Module 3

Automation and Cost Management Project Based on TAGs in AWS

Institution: Instituto de Tecnologia e Liderança – INTELI
Business Partner: Thomson Reuters

Authors:

Gustavo Monteiro

Advisor: Rodolfo Riyoei Goya

Agenda

- Module 2 Recap
- Initial API Development and Architecture
- Architectural Evolution 1: Database Persistence
- Architectural Evolution 2: Serverless Automation
- System Validation & Large-Scale Cost Analysis
- Challenges & Key Learnings
- Conclusion & Next Steps

Module 2 Recap

- **Key Delivery:** A formalized Tagging Standard was established.
  - Defined 6 mandatory tags for governance (project-id, cost-center, etc.).
  - Created a Quick Start Guide to facilitate adoption.
- **Starting Point for Module 3:** With a governance standard in place, the goal is now to automatically collect and expose the data generated by these tags.

Initial API Development and Architecture

- **Objective:** Create a service to collect cost and resource data from AWS.
- **Initial Technologies:**
  - FastAPI: High-performance Python framework with automatic API documentation (Swagger UI).
  - Secure Credentials: Used .env file to manage AWS keys, avoiding hardcoding secrets.
- **First Endpoints Created:**
  - **/custo-diario:** To fetch daily cost data from AWS Cost Explorer.
  - **/instancias-ec2:** To list the EC2 instance inventory and associated tags.

Architectural Evolution 1: Database Persistence

- **Problem**: Real-time API calls to AWS were slow and dependent on network connectivity
- **Solution: Introduced a SQLite database as a persistence layer**:
  - Collector (populate_db.py): A script fetches data from AWS once a day and saves it to the local database.
  - API (app.py): Now reads data from the local database, providing near-instant responses.
- **Benefits**:
  - Performance: Drastically improved API response time.
  - Resilience: API can serve cached data even if AWS is temporarily unavailable.

Architectural Evolution 2: Serverless Automation

- **Problem**: The data collection script still required manual execution
- **Solution: Migrated to a 100% automated, serverless architecture in AWS**:
- **Core Components**:
  - **AWS Lambda**: Executes the data collection code without managing servers.
  - **Amazon S3**: Stores the collected data (JSON files) as a centralized, durable data lake.
  - **Amazon EventBridge**: A scheduler that automatically triggers the Lambda function once every 24 hours.

System Validation & Large-Scale Cost Analysis

- **Governance Validation**:
  - Used CloudFormation to deploy a test environment simulating multiple products.
  - Crucially, included a non-compliant (untagged) resource to successfully test the system's detection capabilities.
- **Large-Scale Cost Analysis (Corporate Scenario: 25,000 resources)**:
  - **Projected Monthly Cost for the Architecture: ~$7.93**
  - This result proves the solution is extremely cost-effective and financially viable for enterprise-scale deployment.

Challenges & Key Learnings

- **Challenge:** Temporary AWS Academy account block due to exceeding lab limits
  - Resolution: Quickly resolved with advisor support.
  - Learning: A practical lesson in managing constraints and quotas in cloud environments.

- **Key to Success:** Proactive & Agile Planning
  - An initial planning meeting to address difficult integration questions before development was the most critical factor for success.
  - This approach transformed a potentially challenging module into a smooth and efficient execution.

Conclusion and Next Steps

Module 3 Achievements:

- Delivered a fully automated API for cost and resource data collection.
- Implemented a scalable, resilient, and highly cost-effective serverless architecture.
- Validated the solution's technical and financial viability

Next Steps (Module 4):

- The project will now focus on **Data Visualization**.
- The goal is to develop dashboards that turn the collected data into **actionable insights** for stakeholders, enabling visual analysis of costs, resource allocation, and governance compliance.

Thank you