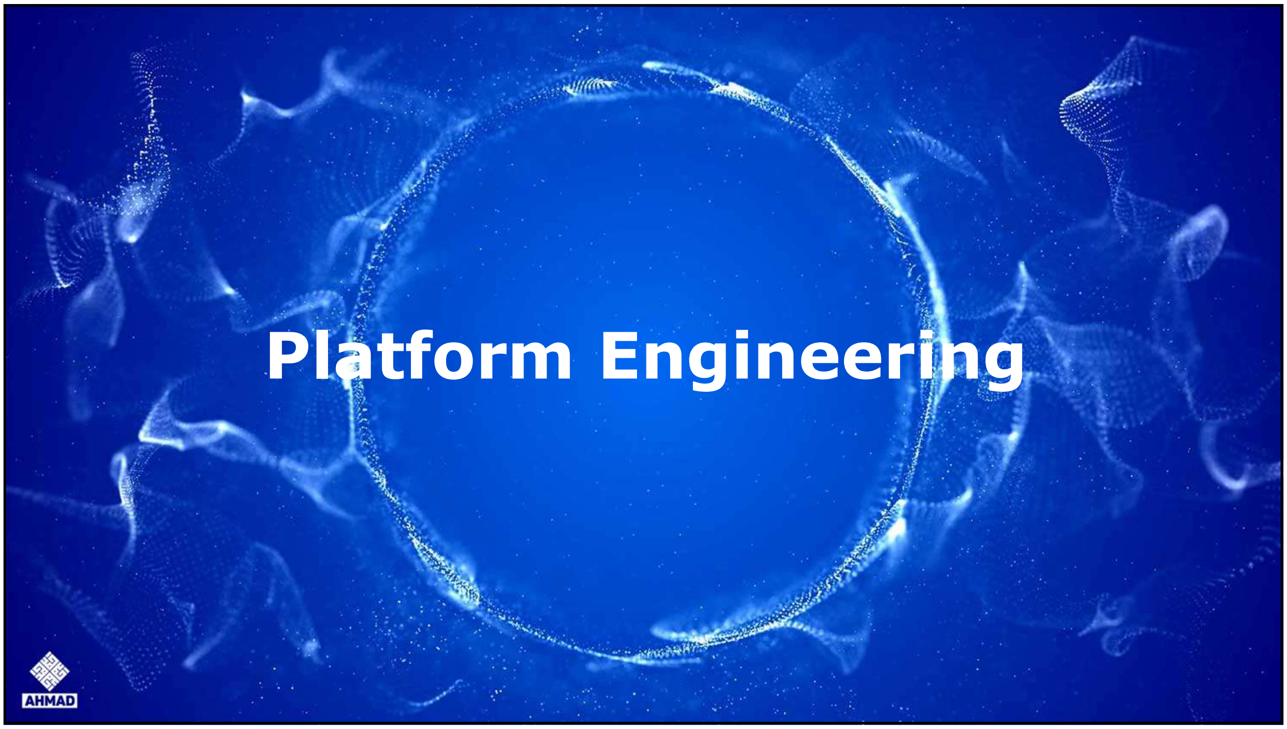
Platform Engineering, AIOps & Continuous Improvement

1



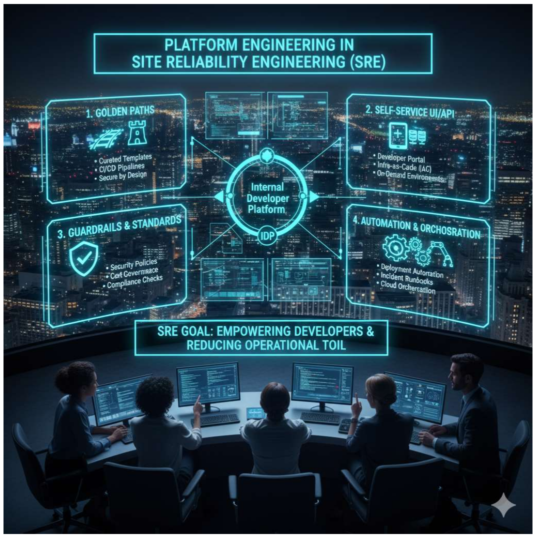
Platform Engineering

2

What is Platform Engineering

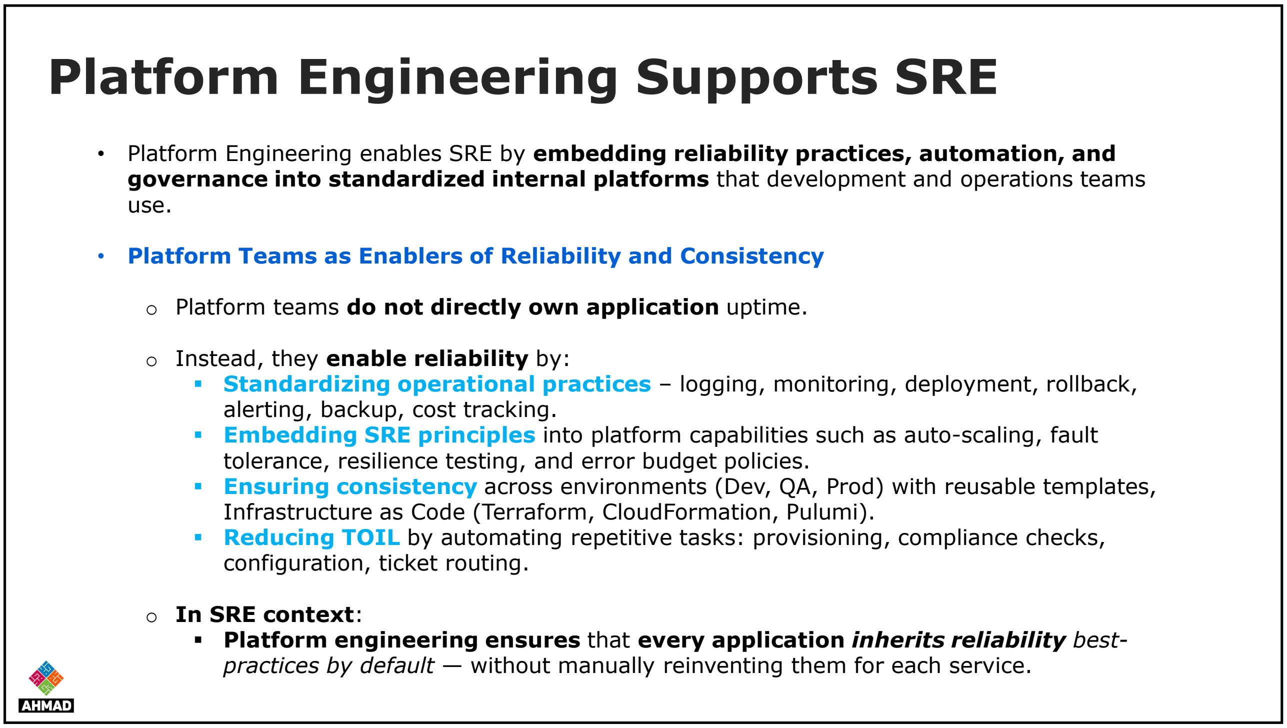
- **Platform Engineering** is the practice of **designing**, **building**, and **maintaining** ***internal platforms*** that **provide engineers** with self-service **capabilities — such as automated provisioning, CI/CD workflows, observability, security, and compliance—through** standardized **tools**, **templates**, and **processes**.

- It bridges the gap between development and operations by providing **reliable, consistent, and automated platforms** that accelerate delivery while ensuring reliability.



Platform Engineering … continue

- **It is characterized by**:
  - **Product Mindset**
    - The **platform** itself is treated as an **internal product**, with **developers** as its **customers**.
    - The platform team **gathers feedback**, **manages** a **roadmap**, and aims to deliver a cohesive, user-friendly experience.
  - **Abstraction and Self-Service**
    - The platform team **provides managed services** and **APIs** that allow **developers** to provision infrastructure, deploy code, and set up monitoring.
  - **Standardization and Consistency**
    - By creating **standardized tools** and **workflows**, the platform ensures that all applications are built, deployed, and run in a consistent, compliant, and secure manner across the organization.



Platform Engineering Supports SRE

- Platform Engineering enables SRE by **embedding reliability practices, automation, and governance into standardized internal platforms** that development and operations teams use.

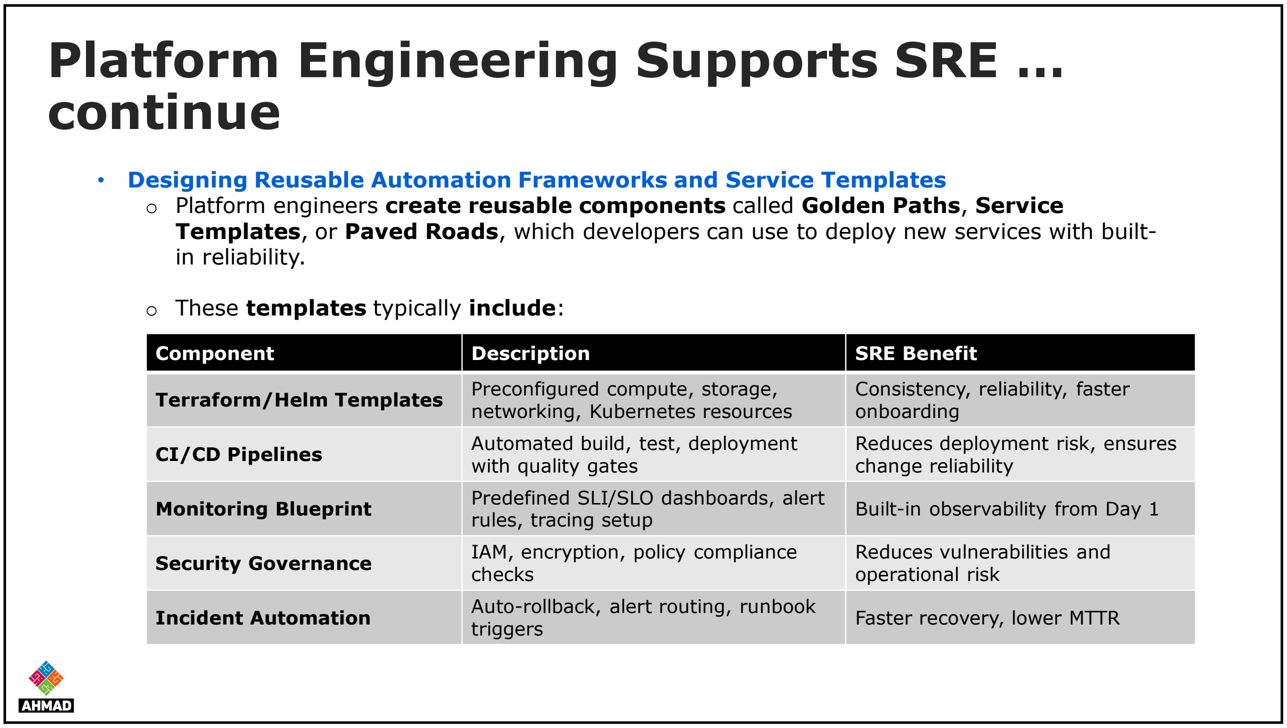
- **Platform Teams as Enablers of Reliability and Consistency**

  - Platform teams **do not directly own application** uptime.

  - Instead, they **enable reliability** by:
    - **Standardizing operational practices** – logging, monitoring, deployment, rollback, alerting, backup, cost tracking.
    - **Embedding SRE principles** into platform capabilities such as auto-scaling, fault tolerance, resilience testing, and error budget policies.
    - **Ensuring consistency** across environments (Dev, QA, Prod) with reusable templates, Infrastructure as Code (Terraform, CloudFormation, Pulumi).
    - **Reducing TOIL** by automating repetitive tasks: provisioning, compliance checks, configuration, ticket routing.

  - **In SRE context**:
    - **Platform engineering ensures** that **every application *inherits reliability*** *best-practices by default* — without manually reinventing them for each service.

5



Platform Engineering Supports SRE … continue

- **Designing Reusable Automation Frameworks and Service Templates**
  - Platform engineers **create reusable components** called **Golden Paths**, **Service Templates**, or **Paved Roads**, which developers can use to deploy new services with built-in reliability.

  - These **templates** typically **include**:

Component	Description	SRE Benefit
Terraform/Helm Templates	Preconfigured compute, storage, networking, Kubernetes resources	Consistency, reliability, faster onboarding
CI/CD Pipelines	Automated build, test, deployment with quality gates	Reduces deployment risk, ensures change reliability
Monitoring Blueprint	Predefined SLI/SLO dashboards, alert rules, tracing setup	Built-in observability from Day 1
Security Governance	IAM, encryption, policy compliance checks	Reduces vulnerabilities and operational risk
Incident Automation	Auto-rollback, alert routing, runbook triggers	Faster recovery, lower MTTR

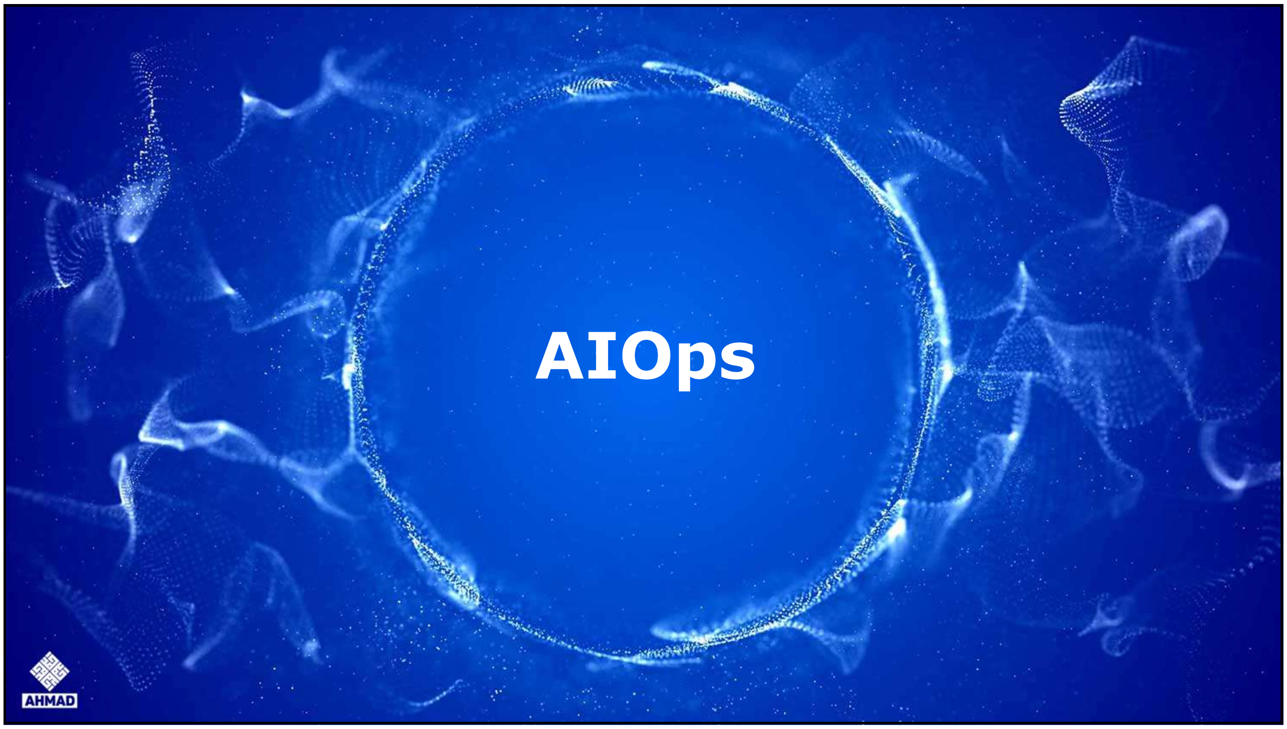
6

Platform Engineering Supports SRE … continue

- **Example**:
  - A Platform team provides a **"Service Deployment Template"**.
  - **Developers select** a **template** and **automatically get**:
    - SLO monitoring (latency, error rate)
    - Canary deployment
    - Auto-restart and self-healing
    - Logging and tracing integration
    - Error budget tracking
- No manual setup — *reliability is built-in.*
- Platform Engineering **empowers SRE practices by design**, making reliability, consistency, and automation *default rather than optional*.
- It transforms operations from "manual firefighting" to **reusable engineering**.

AHMAD

7



AIOps

AHMAD

8

What is AIOps

- AIOps (**Artificial Intelligence for IT Operations**) is a multi-layered technology platform that combines **Big Data** and **Machine Learning (ML)** to automatically analyze the massive, complex data streams generated by IT infrastructure, applications, and performance monitoring tools.
- Its primary goal is to **automate and enhance IT operations** by turning reactive, manual processes into proactive, intelligent, and autonomous workflows.
- **Tools**: BigPanda, Moogsoft, IBM Watson AIOps

AIOps: ARTIFICIAL INTELLIGENCE FOR IT OPERATIONS
1. ALERT NOISE REDUCTION
2. ANOMALY DETECTION
AI CORRELLATION ENGINE
3. PREDICTIVE ANALYTICS
3. AUTOMATED ROOT CAUSE ANALYSIS
SRE GOAL: PROACTIVE INCIDENT PREVENTION & INTELLIGENT AUTOMATION

9

AIOps | Types

- AIOps solutions are generally **categorized** by the **breadth** of their **focus**:
  - **Domain-Centric AIOps**
    - **Focused** on a specific, **single IT domain** (*e.g., Network Performance Monitoring, Application Performance Monitoring (APM), or Secu*rity).
    - These tools use **AI/ML tailored** to a narrow set of **data types** and **problems within** that **domain**.
  - **Domain-Agnostic AIOps**
    - Designed to **integrate data** from across the e**ntire IT stack** (*logs, metrics, events, topology, change data, etc.*) regardless of the source.
    - These platforms **correlate alerts** and **find root causes** across **network**, **application**, and infrastructure boundaries.

10

AIOps | Capabilities

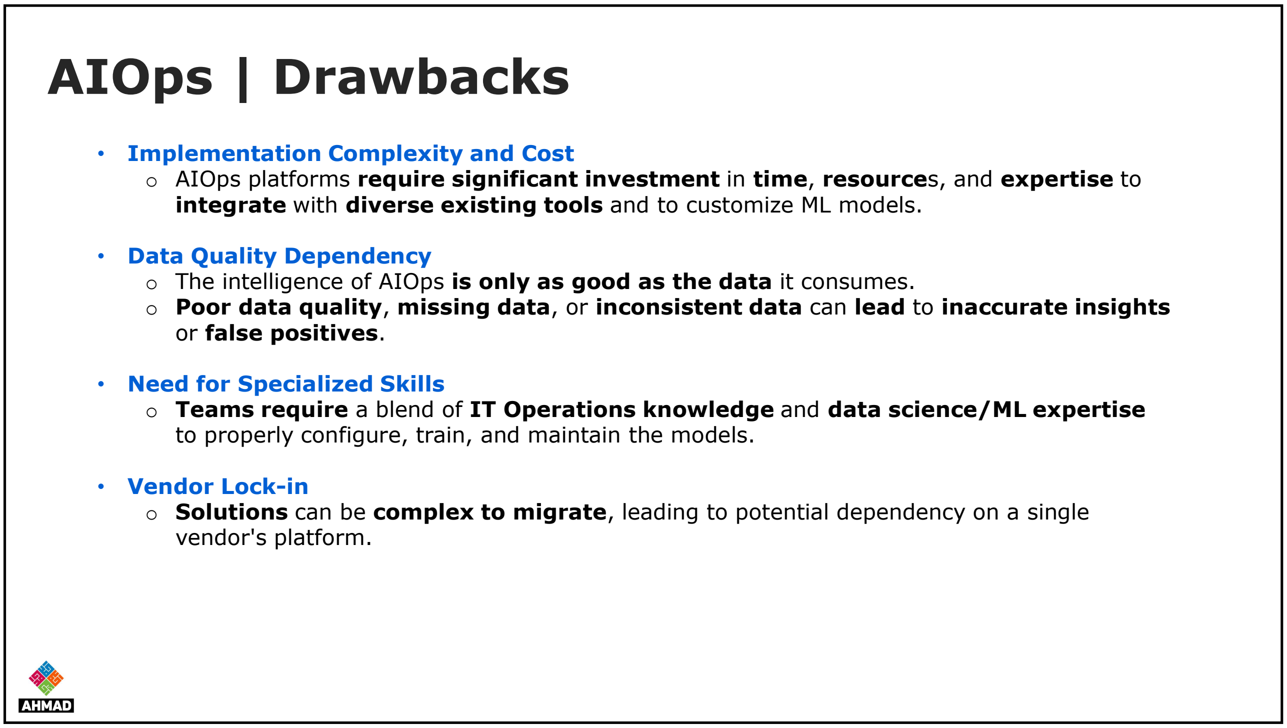
AIOps	Description	Benefit	Example in Real Scenario
Anomaly Detection	Detects **unusual patterns**, **sudden metric spikes**, or **abnormal system behavior**	**Prevents silent failures** and unexpected outages	AI detects **unusual memory consumption** in a database server **at 2 AM** (*much higher than usual*)
Event Correlation	**Groups related alerts** from multiple tools into a single meaningful incident	**Reduces alert fatigue** and noise	**150 alerts from servers**, DB, and network are automatically correlated into one major **incident**: *"Payment service API failure caused by DB connection timeout."*
Root Cause Analysis (RCA)	**Identifies** the **most probable cause** of failure using historical and dependency data	Faster troubleshooting and **MTTR reduction**	AI **analyzes event history** and reports: *"Likely root cause: recent change in firewall policy blocking API traffic."*
Predictive Analytics	**Uses ML** to **forecast failures**, performance degradation, or capacity exhaustion	Enables **proactive incident prevention**	AI **predicts** that **storage** will **reach 90% capacity** in **12 days** — **triggers auto-scale** and sends proactive alert
Auto-Remediation	**Automatically resolves recurring** or **known issues** using scripts or workflows	**Reduces manual toil** and speeds up resolution	When a service becomes unresponsive, AI **automatically restarts** the Kubernetes pod or clears cache without human intervention
Business Impact Analysis	**Maps incidents** to business services, **SLAs**, or revenue impact	Helps **prioritize critical incidents**	AI **detects** that **latency issues** in **login API** are impacting 8,000 users and **may breach SLA** — marks it as *high priority business incident*

AHMAD

AIOps | Benefits

- The core value of AIOps lies in its ability to manage the scale and complexity of modern, cloud-native environments that overwhelm human operators.
- **Alert Noise Reduction:** ML models correlate thousands of raw, noisy events and alerts into a few **actionable incidents**. This drastically reduces **alert fatigue** for on-call engineers.
- **Faster Root Cause Analysis (RCA):** AIOps quickly analyzes system dependencies and change data to pinpoint the most likely root cause, significantly lowering the **Mean Time To Resolution (MTTR)**.
- **Proactive/Predictive Detection:** By learning normal behavior patterns, AIOps can perform **anomaly detection**, flagging subtle deviations *before* a traditional static threshold is breached, preventing outages rather than reacting to them.
- **Automation:** Automates repetitive tasks, from triaging and routing tickets to executing self-healing runbooks (e.g., restarting a service or scaling a resource).
- **Reduced Operational Costs:** Increased automation and faster incident resolution reduce the time SREs and Operations teams spend on firefighting, lowering overall operational expenditure.

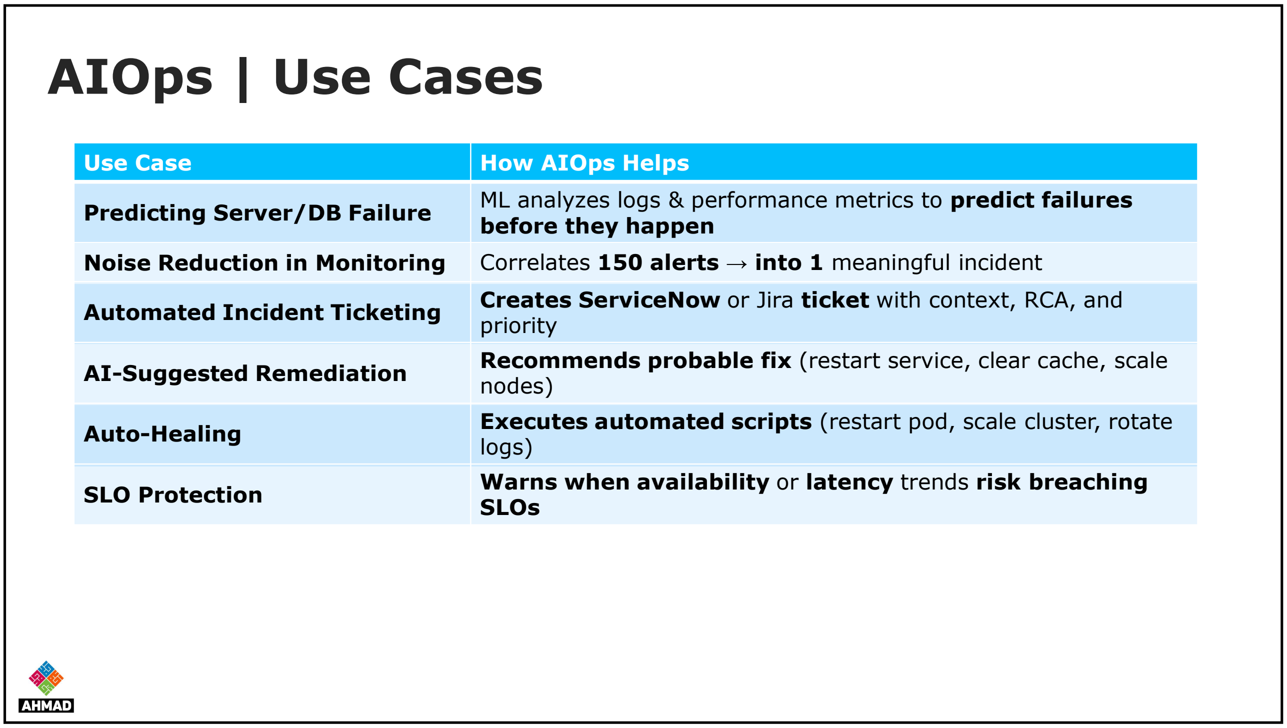
AHMAD



AIOps | Drawbacks

- **Implementation Complexity and Cost**
  - AIOps platforms **require significant investment** in **time**, **resource**s, and **expertise** to **integrate** with **diverse existing tools** and to customize ML models.
- **Data Quality Dependency**
  - The intelligence of AIOps **is only as good as the data** it consumes.
  - **Poor data quality**, **missing data**, or **inconsistent data** can **lead** to **inaccurate insights** or **false positives**.
- **Need for Specialized Skills**
  - **Teams require** a blend of **IT Operations knowledge** and **data science/ML expertise** to properly configure, train, and maintain the models.
- **Vendor Lock-in**
  - **Solutions** can be **complex to migrate**, leading to potential dependency on a single vendor's platform.

13



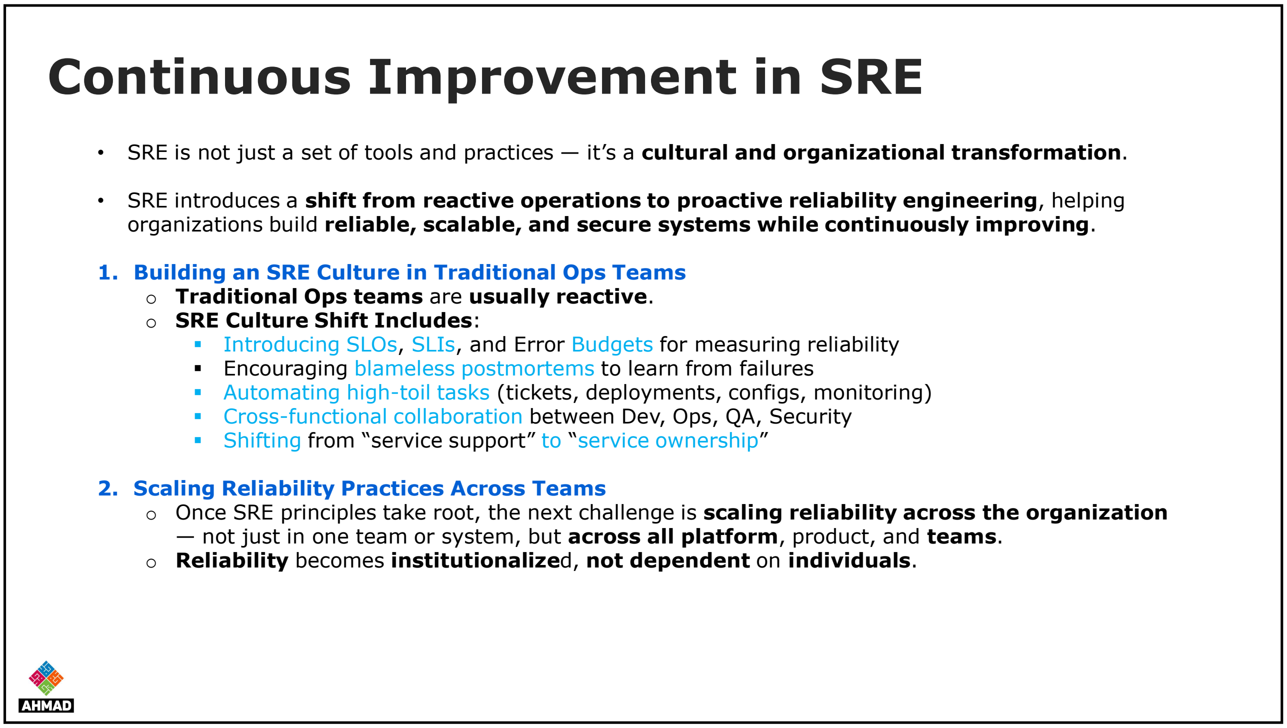
AIOps | Use Cases

Use Case	How AIOps Helps
Predicting Server/DB Failure	ML analyzes logs & performance metrics to **predict failures before they happen**
Noise Reduction in Monitoring	Correlates **150 alerts** → **into 1** meaningful incident
Automated Incident Ticketing	**Creates ServiceNow** or Jira **ticket** with context, RCA, and priority
AI-Suggested Remediation	**Recommends probable fix** (restart service, clear cache, scale nodes)
Auto-Healing	**Executes automated scripts** (restart pod, scale cluster, rotate logs)
SLO Protection	**Warns when availability** or **latency** trends **risk breaching SLOs**

14

Organisational Impact & Continuous Improvement in SRE

15



Continuous Improvement in SRE

- SRE is not just a set of tools and practices — it's a **cultural and organizational transformation**.
- SRE introduces a **shift from reactive operations to proactive reliability engineering**, helping organizations build **reliable, scalable, and secure systems while continuously improving**.

1. **Building an SRE Culture in Traditional Ops Teams**
   - **Traditional Ops teams** are **usually reactive**.
   - **SRE Culture Shift Includes**:
     - Introducing SLOs, SLIs, and Error Budgets for measuring reliability
     - Encouraging blameless postmortems to learn from failures
     - Automating high-toil tasks (tickets, deployments, configs, monitoring)
     - Cross-functional collaboration between Dev, Ops, QA, Security
     - Shifting from "service support" to "service ownership"
2. **Scaling Reliability Practices Across Teams**
   - Once SRE principles take root, the next challenge is **scaling reliability across the organization** — not just in one team or system, but **across all platform**, product, and **teams**.
   - **Reliability** becomes **institutionalize**d, **not dependent** on **individuals**.

16

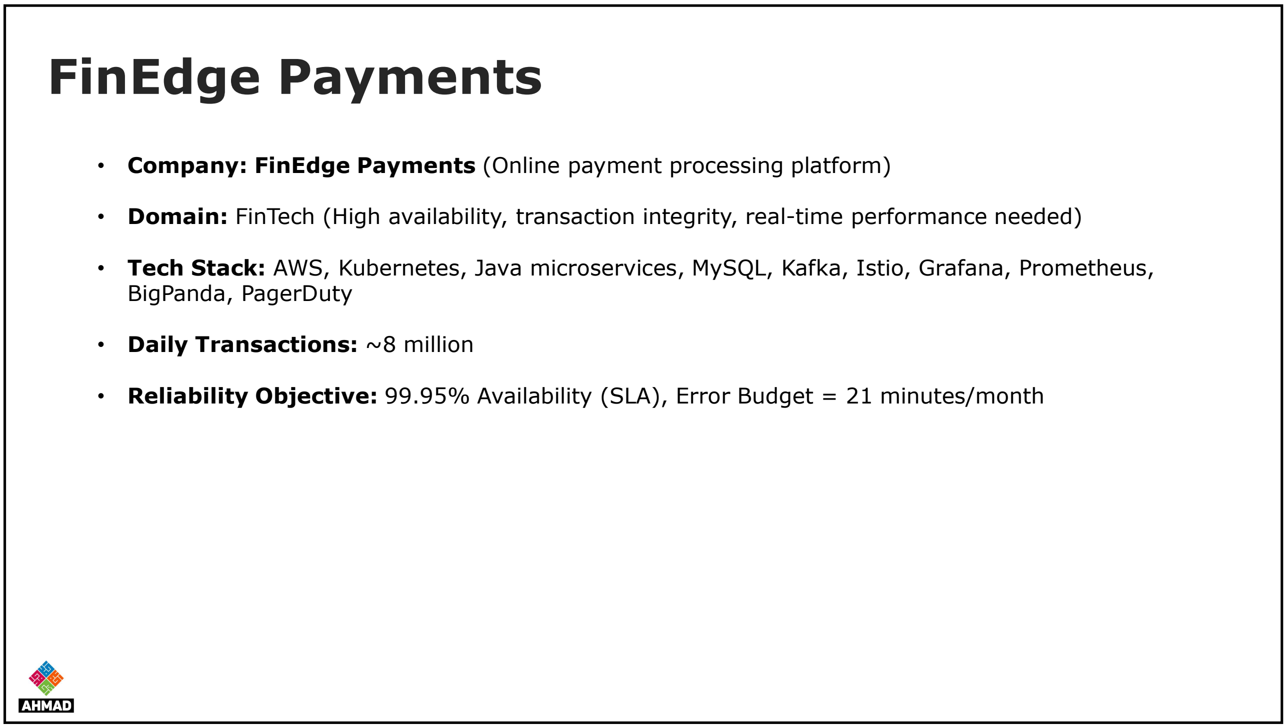
Scaling SRE in an Enterprise | Use Case

- A **large enterprise** has **multiple application teams** with inconsistent reliability practices.
- **Some have monitoring**, **others don't**.
- **Some use automation**, **others manually** run deployments.
- **Production breaches happen frequently**.
- **SRE-driven Solution**:
  1. Introduced Service Ownership Model → Teams own their reliability
  2. Created a Standard Reliability Playbook (templates for SLOs, alerts, resilience tests)
  3. Established a Central SRE Enablement Team
  4. Onboarded each app team to:
     - Define SLOs/SLIs
     - Implement monitoring and auto-rollback
     - Use golden CI/CD templates
  5. Introduced Quarterly Reliability Reviews and Error Budget tracking
- **Outcome**:
  - Incidents reduced
  - Faster recovery (MTTR improved)
  - Consistent automation across teams

17

Use Case

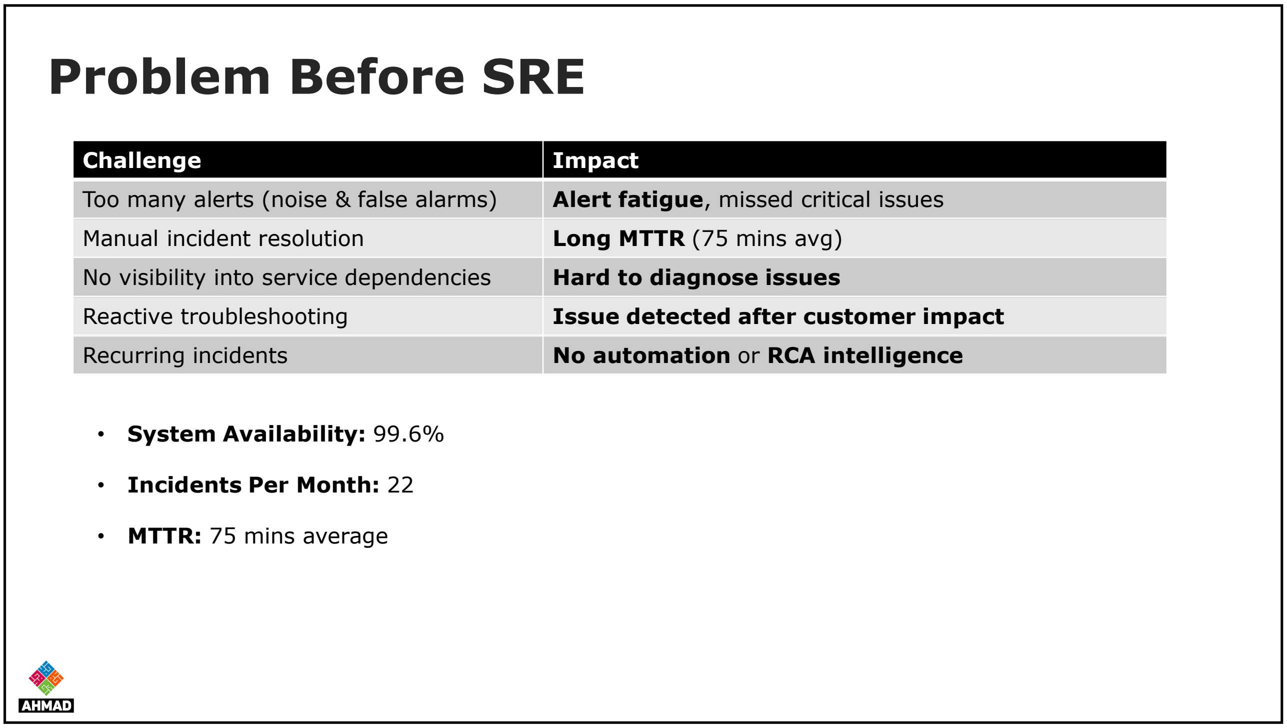
18



FinEdge Payments

- **Company: FinEdge Payments** (Online payment processing platform)
- **Domain:** FinTech (High availability, transaction integrity, real-time performance needed)
- **Tech Stack:** AWS, Kubernetes, Java microservices, MySQL, Kafka, Istio, Grafana, Prometheus, BigPanda, PagerDuty
- **Daily Transactions:** ~8 million
- **Reliability Objective:** 99.95% Availability (SLA), Error Budget = 21 minutes/month

19



Problem Before SRE

Challenge	Impact
Too many alerts (noise & false alarms)	**Alert fatigue**, missed critical issues
Manual incident resolution	**Long MTTR** (75 mins avg)
No visibility into service dependencies	**Hard to diagnose issues**
Reactive troubleshooting	**Issue detected after customer impact**
Recurring incidents	**No automation** or **RCA intelligence**

- **System Availability:** 99.6%
- **Incidents Per Month:** 22
- **MTTR:** 75 mins average

20

SRE Intervention Strategy

- **Introduced Observability (Not Just Monitoring)**

Tool	Use
Prometheus	Metrics collection
Grafana	Visualization, dashboards
Loki / ELK	Log analytics
Jaeger	Distributed tracing
BigPanda	AI-based event correlation & RCA

- **Enabled Full-stack visibility** — metrics, logs, traces, dependencies.

21

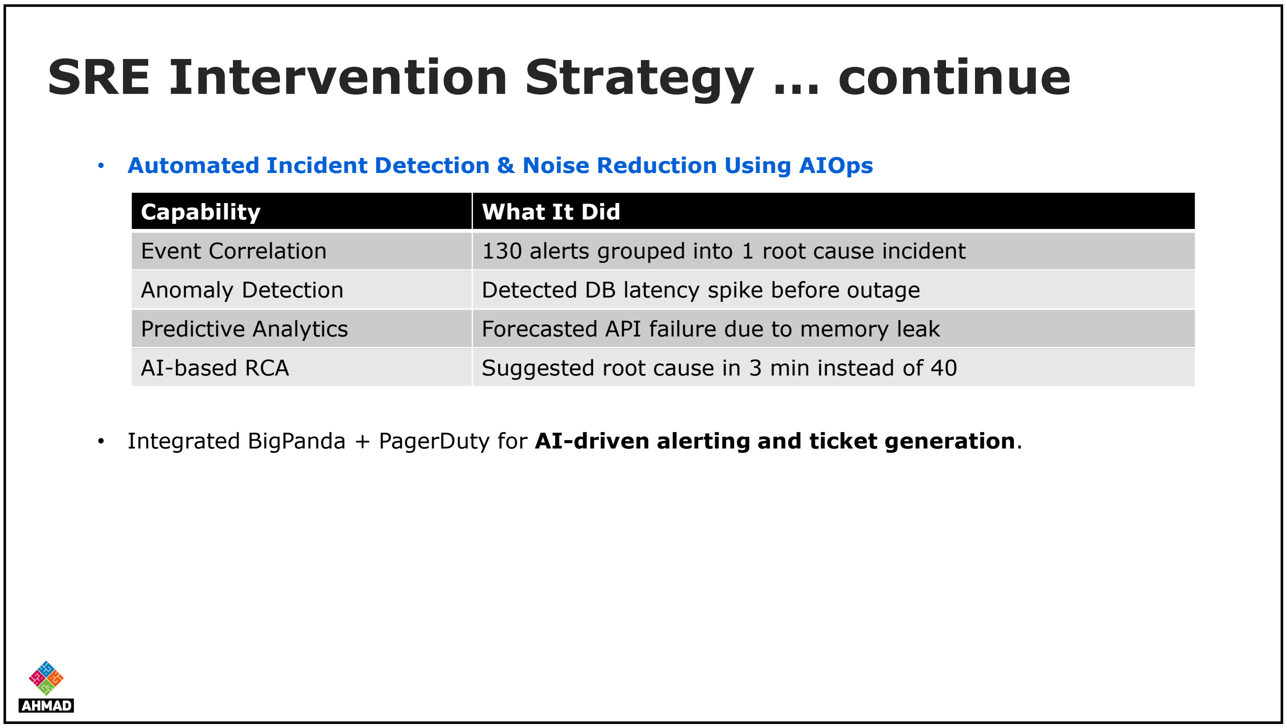
SRE Intervention Strategy … continue

- **Implemented SLOs & Error Budgets**

Service	SLO	Error Budget (Monthly)
Transaction API	99.95% availability	21 mins
Payment Gateway	<400 ms latency	5 hrs. slow response budget

- Triggered **SLO-based alerting** — alerts only when customer experience was at risk, reducing noise.

22



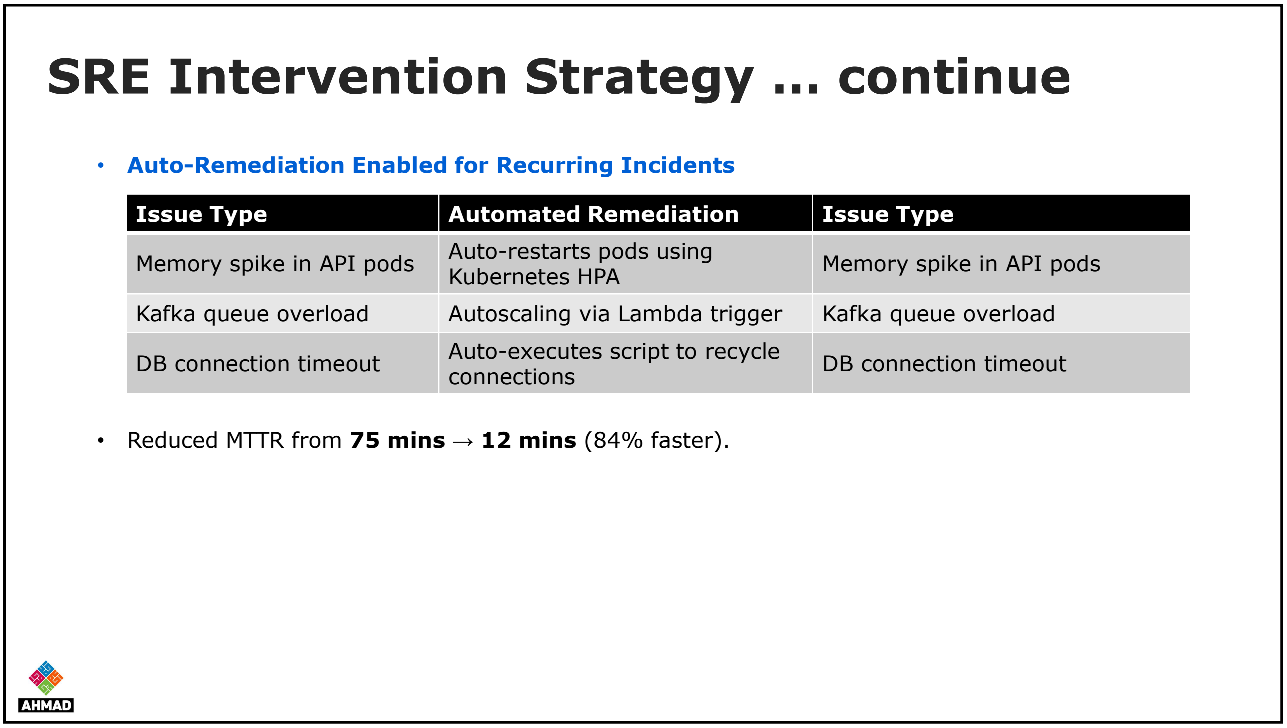
SRE Intervention Strategy … continue

- **Automated Incident Detection & Noise Reduction Using AIOps**

Capability	What It Did
Event Correlation	130 alerts grouped into 1 root cause incident
Anomaly Detection	Detected DB latency spike before outage
Predictive Analytics	Forecasted API failure due to memory leak
AI-based RCA	Suggested root cause in 3 min instead of 40

- Integrated BigPanda + PagerDuty for **AI-driven alerting and ticket generation**.

23



SRE Intervention Strategy … continue

- **Auto-Remediation Enabled for Recurring Incidents**

Issue Type	Automated Remediation	Issue Type
Memory spike in API pods	Auto-restarts pods using Kubernetes HPA	Memory spike in API pods
Kafka queue overload	Autoscaling via Lambda trigger	Kafka queue overload
DB connection timeout	Auto-executes script to recycle connections	DB connection timeout

- Reduced MTTR from **75 mins** → **12 mins** (84% faster).

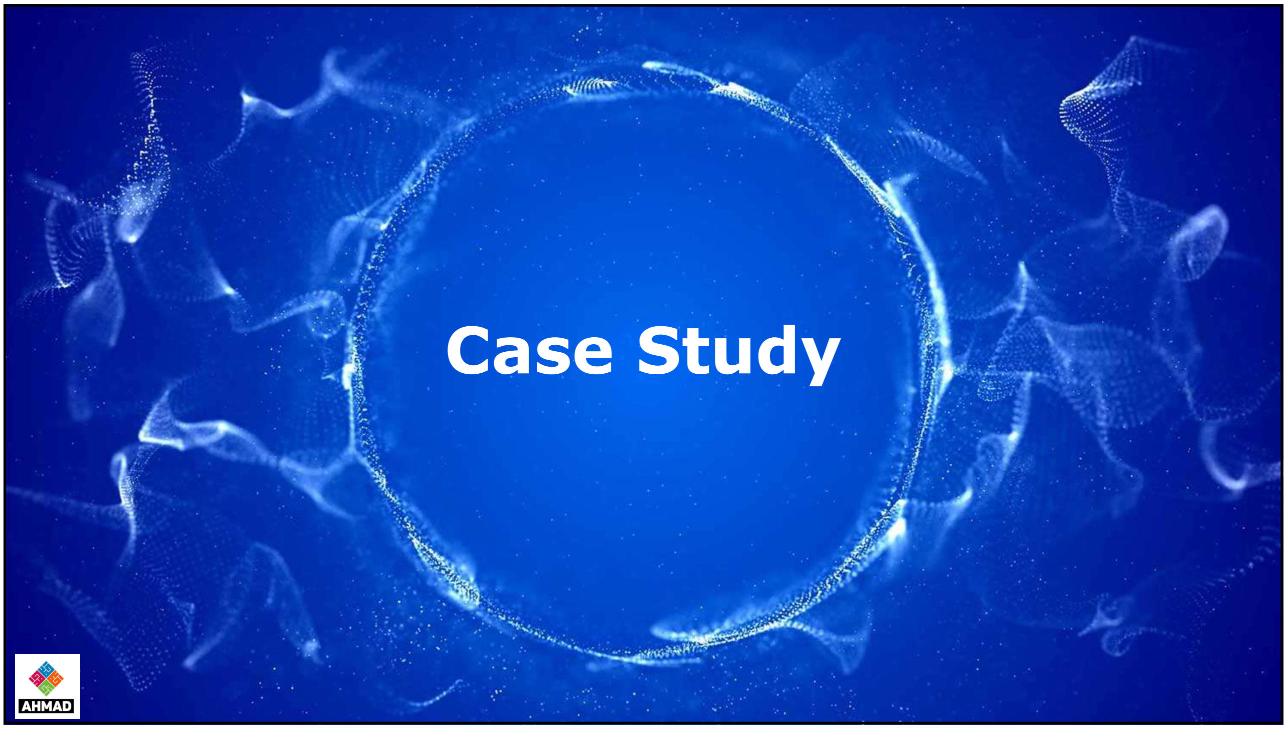
24

Business Outcome After Applying SRE

Metric	Before SRE	After SRE
Incidents per month	22	6
MTTR (Mean Time to Recovery)	75 mins	12 mins
Alert Noise	320/month	70/month
On-call Fatigue	High	Low
Availability	99.6%	99.94%
Customer Complaints	High	Reduced by 60%

AHMAD

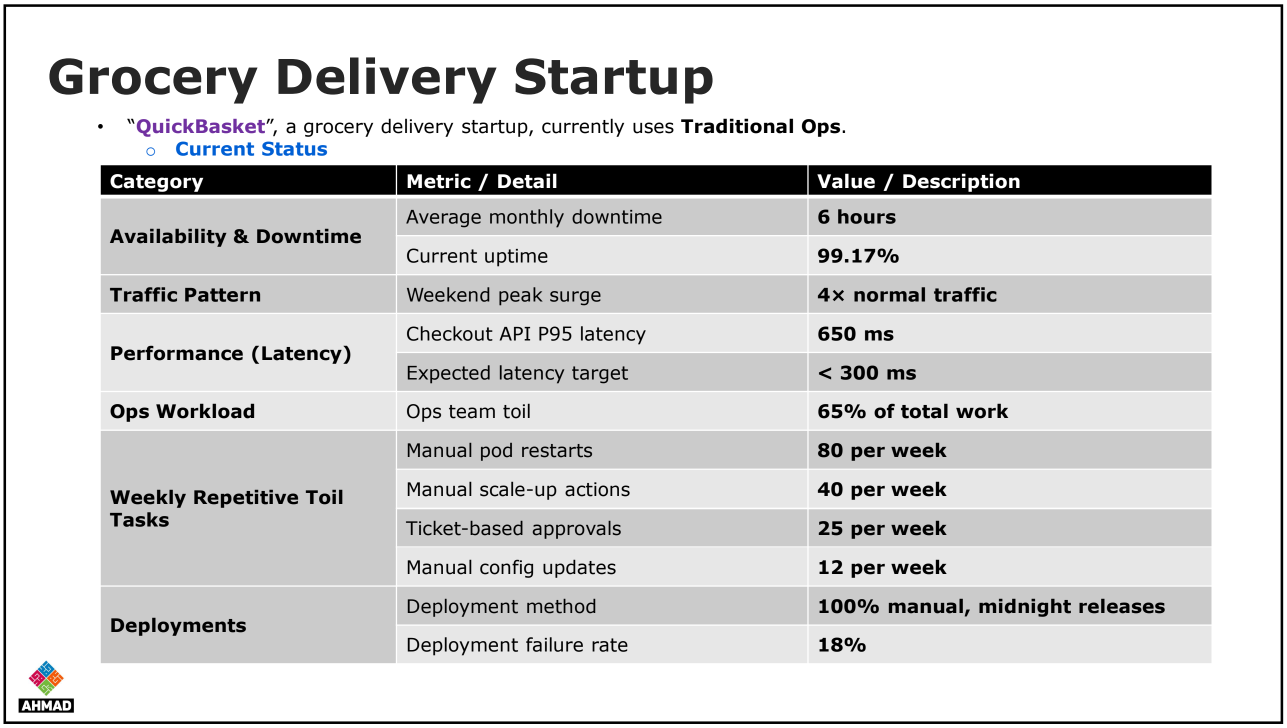
25



Case Study

AHMAD

26



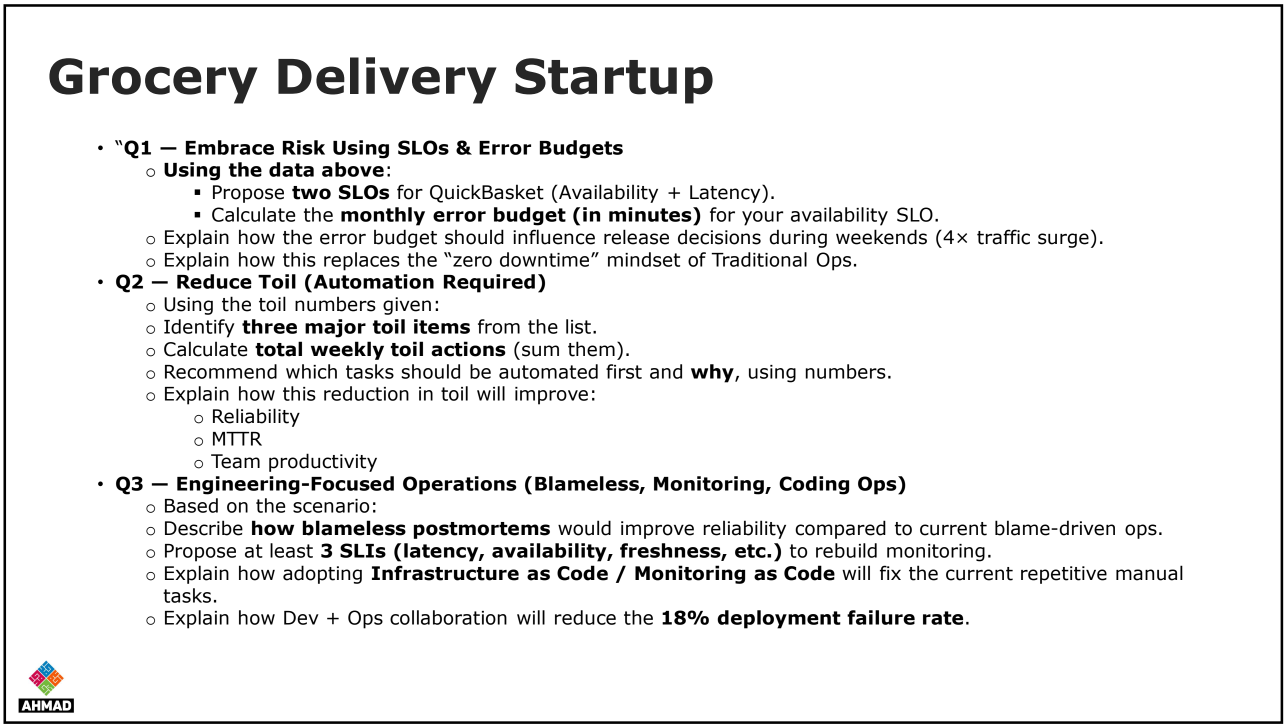
Grocery Delivery Startup

- "**QuickBasket**", a grocery delivery startup, currently uses **Traditional Ops**.
  - **Current Status**

Category	Metric / Detail	Value / Description
Availability & Downtime	Average monthly downtime	**6 hours**
	Current uptime	**99.17%**
Traffic Pattern	Weekend peak surge	**4× normal traffic**
Performance (Latency)	Checkout API P95 latency	**650 ms**
	Expected latency target	**< 300 ms**
Ops Workload	Ops team toil	**65% of total work**
Weekly Repetitive Toil Tasks	Manual pod restarts	**80 per week**
	Manual scale-up actions	**40 per week**
	Ticket-based approvals	**25 per week**
	Manual config updates	**12 per week**
Deployments	Deployment method	**100% manual, midnight releases**
	Deployment failure rate	**18%**

AHMAD

27



Grocery Delivery Startup

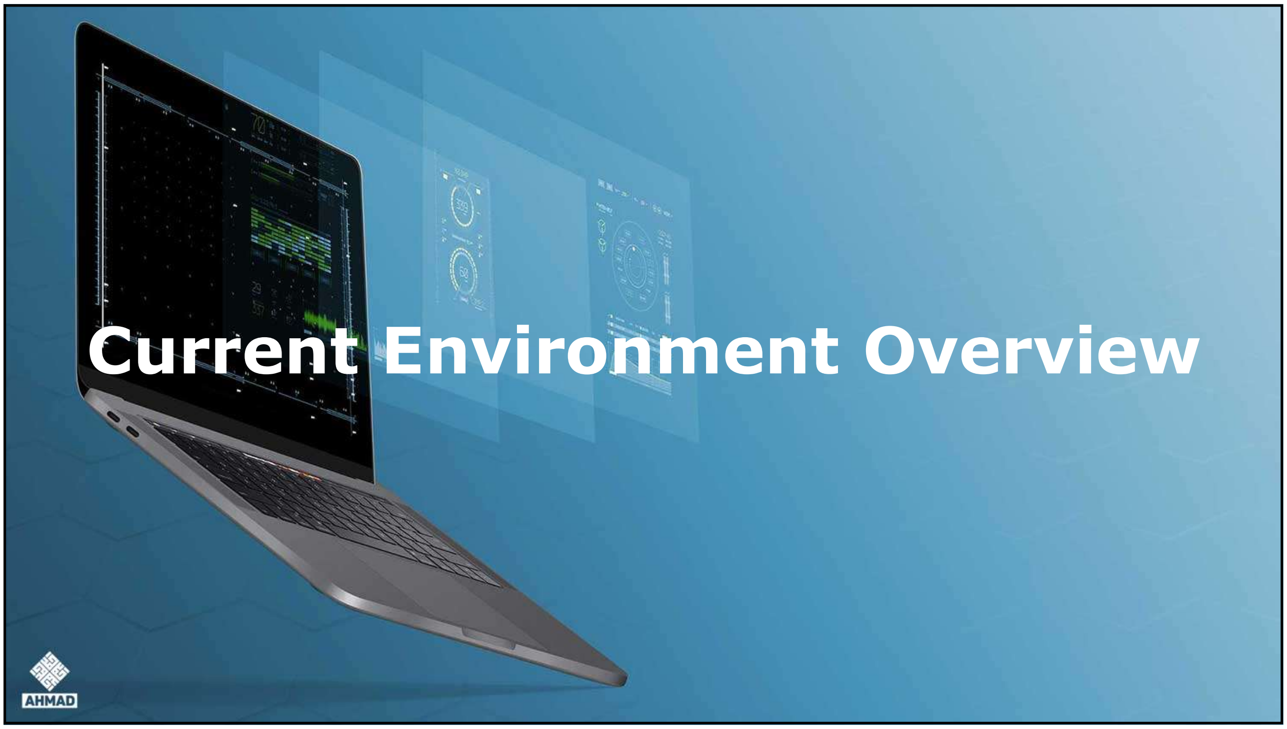
- "**Q1 — Embrace Risk Using SLOs & Error Budgets**
  - **Using the data above**:
    - Propose **two SLOs** for QuickBasket (Availability + Latency).
    - Calculate the **monthly error budget (in minutes)** for your availability SLO.
  - Explain how the error budget should influence release decisions during weekends (4× traffic surge).
  - Explain how this replaces the "zero downtime" mindset of Traditional Ops.
- **Q2 — Reduce Toil (Automation Required)**
  - Using the toil numbers given:
  - Identify **three major toil items** from the list.
  - Calculate **total weekly toil actions** (sum them).
  - Recommend which tasks should be automated first and **why**, using numbers.
  - Explain how this reduction in toil will improve:
    - Reliability
    - MTTR
    - Team productivity
- **Q3 — Engineering-Focused Operations (Blameless, Monitoring, Coding Ops)**
  - Based on the scenario:
  - Describe **how blameless postmortems** would improve reliability compared to current blame-driven ops.
  - Propose at least **3 SLIs (latency, availability, freshness, etc.)** to rebuild monitoring.
  - Explain how adopting **Infrastructure as Code / Monitoring as Code** will fix the current repetitive manual tasks.
  - Explain how Dev + Ops collaboration will reduce the **18% deployment failure rate**.

AHMAD

28

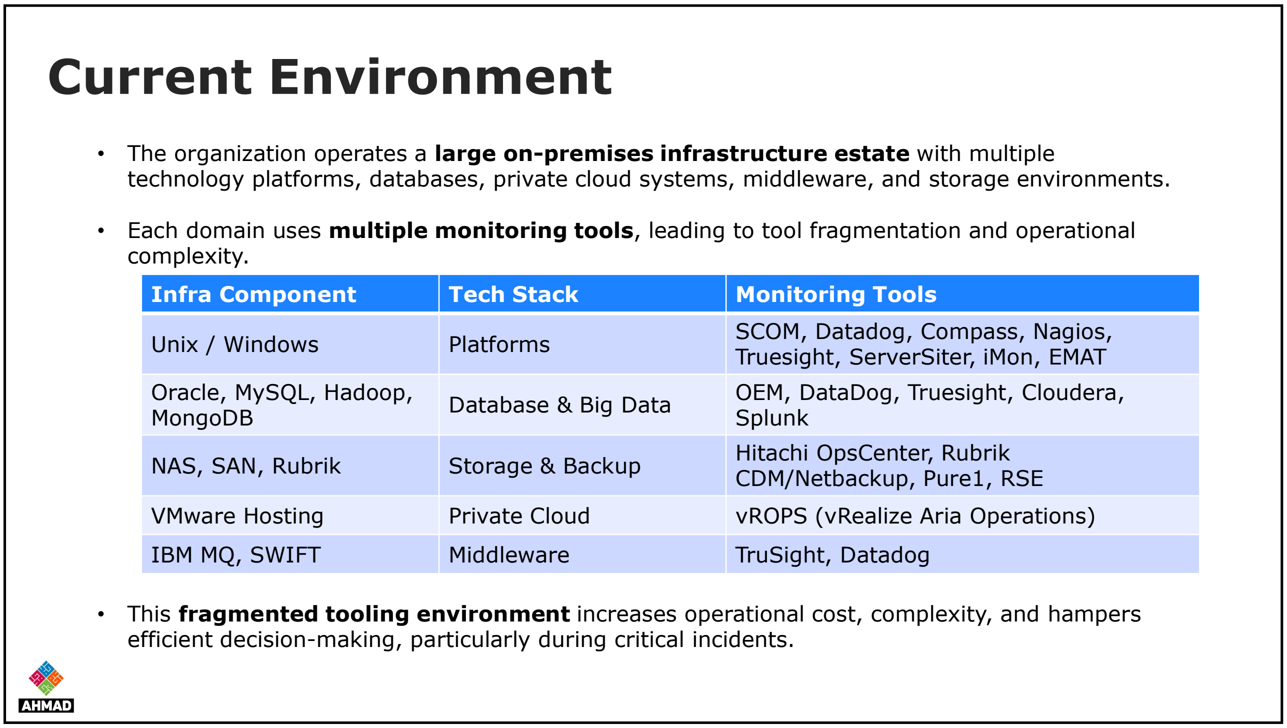
Business Case for Adopting Site Reliability Engineering (SRE)

29



Current Environment Overview

30



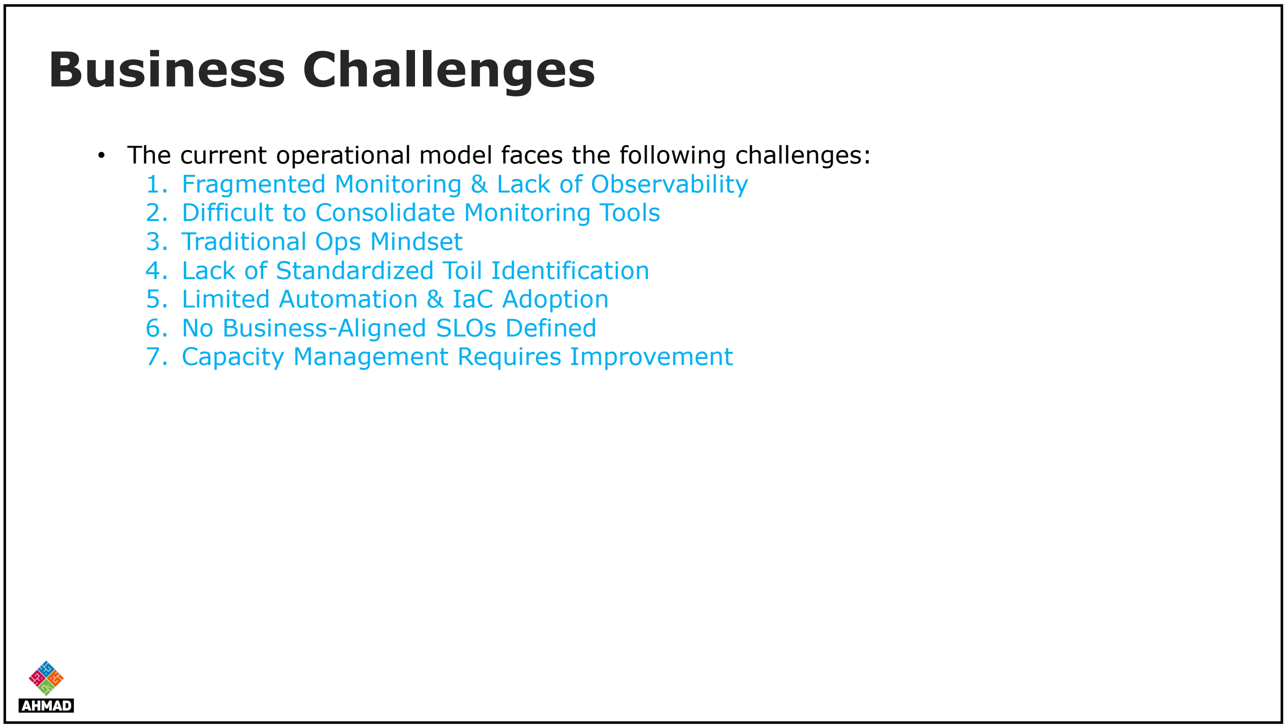
Current Environment

- The organization operates a **large on-premises infrastructure estate** with multiple technology platforms, databases, private cloud systems, middleware, and storage environments.
- Each domain uses **multiple monitoring tools**, leading to tool fragmentation and operational complexity.

Infra Component	Tech Stack	Monitoring Tools
Unix / Windows	Platforms	SCOM, Datadog, Compass, Nagios, Truesight, ServerSiter, iMon, EMAT
Oracle, MySQL, Hadoop, MongoDB	Database & Big Data	OEM, DataDog, Truesight, Cloudera, Splunk
NAS, SAN, Rubrik	Storage & Backup	Hitachi OpsCenter, Rubrik CDM/Netbackup, Pure1, RSE
VMware Hosting	Private Cloud	vROPS (vRealize Aria Operations)
IBM MQ, SWIFT	Middleware	TruSight, Datadog

- This **fragmented tooling environment** increases operational cost, complexity, and hampers efficient decision-making, particularly during critical incidents.

31



Business Challenges

- The current operational model faces the following challenges:
  1. Fragmented Monitoring & Lack of Observability
  2. Difficult to Consolidate Monitoring Tools
  3. Traditional Ops Mindset
  4. Lack of Standardized Toil Identification
  5. Limited Automation & IaC Adoption
  6. No Business-Aligned SLOs Defined
  7. Capacity Management Requires Improvement

32

Transitioning from Traditional ITOps to SRE Recommendations

33

Landscape

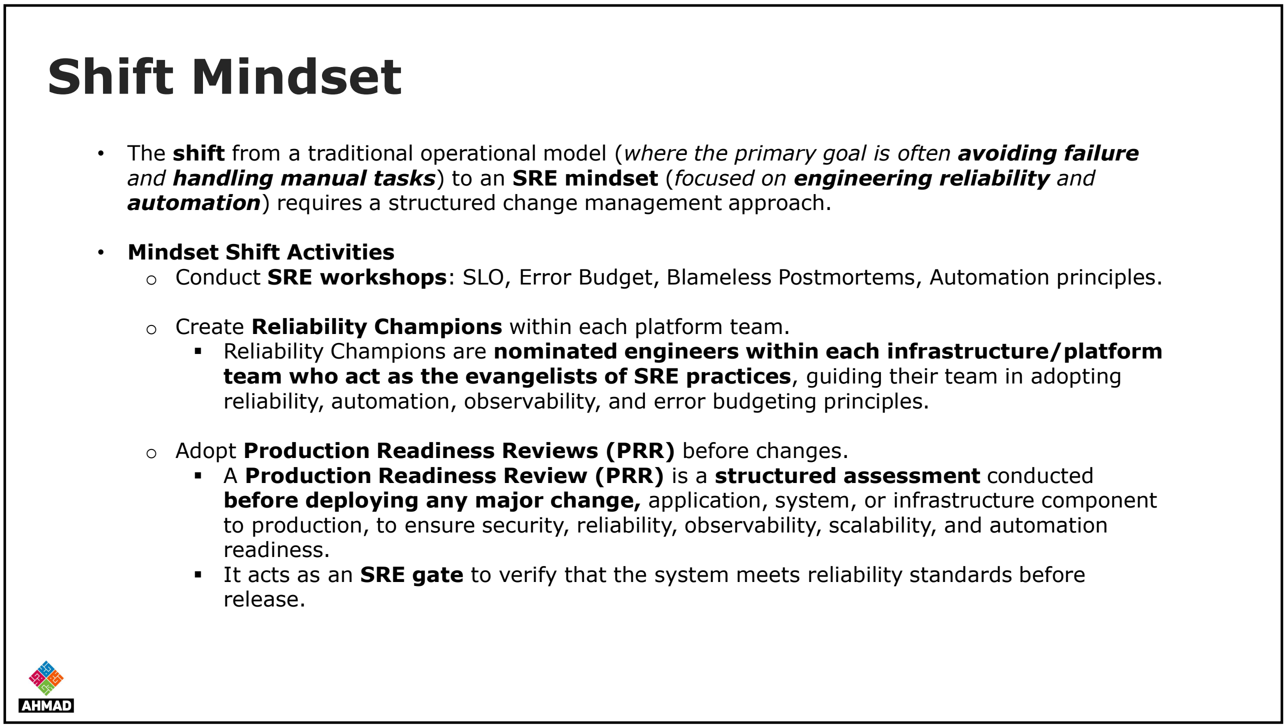
- Your infrastructure spans multiple layers (*Platform, DB, Storage, Cloud, Middleware*) and uses **multiple isolated monitoring tools**, with focus mainly on **reactive monitoring** and **availability management**.

Current State	Target SRE State
Tool-centric monitoring	**Service-centric observability**
Reactive incident handling	**Proactive** + **predictive** reliability management
Ticket-driven Ops	**Automation**, IaC, AI-driven insights
High manual toil	**Reduced Toil**, Continuous improvement
Infra availability focus	**Measure business reliability** via SLOs, SLIs

34

Shift Mindset

35



Shift Mindset

- The **shift** from a traditional operational model (*where the primary goal is often **avoiding failure** and **handling manual tasks***) to an **SRE mindset** (*focused on **engineering reliability** and **automation***) requires a structured change management approach.

- **Mindset Shift Activities**
  - Conduct **SRE workshops**: SLO, Error Budget, Blameless Postmortems, Automation principles.

  - Create **Reliability Champions** within each platform team.
    - Reliability Champions are **nominated engineers within each infrastructure/platform team who act as the evangelists of SRE practices**, guiding their team in adopting reliability, automation, observability, and error budgeting principles.

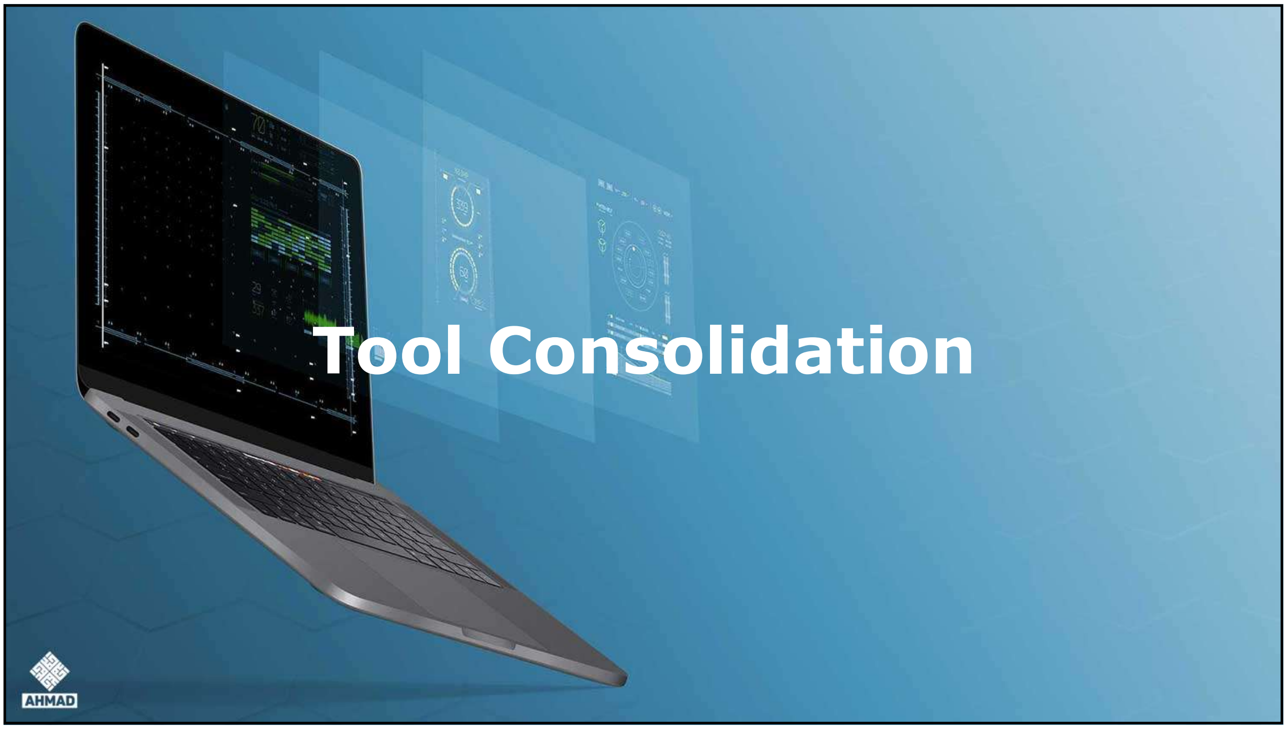
  - Adopt **Production Readiness Reviews (PRR)** before changes.
    - A **Production Readiness Review (PRR)** is a **structured assessment** conducted **before deploying any major change,** application, system, or infrastructure component to production, to ensure security, reliability, observability, scalability, and automation readiness.
    - It acts as an **SRE gate** to verify that the system meets reliability standards before release.

36

Shift Mindset … continue

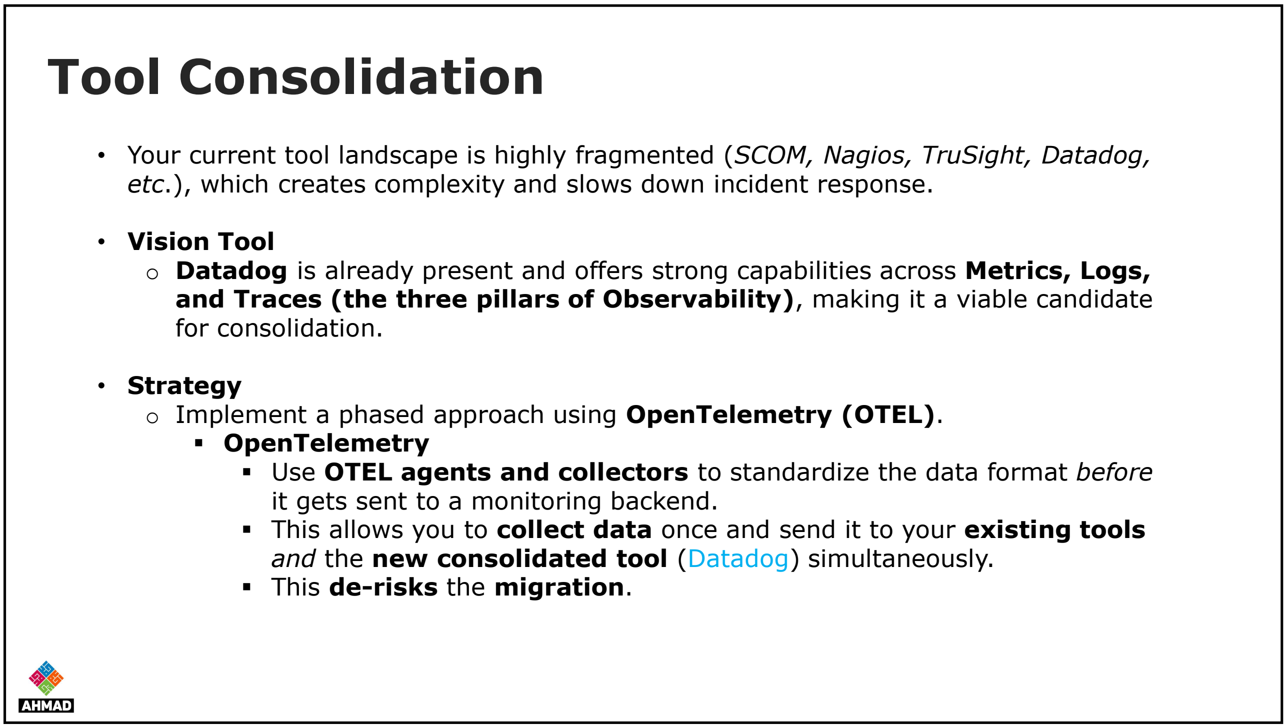
Traditional Mindset	SRE Mindset	Implementation Strategy
"Keep the Lights On"	**"Automate the Operations"**	• Introduce the **Error Budget** concept. • When the service is reliable enough (*error budget is high*), the team can ship features; when the error budget is depleted, they stop and focus on reliability work (***TOIL reduction, bug fixes***).
Manual Escalation/Fixes	**Eliminate Toil**	• Mandate that SREs spend **50% of their time** on engineering work (automation, new features, design). • Any operational work exceeding 50% must be automated or eliminated.
Blame-Oriented Reviews	**Blameless Postmortems**	• Establish a culture where incidents are viewed as **systemic failures**, not personal ones. • Focus on **what** happened, **why** it happened, and **how** to prevent recurrence, documenting actions in a **postmortem document**.

37



Tool Consolidation

38



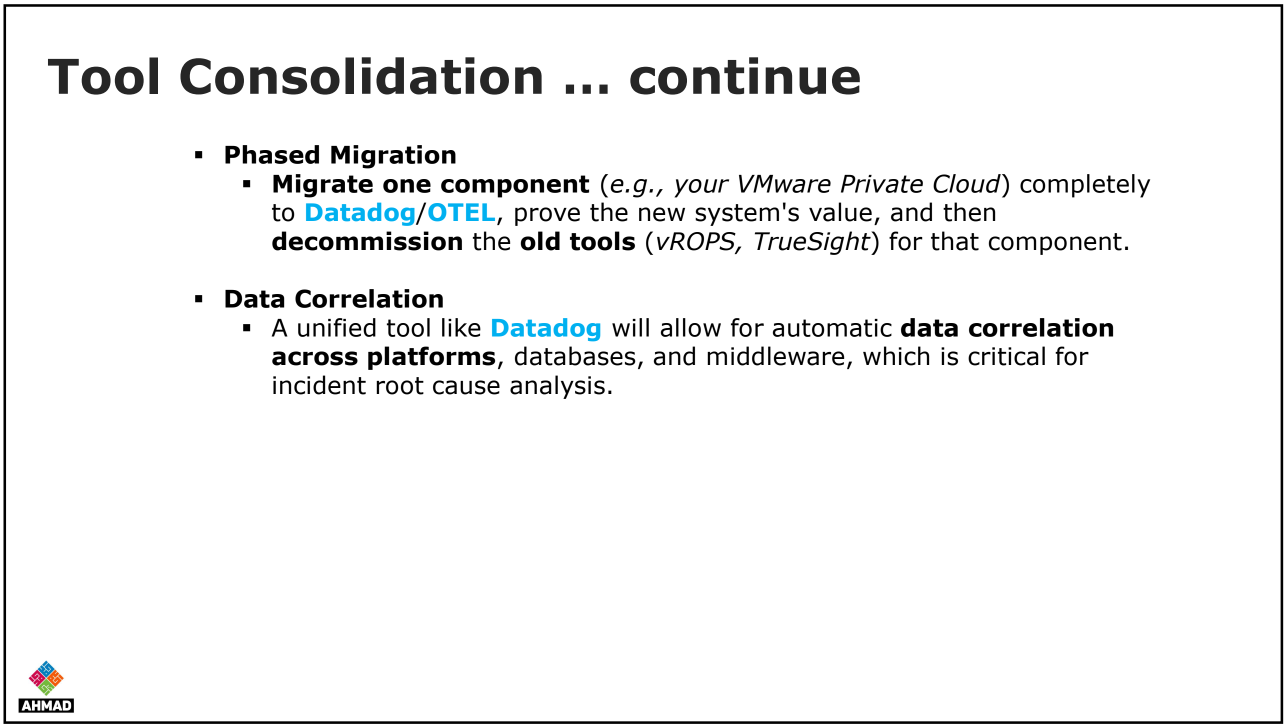
Tool Consolidation

- Your current tool landscape is highly fragmented (*SCOM, Nagios, TruSight, Datadog, etc.*), which creates complexity and slows down incident response.

- **Vision Tool**
  - **Datadog** is already present and offers strong capabilities across **Metrics, Logs, and Traces (the three pillars of Observability)**, making it a viable candidate for consolidation.

- **Strategy**
  - Implement a phased approach using **OpenTelemetry (OTEL)**.
    - **OpenTelemetry**
      - Use **OTEL agents and collectors** to standardize the data format *before* it gets sent to a monitoring backend.
      - This allows you to **collect data** once and send it to your **existing tools** *and* the **new consolidated tool** (Datadog) simultaneously.
      - This **de-risks** the **migration**.

39



Tool Consolidation … continue

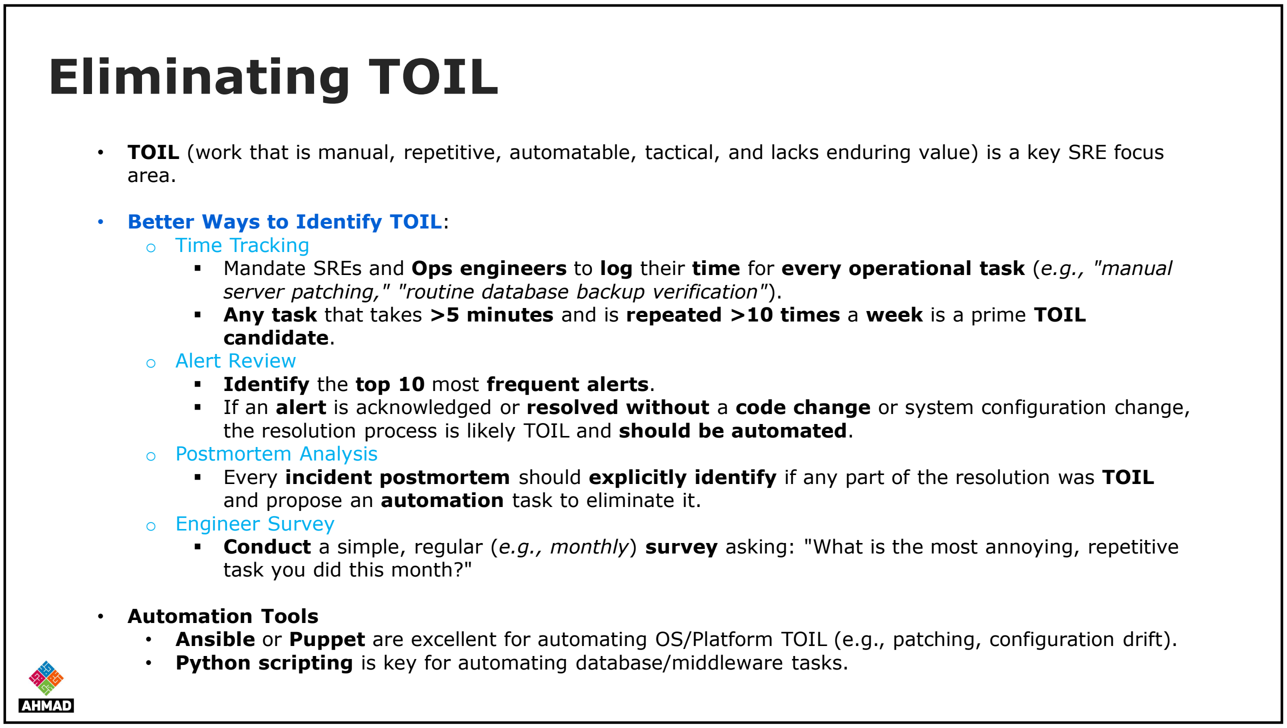
- **Phased Migration**
  - **Migrate one component** (*e.g., your VMware Private Cloud*) completely to **Datadog**/**OTEL**, prove the new system's value, and then **decommission** the **old tools** (*vROPS, TrueSight*) for that component.

- **Data Correlation**
  - A unified tool like **Datadog** will allow for automatic **data correlation across platforms**, databases, and middleware, which is critical for incident root cause analysis.

40

Eliminating TOIL

41



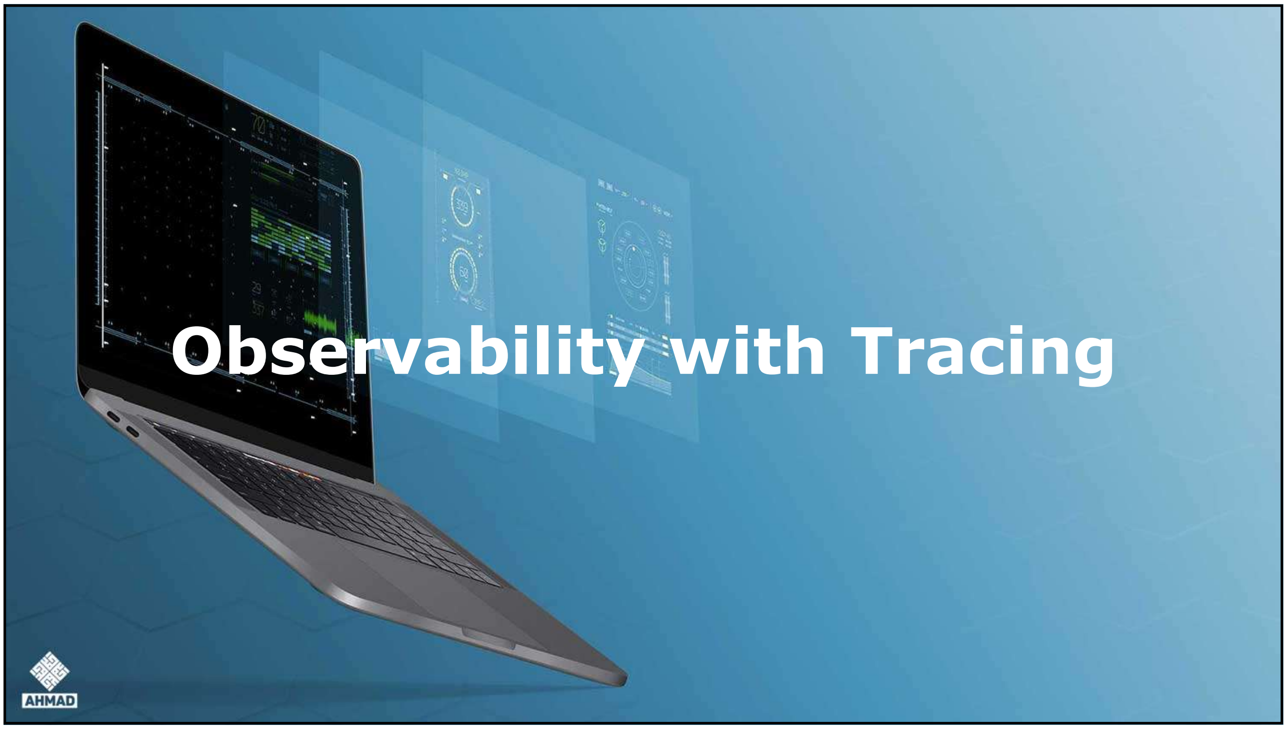
Eliminating TOIL

- **TOIL** (work that is manual, repetitive, automatable, tactical, and lacks enduring value) is a key SRE focus area.

- **Better Ways to Identify TOIL**:
  - Time Tracking
    - Mandate SREs and **Ops engineers** to **log** their **time** for **every operational task** (*e.g., "manual server patching," "routine database backup verification"*).
    - **Any task** that takes **>5 minutes** and is **repeated >10 times** a **week** is a prime **TOIL candidate**.
  - Alert Review
    - **Identify** the **top 10** most **frequent alerts**.
    - If an **alert** is acknowledged or **resolved without** a **code change** or system configuration change, the resolution process is likely TOIL and **should be automated**.
  - Postmortem Analysis
    - Every **incident postmortem** should **explicitly identify** if any part of the resolution was **TOIL** and propose an **automation** task to eliminate it.
  - Engineer Survey
    - **Conduct** a simple, regular (*e.g., monthly*) **survey** asking: "What is the most annoying, repetitive task you did this month?"

- **Automation Tools**
  - **Ansible** or **Puppet** are excellent for automating OS/Platform TOIL (e.g., patching, configuration drift).
  - **Python scripting** is key for automating database/middleware tasks.

42



Observability with Tracing

AHMAD

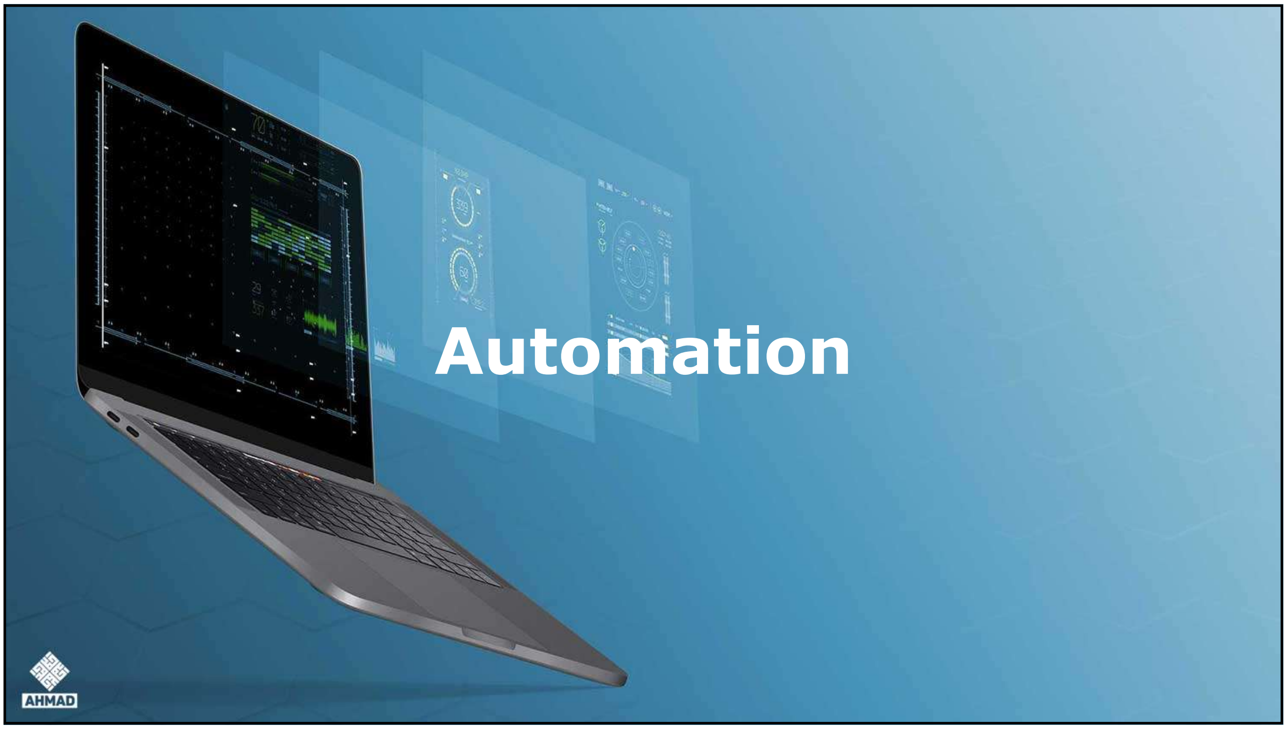
43

Observability with Tracing

- You are currently monitoring Metrics (e.g., CPU, Memory) and Logs (e.g., application error messages).
- To **achieve end-to-end observability**, you **need Traces**.
- Implementation Example: **Tracing Middleware/Database Calls**
  - **Goal**
    - **Track** a **transaction's journey** from the platform (*Unix/Windows*) through the middleware (*IBM MQ*) to the database (*Oracle/MySQL*).
  - **Tool**
    - **Implement OpenTelemetry** (*OTEL*) or **use Datadog APM** (*Application Performance Monitoring*).
- **Benefit**
  - If a **transaction takes 10 seconds**, a trace will **show** exactly **how much time** was **spent** in the **application**, how long the **message** sat in the **MQ queue**, and **how long** the **database query took** (e.g., $9$ seconds spent in the database).
  - This pinpoints the **performance bottleneck immediately**.

AHMAD

44



Automation

45

Automation

- IaC is fundamental to SRE, reducing TOIL and ensuring reproducibility.

- Recommended **IaC Tools**:
  - **Platform (OS/VMware)**
    - **Terraform** for **provisioning** and **managing** your VMware Cloud infrastructure (VMs, networks, storage).
    - This is your primary IaC tool.

  - **Configuration Management**
    - A**nsible** or **Puppet** for c**onfiguring** the OS (*Unix/Windows*), **installing** necessary packages, and hardening the platforms.

AHMAD

46

Automation … continue

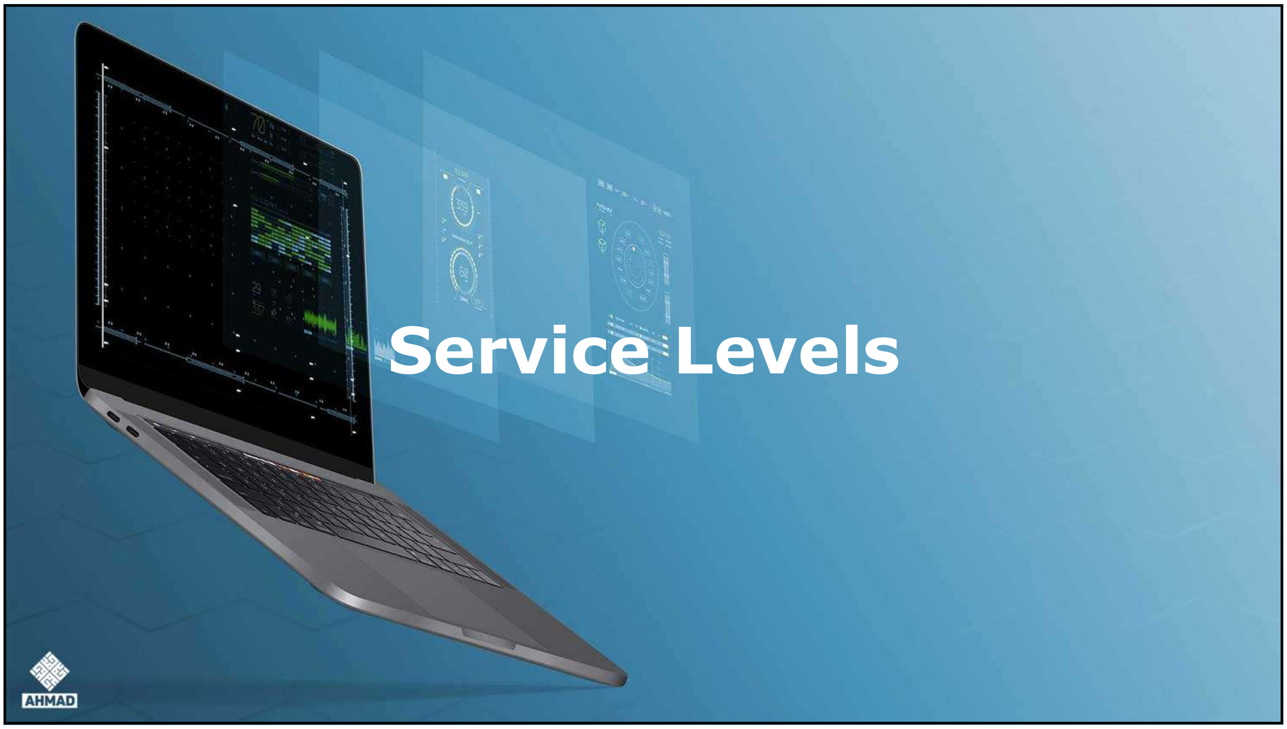
- **Automation Opportunities**
  - **Self-Healing/Auto-Remediation**
    - **Automate** responses to **common alerts** (*e.g., if a server's disk utilization exceeds 90%, automatically run a script to clear temporary files and alert the team*).
  - **Environment Provisioning**
    - **Automate** the end-to-end build of a **new environment** *(Platform, Middleware, Database*) using **Terraform** and **Ansible/Puppet**.
  - **AIOps Integration**
    - Use your new **AIOps tool** (*BigPanda*) to **ingest alerts** from all your remaining monitoring tools and **automatically trigger** these Ansible/Puppet automation **playbooks** for **remediation**.

BigPanda

BigPanda

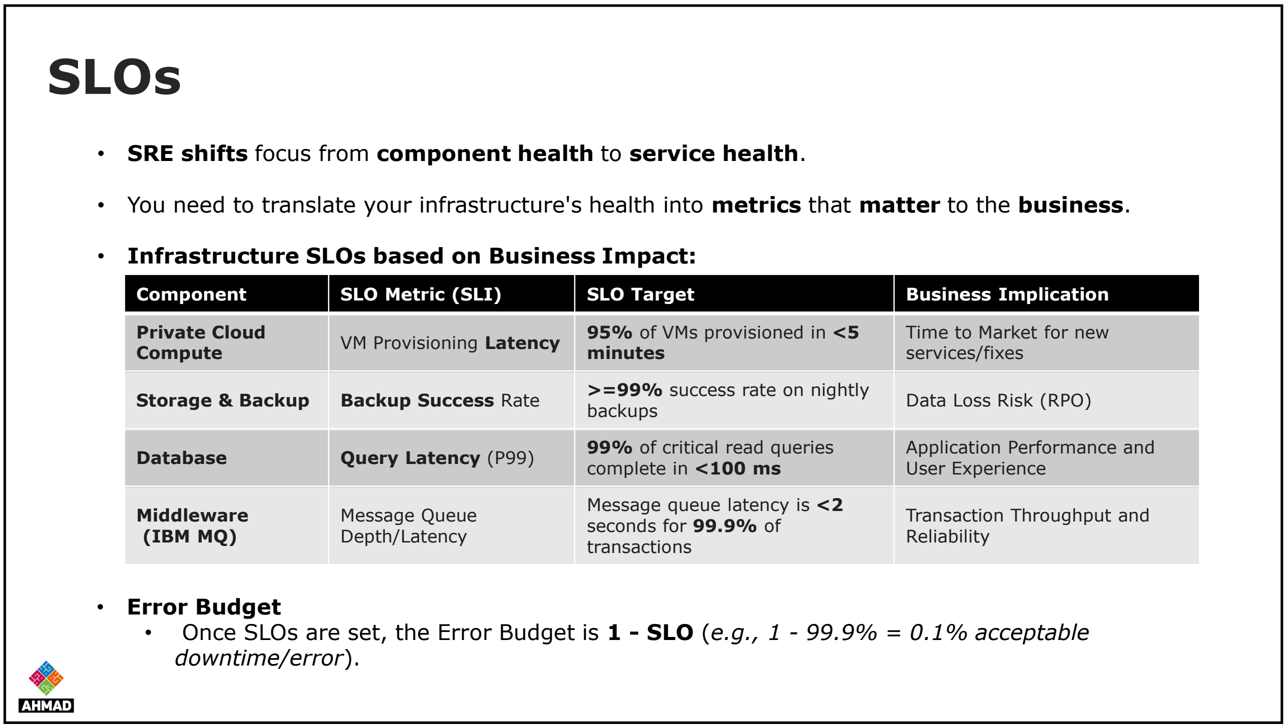
- **BigPanda** acts as an **Event Correlation Engine** and **Noise Reduction tool**.
- **Role of BigPanda:**
  1. **Ingestion**
     - **Ingest alerts** and **data** from all your **disparate tools** (*SCOM, Nagios, vROPS, Datadog*).
  2. **Correlation**
     - **Use Machine Learning** to **group related alerts** into a single **Incident**.
     - For example, a single network switch failure might trigger 50 alerts in Nagios, vROPS, and SCOM; BigPanda should consolidate this into **one incident**.
  3. **Automation Trigger**
     - The consolidated incident in BigPanda should be used as the **single source of truth** to:
       - **Automatically open a ticket** in your **ITSM** system (e.g., ServiceNow).
       - Automatically trigger the necessary **Ansible/Puppet automation playbook** for **auto-remediation**.

49



Service Levels

50



SLOs

- **SRE shifts** focus from **component health** to **service health**.
- You need to translate your infrastructure's health into **metrics** that **matter** to the **business**.
- **Infrastructure SLOs based on Business Impact:**

Component	SLO Metric (SLI)	SLO Target	Business Implication
Private Cloud Compute	VM Provisioning **Latency**	**95%** of VMs provisioned in **<5 minutes**	Time to Market for new services/fixes
Storage & Backup	**Backup Success** Rate	**>=99%** success rate on nightly backups	Data Loss Risk (RPO)
Database	**Query Latency** (P99)	**99%** of critical read queries complete in **<100 ms**	Application Performance and User Experience
Middleware (IBM MQ)	Message Queue Depth/Latency	Message queue latency is **<2** seconds for **99.9%** of transactions	Transaction Throughput and Reliability

- **Error Budget**
  - Once SLOs are set, the Error Budget is **1 - SLO** (*e.g., 1 - 99.9% = 0.1% acceptable downtime/error*).

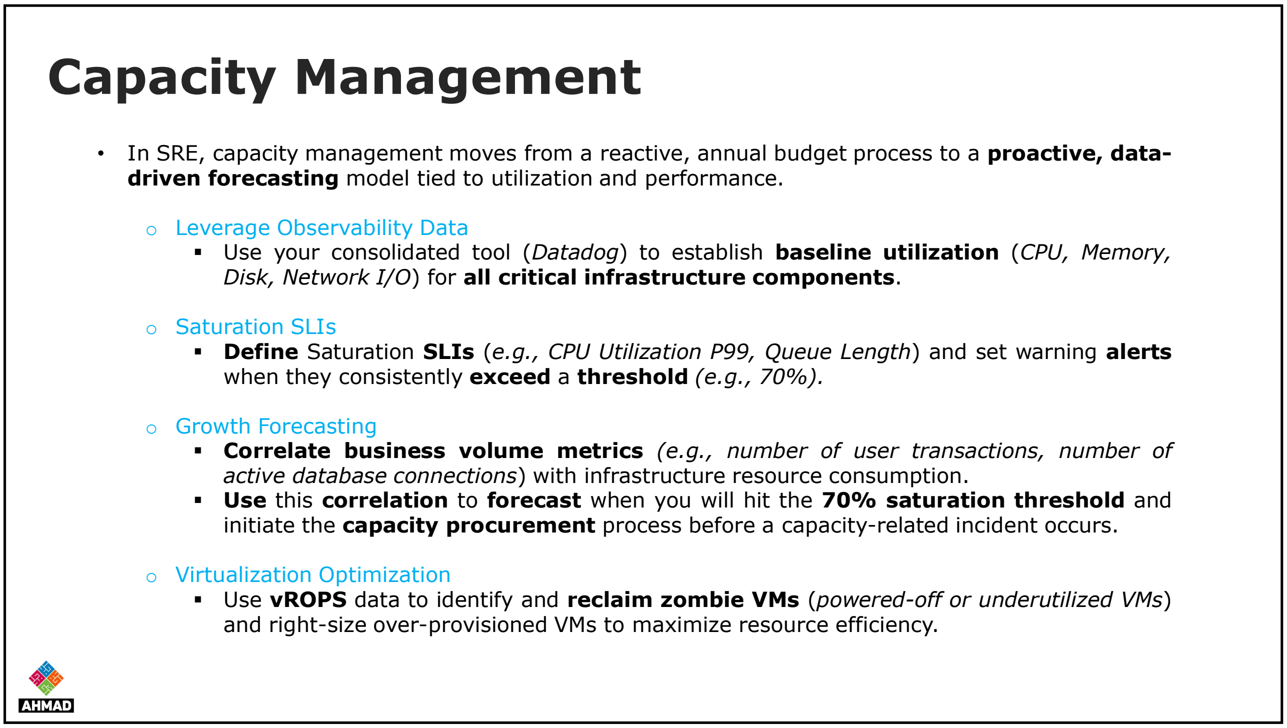
AHMAD

51

Capacity Management

AHMAD

52

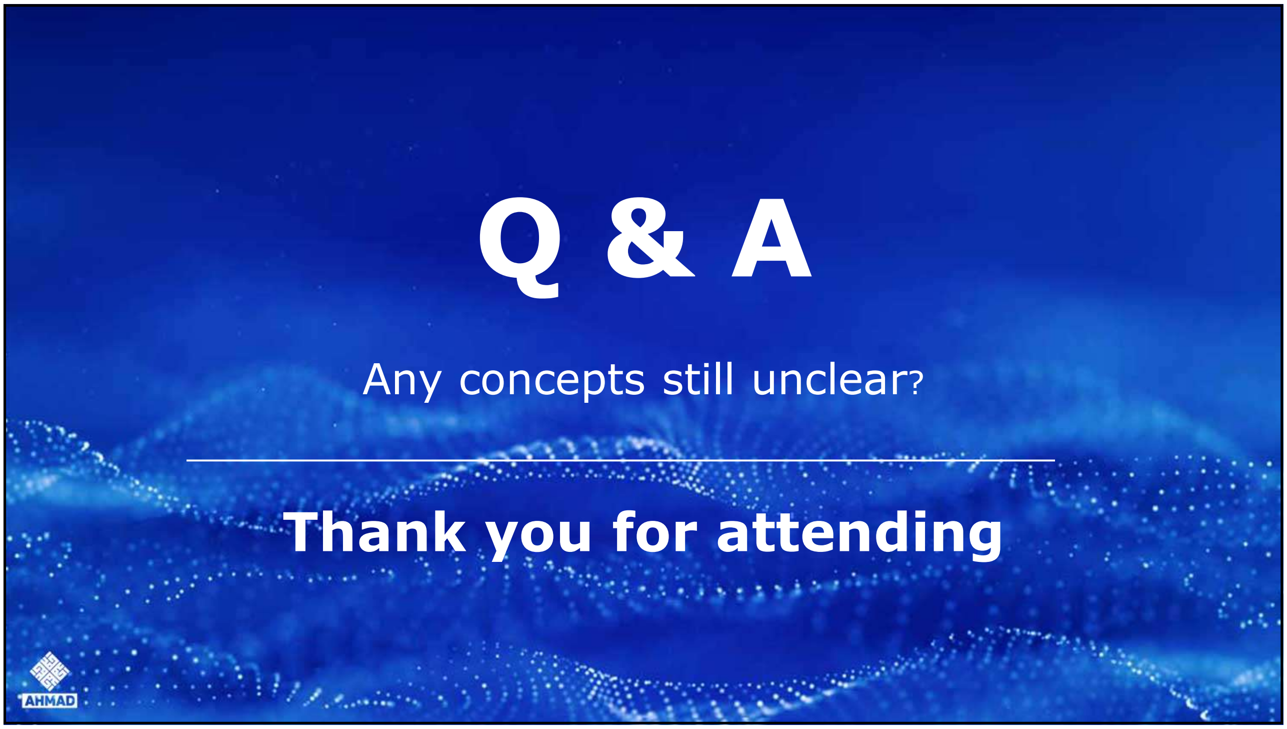


Capacity Management

- In SRE, capacity management moves from a reactive, annual budget process to a **proactive, data-driven forecasting** model tied to utilization and performance.
  - Leverage Observability Data
    - Use your consolidated tool (*Datadog*) to establish **baseline utilization** (*CPU, Memory, Disk, Network I/O*) for **all critical infrastructure components**.
  - Saturation SLIs
    - **Define** Saturation **SLIs** (*e.g., CPU Utilization P99, Queue Length*) and set warning **alerts** when they consistently **exceed** a **threshold** *(e.g., 70%).*
  - Growth Forecasting
    - **Correlate business volume metrics** *(e.g., number of user transactions, number of active database connections*) with infrastructure resource consumption.
    - **Use** this **correlation** to **forecast** when you will hit the **70% saturation threshold** and initiate the **capacity procurement** process before a capacity-related incident occurs.
  - Virtualization Optimization
    - Use **vROPS** data to identify and **reclaim zombie VMs** (*powered-off or underutilized VMs*) and right-size over-provisioned VMs to maximize resource efficiency.

AHMAD

53



Q & A

Any concepts still unclear?

Thank you for attending

AHMAD

54