**Solution Title:**  
Automated and Sustainable Data Lineage Harvesting and Management for ERFT Pipelines

**Solution Overview:**  
The proposed solution aims to automate the collection, management, and persistence of data lineage within ERFT data pipelines. By utilizing a combination of metadata harvesting, event-based triggers, and change detection mechanisms, this solution will provide a continuous, real-time view of data flow. The approach will leverage modern data lineage tools and frameworks to capture the journey of data from source to destination, including all transformations and system hops, ensuring regulatory compliance, transparency, and data quality. The solution will also incorporate mechanisms to handle evolving pipelines by automatically updating the lineage as the system changes, maintaining an up-to-date lineage map.

**Implementation Approach with Technical Details:**

1. **Data Lineage Harvesting/Gathering:**
   * **Metadata Extraction**: Implement metadata harvesting at critical points in the data pipeline (source, transformation, and destination) using automated tools such as Apache Atlas, OpenLineage, or custom ETL tool extensions.
   * **Event-Driven Harvesting**: Use change data capture (CDC) mechanisms or event-driven architectures (via Kafka, AWS Lambda, or Azure Event Grid) to trigger lineage harvesting when key data events (like schema changes, job executions) occur.
   * **Automated Lineage Mapping**: Integrate with pipeline orchestration tools (like Airflow, Prefect) to map each step of the data flow dynamically, capturing data movement between systems and transformations.
   * **Tools**: Apache Atlas, OpenLineage, DataHub, or Collibra can be leveraged for lineage tracking and visualization.
2. **Data Lineage Management and Sustainability:**
   * **Versioning and Change Detection**: Implement version control for data lineage, allowing the system to track changes over time as pipelines evolve. Use tools that support lineage versioning and comparison (such as OpenLineage or Apache Atlas).
   * **Scheduled Lineage Updates**: Use cron jobs or orchestrator-based scheduling to periodically check for changes in pipelines and update lineage accordingly.
   * **Automated Trigger-Based Updates**: Leverage event-driven architectures to trigger lineage updates when systems or data flows are modified, such as new transformations or updated data sources.
   * **Real-Time Monitoring**: Implement real-time monitoring of data pipelines using log parsing and event streaming platforms (e.g., ELK Stack or Kafka) to detect when lineage changes are necessary.
   * **Security and Compliance**: Ensure the data lineage system is compliant with relevant data security protocols, especially if dealing with sensitive or personal data.
3. **Data Lineage Persistence:**
   * **Graph Database Storage**: Store lineage data in a graph database like Neo4j, which is well-suited for modeling and querying complex relationships between data entities, transformations, and systems. This will allow for efficient querying of the lineage.
   * **API and Query Interface**: Provide a user-friendly API and graphical interface for querying and analyzing lineage data. Users can visualize data flows, trace back errors, and ensure compliance with regulatory standards.
   * **Data Governance and Auditing**: Implement auditing capabilities for compliance. Lineage should capture when and by whom data was modified, ensuring audit trails are readily available.
4. **Technical Stack:**
   * **Lineage Tracking**: Apache Atlas, OpenLineage, or custom-built solutions.
   * **Orchestration/Triggering**: Apache Airflow, Prefect, or custom event-driven triggers via Kafka or serverless solutions (AWS Lambda/Azure Functions).
   * **Storage**: Neo4j for graph-based persistence, Elasticsearch for log-based monitoring.
   * **Visualization**: Kibana, DataHub UI, or custom-built dashboards for querying and visualizing the lineage.

This automated and scalable approach ensures that data lineage within ERFT pipelines is continuously maintained and adapts as systems evolve, improving transparency, data governance, and regulatory compliance.