TEC 5900 Summer 2025: Team 6 Project Handoff

Integrating AI and Cloud Infrastructure for GLUU

Table of Contents

Project Introduction

Gluu is in the process of enhancing its technological infrastructure to streamline operations and improve asset management. The project involves updating an Amazon EC2 instance to support the integration of advanced AI models and cloud services.

This document details the handoff of the project from the Planning/Project Management part of Team 6 to the Development Team. The project managers will still be involved but the bulk of the work will be completed by the development team. The table below outlines the general team assignments.

Team 6 is composed of:

Team Member	Project Work Area
James Scott	Enterprise Networks Cloud Computing
Joel Palmateer	Enterprise Networks Cloud Computing
Jonathan Siriano	General Information Technology
Chukwudi Chukwu	Enterprise Networks Cloud Computing
Cledonat Francois	Project Management
Karen Elliott	Project Management

Project Goal

This project will apply data analytics methodologies and backend API development practices to design and build the KPI data pipeline l. Students will deliver a fully integrated solution that provides actionable insights based on training data, model performance metrics, and operational efficiency, enabling continuous improvement of the core machine learning asset.

Project Scope

During the initial work of this project, Team 6 and Team Gluu updated the initial project scope to the following:

The scope of this project is focused on changes to Gluu’s current system to improve efficiency and security in image recognition and processing. The primary task of the project is to integrate the Dify Bot into Supabase for efficient data management and Vercel for a responsive front-end interface.

This integration will facilitate real-time learning and condition reporting for Gluu's real-world assets, leveraging a Multi-Modal AI model composed of Blip 2 model, GPT 4oD 40, and various AWS services.

Reducing costs by triggering the model only upon usage and implementing strategies like auto shutdown of SageMaker and multi-model deployment under one instance. Key performance indicators will include high availability and high accuracy from the model

Other key tasks include configuring the EC2 instance, setting up the Dify Bot, and ensuring seamless communication between Supabase and Vercel.

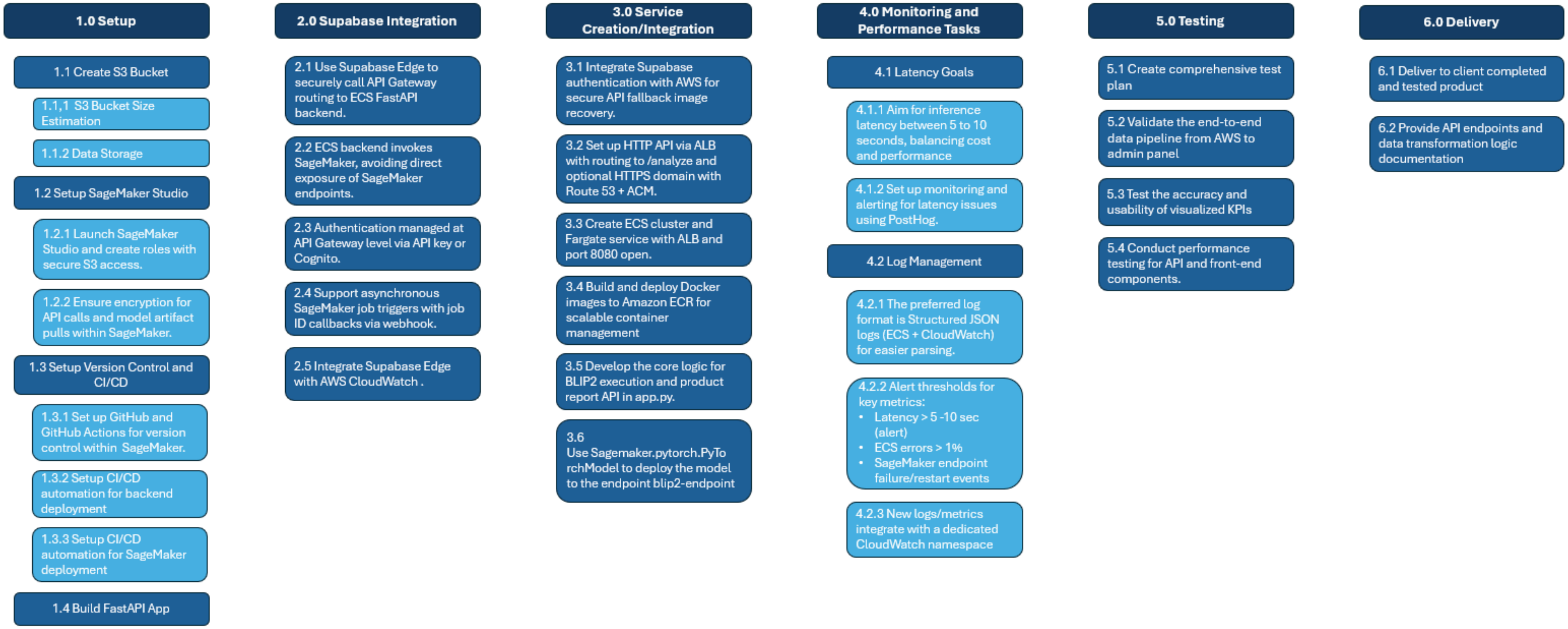
Work Completed

Team 6 completed 2 sprints of the first Work Breakdown Structure. Those sprints included extensive research on the tools in the existing systems, client’s goals and well as outlining, in great detail, the development work to be completed in the next quarter.

Work Breakdown Structure (WBS)

After an intensive Question and Answer document exchange, an updated WBS was created. The first section displays the WBS diagram and then is followed by a detailed breakdown of the tasks as well as some important information to consider when creating deliverables for the project.WBS Diagram

Integrating AI and Cloud Infrastructure Work Breakdown Structure



- **1.0 Setup**
  - 1.1 Create S3 Bucket
    - 1.1,1 S3 Bucket Size Estimation
    - 1.1.2 Data Storage
  - 1.2 Setup SageMaker Studio
    - 1.2.1 Launch SageMaker Studio and create roles with secure S3 access.
    - 1.2.2 Ensure encryption for API calls and model artifact pulls within SageMaker.
  - 1.3 Setup Version Control and CI/CD
    - 1.3.1 Set up GitHub and GitHub Actions for version control within SageMaker.
    - 1.3.2 Setup CI/CD automation for backend deployment
    - 1.3.3 Setup CI/CD automation for SageMaker deployment
  - 1.4 Build FastAPI App
- **2.0 Supabase Integration**
  - 2.1 Use Supabase Edge to securely call API Gateway routing to ECS FastAPI backend.
  - 2.2 ECS backend invokes SageMaker, avoiding direct exposure of SageMaker endpoints.
  - 2.3 Authentication managed at API Gateway level via API key or Cognito.
  - 2.4 Support asynchronous SageMaker job triggers with job ID callbacks via webhook.
  - 2.5 Integrate Supabase Edge with AWS CloudWatch .
- **3.0 Service Creation/Integration**
  - 3.1 Integrate Supabase authentication with AWS for secure API fallback image recovery.
  - 3.2 Set up HTTP API via ALB with routing to /analyze and optional HTTPS domain with Route 53 + ACM.
  - 3.3 Create ECS cluster and Fargate service with ALB and port 8080 open.
  - 3.4 Build and deploy Docker images to Amazon ECR for scalable container management
  - 3.5 Develop the core logic for BLIP2 execution and product report API in app.py.
  - 3.6 Use Sagemaker.pytorch.PyTorchModel to deploy the model to the endpoint blip2-endpoint
- **4.0 Monitoring and Performance Tasks**
  - 4.1 Latency Goals
    - 4.1.1 Aim for inference latency between 5 to 10 seconds, balancing cost and performance
    - 4.1.2 Set up monitoring and alerting for latency issues using PostHog.
  - 4.2 Log Management
    - 4.2.1 The preferred log format is Structured JSON logs (ECS + CloudWatch) for easier parsing.
    - 4.2.2 Alert thresholds for key metrics:
      - Latency > 5 -10 sec (alert)
      - ECS errors > 1%
      - SageMaker endpoint failure/restart events
    - 4.2.3 New logs/metrics integrate with a dedicated CloudWatch namespace
- **5.0 Testing**
  - 5.1 Create comprehensive test plan
  - 5.2 Validate the end-to-end data pipeline from AWS to admin panel
  - 5.3 Test the accuracy and usability of visualized KPIs
  - 5.4 Conduct performance testing for API and front-end components.
- **6.0 Delivery**
  - 6.1 Deliver to client completed and tested product
  - 6.2 Provide API endpoints and data transformation logic documentation

WBS Section Details/Considerations:

1.0 Setup

1.1 Create S3 Bucket

1.1.1 S3 Bucket Size Estimation

Begin with one region US East (N. Virginia) closest to Canada (same as SageMaker + ECS) for low latency. Multi-region replication is required at later stages and is optional at project beginning. We could use cross-region replication for disaster recovery or compliance, but this is not required at baseline.

1.1.2 Data Storage

Data storage will be small, typically temporary images less than 100 GB in size and model artifacts. The preferred data should be private and restricted to ECS and SageMaker roles. Data retention is short term. Enable S3 lifecycle policies to auto delete temporary inference images as a cost saving measure. Consider an EOD previous 24-hour cleanup of data. S3 storage should integrate with SageMaker (model artifacts) for SageMaker inference and ECS (async outputs)

Other Considerations

- Enable S3□ EventBridge for async inference completion.
- Data Encryption is required – default **S3 SSE (AES-256)** + HTTPS for transit
- Versioning for data changes is not necessary for transient inference files; however, it could be enabled for model artifacts

1.2 Setup SageMaker Studio:

- Launch Sage Maker Studio
- Create a role with S3 access
- Ensure data encryption for API calls and model pulling to SageMaker
- Upload blip-2 endpoint tar.gz to the S3 bucket

1.3 Setup Version Control & CI/CD

- Set up GitHub and GitHub Actions for version control
- Setup CI/CD automation for backend deployment
- Setup CI/CD automation for SageMaker deployment

1.4 Build FastAPI App

Develop the FastAPI app to handle file uploads, call SageMaker, format OpenAI prompts, and return structured JSON responses.

2.0 Supabase Integration

2.1 Use Supabase Edge to call **API Gateway**, which routes to the **ECS FastAPI backend**. The backend then invokes SageMaker. This avoids exposing SageMaker directly and provides authentication, throttling, and monitoring.

2.2 Authentication is handled at the **API Gateway level** (API key or Cognito). ECS uses **IAM task roles** to access SageMaker and S3. NOTE: Supabase does not directly assume AWS roles.

2.3 GitHub README can include usage examples, but in the current design results are returned as a **JSON response from the ECS FastAPI app.** If Supabase storage is needed, integration can be handled at the front end (store API response in Supabase).

2.4 Supabase Edge can be used to trigger SageMaker jobs asynchronously**.** ECS can return a job ID immediately, and results can later be fetched or pushed to Supabase via webhook.

2.5 Integrate Supabase Edge with AWS CloudWatch with focus on asynchronous handling of logs and the impact on performance.

3.0 Service Creation/Integration

3.1 Integrate Superbase authentication with AWS services, ensuring secure API calls. This integration is required as a fallback to recover images in case they are lost or not stored in the S3 bucket during the API call event.

3.2 Create an HTTP API with ALB as integration, route POST requests to /analyze, and optionally connect Route 53 + ACM for HTTPS domain

3.3 Create an ECS cluster and Fargate service, attach an ALB, and open port 8080.

3.4 Create a Dockerfile and requirements.txt, build, tag, and push the Docker image to Amazon ECR.

3.5 Develop the core logic for BLIP2 execution and product report API in app.py.

3.6 Use Sagemaker.pytorch.PyTorchModel to deploy the model to the endpoint blip2-endpoint

4.0 Monitoring and Performance Tasks

4.1 Latency Goals

4.1.1 Aim for inference latency between 5 to 10 seconds, balancing cost and performance.

4.1.2 Set up monitoring and alerting for latency issues using PostHog.

Relevant KPIs:

- Inference latency (SageMaker response time)
- ECS request handling time
- API Gateway request volume & error rate
- Cost metrics (SageMaker hourly usage, ECS runtime, OpenAI API calls)

Note: The impact of traffic location of the services on latency is minimal if all services stay within the same AWS region as Supabase clients

4.2 Log Management

4.2.1 The preferred log format is Structured JSON logs (ECS + CloudWatch) for easier parsing.

4.2.2 Alert thresholds for key metrics:

Latency > 5 -10 sec (alert)

ECS errors > 1%

SageMaker endpoint failure/restart events

4.2.3 New logs/metrics integrate with a dedicated CloudWatch namespace

5.0 Testing

5.1 Create comprehensive test plan

5.2 Validate the end-to-end data pipeline from AWS to admin panel

5.3 Test the accuracy and usability of visualized KPIs

5.4 Conduct performance testing for API and front-end components.

6.0 Delivery

6.1 Deliver to client completed project

6.1 Provide API endpoints and data transformation logic documentation

Other Considerations

- Set up **CloudWatch dashboards + SNS alerts**. The Engineering team will be checking the CloudWatch data.
- Currently there is single-region redundancy, but ECS + SageMaker provide auto-restart. Multi-region failover not needed in the baseline

Next Steps

With a major update to the scope putting Team 6 a bit behind in our work, we will meet during our break to do sprint planning. This short delay will allow the development part of our team to review the WBS, come up with a preliminary sprint plan as well as determine whose skills best align to the tasks. When the entire team meets over the break, we will firm up the Sprint Schedule and assignments, then send the schedule to our client. We feel the additional work on research and refining the scope will assist us in successfully completing this project in the next quarter.

References

Amazon CloudWatch: Monitor resources and applications. AWS Documentation. (n.d.). Retrieved from Amazon Web Services: https://docs.aws.amazon.com/cloudwatch

A Simple Guide to Understanding Agile Principles. (2024, March 29). Retrieved from Smartify Software Solutions: https://smartifysol.com/embracing-agility-a-simple-guide-to-understanding-agile-principles/

Gemini, G. (2025, August 8). *Response to How Does Supabase wirk with AWS CloudWatch*. Retrieved from Google Gemini: https://www.google.com/how+does+supabase+edge+work+with+aws+cloudwatch_AI_Response

Grandi, M. (2023). *Optimizing your modernization journey with AWS. Best practices for transforming your applications and infrastructure on the cloud (1st: 1 ed.).* Packt Publishing.

Grouse, A. (2025, September 7). *Gluu| AWS| Capella University*. Retrieved from https://mail.google.com/chat/u/0/#chat/home

PutLogEvents – Amazon CloudWatch Logs. AWS Documentation. (n.d.). Retrieved from Amazon Web Services: https://docs.aws.amazon.com/AmazonCloudWatchLogs/latest/APIReference/API_PutLogEvents.html

Rajkumar, S. (2010). Art of communication in project management. *PMI® Research Conference: Defining the Future of Project Management.* Washington, DC. Newtown Square, PA: Project Management Institute.

Security best practices in IAM. AWS Documentation. (n.d.). Retrieved from Amazon Web Services: https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html

Siriano, J. (2025, August 23). *GluuTeam6Meeting [Transcript]*. Retrieved from https://docs.google.com/document/d/1ZOo0of1P25tUZ7HiIemkDYudoLfiRxqh/edit?usp=drive_link&ouid=100573224961273174963&rtpof=true&sd=true

Siriano, J. (2025, August 13). *Team 6 Week 6 Team Meeting [Transcript]*. Retrieved from docs.google.com/document/Team6August13Meeting

Siriano, J. (2025, Septmeber 3). *Team 6 Weekly Meeting (Transcript)*. Retrieved from https://docs.google.com/document/d/1kyv6gwfuTly1RsCGdhzg9eeNcSTlgwPO/edit?usp=drive_link&ouid=100573224961273174963&rtpof=true&sd=true

What is Amazon S3? (n.d.). Retrieved from Amazon Simple Storage Service User Guide: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

What is SageMaker in AWS? (2025, July 15). Retrieved from Geek for Geeks: https://www.geeksforgeeks.org/machine-learning/what-is-sagemaker-in-aws/#what-is-amazon-sagemaker