Self-Healing Infrastructure Project Report

Author: Ardra S
Date: 2025-09-06

Objective

The goal of this project is to detect service failures automatically and recover without manual intervention. Specifically:

- Monitor critical services (Apache, NGINX) using Prometheus.
- Configure Alertmanager to send alerts when a service goes down.
- Trigger Ansible playbooks automatically via a webhook to recover the service.
- Reduce system downtime and minimize manual troubleshooting.

System Architecture

Component	Function
Prometheus	Collect metrics from services
Alertmanager	Route alerts based on rules
Flask Webhook	Receives alerts and triggers Ansible playbooks
Ansible Playbooks	Automates recovery actions

Tools & Technologies

Tool/Technology	Purpose
Prometheus	Monitoring metrics and detecting failures
Alertmanager	Alert routing, grouping, and suppression
Flask	Webhook server to trigger automation
Ansible	Automated remediation tasks
AWS EC2 Ubuntu	Host environment for monitoring and automation
Git & GitHub	Version control and project repository management

---

Configuration Details

Prometheus

```
global:
  scrape_interval: 10s

scrape_configs:
  - job_name: 'nginx'
    static_configs:
      - targets: ['localhost:9113']
  - job_name: 'node'
    static_configs:
      - targets: ['localhost:9100']
```

Alertmanager

```
route:
  receiver: 'webhook_receiver'

receivers:
- name: 'webhook_receiver'
  webhook_configs:
  - url: 'http://13.61.151.15:5001/'
```

Flask Webhook

- Listens on port 5001 for incoming alerts and triggers Ansible playbooks.

Ansible Playbooks

- Automates recovery tasks: restart NGINX/Apache, free disk space, reboot servers.

Implementation Steps

1. Launch AWS Ubuntu EC2 instance.
2. Install Prometheus and Alertmanager.
3. Configure Prometheus to scrape metrics and send alerts.
4. Deploy Flask webhook to receive alerts.
5. Write Ansible playbooks for automated remediation.
6. Integrate Prometheus → Alertmanager → Webhook → Ansible pipeline.
7. Test by simulating service failures and verify automatic recovery.

Results

- Detects service failures (Apache/NGINX down).
- Alerts routed to Flask webhook trigger Ansible playbooks.
- Automated recovery reduces downtime and manual intervention.
- Metrics and alerts visible via Prometheus dashboards.

Challenges

- Large binaries exceeded GitHub size limits – resolved using `.gitignore`.
- SSH key authentication issues – resolved using correct `.pem` file.
- Webhook and Ansible integration required multiple tests.

Conclusion

The project demonstrates a fully automated self-healing infrastructure. Ensures high availability, reduces manual intervention, and provides a foundation for scalable monitoring and automation.

Future Work

- Integrate Slack/Email notifications for alerts.
- Extend playbooks for more services.
- Deploy Grafana dashboards for visualization.
- Add Kubernetes monitoring for containerized environments.

References

1. Prometheus Documentation
2. Alertmanager Documentation
3. Ansible Documentation
4. Flask Documentation