2024 경상북도 제59회 전국기능경기대회 채점기준

1. 채점상의 유의사항	직 종 명	클라우드컴퓨팅

※ 다음 사항을 유의하여 채점하시오.

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

2. 채점기준표

1) 주요항목별 배점

직 종 명	클라우드컴퓨팅

과제 번호	일련 번호	주요항목	배점	채점방법		채점시기		비고
				독립	합의	경기 진행중	경기 종료후	
제1과제	1	네트워크 구성	3		○		○	
	2	Bastion 서버	2		○		○	
	3	관계형 데이터베이스	1.5		○		○	
	4	NoSQL 데이터베이스	1.5		○		○	
	5	웹 어플리케이션	3		○		○	
	6	Secret Store	1		○		○	
	7	ECR	1		○		○	
	8	EKS	4		○		○	
	9	로드 밸런서	1		○		○	
	10	S3	2		○		○	
	11	CloudFront	2		○		○	
	12	Logging	3		○		○	
	13	보안 구성	5		○		○	
합 계			30					

2) 채점방법 및 기준

과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
1과제	1	네트워크 구성	1	VPC, Subnet	1
			2	Routing	1
			3	VPC Endpoint	1
	2	Bastion 서버	1	Bastion configuration	1
			2	Bastion SG ingress rule	1
	3	관계형 데이터베이스	1	RDS configuration	1.5
	4	NoSQL 데이터베이스	1	DynamoDB configuration	1.5
	5	웹 어플리케이션	1	customer service	1
			2	Product service	1
			3	order service	1
	6	Secret Store	2	Secret Manager 구성 확인	1
	7	ECR	1	ECR Scan 취약점 확인	1
	8	EKS	1	Node Group 구성 확인	2
			2	pod 구성 확인	2
	9	로드 밸런서	1	ALB allow from only Cloudfront	1
	10	S3	1	S3 Bucket configuration	1
			2	S3 Bucket encryption	1
	11	CloudFront	1	Cloudfront to ALB	1
			2	Cloudfront to S3	1
	12	Logging	1	로그 그룹 확인	1
			2	로그 출력 확인	1
			3	로그 제외 확인	1
	13	보안 구성	1	product - customer 간 통신 제한	2
			2	ALB Security Group 구성	1
			3	ALB Header Block	1
			4	CloudFront User-Agent Block	1
	총점				30

3) 채점내용

순번	사전준비
0	1) Bastion 서버에 SSH를 통해 접근합니다. (별도 명시가 없는 경우 모든 채점은 Bastion 서버에서 진행합니다.) 2) Bastion 명령어 및 권한을 확인합니다. (awscli permission, jq, curl, awscli region) 3) marking 스크립트들을 /root/marking에 다운로드 합니다. 4) /root/marking 경로에서 스크립트를 실행합니다. 실행 결과를 기반으로 채점을 진행하되 선수가 이의를 제기할 경우 수동으로 채점을 진행할 수 있도록 합니다. 5) 채점을 진행하는 Bastion 서버의 쉘을 초기 실행할 때 다음 명령어을 실행하여 환경 변수를 초기화합니다. (**채점 스크립트로 진행 시 생략**) # set default region of aws cli aws configure set default.region ap-northeast-2

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

순번	채점 항목	
1-3	1-3-A (명령어 입력)	aws ec2 describe-vpc-endpoints --query "VpcEndpoints[].ServiceName"
	1-3-A (예상 출력) **최소_내용 포함** **순서 무관**	["com.amazonaws.ap-northeast-2.dynamodb", "com.amazonaws.ap-northeast-2.ecr.dkr", "com.amazonaws.ap-northeast-2.ecr.api"]
2-1	2-1-A (명령어 입력)	aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-bastion ₩ --query "Reservations[0].Instances[0].InstanceType"
	2-1-A (예상 출력) **정확히 일치**	"t3.small"
2-2	2-2-A (명령어 입력)	aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-bastion --query "Reservations[0].Instances[0].SecurityGroups[0].GroupName" ₩ ; aws ec2 describe-security-groups --filter Name=group-name,Values=wsi-bastion-sg --query "SecurityGroups[0].IpPermissions[].{FromPort:FromPort,ToPort:ToPort,IpRanges:IpRanges}"

	2-2-A
	(예상 출력)
	정확히 일치
	Description무시
		"wsi-bastion-sg"
		[
		{
		"ToPort": 4272,
		"FromPort": 4272,
		"IpRanges": [
		{
		"CidrIp": "0.0.0.0/0" → 단일 IP만 허용할 수도 있음
		}
		]
		}
		]

순번	채점 항목	
3-1	3-1-A (명령어 입력)	aws rds describe-db-clusters --db-cluster-identifier wsi-aurora-mysql ₩ --query 'DBClusters[*].{DBClusterIdentifier: DBClusterIdentifier, ₩ EngineVersion: EngineVersion, Encryption: StorageEncrypted, ₩ LogExports: EnabledCloudwatchLogsExports}' --output json

	3-1-A
	(예상 출력)
	순서 무관
		{
		"DBClusterIdentifier": "wsi-aurora-mysql",
		"EngineVersion": "8.0.mysql_aurora.3.05.2",
		"Encryption": true,
		"LogExports": [
		"audit",
		"error"
		]
		}

	3-1-B (명령어 입력)	aws rds describe-db-clusters --db-cluster-identifier wsi-aurora-mysql ₩ --query 'DBClusters[*].KmsKeyId' --output text
	3-1-B (예상 출력)	"arn:aws:kms:" 로 시작하는 문자열

순번	채점 항목	
4-1	4-1-A (명령어 입력)	aws dynamodb describe-table --table-name order --query ₩ 'Table.{TableName: TableName, EncryptionAtRest: SSEDescription.SSEType}" ₩ --output json

	4-1-A
	(예상 출력)
	정확히 일치
	순서 중요
		{
		"TableName": "order",
		"EncryptionAtRest": "KMS"
		}

	4-1-B (명령어 입력)	aws dynamodb describe-table --table-name order ₩ --query 'Table.SSEDescription.KMSMasterKeyArn' --output text
	4-1-B (명령어 입력)	"arn:aws:kms:" 로 시작하는 문자열
5-1	5-1-A (명령어 입력)	export cdn=$(aws resourcegroupstaggingapi get-resources ₩ --tag-filters Key=Name,Values=wsi-cdn ₩ --resource-type-filters 'cloudfront' --region us-east-1 ₩ --query "ResourceTagMappingList[0].ResourceARN" ₩ --output text \| sed 's:.*/::' \| xargs -I {} aws cloudfront get-distribution ₩ --id {} ₩ --query "Distribution.DomainName" --output text) curl -X POST "https://$cdn/v1/customer" ₩ -H "Content-Type: application/json" ₩ -H "user-agent: safe-client" ₩ -d '{ "id": "123", "name": "wsiman", "gender": "man" }'
	5-1-A (예상 출력)	{"customer":{"id":"123","name":"wsiman","gender":"man"},"message":"The customer is created."}

순번	채점 항목	
5-1	5-1-B (명령어 입력)	export cdn=$(aws resourcegroupstaggingapi get-resources ₩ --tag-filters Key=Name,Values=wsi-cdn ₩ --resource-type-filters 'cloudfront' --region us-east-1 ₩ --query "ResourceTagMappingList[0].ResourceARN" ₩ --output text \| sed 's:.*/::' \| xargs -I {} aws cloudfront get-distribution ₩ --id {} ₩ --query "Distribution.DomainName" --output text) curl -X GET "https://$cdn/v1/customer?id=123" ₩ -H "Content-Type: application/json" ₩ -H "user-agent: safe-client"
	5-1-B (예상 출력)	{"customer":{"id":"123","name":"wsiman","gender":"man"},"message":"The customer is well in database."}
5-2	5-2-A (명령어 입력)	export cdn=$(aws resourcegroupstaggingapi get-resources ₩ --tag-filters Key=Name,Values=wsi-cdn ₩ --resource-type-filters 'cloudfront' --region us-east-1 ₩ --query "ResourceTagMappingList[0].ResourceARN" ₩ --output text \| sed 's:.*/::' \| xargs -I {} aws cloudfront get-distribution ₩ --id {} ₩ --query "Distribution.DomainName" --output text) curl -X POST "https://$cdn/v1/product" ₩ -H "Content-Type: application/json" ₩ -H "user-agent: safe-client" ₩ -d '{ "id": "123", "name": "wsiman", "category": "book" }'
	5-2-A (예상 출력)	{"product":{"id":"123","name":"wsiman","category":"book"},"message":"The product is created."}

순번	채점 항목	
5-2	5-2-B (명령어 입력)	export cdn=$(aws resourcegroupstaggingapi get-resources ₩ --tag-filters Key=Name,Values=wsi-cdn ₩ --resource-type-filters 'cloudfront' --region us-east-1 ₩ --query "ResourceTagMappingList[0].ResourceARN" ₩ --output text \| sed 's:.*/::' \| xargs -I {} aws cloudfront get-distribution ₩ --id {} ₩ --query "Distribution.DomainName" --output text) curl -X GET "https://$cdn/v1/product?id=123" ₩ -H "Content-Type: application/json" ₩ -H "user-agent: safe-client"
	5-2-B (예상 출력)	{"product":{"id":"123","name":"wsiman","category":"book"},"message":"The product is well in database."}
5-3	5-3-A (명령어 입력)	export cdn=$(aws resourcegroupstaggingapi get-resources ₩ --tag-filters Key=Name,Values=wsi-cdn ₩ --resource-type-filters 'cloudfront' --region us-east-1 ₩ --query "ResourceTagMappingList[0].ResourceARN" ₩ --output text \| sed 's:.*/::' \| xargs -I {} aws cloudfront get-distribution ₩ --id {} ₩ --query "Distribution.DomainName" --output text) curl -X POST "https://$cdn/v1/order" ₩ -H "Content-Type: application/json" ₩ -H "user-agent: safe-client" ₩ -d '{ "id": "123", "customerid": "123", "productid": "123" }'
	5-3-A (예상 출력)	{"order":{"id":"123","customerid":"123","productid":"123"},"message":"The order is created."}

순번	채점 항목	
5-3	5-3-A (명령어 입력)	export cdn=$(aws resourcegroupstaggingapi get-resources ₩ --tag-filters Key=Name,Values=wsi-cdn ₩ --resource-type-filters 'cloudfront' --region us-east-1 ₩ --query "ResourceTagMappingList[0].ResourceARN" ₩ --output text \| sed 's:.*/::' \| xargs -I {} aws cloudfront get-distribution ₩ --id {} ₩ --query "Distribution.DomainName" --output text) curl -X GET "https://$cdn/v1/order?id=123" ₩ -H "Content-Type: application/json" ₩ -H "user-agent: safe-client"
	5-3-A (예상 출력)	{"order":{"id":"123","customerid":"123","productid":"123"},"message":"The order is well in database."}
6-1	6-1-A (명령어 입력)	aws secretsmanager describe-secret --secret-id customer --query 'Name' ₩ --output text ₩ ; aws secretsmanager describe-secret --secret-id product --query 'Name' ₩ --output text ₩ ; aws secretsmanager describe-secret --secret-id order --query 'Name' ₩ --output text
	6-1-A (예상 출력)	customer product order
7-1	7-1-A (명령어 입력)	aws ecr describe-images --repository-name customer-ecr ₩ --query "imageDetails[].imageTags[]" ₩ ; aws ecr describe-images --repository-name product-ecr ₩ --query "imageDetails[].imageTags[]" ₩ ; aws ecr describe-images --repository-name order-ecr ₩ --query "imageDetails[].imageTags[]"
	7-1-A (예상 출력)	"latest" "latest" "latest"

순번	채점 항목	
7-1	7-1-B (명령어 입력)	aws ecr describe-image-scan-findings --repository-name customer-ecr ₩ --image-id imageTag=latest --query "imageScanFindings.findingSeverityCounts" aws ecr describe-image-scan-findings --repository-name product-ecr ₩ --image-id imageTag=latest --query "imageScanFindings.findingSeverityCounts" aws ecr describe-image-scan-findings --repository-name order-ecr ₩ --image-id imageTag=latest --query "imageScanFindings.findingSeverityCounts"
	7-1-B (예상 출력)	출력이 없어야 함
8-1	8-1-A (명령어 입력)	aws eks describe-nodegroup --cluster-name wsi-eks-cluster ₩ --nodegroup-name wsi-app-nodegroup ₩ --query "nodegroup.nodegroupName" ₩ ; aws eks describe-nodegroup --cluster-name wsi-eks-cluster ₩ --nodegroup-name wsi-addon-nodegroup --query "nodegroup.nodegroupName"
	8-1-A (예상 출력)	"wsi-app-nodegroup" "wsi-addon-nodegroup"
	8-1-B (명령어 입력)	kubectl get nodes --no-headers ₩ -l eks.amazonaws.com/nodegroup=wsi-app-nodegroup | wc -l ; kubectl get nodes --no-headers ₩ –l eks.amazonaws.com/nodegroup=wsi-addon-nodegroup | wc -l
	8-1-B (예상 출력)	4 4

순번	채점 항목	
8-1	8-1-C (명령어 입력)	kubectl get no -l "eks.amazonaws.com/nodegroup=wsi-app-nodegroup" \ --output json | \ jq ".items[].metadata.labels.\"node.kubernetes.io/instance-type\"" ; kubectl get no –l "eks.amazonaws.com/nodegroup=wsi-addon-nodegroup" --output json | \ jq ".items[].metadata.labels.\"node.kubernetes.io/instance-type\""
	8-1-C (예상 출력)	"t3.large" "t3.large" "t3.large" "t3.large" "t3.medium" "t3.medium" "t3.medium" "t3.medium"
8-2	8-2-A (명령어 입력)	kubectl get pods -n wsi --no-headers -l app=customer | wc -l kubectl get pods -n wsi --no-headers -l app=product | wc -l kubectl get pods -n wsi --no-headers -l app=order | wc -l
	8-2-A (예상 출력)	2 2 2
	8-2-B (명령어 입력)	kubectl get pods -n wsi -l app=order -o jsonpath="{.items[0].spec.nodeName}"
	8-2-B (예상 출력)	"fargate-ip" 로 시작하는 문자열

순번	채점 항목	
9-1	9-1-A (명령어 입력)	albDNS=$(aws elbv2 describe-load-balancers --name wsi-alb ₩ --output text --query "LoadBalancers[].DNSName") curl -s -o /dev/null -w "%{http_code}" -X GET ₩ http://${albDNS}/v1/customer?id=123 -H "X-wsi-header: Skills2024" --max-time 5
	9-1-A (예상 출력)	000
10-1	10-1-A (명령어 입력)	aws s3 ls | grep -E "apne2-wsi-static"
	10-1-A (예상 출력) **밑줄 부분 일치** **날짜, 시간 무관**	2024-05-28 01:56:07 **apne2-wsi-static-<비번호 3자리>**
10-2	10-2-A (명령어 입력)	aws s3api get-bucket-encryption --bucket $(aws s3 ls | grep apne2-wsi-static ₩ | awk '{print $3}') --query ₩ 'ServerSideEncryptionConfiguration.Rules[0].ApplyServerSideEncryptionByDefault.SSEAlgorithm'
	10-2-A (예상 출력)	"aws:kms"

순번	채점 항목	
11-1	11-1-A (명령어 입력)	export cdn=$(aws resourcegroupstaggingapi get-resources ₩ --tag-filters Key=Name,Values=wsi-cdn ₩ --resource-type-filters 'cloudfront' --region us-east-1 ₩ --query "ResourceTagMappingList[0].ResourceARN" ₩ --output text \| sed 's:.*/::' \| xargs -I {} aws cloudfront get-distribution --id {} ₩ --query "Distribution.DomainName" --output text) curl -X GET "https://$cdn/v1/customer?id=123" ₩ -H "Content-Type: application/json" ₩ -H "user-agent: safe-client"
	11-1-A (예상 출력)	{"customer":{"id":"123","name":"wsiman","gender":"man"},"message":"The customer is well in database."}
11-2	11-2-A (명령어 입력)	export cdn=$(aws resourcegroupstaggingapi get-resources ₩ --tag-filters Key=Name,Values=wsi-cdn ₩ --resource-type-filters 'cloudfront' --region us-east-1 ₩ --query "ResourceTagMappingList[0].ResourceARN" ₩ --output text \| sed 's:.*/::' \| xargs –I {} aws cloudfront get-distribution --id {} ₩ --query "Distribution.DomainName" --output text) curl -X GET "https://$cdn/static/index.html" ₩ -H "Content-Type: application/json" ₩ -H "user-agent: safe-client"
	11-2-A (예상 출력)	2024 wsi static web page

순번	채점 항목	
12-1	12-1-A (명령어 입력)	aws logs describe-log-groups ₩ --log-group-name-prefix /wsi/webapp/customer ₩ --query 'logGroups[*].kmsKeyId' --output text
	12-1-A (예상 출력)	"arn:aws:kms" 로 시작하는 문자열
	12-1-B (명령어 입력)	aws logs describe-log-groups ₩ --log-group-name-prefix /wsi/webapp/product ₩ --query 'logGroups[*].kmsKeyId' --output text
	12-1-B (예상 출력)	"arn:aws:kms" 로 시작하는 문자열
	12-1-C (명령어 입력)	aws logs describe-log-groups ₩ --log-group-name-prefix /wsi/webapp/order ₩ --query 'logGroups[*].kmsKeyId' --output text
	12-1-C (예상 출력)	"arn:aws:kms" 로 시작하는 문자열

순번	채점 항목	
12-2	12-2-A (명령어 입력)	export cdn=$(aws resourcegroupstaggingapi get-resources ₩ --tag-filters Key=Name,Values=wsi-cdn ₩ --resource-type-filters 'cloudfront' --region us-east-1 ₩ --query "ResourceTagMappingList[0].ResourceARN" ₩ --output text | sed 's:.*/::' | xargs -I {} aws cloudfront get-distribution --id {} ₩ --query "Distribution.DomainName" --output text)
	12-2-B (명령어 입력)	curl --silent --output /dev/null -X GET "https://${cdn}/v1/customer?id=skills2024" ₩ -H "Content-Type: application/json" -H "user-agent: safe-client" curl --silent --output /dev/null -X GET "https://${cdn}/v1/product?id=skills2024" ₩ -H "Content-Type: application/json" -H "user-agent: safe-client" curl --silent --output /dev/null -X GET "https://${cdn}/v1/order?id=skills2024" ₩ -H "Content-Type: application/json" -H "user-agent: safe-client" sleep 1m aws logs filter-log-events --log-group-name /wsi/webapp/customer ₩ --filter-pattern '"/v1/customer?id=skills2024"' | jq ".events | length" aws logs filter-log-events --log-group-name /wsi/webapp/product ₩ --filter-pattern '"/v1/product?id=skills2024"' | jq ".events | length" aws logs filter-log-events --log-group-name /wsi/webapp/order ₩ --filter-pattern '"/v1/order?id=skills2024"' | jq ".events | length"
	12-2-B (예상 출력)	1 1 1
12-3	12-3-A (명령어 입력)	aws logs filter-log-events --log-group-name /wsi/webapp/customer ₩ --filter-pattern '"/healthcheck"' | jq ".events | length" aws logs filter-log-events --log-group-name /wsi/webapp/product ₩ --filter-pattern '"/healthcheck"' | jq ".events | length" aws logs filter-log-events --log-group-name /wsi/webapp/order ₩ --filter-pattern '"/healthcheck"' | jq ".events | length"
	12-3-A (예상 출력)	0 0 0

순번	채점 항목	
13-1	13-1-A (명령어 입력)	pod=$(kubectl get pods -n wsi --no-headers ₩ -o custom-columns=":metadata.name" | grep product| head -n 1) kubectl exec -it $pod -n wsi -- curl -X GET --max-time 5 ₩ -w "₩n%{http_code}₩n" customer-service.wsi.svc.cluster.local/v1/customer
	13-1-A (예상 출력) **밑줄 달라도 됨**	curl: (28) Connection timed out after **5002** milliseconds 000
	13-1-B (명령어 입력)	pod=$(kubectl get pods -n wsi --no-headers ₩ -o custom-columns=":metadata.name" | grep customer | head -n 1) kubectl exec -it $pod -n wsi -- curl -X GET --max-time 5 ₩ -w "₩n%{http_code}₩n" product-service.wsi.svc.cluster.local/v1/product
	13-1-B (예상 출력) **밑줄 달라도 됨**	curl: (28) Connection timed out after **5002** milliseconds 000
13-2	13-2-A (명령어 입력)	albDNS=$(aws elbv2 describe-load-balancers --name wsi-app-alb ₩ --output text --query "LoadBalancers[].DNSName") curl -X GET -H 'X-wsi-header: Skills2024' --max-time 5 ₩ -w "₩n%{http_code}₩n" http://${albDNS}/v1/product
	13-3-A (예상 출력) **밑줄 달라도 됨**	curl: (28) Connection timed out after **5002** milliseconds 000

순번	채점 항목	
13-3	13-3-A (명령어 입력)	albDNS=$(aws elbv2 describe-load-balancers --name wsi-app-alb ₩ --output text --query "LoadBalancers[].DNSName") albID=$(aws ec2 describe-security-groups ₩ --filter Name=group-name,Values=wsi-app-alb-sg ₩ --query "SecurityGroups[0].GroupId" | sed s/₩"//g) aws ec2 authorize-security-group-ingress --group-id $albID --protocol tcp ₩ --port 80 --cidr 0.0.0.0/0 aws ec2 authorize-security-group-egress --group-id $albID --protocol tcp ₩ --port 80 --cidr 0.0.0.0/0
	13-3-B (명령어 입력)	curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${albDNS}/v1/product
	13-3-B (예상 출력) **정확히 일치**	Access Denied 403
	13-3-C (명령어 입력)	curl -X GET -H 'X-wsi-header: Skills2024' --max-time 5 -w "₩n%{http_code}₩n" http://${albDNS}/v1/product
	13-3-C (예상 출력)	"Access Denied", "403" 이 아닌 문자열
13-4	13-4-A (명령어 입력)	export cdn=$(aws resourcegroupstaggingapi get-resources ₩ --tag-filters Key=Name,Values=wsi-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)
	13-4-B (명령어 입력)	curl -X GET -A safe-client --max-time 5 -w "₩n%{http_code}₩n" ₩ https://${cdn}/static/index.html
	13-4-B (예상 출력) **정확히 일치**	2024 wsi static web page 200
	13-4-C (명령어 입력)	curl -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${cdn}/static/index.html
	13-4-C (예상 출력) **정확히 일치**	Access Denied 403