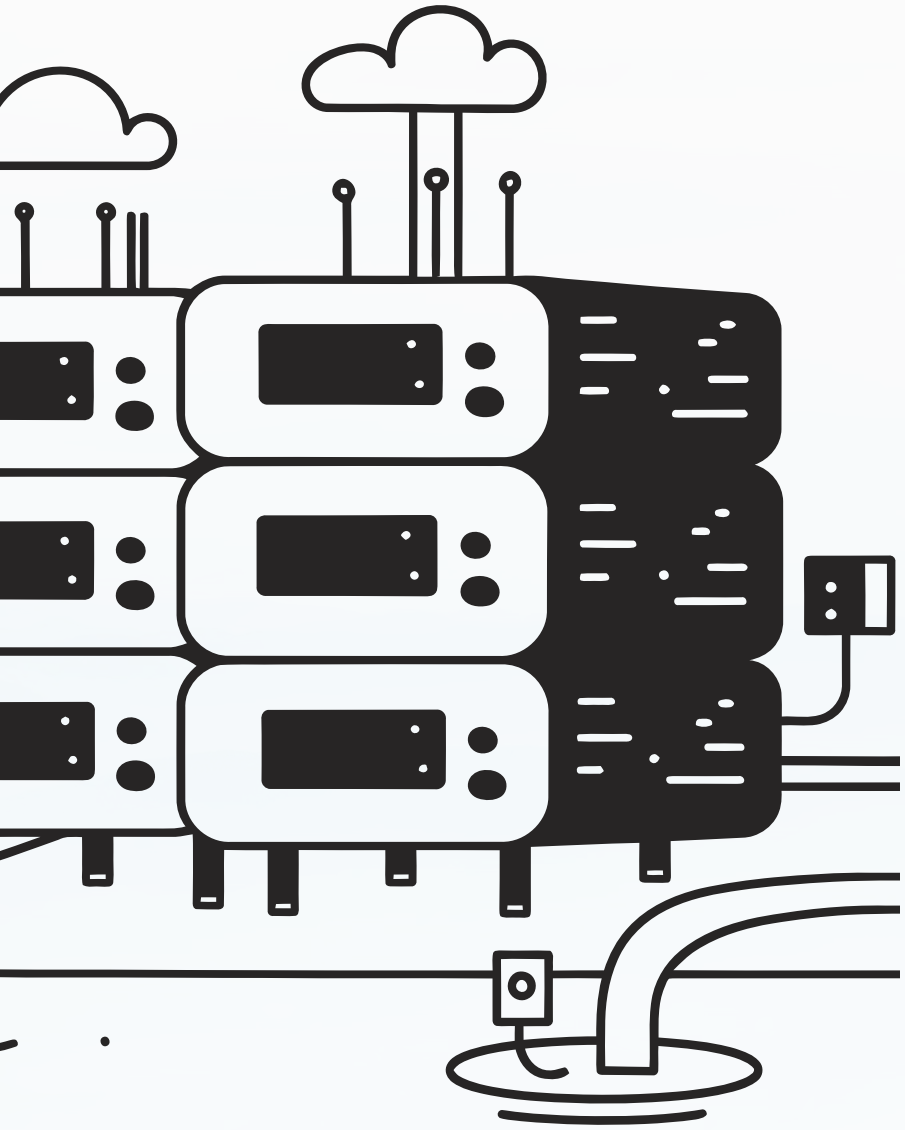


# Why Monitoring Matters

Monitoring is the cornerstone of reliable, predictable systems. It empowers teams to detect issues *before* users experience them, maintaining trust and service quality.

### System Health

Answer critical questions: "Is my application healthy?" and "Why is performance degrading?"

### Proactive Detection

Identify anomalies and bottlenecks before they cascade into user-facing failures

### Core Principles

Built on four pillars: **Metrics**, **Logs**, **Traces**, and Alerts

# Observability vs Monitoring

Whilst monitoring tracks **known metrics** against predefined thresholds, observability enables teams to explore and diagnose **unknown issues** through dynamic querying and correlation.

> Observability = The ability to ask *new* questions about your system without predicting them in advance.

---

| Concept | Purpose | Example Tools |
| --- | --- | --- |
| Monitoring | Tracks known metrics | CloudWatch, Prometheus |
| Observability | Explores unknown issues | Grafana, New Relic |

# Key Metrics to Watch

Effective monitoring isn't about collecting everything—it's about tracking what truly matters to system health and business outcomes.

### Infrastructure

Monitor foundational resources: CPU utilisation, memory consumption, disk I/O, and network throughput

### Application

Track performance indicators: response time, error rate, request throughput, and service availability

### Business

Measure outcomes that matter: user signups, revenue generation, active users, and conversion rates

**Remember:** Monitoring ≠ collecting everything. Focus on actionable metrics that drive decisions and prevent incidents.

# AWS CloudWatch Overview

CloudWatch is AWS's native monitoring and observability service, providing comprehensive visibility across your entire AWS infrastructure and applications.

### Application

Generates logs and metrics

### Alarms

Trigger on thresholds

### CloudWatch Logs

Aggregates and stores data

### SNS Notification

Alert teams instantly

### Metrics Collection

Automatically tracks EC2, Lambda, RDS, and 70+ AWS services

### Log Management

Centralised log aggregation with query and analysis capabilities

### Visualisation

Custom dashboards and alarms for proactive monitoring

# Prometheus & Grafana Stack

The Prometheus and Grafana combination has become the de facto standard for monitoring cloud-native applications, particularly in Kubernetes and microservice environments.

> "Prometheus collects, Grafana displays."

## Prometheus

Time-series database that scrapes metrics from instrumented endpoints using a pull model

- Multi-dimensional data model
- Powerful query language (PromQL)
- Service discovery integration

## Grafana

Visualisation and analytics platform that transforms raw metrics into actionable dashboards

- Rich, customisable dashboards
- Supports multiple data sources
- Alerting and notification system

# Alerting in Practice

Effective alerting transforms metrics into actionable insights, triggering notifications when critical thresholds are breached.

1

### CPU Threshold Alert

If CPU utilisation exceeds 80% for 5 consecutive minutes → Send alert to on-call engineer

2

### Response Time Alert

When application response time surpasses 2 seconds → Notify Slack channel for immediate investigation

3

### Error Rate Spike

Error rate above 5% for 3 minutes → Page incident response team via PagerDuty

---

**Tip:** Avoid alert fatigue. Only alert on actionable items that require human intervention. Use severity levels and smart routing to ensure the right people receive the right alerts.

# Cloud in DevOps

### Scalability

Elastically scale infrastructure up or down based on demand, paying only for what you use

### Automation

Infrastructure as Code and managed services eliminate manual configuration and reduce human error

### Speed

Deploy environments in minutes rather than weeks, accelerating development cycles

> Cloud = foundation for continuous delivery at scale. AWS, Azure, and GCP provide the building blocks for modern DevOps practices.

# AWS Core Services for DevOps

AWS offers a comprehensive suite of services that form the backbone of DevOps workflows, from compute to infrastructure management.

| Category | Service | Use Case |
| --- | --- | --- |
| Compute | EC2 | Run virtual servers for applications and workloads |
| Storage | S3 | Store build artifacts, backups, and static assets |
| Identity | IAM | Manage access controls and security policies |
| Infrastructure Mgmt | CloudFormation | Infrastructure as Code for AWS resource provisioning |
| CI/CD | CodePipeline | Automate build, test, and deployment workflows |
| Container Orchestration | EKS | Managed Kubernetes for containerised applications |

# DevSecOps – Adding Security

DevSecOps integrates security practices into every stage of the development lifecycle, transforming security from a gate at the end into a continuous process.

### Shift Security Left

Embed security early in the pipeline, not as an afterthought before production

### Automate Everything

Implement automated vulnerability checks, dependency scans, and compliance validation

### Access Control

Enforce least-privilege access and role-based permissions across all environments

### Continuous Monitoring

Monitor for security threats and compliance violations in real-time

# Why DevSecOps is Critical

**90%**

### Known Vulnerabilities

Breaches exploit known misconfigurations or unpatched systems

**76%**

### Faster Detection

Automated security reduces time to identify vulnerabilities

**3x**

### Cost Savings

Finding issues early costs significantly less than post-production fixes

CI/CD pipelines should enforce security policies automatically at every stage, preventing vulnerabilities from reaching production.

### SAST

Static Application Security Testing analyses source code for vulnerabilities before compilation

### DAST

Dynamic Application Security Testing tests running applications for runtime vulnerabilities

### Secrets Scanning

Automatically detect API keys, passwords, and credentials committed to repositories

# Monitoring Workflow

1. **Collect:** Gather metrics, logs, and traces from all services.
2. **Store:** Push them to a centralized service like CloudWatch or Prometheus.
3. **Visualize:** Use Grafana or AWS Dashboards for real-time charts.
4. **Alert:** Configure alerts via CloudWatch Alarms or Alertmanager.
5. **Act:** Trigger Lambda functions or Slack notifications to auto-heal or notify.

> Goal: Continuous visibility → faster incident response → higher uptime.

# Setting up CloudWatch

1. **Go to AWS Console** → Navigate to the CloudWatch service.
2. **Choose “Metrics”** → Select a service (e.g., EC2) → Choose a specific metric (e.g., CPUUtilization).
3. **Click “Create Alarm”** → Define the threshold (e.g., >80% for 5 minutes) and conditions.
4. **Create an SNS topic** → Add your email, Slack webhook, or other endpoints for notifications.
5. **Monitor** → The alarm will automatically trigger and notify when the threshold is breached.

---

**Tip:** Use CloudWatch Dashboards to visualise multiple service metrics and alarms in a single, unified view for better operational oversight.

# Prometheus Setup (Conceptual)

### Install Prometheus

Deploy Prometheus on your dedicated monitoring server or within your Kubernetes cluster.

### Configure Targets

Edit the `prometheus.yml` configuration file to specify the endpoints (targets) that expose `/metrics` data.

### Integrate Grafana

Connect your Grafana instance by adding Prometheus as a new data source.

### Build Dashboards

Design custom dashboard panels in Grafana to visualise critical metrics like CPU, memory, and specific application data.

## Common Exporters for Prometheus

Exporters convert metrics from various systems into a format Prometheus can ingest.

### Node Exporter

Collects comprehensive system-level metrics (CPU, memory, disk I/O, network) from Linux servers.

### cAdvisor

Provides container resource usage and performance metrics, essential for Docker and Kubernetes environments.

### Blackbox Exporter

Probes endpoints over various protocols (HTTP, HTTPS, TCP, ICMP) to check their availability and response time.

# AWS in DevOps Lifecycle

| Phase | AWS Service | Role |
| --- | --- | --- |
| Code | CodeCommit | Source control |
| Build | CodeBuild | CI builds |
| Deploy | CodeDeploy | Rolling deployments |
| Monitor | CloudWatch | Logs & metrics |
| Secure | IAM, GuardDuty | Access & threat detection |

> AWS provides a complete DevOps ecosystem when combined with Jenkins or GitHub Actions.

# DevSecOps Pipeline Example

### Commit Stage

Code scanned with tools like SonarQube or Trivy for vulnerabilities and code quality issues.

### Build Stage

Docker images and dependencies checked with Aqua Security or Anchore for known vulnerabilities.

### Deploy Stage

Policy enforcement using Open Policy Agent (OPA) to validate configurations and ensure compliance before deployment.

### Runtime Stage

Continuous monitoring of container behavior and network anomalies to detect threats in production environments.

Security shouldn’t slow you down — automate it by integrating checks and policies directly into your CI/CD pipeline.

# Monitoring in the Real World

In a production-grade DevOps setup:

- Application metrics are continuously streamed to a monitoring service (CloudWatch or Prometheus).
- Dashboards visualize latency, error rates, and system health.
- Alerts trigger automated remediation (restart pods, scale up EC2, etc.).
- Logs are centralized using tools like **ELK Stack** or **CloudWatch Logs**.

The goal isn’t just to monitor — it’s to **predict** failures before they impact customers.

# Integrating AWS Monitoring with CI/CD

Monitoring should begin the moment you deploy. Integrating AWS monitoring directly into your CI/CD pipelines ensures immediate visibility and rapid response to any issues.

### Automated Alarm Registration

Pipelines automatically register CloudWatch alarms for new deployments or resources.

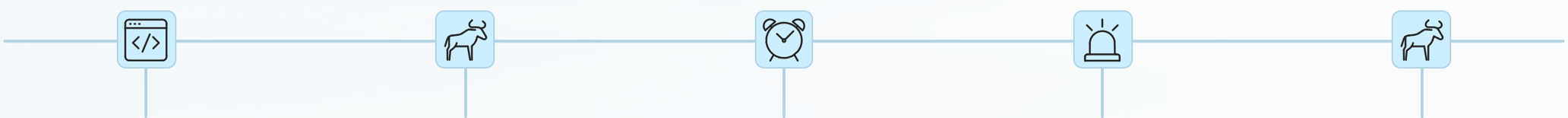
### Custom Log Ingestion

Deployment scripts push custom application logs and metrics directly to CloudWatch.

### Proactive Incident Response

Alarms trigger automated rollback jobs, notify DevOps teams, or initiate auto-healing actions.

## Example: End-to-End Monitoring Workflow



### Jenkins Pipeline

Initiates deployment to AWS resources.

### AWS CLI

Used to configure CloudWatch alarms and push logs.

### CloudWatch Alarm

Monitors critical metrics and triggers on thresholds.

### SNS Alert

Publishes messages to subscribed endpoints.

### Slack Notification

Informs the DevOps team of critical events.

# Security Integration in Pipelines

Modern CI/CD pipelines must embed security gates:

- **Pre-build:** Dependency and license checks (npm audit, pip-audit).
- **Post-build:** Image scanning for vulnerabilities (Trivy, Clair).
- **Pre-deploy:** Secrets detection using Gitleaks.
- **Runtime:** Container behavior monitored using Falco or GuardDuty.

This ensures that even automated deployments stay compliant and secure.

# Cloud Security Fundamentals

Every AWS environment should follow baseline security:

- Enable **IAM Roles** with least privilege access.
- Use **Security Groups** to control inbound/outbound traffic.
- Enforce **MFA** for root accounts.
- Use **AWS Config** and **CloudTrail** for continuous auditing.

A good DevOps team automates these configurations using Terraform or CloudFormation templates.

# Congratulations!

You've successfully completed the DevOps and Monitoring training. Let's recap the key areas we covered today:

## Monitoring & Observability

Understood monitoring and observability fundamentals, and their critical importance in modern software delivery.

## Tooling & Workflows

Explored the practical workflows of AWS CloudWatch, and the Prometheus & Grafana open-source stack.

## AWS Services for DevOps

Reviewed core AWS services like EC2, S3, and IAM that are crucial for building and operating DevOps pipelines.

## DevSecOps Integration

Learned the basics of integrating security practices directly into your CI/CD pipelines, adopting a DevSecOps approach.

Keep building on this comprehensive knowledge to create more resilient, efficient, and secure systems in your future projects!