1. **Users/Clients**: Interact with the system to submit documents for processing.
2. **API Gateway**:
   * Routes incoming requests to the appropriate microservices.
   * Queries the **Service Registry** to discover available instances of microservices and route requests accordingly.
3. **Load Balancer**:
   * Distributes traffic across multiple instances of the API Gateway for high availability and fault tolerance.
4. **Message Queue**:
   * Decouples document ingestion from processing.
   * Ensures reliable message delivery.
   * Tools: RabbitMQ, Apache Kafka, AWS SQS.
5. **Conversion Microservices**:
   * Handle specific document types (e.g., PDF, DOCX).
   * Scalable and can be replicated for high availability.
   * **Circuit Breakers**: Protect microservices from cascading failures by detecting failures and halting requests when a service is in a failure state.
   * **Bulkhead**: isolate different types of processing tasks (e.g., document conversion vs. metadata handling).
6. **Result Storage & Retrieval**:
   * Stores processed documents and metadata.
   * Uses SQL databases for structured data and object storage (e.g., AWS S3) for large files.
7. **Caching Layer**:
   * Caches frequently accessed data to improve performance.
   * Tools: Redis, Memcached.
8. **Monitoring & Logging**:
   * Aggregates logs and monitors system performance.
   * Tools: ELK Stack (Elasticsearch, Logstash, Kibana), Prometheus with Grafana.

**Handling Various Scenarios**

1. **Sudden Spikes in Conversion Requests**
   * **Horizontal Scaling**: Auto-scaling groups for conversion microservices and API Gateway ensure that additional instances are provisioned automatically.
   * **Load Balancer**: Distributes the load across available instances to prevent any single instance from becoming overwhelmed.
   * **Rate Limiting**: Applied at the API Gateway level to prevent abuse and ensure fair use of resources.
2. **Large Documents That Take a Long Time to Process**
   * **Asynchronous Processing**: Long-running tasks are handled asynchronously. Document processing is queued, and users can check the status through a separate endpoint.
   * **Chunking**: For explosible documents, consider chunking the document into smaller pieces if possible, and process them in parallel.
   * **Timeout Handling**: Implement timeouts and retries in the processing microservices to handle and recover from processing delays.
3. **System Failures During Conversion Processes**
   * **Retry Mechanism**: Configure retry policies in the message queue to handle transient failures. Failed messages can be retried a set number of times before being moved to a dead-letter queue for manual intervention.
   * **Circuit Breaker**: Protects services from being overwhelmed by failures and allows them to recover gracefully.
   * **Failover Strategy**: Deploy microservices across multiple availability zones or regions to handle instance or zone failures.

**Security Considerations**

1. **Data Encryption**
   * **In Transit**: Use HTTPS for secure communication between clients and the API Gateway, and between services.
   * **At Rest**: Encrypt sensitive data stored in databases and object storage using appropriate encryption mechanisms.
2. **Authentication and Authorization**
   * **API Gateway**: Implement OAuth 2.0 or API keys for secure access control.
   * **Service-to-Service Authentication**: Use mutual TLS or signed tokens for communication between microservices.
3. **Vulnerability Management**
   * **Regular Updates**: Keep all components, including libraries and frameworks, up to date with security patches.
   * **Security Scanning**: Regularly scan code and dependencies for vulnerabilities.

**AI-Powered Document Classification and Extraction**

* **AI Model**: Develop a machine learning model to automatically classify documents into different types and extract key information (e.g., text from scanned images using OCR, data fields from forms).
* **Integration**: Integrate this AI model into the document ingestion process to pre-process documents and determine the appropriate conversion microservice.
* **Benefits**:
  + **Improved Accuracy**: AI can enhance the accuracy of document classification and data extraction, reducing manual effort.
  + **Dynamic Adaptation**: AI can adapt to new document formats and types over time with continuous learning.