

# CNA Final Assessment - 1일차

☰ Tags	2020 DT 교육
▼ 상태	완료
📅 시작	@Aug 31, 2020
📅 종료	@Aug 31, 2020

## 1일차

### 1. 바로 잡습니다.

1. CQRS는 서비스 내에 구현하면 ID가 중복 생성되므로, ContextBoundary 밖에 별도의 서비스로 구현 해야함
  - BookingList 관련 클래스는 모두 삭제한 후 commit 했음
2. 각 이벤트에 메시지 퍼블리싱 기능이 내장이 되어 있어, 추가로 코딩한 부분에서 메시지를 중복 퍼블리싱함... 해당 코드들을 전부 삭제함

```
st_case.txt ● Booking.java x AbstractEvent.java application.yml Terminal 9
11 @GeneratedValue(strategy=GenerationType.AUTO)
12 private Long id;
13 private Long roomId;
14 private String useStartDtm;
15 private String useEndDtm;
16 private String bookingUserId;
17
18 @PostPersist
19 public void onPostPersist(){
20
21     // 이벤트 인스턴스 생성
22     BookingCreated bookingCreated = new BookingCreated();
23
24     // 속성값 할당
25     BeanUtils.copyProperties(this, bookingCreated);
26     bookingCreated.publishAfterCommit();
27
28 }
29
30 @PostUpdate
31 public void onPostUpdate(){
32
33     // 이벤트 인스턴스 생성
34     BookingChanged bookingChanged = new BookingChanged();
35
36     // 속성값 할당
37     BeanUtils.copyProperties(this, bookingChanged);
38     bookingChanged.publishAfterCommit();
39
40 }
41
```

- 위 빨간색 표시한 부분이 kafka에 메시지를 발행

```
case.txt ● Booking.java AbstractEvent.java x application.yml Terminal 9 Terminal 10
39
40 public void publish(String json){
41     if( json != null ){
42
43         /**
44          * spring streams 발식
45          */
46         KafkaProcessor processor = BookingApplication.applicationContext.getBean(KafkaProcessor.class);
47         MessageChannel outputChannel = processor.outboundTopic();
48
49         outputChannel.send(MessageBuilder
50             .withPayload(json)
51             .setHeader(MessageHeaders.CONTENT_TYPE, MimeTypeUtils.APPLICATION_JSON)
52             .build());
53     }
54 }
55
56
```

- AbstractEvent 클래스의 publish 내에 메시지 생성 로직이 들어 있음

## 2. 추가 작업 내역

### 1. EKS 클러스터(TeamE)에 Kafka 설치 완료

```
Every 2.0s: kubectl get all -n kafka

NAME                                READY   STATUS    RESTARTS   AGE
pod/my-kafka-0                      1/1     Running   2           10m
pod/my-kafka-1                      1/1     Running   0           8m16s
pod/my-kafka-2                      1/1     Running   0           6m56s
pod/my-kafka-zookeeper-0           1/1     Running   0           10m
pod/my-kafka-zookeeper-1           1/1     Running   0           9m35s
pod/my-kafka-zookeeper-2           1/1     Running   0           8m39s

NAME                                TYPE                      CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/my-kafka                    ClusterIP                10.100.8.43     <none>        9092/TCP         10m
service/my-kafka-headless            ClusterIP                None            <none>        9092/TCP         10m
service/my-kafka-zookeeper           ClusterIP                10.100.112.231 <none>        2181/TCP         10m
service/my-kafka-zookeeper-headless ClusterIP                None            <none>        2181/TCP,3888/TCP,2888/TCP 10m

NAME                                READY   AGE
statefulset.apps/my-kafka            3/3     10m
statefulset.apps/my-kafka-zookeeper 3/3     10m
```

- curl  
<https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3> | bash
- kubectl --namespace kube-system create sa tiller
- kubectl create clusterrolebinding tiller --clusterrole cluster-admin --serviceaccount=kube-system:tiller
- helm repo add incubator  
<http://storage.googleapis.com/kubernetes-charts-incubator>

### 2. EKS 클러스터(TeamE)에 Istio 설치 완료

```
Every 2.0s: kubectl get pod -n istio-system

NAME                                READY   STATUS    RESTARTS   AGE
grafana-584949b9c6-k8flg           1/1     Running   0           7m
istio-citadel-794467d89f-l9hws      1/1     Running   0           6m59s
istio-egressgateway-577d8dcd7d-7qpkz 1/1     Running   0           7m
istio-galley-5959f56ddf-c7j7f       1/1     Running   0           7m1s
istio-grafana-post-install-1.4.5-tlxlf 0/1     Completed 0           7m5s
istio-ingressgateway-7c54fb8897-s7zsx 1/1     Running   0           7m
istio-pilot-5cfbc6b7f5-lw7j2        2/2     Running   1           7m
istio-policy-69559fadb-x5pp7        2/2     Running   2           7m
istio-security-post-install-1.4.5-q6t9s 0/1     Completed 0           7m4s
istio-sidecar-injector-7b77958798-vlwzg 1/1     Running   0           6m59s
istio-telemetry-764668f5cd-w566c     2/2     Running   1           7m
istio-tracing-795c9c64c4-ppbzm       1/1     Running   0           6m59s
kiali-7d4cf866cc-b94vn              1/1     Running   0           7m
prometheus-8685f659f-xnlfh          1/1     Running   0           7m
```

- curl -L <https://git.io/getLatestIstio> | ISTIO\_VERSION=1.4.3 sh -
- cd istio-1.4.3

- export PATH=\$PWD/bin:\$PATH
- for i in install/kubernetes/helm/istio-init/files/crd\*.yaml; do kubectl apply -f \$i; done
- kubectl apply -f install/kubernetes/istio-demo.yaml

### 3. Booking 서비스를 EKS 클러스터에 배포

- /kubernetes/deployment.yml 생성

```

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: booking
5    labels:
6      app: booking
7  spec:
8    replicas: 1
9    selector:
10     matchLabels:
11       app: booking
12  template:
13    metadata:
14     labels:
15       app: booking
16    spec:
17     containers:
18     - name: booking
19       image: 052937454741.dkr.ecr.ap-northeast-2.amazonaws.com/team-e-booking:latest # ECR image URI
20       ports:
21       - containerPort: 8080
22       readinessProbe:
23         httpGet:
24           path: '/actuator/health'
25           port: 8080
26         initialDelaySeconds: 10
27         timeoutSeconds: 2
28         periodSeconds: 5
29         failureThreshold: 10
30       livenessProbe:
31         httpGet:
32           path: '/actuator/health'
33           port: 8080
34         initialDelaySeconds: 120
35         timeoutSeconds: 2
36         periodSeconds: 5
37         failureThreshold: 5

```

- 다른 서비스에 적용하려면 "booking" 부분을 서비스에 맞는 이름으로 변경하면 됨
- image 값을 "ECR URI:TagName" 으로 변경
- /kubernetes/service.yml 생성

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: booking
5    labels:
6      app: booking
7  spec:
8    ports:
9    - port: 8080
10     targetPort: 8080
11  selector:
12    app: booking

```

- buildspec.yml 변경

```

buildspec.yml x  commands.txt  Terminal 9  Terminal 10  Terminal 11
1  version: 0.2
2
3  phases:
4  install:
5    runtime-versions:
6      java: corretto8 # Amazon Corretto 8 - production-ready distribution of the OpenJDK
7      docker: 18
8
9    commands:
10     - curl -o kubect1 https://amazon-eks.s3.us-west-2.amazonaws.com/1.16.12/2020-07-08/bin/linux/amd64/kubect1 # Download kubect1
11     - chmod +x ./kubect1
12     - mkdir ~/.kube
13     - aws eks --region $AWS_DEFAULT_REGION update-kubeconfig --name TeamE # Set cluster TeamE as default cluster
14
15  pre_build:
16    commands:
17     - echo Region = $AWS_DEFAULT_REGION # Check Environment Variables
18     - echo Account ID = $AWS_ACCOUNT_ID # Check Environment Variables
19     - echo ECR Repo = $IMAGE_REPO_NAME # Check Environment Variables
20     - echo Docker Image Tag = $IMAGE_TAG # Check Environment Variables
21     - echo Logging in to Amazon ECR...
22     - $(aws ecr get-login --no-include-email --region $AWS_DEFAULT_REGION) # Login ECR
23
24  build:
25    commands:
26     - echo Build started on `date`
27     - echo Building the Docker image...
28     - mvn clean
29     - mvn package -Dmaven.test.skip=true # Build maven
30     - docker build -t $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG . # Build docker image
31
32  post_build:
33    commands:
34     - echo Build completed on `date`
35     - echo Pushing the Docker image...
36     - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG # Push docker image to ECR
37     - echo Deploy service into EKS
38     - kubect1 apply -f ./kubernetes/deployment.yml # Deploy
39     - kubect1 apply -f ./kubernetes/service.yml # Service

```

- install.commands 추가 부분: 빌드 환경에 kubect1 설치
- post\_build.commands 추가 부분: deployment.yml, service.yml 파일로 서비스 배포

#### 4. EKS 배포를 위한 CodeBuild 권한 추가

- CodeBuild에 EKS 권한 추가

```

41  {
42  }
43  "Action": [
44    "ecr:BatchCheckLayerAvailability",
45    "ecr:CompleteLayerUpload",
46    "ecr:GetAuthorizationToken",
47    "ecr:InitiateLayerUpload",
48    "ecr:PutImage",
49    "ecr:UploadLayerPart",
50    "eks:DescribeCluster"
51  ],
52  "Resource": "*",
53  "Effect": "Allow"
54  }
55 }

```

- EKS의 역할에 CodeBuild 서비스 역할 추가
- EKS의 ConfigMap 다운로드

- `kubectl get configmaps aws-auth -n kube-system -o yaml`  
`> aws-auth.yml`
- ConfigMap 에 CodeBuild 서비스 역할 추가

```
aws-auth.yml x  commands.txt  buildspec.yml  Terminal 9  Terminal 10  Terminal 11
1  apiVersion: v1
2  data:
3    mapRoles: |
4      - groups:
5        - system:bootstrappers
6        - system:nodes
7        rolearn: arn:aws:iam::052937454741:role/eksctl-TeamE-nodegroup-standard-w-NodeInstanceRole-GXDMGJLPHR40
8        username: system:node:{{EC2PrivateDNSName}}
9      - rolearn: arn:aws:iam::052937454741:role/CodeBuildServiceRoleForTeamE
10      username: CodeBuildServiceRoleForTeamE
11      groups:
12        - system:masters
13    mapUsers: |
14      []
15  kind: ConfigMap
16  metadata:
17    creationTimestamp: "2020-08-31T09:06:31Z"
18    name: aws-auth
19    namespace: kube-system
20    resourceVersion: "854"
21    selflink: /api/v1/namespaces/kube-system/configmaps/aws-auth
22    uid: cf038f09-ab94-4b60-9937-33acc0be86d8
23
```

- EKS에 변경된 ConfigMap 적용

```
root@labs--1475502367:~/cna-booking/kubernetes# kubectl apply -f aws-auth.yml --force
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
configmap/aws-auth configured
```

- `kubectl apply -f aws-auth.yml --force`

## 5. CodeBuild 성능 향상을 위한 캐시 추가

- buildspec.yml 에 cache 설정을 해 줌

```
buildspec.yml x  Terminal 9  Terminal 10
32
33
34
35
36
37  cache:
38    paths:
39      - '/root/.m2/**/*'
```

- 캐시 설정을 하여 빌드에 필요한 dependency 들을 매번 다운로드 하지 않고 S3 버킷에 저장해 둘 수 있음

## 아티팩트 편집

### 아티팩트

아티팩트 추가

#### 아티팩트 1 - 기본

유형

아티팩트 없음

테스트를 실행하거나 도커 이미지를 Amazon ECR에 넣는 경우 [아티팩트 없음]을 선택할 수 있습니다.

#### 서비스 역할 권한

☒ 이 서비스 역할을 이 빌드 프로젝트에 사용할 수 있도록 AWS CodeBuild에서 수정하도록 허용  
arn:aws:iam::052937454741:role/CodeBuildServiceRoleForTeamE

#### ▼ 추가 구성

캐시, 암호화 키

##### 암호화 키 - 선택 사항

이 빌드의 출력 아티팩트를 암호화하는 데 사용되는 AWS KMS 고객 마스터 키를 제공합니다. 기본값은 S3에 대한 AWS 관리형 고객 마스터 키입니다.

arn:aws:kms:ap-northeast-2:052937454741:alias/aws/s3

arn:aws:kms:<region-ID>:<account-ID>:key/<key-ID>

##### 캐시 유형

Amazon S3

##### 캐시 버킷

team-e-build-cache

##### 캐시 경로 접두사 - 선택 사항

cache

##### 캐시 수명 주기 - 선택 사항

경로 접두사에 따라 캐시 버킷 내 전체 또는 일부 객체에 수명 주기 만료 작업을 적용할 수 있습니다.

+ 만료 추가

취소

아티팩트 업데이트

- 캐싱 이후 빌드 속도가 현저히 빨라짐을 확인할 수 있음: 4분 59초 → 1분 40초

□ 빌드 설명	상태	빌드 번호	소스 버전	제출자	기간	완료됨
TeamE-booking:55e4c18f-7afe-40d5-a9b1-42e90fcbf380	성공함	13	-	TeamE	1분 40초	방금
TeamE-booking:06de8537-666c-4274-8604-8069fcbf380	성공함	12	-	TeamE	4분 59초	3분 전