<http://3.36.101.198:5000/>

# **[ Databricks 사용 ]**

### **1. 데이터 수집 및 전처리**

* 데이터 수집: Databricks를 사용하여 데이터 수집 및 통합. 외부 데이터베이스에서 배치 데이터를 가져올 수 있음
* 데이터 전처리: Databricks 노트북을 사용하여 데이터 정제, 특성 추출 및 변환 과정을 자동화합니다.

### **2. 모델 학습 및 평가**

* 모델 학습: Databricks의 MLlib을 사용하거나, PySpark과 같은 라이브러리를 활용하여 머신러닝 모델을 학습. 딥러닝도 가능. Horovod라는 분산 딥러닝 훈련 프레임워크를 지원. 주피터 스타일의 노트북 제공. 데이터 전처리와 피처 엔지니어링 작업을 용이
* 모델 평가: 학습된 모델을 평가하여 성능 지표를 측정하고, 이를 MLflow에 기록
* 모델 저장: 학습된 모델을 MLflow를 통해 Databricks의 DBFS(Databricks File System)에 저장하거나, 외부 저장소에 저장.

### **3. 모델 배포 및 모니터링**

* 모델 배포: MLflow를 통해 학습된 모델을 배포하며, 이 모델을 Databricks의 REST API를 통해 사용할 수 있음
* 모델 모니터링: Databricks와 MLflow를 통해 모델 성능을 지속적으로 모니터링하고, 성능 저하 시 재학습을 트리거할 수 있음

### **4. 웹 설계**

* Streamlit 앱: Streamlit을 사용하여 모델 결과 시각화, 실시간 데이터 입력, 예측 결과를 받기

### **5. 통합 및 자동화**

* 자동화 파이프라인: Databricks의 작업 스케줄러를 사용하여 데이터 수집부터 모델 학습, 평가, 배포까지의 과정을 자동화
* CI/CD: 코드 변경 사항을 자동으로 배포하기 위해 GitHub Actions나 Jenkins와 같은 CI/CD 도구 사용 가능

# **[ AWS 정석 사용 ]**

### **1. 데이터 수집 및 전처리**

* AWS S3: 모든 원시 데이터를 저장하고 관리하는 데 사용. S3는 고가용성과 확장성을 제공하는 객체 스토리지 서비스.
* AWS Glue 또는 AWS Lambda: 데이터 카탈로그 작업을 자동화하고, 다양한 데이터 소스에서 ETL(추출, 변환, 로드) 작업을 수행. Glue는 데이터를 전처리하고 S3로 다시 저장하는데 유용

### **2. 모델 학습 및 평가**

* Amazon SageMaker: 머신러닝 모델을 빠르게 생성, 훈련 및 배포할 수 있게 지원하는 종합적인 머신러닝 서비스. SageMaker는 Jupyter 노트북 인스턴스를 제공하며, 다양한 머신러닝 알고리즘과 프레임워크를 지원
* Amazon SageMaker Experiments: 여러 훈련 실험을 추적하고 비교하는데 사용되고 실험 관리와 모델 평가를 간소화함

### **3. 모델 배포 및 모니터링**

* Amazon SageMaker Endpoints: 학습된 모델을 프로덕션 환경에 쉽게 배포할 수 있게 해주며, 이 엔드포인트를 통해 실시간으로 예측을 제공
* AWS CloudWatch: 모델의 성능을 모니터링하고 로그를 기록하는데 사용. 성능 지표가 특정 임계값 아래로 떨어지면 경고를 발생시킬 수 있음

### **4. CI/CD 및 자동화**

* AWS CodePipeline: 소스 코드 변경사항을 자동으로 감지하여 빌드, 테스트 및 배포하는 CI/CD 파이프라인을 구축
* AWS CodeBuild: 코드 컴파일, 테스트 실행 및 패키지 생성을 자동으로 처리하는 빌드 서비스

### **5. 사용자 인터페이스**

* Amazon API Gateway와 AWS Lambda: SageMaker 엔드포인트에 대한 접근을 관리하며, 사용자의 요청을 처리하고 결과를 반환하는 서버리스 방식으로 사용자 인터페이스를 제공

# **[ AWS + MLFLOW 통합 사용 ]**

### **1. 데이터 수집 및 전처리 (AWS)**

* AWS S3: 원본 데이터 저장
* AWS Glue 또는 AWS Lambda: 데이터를 자동으로 로드하고 전처리. 전처리된 데이터는 다시 S3에 저장

⇒ 데이터 수집 및 전처리 과정을 AWS Lambda/AWS Glue를 사용하여 자동화해서 새 데이터가 들어올 때마다 자동으로 처리되어 최신 데이터로 모델을 학습시킬 수 있음.

### **2. 모델 개발 및 실험 관리 (Google Colab & MLflow)**

* Google Colab: 머신러닝 모델 개발 및 테스트. MLflow와 연동하여 실험 관리를 수행
* MLflow: 모델의 트레이닝 파라미터와 메트릭을 추적하고 관리. MLflow는 Colab 환경 내에서 설정할 수 있으며, 트레이닝 로그와 아티팩트를 기록함.

### **3. 모델 아티팩트 저장 및 배포 및 모니터링 (AWS?)**

다양한 방법이 있음. 논의 필요

* AWS S3: MLflow를 통해 생성된 모델 아티팩트를 저장하며, 이 아티팩트는 추후 모델 배포에 사용됨
* AWS EC2/AWS Lambda: MLflow에서 저장한 모델을 로드하여 API 서비스로 배포. EC2 인스턴스에서 Flask 등을 사용하여 모델을 호스팅하거나, Lambda를 사용하여 서버리스 아키텍처를 구성할 수 있음
* MLflow의 모델을 직접적으로 배포하거나, MLflow에서 생성된 아티팩트를 사용하여 다른 서버나 클라우드 서비스에 배포할 수 있음. 모델 파일을 서버에 업로드하고 Flask 같은 웹 프레임워크를 사용하여 API로 제공할 수 있음
* 모니터링: 모델 성능을 지속적으로 모니터링. AWS CloudWatch를 사용하여 모델의 성능 지표를 모니터링하고, 성능 저하가 감지되면 알림을 설정

=> MLflow의 'Models' 탭에서 모델을 선택하고 'Serving' 옵션을 사용하여 모델을 REST API/FLASK 등으로 배포할 수 있습니다. 이 API를 통해 어플리케이션이 모델을 사용하여 예측을 수행할 수 있음

### **4. 웹 설계 (Streamlit)**

* Streamlit: EC2 또는 Lambda에서 호스팅하는 모델을 이용해 Streamlit을 구축. S3에서 데이터를 불러와 사용자에게 시각화를 제공하며, 사용자의 입력을 받아 실시간으로 예측 결과를 보여줄 수 있음.
* AWS Route 53 또는 AWS Elastic Beanstalk: Streamlit을 인터넷에 배포하고, 사용자가 접근할 수 있도록 관리

### **+) 모델 배포와 MLflow 피클 파일 사용하기**

모델을 배포하는 방식은 다양하며, AWS SageMaker를 사용하지 않고도 MLflow에서 저장된 피클(.pkl) 파일을 사용하여 모델을 로드하고 사용할 수 있음. 그리고 이 파일을 Streamlit 애플리케이션과 함께 사용하여 예측 서비스를 구현할 수 있음.

그리고 반드시 Google Colab에서 작성한 .ipynb 파일을 AWS로 옮길 필요 없음. 대신, Colab에서 개발한 모델을 MLflow를 통해 관리하고, 생성된 모델 아티팩트(예: 모델 파라미터 파일 등)만 AWS 상에 배포

* 모델 로딩: Streamlit 앱에서 MLflow로부터 저장된 피클 파일을 로드
* 예측 수행: 사용자 입력에 대해 실시간으로 예측을 수행하고 결과를 반환

[ 모델 개발 및 실험관리 ]

* Google Colab에서 모델 개발, 모델 트레이닝, MLflow를 사용하여 실험 기록 및 관리
* Colab에서 MLflow를 설정하여 모델 파라미터, 메트릭, 아티팩트 등을 로깅. 이 로깅은 MLflow가 호스팅되는 서버(예: AWS EC2 인스턴스에서 실행되는 MLflow 서버)로 전송될 수 있음

[ 모델 아티팩트 저장 ]

* MLflow는 모델과 관련 파일들을 AWS S3 버킷에 저장. 이것은 MLflow의 설정을 통해 자동으로 이루어짐. MLflow의 Artifact Store로 S3 버킷을 지정하면, 모든 아티팩트 파일이 해당 버킷에 저장됨

[ 모델 배포 ]

* 저장된 모델 아티팩트를 AWS의 배포 대상 서비스(예: AWS Lambda, AWS SageMaker, AWS EC2 등)에 로드하여 사용. 예를 들어, AWS Lambda를 사용하여 모델을 서버리스 환경에서 실행하거나, EC2 인스턴스에서 Flask 애플리케이션을 통해 모델을 API로 제공

[ 웹 설계 ]

* Streamlit 앱을 AWS EC2 인스턴스나 Elastic Beanstalk 등에 배포하여 사용자가 모델을 사용. 이 앱은 필요한 경우 S3에서 추가 데이터를 로드하거나, 모델을 사용하여 결과를 제공

### **++) CI/CD 및 자동화의 포함 여부 결정**

\*\*\* 포함 시 \*\*\*

1. 자동화된 배포: 모델 업데이트가 필요할 때마다 수동으로 배포하는 대신, CI/CD 파이프라인을 통해 자동으로 배포할 수 있습니다. 이는 시간을 절약하고 인간의 실수를 줄일 수 있습니다.
2. 표준화된 테스트 프로세스: 코드가 모든 테스트를 통과했을 때만 배포되므로, 코드의 품질을 일관되게 유지할 수 있습니다.
3. 더 빠른 반복 개발: 소스 코드의 변경 사항을 빠르게 반영하고 테스트할 수 있어 개발 프로세스가 더욱 민첩해집니다.

\*\*\* 생략 시 \*\*\*

* 기술적 복잡성: CI/CD를 설정하고 유지 관리하는 것은 AWS를 처음 사용하는 사람들에게는 다소 어려울 수 있습니다. 이를 관리하기 위한 지속적인 노력과 기술적 이해가 필요합니다.
* 비용: AWS CodePipeline과 AWS CodeBuild는 사용량에 따라 비용이 발생합니다. 프로젝트의 예산이 제한적이라면 초기 단계에서는 이 부분을 생략하고 필요에 따라 나중에 추가하는 것도 하나의 방법입니다.

\*\*\* 추천 \*\*\*

* 시작은 간소하게: 처음에는 CI/CD 파이프라인 없이 프로젝트를 시작하고, 기본적인 MLOps 파이프라인에 익숙해지면 그 다음 단계로 CI/CD를 통합하는 것을 고려해보세요.
* 점진적 통합: 프로젝트의 규모가 확장되거나 팀이 AWS 서비스에 더 익숙해지면, 자동화와 효율성을 높이기 위해 CI/CD를 도입할 수 있습니다.

# **[ 자동화된 파이프라인 설계 ]**

### 1. 데이터 변경 감지:

AWS S3: AWS Lambda와 같은 서비스를 사용하여 S3 버킷에 새로운 데이터가 업로드되거나 기존 데이터가 변경되었을 때 이벤트를 트리거합니다. 이를 통해 데이터 변동을 자동으로 감지할 수 있습니다.

### 2. 데이터 전처리:

AWS Lambda/ AWS Glue: 데이터가 변경되면 자동으로 실행되는 ETL 작업을 통해 데이터를 전처리합니다. 이러한 전처리 작업은 새로운 데이터에 대해 필요한 변환을 수행하고, 결과를 다시 S3에 저장합니다.

### 3. 모델 재학습:

AWS SageMaker: 전처리된 데이터를 사용하여 모델을 자동으로 재학습할 수 있습니다. SageMaker의 자동화 기능을 사용하여 새 데이터로 모델을 주기적으로 업데이트하거나 특정 조건을 충족할 때 재학습을 트리거할 수 있습니다.

### 4. 모델 평가 및 배포:

모델이 재학습되면 새로운 데이터셋에 대한 성능을 평가하고, 특정 성능 기준을 만족하는 경우 자동으로 프로덕션 환경에 배포됩니다. 이 과정은 AWS CodePipeline과 같은 CI/CD 도구를 사용하여 자동화할 수 있습니다.

### 5. 모니터링 및 알림:

AWS CloudWatch: 모델의 성능을 모니터링하고 잠재적 문제에 대해 알림을 받을 수 있습니다. 성능이 기준치 아래로 떨어지면 재학습이나 추가 조치가 필요하다는 알림을 설정할 수 있습니다.

### 

### **[ AWS Glue ]**

AWS Glue는 관리형 ETL(Extract, Transform, Load) 서비스로, 데이터 통합 작업을 간편하게 설정하고 관리할 수 있습니다. AWS Glue를 사용하여 데이터를 자동으로 로드, 변환 및 S3에 저장할 수 있습니다. Glue는 대규모 데이터셋 처리에 적합하며, AWS의 다른 데이터 서비스와 원활하게 통합됩니다.

* 전처리 코드 작성: AWS Glue에서는 PySpark(Python 기반의 Spark) 스크립트를 작성하여 데이터 전처리 로직을 구현할 수 있습니다. Colab에서 사용한 Python 코드를 기반으로 PySpark 코드로 변환하는 작업이 필요할 수 있습니다.
* 자동화: Glue는 데이터 소스 변경을 감지할 수 있는 트리거를 설정하여 데이터가 변경될 때마다 자동으로 ETL 작업을 수행하도록 구성할 수 있습니다.

### **[ AWS Lambda ]**

AWS Lambda는 서버리스 컴퓨팅 서비스로, 코드를 실행하기 위한 서버 관리의 필요성 없이 코드 실행을 자동화할 수 있습니다. Lambda는 작은 규모의 데이터 처리나 간단한 전처리 작업에 적합합니다.

* 전처리 코드 작성: AWS Lambda에서는 Python, Node.js 등 다양한 프로그래밍 언어를 사용하여 함수를 작성할 수 있습니다. Colab에서 개발한 Python 전처리 코드를 Lambda 함수로 쉽게 포팅할 수 있습니다.
* 트리거 설정: S3 버킷에 새로운 데이터가 업로드되면 Lambda 함수가 자동으로 트리거되어 데이터를 처리하고 결과를 다시 S3에 저장하는 구조를 만들 수 있습니다.

### **[ 전처리 코드의 이전 ]**

* Colab에서 작성한 Python 코드가 있다면, 이 코드를 AWS Glue나 Lambda에서 실행할 수 있도록 적절히 조정해야 합니다. 주로 라이브러리 호환성 확인, 데이터 로딩 및 저장 방식의 차이를 고려해야 하며, AWS 환경에 최적화된 코드로 재작성할 수 있습니다.

AWS Glue와 Lambda를 사용하면 Colab에서 개발한 전처리 로직을 클라우드 환경에서 확장하여 실행할 수 있으며, 데이터가 변경될 때마다 자동으로 처리하도록 설정할 수 있습니다. 이는 데이터 처리의 자동화와 효율성을 크게 향상시키는 방법입니다. 특히, 전처리 작업을 자동화하기 위해 AWS Lambda에서 S3 이벤트 트리거를 설정할 수 있습니다. 이는 새 파일이 업로드될 때마다 자동으로 데이터 전처리 함수를 실행합니다.

AWS Glue는 대용량 데이터 처리에 적합하며, 데이터 카탈로그 기능을 통해 데이터 관리를 용이하게 합니다. AWS Lambda는 더 작은 규모의 데이터나 간단한 변환에 적합합니다. 사용 사례에 따라 적절한 서비스를 선택하세요.

### **[ 논의 필요 사항 ]**

### **1. AWS + MLFLOW 통합 사용했을 때 “2. 모델 개발 및 실험 관리”를 Colab+mlflow로 한다면 자동화 요소가 아닌 수동 요소가 개입됨**

⇒ AWS에서 직접 Jupyter 환경을 구축 가능하긴 함.. Amazon SageMaker는 Jupyter 노트북을 지원하며, S3에서 직접 데이터를 불러와 모델을 학습하고, MLflow와 통합하여 실험 관리를 자동화함

=> MLflow와 S3 통합: MLflow를 설정하여 모델 아티팩트와 로그를 AWS S3에 저장하도록 하기

=> AWS Lambda/EC2를 이용한 모델 배포: MLflow에서 관리하는 모델 아티팩트를 사용하여 모델을 AWS Lambda 또는 EC2 인스턴스에 배포. 이 인스턴스는 모델을 로드하여 REST API 형태로 서비스를 제공함

============>

### **2. 모델 개발 및 실험 관리 자동화 전략**

1. AWS 내에서 자동화 구현
   * AWS SageMaker: 이 서비스를 사용하지 않는다면, AWS 내에서 제공하는 다른 서비스를 활용할 수 있습니다. 예를 들어, AWS Lambda를 사용하여 데이터 전처리가 완료되면 자동으로 트리거되는 Python 스크립트를 실행할 수 있습니다. 이 스크립트는 훈련 코드를 실행하고 MLflow에 결과를 로그할 수 있습니다.
   * EC2 인스턴스: AWS EC2 인스턴스에서 Jupyter Notebook 서버를 설정하고, 필요한 라이브러리와 MLflow를 설치하여 사용할 수 있습니다. 이 서버는 데이터 전처리가 완료되면 자동으로 모델을 훈련시키고, 결과를 MLflow로 로그하는 스크립트를 실행할 수 있습니다.
2. CI/CD 파이프라인 활용
   * AWS CodePipeline과 AWS CodeBuild: 이 도구들을 사용하여 모델 훈련 코드의 변경 사항을 자동으로 감지하고, 새 코드로 모델을 재학습시킬 수 있습니다. 데이터가 변경되었을 때 훈련 작업을 자동으로 시작하는 트리거를 설정할 수 있습니다.
3. Docker 컨테이너 사용
   * AWS ECS 또는 EKS: Docker 컨테이너 내에서 모델 훈련 코드를 실행하고, 컨테이너를 AWS ECS(Elastic Container Service) 또는 EKS(Elastic Kubernetes Service)에 배포할 수 있습니다. 데이터 전처리 후 자동으로 컨테이너가 모델 학습을 시작하도록 설정할 수 있습니다.
4. MLflow와의 통합
   * MLflow 서버: MLflow 서버를 EC2 인스턴스나 ECS/EKS 서비스 위에 설정하여, 모든 학습 로그와 아티팩트를 중앙 집중식으로 관리할 수 있습니다. 이 서버는 모델 버전, 메트릭, 파라미터 등을 추적하고, 아티팩트를 S3에 저장합니다..

### **3. AWS EC2에서 Jupyter Notebook 서버 설정 및 사용 방법**

1. EC2 인스턴스 시작: AWS Management Console에서 EC2 인스턴스를 시작합니다. 여기서는 계산 리소스가 필요한 머신러닝 작업에 적합한 인스턴스 타입을 선택해야 합니다.
2. Jupyter Notebook 설치 및 설정:
   * 인스턴스에 SSH 접속 후 필요한 소프트웨어(예: Python, Jupyter)를 설치합니다.
   * Jupyter Notebook (또는 JupyterLab)을 설정하여 원격에서 접속할 수 있도록 합니다. 이를 위해 Jupyter Notebook 서버를 백그라운드에서 실행하고, 보안 설정을 통해 인터넷에서 접근 가능하게 해야 합니다.
   * 필요한 라이브러리(예: Pandas, NumPy, Scikit-learn, TensorFlow, PyTorch 등)와 MLflow를 설치합니다.
3. 코드 업로드 및 자동 실행 설정:
   * Colab에서 작성한 .ipynb 파일을 EC2 인스턴스에 업로드합니다.
   * 필요한 경우, 데이터 소스로부터 데이터를 자동으로 불러오고 처리할 수 있는 스크립트를 작성합니다.
   * MLflow와 통합하여 모델의 파라미터, 메트릭, 아티팩트 등을 자동으로 기록하도록 설정합니다.
   * 데이터 전처리 완료 후 모델 학습을 자동으로 시작할 수 있도록 Cron 작업이나 AWS Lambda 함수를 설정합니다.
4. 모델 훈련 및 자동화:
   * EC2 인스턴스에서 자동으로 모델을 훈련하고 결과를 MLflow에 로그합니다.
   * 학습 완료 후 MLflow를 통해 모델을 평가, 선택 및 배포합니다.
5. 모니터링 및 유지보수:
   * AWS CloudWatch를 사용하여 인스턴스와 Jupyter Notebook 서버의 성능을 모니터링합니다.
   * 정기적으로 시스템을 검토하고 필요에 따라 업데이트하거나 확장합니다.

MLflow 서버가 AWS EC2 인스턴스에 설정되어 있다면, 그 인스턴스에서 직접 모델 서빙이 가능함!!

1. MLflow Models

MLflow의 모델 서빙 기능은 MLflow Models 컴포넌트를 통해 제공됩니다. 이를 통해 모델을 REST API 서비스로 배포할 수 있으며, 이 API를 통해 예측을 수행할 수 있습니다.

* MLflow Model Serving: MLflow 모델 서빙은 간단한 명령어로 MLflow 모델을 로드하고, HTTP 서버를 시작하여 모델을 REST API 형태로 제공합니다. 이는 MLflow의 mlflow models serve 명령을 사용하여 설정할 수 있습니다.
* EC2 인스턴스 사용: EC2 인스턴스에서 이 명령을 실행하면, 인스턴스의 특정 포트에서 모델 서빙 서비스가 실행됩니다. 인스턴스의 보안 그룹 설정을 통해 해당 포트(기본적으로 5000 포트)에 대한 접근을 허용해야 합니다.

2. Containerization and Deployment

모델을 더욱 견고하게 배포하려면, Docker 컨테이너를 사용한 배포를 고려할 수 있습니다. MLflow는 모델을 Docker 이미지로 패키징하고, 이 이미지를 Amazon ECS (Elastic Container Service) 또는 EKS (Elastic Kubernetes Service)와 같은 컨테이너 오케스트레이션 서비스에 배포하는 기능을 지원합니다.

* Docker Image: MLflow는 mlflow models build-docker 명령을 통해 모델을 Docker 이미지로 패키징할 수 있습니다. 이 이미지는 모든 종속성과 함께 모델을 포함하므로, 어떤 환경에서도 일관된 방식으로 모델을 실행할 수 있습니다.
* AWS ECS/EKS 배포: 생성된 Docker 이미지를 AWS의 컨테이너 서비스에 배포하여 모델을 서빙할 수 있습니다. 이 방법은 확장성이 높고, 관리가 용이하며, 보다 복잡한 프로덕션 환경에 적합합니다.

모니터링

모델을 서빙한 후에는 성능 모니터링과 로깅이 중요합니다. AWS CloudWatch와 같은 서비스를 사용하여 서빙 인프라의 상태를 모니터링하고, MLflow를 통해 로깅된 데이터를 분석할 수 있습니다.

### **4. 모니터링 및 Streamlit 통합**

AWS CloudWatch를 사용하여 서빙 인프라의 상태를 모니터링하고, MLflow로 로깅된 데이터를 분석하는 것은 가능합니다. 이 데이터를 Streamlit 앱에서 시각화하여 볼 수 있는지에 대한 질문에 대한 답은 '예'입니다. 다음 단계를 통해 이를 구현할 수 있습니다:

1. CloudWatch 로그 수집:
   * 서버 상태, 애플리케이션 로그, 사용 메트릭 등을 CloudWatch로 수집합니다. 이 정보는 시스템의 건강 상태를 모니터링하는 데 사용됩니다.
2. MLflow 데이터 액세스:
   * MLflow는 실험, 모델 학습 결과, 메트릭을 저장합니다. 이 데이터는 REST API 또는 MLflow의 Python 라이브러리를 통해 접근할 수 있습니다.
3. Streamlit 애플리케이션 구축:
   * Streamlit을 사용하여 CloudWatch와 MLflow에서 수집한 데이터를 시각화하는 웹 애플리케이션을 구축합니다. Streamlit 앱은 이 데이터를 불러와 다양한 차트와 그래프로 표시할 수 있습니다.
   * 예를 들어, 모델 성능의 변화, 시스템 로드, API 호출 수 등을 실시간으로 시각화하여 사용자가 쉽게 분석할 수 있도록 합니다.
4. 데이터 통합 및 액세스:
   * Streamlit 앱에서 필요한 데이터를 API 호출을 통해 불러오거나, AWS SDK(Software Development Kit)를 사용하여 CloudWatch 데이터에 직접 액세스할 수 있습니다.

### **5. 계정 접근 권한**

EC2 인스턴스를 설정하거나, S3 버킷을 관리하고, Lambda 함수를 생성 및 구성하려면 적절한 권한이 필요합니다. 여러분의 상황에서 가능한 접근 방법은 다음과 같습니다:

1. 계정 접근 권한 요청

* 관리자에게 접근 요청: 계정을 관리하고 있는 사람 또는 팀에게 필요한 작업과 목적을 설명하고, 필요한 서비스(예: EC2, S3, Lambda 등)에 대한 접근 권한을 요청하세요. 이는 IAM(Identity and Access Management)을 통해 구체적인 권한을 설정하여 진행할 수 있습니다.

2. IAM 사용자 생성 및 권한 관리

* IAM 사용자로 작업: AWS 계정의 소유자가 여러분에게 별도의 IAM 사용자 계정을 생성하고, 필요한 권한을 부여할 수 있습니다. 이 방법은 보안을 유지하면서 필요한 서비스에 대한 접근을 허용할 수 있는 좋은 방법입니다.
* 권한 제한: 최소 권한 원칙에 따라 필요한 서비스에 대해서만 접근을 허용받도록 설정하는 것이 좋습니다. 이는 보안 사고를 방지하고 관리를 용이하게 합니다.

3. 협업 도구 사용

* AWS Organizations: 대규모 조직의 경우, AWS Organizations를 사용하여 계정을 관리하고 정책을 통해 거버넌스를 적용할 수 있습니다. 이 도구를 사용하면 여러 AWS 계정의 리소스와 사용자를 효율적으로 관리할 수 있습니다.