

2025년 전기 졸업과제 착수보고서

멀티 클라우드 인프라 기반 연합학습 환경 구축 플랫폼 개발



팀명 : 뭉게구름

202055595 전진혁

202155526 김민경

201924474 박재일

지도교수 : 염근혁 (인)

목차

| | |
|-------------------------|----|
| 1. 과제 목표 | 3 |
| 1) 과제 배경 | 3 |
| 2) 과제 세부 목표 | 3 |
| 2. 대상 문제 및 요구사항 분석 | 3 |
| 1) 유사 시스템 분석 | 3 |
| 2) 문제점 분석 | 4 |
| 3) 시스템의 필요성 | 5 |
| 4) 요구사항 분석 | 5 |
| 5) 유스케이스 분석 | 8 |
| 3. 현실적 제약 사항 분석 결과 및 대안 | 23 |
| 1) 현실적 제약 사항 | 23 |
| 2) 대안 | 23 |
| 4. 시스템 구성 | 23 |
| 1) 시스템 구성도 | 23 |
| 2) 개발 환경 | 25 |
| 3) 사용 기술 | 26 |
| 5. 개발 일정 및 역할 분담 | 39 |
| 1) 개발 일정 | 39 |
| 2) 역할 분담 | 40 |

1. 과제의 목표

1) 과제 배경

연합학습(Federated Learning)은 참여자(Client)와 집계자(Aggregator)로 구성된 분산학습 기술이다. 연합학습은 기존의 중앙집중형 기계학습(Machine Learning)과 달리 데이터를 중앙 서버로 모으지 않고 각 참여자의 로컬 환경에서 모델을 학습한 후 모델의 파라미터만 집계자에게 전송하고, 원본 데이터는 로컬 환경에 유지한다. 이러한 연합학습의 특성은 데이터 유출 위험을 감소시켜, 민감한 데이터를 다루는 의료, 제조, 스마트시티 등의 분야에서 활발하게 활용되고 있다.

멀티 클라우드(Multi-Cloud)는 이형의 클라우드 플랫폼(퍼블릭 클라우드, 프라이빗 클라우드)을 통합하여 단일 클라우드 플랫폼처럼 활용할 수 있는 기술이다. 멀티 클라우드와 연합학습을 결합한 플랫폼은 개별 클라우드에서 획득할 수 있는 장점(비용 효율성, 접근성, 보안성 등)을 적용하여 구축할 수 있다. 예를 들어, 의료 정보와 같이 강력한 보안성을 요구하는 데이터를 관리하는 도메인에서는 개인의 의료정보와 같이 외부 노출을 막아야 하는 경우 프라이빗 클라우드에 저장하여 보안을 강화하고, 질병 통계와 같이 다수의 데이터가 집합되었을 때 의미를 갖는 경우 퍼블릭 클라우드에 저장하여 접근성을 높이는 전략을 수립할 수 있다.

하지만 대부분의 기존 연합학습 플랫폼은 단일 클라우드 환경을 기반으로 연합학습에 필요한 환경을 구축하므로 비용, 리소스의 낭비가 발생할 수 있으며, 클라우드 플랫폼별 가용 리전의 물리적 위치에 따라 지연 시간(latency)이 증가하여 실시간 응답성이나 학습 속도에 영향을 미칠 수 있다. 또한 퍼블릭 클라우드 상에서 연합학습을 수행할 경우, 민감 데이터가 외부 인프라에 저장되어 보안 위협에 노출될 가능성이 높아진다.

이에 본 과제에서는 멀티 클라우드 환경에서 효율적인 연합학습 환경 구성을 위한 플랫폼을 제안한다. 본 과제에서 제안하는 플랫폼은 1) 클라우드 별 비용 및 지연시간을 고려한 멀티 클라우드 기반 연합학습 환경 구축, 2) 연합 학습 집계자 - 연합학습 참여자 계층 기반의 멀티 클라우드 지원 연합학습 방법 도출, 3) 연합학습 참여자 모니터링을 통한 동적 학습 작업 오케스트레이션 기술을 포함한다.

2) 과제 세부 목표

- ① 클라우드 별 비용 및 지연시간을 고려한 멀티 클라우드 기반 연합학습 환경 구축
- ② 연합학습 집계자 - 연합학습 참여자 계층 기반의 멀티 클라우드 지원 연합학습 방법 도출
- ③ 연합학습 참여자 모니터링을 통한 동적 학습 태스크 오케스트레이션 기술 구현

2. 대상 문제 및 요구사항 분석

1) 유사 시스템 분석

① IBM Federated Learning: IBM의 상용 연합학습 플랫폼으로, 여러 클라우드 환경에서 민감한 데이터를 공유하지 않고도 안전하게 AI 모델을 공동으로 학습할 수 있도록 지원한다.

i) Red Hat OpenShift와 IBM Cloud 기반 사설 클라우드 환경에 최적화된 Cross-Silo에 초점이

맞춰진 연합학습 플랫폼으로, 보안성과 안정적인 배포 환경을 제공한다.

ii) OpenShift를 기반으로 컨테이너화된 환경에서의 배포를 제공한다.

iii) 실시간 연합학습 모니터링 및 대시보드 제공 : 학습 과정의 상태를 실시간으로 확인할 수 있는 그래픽 대시보드를 제공하여 각 참여 클라우드 노드의 학습 진척도, 정확도 등의 지표를 직관적으로 모니터링할 수 있으며 각 라운드의 성능을 시각화 기능을 제공한다.

② AWS - SageMaker Federated Learning: SageMaker는 Edge Manager와 연동하여 사내 또는 엣지 디바이스에서 학습된 모델을 중앙 모델과 통합할 수 있는 연합학습 워크플로우를 구성할 수 있다.

③ Vertex AI(GCP): GCP의 Vertex AI는 사용자 정의 training job을 통해 연합학습을 간접적으로 구현할 수 있다.

④ FedML: 멀티 클라우드 및 엣지 디바이스를 연결하여 쉽고 효과적으로 연합학습 환경을 구축할 수 있게 해주는 플랫폼이다.

i) 클라우드 환경과 엣지 디바이스 간 복잡한 설정 없이, 간단한 명령어를 통해 다양한 지역의 연합학습 노드 연결을 지원한다. 데이터를 로컬에 두고 모델 업데이트만 주고받아 데이터 전송 비용과 지연을 줄여준다.

ii) 연합학습 참여자 각각의 학습 상태, 성능 지표(정확도, 손실률), 시스템 리소스를 상세히 모니터링할 수 있는 중앙 집중형 대시보드를 제공한다.

2) 문제점 분석

① 클라우드 벤더 종속성(Vendor Lock-in)

AWS, Azure, GCP 등 퍼블릭 클라우드 플랫폼은 각기 고유한 API, 서비스 구조 및 관리 도구를 제공한다. 이들은 자사 생태계 내에서의 통합과 확장은 용이하게 설계되어 있으나, 플랫폼 간 호환성이 제한적이라는 문제가 존재한다. 이러한 클라우드 벤더 종속성은 비용 최적화 기회를 놓치게 하고, 서로 다른 클라우드 플랫폼 간의 자원을 통합적으로 활용하기 어렵게 한다.

② 지연 시간 및 비용 최적화 부재

IBM Federated Learning, AWS Sage Maker, GCP Vertex AI와 같은 클라우드 기반 연합학습 지원 플랫폼은 일반적으로 클라우드 리전 간 비용 차이와 연합학습 참여자와의 네트워크 지연을 고려한 최적화 기능을 제공하지 않는다. 이로 인해 Aggregator와 연합학습 참여자가 물리적으로 멀리 떨어져 있는 경우, 네트워크 지연이 발생하여 모델의 학습 속도가 저하될 수 있다. 또한 클라우드 리전마다 비용이 상이하므로 비용 효율적인 연합학습 환경을 구성하는 데 제약이 발생한다.

③ 보안 취약성 및 데이터 프라이버시 문제

IBM Federated Learning, AWS Sage Maker 시스템들은 프라이빗-퍼블릭 클라우드 간 계층적 구조를 통한 데이터 보안 강화 메커니즘이 부재하다. 대부분의 플랫폼은 모든 연합학습 프로세스를 단일 클라우드 환경에서 처리하므로, 민감한 데이터가 퍼블릭 클라우드에 저장된다는 문제점이

존재한다. 이로 인해 원본 데이터가 공용 클라우드 환경에서 처리되고 기업이 데이터에 대한 완전한 통제권을 갖지 못하여 데이터 유출 및 보안 위협이 증가한다.

④ 동적 학습 오케스트레이션 부재

FedML은 클라우드 환경에서의 연합학습을 지원하는 프레임워크이다. 그러나 이러한 프레임워크는 클라우드 내에 구축된 가상머신의 리소스(e.g CPU, GPU, RAM) 상태를 고려하여 학습 작업을 배분하거나 재구성하는 동적 학습 오케스트레이션 기능이 제한적이다. 이에 따라 자원 사용의 비효율성, 학습 지연과 같은 문제가 발생할 수 있다.

3) 시스템의 필요성

① 클라우드 업체 종속성 해소

AWS, Azure, GCP 등 각기 고유한 API, 서비스 구조 및 관리 도구를 제공하는 퍼블릭 클라우드 플랫폼 간의 호환성 제한과 벤더 종속 문제를 해결하기 위해, 다양한 클라우드 서비스 제공업체(Cloud Service Provider, CSP)를 유연하게 활용할 수 있는 멀티 클라우드 플랫폼이 필요하다. 특정 벤더에 대한 기술적 종속성을 최소화함으로써, 다양한 클라우드 업체가 제공하는 기술 옵션을 선택할 수 있는 유연성을 확보할 수 있다.

② 지연 시간 및 비용 최적화

클라우드 리전 간 비용 차이와 네트워크 지연을 고려하지 않는 기존 플랫폼의 한계를 극복하기 위해, 비용과 지연 시간(latency)을 분석하여 최적의 집계자 배치를 결정하는 시스템이 필요하다. 이는 집계자와 연합학습 참여자 간의 물리적 거리로 인한 모델 학습 속도 저하 문제를 해결하고, 리전별로 상이한 비용을 고려하여 비용 효율적인 연합학습 환경을 구축할 수 있게 한다.

③ 계층적 멀티 클라우드 구조를 통한 보안 강화

연합학습 환경에서는 민감 데이터를 가진 연합학습 참여자들이 존재한다. 퍼블릭 클라우드 환경에서의 보안 취약성과 데이터 프라이버시 문제를 해결하기 위해, 원본 데이터는 프라이빗 클라우드에 격리하여 학습을 수행하도록 하고, 퍼블릭 클라우드는 높은 접근성이 필요한 모델 집계 작업을 수행하게 하는 계층화된 아키텍처가 필요하다.

④ 동적 학습 오케스트레이션 지원

가상머신의 리소스 가용성을 고려하지 않고 연합학습 작업을 배치하면, 전체적인 클라우드 리소스 효율성이 떨어질 수 있고 학습 지연이 발생할 수 있다. 따라서 프라이빗 클라우드 내 가상머신의 시스템 리소스를 실시간으로 모니터링하고, 과부하나 장애 상황을 감지하여 이를 관리하는 동적 학습 태스크 오케스트레이션 시스템이 필요하다.

4) 요구사항 분석

① 시스템 요구사항

i) 이기종 클라우드 연합학습 기능

1. 시스템은 이기종 클라우드(Public Cloud, Private Cloud) 플랫폼을 활용하여 연합학습을 수행할 수 있어야 한다.
2. 시스템은 다양한 클라우드 서비스 제공자(CSP)의 인증 정보를 관리할 수 있어야 한다.

ii) 연합학습 수행

1. 시스템은 연합학습을 위한 초기 모델 설정(모델 계층, 활성화 함수 등)을 지원해야 한다.
2. 시스템은 연합학습 완료 후 사용자에게 알림을 송신할 수 있어야 한다.
3. 시스템은 연합학습 완료 후 최종 모델을 저장해야 한다.

iii) 최적 모델 선정

1. 시스템은 모델의 성능 지표인 Accuracy, Recall, Precision, f1-score를 저장할 수 있어야 한다.
2. 시스템은 사용자가 정의한 성능 지표의 기준을 충족한 경우 평가 결과가 가장 좋은 모델을 선정할 수 있어야 한다.

iv) 연합학습 집계자 배치 최적화

1. 시스템은 클라우드 가상머신 운용 비용과 연합학습 참여자와의 지연시간을 종합적으로 고려하여 최적의 집계자 배치 위치를 추천해야 한다.

v) 클라우드 플랫폼 계층화를 통한 연합학습 구축

1. 시스템은 프라이빗 클라우드의 가상머신을 연합학습 참여자로 지정하여 민감한 데이터를 격리할 수 있어야 한다.
2. 시스템은 퍼블릭 클라우드의 가상머신을 연합학습 집계자로 지정하여 모델 집계 작업만을 수행하도록 해야 한다.

vi) 연합학습 참여자 가상머신 상태 기반 동적 학습 오케스트레이션

1. 시스템은 연합학습 참여자 내 가상머신의 자원(CPU, GPU, 메모리) 상태(사용량 및 가용량)를 실시간으로 모니터링 해야 한다.
2. 시스템은 연합학습 참여자 내 가상머신의 자원 상태에 따라 학습 작업을 동적으로 할당해야 한다.
3. 시스템은 장애가 발생한 가상머신의 학습 작업을 다른 가상머신으로 자동 이관해야 한다.

vii) 모델 정보 대시보드

1. 시스템은 연합학습 집계자의 모델 학습 진행 상황을 시각화하여 제공해야 한다.
2. 시스템은 연합학습 집계자의 모델 성능 지표를 실시간으로 표시해야 한다.

3. 시스템은 연합학습 집계자의 모델 구조 및 파라미터 정보를 표시할 수 있어야 한다.

② 사용자 요구사항

i) 연합학습 참여자 선택

1. 사용자는 연합학습에 참여 중인 모든 연합학습 참여자의 지리적 위치를 지도 형태로 확인할 수 있어야 한다.
2. 사용자는 웹 인터페이스를 통해 연합학습에 참여할 클라이언트를 선택할 수 있어야 한다.

ii) 연합학습 집계자 설정 및 배포

1. 사용자는 웹 인터페이스를 통해 연합학습 집계자의 컴퓨팅 사양(CPU, 메모리)을 선택할 수 있어야 한다.
2. 사용자는 웹 인터페이스를 통해 연합학습 집계자의 배포 지역을 선택할 수 있어야 한다.
3. 사용자는 웹 인터페이스를 통해 연합학습 집계자 네트워크 설정을 구성할 수 있어야 한다.

iii) 연합학습 집계자 모니터링

1. 사용자는 배포된 연합학습 집계자의 상태(실행 중, 중지됨, 오류 등)를 확인할 수 있어야 한다.
2. 사용자는 배포된 연합학습 집계자의 네트워크 지연시간을 실시간으로 확인할 수 있어야 한다.
3. 사용자는 배포된 연합학습 집계자의 운용 비용(시간당, 누적)을 확인할 수 있어야 한다.
4. 사용자는 배포된 연합학습 집계자의 지역 정보를 확인할 수 있어야 한다.
5. 사용자는 연합학습 집계자와 연결된 각 참여자 간 네트워크 연결 상태를 확인할 수 있어야 한다.

iv) 모델 선택 및 연합학습 실행

1. 사용자는 연합학습에 사용할 머신러닝 혹은 딥러닝 모델을 구성하거나 업로드할 수 있어야 한다.
2. 사용자는 모델의 하이퍼파라미터(학습률, 배치 크기, 에포크 등)를 설정할 수 있어야 한다.
3. 사용자는 모델의 배포의 기준이 되는 모델 성능 지표에 대한 기준값을 설정할 수 있어야 한다.
4. 사용자는 모델 구조를 웹 인터페이스에서 시각적으로 확인할 수 있어야 한다.
5. 사용자는 선택한 모델과 설정으로 연합학습 작업을 시작할 수 있어야 한다.

v) 연합학습 모니터링

1. 사용자는 연합학습의 현재 라운드 진행 상황을 실시간으로 확인할 수 있어야 한다.
2. 사용자는 각 라운드별 글로벌 모델의 정확도, 손실 등의 성능 지표를 그래프로 확인할 수 있어야 한다.
3. 사용자는 각 연합학습 참여자의 로컬 학습 상태(완료, 진행 중, 오류)를 확인할 수 있어야 한다.

4. 사용자는 연합학습 결과로 생성된 최종 모델의 성능 평가 지표를 확인할 수 있어야 한다.
5. 사용자는 연합학습 과정에서 발생한 오류 및 경고 메시지를 확인할 수 있어야 한다.

vi) 연합학습 참여자 가상머신 자원 모니터링

1. 사용자는 연합학습 참여자 내 가상머신의 자원(CPU, GPU, RAM 사용률)을 실시간으로 확인할 수 있어야 한다.

vii) 연합학습 결과 관리

1. 사용자는 완료된 연합학습 작업의 결과 모델을 연합학습 참여자에게 배포할 수 있어야 한다.
2. 사용자는 연합학습 수행 이력을 조회할 수 있어야 한다.

5) 유스케이스 분석

① 유스케이스 다이어그램

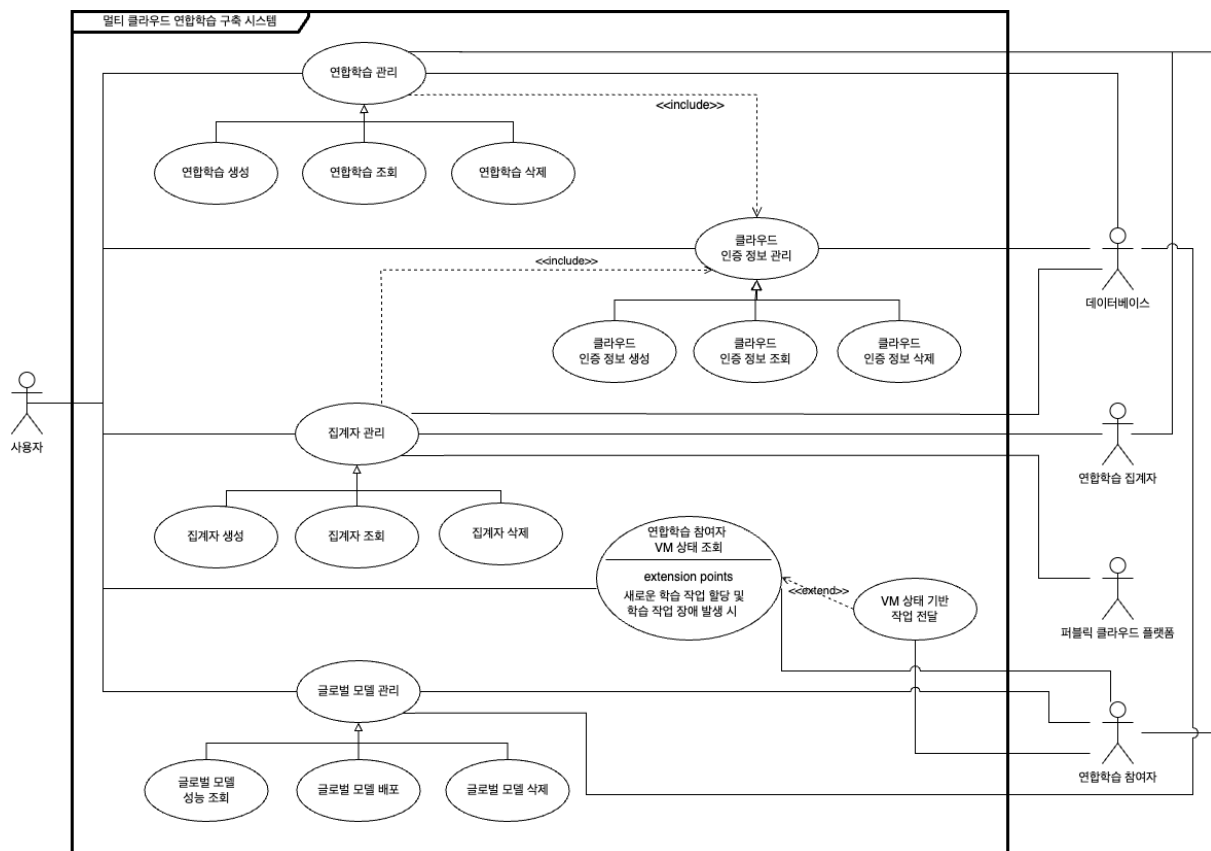


그림 1 멀티 클라우드 기반 연합학습 시스템 유스케이스 다이어그램

② Actor

표 1 Actor 목록

| 요건 | 설명 |
|--------------|---|
| 사용자 | 시스템의 기능에 접근하는 사용자 |
| 퍼블릭 클라우드 플랫폼 | 연합학습 집계자를 호스팅하는 퍼블릭 클라우드 인프라(AWS, GCP) |
| 연합학습 집계자 | 퍼블릭 클라우드 상에 배포되어 연합학습 참여자로부터 모델 파라미터를 제공받아 글로벌 모델을 생성하는 시스템 |
| 연합학습 참여자 | 프라이빗 클라우드로 구성되어 개별 학습 데이터를 사용해 로컬 학습을 수행하고, 학습된 모델 파라미터를 연합학습 집계자로 전송하는 시스템 |
| 데이터베이스 | 사용자 정보, 연합학습 이력, 모델 설정 정보, 학습 결과 등을 저장하는 시스템 |

③ 유스케이스 명세

i) 연합학습 생성

| 유스케이스명 | 연합학습 생성 |
|--------|---|
| 개요 | 특정 모델을 학습하기 위해 연합학습 작업을 생성하는 기능 |
| 관련 액터 | 사용자, 데이터베이스, 연합학습 참여자, 연합학습 집계자 |
| 선행 조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. • 클라우드 인증 정보가 등록된 상태여야 한다. • 연합학습 참여자가 등록되어 있는 상태여야 한다. |

| | |
|---------|--|
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 사용자는 연합학습 관리 메뉴를 선택한다. 2. 시스템은 기존 연합학습 목록과 연합학습 생성 버튼을 출력한다. 3. 사용자는 연합학습 생성 버튼을 선택한다. 4. 시스템은 연합학습 설정 값 입력 폼(라운드 수, 모델 업로드, 참여자 선택, 모델 평가 기준)을 출력한다. 5. 사용자는 설정 값을 입력하고 생성 버튼을 선택한다. 6. 시스템은 라운드 수가 0보다 큰지 검증한다. 7. 시스템은 모델 파일이 유효한 확장자를 가지는지 검증한다. 8. 시스템은 참여자 수가 0보다 큰지 검증한다. 9. 시스템은 모델 평가 기준이 Accuracy, Recall, Precision, f1-score 중 하나에 포함되는지 검증한다. 10. 시스템은 연합학습 집계자를 생성한다. 11. 시스템은 연합학습 집계자 및 연합학습 참여자의 VM에 필수 소프트웨어 설치 및 환경 변수 설정을 수행한다. 12. 시스템은 연합학습 집계자와 연합학습 참여자에게 학습 작업 정보를 전달한다. 13. 연합학습 참여자와 연합학습 집계자는 연합학습을 수행한다. 14. 연합학습 정보는 데이터베이스에 저장되어 모니터링 대시보드에 반영된다. |
| 대안 시나리오 | <p>A1: 연합학습 설정 값 유효성 검증 실패 시</p> <p>6.~9. 에서 설정 값 유효성 검증에 실패했을 때</p> <ol style="list-style-type: none"> 1. 시스템은 연합학습 생성 실패 메시지와 오류 내용을 출력한다. 2. 기본 시나리오 2.로 돌아간다. <p>A2: 연합학습 집계자 생성 실패 시</p> <p>5. 에서 연합학습 집계자 생성에 실패했을 때</p> <ol style="list-style-type: none"> 1. 시스템은 집계자 생성에 실패하였다는 메시지를 출력한다. 2. 기본 시나리오 2.로 돌아간다. |
| 후행 조건 | <ul style="list-style-type: none"> • 시스템에 연합학습 프로젝트가 생성된다. • 연합학습 참여자와 연합학습 집계자가 연계되어 연합학습을 수행한다. • 생성한 연합학습을 연합학습 모니터링 대시보드에서 확인할 수 있다. |

ii) 연합학습 조회

| | |
|---------|--|
| 유스케이스명 | 연합학습 조회 |
| 개요 | 실행 중이거나 실행되었던 연합학습의 상태 및 결과를 조회하는 기능 |
| 관련 액터 | 사용자, 데이터베이스 |
| 선행 조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. • 실행 중이거나 실행되었던 연합학습이 존재해야 한다. |
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 사용자는 연합학습 관리 메뉴를 선택한다. 2. 시스템은 기존 연합학습 목록과 연합학습 생성 버튼을 출력한다. 3. 사용자는 조회할 연합학습을 선택한다. 4. 시스템은 데이터베이스에서 선택한 연합학습에 대한 정보(모델 정보, 평가 결과, 걸린 시간, 참여자 정보)를 불러온다. 5. 시스템은 사용자가 선택한 연합학습의 정보(모델 정보, 평가 결과, 걸린 시간, 참여자 정보)를 출력한다. |
| 후행 조건 | <ul style="list-style-type: none"> • 없음 |

iii) 연합학습 삭제

| 유스케이스명 | 연합학습 삭제 |
|---------|--|
| 개요 | 실행되었던 연합학습의 기록을 삭제하는 기능 |
| 관련 액터 | 사용자, 데이터베이스 |
| 선행 조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. • 실행되었던 연합학습이 존재해야 한다. |
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 사용자는 연합학습 관리 메뉴를 선택한다. 2. 시스템은 기존 연합학습 목록과 연합학습 생성 버튼을 출력한다. 3. 사용자는 삭제할 연합학습 이력을 선택한다. 4. 시스템은 사용자가 선택한 연합학습의 정보 및 삭제 버튼을 출력한다. 5. 사용자는 삭제 버튼을 선택한다. 6. 시스템은 삭제 재확인 메시지 및 버튼을 출력한다. 7. 사용자는 삭제 재확인 버튼을 선택한다. 8. 시스템은 데이터베이스에서 해당 연합학습 정보를 삭제한다. 9. 시스템은 연합학습 이력이 성공적으로 삭제되었다는 메시지를 출력한다. |
| 대안 시나리오 | <p>A1: 사용자가 삭제 취소 시</p> <p>6. 에서 사용자가 연합학습 삭제를 취소한 경우</p> <ol style="list-style-type: none"> 1. 기본 시나리오 4. 로 돌아간다. <p>A2:연합학습 이력 삭제 실패 시</p> <p>8. 에서 연합학습 이력 삭제에 실패했을 때</p> <ol style="list-style-type: none"> 1. 시스템은 연합학습 삭제에 실패하였다는 메시지를 출력한다. 2. 기본 시나리오 2. 로 돌아간다. |
| 후행 조건 | <ul style="list-style-type: none"> • 데이터베이스에서 연합학습 이력이 삭제된다. |

iv) 클라우드 인증 정보 등록

| | |
|---------|--|
| 유스케이스명 | 클라우드 인증 정보 등록 |
| 개요 | 다양한 클라우드 서비스 제공자(AWS, GCP)의 인증 정보를 등록하는 기능 |
| 관련 액터 | 사용자, 데이터베이스 |
| 선행 조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. |
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 사용자는 클라우드 인증 정보 관리 메뉴를 선택한다. 2. 시스템은 등록된 인증 정보 목록과 새 인증 정보 등록 버튼을 출력한다. 3. 사용자는 새 인증 정보 등록 버튼을 클릭한다. 4. 시스템은 등록 가능한 클라우드 종류 목록을 출력한다. 5. 사용자는 등록할 인증정보의 클라우드 플랫폼을 선택한다. 6. 사용자는 선택한 클라우드 인증 정보 등록에 필요한 정보를 입력하고 저장 버튼을 클릭한다. 7. 시스템은 입력된 인증 정보를 기반으로 API 호출을 통해 유효성 검증을 수행한다. 8. 시스템은 인증 정보를 암호화하여 데이터베이스에 저장한다. 9. 시스템은 사용자에게 등록 성공 메시지를 출력한다. |
| 대안 시나리오 | <p>A1: 입력된 인증 정보 유효성 검증 실패 시</p> <p>7. 에서 입력된 인증 정보 유효성 검증에 실패했을 때</p> <ol style="list-style-type: none"> 1. 시스템은 잘못된 인증 정보라는 화면을 출력한다. 2. 기본 시나리오 2. 로 돌아간다. |
| 후행 조건 | <ul style="list-style-type: none"> • 데이터베이스에서 클라우드 인증 정보가 삭제된다. |

v) 클라우드 인증 정보 조회

| | |
|---------|---|
| 유스케이스명 | 클라우드 인증 정보 조회 |
| 개요 | 다양한 클라우드 서비스 제공자(AWS, Azure, GCP 등)의 인증 정보를 조회하는 기능 |
| 관련 액터 | 사용자, 데이터베이스 |
| 선행 조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. • 데이터베이스에 등록된 인증 정보가 존재해야 한다. |
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 사용자는 클라우드 인증 정보 관리 메뉴를 선택한다. 2. 시스템은 데이터베이스에 등록된 클라우드 인증 정보 목록을 출력한다. 3. 시스템은 데이터베이스에 저장된 클라우드 인증 정보를 조회한다. 4. 사용자는 특정 클라우드 인증 정보 목록을 선택한다. 5. 시스템은 해당 클라우드 인증 정보를 데이터베이스에서 조회한다. 6. 시스템은 사용자가 선택한 클라우드 인증 정보에 대해 클라우드 종류, 인증 정보 이름, 등록 시기를 출력한다. |
| 후행 조건 | <ul style="list-style-type: none"> • 없음 |

vi) 클라우드 인증 정보 삭제

| | |
|---------|---|
| 유스케이스명 | 클라우드 인증 정보 삭제 |
| 개요 | 다양한 클라우드 서비스 제공자(AWS, Azure, GCP 등)의 인증 정보를 삭제하는 기능 |
| 관련 액터 | 사용자, 데이터베이스 |
| 선행 조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. • 데이터베이스에 등록된 클라우드 인증 정보가 존재해야 한다. |
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 사용자는 클라우드 인증 정보 관리 메뉴를 선택한다. 2. 시스템은 데이터베이스에 등록된 클라우드 인증 정보 목록을 출력한다. 3. 사용자는 특정 클라우드 인증 정보를 클릭한다. 4. 시스템은 특정 클라우드에 인증 정보 내용과 삭제 버튼을 출력한다. 5. 사용자는 삭제하고싶은 클라우드 인증 정보의 삭제 버튼을 클릭한다. 6. 시스템은 삭제 재확인 메시지와 버튼을 출력한다. 7. 사용자는 삭제 재확인 버튼을 클릭한다. 8. 시스템은 해당 인증 정보를 데이터베이스에서 삭제한다. 9. 시스템은 성공적으로 삭제되었다는 메시지를 출력한다. |
| 대안 시나리오 | <p>A1: 사용자가 삭제 취소 시</p> <p>6. 에서 사용자가 클라우드 인증 정보 삭제를 취소한 경우</p> <ol style="list-style-type: none"> 1. 기본 시나리오 4. 로 돌아간다. <p>A1: 클라우드 인증 정보 삭제 실패 시</p> <p>8. 에서 클라우드 인증 정보 삭제에 실패했을 때</p> <ol style="list-style-type: none"> 1. 시스템은 삭제에 실패하였다는 메시지와 오류 메시지를 출력한다. 2. 기본 시나리오 2. 로 돌아간다. |
| 후행 조건 | <ul style="list-style-type: none"> • 클라우드 인증 정보가 데이터베이스에서 삭제된다. |

vii) 집계자 생성

| | |
|---------|---|
| 유스케이스명 | 집계자 생성 |
| 개요 | 새로운 연합학습 집계자를 생성하여 연합학습에 배치하는 기능 |
| 관련 액터 | 사용자, 데이터베이스, 퍼블릭 클라우드 플랫폼, 연합학습 집계자 |
| 선행 조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. • 시스템에 클라우드 인증 정보가 등록되어있어야 한다. • 사용자가 연합학습 생성을 위해 참여자를 선택한 상태여야 한다. |
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 시스템은 연합학습 집계자 생성 버튼을 출력한다. 2. 시스템은 지연시간 및 비용을 최적화하고 집계자를 배포할 리전을 결정한다. 3. 시스템은 최적화한 지연시간 및 비용, 결정된 리전을 출력한다. 4. 사용자는 배포할 리전을 확인하고 연합학습 집계자 생성 버튼을 선택한다. 5. 시스템은 퍼블릭 클라우드 플랫폼에 집계자 생성 요청을 보낸다. 6. 퍼블릭 클라우드 플랫폼은 결정된 리전에 연합학습 집계자를 생성한다. 7. 시스템은 연합학습 집계자 정보를 데이터베이스에 저장한다. 8. 시스템은 연합학습 집계자 생성 완료 메시지를 출력한다. |
| 대안 시나리오 | <p>A1: 연합학습 집계자 배포 리전 변경 시</p> <p>4. 에서 사용자가 연합학습 집계자 배포 리전을 변경했을 때</p> <ol style="list-style-type: none"> 1. 시스템은 변경된 리전에 따른 지연시간과 비용을 출력한다. 2. 사용자는 연합학습 집계자 생성 버튼을 선택한다. 3. 기본 시나리오 5. 로 돌아간다. <p>A2: 퍼블릭 클라우드 플랫폼에서 연합학습 집계자 생성 실패 시</p> <p>6. 에서 연합학습 집계자 생성에 실패했을 때</p> <ol style="list-style-type: none"> 1. 시스템은 집계자 생성에 실패하였다는 메시지와 오류 메시지를 출력한다. 2. 기본 시나리오 1. 로 돌아간다. |
| 후행 조건 | <ul style="list-style-type: none"> • 비용 및 지연시간이 최적화된 집계자가 생성된다. • 생성된 연합학습 집계자가 연합학습에 배치된다. |

viii) 집계자 조회

| | |
|---------|---|
| 유스케이스명 | 집계자 조회 |
| 개요 | 연합학습에 배치된 집계자를 조회하는 기능 |
| 관련 액터 | 사용자, 데이터베이스, 퍼블릭 클라우드 플랫폼, 연합학습 집계자 |
| 선행조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. • 시스템에 클라우드 인증 정보가 등록되어있어야 한다. • 시스템은 집계자를 생성한 상태여야 한다. |
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 사용자는 집계자 관리 메뉴를 선택한다. 2. 시스템은 등록된 집계자 목록과 집계자 삭제 버튼을 출력한다. 3. 사용자는 조회할 집계자를 선택한다. 4. 시스템은 선택한 집계자의 정보(관련 연합학습, 하드웨어 정보, 연결된 클라이언트)를 데이터베이스에서 조회한다. 5. 데이터베이스는 집계자의 정보를 시스템에 전달한다. 6. 시스템은 퍼블릭 클라우드 플랫폼에 집계자의 정보(고유 식별자, 네트워크 구성, 비용 정보 등)를 요청한다. 7. 퍼블릭 클라우드 플랫폼은 집계자의 정보를 시스템에 전달한다. 8. 시스템은 집계자의 자원 사용률 및 정보를 출력한다. |
| 후행 조건 | <ul style="list-style-type: none"> • 없음 |

ix) 집계자 삭제

| | |
|---------|---|
| 유스케이스명 | 집계자 삭제 |
| 개요 | 연합학습에 배치된 집계자를 삭제하는 기능 |
| 관련 액터 | 사용자, 데이터베이스, 퍼블릭 클라우드 플랫폼, 연합학습 집계자 |
| 선행 조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. • 시스템에 클라우드 인증 정보가 등록되어있어야 한다. • 해당 집계자를 사용한 연합학습이 종료된 상태여야한다. |
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 사용자는 집계자 관리 메뉴를 선택한다. 2. 시스템은 등록된 집계자 목록과 집계자 삭제 버튼을 출력한다. 3. 사용자는 삭제하고자 하는 집계자의 삭제 버튼을 선택한다. 4. 시스템은 사용자에게 삭제 확인 메시지를 출력한다. 5. 사용자가 삭제를 확정한다. 6. 시스템은 퍼블릭 클라우드 플랫폼에 해당 집계자 삭제 요청을 보낸다. 7. 퍼블릭 클라우드 플랫폼은 집계자를 삭제한다. 8. 퍼블릭 클라우드 플랫폼은 집계자에 활용된 네트워크 및 키페어 구성 요소를 삭제한다. 9. 시스템은 데이터베이스에서 해당 집계자 정보를 삭제한다. 10. 시스템은 집계자가 성공적으로 삭제되었다는 메시지를 출력한다. |
| 대안 시나리오 | <p>A1: 사용자가 집계자 삭제 요청 취소 시</p> <p>4. 에서 사용자가 집계자 삭제 요청을 취소했을 시</p> <ol style="list-style-type: none"> 1. 기본 시나리오 2. 로 돌아간다. <p>A2: 집계자 삭제 실패 시</p> <p>6. 에서 집계자 삭제에 실패했을 시</p> <ol style="list-style-type: none"> 1. 시스템은 삭제에 실패하였다는 오류 메시지를 출력한다. 2. 기본 시나리오 2. 로 돌아간다. |
| 후행 조건 | <ul style="list-style-type: none"> • 집계자가 데이터베이스에서 삭제된다. |

x) VM 상태 기반 작업 전달

| | |
|---------|--|
| 유스케이스명 | VM 상태 기반 작업 전달 |
| 개요 | 연합학습 참여자 VM의 상태를 기반으로 적절한 VM에 학습 작업 할당 및 이관하는 기능 |
| 관련 액터 | 연합학습 참여자 |
| 선행 조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. • 연합학습이 생성된 상태여야 한다. • 연합학습 집계자와 연합학습 참여자의 VM이 생성되어 있어야 한다. • 각 연합학습 참여자의 VM 상태 정보가 모니터링되고 있어야 한다. |
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 시스템은 연합학습 참여자에게 연합학습 작업을 할당한다. 2. 시스템은 각 연합학습 참여자 VM의 리소스 상태(CPU, GPU, 메모리 사용량)를 확인한다. 3. 시스템은 VM의 리소스 상태에 따라 작업량을 조절한다. 4. 시스템은 참여자 VM의 현재 자원 사용량이 기준치 이하인 VM을 식별한다. 5. 시스템은 해당 VM에 연합학습 작업을 전달한다. 6. 시스템은 각 VM에 전달된 작업의 상태를 모니터링한다. 7. 작업이 완료되면 시스템은 결과를 수집하고 다음 작업 할당을 준비한다. |
| 대안 시나리오 | <p>A1: VM 상태가 기준치 이상일 경우</p> <p>3. 에서 모든 VM의 자원 사용량이 기준치 이상일 경우</p> <ol style="list-style-type: none"> 1. 시스템은 작업 대기열에 해당 작업을 추가한다. 2. 시스템은 VM 상태가 개선될 때까지 주기적으로 상태를 확인한다. 3. VM 상태가 개선되면 대기 중인 작업을 할당한다. <p>A2: VM에 장애가 발생한 경우</p> <p>5. 에서 VM에 장애가 발생했을 경우</p> <ol style="list-style-type: none"> 1. 시스템은 해당 VM에 할당된 작업을 중단한다. 2. 시스템은 다른 VM에 해당 작업을 재할당한다. 3. 시스템은 VM 장애에 대한 알림을 전송한다. 4. 기본 시나리오 1. 로 돌아간다. |
| 후행 조건 | <ul style="list-style-type: none"> • 각 VM의 자원 상태에 최적화된 작업 할당이 이루어진다. • 장애 발생 시 작업이 다른 VM으로 자동 이관된다. |

xi) 글로벌 모델 성능 조회

| | |
|---------|--|
| 유스케이스명 | 글로벌 모델 성능 조회 |
| 개요 | 연합학습으로 생성된 글로벌 모델의 성능 지표를 조회하는 기능 |
| 관련 액터 | 사용자, 데이터베이스 |
| 선행 조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. • 학습이 완료된 글로벌 모델이 최소 1개 이상 존재하여야 한다. |
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 사용자는 글로벌 모델 관리 메뉴를 선택한다. 2. 시스템은 글로벌 모델 목록을 출력한다. 3. 사용자는 성능을 조회하고자 하는 글로벌 모델을 선택한다. 4. 시스템은 데이터베이스에서 선택한 글로벌 모델에 대한 정보를 조회한다.. 5. 시스템은 선택된 모델의 성능 지표(Accuracy, Recall, Precision, F1-score)를 꺾은선 그래프 형태로 출력한다. 6. 시스템은 해당 모델 학습을 수행한 연합학습 정보와 참여자 정보자 정보를 출력한다. |
| 후행 조건 | <ul style="list-style-type: none"> • 없음 |

xii) 글로벌 모델 배포

| | |
|---------|--|
| 유스케이스명 | 글로벌 모델 배포 |
| 개요 | 연합학습으로 생성된 글로벌 모델을 연합학습 참여자에 배포하는 기능 |
| 관련 액터 | 사용자, 데이터베이스, 연합학습 참여자 |
| 선행 조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. • 연합학습이 완료되어 배포 가능한 글로벌 모델이 생성되어 있어야 한다. • 클라우드 인증 정보가 등록되어 있어야 한다. |
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 사용자는 글로벌 모델 관리 메뉴를 선택한다. 2. 시스템은 글로벌 모델 목록을 출력한다. 3. 사용자는 배포하고자 하는 글로벌 모델을 선택한다. 4. 시스템은 선택된 모델의 정보와 함께 배포 버튼을 출력한다. 5. 사용자는 배포 버튼을 선택한다. 6. 시스템은 선택된 글로벌 모델을 해당 모델의 연합학습에 참여한 참여자에게 배포한다. 7. 시스템은 배포 상태와 진행 상황을 출력한다. 8. 배포가 완료되면 시스템은 배포 성공 메시지를 출력한다. |
| 대안 시나리오 | <p>A1: 글로벌 모델 배포 실패 시</p> <p>8. 에서 글로벌 모델 배포에 실패했을 때</p> <ol style="list-style-type: none"> 1. 시스템은 배포 실패 메시지와 에러 메시지를 출력한다. 2. 시스템은 글로벌 모델을 배포받지 못한 연합학습 참여자 목록을 출력한다. 3. 기본 시나리오 4. 로 돌아간다. |
| 후행 조건 | <ul style="list-style-type: none"> • 글로벌 모델이 연합학습 참여자에게 배포된다. • 배포 정보가 데이터베이스에 저장된다. |

xiii) 글로벌 모델 삭제

| | |
|---------|---|
| 유스케이스명 | 글로벌 모델 삭제 |
| 개요 | 학습이 완료된 글로벌 모델을 데이터베이스에서 삭제하는 기능 |
| 관련 액터 | 사용자, 데이터베이스 |
| 선행 조건 | <ul style="list-style-type: none"> • 사용자는 시스템에 로그인된 상태여야 한다. |
| 기본 시나리오 | <ol style="list-style-type: none"> 1. 사용자는 글로벌 모델 관리 메뉴를 선택한다. 2. 시스템은 글로벌 모델 목록을 출력한다. 3. 사용자는 삭제하고자 하는 글로벌 모델을 선택한다. 4. 시스템은 선택된 모델의 정보와 함께 삭제 버튼을 출력한다. 5. 사용자는 삭제 버튼을 선택한다. 6. 시스템은 삭제 재확인 메시지 및 버튼을 출력한다. 7. 사용자는 삭제 재확인 버튼을 선택한다. 8. 시스템은 데이터베이스에서 해당 모델을 삭제한다. 9. 시스템은 삭제 완료 메시지를 출력한다. |
| 대안 시나리오 | <p>A1: 사용자가 삭제 취소 시 7. 에서 사용자가 글로벌 모델 삭제를 취소한 경우</p> <ol style="list-style-type: none"> 1. 기본 시나리오 4. 로 돌아간다. <p>A2: 모델 삭제 실패 시 8. 에서 모델 삭제에 실패한 경우</p> <ol style="list-style-type: none"> 1. 시스템은 삭제 실패 메시지와 함께 실패 원인을 출력한다. 2. 기본 시나리오 4. 로 돌아간다. |
| 후행 조건 | <ul style="list-style-type: none"> • 데이터베이스에서 해당 글로벌 모델에 대한 정보가 삭제된다. |

3. 현실적 제약 사항 분석 결과 및 대안

1) 현실적 제약 사항

- ① 지리적으로 분산된 참여자를 구성하는 것은 매우 어렵다.
- ② 다양한 기관으로부터 실제 데이터를 가져와 연합학습을 수행하는 것이 이상적이지만, 실제 기관의 데이터를 바탕으로 연합학습을 수행하는 것은 어렵다.
- ③ 다수의 클라우드 플랫폼을 동시에 대규모로 활용하는 데에 어려움이 있다.

2) 대안

- ① 각 클라우드의 리전이 해당하는 연합학습 참여자의 물리적인 지역과 일치한다고 가정하고 실험을 진행한다.
- ② 공개된 데이터셋을 연합학습 참여자 간에 분산시켜 각 기관에서 개별적으로 수집된 데이터라고 가정하여 시뮬레이션한다.
- ③ 퍼블릭 클라우드는 범용적으로 사용되는 AWS와 GCP 두 개의 플랫폼만을 사용하여 검증한다.

4. 시스템 구성

1) 시스템 구성도

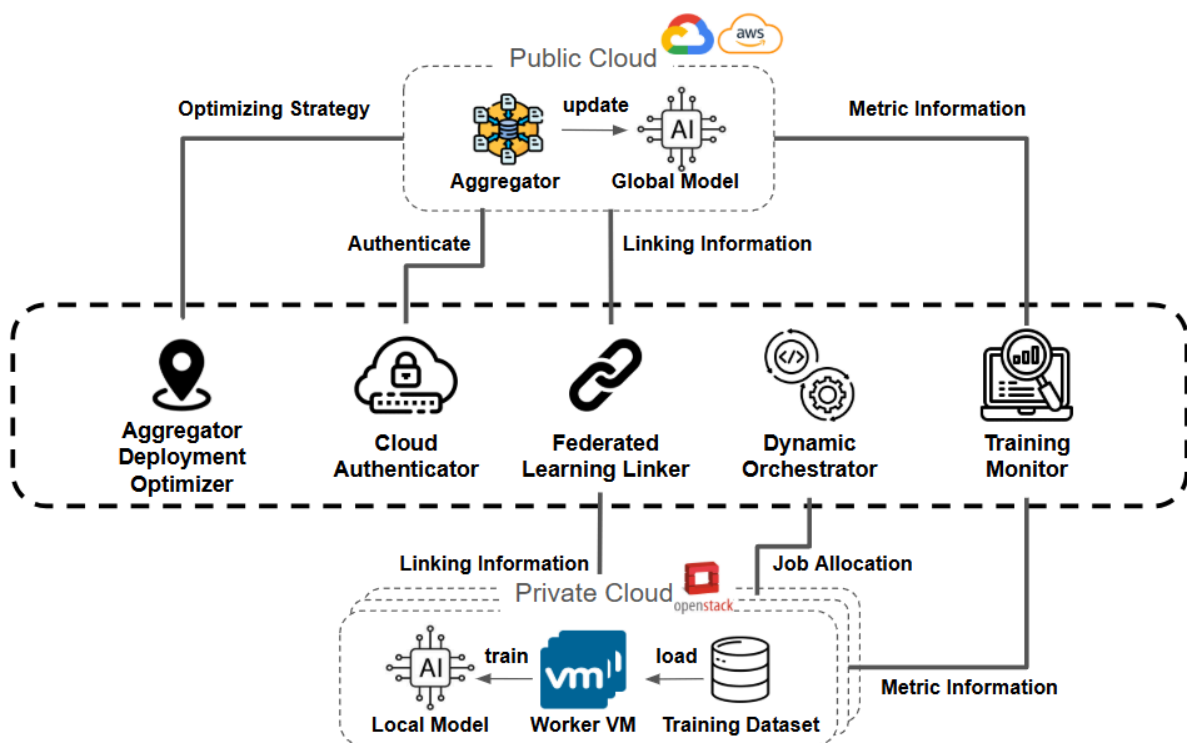


그림 2. 시스템 구성도

본 과제에서는 퍼블릭 클라우드와 프라이빗 클라우드의 기능이 명확히 분리된 환경에서 연합학습을 수행하는 멀티 클라우드 기반 시스템을 구축한다.

[그림 2]는 시스템의 구성도를 나타낸 것이다. 퍼블릭 클라우드에는 연합학습 집계자와 글로벌 모델이 위치하며, 프라이빗 클라우드(OpenStack 기반)에는 로컬 모델, 연합학습 참여자 VM, 학습 데이터셋이 배치되어 로컬 환경에서 모델 학습을 수행한다. 학습을 수행하며, 프라이빗 클라우드는 로컬 모델의 파라미터를 퍼블릭 클라우드의 집계자에게 전송하고, 집계자는 이를 집계하여 글로벌 모델을 업데이트한다.

시스템은 연합학습 환경을 구축하고 연합학습을 지원하기 위해 다음과 같은 핵심 모듈들을 포함한다.

- **Aggregator Deployment Optimizer**: 최적의 연합학습 집계자 위치를 선정
- **Cloud Authenticator**: 퍼블릭 클라우드 인증 처리
- **Federated Learning Linker**: 연합학습 집계자와 연합학습 참여자 연결 수행
- **Dynamic Orchestrator**: 연합학습 작업 프라이빗 클라우드의 VM에 동적 할당
- **Training Monitor**: 연합학습 작업 및 연합학습 참여자 자원 상태 모니터링

사용자는 위 시스템에 연합학습 수행에 필요한 정보(참여 노드, 학습 조건 등)를 입력하고, 시스템은 이를 바탕으로 퍼블릭 클라우드 및 프라이빗 클라우드와 상호작용 하며 연합학습을 수행하는 환경을 구축한다.

2) 개발 환경

















| | |
|------------------------------------|--|
| 소프트웨어 형상관리 |  |
| 컨테이너 기술 |   |
| 클라우드 플랫폼 |    |
| 모니터링 |    |
| VM Provisioning 자동화(IaC) |  |
| MLOps |  |
| Federated Learning framework |  |
| DBMS |  |
| Backend |   |
| Frontend |  |

표 2 개발 환경

| 개발환경 | 사용기술 |
|------------------------------|-----------------------------|
| 소프트웨어 형상관리(SCM) | GitHub |
| 컨테이너 기술 | Docker, Kubernetes |
| 클라우드 플랫폼 | AWS, GCP, OpenStack |
| 모니터링 | Prometheus, Grafana, Jaeger |
| VM Provisioning 자동화(IaC) | Terraform |
| MLOps | Kubeflow |
| Federated Learning framework | Flower |
| DBMS | PostgreSQL |
| Backend | gin-gonic |
| Frontend | Next.js |
| 프로젝트 문서화 관리 | Notion, Google Drive |
| 개발 IDE | VSCode |

3) 사용 기술

① Docker: 도커(Docker)는 리눅스 컨테이너에 리눅스 어플리케이션을 프로세스 격리기술을 사용하여 더 쉽게 컨테이너로 실행하고 관리할 수 있게 해주는 오픈소스 프로젝트이다. 도커는 일반적으로 도커 엔진(Docker Engine) 혹은 도커에 관련된 모든 프로젝트를 말한다.

아래 [그림 3]은 도커의 아키텍처를 나타내며, 아키텍처를 구성하는 각 컴포넌트의 기능은 다음과 같다.

- i) Docker daemon: 도커 API 요청을 수신하고 이미지, 컨테이너, 네트워크 및 볼륨과 같은 도커 개체를 관리한다.
- ii) Docker Client: 사용자가 도커와 상호 작용하기 위한 미들웨어로 이미지 생성, 컨테이너 생성 및 실행과 같은 명령어들을 위생 할 수 있으며, 해당 명령어들은 도커 데몬과의 통신을 통하여 수행된다.
- iii) Docker Desktop: 도커 데스크탑 지원용 프로그램이다. 도커 데몬, 도커 클라이언트와 쿠버네티스(minikube) 와 같은 다양한 응용프로그램들이 함께 탑재되어있다.
- iv) Docker Registry: 도커 이미지를 저장하는 레지스트리로 도커 데스크탑 설치 시 개인 레지스트리가 함께 설치되며, 공용 레지스트리인 Docker hub에서 이미지 검색을 통하여 이미지를 다운 받을 수 있다.
- v) Docker object: 도커를 사용하는데 있어서 필요한 이미지, 컨테이너, 네트워크, 볼륨, 플러그인들이 여기에 속한다.

- vi) Docker image: 도커 컨테이너를 만드는데 사용되어지는 템플릿이다. 다른 이미지에 사용자 설정을 추가하여 사용할 수 있는 구성 세부 정보 설치 기능을 수행할 수 있다.
- vii) Container: 이미지의 실행 가능한 인스턴스로 Docker API 혹은 CLI를 통하여 생성, 중지, 시작, 이동, 삭제가 가능하다.

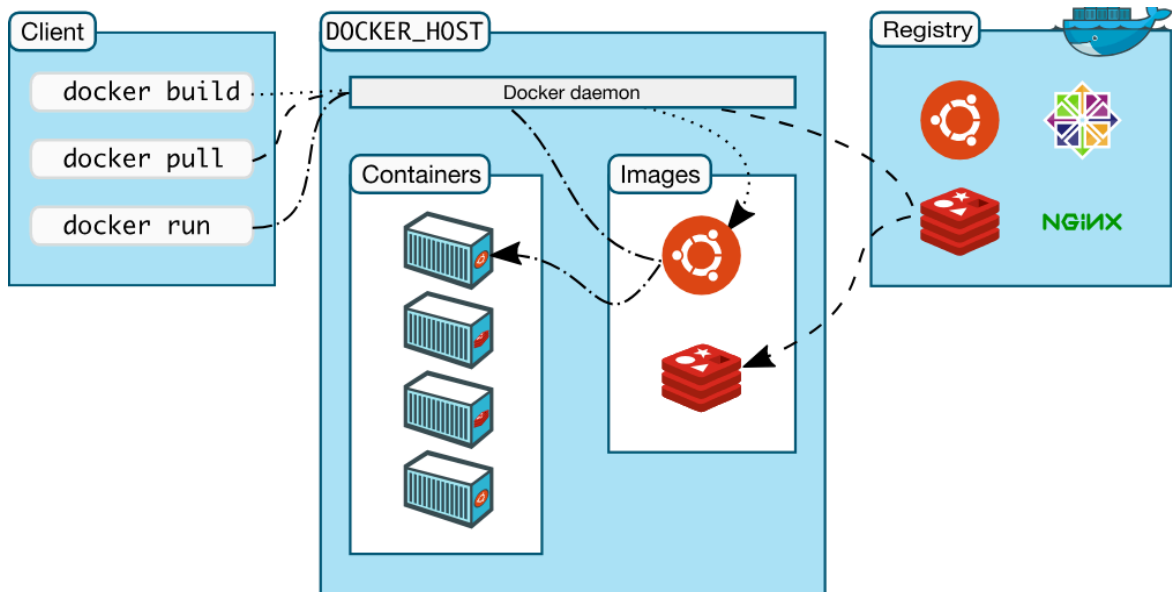


그림 3. Docker Architecture

② Kubernetes: Kubernetes(쿠버네티스)는 컨테이너화 된 워크로드와 서비스를 관리하기 위한 이식성이 있고, 확장 가능한 오픈소스 플랫폼이다. 쿠버네티스는 선언적 구성과 자동화를 모두 용이하게 해준다. 쿠버네티스는 크고, 빠르게 성장하는 생태계를 가지고 있다. 쿠버네티스 서비스, 기술 지원 및 도구는 어디서나 쉽게 이용할 수 있다.

쿠버네티스는 Control Plane과 Worker Nodes로 구성되며 아래는 각각의 요소에 대한 설명이다.

i) Control Plane(쿠버네티스 클러스터 전체를 컨트롤하는 시스템)

1. kube-apiserver: 쿠버네티스 내부의 모든 컴포넌트들이 서로 호출하기 위한 컴포넌트로 쿠버네티스의 모든 기능은 REST API로 제공하고 그에 대한 명령을 처리하는 컴포넌트이다.
2. kube-controller-manager: 노드를 사용할 수 없게 될 때 조치를 취하고, 컨테이너 Pod 수가 예상대로인지 확인하나는 등 다양한 컴포넌트의 상태를 지속적으로 모니터링하는 동시에 실행 상태를 유지하는 역할을 한다.
3. kube-scheduler: Pod(워크로드 객체)를 어떤 노드에 할당해야할지 결정한다. 스케줄링 하는 동안 현재 클러스터 상태와 Pod의 요구 사항을 기반으로 결정이 내려진다(API Server에 요청을 보내 정보를 얻는다).
4. etcd(key-value data store): 쿠버네티스 클러스터의 상태를 저장하기 위한 분산 Key-Value 저장소이다. 오직 API Server만 etcd data store와 통신할 수 있다.

ii) Worker Node(클라이언트 애플리케이션 구동하기 위한 환경)

1. Container Runtime : 컨테이너의 라이프사이클 관리. 클러스터의 각 노드(Control plane, Worker Node)에 필요하다. CRI-O, docker, containerd 등을 지원한다.
2. Node Agent - kubelet : 각 노드에서 동작하는 agent이며, control plane과 통신한다.
3. Proxy - kube-proxy: 각 노드에서 동작하는 network agent이며, networking rules에 대한 동적 업데이트와 유지관리를 수행한다. 또한 Pod 네트워킹 세부 정보를 추상화하여, 클라이언트 요청을 Pod 내부 컨테이너로 전달한다.

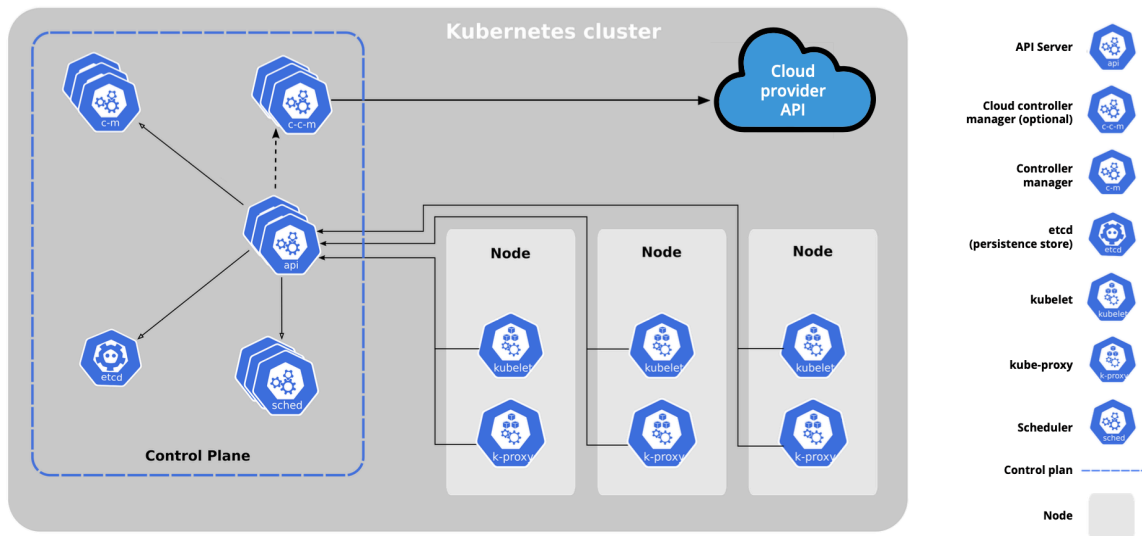


그림 4. Kubernetes Architecture

③ AWS: AWS는 기업과 조직이 클라우드 기반 인프라와 서비스를 구축, 배포 및 관리할 수 있도록 설계된 포괄적인 클라우드 컴퓨팅 플랫폼이다. 이는 높은 가용성, 확장성, 그리고 비용 효율성을 제공하여 컴퓨팅, 스토리지, 데이터베이스, 분석, 네트워킹, 모바일, 개발자 도구, 관리 도구, 보안 및 엔터프라이즈 애플리케이션과 같은 광범위한 클라우드 서비스를 관리한다. AWS는 다수의 모듈식 서비스로 구성되어 있으며, 각 서비스는 클라우드 시스템 내에서 특정 기능을 담당하고 있다. 이러한 서비스들은 서로 긴밀하게 통합되어 있어 독립적으로 또는 조합하여 사용할 수 있다. 사용자는 필요에 따라 AWS 서비스를 유연하게 선택하고 프로비저닝할 수 있다.

i) EC2: EC2는 가상 서버 인스턴스를 제공하는 서비스이다. 사용자는 이를 통해 컴퓨팅 리소스를 필요에 따라 유연하게 할당하고 관리할 수 있다. EC2는 다양한 하드웨어 설정, 운영 시스템, 네트워크 설정 등을 사용자가 선택할 수 있도록 지원하며, 애플리케이션의 요구 사항에 맞게 서버를 쉽게 확장하거나 축소할 수 있다. EC2는 클라우드 컴퓨팅의 핵심 요소로, 온디맨드 컴퓨팅 리소스를 제공함으로써 기업이 물리적 하드웨어의 구매 및 관리 부담을 줄일 수 있게 해준다.

ii) VPC: VPC는 사용자가 AWS 클라우드 내에 개인적인 가상 네트워크를 생성할 수 있게 해주는 서비스이다. 사용자는 자신의 IP 주소 범위를 선택하고, 서브넷, 라우트 테이블, 네트워크 게이트웨이

등을 구성하여 자신만의 격리된 섹션을 만들 수 있다. 이는 데이터 센터 내에 물리적으로 격리된 네트워크를 구성하는 것과 유사한 환경을 제공하며, 보안과 네트워크 구성의 유연성을 크게 향상시킨다.

iii) IAM: IAM은 AWS 리소스에 대한 액세스를 보안적으로 관리할 수 있게 해주는 서비스이다. 사용자, 그룹, 역할을 만들어 특정 AWS 리소스에 대한 액세스 권한을 세밀하게 제어할 수 있다. IAM을 통해 어떤 사용자가 어떤 AWS 서비스에 접근할 수 있는지 정책을 통해 관리할 수 있으며, 멀티 팩터 인증(MFA) 같은 보안 기능을 통해 계정의 보안을 강화할 수 있다.

iv) ECS: ECS는 AWS에서 제공하는 컨테이너 관리 서비스로, Docker 컨테이너를 쉽게 배포, 관리 및 스케일링할 수 있게 해준다. ECS는 컨테이너화된 애플리케이션을 자동으로 배치하고, 로드 밸런싱 및 오토 스케일링과 같은 기능을 통해 애플리케이션의 가용성과 확장성을 관리한다.

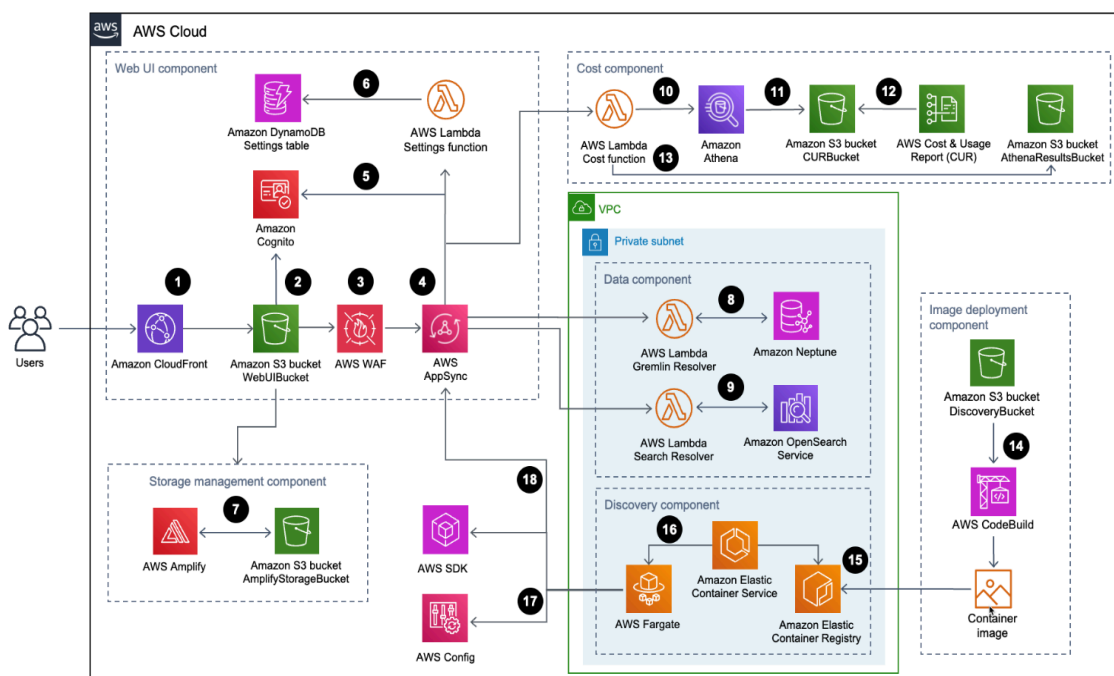


그림 5. AWS Architecture

④ GCP: GCP는 구글이 제공하는 클라우드 컴퓨팅 서비스로, 컴퓨팅, 스토리지, 데이터베이스, 머신러닝, 네트워킹 등 다양한 클라우드 서비스를 제공한다. GCP는 구글의 글로벌 인프라를 기반으로 하여 높은 확장성, 안정성, 보안성을 갖추고 있으며, 기업들이 디지털 트랜스포메이션을 가속화할 수 있도록 지원한다.

i) Compute Engine: Compute Engine은 GCP의 IaaS(Infrastructure as a Service) 솔루션으로, 가상 머신 인스턴스를 제공한다. 사용자는 다양한 머신 타입, 운영 체제, 디스크 옵션을 선택하여 워크로드에 맞는 가상 머신을 구성할 수 있으며, 오토스케일링 기능을 통해 트래픽 변화에 따라 자동으로 리소스를 조정할 수 있다.

ii) Cloud Storage: Cloud Storage는 GCP의 객체 스토리지 서비스로, 대용량 데이터를 안전하게 저장하고 액세스할 수 있게 해준다. 데이터의 중요도와 액세스 빈도에 따라 Standard, Nearline,

Coldline, Archive 등 다양한 스토리지 클래스를 제공하여 비용 효율적인 데이터 관리가 가능하다.

iii) Cloud VPC: Cloud VPC(Virtual Private Cloud)는 GCP 내에서 사용자가 자신만의 가상 네트워크를 구성할 수 있게 해주는 서비스이다. IP 주소 범위, 서브넷, 방화벽 규칙 등을 설정하여 보안성이 강화된 네트워크 환경을 구축할 수 있다.

iv) Cloud IAM: Cloud IAM(Identity and Access Management)은 GCP 리소스에 대한 접근 권한을 세밀하게 제어할 수 있는 서비스이다. 사용자, 그룹, 서비스 계정에 대한 권한을 관리하고, 최소 권한 원칙에 따라 필요한 권한만 부여함으로써 보안을 강화할 수 있다.

v) GKE: GKE(Google Kubernetes Engine)는 컨테이너화된 애플리케이션을 배포, 관리, 스케일링할 수 있는 관리형 Kubernetes 서비스이다. GKE를 사용하면 인프라 관리에 신경 쓰지 않고 애플리케이션 개발에 집중할 수 있으며, 구글의 강력한 네트워킹, 모니터링, 로깅 기능과 통합하여 운영 효율성을 높일 수 있다.

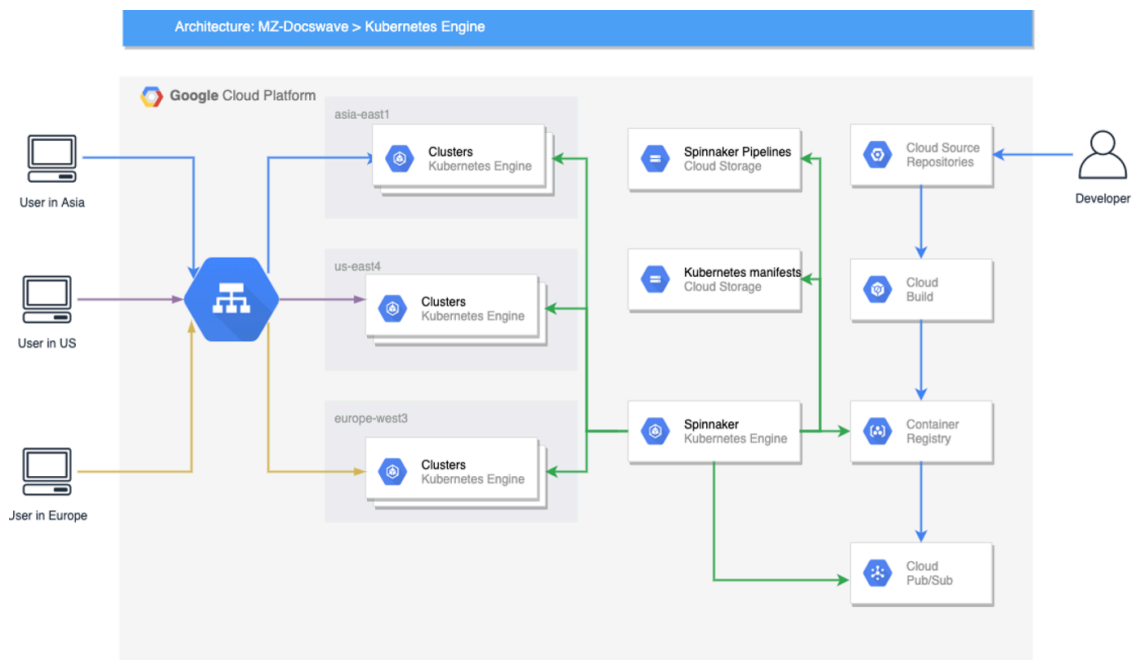


그림 6. GCP Architecture

⑤ OpenStack: OpenStack은 기업과 조직이 프라이빗 또는 퍼블릭 클라우드 인프라를 생성, 배포 및 관리할 수 있도록 설계된 포괄적인 오픈 소스 클라우드 컴퓨팅 플랫폼이다. 이는 유연하고 확장 가능하며 비용 효율적인 솔루션을 제공하여 컴퓨팅, 스토리지 및 네트워킹과 같은 광범위한 클라우드 리소스를 관리한다. 오픈스택 플랫폼은 여러 모듈식 서비스로 구성되어 있으며 각각은 클라우드 시스템 내의 특정 기능을 담당하고, 서비스들끼리 상호 연결되어 동작한다.

i) Nova: Nova는 오픈스택의 컴퓨팅 서비스로, 가상머신 인스턴스의 프로비저닝과 관리를 담당한다.

Nova를 사용하면 사용자는 가상머신 인스턴스를 생성, 시작, 정지, 종료하는 등의 작업을 수행할 수 있으며, 리소스의 가용성과 성능을 고려하여 자동으로 배치 및 관리한다. Nova는 다양한 가상화 백엔드와 통합되어 가상머신 인스턴스의 생성과 관리를 지원한다.

ii) Neutron: Neutron은 오픈스택의 네트워크 서비스로, 가상 네트워크 및 네트워크 리소스를 프로비저닝하고 관리하는 기능을 제공한다. Neutron을 사용하면 사용자는 가상 네트워크, 서브넷, 라우터, 로드 밸런서 등을 생성하여 가상머신 인스턴스 간의 통신과 네트워크 연결을 관리할 수 있다. 또한, Neutron은 다양한 네트워크 토폴로지와 고급 네트워크 서비스를 지원한다.

iii) Cinder: Cinder는 Nova 서비스가 제공하는 인스턴스에 지속적으로 사용이 가능한 블록 스토리지 장치를 제공한다. 여기서 블록 스토리지 시스템은 블록 장치를 생성하고 서버에 부착하고 분리하는 업무를 담당한다.

iv) Keystone: Keystone은 오픈스택의 식별, 인증, 권한 부여 서비스이다. Keystone은 오픈스택 클라우드 환경에서 사용자, 서비스, 역할 등의 식별 정보를 관리하고 보안 인증 및 권한 부여를 처리한다.

v) Glance: Glance는 오픈스택의 이미지 서비스로, 가상머신 및 컨테이너 등 가상화 환경에서 사용되는 이미지 관리를 담당한다. Glance는 이미지를 중앙 집중식으로 관리하고 이를 사용자에게 제공하는 역할을 수행한다. Glance는 다양한 이미지 형식을 지원하며, 사용자는 자체 생성한 이미지나 공개적으로 제공되는 이미지를 사용할 수 있다.

vi) Horizon: Horizon은 오픈스택의 웹 기반 대시보드 인터페이스이다. Horizon은 사용자가 오픈스택 클라우드 환경을 관리하고 모니터링할 수 있는 그래픽 사용자 인터페이스를 제공한다. 이를 통해, 사용자는 가상머신 인스턴스를 생성하고 관리하며, 네트워크 및 스토리지 설정을 구성하고, 사용자 및 프로젝트 관리 등 오픈스택의 다른 구성요소와 관련된 작업들을 수행할 수 있다.

vii) Heat: Heat는 오픈스택의 오케스트레이션 서비스로, 클라우드 환경에서 인프라 리소스를 자동으로 프로비저닝하고 관리하는 기능을 제공한다. Heat를 사용하면 사용자는 템플릿을 정의하여 인프라를 구성하고, 해당 템플릿을 실행하여 자원을 프로비저닝할 수 있다.

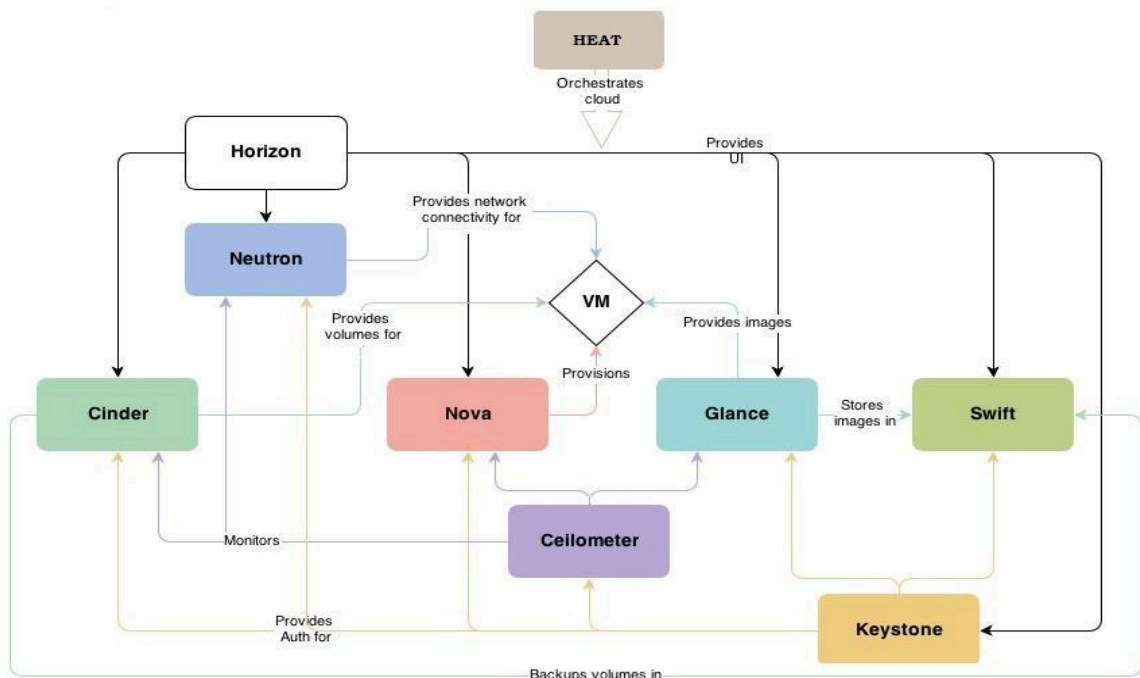


그림 7. OpenStack Architecture

⑥ Prometheus: Prometheus는 오픈 소스 시스템 모니터링 및 알림 툴킷이다. 시계열 데이터베이스를 기반으로 메트릭을 수집하고 저장하며, 강력한 쿼리 언어(PromQL)를 통해 데이터를 분석하고 시각화할 수 있다. Prometheus는 서비스 디스커버리(Service Discovery) 기능을 통해 모니터링 대상을 자동으로 발견하고, 다양한 알림 규칙을 설정하여 시스템 이상을 감지할 수 있다. 특히 Kubernetes와 같은 컨테이너 환경에서 많이 사용되며, 마이크로서비스 아키텍처의 모니터링에 적합하다.

i) Prometheus server: Prometheus Server는 Prometheus 본체이다. Prometheus Server 자체는 각 모니터링 대상에서 Metrics(메트릭스)를 수집하거나, 수집한 Metrics에 대해 쿼리를 실행하여 Metrics 정보를 참조하거나, 자동으로 내부적으로 정기적으로 쿼리를 실행하여 경고를 관리하거나 하는 것을 담당한다.

ii) Service discovery: Service Discovery는 모니터링되는 정보를 자동으로 받아오는 구조이다. 이를 사용함으로써, 클라우드 플랫폼 또는 특정 소프트웨어 등의 해당 API를 주기적으로 호출하여, 거기에 등록된 인스턴스 정보를 수집한다.

iii) Exporter: Exporter는 감시 에이전트이다. Exporter는 모니터링 대상에서 Metrics를 수집하여 Prometheus에 공개한다. 예를 들어, Nginx의 CPU 사용률이 어떤지, Request가 몇 건 오고 있는지 등의 정보는 받아올 수 있는 있지만, 각각의 포맷을 Prometheus는 모른다. 매번 각 포맷에 대한 형식을 Prometheus에서 유지 보수하는 것은 어렵다. 그래서 Exporter를 둬으로써, 대상의 시스템으로부터 Metrics 정보를 받아올 수 있다. 받아온 Metrics 정보를 Prometheus가 읽을 수 있는 형태로 변환해 주는 기능을 제공한다.

iv) Alert Manager: 설정된 Rule에 따른 알림 Notification을 담당한다. Slack 등에 연동해서 알림 시스템 구축이 가능하다.

v) PromQL: PromQL은 Prometheus Query Language의 약자이다. Metrics 라벨로 필터링 할 수 있는 기능을 제공한다. 예를 들어, 인스턴스의 이름, IP 주소, 클라우드 플랫폼의 리전 등을 라벨로 사용할 수 있다.

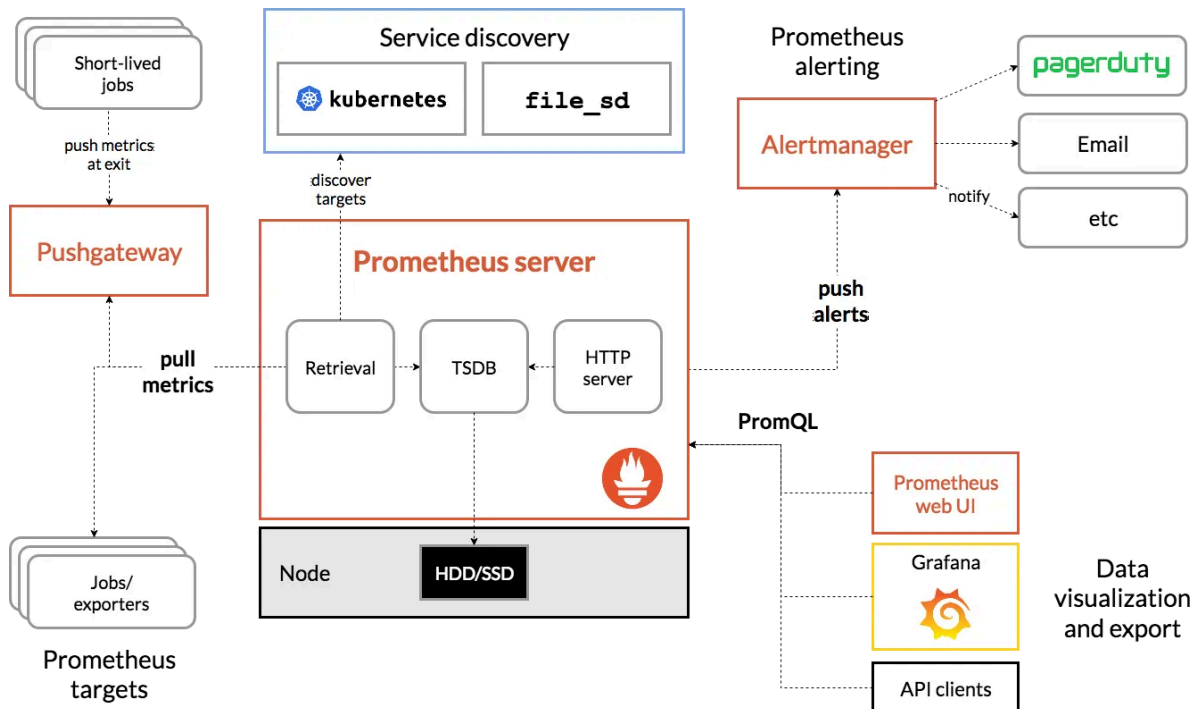


그림8. Prometheus Architecture

⑦ Grafana: Grafana는 메트릭 데이터를 시각화하고 분석하기 위한 오픈 소스 대시보드 플랫폼이다. Prometheus, Elasticsearch, InfluxDB 등 다양한 데이터 소스와 연동할 수 있으며, 사용자 정의 대시보드를 통해 시스템 성능, 애플리케이션 메트릭, 비즈니스 지표 등을 실시간으로 모니터링할 수 있다. Grafana는 알림 기능을 제공하여 특정 조건이 충족될 때 이메일, Slack 등 다양한 채널로 알림을 보낼 수 있으며, 대시보드 공유 및 팀 협업 기능도 지원한다.

i) Distributor: Distributor는 Jaeger, OTLP, Zipkin 등 다양한 포맷의 트레이스 데이터를 수신하는 역할을 한다. 받은 트레이스는 traceID를 기준으로 해시해서 여러 Ingester로 분산시킨다. 데이터 유실을 막기 위해 일시적인 버퍼링도 수행할 수 있다.

ii) Ingester: Ingester는 실시간으로 들어오는 트레이스 데이터를 메모리에 저장하고, 일정 시간이나 조건에 따라 블록 단위로 압축해 오브젝트 스토리지에 업로드한다. 동시에 쿼리 요청이 들어오면 메모리에 있는 데이터를 바로 제공할 수 있다. 데이터 정합성과 가용성도 책임진다.

iii) Query Frontend: Query Frontend는 사용자의 쿼리를 받아서 병렬 처리하기 좋게 나누고, 캐싱을 통해 응답 속도를 높인다. 부하를 분산시켜 전체 시스템의 성능을 안정적으로 유지하는 데 도움을 준다. Querier 앞단에서 프록시처럼 작동한다.

iv) Querier: Querier는 쿼리 실행을 실제로 담당하는 컴포넌트다. Ingestor와 오브젝트 스토리지에서 데이터를 조회해 사용자에게 전달한다. 필터링, 정렬, 트레이스 재구성 같은 작업도 맡는다.

v) Compactor: Compactor는 오브젝트 스토리지에 저장된 블록들을 주기적으로 압축하고 병합하는 역할을 한다. 이 과정에서 중복 데이터를 제거하고, 저장 공간과 조회 성능을 모두 최적화한다. 자동으로 백그라운드에서 동작한다.

vi) Metrics generator: Metrics Generator는 트레이스 데이터를 분석해 지연시간, 오류율 같은 메트릭을 만든다. 이렇게 생성된 메트릭은 Prometheus 같은 모니터링 톨과 연동해서 알림이나 대시보드로 활용된다. 트레이스 기반의 인사이트를 제공하는 데 핵심적인 역할을 한다.

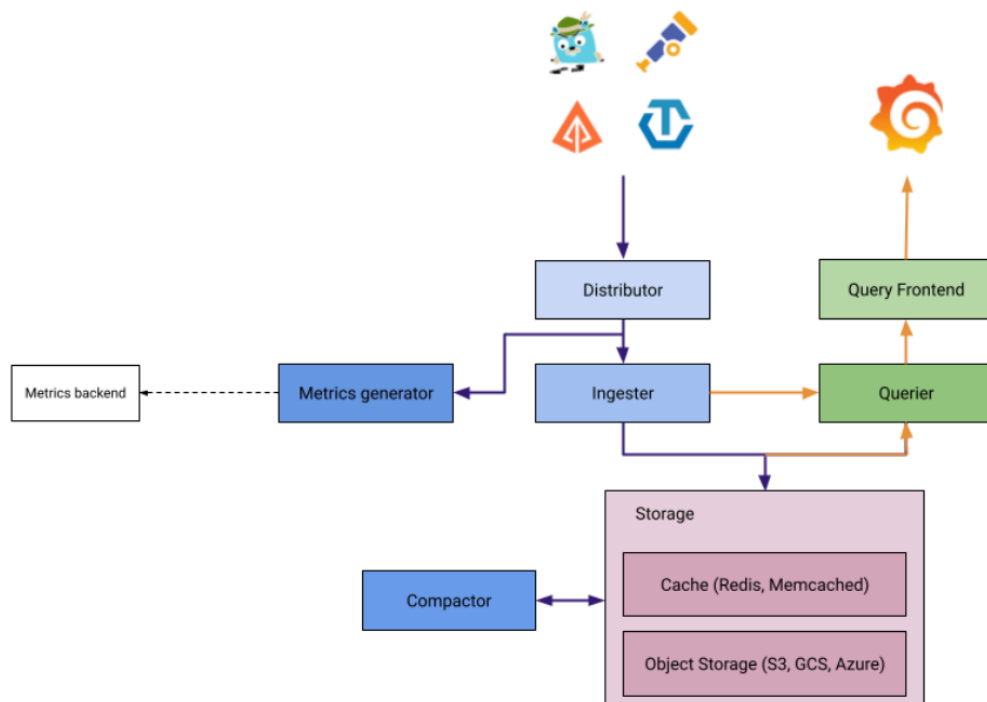


그림 9. Grafana Architecture

⑧ Jaeger: Jaeger는 마이크로서비스 환경에서 분산 트레이싱을 위한 오픈 소스 솔루션이다. 복잡한 시스템에서 서비스 간 요청 흐름을 추적하고, 성능 병목 현상과 오류를 식별하는 데 도움을 준다. Jaeger는 OpenTracing/OpenTelemetry 표준을 지원하며, 트레이스 데이터를 수집, 저장, 쿼리 및 시각화하는 기능을 제공한다. 또한 다양한 스토리지 백엔드(Elasticsearch, Cassandra 등)를 지원하고, 서비스 의존성 분석 및 성능 최적화에 필요한 인사이트를 제공한다.

i) Jaeger client: OpenTracing API로 만들어진 언어별 구현체. OpenTracing API는 CNCF(클라우드

네이티브 컴퓨팅 재단) 산하의 프로젝트로, 애플리케이션 간 분산 추적을 위한 표준처럼 사용되는 비공식 API이다.

ii) Jaeger agent: 호스트에서 실행되는 네트워크 데몬으로, UDP를 통해 전송된 Span(분산 추적에서 기본이 되는 블록 단위, 작업 단위)을 처리한후 Collector로 trace를 전송한다. 타겟 애플리케이션과 같은 호스트에 배치한다.

iii) Jaeger-collector: Agent로부터 Trace를 수신하여 처리 파이프라인을 통해 유효성 검사, index 생성/변환을 수행한후 DB에 저장한다.

iv) Jaeger-query: DB에서 trace를 조회 후 UI구성에 필요한 API를 제공한다.

v) UI: 사용자 친화적인 웹 인터페이스를 제공한다.

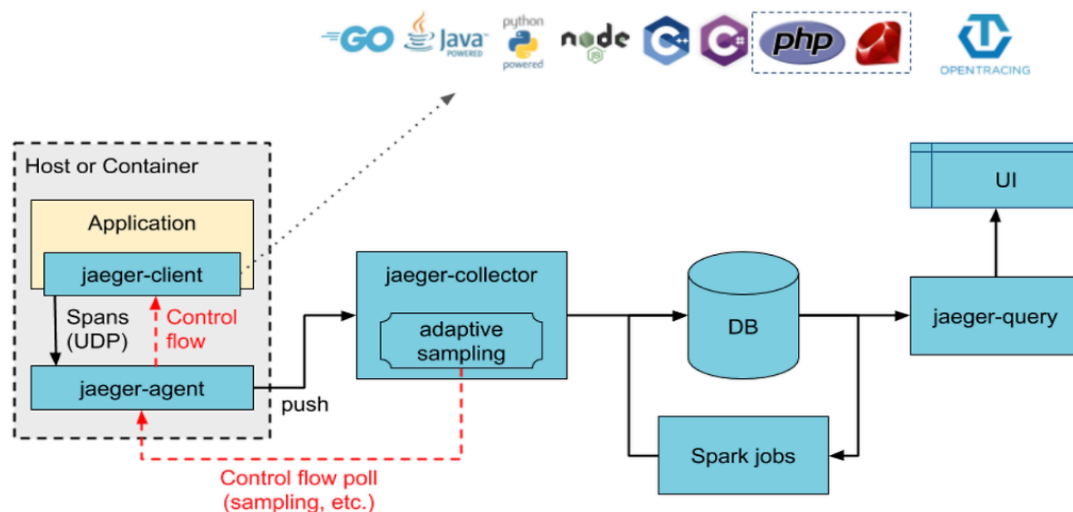


그림 10. Jaeger Architecture

⑨ Terraform: Terraform은 HashiCorp에서 개발 Infrastructure as Code(IaC) 도구로, 클라우드 및 온프레미스 리소스를 코드로 관리하고 배포할 수 있게 해준다. 선언적인 구성 파일을 통해 인프라를 정의하고, 실행 계획을 미리 확인한 후 배포할 수 있다. Terraform은 AWS, GCP, Azure 등 다양한 클라우드 제공업체와 통합되어 있으며, 모듈화 및 재사용성을 통해 인프라 관리의 효율성을 높인다. 상태 관리 기능을 통해 인프라의 현재 상태를 추적하고, 협업을 위한 원격 상태 저장소도 지원한다. [그림 11]은 Terraform의 아키텍처이며, 아키텍처를 구성하는 컴포넌트의 설명은 아래와 같다.

i) Terraform Core: Terraform CLI라고도 하는 Terraform Core는 Go 언어를 사용하여 개발된 정적으로 컴파일된 바이너리를 기반으로 만들어졌다. Terraform 사용자를 위한 기본 인터페이스 역할을 한다.

ii) Terraform Provider: Terraform provider는 테라폼이 다양한 범주의 서비스 및 리소스와 통신을

가능하게끔 해주는 모듈이다. 각 Provider는 특정 서비스 내에서 Terraform이 관리할 수 있는 리소스에 대한 정의 및 Terraform configuration 내용을 해당 서비스에 대한 특정 API 호출로 변환하는 역할을 가지고 있다.

iii) State File: state file은 테라폼의 기능 중 필수 요소로, 테라폼이 관리하는 리소스 정보뿐만 아니라, 리소스에 대한 현재 상태, 종속성 등을 저장하는 JSON 파일이다. Terraform은 State file을 활용하여 새로운 테라폼 configuration이 적용될 때에 실제 인프라 리소스에 반영되어야 할 변경사항을 결정한다. State file에는 현재 관리 중인 인프라에 대한 민감한 정보가 포함되어 있으므로, 안전하게 보호 및 보관되어야 하고, 자주 백업하는 것이 좋다.

iv) Terraform Provisioners: 새로 생성된 리소스나 인스턴스에 대해 스크립트나 명령을 실행할 수 있도록 하는 기능이다. 여기에 사용되는 스크립트는 인프라 설정 및 구성, 소프트웨어 설치, 테스트 실행, 기타 필요한 작업 수행 등의 다양한 용도로 사용될 수 있다. Provisioner는 리소스가 생성된 후에 실행되며, 리소스가 파괴될 때에 트리거되는 것도 가능하다.

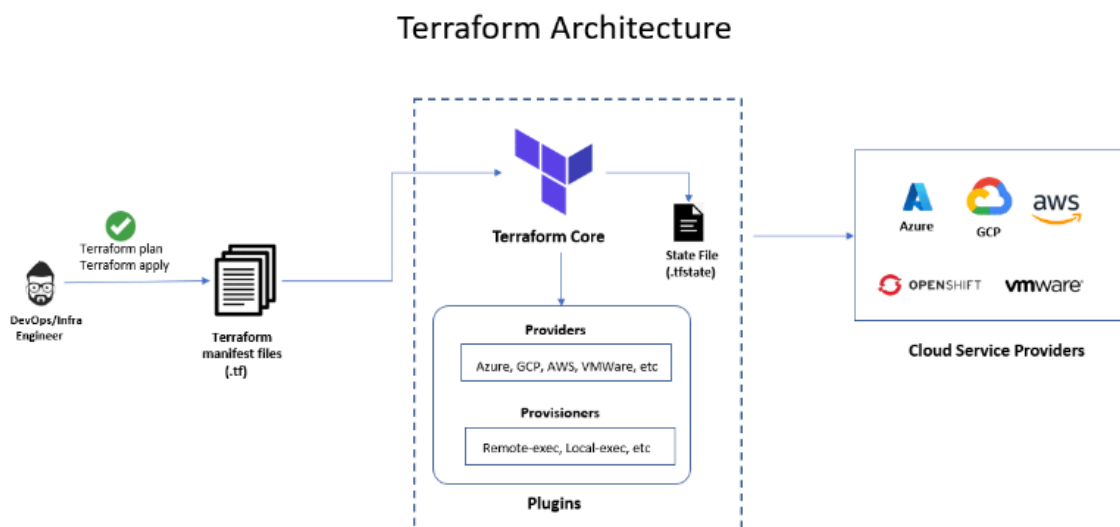


그림 11. Terraform Architecture

⑩ Kubeflow: Kubeflow는 Kubernetes 기반의 머신러닝 워크플로우를 구축하고 배포하기 위한 오픈 소스 플랫폼이다. 데이터 준비, 모델 훈련, 모델 서빙, 파이프라인 자동화 등 머신러닝 라이프사이클의 모든 단계를 지원한다. Kubeflow는 TensorFlow, PyTorch, MXNet 등 다양한 ML 프레임워크와 통합되어 있으며, 분산 훈련, 하이퍼파라미터 튜닝, 모델 버전 관리 등의 기능을 제공한다. 또한 Jupyter Notebook을 통한 대화형 개발 환경과 ML 워크플로우 관리를 위한 파이프라인 기능도 지원한다. [그림12]는 Kubeflow의 아키텍처이며 각 컴포넌트의 설명은 다음과 같다.

i) Central Dashboard: 웹 브라우저를 통해 대시보드 UI로 Notebooks, Experiments (AutoML), Experiments (KFP) 등의 컴포넌트를 이용할 수 있다.

ii) Notebooks: 웹 브라우저에서 파이썬 코드를 작성하고 실행할 수 있는 주피터(Jupyter) Notebook

개발 도구를 제공한다. 이미지 경로와 자원 등을 설정하여 쿠버네티스상에 Notebook을 생성할 수 있다. 사용자는 생성한 Notebook을 이용해 데이터 전처리와 탐색적 데이터 분석 등을 수행하여 머신러닝 모델 코드를 개발할 수 있다.

iii) Training Operators: Training Operators는 텐서플로우(TensorFlow), PyTorch, MXNet 등 다양한 딥러닝 프레임워크에 대해 분산 학습을 지원한다. 쿠버네티스 상에서 머신러닝 모델을 분산 학습하여 학습에 드는 시간을 줄일 수 있다. 사용자가 분산 학습 명세서를 작성하여 쿠버네티스에 배포하면 쿠브플로우 Training Operator는 명세서에 따라 워크로드를 실행한다. 명세서에는 머신러닝 모델 코드를 담고 있는 도커 이미지 경로와 분산 학습 클러스터 정보 등을 정의한다.

iv) Experiments(AutoML): AutoML은 머신러닝 모델의 예측 정확도와 성능을 높이기 위한 반복 실험을 자동화하는 도구이다. Kubeflow에서는 카티브(Katib)를 사용하여 AutoML 기능을 제공한다. 카티브는 하이퍼 파라미터 튜닝(Hyper Parameter Tuning), 뉴럴 아키텍처 탐색(Neural Architecture Search, NAS) 기능이 있다. 하이퍼 파라미터 튜닝은 모델의 하이퍼 파라미터를 최적화하는 작업이고 NAS는 모델의 구조, 노드 가중치 등 뉴럴 네트워크 아키텍처를 최적화하는 작업이다. 카티브를 이용하면 학습률(Learning rate), 하이퍼파라미터 중 어느 값이 모델의 예측 정확도를 높이는 값인지 찾는 실험을 자동화할 수 있다. 먼저 Experiment(AutoML) 명세서를 작성한 후 쿠버네티스에 배포하면 카티브가 명세서에 정의한 하이퍼 파라미터와 병렬 처리 설정에 따라 실험을 동시에 수행하여 가장 성능이 좋은 하이퍼 파라미터를 찾는다.

v) KServe: Kubeflow는 KServe를 통해 쿠버네티스에 머신러닝 모델을 배포하고 추론 기능을 제공한다. KServe는 Endpoint, Transformer, Predictor, Explainer로 이루어져 있고, Endpoint가 Predictor에 데이터를 전달하면 Predictor는 데이터를 예측하거나 분류한다. Endpoint는 데이터의 가중치 비율을 조절하여 Predictor에 전달할 수 있어 A/B테스트가 가능하다. Transformer나 Explainer는 필요에 따라 추가 가능하며 Predictor 와 연결되어 사용된다. Explainer는 데이터를 예측하거나 분류한 결과에 대해 판단 이유를 제시하는 설명 가능한 인공지능(eXplainable Artificial Intelligence, XAI) 역할을 하며, Transformer는 데이터 전처리, 후처리 기능을 제공한다.

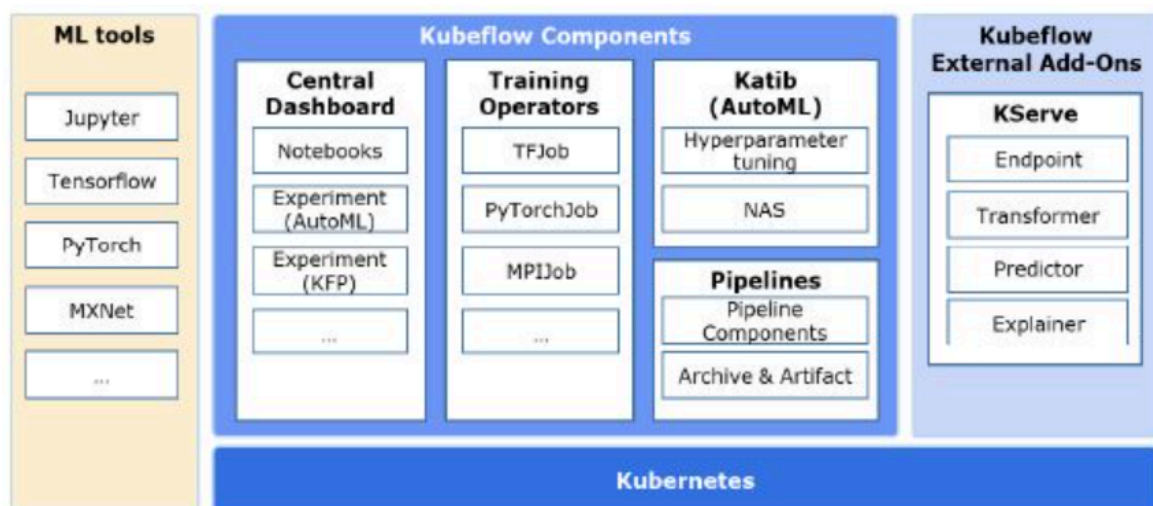


그림 12. Kubeflow Architecture

⑪ Flower: Flower는 연합학습(Federated Learning)을 위한 오픈 소스 프레임워크이다. Client-Server 아키텍처를 기반으로 하며, 각 Client는 로컬 데이터로 모델을 훈련하고 모델 업데이트만 Server에 전송한다. Server는 이러한 업데이트를 집계하여 글로벌 모델을 개선한다.

[그림 13]은 Flower의 Architecture를 나타낸 것으로, 서버 측 구성 요소와 클라이언트 측 구성 요소로 나뉘며, 각각 다음과 같이 구성된다.

i) Flower Server

1. SuperLink: 클라이언트(SuperNode)에게 연합학습 작업 지시를 전달하고, 클라이언트로부터 작업 결과를 수신한다.
2. ServerApp: 짧은 시간 실행되는 프로젝트 전용 서버 코드로, 클라이언트 선택, 클라이언트 설정, 결과 집계 등 연합학습 서버 측 로직을 정의한다.

ii) Flower Client

1. SuperNode: Server의 SuperLink에 연결되어 연합학습 작업 요청을 받고, 로컬 모델 학습을 수행한 후 결과를 반환하는 프로세스이다.
2. ClientApp: 짧은 시간 실행되는 프로젝트 전용 클라이언트 코드로, 로컬 모델 학습, 평가, 전처리 및 후처리 등 클라이언트 측 연합학습 로직을 정의한다.

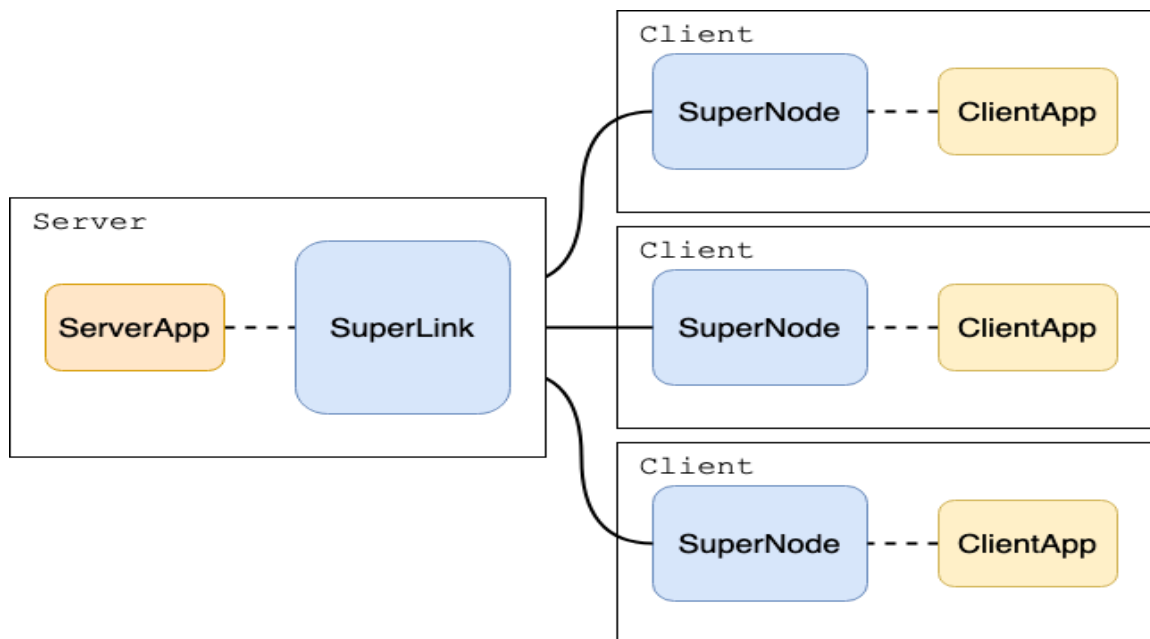


그림 13. Flower Architecture

5. 개발 일정 및 역할 분담

1) 개발 일정

표 3 역할분담

| 기간 | 5월 | | 6월 | | 7월 | | | | | | | 8월 | | | | 9월 | |
|-------------------------------|-----|---|-----|---|-----|---|---|---|-----|-----|---|----|---|-----|---|-----|---|
| 수행내용 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 1 | 2 |
| 필요 지식 습득 | ALL | | | | | | | | | | | | | | | | |
| 멀티 클라우드 인증 및 연동 | | | 전진혁 | | | | | | | | | | | | | | |
| 멀티 클라우드 인프라 자동화(IaC) 개발 | | | 김민경 | | | | | | | | | | | | | | |
| 연합학습 집계자 배치 최적화 구현 | | | 박재일 | | | | | | | | | | | | | | |
| 연합학습 VM 모니터링 개발 | | | | | 김민경 | | | | | | | | | | | | |
| 연합학습 수행 기능 구현 | | | | | 전진혁 | | | | | | | | | | | | |
| MLOps 파이프라인 구축 | | | | | 박재일 | | | | | | | | | | | | |
| 연합학습 동적 오케스트레이션 기능 구현 | | | | | | | | | 김민경 | | | | | | | | |
| 연합학습 관리 기능 개발 | | | | | | | | | | 전진혁 | | | | | | | |
| 연합학습 참여자 VM 모니터링 시스템 구현 | | | | | | | | | | 박재일 | | | | | | | |
| 테스트 및 보완 | | | | | | | | | | | | | | ALL | | | |
| 최종보고서 작성 | | | | | | | | | | | | | | | | ALL | |

2) 역할 분담

표 4 역할 분담

| 이름 | 역할 |
|-----|---|
| 공통 | <ul style="list-style-type: none"> 필요 지식 습득, 테스트 및 보완, 최종보고서 작성 |
| 전진혁 | <ul style="list-style-type: none"> 멀티 클라우드 인증 및 연동 연합학습 수행 기능 구현 연합학습 관리 대시보드 개발 |
| 김민경 | <ul style="list-style-type: none"> 멀티 클라우드 인프라 자동화(IaC) 구축 연합학습 참여자 VM 모니터링 개발 연합학습 동적 오케스트레이션 기능 구현 |
| 박재일 | <ul style="list-style-type: none"> MLOps 파이프라인 구축 연합학습 집계자 관리 기능 구현 연합학습 집계자 배치 최적화 구현 |