

2024년도 전국기능경기대회 채점기준

1. 채점상의 유의사항

직 종 명

클라우드컴퓨팅

※ 다음 사항을 유의하여 채점하십시오.

- 1) AWS 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, 이의 신청까지 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	Networking			○		○	
	2	Bastion			○		○	
	3	NoSQL DataBase			○		○	
	4	Relational DataBase			○		○	
	5	ECR			○		○	
	6	S3			○		○	
	7	EKS			○		○	
	8	Load Balancer			○		○	
	9	CloudFront			○		○	
	10	Application			○		○	
	11	Logging			○		○	
	12	Routing			○		○	
합 계			30					

2) 채점방법 및 기준


과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
제1과제	1	Networking	1	VPC, Subnet 확인	1
			2	Routing Table 확인	1
	2	Bastion	1	Bastion 생성 확인	1
	3	NoSQL DataBase	1	DynamoDB 생성 확인	1
	4	Relational DataBase	1	RDS 생성 확인	1
	5	ECR	1	ECR 생성 확인	1
			2	ECR CVE 확인	1
			3	Multi Region 이미지 확인	1
	6	S3	1	S3 구성 확인	1
	7	EKS	1	EKS 생성 확인	1
			2	EKS Logging 확인	1
			3	EKS Node Group 확인	1
			4	EKS Fargate Profile 확인	1
			5	EKS Pod 구성 확인	1
	8	Load Balancer	1	Load Balancer 생성 확인	1
			2	Load Balancer Listener 확인	1.5
	9	CloudFront	1	CloudFront 생성 확인	1
			2	CloudFront 캐싱 및 구성 확인	1
	10	Application	1	Customer Application 동작 확인	1.5
			2	Product Application 동작 확인	1.5
			3	Order Application 동작 확인	1.5
	11	Logging	1	Worker dashboard	1
			2	match dashboard	1.5

과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
제1과제	12	Routing	1	Customer Routing 확인	1.5
			2	Product Routing 확인	1.5
			3	Order Routing 확인	1.5
	총점				30

3) 채점 내용

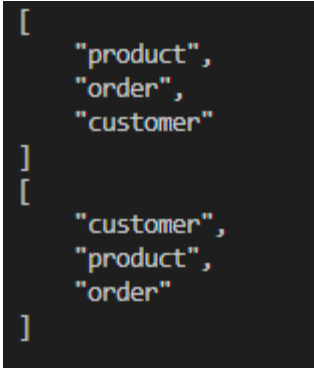
순번	채점 항목
0	<p>1) SSH를 통해 EC2에 접근합니다. (awscli, permission, jq, curl, awscli region)</p> <p>2) 아래 파일들을 EC2의 /root/markings 디렉터리로 복사합니다. - mark.sh</p> <p>3) /root/markings 경로에서 스크립트를 실행합니다. 실행 결과를 기반으로 채점을 진행하되 선수가 이의를 제거할 경우 수동으로 채점을 진행할 수 있도록 합니다.</p> <p>4) 채점을 진행하기 전에 다음 명령어를 수행하여 채점 진행을 위한 사전 작업을 진행합니다. (채점 스크립트로 진행 시 생략)</p> <p>5) 채점은 ap-northeast-2 region에 생성되어 있는 EC2에서 진행합니다.</p>
	<pre>#mark-seoul.sh Set Command # set default region of aws cli aws configure set default.region ap-northeast-2 # set default output of aws cli aws configure set default.output json</pre>
	<pre>#mark-virginia.sh Set Command # set default region of aws cli aws configure set default.region us-east-1 # set default output of aws cli aws configure set default.output json</pre>

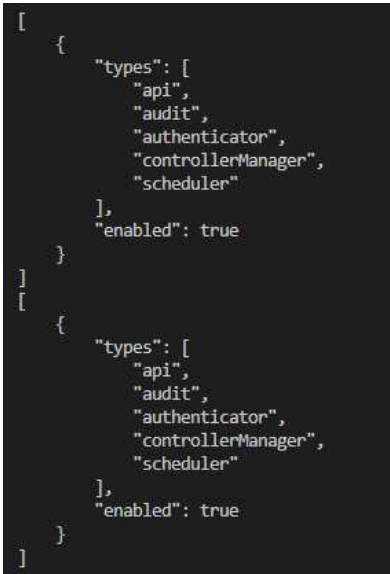
순번	채점 항목
1-1	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어 입력합니다.</p> <pre>aws ec2 describe-vpcs --filter Name=tag:Name,Values=hrdkorea-vpc --query "Vpcs[0].CidrBlock" aws ec2 describe-subnets --filter Name=tag:Name,Values=hrdkorea-public-sn-a --query "Subnets[0].CidrBlock" aws ec2 describe-subnets --filter Name=tag:Name,Values=hrdkorea-public-sn-b --query "Subnets[0].CidrBlock" aws ec2 describe-subnets --filter Name=tag:Name,Values=hrdkorea-private-sn-a --query "Subnets[0].CidrBlock" aws ec2 describe-subnets --filter Name=tag:Name,Values=hrdkorea-private-sn-b --query "Subnets[0].CidrBlock" aws ec2 describe-subnets --filter Name=tag:Name,Values=hrdkorea-protect-sn-a --query "Subnets[0].CidrBlock" aws ec2 describe-subnets --filter Name=tag:Name,Values=hrdkorea-protect-sn-b --query "Subnets[0].CidrBlock"</pre> <pre>aws ec2 describe-vpcs --filter Name=tag:Name,Values=hrdkorea-vpc --query "Vpcs[0].CidrBlock" --region us-east-1 aws ec2 describe-subnets --filter Name=tag:Name,Values=hrdkorea-public-sn-a --query "Subnets[0].CidrBlock" --region us-east-1 aws ec2 describe-subnets --filter Name=tag:Name,Values=hrdkorea-public-sn-b --query "Subnets[0].CidrBlock" --region us-east-1 aws ec2 describe-subnets --filter Name=tag:Name,Values=hrdkorea-private-sn-a --query "Subnets[0].CidrBlock" --region us-east-1 aws ec2 describe-subnets --filter Name=tag:Name,Values=hrdkorea-private-sn-b --query "Subnets[0].CidrBlock" --region us-east-1 aws ec2 describe-subnets --filter Name=tag:Name,Values=hrdkorea-protect-sn-a --query "Subnets[0].CidrBlock" --region us-east-1 aws ec2 describe-subnets --filter Name=tag:Name,Values=hrdkorea-protect-sn-b --query "Subnets[0].CidrBlock" --region us-east-1</pre> <p>3) 출력된 값이 아래 사진과 동일한지 확인합니다.</p>  <pre>"10.129.0.0/16" "10.129.0.0/24" "10.129.1.0/24" "10.129.11.0/24" "10.129.12.0/24" "10.129.21.0/24" "10.129.22.0/24" "10.129.0.0/16" "10.129.0.0/24" "10.129.1.0/24" "10.129.11.0/24" "10.129.12.0/24" "10.129.21.0/24" "10.129.22.0/24"</pre>

순번	채점 항목
1-2	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=hrdkorea-private-a-rt --query "RouteTables[].Routes[].NatGatewayId" grep "nat-" wc -l</pre> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=hrdkorea-private-b-rt --query "RouteTables[].Routes[].NatGatewayId" grep "nat-" wc -l</pre> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=hrdkorea-public-rt --query "RouteTables[].Routes[]" grep "igw-" wc -l</pre> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=hrdkorea-protect-b-rt --query "RouteTables[].Routes[]" grep -E "igw- nat-" wc -l</pre> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=hrdkorea-private-a-rt --query "RouteTables[].Routes[].NatGatewayId" --region us-east-1 grep "nat-" wc -l</pre> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=hrdkorea-private-b-rt --query "RouteTables[].Routes[].NatGatewayId" --region us-east-1 grep "nat-" wc -l</pre> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=hrdkorea-public-rt --query "RouteTables[].Routes[]" --region us-east-1 grep "igw-" wc -l</pre> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=hrdkorea-protect-b-rt --query "RouteTables[].Routes[]" --region us-east-1 grep -E "igw- nat-" wc -l</pre> <p>3) 출력된 값이 아래 사진과 동일한지 확인합니다.</p> 

순번	채점 항목
2-1	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어 후 “i-”가 출력되는지 확인합니다. (배점 0.3)</p> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=hrdkorea-bastion --query "Reservations[].Instances[].InstanceId"</pre> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=hrdkorea-bastion --query "Reservations[].Instances[].InstanceId" --region us-east-1</pre> <p>3) 아래 명령어 입력 후 “t3.small”이 출력되는지 확인합니다. (배점 0.3)</p> <p>(만약 2개 중 1개라도 일치하지 않으면 틀린 것으로 처리합니다.)</p> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=hrdkorea-bastion --query "Reservations[].Instances[].InstanceType"</pre> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=hrdkorea-bastion --region us-east-1 --query "Reservations[].Instances[].InstanceType"</pre> <p>4) 아래 명령어 입력 후 "Amazon Linux 2023"으로 시작되는 텍스트가 출력되는지 확인합니다. (배점 0.4)</p> <p>(만약 2개 중 1개라도 일치하지 않으면 틀린 것으로 처리합니다.)</p> <pre>Seoul_IMAGE_ID=\$(aws ec2 describe-instances --filter Name=tag:Name,Values=hrdkorea-bastion --query "Reservations[].Instances[].ImageId" --output text)</pre> <pre>US_IMAGE_ID=\$(aws ec2 describe-instances --filter Name=tag:Name,Values=hrdkorea-bastion --query "Reservations[].Instances[].ImageId" --output text --region us-east-1)</pre> <pre>aws ec2 describe-images --image-ids \$Seoul_IMAGE_ID --query "Images[].Description"</pre> <pre>aws ec2 describe-images --image-ids \$US_IMAGE_ID --query "Images[].Description" --region us-east-1</pre>
3-1	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 “order”가 출력되는지 확인합니다. (배점 0.5)</p> <p>(만약 2개중 1개라도 일치하지 않으면 틀린 것으로 처리합니다.)</p> <pre>aws dynamodb list-tables --query "TableNames[?@ == 'order']"</pre> <pre>aws dynamodb list-tables --query "TableNames[?@ == 'order']" --region us-east-1</pre> <p>3) 아래 명령어 입력 후 “PAY_PRE_REQUEST”가 출력되는지 확인합니다. (배점 0.5)</p> <p>(만약 2개중 1개라도 일치하지 않으면 틀린 것으로 처리합니다.)</p> <pre>aws dynamodb describe-table --table-name order --query "Table.BillingModeSummary.BillingMode"</pre> <pre>aws dynamodb describe-table --table-name order --query "Table.BillingModeSummary.BillingMode" --region us-east-1</pre>

순번	채점 항목
4-1	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 “aurora-mysql, 3409, hrdkorea-user”가 출력되는지 합니다. (만약 2개 중 1개라도 일치하지 않으면 틀린 것으로 처리합니다.) (배점 0.7)</p> <pre>aws rds describe-db-clusters --query "DBClusters[0].{Engine:Engine,Port:Port,MasterUsername:MasterUsername}" aws rds describe-db-clusters --query "DBClusters[0].{Engine:Engine,Port:Port,MasterUsername:MasterUsername}" --region us-east-1</pre> <pre>{ "Engine": "aurora-mysql", "Port": 3409, "MasterUsername": "hrdkorea_user" }</pre> <p>3) 아래 명령어 입력 후 “aurora-mysql, available, db.r5.large, aurora”가 출력되는지 확인합니다. (배점 0.7) (만약 2개 중 1개라도 일치하지 않으면 틀린 것으로 처리합니다.)</p> <pre>aws rds describe-db-instances --db-instance-identifier hrdkorea-rds-instance --query "DBInstances[0].{Engine:Engine,DBInstanceStatus:DBInstanceStatus,DBInstanceClass:DBInstanceClass,StorageType:StorageType}" aws rds describe-db-instances --db-instance-identifier hrdkorea-rds-instance-us --query "DBInstances[0].{Engine:Engine,DBInstanceStatus:DBInstanceStatus,DBInstanceClass:DBInstanceClass,StorageType:StorageType}" --region us-east-1</pre> <pre>{ "Engine": "aurora-mysql", "DBInstanceStatus": "available", "DBInstanceClass": "db.r5.large", "StorageType": "aurora" }</pre> <pre>{ "Engine": "aurora-mysql", "DBInstanceStatus": "available", "DBInstanceClass": "db.r5.large", "StorageType": "aurora" }</pre>
5-1	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 “hrdkorea-ecr-repo”가 출력되는지 확인합니다.</p> <pre>aws ecr describe-repositories --repository-name hrdkorea-ecr-repo -query "repositories[].repositoryName "</pre>

순번	채점 항목
5-2	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 “COMPLETE”가 출력되는지 확인합니다. (만약 3개 중 1개라도 일치하지 않으면 틀린 것으로 처리합니다.)</p> <pre>aws ecr describe-image-scan-findings --repository-name hrdkorea-ecr-repo --image-id imageTag=customer --query "imageScanStatus.status"</pre> <pre>aws ecr describe-image-scan-findings --repository-name hrdkorea-ecr-repo --image-id imageTag=product --query "imageScanStatus.status"</pre> <pre>aws ecr describe-image-scan-findings --repository-name hrdkorea-ecr-repo --image-id imageTag=order --query "imageScanStatus.status"</pre>
5-3	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 순서에 상관없이 “product,order,customer”가 출력되는지 확인합니다. (배점 0.5)</p> <pre>aws ecr list-images --repository-name hrdkorea-ecr-repo --query "imageIds[].imageTag"</pre> <pre>aws ecr list-images --repository-name hrdkorea-ecr-repo --query "imageIds[].imageTag"</pre> <pre>--region us-east-1</pre>  <pre>["product", "order", "customer"] ["customer", "product", "order"]</pre> <p>3) 아래 명령어를 입력 후 “us-east-1”가 출력되는지 확인합니다. (배점 0.5)</p> <pre>aws ecr describe-registry --query "replicationConfiguration.rules[0].destinations[0].region"</pre>
6-1	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 “hrdkorea-static<임의의 4자리 숫자>”가 출력되는지 확인합니다 (배점 0.5)</p> <pre>aws s3api list-buckets --query "Buckets[].Name"</pre> <p>3) 아래 명령어를 입력 후 “index.html”출력되는지 확인합니다. (배점 0.5)</p> <pre>BUCKET_NAME=\$(aws s3api list-buckets --query "Buckets[].Name" --output text)</pre> <pre>aws s3 ls s3://\$BUCKET_NAME/static/</pre>

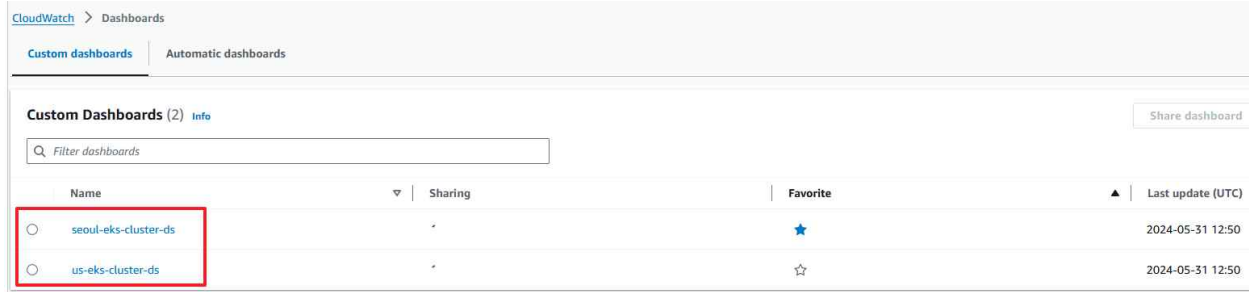
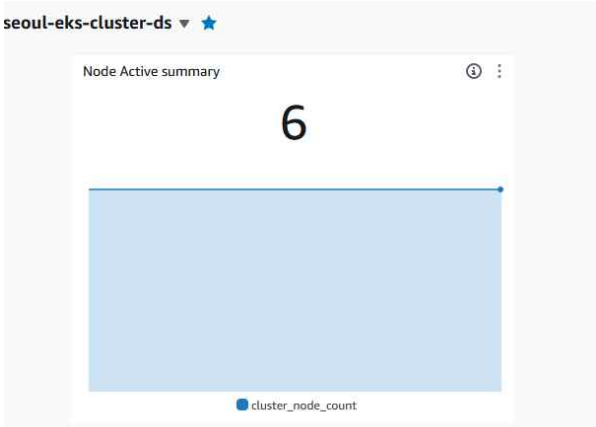
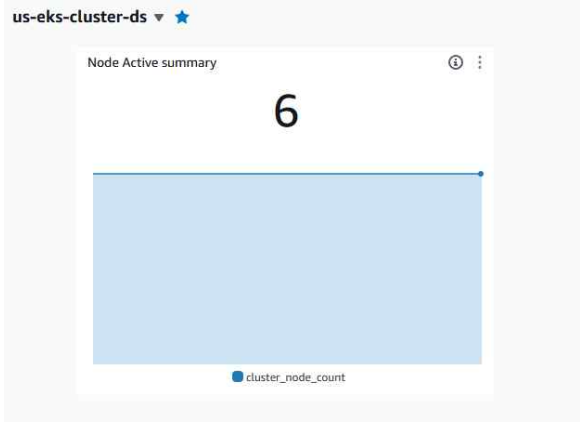
순번	채점 항목
7-1	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 “1.29”가 출력되는 확인합니다. (만약 2개중 1개라도 일치하지 않으면 틀린 것으로 처리합니다.)</p> <pre>aws eks describe-cluster --name hrdkorea-cluster --query "cluster.version" aws eks describe-cluster --name hrdkorea-cluster --query "cluster.version" --region us-east-1</pre>
7-2	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다</p> <pre>aws eks describe-cluster --name hrdkorea-cluster --query "cluster.logging.clusterLogging" aws eks describe-cluster --name hrdkorea-cluster --query "cluster.logging.clusterLogging" --region us-east-1</pre> <p>3) 순서에 상관없이 아래 사진과 같이 “api,audit,authenticator,controllerManager,scheduler”값이 출력되는 확인합니다. (만약 2개중 1개라도 일치하지 않으면 틀린 것으로 처리합니다.)</p>  <pre>[{ "types": ["api", "audit", "authenticator", "controllerManager", "scheduler"], "enabled": true }, { "types": ["api", "audit", "authenticator", "controllerManager", "scheduler"], "enabled": true }]</pre>
7-3	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 “t3.large”가 출력되는지 확인합니다. (만약 2개중 1개라도 일치하지 않으면 틀린 것으로 처리합니다.)</p> <pre>aws eks describe-nodegroup --cluster-name hrdkorea-cluster --nodegroup-name hrdkorea-customer-ng --query "nodegroup.instanceTypes" aws eks describe-nodegroup --cluster-name hrdkorea-cluster --nodegroup-name hrdkorea-customer-ng --query "nodegroup.instanceTypes" --region us-east-1</pre>

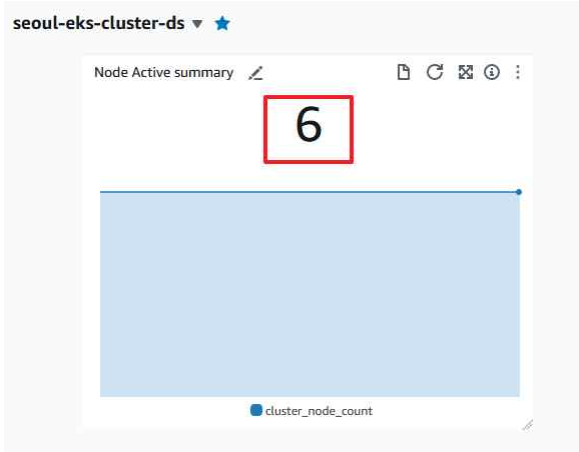
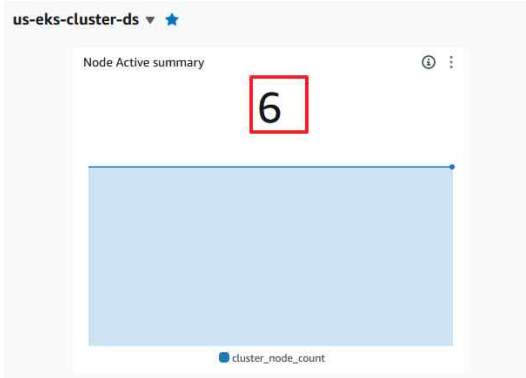
순번	채점 항목
7-4	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 “hrdkorea-addon-profile”가 출력되는지 확인합니다. (만약 2개중 1개라도 일치하지 않으면 틀린 것으로 처리합니다.)</p> <pre>aws eks describe-fargate-profile --cluster-name hrdkorea-cluster --fargate-profile-name hrdkorea-addon-profile --query "fargateProfile.fargateProfileName" aws eks describe-fargate-profile --cluster-name hrdkorea-cluster --fargate-profile-name hrdkorea-addon-profile --query "fargateProfile.fargateProfileName" --region us-east-1</pre>
7-5	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 “hrdkorea-customer-ng,hrdkorea-product-ng,hrdkorea-order-ng”가 순서대로 출력되는지 확인합니다.</p> <pre>POD_NAME=\$(kubectl get pods -n hrdkorea grep customer awk '{print \$1}' head -n 1) NODE_NAME=\$(kubectl get po \$POD_NAME -n hrdkorea -o jsonpath='{.spec.nodeName}') NODE_GROUP=\$(kubectl get node \$NODE_NAME --show-labels grep -o 'hrdkorea-customer-ng') echo "\$NODE_GROUP" POD_NAME=\$(kubectl get pods -n hrdkorea grep product awk '{print \$1}' head -n 1) NODE_NAME=\$(kubectl get po \$POD_NAME -n hrdkorea -o jsonpath='{.spec.nodeName}') NODE_GROUP=\$(kubectl get node \$NODE_NAME --show-labels grep -o 'hrdkorea-product-ng') echo "\$NODE_GROUP" POD_NAME=\$(kubectl get pods -n hrdkorea grep order awk '{print \$1}' head -n 1) NODE_NAME=\$(kubectl get po \$POD_NAME -n hrdkorea -o jsonpath='{.spec.nodeName}') NODE_GROUP=\$(kubectl get node \$NODE_NAME --show-labels grep -o 'hrdkorea-order-ng') echo "\$NODE_GROUP"</pre>
8-1	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 “hrdkorea-app-alb,internet-facing” 각각 2개씩 출력되는지 확인합니다.</p> <pre>aws elbv2 describe-load-balancers --names hrdkorea-app-alb --query "LoadBalancers[].LoadBalancerName" aws elbv2 describe-load-balancers --names hrdkorea-app-alb --query "LoadBalancers[].LoadBalancerName" --region us-east-1 aws elbv2 describe-load-balancers --names hrdkorea-app-alb --query "LoadBalancers[].Scheme" aws elbv2 describe-load-balancers --names hrdkorea-app-alb --query "LoadBalancers[].Scheme" --region us-east-1</pre>

순번	채점 항목
8-2	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 출력이 아래 사진과 같은 지 확인합니다.</p> <pre> SEOUL_ALB_ADDRESS=\$(aws elbv2 describe-load-balancers --query "LoadBalancers[?LoadBalancerName=='hrdkorea-app-alb'].DNSName" --output text) US_ALB_ADDRESS=\$(aws elbv2 describe-load-balancers --query "LoadBalancers[?LoadBalancerName=='hrdkorea-app-alb'].DNSName" --output text --region us-east-1) curl -o /dev/null -s -w "%{http_code}\n" http://\$SEOUL_ALB_ADDRESS/healthcheck?path=skills echo curl http://\$SEOUL_ALB_ADDRESS/healthcheck?path=customer echo curl http://\$SEOUL_ALB_ADDRESS/healthcheck?path=order echo curl http://\$SEOUL_ALB_ADDRESS/healthcheck?path=product echo curl -o /dev/null -s -w "%{http_code}\n" http://\$US_ALB_ADDRESS/healthcheck?path=skills echo curl http://\$US_ALB_ADDRESS/healthcheck?path=customer echo curl http://\$US_ALB_ADDRESS/healthcheck?path=order echo curl http://\$US_ALB_ADDRESS/healthcheck?path=product echo </pre>  <pre> 404 {"status":"ok."} {"status":"ok."} {"status":"ok."} 404 {"status":"ok."} {"status":"ok."} {"status":"ok."} </pre>

순번	채점 항목
9-1	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 “XXXXXXXXXXXX.cloudfront.net”이 출력되는지 확인합니다 (배점 0.5)</p> <pre>aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=hrdkorea-cdn --resource-type-filters 'cloudfront' --region us-east-1 --query "ResourceTagMappingList[0].ResourceARN" --output text sed 's:./::' xargs -I {} aws cloudfront get-distribution --id {} --query "Distribution.DomainName" --output text</pre> <p>3) 아래 명령어를 입력 후 “PriceClass_All”이 출력되는지 확인합니다 (배점 0.5)</p> <pre>aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=hrdkorea-cdn --resource-type-filters 'cloudfront' --region us-east-1 --query "ResourceTagMappingList[0].ResourceARN" --output text sed 's:./::' xargs -I {} aws cloudfront get-distribution --id {} --query "Distribution.DistributionConfig.PriceClass"</pre>
9-2	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>CLOUDFRONT_DNS=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=hrdkorea-cdn --resource-type-filters 'cloudfront' --region us-east-1 --query "ResourceTagMappingList[0].ResourceARN" --output text sed 's:./::' xargs -I {} aws cloudfront get-distribution --id {} --query "Distribution.DomainName" --output text) echo \$CLOUDFRONT_DNS</pre> <p>3) 아래 명령어를 입력합니다.</p> <pre>curl --silent --head https://\$CLOUDFRONT_DNS/static/index.html grep "x-cache"</pre> <p>4) “x-cache: Hit from cloudfront”이라는 문구가 출력되는지 확인합니다. 최대 5가지 실행해볼 수 있습니다</p>
10-1	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>CLOUDFRONT_DNS=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=hrdkorea-cdn --resource-type-filters 'cloudfront' --region us-east-1 --query "ResourceTagMappingList[0].ResourceARN" --output text sed 's:./::' xargs -I {} aws cloudfront get-distribution --id {} --query "Distribution.DomainName" --output text) curl -X POST -H "Content-type: application/json" -d '{"id":"cloud","name":"user","gender":"man"}' https://\$CLOUDFRONT_DNS/v1/customer</pre> <p>3) “{"customer":{"id":"cloud","name":"user","gender":"man"},"message":"The customer is created."}”이 출력되는지 확인합니다. (배점 0.5)</p> <p>4) 아래 명령어를 입력합니다</p> <pre>curl -X GET https://\$CLOUDFRONT_DNS/v1/customer?id=cloud</pre> <p>5) “{"customer":{"id":"cloud","name":"user","gender":"man"},"message":"The customer is well in database."}”이 출력되는지 확인합니다 (배점 1.0)</p>

순번	채점 항목
10-2	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre> CLOUDFRONT_DNS=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=hrdkorea-cdn --resource-type-filters 'cloudfront' --region us-east-1 --query "ResourceTagMappingList[0].ResourceARN" --output text sed 's:.*/::' xargs -I {} aws cloudfront get-distribution --id {} --query "Distribution.DomainName" --output text) curl -X POST -H "Content-type: application/json" -d '{"id":"cloud","name":"user","category":"stduent"}' https://\$CLOUDFRONT_DNS/v1/product </pre> <p>3) “{"product":{"id":"cloud","name":"user","category":"stduent"},"message":"The product is created."}”이 출력되는지 확인합니다. (배점 0.5)</p> <p>4) 아래 명령어를 입력합니다.</p> <pre> curl -X GET https://\$CLOUDFRONT_DNS/v1/product?id=cloud </pre> <p>5) “{"product":{"id":"cloud","name":"user","category":"stduent"},"message":"The product is well in database."}”이 출력되는지 확인합니다. (배점 1.0)</p>
10-3	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre> CLOUDFRONT_DNS=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=hrdkorea-cdn --resource-type-filters 'cloudfront' --region us-east-1 --query "ResourceTagMappingList[0].ResourceARN" --output text sed 's:.*/::' xargs -I {} aws cloudfront get-distribution --id {} --query "Distribution.DomainName" --output text) curl -X POST -H "Content-type: application/json" -d '{"id":"cloud","customerid":"user","productid":"skills"}' https://\$CLOUDFRONT_DNS/v1/order </pre> <p>3) “{"order":{"id":"cloud","customerid":"user","productid":"skills"},"message":"The order is created."}”이 출력되는지 확인합니다. (배점 0.5)</p> <p>4) 아래 명령어를 입력합니다.</p> <pre> curl -X GET https://\$CLOUDFRONT_DNS/v1/order?id=cloud </pre> <p>5) “{"order":{"id":"cloud","customerid":"user","productid":"skills"},"message":"The order is well in database."}”이 출력되는지 확인합니다. (배점 1.0)</p>

순번	채점 항목
11-1	<div data-bbox="212 320 829 488"> <p>1) AWS Web console에 로그인 합니다.</p> <p>2) CloudWatch page로 이동합니다.</p> <p>3) 왼쪽 패널에 Dashboards를 클릭해 이동합니다.</p> <p>4) worker dashboard를 클릭합니다.</p> </div> <div data-bbox="212 499 1468 790">  </div> <div data-bbox="212 790 1299 869"> <p>5) seoul-eks-cluster-ds,us-eks-cluster-ds가 생성되어 있는지 확인합니다 (배점 0.2)</p> <p>6) seoul-eks-cluster-ds를 클릭합니다</p> </div> <div data-bbox="212 880 810 1305">  </div> <div data-bbox="212 1317 1179 1440"> <p>7) 그림처럼 Node Active summary가 생성되어 있는지 확인합니다. (배점 0.4)</p> <p>8) 왼쪽 패널에 Dashboards를 클릭해 이동합니다.</p> <p>9) us-eks-cluster-ds를 클릭합니다.</p> </div> <div data-bbox="212 1451 794 1877">  </div> <div data-bbox="212 1888 1193 1921"> <p>10) 그림처럼 Node Active summary가 생성되어 있는지 확인합니다. (배점 0.4)</p> </div>

순번	채점 항목
11-2	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>kubectl get node grep -v NAME wc -l</pre> <p>3) AWS Web console에 로그인 합니다.</p> <p>4) CloudWatch page로 이동합니다.</p> <p>5) 왼쪽 패널에 Dashboards를 클릭해 이동합니다.</p> <p>6) worker dashboard를 클릭합니다.</p> <p>7) seoul-eks-cluster-ds를 클릭합니다</p> <p>8) 2)번에서 출력된 개수와 같은지 확인합니다 (배점 1.0)</p> <div data-bbox="212 712 793 1162">  </div> <p>9) SSH를 통해 virginia EC2에 접근합니다.</p> <p>10) 아래 명령어를 입력합니다.</p> <pre>kubectl get node grep -v NAME wc -l</pre> <p>11) 6)처럼 대시보드에 접근합니다</p> <p>12) us-eks-cluster-ds를 클릭합니다.</p> <div data-bbox="212 1391 740 1767">  </div> <p>13) 10)번에서 출력된 개수와 같은지 확인합니다 (배점 0.5)</p>

순번	채점 항목
12-1	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>kubectl delete deployment customer-deployment -n hrdkorea CLOUDFRONT_DNS=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=hrdkorea-cdn --resource-type-filters 'cloudfront' --region us-east-1 --query "ResourceTagMappingList[0].ResourceARN" --output text sed 's:.*/::' xargs -I {} aws cloudfront get-distribution --id {} --query "Distribution.DomainName" --output text) sleep 200</pre> <p>3) 아래 명령어를 입력합니다.</p> <pre>curl -X POST -H "Content-type: application/json" -d '{"id":"cloud1","name":"user1","gender":"man1"}' https://\$CLOUDFRONT_DNS/v1/customer</pre> <p>4) '{"customer":{"id":"cloud1","name":"user1","gender":"man1"},"message":"The customer is created."}' "이 출력되는지 확인합니다. (배점 0.5)</p> <p>5) 아래 명령어를 입력합니다.</p> <pre>curl -X GET https://\$CLOUDFRONT_DNS/v1/customer?id=cloud1</pre> <p>6) '{"customer":{"id":"cloud1","name":"user1","gender":"man1"},"message":"The customer is well in database."}' "이 출력되는지 확인합니다. (배점 1.0)</p>
12-2	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>kubectl delete deployment product-deployment -n hrdkorea CLOUDFRONT_DNS=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=hrdkorea-cdn --resource-type-filters 'cloudfront' --region us-east-1 --query "ResourceTagMappingList[0].ResourceARN" --output text sed 's:.*/::' xargs -I {} aws cloudfront get-distribution --id {} --query "Distribution.DomainName" --output text) echo \$CLOUDFRONT_DNS sleep 200</pre> <p>3) 아래 명령어를 입력합니다.</p> <pre>curl -X POST -H "Content-type: application/json" -d '{"id":"cloud2","name":"user2","category":"stduent2"}' https://\$CLOUDFRONT_DNS/v1/product</pre> <p>4) '{"product":{"id":"cloud2","name":"user2","category":"stduent2"},"message":"The product is created."}' "이 출력되는지 확인합니다. (배점 0.5)</p> <p>5) 아래 명령어를 입력합니다.</p> <pre>curl -X GET https://\$CLOUDFRONT_DNS/v1/product?id=cloud2</pre> <p>6) '{"product":{"id":"cloud2","name":"user2","category":"stduent2"},"message":"The product is well in database."}' "이 출력되는지 확인합니다 (배점 1.0)</p>

순번	채점 항목
12-3	<p>1) SSH를 통해 EC2에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>kubectl delete deployment order-deployment -n hrdkorea</pre> <pre>CLOUDFRONT_DNS=\$(aws resourcegroupstaggingapi get-resources --tag-filters</pre> <pre>Key=Name,Values=hrdkorea-cdn --resource-type-filters 'cloudfront' --region us-east-1 --query</pre> <pre>"ResourceTagMappingList[0].ResourceARN" --output text sed 's:.*/::' xargs -I {} aws</pre> <pre>cloudfront get-distribution --id {} --query "Distribution.DomainName" --output text)</pre> <pre>sleep 200</pre> <p>3) 아래 명령어를 입력합니다.</p> <pre>curl -X POST -H "Content-type: application/json" -d</pre> <pre>'{"id":"cloud3","customerid":"user3","productid":"skills3"}' https://\$CLOUDFRONT_DNS/v1/order</pre> <p>4) “{"order":{"id":"cloud3","customerid":"user3","productid":"skills3"},"message":"The order is created."}”이 출력되는지 확인합니다. (배점 0.5)</p> <p>5) 아래 명령어를 입력합니다.</p> <pre>curl -X GET https://\$CLOUDFRONT_DNS/v1/order?id=cloud3</pre> <p>6) “{"order":{"id":"cloud3","customerid":"user3","productid":"skills3"},"message":"The order is well in database."}”이 출력되는지 확인합니다. (배점 1.0)</p>