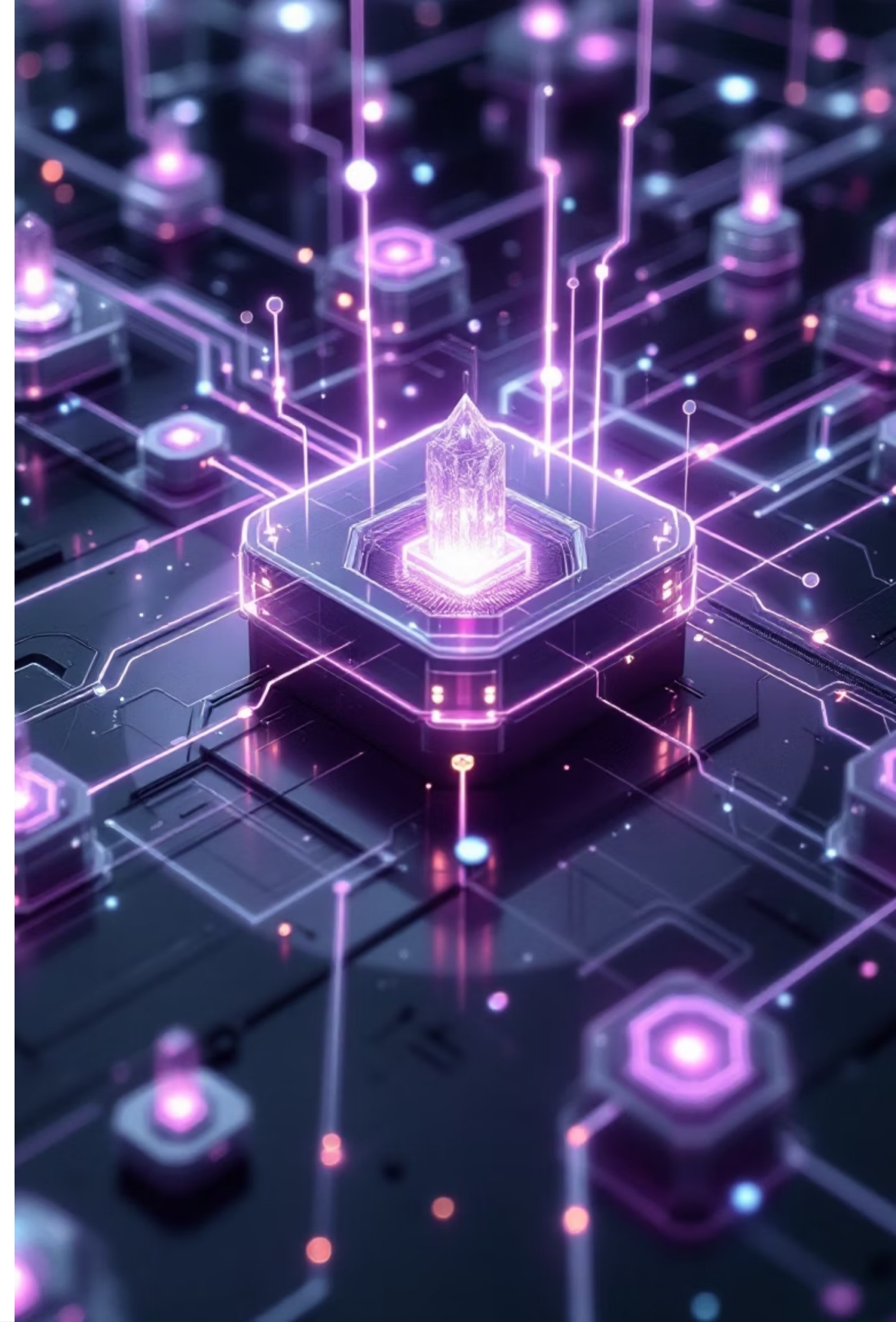


Taming the Chaos: Avoiding Kubernetes Landmines with Observability & AIOps

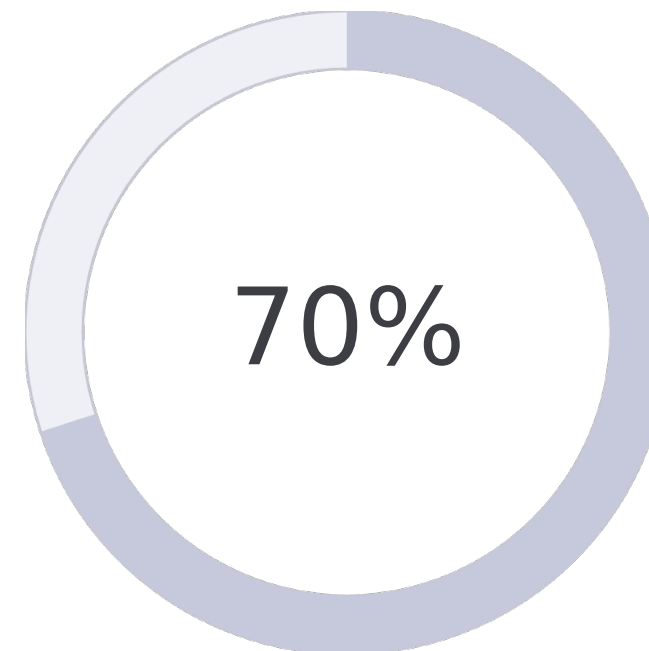
Kaustubha Shravan | Solution Architect | Microsoft
SREday Chennai 2025 Q4



The Promise vs. Reality of Kubernetes

"Scalable until it breaks."

Kubernetes promised infinite scale and seamless orchestration. But the reality? Most organizations face a steep learning curve filled with unexpected pitfalls.



Configuration Errors

Issues stem from misconfigurations and security mistakes

Most production incidents aren't platform failures—they're human errors in implementation and security oversights.



Why Outages Happen (and Keep Repeating)

Configuration Drift

Production gradually diverges from source control as teams apply manual hotfixes that never make it back to Git.

Human Error in YAML

Complex manifests with nested structures create countless opportunities for typos, indentation mistakes, and incorrect values.

Lack of Validation

Changes get deployed without proper policy enforcement or automated testing—only to fail in production.

Culture: Firefighting Over Prevention

Teams spend time reacting to incidents rather than building guardrails and automation to prevent them.

Security Disasters Waiting to Happen

Common Vulnerabilities

-  **Containers Running as Root**
Privileged access creates massive blast radius for exploits
-  **Vulnerable Base Images**
Unpatched dependencies expose critical CVEs
-  **Over-Permissive RBAC**
Excessive permissions violate least privilege principles

Security Fixes



PodSecurityAdmission

Enforce security standards at the cluster level



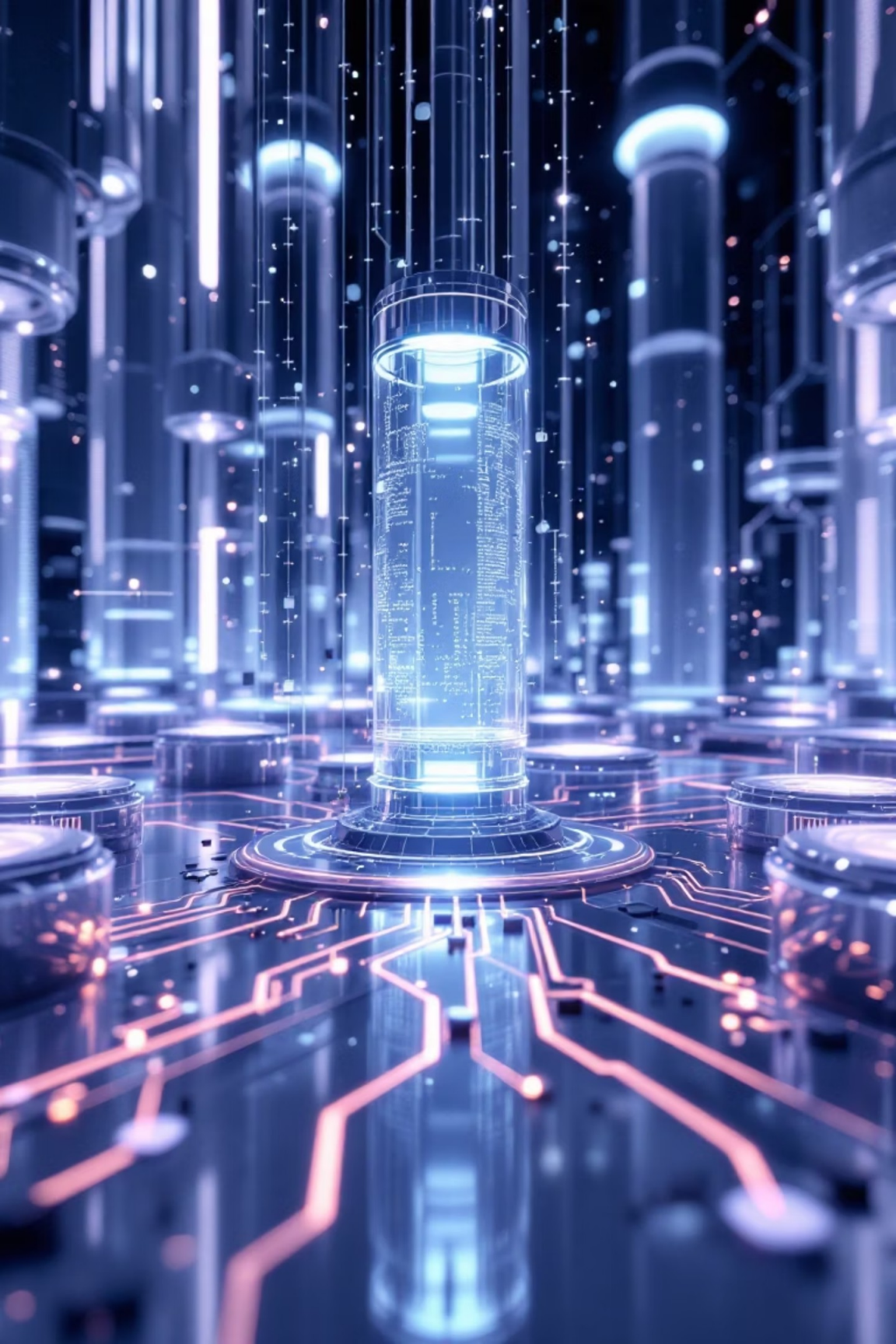
Trivy, Falco, Kyverno

Scan images and monitor runtime behavior



Least Privilege

Grant only necessary permissions



Configuration Catastrophes

The Problems

- **Misaligned requests/limits:** Pods get OOM-killed or waste resources due to incorrect resource specifications
- **Copy-paste manifests:** Teams duplicate configurations without understanding context, propagating errors
- **No policies:** Missing guardrails allow invalid or dangerous configurations into production

The Solutions

01

Admission Controllers

Validate and mutate objects before persistence

02

GitOps Pipelines

Declarative config with version control and automated deployment

03

KubeLinter

Static analysis to catch misconfigurations early

Observability Blind Spots



Limited Metrics

CPU and memory utilization tell you *what* happened, not *why* it happened or what to do next



Missing Runtime Visibility

Without kernel-level insights, critical application behavior remains invisible



Logging Noise

Excessive unstructured logs bury important signals in irrelevant data

The Path Forward

OpenTelemetry + Loki

Unified telemetry collection with fast log aggregation and querying capabilities

eBPF Observability

Kernel-level visibility into network flows, syscalls, and resource usage without overhead

Grafana + SLOs

Dashboards directly tied to service-level objectives for actionable insights



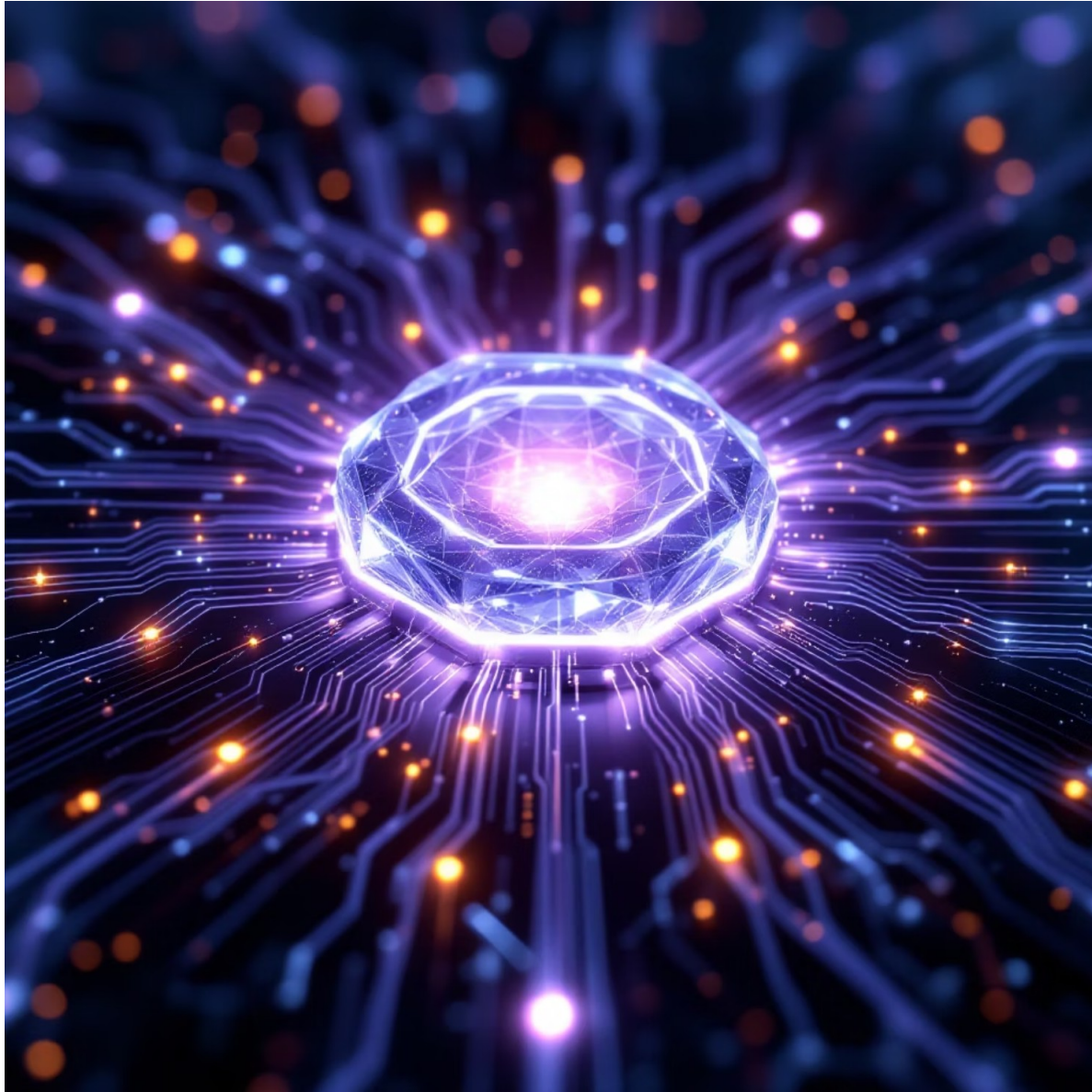
Scaling Traps That Crush Performance

HPA Thrashing Horizontal Pod Autoscaler oscillates wildly between scaling up and down, creating instability instead of stability	Uneven Node Scheduling Pods cluster on specific nodes while others sit idle, leading to resource hotspots and wasted capacity	Reactive Lag Scaling decisions based on current load arrive too late, causing degraded user experience during traffic spikes
--	---	--

Smart Scaling Strategies

Vertical Pod Autoscaler Right-size resource requests based on actual usage patterns	Predictive Scaling Leverage historical data to scale ahead of anticipated demand	Custom Metrics Scale on business metrics like queue depth or request latency
---	--	--

AIOps to the Rescue



Machine learning transforms reactive incident response into proactive problem prevention by analyzing patterns humans can't detect.



Anomaly Detection

ML models identify deviations from normal behavior



Auto-Triage Patterns

Correlate signals to prioritize critical issues



Root Cause Suggestion

Surface likely causes based on historical incidents



Pipeline: Metrics → Correlation → Action | AIOps closes the loop from detection to remediation

The Hardening Checklist



These three pillars work together—security prevents breaches, reliability ensures uptime, and observability provides the visibility to maintain both.

Automate Prevention, Not Just Response



Continuous Compliance

Automatically enforce policies with every deployment, catching violations before they reach production



Drift Detection

Monitor and alert when live state diverges from declared configuration in Git



Chaos Experiments

Proactively test resilience by injecting failures in controlled environments

Essential Tools

ArgoCD

GitOps continuous delivery

Policy Controller

Enforce organizational standards

LitmusChaos

Chaos engineering platform

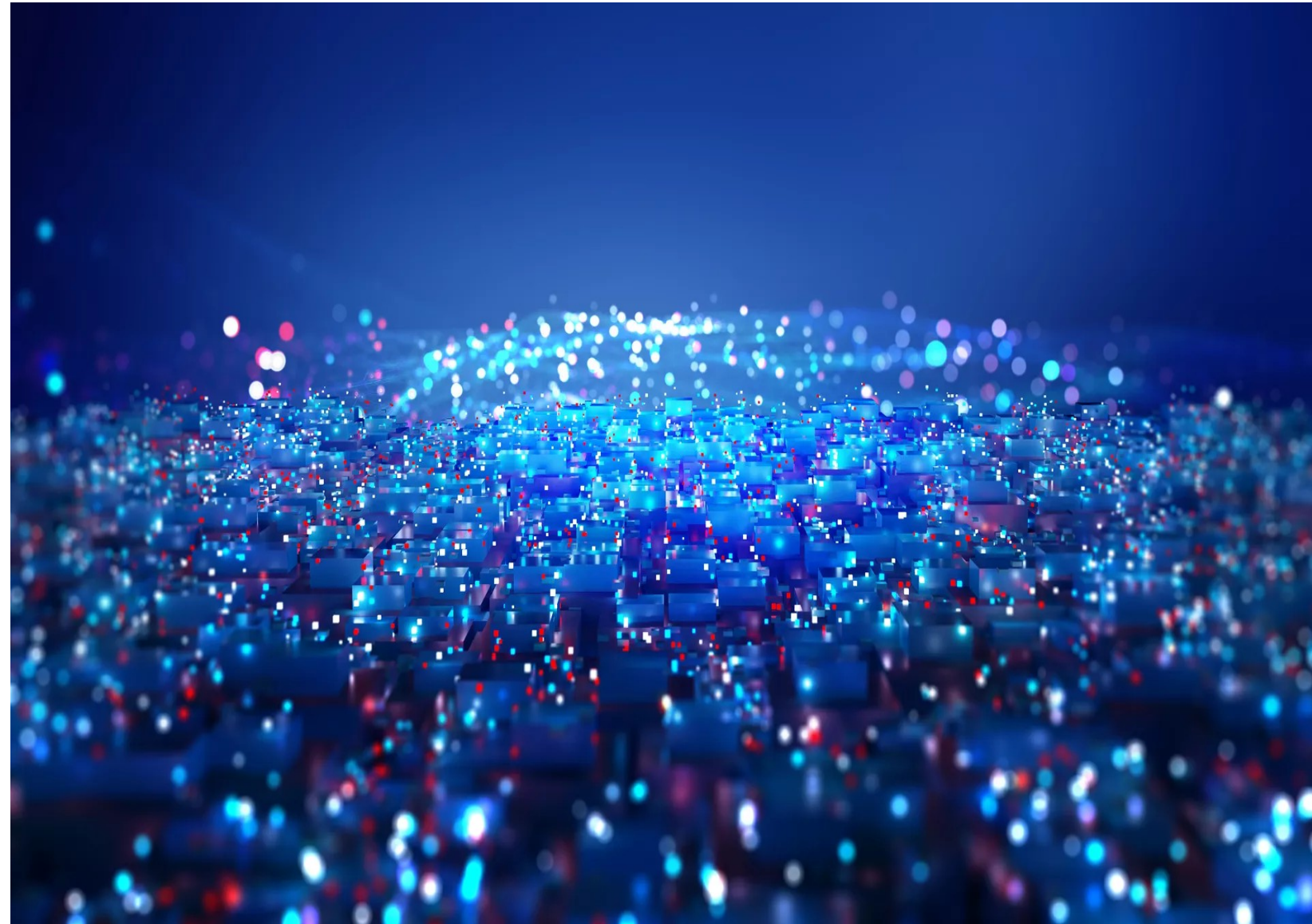
The goal isn't perfection—it's building systems that fail gracefully and recover automatically.

Lessons from the Field

- Reliability = instrumentation
- Postmortem loops → improvement
- ML-driven anomaly detection ↓ MTTR 40%

Closing Thoughts

- Reliability = Security × Observability × Culture
- Make failures routine before disasters
- Shift: reactive → proactive engineering





Q&A

Connect with me



[linkedin.com/in/
kaustubhashravan](https://www.linkedin.com/in/kaustubhashravan)



[kaustubhashravan@gmail.co
m](mailto:kaustubhashravan@gmail.com)