

1 Lab3. Kafka Producer Application을 ECR에 배포하기

3 1. Docker Engine 설치하기

```
4 -$ sudo yum update -y
5 -$ sudo amazon-linux-extras install docker
6 -$ sudo service docker start
7 -$ sudo setfacl -m user:ec2-user:rw /var/run/docker.sock
8 -$ docker -version
9 Client:
10 Version:      20.10.23
11 API version:  1.41
12 Go version:   go1.18.9
13 Git commit:   7155243
14 Built:        Tue Apr 11 22:56:36 2023
15 OS/Arch:      linux/amd64
16 Context:      default
17 Experimental: true
18
19 Server:
20 Engine:
21 Version:      20.10.23
22 API version:  1.41 (minimum version 1.12)
23 Go version:   go1.18.9
24 Git commit:   6051f14
25 Built:        Tue Apr 11 22:57:17 2023
26 OS/Arch:      linux/amd64
27 Experimental: false
28 containerd:
29 Version:      1.6.19
30 GitCommit:    1e1ea6e986c6c86565bc33d52e34b81b3e2bc71f
31 runc:
32 Version:      1.1.4
33 GitCommit:    5fd4c4d144137e991c4acebb2146ab1483a97925
34 docker-init:
35 Version:      0.19.0
36 GitCommit:    de40ad0
```

39 2. requirements.txt 생성하기

```
40 $ pwd
41 /home/ec2-user/kafka-client
42
43 $ nano requirements.txt
44
45
```

46 3. Dockerfile 생성하기

```
47 $ pwd
48 /home/ec2-user/kafka-client
49
50 $ nano Dockerfile
51
52
```

53 4. producer.py 파일을 /home/ec2-user/kafka-client 디렉토리로 이동하기

```
54 $ mv producer.py /home/ec2-user/kafka-client/
55
56
```

57 5. Amazon ECR에서 리포지토리 생성하기

- 1)[서비스] > [컨테이너] > [Elastic Container Registry] 이동한다.
  - 2)[리포지토리 생성] > [시작하기] 버튼 클릭
  - 3)[리포지토리 생성] 페이지에서
    - [일반설정] > [표시 여부 설정] : [프라이빗]
    - [일반설정] > [리포지토리 이름] : msk/producer
  - 4)나머지는 기본값 그대로 사용한다.
  - 5)[리포지토리 생성] 버튼 클릭
6. {계정}-msk-client EC2 인스턴스에서 IAM 역할 수정하기
- ※ aws configure를 이용한 AWS CLI의 Access Key와 Secret Access Key보다 더 보안이 강화되는 효과가 있음.
  - 1)해당 인스턴스 상세 페이지로 이동하여 [작업] > [보안] > [IAM 역할 수정]을 선택한다.
  - 2)[IAM 역할 수정] 페이지에서, [새 IAM 역할 생성] 링크 클릭
  - 3)[Identity and Access Management(IAM)] > [액세스 관리] > [역할] 페이지에서 [역할 만들기] 클릭
  - 4)[신뢰할 수 있는 엔티티 선택] 페이지에서,
    - [신뢰할 수 있는 엔티티 유형] > [AWS 서비스] 선택
    - [사용 사례] > [일반 사용 사례] > [EC2] 선택
    - [다음] 버튼 클릭
  - 5)[권한 추가] 페이지에서 [권한 정책] 목록 중 "AdministratorAccess" 라고 검색하여 [정책 이름]이 [AdministratorAccess]를 찾아서 체크한다.
  - 6)[다음]을 클릭한다.
  - 7)[이름 지정, 검토 및 생성]페이지에서,
    - [역할 이름] : {계정}-ecr-admin-role
    - [역할 생성] 버튼 클릭

85  
86 8) 다시 [IAM 역할 수정] 페이지로 돌아와서,  
87 -[IAM 역할] : 방금 생성한 {계정}-ecr-admin-role (보이지 않으면 리프래쉬 버튼 클릭)  
88 -[IAM 역할 업데이트] 버튼 클릭  
89  
90

91 7. AWS ECR에 이미지 배포하기

92 1) ECR에 로그인하기

93 -Visual Studio Code의 Terminal에서 다음의 명령으로 로그인한다.  
94 \$ aws ecr get-login --no-include-email --region ap-northeast-2  
95 docker login -u AWS -p  
eyJWYXlsb2FkIjoiazmJxSEiVUDdocGFscVWwU44djQ5U3c5NGdLQWRfUGdmWWRwbFYzazdBVlic0RWTG9KcDdMeDZjMjE2K  
1dkY0InZDZpS1B6NEt3QXBMRHkva2ZpbmJaTExDcEhwTm4yVnRvNzIeUUVJjelovM084VEVzZFBQV0Rqd0JPNkZiVktKeFh3aWZ  
0UUIFaGJ2Z2WIDQzdyNUhIZFU3WDVUSkiHSHwEOXNwTtDTT1hvr1IGQ3hHR1VEQy9INU5aYzVXWUUVjdmRW4wbVdHbWhM  
UkdndYkRuaEtZrK01c3JxcHgwctNPLzLSR3d0ci8OVkdOWCt2YjRtTjB5SldVOHBUE5FemxXMIpXYktCdWVdVdMNVA5Zl  
ZEdjlqeElEaDRKaG9kb1RNemNCWkU5YmpHeGhnR2NiNE1KR3VEbmwwUTM3Q2tXWXRmDFg3YU85Nk4rMXZPV0J4WktJWW  
5OS0p0azUwZldNeG5XMKhTMmlrbWIwSnpHTk95Wih0MFBsdXNlVXNzSzg0aWNYVTFoZXRUQ0NpNlVFNlxYnY1QmhyUW1M  
UDBSbDNnZmlDbzlnZTI1VWVWk1hGbnRZM1RWNIYfak9pTfPfa3RqVzIrv3hMalBCbFJ4VW5ySfDjbkNMMEpGdUdmRUdZN1  
o3YVfUcTRtMXVhQVIJQXI5YmJmb21GajdsNGRVsZj5ZGZsXskJ3cINzUWJlb2Q4YUsyYmlsa0JJNVpmUFlqcVlzRUZEdnJGdWoy  
SEpjRTRhS05yZkVMM3FjRWpwZ1RGV0o4cC90emVua3BwamRNQkFhci9RR0V2cFMpU3Z2SlgxL2d5TzJmWWwwvQjFyK1gyL2  
5OMDlCcXpsMFdaV0NiaHkrd2FOUVFOTDVsdVJZ2pyRnVhMnZwNGtVVVZNV2ZGTFRwL3V0YjJPM29US2x3VjlCNXJjZUNqd  
VHaUpPalJKTXRvZm1N1IHapqUmVybXI4d2IPNmpCcWJ6cVpBcHVGODQzKy9wLy9HMG56WIA4TWR6NkcxZytmN0d4UEln  
VG9GRmg5RnBXTjkwM2ZzdZjY1RYN245VXVUVdHHRVpTarnEbGVVSGVrV1prQUhWRHMrMDVvcTZkQnE4SEVWU0lWenN  
DY2RzQ2NRM2ZpYWs1SE0zREIvdktbHBMbnRFVldXVTQzZCswdUttVGQzalJUNnJSUXoza0xiVGkrTjB5MvHBSW9iQ0FXNU9B  
Tm4xZnhxUnVMaDZFMUZjYjM3YnNPeE1NVWVrazQ2aFdnUkZQK2FtUXk4dktamXAwNDQwU0MxaU9QN0VidIJmTjR6MXJ6OU  
ljbEdkU0RPRESBTDZkv3UxKzNuS0dQc29pNmNkbHN0RldiUXV2eW5Ic1dTM2IvbjZWaDFoL0VCZTU3d0tFc2RERnh6dHhEYzJ  
paU9VTnU1clVJm81YnI1alpnOEFaSzZENytXZmdiZ0hUbC80eDhOZUxSVGJORIZERUJvMEpTaIRQTdlyY0tHSHhsa21taWw3  
WVluN3hiK3BIT0ITS9GaltSk9YVEpxTzRpUjVoYWI3YS9FWnpPSHHdmdxa1VzSUJwWWRvcTV3Z1pzOWtnazNzcnhSekhlb1g  
zNmZHU051cFBIcmFiWTdZdHJtVzBucjBTaHJHJZDM4ZXJMEJuWENXc3F5UWtzS1RJQTJKU0NBuUhdnlCTGJLZlc5ayt0QVZ6  
cjJCNXZ2a1p3SUxwVkrNTWUWXRSELBSU10aHpOUjlyYjY6SXpxRjdtQ1hRSTdYMTNnRE1hOXZ2TnRvdFIZcTVS0Z2ZT241Uk  
JMT1czeDBhK25FRUVQb0JZBIY1YktReDlvdVdqY2FadGwxcY4SVRkSzEwblErbkRvVjhFbm9uVXNpNkZ2bmtMN0g0SEJNb0t  
BWEpDUjh5MzN0SEpJWmFKZjVRdUN3Si9FW4ycGtsa2JLZnpJY2JxR3FXd2Zla1RBNUYyZ3FIdHgyM3Z6N1g2VDZ3NW5vO  
WRpU1YUjIYfSUFJqUEs2UEFnUjVasZshML0tSMmFkTEI5VEFhMFkwWG5NZmtNNnMtMfPIU0t0SVpFNG5zQUlUMjdjanBjMVfJ  
aVRySFAxeS9IZXZYnQvWIIISjkyeTIySWpiWU2Q2ZGcWExbzhqZ013OGRpeVFGVHd1SIMxM3RLS3BzTUZxeW10WUg1aURr  
bVYzaUcyZ05ZMmM2WEIgaTkyYmZLYXZiY3RPSnFnNEZLekFkdG1ISktpSVJWN09CVmJsdFNKMDRUT3JzNk1xY1VpSHZaNAk9  
EV3grTgt5d2NFWeg3cEwrUS9iak52eXhieG1DT1oyd0NURTBacGhITU5LZUJJC3NMRWNiY2srTytEcXRUSktrVjh1VIVQeGwyZj  
Y0Y2xCSkpOaVhJTE5NV3hXNHFFbnFLM1JxdHN5eWI1tSkdoK2hWUVNra3ZCendCR1VKdkgrdTZSS0REazVORVJGUgnPN2tER  
DhpNTILU2dJakhik0VJQjZwWk8wUTBOBdNwbjHhZKvHQjRHMUVoV2t5YmZ4dnI0QXZxbWdSTzI3K2dEUHFZS3I2UE9KSmZ  
wYkEwem53ZFJ5czBsVh1vcUlrNGpydDd6MmVTWDB1RldSQ1YzL0NnM28yVFd5L2lvSENvcFdSTGhZYkY5K3BVOUTkZiIsImRh  
dGFRZXMkOiJBUIICQUhOQ9zYVcyZ1pOMDIXTnROR2tYzhxcDExeFN0Wi9kckVFb3kxSGs4TFhXZ0Y3eHBUVmw3d011SUtoU  
FICNmVxQmxBQUFBZmpCOEJna3Foa2IHOXcwQkQ3Y3WdiekJOQWdFQU1HZ0dDU3FHU0liM0RRRUhBVFEiQmdsZ2hrZ0JaUU1F  
QVM0d0VRUU1yYVBIQ01WTXFIzFRGTG9OQWdFUWdEdnRPT0YwZVFkVFJyWkxsMkRiQTBSTZLU0llc3RRS1V1U0tTb3NnaHB  
yM2x6RUVFeU9KScsyZldsN3k2NWp0RUxRZm1NYII0Vyt3WTNCQnc9PSIIsInZlcnNpb24iOiYiIiwidHlwZSI6ImRBEVFFS0VZiIiw  
XhwaXJhdGlvbIi6MTY4MzA0OTYzN30= <https://789534828835.dkr.ecr.ap-northeast-2.amazonaws.com>

96  
97 -받은 키를 이용하여 로그인하기  
98 \$ \$(aws ecr get-login --no-include-email --region ap-northeast-2)  
99 WARNING! Using --password via the CLI is insecure. Use --password-stdin.  
100 WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.  
101 Configure a credential helper to remove this warning. See  
102 <https://docs.docker.com/engine/reference/commandline/login/#credentials-store>  
103

104 Login Succeeded  
105

106 2) Docker Image 생성하기

107 - \$ pwd  
108 /home/ec2-user/kafka-client  
109

110 - \$ ls  
111 Dockerfile producer.py requirements.txt  
112

113 \$ docker build -t kafka-producer . <---마지막 . 주의  
114 Sending build context to Docker daemon 4.608kB  
115 Step 1/8 : FROM python:3  
116 3: Pulling from library/python  
117 b0248cf3e63c: Pull complete  
118 127e97b4daf7: Pull complete  
119 0336c50c9f69: Pull complete  
120 1b89f3c7f7da: Pull complete  
121 2d6277217976: Pull complete  
122 273fcd609d8: Pull complete  
123 58568d3a3a00: Pull complete  
124 56fc9fb54f6e: Pull complete  
125 8a22f29afe36: Pull complete  
126 Digest: sha256:f7382f4f9dbc51183c72d621b9c196c1565f713a1fe40c119d215c961fa22815  
127 Status: Downloaded newer image for python:3  
128 ---> 4665a951a37e  
129 Step 2/8 : ENV PYTHONUNBUFFERED 1  
130 ---> Running in caaaf385541f  
131 Removing intermediate container caaaf385541f  
132 ---> a8e0ddcf30c4  
133 Step 3/8 : RUN mkdir /kafka-client  
134 ---> Running in 97f450e4535c  
135 Removing intermediate container 97f450e4535c

```

136 ---> 330c0c532667
137 Step 4/8 : WORKDIR      /kafka-client
138 ---> Running in ba5a0f8690f3
139 Removing intermediate container ba5a0f8690f3
140 ---> 210140292efd
141 Step 5/8 : COPY      requirements.txt /kafka-client/
142 ---> ef04fbf7e137
143 Step 6/8 : RUN      pip install -r requirements.txt
144 ---> Running in 0b2ec4d5e3af
145 Collecting kafka-python==2.0.1
146   Downloading kafka_python-2.0.1-py2.py3-none-any.whl (232 kB)
147     _____ 232.2/232.2 kB 17.1 MB/s eta 0:00:00
148 Collecting protobuf3==0.2.1
149   Downloading protobuf3-0.2.1.tar.gz (10 kB)
150   Preparing metadata (setup.py): started
151   Preparing metadata (setup.py): finished with status 'done'
152 Building wheels for collected packages: protobuf3
153   Building wheel for protobuf3 (setup.py): started
154   Building wheel for protobuf3 (setup.py): finished with status 'done'
155   Created wheel for protobuf3: filename=protobuf3-0.2.1-py3-none-any.whl size=17491
   sha256=b3fac3dfc8d8495275d169e0c6caadffa72ac5f3478252f55add42298fa79599
   Stored in directory: /root/.cache/pip/wheels/63/c4/1c/423b1c72286b234f6dd1b4b4dbe85d93256119b407e2a89585
156 Successfully built protobuf3
157 Installing collected packages: protobuf3, kafka-python
158 Successfully installed kafka-python-2.0.1 protobuf3-0.2.1
159 WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system
160 package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv

161 [notice] A new release of pip available: 22.3.1 -> 23.1.2
162 [notice] To update, run: pip install --upgrade pip
163 Removing intermediate container 0b2ec4d5e3af
164 ---> 99a0a9ca53a5
165 Step 7/8 : COPY      producer.py /kafka-client/producer.py
166 ---> 955a1116dd6a
167 Step 8/8 : CMD      ["python", "./producer.py"]
168 ---> Running in 45ad295e4113
169 Removing intermediate container 45ad295e4113
170 ---> d15aee541dce
171 Successfully built d15aee541dce
172 Successfully tagged kafka-producer:latest
173
174
175 3)이미지 확인하기
176 $ docker images
177   REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
178 kafka-producer    latest   42ecda211c56   13 seconds ago  936MB
179 python            3        815c8c75dfc0   5 days ago    920MB
180
181
182 4)이미지에 tag 달기
183 $ docker tag kafka-producer:latest {Amazon ECR 리포지토리 URI}:latest
184 ex) $ docker tag kafka-producer:latest 789534828835.dkr.ecr.ap-northeast-2.amazonaws.com/msk/producer:latest
185
186 5>tag 확인하기
187 $ docker images
188   REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
189 789534828835.dkr.ecr.ap-northeast-2.amazonaws.com/msk/producer    latest   42ecda211c56   About a minute ago  936MB
190 kafka-producer          latest   42ecda211c56   About a minute ago  936MB
191 python                    3        815c8c75dfc0   5 days ago    920MB
192
193 6)ECR에 배포하기
194 $ docker push {Amazon ECR 리포지토리 URI}:latest
195 ex) $ docker push 789534828835.dkr.ecr.ap-northeast-2.amazonaws.com/msk/producer:latest
196 The push refers to repository [789534828835.dkr.ecr.ap-northeast-2.amazonaws.com/msk/producer]
197 53b9a455df5d: Pushed
198 b111ef97a510: Pushed
199 093b8aeb7f03: Pushed
200 57113875f15b: Pushed
201 1b38857d7bd5: Pushed
202 c63832a549a2: Pushed
203 67968e3386b6: Pushed
204 807e5e673844: Pushed
205 cfd0811d364e: Pushed
206 b86f260e173a: Pushed
207 6a1ebb98b0dc: Pushed
208 24b48387f467: Pushed
209 ae56c0c5405b: Pushed
210 latest: digest: sha256:e58cc7064810e51133d7f612325f8ab986fdc8b4ed267b8ef026046d8490a0dc size: 3051
211
212 7)ECR에서 Image 확인하기
213 -Amazon ECR > 리포지토리 > msk/producer에서 이미지 확인
214
215
216 8. ECS에서 이미지 사용을 위한 Task 정의 생성

```

```
217 1)[서비스] > [컨테이너] > [Elastic Container Service] 페이지로 이동
218 2)[Amazon Elastic Container Service] 페이지에서 좌측 메뉴 중 [클러스터] 선택
219 3)[클러스터] 페이지에서 [클러스터 생성] 버튼 클릭
220 4)[클러스터 생성] 페이지에서,
221   -[클러스터 구성] > [클러스터 이름] : {계정}-ecs-cluster
222   -[네트워킹]
223     --[VPC] : {계정}-datalake-vpc
224     --[서브넷] : 이미 2개가 선택되어 있음. 여기서 {계정}-datalake-subnet-2a만 선택
225   -나머지 값은 기본값 그대로 사용하기로 함.
226   -[생성] 버튼 클릭
227
228 5)태스크 정의 생성하기
229   -ECS 좌측 메뉴 중 [태스크 정의] 클릭
230   -[태스크 정의]페이지에서 [새 태스크 정의 생성] > [새 태스크 정의 생성] 선택
231   -[태스크 정의 및 컨테이너 구성] 페이지에서
232     --[태스크 정의 구성] > [태스크 정의 패밀리] : {계정}-producer-task
233     --[컨테이너 - 1]
234       ---[컨테이너 세부 정보] > [이름] : {계정}-kafka-producer-container
235       ---[컨테이너 세부 정보] > [이미지 URI] : {Amazon ECR 리포지토리 msk/producer의 이미지 URI}
236       ---나머지는 기본값 그대로
237       ---[다음] 버튼 클릭
238   -[환경, 스토리지, 모니터링 및 태그 구성] 페이지에서,
239     --[환경]
240       ---[앱 환경] : [AWS Fargate(서비스)]
241       ---[운영 체제/아키텍처] : Linux/X86_64
242       ---[CPU] : .25 vCPU
243       ---[메모리] : .5GB
244     --나머지는 기본값 그대로 사용
245     --[다음] 버튼 클릭
246   -[검토 및 생성] 페이지에서 [생성] 버튼 클릭
247
248
249 9. 생성한 태스크 정의로 태스크 생성 후 실행하기
250 1)Tabby에서 Consumer 실행
251   $ pwd
252   /home/ec2-user/kafka_2.12-2.8.1/bin
253
254   $ ./kafka-console-consumer.sh --bootstrap-server {MSK 클러스터 엔드포인트1},{MSK 클러스터 엔드포인트2} --topic {topic 이름}
255   ex)$ ./kafka-console-consumer.sh --bootstrap-server
256   b-2.henrymskcluster.jm4797.c4.kafka.ap-northeast-2.amazonaws.com:9092,b-1.henrymskcluster.jm4797.c4.kafka.ap-north
257   east-2.amazonaws.com:9092 --topic henry-topic
258   <----- Cursor 대기 중
259
260 2)태스크 실행하기
261   -방금 생성한 태스크 {계정}-producer-task:1 상세페이지에서 [배포] > [태스크 실행] 클릭
262   -[생성] 페이지에서
263     --[환경]
264       ---[기존 클러스터] : {계정}-ecs-cluster
265       ---[컴퓨팅 옵션] : [시작 유형] 선택
266       ---[시작 유형] : FARGATE
267       ---[플랫폼 버전] : LATEST
268       ---[배포 구성] > [애플리케이션 유형] > [태스크]
269     --[네트워킹]
270       ---[VPC] : {계정}-datalake-vpc
271       ---[서브넷] : {계정}-datalake-subnet-2a
272       ---[보안 그룹] > [기존 보안 그룹 선택] > {계정}-datalake-sg
273     --나머지 값은 기본값 그대로
274     --[생성] 클릭
275   -[태스크] 목록에서 방금 생성한 태스크 확인
276     --[원하는 상태] : 실행 중
277     --태스크의 링크 클릭
278     --[로그] 탭 클릭
279       ---Timestamp (Local)과 메시지, 그리고 컨테이너를 확인할 수 있다.
280       ---메시지에 sended data : {'num' : '0'} 부터 마지막 elapsed : 60.02159786224365까지 확인
281       ---역시 Tabby의 consumer.sh에도 동일한 결과가 출력됨을 확인
```