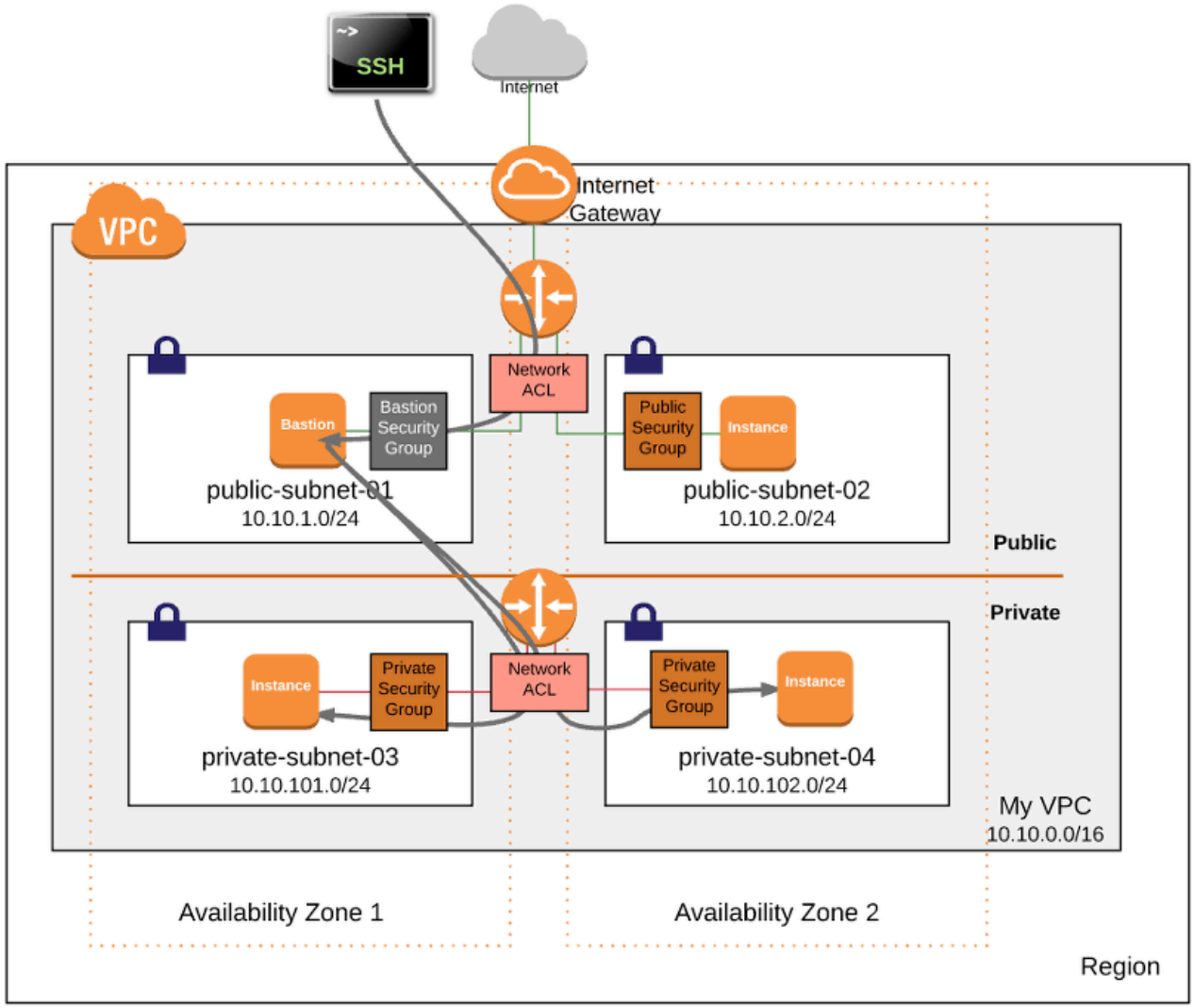
AWS IaaS 과정

AWS VNet 생성 및 VM 생성



// VPC 생성
$ aws ec2 create-vpc --cidr-block 10.0.0.0/16

// VPC 생성 후에 DNS 도 생성시킨다
$ aws ec2 modify-vpc-attribute --vpc-id vpc-0d24454ed82547b28 --enable-dns-hostnames

// VPC 들 정보보기
$ aws ec2 describe-vpcs

// Subnet 생성
$ aws ec2 create-subnet --vpc-id vpc-0d30b814b29ba88f7 --availability-zone us-east-1a --cidr-block 10.0.0.0/24

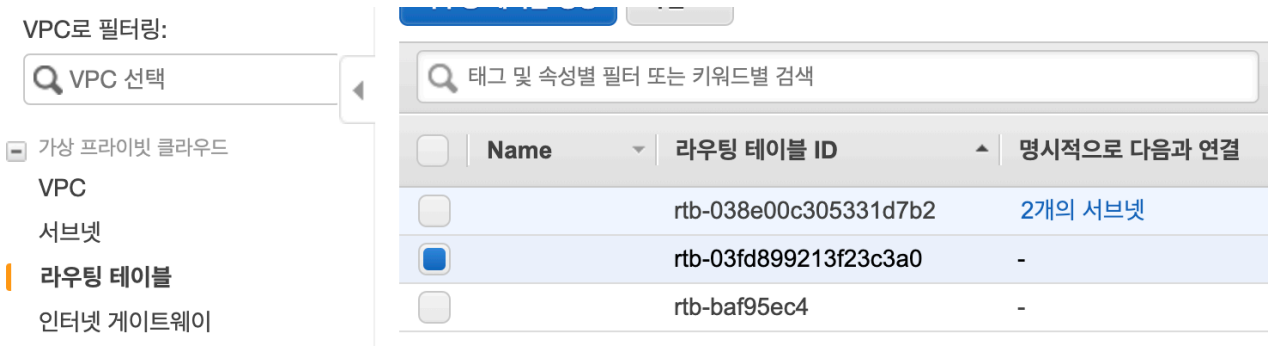
$ aws ec2 create-subnet --vpc-id vpc-0d30b814b29ba88f7 --availability-zone us-east-1b --cidr-block 10.0.1.0/24

// 서브넷 정보 보기
$ aws ec2 describe-subnets

// VPC 생성시 라우트 테이블도 생성됨. -> 메인 라우트 테이블임.
// Subnet 을 생성하면 내부에 라우트 테이블이 이미 존재하는 것이 메인 라우트 테이블임.
// 별도의 라우트 테이블 생성하여 서브넷을 등록하는 것이 좋다 : 라우팅 정보는, 인터넷 게이트웨이를 만든 후 그 정보를 추가하면 됨.

$ aws ec2 create-route-table --vpc-id vpc-0d24454ed82547b28

// 서브넷 등록
$ aws ec2 associate-route-table --route-table-id rtb-038e00c305331d7b2 --subnet-id subnet-01c83de811be9a745



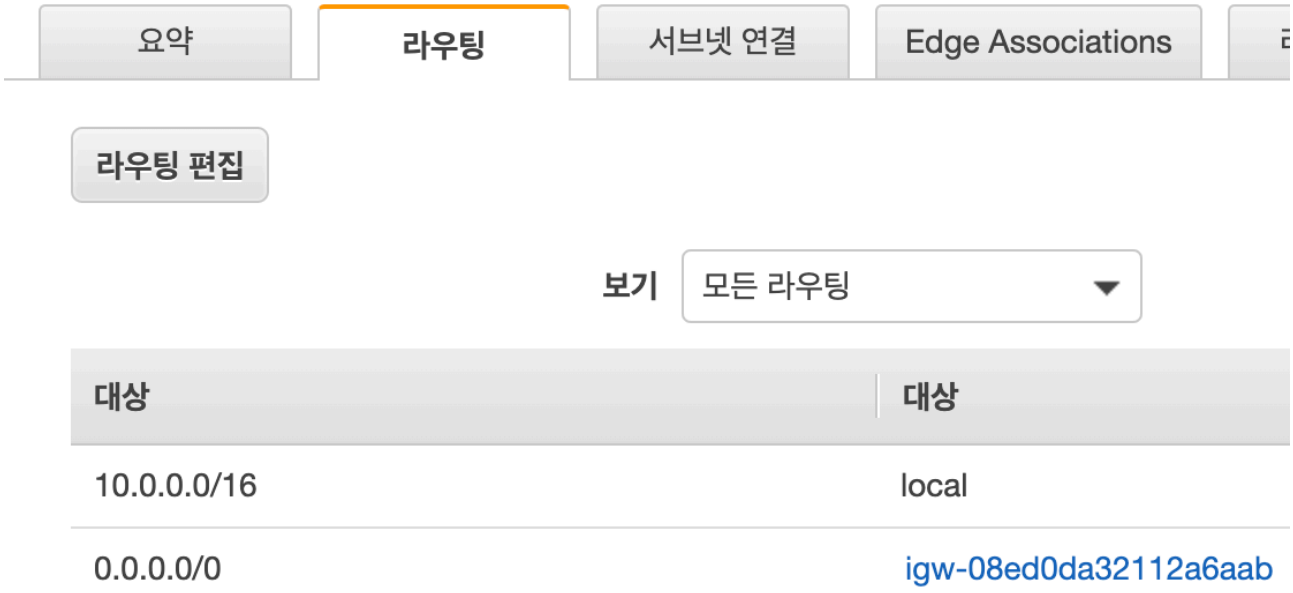
// Internet Gateway 를 생성하고, VPC 에 연결. 1개의 인터넷게이트웨이는 1개의 VPC에 대응! 즉 1:1로만 가능.
// 리전당 최대 VPC는 5개 이므로 인터넷 게이트 웨이 또한 리전당 5개.

$ aws ec2 create-internet-gateway

$ aws ec2 attach-internet-gateway --internet-gateway-id igw-08ed0da32112a6aab --vpc-id vpc-0d24454ed82547b28

// Internet Gateway를 라우팅 대상으로 지정
$ aws ec2 create-route --route-table-id rtb-038e00c305331d7b2 --destination-cidr-block 0.0.0.0/0 --gateway-id igw-08ed0da32112a6aab

// 라우팅 테이블에서 확인



// EC2 에 사용할 KeyPair 생성 : windows 에서는 'KeyMaterial' 아니고 "KeyMaterial"
$ aws ec2 create-key-pair --key-name minkey --query 'KeyMaterial' --output text > mingyu_keypair.pem

// RSA PRIVATE KEY 가 생성되어 있음.
$ cat mingyu_keypair.pem

// Security Group 생성 : description 필수
$ aws ec2 create-security-group --group-name sg01 --description "for my ec2" --vpc-id vpc-0d24454ed82547b28

// 현재 나의 IP 주소 체크
$ curl http://checkip.amazonaws.com
1.235.44.150

// 현재 나의 IP 주소를 시큐리티 그룹에 대입하여 22번 포트로 접속할 수 있도록 설정한다.
$ aws ec2 authorize-security-group-ingress --group-id sg-0c06b9658f88ac112 --protocol tcp --port 22 --cidr 1.235.44.150/32

// 설정 확인
$ aws ec2 describe-security-groups --group-ids sg-0c06b9658f88ac112

// Subnet 1 에 EC2 생성
$ aws ec2 run-instances --image-id ami-085925f297f89fce1 --count 1 --instance-type t2.micro --key-name minkey --security-group-ids sg-0c06b9658f88ac112 --subnet-id subnet-01c83de811be9a745

// Subnet 2 에 EC2 생성

$ aws ec2 run-instances --image-id ami-085925f297f89fce1 --count 1 --instance-type t2.micro --key-name minkey --security-group-ids sg-0c06b9658f88ac112 --subnet-id subnet-0d907438a94b0df9c

---

// Elastic IP 생성
$ aws ec2 allocate-address

// EC2 Instance ID 보기
$ aws ec2 describe-instances | grep InstanceId

// 출력
"InstanceId": "i-08ce1e4d3dd7d153d",
"InstanceId": "i-0bf3e4907b0db54ae",

// EC2 에 EIP 연결
$ aws ec2 associate-address --instance-id i-08ce1e4d3dd7d153d --allocation-id eipalloc-0835a5a038cd1cc05

// EC2 에 할당된 주소 확인
$ aws ec2 describe-addresses

// EIP 할당 취소
$ aws ec2 disassociate-address --association-id [연결을 해제할 association ID]

// EC2 에 접속하기 위한 준비
$ chmod 0600 mingyu_keypair.pem

// DNS 가 있다면 아래와 같이 쿼리가 가능
$ aws ec2 describe-instances --instance-id i-08ce1e4d3dd7d153d | grep PublicDnsName

// 출력
"PublicDnsName": "ec2-54-164-99-151.compute-1.amazonaws.com"

// HTTP Service 를 위해서 Security Group 에 80 포트를 열어준다.
$ aws ec2 authorize-security-group-ingress --group-id sg-0c06b9658f88ac112 --protocol tcp --port 80 --cidr 0.0.0.0/0

// 접속
$ ssh -i mingyu_keypair.pem ubuntu@ec2-54-164-99-151.compute-1.amazonaws.com
$ ssh -i mingyu_keypair.pem ubuntu@3.233.20.194

// Node.js 설치
$ sudo curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -

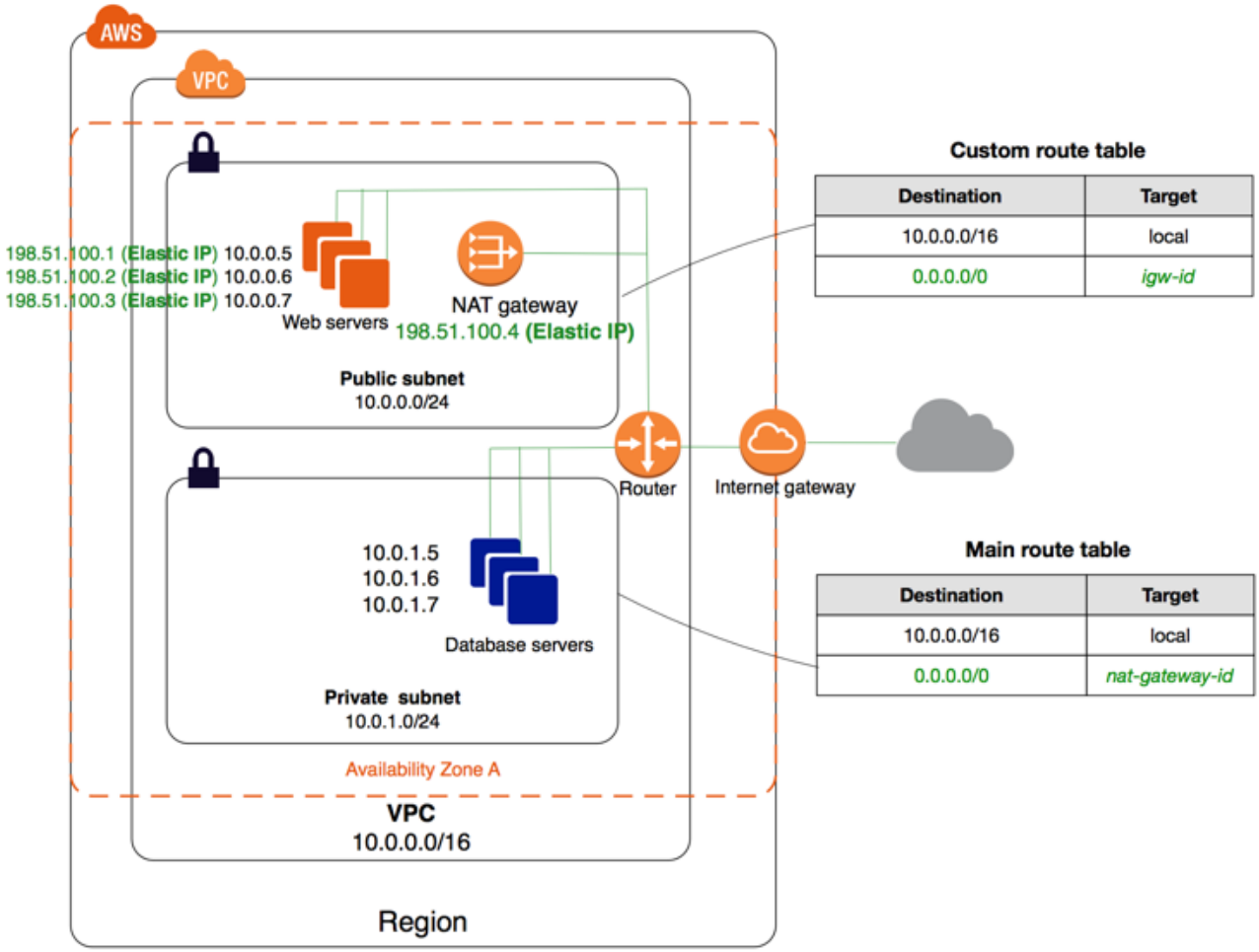
$ sudo apt install -y nodejs

// NginX 가 설치되어 있지 않아서 3000 포트로 연결되어 있지 않음, VM 에서 아래 부분을 수정하여 Nginx 를 Reverse Proxy로 설정한다.

$ sudo apt install -y nginx
$ sudo apt update
$ sudo nano /etc/nginx/sites-available/default

try_files $uri $uri/ =404; 주석 처리 후
proxy_pass http://127.0.0.1:3000/;

$ sudo service nginx restart



// ELB 생성 : 콤마 뒤에 띄워 쓰기 금지!!!
// 내부 로드밸런스 생성시에는 --scheme internal 추가

$ aws elb create-load-balancer --load-balancer-name lb01 --listeners Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80 --tag Key=NAME,Value=MINGYULB --subnet subnet-01c83de811be9a745 subnet-0d907438a94b0df9c --security-groups sg-0c06b9658f88ac112

// 출력 : "DNSName": "lb01-36252487.us-east-1.elb.amazonaws.com"

// BackEndPool 인 인스턴스들을 연결, ID 사이는 공백
$ aws elb register-instances-with-load-balancer --load-balancer-name lb01 --instances i-08ce1e4d3dd7d153d i-0bf3e4907b0db54ae

// ELB 로 HTTP 접속 시도 : http://lb01-36252487.us-east-1.elb.amazonaws.com/

// EIP 분리 및 제거 : association id -> 연결 ID, allocation id -> 할당 ID

$ aws ec2 disassociate-address --association-id eipassoc-02a0680a59903bd8c

$ aws ec2 release-address --allocation-id eipalloc-0835a5a038cd1cc05

$ aws elb create-load-balancer --load-balancer-name lb02 --listeners Protocol=TCP,LoadBalancerPort=27017,InstanceProtocol=TCP,InstancePort=27017 --tag Key=NAME,Value=Mongo --subnet subnet-0a0be1bc106e24c40 subnet-07a22c48b069dba6a --security-groups sg-01469b8f07af86b87 --scheme internal

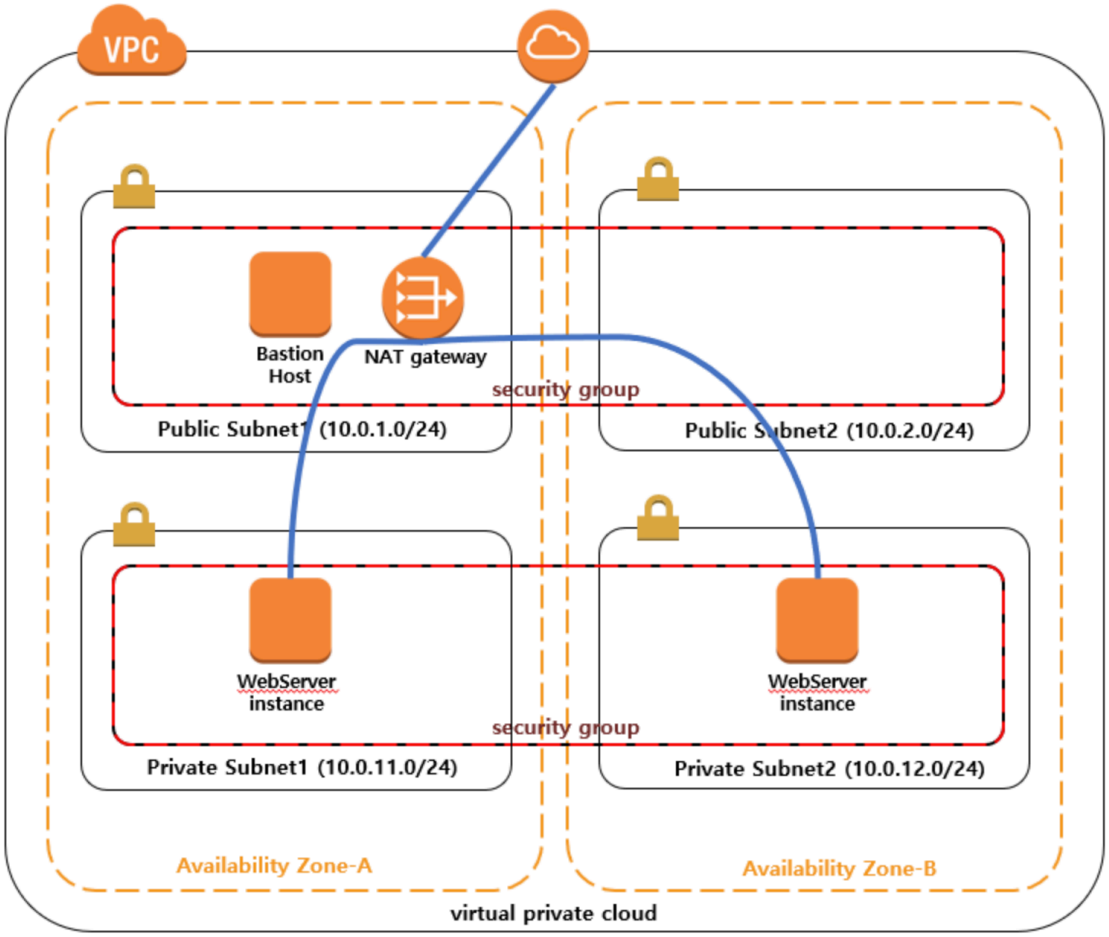
// 80 및 443
aws elb create-load-balancer --load-balancer-name my-load-balancer --listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80" "Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80, SSLCertificateId=arn:aws:iam::123456789012:server-certificate/my-server-cert" --availability-zones us-west-2a us-west-2b

---

AWS Bastion Host 설정

// Public Subnet 즉, 라우트 테이블에 인터넷 게이트웨이가 존재하는 서브넷에는 EC2 를 두지 않는다. NAT Gateway 만 위치 시킨다. 그리고 NAT Gateway 는 Private Subnet 들의 라우트 테이블에 넣는다.

// 외부 로드밸런스에 프라이빗 서브넷 (EC2 가 포함된)및 퍼블릭 서브넷(인터넷게이트웨이가 포함된)을 연결한다. 반드시 Public Subnet 을 포함해야 외부에서 접속이 가능하다.



*NAT Instance와 NAT Gateway의 차이

- 중요한 차이점은 NAT Gateway에서는 포트포워딩을 지원하지 않는다 포트포워딩을 원하는 경우 NAT Instance를 사용해야한다.

// EIP 생성 후 NAT Gateway 생성.

$ aws ec2 allocate-address

$ aws ec2 create-nat-gateway --subnet-id subnet-09e667af6c4651a99 --allocation-id eipalloc-0986a15064da4a466

// NAT Gateway 생성 후 Private Subnet 의 라우팅 테이블에 포함.

Auto Scaling 환경 설정

// AutoScaling 을 위한 AMI 이미지 생성
$ aws ec2 create-image --instance-id i-03c302c1b0ab8d54d --name nginxImg --description "for nginx"

// Amazon Elastic Block Store(EBS)는 대규모로 처리량과 트랜잭션 집약적인 워크로드 모두를 지원하기 위해 Amazon Elastic Compute Cloud(EC2)에서 사용하도록 설계된 사용하기 쉬운 고성능 블록 스토리지 서비스

// --no-associate-public-ip-address : Public IP 의 비활성화

$ aws autoscaling create-launch-configuration --launch-configuration-name **autocon** --image-id ami-09d69d1aa6a6a5ced --key-name minkey **--no-ebs-optimized** --instance-type t2.micro --instance-monitoring Enabled=true --security-groups sg-0c06b9658f88ac112 **--associate-public-ip-address**

// Auto Scaling 그룹 생성 - 갯수 정하고, 헬스체크 시간 간격, 가용성존 설정, 서브넷 설정, 쿨다운 시간 설정

$ aws autoscaling create-auto-scaling-group --auto-scaling-group-name **autoGR** --launch-configuration-name **autocon --desired-capacity 2 --min-size 2 --max-size 3** --health-check-type EC2 --health-check-grace-period 300 --termination-policies "Default" --availability-zones us-east-1a us-east-1c --vpc-zone-identifier "subnet-01c83de811be9a745,subnet-0d907438a94b0df9c" --default-cooldown 300

// [PercentChangeInCapacity, ChangeInCapacity, ExactCapacity]

ChangeInCapacity — 그룹의 현재 용량을 지정된 값만큼 늘리거나 줄입니다. 양의 값은 용량을 늘리고, 음의 조절 값은 용량을 줄입니다. 예제: 그룹의 현재 용량이 3개의 인스턴스이고 조절이 5개인 경우 이 정책이 수행되면 용량 단위가 총 8개가 되도록 용량에 5개의 용량 단위를 추가합니다.

ExactCapacity — 그룹의 현재 용량을 지정된 값으로 변경합니다. 이 조절 유형에는 양의 값을 지정합니다. 예제: 그룹의 현재 용량이 3개의 인스턴스이고 조절이 5개인 경우 이 정책이 수행되면 용량을 5개의 용량 단위로 변경합니다.

PercentChangeInCapacity — 그룹의 현재 용량을 지정된 퍼센트만큼 늘리거나 줄입니다. 양의 값은 용량을 늘리고, 음의 값은 용량을 줄입니다. 예제: 현재 용량이 10개이고 조절이 10%인 경우 이 정책이 수행되면 용량 단위가 총 11개가 되도록 용량에 1개의 용량 단위를 추가합니다.

// 오토스케일링 정책 추가

// EC2 감소 정책
$ aws autoscaling put-scaling-policy --auto-scaling-group-name autoGR --policy-name policy01 --adjustment-type ChangeInCapacity --scaling-adjustment -1 --cooldown 180

// Cloudwatch 를 통해 위의 정책이 실행되게 한다.

$ aws cloudwatch put-metric-alarm --alarm-name **alarm01** --metric-name **CPUUtilization** --namespace AWS/EC2 --statistic Average --period 300 --threshold **20** --comparison-operator LessThanOrEqualToThreshold --dimensions Name=AutoScalingGroupName,Value=autoGR --evaluation-periods 1 --alarm-actions arn:aws:autoscaling:us-east-1:305163539117:scalingPolicy:eed3cfb4-7683-40ac-bf16-8a708b7e3b5c:autoScalingGroupName/autoGR:policyName/policy01

// EC2 추가 정책
$ aws autoscaling put-scaling-policy --auto-scaling-group-name autoGR --policy-name policy02 --adjustment-type ChangeInCapacity --scaling-adjustment 1 --cooldown 180

// Cloudwatch 를 통해 위의 정책이 실행되게 한다.
$ aws cloudwatch put-metric-alarm --alarm-name alarm02 --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average --period 300 --threshold 70 --comparison-operator GreaterThanThreshold --dimensions Name=AutoScalingGroupName,Value=autoGR --evaluation-periods 1 --alarm-actions arn:aws:autoscaling:us-east-1:305163539117:scalingPolicy:72759ae8-e4f6-4c57-8ecf-10ff7fcc0db4:autoScalingGroupName/autoGR:policyName/policy02

// 정책 확인
$ aws autoscaling describe-policies

// VM 접속 후 부하 걸기
$ sudo apt install stress
$ sudo stress --cpu 10 --timeout 300 &

자주 사용하게 되는 AWS 리전(Region)

	리전 이름(Region Name)	리전(Region)
1	미국 동부(오하이오)	us-east-2
2	미국 동부(버지니아 북부)	us-east-1
3	미국 서부(캘리포니아 북부)	us-west-1
4	미국 서부(오레곤)	us-west-2
5	아시아 태평양(뭄바이)	ap-south-1
6	아시아 태평양(오사카-로컬)	ap-northeast-3
7	아시아 태평양(서울)	ap-northeast-2
8	아시아 태평양(싱가포르)	ap-southeast-1
9	아시아 태평양(시드니)	ap-southeast-2
10	아시아 태평양(도쿄)	ap-northeast-1
11	캐나다(중부)	ca-central-1
12	중국(베이징)	cn-north-1
13	중국(닝샤)	cn-northwest-1
14	EU(프랑크푸르트)	eu-central-1
15	EU(아일랜드)	eu-west-1
16	EU(런던)	eu-west-2
17	EU(파리)	eu-west-3
18	EU(스톡홀름)	eu-north-1
19	남아메리카(상파울루)	sa-east-1
20	AWS GovCloud (미국 동부)	us-gov-east-1
21	AWS GovCloud (US)	us-gov-west-1