Monitoring

TEKsystems Global Services

WORKFORCE DEVELOPMENT

Advanced Dashboarding Overview

- You will build a production-grade Grafana dashboard focused on service health.
- The dashboard must support on-call decision making using real metrics and logs.
  - Grafana dashboards
  - Prometheus metrics
  - Loki logs
  - SLO-driven views

Service Scenario

- Your team owns an internal API used by multiple systems.
- When issues occur, dashboards must provide fast answers.
  - Shared dependency
  - High impact
  - On-call usage
  - Operational focus

Service Level Objectives

- SLOs define what success means for the service.
- Dashboards must make SLO violations obvious.
  - Availability ≥ 99%
  - p95 latency ≤ 500ms
  - Measured continuously
  - Visible in Grafana

Operational Questions

- Every panel must answer a real operational question.
- Charts without purpose add noise.
  - Is it up?
  - Is it fast?
  - Is it scaling?
  - Can I debug it?

Community Dashboards

- Grafana provides many community-built dashboards.
- These are good sources of patterns and ideas.
  - grafana.com
  - Import by ID
  - Editable panels
  - Reusable designs

Importing Dashboards

- Imported dashboards are starting points, not final solutions.
- You can copy panels into your own dashboard.
  - Paste dashboard ID
  - Select data source
  - Explore queries
  - Reuse panels

Creating a New Dashboard

- You will build a custom dashboard tailored to your service.
- This ensures metrics match your SLOs.
  - New dashboard
  - Custom panels
  - Service-specific queries
  - Clear layout

Logs Panel

- Logs provide context when metrics show anomalies.
- Loki allows real-time log searching.
  - Logs panel
  - Search variable
  - Filter by app
  - Time correlation

Replica Count Panel

- Replica count shows scaling and stability.
- Unexpected drops indicate failures.
  - Stat or Gauge
  - Available replicas
  - Expected vs actual
  - Autoscaling insight

Latency Panel

- Latency reflects user experience.
- Percentiles capture worst-case behavior.
  - Time series
  - p95 latency
  - histogram_quantile
  - Trend analysis

Availability Panel

- Availability measures successful request rates.
- It directly reflects the SLO.
  - Success ratio
  - Exclude 5xx
  - Stat panel
  - Thresholds

Error Rate Panel

- Errors highlight failures users experience.
- Tracking errors separately improves detection.
  - 5xx responses
  - Rate over time
  - Alert candidate
  - Incident signal

Throughput Panel

- Throughput shows system load.
- It helps explain latency and errors.
  - Requests per second
  - Traffic spikes
  - Correlation
  - Capacity insight

Dashboard Variables

- Variables make dashboards reusable.
- They reduce duplication across environments.
  - Namespace variable
  - Deployment variable
  - Log filters
  - Dynamic queries

Metrics and Logs Together

- Metrics show what is happening.
- Logs explain why it is happening.
  - Start with metrics
  - Jump to logs
  - Time alignment
  - Root cause

Dashboard Polish

- Clear labeling improves usability.
- Dashboards are shared artifacts.
  - Titles
  - Descriptions
  - Panel grouping
  - Consistent naming

Exporting Dashboards

- Dashboards can be versioned as JSON.
- This supports GitOps workflows.
  - Export JSON
  - Store in Git
  - Re-import
  - Share easily

Sharing Dashboards

- Dashboards can be shared internally or publicly.
- Standardization improves operations.
  - Share links
  - Export files
  - grafana.com
  - Team reuse

Cluster Dashboard

- Application dashboards are not enough.
- Cluster dashboards show platform health.
  - Node metrics
  - Pod counts
  - Infra errors
  - Capacity overview

On-Call Workflow

- Dashboards guide incident response.
- They should support fast decisions.
  - Check availability
  - Inspect latency
  - Review errors
  - Debug via logs

MCQ 1 – Purpose of Dashboard

- What is the main goal of the dashboard?
  - A. Store logs
  - B. Replace alerts
  - C. Evaluate service health
  - D. Monitor only infra

MCQ 1 – Purpose of Dashboard – Answer

- What is the main goal of the dashboard?
  - A. Store logs
  - B. Replace alerts
  - **C. Evaluate service health**
  - D. Monitor only infra

MCQ 2 – SLO

- What does an SLO represent?
  - A. Alert rule
  - B. Business target
  - C. Log format
  - D. Metric type

MCQ 2 – SLO – Answer

- What does an SLO represent?
  - A. Alert rule
  - **B. Business target**
  - C. Log format
  - D. Metric type

MCQ 3 – p95

- What does p95 latency show?
  - A. Average
  - B. Fastest
  - C. Worst 5%
  - D. Total

MCQ 3 – p95 – Answer

- What does p95 latency show?
  - A. Average
  - B. Fastest
  - **C. Worst 5%**
  - D. Total

MCQ 4 – Logs

- Why use logs?
  - A. Replace metrics
  - B. Explain failures
  - C. Store dashboards
  - D. Generate alerts

MCQ 4 – Logs – Answer

- Why use logs?
  - A. Replace metrics
  - **B. Explain failures**
  - C. Store dashboards
  - D. Generate alerts

MCQ 5 – Variables

- Why use variables?
  - A. Faster queries
  - B. Reduce cost
  - C. Reusable dashboards
  - D. Replace PromQL

MCQ 5 – Variables – Answer

- Why use variables?
  - A. Faster queries
  - B. Reduce cost
  - **C. Reusable dashboards**
  - D. Replace PromQL

Alerting Overview

- Alerts notify engineers when action is required.
- They complement dashboards.
  - Grafana alerting
  - Prometheus metrics
  - Slack notifications
  - On-call workflows

Why Alerting Matters

- Dashboards require manual checking.
- Alerts automate detection.
  - Faster response
  - Reduced downtime
  - Clear signals
  - Operational readiness

Forking the Repo

- You will work in your own fork.
- Git tracks all changes.
  - Fork repo
  - Clone locally
  - Safe experimentation
  - GitOps flow

Slack Webhooks

- Slack receives alert notifications.
- Webhooks enable integration.
  - Slack app
  - Incoming webhook
  - Channel selection
  - Secure URL

MySQL Availability Metric

- mysql_up indicates database health.
- It drives the alert.
  - 1 = up
  - 0 = down
  - Prometheus metric
  - Binary signal

Creating an Alert Rule

- Alerts are defined on panels.
- Rules define failure conditions.
  - mysql_up == 0
  - Evaluate every minute
  - For 1 minute
  - Treat no data as alert

Contact Points

- Contact points define where alerts go.
- Slack is used here.
  - Add contact point
  - Slack type
  - Webhook URL
  - Attach to rule

Simulating Failure

- GitOps changes simulate outages.
- ArgoCD enforces desired state.
  - Remove deployment
  - Commit change
  - ArgoCD prunes
  - Failure occurs

Alert Firing

- Grafana detects the failure.
- Slack receives the alert.
  - Slack message
  - Rule validation
  - End-to-end test
  - Incident flow

Restoring Service

- Restoring Git restores the service.
- Alerts automatically resolve.
  - Revert change
  - ArgoCD redeploys
  - mysql_up returns
  - Alert clears

MCQ 1 – Alerting

- What is alerting for?
  - A. Visualize logs
  - B. Notify engineers
  - C. Store metrics
  - D. Replace dashboards

MCQ 1 – Alerting – Answer

- What is alerting for?
  - A. Visualize logs
  - **B. Notify engineers**
  - C. Store metrics
  - D. Replace dashboards

MCQ 2 – Webhook

- Slack webhook does what?
  - A. Scrapes metrics
  - B. Sends alerts
  - C. Deploys apps
  - D. Stores logs

MCQ 2 – Webhook – Answer

- Slack webhook does what?
  - A. Scrapes metrics
  - **B. Sends alerts**
  - C. Deploys apps
  - D. Stores logs

MCQ 3 – mysql_up

- mysql_up value 0 means?
  - A. Slow
  - B. Down
  - C. Healthy
  - D. Restart

MCQ 3 – mysql_up – Answer

- mysql_up value 0 means?
  - A. Slow
  - **B. Down**
  - C. Healthy
  - D. Restart

MCQ 4 – ArgoCD

- Removing manifest does what?
  - A. Nothing
  - B. Deletes pod
  - C. Prunes resource
  - D. Stops Prometheus

MCQ 4 – ArgoCD – Answer

- Removing manifest does what?
  - A. Nothing
  - B. Deletes pod
  - **C. Prunes resource**
  - D. Stops Prometheus

MCQ 5 – Resolve

- When does alert resolve?
  - A. Slack restart
  - B. Grafana restart
  - C. MySQL restored
  - D. Prometheus restart

MCQ 5 – Resolve – Answer

- When does alert resolve?
  - A. Slack restart
  - B. Grafana restart
  - **C. MySQL restored**
  - D. Prometheus restart

Exporters Overview

- Exporters expose metrics for apps without native support.
- They bridge legacy systems and Prometheus.
  - Standalone processes
  - Expose /metrics
  - Scraped by Prometheus
  - Operational overhead

Why Exporters Exist

- Many systems lack Prometheus metrics.
- Exporters fill this gap.
  - MySQL
  - Redis
  - Nginx
  - Legacy systems

Exporter Tradeoffs

- Exporters add complexity.
- They must be maintained.
  - Extra deployment
  - Versioning
  - Security
  - Failure points

Native Metrics Preferred

- Native metrics are simpler and safer.
- Use exporters only when required.
  - Fewer components
  - Better context
  - Lower latency
  - Simpler ops

MySQL Exporter

- The MySQL exporter exposes database internals.
- It runs as a separate service.
  - mysqld-exporter
  - Queries internals
  - Prometheus format
  - Metrics endpoint

Exporter Docs

- Documentation explains configuration.
- Metrics and flags are listed.
  - DockerHub
  - GitHub
  - Env vars
  - Supported metrics

Testing Exporters

- Always verify exporters work.
- Curl the metrics endpoint.
  - Find pod
  - Use netshoot
  - Curl /metrics
  - Inspect output

Netshoot Pod

- Netshoot is a debug container.
- It includes network tools.
  - curl
  - dig
  - tcpdump
  - Cluster debugging

ServiceMonitor

- ServiceMonitors define scraping.
- They are Kubernetes CRDs.
  - Label selectors
  - Scrape interval
  - Declarative config
  - GitOps-friendly

Exporter Workflow

- Exporters fit into the scrape pipeline.
- They behave like any metrics endpoint.
  - App stats
  - Exporter
  - ServiceMonitor
  - Prometheus

MCQ 1 – Exporters

- Purpose of exporters?
  - A. Replace Prometheus
  - B. Expose metrics
  - C. Store logs
  - D. Trigger alerts

MCQ 1 – Exporters – Answer

- Purpose of exporters?
  - A. Replace Prometheus
  - **B. Expose metrics**
  - C. Store logs
  - D. Trigger alerts

MCQ 2 – Endpoint

- Metrics exposed at?
  - A. /health
  - B. /status
  - C. /metrics
  - D. /debug

MCQ 2 – Endpoint – Answer

- Metrics exposed at?
  - A. /health
  - B. /status
  - **C. /metrics**
  - D. /debug

MCQ 3 – MySQL

- MySQL exporter does?
  - A. Backup DB
  - B. Run queries
  - C. Expose metrics
  - D. Manage users

MCQ 3 – MySQL – Answer

- MySQL exporter does?
  - A. Backup DB
  - B. Run queries
  - **C. Expose metrics**
  - D. Manage users

MCQ 4 – ServiceMonitor

- ServiceMonitor role?
  - A. Deploy exporters
  - B. Define scraping
  - C. Visualize data
  - D. Store logs

MCQ 4 – ServiceMonitor – Answer

- ServiceMonitor role?
  - A. Deploy exporters
  - **B. Define scraping**
  - C. Visualize data
  - D. Store logs

MCQ 5 – Best Practice

- When use exporters?
  - A. Always
  - B. Only DBs
  - C. When no native metrics
  - D. Only prod

MCQ 5 – Best Practice – Answer

- When use exporters?
  - A. Always
  - B. Only DBs
  - **C. When no native metrics**
  - D. Only prod

Final Lab Overview

- You will build a centralized Automation Gateway API.
- It exposes business metrics and uses GitOps deployment.
  - Automation API
  - Business metrics
  - Argo CD
  - Grafana proof

Problem Statement

- Automation is often fragmented.
- This lab centralizes control.
  - Ad-hoc scripts
  - No visibility
  - Inconsistent tooling
  - Operational risk

Given Environment

- The cluster and observability stack already exist.
- You build on top of them.
  - EKS
  - LGTM stack
  - Reference API
  - Argo CD ready

Your Objective

- Create automation-gateway.
- Deploy and observe it.
  - FastAPI service
  - Argo CD app
  - Custom metrics
  - REST endpoints

API Endpoints

- The API must expose core endpoints.
- Jobs must run asynchronously.
  - /healthz
  - /v1/jobs
  - /v1/jobs/{id}
  - /metrics

Business Metrics

- Metrics reflect automation outcomes.
- They are not infrastructure metrics.
  - Job counter
  - Duration histogram
  - Queue depth
  - Action labels

Argo CD Deployment

- Use apps-of-apps pattern.
- All manifests live in Git.
  - Deployment
  - Service
  - ServiceMonitor
  - Application CR

Validation Steps

- Verify deployment and metrics.
- End-to-end proof is required.
  - kubectl checks
  - Test API
  - Prometheus scrape
  - Grafana panels

Grafana Dashboard

- Create a minimal dashboard.
- It proves business visibility.
  - Jobs per minute
  - p95 duration
  - Queue depth
  - Live data

Why This Matters

- This is a platform control plane.
- Tools can change without breaking APIs.
  - Stable interfaces
  - Replaceable tools
  - Observable automation
  - Production-ready