Infrastructure Cost Estimates

This document outlines the required infrastructure for running the AI Investment Agent in a production environment on AWS, and provides a framework for estimating the monthly costs. This only covers costs for the initial MVP deployment with minimal usage expected. Costs will increase with usage.

Cloud services have been chosen for ease of deployment and ability to reduce costs at low scale.

1. Required Production Infrastructure (AWS)

Here is a component-by-component breakdown of the recommended AWS services to host, run, and maintain the application.

Core Application Hosting

- **Service: AWS App Runner** (for the Flask Backend API)
    - **Justification:** App Runner is a fully managed service that simplifies the deployment and scaling of containerized web applications. It automatically handles load balancing, auto-scaling (from zero), and deployments from a container registry. This is highly cost-effective for new applications as you pay only for active compute, and it scales down to zero when there is no traffic.
- **Service: Amazon S3** + **Amazon CloudFront** (for the React Frontend)
    - **Justification:** The compiled React application consists of static files (HTML, CSS, JS). The most efficient and cost-effective way to host this is by placing the files in an S3 bucket and serving them globally via the CloudFront CDN. CloudFront provides low-latency delivery to users worldwide and integrates seamlessly with AWS Certificate Manager for free SSL/TLS certificates.

Database

- **Service: Amazon RDS for PostgreSQL**
    - **Justification:** As a managed database service, RDS handles tedious administrative tasks like patching, backups, and replication. Using a `db.t4g.small` or `db.t3.micro` instance provides a cost-effective starting point that can be easily scaled up as the application grows. Using the Graviton (g) instances often provides better price-performance.

Machine Learning Infrastructure

- **Service: Amazon EC2** (for Model Training)
    - **Justification:** A dedicated EC2 instance (e.g., `t3.large`) is ideal for running the scheduled training script. It can be provisioned with the necessary compute and memory, run the training job, and then be stopped or terminated to save costs. This is more flexible and cheaper than using a persistent, high-power instance.
- **Service: Amazon SageMaker Serverless Inference** (for Model Hosting)
    - **Justification:** The deep learning models will likely have intermittent traffic initially. SageMaker Serverless Inference is the most cost-effective solution for this pattern. It automatically provisions and scales compute based on the number of incoming requests and scales down to zero, meaning you pay nothing when the model is not being used.
- **Service: Amazon S3** (for ML Data & Artifacts)
    - **Just-ification:** An S3 bucket is required to act as a data lake and artifact store. It will hold the raw historical market data, the cleaned and feature-engineered datasets for training, and the versioned, trained model artifacts (`.joblib` files).

Networking, Security & DNS

- **Service: Amazon VPC (Virtual Private Cloud)**
    - **Justification:** All resources will be deployed within a VPC to ensure they are isolated from the public internet and can communicate with each other securely.
- **Service: AWS IAM (Identity and Access Management)**
    - **Justification:** IAM roles are essential for security. They provide granular permissions for services to interact (e.g., allowing App Runner to access RDS) without needing to hardcode credentials.
- **Service: Amazon Route 53**
    - **Justification:** This service is required to manage the application's custom domain name and route traffic to the appropriate resources (like the CloudFront distribution for the frontend and the App Runner service for the backend).

External Services (Non-AWS)

- **Service: Anthropic Claude API**
    - **Justification:** The chat functionality relies on the Claude 3 Opus model. This is an operational cost paid directly to Anthropic, based on the number of tokens processed. This is expected to be a significant component of the monthly operational cost.

2. Estimated Monthly Production Costs

This table provides a cost estimate for running the application on the recommended AWS infrastructure for one month. The estimates assume **minimal traffic** (equivalent to approximately 5 hours of active compute time).

Service	Pricing Model	Estimated Monthly Cost (USD)	Notes
AWS App Runner (Backend)	Pay-per-use (vCPU/Memory hours)	< $10.00	App Runner scales to zero when idle. This cost is based on 5 hours of active use with a small instance (1 vCPU, 2GB RAM).
Amazon S3 + CloudFront (Frontend)	Pay-per-use (Storage/Data Transfer)	$0.00	Usage for a new application will be well within the generous AWS Free Tier (5GB storage, 1TB data transfer).
Amazon RDS for PostgreSQL (DB)	Per-hour (Instance runs 24/7)	~$15.00	**This is the main fixed cost.** Based on a `db.t4g.micro` instance running continuously, which does not scale to zero.
AWS SageMaker Serverless	Pay-per-use (Compute duration/data)	< $10.00	**This scales to zero.** Unlike a standard SageMaker endpoint,

Service	Pricing Model	Estimated Monthly Cost (USD)	Notes
Inference (ML Model)			Serverless Inference only incurs costs when actively processing requests, making it ideal for intermittent traffic.
CI/CD & Monitoring	Free Tier	$0.00	GitHub Actions, AWS CodePipeline, and CloudWatch usage will fall within their respective free tiers at this scale.
Anthropic Claude API (LLM)	Pay-per-use (per 1M tokens)	**Variable**	This is the main variable cost and is highly dependent on usage. For example, a single chat could cost $0.10 - $0.20.
Total Estimated Monthly Cost		**<$50 + Variable LLM Costs**	The primary fixed cost is the database. The LLM API is the main variable cost that will grow with user engagement.