

제 목	02장 AWS EC2에서 도커 활용하기	
상세내용	AWS EC2에서 Docker를 활용해 배포해보기	

1. Ubuntu에서 Docker, Docker Compose 설치하기

✔ Ubuntu에서 Docker, Docker Compose 설치하기

```
$ sudo apt-get update && \
  sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common && \
  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - && \
  sudo apt-key fingerprint 0EBFCD88 && \
  sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" && \
  sudo apt-get update && \
  sudo apt-get install -y docker-ce && \
  sudo usermod -aG docker ubuntu && \
  newgrp docker && \
  sudo curl -L "https://github.com/docker/compose/releases/download/2.27.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose && \
  sudo chmod +x /usr/local/bin/docker-compose && \
  sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

✔ 잘 설치됐는 지 확인하기

```
$ docker -v # Docker 버전 확인
$ docker compose version # Docker Compose 버전 확인
```

2. AWS ECR(Elastic Container Registry)이 뭘까? 왜 배울까?

✓ AWS ECR이 뭘까?

필요한 이미지를 다운로드 받을 때 Dockerhub이라는 곳에서 다운받는다고 했었다. Dockerhub에서는 이미지를 저장 및 다운받을 수 있는 저장소 역할을 한다고도 했다.

Dockerhub과 동일한 역할을 하는 서비스가 하나 더 있다. 그게 바로 AWS ECR이다. AWS ECR도 이미지를 저장 및 다운받을 수 있는 저장소 역할을 한다. 우리는 이 AWS ECR에 대해 배울 것이다.

✓ 왜 Dockerhub 대신에 AWS ECR을 사용하는가?

최근에는 AWS 클라우드 환경에서 인프라를 구축하는 일이 많아졌다. AWS ECR을 사용하면 다른 AWS Resource와의 연동이 편하고, AWS 내에서 한 번에 관리할 수 있기에 편하다는 장점이 있다.

(물론, Dockerhub을 사용해도 크게 문제는 없다.)

✓ AWS ECR을 왜 배우는지?



Docker를 사용하지 않았을 때 많은 사람들이 사용하는 배포 전략 중 하나는 Github을 활용하는 방법이다. 프로젝트 코드를 Github에 Push 한 뒤에, AWS EC2에 접속해서 해당 코드를 Pull 받아서 실행시키는 방식을 많이 사용한다. 이 방식은 프로젝트 코드 전체를 EC2로 이동시켜야 하며, 프로젝트 코드를 실행시킬 런타임 환경(Node, JDK 등)도 설치되어 있어야만 실행이 된다.



Docker의 가장 큰 장점은 ****이식성****이다. ****Docker만 깔려있으면 어디에서든 내가 원하는 프로젝트를 실행시킬 수 있다는 게 장점****이다. 이 때 Github을 활용해 프로젝트 코드 전체를 EC2로 옮겨 Docker 기반으로 실행시켜도 된다. 하지만 프로젝트에서 필요한 코드에 대해서만 Docker 이미지로 빌드해, EC2에서는 그 이미지만 다운로드해서 실행시키는 게 훨씬 심플하다.

정리하자면 ****AWS ECR을 배우는 이유는 훨씬 간단하게 프로젝트를 배포하고 실행시키기 위해서이다.****

[실습] AWS ECR(Elastic Container Registry) 사용해보기

✓ AWS CLI 설치

참조: https://docs.aws.amazon.com/ko_kr/cli/latest/userguide/getting-started-install.html

[우분투(Ubuntu)]

```

$ sudo apt install unzip
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
$ unzip awscliv2.zip
$ sudo ./aws/install
$ aws --version # 잘 출력된다면 정상 설치된 상태
  
```

[맥(Mac OS)]

```

$ brew install awscli
$ aws --version # 잘 출력된다면 정상 설치된 상태
  
```

[윈도우(Windows)]

1. 이 링크(<https://awscli.amazonaws.com/AWSCLIV2.msi>)를 다운받아 설치하기
2. cmd를 실행시켜서 아래 명령어 입력해보기

```
$ aws --version # 잘 출력된다면 정상 설치된 상태
```

✔ IAM 생성하기

1. IAM에서 사용자 생성하기

The screenshot shows the 'Create New User' page in the AWS IAM console. The left sidebar indicates the current step is '1단계 사용자 세부 정보 지정' (Step 1: Set user details). The main area is titled '사용자 세부 정보 지정' (Set user details). Under '사용자 세부 정보' (User details), the '사용자 이름' (User name) field is filled with 'my-computer'. Below this, a note states: '사용자 이름은 최대 64자까지 가능합니다. 유효한 문자: A~Z, a~z, 0~9 및 +, =, ., @, _ (하이픈)'. There is an unchecked checkbox for 'AWS Management Console에 대한 사용자 액세스 권한 제공 - 선택 사항' (Provide user access to the AWS Management Console - optional). A blue information box at the bottom states: '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'. At the bottom right are '취소' (Cancel) and '다음' (Next) buttons.

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.

권한 정책 (1/1204) 정책 생성

새 사용자에게 연결할 정책을 하나 이상 선택합니다.

필터링 기준 유형

container 모든 유형 13 개 일치

정책 이름	유형	연결된 엔터티
<input checked="" type="checkbox"/> AmazonEC2ContainerRegistryFullAccess	AWS 관리형	1
<input type="checkbox"/> AmazonEC2ContainerRegistryPowerUser	AWS 관리형	0

검토 및 생성

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

사용자 세부 정보

사용자 이름
my-computer

콘솔 암호 유형
None

암호 재설정 필요
아니요

권한 요약

이름

유형

다음과 같이 사용

[AmazonEC2ContainerRegistryFullAccess](#)

AWS 관리형

권한 정책

태그 - 선택 사항

태그는 리소스를 식별, 구성 또는 검색하는 데 도움이 되도록 AWS 리소스에 추가할 수 있는 키 값 페어입니다. 이 사용자와 연결할 태그를 선택합니다.

리소스와 연결된 태그가 없습니다.

새 태그 추가

최대 50개의 태그를 더 추가할 수 있습니다.

취소

이전

사용자 생성

2. Access Key 발급받기

Identity and Access Management(IAM)

IAM 검색

대시보드

- ▼ 액세스 관리
 - 사용자 그룹
 - 사용자**
 - 역할
 - 정책
 - 자격 증명 공급자
 - 계정 설정
- ▼ 보고서 액세스
 - 액세스 분석기
 - 아카이브 규칙
 - 분석기
 - 설정

MFA 디바이스가 없습니다. MFA 디바이스를 할당하여 AWS 환경 보안 개선하기

[MFA 디바이스 할당](#)

액세스 키 (0)

액세스 키를 사용하여 AWS CLI, AWS Tools for PowerShell, AWS SDK 또는 직접 AWS API 호출을 통해 AWS에 프로그래밍 방식 호출을 전송합니다. 한 번에 최대 두 개의 액세스 키(활성 또는 비활성)를 가질 수 있습니다. 자세히 알아보기

[액세스 키 만들기](#)

액세스 키 없음

액세스 키와 같은 장기 자격 증명을 사용하지 않는 것이 모범 사례입니다. 대신 단기 자격 증명을 제공하는 도구를 사용하세요. 자세히 알아보기

[액세스 키 만들기](#)

AWS CodeCommit에 대한 SSH 퍼블릭 키 (0)

AWS CodeCommit 리포지토리에 대한 액세스를 인증하는 사용자 SSH 퍼블릭 키입니다. 한 번에 최대 5개의 SSH 퍼블릭 키(활성 또는 비활성)를 보유할 수 있습니다. 자세히 알아보기

[작업](#) [SSH 퍼블릭 키 업로드](#)

IAM > 사용자 > jscode-backend-server > 액세스 키 만들기

1단계
액세스 키 모범 사례 및 대안

2단계 - 선택 사항
설명 태그 설정

3단계
액세스 키 검색

액세스 키 모범 사례 및 대안

보안 개선을 위해 액세스 키와 같은 장기 자격 증명을 사용하지 마세요. 다음과 같은 사용 사례와 대안을 고려하세요.

☐ **Command Line Interface(CLI)**
AWS CLI를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ **로컬 코드**
로컬 개발 환경의 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ **AWS 컴퓨팅 서비스에서 실행되는 애플리케이션**
Amazon EC2, Amazon ECS 또는 AWS Lambda와 같은 AWS 컴퓨팅 서비스에서 실행되는 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ **서드 파티 서비스**
AWS 리소스를 모니터링 또는 관리하는 서드 파티 애플리케이션 또는 서비스에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☒ **AWS 외부에서 실행되는 애플리케이션**
애플리케이션을 온프레미스 호스트에서 실행하거나 로컬 AWS 클라이언트 또는 서드 파티 AWS 플러그 인을 사용할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ **기타**
귀하의 사용 사례가 여기에 나열되어 있지 않습니다.

이 사용 사례에서는 액세스 키를 사용해도 되지만 모범 사례를 따릅니다.

- 액세스 키를 일반 텍스트, 코드 리포지토리 또는 코드로 저장해서는 안 됩니다.
- 더 이상 필요 없는 경우 액세스 키를 비활성화하거나 삭제합니다.
- 최소 권한을 활성화합니다.
- 액세스 키를 정기적으로 교체합니다.

액세스 키 관리에 대한 자세한 내용은 [AWS 액세스 키 관리 모범 사례](#)를 참조하세요.

취소 다음

IAM > 사용자 > jscode-backend-server > 액세스 키 만들기

1단계
액세스 키 모범 사례 및 대안

2단계 - 선택 사항
설명 태그 설정 - 선택 사항

3단계
액세스 키 검색

설명 태그 설정 - 선택 사항

이 액세스 키에 대한 설명은 이 사용자에게 태그로 연결되고, 액세스 키와 함께 표시됩니다.

설명 태그 값

이 액세스 키의 용도와 사용 위치를 설명합니다. 좋은 설명은 나중에 이 액세스 키를 자신있게 교체하는 데 유용합니다.

최대 256자까지 가능합니다. 허용되는 문자는 문자, 숫자, UTF-8로 표현할 수 있는 공백 및 _ . : / = * - @입니다.

취소 이전 **액세스 키 만들기**

IAM > 사용자 > jscode-backend-server > 액세스 키 만들기

1단계
액세스 키 모범 사례 및 대안

2단계 - 선택 사항
설명 태그 설정

3단계
액세스 키 검색

액세스 키 검색

액세스 키
본실하거나 잊어버린 비밀 액세스 키는 검색할 수 없습니다. 대신 새 액세스 키를 생성하고 이전 키를 비활성화합니다.

액세스 키	비밀 액세스 키
AKIAQBAOIYCI6V2WV7J	***** 표시

액세스 키 모범 사례

- 액세스 키를 일반 텍스트, 코드 리포지토리 또는 코드로 저장해서는 안 됩니다.
- 더 이상 필요 없는 경우 액세스 키를 비활성화하거나 삭제합니다.
- 최소 권한을 활성화합니다.
- 액세스 키를 정기적으로 교체합니다.

액세스 키 관리에 대한 자세한 내용은 [AWS 액세스 키 관리 모범 사례](#)를 참조하세요.

.csv 파일 다운로드 완료

3. AWS CLI로 액세스 키 등록하기

```
$ aws configure
```

AWS Access Key ID [None]: <위에서 발급한 Key id>

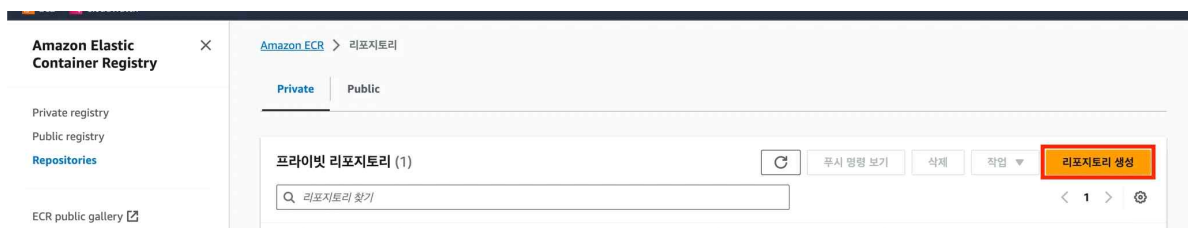
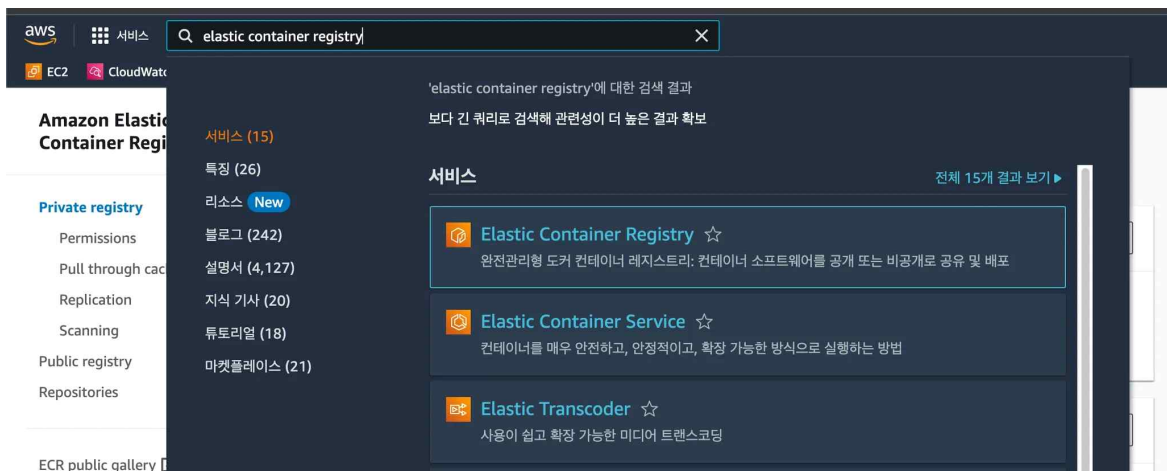
AWS Secret Access Key [None]: <위에서 발급한 Secret Access Key>

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

Default output format [None]:

✓ AWS ECR(Elastic Container Registry) 셋팅하기

▶ Docker 이미지를 저장할 수 있는 저장소를 만들어보자.



일반 설정

표시 여부 설정 **정보**
리포지토리에 대한 가시성 설정을 선택합니다.

☒ 프라이빗
액세스는 IAM 및 리포지토리 정책 권한에 의해 관리됩니다.

☐ 퍼블릭
이미지 풀에 대해 공개적으로 표시되고 액세스할 수 있습니다.

리포지토리 이름
간결한 이름을 제공합니다. 개발자는 이름으로 리포지토리 콘텐츠를 식별할 수 있어야 합니다.

002177417362.dkr.ecr.ap-northeast-2.amazonaws.com/

최대 256자 중 16자(최소 2자 이상) The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

태그 변경 불가능 **정보**
동일한 태그를 사용하는 후속 이미지 푸시가 이미지 태그를 덮어쓰지 않도록 방지하려면 [태그 변경 불가능]을 활성화합니다. 이미지 태그를 덮어쓰려면 [태그 변경 불가능]을 비활성화합니다.

☐ 비활성화됨

리포지토리가 생성되면 해당 리포지토리의 가시성 설정을 변경할 수 없습니다.

- 일반적으로 하나의 리포지토리에는 한 종류의 이미지만 저장하고 관리한다.

✓ 이미지 빌드해서 AWS ECR에 Push, Pull 해보기

1. Dockerfile 작성하기

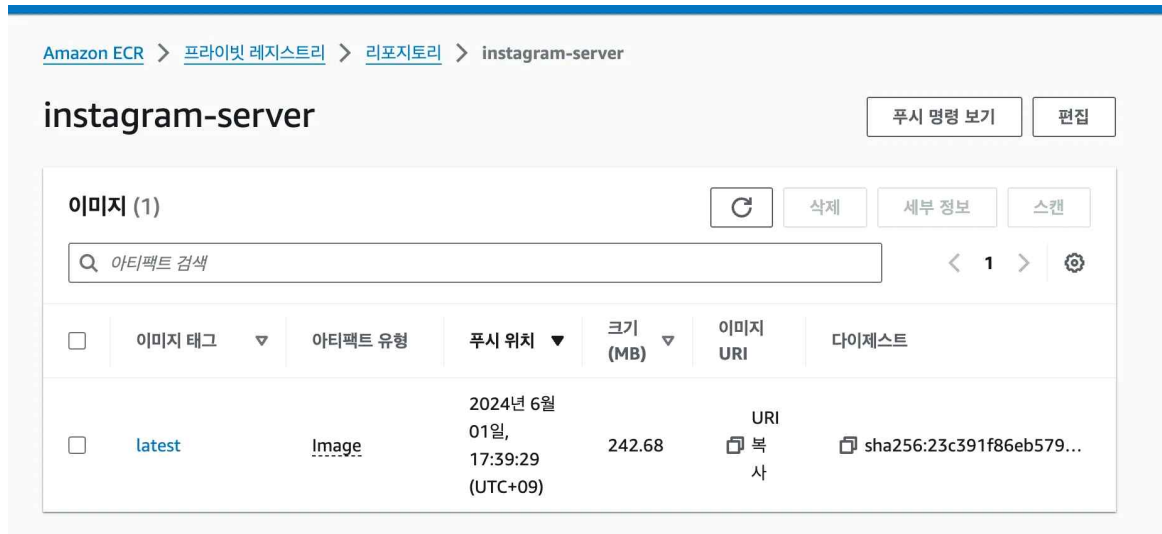
```
FROM openjdk:17-jdk

ENTRYPOINT ["/bin/bash", "-c", "sleep 500"]
```

2. 이미지 빌드 및 push 하기

이미지 Push할 때 어떤 명령어를 써야 하는 지 가르쳐주는 위치

```
$ aws ecr get-login-password --region ap-northeast-2 | docker login --username AWS --password-stdin 002177417362.dkr.ecr.ap-northeast-2.amazonaws.com
$ docker build -t instagram-server .
$ docker tag instagram-server:latest 002177417362.dkr.ecr.ap-northeast-2.amazonaws.com/instagram-server:latest
$ docker push 002177417362.dkr.ecr.ap-northeast-2.amazonaws.com/instagram-server:latest
```

3. 이미지 Pull 받아보기



```
$ docker image rm -f [Container ID] # 기존 갖고있던 이미지 지우기
$ docker pull 002177417362.dkr.ecr.ap-northeast-2.amazonaws.com/instagram-server
$ docker image ls
```

- **002177417362.dkr.ecr.ap-northeast-2.amazonaws.com/instagram-server** : 이 값 자체가 이미지 이름이다. 길어서 어색해보일 뿐이다.