Self-Healing Infrastructure Using Prometheus, Alertmanager, and Ansible

Introduction

In the modern DevOps ecosystem, minimizing service downtime is crucial. This project implements a basic self-healing infrastructure that detects service failures using Prometheus, sends alerts through Alertmanager, and automatically resolves issues using Ansible and Shell scripting. The entire setup runs on Docker and can be deployed on any cloud VM (tested on AWS EC2 Ubuntu instance).

Abstract

The goal of this project is to simulate a production-grade monitoring system that not only detects issues but also self-recovers from them. The core of this system is Prometheus, which scrapes metrics and evaluates alerting rules. When a service (e.g., Nginx) goes down, Alertmanager sends an alert which can be captured by a webhook or email. With a simple shell/Ansible integration, the system can restart the service or perform custom recovery actions, ensuring high availability.

Tools Used

- Prometheus: Monitoring and alerting
- Alertmanager: Alert dispatching
- Nginx: Sample service to monitor
- Ansible: Recovery automation
- Docker & Docker Compose: Container orchestration
- AWS EC2 (Ubuntu): Hosting environment
- Shell Scripting: Triggering self-healing logic

Steps Involved in Building the Project

1. Setup Folder Structure: Created directories for Prometheus rules and Alertmanager config.

2. Prometheus Configuration: Defined prometheus.yml with scraping jobs and alerting rules.

3. Alertmanager Setup: Configured alertmanager.yml to send alerts via email or webhook.

4. Nginx as Target Service: Deployed Nginx in a container as the monitored application.

5. Docker Compose Orchestration: Wrote docker-compose.yml to spin up Prometheus, Alertmanager, and Nginx.

6. Testing Alerts: Manually stopped Nginx to trigger an alert and verified recovery actions.

7. Pushed to GitHub: Set up SSH and pushed project to a public GitHub repo.

Conclusion

This project demonstrates how containerized services can self-heal in response to system faults using open-source tools. While the implementation is simple, it reflects real-world SRE (Site Reliability Engineering) principles. This framework can be extended to include Slack/Teams integration, Grafana dashboards, and advanced Ansible automation. It serves as a solid foundation for more complex, production-grade observability stacks.