



**Builders Online - DevOps**  
AWS CDK Hands-on Lab

---

June 24, 2021

# 1. AWS CDK 로 ECS DevOps 실전 적용하기

본 실습은 소수의 DevOps Engineer 가 동시에 다수의 서비스 개발에 참여하여 MSA/IaC 기반으로 전체 Infrastructure 를 구성 및 배포하기 위한 Best Practice 를 AWS CDK 기반으로 실습합니다. 본 실습을 통해서 완성된 CDK Project 를 활용하면 Container 기반의 DevOps(Culture+Practice+Tool) 역량을 빠르게 끌어 올릴 수 있습니다.

## 4-1. 소스 코드 준비(DevOps Team)

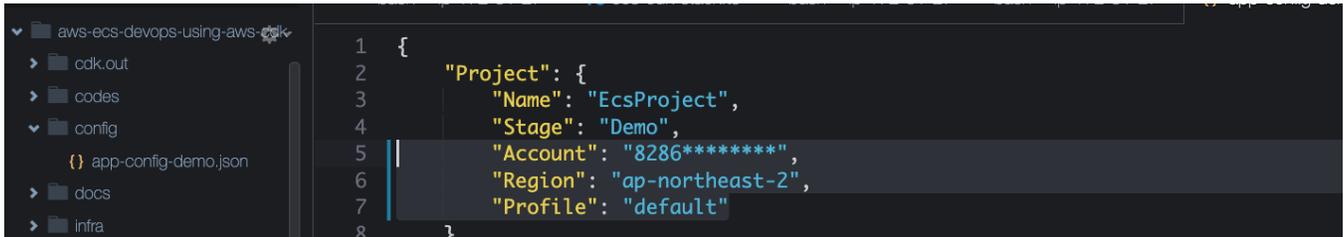
[여기](#)를 클릭하여 본 실습을 위한 기본 소스 코드를 확인하고, 다음과 같이 실습용 코드를 다운로드합니다. 오늘 실습을 위해서 hol\_20210624 branch 가 준비되어 있습니다.

```
git clone https://github.com/aws-samples/aws-ecs-devops-using-aws-cdk.git
-b hol_20210624 aws-ecs-devops-using-aws-cdk-infra
cd aws-ecs-devops-using-aws-cdk-infra
```

다음 작업을 위하여 clone 한 프로젝트를 IDE 를 통하여 CDK 프로젝트를 Open 합니다.

## 4-2. 배포 Target 설정(DevOps Team)

Config/app-config-demo.json 파일을 열어 배포 Target 을 지정합니다. 여기서 Target 은 account(aws sts get-caller-identity), region(ap-northeast-2)그리고 profile(default 인 경우 "default"로 입력)을 뜻합니다.



끝으로 다음 명령어를 실행하여 CDK 프로젝트 초기 셋업을 시작합니다.

```
export APP_CONFIG=config/app-config-demo.json
sh scripts/setup_initial.sh config/app-config-demo.json
```

## 4-3. 기본 Infrastructure 배포하기(DevOps Team)

infra/app-main.ts 파일을 열어서 다음을 작성합니다. VPC 와 ECS Cluster 를 생성해주는 Stack 입니다.

```
import { VpcInfraStack } from './common-infra/vpc-infra-stack';
...
...
new VpcInfraStack(appContext, appContext.appConfig.Stack.VpcInfra);
```

## 완성된 코드 모습

```

1  #!/usr/bin/env node
2  import 'source-map-support/register';
3
4  import { AppContext } from '../lib/template/app-context';
5  import { VpcInfraStack } from '../common-infra/vpc-infra-stack';
6
7
8
9  const appContext = new AppContext({
10     appConfigEnvName: 'APP_CONFIG',
11 });
12
13 if (appContext.stackCommonProps !== undefined) {
14     new VpcInfraStack(appContext, appContext.appConfig.Stack.VpcInfra);
15 }
16
    
```

최종적으로 다음과 같이 새로 추가된 스택을 배포해줍니다.

```

cdk list
cdk deploy *VpcInfraStack
    
```

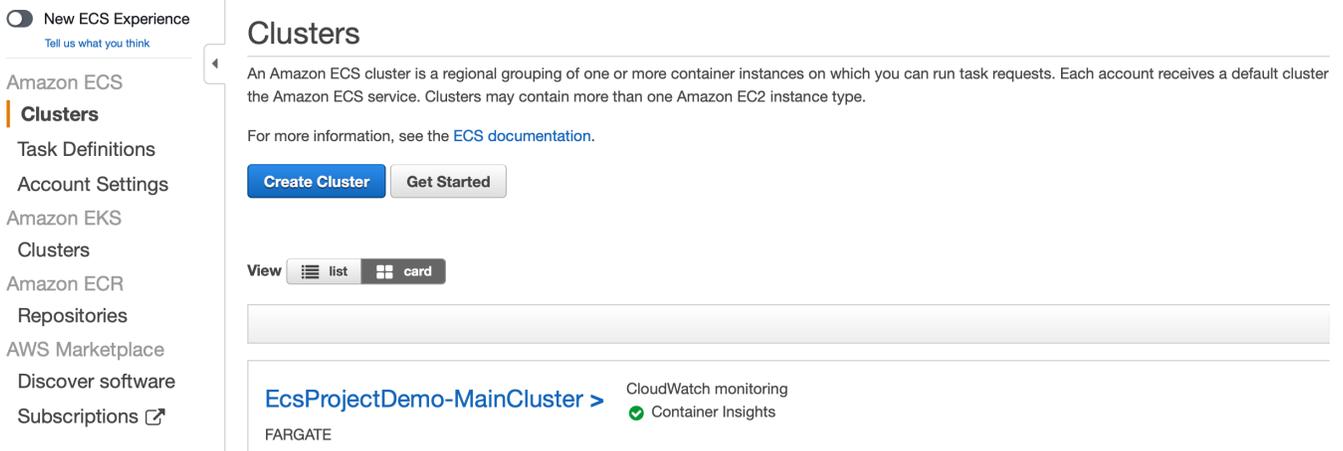
배포가 완료되면 VPC와 ECS Cluster가 배포된 것을 CloudFormation이나 각 서비스 화면에서 확인할 수 있습니다.

The screenshot shows the AWS CloudFormation console. At the top, there's a search bar for 'EcsProjectDemo-Vpc' and a status dropdown set to 'Active'. Below this is a table of stacks:

Stack name	Status	Created time	Description
EcsProjectDemo-VpcInfraStack	UPDATE_COMPLETE	2021-06-22 17:39:38 UTC+0900	-

Below the stacks table, there's a section for 'Your VPCs (1/1)'. It includes a search bar and a filter 'search: Ecs'. Below that is a table of VPCs:

Name	VPC ID	State	IPv4 CIDR	IPv6
EcsProjectDemo-VpcInfraStack/CommonVPC	vpc-08615067c04bfa72a	Available	10.0.0.0/16	-



#### 4-4. Web Frontend Infrastructure 배포하기(DevOps Team)

DevOps Engineer 는 Service 팀과 미팅을 하고, 이를 바탕으로 아키텍처를 설계 후, 다음과 같이 config/app-config-demo.json 에 Configuration 정보를 작성합니다. 본 예제에서는 Frontend 이기 때문에 ALB 가 Public 80 포트로 오픈됩니다. 또한 운영 시 Notification 을 받을 e-mail 주소를 SubscriptionEmails 에 입력합니다.

```

EXPLORER
OPEN EDITORS 1 UNSAVED
AWS-ECS-DEVOPS-USING-AWS-CDK
  > cdk.out
  > codes
  > config
  {} app-config-demo.json M
  > docs
  > infra
    > common-infra
    > ecs-service
  TS app-main.ts
  > lib
  > node_modules
  > scripts
  > test
  .gitignore
  .npmignore
  {} cdk.context.json
  {} cdk.json
  CONTRIBUTING.md
  JS jest.config.js
  LICENSE
  LICENSE-SAMPLECODE

{} app-config-demo.json M • TS app-main.ts
config > {} app-config-demo.json > {} Stack > {} SampleFrontendFlask > AppPath
44 "SampleFrontendFlask": {
45   "Name": "SampleFrontendFlaskStack",
46   "InfraVersion": "'1.0.0'",
47
48   "PortNumber": 80,
49   "InternetFacing": true,
50
51   "AppPath": "codes/sample-front-end-flask",
52   "DesiredTasks": 1,
53   "Cpu": 1024,
54   "Memory": 2048,
55
56   "AutoScalingEnable": false,
57   "AutoScalingMinCapacity": 1,
58   "AutoScalingMaxCapacity": 2,
59   "AutoScalingTargetInvocation": 50,
60
61   "AlarmThreshold": 200,
62   "SubscriptionEmails": ["kwonyul@amazon.com"]
63 },

```

infra/app-main.ts 파일을 열어서 다음을 작성합니다. ALB 와 ECS Service/Task 를 생성해주는 Stack 입니다.

```

import { EcsAlbServiceStack } from './ecs-service/ecs-alb-service-stack';
...

```

```
...
new EcsAlbServiceStack(appContext,
appContext.appConfig.Stack.SampleFrontendFlask);
```

최종적으로 다음과 같이 새로 추가된 스택을 배포해줍니다.

```
cdk list
cdk deploy *SampleFrontendFlaskStack
```

배포가 완료되면 ALB 와 ECS Service/Task 가 배포된 것을 CloudFormation 이나 각 서비스 화면에서 확인할 수 있습니다.

The screenshot shows the AWS CloudFormation console. At the top, there's a 'Stacks (1)' section with a search bar containing 'EcsProjectDemo-SampleF' and a filter set to 'Active'. Below this is a table with one entry: 'EcsProjectDemo-SampleFrontendFlaskStack' with status 'CREATE\_COMPLETE' and created time '2021-06-22 23:31:27 UTC+0900'. The left sidebar shows navigation options for Amazon ECS, EKS, and ECR. The main content area displays the details for the service 'EcsProjectDemo-SampleFrontendFlaskStack-EcsAlbInfraConstrunctService9FBF6028-EI7P2SycuVLK'. Key details include: Cluster 'EcsProjectDemo-MainCluster', Status 'ACTIVE', Task definition 'EcsProjectDemoSampleFrontendFlaskStackEcsAlbInfraConstrunctServiceTaskDefBE18951E:2', Service type 'REPLICA', Launch type 'FARGATE', Service role 'AWSServiceRoleForECS', and Created By 'arn:aws:iam::751571716296:user/ky-01'. Below the details are tabs for 'Details', 'Tasks', 'Events', 'Auto Scaling', 'Deployments', 'Metrics', 'Tags', and 'Logs'. The 'Load Balancing' section shows a table with columns 'Target Group Name', 'Container Name', and 'Container Port', containing one entry: 'EcsPr-EcsAl-168XO2G43SUDV', 'EcsProjectDemo-SampleFrontendFlaskStackContainer', and '80'.

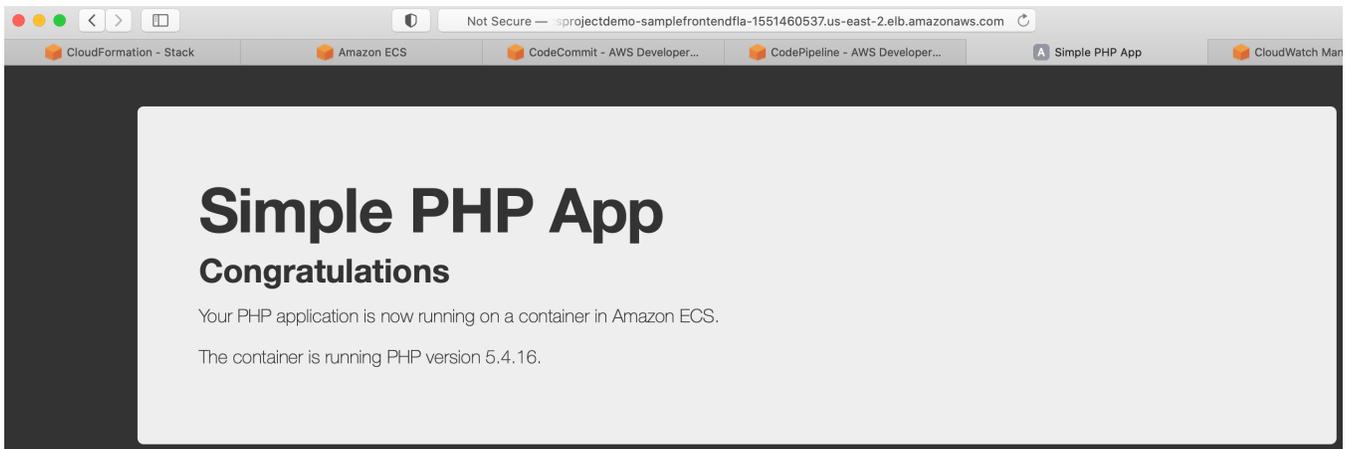
ALB 의 DNS 주소를 웹 브라우저를 통하여 정상적으로 open 되는지 확인합니다. ALB 의 주소는 다음 그림과 같이 CDK 배포 과정에 출력됩니다.

✔ EcsProjectDemo-SampleFrontendFlaskStack

Outputs:

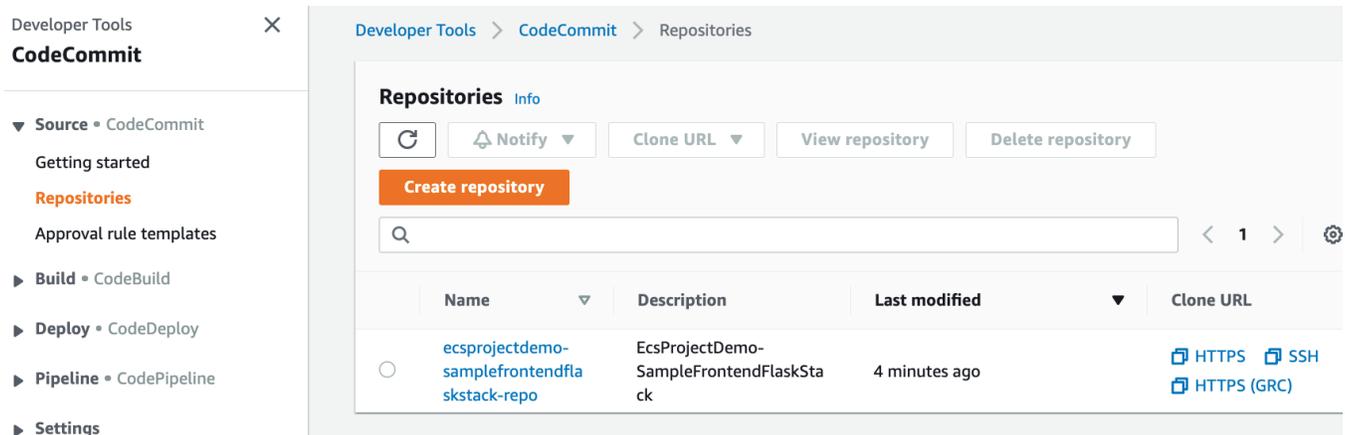
EcsProjectDemo-SampleFrontendFlaskStack.EcsAlbCidConstructCodeCommitName916C195C = [ecsprojectdemo-samplefrontendflaskstack-repo](https://github.com/ecsprojectdemo-samplefrontendflaskstack-repo)  
 EcsProjectDemo-SampleFrontendFlaskStack.EcsAlbInfraConstructServiceLoadBalancerDNSF445CBCD = [EcsProjectDemo-SampleFrontendFla-1551460537.us-east-2.elb.amazonaws.com](https://EcsProjectDemo-SampleFrontendFla-1551460537.us-east-2.elb.amazonaws.com)  
 EcsProjectDemo-SampleFrontendFlaskStack.EcsAlbInfraConstructServiceServiceURL290953F6 = <http://EcsProjectDemo-SampleFrontendFla-1551460537.us-east-2.elb.amazonaws.com>

본 초기 웹 페이지는 PHP Sample Page 로서 github 를 통해서 제공되는 초기 화면일 뿐이며 향후 스텝에 CodePipeline 을 통하여 정식 ECR 에 있는 우리만의 Docker Image 가 배포될 예정입니다. 아직 서비스팀에 의해서 정상 Push 되지 않은 상태이기 때문에 최초 배포시에 임시방편으로 지정한 것 뿐입니다.



DevOps 팀은 최종적으로 codecommit 주소(ecsprojectdemo-samplefrontendflaskstack-repo) 및 소스 위치 directory(codes/sample-frontend-flask)를 Service 팀에게 전달합니다.

### 새로 생성된 CodeCommit



## 새로 생성된 ECR Repository

The screenshot shows the AWS ECR console interface. On the left, there is a navigation menu with options for Amazon Container Services, Amazon ECS, Amazon EKS, and Amazon ECR. The 'Amazon ECR' section is expanded, showing 'Repositories' as the selected option. The main content area displays the 'Private repositories (1)' section. A search bar is present with the text 'Find repositories'. Below the search bar, there is a table with the following columns: Repository name, URI, Created at, and Tag immutability. The table contains one entry for the repository 'ecsprojectdemo-samplefrontendflaskstack-repo'.

Repository name	URI	Created at	Tag immutability
ecsprojectdemo-samplefrontendflaskstack-repo	828693440215.dkr.ecr.ap-northeast-2.amazonaws.com/ecsprojectdemo-samplefrontendflaskstack-repo	Jun 24, 2021 03:19:24 AM	Disabled

## 4-5. Web Frontend Application 배포하기(Service Team)

Service 팀은 이제 사전에 전달받은 codecommit 주소로 clone 후, 사전에 전달받은 directory(codes/sample-frontend-flask)에 service 로직 코드를 구현하여 준비합니다. 본 실습에서는 편의 상 codes 디렉토리에 미리 준비해두었으며, 실제로는 codes 하위의 각 비즈니스 로직 구현은 각 서비스 개발팀이 자신의 Repository 에서 구현합니다.

```
git clone https://git-codecommit.ap-northeast-2.amazonaws.com/v1/repos/ecsprojectdemo-samplefrontendflaskstack-repo
sample-frontend-flask
  cd sample-frontend-flask
  mkdir codes
  cp -r ../aws-ecs-devops-using-aws-cdk-infra/codes/sample-frontend-flask ./codes/
  git add .
  git commit -m "initial commit"
  git push origin master
```

directory 구조는 아래와 같습니다.

```

EXPLORER
... Dockerfile x
OPEN EDITORS
ECSPROJECTDEMO-SAMPLEFRONTENDFLAS...
  codes / sample-frontend-flask
    app
      static / css
      templates
      main.py
      Dockerfile
codes > sample-frontend-flask > Dockerfile > ...
1 FROM alpine:3.10
2
3 RUN apk add python3 py-pip && \
4 python3 -m ensurepip && \
5 pip install --upgrade pip && \
6 pip install flask
7
8 WORKDIR /app
9 COPY ./app/ /app/
10
11 CMD ["python", "main.py"]
12
  
```

CodeCommit 의 master branch 로 push 이벤트 발생 시에 배포 Pipeline 이 시작되도록 구현되어 있습니다. 이제 CodePipeline 으로 이동하여 소스가 자동 배포되는지 확인합니다.

Developer Tools > CodePipeline > Pipelines

Pipelines Info [Refresh] [Notify] [View history] [Release change] [Delete pipeline] [Create pipeline]

Q Front X < 1 > [Settings]

Name	Most recent execution	Latest source revisions	Last executed
EcsProjectDemo-SampleFrontendFlaskStack-Pipeline	In progress	CodeCommit_SourceMerge - daba8854: initial commit	27 minutes ago

다음과 같이 Approve 상태에서 Review 버튼을 클릭해야지 다음 스텝으로 진행되어 최종 배포됩니다.

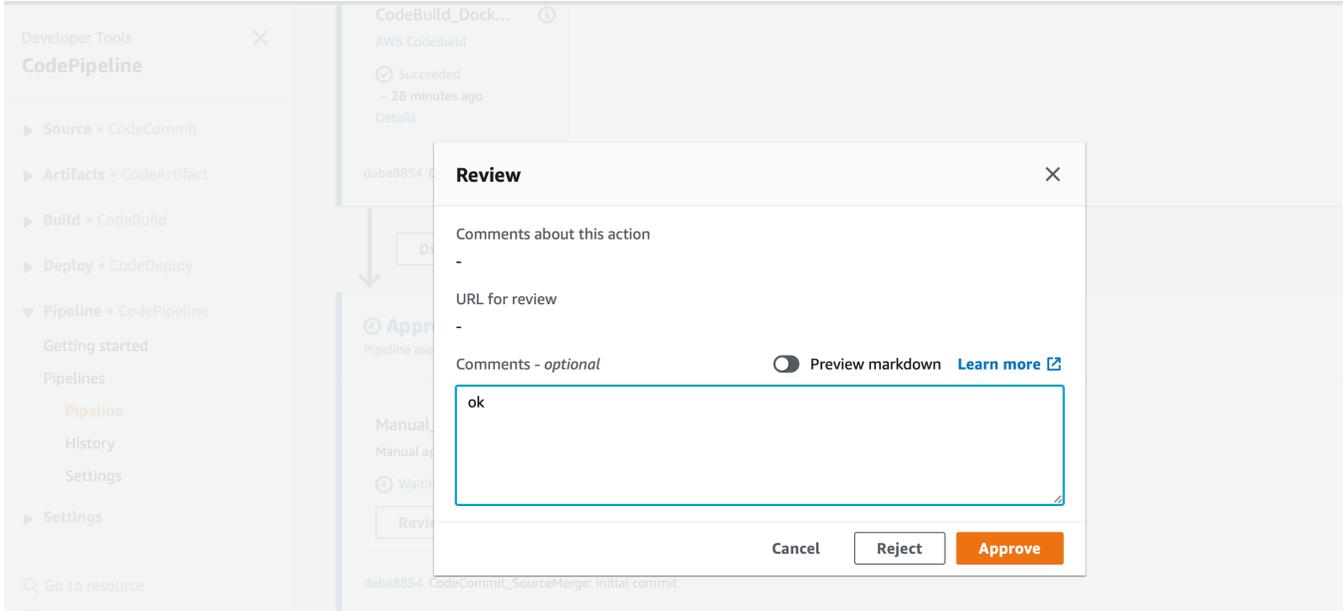
Approve Pending  
Pipeline execution ID: 0d697f9e-702e-4d5e-a342-0f4592df4d51

Manual\_Approve ⓘ  
Manual approval

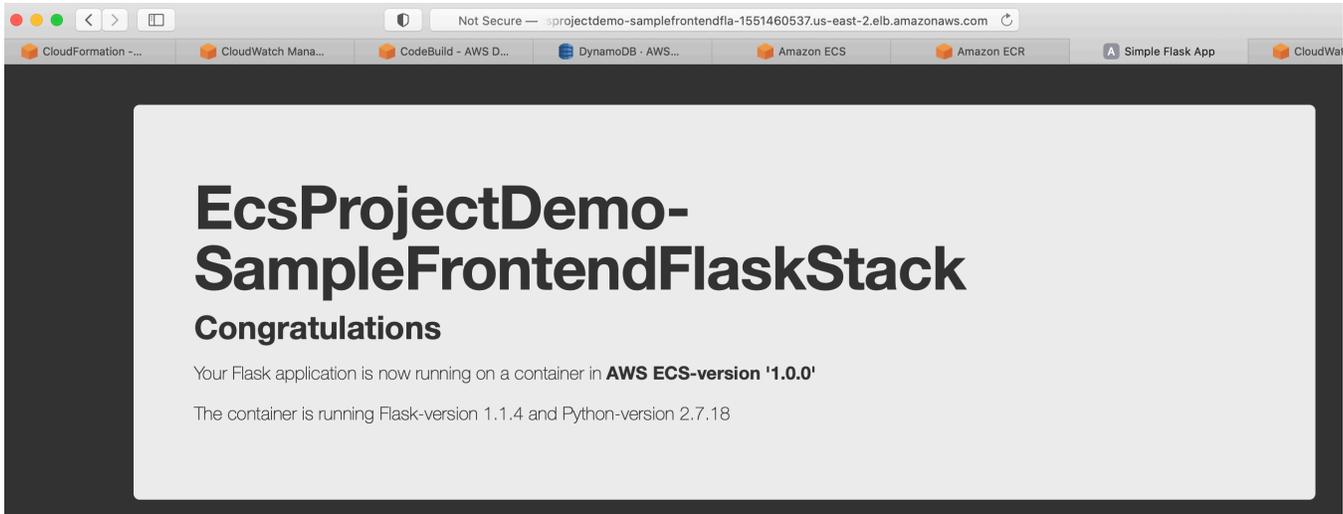
Waiting for approval -

Review

daba8854 CodeCommit\_SourceMerge: initial commit



최종 배포 완료 후에는 다음과 같이 Flask 로 구현된 웹 페이지를 확인할 수 있습니다.



#### 4-6. API Backend Infrastructure 배포하기(DevOps Team)

본 단계는 4-4 단계와 거의 유사합니다. 단 본 서비스는 Backend API Service 이기 때문에 Private 으로 배포되고 이때문에 Web Browser 를 통하여 육안으로 직접 확인이 불가하며 대신 Test 용 ECS Task 를 동일 VPC 에 배포하여 AB(Apache Benchmarking) Tool 을 이용하여 원격 자동 부하 테스트합니다. 또한 이때 CloudMap 을 이용하여 Private DNS 로 해당 서비스를 ServiceDiscovery 하여 접속합니다.

- Service Infra Config 작성:

운영 시 Notification 을 받을 e-mail 주소를 SubscriptionEmails 에 입력합니다.

```

22 },
23
24 "SampleBackendFastapi": {
25   "Name": "SampleBackendFastapiStack",
26   "InfraVersion": "1.0.0",
27
28   "PortNumber": 80,
29   "InternetFacing": false,
30
31   "AppPath": "codes/sample-backend-fastapi",
32   "DesiredTasks": 1,
33   "Cpu": 1024,
34   "Memory": 2048,
35
36   "AutoScalingEnable": false,
37   "AutoScalingMinCapacity": 1,
38   "AutoScalingMaxCapacity": 2,
39   "AutoScalingTargetInvocation": 50,
40
41   "TableName": "LogTable",
42
43   "AlarmThreshold": 200,
44   "SubscriptionEmails": ["kwonyul@amazon.com"]
45 },

```

- Service Infra 코드 작성:

```

import { EcsAlbServiceStack } from './ecs-service/ecs-alb-service-stack';
...
...
new EcsAlbServiceStack(appContext,
appContext.appConfig.Stack.SampleBackendFastapi);

```

- Service Infra 배포:

```

cdk list
cdk deploy *SampleBackendFastapiStack

```

- Test Infra 코드 작성:

```

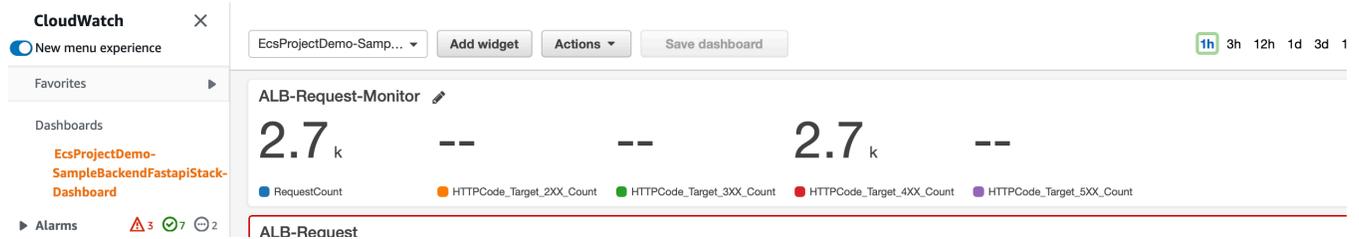
import { EcsCommonServiceStack } from './ecs-service/ecs-common-service-stack';
...
...
new EcsCommonServiceStack(appContext,
appContext.appConfig.Stack.LoadTesterScript);

```

- Test Infra 배포:

```
cdk list
cdk deploy *LoadTesterScriptStack
```

최종적으로 CloudWatch 를 통하여 실시간 현황을 확인할 수 있습니다. 하지만 아직 서비스팀을 통해서 서비스 로직이 배포되지 않았기 때문에 사전에 정의된 Test 용 Request URL 이 구현되어 있지 않아서 2XX 가 아닌 4XX 의 에러들만 감지되고 있습니다.



DevOps 팀은 최종적으로 codecommit 주소(ecspromjctdemo-samplebackendfastapistack-repo) 및 소스 위치 directory(codes/sample-backend-fastapi)를 Service 팀에게 전달합니다.

#### 4-7. API Backend Application 배포하기(Service Team)

서비스팀은 동일하게 CodeCommit ecspromjctdemo-samplebackendfastapistack-repo 를 clone 하고, sample-backend-fastapi 를 복사 후, 최종 push 합니다. 최종적으로 CodePipeline 이 트리거되어 배포가 자동화되고 Accept 과정을 거쳐 최종 배포된 것을 확인할 수 있습니다.

```
git clone https://git-codecommit.ap-northeast-2.amazonaws.com/v1/repos/ecspromjctdemo-samplebackendfastapistack-repo
cd sample-backend-fastapi
mkdir codes
cp -r ../aws-ecs-devops-using-aws-cdk-infra/codes/sample-backend-fastapi ./codes/
git add .
git commit -m "initial commit"
git push origin master
```

최종 directory 모습

```

EXPLORER
OPEN EDITORS
  Dockerfile codes/sample-backend-f...
ECSPROJECTDEMO-SAMPLEBACKENDFASTA...
  codes / sample-backend-fastapi
    app
      ddb_handler.py
      main.py
      Dockerfile
Dockerfile x
codes > sample-backend-fastapi > Dockerfile > ...
1 FROM tiangolo/uvicorn-gunicorn-fastapi:python3.7
2
3 RUN pip install boto3
4
5 COPY ./app /app
6 |
    
```

## Pipeline 배포 모습

Developer Tools

**CodePipeline**

- ▶ Source • CodeCommit
- ▶ Build • CodeBuild
- ▶ Deploy • CodeDeploy
- ▼ Pipeline • CodePipeline
  - Getting started
  - Pipelines**
  - Settings

Developer Tools > CodePipeline > Pipelines

**Pipelines** Info

Name	Most recent execution	Latest source revisions	Last executed
EcsProjectDemo-SampleBackendFastapiStack-Pipeline	In progress	CodeCommit_SourceMerge – fbcc2f89: initial commit	Just now

---

Developer Tools > CodePipeline > Pipelines > EcsProjectDemo-SampleBackendFastapiStack-Pipeline

**EcsProjectDemo-SampleBackendFastapiStack-Pipeline**

✔ **Source** Succeeded  
 Pipeline execution ID: 22530b0e-7230-444c-a8ee-bb96e7d98b31

CodeCommit\_So... ⓘ

AWS CodeCommit

✔ Succeeded - 3 hours ago  
eb6357f7

eb6357f7 CodeCommit\_SourceMerge: initial commit

↓ Disable transition

✔ **Build** Succeeded  
 Pipeline execution ID: 22530b0e-7230-444c-a8ee-bb96e7d98b31

CodeBuild\_Dock... ⓘ

AWS CodeBuild

✔ Succeeded - 3 hours ago  
Details

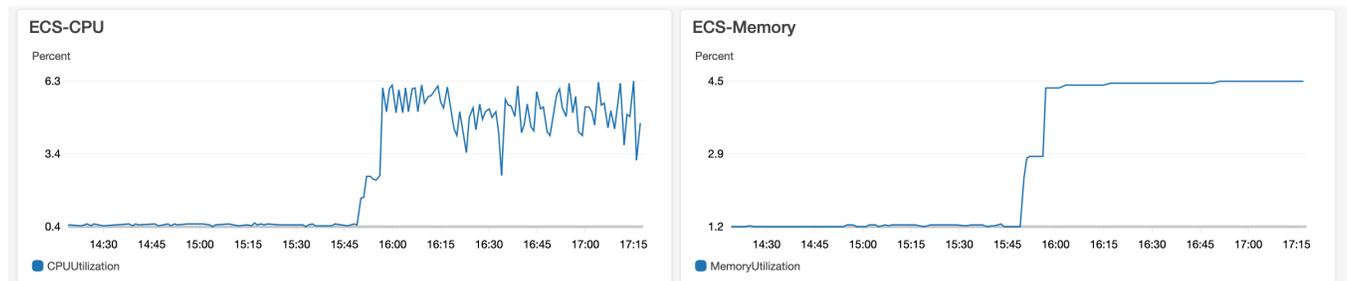
최종 배포되어 CloudWatch Dashboard 를 통하여 실시간 모니터링되는 화면은 다음과 같습니다. 전체 Request 갯수와 2XX Request 가 동일하여 이슈가 없음을 확인할 수 있습니다.

### ALB Metric



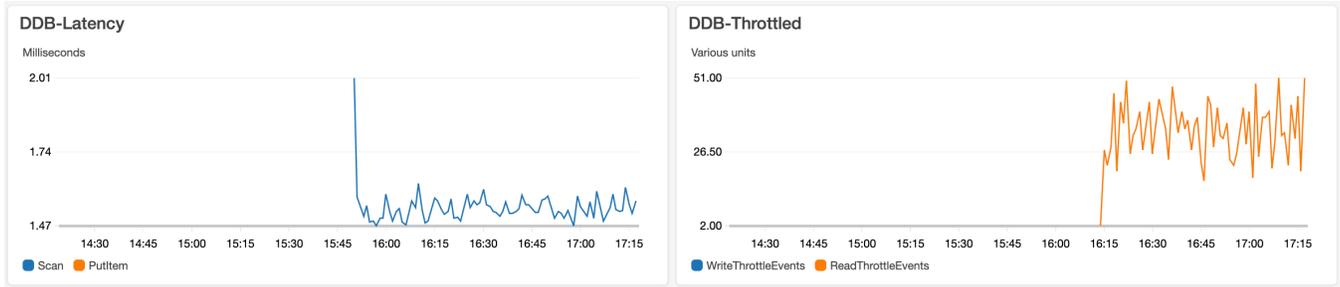
### ECS Service Metric

CPU 와 Memory 의 Utilization 이 10% 이하로 현재 여유 가동되고 있음을 확인할 수 있습니다.



### DynamoDB Metric

내부적으로 DDB 의 Capacity 가 낮아서 Throttle 이 발생하고 있음을 확인할 수 있습니다.



#### 4-8. 도전 과제

- DDB 의 Read/Write Capacity 를 높여서 Throttle 을 제거하세요.
- Frontend 에서 Backend 를 ServiceDiscovery 를 통하여 Endpoint 를 생성하여 호출하세요.
- ECS Service 에 AutoScaling 을 구현하고, Scaling 될 수 있도록 Load Test 강도를 높여 보세요.

#### 4-9. 정리하기

최종적으로 다음과 같은 사전에 준비된 script 파일을 실행하여 리소스를 한번에 정리합니다. 참고로 DynamoDB, CodeCommit 그리고 ECR 과 같은 저장소들은 CDK 를 통해서 자동 삭제되지 않을 수 있으므로 수동 삭제하셔야 합니다.

```
sh scripts/destroy_stacks.sh config/app-config-demo.json
```

물론 다시 새롭게 모든 스택을 한번에 배포하려면 다음과 같이 script 파일을 실행시킬 수 있습니다.

```
sh scripts/deploy_stacks.sh config/app-config-demo.json
```