

## 클라우드 환경의 이해

개발자가 되기 전에 알았으면 좋았을 것들



## 클라우드 컴퓨팅

---

- **클라우드 컴퓨팅의 배경**

- 하드웨어 성능의 비약적인 성장
  - CPU, 스토리지, 네트워크 장치 등 컴퓨터를 이루는 주요 장치들의 성능이 과거에 비하여 크게 성장을 이루어 네트워크를 통해 실시간으로 자원을 제공하는 것이 가능해졌다.
- 네트워크 기술의 발전
  - 네트워크 기술이 발전함에 따라, 언제 어디서나 원격으로 자원에 접근할 수 있게 되었다.
- 가상화 기술의 발전
  - 가상화 기술이 발전하면서 여러 사용자가 물리적 자원(하드웨어)를 공유하여 사용하고 필요한 만큼 자원을 할당받을 수 있게 되었다.

## 클라우드 컴퓨팅

- 클라우드 컴퓨팅의 발전과정



## 클라우드 컴퓨팅

---

- **클라우드 컴퓨팅의 유형**

- 퍼블릭 클라우드 (Public Cloud)
  - CSP(Cloud Service Provider)가 제공하는 일반적인 형태의 클라우드 환경
- 프라이빗 클라우드 (Private Cloud)
  - 특정 조직을 위한 전용 클라우드 환경으로, 보안성을 강화할 수 있다.
  - 온프레미스로 구성을 할 수도 있고, CSP를 통해 호스팅을 맡길 수 있다.
  - 클라우드 환경 자체를 관리할 인력이 필요하다.
- 하이브리드 클라우드 (Hybrid Cloud)
  - 퍼블릭 클라우드와 프라이빗 클라우드를 결합하여 사용하는 형태로, 각각의 이점을 모두 활용할 수 있다.
  - 구현과 운영의 난이도가 높고, 자칫 잘못하면 각 클라우드의 관리 체계가 나뉘어져 하이브리드로서의 장점을 잃게 될 수 있다.
- 멀티 클라우드 (Multi Cloud)
  - 여러 CSP의 서비스를 혼합하여 운영하는 형태이다.
  - 서비스별로 유리한 업체를 선정하여 비용을 최적화할 수 있고, 클라우드 환경 자체에서 발생할 수 있는 위험을 분산 할 수 있다.
  - 다양한 클라우드 환경으로 이루어져 있어 관리의 어려움이 있다.

## 클라우드 컴퓨팅

---

- **클라우드 컴퓨팅의 모델**

- IaaS (Infrastructure as a Service, IaaS)

- 가상화된 컴퓨팅 자원(컴퓨팅, 스토리지, 네트워크 등)을 제공하여 사용자가 인프라를 직접 구성하고 관리할 수 있도록 제공한다.
    - ex) AWS EC2, Azure VM, GCP GCE

- PaaS (Platform as a Service, PaaS)

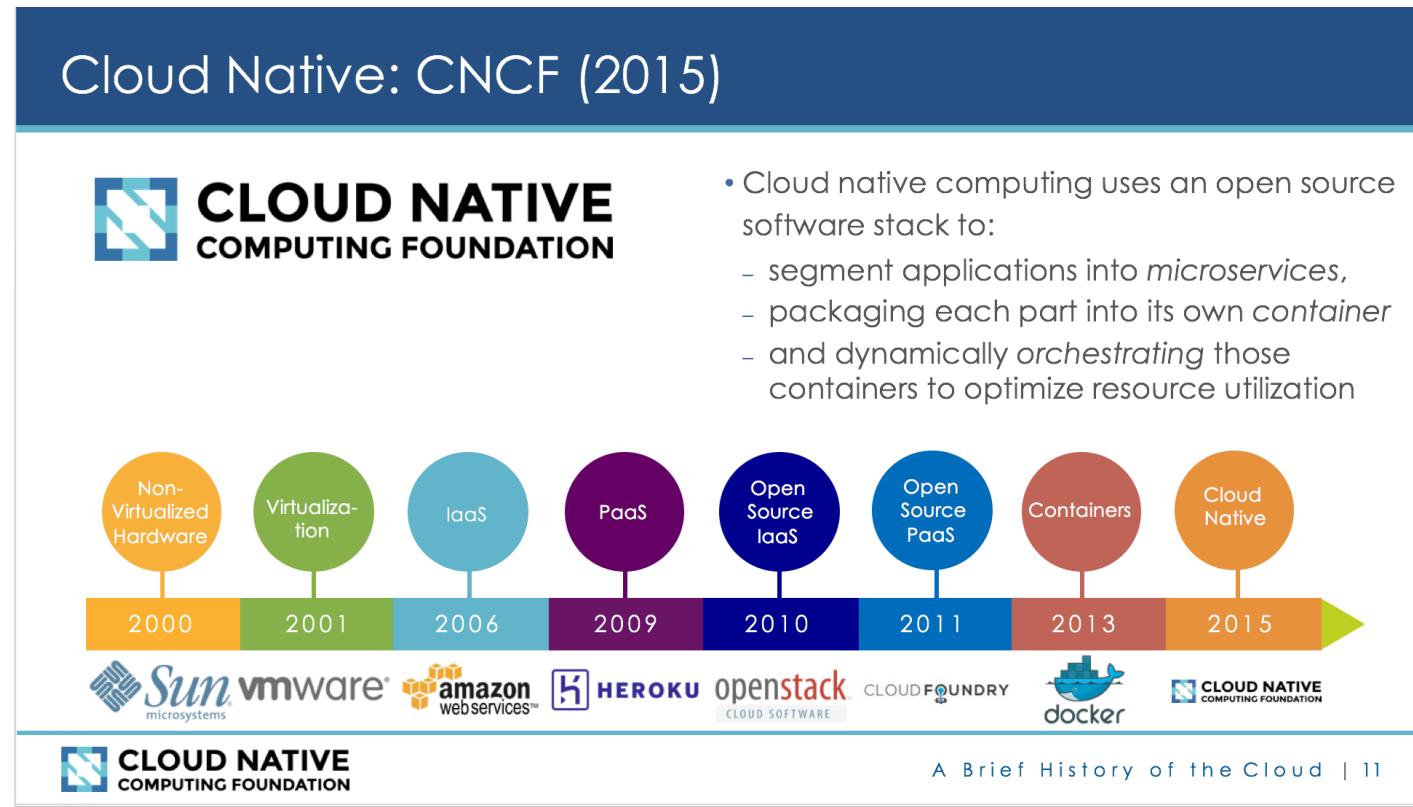
- 애플리케이션을 개발, 운영하는데 필요한 플랫폼과 환경을 제공한다. 직접 인프라를 구성하거나 관리할 필요가 없어 개발에 집중할 수 있다.
    - ex) AWS Elastic Beanstalk, Azure App Services, GCP App Engine, Vercel, Heroku

- SaaS (Software as a Service, SaaS)

- 소프트웨어 애플리케이션을 클라우드 환경을 통해 제공하며 별도의 설치, 유지보수 없이 애플리케이션을 사용할 수 있다.
    - ex) 인터넷을 통해 사용할 수 있는 대부분의 서비스

## 클라우드 네이티브

- 클라우드 네이티브란?
  - 클라우드 환경의 이점을 최대한 활용하여 아키텍처를 설계하고 애플리케이션을 개발하여 서비스를 운영하는 방법론



클라우드 네이티브로의 발전 과정

## 클라우드 네이티브

- **클라우드 네이티브를 구성하는 요소**

- 컨테이너
  - 애플리케이션과 환경을 컨테이너로 만들어 실행 환경 간의 일관성을 보장한다.
  - 컴퓨팅의 최소 자원을 컨테이너를 기준으로 하여 자원을 효율적으로 사용할 수 있다.
  - 애플리케이션 환경을 경량화하여 빠른 배포가 가능하다.
- 마이크로서비스 아키텍처(MSA, Micro Service Architecture)
  - 애플리케이션을 기능을 중심으로 작은 단위의 서비스로 분할한다.
    - 결합도를 낮추는 효과를 가지며, 장애가 발생하더라도 애플리케이션의 동작이 중지하지는 않는다.
    - 서비스에 적합한 환경(기술스택)을 기능에 맞게 선택하여 적용할 수 있다.
- 지속적 통합 및 배포(CI/CD, Continuous Integration/Continuous Deployment)
  - 지속적 통합(CI, Continuous Integration)
    - 코드의 변경 사항을 감지하여 자동화된 파이프라인을 통해 빌드 및 테스트를 진행합니다
  - 지속적 배포(CD, Continuous Deployment)
    - 코드의 변경사항을 애플리케이션 운영 환경에 반영합니다.

## 클라우드 네이티브

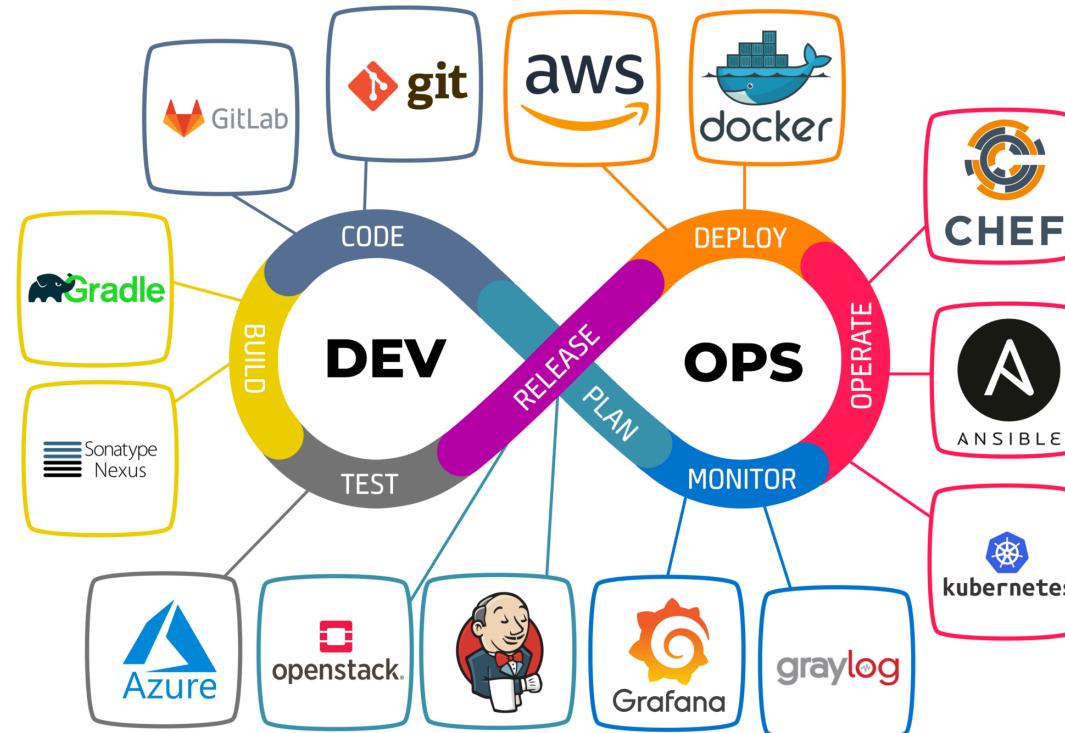
---

- **클라우드 네이티브를 구성하는 요소**

- DevOps(Development + Operations)
  - 문자 그대로 개발 조직과 운영 조직의 결합
  - 분리되어 있는 두 개의 조직 사이에서 발생하던 사일로 현상을 최소화하고, 소프트웨어 개발 주기를 단축시키는 방법론
  - 이를 통해 고객에게 지속적으로 가치를 제공할 수 있다.
    - 고객의 피드백을 받아 빠르게 제품을 개선하나가는 애자일 방법론과 비슷한 흐름을 갖고 있다.

## 클라우드 네이티브

- 클라우드 네이티브를 구성하는 요소



## DevOps ++

---

- **SRE(Site Reliability Engineering, 사이트 신뢰성 엔지니어링)**

- 안정적인 서비스 운영을 지원하기 위한 방법론으로 구글에서 개발되었다.
- SRE에서는 시스템의 신뢰성과 안정성을 강조한다.
- 주요 업무 및 목적
  - 모니터링
    - 안정적인 서비스 운영을 위해 애플리케이션의 성능 및 오류 등을 모니터링하여 개발 조직과의 협업을 통해 개선한다.
  - 자동화
    - 반복적으로 수행되는 작업을 자동화하여 효율성을 높이고, 시스템 운영에 대한 부담을 줄인다.

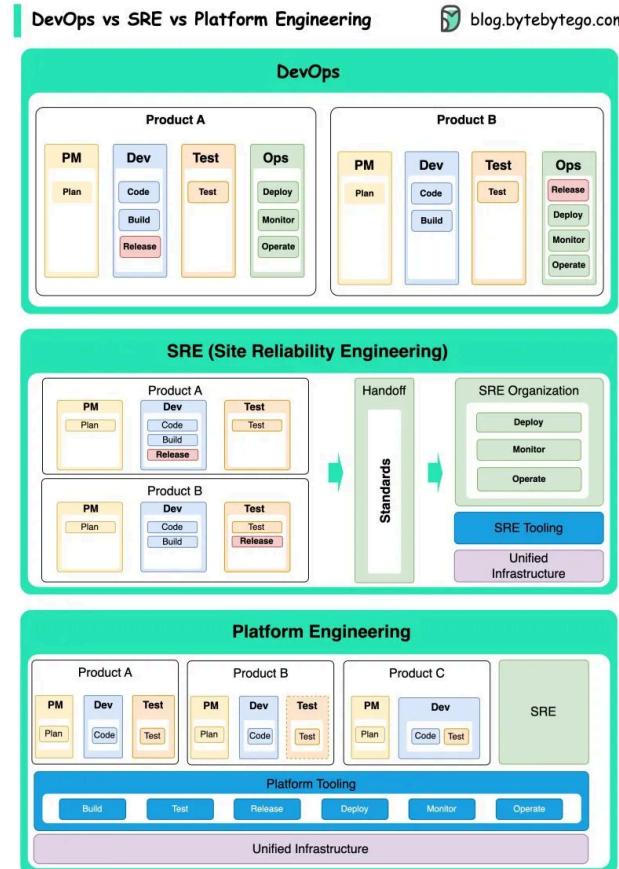
## DevOps ++

---

- **플랫폼 엔지니어링(Platform Engineering)**

- 제품 개발 조직이 개발에 집중할 수 있도록 플랫폼을 구축하고 관리한다.
  - 여기서 **플랫폼**은 인프라 구축, 배포 자동화, 서비스 모니터링 등을 개발자들이 쉽게 이용할 수 있도록 추상화시킨 것을 의미한다. IDP(Internal Developer Platform)이라고도 부른다.
- *DevOps의 최종진화*라고 부르기도 한다.
- 주요 업무 및 목적
  - 제품 개발 조직의 업무 효율성 향상
    - 개발 조직에서 인프라와 다양한 도구들을 쉽게 사용할 수 있도록 추상화하여 지원하여 개발 속도를 향상 시킨다.

## DevOps ++



## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- **Github**

- git 을 기반으로 개발자들의 협업을 위해 분산 버전 관리를 지원해주는 서비스

- **AWS Codepipeline**

- 소프트웨어를 빌드하고, 테스트하고, 배포하는 일련의 과정을 하나로 통합하고 소스 코드의 변경점을 자동으로 감지하여 자동으로 배포하는 서비스

- **AWS CodeBuild**

- 소스 코드를 컴파일, 빌드하고 테스트를 실행하여 CI 서비스
    - 비슷한 도구로 Jenkins, Bamboo, Travis CI 등이 있다.

- **AWS Elastic Container Registry(ECR)**

- 컨테이너 이미지를 저장하는 원격 레지스트리

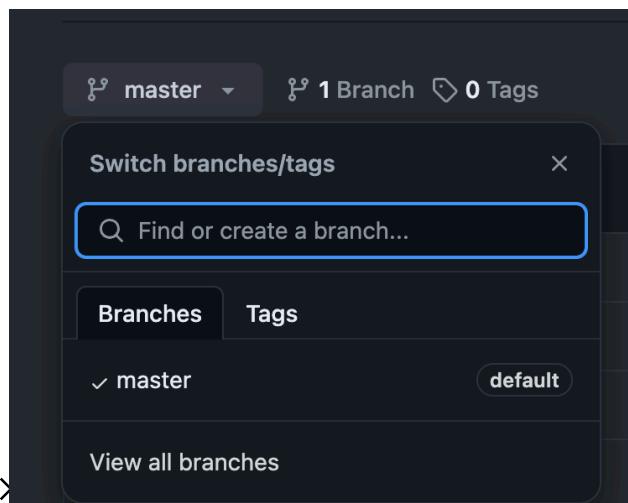
- **AWS Elastic Container Service(ECS)**

- 컨테이너화된 애플리케이션을 손쉽게 배포, 관리할 수 있도록 지원하는 완전 관리형 컨테이너 오케스트레이션 도구

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- Github

- 이번 실습에서 파이프라인의 시작점
- <https://github.com/rlatjdwn4926/ktb-repo> 를 기준으로 진행
  - 빠대 그대로 진행을 원하시면 fork 후 진행하시면 됩니다.
  - 별도의 테스트용 코드를 작성하셔도 무방합니다.
- 어떤 Branch를 기준으로 진행할 것인지 확인 필수
  - 자료는 master 를 기준으로 진행하였으나 기본 Branch는 main 으로 생성됨



## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

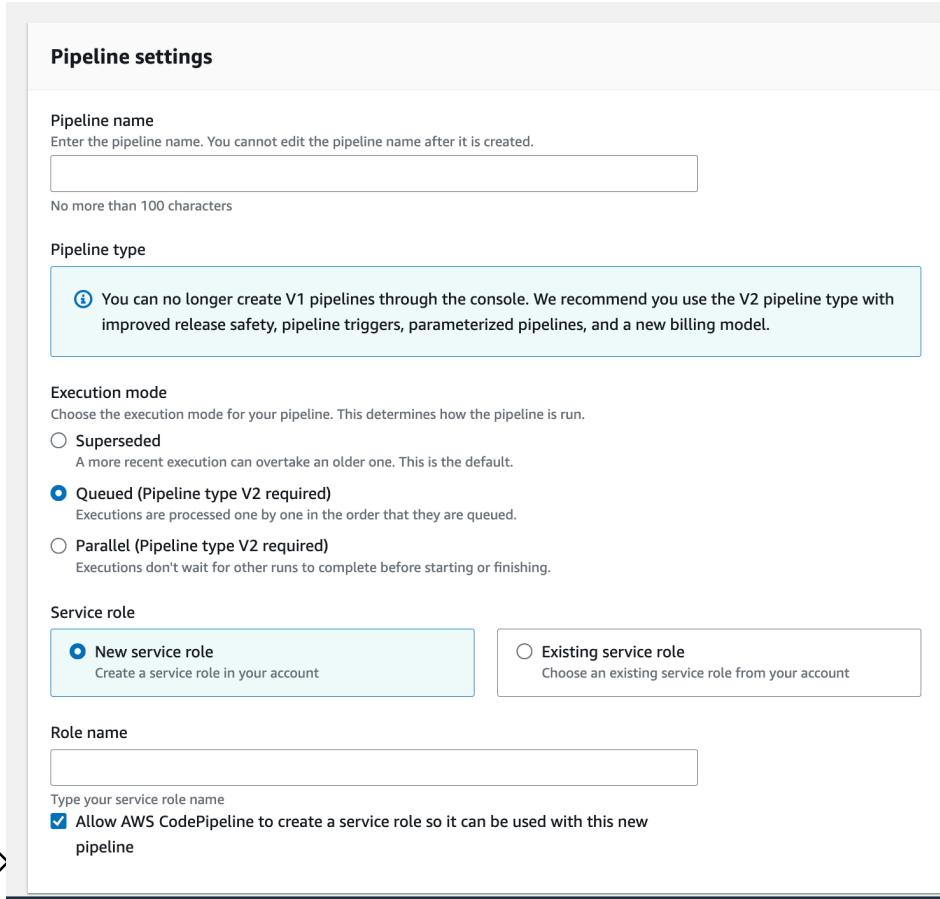
- AWS Codepipeline

The screenshot shows the AWS CodePipeline interface under the 'Developer Tools' section. The navigation path is 'Developer Tools > CodePipeline > Pipelines'. The main area displays a table with columns: Name, Latest execution status, Latest source revisions, Latest execution started, and Most recent executions. A search bar and pagination controls are also present. The message 'No results' and 'There are no results to display.' is centered below the table.

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
No results				
There are no results to display.				

# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline



## Execution mode

- Superseded: 이전 실행이 최근 실행으로 대체됨
- Queued: 한 번에 하나씩 실행되며, 대기열에 등록됨
- Parallel: 대기없이 동시에 모든 파이프라인을 실행함

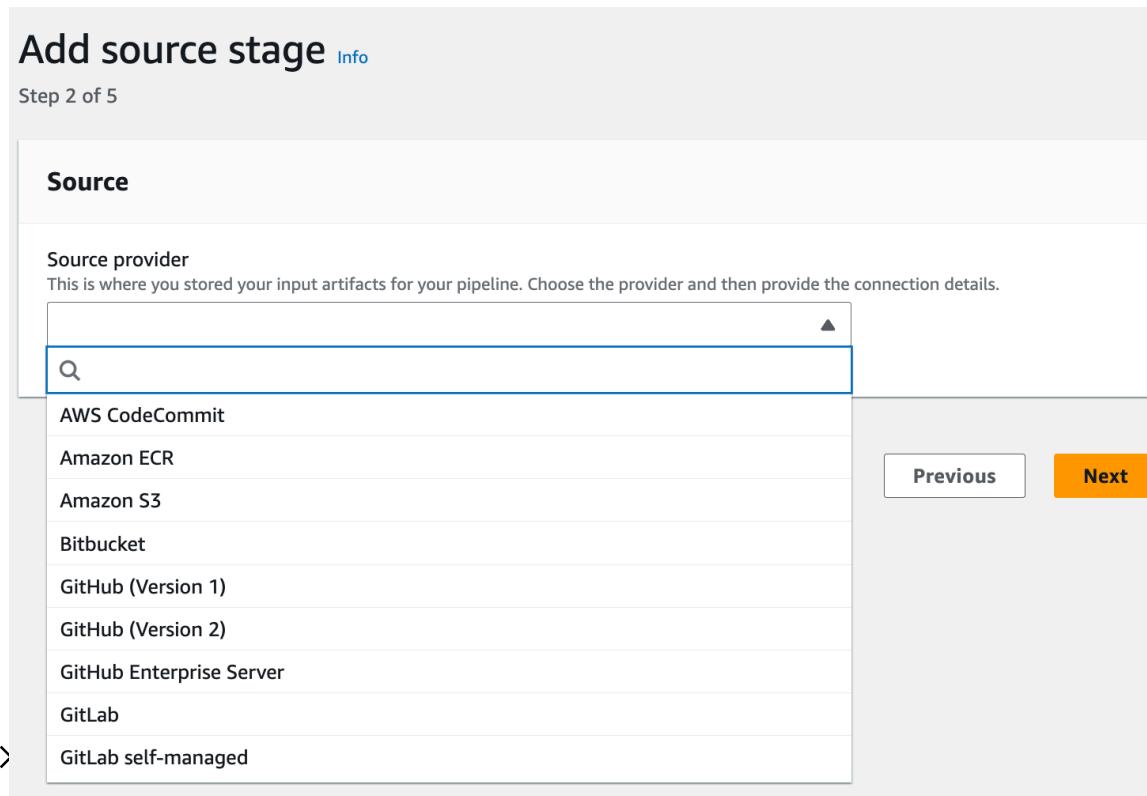
## Service Role

- 생성하는 Codepipeline이 갖게되는 권한에 대해 정의된 IAM Role

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline

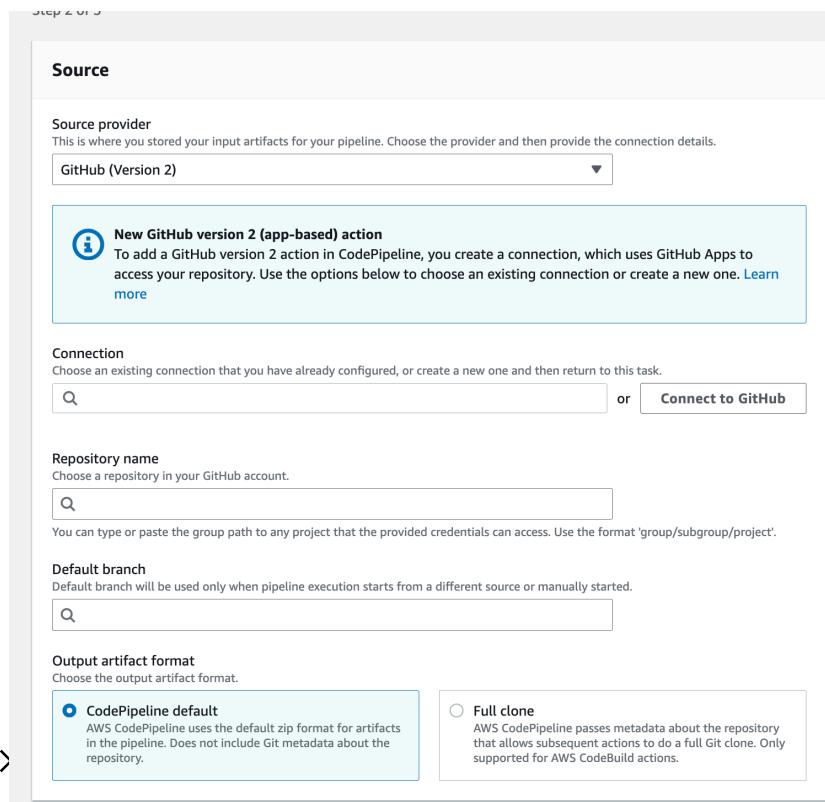
- 어떤 곳에서 코드를 가져올 것인지 선택할 수 있다. 이번 실습에서는 Github(Version 2)



# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline

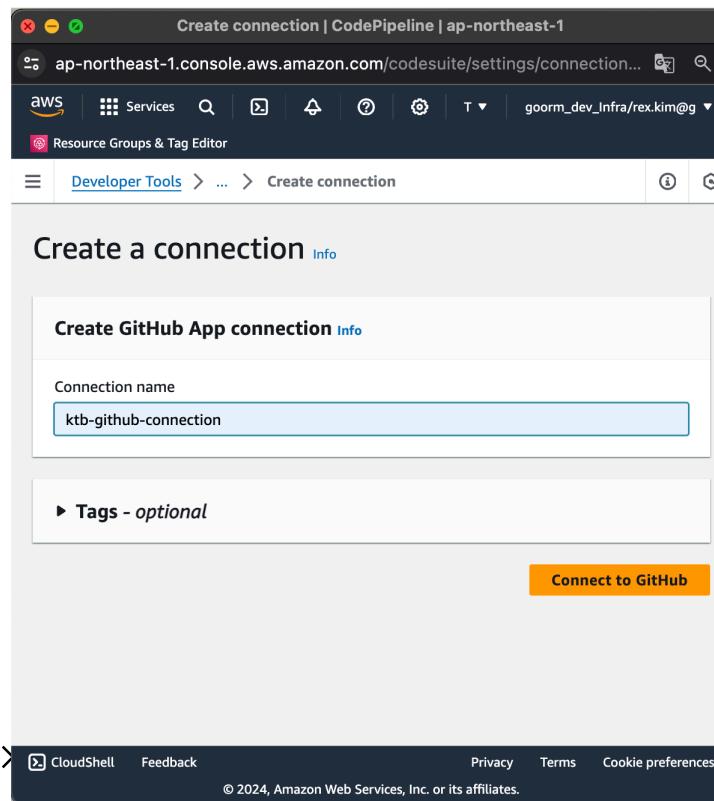
- Connect to Github 으로 생성한 Github과 연결



# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline

- Connect to Github 으로 생성한 Github과 연결



# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- **AWS Codepipeline**

- Github에 App 설치가 정상적으로 완료되면, 아래와 같이 Repository와 Branch를 설정할 수 있다.
  - Default branch는 실습에 사용할 Branch에 맞춰서 입력

**Connection**  
Choose an existing connection that you have already configured, or create a new one and then return to this task.

or [Connect to GitHub](#)

 **Ready to connect**  
Your GitHub connection is ready for use.

**Repository name**  
Choose a repository in your GitHub account.

[X](#)

You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.

**Default branch**  
Default branch will be used only when pipeline execution starts from a different source or manually started.

[X](#)

**Output artifact format**  
Choose the output artifact format.

**CodePipeline default**  
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.

**Full clone**  
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions.

# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline

- Trigger 에서는 Codepipeline이 시작되는 조건을 설정한다.

**Trigger**

Trigger type  
Choose the trigger type that starts your pipeline.

- No filter  
Starts your pipeline on any push and clones the HEAD.
- Specify filter  
Starts your pipeline on a specific filter and clones the exact commit. Pipeline type V2 is required.
- Do not detect changes  
Don't automatically trigger the pipeline.

Event type  
Choose the event type for the trigger that starts your pipeline.

- Push
- Pull request

Filter type  
Choose the filter type for the event that starts your pipeline.

- Branch
- Tags

Branches  
You can specify the target branch or branches you are pushing to. Use a comma to specify multiple entries.

Include

Exclude

File paths - optional  
You can specify the file path or file paths you are pushing to. Use a comma to separate multiple entries.

Include

Exclude

## Trigger Type

- No Filter: 별도의 필터 없이 HEAD의 변경점을 기준으로 시작
- Specify Filter: 다양한 조건으로 파이프라인이 시작될 수 있도록 지정 가능
- Do not detect changes: 자동으로 파이프라인 시작을 하지 않음

## Event Type

- Push 혹은 PR을 기준으로 파이프라인 시작 가능

## Filter Type

- 필터를 적용할 타입을 Branch와 Tag 중에서 고를 수 있음

## Branches

- 파이프라인이 시작되는 Branch의 조건을 지정할 수 있음

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline

- Build에 사용할 도구를 선택한다.
  - CodeBuild와 Jenkins를 제공하는데, CodeBuild 를 사용

### Add build stage Info

Step 3 of 5

**Build - optional**

**Build provider**  
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

▼

**Cancel** **Previous** **Skip build stage** **Next**

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline

- Create Project 로 신규 CodeBuild 생성

Add build stage Info

Step 3 of 5

**Build - optional**

**Build provider**  
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild

**Region**  
Asia Pacific (Tokyo)

**Project name**  
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

or [Create project](#)

**Environment variables - optional**  
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

[Add environment variable](#)

**Build type**

Single build  
Triggers a single build.

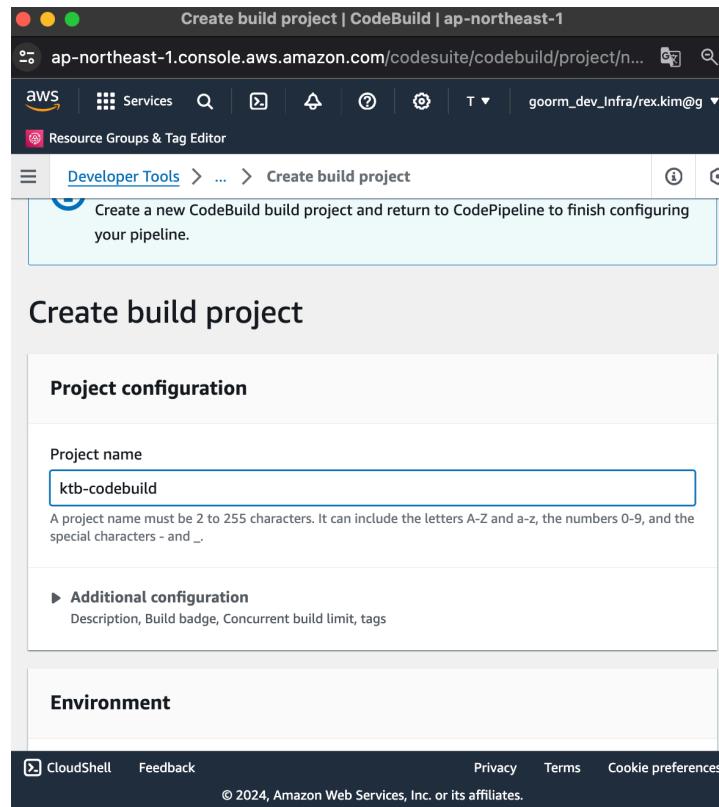
Batch build  
Triggers multiple builds as a single execution.

[Cancel](#) [Previous](#) [Skip build stage](#) [Next](#)

# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline

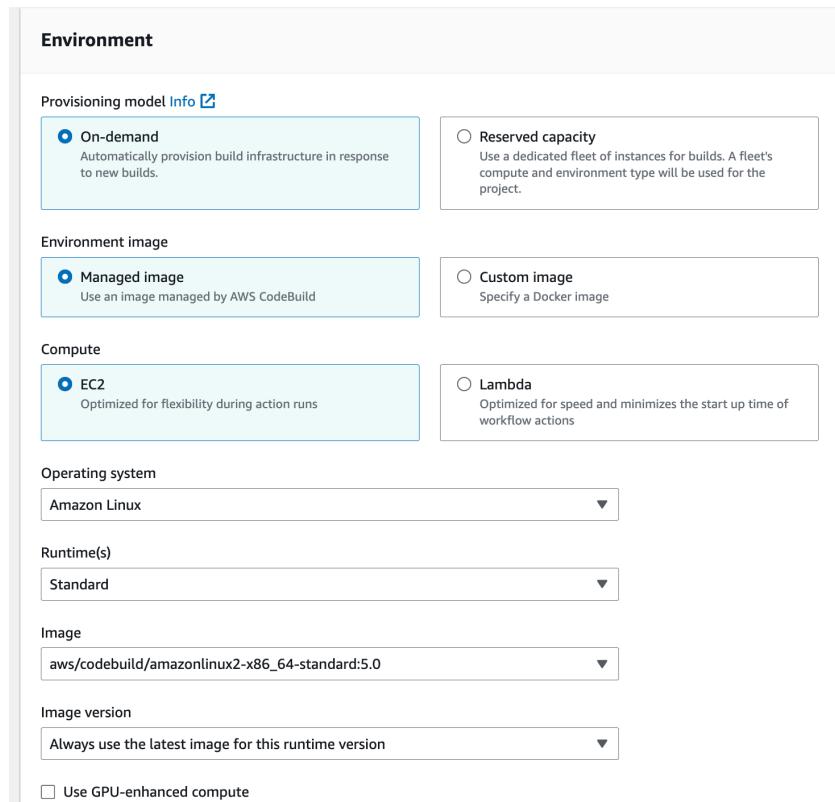
- Create Project 로 신규 CodeBuild 생성



# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS CodeBuild

- Create Project 로 신규 CodeBuild 생성



## Provisioning Model

- On Demand: 새로운 요청이 있을 때마다 필요한 리소스를 구성하여 빌드
- Reserved capacity: 빌드에 사용할 리소스를 사전에 정의하여 운영

## Environment Image

- 빌드 환경의 이미지를 지정할 수 있다.

## Compute

- 컴퓨팅 리소스를 어디에서 사용할 것인지 선택할 수 있다.

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS CodeBuild

- CodeBuild 에서 사용된 IAM Role 생성

### Service role

New service role

Create a service role in your account

Existing service role

Choose an existing service role from your account

### Role name

ktb-rex-build-role

Type your service role name

### ▶ Additional configuration

Timeout, privileged, certificate, VPC, compute type, environment variables, file systems

# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS CodeBuild

- 파이프라인 정상 동작을 위해 생성 단계에서는 아래와 같이 설정한다.

## Buildspec

### Build specifications

Insert build commands

Store build commands as build project configuration

Use a buildspec file

Store build commands in a YAML-formatted buildspec file

### Build commands

Enter commands you want to run during the build phase. Separate each build command with "&&." For example, "mvn test && mvn package." Use a buildspec file to run commands in other phases or if you have a long list of commands.

npm install && npm test

Switch to editor

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS CodeBuild

- 처음 생성 시도 시, 아래와 같은 에러가 발생 후 다시 생성 버튼을 누르면 두 번째 이미지와 같이 이미 생성되었다는 에러 발생
  - AWS 에러로 실습 중에 잘못하신게 아닙니다!!
- 생성된 CodeBuild가 노출되지 않으므로 새로고침을 한번 진행해주셔야합니다.

✖ There is no reference to the window opener.

✖ Role with name ktb-rex-build-role already exists.

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline

- Deploy 는 초기 구성에는 설정하지 않고 Skip

Add deploy stage [Info](#)

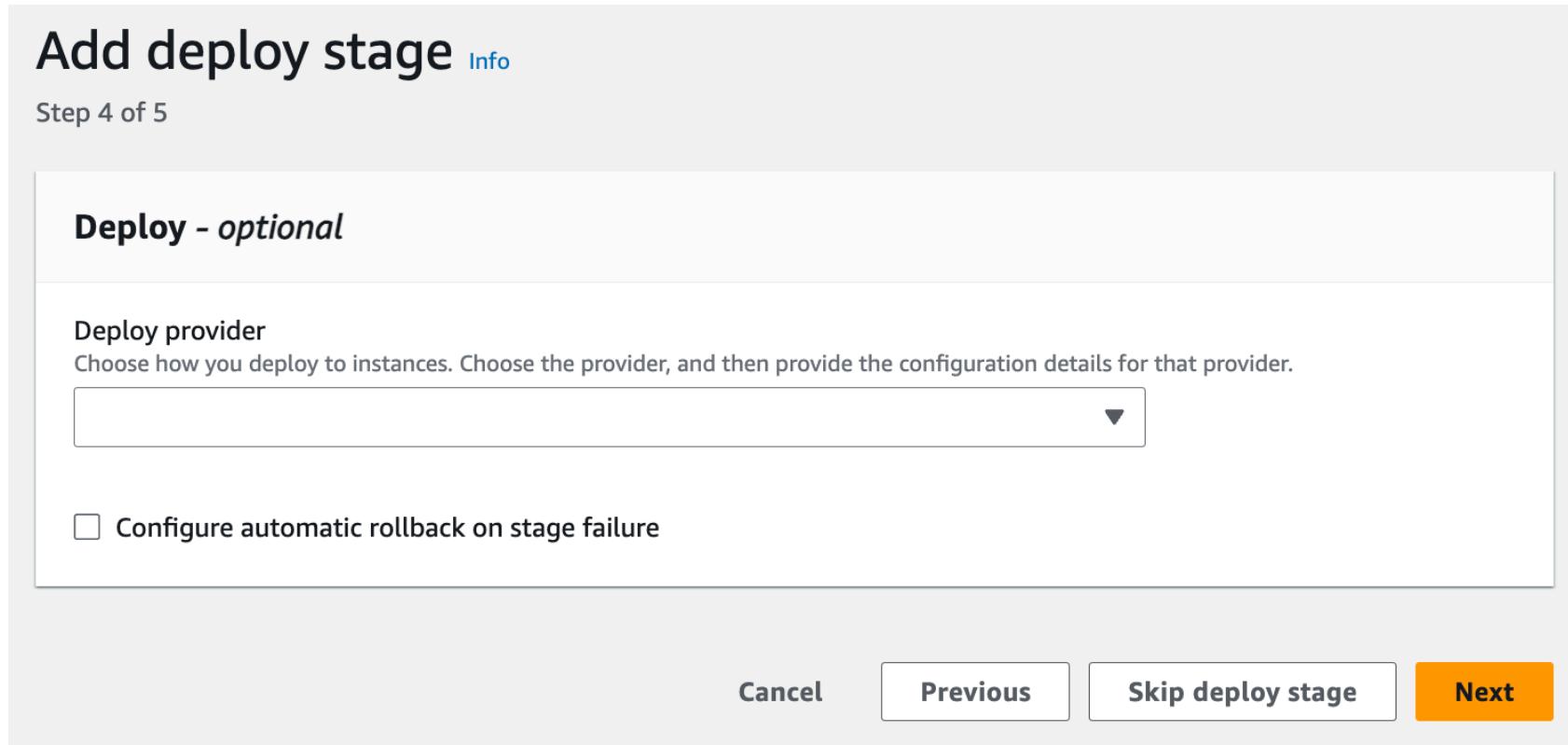
Step 4 of 5

**Deploy - optional**

**Deploy provider**  
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

Configure automatic rollback on stage failure

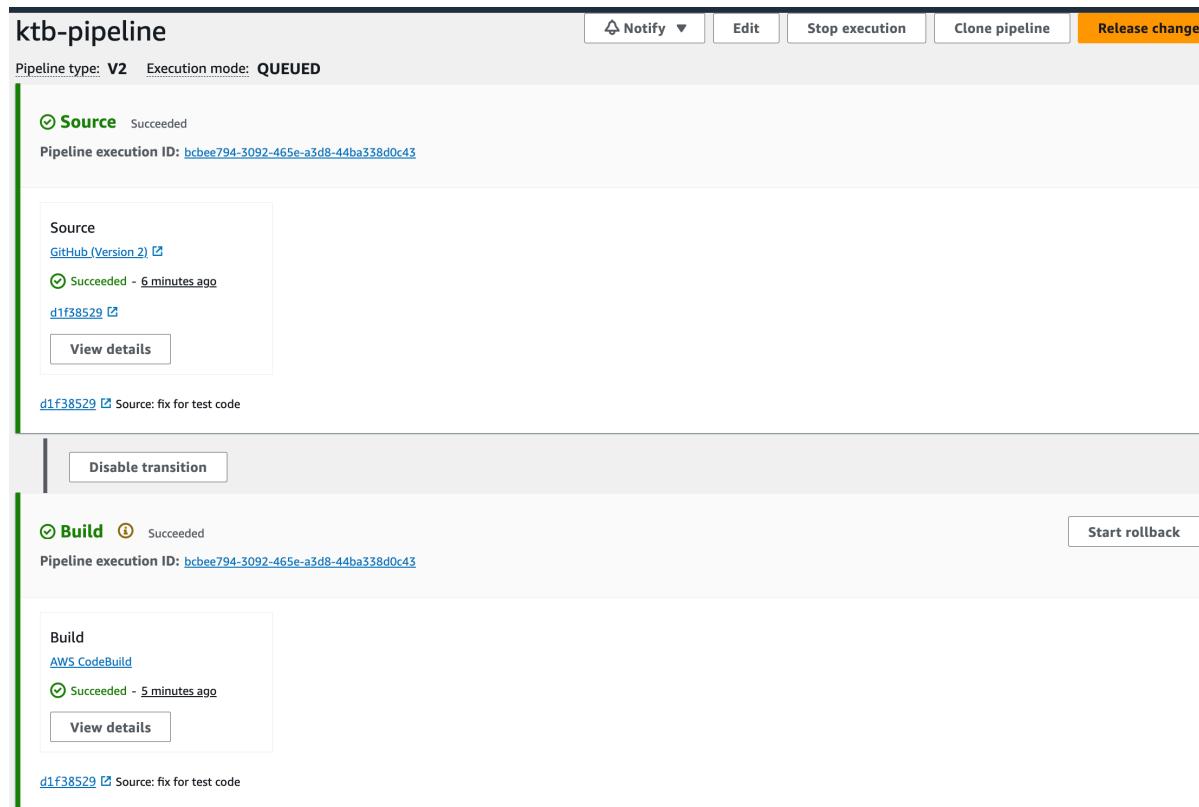
[Cancel](#) [Previous](#) [Skip deploy stage](#) **Next**



# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- **AWS Codepipeline**

- 생성이 잘되었다면 아래 이미지와 같이 파이프라인이 최초 동작하여 빌드까지 성공된 것을 확인할 수 있다.



## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Registry(ECR)

Private repositories

Repositories	C	View push commands	Delete	Actions ▾	Create repository		
<input type="text"/> Filter status				< 1 >			
Repository name	▲	URI	Created at	▼	Tag immutability	Scan frequency	Encryption type
No repositories No repositories were found							

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Registry(ECR)

**Create repository**

**General settings**

**Visibility settings** [Info](#)  
Choose the visibility setting for the repository.

**Private**  
Access is managed by IAM and repository policy permissions.

**Public**  
Publicly visible and accessible for image pulls.

**Repository name**  
Provide a concise name. A developer should be able to identify the repository contents by the name.

587169513591.dkr.ecr.ap-northeast-1.amazonaws.com/ **ktb-repo**

8 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

**Tag immutability** [Info](#)  
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

**Disabled**

**Once a repository is created, the visibility setting of the repository can't be changed.**

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Registry(ECR)

Private repositories							
Repositories (1)			C	View push commands	Delete	Actions ▾	Create repository
Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type		
ktb-repo	587169513591.dkr.ecr.ap-northeast-1.amazonaws.com/ktb-repo	2024년 8월 04일, 17:02:08 (UTC+09)	Disabled	Manual	AES-256		

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS CodeBuild
  - 테스트용 빌드 명령어에서 컨테이너 이미지 빌드를 위한 명령어로 변경

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS CodeBuild

- CodeBuild Project의 IAM을 수정한다.
- Service Role의 링크를 통해 이동

The screenshot shows the AWS CodeBuild console with the following details:

Navigation: Developer Tools > CodeBuild > Build projects > ktb-codebuild

Project Name: ktb-codebuild

Action Buttons: Actions ▾, Create trigger, Edit, Clone, Debug build, Start build with overrides, Start build (highlighted in orange)

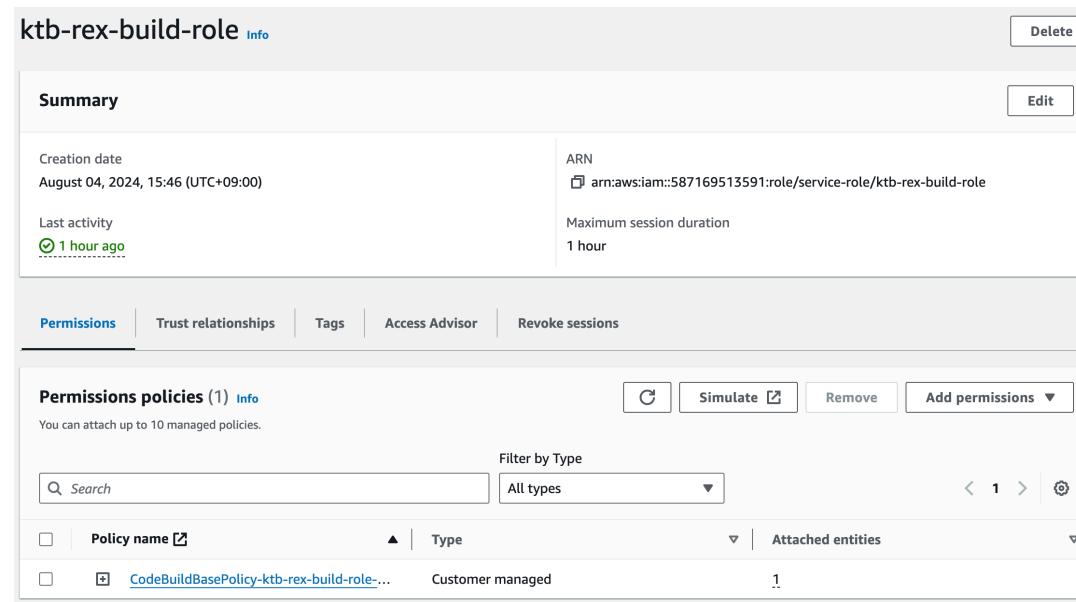
Configuration Details:

Configuration			
Source provider AWS CodePipeline	Primary repository -	Artifacts upload location -	Service role arn:aws:iam::587169513591:role/service-role/ktb-rex-build-role
Public builds Disabled			

# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS CodeBuild

- 기본 IAM Role은 아래와 같이 설정되어 있다.



## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS CodeBuild

- 빌드 과정에서 ECR 권한이 필요하기 때문에 `AmazonEC2ContainerRegistryPowerUser` 를 추가한다.

Other permissions policies (1/1032)

Filter by Type: All types | 7 matches

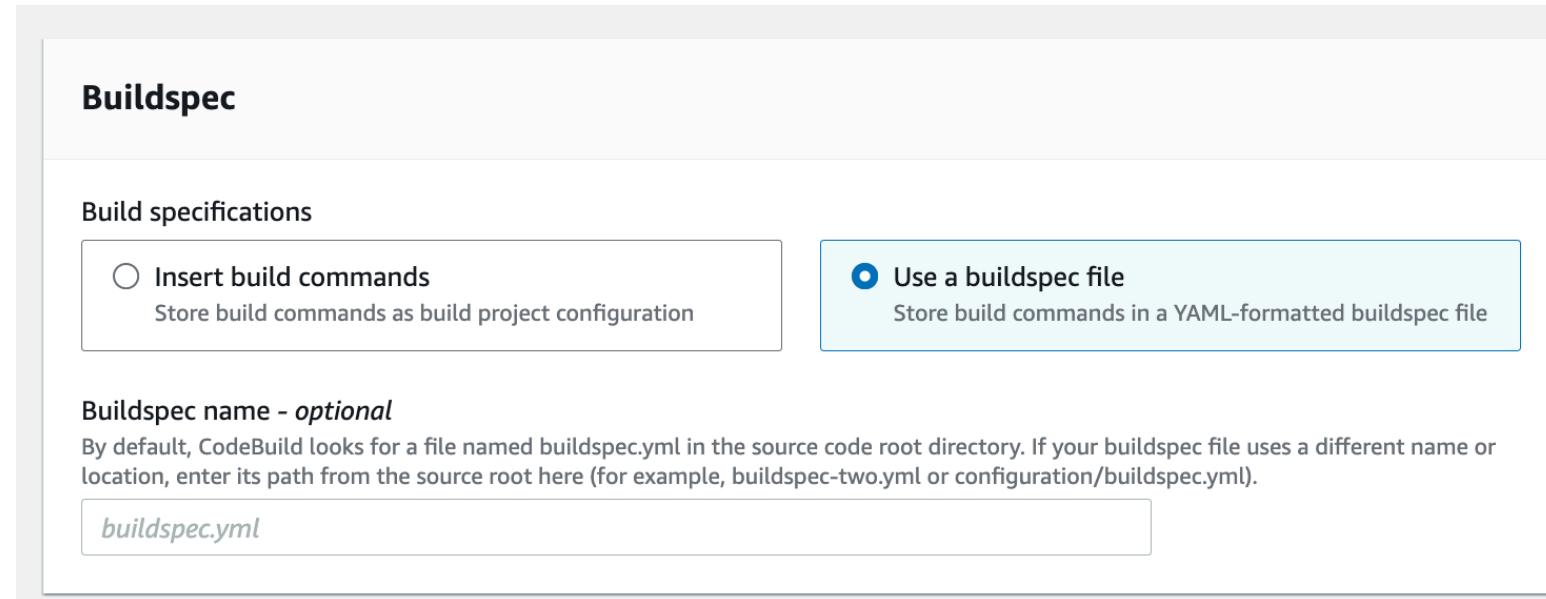
Policy name	Type	Description
<input type="checkbox"/> <a href="#">AmazonEC2ContainerRegistryFullAccess</a>	AWS managed	Provides administrative access to Ama...
<input checked="" type="checkbox"/> <a href="#">AmazonEC2ContainerRegistryPowerUser</a>	AWS managed	Provides full access to Amazon EC2 Co...
<input type="checkbox"/> <a href="#">AmazonEC2ContainerRegistryReadOnly</a>	AWS managed	Provides read-only access to Amazon E...
<input type="checkbox"/> <a href="#">AmazonEC2ContainerServiceAutoscale</a>	AWS managed	Policy to enable Task Autoscaling for A...
<input type="checkbox"/> <a href="#">AmazonEC2ContainerServiceEventsRole</a>	AWS managed	Policy to enable CloudWatch Events fo...
<input type="checkbox"/> <a href="#">AmazonEC2ContainerServiceforEC2Role</a>	AWS managed	Default policy for the Amazon EC2 Rol...
<input type="checkbox"/> <a href="#">AmazonEC2ContainerServiceRole</a>	AWS managed	Default policy for Amazon ECS service ...

Cancel Add permissions

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS CodeBuild

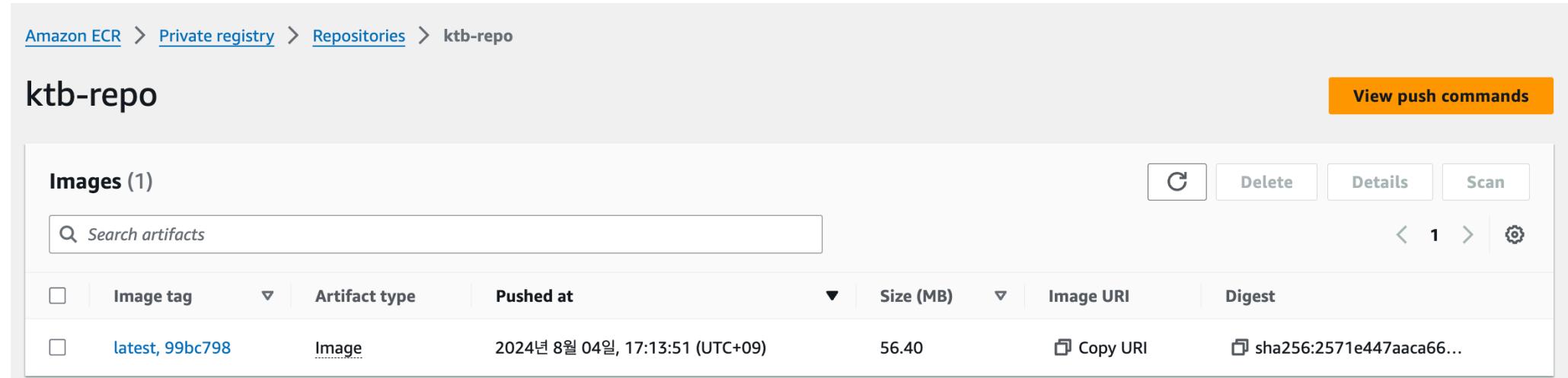
- 빌드 과정에서 ECR 권한이 필요하기 때문에 `AmazonEC2ContainerRegistryPowerUser` 를 추가한다.



## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS CodeBuild

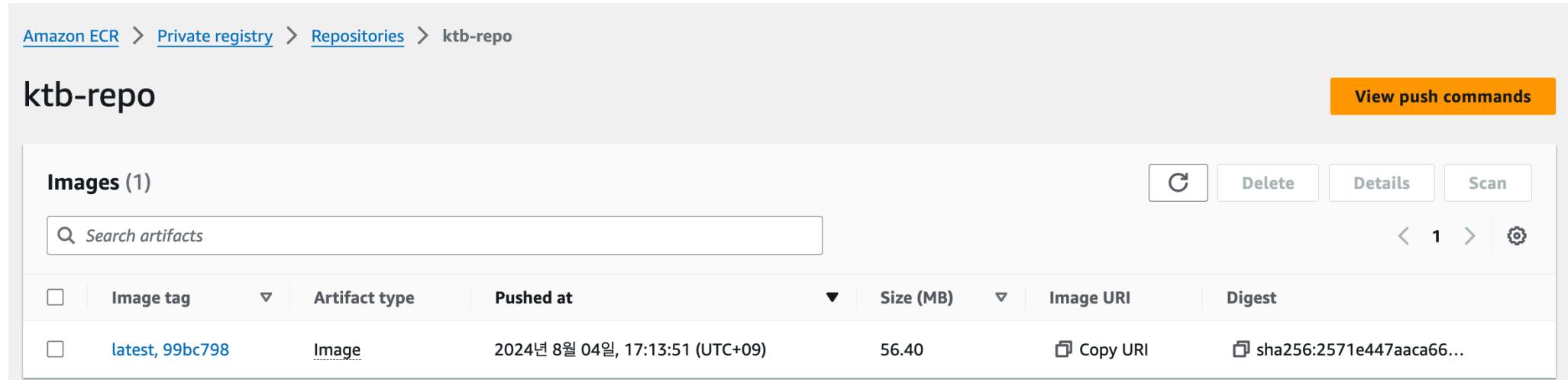
- CodeBuild 설정과 `buildspec.yml` 을 수정하여 Git을 최신화하면 아래와 같이 ECR에 이미지가 올라온 것을 확인할 수 있다.



## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

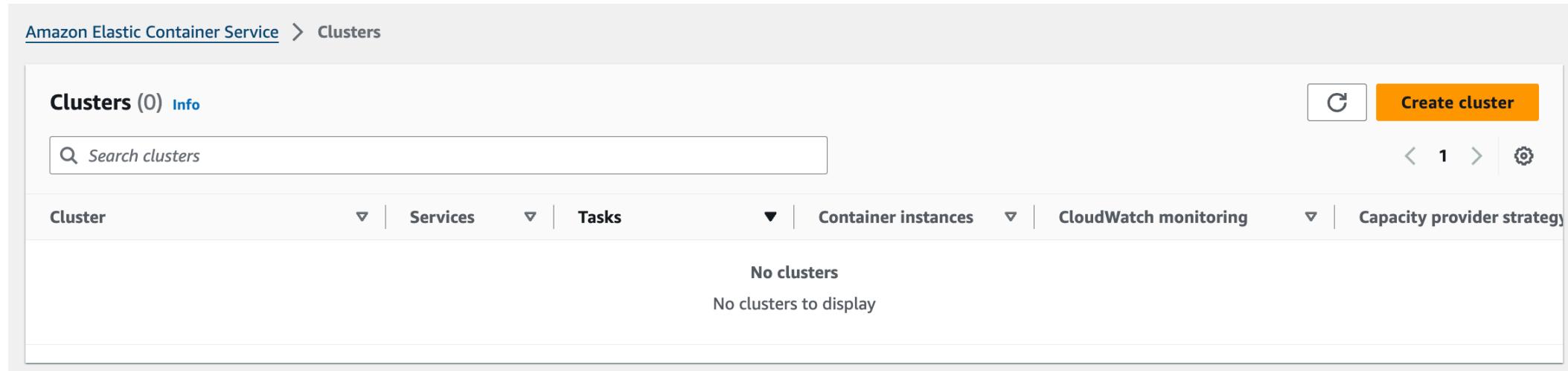
- AWS CodeBuild

- CodeBuild 설정과 `buildspec.yml` 을 수정하여 Git을 최신화하면 아래와 같이 ECR에 이미지가 올라온 것을 확인할 수 있다.



## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)



# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)

An Amazon ECS cluster groups together tasks, and services, and allows for shared capacity and common configurations. All of your tasks, services, and capacity must belong to a cluster.

**Cluster configuration**

**Cluster name**  
ktb-cluster  
Cluster name must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (\_).

**Default namespace - optional**  
Select the namespace to specify a group of services that make up your application. You can overwrite this value at the service level.  
ktb-cluster

**▼ Infrastructure Info** Serverless

Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances.

**AWS Fargate (serverless)**  
Pay as you go. Use if you have tiny, batch, or burst workloads or for zero maintenance overhead. The cluster has Fargate and Fargate Spot capacity providers by default.

**Amazon EC2 instances**  
Manual configurations. Use for large workloads with consistent resource demands.

**External instances using ECS Anywhere** can be registered after cluster creation is complete.

## Infrastructure

- 컨테이너를 구성할 인프라 풀을 지정한다.
- AWS Fargate
- Amazon EC2 instances

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)

The screenshot shows two related pages from the AWS ECS console.

**Cluster overview:**

- ARN:** arn:aws:ecs:ap-northeast-1:58716951359:cluster/ktb-cluster
- Status:** Active
- CloudWatch monitoring:** Default
- Registered container instances:** -
- Services:**
  - Draining:** Active
  - Pending:** Pending
  - Running:** -
- Encryption:**
  - Managed storage:** Fargate ephemeral storage

**Services:** Services (0) Info

Filter launch type: Any launch type | Filter service type: Any service type

Service name | ARN | Status | Service t... | Deployments and tasks | Last deploy... | Task de...

No services

No services to display.

Create

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)

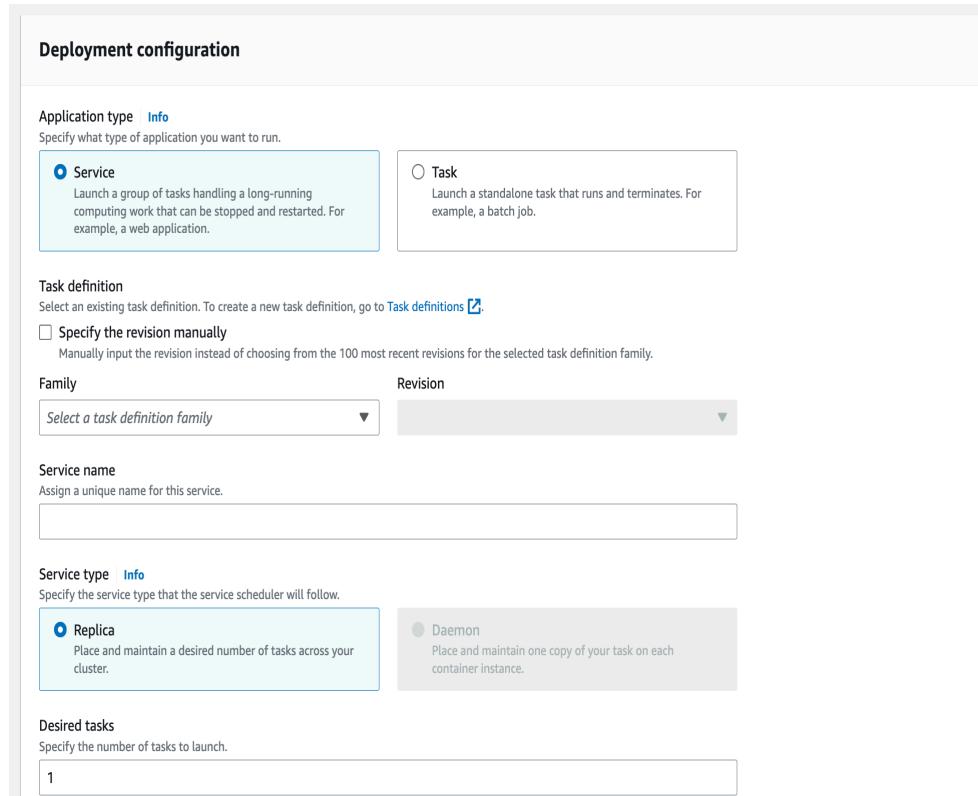
The screenshot shows the 'Create' screen for setting up a new AWS Elastic Container Service (ECS) cluster. The 'Environment' tab is selected, and the 'AWS Fargate' option is chosen. Under 'Compute configuration (advanced)', the 'Launch type' section is highlighted, indicating the choice of launching tasks directly without a capacity provider strategy. Other sections shown include 'Existing cluster' (ktb-cluster), 'Compute options' (Capacity provider strategy or Launch type), 'Launch type' (FARGATE), and 'Platform version' (LATEST).

### Compute options

- 인프라 풀을 구성할 전략을 지정한다.
- Capacity Provider Strategy
- Launch Type

# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)



## Application Type

- 컨테이너를 띄울 애플리케이션 종류를 지정
- Service: 웹 애플리케이션과 같이 오랜 시간 컴퓨팅을 쓰는 종류의 애플리케이션
- Task: 일회성으로 실행되는 애플리케이션

## Task Definition

- 서비스 내에서 실행될 컨테이너 스펙을 정의함
- 첫 생성 시에는 정의된 Task Definition이 없으므로, go to Task Definition을 통해 생성

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)

The screenshot shows the AWS Elastic Container Service (ECS) Task definitions page. At the top, there is a breadcrumb navigation: [Amazon Elastic Container Service](#) > [Task definitions](#). Below the breadcrumb, there is a header with the text "Task definitions (0)" and an "Info" link. To the right of the header are several buttons: a refresh icon, "Deploy ▾", "Create new revision ▾", and "Create new task definition ▾". The "Create new task definition" button is highlighted with a yellow background. Below the header is a search bar with the placeholder "Filter task definitions" and a dropdown menu labeled "Active" under "Filter by status". On the far right, there are navigation icons for pages 1 and 2, and a gear icon for settings. The main content area has two columns: "Task definition" and "Status of last revision". A message "No task definitions" is displayed, followed by "No task definitions to display." At the bottom of this section is a "Create new task definition" button.

# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)

▼ Infrastructure requirements  
Specify the infrastructure requirements for the task definition.

Launch type [Info](#)  
Selection of the launch type will change task definition parameters.

**AWS Fargate**  
Serverless compute for containers.

**Amazon EC2 instances**  
Self-managed infrastructure using Amazon EC2 instances.

OS, Architecture, Network mode  
Network mode is used for tasks and is dependent on the compute type selected.

Operating system/Architecture [Info](#)

Network mode [Info](#)

Task size [Info](#)  
Specify the amount of CPU and memory to reserve for your task.

CPU   
Memory

▼ Task roles - conditional

Task role [Info](#)  
A task IAM role allows containers in the task to make API requests to AWS services. You can create a task IAM role from the [IAM console](#).

Task execution role [Info](#)  
A task execution IAM role is used by the container agent to make AWS API requests on your behalf. If you don't already have a task execution IAM role created, we can create one for you.

# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)

▼ Container - 1 [Info](#)

[Essential container](#) [Remove](#)

**Container details**  
Specify a name, container image, and whether the container should be marked as essential. Each task definition must have at least one essential container.

Name	Image URI	Essential container
ktb-repo	587169513591.dkr.ecr.ap-northeast-1.amazonaws.com/ktb-repo	Yes

Up to 255 letters (uppercase and lowercase), numbers, hyphens, underscores, colons, periods, forward slashes, and number signs are allowed.

**Private registry** [Info](#)  
Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.

Private registry authentication

**Port mappings** [Info](#)  
Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol
3000	TCP	web-server	HTTP

[Add port mapping](#)

**Read only root file system** [Info](#)  
When this parameter is turned on, the container is given read-only access to its root file system.

Read only

**Resource allocation limits - conditional** [Info](#)  
Container-level CPU, GPU, and memory limits are different from task-level values. They define how much resources are allocated for the container. If container attempts to exceed the memory specified in hard limit, the container is terminated.

CPU	GPU	Memory hard limit	Memory soft limit
1 in vCPU	1 in GB	3 in GB	1 in GB

## Container Detail

- Name: buildspec.yml에서 post\_build에서 입력한 name과 같게 설정(레포 이름으로 설정되어 있음)
- Image URI: ECR URI를 입력

## Port Mapping

- 현재 애플리케이션이 3000으로 실행되므로 3000으로 설정

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)

▼ **Load balancing - optional**  
Configure load balancing using Amazon Elastic Load Balancing to distribute traffic evenly across the healthy tasks in your service.

Load balancer type [Info](#)  
Configure a load balancer to distribute incoming traffic across the tasks running in your service.

Application Load Balancer

Container  
The container and port to load balance the incoming traffic to  
ktb-repo 3000:3000

Host port:Container port

Application Load Balancer  
Specify whether to create a new load balancer or choose an existing one.

Create a new load balancer  
 Use an existing load balancer

Load balancer name  
Assign a unique name for the load balancer.  
ktb-repo-alb

Health check grace period [Info](#)  
0

### Load balancing

- 서비스를 외부에 노출시키기 위해 로드밸런서 생성

# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)

**Listener [Info](#)**  
Specify the port and protocol that the load balancer will listen for connection requests on.

Create new listener  Use an existing listener  
You need to select an existing load balancer.

Port

Protocol

**Target group [Info](#)**  
Specify whether to create a new target group or choose an existing one that the load balancer will use to route requests to the tasks in your service.

Create new target group  Use an existing target group

Target group name

Protocol

Deregistration delay  
The amount of time to wait before the state of a deregistering target changes from draining to unused.

seconds

Health check protocol

Health check path [Info](#)

# 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)

The screenshot shows two related AWS ECS management pages:

**Cluster overview** (Top Section):

ARN	Status	CloudWatch monitoring	Registered container instances
arn:aws:ecs:ap-northeast-1:587169513591:cluster/ktb-cluster	Active	Default	-

**Services** (Bottom Section):

Draining	Active	Pending	Running
-	1	-	1

**Encryption** (Bottom Section):

Managed storage	Fargate ephemeral storage
-	-

**Services (1) Info** (Second Page):

Service name	ARN	Status	Service type	Deployments and tasks	Last deploy...	Task de...
ktb-repo-ecs-service	arn:aws:ec...	Active	REPLICA	1/1 Tasks running	Completed	ktb-def...

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)

- 기본 Security Group으로는 외부에서의 접근이 허용되지 않기 때문에, 추가가 필요하다.

Network configuration			
Network <a href="#">vpc-cbac8eac</a>	Security groups <a href="#">sg-5f189c23</a>	Service role <a href="#">AWSServiceRoleForECS</a>	Load balancers <a href="#">ktb-repo-alb</a>
Subnets <a href="#">subnet-76f9163e</a> <a href="#">subnet-a9540df2</a> <a href="#">subnet-beee1895</a>	Auto-assign public IP <span>Turned on</span>	Health check grace period -	DNS names <a href="#">ktb-repo-alb-472330986.ap-northeast-1.elb.amazonaws.com</a>   <a href="#">open address</a>

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)
  - 현재 만든 웹서비스의 경우 별도 인증서는 없으므로 HTTP(80)만 오픈

Create security group [Info](#)

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

**Basic details**

Security group name [Info](#)  
 Name cannot be edited after creation.

Description [Info](#)

VPC [Info](#)

**Inbound rules [Info](#)**

Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>
HTTP	TCP	80	Anywhere... <input type="button" value="Delete"/>	<input type="text" value="0.0.0.0"/> <input type="button" value="X"/>

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)

Listeners and rules		Network mapping	Resource map - new	Security	Monitoring	Integrations	Attributes	Tags
<strong>Security groups (1)</strong>								
A security group is a set of firewall rules that control the traffic to your load balancer.								
Security Group ID	▼	Name	▼	Description				
<a href="#">sg-5f189c23</a>		default		default VPC security group				

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Elastic Container Service(ECS)

- 위에서 생성한 Security Group을 추가
- 이후 LB의 DNS에 접근하면 잘 접근되는 것을 확인할 수 있다.

The screenshot shows the AWS Lambda console with the following navigation path: EC2 > Load balancers > ktb-repo-alb > Edit security groups. The main title is "Edit security groups". A sub-section header "Load balancer details: ktb-repo-alb" is visible. Below it, the "Security groups" section is shown. It contains a sub-header "Security groups" and a note: "A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#)." A search bar and a "Select up to 5 security groups" dropdown are present. Two security groups are listed: "LB-sg" (selected) and "default". Both entries include their SG ID, creation date, and VPC information. At the bottom right are "Cancel" and "Save changes" buttons. The "Save changes" button is highlighted with a yellow background.

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline

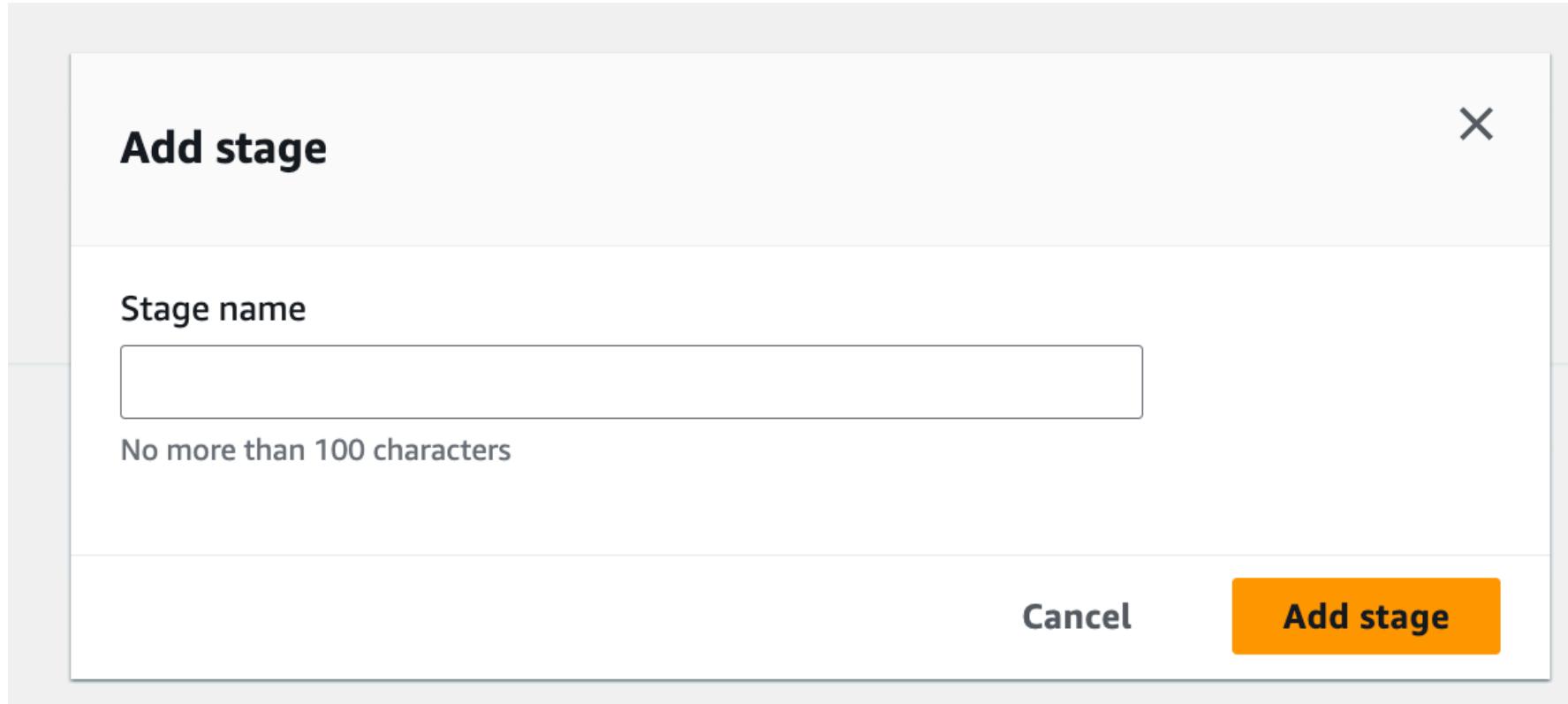
- Codepipeline 생성 단계에서 Skip했던 Deploy Stage를 추가한다.

The screenshot shows the AWS CodePipeline console with the following details:

- Path:** Developer Tools > CodePipeline > Pipelines > ktb-pipeline
- Pipeline Name:** ktb-pipeline
- Pipeline type:** V2
- Execution mode:** QUEUED
- Source Stage:** Succeeded (GitHub (Version 2)) - 1 hour ago (commit 99bc798a)
- Notify:** (dropdown menu)
- Edit:** (button)
- Stop execution:** (button)
- Clone pipeline:** (button)
- Release change:** (orange button)
- Pipeline execution ID:** 3afde500-84a2-4183-ba86-21928c70e7e6
- Source Details:** View details
- Success Status:** 99bc798a [ ] Source: fix yml
- Action Types:** (checkboxes) [ ] [ ]
- Buttons at bottom:** Disable transition

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline
  - Codepipeline 생성 단계에서 Skip했던 Deploy Stage를 추가한다.



## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- **AWS Codepipeline**

- Codepipeline 생성 단계에서 Skip했던 Deploy Stage를 추가한다.

**Edit action**

Action name  
Choose a name for your action  
  
No more than 100 characters

Action provider

Region

Input artifacts  
Choose an input artifact for this action. [Learn more](#)  
  
No more than 100 characters

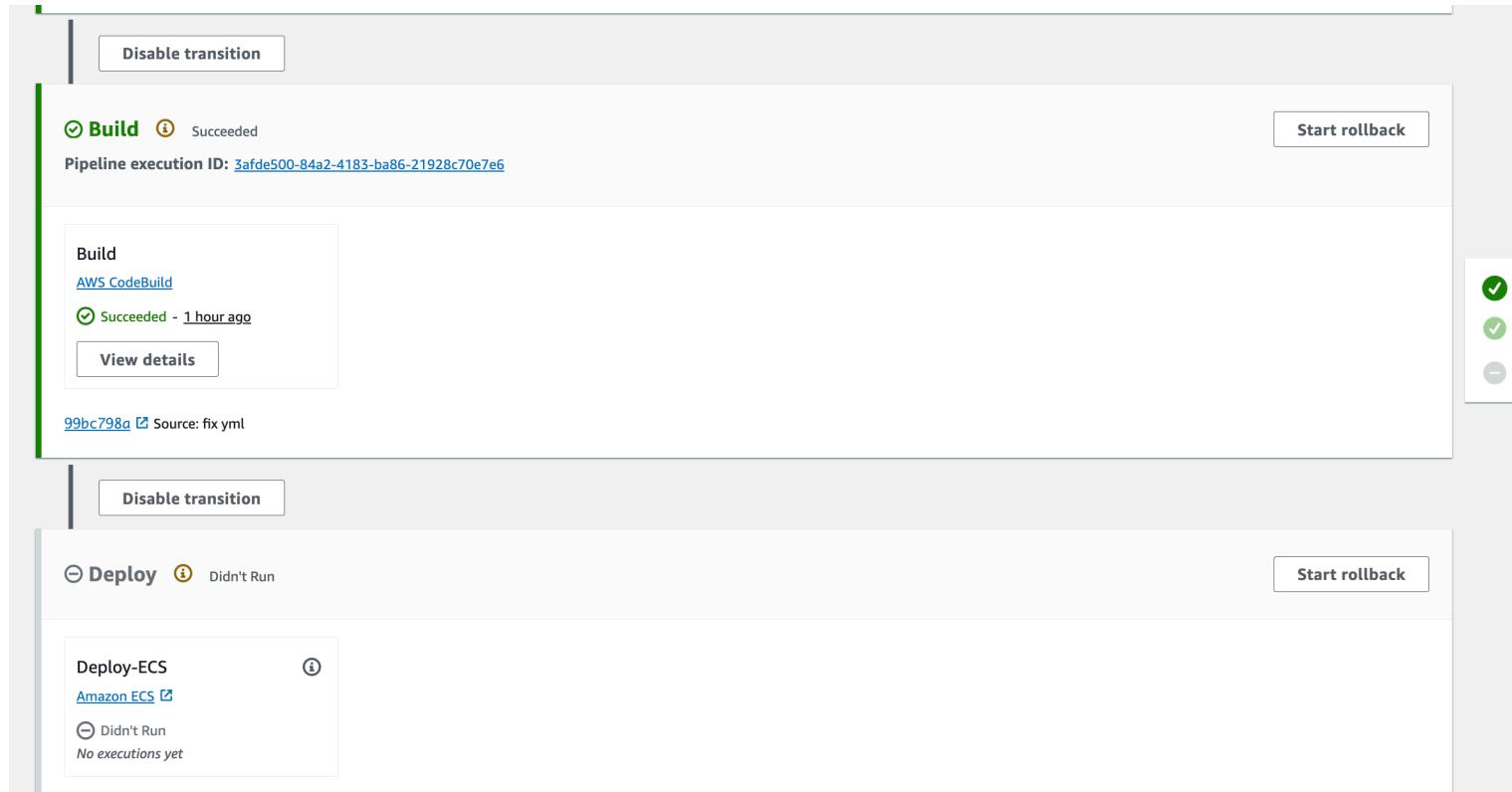
Cluster name  
Choose a cluster that you have already created in the Amazon ECS console. Or create a cluster in the Amazon ECS console and then return to this task.  
 X C

Service name  
Choose a service that you have already created in the Amazon ECS console for your cluster. Or create a new service in the Amazon ECS console and then return to this task.  
 X C

## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

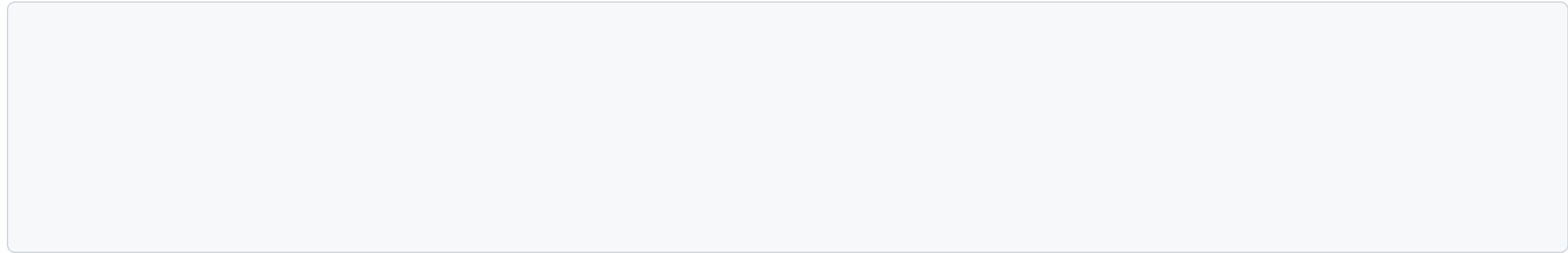
- AWS Codepipeline

- Codepipeline 생성 단계에서 Skip했던 Deploy Stage를 추가한다.



## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline
  - 파이프라인 Trigger를 위해 코드 수정 후 Push



## 클라우드 환경에서 CI/CD 파이프라인 구성해보기

- AWS Codepipeline
  - Deploy까지 잘 수행되면 CI/CD 파이프라인 구성 완료

