IT Operations Management

A Complete Guide to Modern ITOM: Monitoring, Discovery, Orchestration & Cloud Operations

Course Study Guide | *Desqcon Learning*

Contents

1. **Foundations of IT Operations Management**
    - What is ITOM? Scope, Objectives, and Business Value
    - ITOM vs ITSM vs ITAM: How They Fit Together
2. **Monitoring & Event Management**
    - IT Infrastructure Monitoring Fundamentals
    - Event Management: Detection, Correlation, and Noise Reduction
    - Alerting, Thresholds, and On-Call Practices
3. **Discovery & Service Mapping**
    - IT Discovery: Finding and Classifying Assets
    - Service Mapping and Dependency Visualization
4. **Orchestration & Automation**
    - Runbook Automation and Self-Healing
    - Orchestration Across Hybrid & Cloud Environments
5. **Cloud Operations & Performance**
    - Cloud Operations Management (CloudOps)
    - Capacity Planning & Performance Optimization
    - ITOM Metrics, KPIs, and Maturity Model

Section 1: Foundations of IT Operations Management

What is ITOM? Scope, Objectives, and Business Value

IT Operations Management (ITOM) is the set of practices, processes, and tools used to provision, monitor, and manage the technology infrastructure and services an organization depends on — servers, networks, storage, cloud resources, and the applications running on top of them. ITOM ensures that IT infrastructure runs reliably, efficiently, and securely, day in and day out.

While IT Service Management (ITSM) focuses on delivering services to end users through defined processes (incident, problem, change management), ITOM focuses on the underlying infrastructure and systems that make those services possible. ITOM is the engine room; ITSM is the front office.

The Scope of ITOM

Infrastructure Monitoring Event & Incident Response Discovery & Service Mapping Orchestration & Automation Cloud & Hybrid Operations Capacity & Performance Reliable, Efficient, Secure Infrastructure

Core Objectives

- **Availability:** Keep systems and services up and running to agreed service levels.
- **Performance:** Ensure infrastructure delivers acceptable response times under real-world load.
- **Efficiency:** Use compute, storage, and network resources cost-effectively, avoiding waste.
- **Visibility:** Maintain an accurate, real-time understanding of what infrastructure exists and how it behaves.
- **Risk reduction:** Prevent outages, security incidents, and compliance failures through proactive management.

Business Value of Mature ITOM

Capability	Business Impact
Proactive monitoring	Fewer customer-facing outages, protecting revenue and reputation
Automation	Lower operating costs and faster response to incidents
Capacity planning	Right-sized infrastructure spend, avoiding both outages and waste
Service mapping	Faster root cause analysis and reduced downtime cost

Key Takeaways

- ITOM manages the infrastructure and systems layer that IT services depend on.
- Its scope spans monitoring, incident response, discovery, orchestration, cloud operations, and capacity management.
- Core objectives are availability, performance, efficiency, visibility, and risk reduction.
- Mature ITOM directly protects revenue, controls cost, and improves organizational resilience.

ITOM vs. ITSM vs. ITAM: How They Fit Together

These three disciplines are frequently confused because they overlap and share tooling, but each answers a different question. Understanding the distinction — and how they integrate — is foundational to working effectively across IT operations.

Three Disciplines, Three Questions

ITOM Runs infrastructure ITSM Delivers services ITAM Tracks assets

Discipline	Core Question	Example Activity
ITOM	Is the infrastructure running well?	Monitoring server CPU, responding to a network outage
ITSM	Is the service being delivered to users well?	Resolving a help desk ticket, managing a change request
ITAM	What do we own, and is it compliant and cost-effective?	Tracking software license counts, planning hardware refresh

How They Work Together

In practice, these disciplines are deeply interconnected:

- ITOM's **discovery** data feeds ITAM's asset inventory, keeping license and hardware records accurate.
- ITOM's **monitoring** generates events that become ITSM **incidents** when user-facing impact occurs.
- ITSM's **change management** process governs and approves changes that ITOM teams execute on infrastructure.
- ITAM's **asset lifecycle** data informs ITOM's capacity planning — knowing what hardware is aging or under warranty.
- All three share a common data foundation: the **Configuration Management Database (CMDB)**, which records assets, their relationships, and their configurations.

Why the Distinction Matters

Organizations that blur these disciplines often end up with fragmented tooling, duplicate data, and unclear ownership — a server might be tracked in three different systems with three different owners. Clear boundaries, supported by an integrated CMDB and shared discovery data, let teams collaborate without duplicating effort.

Key Takeaways

- ITOM keeps infrastructure running; ITSM delivers services to users; ITAM tracks and governs what the organization owns.
- The three disciplines share data and processes — especially through discovery and the CMDB.
- Clear ownership boundaries prevent fragmented tooling and duplicated effort.
- Mature organizations integrate all three into a single operational data foundation.

Section 2: Monitoring & Event Management

IT Infrastructure Monitoring Fundamentals

Monitoring is the foundation of ITOM — you cannot manage what you cannot see. Infrastructure monitoring continuously collects data about the health, performance, and availability of servers, networks, storage, applications, and cloud resources, turning raw telemetry into actionable insight.

The Three Pillars of Observability

Metrics Numeric time-series data CPU %, latency, error rate “What is happening?” Logs Timestamped event records Errors, transactions, audit trails “Why did it happen?” Traces Request paths across services Distributed transaction flow “Where did it happen?”

What to Monitor

Layer	Example Signals
Infrastructure	CPU, memory, disk I/O, network throughput
Platform	Database connections, queue depth, container health
Application	Response time, error rate, throughput (requests/sec)
Business	Transactions completed, orders placed, active users

Monitoring Approaches

- **Synthetic monitoring:** Simulating user transactions on a schedule to detect problems before real users encounter them.
- **Real user monitoring (RUM):** Capturing actual user experience data directly from production traffic.
- **Agent-based monitoring:** Software installed on hosts that reports detailed system metrics.
- **Agentless monitoring:** Polling devices via SNMP, APIs, or protocols without installing software.

From Data to Insight

Collecting data is only the first step. Effective monitoring requires: centralizing telemetry into a unified platform, establishing baselines for “normal” behavior, visualizing trends on dashboards tailored to different audiences (NOC, engineering, executives), and, critically, converting the right signals into actionable alerts — a topic covered in depth later in this section.

Key Takeaways

- Metrics, logs, and traces are the three pillars of observability, answering what, why, and where.
- Monitoring spans infrastructure, platform, application, and business layers.
- Synthetic, real-user, agent-based, and agentless approaches each have distinct strengths.
- Raw data must be centralized, baselined, and visualized to become actionable insight.

Event Management: Detection, Correlation, and Noise Reduction

An **event** is any detectable occurrence with significance for infrastructure or service management — a threshold breach, a status change, a completed process. Event management is the discipline of detecting, filtering, correlating, and responding to these events efficiently, without drowning operations teams in noise.

The Event Management Lifecycle

Detect Filter Correlate Classify Respond

The Alert Fatigue Problem

A single infrastructure failure can trigger hundreds of individual alerts across dependent systems — a "storm" that overwhelms responders and hides the real root cause. This is the central challenge event management must solve: too much noise buries the signal that actually matters.

Correlation Techniques

- **Topology-based correlation:** Using the service map to recognize that many downstream alerts stem from a single upstream failure.
- **Time-based correlation:** Grouping events that occur within a short window as likely related.
- **Pattern-based correlation:** Recognizing recurring event sequences that historically preceded known incidents.
- **Machine learning / AIOps correlation:** Using statistical models to automatically cluster related events and suppress duplicates, even without predefined rules.

Noise Reduction Strategies

Strategy	Effect
Deduplication	Collapse repeated identical alerts into a single event
Suppression windows	Mute known-noisy alerts during planned maintenance
Severity tuning	Ensure thresholds reflect actual business impact, not arbitrary defaults
Auto-close on resolution	Automatically clear alerts once the underlying condition resolves

Key Takeaways

- Event management detects, filters, correlates, classifies, and drives response to significant occurrences.
- Alert fatigue — too much noise burying real signal — is the central challenge to solve.
- Topology, time, pattern, and ML-based correlation techniques reduce hundreds of alerts to a handful of root causes.
- Deduplication, suppression, severity tuning, and auto-close are practical noise reduction levers.

Alerting, Thresholds, and On-Call Practices

An alert is a notification that demands human attention or automated action. Getting alerting right — the thresholds that trigger alerts and the on-call practices that route them to the right person — is one of the highest-leverage investments an ITOM team can make. Poor alerting causes both missed outages and burned-out engineers.

Designing Effective Thresholds

- **Static thresholds:** Fixed values (e.g., alert if CPU > 90%) — simple but prone to false positives during legitimate load spikes.
- **Dynamic / baseline thresholds:** Alert when a metric deviates significantly from its learned historical pattern, accounting for time-of-day and seasonality.
- **Composite thresholds:** Requiring multiple conditions together (e.g., high latency *and* high error rate) to reduce false positives from any single noisy metric.
- **Duration-based thresholds:** Requiring a condition to persist for a set period before alerting, filtering out transient blips.

Alert Severity Levels

Severity	Response Expectation
Critical (P1)	Immediate page, 24/7, active service outage
High (P2)	Page within business SLA, significant degradation
Medium (P3)	Ticket created, addressed next business day
Low (P4)	Logged for trend analysis, no immediate action

On-Call Best Practices

Rotation Fair, sustainable schedule Escalation Clear secondary path Runbooks Documented response steps Post-Incident Review & improve

- **Sustainable rotations:** Distribute on-call fairly across the team; avoid single points of human failure and burnout.
- **Clear escalation paths:** If the primary responder doesn't acknowledge within a defined window, automatically escalate to a secondary and then a manager.
- **Actionable runbooks linked to alerts:** Every alert should link directly to documented remediation steps, not require the responder to start from scratch.
- **Post-incident review:** Treat every page as a data point — recurring low-value alerts should be tuned or eliminated, not just tolerated.

Key Takeaways

- Effective thresholds balance sensitivity against false positives using static, dynamic, composite, or duration-based logic.

- Severity levels (P1–P4) set clear expectations for response time and escalation.
- Sustainable on-call rotations, clear escalation, and linked runbooks are essential to a healthy on-call culture.
- Every page should be reviewed afterward to continuously reduce noise and improve response quality.

Section 3: Discovery & Service Mapping

IT Discovery: Finding and Classifying Assets

Discovery is the automated process of identifying every device, application, service, and dependency in an IT environment. Without accurate discovery, a Configuration Management Database (CMDB) becomes stale and unreliable, undermining every downstream ITOM and ITSM process — from incident management to change risk assessment.

Why Discovery Matters

Modern environments change constantly: cloud instances spin up and down, containers are ephemeral, employees provision shadow IT, and mergers introduce unknown infrastructure. Studies consistently show that manually maintained asset inventories are 20–30% inaccurate within months. Automated discovery closes this gap by continuously scanning the environment and reconciling what's found against the CMDB.

Discovery Techniques

Agent-Based Software installed on each host Agentless / Network Scan SNMP, WMI, SSH probing API-Based Cloud provider & SaaS APIs Passive Discovery Network traffic analysis Integration-Based Pulling from existing tools (hypervisor, IAM)

Classifying Discovered Assets

Discovery finds assets; classification makes them useful. Assets should be categorized along multiple dimensions:

- **Asset type:** Server, network device, container, cloud service, SaaS application, endpoint.
- **Environment:** Production, staging, development, test.
- **Business criticality:** Mission-critical, important, standard, low-priority.
- **Ownership:** Responsible team, cost center, business unit.
- **Compliance scope:** Whether the asset falls under PCI-DSS, HIPAA, SOX, or other regulatory frameworks.

Reconciliation and CMDB Health

Discovery data must be reconciled against the CMDB on a regular cadence. Reconciliation identifies three categories of discrepancy:

- **Ghost assets:** Recorded in the CMDB but no longer discoverable — likely decommissioned without being recorded.
- **Rogue assets:** Discovered in the environment but missing from the CMDB — unmanaged or unauthorized infrastructure.
- **Attribute drift:** The asset exists in both places but key attributes (owner, IP, version) have changed and gone unrecorded.

Key Takeaways

- Automated discovery is essential to keeping a CMDB accurate as environments change continuously.
- Agent-based, agentless, API-based, passive, and integration-based are the five main discovery techniques.
- Classification by type, environment, criticality, ownership, and compliance scope makes discovery data actionable.
- Regular reconciliation catches ghost assets, rogue assets, and attribute drift before they cause incidents.

Service Mapping and Dependency Visualization

Service mapping builds on discovery by connecting individual assets into a picture of how they work together to deliver a business service. Where discovery answers "what exists?", service mapping answers "how does this application actually function, and what breaks if this server goes down?"

Why Service Maps Matter

- **Incident triage:** When an alert fires on a database server, a service map instantly shows which applications and business services are impacted.
- **Change risk assessment:** Before approving a change, teams can see every downstream dependency that could be affected.
- **Root cause analysis:** Tracing an outage back through the dependency chain to the true source is far faster with a visual map.
- **Business impact context:** Translating technical failures into business language ("checkout is down" instead of "server db-07 is unreachable").

Example Service Dependency Map

Checkout Service Payment API Inventory Service Notification Svc Payment DB Inventory DB

If *Inventory DB* fails, the map instantly shows the blast radius: Inventory Service → Checkout Service → every customer trying to buy something.

Mapping Techniques

- **Traffic-based mapping:** Observing real network flows between components to infer dependencies automatically.
- **Application performance monitoring (APM) tracing:** Using distributed tracing to follow a single transaction across services.
- **Manual/collaborative mapping:** Architecture workshops where teams document known dependencies — useful for legacy systems that resist automated discovery.
- **Configuration-based mapping:** Parsing infrastructure-as-code, deployment manifests, and connection strings to infer relationships.

Keeping Maps Current

Static, manually-drawn diagrams go stale almost immediately in dynamic environments. Mature ITOM practice treats service maps as living data, continuously refreshed by automated discovery and traffic analysis, with periodic human review to validate business-context accuracy (which discovery alone cannot infer).

Key Takeaways

- Service mapping connects discovered assets into a picture of how business services actually function.
- Maps accelerate incident triage, change risk assessment, root cause analysis, and business impact communication.
- Traffic-based, APM tracing, manual, and configuration-based techniques all contribute to building maps.
- Maps must be continuously refreshed — a stale map is often worse than no map at all.

Section 4: Orchestration & Automation

Runbook Automation and Self-Healing

Runbooks are documented, step-by-step procedures for diagnosing and resolving specific operational issues. Runbook automation converts these manual procedures into executable scripts or workflows, and self-healing systems take automation a step further by triggering remediation automatically when a defined condition is detected — without waiting for human approval.

From Manual Runbook to Self-Healing System

Manual Runbook Human reads & executes Scripted Runbook Human triggers script Approved Automation Alert → approval → run Self-Healing Fully automatic

Common Self-Healing Use Cases

- **Service restarts:** Automatically restarting a hung or crashed service process when health checks fail.
- **Auto-scaling remediation:** Adding capacity automatically when resource thresholds are breached.
- **Disk cleanup:** Automatically clearing temp files or rotating logs when disk usage crosses a threshold.
- **Failover orchestration:** Automatically redirecting traffic to a healthy region or node when a primary fails health checks.
- **Credential/certificate renewal:** Automatically rotating expiring certificates or API keys before they cause outages.

Designing Safe Automation

Automating remediation carries risk — a poorly designed self-healing action can make an incident worse. Mature ITOM teams apply these safeguards:

- **Idempotency:** Runbooks should be safe to run multiple times without unintended side effects.
- **Blast radius limits:** Automated actions should be scoped to affect the smallest possible set of resources.
- **Circuit breakers:** If an automated fix fails or repeats too many times, the system should halt and escalate to a human rather than looping indefinitely.

- **Audit logging:** Every automated action must be logged with context for post-incident review.
- **Staged rollout:** New automations should first run in "recommend" mode (suggesting the fix) before being promoted to fully autonomous execution.

Key Takeaways

- Runbook automation progresses through stages: manual, scripted, approved automation, and full self-healing.
- Service restarts, auto-scaling, disk cleanup, failover, and credential renewal are common self-healing use cases.
- Safe automation requires idempotency, blast-radius limits, circuit breakers, audit logging, and staged rollout.
- Self-healing reduces MTTR dramatically but must be introduced carefully to avoid amplifying incidents.

Orchestration Across Hybrid & Cloud Environments

Orchestration coordinates multiple automated tasks — provisioning, configuration, deployment, and scaling — into a single, repeatable workflow. As infrastructure spans on-premises data centers, private clouds, and multiple public cloud providers, orchestration becomes the connective tissue that ensures consistent, coordinated operations across all of them.

Orchestration vs. Automation

These terms are often used interchangeably but describe different scopes of work:

Automation	Orchestration
A single task performed without human intervention (e.g., restart a service)	Coordinating many automated tasks across systems into an end-to-end workflow
Scoped to one system or component	Spans multiple systems, teams, and environments
Example: auto-scaling a VM group	Example: provisioning a full application stack across on-prem and two clouds

Hybrid Orchestration Architecture

Orchestration Layer On-Premises Data Center Private Cloud VMware / OpenStack Public Cloud(s) AWS / Azure / GCP

Key Orchestration Tools & Patterns

- **Infrastructure as Code (IaC):** Terraform and similar tools provide a cloud-agnostic provisioning layer that can target multiple providers from a single codebase.
- **Container orchestration:** Kubernetes abstracts compute across clouds and on-prem clusters, enabling workload portability.

- **Workflow engines:** Tools that sequence multi-step processes (e.g., provision network → deploy app → run health checks → register in load balancer) with rollback on failure.
- **Event-driven orchestration:** Triggering cross-environment workflows in response to events (e.g., a deployment pipeline that provisions cloud resources when a build passes).

Common Challenges

- **Latency and connectivity:** Orchestrating across environments introduces network dependencies that must be resilient to partial failures.
- **Credential management:** Securely managing service accounts and API keys across multiple platforms without hard-coding secrets.
- **State drift:** Keeping the orchestration layer's view of infrastructure state consistent with what actually exists in each environment.
- **Vendor-specific features:** Balancing portability with the desire to use unique capabilities of a specific cloud provider.

Key Takeaways

- Orchestration coordinates multiple automated tasks into end-to-end workflows across environments.
- A hybrid orchestration layer sits above on-premises, private cloud, and public cloud resources.
- IaC, container orchestration, workflow engines, and event-driven triggers are core orchestration tools.
- Latency, credential management, state drift, and vendor lock-in are key challenges to design around.

Section 5: Cloud Operations & Performance

What Is Cloud Operations Management (CloudOps)?

CloudOps is the discipline of managing, monitoring, and optimizing applications and infrastructure that run in public, private, or hybrid cloud environments. It extends traditional IT Operations Management practices — monitoring, incident response, change management — to address the unique characteristics of the cloud: elasticity, consumption-based billing, API-driven infrastructure, and shared responsibility with cloud providers.

As organizations shift workloads to AWS, Azure, Google Cloud, and hybrid platforms, ITOM teams must adapt their tooling and processes to maintain visibility and control across environments that scale up and down automatically, span multiple regions, and are provisioned through code rather than manual configuration.

Visibility & Monitoring Automation & Orchestration Cost & Optimization Security & Compliance Reliability & Incident Response CloudOps Pillars

The Five Pillars of CloudOps

- **Visibility & Monitoring:** Aggregating telemetry (metrics, logs, traces) across cloud-native services, containers, and serverless functions into a single pane of glass.
- **Automation & Orchestration:** Using Infrastructure as Code (IaC), auto-scaling policies, and self-healing runbooks to reduce manual intervention.
- **Cost & Resource Optimization:** Right-sizing instances, eliminating idle resources, and applying FinOps principles to control consumption-based spend.

- **Security & Compliance:** Enforcing guardrails, identity and access management, and continuous compliance scanning within the shared responsibility model.
- **Reliability & Incident Response:** Designing for failure with redundancy, chaos testing, and rapid, well-rehearsed incident response procedures.

Multi-Cloud and Hybrid Cloud Challenges

Most enterprises operate in a hybrid or multi-cloud reality, combining on-premises data centers with two or more public cloud providers. This introduces operational complexity:

- **Tooling fragmentation** — each provider has its own native monitoring, IAM, and automation stack.
- **Inconsistent governance** — tagging standards, cost allocation, and security policies must be normalized across platforms.
- **Network complexity** — interconnecting VPCs, on-prem networks, and SaaS endpoints while maintaining low latency and security.
- **Skills gaps** — teams need cross-platform expertise rather than single-vendor specialization.

Core CloudOps Practices

Practice	Purpose
Infrastructure as Code (IaC)	Version-controlled, repeatable provisioning using Terraform, CloudFormation, or ARM templates.
Auto-scaling & elasticity	Automatically matching compute capacity to real-time demand.
Tagging & governance	Consistent metadata for cost allocation, ownership, and lifecycle management.
FinOps alignment	Cross-functional collaboration between finance, engineering, and operations on cloud spend.
Site Reliability Engineering (SRE)	Applying error budgets and SLOs to balance velocity with reliability.

Key Takeaways

- CloudOps extends traditional ITOM to address the elasticity and API-driven nature of cloud infrastructure.
- Five pillars — visibility, automation, cost optimization, security, and reliability — form the foundation of mature CloudOps practice.
- Multi-cloud and hybrid environments demand normalized governance and cross-platform tooling.
- IaC, auto-scaling, tagging, and FinOps are core practices that separate mature CloudOps teams from reactive ones.

Capacity Planning & Performance Optimization

Capacity planning ensures that IT infrastructure has enough compute, storage, and network resources to meet current and future demand — without over-provisioning and wasting budget. Performance optimization ensures that the resources in place are being used efficiently to deliver acceptable response times and throughput.

The Capacity Planning Process

1. Measure 2. Forecast 3. Model 4. Plan 5. Review

This cycle repeats continuously: measure current utilization, forecast demand trends (seasonal spikes, business growth), model future resource needs, plan procurement or scaling actions, and review outcomes against forecasts to refine the model.

Key Capacity Metrics

Metric	What It Tells You
CPU / Memory Utilization	Whether compute resources are under or over-provisioned
Storage Growth Rate	Time until storage capacity is exhausted
Network Throughput	Bandwidth headroom against peak traffic
Response Time / Latency	End-user experience under current load
Queue Depth	Whether requests are backing up faster than they're processed

Performance Optimization Techniques

- **Right-sizing:** Matching instance/VM sizes to actual workload requirements based on historical utilization data.
- **Caching & content delivery:** Reducing load on backend systems by serving frequently requested data from cache or CDN edge nodes.
- **Load balancing:** Distributing traffic evenly across available resources to avoid hotspots.
- **Database tuning:** Index optimization, query tuning, and connection pooling to reduce latency.
- **Horizontal vs. vertical scaling:** Adding more instances (horizontal) versus increasing the size of existing instances (vertical), chosen based on workload characteristics.

Predictive vs. Reactive Approaches

Mature ITOM teams shift from *reactive* capacity management (responding to alerts after resources are exhausted) to *predictive* capacity management, using historical trend analysis and machine learning-based forecasting to anticipate needs weeks or months in advance. This reduces the risk of performance degradation while avoiding the cost of permanent over-provisioning.

Key Takeaways

- Capacity planning is a continuous cycle: measure, forecast, model, plan, and review.
- Track utilization, growth rate, throughput, latency, and queue depth as core capacity metrics.
- Right-sizing, caching, load balancing, and database tuning are core performance optimization levers.
- Predictive, data-driven forecasting outperforms reactive, alert-driven capacity management.

ITOM Metrics, KPIs, and the Maturity Model

To improve IT Operations Management continuously, teams need objective measures of both operational health and process maturity. This lesson covers the core metrics that quantify ITOM performance and a maturity model for benchmarking progress.

Core ITOM Metrics

Category	Metric	Why It Matters
Availability	System / Service Uptime %	Directly reflects reliability delivered to the business
Incidents	MTTD / MTTR	Mean Time to Detect / Resolve — speed of operational response
Change	Change Success Rate	% of changes deployed without causing incidents
Automation	% of Tasks Automated	Reduction in manual, repetitive operational work
Cost	Cost per Managed Asset/Workload	Operational efficiency relative to infrastructure footprint
Alerting	Alert-to-Incident Ratio	Signal-to-noise quality of the monitoring stack

The ITOM Maturity Model

1. Reactive 2. Proactive 3. Service- Aligned 4. Predictive 5. Autonomous

The Five Maturity Levels

- **1. Reactive:** Operations respond to outages after users report them. No formal monitoring or documented processes.
- **2. Proactive:** Monitoring tools detect issues before users notice; basic incident and change processes are documented.
- **3. Service-Aligned:** IT operations are mapped to business services; SLAs are defined and tracked; ITSM/ITOM tools are integrated.
- **4. Predictive:** AIOps and analytics anticipate failures and capacity constraints before they occur; automation handles routine remediation.
- **5. Autonomous:** Self-healing systems detect, diagnose, and resolve the majority of issues without human intervention; operations teams focus on strategic improvement.

Key Takeaways

- Availability, MTTD/MTTR, change success rate, automation %, cost, and alert quality are core ITOM metrics.

- The ITOM maturity model spans five levels: Reactive, Proactive, Service-Aligned, Predictive, and Autonomous.
- Progression through maturity levels requires investment in integration, automation, and analytics — not just tooling purchases.
- Metrics should be tied to business outcomes, not just technical performance, to demonstrate ITOM's value.