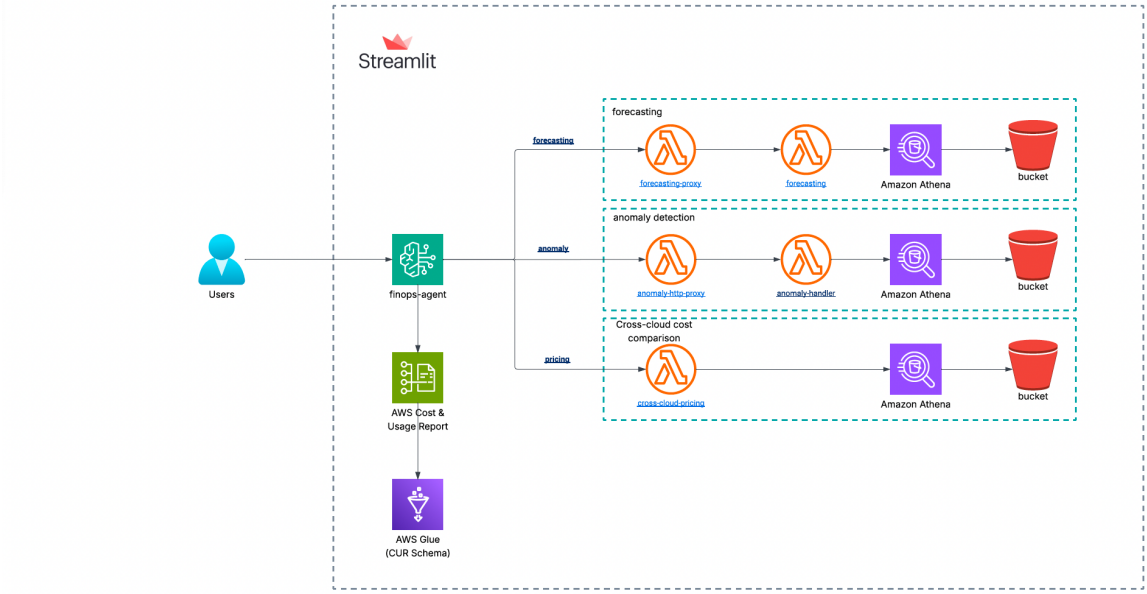
Agentic FinOps Assistant Walkthrough Document

Agentic FinOps Assistant, is an AI-driven cloud cost optimization platform that brings intelligence and automation into financial operations.

Cloud bills today are complex and unpredictable — teams often find out about cost spikes after the fact. We wanted to change that. So, we built an *agentic* application that proactively detects anomalies, forecasts future spend, and compares equivalent workloads across AWS, Azure, and GCP — all through an AI-powered interface.

Architecture diagram

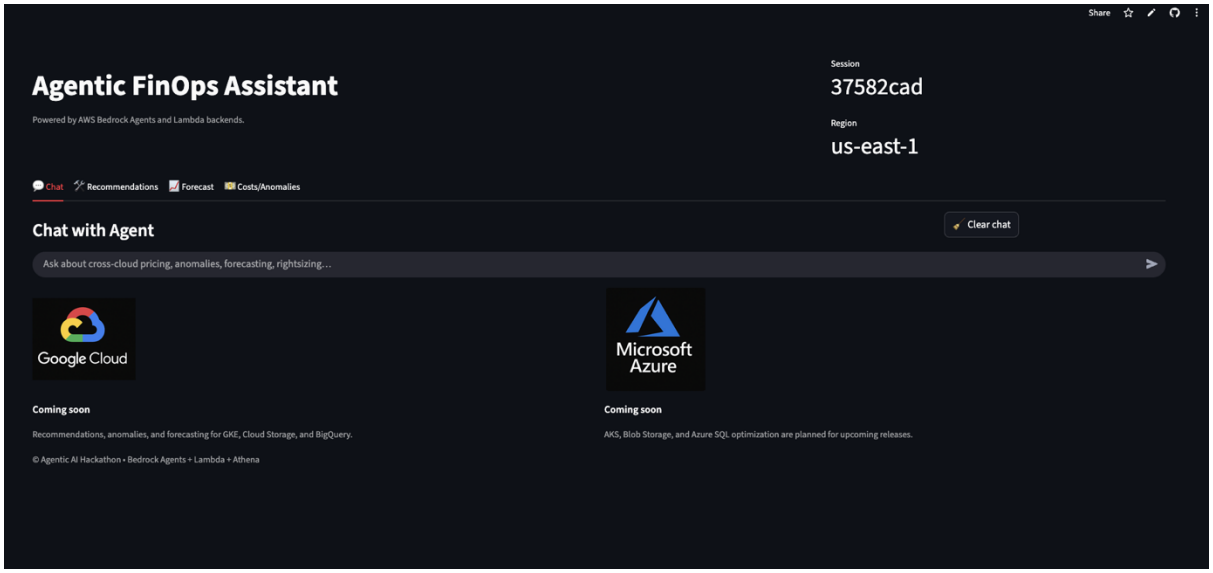


Components involved:

- **Frontend**: Streamlit
- **AI Agent:** Amazon Bedrock(AWS Nova Pro 1.0 LLM model)
- **Serverless compute:** AWS lambda functions
- **Data layer:** AWS Athena, AWS S3
- **IAM**: IAM roles with scoped permissions for Lambda ↔ Athena ↔ S3 ↔ Bedrock
- **Visualisation**: Real-time cost trends, anomalies, and savings charts

Components

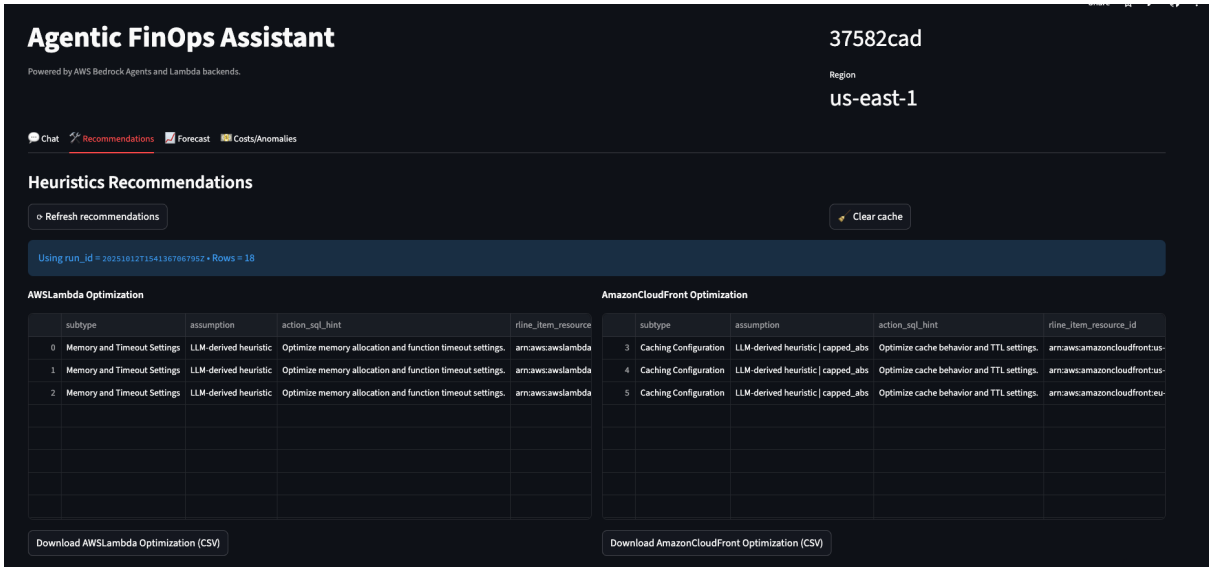
The main UI page can be accessed via URL: https://finops-agent.streamlit.app/ and the home page should come up like this:

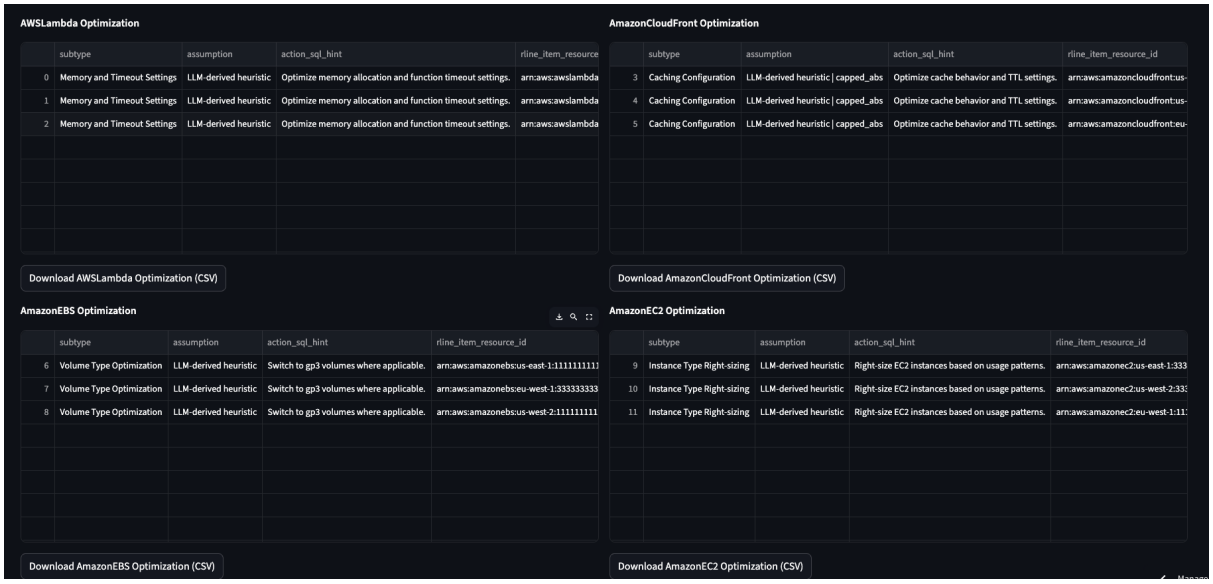


The home page allows the users to chat with bedrock agent about cost anomalies, cross-cloud pricing, forecasting and rightsizing. The page also has the option to clear the older chat. **Sample prompts are as follows:**

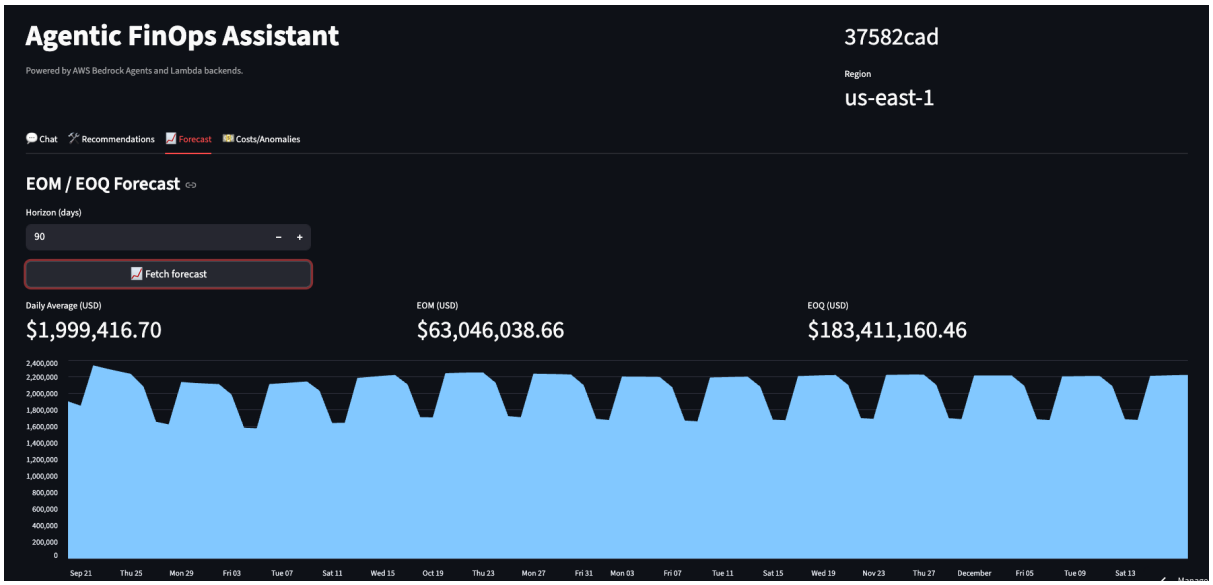
- “Cheapest Kubernetes in us-east-1 region.”
- “Storage cost in Azure in us-east-1 region.”
- “Summarise the cost anomalies in last 10 days.”
- “Forecast my cost in next 60 days”
- “when did the cost spike in last 60 days?”

The second tab gives the LLM driven right-sizing recommendations for AWS resources, along with the hints on how to right-size them. “Refresh recommendations” makes a call to bedrock agent, which in turn uses its intelligence to make a call to recommendations lambda function and populates the fresh recommendations:

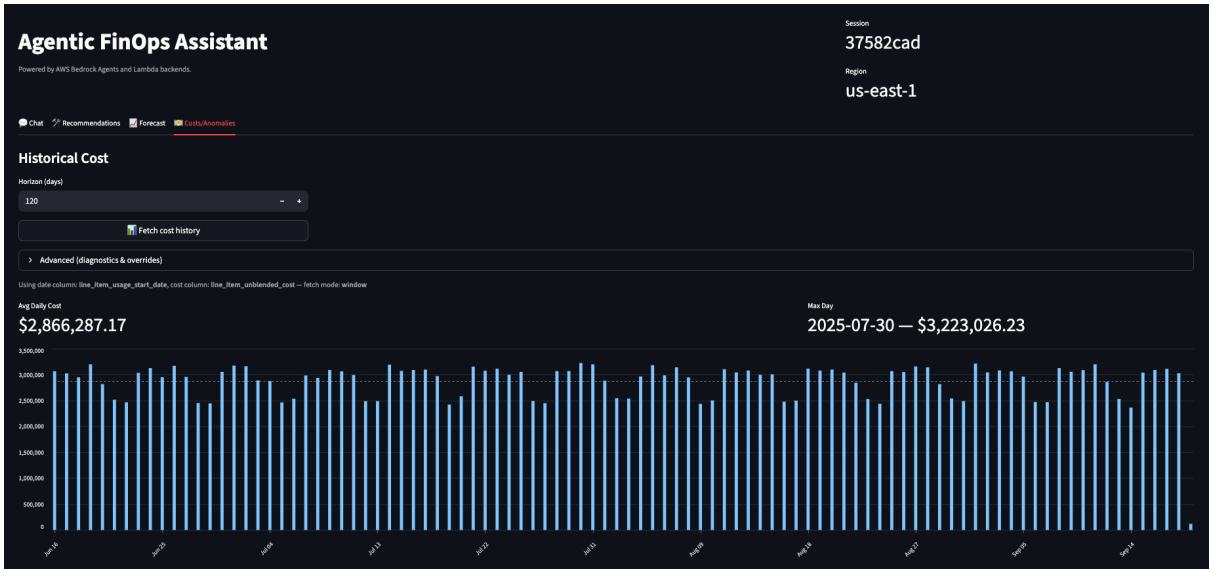




The third tab generates the cost forecasting for custom dates and also populates EOM/EOQ forecasting and daily average cost. “Fetch Forecast” will make a call to bedrock agent, which in turn uses its intelligence to make a call to recommendations lambda function and populates the fresh graphs:



The fourth tab generates the historical cost and anomalies and populates the graph. “Fetch Cost History” will make a call to bedrock agent, which in turn uses its intelligence to make a call to anomalies lambda function and populates the fresh graphs. It also populates the max day where the cost spiked along with average daily cost:

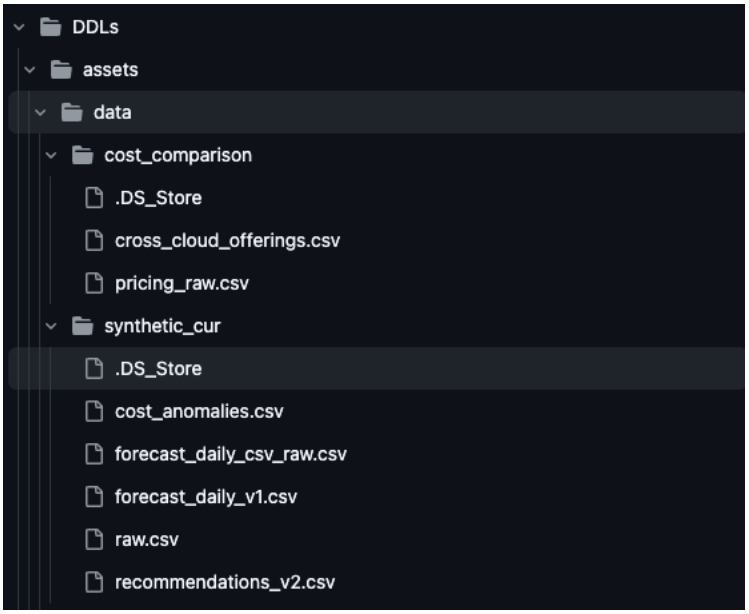


Deployments

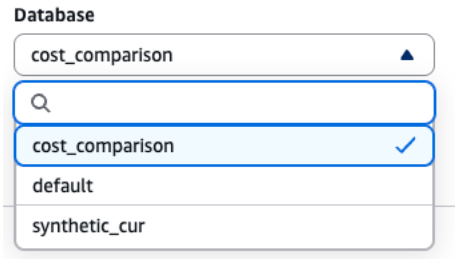
- The source code for the application can be found here: : https://github.com/Darksider379/aws-agentic-ai-hackathon/tree/main

- The DDL statements for the tables can be found here: https://github.com/Darksider379/aws-agentic-ai-hackathon/blob/main/DDLs/tables.sql . Sample data can also be found here: https://github.com/Darksider379/aws-agentic-ai-

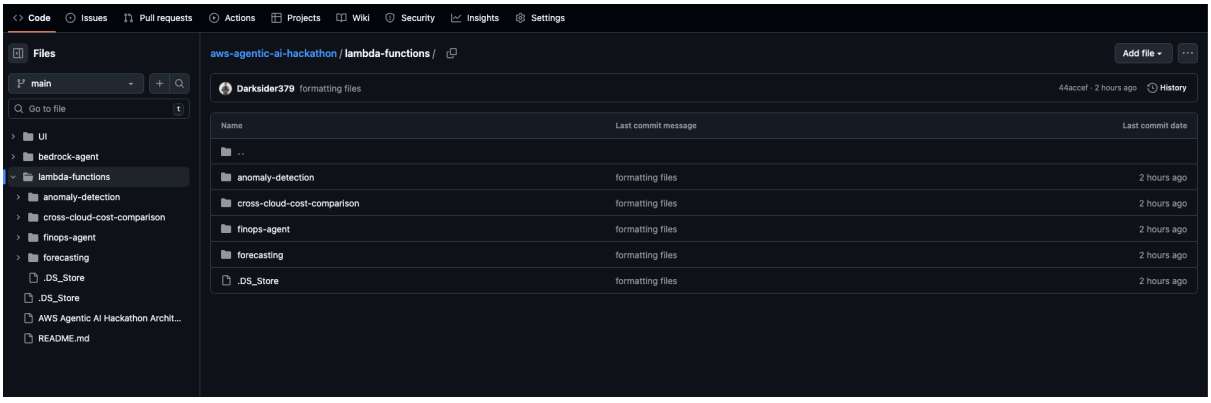
hackathon/tree/main/DDLs/assets/data . Please note that each folder represents the database under which the tables would go:



Athena snippet:

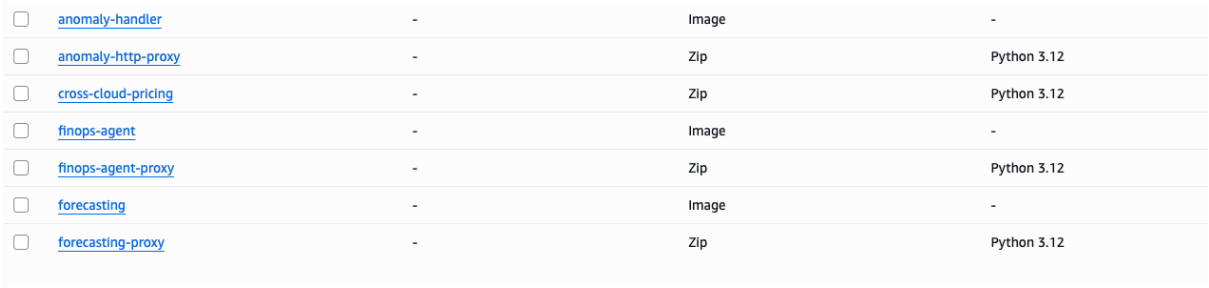


- The lambda functions to be deployed are shown below:



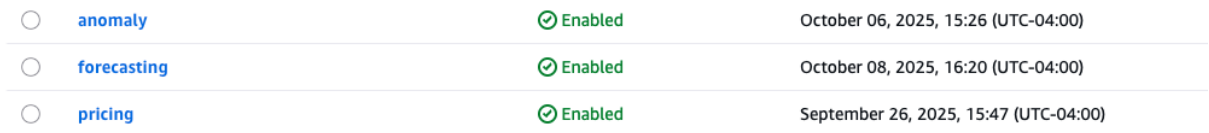
- Note that anomaly-detection, forecasting and finops-agent have an additional proxy functions which need to be deployed as well, that generated the output acceptable to bedrock. The dockerfiles for the functions that are deployed via image have also been added in the repository for reference. Config.ini represents the environment variables that need to be defined for the lambda

functions. Function names can be as below:



☐	anomaly-handler	-	Image	-
☐	anomaly-http-proxy	-	Zip	Python 3.12
☐	cross-cloud-pricing	-	Zip	Python 3.12
☐	finops-agent	-	Image	-
☐	finops-agent-proxy	-	Zip	Python 3.12
☐	forecasting	-	Image	-
☐	forecasting-proxy	-	Zip	Python 3.12

- Create the bedrock agent with name “finops-agent”. This application uses Amazon Nova Pro 1.0 native LLM. The instructions given to the model can be found here: https://github.com/Darksider379/aws-agentic-ai-hackathon/blob/main/bedrock-agent/agent_instruction.txt
  The model should have 3 action-groups as shown below:



○	anomaly	Enabled	October 06, 2025, 15:26 (UTC-04:00)
○	forecasting	Enabled	October 08, 2025, 16:20 (UTC-04:00)
○	pricing	Enabled	September 26, 2025, 15:47 (UTC-04:00)

  Anomaly and forecasting should point to the proxy lambda functions(anomaly-http-proxy and forecasting-proxy), where as pricing should point to cross-cloud-pricing lambda functions. Each action group should have an OpenAPI schema specifications defined, which can be found inside each action_groups folder(https://github.com/Darksider379/aws-agentic-ai-hackathon/tree/main/bedrock-agent/action_groups). The OpenAPI spec also includes the API gateway endpoints attached to the lambda functions. Those need to be updated as well.

- The streamlit UI code can be found here: https://github.com/Darksider379/aws-agentic-ai-hackathon/tree/main/UI/streamlit . Currently it is deployed to Streamlit cloud, however it can be hosted on AWS Elastic beanstalk or AWS app runner using nginx.

- Note that bedrock(and its aliases) should have lambda invocation role as well, which should be defined under “Resource-based policy statements”. Sample command can be as follows:

```
aws lambda add-permission \
  --region us-east-1 \
  --function-name forecasting-proxy \
  --statement-id bedrock-GXY0ETOUIJ \
  --action lambda:InvokeFunction \
  --principal bedrock.amazonaws.com \
--source-arn arn:aws:bedrock:us-east-1:784161806232:agent/IO47D3HMWR/alias/GXY0ETOUIJ
```

Links:

App URL: https://finops-agent.streamlit.app/
Source code repo: https://github.com/Darksider379/aws-agentic-ai-hackathon
Architecture diagram: https://github.com/Darksider379/aws-agentic-ai-hackathon/blob/main/AWS%20Agentic%20AI%20Hackathon%20Architecture%20Diagram.pdf
Bedrock agent instruction: https://github.com/Darksider379/aws-agentic-ai-hackathon/blob/main/bedrock-agent/agent_instruction.txt
Table DDLs: https://github.com/Darksider379/aws-agentic-ai-hackathon/blob/main/DDLs/tables.sql
Sample data: https://github.com/Darksider379/aws-agentic-ai-hackathon/tree/main/DDLs/assets/data