Self-Healing Infrastructure with Prometheus, Alertmanager & Ansible

Abstract

This project demonstrates the implementation of a self-healing infrastructure using Prometheus, Alertmanager, and Ansible. The goal was to automate the detection and recovery of service failures in a containerized environment. A sample NGINX service was deployed and monitored for uptime. If the service went down, Prometheus triggered an alert, Alertmanager forwarded it to a webhook, and Ansible automatically restarted the service. This ensures high availability and reduces the need for manual intervention in handling failures.

Introduction

In modern cloud and DevOps practices, ensuring uptime and resiliency of applications is crucial. Traditional monitoring tools only notify administrators when a service fails, requiring manual intervention. This project enhances the monitoring system by adding an automation layer, enabling services to recover automatically from failures. By combining Prometheus for monitoring, Alertmanager for alert handling, and Ansible for automation, we built a self-healing infrastructure that recovers NGINX whenever it goes down.

Tools Used

1. Prometheus – Metrics collection and alerting based on rules.
2. Alertmanager – Handles alerts and forwards them to webhooks.
3. Ansible – Automates service recovery using playbooks.
4. Blackbox Exporter – Probes HTTP endpoints to check service health.
5. Node Exporter – Collects system-level metrics.
6. Docker & Docker Compose – Containerized deployment environment.
7. Flask Webhook – Receives alerts from Alertmanager and triggers Ansible playbooks.

Steps Involved in Building the Project

1. Deployed a sample NGINX service using Docker.
2. Configured Prometheus to monitor Node Exporter and Blackbox Exporter.
3. Defined alerting rules for service downtime and high CPU usage.
4. Set up Alertmanager to receive alerts from Prometheus and forward them to a custom Flask webhook.
5. Implemented a Flask webhook service to handle incoming alerts.
6. Created an Ansible playbook to automatically restart the NGINX container upon receiving alerts.
7. Tested the workflow by stopping the NGINX container and verifying that it restarted automatically.

Conclusion:

This project successfully demonstrated a self-healing infrastructure where a failed service “Nginx” was automatically recovered without manual intervention. By integrating Prometheus, Alertmanager, and Ansible, we built a resilient monitoring and recovery system. This approach can be extended to more complex applications, ensuring reliability and reducing downtime in production environments.