

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 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) 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. 14) 모든 채점 사항은 skills-bastion으로 ssh 접속 후 진행합니다. 		

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	VPC			○		○	
	2	Transit Between VPC			○		○	
	3	Network Firewall			○		○	
	4	Bastion Server			○		○	
	5	Secret Store			○		○	
	6	RDBMS			○		○	
	7	S3			○		○	
	8	Container Registry			○		○	
	9	Container Orchestration			○		○	
	10	Load Balancer			○		○	
	11	Logging			○		○	
	12	Monitoring			○		○	
	13	Continuous Delivery			○		○	
	14							
합 계			30					

2) 채점방법 및 기준

과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
1과제	1	VPC	1	VPC	0.5
			2	Subnet	0.5
			3	VPC Endpoint	0.5
	2	Peering between VPC	1	Peering Connection Configure	0.5
	3	Network Firewall	1	Network Firewall Configure	1.0
			2	Network Firewall Drop Test	1.0
			3	Security Group Outbound Rule	1.0
			4	Network Firewall Pass Test	1.0
	4	Bastion Server	1	Instance Type	0.5
			2	Public IP	0.5
			3	Instance Configure	0.5
			4	Attached Policies	0.5
	5	Secret Store	1	Secret	0.5
			2	ENV ValueFrom, envFrom	1
	6	RDBMS	1	RDS Configure	0.5
			2	Backtrack Window	0.5
	7	S3	1	Bucket Configure	0.5
			2	Helm Chart	0.5
			3	S3 RepoURL	1.5
	8	Container Registry	1	ECR Configure	0.5
			2	Image Tag	0.5

	9	EKS	1	Cluster Configure	0.5
			2	Cluster Encryption	1.0
			3	App Nodegroup Configure	0.5
			4	Addon Nodegroup Configure	0.5
			5	Fargate Profile Configure	1.0
			6	Deployment	0.5
	10	Load Balancer	1	ELB Configure	1.0
			2	Green API Test	1.5
			3	Red API Test	1.5
	11	Logging	1	Domain Check	0.5
			2	Domain Configure	1.0
			3	Index Check	1.0
			4	Log Format Check	1.5
			5	Health Check Log	1.5
	12	Monitoring	1	Container Insights	1.0
	13	Continuous Delivery	1	Continuous Delivery Test	1.5
	총점				30

3) 채점내용

순번	사전준비	
0	1) skills-bastion 서버에 SSH를 통해 접근합니다. 2) <code>rm -rf ~/.aws</code> 를 진행합니다. 3) aws configure를 입력하고 default.region을 ap-northeast-2으로 설정합니다. 4) 채점 명령어에 사용될 환경 변수를 설정합니다. <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <pre>\$ BUCKET_NAME=skills-chart-bucket-<영문 4자리> \$ GITHUB_USER=\$(gh api user --jq .login)</pre> </div>	
순번	채점항목	
1	1-1-A (명령어 입력)	<pre>aws ec2 describe-vpcs --filter Name=tag:Name,Values=skills-hub-vpc --query "Vpcs[].CidrBlock" \ ; aws ec2 describe-vpcs --filter Name=tag:Name,Values=skills-app-vpc --query "Vpcs[].CidrBlock"</pre>
	1-1-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	<pre>["10.0.0.0/16"] ["192.168.0.0/16"]</pre>

1	1-2-A (명령어 입력)	<pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-hub-subnet-a --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-hub-subnet-b --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-inspect-subnet-a --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-inspect-subnet-b --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-app-subnet-a --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-app-subnet-b --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-workload-subnet-a --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-workload-subnet-b --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-db-subnet-a --query "Subnets[0].CidrBlock" \ ; aws ec2 describe-subnets --filter Name=tag:Name,Values=skills-db-subnet-b --query "Subnets[0].CidrBlock"</pre>
	1-2-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	<pre>"10.0.0.0/24" "10.0.1.0/24" "10.0.2.0/24" "10.0.3.0/24" "192.168.0.0/24" "192.168.1.0/24" "192.168.2.0/24" "192.168.3.0/24" "192.168.4.0/24" "192.168.5.0/24"</pre>
1	1-3-A (명령어 입력)	aws ec2 describe-vpc-endpoints --query "VpcEndpoints[].ServiceName"
	1-3-A (예상 출력)	<pre>ecr.dkr, s3가 존재해야 합니다. vpce-svc가 포함된 문자열이 3개 있어야 합니다. ["com.amazonaws.ap-northeast-2.ecr.dkr", "com.amazonaws.ap-northeast-2.s3", "com.amazonaws.vpce.ap-northeast-2.vpce-svc-0ef7897ccf32a4097", "com.amazonaws.vpce.ap-northeast-2.vpce-svc-0a3b80fd9bb500a1f", "com.amazonaws.vpce.ap-northeast-2.vpce-svc-0bba008e8e6bc131f"]</pre>

2	2-1-A (명령어 입력)	aws ec2 describe-vpc-peering-connections --filters "Name=tag:Name,Values=skills-peering" --query "VpcPeeringConnections[?Status.Code=='active'].{PeeringId:VpcPeeringConnectionI d,Requester:RequesterVpcInfo.VpcId,Acceptor:AcceptorVpcInfo.VpcId,Status:Status. Code}" --output json
	2-1-A (예상 출력)	pcx, vpc, vpc 가 각각 포함되어야 합니다. [{ "PeeringId": " pcx -03ff7de2f279d6bbf", "Requester": " vpc -09c86a22e990d0e63", "Acceptor": " vpc -0b3017b5748c862d6", "Status": "active" }]
3	3-1-A (명령어 입력)	FIREWALL_NAME=\$(aws network-firewall describe-firewall --firewall-name skills-firewall \ --query 'Firewall.FirewallName' --output text) FIREWALL_POLICY_NAME=\$(aws network-firewall describe-firewall --firewall-name skills-firewall \ --query 'Firewall.FirewallPolicyArn' --output text awk -F/ '{print \$NF}') SUBNET_NAMES=\$(for subnet_id in \$(aws network-firewall describe-firewall --firewall-name skills-firewall \ --query 'Firewall.SubnetMappings[].SubnetId' --output text); do aws ec2 describe-subnets --subnet-ids \$subnet_id \ --query 'Subnets[].Tags[?Key=='Name'].Value' --output text done paste -sd "," -) echo "\$FIREWALL_NAME" echo "\$FIREWALL_POLICY_NAME" echo "\$SUBNET_NAMES"
	3-1-A (예상 출력) 정확히 일치 순서 중요	skills-firewall skills-firewall-policy skills-inspect-subnet-a,skills-inspect-subnet-b

3	3-2-A (명령어 입력)	curl --max-time 10 ifconfig.me
	3-2-A (예상 출력)	Operation timed out이 포함되어야 합니다. curl: (28) Operation timed out after 10002 milliseconds with 0 bytes received
	3-3-A (명령어 입력)	aws ec2 describe-security-groups \\\n--filters Name=group-name,Values=skills-bastion-sg \\\n--query "length(SecurityGroups[0].IpPermissionsEgress)" \\\n--output text
	3-3-A (예상 출력)	1
	3-4-A (명령어 입력)	curl --max-time 10 ifconfig.io
	3-4-A (예상 출력) <u>IP 출력</u>	52.79.107.118

4	4-1-A (명령어 입력)	aws ec2 describe-instances --filter Name=tag:Name,Values=skills-bastion --query "Reservations[].Instances[].InstanceType"
	4-1-A (예상 출력) 정확히 일치	["t3.small"]
	4-2-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"
	4-2-A (예상 출력) 순서 무관	위 IP와 동일한 IP가 아래에 있는지 확인 ["52.79.107.118"] ["13.124.182.24", "15.165.78.229", "43.203.26.79", "52.78.18.240", "52.79.107.118"]
	4-3-A (명령어 입력)	INSTANCE_NAME_TAG="skills-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"
	4-3-A (예상 출력)	Amazon Linux 가 포함되어야 합니다. skills-bastion-sg 가 일치해야 합니다. Amazon Linux 2023 AMI 2023.7.20250512.0 x86_64 HVM kernel-6.1 skills-bastion-sg

	4-4-A (명령어 입력)	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
	4-4-A (예상 출력) 정확히 일치	<pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": "*", "Resource": "*" }] }</pre>
5	5-1-A (명령어 입력)	aws secretsmanager describe-secret --secret-id skills-secrets --query '{Name:Name, KmsKeyId:KmsKeyId}' --output json
	5-1-A (예상 출력) 정확히 일치	Name에 대한 value 값은 일치하고, KmsKeyId에 대한 value 값은 arn:aws:kms: 로 시작해야 합니다. <pre>{ "Name": "skills-secrets", "KmsKeyId": "arn:aws:kms:..." }</pre>
	5-2-A (명령어 입력)	kubect exec -n skills \$(kubect exec -n skills -l app=green -o name head -n1 cut -d'/' -f2) -- env grep DB
	5-2-A (예상 출력) 순서 무관	아래의 환경변수가 포함되어 출력되는지 확인합니다. DB_USER DB_PASSWD DB_URL

6	6-1-A (명령어 입력)	aws rds describe-db-clusters --db-cluster-identifier skills-db-cluster --query 'DBClusters[0].EngineVersion' --output text \\ ; aws rds describe-db-clusters --db-cluster-identifier skills-db-cluster --query 'DBClusters[0].MasterUsername' --output text \\ ; aws rds describe-db-instances --query 'DBInstances[?DBClusterIdentifier=='skills-db-cluster'].DBInstanceClass' --output text
	6-1-A (예상 출력) 정확히 일치 순서 중요	8.0.mysql_aurora.3.08.2 admin db.t3.medium <-- 여러 개 출력되어도 무관
	6-2-A (명령어 입력)	aws rds describe-db-clusters --db-cluster-identifier skills-db-cluster --query "DBClusters[0].BacktrackWindow" --output text
	6-2-A (예상 출력)	양의 정수가 출력되어야 합니다. 14400
7	7-1-A (명령어 입력)	aws s3api list-buckets --query "Buckets[].Name" --output text
	7-1-A (예상 출력) 정확히 일치	skills-chart-bucket-<영문 4자리>
	7-2-A (명령어 입력)	aws s3 ls s3://\$BUCKET_NAME/app/ --recursive grep '.tgz'
	7-2-A (예상 출력)	버전값 제외 .tgz로 끝나는 문자열이 하나만 출력되어야 합니다. app/app-0.1.0.tgz
	7-3-A (명령어 입력)	argocd app get green -o json jq '.spec.sources[0].repoURL // null' argocd app get red -o json jq '.spec.sources[0].repoURL // null'
	7-3-A (예상 출력)	영문 4자리 제외, s3:// 로 시작되고 app 으로 끝나야 합니다. "s3://skills-chart-bucket-abcd/app" "s3://skills-chart-bucket-abcd/app"

8	8-1-A (명령어 입력)	aws ecr describe-repositories --repository-names "skills-green-repo" "skills-red-repo" --query "repositories[].{imageTagMutability:imageTagMutability,scanOnPush:imageScanning Configuration.scanOnPush,encryptionConfiguration:encryptionConfiguration.encryp tionType}"
	8-1-A (예상 출력) <u>정확히 일치</u>	[{ "imageTagMutability": "IMMUTABLE", "scanOnPush": true, "encryptionConfiguration": "KMS" }, { "imageTagMutability": "IMMUTABLE", "scanOnPush": true, "encryptionConfiguration": "KMS" }]
	8-2-A (명령어 입력)	aws ecr describe-images --repository-name skills-green-repo --query "imageDetails[].imageTags[]" aws ecr describe-images --repository-name skills-red-repo --query "imageDetails[].imageTags[]"
	8-2-A (예상 출력) <u>정확히 일치</u>	["v1.0.0"] ["v1.0.0"]

9	9-1-A (명령어 입력)	aws eks describe-cluster --name skills-eks-cluster --query 'cluster.version' --output text \\ ; aws eks describe-cluster --name skills-eks-cluster --query 'cluster.logging.clusterLogging[].types' jq . \\ ; aws eks describe-cluster --name skills-eks-cluster --query "cluster.resourcesVpcConfig.[endpointPublicAccess, endpointPrivateAccess]"
	9-1-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	1.32 [["api", "audit", "authenticator", "controllerManager", "scheduler"]] [false, true]
	9-2-A (명령어 입력)	aws eks describe-cluster --name skills-eks-cluster --query "cluster.encryptionConfig[].provider.keyArn" --output text
	9-2-A (예상 출력)	arn:aws:kms로 시작해야 합니다. arn:aws:kms :ap-northeast-2...
	9-3-A (명령어 입력)	kubectl get node -l skills=app -o json jq -r '.items[].metadata.labels."eks.amazonaws.com/nodegroup"' kubectl get nodes -l skills=app -o json jq -r '.items[].metadata.name' kubectl get nodes -l skills=app -o json jq -r '.items[] .metadata.labels["beta.kubernetes.io/instance-type"]'
	9-3-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	IP 제외 모두 동일해야 합니다. skills-app-nodegroup skills-app-nodegroup ip-192-168-2-229.ap-northeast-2.compute.internal ip-192-168-3-197.ap-northeast-2.compute.internal t3.medium t3.medium

9	9-4-A (명령어 입력)	kubectl get node -l skills=addon -o json jq -r 'items[].metadata.labels["eks.amazonaws.com/nodegroup"]' kubectl get nodes -l skills=addon -o json jq -r 'items[].metadata.name' kubectl get nodes -l skills=addon -o json jq -r 'items[] .metadata.labels["beta.kubernetes.io/instance-type"]'
	9-4-A (예상 출력) 정확히 일치 순서 중요	IP 제외 모두 동일해야 합니다. skills-addon-nodegroup skills-addon-nodegroup ip-192-168-2-226.ap-northeast-2.compute.internal ip-192-168-3-253.ap-northeast-2.compute.internal t3.medium t3.medium
	9-5-A (명령어 입력)	aws eks describe-fargate-profile --cluster-name skills-eks-cluster --fargate-profile-name skills-fargate-profile --query "fargateProfile.fargateProfileName"
	9-5-A (예상 출력)	"skills-fargate-profile"
	9-6-A (명령어 입력)	kubectl get deploy -n skills
	9-6-A (예상 출력)	NAME 및 READY가 일치해야 합니다. NAME READY UP-TO-DATE AVAILABLE AGE green-deploy 2/2 2 2 177m red-deploy 2/2 2 2 177m

10	10-1-A (명령어 입력)	aws elbv2 describe-load-balancers --query "LoadBalancers[].{Name:LoadBalancerName,Type:Scheme,Zones:AvailabilityZones[],ZoneName}" --output json
	10-1-A (예상 출력) <u>순서 무관</u>	강조된 부분이 모두 일치해야 합니다. [{ "Name": "skills-internal-nlb", "Type": "internal", ... }, { "Name": "skills-nlb", "Type": "internet-facing", ... }, { "Name": "skills-alb", "Type": "internal", ... }]
	10-2-A (명령어 입력)	EXTERNAL_NLB_DNS=\$(aws elbv2 describe-load-balancers --names skills-nlb --query "LoadBalancers[].DNSName" --output text) curl http://\$EXTERNAL_NLB_DNS/green -X POST -H 'Content-Type: application/json' -d '{"x": "alice", "y": 21}'
	10-2-A (예상 출력)	id 값은 달라도 무관합니다. {"id": "3Jzpxlnl", "status": "inserted"}
	10-3-A (명령어 입력)	EXTERNAL_NLB_DNS=\$(aws elbv2 describe-load-balancers --names skills-nlb --query "LoadBalancers[].DNSName" --output text) curl http://\$EXTERNAL_NLB_DNS/red -X POST -H 'Content-Type: application/json' -d '{"name": "bob"}
	10-3-A (예상 출력)	id 값은 달라도 무관합니다. {"id": "cLTOrVPi", "status": "inserted"}

11	11-1-A (명령어 입력)	aws opensearch list-domain-names grep skills-opensearch
	11-1-A (예상 출력) 정확히 일치	"DomainName": "skills-opensearch",
	11-2-A (명령어 입력)	aws opensearch describe-domain --domain-name skills-opensearch --query "DomainStatus.ClusterConfig.[InstanceCount, DedicatedMasterCount]" aws opensearch describe-domain --domain-name skills-opensearch --query "DomainStatus.EngineVersion"
	11-2-A (예상 출력) 정확히 일치 순서 중요	[2, 3] "OpenSearch_2.19"
	11-3-A (명령어 입력)	OPENSEARCH_ENDPOINT=\$(aws opensearch describe-domain --domain-name skills-opensearch jq -r '.DomainStatus.Endpoint') curl -s -u admin:Skill53## "https://\$OPENSEARCH_ENDPOINT/_cat/indices?v" grep "app-log"
	11-3-A (예상 출력)	app-log 가 포함되어야 합니다. green open app-log
	11-4-A (명령어 입력)	curl --silent \ -u "admin:Skill53##" \ "https://\${OPENSEARCH_ENDPOINT}/app-log/_search?size=1" \ jq -r '.hits.hits[0]._source'
	11-4-A (예상 출력)	date, timestamp, method, path, ip, port, tag 키가 모두 존재해야 합니다. 다른 키가 추가로 존재하거나, 현재 키가 모두 존재하지 않으면 오답 처리합니다. { " date ": "2025/06/16", " timestamp ": "2025-06-16T09:06:51Z", " method ": "POST", " path ": "/red", " ip ": "192.168.2.233", " port ": "63708", " tag ": "red" }
	11-5-A (명령어 입력)	curl --silent -u "admin:Skill53##" "https://\${OPENSEARCH_ENDPOINT}/app-log/_search?q=path:%22/health%22&size=1" jq -r '.hits.hits[0]._source'
	11-5-A (예상 출력)	null

	<div>12-1-A</div> <div>(콘솔 진행)</div>	<div>AWS CloudWatch 콘솔 -> Container Insights로 이동</div> <div>▼ Insights</div> <div>Container Insights</div>
<div>12</div>	<div>12-1-A</div> <div>(예상 출력)</div>	<div>사진과 같이 정상적으로 그래프가 표시되어야 합니다.</div> <div> <div>Container Insights Service: EKS</div> <div>Add to dashboard</div> <div>View in maps</div> <div>View performance dashboards</div> </div> <div> <div> <div>Clusters state summary (2)</div> <div>As of May 28, 2025, 오후 03:56 (UTC+09:00)</div> <div> <div>Clusters with alarms</div> <div>0 In alarm 0 OK</div> </div> <div> <div>Clusters without alarms</div> <div>0 High utilization 2 Low utilization</div> </div> <div> </div> </div> <div> <div>Performance and status summary</div> <div>Last 1 min</div> <div> <div>Clusters CPU (avg)</div> <div>Utilization: 2% Reserved: 27%</div> </div> <div> <div>Clusters Memory (avg)</div> <div>Utilization: 27% Reserved: 11%</div> </div> <div> <div>Pods (sum)</div> <div>Desired: 45 Ready: 45</div> </div> <div> <div>Nodes (sum)</div> <div>Unavailable: 0 Available: 4</div> </div> </div> <div> <div>Control plane summary</div> <div>Last 3 hours</div> <div> <div>202</div> <div>Max API server requests</div> </div> <div> <div>30.8</div> <div>Average API server requests latency</div> </div> <div> <div>121</div> <div>Total number of stored objects</div> </div> <div> <div>0</div> <div>Average admission controller latency</div> </div> </div> </div>

13	13-1-A (명령어 입력) 최대 1분 대기	<pre>EXTERNAL_NLB_DNS=\$(aws elbv2 describe-load-balancers --names skills-nlb --query "LoadBalancers[0].DNSName" --output text) export ID_GREEN=\$(curl -s -X POST -H 'Content-Type: application/json' -d '{"x": "charlie", "y": 21}' http://\$EXTERNAL_NLB_DNS/green jq -r '.id') export ID_RED=\$(curl -s -X POST -H 'Content-Type: application/json' -d '{"name": "dave"}' http://\$EXTERNAL_NLB_DNS/red jq -r '.id') curl http://\$EXTERNAL_NLB_DNS/green?id=\$ID_GREEN curl http://\$EXTERNAL_NLB_DNS/red?id=\$ID_RED echo; echo ===== cd /home/ec2-user/images AWS_REGION=\$(aws configure get region) ACCOUNT_ID=\$(aws sts get-caller-identity --query Account --output text) docker rmi -f \$(docker images) aws ecr get-login-password --region "\$AWS_REGION" docker login --username AWS --password-stdin "\$ACCOUNT_ID.dkr.ecr.\$AWS_REGION.amazonaws.com" > /dev/null cd green aws s3 cp s3://\$BUCKET_NAME/images/green_1.0.1 . aws ecr get-login-password --region \$AWS_REGION docker login --username AWS --password-stdin \$ACCOUNT_ID.dkr.ecr.\$AWS_REGION.amazonaws.com docker build -t \$ACCOUNT_ID.dkr.ecr.\$AWS_REGION.amazonaws.com/skills-green-repo:v1.0.1 . docker push \$ACCOUNT_ID.dkr.ecr.\$AWS_REGION.amazonaws.com/skills-green-repo:v1.0.1 cd ..; cd red aws s3 cp s3://\$BUCKET_NAME/images/red_1.0.1 . aws ecr get-login-password --region \$AWS_REGION docker login --username AWS --password-stdin \$ACCOUNT_ID.dkr.ecr.\$AWS_REGION.amazonaws.com docker build -t \$ACCOUNT_ID.dkr.ecr.\$AWS_REGION.amazonaws.com/skills-red-repo:v1.0.1 . docker push \$ACCOUNT_ID.dkr.ecr.\$AWS_REGION.amazonaws.com/skills-red-repo:v1.0.1 cd ~ gh repo clone "\$GITHUB_USER/day1-values" "day1-values-mark" cd /home/ec2-user/day1-values-mark for file in green.values.yaml red.values.yaml; do sed -i 's/tag: .*/tag: v1.0.1/g' "\$file" done git add green.values.yaml red.values.yaml git commit -m "chore: bump tag to v1.0.1 in values files" git push -u origin main argocd app sync green argocd app sync red echo wait 1 minutes sleep 1m EXTERNAL_NLB_DNS=\$(aws elbv2 describe-load-balancers --names skills-nlb --query "LoadBalancers[0].DNSName" --output text) curl -s http://\$EXTERNAL_NLB_DNS/green?id=\$ID_GREEN curl -s http://\$EXTERNAL_NLB_DNS/red?id=\$ID_RED</pre>
13	13-1-A (예상 출력)	강조된 버전이 정확히 일치해야 합니다. <pre>{ "version": "1.0.0", "x": "charlie", "y": 21 } {"name": "dave", "version": "1.0.0"}</pre> <p>=====</p>

