2025년도 경기기능대회 채점기준

1. 채점 시 유의사항	직종명	클라우드컴퓨팅

* 다음 사항을 유의하여 채점하시오.

1) AWS의 리전은 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) 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, jq, kubectl이 설치되어 있는지 확인합니다.
- Bastion에서 IAM Role이 매핑되어 AWS CLI로 AWS의 모든 리소스에 접근 가능한지 확
인합니다.
- aws sts get-caller-identity 명령을 통해 선수의 계정이 아닌 다른 계정에 접근 하고 있는지 확인합니다. 만약, 다른 계정이라면 부정행위를 의심할 수 있습니다.
12) 채점 전 채점환경 구성을 위해 ~/.aws/config에 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			직종명			클라우드컴퓨팅		
과제 번호	일련 번호	주요항목	배점	채점방법		채점시기		비고
				독립	합의	경기 진행중	경기 종료후	
제1과제	1	Networking			O		O	
	2	Bastion Server			O		O	
	3	Storage			O		O	
	3	Database			O		O	
	4	Security			O		O	
	5	Image Repository			O		O	
	6	Kubernetes			O		O	
	7	EMR			O		O	
합계								

2) 채점방법 및 기준

과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
제1과제	1	Networking	1	VPC 확인	
			2	Subnets 확인	
			3	Internet Gateway 확인	
			4	NAT Gateway 사용 확인	
	2	Bastion Server	1	인스턴스 타입 확인	
			2	Public IP 확인	
			3	OS 확인	
			4	SSH 포트번호 확인	
	3	Storage	1	S3 버킷 생성 확인	
			2	S3 버킷 암호화 설정 확인	
			3	S3 버전관리 활성화 확인	
	4	Database	1	RDS Subnet 확인	
			2	RDS 버전 확인	
			3	RDS 포트 확인	
			4	RDS 인스턴스 타입 확인	

			5	RDS 암호화 활성화 확인	
			6	RDS Standby 활성화 확인	
	5	Security	1	KMS 키 생성 확인	
			2	자동 키 회전 활성화 확인	
	6	Image Repository	1	ECR Repository 생성 확인	
			2	ECR Repository 암호화 확인	
			3	ECR Repository Scanning 확인	
			4	ECR Repository Immutable 확인	
	7	Kubernetes	1	EKS Cluster 버전 확인	
			2	EKS 프라이빗 엔드포인트 확인	
			3	EKS 로깅 설정 확인	
			4	ws-ddon 노드그룹 확인	
			5	ws-system 노드그룹 확인	
			6	Karpenter 설치 확인	
			7	Nodepool 구성 확인	
	8	EMR	1	EMR 가상 클러스터 구성 확인	
			2	PySpark 분석 스크립트 확인	
			3	분석 결과 확인	
			4	RDS 데이터베이스 연동 확인	
			5	EMR 작업 제출 및 모니터링 확인	

3) 채점 내용

순번	채점 항목
1-1	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws ec2 describe-vpcs --filter Name=tag:Name,Values=ws-vpc ₩ --query "Vpcs[].CidrBlock" 3) 10.100.0.0/16이 출력되는지 확인합니다. 0.5점
1-2	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws ec2 describe-subnets --filter Name=tag:Name,Values=ws-public -a --query "Subnets[].[AvailabilityZone, CidrBlock][]" 3) ap-northeast-2a와 10.0.1.0/24가 출력되는지 확인합니다. 0.1점 4) 아래 명령어를 입력합니다. aws ec2 describe-subnets --filter Name=tag:Name,Values=ws-public-b ₩ --query "Subnets[].[AvailabilityZone, CidrBlock][]" 5) ap-northeast-2b와 10.0.2.0/24가 출력되는지 확인합니다. 0.1점 6) 아래 명령어를 입력합니다. aws ec2 describe-subnets --filter Name=tag:Name,Values=ws-private-a ₩ --query "Subnets[].[AvailabilityZone, CidrBlock][]" 7) ap-northeast-2a와 10.0.3.0/24가 출력되는지 확인합니다. 0.1점 8) 아래 명령어를 입력합니다.

	aws ec2 describe-subnets --filter Name=tag:Name,Values=ws-private-b ₩ --query "Subnets[].[AvailabilityZone, CidrBlock][]" 9) ap-northeast-2b와 10.0.4.0/24가 출력되는지 확인합니다. 0.1점 10) 아래 명령어를 입력합니다. aws ec2 describe-subnets --filter Name=tag:Name,Values= ws-private-c ₩ --query "Subnets[].[AvailabilityZone, CidrBlock][]" 11) ap-northeast-2a와 10.0.5.0/24가 출력되는지 확인합니다. 0.1점 12) 아래 명령어를 입력합니다. aws ec2 describe-subnets --filter Name=tag:Name,Values= ws-private-d ₩ --query "Subnets[].[AvailabilityZone, CidrBlock][]" 13) ap-northeast-2b와 10.0.6.0/24가 출력되는지 확인합니다. 0.1점
1-3	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws ec2 describe-route-tables --filter Name=tag:Name,Values=ws-public-rtb ₩ --query "RouteTables[].Routes[].GatewayId" 3) igw- 로 시작하는 문구가 출력되는지 확인합니다. 4) 아래 명령어를 입력합니다. aws ec2 describe-internet-gateways --filter Name=tag:Name,Values=ws-igw ₩ --query "InternetGateways[].InternetGatewayId" 5) igw- 로 시작하는 문구가 2)에서 출력된 문구와 동일한지 확인합니다. 0.5점.
1-4	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws ec2 describe-nat-gateways 3) nat- 로 시작하는 문구가 2개 출력되는지 확인합니다. 정확히 2개여야 0.5점
2-1	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws ec2 describe-instances --filter Name=tag:Name,Values=ws-bastion-2025 ₩ --query "Reservations[].Instances[].InstanceType" 3) c5.large이 출력되는지 확인합니다. 0.5 점
2-2	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws ec2 describe-instances --filter Name=tag:Name,Values=ws-bastion-2025 ₩ --query "Reservations[].Instances[].PublicIpAddress" 3) 2)에서 출력된 IP를 기록합니다. 4) 아래 명령어를 입력합니다. aws ec2 describe-addresses --query "Addresses[].PublicIp" 5) 출력되는 IP 리스트 중에 3)에서 기록한 IP가 존재하는지 확인합니다. 0.5점
2-3	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws ec2 describe-instances --filter Name=tag:Name,Values=ws-bastion-2025 ₩ --query "Reservations[].Instances[].ImageId" 3) ami-로 시작하는 AMI ID를 기록합니다. 4) 아래 명령어를 입력합니다. aws ec2 describe-images --image-ids <2)에서 기록한 AMI ID> --query "Images[].Description" 5) Amazon Linux 2023이 포함된 문구가 출력되는지 확인합니다. 0.5점
2-4	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어로 현재 설정된 SSH 포트 번호를 확인합니다:

	aws ec2 describe-instances --filter Name=tag:Name,Values=ws-bastion-2025 ₩ --query "Reservations[].Instances[].SecurityGroups[].GroupId" --output text | ₩ xargs -I {} aws ec2 describe-security-group-rules ₩ --filter Name=group-id,Values={} ₩ --query "SecurityGroupRules[?IpProtocol=='tcp'].[FromPort,ToPort]" 3) 출력된 결과에서 22번 포트가 없는지 확인합니다. 0.5점
3-1	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. export AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text) aws s3api list-buckets --query 'Buckets[?Name==`ws-cc-raw-data- '${AWS_ACCOUNT_ID}'`].Name' --output text 3) ws-cc-raw-data-<본인의 ACCOUNT ID>가 출력되는지 확인합니다. 0.5점
3-2	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws s3api get-bucket-encryption --bucket ws-cc-raw-data-${AWS_ACCOUNT_ID} 2>/dev/null 3) aws:kms 문구와 함께 ws-secret-key가 출력되는지 확인합니다. 0.5점
3-3	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws s3api get-bucket-versioning --bucket ws-cc-raw-data-${AWS_ACCOUNT_ID} --query 'Status' --output text 3) Enabled가 출력되는지 확인합니다. 0.5점
4-1	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws rds describe-db-subnet-groups ₩ --query 'DBSubnetGroups[].{Name:DBSubnetGroupName,Description:DBSubnetGroupDescription}' 3) 명령어를 실행했을때 출력되는 서브넷 그룹 이름을 기록합니다. 4) 아래 명령어를 입력합니다 aws rds describe-db-subnet-groups ₩ --db-subnet-group-name <3에서 기록한 문구> ₩ --query 'DBSubnetGroups[].Subnets[].{SubnetId:SubnetIdentifier,AZ:SubnetAvailabilityZone}' 5) SubnetId가 2개만 출력되는지 확인합니다. 0.5 점 6) 아래 명령어를 입력합니다. aws ec2 describe-subnets ₩ --subnet-ids <4에서 출력된 문구의 첫번째> <4에서 출력된 문구의 두번째> ₩ --query "Subnets[].CidrBlock" 7) 10.0.4.0과 10.0.5.0 2개만 출력되는지 확인합니다. 하나라도 다르면 오답입니다. 0.5점
4-2	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws rds describe-db-engine-versions ₩ --engine postgres ₩ --query 'DBEngineVersions[-1].EngineVersion' ₩ --output text 3) 2)에서 출력된 버전을 기록합니다. 4) 아래 명령어를 입력합니다. aws rds describe-db-instances ₩ --db-instance-identifier ws-db-instance ₩ --query 'DBInstances[].EngineVersion' ₩ --output text 5) 4)의 결과가 3)에서 기록한 숫자와 같은지 확인합니다. 1점
4-3	1) SSH를 통해 Bastion 서버에 접근합니다.

	2) 아래 명령어를 입력합니다. aws rds describe-db-instances ₩ --db-instance-identifier ws-db-instance ₩ --query 'DBInstances[].Endpoint.Port' 3) 출력되는 포트 번호가 5433인지 확인합니다. 0.5점
4-4	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws rds describe-db-instances ₩ --db-instance-identifier ws-db-instance ₩ --query 'DBInstances[].DBInstanceClass' 3) db.t4g.medium이 출력되는지 확인합니다. 0.5점
4-5	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws rds describe-db-instances ₩ --db-instance-identifier ws-db-instance ₩ --query 'DBInstances[].StorageEncrypted' 3) true가 출력되는지 확인합니다. 0.5점
4-6	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws rds describe-db-instances ₩ --db-instance-identifier ws-db-instance ₩ --query 'DBInstances[].MultiAZ' 3) true가 출력되는지 확인합니다. 0.5점
5-1	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws kms list-aliases ₩ --query 'Aliases[?starts_with(AliasName, `alias/ws-secret-key`)].AliasName' 3) alias/ws-secret-key가 조회되는 것을 확인합니다. 0.5점
5-2	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws kms get-key-rotation-status ₩ --key-id $(aws kms list-aliases ₩ --query 'Aliases[?starts_with(AliasName, `alias/ws-secret-key`)].TargetKeyId' ₩ --output text) 3) KeyRotationEnabled가 true인지 확인합니다. 1점
6-1	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws ecr describe-repositories --repository-names ws-emr/spark ₩ --query "repositories[].repositoryName" 3) ws-emr/spark가 출력되는지 확인합니다. 1점
6-2	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws ecr describe-repositories ₩ --repository-names ws-emr/spark ₩ --query 'repositories[].{Name:repositoryName,Encryption:encryptionConfiguration}' 3) KMS가 출력되는지 확인합니다. 0.5점
6-3	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws ecr describe-repositories ₩ --repository-names ws-emr/spark ₩ --query 'repositories[].{Name:repositoryName,ScanOnPush:imageScanningConfiguration.scanOnPush

	}' 3) true가 출력되는지 확인합니다. 0.5점
6-4	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws ecr describe-repositories ₩ --repository-names ws-emr/spark ₩ --query 'repositories[].{Name:repositoryName,ImageTagMutability:imageTagMutability}' 3) IMMUTABLE이 출력되는지 확인합니다. 0.5점
7-1	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws eks describe-cluster ₩ --name ws-eks-cluster ₩ --query 'cluster.version' ₩ --output text 3) 버전이 1.32인지 확인합니다. 0.5점
7-2	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws eks describe-cluster ₩ --name ws-eks-cluster ₩ --query 'cluster.resourcesVpcConfig.{ PublicAccess:endpointPublicAccess, PrivateAccess:endpointPrivateAccess, PublicCIDRs:publicAccessCidrs }' 3) PublicAccess가 false이고 PrivateAccess가 true인지 확인합니다. 하나라도 다르면 오답입니다. 0.5점
7-3	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws eks describe-cluster ₩ --name ws-eks-cluster ₩ --query 'cluster.logging.clusterLogging[].{ Types:types, Enabled:enabled }' 3) api Type이 Enabled:true인지 확인합니다. 1점
7-4	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws eks describe-nodegroup --cluster-name ws-eks-cluster --nodegroup-name ws-addon --query "nodegroup.nodegroupName" 3) ws-addon이 출력되는지 확인합니다. 0.5점 4) 아래 명령어를 입력합니다. kubectl get no -l "eks.amazonaws.com/nodegroup=ws-addon" --output json ₩ | jq ".items[].metadata.labels | .₩"eks.amazonaws.com/nodegroup₩" + ₩" ₩" + .₩"topology.kubernetes.io/zone₩"" 5) ws-addon이 ap-northeast-2a와ws-addon ap-northeast-2b가 각각 1개 이상 출력되는지 확인합니다. 둘다 있어야 0.5점 6) 아래 명령어를 입력합니다. aws eks describe-nodegroup ₩ --cluster-name ws-eks-cluster ₩ --nodegroup-name ws-addon ₩ --query 'nodegroup.{ Name: nodegroupName,

	Status: status,
	InstanceType: instanceTypes[0],
	DesiredSize: scalingConfig.desiredSize,
	MinSize: scalingConfig.minSize,
	MaxSize: scalingConfig.maxSize,
	Labels: labels,
	Taints: taints[].{
	Key: key,
	Value: value,
	Effect: effect
	}
	}'
	7) 인스턴스 타입이 t3.xlarge인지 확인합니다. 0.5점
	8) 노드수가 최소 2개, 최대 3개로 설정되어 있는지 확인합니다. 0.5점
	9) Labels이"nodegroup": "addon"로 되어 있는지 확인합니다. 0.5점
	10) Taints이key=node-role.kubernetes.io/ws-addon, value=null, effect=NO_SCHEDULE로 되어 있는지 확인합니다. (3개 모두 일치해야 0.5점)

7-5	1) SSH를 통해 Bastion 서버에 접근합니다.
	2) 아래 명령어를 입력합니다.
	aws eks describe-nodegroup --cluster-name ws-eks-cluster --nodegroup-name ws-system --query "nodegroup.nodegroupName"
	3) ws-system이 출력되는지 확인합니다. 0.5점
	4) 아래 명령어를 입력합니다.
	kubectl get no -l "eks.amazonaws.com/nodegroup=ws-system" --output json ₩
	| jq ".items[].metadata.labels | .₩"eks.amazonaws.com/nodegroup₩" + ₩" ₩" + .₩"topology.kubernetes.io/zone₩""
	5) ws-system이 ap-northeast-2a와ws-addon ap-northeast-2b가 각각 1개 이상 출력되는지 확인합니다. 둘다 있어야 0.5점
	6) 아래 명령어를 입력합니다.
	aws eks describe-nodegroup ₩
	--cluster-name ws-eks-cluster ₩
	--nodegroup-name ws-system ₩
	--query 'nodegroup.{
	Name: nodegroupName,
	Status: status,
	DesiredSize: scalingConfig.desiredSize,
	MinSize: scalingConfig.minSize,
	MaxSize: scalingConfig.maxSize,
	Labels: labels,
	Taints: taints[].{
	Key: key,
	Value: value,
	Effect: effect
	}
	}'
	7) 노드수가 최소 2개, 최대 3개로 설정되어 있는지 확인합니다. 0.5점
	8) Labels이"nodegroup": "system"로 되어 있는지 확인합니다. 0.5점

7-6	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. kubectl get po -n karpenter 3) karpenter- 로 시작하는 파드만 존재하는지 확인합니다. (karpenter- 파드만 있어야 0.5점)

	4) kubectl get nodes -l eks.amazonaws.com/nodegroup=ws-addon 위 명령어를 통해 출력되는 NAME을 기록합니다. 5) 아래 명령어를 입력합니다. kubectl get po -n karpenter -o wide 7) 4에서 기록한 NAME과 주소가 일치하는지 확인합니다 . 모두 일치해야 0.5점 8) 아래 명령어를 입력합니다. kubectl get pods -n karpenter -l app.kubernetes.io/name=karpenter -o json | jq '.items[].spec.tolerations[] | select(.key == "node-role.kubernetes.io/ws-addon")' 9) 일치하는 값이 2개 나오는지 확인합니다. 0.5점
7-7	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. kubectl get nodepool -o json | jq '{ "Node Labels": .items[].spec.template.metadata.labels, "Node Taints": .items[].spec.template.spec.taints, "Requirements": .items[].spec.template.spec.requirements, "Resource Limits": .items[].spec.limits, "Consolidation After": .items[].spec.disruption.consolidateAfter }' 3) Node Labels이"eks.amazonaws.com/compute-type": "emr", "nodegroup": "emr"인지 확인합니다. 0.5점 4) Node Taints가"effect":"NoSchedule","key": "eks.amazonaws.com/compute-type","value": "emr"인지 확인합니다. 0.5점 5) Requirements가 "amd64", "linux", "on-demand" 를 가지는지 확인합니다. 0.5점 6) Requirements가node.kubernetes.io/instance-family에서 m5,m5d,c5,c5d,c4,r4를 모두 포함하는지 확인합니다. 정확히 일치해야 0.5점 7) node.kubernetes.io/instance-cpu에서 4,8,16,32를 모두 포함하는지 확인합니다. 정확히 일치해야 0.5점 8) Resource Limits에서 cpu:40, memory:160Gi, Consolidation After:90s 인지 확인합니다. 정확히 일치해야 0.5점
8-1	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws eks list-clusters aws eks describe-cluster ₩ --name ws-eks-cluster ₩ --query 'cluster' | grep '"status"' 3)출력에 "emr-eks-cluster"클러스터 이름을 확인하고 상태가 "ACTIVE"여야 합니다. 0.5점 4) SSH를 통해 Bastion 서버에 접근합니다. 5) 아래 명령어를 입력합니다. aws emr-containers list-virtual-clusters aws iam list-roles | grep -E "EMR|EKS" 6) 출력에 가상 클러스터가 있고 상태가 "RUNNING"이어야 합니다. 0.5점 IAM 권한에는 EMR 및 EKS 관련 역할만이 존재해야 합니다.0.5점
8-2	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 계정 ID와 버킷 이름을 환경변수로 설정합니다. export ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text) export S3_BUCKET=ws-cc-raw-data-${ACCOUNT_ID} 3) 아래 명령어를 입력하여 파일이 존재하는지 확인합니다. 0.2 점 aws s3 ls s3://${S3_BUCKET}/credit_card_analysis.py 4) 스크립트 내용 확인 후, 데이터 로드, 그룹화, 필터링 코드가 있어야 합니다. 0.8점
8-3	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다.

	aws s3 ls s3://${S3_BUCKET}/output/fraud_analysis_result/ 3) 아래 디렉토리들이 존재하는지 확인합니다. 모두 있어야0.5점 # - summary/ # - outliers/ # - top_outliers/ 4) 아래 명령어를 입력합니다. aws s3 ls s3://${S3_BUCKET}/output/fraud_analysis_result/summary/ aws s3 ls s3://${S3_BUCKET}/output/fraud_analysis_result/outliers/ 5) Parquet 파일이 존재하는지 확인합니다. 0.5점
8-4	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. aws rds describe-db-instances ₩ --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]' ₩ --output table 3) RDS 인스턴스가 존재하고 상태가 "available"이어야 합니다. 0.5점 4) 아래 명령어로 RDS 엔드포인트를 환경변수에 저장합니다. export RDS_ENDPOINT=$(aws rds describe-db-instances ₩ --query "DBInstances[0].Endpoint.Address" ₩ --output text) export SECRET_ID=$(aws secretsmanager list-secrets ₩ --filters Key=name,Values=rds! ₩ --query 'SecretList[0].Name' ₩ --output text) export PGPASSWORD=$(aws secretsmanager get-secret-value ₩ --secret-id ${SECRET_ID} ₩ --query 'SecretString' ₩ --output text \| jq -r '.password') 5) 저장된 엔드포인트를 사용하여 아래 명령어를 입력합니다. PGPASSWORD=${PGPASSWORD} psql -h ${RDS_ENDPOINT} -p 5433 -U wsadmin -d fraud_detection -c "₩dt" 6) fraud_summary 및 fraud_top_outliers 테이블이 존재해야 합니다. 모두 있어야 0.5점
8-5	1) SSH를 통해 Bastion 서버에 접근합니다. 2) 아래 명령어를 입력합니다. cat /home/ec2-user/submit-spark-job.sh 3) EMR on EKS 작업 제출 명령어가 포함되어 있어야 합니다. 0.5점 4) 아래 명령어를 입력합니다. export VIRTUAL_CLUSTER_ID=$(aws emr-containers list-virtual-clusters --query "virtualClusters[0].id" --output text) export JOB_RUN_ID=$(aws emr-containers list-job-runs --virtual-cluster-id ${VIRTUAL_CLUSTER_ID} --query "jobRuns[0].id" --output text) 5) 작업 상태가 "COMPLETED"인지 확인합니다. 0.5점