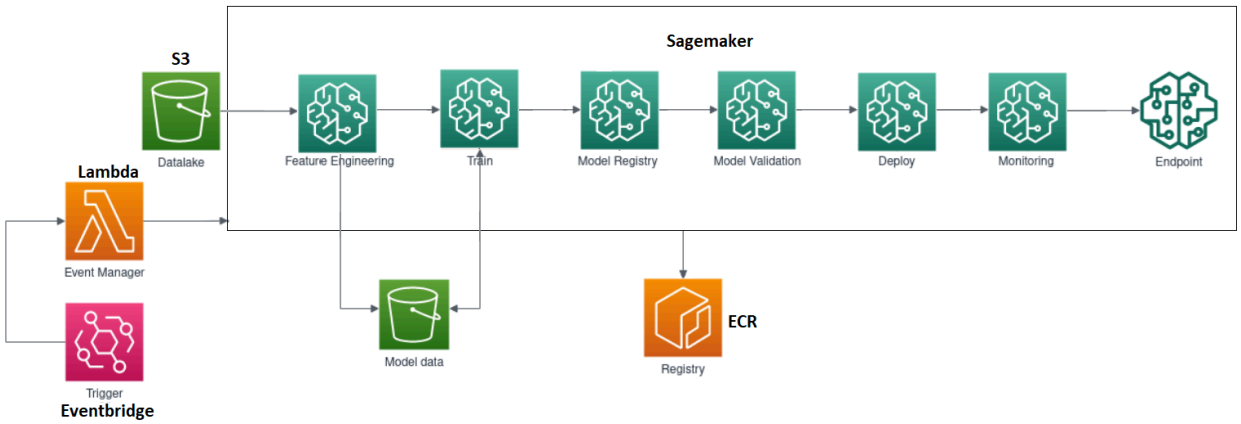
The chosen cloud provider is AWS, due to its flexibility and variety of services.
The designed pipeline consists of a combination of services:

- S3 for data storage.
- Sagemaker for ML specific steps (training, deployment, etc.). It allows to easily scale training and inference resources in relation to the incoming volume, so that if the number of users increases substantially in the future, it won't be a problem.
- Eventbridge for performance monitoring.
- Lambda to trigger pipeline execution.
- ECR to store Docker images.

Pipeline schema:



1. **Data Ingestion and Storage**: set up a data lake on S3, where incoming raw data is stored.
2. **Data Preprocessing and Feature Engineering**: read the latest data from the data lake, and use SageMaker Processing to preprocess it and engineer features. Output processed features are stored in S3.
3. **Model Training**: read processed features and train the ML models using SageMaker training jobs. Resulting model weights are stored in the model registry.
4. **Model registry and Model validation**: Sagemaker Model Registry is useful to manage model versions, keep metadata associated with each model and automate model deployment with CI/CD, for example. The user can evaluate and compare different versions, and manually validate which one will be deployed.
5. **Model Deployment**: deploy trained models using SageMaker endpoints, which can serve both real-time and batch inference. The deployed models can be accessed through a REST API. An important feature supported by Sagemaker is A/B testing, which allows the deployment of a new model while keeping the current one, to test whether the new model does indeed perform better.
6. **Monitoring and Management**: monitor the performance of the deployed models using Eventbridge. I can be used to monitor the input data distribution, and raise an alarm in case it changes significantly. Or, in a more complex setup, it could be used to monitor if model performance degrades below a certain threshold. In any case, if the alarm is raised, it calls a Lambda function that launches a new pipeline run using the latest data available (which must be previously annotated in this case).

Each step can have its own Docker image. Once built, they can be stored in ECR (Elastic Container Registry) to be accessed later.