High-Level Design Document

for

Data and Authentication Monitoring Platform (DAMP)

[1. Introduction 1](#_Toc151582875)

[2. Scope 2](#_Toc151582876)

[3. Requirements 3](#_Toc151582877)

[2.1. Functional 3](#_Toc151582880)

[2.2. Non-Functional 3](#_Toc151582881)

[4. Conclusion 4](#_Toc151582882)

# Introduction

The Data and Authentication platform has multiple distributed systems that need to be proactively monitored. These systems host critical services utilising multiple technologies and running variety of applications which generate stats, metrics and logs.

The monitoring system the platform is designed to ensure continuous availability, security, and optimal performance. It encompasses a diverse set of components for collecting, analysing, and visualising metrics and logs from the multiple distributed systems on the platform.

# System Architecture

# Monitoring Components

* + 1. Monitoring Agents

These will be:

* Deployed on each system for real-time data collection.
* Collect metrics and logs for both the operating system and services/applications.
  + 1. Centralized Monitoring Server

This will be provisioned with:

* Prometheus for metrics and alerting.
* ELK Stack (Elasticsearch, Logstash, Kibana) for log management and visualisation.
* Grafana for creating dashboards and visualization.
  + 1. Security Layer

This will facilitate:

* Encryption for data in transit and at rest.
* Strong authentication mechanisms for access control.
  + 1. Incident Management Integration

Integration with an incident management system like PagerDuty for immediate alert notification and response coordination by the operational team.

# Scalability

* + 1. Horizontal Scaling

Monitoring components will be deployed to be scalable horizontally handle growing infrastructure.

* + 1. Container Orchestration

To facilitate dynamic scaling of applications hosted on the platform, container orchestration like Kubernetes will be leveraged.

* + 1. Database Scalability

databases for storing metrics and logs.

* + 1. Load Balancing:

Implement load balancing for high availability and efficient distribution of monitoring tasks.

# Self-Diagnosis and Notification

* + 1. Anomaly Detection

Anomaly detection algorithms will be implemented in Prometheus to identify deviations from normal behaviour and annunciate such anomalies within the system.

* + 1. Automated Alerts

Automation alert will be configured based on predefined thresholds and anomaly detection.

* + 1. Self-Healing Mechanisms

Self-healing mechanisms like automated service restarts or resource scaling will be implemented where possible.

3. Security Considerations:

3.1 Data Protection:

Personal Data Handling:

Anonymize or pseudonymize personal data in logs to protect privacy.

Implement access controls to restrict access to sensitive monitoring data.

Encryption:

Use TLS for encrypting data in transit.

Encrypt stored data, especially logs containing personal information.

3.2 Access Controls:

Role-Based Access Control (RBAC):

Implement RBAC to control access to monitoring data based on roles and responsibilities.

Audit Logs:

Generate audit logs to track access and changes to the monitoring system.

4. Cost-Effective Design:

4.1 Cloud Resource Optimization:

Reserved Instances:

Utilize reserved instances or spot instances to optimize costs in hybrid cloud deployments.

Resource Scaling:

Implement auto-scaling to dynamically adjust resources based on demand, minimizing unnecessary costs.

4.2 Open-Source Tools:

Open-Source Monitoring Tools:

Leverage open-source monitoring tools (Prometheus, Grafana, ELK Stack) to reduce licensing costs.

Cost-Aware Configurations:

Optimize configurations to reduce resource usage and associated costs.

5. Documentation and Training:

Comprehensive Documentation:

Maintain detailed documentation for installation, configuration, and troubleshooting.

Training for Operational Support:

Provide training sessions for the Operational Support team on monitoring tool usage and incident response procedures.

6. Continuous Improvement:

Regular Reviews:

Conduct regular reviews of the monitoring system to identify areas for improvement.

Feedback Mechanism:

Establish a feedback mechanism with the Operational Support team to incorporate their experiences and insights.

By adhering to these design principles, the monitoring system ensures the continuous health and security of the Data and Network Authentication platform. Regular updates and continuous improvement practices will contribute to the system's long-term effectiveness.

# Requirements



# Functional

The monitoring system shall:

1. Be operational and available 24/7.
2. Monitor services hosted both on-premises(virtual or otherwise) and in hybrid clouds.
3. Collect metrics and logs for Operating System and services/applications.
4. Secure personal data that flows through the system.
5. Handle Unix based tech stack running applications and infrastructure built with Ruby and Java based applications.

# Non-Functional

The system should be:

1. Scalable.
2. Cost effective.
3. Self-diagnose and ensure operational support team are aware of any issues.

# Conclusion

The requirements set out in this document are the minimum to implement a robust monitoring system that is operational and available 24/7. This is a living document that will evolve over time to accommodate changing requirements and latest changes in processes and tools as the system matures.