AWS 쇼핑몰 마이그레이션 및 고도화

아키텍처 변경 이력서

2023. 11. 14.

목 차

그림 목차

그림 목차 항목을 찾을 수 없습니다.

1. 아키텍처 변경 이력서 개요

1.1. 소개

본 아키텍처 변경 이력서는 "AWS 쇼핑몰 마이그레이션 및 고도화" 프로젝트 진행 과정에서 AWS 인프라 및 사용 기술에 대한 변경 이력을 기술한다. 해당 변경 이력에 대한 서비스 영향 평가의 경우 모든 영향 평가가 완료되어 서비스 기능적 오류를 제외한 AWS 인프라 운영과 관련된 사항은 모두 정상인 것을 전제로 한다.

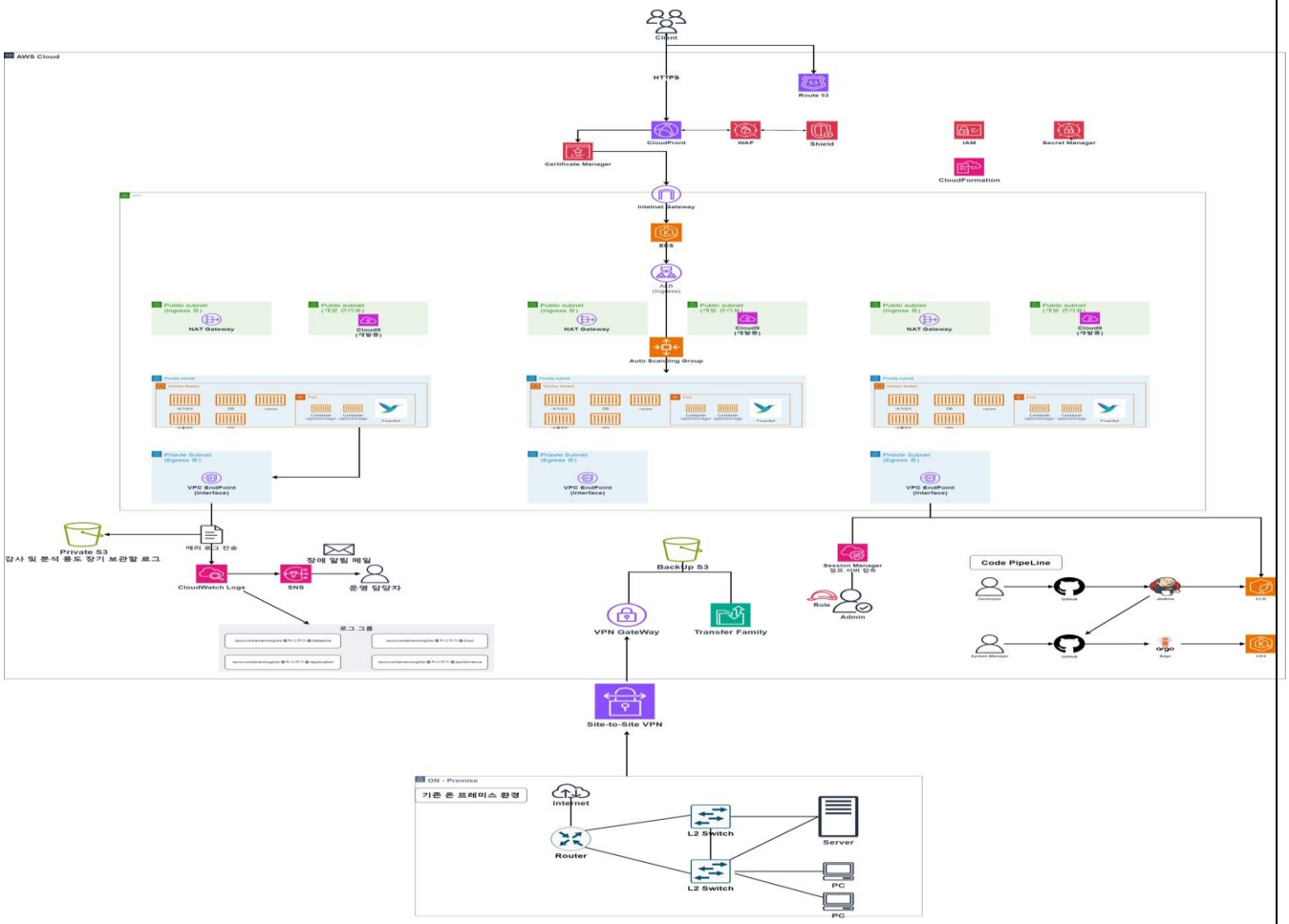
1.2. AWS 기술 목록

항목	구현 기술	세부 사항
인프라 (서비스)	마이크로서비스 아키텍처(MSA)	쇼핑몰 시스템 서비스를 독립적으로 구성하여 각 서비스를 컨테이너화
	AWS Elastic Kubernetes Service(EKS)	EKS 클러스터 환경에서 컨테이너화된 각 서비스 배포 및 관리
	AWS Application Load Balancer(ALB)	로드밸런서를 통한 각 서비스별 트래픽 부하 분산
데이터베이스	Maria DB Container	MariaDB 인스턴스를 Docker 컨테이너로 구성하여, 확장성, 이식성 강화
	DB 읽기 복제본	읽기, 쓰기 요청을 Master DB, Read Replica 로 분산처리 구현
CI/CD	Github	각 서비스별 코드 형상 관리
	Jenkines	Jenkins 는 코드 변경사항을 자동으로 감지하고 빌드 파이프라인을 실행
	Argo CD	GitHub 에 저장된 구성 파일을 기반으로 클러스터 상태를 동기화하여 배포 과정을 간소화
	AWS Elastic Container Registry(ECR)	각 서비스별로 빌드된 컨테이너 이미지를 보관 및 관리
모니터링	Prometheus	AWS EKS 로 관리되는 서버 및 컨테이너에 대한 데이터 지표 수집
	Grafana	Prometheus 와 통합하여 데이터 시각화 및 대시보드 구성
마이그레이션	스테이징 서버	기존 온프레미스 부하를 줄이기 위한 임시 서버 구성
	VPN	온프레미스, 클라우드간의 데이터를 암호화 하여 안전한 통신을 할 수 있도록 구성

2. 아키텍처 변경 이력

2.1. 1 차 아키텍처 변경 이력 – 버전 1.00

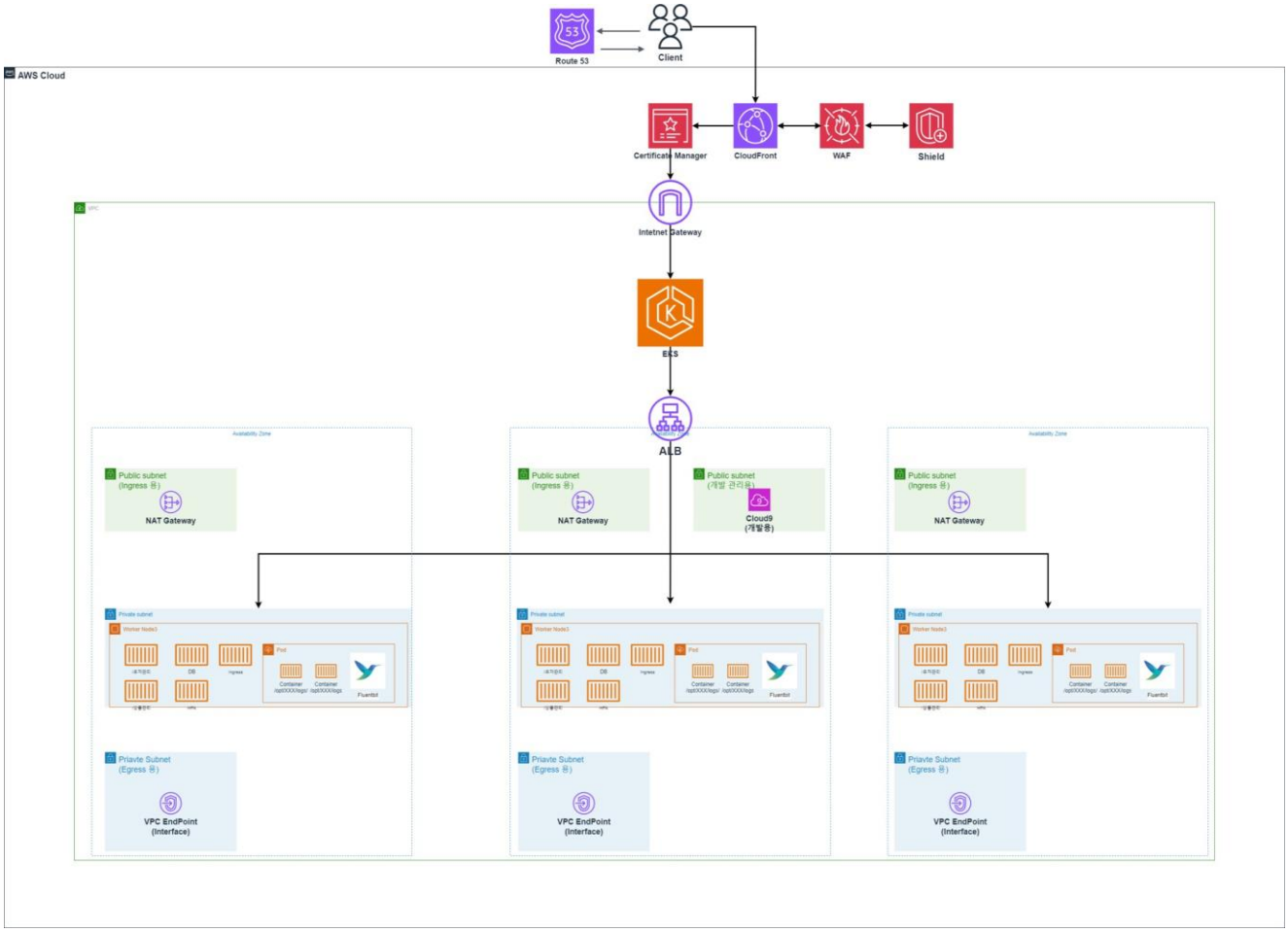
- 초기 아키텍처 구성으로 변경 사항 없음



2.2. 2 차 아키텍처 변경 이력 – 버전 1.01

- **CI/CD**
  - 변경 내역
    - CI / CD 파이프 라인 흐름도 제거
  - 변경 사유
    - 흐름도가 아닌 마이그레이션 후 구성도만 보여주기 위해 제거
- **모니터링**
  - 변경 내역
    - 모니터링 흐름도 제거
  - 변경 사유
    - 흐름도가 아닌 마이그레이션 후 구성도만 보여주기 위해 제거
- **인프라(서비스)**
  - 변경 내역
    - 기존에 표현했던 온 프레미스 망 구성도 제거
    - 기존 각 Public 3 개의 서브넷에서 각각 Cloud9 사용하던 것 1 개로 변경

- 변경 사유
  - 흐름도가 아닌 마이그레이션 후 구성도만 보여주기 위해 제거
  - 개발용 Cloud9 3 개 구성 불필요하여 제거



2.3. 3 차 아키텍처 변경 이력 – 버전 1.02

- CI / CD
  - 변경 내역
    - 기존에 없앴던 CI / CD 흐름도를 운영 관리자 접근 로직도 추가
  - 변경 사유
    - 배포 과정에 운영 관리자의 접근 로직을 가시성 있게 표현하기 위해서 수정
- **모니터링**
  - 변경 내역
    - 인터페이스 VPC 엔드포인트를 활용하여 Amazon CloudWatch Logs 로 직접 로그를 전송하도록 구성

- ◆ Amazon Simple Notification Service(SNS)를 이용하여, 정의된 임계값을 초과하는 이벤트 발생 시 알람을 발송하도록 설정
- ◆ 게이트웨이 VPC 엔드포인트를 통해 Amazon S3 로 로그를 백업하고, 이 로그들이 자동으로 Amazon S3 Glacier 로 전송되도록 수명 주기 정책을 구성

■ 변경 사유

- ◆ 기존 구성은 복수의 인터페이스 VPC 엔드포인트가 필요했으나, 이는 관리 복잡성과 비용 측면에서 비효율적이었습니다. 통합된 인터페이스 VPC 엔드포인트를 통한 로그 전송은 관리의 단순화와 불필요한 비용을 최소화 시키고 네트워크 트래픽의 최적화 환경 구성
- ◆ 데이터 전송 속도와 보안성 향상을 위해 로그 데이터의 중요성과 증가하는 데이터 양을 고려해 게이트웨이 VPC 엔드포인트를 통한 로그 백업 방식 구성
- ◆ S3 수명 주기 정책을 적용해 자동화 된 데이터 이동으로 장기적인 데이터 효율적으로 저장 및 관리

● **인프라(서비스)**

■ 변경 내역

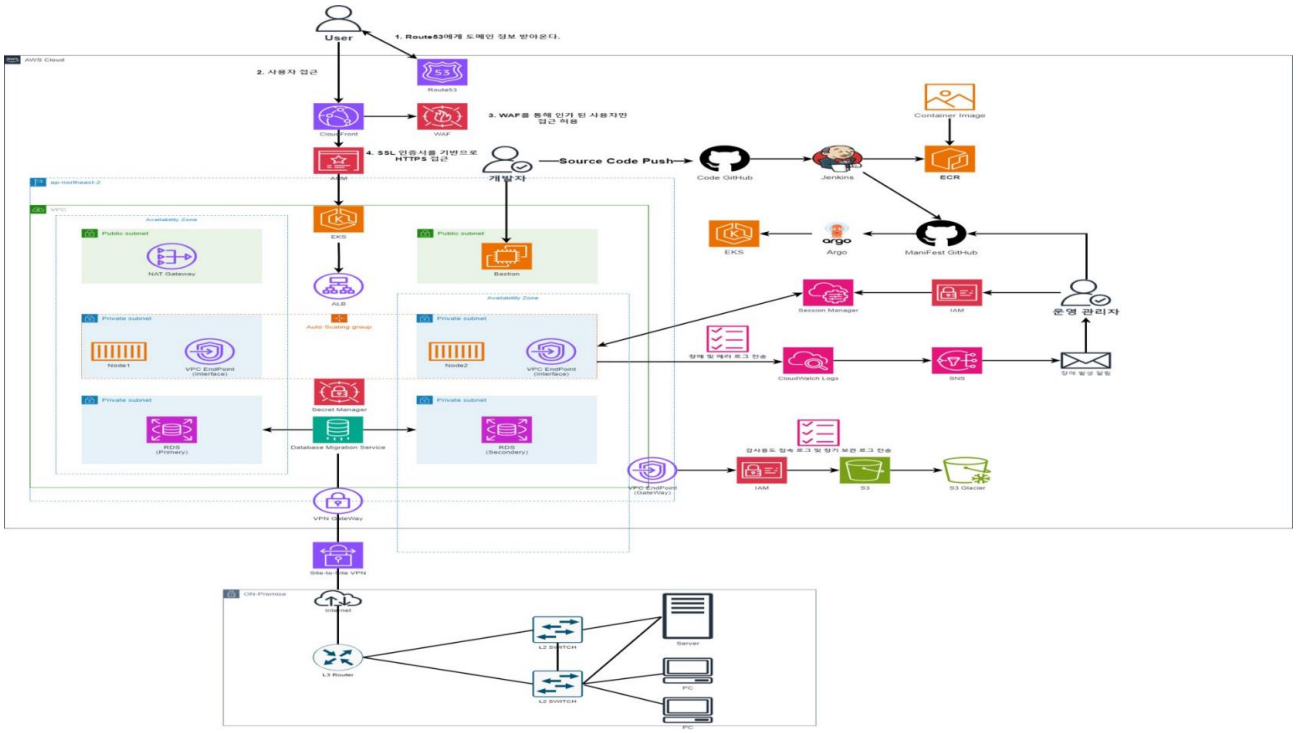
- ◆ 이전의 3 개의 가용 영역을 2 개로 축소
- ◆ 여러 개의 NAT Gateway 를 1 개로 단일로 변경
- ◆ Cloud9 에서 아닌 Bastion 호스트로 구성 변경
- ◆ 마이그레이션 될 온프레미스 망 다시 확인되도록 구성 변경
- ◆ EKS 내부 구성을 세부 파드 단위에서 더 높은 수준인 노드 단위로 구성 단순화
- ◆ Session Manager 를 이용해 Bastion 장애 상황 시에도 바로 프라이빗 서버로 보안성 있는 접근 가능하도록 구성

■ 변경 사유

- ◆ 기존 가용 영역 3 개는 과도한 것으로 판단되어 2 개의 AZ 로 축소하여 관리 단순화
- ◆ 여러 NAT Gateway 를 사용하는 것보다 단일 NAT Gateway 를 사용해 보안성을 강화하고 운영 단순화
- ◆ 온 프레미스 망을 확인하고 마이그레이션 시 필요한 서비스를 확인할 수 있기 위해 변경
- ◆ EKS 내부 구성을 파드 단위에서 노드 단위로 단순화하여 가시성을 향상시키기 위해 변경

● **데이터베이스**

- 변경 내역
  - 컨테이너화 된 DB 에서 관리형 DB 서비스 RDS 로 변경
  - 이전 구성도에서 제거되었던 DB 마이그레이션 흐름도를 다시 추가
  - Amazon Database Migration Service(DMS)를 사용하여 RDS 로의 데이터 마이그레이션 구현.
- 변경 사유
  - Amazon RDS 를 사용함으로써 운영 및 관리의 복잡성을 줄이고, 자동화된 백업, 패치 관리, 그리고 확장성 제공과 같은 추가적인 이점을 활용하기 위해 변경
  - 데이터베이스 마이그레이션 과정의 중요성을 감안하여, 마이그레이션 흐름도를 다시 포함시키는 것이 시스템의 이해도를 높이고, 마이그레이션 계획을 명확히 하는 데 필수적이라고 판단
  - Data Sync 와 DMS 중에서, DMS 를 선택한 주된 이유는 다양한 소스 데이터베이스를 지원하고, 중단 없는 실시간 데이터 복제를 제공하여 비즈니스 연속성을 유지하기 위해 변경



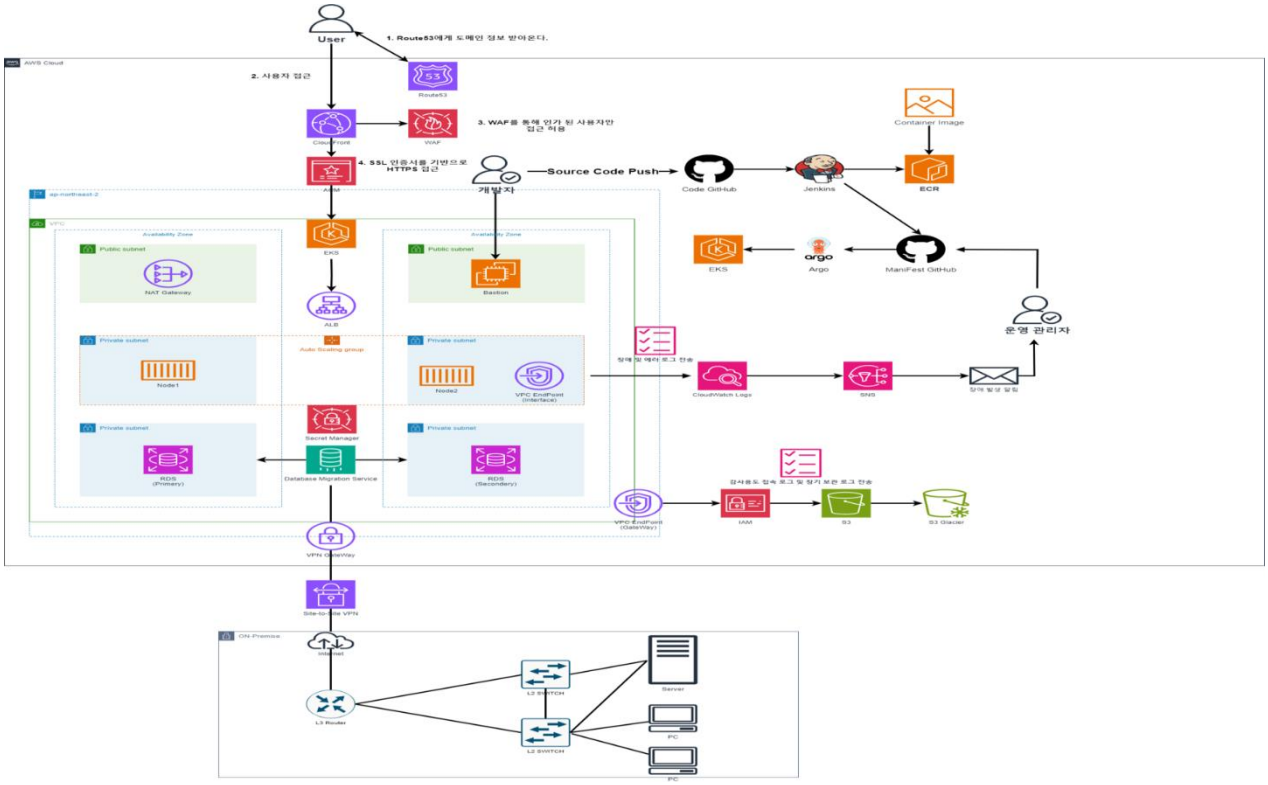
2.4. 4 차 아키텍처 변경 이력 – 버전 1.03

- **인프라(서비스)**
  - 변경 내역

- ◆ Session Manager 를 사용한 접근 방식을 제거하고 Bastion 호스트를 통한 단일 접근 방식 선택
- ◆ 인터페이스 VPC 엔드포인트를 하나의 서브넷에만 배치하도록 구성

■ 변경 사유

- ◆ Bastion 호스트를 사용해 관리자의 접근을 단순화하고 관리 용이하도록 변경
- ◆ VPC 내부에서 하나의 엔드포인트로 통신 가능하기 때문에 불필요한 비용 최소화 및 관리 용이성을 위해서 변경



2.5. 5 차 아키텍처 변경 이력 – 버전 1.04

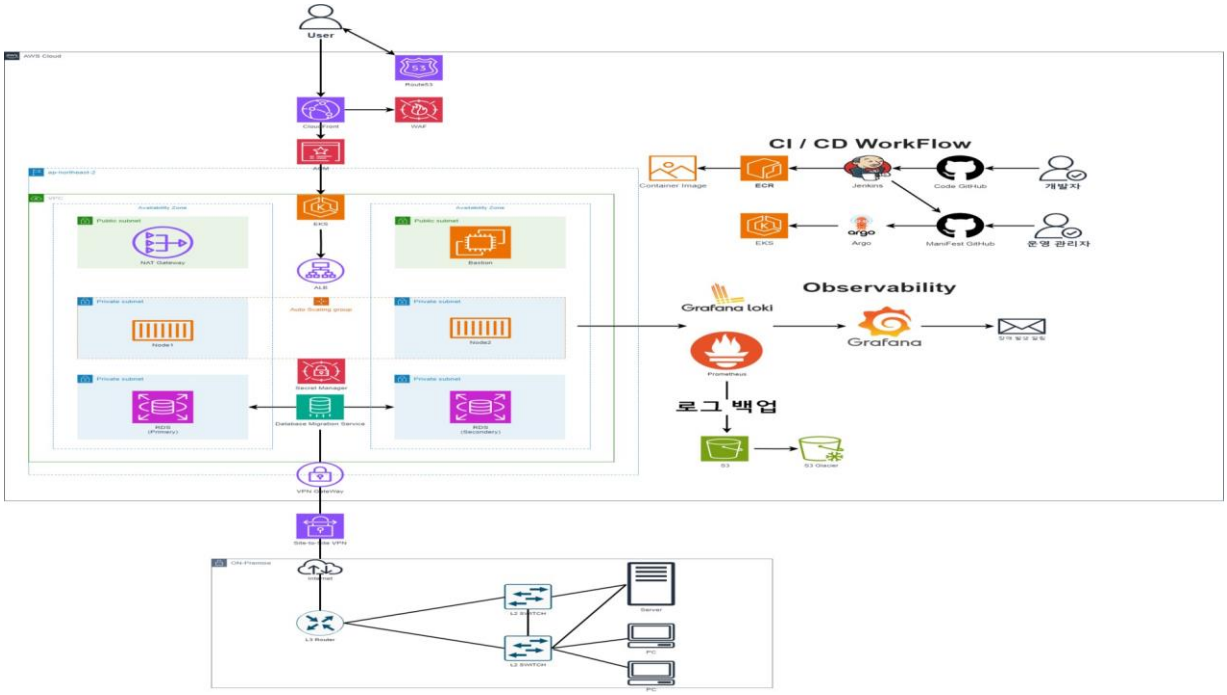
● **모니터링**

■ 변경 내역

- ◆ AWS CloudWatch 와 AWS SNS 를 활용하여 로그 및 지표를 수집하고 장애 발생 시 알람을 전송하는 구성이었지만 새로운 아키텍처에서는 Grafana Loki 를 통한 로그 집계 및 Prometheus 와 Grafana 를 사용해 지표를 수집하고 경보 알람을 전송하도록 변경

■ 변경 사유

- ◆ Grafana Loki 와 Prometheus 를 통합하여 사용함으로써 로그와 지표를 한 곳에서 관리할 수 있고 이는 모니터링의 일관성을 향상시키고, 관리자가 시스템 상태에 대한 더 깊은 통찰력을 얻을 수 있기 때문에 변경



2.6. 6 차 아키텍처 변경 이력 – 버전 1.05

- **모니터링**
  - 변경 내역
    - AWS Managed Prometheus 와 AWS Managed Grafana 를 사용해 메트릭 지표 수집 및 시각화 구성
    - AWS CloudTrail 을 이용해 AWS 서비스 내부의 감사 로그를 수집해 S3 로 저장하도록 구성
    - FluentBit 를 이용해 AWS S3 와 AWS CloudWatch 로 각각 로그를 라우팅하도록 구성
    - AWS CloudWatch 경보 알람과 AWS SNS 와 AWS ChatBot 을 이용해 특정 임계치에 경보 알람 실시간으로 Slack 으로 보내도록 구성
  - 변경 사유
    - 기존 고객사 소수의 인원으로 운영되기에 Prometheus 서버가 죽는 경우 대처 할 인력이 없고 관리형 서비스를 통한 운영 오버헤드를 줄일 수 있고 AWS 로 이전 할 계획이기 때문에 AWS 서비스간 통합이 원할하며 로그 및 메트릭 모두 AWS 내부 네트워크에서 동작하기 때문에 보안적인 측면에서도 훨씬 유리하다고 판단하여 변경

- ◆ AWS 내부 리소스의 API , 사용자 접근 , 리소스 생성 및 변경에 대한 감사가 필요하기 때문에 CloudTrail 을 이용한 감사 로그를 S3 로 저장하도록 구성
- ◆ 특정 임계치를 지정해서 확장되더라도 관리자가 바로 확인할 수 없었기 때문에 AWS CloudWatch 경보 알람과 AWS SNS 와 AWS ChatBot 을 이용해 실시간으로 장애 파악하기 위해서 사용
- ◆ 컨테이너 로그 수집 및 전달에 특화 된 FluentBit 를 통해서 로그를 CloudWatch 와 S3 에 저장해 로그 보존성을 더 높일 수 있기에 사용

- **인프라(서비스)**
  - 변경 내역
    - ◆ 기존 아키텍처에서는 CDN(Contents Delivery Network)으로서 CloudFront 를 구성하였으나, 새 아키텍처에서는 제거
    - ◆ 온 프레미스 네트워크 구성을 새로운 아키텍처에서는 제외하였습니다.
  - 변경 사유
    - ◆ CloudFront 가 Proxy 기능을 넘어서는 추가적인 이점을 제공하지 못할 것으로 분석되어 구성에서 제거
    - ◆ 온 프레미스 망 구성도 대신 클라우드로 마이그레이션 된 환경 중심의 아키텍처로 구성했기 때문에 제거
- **데이터베이스**
  - 변경 내역
    - ◆ 기존에 AWS RDS 를 사용하던 구성에서 컨테이너화 된 데이터베이스 솔루션으로의 구성을 변경
  - 변경 사유
    - ◆ 컨테이너화 된 데이터베이스는 기존 모니터링 시스템과의 통합을 용이하게 하여, 데이터베이스 서버의 상태를 보다 효과적으로 가시화하여 시스템의 안정성을 강화
    - ◆ AWS EKS 의 고급 스케줄링 및 자동 복구 기능을 활용하여, 데이터베이스의 가용성과 복원력을 높이도록 구성
    - ◆ 관리형 서비스 AWS RDS 대신 컨테이너화 된 데이터베이스를 선택함으로써, 데이터베이스 관리에 있어 더 큰 자율성과 유연성을 확보 가능
- 마이그레이션
  - 변경 내역
    - ◆ Bastion 을 통해서 온프레미스 데이터를 가져오는구성 변경
  - 변경 사유

◆ 현재 RDS 를 사용하는 것이 아닌 데이터베이스를 컨테이너화 시키는 방식의 마이그레이션 진행하기 때문에 해당 방법이 가장 적합하다고 판단