

2025 광주광역시 제60회 전국기능경기대회 채점기준

1. 채점상의 유의사항	직 종 명	클라우드컴퓨팅
<p>※ 다음 사항을 유의하여 채점하십시오.</p> <ol style="list-style-type: none"> 1) AWS의 지역은 과제에 명시된 지역을 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) (예상 출력)은 바로 이전 (명령어 입력)의 예상 출력을 의미합니다. 12) 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. 채점 스크립트는 root 경로에 지정하도록 합니다. 13) 배포된 채점 스크립트(mark.sh) 는 ec2-user에 최상위 경로에 위치 하도록 합니다. 		

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	S3 Multi-Region Access Point	7.5		○		○	
	2	VPC Lattice	7.5		○		○	
	3	Actions Runner Controller	7.5		○		○	
	4	ECS Firelens	7.5		○		○	
합 계			30					

2) 채점방법 및 기준

과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
2과제	1	S3 Multi-Region Access Point	1-1	S3 Configure	0.5
			1-2	S3 Policy Check	0.5
			2-1	Instance Type	0.25
			2-2	Public IP	0.25
			2-3	Instance Configure	0.25
			2-4	Attached Policies	0.25
			3-1	CF Domain Check	0.25
			3-2	CF Origin Check	0.25
			3-3	CF Price Class Check	0.5
			3-4	CF Function Name Check	0.5
			4-1	Lambda Configure	0.5
			4-2	Lambda Role Name Check	0.5
			5-1	CF invalidations Test	0.5
			5-2	CF invalidations Test	0.5
			6-1	MRAP Latency Check	0.5
			7-1	CF Country Block Check	0.375
			7-2	CF Static Access Check - Seoul	0.375
			7-3	CF Static Access Check - Tokyo	0.375
			7-4	CF Header Block Check	0.375

2과제	2	VPC Lattice	1-1	VPC	0.5
			1-2	Subnet	0.5
			1-3	Routing Table	0.5
			2-1	VPC Lattice Service Network Check	0.5
			2-2	VPC Lattice Service Check	0.5
			2-3	VPC Lattice Target Group Check	0.5
			3-1	Instance Type	0.125
			3-2	Public IP	0.125
			3-3	Instance Configure	0.125
			3-4	Attached Policies	0.125
			4-1	Load Balancer Configure	1.5
			5-1	App Health Check	0.3
			5-2	App API Test	0.3
			5-3	App API Test	0.3
			5-4	App API Test	0.3
			5-5	App API Test	0.3
			6-1	DynamoDB Table	0.25
			6-2	Table Mode	0.25
			6-3	Table Protection	0.5

2과제	3	Actions Runner Controller	1-1	VPCs, Subnets Configure	0.1
			2-1	EKS Cluster Configure	0.1
			2-2	Running Application Pods	0.1
			2-3	ArgoCD Helm Chart	0.3
			3-1	Repository Configure	0.1
			3-2	Github IAM Configure	0.5
			4-1	Repository	0.1
			4-2	Repository Items	0.2
			4-3	app Chart Items	0.3
			4-4	Branches	0.2
			4-5	Default Branch	0.2
			4-6	Label	0.2
			4-7	Workflows	0.6
			4-8	Self-hosted Runners	0.5
			5-1	Dev Pipeline Test	0.6
			5-2	Dev ECR Artifact Type	0.5
			5-3	Dev Pipeline Duration	0.9
			5-4	Prod Pipeline Test	0.6
			5-5	Prod ECR Artifact Type	0.5
			5-6	Prod Pipeline Duration	0.9

2과제	4	ECS Firelens	1-1	VPC	0.1
			1-2	Subnets	0.1
			1-3	Route Tables	0.1
			2-1	Bastion Server	0.1
			3-1	ALB	0.5
			4-1	Application	0.5
			5-1	ECR	0.5
			6-1	ECS Cluster	0.5
			6-2	ECS Service	0.5
			6-3	ECS Task definition	0.6
			6-4	ECS Service Subnets	0.5
			7-1	Firelens LogDriver	1
			8-1	CloudWatch Log Groups	0.5
			8-2	CloudWatch Log Streams	1
			8-3	CloudWatch Log	1

3) 채점내용

순번	사전준비	
1	1) skills-bastion 서버에 SSH를 통해 접근합니다. 2) aws configure set default.region ap-northeast-2	
	1-1 (명령어 입력)	<pre>aws s3 ls s3://skills-kr-cdn-web-static-\$ACCOUNT_ID/ --recursive awk '{print \$4}'</pre> <pre>aws s3 ls s3://skills-us-cdn-web-static-\$ACCOUNT_ID/ --recursive awk '{print \$4}'</pre>
	1-1-A (예상 출력) <u>정확히 일치</u>	<pre>index.html</pre> <pre>index.html</pre>

순번	채점항목	
1	1-2-A (명령어 입력)	aws s3control list-multi-region-access-points \ --account-id \$ACCOUNT_ID \ --region us-west-2
	1-2-A (예상 출력) <u>정확히 일치</u> <u>이 외 JSON</u> <u>출력되면 인정</u> <u>X</u> <u>Regions</u> <u>순서무관</u>	{ "AccessPoints": [{ "Name": "skills-mrap", "Alias": "m19awd3puom5b.mrap", "CreatedAt": "2025-06-16T07:40:19.186000+00:00", "PublicAccessBlock": { "BlockPublicAcls": true, "IgnorePublicAcls": true, "BlockPublicPolicy": true, "RestrictPublicBuckets": true }, "Status": "READY", "Regions": [{ "Bucket": "skills-kr-cdn-web-static- <u>ACCOUNT_ID</u> ", "Region": "ap-northeast-2", "BucketAccountId": " <u>ACCOUNT_ID</u> " }, { "Bucket": "skills-us-cdn-web-static- <u>ACCOUNT_ID</u> ", "Region": "us-east-1", "BucketAccountId": " <u>ACCOUNT_ID</u> " }] }] } } * 밑줄 친 부분을 제외하고 전부 동일해야 함

순번	채점항목	
1	2-1-A (명령어 입력)	aws ec2 describe-instances --filter Name=tag:Name,Values=mrp-bastion --query "Reservations[].Instances[].InstanceType"
	2-1-A (예상 출력) <u>정확히 일치</u>	["t3.micro"]
1	2-2-A (명령어 입력)	aws ec2 describe-instances --filter Name=tag:Name,Values=mrp-bastion --query "Reservations[].Instances[].PublicIpAddress" aws ec2 describe-addresses --query "Addresses[].PublicIp"
	2-2-A (예상 출력) <u>같은 IP가 2개 있는지 확인</u>	[<u>"15.165.13.151"</u>] [<u>"15.165.13.151"</u>]

순번	채점항목	
1	<p>2-3-A (명령어 입력)</p>	<pre> echo =====2-3===== INSTANCE_NAME_TAG="mrapp-bastion" INSTANCE_ID=\$(aws ec2 describe-instances --filters "Name=tag:Name,Values=\$INSTANCE_NAME_TAG" --query "Reservations[0].Instances[0].InstanceId" --output text) AMI_ID=\$(aws ec2 describe-instances --instance-ids "\$INSTANCE_ID" --query "Reservations[0].Instances[0].ImageId" --output text) AMI_DESCRIPTION=\$(aws ec2 describe-images --image-ids "\$AMI_ID" --query "Images[0].Description" --output text) INSTANCE_SG_NAME=\$(aws ec2 describe-instances --instance-ids "\$INSTANCE_ID" --query "Reservations[0].Instances[0].SecurityGroups[0].GroupName" --output text) echo "\$AMI_DESCRIPTION" echo "\$INSTANCE_SG_NAME" echo </pre>
	<p>2-3-A (예상 출력) <u>부분 일치</u> <u>미줄 친부분</u> <u>일치</u></p>	<pre> Amazon Linux 2023 AMI 2023.7.20250428.1 x86_64 HVM kernel-6.1 mrapp-bastion-sg </pre>

순번	채점항목	
1	2-4-A (명령어 입력)	<pre> POLICY_ARNS=\$(aws iam list-attached-role-policies --role-name mrap-bastion-role --query "AttachedPolicies[].PolicyArn" --output text) for POLICY_ARN in \$POLICY_ARNS; do POLICY_VERSION=\$(aws iam get-policy --policy-arn \$POLICY_ARN --query "Policy.DefaultVersionId" --output text) POLICY_DOCUMENT=\$(aws iam get-policy-version --policy-arn \$POLICY_ARN --version-id \$POLICY_VERSION --query "PolicyVersion.Document" --output json) echo "\$POLICY_DOCUMENT" done </pre>
	2-4-A (예상 출력) <u>정확히 일치</u> <u>이 외 JSON</u> <u>출력되면 인정</u> X	<pre> { "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": "*", "Resource": "*" }] } </pre>

순번	채점항목	
1	3-1-A (명령어 입력)	aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=skills-global-distribution --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"
	3-1-A (예상 출력)	"d325x8sp3kdikz.cloudfront.net"
1	3-2-A (명령어 입력)	aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=skills-global-distribution --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 "accesspoint.s3-global.amazonaws.com" tr -d '[:space:]'
	3-2-A (예상 출력)	"m19awd3puom5b.mrap.accesspoint.s3-global.amazonaws.com"
1	3-3-A (명령어 입력)	aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=skills-global-distribution --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"
	3-3-A (예상 출력) 정확히 일치	"PriceClass_All"

순번	채점항목	
1	3-4-A (명령어 입력)	aws cloudfront list-functions --region us-east-1 --query "FunctionList.Items[].{Name:Name, Stage:FunctionMetadata.Stage}" --output json
	3-4-A (예상 출력) 정확히 일치 이 외 JSON 출력되면 인정 X	<pre>[{ "Name": "skills-cf-function", "Stage": "DEVELOPMENT" }, { "Name": "skills-cf-function", "Stage": "LIVE" }]</pre>
1	4-1-A (명령어 입력)	aws lambda list-functions --region us-east-1 --query "Functions[].FunctionName" --output text aws lambda list-functions --region ap-northeast-2 --query "Functions[].FunctionName" --output text
	4-1-A (예상 출력) 순서무관 정확히 일치	skills-cdn-edge-function skills-lambda-function-us skills-lambda-function-kr

순번	채점항목	
1	4-2-A (명령어 입력)	<pre>aws lambda list-functions \ --region us-east-1 \ --query "Functions[?contains(Role, 'skills-lambda-role-us')] contains(Role, 'skills-lambda-role-us')].Role" \ --output text awk -F / '{print \$NF}'</pre> <pre>aws lambda list-functions \ --region ap-northeast-2 \ --query "Functions[?contains(Role, 'skills-lambda-role-kr')] contains(Role, 'skills-lambda-role-kr')].Role" \ --output text awk -F / '{print \$NF}'</pre>
	4-2-A (예상 출력) <u>정확히 일치</u>	<pre>skills-lambda-role-us skills-lambda-role-kr</pre>
1	5-1-A (명령어 입력)	<pre>CF_DOMAIN=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=skills-global-distribution --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) ACCOUNT_ID=\$(aws sts get-caller-identity --query "Account" --output text) echo "CloudFront Invalidation Test File 1" > test.html aws s3 cp test.html s3://skills-kr-cdn-web-static-\$ACCOUNT_ID/test.html sleep 10 curl -L https://{CF_DOMAIN}/test.html</pre>
	5-1-A (예상 출력) <u>정확히 일치</u>	<pre>upload: ./test.html to s3://skills-kr-cdn-web-static-ACCOUNT_ID/test.html CloudFront Invalidation Test File 1 * 밑줄 친 부분을 제외하고 전부 동일해야 함</pre>

순번	채점항목	
1	5-2-A (명령어 입력)	<pre>CF_DOMAIN=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=skills-global-distribution --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) ACCOUNT_ID=\$(aws sts get-caller-identity --query "Account" --output text) echo "CloudFront Invalidation Test File 2" > test.html aws s3 cp test.html s3://skills-kr-cdn-web-static-\$ACCOUNT_ID/test.html sleep 10 curl -L https://{CF_DOMAIN}/test.html</pre>
	5-2-A (예상 출력) 정확히 일치	<pre>upload: ./test.html to s3://skills-kr-cdn-web-static-ACCOUNT_ID/test.html CloudFront Invalidation Test File 2 * 밑줄 친 부분을 제외하고 전부 동일해야 함</pre>

순번	채점항목	
1	6-1-A (명령어 입력)	<pre> echo =====us-east-1 버킷 직접 접근===== for i in {1..2}; do echo -n "Test \$i: " time aws s3 cp ~/1MiB.file s3://\$us_east_1_bucket/test-direct-\$(date "+%Y%m%d-%H%M%S").file sleep 1 done echo echo =====ap-northeast-2 버킷 직접 접근===== for i in {1..2}; do echo -n "Test \$i: " time aws s3 cp ~/1MiB.file s3://\$ap_northeast_2_bucket/test-direct-\$(date "+%Y%m%d-%H%M%S").file sleep 1 done echo echo =====mrap 접근===== for i in {1..2}; do echo -n "Test \$i: " time aws s3 cp ~/1MiB.file s3://\$mrap_arn/test-mrap-\$(date "+%Y%m%d-%H%M%S").file sleep 1 done echo * 쉘 파일에서 환경변수 참고해서 지정 후 실행해야함 </pre>

순번	채점항목
1	<div> <div>6-1-A (예상 출력) <u>레이턴시 체크</u></div> <div> <p>=====us-east-1 버킷 직접 접근=====</p> <p>upload: ./1MiB.file to s3://skills-us-cdn-web-static-<u>ACCOUNT_ID</u>/test-direct-20250617-011748.file 2.599</p> <p>upload: ./1MiB.file to s3://skills-us-cdn-web-static-<u>ACCOUNT_ID</u>/test-direct-20250617-011751.file 2.739</p> <p>=====ap-northeast-2 버킷 직접 접근=====</p> <p>upload: ./1MiB.file to s3://skills-kr-cdn-web-static-<u>ACCOUNT_ID</u>/test-direct-20250617-011755.file 0.783</p> <p>upload: ./1MiB.file to s3://skills-kr-cdn-web-static-<u>ACCOUNT_ID</u>/test-direct-20250617-011757.file 0.762</p> <p>=====mrap 접근=====</p> <p>upload: ./1MiB.file to s3://arn:aws:s3::<u>ACCOUNT_ID</u>:accesspoint/m19awd3puom5b.mrap/test-mrap-20250617-011758.file 0.798</p> <p>upload: ./1MiB.file to s3://arn:aws:s3::<u>ACCOUNT_ID</u>:accesspoint/m19awd3puom5b.mrap/test-mrap-20250617-011800.file 0.761</p> <p>* MRAP 업로드 시간이 더 짧은 업로드 시간의 리전과 비슷한지 확인</p> </div> </div>

순번	채점항목	
1	7-1-A (명령어 입력)	<pre>CF_DOMAIN=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=skills-global-distribution --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 -L https://{CF_DOMAIN}/index.html</pre>
	7-1-A (예상 출력) <u>정확히 일치</u> <u>시드니 리전</u>	 <p>The screenshot shows a CloudShell terminal window with the region set to 'ap-southeast-2'. The user has entered the commands to fetch the CloudFront distribution ID and domain name. The final command, <code>curl -L https://{CF_DOMAIN}/index.html</code>, has been executed, resulting in the error message: <code>Access denied: unsupported country.</code> This error is highlighted with a red box.</p>
1	7-2-A (명령어 입력)	<pre>CF_DOMAIN=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=skills-global-distribution --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 -L https://{CF_DOMAIN}/index.html</pre>
	7-2-A (예상 출력) <u>정확히 일치</u> <u>서울 리전</u>	 <p>The screenshot shows a CloudShell terminal window with the region set to 'ap-northeast-2'. The user has entered the same commands as in the previous row. The final command, <code>curl -L https://{CF_DOMAIN}/index.html</code>, has been executed successfully, displaying the HTML content of the Korean homepage. The title of the page is '한국 페이지' (Korean Page), which is highlighted with a red box.</p>

순번	채점항목	
1	7-3-A (명령어 입력)	<pre>CF_DOMAIN=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=skills-global-distribution --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 -L https://{CF_DOMAIN}/index.html</pre>
	7-3-A (예상 출력) <u>정확히 일치</u> <u>버니지아북부</u> <u>리전</u>	 <pre>CloudShell us-east-1 + ~ \$ CF_DOMAIN=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name, ~ \$ curl -L https://{CF_DOMAIN}/index.html <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8" /> <meta name="viewport" content="width=device-width, initial-scale=1.0" /> <title>English Page</title> </head> <body> English Page </body> </html>~ \$</pre>
1	7-4-A (명령어 입력)	<pre>CF_DOMAIN=\$(aws resourcegroupstaggingapi get-resources --tag-filters Key=Name,Values=skills-global-distribution --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 -H "user-agent: bot" -L https://{CF_DOMAIN}/index.html</pre>
	7-4-A (예상 출력) <u>정확히 일치</u> <u>버니지아북부</u> <u>리전</u>	 <pre>CloudShell us-east-1 + ~ \$ CF_DOMAIN=\$(aws resourcegroupstaggingapi get-resources --tag-fi ~ \$ curl -H "user-agent: bot" -L https://{CF DOMAIN}/index.html Request blocked due to suspicious User-Agent~ \$</pre>

순번	채점항목	
2	1) skills-bastion 서버에 SSH를 통해 접근합니다. 2) aws configure set default.region ap-southeast-1	
2	1-1 (명령어 입력)	aws ec2 describe-vpcs --filter Name=tag:Name,Values=skills-consumer-vpc --query "Vpcs[].CidrBlock" \ ; aws ec2 describe-vpcs --filter Name=tag:Name,Values=skills-service-vpc --query "Vpcs[].CidrBlock"
	1-1-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	["172.168.0.0/16"] ["10.0.0.0/16"]

2	1-2-A (명령어 입력)	<pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-consumer-public-subnet-a --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-consumer-public-subnet-c --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-consumer-workload-subnet-a --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-consumer-workload-subnet-c --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-service-public-subnet-a --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-service-public-subnet-c --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-service-workload-subnet-a --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-service-workload-subnet-c --query "Subnets[0].CidrBlock"</pre>
	1-2-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	<pre>"172.168.0.0/24" "172.168.1.0/24" "172.168.2.0/24" "172.168.3.0/24" "10.0.0.0/24" "10.0.1.0/24" "10.0.2.0/24" "10.0.3.0/24"</pre>

2	1-3-A (명령어 입력)	<pre> aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-consumer-public-rt --query "RouteTables[].Routes[].GatewayId" \ ; aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-service-public-rt --query "RouteTables[].Routes[].GatewayId" \ ; aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-consumer-workload-rt-a --query "RouteTables[].Routes[].NatGatewayId" \ ; aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-consumer-workload-rt-c --query "RouteTables[].Routes[].NatGatewayId" \ ; aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-service-workload-rt-a --query "RouteTables[].Routes[].NatGatewayId" \ ; aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-service-workload-rt-c --query "RouteTables[].Routes[].NatGatewayId" </pre>
	1-3-A (예상 출력) <u>순서 중요</u>	<pre> "igw-" <u>로 시작하는 문구가 출력이 되는지 확인</u> "VpcLattice" "VpcLattice" "VpcLattice" "igw-" <u>로 시작하는 문구가 출력이 되는지 확인</u> "VpcLattice" "VpcLattice" "nat-" <u>로 시작하는 문구가 출력이 되는지 확인</u> "nat-" <u>로 시작하는 문구가 출력이 되는지 확인</u> "nat-" <u>로 시작하는 문구가 출력이 되는지 확인</u> "nat-" <u>로 시작하는 문구가 출력이 되는지 확인</u> </pre>

2	2-1-A (명령어 입력)	aws vpc-lattice list-service-networks --query "items[?contains(name, 'skills-app-service-network')].name" --output text
	2-1-A (예상 출력) <u>정확히 일치</u>	skills-app-service-network
	2-2-A (명령어 입력)	aws vpc-lattice list-services --query "items[?contains(name, 'skills-app-service')].name" --output text
	2-2-A (예상 출력) <u>정확히 일치</u>	skills-app-service
	2-3-A (명령어 입력)	aws vpc-lattice list-target-groups --query "items[?contains(name, 'skills-alb-tg')].name" --output text
	2-3-A (예상 출력) <u>정확히 일치</u>	skills-alb-tg

2	3-1-A (명령어 입력)	aws ec2 describe-instances --filter Name=tag:Name,Values=skills-bastion --query "Reservations[].Instances[].InstanceType"
	3-1-A (예상 출력) <u>정확히 일치</u>	["t3.micro"]
	3-2-A (명령어 입력)	aws ec2 describe-instances --filter Name=tag:Name,Values=skills-bastion --query "Reservations[].Instances[].PublicIpAddress" aws ec2 describe-addresses --query "Addresses[].PublicIp"
	3-2-A (예상 출력) <u>같은 IP가 2개 있는지 확인</u>	[<u>"3.35.124.14"</u>] ["13.125.100.208", <u>"3.35.124.14"</u> , "3.36.156.140", "3.37.54.103", "43.200.194.248"]

2	3-3-A (명령어 입력)	<pre> INSTANCE_NAME_TAG="skills-bastion" INSTANCE_ID=\$(aws ec2 describe-instances --filters "Name=tag:Name,Values=\$INSTANCE_NAME_TAG" "Name=instance-state-name,Values=running,pending,stopped" --query "Reservations[0].Instances[0].InstanceId" --output text) AMI_ID=\$(aws ec2 describe-instances --instance-ids "\$INSTANCE_ID" --query "Reservations[0].Instances[0].ImageId" --output text) AMI_DESCRIPTION=\$(aws ec2 describe-images --image-ids "\$AMI_ID" --query "Images[0].Description" --output text) INSTANCE_SG_NAME=\$(aws ec2 describe-instances --instance-ids "\$INSTANCE_ID" --query "Reservations[0].Instances[0].SecurityGroups[0].GroupName" --output text) echo "\$AMI_DESCRIPTION" echo "\$INSTANCE_SG_NAME" </pre>
	3-3-A (예상 출력) <u>미출</u> <u>친부분</u> <u>일치</u>	<p><u>Amazon Linux 2023</u> AMI 2023.7.20250428.1 x86_64 HVM kernel-6.1</p> <p><u>skills-bastion-sg</u></p>

2	3-4-A (명령어 입력)	<pre>POLICY_ARNS=\$(aws iam list-attached-role-policies --role-name skills-bastion-role --query "AttachedPolicies[].PolicyArn" --output text) for POLICY_ARN in \$POLICY_ARNS; do POLICY_VERSION=\$(aws iam get-policy --policy-arn \$POLICY_ARN --query "Policy.DefaultVersionId" --output text) POLICY_DOCUMENT=\$(aws iam get-policy-version --policy-arn \$POLICY_ARN --version-id \$POLICY_VERSION --query "PolicyVersion.Document" --output json) echo "\$POLICY_DOCUMENT" done</pre>
	3-4-A (예상 출력) 정확히 일치 이 외 JSON 출력되면 인정 X	<pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": "*", "Resource": "*" }] }</pre>
	4-1-A (명령어 입력)	<pre>aws elbv2 describe-load-balancers --query "LoadBalancers[].{Name:LoadBalancerName,Type:Scheme,Zones:AvailabilityZones[].Z oneName}" --output json</pre>
	4-1-A (예상 출력) 정확히 일치 순서 무관	<pre>"Name : skills-app-alb" <u>문구가 출력이 되는지 확인</u> "Type : internal" <u>문구가 출력이 되는지 확인</u> "Name : skills-consumer-alb" <u>문구가 출력이 되는지 확인</u> "Type : internet-facing" <u>문구가 출력이 되는지 확인</u></pre>

2	5-1-A (명령어 입력)	EXTERNAL_ALB_DNS=\$(aws elbv2 describe-load-balancers --names skills-consumer-alb --query "LoadBalancers[].DNSName" --output text) curl -s http://\$EXTERNAL_ALB_DNS/health
	5-1-A (예상 출력) 정확히 일치	{ "status": "OK" }
	5-2-A (명령어 입력)	EXTERNAL_ALB_DNS=\$(aws elbv2 describe-load-balancers --names skills-consumer-alb --query "LoadBalancers[].DNSName" --output text) curl http://\$EXTERNAL_ALB_DNS/users -X POST -H "Content-Type: application/json" -d '{"UserId": "user123", "Name": "John Doe", "SkillLevel": "Intermediate"}'
	5-2-A (예상 출력) 정확히 일치	{ "UserId": "user123", "Name": "John Doe", "SkillLevel": "Intermediate" }

2	5-3-A (명령어 입력)	EXTERNAL_ALB_DNS=\$(aws elbv2 describe-load-balancers --names skills-consumer-alb --query "LoadBalancers[].DNSName" --output text) curl -s http://\$EXTERNAL_ALB_DNS/users/user123
	5-3-A (예상 출력) <u>정확히 일치</u>	{ "UserId": "user123", "Name": "John Doe", "SkillLevel": "Intermediate" }
	5-4-A (명령어 입력)	EXTERNAL_ALB_DNS=\$(aws elbv2 describe-load-balancers --names skills-consumer-alb --query "LoadBalancers[].DNSName" --output text) curl -s http://\$EXTERNAL_ALB_DNS/users/user123 -X PUT -H "Content-Type: application/json" -d '{"Name": "Jane Doe", "SkillLevel": "Advanced"}'
	5-4-A (예상 출력) <u>정확히 일치</u>	{ "UserId": "user123", "Name": "Jane Doe", "SkillLevel": "Advanced" }

2	5-5-A (명령어 입력)	EXTERNAL_ALB_DNS=\$(aws elbv2 describe-load-balancers --names skills-consumer-alb --query "LoadBalancers[].DNSName" --output text) curl -s http://\$EXTERNAL_ALB_DNS/users/user123 -X DELETE
	5-5-A (예상 출력) 정확히 일치	{ "message": "User user123 deleted successfully" }

2	6-1-A (명령어 입력)	aws dynamodb describe-table --table-name "skills-app-table" --query "Table.TableStatus"
	6-1-A (예상 출력) 정확히 일치	"ACTIVE"
	6-2-A (명령어 입력)	aws dynamodb describe-table --table-name "skills-app-table" --query "Table.BillingModeSummary.BillingMode"
	6-2-A (예상 출력) 정확히 일치	"PAY_PER_REQUEST"
	6-3-A (명령어 입력)	aws dynamodb describe-table --table-name "skills-app-table" --query "Table.DeletionProtectionEnabled"
	6-3-A (예상 출력) 정확히 일치	true

3	1) dev-bastion 서버에 SSH를 통해 접근합니다. 2) aws configure set default.region eu-central-1 3) 채점 명령어에 사용될 환경 변수를 설정합니다. <div style="border: 1px solid black; padding: 2px; margin-top: 5px;">\$ GITHUB_USER=\$(gh api user --jq .login)</div>	
3	1-1 (명령어 입력)	for name in dev-vpc prod-vpc; do id=\$(aws ec2 describe-vpcs --filters "Name=tag:Name,Values=\$name" --query "Vpcs[0].VpcId" --output text); echo "\$id \$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=\$id" --query "length(Subnets)" --output text)"; done
	1-1 (예상 출력)	vpc 로 시작하는 문자열과 4가 총 2번 출력되어야 합니다. vpc -02d5c9fffc8eca645 4 vpc -0c54e3b8561521b8e 4
	2-1 (명령어 입력)	for cluster in dev-cluster prod-cluster; do subnets=\$(aws eks describe-cluster --name \$cluster --query "cluster.resourcesVpcConfig.subnetIds[]" --output text) subnet_count=\${#subnets[@]} vpc_id=\$(aws ec2 describe-subnets --subnet-ids \${subnets[0]} --query "Subnets[0].VpcId" --output text) vpc_name=\$(aws ec2 describe-tags --filters "Name=resource-id,Values=\$vpc_id" "Name=key,Values=Name" --query "Tags[0].Value" --output text) public=false for s in "\${subnets[@]}"; do mp=\$(aws ec2 describe-subnets --subnet-ids \$s --query "Subnets[0].MapPublicIpOnLaunch" --output text) [["\$mp" == "True"]] && public=true && break done pub_status=\$((["\$public" = true] && echo "Public" echo "Private") echo "\$cluster \$subnet_count \$vpc_name \$pub_status" done
	2-1 (예상 출력) 정확히 일치	dev-cluster 2 dev-vpc Private prod-cluster 2 prod-vpc Private
	2-2 (명령어 입력)	kubectl get po -n app --output name grep product kubectl get runner -n app --output name
	2-2 (예상 출력) 순서 무관	강조된 부분이 모두 일치하는지 확인합니다. pod/ product -56d786c497-9crdq pod/ product -56d786c497-gr5vm runner.actions.summerwind.dev /dev-runner-5jwkn-7crdw runner.actions.summerwind.dev /dev-runner-5jwkn-7tlfm <div style="text-align: center;">클라우드컴퓨팅 제2과제 채점기준 43 - 31</div>

3	2-3 (명령어 입력)	argocd app list --output json jq -r '[] [.metadata.name, .spec.sources[0].repoURL, .s pec.source.repoURL] @tsv'
	2-3 (예상 출력)	강조된 부분이 모두 동일한지 확인합니다. dev https://user.github.io/ day2-product/charts prod https://user.github.io/ day2-product/charts
	3-1 (명령어 입력)	aws ecr describe-repositories --query "repositories[?repositoryName=='product/dev'] repositoryName==" --output text
	3-1 (예상 출력) 순서 무관	product/prod product/dev
	3-2 (명령어 입력)	git clone --quiet https://github.com/\${GITHUB_USER}/day2-product marking_product cd marking_product aws iam list-open-id-connect-providers \\ --query "OpenIDConnectProviderList[?contains(Arn, 'token.actions.githubusercontent.com')].Arn" \\ --output text if grep -qE 'aws_access_key_id aws_secret_access_key' .github/workflows/*.yml 2>/dev/null; then echo "warning!" else echo "pass" fi
	3-2 (예상 출력)	숫자를 제외한 모든 부분이 일치해야 합니다. 또한 pass 가 출력되어야 합니다. arn:aws:iam::123456789123:oidc-provider/token.actions.githubusercontent.com pass

3	4-1 (명령어 입력)	gh repo list --limit 5 --json name,visibility --jq '[] "\(.name) (\(.visibility))"'
	4-1 (예상 출력) <u>정확히 일치</u>	day2-product 하나 이외에 다른 Repository가 출력될 경우 오답 처리합니다. 또한 PUBLIC 타입이어야 합니다. day2-product (PUBLIC)
	4-2 (명령어 입력)	gh api repos/"\$GITHUB_USER"/day2-product/contents?ref=main jq -r '[] .name'
	4-2 (예상 출력) <u>순서 무관</u>	.github, Dockerfile, app.py, charts, values 가 존재하는지 확인합니다. .github Dockerfile app.py charts values
	4-3 (명령어 입력)	gh api repos/\$GITHUB_USER/day2-product/contents/charts --jq '[] .name'
	4-3 (예상 출력)	최소 app, index.yaml이 있어야 합니다. app-0.1.0.tgz app index.yaml

3	4-4 (명령어 입력)	gh api repos/"\$GITHUB_USER"/day2-product/git/refs/heads --jq '[] select(.ref=="refs/heads/dev" or .ref=="refs/heads/prod") .ref sub("^refs/heads/",";")'
	4-4 (예상 출력) <u>순서 무관</u>	dev prod
	4-5 (명령어 입력)	gh repo view \$GITHUB_USER/day2-product --json defaultBranchRef --jq '.defaultBranchRef.name'
	4-5 (예상 출력)	dev
	4-6 (명령어 입력)	gh label list --repo \$GITHUB_USER/day2-product grep approval
	4-6 (예상 출력)	강조된 부분이 일치하는지 확인합니다. approval #008fff
	4-7 (명령어 입력)	gh api repos/"\$GITHUB_USER"/day2-product/contents/.github/workflows --jq '[] .name'
	4-7 (예상 출력)	yml 또는 yaml 포맷의 dev, prod가 출력되는지 확인합니다. dev.yml prod.yml

3	4-8 (명령어 입력)	gh api repos/"\$GITHUB_USER"/day2-product/actions/runners --paginate --jq 'runners[] select(any(.labels[].name; . == "dev" or . == "prod")) "\(.name)\t\([.labels[].name] join(", "))"'
	4-8 (예상 출력) <u>정확히 일치</u> <u>순서 무관</u>	강조된 부분이 일치하는지 확인합니다. 2개씩 출력되어야 합니다. dev-runner-5jwkn-7crdw self-hosted,Linux,X64,dev dev-runner-5jwkn-7tlfm self-hosted,Linux,X64,dev prod-runner-qbg9c-9h7zc self-hosted,Linux,X64,prod prod-runner-qbg9c-w7f9k self-hosted,Linux,X64,prod

3	5-1 (명령어 입력)	<pre> export DEV_ALB_ENDPOINT=\$(aws elbv2 describe-load-balancers --names dev-alb --query "LoadBalancers[0].DNSName" --output text) curl -s "http://\${DEV_ALB_ENDPOINT}/api"; echo echo ===== cd marking_product git checkout dev > /dev/null 2> /dev/null git pull origin dev > /dev/null 2> /dev/null git checkout -b feature/marking-v2 > /dev/null 2> /dev/null sed -i 's/product.*/product marking!\n/g' app.py git add app.py > /dev/null 2> /dev/null git commit -m "feat: product marking" > /dev/null 2> /dev/null git push origin feature/marking-v2 > /dev/null 2> /dev/null gh pr create --base dev --head feature/marking-v2 --title "Feat: product marking" --body "made with♥️" echo ===== echo wait 1 minutes sleep 1m export DEV_ALB_ENDPOINT=\$(aws elbv2 describe-load-balancers --names dev-alb --query "LoadBalancers[0].DNSName" --output text) curl -s "http://\${DEV_ALB_ENDPOINT}/api" </pre>
	5-1 (예상 출력) 최대 1분 대기	<pre> product v1 ===== ... ===== wait 1 minutes product marking! </pre>
	5-2 (명령어 입력)	<pre> gh run list -R "\$GITHUB_USER/day2-product" -w dev.yml --status completed --json startedAt,updatedAt --limit 1 -q '[0] ((.updatedAt fromdateiso8601)-(.startedAt fromdateiso8601))' </pre>
	5-2 (예상 출력)	<p>60 이하의 수가 출력되면 정답 처리합니다.</p> <p>47</p>

<p>5-3 (명령어 입력)</p>	<pre>aws ecr describe-images --repository-name product/dev --query 'sort_by(imageDetails,&imagePushedAt)[-1].[imageTags[0], imageManifestMediaType]' --output text while read -r TAG MEDIA; do if ["\$MEDIA" = "application/vnd.oci.image.index.v1+json"]; then TYPE="ImageIndex"; else TYPE="Image"; fi; printf "%s\t%s\n" "\$TAG" "\$TYPE"; done</pre>
<p>5-3 (예상 출력)</p>	<p>latest가 아닌 문자열과 ImageIndex가 출력되어야 합니다.</p> <pre>cc796ad ImageIndex</pre>
<p>5-4 (명령어 입력)</p>	<pre>export PROD_ALB_ENDPOINT=\$(aws elbv2 describe-load-balancers --names prod-alb --query "LoadBalancers[0].DNSName" --output text) curl -s "http://\${PROD_ALB_ENDPOINT}/api"; echo echo ===== cd marking_product git checkout dev > /dev/null 2> /dev/null gh pr create --base prod --head dev --title "Feat: product marking" --body "made with ♥" echo \# Proceed on https://github.com/\${GITHUB_USER}/day2-product/pulls while ! gh run list -R \${GITHUB_USER}/day2-product --status in_progress grep -q "in_progress"; do sleep 1s; done; echo ===== echo wait 1 minutes sleep 1m export PROD_ALB_ENDPOINT=\$(aws elbv2 describe-load-balancers --names prod-alb --query "LoadBalancers[0].DNSName" --output text) curl -s "http://\${PROD_ALB_ENDPOINT}/api" cd</pre>
<p>5-4 (예상 출력) <u>최대 5분 대기</u></p>	<p>product v1이 출력되는지 확인합니다.</p> <pre>product v1 =====.</pre> <p>...</p> <p>명령어 실행 후 출력되는 Proceed on https://github.com/.../day2-product/pulls 링크로 접근합니다. 브라우저에서 Github에 로그인 되어있어야 합니다.</p>  <p>생성된 PR을 클릭합니다.</p>

<div>3</div>	<div>5-4</div> <div>(예상 출력)</div> <div>최대 1분 대기</div>	<div>  <p>자동으로 Merge되지 않음을 확인합니다. (이미 Actions 작업이 수행중이라면 오답 처리)</p>  <p>Labels 우측의 톱니바퀴를 클릭하여 approval label을 추가합니다. (approval label이 존재하지 않을 시 오답 처리)</p>  <p>label 추가 이후 Actions가 자동으로 동작하는지 확인합니다.</p> <p>확인 후 다시 Bastion에서 이어서 채점을 진행합니다. Github 웹페이지에서 Actions가 실행되면 스크립트에서 자동으로 1분 대기가 진행됩니다. 1분 이내에 product marking!이 출력되는지 확인합니다.</p> <p>...</p> <pre>===== wait 1 minutes product marking!</pre> </div>
--------------	---	--

3	5-5 (명령어 입력)	gh run list -R "\$GITHUB_USER/day2-product" -w prod.yml --status completed --json startedAt,updatedAt --limit 1 \\\n-q '[0] ((.updatedAt fromdateiso8601)-(.startedAt fromdateiso8601))'
	5-5 (예상 출력)	60 이하의 수가 출력되면 정답 처리합니다. 48
	5-6 (명령어 입력)	aws ecr describe-images --repository-name product/prod --query 'sort_by(imageDetails,&imagePushedAt)[-1].[imageTags[0], imageManifestMediaType]' --output text while read -r TAG MEDIA; do if ["\$MEDIA" = "application/vnd.oci.image.index.v1+json"]; then TYPE="ImageIndex"; else TYPE="Image"; fi; printf "%s\\t%s\\n" "\$TAG" "\$TYPE"; done echo
	5-6 (예상 출력)	latest가 아닌 문자열과 ImageIndex가 출력되어야 합니다. cc796ad ImageIndex
4	1) skills-log-bastion 서버에 SSH를 통해 접근합니다. 2) aws configure set default.region eu-west-1	

4	1-1 (명령어 입력)	aws ec2 describe-vpcs --filter Name=tag:Name,Values=skills-log-vpc --query "Vpcs[].CidrBlock"
	1-1 (예상 출력)	["10.1.0.0/16"]
	1-2 (명령어 입력)	aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-log-priv-a --query "Subnets[].CidrBlock" aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-log-priv-b --query "Subnets[].CidrBlock" aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-log-pub-a --query "Subnets[].CidrBlock" aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-log-pub-b --query "Subnets[].CidrBlock"
	1-2 (예상 출력)	["10.1.0.0/24"] ["10.1.1.0/24"] ["10.1.2.0/24"] ["10.1.3.0/24"]
	1-3 (명령어 입력)	aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-log-priv-rt-a --query "RouteTables[].Routes[].NatGatewayId" aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-log-priv-rt-b --query "RouteTables[].Routes[].NatGatewayId" aws ec2 describe-route-tables --filter Name=tag:Name,Values=skills-log-pub-rt --query "RouteTables[].Routes[].GatewayId"
	1-3 (예상 출력)	강조한 부분이 일치하는지 확인합니다. ["nat-0de5f8e7b66a46c1d"] ["nat-0de5f8e7b66a46c1d"] ["local", "igw-034a42626600d7d02"]

4	2-1 (명령어 입력)	aws ec2 describe-instances --filter Name=tag:Name,Values=skills-log-bastion --query "Reservations[].Instances[].InstanceType"
	2-1 (예상 출력)	["t3.small"]
	3-1 (명령어 입력)	aws elbv2 describe-load-balancers --query "LoadBalancers[?contains(LoadBalancerName, 'skills-log-alb')].{DNSName:DNSName, Scheme:Scheme}"
	3-1 (예상 출력)	강조한 부분이 일치하는지 확인합니다. [{ "DNSName": "skills-log-alb-693898456.eu-west-1.elb.amazonaws.com", "Scheme": "internet-facing" }]
	4-1 (명령어 입력)	ALB_DNS=\$(aws elbv2 describe-load-balancers --names skills-log-alb --query "LoadBalancers[].DNSName" --output text) curl -s http://\$ALB_DNS/check
	4-1 (예상 출력)	{"data":"hello"}
	5-1 (명령어 입력)	aws ecr describe-repositories --query "repositories[?repositoryName=='skills-app'].repositoryName" aws ecr describe-repositories --query "repositories[?repositoryName=='skills-firelens'].repositoryName"
	5-1 (예상 출력)	["skills-app"] ["skills-firelens"]
	6-1 (명령어 입력)	aws ecs describe-clusters --clusters skills-log-cluster --query "clusters[0].status"
	6-1 (예상 출력)	"ACTIVE"

4	6-2 (명령어 입력)	aws ecs describe-services --cluster skills-log-cluster --services app --query "services[].status"
	6-2 (예상 출력)	["ACTIVE"]
	6-3 (명령어 입력)	aws ecs describe-task-definition --task-definition skills-log-app-td --query "taskDefinition.containerDefinitions[].name" aws ecs describe-task-definition --task-definition skills-log-app-td --query "taskDefinition.containerDefinitions[].image"
	6-3 (예상 출력) <u>순서 중요</u>	강조한 부분이 일치하는지 확인합니다. ["app", "log_router"] ["658986583341.dkr.ecr.eu-west-1.amazonaws.com/skills-app:latest", "658986583341.dkr.ecr.eu-west-1.amazonaws.com/skills-firelens:latest"]
	6-4 (명령어 입력)	aws ecs describe-services --cluster skills-log-cluster --services app --query "services[].networkConfiguration.awsVpcConfiguration[].subnets[]"
	6-4 (예상 출력)	강조한 부분이 일치하는지 확인합니다. ["subnet-08542bfe191dce1bd", "subnet-0c818d2e8d873f981"]
	7-1 (명령어 입력)	aws ecs describe-task-definition --task-definition skills-log-app-td --query "taskDefinition.containerDefinitions[].logConfiguration.logDriver"
	7-1 (예상 출력)	awsfirelens가 포함되어 있는지 확인합니다. ["awsfirelens", "awslogs"]
	8-1 (명령어 입력)	aws logs describe-log-groups --log-group-name-prefix /skills/app --query "logGroups[].logGroupName"
	8-1 (예상 출력)	["/skills/app"]

4	8-2 (명령어 입력)	aws logs describe-log-streams --region eu-west-1 --log-group-name "/skills/app" --log-stream-name-prefix logs/\$(aws ecs list-tasks --cluster skills-log-cluster --service-name app --desired-status RUNNING --region eu-west-1 --output text --query taskArns[0] awk -F/ '{print \$NF}') jq -r '.logStreams[].logStreamName'
	8-2 (예상 출력)	logs/ 로 시작하고 뒤에 ECS Task ID 문자열이 출력되는지 확인합니다. logs/869b921303a54e06b34a46620bb96a6c
	8-3 (명령어 입력) 2분 대기	curl http://\$(aws elbv2 describe-load-balancers --names skills-log-alb --region eu-west-1 --query 'LoadBalancers[0].DNSName' --output text)/check echo waiting for 2 minutes sleep 120s aws logs get-log-events --region eu-west-1 --log-group-name "/skills/app" --log-stream-name \$(aws logs describe-log-streams --region eu-west-1 --log-group-name "/skills/app" --log-stream-name-prefix logs/\$(aws ecs list-tasks --cluster skills-log-cluster --service-name app --desired-status RUNNING --region eu-west-1 --query 'taskArns[0]' --output text awk -F/ '{print \$NF}') --query 'logStreams[0].logStreamName' --output text) --limit 100 --no-start-from-head --query 'reverse(sort_by(events[?contains(message, `"/check`)], ×tamp)) [0].timestamp' --output text grep -E '^'[0-9]+' xargs -l{} bash -c 'date -u -d @\$((expr {} / 1000)) '+%Y-%m-%d %H:%M:%S'' date -u '+%Y-%m-%d %H:%M:%S'
	8-3 (예상 출력)	강조한 부분이 일치하는지 확인합니다. {"data":"hello"} waiting for 2 minutes 2025-06-25 00:33:06 2025-06-25 00:35:09