Self-Healing Infrastructure with Prometheus, Alertmanager & Ansible Introduction

ELEVATE LABS

Prepared by: Suvratam Patra

Role: DevOps Engineer / Intern

INTRODUCTION :

In modern DevOps environments, maintaining high availability and reliability is crucial. System downtime can lead to significant losses and user dissatisfaction. This project focuses on building a self-healing infrastructure that can automatically detect service failures and recover from them without manual intervention. Using Prometheus, Alertmanager, and Ansible, the system monitors service uptime and performs automated remediation actions when failures are detected.

ABSTRACT :

This project demonstrates the implementation of an automated self-healing system using Prometheus for monitoring, Alertmanager for alert routing, and Ansible for automated recovery. When Prometheus detects that a service (like NGINX) is down via the Black-box Exporter, it triggers an alert that Alertmanager forwards to a webhook service. The webhook then executes an Ansible playbook to restart the affected service, restoring system functionality automatically.

TOOLS USED :

- Prometheus – for monitoring system metrics and service uptime.
- Alertmanager – for receiving alerts and triggering automated actions.
- Blackbox Exporter – to probe HTTP endpoints and detect failures.
- Ansible – to automate service recovery using playbooks.
- Docker – to containerize and deploy all services consistently.

STEPS INVOLVED IN BUILDING THE PROJECT :

1. Deployed an NGINX container as the sample web service.
2. Configured the Blackbox Exporter to probe the NGINX endpoint and report its health.
3. Set up Prometheus to scrape metrics from Blackbox Exporter and evaluate alerting rules.
4. Created an alert rule to trigger when the service is unavailable (probe_success == 0).
5. Configured Alertmanager to send alerts to a webhook receiver.
6. Built a webhook service to trigger an Ansible playbook upon receiving alerts.
7. Wrote an Ansible playbook that restarts the NGINX container automatically.
8. Tested the system by stopping NGINX and confirming automatic recovery.

CONCLUSION :

The self-healing infrastructure built using Prometheus, Alertmanager, and Ansiblesuccessfully demonstrated automated service recovery. This project highlights the power-of integrating monitoring, alerting, and automation tools to maintain system reliability with minimal manual intervention. Such architectures are foundational to modern cloud andSRE practices, enabling resilient and self-managing systems.

INTRODUCTION :

Monitoring and alerting are essential pillars of any modern DevOps infrastructure.However, setting up a complete severability stack for experimentation or training often requires multiple systems and complex configurations. This project simplifies that process by creating asingle-VM sandbox that runs Prometheus, Grafana, and Alertmanager using Docker containers. Itprovides a ready-to-use environment where developers and DevOps engineers can explore how metrics are collected, visualized, and used for alerting all within a local environment that mirrors real-world setups.

ABSTRACT :

This project aims to build a local, all-in-one DevOps monitoring sandbox that integratesPrometheus, Grafana, and Alertmanager within a single virtualized environment. The sandbox provides a compact, easily reproducible platform for learning, experimenting, and demonstrating severability concepts. Using Vagrant with the Libvirt provider not (virtualbox) it would be same steps either, a fully automated VM was provisioned and configured with Dockerized monitoring components. This setup helps users simulate real-world infrastructure monitoring workflows locally without needing multiple servers or cloud resources.

TOOLS USED :

Vagrant: To automate VM provisioning and configuration. Libvirt + KVM:
Docker: For containerized deployment of Prometheus,Grafana, and Alertmanager. Prometheus: Used for metrics scraping and system monitoring Grafana: Provides dashboards for visualization and analysis. Alertmanager: Handles alerts and routes notifications. Shell Script (provision.sh): Automates installation and container setup inside the VM.

STEPS INVOLVED IN BUILDING THE PROJECT VAGRANT CONFIGURATION:

Created a Vagrantfile using the Libvirt provider to define the VM (Ubuntu 22.04, 2 GB RAM, 2 CPUs, private network) Provisioning

Step 1: Wrote a provision.sh script that installs Docker, pulls Prometheus, Grafana, and Alert manager images, and starts them as containers.

Step 2 Prometheus Configuration: Configured prometheus.yml to scrape metrics from local services and the host system.

Step 3 Grafana: Set up Grafana dashboards and connected it to Prometheus as a data source. Alertmanager Configuration:Configured basic alert rules in Prometheus and integrated them with Alertmanager for testing alertnotifications. Testing and Verification: Verified all services by accessing Prometheus(http://vm-ip:9090), Grafana (http://vm-ip:3000), and Alertmanager (http://vm-ip:9093).

Conclusion :

The project successfully demonstrates a fully functional, local monitoring and alerting sandbox. Using open-source DevOps tools, it enables practical, hands-on exploration serviceability concepts. The modular and automated design makes it easy to reproduce or extend-for training, testing, or integration with larger environments.