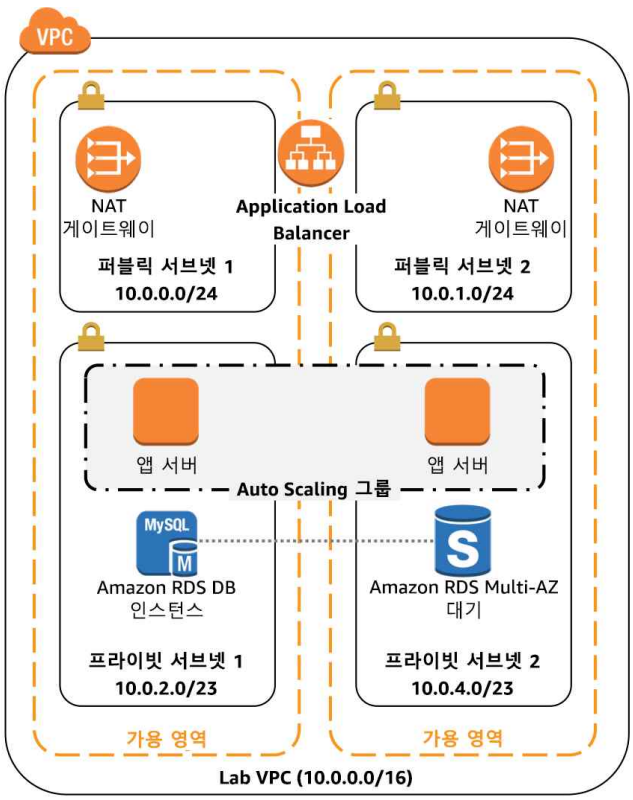
Architecting on AWS – 실습 4
고가용성 환경 생성

실습 개요

핵심 비즈니스 시스템은 고가용성 애플리케이션으로 배포되어야 합니다. 그렇게 배포되어야 일부 구성 요소에 장애가 발생해도 정상적으로 운영될 수 있습니다. AWS 에서 고가용성을 달성하려면 여러 가용 영역에서 서비스를 실행하는 것이 좋습니다.

로드 밸런서와 같은 많은 AWS 서비스는 본질적으로 가용성이 높습니다. 여러 가용 영역에 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 배포하는 등 고가용성을 위해 다른 서비스를 구성할 수 있습니다.

본 실습에서는 단일 Amazon EC2 인스턴스에서 실행되는 애플리케이션으로 시작한 다음, 이를 변환하여 가용성을 높입니다. Application Load Balancer 및 Auto Scaling 그룹을 생성하고, 보안 그룹을 업데이트하고, 애플리케이션의 가용성이 높은 지 테스트합니다. 다음 이미지는 최종 아키텍처를 보여줍니다.



두 가지 선택 사항 **챌린지** 작업이 제공됩니다. 첫 번째는 데이터베이스의 가용성을 높이는 것입니다. 두 번째는 NAT 게이트웨이의 가용성을 높이는 것입니다.

목표

이 실습을 완료하면 다음을 할 수 있게 됩니다.

- Application Load Balancer 생성
- Amazon EC2 Auto Scaling 그룹 생성
- 보안 그룹을 업데이트하여 3 티어 아키텍처 적용

소요 시간

이 실습은 완료하는 데 약 5**0 분**이 소요됩니다.

실습 시작

1. 이 링크를 마우스 오른쪽 버튼으로 클릭한 다음 자신의 컴퓨터로 **arc_lab4_template.json** 을 다운로드합니다.
2. AWS Management Console 의 **서비스** 메뉴에서 **Management & Governance > CloudFormation** 을 클릭합니다.
3. **Create stack** 을 클릭하고 아래 단계에 따라 스택을 생성합니다.

   1 단계: 템플릿 지정

   - **Template source:** **Upload a template file** 을 선택합니다.
   - **Upload a template file:** **Choose file** 을 클릭하고 다운로드한 **arc_lab4_template.json** 파일을 선택합니다.
   - **Next** 를 클릭합니다.

   2 단계: 스택 세부 정보 지정

   - **Stack name:** arclab4
   - **Next** 를 클릭합니다.

   3 단계: 스택 옵션 구성

- **Next** 를 클릭합니다.

4 단계: 검토

- **I acknowledge that...** 의 체크박스에 체크합니다.
- **Create stack** 을 클릭합니다.

AWS CloudFormation 에서는 이제 템플릿을 사용하여 리소스의 *스택* 을 생성합니다.

Stack info 탭을 클릭합니다.

- **Status** 가 **CREATE_COMPLETE** 로 변경될 때까지(약 12 분) 대기합니다.

  참고 필요한 경우 새로 고침 아이콘을 15 초마다 클릭하면 화면이 업데이트됩니다.

4. **Outputs** 탭을 클릭합니다.

AWS CloudFormation 스택에서 지정된 리소스 ID 및 리소스 링크와 같은 *출력 정보* 를 제공할 수 있습니다.

- **Endpoint:** RDS 에 생성 된 Database Endpoint.
- **NATGateway1:** NAT Gateway 의 ID.
- **PublicSubnet2:** Public Subnet 2 의 ID.
- **Region:** 생성된 리소스들의 리전 코드입니다.

작업 1: VPC 검사

실습의 일환으로 AWS CloudFormation 을 통해 다음 리소스가 이미 프로비저닝 되었습니다.

- Amazon Virtual Private Cloud(Amazon VPC)
- 2 개의 가용 영역에 있는 퍼블릭 및 프라이빗 서브넷
- 퍼블릭 서브넷에 연결된 인터넷 게이트웨이(다이어그램에 표시되지 않음)
- 퍼블릭 서브넷 중 하나에 있는 NAT 게이트웨이
- 프라이빗 서브넷 중 하나에 있는 Amazon Relational Database Service(Amazon RDS) DB 인스턴스

다음 이미지는 최초 아키텍처를 보여줍니다.

본 작업에서는 이미 생성된 VPC 구성을 검토합니다.

5. AWS Management Console 의 **Services** 메뉴에서 **Networking & Content Delivery > VPC** 를 클릭합니다.

6. 화면 왼쪽 상단에 **New VPC Experience** 가 표시되면, **New VPC Experience** 가 선택되었는지 확인하십시오. 이 실습은 새로운 EC2 콘솔을 사용하도록 설계되었습니다.

7. 왼쪽 탐색 창의 **Filter by VPC** 에서 **Select a VPC** 상자를 클릭하고 *Lab VPC* 를 선택합니다.

그러면 콘솔이 Lab VPC 에 연결된 리소스만 표시하도록 제한됩니다.

8. 왼쪽 탐색 창에서 **Your VPCs** 를 클릭합니다.

여기에서 사용자가 생성한 Lab VPC 를 볼 수 있습니다.

9. 왼쪽 탐색 창에서 **Subnets** 를 클릭합니다.

여기에서는 Lab VPC 의 일부인 서브넷을 볼 수 있습니다. **Public Subnet 1** 에서 열의 세부 정보를 확인합니다.

- **VPC** 열에서 이 서브넷이 **Lab VPC** 내부에 존재하는 것을 확인할 수 있습니다.
- **IPv4 CIDR** 열에서 값은 **10.0.0.0/24** 입니다. 이는 해당 서브넷이 10.0.0.0 과 10.0.0.255 사이의 IP 256 개(이 중 5 개는 예약되었거나 사용 불가능)를 포함한다는 뜻입니다.
- **가용 영역** 열에서 이 서브넷이 존재하는 가용 영역을 볼 수 있습니다.

10. **Public Subnet 1** 을 선택하면 페이지 하단에 세부 정보가 표시됩니다.

팁: 구분선을 위아래로 끌어 하단 창을 확장할 수 있습니다.

11. 페이지 하단에서 **Route Table** 탭을 클릭합니다.

여기에서 이 서브넷의 라우팅에 대한 세부 정보를 볼 수 있습니다.

- 첫 번째 항목은 VPC 의 CIDR 범위(**10.0.0.0/20**) 내로 향하는 트래픽이 VPC(**로컬**) 내에서 라우팅되도록 지정합니다.
- 두 번째 항목은 인터넷(**0.0.0.0/0**)으로 향하는 트래픽이 인터넷 게이트웨이(**igw-xxxx**)로 라우팅되도록 지정합니다. 이 설정은 *퍼블릭* 서브넷을 만듭니다.

12. **Network ACL** 탭을 클릭합니다.

여기에서 서브넷과 연결된 네트워크 ACL(액세스 제어 목록)을 볼 수 있습니다. 규칙은 현재 **모든 트래픽**이 서브넷 내외로 흐르도록 허용합니다. 보안 그룹을 사용하여 트래픽을 추가로 제한할 수 있습니다.

13. 왼쪽 탐색 창에서 **Internet Gateways** 를 클릭합니다.

참고로 인터넷 게이트웨이는 이미 Lab VPC 에 연결되어 있습니다.

14. 왼쪽 탐색 창에서 **Security Groups** 를 클릭합니다.

15. **Inventory-DB** 보안 그룹을 선택합니다.

이는 데이터베이스의 수신 트래픽을 제어하는 데 사용되는 보안 그룹입니다.

16. 페이지 하단에서 **Inbound rules** 탭을 클릭합니다.

보안 그룹은 VPC(*10.0.0.0/20*) 내 어디에서나 인바운드 MYSQL/Aurora 트래픽(포트 3306)을 허용합니다. 나중에 애플리케이션 서버의 트래픽만 허용하도록 이를 수정할 것입니다.

17. **Outbound rules** 탭을 클릭합니다.

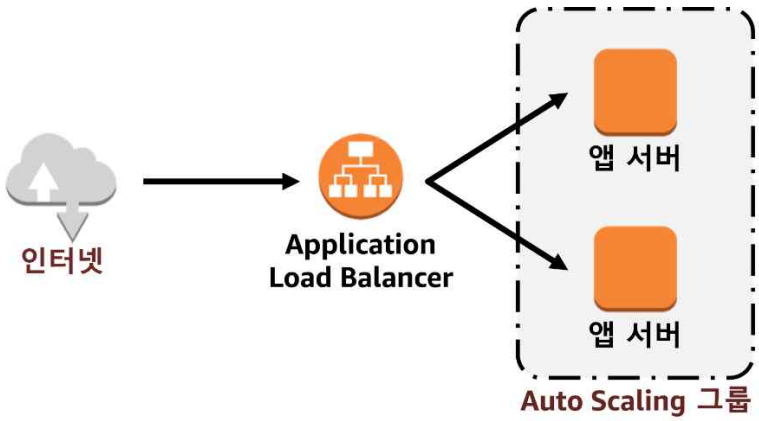
기본적으로 보안 그룹은 모든 아웃바운드 트래픽을 허용합니다. 하지만 필요에 따라 이러한 규칙을 수정할 수 있습니다.

작업 2: Application Load Balancer 생성

고가용성 애플리케이션을 구축하려면 *여러 가용 영역* 에서 리소스를 시작하는 것이 좋습니다. 가용 영역은 동일 리전 내에서 물리적으로 분리된 데이터 센터(또는 데이터 센터 그룹)입니다. 여러 가용 영역에서 애플리케이션을 실행하면 데이터 센터에서 장애가 발생한 경우에 뛰어난 *가용성* 을 제공합니다.

애플리케이션 서버에서 애플리케이션이 실행되는 것을 고려하면 해당 서버 간에 트래픽을 분산할 수 있는 방법이 필요합니다. *로드 밸런서* 를 사용하여 이 작업을 수행할 수 있습니다. 로드 밸런서는 수신되는 애플리케이션 트래픽을 여러 인스턴스로 자동 분산합니다. 또한 로드 밸런서는 인스턴스 상태 확인을 수행하고 정상인 인스턴스에만 요청을 전송합니다.

다음 다이어그램은 Application Load Balancer 가 수신 트래픽을 여러 애플리케이션 서버로 분산하는 방법을 보여줍니다.



이 작업에서는 Application Load Balancer 를 생성 및 구성합니다.

18. **AWS Management Console** 에서 **Services** 메뉴에서 **Compute > EC2** 를 클릭합니다.

19. 왼쪽 탐색 창에서 **Target groups** 을 클릭합니다.

Target groups 은 로드 밸런서로 들어오는 트래픽을 *전송* 할 위치를 정의합니다. Application Load Balancer 은 수신 요청의 URL 을 기준으로 트래픽을 여러 대상 그룹으로 전송할 수 있습니다. 예를 들어 모바일 앱으로의 요청은 다른 서버 집합으로 전송될 수 있습니다. 이 실습에서 웹 애플리케이션은 하나의 대상 그룹만 사용합니다.

20. **Create target group** 을 클릭합니다.

Specify group details 페이지가 표시가 될 것입니다.

21. **Basic configuration** 섹션에서 아래와 같이 구성합니다:

- **Choose a target type** 을 *Instances* 으로 선택합니다.
- **Target group name** 에 Inventory-App 을 입력합니다.
- **VPC** 를 *LabVPC* 으로 선택합니다.

22. **Advanced health check settings** 섹션을 확장합니다.

Application Load Balancer 는 모든 인스턴스에서 *Health checks* 를 자동으로 수행하여 요청에 응답하는지 확인합니다. 기본 설정이 권장되지만 본 실습에서 사용할 때는 약간 더 빠르게 설정하겠습니다.

23. 다음 값을 구성합니다.(나머지는 그대로 둔다)

- **Healthy threshold:** 2 (정상 임계값)
- **Interval:** 10 (간격)

이를 통해서 10 초마다(*interval*) 상태 검사를 수행하게 됩니다. 인스턴스가 한 행에서 두 번 올바르게 응답(*threshold*)하면 정상 상태로 간주됩니다.

이 페이지의 나머지 설정 값들은 기본으로 두시면 됩니다.

24. **Next**을 클릭합니다.

Register targets 페이지가 표기될 것입니다.

*Targets*은 로드 밸런서의 요청에 응답할 개별 인스턴스입니다. 아직 웹 애플리케이션 인스턴스가 없기 때문에 이 단계를 건너뛸 수 있습니다.

25. **Create target group**을 클릭합니다.

26. 아래와 같은 메시지가 표시 될 것입니다.

- Successfully created target group: Inventory-App

27. 왼쪽 탐색 창에서 **Load Balancers**를 클릭합니다.

28. **Create Load Balancer**를 클릭합니다.

29. **Application Load Balancer** 세션에서의 `Create`를 클릭합니다.

Create Application Load Balancer 페이지가 표시될 것입니다.

30. **Basic Configuration** 섹션에서 아래와 같이 구성합니다:

- **Load balancer name** 에 `Inventory-LB`를 입력합니다.

31. **Network mapping** 섹션에서 아래와 같이 구성합니다:

- **VPC**: *LabVPC*를 선택합니다.
- **Mappings**:
  - 첫번째 가용 영역 체크박스를 선택하고 서브넷 리스트에서 `PublicSubnet1`을 선택합니다.
  - 두번째 가용 영역 체크박스를 선택하고 서브넷 리스트에서 `PublicSubnet2`을 선택합니다.

32. **Security groups** 섹션에서 아래와 같이 구성합니다.

- *default* security group 을 삭제합니다.
- 드롭다운 메뉴에서 `LabALBSecurityGroup`을 선택합니다.

33. **Listeners and routing** 에서 아래와 같이 설정합니다.

- **Listener HTTP:80** 에서 Default Action 드롭다운 메뉴에서 *Inventory-App* 을 선택합니다.

34. **Create load balancer** 을 클릭합니다.

35. 아래와 같은 메시지가 표시될 것입니다:

- Successfully created load balancer:Inventory-LB

Load balancer **Inventory-LB** 가 성공적으로 생성되었습니다.

36. **View load balancer** 를 클릭합니다.

37. 이제 Application Load Balancer 가 백그라운드에서 프로비저닝됩니다. 기다리지 않고 다음 작업을 계속할 수 있습니다.

시작 템플릿을 사용하여 그룹을 생성하려면 먼저 Amazon 머신 이미지(AMI)의 ID 및 인스턴스 유형 등 EC2 인스턴스를 시작하는 데 필요한 파라미터를 포함하는 시작 템플릿을 생성해야 합니다.

이 작업에서는 시작 템플릿을 생성합니다.

38. **Services** 메뉴에서 **EC2** 를 클릭합니다.

39. 왼쪽 탐색 창의 **Instances** 아래에서 **Launch Templates** 를 선택합니다.

만약에 템플릿이 이미 있다면, 그것을 선택하시고, **Actions** 을 클릭한 다음에, **Delete template** 을 눌러 삭제하십시오.

40. **Create launch template** 을 선택합니다.

41. **Launch template name and description** 섹션에서 다음을 구성합니다.

- **Launch template name:** Lab-template-NUMB

주의 - NUMBER 값은 임의의 숫자를 사용하십시오.

예를 들어) Lab-template-9846954

만일 탬플릿의 이름이 이미 존재한다고 나오면 다른 숫자를 사용하시고 다시 시도해 보십시오.

- **Template version description:** version 1

인스턴스의 루트 볼륨에 대한 템플릿이며 운영 체제, 애플리케이션 서버 및 애플리케이션을 포함할 수 있는 **Amazon Machine Image(AMI)** 를 선택하라는 메시지가 표시됩니다. AMI 에서 **인스턴스**를 바로 시작할 수 있는데, 이 인스턴스는 AMI 의 사본으로, 클라우드에서 실행되는 가상 서버입니다.

AMI 는 다양한 버전의 Windows 및 Linux 에서 사용할 수 있습니다. 이 실습에서는 **Amazon Linux** 를 실행하는 인스턴스를 시작합니다.

42. **Application and OS Images (Amazon machine Image)**의 Quick Start 에서 *Amazon Linux* 를 선택합니다.

참조 *Amazon Linux 2 AMI* 에 *64-bit (x86) Architecture* 인지 다시 한번 확인하십시오.

43. **Instance type** 에서 *t2.micro* 를 선택합니다.

인스턴스를 시작할 때 **인스턴스 유형**에 따라 인스턴스에 할당된 하드웨어가 결정됩니다. 각 인스턴스 유형은 서로 다른 컴퓨팅, 메모리, 스토리지 용량을 제공하는데, 이 용량에 따라 서로 다른 **인스턴스 패밀리**로 분류됩니다

44. **Security groups** 에서 *Inventory-App* 을 선택합니다.

45. 아래로 스크롤하여 **Advanced Details** 섹션을 찾습니다.

46. **Advanced details** 를 확장합니다.

47. **IAM instance profile**:에서 *Inventory-App-Role* 을 선택합니다.

48. **User data** 섹션에서 다음을 추가합니다.

```
#!/bin/bash
# Install Apache Web Server and PHP
yum install -y httpd mysql
amazon-linux-extras install -y php7.2
# Download Lab files
wget https://s3.us-west-2.amazonaws.com/arclab.applaycrew.com/Lab4Files/inventory-app.zip
unzip inventory-app.zip -d /var/www/html/
# Download and install the AWS SDK for PHP
wget https://s3.us-west-2.amazonaws.com/arclab.applaycrew.com/Lab4Files/aws.zip
```

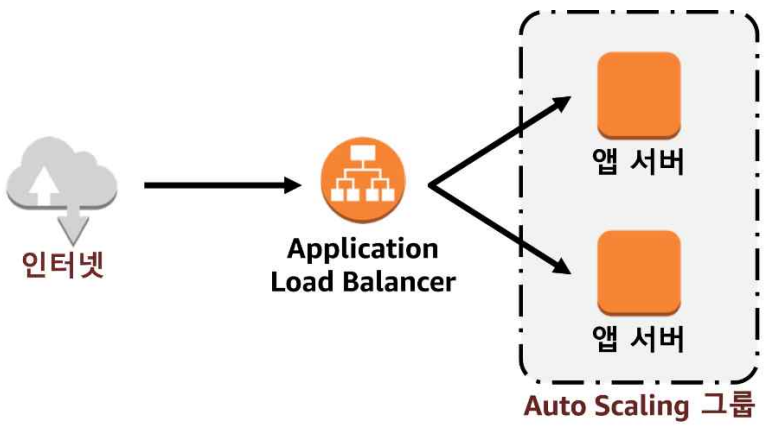
```
unzip aws -d /var/www/html
# Turn on web server
chkconfig httpd on
service httpd start
```

49. **Create launch template** 을 클릭합니다.
50. **View launch templates** 를 클릭합니다.

작업 3: Auto Scaling 그룹 생성

Amazon EC2 Auto Scaling 은 사용자가 정의한 정책, 일정 및 상태 확인에 따라 자동으로 Amazon EC2 인스턴스를 *시작* 또는 *종료* 하도록 설계된 웹 서비스입니다. 또한 여러 가용 영역에 인스턴스를 자동으로 분산하여 고가용성 애플리케이션을 생성할 수 있습니다.

프라이빗 서브넷 에 Amazon EC2 인스턴스를 배포하는 Auto Scaling 그룹을 생성합니다. 프라이빗 서브넷의 인스턴스는 인터넷에서 액세스할 수 없기 때문에 애플리케이션을 배포하기 위한 보안 모범 사례입니다. 대신 사용자가 Application Load Balancer 에 요청을 보내면 다음 다이어그램과 같이 해당 요청을 프라이빗 서브넷에 있는 Amazon EC2 인스턴스에 전달합니다.



51. 왼쪽 탐색 창의 **Auto Scaling** 아래에서 **Auto Scaling Groups** 를 클릭합니다.
52. **Create Auto Scaling group** 을 클릭하고 다음을 구성합니다.

- **Name:** Inventory-ASG

- **Launch template:** 생성한 시작 템플릿을 선택합니다.
- **Next**를 클릭합니다.

53. **Network** 섹션에서 다음을 구성합니다.

- **VPC:** *Lab VPC*
- **Availability Zones and Subnets:** *Private Subnet 1 and Private Subnet 2* 를 선택합니다.

54. **Next**를 클릭합니다.

55. **Configure advanced options** 페이지에서 다음을 구성합니다.

- **Attach to an existing load balancer** 를 선택합니다.
- **Choose from your load balancer target groups** 를 선택합니다.
- **Existing load balancer target groups:** *Inventory-App\|HTTP* 를 선택합니다.

그러면 자동 스케일링 그룹이 이전에 생성한 *Inventory-App|HTTP* 대상 그룹의 일부로 새 EC2 인스턴스를 등록하도록 합니다. 로드 밸런서가 이 대상 그룹에 있는 인스턴스로 트래픽을 보냅니다.

- **Health check grace period:** 200
- **Monitoring:** *Enable group metrics collection within CloudWatch*
- **Next**를 클릭합니다.

기본적으로 상태 확인 유예 기간은 300 으로 설정됩니다. 이 환경은 실습 환경이므로 Auto Scaling 이 첫 번째 상태 확인을 수행하는 데 매우 오래 기다릴 필요가 없도록 200 으로 설정했습니다.

56. **Configure group size and scaling policies** 페이지에서 다음을 구성합니다.

- **Desired capacity:** 2
- **Minimum capacity:** 2
- **Maximum capacity:** 2
- **Next**를 클릭합니다.

본 실습에서는 항상 2 개의 인스턴스를 유지하여 고가용성을 보장합니다. 애플리케이션이 다양한 트래픽 부하를 수신할 것으로 예상되는 경우 인스턴스를

시작/종료할 시기를 정의하는 *조정 정책*을 생성할 수도 있습니다. 하지만 본 실습의 Inventory 애플리케이션에서는 필요하지 않습니다.

57. **Tags** 페이지가 표시될 때까지 **Next**를 클릭합니다.

58. **Add tag**를 클릭하고 다음을 구성합니다.

- **Key:** Name
- **Value:** Inventory-App

그러면 Auto Scaling 그룹에 이름으로 태그가 지정되고 Auto Scaling 그룹에서 시작한 EC2 인스턴스에도 표시됩니다. 그러면 어떤 애플리케이션에 어떤 EC2 인스턴스가 연결되어 있는지 더 쉽게 식별할 수 있습니다. 또한 비용 센터 같은 태그를 추가하면 결제 파일에서 애플리케이션 비용을 쉽게 지정할 수도 있습니다.

59. **Next**를 클릭합니다.

60. Auto Scaling 그룹의 세부 정보를 확인한 다음 **Create Auto Scaling group**을 클릭합니다.

애플리케이션이 곧 2 개의 가용 영역에서 실행될 것이며 Auto Scaling 이 인스턴스 또는 가용 영역에서 장애가 발생하는 경우에도 해당 구성을 유지합니다.

이제 Auto Scaling 그룹을 생성했으며 해당 그룹에서 EC2 인스턴스를 시작했는지 확인할 준비가 완료되었습니다.

61. 생성된 Auto Scaling 그룹 이름을 클릭합니다.

Details 탭에서 보여지는 내용을 검토하여 Auto Scaling 그룹에 대한 정보를 확인합니다.

62. **Activity** 탭을 클릭합니다.

Status 열에는 인스턴스의 현재 상태가 포함됩니다. 인스턴스를 시작하면 상태 열에 *PreInService* 가 표시됩니다. 인스턴스가 시작되면 상태가 *Successful* 로 변경됩니다. 새로 고침 버튼 을 클릭하여 인스턴스의 현재 상태를 볼 수도 있습니다.

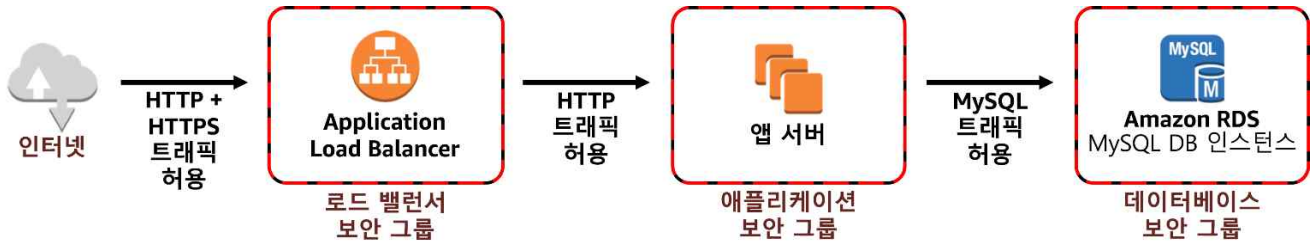
63. **Instance management** 탭을 클릭합니다.

Auto Scaling 그룹에서 EC2 인스턴스를 시작했으며 해당 인스턴스가 *InService* 수명 주기 상태에 있음을 알 수 있습니다. Health Status 열에 해당 인스턴스에 대한 EC2 인스턴스 상태 확인 결과가 표시됩니다.

64. **Monitoring** 탭을 클릭합니다. 여기에서 Auto Scaling 그룹에 대한 모니터링 관련 정보를 볼 수 있습니다.

작업 4: 보안 그룹 업데이트

배포한 애플리케이션에는 *3 티어 아키텍처* 가 있습니다. 이 작업에서는 다음 이미지와 같이 이러한 티어를 적용하도록 보안 그룹을 구성합니다.



로드 밸런서 보안 그룹

Application Load Balancer 를 생성할 때 이미 로드 밸런서 보안 그룹을 구성했습니다. 이 보안 그룹은 모든 수신 HTTP 및 HTTPS 트래픽을 허용합니다.

로드 밸런서는 수신 요청을 대상 그룹으로 전달하도록 구성되었습니다. Auto Scaling 이 새로운 인스턴스를 시작하면 해당 인스턴스를 대상 그룹에 자동으로 추가합니다.

애플리케이션 보안 그룹

애플리케이션 보안 그룹은 본 실습 설정의 일부로 제공되었습니다. 이제 로드 밸런서에서 수신되는 트래픽만 허용하도록 구성합니다.

65. 왼쪽 탐색 창에서 **Security Groups** 를 클릭합니다.

66. **Inventory-App** 을 선택합니다(다른 보안 그룹을 선택하면 안 됩니다).

67. 페이지 하단에서 **Inbound rules** 탭을 클릭합니다.

보안 그룹은 현재 비어 있습니다. 로드 밸런서에서 수신되는 HTTP 트래픽을 허용하는 규칙을 추가합니다. 로드 밸런서는 HTTP 를 통해 HTTPS 요청을 전달하도록 구성되었기 때문에 HTTPS 트래픽을 구성할 필요가 없습니다. 그러면 보안을 로드 밸런서로 오프로드하여 개별 애플리케이션 서버에서 필요한 작업량이 감소합니다.

68. **Edit inbound rules** 를 클릭합니다.
69. **Add rule** 을 클릭하고 다음을 구성합니다.

- **Type:** *HTTP*
- **Source:**
  - **Custom** 오른쪽에 있는 상자에 `sg` 를 입력합니다.
  - 표시되는 목록에서 *LabALBSecurityGroup* 을 선택합니다.
- **Description:** `Traffic from load balancer`

70. **Save rules** 를 클릭합니다.

이제 애플리케이션 서버가 로드 밸런서의 트래픽을 수신할 수 있습니다. 여기에는 로드 밸런서가 자동으로 수행하는 상태 확인이 포함됩니다

.

데이터베이스 보안 그룹

이제 애플리케이션 서버에서 수신되는 트래픽만 허용하도록 데이터베이스 보안 그룹을 구성합니다.

71. **Inventory-DB** 를 선택합니다(다른 보안 그룹을 선택하면 안 됩니다).

기존 인바운드 규칙은 포트 3306(MySQL 에서 사용됨)에서 VPC 내에 있는 모든 IP 주소의 트래픽을 허용합니다. 좋은 규칙이지만 보안을 추가로 강화할 수 있습니다.

72. **Inbound rules** 탭에서 **Edit inbound rules** 를 클릭합니다.
73. **Delete** 를 클릭해서 기존 inbound rule 을 삭제합니다.
74. **Add rule** 를 클릭한 후 아래와 같이 구성합니다.

- **Type** 에서 드롭다운을 클릭해서 MYSQL/Aurora 를 선택합니다.

- **Source**:의 경우 **Custom** 오른쪽에 있는 상자에 `sg` 를 입력합니다.
- 표시되는 목록에서 *Inventory-App* 을 선택합니다.
- **Description:** `Traffic from app servers` 를 입력합니다.

75. **Save rules** 를 클릭합니다.

이제 3 티어 보안이 구성되었으며, 티어에 있는 각 요소는 위 티어의 트래픽만 허용합니다.

또한 프라이빗 서브넷을 사용한다는 것은 인터넷과 사용자의 애플리케이션 리소스 사이에 두 가지 보안 장벽이 있음을 의미합니다. 이는 다중 보안 계층을 적용하는 모범 사례와 일치합니다.

작업 5: 애플리케이션 테스트

이제 애플리케이션을 테스트할 준비가 되었습니다. 이 작업에서 웹 애플리케이션이 실행 중이고 가용성이 높은 지 확인합니다.

76. EC2 에 가서 왼쪽 탐색 창에서 **Target Groups** 를 선택합니다.

77. **Name** 의 *Inventory-App* 을 클릭합니다.

78. 페이지 하단에서 **Targets** 탭을 클릭합니다.

Registered targets 섹션에 2 개의 인스턴스가 표시될 것입니다. **Status** 열에는 인스턴스에 대해 수행한 로드 밸런서 상태 확인 결과가 표시됩니다.

79. 두 인스턴스 모두 **Status** 가 *healthy* 로 표시될 때까지 몇 초마다 오른쪽 상단 새로 고침 아이콘을 수시로 클릭합니다.

상태가 끝까지 *healthy* 로 변경되지 않을 경우 강사에게 구성 진단 지원을 요청하십시오..

애플리케이션은 Application Load Balancer 에 연결하여 테스트하게 됩니다. 그러면 Amazon EC2 인스턴스 중 하나로 사용자의 요청을 전송합니다. 먼저 Application Load Balancer 의 DNS 이름을 검색해야 합니다.

80. 왼쪽 탐색 창에서 **Load Balancers** 를 클릭합니다.

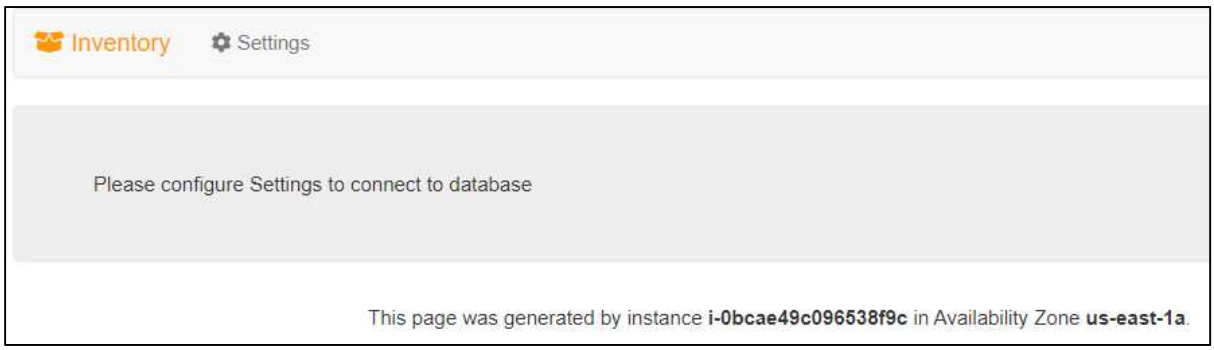
81. Inventory-LB 를 클릭하여는 **Description** 에서 **DNS name** 을 찾아 클립보드로 복사합니다.

Inventory-LB-xxxx.elb.amazonaws.com 과 유사할 것입니다.

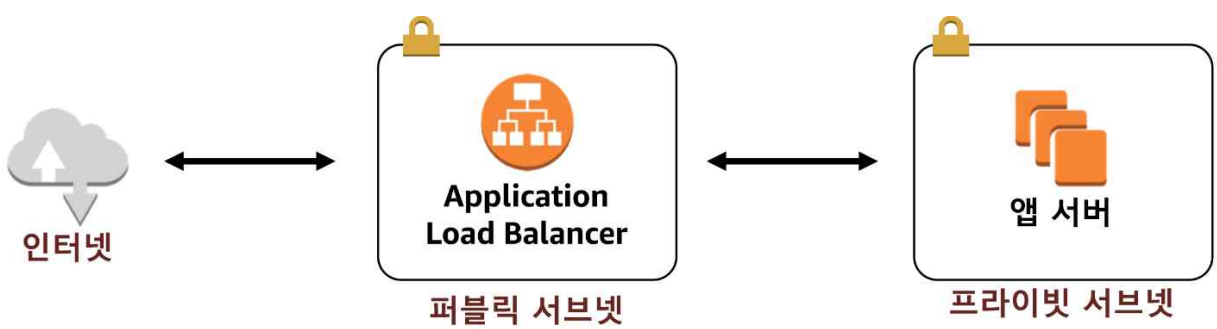
82. 새 웹 브라우저 탭을 열고 클립보드에서 DNS 이름을 붙여넣고 Enter 키를 누릅니다.

로드 밸런서가 Amazon EC2 인스턴스 중 하나로 사용자의 요청을 전달합니다. 인스턴스 ID 와 가용 영역은 웹 페이지 하단에 표시됩니다.

83. 웹 브라우저에서 페이지를 몇 번 새로 고칩니다. 인스턴스 ID 와 가용 영역은 두 인스턴스 사이에서 변경되는 경우가 있다는 점에 유의해야 합니다.



다음 이미지는 이 웹 애플리케이션에 대한 정보 흐름을 표시합니다.



정보의 흐름은 다음과 같습니다.

- *퍼블릭* 서브넷에 상주하는 Application Load Balancer 로 요청을 전송했습니다. 퍼블릭 서브넷은 인터넷에 연결되어 있습니다.
- Application Load Balancer 가 *프라이빗* 서브넷에 상주하는 Amazon EC2 인스턴스 중 하나를 선택했고 인스턴스로 요청을 전달했습니다.
- 이후 Amazon EC2 인스턴스는 Application Load Balancer 로 웹 페이지를 반환했고, 이는 사용자의 웹 브라우저로 반환되었습니다.

작업 6: 고가용성 테스트

애플리케이션은 고가용성이 되도록 구성되었습니다. Amazon EC2 인스턴스 중 하나를 종료하여 이를 테스트할 수 있습니다.

84. **EC2 Management Console** 로 돌아갑니다. 단, 애플리케이션 탭은 닫지 마십시오. (곧 이 탭으로 돌아올 것입니다.)

85. 왼쪽 탐색 창에서 **Instances** 를 클릭합니다.

이제 웹 애플리케이션 인스턴스 중 하나를 중지하여 장애를 시뮬레이션합니다.

86. **Inventory-App** 인스턴스 중 하나를 선택합니다. (무엇을 선택해도 상관이 없습니다.)

87. Instance State 버튼을 클릭한 다음 **Terminate instance** 를 선택 합니다.

88. 그런 후 Terminate 를 클릭합니다.

잠시 후에, 로드 밸런서 상태 확인 기능이 인스턴스가 응답하지 않는 것을 감지하고 모든 요청을 나머지 인스턴스에 자동으로 라우팅합니다.

89. 웹 브라우저에 있는 Inventory 애플리케이션 탭으로 돌아가서 페이지를 여러 번 새로 고칩니다.

참고로 페이지 하단에 표시된 가용 영역은 동일하게 유지됩니다. 인스턴스에 장애가 발생한 경우에도 애플리케이션은 계속 사용할 수 있습니다.

몇 분 후 Auto Scaling 도 인스턴스 장애를 확인합니다. 2 개의 인스턴스를 실행하도록 Auto Scaling 을 구성했기 때문에 Auto Scaling 이 자동으로 대체 인스턴스를 시작합니다.

90. EC2 의 인스턴스에서 목록에 새 EC2 인스턴스가 나타날 때까지 상단 새로 고침 아이콘을 30 초마다 클릭합니다.

몇 분 후 새 인스턴스에 대한 상태 확인이 정상 상태가 되어야 하며 로드 밸런서는 두 가용 영역 간에 트래픽 전송을 계속합니다. (상태검사의 초기화가 끝나야 동작함)

91. Inventory 애플리케이션 탭으로 돌아가서 페이지를 여러 번 새로 고칩니다. 페이지를 새로 고치면 인스턴스 ID 가 변경되는 것을 볼 수 있습니다.

이를 통해 현재 사용자의 애플리케이션이 고가용성 상태임을 알 수 있습니다.

챌린지: 데이터베이스를 고가용성으로 만들기

참고 이 챌린지 작업은 **선택 사항**이며 실습 시간이 남는 경우(결과 확인까지 약 40 분 소요)에 제공됩니다..

이제 애플리케이션 아키텍처는 고가용성입니다. 하지만, Amazon RDS 데이터베이스는 여전히 하나의 데이터베이스 인스턴스에서만 작동하고 있습니다.

본 과제에서는 데이터베이스를 여러 가용 영역("다중 AZ")에서 실행하여 고가용성으로 만들어보겠습니다.

92. AWS Management Console 의 **Services** 메뉴에서 **Database** > **RDS** 를 클릭합니다.

93. 왼쪽 탐색 창에서 **Databases** 를 클릭합니다.

94. **inventory-db** 식별자를 클릭합니다.

95. **Modify** 를 클릭합니다.

96. 인스턴스 구성의 **DB instance class** 에서 **db.t3.small** 을 선택합니다.

그러면 인스턴스 크기가 두 배로 늘어납니다. (RAM : 1GB → 2GB)

97. **Allocated storage** 에 10 을 입력합니다.

그러면 데이터베이스에 할당된 공간이 두 배로 늘어납니다.

페이지의 다른 옵션을 자유롭게 검토할 수 있지만, 값을 변경하지는 마십시오.

98. **가용성 및 내구성의 Multi-AZ deployment** 아래에서 첫번째 **Create a standby instance** 를 선택합니다.

여러 데이터 센터(가용 영역)에서 실행하도록 데이터베이스를 변환하는 데 필요한 유일한 단계입니다.

이는 데이터베이스가 여러 인스턴스에 *분산* 된다는 뜻은 아닙니다. 오히려 하나의 인스턴스가 *기본* 인스턴스로 모든 요청을 처리합니다. 다른 인스턴스는 *보조* 인스턴스로 시작되고 기본 인스턴스에 장애가 발생할 경우 이를 대신합니다. 애플리케이션은 데이터베이스와 동일한 DNS 이름을 계속 사용하지만 연결은 현재 활성 상태인 데이터베이스 서버로 자동으로 리디렉션합니다.

속성을 변경하여 Amazon EC2 인스턴스를 확장할 수 있는 것과 마찬가지로 Amazon RDS 데이터베이스 인스턴스도 확장할 수 있습니다. 이제 데이터베이스를 확장합니다.

99. 페이지 하단에서 **Continue** 를 클릭합니다.

변경이 성능에 잠재적인 영향을 미칠 수 있는 경고에 유의하십시오. 성능에 미치는 영향으로 인해 예약된 다음 유지 관리 기간 동안 또는 즉시 변경 사항이 발생하도록 예약할 수 있습니다.

100. **Scheduling of modifications** 섹션에서 **Apply immediately** 를 선택합니다.

101. **Modify DB Instance** 를 클릭합니다.

변경 사항이 적용되는 동안 데이터베이스는 *Modifying* 상태가 됩니다. 기다리지 않고(결과 확인까지 약 40 분 소요됩니다) 다음 작업을 계속할 수 있습니다.

챌린지: NAT 게이트웨이를 고가용성으로 만들기

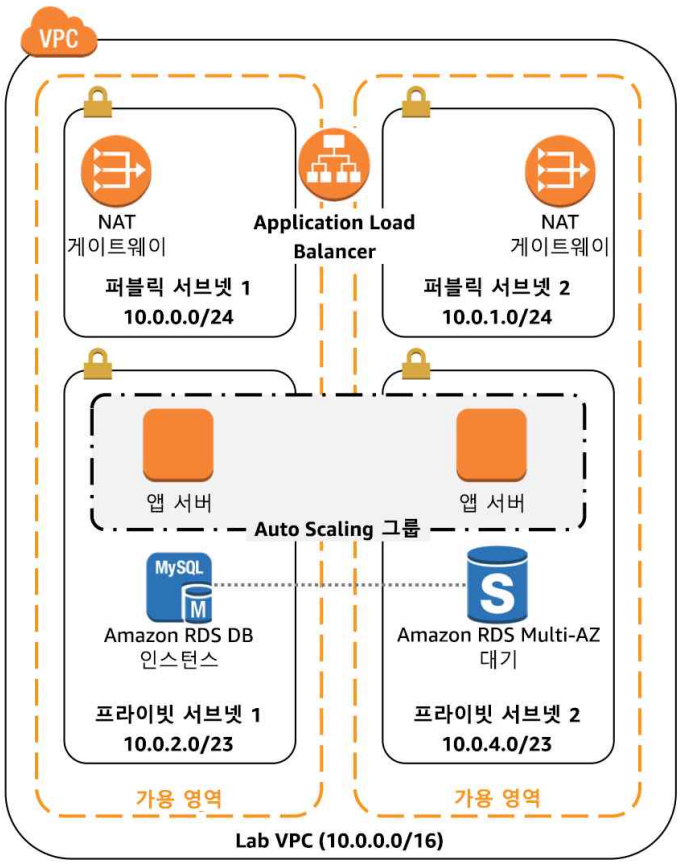
참고 이 챌린지 작업은 **선택 사항**이며 실습 시간이 남는 경우에 제공됩니다

애플리케이션 서버는 프라이빗 서브넷에서 실행되고 있습니다. 인터넷에 액세스해야 하는 경우(예: 데이터 다운로드) 요청은 퍼블릭 서브넷에 있는 *NAT 게이트웨이* 를 통해 리디렉션되어야 합니다.

현재 아키텍처에는 퍼블릭 서브넷 1 에 NAT 게이트웨이 하나만 있습니다. 즉, 가용 영역 1 에서 장애가 발생한 경우 애플리케이션 서버가 인터넷과 통신할 수 없습니다.

이런 문제가 발생한 경우 다른 가용 영역에 있는 또 다른 NAT 게이트웨이를 시작하여 NAT 게이트웨이를 고가용성으로 만들게 됩니다.

다음 다이어그램에 표시된 그 결과 아키텍처는 가용성이 높습니다.



102. **Services** 메뉴에서 **VPC** 를 클릭합니다.

103. 왼쪽 탐색 창에서 **NAT Gateways** 를 클릭합니다.

기존 NAT 게이트웨이가 표시됩니다. 이제 다른 가용 영역에 게이트웨이를 생성합니다.

104. **Create NAT Gateway**를 클릭하고 다음을 구성합니다.

- **Name:** my-nat-gateway
- **Subnet:** 이 지침의 왼쪽에 표시되는 서브넷을 **PublicSubnet2** 로 선택합니다.
- **Allocate Elastic IP address**를 클릭합니다.
- **Create a NAT Gateway**를 클릭합니다.

이제 트래픽을 새 NAT 게이트웨이로 리디렉션하는 프라이빗 서브넷 2 에 새 라우팅 테이블을 생성합니다.

105. 왼쪽 탐색 창에서 **Route Tables**를 클릭합니다.

106. **Create route table**을 클릭하고 다음을 구성합니다.

- **Name tag:** Private Route Table 2
- **VPC:** *Lab VPC*

107. **Create Route Table**를 클릭합니다.

108. **Routes** 탭을 클릭합니다.

현재는 모든 트래픽을 *로컬로* 보내는 하나의 라우팅이 있습니다.

이제 새 NAT 게이트웨이를 통해 인터넷 바운드 트래픽을 보내는 라우팅을 추가합니다.

109. **Edit routes**를 클릭합니다.

110. **Add route**를 클릭하고 다음을 구성합니다.

- **Destination:** 0.0.0.0/0
- **Target:** NAT Gateway > *my-nat-gateway* 선택
- **Save changes**를 클릭합니다.

참고 이 지침의 왼쪽에 표시된 NAT 게이트웨이는 퍼블릭 서브넷 1 에 사용됩니다. 이제 *다른* NAT 게이트웨이를 사용할 라우팅 테이블을 구성하고 있습니다.

111. **Subnet Associations** 탭을 클릭합니다.

112. Edit subnet associations 를 클릭합니다. (명시적 서브넷연결에서)

113. **Private Subnet 2** 를 선택합니다.

114. Save Associations 를 클릭합니다.

그러면 이제 프라이빗 서브넷 2 의 인터넷 바운드 트래픽을 동일한 가용 영역에 있는 NAT 게이트웨이로 보냅니다.

사용자의 NAT 게이트웨이는 이제 고가용성입니다. 한 가용 영역의 장애는 다른 가용 영역의 트래픽에 영향을 미치지 않습니다.

결론

축하합니다! 다음 작업이 성공적으로 완료되었습니다.

- Application Load Balancer 생성
- Amazon EC2 Auto Scaling 그룹 생성
- 보안 그룹을 업데이트하여 3 티어 아키텍처 적용

실습 종료

다음 순서 따라 실습 과정에서 생성된 리소스를 정리하십시오.

1. **EC2 :** Auto Scaling Group 에서 Inventory-ASG 삭제(시간 소요됨, 기다리지 말고 그대로 다음 단계를 수행)
2. **EC2 :** Launch Template 삭제(시작 템플릿 ID 를 선택하고 작업→템플릿삭제)
3. **EC2 :** Load Balancer 삭제 (작업→삭제)
4. **EC2 :** Target Group 삭제 (작업→삭제)

5. **VPC :** NAT Gateway 에서 my-nat-gateway 에 할당된 EIP(탄력적 IP 주소)를 메모장에 복사해 놓고 (나중에 12 번에서 사용됨) my-nat-gateway 를 삭제(작업→NAT 게이트 웨이 삭제)

6. **VPC :** Security Group 에서 Inventory-App 의 Inbound rules 탭에서 인바운드 규칙 편집을 클릭하고 Inbound rules 을 삭제하고 규칙 저장을 클릭한다

7. **VPC :** Security Group 에서 Inventory-LB 삭제 (작업→ 보안 그룹 삭제)

8. **VPC :** Security Group 에서 Inventory-DB 의 Inbound rules 삭제 (6 번 처럼)

   아직 Inventory-DB 는 삭제할 수 없다 (13 번에서 삭제)

9. **VPC :** Security Group 에서 Inventory-App 삭제 (작업→ 보안 그룹 삭제)

10. **VPC :** Route Table 에서 Private Route Table 2 의 라우팅 테이블 ID 를 클릭하고 [서브넷 연결] 탭에서 서브 연결 편집을 클릭하고 Private subnet 2 를 체크 해제한 후 [연결 저장]을 누른다

11. **VPC :** Route Table 에서 Private Route Table 2 삭제 (작업→ 라우팅 테이블 삭제)

12. **VPC :** EIP(탄력적 IP)에서 my-nat-gateway 에 할당되었던(5 번에서 복사해 놓은 IP 와 동일한) 할당된 IPv4 주소를 찾아서 체크해주고 작업 → 탄력적 IP 주소 릴리스를 실행한다

13. **RDS :** 데이터 베이스에서 inventory-db DB 를 삭제 (작업→삭제,시간소요)

    최종 스냅샷 생성 체크 해제, 자동 백업 보존 체크 해제, "인스턴스 삭제시 시스템 ..... 점을 인정합니다"는 체크해준 다음 DB 삭제한다

14. **S3:** Bucket(cf-templates 으로 시작하는 이름)의 내부 파일(json 파일)을 먼저 삭제하고 Bucket 도 삭제한다

15. **CloudFormation** : Stack 에서 arclab4 스택을 삭제한다 (시간 소요)

    이 때 Lab VPC 와 Subnet 도 함께 삭제된다, 만일 삭제되지 않은 리소스들이 있으면 찾아서 모두 지운다 (EC2, DB 스냅샷, NAT Gateway 등은 과금이 되는 서비스)

16. **끝.**