William Chang Liu - 1009048852

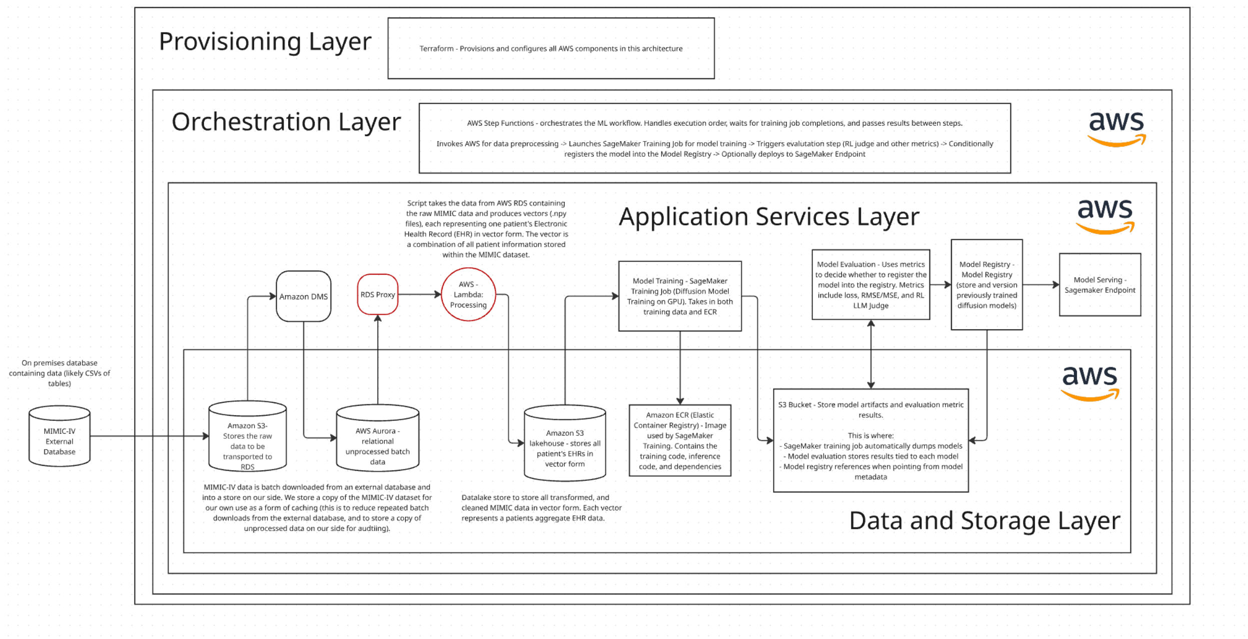
Ryan Zhang - 1009020453

Taiyi Jin - 1009075796

Abdus Shaikh - 1007288533

Assignment 3: Deleting Prod

Part 1: Diagramming Infrastructure


Provisioning Layer
Orchestration Layer
Application Services Layer
Data and Storage Layer
aws
aws
aws
On premises database containing data (likely CSVs of tables)
MIMIC-IV External Database
Amazon S3- Stores the raw data to be transported to RDS
Amazon DMS
AWS Aurora - relational unprocessed batch data
RDS Proxy
AWS - Lambda: Processing
Amazon S3 lakehouse - stores all patient's EHRs in vector form
Model Training - SageMaker Training Job (Diffusion Model Training on GPU). Takes in both training data and ECR
Amazon ECR (Elastic Container Registry) - Image used by SageMaker Training. Contains the training code, inference code, and dependencies
Model Evaluation - Uses metrics to decide whether to register the model into the registry. Metrics include loss, RMSE/MSE, and RL LLM Judge
Model Registry - Model Registry (store and version previously trained diffusion models)
Model Serving - Sagemaker Endpoint

Terraform - Provisions and configures all AWS components in this architecture

AWS Step Functions - orchestrates the ML workflow. Handles execution order, waits for training job completions, and passes results between steps.

Invokes AWS for data preprocessing -> Launches SageMaker Training Job for model training -> Triggers evalutation step (RL judge and other metrics) -> Conditionally registers the model into the Model Registry -> Optionally deploys to SageMaker Endpoint

Script takes the data from AWS RDS containing the raw MIMIC data and produces vectors (.npy files), each representing one patient's Electronic Health Record (EHR) in vector form. The vector is a combination of all patient information stored within the MIMIC dataset.

S3 Bucket - Store model artifacts and evaluation metric results.

This is where:
- SageMaker training job automatically dumps models
- Model evaluation stores results tied to each model
- Model registry references when pointing from model metadata

MIMIC-IV data is batch downloaded from an external database and into a store on our side. We store a copy of the MIMIC-IV dataset for our own use as a form of caching (this is to reduce repeated batch downloads from the external database, and to store a copy of unprocessed data on our side for audtiing).

Datalake store to store all transformed, and cleaned MIMIC data in vector form. Each vector represents a patients aggregate EHR data.

Miro Link:
https://miro.com/app/board/uXjVJDh--SI=/?share_link_id=501292272666

The components highlighted in red will be implemented in the future. This was decided because our data processing infrastructure is subject to the most change as we try to tackle ways to add a temporal dimension to our model. Furthermore, the data format of the EHRDiff model, which we used as a placeholder, is not compatible with the current output of the data-processing script. As a result, the data-ingestion pipeline cannot be connected to the model serving and training pipeline at the moment.

Given that the group had access to only AWS free-tier for the majority of the development process some compromises had to be made. Namely, the model training algorithm is currently cpu-only as the group did not have access to EC2 instances with GPUs. Creating a good model necessitates GPU access and more work will be done to create a deployable GPU training script in the future.

Since this is a research-oriented project and we have no client-facing application, we do not have separate environments for development and production. The decision to have one simple data-processing to model training and evaluation environment was made because it allows changes to be made and applied more quickly.

Part 2: Infrastructure as Code

Terraform was used as the primary framework for Infrastructure as code. No scripts were used and all infrastructure is defined declaratively. All of the terraform code can be found under a3/terraform. a3/terraform/model_train_serve contains all of the infrastructure for model training and serving. a3/terraform/data_ingestion contains the work-in-progress code for our data processing pipeline.

Several pieces of open-source code were modified and used to create our infrastructure. Firstly, an official AWS blog by Oliver Zollikofer provided us with a guide as well as a repository to starter code that served as the foundation for our model training and serving IaC. Secondly, our demonstration model was based on EHRDiff created by Yuan et al.

Part 3: Deleting Production

The video for deleting production can be found on this google drive link: a3_demo_video.mp4 .
It will also be included in the quercus submission.

References

Yuan, H., Zhou, S., & Yu, S. (2023). *EHRDiff: Exploring realistic EHR synthesis with diffusion models* (arXiv preprint arXiv:2303.05656). https://doi.org/10.48550/arXiv.2303.05656

Yuan, H., Zhou, S., & Yu, S. (2023). *EHRDiff* [Source code]. https://github.com/sczzz3/EHRDiff

Zollikofer, O. (2022, May 4). *Deploy and manage machine learning pipelines with Terraform using Amazon SageMaker*. AWS Machine Learning Blog. https://aws.amazon.com/blogs/machine-learning/deploy-and-manage-machine-learning-pipelines-with-terraform-using-amazon-sagemaker/

Zollikofer, O. (2022, May 4). *amazon-sagemaker-ml-pipeline-deploy-with-terraform* [Source code]. GitHub. https://github.com/aws-samples/amazon-sagemaker-ml-pipeline-deploy-with-terraform