AWS 인프라 구축

개요

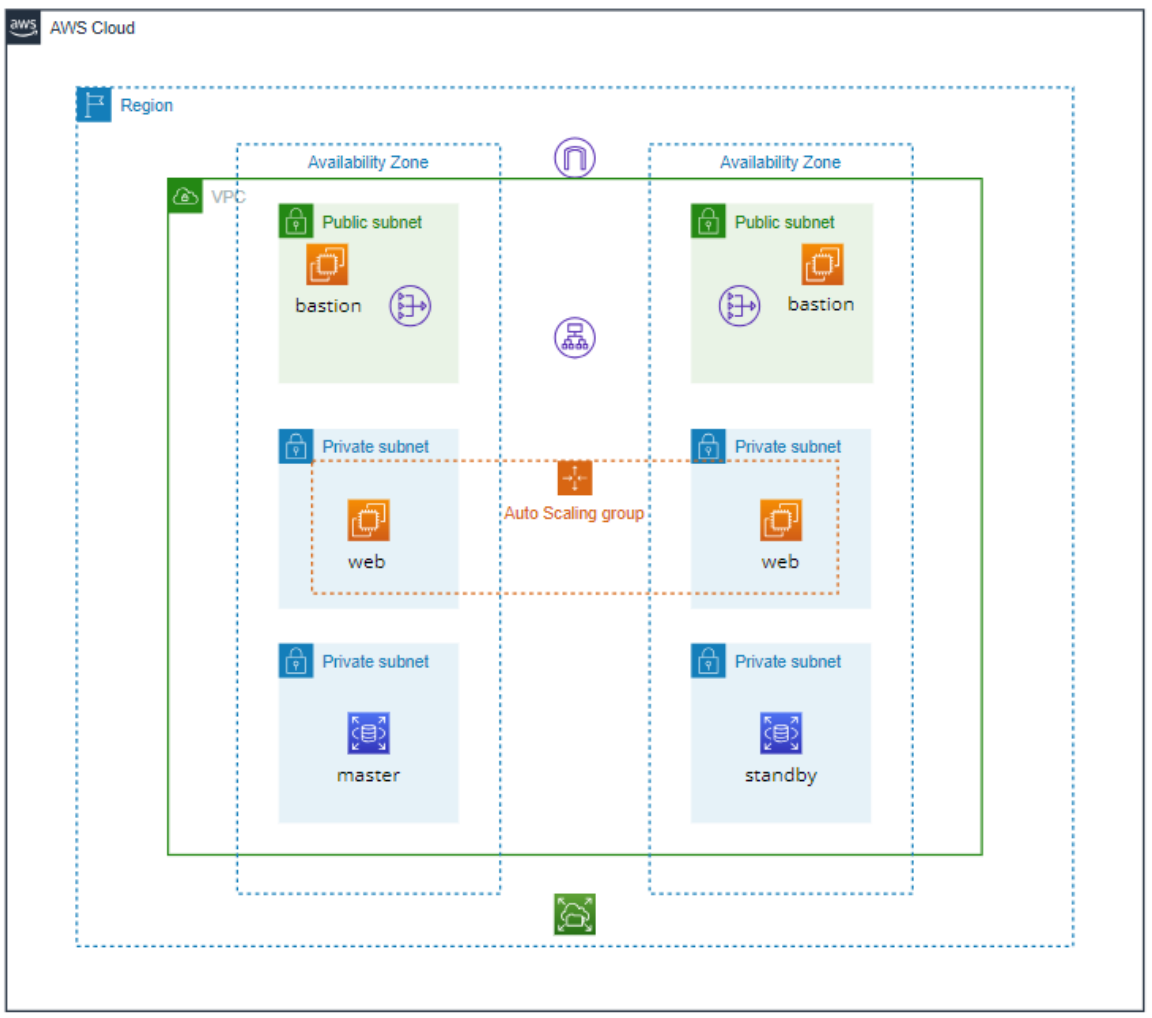
AWS의 주요 서비스에 대해서 학습하고 직접 인프라를 구성해보는 프로젝트이다.

요구조건은 다음과 같다.

1. bastion host를 통해서만 웹서버 인스턴스들에 ssh로 접속 가능하게 한다.
2. private subnet에 있는 웹서버 인스턴스로 로드밸런싱
3. 웹서버 인스턴스는 인터넷 사용가능(인터넷에서 웹서버에 접속 시도는 불가)
4. 웹서버 인스턴스는 오토스케일링으로 가용성을 유지한다.
5. rds의 경우, master에 문제가 생길 시 failover(standby → active)가 가능해야 한다.
6. 웹서버 인스턴스들은 하나의 EFS에 mount되어야 한다.
7. 최소한의 네트워크 트래픽만 허용

아키텍처

전체 아키텍처는 다음과 같다.



subnet들의 IP CIDR는 아래와 같다.

10.16.1.0/24	10.16.2.0/24
10.16.3.0/24	10.16.4.0/24
10.16.5.0/24	10.16.6.0/24

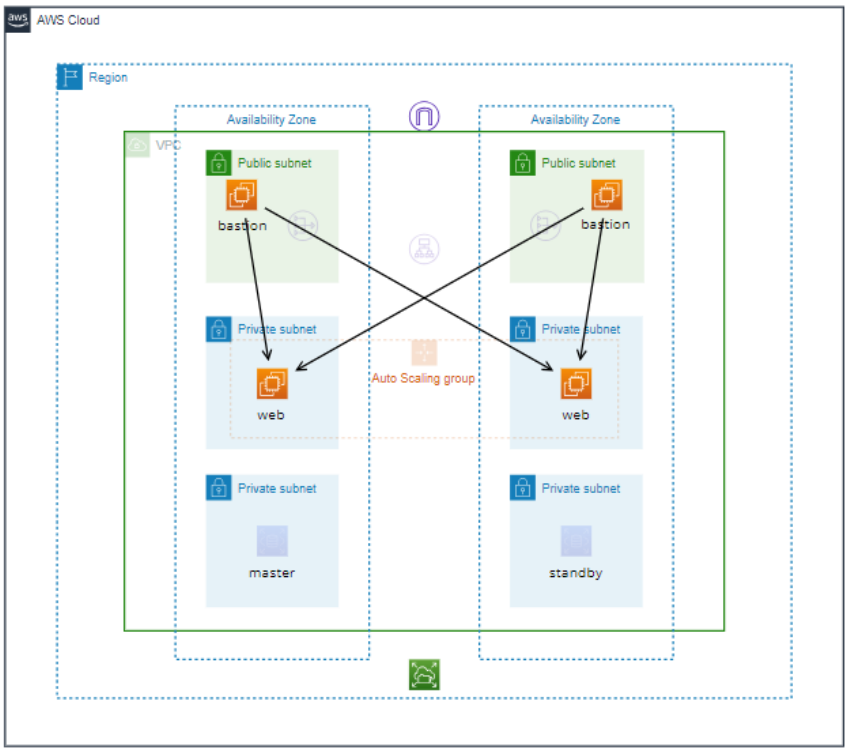
(1행은 public subnet 2개의 CIDR, 2,3행은 private subnet 4개의 CIDR

위 그림에 있는 아키텍처와 대조해보면 된다.)

요구조건 별 설명

1. bastion host를 통해서만 웹서버 인스턴스들에 ssh로 접속 가능하게 한다.

bastion host란 외부망과 내부망을 중계하는 호스트로, 내부망을 접속할 때 반드시 bastion host를 통해서만 접속 가능하다. 이렇게 bastion host를 두는 이유는 바깥에 노출되는 지점을 bastion host로 최소화하여 보안을 강화하기 위해서다.



client는 ssh로 public에 있는 bastion host를 통해서만 web인스턴스에 접속할 수 있게 했다.

bastion host에는 web instance에 ssh인증 가능한 private key(web-key.pem)이 있다.

(bastion호스트를 이중화한 이유는 장애에 대비하기 위해서이다. 한 가용영역에 문제가 생겨도 다른 가용영역의 bastion host가 관문 역할을 할 수 있다.)

테스트

bastion a1를 거쳐서 web서버에 접속 가능한지 확인

```
ec2-user@ip-10-16-1-144:~
[ec2-user@ip-10-16-1-144 ~]$ ls
web-key.pem
[ec2-user@ip-10-16-1-144 ~]$
```

bastion host에 ssh로 접속 성공

```
[ec2-user@ip-10-16-1-144 ~]$ ls
web-key.pem
[ec2-user@ip-10-16-1-144 ~]$ ssh -i web-key.pem ec2-user@10.16.3.201
Last login: Sun Mar 17 15:51:19 2024 from ip-10-16-1-144.ap-northeast-2.compute
internal
       __|  __|_  )
       _|  (     /   Amazon Linux 2
      ___|\___|___|
                      AL2 End of Life is 2025-06-30.
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
  https://aws.amazon.com/linux/amazon-linux-2023/
[ec2-user@ip-10-16-3-201 ~]$
```

bastion host에 저장된 private key로 웹서버에 ssh 접속 성공

2. private subnet에 있는 웹서버 인스턴스로 로드밸런싱

ALB를 생성하여 private subnet에 있는 웹서버에 로드밸런싱한다.

AWS ALB의 원리를 설명하면 다음과 같다. ALB를 생성하면 ALB에 연결된 가용영역에 ALB node라는 것이 생성된다.(ALB node는 ENI 다.)

ALB DNS 네임에는 ALB node의 public ip 여러개가 매핑되어 있다. client가 ALB의 DNS 네임으로 요청을 보내려고 하면 라운드로빈 식으로 resolve(ip 찾아오기)한다.

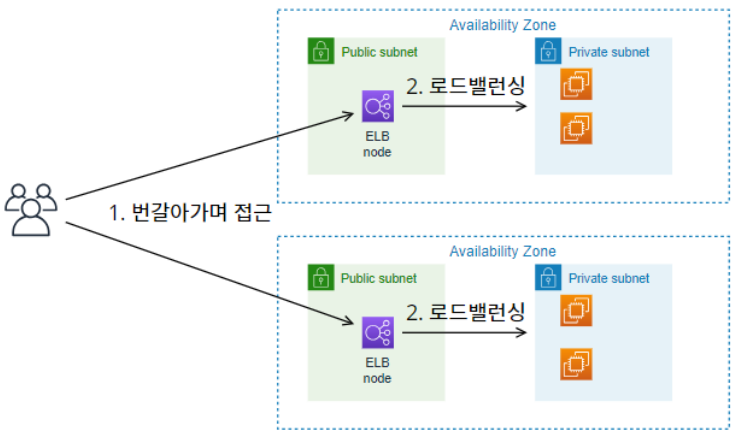
그럼 client들은 각 ALB node에 동일한 비율로 접속을 하게 된다. 패킷을 받은 ALB node는 자신이 속한 가용영역에 있는 대상 인스턴스들에게 트래픽을 로드밸런싱하게 된다.

(교차영역 옵션이 ON되면 다른 가용영역에 있는 대상 인스턴스들도 로드밸런싱 후보에 넣는다.)

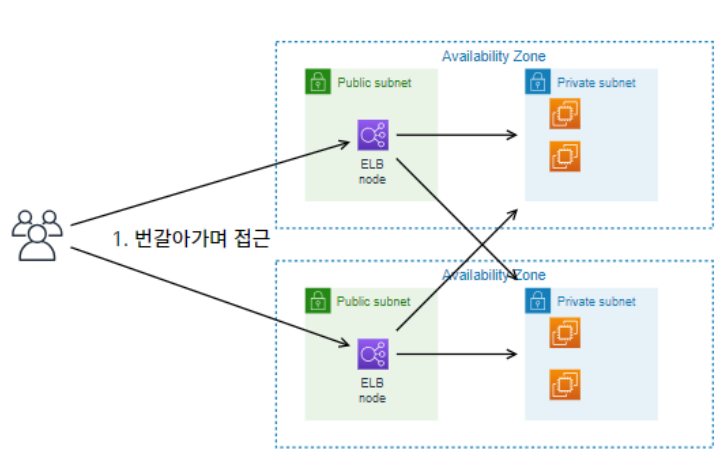
이와 같은 메커니즘 때문에 외부에서 ALB에 접근하는경우 ALB node는 반드시 public subnet에 생성되어야 한다.(외부 인터넷에서 직접 ALB node에 접근해야 하므로)

ALB는 리스너의 조건에 만족하는 트래픽이 들어오면 이와 연결된 타겟 그룹(로드밸런싱 후보의 모음)으로 로드밸런싱한다.

요약그림은 아래와 같다.



기본 로드밸런싱

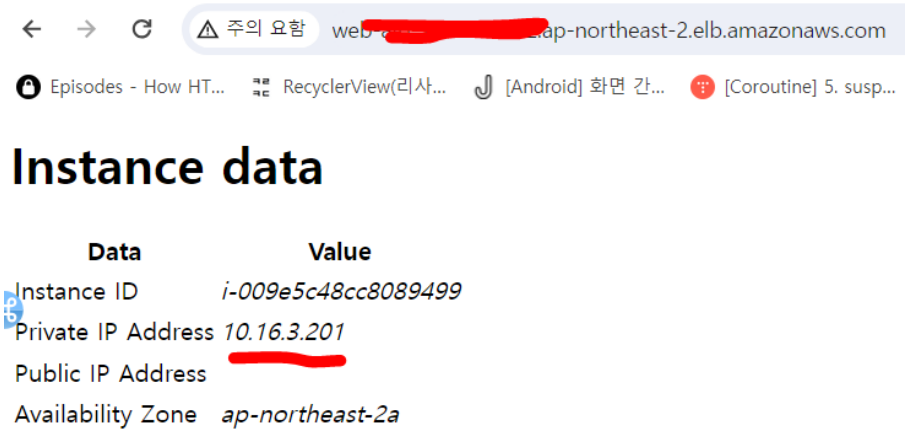


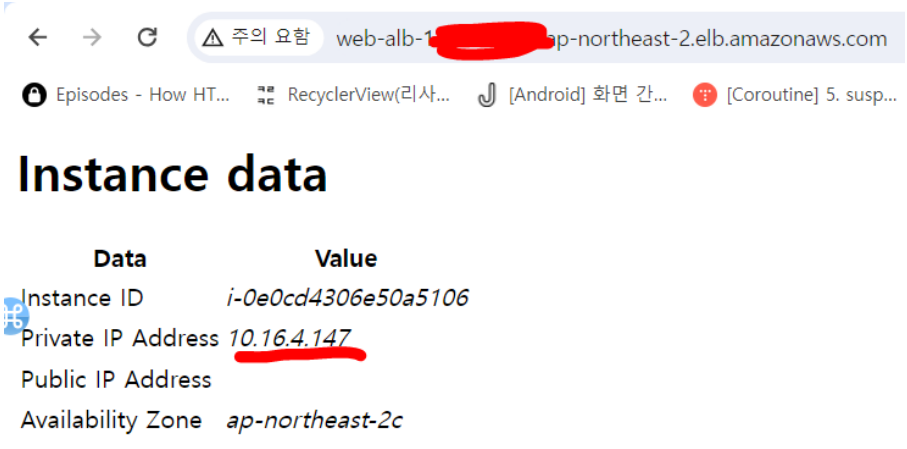
교차영역 로드밸런싱

테스트

웹서버 EC2의 private ip가 출력되는 웹페이지를 통해 로드밸런싱 여부를 테스트한다.

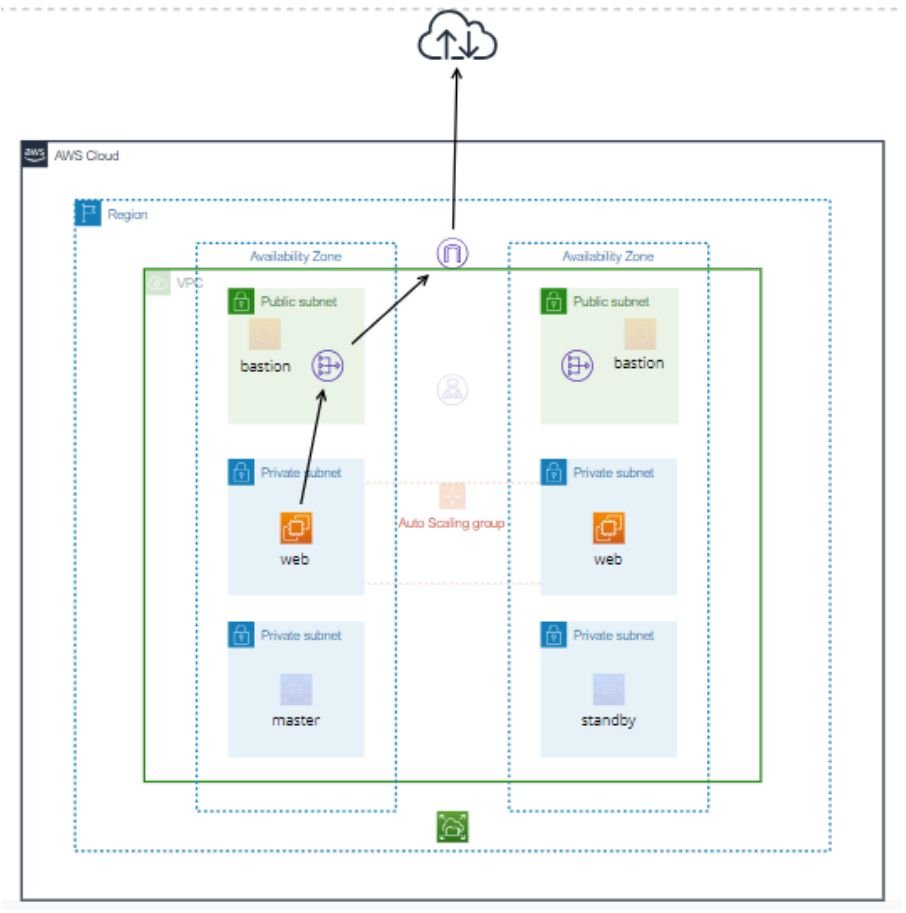
ALB의 DNS주소로 접근하였을 때 아래처럼 두 private ip가 번갈아가며 출력된다.

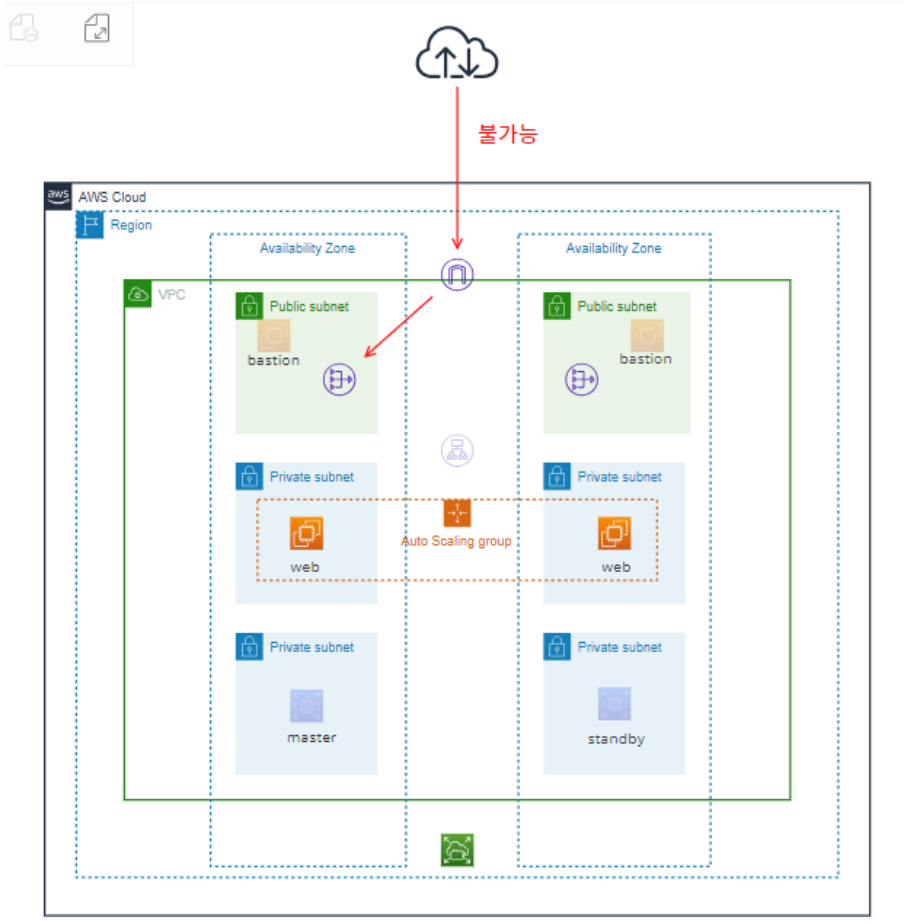




로드밸런싱이 잘 되는 것을 볼 수 있다.

3. 웹서버 인스턴스는 인터넷 사용가능(인터넷에서 웹서버에 접속 시도는 불가)





private subnet에 있는 웹서버 인스턴스에서 인터넷에 접속할 수 있게 하려면 public subnet에 있는 NAT gateway를 통해야 한다. NAT gateway를 거친 패킷은 마치 public subnet에서 시작한 것처럼 송신IP가 변경된다. 또한 NAT gateway는 외부에서 private subnet내로 들어오는 트래픽은 거부하기 때문에 안전하게 통신가능하다.

테스트

private subnet에 있는 web 인스턴스가 인터넷이 가능한지 확인한다.

ping 8.8.8.8이 성공하면 NAT gateway를 통해서 인터넷 접속성공(8.8.8.8 → 구글 DNS서버)

```
[ec2-user@ip-10-16-3-201 ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=104 time=29.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=104 time=28.1 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=104 time=28.2 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=104 time=28.2 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=104 time=28.1 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=104 time=28.3 ms
```

웹서버에서 nat gateway를 통해 인터넷 접속 성공

---

4. 웹서버 인스턴스는 오토스케일링으로 가용성을 유지한다.

트래픽에 수요에 비례하여 웹서버 EC2 인스턴스 개수를 조정한다. 스케일링 기준은 CPU 사용률로 정했다. 각 EC2의 CPU 사용률이 일정 기준을 초과하면 scale out(인스턴스 개수 증가), 일정 기준 이하면 scale in(인스턴스 개수 감소)한다.

테스트

- **Scale out**

인스턴스 ID	수명 주기	인스턴스 유형	가중치 기반 용량	시작 템플릿/구성	가용 영역	상태
i-033aaac782eb4ad14	InService	t2.micro	-	web-st | 버전 1	ap-northeast-2c	Healthy
i-08619b72af508375a	InService	t2.micro	-	web-st | 버전 1	ap-northeast-2a	Healthy

기존 웹서버 인스턴스(2개)

여기서 웹서버에 빈번히 접속하여 CPU 사용률을 의도적으로 높혀보았다.

상태	설명	원인	시작 시간	종료 시간
성공	Launching a new EC2 instance: i-0def3cd821bb107fc	At 2024-03-15T03:28:20Z a monitor alarm TargetTracking-web-asg-AlarmHigh-00dc8bae-0445-4d67-a845-b5f45a597b7f in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 3. At 2024-03-15T03:28:24Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 3.	2024 March 15, 12:28:26 PM +09:00	2024 March 15, 12:29:57 PM +09:00

타겟 조정 정책에 의해 인스턴스 개수를 2 → 3 으로 조정했다는 활동기록이 뜬다.

인스턴스 ID	수명 주기	인스턴스 유형	가중치 기반 용량	시작 템플릿/구성	가용 영역	상태
i-033aaac782eb4ad14	InService	t2.micro	-	web-st | 버전 1	ap-northeast-2c	Healthy
i-08619b72af508375a	InService	t2.micro	-	web-st | 버전 1	ap-northeast-2a	Healthy
i-0def3cd821bb107fc	InService	t2.micro	-	web-st | 버전 1	ap-northeast-2c	Healthy

웹서버 인스턴스가 3개로 증가

- **Scale In**

이번엔 아무짓도 하지 않고 그대로 기다린다. CPU사용률이 일정 기준 이하로 내려가면 인스턴스 개수를 감소시킨다.

상태	설명	원인	시작 시간	종료 시간
성공	Terminating EC2 instance: i-033aaac782eb4ad14	At 2024-03-15T03:32:49Z a monitor alarm TargetTracking-web-asg-AlarmLow-66938b46-ac17-4d5c-b051-309840613fae in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 3 to 2. At 2024-03-15T03:33:00Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 3 to 2. At 2024-03-15T03:33:00Z instance i-033aaac782eb4ad14 was selected for termination.	2024 March 15, 12:33:00 PM +09:00	2024 March 15, 12:38:46 PM +09:00

타겟 조정 정책에 의해 인스턴스 개수를 3 → 2로 조정한다는 활동기록이 뜬다.

인스턴스 ID	수명 주기	인스턴스 유형	가중치 기반 용량	시작 템플릿/구성	가용 영역	상태	다음으로부터 보...
i-08619b72af508375a	InService	t2.micro	-	web-st | 버전 1	ap-northeast-2a	Healthy	
i-0def3cd821bb107fc	InService	t2.micro	-	web-st | 버전 1	ap-northeast-2c	Healthy	

웹서버 인스턴스가 2개로 줄어들었다.(scale in)

이런 식으로 Auto scaling을 사용하여 트래픽 수요에 유연하게 대처할 수 있다.

5. rds의 경우, master에 문제가 생길 시 failover(standby → active)가 가능해야 한다.

RDS에서 Multi AZ instance 배포 옵션을 사용하여 데이터베이스를 생성할 경우, 서브넷 그룹 중에서 임의의 서브넷에 master(실제 입출력 수행 DB 인스턴스)와 다른 가용영역에 있는 서브넷에 standby(백업 DB 인스턴스)를 생성한다.

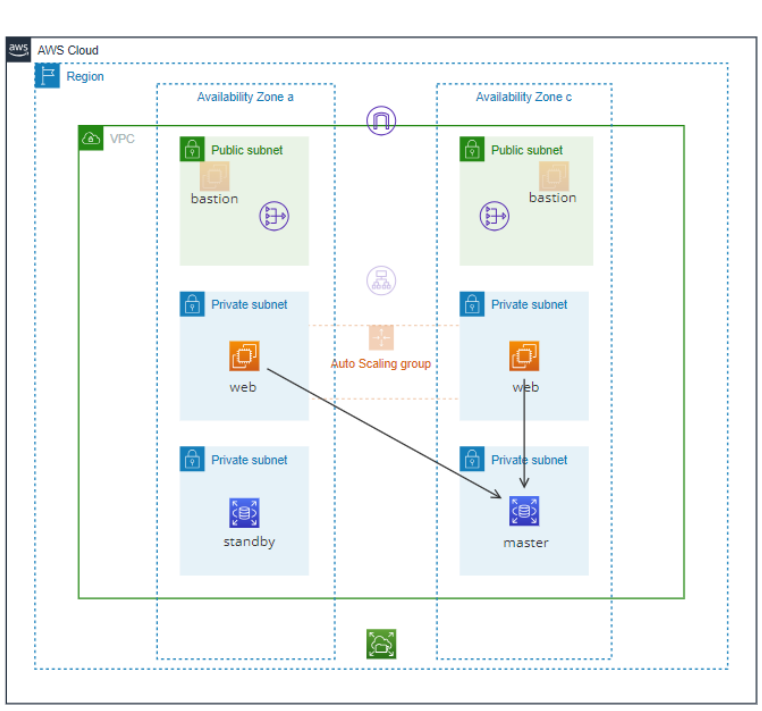
standby는 입출력 불가하고, master에 write가 발생할 때마다 synchronous replication방식으로 standby를 동기화 시킨다. 만일 master에 문제가 생겼을 경우 자동적으로 standby를 failover하여 서비스를 이어서 제공한다.

테스트

failover가 잘되는지 테스트한다. 가용영역에 일부러 장애를 만들기는 어려우니 AWS 콘솔에서 RDS master 인스턴스를 의도적으로 “장애 조치로 재부팅”한다.

failover여부는 RDS DNS네임을 resolve했을 때 private IP가 달라졌는지로 판단한다.

현재상태



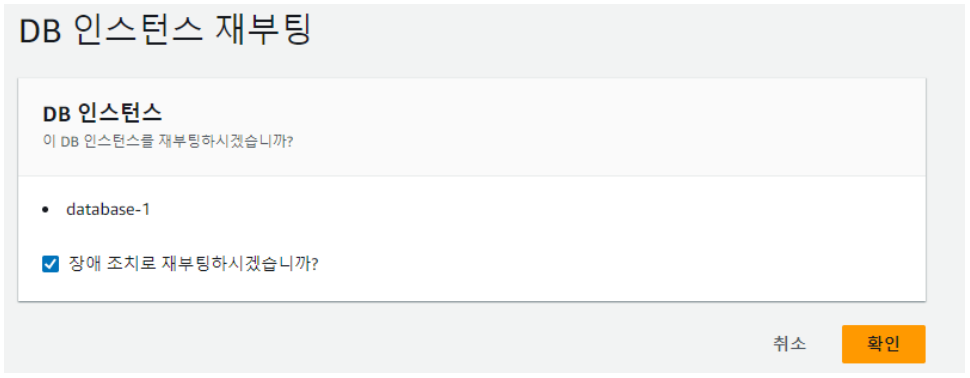
현재 웹서버들은 가용영역 c에 있는 RDS 인스턴스(master)에 접속하여 입출력을 수행 중이다.

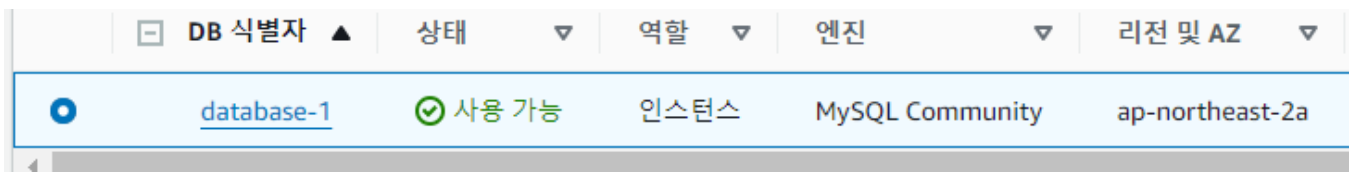
```
[ec2-user@ip-10-16-4-18 html]$ nslookup database-1.cbsg6ogyabnr.ap-northeast-2.r
ds.amazonaws.com
Server:         10.16.0.2
Address:        10.16.0.2#53

Non-authoritative answer:
Name:   database-1.cbsg6ogyabnr.ap-northeast-2.rds.amazonaws.com
Address: 10.16.6.137
```

RDS 엔드포인트 DNS를 resolve했을 때도 IP가 가용영역 C에 있는 서브넷의 CIDR(10.16.6.0/24)에 속한다. 해당 IP는 master DB instance의 ENI의 private IP이다.

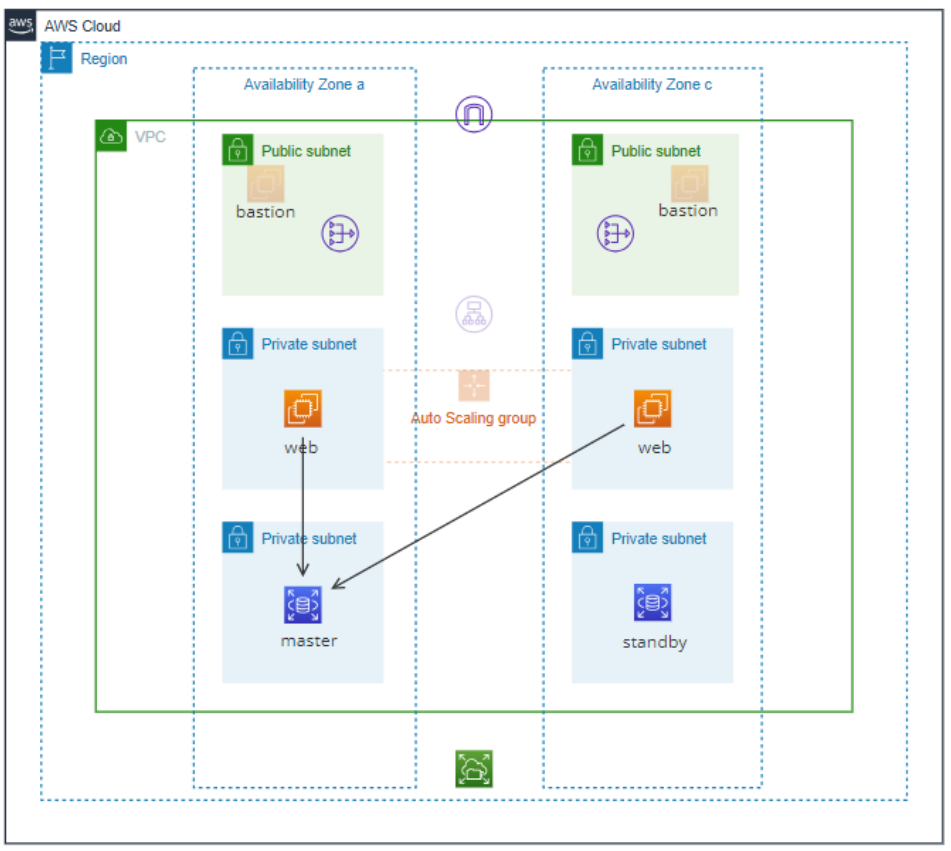
이 상태에서 의도적으로 master에 장애(재부팅)을 발생시켜보겠다.





failover되는데 1분도 걸리지 않았다. DB 인스턴스의 가용영역이 a로 바뀐 것을 볼 수 있다(a == ㅁap-northeast-2a)

failover 후의 구조는 다음과 같다.



가용영역 a에 있는 DB 인스턴스가 master가 됐다. 이제 웹서버들은 가용영역 a에 있는 DB 인스턴스에 입출력을 수행한다.

```
[ec2-user@ip-10-16-4-18 html]$ nslookup database-1.cbsg6ogyabnr.ap-northeast-2.r
ds.amazonaws.com
Server:         10.16.0.2
Address:        10.16.0.2#53

Non-authoritative answer:
Name:   database-1.cbsg6ogyabnr.ap-northeast-2.rds.amazonaws.com
Address: 10.16.6.137

[ec2-user@ip-10-16-4-18 html]$ nslookup database-1.cbsg6ogyabnr.ap-northeast-2.r
ds.amazonaws.com
Server:         10.16.0.2
Address:        10.16.0.2#53

Non-authoritative answer:
Name:   database-1.cbsg6ogyabnr.ap-northeast-2.rds.amazonaws.com
Address: 10.16.5.149
```

동일한 DNS네임으로 한번 더 resolve했더니 다른 IP가 반환된다. 즉, 가용영역 c가 아닌 가용영역 a에 있는 DB 인스턴스의 private IP를 반환한다.

웹서버들은 RDS의 DNS네임으로 DB인스턴스에 접속하기 때문에, failover하면 DNS로 resolve되는 IP가 자동적으로 바뀌어 웹서버들은 문제없이 DB 인스턴스에 접속할 수 있다.(관리자의 별도의 조치가 필요없다.)

Instance data

Data	Value
Instance ID	*i-0def3cd821bb107fc*
Private IP Address	*10.16.4.18*
Public IP Address	
Availability Zone	*ap-northeast-2c*

RDS Practice

ID	NAME	ADDRESS
1	KIM	SEOUL

여전히 DB에 접속할 수 있다.

Instance data

Data	Value
Instance ID	*i-08619b72af508375a*
Private IP Address	*10.16.3.218*
Public IP Address	
Availability Zone	*ap-northeast-2a*

RDS Practice

ID	NAME	ADDRESS
1	KIM	SEOUL

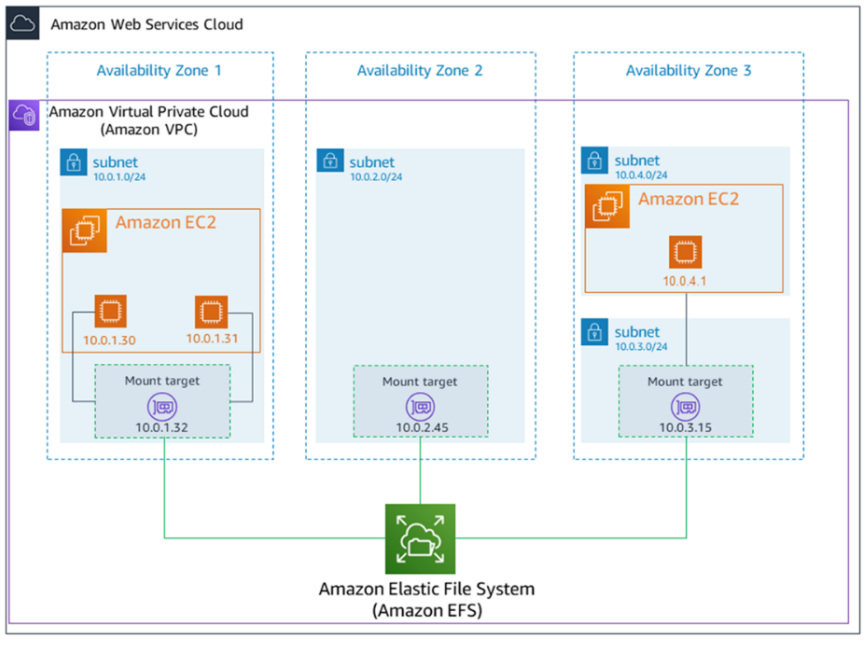
6. 웹서버 인스턴스들은 하나의 EFS에 mount되어야 한다.

EFS는 AWS에서 지원하는 NFS 기반 File Storage다. EFS를 EC2에 mount하여 마치 로컬에 있는 파일시스템인 것처럼 파일을 read/write 할 수 있다.

EFS에는 2가지 옵션이 있다.

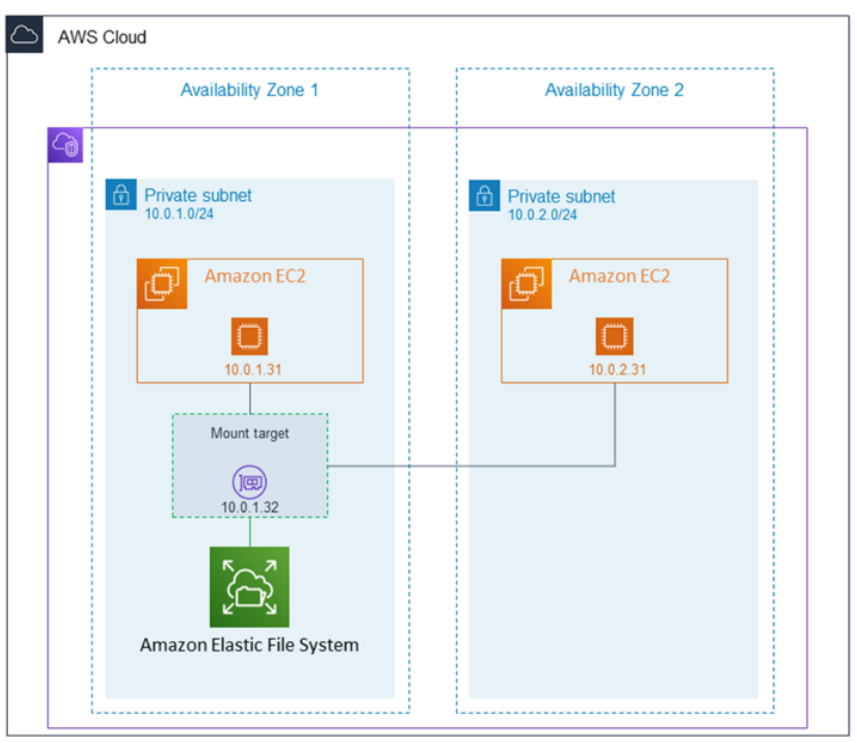
- Regional(기본) : 데이터를 여러 가용영역에 중복하여 저장한다. 때문에 한 가용영역에 문제가 생겨도 데이터는 유지된다.

  Mount target(탑재 대상)을 여러 가용영역에 생성한다.(Mount target은 EFS에 접근하기 위한 ENI이다.) 가급적이면 같은 가용영역에 있는 Mount target을 이용하는 것이 좋다.(가용영역간 데이터 이동에는 비용이 발생하므로)



각 EC2인스턴스는 자신이 속한 가용영역에 있는 Mount target을 이용하여 EFS를 mount한다.

- One zone : 한 가용영역에만 데이터를 저장하기 때문에 해당 가용영역에 문제가 생기면 데이터를 영구적으로 잃는다. Mount target은 해당 가용영역에만 생성된다.



테스트

EFS를 웹서버 EC2들에게 mount하여 파일 공유가 되는지 확인한다.

mount방법은 다음과 같다.

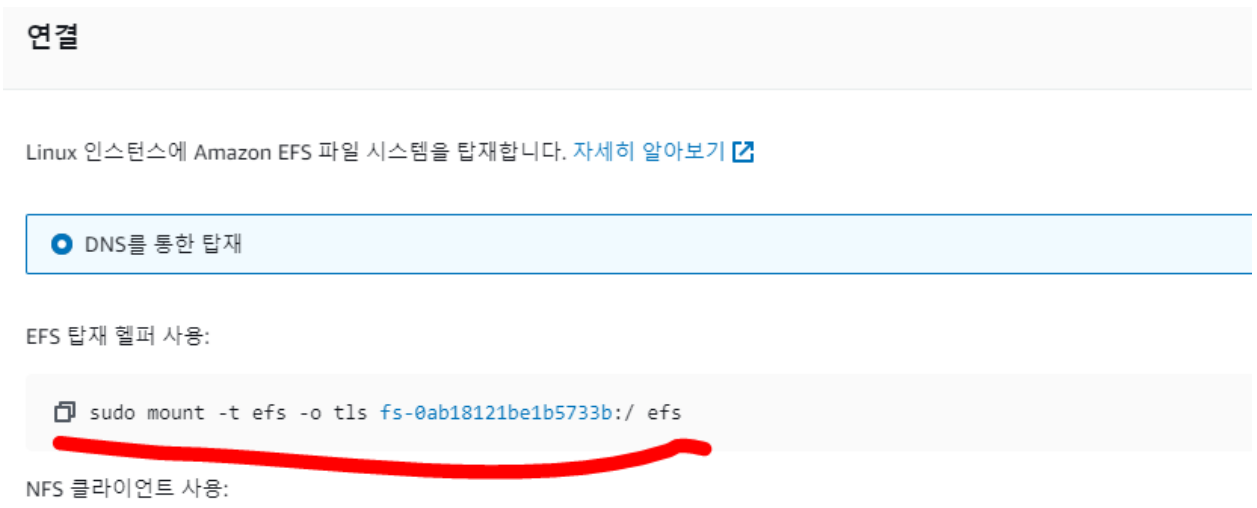
1. EFS client인 amazon-efs-utils를 설치(EFS와 통신하여 read/write하는 프로그램)

```
sudo yum install -y amazon-efs-utils
```

2. mount 대상 디렉터리 생성

```
[ec2-user@ip-10-16-4-18 html]$ mkdir efs
```

3. mount명령 복사



4. mount

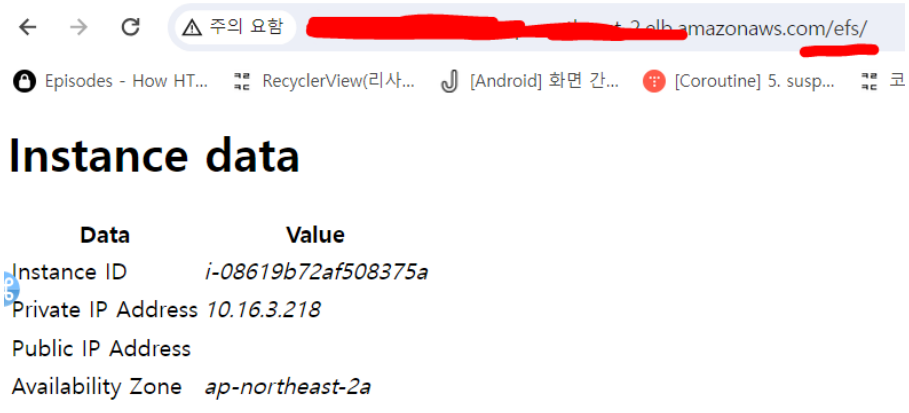
```
[ec2-user@ip-10-16-4-18 html]$ sudo mount -t efs -o tls fs-0ab18121be1b5733b:/ efs
[ec2-user@ip-10-16-4-18 html]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        468M     0  468M   0% /dev
tmpfs           477M     0  477M   0% /dev/shm
tmpfs           477M  524K  476M   1% /run
tmpfs           477M     0  477M   0% /sys/fs/cgroup
/dev/xvda1      8.0G  2.1G  6.0G  26% /
tmpfs            96M     0   96M   0% /run/user/1000
127.0.0.1:/     8.0E     0  8.0E   0% /var/www/html/efs
```

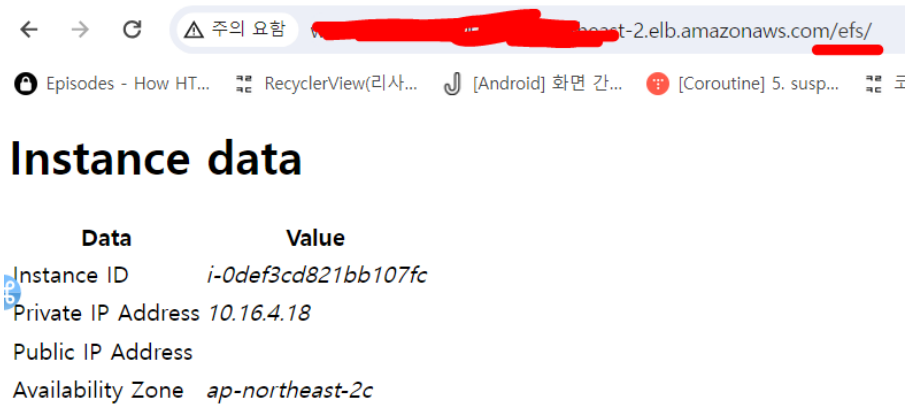
다른 웹서버에도 똑같은 과정을 거친다.

이제 모든 웹서버EC2가 EFS를 /var/www/html/efs라는 디렉터리에 mount하였으므로 파일을 공유할 수 있다. 해당 디렉터리로 파일을 옮겨서 웹페이지에서 접근 가능한지 확인해보자.

```
[ec2-user@ip-10-16-4-18 efs]$ ls
dbinfo.inc  index.php
```

둘 중 한 웹서버EC2에서 파일을 /var/www/html/efs디렉터리로 이동





두 웹서버에서 웹페이지가 보인다.(두 웹서버EC2가 index.html파일을 공유한다.)

---

7. 최소한의 네트워크 트래픽만 허용

침입을 방지하기 위해 보안그룹, 네트워크 ACL에 규칙을 설정하여 최소한의 트래픽만 허용한다.

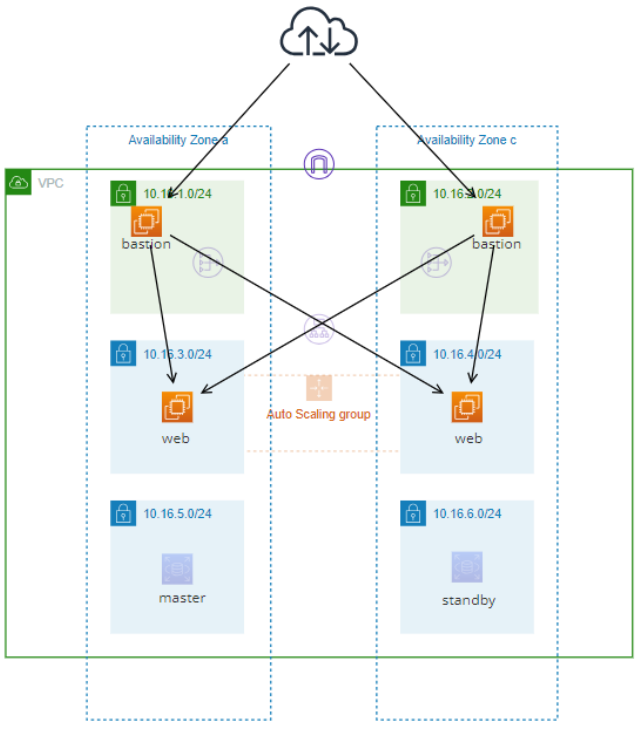
지금부터 기능별로 규칙을 정의한다.

외부 관리자가 bastion을 통해 웹서버에 ssh접속

설명

외부에서 관리자가 bastion host를 거쳐 web 인스턴스에 접속할 수 있어야한다.

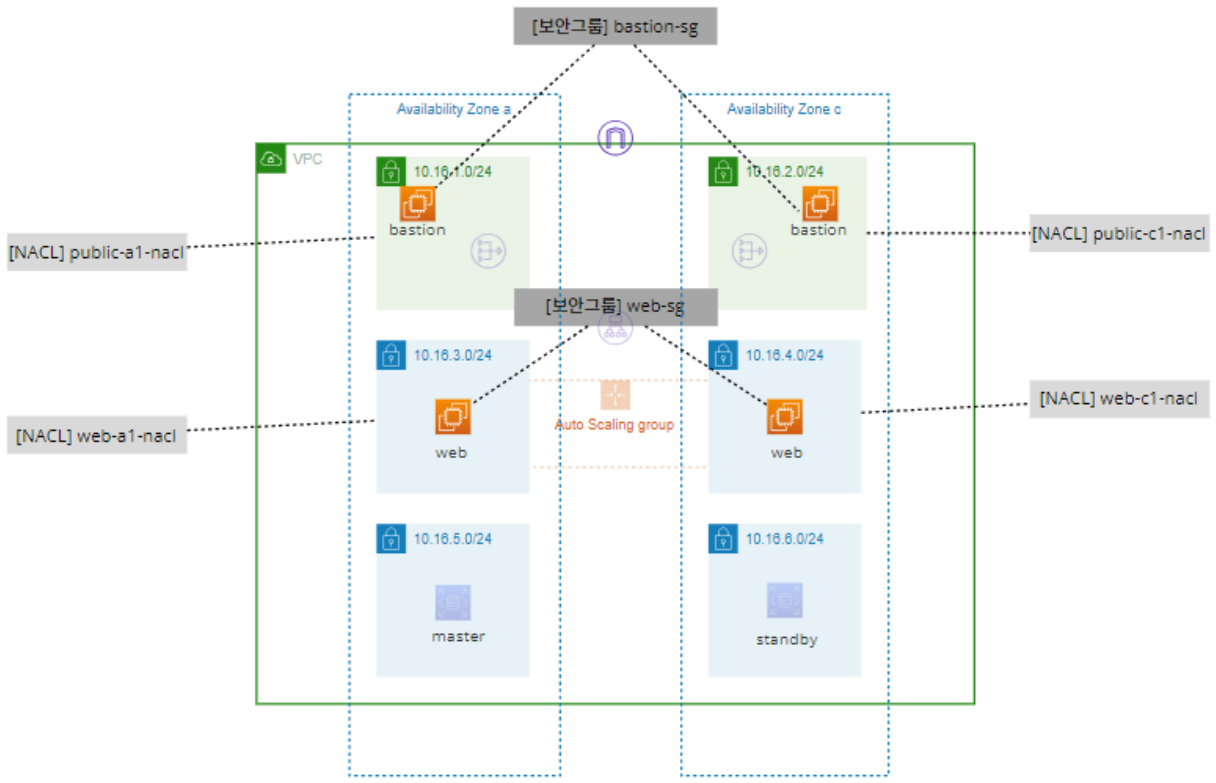
ssh 트래픽은 다음과 같다.



ssh로 bastion host를 거쳐 web서버에 접속할 수 있다.

보안규칙

네트워크 ACL, 보안그룹의 배치는 다음과 같다.



규칙은 다음과 같다.

[보안그룹] bastion-sg

- 인바운드

유형	포트 범위	소스	설명
SSH	22	0.0.0.0/0	[ssh] from client to bastion

- 아웃바운드

유형	포트 범위	대상	설명
SSH	22	sg-09fe4965f4f35451d / web-sg	[ssh] from bastion to web

[보안그룹] web-sg

- 인바운드

유형	포트 범위	소스	설명
SSH	22	sg-01131e85acdf860ce / bastion-sg	allow ssh connection from bastion to web

- 아웃바운드

  X

[NACL] public-a1-nacl, public-c1-nacl

- 인바운드

규칙 번호	유형	프로토콜	포트 범위	소스	허용/거부
100	SSH(22)	TCP(6)	22	0.0.0.0/0	Allow
200	사용자 지정 TCP	TCP(6)	32768 - 60999	10.16.3.0/24	Allow
300	사용자 지정 TCP	TCP(6)	32768 - 60999	10.16.4.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

- 아웃바운드

규칙 번호	유형	프로토콜	포트 범위	대상	허용/거부
100	사용자 지정 TCP	TCP(6)	49152 - 65535	0.0.0.0/0	Allow
200	SSH(22)	TCP(6)	22	10.16.3.0/24	Allow
300	SSH(22)	TCP(6)	22	10.16.4.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

[NACL] web-a1-nacl, web-c1-nacl

- 인바운드

규칙 번호	유형	프로토콜	포트 범위	소스	허용/거부
100	SSH(22)	TCP(6)	22	10.16.1.0/24	Allow
200	SSH(22)	TCP(6)	22	10.16.2.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

- 아웃바운드

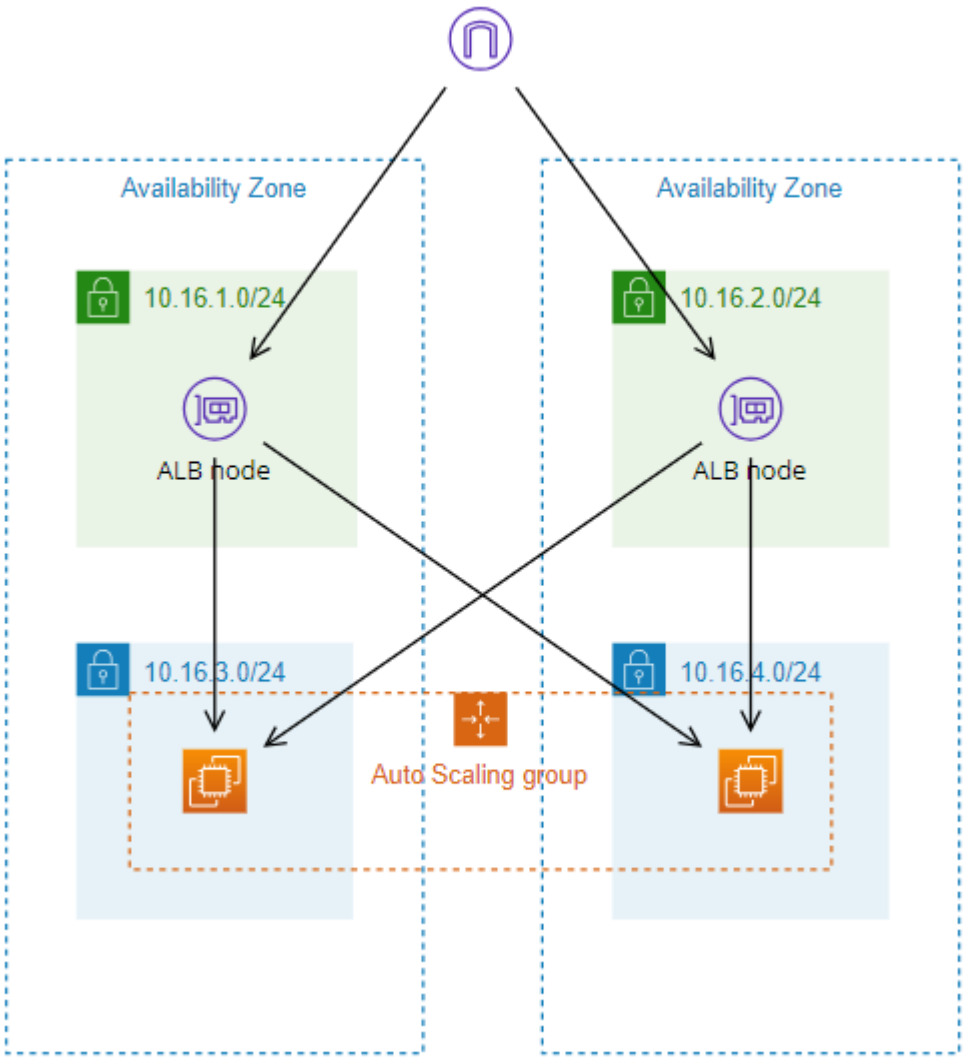
규칙 번호	유형	프로토콜	포트 범위	대상	허용/거부
100	사용자 지정 TCP	TCP(6)	32768 - 60999	10.16.1.0/24	Allow
200	사용자 지정 TCP	TCP(6)	32768 - 60999	10.16.2.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

외부 클라이언트가 웹서버에 접속

설명

외부 브라우저에서 private subnet에 있는 웹서버에 http접속하여 웹사이트를 볼 수 있어야 한다.

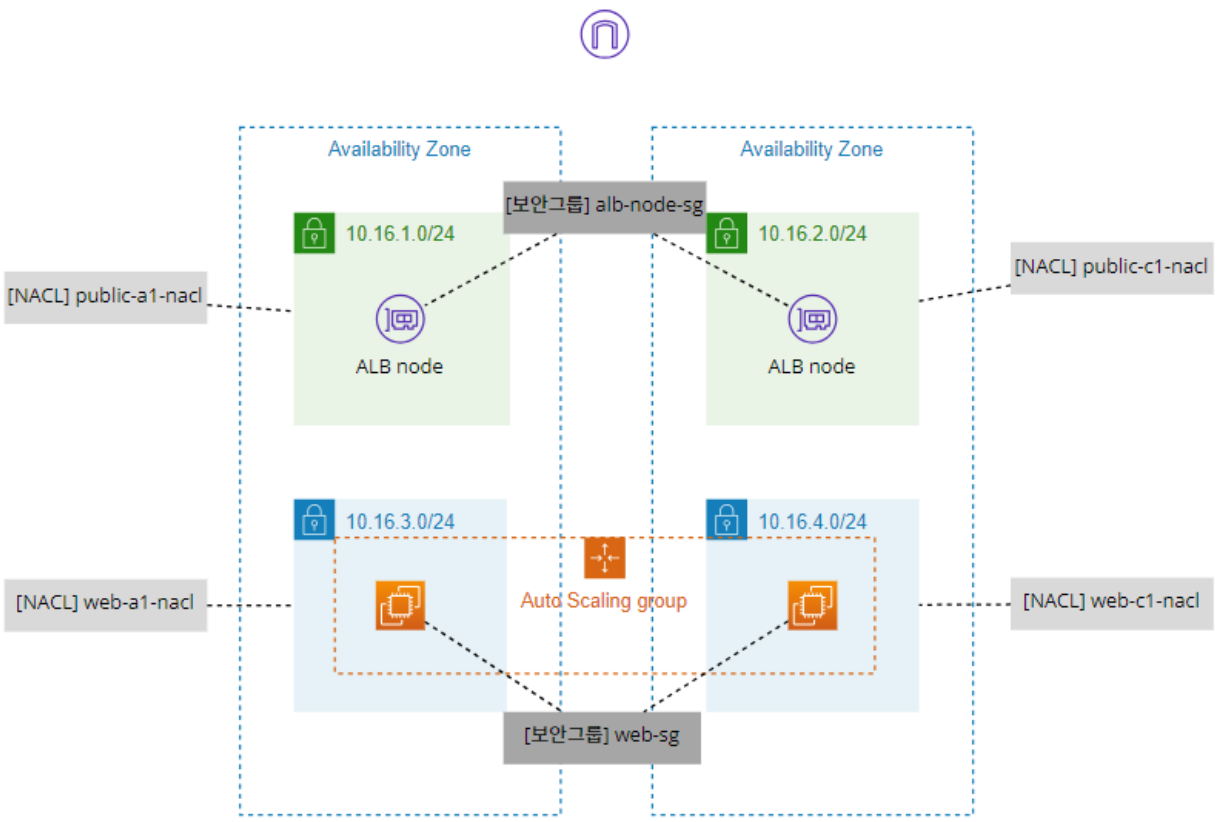
http트래픽은 다음과 같다.



ALB를 생성하면 각 가용영역의 public subnet에 alb node를 생성한다. 실질적인 로드밸런싱은 해당 node가 수행한다(alb node는 ENI). 교차영역 로드밸런싱 옵션을 ON한 상태이다.

보안규칙

네트워크 ACL, 보안그룹의 배치는 다음과 같다.



규칙은 다음과 같다.

[보안그룹] alb-node-sg

- 인바운드

유형	포트 범위	소스	설명
HTTP	80	0.0.0.0/0	[http] from client to alb node

- 아웃바운드

유형	포트 범위	대상	설명
HTTP	80	sg-09fe4965f4f35451d / web-sg	[http, health check] from web to alb node

[보안그룹] web-sg

- 인바운드

유형	포트 범위	소스	설명
HTTP	80	sg-017dddf7bf06b2993 / alb-node-sg	[http] from alb node to web

- 아웃바운드

  X

[NACL] public-a1-nacl, public-c1-nacl

- 인바운드

규칙 번호	유형	프로토콜	포트 범위	소스	허용/거부
1100	HTTP(80)	TCP(6)	80	0.0.0.0/0	Allow
1200	모든 TCP	TCP(6)	모두	10.16.3.0/24	Allow
1300	모든 TCP	TCP(6)	모두	10.16.4.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

- 아웃바운드

규칙 번호	유형	프로토콜	포트 범위	대상	허용/거부
1100	HTTP(80)	TCP(6)	80	10.16.3.0/24	Allow
1200	HTTP(80)	TCP(6)	80	10.16.4.0/24	Allow
1300	사용자 지정 TCP	TCP(6)	49152 - 65535	0.0.0.0/0	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

[NACL] web-a1-nacl, web-c1-nacl

- 인바운드

규칙 번호	유형	프로토콜	포트 범위	소스	허용/거부
1100	HTTP(80)	TCP(6)	80	10.16.1.0/24	Allow
1200	HTTP(80)	TCP(6)	80	10.16.2.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

- 아웃바운드

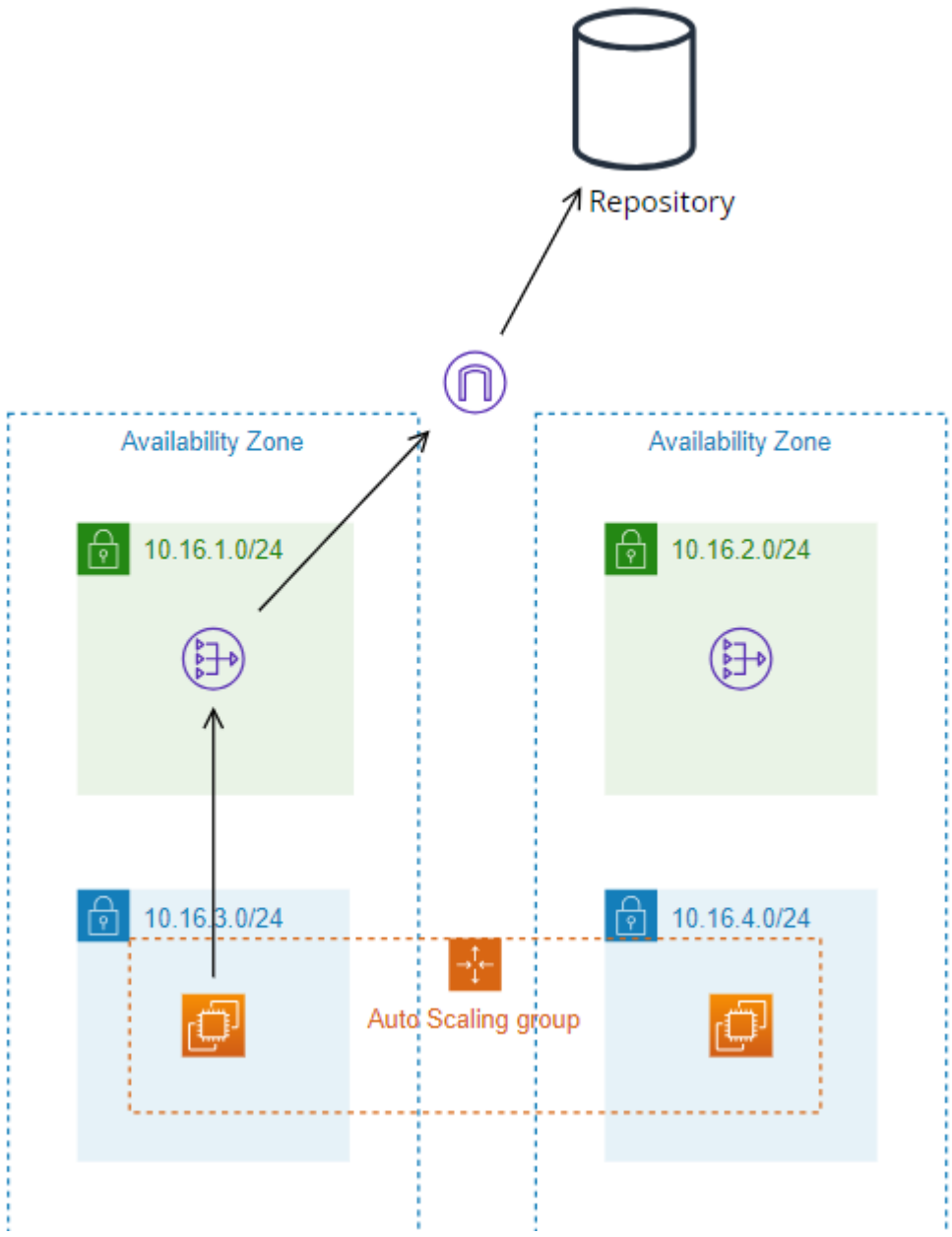
규칙 번호	유형	프로토콜	포트 범위	대상	허용/거부
1100	모든 TCP	TCP(6)	모두	10.16.1.0/24	Allow
1200	모든 TCP	TCP(6)	모두	10.16.2.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

웹서버에서 Yum 사용

설명

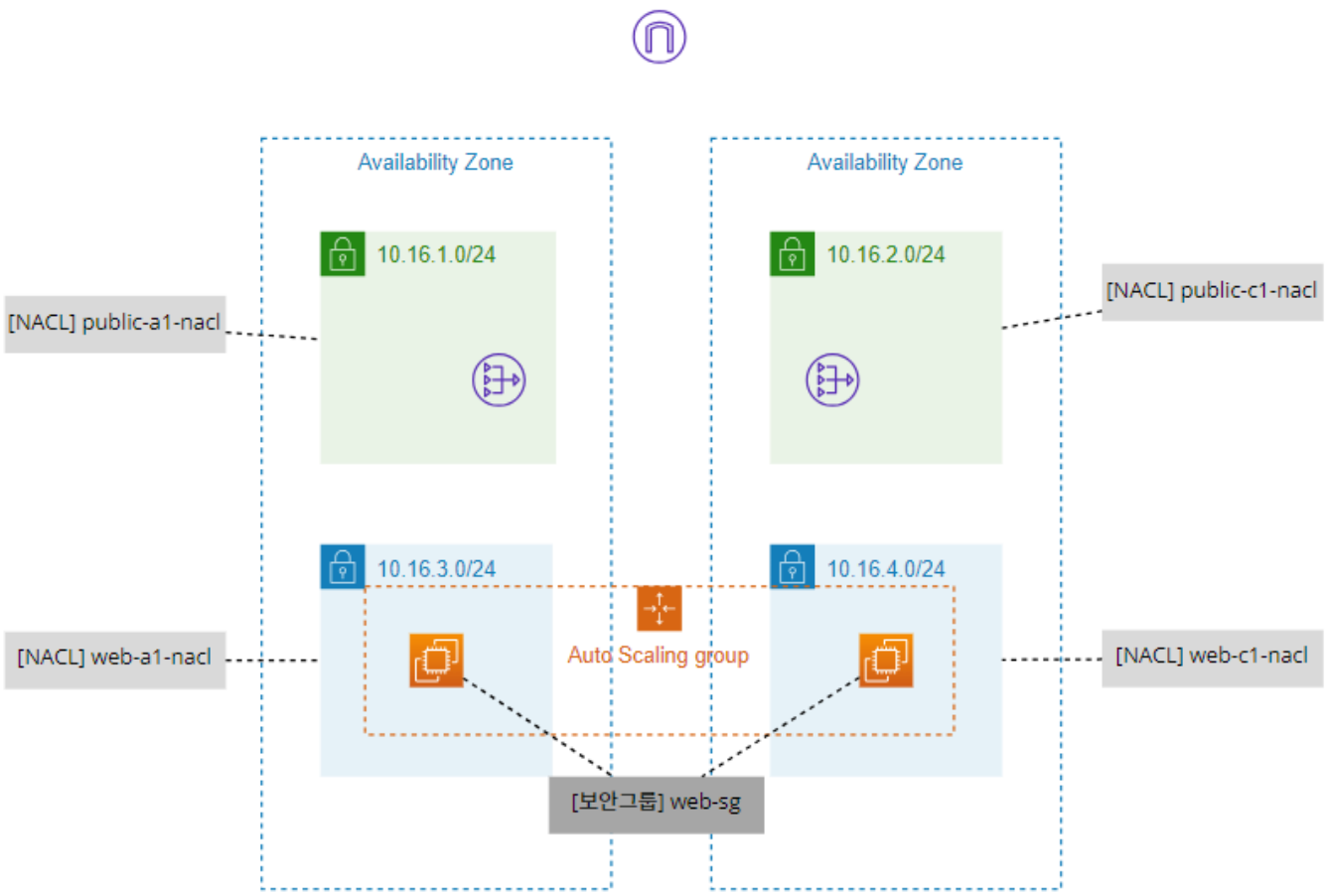
웹서버에서 yum 레포지토리에 접근하여 패키지 설치가 가능해야한다.

(레포지토리 외에 다른 ip로 접근은 허용X)



보안규칙

네트워크 ACL, 보안그룹의 배치는 다음과 같다.



nat gateway도 ENI긴 하나 보안그룹은 붙지 않는다.

※ 확인해본 결과, 현재 웹서버들은 52.219.0.0/16 대역 ip를 가진 레포지토리에 https로 접근했다.

따라서 보안그룹 및 네트워크 ACL에서 해당 트래픽만 허용해주었다.

[보안그룹] web-sg

- 인바운드

  X

- 아웃바운드

유형	포트 범위	대상	설명
HTTPS	443	52.219.0.0/16	repository access for yum

[NACL] public-a1-nacl

- 인바운드

규칙 번호	유형	프로토콜	포트 범위	소스	허용/거부
2100	사용자 지정 TCP	TCP(6)	32768 - 60999	52.219.0.0/16	Allow
2200	HTTPS(443)	TCP(6)	443	10.16.3.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

- 아웃바운드

규칙 번호	유형	프로토콜	포트 범위	대상	허용/거부
2100	HTTPS(443)	TCP(6)	443	52.219.0.0/16	Allow
2200	사용자 지정 TCP	TCP(6)	32768 - 60999	10.16.3.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

[NACL] web-a1-nacl

- 인바운드

규칙 번호	유형	프로토콜	포트 범위	소스	허용/거부
2100	사용자 지정 TCP	TCP(6)	32768 - 60999	52.219.0.0/16	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

- 아웃바운드

규칙 번호	유형	프로토콜	포트 범위	대상	허용/거부
2100	HTTPS(443)	TCP(6)	443	52.219.0.0/16	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

[NACL] public-c1-nacl

- 인바운드

규칙 번호	유형	프로토콜	포트 범위	소스	허용/거부
2100	사용자 지정 TCP	TCP(6)	32768 - 60999	52.219.0.0/16	Allow
2200	HTTPS(443)	TCP(6)	443	10.16.4.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

- 아웃바운드

규칙 번호	유형	프로토콜	포트 범위	대상	허용/거부
2100	HTTPS(443)	TCP(6)	443	52.219.0.0/16	Allow
2200	사용자 지정 TCP	TCP(6)	32768 - 60999	10.16.4.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

[NACL] web-c1-nacl

- 인바운드

규칙 번호	유형	프로토콜	포트 범위	소스	허용/거부
2100	사용자 지정 TCP	TCP(6)	32768 - 60999	52.219.0.0/16	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

- 아웃바운드

규칙 번호	유형	프로토콜	포트 범위	대상	허용/거부
2100	HTTPS(443)	TCP(6)	443	52.219.0.0/16	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

웹서버를 EFS 파일 시스템에 연결

설명

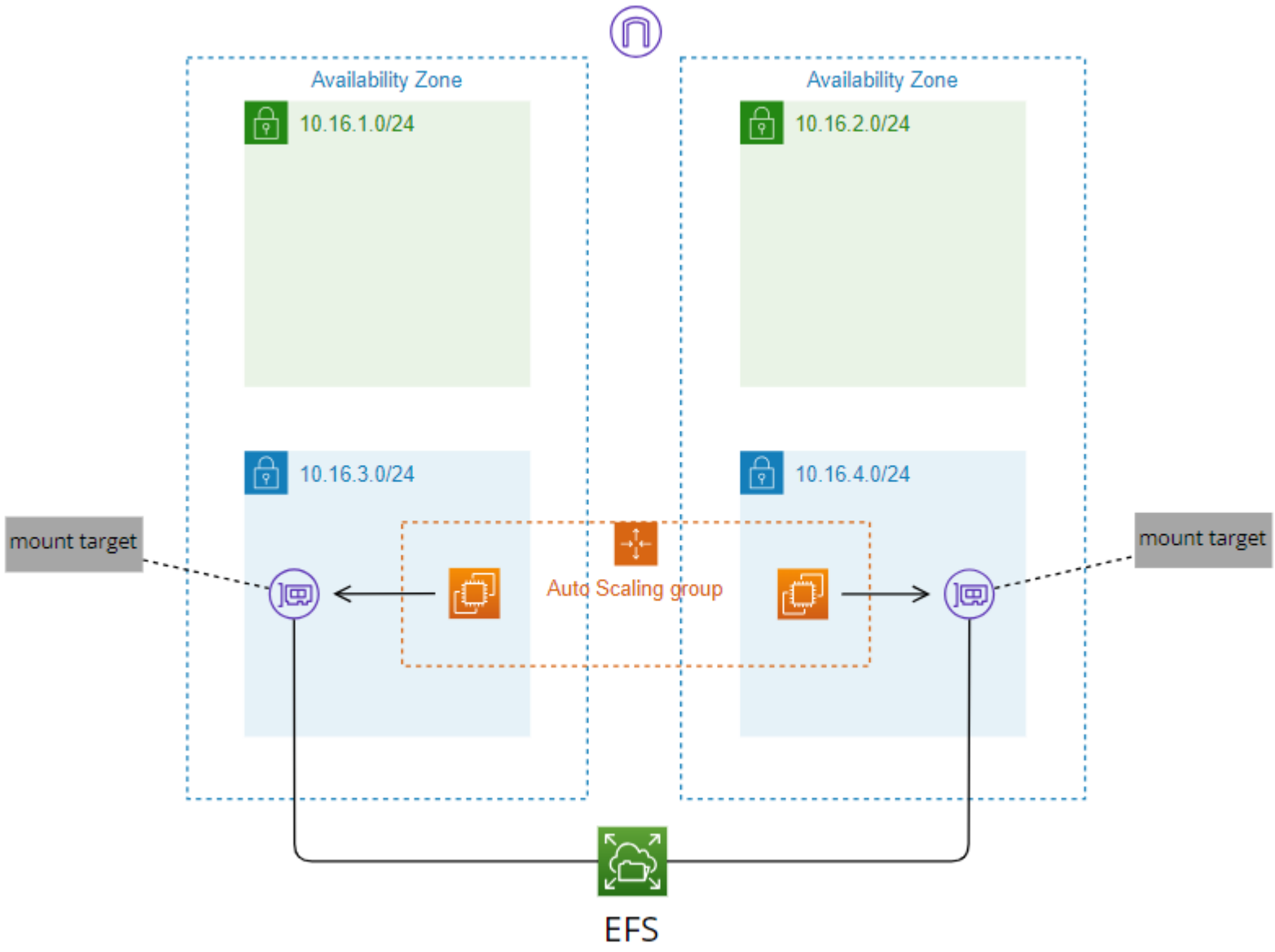
EFS(Elastic File System)은 NFS프로토콜을 사용하는 원격 파일 시스템이다.

client의 NFS client가 EFS와 통신하여 파일을 read/write한다.

client는 EFS를 디렉터리에 mount함으로써 사용가능하다.

(정확히는 mount target(ENI)를 인스턴스에 mount)

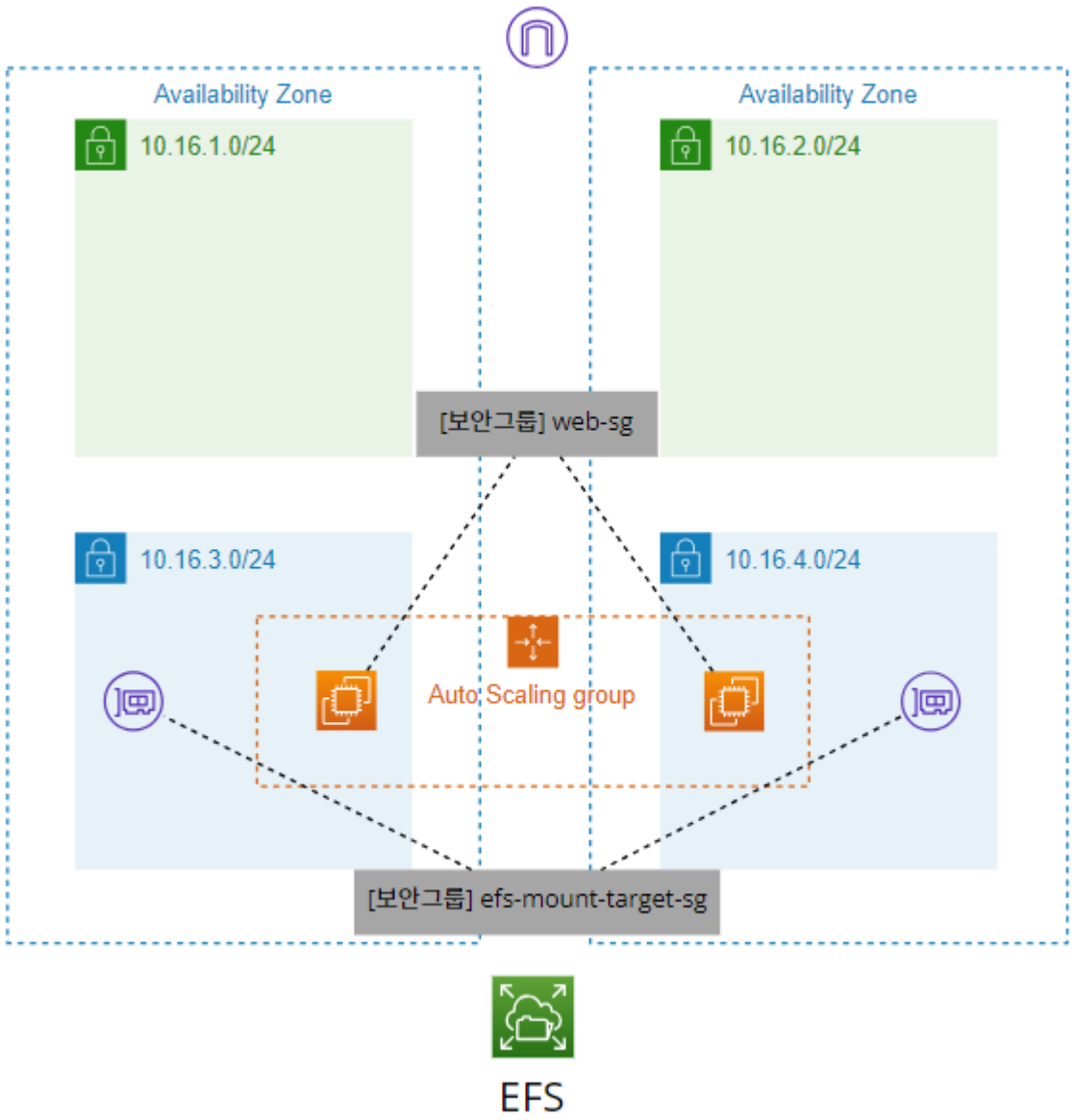
EFS 관련 트래픽은 아래와 같다.



web ec2는 각각 자신과 같은 가용영역에 존재하는 mount target에 mount하여 파일을 read/write한다. (TIP : 같은 subnet내의 통신이라면 라우팅 테이블 규칙 설정 필요X)

보안규칙

네트워크 ACL, 보안그룹의 배치는 다음과 같다.



web ec3가 efs mount target에게 nfs트래픽을 보낼 수 있도록 규칙을 설정한다.

이번 경우, 같은 subnet 내에서 통신하므로 보안그룹만으로 트래픽을 제어한다

[보안그룹] web-sg

- 인바운드

  X

- 아웃바운드

유형	포트 범위	대상	설명
NFS	2049	sg-06d9028662a29df7a / efs-mount-...	[efs] from web to efs mount target

[보안그룹] efs-mount-target-sg

- 인바운드

유형	포트 범위	소스	설명
NFS	2049	sg-09fe4965f4f35451d / web-sg	[efs] from web to efs mount target

- 아웃바운드

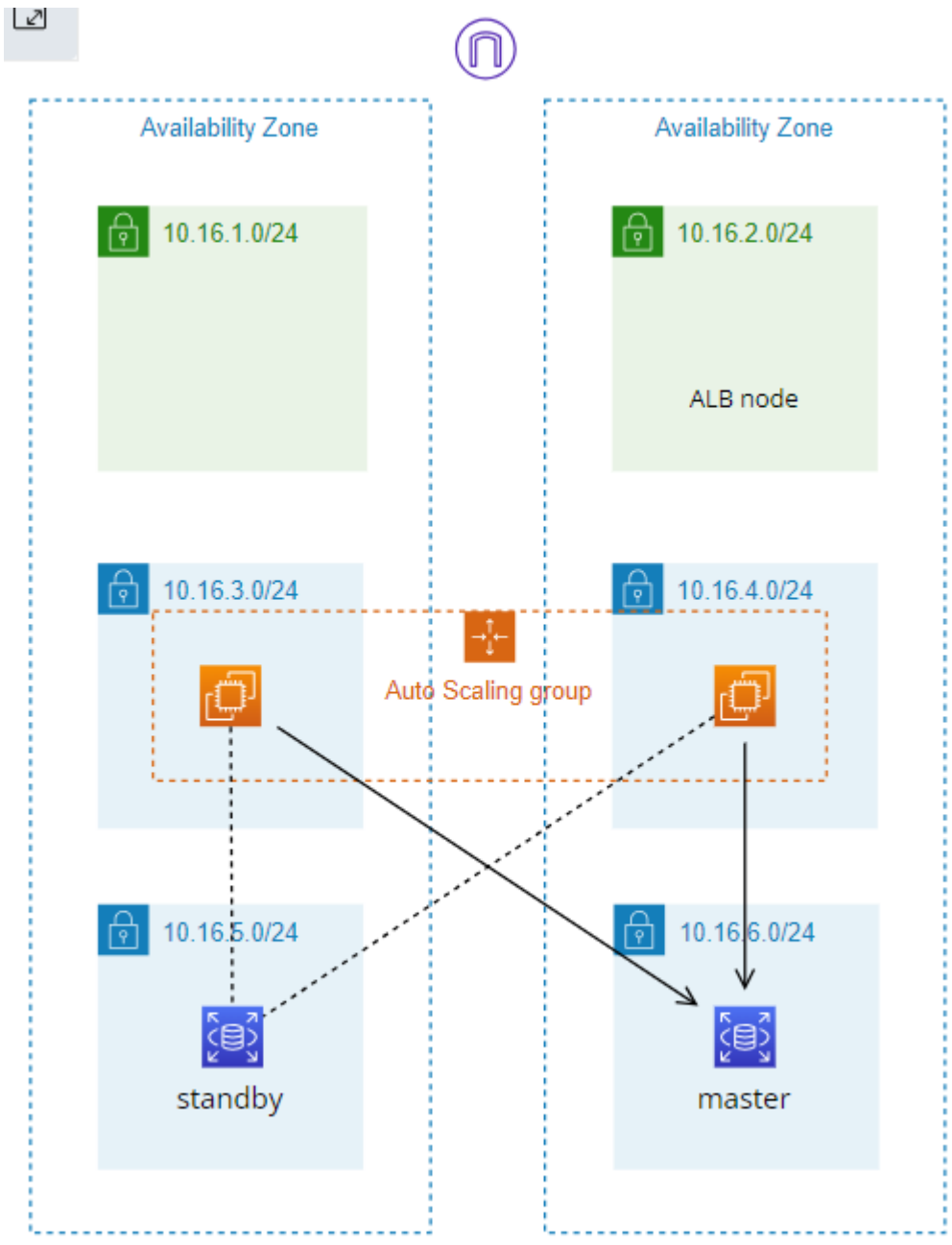
  X

웹서버가 RDS에 접속 가능하도록(failover고려)

설명

웹서버가 RDS에 접속할 수 있도록 트래픽을 허용한다. 이때, master instance에 문제가 생겨 failover(standby→active)되어도 정상적으로 접속이 되도록 규칙을 설정한다.

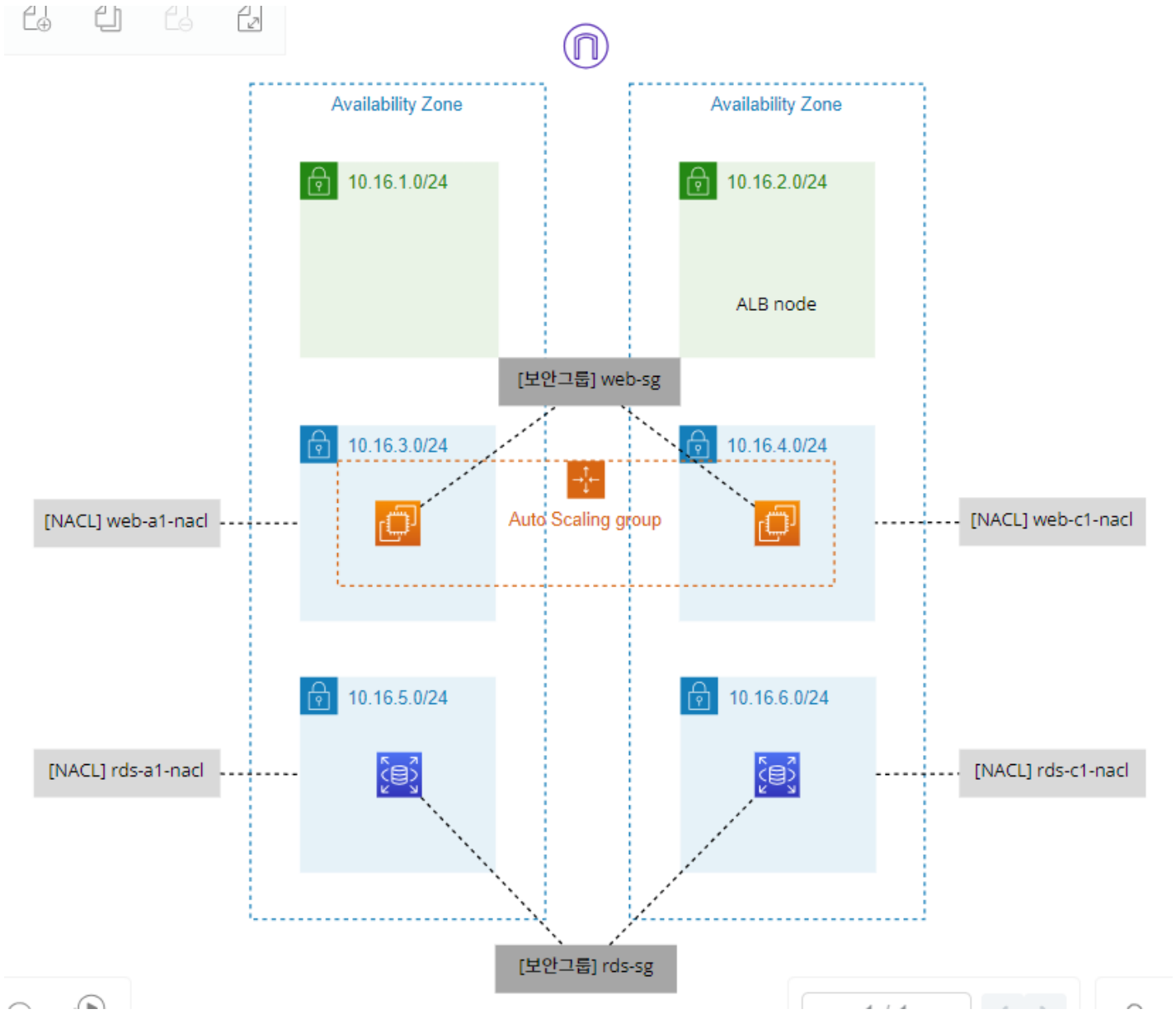
트래픽의 흐름은 아래와 같다.



실선은 Master instance에 수행하는 실제 입출력, 점선은 failover될 경우의 입출력 트래픽이다.

보안규칙

네트워크 ACL, 보안그룹의 배치는 다음과 같다.



[보안그룹] web-sg

- 인바운드

  X

- 아웃바운드

유형	포트 범위	대상	설명
MYSQL/Aurora	3306	sg-055006d46741e3e03 / rds-sg	[mysql] from web to rds

[보안그룹] rds-sg

- 인바운드

유형	포트 범위	소스	설명
MYSQL/Aurora	3306	sg-09fe4965f4f35451d / web-sg	[mysql] from web to rds

- 아웃바운드

  X

[NACL] web-a1-nacl, web-c1-nacl

- 인바운드

규칙 번호	유형	프로토콜	포트 범위	소스	허용/거부
3100	사용자 지정 TCP	TCP(6)	32768 - 60999	10.16.5.0/24	Allow
3200	사용자 지정 TCP	TCP(6)	32768 - 60999	10.16.6.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

- 아웃바운드

규칙 번호	유형	프로토콜	포트 범위	대상	허용/거부
3100	MySQL/Aurora(3306)	TCP(6)	3306	10.16.5.0/24	Allow
3200	MySQL/Aurora(3306)	TCP(6)	3306	10.16.6.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

[NACL] rds-a1-nacl, rds-c1-nacl

- 인바운드

규칙 번호	유형	프로토콜	포트 범위	소스	허용/거부
3100	MySQL/Aurora(3306)	TCP(6)	3306	10.16.3.0/24	Allow
3200	MySQL/Aurora(3306)	TCP(6)	3306	10.16.4.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

- 아웃바운드

규칙 번호	유형	프로토콜	포트 범위	대상	허용/거부
3100	사용자 지정 TCP	TCP(6)	32768 - 60999	10.16.3.0/24	Allow
3200	사용자 지정 TCP	TCP(6)	32768 - 60999	10.16.4.0/24	Allow
*	모든 트래픽	모두	모두	0.0.0.0/0	Deny

결론

이렇게 하여 AWS에 이중화된 웹서비스 인프라를 구축해보았다. 이번 프로젝트를 하면서 AWS VPC 네트워킹에 대해서 확실히 이해하였다.

추가적으로 깨달은 점은 다음과 같다.

- ALB를 사용하는 환경에서 health check를 하도록 트래픽을 허용해주지 않으면 target group에 있는 인스턴스들이 unhealthy가 되어 ALB가 로드밸런싱하지 않는다.
- 리눅스와 윈도우의 dynamic port는 다르게 설정되어있기 때문에 Network ACL에 응답에 대한 규칙을 설정할 때 이를 고려해야한다. (security group은 stateful이므로 응답을 고려하여 따로 규칙을 설정할 필요는 없다.)
- 같은 subnet내에서 통신할 경우, 라우팅 테이블을 따로 설정할 필요는 없다.
- RDS에 접근할 때 DNS 주소를 사용하므로, failover되어도 변경된 IP를 사용하여 새로운 DB 인스턴스에 접근가능

END