**1. Data Sources (100k TPS)**

**Applications**:

* ISO 8583 transaction services (authorization, clearing, fraud detection).
* Legacy systems (mainframes) and microservices.
* **Instrumentation**:
  + **OpenTelemetry SDKs**: Auto-instrument apps for traces, metrics, and logs.
  + **Custom Trace Injection**: Embed trace\_id into ISO 8583 messages (e.g., DE 48 or reserved fields).

**Infrastructure**:

* Servers, switches, payment gateways.
* **Telegraf Agents**: Collect node metrics (CPU, disk, network).

**2. Data Collection & Ingestion**

**Metrics & Traces**:

* **OpenTelemetry Collectors**:
  + Deployed as DaemonSets (Kubernetes) or sidecars (VMs) on every transaction node.
  + **Functions**:
    - Collect traces, metrics, and logs.
    - Perform head-based sampling (10% for traces, 100% for errors).
    - Redact sensitive fields (PANs) using OTel processors.
  + **Output**: Send data to Kafka topics (traces, metrics, logs).

**Logs**:

* **Filebeat/Logstash**:
  + Ship ISO 8583 transaction logs to Kafka.
  + Enrich logs with trace\_id from OpenTelemetry context.

**3. Central Data Bus (Apache Kafka)**

**Cluster**:

* **7 Brokers** (bare-metal servers or high-CPU VMs).
  + **Specs**: 32 vCPU, 128GB RAM, 10TB NVMe per broker (RAID-10).
* **Topics**:
  + traces: 100 partitions (for parallel processing).
  + metrics: 50 partitions.
  + logs: 50 partitions.
* **Schema Registry**: Enforce Avro schemas for ISO 8583 message validation.

**4. Stream Processing (Apache Flink)**

**Cluster**:

* **10 Task Managers** (16 vCPU, 64GB RAM each).
* **Key Tasks**:
  1. **Trace Enrichment**: Add merchant metadata, GeoIP, and business context.
  2. **Tail-Based Sampling**: Retain 100% of error traces and 5% of successful ones.
  3. **Data Routing**:
     + Traces → Tempo/Jaeger.
     + Metrics → VictoriaMetrics.
     + Logs → Elasticsearch.
  4. **Compliance**: Mask PANs/CVVs in real-time.

**5. Storage Layer**

**A. Tracing**:

* **Grafana Tempo** (Scalable and Cost-Effective):
  + **Backend**: MinIO (on-prem S3) for trace storage.
  + **Specs**: 5-node MinIO cluster (32 vCPU, 128GB RAM, 50TB HDD each).
  + **Retention**: 30 days (adjustable via MinIO lifecycle policies).

**B. Metrics**:

* **VictoriaMetrics Cluster** (Prometheus-Compatible):
  + **Ingest Nodes**: 5x (16 vCPU, 64GB RAM, 2TB NVMe) – Handle 500k samples/sec.
  + **Storage Nodes**: 10x (32 vCPU, 128GB RAM, 10TB SSD) – 3x replication.
  + **Retention**: 2 years (downsampled after 15 days).

**C. Logs**:

* **Elasticsearch Cluster** (Hot-Warm Architecture):
  + **Hot Nodes**: 10x (32 vCPU, 128GB RAM, 4TB NVMe) – For recent logs.
  + **Warm Nodes**: 10x (16 vCPU, 64GB RAM, 16TB HDD) – For older logs.
  + **Indexing**: 1k+ writes/sec per node.

**6. Query & Alerting Layer**

**A. Query Services**:

* **PromQL**: VictoriaMetrics for metrics (e.g., rate(iso8583\_errors[1m])).
* **TraceQL**: Grafana Tempo for traces (e.g., { status="failed" && duration > 2s }).
* **Elasticsearch**: Logs correlated with trace\_id.

**B. Alerting**:

* **VictoriaMetrics Alerting**:
  + Rules: ALERT HighFailureRate IF rate(iso8583\_failures[5m]) > 1%.
* **Grafana Alerting**:
  + Notify via Slack/Email/PagerDuty.
* **Splunk** (Optional):
  + Fraud detection alerts (e.g., stats count by user\_id | where count > 1000).

**7. Visualization & Analysis**

**A. Grafana**:

* **Dashboards**:
  + **Real-Time Transactions**: Success rate, latency heatmaps.
  + **Trace Waterfalls**: Drill into slow transactions.
  + **Infrastructure Health**: Node CPU/memory from VictoriaMetrics.

**B. Kibana**:

* **Log Analysis**: Filter by trace\_id to debug declines.
* **Lens Visualizations**: GeoIP maps of transaction origins.

**C. Jaeger UI** (Optional):

* Alternative trace viewer for deep dependency analysis.

**8. Security & Compliance**

**A. Data Protection**:

* **Encryption**: TLS everywhere (Kafka, Elasticsearch, MinIO).
* **Masking**: PANs/CVVs stripped in Flink/OTel Collectors.
* **Access Control**:
  + Grafana/Kibana: LDAP/AD integration.
  + **RBAC**: Restrict access to PCI data (e.g., only fraud team sees PANs).

**B. Audit & Compliance**:

* **Splunk**: Retain audit logs for 7 years (PCI-DSS).
* **MinIO**: Immutable backups for forensic analysis.

**9. Scalability & HA**

**A. Kafka**:

* **MirrorMaker**: Replicate data to a DR site.
* **Monitoring**: Burrow for consumer lag alerts.

**B. VictoriaMetrics**:

* **Replication**: 3x cross-datacenter replication.

**C. Elasticsearch**:

* **Index Rollover**: Daily indices with 50 shards each.

**10. Hardware Sizing (Example)**

| **Component** | **Nodes** | **Specs (Per Node)** | **Notes** |
| --- | --- | --- | --- |
| **Kafka Brokers** | 7 | 32 vCPU, 128GB RAM, 10TB NVMe | 100k TPS with 70% headroom. |
| **Flink Task Mgrs** | 10 | 16 vCPU, 64GB RAM | Autoscale during peak hours. |
| **VictoriaMetrics** | 15 | 16–32 vCPU, 64–128GB RAM, 10TB SSD | Separate ingest/storage roles. |
| **Elasticsearch** | 20 | 16–32 vCPU, 64–128GB RAM, 4–16TB SSD | Hot-warm architecture. |
| **Grafana Tempo** | 5 | 16 vCPU, 64GB RAM, 50TB HDD | MinIO for scalable trace storage. |

**Data Flow for 100k TPS**

ISO 8583 Apps → OTel Collector → Kafka → Flink → Tempo/VictoriaMetrics/Elasticsearch

↓ ↓ ↓

Telegraf → Kafka → VictoriaMetrics (Infra Metrics)

↓

Zabbix → Alertmanager (Infra Alerts)

**Why This Works at 100k TPS**

1. **Kafka as a Buffer**: Handles traffic spikes and decouples components.
2. **Flink for Scalability**: Distributed processing ensures data is enriched/routed in real-time.
3. **Cost-Effective Storage**:
   * VictoriaMetrics: 10x compression vs. Prometheus.
   * MinIO: Cheap HDD storage for traces.
4. **Compliance**: PAN masking, audit trails, and access controls meet PCI-DSS.
5. **Unified Observability**: Traces, metrics, and logs share trace\_id for cross-correlation.

**Example Troubleshooting Workflow**

1. **Alert**: VictoriaMetrics triggers HighFailureRate.
2. **Investigate**: Grafana dashboard shows failures correlate with high latency.
3. **Trace Analysis**: Query Tempo for traces with status="failed" to find stuck spans.
4. **Logs**: Use Kibana to search logs by trace\_id for error messages.
5. **Fix**: Identify faulty fraud service node and reroute traffic.