Lab3. Kafka Producer Application을 ECR에 배포하기

1. Docker Engine 설치하기
   -$ sudo yum update -y
   -$ sudo amazon-linux-extras install docker
   -$ sudo service docker start
   -$ sudo setfacl -m user:ec2-user:rw /var/run/docker.sock
   -$ docker -version
      Client:
       Version: 20.10.23
       API version: 1.41
       Go version: go1.18.9
       Git commit: 7155243
       Built: Tue Apr 11 22:56:36 2023
       OS/Arch: linux/amd64
       Context: default
       Experimental: true

      Server:
       Engine:
        Version: 20.10.23
        API version: 1.41 (minimum version 1.12)
        Go version: go1.18.9
        Git commit: 6051f14
        Built: Tue Apr 11 22:57:17 2023
        OS/Arch: linux/amd64
        Experimental: false
       containerd:
        Version: 1.6.19
        GitCommit: 1e1ea6e986c6c86565bc33d52e34b81b3e2bc71f
       runc:
        Version: 1.1.4
        GitCommit: 5fd4c4d144137e991c4acebb2146ab1483a97925
       docker-init:
        Version: 0.19.0
        GitCommit: de40ad0

2. requirements.txt 생성하기
   $ pwd
   /home/ec2-user/kafka-client

   $ nano requirements.txt

3. Dockerfile 생성하기
   $ pwd
   /home/ec2-user/kafka-client

   $nano Dockerfile

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

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 엔터티 선택] 페이지에서,
      -[신뢰할 수 있는 엔터티 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] : henry-ecr-admin-role
      -[역할 생성] 버튼 클릭

8)다시 [IAM 역할 수정] 페이지로 돌아와서,
-[IAM 역할] : 방금 생성한 henry-ecr-admin-role (보이지 않으면 리프레쉬 버튼 클릭)
-[IAM 역할 업데이트] 버튼 클릭

7. AWS ECR에 이미지 배포하기
1)ECR에 로그인하기
-Visual Studio Code의 Terminal에서 다음의 명령으로 로그인한다.
$ aws ecr get-login --no-include-email --region ap-northeast-2
docker login -u AWS -p
eyJwYXlsb2FkIjoiZmJxSlEvUDdocGFsclVWcU44djQ5U3c5NGdLQWRFUGdmWWRwbFYzazdSbVlic0RWTG9KcDdMeDZjMjE2K
1dkY0lnZDZpS1B6NEt3QXBMRHkva2ZpbnJaTExDcEhwTm4yVnRvNzlEUVJjelovM084VEVzZFBQV0Rqd0JPNkZIVktKeFh3aWZ
0UUlFaGJ2ZWlDQzdyNUhlZFU3WDVUSklHSWhEOXNwTTdTT1hvR1lGQ3hHR1VEQy9INU5aYzVXWUlUVjdmRW4wbVdHbWhM
UkdnYkRuaEtZRk01c3JxcHgwcTNPLzlSR3d0ci80VkdOWCt2Y0xTZjRtTjB5SldVOHBUWE5FemxXMlpXYktCdWVDaVdMNVA5ZI
ZEdjlqeElEaDRKaG9kb1RNemNCWkU5YmpHeGhnR2NiNE1KR3VEbmwwUTM3Q2tXWXRMdFg3YU85Nk4rMXZPV0J4WktJWW
5OS0p0azUwZldNeG5XMkhTMmlrbWIwSnpHTk95Wlh0MFBsdXNlVXNzSzg0aWNYVTFoZXRUQ0NpNlVFQnlxYnY1QmhyUW1M
UDBsbDNnZmlDbzlNZTI1VWVwK1hGbnRZM1RWNlFyak9pTFpFa3RqVzIrV3hMalBCbFJ4VW5ySFdJbkNMMEpGdUdmRUdZN1
o3YVFUcTRtMXVhQVlJQXI5YmJmb21GajdsNGRVSzJ5ZGZxSkJ3clNzUWJlb2Q4YUsyYmlsa0JJNVpmUFlqcVIzRUZEdnJGdWoy
SEpjRTRhS05yZkVMM3FjRWpwZ1RGV0o4cC90emVua3BwamRNQkFhci9RR0V2cFMyU3Z2SlgxL2d5TzJMWWwvQjFyK1gyL2
5OMDlCcXpsMFdaV0NiaHkrd2FOUVFOTDVsdVJVZ2pyRnVhMnZwNGtVVVZNV2ZGTFRwL3V0YjJPM29US2x3VjlCNXJjZUNqdn
VHaUpPalJKTXRxVzM1N1lHaWpqUmVybXl4d2lPNmpCcWJ6cVpBcHVGODQzKy9wLy9HMG56WlA4TWR6NkcxZytmN0d4UEln
VG9GRmg5RnBXTjkwM2ZzdzJnY1RYN245VXVUVDhHRVpTamNEbGVVSGVrV1prQUhWRHMrMDVVcTZkQnE4SEVWU0lWenN
DY2RzQ2NRM2ZpYWs1SE0zREIvdktrbHZMbnRFVldXVTQzZCswdUttVGQzalJUNnJSUXoza0xiVGkrTjB5MVhBSW9iQ0FXNU9B
Tm4xZnhxUnVMaDZFMUZJYjM3YnNPeE1NVWVrazQ2aFdnUkZQK2FtUXk4dktaMXAwNDQwU0MxaU9QN0VidlJmTjR6MXJ6OU
ljbEdkU0RPRE5BTDZkV3UxKzNuS0dQc29pNmNkbHN0RldiUXV2eW5Ic1dTM2IvbjZWaDFoL0VCZTU3d0tFc2RERnh6dHhEYzJ
paU9VTnU1clVJVm81YnI1alpnOEFaSzZENytxZmdiZ0hUbC80eDhOZUxSVGJORlZERUJvMEpTalRQTDlyY0tHSHhsa21taWw3
WVluN3hiK3BlT0lITS9GazltSk9YVEpxTzRpUjVoYWl3YS9FWnpPSHhHdmdxa1VzSUJwWWRvcTV3Z1pzOWtnazNzcnhSekhlb1g
zNmZHU051cFBIdmFiWTdZdHJtVzBucjBTaHJHZDM4ZXJJMEJuWENXc3F5UWtzS1RJQTJKU0NBeUhwdnlCTGJLZlc5ayt0QVZ6
cjJCNXZ2a1p3SUxwVkRNTWlUWXRSelBKSUI0aHpOUjlhYjJ6SXpxRjdtQ1hRSTdYMTNnRE1hOXZ2TnRvdFlZcTVSZ0ZZT241Uk
JMT1czeDBhK25FRUVQb0JBZlY1YktReDlvdVdqY2FadGwxcmY4SVRkSzEwblErbkRvVjhFbm9uVXNpNkZ2bmtMN0g0SEJNb0t
BWEpDUjh5MzNOSEpJWmFKZjVRdUN3Si9FVW4ycGtsa2JLZnpJY2JxR3FXd2Zla1RBNUYyZ3FIdHgyM3Z6N1g2VDZ3NW5vO
WRpU1lYUjllYlFSUFJqUEs2UEFnUlVaSzhML0tSMmFkTEI5VEFhMFkwWG5NZmtNNnNtMFpIU0t0SVpFNG5zQUlUMjdjanBjMVFJ
aVRySFAxeS9IZXZYNjQvWlllSjkyeTIySWpiWlU2Q2ZGcWExbzhqZ013OGRpeVFGVHd1SlMxM3RLS3BzTUZxeW10WUg1aURr
bVYzaUcyZ05ZZmM2WElGaTkxYmZLYXZiY3RPSnFnNEZLekFkdG1lSktpSVJWN09CVmJsdFNKMDRUT3JzNk1xY1VpSHZaNk9
EV3grTGt5d2NFWEg3cEwrUS9iak52eXhieG1DT1oyd0NURTBacGhITU5LZUJJc3NMRWNIY2srTytEcXRUSktrVjh1VlVQeGwyZj
Y0Y2xCSkpOaVhJTE5NV3hXNHFFbnFLM1JxdHN5eWI1SkdoK2hWUVNra3ZCendCR1VKdkgrdTZSS0REazVORVJGUGNPN2tER
DhpNTlLU2dJakhiK0VJQjZwWk8wUTBObDNwbjhHZkVHQjRHMUVoV2t5YmZ4dnI0QXZxbWdSTzI3K2dEUHFZS3I2UE9KSmZ
wYkEwem53ZFJ5czBsVk1vcUlrNGpydDd6MmVTWDB1RldSQ1YzL0NnM28yVFd5L2lvSENvcFdSTGhZYkY5K3BVOUtkZiIsImRh
dGFrZXkiOiJBUUlCQUhoQU9zYVcyZ1pOMDlXTnROR2tZYzhxcDExeFNoWi9kckVFb3kxSGs4TFhXZ0Y3eHBUVmw3d011SUtoU
FlCNmVxQmxBQUFBZmpCOEJna3Foa2lHOXcwQkJ3YWdiekJ0QWdFQU1HZ0dDU3FHU0liM0RRRUhBVEFlQmdsZ2hrZ0JaUU1F
QVM0d0VRUU1yYVBlQ01WTXFlZFRGTG9OQWdFUWdEdnRPT0YwZVFkVFJyWkxsMkRiQTBSTzZLU0llc3RRS1V1U0tTb3NnaHB
yM2x6RUVFeU9kSCsyZldsN3k2NWp0RUxRZm1NYll0Vyt3WTNCQnc9PSIsInZlcnNpb24iOiIyIiwidHlwZSI6IkRBVEFfS0VZIiwiZ
XhwaXJhdGlvbiI6MTY4MzA0OTYzN30= https://789534828835.dkr.ecr.ap-northeast-2.amazonaws.com

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

Login Succeeded

2)Docker Image 생성하기
-$ pwd
/home/ec2-user/kafka-client

-$ ls
Dockerfile producer.py requirements.txt

$ docker build -t kafka-producer . <---마지막 . 주의
Sending build context to Docker daemon 4.608kB
Step 1/8 : FROM python:3
3: Pulling from library/python
b0248cf3e63c: Pull complete
127e97b4daf7: Pull complete
0336c50c9f69: Pull complete
1b89f3c7f7da: Pull complete
2d6277217976: Pull complete
273fcda609d8: Pull complete
58568d3a3a00: Pull complete
56fc9fb54f6e: Pull complete
8a22f29afe36: Pull complete
Digest: sha256:f7382f4f9dbc51183c72d621b9c196c1565f713a1fe40c119d215c961fa22815
Status: Downloaded newer image for python:3
 ---> 4665a951a37e
Step 2/8 : ENV PYTHONUNBUFFERED 1
 ---> Running in caaaf385541f
Removing intermediate container caaaf385541f
 ---> a8e0ddcf30c4
Step 3/8 : RUN mkdir /kafka-client
 ---> Running in 97f450e4535c
Removing intermediate container 97f450e4535c

```
 ---> 330c0c532667
Step 4/8 : WORKDIR /kafka-client
 ---> Running in ba5a0f8690f3
Removing intermediate container ba5a0f8690f3
 ---> 210140292efd
Step 5/8 : COPY requirements.txt /kafka-client/
 ---> ef04fbf7e137
Step 6/8 : RUN pip install -r requirements.txt
 ---> Running in 0b2ec4d5e3af
Collecting kafka-python==2.0.1
  Downloading kafka_python-2.0.1-py2.py3-none-any.whl (232 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.2/232.2 kB 17.1 MB/s eta 0:00:00
Collecting protobuf3==0.2.1
  Downloading protobuf3-0.2.1.tar.gz (10 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Building wheels for collected packages: protobuf3
  Building wheel for protobuf3 (setup.py): started
  Building wheel for protobuf3 (setup.py): finished with status 'done'
  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
Successfully built protobuf3
Installing collected packages: protobuf3, kafka-python
Successfully installed kafka-python-2.0.1 protobuf3-0.2.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system
package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv

[notice] A new release of pip available: 22.3.1 -> 23.1.2
[notice] To update, run: pip install --upgrade pip
Removing intermediate container 0b2ec4d5e3af
 ---> 99a0a9ca53a5
Step 7/8 : COPY producer.py /kafka-client/producer.py
 ---> 955a1116dd6a
Step 8/8 : CMD ["python", "./producer.py"]
 ---> Running in 45ad295e4113
Removing intermediate container 45ad295e4113
 ---> d15aee541dce
Successfully built d15aee541dce
Successfully tagged kafka-producer:latest
```

3)이미지 확인하기

```
$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED         SIZE
kafka-producer  latest   d15aee541dce  11 minutes ago  937MB
python          3        4665a951a37e  2 weeks ago     921MB
```

4)이미지에 tag 달기

```
$ docker tag kafka-producer:latest {Amazon ECR 리포지토리 URI}:latest
ex) $ docker tag kafka-producer:latest 789534828835.dkr.ecr.ap-northeast-2.amazonaws.com/msk/producer:latest
```

5)tag 확인하기

```
$ docker images
REPOSITORY                                                TAG      IMAGE ID      CREATED         SIZE
789534828835.dkr.ecr.ap-northeast-2.amazonaws.com/msk/producer  latest   d15aee541dce  15 minutes ago  937MB
kafka-producer                                            latest   d15aee541dce  15 minutes ago  937MB
python                                                    3        4665a951a37e  2 weeks ago     921MB
```

6)ECR에 배포하기

```
$ docker push 789534828835.dkr.ecr.ap-northeast-2.amazonaws.com/msk/producer:latest
The push refers to repository [789534828835.dkr.ecr.ap-northeast-2.amazonaws.com/msk/producer]
53b9a455df5d: Pushed
b111ef97a510: Pushed
093b8aeb7f03: Pushed
57113875f15b: Pushed
1b38857d7bd5: Pushed
c63832a549a2: Pushed
67968e3386b6: Pushed
807e5e673844: Pushed
cfd0811d364e: Pushed
b86f260e173a: Pushed
6a1ebb98b0dc: Pushed
24b48387f467: Pushed
ae56c0c5405b: Pushed
latest: digest: sha256:e58cc7064810e51133d7f612325f8ab986fdc8b4ed267b8ef026046d8490a0dc size: 3051
```

7)ECR에서 Image 확인하기

-Amazon ECR > 리포지토리 > msk/producer에서 이미지 확인

8. ECS에서 이미지 사용을 위한 Task 정의 생성

1)[서비스] > [컨테이너] > [Elastic Container Service] 페이지로 이동

2)[Amazon Elastic Container Service] 페이지에서 좌측 메뉴 중 [클러스터] 선택

3)[클러스터] 페이지에서 [클러스터 생성] 버튼 클릭
4)[클러스터 생성] 페이지에서,
-[클러스터 구성] > [클러스터 이름] : {계정}-ecs-cluster
-[네트워킹]
--[VPC] : {계정}-datalake-vpc
--[서브넷] : 이미 2개가 선택되어 있음. 여기서 {계정}-datalake-subnet-2a만 선택
-나머지 값은 기본값 그대로 사용하기로 함.
-[생성] 버튼 클릭

5)태스크 정의 생성하기
-ECS 좌측 메뉴 중 [태스크 정의] 클릭
-[태스크 정의]페이지에서 [새 태스크 정의 생성] 버튼 클릭
-[태스크 정의 및 컨테이너 구성] 페이지에서
--[태스크 정의 구성] > [태스크 정의 패밀리] : {계정}-producer-task
--[컨테이너 - 1]
---[컨테이너 세부 정보] > [이름] : {계정}-kafka-producer-container
---[컨테이너 세부 정보] > [이미지 URI] : {Amazon ECR 리포지토리 msk/producer의 이미지 URI}
---나머지는 기본값 그대로
---[다음] 버튼 클릭
-[환경, 스토리지, 모니터링 및 태그 구성] 페이지에서,
--[환경]
---[앱 환경] : [AWS Fargate(서비리스)
---[운영 체제/아키텍처] : Linux/X86_64
---[CPU] : .25 vCPU
---[메모리] : .5GB
--나머지는 기본값 그대로 사용
--[다음] 버튼 클릭
-[검토 및 생성] 페이지에서 [생성] 버튼 클릭

9. 생성한 태스크 정의로 태스크 생성 후 실행하기
1)Tabby에서 Consumer 실행
$ ./kafka-console-consumer.sh --bootstrap-server {MSK 클러스터 엔드포인트1},{MSK 클러스터 엔드포인트2} --topic {topic 이름}
ex)$ ./kafka-console-consumer.sh --bootstrap-server b-2.henrymskcluster.jm4797.c4.kafka.ap-northeast-2.amazonaws.com:9092,b-1.henrymskcluster.jm4797.c4.kafka.ap-northeast-2.amazonaws.com:9092 --topic henry-topic
<---- Cursor 대기 중

2)태스크 실행하기
-방금 생성한 태스크 {계정}-producer-task:1 상세페이지에서 [배포] > [태스크 생성] 클릭
-[생성] 페이지에서
--[환경]
---[기존 클러스터] : {계정}-ecs-cluster
---[컴퓨팅 옵션] : [시작 유형] 선택
---[시작 유형] : FARGATE
---[플랫폼 버전] : LATEST
---[배포 구성] > [애플리케이션 유형] > [태스크]
--[네트워킹]
---[VPC] : {계정}-datalake-vpc
---[서브넷] : {계정}-datalake-subnet-2a
---[보안 그룹] > [기존 보안 그룹 선택] > {계정}-datalake-sg
--나머지 값은 기본값 그대로
--[생성] 클릭
-[태스크] 목록에서 방금 생성한 태스크 확인
--[원하는 상태] : 실행 중
--태스크의 링크 클릭
--[로그] 탭 클릭
---Timestamp (Local)과 메시지, 그리고 컨테이너를 확인할 수 있다.
---메시지에 sended data : {'num' : '0'} 부터 마지막 elpased : 60.02159786224365까지 확인
---역시 Tabby의 consumer.sh에도 동일한 결과가 출력됨을 확인