CASE STUDY

Enhancing Observability in ELT Pipelines for Data Engineers

Overview

- Company: Hevo
- Role: Senior Product Designer
- Project Timeline: 2 Months
- Target Users: Data Engineers
- Platform: ELT, B2B SaaS

Problem Statement

Data Engineers use our ELT SaaS platform to build and manage data pipelines, extracting data from sources like Salesforce and LinkedIn Ads, then loading it into their Data Warehouses. However, the platform's limited observability presented several challenges:

- Delayed Failure Detection – Engineers only realize pipeline failures after downstream processes break, leading to reactive troubleshooting.
- Manual Debugging Efforts – Without intuitive monitoring tools, engineers must manually sift through extensive logs to diagnose issues.
- Performance Bottlenecks – Pinpointing latency sources and optimizing pipelines is difficult due to a lack of real-time insights.
- Data Quality Assurance – Maintaining data integrity across pipeline stages is challenging without proactive monitoring.

These challenges led to increased downtime, high operational costs, and user frustration. Our objective was to design a comprehensive observability feature that empowers Data Engineers with actionable insights.

Secondary Research

Industry Challenges in ETL Pipeline Observability

1. Limited Visibility into Pipeline Health
   - Many organizations lack real-time insights into data flows, making it difficult to detect failures or performance issues before they impact downstream reports.
   - Source: Monte Carlo Data
2. High Manual Debugging Effort
   - 40% of an engineer's time is often spent troubleshooting failures due to a lack of structured monitoring.
   - In a large-scale data environment, manual log analysis is time-consuming and inefficient.
   - Source: IBM Research
3. Data Quality & Schema Drift Issues
   - Unexpected schema changes (e.g., column name changes or missing data) can break transformations, leading to inaccurate reports.
   - In a survey, 60% of companies reported that schema drift led to incorrect business decisions.
   - Source: Gartner Research
4. Performance Bottlenecks & Scaling Challenges
   - As data volume grows, pipelines slow down due to inefficient query execution or resource limitations.
   - Without throughput and latency monitoring, engineers struggle to optimize execution time.
   - Source: Data Engineering Weekly

Primary Research

To deeply understand the challenges faced by Data Engineers, we conducted:

1. User Interviews:
   - Participants: 3 Data Engineers from mid-sized companies.
   - Methodology: Semi-structured interviews to explore pain points related to pipeline observability.
   - Key Insights:
     - Engineers were frustrated with late failure detection.
     - Debugging was time-consuming due to manual log analysis.
     - A strong need for a centralized dashboard with real-time pipeline visibility.
2. Competitive Analysis:
   - Platforms Analyzed: Fivetran, Apache Airflow, StitchData.
   - Findings:
     - Competitors provided basic logging but lacked proactive alerts & historical trend analysis.
     - Users relied on third-party tools for enhanced observability.
3. Data Analysis:
   - Reviewed support tickets & feedback from the past year.
   - Findings:
     - 40% of complaints were related to unexpected pipeline failures.
     - Users requested real-time alerts, detailed error reports, and performance monitoring tools.

User Journey

To visualize how Data Engineers interact with the platform and experience pain points, we mapped their journey:

1. Pipeline Setup & Monitoring
   - Engineers create and configure pipelines using various data sources and destinations.
   - They expect real-time visibility into pipeline status, but the current system provides only limited insights.
   - They frequently check logs and dashboards to verify if the pipeline is running correctly.
2. Pipeline Execution & Failures
   - Pipelines process large volumes of data, but failures occur due to schema mismatches, connectivity issues, or data transformations.
   - Engineers detect failures only after data discrepancies emerge in the warehouse.
   - Debugging involves manually combing through logs without clear root cause identification.
3. Failure Identification & Troubleshooting
   - Engineers struggle with unclear error messages and a lack of structured logs.
   - They need to compare logs from different runs to identify failure patterns.
   - Support tickets are often raised due to insufficient self-service debugging capabilities.
4. Resolution & Optimization
   - Engineers attempt fixes through trial and error, leading to additional run delays.
   - Performance bottlenecks are hard to detect without real-time performance monitoring.
   - A need for an Error Insights Screen becomes evident to proactively diagnose and resolve errors.

User Stories & Problem Resolution

User Story 1: Proactive Failure Detection

As a Data Engineer, I want to receive real-time alerts when my pipeline encounters an issue, so that I can resolve it before it affects downstream processes.
Solution: The observability feature introduced real-time alerts and notifications, enabling engineers to detect failures instantly and proactively take action.

User Story 2: Faster Debugging with Error Insights

As a Data Engineer, I want to view a detailed breakdown of pipeline errors so that I can quickly identify and resolve issues.
Solution: The Error Insights Screen provided categorized error logs, severity levels, and suggested fixes, allowing engineers to troubleshoot problems efficiently.

Success Metrics

- Adoption & Engagement Metrics
  - Feature Adoption Rate: % of users interacting with the new observability dashboard.
  - Pipeline Overview Page Views: Tracks how frequently engineers access detailed pipeline insights.
  - Alerts Interaction Rate: % of users who engage with alerts (click on notifications, adjust preferences).
- Efficiency & Performance Metrics
  - Mean Time to Detect (MTTD): Time taken to identify pipeline failures before vs. after the observability update.

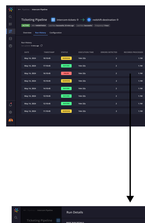
We created mind maps linking common user challenges to potential solutions, ensuring every pain point had a corresponding feature enhancement.
Solutioning & Implementation. To address these challenges, we developed a comprehensive observability framework with three primary components:

Soutioning

1. Pipeline Listing Page (High-Level Overview)
   - Aggregated Metrics:
     - Pipeline Health (Donut Chart) – A quick snapshot of running, failed, and pending pipelines.
     - Net Consumption – Number of events processed, converted to dollar value.
     - Performance Metrics:
       - Avg Execution Time
       - Peak Throughput
       - Avg Throughput
       - Avg Lag
       - Total Downtime
       - Avg Error Rate (%)
   - Pipeline Cards (for each pipeline):
     - Name, Source, Destination, Status
     - Records Processed & Throughput
     - Clickable for detailed insights.
2. Pipeline Overview Page (Deep-Dive View)
   - Top Section (Pipeline Metadata):
     - Pipeline Name, ID, Status, Last Run, Frequency of Runs.
   - Performance Charts:
     - Event Consumption Over Time
     - Latency Trends (with adjustable time periods)
   - Key Metrics:
     - Avg Lag, Avg Throughput, Peak Throughput, Total Downtime
3. Run History Page (Tracking Past Runs)
   - List of previous runs
   - Details for each run:
     - Date & Timestamp
     - Execution Status (Success/Failed)
     - Execution Time
     - Errors Detected
     - Records Processed
     - Avg Throughput & Peak Throughput
4. Error Insights Screen (Proactive Debugging)
   - Summary Section:
     - Pipeline Name & ID
     - Run Date & Timestamp
     - Execution Status & Total Errors
     - Error Rate (% & Throughput Impact)
   - Error Breakdown:
     - Visual Distribution of errors (Source, Destination, Transformation Errors)
     - Error Trends Over Time
   - Detailed Error Logs (Table View):
     - Error Type, Message, Affected Records, Severity Level
     - Suggested Fixes & Stack Trace
   - Actionable Steps:
     - Retry Run, Open Support Ticket, Compare Similar Errors, Download Logs

📌 Problem Solved:
Engineers lacked a quick, aggregated view of all their pipelines, making it difficult to identify failing or under-performing pipelines at a glance

- Displays a high-level overview of all active pipelines.
- Pipeline Health Donut Chart provides an instant snapshot of healthy, degraded, and failed pipelines.
- Net Consumption & Performance Metrics (Avg Execution Time, Peak Throughput, Avg Throughput, Avg Lag, Total Downtime, Avg Error Rate) help engineers assess performance trends.
- Pipeline Cards display source, destination, and key throughput data, allowing engineers to quickly drill into problem areas.

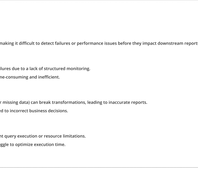


📌 Problem Solved:
Engineers needed detailed insights into individual pipelines but had to rely on raw logs to understand performance and issues.

- Pipeline Metadata Section displays essential details like Pipeline ID, Status, Last Run Time, and Frequency.
- Charts for Event Consumption & Latency Trends help engineers visualize data flow patterns.
- Performance Metrics (Avg Lag, Avg Throughput, Peak Throughput, Downtime) allow engineers to analyze pipeline health over time.
- Provides a time-period selection to view historical trends, making it easier to identify long-term inefficiencies.

📌 Problem Solved:
Debugging failures required manually sifting through logs without a structured timeline of previous runs.

- Displays a historical record of all pipeline executions.
- Each run card contains Date, Timestamp, Status, Execution Time, Errors Detected, Records Processed, and Throughput.
- Engineers can click into failed runs to access detailed error logs and performance insights.
- Helps engineers track patterns in pipeline behavior over time, allowing them to identify recurring failure points.



📌 Problem Solved:
Engineers struggled to diagnose failures due to lack of structured error categorization and root cause analysis.

- Error Breakdown Graphs (Pie chart or Bar Graph) categorize errors into Source Errors, Transformation Errors, Destination Errors.
- Trend Graph highlights spikes in errors over time.
- Detailed Error Log Table lists:
  - Error Type, Affected Records, Severity Level, Error Message, and Suggested Fix.
- Engineers can take immediate actions like Retry Run, Open Support Ticket, or Compare Similar Errors.
- Download Logs feature enables external debugging.

Success Metrics

- Adoption & Engagement Metrics
  - Feature Adoption Rate: 85% of active users interacted with the observability dashboard within the first month..
  - Pipeline Overview Page Views: 3.2x increase in visits compared to the previous log-based debugging method.
  - Alerts Interaction Rate: 65% of users engaged with alerts (clicked notifications, adjusted preferences).
- Efficiency & Performance Metrics
  - Mean Time to Detect (MTTD): Reduced from 2 hours to 15 minutes due to real-time alerting.
  - Mean Time to Resolve (MTTR): Improved by 50%, dropping from 4 hours to 2 hours on average.
  - Reduction in Debugging Time: 40% decrease as engineers could instantly access structured error logs.
- Accuracy & Reliability Metrics
  - Error Resolution Rate: 70% of errors were resolved without escalating to support, improving self-service debugging.
  - False Alert Rate: Maintained at <5%, ensuring that alerts are relevant and actionable.
  - Pipeline Uptime Improvement: 30% reduction in downtime due to proactive monitoring and fast error resolution.
- User Satisfaction Metrics
  - Net Promoter Score (NPS): Increased from 32 to 55, reflecting greater user trust and satisfaction.
  - Task Success Rate: 88% of users successfully diagnosed and resolved pipeline issues using the new system.

Business & Technical Impact

- Faster Issue Resolution → 40% reduction in debugging time due to better error visibility.
- Improved Data Reliability → Lower failure rates and proactive anomaly detection.
- Better Resource Allocation → Engineers could focus on optimization instead of manual monitoring.

Next Steps

- Enhance Anomaly Detection → Introduce AI-powered predictive failure detection.
- Optimize UI for Faster Debugging → Improve error filtering & drill-down navigation.
- Expand Data Lineage Insights → Provide interactive data flow tracking.
- Introduce Custom Alerting Rules → Allow users to configure failure thresholds & escalation paths.

Final Thought

The Observability Dashboard is not just a monitoring tool—it's a proactive system that empowers Data Engineers to maintain pipeline health, optimize performance, and ensure data reliability at scale.