NANOG

From Testing to Reliability:

Detecting Silent Failures in Production Systems

When Infrastructure Is Green and Your System Is Still Wrong

Shruthi Sepuri

Software Development Engineer · Production Systems Reliability

When a Healthy System Still Produces Wrong Outputs

INFRASTRUCTURE STATUS

Uptime	99.99%
Error Rate	0.01%
P99 Latency	42ms
CPU Usage	31%
Memory	48%

● ALL SYSTEMS OPERATIONAL

VS

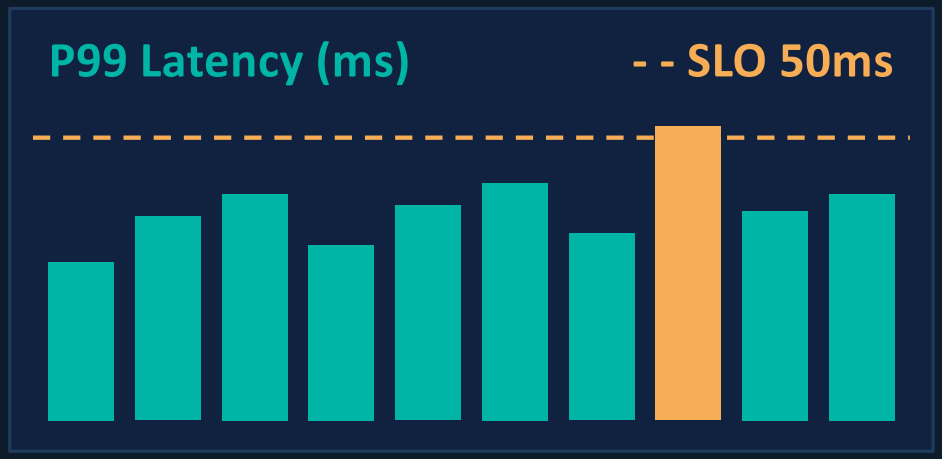
SERVICE PIPELINE

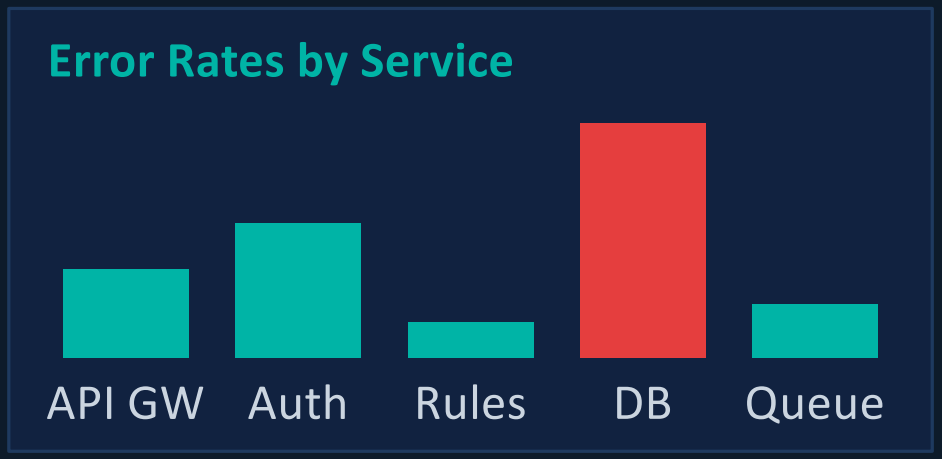
Data Ingestion	✓ PASS
Rule Engine Processing	✗ ERROR
Decision Output	✗ WRONG
Config Validation	✗ SKIPPED
Audit Trail	✗ GAP

⚠ Rule misconfiguration — no infrastructure alert triggered

How Reliability Is Typically Measured

Service Availability	Error Rates	Latency (P99)	CPU & Memory	Infrastructure Uptime
99.9%	0.02%	38ms	45%	100%





These metrics measure infrastructure health — but not service output correctness.

Reliability Beyond Infrastructure

INFRASTRUCTURE LAYER
Servers · Databases · Networks · Cloud Resources
● MONITORED

APPLICATION SERVICES
APIs · Microservices · Queues · Caches
● MONITORED

SERVICE PIPELINE LAYER ← THE MISSING LAYER
Config-Driven Services · Rule Execution · Automated Decision Pipelines · Event Processing · Audit Logging
● UNDER-MONITORED

SYSTEM CORRECTNESS
Correct Outputs · SLO Adherence · Audit Trails · Change Safety
● UNMEASURED

A system can be fully operational while its automated outputs are wrong.

Example: Silent Workflow Failure

SCENARIO

1. Config-driven service shows normal infrastructure metrics
2. A config change silently alters service decision logic
3. Infrastructure monitoring remains green throughout
4. Service outputs drift from expected behavior — undetected by monitoring.

IMPACT

✗ Incorrect outputs propagated at scale across the service pipeline

✗ Audit trail incomplete — gaps in system-of-record for the change

✗ No infrastructure alert triggered — invisible to on-call

Infrastructure health does not guarantee service output correctness.

Define Reliability at the Workflow Level



PIPELINE-LEVEL RELIABILITY INDICATORS

Output Correctness Rate
% of automated service outputs falling within expected behavior range

Config Coverage Completeness
Percentage of required validation attributes present per transaction

Output Distribution Anomaly
Statistical deviation from expected output distribution baseline

Rule Execution Correctness
Rules executing as configured — validated against known test cases

Hidden Risks in Rule-Driven Systems

Config error — no alert fired

Data Ingestion	Validation Service	Rule Engine	Decision Output	Downstream Processing
HEALTHY	HEALTHY	SILENT FAIL	PROPAGATED	PROPAGATED

Infrastructure monitoring layer — all green, no alerts triggered

COMMON CAUSES OF SILENT FAILURES

Rule Config Errors
Wrong thresholds or logic

Decision Threshold Drift
Baseline shifts go undetected

Data Mapping Issues
Field mismatches pass wrong values

Missing Validation Attributes
Required fields absent or unchecked

Silent failures propagate incorrect outputs, introduce change risk, and create operational exposure — while infrastructure remains green.

Extending Testing into Production

Testing
Coverage & risk design

Deployment
Release gates

Monitoring
Service SLOs

Incident Learning
Postmortems

Improved Testing
Feedback loop

PRACTICAL IMPLEMENTATION METHODS

1 **Synthetic Service Traffic**
Replay realistic requests against live logic

2 **Service Pipeline Monitoring**
Track outputs and rule execution in real time

3 **Rule Validation in Production**
Diff outputs vs. known-good after each change

4 **Decision Anomaly Detection**
Alert on output shifts, not just error rates

Testing evolves into continuous production behavior validation.

How Teams Can Implement Production Reliability

1	**Define Pipeline-Level Reliability Indicators** Instrument service output accuracy, config coverage, and rule execution validity — in addition to uptime and latency.
2	**Monitor Automated Service Outputs** Track output distributions and service behavior ranges. Alert when patterns deviate from expected baselines.
3	**Detect Anomalies in Output and Behavior Patterns** Use statistical thresholds to surface behavior drift before it compounds into production incidents.
4	**Feed Production Behavior Back into Testing** Use real output distributions to calibrate test cases, update thresholds, and validate config changes pre-deployment.

Testing + Observability = Workflow Reliability

Redefining Reliability in Distributed Production Systems

Infrastructure Monitoring

- Uptime & error rates
- Latency & throughput
- Resource utilization

Necessary — but not sufficient

Workflow Reliability

- Decision accuracy rates
- Rule execution validity
- Anomaly detection on outputs

The missing layer

System Correctness

- Audit trail completeness
- Output correctness verification
- Correct automated outcomes

The SRE view

Reliability in distributed production systems means ensuring service outputs remain correct, observable, and change-safe.

Your Next Steps

- [ ] Pick one config-driven pipeline and list its decision outputs
- [ ] Add one output-correctness metric to the on-call dashboard
- [ ] Gate every config change: replay known-good cases, diff the results
- [ ] Alert on output distribution shifts, not only on errors
- [ ] Turn your next incident into a regression test case

Start with one pipeline. You don't need a new platform.

Thank You

Reliability in production systems means ensuring service outputs remain correct, observable, and change-safe.

Questions?

Shruthi Sepuri · NANOG