Project Report: Self-Healing Infrastructure with Prometheus, Alertmanager & Ansible

Introduction

In modern DevOps environments, system reliability and uptime are crucial. Manual intervention during service failures can lead to delays and downtime. This project aims to create a **self-healing infrastructure** that automatically detects service failures and recovers them without human involvement. Using open-source monitoring and automation tools **– Prometheus, Alertmanager, and Ansible** — the system continuously monitors a sample service (NGINX) and performs automated recovery actions based on predefined alert rules.

Abstract

The objective of this project is to build an **automated recovery system** capable of identifying service disruptions and triggering corrective actions autonomously.

Prometheus collects performance metrics and monitors the health of services. When an anomaly or downtime is detected, **Alertmanager** sends an alert to a **custom webhook**, which then triggers an **Ansible playbook**. The playbook executes a remediation task, such as restarting a failed NGINX service. This ensures minimal downtime, improved service reliability, and operational efficiency – key aspects of DevOps automation.

Tools Used

- **Prometheus** – For monitoring and metrics collection
- **Alertmanager** – For managing alerts and sending webhook notifications
- **Ansible** – For automating service recovery actions
- **NGINX** – Sample web service used for uptime monitoring
- **Shell Script** – To integrate alert triggers and execute Ansible playbooks
- **Ubuntu VM** – Environment for deployment and testing

Steps Involved in Building the Project

1. **Setup Environment**
   Installed Prometheus, Alertmanager, and Ansible on an Ubuntu VM. Deployed NGINX as the sample monitored service.

2. **Prometheus Configuration**
   Configured `prometheus.yml` to scrape metrics from NGINX and system endpoints. Created `alert_rules.yml` to detect when NGINX is down or CPU usage exceeds 90%.

3. **Alertmanager Integration**
   Configured `alertmanager.yml` with a webhook receiver. When Prometheus triggers an alert, Alertmanager sends a JSON payload to the webhook endpoint.

4. **Webhook and Ansible Automation**
   Developed a `webhook.py` script that receives the webhook trigger and executes an Ansible playbook (`restart_nginx.yml`). The playbook automatically restarts the NGINX service if it is found inactive.

5. **Testing and Verification**
   Manually stopped the NGINX service to simulate a failure. Verified that Prometheus detected the issue, Alertmanager triggered the alert, and Ansible successfully restarted the service, confirming the self-healing functionality.

Conclusion

This project successfully demonstrates a **self-healing DevOps infrastructure** where system failures are automatically detected and resolved without human intervention. It highlights the power of integrating **monitoring** (Prometheus), alerting (Alertmanager), and automation (Ansible) tools to achieve intelligent, reliable, and automated IT operations. Such systems reduce downtime, enhance fault tolerance, and form the foundation of modern **Site Reliability Engineering** practices.