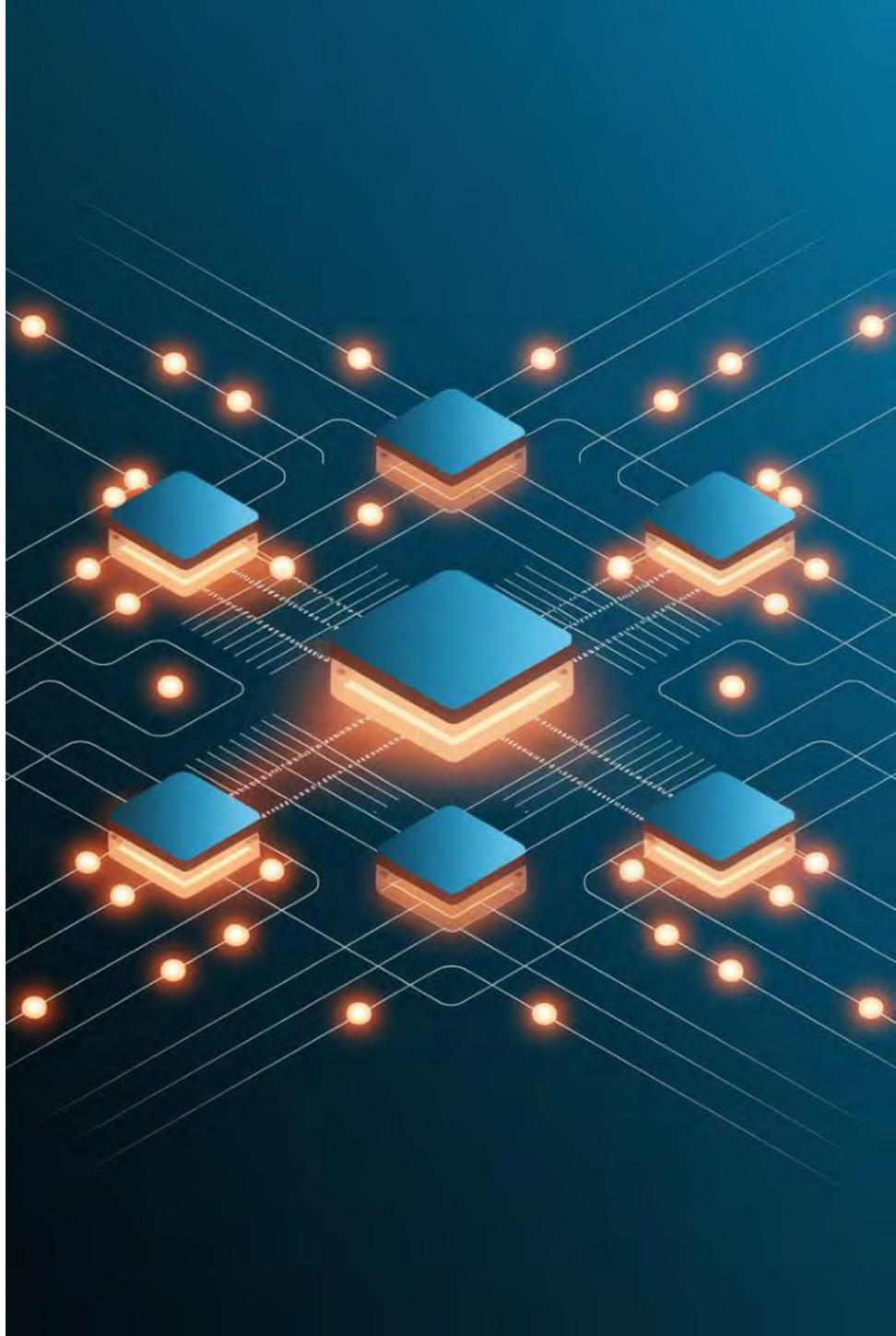
Enhancing Operational Efficiency with Observability

Discover how observability transforms complex distributed systems by providing deep insights into application behavior, system health, and user experience. This presentation reveals practical strategies for implementing observability to dramatically improve reliability, accelerate troubleshooting, and optimize performance across your entire technology stack.

By: Hardik Patel



Hardik Patel

Director of Engineering - Data Platforms

- 20+ years in tech industry (small startups to large enterprises)
- Masters in Software Engineering from BITS, Pilani
- Held various roles from SWE to TL to Architect to Manager to Director
- Technical Leadership at high-tech places Yahoo!, Symantec, Groupon, PayPal
- Led varied sizes of teams from 5 to 30+ engineers, leads & managers
- 10+ years of designing, developing Distributed Systems

The Challenges of Distributed Systems

Increased Complexity

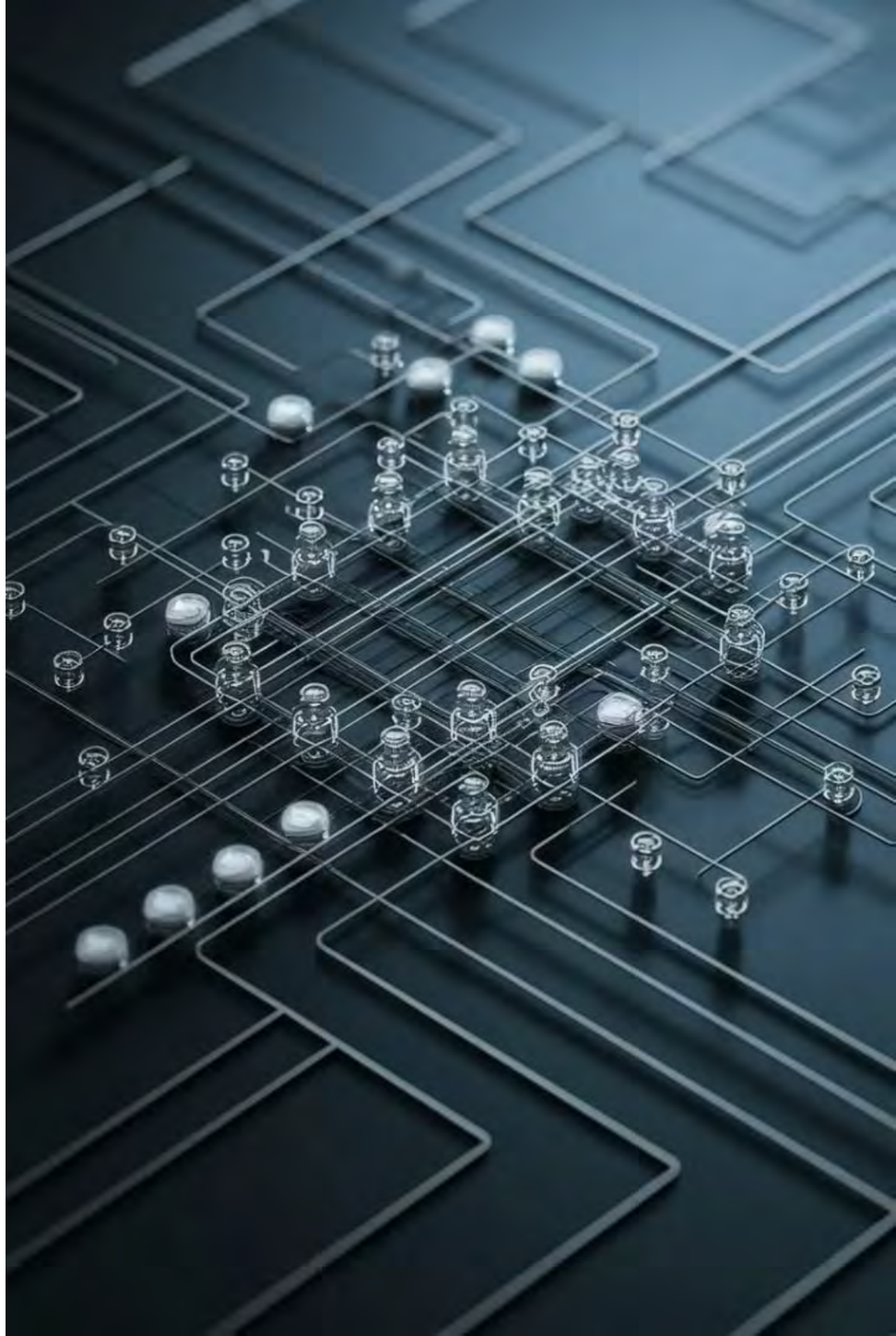
Modern architectures contain hundreds of interconnected services. Traditional monitoring falls short.

Rapid Change

Continuous deployment creates constantly evolving systems. Pinpointing issues becomes harder.

Scale Issues

Problems may only appear at scale. Test environments can't replicate production production loads.



The Three Pillars of Observability

Metrics

Numerical data points collected at regular regular intervals, revealing system performance performance patterns and health. Enables Enables identification of trends, anomalies, and anomalies, and potential bottlenecks before before they become critical.

Logs

Timestamped records of discrete events and and actions within a system. Provides rich contextual information for debugging, forensic forensic analysis, and understanding the sequence of incidents.

Traces

End-to-end journey tracking of requests as they as they flow through distributed services. Illuminates complex interactions, latency issues, and dependencies that impact overall overall system performance.

Metrics That Matter

System Latency

Measure response times across service boundaries. Track p95 and p99 percentiles. percentiles.

Error Rates

Monitor failed requests and exceptions. Set thresholds based on business impact. impact.

Resource Utilization

Track CPU, memory, and network usage. Prevent resource exhaustion before it before it occurs.

User Experience

Measure real user metrics. Connect backend performance to frontend experiences. experiences.

Distributed Tracing in Action

Frontend Request

User interaction initiates a traced traced HTTP request with unique unique correlation ID

Gateway Service

Propagates trace context while while routing to downstream microservices

Database Query

Span indicates performance bottleneck with 200ms latency latency exceeding threshold

Response Chain

Completed trace shows full request path with timing metrics metrics across services

Business Impact of Observability

60%

Faster Troubleshooting

Reduction in time to identify root causes

25%

Improved MTTR

Faster mean time to recovery after incidents

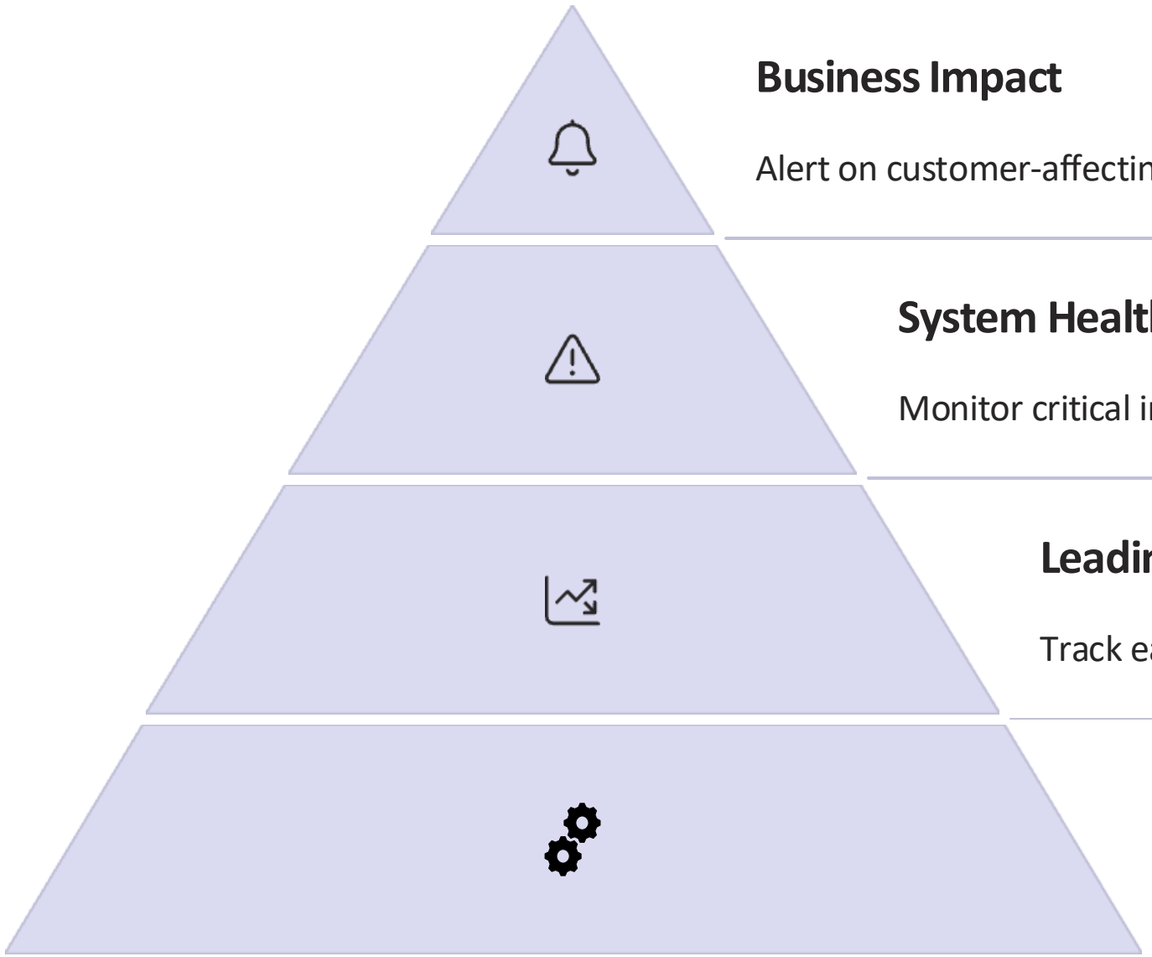
30%

Increased Uptime

More reliable systems with fewer outages

outages

Implementing Effective Alerting



Business Impact

Alert on customer-affecting issues

System Health

Monitor critical infrastructure components

Leading Indicators

Track early warning signs

Resource Utilization

Watch capacity trends

Observability Implementation Roadmap

Define Key Metrics

Identify the critical signals that indicate system health. Focus on business outcomes. outcomes.

Instrument Code

Add metrics, logging, and tracing to applications. Use consistent naming conventions.

Centralize Data

Implement collection and storage solutions. Enable correlation across services. services.

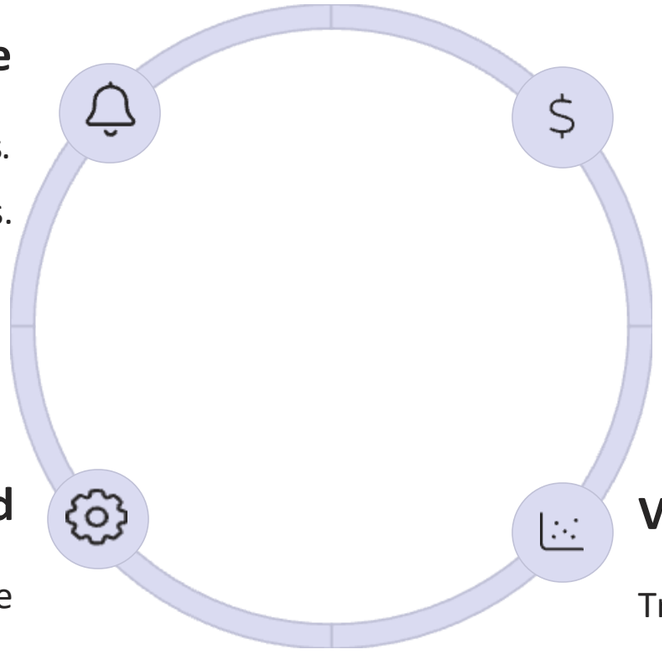
Visualize & Alert

Create dashboards and alerting rules. Eliminate noise and false positives. positives.

Common Pitfalls to Avoid

Alert Fatigue

Too many notifications lead to ignored warnings. warnings. Focus on actionable alerts.



Cost Overruns

Collecting everything can be expensive. Sample Sample high-volume telemetry.

Tool Overload

Using too many different systems. Consolidate Consolidate where possible.

Vanity Metrics

Tracking numbers that don't matter. Tie metrics metrics to business outcomes.

Key Takeaways

Observability is essential

Not optional for modern distributed systems

Integrate all three pillars

Metrics, logs, and traces work together

Measure business impact

Connect technical metrics to business outcomes

Build observability culture

Train teams to leverage these insights

Thank You!