

2021 대전광역시 제56회 전국기능경기대회 채점기준

1. 채점상의 유의사항	직 종 명	클라우드컴퓨팅
<p>※ 다음 사항을 유의하여 채점하시오.</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) 채점 진행 전 환경 셋업을 위해 다음 사항을 확인해야 합니다. <ul style="list-style-type: none"> - Bastion에 SSH로 접근 가능한지 확인합니다. - Bastion에서 curl, jq, awscli가 설치되었는지 확인합니다. - Bastion에서 IAM Role이 맵핑되어 awscli로 AWS 모든 리소스에 접근 가능한지 확인합니다. - aws sts get-caller-identity 명령을 통해 선수의 계정이 아닌 다른 계정에 접근하고 있는지 확인합니다. 만약, 다른 계정이라면 부정행위를 의심할 수 있습니다. 6) 문제지와 채점지에 있는 <> 는 변수입니다. 해당 부분을 변경해 입력합니다. 7) 채점은 문항 순서대로 진행해야 합니다. 8) 삭제된 내용은 되돌릴 수 없음으로 유의하여 채점을 진행합니다. 9) 이의신청까지 종료된 이후 선수가 생성한 클라우드 리소스를 삭제합니다. 10) 부분 점수가 있는 문항은 채점 항목에 부분 점수가 적혀져 있습니다. 11) 부분 점수가 따로 없는 문항은 전체 다 맞아야 점수로 인정 됩니다. 12) 채점 전 채점환경 구성을 위해 ~/.aws/config 에 아래 내용이 추가 되도록 합니다. <pre> ///// [default] region = ap-northeast-2 output = json ///// </pre>		

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	네트워킹	8.5		○		○	
	2	S3	3.4		○		○	
	3	Bastion	1.6		○		○	
	4	CloudFront	5		○		○	
	5	App	4.5		○		○	
	6	Load Balancer	3.9		○		○	
	7	오토스케일링그룹	5.9		○		○	
	8	로깅	3.7		○		○	
	9	모니터링	3.5		○		○	
합 계			40					

2) 채점방법 및 기준

과제 번호	일련 번호	주요항목	일련 번호	세부항목(채점방법)	배점
제1과제	1	네트워킹	1	VPC	1.5
			2	서브넷	2
			3	HA 구성	2
			4	게이트웨이	3
	2	S3	1	hosting 파일 확인	1
			2	artifact 확인	1
			3	다운로드 권한 확인	1.4
	3	Bastion	1	인스턴스 생성 확인	0.7
			2	Public IP 확인	0.9
	4	Cloud Front	1	Cloud Front 생성 확인	0.9
			2	S3 origin	1.5
			3	LB Origin	1.5
			4	Edge Type 확인	1.1
	5	App	1	Request 확인	1.5
			2	Response 확인	1.5
			3	Default Response 확인	1.5
	6	Load Balancer	1	로드밸런서 생성 확인	0.9
			2	Request 확인	1.5
			3	보안 설정 확인	1.5
	7	오토스케일링그룹	1	오토스케일링 그룹 생성 확인	0.8
			2	HA 확인	1.1
			3	auto scaling 정책 확인	1.1
			4	scale up 확인	1.5
			5	자동화 확인	1.4
	8	로깅	1	로그 그룹 생성 확인	1.1
			2	로그 레코드 확인	1.5
			3	로그 스트림 확인	1.1
	9	모니터링	1	Dash Board 생성 확인	0.7
			2	서버 에러 메트릭 확인	1.5
			3	오토스케일링 그룹 메트릭 확인	1.3
	총점				40

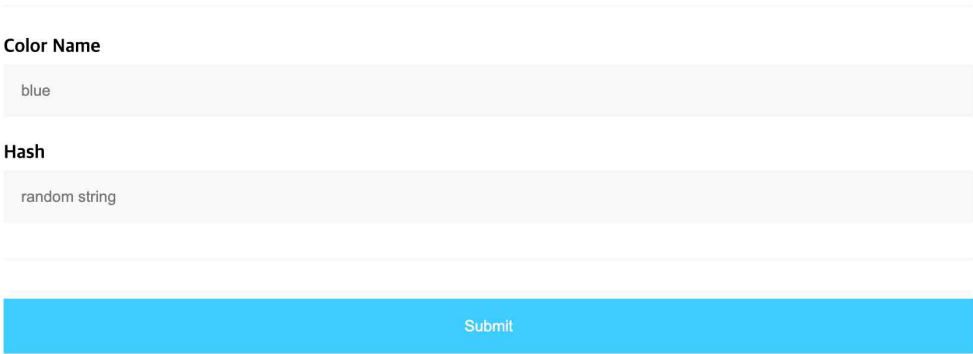
3) 채점 내용

순번	채점 항목
1.1	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어를 입력 합니다.</p> <pre>aws ec2 describe-vpcs --filter Name=tag:Name,Values=ws1-vpc --query "Vpcs[].CidrBlock"</pre> <p>3) 10.1.0.0/16이 출력 되는지 확인 합니다.</p>
1-2	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어를 입력 후 10.1.2.0/24이 출력 되는지 확인 합니다. 출력 될시 1점</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=ws1-public-a --query "Subnets[].CidrBlock"</pre> <p>3) 아래 명령어를 입력 후 10.1.0.0/24이 출력 되는지 확인 합니다. 출력 될시 1점</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=ws1-private-a --query "Subnets[].CidrBlock"</pre>
1-3	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어 입력 후 ap-northeast-2a가 출력 되는지 확인 합니다. 출력 될시 1점</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=ws1-public-a --query "Subnets[].AvailabilityZone"</pre> <p>3) 아래 명령어 입력 후 ap-northeast-2b가 출력 되는지 확인 합니다. 출력 될시 1점</p> <pre>aws ec2 describe-subnets --filter Name=tag:Name,Values=ws1-private-b --query "Subnets[].AvailabilityZone"</pre>

순번	채점 항목
1-4	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어 입력</p> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=ws-private-a-rt ₩ --query "RouteTables[].Routes[].NatGatewayId"</pre> <p>3) "nat-" 로 시작하는 문구가 출력 되는지 확인 합니다. 출력 될시 1점</p> <p>4) 아래 명령어 입력</p> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=ws-private-b-rt ₩ --query "RouteTables[].Routes[].NatGatewayId"</pre> <p>5) "nat-" 로 시작 문구가 출력 되는지 확인하고, 3)의 결과와 다른 ID를 갖는지 확인 합니다. 1점</p> <p>4) 아래 명령어 입력 후 "igw-" 로 시작하는 문구가 있는지 확인 합니다. 출력 시 1점</p> <pre>aws ec2 describe-route-tables --filter Name=tag:Name,Values=ws-public-rt ₩ --query "RouteTables[].Routes[].GatewayId"</pre>
2-1	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) <code>aws s3 ls s3://ws-<비번호>-<4자리 영문>-web-static</code> 을 입력 합니다. (4번 문제 버킷 이름)</p> <p>3) 아래와 같이 index.html 파일의 파일명과 크기가 일치 하는지 확인 합니다.</p> <p>1476 부분이 크기, index.html 부분이 파일 이름 입니다.</p> <pre>2021-06-09 01:52:54 1476 index.html</pre>
2-2	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) <code>aws s3 ls s3://ws-<비번호>-<4자리 임의값>-artifactory</code> 를 입력 합니다. (6번 문제 버킷 이름)</p> <p>3) 아래와 같이 app.py 파일의 파일명과 크기가 일치 하는지 확인 합니다.</p> <p>1043 부분이 크기, app.py 부분이 파일 이름 입니다.</p> <pre>2021-06-09 01:52:54 1043 app.py</pre>

순번	채점 항목
2-3	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어를 입력하여 나오는 IP 하나를 기록 합니다.</p> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-web-api-asg --query 'Reservations[].Instances[].PrivateIpAddress'</pre> <p>3) ssh ec2-user@< 2) 출력 IP> 명령어로 서버에 접근하고 "sudo su" 를 통해 root로 전환 합니다. (SSH 접속시 -i 옵션으로 키를 지정해도 상관 없습니다.)</p> <p>4) mkdir /opt/tmp/ 명령어를 입력 합니다.</p> <p>5) aws s3 cp s3://wsi-<비번호>-<4자리 임의값>-artifactory/app.py /opt/tmp/app-zxzc39 명령어를 입력</p> <p>6) 5)번 명령어 수행시 에러가 화면에 출력되지 않았는지 확인 합니다.</p> <p>7) ls /opt/tmp/app-zxzc39 명령어로 파일이 다운로드 되었는지 확인 합니다.</p> <p>8) echo "hellow cloud" > /opt/app-yzkz-39.txt 명령어를 입력 합니다.</p> <p>9) aws s3 cp /opt/app-yzkz-39.txt s3://wsi-<비번호>-<4자리 임의값>-artifactory/ 명령어를 수행 합니다.</p> <p>10) 9)명령어 수행시 PutObject operation: Access Denied 와 같은 권한 에러메시지가 출력 되는지 확인 합니다.</p> <p>전체 다 맞아야 점수로 인정 되며 부분 점수는 없습니다.</p>
3-1	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어를 입력하여 "i-"로 시작하는 문구를 받아 오는지 확인 합니다.</p> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-bastion-ec2 --query 'Reservations[].Instances[].InstanceId'</pre>
3-2	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어를 입력해 나오는 IP를 기록 합니다.</p> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-bastion-ec2 --query 'Reservations[].Instances[].PublicIpAddress'</pre> <p>3) 아래 명령어 입력 후 출력 되는 IP리스트 중에 2)번에서 출력된 IP가 있는지 확인 합니다.</p> <pre>aws ec2 describe-addresses --query "Addresses[].PublicIp"</pre>
4-1	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어를 입력 합니다.</p> <pre>aws cloudfront list-distributions --query "DistributionList.Items[].Id"</pre> <p>3) 약 14자리의 영문과 숫자가 섞인 ID가 출력 되는지 확인 합니다.</p>

순번	채점 항목
4-2	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어를 입력 합니다.</p> <pre>aws cloudfront list-distributions --query "DistributionList.Items[].Origins.Items[]" # jq ".[].DomainName" grep s3</pre> <p>3) wsi-<비번호>-<4자리 영문>-web-static.s3.amazonaws.com 이 출력 되는지 확인 합니다. (혹은 기타 방법으로 web-static 버킷을 cloud front에 연결 하였는지 확인)</p>
4-3	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어를 입력 합니다.</p> <pre>aws cloudfront list-distributions --query "DistributionList.Items[].Origins.Items[]" # jq ".[].DomainName" grep elb</pre> <p>3) wsi-web-api-alb-<임의숫자>-ap-northeast-2.elb.amazonaws.com 이 출력 되는지 확인 합니다. (혹은 기타 방법으로 api-alb를 cloud front에 연결 하였는지 확인)</p>
4-4	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어를 입력하여 나오는 ID를 기록 합니다.</p> <pre>aws cloudfront list-distributions --query "DistributionList.Items[].Id"</pre> <p>3) 2)번에서 나온 ID로 아래 명령어를 입력 합니다.</p> <pre>aws cloudfront get-distribution-config --id <2번 ID> --query "DistributionConfig.PriceClass"</pre> <p>4) "PriceClass_All" 가 출력 되는지 확인 합니다.</p>

순번	채점 항목
5-1	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령을 통해 CloudFront의 DNS를 기록 합니다.</p> <pre>aws cloudfront list-distributions --query "DistributionList.Items[].DomainName"</pre> <p>3) 크롬 웹브라우저 창을 하나 열고 아래와 같이 2)번에서 나온 주소에 index.html를 입력 합니다.</p> <pre>https://xxxxxxxxx.cloudfront.net/index.html</pre> <p>4) 브라우저에서 아래와 같은 페이지를 받아 오는지 확인 합니다.</p> <p>RGB code</p> 
5-2	<p>1) 5-1 채점 항목의 웹페이지에서 color name에 "red", hash에 "ab12" 를 입력 하고 submit 버튼을 누릅니다.</p> <p>2) 페이지가 이동 되고 {"code":"f34a07","name":"orange"} 라는 문자를 받아 오는지 확인 합니다.</p>
5-3	<p>1) 5-1 채점항목의 페이지인 index.html 로 다시 돌아갑니다.</p> <p>2) color name에 "diamond", hash에 "aaaa"를 입력 하고 submit 버튼을 누릅니다.</p> <p>3) 페이지가 이동 되고 {"code":"ff00ff","name":"pink"} 라는 문자를 받아 오는지 확인 합니다.</p>
6-1	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어를 입력 하여 "internet-facing" 으로 표기 되는지 확인 합니다.</p> <pre>aws elbv2 describe-load-balancers --names "wsi-web-api-alb" --query "LoadBalancers[].Scheme"</pre> <p>3) 아래 명령어를 입력하여 "application" 으로 표기 되는지 확인 합니다.</p> <pre>aws elbv2 describe-load-balancers --names "wsi-web-api-alb" --query "LoadBalancers[].Type"</pre>

순번	채점 항목
6-2	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어로 나오는 LB의 DNS를 기록 합니다.</p> <pre>aws elbv2 describe-load-balancers --names "wsi-web-api-alb" --query "LoadBalancers[].DNSName"</pre> <p>3) 아래 명령어를 입력 하여 {"status":"ok"} 가 표기 되는지 확인 합니다.</p> <pre>curl http://< 2)번에서 나온 DNS>/health</pre>
6-3	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어로 나오는 EC2의 아이피 하나를 기록 합니다.</p> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-web-api-asg ₩ --query "Reservations[].Instances[].PrivateIpAddress"</pre> <p>3) 아래 명령어로 호출 실패를 확인 합니다. 보안 설정 채점임으로 호출이 되지 않아야 성공 입니다.</p> <pre>timeout 10 curl http://<2번에서 나온 IP 중 하나>:8080/health</pre>
7-1	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어를 입력 하여 모두 Healthy 로 표기 되는지 확인 합니다.</p> <pre>aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names wsi-web-api-asg ₩ --query "AutoScalingGroups[].Instances[].HealthStatus"</pre>
7-2	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어 입력하여 "ap-northeast-2a", "ap-northeast-2b" 두개 모두 나오는지 확인 합니다.</p> <pre>aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names wsi-web-api-asg ₩ --query "AutoScalingGroups[].AvailabilityZones"</pre> <p>3) 아래 명령어 입력 하여 2 이상의 숫자를 가지는지 확인 합니다.</p> <pre>aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names wsi-web-api-asg ₩ --query "AutoScalingGroups[].MinSize"</pre>
7-3	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어를 입력해 "wsi-web-api-asg" 라는 값을 받아 오는지 확인 합니다.</p> <pre>aws autoscaling describe-policies --auto-scaling-group-name wsi-web-api-asg ₩ --query "ScalingPolicies[].AutoScalingGroupName"</pre>

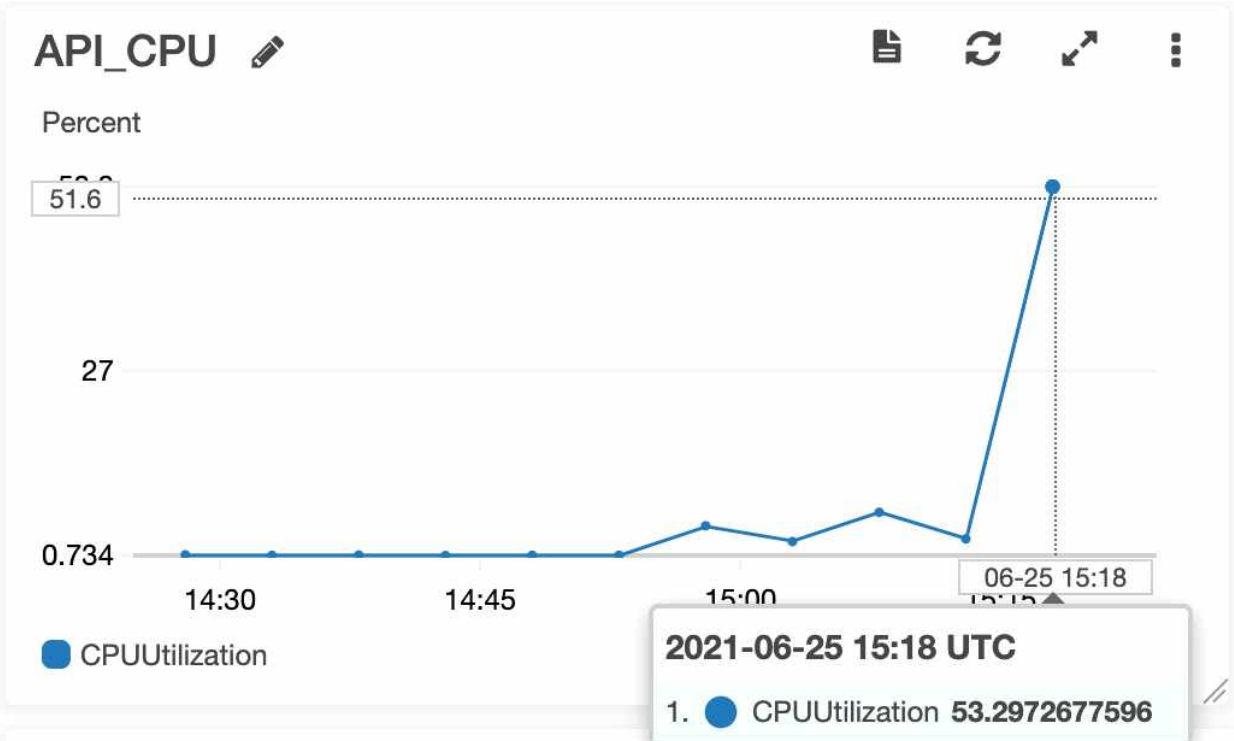
순번	채점 항목
7-4	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어로 현재 인스턴스가 2대가 실행 되도록 변경 합니다.</p> <pre>aws autoscaling update-auto-scaling-group --auto-scaling-group-name wsi-web-api-asg ₩ --min-size 2 --max-size 5 --desired-capacity 2</pre> <p>3) 위의 2) 작업이 끝나 EC2가 2대로 정리되면 아래 명령어로 EC2의 아이피 하나를 기록 합니다. (IP가 1개 이거나, 3개 이상이 나온다면 2개가 나올때 까지 계속 입력합니다.)</p> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-web-api-asg ₩ --query "Reservations[].Instances[].PrivateIpAddress"</pre> <p>4) 아래 명령어로 wsi-web-api-asg EC2에 접근 합니다. -i 옵션을 사용해도 상관 없습니다.</p> <pre>ssh ec2-user@< 2)번에서 나온 IP ></pre> <p>5) 아래 명령어로 stress 패키지를 설치 합니다.</p> <pre>sudo amazon-linux-extras install -y epel; sudo yum install -y stress</pre> <p>6) stress -c 2 명령어로 서버에 부하를 생성 합니다.</p> <p>7) 새로운 터미널을 열어 bastion에 SSH 로 접근 합니다.</p> <p>8) 아래 명령어로 3 이상의 값으로 변하는지 확인 합니다.</p> <p>위의 6)번 명령어로 부하를 발생 시킨 후 7분 이내에 3대 이상으로 변해야 합니다.</p> <pre>aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names wsi-web-api-asg ₩ --query "AutoScalingGroups[].DesiredCapacity"</pre>

순번	채점 항목
7-5	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어로 현재 인스턴스가 모두 종료 되도록 변경 합니다.</p> <pre>aws autoscaling update-auto-scaling-group --auto-scaling-group-name wsi-web-api-asg ₩ --min-size 0 --max-size 0 --desired-capacity 0</pre> <p>3) 아래 명령으로 빈값을 받아와 실행중인 인스턴스가 모두 종료 되었는지 확인 합니다.</p> <pre>aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names wsi-web-api-asg ₩ --query "AutoScalingGroups[].Instances[].InstanceId"</pre> <p>4) 모두 종료되면 아래 명령으로 1대의 인스턴스를 생성 합니다.</p> <pre>aws autoscaling update-auto-scaling-group --auto-scaling-group-name wsi-web-api-asg ₩ --min-size 1 --max-size 1 --desired-capacity 1</pre> <p>5) 아래 명령어로 나오는 LB의 DNS를 기록 합니다.</p> <pre>aws elbv2 describe-load-balancers --names "wsi-web-api-alb" --query "LoadBalancers[].DNSName"</pre> <p>6) 아래 명령어를 입력 하여 {"status":"ok"} 가 표기 되는지 확인 합니다.</p> <p>최대 7분까지 기다리고 이후에도 받아 오지 못하면 틀린것으로 간주 합니다.</p> <pre>curl http://< 5)번에서 나온 DNS >/health</pre>
8-1	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령을 통해 1 이상의 숫자를 받아 오는지 확인 합니다.</p> <pre>aws logs describe-log-groups --log-group-name-prefix /aws/ec2/wsi ₩ --query "logGroups[].storedBytes"</pre>
8-2	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어로 나오는 LB의 DNS를 기록 합니다.</p> <pre>aws elbv2 describe-load-balancers --names "wsi-web-api-alb" --query "LoadBalancers[].DNSName"</pre> <p>3) 아래 명령으로 API를 호출 합니다.</p> <pre>curl http://< 2)번 LB DNS >/v1/color₩?name₩=blue₩&hash₩=999wsi2021abcd</pre> <p>4) 아래 명령으로 로그가 CloudWatch logs로 전송 된지 확인 합니다. 최대 3분 기다릴 수 있습니다.</p> <pre>aws logs tail '/aws/ec2/wsi' --since 5m grep 999wsi2021abcd</pre> <p>5) 위의 4)번 결과가 현재 날짜와 GET /v1/color?name=blue&hash=999wsi2021abcd 이 포함된 문구를 출력 하는지 확인 합니다.</p>

순번	채점 항목
8-3	<p>1) SSH를 통해 Bastion 서버에 접근 합니다.</p> <p>2) 아래 명령어로 인스턴스 ID 하나를 기록 합니다.</p> <pre>aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names wsi-web-api-asg --query "AutoScalingGroups[].Instances[].InstanceId"</pre> <p>3) 아래 명령어를 입력하여 log stream을 받아옵니다.</p> <pre>aws logs describe-log-streams --log-group-name '/aws/ec2/wsi' --query "logStreams[].logStreamName"</pre> <p>4) 위의 3)번에서 출력된 리스트 중 api_ < 2)번의 인스턴스 ID > 로 출력 되는 값이 있는지 확인 합니다. 만약 api_없이 i-xxxx 등 그냥 인스턴스 ID만 존재하면 틀린것으로 간주 합니다.</p>

순번	채점 항목
9-1	<p>1) 크롬 브라우저를 통해 AWS 콘솔에 접속해 CloudWatch 페이지로 이동 합니다.</p> <p>2) 페이지 왼쪽 상단에 Dashboards로 이동 합니다.</p>  <p>3) WSI_APP dash board를 클릭 합니다.</p>  <p>4) 아래와 같이 dashboard가 4개의 그래프를 가지고 있는지 확인 합니다. HTTP_ERROR 그래프 = Target_5xx_Count, ELB_5XX_Count 범례를 가짐. HTTP_COUNT 그래프 = RequestCount 범례를 가짐. API_CPU = CPUUtilization 범례를 가짐. RESPONSE_TIME = TargetResponseTime을 가짐.</p> 

순번	채점 항목
9-2	<p>1) Bastion 서버에 SSH로 접근 합니다.</p> <p>2) 아래 명령어 입력 후 로드밸런서 DNS를 기록 합니다.</p> <pre>aws elbv2 describe-load-balancers --names "wsi-web-api-alb" --query "LoadBalancers[].DNSName"</pre> <p>3) 아래 명령어를 입력하여 서버에 500 에러를 계속 생성 합니다.</p> <pre>while true; do curl --silent http://< 3)번의 LB DNS>/v1/color > /dev/null; done</pre> <p>4) 9-1번 채점 항목을 참고하여 CloudWatch의 WSI_APP dashboard로 이동 합니다.</p> <p>5) 아래와 같이 HTTP_ERROR 그래프의 제일 최신 메트릭에 마우스를 올려 값을 확인 합니다.</p> <p>HTTPCode_Target_5XX_Count가 200 이상 이어야 하며, HTTPCode_ELB_5XX_Count의 값은 최대 50 미만으로 거의 없는 수준으로 유지 되어야 합니다.</p> <p>HTTP_ERROR </p> <p>Count</p> <p>1.75k 1.60k</p> <p>876</p> <p>2</p> <p>14:15 14:30 14:45 15:00 06-25 15:08</p> <p>● HTTPCode_Target_5XX_Count ● HTTPCode_ELB_5XX_Count</p> <p>CPUUtilization</p> <p>2021-06-25 15:08 UTC</p> <p>1. ● HTTPCode_Target_5XX_Count 1,711</p> <p>2. ○ HTTPCode_ELB_5XX_Count -</p>

순번	채점 항목
9-3	<p>1) Bastion 서버에 SSH로 접근 합니다.</p> <p>2) 아래 명령을 통해 한대의 서버만 동작 하도록 변경 합니다.</p> <pre>aws autoscaling update-auto-scaling-group --auto-scaling-group-name wsi-web-api-asg --min-size 1 --max-size 1 --desired-capacity 1</pre> <p>3) 아래 명령어 입력 후 하나 남은 EC2의 IP를 기록 합니다.</p> <pre>aws ec2 describe-instances --filter Name=tag:Name,Values=wsi-web-api-asg --query "Reservations[].Instances[].PrivateIpAddress"</pre> <p>4) ssh ec2-user@< 2)번의 EC2 IP> 를 이용해 서버에 접근 합니다. 필요시 key를 이용해 접근</p> <p>5) 아래 명령어로 stress 패키지를 설치 합니다.</p> <pre>sudo amazon-linux-extras install -y epel; sudo yum install -y stress</pre> <p>6) stress -c 2 명령어로 서버에 부하를 줍니다.</p> <p>7) 9-1번 채점 항목을 참고하여 CloudWatch의 WSI_APP dashboard로 이동 합니다.</p> <p>8) stress 명령어 입력 후 5분 이내에 CPU 사용량이 급격히 올라 약 30% 이상 올라 가는지 확인 합니다.</p>
	 <p>The screenshot shows the AWS CloudWatch console for the 'API_CPU' dashboard. The graph displays 'CPUUtilization' as a blue line. The y-axis is labeled 'Percent' with values 0.734, 27, and 51.6. The x-axis shows time from 14:30 to 15:00. A data point at 15:18 shows a sharp spike in CPU utilization to 53.2972677596%.</p>