# Autonomous Cloud Infrastructure with Go Building Self-Healing Systems

SUDHAKAR PALLAPROLU

INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, SHIBPUR, INDIA

Conf42 Golang 2026

THE PROBLEM

# The Problem: Unplanned Downtime Is Costly


Enterprises face significant revenue loss every minute infrastructure goes dark. Unplanned IT downtime remains one of the most expensive operational failures in cloud-native environments.

---

- **Manual Response Is Too Slow**

  Human-in-the-loop remediation cannot match the pace of distributed system failures.

- **Complexity Is Accelerating**

  Go-based microservices spread failure across dozens of interdependent services before detection occurs.

- **The Gap Demands Autonomy**

  Self-healing systems close the window between incident detection and automated remediation.

SESSION ROADMAP

# What We'll Cover Today

01

## The Problem

Why traditional ops models break under distributed, cloud-native load

02

## AIOps Framework

How ML and predictive analytics automate incident lifecycle management

03

## Architecture Layers

Observability, telemetry, and the self-healing control plane in Go

04

## Policy-Driven Orchestration

From simple provisioning to closed-loop automation

05

## Platform Tooling & Takeaways

Azure Automanage, AWS FIS, GCP Active Assist — and what to build next

# The Rise of the Autonomous Enterprise

AIOps adoption has accelerated rapidly, enabling engineering teams to move beyond reactive operations. The autonomous enterprise is defined by infrastructure that detects, diagnoses, and repairs itself  without waiting for human intervention.

### Detect

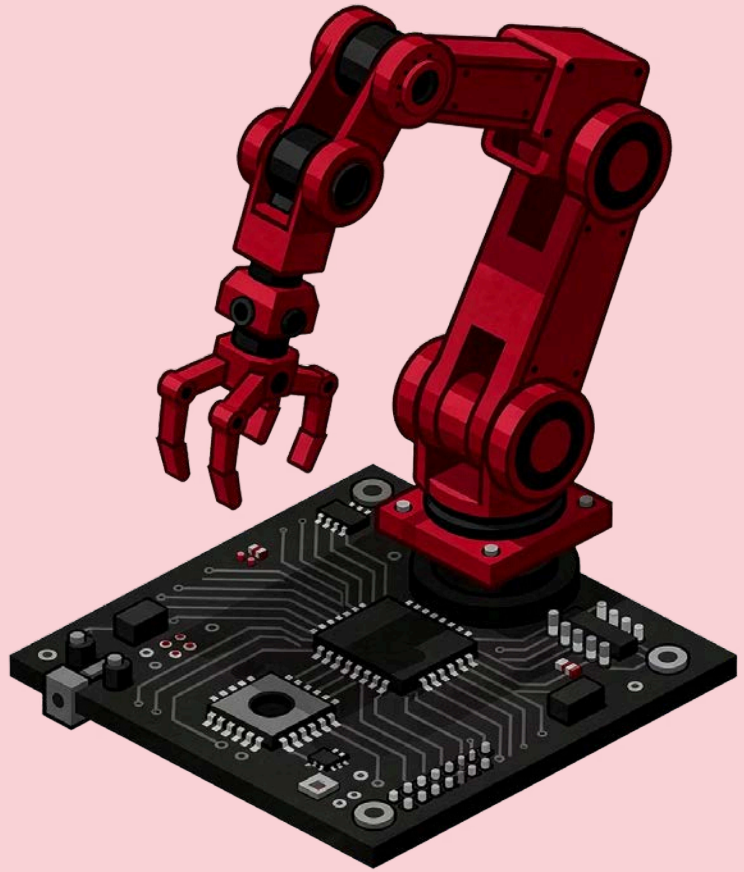
Real-time anomaly detection across logs, metrics, and traces

### Diagnose

ML-driven root-cause analysis across distributed Go services

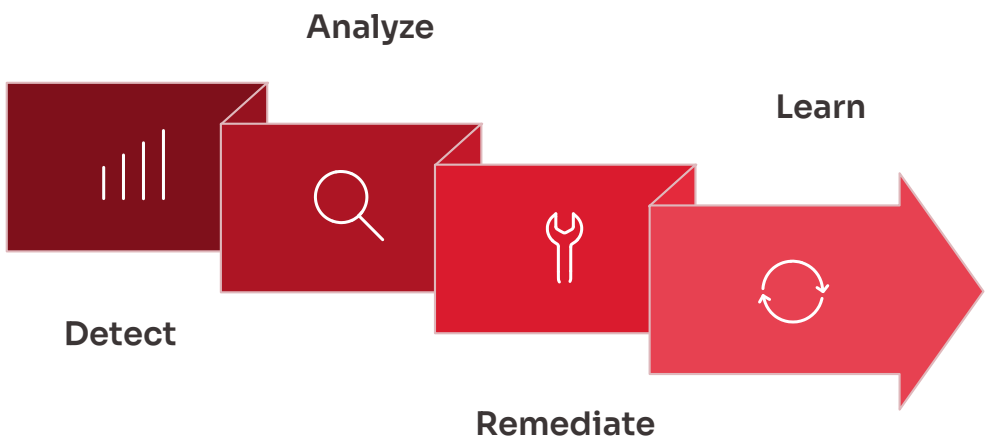
### Remediate

Automated, policy-gated responses that restore service without human escalation



AIOPS

# Automating the Incident Lifecycle




Go's concurrency model goroutines, channels, and context propagation makes it particularly well-suited for building low-latency AIOps pipelines that process high-volume telemetry streams in real time.

OBSERVABILITY

# Observability: The Foundation of Self-Healing

Effective self-healing requires full-spectrum observability. Go services instrumented with OpenTelemetry emit structured signals that feed AI models continuously.

### Structured Logs

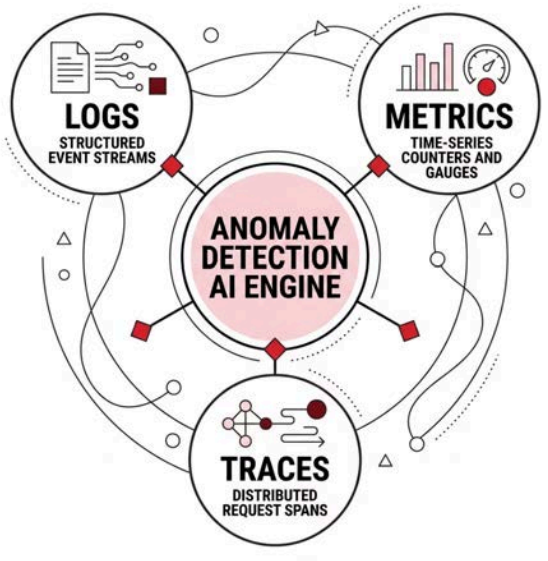
Machine-parseable event streams for correlation at scale

### Time-Series Metrics

Latency, error rates, and saturation signals drive alerting thresholds

### Distributed Traces

End-to-end request context across Go microservice boundaries



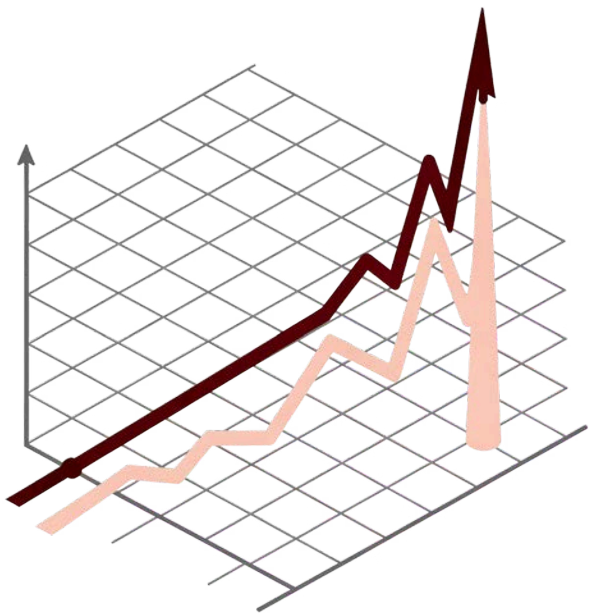
PREDICTIVE RELIABILITY

# Preventing Failures Before They Happen

## Predictive Failure Prevention

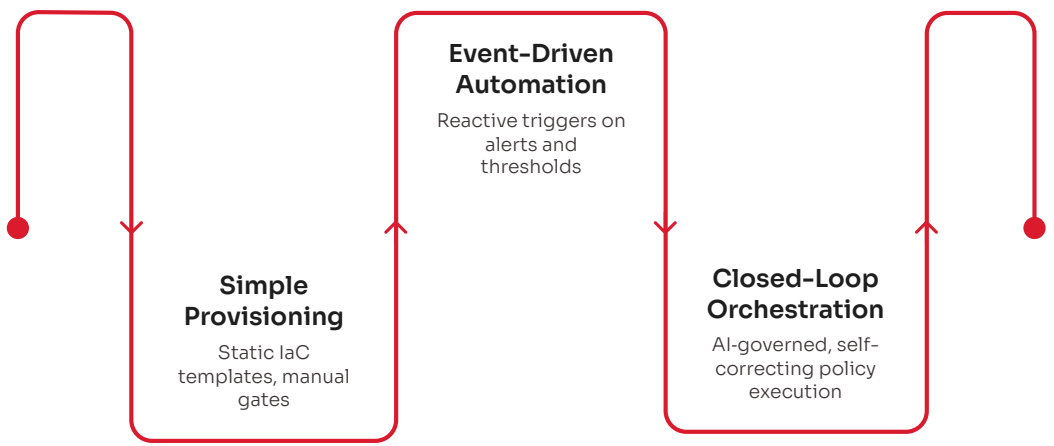
Rather than reacting to outages, predictive models analyze leading indicators memory pressure, connection pool saturation, disk I/O trends and trigger preemptive scaling or restarts before user-visible impact occurs.

- Train models on historical incident data from Go service telemetry
- Use sliding window aggregations to surface degradation trends early
- Trigger preemptive actions: pod restarts, traffic shifting, or capacity scaling
- Continuously retrain models as service behavior evolves



# Policy-Driven Orchestration

The orchestration model has evolved from simple resource provisioning to fully closed-loop automation where systems self-govern based on declared intent rather than manual runbooks.




Go's strong typing and interface-driven design make it an ideal language for building policy engines that evaluate operational context at runtime and dispatch corrective actions deterministically.

PLATFORM TOOLING

# Cloud Platforms Enabling Autonomous Operations

## Azure Automanage

Automates VM best practices, patch management, and configuration drift remediation across Azure workloads

## AWS Fault Injection Simulator

Validates resilience by injecting controlled failures into production-grade environments, surfacing hidden weaknesses

## GCP Active Assist

Delivers ML-powered recommendations for cost optimization, security posture, and operational efficiency

# Intelligent Infrastructure-as-Code

The next phase of IaC moves beyond static templates. Intelligent configurations dynamically adapt to operational context scaling policies adjust to traffic patterns, security rules respond to threat signals, and network topology reconfigures around faults.

## Context-Aware Configuration

IaC modules consume live operational signals to modify their own execution behavior

## Drift Detection and Auto-Correction

Continuous reconciliation loops written in Go enforce declared state, reverting unauthorized changes

## Policy as Code

Guardrails embedded in the provisioning pipeline prevent non-compliant deployments from reaching production

# Key Takeaways

- **Build for Observability First**

  Self-healing only works when telemetry is complete, structured, and machine-readable from day one

- **Close the Loop with Policy**

  Move from reactive alerting to closed-loop orchestration let Go policy engines govern remediation

- **Predict, Don't Just React**

  Apply ML to leading indicators so infrastructure acts before failures surface to users

- **Let Engineers Focus on Innovation**

  Autonomous systems absorb operational toil, freeing teams to build rather than firefight

# Thank you !!