국민대학교
소프트웨어융합대학
소프트웨어학부

캡스톤 디자인 I
종합설계 프로젝트

프로젝트 명	최적의 GenAIOps 환경을 제공하는 Platform
팀 명	SSKAI
문서 제목	수행결과보고서

Version	2.3
Date	2024-05-23

팀원	송 무현 (조장)
	김 규민
	김 유림
	문 지훈
	박 정명
	정 승우
지도교수	김 장호 교수

문서 정보 / 수정 내역

Filename	수행결과보고서-최적의_GenAIOps_환경을_제공하는_Platform.docx
원안작성자	송무현, 김규민, 김유림, 문지훈, 박정명, 정승우
수정작업자	송무현, 김규민, 김유림, 문지훈, 박정명, 정승우

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2024-03-25	송무현	1.0	최초 작성	
2024-04-01	송무현	1.1	그림 추가	각종 아키텍처 및 알고리즘 내용 추가
2024-05-20	송무현	2.0	최종 결과 반영	실제 연구 및 개발 완료된 내용 반영
2024-05-23	송무현	2.1	그림 추가 및 오타 수정	
2024-05-23	송무현	2.2	부록 추가	
2024-05-23	정승우	2.3	그림 크기 수정	그림 크기 수정 및 내용 검토

목 차

1 배경 지식

1.1 클라우드 컴퓨팅 환경에서의 Generative AI

최근, 텍스트, 이미지, 음악, 비디오 등 다양한 형태의 콘텐츠 생성에 사용되는 생성형 AI (Generative AI) 모델이 등장하고 있으며 실제 우리 실생활에서도 많이 이용되는 ChatGPT와 같은 대표적인 사례가 존재한다.

ChatGPT는 학습 데이터셋과 모델의 크기를 크게 늘림으로써 성능을 향상시켰으며, 이는 클라우드 컴퓨팅에서 제공하는 방대한 인프라 환경이 있었기 때문에 가능한 혁신이라고 여겨진다. 특히 확장성, 접근성, 비용 효율성 등에서 클라우드 컴퓨팅은 기존 일반 데이터센터에 비해 우월한 이점을 가지고 있다.

현재 Generative AI를 클라우드 환경에서 배포하면서 가장 고려되는 지표는 비용과 성능 크게 2가지로 볼 수 있다. 또한, Generative AI가 등장하면서 여러가지 클라우드 제공업체에서 제공하는 완전 관리형 (Fully-managed) 형태의 Generative AI. Machine Learning 서비스가 등장하고 있다.

이때, 완전 관리형으로 제공하는 Generative AI 서비스 중 대표적인 서비스인 Amazon Bedrock의 경우, Generative AI Model을 제작하는 회사와 라이센스를 맺어 Anthropic의 Claude나 Stability AI의 Stable Diffusion과 같은 파운데이션 모델을 즉시 이용할 수 있도록 제공하고, 사용자가 본인의 도메인에 맞도록 Fine-Tuning 또한 제공한다. 파운데이션 모델은 수많은 Raw Data를 학습하여 광범위한 작업에 응용할 수 있는 Model을 칭한다 [1].

1.2 서버리스 컴퓨팅

2014년 Amazon Web Services에서 출시한 AWS Lambda를 시작으로, Function-as-a-Service (FaaS) 형태의 서버리스 컴퓨팅 모델이 대중화되기 시작되었다. 서버리스 컴퓨팅은 사용자가 직접 컴퓨팅 자원을 관리하지 않아도 자원 사용할 수 있는 패러다임이다. 그 중, FaaS 모델의 경우 실행할 코드만 작성하면 즉시 실행할 수 있다.

사용자는 해당 코드를 실행한 시간만큼만 비용을 지불하면 되기에 간헐적으로 처리되거나 API와 같은 가벼운 작업에서 주로 사용될 수 있다. 최근에는 머신러닝 추론 환경도 서버리스 컴퓨팅에서 진행하고자 하는 시도가 이어지고 있다 [2].

1.3 스팟 과금 정책

클라우드 제공업체에서 제공하는 서비스 중, 가장 대표적인 서비스인 “인스턴스(Instance)"라고 불리는 서버를 대여해주는 컴퓨팅 서비스에는 일반적으로 온디맨드(On-Demand), 예약(Reservation), 스팟(Spot) 3가지 비용 정책이 있다. 일반적으로 이용되는 과금 정책은 온디맨드로, 사용자는 즉시 컴퓨팅 자원을 이용할 수 있고, 이용한 시간만큼 비용을 지불하면 된다. Reservation 과금 정책의 경우, 사용자가 2년 또는 3년간 약정을 맺어 해당 컴퓨팅 자원을 이용할 것이라는 계약과 선납금 등을 지불하여 더 저렴하게 이용할 수 있는 과금정책이다. 마지막으로 스팟 과금 정책은 클라우드 제공업체의 경우, 온디맨드를 이용하고자 하는 사용자를 위해 대규모 데이터센터 환경을 구축해놓지만, 사용자가 이용하지 않을 경우에는 클라우드 제공업체 입장에서는 유휴 자원으로서 비용이 낭비하게 된다. 따라서 클라우드 제공업체는 남는 자원을 최대 90% 까지 저렴한 요금으로 사용자에게 제공해주는 스팟 과금 정책을 제공한다.

그러나, 스팟 과금 정책은 클라우드 제공업체가 온디맨드로 할당할 자원이 부족해지거나, 스팟으로 할당된 자원이 많아질 경우 할인율이 적용된 가격을 변경하거나 자원을 회수해갈 수 있다. 이 경우 사용자는 2분 후에 컴퓨팅 자원이 종료된다는 알림을 받게되고, 사용자는 적절하게 대응해야 한다.

1.4 MLOps & GenAIOps

Machine Learning Operations (MLOps)는 머신러닝 모델의 개발 워크플로우와 배포를 단순화하고 자동화하기 위한 일련의 관행이다 [3]. 소프트웨어 개발에서 사용되는 Development Operations (DevOps)의 원칙을 차용하여 MLOps는 모델의 개발에서 배포, 모니터링, 유지보수에 이르는 전 과정을 통합한다. Generative AI Operations (GenAIOps)는 MLOps를 확장하여 생성형 AI 솔루션을 개발 및 운영한다. MLOps와의 가장 큰 차이는 파운데이션 모델의 관리 및 상호작용이다 [4].

2 프로젝트 목표

2.1 기존 Generative AI 개발/운영 환경의 혁신

기존 MLOps 솔루션들은 사용자가 이용하기 불편한 부분들이 있었다. 예를 들어, 클라우드 제공업체에서 제공하는 MLOps 솔루션들은 해당 클라우드 제공업체에서 제공되는 네트워크, 컴퓨팅, 스토리지 서비스에 대한 이해도가 있어야 사용이 원활하게 가능하며 종속적이다.

대표적인 예시로, 특정 클라우드 제공업체에서 제공하는 머신러닝 서비스를 통해 머신러닝 모델을 학습시키기 위해서는 각기 다른 컴퓨팅 자원의 유형의 성능 정보 및 가격 정보 등을 다 알고 있어야 효율적으로 머신러닝 학습을 진행할 수 있다. 이는 기존 머신러닝 엔지니어에게 있어 진입장벽이 될 수 있다.

따라서, 이 프로젝트의 결과물을 이용하는 것을 통해 머신러닝 개발자는 학습이나 추론을 위한 인프라에 대한 생각을 하지 않고도 손쉽게 머신러닝 개발/운영 환경을 구축할 수 있게 될 것이다.

2.1.1 기존 MLOps 솔루션의 제한점

기존 MLOps 솔루션은 크게 2가지 유형이 존재한다. 첫 번째는 클라우드 제공업체에서 제공하는 완전 관리형 (Fully-managed) 머신러닝 서비스이며, 두 번째는 사용자가 직접 인프라에 배포하여 구성할 수 있는 Kubeflow, Ray와 같은 프레임워크가 존재한다.

그러나 첫 번째 예시인 클라우드 제공업체에서 제공하는 서비스의 경우에는 다음과 같은 문제점이 존재한다.

1) 특정 클라우드 제공업체에 종속되는 문제

사용자는 특정 클라우드 제공업체의 머신러닝 서비스를 이용하기 위해 해당 클라우드 제공업체의 네트워킹, 스토리지, 컴퓨팅 서비스에 대한 이해도가 필요하다. 대표적으로 AWS Sagemaker의 경우, 사용자가 학습을 진행하기 위해 어떤 컴퓨팅 자원을 사용해야 본인이 수행하고자 하는 작업에 적절한지 판단할 필요가 있다.

2) 일반적인 서비스에 비해 높은 가격 및 제한점

클라우드 제공업체의 머신러닝 서비스는 해당 제공업체의 컴퓨팅, 네트워크, 스토리지 서비스를 기반으로 동작한다. 따라서 전문적인 지식을 가진 사용자라면 해당 클라우드 제공업체가 제공하는

서비스를 이용해 유사한 서비스를 제작할 수 있으나 그럼에도 관리가 편리하다는 이유로 이는 쉽지않은 실정이다. 클라우드 제공업체가 편리함을 가져다 준다고 고려하였을 때, 대표적으로 AWS Lambda와 같은 서버리스를 이용한 서버리스 추론 서비스인 Amazon Sagemaker Serverless Inference를 메모리 GB/초당 25% 더 높은 가격에 이용해야 하며, *($0.000016, $0.00002) AWS Lambda는 메모리를 최대 10GB 까지 제공하지만, Amazon Sagemaker Serverless Inference의 경우에는 2024년 3월 기준 사용가능 최대 메모리가 6GB에서 그친다.

두 번째 예시인 사용자가 직접 인프라에 배포하여 구성할 수 있는 Kubeflow와 Ray의 경우에는 다음과 같은 문제점이 존재한다.

1) 사용자가 직접 기반 인프라를 구현해야함

오픈 소스로 제공되는 MLOps 프레임워크를 사용하기 위해서는 결국 사용자가 직접 해당 프레임워크를 구동하기 위한 인프라를 구현해야한다. 따라서 사용자는 해당 인프라를 관리할 책임과 해당 프레임워크를 관리할 책임 2가지를 모두 지게 되며 MLOps를 실현하기 위해 실제 MLOps와는 관련없는 작업을 다수 수행해야한다. 결정적으로 사용자의 숙련도에 따라 성능, 안정성, 비용이 천차만별로 달라질 수 있어 사용자의 높은 숙련도가 요구된다.

2.2 주요 기능

2.2.1 이용이 편리한 GenAIOps Pipeline 제공

사용자는 플랫폼을 이용할 수 있는 웹 서비스에 접근하여 학습용 데이터셋을 업로드하고, 파운데이션 모델을 선택하거나 본인이 설계한 ML 모델을 업로드 하는 것을 통해 Fine-tuning, Training, Deploy, Monitoring을 손쉽게 이용할 수 있다.

2.2.2 최적의 비용 및 성능 인프라 제공

모델의 크기, 모델의 연산자 수, 모델에 필요한 성능 등을 종합적으로 판단해 최적의 인프라를 선정하는 알고리즘을 개발하여 해당 모델에 맞는 최적의 비용 및 성능을 가진 인프라를 제공한다. 따라서 사용자는 비용 및 성능을 최적화 하기위해 별도로 고려하지 않아도 된다. 이 때, 최적의 인프라 선정 결과에 따라 유지비용이 전혀 들지 않는 서버리스 컴퓨팅 환경에서 추론 환경이 지속적으로 가동되도록 하거나, 학습이나 Fine-Tuning 시에는 스팟 과금 정책을 활용하여 사용자는 최대 90% 저렴한 환경에서 작업을 수행하거나, 동일한 가격 대비 더 뛰어난 성능의 컴퓨팅

자원에서 작업을 수행하도록 할 수 있다.

2.2.3 지속적 비용 Monitoring 및 CI/CD 환경 제공

학습, Fine-tuning과 추론 서비스 배포 후 운영 중 지속적인 비용 Monitoring을 통해 사용자는 실시간으로 사용한 비용을 확인할 수 있다. 이를 통해 사용자는 과금이 많이 발생할 경우, 현재 운영중인 인프라의 배포 수준을 재검토할 수 있게된다. 또한,모델을 지속적으로 학습 후 배포하여 모델 개발 및 운영의 안정성, 지속성을 손쉽게 확보할 수 있다.

3 개발 내용 및 결과물

3.1 개발 및 연구 목록

클라우드의 비용 과금 정책을 최대한으로 활용하면서, 사용자는 웹 페이지를 통해 GenAIOps Workflow를 손쉽게 구성할 수 있도록 하기 위해서 아래와 같은 여러가지 API 개발, 자동화 코드 작성 및 구현을 완료하였다.

세부적으로는 다음과 같다.

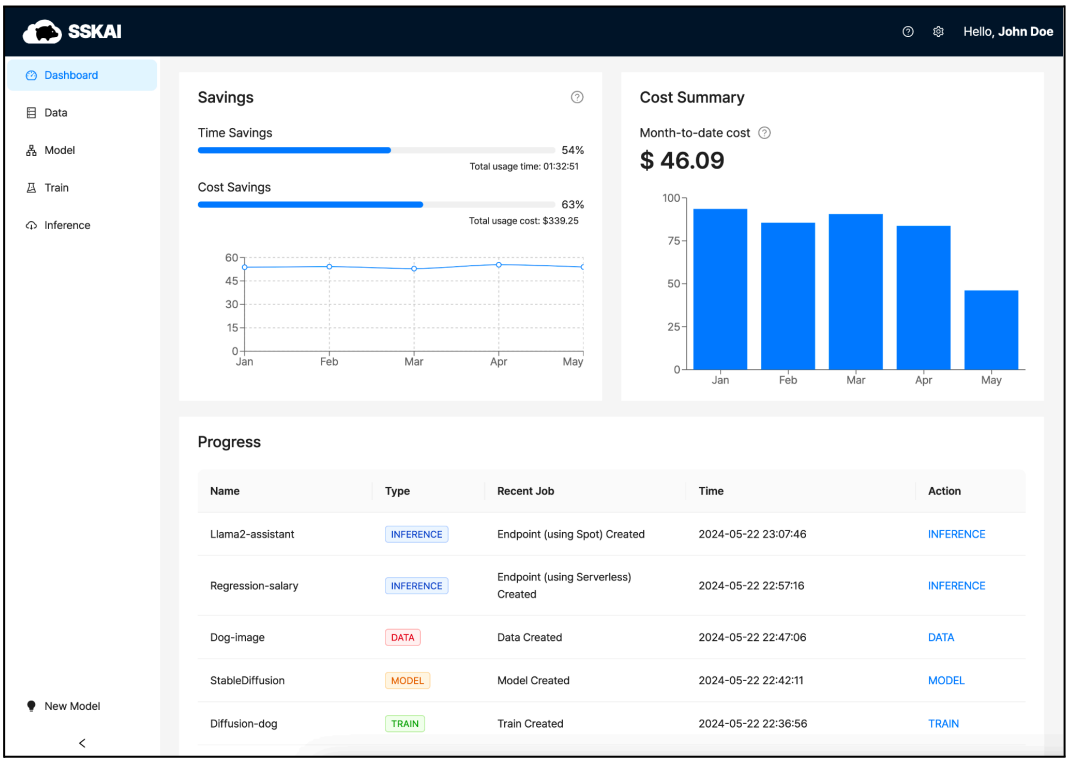
- 사용자에게 서비스를 제공할 웹 서비스
- 서버리스 환경에서 추론을 위한 아키텍처 설계
- 스팟 과금 환경에서 추론을 위한 아키텍처 설계
- 스팟 과금 환경에서 분산 학습을 위한 아키텍처 설계
- 스팟 과금 환경에서 모델 학습 시 중단이 될 수 있음을 고려한 기법 적용
- 각 모델에 맞는 최적의 비용 및 성능을 가진 컴퓨팅 자원을 선출하는 알고리즘 및 배포 플랫폼 추천 기능

3.2 연구 및 개발 내용

3.2.1 사용자에게 서비스를 제공할 웹 서비스

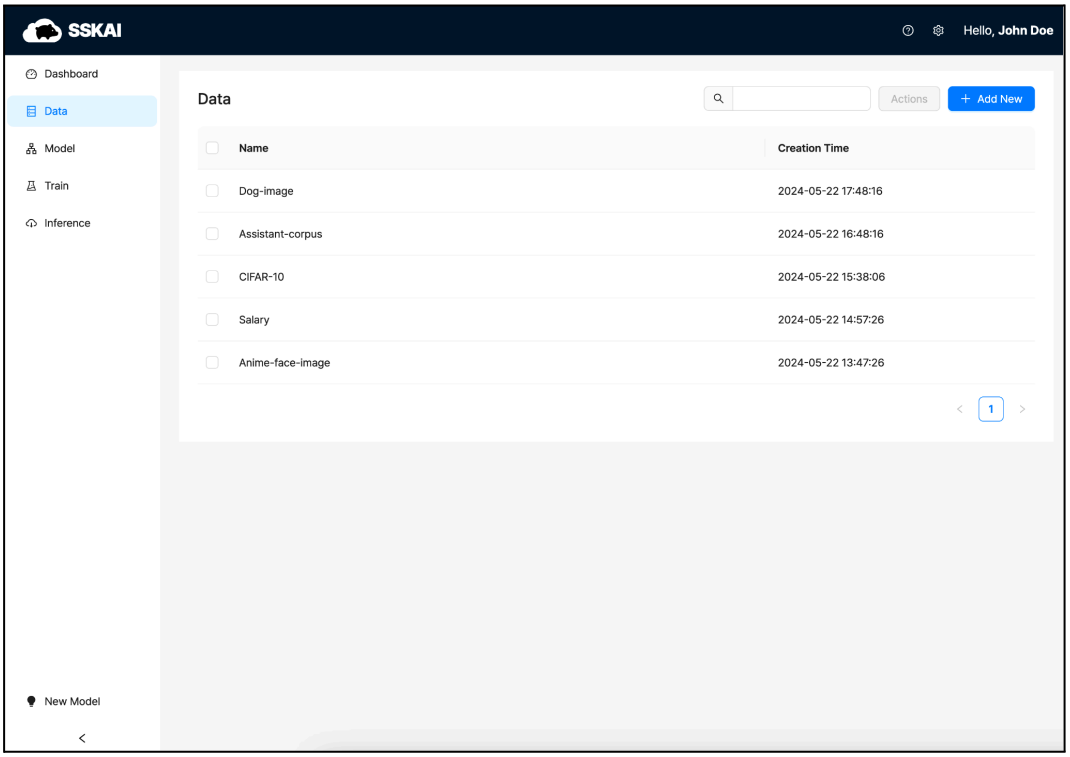
사용자는 주로 웹 서비스를 통해 이 프로젝트의 결과물을 이용할 수 있게 된다. 웹 서비스에서는 Dashboard, Model, Data, Train, Inference 페이지를 제공한다. 웹 서비스는 별도의 추가적인 프론트엔드, 백엔드나 데이터베이스 서버 구동을 위한 지속적으로 구동되는 컴퓨팅 자원이 불필요하게 설계 되었다. 정적 웹 페이지 호스팅은 Amazon S3의 정적 웹 페이지 호스팅 기능을 사용하여 구현되었으며, 모든 백엔드 API는 AWS Lambda를 사용해 서버리스 형태의 API로 구현되었다. 데이터베이스 또한, Amazon DynamoDB를 사용하여 서버리스 NoSQL 데이터베이스를 사용하여 구현되었다.

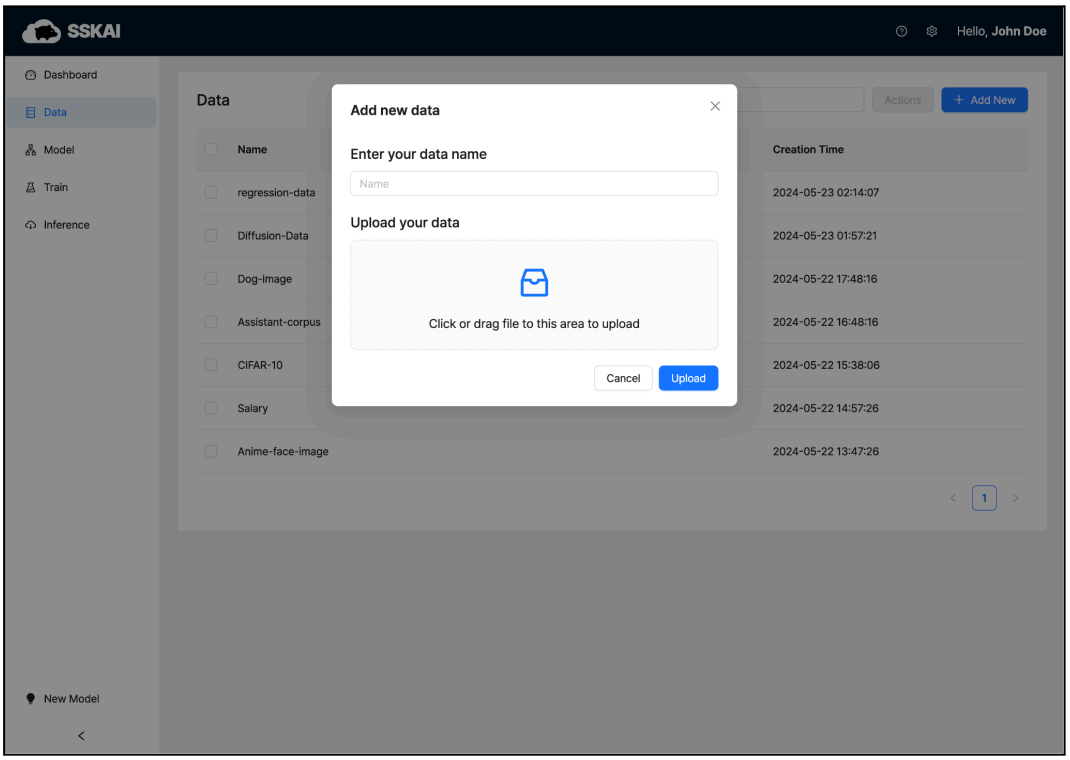
3.2.1.1. Dashboard 페이지



Dashboard 페이지에는 사용자가 이 플랫폼을 통해 낮은 비용으로 좋은 컴퓨팅 자원을 사용하여 절약한 시간과 비용을 얼마나 절약했는지에 대한 그래프와 현재 및 월별 총 사용 금액에 대한 그래프가 나타난다. 하단 부분에는 사용자가 최근에 작업한 내용이 무엇인지에 대한 로그 목록이 나타난다.

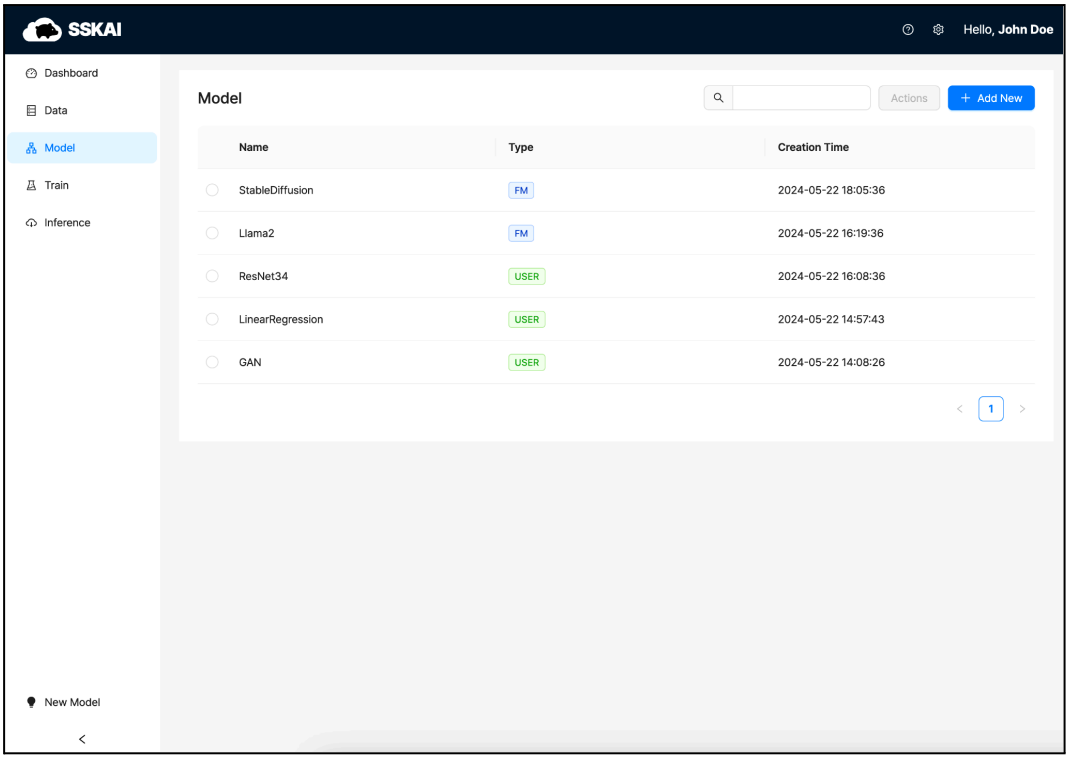
3.2.1.2. Data 페이지

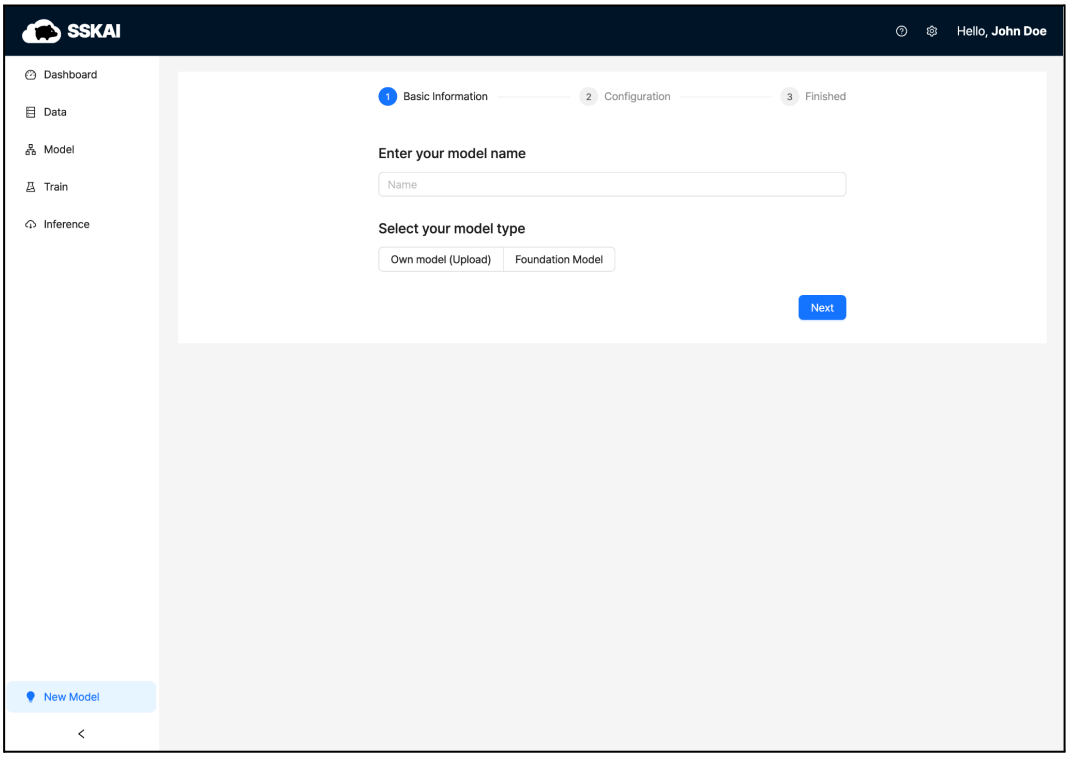


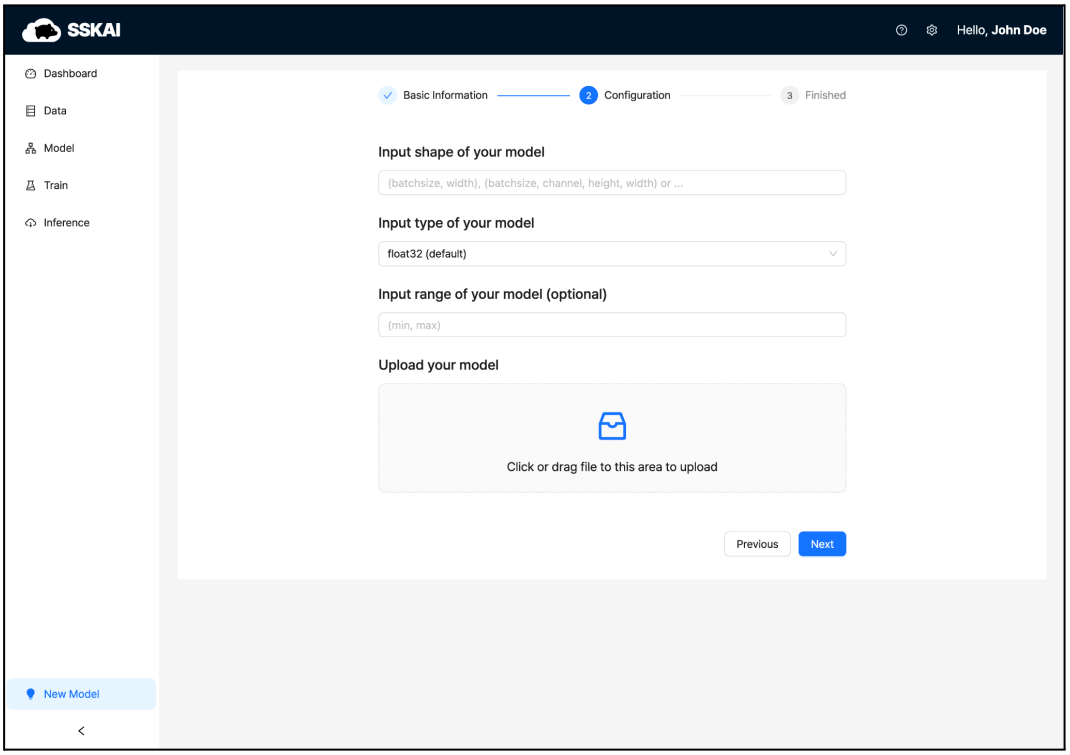


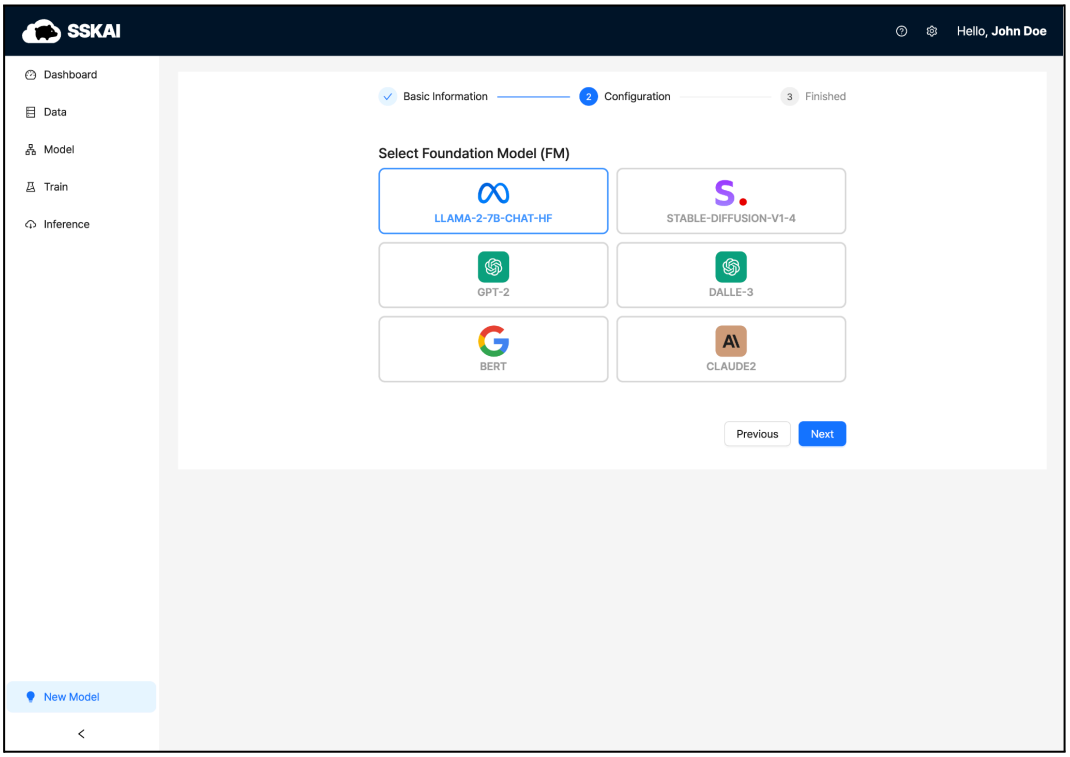
사용자는 Data 페이지를 통해 학습할 때에 이용하기 위한 데이터를 업로드할 수 있게 된다. 업로드하는 파일은 ZIP 파일로 지정되어 있으며, 사용자는 어떠한 형태의 데이터도 업로드할 수 있다. Data 페이지에서 업로드된 데이터는 학습 단계에서 활용되게 된다.

3.2.1.3. Model 페이지

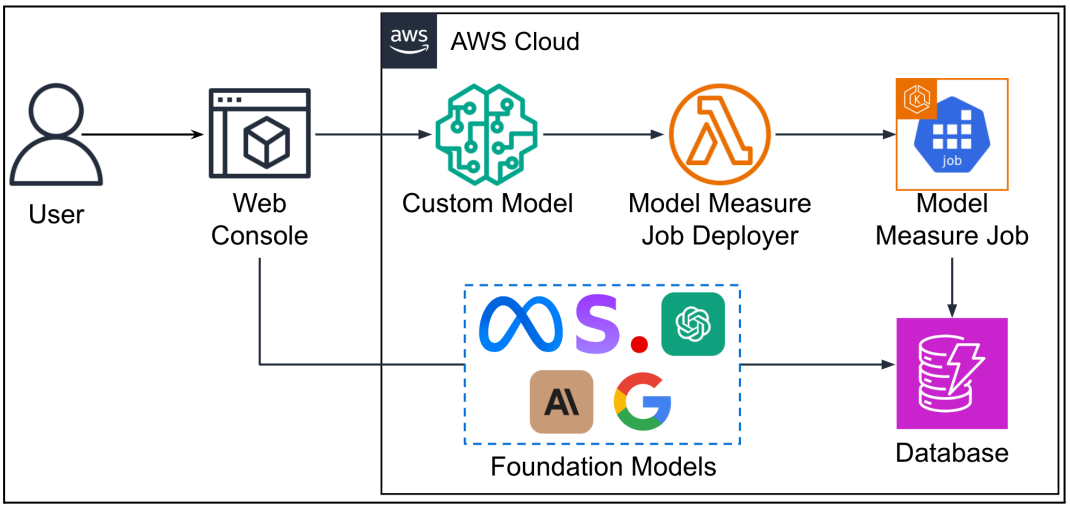






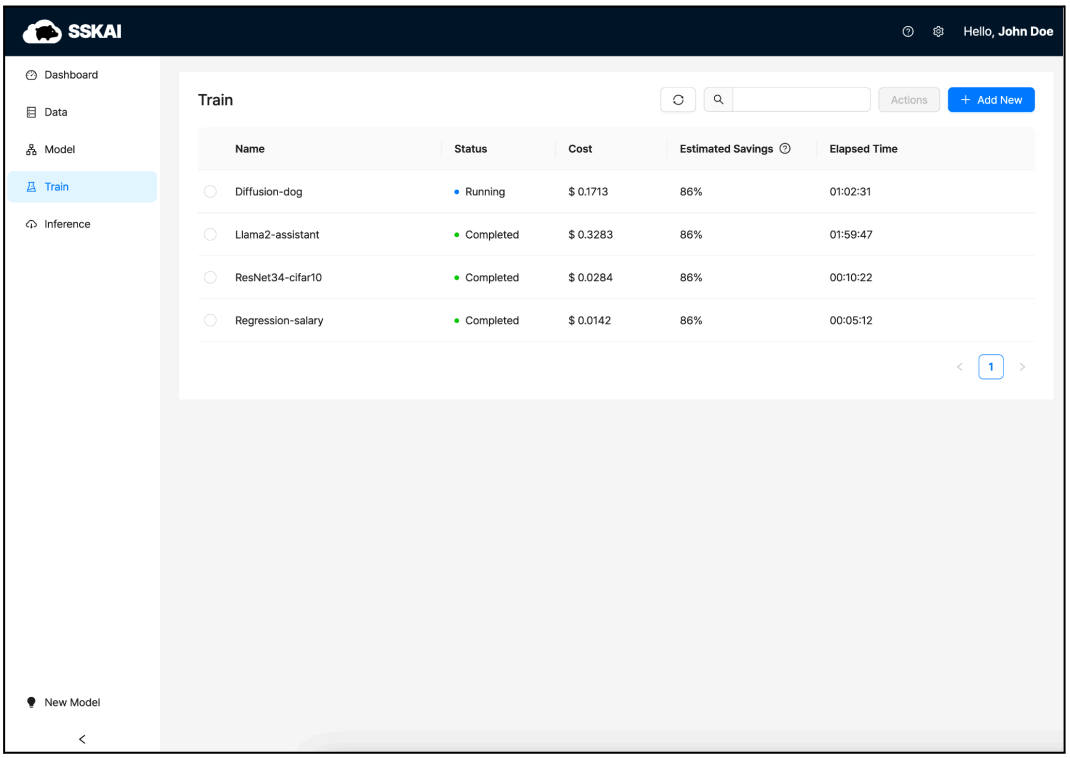


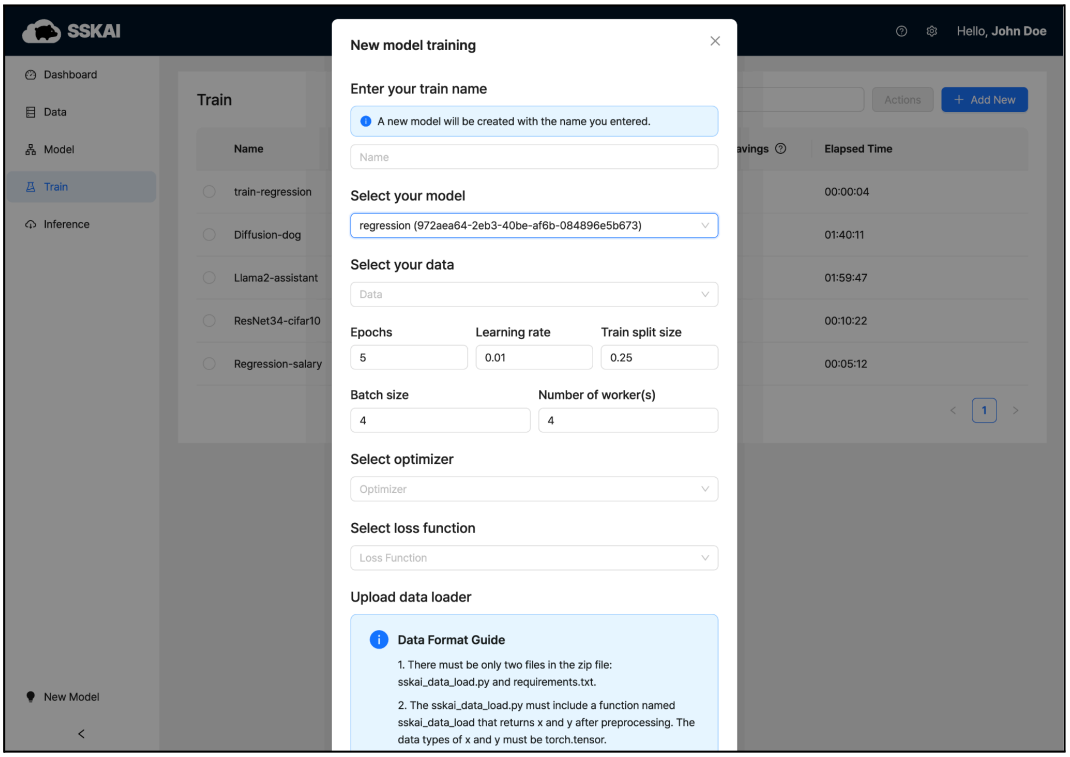
사용자는 Model 페이지를 통해 학습 또는 배포하고자 하는 모델 구조 및 파라미터를 업로드할 수 있게된다. 또한, 사용자 정의 모델 뿐 만 아니라 사전에 학습된 이미지나 텍스트 생성 모델같은 파운데이션 모델도 배포할 수 있다. 사용자가 사용자 정의 모델을 업로드하는 경우, 추후 최적화된 컴퓨팅 자원 선출을 위해 모델 측정이 이루어진다. 이 때 본 플랫폼에서는 사용자로부터 입력 형식과 입력 범위와 같은 모델 정보를 추가적으로 입력받아 해당 모델의 학습이나 추론에 필요한 컴퓨팅 자원이 얼마인지 실제로 측정하여 데이터베이스에 기록되게 한다. 이와 같은 절차는 아래 그림과 같다.

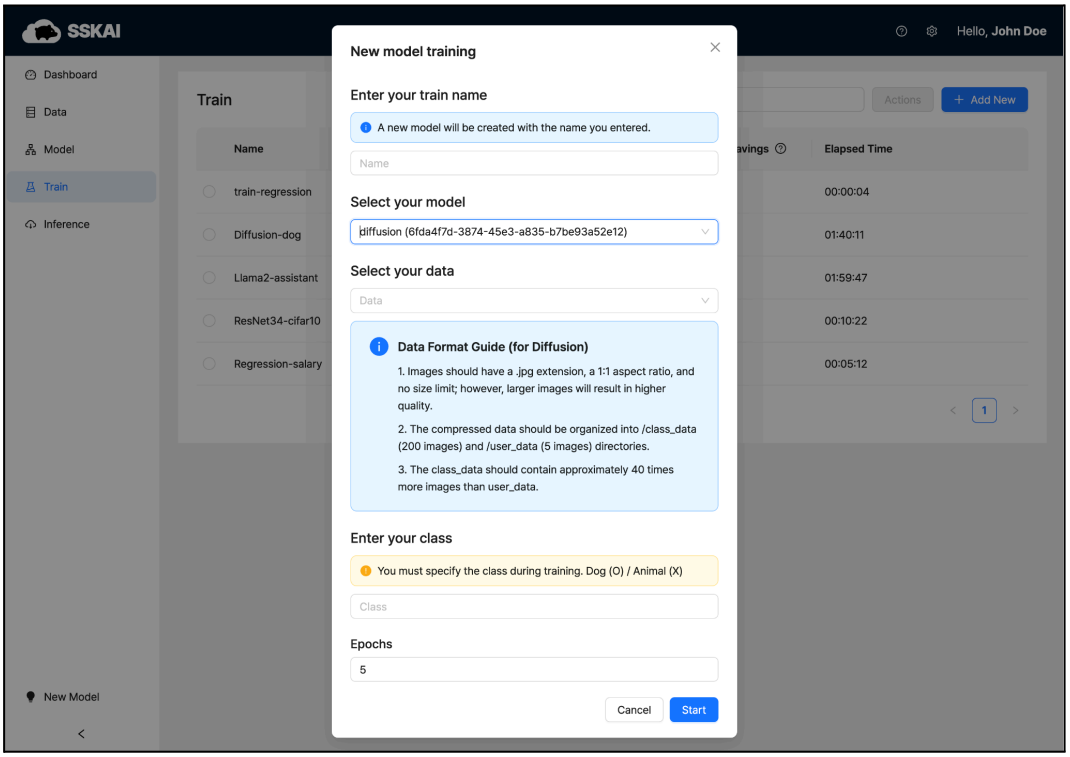


사용자 정의 모델은 ZIP 파일 형태로 업로드를 해야하며, 플랫폼에서는 추가적으로 입력된 정보를 사용하여 적합한 학습/추론 아키텍처 선택을 도와줄 수 있도록 모델을 측정한다. 모든 값 입력 및 모델 업로드가 완료되면, 위 그림과 같이 플랫폼 컴퓨팅 클러스터에 측정 요청을 보내 모델을 학습/추론하는데 필요한 성능 지표를 기록하고 이를 추후에 활용할 수 있도록 데이터베이스에 기록한다.

3.2.1.4. Train 페이지





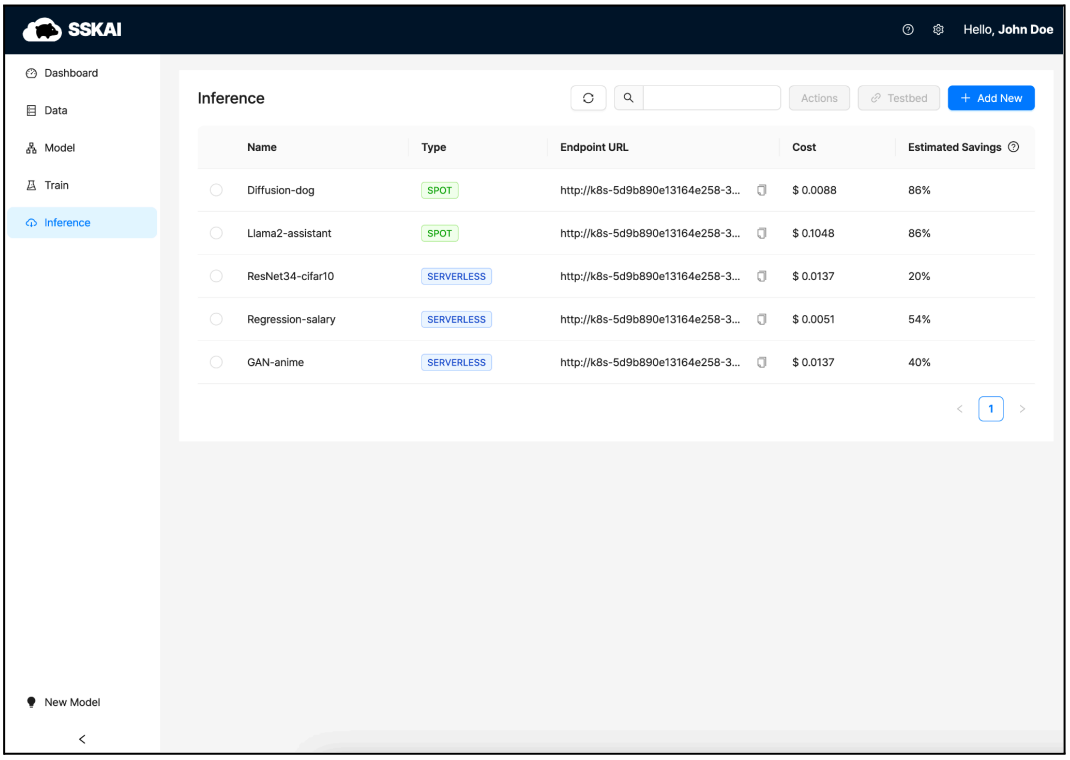


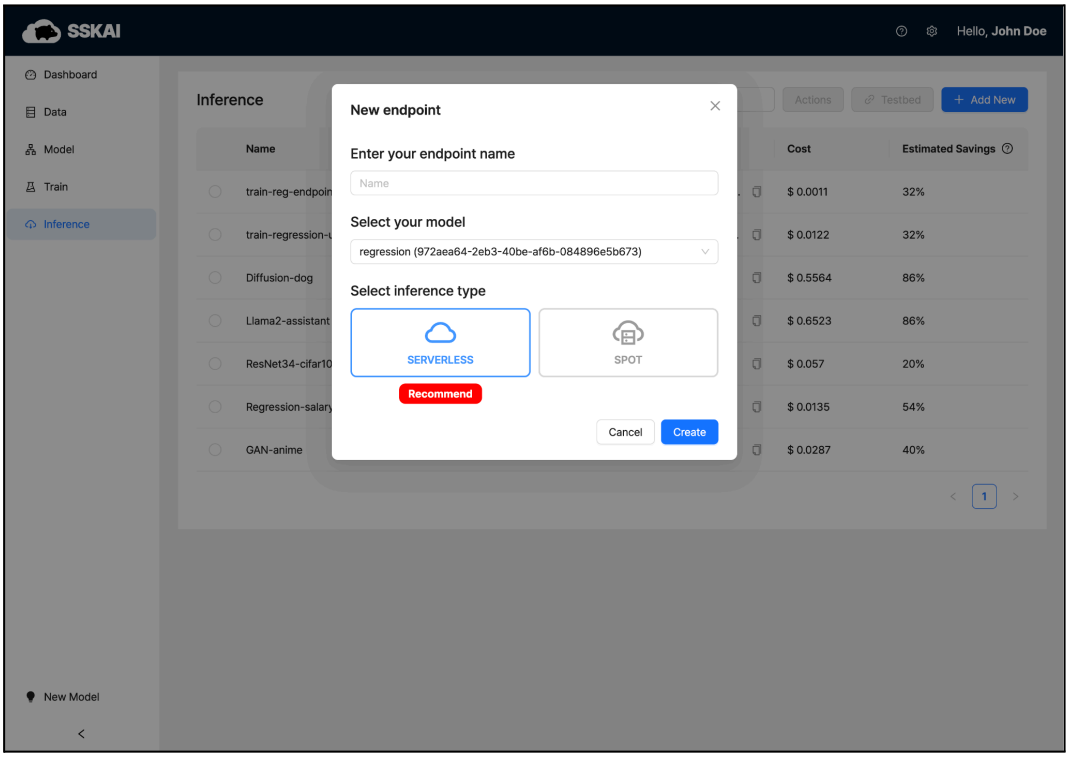
사용자는 Train 페이지를 통해 Model 및 Data 페이지에서 업로드한 정보를 바탕으로 학습 작업을 배포할 수 있게 된다. 이 때, 사용자 정의 모델을 학습하고자 하는 경우에는 사용자가 사용할 옵티마이저와 손실 함수를 지정해주는 절차가 필요하며, 데이터에서 학습 및 테스트 데이터 분할 비율, 에포크, Learning Rate 값 및 옵티마이저와 손실 함수의 종류를 지정할 수 있다. 또한, 학습 과정에서 데이터를 불러오기 위해 사용자는 x, y 데이터를 불러오는 Python 함수 코드 및 이에 필요한 라이브러리 목록(requirements.txt)을 압축해서 제공해주어야 한다.

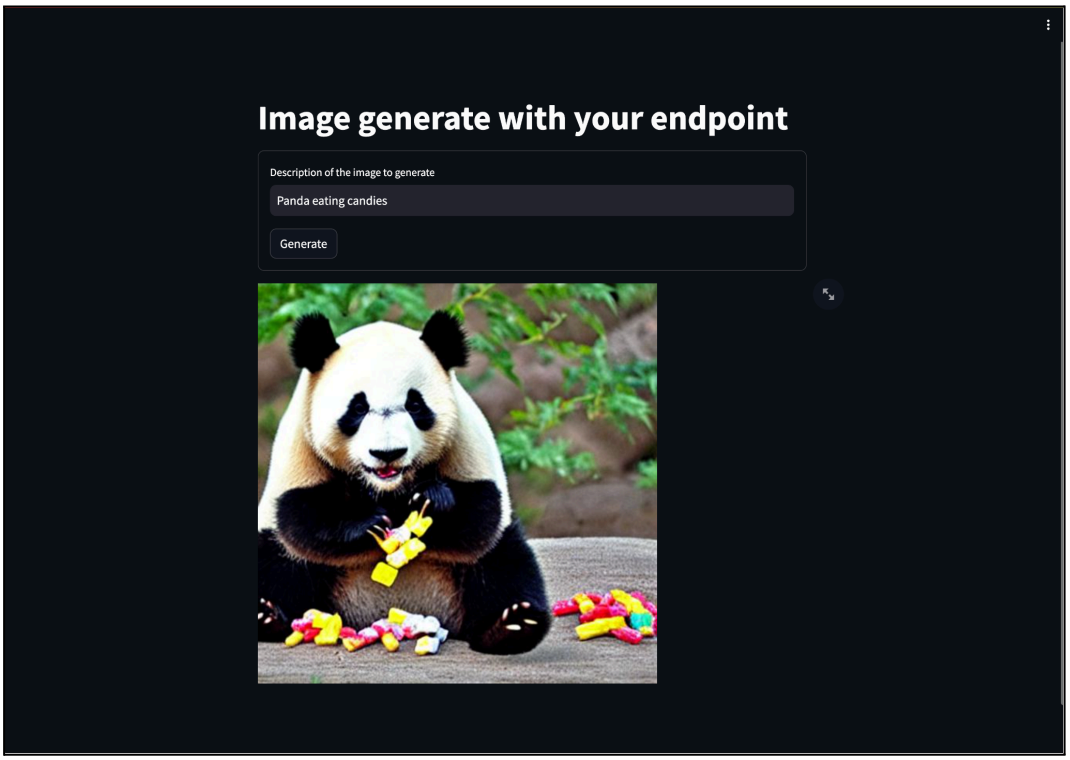
학습이 진행 됨에 따라 Train 메인 화면에서는 현재 학습의 진행 상태를 초기화(Pending), 진행중(Running), 완료(Complete) 3단계로 확인할 수 있게되며, 학습에 사용된 시간, 비용 정보를 확인할 수 있게 된다.

학습이 완료된 모델은 Model 페이지에서 확인할 수 있게되며 배포에 이용할 수 있게 된다.

3.2.1.5. Inference 페이지







사용자는 Inference 페이지를 통해 직접 업로드한 모델이나, 학습을 통해 생성된 모델을 배포할 수 있게 된다. Inference 페이지에서는 사용자가 배포한 모델의 엔드포인트 주소와 배포 시간 동안 사용된 요금 및 절약된 가격 비율을 확인할 수 있게된다.

사용자가 사용자 정의 모델을 배포한 경우에는 모델 엔드포인트 주소에 추론에 필요한 텐서를 pickle과 같은 직렬화 도구로 POST로 전송하면 사용자는 직렬화된 추론 결과 데이터를 수신할 수 있게된다.

사용자는 사용자 정의 모델을 배포할 컴퓨팅 환경을 서버리스와 스팟 두가지 중 하나를 선택할 수 있다. 플랫폼에서는 Model 페이지에서 업로드할 때 입력받은 입력 형태 등의 정보를 기반으로 사용자에게 서버리스 또는 스팟 컴퓨팅 자원 중 하나를 추천해주는 기능을 제공한다.

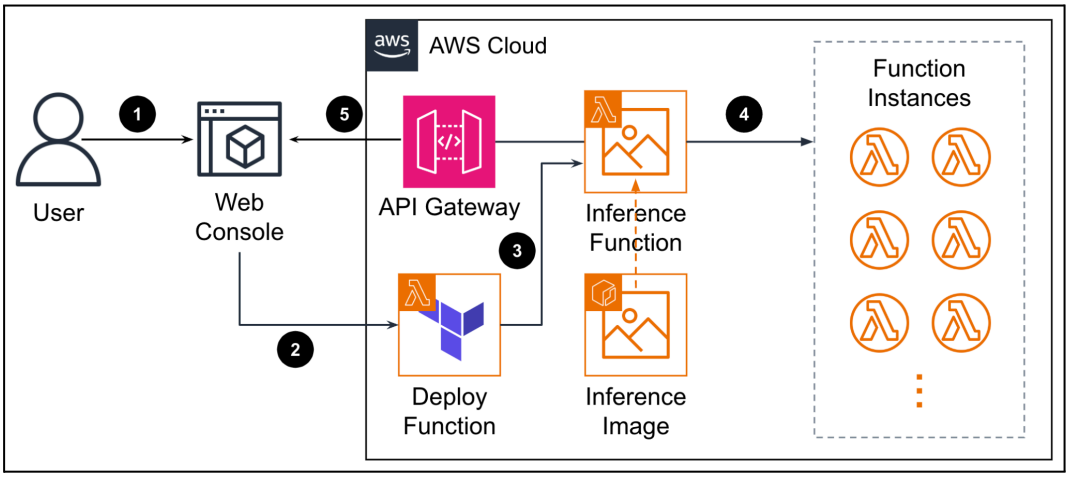
만약 사용자가 생성한 파운데이션 모델이나 학습한 파운데이션 모델을 배포하는 경우에는 프롬프트를 JSON 형식으로 전송하여 결과를 JSON 형식으로 받아볼 수 있다.

또한, 플랫폼에서는 테스트 페이지를 추가적으로 배포해 Streamlit을 통해 테스트할 수 있는 환경을 제공한다. 일반적으로 생성형 AI 모델 개발자들은 Hyper Parameter Tuning 결과 확인을 위해 모델을 배포하여 테스트를 수행 후, Production에 실제로 배포하는 과정을 거친다. 이를 위해 기존의 많은 개발자들은 테스트하는 코드를 별도로 작성하거나, 프론트엔드 개발팀과 같은 부서에서

테스트를 하기 위한 웹 페이지를 제작한 것을 사용할 필요가 있어 실제로 원하는건 테스트이지만 여러가지 병목현상이 생기는 경우가 잦았다. 이 프로젝트에서는 머신러닝 개발자가 빠르게 테스트를 수행할 수 있도록 도움을 주기 위해 여러 모델에 대응하는 테스트 페이지를 제작하여 이를 해소하고자 하였다. 이를 통해 사용자는 파운데이션 모델의 기본적인 결과나 학습된 결과를 손쉽게 확인할 수 있게된다.

3.2.2 서버리스 환경에서 추론을 위한 아키텍처

사용자가 개발 및 학습을 완료한 모델을 배포할 때에는 서버리스 환경과 스팟 인스턴스 환경 2가지를 모두 고려할 수 있다. 그 중에서 서버리스 환경의 경우, 사용자의 모델에 필요한 성능이 많지 않아 CPU 만으로도 추론이 충분한 경우에 활용할 수 있다. 서버리스 컴퓨팅은 실제로 실행된 시간에 대해서만 클라우드 제공업체에서 과금을 하기에 이 프로젝트를 실제 상용 서비스로 운영할때 고객에게 과금하는 비용을 줄일 뿐만아니라 이 플랫폼을 운영할 때에도 비용을 줄일 수 있어 매우 효과적이다.



이 프로젝트에서 사용된 서버리스를 통한 추론 아키텍처는 위 그림과 같다.

사용자는 웹 서비스를 통해 추론 엔드포인트 배포를 요청할 수 있다. 서버리스 아키텍처에 배포하는 경우에는 배포를 자동화하는 함수를 실행하여 사용자의 모델 추론 요청을 처리할 수 있는 서버리스 함수를 구축하게 된다. 구축이 완료되면, 사용자는 웹 콘솔을 통해 API에 접근할 수 있는 엔드포인트 주소를 받아 해당 주소로 추론 요청을 전송할 수 있게 된다.

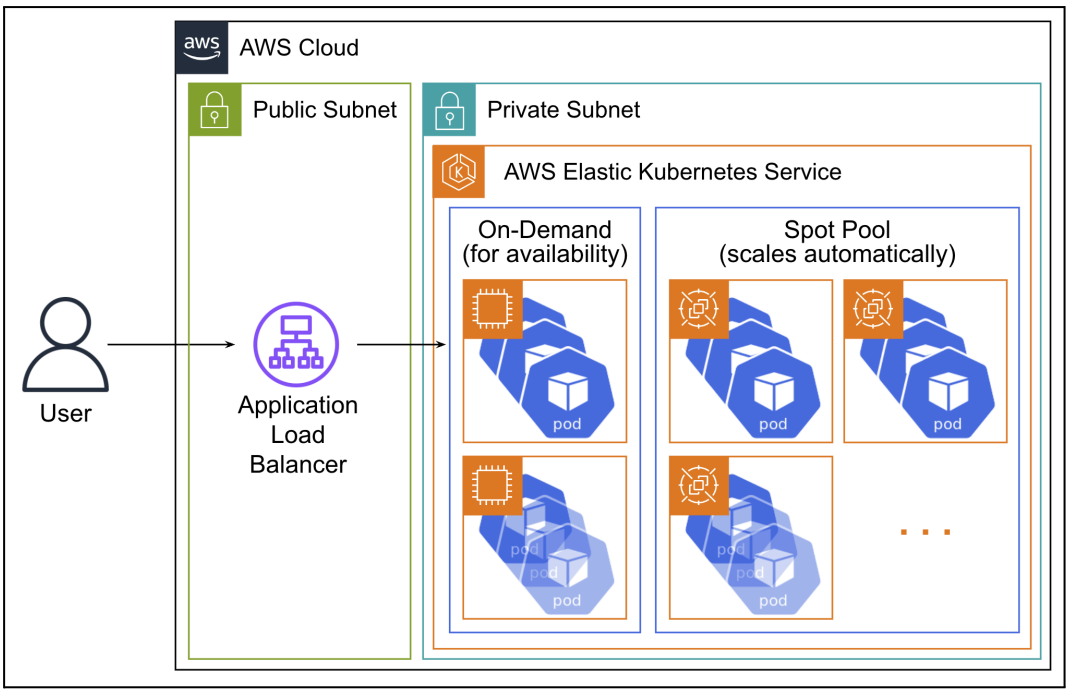
서버리스 컴퓨팅에서 추론 시 가장 고려해야할 문제는 부족한 성능 문제와 Cold Start이다.

서버리스 컴퓨팅은 최대 10GB의 Memory를 사용할 수 있으므로 최대 10GB의 메모리만 사용가능

하여 모델의 크기가 제한된다. 또한, 현재 제공되는 대부분 서버리스 컴퓨팅 서비스는 GPU와 같은 가속기를 사용할 수 없어 성능 측면에서 불리할 수 있다. 따라서 높은 성능이 요구되는 경우에는 GPU와 같은 가속기가 적용된 일반적인 서버를 사용하는 컴퓨팅 서비스를 이용하여 구현하도록 하였다. 그러나, 이러한 점은 사용자가 판단하기 어려운 내용일 수 있으므로 이 플랫폼에서는 모델 업로드 시, 어떤 아키텍처를 선택해야하는지 추천해주는 것을 통해 이러한 제한점을 해결하였다. Cold Start문제는 지속적으로 초기화하는 방식으로 해결할 수 있어 Linear Regression, 단순한 Image Classification Model 등에 효과적으로 활용할 수 있다.

3.2.3 스팟 과금 환경에서 추론을 위한 아키텍처

모델에서 필요로 하는 컴퓨팅 자원의 성능이 낮다면 서버리스 컴퓨팅을 활용하는 것이 운영하는 측면에서 경제적으로 이점을 가져갈 수 있으나, 만약 텍스트 생성이나 이미지 생성과 같은 GenAI Model을 사용하는 경우, 모델의 크기가 기본적으로 최소 10GB 이상으로 구성되어 있다. 그 예시로 매개변수가 60억인 GPT-J-6B(24.2GB)이나 매개변수가 70억인 LLama-2-7b(13.5GB)가 존재한다. 이 경우에는 물리적으로 서버리스 컴퓨팅을 활용할 수 없어 컴퓨팅 서비스를 비용 효과적인 과금 정책인 스팟 과금 정책을 통해 이용하는 것으로 비용을 줄이면서 성능 효율적인 컴퓨팅 자원을 이용할 수 있게 된다.



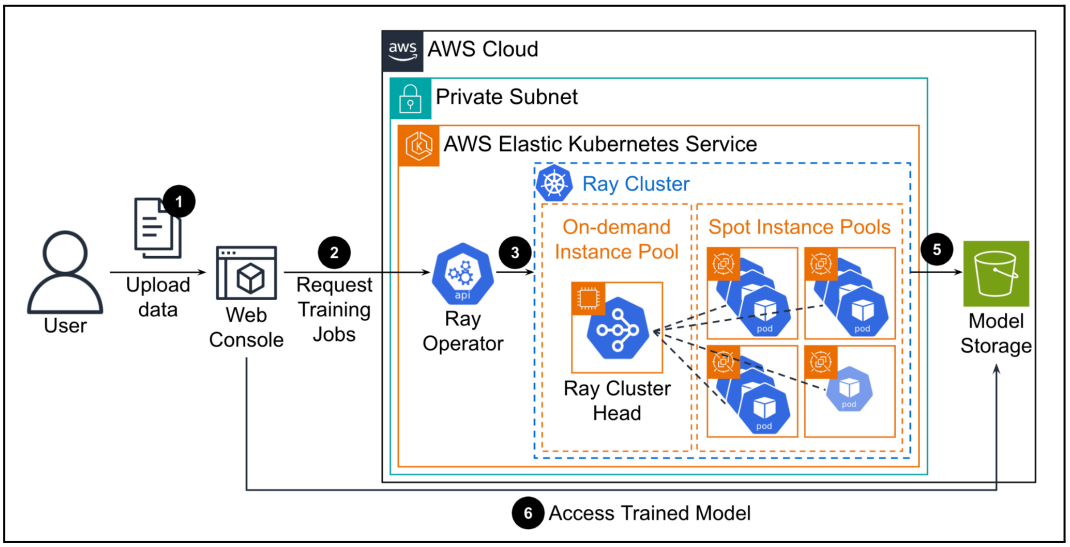
추론의 경우, 요청 처리 중 만약 스팟 인스턴스 종료 알림이 들어오면 그를 대체하는 스팟

인스턴스를 새롭게 시작하여 서비스 가용성을 확보할 수 있다. 또한, 온디맨드 인스턴스를 최소 서비스 가용성 유지목적으로 최소한으로 배치시키는 방법을 활용할 수 있다.

3.2.4 스팟 과금 환경에서 분산 학습을 위한 아키텍처

머신러닝 학습의 경우, 추론과 다르게 학습에 필요한 대규모 데이터를 메모리에 추가로 로드해야할 필요가 있으며, 학습에 필요한 추가적인 연산을 수행하기에 일반적으로 GPU, TPU와 같은 가속기가 장착된 하드웨어 환경에서 학습을 진행한다.

따라서, 별도의 가속기가 장착되어 있지 않은 서버리스 컴퓨팅 환경에서는 학습을 진행하기 어려운 측면이 있어 학습은 스팟 과금 정책을 적용한 컴퓨팅 자원을 사용하도록 구성한다.



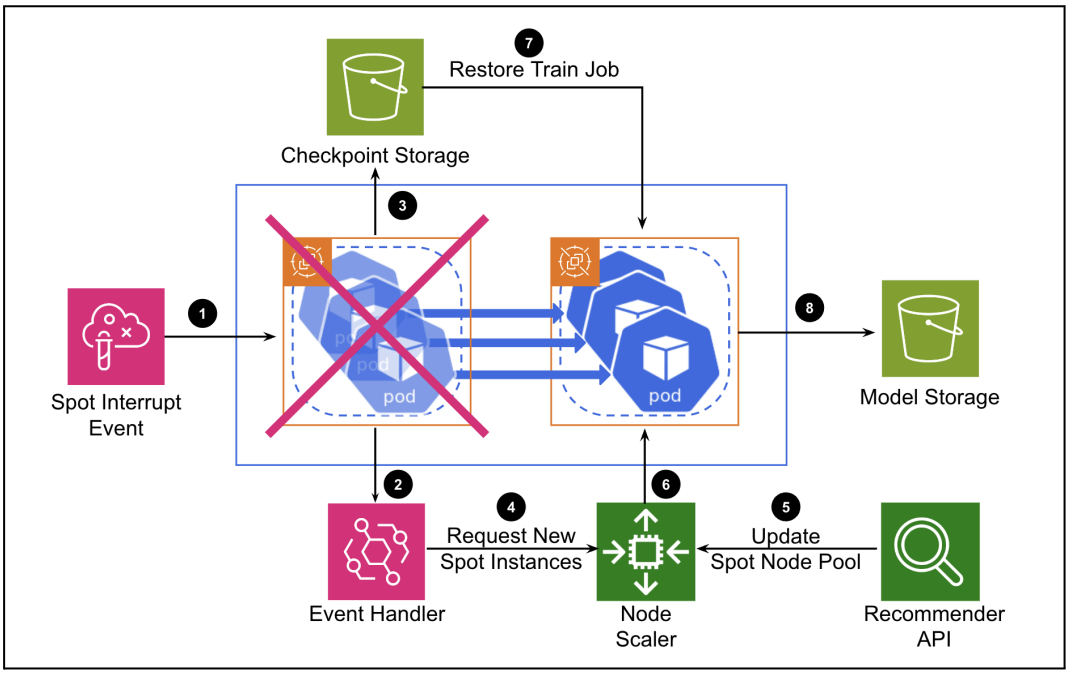
이 때, 클라우드 제공업체가 제공하는 방대한 인프라를 활용하기 위해 분산 학습 기법을 적용하며, 이 중 이 프로젝트의 목표로는 Data Parallelism을 통해 효율적인 분산 학습을 진행하도록 하였다.

먼저 사용자는 웹 서비스를 통해 모델과 학습용 데이터 셋을 업로드하고, 웹 콘솔에 학습 요청을 보내게 된다. 그러면 웹 콘솔은 플랫폼의 컴퓨팅 클러스터에 학습 Job 생성 요청을 보내게 된다. 요청을 받은 컴퓨팅 클러스터는 학습을 위한 분산 학습 Framework인 Ray 클러스터를 생성하게 되며 학습을 진행하게 된다. 학습이 완료되면 생성된 학습 클러스터는 컴퓨팅 클러스터에서 제거되며 모델 저장소에 학습이 완료된 모델이 저장된다. 사용자는 웹 서비스의 Model 페이지를 통해 해당 모델에 접근할 수 있게되며 이 Inference 페이지에서 학습된 모델을 배포할 수도 있게 된다.

3.2.5 스팟 과금 환경에서 모델 학습 시 중단이 될 수 있음을 고려한 기법 적용

추론은 Stateless하게 진행되기에 추론 요청을 하던 중 스팟 자원 회수 알림이 들어오더라도 현재 사용자의 요청을 처리하고 이후 요청은 처리하지 않게하면서 새로운 자원을 시작하면 되지만, Stateful하게 진행되는 학습의 경우 학습 도중 스팟 자원 회수가 발생하게 되면 진행 내용이 손실될 가능성이 존재한다. 3.2.6에서 소개한 알고리즘을 통해 안정적인 스팟 컴퓨팅 자원을 선출하더라도 실시간으로 안정성 및 가격 정보가 변동되기에 항상 스팟 자원 회수를 대비할 수 있어야한다.

따라서, 학습 작업에서 체크포인트와 같은 방법으로 현재 진행 상황을 지속적으로 혹은 스팟 자원 회수 요청 알림이 왔을 때 백업할 필요가 있다. 이때 두가지 체크포인트 기법을 활용할 수 있다. 첫 번째 방법은, 프로세스 단위에서의 체크포인트이다. 이때 사실상 표준이라고 여겨지는 CRIU (Checkpoint Restore In Userspace) [5] 도구를 사용하여 프로세스 단위의 체크포인트를 수행하고 새로운 스팟 컴퓨팅 자원에서 리스토어를 하는 것으로 작업 진행 상황의 손실없이 이어서 진행할 수 있게 된다. 두 번째 방법은 머신러닝 프레임워크에서 지원하는 체크포인트 기법을 활용하는 것이다. 이 기법을 활용하여 학습 중 갱신되는 버퍼와 매개변수들을 포함하는 옵티마이저와 중단 시점의 epoch, training loss등을 기록하여 스팟 자원 회수가 되더라도 중단된 시점부터 이어서 작업을 수행할 수 있게된다 [6]. 두 가지 기법 모두 체크포인트를 파일로서 저장하게 되며, 이 파일들은 S3나 NFS와 같은 원격 네트워크 스토리지에 저장되게 된다.



이

프로젝트에서는 활용할 수 있는 2가지 체크포인트 기법 중 모델 체크포인트 기법을 활용하였으며, 사용자가 요청한 학습 수행 중, 일부 작업 노드에서 스팟 인터럽트가 발생하는 것을 대비하기 위해 매 에포크마다 모델 체크포인트를 수행하도록 하였다. 만약 중간에 일부 작업 노드에서 스팟 인터럽트가 발생하는 경우, 그 즉시 노드 스케일러가 새롭게 스팟 인스턴스를 요청하도록 하였으며 새롭게 생성된 작업 노드는 가장 최근 저장된 체크포인트를 불러와서 작업을 이어서 수행할 수 있도록 하였다. 이를 통해 스팟 인스턴스의 중단될 수 있다는 리스크를 제거하면서, 비용 및 성능 최적화된 컴퓨팅 자원을 사용할 수 있도록 구현하였다.

3.2.6 각 모델에 맞는 최적의 비용 및 성능을 가진 컴퓨팅 자원을 선출하는 알고리즘 및 배포 플랫폼 추천 기능

학습이나 추론을 진행할 때, 사용자가 사용하고자 하는 모델과 기능에 따라 적합한 컴퓨팅 자원을 선출해야할 필요가 있다. 예를 들어, 추론을 진행하고자 할 때 모델에서 요구하는 성능이 높지 않고 모델의 크기 작은 경우에는 서버리스 컴퓨팅을 활용한 추론 시스템 구축이 필요하며, 서버리스 컴퓨팅에서 사용할 메모리의 크기의 경우에는 학습 시 사용된 CPU / Memory 사용량을 통해 구하거나 추론 시뮬레이션을 통해 필요한 최소 메모리의 크기를 확인할 수 있다. 이는 스팟 과금 환경에서도 동일하게 적용할 수 있다.

그러나, 메모리의 크기만 설정하면 되는 서버리스 컴퓨팅 환경과 다르게 스팟 과금 환경을 사용하기 위해서는 컴퓨팅 자원 (서버)의 성능, 컴퓨팅 자원의 당시 요금, 컴퓨팅 자원의 스팟 안정성 지표등을 검토할 필요가 있다. 따라서 아래와 같은 절차를 거쳐 최적의 컴퓨팅 자원 (서버) 유형을 선출하게 된다. 최적의 컴퓨팅 자원 유형 선출 알고리즘은 아래 그림과 같다.

Algorithm 1 Instance Elect Algorithm

Output: Set of instance which optimized for cost S

Obtain total instance $I = \{i_1, \ldots, i_n\}$
Compute set of GPU memory group $R = \{r_1, \ldots, r_n\}$
Initialize dictionary $D = \{\}$
Set $\alpha = 0.2$
for each lowerbound resources r_j **do**
    $S = \emptyset$
    Get I_j which set of instance that satisfy r_j
    Get B_j which set of Benchmark of I_j ▷ will be binded to (i, b)
    Normalize each cost of i where in I_j
    Normalize each benchmark of b where in B_j
    Compute $score = (1 - \alpha) \times i + \alpha \times b$
    Sort I_j by descending order of $score$
    $S =$ Top 5 instance in I_j
    $D[r_j] = S$
end for
return D

첫 번째로, 사용 할 GPU 스팟 인스턴스들이 제공가능한 GPU 메모리를 그룹화하여 묶는다. 각각의 그룹은 우리의 플랫폼 환경에서 노드풀을 구성하게 될 것이며, 모델 측정 알고리즘을 통해서 자동으로 노드풀에 사용자의 파드가 할당되게 된다. 두 번째로, 그룹 내부의 각각의 인스턴스마다 미리 측정된 벤치마크를 가져온다. 이 벤치마크를 해당 스팟 인스턴스의 성능 지표라고 가정한다. 세 번째로, 스팟 인스턴스의 가격과 미리 측정된 벤치마크를 선형 접합 연산을 통하여 최종 점수를 계산한다. 선형 접합 연산에서의 비율인 알파값은 하이퍼 파라미터 값이며, 우리는 경험을 통하여 0.2가 적절하다고 판단하였다. 네 번째로, 계산된 각 인스턴스의 최종 점수를 기준으로 내림차순으로 정렬하여 상위 5개의 인스턴스들을 추출한다. 마지막으로, 추출된 그룹별 상위 5개의 인스턴스들을 딕셔너리에 넣어서 최종 결과를 반환한다. 해당 알고리즘에 의해 선택된 인스턴스들은 EKS 환경에서 자동으로 노드를 프로비저닝 해 주는 도구인 Karpenter의 노드 풀 환경으로 구성되게 된다.

각 모델별로 최적의 컴퓨팅 자원을 할당 해 주기 위해서는 해당 모델이 어느 정도의 자원을 사용하는 지 측정하는 기능이 필요하다. 본 플랫폼에서는 각 모델을 측정하는 모델 측정 기능을 개발하여 해당 모델이 필요한 자원량을 추정하고 배포 할 컴퓨팅 플랫폼을 결정하게 된다. 해당 기능을 통하여 모델이 충분히 작은 모델이라고 판단될 경우 서버리스로 배포 플랫폼이 결정되며, 아닐 경우 GPU가 장착된 스팟 인스턴스를 사용한다.

3.3 평가

3.3.1 기존 솔루션 대비 비용 효율성 평가

이 프로젝트에서 제안된 여러 비용 절감 기법 활용을 통해서 얼마나 실제 비용 절약이 이루어졌는지를 평가할 필요가 있다. 따라서 Amazon Web Services, Microsoft Azure, Google Cloud Platform의 머신러닝과 생성형 AI 서비스들과의 비용 비교를 진행하였다.

비교 대상 서비스로 Amazon Web Services의 경우 인공지능 서비스로 Amazon SageMaker, 생성형 AI 서비스로 Amazon Bedrock을 선정하였다. Microsoft Azure의 경우에는 Azure Machine Learning을 선정하였으며, Google Cloud Platform의 경우에는 Vertex AI를 선정하였다.

Provider	AWS				Azure			GCP	
Service	SageMaker			Bedrock	Machine Learning			Vertex AI	**Proposed**
Price Plan	-	S/1y	S/3y	-	-	S/1y	S/3y	-	
Per Hour (USD)	37.7	27.8	19.2	57.2	32.8	28.6	17.2	33.8	**13.1**
Daily (USD)	904.8	667.2	460.8	1372.8	787.2	686.4	412.8	811.2	**316.8**

- : On-demand, S : Savings plan, y : year

위 표는 각 솔루션에서 자연어를 생성하는 생성형 AI 모델인 Meta의 Llama-2-13B 모델을 배포하면서 시간 당 8,730,000개의 입력 토큰과 1,166,400개의 출력 토큰을 처리한다고 가정하였을 때, 시간 당 추론 비용을 시뮬레이션한 결과를 비교한 표이다.

위 결과를 보았을 때, 이 프로젝트가 기존 타 클라우드 제공업체의 인공지능 솔루션에 비해 최대 77%의 가격이 절감됨을 확인하였다.

3.3.2 체크포인트로 인한 오버헤드 평가

스팟 환경에서 분산 학습시에 간헐적으로 발생하는 컴퓨팅 자원 회수로 인한 작업 취소에 대비하기 위해 매 에포크마다 모델 체크포인트 작업을 수행하는 것이 어느정도의 오버헤드가 소요되는지 평가할 필요가 있다. 실험을 위해 초급 영어로 작성된 위키피디아 백과사전인 Simple Wikipedia

데이터셋과, GPT-2 모델을 사용하였으며, 사용한 컴퓨팅 자원은 Nvidia Tesla T4 GPU가 장착된 AWS의 g4dn.xlarge 인스턴스 유형을 사용하였다. 실험을 위한 에포크 횟수는 10을 사용하였다.

실험 결과로 에포크 당 약 9분 50 가량 학습이 소요되었으며, 추가적으로 에포크당 체크포인트 작업을 수행하는데 평균 약 3.02초 소요되는 것을 확인하였다. 이를 통해 체크포인트 작업이 많은 오버헤드를 차지하지 않음을 확인하였다.

3.4 현실적 제한 요소 및 해결방안

스팟 인스턴스를 활용하는 이 프로젝트에서는 스팟 인터럽트를 어떻게 대응하는지가 가장 핵심적인 요소였다. 따라서 분산 학습 시나리오에서는 매 에포크마다 모델 체크포인트를 수행하는 것을 통해 스팟 인터럽트에 대응하는것으로 해결하였다. 그러나 모델을 배포하는 추론 엔드포인트 시나리오에서는 현재 Stateless하다고 가정하고 있으나, 기술의 발전에 따라서 Stateful하게 추론 작업이 진행될 가능성을 배제할 수는 없다. 따라서 추후에는 모델 체크포인트 뿐만 아니라 추론 엔드포인트서버 마이그레이션도 대응하기 위해 컨테이너 마이그레이션 기술을 활용하면 이를 해결할 수 있다.

또한 현재에는 특정 클라우드 제공업체와 해당 업체의 특정 지역에만 배포되는것을 전제로한다. 모델 배포의 경우에는 사용자가 위치한 지역에 가까워야 지연시간이 줄어들기 때문에 상관 없으나, 학습에 경우에는 지연시간이 중요하지 않기 때문에 최대한 저렴하고 성능이 좋은 컴퓨팅 자원을 사용하는 것이 합리적이다. 따라서 추후에 발전시키기 위해서는 다른 지역의 컴퓨팅 자원 가격 정보도 수집하여 모든 지역에서의 자원 중 가장 자원을 선택하여 그를 컴퓨팅 클러스터에 가입할 수 있도록 할 수 있으며, 더 나아가 타 클라우드 제공업체의 컴퓨팅 자원 가격 정보까지 분석하여 멀티 클라우드 환경에서 가장 저렴하고 성능이 좋은 컴퓨팅 자원을 사용하도록 구성할 수 있다.

3.5 결과물 목록

프로젝트 진행을 통한 산출된 결과물 목록은 아래의 표와 같다.

모듈	코드 경로 (Github Repository)	기술문서
프론트엔드	frontend/sskai-console/src/*	무

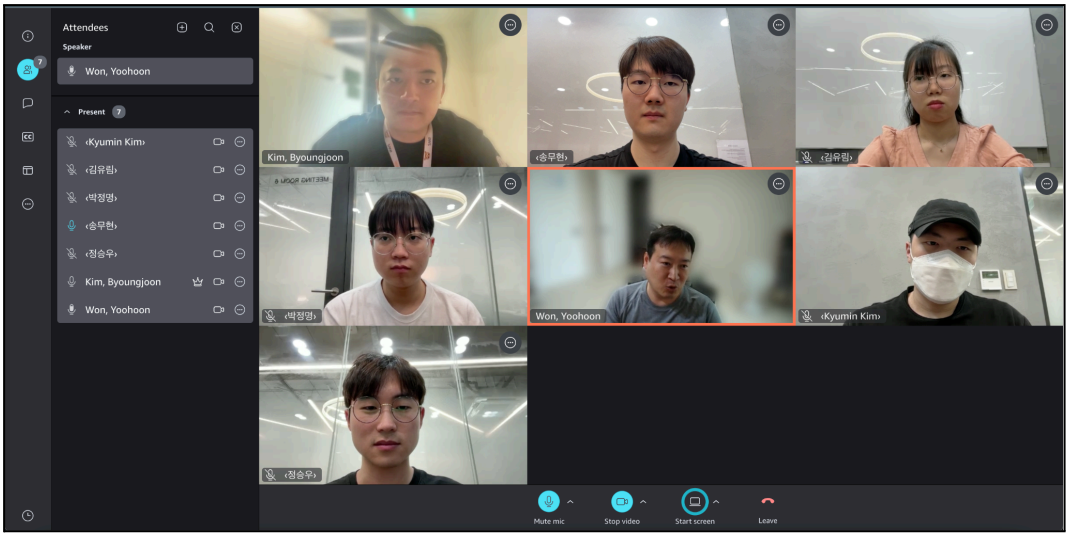
백엔드 (DB API)	backend/sskai-cost-calculate/index.mjs backend/sskai-ddb-data-api/index.mjs backend/sskai-ddb-inferences-api/index.mjs backend/sskai-ddb-logs-api/index.mjs backend/sskai-ddb-models-api/index.mjs backend/sskai-ddb-trains-api/index.mjs backend/sskai-ddb-users-api/index.mjs	무
백엔드 (S3 API)	backend/sskai-s3-multipart-presigned-url/index.mjs backend/sskai-s3-presigned-url-api/index.mjs	무
쿠버네티스 클러스터 배포 자동화	IaC/kubernetes_cluster/*	무
서버리스 API 배포 자동화	IaC/serverless_api_template/*	무
서버리스 추론 배포 자동화	IaC/serverless_inference/*	무
DB API 배포 자동화	automation/deploy_db_api/*	무
Frontend 배포 자동화	automation/deploy_s3_web/*	무
Streamlit (Testbed) 배포 자동화 API	automation/dploy_streamlit/*	무
학습 작업 배포 자동화 API	automation/deploy_train/*	무
서버리스 추론 배포 자동화 API	automation/serverless_inference_deploy/*	무

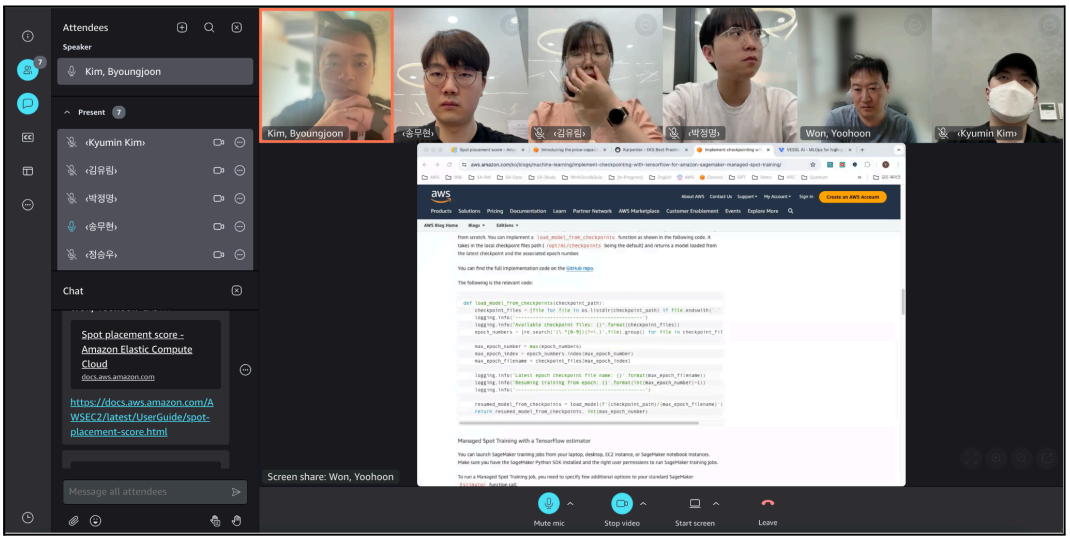
쿠버네티스 추론 배포 자동화 API	automation/kubernetes_inference_deploy/*	무
모델 평가 배포 자동화 API	automation/kubernetes_model_profiler_deploy/*	무
Llama2 모델 배포 자동화 API	automation/llama_inference_deploy/*	무
Llama2 모델 학습 작업 배포 자동화 API	automation/llama_train_deploy/*	무
Diffusion 모델 배포 자동화 API	automation/diffusion_inference_deploy/*	무
Diffusion 모델 학습 작업 배포 자동화 API	automation/diffusion_train_deploy/*	무
Karpenter 노드 풀 배포 자동화 API	automation/karpenter_node_pool_deploy/*	무
GAN 모델 예시	example/GAN_Anime/*	무
Regression 모델 예시	example/regression/*	무
ResNet34 모델 예시	example/ResNet34_CIFAR-10/*	무
Llama2 모델 예시	foundation_model/llama2_7b_chat_hf/*	무
Diffusion 모델 예시	foundation_model/stable_diffusion/*	무

서버리스 추론 애플리케이션 템플릿	inference/template_code/lambda_app.py inference/template_code/requirements_lambda.txt inference/template_code/Dockerfile.lambda	무
쿠버네티스 추론 애플리케이션 템플릿	inference/template_code/kubernetes_app.py inference/template_code/requirements_kubernetes_gpu.txt inference/template_code/Dockerfile.kubernetes_gpu	무
Llama2 추론 애플리케이션 템플릿	inference/template_code/llama/*	무
Diffusion 추론 애플리케이션 템플릿	inference/template_code/diffusion/*	무
모델 평가 애플리케이션 템플릿	model_profile/template_code/*	무
최적의 컴퓨팅 자원 선출 API 배포 자동화	recommend/family_recommend/IaC/*	무
최적의 컴퓨팅 자원 선출 API 애플리케이션	recommend/family_recommend/family/*	무
CPU를 사용한 분산학습 컨테이너 템플릿	train/ray/cpu/*	무
GPU를 사용한 분산학습	train/ray/gpu/*	무

컨테이너 템플릿		
컨테이너 빌드 자동화 도구	container_build.sh delete_container.sh	무
전체 인프라 배포 코드	main.tf var.tf	무
프로젝트 배포 도구	skkai_execute.py	무
소개 페이지	README.md index.md	무

3.6 산학 협력 내용





산학 협력으로 Amazon Web Services Korea의 김병준 이사님 및 원유훈 솔루션 아키텍트님과 주기적으로 심도있는 온라인 화상회의 Q&A를 진행하였다.

진행한 내용으로는 이 프로젝트에 대한 내용 소개 및 이에 대한 조언을 해주셨으며, 기존 Amazon Web Services에서 현재로서는 되지 않아 추가적으로 구현이 된다면 좋을 만한 기능들을 소개해주셔서 많은 도움이 되었다.

추가적으로, AWS 교육 관련 Slack에 초대 받아 AWS 관련 최신 뉴스, AWS에서 주최하는 세미나 정보를 빠르게 알 수 있게되어 도움이 많이 되었으며, 프로젝트 진행을 위한 슬랙 채널을 개설하여 지속적인 Q&A를 진행하여 많은 도움을 받았다.

3.7 기대효과 및 활용방안

이 프로젝트를 사용하여 머신러닝 파이프라인을 운영함으로써 기존 클라우드 제공업체의 인공지능 서비스 대비 최대 77% 까지 비용 절감이 가능함을 보였다. 또한, 모델의 학습부터 배포까지의 파이프라인 절차를 스팟 인스턴스와 같이 리스크가 있는 컴퓨팅 자원을 최소한의 오버헤드로 안정적으로 이용할 수 있음을 보였다. 이 플랫폼을 통해 머신러닝 서비스를 배포한 경우, 지속적인 비용 모니터링을 통해 운영중인 인프라의 배포 수준을 재검토하는 것으로 머신러닝 인프라 비용을 최적해나갈 수 있음을 보였다.

이 프로젝트는 여기서 끝나는 것이 아닌 추후에도 지속적으로 발전시켜 현재는 단일 클라우드 제공업체의 컴퓨팅 자원 및 단일 지역(Region)의 컴퓨팅 자원만 활용하는것이 아닌 Amazon Web Services, Microsoft Azure, Google Cloud Platform 등의 여러 클라우드 제공업체를 활용하여 더

넓은 컴퓨팅 자원 풀에서 최적의 비용 및 성능을 가진 컴퓨팅 자원을 선출하여 사용하는 것 뿐만 아닌, 전세계에 흩어져있는 클라우드 지역 별 자원까지 활용할 수 있도록 할 계획이다. 이번에 산학협력을 수행한 아마존 웹 서비스와도 지속적으로 긴밀하게 소통하여 이 프로젝트를 다른 기관에 소개하거나, 발전시켜나가 실제 프로덕트까지도 가능하다면 도전해볼 계획이다. 또한 오픈소스로 공개되어있어 이 프로젝트를 우리뿐만 아닌 다른 사람들도 적극적으로 참여할 수 있도록 지속적인 지원을 아끼지 않을 예정이다.

4 자기평가

송무현	“최적의 GenAIOps 환경을 제공하는 플랫폼” 프로젝트를 진행하면서 상당한 좋은 경험을 얻어간 것 같습니다. 이 프로젝트를 진행하면서 많은 개발 역량 향상 뿐만 아니라, 소프트웨어공학같은 개발 방법론에 대해서도 다시 한번 돌아보게되는 좋은 계기가 되었습니다. 저는 이 프로젝트의 여러 Sub System의 아키텍처 설계 제안 및 스팟 과금 환경에서의 추론 환경 구현과 진행 총괄을 담당하였습니다. 또한, 이 프로젝트의 팀장으로서 제가 맡은 개발 파트 뿐만 아니라 프로젝트의 각기 다른 Sub System에 대해서도 진행방향을 제안하고 코드 리뷰를 진행하는 것은 기존 소규모 프로젝트에서는 경험하지 못할 일이었을 겁니다. 또한, 처음 이 프로젝트가 진행이 중간 정도 되었을때에는 ‘우리가 너무 어려운 주제를 선택했나?’ 라는 막연한 생각이 들기도 하였으나, 팀원 모두가 책임감을 가지고 하나되어 결국 완성한 것에 대해 매우 자랑스럽게 생각합니다. 이 프로젝트를 통해 팀원이 모두 성장했음 또한 느낍니다. 이 프로젝트를 더욱이 발전시켜 많은 사람들이 사용하게 된다면 인공지능에서의 학습 및 추론, 데이터 처리 같은 컴퓨팅 자원이 많이 필요한 작업을 비용과 성능 효율적으로 처리할 수 있을 것으로 기대합니다.
김규민	저는 본 프로젝트에서 학습 전반에 관련된 구조 설계 및 구현을 담당하였습니다. 제공받은 모델을 바탕으로 학습 요청이 들어오면 요청을 분산 처리 작업으로 변환하여 클러스터에서 동작할 수 있도록 하고, 학습 도중에 컴퓨팅 자원이 종료되더라도 학습 상황이 저장된 체크포인트로 학습을 재개하는 부분하도록 구현하였습니다. 수행 과정에서 컨테이너 오케스트레이션을 담당하는 쿠버네티스와 분산 처리 프레임워크 이해도, ML 모델 관련 지식들이 크게 향상되었습니다. 특히 ML의 경우 사용자로부터 제공받는 PyTorch 기반 모델을 우리의 서비스에서 읽을 수 있도록 추상화하는 과정을 통해 모델과 추론 제공 과정에 대한 이해도가 크게 향상되었습니다. 하지만 그러한 사항들을 구현하기 위해 처음 접하는 프레임워크를 사용하는 것은 힘든 부분으로 작용하였습니다. 특히나 분산 처리 프레임워크의 경우 모든 모델들이 병렬 학습 과정 이후 올바르게 동기화하고 체크포인트로서 저장하게 하는 부분에서 어려움이 있었습니다. 하지만 분산처리 과정을 이해하고 학습 처리 단계를 명확하게 구분하여 설계하며 각 프레임워크의 역할과 사용법을 이해할 수 있었습니다. 완성된 프로젝트는 사용자가 간단하게 인공지능 학습 및 추론 플랫폼을 구축할 수 있게 해주는 서비스로, 클라우드 컴퓨팅을 통한 대용량 모델 처리에 익숙하지 못한 사람들도 간단하게 플랫폼을 구축할 수 있다는 장점이 있습니다. 인공지능의

	중요성과 대용량 모델의 위력은 널리 알려지고 있지만, 그에 비해 해당 기술들이 동작할 수 인프라 구축은 아직까지 덜 조명되어있으며 관련 지식들을 모르는 사람들이 많습니다. 저희 프로젝트의 결과물은 그런 사용자들에게 자신이 원하는 규모의 플랫폼을 비용효율적으로 구축할 수 있게 해주어 이미 다가온 AI시대에서 각자의 아이디어를 실현할 수 있게 해주는 좋은 도구가 되어줄 수 있을 것 입니다. 6인 팀인것을 고려하여도 큰 프로젝트 하나를 성공적으로 완성할 수 있게되어 매우 기쁩니다. 특히, 각자 맡은 기능 개발을 진행하면서도 누군가 문제가 발생하면 이에 대해 서로 논의하고 해결방안을 제시해주며, 아는 부분에 대해 친절하게 설명해준 팀원들에게 큰 감사를 표합니다.
김유림	이 프로젝트에서 저는 핵심 개발 요소 중 Train 파트를 담당했습니다. 특히 스팟 인스턴스 인터럽트 대응을 위한 체크포인트 기능 구현과 파운데이션 모델의 파인튜닝을 위한 'Llama-2-7b-chat-hf' 및 'Stable-Diffusion-v1-4' 학습 및 추론 코드를 구현하였습니다. 프로젝트를 진행하면서 모델에 대한 이해도가 크게 향상되었습니다. 파인튜닝을 위해 학습 코드를 작성하면서 모델의 학습 과정을 보다 깊이 이해할 수 있게 되었습니다. 무엇보다도 협업을 통해 다른 사람의 코드를 쉽게 이해하는 능력이 향상되었고, 협업 개발에 더 익숙해질 수 있었습니다. 머신러닝을 이용한 프로젝트 경험이 처음이었기에 배경지식이 부족한 부분이 많았습니다. 이로 인해 파운데이션 모델의 파인튜닝 코드를 구현할 때 자주 에러가 발생했고, 이를 해결하는 것이 가장 어려운 부분이었습니다. 또한 체크포인트 기능을 구현하기 위해 구체적인 시나리오를 작성하고, 이를 의도대로 구현하는 과정도 힘들었습니다. 그러나 이러한 과정을 통해 좋은 개발 방식을 깨닫게 되었고, 시간 관리의 중요성과 문제 해결 능력을 키우는 데 많은 도움이 되었습니다. 이 프로젝트는 모델 학습, 추론, 배포 과정을 어렵게 느끼는 MLOps나 GenAIOps에게 유용한 도구가 될 것입니다. 인프라 구축이 어려운 분들도 쉽게 사용할 수 있도록 많은 배려를 기울였으며, 사용자가 자신만의 모델을 학습할 수 있도록 일반화하는 작업도 진행하였습니다. 이를 통해 많은 분들에게 도움이 될 수 있는 머신러닝 파이프라인 플랫폼이 될 것이라 기대합니다. 지난 겨울 학기부터 약 반년 동안 함께 고생한 팀원 여러분, 정말 수고 많으셨고 고생하셨습니다. 함께 했기에 많은 것을 배울 수 있었고, 혼자가 아니라서 포기하지 않을 수 있었습니다. 보람차기도 했고 힘들기도 했지만, 함께 마무리할 수 있어서 정말 뿌듯했습니다. 감사합니다!
문지훈	팀원 모두가 각자의 영역에서 진행하고 의견을 공유하면서 성공적으로 끝마친 이 프로젝트, “최적의 GenAIOps 환경을 제공하는 플랫폼”은 인공지능 모델 개발자가 클라우드 인프라 및 배포 환경에 대한 배경지식이 부족해도 손쉽게 모델 학습부터

	배포까지 진행할 수 있도록 돕습니다. 이를 통해 모델 개발자는 모델 개발에만 집중할 수 있으며, 모델의 성능을 평가해 최적의 컴퓨팅 자원을 사용하여 시간과 비용을 절약할 수 있다는 것이 가장 큰 장점입니다. 저는 이 프로젝트의 추론 작업을 담당했습니다. 서버리스 추론 자동화 구현을 통해 구성된 모델을 컨테이너 이미지로 빌드하고 서버리스 환경으로 배포하여 추론을 진행할 수 있도록 했습니다. 또한, Foundation 모델로 선정된 Llama와 Stable Diffusion의 추론 자동화도 구현했습니다. 이렇게 생성한 추론 요청을 더욱 손쉽게 보낼 수 있도록 채팅 형태의 앱인 Streamlit을 자동으로 배포하도록 설정했습니다. 처음 이 프로젝트를 시작할 때 머신러닝에 대한 배경지식이 전무한 상태로 시작해 학습과 동시에 구현한다는 것이 어려웠지만, 프로젝트를 진행하면서 기초부터 차근차근 배워 나갔고, 머신러닝의 원리와 적용 방법을 이해하게 되었습니다. 특히, 데이터 전처리, 모델 학습, 평가 및 튜닝, 그리고 추론 과정까지 전체적인 구조를 알게 되었습니다. 뿐만 아니라 쿠버네티스, 서버리스, 클라우드 기술을 직접 적용해보며 인프라 관리와 자동화의 중요성도 알게 되었습니다. 이러한 기술들을 활용하여 시스템의 유연성과 확장성을 극대화할 수 있었고, 운영 효율성을 높일 수 있었습니다. 특히, 서버리스 아키텍처를 도입함으로써 유지보수 부담을 줄이고, 비용 효율적인 운영을 달성했습니다. 이러한 경험을 통해 클라우드 환경에서의 개발과 운영에 대한 실질적인 지식을 쌓을 수 있었습니다. 기술 외적인 부분에서는 협업의 중요성을 깨달았을 뿐만 아니라 책임감도 크게 성장하게 되었습니다. 이 프로젝트는 팀 구성원들 각자가 맡은 역할만 수행하는 것이 아니라, 모두가 서로의 영역을 함께 진행하며 성공적으로 완성했습니다. 덕분에 다양한 관점에서 문제를 해결하고, 팀의 시너지를 극대화할 수 있었습니다. 또한, 이러한 협업을 통해 서로의 강점을 최대한 활용할 수 있었고, 개인의 성장뿐만 아니라 팀원 모두 성장하게 된 계기가 되었습니다.
박정명	다학제간 캡스톤디자인 프로젝트를 진행하며 데이터셋, 모델과 같은 자원관리 및 모델 학습, 추론 엔드 포인트를 실제로 사용자에게 도출할 수 있는 프론트엔드 콘솔 제작 및 필요한 요소들을 저장하고 관리하는 데이터베이스 및 스토리지 관련 API 작업을 진행하였습니다. 각 분야(Model, Train, Inference)를 담당한 팀원들의 작업물을 React.js를 이용하여 마치 물 흐르듯 하나의 서비스가 동작하는 것처럼 사용자가 이용할 수 있도록 코드를 작성하였고, 실제 서비스에서 필요한 일련의 데이터를 저장할 수 있도록 효율적인 NoSQL 데이터베이스 테이블을 설계 및 구성하였으며 테이블, 스토리지를 관리할 수 있는 API를 작성하였습니다. 진행하며 클라우드, 생성형 AI, MLOps 등 학부 생활을 하면서 지금까지 몰랐던 생소한 용어와 주제에 마주하게 되었으나 프로젝트를 성공적으로 마치며 전보다 많은 이론에 대해 공부할 수 있었고 이를 직접 적용하며 작업하다 보니 어느새 친숙하게 인공지능이라는 이론을 접해볼 수 있었습니다. 그 외에도, 프론트엔드 콘솔을 제작해

	사용자에게 어려운 내용을 UI/UX 적으로 풀어나갈 수 있을지 고민하는 계기를 갖기도 하였으며 새롭게 사용했던 디자인 프레임워크를 통해 빠른 개발도 진행해 볼 수 있었습니다. 물론, 프론트엔드 콘솔 및 API를 작업하면서 혼자 한다는 것이 힘든 것도 있었으나 이는 기존에도 진행해 봤던 내용이기에 큰 어려움은 없었고 이보다 팀원들이 작업한 내용을 하나의 서비스로 제공할 수 있도록 협업하며 소통하는 과정 및 이를 코드로 풀어쓰는 작업에서 많은 힘을 들였다고 생각합니다. 팀원들과 이슈 공유 및 주기적인 스크럼을 통해 좋은 방향으로 해결할 수 있었고 결론적으로 프로젝트가 성공적으로 진행될 수 있었다고 생각합니다. 결론적으로 도출된 프로젝트는 처음 작업에 임하기 전, 과거의 저와같이 배경지식이 부족한 사용자에게 최적의 솔루션이 될 수 있다고 예상합니다. 클라우드의 인프라를 전혀 모르더라도 클라우드를 이용하여 머신러닝을 진행하고 모델을 배포해볼 수 있으며 생성형 AI를 자신만의 데이터셋을 이용해 특정 도메인에 특화된 LLM을 만들어 Testbed로 챗봇을 테스트해 볼 수도 있습니다. 이 모든 것을 기존의 클라우드 업체에서 제공하는 서비스보다 저렴한 가격에 편리하게 사용할 수 있다는 것이 저희 서비스만의 장점이라고 생각합니다. 프론트엔드 및 백엔드 부분만 담당했지만, 결과적으로 하나의 거대한 서비스를 만들 수 있었던 건 옆에 있는 팀원들의 능력과 소통 덕이라고 생각합니다. 다학제간 캡스톤 디자인을 통해서 협업이라는 의미에 대해 소중하게 생각하는 계기가 되었고 개발자로서 성공적인 협업을 통해 사회에 긍정적인 효과를 낼 수 있는 사람이 되도록 노력하겠습니다.
정승우	저는 본 프로젝트를 진행하며 다음과 같은 역할을 맡았습니다. 우선 최적의 인스턴스를 선출하는 알고리즘을 구현하여 본 플랫폼이 자동화된 비용 최적화 서비스를 제공할 수 있도록 하였습니다. 해당 알고리즘을 이미지화하여 IaC 배포 시 자동으로 API 로 생성될 수 있게 하였습니다. 또한, Karpenter 라는 node provisioning application 을 사용하여 직접 인스턴스를 키지 않아도 제가 구현한 인스턴스 선출 알고리즘에 의해 선출된 인스턴스 그룹을 노드 풀로 생성하여 본 플랫폼에서 학습 및 추론 서비스 배포 시 자동으로 scale-out 할 수 있도록 하였습니다. 이후에는 팀원들이 테스트 할 수 있게끔 모델 학습 및 추론 예제를 제공하였으며, 문서 작업 등 공동 작업을 지원하였습니다. 해당 프로젝트를 하며 저에게 있어서 가장 어려웠던 것은 팀원과의 생각 일치였습니다. 의견을 공유함에 있어서 생각하는 방향이 같아야 한다는 것을 느꼈습니다. 또한, 이 프로젝트에서 어떠한 기능 개발 뿐만 아니라 배포에 대한 이해를 높일 수 있었으며, 프론트 담당 동료와 함께 일하면서 프론트와의 연결방식도 생각해 볼 수 있었습니다. 이 플랫폼에서는 클라우드 사용에 어려움을 겪는 ML 전문가가 손쉽게 자신이 구현한

	모델을 학습 및 추론할 수 있으며, 타 서비스와의 최대 차이점은 가장 저렴한 비용으로 높은 작업 신뢰성을 제공할 수 있다는 것입니다. 해당 프로젝트에서 팀 프로젝트란 무엇인가를 처음으로 느끼게 되었던 것 같습니다. 일정 규모 이상의 프로젝트는 혼자서는 할 수 없으므로, 이 프로젝트의 경험으로 다음에는 어떻게 소통하고 진행해야 할 지 많이 배운 것 같습니다.

5 참고 문헌

[1] https://blogs.nvidia.co.kr/2023/04/04/what-are-foundation-models/

[2] https://aws.amazon.com/ko/blogs/korea/amazon-sagemaker-serverless-inference-machine-learning-inference-without-worrying-about-servers/

[3] https://aws.amazon.com/ko/what-is/mlops/?nc1=h_ls

[4] https://developer.nvidia.com/ko-kr/blog/mastering-llm-techniques-llmops/

[5] https://criu.org/

[6] https://tutorials.pytorch.kr/recipes/recipes/saving_and_loading_a_general_checkpoint.html

6 부록

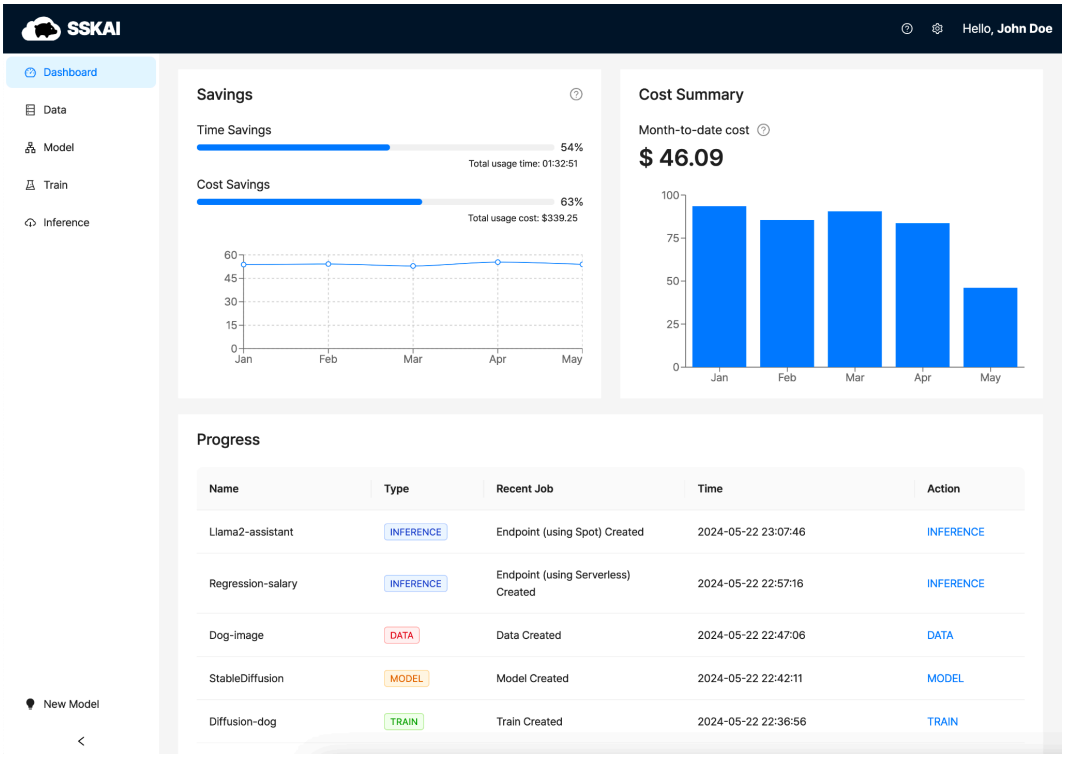
6.1 사용자 매뉴얼

서비스를 제공하는 웹 페이지의 기능은 크게 아래와 같다.

- Dashboard 페이지: 현재까지의 절감 비율, 금액 및 로그 조회
- Data 페이지: 모델 학습에 필요한 데이터셋 관리
- Model 페이지: 모델 관리
- Train 페이지: 모델 학습 관리
- Inference 페이지: 추론 엔드포인트 관리

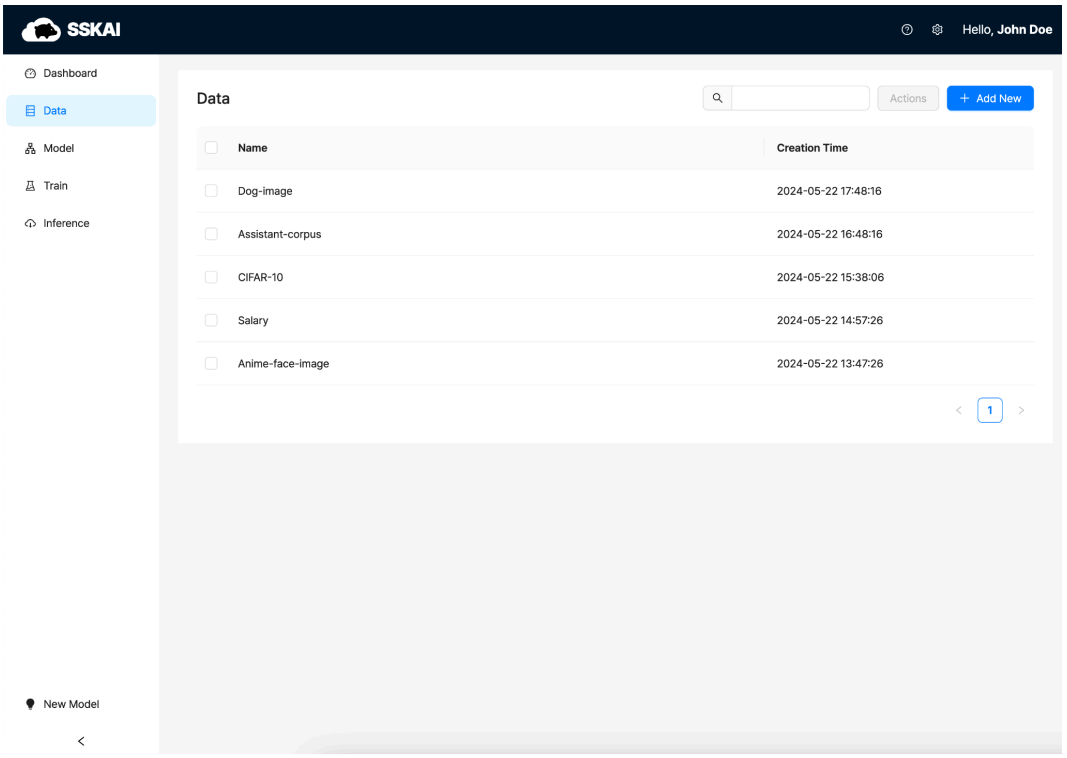
- New Model 페이지: 사용자 모델 또는 파운데이션 모델 생성

6.1.1 Dashboard 페이지

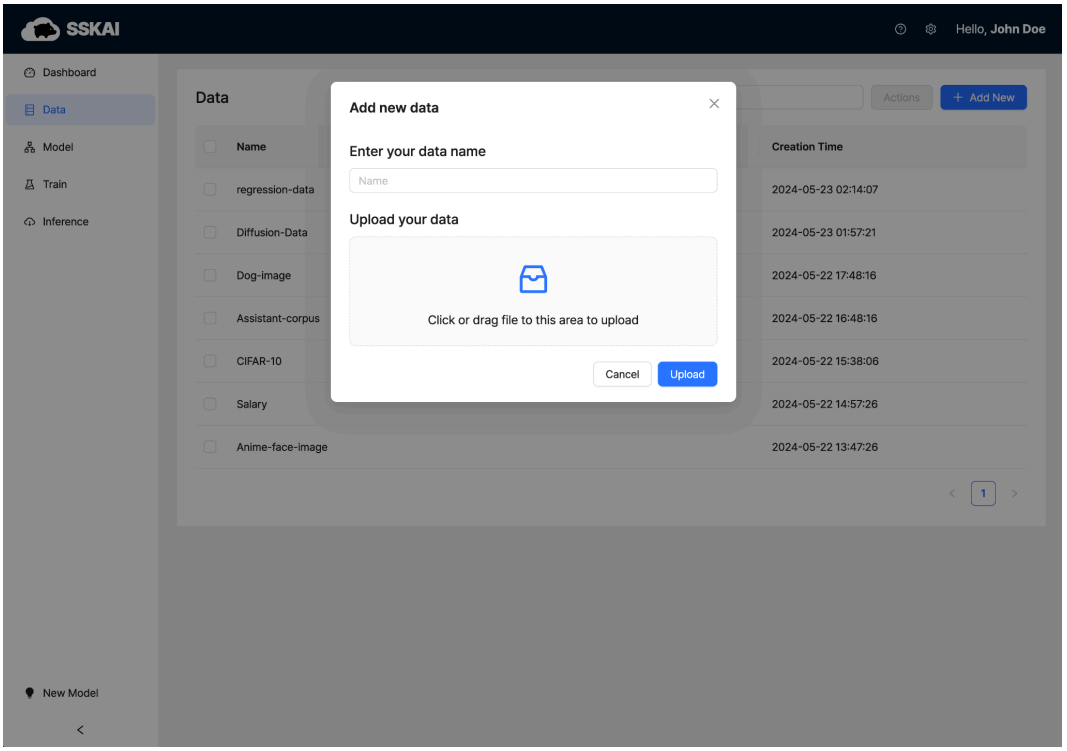


- 서비스 사용을 통한 시간 및 금액에 대한 절감 비율, 가격, 로그 등을 조회할 수 있다.
- 선 그래프, 막대 그래프 등의 지표를 이용하여 최근 추이를 확인할 수 있다.
- 서비스 사용과 관련된 사용자의 로그를 조회할 수 있다.

6.1.2 Data 페이지

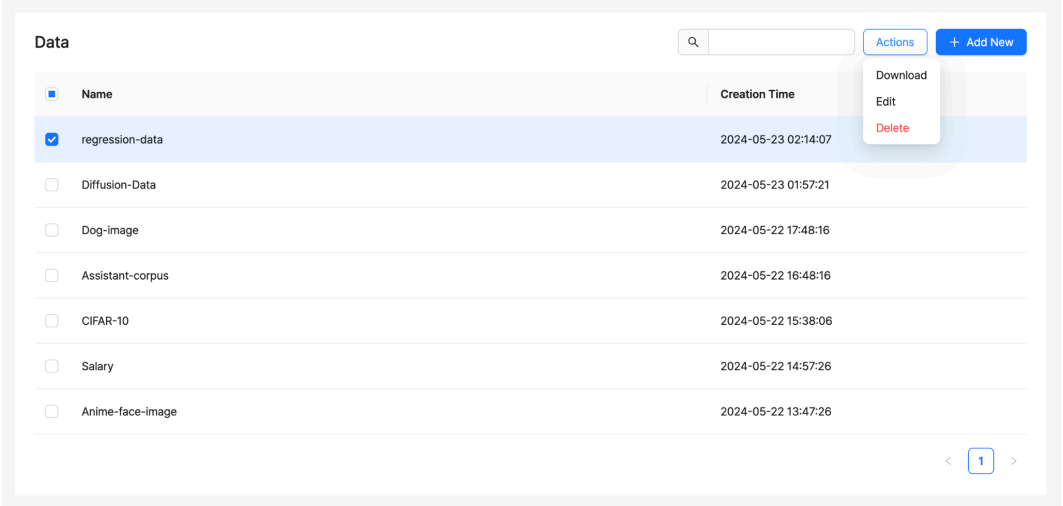


- 데이터셋을 조회, 업로드, 수정, 삭제할 수 있는 기능을 제공한다.
- 업로드 된 데이터셋을 기본적으로 테이블 화면에서 확인할 수 있고, 검색 기능을 활용하여 조회할 수 있다.
- 새로운 데이터 셋을 추가하는 방법은 아래와 같다.

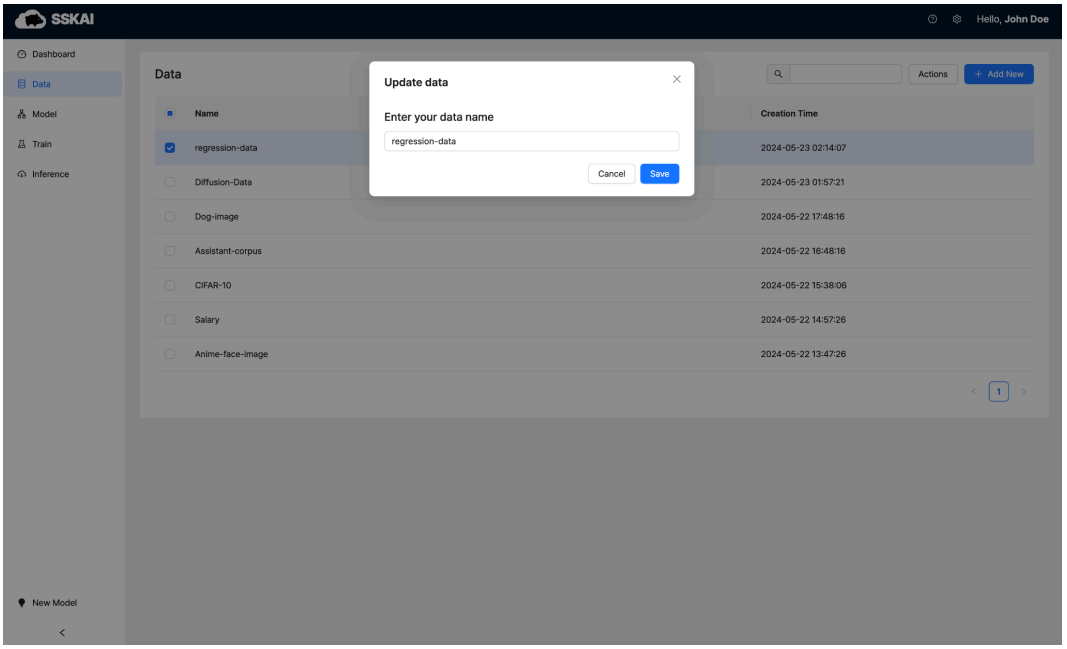


1. 20자 이내 영문과 특수문자 '_', '-'를 이용하여 이름을 작성한다.
2. 파일 선택 또는 드래그 앤 드롭을 이용한 .zip 형식의 파일을 첨부한다.
3. 업로드 버튼을 클릭한다.

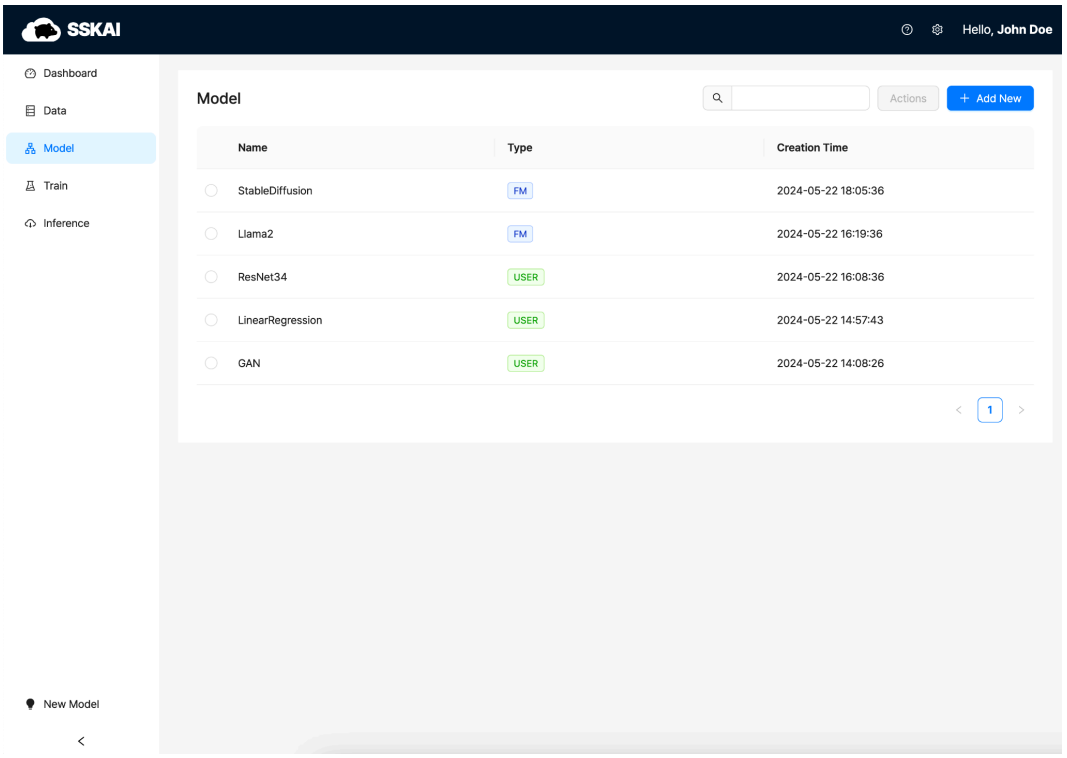
- 업로드 된 데이터셋에 대해 다운로드, 삭제 및 이름 수정과 같은 기능을 제공한다. 방법은 아래와 같다.



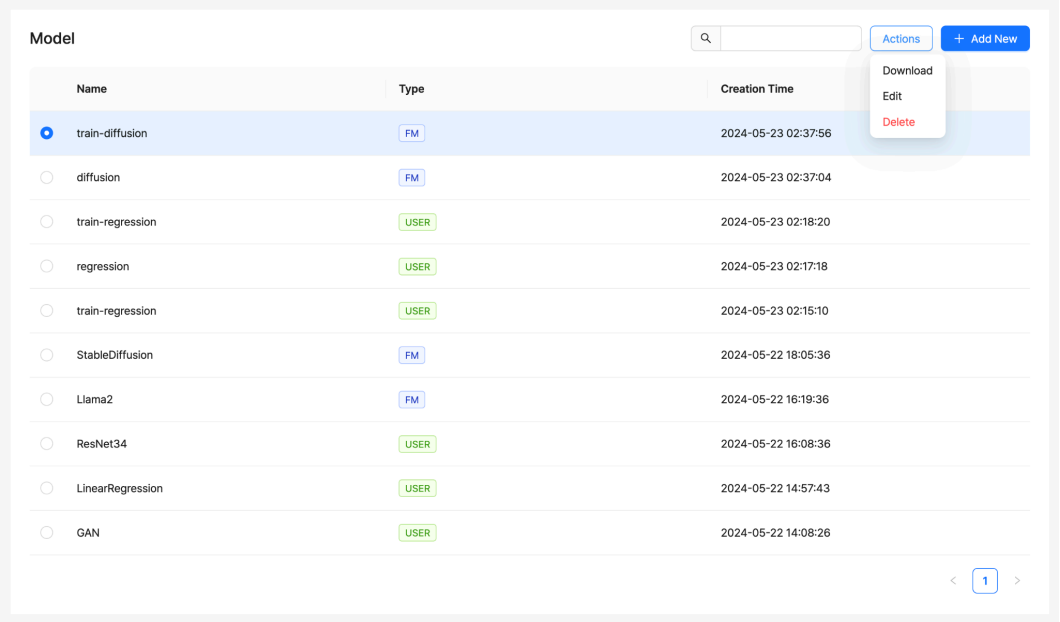
- 해당하는 데이터의 Checkbox 선택 및 Actions → Download를 클릭하면 데이터셋을 다운로드 받을 수 있다.
- 해당하는 데이터의 Checkbox 선택 및 Actions → Delete를 클릭하면 데이터셋을 삭제할 수 있다.
- 해당하는 데이터의 Checkbox 선택 및 Actions → Edit을 클릭하면 아래와 같이 데이터셋의 이름을 수정할 수 있다.



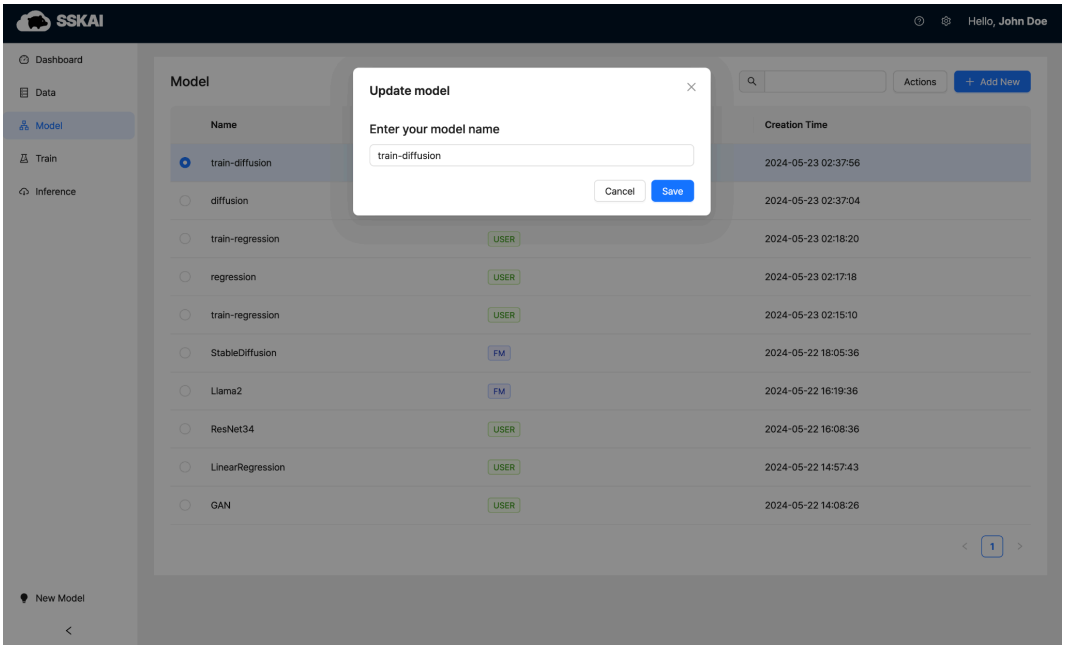
6.1.3 Model 페이지



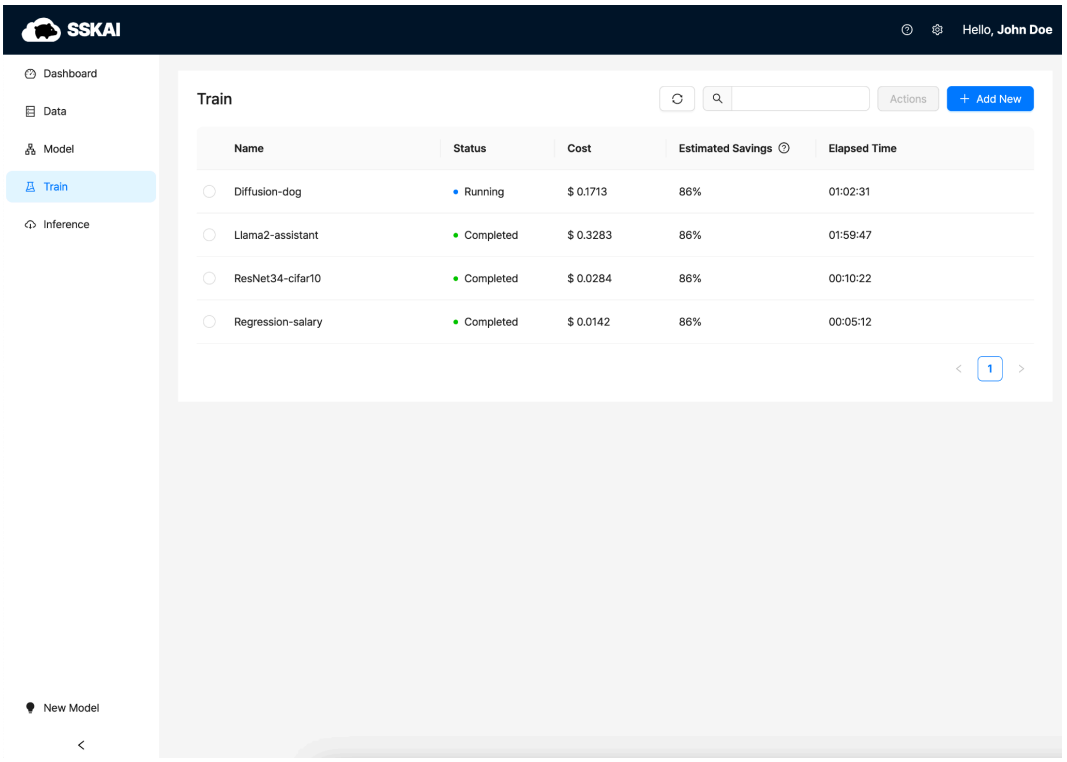
- 모델을 조회, 수정, 삭제할 수 있는 기능을 제공한다.
- 생성 또는 등록한 모델을 기본적으로 테이블 화면에서 확인할수 있고, 검색 기능을 활용하여 조회할 수 있다.
- 새로운 모델을 추가하는 방법은 'New Model' 페이지 설명에 첨부하도록 하겠다.
- 등록되어 있는 모델에 대해 삭제 및 이름 수정과 같은 기능을 제공한다. 방법은 아래와 같다.



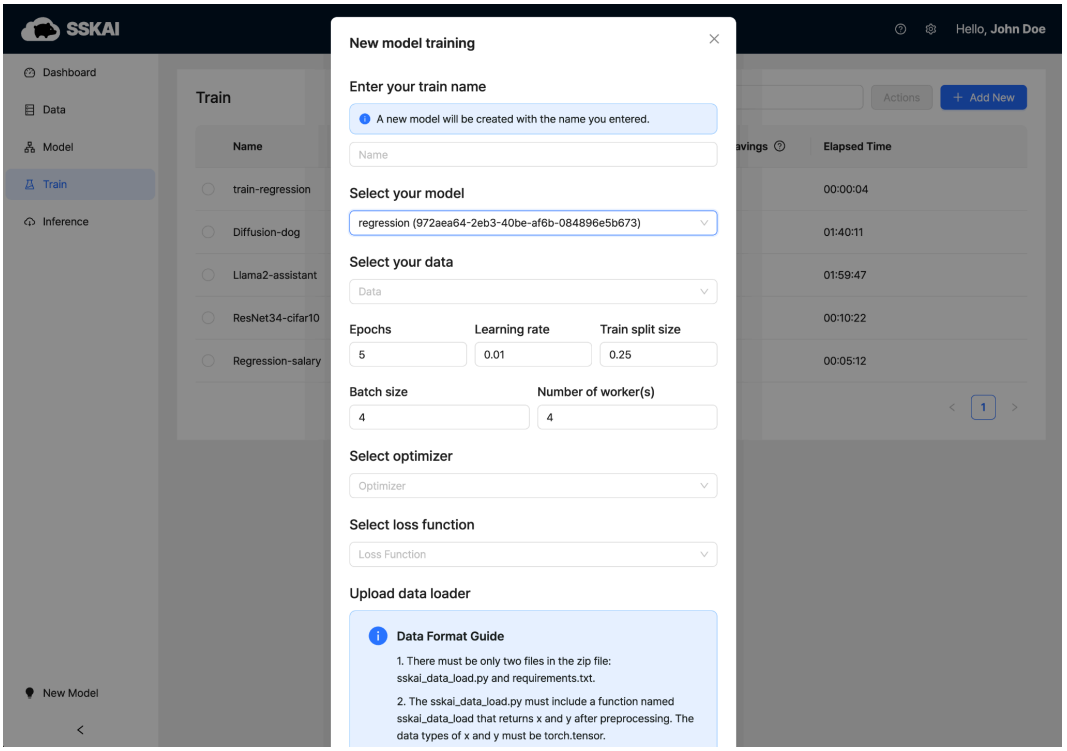
- 해당하는 모델의 Radio 선택 및 Actions → Download를 클릭하면 모델을 다운로드 받을 수 있다.
- 해당하는 모델의 Radio 선택 및 Actions → Delete를 클릭하면 모델을 삭제할 수 있다.
- 해당하는 모델의 Radio 선택 및 Actions → Edit을 클릭하면 모델의 이름을 수정할 수 있다.

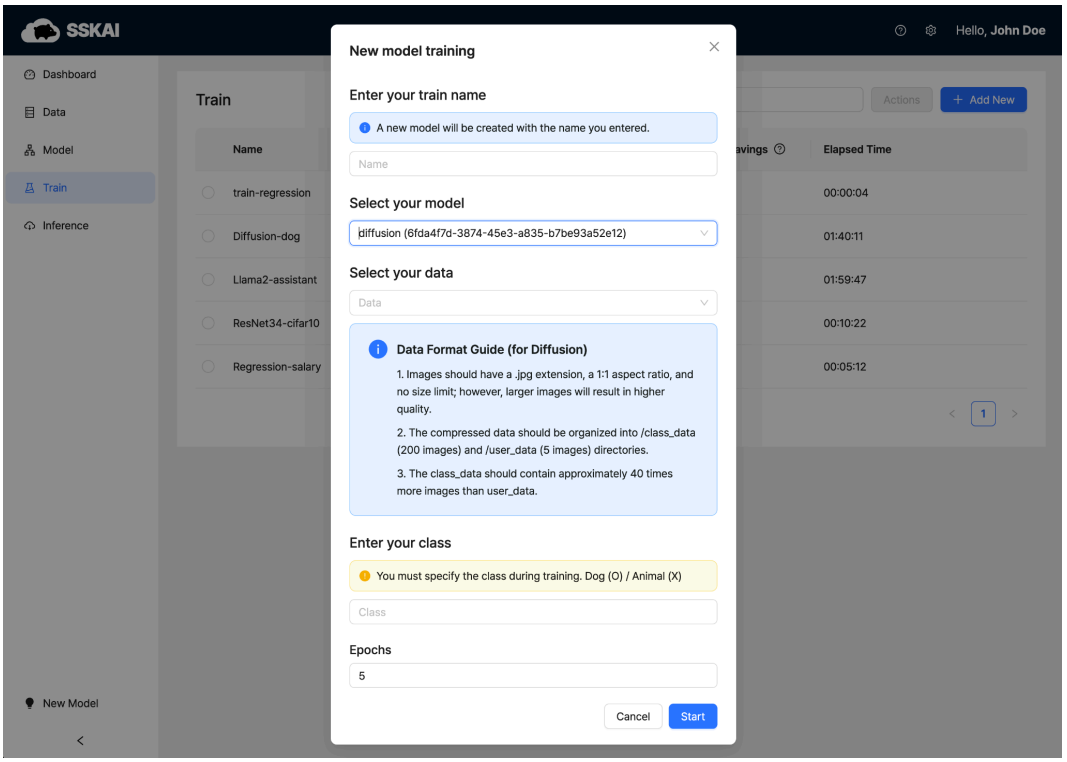


6.1.4 Train 페이지

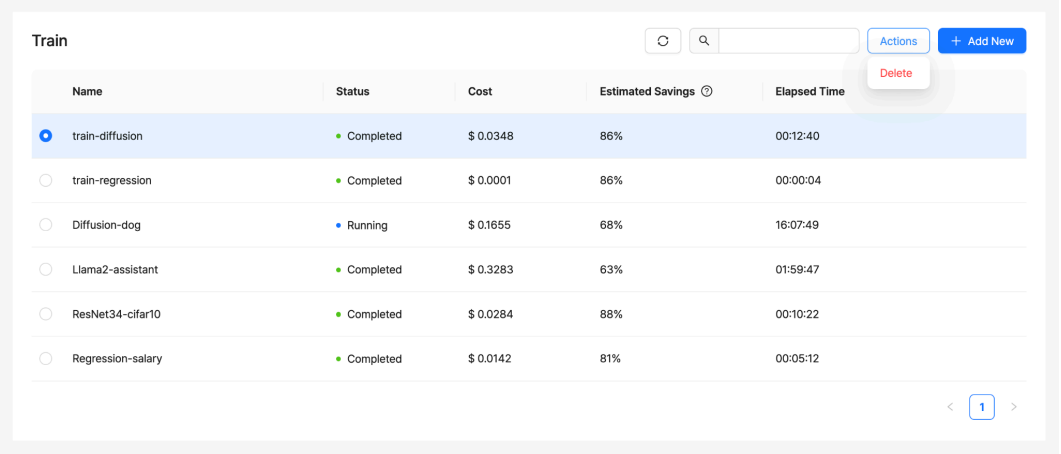


- 학습을 조회, 생성, 삭제할 수 있는 기능을 제공한다.
- 모델에 대한 학습을 기본적으로 테이블 화면에서 확인할수 있고, 검색 기능을 활용하여 조회할 수 있다.
- 추가적으로 학습의 진행 상태, 가격, 진행 시간 등을 조회할 수 있다.
- 새로운 모델에 대한 학습을 진행하는 방법은 아래와 같다.



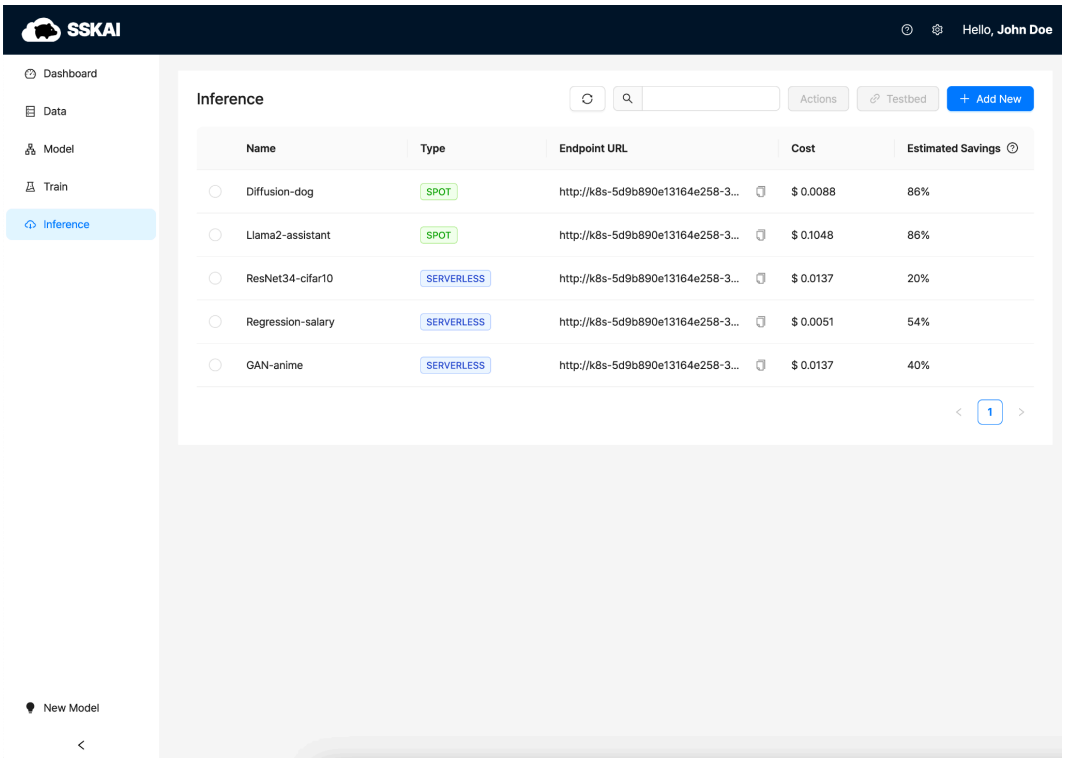


- 20자 이내 영문과 특수문자 '_', '-'를 이용하여 이름을 작성한다.
    - 이름은 학습 후에 나오는 모델의 이름으로 사용된다.
- 학습 대상이 될 모델을 선택한다.
- 업로드 된 데이터셋을 선택한다.
- 모델 타입 (사용자 모델, FM 등)에 따라 요구하는 양식이 변경된다. 작성 양식은 아래와 같다.
    - 사용자 모델의 경우 Epoch, Learning rate, Train split size, Batch Size, Number of worker(s), Optimizer, Loss Function을 입력 및 Data loader를 업로드한다.
        - 파일 선택 또는 드래그 앤 드롭을 이용한 파일을 첨부한다.
    - llama 모델의 경우 Epoch를 입력한다.
    - diffusion 모델의 경우 Epoch, Class를 입력한다.
- 학습에 대해 삭제 기능을 제공한다. 방법은 아래와 같다.

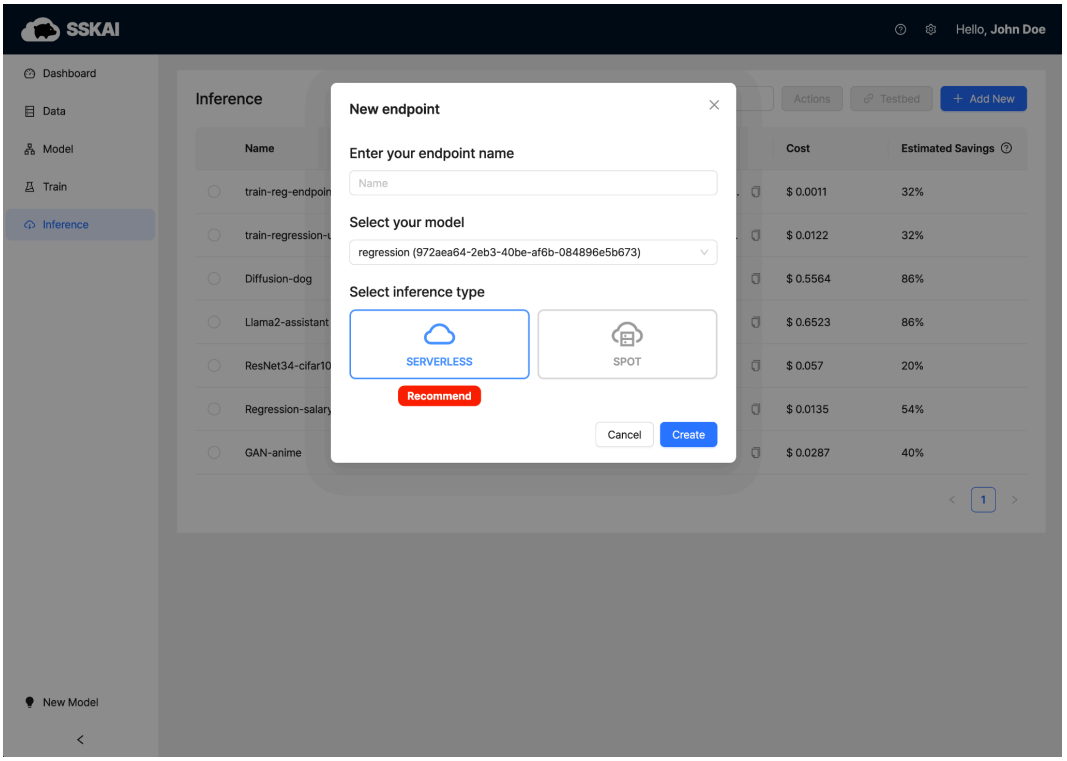


- 해당하는 학습의 Radio 선택 및 Actions → Delete를 클릭하면 학습을 삭제할 수 있다.

6.1.5 Inference 페이지

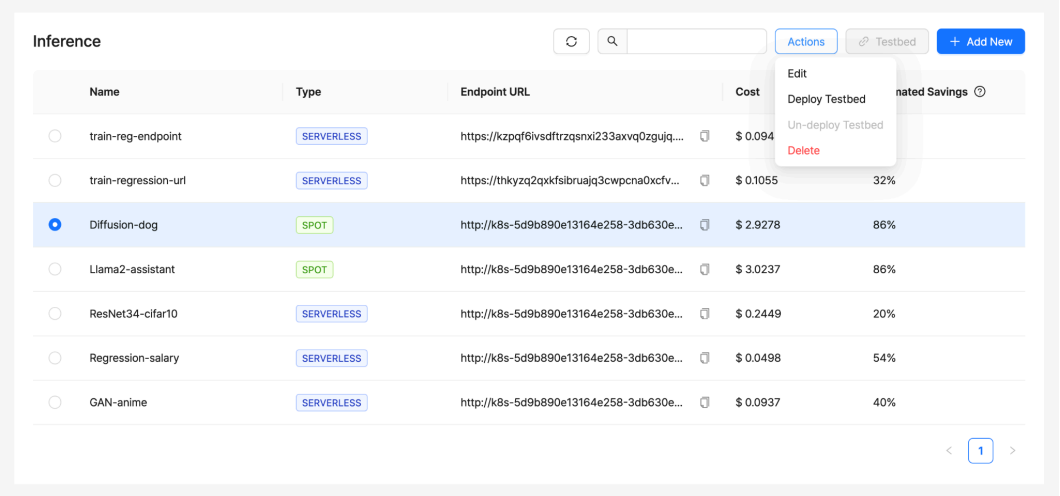


- 추론 엔드포인트를 조회, 생성, 변경, 삭제 및 Streamlit 배포 기능을 제공한다.
- 모델 추론 엔드포인트를 기본적으로 테이블 화면에서 확인할수 있고, 검색 기능을 활용하여 조회할 수 있다.
- 추가적으로, 엔드포인트를 클립보드로 복사할 수 있는 기능을 제공한다.
    - 엔드포인트 주소 옆에 클립보드 아이콘을 클릭하면 복사할 수 있다.
- 새로운 엔드포인트를 생성하는 방법은 아래와 같다.

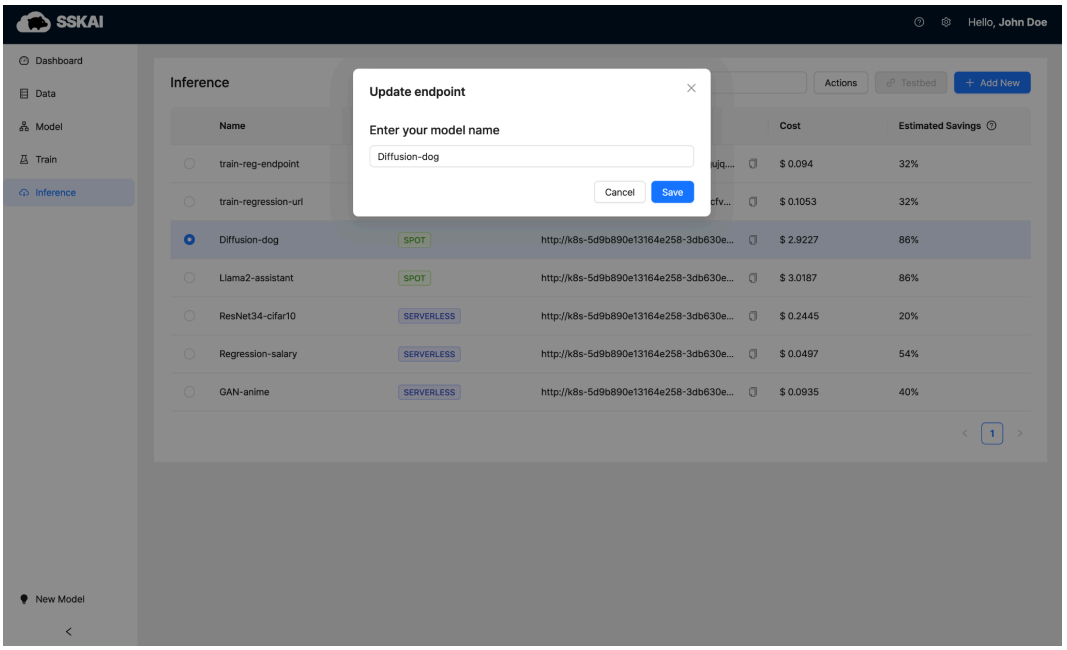


- 20자 이내 영문과 특수문자 '_', '-'를 이용하여 이름을 작성한다.
- 엔드포인트 생성 대상이 될 모델을 선택한다.
- 추론 방식을 선택한다 (Spot, Serverless).
  - 모델에 따라 추천되는 추론 방식을 확인할 수 있다.
- Create 버튼을 클릭한다.

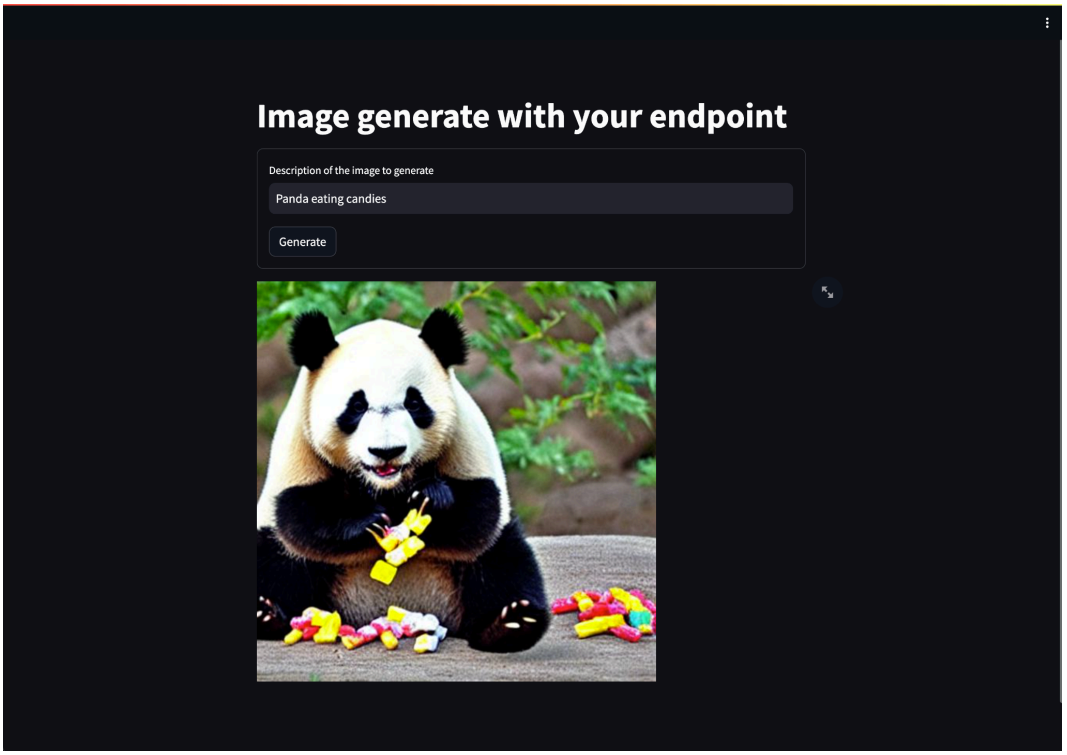
- 생성된 추론 엔드포인트에 대해 삭제 및 이름 수정, 테스트베드 배포와 같은 기능을 제공한다. 방법은 아래와 같다.



- 해당하는 엔드포인트의 Radio 선택 및 Actions → Delete를 클릭하면 모델을 삭제할 수 있다.
- 해당하는 엔드포인트의 Radio 선택 및 Actions → Edit을 클릭하면 모델의 이름을 수정할 수 있다.



- 엔드포인트 모델이 FM인 경우, 테스트베드 배포 기능을 제공한다.

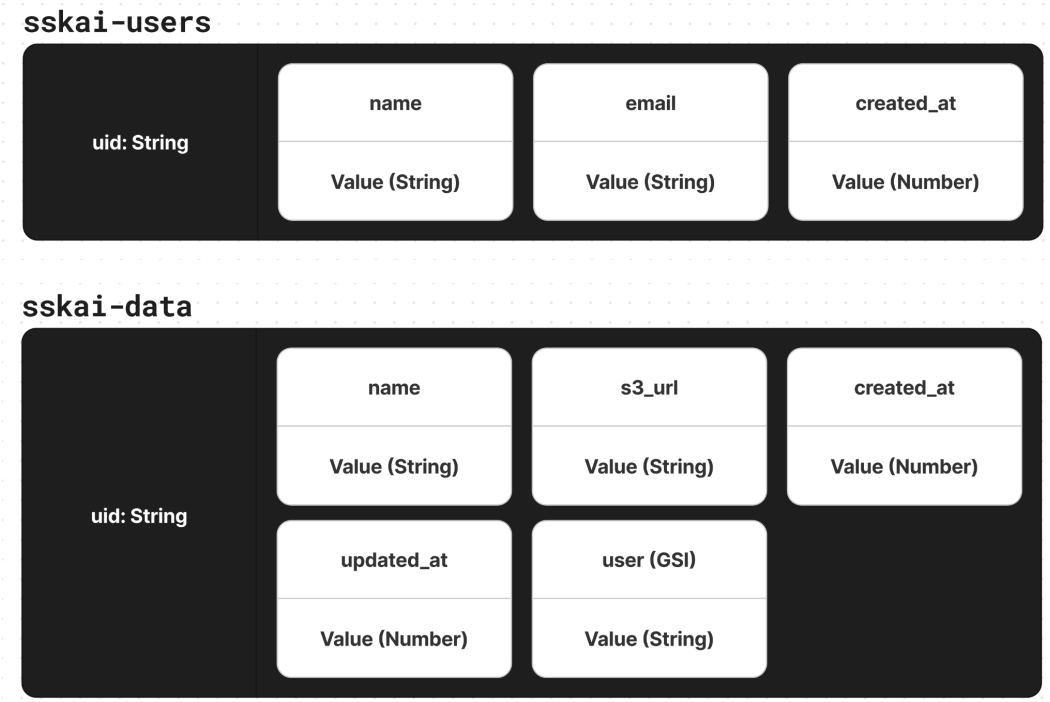


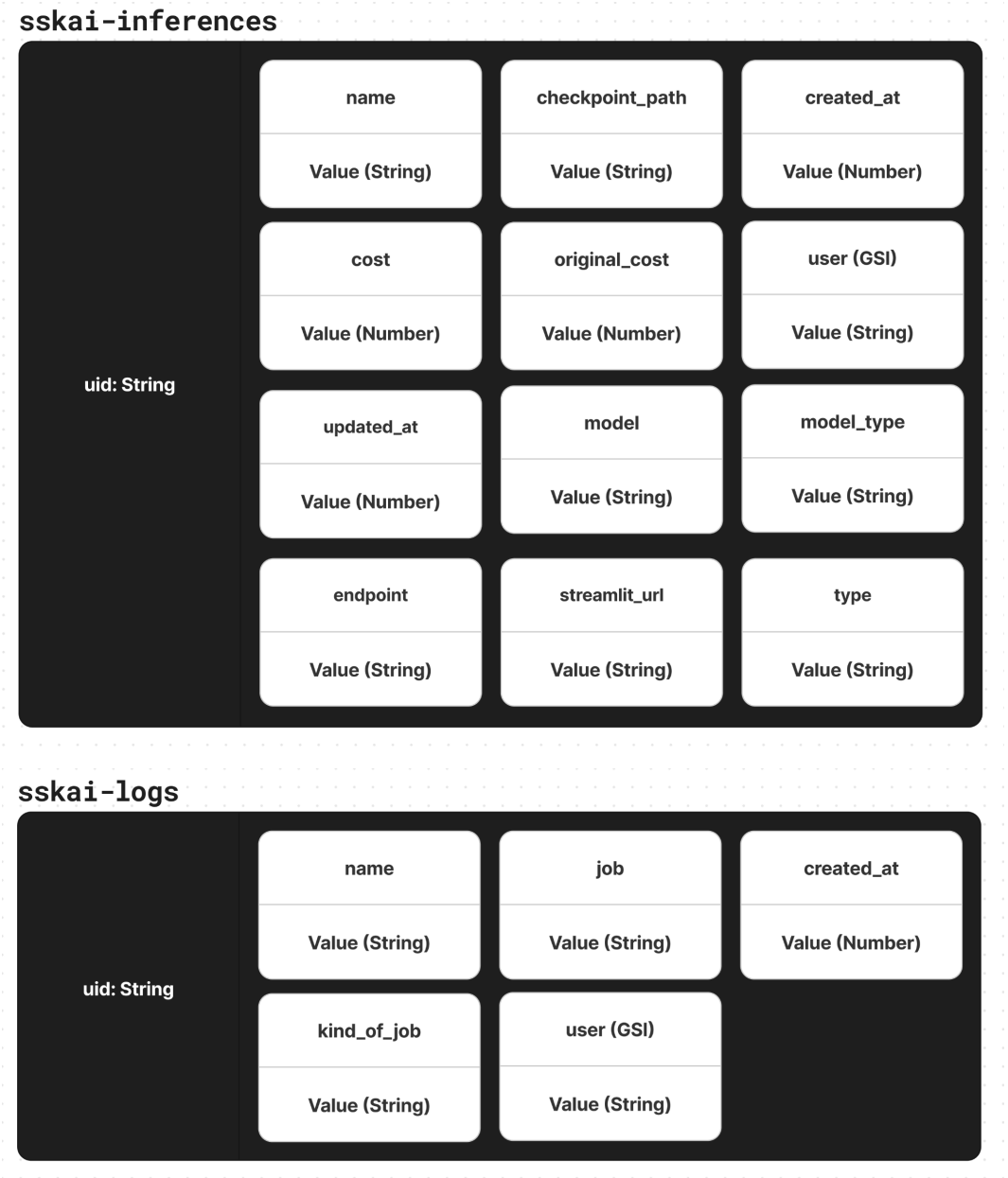
- 해당하는 추론의 Radio 선택 및 Actions → Deploy Testbed 혹은 Un-deploy Testbed를 클릭한다.
- Testbed 버튼 클릭 시, 배포된 사이트로 이동 가능하다.

6.2 운영자 매뉴얼

서비스를 운영하는 관리자의 경우 아래와 같이 데이터베이스 및 API를 관리할 수 있다.

6.2.1 데이터베이스





- 현재 데이터베이스는 AWS의 DynamoDB를 사용하여 위에 그려진 6개의 NoSQL Table을 운영한다.
    - Key-Value 형식으로 저장할 수 있고, NoSQL이기에 필드가 변경될 수 있으나 현재 서비스에서 사용되는 필드는 그림과 같다.
- 테이블의 Partition Key는 테이블 좌측에 적힌 uid (String) 로 고정이며 이는 UUIDv4를 사용하여 랜덤으로 부여된다.

- 또한, sskai-users 테이블을 제외한 모든 5개의 테이블에서는 user 필드에서 user-index 라는 Global Secondary Index를 통해 user의 uid 를 사용하여 쿼리가 가능하도록 설정한다.
- 서비스 운영 중, DynamoDB 테이블의 Record값을 조회하기 위해서는 AWS Web Console, CLI, SDK 등을 통해 배포되어 있는 region과 각 테이블명을 명시하여 조회할 수 있다.
    - 추가적으로 Record를 추가 및 제거하거나 변경하는 것도 위의 방식을 활용하여 진행할 수 있으나, 조회를 제외한 행위는 배포되어 있는 DB API를 이용하는 것이 적합하다.

6.2.2 데이터베이스 API

- DB API는 총 9개의 Lambda Functions를 API Gateway로 묶어서 하나의 URL에서 접근할 수 있도록 되어있다.
- DB API에서 예외 상황이나 오류가 발생한 경우에는, 문제가 발생했던 API 경로를 API Gateway에서 조회하여 해당 경로와 통합되어 있는 Lambda의 Error Log를 Cloudwatch를 이용하여 확인할 수 있다.
    - AWS의 웹 콘솔을 이용하여 확인할 수 있다.

6.2.3 모델 평가, 학습, 추론 API

- 학습 및 추론을 진행하는 API는 서비스의 콘솔에서 호출 하는 방식으로 코드가 작성되어 있다.
- 해당 API들은 Lambda를 통해 Kubernetes에 요청하여 학습이나 추론 등이 진행되고 해당 진행 사항은 Kubernetes 관리 도구인 kubectl와 AWS 콘솔을 통해 확인할 수 있다.

6.3 배포 가이드

이 프로젝트를 배포하기 위해서는 사용자가 사용하고자하는 컴퓨터의 아키텍처가 x86/64여야

하며, 운영체제는 Unix 계열(Linux, macOS)을 사용하여야 한다.

배포하기전 필요한 패키지로 Python3 언어 런타임, 웹 페이지를 빌드하기 위한 node와 yarn, 프로젝트의 API 컨테이너 이미지를 빌드하기 위한 Docker, ContainerD 와 클라우드 환경에 인프라를 배포하기 위한 Infrastructure as Code 프레임워크인 Terraform, AWSCLI를 설치해야 한다.

또한 사용자는 이 프로젝트를 배포하기 위해 아마존 웹 서비스 계정이 필요하며, 해당 계정은 관리자 권한을 소유하고 있어야한다.

사용자는 위 패키지와 계정을 모두 가지고 있다면, 프로젝트의 루트 경로에 위치하는 “sskai_execute.py”을 “python3 sskai_execute.py”와 같은 형태로 실행하면 사용자에게서 AWS Cloud의 Region, AWS CLI Profile 이름을 제공하면 2가지 형태의 작업을 수행할 수 있는 프롬프트를 출력한다.

```
jihun ~/Desktop python3 sskai_execute.py
Enter REGION: us-west-2
Enter AWSCLI PROFILE: mhsong
Enter MAIN SUFFIX: SSKAI
0. Exit this operation.
1. Build and Deploy container image.
2. Deploy SSKAI infrastructure.
Enter the number: 1
You can build only with x86/64 architecture and Unix kernel (Mac/Linux).

Enter the type of operation (create/delete): create
Building and Deploying in progress.
It takes about 15 minutes.
Processing...

Complete.
```

```
jihun ~/Desktop python3 sskai_execute.py
setup.env file exists. Do you want to use this file? (yes/no): yes
0. Exit this operation.
1. Build and Deploy container image.
2. Deploy SSKAI infrastructure.
Enter the number: 2
Enter the type of operation (create/delete): create
It takes about 20 minutes to create.
Processing...

Complete.
```

첫 번째로 실행되어야 할 작업은, 컨테이너 이미지를 빌드 후 배포를 수행해야 한다. 이 작업은

반드시 x86/64 아키텍처의 Unix 커널 운영체제 (Linux, macOS)에서 수행되어야 한다. 이 작업은 사용자가 배포하기 원하는 지역에 따라 최대 15분 가량 소요될 수 있다.

두 번째로 실행되어야 할 작업은, Terraform 을 사용하여 인프라 환경을 배포하는 것이다. 사용자는 추가로 설정을 할 필요가 없이 즉시 설정한 AWS 지역에 모든 인프라가 생성되게 된다.

```
jihun ~/Desktop python3 sskai_execute.py
setup.env file exists. Do you want to use this file? (yes/no): yes
0. Exit this operation.
1. Build and Deploy container image.
2. Deploy SSKAI infrastructure.
Enter the number: 2
Enter the type of operation (create/delete): delete
It takes about 20 minutes to delete.
Processing...

Complete.
```

```
jihun ~/Desktop python3 sskai_execute.py
setup.env file exists. Do you want to use this file? (yes/no): yes
0. Exit this operation.
1. Build and Deploy container image.
2. Deploy SSKAI infrastructure.
Enter the number: 1
You can build only with x86/64 architecture and Unix kernel (Mac/Linux).

Enter the type of operation (create/delete): delete
Deleting in progress.
It takes about 5 minutes.
Processing...

Complete.
```

마지막으로 삭제하는 경우에도 AWS에 배포된 리소스를 삭제하는 것과, AWS 지역에 업로드된 컨테이너 이미지를 삭제하는 작업 2가지를 순차적으로 “sskai_execute.py”를 통해 진행할 수 있다.