

제59회 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 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	Networking	6.5		○		○	
	2	Application	4		○		○	
	3	Load Balancer	1.5		○		○	
	4	RDBMS	3		○		○	
	5	No-SQL	3		○		○	
	6	ECR	1.5		○		○	
	7	Network Firewall	6		○		○	
	8	EKS	3.5		○		○	
	9	CloudWatch	1		○		○	
합 계			30					

2) 채점방법 및 기준

과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
1과제	1	Networking	1	VPC	1.5
			2	Subnets	1.5
			3	Transit Gateway	1.5
			4	workload private nat	1.5
			5	bastion host	0.5
	2	Application	1	static application	0.5
			2	customer api	1.0
			3	product api	1.0
			4	order api	1.5
	3	Load Balancer	1	alb	1.5
	4	RDBMS	1	rds cluster	1.5
			2	db, table	1.5
	5	No-SQL	1	dynamodb	1.5
			2	dynamodb item	1.5
	6	ECR	1	ecr	1.5
	7	Network Firewall	1	network firewall subnet	1.5
			2	network firewall policies	1.5
			3	connection test - deny	1.5
			4	connection test - allow	1.5
	8	EKS	1	eks cluster	1.5
			2	eks cluster node group	1.5
			3	eks deployment	0.5
	9	CloudWatch	1	cloudwatch dashboard	1.0
	총점				30

3) 채점내용

순번	사전준비
0	1) wsc-prod-bastion 서버에 Session Manager를 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 셸을 초기 실행할 때 다음 명령어를 실행하여 환경 변수를 초기화합니다. (채점 스크립트로 진행 시 생략)
	6) 채점을 진행하기 전에 다음 명령어를 수행하여 채점 진행을 위한 사전 작업을 진행합니다. (채점 스크립트로 진행 시 생략)
	<pre># set default region of aws cli aws configure set default.region ap-northeast-2</pre>

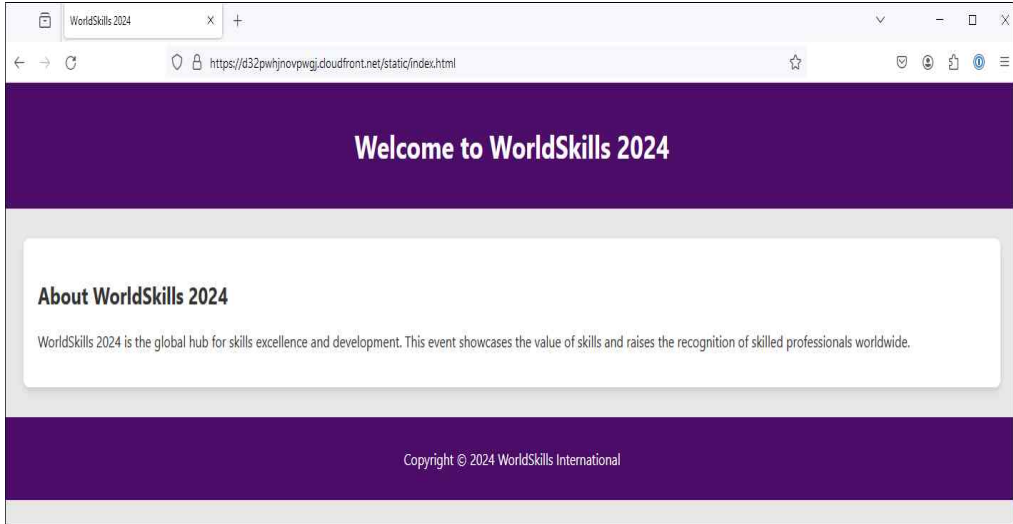
순번	채점 항목	
1-1	1-1-A (명령어 입력)	<pre>aws ec2 describe-vpcs --filter Name=tag:Name,Values=wsc-prod-vpc --query "Vpcs[0].CidrBlock" ₩ ; aws ec2 describe-vpcs --filter Name=tag:Name,Values=wsc-inspect-vpc --query "Vpcs[0].CidrBlock" ₩ ; aws ec2 describe-vpcs --filter Name=tag:Name,Values=wsc-ingress-vpc --query "Vpcs[0].CidrBlock" ₩ ; aws ec2 describe-vpcs --filter Name=tag:Name,Values=wsc-egress-vpc --query "Vpcs[0].CidrBlock"</pre>
	1-1-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	<pre>"10.100.0.0/16" "100.64.0.0/16" "172.20.0.0/16" "172.22.0.0/16"</pre>

1-2	1-2-A (명령어 입력)	<pre># aws ec2 describe-subnets --filters ₩ "Name=tag:Name,Values=wsc-prod-peering-sn-a,wsc-prod-peering-sn-c,wsc-prod-workload-sn-a,wsc-prod-workload-sn-c,wsc-prod-protect-sn-a,wsc-prod-protect-sn-c,wsc-inspect-secure-sn-a,wsc-inspect-secure-sn-c,wsc-inspect-peering-sn-a,wsc-inspect-peering-sn-c,wsc-ingress-pub-sn-a,wsc-ingress-pub-sn-c,wsc-ingress-peering-sn-a,wsc-ingress-peering-sn-c,wsc-egress-pub-sn-a,wsc-egress-pub-sn-c,wsc-egress-peering-sn-a,wsc-egress-peering-sn-c" ₩ --query "Subnets[*].{Tag:Tags[?Key=='Name'] [0].Value,CidrBlock:CidrBlock}" ₩ --output text</pre>
	1-2-A (예상 출력) <u>정확히 일치</u> <u>순서 상관 X</u>	<pre>100.64.0.64/28 wsc-inspect-peering-sn-a 10.100.2.0/24 wsc-prod-peering-sn-c 100.64.0.48/28 wsc-inspect-secure-sn-c 172.22.0.64/28 wsc-egress-pub-sn-c 172.22.0.128/28 wsc-egress-peering-sn-c 172.22.0.32/28 wsc-egress-pub-sn-a 172.20.0.128/28 wsc-ingress-peering-sn-c 172.20.0.64/28 wsc-ingress-pub-sn-c 10.100.11.0/24 wsc-prod-workload-sn-c 10.100.21.0/24 wsc-prod-protect-sn-c 10.100.10.0/24 wsc-prod-workload-sn-a 172.20.0.96/28 wsc-ingress-peering-sn-a 172.22.0.96/28 wsc-egress-peering-sn-a 100.64.0.32/28 wsc-inspect-secure-sn-a 100.64.0.80/28 wsc-inspect-peering-sn-c 10.100.20.0/24 wsc-prod-protect-sn-a 10.100.1.0/24 wsc-prod-peering-sn-a 172.20.0.32/28 wsc-ingress-pub-sn-a</pre>

1-3	<p>1-3-A (명령어 입력)</p>	<pre> tgw_id=\$(aws ec2 describe-transit-gateways --query "TransitGateways[?Tags[?Value=='wsc-vpc-tgw']].TransitGatewayId" --output text) attachment_ids=\$(aws ec2 describe-transit-gateway-attachments --filter "Name=transit-gateway-id,Values=\$tgw_id" --query "TransitGatewayAttachments[*].TransitGatewayAttachmentId" --output text) subnet_ids=\$(aws ec2 describe-transit-gateway-vpc-attachments --transit-gateway-attachment-ids \$attachment_ids --query "TransitGatewayVpcAttachments[*].SubnetIds[]" --output text) for subnet_id in \$subnet_ids; do subnet_name=\$(aws ec2 describe-tags --filters "Name=resource-id,Values=\$subnet_id" "Name=key,Values=Name" --query "Tags[0].Value" --output text) echo \$subnet_name done </pre>
	<p>1-3-A (예상 출력) <u>순서 상관 X</u></p>	<p>출력된 모든 결과에 peering 이라는 문자열이 포함되어있는 지 확인 (만약, 출력 결과에 peering이 포함되지 않을 경우 해당 항목 0점 처리)</p> <pre> wsc-inspect-peering-sn-a wsc-inspect-peering-sn-c wsc-egress-peering-sn-a wsc-egress-peering-sn-c wsc-prod-peering-sn-c wsc-prod-peering-sn-a wsc-ingress-peering-sn-a wsc-ingress-peering-sn-c </pre>

1-4	<div>1-4-A</div> <div>(명령어 입력)</div>	<pre> VPC_ID=\$(aws ec2 describe-vpcs --filters "Name=tag:Name,Values=wsc-prod-vpc" --query "Vpcs[*].VpcId" --output text) SUBNET_A_ID=\$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=\$VPC_ID" "Name=tag:Name,Values=wsc-prod-workload-sn-a" --query "Subnets[*].SubnetId" --output text) SUBNET_C_ID=\$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=\$VPC_ID" "Name=tag:Name,Values=wsc-prod-workload-sn-c" --query "Subnets[*].SubnetId" --output text) ROUTE_TABLE_A_ID=\$(aws ec2 describe-route-tables --filters "Name=association.subnet-id,Values=\$SUBNET_A_ID" --query "RouteTables[*].RouteTableId" --output text) ROUTE_TABLE_C_ID=\$(aws ec2 describe-route-tables --filters "Name=association.subnet-id,Values=\$SUBNET_C_ID" --query "RouteTables[*].RouteTableId" --output text) NAT_GATEWAY_ID_A=\$(aws ec2 describe-route-tables --route-table-ids \$ROUTE_TABLE_A_ID --query "RouteTables[*].Routes[?NatGatewayId].NatGatewayId" --output text) NAT_GATEWAY_ID_C=\$(aws ec2 describe-route-tables --route-table-ids \$ROUTE_TABLE_C_ID --query "RouteTables[*].Routes[?NatGatewayId].NatGatewayId" --output text) echo \$NAT_GATEWAY_ID_A ; echo \$NAT_GATEWAY_ID_C </pre> <div>1-4-A</div> <div>(예상 출력)</div> <div>순서 상관 X</div> <pre> nat-07ad696ba6e0a909d <- nat-* 로 시작하는 지 확인 nat-0f87ade00e4e389a9 <- nat-* 로 시작하는 지 확인 </pre>
-----	--------------------------------------	---

1-4	1-4-B (명령어 입력)	SUBNET_A_TYPE=\$(aws ec2 describe-route-tables --filters "Name=association.subnet-id,Values=\$NAT_SUBNET_A_ID" --query "RouteTables[*].Routes[?GatewayId == 'igw-*'].GatewayId" --output text) SUBNET_C_TYPE=\$(aws ec2 describe-route-tables --filters "Name=association.subnet-id,Values=\$NAT_SUBNET_C_ID" --query "RouteTables[*].Routes[?GatewayId == 'igw-*'].GatewayId" --output text) NAT_SUBNET_A_ID=\$(aws ec2 describe-nat-gateways --nat-gateway-ids \$NAT_GATEWAY_ID_A --query "NatGateways[*].SubnetId" --output text) NAT_SUBNET_C_ID=\$(aws ec2 describe-nat-gateways --nat-gateway-ids \$NAT_GATEWAY_ID_C --query "NatGateways[*].SubnetId" --output text) aws ec2 describe-subnets --subnet-ids \$NAT_SUBNET_A_ID --query "Subnets[*].Tags[?Key=='Name'].Value" --output text aws ec2 describe-nat-gateways --nat-gateway-ids \$NAT_GATEWAY_ID_A --query "NatGateways[*].ConnectivityType" --output text aws ec2 describe-subnets --subnet-ids \$NAT_SUBNET_C_ID --query "Subnets[*].Tags[?Key=='Name'].Value" --output text aws ec2 describe-nat-gateways --nat-gateway-ids \$NAT_GATEWAY_ID_C --query "NatGateways[*].ConnectivityType" --output text
	1-4-B (예상 출력) <u>정확히 일치</u> <u>순서 상관 X</u>	wsc-prod-peering-sn-a private wsc-prod-peering-sn-c private
1-5	1-5-A (명령어 입력)	INSTANCE_DETAILS=\$(aws ec2 describe-instances --filters "Name=tag:Name,Values=wsc-prod-bastion" --query "Reservations[*].Instances[*.{InstanceType:InstanceType}]" --output text) SUBNET_ID=\$(aws ec2 describe-instances --filters "Name=tag:Name,Values=wsc-prod-bastion" --query "Reservations[*].Instances[*.{SubnetId:SubnetId}]" --output text) SUBNET_NAME=\$(aws ec2 describe-subnets --subnet-ids \$SUBNET_ID --query "Subnets[*].Tags[?Key=='Name'].Value" --output text) echo "\$INSTANCE_DETAILS" echo "\$SUBNET_NAME"
	1-5-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	t3.medium wsc-prod-workload-sn-c

2-1	2-1-A (명령어 입력)	<pre>distribution_ids=\$(aws cloudfront list-distributions --query "DistributionList.Items[*].Id" --output text) account_id=\$(aws sts get-caller-identity --query "Account" --output text) for id in \$distribution_ids; do tags=\$(aws cloudfront list-tags-for-resource --resource "arn:aws:cloudfront::\$account_id:distribution/\$id" --query "Tags.Items" --output json) if echo \$tags jq -e '.[] select(.Key=="Name" and .Value=="wsc-prod-cdn")' > /dev/null; then domain=\$(aws cloudfront get-distribution --id \$id --query "Distribution.DomainName" --output text) echo "https://\$domain" fi done</pre>
	2-1-A (예상 출력)	<p>https://d32pwhjnovpwgj.cloudfront.net/static/index.html</p> <p>빨간 색으로 표시된 부분은 달라도 무관합니다.</p>
	2-1-B	2-1-A 예상 출력된 출력된 URL을 브라우저를 통해 접속
	2-1-B (예상 출력) <u>사진과 같은</u> <u>웹 페이지 출력</u>	

2-2	2-2-A (명령어 입력)	<pre>curl -XPOST -H "Content-Type: application/json" -d '{"id": "99999", "name": "kim", "gender": "male"}' https://\$domain/v1/customer</pre> <pre>curl -XPOST -H "Content-Type: application/json" -d '{"id": "100000", "name": "lee", "gender": "female"}' https://\$domain/v1/customer</pre>
	2-2-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	<pre>{"customer":{"id":"99999","name":"kim","gender":"male"},"message":"The customer is created."}</pre> <pre>{"customer":{"id":"100000","name":"lee","gender":"female"},"message":"The customer is created."}</pre>
	2-2-B (명령어 입력)	<pre>curl -XGET https://\$domain/v1/customer?id=99999</pre> <pre>curl -XGET https://\$domain/v1/customer?id=100000</pre>
	2-2-B (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	<pre>{"customer":{"id":"99999","name":"kim","gender":"male"},"message":"The customer is well in database."}</pre> <pre>{"customer":{"id":"100000","name":"lee","gender":"female"},"message":"The customer is well in database."}</pre>
2-3	2-3-A (명령어 입력)	<pre>curl -XPOST -H "Content-Type: application/json" -d '{"id": "99999", "name": "kim", "category": "phone"}' https://\$domain/v1/product</pre> <pre>curl -XPOST -H "Content-Type: application/json" -d '{"id": "100000", "name": "lee", "category": "computer"}' https://\$domain/v1/product</pre>
	2-3-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	<pre>{"product":{"id":"99999","name":"kim","category":"phone"},"message":"The product is created."}</pre> <pre>{"product":{"id":"100000","name":"lee","category":"computer"},"message":"The product is created."}</pre>

2-4	2-4-A (명령어 입력)	<pre>curl -XPOST -H "Content-Type: application/json" -d '{"id": "100", "customerid": "99999", "productid": "p1"}' https://\$domain/v1/order</pre> <pre>curl -XPOST -H "Content-Type: application/json" -d '{"id": "101", "customerid": "100000", "productid": "c1"}' https://\$domain/v1/order</pre>
	2-4-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	<pre>{"order":{"id":"100","customerid":"99999","productid":"p1"},"message":"The order is created."}</pre> <pre>{"order":{"id":"101","customerid":"100000","productid":"c1"},"message":"The order is created."}</pre>
	2-4-B (명령어 입력)	<pre>curl -XGET https://\$domain/v1/product?id=99999</pre> <pre>curl -XGET https://\$domain/v1/product?id=100000</pre>
	2-4-B (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	<pre>{"product":{"id":"99999","name":"kim","category":"phone"},"message":"The product is well in database."}</pre> <pre>{"product":{"id":"100000","name":"lee","category":"computer"},"message":"The product is well in database."}</pre>
3-1	3-1-A (명령어 입력)	<pre>load_balancer_arns=\$(aws elbv2 describe-load-balancers --query "LoadBalancers[*].LoadBalancerArn" --output text)</pre> <pre>for arn in \$load_balancer_arns; do</pre> <pre> load_balancer_info=\$(aws elbv2 describe-load-balancers --load-balancer-arns \$arn --query "LoadBalancers[0].{Name:LoadBalancerName, VpcId:VpcId, Scheme:Scheme}" --output json)</pre> <pre> name=\$(echo \$load_balancer_info jq -r '.Name')</pre> <pre> vpc_id=\$(echo \$load_balancer_info jq -r '.VpcId')</pre> <pre> scheme=\$(echo \$load_balancer_info jq -r '.Scheme')</pre> <pre> echo "\$name \$vpc_id \$scheme"</pre> <pre>done</pre>
	3-1-A (예상 출력) <u>부분 일치</u>	<pre>wsc-prod-lb vpc-0f2fbb291c841ac90 internal</pre> <pre>wsc-ingress-lb vpc-0015f52b839355db9 internet-facing</pre> <p>- 빨간 색으로 표시된 부분은 달라도 무관합니다.</p>


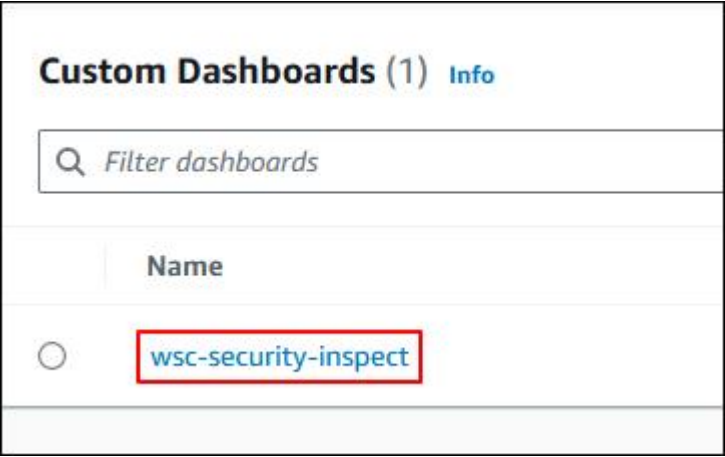
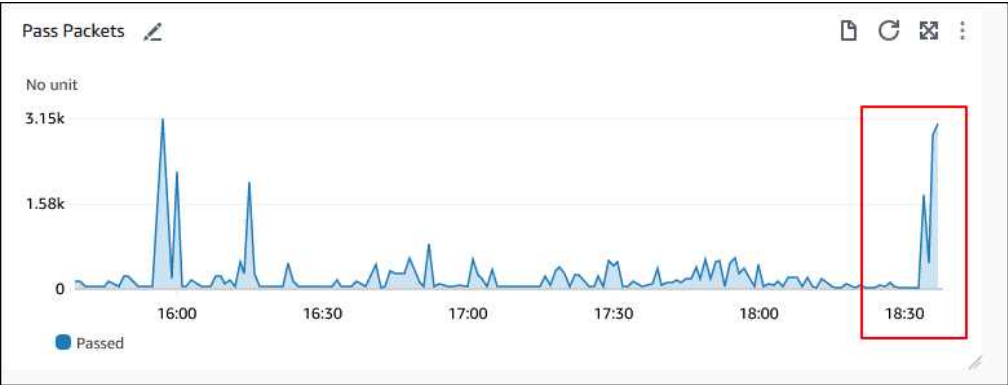
4-1	4-1-A (명령어 입력)	<pre> rds_instance_identifiers=\$(aws rds describe-db-clusters --db-cluster-identifier wsc-prod-db-cluster --query "DBClusters[0].DBClusterMembers[*].DBInstanceIdentifier" --output text) for instance_id in \$rds_instance_identifiers; do instance_info=\$(aws rds describe-db-instances --db-instance-identifier \$instance_id --query "DBInstances[0].{InstanceType:DBInstanceClass, Engine:Engine, EngineVersion:EngineVersion}" --output json) instance_type=\$(echo \$instance_info jq -r '.InstanceType') engine=\$(echo \$instance_info jq -r '.Engine') engine_version=\$(echo \$instance_info jq -r '.EngineVersion') done echo -e "\$instance_type\n\$engine\n\$engine_version" </pre>
	4-1-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	<pre> db.t3.medium aurora-mysql 8.0.mysql_aurora.3.05.2 </pre>
4-2	4-2-A (명령어 입력)	<pre>mysql -u skill -pSkill53## -h \$endpoint -e "use wscdb ; SHOW TABLES"</pre>
	4-1-A (예상 출력) <u>정확히 일치</u> <u>순서 중요</u>	<pre> +-----+ Tables_in_wscdb +-----+ customer product +-----+ </pre>
5-1	5-1-A (명령어 입력)	<pre>aws dynamodb scan --table-name order --attributes-to-get "id" "customerid" "productid"</pre>
	5-1-A (예상 출력) <u>정확히 일치</u>	<pre> [{ "id": { "S": "100" }, "productid": { "S": "p1" }, "customerid": { "S": "99999" } }, { "id": { "S": "101" }, "productid": { "S": "c1" }, "customerid": { "S": "100000" } }] </pre>

5-2	5-2-A (명령어 입력)	<pre>vpc_endpoint_dynamodb=\$(aws ec2 describe-vpc-endpoints --filters "Name=service-name,Values=com.amazonaws.ap-northeast-2.dynamodb" "Name=vpc-id,Values=\$(aws ec2 describe-vpcs --filters "Name=tag:Name,Values=wsc-prod-vpc" --query "Vpcs[0].VpcId" --output text)" --query "VpcEndpoints[0].VpcEndpointId" --output text) if [-z "\$vpc_endpoint_dynamodb"]; then echo "" else echo \$vpc_endpoint_dynamodb fi</pre>
	5-2-A (예상 출력) <u>부분 일치</u>	<pre>vpce-0d9a3fdff50e06abd</pre> <p>- 빨간 색으로 표시된 부분은 달라도 무관합니다.</p>
6-1	6-1-A (명령어 입력)	<pre>aws ecr list-images --repository-name customer --query 'imageIds[*].imageTag' --output text wc -l aws ecr list-images --repository-name order --query 'imageIds[*].imageTag' --output text wc -l aws ecr list-images --repository-name product --query 'imageIds[*].imageTag' --output text wc -l</pre>
	6-1-A (예상 출력) <u>정확히 일치</u>	<pre>1 1 1</pre>
7-1	7-1-A (명령어 입력)	<pre>firewall_info=\$(aws network-firewall describe-firewall --firewall-name wsc-inspect-firewall) echo "\$firewall_info" jq -r '.Firewall.SubnetMappings[] "ID: ₩(.SubnetId)" ' while read -r subnet_info; do subnet_id=\$(echo "\$subnet_info" awk '{print \$2}') subnet_name=\$(aws ec2 describe-subnets --subnet-ids "\$subnet_id" --query 'Subnets[*].Tags[?Key==`Name`].Value' --output text) echo \$subnet_name done</pre>
	7-1-A (예상 출력) <u>정확히 일치</u> <u>순서 상관 X</u>	<pre>wsc-inspect-secure-sn-a wsc-inspect-secure-sn-c</pre>

7-2	7-2-A (명령어 입력)	aws network-firewall describe-firewall-policy --firewall-policy-name wsc-inspect-rules --query 'FirewallPolicyResponse.{FirewallPolicyName: FirewallPolicyName}' --output text arn=\$(aws network-firewall list-rule-groups --query "RuleGroups[?contains(Name, 'wsc-deny')].Arn" --output text) aws network-firewall describe-rule-group --rule-group-arn \$arn --query 'RuleGroup.RulesSource' --output text grep drop wc -l
	7-2-A (예상 출력) <u>정확히 일치</u>	wsc-inspect-rules Suricata
7-3	7-3-A (명령어 입력)	curl --max-time 10 ifconfig.io timeout 5 openssl s_client -connect www.naver.com:443 -tls1 timeout 5 openssl s_client -connect www.naver.com:443 -tls1_1 timeout 2 openssl s_client -connect www.naver.com:443 -tls1_2 grep "SSL handshake has read"
	7-3-A (예상 출력) <u>부분 일치</u> <u>순서 중요</u>	- 빨간색으로 표시된 부분이 순서에 맞게 일치한지 확인 curl: (28) Operation timed out after 10002 milliseconds with 0 bytes received CONNECTED(00000003) CONNECTED(00000003) depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA verify return:1 depth=1 C = US, O = DigiCert Inc, CN = DigiCert TLS Hybrid ECC SHA384 2020 CA1 verify return:1 depth=0 C = KR, ST = Gyeonggi-do, L = Seongnam-si, O = NAVER Corp., CN = *.naver.com verify return:1 SSL handshake has read 3412 bytes and written 303 bytes Terminated

	<p>7-4-A (명령어 입력)</p>	<p>인프라 변경 항목으로 수동으로 채점 진행합니다. (Network Firewall Rule Group의 정책 부분 변경 Drop -> Pass)</p> <pre>cat << EOF > update_rule.json { "RulesSource": { "RulesString": "pass tcp any any -> any any (msg:₩"Allow all traffic₩"; sid:1; rev:1;)" } } EOF</pre> <p>UPDATE_TOKEN=\$(aws network-firewall describe-rule-group --rule-group-name wsc-deny --type STATEFUL --query UpdateToken --output text) aws network-firewall update-rule-group --update-token \$UPDATE_TOKEN --rule-group-name "wsc-deny" --type STATEFUL --rule-group file://update_rule.json</p> <p><약 30초 대기></p> <pre>curl --max-time 10 ifconfig.io timeout 5 openssl s_client -connect www.naver.com:443 -tls1 timeout 5 openssl s_client -connect www.naver.com:443 -tls1_1</pre>
<p>7-4</p>	<p>7-4-A (예상 출력) <u>부분 일치</u> <u>순서 중요</u></p>	<p>- 빨간색으로 표시된 부분이 순서에 맞게 일치한지 확인</p> <p>200</p> <pre>depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA verify return:1 depth=1 C = US, O = DigiCert Inc, CN = DigiCert TLS RSA SHA256 2020 CA1 verify return:1 depth=0 C = KR, ST = Gyeonggi-do, L = Seongnam-si, O = NAVER Corp., CN = *.naver.com verify return:1 009EB7586F7F0000:error:0A00014D:SSL routines:tls_process_key_exchange:legacy sigalg disallowed or unsupported:ssl/statem/statem_clnt.c:2254: SSL handshake has read 3838 bytes and written 133 bytes</pre> <pre>depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA verify return:1 depth=1 C = US, O = DigiCert Inc, CN = DigiCert TLS RSA SHA256 2020 CA1 verify return:1 depth=0 C = KR, ST = Gyeonggi-do, L = Seongnam-si, O = NAVER Corp., CN = *.naver.com verify return:1 00CEF556E47F0000:error:0A00014D:SSL routines:tls_process_key_exchange:legacy sigalg disallowed or unsupported:ssl/statem/statem_clnt.c:2254: SSL handshake has read 3838 bytes and written 133 bytes</pre>

8-1	8-1-A (명령어 입력)	eksctl get cluster --region ap-northeast-2 grep wsc-prod-cluster
	8-1-A (예상 출력) <u>정확히 일치</u>	wsc-prod-cluster ap-northeast-2 True
8-2	8-2-A (명령어 입력)	kubectl get deploy -n wsc-prod
	8-2-A (예상 출력) <u>정확히 일치</u> <u>순서 상관 X</u>	NAME customer-deploy order-deploy product-deploy
8-3	8-3-A (명령어 입력)	for node in \$(kubectl get nodes -o jsonpath='{.items[*].metadata.name}'); do instance_id=\$(aws ec2 describe-instances --filters "Name=private-dns-name,Values=\$node" --query "Reservations[*].Instances[*].InstanceId" --output text) instance_type=\$(aws ec2 describe-instances --instance-ids \$instance_id --query "Reservations[*].Instances[*].InstanceType" --output text) instance_name=\$(aws ec2 describe-instances --instance-ids \$instance_id --query "Reservations[*].Instances[*].Tags[?Key=='Name'].Value" --output text) echo \$instance_type echo \$instance_name done
	8-3-A (예상 출력) <u>정확히 일치</u>	m5.large wsc-prod-nodegroup

<p>9-1</p>	<p>9-1-A (명령어 입력)</p>	<p>AWS Managemnt Console 통해 CloudWatch 서비스 페이지로 이동 https://ap-northeast-2.console.aws.amazon.com/cloudwatch/home?region=ap-northeast-2#</p> <p>1. dashboard 항목 클릭</p>  <p>2. wsc-security-inspect 클릭</p>  <p>3. Bastion 접속 후 아래와 같은 명령어 실행 for i in {1..1000000}; do ping -c 1 1.1.1.1; done</p> <p>4. 약 5분 대기 후 채점 Dashboard 확인</p>
	<p>9-1-A (예상 출력) 정확히 일치</p>	 <p>그림과 같이 Pass Packets 그래프의 지표가 급증하였는지 확인</p>

