Bedrock Access Gateway – Design Spec

Doc status: Draft v1.1
Owner: Miguel Merlin
Date: Aug 11, 2025
Reviewers: Brandon Yen

---

1) Overview

Provide Bedrock access to internal IAM users via a controlled gateway that works from a variety of interfaces, ranging from developer IDE extensions (Cline) to a front-facing UI. The system exposes a stable HTTP API (OpenAI-style), enforces per-user throttling and monthly token budgets, and includes multiple kill-switches.

Primary goals

- Per-user **TPS** throttling and **monthly cost limits**
- Centralized audit, cost visibility, and guardrails

---

2) Requirements

2.1 Functional

1. Users can call the Lambda Function URL from IDE extensions or the UI.
2. Requests from IDE extensions authenticated via STS credentials generated from IAM credentials.
3. Requests from UI authenticated via Cognito JWT tokens.
4. Enforce **monthly cost limits** per user; reject after budget exceeded.
5. Support streaming responses (SSE) for chat/completions.
6. Allow list of Bedrock models; requests to others are rejected.
    1. Current list of models includes:

    1. Claude 3 Haiku
    2. Claude 3.5 Sonnet
  2. All other Anthropic models were deemed not necessary or require access through provisioned throughput.
7. Provide **usage/remaining quota** endpoint.
8. Emit structured logs and metrics for cost and auditing.

2.2 Operational

1. **Hard kill**: SCP denying Bedrock; model access toggle in Bedrock.
2. **Per-user soft kill:** activate/deactivate access keys for each user.
3. Config changes (caps, models, plans) without redeploy.
4. SLO: 99.9% monthly availability for gateway.
5. P90 end-to-end latency target: ≤ 1.5s for 2-KB prompts, non-streaming.

2.3 Security & Compliance

- Users cannot directly call Bedrock; only the gateway’s IAM role can.

---

3) High-Level Architecture

```
UI/IDE ──> Lambda Function URL ──> Lambda Proxy ──> Bedrock
                                          └─> DynamoDB (usage)



Control plane:  DynamoDB (config/limits)
```

Key Components

- **Lambda Proxy**: Validates payload, authenticates users, calls Bedrock, streams results, updates usage counters from actual token usage.
- **DynamoDB**: Token metering per user/day; user profiles; config.

---

4) Data Model

4.1 DynamoDB Tables

Transaction Table:

- **PK**: `userId` (S)
- **SK**: `timestamp` (S, `YYYY-MM-DDTHH:mm:ss`)
- Attributes: `cost` (float), `modelId` (S), `usage` (`outputTokens` (S), `inputTokens` (S))

Monthly Usage Table:

- **PK**: `userArn` (S)
- **SK:** `month_year` (S, `MM_YYYY`)
- Attributes: `cost` (float), `invocations` (int)

---

5) API Design

5.1 Authentication

- **Headers**:
  - `x-aws-session-token: <key>` (STS session token)
  - `x-aws-access-key: <key>` (STS access key)
  - `x-aws-secret-key: <key>` (STS secret key)
- Inference proxy finds user ARN based on credentials

5.2 Endpoints (JSON)

`POST (Lambda Function URL)`

Request:

```
{
    "modelId": "anthropic.claude-3-haiku-20240307-v1:0",
    "messages": [
        {
            "role": "user",
            "content": [
                {
                    "text": "<task>"
```

```
                },
                {
                    "text": "<system prompt>"
                },
                {
                    "text": "<environment details>"
                }
            ]
        }
    ],
    "system": [
        {
            "text": "<system prompt>"
        }
    ],
    "inferenceConfig": {
        "maxTokens": 4096,
        "temperature": 0
    },
    "additionalModelRequestFields": {}
}
```

GET /v1/usage

Include the `userArn` in the header to retrieve their monthly usage statistics and the monthly limit.

Revision #15
Created 12 August 2025 00:24:18 by Miguel Merlin
Updated 2 December 2025 18:19:39 by Brandon Yen