

2024년도 지방기능경기대회 채점기준

1. 채점 시 유의사항

직 종 명

클라우드컴퓨팅

※ 다음 사항을 유의하여 채점하시오.

- 1) AWS의 리전은 ap-northeast-2를 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) 삭제 채점은 되돌릴 수 없으므로 유의하여 진행합니다.
- 8) 이의신청까지 완료 이후 선수가 생성한 클라우드 리소스를 삭제합니다.
- 9) 부분 점수가 있는 문항은 채점 항목에 부분 점수가 적혀져 있습니다.
- 10) 부분 점수가 따로 없는 문항은 전체 다 맞아야 점수로 인정됩니다.
- 11) 채점 진행 전 환경 셋업을 위해 다음 사항을 확인해야 합니다.
 - Bastion에 SSH로 접근할 수 있는지 확인합니다.
 - Bastion에서 AWS CLI v2, curl, jq가 설치되어 있는지 확인합니다.
 - Bastion에서 IAM Role이 매핑되어 AWS CLI로 AWS의 모든 리소스에 접근 가능한지 확인합니다.
 - `aws sts get-caller-identity` 명령을 통해 선수의 계정이 아닌 다른 계정에 접근하고 있는지 확인합니다. 만약, 다른 계정이라면 부정행위를 의심할 수 있습니다.
- 12) 채점 전 채점환경 구성을 위해 ~/.aws/config 에 아래 내용이 추가되도록 합니다.

```
[default]
region = ap-northeast-2
output = json
```
- 13) 채점 시에는 별도로 제공한 채점 스크립트(mark.sh)를 실행하여 채점할 수 있습니다. 다만, 선수가 직접 입력을 원할 경우 채점기준표에 명시된 명령어 그대로 입력하여 채점할 수 있습니다.

2. 채점기준표

1) 주요항목별 배점			직 종 명		클라우드컴퓨팅			
과제 번호	일련 번호	주요항목	배점	채점방법		채점시기		비고
				독립	합의	경기 진행중	경기 종료후	
제2과제	1	Networking	4.0		○		○	
	2	Bastion Server	4.5		○		○	
	3	Version Control	6.0		○		○	
	4	Continuous Integration	3.0		○		○	
	5	Continuous Delivery	3.0		○		○	
	6	Content Delivery Network	7.5		○		○	
	7	CI/CD Pipeline	3.0		○		○	
	8	Application Load Balancer	3.0		○		○	
	9	Application	6.0		○		○	
합 계			40					

2) 채점방법 및 기준

과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
제2과제	1	Networking	1	VPC 확인	1.0
			2	Subnets 확인	1.5
			3	Route Tables 확인	1.5
	2	Bastion Server	1	인스턴스 타입 확인	1.5
			2	Public IP 확인	1.5
			3	OS 확인	1.5
	3	Version Control	1	Front-End CodeCommit 구성	1.5
			2	Back-End CodeCommit 구성	1.5
			3	Front-End Branch 구성	1.5
			4	Back-End Branch 구성	1.5
	4	Continuous Integration	1	Back-End CodeBuild 구성	1.5
			2	Back-End CodeBuild 로깅 구성	1.5
	5	Continuous Delivery	1	Back-End CodeDeploy 구성	1.5
			2	Back-End CodeDeploy ECS 구성	1.5
	6	Content Delivery Network	1	S3 구성	1.5
			2	CloudFront 구성	1.5
			3	CloudFront Origin 구성	1.5
			4	CloudFront Edge 구성	1.5
			5	CloudFront Cache 구성	1.5
	7	CI/CD Pipeline	1	Front-End CodePipeline 구성	1.5
			2	Back-End CodePipeline 구성	1.5
	8	Application Load Balancer	1	ALB 생성 확인	1.5
			2	ALB Scheme 확인	1.5
	9	Application	1	Front-End 동작 테스트	1.5
			2	Back-End 동작 테스트	1.5
			3	Front-End 배포 테스트	1.5
			4	Back-End 배포 테스트	1.5
	총점				40

3) 채점 내용

순번	채점 항목
1-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-vpcs --filter Name=tag:Name,Values=skills-vpc \ --query "Vpcs[].CidrBlock"</pre> <p>3) 172.16.0.0/16이 출력되는지 확인합니다.</p>
1-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-public-subnet-a \ --query "Subnets[].[AvailabilityZone, CidrBlock][]"</pre> <p>3) ap-northeast-2a와 172.16.1.0/24가 출력되는지 확인합니다.</p> <p>4) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-public-subnet-b \ --query "Subnets[].[AvailabilityZone, CidrBlock][]"</pre> <p>5) ap-northeast-2b와 172.16.2.0/24가 출력되는지 확인합니다.</p> <p>6) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-private-subnet-a \ --query "Subnets[].[AvailabilityZone, CidrBlock][]"</pre> <p>7) ap-northeast-2a와 172.16.11.0/24가 출력되는지 확인합니다.</p> <p>8) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-private-subnet-b \ --query "Subnets[].[AvailabilityZone, CidrBlock][]"</pre> <p>9) ap-northeast-2b와 172.16.12.0/24가 출력되는지 확인합니다.</p>


순번	채점 항목
1-3	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-public-rtb \ --query "RouteTables[].Routes[].GatewayId"</pre> <p>3) igw- 로 시작하는 문구가 출력되는지 확인합니다.</p> <p>4) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-internet-gateways --filter Name=tag:Name,Values=skills-igw \ --query "InternetGateways[].InternetGatewayId"</pre> <p>5) igw- 로 시작하는 문구가 2)에서 출력된 문구와 동일한지 확인합니다.</p> <p>6) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-private-rtb-a \ --query "RouteTables[].Routes[].NatGatewayId"</pre> <p>7) nat- 로 시작하는 문구가 출력되는지 확인합니다.</p> <p>8) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-nat-gateways --filter Name=tag:Name,Values=skills-nat-a \ --query "NatGateways[].NatGatewayId"</pre> <p>9) nat- 로 시작하는 문구가 6)에서 출력된 문구와 동일한지 확인합니다.</p> <p>10) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-private-rtb-b \ --query "RouteTables[].Routes[].NatGatewayId"</pre> <p>11) nat- 로 시작하는 문구가 출력되는지 확인합니다.</p> <p>12) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-nat-gateways --filter Name=tag:Name,Values=skills-nat-b \ --query "NatGateways[].NatGatewayId"</pre> <p>13) nat- 로 시작하는 문구가 10)에서 출력된 문구와 동일한지 확인합니다.</p>
2-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=skills-bastion-ec2 \ --query "Reservations[].Instances[].InstanceType"</pre> <p>3) t3a.small이 출력되는지 확인합니다.</p>

순번	채점 항목
2-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=skills-bastion-ec2 \ --query "Reservations[].Instances[].PublicIpAddress"</pre> <p>3) 2)에서 출력된 IP를 기록합니다.</p> <p>4) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-addresses --query "Addresses[].PublicIp"</pre> <p>5) 출력되는 IP 리스트 중에 3)에서 기록한 IP가 존재하는지 확인합니다.</p>
2-3	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=skills-bastion-ec2 \ --query "Reservations[].Instances[].ImageId"</pre> <p>3) ami-로 시작하는 AMI ID를 기록합니다.</p> <p>4) 아래 명령어를 입력합니다.</p> <pre>aws ec2 describe-images --image-ids <2)에서 기록한 AMI ID> --query "Images[].Description"</pre> <p>5) Amazon Linux 2023이 포함된 문구가 출력되는지 확인합니다.</p>
3-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws codecommit get-repository --repository-name skills-frontend-code \ --query "repositoryMetadata.repositoryName"</pre> <p>3) skills-frontend-code가 출력되는지 확인합니다.</p>
3-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws codecommit get-repository --repository-name skills-backend-code \ --query "repositoryMetadata.repositoryName"</pre> <p>3) skills-backend-code가 출력되는지 확인합니다.</p>
3-3	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws codecommit get-repository --repository-name skills-frontend-code \ --query "repositoryMetadata.defaultBranch"</pre> <p>3) main이 출력되는지 확인합니다.</p>

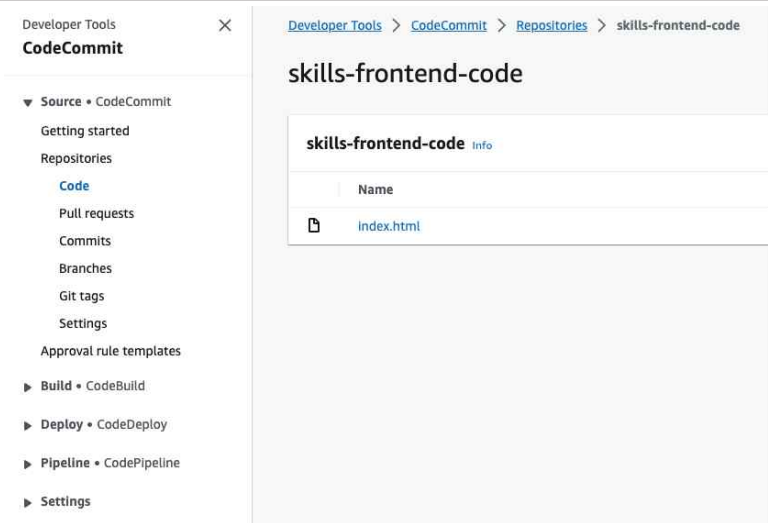
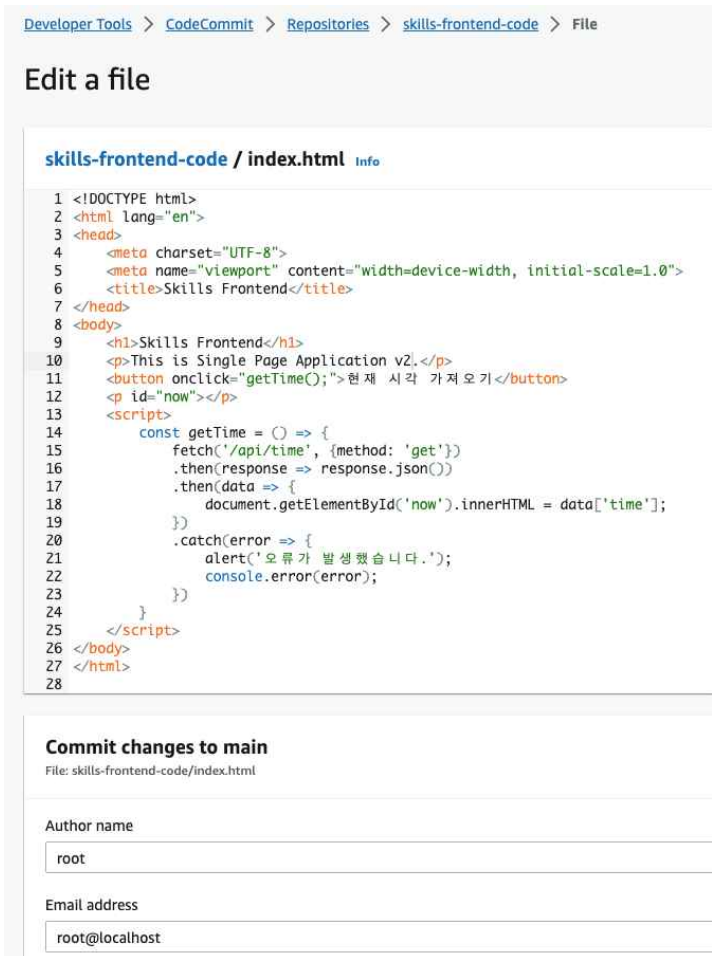
순번	채점 항목
3-4	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws codecommit get-repository --repository-name skills-backend-code \ --query "repositoryMetadata.defaultBranch"</pre> <p>3) main이 출력되는지 확인합니다.</p>
4-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws codebuild batch-get-projects --names skills-backend-build --query "projects[*].name"</pre> <p>3) skills-backend-build가 출력되는지 확인합니다.</p>
4-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws codebuild batch-get-projects --names skills-backend-build \ --query "projects[*].logsConfig.cloudWatchLogs.status"</pre> <p>3) ENABLED가 출력되는지 확인합니다.</p>
5-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws deploy get-application --application-name skills-backend-app \ --query "application.applicationName"</pre> <p>3) skills-backend-app이 출력되는지 확인합니다.</p>
5-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws deploy get-deployment-group --application-name skills-backend-app \ --deployment-group-name skills-backend-dg --query "deploymentGroupInfo.ecsServices[0]"</pre> <p>3) {"serviceName": "backend","clusterName": "skills-ecs-cluster"} 가 출력되는지 확인합니다.</p>
6-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws s3api list-buckets --query "Buckets[*].Name" grep "skills-frontend-"</pre> <p>3) skills-frontend-<랜덤 문자 4자리>가 출력되는지 확인합니다.</p>

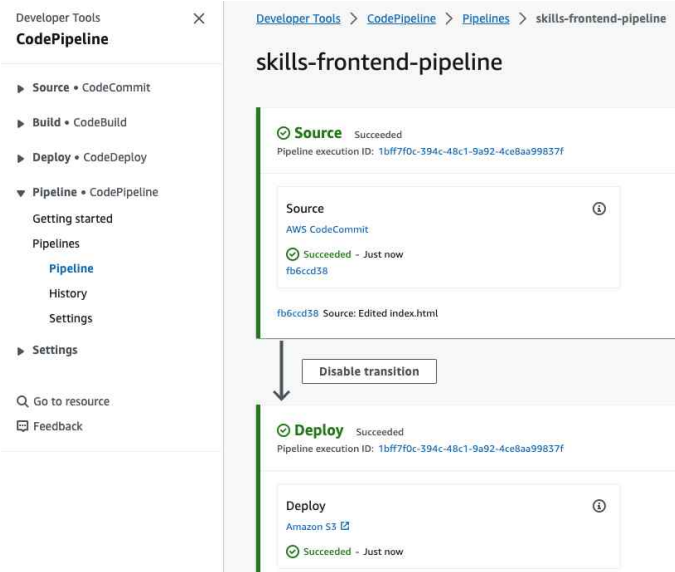
순번	채점 항목
6-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws resourcegroupstaggingapi get-resources \ --tag-filters Key=Name,Values=skills-frontend-cdn \ --resource-type-filters 'cloudfront' --region us-east-1 \ --query "ResourceTagMappingList[0].ResourceARN" \ --output text sed 's:./::' xargs -I {} aws cloudfront get-distribution --id {} \ --query "Distribution.DomainName"</pre> <p>3) XXXXXXXXXXXX.cloudfront.net이라는 도메인이 출력되는지 확인합니다.</p>
6-3	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws resourcegroupstaggingapi get-resources \ --tag-filters Key=Name,Values=skills-frontend-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.Origins.Items[*].DomainName" \ grep "skills-frontend-"</pre> <p>3) skills-frontend-<랜덤 문자 4자리>.s3.ap-northeast-2.amazonaws.com이라는 문구가 출력되는지 확인합니다.</p>
6-4	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws resourcegroupstaggingapi get-resources \ --tag-filters Key=Name,Values=skills-frontend-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> <p>3) PriceClass_All이라는 도메인이 출력되는지 확인합니다.</p>

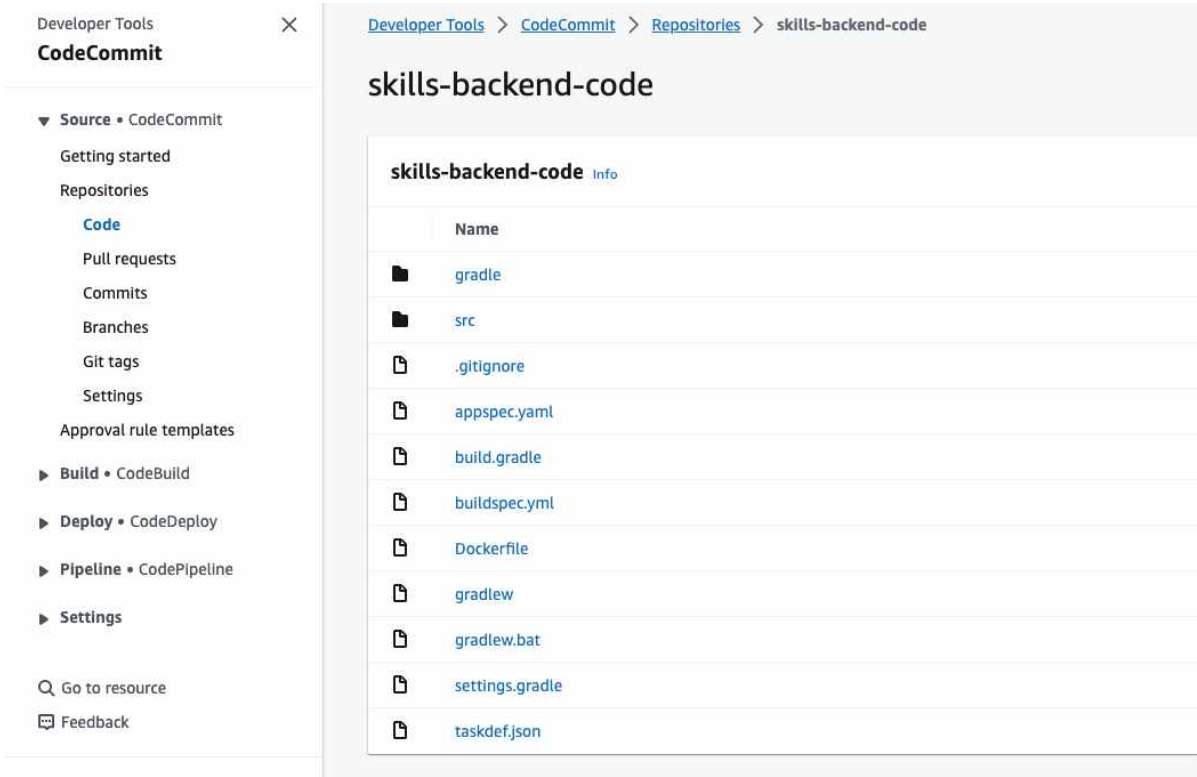
순번	채점 항목
6-5	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws resourcegroupstaggingapi get-resources \ --tag-filters Key=Name,Values=skills-frontend-cdn \ --resource-type-filters 'cloudfront' --region us-east-1 \ --query "ResourceTagMappingList[0].ResourceARN" \ --output text sed 's:./:.' xargs -I {} aws cloudfront get-distribution --id {} \ --query "Distribution.DomainName"</pre> <p>3) XXXXXXXXXXXX.cloudfront.net이라는 도메인이 출력되는지 확인합니다.</p> <p>4) 아래 명령어를 입력합니다.</p> <pre>curl --silent --head https://<3>에서 출력된 도메인>/index.html grep "x-cache"</pre> <p>5) x-cache: Hit from cloudfront이라는 문구가 출력되는지 확인합니다. 최대 5번까지 실행해볼 수 있습니다.</p>
7-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws codepipeline get-pipeline --name skills-frontend-pipeline \ --query "pipeline.stages[*].[name, actions[*].actionTypeId.provider]"</pre> <p>3) Source, Deploy가 출력되고, 각각 CodeCommit, S3가 출력되는지 확인합니다. 3개를 제외한 다른 것들도 출력되도 무관하나, 3개는 반드시 출력되어야 합니다.</p>
7-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws codepipeline get-pipeline --name skills-backend-pipeline \ --query "pipeline.stages[*].[name, actions[*].actionTypeId.provider]"</pre> <p>3) Source, Build, Deploy가 출력되고, 각각 CodeCommit, CodeBuild, CodeDeployToECS가 출력되는지 확인합니다. 3개를 제외한 다른 것들은 출력되어서는 안됩니다.</p>
8-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws elbv2 describe-load-balancers --names skills-user-alb \ --query "LoadBalancers[].LoadBalancerName"</pre> <p>3) skills-user-alb가 출력되는지 확인합니다.</p>

순번	채점 항목
8-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws elbv2 describe-load-balancers --names skills-user-alb \ --query "LoadBalancers[].Scheme"</pre> <p>3) internet-facing이 출력되는지 확인합니다.</p>
9-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws resourcegroupstaggingapi get-resources \ --tag-filters Key=Name,Values=skills-frontend-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"</pre> <p>3) XXXXXXXXXXXX.cloudfront.net이라는 도메인이 출력되는지 확인합니다.</p> <p>4) https://<3)에서 출력된 도메인>/index.html을 웹 브라우저를 통해 접속합니다.</p> <div>  </div> <p>5) 웹 브라우저에서 위와 같은 화면이 출력되는지 확인합니다.</p>

순번	채점 항목
9-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws resourcegroupstaggingapi get-resources \ --tag-filters Key=Name,Values=skills-frontend-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"</pre> <p>3) XXXXXXXXXXXX.cloudfront.net이라는 도메인이 출력되는지 확인합니다.</p> <p>4) https://<3)에서 출력된 도메인>/을 웹 브라우저를 통해 접속합니다.</p> <div> <h1>Skills Frontend</h1> <p>This is Single Page Application v1.</p> <div>현재 시각 가져오기</div> </div> <p>5) 웹 브라우저에서 위와 같은 화면이 출력되면, ‘현재 시각 가져오기’를 클릭합니다.</p> <div> <h1>Skills Frontend</h1> <p>This is Single Page Application v1.</p> <div>현재 시각 가져오기</div> <p>2023-09-26T13:36:51.974525178</p> </div> <p>6) 위와 같이 오류 없이 하단에 현재 시각이 출력되는지 확인합니다.</p>

순번	채점 항목
9-3	<div data-bbox="212 241 983 763">  </div> <p>1) 위와 같이 웹 브라우저를 통해 AWS Console로 접속한 후, CodeCommit의 skills-frontend-code 리포지토리로 접근합니다.</p> <div data-bbox="212 891 927 1845">  </div> <p>2) index.html에서 Edit을 클릭한 후, v1을 v2라는 문구로 변경합니다. Author name은 root, Email address는 root@localhost를 입력한 후, 하단의 Commit changes를 클릭해 커밋을 실행합니다.</p>

순번	채점 항목
9-3	<div data-bbox="215 246 893 817">  </div> <p>3) CodePipeline의 skills-frontend-pipeline으로 접근합니다. 방금 새로운 커밋을 실행하였으므로 위와 같이 파이프라인이 동작중이어야 합니다. 파이프라인이 실패 없이 15분만에 완료되는지 확인합니다. 파이프라인이 완료되면 채점을 재개합니다.</p> <p>4) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>5) 아래 명령어를 입력합니다.</p> <pre>aws resourcegroupstaggingapi get-resources \ --tag-filters Key=Name,Values=skills-frontend-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"</pre> <p>6) XXXXXXXXXXXXXXX.cloudfront.net이라는 도메인이 출력되는지 확인합니다.</p> <p>7) https://<3>에서 출력된 도메인>/index.html을 웹 브라우저를 통해 접속합니다.</p> <div data-bbox="255 1556 893 1646"> <h1>Skills Frontend</h1> </div> <div data-bbox="255 1724 941 1780"> <h2>This is Single Page Application v2.</h2> </div> <div data-bbox="263 1825 606 1892"> <div>현재 시각 가져오기</div> </div> <p>8) 웹 브라우저에서 위와 같이 v2가 출력되는지 확인합니다.</p>

순번	채점 항목
9-4	<div data-bbox="212 286 1415 1061">  </div> <p>1) 위와 같이 웹 브라우저를 통해 AWS Console로 접속한 후, CodeCommit의 skills-backend-code 리포지토리로 접근합니다.</p>

순번	채점 항목
9-4	<div data-bbox="233 255 1225 1240"> <p>Developer Tools > CodeCommit > Repositories > skills-backend-code > File</p> <p>Edit a file</p> <p>skills-backend-code / src / main / java / org / worldskills / cloud / skillsbackend / SkillsBackendApplication.java Info</p> <pre> 1 package org.worldskills.cloud.skillsbackend; 2 3 import org.slf4j.Logger; 4 import org.slf4j.LoggerFactory; 5 import org.springframework.boot.SpringApplication; 6 import org.springframework.boot.autoconfigure.SpringBootApplication; 7 import org.springframework.http.HttpStatus; 8 import org.springframework.http.ResponseEntity; 9 import org.springframework.web.bind.annotation.CrossOrigin; 10 import org.springframework.web.bind.annotation.GetMapping; 11 import org.springframework.web.bind.annotation.RestController; 12 13 import java.time.LocalDateTime; 14 import java.util.HashMap; 15 import java.util.Map; 16 17 @SpringBootApplication 18 @RestController 19 public class SkillsBackendApplication { 20 private final Logger log = LoggerFactory.getLogger(this.getClass().getSimpleName()); 21 22 public static void main(String[] args) { 23 SpringApplication.run(SkillsBackendApplication.class, args); 24 } 25 26 @GetMapping("/api/") 27 public ResponseEntity<String> index() { 28 String text = "Hello, World!"; 29 30 log.info("GET / 200"); 31 32 return new ResponseEntity<>(text, HttpStatus.OK); 33 } 34 35 @GetMapping("/api/time") 36 public ResponseEntity<Map<String, Object>> time() { 37 Map<String, Object> data = new HashMap<>(); 38 39 LocalDateTime currentDateTime = LocalDateTime.now(); 40 41 data.put("time", currentDateTime + " v2"); 42 43 log.info("GET /time 200"); 44 45 return new ResponseEntity<>(data, HttpStatus.OK); 46 } 47 48 @GetMapping("/api/health") 49 public ResponseEntity<Map<String, Object>> health() { 50 Map<String, Object> data = new HashMap<>(); 51 data.put("status", "OK"); 52 } </pre> </div> <p>2) src/main/java/org/worldskills/cloud/skillsbackend/SkillsBackendApplication.java에 서 Edit을 클릭한 후, currentDateTime 뒤에 v2라는 문구를 입력합니다. Author name은 root, Email address는 root@localhost를 입력한 후, 하단의 Commit changes를 클릭해 커밋을 실행 합니다.</p>

순번	채점 항목
9-4	<div data-bbox="213 241 970 1279"> <p>The screenshot shows the AWS CodePipeline console for a pipeline named 'skills-backend-pipeline'. The left sidebar shows the navigation menu with 'CodePipeline' selected. The main area displays the pipeline's execution history. The 'Source' stage (AWS CodeCommit) is 'Succeeded'. The 'Build' stage (AWS CodeBuild) is 'In progress'. The 'Deploy' stage (Amazon ECS) is 'Succeeded'. The pipeline execution ID is 'ffa0a15f-c025-4cf1-bef0-05c3963b74eb'.</p> </div> <p>3) CodePipeline의 skills-backend-pipeline으로 접근합니다. 방금 새로운 커밋을 실행하였으므로 위와 같이 파이프라인이 동작중이어야 합니다. 파이프라인이 실패 없이 15분만에 완료되는지 확인합니다. 파이프라인이 완료되면 채점을 재개합니다.</p> <p>4) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>5) 아래 명령어를 입력합니다.</p> <pre>aws resourcegroupstaggingapi get-resources \ --tag-filters Key=Name,Values=skills-frontend-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"</pre> <p>6) XXXXXXXXXXXX.cloudfront.net이라는 도메인이 출력되는지 확인합니다.</p> <p>7) https://<3>에서 출력된 도메인>/index.html을 웹 브라우저를 통해 접속합니다.</p>

순번	채점 항목
9-4	<div data-bbox="220 286 790 369"> <h1>Skills Frontend</h1> </div> <div data-bbox="220 436 833 488"> <p>This is Single Page Application v2.</p> </div> <div data-bbox="225 533 534 589"> <div>현재 시각 가져오기</div> </div> <div data-bbox="209 656 1279 694"> <p>8) 웹브라우저에서 위와 같은 화면이 출력되면, ‘현재 시각 가져오기’를 클릭합니다.</p> </div> <div data-bbox="229 743 938 835"> <h1>Skills Frontend</h1> </div> <div data-bbox="229 927 992 987"> <p>This is Single Page Application v2.</p> </div> <div data-bbox="234 1039 620 1113"> <div>현재 시각 가져오기</div> </div> <div data-bbox="229 1169 1018 1223"> <p>2023-09-26T14:17:05.584585871v2</p> </div> <div data-bbox="209 1323 1209 1359"> <p>9) 위와 같이 오류 없이 하단에 현재 시각이 v2와 함께 출력되는지 확인합니다.</p> </div>