GLUU: BLIP2 Product Summary Inference Architecture & Deployment Document

Project Overview

This document outlines the architecture, components, and deployment strategy for building a scalable, inference-based system using the BLIP2 model (Salesforce/blip2-flan-t5-xl) on AWS SageMaker, with a FastAPI backend hosted on ECS, API Gateway exposure, and OpenAI integration for final text product report.

This project will be executed by a designated team using the following resources:

- **Base Code:** `app.py` — core logic for BLIP2 execution and product report API.
- **Notebook Reference:** An official AWS sample notebook for BLIP2 inference using SageMaker.
- **Additional Resources:** Relevant AWS documentation, cost estimators, and architectural references.

Project Goal

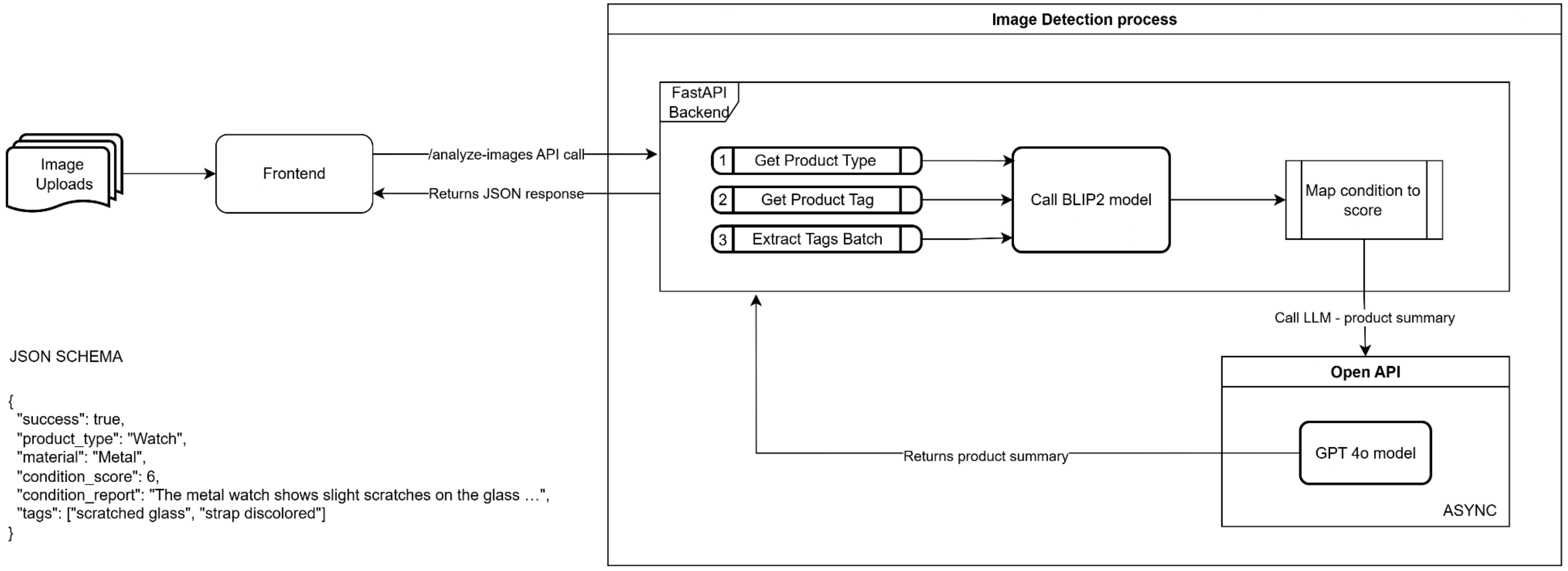
Implement the end-to-end architecture as described below, with a strong emphasis on:

- Reducing total predicted cost
- Finding cost-optimized alternatives without sacrificing core functionality

Leaving certain infrastructure choices **open-ended** for the technical team to decide during implementation, such as:

1. Whether to use a Load Balancer or API Gateway alone for routing
2. Whether to host the backend on ECS (Fargate) or Elastic Beanstalk
3. Finalizing the async vs. sync inference flow depending on performance testing
4. Whether inference is executed through a SageMaker notebook or via separate dedicated inference code in SageMaker Studio

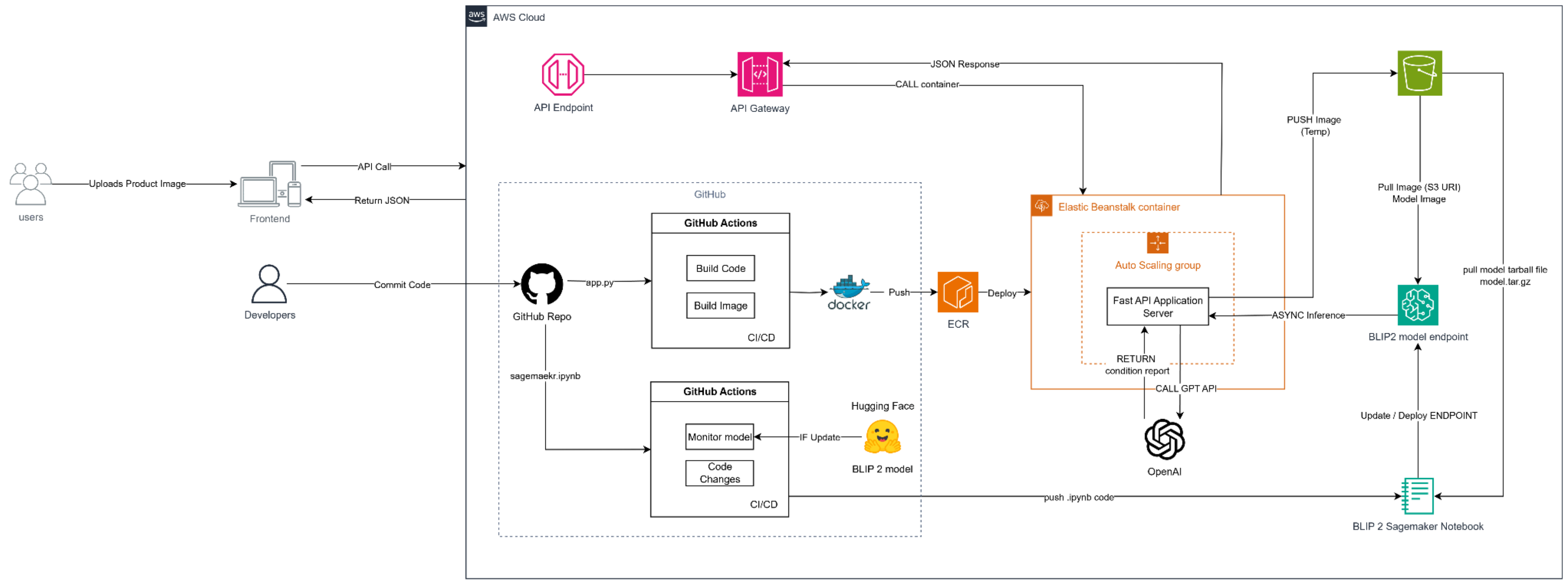
System Overview



JSON SCHEMA

```
{
  "success": true,
  "product_type": "Watch",
  "material": "Metal",
  "condition_score": 6,
  "condition_report": "The metal watch shows slight scratches on the glass …",
  "tags": ["scratched glass", "strap discolored"]
}
```

System Architecture (AWS)



Core Services (AWS and External)

Git & GitHub: Source code version control and collaboration

GitHub Actions: CI/CD automation for both backend and SageMaker deployment

Amazon ECS (Fargate) or Elastic Beanstalk: Hosts the API container in a serverless or managed environment (final choice to be determined during implementation)

Amazon ECR: Stores the container images

ALB + API Gateway: Handles HTTPS routing from client to ECS/Beanstalk (choice between ALB and direct API Gateway integration left open)

Route 53 + ACM (Optional): Custom domain and SSL management

Amazon S3: Temporary storage for uploaded images and model output (if async)

AWS SageMaker : Hosts the BLIP2 inference logic; may be executed directly via SageMaker notebook or via separate inference code in SageMaker Studio (final method open-ended)

OpenAI GPT-4: Generates product summaries from inference output

AWS Components

1. SageMaker Model Endpoint

Model: `Salesforce/blip2-flan-t5-xl`

Deployment Type: Real-time or Async Inference (Cost Saving)

Instance Type: `ml.g4dn.xlarge` (GPU)

Source Code:

- `inference.py`: Handles model loading and prediction
- `requirements.txt`: Contains dependencies (transformers, optimum, torch, etc.)
- Packaged as `blip2-endpoint.tar.gz` and uploaded to `s3://<bucket>/blip2/` via the sagemaker notebook code.

2. ECS FastAPI App (Backend API)

- Hosted in **ECS Fargate** with public ALB (Application Load Balancer) [Open Ended Requirement]
- Container built using `Dockerfile`
- Responsibilities:
  - Accept image uploads
  - Call SageMaker endpoint for inference
  - Format prompt and call OpenAI
  - Return structured JSON response

3. API Gateway

- Exposes a clean, secure REST/HTTP API to frontend or external apps
- Routes requests to ECS via ALB
- Can be enhanced with:
  - API Key
  - Lambda Authorizers or Cognito

4. OpenAI GPT-4o Integration

- Converts structured inference output into 50-word product summaries
- Model: `gpt-4o` or fallback to `gpt-4o-mini`, etc.

AWS Deployment Stages

STAGE 1: Setup SageMaker Studio & S3

- Launch Studio, create role with S3 access
- Use default or custom bucket
- Upload `blip2-endpoint.tar.gz` to `s3://<bucket>/blip2/`

STAGE 2: Deploy Model via Notebook or Inference Code File

Use `sagemaker.pytorch.PyTorchModel` and deploy to endpoint `blip2-endpoint`

STAGE 3: Build FastAPI App

- Code: `app.py`
- Handles file upload, calls SageMaker, formats OpenAI prompt
- Environment variables:
  - `SAGEMAKER_ENDPOINT=blip2-endpoint`
  - `OPENAI_API_KEY=<your-key>`

STAGE 4: Dockerize App

- `Dockerfile` and `requirements.txt`
- Build, tag, and push to Amazon ECR

STAGE 5: Deploy to ECS

- Create ECS cluster & Fargate service
- Attach ALB and open port 8080

STAGE 6: API Gateway Integration

- Create HTTP API with ALB as integration (open ended)
- Route: `POST /analyze`
- Optional: Connect Route 53 + ACM for HTTPS domain

Version Control and CI/CD

GitHub & GitHub Actions

All project code including: `app.py` (FastAPI backend), `Dockerfile`, `requirements.txt`, SageMaker notebook or inference model code is version controlled in GitHub via Git.

The CI/CD automation managed via **GitHub Actions** covers two primary components:

1. FastAPI Docker Backend (ECS Deployment)

On push to `main` or `release/*`:

1. Build Docker image from `app.py`
2. Push image to **Amazon ECR**
3. Trigger ECS service update to pull new image via AWS CLI/API

On changes to SageMaker notebook or inference code:

1. Automatic deployment to SageMaker using SageMaker Notebook Jobs or Papermill
2. Code review required before merge
3. If model updates are detected from Hugging Face, automatically retrain/redeploy the SageMaker endpoint

These pipelines ensure the entire backend and model logic remain version-controlled and automatically deployed.

SageMaker Endpoint Deployment

- The SageMaker deployment process is also automated through GitHub Actions.
- The SageMaker **notebook file** used to deploy the BLIP2 inference endpoint is version-controlled in GitHub.

CI/CD Steps:

Trigger:

- GitHub Actions is triggered on push/merge to `main`, `release/*`, or `sagemaker/*` branch
- Detects changes in the `sagemaker/` folder (including the deployment notebook)

Model Preparation in Notebook:

- The model (e.g. `Salesforce/blip2-flan-t5-xl`) is dynamically downloaded from Hugging Face within the notebook
- Inference logic and model code are bundled into a `.tar.gz` archive inside the notebook cell itself

Upload to S3:

- The notebook uploads the `blip2-endpoint.tar.gz` artifact to a configured S3 bucket (e.g., `s3://<bucket>/blip2/`)

Deploy/Update Endpoint (via notebook code):

- Using SageMaker SDK
- Notebook includes logic to **check for existing endpoint** and update it if necessary

Model Monitoring & Automation

- GitHub Actions periodically **checks for updates** to the model on Hugging Face using their API
- If a new version is found (e.g. SHA or version tag change):
  - Trigger the SageMaker notebook automatically using Amazon SageMaker Notebook Jobs or an nbconvert + CLI + Papermill-based execution
  - This allows auto-retraining or auto-redeployment based on upstream changes

Cost Considerations

Component	Estimated Monthly	Notes
SageMaker	~$876	g4dn.xlarge - $1.2/hr 24x7
ECS Fargate Task	~$18	minimum 0.5vCPU, 1GB RAM full-time
OpenAI API (1k calls)	~$6	$0.006/call for gpt-4
API Gateway	Free or ~$0.10	Under 1M reqs/month

Recommended - using async inference or cheaper OpenAI models for cost reduction.

Pricing Guidelines

SageMaker Pricing - https://aws.amazon.com/sagemaker/pricing/

ECS (Fargate) Pricing - https://aws.amazon.com/fargate/pricing/

API Gateway Pricing - https://aws.amazon.com/api-gateway/pricing/

Amazon S3 Pricing - https://aws.amazon.com/s3/pricing/

CloudWatch Logs Pricing - https://aws.amazon.com/cloudwatch/pricing/

Recommended Practices & Security

IAM Roles:

- ECS/Beanstalk task role to access S3 and invoke SageMaker
- SageMaker execution role for S3 input/output

Secrets: Use AWS Secrets Manager or SSM for storing OpenAI API keys

Networking: Restrict access with security groups, VPCs, and HTTPS-only APIs

Lifecycle Policies: Auto-delete S3 images after N days if async is used

Project Resources

Gluu GitHub Repo: https://github.com/gluudevteam/gluu-image-detection-blip2

AWS SageMaker Documentation:
https://docs.aws.amazon.com/pdfs/sagemaker/latest/dg/sagemaker-dg.pdf

AWS Docker Deployment:
https://aws.amazon.com/blogs/devops/deploy-a-docker-application-on-aws-elastic-beanstalk-with-gitlab/

AWS API Documentation:

- https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-deploy-api.html
- https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

AWS Cost Optimization:

- Async Inference:
  https://docs.aws.amazon.com/sagemaker/latest/dg/async-inference.html
- Multi-model Endpoints:
  https://docs.aws.amazon.com/sagemaker/latest/dg/multi-model-endpoints.html
- SageMaker Auto Shutdown:
  https://github.com/aws-samples/sagemaker-studio-auto-shutdown-extension

AWS Sample Notebooks:

- BLIP2 on SageMaker:
  https://github.com/aws-samples/amazon-sagemaker-genai-content-moderation/blob/main/blip2-sagemaker.ipynb
- Gen AI on SageMaker:
  https://github.com/aws-samples/sagemaker-genai-hosting-examples/blob/main/Llama2/Llama2-7b/LMI/llama2-7b.ipynb

Open AI Documentation:

- https://openai.github.io/openai-agents-python/
- https://platform.openai.com/docs/models

YouTube Playlist for SageMaker:
https://youtube.com/playlist?list=PLnJDJmQkmJTc2NgcqRzDuhcQqXZTSxxOm

Extra Reading:
https://medium.com/@himanshusangshetty/deploying-a-ml-model-on-amazon-sagemaker-421f3510fb8f