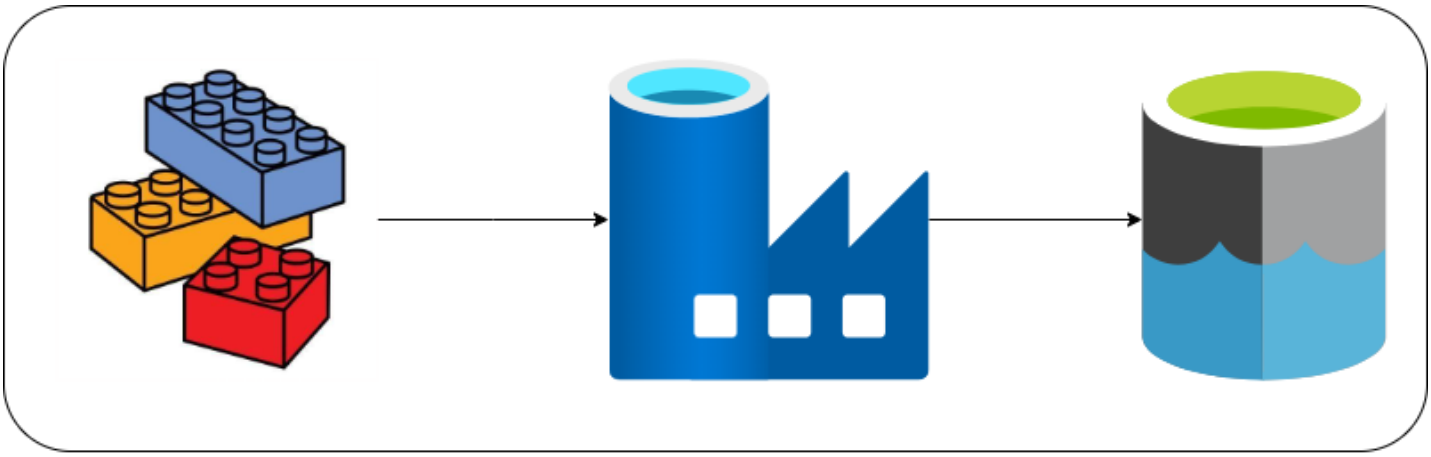
End-to-end implementation plan for the Rebrickable ingestion challenge.

Azure Data Ingestion Pipeline



A streamlined solution using:

- **Azure Data Factory (ADF)** – Ingestion & orchestration
- **Logic Apps/Monitor** – Notifications & alerts
- **Key Vault** – Secure secrets (managed identity)

Data Sources:
✓ Daily zipped CSV downloads
✓ REST API (“users” endpoint)

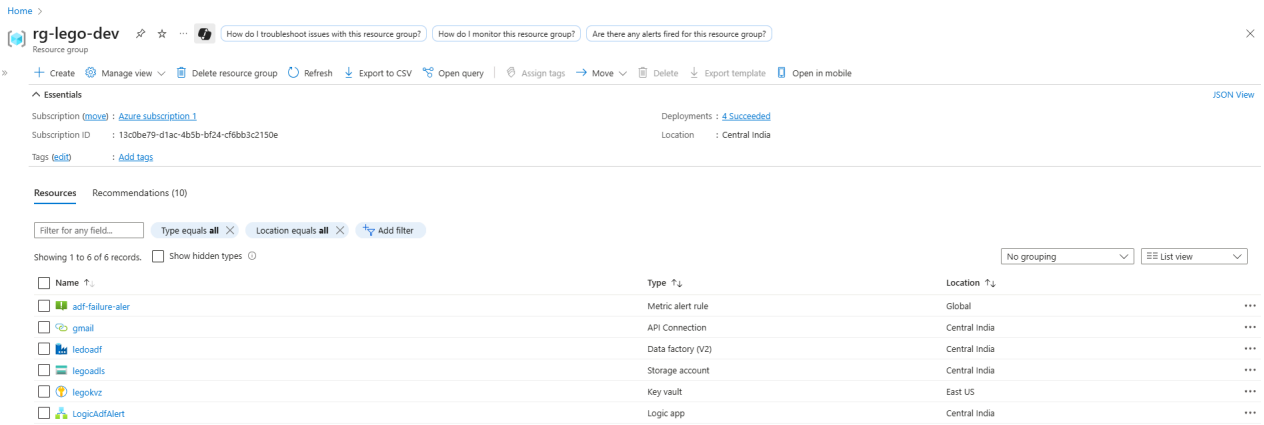
Key Features:

- **Config-driven** – Add sources without code via JSON config file
- **Structured ADLS Gen2** – Raw container with logical segmentation (`/lego` , `/users`), dataset folders, and date partitioning
- **Environment-aware** – Dev (LRS) & Prod (GRS) with CI/CD pipelines
- **Reliable & monitored** – Scheduled runs with failure alerts
  Built for scalability, security, and maintainability.

1) Target architecture

- Resource groups
  - rg-lego-dev (development)
  - rg-lego-prod (production)
- Core resources in each RG
  - ADLS Gen2 storage account with hierarchical namespace enabled (Dev=LRS, Prod=GRS)
  - Azure Data Factory
  - Azure Key Vault

- Entra ID groups for role-based access

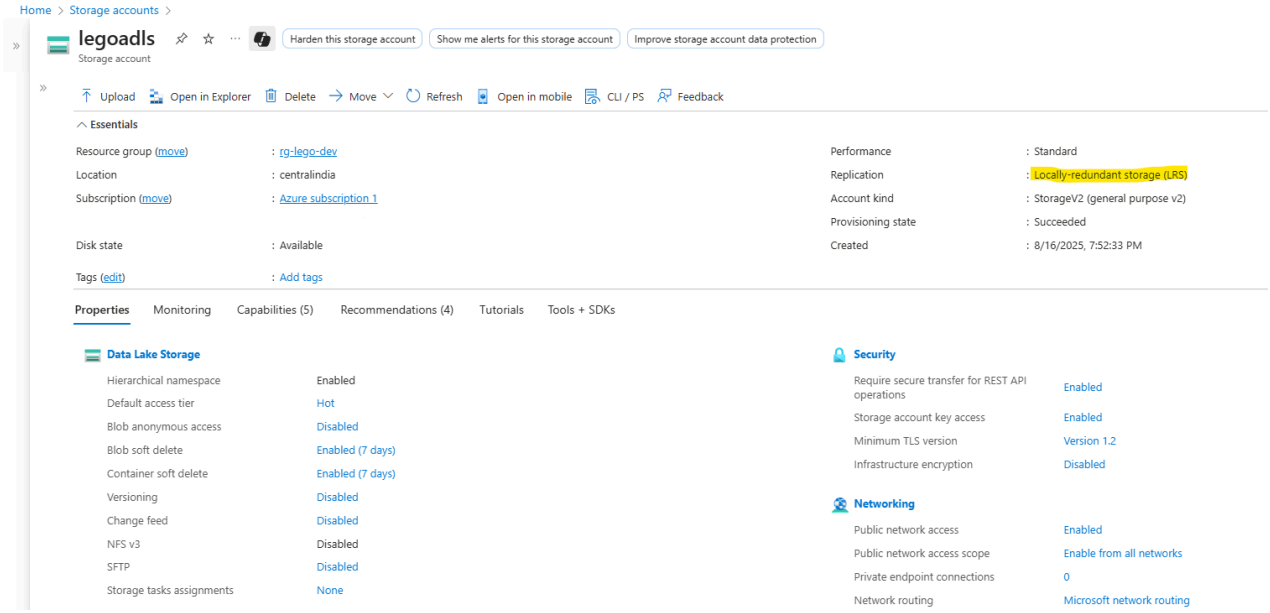


Data sources:

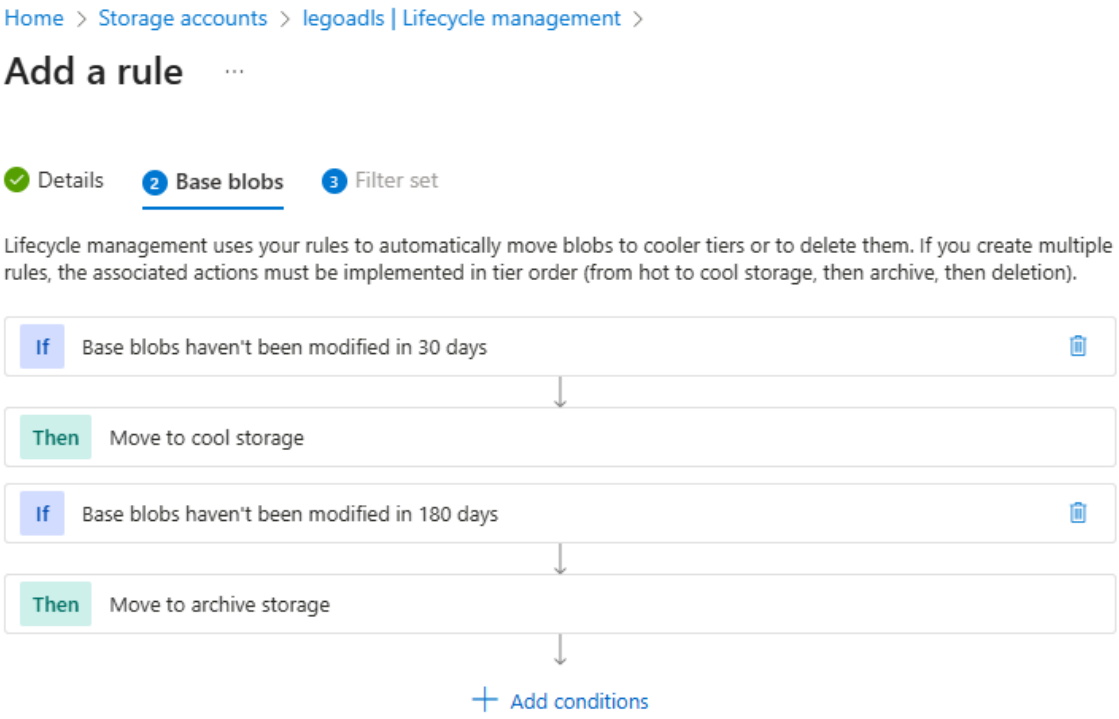
- Downloads site for daily bulk CSVs (e.g., minifigs, sets, inventories) updated once per day.
- REST API for “users” domain: requires API key auth and, for user-specific endpoints, a user token workflow; all API calls must include Authorization header with prefix “key ”. (From the Website)

2) ADLS Gen2 design and data layout

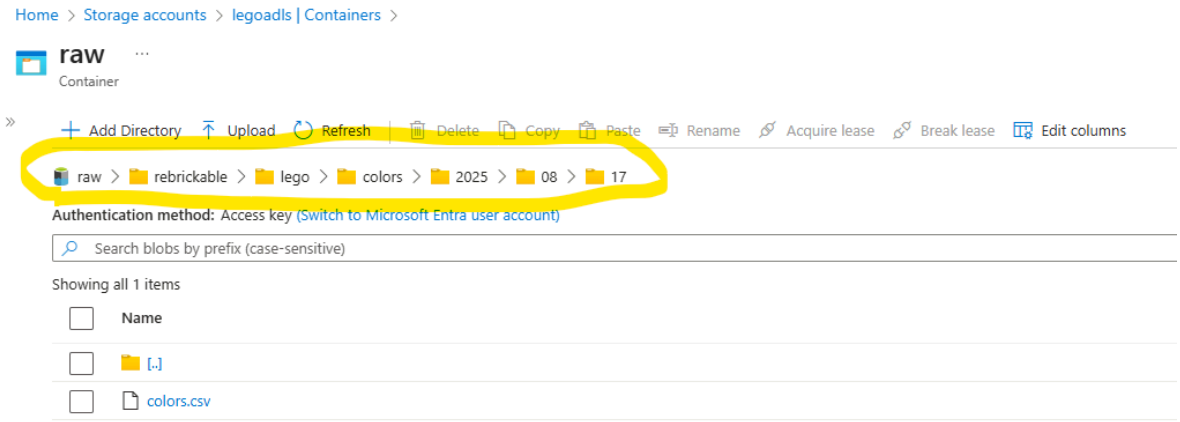
- Storage account
  - Dev redundancy: LRS; Prod redundancy: GRS (Geo-redundant), per requirement.



- Lifecycle management and tiers
  - Downloads CSVs are large but re-published daily; default Hot for 30 days, auto-tier to Cool after 30 days, and optional Archive after 180 days (adjust per cost/perf policy).



- Container and folder structure
  - Container: raw
  - Hierarchy:
    - raw/rebrickable/lego/YYYY/MM/DD/
    - raw/rebrickable/users/YYYY/MM/DD/



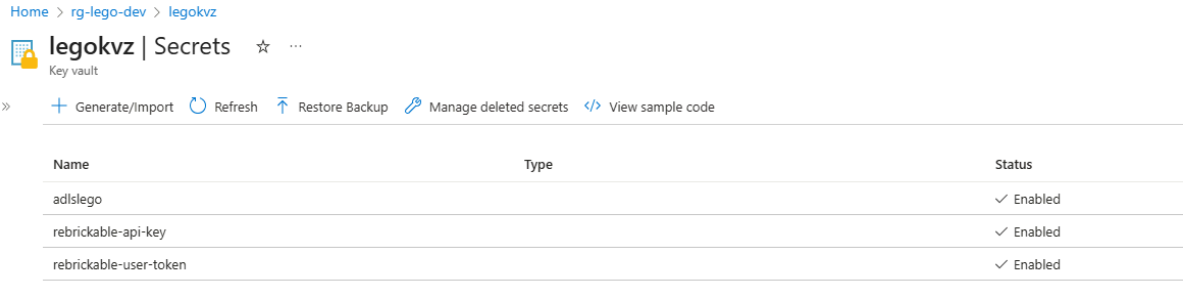
This structure is created dynamically at runtime per dataset, for both “lego” (downloads) and “users” (API) domains, matching the request for flexible, partitioned ingestion.

Rebrickable data relationships provided on the download page help later modeling; initial ingestion stores native formats (CSV from downloads; JSON from API).

3) Security and access

- Managed identities
  - Enable ADF system-assigned managed identity.
  - Grant access to ADLS via RBAC roles:
    - Storage Blob Data Contributor on the storage account (scope: container raw, or storage account if preferred).
- Key Vault
  - Secrets:

- rebrickable-api-key
- rebrickable-user-token (if needed for specific “users” endpoints)
- Access policy or RBAC: grant ADF managed identity get/list secrets.



4) Configuration-driven ingestion

Create a “ingestion_config.json” configuration file (JSON) in ADLS.

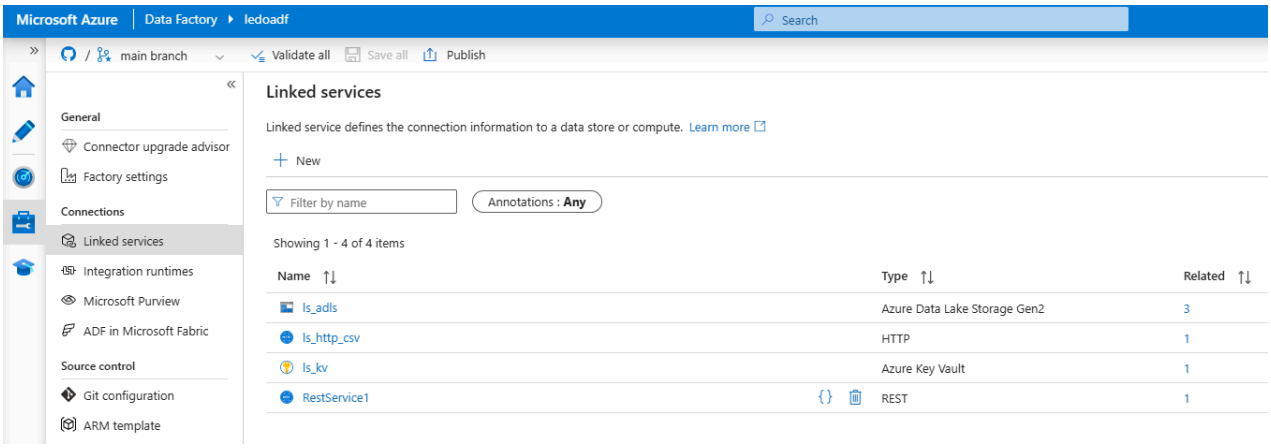
- For downloads (“lego” domain)
  - baseUrl: “**https://rebrickable.com/downloads/”** (static page hosts links; actual CSV URLs are direct file links referenced there)
  - datasets: array of objects:
    - name: “minifigs” (or other CSV sets such as “sets.csv”, “inventories.csv”, “inventory_minifigs.csv”, etc.)
    - fileUrl: fully qualified direct link to the daily CSV (see the downloads page each dataset’s direct CSV link)
    - compression: “zip” or null (downloads are zipped)
    - fileType: “csv”
- For API (“users” domain)
  - baseUrl: “**https://rebrickable.com/api/v3/”**
  - datasets: array of objects:
    - name: “users_minifigs” (example)
    - relativePath: “users/minifigs/” plus query params as needed for the endpoint (document exact endpoint chosen)
    - pagination: details (token name, next URL, or offset/limit scheme)
    - rateLimits: maxRequestsPerMinute, requestIntervalMs

The downloads are published daily; if the operator needs to add another CSV later, only edit this config to add the dataset entry—no ADF recoding required.

5) ADF linked services and datasets

- Linked services
  - Azure Data Lake Storage Gen2 (MI auth)
  - HTTP (for downloads CSV files)
    - Base URL can be blank; actual file URLs come from config and are per-dataset.
  - REST (for Rebrickable API)
    - Base URL: **https://rebrickable.com/api/v3/**
    - Authentication: None or Basic set to “Anonymous”; add Authorization header dynamically in Source settings (“key ”) per docs.
  - Azure Key Vault (to retrieve API key and user token)
- Datasets

- ADLS Gen2 dataset for sink (binary sink for zipped files; delimited text sink for CSV after unzip; JSON sink for API)
- REST dataset for API source (relativePath parameterized)
- HTTP dataset for files (URL parameterized)



6) Pipelines and activities

Create a master pipeline “PL_Ingest_Rebrickable” that executes two child pipelines in parallel:

- PL_Ingest_Downloads (lego)
- PL_Ingest_API (users)

Both read the same ingestion_config.json and loop through datasets of their domain.

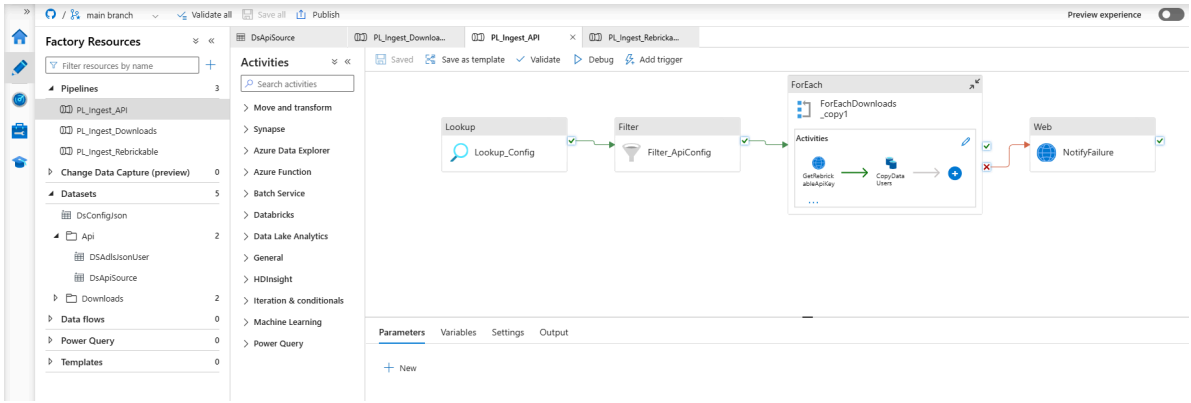
6.1) PL_Ingest_Downloads (lego)

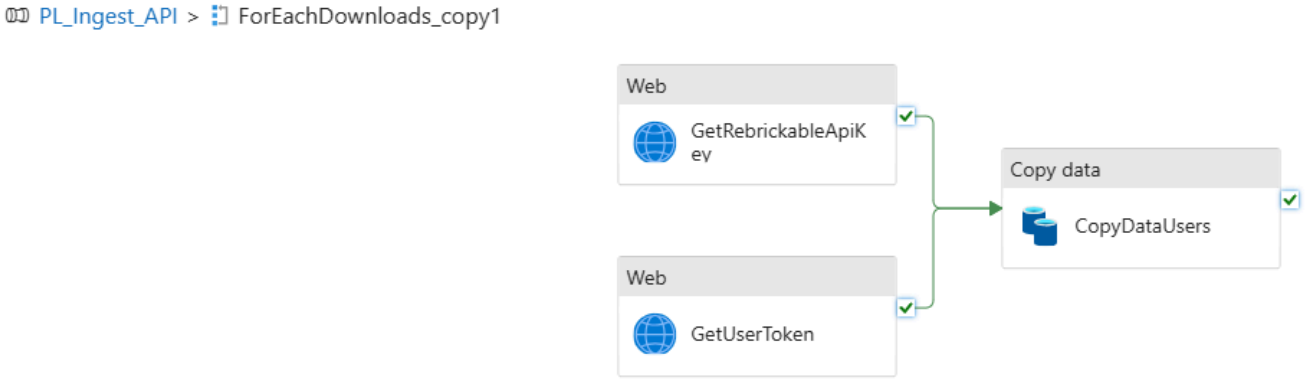
- Get Metadata/Lookup: Read ingestion_config.json (filtered to domain=lego).
- ForEach over lego datasets:
  - Copy Activity (HTTP to ADLS Binary)
    - Source: HTTP dataset with fileUrl param
    - Sink: ADLS Gen2 Binary to path, e.g. raw/rebrickable/staging/lego/@{item().name}/@{formatDateTime(utcnow(),'yyyy')}/@{formatDateTime(utcnow(),'MM')}/@{formatDateTime(utcnow(),'dd')}/part-*.csv
    - Use request retry and backoff in ADF

6.2) PL_Ingest_API (users)

- Get Metadata/Lookup: Read ingestion_config.json (filtered to domain=users).
- Secure headers
  - Add Source additional headers in Copy activity:
    - Authorization: “key @{linkedService().RebrickableApiKey}” loaded via Key Vault reference.
  - The endpoint requires a separate user token, add the appropriate header or query parameter per the endpoint’s spec; store token in Key Vault and pull into header/param.
- ForEach over users datasets:
  - Copy Activity (REST to ADLS JSON)
    - REST dataset with parameterized relativePath (from config)
    - Sink: ADLS Gen2 to @{item().name}/@{formatDateTime(utcnow(),'yyyy')}/@{formatDateTime(utcnow(),'MM')}/@{formatDateTime(utcnow(),'dd')}/part-*.json

- File pattern: Set sink to “Array of objects” or “Set of objects” for valid JSON if responses are concatenated across pages.





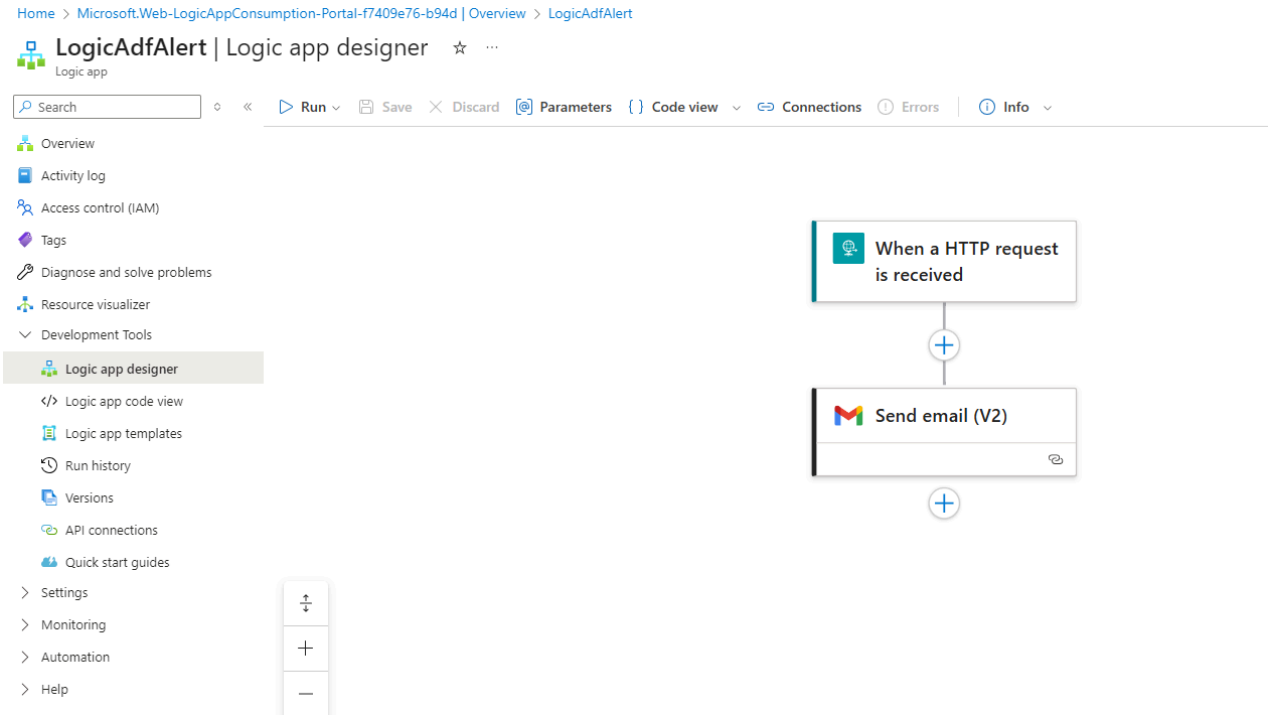
7) Scheduling and orchestration

- Trigger: Daily Schedule Trigger set for after Rebrickable refresh window (downloads page says files are “automatically updated daily”; schedule a safe time, e.g., 09:00 .
- Dependency: PL_Ingest_Rebrickable triggers children in parallel

8) Error handling and notifications

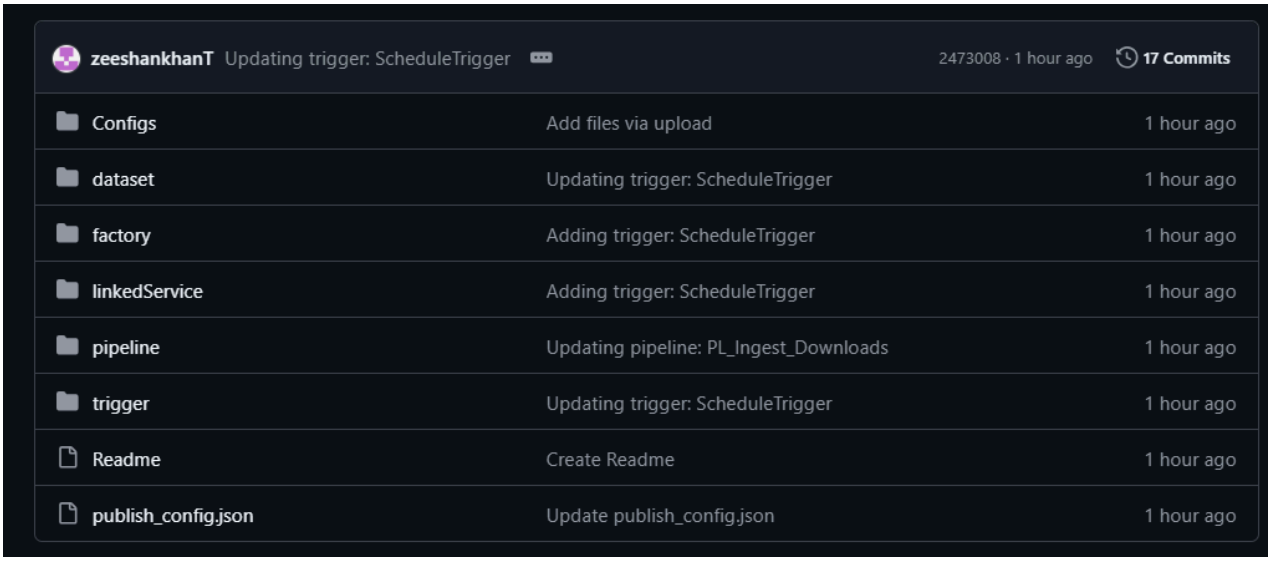
- Activity retry policies configured (e.g., 3 retries with exponential backoff) on Copy activities for transient faults.
- On Failure path:

- Webhook (Logic App) activity to send email via an Action Group.



9) CI/CD and dev-prod parity

- Git integration
  - Connect ADF to Azure Repos Git repository.
  - Work in feature branches; PRs to main.
- Release pipelines
  - Export ARM templates from ADF “Publish” .
  - Two environments: dev and prod.
  - Parameterize environment-specific values:
    - Storage account names
    - Key Vault names
    - Email recipients
- Deployment strategy
  - Either full ARM deployment or selective deployment (by pipeline/folder) per preference
  - Ensure no manual steps are required; upon PR merge, release pipeline deploys to Dev, then approval to Prod.



10) How to add a new dataset in 10 minutes (no ADF recoding)

- Edit ingestion_config.json and add:
  - For downloads: name, fileUrl, compression, fileType.
  - For API: name, relativePath
- upload to ADLS config folder.
- Next scheduled run picks it up automatically and creates dynamic folders/partitions.

11) Requirements Achieved

- Uses ADF for ingestion and orchestration, Logic App/Monitor for notifications, and Key Vault for secrets, following managed identity best practices.
- Handles both sources: daily zipped CSV downloads and REST “users” endpoints.
- Flexible, no-recode expansion through a config file controlling endpoints/files and parameters.
- Proper ADLS structure with raw container, source segmentation (lego vs users), per-dataset folders, and date partitioning.
- Environment separation with Dev(LRS) and Prod(GRS) and CI/CD pipelines for automated deployments.