

# AWS 보안

기본 보안

# 아마존 보안

아마존 플랫폼 기반으로 서비스를 운영하면, 다음과 같은 공격 유형들이 많이 발생한다.

1. 아마존 자원(가상머신)에 대한 공격
2. 아마존 계정에 대한 공격(IAM)
3. 설정 문제로 발생한 취약점 공격
4. 애플리케이션에서 사용하는 포트번호 공격
5. 대용량 트래픽 혹은 요청 기반으로 서비스 거부 공격

위와 같은 문제가 발생할 수 있다. 이러한, 공격에 대해서 아마존 클라우드에서 제어가 가능하다. 이 교육에서는 다루는 서비스 부분은 다음과 같다.

# 아마존 기본 보안 서비스

아마존에서 다음과 같은 기본 보안 서비스를 제공하고 있다.

1. AWS Identity/Access Management, IAM
2. Virtual Private Cloud(VPC)
3. Key Management Service(KMS)
4. Web Application Firewall(WAF)
5. Distributed Denial of Service(DDoS)

# 아마존 확장 보안 서비스

위의 서비스 기반으로 손쉽게 인프라 상태를 확인할 수 있도록 하는 확장 서비스도 있다.

1. AWS Lambda automation for services
2. AWS Simple notification Service, SNS

위의 서비스를 통해서 아마존에서 사용하는 자원에 대해서 관리 및 운영이 가능하다.

# 교육 및 랩 목적

이 교육에서는 사용자에게 다음과 같은 경험 제공이 주요 목적이다.

- 아마존에서 제공하는 VPC의 기본 보안 서비스를 살펴보고 직접 구성한다.
- 아마존에서 확장적으로 제공하는 서비스, **KMS/Cloud Watch/Shield/Cloud Trail**사용 및 경험한다.

아마존 기반의 플랫폼 교육은 기존 **온-프레미스**와 다른 부분은 다음과 같다.

- 모든 보안 서비스를 한눈에 관리 및 사용이 가능하다.
- 온-프레미스는 각 분야별로 담당자가 구성 및 관리 하기 때문에 한눈에 확인이 어렵다.
- 인프라에 관련된 모든 사용자들, **C 레벨 관리자/소프트웨어 개발자/인프라 관리자/프로젝트 매니저/서비스분석가**도 해당 분야 지식이 없어도 참여가 가능.

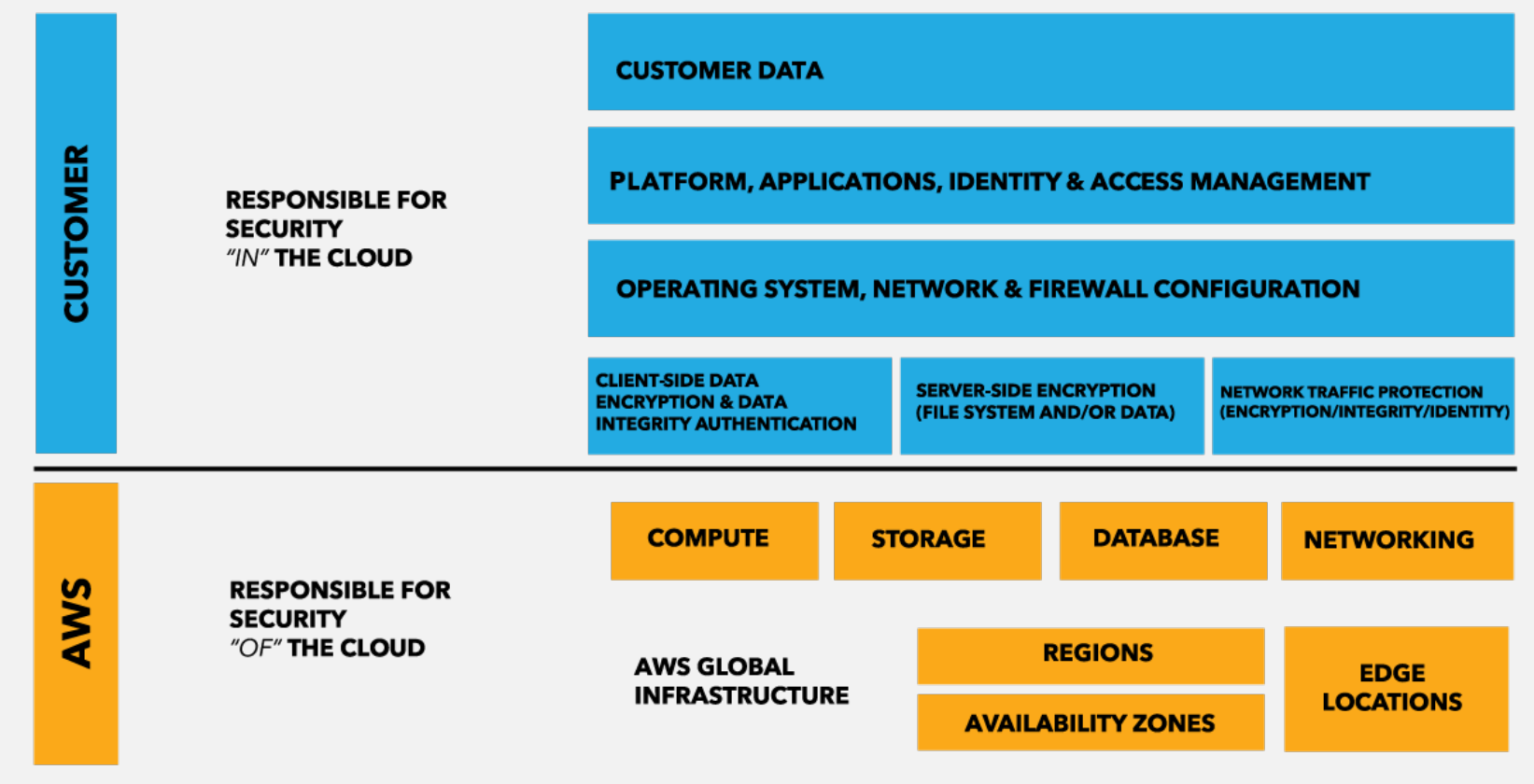
# 아마존 책임 모델

아마존에서 제공하는 보안 모델은 공통 책임 형태의 보안모델이다. 아마존은 아래와 같은 영역에서 사용자에게 보안을 제공한다.

1. 서버가 운영되는 데이터센터에 대한 보안
2. 아마존 클라우드에서 사용하는 워크로드에 대한 예측 및 책임
3. 클라우드에서 저장되는 데이터에 대한 보안 및 안전
4. 데이터 센터의 네트워크(region, availability, zone, edge, location, end point)

이러한 내용들 기준으로 아마존은 "AWS shared security responsibility model"이라고 부른다. 아마존 보안 모델에 대해서는 아래 그림을 참고한다.

# 아마존 책임 모델(공용)



# 아마존 책임 모델(사용자)

위의 그림처럼, 사용자가 책임지는 영역은 다음과 같다.

1. 사용자 데이터(EBS, S3)
2. IAM 및 KMS
3. ACL, Network 혹은 Routing 같은 설정 부분
4. 이외 Endpoint에 대한 보안 연결성 구성

이 부분에 대해서는 사용자가 전적으로 책임을 가져야 하는 부분이다. 해당 부분에 대해서 설정 미숙 혹은 사용 미숙으로 발생한 문제에 대해서는 비용을 포함한 전체적인 부분에 대해서 사용자가 책임진다.



# 아마존 책임 모델(사용자)

위의 내용을 그림으로 표현하면 다음과 같다.



# 아마존 책임 모델(아마존)

위의 그림처럼, 아마존에 책임지는 영역은 다음과 같다.

1. 하드웨어 및 소프트웨어 관련된 운영책임
2. 워크로드에 따른 하드웨어 확충 및 안전성
3. 국가별 관련된 엣지 및 엔드 포인트에 대한 보안 및 접근성 제공
4. IDC에 대한 보안 및 시설 유지

이 부분에 대해서는 아마존이 전적으로 문제가 발생 시 책임을 진다. 하지만, 정치적 및 특별한 기후적인 이변으로 발생한 문제에 대해서는 면책 특권을 가진다.

# 자원 관리 영역

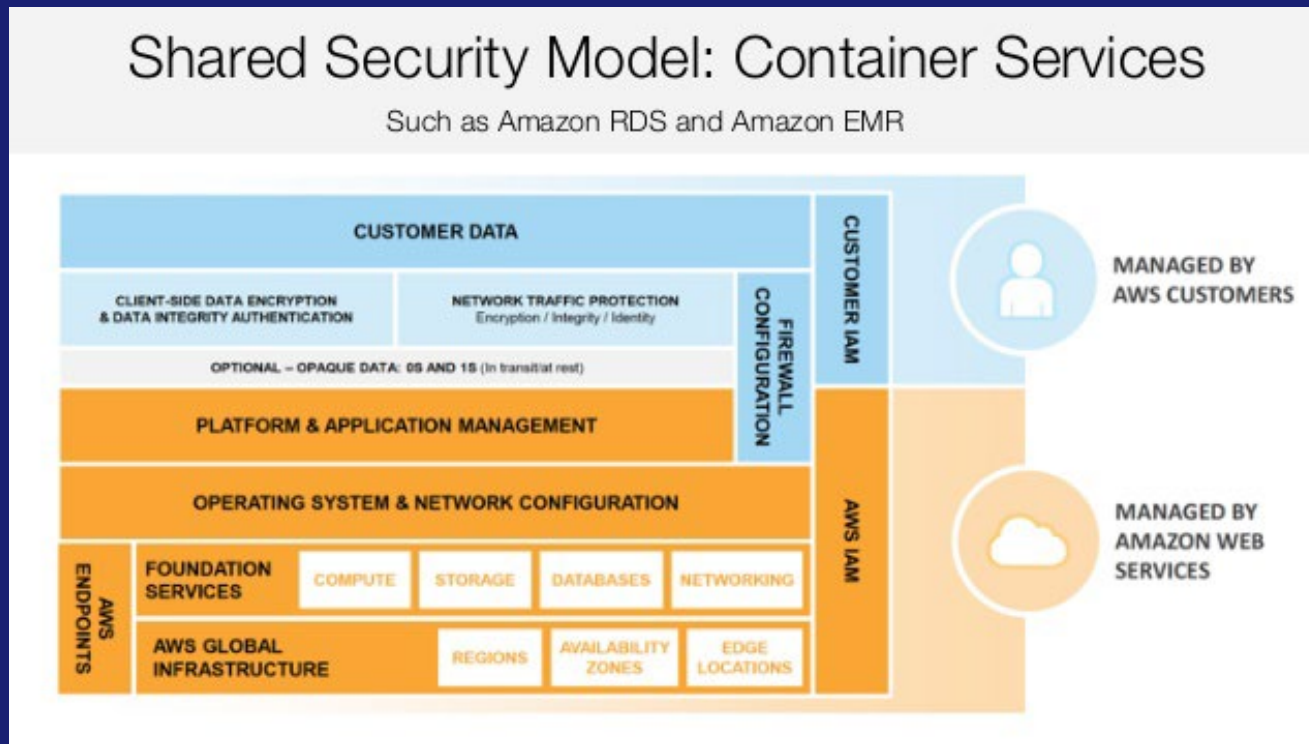
위의 **아마존 책임 모델**과 동일하게, 아마존 및 사용자의 관리 영역이 구분이 되어 있다. 사용자는 웹 인터페이스 혹은 AWS CLI를 통해서 자원들을 적절하게 올바른 설정으로 구성 및 운영해야 하는 책임이 있다.

**아마존**은 사용자가 자원을 잘 접근 및 사용이 가능하도록 API를 제공해야 한다. API는 보통 **엔드 포인트**라고 부르며, 이는 각 서비스(자원)/클러스터/지역별로 구성이 되어 있다.

아래 그림에서 해당 자원 관리 영역에 대한 표현이 되어 있다.

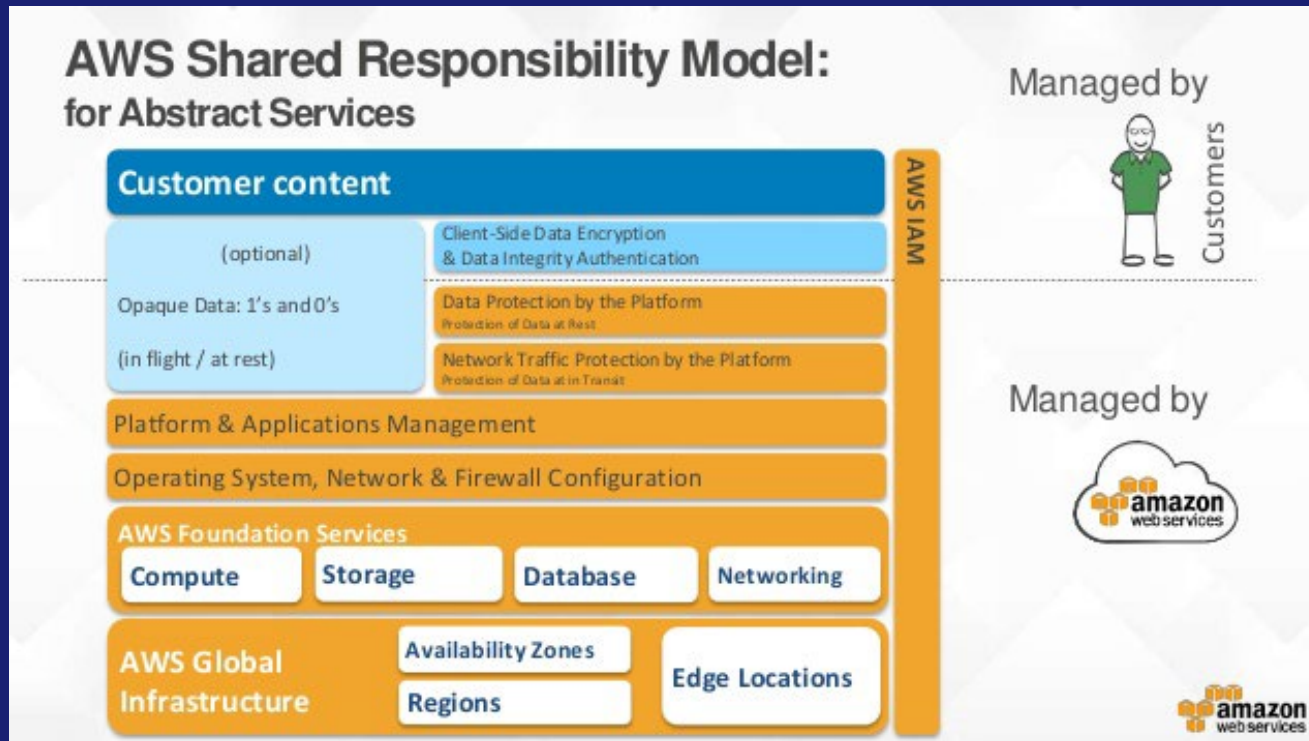
# 자원 관리 영역(전체)

아마존에 제공하는 자원 관리 영역이다. 여기서는 RDS, ERM를 예제로 설명한다.



# 아마존 책임 모델(추상)

아마존의 모든 서비스는 HTTPS기반의 API를 통해서 구성이 되며, 이를 통해서 S3, EC2와 같은 서비스를 구축한다. 아마존 기본 서비스 및 보안 서비스는 전부 API를 통해서 생성 및 구성이 된다.



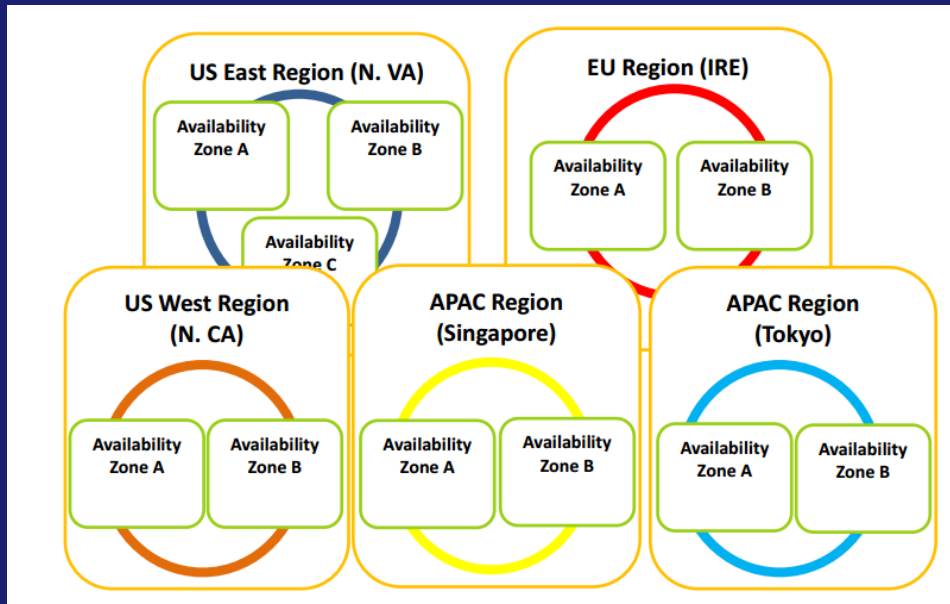
# 아마존 제공 영역

아마존은 다음과 같은 서비스 영역을 사용자에게 제공한다. 아마존은 인프라 및 글로벌 리전 서비스 및 플랫폼 제공을 한다. 모든 글로벌 서비스에 최소한 기본 서비스는 제공하며, 이를 통해서 각 국가에 서비스를 제공한다.



# 아마존 리전 서비스

아마존 리전 서비스는 말 그대로 대륙 혹은 국가별로 제공하는 인프라 서비스이다. 각 리전에는 사용 가능한 가용 영역이 있으며, 이 영역을 통해서 서비스를 제공한다. 각 리전은 각기 다른 인프라를 가지고 있으며, 이들은 서로 인프라 공유가 불가능하다. 서로 자원을 연결하기 위해서는 네트워크를 통해서 연결해야 한다.



# 아마존 책임 모델

AWS	Customer
Facility operations	Choice of guest operating system
Physical security	Configuring application options
Physical infrastructure	AWS account management
Network infrastructure	Configuring security groups (firewall)
Virtualization infrastructure	ACL
Hardware lifecycle management	IAM



# 아마존 IAM

Business requirement	Proposed design	Comments
Centralized security management 중앙에서 아마존 계정 관리	한 개의 아마존 계정	중앙에서 정보 및 보안 관리하기 때문에 관리에 대해서 오버헤드를 최소화로 가져간다.
제품별로 다른 환경을 가져간다. 예를 들어서 개발/테스트/제품으로 나눈다.	3개의 아마존 계정	각 계정별로 제품/개발/테스트 환경으로 나누어서 할당 및 관리 한다.
여러 자치부서(Multiple autonomous departments)	여러 개의 아마존 계정	하나의 계정이 여러 부서를 관리한다. 중앙 관리자 계정에서 각기 단독 계정을 생성한 후 퍼미션을 할당하여 사용 범위를 제한한다. 이를 통해서 비용 발생 범주를 제약 및 제한 할 수 있다.
중앙 집중식 보안관리, 여러 개의 프로젝트를 독립적/자율적으로 관리한다.	여러 개의 아마존 계정	도메인 이름 서비스, 사용자 데이터베이스 등과 같은 공유 프로젝트 리소스에 대해 하나의 AWS 계정을 생성한다. 각 자율 독립 프로젝트에 대해 하나의 AWS 계정을 생성하고 세부적인 수준에서 권한을 부여한다.

# IAM

위의 자원들을 제한 및 제어하기 위해서 IAM(Identity and Access Management)를 통해서 관리한다. 모든 자원들은 IAM를 통해서 특정 사용자에게 특정 자원에 대한 권한을 허용한다.

IAM은 모든 할당 및 요청들을 API를 통해서 할당 및 제어한다.

AWS	사용자
시설운영	게스트 가상머신 선택
물리적 보안	애플리케이션 옵션 설정
물리적 인프라	AWS 계정 관리자
네트워크 인프라	AWS 보안 그룹
가상화 인프라	ACL
하드웨어 라이프 사이클 관리자	IAM

# IAM

AWS에서 제공하는 IAM인증은 다음과 같다.

1. 비밀번호
2. 멀티 팩터 인증
3. 접근 키(access key)
4. 쌍키(key pair)
5. X.509 인증서

아마존 사용자는 위의 방법을 통해서 접근하며, 사용자들은 다음과 같은 방법으로 아마존 인프라에 API를 통해서 접근이 가능하다.

- CLI
- API
- AWS IAM(DashBoard)

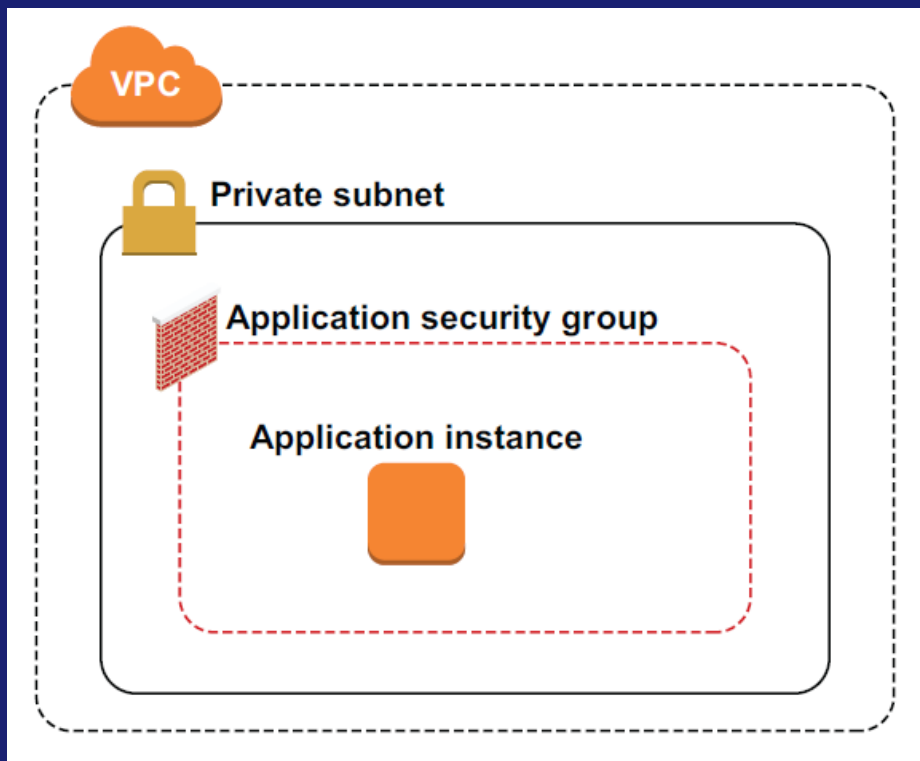
아마존에서 발생하는 로그는 각기 서비스별로 발생 및 저장이 되며, 인프라에서 발생하는 로그는 AWS CloudTrail logs이나 Cloud Watch를 통해서 확인이 가능하다.

# 아마존 보안

인증

# 랩 구성

위의 랩을 실행하는 경우 아래와 같이 구성이 된다.



# 시작 전

혹은 아래 대시보드 기반으로 진행한다. 다만, 대시보드는 복잡한 구성을 가지고 있기 때문에, 천천히 하나씩 생성하면서 진행한다.

# IAM 소개

아마존에서 제공하는 제일 기본적인 RBAC(Role Base Access Control)기능이다. IAM를 통해서 사용자를 생성하여, VPC, RDS, S3와 같은 자원을 사용한다. 아마존의 모든 자원은 IAM를 통해서 접근이 되어야 하기 때문에, 올바르게 설정이 되지 않는 경우, 접근이 차단 혹은 자원 오남용이 발생한다.

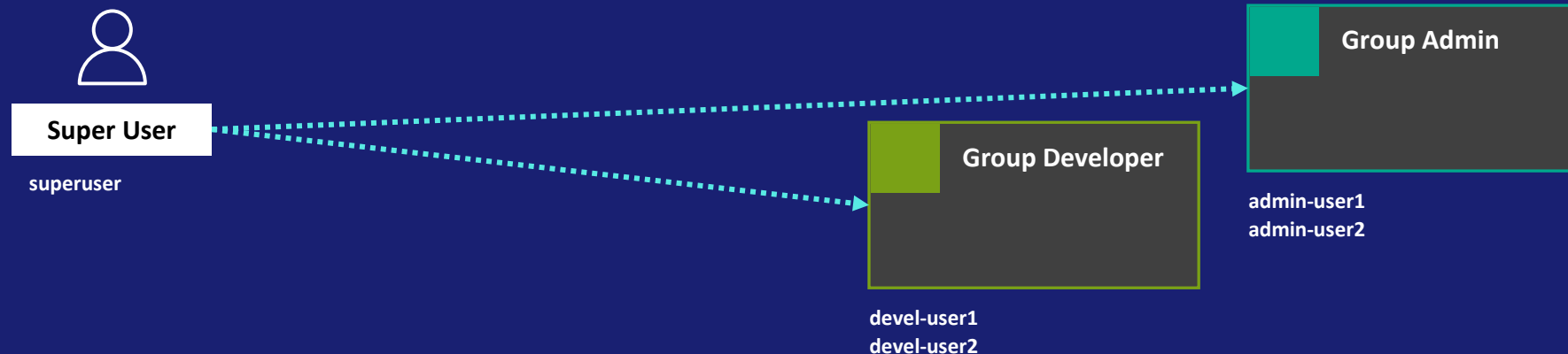
IAM에 다루는 영역은 다음과 같다.

- 기능 및 도구
- 인증 및 권한 부여
- 아마존 증명서
- IAM 제한 사항

# IAM 설명

기본적으로 IAM으로 생성 및 관리하며 이를 통해서 제어를 한다. 처음 생성한 IAM계정은 최고 사용자(Super User) 권한을 가지고 있으며, 이를 통해서 일반 사용자 계정이 생성 및 설정이 가능하다. 예를 들어서 다음과 같다.

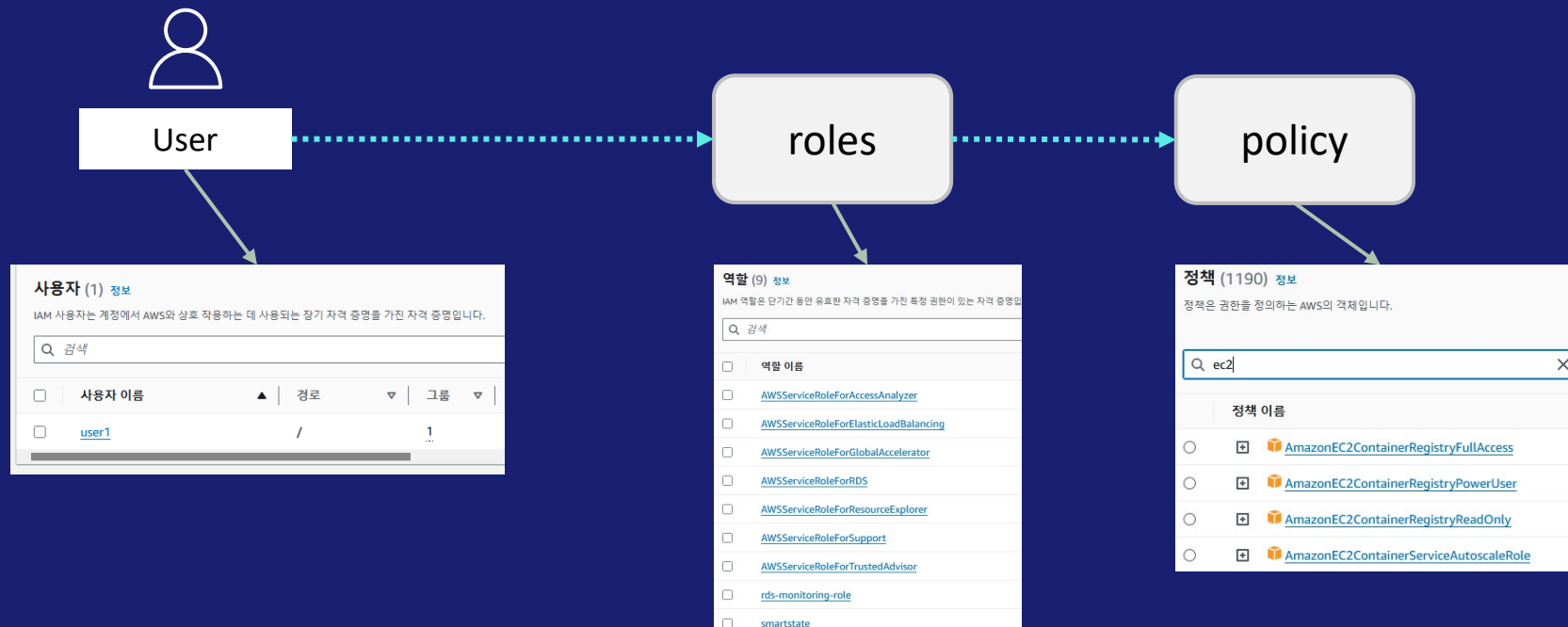
계정	권한	설명
superuser	모든 권한	IAM에 모든 권한을 가지고 있는 계정. 이 계정을 통해서 기능 및 역할을 설정한다.
admin-user1/2	특정 자원/VPC생성 및 구성 권한	특정 계정안에서 AWS 자원 및 VPC 생성 및 설정 권한이 있다.
devel-user1/2	접근 권한	개발자 권한. 특정 자원에서 확인 및 접근만 가능하며 생성 및 제거는 불가능하다.





# IAM 역할/정책

IAM 역할은 메타정보이다. IAM에서 역할을 생성하면 역할은 최소 한 개의 정책이 할당이 된다. 각 기능은 IAM에서 아래와 같이 확인이 가능하다. 사용자를 생성 후, 해당 사용자에게 특정 자원에 대해서 접근 할 수 있도록 역할(role)를 생성 후, 정책(policy)를 할당한다.



# 역할

역할은 앞서 이야기 하였지만, 이름만 존재하며 각 서비스에서 제공하는 JSON형태의 정책에 따라서 사용자 접근을 허용한다. 확인 및 생성하기 위해서 IAM에서 다음과 같은 메뉴로 접근한다.

▼ 액세스 관리

사용자 그룹

사용자

**역할**

정책

자격 증명 공급자

계정 설정

IAM > 역할

역할 (9) 정보

IAM 역할은 단기간 동안 유효한 자격 증명을 가진 특정 권한이 있는 자격 증명입니다. 신뢰할 수 있는 개체가 역할을 받을 수 있습니다.

↺ 1 ↻

🔍

<input type="checkbox"/>	역할 이름	신뢰할 수 있는 개체	마지막 활동
<input type="checkbox"/>	<a href="#">AWSServiceRoleForAccessAnalyzer</a>	AWS 서비스: access-analyzer (서비스)	5시간 전
<input type="checkbox"/>	<a href="#">AWSServiceRoleForElasticLoadBalancing</a>	AWS 서비스: elasticloadbalancing (서비스)	351일 전
<input type="checkbox"/>	<a href="#">AWSServiceRoleForGlobalAccelerator</a>	AWS 서비스: globalaccelerator (서비스)	-
<input type="checkbox"/>	<a href="#">AWSServiceRoleForRDS</a>	AWS 서비스: rds (서비스 연결 역할)	1시간 전
<input type="checkbox"/>	<a href="#">AWSServiceRoleForResourceExplorer</a>	AWS 서비스: resource-explorer-2 (서비스)	40분 전
<input type="checkbox"/>	<a href="#">AWSServiceRoleForSupport</a>	AWS 서비스: support (서비스 연결 역할)	-
<input type="checkbox"/>	<a href="#">AWSServiceRoleForTrustedAdvisor</a>	AWS 서비스: trustedadvisor (서비스)	-
<input type="checkbox"/>	<a href="#">rds-monitoring-role</a>	AWS 서비스: monitoring.rds	-
<input type="checkbox"/>	<a href="#">smartstate</a>	AWS 서비스: ec2	-

↻

삭제

**역할 생성**

# 역할

엔터티 유형에 따라서 역할은 달라진다. 여기서 말하는 엔터티는 어떠한 인증 접근 방식에 사용할지 명시하는 부분이다. 보통은 AWS 서비스 혹은 AWS 계정으로 역할을 구성한다. 기본적으로 AWS에서 제공하는 서비스 기반으로 구성하기 때문에 AWS서비스를 선택한다.

[IAM](#) > [역할](#) > 역할 생성

1단계  
신뢰할 수 있는 엔터티 선택

2단계  
권한 추가

3단계  
이름 지정, 검토 및 생성

신뢰할 수 있는 엔터티 선택 정보

신뢰할 수 있는 엔터티 유형

☒ **AWS 서비스**  
EC2, Lambda 등의 AWS 서비스가 이 계정에서 작업을 수행하도록 허용합니다.

☐ **AWS 계정**  
사용자 또는 서드 파티에 속한 다른 AWS 계정의 엔터티가 이 계정에서 작업을 수행하도록 허용합니다.

☐ **웹 자격 증명**  
지정된 외부 웹 자격 증명 공급자와 연동된 사용자가 이 역할을 맡아 이 계정에서 작업을 수행하도록 허용합니다.

☐ **SAML 2.0 연동**  
기업 디렉터리에서 SAML 2.0과 연동된 사용자가 이 계정에서 작업을 수행할 수 있도록 허용합니다.

☐ **사용자 지정 신뢰 정책**  
다른 사용자가 이 계정에서 작업을 수행할 수 있도록 사용자 지정 신뢰 정책을 생성합니다.

사용 사례  
EC2, Lambda 등의 AWS 서비스가 이 계정에서 작업을 수행하도록 허용합니다.

서비스 또는 사용 사례  

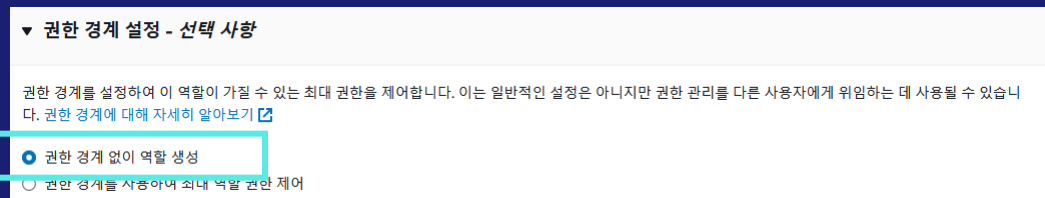
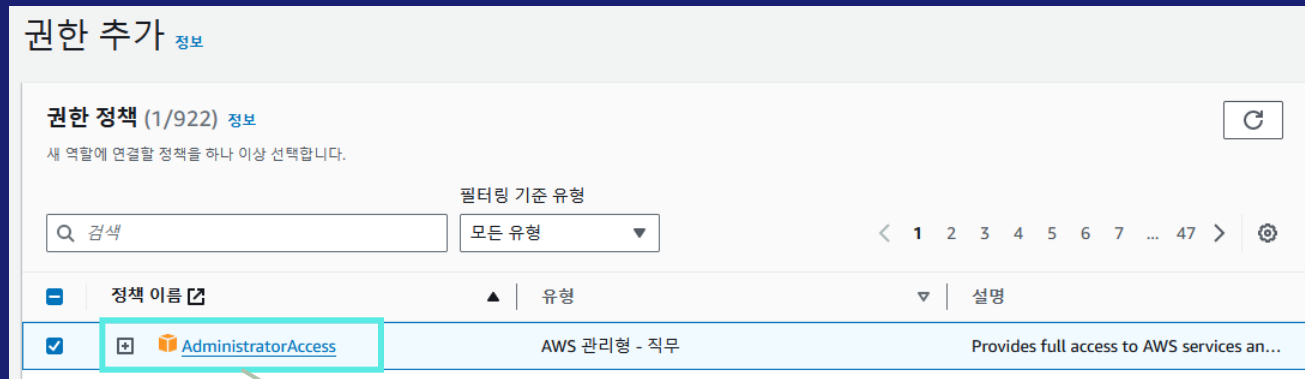
서비스 또는 사용 사례 선택

취소

다음

# 정책할당

역할을 생성하기 위해서 최소 하나의 권한이 있어야 한다. EC2를 사용하기 위해서 "Administrator Access"를 선택한다.



# 역할확인

역할을 생성하기 위해서 간단하게 설명을 적고 생성한다. 테스트 용도이니 기본값 그대로 사용한다. 생성이 되면 다음처럼 목록에 "ec2-admin"라고 출력이 된다.

역할 세부 정보

**역할 이름**  
이 역할을 식별하는 의미 있는 이름을 입력합니다.

ec2-admin

최대 64자입니다. 영숫자 및 '+', '@', '\_' 문자를 사용하세요.

**설명**  
이 역할에 대하여 간단한 설명을 추가합니다.

Allows EC2 instances to call AWS services on your behalf.

최대 문자 수: 1000. 문자(A~Z 및 a~z), 숫자(0~9), 탭, 새 줄 또는 다음 문자 중 하나를 사용합니다: \_+=, . @-/[()!#\$%^&\*():"'<>`

<input type="checkbox"/>	<a href="#">ec2-admin</a>	AWS 서비스: ec2	-
--------------------------	---------------------------	--------------	---

# 사용자 확인

이젠 역할 및 정책을 생성하였기 때문에, 이를 실제로 사용할 사용자에게 할당한다.

사용자 세부 정보 지정

사용자 세부 정보

사용자 이름  
ec2-admin

☒ AWS Management Console에 대한 사용자 액세스 권한 제공 - 선택 사항  
사용자에게 콘솔 액세스 권한을 제공하는 경우 IAM Identity Center에서 액세스를 관리하는 것은 **모범 사례**입니다.

사용자에게 콘솔 액세스 권한을 제공하고 있습니까?  
사용자 유형

☐ Identity Center에서 사용자 지정 - 권장  
Identity Center를 사용하여 사용자에게 콘솔 액세스 권한을 제공하는 것이 좋습니다. Identity Center를 사용하면 AWS 계정 및 클라우드 애플리케이션에 대한 사용자 액세스를 중앙에서 관리할 수 있습니다.

☒ IAM 사용자를 생성하고 싶음  
액세스 키, AWS CodeCommit이나 Amazon Keyspaces에 대한 서비스별 보안 인증 정보 또는 비상 계정 액세스를 위한 백업 보안 인증 정보를 통해 프로그래밍 방식 액세스를 활성화해야 하는 경우에만 IAM 사용자를 생성하는 것이 좋습니다.

콘솔 암호

☒ 자동 생성된 암호  
사용자를 생성할 때 암호를 자동으로 생성합니다.

☐ 사용자 지정 암호  
사용자의 사용자 지정 암호를 입력합니다.

• 8자 이상이어야 합니다.  
• 다음 문자 유형 중 세 가지 이상을 조합하여 포함해야 합니다. 대문자(A-Z), 소문자(a-z), 숫자(0-9), 기호 ! @ # \$ % ^ & \* ( ) \_ + - (하이픈) = [ ] { } ' "

☐ 암호 표시

☐ 사용자는 다음 로그인 시 새 암호를 생성해야 합니다 - 권장  
사용자는 IAMUserChangePassword 정책을 자동으로 가져와 암호를 변경할 수 있도록 허용합니다.

이 IAM 사용자를 생성한 후 액세스 키 또는 AWS CodeCommit이나 Amazon Keyspaces에 대한 서비스별 보안 인증 정보를 통해 프로그래밍 방식 액세스를 생성할 수 있습니다. [자세히 알아보기](#)

취소 다음

이 옵션은 랩 진행 시, 사용하지 않아도 됩니다.

# 그룹생성

바로 사용자에게 권한을 할당하지 말고, 그룹을 생성 후 할당한다.

### 권한 설정

기존 그룹에 사용자를 추가하거나 새 그룹을 생성합니다. 직무별로 사용자의 권한을 관리하려면 그룹을 사용하는 것이 좋습니다. [자세히 알아보기](#)

#### 권한 옵션

☒ **그룹에 사용자 추가**  
기존 그룹에 사용자를 추가하거나 새 그룹을 생성합니다. 그룹을 사용하여 직무별로 사용자 권한을 관리하는 것이 좋습니다.

☐ **권한 복사**  
기존 사용자의 모든 그룹 멤버십, 연결된 관리형 정책 및 인라인 정책을 복사합니다.

☐ **직접 정책 연결**  
관리형 정책을 사용자에게 직접 연결합니다. 사용자에게 연결하는 대신, 정책을 그룹에 연결한 후 사용자를 적절한 그룹에 추가하는 것이 좋습니다.

#### 사용자 그룹 (1)

<input type="checkbox"/>	그룹 이름	사용자	연결된 정책	생성됨
<input type="checkbox"/>	<a href="#">lab-windows</a>	1	<a href="#">AmazonEC2ReadOnlyAccess</a>	2024-03-13 (1개월 전)

▶ 권한 경계 설정 - 선택 사항

### 사용자 그룹 생성

사용자 그룹을 생성하고 그룹에 연결할 정책을 선택합니다. 그룹을 사용하여 직무, AWS 서비스 액세스 또는 사용자 지정 권한별로 사용자 권한을 관리하는 것이 좋습니다. [자세히 알아보기](#)

사용자 그룹 이름  
이 그룹을 식별하는 의미 있는 이름을 입력합니다.

최대 128자입니다. 영숫자 및 '+', '@', '-' 문자를 사용하세요.

#### 권한 정책 (1/924)

필터링 기준 유형: 모든 유형 31 개 일치

<input type="checkbox"/>	정책 이름	유형	다음...	설명
<input type="checkbox"/>	<a href="#">AmazonEC2ContainerRegistryFullAccess</a>	AWS 관리형	없음	Provi
<input type="checkbox"/>	<a href="#">AmazonEC2ContainerRegistryPowerUser</a>	AWS 관리형	없음	Provi
<input type="checkbox"/>	<a href="#">AmazonEC2ContainerRegistryReadOnly</a>	AWS 관리형	없음	Provi
<input type="checkbox"/>	<a href="#">AmazonEC2ContainerServiceAutoscaleRole</a>	AWS 관리형	없음	Polic
<input type="checkbox"/>	<a href="#">AmazonEC2ContainerServiceEventsRole</a>	AWS 관리형	없음	Polic
<input type="checkbox"/>	<a href="#">AmazonEC2ContainerServiceforEC2Role</a>	AWS 관리형	없음	Defa
<input type="checkbox"/>	<a href="#">AmazonEC2ContainerServiceRole</a>	AWS 관리형	없음	Defa
<input checked="" type="checkbox"/>	<a href="#">AmazonEC2FullAccess</a>	AWS 관리형	권한 정...	Provi

# 그룹 및 사용자 확인

생성이 완료가 되면, 다음처럼 사용자를 그룹에 할당해야 한다. 아래 admin-group를 선택한다.

사용자 그룹 (2)					그룹 생성
<input type="text" value="검색"/>					< 1 > ⚙
<input type="checkbox"/>	그룹 이름	▲	사용자 ▼	연결된 정책	생성됨 ▼
<input type="checkbox"/>	admin-group		0	<a href="#">AmazonEC2FullAccess</a>	2024-04-28 (지금)
<input type="checkbox"/>	lab-windows		1	<a href="#">AmazonEC2ReadOnlyAccess</a>	2024-03-13 (1개월 전)



# 그룹 및 사용자 확인

선택 후, 다음 누르면 다음처럼 화면이 나온다.

### 검토 및 생성


선택 사항을 검토합니다. 사용자를 생성한 후 자동 생성된 암호를 보고 다운로드할 수 있습니다(활성화된 경우).

#### 사용자 세부 정보

사용자 이름 admin-user1	콘솔 암호 유형 Custom password	암호 재설정 필요 예
-----------------------	-----------------------------	----------------

#### 권한 요약

< 1 >

이름 	▲	유형	▼	다음과 같이 사용	▼
<a href="#">admin-group</a>		그룹		권한 그룹	
<a href="#">IAMUserChangePassword</a>		AWS 관리형		권한 정책	

# 사용자 로그인 확인

생성이 완료가 되면, 아래처럼 암호와 주소가 화면에 출력이 된다. ".csv"파일을 다운로드 받는다. 아래 "콘솔 로그인 URL"에 접근하면 아래처럼 로그인 화면이 출력된다.

### 암호 검색

아래에서 사용자의 암호를 보고 다운로드하거나 AWS 관리 콘솔에 로그인하기 위한 사용자 지침을 이메일로 보낼 수 있습니다. 지금이 이 암호를 확인 및 다운로드할 수 있는 유일한 시간입니다.

콘솔 로그인 세부 정보

이메일 로그인 지침

콘솔 로그인 URL  
https://069515814364.signin.aws.amazon.com/console

사용자 이름  
admin-user1

콘솔 암호  
\*\*\*\*\* 표시

취소

.csv 파일 다운로드

사용자 목록으로 돌아가기

# 사용자 접근

생성한 아이디로 접근 하면, 아래와 같이 화면에 출력이 된다.

AWS 계정 069515814364

IAM 사용자 이름 admin-user1

이전 비밀번호

새 비밀번호

새 비밀번호 재입력

비밀번호 변경 확인

[루트 사용자 이메일을 사용하여 로그인](#)

# CLI 시작 전

사용자를 미리 두 개를 생성 합니다. 생성 방법은 위에서 다룬 내용을 가지고 진행 합니다.

이름	역할
ec2-admin	AdministratorAccess
ec2-user	PowerUserAccess

이 두 개의 계정을 가지고 랩을 진행 합니다. 생성 및 사용 방법은 아래에서 확인이 가능 합니다. 가급적이면, 루트 계정으로 작업은 권장하지 않습니다.

```
$ aws configure      ## ec2-admin
$ aws configure --profile ec2-user      ## ec2-user
> set AWS_PROFILE=ec2-user
> $env:AWS_PROFILE = ec2-user
$ aws configure list
$ export AWS_PROFILE=ec2-user
```

# CLI 시작 전

AWS명령어로 진행하기 전, CLI를 사용할 수 있도록 로그인을 해야 한다. 아래와 같이 시스템에 로그인이 가능하다. 앞에서 만든 계정으로 미리 접근한다.

```
$ aws configure
```

```
AWS Access Key ID [None]: AKIARAL~V
```

```
AWS Secret Access Key [None]: t0!!!@@#eht9G0l+xNR1iNPGkxZUd/F!
```

```
Default region name [None]: ap-northeast-2
```

```
Default output format [None]: yaml
```

```
$ aws sts get-session-token
```

위의 정보는 다운로드 받은 .csv파일에서 확인이 가능하다.

```
Credentials:
```

```
AccessKeyId: AKIARAL~V
```

```
Expiration: '2024-10-13T01:20:50+00:00'
```

```
SecretAccessKey: t0!!!@@#eht9G0l+xNR1iNPGkxZUd/F!
```

# CLI 시작 전(랩 생성 예제)

앞서 진행하기 전에 간단하게 자원 생성을 해야 한다. 깃헙에서 내려받기 하거나 혹은 아래처럼 작성 후 실행한다. build-lab.yaml이라는 파일로 작성한다.

```
AWSTemplateFormatVersion: '2010-09-09'
```

```
Resources:
```

```
VPC:
```

```
  Type: AWS::EC2::VPC
```

```
  Properties:
```

```
    CidrBlock: 172.16.0.0/16
```

```
    EnableDnsSupport: true
```

```
    EnableDnsHostnames: true
```

```
PrivateSubnet:
```

```
  Type: AWS::EC2::Subnet
```

```
  Properties:
```

```
    VpcId: !Ref VPC
```

```
    CidrBlock: 172.16.0.0/24
```

```
    AvailabilityZone: ap-northeast-2a
```

VPC생성. 네트워크 대역 변경이 가능하다.

가상머신이 사용 예정인 내부 네트워크 생성.  
생성 위치는 APAC-SEOUL로 한다.

# CLI 시작 전(랩 생성 예제)

위의 내용에 계속 이어서... 혹은 github에서 내려받기가 가능하다.

```
ApplicationSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: For Sample Internal Application
    VpcId: !Ref VPC
```

보안 그룹을 생성한다.

```
ApplicationInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: ami-0ee82191e264e07cc
    InstanceType: t2.micro
    KeyName: mykeypair
    SubnetId: !Ref PrivateSubnet
    SecurityGroupIds:
      - !Ref ApplicationSecurityGroup
```

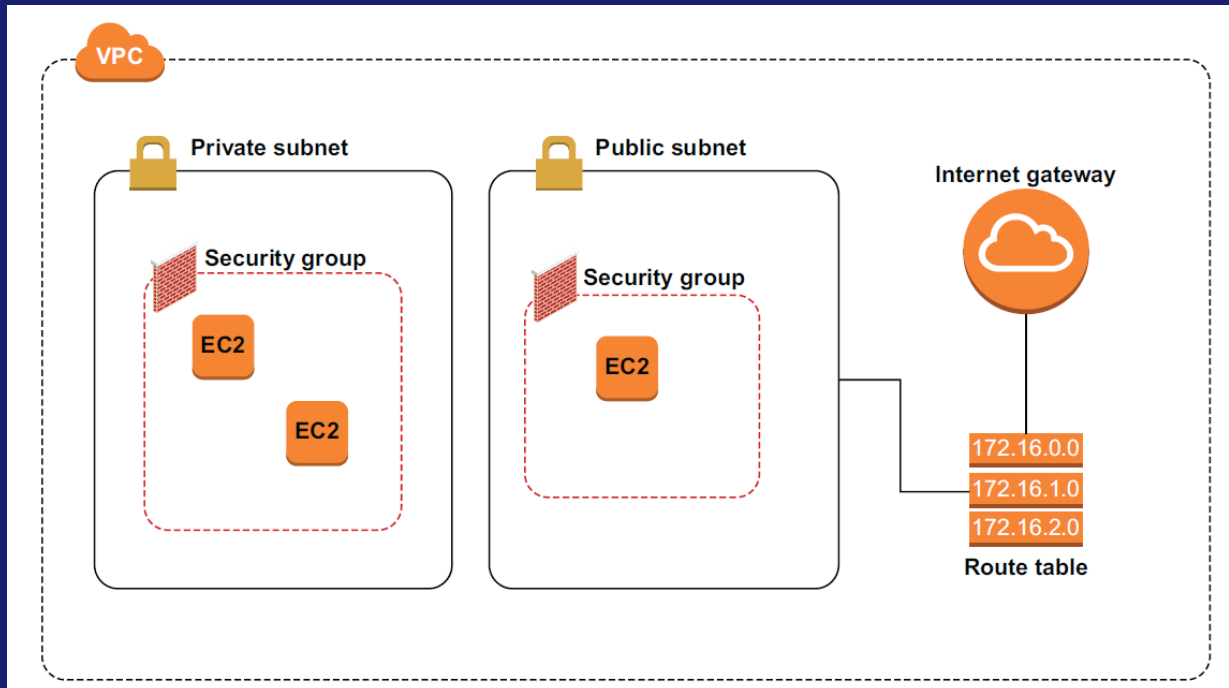
가상머신 생성 시, 사용할 네트워크/이미지/형식 및 보안그룹을 선택한다.

```
ApplicationKeyPair:
  Type: AWS::EC2::KeyPair
  Properties:
    KeyName: mykeypair
```

가상머신 접근 시, 사용할 보안키를 명시 및 생성합니다.

# CLI 시작 전(CLI LAB DESIGN)

CLI에서 사용할 랩은 다음과 같이 구성한다. 다만, 이번에는 대시보드에서 생성하는 게 아니라, CLI를 통해서 구성한다. 대시보드에는 다중으로 구성하였지만, 여기서는 설명을 쉽게 하기 위해서 단순한 VPC기반으로 구성한다.





# CLI 시작 전

전체 랩 파일은 깃헙에서 내려받기 합니다. SSH키는 SSM에서 내려받기 합니다. 명령어는 아래 내용을 참조 합니다.

```
# aws ec2 describe-key-pairs --filters Name=key-name,Values=mykeypair --  
query KeyPairs[*].KeyId --output text  
# aws ssm get-parameter --name /ec2/keypair/key-0cf9b88d1f22bb99b --with-  
decryption --query Parameter.Value --output text > mykeypair.pem
```

# CLI 시작 전

# CLI(사용자 생성)

웹 브라우저로 작업하는 경우, 안전하게 계정 관리 및 운영, 그리고 자동화 하기가 어렵다. 이러한 이유로, CLI기반으로 사용자 생성 및 구성을 권장한다.

```
$ aws iam create-user --user-name test-example
$ aws iam create-access-key --user-name test-example
$ aws iam create-user --user-name foo-user
$ aws iam create-login-profile --user-name foo-user --password Foo-
Password-1 --password-reset-required
```

# CLI(사용자 생성)

testuser를 생성한다.

```
$ aws iam create-user --user-name testuser  
$ aws iam create-access-key --user-name testuser  
$ aws iam create-login-profile --user-name foo-user --password Testuser1
```

# CLI(사용자 제거)

사용자 제거는 delete-user명령어로 가능하다.

```
$ aws iam list-access-keys --profile test-example  
$ aws iam delete-access-key --user-name test-example --profile test-example  
$ aws iam delete-user --user-name test-example
```

# CLI(정책)

보안정책은 다음과 같이 CLI에 적용이 가능하다. 적용하기 위해서는 JSON이나 혹은 YAML형태로 작성해야 한다. 아래는 정책파일 예제이다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "dynamodb:ListTables",
      "Resource": "*"
    }
  ]
}
```

명시된 권한(Permission)를 허용한다.

특정 서비스에서 특정 행동(action)를 허용한다.

특정 서비스에서 접근 자원만 허용한다.  
여기서는 ListTables로 되어 있기 때문에 \*으로 표시한다.

# CLI(정책)

액션(Action)블록에는 다음과 같이 적용이 가능하다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "dynamodb:ListTables",
      "Action": "*",
      "Action": "ec2:List",
      "Action": "s3:*",
      "Resource": "*"
    }
  ]
}
```

위의 내용을 아래처럼, 복수로 설정 및 할당이 가능하다.

# CLI(정책)

혹은 특정 자원(resource)에 대해서 접근을 제한하려고 하면 다음과 같이 설정이 가능하다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::my-bucket-name/*"
    }
  ]
}
```

**Amazon Resource Names (ARNs)** uniquely identify AWS resources. We require an ARN when you need to specify a resource unambiguously across all of AWS, such as in IAM policies, Amazon Relational Database Service (Amazon RDS) tags, and API calls.

```
arn:aws:iam::123456789012:user/Development/product_1234/*
```



# CLI(정책)

복수로 적용하기 위해서는 다음과 같이 작성한다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "NotAction": "ec2:TerminateInstances",
      "Resource": "*"
    }
  ]
}
```

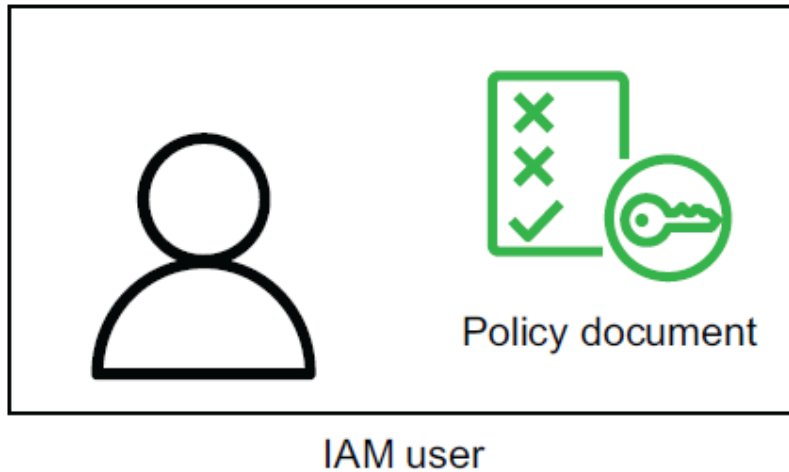
일반적으로 적용 예제.

특정 행동만 제한이 가능하다.

# CLI(정책)

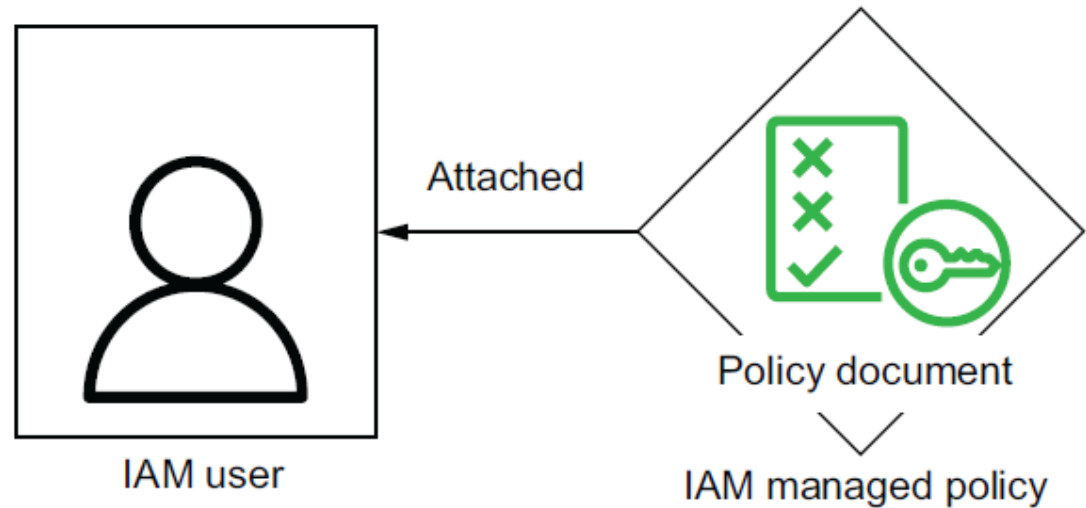
## Inline policy

Inline policies are applied directly to and are a property of the user.



## Managed policy

Managed policies are their own resource. They can be attached to any number of users.



# CLI(정책)

정책을 진행하기 전, "policy1.json"파일을 생성한다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "ec2:TerminateInstances",
      "Resource": "*"
    }
  ]
}
```

# CLI(정책)

사용자에게 정책을 다음과 같은 명령어로 할당이 가능하다. 아래는 inline형식으로 정책을 할당한다.

```
# aws iam create-user --user-name ec2-user
# aws iam put-user-policy --user-name ec2-user --policy-name sampleinline-
policy --policy-document file://policy1.json
# aws iam list-user-policies --user-name ec2-user
# aws iam get-user-policy --policy-name sampleinline-policy --user-name
ec2-user
```

# CLI(정책)

사용자에게 정책을 다음과 같은 명령어로 할당이 가능하다. 아래는 managed policy형식으로 정책을 할당한다.

```
# aws sts get-caller-identity
# aws iam create-policy --policy-name sample-policy --policy-document
file://policy1.json
# aws iam attach-user-policy --user-name ec2-user --policy-arn
arn:aws:iam::0695158****:policy/sample-policy
# aws iam attach-user-policy --user-name ec2-user --policy-arn
arn:aws:iam::0695158****:policy/sample-policy --profile ec2-user
```

# CLI(정책)

특정 사용자에게 자원을 할당하기 위해서는 다음과 같은 방법으로 가능하다. 파일 이름은 policy2.json으로 한다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::my-bucket-name/*"
    }
  ]
}
```

사용 bob에게 "s3"의 "sample-bucket" 자원만 허용한다.

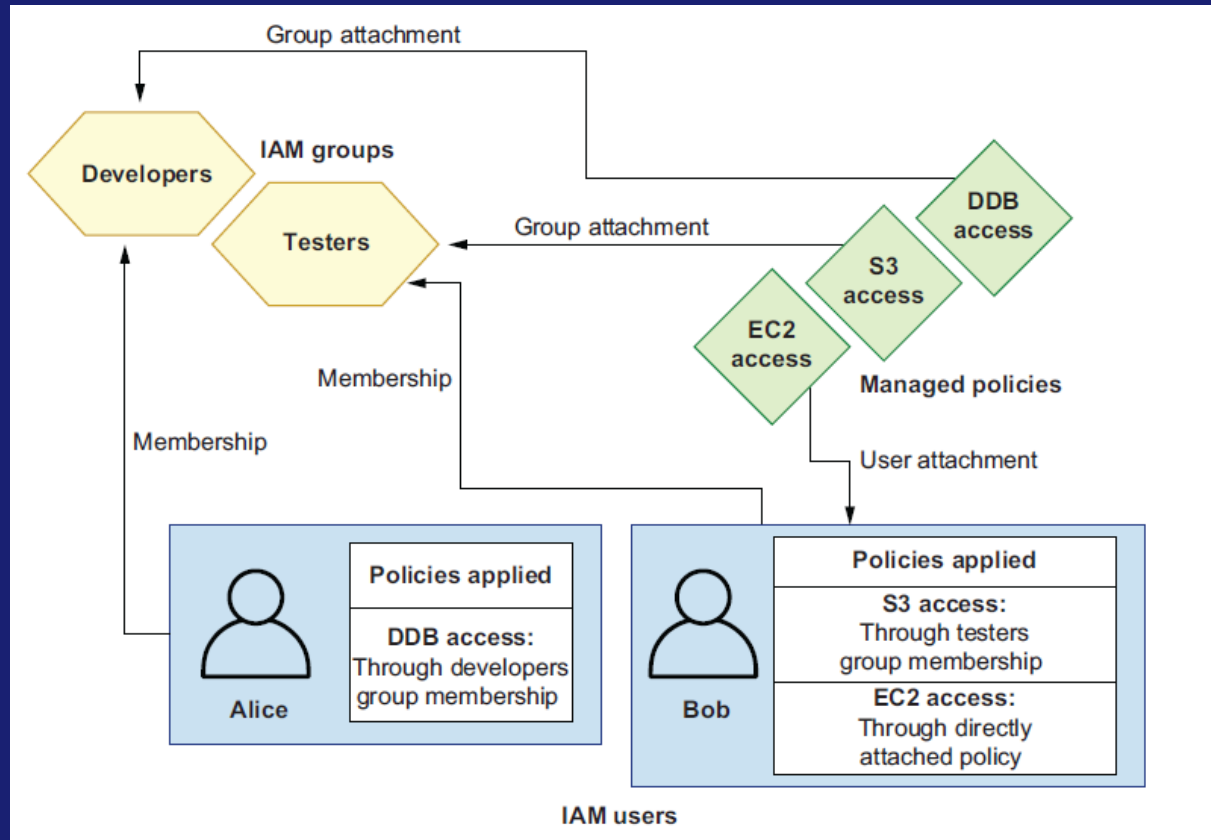
# CLI(정책)

test사용자를 생성하면서 정책을 같이 할당하고 싶은 경우, 아래와 같이 적용이 가능하다. 사용자를 생성하면서 "--policy-document" 옵션을 사용한다.

CLI에서 AWS API접근하기 위해서 access-key를 생성 합니다.

```
$ aws iam create-user --user-name testuser2
$ aws iam create-access-key --user-name testuser2
$ aws iam create-policy --policy-name SamplePolicy --policy-document
file://policy2.json
$ aws iam attach-user-policy --user-name testuser2 \
--policy-arn arn:aws:iam::0695158****:policy/SamplePolicy
```

# CLI(그룹)





# CLI(GROUP JSON)

그룹에 할당 예정인 정책은 다음과 같다. 아래 정책을 다음과 같은 명령어로 등록한다. 파일 이름은 policy3.json으로 생성한다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::my-sample-bucket/*"
    }
  ]
}
```

# CLI(GROUP)

위의 정책을 아래 명령어로 등록한다. 먼저, 그룹을 만들기 위해서 아래와 같이 실행한다.

```
$ aws iam create-group --group-name Developers
```

생성한 그룹에 아래와 같이 사용자 추가가 가능하다.

```
$ aws iam add-user-to-group --group-name Developers --user-name testuser
```

# CLI(GROUP)

생성된 그룹에 정책을 연결하기 위해서 다음과 같이 실행이 가능하다.

```
# aws iam create-policy --policy-name Developers --policy-document  
file://policy3.json
```

등록된 정책 "TestgroupPolicy"를 앞에서 만든 그룹에 연결 한다.

```
# aws iam attach-group-policy --group-name Developers --policy-arn  
arn:aws:iam::069515814364:policy/Developers
```

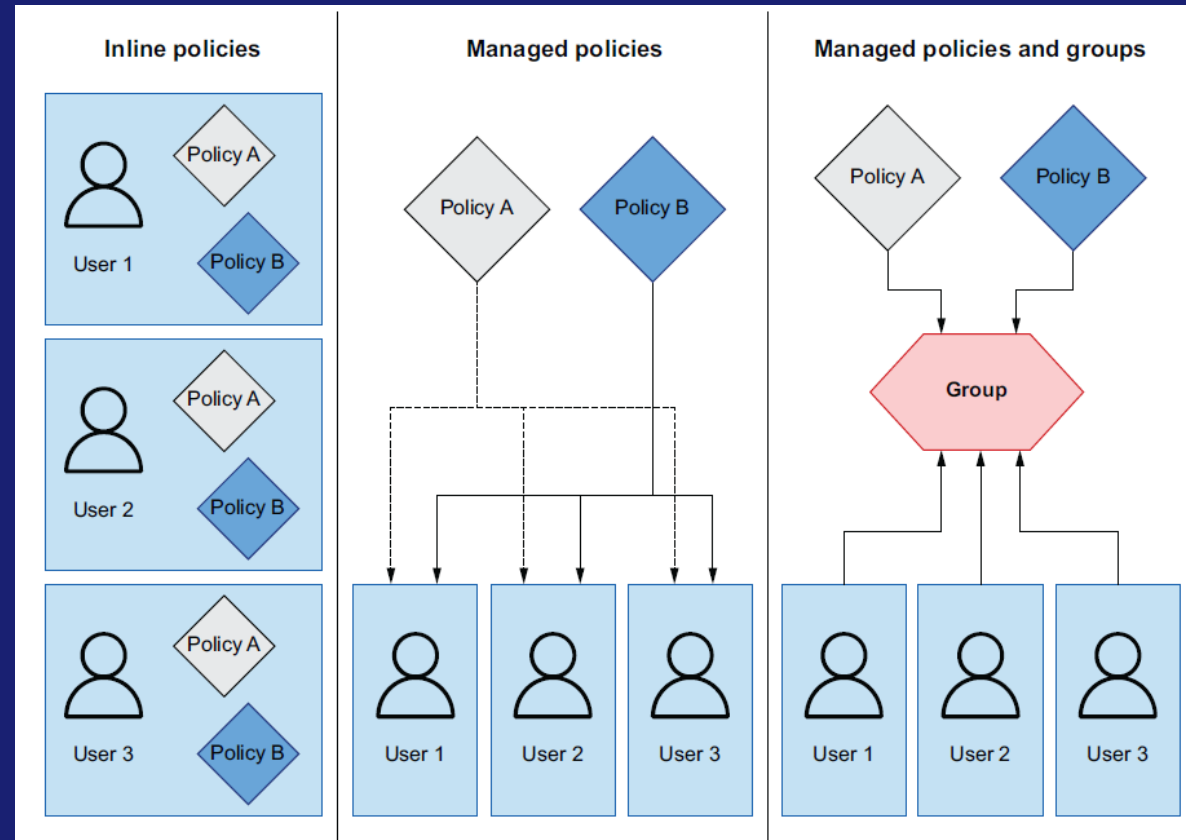
arn주소는 아래 명령어로 확인이 가능하다.

```
# aws sts get-caller-identity
```



STS: Secure Token Service

# CLI(GROUP)



# CLI(ROLE)

역할은 하나의 역할에 여러 정책을 할당 및 연결이 가능하다. 예를 들어서 아래와 같이 정책 파일을 생성한다. 파일 이름은 "resourcepolicy.json"으로 생성한다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/testuser"
      }
    }
  ]
}
```

# CLI(ROLE)

정책 파일을 하나 더 생성한다. 이름은 "policy4.json"으로 생성한다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:TerminateInstances",
      "Resource": "*"
    }
  ]
}
```

# CLI(ROLE)

정책 파일을 하나 더 생성한다. 이름은 "SampleRole-policy.json"으로 생성한다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "dynamodb:Query",
      "Resource": "arn:aws:dynamodb:ap-northeast-2:123456789012:table/MySampleTable"
    }
  ]
}
```

# CLI(ROLE)

역할을 생성한다. 테스트용으로 SampleRole라는 이름으로 생성한다.

```
aws iam create-role --role-name SampleRole --assume-role-policy-document  
file://resourcepolicy.json
```



# CLI(ROLE)

정책파일을 등록 후, 역할에 연결한다.

```
# aws iam create-policy --policy-name SampleRolePolicy --policy-document  
file://policy4.json  
# aws iam attach-role-policy --role-name SampleRole --policy-arn  
arn:aws:iam::069515814364:policy/SampleRolePolicy
```

해당 사용자로 전환 후, EC2의 인스턴스를 종료한다.

```
export AWS_PROFILE="ec2-user"  
$env:AWS_PROFILE='ec2-user'  
# aws ec2 describe-instances --query  
'Reservations[].Instances[].[Tags[?Key==`Name`].Value[] | [0],  
Placement.AvailabilityZone,InstanceType,State.Name]' --output text  
# aws ec2 terminate-instances --instance-ids {INSERT_INSTANCE_ID}
```

# 연습문제

# 아마존 보안

VPC

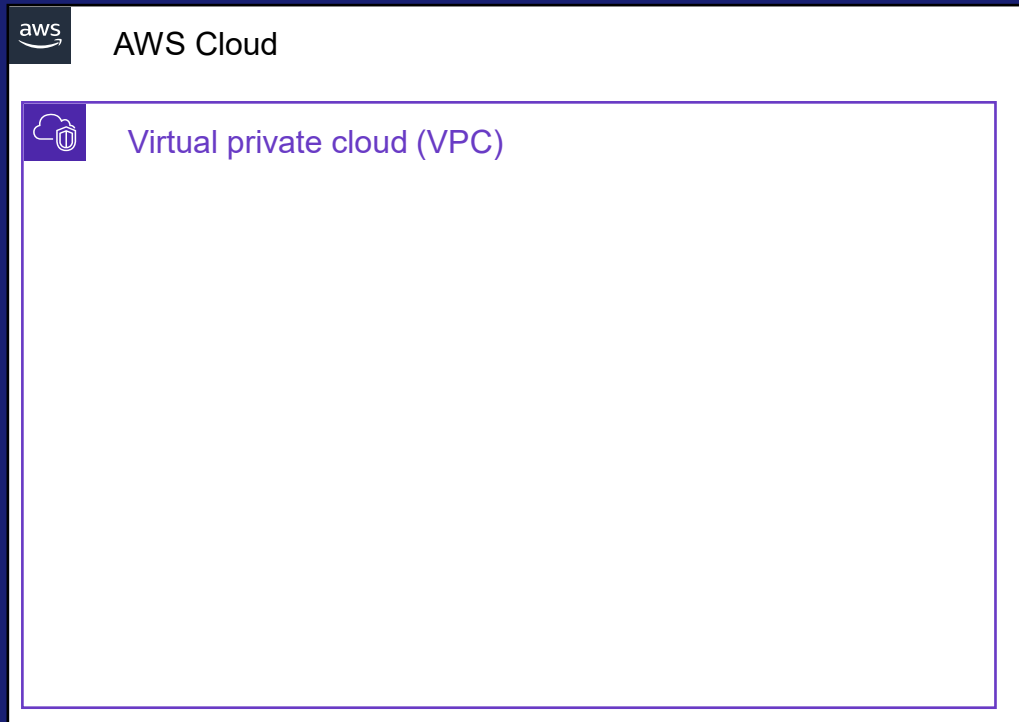
# VPC

VPC에서 사용할 이름은 아래와 같다.

자원 형식	이름
VPC	SecVPC

# VPC

이와 같은 형태로 자원을 생성할 예정.



# VPC

기본교육에서 최소 한 개의 VPC를 생성해야 올바르게 AWS자원 생성이 가능하다. VPC는 Virtual Private Computing의 약자다. VPC는 별도로 설정하지 않으면 기본 VPC값 "-"으로 하나가 생성이 되어 있다.

VPC (1) 정보								작업 ▼	VPC 생성
Q 검색								< 1 >	⚙
<input type="checkbox"/>	Name ▼	VPC ID ▼	상태 ▼	IPv4 CIDR ▼	IPv6 CIDR ▼	DHCP			
<input type="checkbox"/>	-	<a href="#">vpc-ad8567c6</a>	✔ Available	172.31.0.0/16	-			<a href="#">dopt-f</a>	

해당 VPC는 서비스 용도로 사용이 불가능하며, 최소 한 개의 비어 있는 VPC를 생성해야 한다. VPC에 들어가는 구성 요소는 다음과 같다.

1. 서브넷
2. 라우트
3. 네트워크

# VPC

VPC를 생성하기 위해서 IETF/IANA에서 권장하는 16비트 네트워크인 **192.168.0.0/16**를 사용한다. 이름은 "SecVPC"로 설정한다. 올바르게 내용 입력 후, VPC 생성 버튼을 누른 후 진행한다.

VPC > VPC > VPC 생성

### VPC 생성 정보

VPC는 AWS 클라우드의 격리된 부분으로서, Amazon EC2 인스턴스와 같은 AWS 객체로 채워집니다.

#### VPC 설정

**생성할 리소스 정보**  
VPC 리소스 또는 기타 네트워킹 리소스만 생성합니다.

☒ VPC만 ☐ VPC 등

**이름 태그 - 선택 사항**  
\*Name: 키와 사용자가 지정하는 값을 포함하는 태그를 생성합니다.

SecVPC

**IPv4 CIDR 블록 정보**  
☒ IPv4 CIDR 수동 입력  
☐ IPAM 할당 IPv4 CIDR 블록

**IPv4 CIDR**  
192.168.0.0/16  
CIDR 블록 크기는 /16에서 /28 사이여야 합니다.

**IPv6 CIDR 블록 정보**  
☒ IPv6 CIDR 블록 없음  
☐ IPAM 할당 IPv6 CIDR 블록  
☐ Amazon 제공 IPv6 CIDR 블록  
☐ 내가 소유한 IPv6 CIDR

**태연시 정보**  
기본값

#### 태그

태그는 AWS 리소스에 할당하는 레이블입니다. 각 태그는 키와 선택적 값으로 구성됩니다. 태그를 사용하여 리소스를 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.

**키** **값 - 선택 사항**

Q Name X Q SecVPC X 태그 제거

태그 추가

태그를 49개 더 추가할 수 있음

취소 **VPC 생성**

# VPC

이름	값
이름 태그	SecVPC
IPv4 CIDR	수동 입력
IPv4 CIDR	192.168.0.0/16
IPv6 CIDR	IPv6 CIDR 블록 없음
테넌시	기본값

VPC 설정

생성할 리소스 정보

VPC 리소스 또는 VPC 및 기타 네트워킹 리소스만 생성합니다.

☒ VPC만
 ☐ VPC 등

이름 태그 - 선택 사항

'Name' 키와 사용자가 지정한 값을 포함하는 태그를 생성합니다.

SecVPC

IPv4 CIDR 블록 정보

☒ IPv4 CIDR 수동 입력
 ☐ IPAM 할당 IPv4 CIDR 블록

IPv4 CIDR

192.168.0.0/16

IPv6 CIDR 블록 정보

☒ IPv6 CIDR 블록 없음
 ☐ IPAM 할당 IPv6 CIDR 블록
 ☐ Amazon 제공 IPv6 CIDR 블록
 ☐ 내가 소유한 IPv6 CIDR

테넌시 정보

기본값

태그

태그는 AWS 리소스에 할당하는 레이블입니다. 각 태그는 키와 선택적 값으로 구성됩니다. 태그를 사용하여 리소스를 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.

키

값 - 선택 사항

2024-10-14

:^)<-<

74



# VPC CLI

VPC를 CLI로 생성하는 경우 아래와 같이 명령어를 수행한다.

```
aws ec2 create-vpc --cidr-block 192.168.0.0/16 \  
--tag-specifications "ResourceType=vpc,Tags=[{Key=Name,Value=SecVPC}]"
```

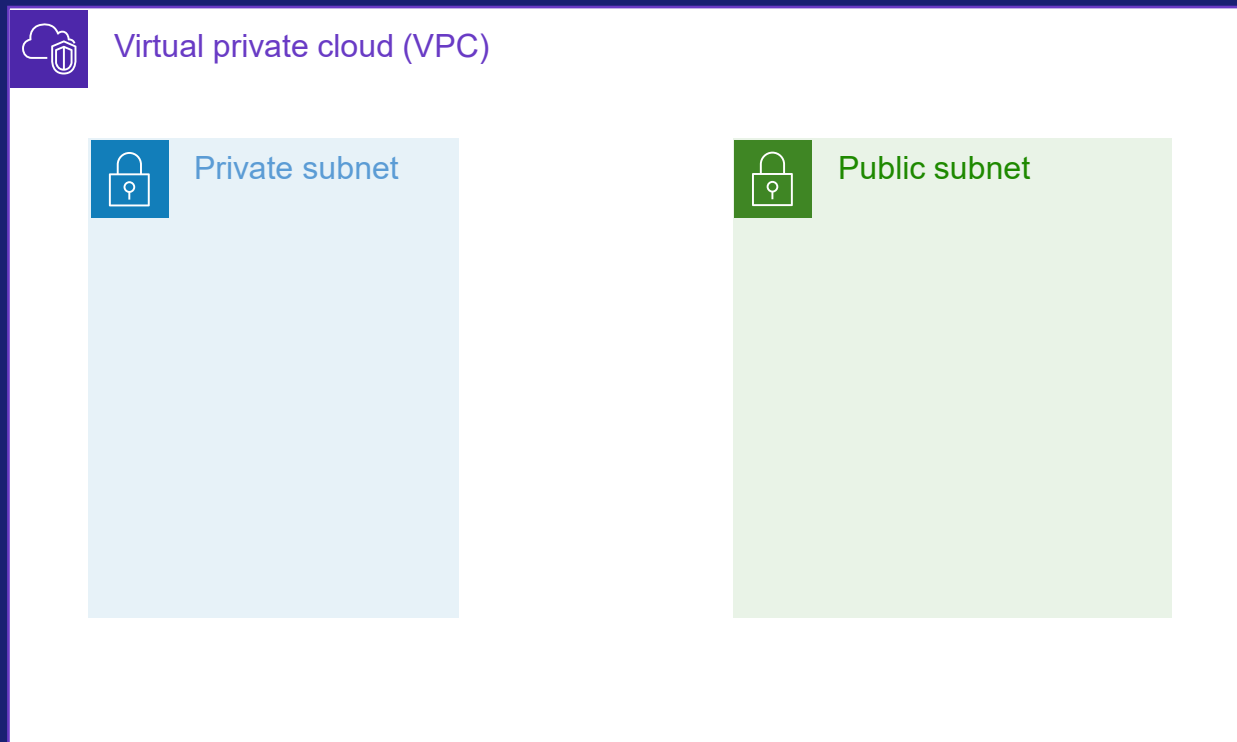
VPC를 CLI로 생성하는 경우, SecVPC라는 이름 구성이 불가능하다. 좀 복잡하지만 위의 방식으로 선언 혹은 태그 선언을 별도로 한다.

```
aws ec2 describe-vpcs --filter Name=tag:Name,Values=SecVPC \  
--query Vpcs[].VpcId --output text
```

```
aws ec2 create-tags --resources vpc-1a2b3c4d \  
--tags Key=Name,Value=Production
```

# 서브넷

서브넷은 다음과 같은 형태로 구성한다.



# 서브넷

인스턴스가 네트워크를 사용하기 위해서는 최소 한 개의 서브넷이 필요하다. 서브넷은 VPC의 상속 자원이다. 우리가 만들 서브넷은 다음과 같다. 각각 서브넷은 **가용영역(Availability Zone)**를 하나씩 가지고 있다.

1. 프라이빗 네트워크
2. 퍼블릭 네트워크

# 서브넷

서브넷 생성 및 구성한다. 현재 사용중인 192.168.0.0/16 대역에서, 192.168.1.0/24 대역으로 생성한다.

[VPC](#) > [서브넷](#) > 서브넷 생성

## 서브넷 생성 [정보](#)

**VPC**

VPC ID

이 VPC에 서브넷을 생성합니다.

vpc-06a65ad0c64ad0440 (SecVPC) ▼

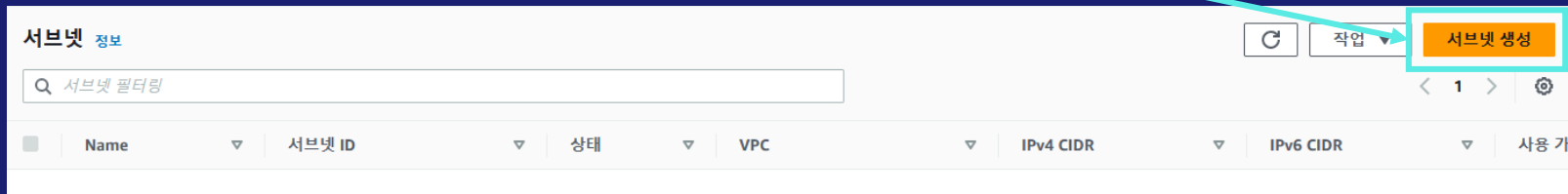
연결된 VPC CIDR

IPv4 CIDR

192.168.0.0/16

# 서브넷

서브넷을 생성한다. 서브넷 생성은 EC2 메뉴에서 확인이 가능하다.



# 서브넷

서브넷이 생성될 VPC위치를 선택한다. "aws-lab-vpc"를 선택한다.

서브넷 생성 정보

VPC

VPC ID  
이 VPC에 서브넷을 생성합니다.

VPC 선택

Q |

a DC≥;

새 서브넷을 생성하려면 먼저 VPC를 선택합니다.

새 서브넷 추가

취소

서브넷 생성

# 서브넷

선택이 되면 다음과 같이 상단 화면에 VPC정보가 출력이 되며, 아래로 서브넷 입력 화면이 출력이 된다.

**VPC**

**VPC ID**  
이 VPC에 서브넷을 생성합니다.

**연결된 VPC CIDR**

**IPv4 CIDR**  
192.168.0.0/20

서브넷 설정

서브넷의 CIDR 블록 및 가용 영역을 지정합니다.

1/1개 서브넷

서브넷 이름

'Name' 키와 사용자가 지정하는 값을 포함하는 태그를 생성합니다.

private-subnet

이름은 최대 256자까지 입력할 수 있습니다.

가용 영역 정보

서브넷이 상주할 영역을 선택합니다. 선택하지 않으면 Amazon이 자동으로 선택합니다.

ap-northeast-2a

IPv4 CIDR 블록 정보

192.168.10.0/24

▼ 태그 - 선택 사항

키

aws-public-az1-subnet

값 - 선택 사항

제거

새 태그 추가

49글(을) 태그.개 더 추가할 수 있습니다.

제거

새 서브넷 추가

# SUBNET

서브넷 정보를 입력한다. 서브넷 관련된 정보는 PPT에 103페이지를 참고한다.

화면에 나온 입력 정보는 "aws-public-az1-subnet"이다. 생성이 완료가 되면 public-az2, private-az1, private-az2 반복적으로 생성한다.

이름	값
서브넷 이름	sec-internal-subnet-1
IPv4 CIDR	192.168.10.0/24
서브넷 이름	sec-public-subnet-1
아이피 주소	192.168.20.0/24
가용영역	ap-northeast-2a

2024-10-15

:^)<-<

82



# 서브넷

완성이 되면 다음과 같이 서브넷 생성이 완료가 되었다.

<input type="checkbox"/>	Name ▾	서브넷 ID ▾	상태 ▾	VPC ▾	IPv4 CIDR
<input type="checkbox"/>	aws-lab-public-az2-subnet	subnet-04fc6551e07e78d57	✔ Available	vpc-02f633be0f83e3235   aws...	192.168.20.0/24
<input type="checkbox"/>	aws-lab-private-az2-subnet	subnet-0a99af84a6528e526	✔ Available	vpc-02f633be0f83e3235   aws...	192.168.60.0/24
<input type="checkbox"/>	aws-lab-private-az1-subnet	subnet-0e7b02410d9883a95	✔ Available	vpc-02f633be0f83e3235   aws...	192.168.50.0/24
<input type="checkbox"/>	aws-lab-public-az1-subnet	subnet-0a55a7645b632b47c	✔ Available	vpc-02f633be0f83e3235   aws...	192.168.10.0/24

# 서브넷

EC2의 VPC에서 사용할 이름은 아래와 같다.

자원 형식	이름	값	가용영역
서브넷	sec-public-subnet-1	192.168.10.0/24	ap-northeast-2c
서브넷	sec-public-subnet-2	192.168.20.0/24	ap-northeast-2d
서브넷	sec-internal-subnet-1	192.168.1.0/24	ap-northeast-2a
서브넷	sec-internal-subnet-2	192.168.2.0/24	ap-northeast-2b

# 서브넷

서브넷 정보는 아래와 같이 입력한다. 입력이 완료가 되면 서브넷 생성을 실행한다. 서브넷 정보는 아래와 같이 입력 및 구성한다.

서브넷 설정

서브넷의 CIDR 블록 및 가용 영역을 지정합니다.

1/1개 서브넷

서브넷 이름

'Name' 키와 사용자가 지정하는 값을 포함하는 태그를 생성합니다.

sec-internal-subnet

이름은 최대 256자까지 입력할 수 있습니다.

가용 영역 정보

서브넷이 상주할 영역을 선택합니다. 선택하지 않으면 Amazon이 자동으로 선택합니다.

아시아 태평양 (서울) / ap-northeast-2a

IPv4 VPC CIDR 블록 정보

서브넷을 생성할 IPv4 VPC CIDR 블록을 선택합니다.

192.168.0.0/16

IPv4 서브넷 CIDR 블록

192.168.1.0/24 256 IPs

▼ 태그 - 선택 사항

키

값 - 선택 사항

Q Name X Q sec-internal-subnet X 제거

새 태그 추가

49글(字) 태그 개 더 추가할 수 있습니다.

제거

새 서브넷 추가

<input checked="" type="checkbox"/>	Name ▲	서브넷 ID ▼	상태 ▼	VPC ▼	IPv4 CIDR ▼
<input checked="" type="checkbox"/>	sec-internal-subnet-1	<a href="#">subnet-06ea8366ac58cd9d7</a>	✔ Available	<a href="#">vpc-06a65ad0c64ad0440</a>   <a href="#">Sec...</a>	192.168.1.0/24
<input checked="" type="checkbox"/>	sec-internal-subnet-2	<a href="#">subnet-09fd0b4c425915ddd</a>	✔ Available	<a href="#">vpc-06a65ad0c64ad0440</a>   <a href="#">Sec...</a>	192.168.2.0/24
<input checked="" type="checkbox"/>	sec-public-subnet-1	<a href="#">subnet-0d70099f3bda5d0d6</a>	✔ Available	<a href="#">vpc-06a65ad0c64ad0440</a>   <a href="#">Sec...</a>	192.168.10.0/24
<input checked="" type="checkbox"/>	sec-public-subnet-2	<a href="#">subnet-0adf1cf5cfd881e18</a>	✔ Available	<a href="#">vpc-06a65ad0c64ad0440</a>   <a href="#">Sec...</a>	192.168.20.0/24

2023-05-30

:^)<-<

85

# 서브넷(CLI)

서브넷을 생성하기 위해서 아래와 같이 명령어를 실행한다.

```
aws ec2 create-subnet --vpc-id vpc-08f6c9d471f2a96e5 --cidr-block  
192.168.10.0/24 --availability-zone ap-northeast-2c  
aws ec2 create-tags --resources subnet-0624aa32c4e66c232 `  
--tags Key=Name,Value=sec-public-subnet-1  
  
aws ec2 create-subnet --vpc-id vpc-08f6c9d471f2a96e5 --cidr-block  
192.168.20.0/24 --availability-zone ap-northeast-2d  
aws ec2 create-tags --resources subnet-07c996bf1d7277d61 `br/>--tags Key=Name,Value=sec-public-subnet-2
```

# 서브넷(CLI)

서브넷을 생성하기 위해서 아래와 같이 명령어를 실행한다.

```
aws ec2 create-subnet --vpc-id vpc-08f6c9d471f2a96e5 --cidr-block  
192.168.1.0/24 --availability-zone ap-northeast-2a  
aws ec2 create-tags --resources subnet-0ff1ec59cb43b6ef4 `  
--tags Key=Name,Value=sec-internal-subnet-1  
  
aws ec2 create-subnet --vpc-id vpc-08f6c9d471f2a96e5 --cidr-block  
192.168.2.0/24 --availability-zone ap-northeast-2b  
aws ec2 create-tags --resources subnet-0818da96e53221b13 `  
--tags Key=Name,Value=sec-internal-subnet-2
```

# 인스턴스

구성한 서브넷에 인스턴스를 생성하여 연결한다. 이때 사용하는 자원은 ENI/EIP를 사용한다. 가상머신 생성 시, 서브넷의 아이디 및 사용할 아마존 가상머신 이미지(AMI)를 확인해야 한다.

이름	설명
ENI(Elastic Network Interface)	가상머신에서 사용하는 가상 인터페이스 입니다.
EIP(Elastic IPaddress)	가상머신에서 사용하는 아이피 주소 입니다.

```
aws ec2 describe-vpcs --filter Name=tag:Name,Values=SecVPC `
--query Vpcs[].VpcId --output text
aws ec2 describe-subnets --filter "Name=vpc-id,Values=vpc-08f6c9d471f2a96e5" --query
"Subnets[*].[SubnetId,Tags[*].Value]" --output text

aws ec2 run-instances `
--instance-type t2.micro `
--subnet-id subnet-0ff1ec59cb43b6ef4 --image-id ami-0e18fe6ecdad223e5

aws ec2 describe-instances --filters "Name=instance-type,Values=t2.micro,t2.small" --query
"Reservations[].Instances[].{InstanceType:InstanceType}"

aws ec2 terminate-instances --instance-ids i-0e135389904c17d60
```

# 인스턴스

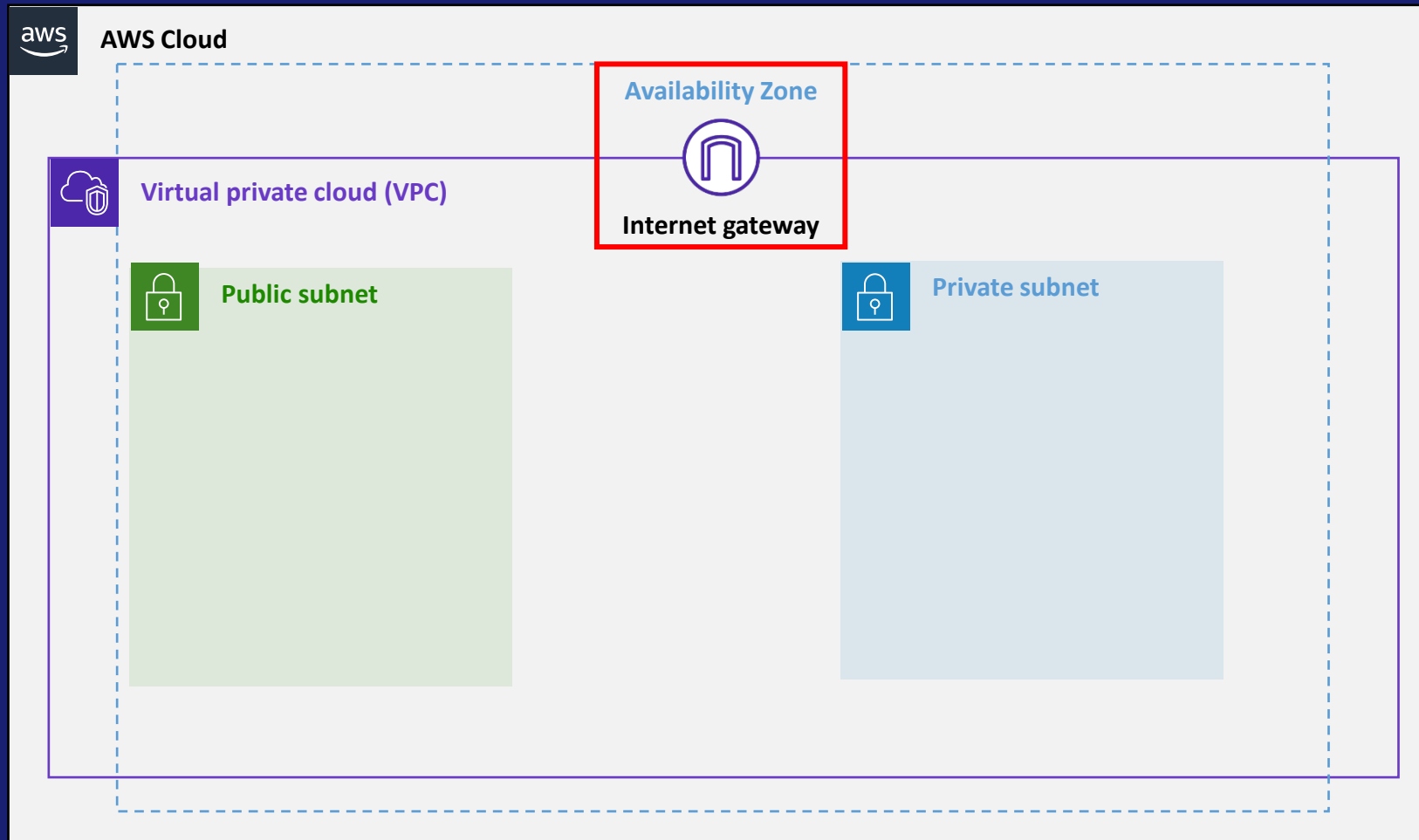
가상머신 한 대를 더 구성한다. 이 가상머신은 외부 네트워크로 연결이 된다.

```
aws ec2 describe-vpcs --filter Name=tag:Name,Values=SecVPC `
--query Vpcs[].VpcId --output text
aws ec2 describe-subnets --filter "Name=vpc-id,Values=\"vpc-
08f6c9d471f2a96e5\" --query \"Subnets[*].[SubnetId,Tags[*].Value]\" --output
text
```

```
aws ec2 run-instances `
--instance-type t2.micro `
--subnet-id subnet-0624aa32c4e66c232 --image-id ami-0e18fe6ecdad223e5
```

```
aws ec2 describe-instances --filters "Name=instance-
type,Values=t2.micro,t2.small" --query
"Reservations[].Instances[].{InstanceType:InstanceType}"
```

# IGW





# IGW

VPC생성이 완료가 되면, IGW(Internet Gate Way)를 생성해야 한다. 이를 통해서 외부 네트워크와 통신이 가능하다.

VPC > 인터넷 게이트웨이 > 인터넷 게이트웨이 생성

## 인터넷 게이트웨이 생성 [정보](#)

인터넷 게이트웨이는 VPC를 인터넷과 연결하는 가상 라우터입니다. 새 인터넷 게이트웨이를 생성하려면 아래에서 게이트웨이 이름을 지정해야 합니다.

### 인터넷 게이트웨이 설정

이름 태그  
'Name' 키와 사용자가 지정하는 값을 포함하는 태그를 생성합니다.

### 태그 - 선택 사항

태그는 AWS 리소스에 할당하는 레이블입니다. 각 태그는 키와 선택적 값으로 구성됩니다. 태그를 사용하여 리소스를 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.

키

값 - 선택 사항

49글(글) 태그 개 더 추가할 수 있습니다.

2023-05-30

:^)<-<

91

# IGW

IGW를 VPC에 연결한다.

인터넷 게이트웨이 (1/2) 정보

검색

작업 ▲

인터넷 게이트웨이 생성

	Name	인터넷 게이트웨이 ID	상태	VPC ID
<input type="checkbox"/>	-	<a href="#">igw-69c01101</a>	✔ Attached	<a href="#">vpc-ad8567c6</a>
<input checked="" type="checkbox"/>	SecIGW	<a href="#">igw-09e12a95bef918470</a>	⊖ Detached	-

세부 정보 보기

VPC에 연결

VPC에서 분리

태그 관리

인터넷 게이트웨이 삭제

1 > ⚙

사용자

69515814364

69515814364

VPC에 연결(igw-09e12a95bef918470) 정보

VPC

인터넷 게이트웨이를 VPC에 연결하여 인터넷과의 통신을 활성화합니다. 아래에서 연결하려는 VPC를 지정하십시오.

사용 가능한 VPC

인터넷 게이트웨이를 이 VPC에 연결합니다.

Q VPC 선택

vpc-06a65ad0c64ad0440 - SecVPC

▶ AWS Command Line Interface 명령

취소

인터넷 게이트웨이 연결

# IGW(CLI)

인터넷 게이트웨이는 아래 명령어로 생성이 가능하다.

```
aws ec2 create-internet-gateway  
> InternetGatewayId: igw-05e3998ab0201bc9c  
aws ec2 attach-internet-gateway `--internet-gateway-id igw-05e3998ab0201bc9c `--vpc-id vpc-08f6c9d471f2a96e5
```

# NAT

## NAT 게이트웨이 생성 [정보](#)

프라이빗 서브넷의 인스턴스가 다른 VPC, 온프레미스 네트워크 또는 인터넷의 서비스에 연결하는 데 사용할 수 있는 가능성이 뛰어난 관리형 NAT(Network Address Translation) 서비스입니다.

### NAT 게이트웨이 설정

#### 이름 - 선택 사항

'Name' 키와 사용자가 지정하는 값을 포함하는 태그를 생성합니다.

SecNAT

이름은 최대 256자까지 입력할 수 있습니다.

#### 서브넷

NAT 게이트웨이를 생성할 서브넷을 선택합니다.

subnet-0624aa32c4e66c232 (sec-public-subnet-1)

#### 연결 유형

NAT 게이트웨이에 대한 연결 유형을 선택합니다.

☒ 퍼블릭

☐ 프라이빗

#### 탄력적 IP 할당 ID [정보](#)

NAT 게이트웨이에 탄력적 IP 주소를 할당합니다.

탄력적 IP 선택

탄력적 IP 할당

### ▶ 추가 설정 정보

#### 태그

태그는 AWS 리소스에 할당하는 레이블입니다. 각 태그는 키와 선택적 값으로 구성됩니다. 태그를 사용하여 리소스를 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.

#### 키

#### 값 - 선택 사항

Q Name

Q SecNAT

제거

새 태그 추가

49글(줄) 태그 개 더 추가할 수 있습니다.

취소

NAT 게이트웨이 생성

2023-05-30

# NAT

NAT는 아래 명령어로 생성한다.

```
aws ec2 allocate-address `
    --domain vpc `
    --network-border-group ap-northeast-2
```

```
aws ec2 describe-subnets --filter "Name=vpc-id,Values=vpc-
08f6c9d471f2a96e5" --query "Subnets[*].[SubnetId,Tags[*].Value]" --output
text
```

```
aws ec2 create-nat-gateway --subnet-id subnet-0624aa32c4e66c232 --
allocation-id eipalloc-00d0192becb875dbd
```

# 라우트 테이블

라우트 테이블 구성한다. 이를 통해서 VPC/Subnet/Internet간에 패킷 전달이 된다.

[VPC](#) > [라우팅 테이블](#) > 라우팅 테이블 생성

## 라우팅 테이블 생성 [정보](#)

라우팅 테이블은 VPC, 인터넷 및 VPN 연결 내 서브넷 간에 패킷이 전달되는 방법을 지정합니다.

### 라우팅 테이블 설정

**이름 - 선택 사항**  
'Name' 키와 사용자가 지정하는 값을 포함하는 태그를 생성합니다.

**VPC**  
이 라우팅 테이블에 대해 사용할 VPC입니다.

vpc-06a65ad0c64ad0440 (SecVPC) ▼

### 태그

태그는 AWS 리소스에 할당하는 레이블입니다. 각 태그는 키와 선택적 값으로 구성됩니다. 태그를 사용하여 리소스를 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.

키	값 - 선택 사항
<input type="text" value="Name"/>	<input type="text" value="SecVPC-RT"/>

49을(들) 태그.개 더 추가할 수 있습니다.

[취소](#) [라우팅 테이블 생성](#)

# 라우트 테이블

문제 없이 생성이 되면, 아래처럼 앞서 생성한 CIDR, 192.168.0.0/16대역이 보인다.

라우팅

서브넷 연결

엣지 연결

라우팅 전파

태그

라우팅 (1)

모두 ▾

라우팅 편집

Q 라우팅 필터링

< 1 > ⚙

대상 ▾	대상 ▾	상태 ▾	전파됨 ▾
192.168.0.0/16	local	✔ 활성화	아니요

# 보안그룹

대시보드에서 다음과 같이 보안그룹 생성이 가능하다.

## 보안 그룹 생성 정보

보안 그룹은 인바운드 및 아웃바운드 트래픽을 관리하는 인스턴스의 가상 방화벽 역할을 합니다. 새 보안 그룹을 생성하려면 아래의 필드를 작성하십시오.

### 기본 세부 정보

#### 보안 그룹 이름 정보

생성 후에는 이름을 편집할 수 없습니다.

#### 설명 정보

#### VPC 정보

▼



# 보안그룹

대시보드에서 다음과 같이 보안그룹 생성이 가능하다.

인바운드 규칙 정보

유형 정보

프로토콜 정보

포트 범위 정보

소스 정보

설명 - 선택 사항 정보

SSH

TCP

22

내 IP

삭제

121.134.6.107/32

규칙 추가

# 보안그룹

대시보드에서 다음과 같이 보안그룹 생성이 가능하다.

인바운드 규칙 정보

유형 정보

프로토콜 정보

포트 범위 정보

소스 정보

설명 - 선택 사항 정보

SSH

TCP

22

내 IP

삭제

121.134.6.107/32

규칙 추가

# 보안그룹

대시보드에서 다음과 같이 보안그룹 생성이 가능하다.

아웃바운드 규칙 정보

유형 정보

프로토콜 정보

포트 범위 정보

대상 정보

설명 - 선택 사항 정보

모든 트래픽 ▼

전체

전체

내 IP ▼

121.134.6.107/32 ✕

삭제

규칙 추가

# 보안그룹

보안그룹은 다음과 같이 생성한다.

```
aws ec2 create-security-group --group-name service --description "My security group" --vpc-id vpc-08f6c9d471f2a96e5
```

보안그룹이 생성이 되면, 다음과 같이 정책을 추가한다.

```
aws ec2 authorize-security-group-ingress `
  --group-id sg-0e752815a29678b0e `
  --cidr "121.134.6.107/32" `
  --port 22 `
  --protocol 6
```

# 보안그룹

구성된 보안 그룹은 가상머신에 연결이 필요한 경우, 아래와 같이 작업을 수행한다.

```
aws ec2 modify-instance-attribute `
  --instance-id i-05920875a5ae7ddbc `
  --groups sg-0e752815a29678b0e
```

보안 그룹은 복수로 적용이 가능하다.

```
aws ec2 modify-instance-attribute `
  --instance-id i-05920875a5ae7ddbc `
  --groups sg-0e752815a29678b0e sg-1234 sg-42483
```

# 보안그룹

추가로 웹 서버를 정책에 포함하고 싶으면 다음과 같이 입력한다.

```
aws ec2 authorize-security-group-ingress `
  --group-id sg-0e752815a29678b0e `
  --cidr "121.134.6.107/32" `
  --port 80 `
  --protocol 6
```

# VPC PEERING

VPC TO VPC기반의 보안 기능을 구성하기 위해서 추가적으로 VPC를 두개 더 생성한다.

```
aws ec2 create-vpc --cidr-block 10.0.0.0/24 --tag-specifications  
'ResourceType=vpc,Tags=[{Key=Name,Value=TestVPC-1}]'
```

```
aws ec2 create-vpc --cidr-block 10.0.1.0/24 --tag-specifications  
'ResourceType=vpc,Tags=[{Key=Name,Value=TestVPC-2}]'
```

```
aws ec2 create-subnet \  
--vpc-id vpc-0756895276e892db2 \  
--cidr-block 10.0.0.0/24
```

```
aws ec2 create-subnet \  
--vpc-id vpc-0384b34a620bfff789 \  
--cidr-block 10.0.1.0/24
```

# VPC PEERING

올바르게 구성이 되었는지 EC2에 인스턴스를 생성한다.

```
aws ec2 run-instances --instance-type t2.micro --subnet-id subnet-0c69aa7f5d1bf384e --image-id ami-0e18fe6ecdad223e5
```

```
aws ec2 run-instances --instance-type t2.micro --subnet-id subnet-088e1b400e8015007 --image-id ami-0e18fe6ecdad223e5
```



# 종합문제

아마존 보안