ARIESVIEW

AriesView – Recommendation Presentation

Created by Anish Reddy Illuri

ARIESVIEW

Recommendation 1:

Hybrid OCR for Structured Field Extraction

Deploying a hybrid OCR pipeline using DocTR with regex-based extractors will significantly improve AriesView's ability to capture key structured fields like "Base Rent" and "Start Date" from lease documents.

This enhancement increases document parsing accuracy, reduces downstream errors in financial models and clause detection, and boosts user trust in AI-generated outputs.

ARIESVIEW

Recommendation 1 : Problem Statement / Gaps

Problem: Inaccurate Extraction of Structured Fields in OCR Pipeline

Missing Key Fields: Lease documents include structured data such as “Base Rent,” “Start Date,” and “Tenant Name,” often embedded in tables or aligned columns. AriesView’s current OCR engine (PaddleOCR) frequently fails to extract these accurately, especially from scanned or rotated documents.
Lack of Layout Awareness: PaddleOCR lacks spatial context, causing it to misread or skip over tabular/key-value formats common in legal and lease contracts. This leads to inconsistent data extraction and unreliable downstream usage.

Problem: Inconsistent Output and Poor Downstream Performance
Due to these inaccuracies, AriesView’s document pipeline generates unreliable outputs that impact core features. Financial models may miscalculate rental income or escalations, and the AI chatbot often provides incomplete answers. Users are forced to manually review OCR results, delaying lease abstraction and reducing trust in the platform’s automation.

ARIESVIEW

Business Spec

Hybrid OCR for Structured Field Extraction — User-Focused Business Spec

Objective: Increase trust, accuracy, and speed in lease processing by improving the platform's ability to extract structured fields from legal documents.

User Benefit:

Users spend less time manually correcting OCR errors and more time analyzing insights. Lease review, risk flagging, and memo generation become faster and more reliable.

Business Impact:

Reduces analyst review time and manual QA costs. Increases confidence in AI-generated data and output. Enhances client satisfaction through faster and more accurate document processing.

Strategic Fit:

Supports AriesView's roadmap to reduce legal overhead and improve automation accuracy for underwriting and asset management workflows.

ARIESVIEW

Tech Spec

1. Hybrid OCR Pipeline
Keep PaddleOCR as default, but add a regex-based post-processor for extracting fields like "Base Rent" or "Start Date" from scanned leases. For complex layouts, use DocTR or LayoutParser to retain spatial structure.

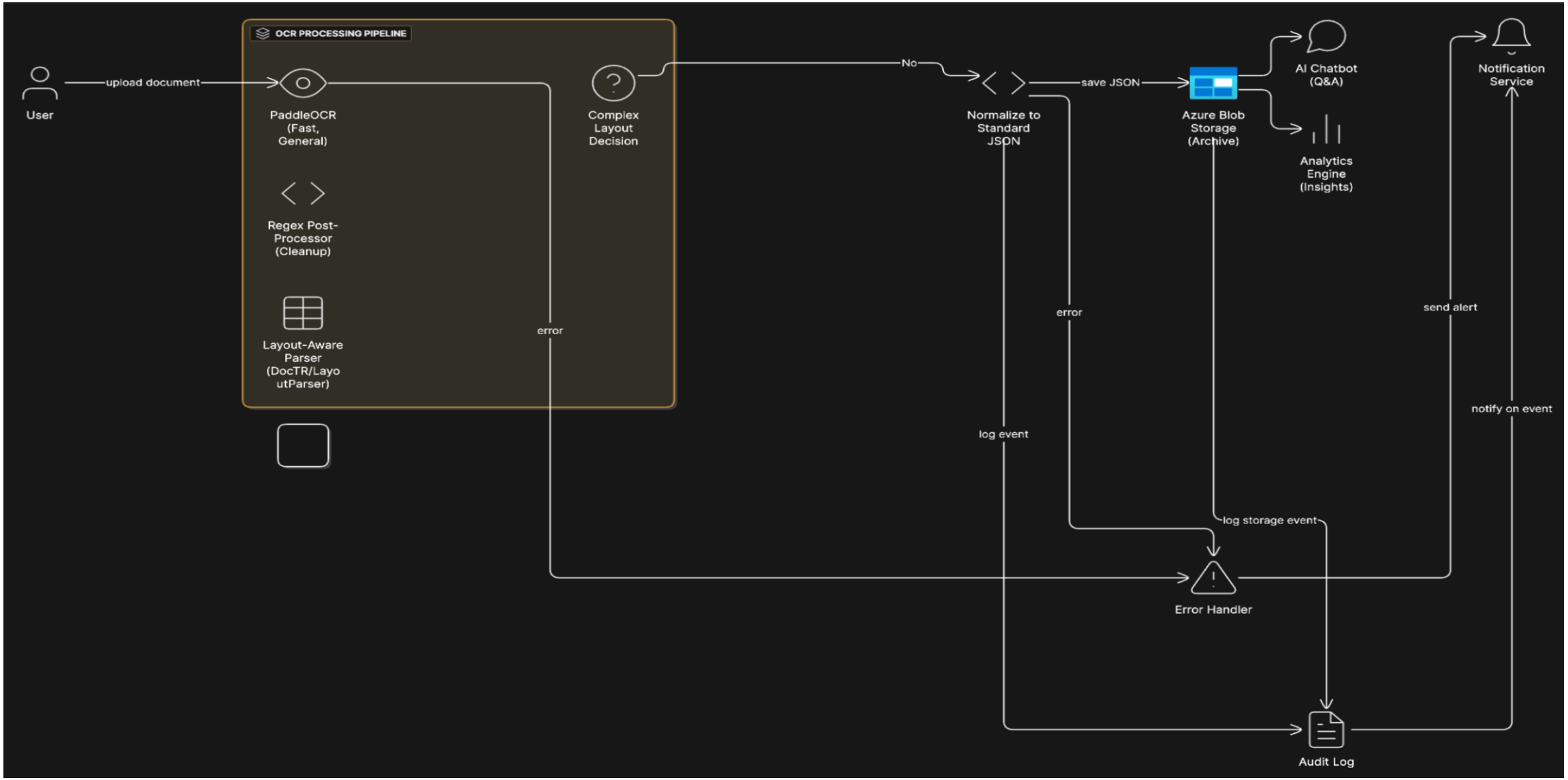
2. Parsing Logic
Run PaddleOCR → apply regex rules → apply layout-aware parsing (if needed) → output normalized JSON. Include confidence scores. Flag low-confidence fields for manual review.

3. Integration & Deployment
Add as a new microservice in existing pipeline. Use Python-based containers deployed via GitLab CI/CD. No changes to current storage or PDF parsing setup.

4. Tools & Dependencies
Python (PaddleOCR, regex, DocTR/LayoutParser), PyMuPDF (for digital PDFs), Azure Blob Storage, GitLab CI/CD. LayoutParser requires PubLayNet or Detectron2.

ARIESVIEW

Visual Design of the Architecture/Flow



ARIESVIEW

Visual Design of the Architecture/Flow

The diagram in the previous slide illustrates AriesView's **hybrid OCR pipeline** for intelligent document ingestion and processing. When a user uploads a lease document, it first enters the **OCR Processing Pipeline**, where:

1. **PaddleOCR** extracts raw text quickly.
2. A **Regex Post-Processor** cleans and structures this text.
3. A **Complex Layout Decision** node determines whether the document requires spatial parsing.
4. If needed, it routes the document to a **Layout-Aware Parser** like DocTR or LayoutParser.

Once the extraction is complete, the system **normalizes the output into standard JSON format** and stores it in **Azure Blob Storage**.

From there:

1. The **AI Chatbot** accesses the structured data to power contextual Q&A.
2. The **Analytics Engine** uses it to generate business insights.
3. If errors occur at any point, they are sent to a centralized **Error Handler**, which logs them in an **Audit Log** and notifies the **Notification Service** to alert the team or client.

This modular flow ensures speed, accuracy, and traceability across the entire document pipeline.

ARIESVIEW

Recommendation 2:

Automate Document Intake & Routing with N8n

AriesView's document intake and processing pipeline currently relies on custom API endpoints and manual steps to transform and route data. As new document types and data flows are added, this approach creates engineering bottlenecks and introduces inconsistencies.

We recommend introducing **N8n**, an open-source workflow automation tool, to orchestrate document intake, metadata tagging, user-specific routing, and downstream API triggering. This no-code/low-code platform can standardize intake workflows, reduce engineering effort, and improve speed, flexibility, and reliability across the document lifecycle.

ARIESVIEW

Recommendation 2 : Problem Statement / Gaps

Problem: Manual API Logic for Document Intake & Routing

Rigid Document Intake Logic: AriesView currently manages intake and routing through manually written API endpoints that are tightly coupled to specific document types and workflows. For each new lease format or client-specific rule, engineers must create custom logic — increasing development time, technical debt, and the chance for routing errors during ingestion.

Scaling & Maintenance Bottlenecks: As the number of document types and client needs grow, the current approach lacks flexibility and becomes harder to maintain. Adding new automations (e.g., auto-tagging, triage, or analyst assignment) requires editing backend code. This slows down onboarding timelines and reduces the team's ability to iterate quickly on process improvements.

ARIESVIEW

Business Spec

Automated Routing with N8n — User-Focused Business Spec

Objective: Streamline document intake and routing by using a visual, logic-based automation platform (e.g., N8n) instead of hardcoding each workflow rule.

User Benefit:

Users experience faster onboarding of new document formats and clients, with fewer routing errors. This reduces delays in document processing and ensures lease data reaches the correct downstream tools (chatbot, memo engine, analytics).

Business Impact:

Lowers developer burden and accelerates time-to-market for internal process updates. Enhances scalability of operations without increasing engineering overhead. Reduces cost of future maintenance by allowing business users or analysts to define routing logic without needing to write backend code.

Strategic Fit:

Aligns with AriesView's vision of modular, low-code automation. Supports long-term platform scalability, reduces tech debt, and enables faster experimentation with client-specific workflows.

ARIESVIEW

Tech Spec

1. Integrate N8n for Routing Automation

Deploy N8n as a microservice within AriesView's architecture to handle dynamic routing of documents, memos, and metadata. Use N8n's visual flow builder to model conditional logic (e.g., "If document type = Lease → send to AI Chatbot & Memo Engine").

2. Trigger-Based Routing Logic

N8n will listen to document upload events from Azure Blob Storage or the existing API. Based on metadata (file type, source, tenant, etc.), it dynamically routes the document to the appropriate module (e.g., chatbot, analytics, QA).

3. Integration & Deployment

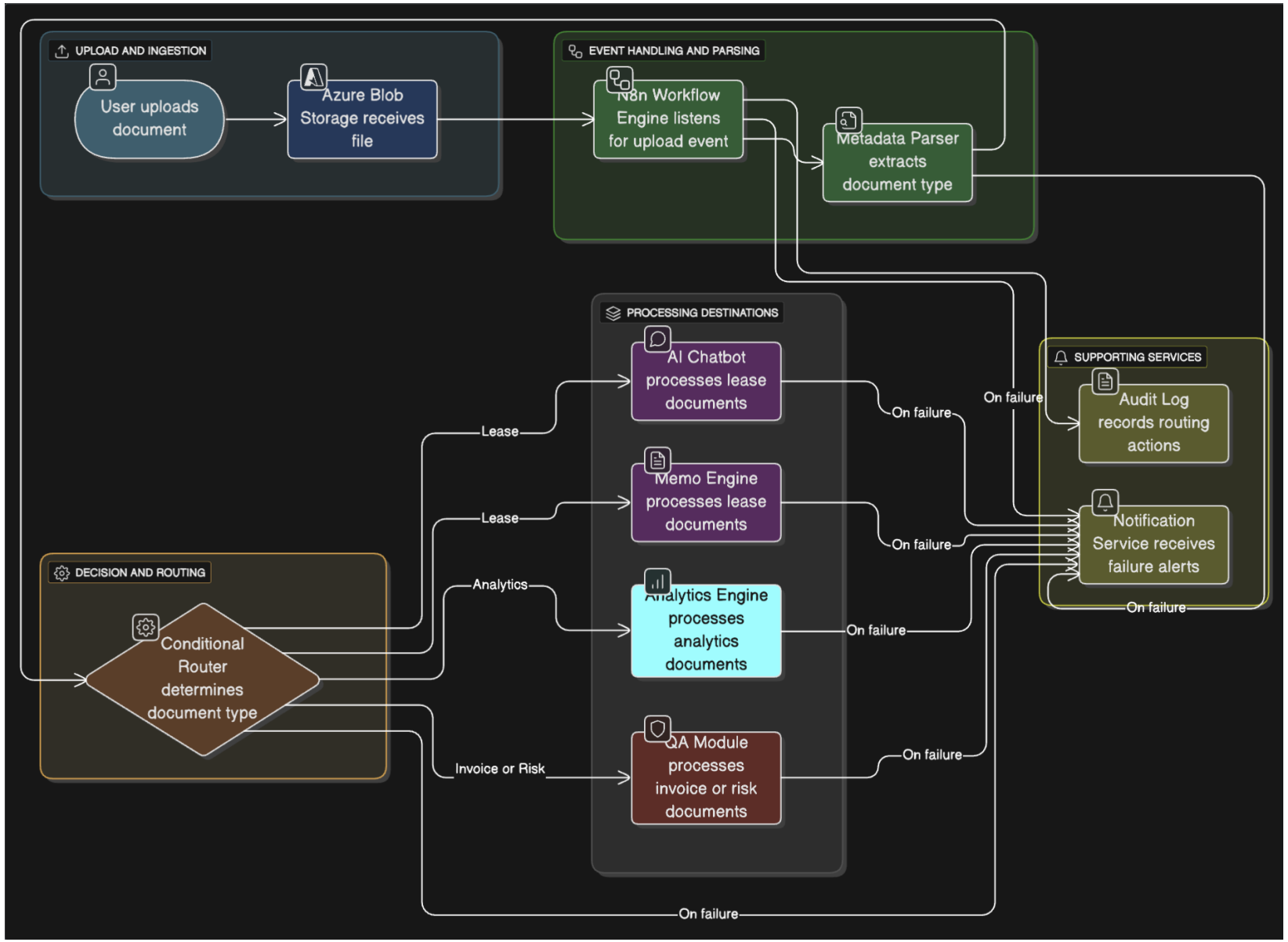
Run N8n in a Docker container (on the same VM or Kubernetes cluster as other microservices). Flows are created via GUI and version-controlled via Git. Integrate authentication using existing backend or OAuth2.

4. Tools & Dependencies

- N8n (self-hosted, Docker)
- Azure Blob Storage triggers
- Webhooks for memo engine, chatbot, analytics modules
- Git for flow versioning
- PostgreSQL or Redis (optional) for workflow state tracking

ARIESVIEW

Visual Design of the Architecture/Flow



ARIESVIEW

Visual Design of the Architecture/Flow

The diagram in the previous slide illustrates AriesView's **event-driven document routing** architecture using **N8n**.

Once a user uploads a document, it's first stored in **Azure Blob Storage**. From there:

1. The **N8n Workflow Engine** listens for new file events.
2. A **Metadata Parser** extracts document type (e.g., Lease, Analytics, Invoice).
3. A **Conditional Router** directs the file to the appropriate destination:
    - **Lease** → routed to **AI Chatbot** and **Memo Engine**
    - **Analytics** → routed to **Analytics Engine**
    - **Invoice or Risk Docs** → routed to **QA Module**
4. All processing happens asynchronously in specialized services.

If any routing or processing fails:

- The **Audit Log** captures the error.
- The **Notification Service** alerts the appropriate team or client.

This modular orchestration improves flexibility, error handling, and document-specific workflows — making AriesView's pipeline more intelligent, responsive, and maintainable.

ARIESVIEW

Recommendation 3:

SLA Monitor for Stuck or Delayed Document Pipelines

Overview:
Implement a monitoring layer that tracks the processing time of each document as it moves through AriesView's ingestion pipeline. If processing exceeds predefined SLAs (e.g., 10 minutes for leases, 5 minutes for invoices), the system automatically flags the item, logs the event, and alerts the team. This ensures issues are caught before they escalate into client-visible delays.

Benefit:

- Proactively detects *silent failures* and delays not captured by existing error handlers.
- Reduces missed deadlines for document processing.
- Improves operational reliability and client confidence in platform uptime and performance.

ARIESVIEW

Recommendation 3：Problem Statement / Gaps

Problem: No SLA-Based Monitoring for Document Processing

Lack of Time-Based Visibility: AriesView's current monitoring focuses on technical errors during document ingestion and processing, but it does not track whether a document has been processed within an acceptable timeframe. A document can remain in the pipeline without triggering an error, leaving delays undetected until a user reports them.

Operational and Client Impact: Without SLA-based tracking, processing bottlenecks may go unnoticed, causing critical documents — such as leases, risk reports, or financial statements — to be delivered late. This erodes client trust, reduces operational predictability, and can create compliance risks for time-sensitive workflows.

ARIESVIEW

Business Spec

SLA Monitoring for Pipeline Reliability — User-Focused Business Spec

Objective: Implement an automated SLA tracking system to monitor the time each document spends in the ingestion and processing pipeline, ensuring completion within predefined thresholds.

User Benefit:

Users gain confidence that documents will be processed promptly, and are proactively notified of any delays. Internal teams can respond quickly to bottlenecks, preventing escalation to clients.

Business Impact:

Minimizes missed deadlines, improves platform reliability, and strengthens client relationships through timely delivery. Reduces time spent manually investigating slow processes and enables data-driven performance reviews of the pipeline.

Strategic Fit:

Supports AriesView's mission to deliver fast, reliable, and transparent document intelligence. Aligns with operational excellence goals by providing real-time visibility into processing times and system health.

ARIESVIEW

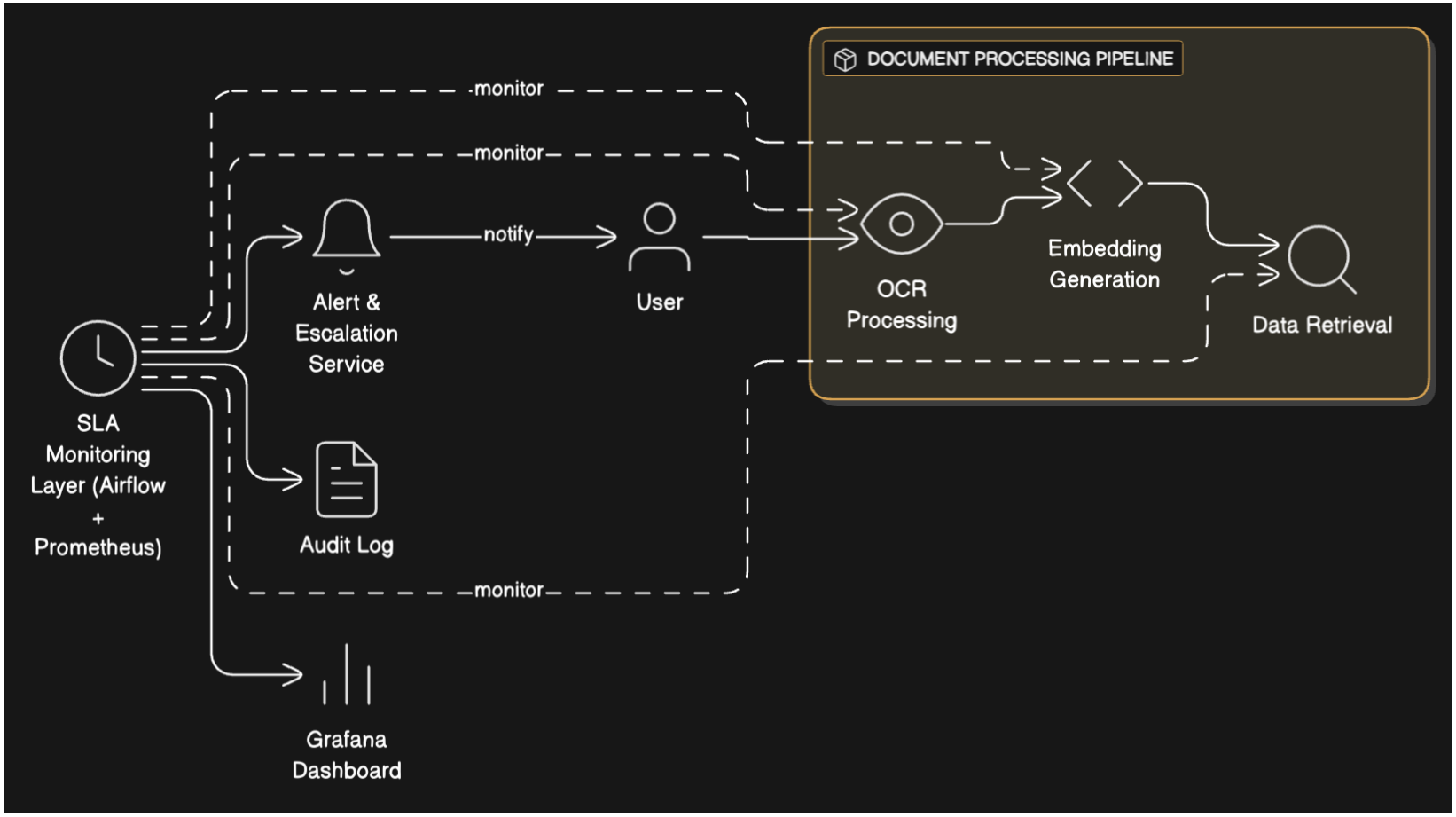
Tech Spec

Proposed Technology or Architectural Change:
Implement a Service Level Agreement (SLA) Monitoring Layer using **Apache Airflow** for scheduled health checks and **Prometheus + Grafana** for real-time pipeline latency metrics. Each pipeline stage will have defined SLA thresholds (e.g., OCR completion within 5 minutes, embedding within 2 minutes). Airflow DAGs will trigger escalation workflows if a stage exceeds its SLA, and alerts will be pushed via Microsoft Teams/Slack integration.

Impact on Existing Tech Stack:
No core replacements—integrates into AriesView's existing Azure + containerized architecture. Airflow will be deployed alongside current GitLab CI/CD workflows. Prometheus/Grafana stack will consume metrics from each microservice. Alerting integrates with the existing Notification Service module.

ARIESVIEW

Visual Design of the Architecture/Flow



ARIESVIEW

Visual Design of the Architecture/Flow

The diagram in the previous slide illustrates AriesView's **SLA Monitoring for Stuck or Delayed Document Pipelines** using *Airflow* and *Prometheus*.

1. **SLA Monitoring Layer** continuously tracks processing time for each stage in the *Document Processing Pipeline* (OCR, Embedding Generation, Data Retrieval).
2. If processing exceeds predefined SLAs, the **Alert & Escalation Service** notifies the user and logs the incident.
3. **Audit Log** captures all SLA violations for compliance, reporting, and process improvement.
4. **Grafana Dashboard** provides real-time visualization of SLA compliance, bottlenecks, and pipeline health.

If any SLA violation is detected:

- The **Alert & Escalation Service** informs the relevant team or client.
- The **Audit Log** records the breach for traceability and post-incident review.

This monitoring architecture ensures **real-time detection of delays**, **proactive intervention**, and **greater operational reliability**, ultimately improving client trust and platform performance.

ARIESVIEW

Recommendation 4:

Optimizing Postgres / Flask / Python for Scalability and Performance

Overview:
Enhance AriesView's database and API efficiency by introducing **PostgreSQL connection pooling** with pgBouncer, **async query execution** in Flask using SQLAlchemy 2.0 async ORM, and **database optimizations** such as table partitioning and materialized views for heavy analytic queries. Implement automated schema migrations using Alembic integrated into CI/CD to ensure consistency across environments.

Benefit:
Increases throughput and reduces latency for concurrent API requests.
Improves query performance for large datasets and analytics workloads.
Reduces downtime and deployment risks through automated, versioned schema changes.

ARIESVIEW

Recommendation 4：Problem Statement / Gaps

Problem: Inefficient Postgres / Flask / Python Stack for High-Concurrency and Analytics Workloads

Limited Connection Handling:
AriesView's current Flask + SQLAlchemy setup uses default database connections, which do not efficiently handle spikes in concurrent requests. Without connection pooling, APIs risk slowdowns or failures under high load.

Suboptimal Query Performance:
Large datasets and analytical queries run directly on primary tables, causing slow response times and increased CPU load. The absence of partitioning or caching layers results in repeated expensive computations.

Manual and Risk-Prone Schema Changes:
Database schema updates are performed manually or with minimal automation, creating potential for inconsistencies across environments and longer deployment cycles.

ARIESVIEW

Business Spec

Postgres Optimization & Flask/Python Performance Enhancements — User-Focused Business Spec

Objective:
Enhance AriesView's Postgres + Flask/Python architecture to improve API response times, handle high-concurrency workloads, and support faster analytics queries without sacrificing reliability.

User Benefit:
Users experience faster load times, reduced API timeouts, and smoother performance during peak usage. Analytical reports and dashboard views update more quickly, improving decision-making speed.

Business Impact:
Reduces downtime and lost productivity from slow query responses. Improves system scalability to support a growing user base and larger datasets. Minimizes infrastructure costs by optimizing query execution and resource usage.

Strategic Fit:
Aligns with AriesView's goal to deliver high-performance, scalable document intelligence. Supports both operational efficiency and analytics-driven growth by strengthening the backend foundation.

ARIESVIEW

Tech Spec

1. Optimize Postgres for High-Concurrency & Analytics

Implement query indexing, connection pooling (via **PgBouncer**), and caching for frequent queries. Use partitioned tables and materialized views to speed up analytics queries while reducing load on primary tables.

2. Enhance Flask/Python Performance

Introduce asynchronous request handling (e.g., **Gunicorn + gevent** or **uvicorn** with FastAPI-compatible blueprints for heavy workloads). Profile and refactor slow endpoints using tools like **cProfile** and **PyInstrument**.

3. Integration & Deployment

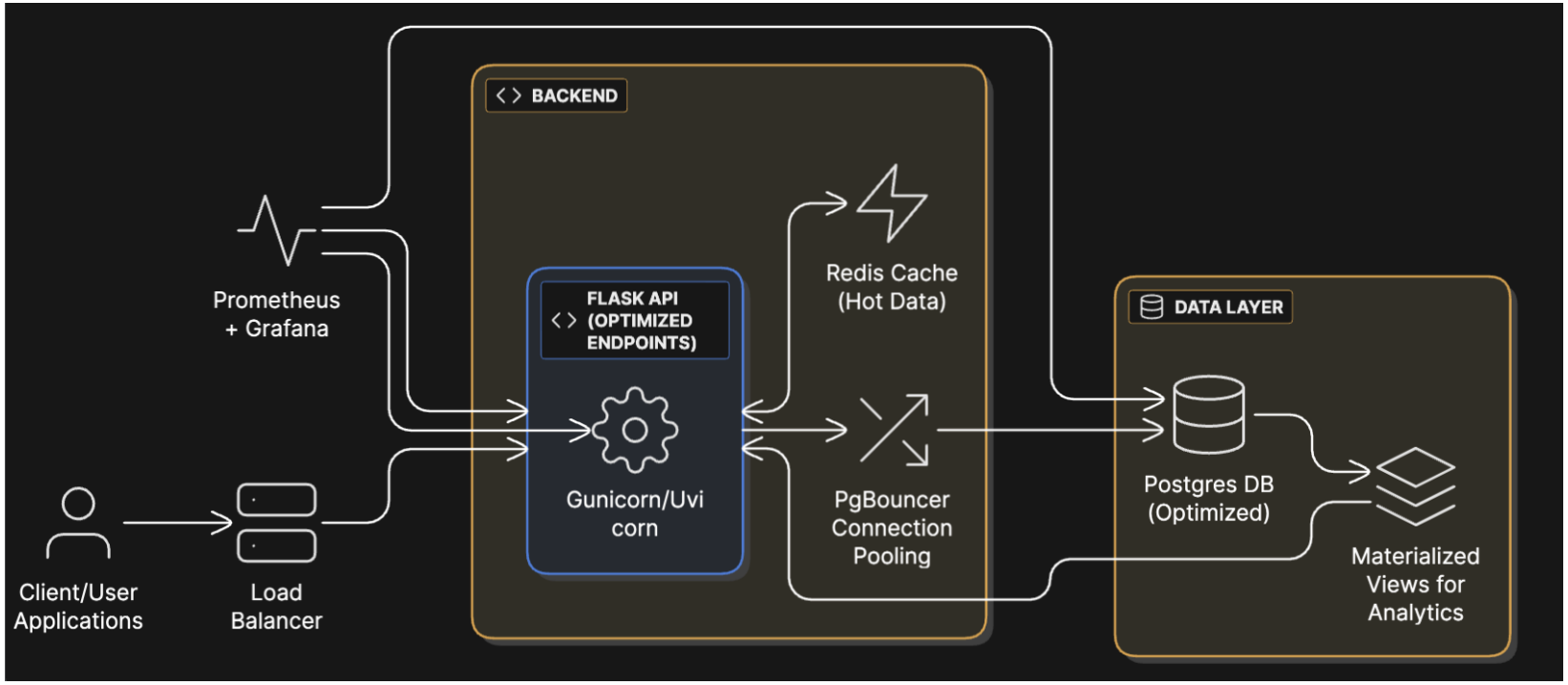
Apply optimizations in a staging environment first, with load testing using **Locust** or **k6**. Roll out changes in production with feature flags and monitor impact on API latency and query execution time.

4. Tools & Dependencies

- **PgBouncer** (Postgres connection pooling)
- **EXPLAIN ANALYZE** for query optimization
- **Materialized Views** for cached analytics queries
- **Redis** for hot data caching
- **Gunicorn/Uvicorn** for async request handling
- **Locust/k6** for load testing
- **Prometheus + Grafana** for performance monitoring

ARIESVIEW

Visual Design of the Architecture/Flow



ARIESVIEW

Visual Design of the Architecture/Flow

The diagram in the previous slide illustrates AriesView's **optimized Postgres + Flask/Python architecture** for high-performance data processing and analytics.

1. **Client/User Applications** send requests through a **Load Balancer**, distributing traffic evenly across backend instances.
2. The **Flask API** (optimized endpoints) runs on **Gunicorn/Uvicorn**, handling incoming requests efficiently.
3. **PgBouncer Connection Pooling** manages database connections to prevent bottlenecks and reduce latency.
4. **Redis Cache** stores frequently accessed "hot" data, reducing repeated Postgres queries.
5. The **Postgres Database** is tuned for query performance, with **materialized views** supporting complex analytics workloads.
6. **Prometheus + Grafana** continuously monitor system health and performance, alerting teams to anomalies.

This architecture reduces query times, improves concurrent request handling, and ensures stable performance under heavy load, aligning with AriesView's goals for scalability and reliability.

ARIESVIEW

Recommendation 5:

Automated Data Quality Validation Layer for Incoming Documents

Overview:
Implement a dedicated validation microservice that verifies the accuracy, completeness, and structure of extracted document data before it enters AriesView's main processing pipeline. Using a combination of AI-assisted anomaly detection and rule-based validation (e.g., regex, field presence checks, and business logic rules), the system flags or quarantines problematic records. This ensures only clean, reliable data moves forward, minimizing downstream errors.

Benefit:
Ensures high data integrity by catching errors and inconsistencies early in the pipeline.
Reduces reprocessing effort and manual data cleanup.
Improves analytics accuracy, chatbot reliability, and client confidence in system outputs.

Recommendation 5 : Problem Statement / Gaps

Problem: Lack of Pre-Processing Data Quality Checks

Inconsistent Data Integrity:

AriesView's current pipeline processes OCR and extracted data without a formalized validation step. Errors such as missing fields, malformed dates, or misclassified document types can pass through undetected, leading to inaccuracies in analytics and chatbot responses.

Downstream Impact:

Without early-stage quality validation, faulty data propagates through multiple services — causing misleading insights, failed client queries, and time-consuming manual corrections. This undermines operational efficiency and erodes trust in the platform's outputs.

ARIESVIEW

Business Spec

Objective:
Introduce a real-time validation layer that ensures all incoming document data meets predefined quality, accuracy, and completeness standards before entering the main AriesView pipeline.

User Benefit:
Clients receive analytics, chatbot responses, and financial models based on verified, trustworthy data. This improves decision-making confidence and reduces frustration from incorrect or incomplete outputs.

Business Impact:
- Cuts manual data-cleaning time by preventing bad data from entering the system.
- Reduces reprocessing costs and speeds up delivery timelines.
- Enhances client trust and retention by ensuring data reliability.

Strategic Fit:
Supports AriesView's mission to deliver accurate, high-value insights with speed and reliability.
Strengthens the company's competitive edge by offering a more dependable, "error-resistant" platform.

ARIESVIEW

Tech Spec

Proposed Technology or Architectural Change:
Add a Python/Flask microservice between OCR/Parsing and embedding stages to validate extracted data. Uses AI anomaly detection, regex checks, and business rules to flag or quarantine problematic records before they enter the main pipeline.

Impact on Existing Tech Stack:
Integrates seamlessly into the current Azure ingestion workflow. No replacements — adds a validation step to improve downstream accuracy while working with existing Blob Storage and PostgreSQL.

Implementation Approach:

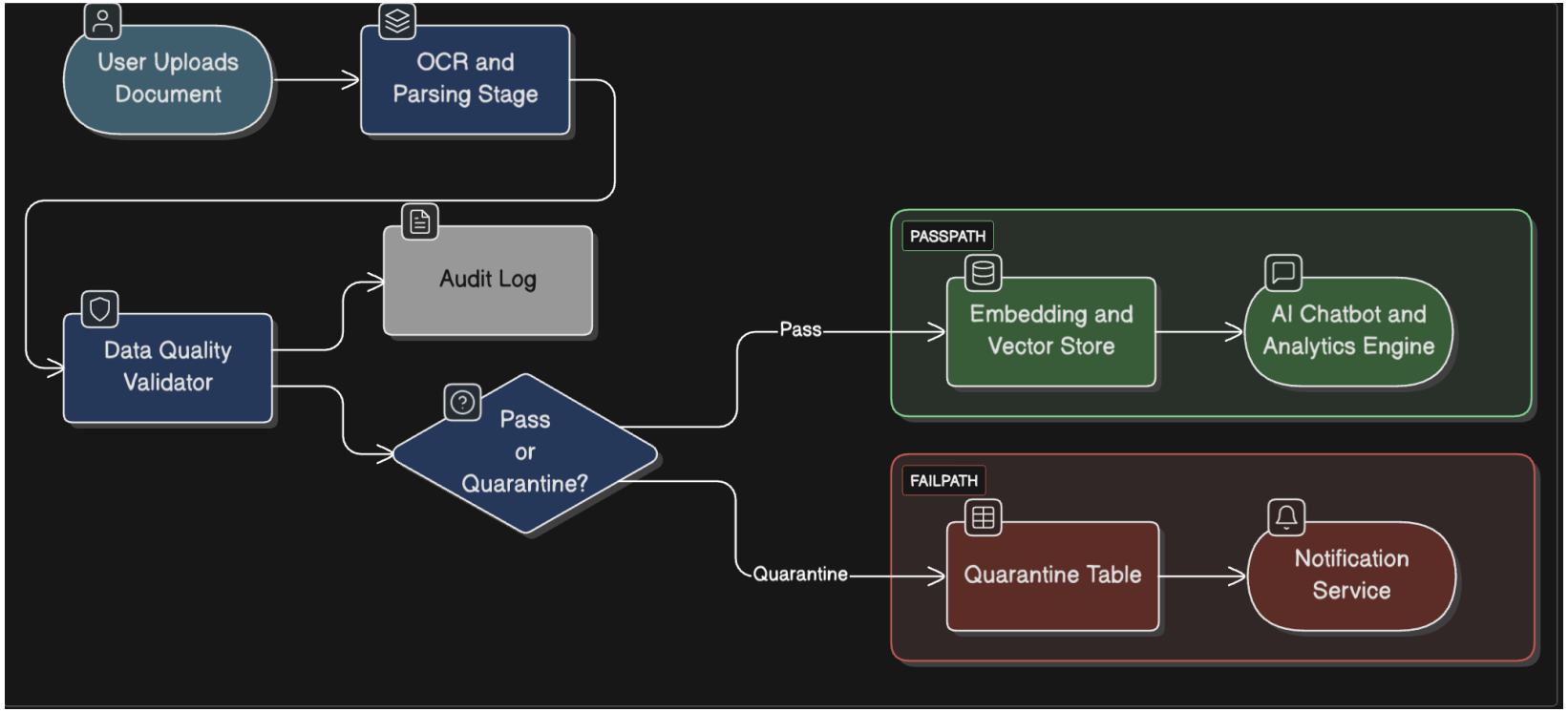
- Build REST API service for validation.
- Apply structural, rule-based, and statistical checks.
- Quarantine failed records in PostgreSQL with error logs.
- Deploy via Docker and GitLab CI/CD.

Dependencies:

- Python (Flask/FastAPI, pandas, regex)
- Azure Blob Storage, PostgreSQL
- Optional: scikit-learn for anomaly detection
- GitLab CI/CD, OpenTelemetry for monitoring

ARIESVIEW

Visual Design of the Architecture/Flow



ARIESVIEW

Visual Design of the Architecture/Flow

The diagram in the previous slide illustrates AriesView's **automated data quality validation pipeline** for incoming documents.

1. **User Uploads Document** → The file enters the **OCR and Parsing Stage** to extract raw text and structure.
2. The extracted data is sent to the **Data Quality Validator**, which performs:
    - AI-assisted anomaly detection
    - Regex field checks
    - Business logic validation
3. **Audit Log** records validation results for traceability.
4. **Pass or Quarantine Decision**:
    - **Pass Path** → Data is sent to **Embedding and Vector Store** and then to the **AI Chatbot & Analytics Engine** for downstream use.
    - **Fail Path** → Data is stored in a **Quarantine Table** and triggers the **Notification Service** to alert the team.
5. The entire process ensures that only validated, high-quality data flows through AriesView's primary processing systems.

Key Benefit:

By integrating this validation layer, AriesView prevents bad data from contaminating downstream analytics, reduces reprocessing costs, and enhances client confidence in the platform's accuracy.

ARIESVIEW

Recommendation 6: Automated Index Rebuild & Optimization Scheduler for Vector Store

Overview:
Implement an automated process that periodically rebuilds and optimizes Weaviate's vector indexes to maintain high retrieval accuracy and performance as data grows. The scheduler will detect index fragmentation, retrain embeddings for stale data, and execute rebuilds during off-peak hours. This ensures consistently fast, relevant search results without manual intervention.

Key Benefits:

- Maintains high retrieval precision and recall.
- Improves query latency and end-user experience.
- Reduces manual maintenance load for engineering.
- Supports scalability as document volume increases.

ARIESVIEW

Recommendation 6: Problem Statement / Gaps

Problem Statement / Gap Identified

AriesView's Weaviate vector store accumulates a growing volume of embeddings as more documents are processed. Without automated index health checks and optimization routines, the system gradually experiences:

- **Fragmented indexes** leading to slower query response times.
- **Stale embeddings** that reduce retrieval accuracy.
- **Manual rebuilds** that are resource-intensive and often delayed.

This lack of proactive maintenance creates operational risk, increases infrastructure costs, and impacts the relevance and speed of AI-driven search and chatbot responses.

ARIESVIEW

Business Spec

Objective:
Automate vector index rebuilds and optimizations to maintain peak search accuracy, retrieval speed, and AI chatbot responsiveness, even as document volume scales.

User Benefit:
Users experience consistently fast and accurate results from searches and chatbot queries, with minimal downtime during index updates. This enhances trust in AriesView's platform reliability and output quality.

Business Impact:
- Eliminates manual index maintenance, freeing up engineering resources.
- Prevents performance degradation and search inaccuracies as data grows.
- Supports seamless scaling without compromising system responsiveness.

Strategic Fit:
Aligns with AriesView's mission to deliver precise, timely insights by ensuring the underlying search infrastructure remains optimized. Strengthens the platform's competitive advantage in high-volume, AI-driven document intelligence.

ARIESVIEW

Tech Spec

1. Automated Vector Index Optimization Service

Deploy a scheduled background service that monitors index size, query latency, and data changes to trigger partial or full rebuilds of vector indexes (e.g., FAISS, Milvus, or Pinecone).

2. Zero-Downtime Index Swapping

Implement a dual-index strategy where new indexes are built in parallel and swapped atomically to avoid downtime for users during rebuilds.

3. Integration & Deployment

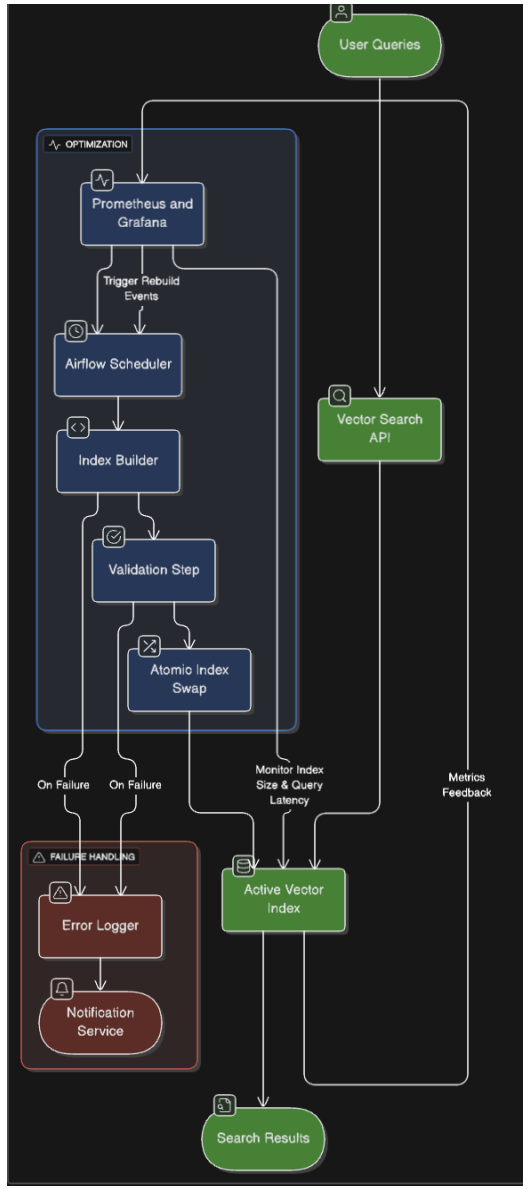
Run the optimization service as a containerized microservice within AriesView's architecture. Schedule rebuild triggers via Airflow or CronJobs, using metrics from Prometheus and Grafana for decision-making.

4. Tools & Dependencies

- Vector DB (FAISS, Milvus, or Pinecone)
- Airflow or CronJobs for scheduling
- Prometheus & Grafana for monitoring rebuild triggers
- Docker/Kubernetes for deployment
- Python scripts for optimization and index validation

ARIESVIEW

Visual Design of the Architecture/Flow



The diagram here illustrates **AriesView's Automated Vector Index Optimization Service**, designed to maintain peak search performance without downtime.

1. **User Queries** are processed by the **Vector Search API**, which retrieves results from the **Active Vector Index**.
2. **Prometheus and Grafana** continuously monitor index size and query latency.
3. If thresholds are exceeded, the **Airflow Scheduler** triggers the **Index Builder** to create a new, optimized index in parallel.
4. The **Validation Step** ensures accuracy, completeness, and data integrity of the new index.
5. Once validated, the **Atomic Index Swap** seamlessly replaces the old index with the new one without disrupting active queries.
6. **Failure Handling:** Any failure in the build or validation process is logged by the **Error Logger** and communicated via the **Notification Service**.

This modular optimization flow ensures:

- Continuous search availability with zero downtime.
- Improved query performance and accuracy.
- Real-time adaptability based on system performance metrics.

ARIESVIEW

Recommendation 7: Intelligent Document Classification & Routing with Adaptive Learning

Overview:
Implement an AI-powered classification layer that automatically identifies document type (e.g., Lease, Invoice, Risk Report, Analytics) immediately after OCR. The system uses a pre-trained NLP model fine-tuned on AriesView's historical data, and incorporates a feedback loop: when users correct misclassifications, the model learns and improves over time. This adaptive approach ensures document routing accuracy grows continuously without manual rule updates.

Benefit:

- Improves routing accuracy and reduces processing delays caused by misclassification.
- Reduces manual sorting effort and rule maintenance for engineering teams.
- Continuously adapts to new document formats, improving system resilience and scalability.

ARIESVIEW

Recommendation 7: Problem Statement / Gaps

Problem: Manual & Rule-Based Document Classification Limits Scalability

Static Rules Are Prone to Errors:

AriesView's current routing relies on predefined rules and metadata tags. This approach struggles with documents that deviate from expected formats or contain ambiguous layouts, resulting in misclassifications.

No Continuous Learning:

When classification errors occur, corrections made by users are not fed back into the system. This prevents the model from improving over time, forcing engineering teams to manually update rules.

Operational Impact:

Misclassified documents can be routed to incorrect processing modules, leading to delays, reprocessing, and potential compliance risks for time-sensitive documents. This slows delivery timelines and erodes client trust in platform automation.

ARIESVIEW

Business Spec

Objective:
Enhance document routing accuracy and reduce manual intervention by replacing static classification rules with an AI-driven, self-improving classification engine.

User Benefit:
Users receive faster, more accurate processing of leases, contracts, invoices, and other documents, reducing delays caused by misroutes and minimizing the need for manual corrections.

Business Impact:

- Cuts manual sorting time by up to 80%, freeing staff for higher-value tasks.
- Reduces downstream processing delays, improving overall SLA adherence.
- Enhances client satisfaction and trust through consistently accurate automation.

Strategic Fit:
Supports AriesView's mission to streamline CRE workflows and increase operational efficiency. Adaptive learning aligns with the platform's AI-first strategy, ensuring long-term scalability and resilience against evolving document formats.

ARIESVIEW

Tech Spec

Proposed Technology or Architectural Change:
Integrate a document classification microservice leveraging a fine-tuned NLP model (e.g., LayoutLM, BERT, or Azure Document Intelligence). Feed OCR output directly into the model to determine document type, then route to the appropriate processing pipeline. Implement a feedback API to capture user corrections and retrain the model periodically.

Impact on Existing Tech Stack:

- Works downstream of existing OCR (PaddleOCR) without replacing it.
- Integrates with AriesView's routing logic in N8n or Flask API.
- Requires minimal changes to storage structure — classification results stored alongside metadata in Postgres.

Implementation Approach:

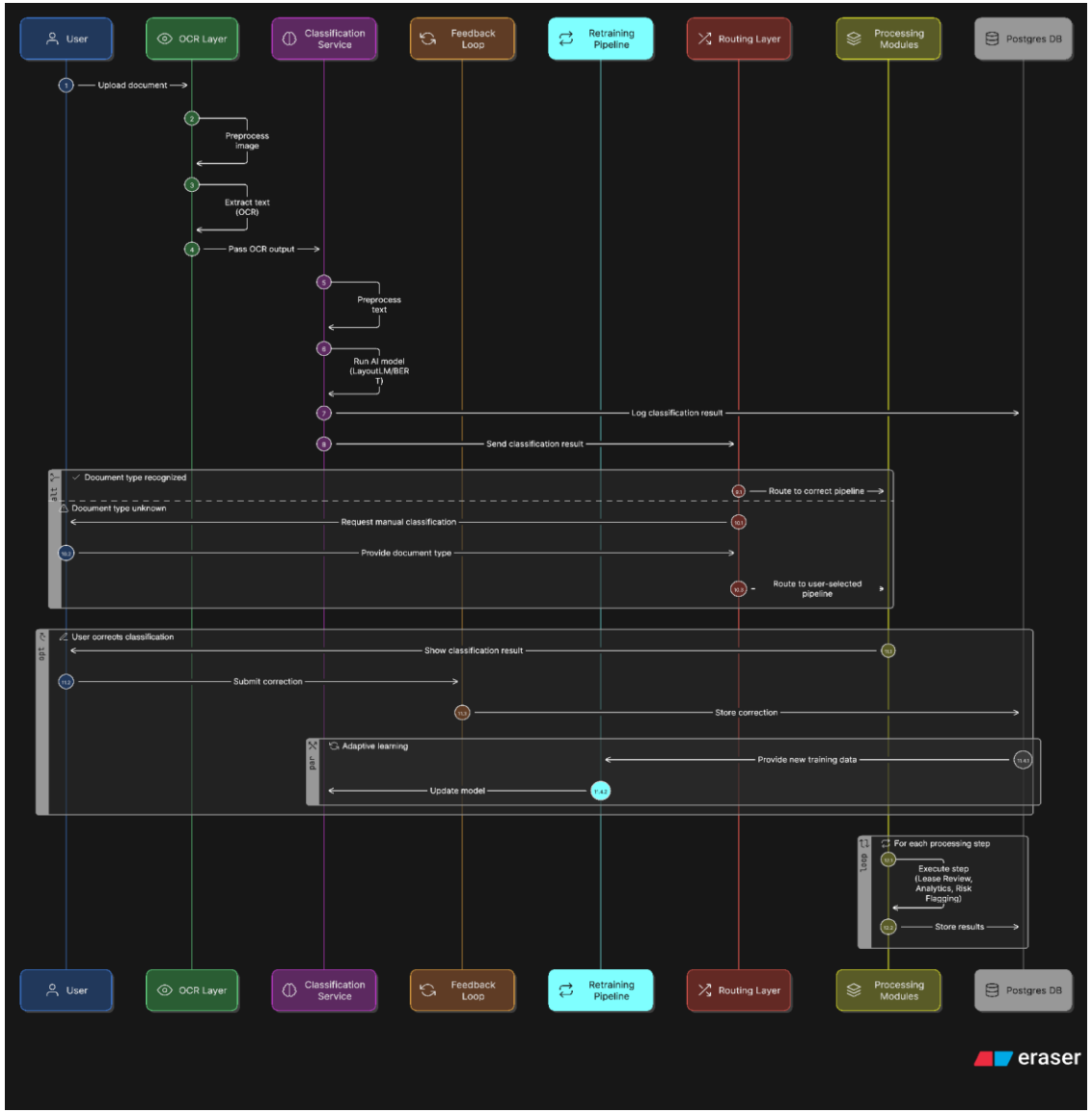
1. Fine-tune pre-trained NLP model with historical AriesView document set.
2. Build Flask or FastAPI service to handle classification requests.
3. Add feedback capture endpoints for user corrections.
4. Schedule periodic retraining using Airflow pipelines.
5. Deploy service in Docker container, integrated into CI/CD pipeline.

Dependencies and Required Tools/Integrations:

- Python (Transformers, PyTorch/TensorFlow, scikit-learn)
- Flask/FastAPI for API service
- N8n for workflow routing integration
- Airflow for retraining scheduling
- Docker & GitLab CI/CD for deployment
- Postgres for storing classification results & feedback logs

ARIESVIEW

Visual Design of the Architecture/Flow



1. **OCR Layer** – Preprocesses and extracts text from uploaded documents.
2. **Classification Service** – Uses ML (e.g., LayoutLM/BERT) to identify document type.
3. **Routing Layer** – Directs documents to the correct processing pipeline.
4. **Feedback Loop** – Captures user corrections for misclassifications.
5. **Retraining Pipeline** – Updates the ML model with feedback for continuous improvement.
6. **Processing Modules** – Executes domain-specific tasks and stores results in **Postgres DB**.
7. **If confidence is low** → Requests manual classification → Uses correction for retraining.

This architecture improves classification accuracy, reduces manual effort, and adapts over time.

ARIESVIEW

Recommendation 8: Context-Aware Search & Query Expansion Engine

Core Idea:
Enhance AriesView's document Q&A by introducing an **AI-powered query expansion system** that automatically rewrites or augments user queries to retrieve more relevant context from the vector database.

Why it's unique:

- Instead of relying solely on exact keyword or embedding matches, the system expands queries with **synonyms, domain-specific jargon, and related clauses**.
- Uses a **real estate-specific semantic ontology** (e.g., "CAM Charges" → "Common Area Maintenance Costs") to improve retrieval precision.
- Integrates with current Weaviate vector DB without major re-architecture.

Example:
A user asks: "What's the lease end date?" → Query expansion adds variations like "termination date," "expiration," "lease expiry," then runs retrieval.

Benefits:

- Reduces missed answers caused by wording differences.
- Improves chatbot accuracy without retraining the LLM.
- Speeds up user decisions by surfacing more complete context.

ARIESVIEW

Recommendation 8: Problem Statement / Gap Identified

AriesView's current RAG pipeline relies on **direct semantic and keyword matches** for retrieving relevant chunks from the vector database. While effective for exact phrasing, it can miss important context when users phrase questions differently from how terms appear in documents.
For example, a lease document may use "termination date" while a user asks "lease end date," resulting in incomplete or missed results.
This gap leads to:

- **Reduced chatbot accuracy** when retrieving clauses or dates expressed in varied language.
- **Inconsistent user experience** depending on the wording of queries.
- **Missed insights** that could affect underwriting, risk assessment, and compliance decisions.

Without a query expansion layer, AriesView's AI cannot fully leverage its document store's depth, reducing the platform's ability to deliver complete, context-rich answers.

ARIESVIEW

Business Spec

Business Specification

Implementing a context-aware search and query expansion layer will significantly improve **accuracy, completeness, and reliability** of AriesView's AI-driven Q&A. By automatically expanding queries with synonyms, related legal and financial terms, and domain-specific jargon, users will receive more **comprehensive results** without needing to rephrase questions.

Business Impact:

- **Higher decision confidence** for underwriting, due diligence, and asset management.
- **Reduced time-to-answer** by minimizing follow-up queries.
- **Lower operational risk** from missed clauses or incorrect interpretations.
- **Enhanced client satisfaction** through more consistent and accurate outputs.

Alignment with AriesView's Business Goals:

- Supports AriesView's mission to deliver **faster, smarter deal execution** with fewer errors.
- Increases differentiation from competitors by combining **real estate expertise** with advanced AI retrieval capabilities.
- Strengthens trust in AI-driven insights, encouraging greater adoption across client teams.

ARIESVIEW

Tech Spec

Proposed Technology or Architectural Change:

Integrate an AI-driven **query expansion module** between the chatbot interface and the retrieval layer. This module uses domain-trained language models and a curated real estate term ontology to rewrite and augment incoming queries with synonyms, related terms, and alternative phrasings before sending them to Weaviate.

Impact on Existing Tech Stack:

- No major architectural disruption.
- Module sits between the Node.js API and Weaviate vector DB.
- Minimal latency impact if implemented with lightweight LLM or embedding-based synonym lookup.

Implementation Approach:

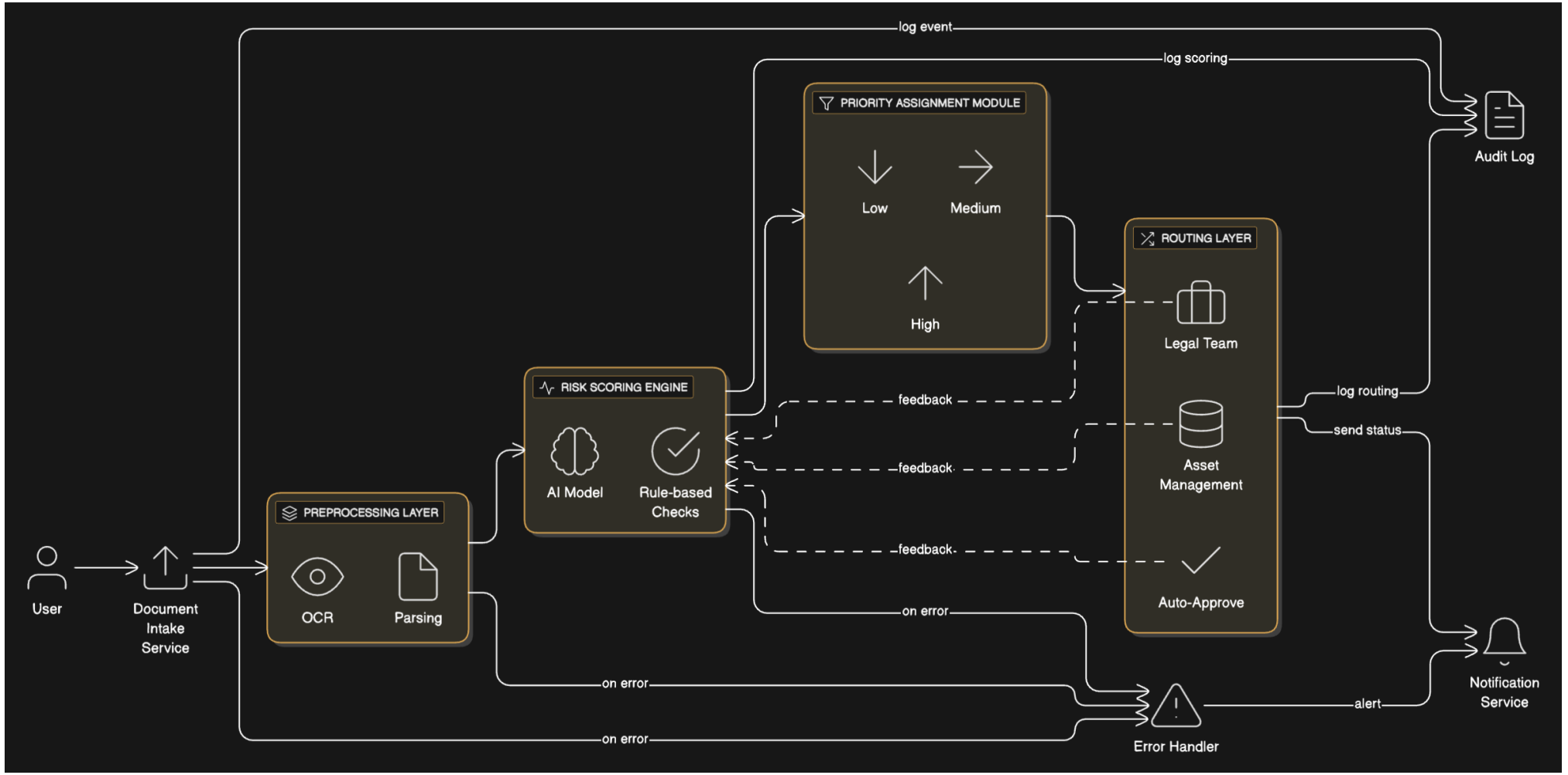
1. Maintain a **real estate domain ontology** (e.g., “lease end date” ↔ “termination date,” “CAM charges” ↔ “common area maintenance”).
2. On query submission, run expansion logic to create multiple semantic variants.
3. Submit expanded queries to Weaviate for hybrid retrieval (semantic + keyword).
4. Merge and rank results based on confidence scores.
5. Pass ranked results to the LLM for answer generation.

Dependencies and Required Tools/Integrations:

- Python/Node.js service for expansion logic.
- Domain-specific thesaurus or ontology database.
- Weaviate hybrid search API.
- Optionally, SentenceTransformers for semantic similarity scoring.

ARIESVIEW

Visual Design of the Architecture/Flow



ARIESVIEW

Visual Design of the Architecture/Flow

The diagram above illustrates AriesView's **Automated Document Risk Scoring & Prioritization System**. Once a user uploads a document:

1. The **Document Intake Service** receives the file and sends it to the **Preprocessing Layer** for OCR and text parsing.
2. The **Risk Scoring Engine** evaluates the document using an AI model combined with rule-based checks to generate a risk score.
3. The **Priority Assignment Module** classifies the document into Low, Medium, or High priority.
4. The **Routing Layer** directs:
    - **High Priority** → Legal Team
    - **Medium Priority** → Asset Management
    - **Low Priority** → Auto-Approve
5. **Audit Logs** record all scoring and routing events for compliance.
6. A **Feedback Loop** from each routing outcome improves the AI model over time.

If any errors occur:

- The **Error Handler** captures the issue.
- The **Notification Service** alerts the appropriate team.

This architecture ensures **proactive risk mitigation, faster decision-making, and continuous model improvement** for document workflows.

ARIESVIEW

Recommendation 9: Adaptive Infrastructure Auto-Scaler with Cost Optimization

Overview:
Implement a predictive auto-scaling system that dynamically adjusts compute, database, and caching resources based on real-time usage and forecasted demand. The system integrates with Kubernetes/AWS/GCP auto-scaling to preemptively scale up during peak traffic and scale down during idle periods, while a cost-governor minimizes unnecessary cloud spend.

Benefit:

- Maintains optimal performance and low latency during traffic spikes.
- Reduces cloud infrastructure costs by eliminating idle or underutilized resources.
- Improves platform reliability, ensuring consistent service availability during unpredictable load changes.

ARIESVIEW

Recommendation 9: Problem Statement / Gaps

Problem – Static Infrastructure Scaling

AriesView's current infrastructure scaling is reactive, with manual intervention or basic thresholds triggering resource changes. This can result in over-provisioning during low demand (wasting cloud spend) or under-provisioning during sudden spikes (causing latency or downtime).

Operational & Cost Impact:

- **Performance Risk:** Delayed scaling during traffic surges leads to slower response times and potential service outages.
- **Inefficiency:** Idle resources remain active during low-traffic periods, inflating operational costs.

Predictability Gap: No forecasting model exists to anticipate usage patterns and proactively optimize capacity.

ARIESVIEW

Business Spec

Efficiency & Effectiveness Gains

A predictive auto-scaling system ensures AriesView's platform maintains peak performance under varying workloads while eliminating waste from idle resources. Proactive scaling based on usage patterns prevents slowdowns during client onboarding, AI processing, or portfolio data updates.

Business Impact

- **Cost Savings:** Reduces unnecessary cloud expenditure by scaling down unused compute and database resources during off-peak hours.
- **Scalability:** Seamlessly supports user growth, high-volume imports, or seasonal spikes without downtime.
- **Client Value:** Guarantees a consistent, fast user experience regardless of load conditions, improving trust and satisfaction.

Alignment with AriesView's Goals

This initiative supports AriesView's roadmap for **performance reliability, operational efficiency, and cost optimization**—key factors in delivering premium AI-powered real estate solutions at competitive margins.

ARIESVIEW

Tech Spec

Proposed Technology / Architectural Change

Integrate a predictive auto-scaling layer into AriesView's cloud infrastructure using Kubernetes Horizontal/Vertical Pod Autoscaler (HPA/VPA) or equivalent on Azure Kubernetes Service (AKS). Leverage historical workload metrics from Prometheus/Grafana to train a lightweight forecasting model (e.g., Prophet, ARIMA) that anticipates load changes and triggers proactive scaling.

Impact on Existing Tech Stack

- Works within AriesView's existing Azure-based deployment.
- Enhances CI/CD pipelines to deploy scaling rules alongside application updates.
- Adds cost-governor scripts to manage maximum scaling thresholds.

Implementation Approach

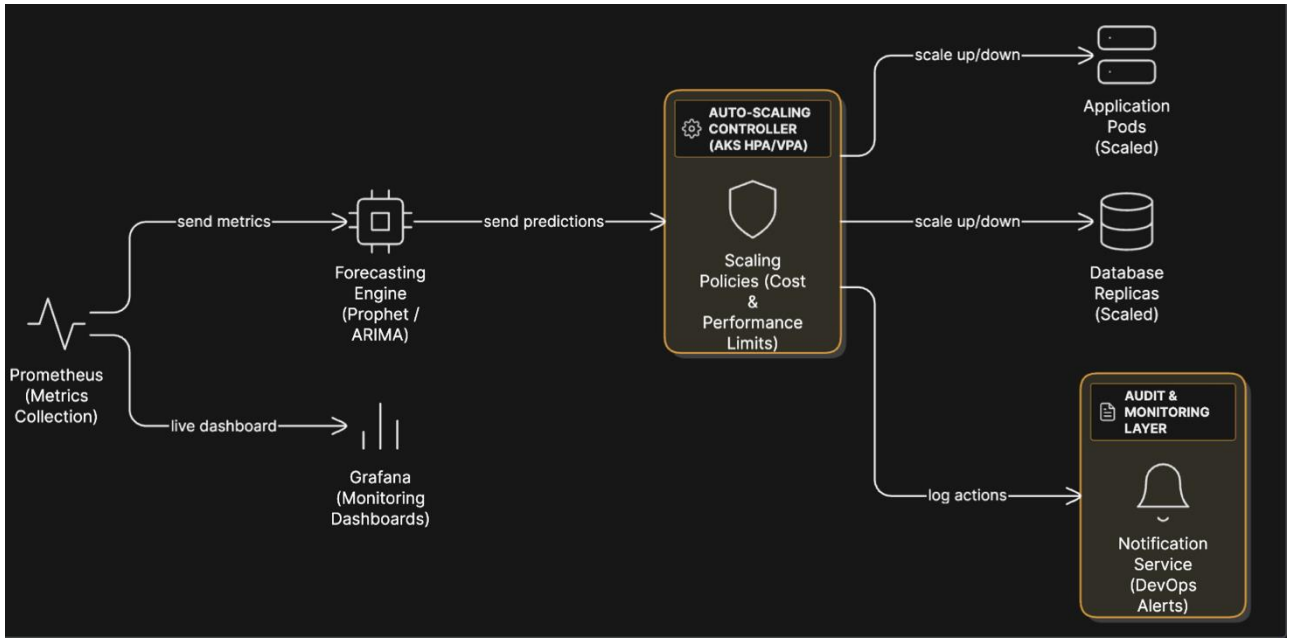
1. **Data Collection:** Gather CPU, memory, and request rate metrics via Prometheus.
2. **Forecasting Engine:** Train a demand prediction model on historical usage.
3. **Auto-Scaling Rules:** Implement proactive scaling triggers in AKS or Docker Swarm.
4. **Cost Control:** Apply budget-aware scaling policies.
5. **Testing & Deployment:** Deploy incrementally in staging before production rollout.

Dependencies / Required Tools

- Azure Kubernetes Service (AKS) or Docker Swarm
- Prometheus + Grafana for metrics
- Python libraries (Prophet, Statsmodels) for forecasting
- Azure Cost Management APIs
- GitLab CI/CD integration

ARIESVIEW

Visual Design of the Architecture/Flow



The diagram illustrates AriesView's **predictive auto-scaling** system that dynamically adjusts compute and database resources based on forecasted demand.

1. **Prometheus (Metrics Collection)** gathers real-time performance data (CPU, memory, query latency).
2. **Forecasting Engine** (using Prophet/ARIMA) analyzes trends and predicts upcoming workload surges or drops.
3. Predictions are sent to the **Auto-Scaling Controller** (AKS HPA/VPA) which applies **scaling policies** (cost caps, performance thresholds).
4. The controller scales **Application Pods** and **Database Replicas** up or down accordingly.
5. **Grafana** provides live dashboards for monitoring.
6. All scaling actions are logged in the **Audit & Monitoring Layer**, with alerts sent via **Notification Service** to DevOps.

This predictive scaling approach ensures **optimal performance**, **cost efficiency**, and **system reliability** during fluctuating workloads.

ARIESVIEW

Recommendation 10: Automated Compliance & Security Drift Detection Layer

Overview:
Implement a continuous monitoring layer that detects configuration drift, security vulnerabilities, and compliance gaps across AriesView's infrastructure and applications. Using Open Policy Agent (OPA) and custom compliance rule sets, the system validates infrastructure and application configurations against SOC 2, GDPR, and client-specific requirements. Unauthorized changes trigger real-time alerts and, where safe, automated remediation actions.

Benefit:

- Maintains continuous compliance without manual audit cycles.
- Detects and mitigates potential security breaches before exploitation.
- Reduces downtime and reputational risk through proactive remediation.

ARIESVIEW

Recommendation 10: Problem & Gaps – Compliance & Security Drift in AriesView Infrastructure

Content:

- **Manual Audits** – Current compliance checks rely on periodic manual reviews, leaving long windows where drift or vulnerabilities can go undetected.
- **Configuration Drift Risk** – Unmonitored infrastructure changes in Azure, PostgreSQL, or backend services may break SOC 2/GDPR alignment.
- **No Real-Time Alerts** – Security or compliance deviations are discovered late, increasing remediation cost and risk exposure.
- **Limited Automation** – No system in place to auto-validate configurations against predefined policy baselines.

Impact:

- Increases downtime and security risk.
- Slower incident response times.
- Potential reputational and financial damage if compliance breaches occur undetected.

ARIESVIEW

Business Spec

This enhancement will ensure AriesView maintains continuous compliance with SOC 2, GDPR, and client security requirements by automatically detecting and alerting on infrastructure or configuration drift. By integrating automated checks into the CI/CD pipeline and runtime monitoring, the platform will significantly reduce risk exposure and compliance audit effort.

Business Impact:

- **Risk Mitigation:** Immediate alerts on non-compliant changes prevent costly breaches.
- **Operational Efficiency:** Reduces manual audit workload by up to 70%.
- **Client Confidence:** Demonstrates a proactive security posture, enhancing trust.
- **Regulatory Alignment:** Ensures ongoing readiness for audits without last-minute scrambles.

Alignment with AriesView Goals:

Supports AriesView's commitment to **enterprise-grade security**, **data governance**, and **client trust**, while improving operational scalability and resilience.

ARIESVIEW

Tech Spec

Proposed Technology or Architectural Change:
Integrate an automated compliance monitoring system that runs configuration and security drift checks both during CI/CD deployments and continuously in production. Use tools like **Open Policy Agent (OPA)** for policy enforcement, **Terraform Cloud/Driftctl** for IaC drift detection, and **AWS Config or Azure Policy** for runtime compliance.

Impact on Existing Tech Stack:
Minimal disruption: Integrates into existing **GitLab CI/CD** pipelines and **Azure** environment.
Adds a dedicated compliance microservice that works in parallel with current deployment workflows.

Implementation Approach:

- Define compliance & security policies in OPA or Rego.
- Integrate IaC drift detection into GitLab CI/CD for pre-deployment checks.
- Enable continuous drift monitoring via Azure Policy / AWS Config (for multi-cloud scenarios).
- Configure alerting through Slack/MS Teams and incident logging.
- Store compliance logs in a secure audit repository for reporting.

Dependencies & Required Tools/Integrations:

- **Open Policy Agent (OPA)** or **Conftest**
- **Terraform Cloud** or **Driftctl**
- **Azure Policy** / **AWS Config**
- **GitLab CI/CD** pipelines
- **Slack/MS Teams API** for notifications

ARIESVIEW

Summary of Recommendations

1. Hybrid OCR for Structured Field Extraction – Combines PaddleOCR with regex & layout-aware parsing for higher accuracy in extracting key fields from complex documents.
2. Automated Document Intake & Routing (N8n) – Uses N8n to standardize and automate routing workflows, reducing engineering effort and errors.
3. SLA Monitor for Processing Delays – Tracks pipeline stage durations; alerts when SLAs are breached to prevent silent failures.
4. Postgres / Flask / Python Performance Optimization – Adds connection pooling, async queries, and table partitioning for scalable, low-latency APIs and analytics.
5. Automated Data Quality Validation Layer – Validates and quarantines bad data pre-ingestion using AI anomaly detection and rule-based checks.
6. Automated Vector Index Optimization – Periodically rebuilds/optimizes Weaviate indexes to maintain fast, accurate search without downtime.
7. Intelligent Document Classification with Adaptive Learning – AI-driven classification learns from corrections to improve routing accuracy over time.
8. Context-Aware Search & Query Expansion – Expands queries with synonyms and domain terms to improve chatbot & search accuracy.
9. Adaptive Infrastructure Auto-Scaler with Cost Optimization – Predictively scales resources based on demand while minimizing cloud costs.
10. Automated Compliance & Security Drift Detection – Continuously monitors for security/configuration drift and enforces compliance policies in real time.