05. AWS 스토리지 서비스

5.1. 스토리지 개요

스토리지(storage)란 데이터를 보관하는 장소로, 우리가 사용하는 모든 저장 장치를 스토리지라고 할 수 있다. IT가 발전하면서 스토리지도 변화했으며, 클라우드 환경에서 사용되는 스토리지 역시 그 목적에 따라 다양하게 변화했다. AWS에서 사용되는 스토리지는 어떤 목적에서 쓰며, 그에 따라 어떤 특징이 있는지 알아본다.

5.2. 스토리지 서비스 및 주요 기능

5.2.1. 블록 스토리지

단일 스토리지 볼륨(volume)을 "블록"이라는 개별 단위로 분할해서 저장한다. 각 블록은 저장된 위치에 고유한 주소가 있기 때문에 서버에서 파일을 요청하면 블록들을 재구성하여 하나의 데이터로 서버에 전달한다. 클라우드 환경에서 블록 스토리지의 각 블록은 인스턴스에 위치하며, 마치 일반 컴퓨터에 하드디스크를 추가하여 C 드라이브, D 드라이브처럼 논리적으로 구분해서 사용하는 것과 같다.

5.2.2. 파일 스토리지

디렉터리(directory) 구조로 파일을 저장한다. 각 파일은 폴더에 종속되고 폴더 역시 다른 폴더에 종속되어 계층 구조를 이룬다. 따라서 파일을 찾으려면 어느 위치에 있는지 알아야 한다. 파일 스토리지는 개인용 컴퓨터와 서버에 일상적인 작업을 공유하여 사용할 수 있지만, 파일이 늘어나면 분류하거나 정리하는 데 시간이 점점 더 소요된다는 단점이 있다.

5.2.3. 객체 스토리지

각 데이터 조각을 가져와서 객체로 지정하고, 개별 단위로 저장한다. 파일 스토리지와 다르게 모든 객체는 중첩된 계층 구조 없이 단일한 평면적인 주소 공간에 저장된다. 평면 주소 공간에는 데이터 및 관련 메타데이터로 구성된 객체 고유 식별자가 있다. OS나 파일 시스템에 의존하지 않으면서 데이터를 저장하고 객체에 쉽게 접근할 수 있다.

객체 스토리지 시스템에서는 객체의 키(이름)만 알고 있으면 쉽고 빠르게 대상을 검색할 수 있다. 객체 스토리지 접근에는 HTTP 프로토콜 기반의 REST(REpersentational State Transfer) API(Application Programming Interface)를 사용한다.

객체 스토리지에 저장할 수 있는 데이터의 수와 파일 크기에 제한이 없으며 데이터 저장의 총 용량 역시 무제한에 가깝다고 할 수 있다. 오늘날 많은 사용자가 사용하는 이미지, 영상 등 복잡하고 대용량인 비정형 데이터를 효율적으로 처리할 수 있어 대부분의 스토리지 서비스로 사용된다.

5.2.4. 스토리지 선택 기준

클라우드의 스토리지는 어디에서나 접근할 수 있는 편의성, 대량의 데이터를 수용할 수 있는 확장성을 갖추고 있다. 따라서 스토리지를 도입할 때는 클라우드 스토리지의 종류별 특징을 충분히 이해하고 목적과 상황에 맞게 선택해야 한다.

5.3. Amazon EBS

5.3.1. EBS란

Amazon EBS(Elastic Block Store)는 EC2 인스턴스에 사용할 수 있는 블록 스토리지 볼륨을 제공하는 서비스이다. 블록 스토리지 특성을 이용한 저장 방식이므로 데이터를 일정한 크기의 블록으로 나누어 분산 저장하는데, 볼륨 위

에 파일 시스템을 생성하거나 하드디스크 드라이브 같은 블록 디바이스를 사용하는 것처럼 볼륨을 쓸 수 있다. 쉽게 운영 체제에 외장 하드디스크를 연결해서 데이터를 저장한다고 생각하면 된다. 또한 인스턴스에 연결된 EBS 볼륨의 구성을 동적으로 변형할 수 있다. 이처럼 EBS는 데이터베이스처럼 데이터 출입이 많은 서비스에 적합하다.

EBS 스토리지는 AWS 관리 콘솔에서 필요한 용량과 성능에 맞추어서 볼륨을 생성한 후 EC2 인스턴스에 연결하고 파일 시스템을 포맷한 후 사용한다. 이때 파일 시스템 포맷은 운영 체제에 따라 다르게 사용된다. 리눅스는 xfs 또는 ext4 유형이 주로 사용되며, 윈도우는 NTFS 포맷이 주로 사용된다. 포맷이 완료되면 해당 볼륨을 서버에 마운트한 후 데이터를 해당 디렉터리에 저장해서 사용한다.

5.3.2. EBS 특징

- 데이터 가용성: 단일 하드웨어 구성 요소의 장애 때문에 데이터가 손실되지 않도록 해당 가용 영역 내에서 자동으로 데이터를 복제한다.
- 데이터 지속성: EBS 볼륨은 인스턴스 수명과 관계없이 유지되는 비관계형 인스턴스 스토리지 이다.
- 데이터 안정성: Amazon EBS 암호화(encryption) 기능으로 암호화된 EBS 볼륨을 생성할 수 있으며, 암호화 표준 알고리즘(AES-256)을 사용한다.
- 데이터 백업: 모든 EBS 볼륨의 스냅샷(snapshot)(백업)을 생성하고, 다중 가용 영역에 중복 저장이 가능한 Amazon S3(Simple Storage Service)에 볼륨 내 데이터 사본을 백업할 수 있다.
- 데이터 확장성: 서비스를 중단할 필요 없이 볼륨 유형, 볼륨 크기, IOPS(Input/Output operations Per Second) 용량을 수정할 수 있다.

5.3.3. EBS 볼륨 유형

EBS 볼륨에는 크게 SSD와 HDD 유형이 있다. SSD는 메모리형 디스크를 사용하며, HDD는 플래터(platter) 디스크를 사용한다. 데이터를 빠르고 읽고 처리하는 능력이 좋은 SSD는 주로 서버의 운영 체제가 설치되는 OS 영역이나 일반 데이터베이스 보관용 스토리지 유형으로 사용된다. HDD는 속도와 상관없이 데이터 저장 용량이 많이 필요할 때 사용하므로 데이터 분석에 주로 활용된다.

처음 EBS 볼륨을 생성할 때 알맞게 잘 선택하면 좋겠지만, 사용하다가 성능이나 비용 때문에 볼륨 유형을 변경해야 하는 상황이 생기면 서비스를 중단할 필요 없이 다른 볼륨 유형으로 변경할 수 있다.

5.3.4. EBS 스냅샷

EBS 스토리지의 대표적인 특징인 EBS 스냅샷 기능은 말 그대로 특정 시점에 포인트를 찍어서 그 시점으로 되돌아갈 수 있는 지점을 만드는 기능이다. 증분식 백업 방식을 이용하여 마지막 스냅샷 이후 변경되는 데이터 블록만 기록하고 복제하므로 저장 비용과 시간도 효과적으로 절감할 수 있다.

변경된 부분만 증분되어 백업되기 때문에 특정 데이터를 삭제해도 어느 한 부분의 스냅샷만 가지고 있다면 복구할 수 있다. 스냅샷을 삭제할 때 다른 스냅샷에서 참조되지 않는 영역만 데이터에서 제거되기 때문이다.

스냅샷은 기본적으로 Amazon S3라는 스토리지 공간에 저장되어 여러 가용 영역에 자동으로 복제된다. 그리고 스냅샷은 다른 계정으로 공유할 수도 있고 스냅샷을 이용하여 손쉽게 다른 리전으로 복제할 수도 있다.

5.3.5. EFS란

Amazon EFS(Elastic File System)는 클라우드 환경과 온프레미스 환경에서 사용할 수 있는 완전 관리형 네트워크 파일 시스템이다. 여기에서 완전 관리형이란 클라우드에서 하드웨어 프로비저닝 유지 관리, 소프트웨어 구성, 모니터링, 복잡한 성능 조절 등 모든 것을 관리하기 때문에 사용자 입장에서는 별다른 관리가 필요 없다는 의미이다.

처음 파일 시스템을 생성한 후 서버에 연결하면 사용한 만큼 자동으로 스토리지 크기가 확장되고, 사용한 만큼만 비용을 지불하면 되기 때문에 사실상 용량 제한 없이 사용할 수 있다.

5.3.6. EFS 특징

EFS는 고성능 네트워크 파일 시스템으로, 여러 대의 컴퓨터가 네트워크상의 동일한 데이터에 접근해야 할 때 사용한다. NFS(Network File System) 표준 프로토콜 기반의 dussruf을 지원하고 있어 기존 다양한 애플리케이션과 유연하게 통합할 수 있고, 여러 컴퓨팅 인스턴스에서 동시에 사용할 수 있다. 또한 기존 NAS처럼 사용자 홈 디렉터리를 공유하여 애플리케이션을 개발하고, 테스트 환경의 다양한 웹 서비스와 콘텐츠 관리, 분석이나 미디어 업무에도 활용할 수 있다.

EFS 볼륨은 IOPS가 높고 용량이 매우 크기 때문에 처리량이 많고 대기 시간이 짧다. 또한 파일 시스템의 사용 증가에 따라 자동으로 용량 및 성능이 조정되는 탄력성을 제공하여 네트워크 스토리지의 용량 및 성능 부족을 걱정할 필요가 없다.

EFS는 EBS처럼 가용 영역 단위의 서비스가 아니라 가용 영역 전반에 걸쳐 사용할 수 있는 스토리지이므로, 가용 영역 장애를 고려한 디자인이 가능해서 전통적인 NAS보다 뛰어난 가용성을 가진다.

5.4. Amazon S3

5.4.1. S3 소개

Amazon S3(Simple Storage Service)는 AWS 서비스 중에서 EC2 서비스와 더불어 가장 오래되고, 기본이 되는 객체 스토리지 서비스이다. S3에 저장되는 데이터를 “객체”라고 하며, 이 객체 저장소를 “버킷(bucket)”이라고 한다. 그리고 객체에 대한 입출력은 HTTP 프로토콜로 하며, REST API를 이용하여 명령이 전달된다.

이 스토리지의 가장 큰 특징은 높은 내구성인데, 99.9999999999%의 내구성으로 디자인되어 있어 데이터 손실을 최소화하도록 규정되어 있다. 또한 데이터 저장 공간이 거의 무제한에 가까워 특별한 용량 제한 없이 데이터를 저장할 수 있다. 필요한 경우 다양한 스토리지 계층으로 데이터를 분류하여 데이터 저장 비용을 절감할 수도 있다.

S3의 객체는 기본적으로 웹 접속이 가능하기 때문에 간단한 정적 웹 콘텐츠를 S3에 올려 웹 서비스의 도움 없이 바로 웹 서비스가 가능하다. 그리고 S3는 보안 규정 준수 및 감시 기능을 제공하고 있어 데이터가 안전하게 저장되고 인증된 사용자만 접근할 수 있도록 구성한다. 필요한 경우 접근 권한 정책을 이용하여 해당 객체의 접근을 제어할 수 있다.

5.4.2. S3 구성 요소

- 버킷: 데이터 스토리지를 위한 S3의 기본 컨테이너이다. 객체는 반드시 버킷에 저장되어야 하며, 하나의 리전에서 생성된 후에는 버킷 이름과 리전을 변경할 수 없다.
- 객체: S3에 저장되는 기본 매체로, 객체 데이터와 객체 메타데이터로 구성되어 있다. 메타데이터는 객체를 설명하는 이름-값에 대한 하나의 쌍으로 존재한다. 객체를 저장할 때 사용자 정의 메타데이터를 지정할 수 있으며, 객체는 키(이름) 및 버전 ID를 이용하여 버킷내에서 고유하게 식별된다.
- 키: 버킷 내에서 객체의 고유한 식별자이다. 버킷 내 모든 객체는 고유한 하나의 키를 갖게 되며, S3는 “버킷 + 키 + 버전”과 객체 사이의 기본 데이터 맵으로 생각할 수 있다.
- S3 데이터 일관성: S3 버킷에 있는 객체에 대해 여러 서버로 데이터를 복제하여 고가용성 및 내구성을 구현하고 데이터 일관성 모델을 제공한다.

5.4.3. S3 특징

Amazon S3는 객체에 대해 빠르고 내구성과 가용성이 높은 키 기반의 접근성을 제공하여 데이터의 저장 및 검색에 특화된 객체 스토리지이다. S3는 하나의 리전 내 최소 세 개 이상의 물리적으로 분리된 가용 영역에 데이터를 복제해서 저장하므로 높은 내구성과 고가용성을 제공하며, 서버의 OS 도움 없이 객체별 접근이 가능하므로 데이터 저장 및 활용에 용이하다.

또한 Amazon S3는 동일 버킷 내 여러 개의 객체 변형을 보유하여 복구에 특화된 객체 스토리지이다. S3에 버전 관리 기능을 이용하면 버킷에 저장된 모든 객체를 보존하거나 검색 및 복원할 수 있으며, 의도하지 않은 사용자 작업 및 애플리케이션 장애에서 쉽게 복구될 수 있다.

S3는 AWS와 다른 서비스를 유기적으로 연동시켜 다양한 형태로 사용할 수도 있고 뛰어난 내구성 덕분에 최근에는 빅데이터, 머신 러닝을 위한 데이터를 저장하거나 기업의 데이터를 영구적으로 보관하는 등 데이터를 보관하거나 분석할 때 자주 사용된다.

5.4.4. S3 스토리지 클래스

S3는 여러 사용 사례에 맞게 설계된 다양한 스토리지 클래스를 제공한다. 자주 접근하는 데이터를 저장하는 S3 standard와 알 수 없거나 변화하는 접근 패턴이 있는 데이터를 저장하는 S3 intelligent-tiering, 데이터 수명은 길지만 자주 접근하지 않는 데이터를 저장하는 S3 standard-IA 및 S3 one zone-IA, 데이터 수명이 제한되어 추후 삭제 예정인 데이터 및 백업을 위한 S3 glacier 등이 있다.

5.4.5. S3 보안

S3 버킷 접근은 일반적으로 버킷을 생성할 때 사용한 버킷 이름을 포함하여 생성된 유일한 식별자를 기반으로 언제, 어디어서든 접근이 가능하다. 그런 점에서 보안은 가장 중요한 부분이다. 그렇기 때문에 AWS의 자격 증명 서비스인 IAM(Identity and Access Management)과 밀접하게 연관되어 있고, IAM으로 접속 사용자나 데이터 접근을 관리할 수 있다.

5.5. 다양한 AWS 스토리지 서비스 구성하기

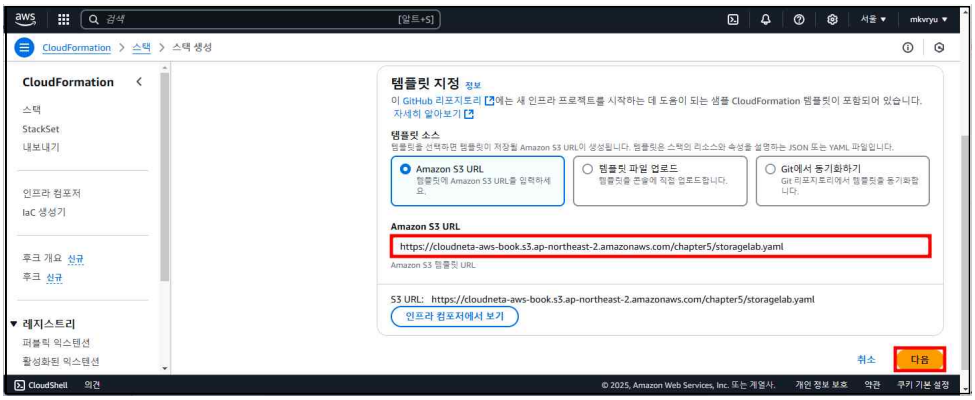
5.5.1. 실습에 필요한 기본 인프라 배포하기

실습 동작에 필요한 기본 인프라 자원은 AWS CloudFormation을 이용해 자동으로 배포한다. 먼저 스택을 생성한다.

AWS 관리 콘솔에서 서비스 => 관리 및 거버넌스 => CloudFormation 메뉴로 들어간 후 “스택 생성”을 누른다.

아래쪽에 있는 Amazon S3 URL에 다음 URL을 입력하고 “다음”을 누른다.

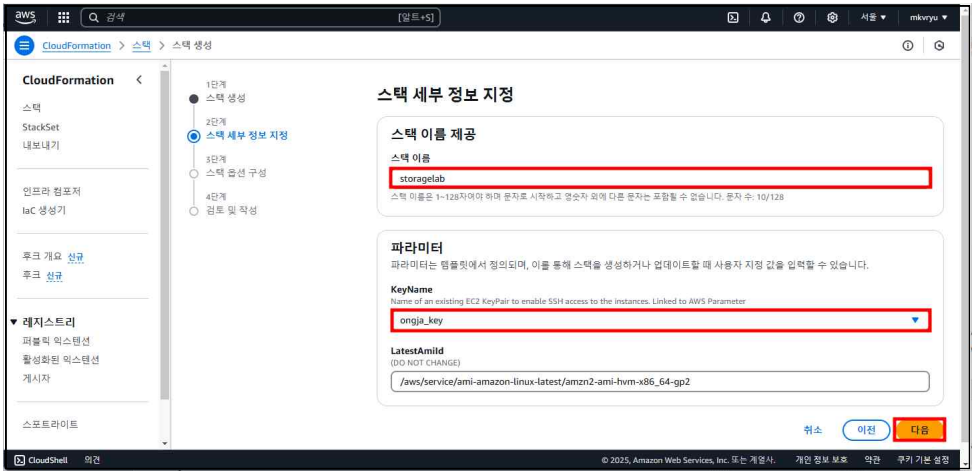
· https://cloudneta-aws-book.s3.ap-northeast-2.amazonaws.com/chapter5/storagelab.yaml



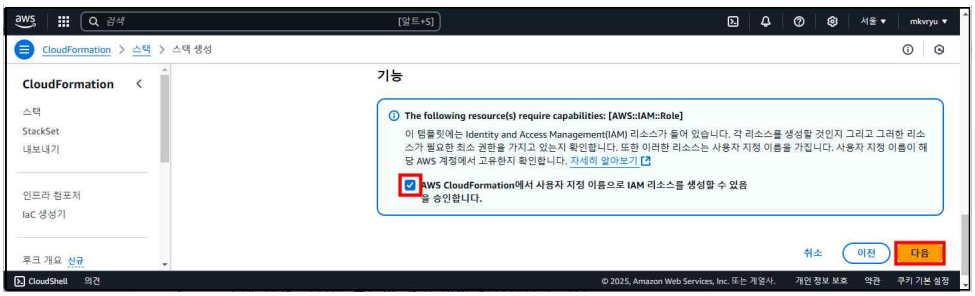
스택 세부 정보 지정 페이지에서 다음과 같이 설정하고 “다음”을 누른다.

· 스택 이름에 “storagelab” 입력

· KeyName은 사용자 키 페어 파일 선택

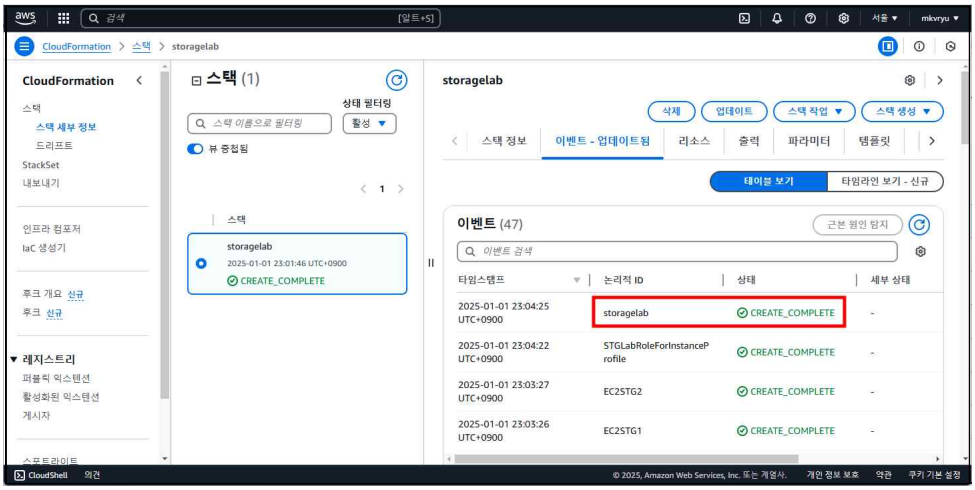


스택 옵션 구성은 별도의 설정은 없으나 가장 아래쪽에 있는 “AWS CloudFormation에서 사용자 지정 이름으로 IAM 리소스를 생성할 수 있음을 승인합니다.”에 체크하고 “다음”을 누른다.



검토 및 작성은 별도의 설정 없이 “전송”을 누른다.

AWS CloudFormation 기본 인프라를 배포하고 일정 시간(5분 이상 소요)이 지나 스택 상태가 “CREATE_COMPLETE”가 되면 모든 인프라 배포가 정상적으로 완료된 것이다.



AWS CloudFormation으로 생성된 기본 인프라 자원 정보는 다음과 같다.

생성 자원	이름	정보
VPC	CH5-VPC	10.40.0.0/16
인터넷 게이트웨이	CH5-IGW	
퍼블릭 라우팅 테이블	CH5-Public-RT	CH5-IGW 연결
서브넷 1(퍼블릭)	CH5-Public-Subnet1	CH5-PublicRT 연결[10.40.1.0/24]
서브넷 2(퍼블릭)	CH5-Public-Subnet2	CH5-PublicRT 연결[10.40.2.0/24]
IAM Role	STGLabInstanceRole	S3 Full 정책 연동
InstanceProfile	STGLabRoleForInstance	STGLabInstanceRole 연동
EC2 인스턴스 1	EC2-STG1	CH5-Subnet1 위치
EC2 인스턴스 2	EC2-STG2	CH5-Subnet2 위치
보안 그룹 1	CH5-SG	TCP 22/80/2049, ICMP 허용

이제 기본 인프라 자원이 생성되었으니 EC2 인스턴스(EC2-STG1, EC2-STG2)의 SSH 터미널에 접속하여 EBS 스토리지 기본 정보를 확인해 보자.

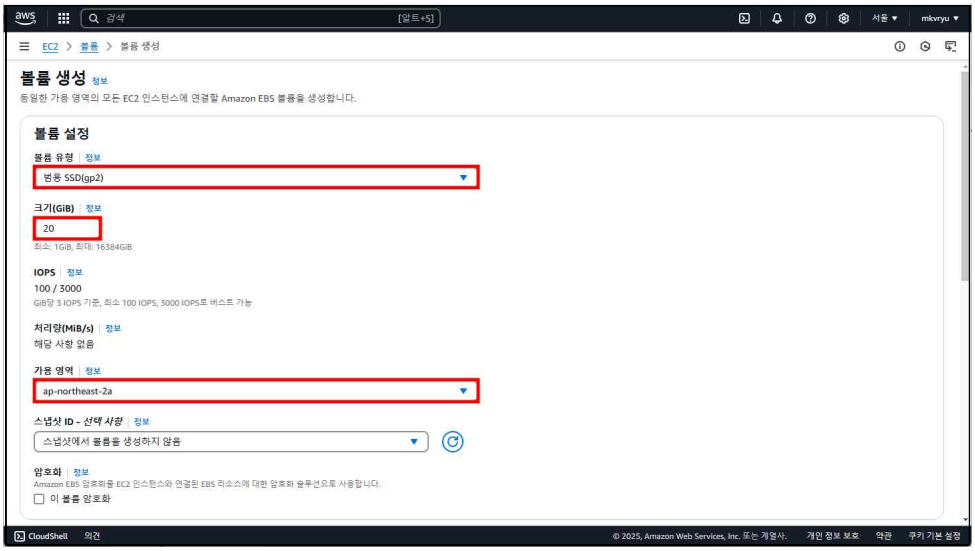
다음과 같이 명령어를 하나씩 입력하여 EBS 스토리지 기본 정보를 확인한다.

```
# STG1 SSH 터미널 접속
# df(disk free) 디스크 여유 공간 확인
[root@EC2-STG1 ~]# df -hT /dev/xvda1
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/xvda1     xfs   8.0G  2.2G  5.8G  28% /

# lsblk 사용 가능한 디스크 디바이스와 마운트 포인트(해당하는 경우) 확인
[root@EC2-STG1 ~]# lsblk
NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0   8G  0 disk
└─xvda1 202:1    0   8G  0 part /

# 디바이스의 UUID 확인
# /dev/xvda1: LABEL="/" UUID="고유정보" TYPE="xfs" PARTLABEL="Linux" PARTUUID="고유정보"
[root@EC2-STG1 ~]# blkid
/dev/xvda1:    LABEL="/"    UUID="5f28a6bd-3501-4b13-b8d6-57bdc3754cc7"    TYPE="xfs"    PARTLABEL="Linux"
PARTUUID="137eba45-3f25-40e8-a818-defe888e7cd6"

# 디바이스의 탑재 지점 확인
[root@EC2-STG1 ~]# cat /etc/fstab
#
UUID=5f28a6bd-3501-4b13-b8d6-57bdc3754cc7     /           xfs    defaults,noatime  1   1
```

5.5.2. EBS 스토리지를 추가로 생성한 후 사용하기

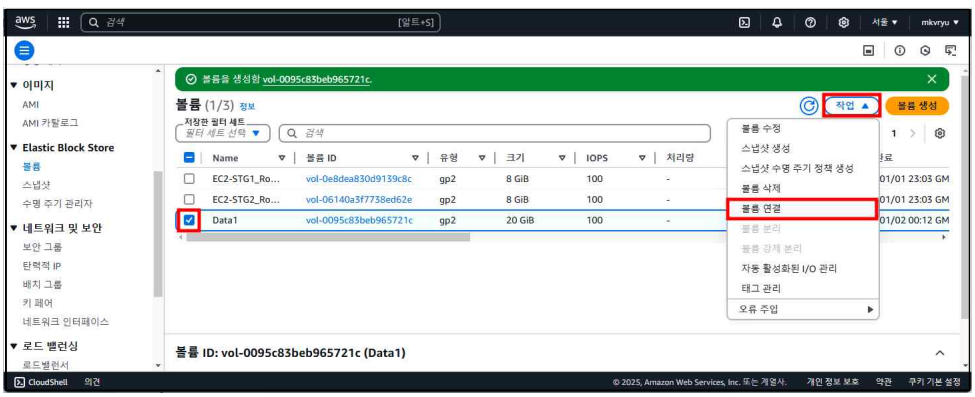
새로운 EBS 볼륨을 AZ 1(가용 영역, ap-northeast-2a) 영역에 생성한다. AWS 관리 콘솔에서 서비스 => 컴퓨팅 => EC2 => Elastic Block Store(이하 EBS) => 볼륨 => 볼륨 생성을 선택한다. 볼륨 생성 페이지가 나타나면 다음과 같이 설정한다. 별도의 언급이 없는 부분은 기본값을 유지한다.

· 볼륨 유형은 “범용 SSD(gq2)” 선택
· 크기(GiB)는 “20” 입력
· 가용 영역은 “ap-northeast-2a” 선택

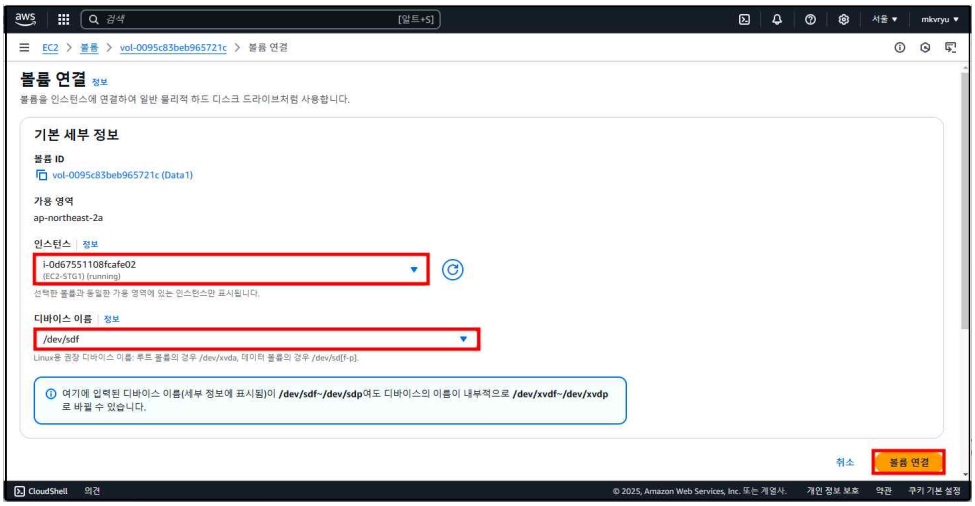


· 태그 추가를 누르고 키는 “Name”, 값은 “Data1” 입력
· 아래쪽 “볼륨 생성” 누르기

생성된 EBS 스토리지를 EC2 인스턴스에 연결한다. 추가된 Data1 볼륨(사용 가능 상태인지 확인)을 체크한 후 작업 => 볼륨 연결을 선택한다.



볼륨 연결에서 인스턴스는 “EC2-STG1” 선택하고 디바이스 이름은 “/dev/sdf”를 선택한 후 “볼륨 연결” 누르기



볼륨(디스크)을 추가할 때는 일반적으로 인스턴스를 중지한 후 연결해야 하지만 핫스왑(hot swap) 기능이 있기 때문에 동작(라이브) 상태에서 자동으로 추가된 볼륨을 인식한다.

Data 볼륨을 연결할 때 AZ 3(가용 영역, ap-northeast-2c)의 인스턴스(EC2-STG2)가 나오지 않는 이유는 EBS 볼륨은 해당 AZ만 사용할 수 있기 때문이다.

인스턴스(EC2-STG1)에 추가로 연결된 EBS 볼륨을 사용하도록 설정한다.

```
# STG1 SSH 터미널 접속
# 라이브 상태에서 디바이스 추가 확인
[root@EC2-STG1 ~]# lsblk
NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0   8G  0 disk
└─xvda1 202:1    0   8G  0 part /
xvdf    202:80   0  20G  0 disk

# 볼륨을 포맷하여 파일 시스템 생성
[root@EC2-STG1 ~]# mkfs -t xfs /dev/xvdf
meta-data=/dev/xvdf              isize=512    agcount=4, agsize=1310720 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=0 inobtcount=0
data     =                       bsize=4096   blocks=5242880, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0

# 디렉터리를 생성한 후 마운트
[root@EC2-STG1 ~]# mkdir /data
[root@EC2-STG1 ~]# mount /dev/xvdf /data

# 파일을 생성한 후 확인
[root@EC2-STG1 ~]# echo "EBS Test" > /data/memo.txt
```

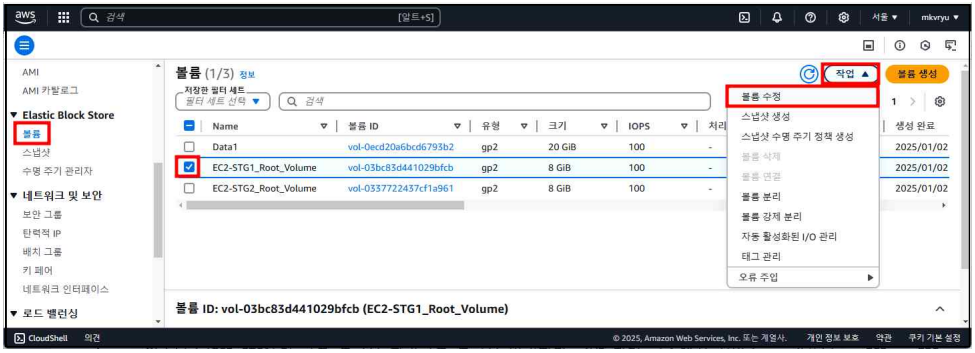
```
[root@EC2-STG1 ~]# cat /data/memo.txt
EBS Test

# 디바이스 확인
[root@EC2-STG1 ~]# lsblk
NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0   8G  0 disk
└─xvda1 202:1    0   8G  0 part /
xvdf    202:80   0  20G  0 disk /data

[root@EC2-STG1 ~]# df -hT /dev/xvdf
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/xvdf      xfs    20G  176M   20G   1% /data

# 재부팅 이후에도 볼륨 자동 탑재를 위한 fstab 설정
# 파일 시스템 정보를 저장하고 있으며, 부팅할 때 파일 안에 구성 값으로 자동 마운트되도록 하는 정보 확인
[root@EC2-STG1 ~]# cat /etc/fstab
#
UUID=5f28a6bd-3501-4b13-b8d6-57bdc3754cc7     /           xfs    defaults,noatime  1   1

# 장착할 볼륨 정보 확인
[root@EC2-STG1 ~]# blkid
/dev/xvda1:    LABEL="/"    UUID="5f28a6bd-3501-4b13-b8d6-57bdc3754cc7"    TYPE="xfs"    PARTLABEL="Linux"
PARTUUID="137eba45-3f25-40e8-a818-defe888e7cd6"
/dev/xvdf: UUID="bbd93bb7-8f1f-4edc-960f-8d0802afbbf2" TYPE="xfs"

# fstab 설정(앞 코드 결과에서 확인한 자신의 UUID 입력)
[root@EC2-STG1 ~]# echo "UUID=bbd93bb7-8f1f-4edc-960f-8d0802afbbf2 /data xfs defaults,nofail 0 2" >>
/etc/fstab

# fstab 설정 확인
[root@EC2-STG1 ~]# cat /etc/fstab
#
UUID=5f28a6bd-3501-4b13-b8d6-57bdc3754cc7 /     xfs defaults,noatime  1   1
UUID=bbd93bb7-8f1f-4edc-960f-8d0802afbbf2 /data xfs defaults,nofail   0   2
```

5.5.3. EBS 스토리지의 볼륨 크기를 변경하고 스냅샷 기능 확인하기

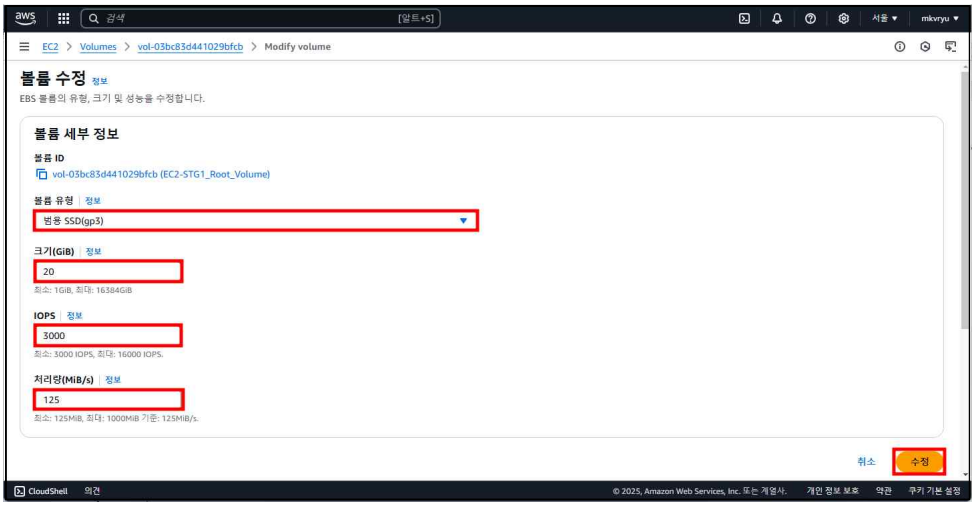
EBS 볼륨 크기 변경하기

서버를 운영할 때 로그 파일을 저장하면서 문제가 발생할 수 있는데, 저장 공간 대부분을 로그 파일이 차지해서 특정 서비스 접근이 차단되어 장애로 이루어지는 경우이다. 하지만 EBS 스토리지는 서버를 운영할 때 저장 공간 크기를 탄력적으로 조정하기 때문에 이런 서비스 장애에 대처할 수 있다. 다음 실습에서 장애 조치 상황이 아닌 EBS가 어떻게 탄력적으로 크기를 조정할 수 있는지 정도만 알아보자.

인스턴스(EC2-STG1)의 루트 볼륨을 확장하고 볼륨을 변경한다. AWS 관리 콘솔에서 서비스 => 컴퓨팅 => EC2 => EBS => 볼륨을 차례대로 선택한다. “EC2-STG1_Root_Volume”을 체크하고 작업 => 볼륨 수정을 선택하여 다음과 같이 설정을 변경한다. 별도로 언급이 없는 부분은 기본 값을 유지한다.



- 볼륨 유형은 “범용 SSD(gp3)” 선택
- 크기(GiB)는 “20” 입력
- IOPS는 “3000” 입력
- 처리량(MiB/s)은 “125” 입력
- “수정” 누르기



설정 후 동작 상태에서 다시 설정을 변경할 수 있는데, 3 ~ 10분 정도 시간이 소요된다.

볼륨 최적화(optimizing)를 시작하면 파일 시스템 크기를 조정할 수 있으므로 파티션 조정 => 파일 시스템 조정 순서로 파일 시스템을 확장한다.

```
# STG1 SSH 터미널 접속
# 현재 루트
# 볼륨 상태 확인
[root@EC2-STG1 ~]# lsblk
NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0  20G  0 disk
└─xvda1 202:1    0   8G  0 part /
```

```
xvdf    202:80   0  20G  0 disk /data

[root@EC2-STG1 ~]# df -hT /dev/xvda1
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/xvda1     xfs   8.0G  2.2G  5.8G  28% /

# growpart 명령어로 파티션 확장(파티션 조정)
[root@EC2-STG1 ~]# growpart /dev/xvda 1
CHANGED: partition=1 start=4096 old: size=16773087 end=16777183 new: size=41938911 end=41943007

[root@EC2-STG1 ~]# lsblk
NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0  20G  0 disk
└─xvda1 202:1    0  20G  0 part /
xvdf    202:80   0  20G  0 disk /data

# xfs_growfs 명령어로 볼륨의 파일 시스템 확장(파일 시스템 조정)
[root@EC2-STG1 ~]# xfs_growfs -d /
meta-data=/dev/xvda1             isize=512    agcount=4, agsize=524159 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=0, rmapbt=0
         =                       reflink=0    bigtime=0 inobtcount=0
data     =                       bsize=4096   blocks=2096635, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 2096635 to 5242363

[root@EC2-STG1 ~]# df -hT /dev/xvda1
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/xvda1     xfs    20G  2.3G   18G  12% /
```

EBS 스냅샷 기능 확인하기

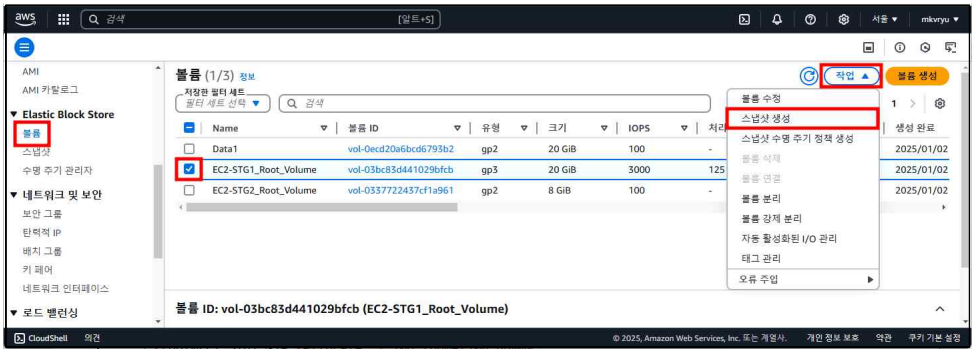
앞서 실습으로 EBS 크기를 탄력적으로 변경하여 데이터 저장 크기를 조정할 수 있다는 것을 확인했다. 하지만 지속적으로 크기를 확장하기는 쉽지 않으므로 백업으로 데이터를 분산 저장해야 한다. 이런 백업을 EBS에서는 스냅샷 기능으로 제공한다. 다음 실습으로 EBS의 스냅샷 기능을 확인해 본다.

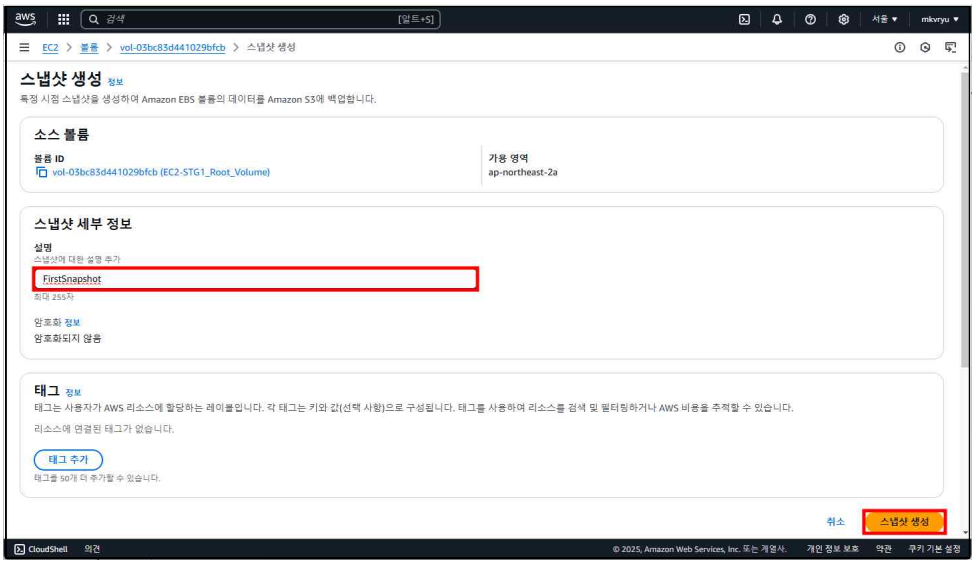
실습을 위해 가상 파일을 10G 크기로 만들어 테스트한다.

```
# STG1 SSH 터미널 접속
# 가상 파일 생성 후 확인
[root@EC2-STG1 ~]# fallocate -l 10G /home/10G.dummy
[root@EC2-STG1 ~]# df -hT /dev/xvda1
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/xvda1     xfs    20G   13G  7.8G  62% /
```

인스턴스(EC2-STG1)의 루트 볼륨을 스냅샷으로 백업한다. AWS 관리 콘솔에서 서비스 => 컴퓨팅 => EC2 => EBS => 볼륨에서 “EC2-STG1_Root_Volume”에 체크하고 작업 => 스냅샷 생성을 차례대로 선택한다. 스냅샷 생성 페이지에서

설명에 “FirstSnapshot”을 입력하고 “스냅샷 생성”을 누른다.



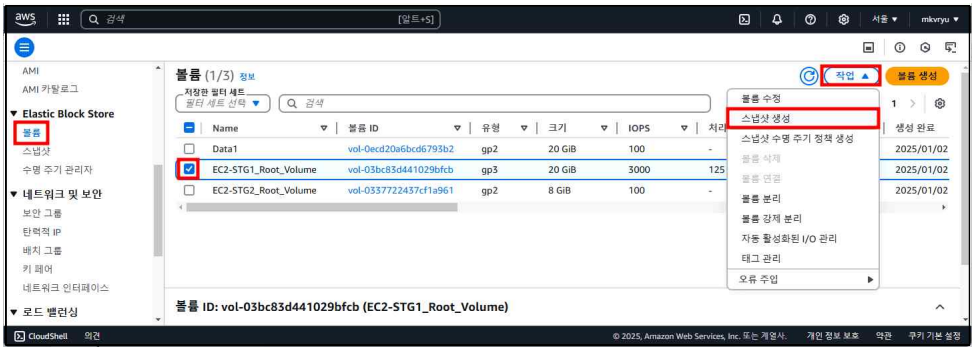


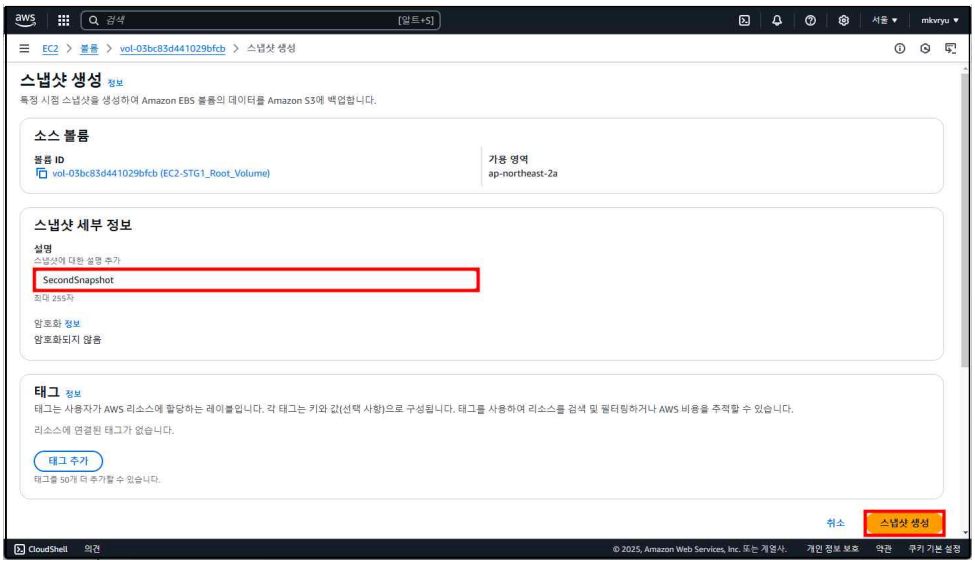
서비스 => 컴퓨팅 => EC2 => EBS => 스냅샷에 들어가 진행 상황을 확인한다(최초 백업은 볼륨 전체 백업이므로 다소 시간이 소요된다).

스냅샷 기능은 증분 백업하므로 실습을 위한 가상 파일을 추가로 생성해서 확인한다.

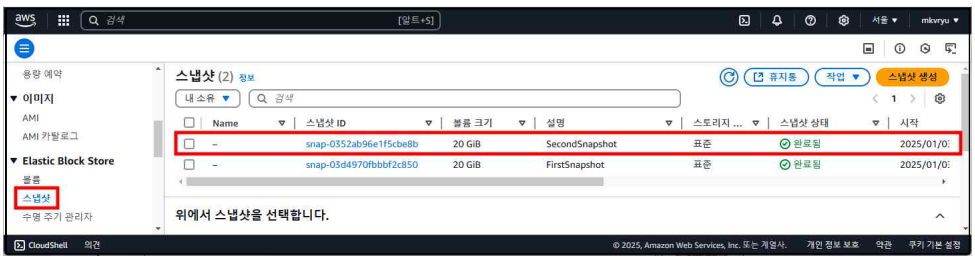
```
# STG1 SSH 터미널 접속
# 가상 파일 생성 후 확인
[root@EC2-STG1 ~]# fallocate -l 5G /home/5G.dummy
[root@EC2-STG1 ~]# df -hT /dev/xvda1
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/xvda1     xfs    20G   18G  2.8G  87% /
```

AWS 관리 콘솔에서 서비스 => 컴퓨팅 => EC2 => EBS => 볼륨에서 “EC2-STG1_Root_Volume”에 체크하고 작업 => 스냅샷 생성을 차례대로 선택한다. 스냅샷 생성 페이지에서 설명에 “SecondSnapshot”을 입력하고 “스냅샷 생성”을 누른다.





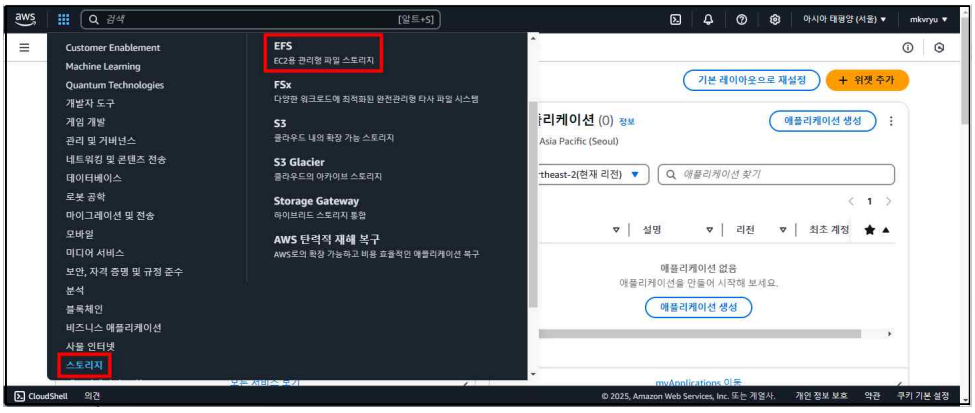
서비스 => 컴퓨팅 => EC2 => EBS => 스냅샷에 들어가 진행 상황을 확인한다(증분 백업은 변경된 볼륨만 백업하므로 다소 빠르게 진행된다).



5.5.4. EFS 스토리지 생성하고 사용하기

웹 서비스에 사용되는 루트 역할의 디렉터리를 EFS 스토리지로 마운트 하고 다른 AZ에서 같은 스토리지를 공유한다.

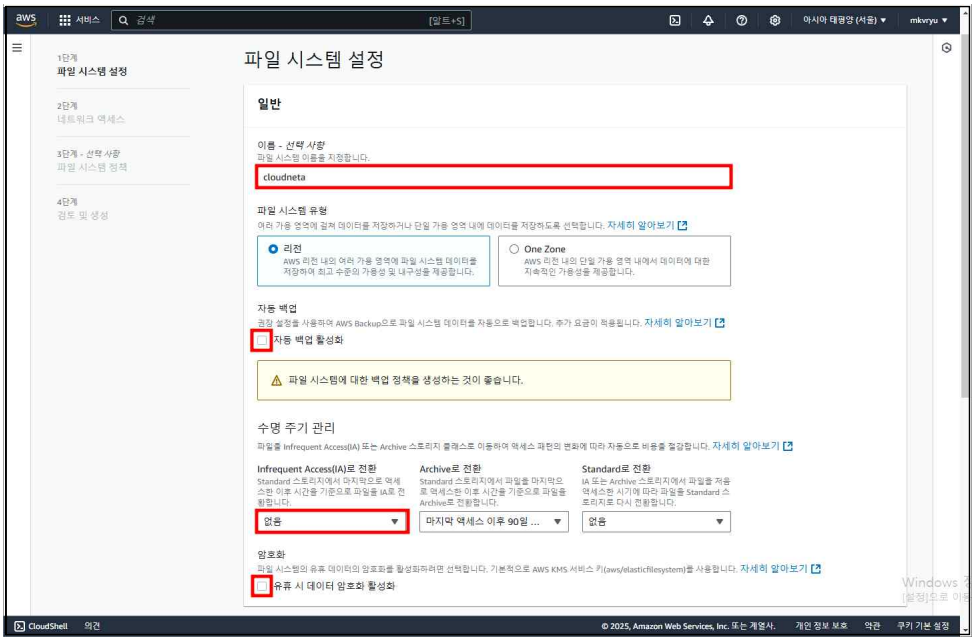
EFS 파일 시스템을 생성하기 위해 AWS 관리 콘솔의 서비스 => 스토리지 => EFS 누르기



파일 시스템 => 파일 시스템 생성 => 사용자 지정을 누른다.

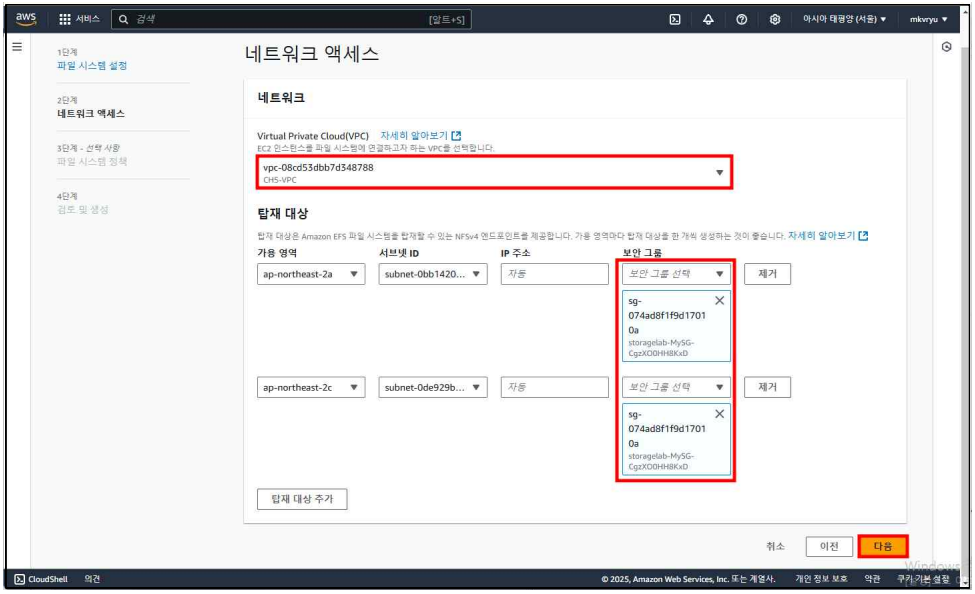
파일 시스템 설정 화면이 뜨면 다음과 같이 설정한다. 별도의 언급이 없는 부분은 기본값을 유지한다.

- 이름에 “cloudneta” 입력
- “자동 백업 활성화” 체크 해제
- Infrequent Access(IA)로 전환은 “없음” 선택
- “유휴 시 데이터 암호화 활성화” 체크 해제 후 “다음” 누르기



네트워트 액세스 설정 화면이 뜨면 다음과 같이 설정한다. 별도의 언급이 없는 부분은 기본값을 유지한다.

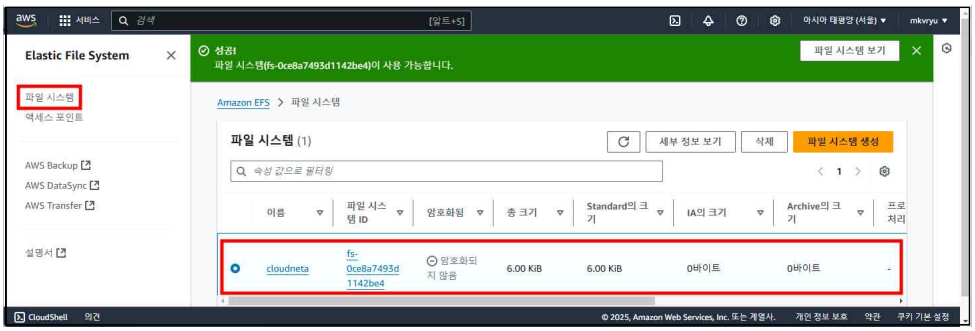
- 네트워크 Virtual Private Cloud(VPC)는 “CH5-VPC” 선택
- 보안 그룹에서 기존 default를 제거하고 “storagelab~~~” 선택
- “다음” 누르기



파일 시스템 정책 - 선택 사항 화면에서 “다음” 누르기

검토 및 생성 화면에서 “생성”을 누른다.

다시 “파일 시스템” 메뉴로 돌아가 EFS 스토리지가 생성되었는지 확인한다. 파일 시스템 ID는 SSH 터미널에서 바로 사용하므로 잘 기억해 둔다.



인스턴스(EC2-STG1)에 생성된 EFS를 사용하도록 연결한다.

```
# STG1 SSH 터미널 접속
# 현재 리전별 웹 서버 동작 확인
[root@EC2-STG1 ~]# curl localhost
<html><h1>Storage LAB - Web Server 1</h1></html>

# STG2 SSH 터미널 접속
# 현재 리전별 웹 서버 동작 확인
[root@EC2-STG2 ~]# curl localhost
<html><h1>Storage LAB - Web Server 2</h1></html>

# STG1 SSH 터미널 접속
# efs 디렉터리 생성
[root@EC2-STG1 ~]# mkdir /var/www/html/efs

# 자신의 EFS ID 확인 후 마운트
# 변수 지정, 반드시 자신의 파일 시스템 ID 확인 후 입력
[root@EC2-STG1 ~]# EFS=fs-0ce8a7493d1142be4(자신의-EFS-ID)
[root@EC2-STG1 ~]# echo $EFS
fs-0ce8a7493d1142be4
[root@EC2-STG1 ~]# mount -t efs -o tls $EFS:/ /var/www/html/efs

# EFS 마운트한 곳에 파일 생성 후 확인
[root@EC2-STG1 ~]# echo "<html><h1>Hello from Amazon EFS</h1></html>" > /var/www/html/efs/index.html
[root@EC2-STG1 ~]# curl localhost/efs/
<html><h1>Hello from Amazon EFS</h1></html>

# EFS Size 확인 - 사용자는 표시된 용량과 상관없이 실제 사용한 용량만큼 비용 지불
[root@EC2-STG1 ~]# df -hT ¦ grep efs
127.0.0.1:/    nfs4      8.0E     0  8.0E   0% /var/www/html/efs

# EFS DNS 주소를 조회할 때 표시되는 IP는 각 AZ에 속한 인터페이스 IP 주소
# (dig +short 자신의-EFS-ID.efs.ap-northeast-2.amazonaws.com)
[root@EC2-STG1 ~]# dig +short $EFS.efs.ap-northeast-2.amazonaws.com
10.40.1.123
```

인스턴스(EC2-STG2)에 EFS를 사용하도록 설정한다.

```
# STG2 SSH 터미널 접속
# efs 디렉터리 생성
[root@EC2-STG2 ~]# mkdir /var/www/html/efs

# 자신의 EFS ID 확인 후 마운트(mount -t efs -o tls 자신의-EFS-ID:/ /var/www/html/efs)
# 변수 지정, 반드시 자신의 파일 시스템 ID 확인 후 입력
[root@EC2-STG2 ~]# EFS=fs-0ce8a7493d1142be4(자신의-EFS-ID)
[root@EC2-STG2 ~]# echo $EFS
fs-0ce8a7493d1142be4
[root@EC2-STG2 ~]# mount -t efs -o tls $EFS:/ /var/www/html/efs
```

```
# EFS 마운트 확인
[root@EC2-STG2 ~]# curl localhost/efs/
<html><h1>Hello from Amazon EFS</h1></html>

# EFS Size 확인 - 사용자는 표시된 용량과 상관없이 실제 사용한 용량만큼 비용 지불
[root@EC2-STG2 ~]# df -hT ¦ grep efs
127.0.0.1:/    nfs4      8.0E     0  8.0E   0% /var/www/html/efs

# EFS DNS 주소를 조회할 때 표시되는 IP는 각 AZ에 속한 인터페이스 IP 주소
# (dig +short 자신의-EFS-ID.efs.ap-northeast-2.amazonaws.com)
[root@EC2-STG2 ~]# dig +short $EFS.efs.ap-northeast-2.amazonaws.com
10.40.2.118
```

EFS 주소를 조회할 때 다른 IP로 표시되는 이유는 EFS는 AZ당 하나의 Mount Target(네트워크 인터페이스)를 사용하기 때문이다.

EFS를 이용하여 각 인스턴스의 파일을 공유한다. 각 인스턴스의 SSH 터미털에 맞는 명령어를 잘 입력해야 한다.

```
# STG1 SSH 터미널 접속
# 인스턴스(EC2-STG1)에서 파일 생성
[root@EC2-STG1 ~]# for i in {1..100}; do touch /var/www/html/efs/deleteme.$i; done;
```

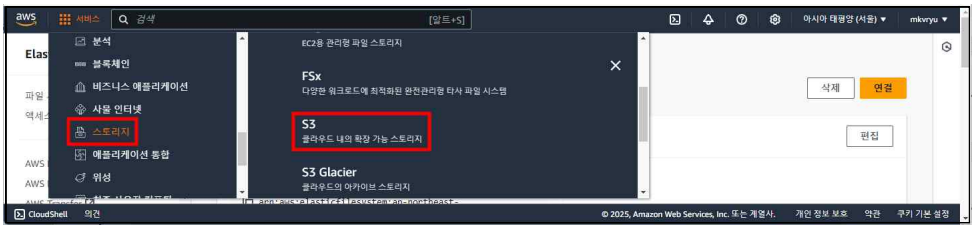
```
# STG2 SSH 터미널 접속
# 인스턴스(EC2-STG2)에서 파일 생성
[root@EC2-STG2 ~]# ls /var/www/html/efs
deleteme.1 deleteme.2 deleteme.30 deleteme.41 deleteme.52 deleteme.63 deleteme.74 deleteme.85 deleteme.96
deleteme.10 deleteme.20 deleteme.31 deleteme.42 deleteme.53 deleteme.64 deleteme.75 deleteme.86 deleteme.97
deleteme.100 deleteme.21 deleteme.32 deleteme.43 deleteme.54 deleteme.65 deleteme.76 deleteme.87 deleteme.98
deleteme.11 deleteme.22 deleteme.33 deleteme.44 deleteme.55 deleteme.66 deleteme.77 deleteme.88 deleteme.99
deleteme.12 deleteme.23 deleteme.34 deleteme.45 deleteme.56 deleteme.67 deleteme.78 deleteme.89 index.html
deleteme.13 deleteme.24 deleteme.35 deleteme.46 deleteme.57 deleteme.68 deleteme.79 deleteme.9
deleteme.14 deleteme.25 deleteme.36 deleteme.47 deleteme.58 deleteme.69 deleteme.8 deleteme.90
deleteme.15 deleteme.26 deleteme.37 deleteme.48 deleteme.59 deleteme.7 deleteme.80 deleteme.91
deleteme.16 deleteme.27 deleteme.38 deleteme.49 deleteme.6 deleteme.70 deleteme.81 deleteme.92
deleteme.17 deleteme.28 deleteme.39 deleteme.5 deleteme.60 deleteme.71 deleteme.82 deleteme.93
deleteme.18 deleteme.29 deleteme.4 deleteme.50 deleteme.61 deleteme.72 deleteme.83 deleteme.94
deleteme.19 deleteme.3 deleteme.40 deleteme.51 deleteme.62 deleteme.73 deleteme.84 deleteme.95

# 인스턴스(EC2-STG2)에서 생성된 파일 삭제
[root@EC2-STG2 ~]# rm -rf /var/www/html/efs/deleteme*.*
```

```
# STG1 SSH 터미널 접속
# 인스턴스(EC2-STG1)에서 삭제된 파일 생성
[root@EC2-STG1 ~]# ls /var/www/html/efs
index.html
```

인스턴스 간 공유 스토리지를 이용하여 동일한 파일 정보를 확인해 보았다. 이처럼 EFS 스토리즈는 여러 가용 영역에 걸쳐 실행되는 웹 서비스에 정적 및 동적 콘텐츠를 공유하고, 새로운 파일이나 변경된 콘텐츠를 즉시 반영하여 사용해야 할 때 매우 유용하다.

5.5.5. Public 3S 스토리지로 외부 접근 확인하기

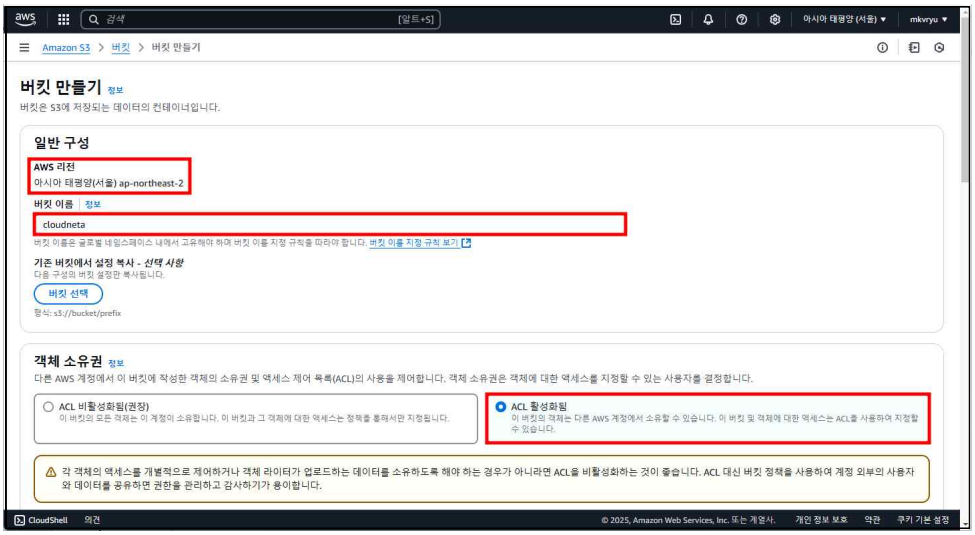
AWS 관리 콘솔의 서비스 => 스토리지 => S3 누르기



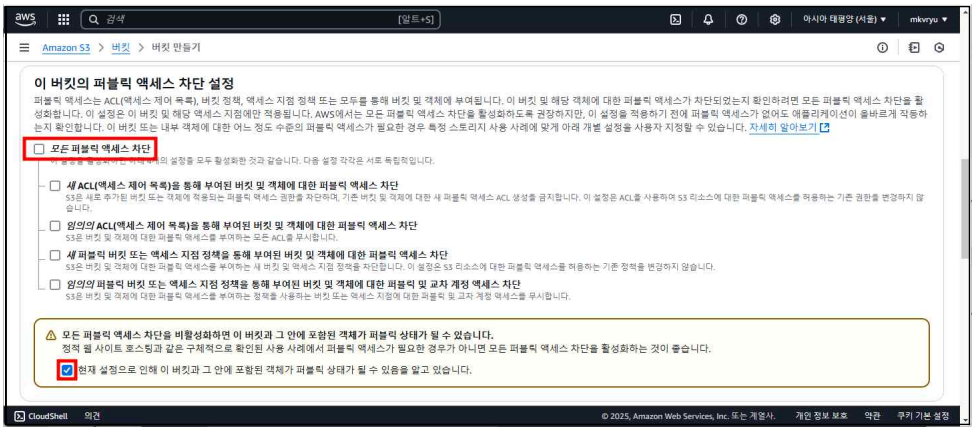
"버킷 만들기" 누르기

설정 화면이 뜨면 다음과 같이 설정한다. 별도로 언급이 없은 부분은 기본값을 유지한다.

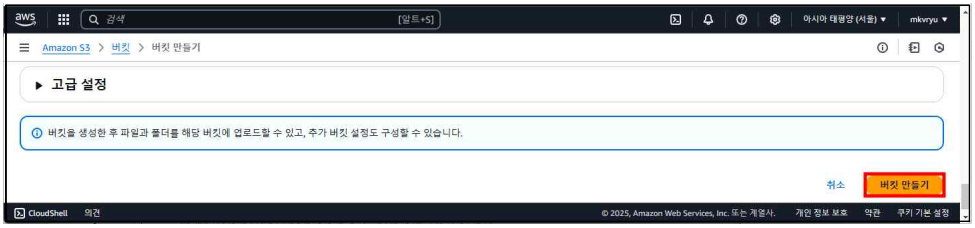
- 일반 구성의 버킷 이름에 "cloudneta-beas" 입력
- 객체 소유권은 "ACL 활성화됨" 선택



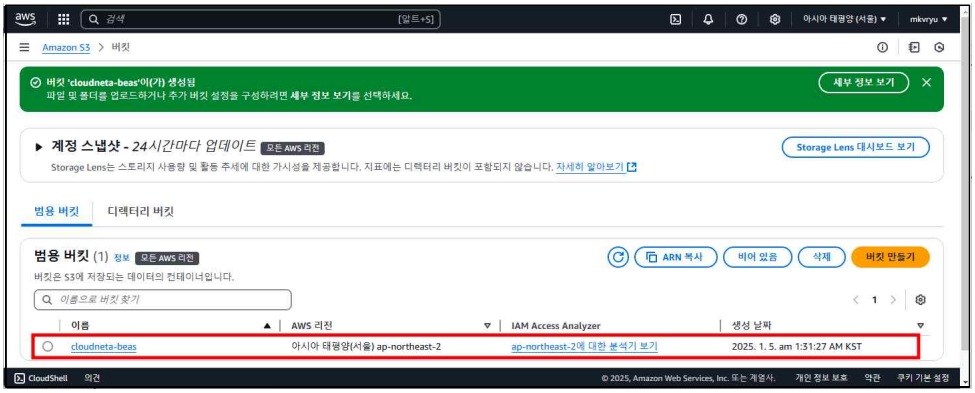
- "모든 퍼블릭 액세스 차단"의 체크를 해제하고 "현재 설정으로 인해 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있음을 알고 있습니다."에 체크



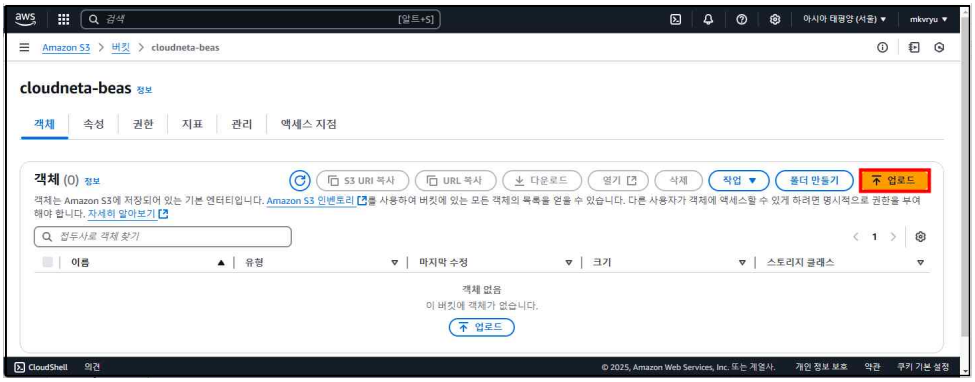
“버킷 만들기”를 누른다.



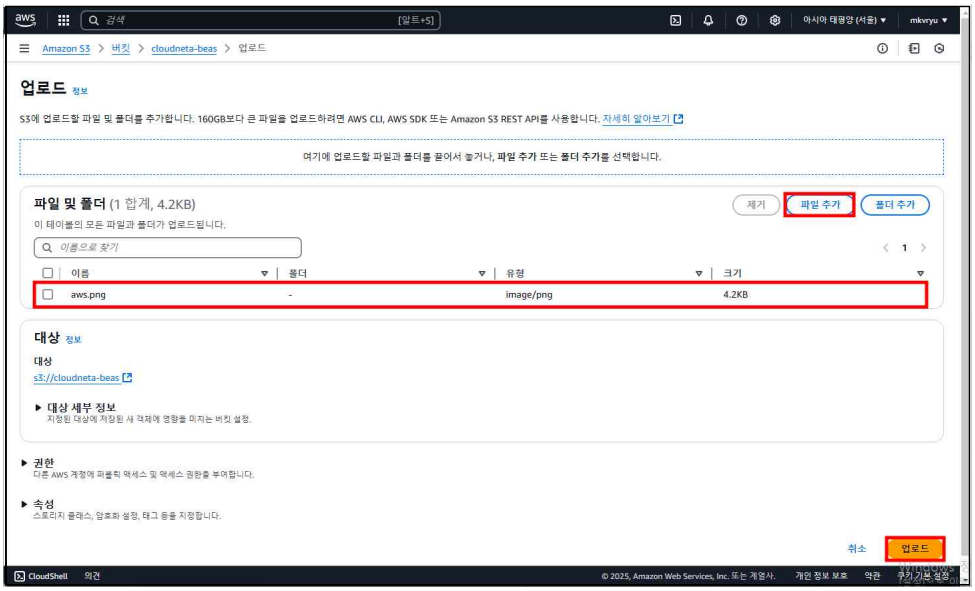
생성된 버킷을 확인한다.



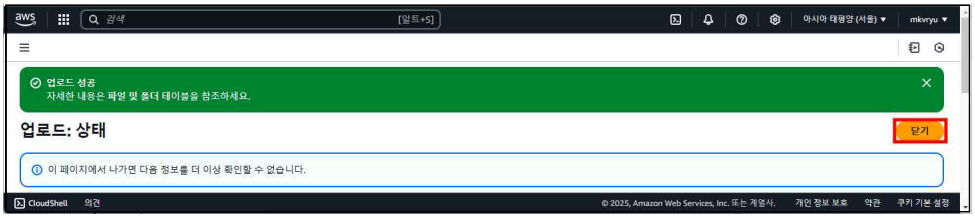
생성된 버킷 이름을 누르고 “업로드”를 누른다.



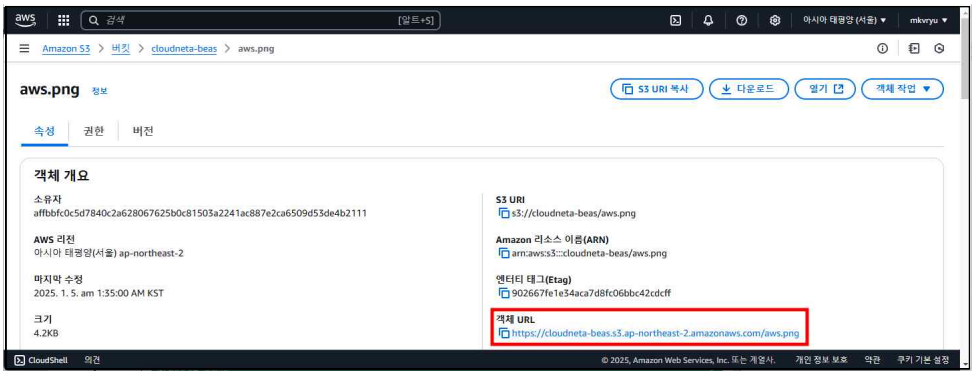
“파일 추가”를 누르고, 임의의 테스트용 이미지 파일을 선택한 후 “업로드”를 누른다.



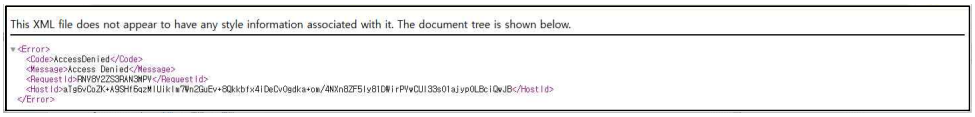
파일 업로드가 완료되면 “닫기”를 누른다.



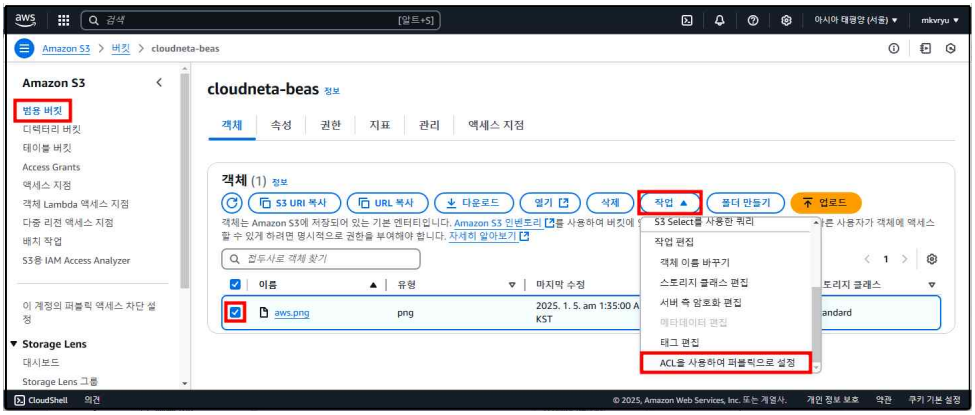
업로드된 파일을 선택하면 다음 그림과 같이 객체 URL을 확인할 수 있다.



업로드된 파일을 선택한 후 표시된 객체 URL로 웹에 접근하면 다음 그림과 같이 파일 정보가 표시되지 않는다. 버킷에 업로드된 파일은 기본적으로 외부에서 접근하는 것이 차단되어 있기 때문이다.

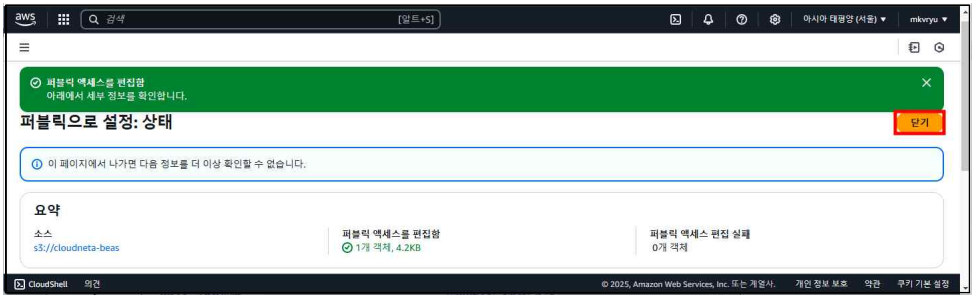


외부에서 접근한 후 사용하려면 다음과 같은 설정이 추가로 필요하다. 업로드된 파일을 선택한 후 작업 => ACL을 사용하여 퍼블릭으로 설정을 누른다.



“퍼블릭으로 설정”을 누른다.

퍼블릭 액세스로 변경되면 “닫기”를 누른다.

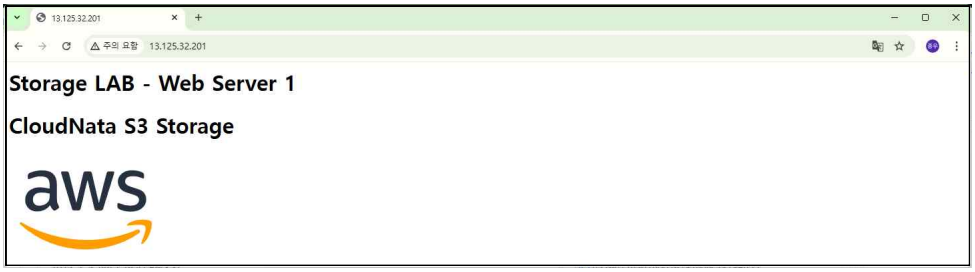


업로드된 파일을 선택한 후 객체 개요 페이지에서 표시된 객체 URL로 웹에 접근하면 다음 그림과 같이 파일 정보가 정상적으로 표시되는지 확인한다.

인스턴스(EC2-STG1)에 웹 접근을 할 때 S3 스토리지에 업로드된 파일을 사용하도록 웹 서버의 index.html 파일을 다음과 같이 설정한다.

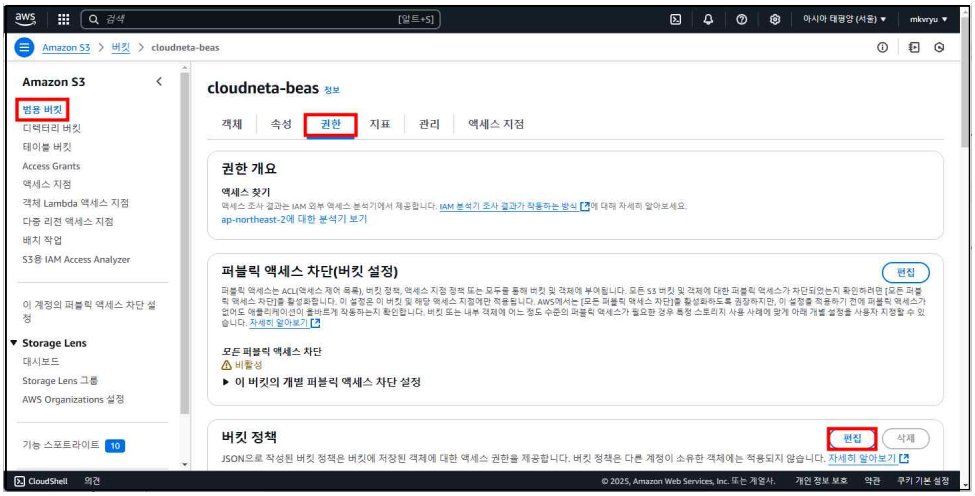
```
# STG1 SSH 터미널 접속
# 인스턴스(EC2-STG1)에서 index.html 파일 생성
[root@EC2-STG1 ~]# cat <<EOF>> /var/www/html/index.html
> <html>
> <body>
> <h1>CloudNata S3 Storage</h1>
// 앞서 업로드한 테스트 파일 객체 주소 입력
> <img src="https://cloudneta-beas.s3.ap-northeast-2.amazonaws.com/aws.png">
> </body>
> </html>
> EOF

[root@EC2-STG1 ~]# cat /var/www/html/index.html
<html><h1>Storage LAB - Web Server 1</h1></html>
<html>
<body>
<h1>CloudNata S3 Storage</h1>
<img src="https://cloudneta-beas.s3.ap-northeast-2.amazonaws.com/aws.png">
</body>
</html>
```

인스턴스(EC2-STG1) 웹 서버에 접근해서 표시되는 사진이 S3에 업로드한 파일임을 확인한다.



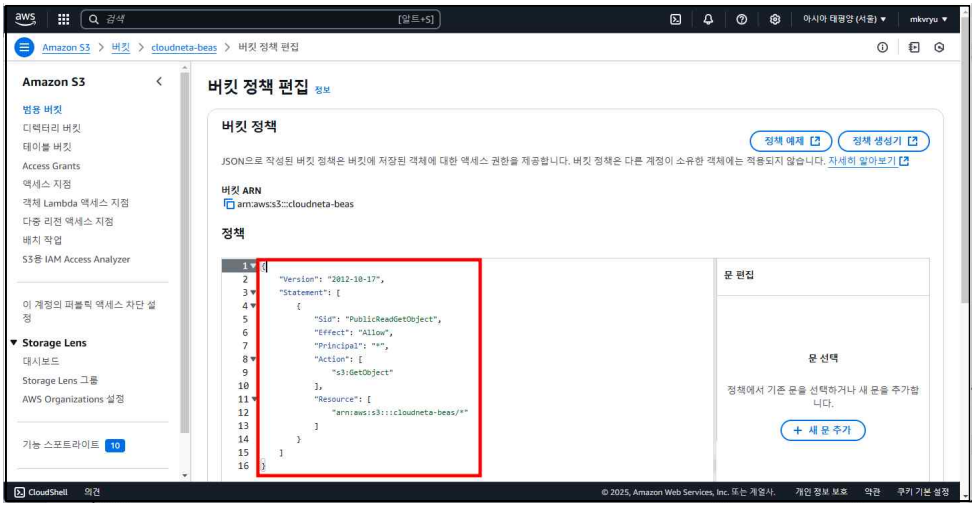
이제 객체마다 추가 설정 없이도 외부에서 접근하면 자동으로 사용할 수 있도록 설정한다.

앞서 생성한 버킷을 선택하고 “권한 탭”을 클릭하고 버킷 정책에서 편집을 누른다.

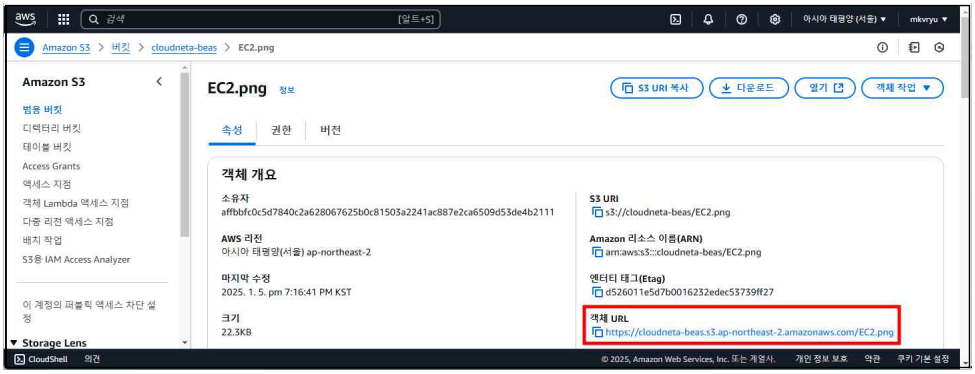


버킷 정책에 다음 정책을 추가하고 아래쪽의 “변경 사항 저장”을 누른다.

```
# S3의 보안 정책을 이용하여 외부 접근을 허용하는 설정
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::cloudneta-beas/*"
            ]
        }
    ]
}
```



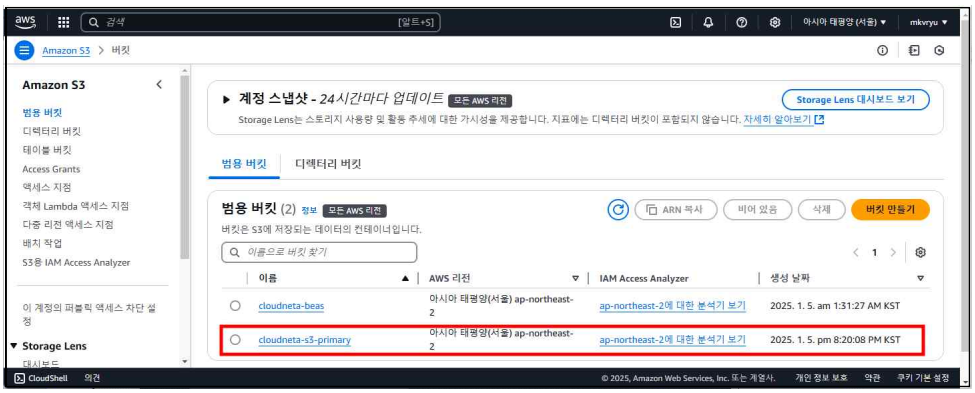
버킷 메뉴로 돌아와 테스트용 이미지 파일을 새롭게 업로드한 수 객체 URL을 클릭하면 외부에서 별다른 설정없이 접근할 수 있다.



5.5.6. Private S3 스토리지의 제한된 접근 및 데이터 백업하기

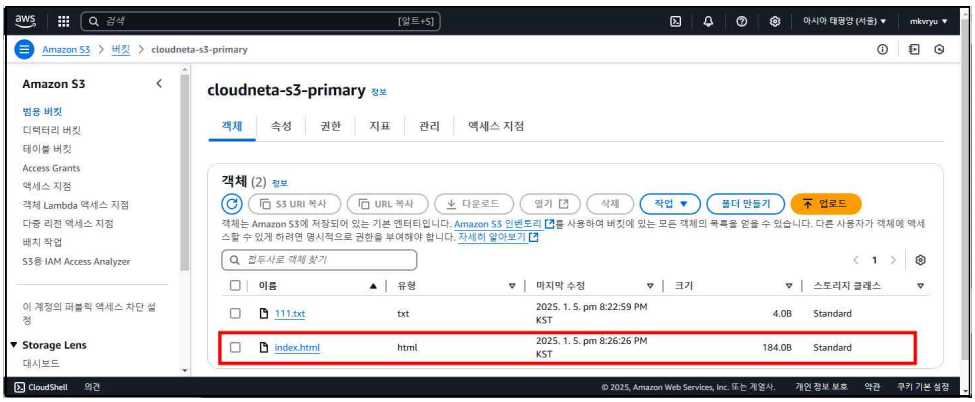
AWS CLI를 이용하여 Private S3 버킷을 생성한 후 사용할 수 있게 다음과 같이 설정한다.

```
# STG1 SSH 터미널 접속
# 기존 S3 조회
[root@EC2-STG1 ~]# aws s3 ls
2025-01-05 10:29:09 cloudneta-beas

# S3 버킷 생성(aws s3 mb s3://자신의 고유한 버킷 이름) - 버킷 생성 수 S3 웹 콘솔에서 확인
[root@EC2-STG1 ~]# aws s3 mb s3://cloudneta-s3-primary
make_bucket: cloudneta-s3-primary
```




```
# 버킷 이름을 변수에 지정(MyS3=버킷)
[root@EC2-STG1 ~]# MyS3=cloudneta-s3-primary
[root@EC2-STG1 ~]# echo $MyS3
cloudneta-s3-primary

# 파일 생성 후 S3에 업로드
```

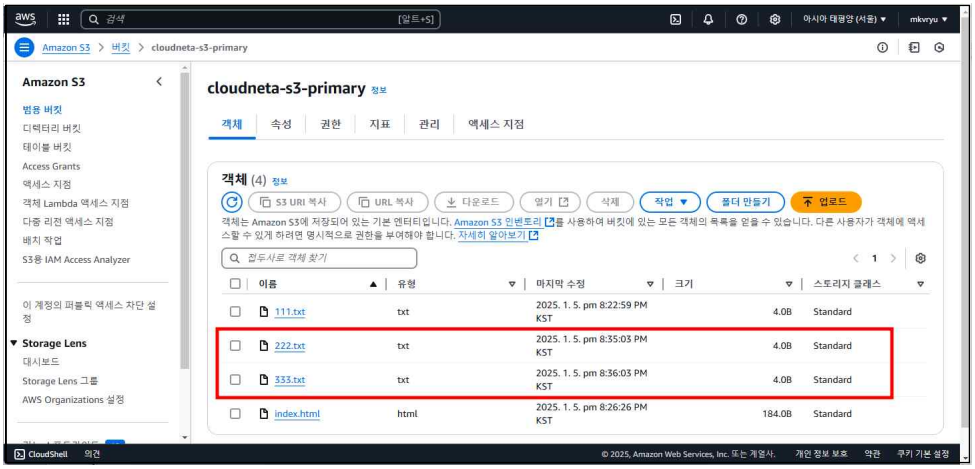
```
[root@EC2-STG1 ~]# echo "111" > /var/www/html/111.txt
[root@EC2-STG1 ~]# ls /var/www/html
111.txt  efs  index.html
[root@EC2-STG1 ~]# aws s3 cp /var/www/html/111.txt s3://$MyS3
upload: ../var/www/html/111.txt to s3://cloudneta-s3-primary/111.txt
[root@EC2-STG1 ~]# aws s3 ls s3://$MyS3
2025-01-05 11:22:59          4 111.txt

# 웹 디렉터리(하위 포함)를 S3 버킷에 업로드(수동 백업) 후 확인
[root@EC2-STG1 ~]# aws s3 sync --delete /var/www/html s3://$MyS3
upload: ../var/www/html/index.html to s3://cloudneta-s3-primary/index.html
[root@EC2-STG1 ~]# aws s3 ls s3://$MyS3 --recursive
2025-01-05 11:22:59          4 111.txt
2025-01-05 11:26:26        184 index.html
```



다음 코드를 입력하여 파일으로 업로드한 후 버킷을 조회하고 로그를 확인한다. 확인한 후에는 Ctrl + C를 눌러서 빠져나온다.

```
# STG1 SSH 터미널 접속
# crontab 내용 추가
[root@EC2-STG1 ~]# cat <<EOF>> /etc/crontab
> */1 * * * * root aws s3 sync --delete /var/www/html s3://$MyS3
> EOF

[root@EC2-STG1 ~]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

```
*/1 * * * * root aws s3 sync --delete /var/www/html s3://cloudneta-s3-primary

# 적용 및 추가 파일 생성
[root@EC2-STG1 ~]# systemctl restart crond
[root@EC2-STG1 ~]# echo "222" > /var/www/html/222.txt
[root@EC2-STG1 ~]# echo "333" > /var/www/html/333.txt
```

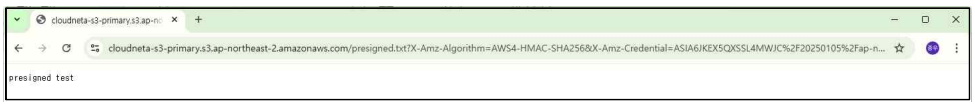



```
# 실시간 버킷 조회 후 로그 확인
[root@EC2-STG1 ~]# while true; do aws s3 ls s3://$MyS3; date; echo "---[S3 l3]---"; sleep 3; done
2025-01-05 11:22:59          4 111.txt
2025-01-05 11:35:03          4 222.txt
2025-01-05 11:36:03          4 333.txt
2025-01-05 11:26:26        184 index.html
Sun Jan  5 11:38:42 UTC 2025
---[S3 l3]---
2025-01-05 11:22:59          4 111.txt
2025-01-05 11:35:03          4 222.txt
2025-01-05 11:36:03          4 333.txt
2025-01-05 11:26:26        184 index.html
Sun Jan  5 11:38:46 UTC 2025
---[S3 l3]---
^C
You have new mail in /var/spool/mail/root

[root@EC2-STG1 ~]# tail -f /var/log/cron
Jan  5 11:34:01 EC2-STG1 CROND[18108]: (root) CMD (aws s3 sync --delete /var/www/html s3://cloudneta-s3-primary)
Jan  5 11:34:22 EC2-STG1 crond[3036]: (CRON) INFO (Shutting down)
Jan  5 11:34:22 EC2-STG1 crond[18142]: (CRON) INFO (RANDOM_DELAY will be scaled with factor 87% if used.)
Jan  5 11:34:22 EC2-STG1 crond[18142]: (CRON) INFO (running with inotify support)
Jan  5 11:34:22 EC2-STG1 crond[18142]: (CRON) INFO (@reboot jobs will be run at computer's startup.)
Jan  5 11:35:01 EC2-STG1 CROND[18281]: (root) CMD (aws s3 sync --delete /var/www/html s3://cloudneta-s3-primary)
Jan  5 11:36:01 EC2-STG1 CROND[18358]: (root) CMD (aws s3 sync --delete /var/www/html s3://cloudneta-s3-primary)
Jan  5 11:37:01 EC2-STG1 CROND[18438]: (root) CMD (aws s3 sync --delete /var/www/html s3://cloudneta-s3-primary)
Jan  5 11:38:01 EC2-STG1 CROND[18506]: (root) CMD (aws s3 sync --delete /var/www/html s3://cloudneta-s3-primary)
Jan  5 11:39:01 EC2-STG1 CROND[18606]: (root) CMD (aws s3 sync --delete /var/www/html s3://cloudneta-s3-primary)
^C
```

생성된 Private S3 버킷은 일반적으로 외부 접근이 불가능하지면 Pre-sign URL 기능을 이용하여 특정한 사용자에게 제한된 시간 동안 외부 접근을 허용할 수 있다.

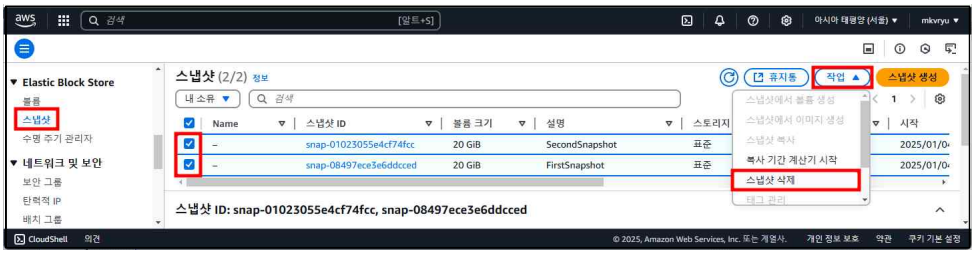
```
# STG1 SSH 터미널 접속
# 테스트 파일 생성 후 S3에 업로드
[root@EC2-STG1 ~]# echo "presigned test" > /var/www/html/presigned.txt
[root@EC2-STG1 ~]# aws s3 cp /var/www/html/presigned.txt s3://$MyS3
upload: ../var/www/html/presigned.txt to s3://cloudneta-s3-primary/presigned.txt

# 객체 URL로 직접 접근은 불가하나 Pre-sign URL 생성
[root@EC2-STG1 ~]# aws s3 presign s3://$MyS3/presigned.txt --expires-in 300
https://cloudneta-s3-primary.s3.ap-northeast-2.amazonaws.com/presigned.txt?X-Amz-Algorithm=AWS4-HMAC-SHA
256&X-Amz-Credential=ASIA6JKEX5QXSSL4MWJC%2F20250105%2Fap-northeast-2%2Fs3%2Faws4_request&X-Amz-Date=202
50105T114924Z&X-Amz-Expires=300&X-Amz-SignedHeaders=host&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEEMaDmFwLW
5vcnRoZWFzdC0yIkcwRQIgH66n93Q%2FFacwyN0W4KPUvd8cfli7g%2BCLQpkqIsfjs9cCIQDNBrw87CA7xz2v7ukap77S2%2BrZOE%2
F66IC4uGMHD8A8mSrJBQgtEAAaDDk4MjA4MTA3MjE3NSIMff0xbrT4u611eDQuKqYFun215AJNw1MUtdnz%2BVEPzAkRExJMEmS82tiy
srIwvggGNRN%2F864RLQ5MOWzsoGGrcgz3tiUhzWeb%2FY5nra8kIr8oaJ0cwheqIajX0yVx6GHssaDZJC%2FLuZ3z8Mcq0%2FWrEagr
cMhA80cycSX69gNmmnbDjGHv0MKnUbJAcWeXlOm0EJt6eyRkq7L8mh9w%2BXvIj%2B7yvZssAKOo6CfINGctK2ZK94OpTw6DbZ3xIrN%
2FWu5J5a0APXhMrGnavDeG5qpPI7oQY6lG1vwSe%2FYABUHuhT2WONwj11ZXxRJtvLvAr0kti9qpX8o9rd9VkhVOy4nSVDn%2FyFsBaD
4C%2BuRe0CCZpjM4nzAVpmMNd1O1SbziPzTScg47mpryGfBUoK5o1Fd4yd2OSeaE%2BiqIPrnclD%2Bf7AQ3V2gpvt%2FmhzFLh4pKkA
1q7I0q3kud3FYXD7UspIj6erWDhspkeXzIw3JrL%2FD%2B6utggc%2BL6%2Be7yxx9itmIGljPnUWuoOMeN5UVqNGVnCmN8bG08ANQmz
zUakNd4Lu4ttPGuOE8eLufw1J%2FxXCGwYPB1YZrVbTzU2tASCJVDlAKkPDi2%2Bvgr5nMfpXuHIOaoMo%2BvpQQC324x5DBzMo7b580
gPPeURt3ptAj1QM8GHgn8uPnh7xzFPw8jvwFpbnx5GYjfyeC5JMC7uhCjO8MZzP4BRGZOnK0riH4HXpHkgSJagQvyIOR%2BM4D6K65%2
FYEtWSuqxayiNofOMyJZJZFuUSBRQ52FC%2Br8BFk2qgQRfzKRxjrlrdLNlSSWeJTS85ZdnkcYZKi6vl5NleX2SYHQ9r4uvOBGvEEeIc
SR2QLYQrszfvH%2BC3m90TqwkJYzVL%2F09mvfIcRr%2B%2F3dv8xZmCNG%2BLypUPEXdOdITlnaNOmxsmebbXvFxowbMM7X6bsGOrEB
z67UQplP3JrrhDPR0T4bUtxAh2JLVbDm%2Fr8W4eDGQSj7Dk%2FYeBBFQ83cQOOie1gtWWLAVgTp9jgD%2F1pA3RmlAb37ItsQBG0dwr
7fmBbezTJhqHRE5V%2FHNrcW0TWxLuQ1HCHCVGECSpgDeSV0C%2BDK3JKj9oVVfVUCC3P0y4kzFH7F3zN7YBDcalaPF5o7maGQTy748I
fjlsRfayj15seAd3g%2FHQiUdHwoLVewxnsJqF2D&X-Amz-Signature=922c1aa31b09e1b63567ba0b76612b63789c7af008760e6
ac84385599e7d66ec
```

앞서 생성된 URL(https://cloudneta-s3-primary.s3.ap-northeast-2.amazonaws.com/presigned.txt ~~~)로 웹에서 접근할 수 있는지 확인한다.(300초 정도)

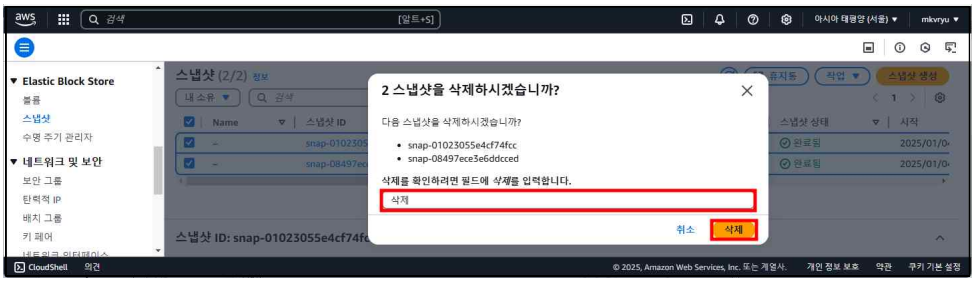


5.5.7. 실습을 위해 생성된 모든 자원 삭제하기

서비스 => 컴퓨팅 => EC2 => 스냅샷 메뉴로 들어가 모든 스냅샷에 체크한 후 작업 => 스냅샷 삭제를 누른다.



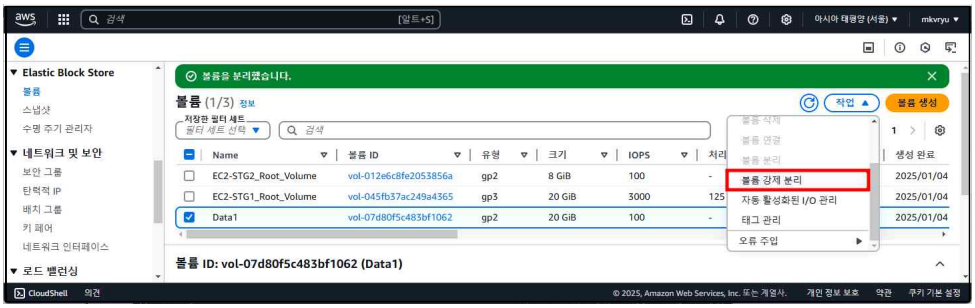
스냅샷 삭제 확인을 위해 “삭제”를 입력하고 삭제를 누른다.



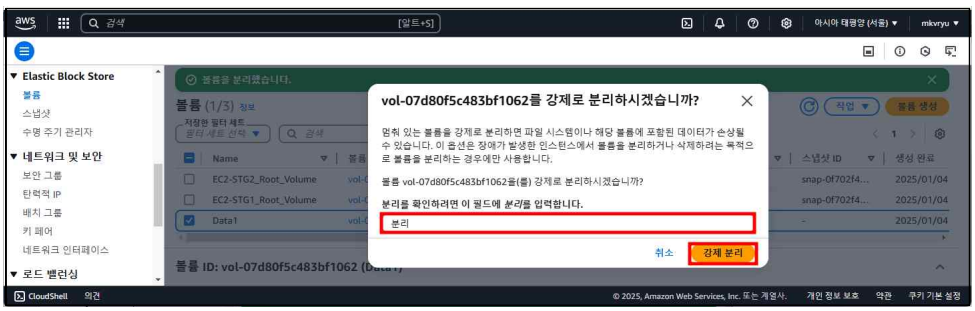
서비스 => 컴퓨팅 => EC2 => 볼륨 메뉴로 들어가 “Data1”을 체크한 후 작업 => 볼륨 분리를 누른다.

“분리”를 누른다. 5 ~ 10분의 시간이 소요된다.

오랜 시간동안 볼륨 분리가 실행이 안되면 “볼륨 강제 분리”를 클릭해서 분리한다.

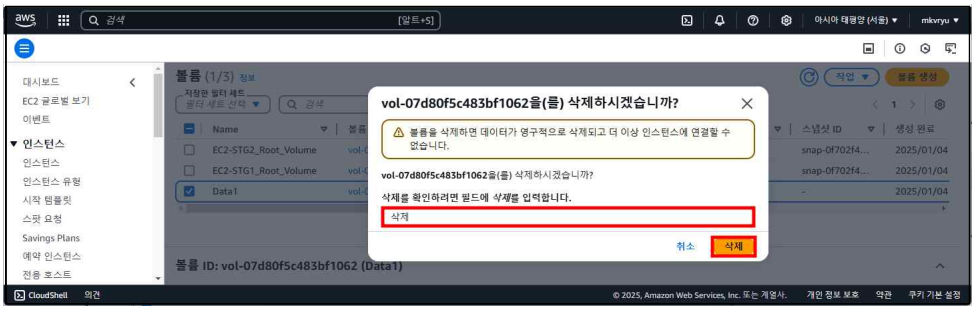


볼륨 강제 분리를 위해 “분리”를 입력하고 “강제 분리”를 누른다.

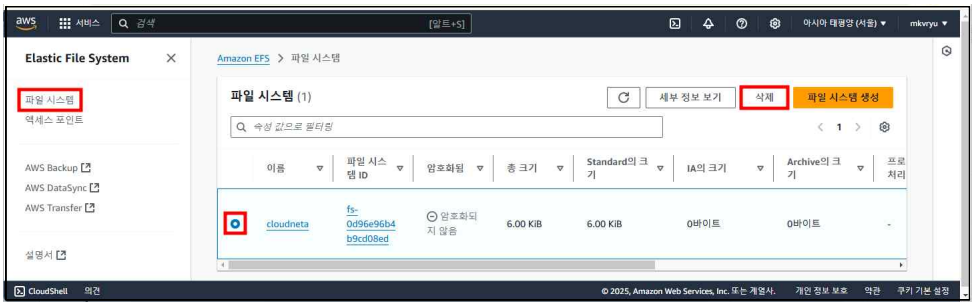


볼륨 분리 또는 강제 분리가 완료되면 서비스 => 컴퓨팅 => EC2 => 볼륨 메뉴로 들어가 “Data1”을 체크한 후 작업 => 볼륨 삭제를 누른다.

볼륨 삭제를 위해 “삭제”를 입력하고 “삭제”를 누른다.

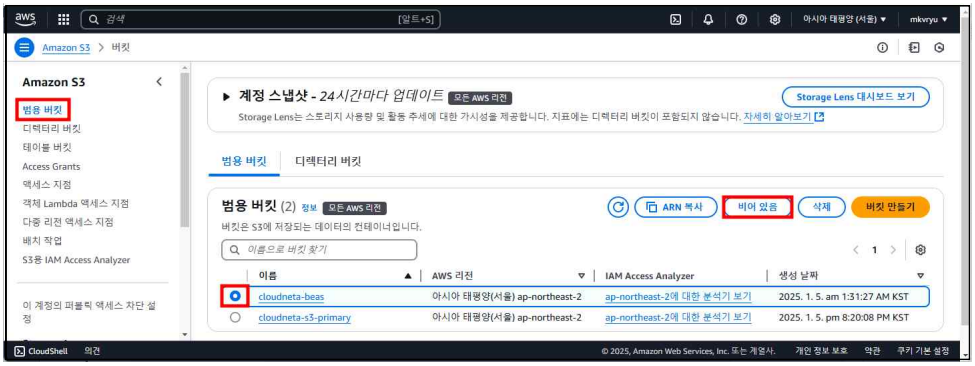


서비스 => 스토리지 => EFS => 파일 시스템 메뉴로 들어간 후 생성한 파일 시스템을 선택하고 “삭제”를 누른다.

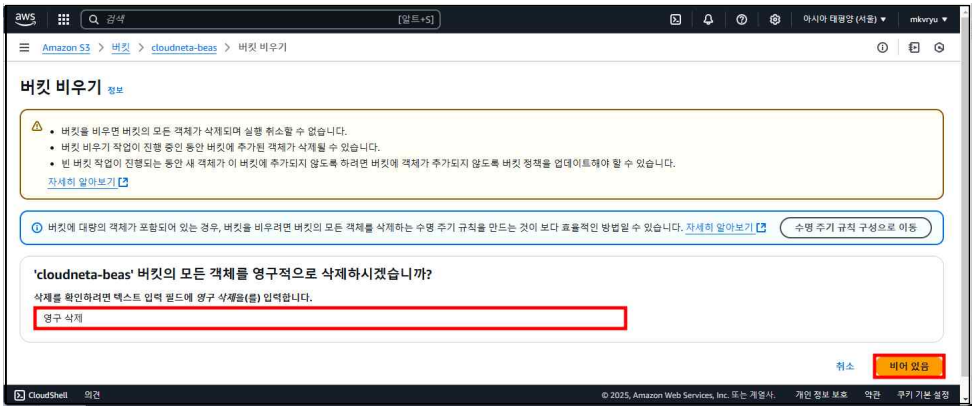


파일 시스템 삭제를 위해 “파일 시스템의 ID”를 입력하고 “확인”을 누른다.

서비스 => 스토리지 => S3 => 버킷 메뉴로 들어간 후 삭제할 버킷을 선택한 후 “비어 있음”을 누른다.



버킷을 비우기 위해 “영구 삭제”를 입력하고 “비어 있음”을 누른다. 버킷에 데이터가 들어있지 않다면 비울 필요 없이 바로 삭제한다.



다시 삭제할 버킷을 선택하고 “삭제”를 누른다.

버킷을 삭제하기 위해 “삭제할 버킷 이름”을 입력하고 “버킷 삭제”를 누른다.

다른 버킷도 동일한 방법으로 삭제한다.

서비스 => 관리 및 거버넌스 => CloudFormation => 스택 메뉴로 들어간 후 삭제할 스택을 선태하고 “삭제”를 누른다. 이후 발생하는 창에서는 “삭제”를 누른다.