

# Closing the State Gap in Large-Scale Network Monitoring

From Assumption to Assurance

## Stephen Stack - CTO



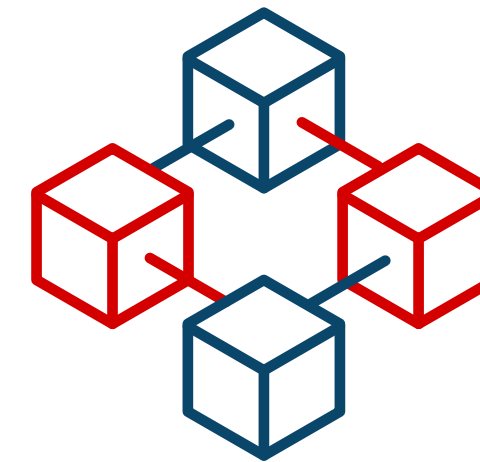
OCTOBER 8 • 10, 2025  
RIGA • LATVIA

# A system that looked healthy—until it wasn't

Dell | 20k+ devices | global footprint

- ✓ Logs: plenty of signals, no alarms
  - ✓ Dashboards: all green, SLIs on target
  - ✓ Backups: nightly, tested, versioned
- 
- ▶ Runtime state can drift even when configs are “compliant.”
  - ▶ Dashboards show symptoms; they don't prove intent is satisfied.
  - ▶ Backups capture text, not behavior.

■ Still got blindsided.



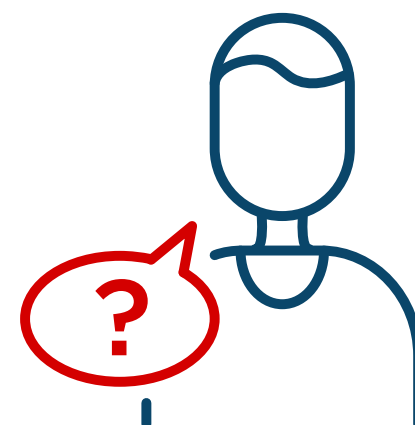
**“not all risk lives in logs, and not all change happens in config”**

# Where we look vs where problems hide

Misplaced confidence creates blind spots

Risk Area	We think it lives in...	Reality
Unauthorized changes	Syslog	CLI / live device
Routing failure	Config	Runtime FIB / session state
Compliance drift	Git repo	Manual edits / post-deploy “fix”
App behavior	Code	Side-effects in infra
TLS breakage	Cert store	Live handshake / chain validation
Capacity exhaustion	Dashboard SLAs	Queue depth / kernel counters
Security posture	Policy docs	Effective ACLs on path
HA readiness	Design diagram	Failover time under load

- ▶ Git ≠ ground truth.
- ▶ Green graphs ≠ correct behavior.
- ▶ Backups capture text, not outcomes.



**So where does the truth live?**

# The Trifecta of Truth

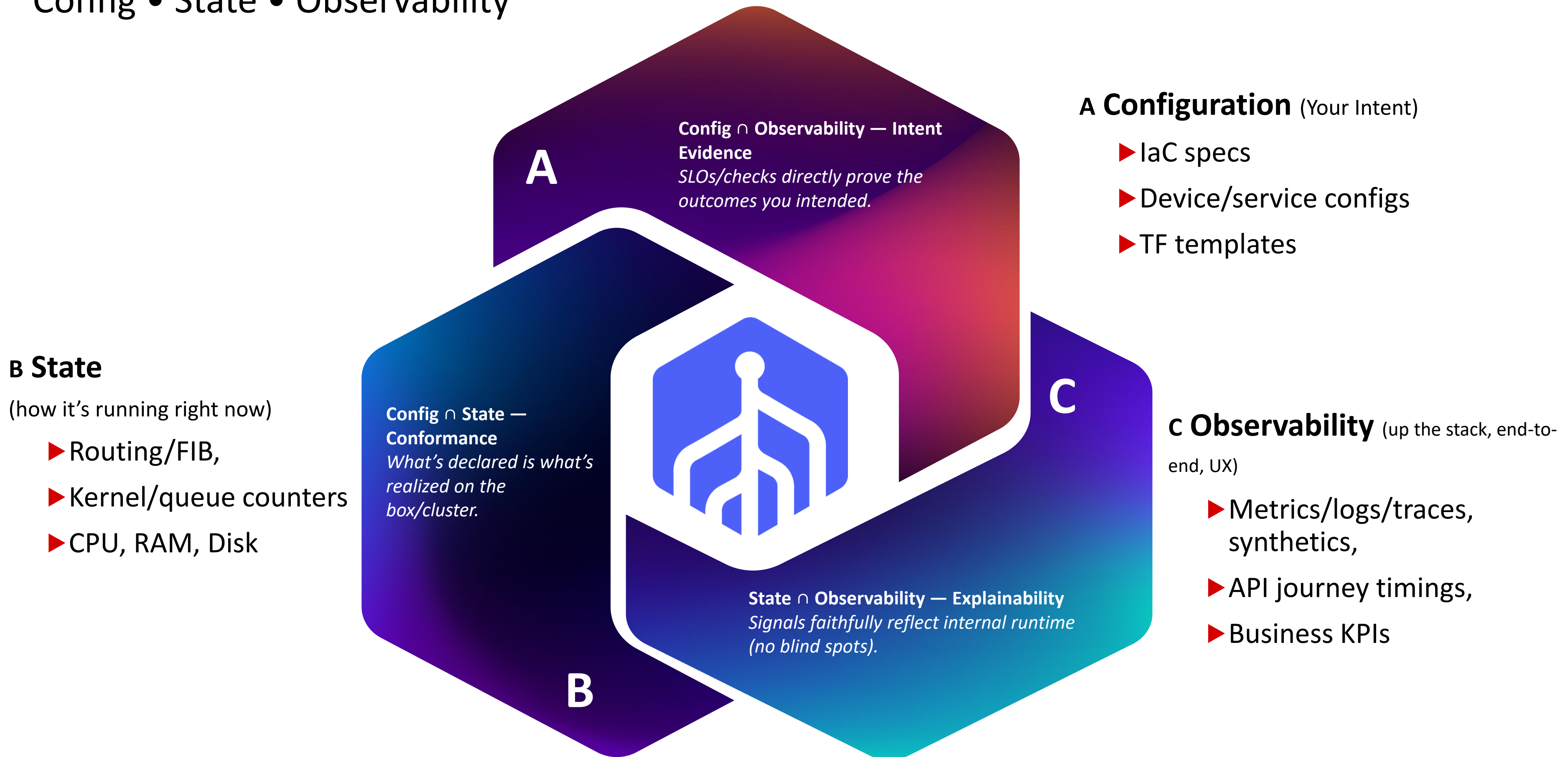
Config (Intent) • State • Observability



OCTOBER 8 • 10, 2025  
RIGA • LATVIA

# The Trifecta of Truth

Config • State • Observability



# The Trifecta of Truth

Config • State • Observability

**Config  $\cap$  State  $\cap$  Observability — Continuous Assurance  
(Verified Outcomes)**

*Design  $\rightarrow$  realization  $\rightarrow$  user-visible proof, all aligned.*

**“If it isn’t realized in state and proven in observability, it doesn’t matter that it’s written in config.”** - Stephen Stack

# The Trifecta of Truth

Overlaps you must verify (this is where risk is reduced):

## Config $\cap$ State — Conformance tests

“Is the route/interface/policy actually **installed** and active?”  
Auto-fail CI/CD if post-deploy state checks don’t match intent.

## State $\cap$ Observability — Explainability tests

“Do our metrics/traces/logs **move** when state changes?” (e.g., drop a route, see latency/availability impact as expected).  
Synthetic probes cover each critical failure mode.

## Config $\cap$ Observability — Intent evidence

SLOs derived from config/policy (e.g., MTTR for a HA policy, path budget for QoS class).  
Dashboards organize by *intent* (policies/services), not only by components.

## All three — Continuous assurance

Nightly (or per-change) job: render config → apply → assert state → verify SLO/synthetic.  
Store pass/fail as an artifact next to the PR.

✓ This is the lens we should use to evaluate our monitoring.

# The Cost of Missing the Middle

When Config and State don't meet, money burns





# The Cost of Missing the Middle

When Config and State don't meet, money burns



*Outage cost = incidents × duration × blended \$/hr → 30 × (8 - 24) × \$1M = \$20-\$30M*



**How do you solve this?**

# Zoom-In: Config $\cap$ State (One Simple Example)

B  $\cap$  C = Config  $\cap$  State (Applied Config  $\neq$  Runtime State)

## Scenario :

**Customer channel:** Global Contact Centers on Black Friday

**Readiness Change:** Prioritize voice (EF) on WAN egress.

**Runtime reality (State):** Class-map didn't match; EF queue/counters stayed 0; DSCP marks stripped at edge.

**What we saw (Obs):** Wall of green. No voice synthetic. Tickets spike at 0800 CST.

## Fix pattern (+6 hrs):

### Post-deploy state assertions:

EF queue depth  $> 0$  during a test call

DSCP 46 on egress packets

Policy/class-map **hit counters**  $> 0$

**Gate the change:** Apply simple class map change to correct

**Add one synthetic:** 60-sec voice MOS probe across CC WAN Links

## Outcome:

Recurring “evening voice” incidents  $\rightarrow$  **reduced to zero**

Voice Synesthetics became standard monitoring deployments for CCs

Never happened again on Black Friday

*Config  $\cap$  State = Conformance.*

Config clean.  
State aligned.  
Dashboards green.

# Still failing? That's an interaction problem

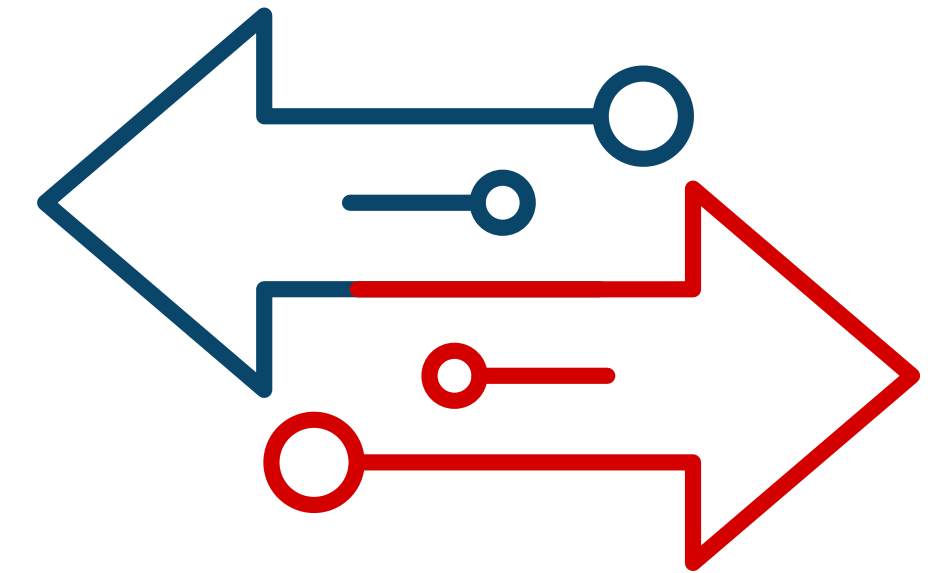
Components work; interactions fail



# Why systems fail when they get too clever

“Catastrophe requires multiple failures — single point failures are not enough.” — Richard Cook, *How Complex Systems Fail*

- ▶ Why is it that methods used to build small and medium sized systems do not work when applied to complex systems? — *Keepence & Mannion, ECBS '97*
- ▶ This is where config, state, and monitoring can all be “right”—and the system still fails.
- ▶ We need observability of interactions (dependencies, handoffs, timing), not just components.
- ▶ In complex systems, failures seldom reduce to a single component; they emerge from **unanticipated interactions** — *Keepence & Mannion, ECBS '97*



# Zabbix for Active Assurance

Not just watchful — extendable  
and verifiable

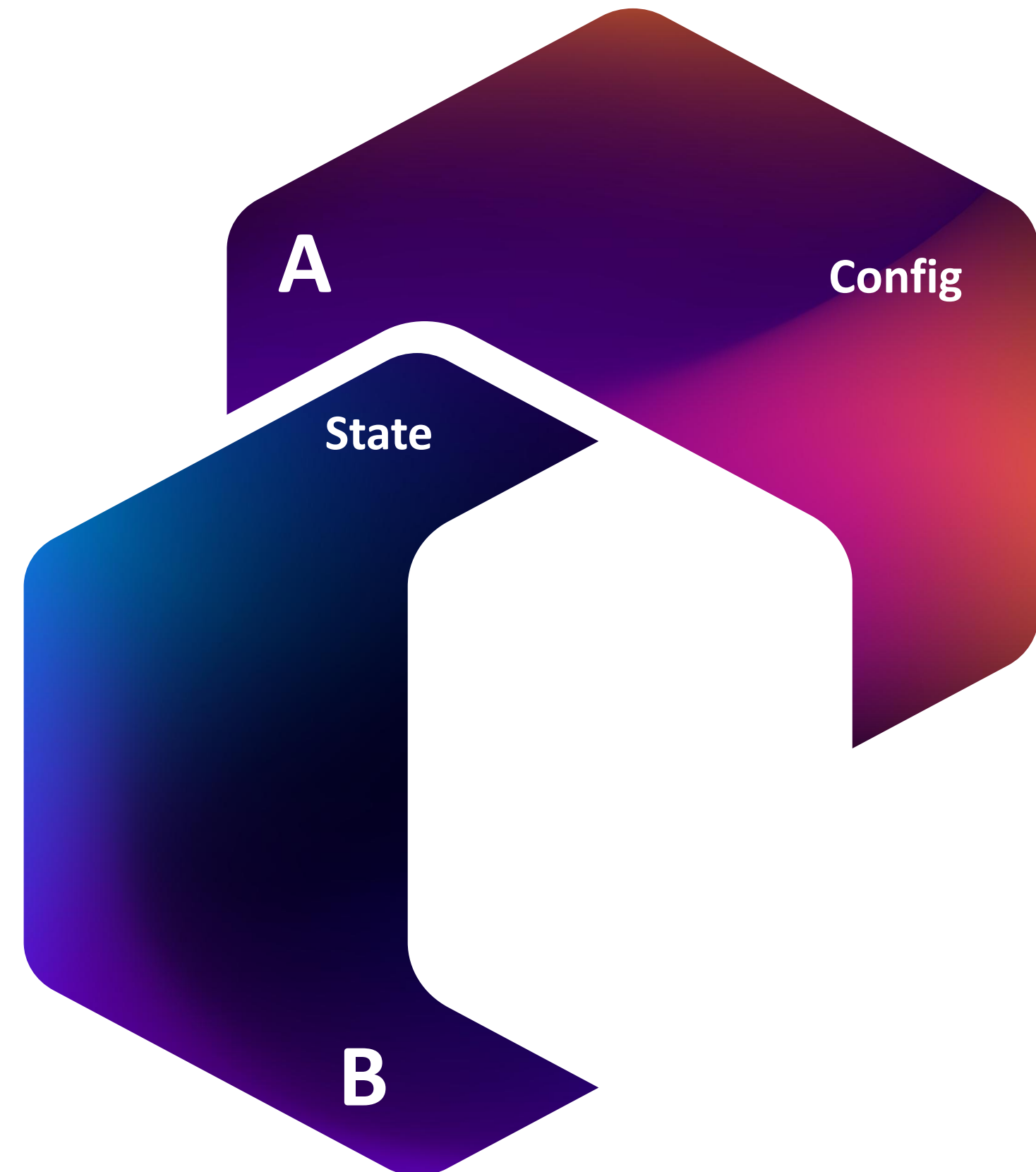


# Config Validation Example ( $A \cap B$ )

Config capture • Runtime state • Mismatch triggers

## ◆ Intent / Desired Config (A)

- ▶ Apache must be configured to serve traffic over HTTPS
- ▶ API endpoint /v1/status should respond with 200
- ▶ TLS 1.2 must be enforced and checked
- ▶ User Experience Checks

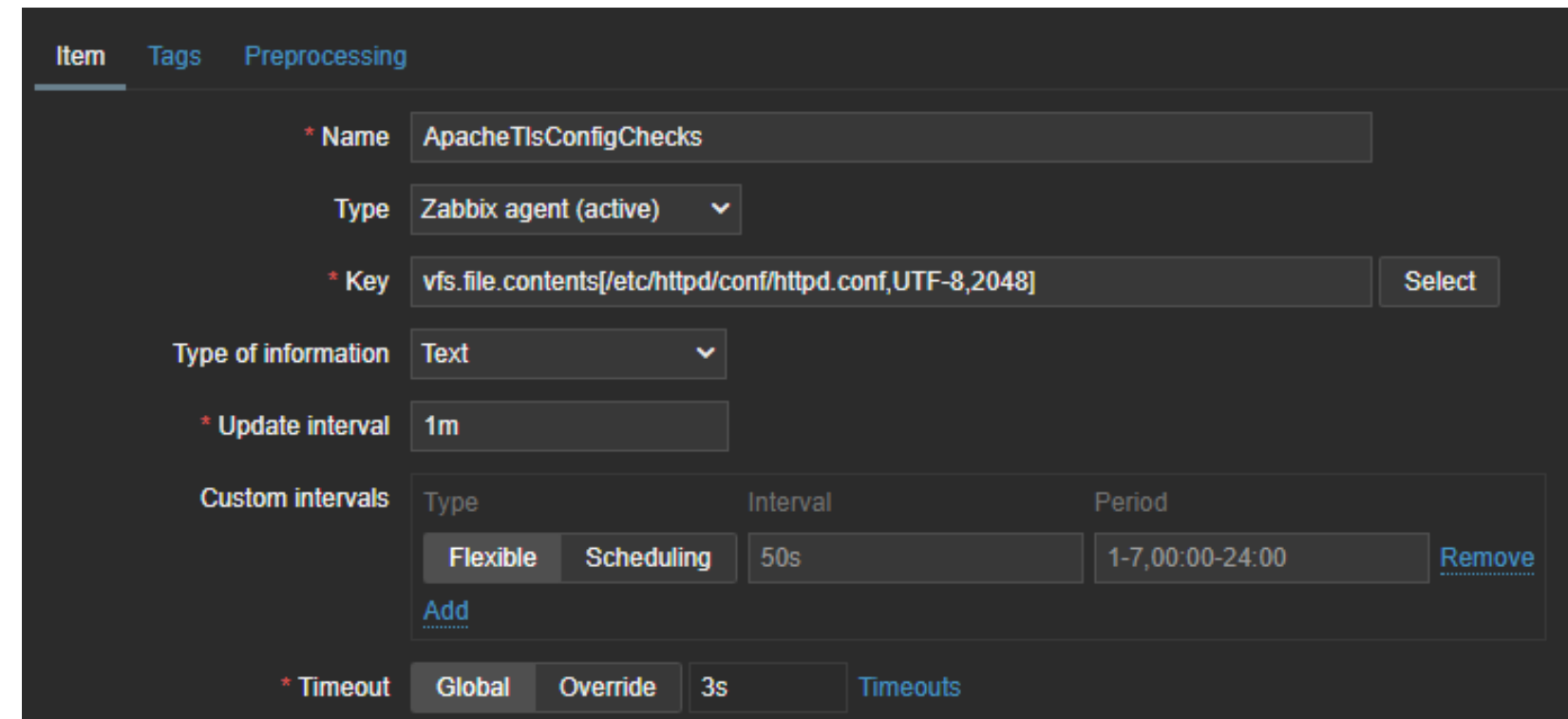


# Config Validation Example (A ∩ B)

Config capture • Runtime state • Mismatch triggers

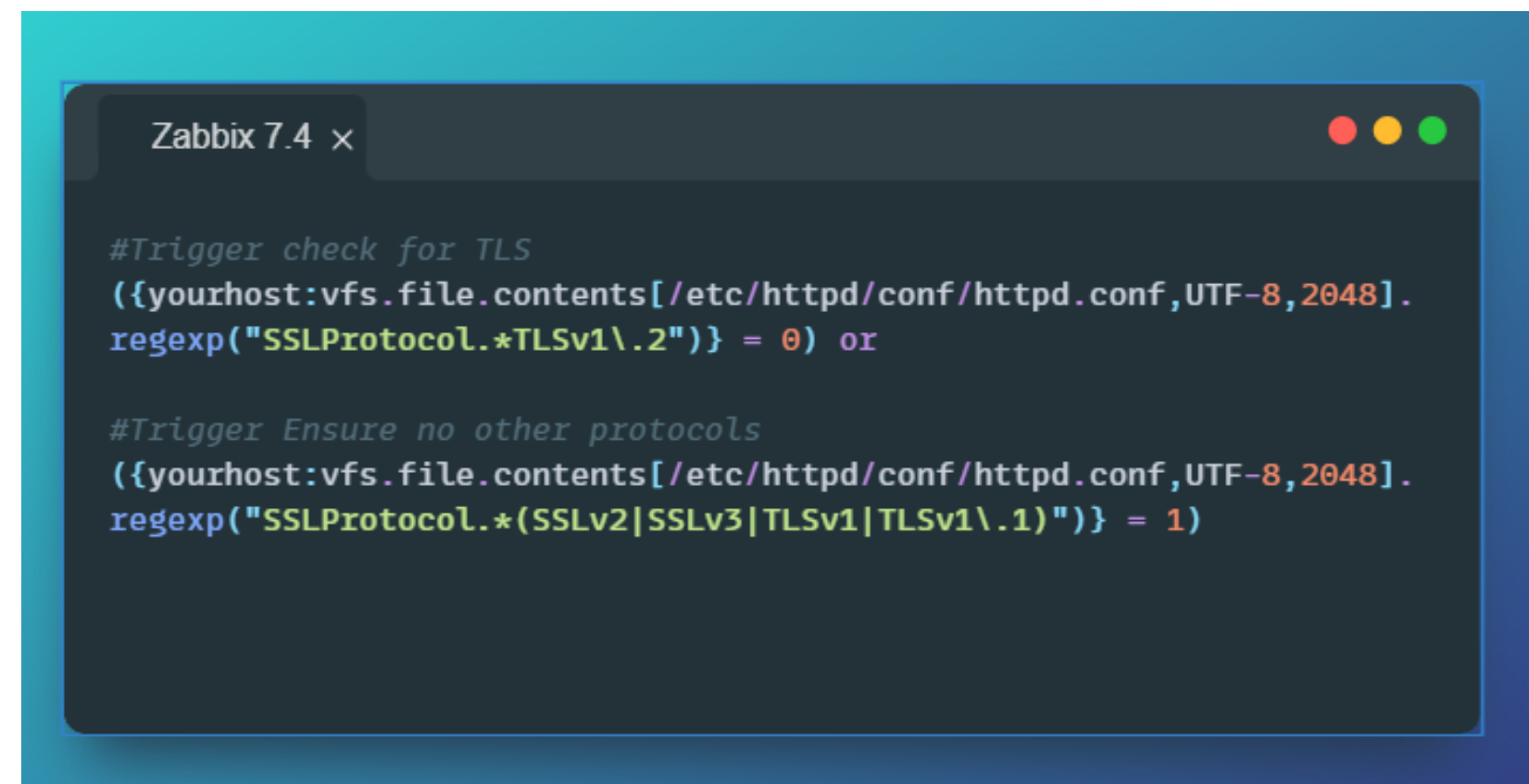
## ◆ Intent / Desired Config (A)

- ▶ Apache is “configured”
- ▶ TLS 1.2 is Configured
- ▶ Other Encryption protocols not configured



The image shows the Zabbix configuration form for an item named 'ApacheTlsConfigChecks'. The form is divided into tabs: 'Item', 'Tags', and 'Preprocessing'. The 'Item' tab is active. The configuration includes:

- Name:** ApacheTlsConfigChecks
- Type:** Zabbix agent (active)
- Key:** vfs.file.contents[/etc/httpd/conf/httpd.conf,UTF-8,2048]
- Type of information:** Text
- Update interval:** 1m
- Custom intervals:** A table with columns 'Type', 'Interval', and 'Period'. It contains one entry: 'Flexible' (Type), '50s' (Interval), and '1-7,00:00-24:00' (Period). There are 'Add' and 'Remove' buttons.
- Timeout:** Global, Override, 3s



The image shows a Zabbix 7.4 configuration window with a dark background. It contains the following configuration:

```
#Trigger check for TLS
({yourhost:vfs.file.contents[/etc/httpd/conf/httpd.conf,UTF-8,2048].
regexp("SSLProtocol.*TLSv1\2")} = 0) or

#Trigger Ensure no other protocols
({yourhost:vfs.file.contents[/etc/httpd/conf/httpd.conf,UTF-8,2048].
regexp("SSLProtocol.*(SSLv2|SSLv3|TLSv1|TLSv1\1)")} = 1)
```

# Config Validation Example (A $\cap$ B)

Config capture • Runtime state • Mismatch triggers

## ◆ State Checks (B)

- ▶ Web Scenario is Configured with triggers
- ▶ SSL checks disabled
- ▶ 200 status returned

The screenshot shows the 'Authentication' tab in the Zabbix configuration interface. The 'HTTP authentication' dropdown is set to 'None'. Below it, the 'SSL verify peer' and 'SSL verify host' checkboxes are both unchecked and are highlighted with a red rectangular box. Further down, there are input fields for 'SSL certificate file', 'SSL key file', and 'SSL key password'. At the bottom right, there are 'Add' and 'Cancel' buttons.

Scenario	Steps 1	Tags	Authentication
HTTP authentication: None			
SSL verify peer: <input type="checkbox"/>			
SSL verify host: <input type="checkbox"/>			
SSL certificate file: <input type="text"/>			
SSL key file: <input type="text"/>			
SSL key password: <input type="text"/>			
Add Cancel			



# Config Validation Example (A ∩ B)

## Config capture • Runtime state • Mismatch triggers

### ◆ Observed State (Zabbix Monitoring)

- ▶ TLS certificate is expired
- ▶ Port 443 is open, but API returns 200
- ▶ May see Latency spikes to 2 - 3 seconds under load.
- ▶ Web scenario does not fail



# Config Validation Example ( $A \cap B$ )

Config capture • Runtime state • Mismatch triggers

## ✦ Outcome

- ▶ NOC sees “Apache is active” → no immediate alert
- ▶ Users report timeouts or slowness
- ▶ TLS errors in browsers
- ▶ RCA takes several hours due to config-state blind spot

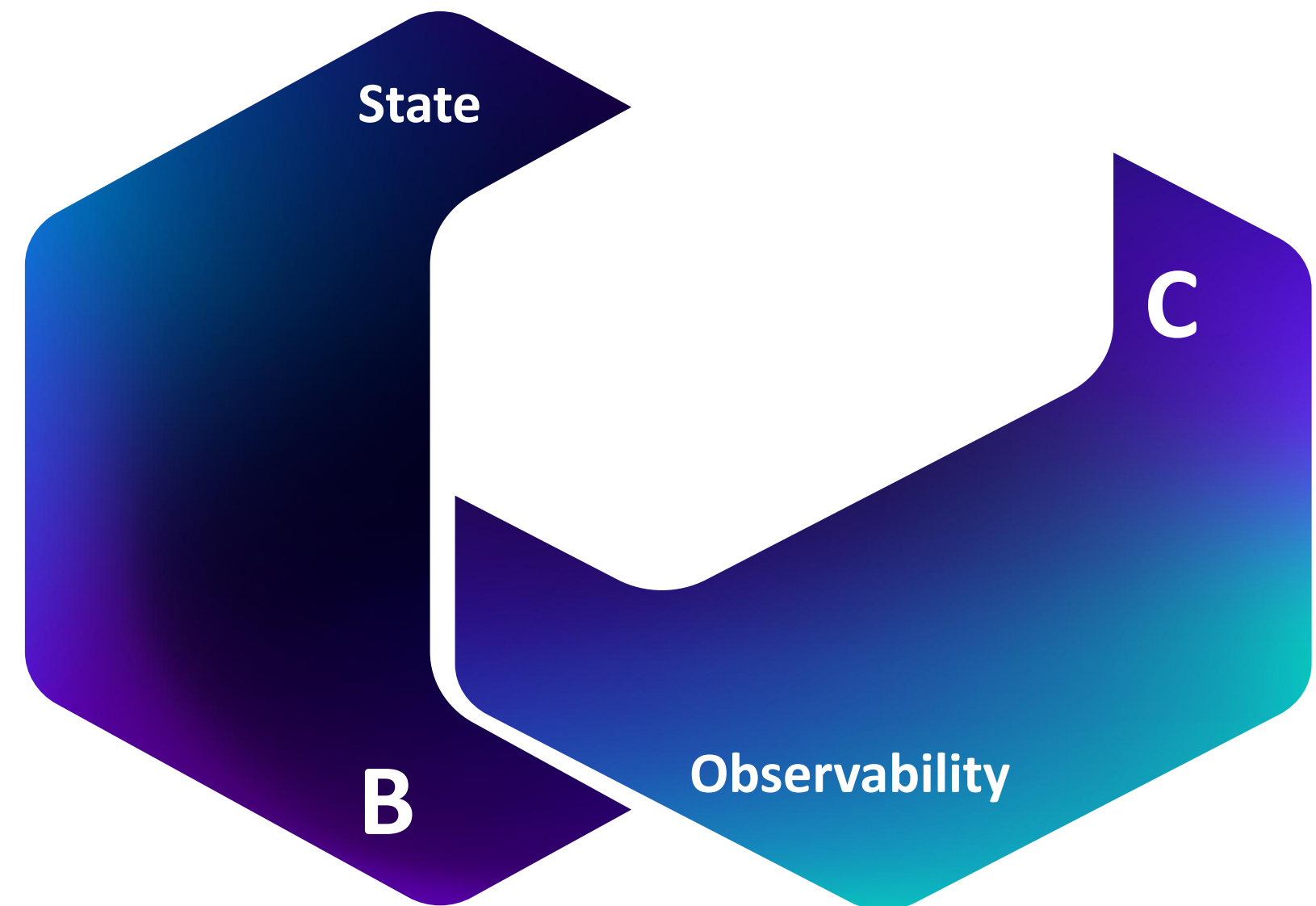
✅ **Lesson: Config is not proof. Config is intent. Only state tells the truth.**

# Runtime State Validation ( $B \cap C$ )

The DB is running — so why is everything slow?

## ◆ State/ Observability (B)

- ▶ PGSQL is up and running
- ▶ DB Is responding to queries
- ▶ User complain of app response times



# Runtime State Validation (B ∩ C)

The DB is running — so why is everything slow?

## ◆ State/ Observability (B)

- ▶ PostgreSQL installed
- ▶ Correct DB, schema, and tables present
- ▶ Connection string is configured in app
- ▶ Service marked “running”

\* Name

Type

\* Key

Type of information

Units

\* Update interval

Custom intervals

Type	Interval	Period	Action
<input checked="" type="checkbox"/> Flexible <input type="checkbox"/> Scheduling	<input type="text" value="50s"/>	<input type="text" value="1-7,00:00-24:00"/>	<a href="#">Remove</a>
<a href="#">Add</a>			

# Runtime State Validation (B $\cap$ C)

The DB is running — so why is everything slow?

## ◆ Observed State (C)

- ▶ This time we know something is wrong
- ▶ Users complain
- ▶ Investigation Begins

... Several Hours later



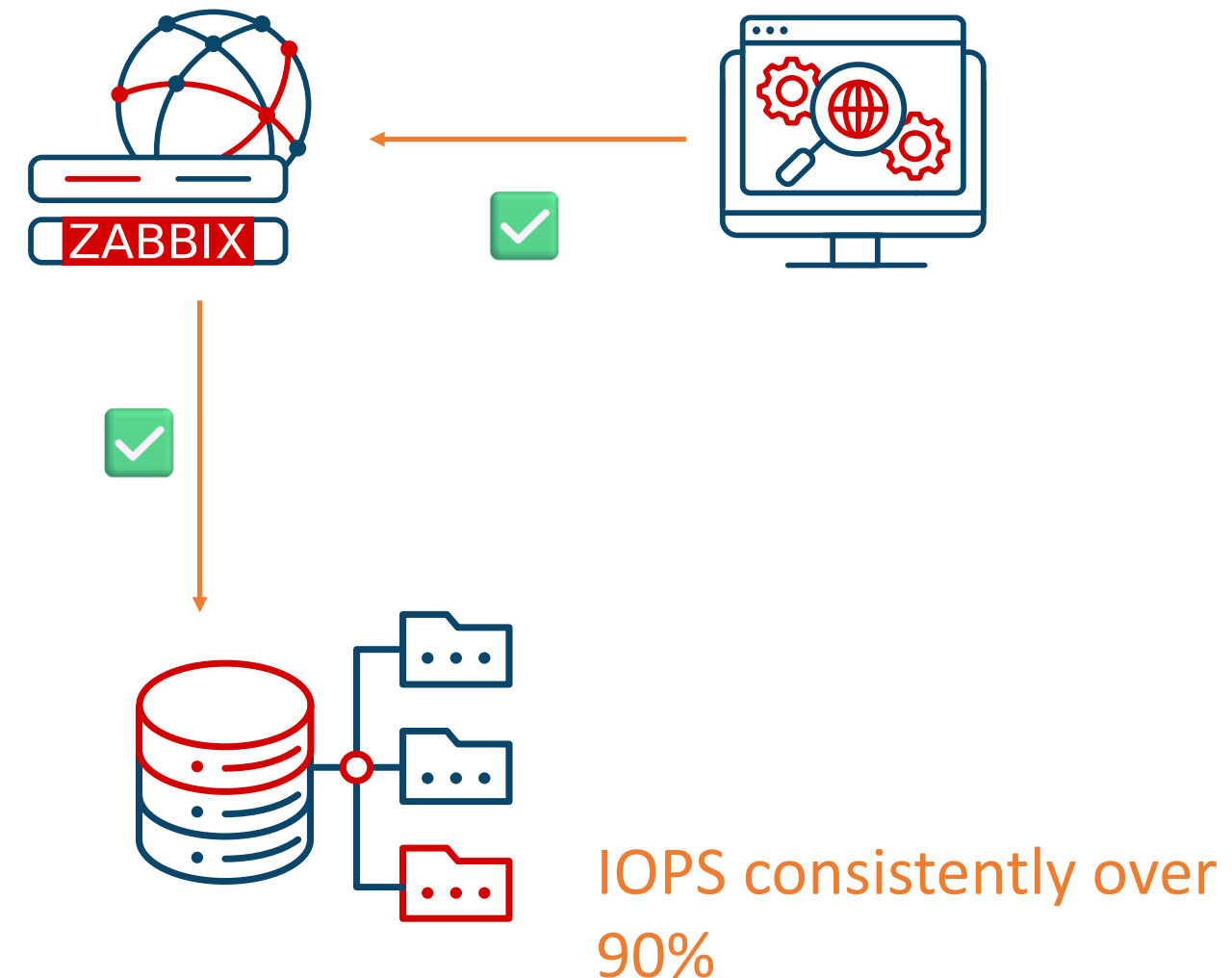
# Runtime State Validation (B $\cap$ C)

The DB is running — so why is everything slow?

## ✦ Outcome

- ▶ Alerts are missed (DB “running” = OK)
- ▶ App response times increase
- ▶ Users experience login timeouts

RCA points to **Failing Disk, High IOPS on remote NAS**



✓ **Lesson:** Running isn't the same as healthy. Healthy isn't the same as performant.

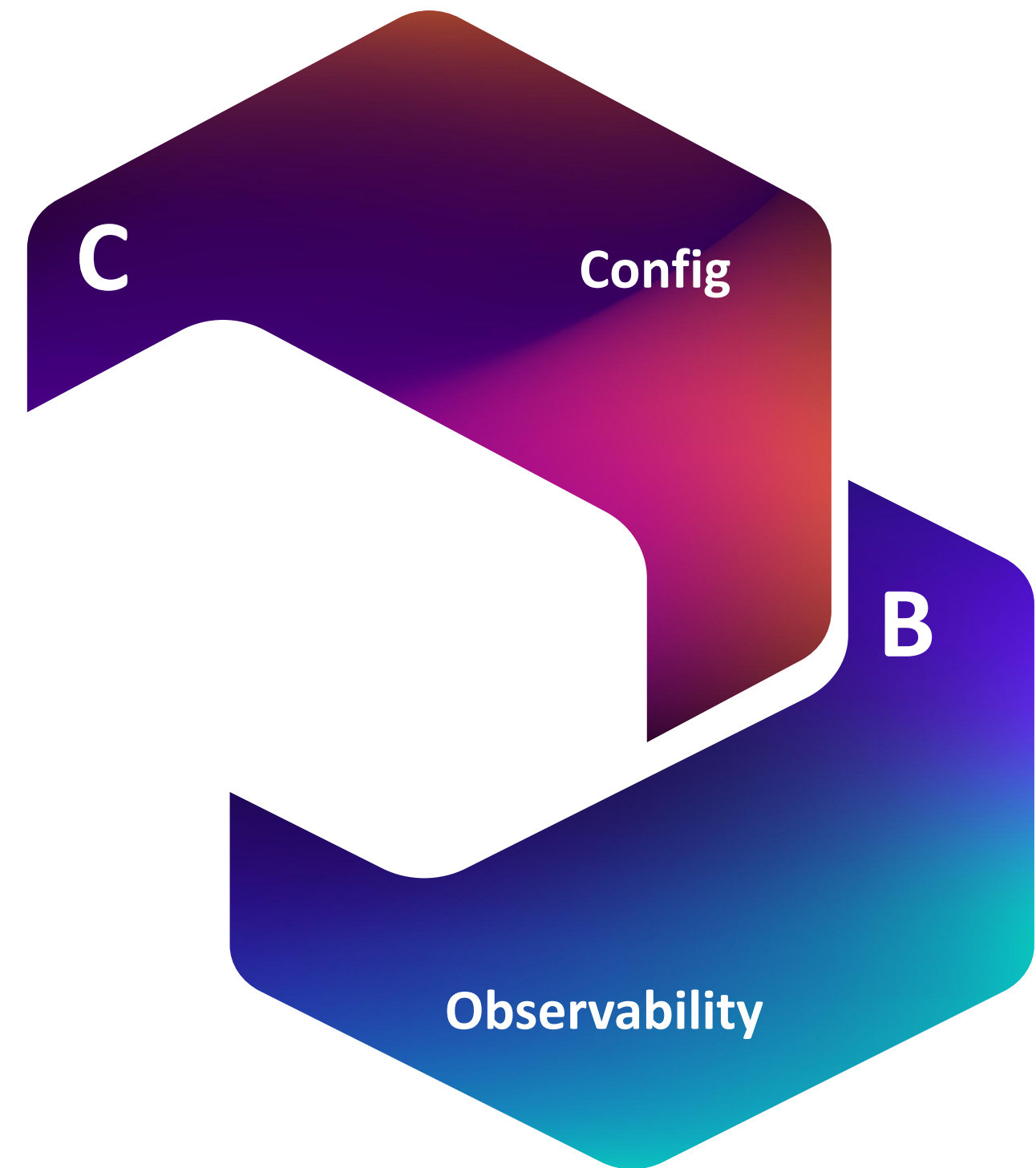


# Intent vs Observed State ( $A \cap C$ )

The routing config looks good — but packets don't lie.

## ◆ Config (A) Intent

- ▶ BGP peers defined
- ▶ AS Path & prefixes configured
- ▶ Route maps applied correctly
- ▶ Expected route to 192.168.1.0/24 via AS65001



# Intent vs Observed State ( $A \cap C$ )

## ◆ Config (A) Intent

- ▶ Configured correctly
- ▶ Peers are *UP*
- ▶ Routes exist



```
Zabbix 7.4

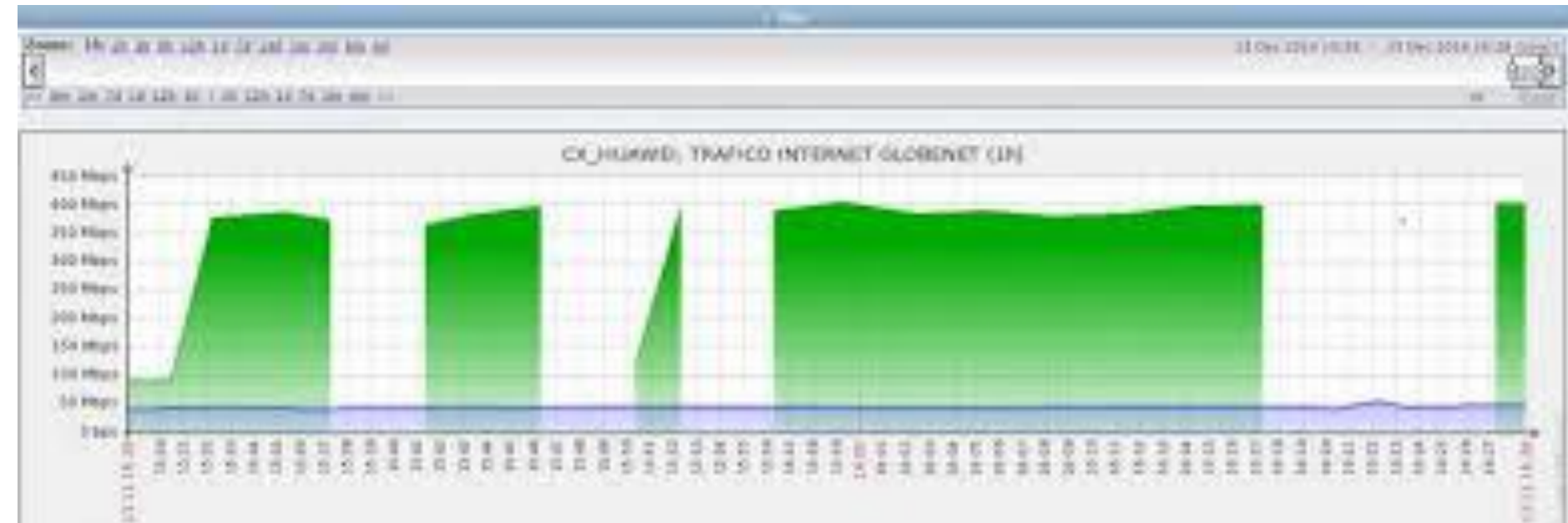
snmp.get[.1.3.6.1.2.1.15.3.1.7] ; BGP peer state
snmp.get[.1.3.6.1.2.1.4.21.1.1] ; routing table
```



# Intent vs Observed State (A n C)

## ◆ Observed State (C)

- ▶ Path latency to 192.0.2.10 = 450ms Frequently
- ▶ Unexpected AS in traceroute path
- ▶ FIB contains alternate next-hop
- ▶ Packets going through backup ISP



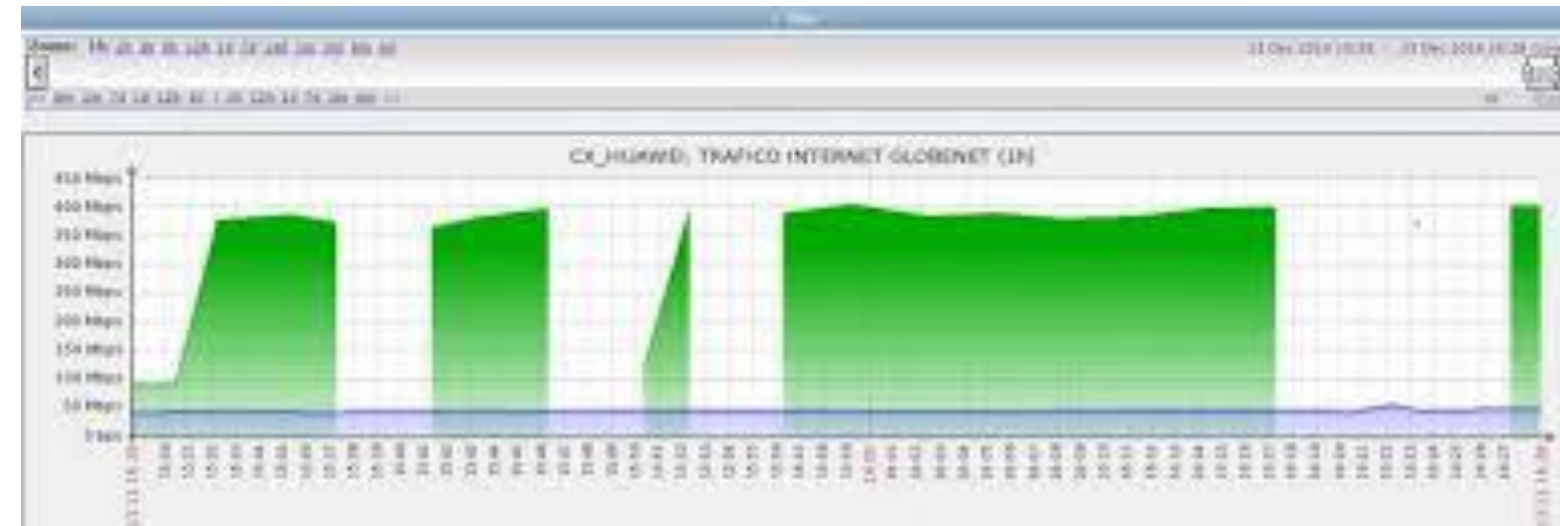
● Observed state ≠ intended path

- Failover path is active
- Traffic *diverged silently*

# Intent vs Observed State (A $\cap$ C)

## ✦ Outcome

- ▶ Network config “looks fine”
- ▶ State reports peers up
- ▶ But latency is degraded



No alerts **without intent-state** correlation

✓ **Lesson:** Routing is intent — but packets reveal the truth.

Expose silent failover, routing drift, and degraded user experience — even when everything looks green.

# From Monitoring to Validation

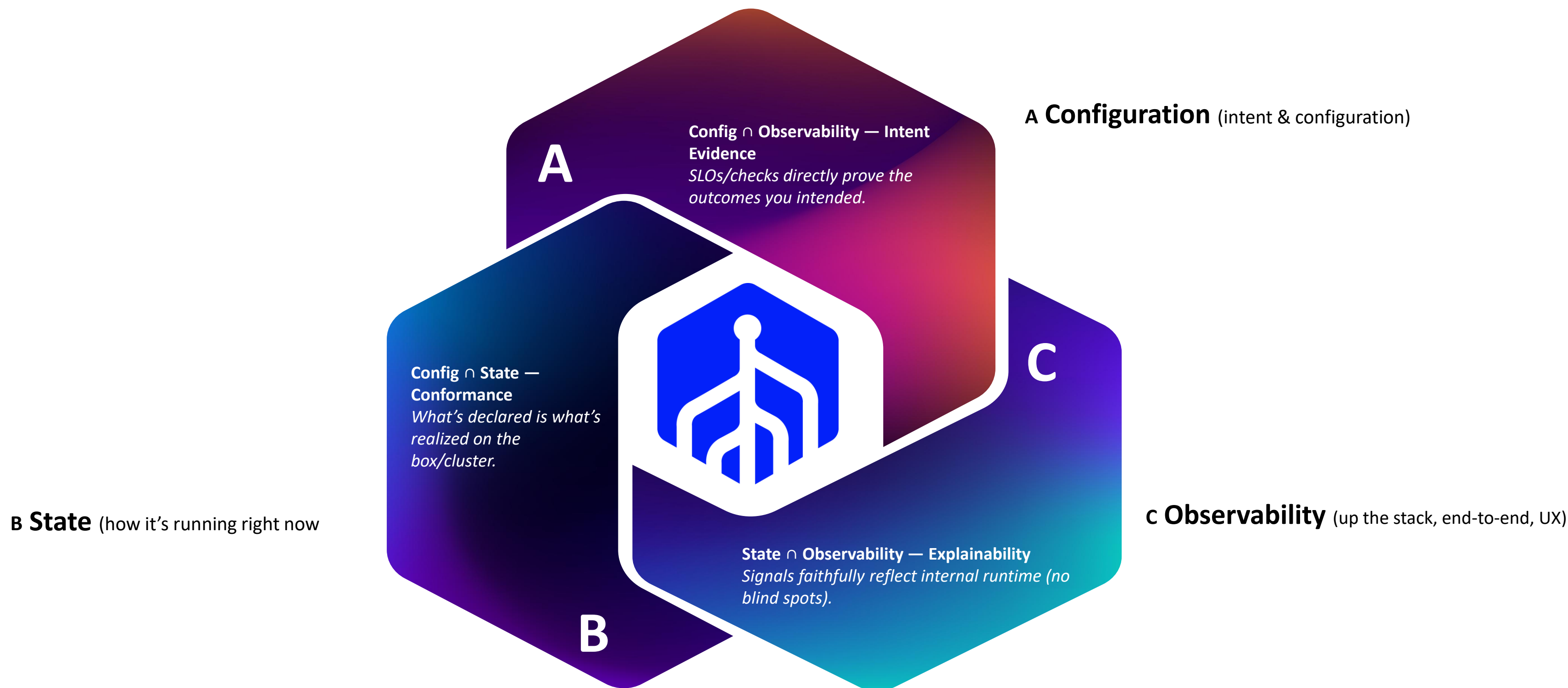
Config • Runtime state • Experience

- Zabbix is not just watchful — it's extendable and verifiable.



# Config $\cap$ State $\cap$ Observed

Where the gaps hide, money burns



# Full Circle: From Config to Customer Outcomes

Design it, run it, prove it — end to end.



# What good looks like

Outcomes of aligning Config • State • Observability

**70%**  
reduction outages

**↓ 1 Hr**  
MTTR down to less than 1 hr

**\$30M**  
AR costs avoided

**Zero Surprises**  
Config, State, Observability aligned

## Behaviour shifts

Firefighting → **Validation loop**

Component dashboards → **Intent checks**

Alerts → **Assertions + SLOs**

Ad hoc fixes → **Per-change state tests**

● This is what mature observability unlocks.

# Final Question: Why It Breaks at Scale

“Why don’t the methods that work for small/medium systems work in complex systems?” — Keepence & Mannion, ECBS '97

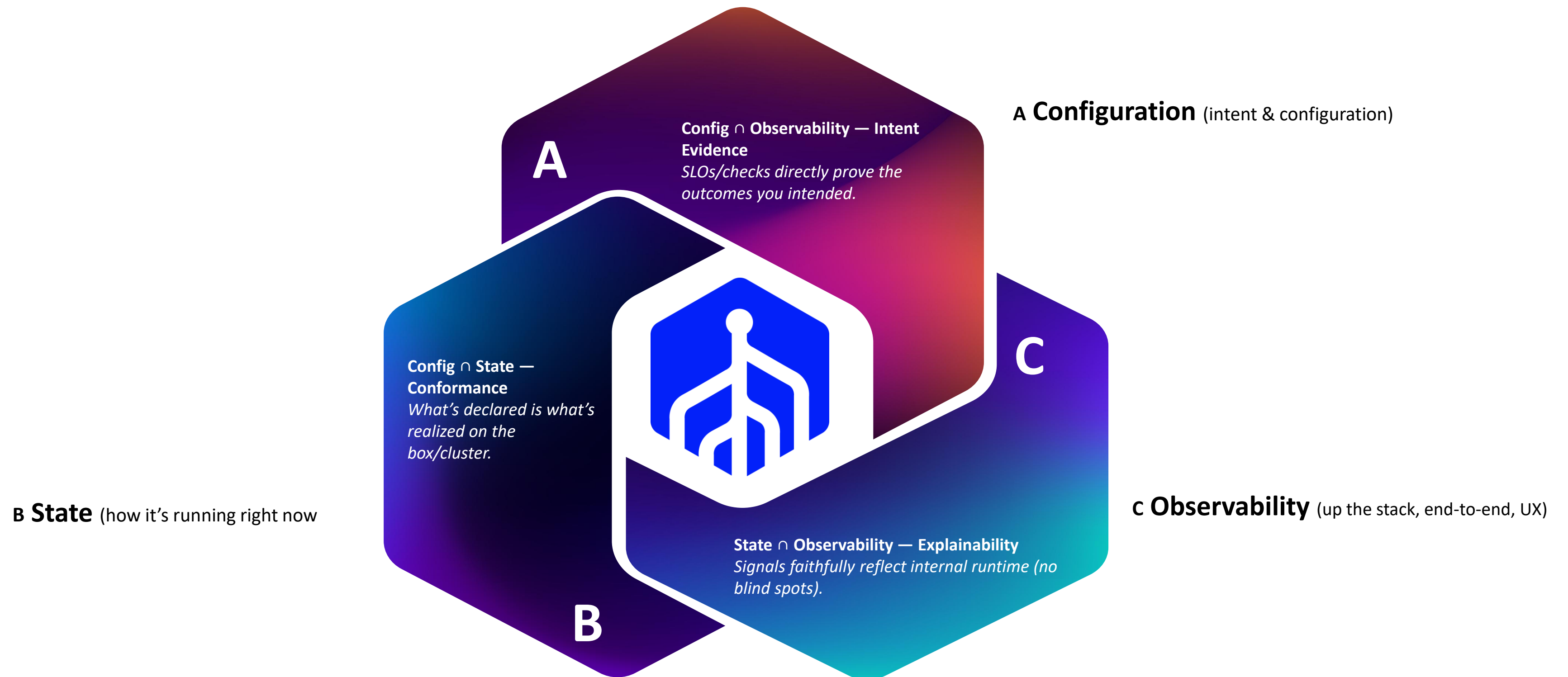
- **Interaction explosion:** more components  $\Rightarrow$  more unexpected behaviors at the joins.
- **Tight coupling & timing:** queues, retries, backoffs, elections—small delays amplify.
- **Partial visibility:** each tool sees a slice; no single source captures intent  $\rightarrow$  state  $\rightarrow$  UX.
- **Asymmetric ops:** config converges slower than state changes; observability lags outcomes.

● “In complex systems, failures rarely map to one component—they emerge from unanticipated interactions.” — Keepence & Mannion (1997)

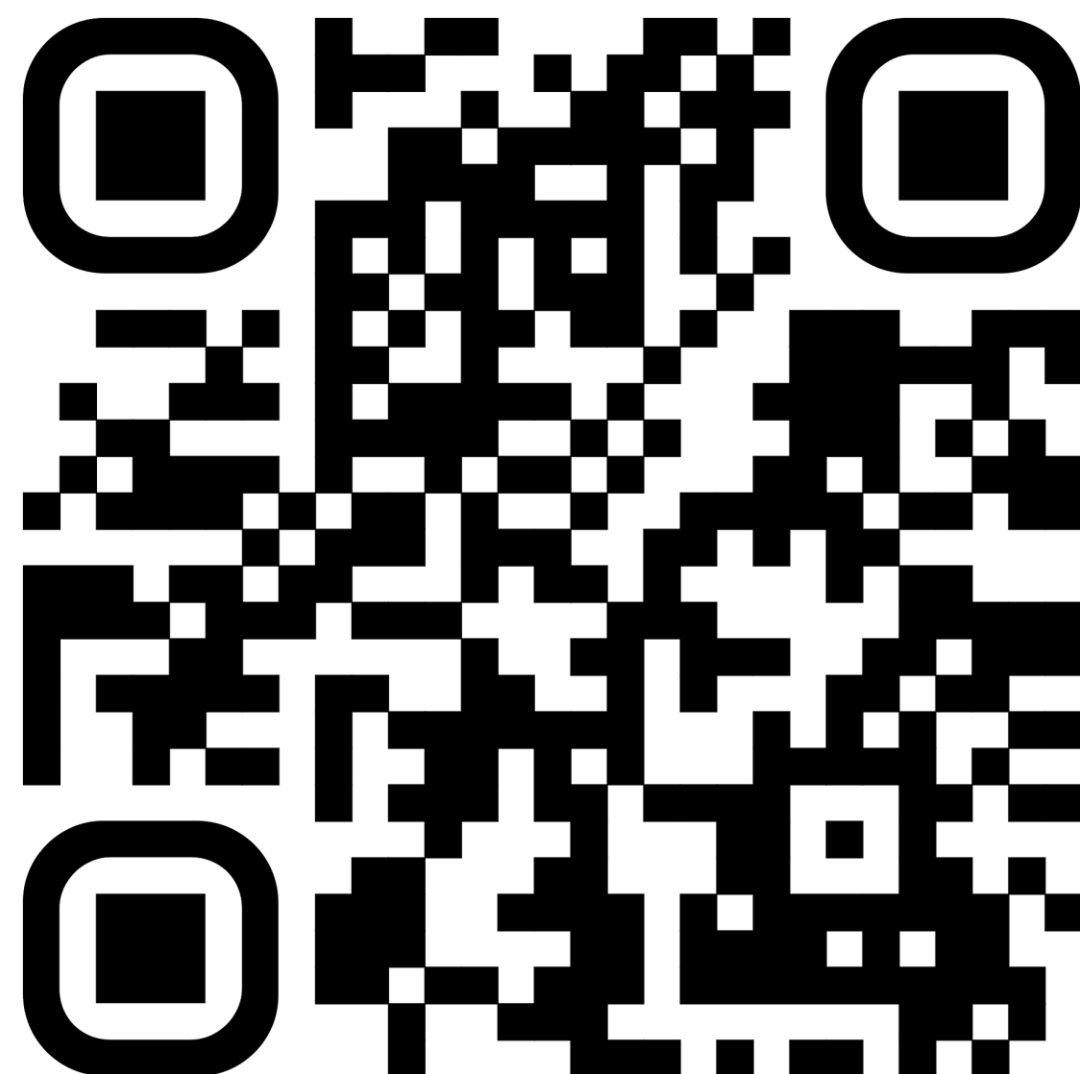


# Final Question: Why It Breaks at Scale

“Why don’t the methods that work for small/medium systems work in complex systems?” — Keepence & Mannion, ECBS '97







## Complex Systems

Keepence, et al. 97

[www.rconfig.com/complexsystems](http://www.rconfig.com/complexsystems)



OCTOBER 8 • 10, 2025  
RIGA • LATVIA

# Thank you



Stephen Stack - CTO



OCTOBER 8 • 10, 2025  
RIGA • LATVIA

# Thank you



Stephen Stack - CTO



OCTOBER 8 • 10, 2025  
RIGA • LATVIA