



## AWS Builders Korea

Amazon ECS Hands-on Lab

---

June 22, 2022

# 1. Amazon ECS 실습 개요

이번 실습은 AWS workshop studio (Cats and Dogs)를 일부 수정해 만든 문서입니다. 웹으로도 접근 가능하니 참고하시기 바랍니다.

<https://catalog.us-east-1.prod.workshops.aws/workshops/8c9036a7-7564-434c-b558-3588754e21f5/ko-KR/>

Amazon Elastic Container Service(Amazon ECS)는 완전관리형 컨테이너 오케스트레이션 서비스입니다. 기본적으로 Amazon Route 53, AWS Identity and Access Management(IAM), Amazon CloudWatch 등의 다른 서비스와 통합을 통해 컨테이너 배포 및 확장을 위한 익숙한 환경을 제공할 수 있습니다. 또한 다른 AWS 서비스와의 신속한 통합을 통해 ECS에 새로운 기능을 추가할 수도 있습니다.

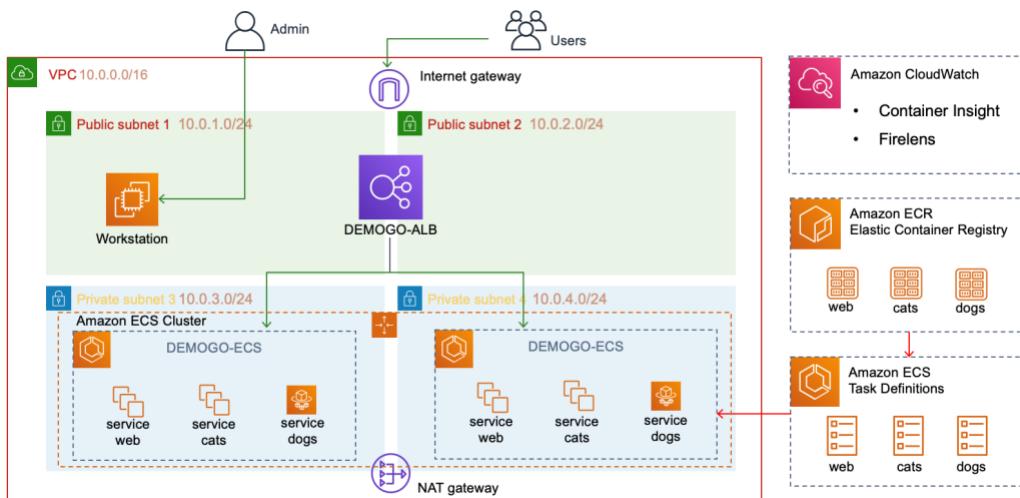
## (1-1) 실습 목적

이 워크샵을 통해 Amazon ECS의 기본 개념과 주요 구성 요소들을 이해하고 Amazon ECS 관리에 활용할 수 있는 유용한 도구들을 직접 테스트해봅니다.

## (1-2) 다루는 서비스

- Amazon Elastic Container Service (ECS)
- Amazon Elastic Container Registry (ECR)
- AWS Fargate
- Amazon CloudWatch
- AWS Identity and Access Management (IAM)
- Amazon CloudFormation
- Amazon Application Load Balancer (ALB)

## (1-3) 실습 아키텍처

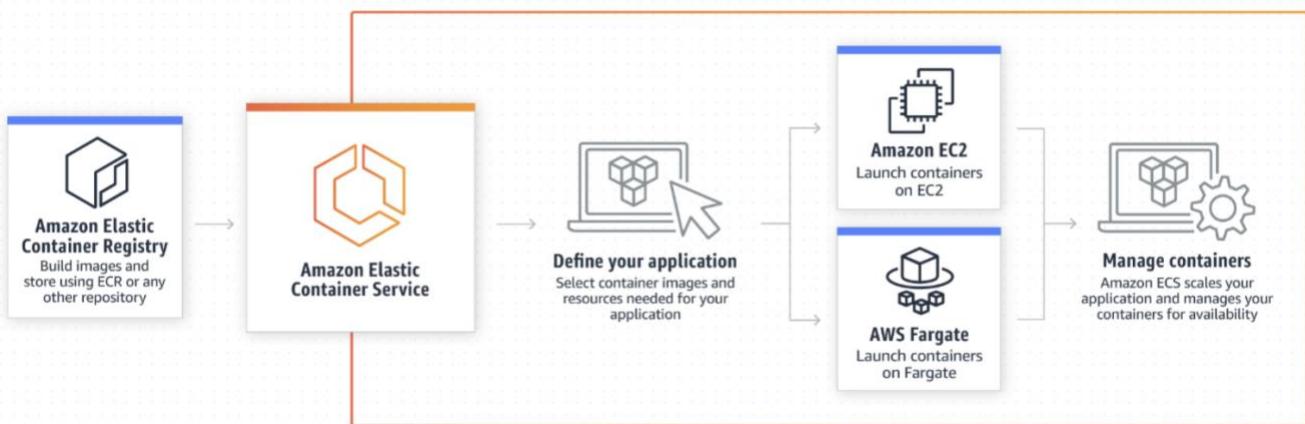


## (1-4) 실습 범위

- Amazon ECR에서 도커 이미지 관리
- Amazon ECS 클러스터, 태스크 정의, 서비스 생성
- 각 서비스별 EC2와 Fargate 중 알맞은 시작 유형 선택
- 컨테이너 웹 애플리케이션 cats and dogs을 Application Load Balancer 경로 기반으로 구성
- Container Insights로 Amazon ECS 모니터링
- Amazon ECS 서비스 오토스케일링
- Amazon ECS 클러스터 오토스케일링

## (1-5) Amazon EC2 와 AWS Fargate

Amazon ECS에서 애플리케이션의 아키텍처 설계에 있어 어떤 시작 유형(Launch Type)을 사용할지 결정하는 것은 중요합니다. 예를 들어, 컴플라이언스 및 정부 요구사항 등으로 EC2 인스턴스에 대한 제어를 강화해야 하는 경우나, 좀 더 폭넓은 사용자 지정 옵션이 필요한 경우라면 EC2를 사용해야만 합니다. 만약 이런 경우가 아니라면, [AWS Fargate](#)를 사용할 수 있습니다. AWS Fargate는 EC2 인스턴스를 프로비저닝하거나 관리할 필요 없이 AWS에서 컨테이너를 시작할 수 있는 가장 쉬운 방법입니다. 시작 유형에 대해 더 상세한 내용은 다음 [지침](#)을 참고합니다.



실습은 web, cats, dogs라는 세 가지 서비스로 구성되어 있습니다. 본 실습에서 DEMOGO-ECS 클러스터에서 web과 cats는 Amazon EC2 시작 유형의 [태스크 정의\(Task Definition\)](#)으로, dogs는 AWS Fargate 시작 유형으로 구성해볼 것입니다.

- 가운데의 Web은 ALB의 DNS 이름으로 접근할 수 있는 메인 웹 페이지로, I♥Cats와 I♥Dogs 배너를 클릭하면 각각 cats와 dogs로 이동합니다.
- Cats는 웹 화면을 새로고침할 때마다 고양이 사진을 랜덤으로 보여줍니다.
- Dogs는 웹 화면을 새로고침할 때마다 강아지 사진을 랜덤으로 보여줍니다.



## 2. 실습 환경 구성

이미 AWS 계정을 가지고 있다면 바로 이 실습의 가이드를 따라 진행할 수 있으나, 계정이 없다면 먼저 AWS 계정을 만들어야 합니다. 만약 임시 계정을 받으셨다면 (2-2) AWS 이벤트용 섹션을 참고하시기 바랍니다. AWS 계정 생성 및 활성화하기 위해서는 [링크](#)를 참조하시기 바랍니다.

\* 실습은 us-east-1 (N.Virginia), us-east-2 (Ohio), us-west-2 (Oregon), eu-west-1 (Ireland), ap-northeast-1 (Tokyo), ap-northeast-2(Seoul), ap-south-1 (Mumbai)에서 동작하며 그 외 리전에서는 지원하지 않습니다. 본 실습 시작 전, 실습에 필요한 리소스는 AWS CloudFormation 을 사용하여 구성합니다.

### (2-1) IAM 사용자

AWS 계정을 생성했지만 직접 IAM 사용자를 생성하지 않은 경우, IAM 콘솔을 사용하여 IAM 사용자를 생성할 수 있습니다. 다음 스텝에 따라 Administrator 를 생성합니다. 이미 관리자 사용자가 있다면, 다음 IAM 사용자 생성 작업을 건너뛰니다.

1. AWS 계정 이메일 주소와 비밀번호를 사용하여 AWS 계정의 **Root** 사용자로 [IAM](#)에 로그인 합니다.
2. IAM 콘솔 왼쪽 메뉴 패널에서 **Users** 를 선택한 다음 **Add user** 를 클릭합니다.
3. **User name** 은 **Administrator**로 입력합니다.
4. **AWS Management Console access** 체크박스를 선택하고, **Custom password** 를 선택한 다음 비밀번호를 입력합니다.
5. **Next: Permissions** 을 클릭합니다.

## Add user

1 2 3 4 5

## Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*	Administrator
<a href="#">Add another user</a>	

## Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

- Select AWS credential type\*
- Access key - Programmatic access  
Enables an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools.
  - Password - AWS Management Console access  
Enables a password that allows users to sign-in to the AWS Management Console.

Console password\*

Autogenerated password  
 Custom password

\*\*\*\*\*  
 Show password

## Require password reset

- User must create a new password at next sign-in  
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

Cancel

Next: Permissions

6. **Attach existing policies directly** 를 선택하고 **AdministratorAccess** 정책에 체크박스를 선택하고 Next: Tags 를 클릭합니다.

## Add user

1 2 3 4 5

## Set permissions

Add user to group	Copy permissions from existing user	<a href="#">Attach existing policies directly</a>																																													
<a href="#">Create policy</a>																																															
<p>Filter policies v Q administrator Showing 14 results</p> <table border="1"> <thead> <tr> <th>Policy name</th> <th>Type</th> <th>Used as</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/> AdministratorAccess</td> <td>Job function</td> <td>Permissions policy (3)</td> </tr> <tr> <td><input type="checkbox"/> AdministratorAccess-Amplify</td> <td>AWS managed</td> <td>None</td> </tr> <tr> <td><input type="checkbox"/> AdministratorAccess-AWSLambda</td> <td>AWS managed</td> <td>None</td> </tr> <tr> <td><input type="checkbox"/> AmazonAPIGatewayAdministrator</td> <td>AWS managed</td> <td>None</td> </tr> <tr> <td><input type="checkbox"/> AWSAppSyncAdministrator</td> <td>AWS managed</td> <td>None</td> </tr> <tr> <td><input type="checkbox"/> AWSAuditManagerAdministratorAccess</td> <td>AWS managed</td> <td>None</td> </tr> <tr> <td><input type="checkbox"/> AWSCloud9Administrator</td> <td>AWS managed</td> <td>None</td> </tr> <tr> <td><input type="checkbox"/> AWSGrafanaAccountAdministrator</td> <td>AWS managed</td> <td>None</td> </tr> <tr> <td><input type="checkbox"/> AWSSSOAdministrator</td> <td>AWS managed</td> <td>None</td> </tr> <tr> <td><input type="checkbox"/> AWSSOMasterAccountAdministrator</td> <td>AWS managed</td> <td>None</td> </tr> <tr> <td><input type="checkbox"/> AWSSOMemberAccountAdministrator</td> <td>AWS managed</td> <td>None</td> </tr> <tr> <td><input type="checkbox"/> DatabaseAdministrator</td> <td>Job function</td> <td>None</td> </tr> <tr> <td><input type="checkbox"/> NetworkAdministrator</td> <td>Job function</td> <td>None</td> </tr> <tr> <td><input type="checkbox"/> SystemAdministrator</td> <td>Job function</td> <td>None</td> </tr> </tbody> </table>			Policy name	Type	Used as	<input checked="" type="checkbox"/> AdministratorAccess	Job function	Permissions policy (3)	<input type="checkbox"/> AdministratorAccess-Amplify	AWS managed	None	<input type="checkbox"/> AdministratorAccess-AWSLambda	AWS managed	None	<input type="checkbox"/> AmazonAPIGatewayAdministrator	AWS managed	None	<input type="checkbox"/> AWSAppSyncAdministrator	AWS managed	None	<input type="checkbox"/> AWSAuditManagerAdministratorAccess	AWS managed	None	<input type="checkbox"/> AWSCloud9Administrator	AWS managed	None	<input type="checkbox"/> AWSGrafanaAccountAdministrator	AWS managed	None	<input type="checkbox"/> AWSSSOAdministrator	AWS managed	None	<input type="checkbox"/> AWSSOMasterAccountAdministrator	AWS managed	None	<input type="checkbox"/> AWSSOMemberAccountAdministrator	AWS managed	None	<input type="checkbox"/> DatabaseAdministrator	Job function	None	<input type="checkbox"/> NetworkAdministrator	Job function	None	<input type="checkbox"/> SystemAdministrator	Job function	None
Policy name	Type	Used as																																													
<input checked="" type="checkbox"/> AdministratorAccess	Job function	Permissions policy (3)																																													
<input type="checkbox"/> AdministratorAccess-Amplify	AWS managed	None																																													
<input type="checkbox"/> AdministratorAccess-AWSLambda	AWS managed	None																																													
<input type="checkbox"/> AmazonAPIGatewayAdministrator	AWS managed	None																																													
<input type="checkbox"/> AWSAppSyncAdministrator	AWS managed	None																																													
<input type="checkbox"/> AWSAuditManagerAdministratorAccess	AWS managed	None																																													
<input type="checkbox"/> AWSCloud9Administrator	AWS managed	None																																													
<input type="checkbox"/> AWSGrafanaAccountAdministrator	AWS managed	None																																													
<input type="checkbox"/> AWSSSOAdministrator	AWS managed	None																																													
<input type="checkbox"/> AWSSOMasterAccountAdministrator	AWS managed	None																																													
<input type="checkbox"/> AWSSOMemberAccountAdministrator	AWS managed	None																																													
<input type="checkbox"/> DatabaseAdministrator	Job function	None																																													
<input type="checkbox"/> NetworkAdministrator	Job function	None																																													
<input type="checkbox"/> SystemAdministrator	Job function	None																																													

## Set permissions boundary

7. **Review** 화면에서 다시 확인 후 **Create user** 버튼을 클릭합니다.

8. 아래와 같이 직접 만든 User 를 클릭 후 Summary > Security credentials 탭을 보면 직접 접근할 수 있는 URL 이 보입니다. Root 사용자를 로그아웃하고 새로 생성한 Administrator 사용자로 로그인을 합니다. (IAM user name: Administrator, Password: 직접 설정한 패스워드)

Users > Administrator

### Summary

User ARN: arn:aws:iam:[REDACTED]user/Administrator

Path: /

Creation time: 2022-06-04 11:26 UTC+0900

**Permissions** **Groups** **Tags** **Security credentials** **Access Advisor**

**Sign-in credentials**

Summary	Console sign-in link: https://[REDACTED].signin.aws.amazon.com/console
Console password	Enabled (never signed in)   Manage
Assigned MFA device	Not assigned   Manage
Signing certificates	None

## (2-2) AWS 이벤트용 계정

1. 임시 계정을 요청하셔서 링크를 받으신 분들은 접속 시 아래와 같은 화면이 보입니다. 여기서 **Email One-Time Password (OTP)**를 클릭합니다.

**Sign in with**  
Pick the sign-in method you prefer

**Email One-Time Password (OTP)**  
Enter your personal or corporate email to receive a one-time password

**Login with Amazon**  
Login with your Amazon.com retail account

**Amazon Employee**  
(For Amazon Employees Only) Login with your Amazon Corporate account

[Get help signing in](#)

2. 아래 화면이 나오면 본인의 이메일 계정을 입력하시고 **Send passcode** 버튼을 누릅니다.

**One-time email passcode**

Send a passcode to the email below.

Email

[Back](#) [Send passcode](#)

[Get help signing in](#)

3. 이메일을 통해 passcode 를 받으면 입력 후 **Sign in** 을 클릭합니다.

4. 이후 아래 화면이 보이면 **AWS Console** 을 클릭합니다.

The screenshot shows the 'Team Dashboard' interface. At the top, there's a navigation bar with 'Event' and three buttons: 'Set Team Name' (green), 'AWS Console' (blue, with a red box around it), and 'SSH Key' (light blue). Below the navigation bar, there's a section titled 'Event: imd for sec1' with the sub-section 'Event Name: (Team Name Not Set Yet)'. Further down, there are fields for 'Event ID:' and 'Team ID:', both of which are redacted with a large red box.

5. 이후 아래 화면에서는 Open AWS Console 을 클릭합니다.

#### AWS Console Login

**Remember to only use "ap-northeast-2" as your region, unless otherwise directed by the event operator.**

Login Link

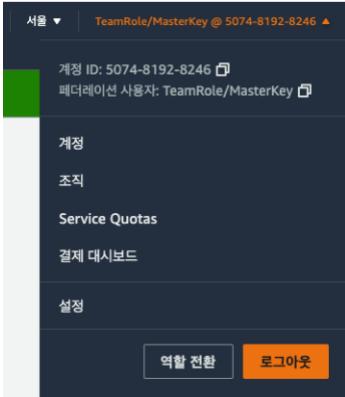


6. 아래와 같이 AWS Console 초기 화면이 보입니다.

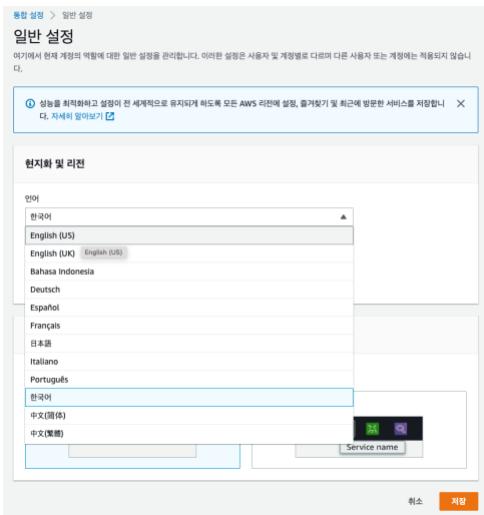
The screenshot shows the 'Console Home' page. At the top, there's a search bar and a user profile. On the left, there's a sidebar with 'Recently visited' services: VPC, S3, CloudFormation, EC2, Cloud9, CloudWatch, Simple Notification Service, and AWS AppConfig. Below this is the 'AWS Health' section, which says 'No health data' and provides a link to 'Go to AWS Health'. To the right of the sidebar are two main sections: 'Welcome to AWS' and 'Cost and usage'. The 'Welcome to AWS' section includes links to 'Getting started with AWS', 'Training and certification', and 'What's new with AWS?'. The 'Cost and usage' section displays current month costs (\$0.00), forecasted month end costs (\$0.00), and last month costs (\$0.00). It also includes a note about cost breakdowns and a link to 'Go to AWS Cost Management'. On the far right, there's a sidebar with tips for customizing the console home, changing the language, and learning more about the AWS Management Console.

7. 실습은 영문으로 진행되므로 필요에 따라 언어 설정을 변경합니다.

8. 기본 설정이 한글로 되어 있다면 우측 상단 계정을 클릭 후 설정을 클릭합니다.



9. 이후 언어 설정에서 영어를 선택하신 후 저장 버튼을 클릭합니다.



10. 이제 설정이 마무리되었고 CloudFormation 스택을 배포해서 본격적으로 실습을 시작합니다.

### 3. CloudFormation Stack 배포

#### (3-1) CloudFormation Template

Amazon ECS Cats and Dogs 실습에 필요한 AWS 리소스를 사전에 생성하기 위해 제공된 CloudFormation template 을 사용하여 CloudFormation stack 을 생성합니다. 스택을 생성하면 실습에 사용할 VPC 리소스, ECS 인스턴스와 ALB 가 사용할 보안 그룹, Workstation 인스턴스와 IAM 리소스 등이 생성됩니다. 이 모든 리소스는 Cats and Dogs 실습을 진행하는 데 필요합니다.

- CloudFormation 스택을 시작하려면, 아래 **Launch Stack** 버튼이나 이 [링크](#)를 클릭해서 CloudFormation 콘솔로 이동합니다.

**Launch Stack**

\* 중요: 이 템플릿은 us-east-1 (N.Virginia), us-east-2 (Ohio), us-west-2 (Oregon), eu-west-1 (Ireland), ap-northeast-1 (Tokyo), ap-northeast-2(Seoul), ap-south-1 (Mumbai)에서 동작하며 그 외 리전에서는 지원하지 않습니다.

다른 방법으로는 이 [링크](#)를 클릭하셔서 직접 yaml 파일을 다운로드 받으신 후 upload 하는 방법도 있습니다.

2. Launch Stack 을 클릭하면 다음과 같이 **Create stack** 화면이 보입니다. Amazon S3 URL 이 자동으로 입력이 되어 있습니다. 만약 yaml 파일을 직접 다운로드 받으셨다면 Upload a template file 을 클릭하신 후 yaml 파일을 업로드하시면 됩니다. 이후 **Next** 버튼을 클릭합니다.

Create stack

Prerequisite - Prepare template

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

Template is ready    Use a sample template    Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL    Upload a template file

Amazon S3 URL  
https://ecs-demogo-pictures.s3.ap-northeast-2.amazonaws.com/ecs-c9.yaml

Amazon S3 template URL

S3 URL: https://ecs-demogo-pictures.s3.ap-northeast-2.amazonaws.com/ecs-c9.yaml   View in Designer

Cancel   **Next**

3. **Stack name** 은 'ecs-demogo'로 보입니다. 그대로 **Next** 를 클릭합니다.

Specify stack details

Stack name

Stack name  
ecs-demogo

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

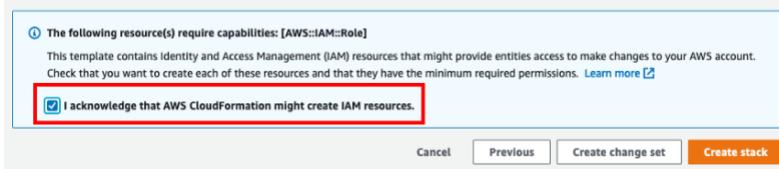
Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

No parameters  
There are no parameters defined in your template

Cancel   Previous   **Next**

4. 이후 **Configure stack options** 화면에서도 변경 없이 **Next** 를 클릭합니다.
5. **Review ecs-demogo** 화면에서는 하단에 있는 창에서 "I acknowledge that AWS CloudFormation might create IAM resources" 부분 체크를 한 후 **Create stack** 버튼을 클릭합니다.



6. 아래와 같이 자동으로 Stack 이 생성됩니다. Resources 탭을 통해 생성되고 있는 목록을 확인하실 수 있습니다.

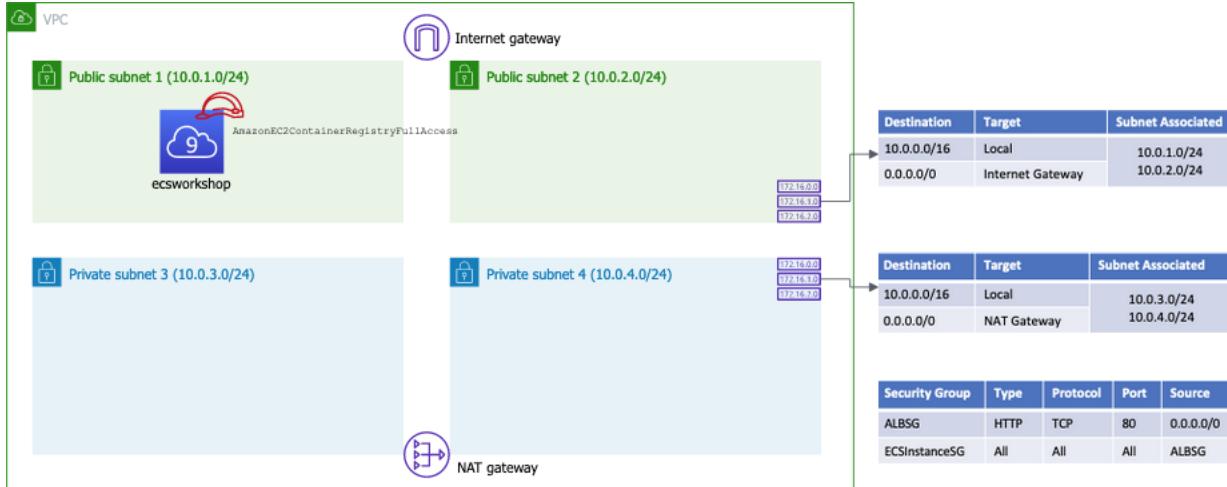
Logical ID	Physical ID	Type	Status	Status reason	Module
ALB	sg-0474bef5a04862164	AWS::EC2::SecurityGroup	CREATE_COMPLETE	-	-
AttachGateway	ecs-d-Attac-7AWDY79P5ADH	AWS::EC2::VPGatewayAttachment	CREATE_COMPLETE	-	-
DHCPOptions	dopt-0cf8a21a3703a926c	AWS::EC2::DHCPOptions	CREATE_COMPLETE	-	-

7. 약 4~5 분 정도 지나면 아래와 같이 Create\_Complete 이라고 보입니다.

Stack ID	arn:aws:cloudformation:ap-northeast-2:183837691404:stack/ecs-demogo/3daa6b80-e3b4-11ec-af98-020a09c91a72	Description	Demo-ECS200 Prerequisite - Network CloudFormation Template
Status	CREATE_COMPLETE	Status reason	-
Root stack	-	Parent stack	-
Created time	2022-06-04 12:13:01 UTC+0900	Deleted time	-
Updated time	-	Last drift check time	-
Drift status	NOT_CHECKED	IAM role	-
Termination protection	Disabled		-

### (3-2) Stack 배포 결과

Stack 생성이 완료되면 실습하는데 필요한 기본적인 네트워크 구성 (VPC, Subnet, Route Table)과 instance 가 생성됩니다. 생성된 아키텍처는 아래와 같습니다.



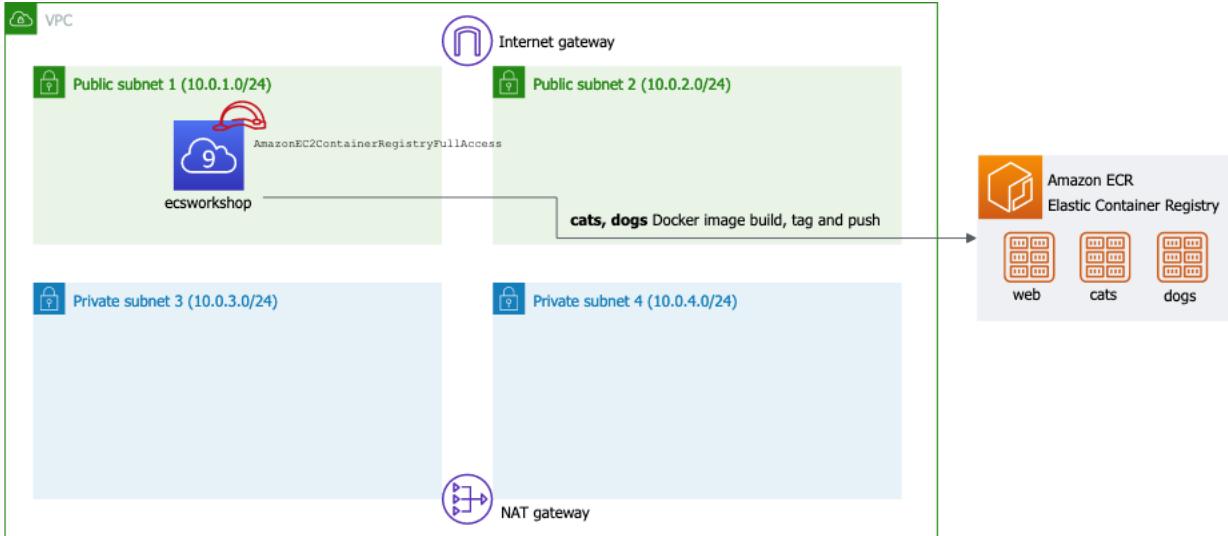
## 4. Amazon Elastic Container Registry (ECR)

Amazon Elastic Container Registry(ECR)를 Amazon ECS 와 통합하여 Amazon ECS 에서 실행되는 애플리케이션에 대한 컨테이너 이미지를 손쉽게 저장, 실행 및 관리할 수 있습니다. 태스크 정의에 Amazon ECR 리포지토리를 지정하기만 하면 Amazon ECS 에서 애플리케이션에 적합한 이미지를 가져옵니다.



실습은 아래 단계로 진행됩니다.

- cats, dogs ECR 리포지토리 생성하기
- Cloud9 IDE에서 cats, dogs 도커 이미지 빌드하고 태깅하기
- 작업한 도커 이미지를 Amazon ECR 로 푸시하기



## (4-1) AWS Cloud9 권한 설정

AWS Cloud9은 브라우저만으로 코드를 작성, 실행 및 디버깅할 수 있는 클라우드 기반 IDE(Integrated Development Environment, 통합 개발 환경)입니다. 실습자는 다음 단계에서 Cloud9에서 도커 이미지를 빌드하고 레포지토리로 푸시하는 작업 등을 수행하게 됩니다. 그러기 위해서는 Cloud9에 Amazon ECR 권한을 가진 IAM Role을 인스턴스 프로파일을 통해 부여하는 작업이 선행되어야 합니다.

1. IAM role을 확인하기 위해 [AWS IAM](#) 메뉴로 이동합니다.
2. 우리는 이미 CloudFormation을 이용해 Cloud9이 사용할 **WorkstationRole**을 자동으로 생성했습니다. IAM Role에서 Search에 workstationrole을 입력하여 검색합니다.

Role name	Trusted entities
ecs-demogo-WorkstationRole-1GP4JUKRFWQ2K	AWS Service: ec2

3. Role 이름을 클릭하면 해당 IAM 역할이 가진 ecs-demogo-WorkstationProfile이라는 **Instance Profile ARNs**를 확인할 수 있습니다. 인스턴스 프로파일은 EC2 인스턴스와 IAM 역할을 매개합니다. Permissions policies에서 Amazon ECR로 작업하기 위한 **AmazonEC2ContainerRegistryFullAccess** 정책이 연결되어 있음을 확인합니다.

**Summary**

Creation date	June 04, 2022, 12:13 (UTC+09:00)	ARN	arn:aws:iam::[REDACTED]role/ecs-demogo-WorkstationRole-1GP4JUKRFWQ2K	Instance profile ARN	arn:aws:iam::[REDACTED]instance-profile/ecs-demogo-WorkstationProfile-2rW7XAo74ecC
Last activity	None	Maximum session duration	1 hour		

**Permissions** | Trust relationships | Tags | Access Advisor | Revoke sessions

**Permissions policies (2)**  
You can attach up to 10 managed policies.

Policy name	Type	Description
AmazonEC2ContainerRegistryFullAccess	AWS managed	Provides administrative access to
AmazonS3FullAccess	AWS managed	Provides full access to all buckets

4. 이번에는 **Cloud9** 으로 사용하는 EC2 instance 에게 Role 을 부여하기 위해 [Amazon EC2](#) 메뉴로 이동합니다. 왼쪽 링크를 클릭하거나 EC2 메뉴를 검색해서 들어갑니다.
5. 아래와 같이 **aws-cloud9-ecsworkshop-###** 이라는 instance 가 보입니다. Role 을 부여하기 위해 **Action** > **Security** -> **Modify IAM role** 을 클릭합니다.

6. 그리고 아래 그림과 같이 "ecs-demogo-workstationprofile-#####" 선택 후 **Update IAM role** 을 클릭합니다.

**Modify IAM role** [Info](#)  
Attach an IAM role to your instance.

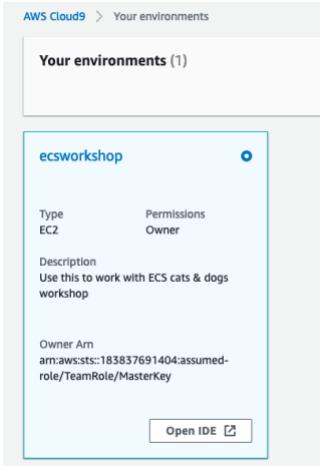
Instance ID  
aws-cloud9-ecsworkshop-[REDACTED]

IAM role  
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

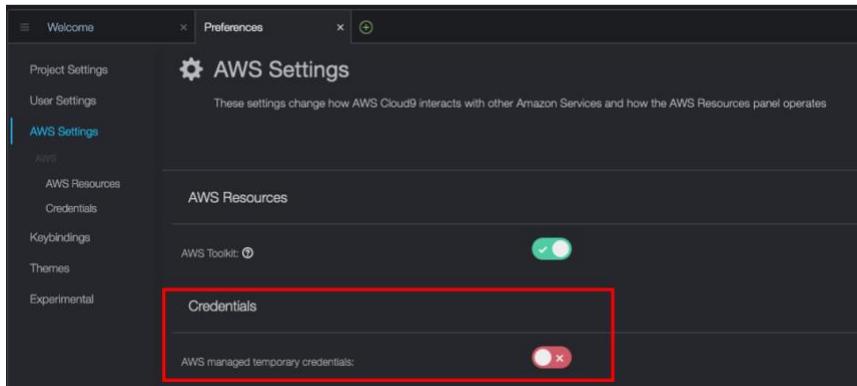
ecs-demogo-WorkstationProfile-2rW7XAo74ecC [Create new IAM role](#)

[Cancel](#) [Update IAM role](#)

7. 이제 [AWS Cloud9](#) 으로 이동합니다. **ecsworkshop** 을 선택하고 **Open IDE** 를 클릭합니다.



8. AWS Cloud9 화면에서 우측 상단에 있는 기어 아이콘을 클릭한 후, 사이드 바에서 **AWS SETTINGS**를 클릭합니다. 그리고 Credentials 항목에서 **AWS managed temporary credentials** 설정을 비활성화합니다.



9. 설정을 마친 후 Preference tab 을 닫고, 하단 터미널 창에서 Temporary credentials 이 없는지 확실하기 위해 기존의 자격 증명 파일도 제거합니다.

```
rm -vf ${HOME}/.aws/credentials
```

10. 그리고 아래 명령어를 통해 Cloud9 IDE 가 **ecs-demogo-WorkstationRole** 을 사용하고 있는지 확인하시기 바랍니다. 아래와 같은 결과가 나오면 정상입니다.

```
aws sts get-caller-identity --query Arn | grep ecs-demogo
```

```
TeamRole:~/environment $ rm -vf ${HOME}/.aws/credentials
TeamRole:~/environment $ aws sts get-caller-identity --query Arn | grep ecs-demogo
"arn:aws:sts::183837691404:assumed-role/ecs-demogo-WorkstationRole-1GP4JUKRFWQ2K/i-07d00255c2fdefea3"
TeamRole:~/environment $
```

11. 이후 Cloud9 에서 사용할 도구들을 업데이트 및 설치합니다.

```
sudo pip install --upgrade awscli
```

```
sudo yum install -y jq
```

12. 현재 실습하고 있는 리전을 기본값으로 설정하기 위해 아래 명령어를 터미널에 입력합니다.

```
export AWS_REGION=$(curl -s 169.254.169.254/latest/dynamic/instance-identity/document | jq -r '.region')
```

```
echo "export AWS_REGION=${AWS_REGION}" | tee -a ~/.bash_profile
```

```
aws configure set default.region ${AWS_REGION}
```

13. 설정한 리전을 확인합니다.

```
aws configure get default.region
```

## (4-2) ECR 생성

이번 실습에서는 cats 와 dogs 가 사용할 ECR 리포지토리를 생성합니다. Amazon ECR 는 개발자가 Docker 컨테이너 이미지를 손쉽게 저장, 관리 및 배포할 수 있게 해주는 완전관리형 Docker 컨테이너 레지스트리로, Amazon ECS 와 통합되어 개발에서 프로덕션까지의 워크플로우를 간소화합니다.

1. [Amazon ECR](#) 로 이동합니다. ECR 을 처음 사용하는 경우 Welcome 페이지가 보입니다. Get Started 를 클릭하거나 좌측 네비게이션 바에서 **Repositories** 를 클릭합니다.



2. 아래 그림과 같이 다른 설정은 그대로 두고 **Repository name** 만 cats 로 지정한 후 **Create repository** 버튼을 클릭합니다.

**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.

dkr.ecr.ap-northeast-2.amazonaws.com/**cats**

4 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.

3. 같은 방식으로 **dogs** repository 를 생성합니다.
4. 아래와 같이 2 개의 repository 가 생성된 것을 확인할 수 있습니다.
5. repository 의 이름은 반드시 cats, dogs 로 하셔야 다음 이미지 태그하고 ECR 로 푸시하기 단계를 진행하실 수 있습니다.

Private repositories (2)						
<input type="text"/> Find repositories		<input type="button"/> View push commands	<input type="button"/> Delete	<input type="button"/> Actions	<input type="button"/> Create repository	
Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
<input type="radio"/> <b>cats</b>	<input type="button"/> dkr.ecr.ap-northeast-2.amazonaws.com/cats	June 04, 2022, 14:20:10 (UTC+09)	Disabled	Manual	AES-256	Inactive
<input type="radio"/> <b>dogs</b>	<input type="button"/> dkr.ecr.ap-northeast-2.amazonaws.com/dogs	June 04, 2022, 14:20:16 (UTC+09)	Disabled	Manual	AES-256	Inactive

### (4-3) 도커 이미지

이번 단계에서 실습자는 ecsworkshop Cloud9에서 Docker CLI를 이용해 cats 와 dogs 도커 이미지를 빌드합니다.

1. 다시 [AWS Cloud9](#) 으로 이동 후 **ecsworkshop** 을 선택하고 **Open IDE** 를 클릭합니다.
2. 앞으로 빌드할 cats 와 dogs 의 Dockerfile 과 애플리케이션 소스 등을 확인합니다. 도커는 Dockerfile 을 읽어 자동으로 이미지를 빌드합니다. cats 디렉토리의 Dockerfile 은 cats 도커 이미지를 빌드하기 위해 필요합니다. 어떤 내용을 담고 있는지 리눅스 cat 명령어를 이용하여 확인합니다.

```
cd ecsworkshop/cats
```

```
cat Dockerfile
```

```
TeamRole:~/environment $ cd ecsworkshop/cats
TeamRole:~/environment/ecsworkshop/cats (master) $ cat Dockerfile
FROM nginx
EXPOSE 80
RUN apt-get update -y && \
    apt-get upgrade -y && \
    apt-get install -y curl && \
    cd /tmp && \
    apt-get install awscli -y && \
    rm -rf /tmp/* && \
    rm -rf /var/lib/apt/lists/*
COPY ./default.conf /etc/nginx/conf.d/default.conf
COPY ./index.html /usr/share/nginx/html/index.html
CMD nginx -g "daemon off;"TeamRole:~/environment/ecsworkshop/cats (master) $
```

3. cats 를 빌드합니다.

```
docker build -t cats .
```

4. dogs 도 동일한 작업을 수행하기 위해 디렉토리 이동 후 빌드합니다. cats 이미지를 빌드하면서 nginx 이미지를 받아오고 여러 도구를 설치하거나 업그레이드하는 과정을 수행했기 때문에 Using cache 라는 메세지와 함께 cats 보다 훨씬 빠르게 수행되는 것을 알 수 있습니다.

```
cd ..
```

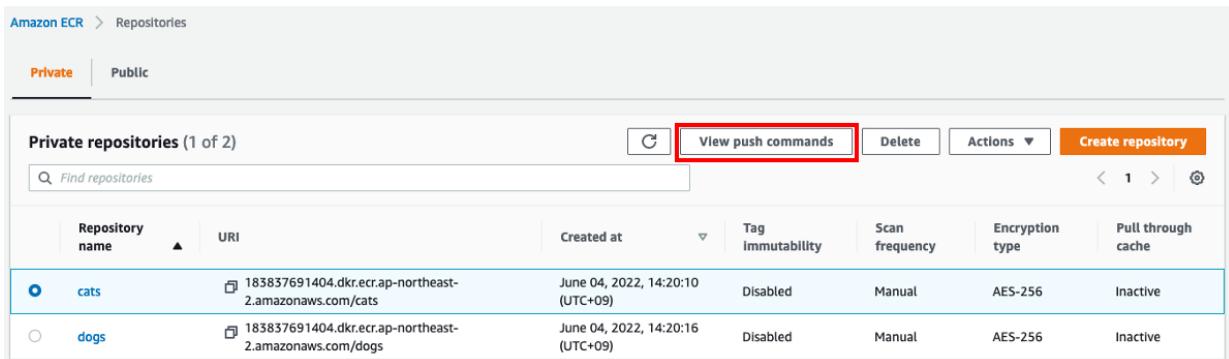
```
cd dogs
```

```
docker build -t dogs .
```

#### (4-4) 이미지 태그하고 ECR 로 푸시하기

이번 단계에서 실습자는 앞 단계에서 작업한 도커 이미지를 latest로 태깅한 후 cats, dogs repository로 푸시합니다.

1. cats 와 dogs 이미지를 태깅한 후 각각 ECR 로 푸시합니다. Amazon ECR 의 Repositories 에서 **View push command** 를 클릭하여 빌드, 태깅, 푸시 명령어를 참조합니다.



Private repositories (1 of 2)						
Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
<input checked="" type="radio"/> cats	<a href="#">183837691404.dkr.ecr.ap-northeast-2.amazonaws.com/cats</a>	June 04, 2022, 14:20:10 (UTC+09)	Disabled	Manual	AES-256	Inactive
<input type="radio"/> dogs	<a href="#">183837691404.dkr.ecr.ap-northeast-2.amazonaws.com/dogs</a>	June 04, 2022, 14:20:16 (UTC+09)	Disabled	Manual	AES-256	Inactive

2. cats 에서 인증 토큰을 발급받는 절차부터 차례대로 명령어 샘플을 제공합니다. 앞 단계에서 cats 이미지를 이미 빌드했으므로 1, 3, 4 번만 차례대로 수행합니다.

**Push commands for cats**

macOS / Linux Windows

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry Authentication](#).

1. Retrieve an authentication token and authenticate your Docker client to your registry.  
Use the AWS CLI:  

```
aws ecr get-login-password --region ap-northeast-2 | docker login --username AWS --password-stdin
183837691404.dkr.ecr.ap-northeast-2.amazonaws.com
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.
2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:  

```
docker build -t cats .
```
3. After the build completes, tag your image so you can push the image to this repository:  

```
docker tag cats:latest 183837691404.dkr.ecr.ap-northeast-2.amazonaws.com/cats:latest
```
4. Run the following command to push this image to your newly created AWS repository:  

```
docker push 183837691404.dkr.ecr.ap-northeast-2.amazonaws.com/cats:latest
```

3. 완료 후 dogs 도 동일하게 **View push command** 를 클릭하여 빌드, 태깅, 푸시 명령어를 참조해서 수행합니다.
4. 명령어 수행 후 cats 와 dogs 각각 latest 태그를 가진 도커 이미지가 잘 저장되었는지 확인합니다.

**cats**

**Images (1)**

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
<input type="checkbox"/>	latest	Image	June 04, 2022, 15:33:08 (UTC+09)	139.27	<a href="#">Copy URI</a>	<a href="#">sha256:8ed51c2d248d9c...</a>	-	-

**dogs**

**Images (1)**

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
<input type="checkbox"/>	latest	Image	June 04, 2022, 15:33:52 (UTC+09)	139.27	<a href="#">Copy URI</a>	<a href="#">sha256:09fcc0eaba7c55f...</a>	-	-

## 5. Amazon Elastic Container Service (ECS)

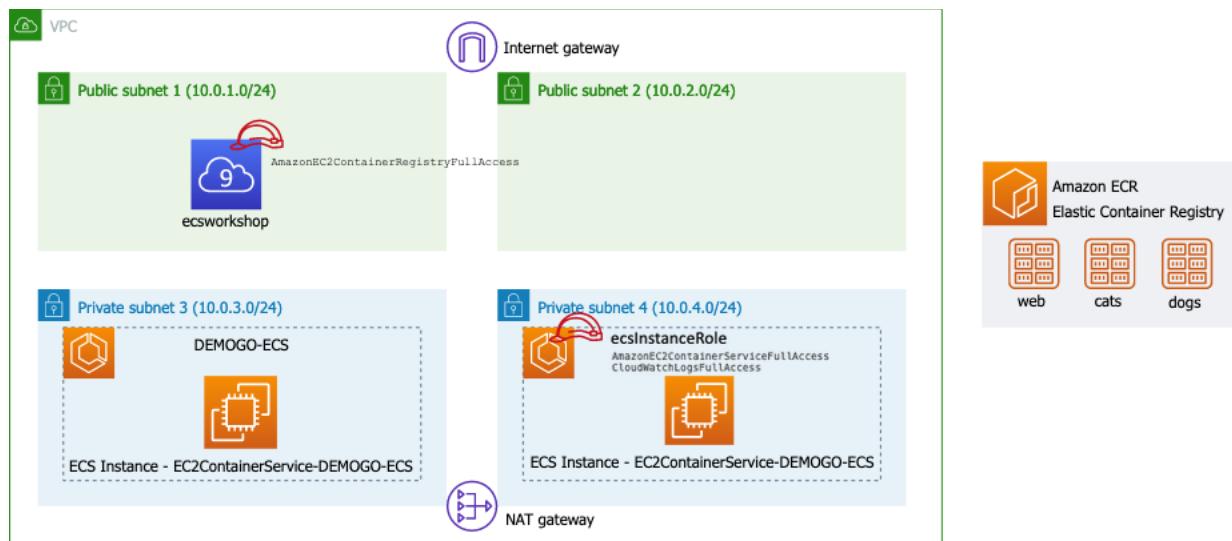
본 실습은 ECS 클러스터 생성, ECS 태스크 정의, ECS 서비스, 서비스 확인 순서대로 진행됩니다.

### (5-1) ECS 클러스터

Amazon ECS 클러스터는 작업 또는 서비스의 논리적 그룹입니다. 이번 실습에서는 고가용성을 위해 2 개 가용영역에 DEMOGO-ECS 클러스터를 배포하고, 모니터링 실습에 필요한 IAM 권한을 ECS 인스턴스에 부여합니다.

클러스터를 생성하면 아래와 같은 아키텍처로 리소스가 배포됩니다. 순서는 아래와 같습니다.

- DEMOGO-ECS 클러스터 생성
- ecsInstanceRole 에 CloudWatch Logs 권한 부여



#### (5-1-1) ECS 클러스터 생성

이번 챕터에서는 EC2 Linux 클러스터 DEMOGO-ECS 를 생성합니다. 현재 새로운 UI 가 일부 적용되었지만 전체 적용되지 않은 상태이므로 실습때는 기존 UI 를 사용합니다.

**Introducing the new ECS console experience!**  
We have redesigned the cluster experience. The actions for updating clusters, deleting cluster and using Spot Instances for your Auto Scaling group allocation strategy are available with the previous console while we complete the migration. To use these actions, [use the classic console](#).

1. [Amazon ECS](#) 로 이동합니다. 초기 화면에서 왼쪽 메뉴 중 **Clusters** 를 선택하고 **Create Cluster** 를 클릭합니다.



2. 우리는 EC2 기반의 클러스터를 생성할 예정이므로 아래와 같이 **EC2 Linux + Networking** 을 선택하고 **Next Step** 을 클릭합니다.

#### Select cluster template

The following cluster templates are available to simplify cluster creation. Additional configuration and integrations can be added later.

Template	Resources to be created
Networking only	Cluster, VPC (optional), Subnets (optional)
EC2 Linux + Networking	Cluster, VPC, Subnets, Auto Scaling group with Linux AMI <small>For use with either AWS Fargate (Windows/Linux) or with External instance capacity.</small>
EC2 Windows + Networking	Cluster, VPC, Subnets <small>Auto Scaling group with Windows AMI</small>

3. **Configure Cluster** 부분에서 상세 항목을 아래와 같이 설정합니다.

- Cluster name: DEMOGO-ECS
- Provisioning model: On-Demand Instance
- EC2 Instance type: t3.medium
- Number of instances: 2
- EC2 AMI id: Amazon Linux 2 AMI
- EBS storage: 30

## Configure cluster

Cluster name\* DEMOGO-ECS 

Create an empty cluster

Instance configuration

Provisioning Model  On-Demand Instance  
With On-Demand Instances, you pay for compute capacity by the hour, with no long-term commitments or upfront payments.

Spot  
Amazon EC2 Spot Instances let you take advantage of unused EC2 capacity in the AWS cloud. Spot Instances are available at up to a 90% discount compared to On-Demand prices.  
[Learn more](#)

EC2 instance type\* t3.medium     
 Manually enter desired instance type

Number of instances\* 2 

EC2 AMI ID\* Amazon Linux 2 AMI [ami-0ee1...   

Root EBS Volume Size (GiB) 30 

Key pair None - unable to SSH     
You will not be able to SSH into your EC2 instances without a key pair. You can create a new key pair in the [EC2 console](#).

4. 다음은 아래쪽 Networking 설정입니다. CloudFormation으로 구성한 환경이 있기 때문에 VPC는 DemoGoECSVPC를 선택하셔야 합니다. 그리고 Subnet은 Private Subnet 2개를 선택하고 Security Group은 ecs-demogo-ECSInstanceSG를 선택합니다.
  - a. VPC: DemoGoECSVPC (10.0.0.0/16)
  - b. Subnets: Private subnet 1,2 (10.0.3.0/24, 10.0.4.0/24)
  - c. Security Group: ecs-demogo-ECSInstanceSG

## Networking

Configure the VPC for your container instances to use. A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You can choose an existing VPC, or create a new one with this wizard.

VPC **vpc-088fd63a7fe55f0bf...**     
Check the structure for [vpc-088fd63a7fe55f0bf](#) in the Amazon EC2 console.

Subnets    
**subnet-0168d3ec12d34ad**  3a  
(10.0.4.0/24) | PrivateSubn  
et2 - ap-northeast-2b  
assign ipv6 on creation: Di  
sabled  
  
**subnet-091e7af56bcb5d42**  0  
(10.0.3.0/24) | PrivateSubn  
et1 - ap-northeast-2a  
assign ipv6 on creation: Di  
sabled  
  
Select a subnet... 

Auto assign public IP Use subnet setting     
Rules for [sg-05ed4c9b5e5f7b86c](#) in the EC2 Console.

5. IAM role 은 create new 를 선택하고 **Enable Container Insights** 를 체크한 후 **Create** 버튼을 클릭해서 클러스터를 생성합니다. **ecsInstanceRole** 라는 이름으로 자동 생성합니다.

Container instance IAM role

The Amazon ECS container agent makes calls to the Amazon ECS API actions on your behalf, so container instances that run the agent require the `ecsInstanceRole` IAM policy and role for the service to know that the agent belongs to you. If you do not have the `ecsInstanceRole` already, we can create one for you.

**Container Instance IAM role** Create new role

For container instances to receive the new ARN and resource ID format, the root user needs to opt in for the container instance IAM role. Opt in and try again.

**Tags**

Key	Value
Add key	Add value

**CloudWatch Container Insights**

CloudWatch Container Insights is a monitoring and troubleshooting solution for containerized applications and microservices. It collects, aggregates, and summarizes compute utilization such as CPU, memory, disk, and network; and diagnostic information such as container restart failures to help you isolate issues with your clusters and resolve them quickly. Learn more

**CloudWatch Container Insights**  Enable Container Insights

6. 아래와 같이 클러스터가 정상적으로 생성된 것을 확인합니다.

**Cluster : DEMOGO-ECS**

Get a detailed view of the resources on your cluster.

Cluster ARN arn:aws:ecs:ap-northeast-2:183837691404:cluster/DEMOGO-ECS

Status ACTIVE

Registered container instances 2

Pending tasks count	0 Fargate, 0 EC2, 0 External
Running tasks count	0 Fargate, 0 EC2, 0 External
Active service count	0 Fargate, 0 EC2, 0 External
Draining service count	0 Fargate, 0 EC2, 0 External

**ECS Instances** **Metrics** **Scheduled Tasks** **Tags** **Capacity Providers**

An Amazon ECS instance is either an External instance registered using ECS Anywhere or an Amazon EC2 instance. To register an External instance, choose Register External Instances and follow the steps. [Learn More](#) To register an Amazon EC2 instance, you can use the Amazon EC2 console. [Learn More](#)

Last updated on June 4, 2022 7:16:17 PM (0m ago)									
Status: <b>(ALL) ACTIVE DRAINING</b> < 1-2 > Page size 50									
Filter by attributes (click or press down arrow to view filter options)									
Container Instance	ECS Instance	Availability Zon...	External Insta...	Agent Connec...	Status	Running tasks...	CPU available	Memory availa...	Agent version
<input type="checkbox"/> 3ce8563fccb24d14b05684...	i-0d173fd8bea9...	ap-northeast-2b	false	true	ACTIVE	0	2048	3883	1.61.1
<input type="checkbox"/> d75ef751e451487d9ebc91...	i-03b9070e093...	ap-northeast-2a	false	true	ACTIVE	0	2048	3883	1.61.1

## (5-1-2) ECS IAM Role 권한 추가

ECS 인스턴스는 CloudWatch Logs 로 컨테이너 로그를 송출하기 위해 액세스 권한이 필요합니다. 앞 단계에서 자동 생성한 ECS 인스턴스의 IAM 역할인 `ecsInstanceRole` 에 `CloudWatchLogsFullAccess` 를 추가합니다.

1. IAM Role 로 이동한 후 **ecsInstancerole** 을 검색합니다.
2. 해당 Role 선택 후 **Attach policies** 를 클릭합니다.

**ecsInstanceRole****Summary****Delete****Edit**

Creation date  
June 04, 2022, 16:34 (UTC+09:00)

ARN  
arn:aws:iam:█████████████████████:role/ecsInstanceRole

Last activity  
37 minutes ago

Maximum session duration  
1 hour

Instance profile ARN  
arn:aws:iam:█████████████████████:instance-profile/ecsInstanceRole

---

**Permissions**    **Trust relationships**    **Tags**    **Access Advisor**    **Revoke sessions**

---

**Permissions policies (1)**  
You can attach up to 10 managed policies.

Filter policies by property or policy name and press enter

**Attach policies** (highlighted with a red box)    **Create inline policy**

Policy name	Type	Description
AmazonEC2ContainerServiceforEC2Role	AWS managed	Default policy for the Amazon EC2 F

3. 아래와 같이 **CloudWatchLogsFullAccess** 정책을 검색하고 선택 후 **Attach policies** 를 클릭합니다.

**Other permissions policies (755)**

Filter policies by property or policy name and press enter    1 match

"CloudWatchLogsFullAccess" X    **Clear filters**

Policy name	Type
CloudWatchLogsFullAccess	AWS managed

4. 이후 아래와 같이 정책이 추가되었는지 확인합니다.

---

**Permissions**    **Trust relationships**    **Tags**    **Access Advisor**    **Revoke sessions**

---

**Permissions policies (2)**  
You can attach up to 10 managed policies.

Filter policies by property or policy name and press enter

Policy name
CloudWatchLogsFullAccess
AmazonEC2ContainerServiceforEC2Role

**(5-2) ECS 태스크 정의**

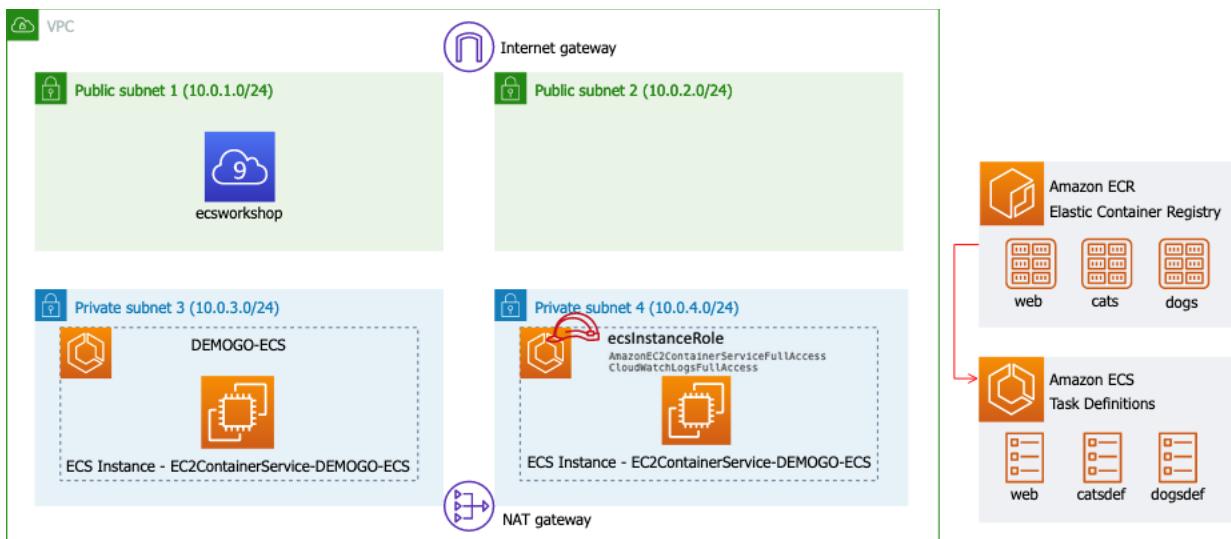
Amazon ECS에서 Docker 컨테이너를 실행하려면 태스크 정의(Task Definition)가 필요합니다. Amazon ECR 실습에서 생성한 cats, dogs의 도커 이미지를 참조하는 catsdef, dogsdef 태스크 정의를 생성합니다. web 태스크 정의가 사용할 도커 이미지는 별도로 제공됩니다.

태스크 정의에서 지정할 수 있는 몇몇 파라미터는 다음과 같습니다. 각 [파라미터](#)에 대해 더 알아봅니다.

- 작업의 각 컨테이너에 사용할 도커 이미지
- 각 작업 또는 작업 내 각 컨테이너에서 사용할 CPU 및 메모리의 양
- 사용할 시작 유형(Launch Type)으로서 해당 작업(Task)이 호스팅되는 인프라를 결정
- 작업의 컨테이너에 사용할 도커 네트워킹 모드
- 작업에 사용할 로깅 구성 등

실습 순서는 아래와 같고 완성되면 그림과 같은 아키텍처가 완성됩니다.

1. web 태스크 정의 생성
2. catsdef 태스크 정의 생성
3. dogsdef 태스크 정의 생성



### (5-2-1) Web 태스크 정의

Web 태스크 정의는 EC2 타입으로 생성합니다. Web, Cats, Dogs의 도커 이미지를 만드는 매우 유사하기 때문에 반복 과정을 줄이기 위해서 Web 이 사용할 도커 이미지는 미리 만들어 둔 실습에서 제공하는 URI를 사용합니다. 그 외에 별도로 설명하지 않은 설정은 기본값으로 둡니다.

1. [Amazon ECS](#)로 이동 후 Task definition로 이동하여 **Create new Task Definition**을 클릭합니다.
2. **Select launch type compatibility**에서는 EC2를 선택합니다.
3. **Configure task and container definitions**에서는 **Task definition name**만 **web**으로 설정합니다.

## Configure task and container definitions

A task definition specifies which containers are included in your task and how they interact with each other. You can also specify data volumes for your containers to use. [Learn more](#)

Task definition name\*

Requires compatibility\* EC2

Task role

Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon Elastic Container Service Task Role in the [IAM Console](#).

Network mode

If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. Windows tasks support the <default> and awsvpc network modes.

4. 중간에 있는 **Container definitions** 아래 **Add container** 를 클릭합니다.

Container definitions

Container ...	Image	Hard/Soft ...	CPU Units	GPU	Inference A...	Essential
No results						

5. Web 컨테이너를 아래와 같이 설정합니다.

- Container name:
- Image:
- Memory Limits - Hard limit
- Port mappings: Host port: , Container port:

Container name\*

Image\*

Private repository  authentication\*

Memory Limits (MiB)\*

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the 'memory' and 'memoryReservation' parameters, respectively, in task definitions. ECS recommends 300-500 MiB as a starting point for web applications.

Port mappings

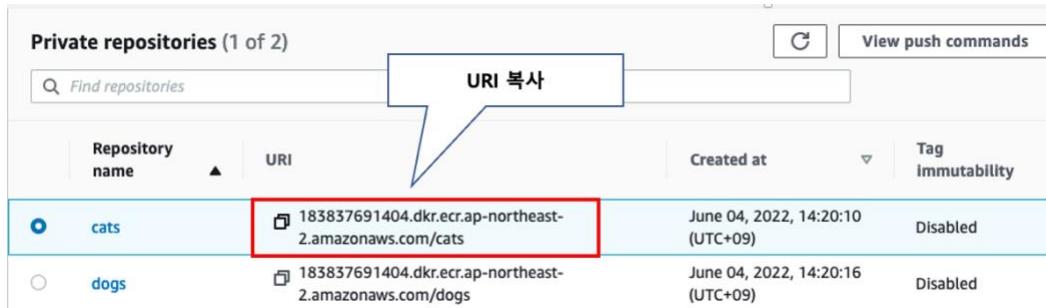
0	80	<input type="button" value="tcp"/>
---	----	------------------------------------

6. 이후 하단의 **Create** 를 클릭해서 web task definition 을 완료합니다.

## (5-2-2) cats 태스크 정의

Cats 태스크 정의는 web 과 마찬가지로 EC2 타입을 선택합니다. 추가로 로그 라우터인 FireLens 를 활성화하여 컨테이너 로그를 Amazon CloudWatch Logs 로 전송하도록 설정합니다. cats 는 web 이나 dogs 와 달리 AWS Firelens 설정이 포함되므로 비교적 복잡하므로 JSON 템플릿을 이용해 진행합니다.

1. [Amazon ECS](#) 로 이동 후 Task definition 로 이동하여 **Create new Task Definition** 을 클릭합니다.
2. **Select launch type compatibility** 에서는 **EC2** 를 선택합니다.
3. 스크롤을 내려 가장 아래쪽 **Configure via JSON** 을 클릭합니다. 혹은 아래 링크를 통해 다운로드 받으신 후 4 번 과정을 진행합니다.
4. json 다운로드 링크: <https://amz.run/5gcC>
5. (중요) 기준 내용을 지우고 아래 표에 있는 내용을 복사해서 붙여넣습니다. 단, 빨간색으로 표기된 부분 중 리전이 제대로 되어있는지 확인하시기 바랍니다. 그리고 아래쪽 **image** 부분은 반드시 본인의 ECR 에 있는 cats 의 **Image URI** 를 사용합니다. 표에서 빨간색으로 표기된 **image** 부분의 URI 를 지우고 본인의 URI 를 복사해서 대체합니다.



Private repositories (1 of 2)			
Repository name	URI	Created at	Tag immutability
cats	183837691404.dkr.ecr.ap-northeast-2.amazonaws.com/cats	June 04, 2022, 14:20:10 (UTC+09)	Disabled
dogs	183837691404.dkr.ecr.ap-northeast-2.amazonaws.com/dogs	June 04, 2022, 14:20:16 (UTC+09)	Disabled

```
{
  "requiresCompatibilities": [
    "EC2"
  ],
  "inferenceAccelerators": [],
  "containerDefinitions": [
    {
      "dnsSearchDomains": null,
      "environmentFiles": null,
      "entryPoint": null,
      "portMappings": [],
      "command": null,
      "linuxParameters": null,
      "cpu": 0,
      "environment": [],
      "resourceRequirements": null,
      "ulimits": null,
      "dnsServers": null,
      "mountPoints": [],
      "workingDirectory": null
    }
  ]
}
```

```

"secrets": null,
"dockerSecurityOptions": null,
"memory": 128,
"memoryReservation": null,
"volumesFrom": [],
"stopTimeout": null,
"image": "906394416424.dkr.ecr.ap-northeast-2.amazonaws.com/aws-for-fluent-bit:latest",
"startTimeout": null,
"firelensConfiguration": {
    "type": "fluentbit",
    "options": null
},
"logConfiguration": {
    "logDriver": "awslogs",
    "options": {
        "awslogs-group": "firelens-container",
        "awslogs-region": "ap-northeast-2",
        "awslogs-create-group": "true",
        "awslogs-stream-prefix": "firelens"
    }
},
"dependsOn": null,
"disableNetworking": null,
"interactive": null,
"healthCheck": null,
"essential": true,
"links": null,
"hostname": null,
"extraHosts": null,
"pseudoTerminal": null,
"user": "0",
"readonlyRootFilesystem": null,
"dockerLabels": null,
"systemControls": null,
"privileged": null,
"name": "log_router"
},
{
    "dnsSearchDomains": null,
    "environmentFiles": null,
    "logConfiguration": {
        "logDriver": "awsfirelens",
        "secretOptions": null,
        "options": {
            "log_group_name": "ecs-demogo-log",
            "auto_create_group": "true",
            "log_stream_prefix": "from-fluent-bit",
            ""region": "ap-northeast-2",
            "Name": "cloudwatch"
        }
    },
    "entryPoint": null,
    "portMappings": [
        {
            "hostPort": 0,
            "protocol": "tcp",
            "containerPort": 80
        }
    ]
}

```

```

        }
    ],
    "command": null,
    "linuxParameters": null,
    "cpu": 0,
    "environment": [],
    "resourceRequirements": null,
    "ulimits": null,
    "dnsServers": null,
    "mountPoints": [],
    "workingDirectory": null,
    "secrets": null,
    "dockerSecurityOptions": null,
    "memory": 128,
    "memoryReservation": null,
    "volumesFrom": [],
    "stopTimeout": null,
    "image": "183837691404.dkr.ecr.ap-northeast-2.amazonaws.com/cats",
    "startTimeout": null,
    "firelensConfiguration": null,
    "dependsOn": null,
    "disableNetworking": null,
    "interactive": null,
    "healthCheck": null,
    "essential": true,
    "links": null,
    "hostname": null,
    "extraHosts": null,
    "pseudoTerminal": null,
    "user": null,
    "readonlyRootFilesystem": null,
    "dockerLabels": null,
    "systemControls": null,
    "privileged": null,
    "name": "cats"
}
],
"volumes": [],
"networkMode": null,
"memory": null,
"cpu": null,
"placementConstraints": [],
"tags": [],
"family": "catsdef"
}

```

6. 이 내용을 붙여넣은 후 **Save** 후 **Create** 을 해서 cats 의 task definition 을 마무리합니다.

### (5-2-3) dogs 태스크 정의

Dogs 는 web, cats 와 달리 Fargate 시작 유형(launch type)을 사용하기 때문에 몇몇 설정 옵션에서 차이가 있습니다.  
예를 들어, Amazon ECS 의 Fargate 태스크 정의(task definitions)는 태스크 수준에서 CPU 와 메모리를 지정해야

합니다. Fargate 태스크는 awsvpc 네트워크 모드를 지정해야 하고 각 태스크별 ENI(Elastic Network Interface)를 부여합니다.

이번 태스크 정의는 console에서 직접 진행합니다.

1. [Amazon ECS](#)로 이동 후 Task definition로 이동하여 **Create new Task Definition**을 클릭합니다.
2. **Select launch type compatibility**에서는 **Fargate**를 선택합니다.
3. Task Definition Name은 **dogsdef**를 입력하고 Task size에서 Task memory는 **0.5GB**, Task CPU는 **0.25vCPU**를 선택합니다.

#### Configure task and container definitions

A task definition specifies which containers are included in your task and how they interact with each other. You can also specify data volumes for your containers to use. [Learn more](#)



Task definition name\* **dogsdef**

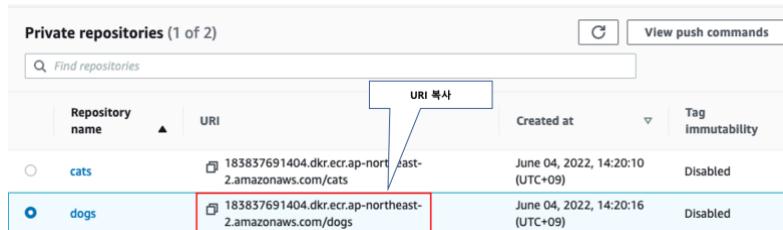
Requires compatibilities\* FARGATE

**Task size**

The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate launch type and is optional for the EC2 or External launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.

Task memory (GB)	0.5GB
The valid memory range for 0.25 vCPU is: 0.5GB - 2GB.	
Task CPU (vCPU)	0.25 vCPU
The valid CPU for 0.5 GB memory is: 0.25 vCPU	

4. 이후 스크롤을 내려 Container Definitions에서 dogs 컨테이너를 추가하기 위해 **Add container**를 클릭합니다.
5. Container name은 dogs, Image는 cats 때와 마찬가지로 ECR에서 본인의 **Image URI**를 복사해서 붙여넣습니다.
  - a. Container name: **dogs**
  - b. Image(아래 그림 참고): **본인의 dogs latest image URI**
  - c. Memory Limits: Soft limit **128**
  - d. Port mappings: Container port **80**



Private repositories (1 of 2)

Repository name	URI	Created at	Tag immutability
cats	183837691404.dkr.ecr.ap-northeast-2.amazonaws.com/cats	June 04, 2022, 14:20:10 (UTC+09)	Disabled
<b>dogs</b>	<b>183837691404.dkr.ecr.ap-northeast-2.amazonaws.com/dogs</b>	June 04, 2022, 14:20:16 (UTC+09)	Disabled

6. 최종 설정은 다음과 같습니다.

Standard

Container name\* dogs

Image\* 183837691404.dkr.ecr-northeast-2.amazonaws.com/dogs

Private repository authentication\*

Memory Limits (MiB) Soft limit 128 Add Hard limit

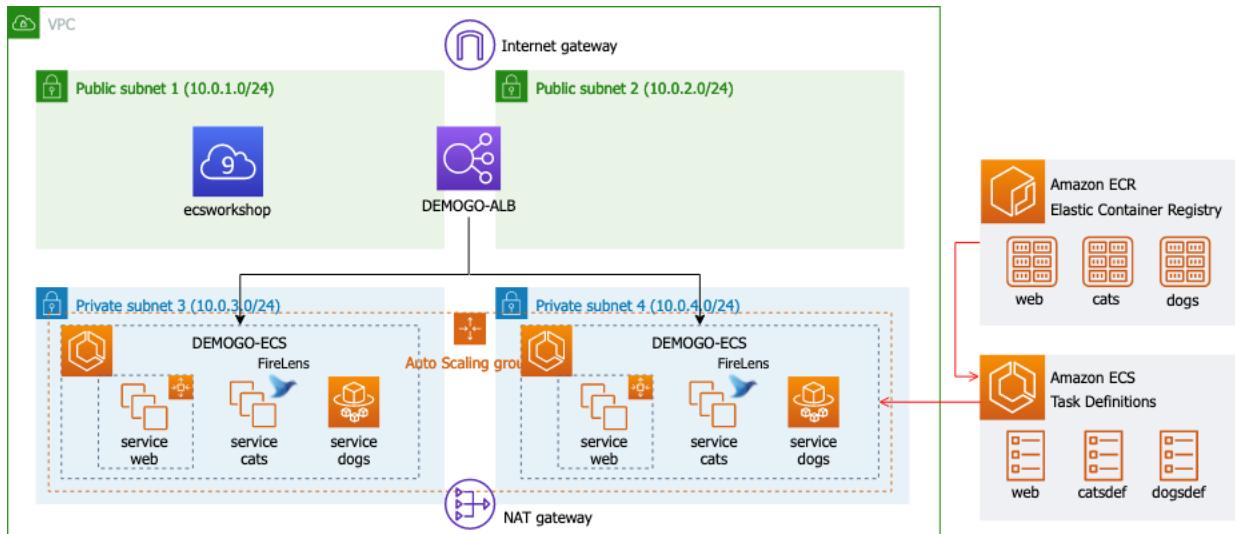
Port mappings Container port 80 Protocol tcp Add port mapping

7. 이후 **Add** 버튼을 클릭해서 Container 설정을 완료하고 **Create**를 클릭해서 Task Definition 을 마무리합니다.

## (5-2) ECS 서비스

Amazon ECS를 사용하여 Amazon ECS 클러스터에서 지정된 수의 작업 정의 인스턴스를 동시에 실행하고 관리할 수 있습니다. 이를 서비스라고 합니다. 어떤 이유로 작업이 실패 또는 중지되는 경우 Amazon ECS 서비스 스케줄러가 작업 정의의 다른 인스턴스를 시작하여 이를 대체하고 사용되는 일정 전략에 따라 서비스의 원하는 작업 수를 유지합니다.

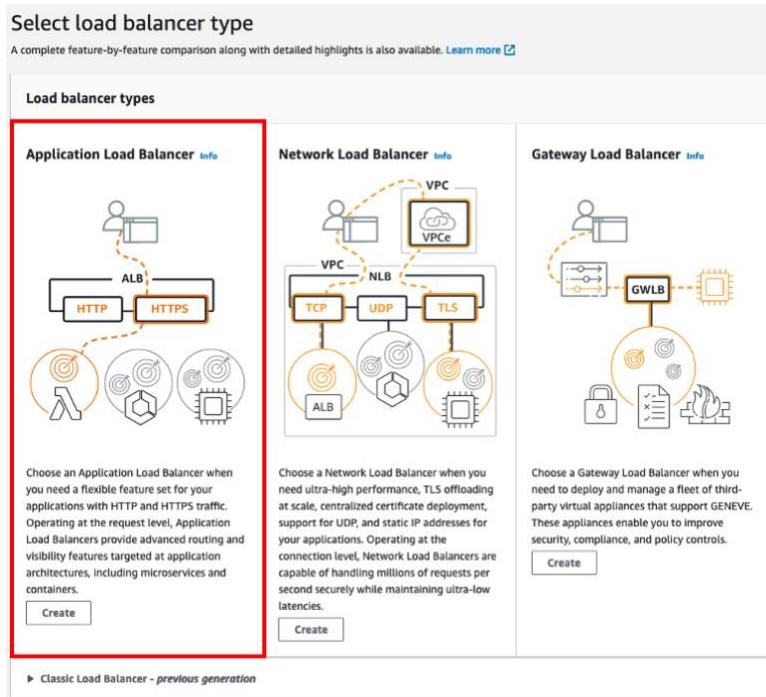
서비스에서 원하는 작업 수를 유지하는 이외에 선택적으로 로드 밸런서를 통해 서비스를 실행할 수 있습니다. 로드 밸런서는 서비스와 연결된 작업 간에 트래픽을 분산합니다. 이번 단계에서는 앞 챕터에서 생성한 작업 정의(Task Definitions)를 참조하는 3 가지 서비스 web, cats, dogs를 생성합니다.



### (5-2-1) ALB 생성

간단한 웹 서비스인 cats and dogs 로 트래픽을 분배할 Application Load Balancer 를 생성합니다. 실습에서 언급하지 않은 설정은 기본값으로 남겨둡니다.

1. [EC2 Load Balancers](#) 로 이동합니다. (EC2 메뉴 하단에서 직접 접근 가능)
2. **Create Load Balancer** 를 클릭하고 **Application Load Balancer** 를 선택합니다.



3. Load balancer name 은 **demogo-alb** 로 설정합니다. 네트워크는 CloudFormation 을 통해 만든 **DemoGoECSVPC** 를 선택하고 ALB 가 연결될 Public Subnet 2 개를 선택합니다. Security Group 은 **ecs-demogo-ALBSG** 를 선택하고 기존 default 는 선택 해제합니다.
  - a. Load balancer name: **demogo-alb**
  - b. Scheme: **Internet-facing**
  - c. IP address type: **IPv4**
  - d. VPC: **DemoGoECSVPC (10.0.0.0/16)**
  - e. Mappings: **ap-northeast-2a** 와 **ap-northeast-2b** 를 모두 체크
  - f. Subnet: **PublicSubnet1, 2** 를 선택
  - g. Security groups: 검색창에서 **ecs-demogo-ALBSG** 를 찾아 선택한 뒤 default 보안그룹은 선택 해제

**Network mapping** [Info](#)

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

**VPC** [Info](#)

Select the virtual private cloud (VPC) for your targets. Only VPCs with an internet gateway are enabled for selection. The selected VPC cannot be changed after confirming the VPC for your targets, view your target groups [\[?\]](#)

DemoGoECSVPC  
vpc-088fd63a7fe5f0bf  
IPv4: 10.0.0.0/16

**Mappings** [Info](#)

Select at least one Availability Zone and one subnet for each zone. We recommend selecting at least two Availability Zones. The load balancer will route traffic to Availability Zones. Zones that are not supported by the load balancer or VPC cannot be selected. Subnets can be added, but not removed, once a load balancer is created.

<input checked="" type="checkbox"/> ap-northeast-2a	
Subnet	
subnet-02da2b26d576acb72	PublicSubnet1
IPv4 settings	
Assigned by AWS	
<input checked="" type="checkbox"/> ap-northeast-2b	
Subnet	
subnet-0066355438045ce1	PublicSubnet2
IPv4 settings	
Assigned by AWS	

- 이어서 Listeners and routing 부분입니다. 우선 **target group** 을 만듭니다. 아래 **Create target group** 을 클릭하면 새로운 탭이 로딩됩니다.

**Listeners and routing** [Info](#)

A listener is a process that checks for connection requests, using the protocol and port you configure. Traffic received by the listener is then routed per your specification. You can specify multiple rules and multiple certificates per listener after the load balancer is created.

▼ Listener HTTP:80

Protocol	Port	Default action
HTTP	: 80	<a href="#">Info</a> Forward to: Select a target group <a href="#">[?]</a>
<a href="#">Create target group</a> <a href="#">[?]</a>		

- Target group 만드는 과정에서 우선 type 은 Instances 로 두고 **Target group name** 은 **web** 으로 설정합니다. 나머지 설정은 그대로 두고 **Next** 를 클릭합니다.

**Basic configuration**

Settings in this section cannot be changed after the target group is created.

## Choose a target type

 Instances

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.

 IP addresses

- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

 Lambda function

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

 Application Load Balancer

- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

## Target group name

web

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

## Protocol

## Port

HTTP : 80

## VPC

Select the VPC with the instances that you want to include in the target group.

DemoGoECSVPC

vpc-088fd63a7fe55f0bf

IPv4: 10.0.0.0/16

## Protocol version

 HTTP1

Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

 HTTP2

Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.

 gRPC

Send requests to targets using gRPC. Supported when the request protocol is gRPC.

6. Register targets 화면에서 instance 를 선택하지 않고 **Create target group** 을 클릭합니다.

7. 다시 Load balancer 를 생성하던 화면 (기준 탭)으로 돌아가서 web 을 선택합니다.

▼ Listener HTTP:80

Protocol	Port	Default action	Info
HTTP	: 80	Forward to	web Target type: Instance, IPv4
1-65535			
<a href="#">Create target group</a>			

8. 이후 **Create load balance** 클릭 후 설정을 완료합니다.**(5-2-2) web 서비스 생성**

- Amazon ECS로 이동하여 DEMOGO-ECS 클러스터를 선택합니다. Web 서비스를 생성하기 위해 Services 탭에서 Create을 클릭합니다. 별도 설명이 없는 설정은 기본값으로 남겨둡니다.

### Cluster : DEMOGO-ECS

Get a detailed view of the resources on your cluster.

Cluster ARN: arn:aws:ecs:ap-northeast-2:183837691404:cluster/DEMOGO-ECS

Status: ACTIVE

Registered container instances: 2

Pending tasks count: 0 Fargate, 0 EC2, 0 External

Running tasks count: 0 Fargate, 0 EC2, 0 External

Active service count: 0 Fargate, 0 EC2, 0 External

Draining service count: 0 Fargate, 0 EC2, 0 External

Services Tasks ECS Instances Metrics Scheduled Tasks Tags Capacity Providers

Create Update Delete Actions ▾

Filter in this page Launch type ALL Service type ALL

Service Name	Status	Service type	Task Definition
			No results

- Configure service 화면에서는 launch type, name, task 수 등을 설정합니다. 아래와 같이 선택 후 Next를 클릭합니다.
  - Launch type: EC2
  - Task Definition Family: web
  - Revision: 1 (latest) (Revision 값이 여러개라면 'latest'를 선택)
  - Service name: web
  - Number of tasks: 2
  - Task Placement: AZ Balanced Spread

Configure service

A service lets you specify how many copies of your task definition to run and maintain in a cluster. You can optionally use an Elastic Load Balancing load balancer to distribute incoming traffic to containers in your service. Amazon ECS maintains that number of tasks and coordinates task scheduling with the load balancer. You can also optionally use Service Auto Scaling to adjust the number of tasks in your service.

Launch type  FARGATE  EC2  EXTERNAL

Switch to capacity provider strategy

**Task Definition**

Family: web	Enter a value
Revision: 1 (latest)	
Cluster: DEMOGO-ECS	
Service name: web	
Service type*	<input checked="" type="radio"/> REPLICA <input type="radio"/> DAEMON
Number of tasks	2

Minimum healthy percent: 100

Maximum percent: 200

Deployment circuit breaker: Disabled

3. 다음 Configure network 에서는 **Application Load Balancer** 를 선택하고 IAM role 까지 설정 후 **Add to load balancer** 를 클릭합니다.

a. Load balancer type: **Application Load Balancer**

b. Service IAM role: **AWSServiceRoleForECS**

c. Load balancer name: **demogo-alb**

**Load balancing**

An Elastic Load Balancing load balancer distributes incoming traffic across the tasks running in your service. Choose an existing load balancer, or create a new one in the [Amazon EC2 console](#).

Load balancer type\*

None  
Your service will not use a load balancer.

Application Load Balancer  
Allows containers to use dynamic host port mapping (multiple tasks allowed per container instance). Multiple services can use the same listener port on a single load balancer with rule-based routing and paths.

Network Load Balancer  
A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. After the load balancer receives a request, it selects a target from the target group for the default rule using a flow hash routing algorithm.

Classic Load Balancer  
Requires static host port mappings (only one task allowed per container instance); rule-based routing and paths are not supported.

Service IAM role **AWSServiceRoleForECS**

Load balancer name **demogo-alb**

**Container to load balance**

Container name : port **web:80**

**Add to load balancer**

4. 이후 새로 로딩되는 화면에서는 아래와 같이 선택합니다.

a. Production listener port: 드롭다운에서 **80:HTTP** 를 선택

b. Target group name: 드롭다운에서 **web** 을 선택

**Container to load balance**

**web : 80**

Production listener port\* **80:HTTP**

Production listener protocol\* **HTTP**

Target group name **web**

Target group protocol **HTTP**

Target type **instance**

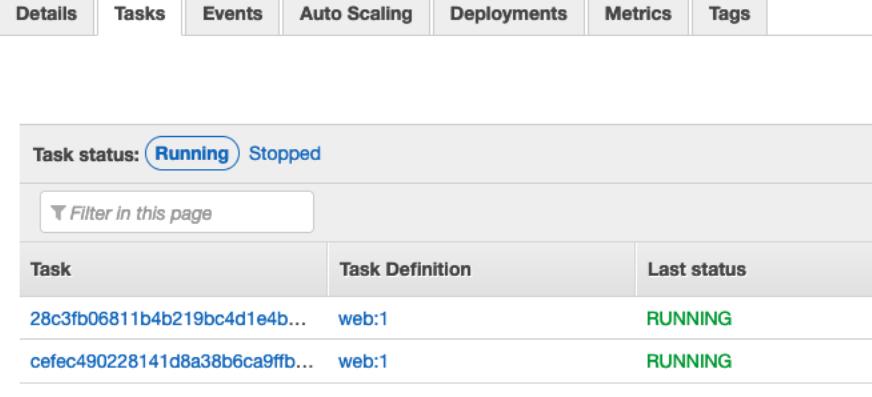
Path pattern **/**

Evaluation order **default**

Health check path **/**

Additional health check options can be configured in the ELB console after you create your service.

5. 다음 단계에서 Set Auto Scaling 은 그대로 두고 다음으로 넘어갑니다. Auto Scaling 은 별도 과정을 통해 진행할 예정입니다. **Review** 화면에서 다시 살펴보고 **Create Service** 을 클릭합니다.
6. 잠시 기다리면 아래와 같이 Tasks 탭에서 EC2 2 개가 실행된 것을 확인할 수 있습니다.



The screenshot shows the AWS CloudWatch Tasks page. At the top, there are tabs: Details, Tasks (which is selected and highlighted in orange), Events, Auto Scaling, Deployments, Metrics, and Tags. Below the tabs, a header bar displays 'Task status: Running Stopped'. A 'Filter in this page' button is available. The main table has columns: Task, Task Definition, and Last status. There are two rows of data:

Task	Task Definition	Last status
28c3fb06811b4b219bc4d1e4b...	web:1	RUNNING
cefec490228141d8a38b6ca9ffb...	web:1	RUNNING

### (5-2-3) cats 서비스 생성

1. web 서비스 생성과 마찬가지로 DEMOGO-ECS 클러스터의 **Services** 탭에서 **Create** 를 클릭합니다.
2. Configure service 화면에서는 launch type, name, task 수 등을 설정합니다. 아래와 같이 선택 후 **Next** 를 클릭합니다.
  - a. Launch type: **EC2**
  - b. Task Definition Family: **catsdef**
  - c. Revision: **1 (latest)** (Revision 값이 여러개라면 'latest'를 선택)
  - d. Service name: **cats**
  - e. Number of tasks: **2**
  - f. Task Placement: **AZ Balanced Spread**
3. 다음 Configure network 에서는 이전과 같이 **Application Load Balancer** 를 선택합니다.
  - a. Load balancer type: **Application Load Balancer**
  - b. Service IAM role: **AWSServiceRoleForECS**
  - c. Load balancer name: **demogo-alb**

## Load balancing

An Elastic Load Balancing load balancer distributes incoming traffic across the tasks running in your service. Choose an existing load balancer, or create a new one in the [Amazon EC2 console](#).

Load balancer type\*

None  
Your service will not use a load balancer.

Application Load Balancer  
Allows containers to use dynamic host port mapping (multiple tasks allowed per container instance). Multiple services can use the same listener port on a single load balancer with rule-based routing and paths.

Network Load Balancer  
A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. After the load balancer receives a request, it selects a target from the target group for the default rule using a flow hash routing algorithm.

Classic Load Balancer  
Requires static host port mappings (only one task allowed per container instance); rule-based routing and paths are not supported.

Service IAM role

Load balancer name

Container to load balance

Container name : port

4. Container to load balance 에서는 아래와 같이 설정합니다. 마지막 Health check path 는 default 인 /cats 가 아니라 /cats/로 변경하시기 바랍니다. 설정 후 Next Step 을 클릭합니다.

- a. Production listener port: **80:HTTP**
- b. Target group name: **create new** 를 선택하고 **cats** 를 입력
- c. path pattern: **/cats\***, Evaluation order: **1**
- d. Health check path: **/cats/**

Container to load balance

cats : 80

[Remove X](#)

Production listener port\* 80:HTTP

Production listener protocol\* HTTP

Target group name  cats

Target group protocol HTTP

Target type instance

Path pattern /cats\* Evaluation order 1

Path pattern: The first path pattern for a listener is the default path (/), which accepts all traffic that does not match another rule. You can later add additional patterns and priority values to this listener for other services.

Evaluation order: Rules are evaluated in priority order, from the lowest value to the highest value. Once a path pattern rule is matched, all other rules are ignored. You can route traffic from this listener to multiple services by creating a path for each service.

Existing paths in use on this listener

The path must include all the possible paths that your application uses, for example a service with the path /webapp1\* will receive traffic sent to /webapp1 and /webapp1/page.html on this listener. We recommend choosing unique paths, and a lower evaluation order enables you to route traffic between multiple conflicting paths.

Evaluation Order	Rule Path	Target Group
default	/	web

Health check path /cats/

Additional health check options can be configured in the ELB console after you create your service.

5. 다음 단계는 default 로 두고 최종 Review 후 서비스를 생성합니다.

#### (5-2-4) dogs 서비스 생성

1. cats 서비스 생성과 마찬가지로 DEMOGO-ECS 클러스터의 **Services** 탭에서 **Create** 를 클릭합니다.
2. Configure service 화면에서는 launch type, name, task 수 등을 설정합니다. 이번에는 EC2 가 아닌 Fargate 로 설정합니다. 아래와 같이 선택 후 **Next** 를 클릭합니다.
  - a. Launch type: **Fargate**
  - b. Operating system family: **Linux**
  - c. Task Definition Family: **dogsdef**
  - d. Revision: **1 (latest)** (Revision 값이 여러개라면 'latest'를 선택)
  - e. Service name: **dogs**
  - f. Number of tasks: **2**
3. 다음 Configure network 에서, Cluster VPC 는 **DemoGoECSVPC** 를 선택합니다. 그리고 Subnet 은 **Private 1,2** 를 선택합니다.

## Configure network

## VPC and security groups

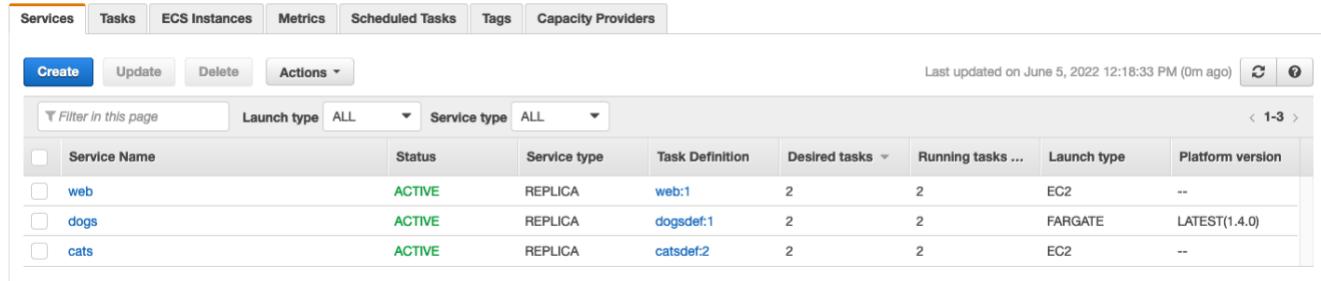
VPC and security groups are configurable when your task definition uses the awsvpc network mode.

4. 아래 Load balancing 에서는 **Application Load Balancer** 를 선택하고 **Add to load balancer** 를 클릭합니다.
5. **Container to load balance** 에서는 아래와 같이 설정합니다. 마지막 **Health check path** 는 default 인 /dogs 가 아니라 **/dogs/**로 변경하시기 바랍니다. 설정 후 Next Step 을 클릭합니다.
  - a. Production listener port: **80:HTTP**
  - b. Target group name: create new 를 선택하고 **dogs** 를 입력
  - c. path pattern: **/dogs\***, Evaluation order: **2**
  - d. Health check path: **/dogs/**

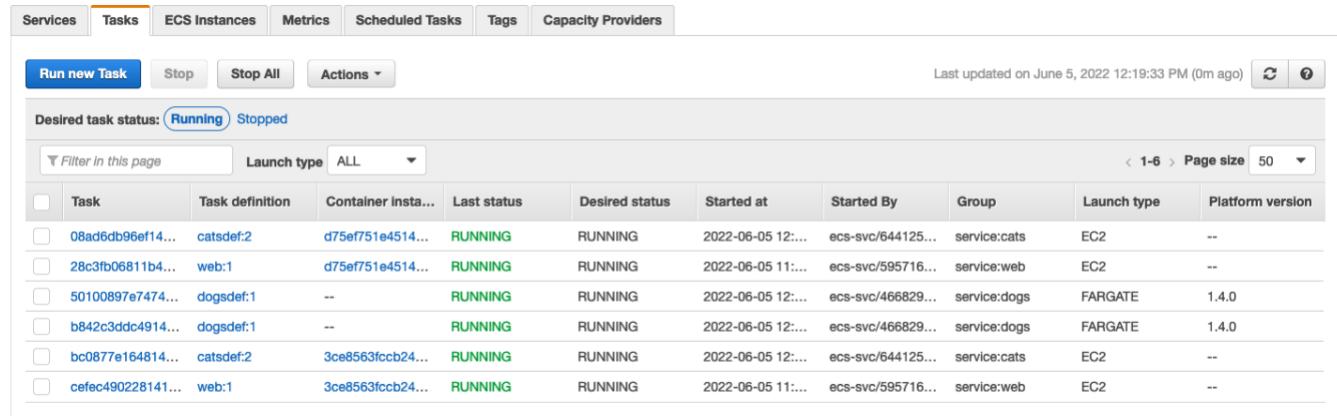
6. 다음 단계는 default 로 두고 최종 Review 후 서비스를 생성합니다

### (5-2-5) 서비스 확인

- 클러스터로 이동해서 Services 탭에서 각 서비스가 **Active** 인지 확인합니다. 그리고 Tasks 탭에서는 상태가 모두 **Running** 인지 확인합니다.

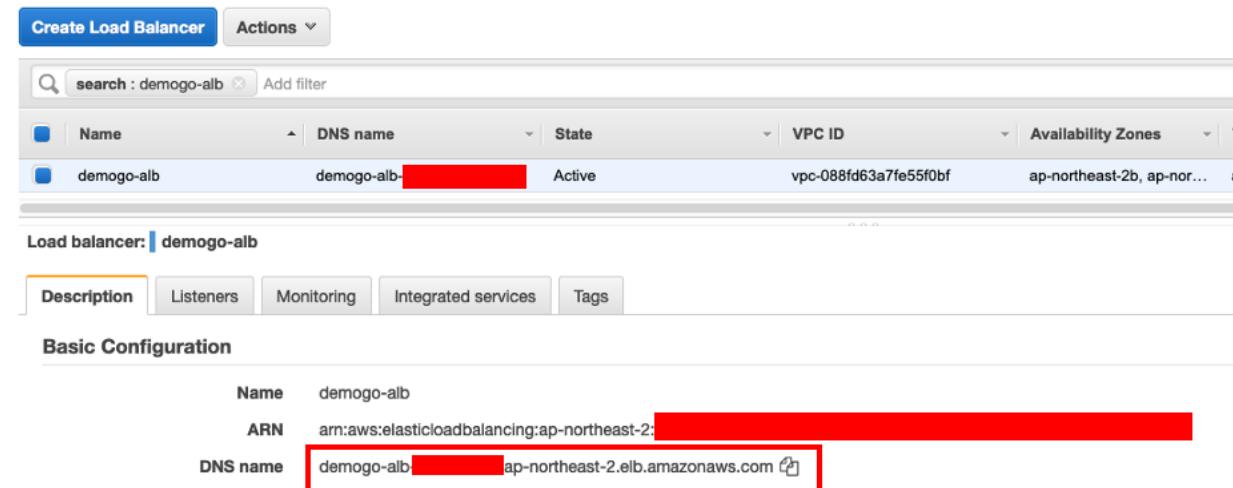


The screenshot shows the AWS CloudWatch Metrics console with the 'Metrics' tab selected. It displays metrics for the 'web' service, including CPU usage, memory usage, and task execution counts over time. The metrics are grouped by dimension, such as Task ID and Container Name.



The screenshot shows the AWS CloudWatch Metrics console with the 'Metrics' tab selected. It displays metrics for the 'dogs' service, including CPU usage, memory usage, and task execution counts over time. The metrics are grouped by dimension, such as Task ID and Container Name.

- [Amazon EC2 Load Balancers](#)로 이동합니다. demogo-alb 의 DNS Name 을 복사하여 웹브라우저에 붙여 넣으면 Web 메인 페이지로 이동합니다.



The screenshot shows the AWS Elastic Load Balancing console with the 'Load Balancers' tab selected. It lists a single load balancer named 'demogo-alb'. The details page for this load balancer is shown, including its name, ARN, and DNS name. The DNS name is highlighted with a red box.

3. 웹 페이지가 뜨면 cats 와 dogs 를 각각 클릭하고 새로고침 해보시기 바랍니다.

<http://demogo-alb-694381042.ap-northeast-2.elb.amazonaws.com/>



THIS IS WHAT LOVE LOOKS LIKE

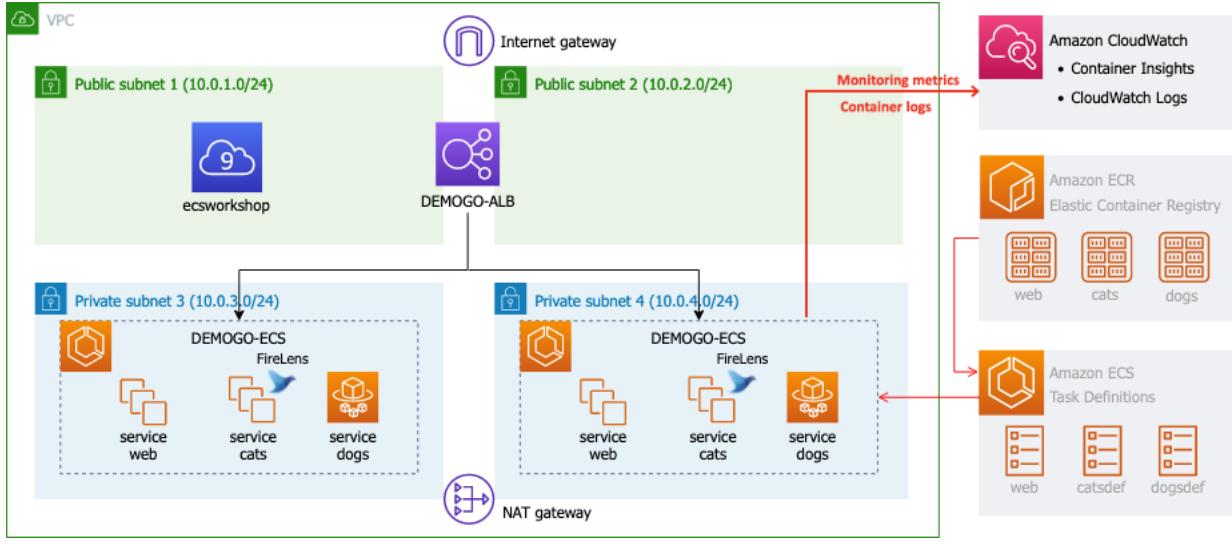


THIS IS WHAT HAPPINESS LOOKS LIKE



## 6. 모니터링

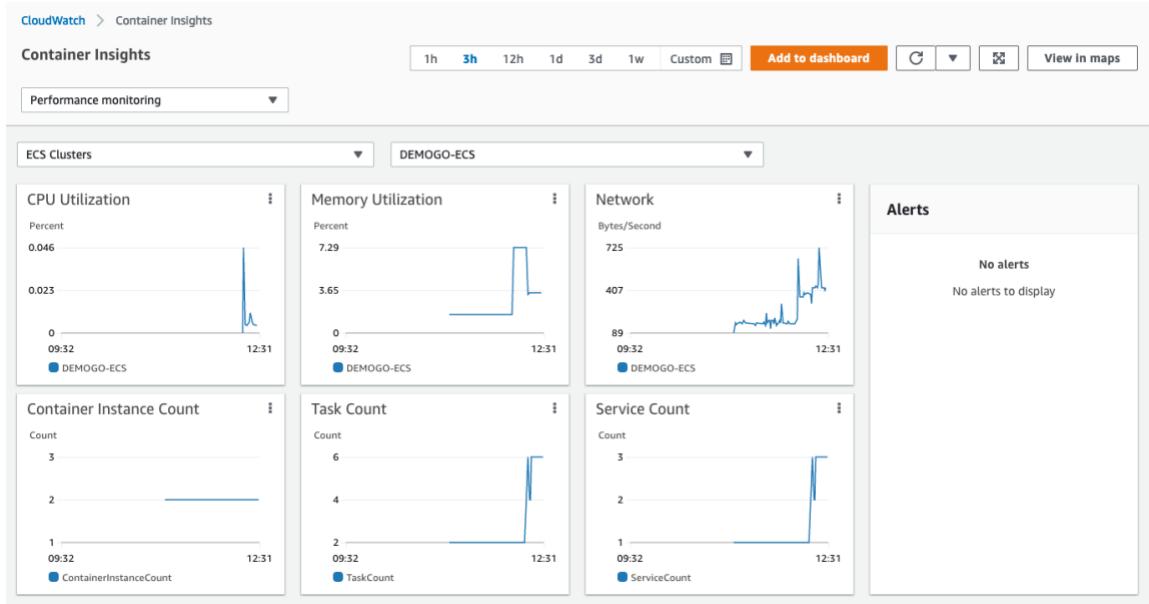
모니터링은 Amazon ECS 와 사용자 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 중요한 역할을 합니다.  
다중 지점 실패(multi-point failure)발생시 이를 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분으로부터  
모니터링 데이터를 수집해야 합니다.

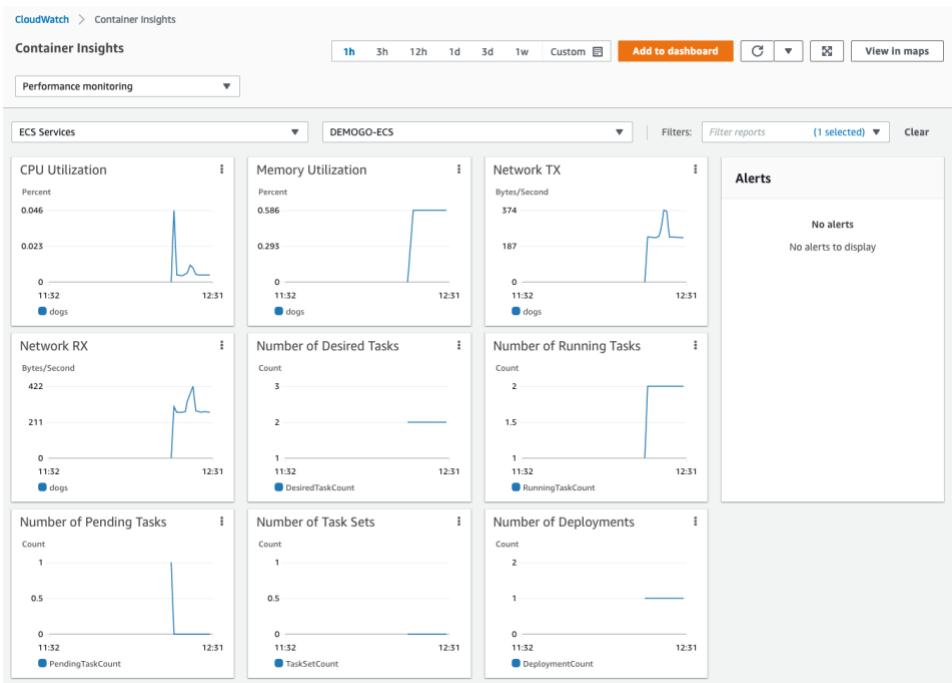


## (6-1) Amazon CloudWatch Container Insights

CloudWatch Container Insights 를 사용해 컨테이너식 애플리케이션 및 마이크로서비스의 지표 및 로그를 수집하고 집계하며 요약할 수 있습니다. 이 지표에는 CPU, 메모리, 디스크, 네트워크 같은 리소스 사용률이 포함되어 있습니다. 또한 Container Insights 는 컨테이너 재시작 오류 같은 진단 정보를 제공하여 문제를 격리하고 신속하게 해결할 수 있도록 도와줍니다.

1. [Amazon CloudWatch](#)로 이동합니다.
2. 왼쪽 메뉴 중에서 **Insights > Container Insights** 를 선택하면 **resource** 가 보입니다.
3. 직접 생성한 resource 를 클릭해서 들어가면 아래와 같이 상세 모니터링을 할 수 있습니다.



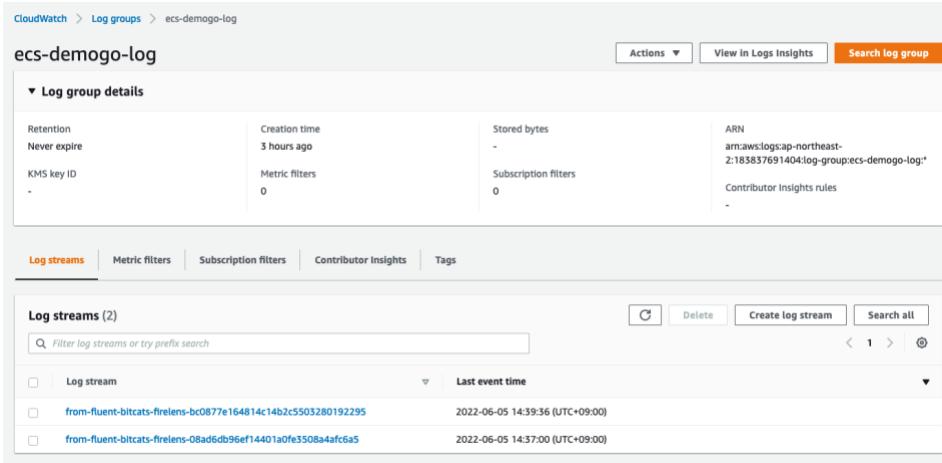


## (6-2) AWS FireLens 를 이용한 로그 라우팅

FireLens는 Amazon ECS 및 AWS Fargate 용 컨테이너 로그 라우터로, AWS의 서비스나 로그 분석 및 저장용 파트너 솔루션을 광범위하게 사용할 수 있는 확장성을 제공합니다. 이번 실습을 위해 앞 실습 ECS 클러스터에서 ecsInstanceRole에 CloudWatchLogsFullAccess를 부여하고, ECS 작업 정의에서 catsdef 작업 정의를 생성할 때 FireLens를 활성화한 것입니다.

Amazon ECS 용 FireLens를 사용하면 작업 정의의 파라미터를 사용하여 로그를 AWS 서비스 또는 AWS 파트너 네트워크(APN) 대상으로 라우팅하여 로그를 저장 및 분석할 수 있습니다. 본 실습에서는 AWS가 제공한 Fluent Bit FireLens를 이용해 AWS 서비스인 CloudWatch Logs로 로그를 전송합니다.

1. [CloudWatch Log groups](#)로 이동해서 **ecs-demogo-log**를 클릭하고 각 로그 스트림을 살펴봅니다.



2. 각 로그를 펼쳐서 더 상세한 정보를 찾을 수 있습니다. container\_id, ecs\_cluster, etcs\_task\_definition 등의 정보를 담고 있습니다.

```

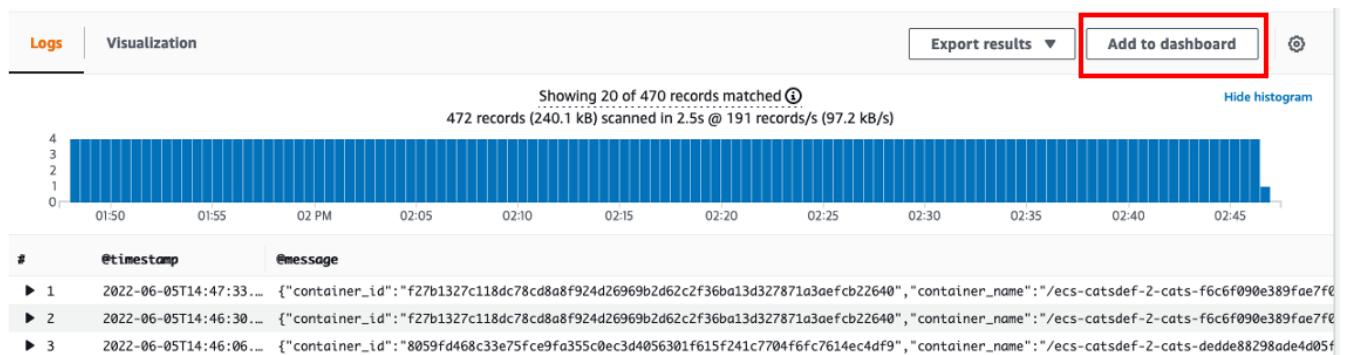
▶ Timestamp Message
▼ 2022-06-05T14:20:04.729+09:00
  There are older events to load. Load more.
  {"container_id": "8059fd468c33e75fce9fa355c0ec3d4056301f615f241c7704f6fc7614ec4df9",
   "container_name": "/ecs-catsdef-2-cats-dedde88298ade4d05f00",
   "ec2_instance_id": "i-0d173f8bed98973",
   "ecs_cluster": "DEMOGO-ECS",
   "ecs_task_arn": "arn:aws:ecs:ap-northeast-2:183837691404:task/DEMOGO-ECS/bc0877e164814c14b2c5503280192295",
   "ecs_task_definition": "catsdef:2",
   "log": "10.0.2.131 - - [05/Jun/2022:05:20:04 +0000] \"GET /cats/ HTTP/1.1\" 200 966 \"-\" \"ELB-HealthChecker/2.0\" \"-\"",
   "source": "stdout"
  }
  
```

3. CloudWatch Logs 가 제공하는 Log Insight 를 이용해 간단히 로그를 분석할 수도 있습니다. **ecs-demogo-log** 그룹을 선택하고, View in Logs Insight 버튼을 클릭하면 CloudWatch Logs Insight 로 리다이렉트됩니다.  
 4. 다음과 같은 샘플 쿼리가 보입니다.

```

fields @timestamp, @message
| sort @timestamp desc
| limit 20
  
```

- fields: 쿼리 결과에 표시할 필드를 지정합니다.
  - sort: 검색된 로그 이벤트를 정렬합니다. 오름차순(asc) 및 내림차순(desc)이 둘 다 지원됩니다.
  - limit: 쿼리에서 반환되는 로그 이벤트 수를 지정합니다.
5. 샘플 쿼리가 어떤 결과를 출력하는지 확인하기 위해 **Run query** 를 클릭합니다. 간단한 시각화와 함께 로그의 timestamp 와 message 를 내림차순으로 20 개 결과를 보여줍니다. 우측 상단의 **Add to dashboard** 버튼을 클릭하여 주요 쿼리문은 여러분들의 대시보드에 추가할 수도 있습니다.

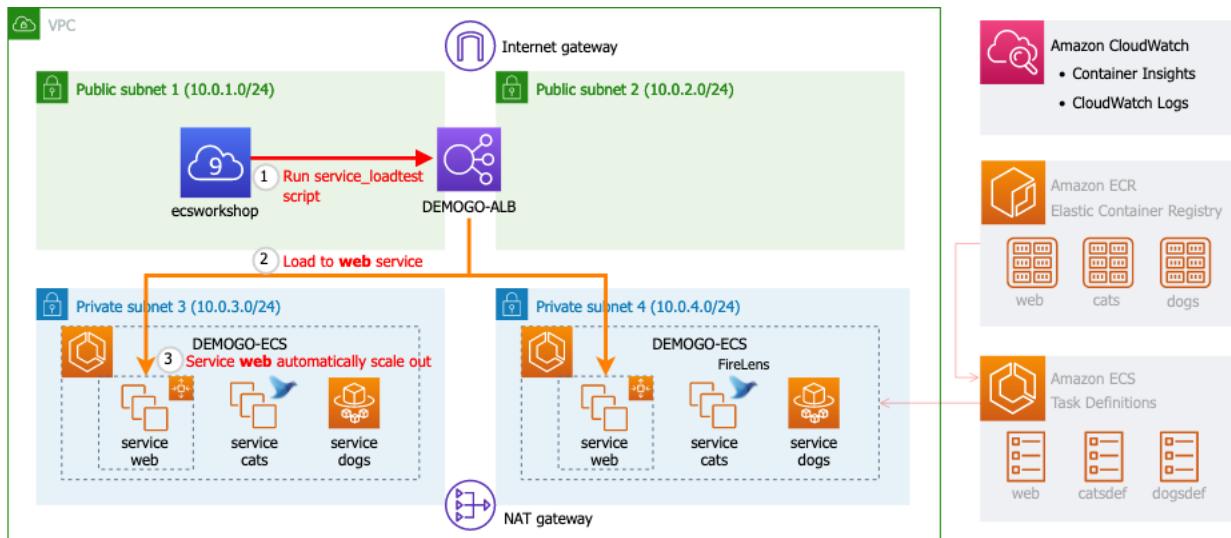


Log Insight 외에도 Amazon CloudWatch Logs 는 다른 AWS 서비스와 통합하여 다양하게 활용하실 수 있습니다. 예를 들어, CloudWatch Alarm 과 함께 로그의 특정 구문을 필터링하여 알람을 설정하거나, AWS Kinesis 를 이용해 로그 데이터를 실시간으로 처리할 수 있습니다.

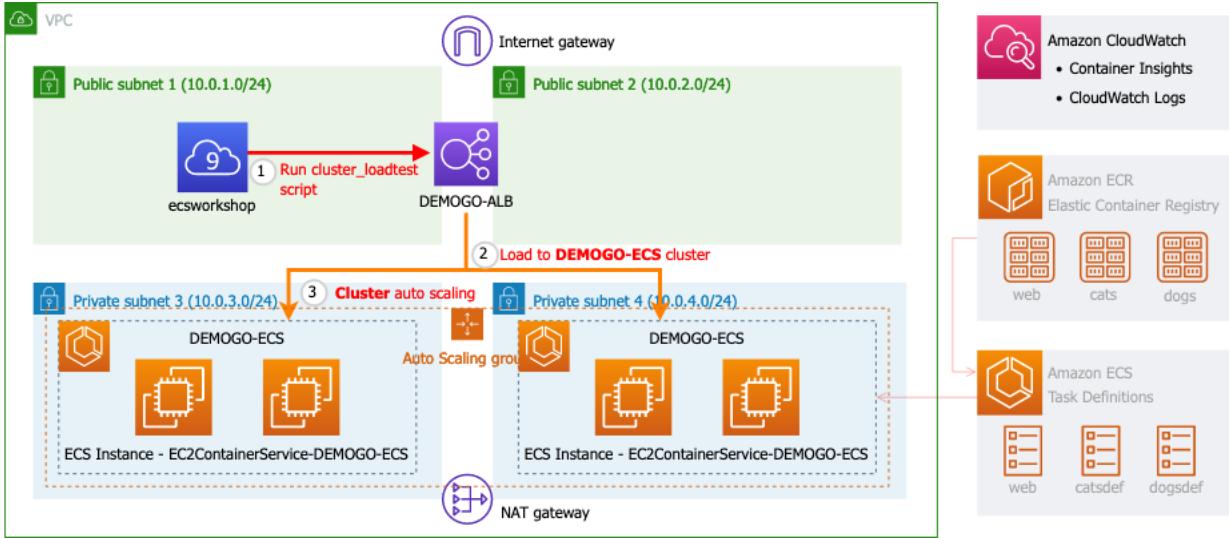
## 7. Auto Scaling

이전 실습 모니터링에서 살펴본 것처럼 Amazon ECS 는 서비스의 평균 CPU, 메모리 사용량 등을 CloudWatch 의 지표를 게시합니다. 이와 함께 다른 CloudWatch 지표를 사용하여 사용률에 맞추어 서비스를 자동으로 확장 또는 축소합니다. 더 나아가 Amazon ECS 클러스터 auto scaling 을 사용하면 클러스터의 모든 태스크 및 서비스의 리소스 수요에 맞게 필요에 따라 자동으로 인스턴스를 확장할 수 있습니다. 본 실습에서 서비스와 클러스터에 부하테스트를 수행하여 어떻게 두 가지 auto scaling 이 동작하는지 알아봅니다.

**Amazon ECS 서비스 auto scaling** - Amazon ECS 는 수요에 맞게 서비스를 스케일 아웃(태스크 수를 추가)하거나 사용량이 적은 때에는 스케일 인(태스크 수를 감소)하여 자동으로 조정할 수 있습니다.

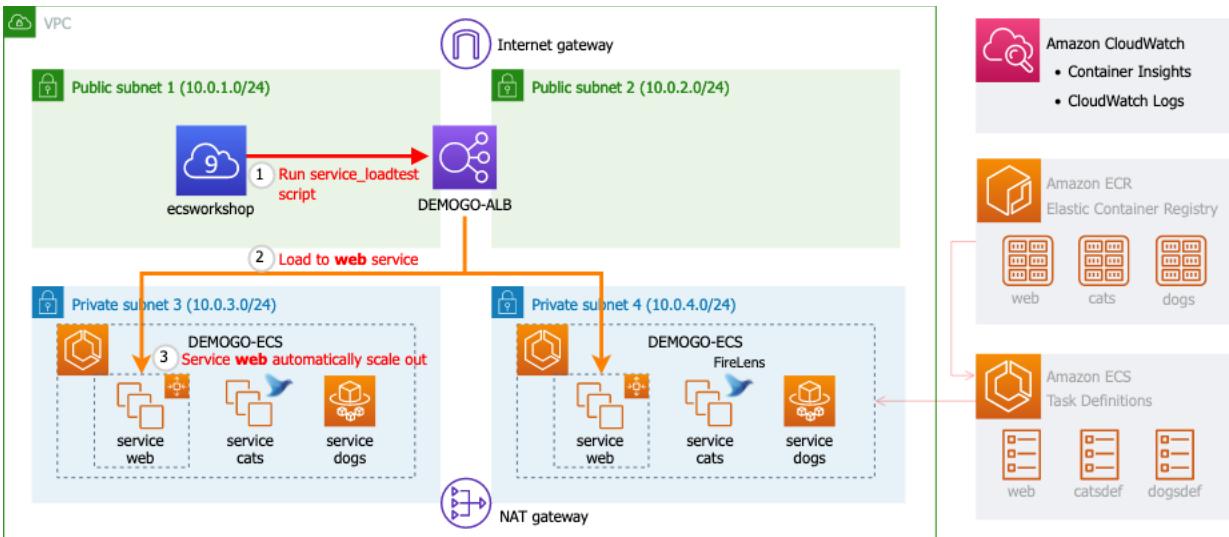


**Amazon ECS 클러스터 auto scaling** - ECS Cluster auto scaling 을 사용하면 EC2 에서 실행되는 ECS 클러스터가 클러스터의 모든 태스크 및 서비스의 리소스 수요에 맞게 필요에 따라 자동으로 확장할 수 있습니다. ECS Cluster Auto Scaling 은 ECS 에서 ECS capacity providers 를 사용하여 사용자를 대신해 Amazon EC2 Auto Scaling 그룹(ASG)을 관리합니다. 사용자는 ASG 의 관리형 확장을 활성화하고, ASG 에서 초과 용량을 예약하고, ASG 에서 인스턴스 종료를 관리하도록 capacity providers 를 구성할 수 있습니다.



## (7-1) 서비스 Auto Scaling

Amazon ECS는 수요에 맞게 서비스를 scale-out(태스크 수를 추가)하거나 사용량이 적은 때에는 scale-in(태스크 수를 감소)하여 자동으로 조정할 수 있습니다. 본 실습에서는 세 가지 서비스 중 web에 서비스 auto scaling을 설정하여 테스트합니다.



### (7-1-1) 서비스 auto scaling 구성

1. [Amazon ECS](#)의 DEMOGO-ECS 클러스터로 이동한 후 서비스 탭에서 web을 선택하고 Update를 클릭합니다.
2. Step 3. Set Auto Scaling 까지 넘어갑니다.
3. **Configure Service Auto Scaling to adjust your service's desired count**를 선택하고 아래와 같이 설정합니다.

- a. Minimum number of tasks: **2**
- b. Desired number of tasks: **2**
- c. Maximum number of tasks: **4**
- d. IAM role for Service Auto Scaling: **AWSServiceRoleForApplicationAutoScaling\_ECSService**

### Set Auto Scaling (optional)

Automatically adjust your service's desired count up and down within a specified range in response to CloudWatch alarms. You can modify your Service Auto Scaling configuration at any time to meet the needs of your application.

Service Auto Scaling     Do not adjust the service's desired count

Configure Service Auto Scaling to adjust your service's desired count

Minimum number of tasks	2
Automatic task scaling policies you set cannot reduce the number of tasks below this number.	
Desired number of tasks	2
Automatic task scaling policies you set cannot increase the number of tasks above this number.	
Maximum number of tasks	4
Automatic task scaling policies you set cannot increase the number of tasks above this number.	
IAM role for Service Auto Scaling	AWSServiceRoleForApplicationAut...

- 그리고 아래에 있는 **Add scaling policy**를 클릭합니다.

### Automatic task scaling policies

**Add scaling policy**

Policy name\*

No results

- 아래와 같이 설정합니다.

- a. Scaling policy type: **Target tracking**
- b. Policy name: **ALB-request-tracking**
- c. ECS service metric: **ALBRequestCountPerTarget**
- d. Target value: **4000**
- e. Scale-out cooldown period: **10** (본 실습에서는 빠른 결과 확인을 위해 의도적으로 낮은 임계치를 설정합니다.)

Add policy

Scaling policy type  Target tracking  Step scaling

Policy name\* ALB-request-tracking

ECS service metric\* ALBRequestCountPerTarget

12h

Target value\* 4000

Scale-out cooldown period 10 seconds between scaling actions

Scale-in cooldown period 300 seconds between scaling actions

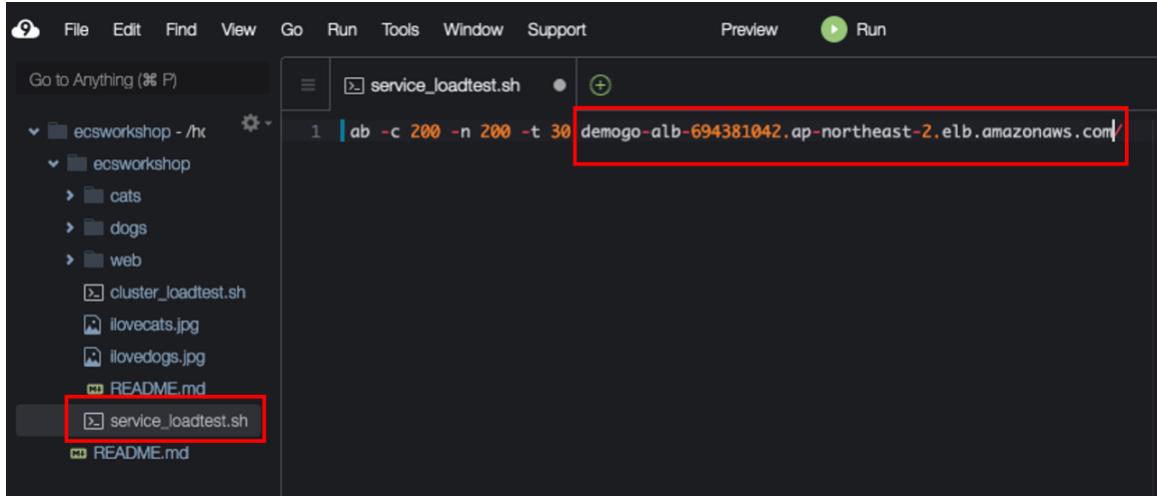
Disable scale-in

6. Save 후 Next Step 을 클릭하고 최종적으로 Update 를 클릭해서 설정을 마무리합니다.

### (7-1-2) 서비스 부하 테스트

Cloud9 워크스테이션에 미리 부하 테스트를 위한 스크립트 템플릿이 저장되어 있습니다. 이를 여러분들의 ALB URL로 수정한 뒤, 서비스 web으로 부하 테스트를 수행할 것입니다.

1. [AWS Cloud9](#) 으로 이동하고 **ecsworkshop** 을 선택한 후 **Open IDE** 를 클릭합니다.
2. ecsworkshop 폴더 아래 **service\_loadtest.sh** 파일을 더블 클릭합니다.
3. 스크립트 내의 DNS name 을 본인의 ALB DNS 로 변경합니다. [EC2 Load Balancers](#) 로 이동하여 demogo-alb 의 DNS 이름을 확인합니다. 변경 후에는 파일 저장합니다.



4. **service\_loadtest.sh** 스크립트를 실행하기 위해 권한을 부여한 후 부하 테스트를 수행합니다. 1 분에 한번씩 3~4 회 실행합니다. 다음 단계에서 모니터링합니다.

**chmod 755 ecsworkshop/service\_loadtest.sh**

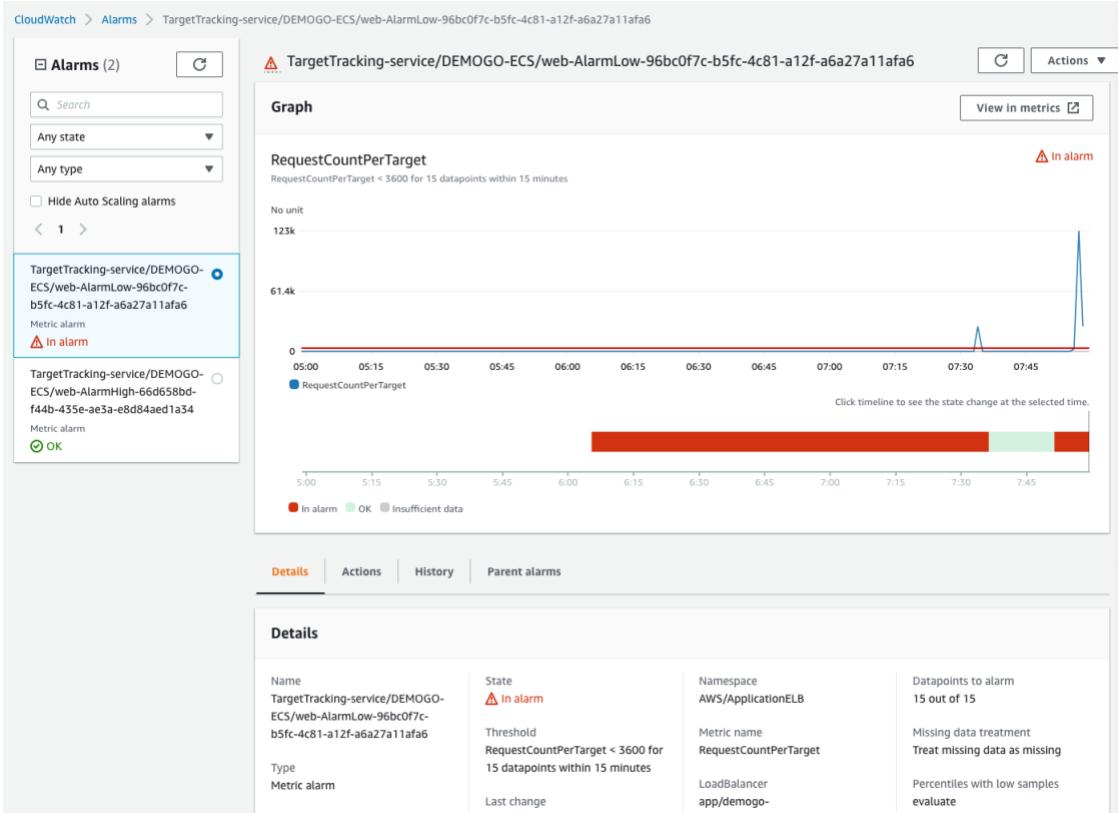
**./ecsworkshop/service\_loadtest.sh**

### (7-1-3) 결과 확인

1. 모니터링을 위해 [CloudWatch Alarms](#)로 이동합니다.
2. 아래와 같이 **State** 가 **In alarm** 으로 보입니다. 단, 초반에는 Insufficient data 로 보일 수 있습니다.

CloudWatch > Alarms					
Alarms (2)		<input type="checkbox"/> Hide Auto Scaling alarms		<input type="button" value="Clear selection"/>	<input type="button" value="Create composite alarm"/>
<input type="checkbox"/>	Name	State	Last state update	Conditions	Actions
<input type="checkbox"/>	TargetTracking-service/DEMOGO-ECS/web-AlarmLow-96bc0f7c-b5fc-4c81-a12f-a6a27a11afa6	<span style="color: red;">⚠ In alarm</span>	2022-06-05 15:05:28	RequestCountPerTarget < 3600 for 15 datapoints within 15 minutes	<span style="color: green;">Actions enabled</span>

3. Alarm 을 클릭하면 자세한 내용을 확인할 수 있습니다. **RequestCountPerTarget** 이 4000 을 초과한 상태로 3 분 이상 유지해야 합니다.



4. 약 3~4 분 뒤 web 서비스에서 Events 탭을 확인합니다. 아래와 같이 task 가 늘어나는 것을 확인할 수 있습니다.

Details	Tasks	Events	Auto Scaling	Deployments	Metrics	Tags
Last updated on June 5, 2022 5:02:21 PM (0m ago)						
<input type="button" value="Filter in this page"/>						
Event Id	Event Time	Message				
1e0857c2-169c-4ee4-b8ff-2c55ddb1af51	2022-06-05 17:02:10	service web has reached a steady state. +0900				
8dd290c9-b9a2-4674-8425-1cec1128588b	2022-06-05 17:01:52	service web registered 2 targets in target-group web +0900				
4b78e108-b904-475a-a213-5f1719b07e7e	2022-06-05 17:01:46	service web has started 2 tasks: task fc9609aafac34d0ead80f2d4b590ee4 task f221ab5018f54a308657fcdec98545e5. +0900				
dbaf1825-9108-49d2-ad7b-6039e1c89946	2022-06-05 17:01:42	Message: Successfully set desired count to 4. Change successfully fulfilled by ecs. Cause: monitor alarm TargetTracking-service/DEMOGO-ECS/web-AlarmHigh-66d658bd-f44b-435e-ae3a-e8d84aed1a34 in state ALARM triggered policy ALB-request-tracking +0900				
cefa8e27-206b-4ea8-93e4-d5c9f6e2450a	2022-06-05 11:06:33	service web has reached a steady state. +0900				
abcc39c0-a992-449c-a392-9898271ffb9d	2022-06-05 11:06:33	service web (deployment ecs-svc/5957168243543997581) deployment completed. +0900				
cbd6af0f-aafd-4ab3-bc6c-86b50548d37f	2022-06-05 11:06:14	service web registered 2 targets in target-group web +0900				
3193a90b-e5ad-4222-afbc-7e15fa208a80	2022-06-05 11:06:05	service web has started 2 tasks: task cefec490228141d8a38b8ca9ffba7c28 task 28c3fb06811b4b219bc4d1e4b0dd61e8. +0900				

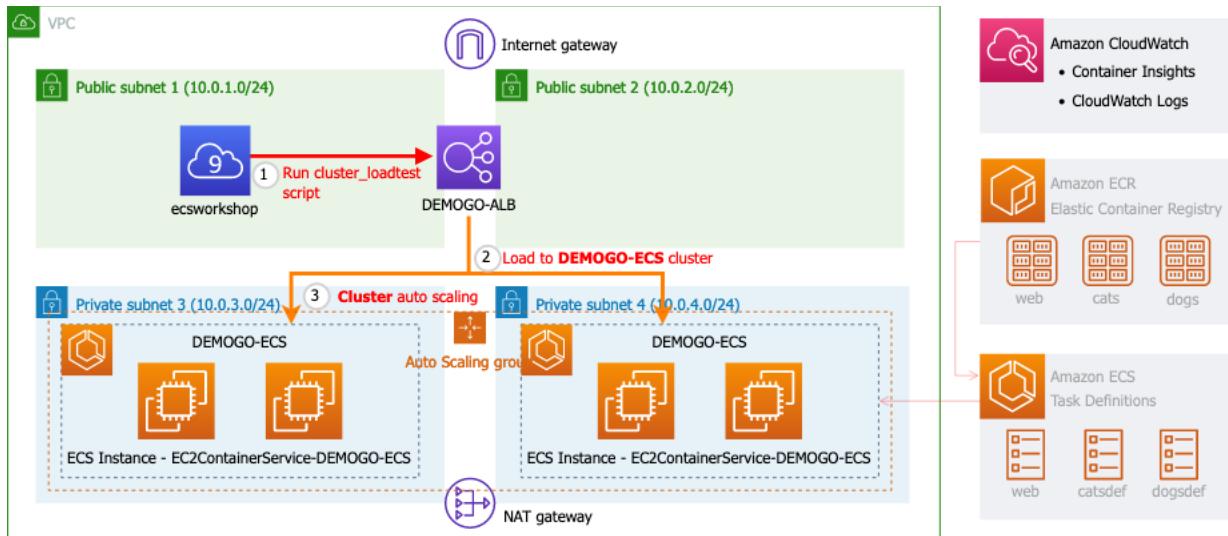
5. 그리고 Tasks 탭을 보면 tasks 수가 max 로 설정한 4 개로 늘어나 있습니다.

Details	Tasks	Events	Auto Scaling	Deployments	Metrics	Tags
Last updated on June 5, 2022 5:02:45 PM (0m ago)						
Task status: <span style="color: blue;">Running</span> Stopped						
<span style="float: left; margin-right: 10px;">Filter in this page</span> <span style="float: right;">Page size 50 ▾</span>						
Task	Task Definition	Last status	Desired status	Group	Launch type	
28c3fb06811b4b219bc4d1e4b...	web:1	RUNNING	RUNNING	service:web	EC2	
cefec490228141d8a38b6ca9ffb...	web:1	RUNNING	RUNNING	service:web	EC2	
f221ab5018f54a308657fcdec98...	web:1	RUNNING	RUNNING	service:web	EC2	
fc9609aaafac34d0eade80f2d4b5...	web:1	RUNNING	RUNNING	service:web	EC2	

## (7-2) 클러스터 Auto Scaling

ECS Cluster Auto Scaling 을 사용하면 ASG 의 확장 정책은 ECS Capacity Providers 를 통해 ECS 에서 관리됩니다. 사용자는 ASG 의 관리형 확장을 활성화하고, ASG 에서 초과 용량을 예약하고, ASG 에서 인스턴스 종료를 관리하도록 Capacity Providers 를 구성할 수 있습니다.

EC2 와 Fargate 모두에서 Capacity Providers 를 사용할 수 있습니다. EC2 를 사용하면 EC2 Auto Scaling 그룹(ASG)과 연결된 capacity providers 를 생성할 수 있습니다. 이를 통해 ASG 의 확장을 관리하여 작업을 실행하는 데 필요한 용량이 아직 사용할 수 없는 경우에도 요청되도록 할 수 있습니다. 작업과 서비스를 실행할 때 여러 capacity providers 로 작업(Task)과 서비스를 분할할 수 있습니다. 본 실습에서는 ASG 와 연결된 EC2 capacity providers 를 생성하고 부하 테스트를 수행하여 어떻게 클러스터 auto scaling 이 동작하는지 살펴봅니다.



### (7-2-1) Capacity Providers 생성

1. [Amazon ECS](#) 의 DEMOGO-ECS 클러스터로 이동하여 **Capacity Providers** 탭을 선택하고 **Create** 를 클릭합니다.

2. 아래와 같이 설정합니다.
  - a. Capacity provider name: **demogo-capacity-provider**
  - b. Auto Scaling group: **EC2ContainerService-[your cluster name]-EcsInstanceAsg** (0| ASG 는 클러스터를 생성할 때 자동 생성된 것입니다.)
  - c. Managed scaling: **Disabled** (실습을 위해 다음 단계에서 수동으로 구성합니다.)
  - d. Target capacity %: **100**
  - e. Managed termination protection: **Disabled**

#### Create Capacity Provider

Cluster name DEMOGO-ECS

Capacity provider name\* demogo-capacity-provider

The name of the capacity provider. Up to 255 characters are allowed, including letters (upper and lowercase), numbers, underscores, and hyphens. The name cannot be prefixed with 'aws', 'ecs', or 'fargate'.

Auto Scaling group EC2ContainerService-DEMOGO-ECS-EcsInstanceA...

Manually enter desired Auto Scaling group ARN

You must create an Auto Scaling group before creating a capacity provider. Use the [Amazon EC2 console](#) to create an Auto Scaling group.

Managed scaling  Enabled  Disabled

Target capacity % 100

Managed termination protection  Enabled  Disabled

3. 완료 후 Create 를 클릭해서 완료합니다.
4. 클러스터 초기 화면으로 이동 후 **Update Cluster** 를 클릭합니다.

#### Cluster : DEMOGO-ECS

Get a detailed view of the resources on your cluster.

Delete Cluster

Cluster ARN	arn:aws:ecs:ap-northeast-2:183837691404:cluster/DEMOGO-ECS
Status	ACTIVE
Registered container instances	2
Pending tasks count	0 Fargate, 0 EC2, 0 External
Running tasks count	2 Fargate, 4 EC2, 0 External
Active service count	1 Fargate, 2 EC2, 0 External
Draining service count	0 Fargate, 0 EC2, 0 External

5. 아래와 같이 **Add provider** 를 클릭해서 방금 생성한 **demogo-capacity-provider** 를 선택하고 **Update** 합니다.

#### (7-2-2) Auto Scaling Group 구성

1. [Amazon EC2 Auto Scaling Group](#) 으로 이동합니다.

2. EC2ContainerService-DEMOGO-ECS-EcsInstanceAsg 을 선택하고 Automatic scaling 탭에서 Create dynamic scaling policy 를 클릭합니다.

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
<input checked="" type="checkbox"/> EC2ContainerService	EC2ContainerService-DEMOGO-ECS...	2	-	2	2	2	ap-northeast-2a, ap-n...

Details    Activity    **Automatic scaling**    Instance management    Monitoring    Instance refresh

Dynamic scaling policies (0) [Info](#)

No dynamic scaling policies have been created  
Dynamic scaling policies use real-time data to scale your group based on configurable metrics.

**Create dynamic scaling policy**

3. 이후 설정은 아래와 같이 입력합니다. (빠른 결과 확인을 위해 임계치를 낮게 설정)
- Policy type: Target tracking scaling
  - Name: cluster-as-policy
  - Metric type: Average Network In (Bytes)
  - Target value: 50000
  - Instances need: 10
4. EC2ContainerService-DEMOGO-ECS-EcsInstanceAsg 을 선택하고 Edit 버튼을 클릭합니다.
5. Group size 에서 Desired: 2, Minimum: 2, Maximum: 4 로 세팅하고 Update 를 클릭합니다.

**Group size** [Info](#)

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

Desired capacity

Minimum capacity

Maximum capacity

### (7-2-3) 클러스터 부하 테스트

클러스터 오토 스케일링 테스트를 위한 스크립트 템플릿이 Cloud9 워크스테이션에 저장되어 있습니다. 서비스 부하 테스트와 마찬가지로, 이를 여러분들의 ALB URL로 수정한 뒤, 서비스 web 으로 부하 테스트를 수행할 것입니다.

1. [AWS Cloud9](#) 으로 이동합니다. **ecsworkshop** 을 선택하고 **Open IDE** 를 클릭합니다.
2. (중요) **cluster\_loadtest.sh** 를 열어서 ALB DNS 를 본인의 주소로 수정하여 저장합니다.

The screenshot shows the AWS Cloud9 IDE interface. On the left, there's a file tree with a folder named 'ecsworkshop'. Inside 'ecsworkshop', there are subfolders 'cats', 'dogs', and 'web'. Under 'web', there are files 'cluster\_loadtest.sh', 'ilovecats.jpg', 'ilovedogs.jpg', 'README.md', 'service\_loadtest.sh', and another 'README.md'. The 'cluster\_loadtest.sh' file is selected and highlighted with a red box. On the right, there are two terminal tabs: 'service\_loadtest.sh' and 'cluster\_loadtest.sh'. The 'cluster\_loadtest.sh' tab contains the command 'ab -c 800 -n 800 -t 10 demogo-alb-694381042.ap-northeast-2.elb.amazonaws.com'. The output of this command is shown below the command, with the entire output area highlighted by a red box.

3. 그리고 터미널에서 아래 명령어를 수행합니다.

**chmod 755 ecsworkshop/cluster\_loadtest.sh**

4. 아래 명령어는 1 분마다 총 3~4 회 수행합니다.

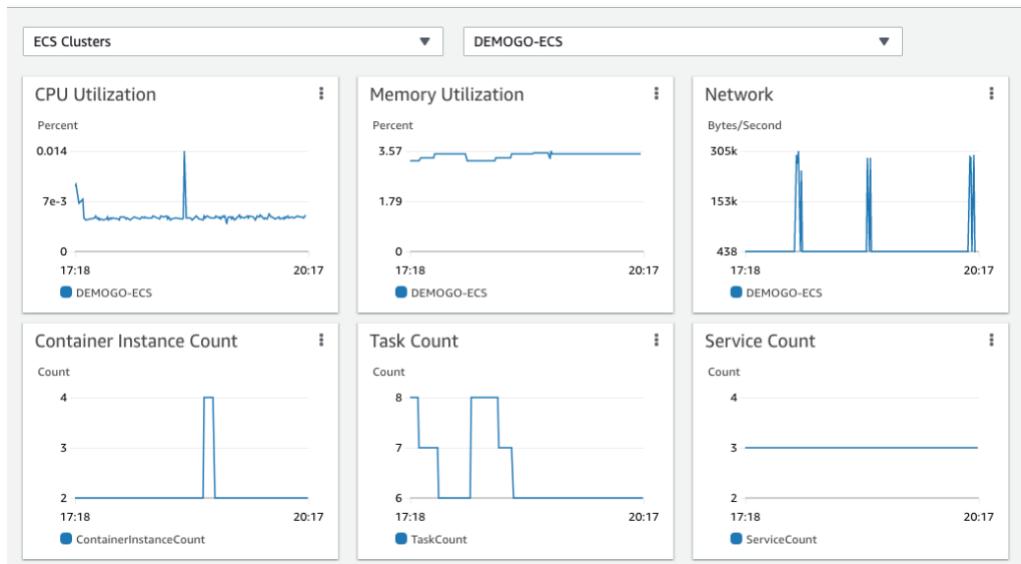
**./ecsworkshop/cluster\_loadtest.sh**

#### (7-2-4) 결과 확인

1. [CloudWatch Alarms](#) 으로 이동합니다.
2. 목표 추적(Target tracking)의 상태가 **Insufficient data** 일 수도 있습니다. 목표 추적 조정 정책은 지정한 메트릭의 데이터가 충분하지 않을 때 조정을 수행하지 않습니다.
3. 만약 상태가 **In alarm** 으로 바뀌지 않는다면, Workstation 에서 **cluster\_loadtest.sh** 스크립트를 연속으로 몇 번 더 실행합니다.
4. 4~5 분 후 [EC2 Auto Scaling](#) 메뉴에서 아래와 같이 instance 가 늘어나는 것을 볼 수 있습니다.

The screenshot shows the AWS EC2 Auto Scaling groups page. At the top, it says 'Auto Scaling groups (1/1)'. There is a search bar and buttons for 'Edit', 'Delete', and 'Create an Auto Scaling group'. Below this, a table lists one group: 'EC2ContainerService' (Launch template: 'EC2ContainerService-DEMOGO-ECS...', Instances: 4, Desired capacity: 4, Min: 2, Max: 4, Availability Zones: 'ap-northeast-2a, ap-northeast-2b'). Below the table, there are tabs for 'Details', 'Activity', 'Automatic scaling', 'Instance management' (which is selected), 'Monitoring', and 'Instance refresh'. Under 'Instance management', there is a table titled 'Instances (4)' with columns: 'Instance ID', 'Lifecycle', 'Instance type', 'Weighted capacity', 'Launch template/configurati...', 'Availability Zone', and 'Health stat...'. The four instances listed are all in 'Pending' or 'InService' state, with 't3.medium' instance type, 'EC2ContainerService-DEMOGO...' launch template, and healthy status in 'ap-northeast-2a' and 'ap-northeast-2b' zones.

5. ECS 클러스터의 **ECS Instances** 탭에서도 총 4 개의 instance 를 볼 수 있습니다.
6. **Metrics** 탭에서도 성능 지표를 확인할 수 있고 **View container insights** 를 클릭해서 상세 정보를 볼 수 있습니다.



**Cluster : DEMOGO-ECS**

Get a detailed view of the resources on your cluster.

Cluster ARN: arn:aws:ecs:ap-northeast-2:183837691404:cluster/DEMOGO-ECS

Status: ACTIVE

Registered container instances: 4

Pending tasks count: 0 Fargate, 0 EC2, 0 External

Running tasks count: 2 Fargate, 4 EC2, 0 External

Active service count: 1 Fargate, 2 EC2, 0 External

Draining service count: 0 Fargate, 0 EC2, 0 External

**ECS Instances** (selected tab)

An Amazon ECS instance is either an External instance registered using ECS Anywhere or an Amazon EC2 instance. To register an External instance, choose Register External Instances and follow the steps. [Learn More](#) To register an Amazon EC2 instance, you can use the Amazon EC2 console. [Learn More](#)

Last updated on June 5, 2022 8:14:55 PM (0m ago)

Container Instance	ECS Instance	Availability Zon...	External Insta...	Agent Connec...	Status	Running tasks...	CPU available	Memory avail...	Agent version
56fb13819d354c29a6a843...	i-027988cdc79...	ap-northeast-2a	false	true	ACTIVE	0	2048	3883	1.61.1
77b2d2a19a994da08b0a9...	i-0c79eb91a41...	ap-northeast-2b	false	true	ACTIVE	0	2048	3883	1.61.1
cf9665ba305449599db03f...	i-06051dadbd3...	ap-northeast-2b	false	true	ACTIVE	2	2048	3499	1.61.1
104a7758689400d91a058...	i-02f33613d141...	ap-northeast-2a	false	true	ACTIVE	2	2048	3499	1.61.1

## 8. 요약

- 지금까지 완전관리형 컨테이너 오케스트레이션 서비스인 Amazon ECS 에 cats and dogs 를 ALB 과 통합하여 배포했습니다.

- 각 서비스별 특성에 맞게 EC2 와 Fargate 중 적합한 시작 유형을 선택했습니다.
- Amazon ECR 에서 cats 와 dogs 의 도커 이미지를 관리했습니다.
- Amazon CloudWatch Container Insights 로 ECS 클러스터와 서비스, 컨테이너를 모니터링했습니다.
- FluentBit 기반의 AWS FireLens 를 이용해 Amazon CloudWatch Logs 로 컨테이너 로그를 수집하고 Log Insight 를 이용해 로그를 분석했습니다.
- 사용량에 맞게 ECS 서비스와 클러스터를 auto scaling 하도록 설정했습니다.
- 그럼에도 이 모든 과제를 수행하면서 단 하나의 서버도 구성, 패치, 리부팅하는 등의 관리를 할 필요가 없었습니다!

## 9. 실습 리소스 정리

- Amazon ECS 클러스터로 이동하여 DEMOGO-ECS 클러스터를 삭제합니다. (만약 삭제되지 않는다면 task 를 수동으로 삭제하고 다시 시도하시기 바랍니다.)

Cluster : DEMOGO-ECS

Get a detailed view of the resources on your cluster.

Cluster ARN	arn:aws:ecs:ap-northeast-2:183837691404:cluster/DEMOGO-ECS
Status	ACTIVE
Registered container instances	4
Pending tasks count	0 Fargate, 0 EC2, 0 External
Running tasks count	2 Fargate, 8 EC2, 0 External
Active service count	1 Fargate, 2 EC2, 0 External
Draining service count	0 Fargate, 0 EC2, 0 External

**Actions**

**Services**    **Tasks**    **ECS Instances**    **Metrics**    **Scheduled Tasks**    **Tags**    **Capacity Providers**

Service Name	Status	Service type	Task Definition	Desired tasks	Running tasks ...	Launch type	Platform version
web	ACTIVE	REPLICA	web:1	2	2	EC2	--
dogs	ACTIVE	REPLICA	dogsdef:1	2	2	FARGATE	LATEST(1.4.0)
cats	ACTIVE	REPLICA	catsdef:2	2	2	EC2	--

- 이번에는 Task Definitions 로 이동하여 **catsdef** 와 **dogsdef** 를 각각 클릭하여 모두 선택한 후 Actions 를 눌러 **deregister** 를 클릭합니다.

Task definition name : catsdef

Select a revision for more details

**Create new revision**    **Actions**

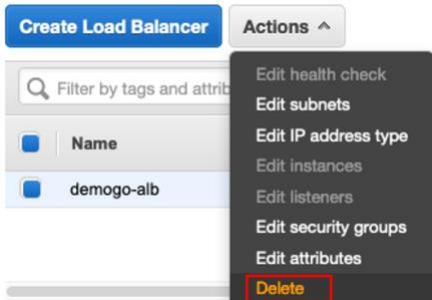
Status: **Active**    **Inactive**

Run Task    Create Service    Update Service

**Task Definition Name**    **Deregister**

**catsdef:2**    **Edit tags**

3. [Amazon ECR](#) Repositories로 이동하여 cats 와 dogs 를 삭제합니다.
4. [Amazon EC2](#) Load Balancer로 이동하여 demogo-alb 를 삭제합니다.



5. Target Groups에서 web, cats, dogs 를 선택 후 삭제합니다.

Target groups (3/3) <a href="#">Info</a>						
<input type="checkbox"/> <a href="#">Search or filter target groups</a>						
	Name	ARN	Port	Protocol	Target type	L
<input checked="" type="checkbox"/>	cats	<a href="#">arn:aws:elasticloadbalancin...</a>	80	HTTP	Instance	e5
<input checked="" type="checkbox"/>	dogs	<a href="#">arn:aws:elasticloadbalancin...</a>	80	HTTP	IP	e5
<input checked="" type="checkbox"/>	web	<a href="#">arn:aws:elasticloadbalancin...</a>	80	HTTP	Instance	e5

Actions ▾ [Create target group](#)

- [Delete](#)
- [Register targets](#)
- [Edit health check settings](#)
- [Edit attributes](#)
- [Manage tags](#)
- [Associate with a new load balancer](#)
- [Associate with an existing load balancer](#)

6. [IAM role](#)에서 ecs-demogo 를 검색하여 삭제합니다.
7. [CloudFormation](#)에서 ecs-demogo 스택을 삭제합니다. (Resources 탭에서 VPC를 제외한 나머지 리소스가 모두 삭제되었고 완료되지 않았다면 아래 8 번을 수행합니다.)

Stacks (2)				
<input type="checkbox"/> <a href="#">Filter by stack name</a>				
Stack name	Status	Created time	Description	
aws-cloud9-ecsworkshop- a9d63d071ba6475490663281b92063b 2	<a href="#">CREATE_COMPLETE</a>	2022-06-04 12:14:11 UTC+0900	-	
ecs-demogo	<a href="#">CREATE_COMPLETE</a>	2022-06-04 12:13:01 UTC+0900	DemoGo-EC...	

Actions ▾ [Delete](#)

8. [VPC](#)로 이동하여 DemoGoECSVPC 를 수동으로 삭제합니다.

Your VPCs (1/2) <a href="#">Info</a>						
<input type="checkbox"/> <a href="#">Filter VPCs</a>						
	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	
<input checked="" type="checkbox"/>	DemoGoECSVPC	vpc-088fd63a7fe55f0bf	<a href="#">Available</a>	10.0.0.0/16	-	e1
<input type="checkbox"/>	-	vpc-090d7898055e6a359	<a href="#">Available</a>	172.31.0.0/16	-	e1

Actions ▾ [Create VPC](#)

- [Create default VPC](#)
- [Create flow log](#)
- [Edit CIDRs](#)
- [Edit DHCP options set](#)
- [Edit DNS hostnames](#)
- [Edit DNS resolution](#)
- [Manage middlebox routes](#)
- [Manage tags](#)
- [Delete VPC](#)

9. **CloudFormation**에서 모든 스택이 삭제되는 것을 확인합니다.
10. [Amazon CloudWatch](#) Log groups 을 확인 후 아래 리소스가 있다면 모두 삭제합니다.
  - a. /aws/ecs/containerinsights/DEMOGO-ECS/performance
  - b. ecs-demogo-log
  - c. firelens-container

# 수고하셨습니다!