**Agentic plan** to automate your AML MVP using a **Master Orchestrator** plus a set of **specialist agents** powered by MCP servers and your syntropAIkit.

**High-level approach (Rule of 3)**

* **Platform stream**: Infra, Security/Gov, Observability
* **Data stream**: Ingestion, Catalog, Transform/Serve
* **Intelligence stream**: Detection, ML Triage, Case/BI

Each stream has one or more **individual agents**. A **Master Agent** plans, routes, verifies, and records every change.

**Architecture (Mermaid)**

flowchart LR

subgraph Master["Master Orchestrator (Planner/Router/Verifier)"]

PLAN[Task Planner]

EXEC[Executor & Gatekeeper]

KB[(State/KB: run logs, lineage, approvals)]

end

subgraph Platform

INFRA[Infra Agent\n(Terraform/Azure APIs)]

SEC[Sec/Governance Agent\n(Purview, Policies, RBAC)]

OBS[Observability Agent\n(Azure Monitor, Sentinel)]

end

subgraph Data

ING[Ingestion Agent\n(ADF, Event Hubs)]

CAT[Catalog Agent\n(Unity Catalog, Purview)]

DBX[Databricks Agent\n(DLT, Jobs, UC)]

SQLS[SQL Serving Agent\n(Synapse Serverless)]

end

subgraph Intelligence

DET[Detection Agent\n(Rules/Typologies)]

ML[ML Triage Agent\n(MLflow, batch scoring)]

CASE[Case & BI Agent\n(Azure SQL, Power BI/Power Apps)]

end

PLAN --> EXEC --> INFRA & SEC & OBS & ING & CAT & DBX & SQLS & DET & ML & CASE

INFRA --> KB

DBX --> KB

ML --> KB

CASE --> KB

SEC --> KB

**Agent roster (who does what)**

| **Agent** | **Primary goal** | **MCP tools / APIs** | **Permissions (least-privilege)** |
| --- | --- | --- | --- |
| **Master** | Decompose requests into task graph, route to agents, enforce gates, verify outcomes, log state | Planner (OpenAI/LLM), kb store (your file-graph), Scheduler | Read all; write to KB; no cloud writes |
| **Infra** | Provision/modify Azure & Databricks resources | Terraform runner, Azure ARM/CLI, Databricks WS API | Contributor on infra RGs only |
| **Sec/Governance** | Purview catalogs/classification; Azure Policies; UC grants | Purview REST, AzPolicy, Databricks UC API | Data Reader/Curator; Policy Contributor |
| **Observability** | Dashboards/alerts for jobs, clusters, costs | Azure Monitor, Log Analytics, Cost Mgmt, Sentinel | Reader + Monitoring Contributor |
| **Ingestion** | ADF pipelines & IR; Event Hubs | ADF Mgmt API, Event Hubs Mgmt/Data, Storage | Data Factory Contributor; Storage Data Contributor (raw) |
| **Catalog** | Unity Catalog external locations/credentials | Databricks UC API | UC admin subset; no workspace admin |
| **Databricks** | DLT pipelines, Jobs, cluster policies, notebooks | Databricks Jobs/Pipelines API, Repo API | Workspace User; Jobs Run As MI |
| **SQL Serving** | Synapse Serverless views & connectivity | Synapse SQL API/sqlcmd, MSI bindings | Synapse SQL Admin (serverless) |
| **Detection** | Maintain baseline rules configs, run backtests | Databricks notebooks, config store | Create/modify in repo; execute via Jobs |
| **ML Triage** | Train/register/promote model, batch scoring | MLflow Registry, Jobs | MLflow Admin; read gold; write alerts\_scored |
| **Case & BI** | Maintain Azure SQL schema, Power BI datasets | SQL (JDBC), Power BI REST, Power Apps | DB DDL/DML; Power BI Dataset Admin |

All secrets via **Key Vault**, all PaaS via **Private Endpoints**, all agents use **Managed Identity** wherever possible.

**How the Master Agent runs work (control loop)**

**Inputs**: natural-language request (“Deploy Step 4 rules”), or schedule/event.  
**Outputs**: signed plan, executed changes, verifications, and an audit trail.

**Pseudocode**

while True:

request = inbox.get() # user/schedule/event

plan = decompose(request) # DAG of tasks with agent owners

gates = compute\_gates(plan) # dry-run, human-approval if prod

record(plan, KB)

for task in topological\_sort(plan):

if requires\_approval(task): wait\_for\_approval(task)

preview = call\_agent(task, dry\_run=True)

assert verify\_preview(preview)

result = call\_agent(task, dry\_run=False)

assert verify\_result(result) # e.g., resource exists, table filled

record(result, KB)

notify\_success(plan) or notify\_failure\_context()

* **Decompose** uses few-shot prompts anchored on your AML blueprint.
* **Verify** = run read-only checks (e.g., SELECT COUNT(\*) after DLT run; az resource show after TF apply).
* **Gates**: dev auto-approve; prod requires human sign-off (Databricks cluster policy + UC grants).

**MCP server plan (concrete)**

You already have MCP servers; extend with these **tool sets**:

**1) mcp\_azure\_iac**

* **Tools**: tf\_plan, tf\_apply, policy\_assign, pe\_create
* **Args**: path, vars, target subscription
* **Dry-run** support: tf\_plan
* **Run-as**: federated MI to GitHub/ADO

**2) mcp\_databricks**

* **Tools**: dlt\_create\_update, jobs\_run, workspace\_import, uc\_grant, cluster\_policy\_set
* **Args**: notebook path, pipeline JSON, catalog/schema, grants
* **Verify**: tables\_exist, run\_status

**3) mcp\_adf\_eventhubs**

* **Tools**: adf\_pipeline\_upsert, adf\_trigger\_run, eh\_ns\_create, eh\_consumer\_group
* **Verify**: lastRun status, checkpoint files

**4) mcp\_synapse\_sql**

* **Tools**: exec\_query, create\_view, cetas
* **Auth**: MSI; **Verify**: row counts, schema hash

**5) mcp\_purview\_uc**

* **Tools**: scan, classify, lineage\_push, uc\_external\_location
* **Verify**: classifications present, lineage edges

**6) mcp\_mlflow**

* **Tools**: train, register, promote, batch\_score
* **Verify**: metrics thresholds, registry state, output table counts

**7) mcp\_sql\_case\_pbi**

* **Tools**: sql\_exec, schema\_ensure, pbi\_refresh, pbi\_rbac\_set
* **Verify**: schema version, refresh success

Wrap all with your syntropAIkit.mcp base (shared BaseSession, retry, idempotency, result adapters).

**Message & state model**

* **Task**: {id, title, env, agent, tool, args, dry\_run}
* **Result**: {task\_id, ok, artifacts[], metrics{}, verify[]}
* **KB graph nodes**: Service, Dataset, Pipeline, Model, Dashboard, CaseTable
* **Edges**: produces, consumes, runs\_on, governed\_by, alerts\_to

Store KB in your file-backed knowledge graph so the Master can **plan with context** (e.g., “alerts table exists? last row time?”).

**Execution plan (3 sprints)**

**Sprint 0 — Skeleton & safety (2–3 days)**

* Master agent service with **Plan→Gate→Execute→Verify** loop.
* Implement mcp\_databricks.jobs\_run and mcp\_synapse\_sql.exec\_query first.
* Set **dev** environment only; no prod creds.
* Cluster policy: **no public IP, runtime version pinned, max node types**.

**Sprint 1 — Data path automation (4–5 days)**

* mcp\_adf\_eventhubs (copy sanctions → raw), mcp\_databricks.dlt\_create\_update (watchlists + rules).
* mcp\_synapse\_sql.create\_view for vw\_alerts\*.
* mcp\_sql\_case\_pbi.schema\_ensure + CDF sync job via mcp\_databricks.jobs\_run.
* Observability: job run workbooks + cost alerts.

**Sprint 2 — Governance & ML (5 days)**

* mcp\_purview\_uc to set UC external locations/credentials and Purview scans.
* mcp\_mlflow.train/register/promote/batch\_score to produce alerts\_scored.
* Power BI dataset refresh via mcp\_sql\_case\_pbi.pbi\_refresh.
* Add **prod** environment with approval gates.

**Concrete “recipes” the Master will run**

1. **Deploy Step-4 rules to dev**
   * Plan: import notebooks → create/update DLT pipeline → trigger run → verify tables/rowcount → create Synapse views → refresh PBI dev.
2. **Promote to prod**
   * Gate: human approval; UC grants check; cluster policy compliance.
   * Execute same as dev with prod workspaces; verify before/after metrics.
3. **Retrain triage model weekly**
   * Train on last 90 days → AUC ≥ target → promote to Staging → batch backtest → approval → promote to Prod → score last 7 days.

**Guardrails & approvals**

* **Dry-run always first** (TF plan, DLT validate, SQL compile).
* **Change windows** and **canary runs** (subset of data or separate target schema).
* **Policy checks**: Private Endpoints enforced, CMK on storage, UC grants bounded to groups.
* **Human-in-the-loop**: Any prod UC grant, model promotion, or ADF trigger to prod requires approval.

**Minimal code stubs (just to anchor)**

**Master→Agent task (JSON)**

{

"title": "Create/Run DLT: baseline rules",

"env": "dev",

"agent": "databricks",

"tool": "dlt\_create\_update",

"args": {

"catalog": "aml",

"target\_schema": "silver",

"notebooks": ["/Repos/aml/dlt/watchlists.py", "/Repos/aml/dlt/rules\_baseline.py"],

"continuous": true,

"storage": "abfss://logs@<sa>.dfs.core.windows.net/dlt/aml-core"

},

"dry\_run": false

}

**Agent result (compact)**

{

"task\_id": "T-1023",

"ok": true,

"artifacts": [{"pipeline\_id": "abc123", "tables": ["aml.silver.watchlists","aml.gold.alerts"]}],

"verify": [{"sql":"select count(\*) from aml.gold.alerts","rows":1204}]

}