

Android _ Aws[ec2]_ node.js 통신

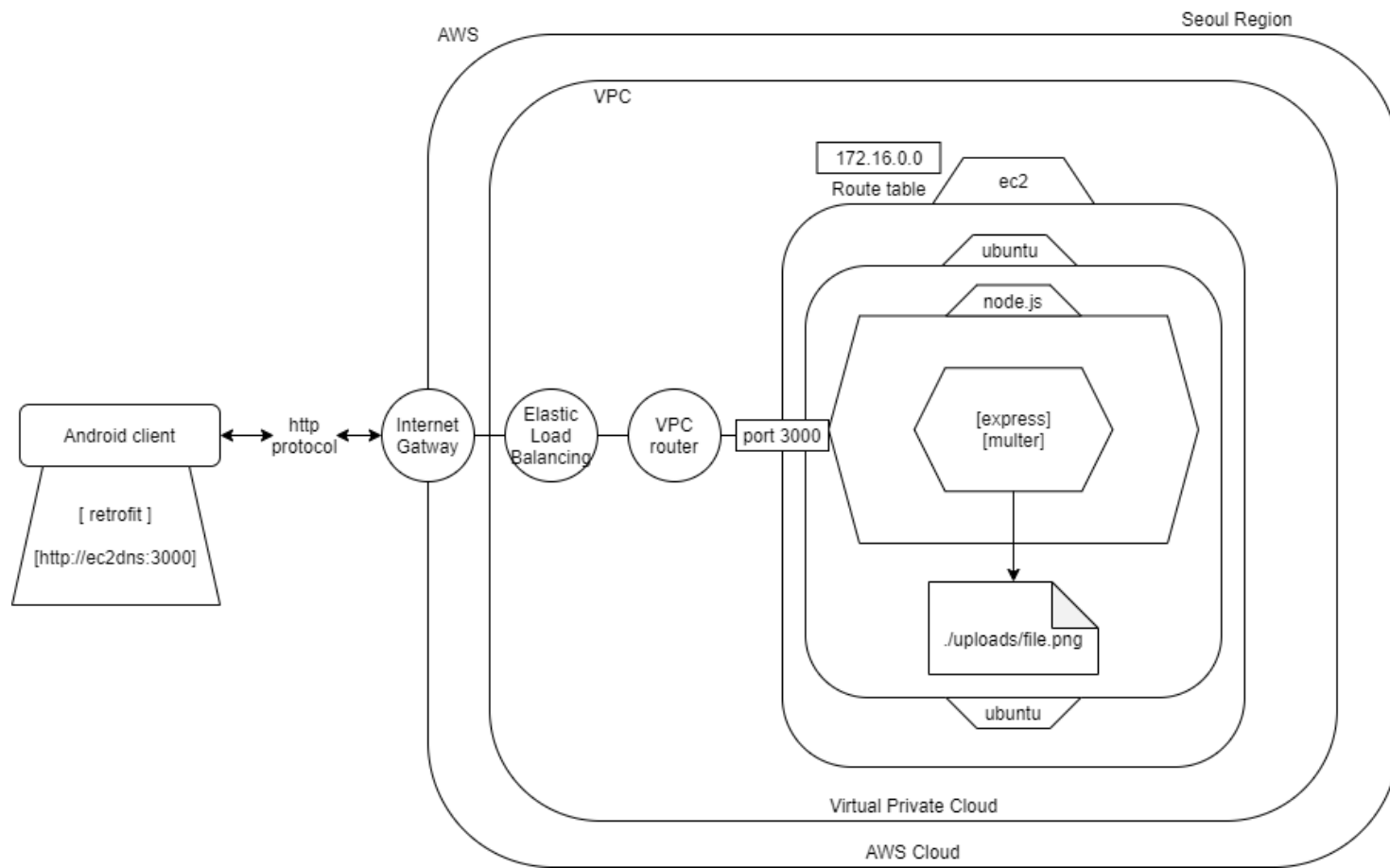
모바일 프로그래밍

2017152047 심정수

목차

- 1. 클라우드 컴퓨팅이란
- 2. AWS_EC2란
- 3. AWS_EC2 시작하기
- 4. AWS_EC2 접속하기
- 5. AWS_EC2 - 중지 및 종료
- 6. ubuntu_node.js install
- 7. AWS_EC2 - node.js 시작하기
- 8. AWS_EC2 - node.js 코드작성
- 9. Android 측 코드작성
- 10. AWS_EC2 - 보안그룹 설정

Android _ EC2 _ node.js 통신 과정



1. 클라우드 컴퓨팅이란

클라우드 컴퓨팅이란 무엇입니까?

클라우드 컴퓨팅은 IT 리소스를 인터넷을 통해 온디맨드로 제공하고 사용한 만큼만 비용을 지불하는 것을 말합니다. 물리적 데이터 센터와 서버를 구입, 소유 및 유지 관리하는 대신, Amazon Web Services(AWS)와 같은 클라우드 공급자로부터 필요에 따라 컴퓨팅 파워, 스토리지, 데이터베이스와 같은 기술 서비스에 액세스할 수 있습니다.

<https://aws.amazon.com/ko/what-is-cloud-computing/>

2. AWS_EC2란

Amazon EC2이란 무엇입니까?

PDF

RSS

Amazon Elastic Compute Cloud(Amazon EC2)는 Amazon Web Services(AWS) 클라우드에서 확장 가능 컴퓨팅 용량을 제공합니다. Amazon EC2를 사용하면 하드웨어에 선투자할 필요가 없어 더 빠르게 애플리케이션을 개발하고 배포할 수 있습니다. Amazon EC2를 통해 원하는 만큼 가상 서버를 구축하고 보안 및 네트워크 구성과 스토리지 관리가 가능합니다. 또한 Amazon EC2는 요구 사항이나 갑작스러운 인기 증대 등 변동 사항에 따라 신속하게 규모를 확장하거나 축소할 수 있어 서버 트래픽 예측 필요성이 줄어듭니다.

https://docs.aws.amazon.com/ko_kr/AWSEC2/latest/UserGuide/concepts.html

3. AWS_EC2 시작하기 - 회원가입



새로운 AWS 계정으로 프리 티어 제품을 살펴보세요.

자세히 알아보려면 aws.amazon.com/free를 방문하세요.



AWS에 가입

이메일 주소
이 이메일 주소를 사용하여 새 AWS 계정에 로그인합니다.

암호

암호 확인

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

계속(1/5단계)

[기존 AWS 계정에 로그인](#)

계정 생성시 필요한 것

1. 해외 결제 가능한 신용카드 혹은 체크카드
2. 영문 주소
3. 요금 결제 시 연락 받을 정확한 정보

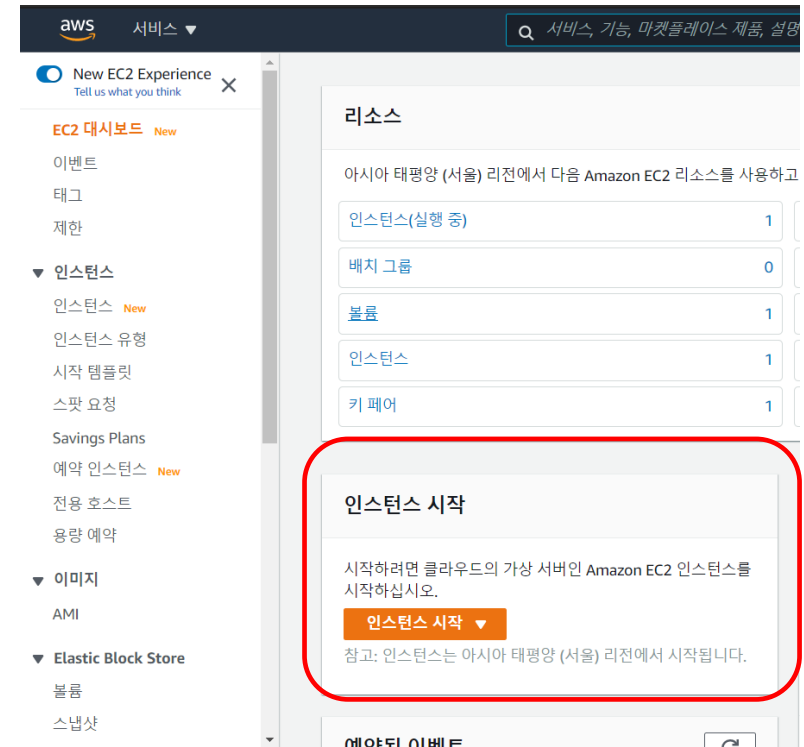
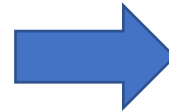
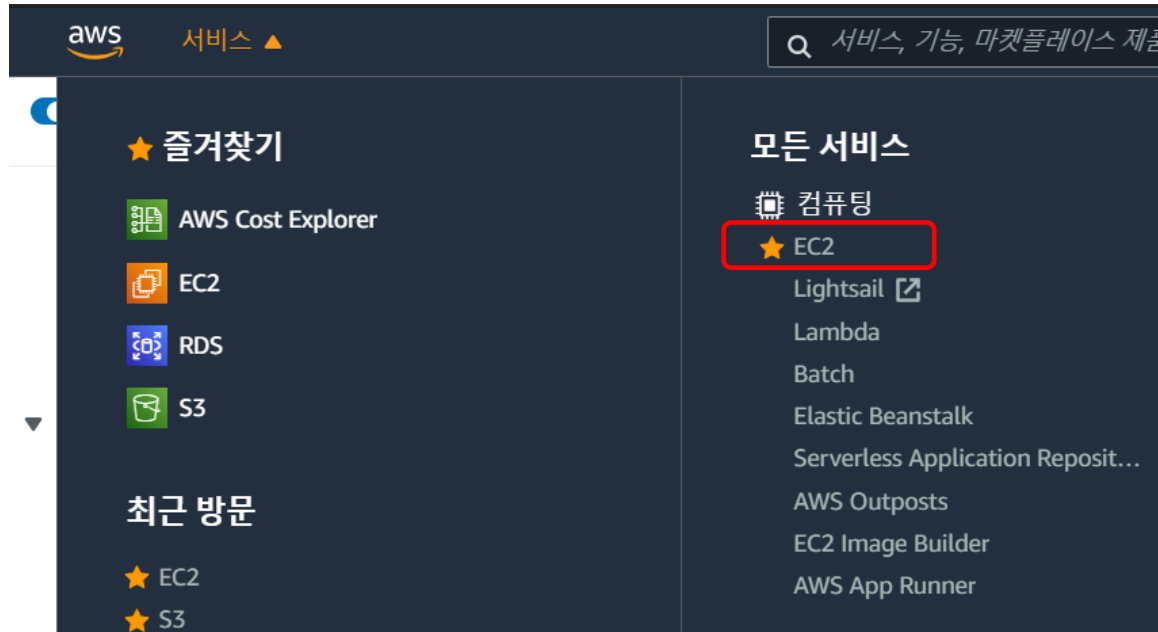
3. AWS_EC2 시작하기 – 리전 선택

The screenshot shows the AWS Management Console interface. On the left is a navigation sidebar with options like 'New EC2 Experience', 'EC2 대시보드', '이벤트', '태그', '제한', '인스턴스', 'AMI', and 'Elastic Block Store'. The main content area is titled '리소스' (Resources) and shows a summary of EC2 resources in the 'Asia Pacific (Seoul)' region. Below this, there's a section for '인스턴스 시작' (Start Instance) with a button to '인스턴스 시작'. To the right, a '서비스 상태' (Service Status) section indicates that the 'Asia Pacific (Seoul)' region is operational. A dropdown menu is open on the right side of the console, displaying a list of available AWS regions. The region '아시아 태평양 (서울) ap-northeast-2' is highlighted with a red box.

Region Name	Region ID
미국 동부 (버지니아 북부)	us-east-1
미국 동부 (오하이오)	us-east-2
미국 서부 (캘리포니아)	us-west-1
미국 서부 (오레곤)	us-west-2
아프리카 (케이프타운)	af-south-1
아시아 태평양 (홍콩)	ap-east-1
아시아 태평양 (뭄바이)	ap-south-1
아시아 태평양 (오사카)	ap-northeast-3
아시아 태평양 (서울)	ap-northeast-2
아시아 태평양 (싱가포르)	ap-southeast-1
아시아 태평양 (시드니)	ap-southeast-2
아시아 태평양 (도쿄)	ap-northeast-1
캐나다 (중부)	ca-central-1
유럽 (프랑크푸르트)	eu-central-1
유럽 (아일랜드)	eu-west-1
유럽 (런던)	eu-west-2

각 리전마다 독립적인 인스턴스를 생성할 수 있어 과금에 주의

3. AWS_EC2 시작하기 - 인스턴스 시작



EC2 서비스 클릭 -> 인스턴스 시작

3. AWS_EC2 시작하기 - AMI 선택

1. AMI 선택 2. 인스턴스 유형 선택 3. 인스턴스 구성 4. 스토리지 추가 5. 태그 추가 6. 보안 그룹 구성 7. 검토

단계 1: Amazon Machine Image(AMI) 선택

AMI는 인스턴스를 시작하는 데 필요한 소프트웨어 구성(운영 체제, 애플리케이션 서버, 애플리케이션)이 포함된 템플릿입니다. AWS, 사용자 커뮤니티 또는 AWS Marketplace에서 제공하는 AMI를 선택하거나, 자체 AMI 중 하나를 선택할 수도 있습니다.

취소 및 종료


빠른 시작

나의 AMI

AWS Marketplace

커뮤니티 AMI


☐ 프리 티어만 ⓘ

**Amazon Linux 2 AMI (HVM), SSD Volume Type** - ami-0f2c95e9fe3f8f80e (64비트 x86) / ami-08fa466ba7f5fe4ed (64비트 Arm)

프리 티어 사용 가능

Amazon Linux 2는 5년간 지원을 제공합니다. Amazon EC2에 성능 최적화된 Linux kernel 4.14와 systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, 최신 소프트웨어 패키지를 추가적으로 제공합니다.


루트 디바이스 유형: ebs 가상화 유형: hvm ENA 활성화: 예

**Ubuntu Server 20.04 LTS (HVM), SSD Volume Type** - ami-04876f29fd3a5e8ba (64비트 x86) / ami-0815e517ed50fd3f4 (64비트 Arm)

프리 티어 사용 가능

Ubuntu Server 20.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

루트 디바이스 유형: ebs 가상화 유형: hvm ENA 활성화: 예

**Ubuntu Server 18.04 LTS (HVM), SSD Volume Type** - ami-0ba5cd124d7a79612 (64비트 x86) / ami-08b051fc14e6c551e (64비트 Arm)

프리 티어 사용 가능

Ubuntu Server 18.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

루트 디바이스 유형: ebs 가상화 유형: hvm ENA 활성화: 예

선택

☒ 64비트(x86)
☐ 64비트(Arm)

선택

☒ 64비트(x86)
☐ 64비트(Arm)

선택

☒ 64비트(x86)
☐ 64비트(Arm)

프리티어

1. 처음 AWS에 가입한 날부터 12개월 동안 사용 가능
2. 주로 사용하게 될 EC2, RDS는 월 별 750시간, 12개월 무료

3. AWS_EC2 시작하기 – 인스턴스 유형 선택

1. AMI 선택 2. 인스턴스 유형 선택 3. 인스턴스 구성 4. 스토리지 추가 5. 태그 추가 6. 보안 그룹 구성 7. 검토

단계 2: 인스턴스 유형 선택

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

필터링 기준: 모든 인스턴스 패밀리 현재 세대 열 표시/숨기기

현재 선택된 항목: t2.micro (- ECU, 1 vCPUs, 2.5 GHz, -, 1 GiB 메모리, EBS 전용)

	그룹	유형	vCPUs	메모리 (GiB)	인스턴스 스토리지 (GB)	EBS 최적화 사용 가능	네트워크 성능	IPv6 지원
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS 전용	-	낮음에서 중간	예
<input checked="" type="checkbox"/>	t2	t2.micro 프리 티어 사용 가능	1	1	EBS 전용	-	낮음에서 중간	예
<input type="checkbox"/>	t2	t2.small	1	2	EBS 전용	-	낮음에서 중간	예
<input type="checkbox"/>	t2	t2.medium	2	4	EBS 전용	-	낮음에서 중간	예
<input type="checkbox"/>	t2	t2.large	2	8	EBS 전용	-	낮음에서 중간	예
<input type="checkbox"/>	t2	t2.xlarge	4	16	EBS 전용	-	보통	예
<input type="checkbox"/>	t2	t2.2xlarge	8	32	EBS 전용	-	보통	예
<input type="checkbox"/>	t3	t3.nano	2	0.5	EBS 전용	예	최대 5 Gbps	예

취소 이전 검토 및 시작 다음: 인스턴스 세부 정보 구성

프리티어 사용 가능한 t2.micro를 선택해야 프리티어가 정상적으로 적용.

3. AWS_EC2 시작하기 – 인스턴스 구성

1. AMI 선택 2. 인스턴스 유형 선택 3. 인스턴스 구성 4. 스토리지 추가 5. 태그 추가 6. 보안 그룹 구성 7. 검토

단계 3: 인스턴스 세부 정보 구성

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

인스턴스 개수 ⓘ 1 Auto Scaling 그룹 시작 ⓘ

구매 옵션 ⓘ ☐ 스팟 인스턴스 요청

네트워크 ⓘ vpc-9fbd06f4 (기본값) ↕ 새 VPC 생성

서브넷 ⓘ 기본 설정 없음(가용 영역의 기본 서브넷) ↕ 새 서브넷 생성

퍼블릭 IP 자동 할당 ⓘ 서브넷 사용 설정(활성화) ↕

배치 그룹 ⓘ ☐ 배치 그룹에 인스턴스 추가

용량 예약 ⓘ 열기 ↕

도메인 조인 디렉터리 ⓘ 디렉터리 없음 ↕ 새 디렉터리 생성

IAM 역할 ⓘ 없음 ↕ 새 IAM 역할 생성

종료 방식 ⓘ 중지 ↕

최대 절전 중지 동작 ⓘ ☐ 추가 종료 동작으로 최대 절전 모드를 활성화

종료 방지 기능 활성화 ⓘ ☐ 우발적인 종료로부터 보호

취소

이전

검토 및 시작

다음: 스토리지 추가

인스턴스 구성은 특별히 설정할 것이 없다면 다음으로 넘어감.

3. AWS_EC2 시작하기 - 스토리지 추가

1. AMI 선택 2. 인스턴스 유형 선택 3. 인스턴스 구성 4. 스토리지 추가 5. 태그 추가 6. 보안 그룹 구성 7. 검토

단계 4: 스토리지 추가

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

볼륨 유형 ⓘ	디바이스 ⓘ	스냅샷 ⓘ	크기(GiB) ⓘ	볼륨 유형 ⓘ	IOPS ⓘ	처리량(MB/초) ⓘ	종료 시 삭제 ⓘ	암호화 ⓘ
루트	/dev/sda1	snap-050dcc44b388532b6	30	범용 SSD(gp2) ▼	100/3000	해당 사항 없음	<input checked="" type="checkbox"/>	암호화되지 않음 ▼

새 볼륨 추가

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

취소

이전

검토 및 시작

다음: 태그 추가

프리티어 최대 30GB 범용 SSD 사용 가능합니다. 지금 설정 안해주면 나중에 서버 내부에서 파티션 설정 까지 해줘야하는 과정이 필요합니다.

3. AWS_EC2 시작하기 – 태그추가

1. AMI 선택 2. 인스턴스 유형 선택 3. 인스턴스 구성 4. 스토리지 추가 5. 태그 추가 6. 보안 그룹 구성 7. 검토

단계 5: 태그 추가

태그는 대소문자를 구별하는 키-값 페어로 이루어져 있습니다. 예를 들어 키가 Name이고 값이 Webserver인 태그를 정의할 수 있습니다.

태그 복사본은 볼륨, 인스턴스 또는 둘 다에 적용될 수 있습니다.

태그는 모든 인스턴스 및 볼륨에 적용됩니다. Amazon EC2 리소스 태그 지정에 대해 [자세히 알아보기](#).

키 (최대 128자)

값 (최대 256자)

인스턴스 ⓘ

볼륨 ⓘ

네트워크 인터페이스 ⓘ

이 리소스에는 현재 태그가 없습니다.

[태그 추가] 버튼 또는 [Name](#) 태그를 추가하려면 클릭합니다. 올(를) 선택합니다.

[IAM 정책](#)에 태그를 생성할 수 있는 권한이 포함되어 있는지 확인합니다.

태그 추가

(최대 50개 태그)

[취소](#)

[이전](#)

[검토 및 시작](#)

[다음: 보안 그룹 구성](#)

태그추가 역시 특별히 설정할 것이 없다면 다음으로 넘어감.

3. AWS_EC2 시작하기 - 보안 그룹 구성

1. AMI 선택 2. 인스턴스 유형 선택 3. 인스턴스 구성 4. 스토리지 추가 5. 태그 추가 6. 보안 그룹 구성 7. 검토

단계 6: 보안 그룹 구성

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

보안 그룹 할당: ☒ 새 보안 그룹 생성

☐ 기존 보안 그룹 선택

보안 그룹 이름: launch-wizard-4

설명: launch-wizard-4 created 2021-05-29T17:19:06.790+09:00

보안 그룹 이름: Mobile-Programming

유형 ⓘ	프로토콜 ⓘ	포트 범위 ⓘ	소스 ⓘ	설명 ⓘ
SSH ▾	TCP	22	사용자 지정 ▾ 0.0.0.0/0	예: SSH for Admin Desktop ✕

규칙 추가



경고

소스가 0.0.0.0/0인 규칙은 모든 IP 주소에서 인스턴스에 액세스하도록 허용합니다. 알려진 IP 주소의 액세스만 허용하도록 보안 그룹을 설정하는 것이 좋습니다.

취소

이전

검토 및 시작

새 보안그룹 생성 -> 검토 및 시작
(저는 보안그룹 이름 변경하였습니다.)

3. AWS_EC2 시작하기 - 키페어 생성

aws 서비스 ▾

서비스, 기능, 마켓플레이스 제품, 설명서 검색 [Alt+S]

Jeonhsu ▾ 서울 ▾ 지원 ▾

1. AMI 선택 2. 인스턴스 유형 선택 3. 인스턴스 구성 4. 스토리지 추가 5. 태그 추가 6. 보안 그룹 구성 7. 검토

단계 7: 인스턴스 시작 검토

인스턴스 시작 세부 정보를 검토하십시오. 이전으로 돌아가서 각 섹션에 대한 변경 내용을 편집할 수 있습니다. 키 페어를 인스턴스에 할당하고 시작 프로세스를 완료하려면 [시작]을 클릭합니다.

인스턴스 보안을 개선하십시오. 보안 그룹 인스턴스를 모든 IP 주소에서 액세스할 수 있습니다. 보...

AMI 세부 정보

Ubuntu Server 20.04 LTS (HVM), SSD Volum...

프리 티어 사용 가능 루트 디바이스 유형: ebs 가상화 유형: hvm

인스턴스 유형

인스턴스 유형	ECU	vCPUs
t2.micro	-	1

보안 그룹

기존 키 페어 선택 또는 새 키 페어 생성

키 페어는 AWS에 저장하는 퍼블릭 키와 사용자가 저장하는 프라이빗 키 파일로 구성됩니다. 이 둘을 모두 사용하여 SSH를 통해 인스턴스에 안전하게 접속할 수 있습니다. Windows AMI의 경우 인스턴스에 로그인하는 데 사용되는 암호를 얻으려면 프라이빗 키 파일이 필요합니다. Linux AMI의 경우, 프라이빗 키 파일을 사용하면 인스턴스에 안전하게 SSH로 연결할 수 있습니다.

참고: 선택한 키 페어가 이 인스턴스에 대해 승인된 키 세트에 추가됩니다. 퍼블릭 AMI에서 기존 키 페어 제거에 대해 자세히 알아보십시오.

새 키 페어 생성

키 페어 이름

MobileProgramming_Keypair

키 페어 다운로드

계속하려면 먼저 프라이빗 키 파일(*.pem 파일)을 다운로드해야 합니다. 액세스할 수 있는 안전한 위치에 저장합니다. 파일은 생성되고 나면 다시 다운로드할 수 없습니다.

취소 인스턴스 시작

MobileProgram....pem

Aws 서버에 접속하기 위해선 KeyPair 라는것을 사용 -> 새 키페어 생성
[키 페어 만으로도 인스턴스에 접속 할 수 있기 때문에 다른 사용자가 접근 할 수 없는 곳에 보관]

3. AWS_EC2 시작하기 - 인스턴스 시작

시작 상태



지금 인스턴스를 시작 중입니다.

다음 인스턴스 시작이 개시됨: i-06ea876e1be620377 [시작 로그 보기](#)



예상 요금 알림 받기

결제 알림 생성 AWS 결제 예상 요금이 사용자가 정의한 금액을 초과하는 경우(예를 들면 프리 티어를 초과하는 경우) 이메일 알림을 받습니다.

인스턴스에 연결하는 방법

인스턴스를 시작 중이며, 사용할 준비가 되어 **실행 중** 상태가 될 때까지 몇 분이 걸릴 수도 있습니다. 새 인스턴스에서는 사용 시간이 즉시 시작되어 인스턴스를 중지 또는 종료할 때까지 계속 누적됩니다.

인스턴스 보기를 클릭하여 인스턴스의 상태를 모니터링합니다. 인스턴스가 **실행 중** 상태가 되고 나면 [인스턴스] 화면에서 인스턴스에 **연결**할 수 있습니다. 인스턴스에 연결하는 방법 [알아보기](#).

▼ 다음은 시작에 도움이 되는 유용한 리소스입니다.

- [Linux 인스턴스에 연결하는 방법](#)
- [Amazon EC2: 사용 설명서](#)
- [AWS 프리 티어에 대해 알아보기](#)
- [Amazon EC2: 토론 포럼](#)

인스턴스가 시작되는 동안 다음을 수행할 수도 있습니다.

- 상태 검사 경보 생성 해당 인스턴스가 상태 검사를 통과하지 못하는 경우 알림을 받습니다. (추가 요금이 적용될 수 있음)
- 추가 **EBS 볼륨** 생성 및 **연결** (추가 요금이 적용될 수 있음)
- **보안 그룹** 관리

인스턴스 시작상태 -> 아래로 스크롤 후 인스턴스 보기를 클릭

3. AWS_EC2 시작하기 - 인스턴스 시작

The screenshot displays the AWS Management Console interface for EC2 instances. The left-hand navigation menu on the left side of the console has the '인스턴스' (Instances) option highlighted with a red rectangular box. The main content area, titled '인스턴스 (1/2) 정보', shows a table of EC2 instances. The table has columns for Name, 인스턴스 ID, 인스턴스 상태, 인스턴스 유형, 상태 검사, 경고 상태, and 가용 영역. Two instances are listed: 'Ubuntu' and 'MobileProgramming'. The 'MobileProgramming' instance is highlighted with a red rectangular box, indicating it is the focus. Its status is '실행 중' (Running), and its type is 't2.micro'. The table also shows that the instance has passed the '2/2개 검사 통과' (2/2 checks passed) and has no alerts ('경고 없음').

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	경고 상태	가용 영역
Ubuntu	i-037ed20b3a7c73ca0	실행 중	t2.micro	2/2개 검사 통과	경고 없음	ap-northeast-2a
MobileProgramming	i-06ea876e1be620377	실행 중	t2.micro	2/2개 검사 통과	경고 없음	ap-northeast-2a

EC2 인스턴스 목록 -> 인스턴스 상태가 [실행중]

4. AWS_EC2 접속하기 - 웹브라우저로 접속하기

The screenshot displays the AWS Management Console interface for EC2 instances. At the top, there's a header with '인스턴스 (1/2) 정보' and buttons for '연결' (Connect), '인스턴스 상태' (Instance State), '작업' (Actions), and '인스턴스 시작' (Start Instance). Below the header is a search bar and a filter for '인스턴스 상태: running'. The main table lists two instances: 'Ubuntu' and 'MobileProgramming'. The 'MobileProgramming' instance is highlighted with a red box, showing its ID 'i-06ea876e1be620377', status '실행 중' (Running), type 't2.micro', and health check '2/2개 검사 통과'. A red box also highlights the '작업' menu, with the '연결' option selected. The bottom section shows the details for the selected instance '인스턴스: i-06ea876e1be620377(MobileProgramming)' with tabs for '세부 정보' (Details), '보안' (Security), '네트워킹' (Networking), '스토리지' (Storage), '상태 검사' (Health Checks), '모니터링' (Monitoring), and '태그' (Tags).

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사
Ubuntu	i-037ed20b3a7c73ca0	실행 중	t2.micro	2/2개 검사 통과
MobileProgramming	i-06ea876e1be620377	실행 중	t2.micro	2/2개 검사 통과

시작할 인스턴스 선택 -> 작업 -> 연결

4. AWS_EC2 접속하기 - 웹브라우저로 접속하기

EC2 > 인스턴스 > i-06ea876e1be620377 > 인스턴스에 연결

인스턴스에 연결 정보

다음 옵션 중 하나를 사용하여 인스턴스 i-06ea876e1be620377 (MobileProgramming)에 연결

EC2 인스턴스 연결

Session Manager

SSH 클라이언트

인스턴스 ID
i-06ea876e1be620377 (MobileProgramming)

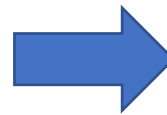
퍼블릭 IP 주소
3.36.114.77

사용자 이름
ubuntu

사용자 지정 사용자 이름을 사용하여 연결하거나 인스턴스 시작에 사용한 AMI의 기본 사용자 이름 ubuntu를(를) 사용합니다.

참고: 대부분의 경우 추정된 사용자 이름은 정확합니다. 하지만 AMI 사용 지침을 읽고 AMI 소유자가 기본 AMI 사용자 이름을 변경했는지 확인하십시오.

취소 **연결**



```
인스턴스에 연결 | EC2 Man x I-06ea876e1be620377 (Mo x Amazon EC2이란 무엇입니 x
ap-northeast-2.console.aws.amazon.com/ec2/v2/connect/ubuntu/i-06ea876e1be6
연 YouTube 지도 뉴스 NAVER SmartPhone Black... Do it! 코틀린 프로...
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-1045-aws x86_64)

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

System information as of Sat May 29 08:59:11 UTC 2021

System load:  0.0               Processes:           100
Usage of /:   4.4% of 29.02GB   Users logged in:    0
Memory usage: 22%              IPv4 address for eth0: 172.31.3.207
Swap usage:   0%

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

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

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

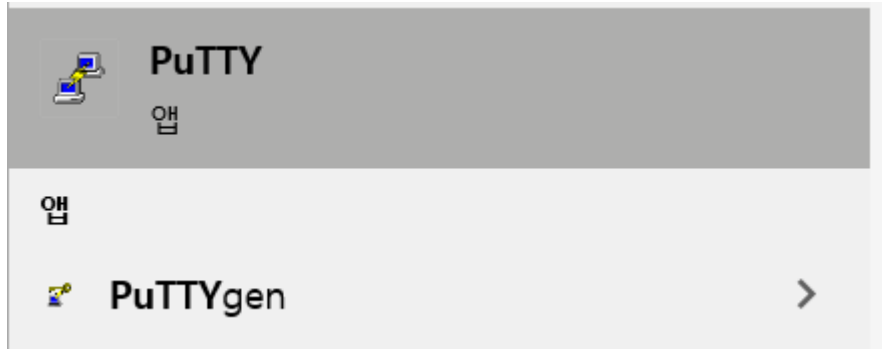
To run a command as administrator (user "root"), use "sudo <command>"
See "man sudo_root" for details.

ubuntu@ip-172-31-3-207:~$
```

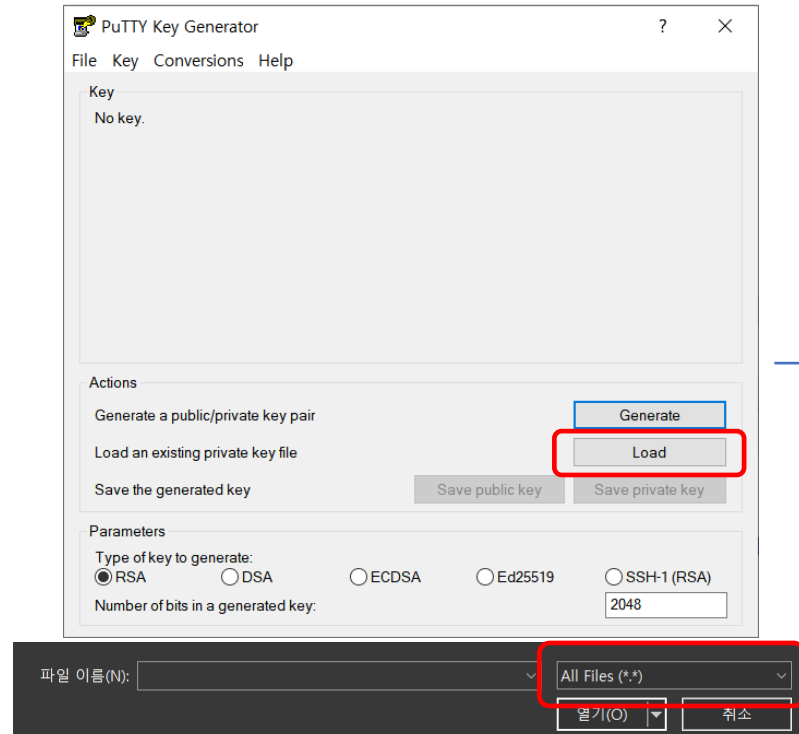
i-06ea876e1be620377 (MobileProgramming)

연결 버튼 클릭 -> EC2 접속

4. AWS_EC2 접속하기 – Putty로 접속하기

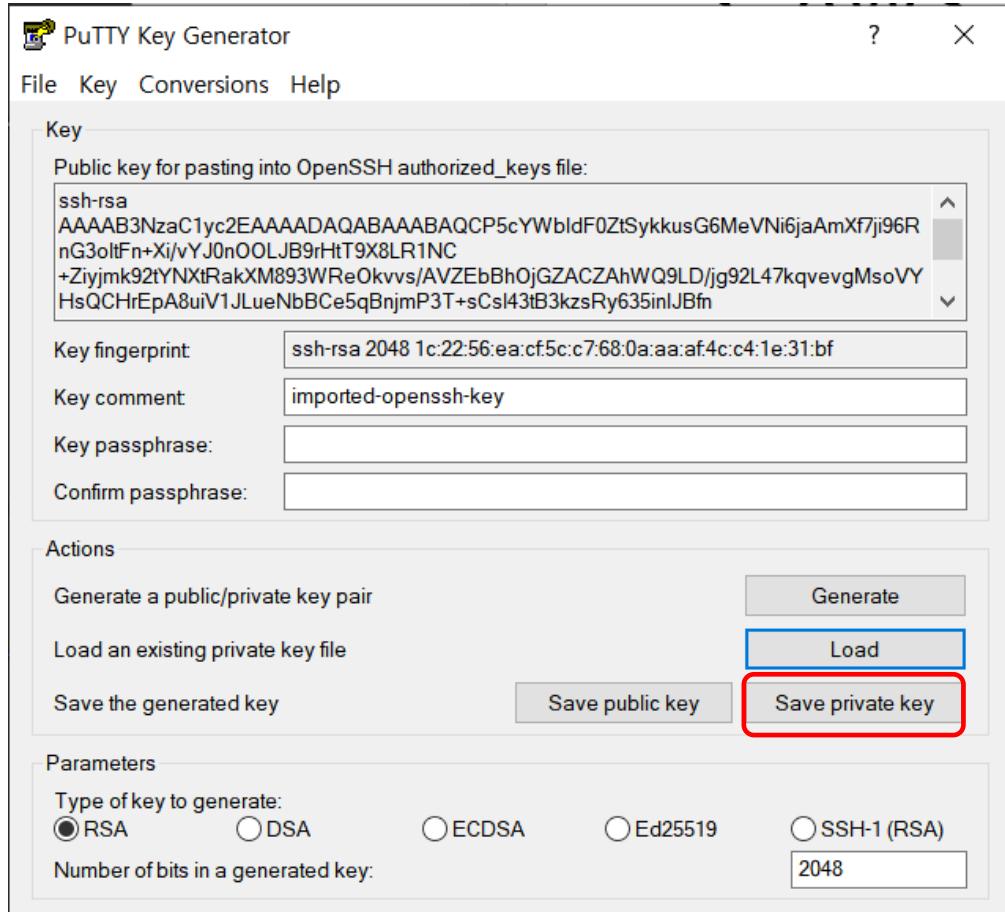


Putty를 다운로드하시면
PuTTY 와 PuTTY gen이 생성됩니다.
먼저 PuTTYgen 열기



[Load] -> [파일 열기]
파일 형식을 All Files로 해야 .pem 파일이
보입니다.

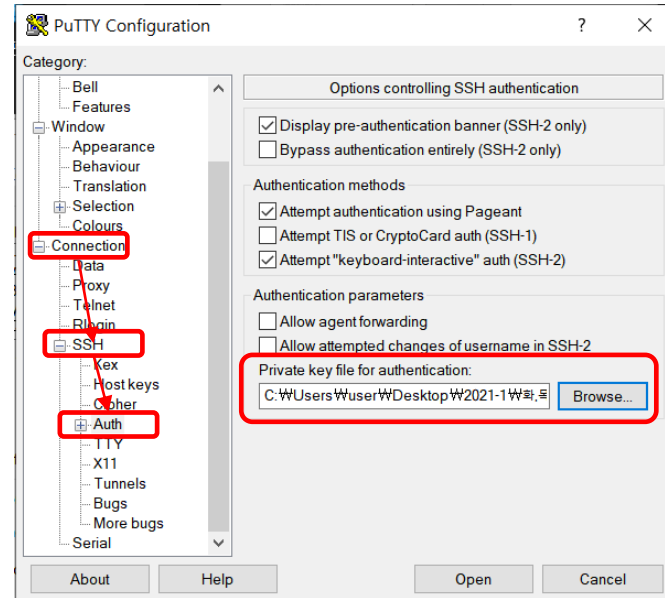
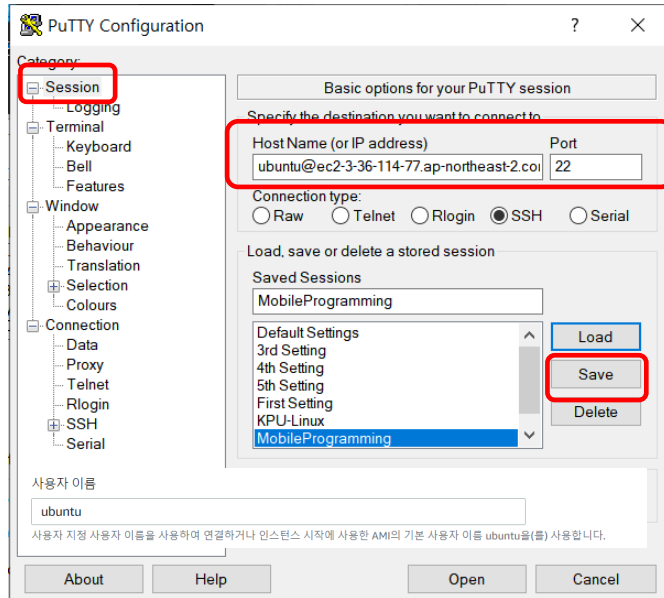
4. AWS_EC2 접속하기 - Putty로 접속하기



[Save private Key] -> [.ppk 파일 생성]

MobileProgramming_Keypair.pem	2021-05-29 오후 5:46	PEM 파일
MobileProgramming_putty.ppk	2021-05-29 오후 6:16	PuTTY Private Key...

4. AWS_EC2 접속하기 - Putty로 접속하기

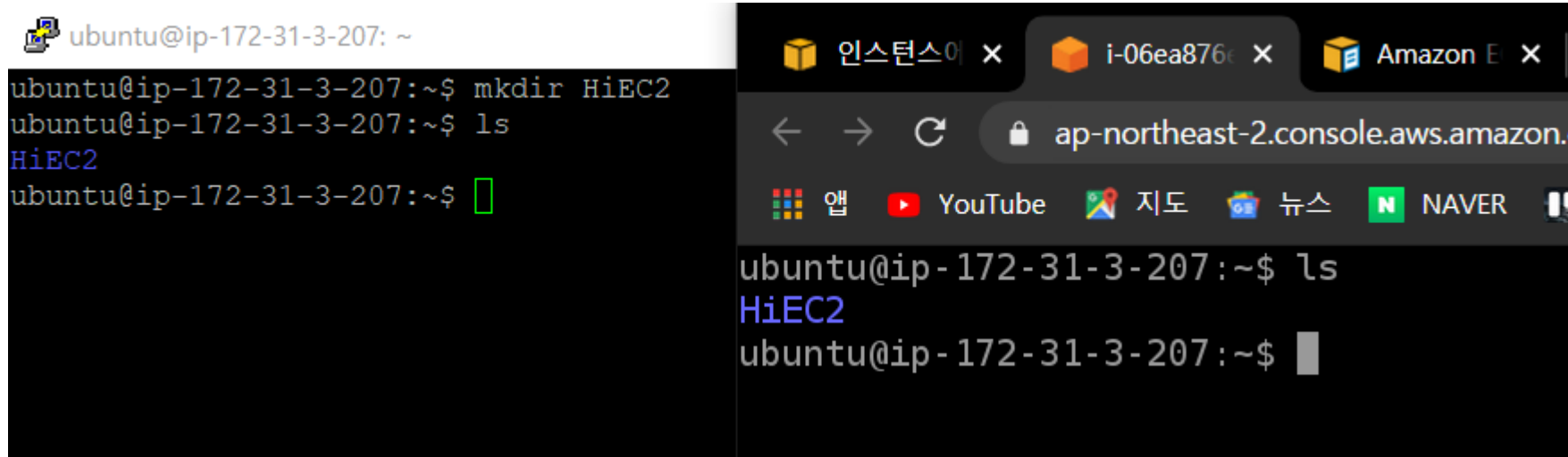


Putty 접속 -> [Session] ->
[Host Name = ubuntu@퍼블릭IP or DNS]
-> [Save]

[Connection] ->
[SSH] ->
[Auth] ->
[Browse..] ->
.ppk 파일 등록

[Session]에서 [Save] 후
[Open] 버튼 클릭
->
알림창이 뜨면 [예]
버튼 클릭

4. AWS_EC2 접속하기 – Putty로 접속하기



The image shows two overlapping windows. The background window is a terminal with the prompt 'ubuntu@ip-172-31-3-207: ~'. It shows the execution of 'mkdir HiEC2', 'ls', and the directory 'HiEC2' being created. The foreground window is a web browser with tabs for '인스턴스', 'i-06ea876', and 'Amazon E'. The address bar shows 'ap-northeast-2.console.aws.amazon.com'. The browser window also displays the same terminal output as the background window.

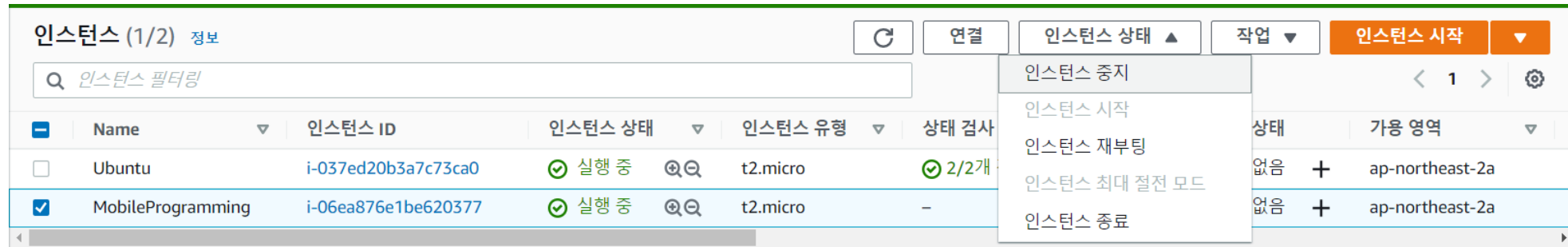
```
ubuntu@ip-172-31-3-207: ~  
ubuntu@ip-172-31-3-207:~$ mkdir HiEC2  
ubuntu@ip-172-31-3-207:~$ ls  
HiEC2  
ubuntu@ip-172-31-3-207:~$
```

EC2_Ubuntu 서버 Putty 정상 연결 확인

오류발생시 해결방안

1. 인스턴스 보안 그룹에서 SSH 접속 설정 확인
2. 개인키가 유효한 파일이 맞는지 확인
3. Host Name 입력 중 실수로 띄어쓰기나 오타가 들어간게 아닌지 확인
4. Host Name 은 직접 인스턴스에 들어가서 확인 해 보세요
(Ubuntu AMI의 경우 사용자 이름은 ubuntu 또는 root)

5. AWS_EC2 - 중지 및 종료



인스턴스 (1/2) 정보

인스턴스 필터링

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사
Ubuntu	i-037ed20b3a7c73ca0	실행 중	t2.micro	2/2개
MobileProgramming	i-06ea876e1be620377	실행 중	t2.micro	-

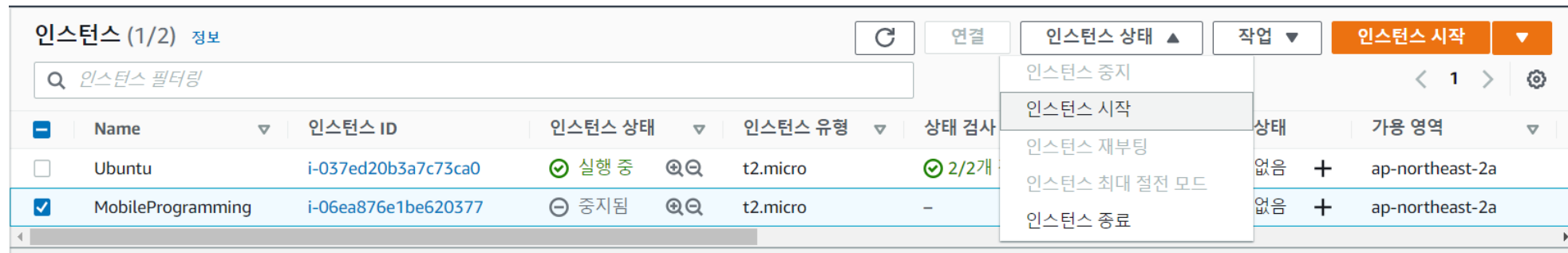
인스턴스 중지
인스턴스 시작
인스턴스 재부팅
인스턴스 최대 절전 모드
인스턴스 종료

인스턴스 시작

상태 가용 영역

없음 + ap-northeast-2a
없음 + ap-northeast-2a

- 인스턴트 중지 (일시정지의 개념입니다. 탄력적 IP설정을 해주지 않으면 재시작시 ip주소 변경)
- 인스턴트 종료 (삭제의 개념입니다. 복구불가)



인스턴스 (1/2) 정보

인스턴스 필터링

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사
Ubuntu	i-037ed20b3a7c73ca0	실행 중	t2.micro	2/2개
MobileProgramming	i-06ea876e1be620377	중지됨	t2.micro	-

인스턴스 중지
인스턴스 시작
인스턴스 재부팅
인스턴스 최대 절전 모드
인스턴스 종료

인스턴스 시작

상태 가용 영역

없음 + ap-northeast-2a
없음 + ap-northeast-2a

- 인스턴스 중지 후 재시작 (시간이 조금 걸립니다.)

Node.js 란

Node.js 특징

- Node.js는 **Single-Thread**의 **non-blocking I/O** 이벤트 기반 비동식 방식으로 작동한다.
- **JavaScript 엔진(V8 Engine)**으로 빌드 된 **JavaScript** 런타임이다.
- 따라서, 사용자의 요청은 한 곳에서 받지만 작업은 실질적으로 멀티쓰레드로 운영하여 결과를 구현한다.

Node.js 장점

- 자바스크립트를 동일하게 사용해서 서버단 로직을 처리할 수 있다는게 가장 큰 장점
새로운 언어를 습득하지 않고도 자바스크립트를 활용해 서버기술을 빨리 향상시킬 수 있다.
- 이벤트 기반 비동기방식이라 서버 무리가 적다.
- npm(node package manager)을 통한 다양한 모듈(패키지) 제공
npm을 이용해 자신이 필요한 라이브러리와 패키지를 검색해서 설치하고 사용할 수 있기 때문에 효율성이 좋다.
- 구글이 만드는 JavaScript 엔진을 사용한다.
구글은 V8 엔진 성능 업그레이드를 계속 하고 있다.
- C++로 개발된 V8 JavaScript 엔진이기 때문에 확장성이 좋다.

6. AWS_EC2 - node.js install

node.js / npm (node package manager)

```
$ sudo apt-get update
```

```
$ curl -sL https://deb.nodesource.com/setup_14.x | sudo bash -E -
```

```
$ sudo apt-get install -y nodejs
```

```
$ node -v
```

```
$ npm -v //npm은 node와 함께 install
```

```
ubuntu@ip-172-31-3-207:~/node$ node -v
v14.17.0
ubuntu@ip-172-31-3-207:~/node$ npm -v
6.14.13
```

7. AWS_EC2 - node.js 시작하기

```
ubuntu@ip-172-31-3-207:~$ mkdir node
ubuntu@ip-172-31-3-207:~$ cd node/
ubuntu@ip-172-31-3-207:~/node$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (node)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /home/ubuntu/node/package.json:
```

```
ubuntu@ip-172-31-3-207:~/node$ ls
package.json
```

\$ npm init // package.json 파일 생성

npm init 명령 후 모두 enter 을 입력하면 자동 default 값으로 설정

7. AWS_EC2 - node.js 시작하기

Node 서버 측 구현 [index.js]

```
$ npm install express --save
```

```
$ npm install multer --save
```

```
$ mkdir uploads
```

```
$ vi index.js
```

8. AWS_EC2 - node.js 코드작성

[index.js]

```
var express = require("express");
var app = express();
var multer,storage;
multer = require('multer');

storage = multer.diskStorage({
  destination: function(req,file,cb){
    cb(null,'./uploads/');},
  filename: function(req,file,cb){
    cb(null, file.originalname);}
});
app.listen(3000, function(){
  console.log('server running..');
});
```

```
app.post("/upload",multer({storage:
storage}).single('upload'),function(req,res){
  console.log(req.file);
  console.log(req.body);

  res.redirect("/uploads/"+req.file.filename);
  console.log(req.file.filename);
  return res.status(200).end();
});
```

8. AWS_EC2 - node.js 코드작성

[index.js]

```
var express = require("express");
var app = express();
var multer, storage;
multer = require('multer');

storage = multer.diskStorage({
  destination: function(req, file, cb) {
    cb(null, './uploads/');
  },
  filename: function(req, file, cb) {
    cb(null, file.originalname);
  }
});

app.listen(3000, function() {
  console.log('server running..');
});

app.post("/upload", multer({storage: storage}).single('upload'), function(req, res) {
  console.log(req.file);
  console.log(req.body);
  res.redirect("/uploads/"+req.file.filename);
  console.log(req.file.filename);
  return res.status(200).end();
});
```

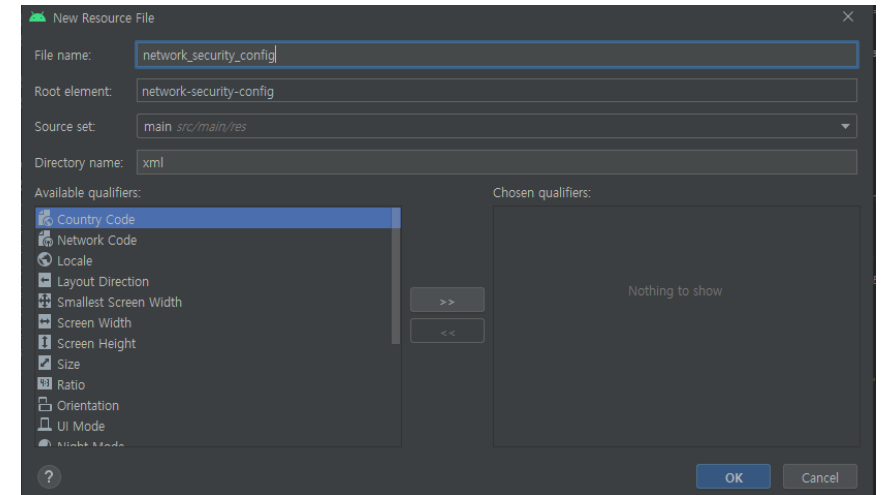
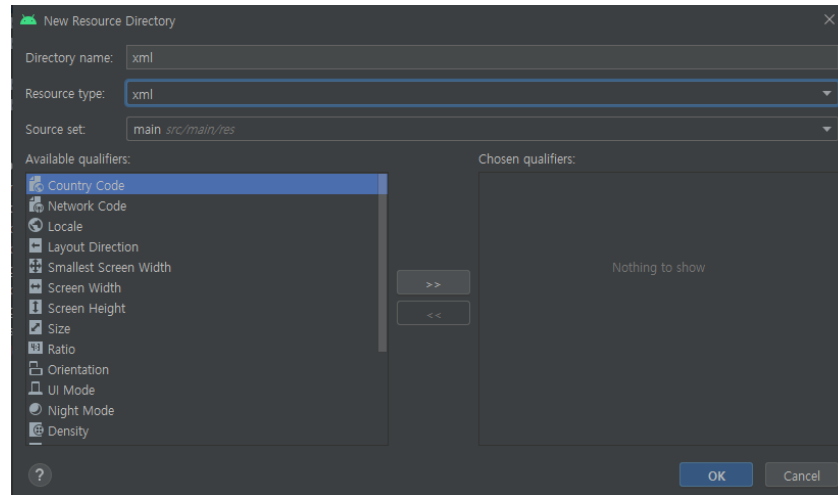
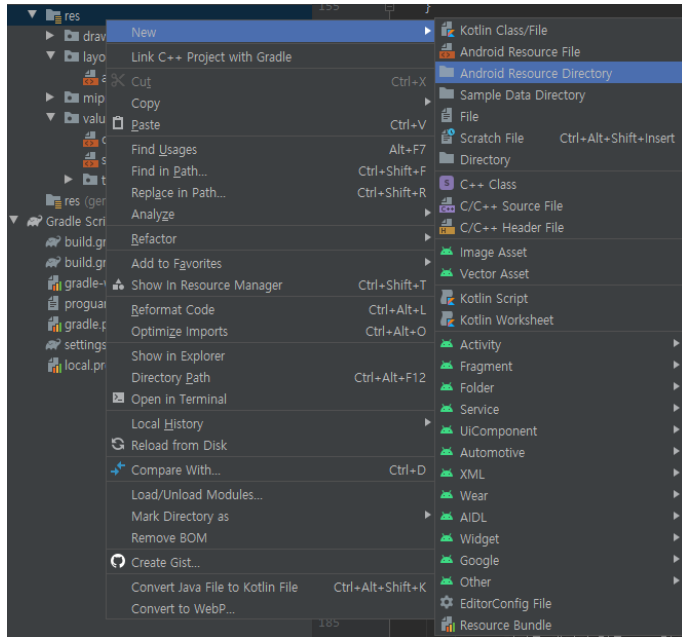
9. Android 측 코드작성

1). Module 수준의 build.gradle : dependencies : retrofit2 모듈 추가

```
dependencies {  
  
    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"  
    implementation 'androidx.core:core-ktx:1.5.0'  
    implementation 'androidx.appcompat:appcompat:1.3.0'  
    implementation 'com.google.android.material:material:1.3.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'  
    testImplementation 'junit:junit:4.+'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'  
  
    //retrofit  
    implementation 'com.squareup.retrofit2:retrofit:2.4.0'  
  
}
```

9. Android 측 코드작성

Network_security_config.xml



9. Android 측 코드작성

Network_security_config.xml

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config xmlns:android="http://schemas.android.com/apk/res/android">
  <domain-config cleartextTrafficPermitted="true">
    <domain includeSubdomains="true">ec2-13-125-162-100.ap-northeast-2.compute.amazonaws.com</domain>
  </domain-config>
</network-security-config>
```

9. Android 측 코드작성

Manifests

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mobile_programming">

    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_MEDIA_LOCATION"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Mobile_Programming"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Mobile_Programming"
        android:requestLegacyExternalStorage="true"
        android:networkSecurityConfig="@xml/network_security_config">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

9. Android 측 코드작성

Activity_main.xml 코드작성

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="@+id/llBottomNav"/>
```

```
<LinearLayout
    android:id="@+id/llBottomNav"
    android:layout_width="0dp"
    android:layout_height="80dp"
    android:background="#96000000"
    android:orientation="horizontal"
    android:gravity="center"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintBottom_toBottomOf="parent">

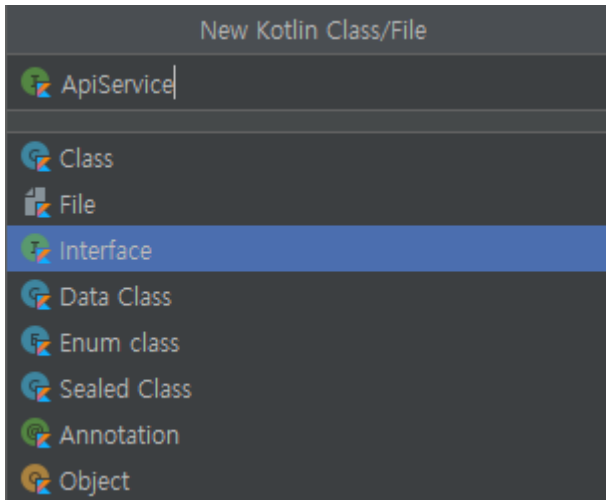
    <Button
        android:id="@+id/btnGallery"
        android:text="imgpic"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="20dp"/>

    <Button
        android:id="@+id/btnUpload"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="imageup"
        android:layout_marginLeft="20dp"/>

</LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

9. Android 측 코드작성

ApiService.kt Interface 코드작성



```
package com.example.mobile_programming

import okhttp3.MultipartBody
import okhttp3.RequestBody
import okhttp3.ResponseBody
import retrofit2.Call
import retrofit2.http.Multipart
import retrofit2.http.POST
import retrofit2.http.Part

interface ApiService {
    @Multipart
    @POST( value: "/upload")
    fun postImage(
        @Part image: MultipartBody.Part?,
        @Part( value: "upload") name: RequestBody?
    ): Call<ResponseBody?>?
}
```

9. Android 측 코드작성

MainActivity.kt 코드작성

```
package com.example.mobile_programming

import android.content.Context
import android.content.Intent
import android.database.Cursor
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.net.Uri
import android.os.Bundle
import android.provider.MediaStore
import android.util.Log
import android.view.View
import android.widget.ImageView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import okhttp3.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import java.io.File
import java.io.FileNotFoundException
import java.io.IOException

//=====
```

9. Android 측 코드작성

MainActivity.kt 코드작성

```
//class start
class MainActivity : AppCompatActivity() {

    private val TAG = "#####"

    //startActivityForResult(Intent intent, int requestCode)를 위한 변수 (activity 식별용)
    private val PICK_FROM_ALBUM = 1

    private var tempFile: File? = null

    private var originalBm: Bitmap? = null

    var apiService: ApiService? = null

    private var filenamesave: File? = null

    //=====
```

9. Android 측 코드작성

onCreate()

```
//=====
// onCreate start
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    ActivityCompat.requestPermissions( activity: this, arrayOf(android.Manifest.permission.WRITE_EXTERNAL_STORAGE),
        Context.MODE_PRIVATE)
    /*-----*/

    val actionBar = supportActionBar
    actionBar!!.hide()
    /*-----*/

    initRetrofitClient()
    /*-----*/

    //IMGPIC 버튼 클릭시 동작
    findViewById<View>(R.id.btnGallery).setOnClickListener { it View!
        goToAlbum()
    }
    /*-----*/

    //IMAGEUP 버튼 클릭시 동작
    findViewById<View>(R.id.btnUpload).setOnClickListener { it View!
        multipartImageUpload()
    }
}
// onCreate end
//=====
```

9. Android 측 코드작성

initRetrofitClient()

```
//=====
//  initRetrofitClient Start
private fun initRetrofitClient() {
    val client = OkHttpClient.Builder().build()
    apiService = Retrofit.Builder()
        .baseUrl(baseUrl: "http://ec2-13-125-162-100.ap-northeast-2.compute.amazonaws.com:3000")
        .client(client).build().create(ApiService::class.java)
}
//=====
```

.baseUrl("http:// [ec2 퍼블릭 IPv4 DNS] :3000")

9. Android 측 코드작성

goToAlbum()

```
//=====
// goToAlbum start
/**
 * 앨범에서 이미지 가져오기
 */
private fun goToAlbum() {
    val intent = Intent(Intent.ACTION_PICK)
    intent.type = MediaStore.Images.Media.CONTENT_TYPE
    startActivityForResult(intent, PICK_FROM_ALBUM)
} //goToAlbum end

//=====
```

9. Android 측 코드작성

onActivityResult()

```
//=====
// onActivityResult Start
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    if (resultCode != RESULT_OK) {
        Log.d(TAG, msg: "onActivityResult에서 resultCode != Activity.RESULT_OK라면 : 실행취소 ")
        Toast.makeText( context: this, text: "취소 되었습니다.", Toast.LENGTH_SHORT).show()

        if (tempFile != null) {
            if (tempFile!!.exists()) {
                if (tempFile!!.delete()) {
                    Log.e(TAG, msg: tempFile!!.getAbsolutePath() + " 삭제 성공")
                    tempFile = null
                }
            }
        }
        return
    }
}
```

```
if (requestCode == PICK_FROM_ALBUM) {
    Log.d(TAG, msg: "onActivityResult에서 requestCode == PICK_FROM_ALBUM이면 : 실행 ")

    // 가져온 데이터 값을 Uri 값으로 변환 -> content// 값 ( 경로로 쓸수 없음 절대경로로 변환 해야함 )
    val photoUri = data!!.data
    Log.d(TAG, msg: "PICK_FROM_ALBUM photoUri : $photoUri")

    var cursor: Cursor? = null
    try {
        /*
         * Uri 스키마를
         * content:/// 에서 file:/// 로 변경한다.
         */
        val proj = arrayOf(MediaStore.Images.Media.DATA)
        Log.d(TAG, msg: "onActivityResult proj : $proj")

        assert(photoUri != null)
        cursor = contentResolver.query(photoUri!!, proj, selection: null, selectionArgs: null, sortOrder: null)
        Log.d(TAG, msg: "onActivityResult cursor : $cursor")
        assert(cursor != null)

        val column_index = cursor!!.getColumnIndexOrThrow(MediaStore.Images.Media.DATA)
        Log.d(TAG, msg: "onActivityResult column_index : $column_index")

        cursor.moveToFirst()
        tempFile = File(cursor.getString(column_index))
        Log.d(TAG, msg: "tempFile Uri : " + Uri.fromFile(tempFile))
        Log.d(TAG, msg: "tempFile : " + tempFile)

        filenamesave = tempFile
    } finally {
        cursor?.close()
    }
    setImage()
} // onActivityResult end
```

9. Android 측 코드작성

setImage()

```
//=====
//  setImage Start
/**
 * tempFile 을 bitmap 으로 변환 후 ImageView 에 설정한다.
 */
private fun setImage() {
    val imageView = findViewById<ImageView>(R.id.imageView)

    //bitmapFactory
    val options = BitmapFactory.Options()

    originalBm = BitmapFactory.decodeFile(tempFile!!.absolutePath, options)

    // tempFile 에서 absolutePath 주소 앞에있던 file:// delete
    Log.d(TAG, msg: "setImage 에서 tempFile.absolutePath를 : " + tempFile!!.absolutePath)

    imageView.setImageBitmap(originalBm)
    /**
     * tempFile 사용 후 null 처리를 해줘야 합니다.
     * (resultCode != RESULT_OK) 일 때 tempFile 을 삭제하기 때문에
     * 기존에 데이터가 남아 있게 되면 원치 않는 삭제가 이뤄집니다.
     */
    tempFile = null
} // setImage end

//=====
```

9. Android 측 코드작성

multipartImageUpload()

```
//=====
// multipartImageUpload start

private fun multipartImageUpload() {
    try {
        /*
        filenamesave = /storage/emulated/0/DCIM/Camera/파일명.확장자
        */

        val reqFile = RequestBody.create(MediaType.parse("image/*"), filenamesave)
        val body = MultipartBody.Part.createFormData("upload", filenamesave!!.name, reqFile)
        val name = RequestBody.create(MediaType.parse("text/plain"), "upload")

        //여기가 전송부분 apiService 인터페이스에 선언해놓은 post 방식으로 이미지를 보냄
        val req = apiService!!.postImage(body, name)

        //이후 처리 부분
        req!!.enqueue(object : Callback<ResponseBody?> {
            override fun onResponse(call: Call<ResponseBody?>, response: Response<ResponseBody?>) {
                Toast.makeText(applicationContext, "Uploaded Successfully!", Toast.LENGTH_SHORT).show()
            }

            override fun onFailure(call: Call<ResponseBody?>, t: Throwable) {
                Toast.makeText(applicationContext, "Request failed", Toast.LENGTH_SHORT).show()
                t.printStackTrace()
            }
        })

    } catch (e: FileNotFoundException) {
        e.printStackTrace()
    } catch (e: IOException) {
        e.printStackTrace()
    }
}

} // multipartImageUpload end

} // class end
```

10. AWS_EC2 – 보안그룹 설정

EC2 > 인스턴스 > i-06ea876e1be620377

i-06ea876e1be620377 (MobileProgramming)에 대한 인스턴스 요약

less than a minute 전에 업데이트됨

🔄

연결

인스턴스 상태 ▼

인스턴스 ID	퍼블릭 IPv4 주소	프라이빗 IPv4 주소
i-06ea876e1be620377 (MobileProgramming)	13.125.162.100 개방 주소법	172.31.3.207
인스턴스 상태	퍼블릭 IPv4 DNS	프라이빗 IPv4 DNS
실행 중	ec2-13-125-162-100.ap-northeast-2.compute.amazonaws.com 개방 주소법	ip-172-31-3-207.ap-northeast-2.compute.internal
인스턴스 유형	탄력적 IP 주소	VPC ID
t2.micro	-	vpc-9fbd06f4
AWS Compute Optimizer 찾기	IAM 역할	서브넷 ID
권장 사항을 위해 AWS Compute Optimizer에 옵트인합니다. 자세히 알아보기	-	subnet-49209522

세부 정보

보안

네트워킹

스토리지

상태 검사

모니터링

태그

▼ 보안 세부 정보

IAM 역할	소유자 ID	시작 시간
--------	--------	-------

10. AWS_EC2 – 보안그룹 설정

세부 정보

보안

네트워킹

스토리지

상태 검사

모니터링


태그

▼ 보안 세부 정보

IAM 역할

-


소유자 ID

 983462001131

시작 시간

Sun May 30 2021 02:06:47 GMT+0900 (대한민국 표준시)

보안 그룹

 sg-044f323c8db6f1dd5 (Mobile-Programming)

▼ 인바운드 규칙

🔍 필터 규칙

< 1 >

포트 범위	프로토콜	원본	보안 그룹
22	TCP	0.0.0.0/0	Mobile-Programming

▶ 아웃바운드 규칙

10. AWS_EC2 – 보안그룹 설정

[EC2](#) > [보안 그룹](#) > sg-044f323c8db6f1dd5 - Mobile-Programming

sg-044f323c8db6f1dd5 - Mobile-Programming

작업 ▼

세부 정보

보안 그룹 이름 ☞ Mobile-Programming	보안 그룹 ID ☞ sg-044f323c8db6f1dd5	설명 ☞ launch-wizard-4 created 2021-05-29T17:19:06.790+09:00	VPC ID ☞ vpc-9fbd06f4 🔗
소유자 ☞ 983462001131	인바운드 규칙 수 1 권한 항목	아웃바운드 규칙 수 1 권한 항목	

인바운드 규칙

아웃바운드 규칙

태그

인바운드 규칙 (1)

인바운드 규칙 편집

유형	프로토콜	포트 범위	소스	설명 - 선택 사항
SSH	TCP	22	0.0.0.0/0	-

10. AWS_EC2 – 보안그룹 설정

EC2 > 보안 그룹 > sg-044f323c8db6f1dd5 - Mobile-Programming > 인바운드 규칙 편집

인바운드 규칙 편집 [정보](#)

인바운드 규칙은 인스턴스에 도달하도록 허용된 수신 트래픽을 제어합니다.

인바운드 규칙 [정보](#)

유형 [정보](#)

SSH ▼

프로토콜 [정보](#)

TCP

포트 범위 [정보](#)

22

소스 [정보](#)

사용자 ... ▼

Q


0.0.0.0/0

X

설명 - 선택 사항 [정보](#)

삭제

규칙 추가

 참고: 기존 규칙을 편집하면 편집된 규칙이 삭제되고 새 세부 정보로 새 규칙이 생성됩니다. 이렇게 하면 새 규칙이 생성될 때까지 해당 규칙에 의존하는 트래픽이 잠시 중단될 수 있습니다.

취소

변경 사항 미리 보기

규칙 저장

10. AWS_EC2 – 보안그룹 설정

인바운드 규칙 편집 정보

인바운드 규칙은 인스턴스에 도달하도록 허용된 수신 트래픽을 제어합니다.

인바운드 규칙 정보

유형 정보

프로토콜 정보

포트 범위 정보

소스 정보

설명 - 선택 사항 정보

SSH ▼

TCP

22

사용자 ... ▼

Q

0.0.0.0/0 ✕

삭제

사용자 지정 TCP ▼

TCP

3000

사용자 ... ▼

Q

0.0.0.0/0 ✕

MobileProgramming_nodePort

삭제

규칙 추가

⚠ 참고: 기존 규칙을 편집하면 편집된 규칙이 삭제되고 새 세부 정보로 새 규칙이 생성됩니다. 이렇게 하면 새 규칙이 생성될 때까지 해당 규칙에 의존하는 트래픽이 잠시 중단될 수 있습니다.

취소

변경 사항 미리 보기

규칙 저장

10. AWS_EC2 – 보안그룹 설정

세부 정보

보안

네트워킹

스토리지

상태 검사

모니터링


태그

▼ 보안 세부 정보


IAM 역할

-

보안 그룹

 sg-044f323c8db6f1dd5 (Mobile-Programming)

소유자 ID

 983462001131

시작 시간

Sun May 30 2021 02:06:47 GMT+0900 (대한민국 표준시)

▼ 인바운드 규칙

Q

필터 규칙

< 1 >

포트 범위	프로토콜	원본	보안 그룹
22	TCP	0.0.0.0/0	Mobile-Programming
3000	TCP	0.0.0.0/0	Mobile-Programming

▶ 아웃바운드 규칙

실행결과

```
ubuntu@ip-172-31-3-207:~$
```

감사합니다