**Data Ingestion: [Needs revision and approval]**

**Source** **to Ingest Alerts** **Format**  
**AWS Cost Anomaly Detection** SNS → Lambda or Email → IMAP JSON / Email  
**VMware Tanzu CloudHealth** Webhooks or Email JSON / Email  
Webhooks or Alerts → S3 / API / Email Varies

**A. Email Alerts (AWS / CloudHealth)**  
• Use an **IMAP Email Reader Agent**:Connect to inbox using IMAP  
• Read new emails with subject like Anomaly Alert or CloudHealth  
• Use LLM to extract fields: Account Name, Service (e.g., AWS Glue), Amount, Date Range and Alert Type

**B. Webhook/SNS Event Stream (Real-Time)**  
• **AWS**:  
• Set up Cost Anomaly Detector → SNS topic  
• SNS triggers **Lambda**, **SQS**, or **EventBridge** → logs alert  
• **VMware CloudHealth**:  
• Set up **Webhook notifications** or **scheduled API reports**  
• Send to API Gateway → Forward to your app / agent

**C. Parse Alert Payload**  
Use **LLM (Anthropic)** or regex logic to extract:  
  
{  
 "account": "CBRE-PortfolioServices-Prod",  
 "service": "AWS Glue",  
 "anomaly\_start\_date": "2024-06-27",  
 "anomaly\_cost": 10000.00,  
 "alert\_id": "aws-glue-cbre-june27"  
}  
  
Store into **PostgreSQL → anomalies table**:  
  
INSERT INTO anomalies (alert\_id, account, service, date, cost, status)  
VALUES (...);  
  
Once stored in DB:  
• Agent 1 picks up new rows

**Tools:**

|  |  |  |
| --- | --- | --- |
| **Component** | **Tool/Platform** | **Purpose** |
| Agentic Framework | Crew AI | Agent-to-agent flow & shared memory |
| AI Model | Anthropic / OpenAI | NLP & prompt-based logic |
| Database | PostgreSQL | Structured logging, ticket & anomaly tracking |
| Workflow Orchestration | Palantir | Human approval loop (ServiceNow logic), Terraform orchestration, Slack/email notifications |
| Observability | Datadog | Observability & tracing |
| LLM Prompt Store | Prompt Store | Storing prompt templates for reuse |

**Step-by-Step Architecture Using Crew AI**

A diagram of an email

AI-generated content may be incorrect.

**Agent 0: Ingestion Agent**  
  
**Role**: Fetch and parse anomaly alert data from emails, webhooks, or SNS events.  
  
**Source** **How It Works**  
**Email (IMAP)** Connects to inbox, reads unseen alerts, extracts text  
**Webhook/API** Accepts POST requests from providers like CloudHealth  
**AWS SNS** Subscribed Lambda or worker pulls from SNS or SQS  
  
**Responsibilities**:  
• Extract anomaly data (account, service, cost, date)  
• Convert emails or JSON into structured format  
• Store initial data into PostgreSQL.anomalies  
  
**Agent 1: Intake Parser Agent**  
  
**Role**: Normalize and log anomalies into PostgreSQL  
  
**Responsibilities**:  
• Use LLM (Anthropic/OpenAI) to parse unstructured alert content  
• Normalize to schema: account, service, anomaly\_start\_date, cost  
• Insert into database: anomalies table  
• Log event to Datadog  
• Pass record ID to Agent 2

**Agent 2: Threshold & Duplication Validator**  
  
**Role**: Filter irrelevant alerts based on business rules  
  
**Responsibilities**:  
• Check if anomaly already exists in DB  
• Apply cost or % increase thresholds  
• Add status to DB: "ignored", "valid"  
• Only forward "valid" alerts  
  
**Agent 3: Stakeholder Resolver**  
  
**Role**: Identify the responsible team and approver  
  
**Responsibilities**:  
• Use account/service to find:  
• ServiceNow Support Group (e.g., Portfolio Services)  
• Billing Approver (email or SN ID)  
• Write mapping to tickets\_meta table  
• Pass enriched anomaly to Agent 4  
  
**Agent 4: Notification & Ticket Agent**  
  
**Role**: Generate and send approval notification (email/SN ticket)  
  
**Responsibilities**:  
• Generate human-readable message using LLM  
• Integrate with **Palantir** to: human manual approval  
• Create ticket in **ServiceNow**  
• Assign to appropriate SN Group  
• Request action approval  
• Monitor for response and log ticket status  
  
Once approved → pass control to Agent 5  
  
**Agent 5: Terraform Planner**  
  
**Role**: Prepare infrastructure remediation plan  
  
**Responsibilities**:  
• Identify impacted infra from anomaly record  
• Map anomaly to Terraform module  
• Run terraform plan  
• Capture plan output (estimated changes, costs)  
• Write result to DB (terraform\_logs)  
• If plan successful → pass to Agent 6  
  
**Agent 6: Terraform Executor (optional for now)**  
  
**Role**: Apply the infrastructure fix  
  
**Responsibilities**:  
• Run terraform apply for approved module  
• Confirm right-sizing or remediation  
• Update ServiceNow ticket with final action  
• Log results to DB  
• Notify stakeholders (email or SN)  
  
**Agent-to-Agent Flow Overview**  
  
graph TD  
 Z[Agent 0: Ingestion Agent] --> A[Agent 1: Intake Parser]  
 A --> B[Agent 2: Validator]  
 B -->|Valid| C[Agent 3: Resolver]  
 C --> D[Agent 4: Notification & SN Ticket]  
 D -->|Approved| E[Agent 5: Terraform Plan]  
 E -->|Plan OK| F[Agent 6: Terraform Apply]  
  
 **Final Agent Table Summary**

|  |  |  |
| --- | --- | --- |
| **Agent #** | **Name** | **Role** |
| 0 | Ingestion Agent | Collects alerts from email/webhook/SNS |
| 1 | Intake Parser Agent | Parses unstructured alerts and logs them |
| 2 | Threshold Validator Agent | Filters by cost/duplication |
| 3 | Stakeholder Resolver Agent | Maps resource to SN group/approver |
| 4 | Notification & Ticket Agent | Sends notification, creates SN ticket |
| 5 | Terraform Planner Agent | Prepares Terraform remediation |
| 6 | Terraform Executor Agent | Applies Terraform fix |

A diagram of an anthropomorphic agent

AI-generated content may be incorrect.

A diagram of an automated algorithm

AI-generated content may be incorrect.