2023 충청남도 제58회 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) 채점 내용의 $ 기호는 명령어에 포함되는 것이 아니라 쉘을 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	
제2과제	1	네트워크 구성	2.5		○		○	
	2	Bastion 서버	4.5		○		○	
	3	웹 어플리케이션	3		○		○	
	4	S3	4.5		○		○	
	5	Cloudfront	6		○		○	
	6	Kinesis	7.5		○		○	
	7	Athena	3		○		○	
	8	로그 백업	4.5		○		○	
	9	보안 대응 자동화	4.5		○		○	
합 계			40					

2) 채점방법 및 기준

과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
2과제	1	네트워크 구성	1	VPC, Subnet	1
			2	Routing	1.5
	2	Bastion 서버	1	Bastion configuration	1.5
			2	Bastion SG ingress rule	1.5
			3	Bastion login notification	1.5
	3	웹 어플리케이션	1	wsi-app configuration	1.5
			2	foo/bar perform	1.5
	4	S3	1	S3 Bucket configuration	1.5
			2	S3 Bucket encryption	1.5
			3	S3 server log	1.5
	5	CloudFront	1	Cloudfront to S3	1.5
			2	Cloudfront redirect	1.5
			3	S3 object updated	1.5
			4	image resizing	1.5
	6	Kinesis	1	wsi-log-stream	1.5
			2	wsi-log-firehose	1.5
			3	application access log stored	1.5
			4	application access log transformed	1.5
			5	without /healthcheck	1.5
	7	Athena	1	Athena Partitioning	1.5
			2	Athena Query	1.5
	8	로그 백업	1	log backup interval 1 minute	1.5
			2	log backup ec2 shutdown	1.5
			3	application restart	1.5
	9	보안 대응 자동화	1	콘솔 로그인 알람	1.5
			2	Bastion 사용자 잠금	1.5
			3	Bastion 격리	1.5
	총점				40

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) 아래 파일들을 Bastion 서버의 /root/marking 디렉터리로 복사합니다. - worldskills-europe.png - marking_script.sh 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**)
	export DISTRIBUTION_ID="E16LB217EE4LQN" # cloudfront distribution id export STATIC_BUCKET="wsi-static-qwer" export LOG_BUCKET="wsi-logs-qwer" export CF_DOMAIN=$(aws cloudfront get-distribution --id ${DISTRIBUTION_ID} --query "Distribution.DomainName" | sed s/₩"//g)
	6) 채점을 진행하기 전에 다음 명령어를 수행하여 채점 진행을 위한 사전 작업을 진행합니다. **(채점 스크립트로 진행 시 생략)**
	# set default region of aws cli aws configure set default.region ap-northeast-2 # clear CDN cache (perform CloudFront invalidation) export INVALIDATION_ID=$(aws cloudfront create-invalidation --distribution-id ${DISTRIBUTION_ID} --paths "/*" --query "Invalidation.Id" | sed s/₩"//g) aws cloudfront wait invalidation-completed --distribution-id ${DISTRIBUTION_ID} --id ${INVALIDATION_ID}

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-public-a --query "Subnets[0].CidrBlock"
	1-1-A (예상 출력) **정확히 일치** **순서 중요**	"10.1.0.0/16" "10.1.0.0/24" "10.1.2.0/24"
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-public-rt --query "RouteTables[].Routes[]" | grep "igw-" | wc -l
	1-2-A (예상 출력) **정확히 일치** **순서 중요**	1 1
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** **순서 중요**	"t3.small"

순번	채점 항목	
2-2	2-2-A (명령어 입력)	export SGID=$(aws ec2 describe-security-groups --filter Name=group-name,Values=wsi-bastion-sg --query "SecurityGroups[0].GroupId" | sed s/₩"//g) ₩ ; 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)
	4272 포트만 존재 하는지 확인
		[
		{
		"ToPort": 4272,
		"FromPort": 4272,
		"IpRanges": [
		{
		"CidrIp": "0.0.0.0/0" → 단일 IP만 허용할 수도 있음
		}
		]
		}
		]

2-3	2-3-A (IP 확인)	내 PC의 Public IP를 확인합니다. - 브라우저에서 ifconfig.me 로 접속하여 확인할 수 있습니다.
	2-3-A (SSH 접속)	(Putty, ssh 명령어 등의 툴을 이용) SSH 프로토콜을 통해 wsi-bastion 서버에 ec2-user 사용자로 로그인합니다.
	2-3-A (명령어 입력)	1분 대기 후 아래 명령어 입력. export QUERY_ID=$(aws logs start-query --log-group-name /wsi/security/bastion-ssh --start-time $(date -d '2 minute ago' "+%s"000) --end-time $(date "+%s"000) --query-string 'fields @message' | jq -r '.queryId') ₩ ; sleep 10 ₩ ; aws logs get-query-results --query-id $QUERY_ID --query "results[].value"
	2-2-A (예상 출력) **내용 포함**	8.8.8.8 ec2-user (8.8.8.8 은 나의 IP로 생각하여 채점 진행.)

순번	채점 항목	
3-1	3-1-A (명령어 입력)	aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-app --query "Reservations[0].Instances[0].InstanceType"
	3-1-A (예상 출력)	"t3.small"
3-2	3-2-A (명령어 입력)	export APP_IP=$(aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-app --query "Reservations[0].Instances[0].PrivateIpAddress" \| jq -r .) ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/foo ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/bar ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/healthcheck
	3-2-A (예상 출력) **정확히 일치** **순서 중요**	{"application":"foo"} 200 {"application":"bar"} 200 {"status":"ok."} 200
4-1	4-1-A (명령어 입력)	aws s3 ls | grep -E "wsi-static-|wsi-logs-"
	4-1-A (예상 출력) **밑줄 부분 일치** **날짜, 시간 무관**	2023-10-15 02:21:55 **wsi-static-<영문 4자리>** 2023-10-15 01:43:43 **wsi-logs-<영문 4자리>**

순번	채점 항목	
4-2	4-2-A (명령어 입력) **<영문 4자리> 수정 후 입력**	aws s3api get-bucket-encryption --bucket $STATIC_BUCKET --query ServerSideEncryptionConfiguration.Rules[0].ApplyServerSideEncryptionByDefault.SSEAlgorithm
	4-2-A (예상 출력) **aws:kms 포함문구**	"aws:kms"
4-3	4-3-A (명령어 입력)	rm -rf output/ ₩ ; mkdir -p output/ ₩ ; aws s3 cp --quiet --recursive s3://${LOG_BUCKET}/s3-accesslog/ output/ ₩ ; find output/ -type f -exec cat {} + \| grep "${STATIC_BUCKET}" | grep "GET /${STATIC_BUCKET}" | wc -l
	4-3-A (예상 출력) **1 이상의 수가 출력되면 허용**	101
5-1	5-1-A (명령어 입력)	cat << EOF >> testobject2.txt This is testobject for CDN perform. EOF
	5-1-A (명령어 입력)	aws s3 cp --quiet testobject2.txt s3://${STATIC_BUCKET}/ ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/testobject2.txt | grep -iE "x-cache:|^200$"
	5-1-A (예상 출력) 정확히 일치	x-cache: Miss from cloudfront 200

순번	채점 항목	
5-1	5-1-B (명령어 입력)	sleep 30 ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/testobject2.txt \| grep -iE "x-cache:\|^200$"
	5-1-B (예상 출력) **정확히 일치**	x-cache: Hit from cloudfront 200
5-2	5-2-A (명령어 입력)	cat << EOF >> testobject3.txt This is testobject for CDN perform. EOF
	5-2-A (명령어 입력)	aws s3 cp --quiet testobject3.txt s3://${STATIC_BUCKET}/ ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${CF_DOMAIN}/testobject3.txt | grep -iE "x-cache:|location:|^301$"
	5-2-A (예상 출력) **밑줄 부분** **달라도 허용** **나머지 일치**	Location: https://**d3mduxaweh9m02**.cloudfront.net/testobject3.txt X-Cache: Redirect from cloudfront 301 (Location에 프로토콜 https 확인) (302 응답코드도 정답으로 인정)
5-3	5-3-A (명령어 입력)	cat << EOF >> testobject4.txt This is testobject for CDN perform (v1.0). EOF
	5-3-A (명령어 입력)	aws s3 cp --quiet testobject4.txt s3://${STATIC_BUCKET}/ ₩ ; curl --silent -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/testobject4.txt
	5-3-A (예상 출력) **정확히 일치**	This is testobject for CDN perform (**v1.0**). 200

순번	채점 항목	
5-3	5-3-B (명령어 입력)	cat << EOF > testobject4.txt This is testobject for CDN perform (v2.0). EOF
	5-3-B (명령어 입력)	aws s3 cp --quiet testobject4.txt s3://${STATIC_BUCKET}/ ₩ ; sleep 60 ₩ ; curl --silent -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/testobject4.txt
	5-3-B (예상 출력) **정확히 일치**	This is testobject for CDN perform (**v2.0**). 200
5-4	5-4-A (명령어 입력)	file worldskills-europe.png | grep -Eo "[[:digit:]]+ *x *[[:digit:]]+"
	5-4-A (예상 출력) **정확히 일치**	225 x 225
	5-4-B (명령어 입력)	rm -rf output/ ₩ ; mkdir -p output/ ₩ ; aws s3 cp --quiet worldskills-europe.png s3://${STATIC_BUCKET}/ ₩ ; curl --silent https://${CF_DOMAIN}/worldskills-europe.png -o output/worldskills-europe.png ₩ ; file output/worldskills-europe.png | grep -Eo "[[:digit:]]+ *x *[[:digit:]]+"
	5-4-B (예상 출력) **정확히 일치**	128 x 128
6-1	6-1-A (명령어 입력)	aws kinesis describe-stream --stream-name wsi-log-stream --query "StreamDescription.StreamStatus"
	6-1-A (예상 출력) **정확히 일치**	"ACTIVE"

순번	채점 항목	
6-2	6-2-A (명령어 입력)	aws firehose describe-delivery-stream --delivery-stream-name wsi-log-firehose ₩ --query "DeliveryStreamDescription.DeliveryStreamStatus"
	6-2-A (예상 출력) **정확히 일치**	"ACTIVE"
6-3	6-3-A (명령어 입력)	aws s3 rm --quiet --recursive s3://${LOG_BUCKET}/accesslog/ ₩ ; export APP_IP=$(aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-app --query "Reservations[0].Instances[0].PrivateIpAddress" | jq -r .) ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/foo ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/foo ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/bar ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/bar
	6-3-A (예상 출력) **정확히 일치**	{"application":"foo"} 200 {"application":"foo"} 200 {"application":"bar"} 200 {"application":"bar"} 200
	6-3-B (명령어 입력)	sleep 60 ₩ ; rm -rf output/ ₩ ; mkdir -p output/ ₩ ; aws s3 cp --quiet --recursive s3://${LOG_BUCKET}/accesslog/ output/ ₩ ; find output/ -type f –exec cat {} + | wc -l
	6-3-B (예상 출력) **정확히 일치**	4

순번	채점 항목	
6-4	6-4-A (명령어 입력)	aws s3 rm --quiet --recursive s3://${LOG_BUCKET}/accesslog/ ₩ ; export APP_IP=$(aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-app --query "Reservations[0].Instances[0].PrivateIpAddress" | jq -r .) ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/foo ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/foo ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/bar ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/bar
	6-4-A (예상 출력)	{"application":"foo"} 200 {"application":"foo"} 200 {"application":"bar"} 200 {"application":"bar"} 200
	6-4-B (명령어 입력)	sleep 60 ₩ ; rm -rf output/ ₩ ; mkdir -p output/ ₩ ; aws s3 cp --quiet --recursive s3://${LOG_BUCKET}/accesslog/ output/ ₩ ; parquet-tools show output/
	6-4-B (예상 출력) **조건 모두 만족**	- 아래의 컬럼을 모두 가지고 있어야 함. clientip, year, month, day, hour, minute, second, method, path, protocol, responsecode, processingtime, useragent - 총 4개의 Record(Row)가 출력되어야 함. - 총 4개의 Record(Row) 중 /v1/foo 경로를 가진 Record(Row)가 2개 있어야 함. - 총 4개의 Record(Row) 중 /v1/bar 경로를 가진 Record(Row)가 2개 있어야 함.

순번	채점 항목	
6–5	6-5-A (명령어 입력)	aws s3 rm --quiet --recursive s3://${LOG_BUCKET}/accesslog/ ₩ ; export APP_IP=$(aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-app --query "Reservations[0].Instances[0].PrivateIpAddress" | jq -r .) ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/foo ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/bar ₩ ; curl -X GET --max-time 5 –w "₩n%{http_code}₩n" ₩ http://${APP_IP}:8080/v1/healthcheck
	6-5-A (예상 출력) **정확히 일치**	{"application":"foo"} 200 {"application":"bar"} 200 {"status":"ok."} 200
	6-5-B (명령어 입력)	sleep 60 ₩ ; rm -rf output/ ₩ ; mkdir -p output/ ₩ ; aws s3 cp --quiet --recursive s3://${LOG_BUCKET}/accesslog/ output/ ₩ ; parquet-tools show output/
	6-5-B (예상 출력) **정확히 일치**	- 총 2개의 Record(Row)가 출력되어야 함. - 총 2개의 Record(Row) 중 /v1/foo 경로를 가진 Record(Row)가 1개 있어야 함. - 총 2개의 Record(Row) 중 /v1/bar 경로를 가진 Record(Row)가 1개 있어야 함. - 총 2개의 Record(Row) 중 /healthcheck 경로를 가진 Record(Row)는 없어야 함.
7–1	7-1-A (명령어 입력)	**Athena Query Editor에서** 다음과 같이 명령어를 입력합니다. DESCRIBE accesslog;
	다음 페이지에 계속...	

순번	채점 항목	
7-1	7-1-A (예상 출력) **내용 포함** **data_type 무관** **comment 무관**	# Partition Information # col_name data_type comment year string month string day string
7-2	7-2-A (Query 실행)	aws s3 rm --quiet --recursive s3://${LOG_BUCKET}/accesslog/ ₩ ; export APP_IP=$(aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-app --query "Reservations[0].Instances[0].PrivateIpAddress" | jq -r .) ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/foo ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/foo ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/bar ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/bar
	7-2-A (예상 출력)	{"application":"foo"} 200 {"application":"foo"} 200 {"application":"bar"} 200 {"application":"bar"} 200
	7-2-B (Query 실행)	1분 대기후 아래 쿼리 실행 Athena Query Editor에서 TrafficPatternQuery 쿼리를 실행합니다. 저장되어 있는 쿼리가 없을 경우 오답처리합니다.
	7-2-B (예상 출력) **시간 무관** **path, statuscode** **일치 해야함.**	| year | month | day | hour | minute | path | statuscode | count | | 2023 | 10 | 12 | 16 | 18 | /v1/foo | 200 | 2 | | 2023 | 10 | 12 | 16 | 18 | /v1/bar | 200 | 2 | (count가 2가 아닐시 3번 재시도 가능)

순번	채점 항목	
8-1	8-1-A (명령어 입력)	; aws s3 rm --quiet --recursive s3://${LOG_BUCKET}/accesslog-backup/interval/ ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/usalion1 ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/usalion2
	8-1-A (예상 출력) **404 code 일치**	<body 무관> 404 <body 무관> 404
	8-1-B (명령어 입력)	sleep 60 ₩ ; rm -rf output/ ₩ ; mkdir -p output/ ₩ ; aws s3 cp --recursive s3://${LOG_BUCKET}/accesslog-backup/interval/ output/ ₩ ; find output/ -type f –exec cat {} + | grep “usalion” | wc -l
	8-1-B (예상 출력) **정확히 일치**	2
8-2	8-2-A (명령어 입력) **다른 채점에 영향이 있어 실행 전 선수와 확인**	aws s3 rm --quiet --recursive s3://${LOG_BUCKET}/accesslog-backup/ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" ₩ http://${APP_IP}:8080/v1/shtest5 export APP_INSTANCE_ID=$(aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-app --query "Reservations[0].Instances[0].InstanceId" | jq -r .) aws ec2 reboot-instances --instance-ids $APP_INSTANCE_ID
	8-2-A (명령어 입력)	sleep 60 # wait for backup time rm -rf output/ mkdir -p output/ aws s3 cp --quiet --recursive s3://${LOG_BUCKET}/accesslog-backup/shutdown/ output/ find output/ -type f -exec cat {} + | grep shtest | wc -l

순번	채점 항목	
8-2	8-2-A (예상 출력) **5 이하의 수가 출력되어야 함**	1
8-3	8-3-A (명령어 입력) **8-2에서 재시작 하였으면 reboot 과정 생략 가능**	export APP_INSTANCE_ID=$(aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-app --query "Reservations[0].Instances[0].InstanceId" | jq -r .) aws ec2 reboot-instances --instance-ids $APP_INSTANCE_ID
	8-3-A (명령어 입력) **다른 채점에 영향이 있어 실행 전 선수와 확인**	export APP_IP=$(aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-app --query "Reservations[0].Instances[0].PrivateIpAddress" \| jq -r .) ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/foo ₩ ; curl -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${APP_IP}:8080/v1/bar
	8-3-A (예상 출력) **정확히 일치**	{"application":"foo"} 200 {"application":"bar"} 200
9-1	9-1-A (Alarm상태확인) **Insufficient 상태도 허용**	loginAlarm 이름을 가진 cloudwatch alarm의 상태가 **OK**인 것을 확인합니다.
	9-1-A (로그인 실패)	(consoleuser) IAM 계정으로의 접근을 5회 이상 실패하도록 합니다.
	9-1-A (Alarm상태확인)	loginAlarm 이름을 가진 cloudwatch Alarm의 상태가 **ALARM** 상태인 것을 확인합니다.

순번	채점 항목	
9-2	9-2-A (로그인 실패)	Bastion 서버에 **ec2-user**에 SSH로 접근합니다. 아래 명령어를 사용하여 user01 계정 로그인을 5번 실패 시킵니다. **ec2-user**@localhost$ su user01 (패스워드는 111로 시도합니다.)
	9-2-A (계정 잠금 확인)	정상 패스워드 Pass@@12를 사용하여 user01 로그인시 접근이 막히는지 확인합니다.
	9-2-B (대기)	120초를 대기합니다.
	9-2-B (계정 접근 확인)	다시 user01 로그인을 시도하여 정상적으로 로그인 되는지 확인합니다. $ su user01
9-3	9-3-A (SSH 접근 시도)	ec2-user를 사용하여 Bastion 서버로의 SSH 접근을 연달아 10번 시도합니다. (로그인 성공 실패 상관 없음)
	9-3-A (SG 확인)	(wsi-bastion-sg) Security Group을 확인하여 아무런 규칙이 없음을 확인합니다. (최대 1분을 기다립니다.)