EcoAgent: Apparel Sustainability AI Agent

Task 1:

The workflow is designed to evaluate the sustainability impact of consumer apparel products available on Amazon. An automated agent dynamically collects, analyzes, and synthesizes information from three distinct sources:

1. **Textile Material Sustainability Database:** An internal database providing scores for the intrinsic environmental and social impacts of raw materials based on Textile Exchange data.
2. **Company ESG Reports & Web Data:** Publicly available information assessing the brand's stated sustainability commitments, reported metrics, and public perception, retrieved via web searches.
3. **Amazon Consumer Reviews:** Real-world consumer feedback scraped from the product page to identify perceptions related to product durability, quality, and sustainability aspects.

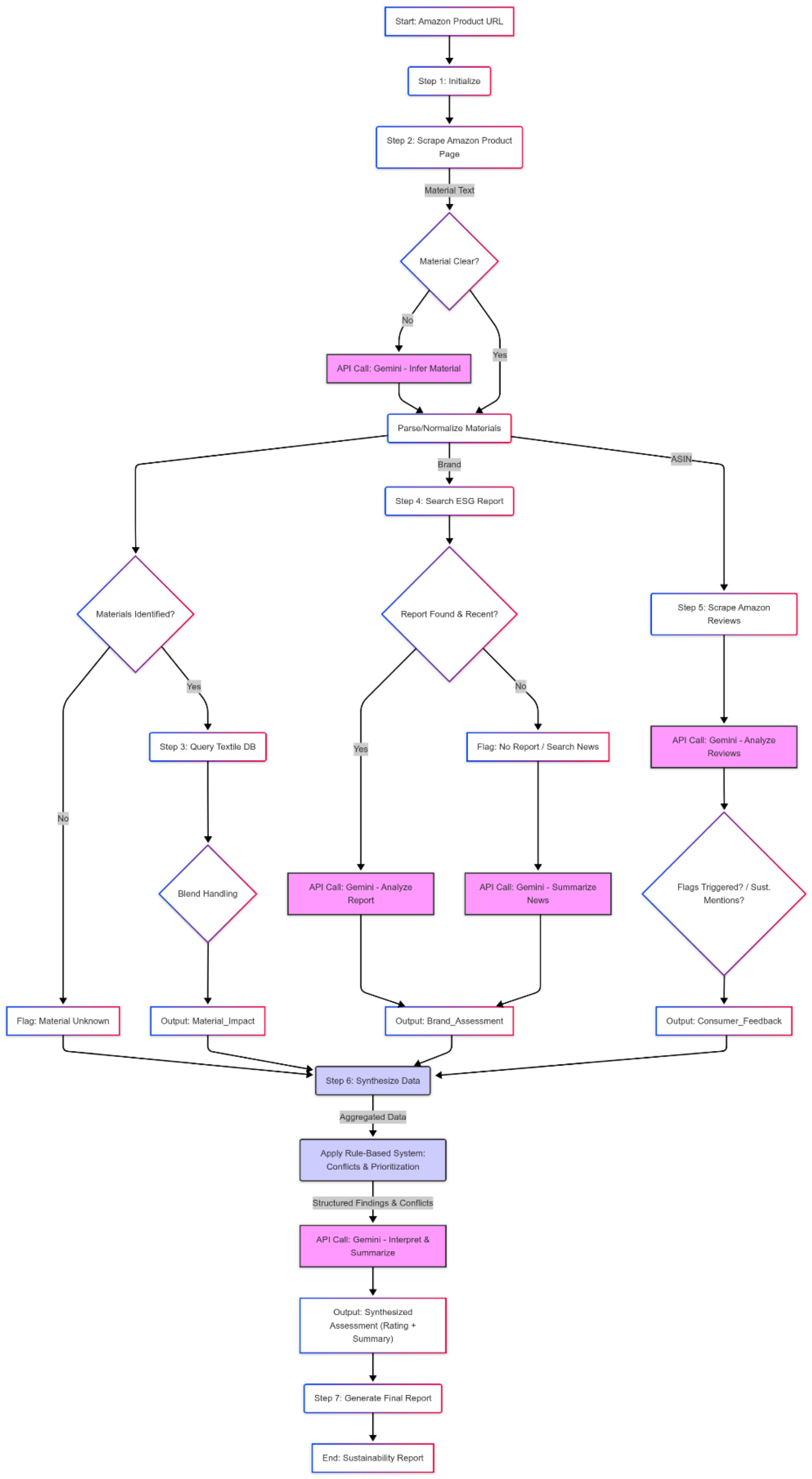
The agent executes a sequence of steps involving data fetching (web scraping, API calls), data processing, rule-based decision-making, and LLM-powered synthesis (using Google Gemini) to generate a final sustainability assessment report for a given Amazon product URL.

2. Data Sources Detailed

- **Textile Exchange Material Database (https://textileexchange.org/pfmm/):**
    - **Content:** This database contains structured sustainability impact data derived from sources like Textile Exchange. It covers 5 primary material types relevant to apparel: **Cotton, Synthetic, MMCF (Man-Made Cellulosic Fibres), Flax, and Wool.**
    - **Attributes:** For each material type (and potentially specific sub-types), the database provides scores or ratings across key dimensions: **Climate, Water, Chemistry, Land Use, Biodiversity, Resource Use & Wastage, Human Rights, and Initiative Integrity.**
    - **Role:** This database serves as the **primary source for objective, science-based assessment of the inherent impact** of the materials identified in the product. Its scores form the baseline for the material impact component of the final assessment.
- **Company ESG Reports & Web Data (Public):**
    - **Content:** Publicly available information such as official Sustainability/ESG reports (PDFs, webpages), corporate website sections detailing initiatives, press releases, and relevant news articles concerning the brand or its parent company.
    - **Retrieval:** The agent uses web search mechanisms (a Search API) targeting official company sources and reputable news outlets.
    - **Role:** This source assesses the **brand's stated commitment, transparency, and accountability** regarding sustainability.
- **Amazon Consumer Reviews (Public):**
    - **Content:** Textual feedback from customers on the specific Amazon product page.
    - **Retrieval:** The agent scrapes recent reviews from the product page.

- **Role:** Provides **real-world insights into product performance (especially durability), quality, and direct user perceptions** related to sustainability, acting as a validation layer.

3. Step-by-Step Workflow Explanation & Justification


Start: Amazon Product URL
Step 1: Initialize
Step 2: Scrape Amazon Product Page
Material Text
Material Clear?
No
Yes
API Call: Gemini - Infer Material
Parse/Normalize Materials
Brand
ASIN
Step 4: Search ESG Report
Materials Identified?
Report Found & Recent?
Step 5: Scrape Amazon Reviews
Yes
No
Step 3: Query Textile DB
Yes
No
Flag: No Report / Search News
API Call: Gemini - Analyze Reviews
Blend Handling
API Call: Gemini - Analyze Report
API Call: Gemini - Summarize News
Flags Triggered? / Sust. Mentions?
Flag: Material Unknown
Output: Material_Impact
Output: Brand_Assessment
Output: Consumer_Feedback
Step 6: Synthesize Data
Aggregated Data
Apply Rule-Based System: Conflicts & Prioritization
Structured Findings & Conflicts
API Call: Gemini - Interpret & Summarize
Output: Synthesized Assessment (Rating + Summary)
Step 7: Generate Final Report
End: Sustainability Report

1. **Start: Amazon Product URL:** Input node; the workflow begins with the product's URL.

2. **Step 1: Initialize:** Agent prepares internal data structures for storing fetched and processed information. *Justification: Standard setup step.*
3. **Step 2: Scrape Amazon Product Page:** Agent accesses the URL, parses HTML, and extracts Product Title, Brand Name, ASIN, and Material Composition Text. *Justification: Collects essential product identifiers and raw material info.*
4. **Decision: Material Clear?** Checks if Material Composition Text was found and is parsable. *Justification: Determines if direct material analysis is possible.*
   - **If NO:** Branch to **API Call: Gemini - Infer Material.** Sends product title/description to Gemini API to predict likely materials. *Justification: Attempts to overcome missing/poor data on Amazon via LLM inference.*
   - **If YES:** Branch directly to Parse/Normalize Materials.
5. **Step: Parse/Normalize Materials:** Processes the material text (scraped or inferred) into a structured list (e.g., [{'material': 'Cotton', 'percentage': 100}]). Normalizes names for DB matching. Extracts Brand and ASIN for other paths. *Justification: Standardizes data for reliable processing and querying.*
6. **Parallel Path Split:** The workflow branches to process Material, Brand, and Consumer data concurrently.
   - **Path 1: Material Impact Assessment**
     - **Decision: Materials Identified?** Checks if the parsing step yielded any known materials. *Justification: Avoids futile DB queries.*
     - **If NO: Flag: Material Unknown.** Records inability to perform material analysis. *Justification: Handles data gap gracefully.* Connects directly to Synthesis step.
     - **If YES: Step 3: Query Textile DB.** Uses normalized material names to query the internal database for sustainability scores across the 8 attributes. *Justification: Retrieves core material impact data.*
     - **Decision: Blend Handling:** Applies logic if multiple materials exist (weighted average, equal average, etc.). *Justification: Calculates representative scores for the product's mix.*
     - **Output: Material_Impact:** Stores the resulting material sustainability scores. Connects to Synthesis step.
   - **Path 2: Brand Sustainability Assessment**
     - **Input:** Brand name.
     - **Step 4: Search ESG Report:** Agent searches the web for the brand's (or parent company's) latest ESG/Sustainability report. *Justification: Seeks official brand commitments.*
     - **Decision: Report Found & Recent?:** Checks for report existence and age (see Section 4 for threshold). *Justification: Assesses data availability and relevance.*

        - **If YES: API Call: Gemini - Analyze Report.** Sends report text to Gemini to extract structured commitments, metrics, etc. *Justification: Efficiently extracts key data from lengthy reports.*
        - **If NO: Flag: No Report / Search News.** Records lack of report. -> **API Call: Gemini - Summarize News.** Agent searches for/summarizes relevant news on brand's sustainability/controversies. *Justification: Gathers alternative public data when official reports are missing.*
        - **Output: Brand_Assessment:** Stores findings (report analysis or news summary) and transparency status. Connects to Synthesis step.
    - **Path 3: Consumer Perception Analysis**
        - **Input:** ASIN.
        - **Step 5: Scrape Amazon Reviews:** Agent scrapes recent reviews. *Justification: Collects user feedback.*
        - **API Call: Gemini - Analyze Reviews:** Sends reviews to Gemini for sentiment analysis, theme extraction (Durability, Quality, Sustainability, Ethics), and flagging critical issues (see Section 4 for thresholds). *Justification: Structures insights from unstructured reviews.*
        - **Decision: Flags Triggered? / Sust. Mentions?:** Checks Gemini output for critical flags or significant sustainability discussion. *Justification: Identifies high-priority feedback.*
        - **Output: Consumer_Feedback:** Stores review analysis results. Connects to Synthesis step.
7. **Step 6: Synthesize Data:** Aggregates outputs from the three paths (Material_Impact, Brand_Assessment, Consumer_Feedback, and Flag: Material Unknown if applicable). *Justification: Central point for combining diverse data types.*
    - **Action: Apply Rule-Based System: Conflicts & Prioritization.** Executes predefined rules to weight data sources, handle conflicts (see Section 5), apply penalties for flags, and calculate preliminary scores/assessments. *Justification: Implements core logic and weighting.*
    - **Output:** Structured Findings & Conflicts.
    - **Action: API Call: Gemini - Interpret & Summarize.** Feeds key structured findings and conflicts to Gemini for nuanced interpretation, overall rating suggestion, and summary generation. *Justification: Leverages LLM for holistic understanding and reporting.*
    - **Output: Synthesized Assessment (Rating + Summary):** The final assessment data.
8. **Step 7: Generate Final Report:** Formats the synthesized assessment, including rating, summary, supporting details, and conflict notes, into a user-readable output. *Justification: Presents results clearly.*
9. **End: Sustainability Report:** Final output node of the workflow.

4. Specify exact conditions or thresholds under which the agent would trigger additional API calls or deeper analysis.

The workflow triggers additional API calls or deeper analysis under these specific conditions:

1. **Material Inference (Gemini):**
    - **Condition:** The Material Clear? decision node (Step 4) evaluates to **NO**, meaning the initial scraping (Step 2) failed to find clear, parsable material composition text on the Amazon page.
    - **Triggered Action:** An API call is made to **Gemini (API Call: Gemini - Infer Material)** to attempt inference based on product title and description text.
2. **Brand News Search & Summary (Gemini):**
    - **Condition:** The Report Found & Recent? decision node (Step 4, Path 2) evaluates to **NO**. This occurs if:
        - No official ESG/Sustainability report is found via web search for the brand/parent company.
        - OR a report is found, but it is older than a predefined recency threshold (e.g., **published more than 3 years ago**).
    - **Triggered Action:** The agent flags the lack of a recent report and initiates a "deeper analysis" consisting of:
        - Performing targeted web searches for recent news articles related to the brand's sustainability practices, initiatives, or controversies.
        - Making an API call to **Gemini (API Call: Gemini - Summarize News)** to summarize the findings from these news results.
3. **Consumer Review Analysis (Gemini - *Note: This API call runs by default but performs deeper analysis based on thresholds*):**
    - **Initial Trigger:** The API Call: Gemini - Analyze Reviews (Step 5, Path 3) is always triggered after scraping reviews.
    - **Thresholds for *Internal* Deeper Flagging:** Within this API call's logic/prompt, specific analysis is triggered based on content:
        - **Durability Flag:** If the percentage of reviews mentioning keywords strongly indicative of poor durability (e.g., "fell apart," "wore out quickly," "broke," "didn't last") exceeds a threshold (e.g., **> 15%** of analyzed reviews), the potential_durability_issue flag is set to True.
        - **Chemical Flag:** If the number or percentage of reviews mentioning keywords related to chemical smells or skin irritation (e.g., "chemical smell," "strong odor," "rash," "irritated skin") exceeds a threshold (e.g., **> 3 mentions** or **> 5%** of reviews), the potential_chemical_issue flag is set to True.
        - **Sustainability Mention Focus:** If the percentage of reviews containing sustainability-related keywords (positive or negative) is below a threshold

(e.g., **< 5%**), this is noted, potentially reducing the weight of consumer sustainability sentiment in the final synthesis.

---

5. Explain how the agent manages conflicting or ambiguous data from different sources.

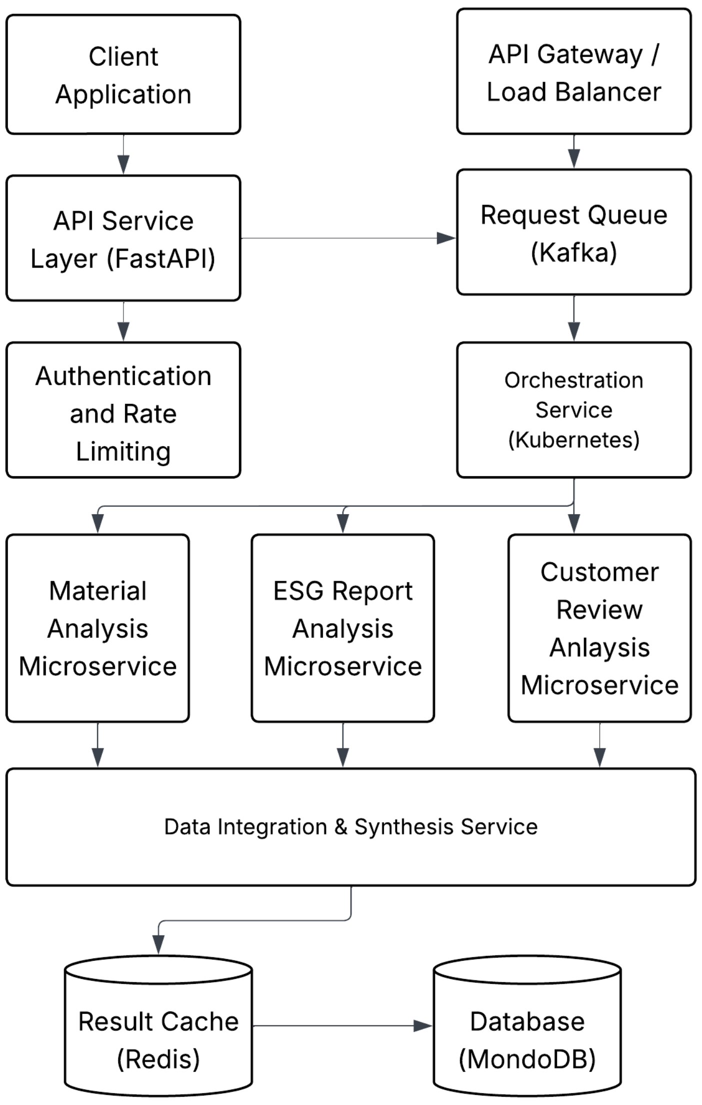
The agent employs a multi-faceted approach during the **Synthesis (Step 6)** phase, primarily using the **Rule-Based System** combined with **LLM Interpretation**:

1. **Prioritization Rules:** The core conflict management strategy relies on a predefined hierarchy of data source credibility and impact:
   - **Highest Priority:** Negative flags from consumer reviews (potential_durability_issue, potential_chemical_issue). These indicate real-world product failures or safety risks and will heavily downgrade the assessment, overriding positive brand claims or material scores. *Example: A t-shirt made of "organic cotton" (good material score) with many reviews saying it disintegrated after one wash gets a poor overall rating.*
   - **High Priority:** Objective material impact scores (Textile Exchange DB). These form the baseline environmental/social footprint. *Example: A product made primarily of conventional polyester will have a high climate impact score, even if the brand makes general sustainability claims.*
   - **Medium Priority:** Brand transparency (report availability) and verified performance/controversies (from report analysis or news). Lack of transparency or known issues negatively impacts the score. Specific, verified positive metrics can mitigate negative material scores somewhat. *Example: A brand using conventional cotton (high water score) but providing verified data on significant water reduction in that specific product line's manufacturing might see a less severe penalty than a brand making vague claims.*
   - **Lower Priority:** General consumer sentiment (if no major flags) and unverified/vague brand claims. These provide context but don't override stronger signals.
2. **Conflict Flagging & Reporting:** The Rule-Based system identifies specific conflict patterns:
   - *Greenwashing Pattern:* High-impact material score + Strong positive but vague brand claims -> Flagged in assessment_notes and reflected negatively in the summary.
   - *Accountability Gap:* Good material score + Low brand transparency (no report/controversies) -> Flagged in assessment_notes.
   - The final report explicitly mentions key conflicts found.
3. **Handling Ambiguity:**
   - *Material Ambiguity:* If Flag: Material Unknown is set, the Material Impact score is absent or estimated with low confidence. The final report explicitly states this limitation ("Material composition unclear, limiting assessment confidence").

- *Vague Brand Claims:* The Gemini analysis prompt for ESG reports specifically looks for vague language vs. quantitative metrics. Vague claims are weighted less by the rules and potentially noted by the LLM summary.
- *Mixed Signals:* If data points in different directions without triggering major conflict rules (e.g., decent material, okay brand report, mixed reviews on fit/style), the LLM synthesis step (API Call: Gemini - Interpret & Summarize) is designed to capture this nuance and reflect a "Mixed" or balanced assessment rather than forcing a strongly positive or negative outcome.

4. **LLM Interpretation:** The final Gemini call in Step 6 receives the structured data, including flags and conflict notes from the rule system. It interprets this combined information to generate a nuanced summary that attempts to reconcile or explain the different facets of the product's sustainability profile based on the provided evidence and applied rules.

Task 3: Orchestration Plan - System Architecture


Client Application
API Gateway / Load Balancer
API Service Layer (FastAPI)
Request Queue (Kafka)
Authentication and Rate Limiting
Orchestration Service (Kubernetes)
Material Analysis Microservice
ESG Report Analysis Microservice
Customer Review Anlaysis Microservice
Data Integration & Synthesis Service
Result Cache (Redis)
Database (MondoDB)

Overview

This plan details how to deploy the EcoAgent workflow at scale to handle 1000+ concurrent product assessments while ensuring real-time processing, reliability, and fault tolerance. The architecture follows a microservices approach using cloud-native technologies to maximize scalability and resilience.

Cloud Infrastructure Components

Compute Layer

- **Primary Platform**: AWS EKS (Elastic Kubernetes Service)
- Provides container orchestration for all microservices
- Enables declarative scaling and self-healing capabilities

Data Processing & Storage

- **Message Queue**: Apache Kafka
- Handles high-throughput event streaming
- Decouples system components for failure isolation
- Retains messages for replay in case of downstream failures
- **Primary Database**: MongoDB Atlas
- Stores assessment results, product information, and material data
- Scales horizontally through sharding for high write throughput
- Provides flexible document schema for varying data structures
- **Caching Layer**: Amazon ElastiCache for Redis
- Stores frequent assessment results and preprocessed data
- Reduces database load and improves response times

Networking & Security

- **API Gateway**:
- Handles authentication, authorization, and rate limiting
- Provides unified entry point to microservices
- Implements request throttling based on client tiers
- **Load Balancing**:
- Distributes traffic across service instances
- Performs health checks and removes unhealthy instances

Microservices Architecture

1. API Service Layer

- **Implementation**: FastAPI for high-performance API endpoints
- **Purpose**:
- Handles direct client requests and returns responses
- Validates input data and enforces schema compliance
- Publishes assessment requests to Kafka
- **Scaling Strategy**:
- Horizontal auto-scaling based on request rate and CPU utilization

2. Authentication and Rate Limiting

- **Implementation**: Custom service with Redis-backed rate limiting
- **Purpose**:
- Authenticates API requests
- Enforces rate limits per client/API key
- Protects backend services from traffic spikes
- **Scaling Strategy**:
- Stateless design for horizontal scaling
- Cache authentication results in Redis
- Distributed rate limiting using Redis counters

3. Material Analysis Microservice

- **Implementation**: Python service with cached database lookups
- **Purpose**:
- Analyzes material information from product descriptions
- Queries textile sustainability database
- Computes impact scores for identified materials
- **Scaling Strategy**:
- Queue-based workload distribution via Kafka
- Predictive auto-scaling based on queue depth

4. ESG Report Analysis Microservice

- **Implementation**: Python service with LLM processing capabilities
- **Purpose**:
- Searches for and analyses company ESG reports
- Extracts sustainability metrics and claims

- Generates standardized assessments of company commitments
- **Scaling Strategy**:
- Horizontal scaling based on queue depth
- Partitioning by company to enable parallel processing
- Caching of ESG results with 24-hour TTL

5. Customer Review Analysis Microservice

- **Implementation**: Python service with sentiment analysis capabilities
- **Purpose**:
- Processes consumer reviews from multiple sources
- Identifies sustainability-related comments
- Generates sentiment metrics related to sustainability perceptions
- **Scaling Strategy**:
- Horizontal scaling based on review volume
- Parallel processing of reviews for same product
- Batch processing for efficiency

6. Data Integration & Synthesis Service

- **Implementation**: Python service with weighted scoring algorithms
- **Purpose**:
- Combines data from all analysis services
- Resolves conflicts between different data sources
- Produces final sustainability assessment
- **Scaling Strategy**:
- Event-driven scaling based on completed analyses
- In-memory processing for speed
- Optimized algorithms for large data sets

Real-time Processing Strategies

Event-driven Architecture

- All components communicate asynchronously via Kafka
- Services process events as they arrive in real-time

Parallel Processing

- Analysis tasks run concurrently across microservices

- Independent analyses (material, ESG, reviews) execute in parallel
- Final integration happens when all data is available

Streaming Analytics

- Continuous processing of incoming assessment requests
- Real-time dashboards showing system throughput

High Throughput Handling (1000+ Concurrent Assessments)

Workload Distribution

- **Kafka Partitioning**:
- Each analysis type uses dedicated Kafka topics with multiple partitions
- Partitioning by product category enables parallel processing
- Consumer groups distribute work across service instances
- **Work Stealing**:
- Services can process messages from multiple partitions
- Idle workers can take tasks from busy queues
- Ensures optimal resource utilization

Queue-based Load Leveling

- Incoming requests are queued in Kafka
- Prioritization based on client tier and request type
- Backpressure mechanisms prevent system overload

Database Optimization

- **Read/Write Separation**:
- Read replicas for high-read operations
- Primary instances for write operations
- Read-your-writes consistency where needed
- **Indexing Strategy**:
- Indexes on frequently queried fields
- Composite indexes for common query patterns
- Regular index maintenance

Connection Pooling

- Database connection pools sized for peak load
- Redis connection multiplexing

- Kafka producer/consumer pools pre-warmed

Reliability and Fault Tolerance

Resilience Patterns

- **Circuit Breakers**:
- Prevent cascading failures when dependencies fail
- Automatic fallback to degraded operation modes
- Gradual recovery with exponential backoff
- **Bulkhead Pattern**:
- Isolated resource pools for critical components
- Failure containment within service boundaries
- Resource limits prevent total system failure
- **Retry Pattern**:
- Automatic retries for transient failures
- Exponential backoff to prevent flooding
- Dead letter queues for failed messages

Graceful Degradation

- **Feature Flags**:
- Dynamic enabling/disabling of non-critical features
- Configurable service quality levels
- Ability to run in "essential services only" mode
- **Fallback Strategies**:
- Cached results when live analysis is unavailable
- Default values for missing data points
- Simplified scoring when full analysis impossible

Latency Optimization

Asynchronous Task Management

- **Non-blocking I/O**:
- Asynchronous code patterns in all services
- Event loops for handling concurrent requests
- Connection pooling to external services
- **Background Processing**:

- Long-running tasks execute asynchronously
- Status webhooks for client notification
- Polling endpoints for progress tracking

Multi-level Caching Strategy

- **Application Caching**:
- In-memory caches for frequent lookups
- Distributed Redis cache for shared data
- Material database fully cached in memory
- **Result Caching**:
- Similar product assessments cached with appropriate TTL
- Brand ESG analyses cached for 24 hours
- Periodic background refresh of popular items
- **Cache Policies**:
- LRU eviction for memory caches
- Time-based expiration for data with freshness requirements
- Cache warming for predictable high-traffic patterns

Query Optimization

- **Materialized Views**:
- Pre-computed results for common queries
- Scheduled updates for data freshness
- Denormalized data structures for read performance
- **Request Batching**:
- Combine multiple material lookups in single database query
- Batch processing of reviews
- Aggregated updates to result cache

Autoscaling Strategy

Reactive Scaling

- **Metrics-based Scaling**:
- CPU utilization threshold (70%)
- Memory usage threshold (80%)
- Queue depth proportional scaling

- **Traffic-based Scaling**:
- Request rate thresholds
- Concurrent connection counts
- Active user sessions

Predictive Scaling

- **Time-based Scaling**:
- Historical traffic patterns analysis
- Pre-scheduled scaling for known peak times
- Gradual scale-down after peaks
- **ML-driven Scaling**:
- Predictive models based on historic patterns
- Anomaly detection for unexpected traffic
- Feedback loop for scaling effectiveness

Monitoring and Observability

Metrics Collection

- **Prometheus & Grafana**:
- System-level metrics (CPU, memory, network)
- Business metrics (assessments/second, response time)
- Custom metrics (cache hit rates, queue depths)
- **Dashboards**:
- Real-time system health
- Service-level SLO/SLI tracking
- Business KPI monitoring

This orchestration plan provides a comprehensive framework for deploying the EcoAgent workflow at scale. By leveraging container orchestration, event-driven architecture, and cloud services, the system can handle 1000+ concurrent product assessments while maintaining real-time performance and fault tolerance. The multi-layered approach to caching, auto-scaling, and resilience ensures the system can adapt to varying workloads and gracefully handle component failures.

Task 4: Monitoring and Evaluation Strategy

1. Critical KPIs for Agent Performance and Decision Accuracy

1.1 Assessment Throughput

- **Definition**: Number of complete product sustainability assessments processed per minute
- **Target**: ≥ 20 assessments/minute (≥ 1200/hour) under normal load
- **Warning Threshold**: < 15 assessments/minute for 5 consecutive minutes
- **Critical Threshold**: < 10 assessments/minute for 10 consecutive minutes
- **Measurement Method**: Count of completed assessments with final scores in the results database

1.2 Assessment Accuracy

- **Definition**: Agreement rate between agent assessments and human expert validation
- **Target**: ≥ 90% agreement on overall sustainability rating (±0.5 points on 10-point scale)
- **Warning Threshold**: < 85% agreement over weekly validation batch
- **Critical Threshold**: < 80% agreement over weekly validation batch
- **Measurement Method**: Random sampling of 100 assessments weekly for expert review

1.3 End-to-End Latency

- **Definition**: Time from assessment request submission to complete report generation
- **Target**: ≤ 30 seconds for standard assessment depth
- **Warning Threshold**: > 45 seconds for 90th percentile of requests
- **Critical Threshold**: > 60 seconds for 90th percentile of requests
- **Measurement Method**: Request timestamps captured at API gateway and final report delivery

1.4 Decision Path Consistency

- **Definition**: Consistency of agent decision paths for similar product inputs
- **Target**: ≥ 95% consistency in workflow paths for products with similar attributes
- **Warning Threshold**: < 90% consistency in weekly analysis
- **Critical Threshold**: < 85% consistency in weekly analysis
- **Measurement Method**: Analysis of decision logs for products in same category/material groups

2. Recommended Monitoring Tools and Services

2.1 Metrics Collection and Visualization

- **Prometheus + Grafana**

- **Implementation**: Prometheus servers deployed in high-availability configuration
- **Data Retention**: 15 days of high-resolution metrics, 90 days of aggregated metrics
- **Custom Dashboards**:
- Executive Overview: High-level system health and business KPIs
- Operational Dashboard: Detailed system metrics for DevOps
- Data Quality Dashboard: Accuracy and consistency metrics
- Performance Dashboard: Latency and throughput metrics

2.2 Log Management

- **Primary Tool**: **ELK Stack** (Elasticsearch, Logstash)
- **Implementation**: Managed Elasticsearch Service with self-hosted Logstash
- **Log Aggregation**: Fluent Bit as log collector on Kubernetes nodes
- **Retention Policy**: 14 days of full logs, 90 days of error logs
- **Log Format**: Structured JSON with correlation IDs and standardized fields
- **Secondary Tool**: **CloudWatch Logs**
- **Implementation**: For AWS Lambda functions and critical infrastructure components
- **Integration**: Forwarded to Elasticsearch via Lambda subscription

2.3 Anomaly Detection

- **Primary Tool**: **Amazon Lookout for Metrics**
- **Implementation**: Machine learning-based anomaly detection
- **Focus Areas**:
- Unusual patterns in assessment results
- Unexpected changes in resource utilization
- Deviations in decision path distributions

3. Alerting Strategy

3.1 Alert Prioritization

- **P1 (Critical)**:
- Complete service outage
- Data loss or corruption
- Security breaches
- Response time: Immediate (24/7)
- **P2 (High)**:

- Partial service degradation
- Significant accuracy drops
- Component failures with active redundancy
- Response time: 30 minutes (24/7)
- **P3 (Medium)**:
- Performance degradation
- Minor accuracy issues
- Warning thresholds crossed
- Response time: 2 hours (business hours)
- **P4 (Low)**:
- Non-customer-impacting issues
- Technical debt markers
- Optimization opportunities
- Response time: Next business day

4. Incident Response Framework

4.1 Incident Classification

- **Severity 1**: Complete system outage or data integrity issue
- Full team response required
- Executive notification
- Post-incident review mandatory
- **Severity 2**: Major functionality impacted or significant performance degradation
- Dedicated incident response team
- Regular stakeholder updates
- Post-incident review mandatory
- **Severity 3**: Minor functionality impacted or slight performance degradation
- Small team response
- Post-incident review for recurring issues
- **Severity 4**: Cosmetic issues or isolated non-critical bugs
- Scheduled fix
- No formal review required

4.2 Incident Response Process

1. **Detection**:
- Automated alert triggered
- User-reported issue
- Proactive monitoring detection
2. **Triage**:
- Acknowledge alert within SLA
- Determine severity and impact
- Assemble appropriate response team
3. **Investigation**:
- Gather diagnostic information
- Review recent changes
- Identify potential root causes
4. **Mitigation**:
- Implement immediate actions to restore service
- Consider rollback of recent deployments
- Apply temporary fixes if needed
5. **Resolution**:
- Implement permanent solution
- Verify resolution across all affected components
- Update status page and notify stakeholders
6. **Post-Incident Review**:
- Conduct blameless postmortem
- Document timeline, root cause, and actions taken
- Identify preventative measures

5. Automated Recovery Mechanisms

5.1 Self-Healing Infrastructure

- **Kubernetes Liveness Probes**:
- Configured for all microservices
- Path: /health endpoint exposing critical dependencies
- Timeout: 5 seconds
- Period: 10 seconds

- Failure threshold: 3 consecutive failures triggers restart

5.2 Automated Remediation Runbooks

- **Database Connection Exhaustion**:
- **Detection**: Connection pool utilization > 90%
- **Actions**:

1. Increase connection pool size temporarily
2. Implement query timeout reduction
3. Activate read replica for read queries
4. Trigger connection cleanup procedure

- **High Memory Usage**:
- **Detection**: Container memory usage > 85%
- **Actions**:

1. Trigger JVM garbage collection
2. Clear application caches
3. Reduce batch processing size
4. Scale horizontally if possible

- **Kafka Consumer Lag**:
- **Detection**: Consumer lag > 1000 messages or 30 seconds
- **Actions**:

1. Scale consumer group instances
2. Activate high-throughput processing mode
3. Temporarily increase resource allocation
4. Reduce non-essential processing

5.3 Fallback Mechanisms

- **Degraded Mode Operations**:
- **Cache Fallback**:
- Return cached results when live analysis unavailable
- Clearly mark results as potentially outdated
- Set shorter TTL for fallback results
- **Simplified Processing**:
- Skip optional analysis steps under high load

- Reduce assessment depth automatically
- Focus on critical product attributes only
- **Default Values**:
- Apply predefined industry averages when specific data unavailable
- Clearly indicate estimation vs. actual analysis
- Apply confidence level indicators

6. Implementation and Maintenance

6.1 Monitoring Infrastructure Deployment

- **Infrastructure as Code**:
- Terraform modules for monitoring components
- Helm charts for Kubernetes monitoring deployments
- Configuration managed via GitOps workflow
- **Deployment Phases**:

1. Core metrics infrastructure (Prometheus, Grafana)
2. Log management system (ELK Stack)
3. Synthetic monitoring and anomaly detection
4. Alert integration and automation

6.2 Regular Review Procedures

- **Weekly Operational Review**:
- KPI performance against targets
- Recent incidents and resolution status
- Alert volume and signal-to-noise ratio
- Resource utilization trends
- **Monthly Comprehensive Review**:
- Accuracy validation results
- Monitoring coverage gaps
- Alert tuning opportunities
- Automation effectiveness
- **Quarterly Strategic Review**:
- Monitoring infrastructure scaling needs
- New KPIs and measurement methods

- Technology refresh considerations
- Cost optimization opportunities

6.3 Documentation and Knowledge Base

- **Runbook Management**:
- Maintained in version-controlled repository
- Linked directly from alerts
- Regularly tested through simulations
- Updated based on incident learnings
- **Dashboard Documentation**:
- Purpose and audience for each dashboard
- Metric definitions and calculation methods
- Threshold rationale and history
- Common troubleshooting patterns

This monitoring and evaluation strategy provides a comprehensive framework for maintaining optimal EcoAgent performance in a live environment. By focusing on critical KPIs across throughput, accuracy, latency, consistency, and data integration, the system can detect and respond to issues before they impact users. The multi-layered monitoring approach combines metrics and logs to provide complete visibility into the system's behavior. The structured alerting and incident response framework ensures timely and appropriate reactions to issues, while automated recovery mechanisms reduce manual intervention and minimize downtime. Regular reviews and continuous improvement of the monitoring system itself will ensure that as the EcoAgent evolves, the ability to observe and maintain it evolves accordingly.