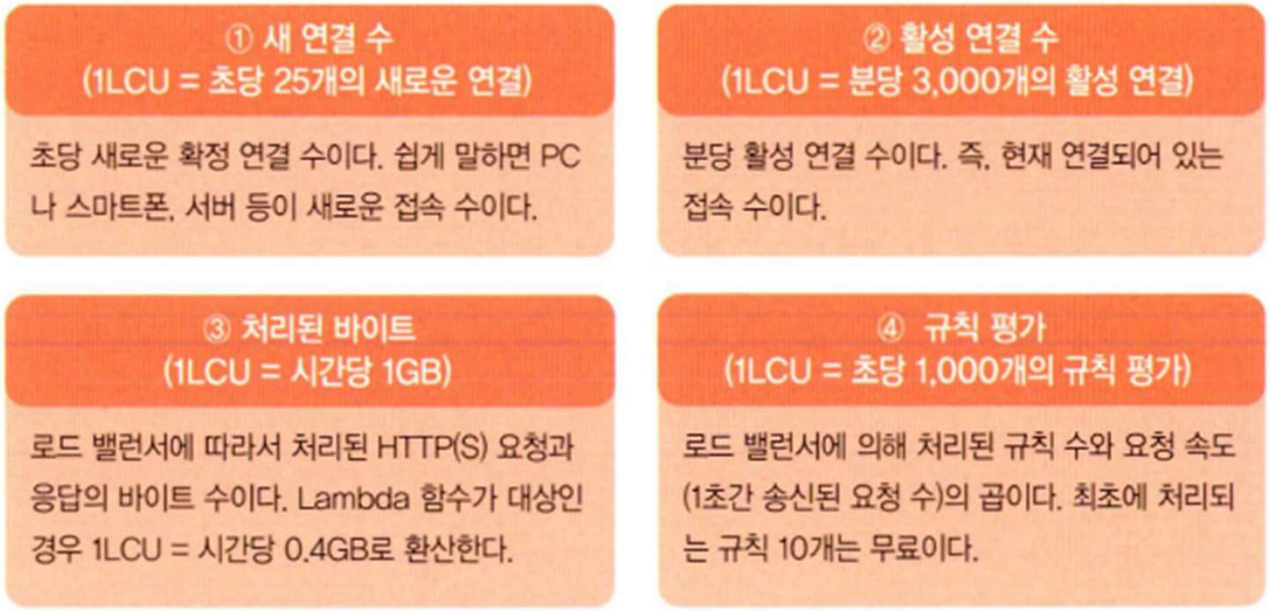
- ❖ Load Balancing
  - ✓ Target Group
    - ● 대상 그룹에 등록할 수 있는 대상(Target)의 종류
      - ◆ 인스턴스(Instance) 기반: 가장 흔한 방식입니다. 특정 EC2 인스턴스들을 직접 지정하여 그룹에 넣음
      - ◆ IP 주소 기반: 특정 IP 주소를 대상으로 지정하는데 AWS 내부의 VPC 안에 있는 자원 뿐 아니라 Direct Connect나 VPN으로 연결된 사내(On-Premises) 서버의 IP도 대상으로 묶을 수 있음
      - ◆ Lambda 함수 기반: 서버리스 아키텍처를 쓸 때 사용하는데 로드 밸런서로 들어온 요청을 AWS Lambda 함수가 받아 처리하도록 연결
      - ◆ Application Load Balancer 기반: 로드 밸런서 뒤에 또 다른 로드 밸런서를 두는 구조(AWS 가상 방화벽 장비를 거쳐 ALB로 갈 때)를 만들 때 사용

애플리케이션 배포

❖ Load Balancing

✓ ELB 요금

- ALB와 NLB의 요금은 시간당 사용 요금과 LCU(Load Balancer 용량 단위) 요금의 합계로 계산
- ALB의 단가는 1시간당 0.0225 US달러 LCU 요금은 1시간당 0.005 US달러 정도
- LCU의 4가지 항목

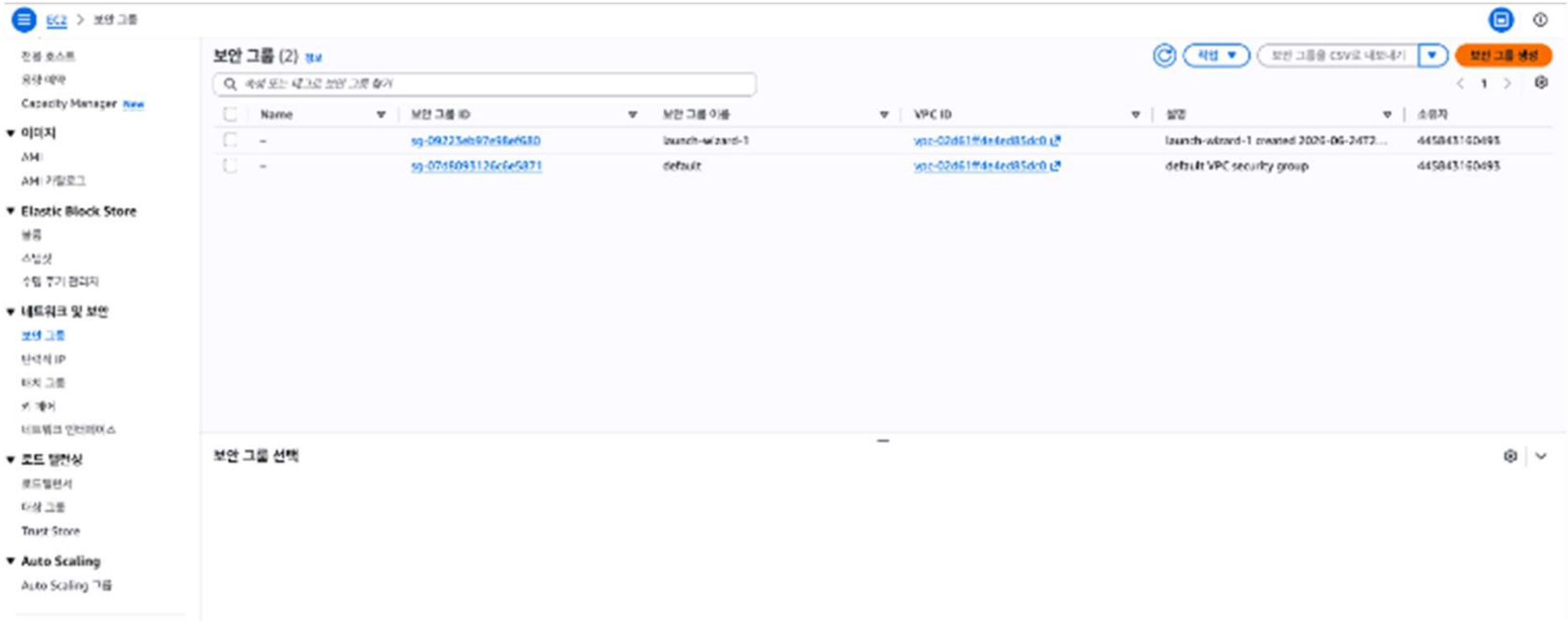


애플리케이션 배포

- ❖ Load Balancing
  - ✓ ELB에서 발생하는 에러
    - ● Elastic Load Balancer에는 최대 접속 시간 제한이라는 설정이 있는데 디폴트는 60초로 지정되어 있으며 Load Balancer가 60초 동안 아무런 데이터도 전달받지 못할 경우 연결을 자동으로 종료하고 타임아웃 에러를 생성하는데 ELB를 사용할 때 발생하는 가장 대표적인 에러는 애플리케이션 또는 데이터 서버에서 특정 시간 내에 응답을 받지 못할 경우 생기는 Load Balancer Error: 504 ERROR 인데 이 문제는 서버 레이어 나 데이터베이스 레이어에서 발생하기 때문에 상대적으로 쉽게 해결할 수 있고 현재 돌아가는 애플리케이션의 규모가 크다면 최대 접속 시간 제한을 변경해서 Load Balancer가 기다려줄 수 있는 시간을 늘려서 해결하거나 직접 애플리케이션을 수정해서 서버로 전송되는 데이터의 양을 조절하거나 서버에서의 응답 시간을 최적화 시켜야 함

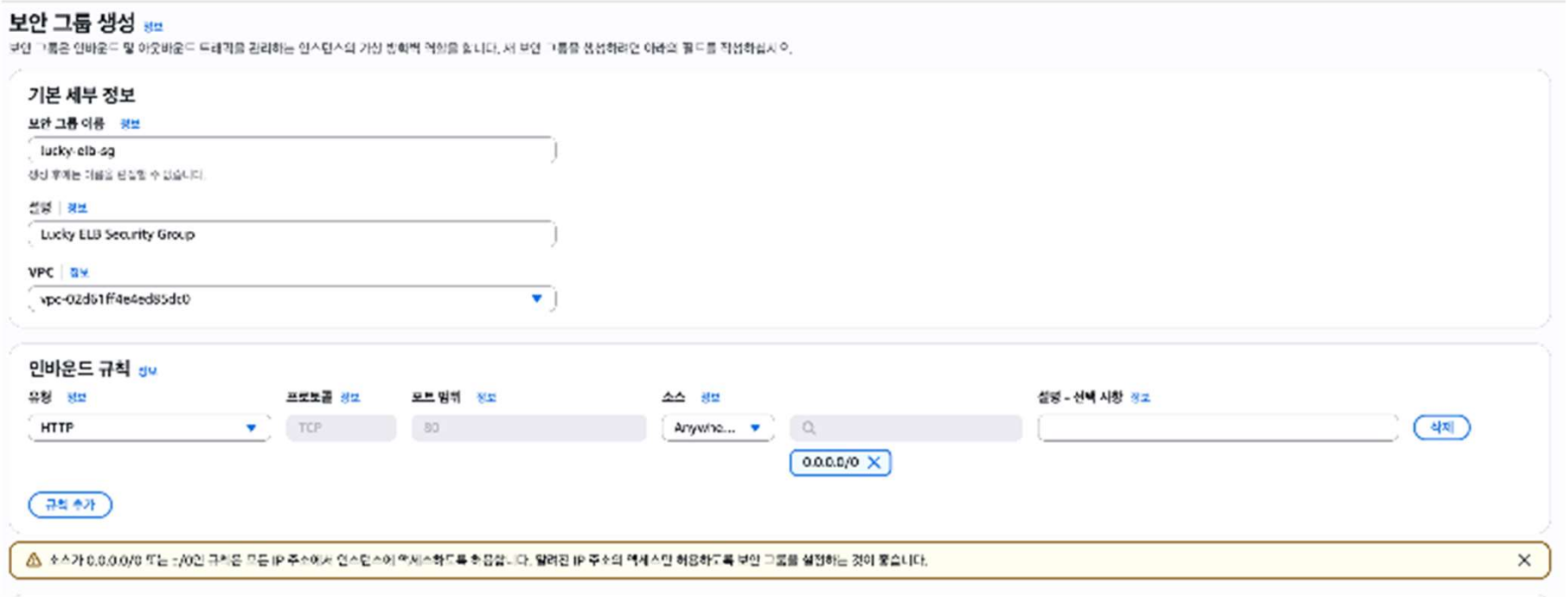
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● 보안 그룹 생성 – InBound 규칙에서 80번 포트에 모든 트래픽을 허용
      - ◆ 보안 그룹 생성 버튼 클릭



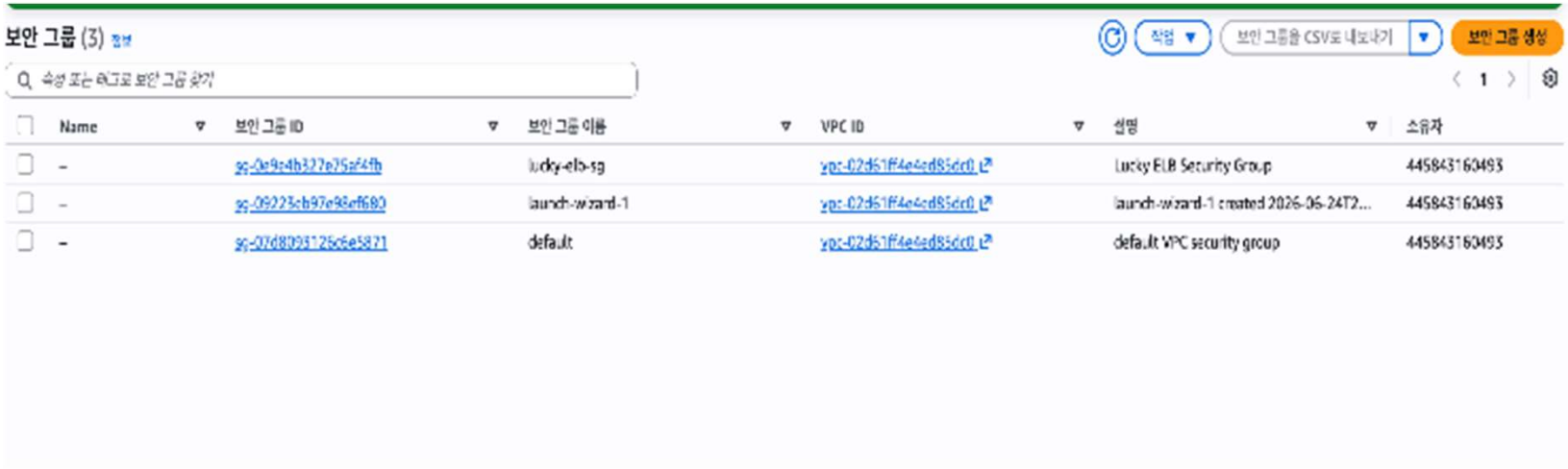
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● 보안 그룹 생성 – InBound 규칙에서 80번 포트에 모든 트래픽을 허용
      - ◆ 보안 그룹 설정



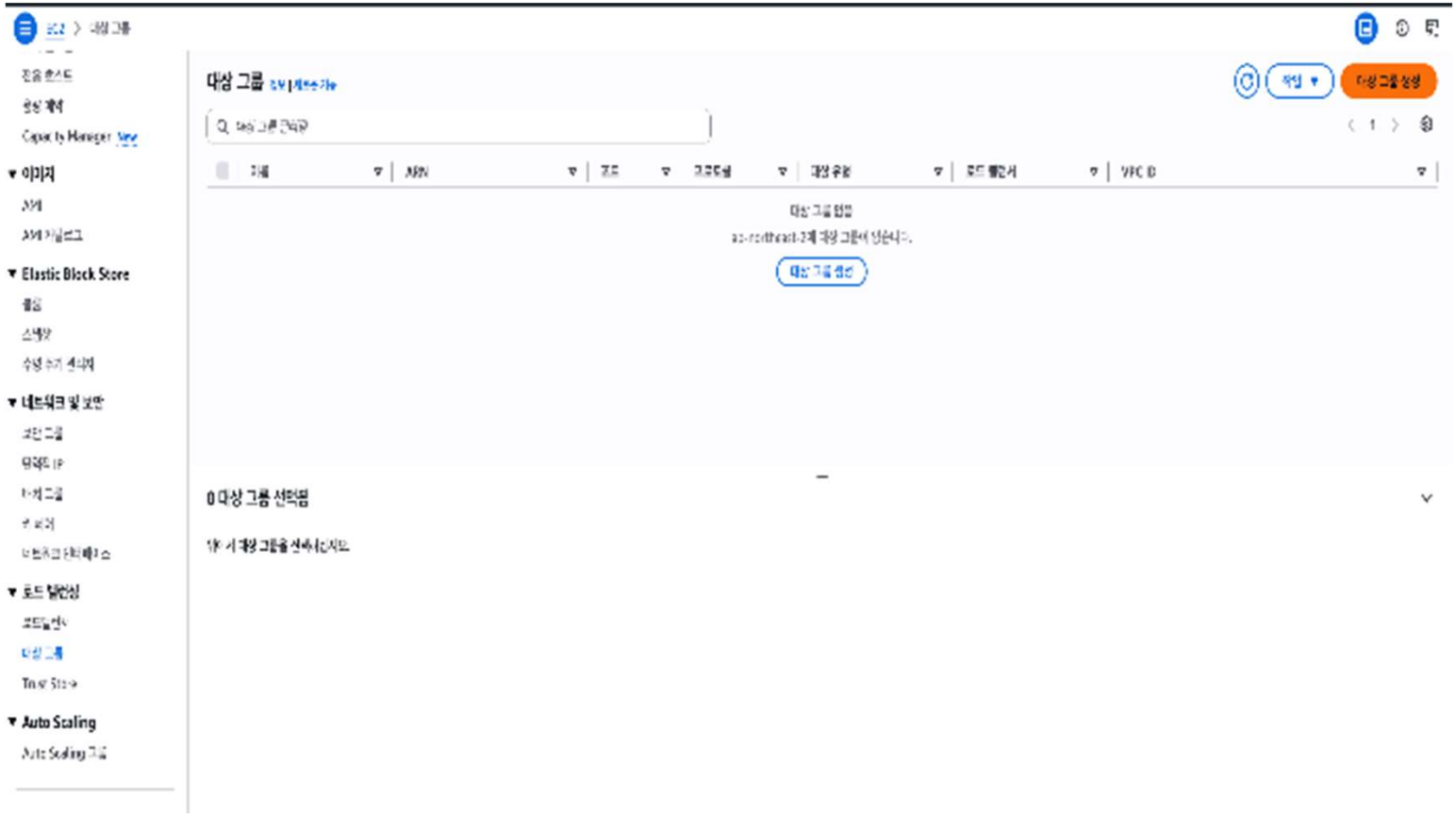
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● 보안 그룹 생성 – InBound 규칙에서 80번 포트에 모든 트래픽을 허용
      - ◆ 보안 그룹 확인



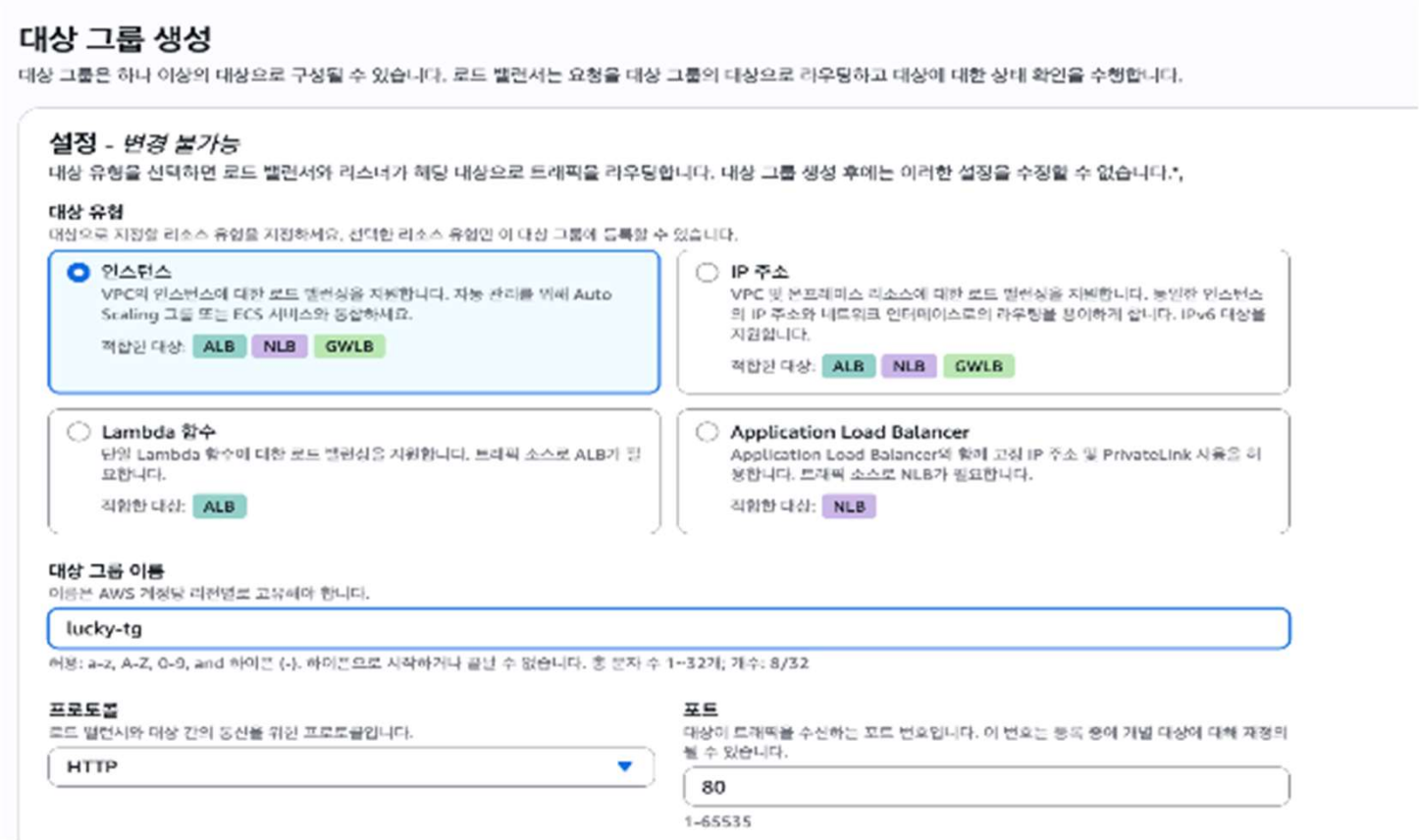
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● 대상 그룹 생성 – 앞에서 만든 인스턴스에 트래픽을 전송하는 대상 그룹 생성
      - ◆ 대상 그룹 생성 버튼 클릭



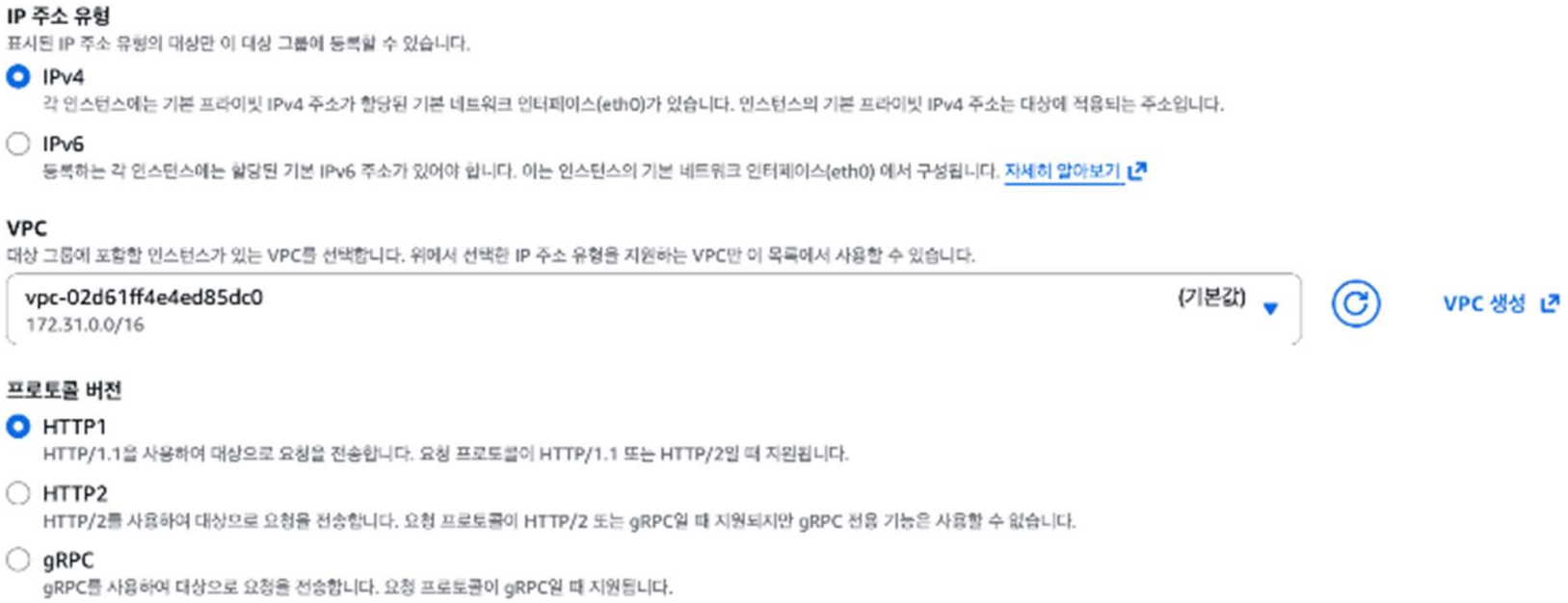
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● 대상 그룹 생성 – 앞에서 만든 인스턴스에 트래픽을 전송하는 대상 그룹 생성
      - ◆ 대상 그룹 설정



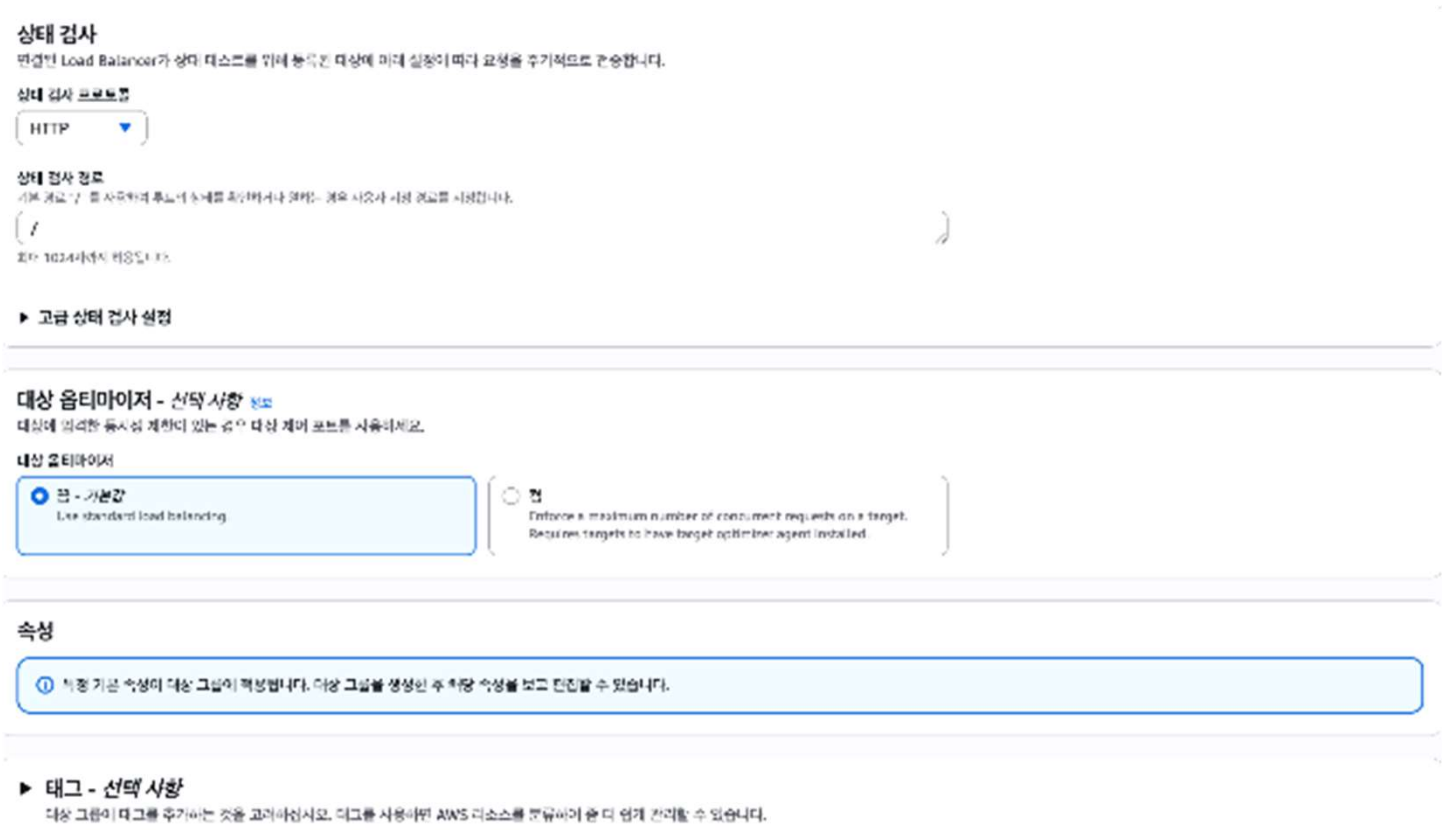
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● 대상 그룹 생성 – 앞에서 만든 인스턴스에 트래픽을 전송하는 대상 그룹 생성
      - ◆ 대상 그룹 설정



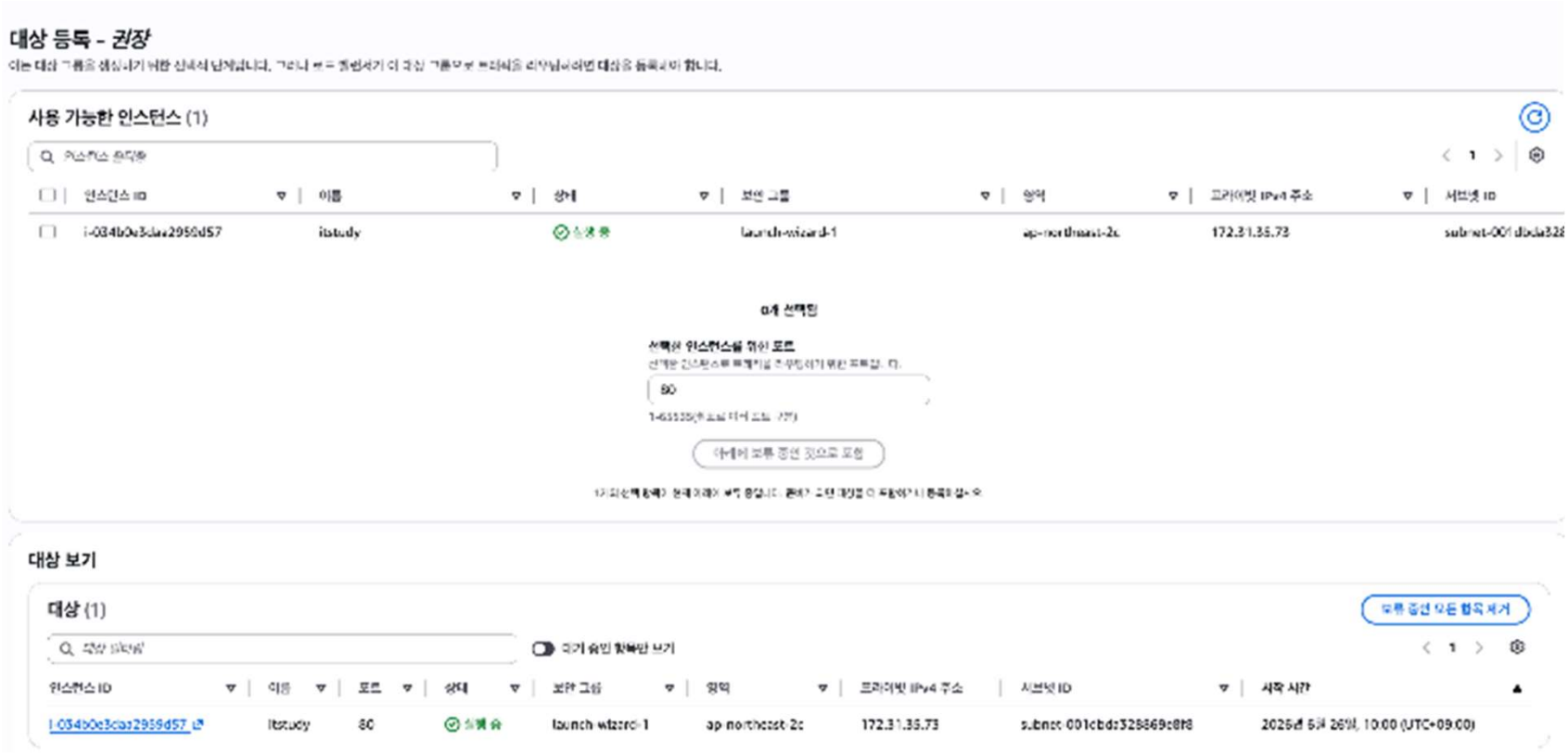
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● 대상 그룹 생성 – 앞에서 만든 인스턴스에 트래픽을 전송하는 대상 그룹 생성
      - ◆ 대상 그룹 설정



애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● 대상 그룹 생성 – 앞에서 만든 인스턴스에 트래픽을 전송하는 대상 그룹 생성
      - ◆ 대상 그룹 설정

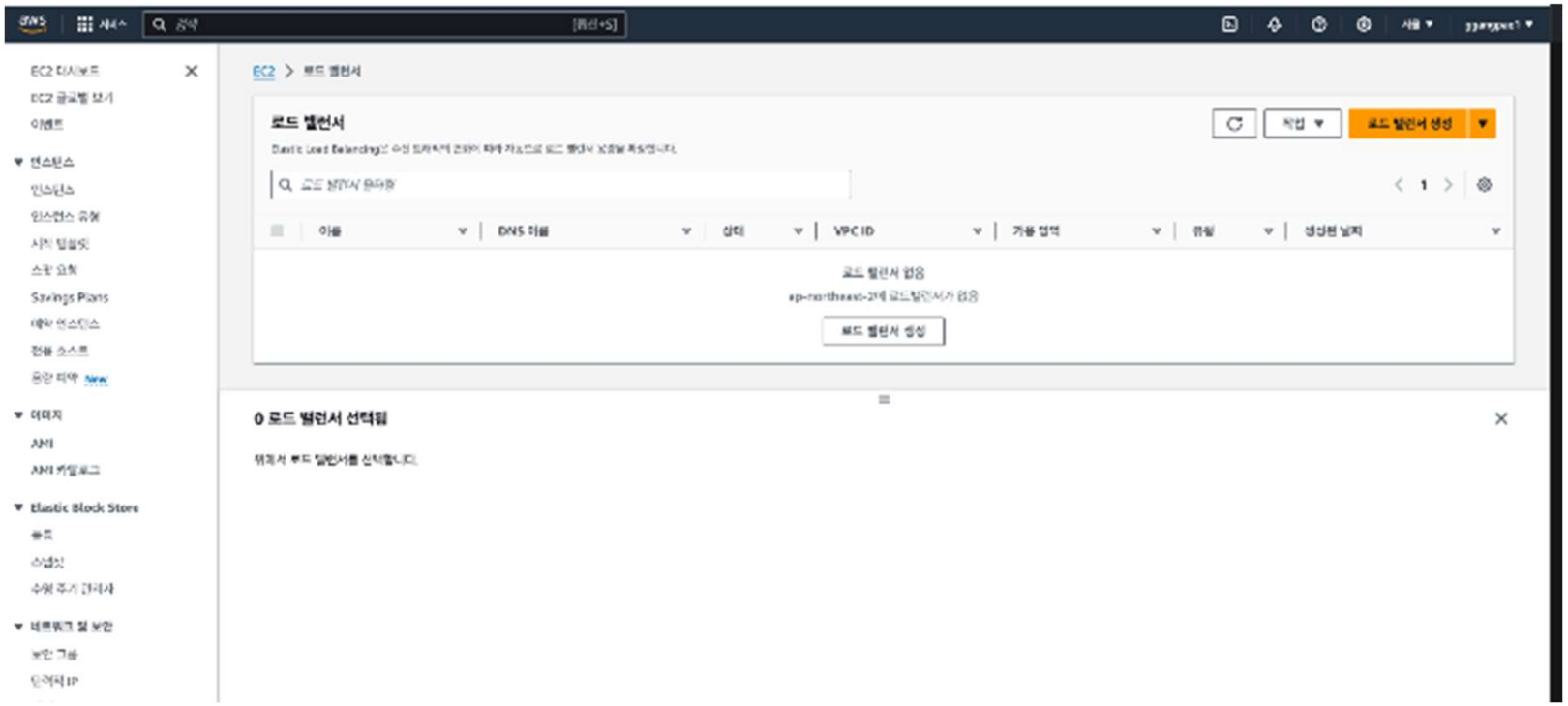


애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● 대상 그룹 생성 – 앞에서 만든 인스턴스에 트래픽을 전송하는 대상 그룹 생성
      - ◆ 대상 그룹 확인

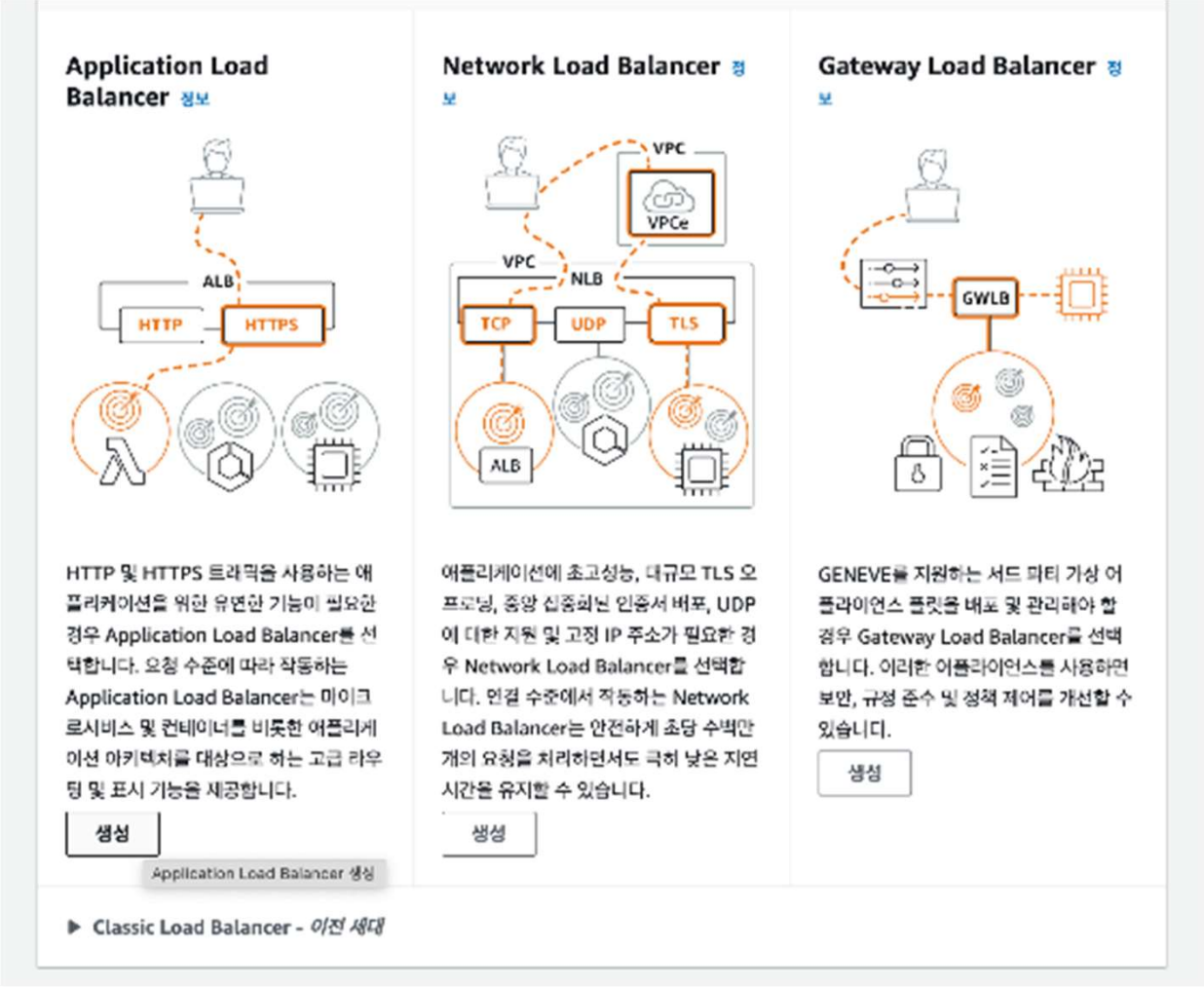
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● EC2 서비스에서 [Load Balancer] 선택한 후 [Load Balancer 생성]을 클릭



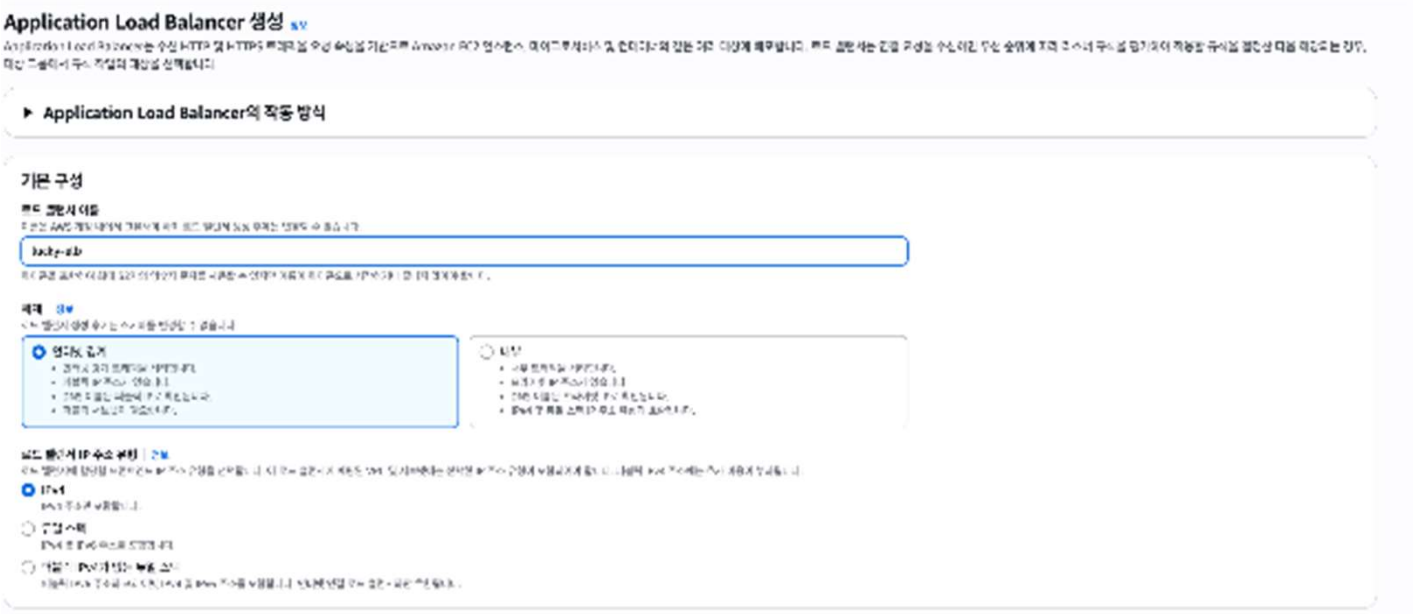
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● Load Balancer 종류 선택 – Application Load Balancer



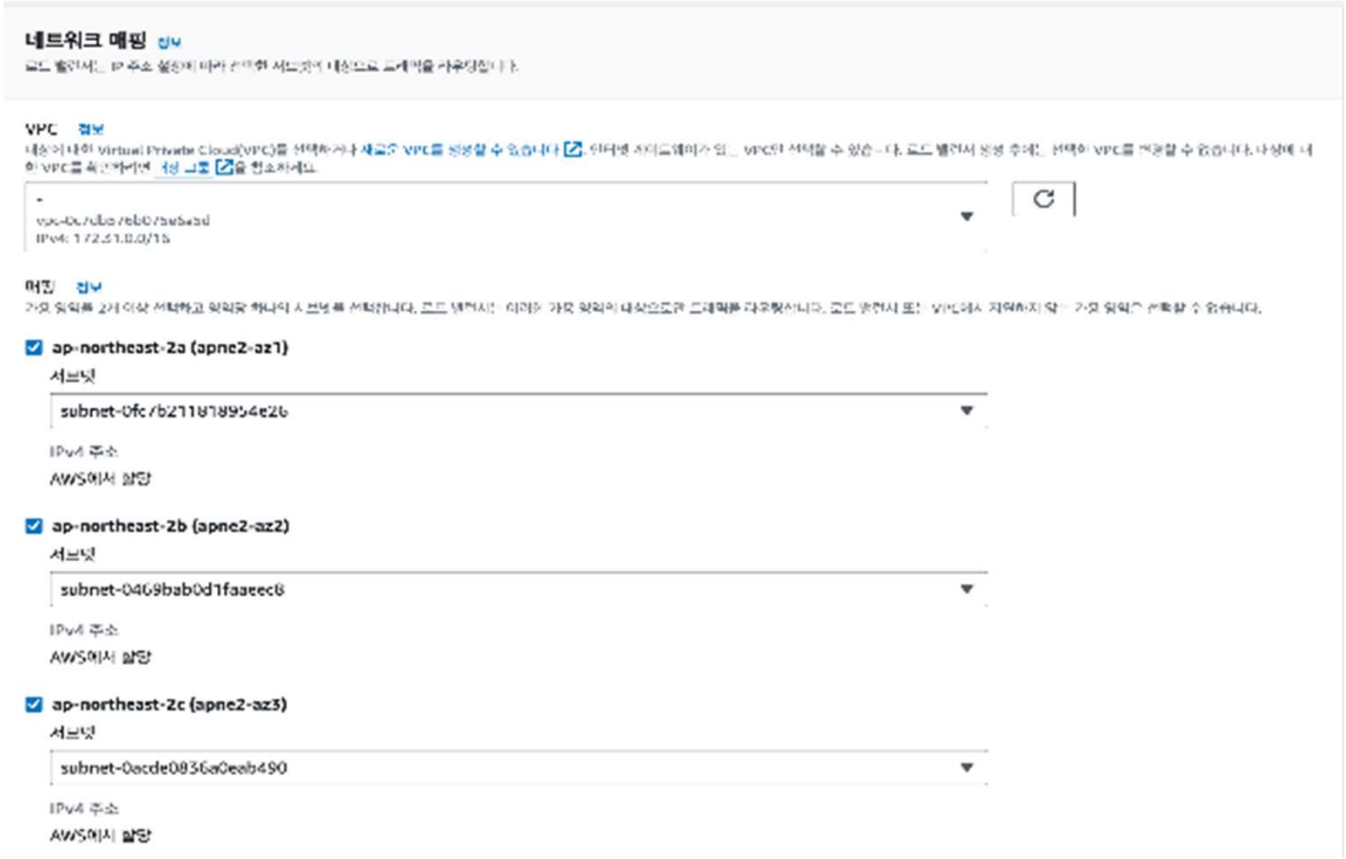
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● Load Balancer 기본 구성 설정
      - ◆ Load Balancer 이름 설정
      - ◆ 체계는 인터넷 경계
      - ◆ IP 주소 유형은 IPv4



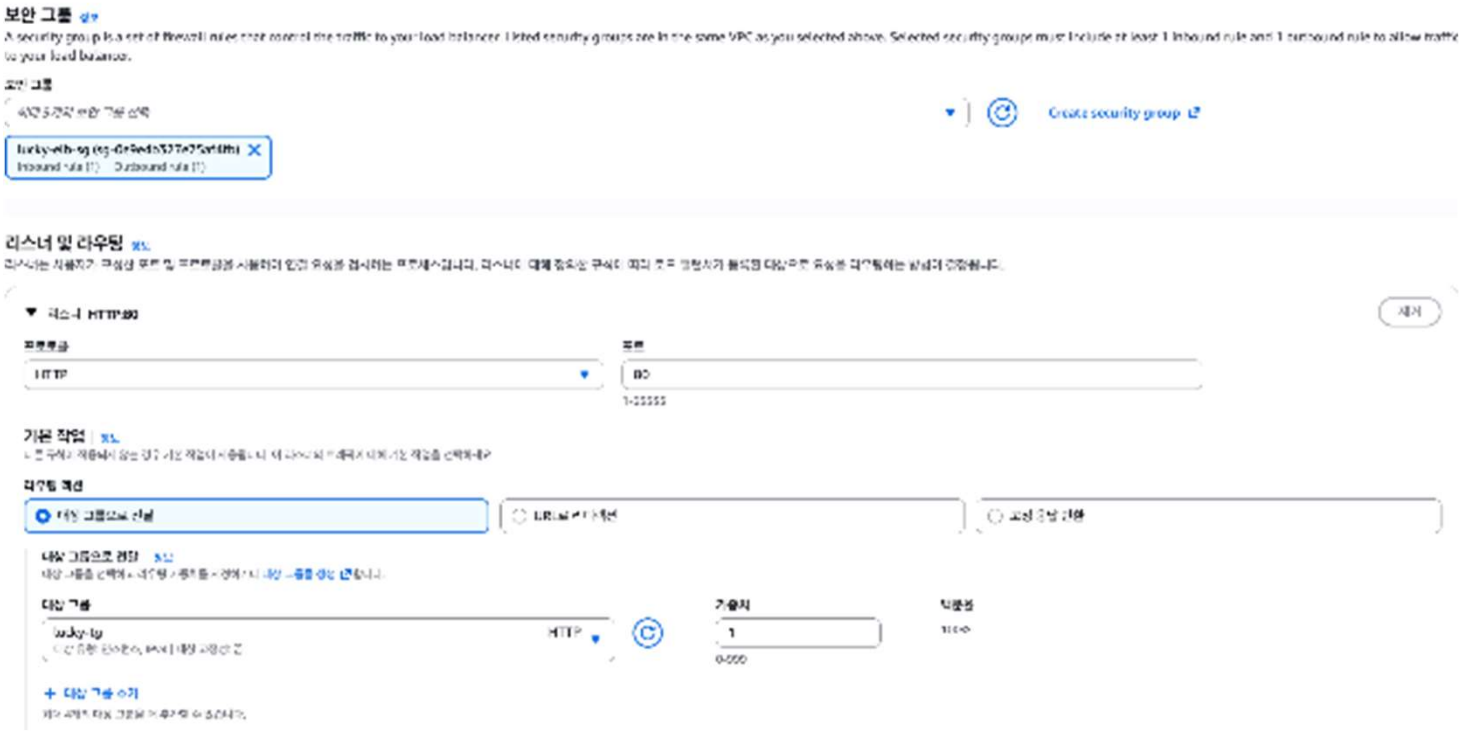
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● Load Balancer 기본 구성 설정
      - ◆ VPC 선택
      - ◆ 가용 영역 선택 – 모든 가용 영역을 선택



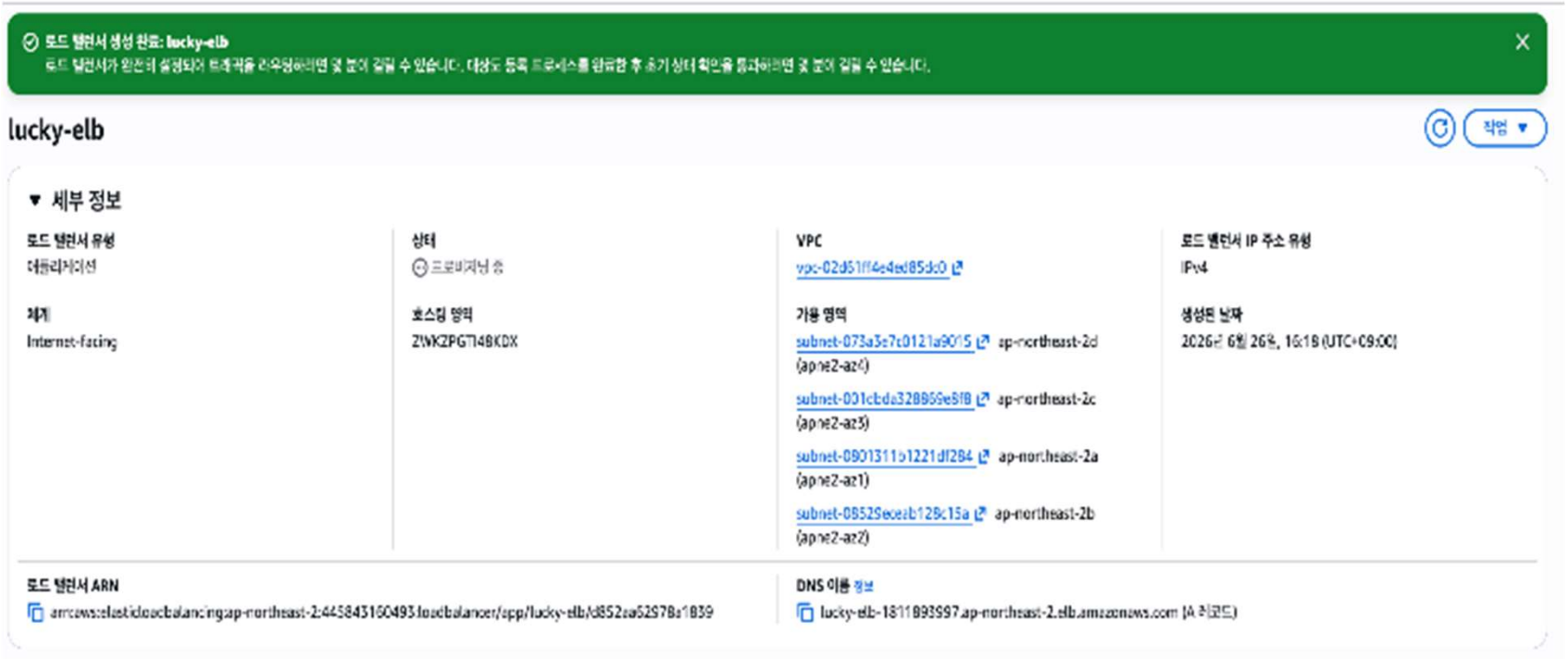
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● Load Balancer 기본 구성 설정
      - ◆ 보안 그룹 선택
      - ◆ 대상 그룹 선택



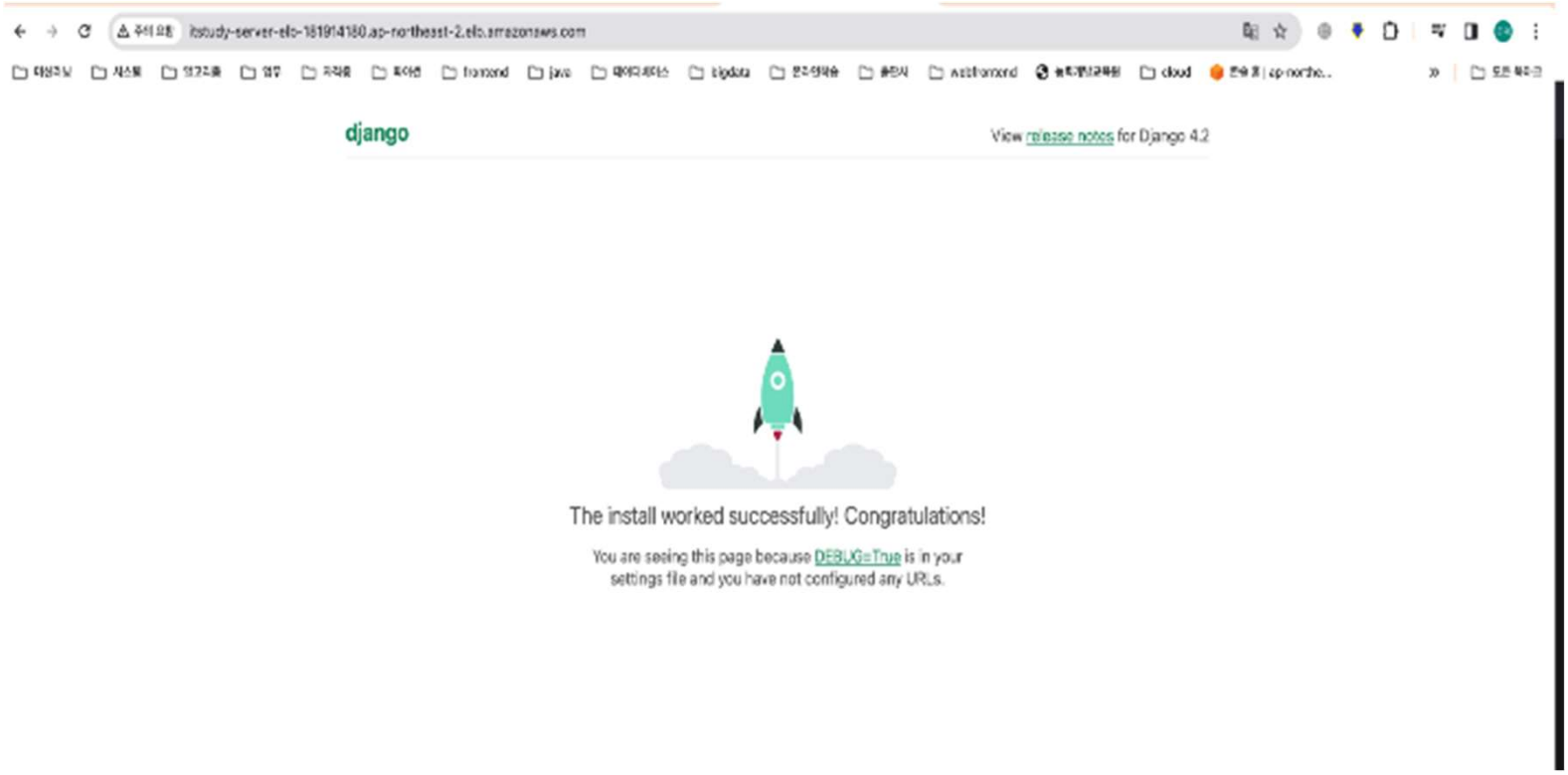
애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● Load Balancer 생성 확인



애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● Load Balancer 생성 확인 - 브라우저에서 Load Balancer DNS로 접속



애플리케이션 배포

- ❖ Load Balancing
  - ✓ Application Load Balancer 적용
    - ● 503 Error가 발생하면 대상 그룹을 다시 추가

애플리케이션 배포

- ❖ 고가용성 아키텍쳐 설계
  - ✓ 고가용성
    - High Availability(고가용성, 줄여서 HA)란 시스템이나 서비스가 전원 장애, 하드웨어 고장, 네트워크 문제, 자연재해 등 어떤 장애 상황에서도 중단되지 않고 지속해서 정상 운영될 수 있는 능력을 의미
    - 장애가 발생해도 성능이 떨어지면 안됨
    - 24시간 365일 언제나 문제없이 안정적으로 서비스를 켜두는 것이 고가용성의 목표
    - AWS에서 ELB는 직접 관리할 필요가 없는 Managed Service라서 트래픽이 많으면 ELB를 알아서 늘리고 트래픽이 적으면 알아서 줄임
    - AWS에서 고가용성을 구현하기 위해서는 다중 가용 영역 과 Auto Scaling을 활용
    - ELB를 만들 때 최소 2개의 가용 영역을 선택하는 이유가 고가용성 아키텍쳐를 만들기 위해서
  - ✓ 가용성 등급
    - 99%: 일반적인 블로그나 내부 테스트 환경
    - 99.9%: 일반적인 엔터프라이즈 서비스의 기본 수준
    - 99.99%: 고가용성을 달성했다고 보는 표준적인 수준
    - 99.999%: 금융권, 의료 시스템, 클라우드 핵심 인프라 수준

애플리케이션 배포

- ❖ 고가용성 아키텍쳐 설계
  - ✓ 고가용성을 달성하기 위한 3대 핵심 원칙
    - 단일 장애점 제거(No Single Point of Failure, SPOF): 시스템 구성 요소 중 이거 하나 고장 나면 전체 시스템이 마비된다고 하는 취약점(SPOF)을 완전히 없애야 하는데 웹 서버가 딱 1대만 있다면 그 서버가 SPOF가 되는데 고가용성을 위해서는 모든 구성 요소를 이중화해야 함
    - 이중화 및 다중화(Redundancy)
      - ◆ 서버, 데이터베이스, 네트워크 장비, 전원 장치 등을 2대 이상으로 복제하여 배치
      - ◆ Active-Active: 여러 대의 서버가 평소에 동시에 일(부하 분산)을 하다가, 하나가 죽으면 남은 서버들이 처리를 이어받는 방식
      - ◆ Active-Standby: 1대만 일하고(Active), 다른 1대는 대기(Standby)하고 있다가 메인 서버가 죽으면 대기 서버가 즉시 투입되는 방식
    - 신속한 장애 복구 메커니즘(Failover): 장애가 발생했을 때 사람이 수동으로 개입해서 복구하면 시간이 오래 걸리므로 모니터링 시스템이 장애를 감지하는 즉시 트래픽을 정상적인 예비 장비로 자동으로 넘겨주는 페일오버(Failover) 시스템이 자동으로 동작해야 함(이 역할을 해주는 대표적인 AWS 서비스가 로드 밸런서와 대상 그룹의 헬스 체크)

애플리케이션 배포

- ❖ 내결함성 아키텍쳐 설계
  - ✓ 내결함성
    - 내결함성(Fault Tolerance)이란 시스템 내의 일부 하드웨어나 소프트웨어에 결함(Fault)이나 오류가 발생하더라도 시스템 전체가 멈추지 않고 전반적인 기능과 서비스를 정상적으로 계속 유지하는 능력
    - 장애 허용 시스템 또는 장애 감내 시스템이라고도 함
    - 고가용성(HA)이 서비스가 중단되지 않는 결과에 초점을 맞춘다면 내결함성은 일부 부품이 깨져도 시스템 스스로 그 충격을 흡수하여 정상 작동을 유지하는 내부적인 복원력을 의미

애플리케이션 배포

- ❖ 내결함성 아키텍쳐 설계
  - ✓ 내결함성을 구현하는 핵심 기술
    - ● 하드웨어 레벨의 예비 부품(Redundancy)
      - ◆ RAID (디스크 이중화): 여러 개의 하드디스크에 데이터를 나누어 저장하여 디스크 하나가 물리적으로 완전히 망가져도 데이터 손실 없이 다른 디스크를 통해 실시간으로 서비스를 계속할 수 있게 함
      - ◆ 이중 전원 공급 장치(Dual Power Supply): 서버 장비에 전원선을 꽂는 곳을 2개 만들어 한쪽 멀티탭이 꺼지거나 파워 서플라이가 고장 나도 반대편 전력을 끌어다 쓰며 서버가 꺼지지 않게 만듬
    - ● 소프트웨어 및 네트워크 레벨의 감지
      - ◆ 복제(Replication): 데이터베이스(DB)의 데이터를 실시간으로 동기화된 예비 DB에 계속 복사해 두었다가 메인 DB가 뻗으면 예비 DB가 1초의 공백도 없이 바통을 이어받음
      - ◆ 에러 핸들링과 서킷 브레이커(Circuit Breaker): 소프트웨어 내부에서 특정 기능에 에러가 반복되면 전체 시스템이 멈추지 않도록 해당 기능으로 가는 길을 잠시 차단하고 현재 점검 중 같은 안내를 띄우며 나머지 기능(결제, 로그인 등)은 정상 작동하게 격리

애플리케이션 배포

❖ 연습문제

✓ 시스템을 운영하는데 t3.large 인스턴스가 6개가 꼭 필요한 시스템이 있다고 가정하고 다음 중 이 시스템의 내결함성을 만족하는 것을 모두 고르시오?

① 2개의 가용영역에 각 3대의 인스턴스를 배치

② 3개의 가용영역에 각 2대의 인스턴스를 배치

③ 2개의 가용영역에 각 6대의 인스턴스를 배치

④ 3개의 가용영역에 각 3대의 인스턴스를 배치

⑤ 1개의 가용영역에 각 6대의 인스턴스를 배치

애플리케이션 배포

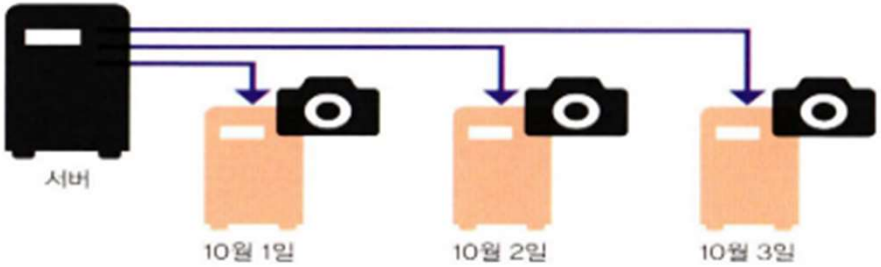
- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling Group
    - AWS Auto Scaling Group(ASG)은 미리 정의한 조건(조건식, 정책)에 따라 EC2 인스턴스의 개수를 자동으로 늘리거나(Scale-out) 줄여주는(Scale-in) 인스턴스들의 관리 그룹
    - Auto Scaling Group이 필요한 이유
      - ◆ 비용 절감: 새벽 시간대처럼 사용자가 적을 때는 최소한의 서버만 켜두고 낮 시간대에만 서버를 늘려 딱 쓴 만큼만 비용을 지불할 수 있음
      - ◆ 내결함성 및 고가용성 보장: 서버 1대가 갑자기 고장(Crash) 나서 죽더라도 ASG가 이를 즉시 감지하고 정상적인 새 서버를 자동으로 똑같이 만들어내어 서비스를 유지
      - ◆ 예측 불가능한 트래픽 대응: 갑자기 마케팅 대박이 나거나 바이럴이 되어 접속자가 폭주해도 개발자가 밤새 모니터링하며 수동으로 서버를 켤 필요가 없음
    - Auto Scaling Group의 3가지 핵심 설정 값(크기 정의)
      - ◆ 최소 크기(Minimum Capacity): 아무리 트래픽이 없어도 최소한 이 개수만큼은 서버를 유지하라는 기준
      - ◆ 최대 크기(Maximum Capacity): 트래픽이 무한정 몰려도 비용 폭탄을 막기 위해 최대 이 개수까지만 늘리라는 마지노선
      - ◆ 원하는 용량 (Desired Capacity): 평상시에 유지하고 싶은 기본 서버 개수로 처음에는 이 개수로 시작하며, 트래픽 변화에 따라 최소와 최대 크기 사이에서 유동적으로 변함

애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling Group
    - ● 확장 정책: ASG가 언제 서버를 늘리고 줄일지 결정하는 규칙(Scaling Policy)
      - ◆ 대상 추적 확장 정책(Target Tracking): 가장 많이 쓰이는 방식으로 특정 목표 수치를 지정하면 ASG가 알아서 맞추는 방식 - 그룹 전체의 평균 CPU 사용량을 60%로 유지해 줘. -> 60%를 넘으면 서버를 늘리고, 낮아지면 줄임
      - ◆ 단계별 확장 정책(Step Scaling): 수치 구간에 따라 단계를 두는 방식 - CPU 사용량이 70%를 넘으면 2대 추가, 90%를 넘으면 4대 추가
      - ◆ 예약된 확장 정책 (Scheduled Scaling): 이벤트 일정을 미리 알고 있을 때 유용 - 매주 월요일 오전 9시 수강신청 시작 직전에 서버를 무조건 5대로 늘리도록 할 수 있음

애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Snapshot
    - 스냅샷이란 어떤 시점의 서버 디스크 상태를 통째로 보존한 파일이나 디렉토리 등의 집합
    - 전체를 보존하기 때문에 데이터나 소프트웨어뿐만 아니라 OS와 설정 정보 등도 모두 포함
    - 소프트웨어 나 OS 갱신 작업 등을 할 때 무언가 문제가 발생하면 바로 되돌릴 수 있도록 백업하기 위해 스냅샷을 사용하는 경우가 많고 개인이 AWS 에서 AMI를 만들기 위해 스냅샷을 사용하기도 함
    - AWS는 Amazon EBS 볼륨의 데이터를 스냅샷으로 보존할 수 있는데 최초에는 전체를 보존하지만 두 번째 이후는 차분(증분)만 보존하는데 이는 스냅샷의 비용이 높아지지 않도록 하기 위한 아마존의 정책
    - 스냅샷을 삭제하면 해당 스냅샷 고유의 데이터만 삭제되므로 최초 데이터를 삭제할 경우 두 번째 스냅샷을 취득했을 때 변경되지 않은 부분(고유 데이터)만 두 번째 스냅샷으로 합쳐진 후 삭제
    - 스냅샷은 특정 시점의 서버 상태



애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Snapshot 생성 방법
    - 스냅샷은 관리 콘솔에서 볼륨 단위(스토리지 전체)로 선택하여 생성
    - 생성한 스냅샷을 기반으로 EBS 볼륨을 생성하면 새로운 볼륨은 원래 볼륨의 복제.
    - AMI를 작성하고 싶을 때도 스냅샷으로 작성하고 데이터 수명 주기 관리자(Amazon DLM)를 사용하면 스냅샷의 생성, 삭제를 자동화할 수 있는데 스냅샷을 정기적으로 생성하면 서버가 망가졌을 때 대응 할 수 있음
    - 스냅샷의 요금은 생성한 스냅샷의 분량 단위(GB 단위)로 발생
    - 스냅샷의 데이터 보존 장소는 Amazon S3이지만 스냅샷의 파일은 사용자가 자유롭게 다운로드 할 수 없는데 사용자가 S3를 사용할 때의 영역(저장소 혹은 저장 장소)과는 다른 영역으로 볼 수 없는 장소에 보존되지만 S3 요금은 발생하지 않음

애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ AMI
    - ● 개요
      - ◆ AMI(Amazon Machine Image)는 AWS EC2 인스턴스(가상 서버)를 생성하기 위한 기본 템플릿(이미지)
      - ◆ 컴퓨터를 포맷한 후 윈도우나 리눅스를 설치하고 자주 쓰는 프로그램을 미리 세팅해 둔 고스트(Ghost) 이미지나 백업 파일과 같은 개념
    - ● AMI의 핵심 구성 요소: AMI는 인스턴스를 부팅하고 운영하는 데 필요한 다음 정보들을 패키징하여 담고 있음
      - ◆ 루트 볼륨 템플릿: 운영 체제(OS: Linux, Windows 등), 애플리케이션 서버, 소프트웨어 및 설정 데이터가 포함된 EBS 스냅샷 또는 인스턴스 스토어 템플릿
      - ◆ 시작 권한: 어떤 AWS 계정이 이 AMI를 사용할 수 있는지 제어하는 권한 설정(개인용, 특정 계정 공유, 전체 공개)
      - ◆ 블록 디바이스 매핑: 인스턴스가 시작될 때 생성되어 연결될 추가 스토리지(EBS 볼륨 등)에 대한 볼륨 매핑 정보

애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ AMI
    - ● 제공 주체별 분류
      - ◆ Quick Start(빠른 시작): AWS가 공식적으로 제공하고 유지 관리하는 안전한 기본 OS 이미지(Amazon Linux, Ubuntu, Windows Server 등)
      - ◆ My AMIs(내 AMI): 사용자가 직접 구축한 EC2 인스턴스의 현재 상태를 복사하여 만든 커스텀 이미지. 동일한 환경의 서버를 복제하거나 백업할 때 사용
      - ◆ AWS Marketplace: 인프라에 필요한 타사 소프트웨어(보안 솔루션, DB 등)가 미리 설치되어 제공되는 유/무료 이미지
      - ◆ Community AMIs(커뮤니티 AMI): 전 세계 AWS 사용자들이 공유 목적으로 등록해 둔 공개 이미지(보안 검증이 완벽하지 않을 수 있으므로 주의 필요).
    - ● 루트 디바이스 유형별 분류
      - ◆ EBS-backed AMI: 인스턴스의 루트 장치가 EBS(Elastic Block Store) 볼륨인 이미지로 인스턴스를 중지(Stop)했다가 재시작할 수 있으며 데이터 보존 및 스냅샷 관리가 용이하여 대부분 이 유형을 사용
      - ◆ Instance Store-backed AMI: 인스턴스의 루트 장치가 물리 서버에 장착된 임시 스토리지(Instance Store)인 이미지로 인스턴스를 중지할 수 없으며 인스턴스가 종료(Terminate)되면 데이터가 영구히 삭제됨

애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ AMI
    - ● AMI의 주요 장점 및 사용 사례
      - ◆ 신속한 인스턴스 복제: 서버를 새로 만들 때마다 OS 설치, 패키지 업데이트, 소스 코드 배포를 매번 반복할 필요 없이 AMI 하나로 수십·수백 대의 동일한 서버를 즉시 실행할 수 있음
      - ◆ Auto Scaling과의 연동: 트래픽이 몰릴 때 자동으로 서버를 늘려주는 Auto Scaling 그룹을 설정할 때 늘어날 서버의 원본 템플릿으로 AMI를 지정
      - ◆ 백업 및 재해 복구(DR): 안정적으로 구동 중인 서버를 AMI로 만들어 두면 나중에 장애가 발생하더라도 해당 시점으로 완벽하게 시스템을 복구할 수 있음
      - ◆ 리전 간 이동: 생성된 AMI는 다른 AWS 리전(Region)으로 복사할 수 있어 글로벌 서비스 배포나 인프라 마이그레이션이 매우 간편해짐

애플리케이션 배포

❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성

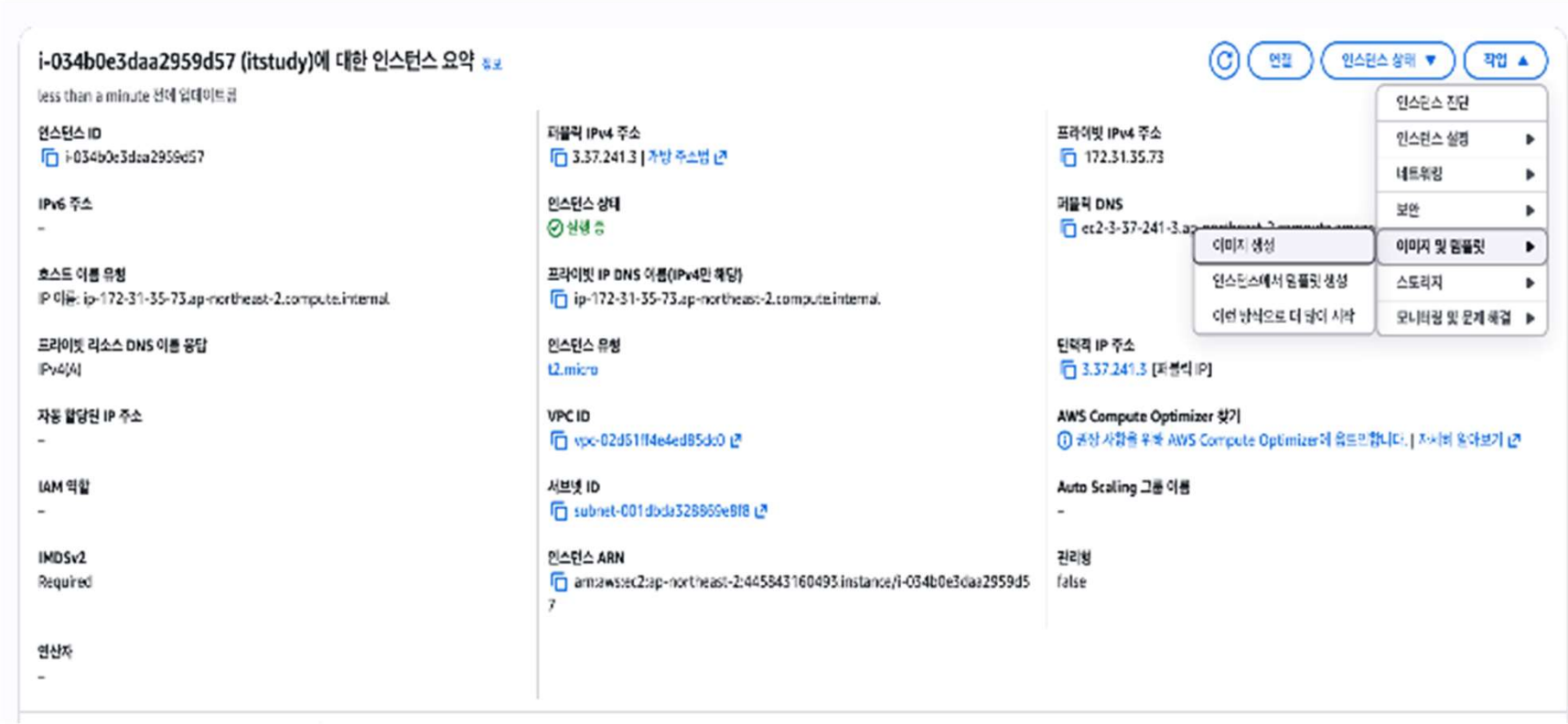
✓ AMI

- SnapShop 과의 차이

구분	스냅샷 (Snapshot)	AMI (Amazon Machine Image)
개념	EBS 볼륨(디스크) 한 장에 대한 특정 시점의 데이터 백업본	컴퓨터(인스턴스) 전체를 부팅하고 구동하기 위한 종합 템플릿
특징	단순히 파일과 데이터만 저장된 상태 (부팅 불가)	OS 정보 및 블록 디바이스 매핑 정보가 포함되어 즉시 부팅 가능
용도	단순 데이터 백업, 볼륨 복구 및 증설	새로운 EC2 가상 서버 인스턴스 생성 및 복제

애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ AMI 생성
    - 생성: 인스턴스를 선택한 후 [작업] – [이미지 및 템플릿 생성] – [이미지 생성]



애플리케이션 배포

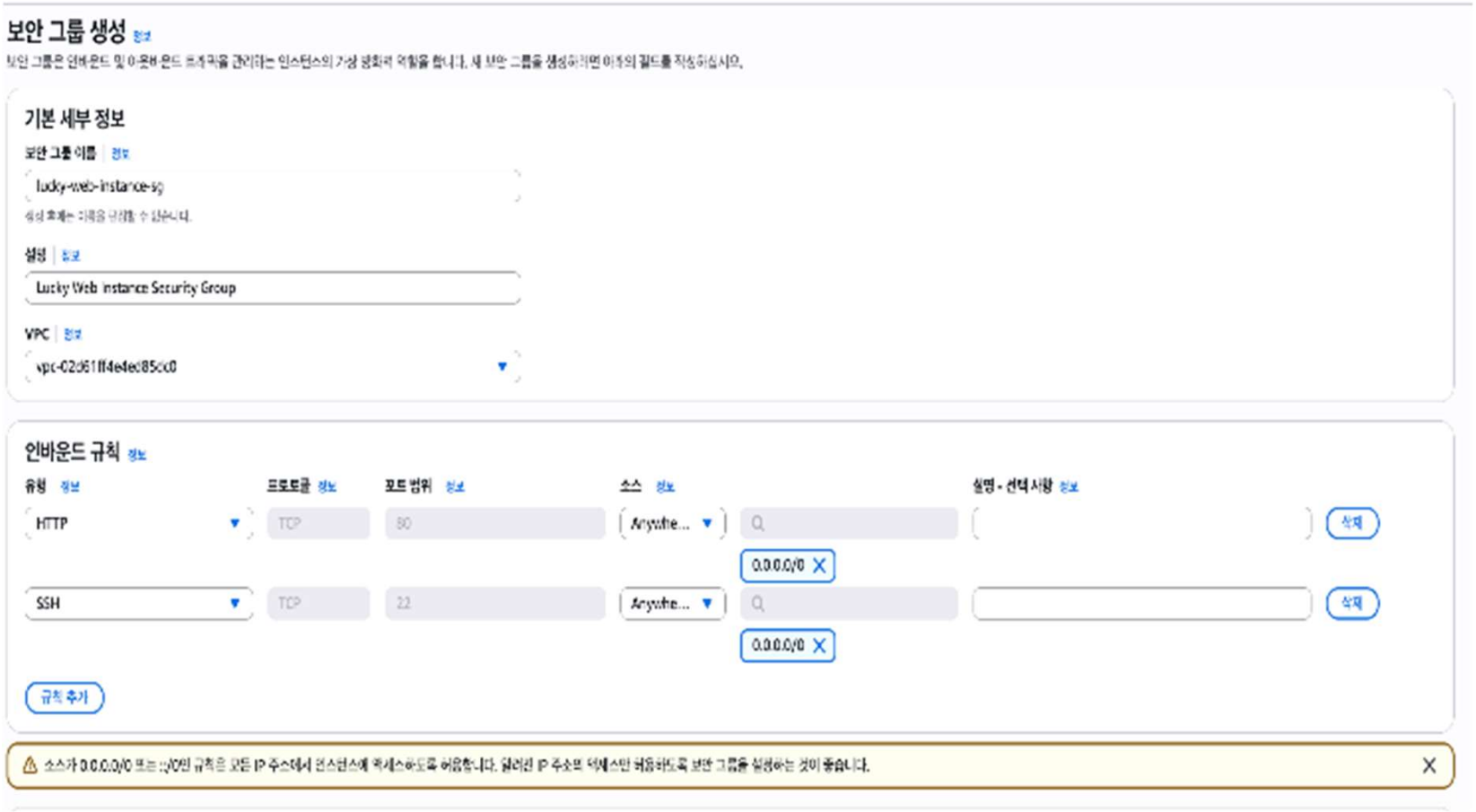
- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ AMI 생성
    - ● 생성: 인스턴스를 선택한 후 [작업] – [이미지 및 템플릿 생성] – [이미지 생성]

애플리케이션 배포

❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성

✓ 보안 그룹 생성

- HTTP 와 SSH에 모든 IP를 받아들이는 그룹을 생성



애플리케이션 배포

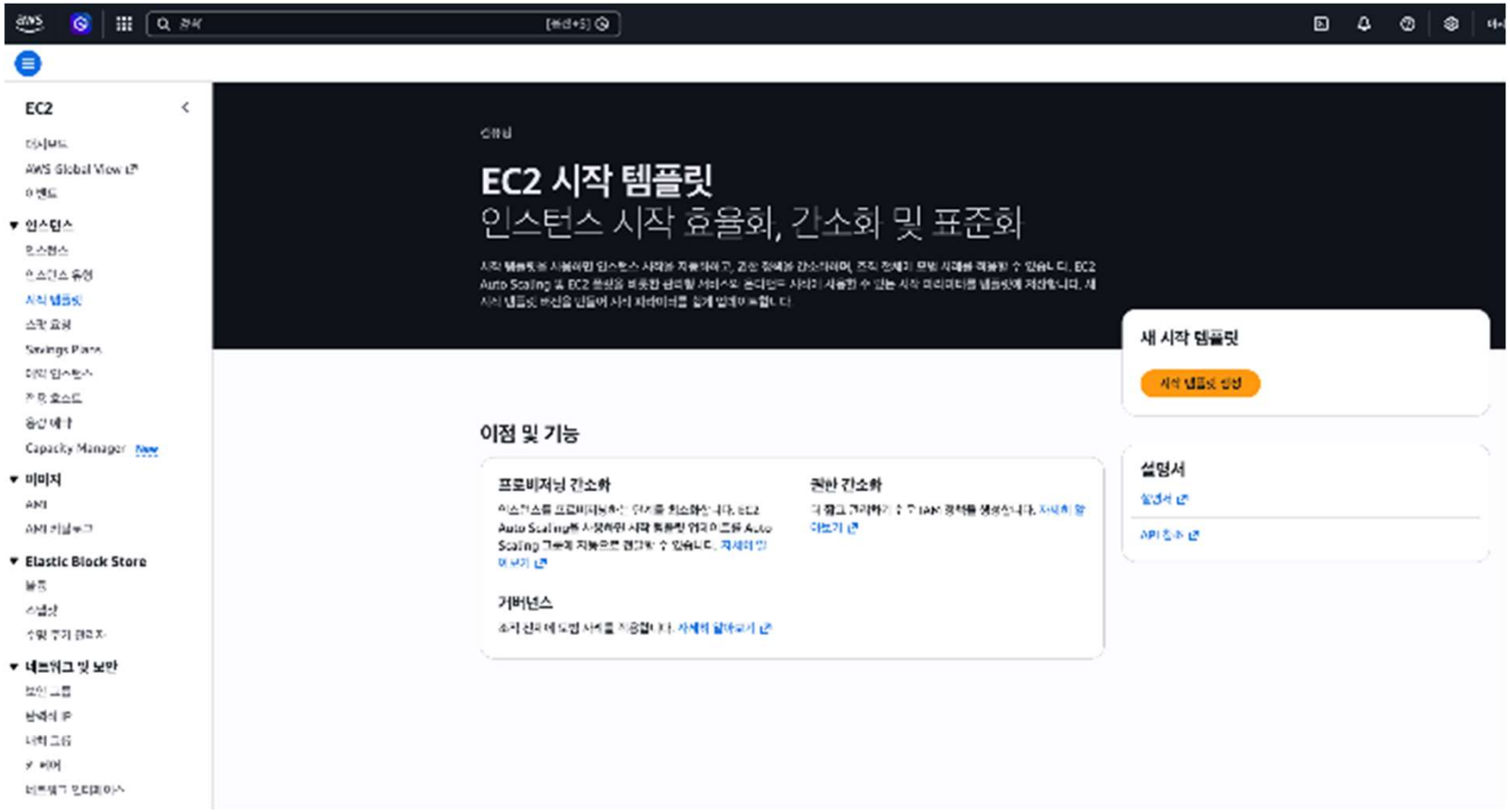
- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ 시작 템플릿
    - ● 개요
      - ◆ EC2 시작 템플릿(Launch Template)은 EC2 인스턴스를 생성할 때 필요한 모든 설정 값을 미리 저장해 두는 주문서 템플릿
      - ◆ AMI가 설치할 OS와 프로그램의 원본 이미지라면 시작 템플릿은 그 AMI를 포함해 어떤 사양의 하드웨어로 어떤 네트워크와 보안 설정을 적용해 서버를 켤지 정의한 마스터 플랜
    - ● 주요 특징과 장점
      - ◆ 버전 관리 (Versioning): 시작 템플릿의 가장 큰 장점으로 설정을 변경해야 할 때 기존 템플릿을 수정하는 것이 아니라 새로운 버전(예: 버전 2, 버전 3)을 생성해서 문제가 생기면 이전 버전으로 빠르게 롤백(Rollback)할 수 있음
      - ◆ 오토 스케일링(Auto Scaling)과의 연동: 서버 트래픽에 따라 인스턴스를 자동으로 늘리고 줄여주는 Auto Scaling Group(ASG)을 만들 때 어떤 스펙의 인스턴스를 늘릴 것인가? 에 대한 기준으로 이 시작 템플릿을 사용
      - ◆ 재사용 및 표준화: 팀 내에서 보안 기준이나 필수 소프트웨어가 설치된 가이드라인을 템플릿으로 고정해 두면 누구나 실수 없이 사내 표준에 맞는 인스턴스를 생성할 수 있음

애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ 시작 템플릿
    - ● 템플릿에 저장되는 주요 정보
      - ◆ AMI(Amazon Machine Image): 운영체제(OS) 및 기본 설치 파일 정보 (예: Ubuntu 22.04, Amazon Linux 2023 등)
      - ◆ 인스턴스 유형: CPU 및 메모리 스펙 (예: t3.medium, c6i.large 등)
      - ◆ 네트워크 설정: VPC, 서브넷, 그리고 방화벽 역할을 하는 보안 그룹(Security Group)
      - ◆ 스토리지: EBS 볼륨의 크기와 종류(gp3 등)
      - ◆ 사용자 데이터(User Data): 인스턴스가 처음 켜질 때 자동으로 실행될 쉘 스크립트 (패키지 업데이트나 Docker 설치 등 자동화 세팅)
    - ● 시작 구성(Launch Configuration)과의 차이점
      - ◆ 이전에는 시작 구성(Launch Configuration)이라는 기능도 자주 쓰였지만 AWS에서는 현재 시작 템플릿 사용을 강력히 권장
      - ◆ 시작 구성은 버전 관리가 불가능하고 최신 최신 EC2 기능(T3 무제한, 스팟 인스턴스 혼합 등) 제한되며 Auto Scaling 전용으로만 사용 가능했지만 단일 EC2 생성 시에도 사용 가능

애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ 시작 템플릿 생성
    - EC2 -> 시작 템플릿 -> 시작 템플릿 생성

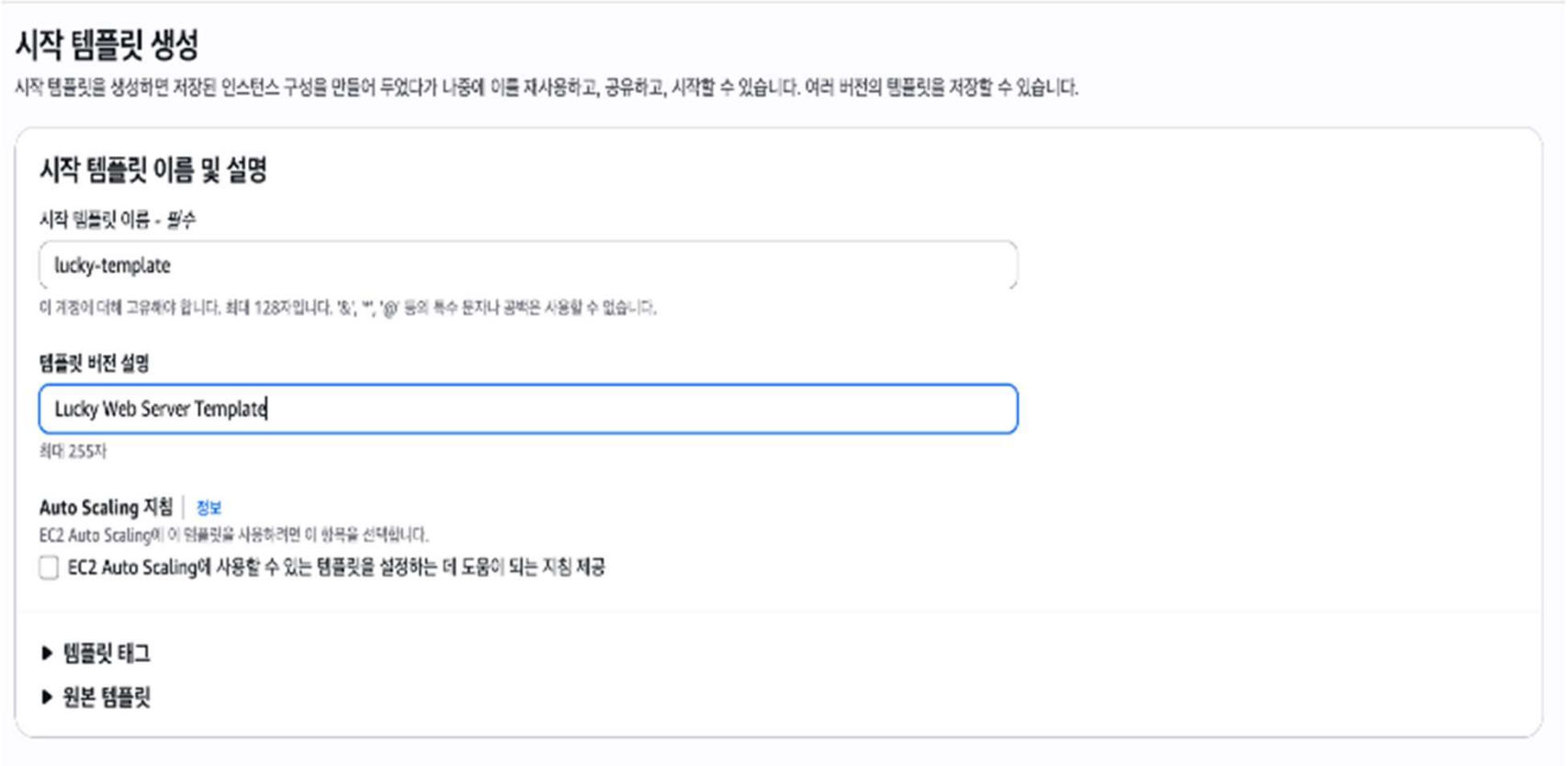


애플리케이션 배포

❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성

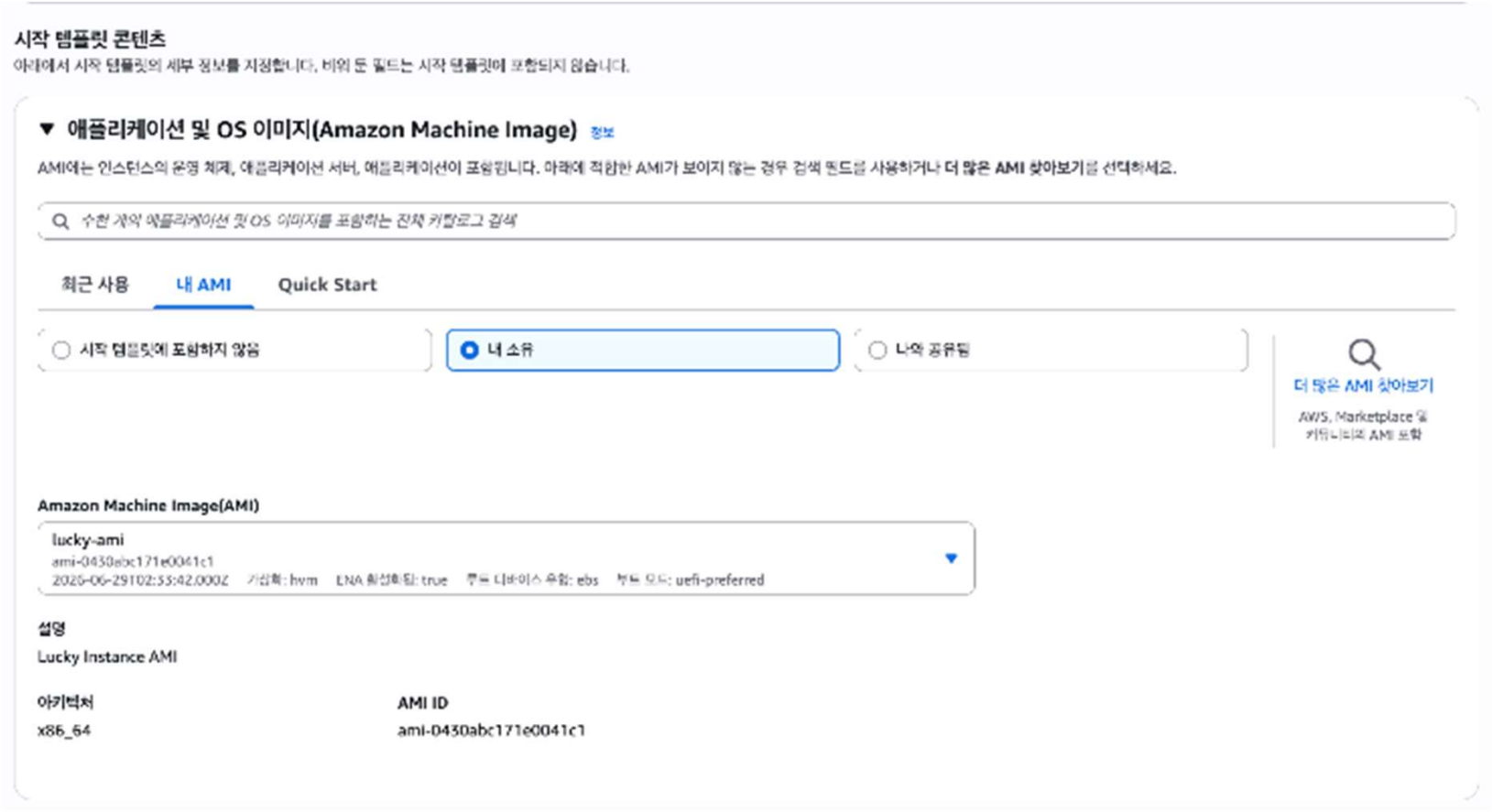
✓ 시작 템플릿 생성

● 설정



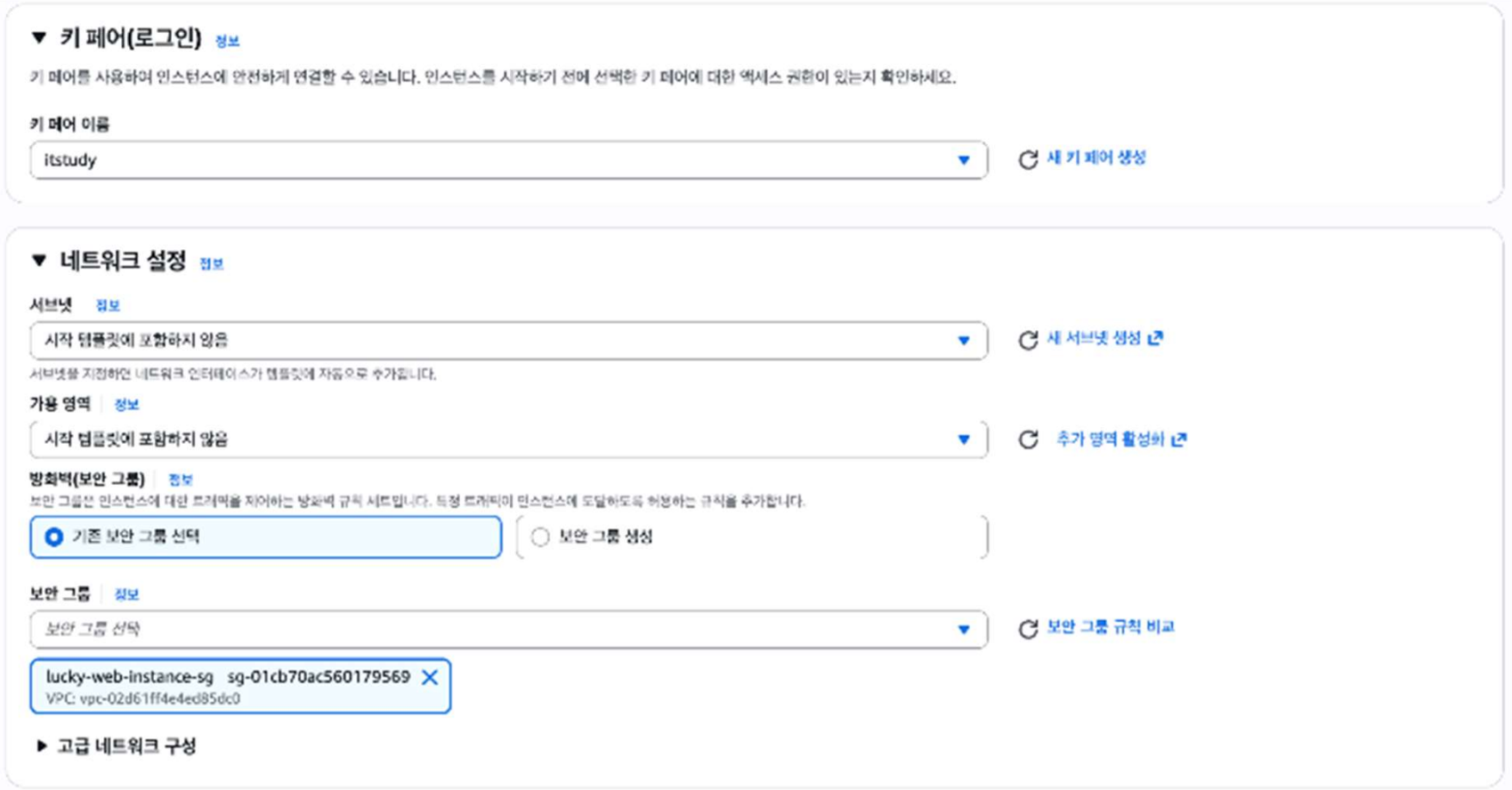
애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ 시작 템플릿 생성
    - ● 설정



애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ 시작 템플릿 생성
    - ● 설정

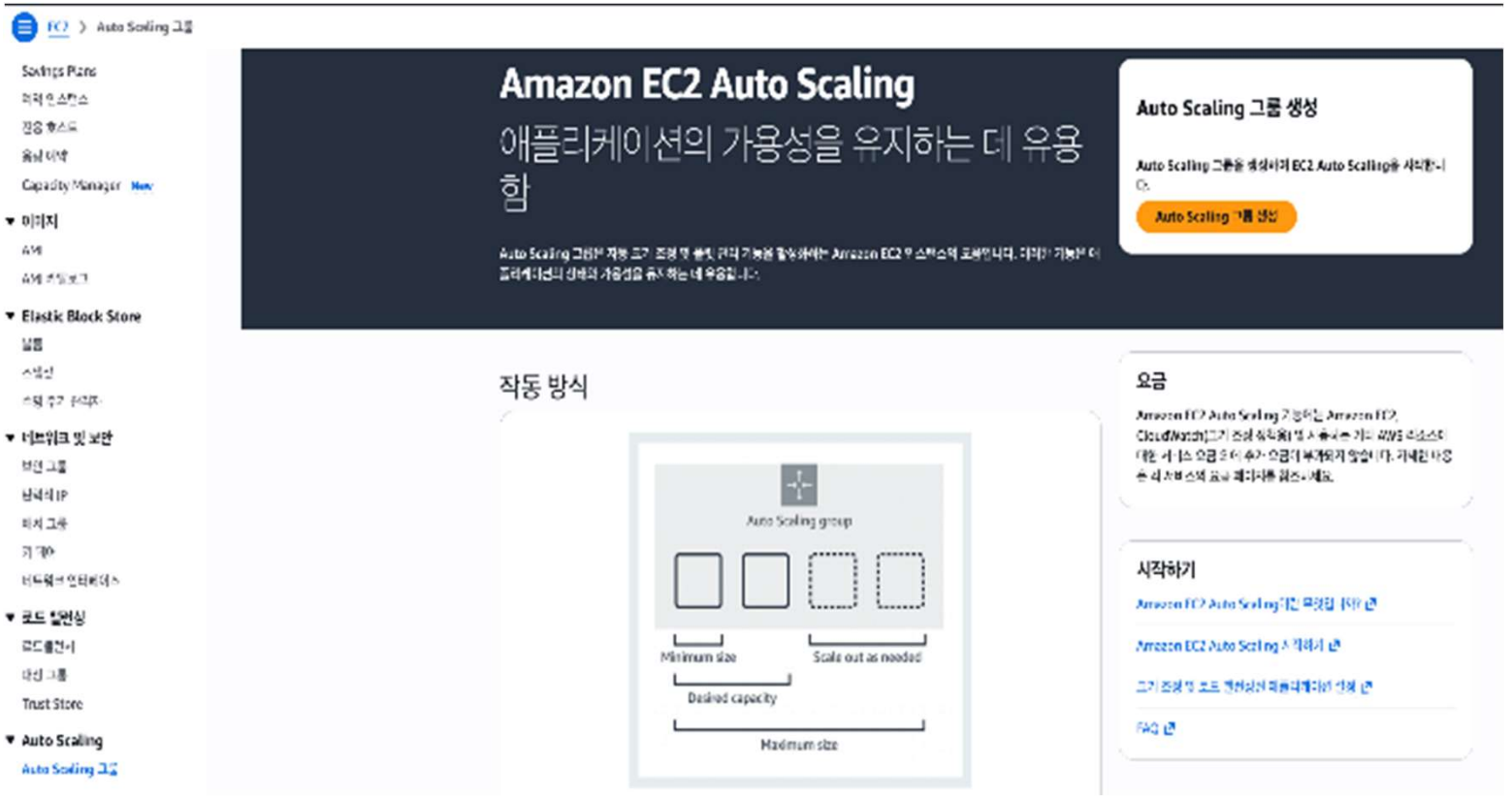


애플리케이션 배포

❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성

✓ Auto Scaling 그룹 생성

- Auto Scaling 그룹 생성 버튼을 클릭

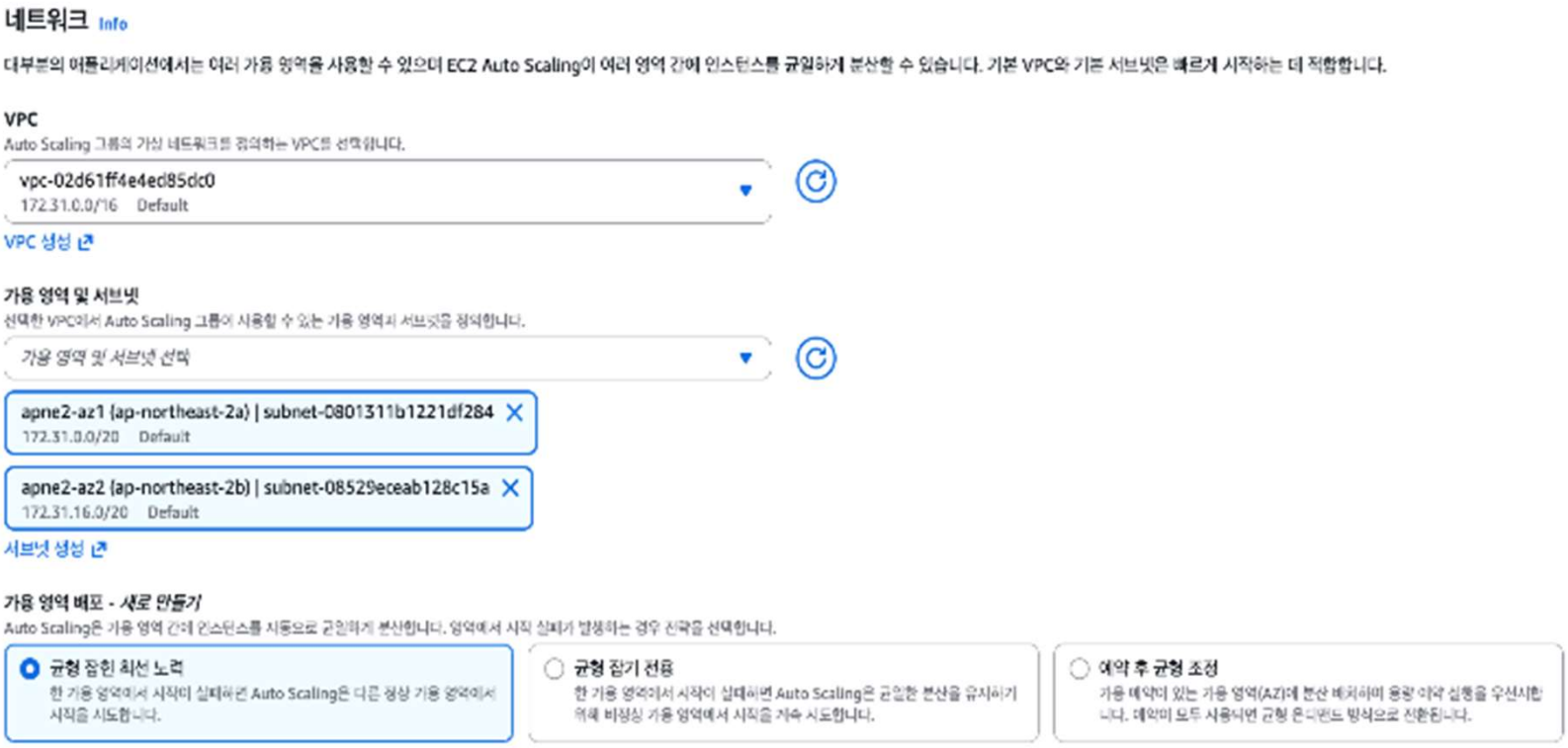


애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 생성
    - ● 시작 템플릿 설정

애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 생성
    - ● 인스턴스 시작 옵션 설정

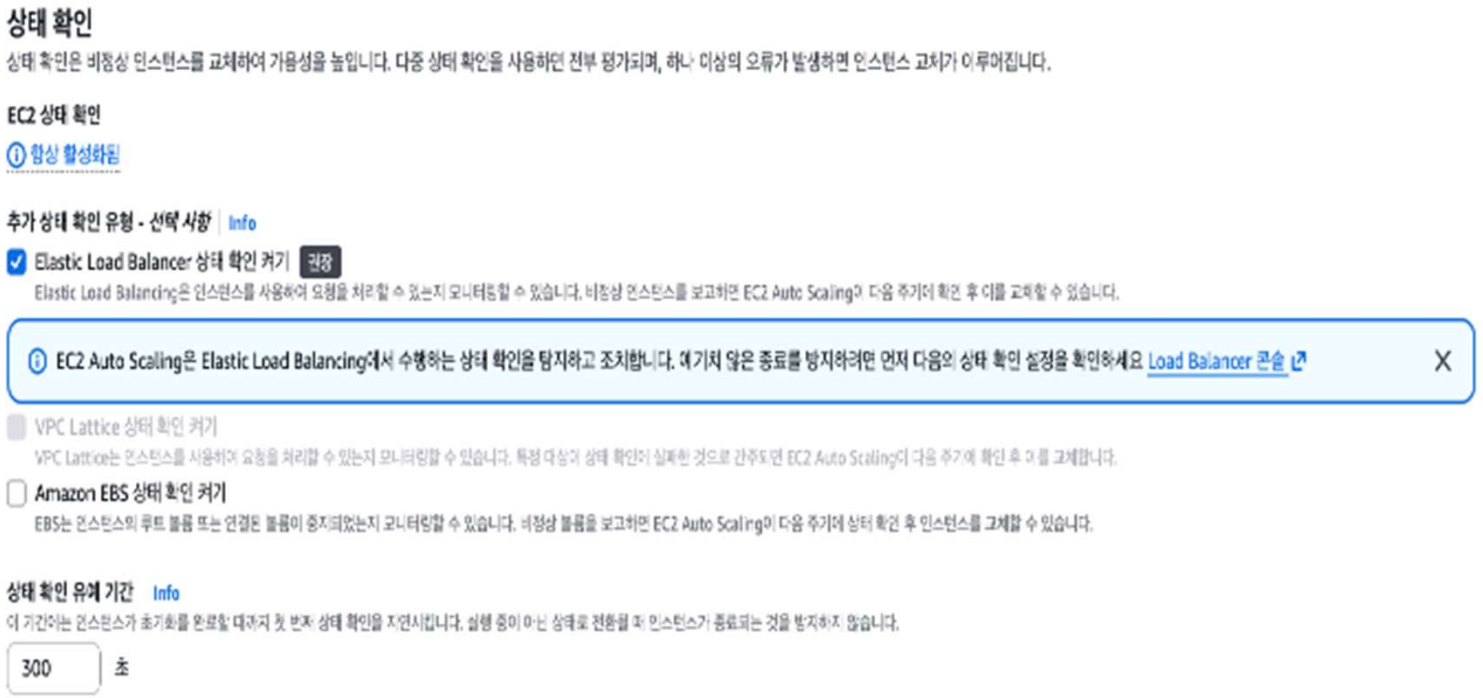


애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 생성
    - ● 다른 서비스와 통합 – 기존 로드 밸런서 연결

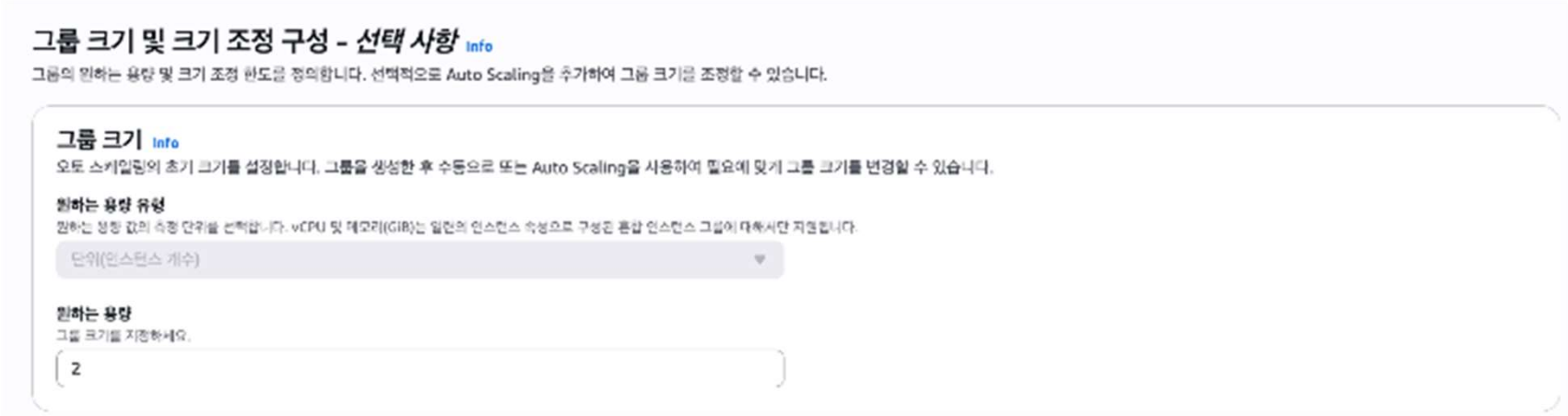
애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 생성
    - ● 다른 서비스와 통합 – 로드 밸런서가 인스턴스의 상태를 확인하고 문제가 있다면 오토 스케일링이 인스턴스를 교체하는 옵션



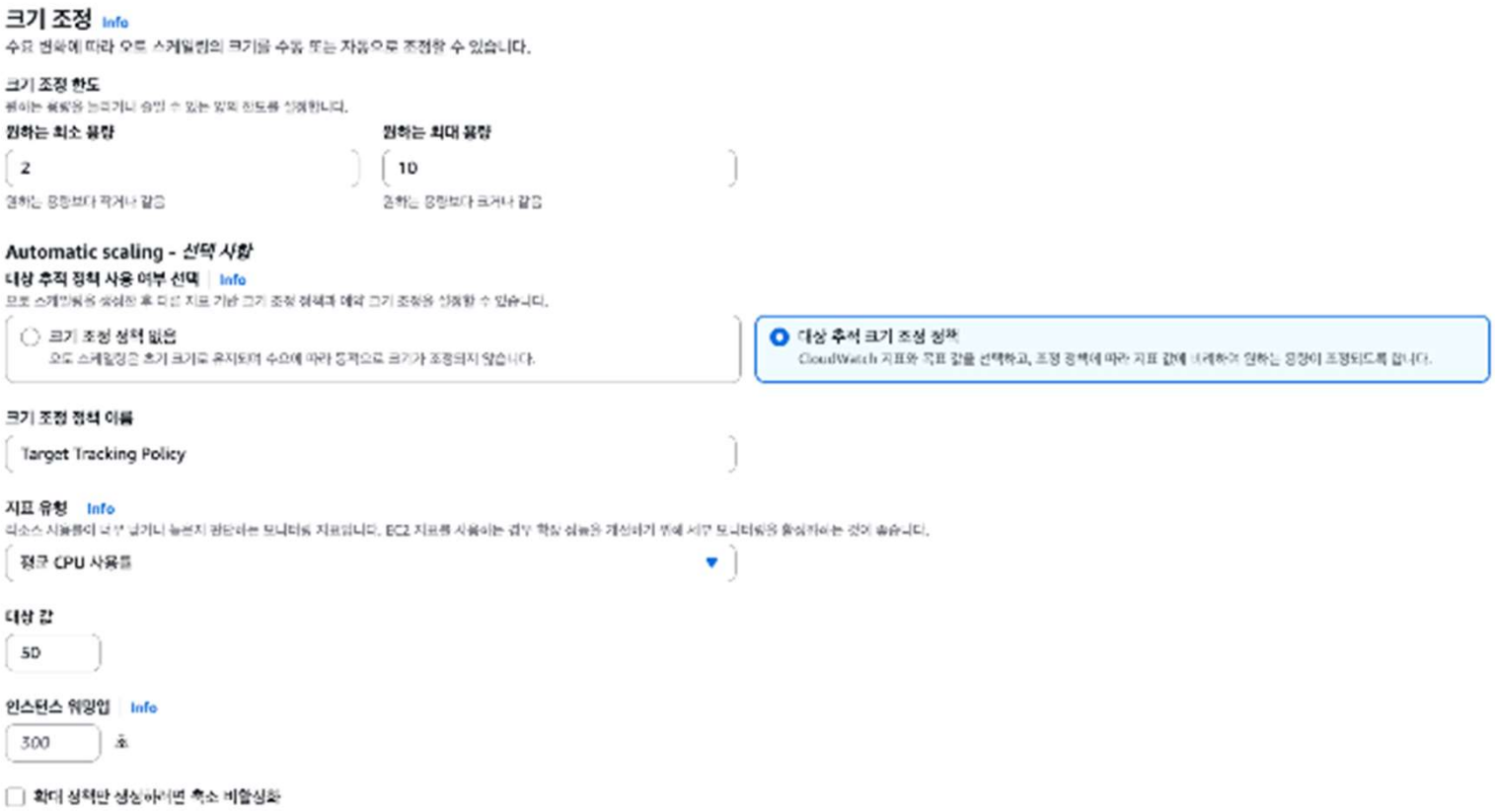
애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 생성
    - ● 그룹 크기 조정



애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 생성
    - ● 그룹 크기 조정

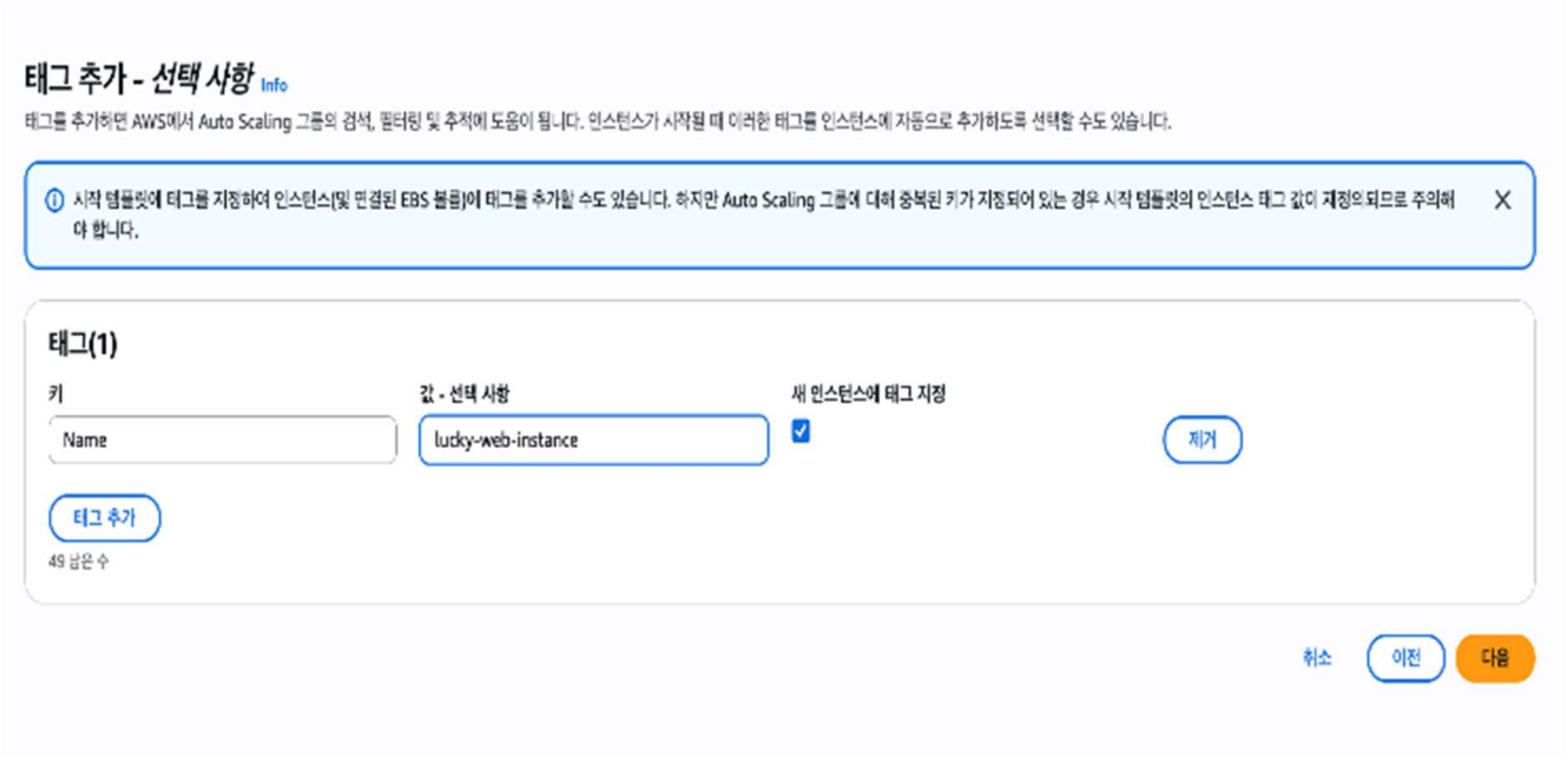


애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 생성
    - ● 알림 추가 설정

애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 생성
    - ● 태그 추가 – 오토 스케일링에 의해 인스턴스가 자동으로 생성될 때 인스턴스 이름에 추가될 내용

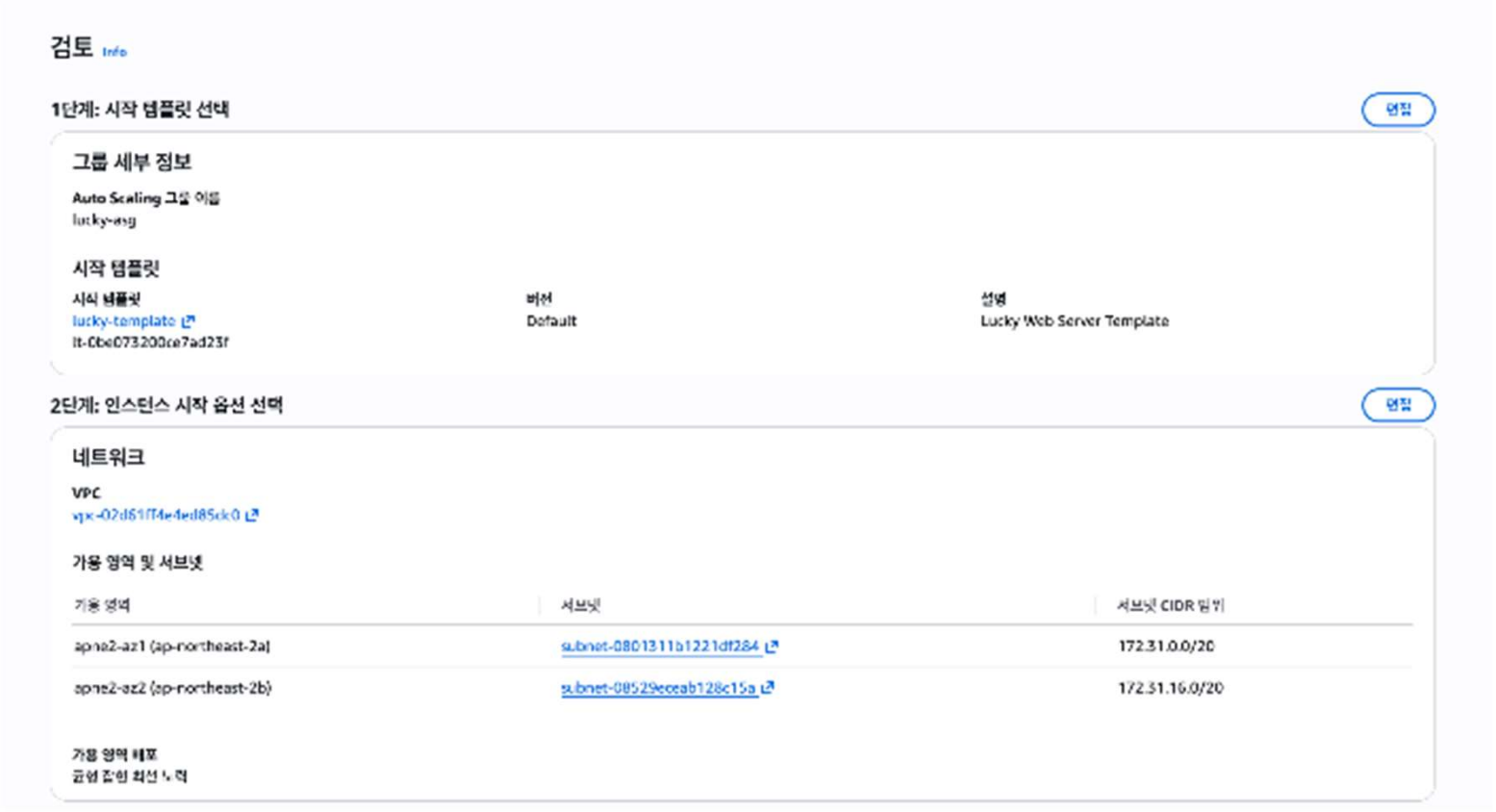


애플리케이션 배포

❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성

✓ Auto Scaling 그룹 생성

● 검토

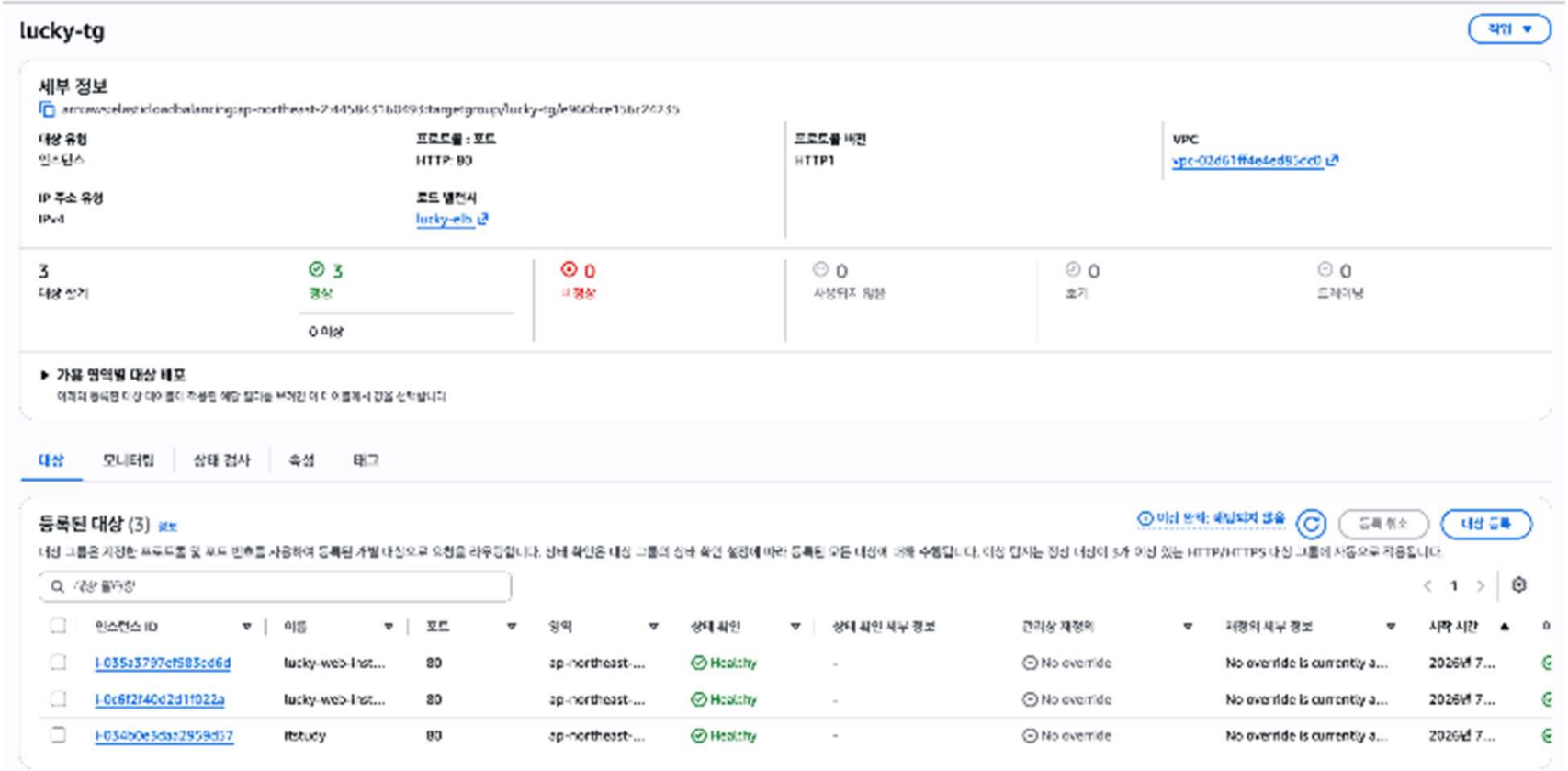


애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 생성
    - ● 확인

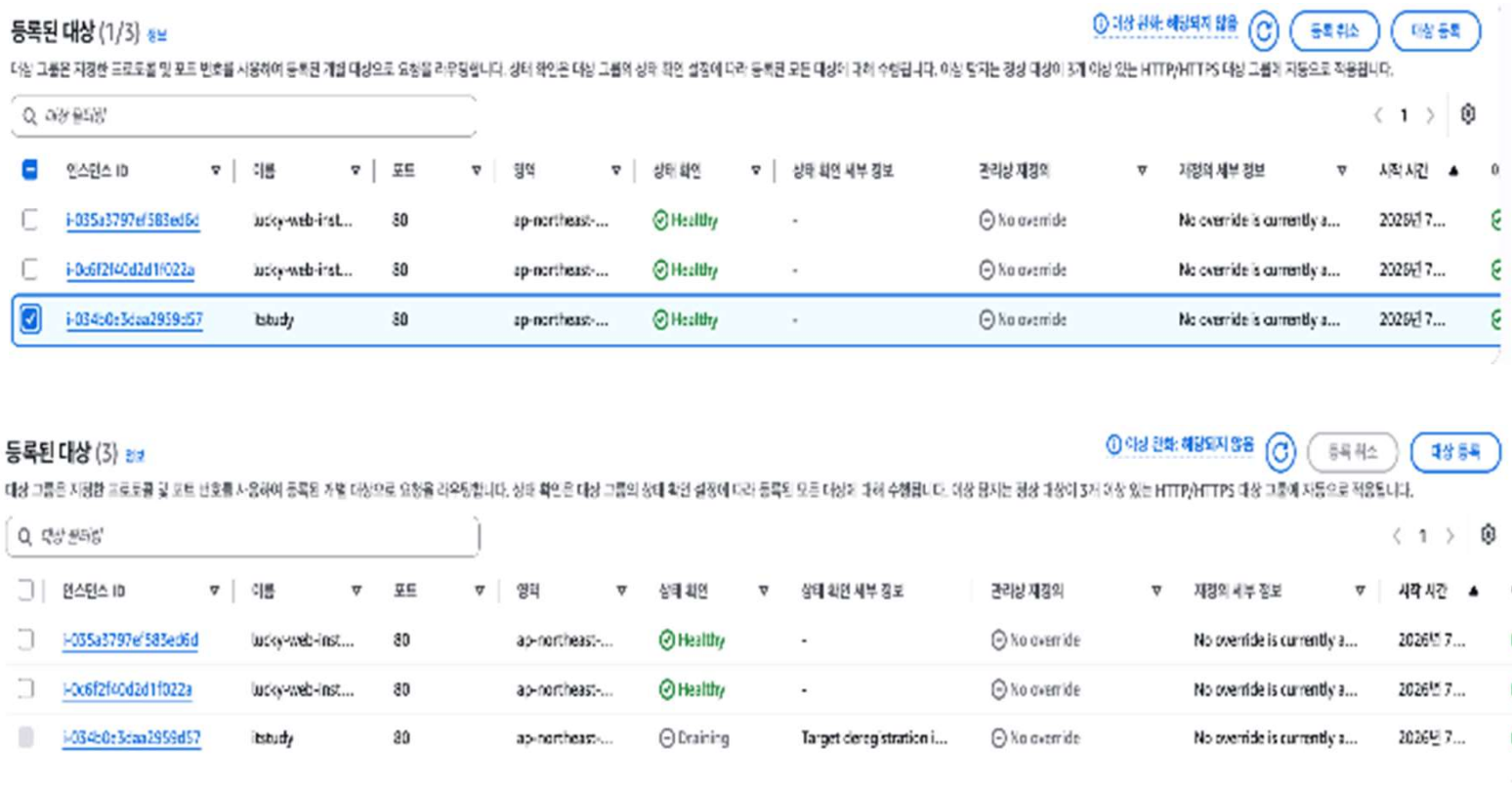
애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 테스트
    - ● 대상 그룹 확인



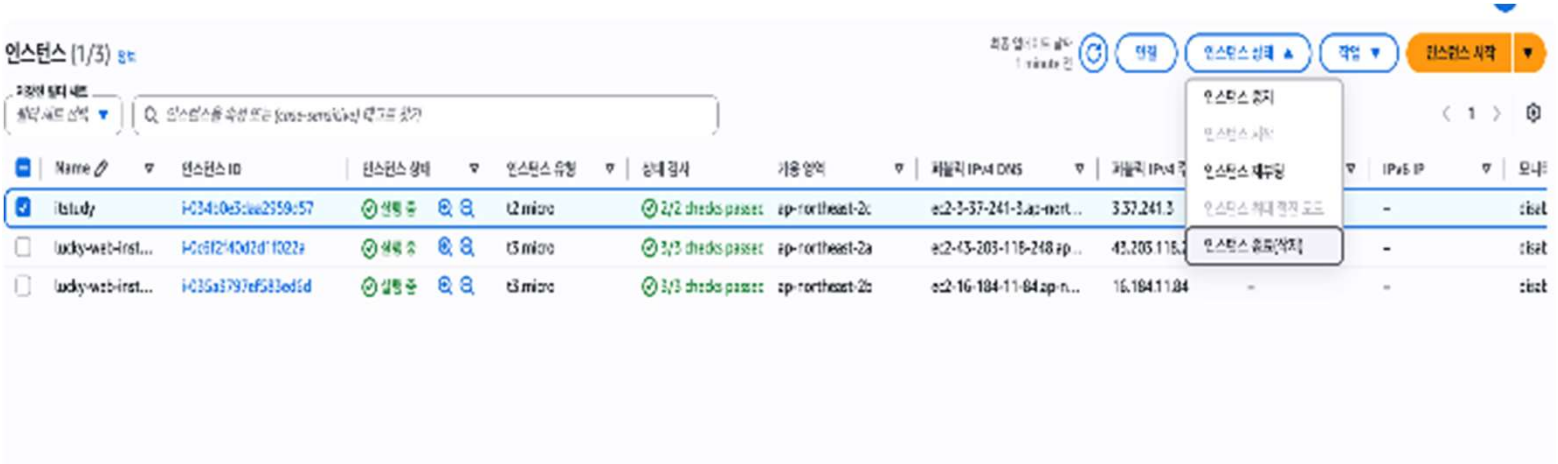
애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 테스트
    - ● 기존 인스턴스 종료 – 대상 그룹에서 인스턴스를 선택하고 등록 취소를 클릭(5분 소요)



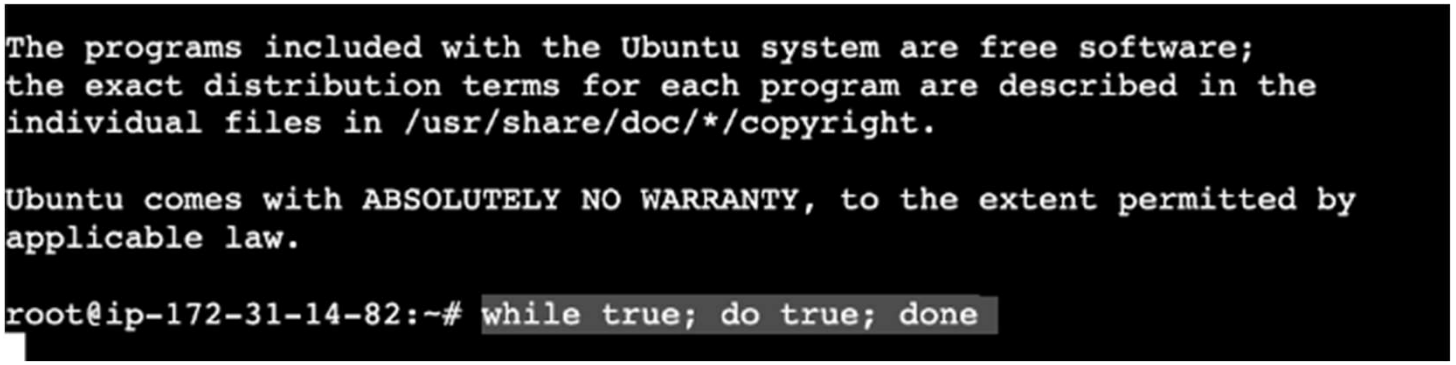
애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 테스트
    - ● 기존 인스턴스 종료 – 인스턴스 메뉴로 이동해서 [인스턴스 상태] – [인스턴스 종료]



애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 테스트
    - ● 테스트를 위해서 하나의 인스턴스를 선택하고 연결을 클릭한 후 작성: while true; do true; done

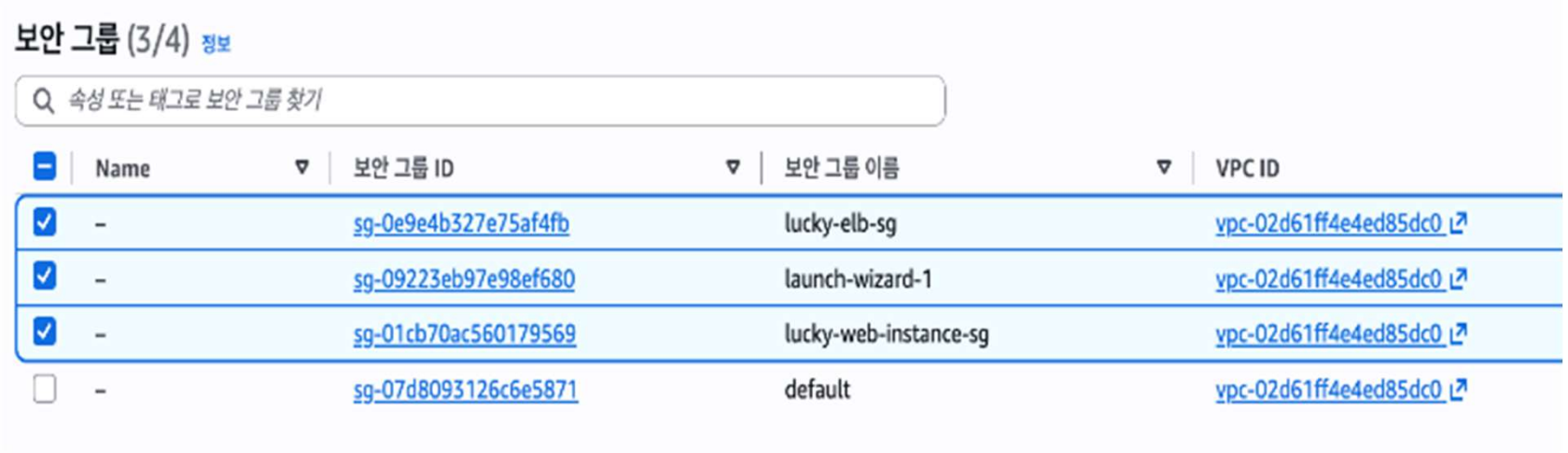


애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ Auto Scaling 그룹 한계
    - 갑자기 트래픽이 폭주하게 되면(오토 스케일링이 동작해서 인스턴스를 생성하는 속도보다 트래픽이 더 빨리 증가하는 경우) 전체 시스템에 장애가 발생
    - 오토 스케일링은 트래픽이 점진적으로 증가하는 경우에 적합
    - 갑자기 폭주하는 트래픽이 예상된다면 미리 인스턴스를 생성해야 함
    - AWS에서 오토 스케일링을 설정하는 방법
      - ◆ 동적 크기 조정 정책 생성: 점진적으로 증가하는 트래픽에 적합
      - ◆ 예약된 작업: 특정 시간에 맞춰 인스턴스를 생성해 놓는 방법
      - ◆ 예측 크기 조정 정책: 일정한 패턴으로 들어오는 트래픽이 있다면 며칠동안 오토 스케일링을 교육 시키고 그 이후로 예측되는 트래픽에 맞춰 미리 인스턴스를 만들어 놓는 방법

애플리케이션 배포

- ❖ Auto Scaling Group을 활용한 고가용성 아키텍쳐 구성
  - ✓ 리소스 삭제
    - Auto Scaling 그룹 삭제
    - 시작 템플릿 삭제
    - AMI 삭제(등록 취소)
    - 로드 밸런서 삭제
    - 대상 그룹 삭제
    - 보안 그룹 삭제



EC2를 외부에서 접속 가능한 서버로

- MySQL 서버
  - ✓ SSH 연결 후 작업
    - 패키지 정보 업데이트: sudo apt update
    - MySQL 설치: sudo apt install mysql-server
    - 버전 확인: mysql --version
    - MySQL 접속(비밀번호는 없거나 ubuntu): sudo mysql -u root -p
    - 비밀번호 변경
      - use mysql
      - alter user "root"@"localhost" identified with mysql_native_password by "암호";
      - 변경 내용 적용 - FLUSH PRIVILEGES;

EC2를 외부에서 접속 가능한 서버로

- MySQL 서버
  - ✓ SSH 연결 후 작업
    - 외부 접속 허용을 위한 설정
      - ~$ sudo su
      - cd /etc/mysql/mysql.conf.d
      - vi mysqld.cnf

        bind-address = 0.0.0.0
    - 서비스 재시작
      - service mysql restart

EC2를 외부에서 접속 가능한 서버로

- MySQL 서버 생성
  - ✓ SSH 연결 후 작업
    - MySQL에서 유저 생성 및 접속 권한 설정
      - mysql -u root -p
      - 데이터베이스 생성: create database [database 이름];
      - 유저 생성: CREATE USER [user 이름]@'%' IDENTIFIED BY '[password]';
      - 권한 부여: grant all privileges on [database 이름].* to '[user 이름]'@'%';
      - 변경 내용 적용: FLUSH PRIVILEGES;

EC2를 외부에서 접속 가능한 서버로

❖ MySQL 서버 생성

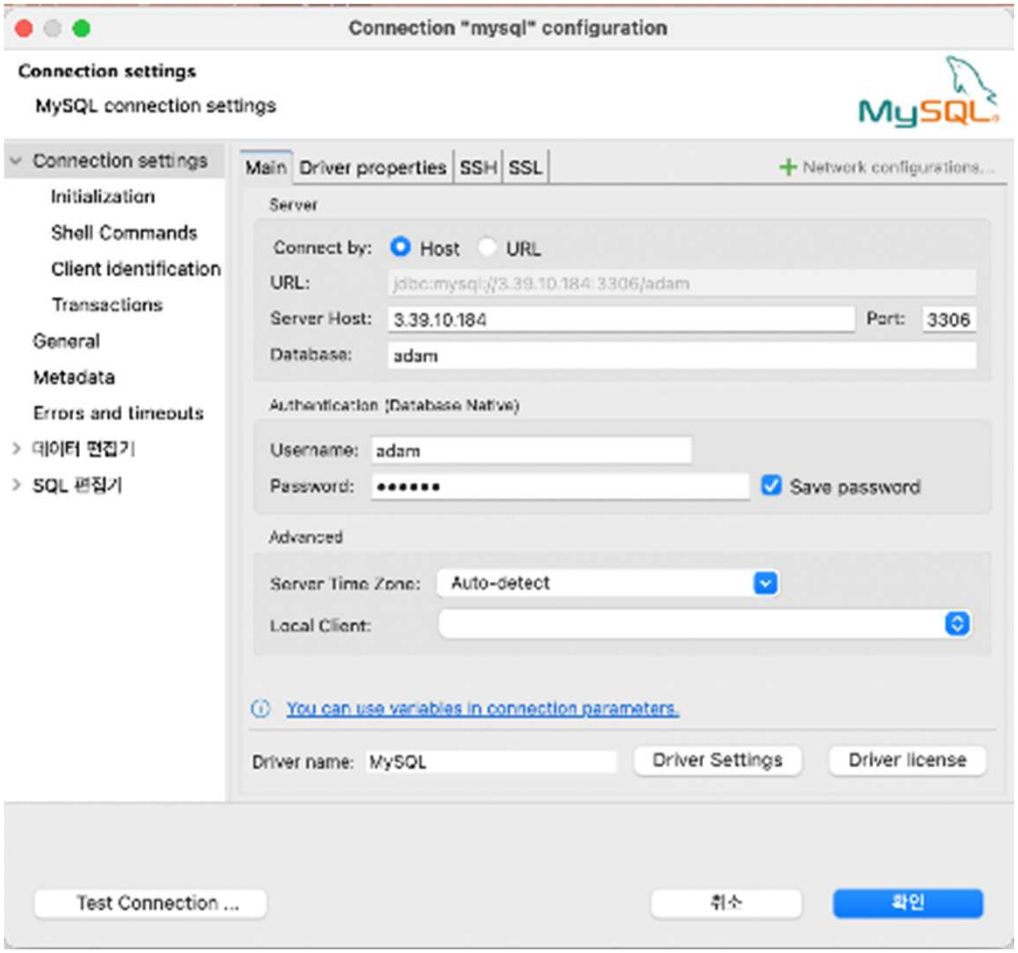
✓ EC2 의 보안 그룹에서 인바운드 규칙 수정(3306번 포트에 접근 가능하도록 추가)

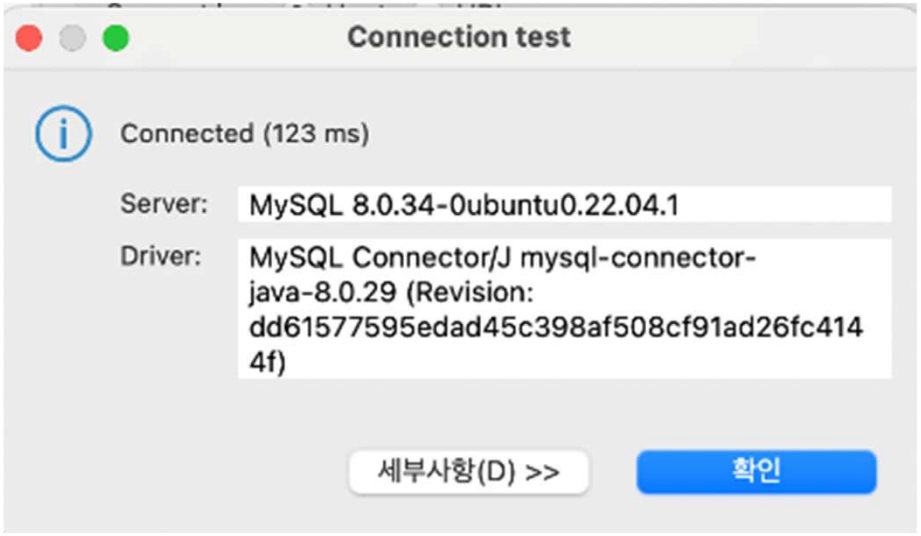
포트 범위	프로토콜	원본	보안 그룹
22	TCP	0.0.0.0/0	launch-wizard-2
3306	TCP	0.0.0.0/0	launch-wizard-2

EC2를 외부에서 접속 가능한 서버로

❖ MySQL 서버 생성

✓ 외부에서 접속





EC2를 외부에서 접속 가능한 서버로

- ❖ Mongo DB 서버
  - ✓ 서버 설치: https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-ubuntu/#std-label-install-mdb-community-ubuntu
    - 리눅스 버전 확인
      - ◆ cat /etc/lsb-release
    - gnupg(암호화 소프트웨어) 와 curl(다양한 통신 프로토콜을 이용하여 데이터를 전송하기 위한 라이브러리와 명령 줄 도구를 제공하는 컴퓨터 소프트웨어) 설치
      - ◆ sudo apt install gnupg curl
    - Mongo DB public key 가져오기
      - ◆ curl -fsSL https://pgp.mongodb.com/server-7.0.asc | sudo gpg -o /usr/share/keyrings/mongodb-server-7.0.gpg --dearmor
    - Mongo DB 목록 만들기
      - ◆ echo "deb [arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-7.0.gpg] https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/7.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-7.0.list deb [arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-7.0.gpg] https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/7.0 multiverse
    - 패키지 정보 업데이트: sudo apt update
    - Mongo DB 설치: sudo apt install -y mongodb-org

EC2를 외부에서 접속 가능한 서버로

- Mongo DB 서버
  - ✓ 서비스 관련 명령
    - 서비스 시작: sudo systemctl start mongod
    - 서비스 확인: sudo systemctl status mongod
    - 부팅 시 시작: sudo systemctl enable mongod
    - 서비스 중지: sudo systemctl stop mongod
    - 서비스 재시작: sudo systemctl restart mongod

EC2를 외부에서 접속 가능한 서버로

- ❖ Mongo DB 서버
  - ✓ 셸 실행
    - mongosh
  - ✓ 삭제
    - sudo service mongod stop
    - sudo apt purge "mongodb-org*"
    - sudo rm -r /var/log/mongodb
    - sudo rm -r /var/lib/mongodb

EC2를 외부에서 접속 가능한 서버로

- ❖ Mongo DB 서버
  - ✓ 외부 접속 허용 – 아이디 와 비밀번호 없이 접속
    - 설정 파일 수정: sudo nano /etc/mongod.conf

      net:

        port: 27017

        #bindIp: 127.0.0.1

        bindIp: 0.0.0.0
    - 방화벽에서 27017 포트 개방

      sudo ufw allow 27017
    - EC2의 보안 그룹에서 27017 포트를 외부에서 접속할 수 있도록 인바운드 규칙 수정
    - EC2에서 Mongo DB 서버 재시작:

      sudo systemctl restart mongod

    - 외부에서 접속
      - ◆ Mongo Shell: mongosh --host IP주소 --port 27017
      - ◆ Compass: mongodb://IP주소:27017

EC2를 외부에서 접속 가능한 서버로

- Mongo DB 서버
  - ✓ 외부 접속 허용 – 계정을 이용해서 접속
    - mongosh 명령으로 셸에 접속
    - 데이터베이스 생성: use admin
    - 유저와 비밀번호 설정

```
db.createUser({
  user: "myAdminUser",
  pwd: "securePassword123!", // 사용할 강력한 비밀번호 입력
  roles: [ { role: "userAdminAnyDatabase", db: "admin" }, "readWriteAnyDatabase" ]
})
```

    - 설정 파일에 추가: sudo nano /etc/mongod.conf

```
security:
  authorization: 'enabled'
```

    - 서버 재시작: sudo systemctl restart mongod

EC2를 외부에서 접속 가능한 서버로

- Mongo DB 서버
  - ✓ 외부 접속 허용 – 계정을 이용해서 접속
    - 접속
      - Mongo Shell: mongosh mongosh "mongodb://myAdminUser:securePassword123!@<EC2_퍼블릭_IP>:27017/admin?authSource=admin"
      - Compass: mongodb://myAdminUser:securePassword123!@<EC2_퍼블릭_IP>:27017/admin?authSource=admin

EC2를 외부에서 접속 가능한 서버로

- Mongo DB 서버 생성
  - ✓ 외부 접속 허용 – 계정을 이용해서 접속
    - MongoCompass를 이용한 접속