Serverless Multi-Model AI Gateway

Secure, Cost-Optimized Access to Amazon Bedrock via LiteLLM

Mike Berggren

TOC

The Challenge

The Objective:

- Enable application access to diverse AI models (e.g. Anthropic, Cohere, Meta, Amazon) without vendor lock-in.
- Secure the infrastructure without managing static API keys or long-term credentials.
- Minimize infrastructure costs for non-production/lab environments.

Key Constraints:

- Security: "Zero Trust" networking and IAM-based authentication.
- Agility: Infrastructure must be deployable and destroyable in minutes.
- Compatibility: Must support standard OpenAI API formats for easy developer integration.

Solution Overview

Solution: A serverless API Gateway using LiteLLM running on AWS Fargate Spot.

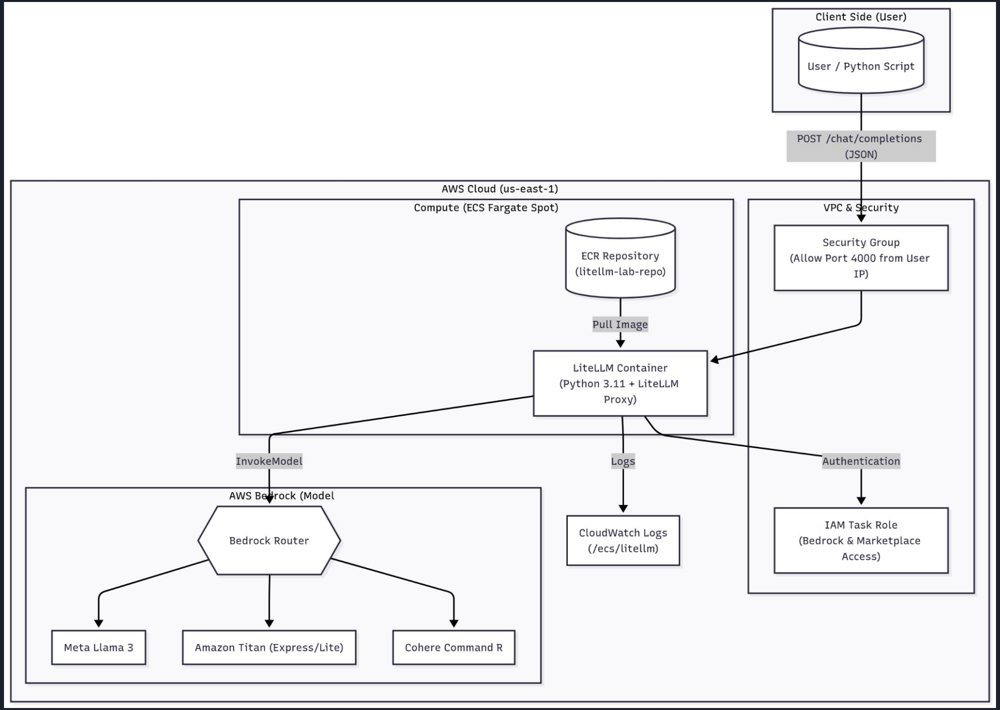
Core Capabilities:

- **Unified Interface:** A single API endpoint routing traffic to Llama 3, Cohere Command R, and Amazon Titan.
- **Serverless Compute:** No EC2 instances to manage or patch.
- **Infrastructure as Code:** 100% defined in Terraform for repeatable, idempotent deployments.
- **Supply Chain Security:** Custom-built Docker image based on verified upstream Python sources

High-Level Architecture

Traffic Flow:

1. **Ingress:** Client requests hit a strict **Security Group** firewall.
2. **Validation:** Traffic is rejected unless it originates from the **Administrator's IP** (Dynamic Allowlist).
3. **Routing:** The **LiteLLM Container** parses the request and selects the correct model.
4. **Auth:** The container assumes an **IAM Task Role** to fetch temporary AWS credentials.
5. **Execution:** The prompt is sent to **Amazon Bedrock**, and the response is streamed back.


Client Side (User)
User / Python Script
POST /chat/completions (JSON)
AWS Cloud (us-east-1)
Compute (ECS Fargate Spot)
ECR Repository (litellm-lab-repo)
Pull Image
LiteLLM Container (Python 3.11 + LiteLLM Proxy)
VPC & Security
Security Group (Allow Port 4000 from User IP)
InvokeModel
Logs
Authentication
AWS Bedrock (Model
Bedrock Router
CloudWatch Logs (/ecs/litellm)
IAM Task Role (Bedrock & Marketplace Access)
Meta Llama 3
Amazon Titan (Express/Lite)
Cohere Command R

Security & Cost Innovations

Security First:

- **No Static Keys:** Eliminated AWS_ACCESS_KEY_ID from the codebase. Authentication relies entirely on ephemeral IAM Task Roles.

- **Least Privilege:** The IAM Policy strictly limits actions to InvokeModel and specific Marketplace subscriptions (aws-marketplace:Subscribe), preventing broad account access.

- **Network Isolation:** The Terraform configuration automatically detects the deployer's public IP and locks the Security Group to that single /32 CIDR block.

Cost Optimization:

- **Fargate Spot:** Utilizes AWS spare capacity, reducing compute costs by ~70% compared to On-Demand pricing.

- **Ephemeral Architecture:** The entire lab can be destroyed (terraform destroy) and rebuilt in <5 minutes, ensuring zero idle costs.

Automated Deployment

Automation scripts (demo.py / demo.sh) that handles the full lifecycle:

1. **Provisions Infrastructure:** Runs Terraform to build the VPC, ECR, and ECS Cluster.

2. **Builds Supply Chain:** Compiles the Docker image from source locally to ensure architecture compatibility.

3. **Deploys:** Pushes the image and forces a Rolling Update on ECS.

4. **Validates:** Waits for health checks to pass, extracts the dynamic Public IP, and runs integration tests against all 4 models.

Result: A fully functional, tested environment in under 7 minutes with a single command.

Getting Started

Repository: https://github.com/edgevolt/aws-litellm-proxy

Prerequisites:

- AWS CLI (Admin Access)
- Terraform & Docker
- Python 3

How to run:

- **Option A: Python Deployment (Recommended)**
  - `python3 demo.py`

- **Option B: Bash Deployment**
  - `chmod +x demo.sh`
  - `./demo.sh`

Thank you!