ETL Pipeline Planning Questions

2025-02-11

Table of contents

1 Data Source Access

1.0.1 How will you handle different API rate limits across these diverse systems?

Answer

We'll implement a centralized rate limiting solution using DynamoDB to store and manage rate limits for each ERP system. Our approach includes:
A DynamoDB table storing each ERP's configuration including:
Rate limits and burst limits Contact information Alert thresholds Version history Approval status
AWS API Gateway for enforcing these limits, with usage plans dynamically configured based on the DynamoDB data. A Lambda function that syncs DynamoDB configurations with API Gateway usage plans, ensuring rate limits are always up to date. CloudWatch monitoring to track API usage against limits, with automated alerts at configurable thresholds (e.g., 80% warning, 95% critical). An admin API for managing rate limits with an approval workflow for any changes.
This solution provides centralized management, version control, audit trails, and automated monitoring - all essential for managing multiple ERP integrations at scale."

1.0.2 What's your backup plan for ERPs that don't provide API access?

Answer

For our ETL pipeline, we have specific strategies for each type of ERP system:
API-Based ERPs:

- Implement centralized rate limiting using DynamoDB to store API configurations
- Use AWS API Gateway to manage and enforce rate limits
- Set up CloudWatch monitoring for usage tracking
- Deploy Lambda functions for each API integration

Database-Driven ERPs:

- Use AWS Database Migration Service (DMS) for continuous replication
- Set up read replicas to minimize load on source systems
- Implement Change Data Capture (CDC) to track only changed records
- Use AWS Secrets Manager for database credentials

File-Based ERPs:

- Create dedicated S3 buckets for each system

- Use AWS Transfer Family for secure SFTP access
- Deploy Lambda functions to monitor and process new files
- Implement file integrity checks and validation

Legacy Systems:

- Custom extract programs for regular data exports
- Scheduled batch processes for data extraction
- File-based transfers to S3 using AWS Transfer Family
- Validation checks for data completeness

Common Infrastructure Across All Types:

- CloudWatch monitoring and alerting
- Data validation before loading
- Audit logging of all transfers
- Error handling and retry mechanisms
- Centralized credential management using AWS Secrets Manager

1.0.3 How will you maintain and secure credentials for 30+ different systems?

Answer

We'll implement a two-part credential management system:

1. AWS Secrets Manager for sensitive credentials:
    - All ERP credentials (API keys, passwords)
    - Automatic credential rotation
    - Built-in encryption using KMS
    - IAM role-based access control
2. DynamoDB for configuration and references:

```
{
    "erpId": "erp_a",
    "version": "v1",
    "secretArn": "arn:aws:secretsmanager:region:account:secret:erp-a-credentials",
    "integration": {
        "type": "API-Based",
        "method": "REST"
    },
    "metadata": {
        "owner": "team_a",
        "contact": "team_a@company.com"
    }
}
```

Lambda functions access credentials by:

1. Fetching configuration from DynamoDB
2. Using secretArn to retrieve actual credentials from Secrets Manager
3. Each Lambda has IAM roles with least-privilege access

This provides:

- Secure credential storage
- Central configuration management
- Audit logging through CloudWatch
- Automated credential rotation
- Version control of configurations"

1.0.4 How will you handle API/connection failures?

Answer

Since our ETL pipeline runs once daily rather than real-time, our failure handling focuses on job completion certainty:
Clear Job Status Tracking

- Each daily job has a unique identifier
- We track success/failure status for each ERP system
- All errors are logged with specific failure reasons

Failure Recovery Process

- Failed jobs automatically retry up to 3 times
- One hour delay between retry attempts
- After all retries fail, triggers team notification
- Manual investigation required before next day's run

Validation Checks

- Verify data completeness before marking job as successful
- Compare record counts with expected ranges
- Check for data quality issues

Monitoring & Alerts

- Daily job completion status dashboard
- Immediate alerts for failed jobs
- Historical job status tracking for pattern analysis

This approach ensures we always know whether each day's data fetch succeeded or failed, and exactly why any failures occurred.

2 Data Standardization

2.0.1 Have you mapped the schema differences between all these ERPs?

Answer

We manage schema differences through a standardized mapping approach:
Central Data Dictionary

- Maintain a master schema in DynamoDB defining our standard format
- Each ERP has a mapping configuration showing how its fields translate to our standard
- Include data type conversions and formatting rules

Data Standardization Process

- Raw data is first stored in its original format
- Transformation layer converts to our standard schema
- All dates normalized to UTC
- Consistent naming conventions applied
- Field values standardized (e.g., 'Y'/'N' to true/false)

Quality Assurance

- Automated validation of transformed data
- Alerts for unexpected data formats
- Track any fields that can't be mapped
- Regular audits of mapping accuracy

2.0.2 How will you handle inconsistent field names/data types across systems?

Answer

We handle field name and data type inconsistencies through a transformation layer:
Field Name Standardization

- Use mapping tables in DynamoDB to define standard names
- Each ERP entry includes:
  - Original field name
  - Standardized field name
  - Business context
  - Required transformations

Data Type Handling

- Number formats (handle different decimal separators)
- Date/time formats (normalize to UTC)
- Text encodings (convert to UTF-8)
- Boolean values (convert Y/N, 1/0, True/False to standard boolean)

Validation Rules

- Required fields checking
- Data type verification
- Range/format validation
- Business rule validation

Error Handling

- Tag records with transformation errors
- Keep original values for audit
- Alert on systematic conversion issues
- Maintain transformation logs

2.0.3 What's your plan for handling different date/time formats from various systems?

Answer

We standardize all date/time handling through a consistent process:
Date/Time Standardization

- Convert all timestamps to UTC during ingestion

- Store original timezone information in metadata
- Use ISO 8601 format (YYYY-MM-DDTHH:mm:ss.sssZ) as our standard
- Maintain source format for audit purposes

Common Format Handling

- Support multiple input formats:
  - US format (MM/DD/YYYY)
  - European format (DD/MM/YYYY)
  - Unix timestamps
  - Various timezone notations
  - Date-only formats
  - Custom ERP formats

Edge Cases

- Handle invalid dates
- Account for daylight savings transitions
- Process missing timezone information
- Manage partial dates (month/year only)

Quality Controls

- Validate all conversions
- Flag impossible dates/times
- Alert on systematic conversion issues
- Regular audits of timestamp accuracy

2.0.4 How will you maintain data lineage tracking across these diverse sources?

Answer

We implement a two-stage data lineage tracking system:
Raw Data Stage (Pre-Iceberg)

- Landing Zone Tracking
  - Original compressed files/CSVs logged in DynamoDB
  - Source ERP system
  - File metadata (size, timestamp, checksum)
  - Processing status

- Batch/job ID

Transformed Data Stage (In Iceberg)

- Leverage Iceberg's features for:
  - Snapshot history
  - Schema evolution
  - Time travel capabilities
  - Transaction logs

Complete Lineage Chain

- Link raw and transformed stages through:
  - Unique batch IDs
  - Processing timestamps
  - Source file references
  - Transformation logs

Audit Capabilities

- Raw data: Track original file to transformation
- Transformed data: Use Iceberg's features
- End-to-end tracing from source file to final table

3 Processing & Storage

3.0.1 What's your strategy for handling late-arriving data?

Answer

Scenarios:

- An ERP system was down during the regular daily pull
- Someone enters January sales data in March
- A business unit delays their data entry (enters last week's data today)
- An ERP has corrected/adjusted historical data

Our strategy for handling late or backfilled data in daily batches:
Identification

- Compare record dates vs current processing date

- Flag any data older than previous day
- Identify which historical partitions need updates

Processing

- Store data in Iceberg tables by business date, not ingestion date
- Use MERGE operations to handle updates to historical data
- Keep log of all backfilled data in DynamoDB for auditing

Monitoring

- Alert if we see unusual amounts of historical data
- Track which ERPs frequently send late data
- Report on data entry lag times

3.0.2 How will you handle schema evolution in your Iceberg tables?

Answer

Our schema evolution strategy uses Iceberg's built-in capabilities:
Schema Changes

- Add new columns as NULLABLE to maintain compatibility
- Track all schema changes in version control
- Document business reason for each change
- Test impact on downstream processes

Backward Compatibility

- Keep previous schema versions accessible
- Maintain default values for new columns
- Document field mappings for each version
- Support queries across schema versions

Migration Process

- Roll out schema changes in phases
- Validate data quality after changes
- Keep historical data accessible
- Update documentation and data dictionaries

Monitoring & Control

- Track which schema version each ERP uses
- Alert on unexpected schema changes
- Regular schema compatibility checks
- Audit schema change history

3.0.3 What's your plan for data validation before loading into S3?

Answer

Our validation strategy leverages AWS Glue and PySpark capabilities:
Drop Zone Validation (Initial S3 Landing)

- AWS Lambda performs quick checks:
  - File presence verification
  - Basic file integrity
  - Naming convention compliance
  - File size validation
  - Expected file count per ERP

Pre-Iceberg Validation (Glue Job)

- PySpark validations during transformation:
  - Schema conformance
  - Data type verification
  - Required field checks
  - Business rule validation
  - Record count verification
  - Date range completeness

Error Handling

- CloudWatch logs capture validation failures
- Glue job bookmarks track processed files
- Glue metrics monitor validation stats
- DynamoDB stores validation history
- Failed files moved to error prefix
- Alert notifications for validation failures

3.0.4 How are you handling data versioning and rollbacks?

> Answer

4 Operational Concerns

4.0.1 What's your monitoring strategy for pipeline failures?

> Answer

4.0.2 How will you handle partial load failures?

> Answer

4.0.3 What's your data retention policy in the drop zone?

> Answer

4.0.4 How will you manage pipeline dependencies between different ERP loads?

> Answer

5 Additional Questions

5.0.1 What are the data sync frequency requirements for each system?

> Answer

5.0.2 Are there any specific SLAs for data freshness that need to be met?

> Answer