

Software project

Analysis and monitoring of product upload dynamics on the marketplace platform

About

This project develops a system to collect, process, and analyze large volumes of seller product-upload events on the Ozon marketplace



About

This project develops a system to collect, process, and analyze large volumes of seller product-upload events on the Ozon marketplace

Focus areas

- Development of architecture
- Statistical metrics computation
- Visualization
- Automated alerting

Technologies



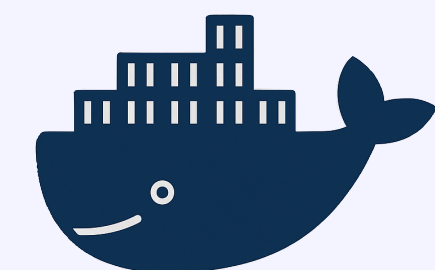
Clickhouse



Prometheus



Grafana



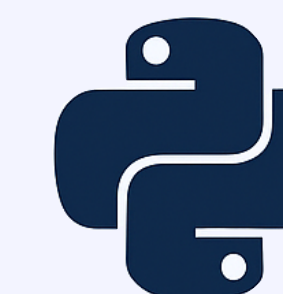
Docker



Consul



Zookeeper



Python



Kafka

Problem statement

Ozon's marketplace generates a high-velocity stream of product load events from thousands of sellers.



Key challenges

- Handling millions of small, frequent events per day
- Ensuring data consistency and replication
- Delivering real-time analytics and alerts

Tasks

Step 1. Draft requirements

Domain description, functional & non-functional specs, constraints

Step 2. Compare alternatives

Step 4. Test the system

Step 3. Implement solution

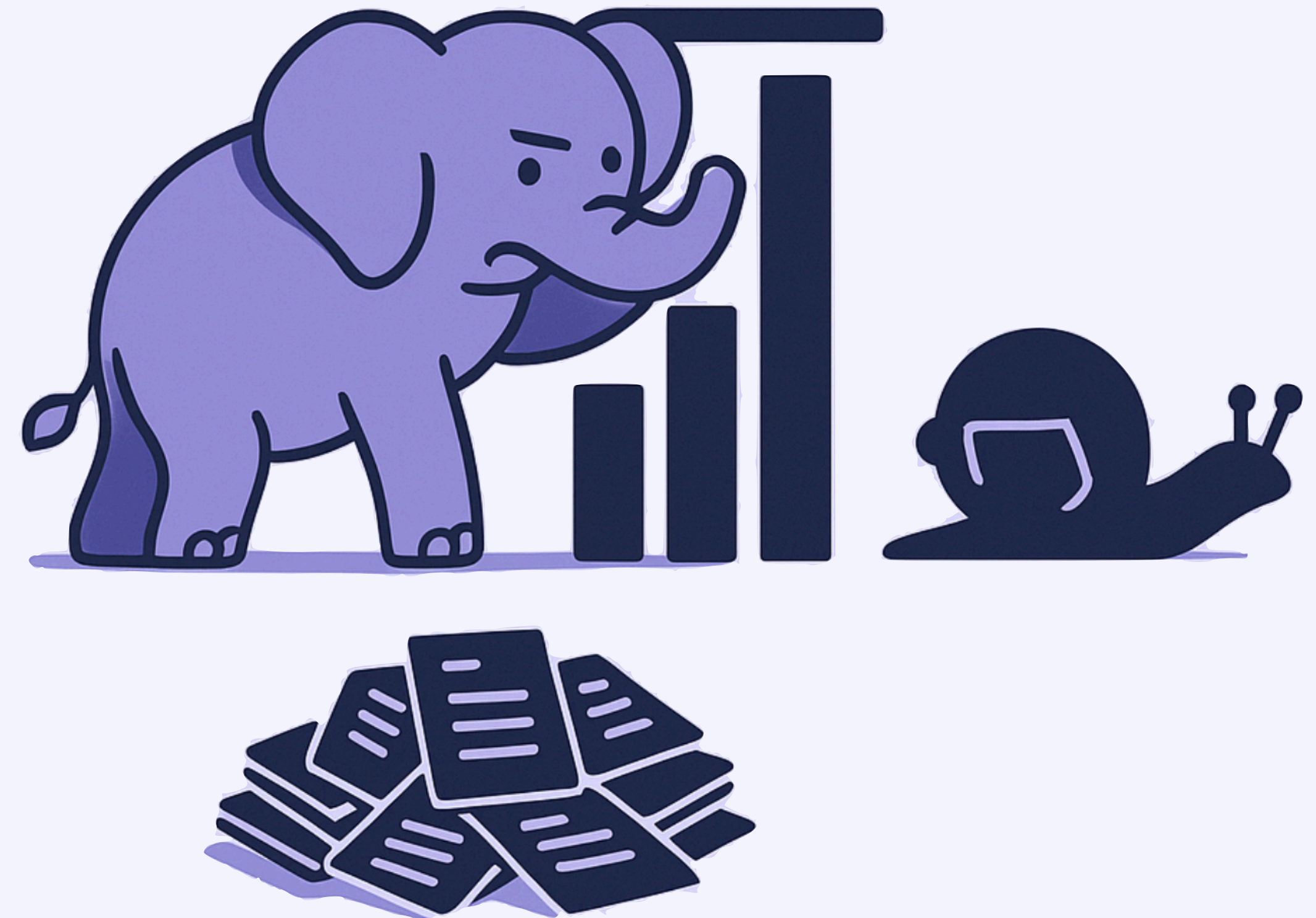
- Implement database sharding locally (via zookeeper)
- Deploy ClickHouse, Grafana, Prometheus (via docker-compose)
- Create microservices that will simulate data stream and compute metrics
- Connect & register microservices (via consul)
- Write simulation script
- Create leader election for microservices
- Develop ETL pipelines & pre-aggregations
- Build anomaly detection modules

Limitations of PostgreSQL

Vertical scaling limits

WAL/MVCC overhead under heavy write load

Slow OLAP queries on growing datasets



Limitations of PostgreSQL

Scalability



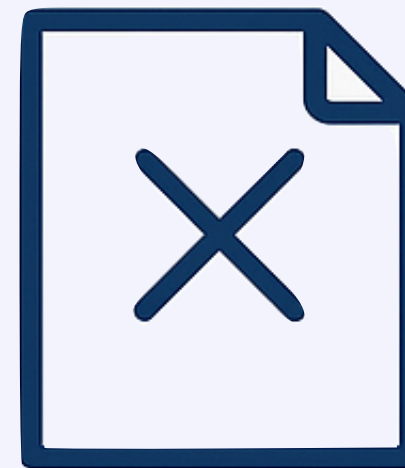
only vertical or external sharding (Citus/Postgres-XL)

Storage



row-oriented, sub-optimal for analytics.

Transactions



single-node ACID; no built-in 2PC across cluster

Maintenance



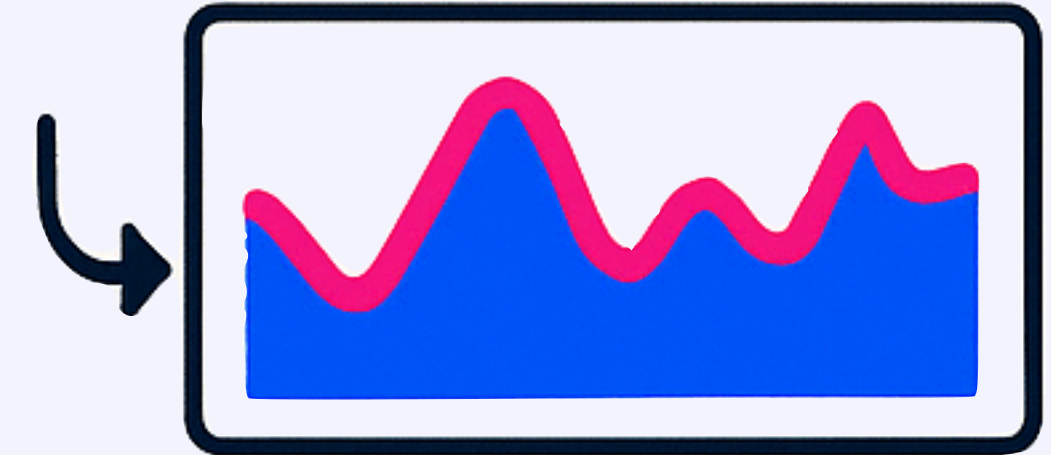
long VACUUM, index rebuilds on TB-scale tables

Write throughput



WAL contention and locking delays

Functional Requirements



The system has to:

- Survive the ingestion of maximum 1000 rps
- Keep one year of history online
- Provide **control-charts** for upload latency, failure rate, and seller anomalies
- Trigger **Prometheus alerts** when specific metrics exceed the limit

Alternative approaches



Cassandra

A highly scalable database optimized for fast writes, but less suitable for real-time analytics due to eventual consistency.



Apache Druid

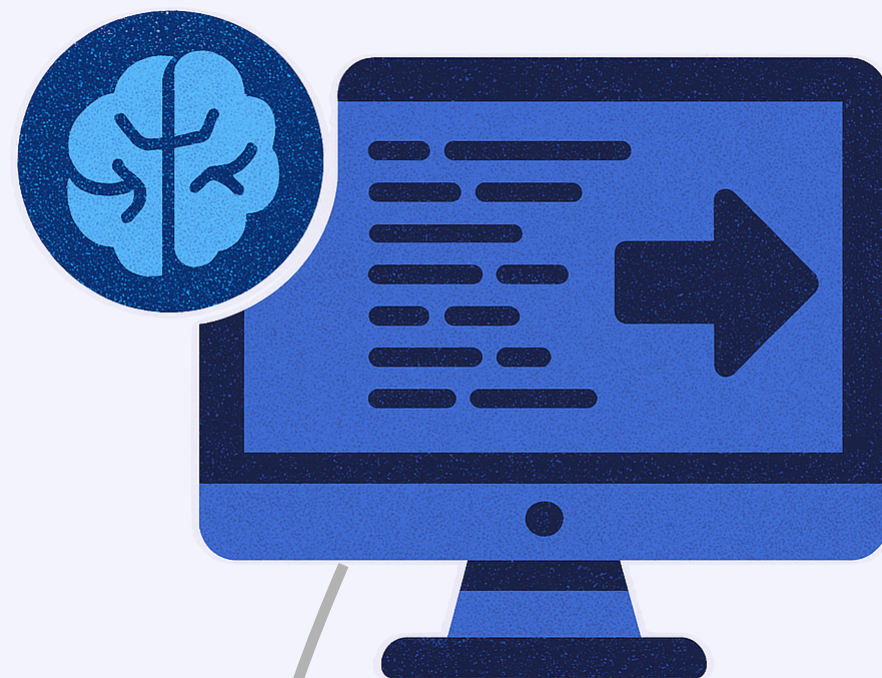
A fault-tolerant, distributed analytics engine built for fast, interactive queries over large-scale time-series data.

Implementation



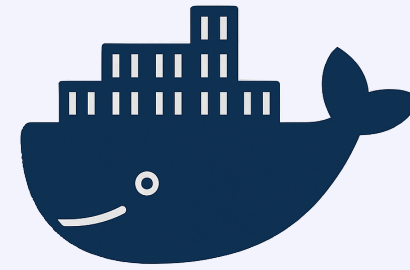
Architecture Overview

Synthetic data generation



id	Unique identifier of the item
ts_ns	Event timestamp in nanoseconds
event	Type of event: "SAVE", "UPDATE", "ERROR"
attempt_id	Identifier for this upload attempt
is_created	Boolean flag
origin	Source of the operation
up_ts	If is_created=1, a ns ts when creation occurred; or 0
country	Production country code
data	JSON: info, company ID and list of media files

Architecture Overview



docker desktop PERSONAL

Search for images, containers, volumes, extensions and more... %K

Containers

Images

Volumes

Builds

Docker Scout

Extensions

Containers [Give feedback](#)

Container CPU usage ⓘ
No containers are running.

Container memory usage ⓘ
No containers are running.

[Show charts](#)

Search

Only show running containers

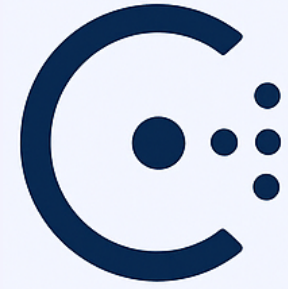
Delete

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
☒	ozon	-	-	-	N/A	2 hours ago	▶ ⋮ 🗑️
☐	consul	87f1e76361f8	consul:1.15.4	8301:8301 Show all ports (4)	N/A	2 hours ago	▶ ⋮ 🗑️
☐	zookeeper	8a1ff273730b	zookeeper:3.8	2181:2181	N/A	2 hours ago	▶ ⋮ 🗑️
☐	prometheus	677e6956bdb1	prom/prometheus:v2.45.0	9090:9090	N/A	2 hours ago	▶ ⋮ 🗑️
☐	clickhouse-2	c170e4a7eed2	clickhouse/clickhouse-server:24.4-alpine	8125:8123 Show all ports (2)	N/A	2 hours ago	▶ ⋮ 🗑️
☐	clickhouse-base-1	d918a0da9a6b	clickhouse/clickhouse-server:24.4-alpine		N/A	2 hours ago	▶ ⋮ 🗑️
☐	clickhouse-3	ba9abed0d71c	clickhouse/clickhouse-server:24.4-alpine	8126:8123 Show all ports (2)	N/A	2 hours ago	▶ ⋮ 🗑️
☐	clickhouse-4	02358479d82d	clickhouse/clickhouse-server:24.4-alpine	8127:8123 Show all ports (2)	N/A	2 hours ago	▶ ⋮ 🗑️
☐	kafka	489589d46fd6	confluentinc/cp-kafka:latest	9092:9092	N/A	2 hours ago	▶ ⋮ 🗑️
☐	clickhouse-1	a4966fe2ca91	clickhouse/clickhouse-server:24.4-alpine	8124:8123 Show all ports (2)	N/A	2 hours ago	▶ ⋮ 🗑️
☐	ch-init-1	88a89c908f4a	clickhouse/clickhouse-server:24.4-alpine		N/A	2 hours ago	▶ ⋮ 🗑️
☐	microservice-2	6e9401794bbe	ozon-microservice		N/A	2 hours ago	▶ ⋮ 🗑️
☐	microservice-3	abdadf07ca12	ozon-microservice		N/A	2 hours ago	▶ ⋮ 🗑️
☐	microservice-1	cbd322408c66	ozon-microservice		N/A	2 hours ago	▶ ⋮ 🗑️
☐	grafana	e269838dd7b2	grafana/grafana:11.5.0	3000:3000	N/A	2 hours ago	▶ ⋮ 🗑️
☐	kafka-init	d7564fc78bfb	confluentinc/cp-kafka:latest		N/A	2 hours ago	▶ ⋮ 🗑️

Selected 1 of 16

System running via docker

Architecture Overview

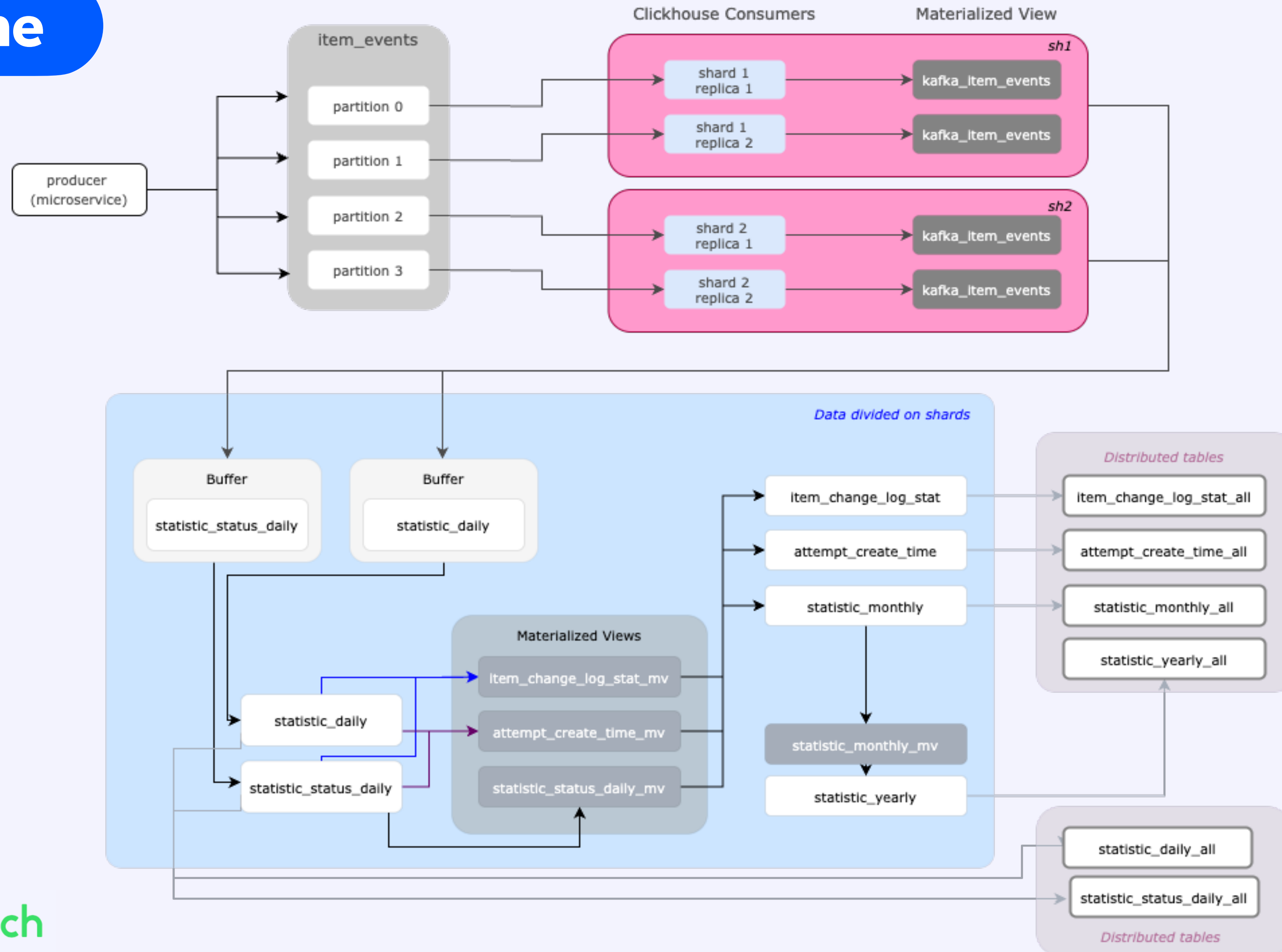


The screenshot displays the Consul web interface for a service named 'microservice'. The left sidebar contains navigation links: Overview, Services, Nodes, Key/Value, Intentions, ACCESS CONTROLS (with a red indicator), Tokens, Policies, Roles, Auth Methods, ORGANIZATION, and Peers. The main content area shows the 'microservice' page with tabs for Instances, Intentions, Routing, and Tags. The 'Instances' tab is active, displaying a list of three service instances, each with a green checkmark indicating 'All service checks passing' and 'All node checks passing'. The instances are identified by their IDs: microservice-6e9401794bbe, microservice-abdadf07ca12, and microservice-cbd322408c66. Each instance also shows its IP address (87f1e76361f8), a port (84), and a metrics link. A search bar and a 'Search Across' dropdown are located above the instance list. A 'Health Status' dropdown and a filter for 'Unhealthy to Healthy' are also present. The bottom of the interface shows 'Consul v1.15.4'.

Instance ID	Health Status	Node Checks	Service Checks	IP Address	Port	Metrics
microservice-6e9401794bbe	✓ All service checks passing	✓ All node checks passing	✓ All service checks passing	87f1e76361f8	6e9401794bbe:84	metrics
microservice-abdadf07ca12	✓ All service checks passing	✓ All node checks passing	✓ All service checks passing	87f1e76361f8	abadf07ca12:84	metrics
microservice-cbd322408c66	✓ All service checks passing	✓ All node checks passing	✓ All service checks passing	87f1e76361f8	cbd322408c66:84	metrics

Orchestration of
microservice instances

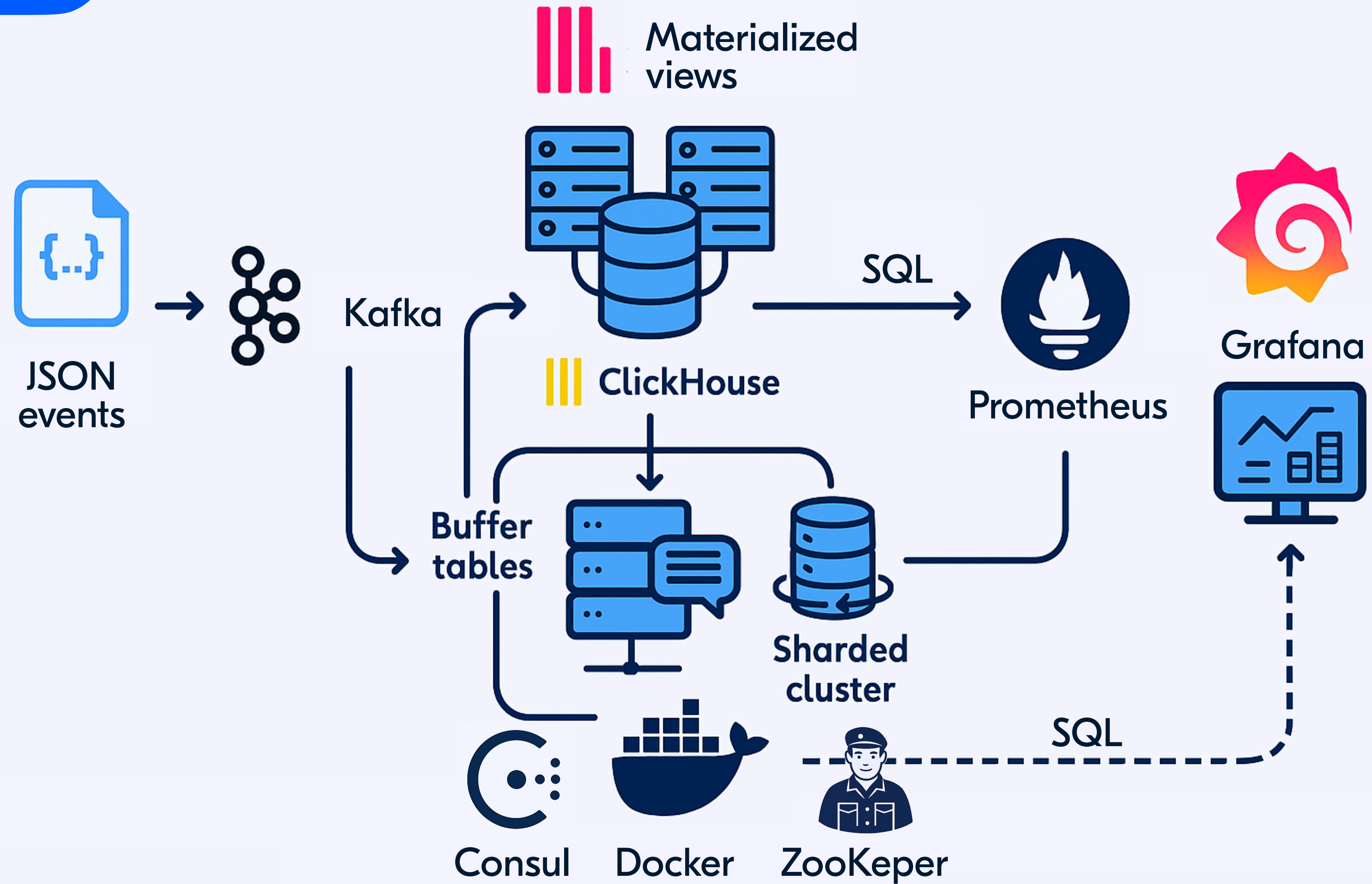
Pipeline



Pipeline

```
79 def on_lose_leadership():
80     logger.warning("Lost leadership")
81
82
83
84
85
86 def leader_election_loop():
87     session_id = create_session()
88     is_leader = False
89     while True:
90         try:
91             got = acquire_lock(session_id)
92             if got and not is_leader:
93                 is_leader = True
94                 on_become_leader()
95             elif not got and is_leader:
96                 is_leader = False
97                 on_lose_leadership()
98
99             if not renew_session(session_id):
100                 logger.warning(f"Session expired: {session_id}, creating new one")
101                 session_id = create_session()
102                 if is_leader:
103                     is_leader = False
104                     on_lose_leadership()
105
106         except Exception as e:
107             logger.exception("Leader election loop error")
108
109     delta = BASE * JITTER_FACTOR
110     sleep_time = BASE + random.uniform(-delta, delta)
```


Pipeline

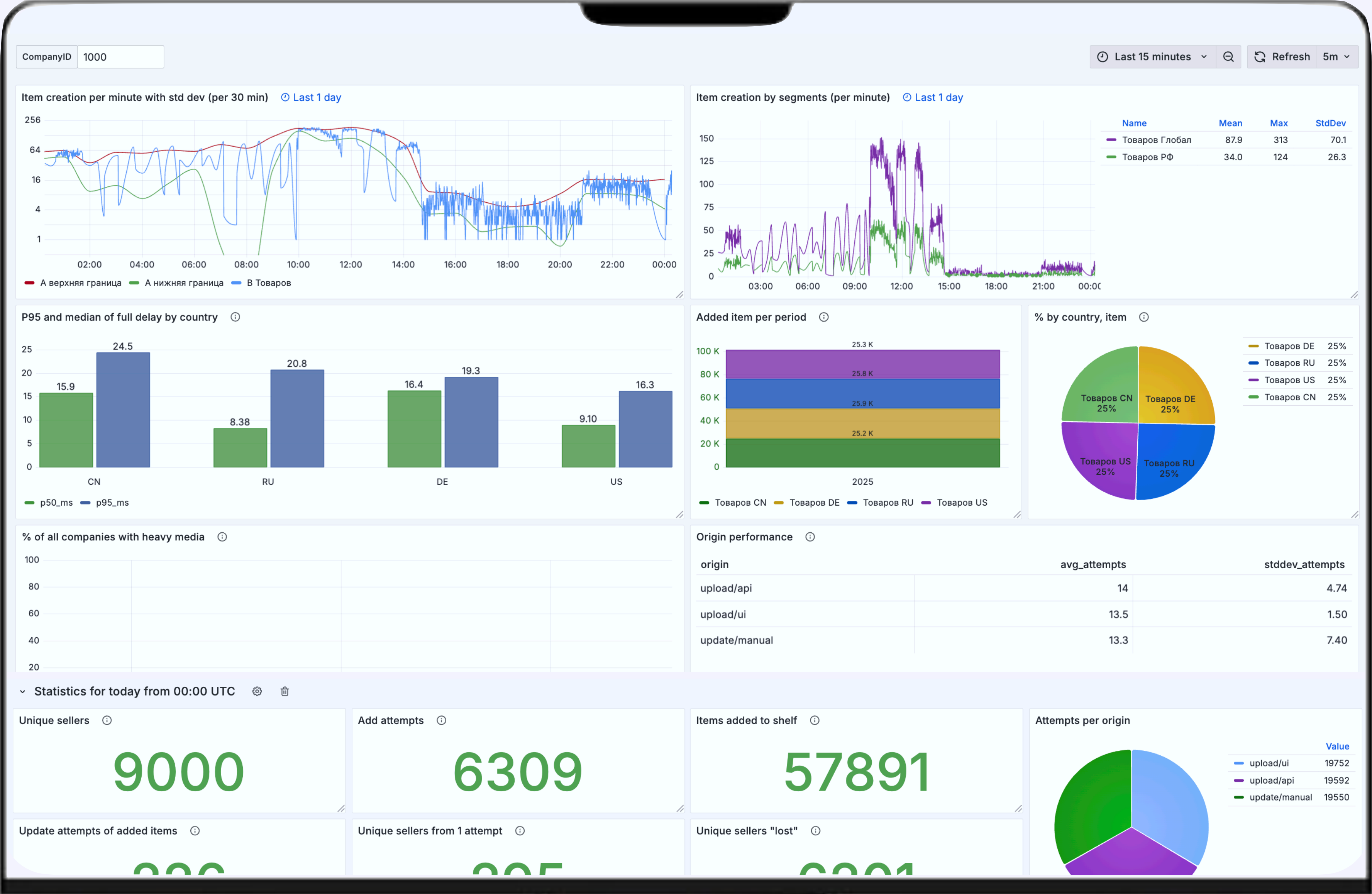


Results & Testing

Vizualization



Vizualization



Alerts



Prometheus Alerts Graph Status Help

Inactive (4)

Pending (2)

Firing (0)

Filter by name or labels

Show annotations

/etc/prometheus/alert_rules.yml > item_upload_alerts

HighFailureRate (1 active)

name: HighFailureRate

expr: create_time_all_failure_rate_percent > 50

for: 5m

labels:
 component: item_upload
 severity: critical

annotations:
 description: The percentage of failed upload attempts (is_created = false) has exceeded 50% over the last 5 minutes. Check microservice logs and ClickHouse ingestion pipeline for potential errors.
 summary: High failure rate in item uploads ({{ \$value }}% for 5m)

Labels	State	Active Since	Value
alertname=HighFailureRate component=item_upload instance=870c128fee88:84 job=from-consul severity=critical	PENDING	2025-06-02T18:24:15.377108999Z	99.21259842519684

HighAverageAttemptTime (0 active)

name: HighAverageAttemptTime

expr: company_statistic_daily_avg_attempt_time > 30

for: 5m

labels:
 component: item_upload
 severity: warning

annotations:
 description: The average time to perform the first creation attempt (in seconds) has stayed above 30 seconds for at least 5 minutes. Investigate ClickHouse query performance, Kafka lag, or microservice processing delays.
 summary: High average attempt time (>{{ \$value }}s for 5m)

HighAttemptTimeStdDev (0 active)

FewUniqueSellers (2 active)

name: FewUniqueSellers

expr: company_statistic_daily_unique_sellers < 100

for: 1m

labels:
 component: item_upload
 severity: warning

annotations:
 description: Over the past day, fewer than 100 distinct sellers have uploaded items.

Firing alert

Pending alert

Inactive alert

Tests

Reference:

Ozon Production Cluster

- 2 replicas (non-sharded)
- 8 vCPU, 48 GB RAM, 700 GB SSD
- 1,000 RPS peak

Test setup:

Local Sharded Cluster

- 2 shards $\times$ 2 replicas
- Load balanced across shards
- No increase in latency

- **Linear scalability** by adding shards;
 - Ready to handle higher loads
- 

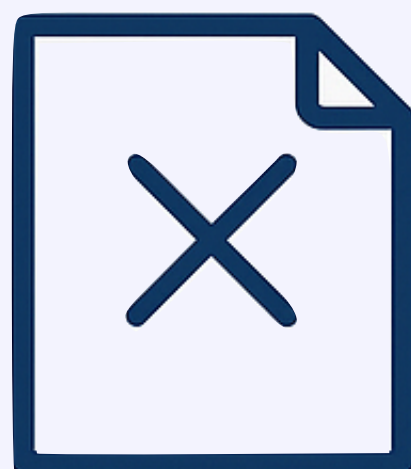
Issues

Data skew



Effect of random
distribution in
Python

Synthetic traffic



Lacks specific
patterns

Limited testing



Pipeline with
scripted chaos tests
required

Cluster orchestration



Deployment with
Kubernetes may be
an improvement

References

- [1] ACID-transactions in PostgreSQL. Accessed: 2025-05-10. 2025. url: <https://blog.ydb.tech/sharded-is-not-distributed>
- [2] Denis Sheahan Adrian Cockcroft. Benchmarking Cassandra Scalability on AWS. url: <https://netflixtechblog.com/benchmarking-cassandra-scalability>
- [3] T. Akidau et al. Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing. O'Reilly Media, 2018.
- [4] Apache Cassandra Documentation. Accessed: 2025-02-03. 2025. url: <https://cassandra.apache.org/doc/latest/>.
- [5] Apache Druid Documentation. Accessed: 2025-02-03. 2025. url: <http://druid.apache.org/docs/latest/>.
- [6] J. Carpenter and E. Hewitt. Cassandra: The Definitive Guide: Distributed Data at Scale. 2nd. O'Reilly Media, 2016.
- [7] Citus. Accessed: 2025-20-05. 2025. url: <https://demirhuseyinn-94.medium.com/scaling-horizontally-on-postgresql-cituss-impact-on-database-architecture>
- [8] ClickHouse Documentation. Accessed: 2025-02-03. 2025. url: <https://clickhouse.com/docs/en/>.
- [9] Horizontal Scaling PostgreSQL. Accessed: 2025-28-05. 2025. url: <https://www.cockroachlabs.com/blog/limitations-of-postgres/>.
- [10] Multiversion concurrency control. Accessed: 2025-18-05. 2025. url: [https://en.wikipedia.org/wiki/PostgreSQL#Multiversion_concurrency_control_\(MVCC\)](https://en.wikipedia.org/wiki/PostgreSQL#Multiversion_concurrency_control_(MVCC)).
- [11] Postgre XL, github. Accessed: 2025-18-05. 2025. url: <https://github.com/pharosnet/postgres-xl>.
- [12] Round-Robin scheduling algorithm. Accessed: 2025-07-10. 2025. url: https://en.wikipedia.org/wiki/Round-robin_scheduling.
- [13] Write-ahead logic. Accessed: 2025-18-05. 2025. url: https://en.wikipedia.org/wiki/PostgreSQL#Storage_and_replication.