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) 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	4		○		○	
	2	Bastion 서버	4.5		○		○	
	3	관계형 데이터베이스	1.5		○		○	
	4	웹 어플리케이션	3		○		○	
	5	컨테이너 오케스트레이션	4.5		○		○	
	6	로드밸런서	3		○		○	
	7	S3	4.5		○		○	
	8	Cloudfront	6		○		○	
	9	로그	4.5		○		○	
	10	로드 테스트	1.5		○		○	
	11	CI/CD 파이프라인	3		○		○	
합 계			40					

2) 채점방법 및 기준

과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
1과제	1	네트워크 구성	1	VPC, Subnet	1.0
			2	Routing	1.5
			3	VPC Endpoint	1.5
	2	Bastion 서버	1	Bastion configuration	1.5
			2	Bastion SG ingress rule	1.5
			3	Bastion SG ingress rule auto revoke	1.5
	3	관계형 데이터베이스	1	RDS configuration	1.5
	4	웹 어플리케이션	1	Stress perform	1.5
			2	Product performcon0	1.5
	5	컨테이너 오케스트레이션	1	Stress service	1.5
			2	Product service	1.5
			3	Product environment	1.5
	6	로드밸런서	1	ALB rules	1.5
			2	ALB allow from only Cloudfront	1.5
	7	S3	1	S3 Bucket configuration	1.5
			2	S3 Bucket replication	1.5
			3	S3 Bucket encryption	1.5
	8	Cloudfront	1	Cloudfront to ALB	1.5
			2	Cloudfront to S3	1.5
			3	Cloudfront redirect	1.5
			4	Cloudfront origin group	1.5
	9	로그	1	Stress 어플리케이션 로그	1.5
			2	Product 어플리케이션 로그	1.5
			3	without /healthcheck	1.5
	10	로드 테스트	1	Stress 서비스 스케일링 확인	1.5
	11	CI/CD 파이프라인	1	정상 배포 확인	1.5
			2	ECR Scan 동작 확인	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) 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**) export DistributionID="**<Cloudfront_Distribution_ID>**" export AP_BUCKET="ap-wsi-static-**<4words>**" export US_BUCKET="us-wsi-static-**<4words>**" export CF_DOMAIN=$(aws cloudfront get-distribution --id ${DistributionID} --query "Distribution.DomainName" \| sed s/₩"//g) export LB_DOMAIN=$(aws elbv2 describe-load-balancers --names "wsi-alb" --query "LoadBalancers[0].DNSName" | 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 InvalidationID=$(aws cloudfront create-invalidation --distribution-id ${DistributionID} --paths "/*" --query "Invalidation.Id" | sed s/₩"//g) aws cloudfront wait invalidation-completed --distribution-id ${DistributionID} --id ${InvalidationID} # clear S3 bucket objects aws s3 rm --quiet --recursive s3://$AP_BUCKET/ aws s3 rm --quiet --recursive s3://$US_BUCKET/

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.s3", "com.amazonaws.ap-northeast-2.ecr.api", "com.amazonaws.ap-northeast-2.ecr.dkr"]
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	
2-3	2-3-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 authorize-security-group-ingress --group-id $SGID --protocol tcp --port 22 --cidr 0.0.0.0/0 ₩ ; sleep 60 ₩ ; aws ec2 describe-security-groups --filter Name=group-name,Values=wsi-bastion-sg --query "SecurityGroups[0].IpPermissions[].{FromPort:FromPort,ToPort:ToPort,IpRanges:IpRanges}"

	2-3-A
	(예상 출력)
	정확히 일치
	순서 중요
		[
		{
		"ToPort": 4272,
		"FromPort": 4272,
		"IpRanges": [
		{
		"CidrIp": "0.0.0.0/0" → 단일 IP만 허용할 수도 있음
		}
		]
		}
		]

3-1	3-1-A (명령어 입력)	aws rds describe-db-instances --db-instance-identifier wsi-rds-instance --query "DBInstances[0].{Engine:Engine,MultiAZ:MultiAZ,DBInstanceStatus:DBInstanceStatus,DBInstanceClass:DBInstanceClass,StorageEncrypted:StorageEncrypted}"

	3-1-A
	(예상 출력)
	정확히 일치
	순서 무관
		{
		"Engine": "mysql",
		"MultiAZ": true,
		"DBInstanceClass": "db.t3.medium",
		"DBInstanceStatus": "available",
		"StorageEncrypted": true
		}

순번	채점 항목	
4-1	4-1-A (명령어 입력)	curl -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/product?name=p-H8ds73vW4liO
	4-1-A (예상 출력) **정확히 일치** **순서 중요**	{"message":"The product is well in database"} 200
4-2	4-2-A (명령어 입력)	curl -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/stress ₩ ; curl -X POST -H "Content-Type: application/json" -d '{"iterator": 1}' --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/stress
	4-2-A (예상 출력) **정확히 일치** **순서 중요**	{"version":"v1.0"} 200 {"message":"The cpu is loaded"} 201
5-1	5-1-A (명령어 입력)	aws ecs describe-services --cluster wsi-ecs-cluster --services "stress" --query "services[0].capacityProviderStrategy[].capacityProvider"
	5-1-A (예상 출력) **내용 포함** **2개 이상 출력 허용**	["FARGATE"]
	5-1-B (명령어 입력)	export TaskDefinition=$(aws ecs describe-services --cluster wsi-ecs-cluster --services "stress" --query "services[0].deployments[0].taskDefinition" | sed s/₩"//g) ₩ ; aws ecs describe-task-definition --task-definition $TaskDefinition --query "taskDefinition.{memory:memory,cpu:cpu}"

	5-1-B
	(예상 출력)
	정확히 일치
		{
		"cpu": "512",
		"memory": "1024"
		}

순번	채점 항목	
5-2	5-2-A (명령어 입력)	aws ecs describe-services --cluster wsi-ecs-cluster --services "product" --query "services[0].capacityProviderStrategy[].capacityProvider"
	5-2-A (예상 출력) **내용 포함** **2개 이상 출력 허용**	["FARGATE"]
	5-2-B (명령어 입력)	export TaskDefinition=$(aws ecs describe-services --cluster wsi-ecs-cluster --services "product" --query "services[0].deployments[0].taskDefinition" | sed s/₩"//g) ₩ ; aws ecs describe-task-definition --task-definition $TaskDefinition --query "taskDefinition.{memory:memory,cpu:cpu}"

	5-2-B
	(예상 출력)
	정확히 일치
	순서 중요
		{
		"cpu": "512",
		"memory": "1024"
		}

5-3	5-3-A (명령어 입력)	export TaskDefinition=$(aws ecs describe-services --cluster wsi-ecs-cluster --services "product" --query "services[0].deployments[0].taskDefinition" | sed s/₩"//g) ₩ ; aws ecs describe-task-definition --task-definition $TaskDefinition --query "taskDefinition.containerDefinitions[].environment" | grep dbinfo | wc -l ₩ ; aws ecs describe-task-definition --task-definition $TaskDefinition --query "taskDefinition.containerDefinitions[].secrets" | grep dbinfo | wc -l
	5-3-A (예상 출력) **정확히 일치** **순서 중요**	0 1

순번	채점 항목	
6-1	6-1-A (명령어 입력)	export LoadBalancerArn=$(aws elbv2 describe-load-balancers --names "wsi-alb" --query "LoadBalancers[0].LoadBalancerArn" | sed s/₩"//g) ₩ ; aws elbv2 describe-listeners --load-balancer-arn $LoadBalancerArn --query "Listeners[0].DefaultActions"

	6-1-A
	(예상 출력)
	정확히 일치
	순서 무관
		[
		{
		"Type": "fixed-response",
		"Order": 1,
		"FixedResponseConfig": {
		"MessageBody": "Contents Not Found",
		"StatusCode": "404",
		~~"ContentType": "text/plain"~~
		}
		}
		]

	6-1-B (명령어 입력)	export LoadBalancerArn=$(aws elbv2 describe-load-balancers --names "wsi-alb" --query "LoadBalancers[0].LoadBalancerArn" \| sed s/₩"//g) ₩ ; export ListenerArn=$(aws elbv2 describe-listeners --load-balancer-arn $LoadBalancerArn --query "Listeners[0].ListenerArn" | sed s/₩"//g) ₩ ; aws elbv2 describe-rules --listener-arn $ListenerArn --query "Rules[].Conditions[].{Field:Field, Values:Values}"

	6-1-B
	(예상 출력)
	정확히 일치
		[
		{
		"Field": "path-pattern",
		"Values": [
		"/v1/stress"
		]
		},
		{
		"Field": "path-pattern",
		"Values": [
		"/v1/product"
		]
		}
		]

순번	채점 항목	
6-2	6-2-A (명령어 입력)	curl http://${LB_DOMAIN}/v1/stress -w ₩₩n%{http_code}₩₩n --max-time 5
	6-2-A (예상 출력) **정확히 일치**	curl: (28) Connection timed out after 5000 milliseconds 000
	6-2-B (명령어 입력)	curl https://${CF_DOMAIN}/v1/stress -w ₩₩n%{http_code}₩₩n --max-time 5
	6-2-B (예상 출력) **정확히 일치**	{"version":"v1.0"} 200
7-1	7-1-A (명령어 입력)	aws s3 ls | grep -E "ap-wsi-static|us-wsi-static"
	7-1-A (예상 출력) **밑줄 부분 일치** **날짜, 시간 무관**	2023-10-14 06:47:27 **ap-wsi-static-<영문 4자리>** 2023-10-14 06:47:03 **us-wsi-static-<영문 4자리>**
7-2	7-2-A (명령어 입력)	cat << EOF >> testobject-ap.txt This is testobject for marking that uploaded to AP-NORHTEAST-2. EOF cat << EOF >> testobject-us.txt This is testobject for marking that uploaded to US-EAST-1. EOF aws s3 cp --quiet testobject-ap.txt s3://$AP_BUCKET/static/ ₩ ; aws s3 cp --quiet testobject-us.txt s3://$US_BUCKET/static/ ₩ ; sleep 60 ₩ ; aws s3 ls s3://$AP_BUCKET/static/ \| grep 'testobject-us.txt' ₩ ; aws s3 ls s3://$US_BUCKET/static/ | grep 'testobject-ap.txt'
	7-2-A (예상 출력) **밑줄 부분 일치** **날짜, 시간 무관**	2023-07-26 05:41:04 59 **testobject-us.txt** 2023-07-26 06:23:51 128 **testobject-ap.txt**

순번	채점 항목	
7-3	7-3-A (명령어 입력) **<영문 4자리> 수정 후 입력**	aws s3api get-bucket-encryption --bucket $AP_BUCKET --query ServerSideEncryptionConfiguration.Rules[0].ApplyServerSideEncryptionByDefault.SSEAlgorithm ₩ ; aws s3api get-bucket-encryption --bucket $US_BUCKET --query ServerSideEncryptionConfiguration.Rules[0].ApplyServerSideEncryptionByDefault.SSEAlgorithm
	7-3-A (예상 출력) **정확히 일치**	"aws:kms" "aws:kms"
8-1	8-1-A (명령어 입력)	curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/stress \| grep -iE "x-cache:\|^200$"
	8-1-A (예상 출력) **정확히 일치**	x-cache: Miss from cloudfront 200
	8-1-B (명령어 입력)	sleep 30 ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/stress \| grep -iE "x-cache:\|^200$"
	8-1-B (예상 출력) **정확히 일치**	x-cache: Miss from cloudfront 200

순번	채점 항목	
8-2	8-2-A (명령어 입력)	cat << EOF >> testobject-cdn.txt This is testobject for marking that CDN perform. EOF
	8-2-A (명령어 입력)	aws s3 cp --quiet testobject-cdn.txt s3://$AP_BUCKET/static/ ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/static/testobject-cdn.txt | grep -iE "x-cache:|^200$"
	8-2-A (예상 출력)	x-cache: Miss from cloudfront 200
	8-2-B (명령어 입력)	sleep 30 ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/static/testobject-cdn.txt \| grep -iE "x-cache:\|^200$"
	8-2-B (예상 출력) **정확히 일치**	x-cache: Hit from cloudfront 200
8-3	8-3-A (명령어 입력)	cat << EOF >> testobject-cdn2.txt This is testobject for marking that CDN perform. EOF
	8-3-A (명령어 입력)	aws s3 cp --quiet testobject-cdn2.txt s3://$AP_BUCKET/static/ ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" http://${CF_DOMAIN}/static/testobject-cdn2.txt | grep -iE "x-cache:|location:|^301$"
	8-3-A (예상 출력) **밑줄 부분 달라도 허용** **나머지 일치**	Location: https://**d3499uzxrak3w3**.cloudfront.net/static/testobject2.txt X-Cache: Redirect from cloudfront 301

순번	채점 항목	
8-4	8-4-A (명령어 입력)	cat << EOF >> testobject-cdn-us.txt This is testobject for marking that CDN perform in US-EAST-1 bucket. EOF cat << EOF >> testobject-cdn-ap.txt This is testobject for marking that CDN perform in AP-NORTHEAST-2 bucket. EOF
	8-4-A (명령어 입력)	aws s3 cp --quiet testobject-cdn-ap.txt s3://$AP_BUCKET/static/ ₩ ; aws s3 cp --quiet testobject-cdn-us.txt s3://$US_BUCKET/static/
	8-4-A (명령어 입력)	sleep 60 ₩ ; aws s3 rm --quiet s3://$AP_BUCKET/static/testobject-cdn-us.txt ₩ ; aws s3 rm --quiet s3://$US_BUCKET/static/testobject-cdn-ap.txt ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/static/testobject-cdn-ap.txt \| grep -iE "x-cache:\|^200$" ₩ ; curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/static/testobject-cdn-us.txt \| grep -iE "x-cache:\|^200$"
	8-4-A (예상 출력) **정확히 일치**	x-cache: Miss from cloudfront 200 x-cache: Miss from cloudfront 200

순번	채점 항목	
9-1	9-1-A (명령어 입력①)	curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/stress > /dev/null 2>&1 ₩ ; curl --silent -i -X POST --max-time 5 -w "₩n%{http_code}₩n" -H "Content-Type: application/json" -d '{"iterator":1}' https://${CF_DOMAIN}/v1/stress > /dev/null 2>&1
	9-1-A (명령어 입력②)	sleep 60 ₩ ; QUERY_ID=$(aws logs start-query --log-group-name /wsi/webapp/stress --start-time $(date -d '3 minute ago' "+%s") --end-time $(date "+%s") --query-string 'fields @message' | jq -r '.queryId') ₩ ; sleep 5 ₩ ; aws logs get-query-results --query-id $QUERY_ID --query "results[].value" | grep "GET /v1/stress" | wc -l ₩ ; aws logs get-query-results --query-id $QUERY_ID --query "results[].value" | grep "POST /v1/stress" | wc -l
	9-1-A (예상 출력) **1 이상의 수가 출력되면 허용**	1 1
9-2	9-2-A (명령어 입력)	curl --silent -i -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/product > /dev/null 2>&1
	9-2-A (명령어 입력)	sleep 60 ₩ ; QUERY_ID=$(aws logs start-query --log-group-name /wsi/webapp/product --start-time $(date -d '3 minute ago' "+%s") --end-time $(date "+%s") --query-string 'fields @message | limit 20' | jq -r '.queryId') ₩ ; sleep 5 ₩ ; aws logs get-query-results --query-id $QUERY_ID --query "results[].value" | grep "GET /v1/product" | wc -l
	9-2-A (예상 출력) **1 이상의 수가 출력되면 허용**	1

순번	채점 항목	
9-3	9-3-A (명령어 입력)	QUERY_ID=$(aws logs start-query --log-group-name /wsi/webapp/stress --start-time $(date -d '24 hours ago' "+%s") --end-time $(date "+%s") --query-string 'fields @message | limit 20' | jq -r '.queryId') sleep 5 aws logs get-query-results --query-id $QUERY_ID --query "results[].value" | grep "GET /healthcheck" | wc -l
	9-3-A (예상 출력) **정확히 일치**	0
	9-3-B (명령어 입력)	QUERY_ID=$(aws logs start-query --log-group-name /wsi/webapp/product --start-time $(date -d '24 hours ago' "+%s") --end-time $(date "+%s") --query-string 'fields @message | limit 20' | jq -r '.queryId') sleep 5 aws logs get-query-results --query-id $QUERY_ID --query "results[].value" | grep "GET /healthcheck" | wc -l
	9-3-B (예상 출력) **정확히 일치**	0

순번	채점 항목	
10-1	10-1-A (명령어 입력)	aws ecs describe-services --cluster wsi-ecs-cluster --services stress --query "services[0].deployments[0].desiredCount"
	10-1-A (예상 출력) **2 이하의 수가 출력되면 허용** **(0 제외)**	2
	10-1-B (명령어 입력)	for i in $(seq 1000000) do curl --silent -i -X POST --max-time 5 -w "₩n%{http_code}₩n" -H "Content-Type: application/json" -d '{"iterator":99999999999}' https://${CF_DOMAIN}/v1/stress > /dev/null 2>&1 done
	10-1-B (대기)	5분간 대기한 뒤 Ctrl + C를 눌러 명령을 중단합니다.
	10-1-B (명령어 입력)	aws ecs describe-services --cluster wsi-ecs-cluster --services stress —query "services[0].deployments[0].desiredCount"
	10-1-B (예상 출력) **3 이상의 수가 출력되면 허용**	5

순번	채점 항목	
11-1	11-1-A (파일 복사)	채점용 파일을 /home/ec2-user/stress 경로로 복사합니다. - stress-amd64 (stress-v1.1 폴더 하위)
	11-1-A (명령어 입력)	/home/ec2-user/push.sh "deploy new version binary without vulnerability"
	11-1-A (대기)	5분간 대기합니다.
	11-1-A (명령어 입력)	curl --silent -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/stress
	11-1-A (예상 출력)	{"version":"v1.1"} 200
11-2	11-2-A (파일 복사)	11-1 이 동작하지 않으면 11-2도 채점하지 않습니다. 채점용 파일을 /home/ec2-user/stress 경로로 복사합니다. - Dockerfile - stress-amd64 (stress-v1.0 폴더 하위)
	11-2-A (명령어 입력)	cd /home/ec2-user/stress /home/ec2-user/push.sh "deploy new version binary with vulnerability"
	11-2-A (대기)	5분간 대기합니다.
	11-2-A (명령어 입력)	curl --silent -X GET --max-time 5 -w "₩n%{http_code}₩n" https://${CF_DOMAIN}/v1/stress
	11-2-A (예상 출력) **1.1 로 출력 되는지 확인**	{"version":"v1.1"} 200