2024 전국기능경기대회 채점기준

1. 채점상의 유의사항	직 종 명	클라우드컴퓨팅

※ 다음 사항을 유의하여 채점하시오.

1) AWS의 지역은 ap-northeast-2을 사용합니다.
2) 웹페이지 접근은 크롬이나 파이어폭스를 이용합니다.
3) 웹페이지에서 언어에 따라 문구가 다르게 보일 수 있습니다.
4) shell에서의 명령어의 출력은 버전에 따라 조금 다를 수 있습니다.
5) 문제지와 채점지에 있는 <> 는 변수입니다. 해당 부분을 변경해 입력합니다.
6) 채점은 문항 순서대로 진행해야 합니다.
7) 삭제된 채점자료는 되돌릴 수 없음으로 유의하여 진행하며, 이의신청까지 완료 이후 선수가 생성한 클라우드 리소스를 삭제합니다.
8) 부분 점수가 있는 문항은 채점 항목에 부분 점수가 적혀져 있습니다.
9) 부분 점수가 따로 없는 문항은 모두 맞아야 점수로 인정됩니다.
10) 리소스의 정보를 읽어오는 채점항목은 기본적으로 스크립트 결과를 통해 채점을 진행하며, 만약 선수가 이의가 있다면 명령어를 직접 입력하여 확인해볼 수 있습니다.
11) [ ] 기호는 채점에 영향을 주지 않습니다.
12) 명령어 입력 Box 안의 명령줄은 한 줄 명령어입니다. 별도의 지시가 없으면 수정 없이 박스 안의 전체 내용을 복사하고 쉘에 붙여넣어 명령을 실행합니다.
13) (예상 출력)은 바로 이전 (명령어 입력)의 예상 출력을 의미합니다.

2. 채점기준표

1) 주요항목별 배점

직 종 명	클라우드컴퓨팅

과제 번호	일련 번호	주요항목	배점	채점방법		채점시기		비고
				독립	합의	경기 진행중	경기 종료후	
제1과제	1	네트워크 구성	3.5		○		○	
	2	Bastion 서버	1		○		○	
	3	관계형 데이터베이스	1		○		○	
	4	비관계형 데이터베이스	1		○		○	
	5	웹 애플리케이션	4.5		○		○	
	6	컨테이너 오케스트레이션	4		○		○	
	7	로드밸런서	2.5		○		○	
	8	S3	2		○		○	
	9	CloudFront	4		○		○	
	10	로그	2.5		○		○	
	11	모니터링	1.5		○		○	
	12	보안	2.5		○		○	
합 계			30					

2) 채점방법 및 기준

과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
1과제	1	네트워크 구성	1	VPC, Subnet	1.0
			2	Routing	1.0
			3	VPC Endpoint	1.5
	2	Bastion 서버	1	Bastion configuration	1.0
	3	관계형 데이터베이스	1	RDS configuration	1.0
	4	비관계형 데이터베이스	1	DynamoDB configuration	1.0
	5	웹 애플리케이션	1	Customer, Product perform	1.5
			2	Order perform	1.5
			3	Service ENV External Secrets	1.5
	6	컨테이너 오케스트레이션	1	Addon, App nodegroup configuration	1.0
			2	Service pod place	1.5
			3	CoreDNS Pod Fargate	1.5
	7	로드밸런서	1	Nginx Ingress Controller configuration	1.0
			2	NLB rules	1.5
	8	S3	1	S3 Bucket configuration	1.0
			2	S3 Bucket encryption	1.0
	9	CloudFront	1	Cloudfront to NLB	1.5
			2	Cloudfront to S3	1.5
			3	Cloudfront redirect	1.0
	10	로그	1	Logging configuration	1.0
			2	Service log 조회	1.5
	11	모니터링	1	Monitoring 동작 확인	1.5
	12	보안	1	WAF HTTP Method	1.5
			2	WAF QueryString	1.0
	총점				30

3) 채점내용

순번	사전준비
0	1) Bastion 서버에 SSH를 통해 접근합니다. (별도 명시가 없는 경우 모든 채점은 Bastion 서버에서 진행합니다.) 2) Bastion 명령어 및 권한을 확인합니다. (awscli permission, jq, curl, awscli region) 3) marking.sh에 script 파일 넣고 chmod +x marking.sh 후에 ./marking.sh로 실행하기 export DistributionID="<Cloudfront_Distribution_ID>" export S3_BUCKET="skills-static-<4words>" export CF_DOMAIN=$(aws cloudfront get-distribution --id ${DistributionID} --query "Distribution.DomainName" \| sed s/₩"//g) aws configure set default.region ap-northeast-2 export InvalidationID=$(aws cloudfront create-invalidation --distribution-id ${DistributionID} --paths "/*" --query "Invalidation.Id" | sed s/₩"//g) aws cloudfront wait invalidation-completed --distribution-id ${DistributionID} --id ${InvalidationID}

순번	채점 항목	
1-1	1-1-A (명령어 입력)	aws ec2 describe-vpcs --filter Name=tag:Name,Values=skills-vpc --query "Vpcs[0].CidrBlock" ₩ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-app-a --query "Subnets[0].CidrBlock" ₩ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-app-b --query "Subnets[0].CidrBlock" ₩ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-public-a --query "Subnets[0].CidrBlock" ₩ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-public-b --query "Subnets[0].CidrBlock" ₩ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-data-a --query "Subnets[0].CidrBlock" ₩ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-data-b --query "Subnets[0].CidrBlock"
	1-1-A (예상 출력) **정확히 일치** **순서 중요**	"10.1.0.0/16" "10.1.0.0/24" "10.1.1.0/24" "10.1.2.0/24" "10.1.3.0/24" "10.1.4.0/24" "10.1.5.0/24"
1-2	1-2-A (명령어 입력)	aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-app-a-rt --query "RouteTables[].Routes[].NatGatewayId" | grep "nat-" | wc -l ₩ ; aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-app-b-rt --query "RouteTables[].Routes[].NatGatewayId" | grep "nat-" | wc -l ₩ ; aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-public-rt --query "RouteTables[].Routes[]" | grep "igw-" | wc -l ₩ ; aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-data-rt --query "RouteTables[].Routes[]" | grep -E "igw-|nat-" | wc -l
	1-2-A (예상 출력) **정확히 일치** **순서 중요**	1 1 1 0

순번	채점 항목	
1-3	1-3-A (명령어 입력)	aws ec2 describe-vpc-endpoints --query "VpcEndpoints[].ServiceName"
	1-3-A (예상 출력) **최소_내용 포함** **순서 무관**	["com.amazonaws.ap-northeast-2.ecr.dkr", "com.amazonaws.ap-northeast-2.ecr.api"]
2-1	2-1-A (명령어 입력)	aws ec2 describe-instances --filter Name=tag:Name,Values=skills-bastion --query "Reservations[0].Instances[0].InstanceType" ₩ ; aws ec2 describe-subnets --subnet-ids $(aws ec2 describe-instances --filters "Name=tag:Name,Values=skills-bastion" | jq -r '.Reservations[].Instances[].SubnetId') | jq -r '.Subnets[0].Tags[] | select(.Key=="Name") | .Value'
	2-1-A (예상 출력) **정확히 일치** **순서 중요**	"t3.small" skills-public-a
3-1	3-1-A (명령어 입력)	aws rds describe-db-instances --db-instance-identifier $(aws rds describe-db-clusters --db-cluster-identifier skills-aurora-mysql --query "DBClusters[0].DBClusterMembers[0].DBInstanceIdentifier" --output text) --query "DBInstances[0].{Engine:Engine,DBInstanceClass:DBInstanceClass,DBInstanceStatus:DBInstanceStatus,StorageEncrypted:StorageEncrypted}" ₩ ; aws ec2 describe-subnets --subnet-ids $(aws rds describe-db-subnet-groups --db-subnet-group-name $(aws rds describe-db-clusters --db-cluster-identifier skills-aurora-mysql --query 'DBClusters[*].DBSubnetGroup' --output text) --query 'DBSubnetGroups[*].Subnets[*].SubnetIdentifier' --output text) --query 'Subnets[*].Tags[?Key==`Name`].Value' --output text

	3-1-A
	(예상 출력)
	정확히 일치
	순서 무관
		{
		"Engine": "aurora-mysql",
		"DBInstanceClass": "db.serverless",
		"DBInstanceStatus": "available",
		"StorageEncrypted": true
		}
		skills-data-b

		skills-data-a
4-1	4-1-A (명령어 입력)	aws dynamodb describe-table --table-name order --query 'Table.{Encryption: {Status: SSEDescription.Status, SSEType: SSEDescription.SSEType}, PrimaryKey: KeySchema[?KeyType==`HASH`].AttributeName | [0],CapacityMode: BillingModeSummary.BillingMode}' --output json

	4-1-A
	(예상 출력)
	정확히 일치
	순서 무관
		{
		"Encryption": {
		"Status": "ENABLED",
		"SSEType": "KMS"
		},
		"PrimaryKey": "id",
		"CapacityMode": "PAY_PER_REQUEST"
		}

5-1	5-1-A (명령어 입력)	curl -X POST --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/customer -H "Content-Type: application/json" -d '{"id":"8fa5cde15b9a","name":"James","gender":"male"}' ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/customer?id=8fa5cde15b9a
	5-1-A (예상 출력) **정확히일치** **순서 중요**	{"customer":{"id":"8fa5cde15b9a","name":"James","gender":"male"},"message":"The customer is created."} 201 {"customer":{"id":"8fa5cde15b9a","name":"James","gender":"male"},"message":"The customer is well in database."} 200
	5-1-B (명령어 입력)	curl -X POST --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/product -H "Content-Type: application/json" -d '{"id":"1b67ba873206","name":"sushi","category":"food"}' ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/product?id=1b67ba873206
	5-1-B (예상 출력) **정확히 일치** **순서 중요**	{"product":{"id":"1b67ba873206","name":"sushi","category":"food"},"message":"The product is created."} 201 {"product":{"id":"1b67ba873206","name":"sushi","category":"food"},"message":"The product is well in database."}

		200
5-2	5-2-A (명령어 입력)	curl -X POST --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/order -H "Content-Type: application/json" -d '{"id":"a596f991de36","customerid":"8fa5cde15b9a","productid":"1b67ba873206"}'
	5-2-A (예상 출력) **정확히 일치** **순서 중요**	{"order":{"id":"a596f991de36","customerid":"8fa5cde15b9a","productid":"1b67ba873206"},"message":"The order is created."} 201
	5-2-B (명령어 입력)	curl -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/order?id=a596f991de36
	5-2-B (예상 출력) **정확히 일치** **순서 중요**	{"order":{"id":"a596f991de36","customerid":"8fa5cde15b9a","productid":"1b67ba873206"},"message":"The order is well in database."} 200
5-3	5-3-A (명령어 입력)	kubectl get ExternalSecret application-db-secret -n app | awk 'NR==2 {print $4}' ₩ ; aws secretsmanager describe-secret --secret-id skills-rds-secret --query RotationEnabled
	5-3-A (예상 출력) **정확히 일치**	SecretSynced true

순번	채점 항목	
6-1	6-1-A (명령어 입력)	aws eks describe-nodegroup --cluster-name skills-eks-cluster --nodegroup-name skills-eks-addon-nodegroup --query 'nodegroup.{NodeGroupName:nodegroupName, Status:status, DesiredSize:scalingConfig.desiredSize, InstanceTypes:instanceTypes, LaunchTemplateExists:launchTemplate != null}' --output json ₩ ; aws eks describe-nodegroup --cluster-name skills-eks-cluster --nodegroup-name skills-eks-app-nodegroup --query 'nodegroup.{NodeGroupName:nodegroupName, Status:status, DesiredSize:scalingConfig.desiredSize, InstanceTypes:instanceTypes, LaunchTemplateExists:launchTemplate != null}' --output json

	6-1-A
	(예상 출력)
	정확히 일치
	순서 중요
		{
		"NodeGroupName": "skills-eks-addon-nodegroup",
		"Status": "ACTIVE",
		"DesiredSize": 2,
		"InstanceTypes": [
		"t3.large"
		],
		"LaunchTemplateExists": true
		}
		{
		"NodeGroupName": "skills-eks-app-nodegroup",
		"Status": "ACTIVE",
		"DesiredSize": 2,
		"InstanceTypes": [
		"t3.large"
		],
		"LaunchTemplateExists": true
		}

6-2	6-2-A (명령어 입력)	POD_NODES=$(kubectl get pod -n app --no-headers | awk '{print $1}' | xargs -I {} kubectl describe pod {} -n app | grep 'Node:' | awk '{gsub(/₩/.*/, "", $2); print $2}' \| sort \| uniq) NODEGROUP_NODES=$(kubectl get nodes -l eks.amazonaws.com/nodegroup=skills-eks-app-nodegroup --no-headers | awk '{print $1}' \| sort \| uniq) DIFF=$(diff <(echo "$POD_NODES") <(echo "$NODEGROUP_NODES"))

		if [-z "$DIFF"]; then echo "True"; else echo "False"; fi
	6-2-A (예상 출력) **정확히 일치**	True
6-3	6-3-A (명령어 입력)	kubectl get pods -n kube-system -o wide --selector=eks.amazonaws.com/fargate-profile=coredns-profile --no-headers | awk '{print $3, $7}'
	6-3-A (예상 출력) **정확히 일치** **Node ip 무관** **Fargate 접두사**	Running fargate-ip-10-1-0-18.ap-northeast-2.compute.internal Running fargate-ip-10-1-0-37.ap-northeast-2.compute.internal
7-1	7-1-A (명령어 입력)	kubectl describe deploy -n ingress-nginx ingress-nginx-controller | grep "Image:" | awk '{print $2}'
	7-1-A (예상 출력) **정확히 일치** **밑줄 친 해시값 무관**	registry.k8s.io/ingress-nginx/controller:v1.10.0**@sha256:42b3f0e5d0846876b1791cd3afeb5f1cbbe4259d6f35651dcc1b5c980925379c**
7-2	7-2-A (명령어 입력)	kubectl get ingress ingress-nginx -n app -o yaml | grep "path:" | awk '{print $2}'
	7-2-A (예상 출력) **정확히 일치** **순서 무관**	/v1/customer /v1/product /v1/order

순번	채점 항목	
8-1	8-1-A (명령어 입력)	aws s3 ls | grep skills-static-
	8-1-A (예상 출력) **시간 무관** **밑줄 부분** **달라도 허용** **나머지 일치**	2024-05-26 04:34:36 skills-static-**test**
	8-1-B (명령어 입력)	aws s3api get-bucket-policy --bucket {위에서 본 버킷 이름} --output text | jq '.Statement[].Principal'

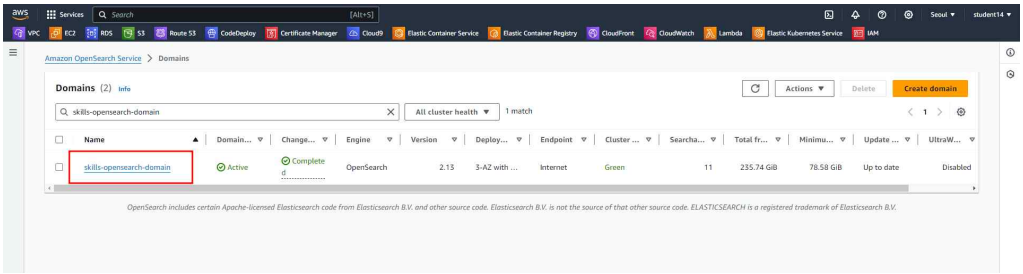
	8-1-B
	(예상 출력)
	정확히 일치
	최소 내용 포함
	순서 무관
		{
		"Service": "cloudfront.amazonaws.com"
		}

8-2	8-2-A (명령어 입력)	SSEAlgorithm=$(aws s3api get-bucket-encryption --bucket {위에서 본 버킷 이름} --query 'ServerSideEncryptionConfiguration.Rules[].ApplyServerSideEncryptionByDefault.SSEAlgorithm' --output text); if [["$SSEAlgorithm" == "aws:kms" || "$SSEAlgorithm" == "aws:kms:dsse"]]; then echo "True"; else echo "False"; fi
	8-2-A (예상 출력) **정확히 일치**	True

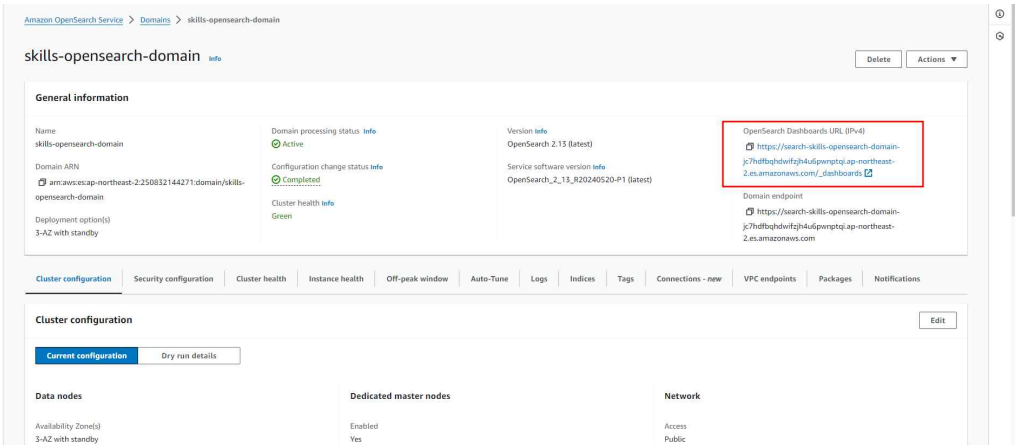
순번	채점 항목	
9-1	9-1-A (명령어 입력)	curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/customer?id=8fa5cde15b9a \| grep -iE "x-cache:\|^200$"
	9-1-A (예상 출력) **정확히 일치** **5개 전부 일치**	x-cache: Miss from cloudfront 200
	9-1-B (명령어 입력)	sleep 30 ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/customer?id=8fa5cde15b9a \| grep -iE "x-cache:\|^200$"
	9-1-B (예상 출력) **정확히 일치** **5개 전부 일치**	x-cache: Miss from cloudfront 200
9-2	9-2-A (명령어 입력)	cat << EOF >> testobject-cdn.txt This is testobject for marking that CDN perform. EOF
	9-2-A (명령어 입력)	aws s3 cp --quiet testobject-cdn.txt s3://${S3_BUCKET}/static/ ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/static/testobject-cdn.txt | grep -iE "x-cache:|^200$"

	9-2-A (예상 출력) **정확히 일치** **5개 전부 일치**	x-cache: Miss from cloudfront 200
	9-2-B (명령어 입력)	sleep 30 ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/static/testobject-cdn.txt \| grep -iE "x-cache:\|^200$"
	9-2-B (예상 출력) **정확히 일치** **5번중 1번이상**	x-cache: Hit from cloudfront 200
9-3	9-3-A (명령어 입력)	cat << EOF >> testobject-cdn2.txt This is testobject for marking that CDN perform. EOF
	9-3-A (명령어 입력)	aws s3 cp --quiet testobject-cdn2.txt s3://${S3_BUCKET}/static/ ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${CF_DOMAIN}/static/testobject-cdn2.txt | grep -iE "x-cache:|location:|^301$"
	9-3-A (예상 출력) **밑줄 부분** **달라도 허용** **정확히 일치**	Location: https://**d742fsr8xncry**.cloudfront.net/static/testobject-cdn2.txt X-Cache: Redirect from cloudfront 301

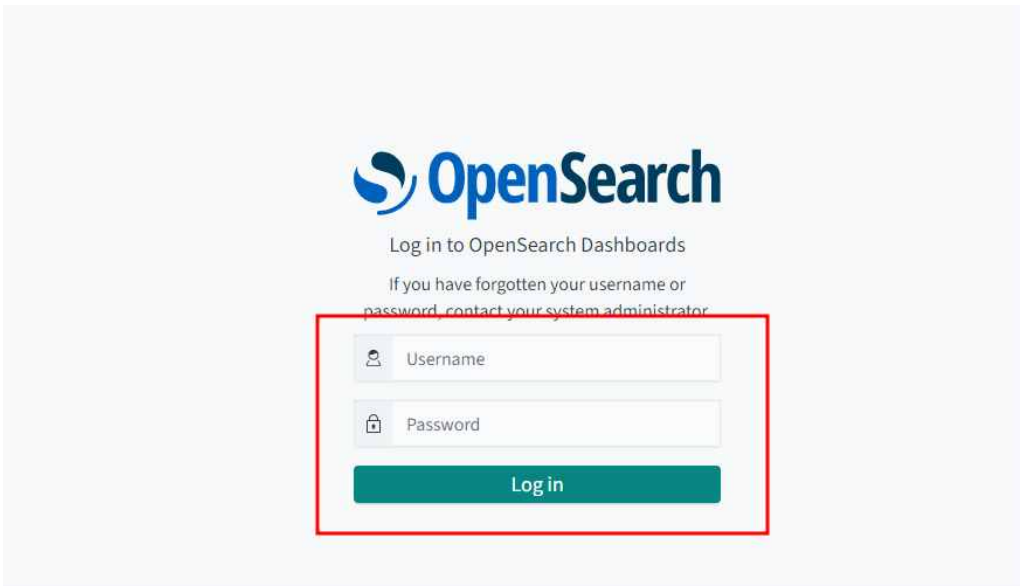
순번	채점 항목	
10-1	10-1-A (명령어 입력)	POD_NODES=$(kubectl get pods -n default | grep fluent-bit | awk '{print $1}' | xargs -I {} kubectl describe pod {} -n default | grep 'Node:' | awk '{gsub(/₩/.*/, "", $2); print $2}' \| sort) NODEGROUP_NODES=$(kubectl get nodes -l eks.amazonaws.com/nodegroup=skills-eks-app-nodegroup --no-headers | awk '{print $1}' \| sort \| uniq) DIFF=$(diff <(echo "$POD_NODES") <(echo "$NODEGROUP_NODES")) if [-z "$DIFF"]; then echo "True"; else echo "False"; fi
	10-1-A (예상 출력) **정확히 일치**	True
	10-1-B (명령어 입력)	aws opensearch describe-domain --domain-name skills-opensearch-domain --query 'DomainStatus.[EngineVersion, ClusterConfig.InstanceType]' --output json
	10-1-B (예상 출력) **정확히 일치**	["OpenSearch_2.13", "t3.medium.search"]
10-2	10-2-A (명령어 입력)	kubectl exec -it -n app deployment.apps/customer -- curl localhost:8080/v1/customer?id=Logtest ₩ ; kubectl exec -it -n app deployment.apps/product -- curl localhost:8080/v1/product?id=Logtest ₩ ; kubectl exec -it -n app deployment.apps/order -- curl localhost:8080/v1/order?id=Logtest
	10-2-A (작업 수행)	OpenSearch Service Console 에서 skills-opensearch-domain 클릭



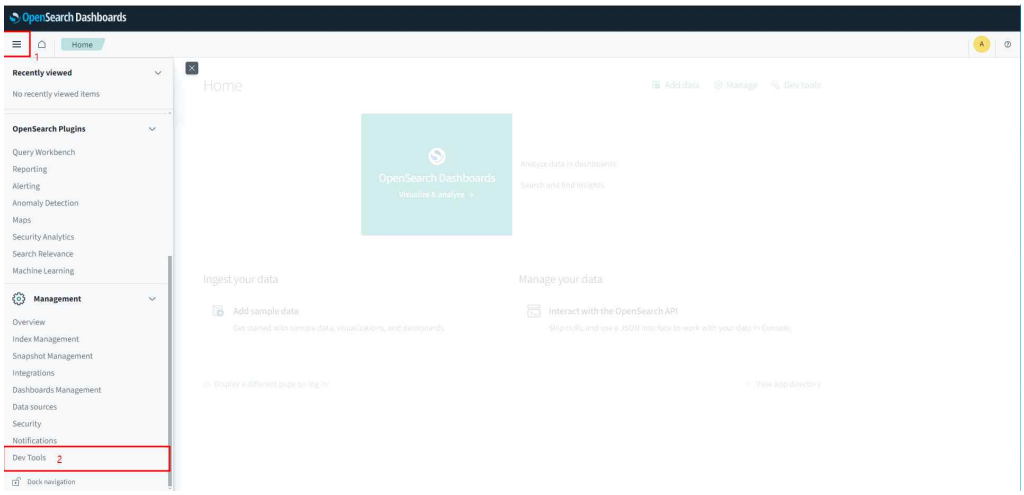
OpenSearch Dashboards URL 접속



선수가 생성한 유저로 로그인



메뉴에서 Management의 Dev Tools 클릭

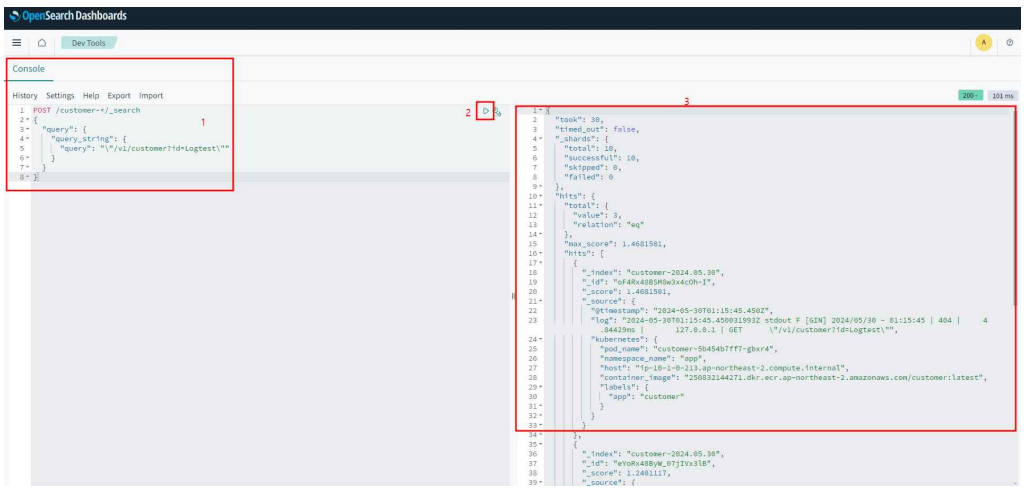


아래 명령어 삽입 및 실행

POST /customer-*/_search

```
{
  "query": {
    "query_string": {
      "query": "₩"/v1/customer?id=Logtest₩""
    }
  }
}
```

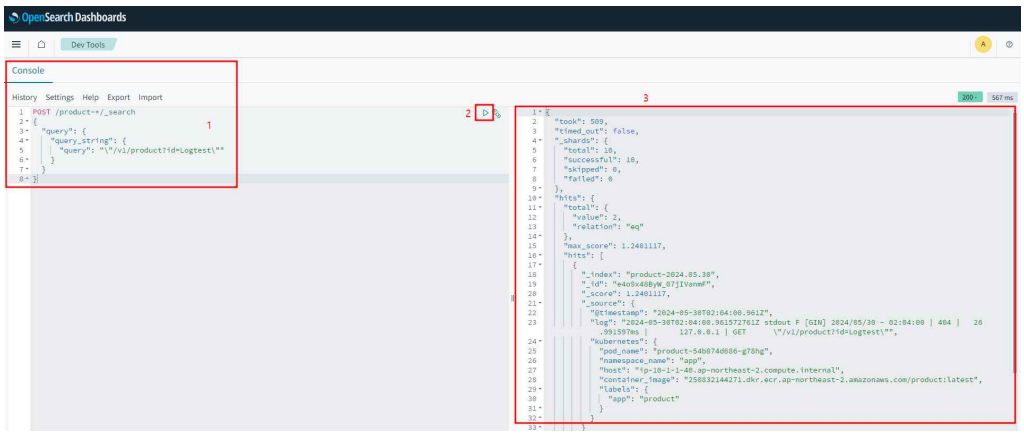
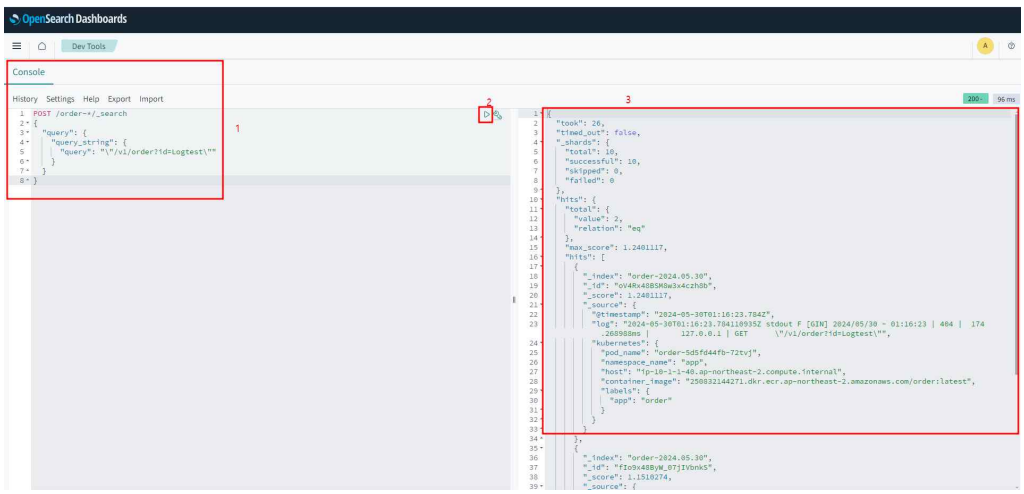
현재시간과 비슷한 시간의 로그가 기록되었는지 확인

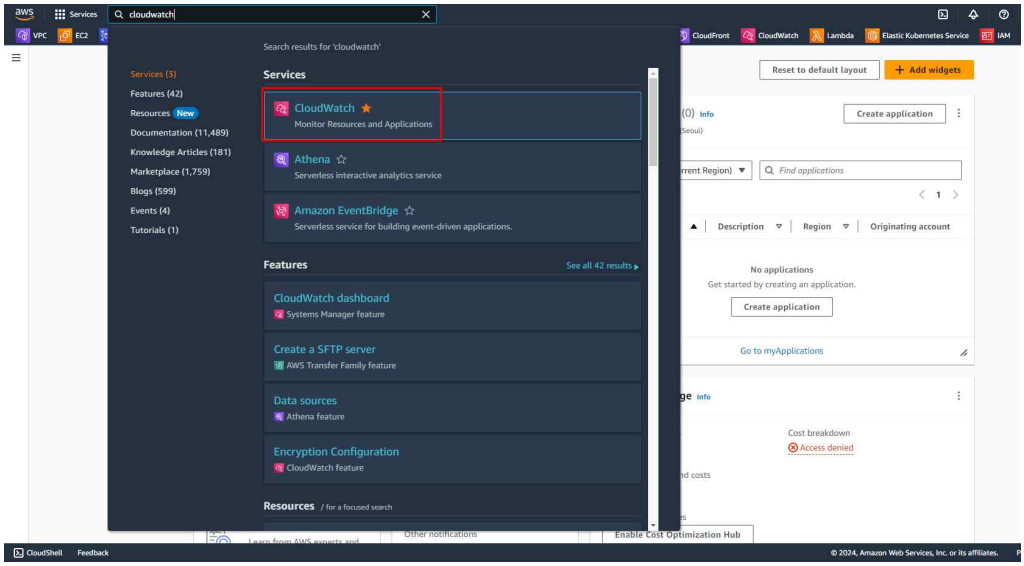
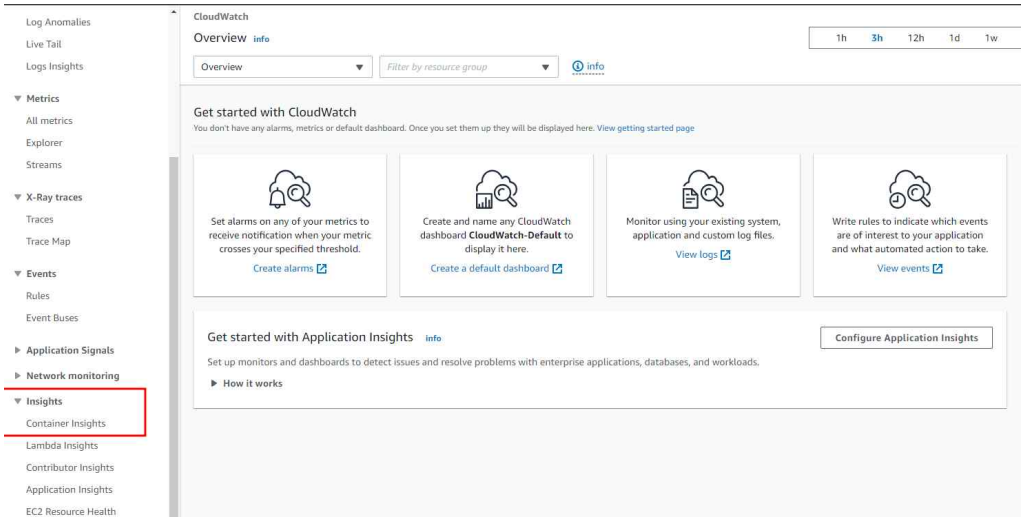
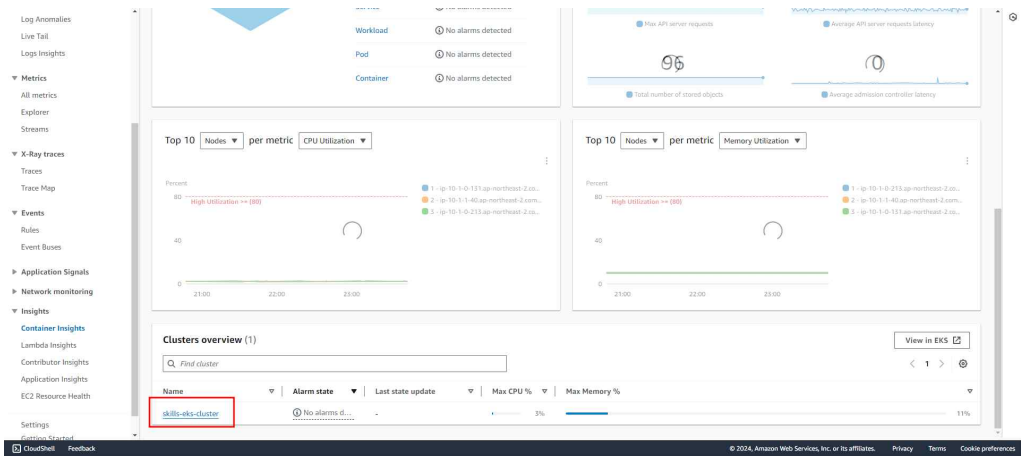


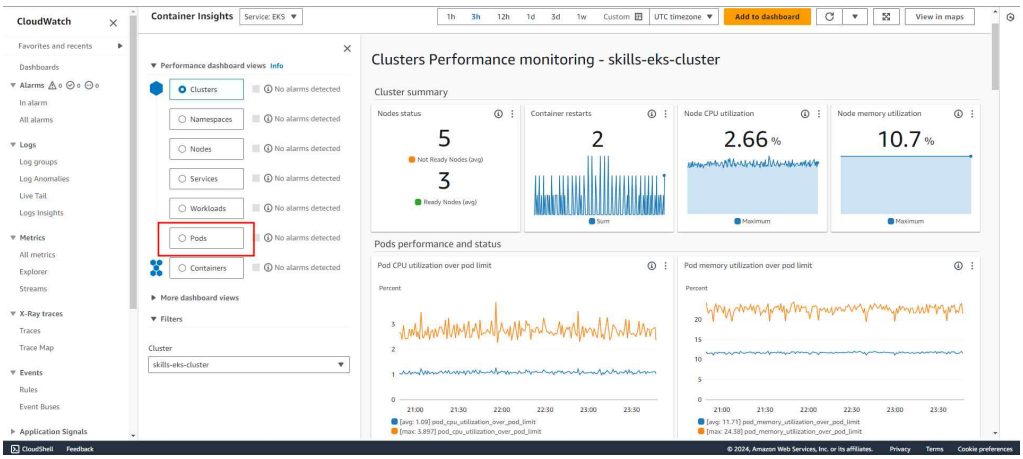
같은 방법으로 아래 명령도 확인

POST /product-*/_search

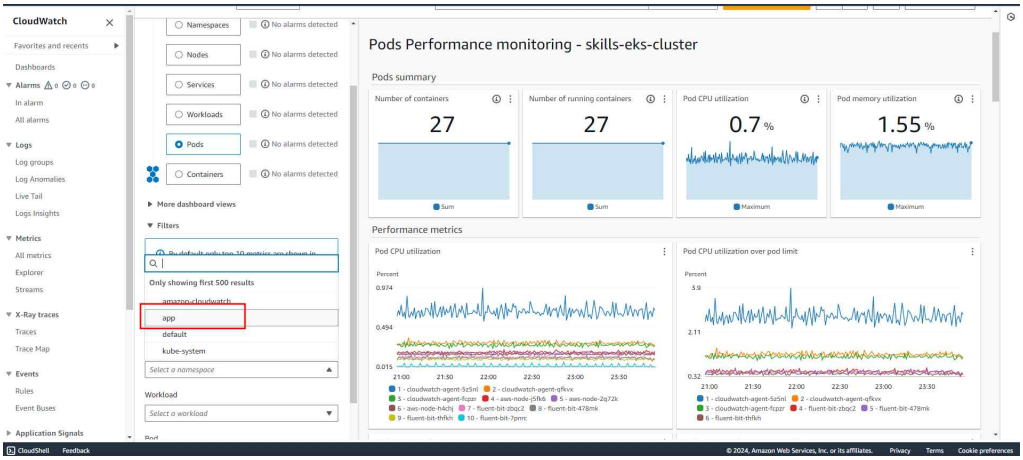
```
{
  "query": {
    "query_string": {
      "query": "₩"/v1/product?id=Logtest₩""
```

		}
		}
		}
		현재시간과 비슷한 시간의 로그가 기록되었는지 확인
		 
		같은 방법으로 아래 명령도 확인
		POST /order-*/_search
		{
		"query": {
		"query_string": {
		"query": "₩"/v1/order?id=Logtest₩""
		}
		}
		}
		현재시간과 비슷한 시간의 로그가 기록되었는지 확인
		* 시간은 UTC Timezone을 고려하여 현재시간 또는 현재시간에서 –9시간, 두 가지 케이스 중에 1가지의 케이스와 비슷한 시간이 나오는지 확인
		 

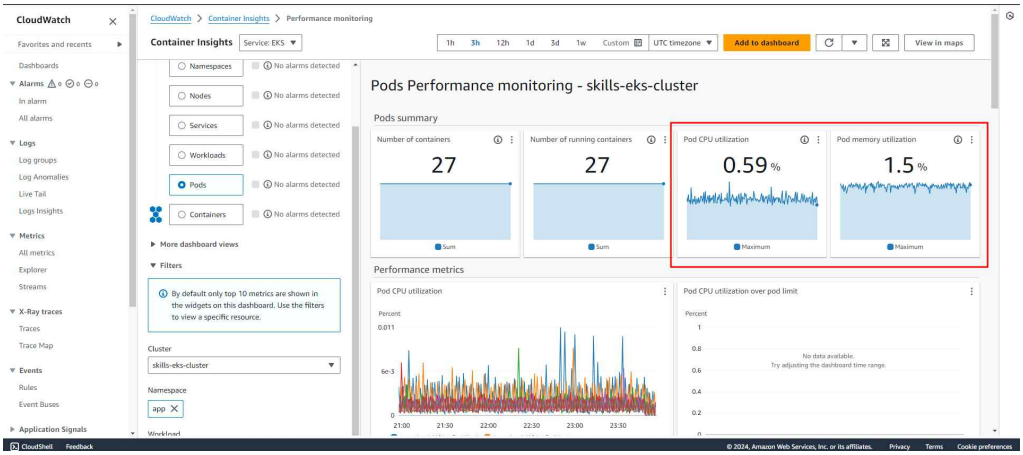
순번	채점 항목	
11-1	11-1-A (작업 수행)	AWS Console에서 Cloudwatch 접속  왼쪽 메뉴에서 Container Insight 클릭  아래로 스크롤 후 skills-eks-cluster 클릭  dashboard view를 pod로 선택



Namespace 드랍다운 메뉴에서 app namespace 선택



Pod CPU utilization이랑 Memory utilization이 정상적으로 수집되는지 확인



* 실제 percentage value는 무관함, 지표가 정상적으로 수집 되는지 확인

순번	채점 항목	
12-1	12-1-A (명령어 입력)	curl -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/customer?id=8fa5cde15b9a
	12-1-A (예상 출력) **정확히 일치**	{"customer":{"id":"8fa5cde15b9a","name":"James","gender":"male"},"message":"The customer is well in database."} 200
	12-1-B (명령어 입력)	curl -X PUT --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/customer?id=8fa5cde15b9a
	12-1-B (예상 출력) **정확히 일치**	405
12-2	12-2-A (명령어 입력)	curl -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/customer?id=skills-baduser-test
	12-2-A (예상 출력) **정확히 일치**	403