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

Introduction
In modern DevOps practices, maintaining application uptime and resiliency is critical. Manual recovery from failures can lead to downtime, which impacts business operations. To overcome this, self-healing infrastructure is implemented. The aim of this project is to build an automated system that detects failures, triggers alerts, and executes recovery actions without human intervention.

Abstract
This project demonstrates a self-healing infrastructure using Prometheus, Alertmanager, and Ansible. Prometheus continuously monitors services and collects metrics. Alertmanager evaluates alerting rules and triggers actions when thresholds are breached (e.g., service down or CPU utilization above 90%). When an alert occurs, Alertmanager sends a webhook to a Python service, which triggers an Ansible playbook to restart the failed service. This ensures automatic recovery and improves reliability without manual involvement.

Tools Used
- Prometheus – for monitoring and metrics collection.
- Alertmanager – for managing alerts and notifications.
- Ansible – for automation and recovery tasks.
- Docker & Docker Compose – for containerized deployment.
- Shell Scripting & Python (webhook service) – for integration and orchestration.

Steps Involved in Building the Project
1. Service Setup: Deployed a sample service (NGINX) inside Docker to simulate production workload.
2. Prometheus Configuration: Set up Prometheus with job configurations to monitor the NGINX service and system health.
3. Alerting Rules: Defined alert rules (e.g., service down or CPU > 90%).
4. Alertmanager Configuration: Configured Alertmanager with alertmanager.yml to listen for alerts from Prometheus and forward them to a webhook receiver.
5. Webhook Service: Built a lightweight Python Flask webhook service to receive alerts and trigger Ansible.
6. Ansible Playbook: Wrote a playbook to restart the NGINX service automatically when triggered.
7. Testing & Validation: Stopped the NGINX container manually → Prometheus detected downtime → Alertmanager fired an alert → Webhook received alert → Ansible playbook restarted the service successfully.
8. Grafana (Optional): Connected Grafana to Prometheus for visualization of metrics and alerts.

Conclusion
The project successfully demonstrates a self-healing infrastructure where failures are automatically detected and resolved without human intervention. This improves system availability, reduces downtime, and enhances reliability. The integration of Prometheus, Alertmanager, and Ansible provides a robust monitoring and automation pipeline, which is a critical skill in real-world DevOps environments.