# Policy Watch: Architecture & Migration Strategy

**Date:** October 26, 2023

**Author:** Senior AWS Architect

**Status:** Hybrid Strategic Review

## 1. Executive Summary

**Policy Watch** will adopt a **Hybrid Architecture** for its modernization journey.

Previously, a full cloud migration was considered. However, due to critical IP reputation risks (scraping) and high complexity in legacy database connectivity (Oracle), the decision has been made to **retain the "Policy Loader" and "Rules Loader" modules On-Premises**.

The **Migration Strategy** now focuses on wrapping these legacy on-prem components in **APIs (FastAPI)** and orchestrating them via a central **AWS Step Functions** workflow. The intelligence layer (Extractor/Analyzer) will remain Cloud-Native (ECS).

## 2. As-Is Architecture Overview

The system operates on a **Producer-Consumer** model with five core components:

| **Component** | **Role** | **Location** | **Schedule** | **Input** | **Output** |
| --- | --- | --- | --- | --- | --- |
| **1. Policy Loader** | Producer (Scraper) | On-Prem Server | Every 4 Hours | Payer Portals | S3 (PDF), Postgres (Metadata) |
| **2. Policy Extractor** | Consumer (Parser) | AWS ECS Service | Every 2 Hours | S3 (PDF) | S3 (JSON), Postgres (Scores) |
| **3. Policy Analyzer** | Consumer (Logic) | AWS ECS Service | Every 2 Hours | Extractor JSON | S3 (Impact JSON) |
| **4. Rules Loader** | Producer (Legacy) | On-Prem Server | Manual | Oracle (Atlas DB) | S3 (Consolidated JSON) |
| **5. Rules Processor** | Consumer (ML) | On-Prem Script | Manual | S3 (JSON) | Postgres (Embeddings) |

### Key Architectural Risks (As-Is)

* **IP Reputation:** Moving scrapers to Cloud invites immediate blocking. **Decision:** Stay On-Prem.
* **Legacy Debt:** Rewriting Perl/Oracle logic is high risk. **Decision:** Wrap in API, stay On-Prem.
* **Fragile Scheduling:** Independent Cron jobs lead to race conditions. **Decision:** Centralize orchestration in Cloud.

## 3. Migration Strategy: The Hybrid Path

The target state is a **Hybrid Control Plane**. AWS Step Functions will act as the "Brain," sending signals to On-Prem "Limbs" (Scripts) and Cloud "Limbs" (ECS Tasks).

### Module 1: Policy Loader (The Scraper)

* **Strategy:** **Hybrid API Wrapper (Option B)**
* **Action:** Wrap the existing Python scraping script in a lightweight **FastAPI** application on the On-Prem server.
* **Orchestration:** AWS Step Functions calls this On-Prem API to trigger scraping.
* **Benefit:** Retains the trusted On-Prem IP address, avoiding WAF blocking.

### Module 2: Policy Extractor (The Parser)

* **Strategy:** **Lift & Shift + Optimize (Cloud)**
* **Action:** Convert the 24/7 Service to a **Scheduled ECS Task** triggered by EventBridge.
* **Enhancement:** Add DocumentParser shared utility for HTML & PDF support.
* **Critical Requirement:** Implement **Dead Letter Queue (DLQ)** for failed parses.

### Module 3: Policy Analyzer (The Logic)

* **Strategy:** **Port & Orchestrate (Cloud)**
* **Action:** Port logic to packages/tools/policy\_analyzer.py (ECS).
* **Enhancement:** Use **Step Functions** to trigger this *only* after the On-Prem Rules Processor has confirmed success.

### Module 4: Rules Loader (The Legacy Migration)

* **Strategy:** **Hybrid Wrapper (Option B)**
* **Action:** Create a Python wrapper (FastAPI/Agent) on the On-Prem server to invoke the legacy Perl script.
* **Orchestration:** AWS Step Functions triggers this wrapper remotely.
* **Benefit:** Avoids complex Oracle DB connectivity setup in AWS.

### Module 5: Rules Processor (The ML Engine)

* **Strategy:** **Hybrid Orchestration**
* **Action:** Keep the script On-Prem.
* **Enhancement:** Trigger it via the local On-Prem API/Agent immediately after Rules Loader completes.

## 4. Architect's Review & Recommendations

### 1. Hybrid Connectivity & Security

* **Risk:** Exposing On-Prem scripts as APIs creates a security surface area.
* **Resolution:**
  + Use **Mutual TLS (mTLS)** or strict **API Key** authentication (stored in AWS Secrets Manager).
  + Ensure traffic flows exclusively over **VPN** or **Direct Connect**, never the public internet.

### 2. Orchestration vs. Scheduling (AWS Step Functions)

* **Objective:** Centralize control. The Cloud should know the status of the On-Prem tasks.
* **Resolution:** Adopt **AWS Step Functions**. Define the hybrid workflow:
  + Start -> Parallel(RemoteCall:Loader, RemoteCall:Rules) -> Wait(Polling) -> Extractor(Cloud) -> Analyzer(Cloud) -> End.

### 3. Legacy Connectivity

* **Risk:** Reliance on VPN stability.
* **Resolution:** Implement robust **Retries with Exponential Backoff** in the Step Functions definition for all remote On-Prem calls.

## 5. Conclusion

The move to a **Hybrid Architecture** pragmatically balances modernization with risk management. By keeping the Scrapers and Legacy Rules logic On-Premises, we avoid the two biggest risks (IP Blocking and Oracle Connectivity) while still gaining the benefits of Cloud-Native Orchestration and Intelligence.