

Terraform을 이용한 AWS 3-tier 아키텍처 구현

[1조]

조장 김민혁

조원 문광훈 서현지 최나라

목차

0. 조원 및 역할 소개

1. 프로젝트 소개

1.1 프로젝트 개요

1.2 프로젝트 진행 일정

2. 문제 정의

2.1 제안 배경 및 필요성

2.2 프로젝트 목표

2.3 아키텍처 구성

2.4 시스템 구성도 예상안

3. 요구사항 분석

4. 설계

4.1 전체 시스템 설계도

4.2 사용 서비스 및 기술

4.3 네트워크

4.4 서버

4.5 DB

4.6 스토리지

4.7 플랫폼

4.8 보안

4.9 백업

5. 구축

6. 프로젝트 결과

6.1 결과

6.2 테스트 시나리오

6.3 검토

6.4 자체 평가 및 후기

7. 부록

7.1 작업일지

(8. 참고문헌)

0. 조원 및 역할 소개

|  |  |
| --- | --- |
| 김민혁(조장) | 1. DB 구축 2. 문제정의서 작성 3. 요구사항 분석 4. 테스트 진행 및 검토 |
| 문광훈 | 1. 프로젝트 일정 조율 2. 아키텍처 설계 3. PPT 작성 4. Terraform 환경 구축 5. 문제정의서 작성 6. 요구사항 분석 |
| 서현지 | 1. Autoscaling 구축 2. Load Balancer 구축 3. 아키텍처 설계 4. PPT 작성 5. 문서 작성 6. 문제정의서 작성 7. 요구사항 분석 |
| 최나라 | 1. VPC 구축 2. Route53구축 및 도메인 설정 3. Instance 설정 및 구축 4. S3 storage 구축 5. 문제정의서 작성 6. 요구사항 분석 |

1. 프로젝트 소개

1.1 프로젝트 개요

- 문제 상황 정의 : 특정 고객, 특정 상황을 가정

- 요구사항 분석 : 해당 문제 상황에서 요구사항을 분석 및 도출

- 요구 사항에 따른 설계 : 최대한 문제 상황과 요구 분석에 알맞은 솔루션 설계

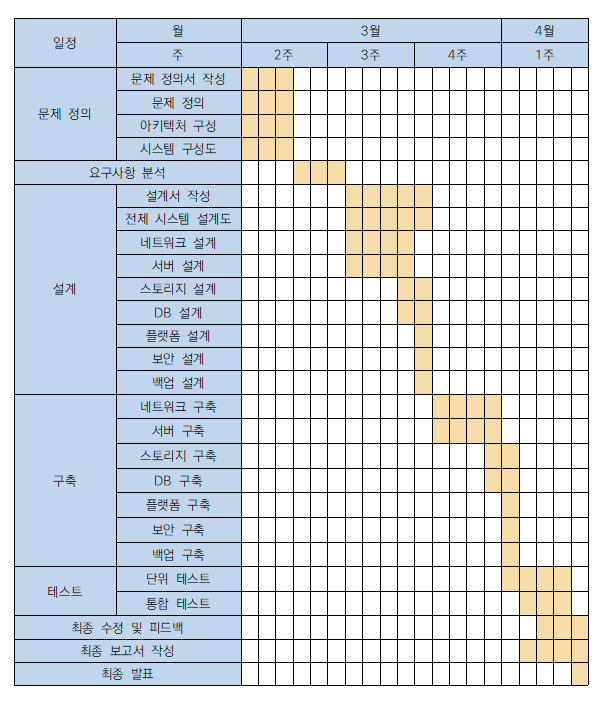
1.2 프로젝트 진행 일정

- 1주차 : 문제정의서 작성

- 2주차 : 요구사항 분석 및 설계

- 3주차 : 설계 및 구축

- 4주차 : 구축 마무리 및 테스트



2. 문제 정의

2.1 제안 배경 및 필요성

[프로젝트 개요]

최근 이커머스 시장에서는 D2C(Direct to Customer)형식의 플랫폼이 최근 트렌드로 떠오르고 있다. 이 D2C 플랫폼은 유통상을 통하지 않고 소비자를 자사몰로 유입시켜 직접 판매하는 형태의 방식이다.

일반적으로 오픈마켓(11번가 , G-market … )을 이용하지 않고 자사몰을 이용하는 클라이언트가 따로 컴퓨터 자원에 투자하는것은 초기 투자비용이 크게 들기때문에 리스크가 크다.

또한 컴퓨터 자원에 투자하여 자사가 운영을 한다고 하더라도 사업의 규모가 커짐에 따라서 증가하는 회원과 물품의 데이터저장 및 처리 또는 해외의 고객유치 및 해외영업의 확장의 상황을 맞닥뜨릴때 탄력적으로 인프라를 운영하기가 어렵다는 문제가 존재한다.

따라서 많은 기업들이 자사의 성장에 따른 사업의 확장을 위하여 퍼블릭 클라우드 서비스로의 이전 및 구축서비스를 필요로 하고있으며 최근 삼성과 아모레 퍼시픽 등의 대기업도 기존의 3-tier 구조 웹서비스를 탄력적인 인프라 운영과 서비스 응답 속도의 개선,글로벌 시장 확장을 위하여 클라우드 서비스를 도입하고 있다.

전자상거래 어플리케이션 및 자사몰 구축에서 가장많이 사용되는 아키텍처인 3-tier 아키텍처를 메인으로 하여,

글로벌 시장을 공략하거나 사업의 규모를 늘리려는 고객을 위한 **'AWS 3-tier Cloud Architecture 구현'**을 목표로 이번 프로젝트를 진행하려고 한다.

[고객사례 : 3계층 온프레미스 서버로 전자상거래 플랫폼을 운영하는 회사 A company]

- 자사에서 운영할 파일럿 어플리케이션을 지속적으로 운영할 서버가 필요함

- 초기 자본금이 넉넉하지 않음

- 인프라 관리 인력의 부족

- 향후 미래의 사업 확장에 따른 인프라 확장 및 시장 확대 가능성에도 염두를 둠

2.2 프로젝트 목표

자체적으로 자사 웹 및 어플리케이션을 구성하려는 회사에게 적합한 범용적 아키텍처 설계

2.3 아키텍처 구성

[3-tier 아키텍처]

1. 프레젠테이션 계층

* 사용자가 마주하는 화면이며 여기서 자신이 원하는 서비스를 선택할 수 있다.
* 온라인 쇼핑 애플리케이션이라면, 프리젠테이션 티어는 상품검색, 상품정보, 구매, 장바구니와 같은 정보들을 화면에 표시한다. 상품검색, 물품구입등의 유저 액션이 발생하면 네트워크를 통해서 액션을 다른 티어에 전달하고 다시 그 결과를 화면에 표시한다.
* 즉, 웹 브라우저, 애플리케이션등, 유저가 직접 상호작용하는 소프트웨어들이다.

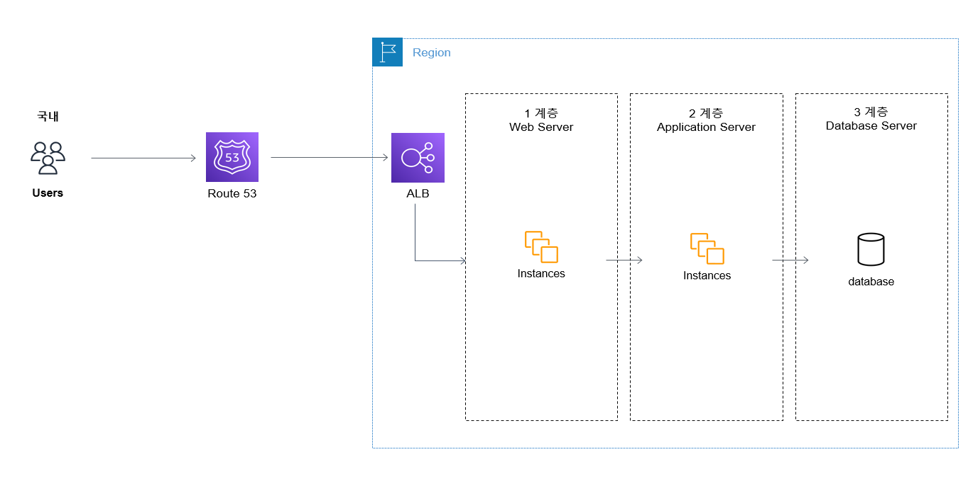
2. 애플리케이션 계층

* 사용자가 원하는 서비스 , 결제 , 주문 , 회원가입 등등의 요청을 처리하는 계층이다.
* 유저인증, 연결관리, 빌링, 구매, 정보출력(상품목록 같은)등의 기능이 포함된다.
* 이 계층에서는 해당 서비스에서 어떤 데이터가 필요한지를 결정 후 데이터계층에 클라이언트로써 접근(요청)하여 서비스에 필요한 정보를 가져온다.
* 중요 정보들은 데이터베이스에 저장을 해야하므로 데이터 팀과 커뮤니케이션해야 한다. 또한 효과적인 API 작성을 위해서 프론트앤드팀과도 커뮤니케이션 해야 한다. 중간에 위치하는 팀이다.

3. 데이터 계층

* 데이터 베이스에 접근하는 프로그램과 데이터 베이스를 포함하여 데이터 계층이라 부른다.
* 각 서비스에 필요한 정보들을 저장하는 공간이다.
* 데이터 티어는 데이터를 안정적/지속적으로(persistence)유지하기 위한 매커니즘을 제공한다.
* 이러한 매커니즘에는 데이터베이스 서버, 파일서버등이 포함된다. 또한 이들 매커니즘들은 데이터를 읽고, 쓰고, 관리하기 위한 기능들을 제공한다.
* 데이터 티어는 이들 기능을 애플리케이션 티어에 제공하는 일을 한다. 일반적으로 데이터베이스, 파일시스템등의 종속성을 피하기 위해서 API 형태로 기능을 개발해서 제공한다.

2.4 시스템 구성도 예상안



3. 요구사항 분석

* 성능 요구사항

|  |  |
| --- | --- |
| 요구사항 명 | 시스템 자원 효율성 |
| 요구사항 내용 | - CPU 평균 사용률은 최대 50%를 초과하지 않도록 구현해야 함  - 테스트 기간과 시험 운영기간에 시스템의 자원(메모리, 디스크) 사용 정도를 평가하며, 메모리 사용률은 위의 요구사항에 충족하도록 한다. |

* 보안 요구사항

|  |  |
| --- | --- |
| 요구사항 명 | 서버 보안 및 데이터 무결성 |
| 요구사항 내용 | - 관리자 외 일반 사용자는 DB 와 어플리케이션 서버에 접근 할 수 없어야 한다. - 데이터의 변경 , 삽입 , 삭제 등은 허가받은 사람만 가능하도록 설계되어야 함 |

|  |  |
| --- | --- |
| 요구사항 명 | 보안취약성 점검 |
| 요구사항 내용 | - 연 1회의 보안점검 진행 및 로그파일로 저장할 수 있어야 함 |

|  |  |
| --- | --- |
| 요구사항 명 | 사용자 인증 |
| 요구사항 내용 | - 사용자(관리자)가 내부 컴퓨터 접근을 시도할 때 인증 처리가 필요 (SSH) |

* 유지보수 요구사항

|  |  |
| --- | --- |
| 요구사항 명 | 정기 백업 정책 -> AWS Back up |
| 요구사항 내용 | - 장애 발생 시에 복구를 위한 백업정책을 수립  <데이터의 특성(중요도, 데이터 용량 등)에 따라 백업주기와 범위 설정>  - 회원정보는 매일 12시간마다 백업을 실행하여 회원정보는 RPO시간이 최대 하루 |

|  |  |
| --- | --- |
| 요구사항 명 | 장애 대처 방안 마련 |
| 요구사항 내용 | - 장애가 발생할 경우를 대비해 구축된 응용시스템 별로 시나리오를 작성하여 장애발생 시 신속하게 대처할 수 있는 계획(장애처리와 복구지침)을 수립해야 함  <각 장애 발생별로 최소 요구되는 소요시간을 포함하여 제시해야 함>  - RTO는 2시간 이내로 진행되어야 함 |

|  |  |
| --- | --- |
| 요구사항 명 | 트래픽 증가 -> Auto Scaling |
| 요구사항 내용 | - 트래픽 감시하고 증가하면 자동으로 늘려 성능을 그대로 유지시켜야 함 |

|  |  |
| --- | --- |
| 요구사항 명 | 시스템 이중화 |
| 요구사항 내용 | - 재해나 사고가 발생하여 메인서버에 문제가 발생하더라도 서버는 동작 되어야함 |

* 데이터베이스 요구사항

|  |  |
| --- | --- |
| 요구사항 명 | 데이터 베이스 서버 이중화 |
| 요구사항 내용 | - 데이터 베이스 고가용성을 위한 DB서버 이중화가 되어 있어야 함 |

|  |  |
| --- | --- |
| 요구사항 명 | 전자 상거래 환경 적합성 |
| 요구사항 내용 | - 전사적 자원 관리(ERP) , 고객 관계 관리(CRM) 등 전자 상거래 맞춤형 데이터 베이스 제품을 적용해야 함 |

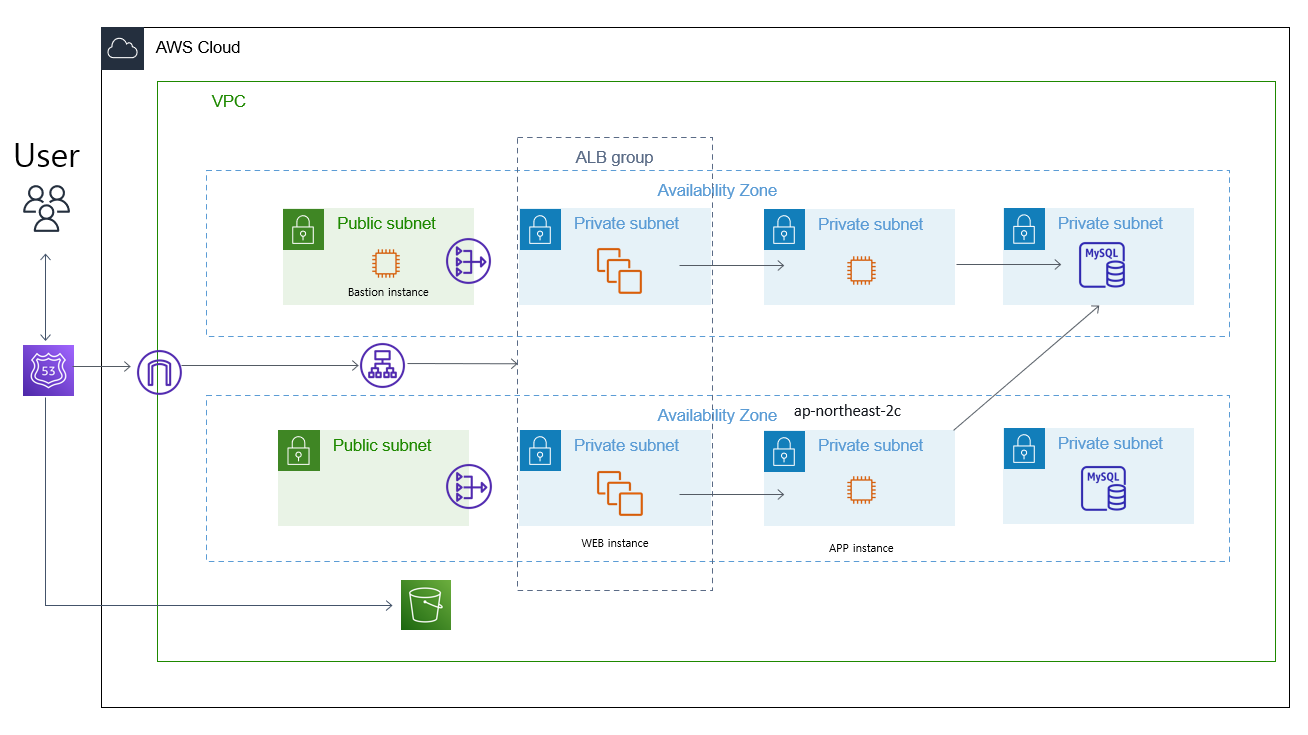
* 네트워크 요구사항

|  |  |
| --- | --- |
| 요구사항 명 | 인터넷 성능 개선 |
| 요구사항 내용 | - 병목현상이나 순간적인 트래픽 급증에 대해 안정적인 웹 서비스 제공을 해야한다. |

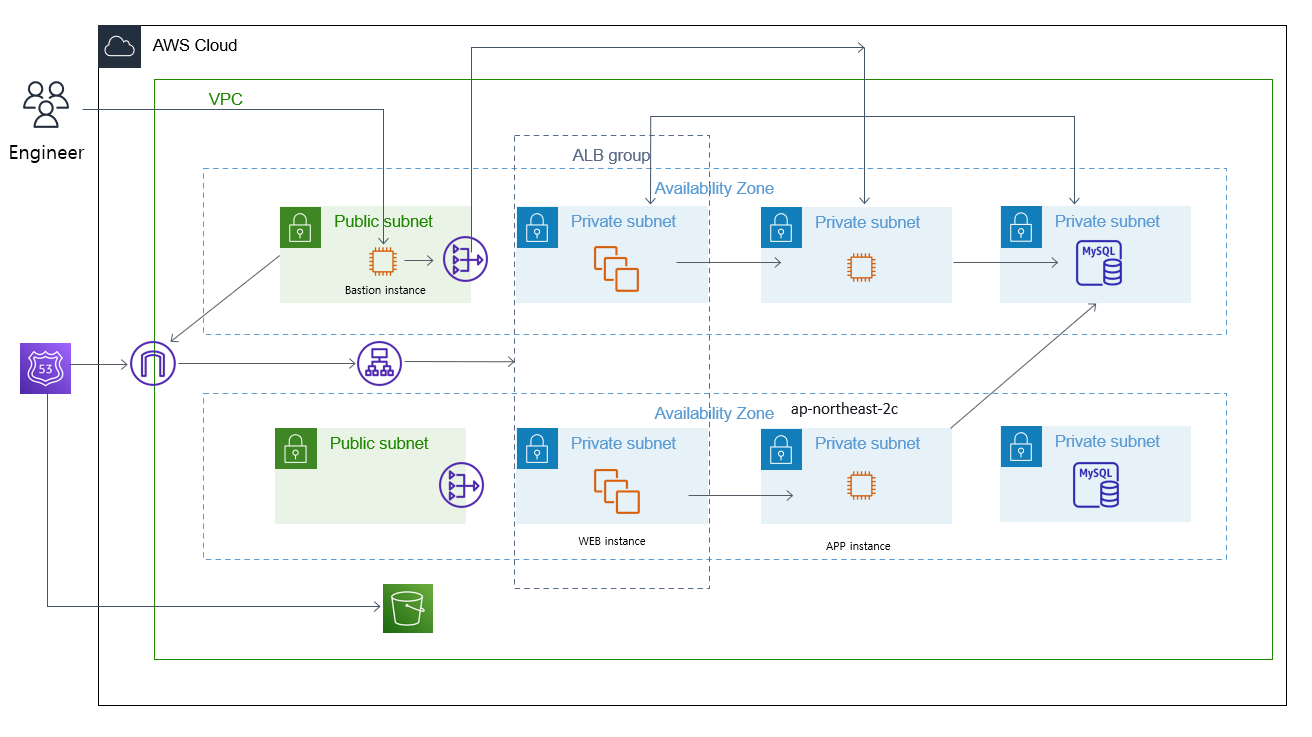
4. 설계 및 구축

4.1 전체 시스템 설계도

* 유저, 쇼핑몰 이용고객

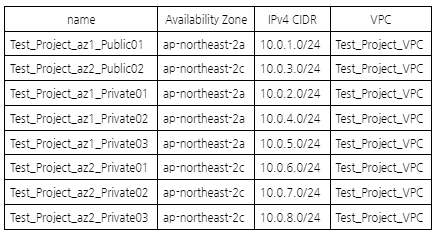


* 관리자



4.2 네트워크

* Region : 서울
* AZ(가용영역) : 2개의 가용영역을 메인 영역으로 사용
  + AZ 1 : ap-northeast-2a
  + AZ 2 : ap-northeast-2c
* VPC
  + vpc 이름 : Test\_Project\_VPC
  + IPv4 CIDR : 10.0.0.0/16
* Subnet : public 2개, private 6개



* IGW(인터넷 게이트웨이)
* Public 서브넷을 외부 인터넷과 연결시켜주기 위함
* 퍼블릭 IPv4 주소가 할당된 인스턴스에 대해 NAT(네트워크 주소 변환)를 수행
* 이름 태그 : Test\_Project\_Internet\_GW
* VPC : Test\_Project\_VPC

|  |  |
| --- | --- |
| 서브넷 | IPv4 CIDR |
| Test\_Project\_az1\_Public01 | 10.0.1.0/24 |
| Test\_Project\_az2\_Public02 | 10.0.3.0/24 |

* 라우팅 테이블
  + public 테이블: Test\_Project\_Public\_Route\_Table
    - VPC : Test\_Project\_VPC
    - 서브넷 연결

|  |  |
| --- | --- |
| 서브넷 | IPv4 CIDR |
| Test\_Project\_az1\_Public01 | 10.0.1.0/24 |
| Test\_Project\_az2\_Public02 | 10.0.3.0/24 |

* + - 라우팅

|  |  |
| --- | --- |
| Destination | Target |
| 10.0.0.0/16 | local |
| 0.0.0.0/0 | Test\_Project\_Internet\_GW |

* + private 테이블 : Test\_Project\_Private\_Route\_Table1
    - VPC : Test\_Project\_VPC
    - 서브넷 연결

|  |  |
| --- | --- |
| 서브넷 | IPv4 CIDR |
| Test\_Project\_az2\_Private01 | 10.0.2.0/24 |
| Test\_Project\_az2\_Private02 | 10.0.4.0/24 |
| Test\_Project\_az2\_Private03 | 10.0.5.0/24 |

* + - 라우팅

|  |  |
| --- | --- |
| Destination | Target |
| 10.0.0.0/16 | local |

* + private 테이블 : Test\_Project\_Private\_Route\_Table2
    - VPC : Test\_Project\_VPC
    - 서브넷 연결

|  |  |
| --- | --- |
| 서브넷 | IPv4 CIDR |
| Test\_Project\_az2\_Private01 | 10.0.6.0/24 |
| Test\_Project\_az2\_Private02 | 10.0.7.0/24 |
| Test\_Project\_az2\_Private03 | 10.0.8.0/24 |

* + - 라우팅

|  |  |
| --- | --- |
| Destination | Target |
| 10.0.0.0/16 | local |

* NAT Gateway
  + Public Subnet에서 Private Subnet로 접근을 위해 구현
  + Public Subnet 아래에 NAT Gateway 생성
  + Private Subnet Routing table에 Rule추가
  + Elastic IP 주소 할당
    - 이름 : Test\_Project\_Elastic\_IP01
    - 이름 : Test\_Project\_Elastic\_IP02
* NAT 게이트웨이 az-1 생성
  + 이름 : Test\_Project\_Nat\_Gateway01
  + 서브넷 : Test\_Project\_az1\_Public01
  + 연결유형 : 퍼블릭
  + 탄력적 IP 할당
  + 태그 : Test\_Project\_Nat\_Gateway01
  + NAT 게이트웨이 az-2 생성
    - 이름 : Test\_Project\_Nat\_Gateway02
    - 서브넷 : Test\_Project\_az2\_Public02
    - 연결유형 : 퍼블릭
    - 탄력적 IP 할당
    - 태그 : Test\_Project\_Nat\_Gateway02
* Route53
  + DNS 구축
* 사전에 사용할 도메인 준비
* 도메인 이름 : test-project.tk
* 퍼블릭 호스팅 영역
* 태그 : Test\_Project\_domain
* S3 스토리지와 연결
* 목적 : 정적 웹 사이트 호스팅

4.3 서버

* 인스턴스
* Bastion Instance
* Private subnet의 호스트 관리 전용 EC2 instance
* 외부에서 접근 불가능한 private subnet에 접근하기 위해 각각의 AZ의 public subnet 중 하나에만 bastion host를 구현
* bastion host를 통해 특정 IP만이 private subnet에 접근 가능하도록 설정
* 관리자만 접근 가능
* 외부에서 접근 불가능한 private subnet에 접근하기 위해 각각의 AZ의 public subnet 중 하나에만 bastion host를 구현
  + AMI : Amazon Linux 2 AMI
  + 인스턴스 유형 : t2.micro
  + 인스턴스 구성
    - 네트워크 : Test\_Project\_VPC
    - 서브넷 : Test\_Project\_az1\_Public01
    - 퍼블릭 IP 자동할당 : enable
  + 태그 : Bastion (Name)
  + 보안 그룹 구성
    - 이름 : Bastion\_Security

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Protocol | Port range | Source |
| Inbound rules | SSH | TCP | 22 | 0.0.0.0/0 |
| SSH | TCP | 22 | ::/0 |

* 키페어 : Test\_Project\_Key
* Web\_Server\_Instance
* 사용자가 접근 하는 프레젠테이션 계층
* Web Server 역할을 할 수 있는 인스턴스
* Private Subnet 안에 위치
* 서브넷 Test\_Project\_az1\_Private01, Test\_Project\_az2\_Private01 각각 하나의 인스턴스 생성
  + AMI : Amazon Linux 2 AMI
  + 인스턴스 유형 : t2.micro
  + 인스턴스 구성
    - 네트워크 : Test\_Project\_VPC
    - 서브넷 : Test\_Project\_az1\_Private01, Test\_Project\_az2\_Private01(각각 하나씩 설정)
    - 퍼블릭 IP 자동할당 : enable
  + 태그 : Web Server (Name)
  + 보안 그룹 구성 :
    - 이름 : Web\_Server\_Security

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Protocol | Port range | Source |
| Inbound rules | SSH | TCP | 22 | Bastion-Security |
| HTTP | TCP | 80 | 0.0.0.0/0 |
| HTTP | TCP | 80 | ::/0 |
| 모든 ICMP -IPv4 | ICMP | 전체 | Bastion-Security |

* 키페어 : Test-Project-Key
* Web\_Application\_Server\_Instance
* Web Application Server 역할을 할 수 있는 인스턴스
* Private Subnet 안에 위치
* 서브넷 Test\_Project\_az1\_Private02, Test\_Project\_az2\_Private02 각각 하나의 인스턴스 생성
  + AMI : Amazon Linux 2 AMI
  + 인스턴스 유형 : t2.micro
  + 인스턴스 구성
    - 네트워크 : Test\_Project\_VPC
    - 서브넷 : Test\_Project\_az1\_Private02, Test\_Project\_az2\_Private02 (각각 하나씩 설정)
  + 태그 : App Server (Name)
  + 보안그룹 구성
    - 이름 : App\_Server\_Security

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Type | Protocol | Port range | Source |
| Inbound rules | SSH | TCP | 22 | Bastion-Security |

* 로드밸런서
  + Internet Gateway로 들어온 트래픽을 Web으로 분산
  + 구성
    - 이름 : Test-Project-ALB
    - VPC :  Test\_Project\_VPC
    - 가용영역 : Test\_Project\_az1\_Public01, Test\_Project\_az2\_Public02
    - 보안그룹 : 외부에서 들어오는 어떤 IP 모두 허용
    - 타겟그룹 : private subnet(Web Instance)
* 로드밸런서 보안 그룹
* 이름 : ALB\_Security
* 네트워크 : Test\_Project\_VPC
* bastion host에서 접근할 수 있도록 ssh 유형 22포트로 Bastion-Security에서 통과한 트래픽에 대해서만 들어올 수 있도록 개방
* 규칙

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | IP 버전 | Type | Protocol | Port range | Source |
| Inbound rules | IPv4 | HTTP | TCP | 80 | 0.0.0.0/0 |
| IPv6 | HTTP | TCP | 80 | ::/0 |
| - | 모든 ICMP - IPv4 | ICMP | 전체 | Bastion-Security |
| - | HTTPS | TCP | 443 | 0.0.0.0/0 |
| - | SSH | TCP | 22 | Bastion-Security |
| Outbound rules | IPv4 | 전체 | 전체 | 전체 | 0.0.0.0/0 |

* Auto Scaling
  + 수요가 급증할 때 EC2 인스턴스 수를 자동으로 늘려 효율적으로 운영을 위해 구축
  + 로드밸런서 ‘Test-Project-ALB’와 Test\_Project\_WebServer\_ASG를 연결
  + 평균 CPU 사용률이 50%가 넘으면 인스턴스의 수를 동적으로 조정하여 인스턴스를 최대 4개까지 추가로 생성하여 시작할 수 있도록 설정
  + Auto Scaling 그룹
    - Test\_Project\_WebServer\_ASG : 가용영역 ap-northeast-2a와 ap-northeast-2c간의 web instance를 균일하게 분산

4.4 DB

* Amazon RDS for MySQL
  + Amazon RDS for MySQL은 클라우드에서 MySQL 배포를 손쉽게 설정, 운영 및 확장할 수 있게 해주는 관계형 데이터베이스 서비스
* RDS 생성
  + 엔진 유형 : MySQL
  + DB 클러스터 식별자 : test-project-db
  + 가용성 및 내구성 : 다중 AZ 배포
  + VPC : Test\_Project\_VPC
  + 보안 그룹 : DB Security
  + 기본 DB 인스턴스 : ap-northeast-2c - Writer Instance(읽기 쓰기)
  + 복제본 인스턴스 : ap-northeast-2a - Reader Instance(읽기)

4.5 스토리지

* **S3 bucket**
  + 버킷 이름 : test-project.tk
  + 리전 : ap-northeast-2
  + 버전 관리 : 활성화
  + 정적 웹 사이트 호스팅 : 활성화
    - 정적 웹 사이트 호스팅을 위해 S3 사용

4.6 플랫폼

* **OS 설계**
  + **Amazon Linux2 사용**
  + 적용 인스턴스
    - Bastion Instance
    - WEB Server Instance
    - WEB Application Server Instance
* **SW 설계**
  + WEB Server Instance
    - Apache WebServer ver 2.4.51
    - PHP ver 7.3
  + Web Application Server Instance
    - tomcat ver 9.0

4.7 보안

* IAM 관리자 설정
  + 관리자 권한 :
    - 전체적으로 모든 곳에 접근 가능해야 함
    - 루트 계정 사용 방지를 위한 IAM 관리자 계정 설정
    - AmazonRDSFullAccess, AmazonEC2FullAccess, AmazonS3FullAccess, CloudWatchFullAccess 권한 부여
* 키 페어 생성
  + EC2 용 SSH 키 페어 정의
  + key\_name : Test\_Project\_Key
* 네트워크 보안
  + Bastion Instance 보안 그룹
* 이름 : Bastion\_Security
* 네트워크 : Test\_Project\_VPC
* 규칙

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | IP 버전 | Type | Protocol | Port range | Source |
| Inbound rules | IPv4 | SSH | TCP | 22 | 0.0.0.0/0 |
| IPv6 | SSH | TCP | 22 | ::/0 |
| Outbound rules | IPv4 | 전체 | 전체 | 전체 | 0.0.0.0/0 |

* + WEB Server Instance 보안 그룹
* 이름 : Web\_Server\_Security
* 네트워크 : Test\_Project\_VPC
* bastion host에서 접근할 수 있도록 ssh 유형 22포트로 Bastion-Security에서 통과한 트래픽에 대해서만 들어올 수 있도록 개방
* 규칙

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | IP 버전 | Type | Protocol | Port range | Source |
| Inbound rules | IPv4 | HTTP | TCP | 80 | 0.0.0.0/0 |
| IPv6 | HTTP | TCP | 80 | ::/0 |
| - | 모든 ICMP - IPv4 | ICMP | 전체 | Bastion-Security |
| - | HTTPS | TCP | 443 | 0.0.0.0/0 |
| - | SSH | TCP | 22 | Bastion-Security |
| Outbound rules | IPv4 | 전체 | 전체 | 전체 | 0.0.0.0/0 |

* + WEB Application Server Instance 보안 그룹
* 이름 : App\_Server\_Security
* 네트워크 : Test\_Project\_VPC
* bastion host에서 접근할 수 있도록 ssh 유형 22포트로 Bastion-Security에서 통과한 트래픽에 대해서만 들어올 수 있도록 개방
* 규칙

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | IP 버전 | Type | Protocol | Port range | Source |
| Inbound rules | - | SSH | TCP | 22 | Bastion-Security |
|  |  |  |  |  |
| Outbound rules | IPv4 | 전체 | 전체 | 전체 | 0.0.0.0/0 |

* + DB 보안 그룹
* 이름 : DB\_Security
* 네트워크 : Test\_Project\_VPC
* 규칙

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | IP 버전 | Type | Protocol | Port range | Source |
| Inbound rules | - | MYSQL/Aurora | TCP | 3306 | Bastion-Security |
| - | MYSQL/Aurora | TCP | 3306 | Web Server Security |
| Outbound rules | IPv4 | 전체 | 전체 | 전체 | 0.0.0.0/0 |

4.8 백업

* EC2 및 RDS 백업 관리
  + AWS BackUp 을 사용한 EC2 리소스 백업 관리
  + AWS BackUp 는 AMI 와 스냅샷 형태로 EC2 리소스를 백업 따라서   
    다른 리전으로 새롭게 복구 및 원하는 시점으로 되돌리기 가능
  + 백업 플랜 적용 resource : DB(Aurora), EC2, S3
    - Aurora : test-project-db
    - EC2 : All instance
    - S3 : test-project.tk
  + 백업계획
    - 이름 : Test\_Project\_BackUpPlan
    - 태그 : Test-Project-BackUpPlan
    - 백업 규칙 구성
      * 규칙이름 : BackUp\_Every\_12Hours
      * 백업빈도 : 12시간마다
      * 백업기간 : 기본값 사용
      * 콜드 스로리지로 전환 : 안함
      * 보존 기간 : 항상
      * 백업 리전 : 버지니아 리전으로 복사본 생성
      * Enable continuous backups for point-in-time recovery (PITR) 특정시점으로 복구 가능하도록 지속 적인 백업 활성화

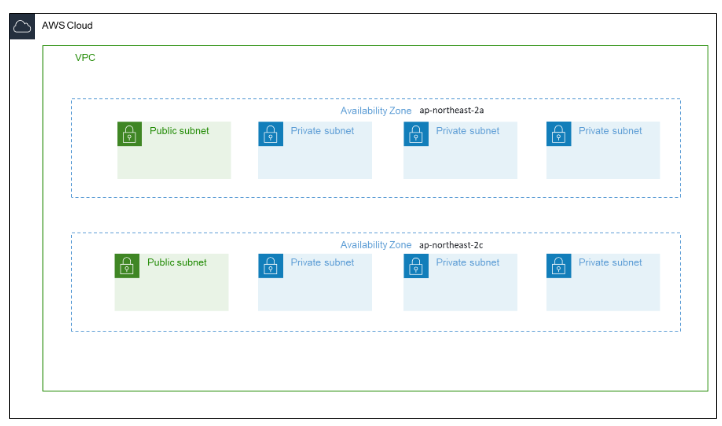
5. 구축

5.1 사용 서비스 및 기술

* Amazon Elastic Compute Cloud(Amazon EC2) : 클라우드 환경에서 서버를 할당 받아 사용할 수 있는 서비스
* Amazon EC2 Auto Scaling : 애플리케이션을 위해 확장 가능한 AWS 리소스를 신속하게 파악하고 동적 규모 조정을 설정
* Elastic Load Balancing : 트래픽을 여러 대의 서버에 균등하게 분산
* Amazon Virtual Private Cloud(Amazon VPC) : 가상 네트워크
* Amazon Route 53 : DNS 서비스, AWS에서 실행되는 인프라에 효과적으로 연결
* Amazon RDS for MySQL : Amazon RDS for MySQL은 클라우드에서 MySQL 배포를 손쉽게 설정, 운영 및 확장할 수 있게 해주는 관계형 데이터베이스 서비스
* Amazon Simple Storage Service(Amazon S3) : 인터넷 스토리지 서비스, 정적 파일 관리
* Amazon Identity and Access Management (Amazon IAM) : AWS 서비스에 대한 액세스를 안전하게 제어하는 웹 서비스, 사용자 설정 및 액세스 관리
* AWS Backup : Cloud 및 Private 서비스 전반에 데이터 백업을 중앙 집중화하고 자동화하는데 필요한 완전 관리형 백업 서비스

5.2 네트워크 구축

* vpc



|  |
| --- |
| # Create VPC  resource "aws\_vpc" "Test\_Project\_VPC" {  cidr\_block = "10.0.0.0/16"  tags = {  Name = "Test\_Project\_VPC"  }  }  # Create az1 Public Subnet  resource "aws\_subnet" "Test\_Project\_az1\_Public01" {  vpc\_id = "${aws\_vpc.Test\_Project\_VPC.id}"  cidr\_block = "10.0.1.0/24"  tags = {  Name = "Test\_Project\_az1\_Public01"  }  availability\_zone = "ap-northeast-2a"  }  # Create az2 Public Subnet  resource "aws\_subnet" "Test\_Project\_az2\_Public02" {  vpc\_id = "${aws\_vpc.Test\_Project\_VPC.id}"  cidr\_block = "10.0.3.0/24"  tags = {  Name = "Test\_Project\_az2\_Public02"  }  availability\_zone = "ap-northeast-2c"  }  # Create az1 Private Subnet01  resource "aws\_subnet" "Test\_Project\_az1\_Private01" {  vpc\_id = "${aws\_vpc.Test\_Project\_VPC.id}"  cidr\_block = "10.0.2.0/24"  tags = {  Name = "Test\_Project\_az1\_Private01"  }  availability\_zone = "ap-northeast-2a"  }  # Create az1 Private Subnet02  resource "aws\_subnet" "Test\_Project\_az1\_Private02" {  vpc\_id = "${aws\_vpc.Test\_Project\_VPC.id}"  cidr\_block = "10.0.4.0/24"  tags = {  Name = "Test\_Project\_az1\_Private02"  }  availability\_zone = "ap-northeast-2a"  }  # Create az1 Private Subnet03  resource "aws\_subnet" "Test\_Project\_az1\_Private03" {  vpc\_id = "${aws\_vpc.Test\_Project\_VPC.id}"  cidr\_block = "10.0.5.0/24"  tags = {  Name = "Test\_Project\_az1\_Private03"  }  availability\_zone = "ap-northeast-2a"  }  # Create az2 Private Subnet01  resource "aws\_subnet" "Test\_Project\_az2\_Private01" {  vpc\_id = "${aws\_vpc.Test\_Project\_VPC.id}"  cidr\_block = "10.0.6.0/24"  tags = {  Name = "Test\_Project\_az2\_Private01"  }  availability\_zone = "ap-northeast-2c"  }  # Create az2 Private Subnet02  resource "aws\_subnet" "Test\_Project\_az2\_Private02" {  vpc\_id = "${aws\_vpc.Test\_Project\_VPC.id}"  cidr\_block = "10.0.7.0/24"  tags = {  Name = "Test\_Project\_az2\_Private02"  }  availability\_zone = "ap-northeast-2c"  }  # Create az2 Private Subnet03  resource "aws\_subnet" "Test\_Project\_az2\_Private03" {  vpc\_id = "${aws\_vpc.Test\_Project\_VPC.id}"  cidr\_block = "10.0.8.0/24"  tags = {  Name = "Test\_Project\_az2\_Private03"  }  availability\_zone = "ap-northeast-2c"  } |

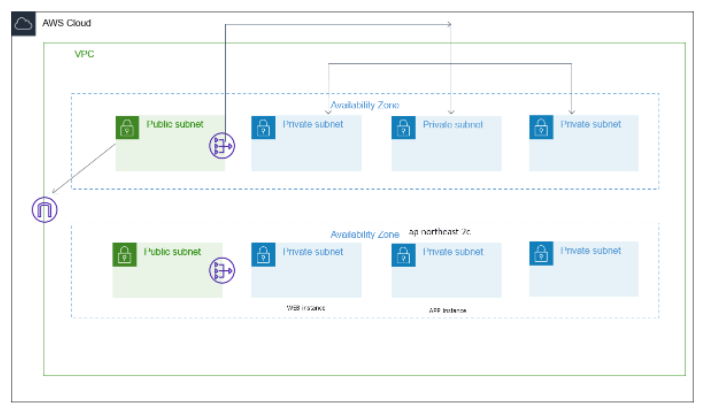
* 인터넷 게이트웨이 & 라우팅테이블



|  |
| --- |
| # Create Internet Gateway  resource "aws\_internet\_gateway" "Test\_Project\_Internet\_GW" {  vpc\_id = "${aws\_vpc.Test\_Project\_VPC.id}"  tags = {  Name = "Test\_Project\_Internet\_GW"  }  } |

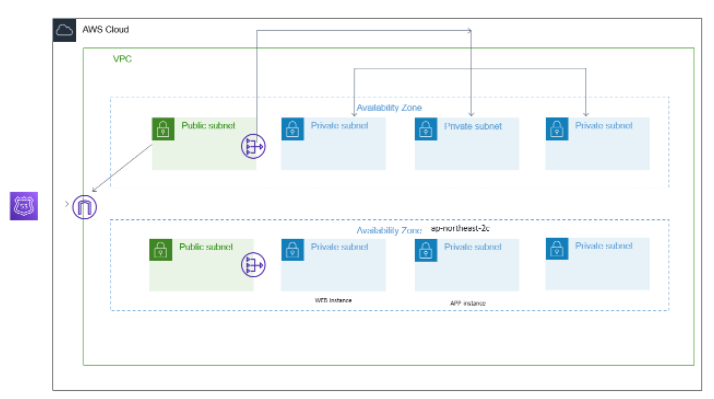
|  |
| --- |
| # Public RT set  resource "aws\_default\_route\_table" "Test\_Project\_Public\_Route\_Table" {  default\_route\_table\_id = "${aws\_vpc.Test\_Project\_VPC.default\_route\_table\_id}"  tags = {  Name = "Test\_Project\_Public\_Route\_Table"  }  }  # Create Private az1-RT  resource "aws\_route\_table" "Test\_Project\_Private\_Route\_Table1" {  vpc\_id = "${aws\_vpc.Test\_Project\_VPC.id}"  tags = {  Name = "Test\_Project\_Private\_Route\_Table1"  }  }  # Create Private az2-RT  resource "aws\_route\_table" "Test\_Project\_Private\_Route\_Table2" {  vpc\_id = "${aws\_vpc.Test\_Project\_VPC.id}"  tags = {  Name = "Test\_Project\_Private\_Route\_Table2"  }  }  # Public route table az1 association  resource "aws\_route\_table\_association" "Test\_Project\_az1\_Public01" {  subnet\_id = "${aws\_subnet.Test\_Project\_az1\_Public01.id}"  route\_table\_id = "${aws\_vpc.Test\_Project\_VPC.default\_route\_table\_id}"  }  # public route table az2 association  resource "aws\_route\_table\_association" "Test\_Project\_az2\_Public02" {  subnet\_id = "${aws\_subnet.Test\_Project\_az2\_Public02.id}"  route\_table\_id = "${aws\_vpc.Test\_Project\_VPC.default\_route\_table\_id}"  }  # Private route table az1 association  resource "aws\_route\_table\_association" "Test\_Project\_az1\_Private01" {  subnet\_id = "${aws\_subnet.Test\_Project\_az1\_Private01.id}"  route\_table\_id = "${aws\_route\_table.Test\_Project\_Private\_Route\_Table1.id}"  }  # Private route table az1 association  resource "aws\_route\_table\_association" "Test\_Project\_az1\_Private02" {  subnet\_id = "${aws\_subnet.PTest\_Project\_az1\_Private02.id}"  route\_table\_id = "${aws\_route\_table.Test\_Project\_Private\_Route\_Table1.id}"  }  # Private route table az1 association  resource "aws\_route\_table\_association" "Test\_Project\_az1\_Private03" {  subnet\_id = "${aws\_subnet.Test\_Project\_az1\_Private03.id}"  route\_table\_id = "${aws\_route\_table.Test\_Project\_Private\_Route\_Table1.id}"  }  # Private route table az2 association  resource "aws\_route\_table\_association" "Test\_Project\_az2\_Private01" {  subnet\_id = "${aws\_subnet.Test\_Project\_az2\_Private01.id}"  route\_table\_id = "${aws\_route\_table.Test\_Project\_Private\_Route\_Table2.id}"  }  # Private route table az2 association  resource "aws\_route\_table\_association" "Test\_Project\_az2\_Private02" {  subnet\_id = "${aws\_subnet.Test\_Project\_az2\_Private02.id}"  route\_table\_id = "${aws\_route\_table.Test\_Project\_Private\_Route\_Table2.id}"  }  # Private route table az2 association  resource "aws\_route\_table\_association" "Test\_Project\_az2\_Private03" {  subnet\_id = "${aws\_subnet.Test\_Project\_az2\_Private03.id}"  route\_table\_id = "${aws\_route\_table.Test\_Project\_Private\_Route\_Table2.id}"  }  # Connect IGW + Publice-Subnet  resource "aws\_route" "Test\_Project\_Internet\_GW" {  route\_table\_id = "${aws\_vpc.Test\_Project\_VPC.default\_route\_table\_id}"  destination\_cidr\_block = "0.0.0.0/0"  gateway\_id = "${aws\_internet\_gateway.Test\_Project\_Internet\_GW.id}"  } |

* NAT 게이트웨이



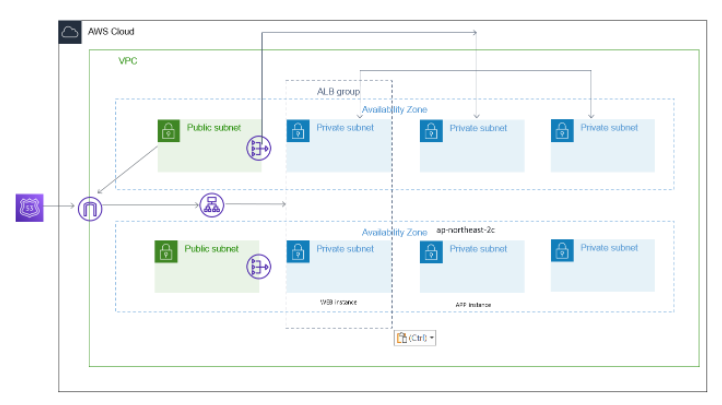
|  |
| --- |
| # Create Elastic-IP 1  resource "aws\_eip" "Test\_Project\_Elastic\_IP01" {  vpc = true  tags = {  Name = "Test\_Project\_Elastic\_IP01"  }  }  # Create NAT Gateway-az1  resource "aws\_nat\_gateway" "Test\_Project\_Nat\_Gateway01" {  allocation\_id = "${aws\_eip.Test\_Project\_Elastic\_IP01.id}"  subnet\_id = "${aws\_subnet.Test\_Project\_az1\_Public01.id}"  tags = {  Name = "Test\_Project\_Nat\_Gateway01"  }  }  # Create Elastic-IP 2  resource "aws\_eip" "Test\_Project\_Elastic\_IP02" {  vpc = true  tags = {  Name = "Test\_Project\_Elastic\_IP02"  }  }  # Create NAT Gateway-az2  resource "aws\_nat\_gateway" "Test\_Project\_Nat\_Gateway02" {  allocation\_id = "${aws\_eip.Test\_Project\_Elastic\_IP02.id}"  subnet\_id = "${aws\_subnet.Test\_Project\_az2\_Public02.id}"  tags = {  Name = "Test\_Project\_Nat\_Gateway02"  }  }  # Connect Private-Subnet-RT - az1-NAT  resource "aws\_route" "Test\_Project\_Nat\_Gateway01" {  route\_table\_id = "${aws\_route\_table.Project-Private-az1-RT.id}"  destination\_cidr\_block = "0.0.0.0/0"  nat\_gateway\_id = "${aws\_nat\_gateway.Test\_Project\_Nat\_Gateway01.id}"  }  # Connect Private-Subnet-RT - az2-NAT  resource "aws\_route" "Test\_Project\_Nat\_Gateway02" {  route\_table\_id = "${aws\_route\_table.Project-Private-az2-RT.id}"  destination\_cidr\_block = "0.0.0.0/0"  nat\_gateway\_id = "${aws\_nat\_gateway.Test\_Project\_Nat\_Gateway02.id}"  } |

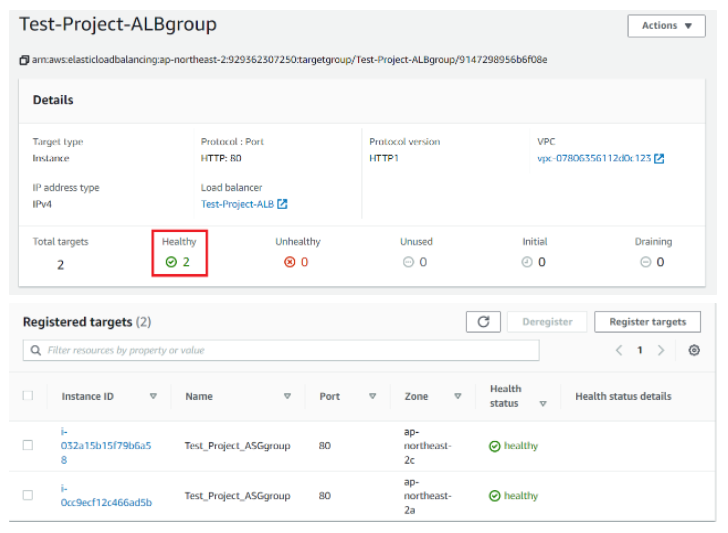
* Route53



|  |
| --- |
| # Create Route 53  resource "aws\_route53\_zone" "test-project.tk" {  name = "test-project.tk"  comment = "test-project.tk"  }  resource "aws\_route53\_zone" "main" {  name = "test-project.tk"  }  resource "aws\_route53\_zone" "dev" {  name = "dev.test-project.tk"  tags = {  Name = "Test\_project\_domain"  }  }  resource "aws\_route53\_record" "dev-ns" {  zone\_id = "${aws\_route53\_zone.main.zone\_id}"  name = "dev.example.com"  type = "NS"  ttl = "30"  records = [  "${aws\_route53\_zone.dev.name\_servers.0}",  "${aws\_route53\_zone.dev.name\_servers.1}",  "${aws\_route53\_zone.dev.name\_servers.2}",  "${aws\_route53\_zone.dev.name\_servers.3}",  ]  } |

* ALB





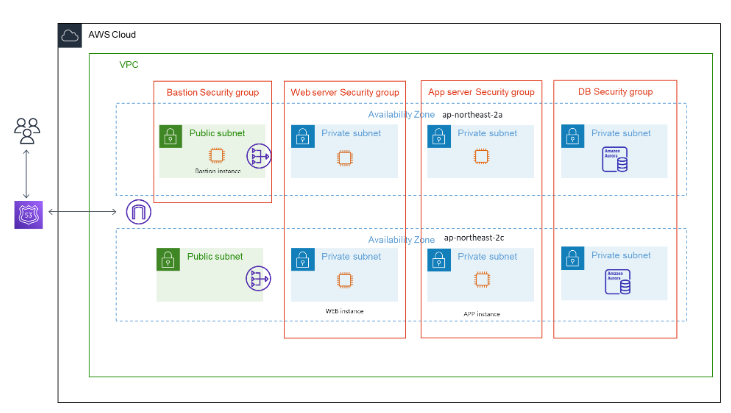
|  |
| --- |
| resource "aws\_alb" "Test-ALB" {  name = "Test-Project-ALB"  internal = false  load\_balancer\_type = "application"  security\_groups = [ aws\_security\_group.ALB\_Security.id]  subnets = [aws\_subnet.Test\_Project\_az1\_Public01.id,  aws\_subnet.Test\_Project\_az2\_Public02.id]  enable\_cross\_zone\_load\_balancing = true  }  resource "aws\_alb\_target\_group" "test-alb-tg" {  name = "Test-Project-ALBgroup"  port = 80  protocol = "HTTP"  vpc\_id = "${aws\_vpc.Test\_Project\_VPC.id}"  }  resource "aws\_alb\_target\_group\_attachment" "privateInstance01" {  target\_group\_arn = aws\_alb\_target\_group.test-alb-tg.arn  target\_id = aws\_instance.Web\_Server\_Instance.id  port = 80  }  resource "aws\_alb\_listener" "test\_listener" {  load\_balancer\_arn = aws\_alb.Test-ALB.arn  port = 80  protocol = "HTTP"  default\_action {  type = "forward"  target\_group\_arn = aws\_alb\_target\_group.test-alb-tg.arn  }  } |

5.3 보안 구축

* Keypair 생성 : EC2용 SSH 키페어 정의(새로운 키 페어 생성)

|  |
| --- |
| resource "aws\_key\_pair" "Test\_Project\_Key" {  key\_name = "Test\_Project\_Key"  public\_key = file("/root/.ssh/Test\_Project\_Key.pub")  } |

* 보안 그룹



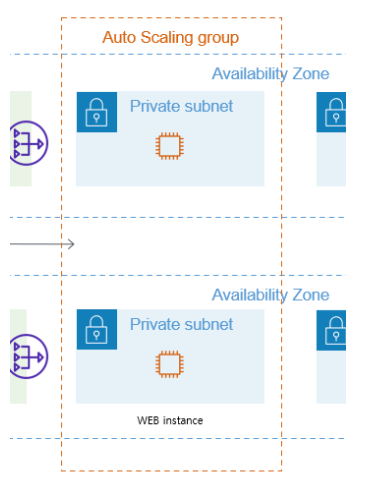
|  |
| --- |
| # Create Bastion SG  resource "aws\_security\_group" "Bastion\_Security" {  name = "Bastion\_Security"  description = "Bastion\_Security"  vpc\_id = aws\_vpc.Test\_Project\_VPC.id  tags = {  Name = "Bastion\_Security"  }  ingress {  from\_port = 22  to\_port = 22  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  description = "SSH"  }  }  # Create WebServer SG  resource "aws\_security\_group" "Web\_Server\_Security" {  name = "Web\_Server\_Security"  description = "Web\_Server\_Security"  vpc\_id = aws\_vpc.Test\_Project\_VPC.id  tags = {  Name = "Web\_Server\_Security"  }  ingress {  from\_port = 22  to\_port = 22  protocol = "tcp"  security\_groups = [ aws\_security\_group.Bastion\_Security.id]  description = "SSH"  }  ingress {  from\_port = 80  to\_port = 80  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  description = "HTTP"  }  ingress {  from\_port = -1  to\_port = -1  protocol = "icmp"  security\_groups = [ aws\_security\_group.Bastion\_Security.id]  description = "Ping test"  }  ingress {  from\_port = 443  to\_port = 443  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  description = "HTTPS"  }  }  # Create Database SG  resource "aws\_security\_group" "DB\_Security" {  name = "DB\_Security"  description = "DB\_Security"  vpc\_id = aws\_vpc.Test\_Project\_VPC.id  tags = {  Name = "DB\_Security"  }  ingress {  from\_port = 3306  to\_port = 3306  protocol = "tcp"  security\_groups = [ aws\_security\_group.Bastion\_Security.id]  description = "Bastion Host"  }  ingress {  from\_port = 3306  to\_port = 3306  protocol = "tcp"  security\_groups = [ aws\_security\_group.Web\_Server\_Security.id]  description = "WebServer Host"  }  }  # Create ApplicationServer SG  resource "aws\_security\_group" "App\_Server\_Security" {  name = "App\_Server\_Security"  description = "App\_Server\_Security"  vpc\_id = aws\_vpc.Test\_Project\_VPC.id  tags = {  Name = "App\_Server\_Security"  }  ingress {  from\_port = 22  to\_port = 22  protocol = "tcp"  security\_groups = [ aws\_security\_group.Bastion\_Security.id]  description = "SSH"  }  }  # Create ALB SG  resource "aws\_security\_group" "ALB\_Security" {  name = "ALB\_Security"  description = "ALB\_Security"  vpc\_id = aws\_vpc.Test\_Project\_VPC.id  tags = {  Name = "ALB\_Security"  }  ingress {  from\_port = 22  to\_port = 22  protocol = "tcp"  security\_groups = [ aws\_security\_group.Bastion\_Security.id]  description = "SSH"  }  ingress {  from\_port = 80  to\_port = 80  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  description = "HTTP"  }  ingress {  from\_port = -1  to\_port = -1  protocol = "icmp"  security\_groups = [ aws\_security\_group.Bastion\_Security.id]  description = "Ping test"  }  ingress {  from\_port = 443  to\_port = 443  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  description = "HTTPS"  }  } |

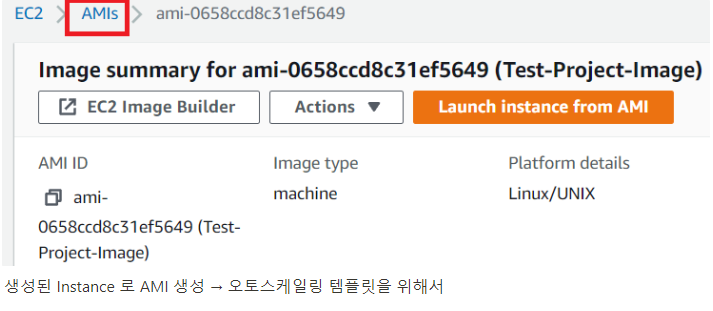
5.4 서버 구축

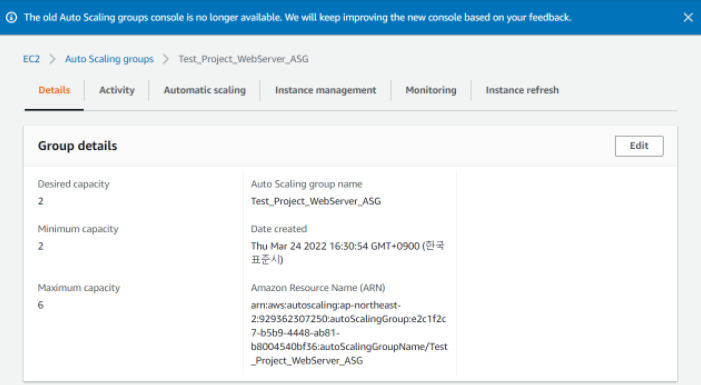
* 인스턴스

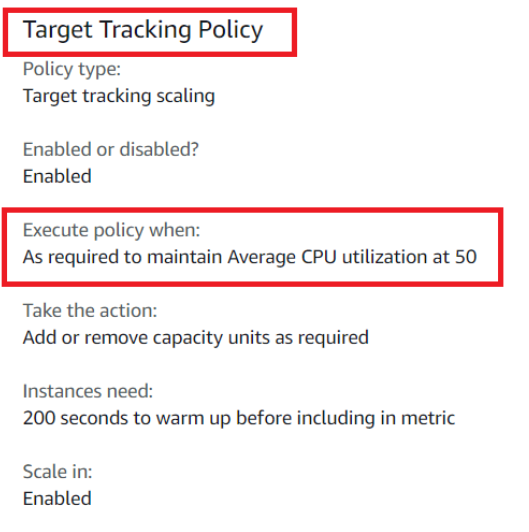
|  |
| --- |
| # Create Bastion EC2 Instance  resource "aws\_instance" "Bastion Instance" {  ami = "ami-033a6a056910d1137"  instance\_type = "t2.micro"  tags = {  Name = "Bastion"  }  key\_name = "Test\_Project\_Key"  associate\_public\_ip\_address = true  subnet\_id = aws\_subnet.Test\_Project\_az1\_Public01.id  vpc\_security\_group\_ids = [  aws\_security\_group.Bastion\_Security.id  ]  }  # Create WebServer EC2 Instance  resource "aws\_instance" "Web\_Server\_Instance" {  ami = "ami-033a6a056910d1137"  instance\_type = "t2.micro"  tags = {  Name = "Web Server"  }  key\_name = "Test\_Project\_Key"  associate\_public\_ip\_address = false  subnet\_id = aws\_subnet.Test\_Project\_az1\_Private01.id  vpc\_security\_group\_ids = [  aws\_security\_group.Web\_Server\_Security.id  ]  user\_data = <<-EOF  #!/bin/bash  echo "\*\*\* Installing apache2"  sudo apt update -y  sudo apt install apache2 -y  echo "\*\*\* Completed Installing apache2"  EOF  }  # Create AppServer EC2 Instance - az1  resource "aws\_instance" "Web\_Application\_Server\_Instance\_az1" {  ami = "ami-033a6a056910d1137"  instance\_type = "t2.micro"  tags = {  Name = "App Server1"  }  key\_name = "Test\_Project\_Key"  associate\_public\_ip\_address = false  subnet\_id = aws\_subnet.Test\_Project\_az1\_Private02.id  vpc\_security\_group\_ids = [  aws\_security\_group.App\_Server\_Security.id  ]  }  # Create AppServer EC2 Instance - az2  resource "aws\_instance" "Web\_Application\_Server\_Instance\_az2" {  ami = "ami-033a6a056910d1137"  instance\_type = "t2.micro"  tags = {  Name = "App Server2"  }  key\_name = "Test\_Project\_Key"  associate\_public\_ip\_address = false  subnet\_id = aws\_subnet.Test\_Project\_az2\_Private02.id  vpc\_security\_group\_ids = [  aws\_security\_group.App\_Server\_Security.id  ]  } |

* 오토스케일링



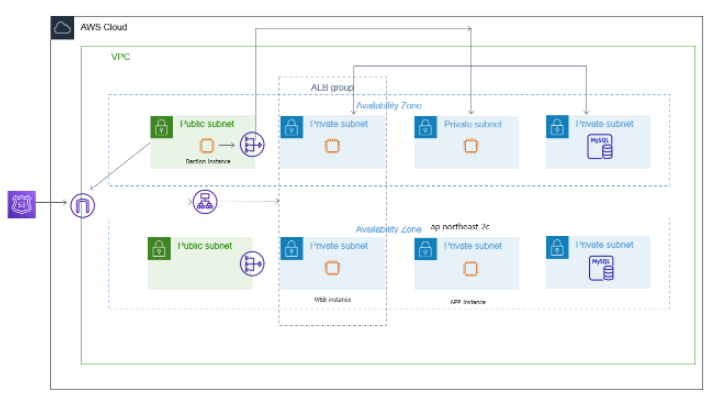






|  |
| --- |
| # Creating launch template  resource "aws\_launch\_template" "test\_launch\_template" {  name = "test\_launch\_template"  image\_id = "ami-0658ccd8c31ef5649" # Web Server AMI 3  instance\_type = "t2.micro"  key\_name = "Test\_Project\_Key"  vpc\_security\_group\_ids = [ aws\_security\_group.Web\_Server\_Security.id]  }  # Creating Auto Scaling group  resource "aws\_autoscaling\_group" "Test\_Project\_WebServer\_ASG" {  name = "Test\_Project\_WebServer\_ASG"  min\_size = 2  max\_size = 6  desired\_capacity = 2  health\_check\_grace\_period = 300  health\_check\_type = "EC2"  vpc\_zone\_identifier = [  aws\_subnet.Test\_Project\_az1\_Private01.id,  aws\_subnet.Test\_Project\_az2\_Private01.id]  target\_group\_arns = [aws\_alb\_target\_group.test-alb-tg.arn]  launch\_template {  id = aws\_launch\_template.test\_launch\_template.id  version = "$Latest"  }  tag {  key = "Name"  value = "Test\_Project\_WebServer\_ASG"  propagate\_at\_launch = true  }  } |

5.5 DB 구축



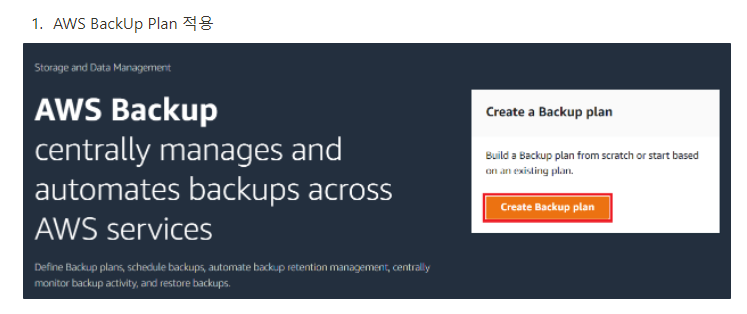
|  |
| --- |
| # RDS DB Instance 생성  resource "aws\_db\_instance" "Test\_Project\_DB" {  allocated\_storage = 10  engine = "mysql"  engine\_version = "8.0.20"  instance\_class = "db.t2.micro"  name = "terraformdb"  username = "admin"  password = "qwer1234"  identifier = "Test\_Project\_DB"  skip\_final\_snapshot = true  db\_subnet\_group\_name = "Test\_Project\_DB\_Subnet\_Group"  vpc\_security\_group\_ids = ["${aws\_security\_group.DB\_Security.id}"]  multi\_az = true #multi-AZ  } |

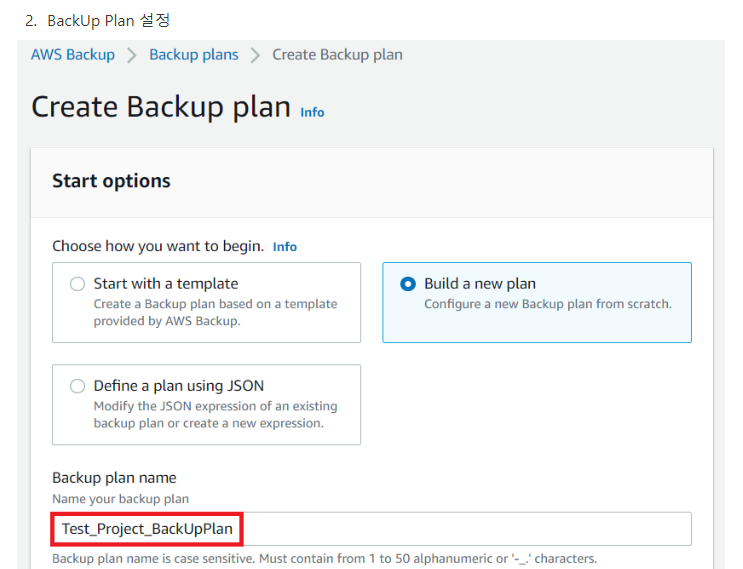
5.4 스토리지 구축

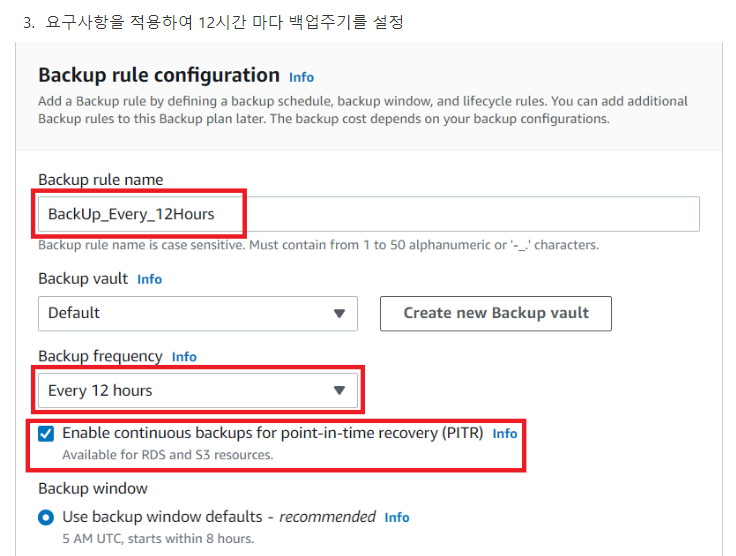
* S3 bucket

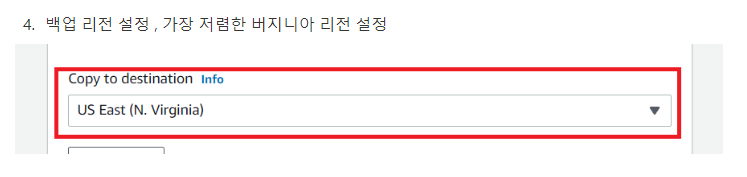
|  |
| --- |
| # Create S3 bucket  resource "aws\_s3\_bucket" "test-project.tk" {  bucket = "test-project.tk"  }  # Set versioning  resource "aws\_s3\_bucket\_acl" "test-project.tk-acl" {  bucket = aws\_s3\_bucket.test-project.tk.id  acl = "private"  }  resource "aws\_s3\_bucket\_versioning" "test-project.tk-versioning" {  bucket = aws\_s3\_bucket.test-project.tk.id  versioning\_configuration {  status = "Enabled"  }  }  locals {  s3\_origin\_id = "test-project.tk-origin"  }  # Set website configuration  resource "aws\_s3\_bucket\_website\_configuration" "test-project.tk-website" {  bucket = aws\_s3\_bucket.test-project.tk.bucket  index\_document {  suffix = "index.html"  }  error\_document {  key = "error.html"  } |

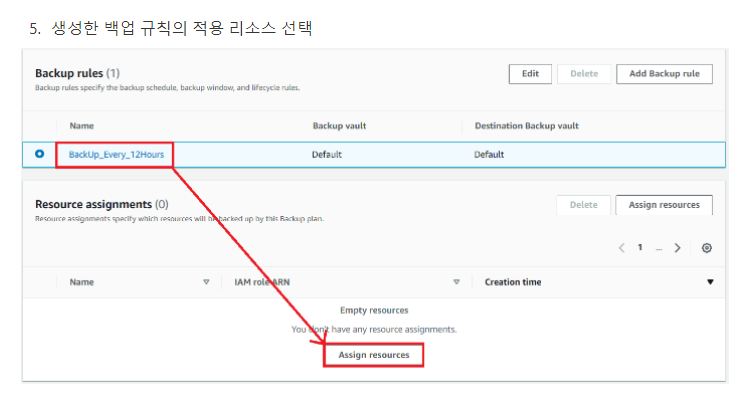
5.5 백업 플랜 구축

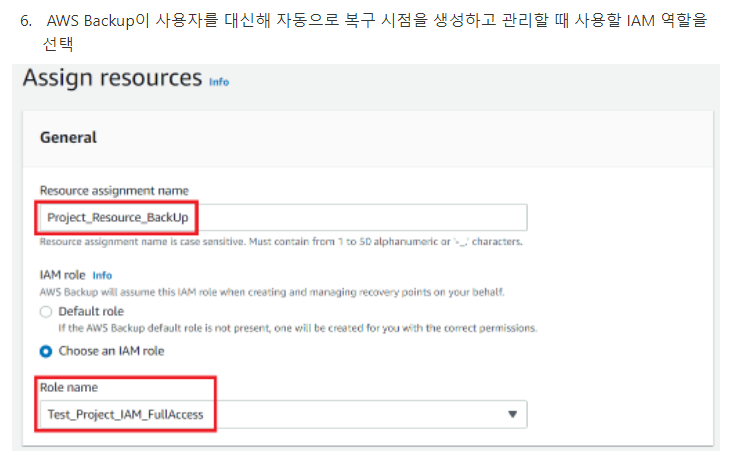




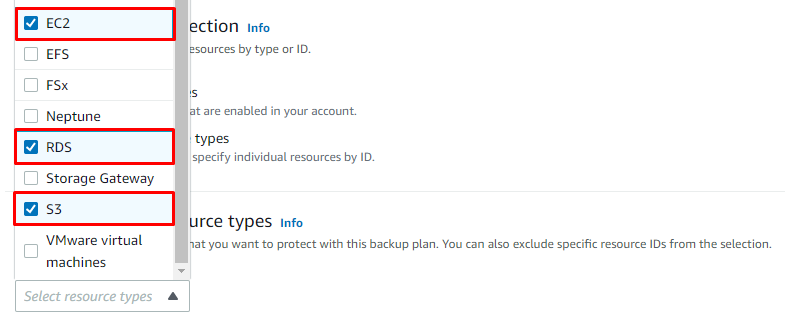


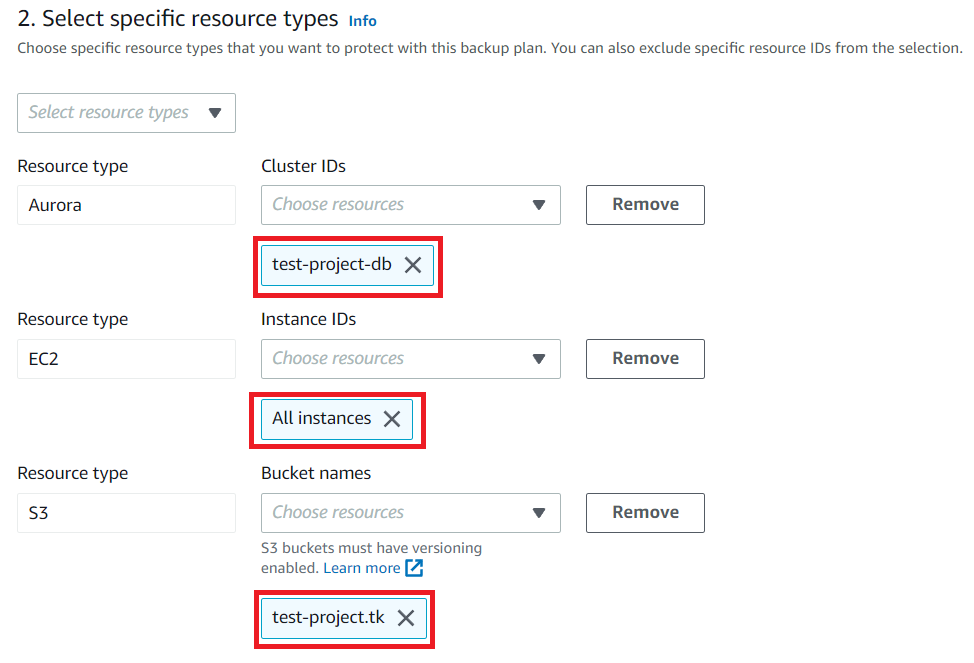






1. 백업플랜을 적용할 리소스 선택 : EC2, RDS, S3





RDS

6. 프로젝트 결과

6.1 결과

6.2 테스트 시나리오

기본 테스트 시나리오

고객이 도메인을 입력하여 웹페이지에 접속, 회원가입을 진행하면 Web 계층에서 회원정보를 Application 계층으로 넘겨준다. Application 계층에 존재하는 Service 로직이 회원정보를 처리하여 Database 계층으로 넘겨준다.

장애 대처 시나리오

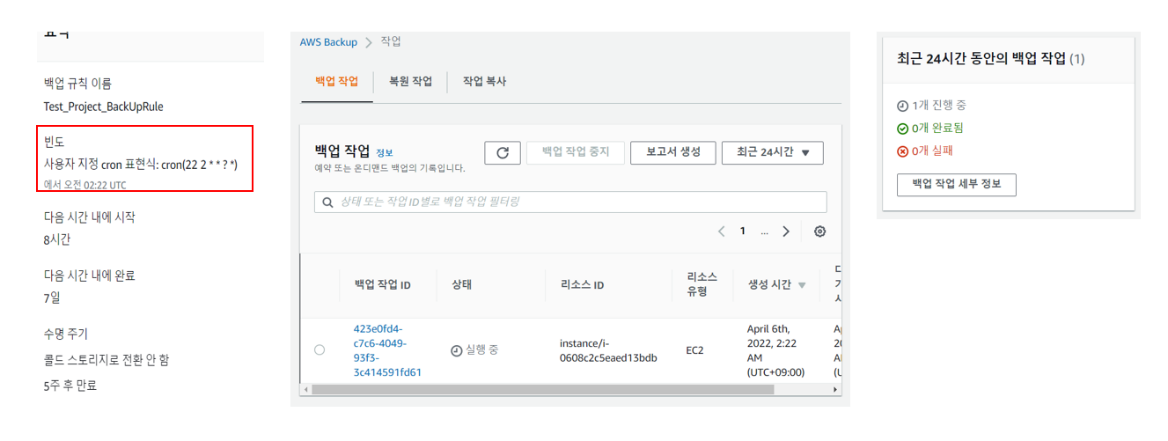
가용영역 2a , 2c 중 2a 를 사용하지 못 할 경우 가용영역 2c에 생성해둔 Web , App 서버는 그대로 실행되며 load balancer는 2a 가용영역의 인스턴스의 healthcheck를 진행하고 상태이상을 확인한 후 가용영역 2c의 오토스케일링을 진행한다.

가용영역 2a 에 존재하는 primaryDB 가 작동하지 않게 되면 가용영역 2c 에 존재하던 standbyDB 가 primary로 승격하여 메인 데이터베이스로써 역할을 한다.

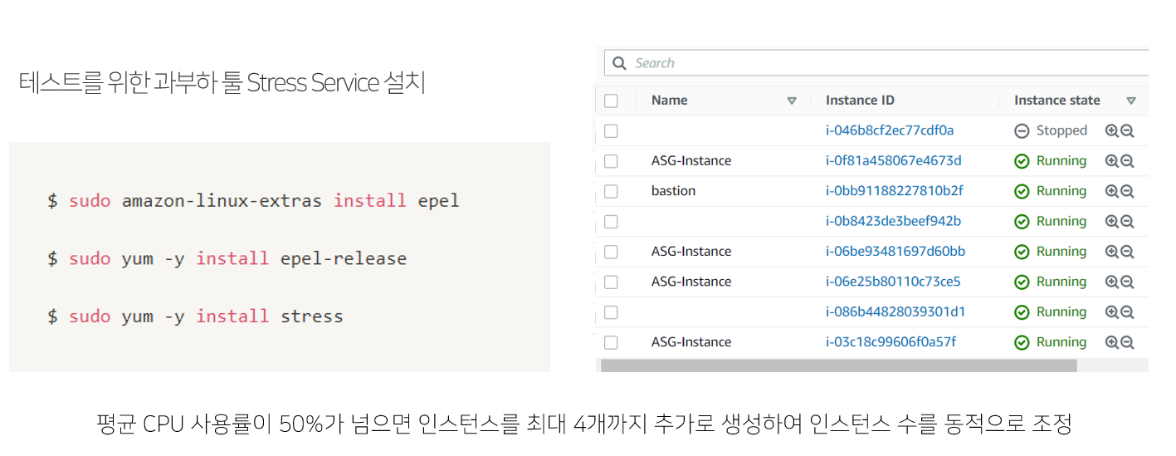
이후 백업정책에 의하여 자동 백업된 EC2 와 DB를 가지고 가용영역 2a 를 정상화 시킨후 복원하는 작업을 2시간 이내로 처리해준다.

6.3 검토

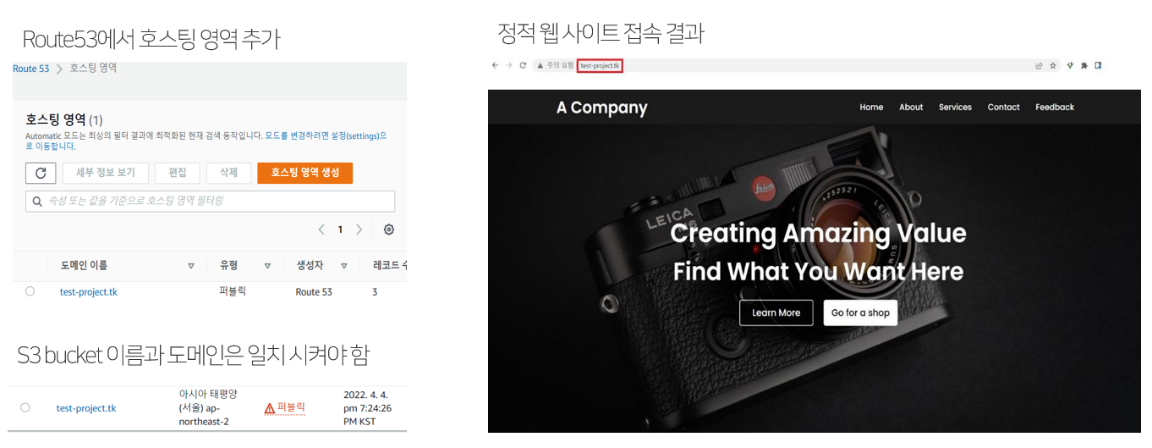
* 백업 계획 테스트 : AWS Back Up 으로 생성한 백업 계획이 잘 작동하는지에 대한 단위 테스트



* 과부하 테스트 : Autoscaling 이 제대로 작동하는지에 대한 과부하 테스트



* 웹 호스팅 테스트 : Route53 으로 생성한 도메인으로 S3 정적 웹페이지 호스팅 확인 테스트



6.4 자체 평가 및 후기

* 김민혁

|  |  |
| --- | --- |
| 프로젝트 전체 평가 | 조원들이 전체적인 틀을 잡아 수행하면 거기에 수정을 하거나 보충을 하는 역할을 하였다 |
| 프로젝트에서 잘됐던 점 | Terraform을 이용한 아키텍쳐 구축을 어느정도 성공적으로 구축함 |
| 프로젝트에서 부족했던 점 | 자신감 부족으로 의견을 잘 내지않아 프로젝트를 하는데 혼선을 빚게 만들었다. 의사소통을 적극적으로 하려는 노력이 필요하다고 생각했다. |

* 문광훈

|  |  |
| --- | --- |
| 프로젝트 전체 평가 | 문제정의서와 요구사항 설계서를 통일시키기 위해 노력했으며 그 과정이 잘 진행되었다. |
| 프로젝트에서 잘됐던 점 | 처음 다뤄보는 IaC tool인 Terraform 을 공부하는 계기가 되었고 구현 결과물도 성공적으로 나왔다. |
| 프로젝트에서 부족했던 점 | 100% 자동화를 하기엔 시간도 여력도 부족했었던 것이 많이 아쉬운 부분이고 , Terraform이 익숙치 않아 더 많은 서비스를 자동화하지 못한 부분이 매우 아쉽다. |

* 서현지

|  |  |
| --- | --- |
| 프로젝트 전체 평가 | 프로젝트가 비대면으로 진행되어 처음에는 의사소통에 문제가 생기는 등 여러 문제들이 생겨 힘들었다. 하지만 문제점을 이겨내고 잘 마무리할 수 있었다.  원하는 결과를 모두 얻지 못했지만 만족스러웠던 결과를 얻게 되게 되었고 프로젝트가 성공적으로 잘 끝낼 수 있었다. |
| 프로젝트에서 잘됐던 점 | 수업에서 배우지 않은 terraform이라는 새로운 기술들을 사용하게 되었는데 각자 따로 시간 내어 공부해서 성공적인 원하는 결과물을 얻을 수 있었다. |
| 프로젝트에서 부족했던 점 | 처음에 설계했던 내용들이 있었는데 시간이 촉박하고 terraform이 익숙하지 않아 전부 다 구현하지 못한 것이 아쉽다. 기회가 된다면 구현하지 못한 서비스들을 구현해봤으면 좋겠다. |

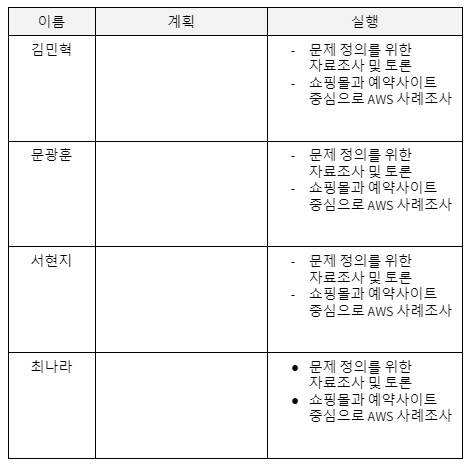
* 최나라

|  |  |
| --- | --- |
| 프로젝트 전체 평가 | 문제정의서와 요구사항 분석, 설계서 등 문서화 작업에 시간이 많이 소요되었으나 그만큼 좋은 결과가 만들어졌다. |
| 프로젝트에서 잘됐던 점 | 새로운 기술을 배우고 사용하는 좋은 계기가 되었고 조원들과의 협력으로 좋은 결과물이 나왔다. |
| 프로젝트에서 부족했던 점 | 프로젝트에 시간과 기술 사용에 대해 부족한 점이 있어 완벽하게 만들지 못해 아쉬웠으나 스스로 보완할 점을 알 수 있게된 계기가 되었다. |

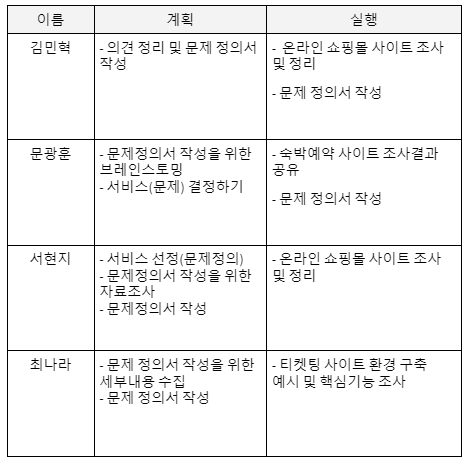
7. 부록

7.1 작업일지

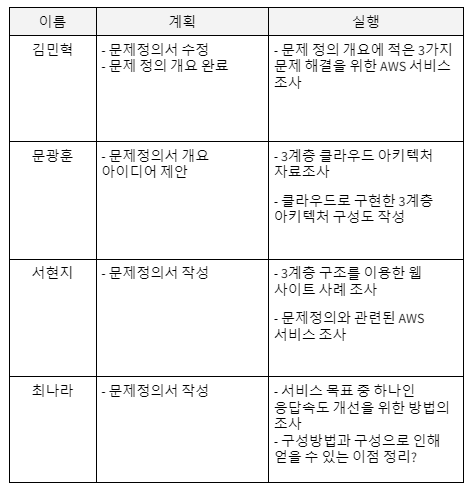
* 1주차
  + 3월11일 작업일지



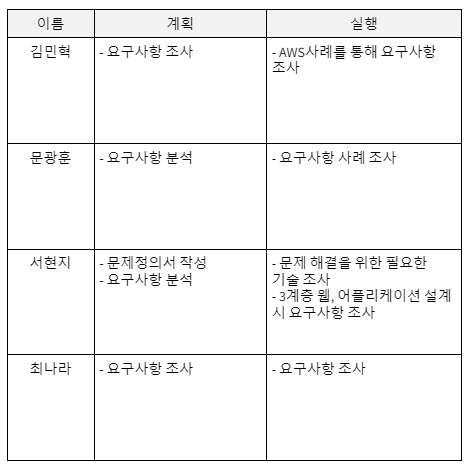
* + 3월 14일 작업일지



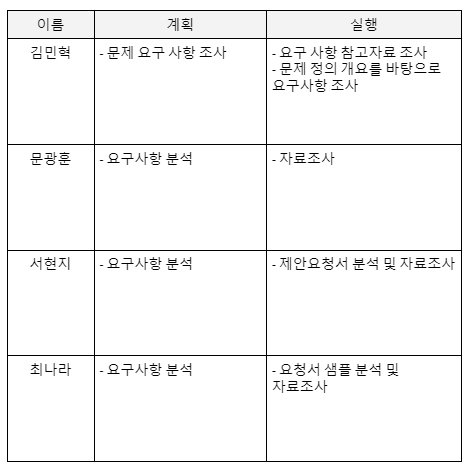
* + 3월 15일 작업일지



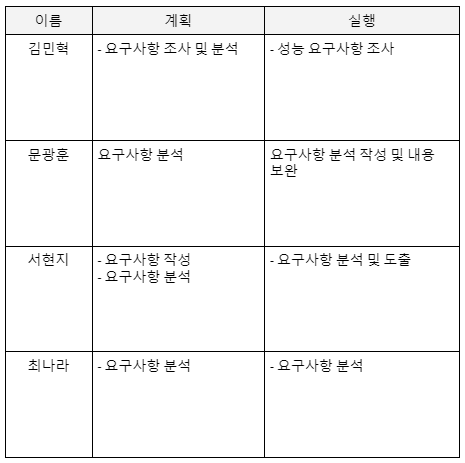
* + 3월 16일 작업일지



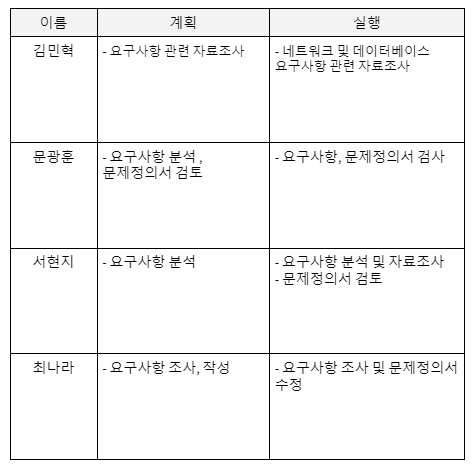
* + 3월 17일 작업일지



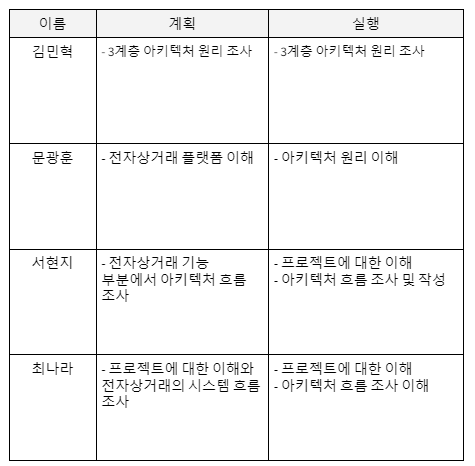
* + 3월 18일 작업일지



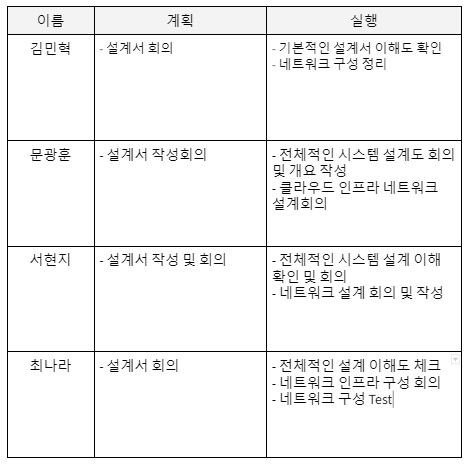
* 2주차
  + 3월 21일 작업일지



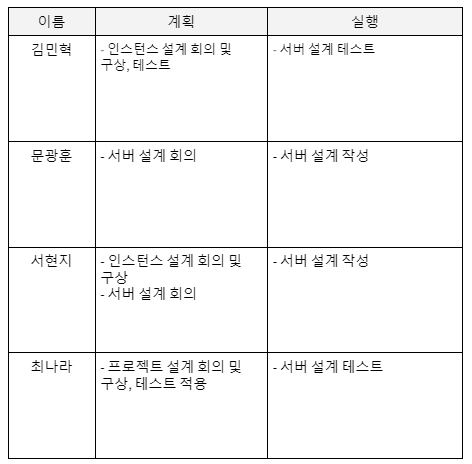
* + 3월 22일 작업일지



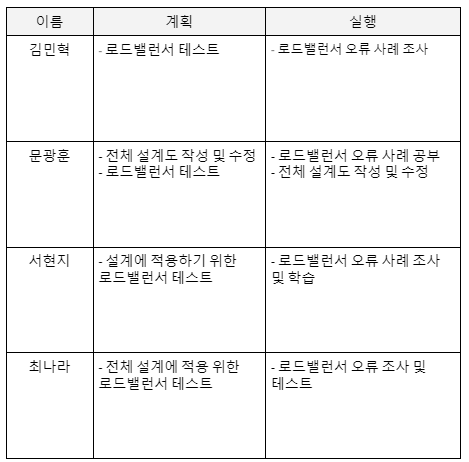
* + 3월 23일 작업일지



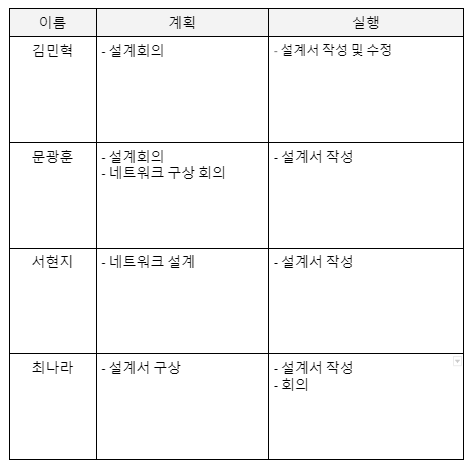
* + 3월 24일 작업일지



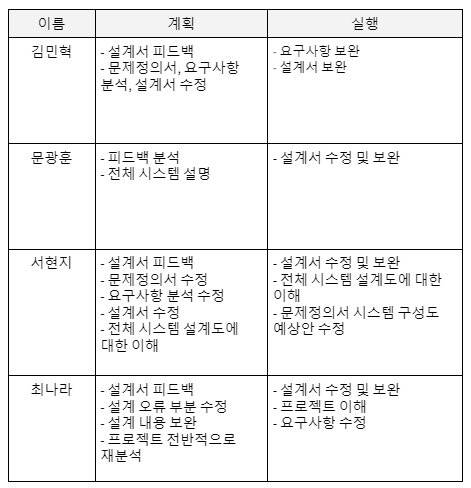
* + 3월 25일 작업일지



* 3주차
  + 3월 28일 작업일지



* + 3월 29일 작업일지



* + 3월 30일 작업일지



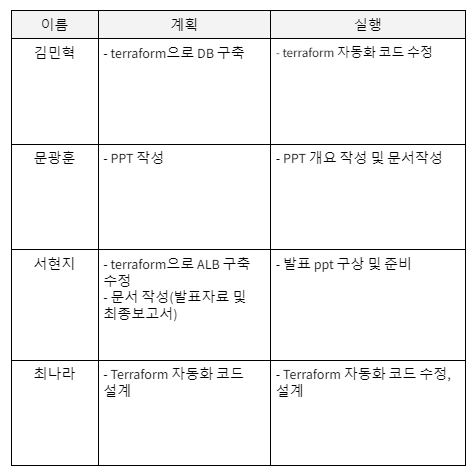
* + 3월 31일 작업일지



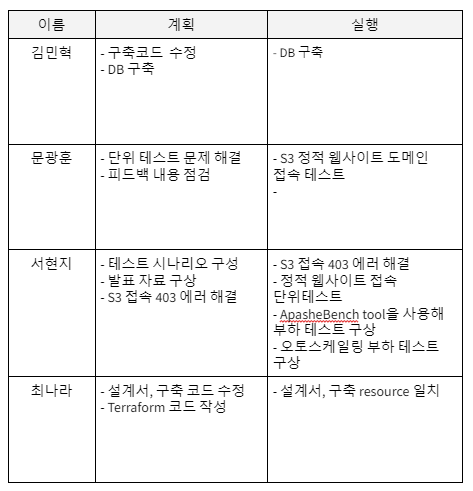
* + 4월 1일 작업일지



* 4주차
  + 4월 4일 작업일지



* + 4월 5일 작업일지



* + 4월 6일 작업일지

