

2022 경상남도 제57회 전국기능경기대회 채점기준

1. 채점상의 유의사항	직 종 명	클라우드컴퓨팅
<p>※ 다음 사항을 유의하여 채점하십시오.</p> <ol style="list-style-type: none"> 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 리소스 수정이 있는 문항의 경우 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. 		

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	네트워크 구성	9.8		○		○	
	2	Code Commit	3		○		○	
	3	Code Build	1.5		○		○	
	4	Code Deploy	4.4		○		○	
	5	Code Pipeline	4.5		○		○	
	6	Route53	3		○		○	
	7	Load Balancer	3		○		○	
	8	Application	1.5		○		○	
	9	Deployment	7.5		○		○	
	10	Bastion	1.8		○		○	
합 계			40					

2) 채점방법 및 기준

과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
제2과제	1	네트워크 구성	1	VPC 생성	0.7
			2	CIDR private a 구성	0.7
			3	CIDR private b 구성	0.7
			4	CIDR private c 구성	0.7
			5	CIDR public a 구성	0.7
			6	CIDR public b 구성	0.7
			7	CIDR public c 구성	0.7
			8	AZ A-zone 구성	0.7
			9	AZ B-zone 구성	0.7
			10	AZ C-zone 구성	0.7
			11	Interget G/W 구성	0.7
			12	NAT G/W A-zone 구성	0.6
			13	NAT G/W B-zone 구성	0.75
			14	NAT G/W C-zone 구성	0.75
	2	Code Commit	1	Code commit 생성	1.5
			2	Branch 구성	1.5
	3	Code Build	1	Code Build 생성	1.5
	4	Code Deploy	1	Code deploy 생성	1.1
			2	Code deploy 타입 구성	1.1
			3	Code deploy ECS 구성	1.1
			4	Code deploy Target Group	1.1
	5	Code Pipeline	1	Pipeline source 구성	1.5
			2	Pipeline build 구성	1.5
			3	Pipeline deploy 구성	1.5
	6	Route53	1	Hosted zone 구성	1.5
			2	Record 구성	1.5
	7	Load Balancer	1	Load Balancer Type	1.5
			2	Load Balancer AZ 구성	1.5
	8	Application	1	동작 테스트	1.5
	9	Deployment	1	CI/CD	1.5
			2	ECR 업로드 확인	1.5
			3	배포 어플리케이션 확인	1.5
			4	배포 롤백 파이프라인 확인	1.5
			5	배포 확인	1.5
	10	Bastion	1	Bastion 생성	0.8
			2	EIP	1
	총점				40


3) 채점내용

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) 10.10.0.0/16이 출력되는지 확인합니다.</p>
1-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 10.10.0.0/24가 출력되는지 확인합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-priv-a --query \$` "Subnets[].CidrBlock"</pre>
1-3	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 10.10.1.0/24가 출력되는지 확인합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-priv-b --query \$` "Subnets[].CidrBlock"</pre>
1-4	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 10.10.2.0/24가 출력되는지 확인합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-priv-c --query \$` "Subnets[].CidrBlock"</pre>
1-5	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 10.10.10.0/24가 출력되는지 확인합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-pub-a --query \$` "Subnets[].CidrBlock"</pre>
1-6	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 10.10.11.0/24가 출력되는지 확인합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-pub-b --query \$` "Subnets[].CidrBlock"</pre>
1-7	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 10.10.12.0/24가 출력되는지 확인합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-pub-c --query \$` "Subnets[].CidrBlock"</pre>

순번	채점 항목
1-8	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 ap-northeast-2a가 출력되는지 확인합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-priv-a --query \backslashSubnets[].AvailabilityZone</pre>
1-9	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 ap-northeast-2b가 출력되는지 확인합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-priv-b --query \backslashSubnets[].AvailabilityZone</pre>
1-10	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 ap-northeast-2c가 출력되는지 확인합니다.</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-priv-c --query \backslashSubnets[].AvailabilityZone</pre>
1-11	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 igw- 로 시작하는 문구가 출력되는지 확인합니다.</p> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-pub-rt \backslash--query "RouteTables[].Routes[].GatewayId"</pre>
1-12	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 nat- 로 시작하는 문구가 출력되는지 확인합니다.</p> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-priv-a-rt \backslash--query "RouteTables[].Routes[].NatGatewayId"</pre>
1-13	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 nat- 로 시작하는 문구가 출력되는지 확인합니다. 그리고 1-12에서 출력된 Id와 틀린지도 확인합니다.</p> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-priv-b-rt \backslash--query "RouteTables[].Routes[].NatGatewayId"</pre>
1-14	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 nat- 로 시작하는 문구가 출력되는지 확인합니다. 그리고 1-13에서 출력된 Id와 틀린지도 확인합니다.</p> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-priv-c-rt \backslash--query "RouteTables[].Routes[].NatGatewayId"</pre>

순번	채점 항목
2-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 skills-commit 라고 출력되는 부분이 있는지 확인합니다.</p> <pre>aws codecommit list-repositories</pre>
2-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 upstream 이라고 출력되는지 확인합니다.</p> <pre>aws codecommit get-branch --repository-name skills-commit --branch-name upstream --query branch.branchName</pre>
3-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 skills-build로 출력되는 것이 있는지 확인합니다.</p> <pre>aws codebuild list-projects --query projects</pre>
4-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 ECS라고 출력되는지 확인합니다.</p> <pre>aws deploy get-application --application-name skills-app --query application.computePlatform</pre>
4-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력 후 BLUE/GREEN 이라고 출력되는지 확인합니다.</p> <pre>aws deploy get-deployment-group --application-name skills-app --deployment-group-name ₩ skills-dg --query deploymentGroupInfo.deploymentStyle.deploymentType</pre>
4-3	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws deploy get-deployment-group --application-name skills-app --deployment-group-name ₩ skills-dg --query deploymentGroupInfo.ecsServices</pre> <p>3) 출력 결과가 아래와 같은지 확인합니다.</p> <pre>"clusterName": "skills-cluster", "serviceName": "skills-svc"</pre>
4-4	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws deploy get-deployment-group --application-name skills-app --deployment-group-name ₩ skills-dg --query "deploymentGroupInfo.loadBalancerInfo.targetGroupPairInfoList[].targetGroups[]"</pre> <p>3) 출력된 내용에 아래 내용을 포함하고 있는지 확인합니다.</p> <pre>"name": "skills-tg1" "name": "skills-tg2"</pre>

순번	채점 항목
5-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어 입력 후 skills-commit 이라고 출력되는지 확인합니다.</p> <pre>aws codepipeline get-pipeline --name skills-pipeline --query ₩ "pipeline.stages[0].actions[].configuration.RepositoryName"</pre>
5-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어 입력후 skills-build 라고 출력되는지 확인합니다.</p> <pre>aws codepipeline get-pipeline --name skills-pipeline --query ₩ "pipeline.stages[1].actions[].configuration.ProjectName"</pre>
5-3	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어 입력 후 skills-app 이라고 출력되는지 확인합니다.</p> <pre>aws codepipeline get-pipeline --name skills-pipeline --query ₩ "pipeline.stages[2].actions[].configuration.ApplicationName"</pre> <p>4) 아래 명령어 입력 후 skills-dg라고 출력되는지 확인합니다.</p> <pre>aws codepipeline get-pipeline --name skills-pipeline --query ₩ "pipeline.stages[2].actions[].configuration.DeploymentGroupName"</pre>
6-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어 입력 후 true 로 출력되는지 확인합니다.</p> <pre>aws route53 list-hosted-zones --query 'HostedZones[?Name == `skills.local.`].Config.PrivateZone'</pre>
6-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) nslookup app.skills.local을 입력합니다.</p> <p>3) 에러 없이 IP를 잘 출력 하는지 확인합니다.</p>
7-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어 입력 후 application으로 출력되는지 확인합니다.</p> <pre>aws elbv2 describe-load-balancers --query 'LoadBalancers[?LoadBalancerName == `skills-alb`].Type'</pre>
7-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력합니다.</p> <pre>aws elbv2 describe-load-balancers --query ₩ 'LoadBalancers[?LoadBalancerName == `skills-alb`].AvailabilityZones[].ZoneName'</pre> <p>3) 아래와 같이 3개의 AZ가 출력되는지 확인합니다.</p> <pre>"ap-northeast-2a", "ap-northeast-2b", "ap-northeast-2c"</pre>

순번	채점 항목						
8-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력해 BLUE라고 출력되는지 확인합니다.</p> <pre>curl http://app.skills.local/v1/dummy</pre>						
9-1	<p>1) AWS web console로 이동합니다.</p> <p>2) Code Commit 페이지로 이동해 skills-commit을 선택합니다.</p> <p>3) upstream branch의 main.go 파일을 엽니다.</p> <p>4) 파일 내용 중 아래와 같이 13번 줄 BLUE 부분을 GREEN으로 수정합니다.</p> <pre>12 func dummy(w http.ResponseWriter, req *http.Request) { 13 fmt.Fprint(w, "GREEN") 14 }</pre> <p>5) Code Pipeline 페이지로 이동하여 skills-pipeline을 선택합니다.</p> <p>6) 실행된 시간을 확인해 방금 코드 변경 후 시작된 것인지 확인합니다. Source, Build, Deploy 상태가 모두 초록색 Succeeded로 변경되는지 확인하며 완료까지 최대 7분 정도 소요될 수 있습니다.</p>  <p>8) Deploy의 details를 클릭한 후 Deployment status에서 Step1 ~ Step4까지 Succeeded 인지, Traffic shifting progress에서 Replacement가 100%인지 확인합니다.</p> <div> <div> <p>Deployment status</p> <p>Step 1: Deploying replacement task set</p> <p>Completed ✔ Succeeded 100%</p> <p>Step 2: Test traffic route setup</p> <p>Completed ✔ Succeeded 100%</p> <p>Step 3: Rerouting production traffic to replacement task set</p> <p>100% traffic shifted ✔ Succeeded 100%</p> <p>Step 4: Terminate original task set</p> <p>Completed ✔ Succeeded 100%</p> </div> <div> <p>Traffic shifting progress</p> <table border="1"> <thead> <tr> <th>Original</th> <th>Replacement</th> </tr> </thead> <tbody> <tr> <td>0.%</td> <td>100.%</td> </tr> <tr> <td>Original task set not serving traffic</td> <td>Replacement task set</td> </tr> </tbody> </table> </div> </div>	Original	Replacement	0.%	100.%	Original task set not serving traffic	Replacement task set
Original	Replacement						
0.%	100.%						
Original task set not serving traffic	Replacement task set						
9-2	<p>1) AWS web console로 이동합니다.</p> <p>2) ECR 페이지로 이동합니다.</p> <p>3) skills-ecr을 선택합니다.</p> <p>4) 아래와 같은 형식으로 방금 이미지가 업로드 되었는지 확인합니다. 시간은 한국시간 기준입니다.</p> <p>년-월-일.시.분.초 (2022-06-26.09.01.59)</p> <div> <div>2022-06-26.09.26.29</div> <div>Image</div> <div>2 5 1</div> </div>						

순번	채점 항목												
9-3	<div>1) SSH를 통해 Bastion 서버에 접근합니다.</div> <div>2) 아래 명령어를 입력하여 GREEN 으로 출력되는지 확인합니다.</div> <div>curl http://app.skills.local/v1/dummy</div>												
9-4	<div>1) AWS web console로 이동합니다.</div> <div>2) Code Commit 페이지로 이동합니다.</div> <div>3) skills-commit을 선택합니다.</div> <div>4) upstream branch의 main.go 파일을 엽니다.</div> <div>5) 파일 내용 중 아래와 같이 2곳을 수정합니다. 아래 그림과 같이 13번째 줄에 GREEN으로 된 부분을 RED로, 19번째 줄의 :80을 :88로 변경 후 저장합니다.</div> <div><pre>12 func dummy(w http.ResponseWriter, req *http.Request) { 13 fmt.Fprint(w, "RED") 14 } 15 16 // Handle dummy request, return dummy response 17 18 http.HandleFunc("/v1/dummy", dummy) 19 http.ListenAndServe(":88", nil)</pre></div> <div>6) Code Pipeline 페이지로 이동하여 skills-pipeline을 선택합니다.</div> <div>7) 각 단계를 확인해 방금 코드 변경 후 Pipeline이 실행되었는지 확인합니다.</div> <div>8) Source와 Build가 성공적으로 완료되어 Succeeded로 변경되고, Deploy 단계가 계속 In progress 인지 확인합니다. In progress 상태가 3분 이상 - 5분 이하 일 때 Details를 클릭합니다.</div> <div><div><div>Deploy</div><div><div>Amazon ECS (Blue/Green)</div><div><div>In progress - 4 minutes ago</div><div>Details</div></div></div></div></div> <div>9) 아래 그림처럼 Install 부분이 In progress로 보이는지 확인합니다.</div> <div><table><tr><th>Event</th><th>Duration</th><th>Status</th></tr><tr><td>BeforeInstall</td><td>less than one second</td><td><div>Succeeded</div></td></tr><tr><td>Install</td><td>-</td><td><div>In progress</div></td></tr><tr><td>AfterInstall</td><td>-</td><td><div>Pending</div></td></tr></table></div>	Event	Duration	Status	BeforeInstall	less than one second	<div>Succeeded</div>	Install	-	<div>In progress</div>	AfterInstall	-	<div>Pending</div>
Event	Duration	Status											
BeforeInstall	less than one second	<div>Succeeded</div>											
Install	-	<div>In progress</div>											
AfterInstall	-	<div>Pending</div>											
9-5	<div>1) SSH를 통해 Bastion 서버에 접근합니다.</div> <div>2) 아래 명령어를 입력하여 GREEN으로 출력되는지 확인합니다. BLUE나 RED로 출력되면 실패입니다.</div> <div>curl http://app.skills.local/v1/dummy</div>												

순번	채점 항목
10-1	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력하여 "-i"로 시작하는 결과가 있는지 확인합니다.</p> <pre>aws ec2 describe-instance --filter Name=tag:Name,Values=skills-bastion2 \\ --query 'Reservations[].Instances[].InstanceId'</pre>
10-2	<p>1) SSH를 통해 Bastion 서버에 접근합니다.</p> <p>2) 아래 명령어를 입력해 나오는 IP를 기록합니다.</p> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=skills-bastion2 \\ --query "Reservations[].Instances[].PublicIpAddress"</pre> <p>3) 아래 명령어 입력 후 출력되는 IP 중 2)번에서 출력된 IP가 있는지 확인합니다.</p> <pre>aws ec2 describe-addresses --query "Addresses[].PublicIp"</pre>