Self-Healing Infrastructure with Prometheus, Alertmanager & Ansible

Loom video:
https://www.loom.com/share/567ecf19c84f44ff9b635eef78c831f5?sid=b22cb210-7e61-468b-9000-efad3e8be193

Github repo: vommidapuchinni/selfhealing

Project Overview

Objective:
Automatically monitor and recover a sample NGINX service using Prometheus, Alertmanager, and Ansible. When the service goes down, Prometheus triggers an alert; Alertmanager sends a webhook, which calls an Ansible playbook to restart the container automatically.

Tools Used:

- Docker – containerized environment
- NGINX – sample service
- Prometheus – metrics collection & alerting
- Alertmanager – alert management and webhook integration
- Ansible – automation to restart containers
- Flask (Python) – webhook server

Infrastructure Diagram

Explanation:

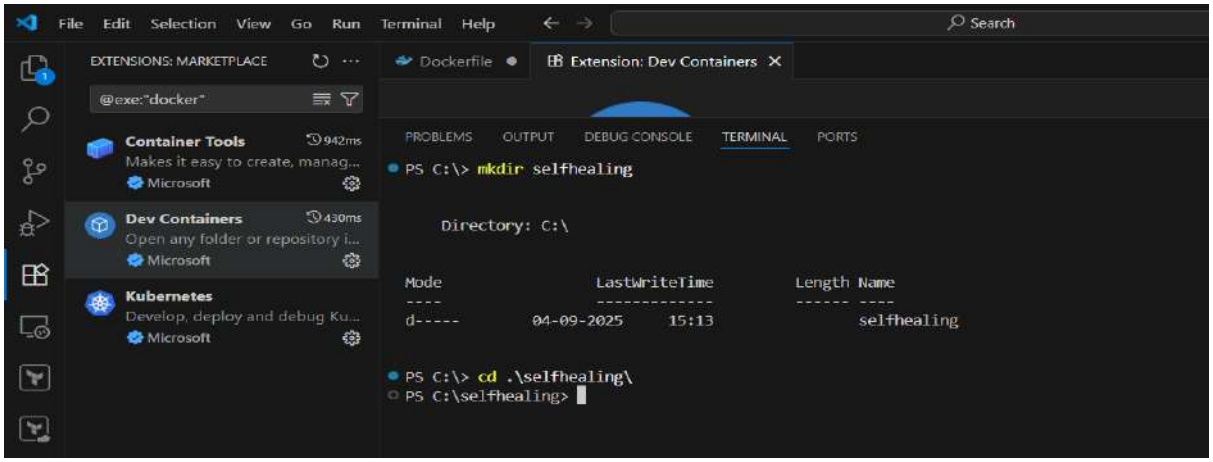
- Prometheus scrapes metrics from NGINX exporter.

- Alertmanager receives alerts and triggers webhook.
- Webhook calls Ansible playbook.
- Ansible restarts failed NGINX container automatically.

Step-by-Step Implementation

Create Project Directory

mkdir prometheus,alertmanager,ansible,nginx,webhook create these folders in selfhealing dir



Setup Docker Network

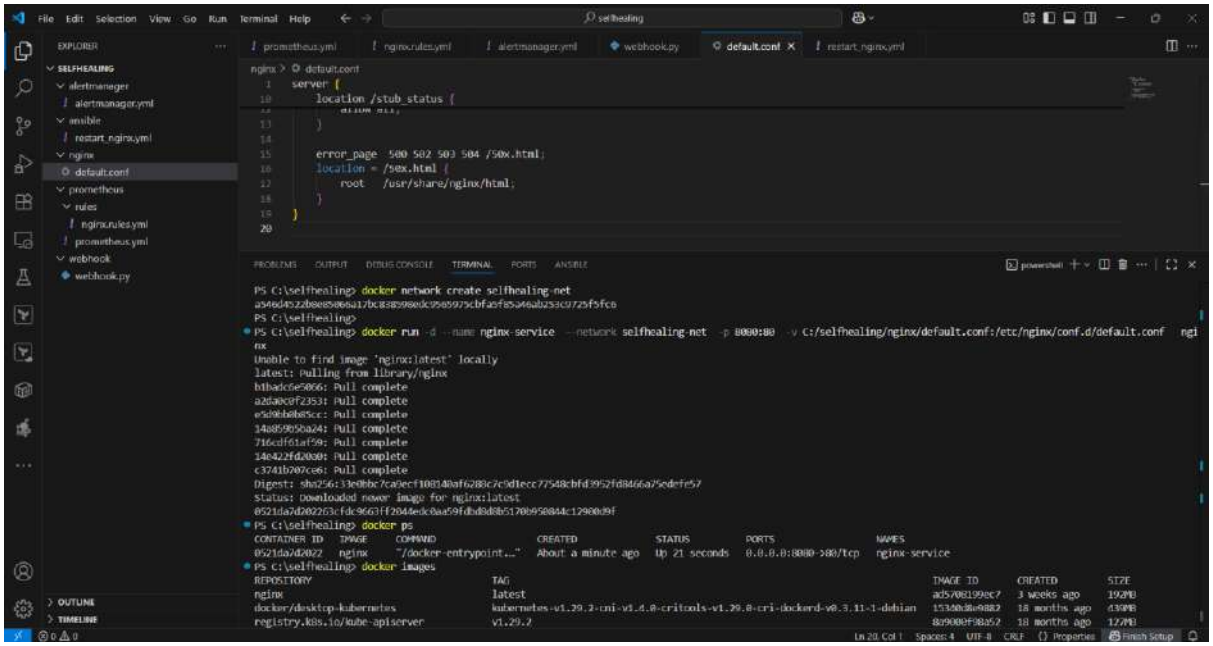
Create a custom network so all containers can communicate.

docker network create selfhealing-net

This network will be used by Prometheus, Alertmanager, and NGINX.

Deploy Sample Service (NGINX)

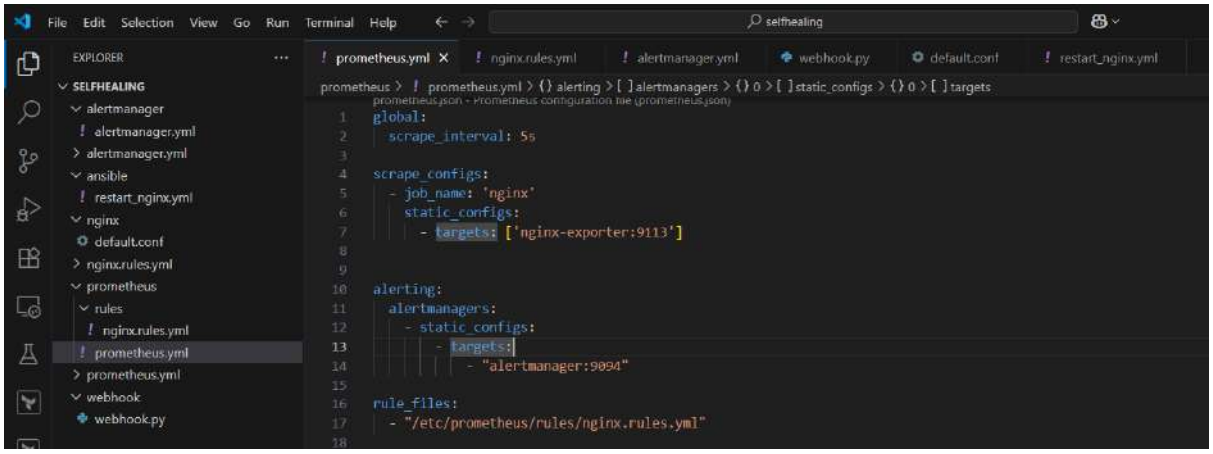
Run an NGINX container to simulate a service that can fail:



Test it: Open browser → http://localhost:8080 → Should see NGINX welcome page.

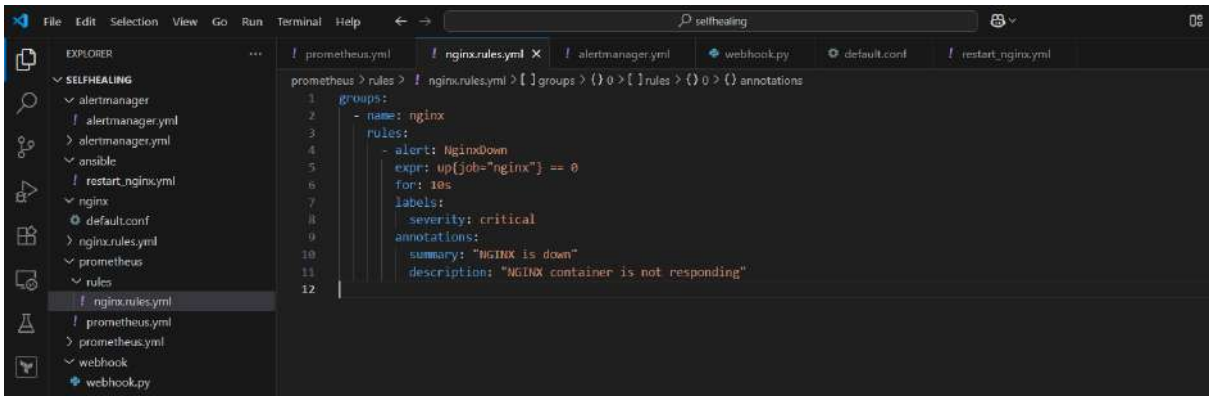
Prepare Prometheus Configuration

File: prometheus/prometheus.yml



- Prometheus scrapes NGINX metrics every 5 seconds from nginx-exporter:9113.
- Alerts are sent to Alertmanager running at alertmanager:9093.
- Alerting rules are defined in /etc/prometheus/rules/nginx.rules.yml.

Alert Rule: prometheus/rules/nginx.rules.yml



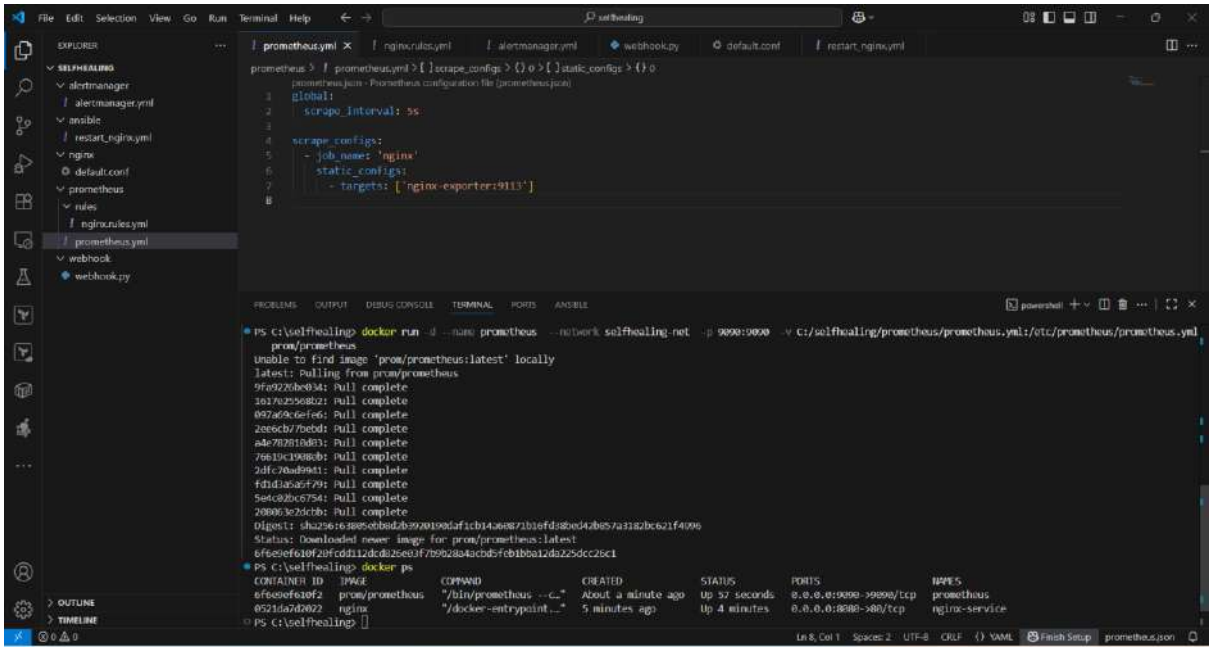
- Defines an alert rule group called nginx.
- Rule NginxDown triggers when up{job="nginx"} == 0 (NGINX not running).
- Condition must hold for 10 seconds before firing.
- Alert is labeled critical with summary and description for clarity.

Run Prometheus Container

docker run -d --name prometheus --network selfhealing-net -p 9090:9090

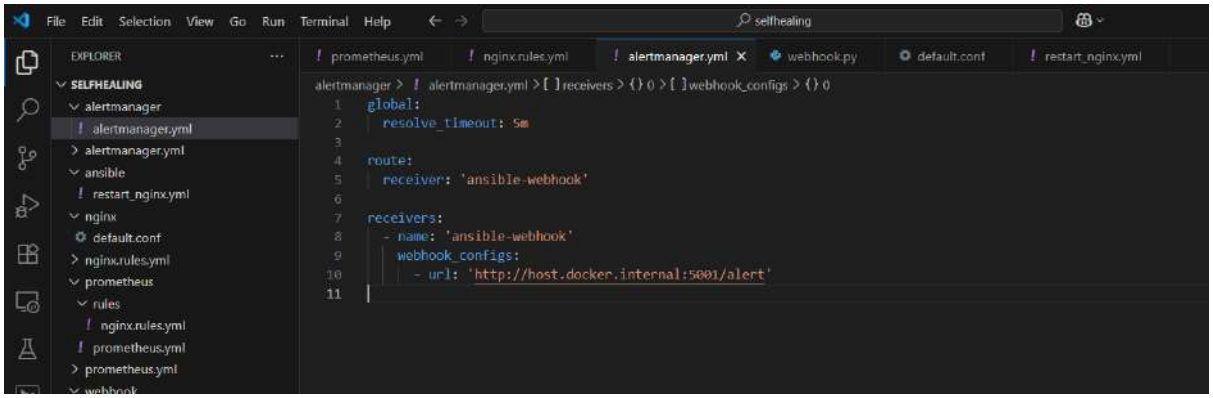
-v /c/selfhealing/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml

-v /c/selfhealing/prometheus/rules:/etc/prometheus/rules prom/prometheus



Prepare Alertmanager

File: alertmanager/alertmanager.yml



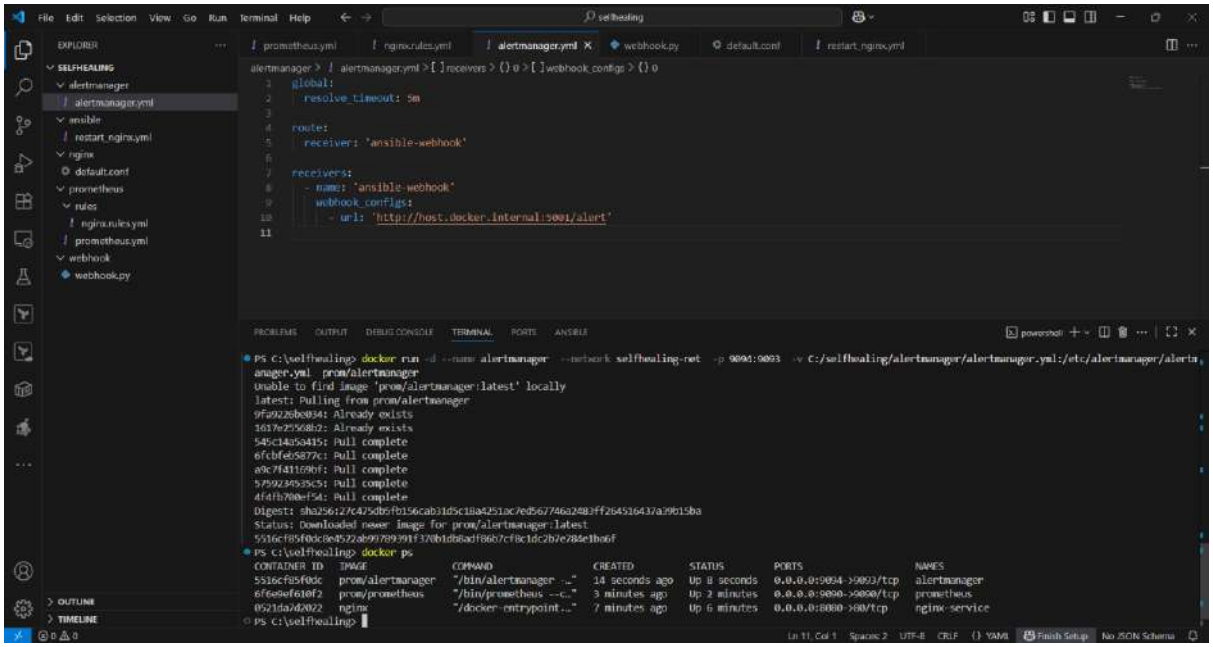
- global → resolve_timeout: 5m → Alerts are marked resolved if not firing within 5 minutes.
- route → receiver: 'ansible-webhook' → All alerts are sent to the ansible-webhook receiver.
- receiver 'ansible-webhook' → Defines a webhook target.
- webhook_configs → url → Sends alerts to Flask webhook server at http://host.docker.internal:5001/alert.

Run Alertmanager:

docker run -d --name alertmanager --network selfhealing-net -p 9094:9093

-v /c/selfhealing/alertmanager/alertmanager.yml:/etc/alertmanager/alertmanager.yml

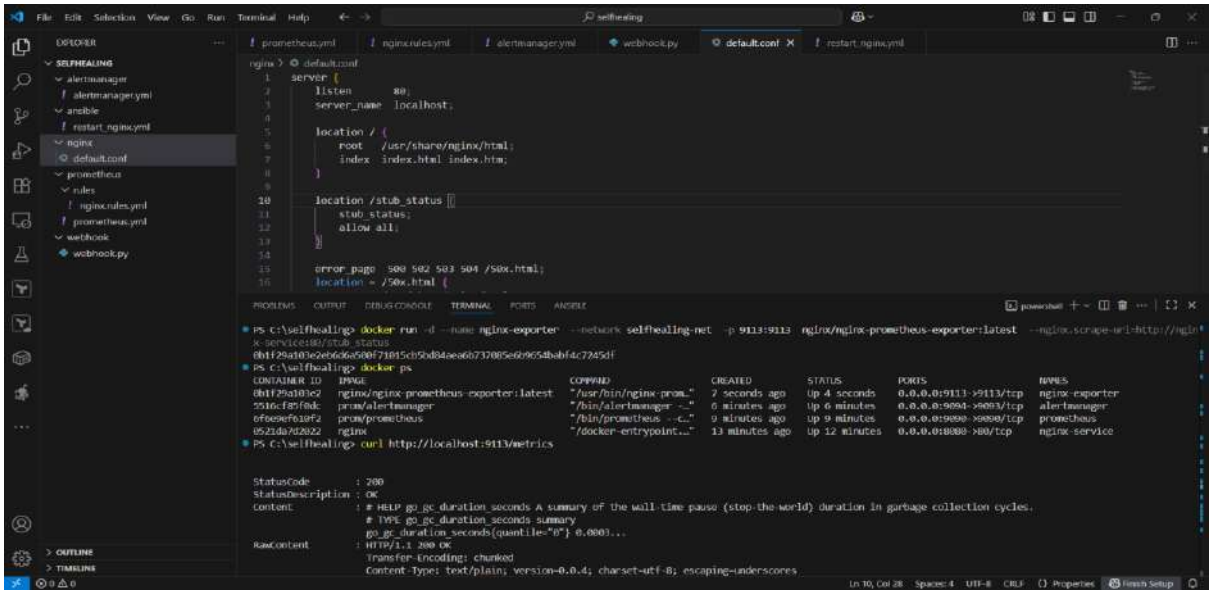
prom/alertmanager --config.file=/etc/alertmanager/alertmanager.yml



C:\selfhealing\nginx\default.conf: We run NGINX with configuration so that Prometheus has a real target to monitor, generate alerts when it goes down, and demonstrate the self-healing process.

Run NGINX Prometheus Exporter:

The NGINX exporter acts as a translator between NGINX and Prometheus, enabling monitoring and alerting.

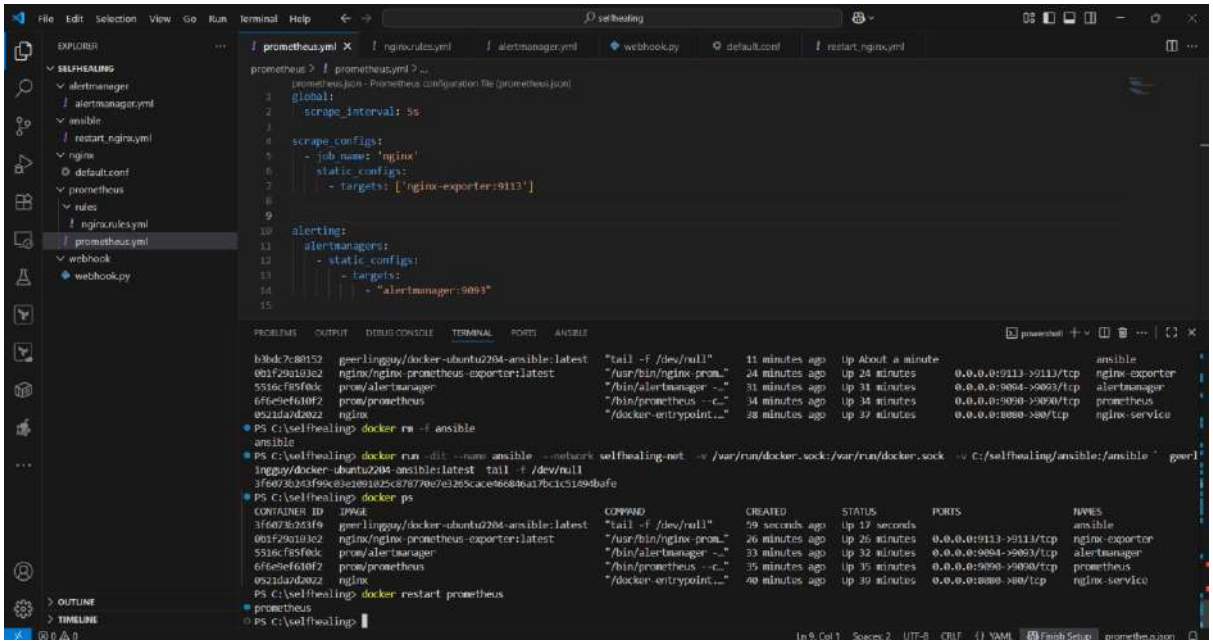


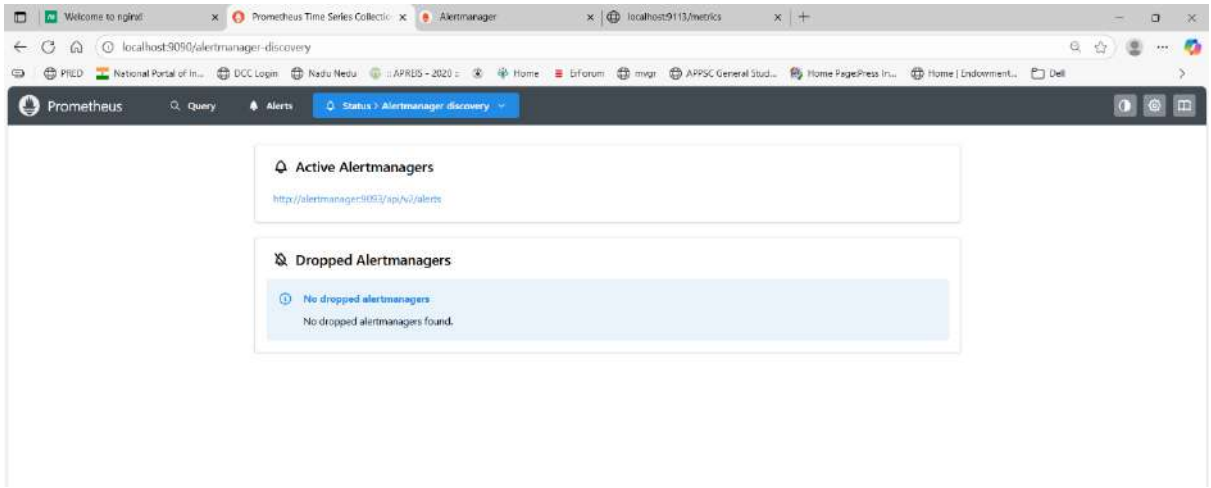
The Ansible container is used to automatically execute playbooks that fix issues (like restarting NGINX) when Prometheus/Alertmanager detect failures.

cat restart_nginx.yml

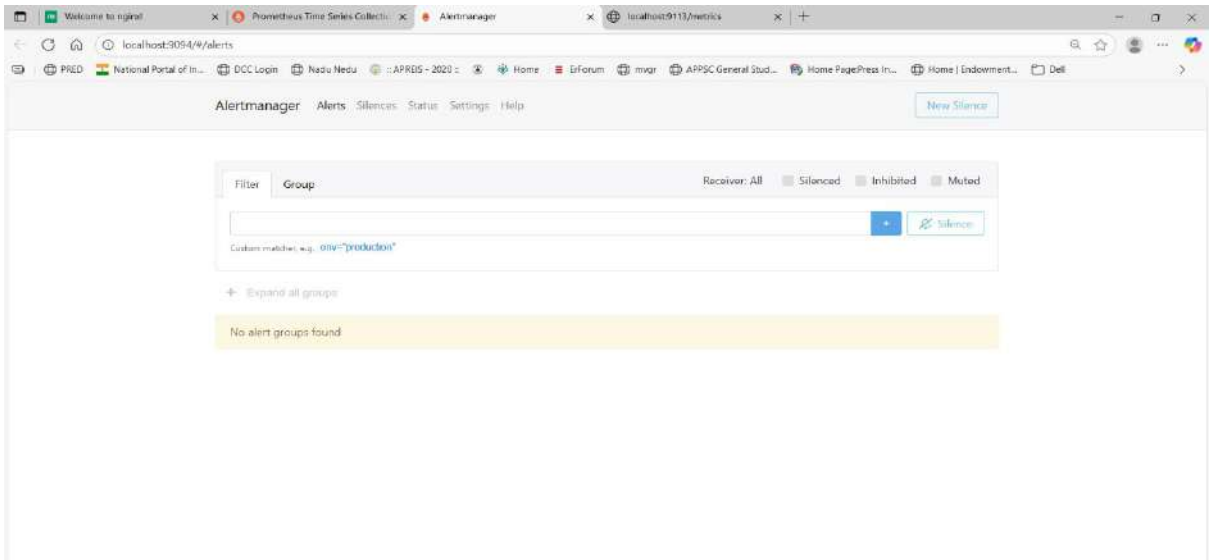
```
- hosts: localhost
  connection: local
  gather_facts: false
  tasks:
    - name: Restart NGINX Docker container
      shell: docker restart nginx-service
```

it will automatically restarts the nginx-service

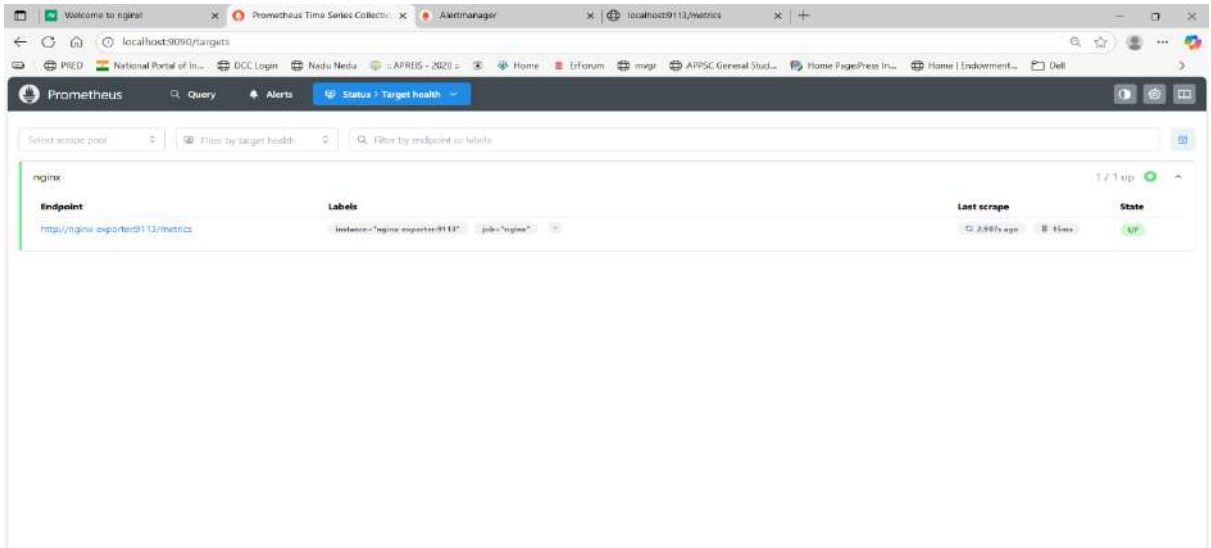




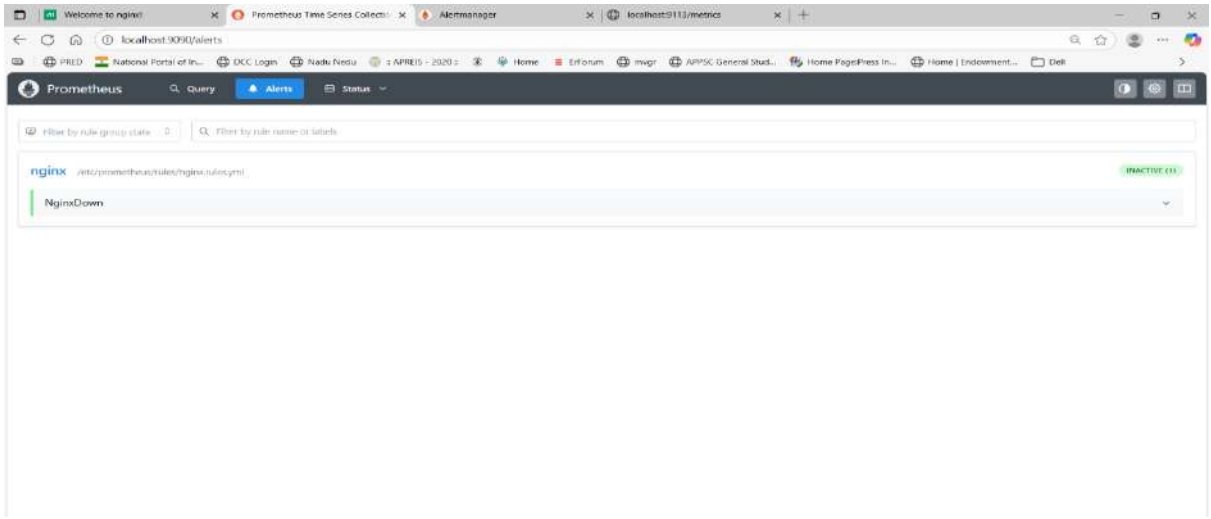
Alertmanager



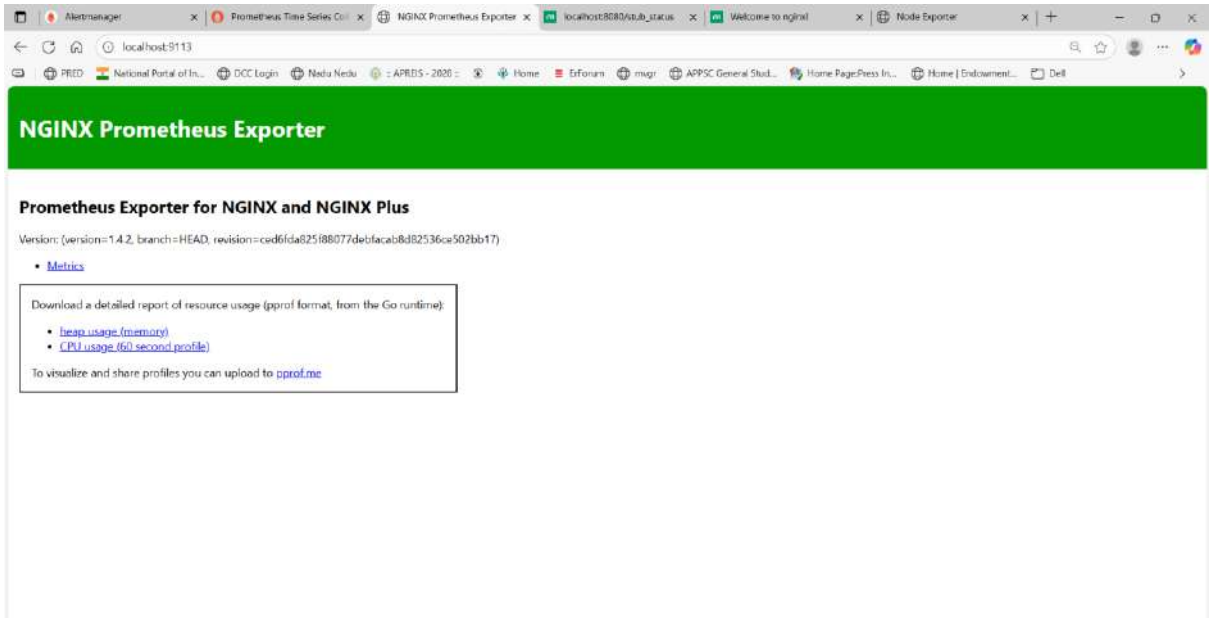
Prometheus target health

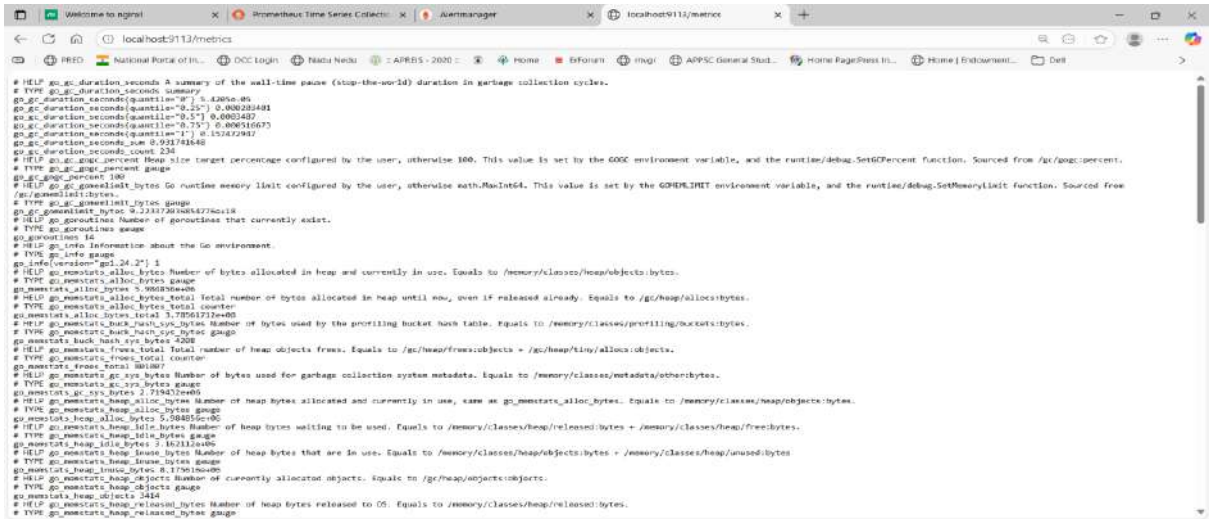


Alerts



Nginx metrics





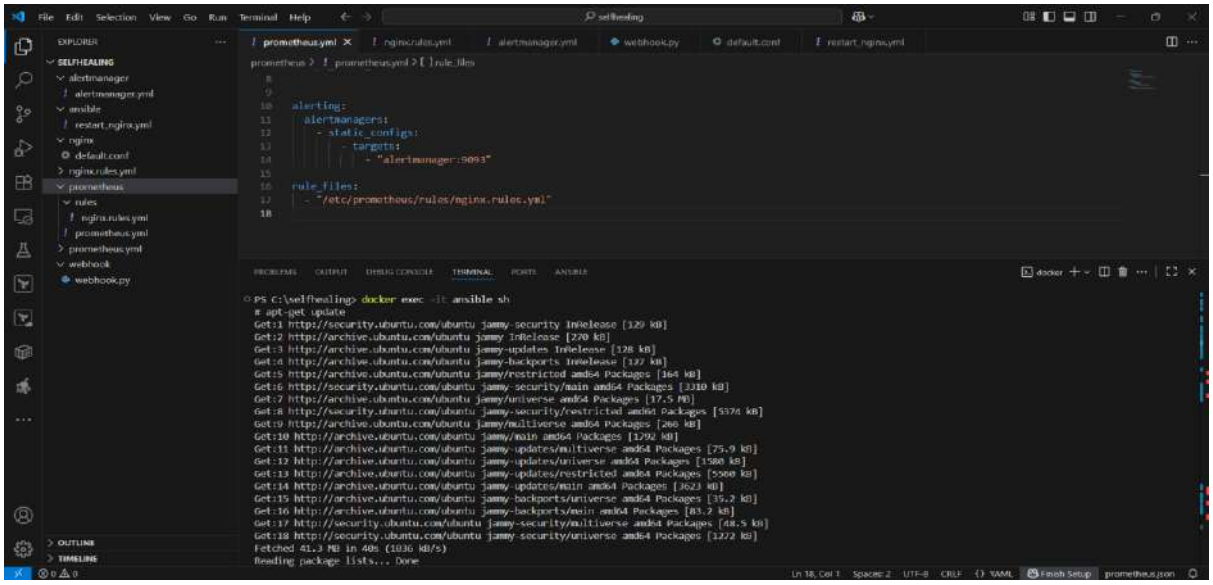
Install Docker CLI inside the Ansible container

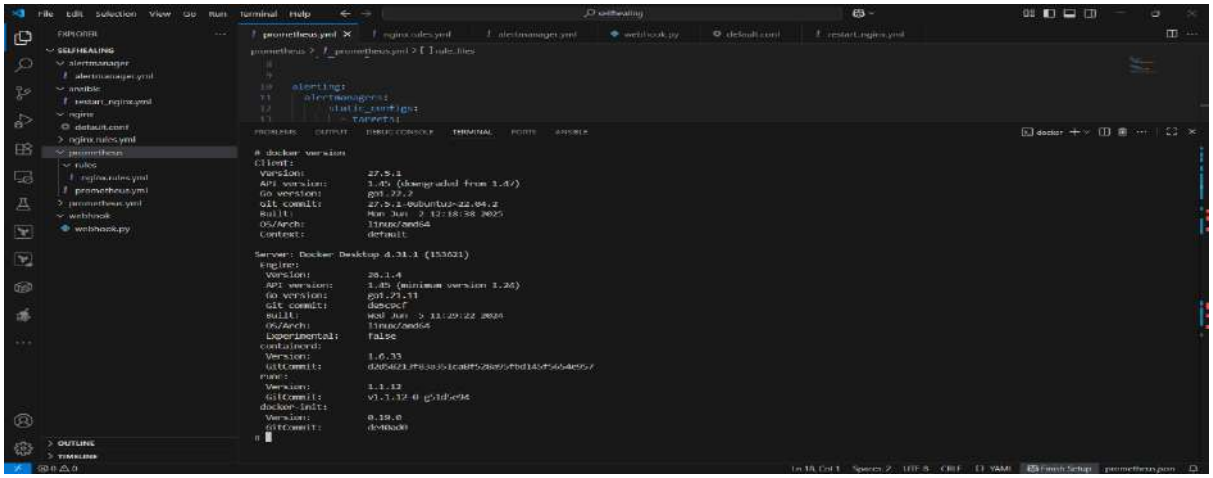
docker exec -it ansible sh

apt-get update

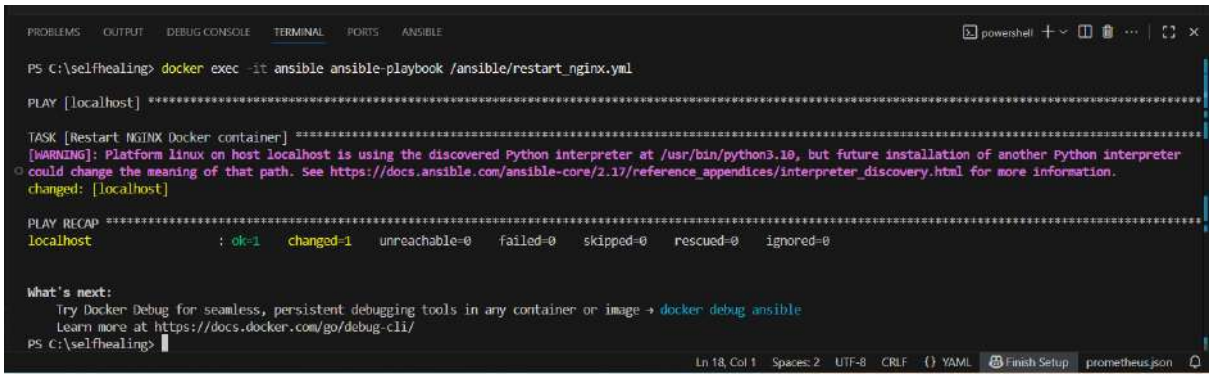
apt-get install docker.io

check docker version: docker version

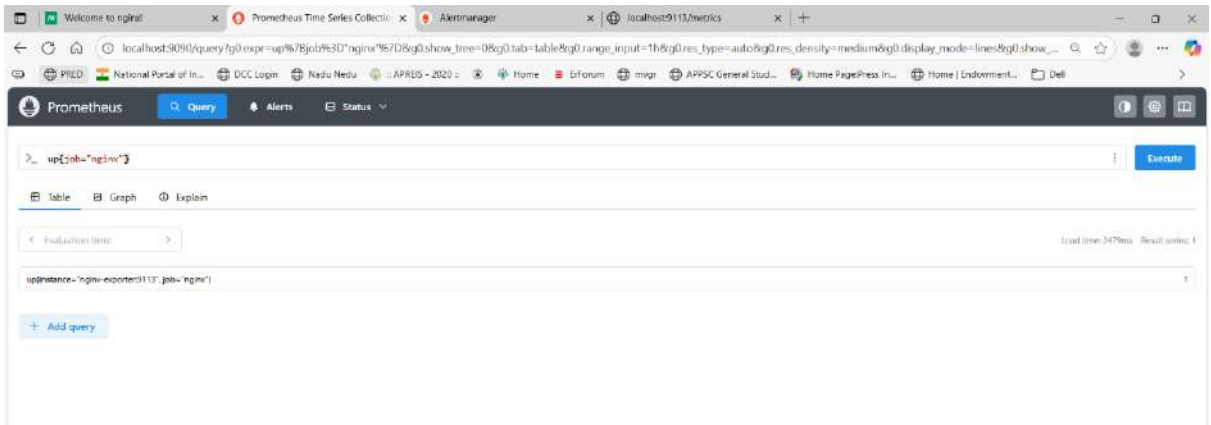




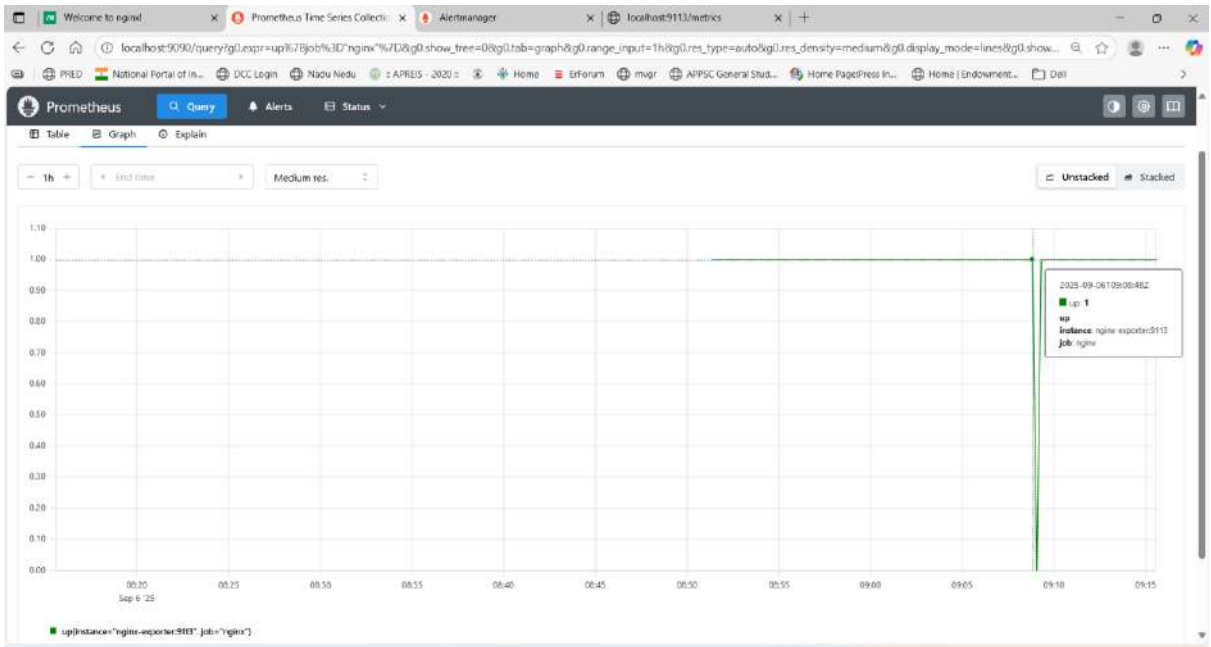
Run the playbook inside the container: docker exec -it ansible ansible-playbook /ansible/restart_nginx.yml



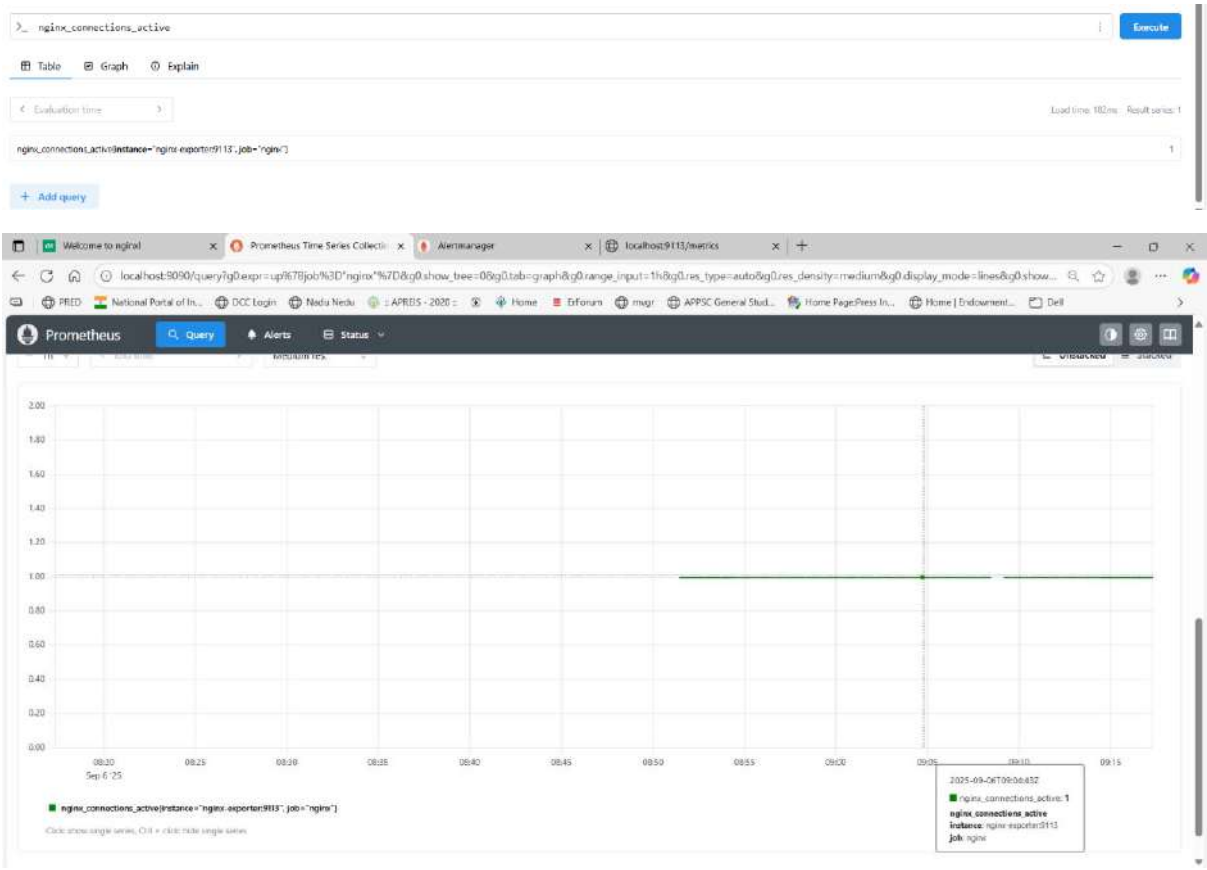
We see nginx job by executing the query we see table that nginx is running



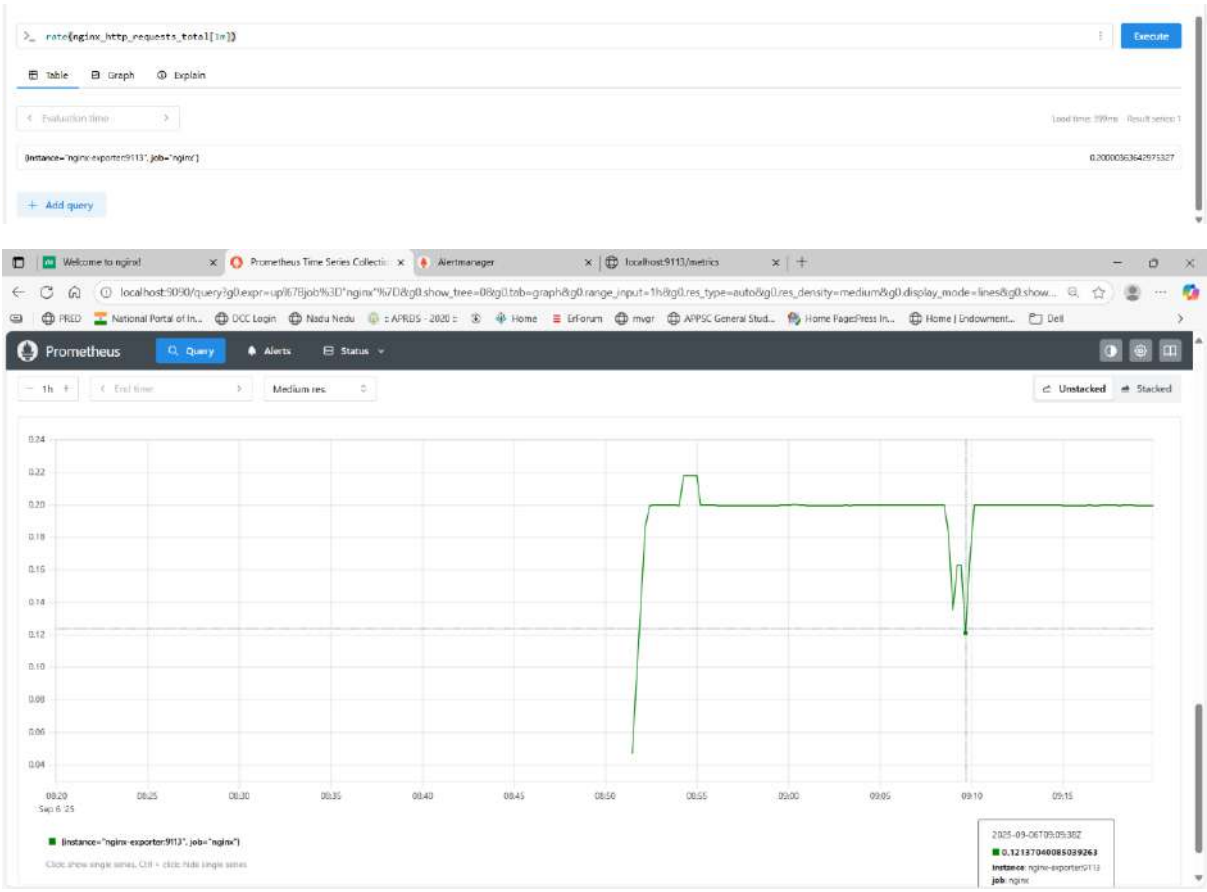
Graph of that query



How many nginx connections are active we can see by this query.



Total request hits to the nginx



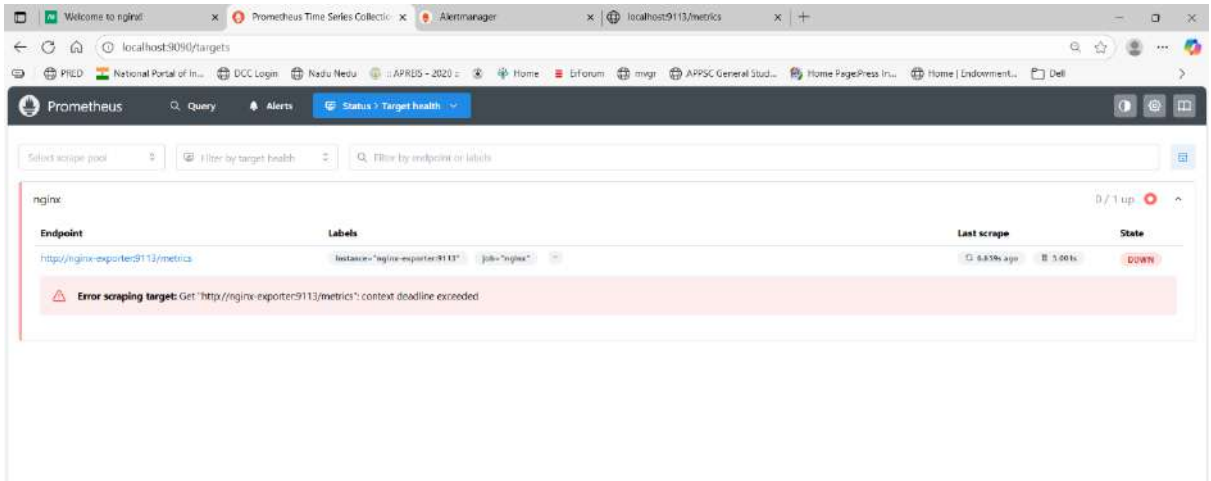
Quick test idea:

Stop NGINX container manually: docker stop nginx-service

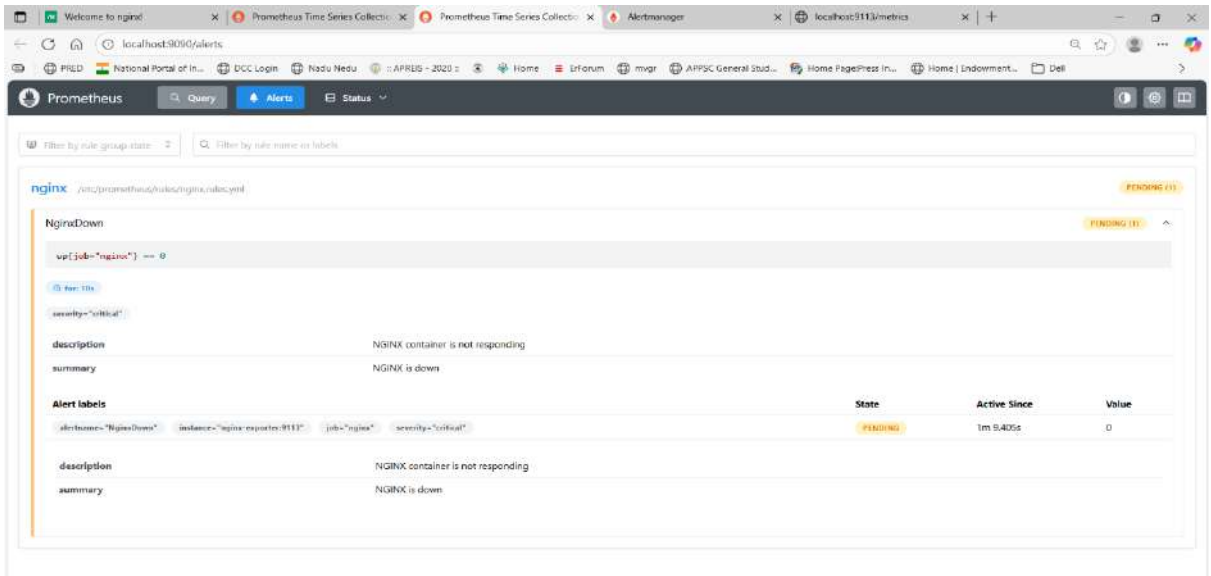
```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   ANSIBLE                                   powershell

PLAY RECAP *********************************************************************************************************************
localhost                  : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


What's next:
    Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug ansible
    Learn more at https://docs.docker.com/go/debug-cli/
PS C:\selfhealing> docker ps
CONTAINER ID   IMAGE                                      COMMAND                  CREATED          STATUS          PORTS                    NAMES
034340a7b2bb   prom/prometheus                            "/bin/prometheus --c…"   33 minutes ago   Up 33 minutes   0.0.0.0:9090->9090/tcp   prometheus
3f6073b243f9   geerlingguy/docker-ubuntu2204-ansible:latest  "tail -f /dev/null"      About an hour ago  Up About an hour                         ansible
0b1f29a103e2   nginx/nginx-prometheus-exporter:latest     "/usr/bin/nginx-prom…"   2 hours ago      Up 2 hours      0.0.0.0:9113->9113/tcp   nginx-exporter
5516cf85f0dc   prom/alertmanager                          "/bin/alertmanager -…"   2 hours ago      Up 2 hours      0.0.0.0:9094->9093/tcp   alertmanager
0521da7d2022   nginx                                      "/docker-entrypoint.…"   2 hours ago      Up 15 minutes   0.0.0.0:8080->80/tcp     nginx-service
PS C:\selfhealing> docker stop nginx-service
nginx-service
PS C:\selfhealing>

Ln 12, Col 1   Spaces: 2   UTF-8   CRLF   {} YAML   Finish Setup   No JSON Schema
```

- Wait for ~10s.
- Prometheus should trigger alert → Alertmanager → webhook → Ansible.
- NGINX container should auto-restart.

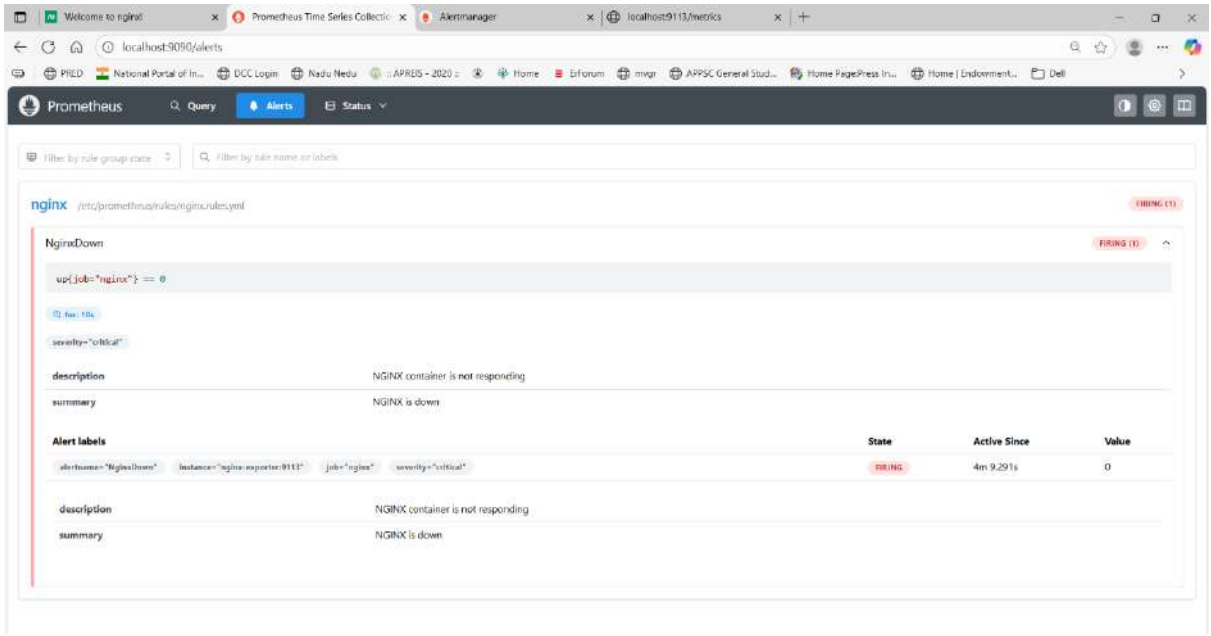
This is Prometheus target health is down because we stopped the nginx



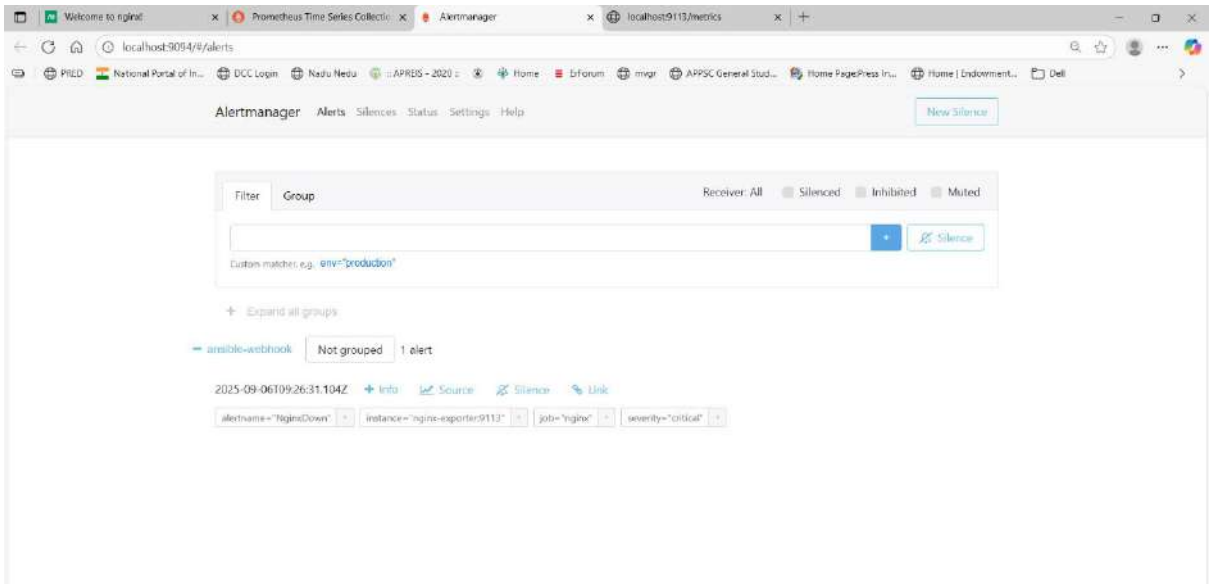
We can see alert is tirggered and showing pending



We see that alert is firring

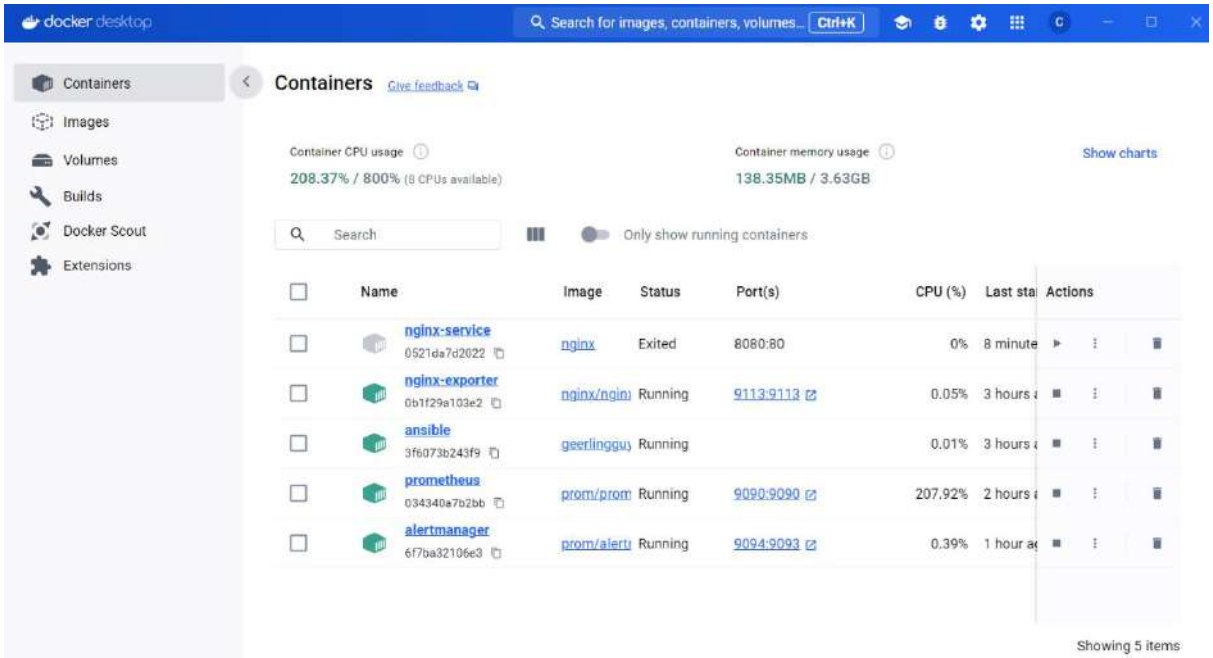


In Alert manager we see nginx down is triggered

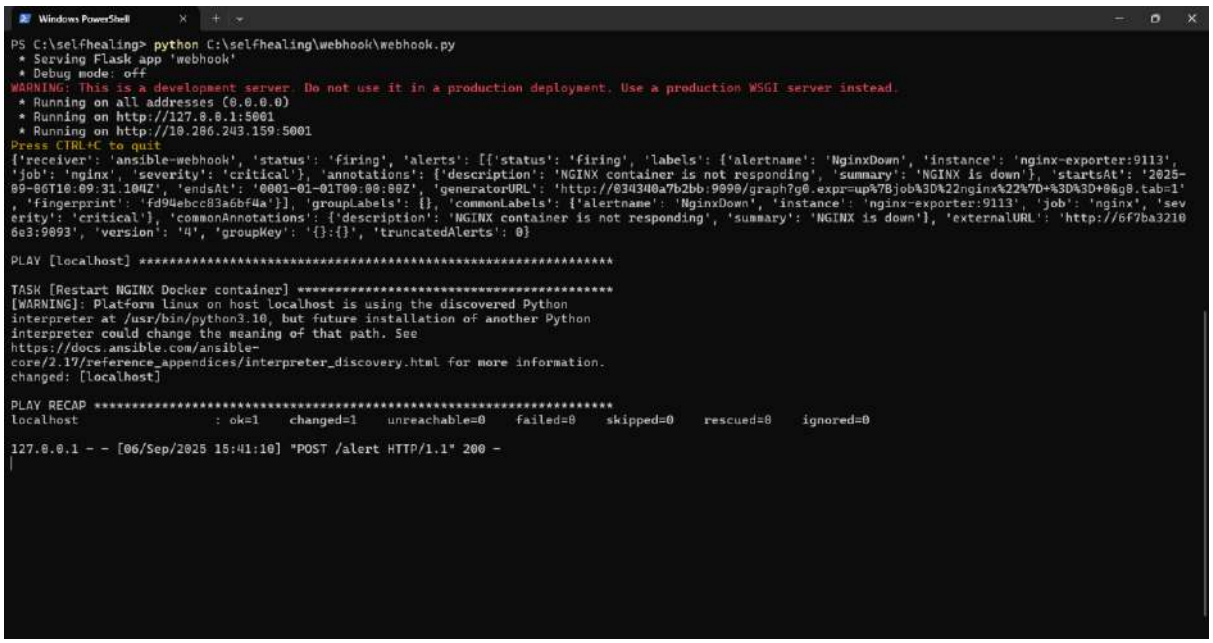


We see nginx is not accessed

We see nginx is stopped that we see it was exited

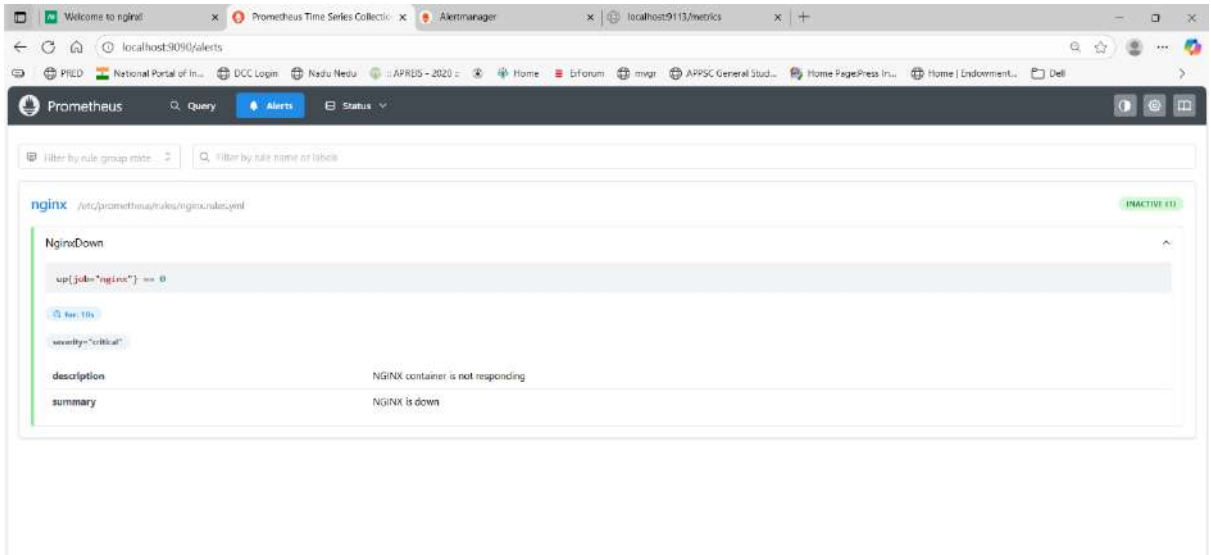


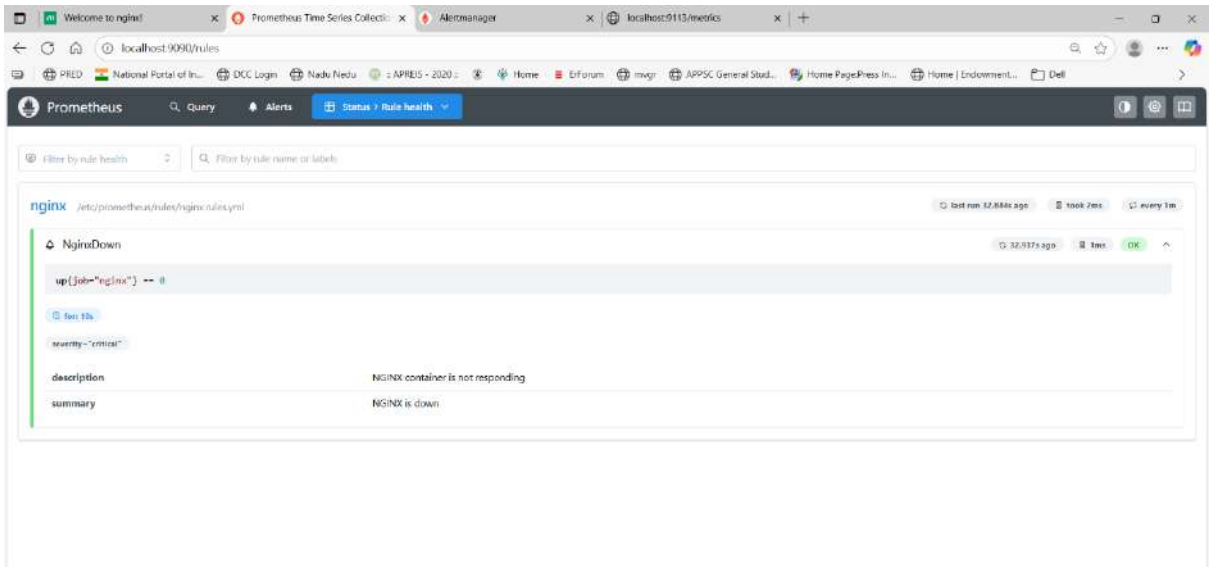
We can see playbook is automatically triggered and ran the playbook successfully

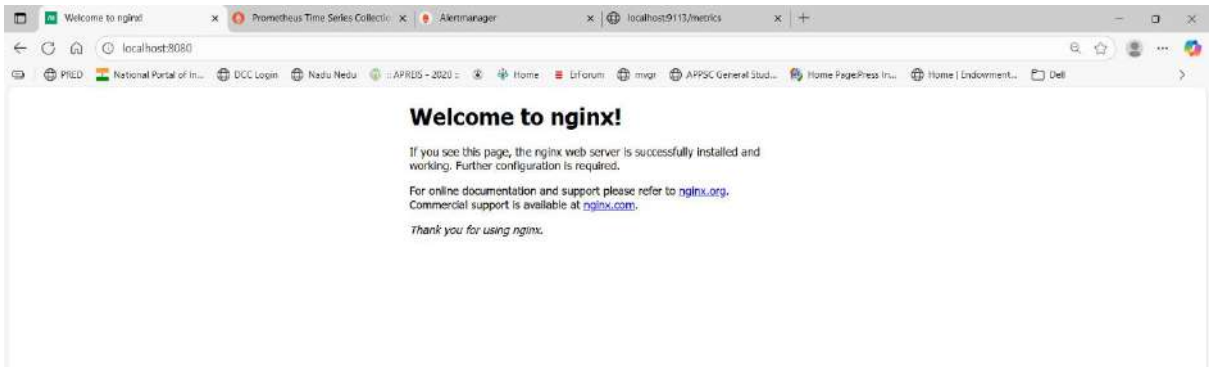


We can see automatically started the nginx-service

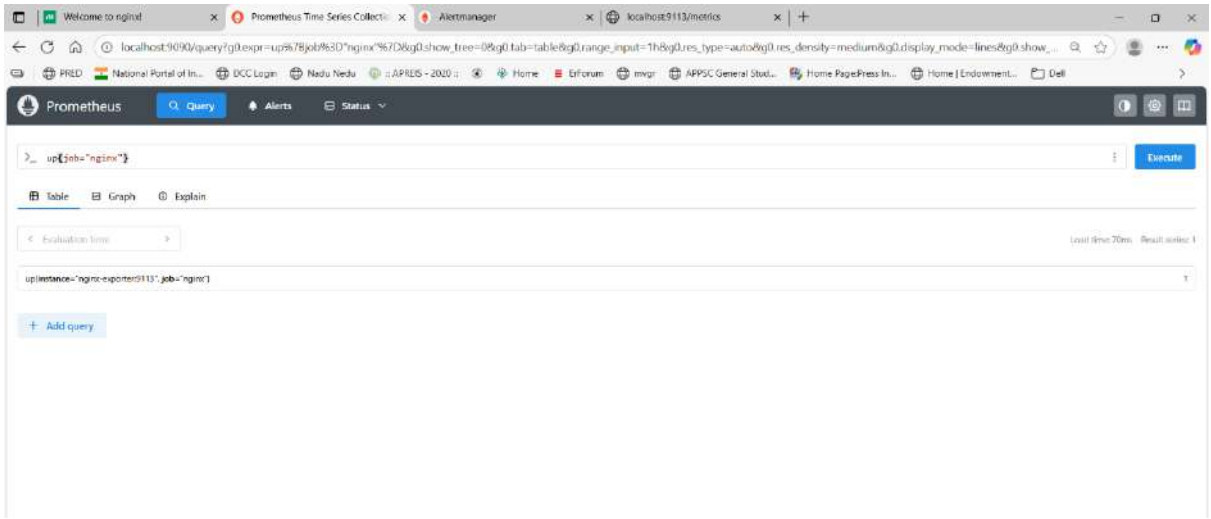
Alerts firing is stopped because nginx is started

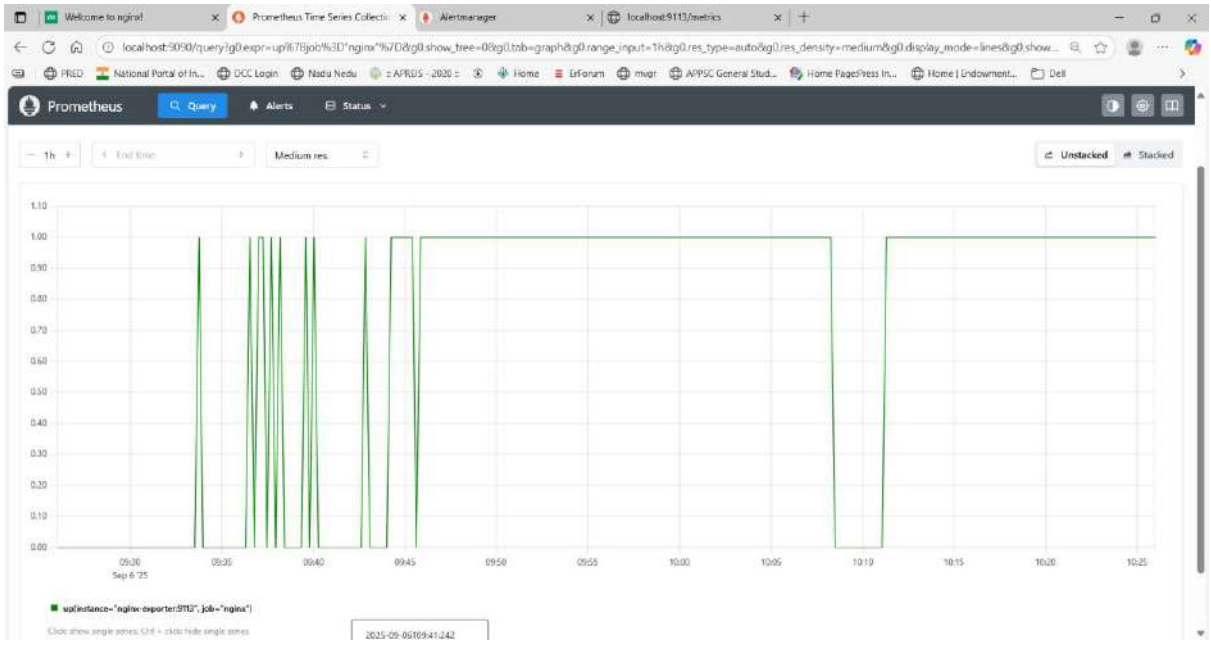




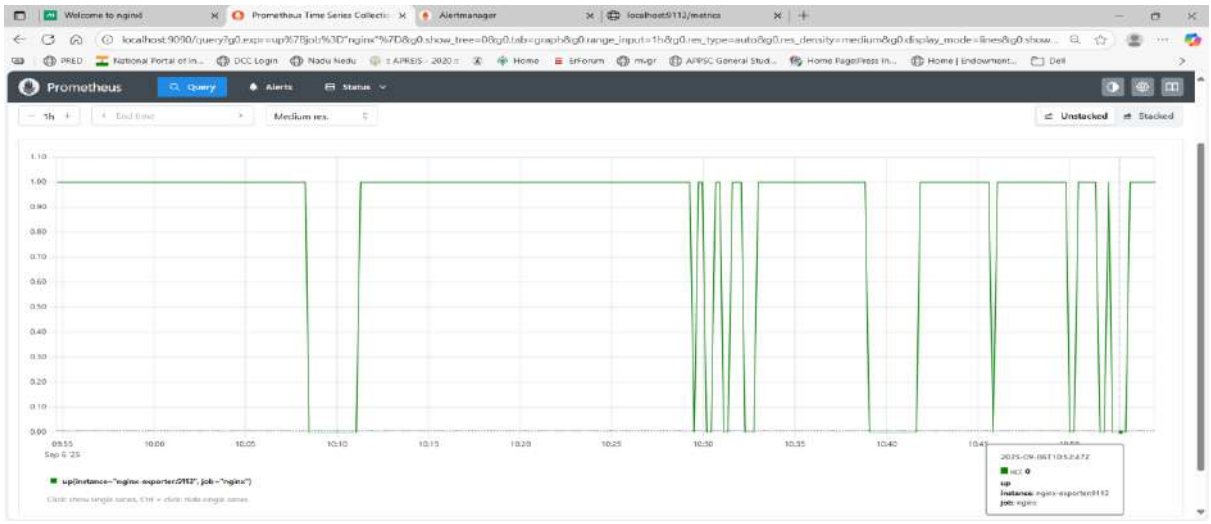


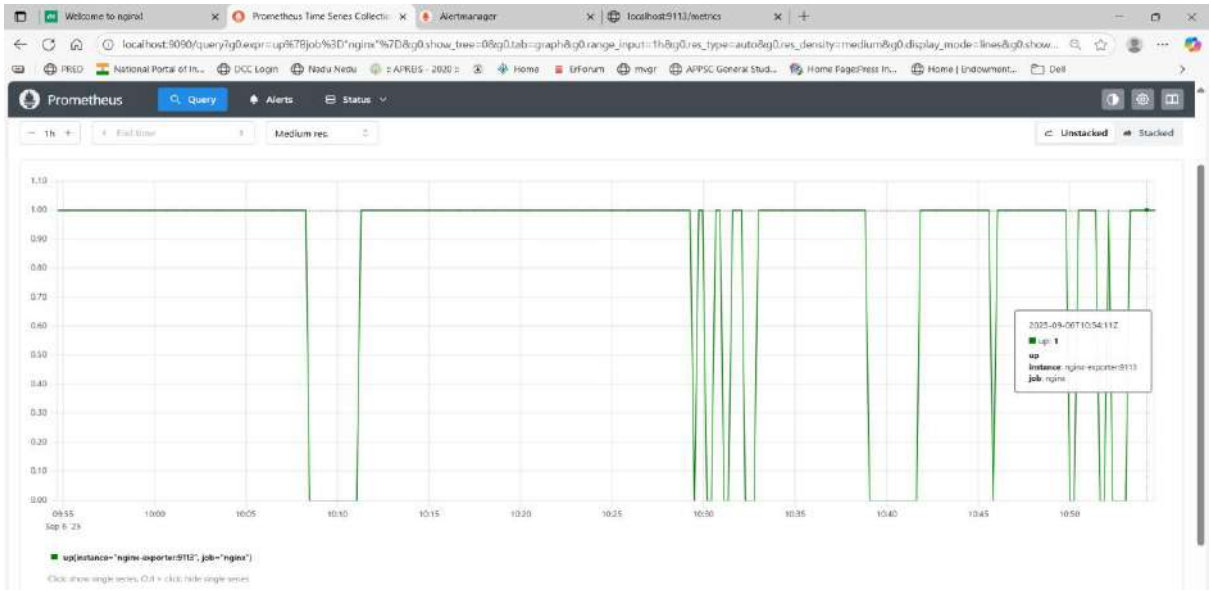
Query of nginx job up



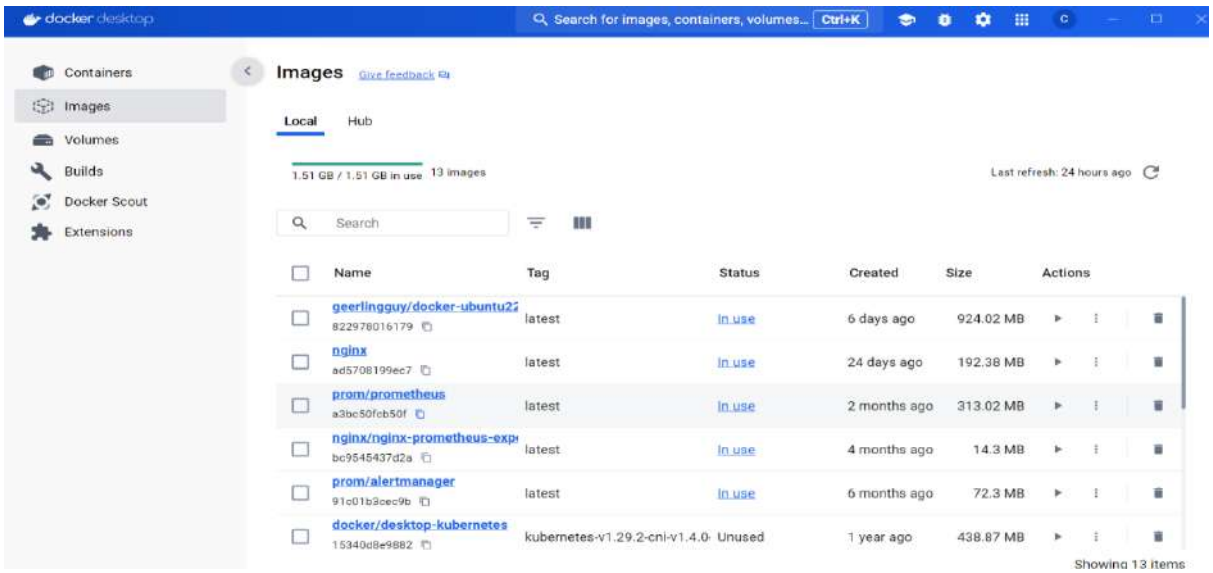


We see nginx down means 0

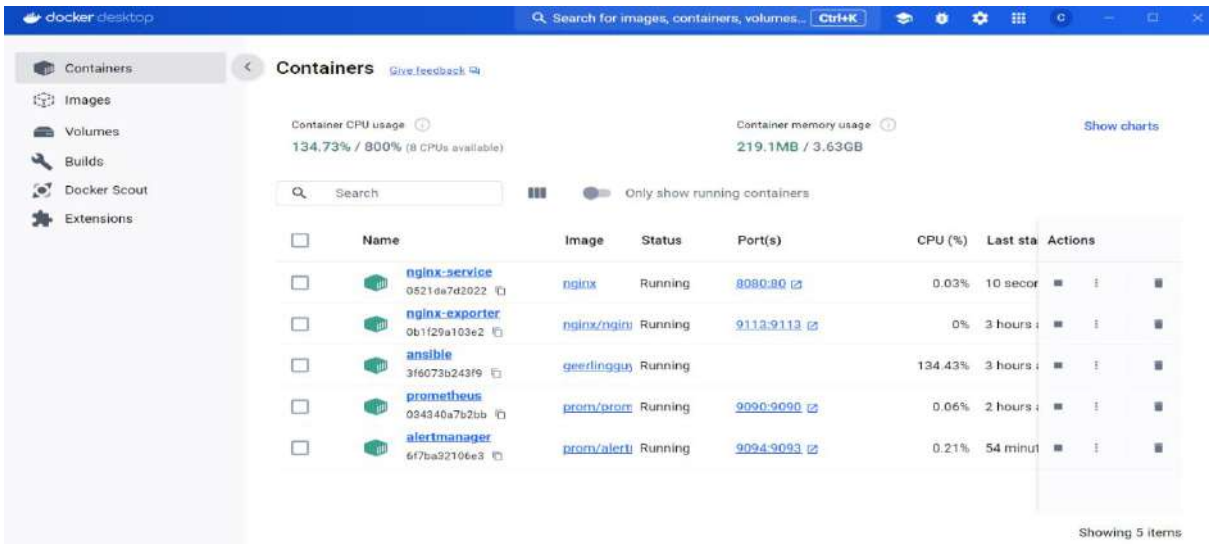




All images are pulled from docker hub



All containers are running in docker desktop



key concepts:

- Service Monitoring → Prometheus scrapes metrics (via NGINX Exporter) to check if the service is up.
- Alerting Rules → Prometheus rules detect when NGINX goes down and trigger alerts.
- Alert Management → Alertmanager receives alerts and forwards them to a webhook.
- Webhook Integration → A Python Flask app listens for alerts from Alertmanager.
- Automation with Ansible → On receiving alerts, the webhook triggers an Ansible playbook.
- Self-Healing → Ansible automatically restarts the failed NGINX container without manual intervention.
- End-to-End Flow → NGINX failure → Prometheus detects → Alertmanager triggers → Webhook calls Ansible → Ansible restarts NGINX → Service restored.

Troubleshooting

- File Mounting Issues → Wrong file paths while mounting Prometheus /Alertmanager configs.

  Fixed by correcting absolute paths in Docker Compose.

- Ansible Not Restarting Containers → Error because Docker was not accessible inside Ansible.

  Fixed by mounting Docker socket and binary into the Ansible container.

- Alertmanager Port Mismatch → Alerts didn't reach Alertmanager.

  Fixed by ensuring correct port mapping (9093).

- Webhook Not Receiving Alerts → Flask webhook didn't get triggered.

  Fixed by pointing to host.docker.internal and verifying connectivity.

- Alerts Not Firing as Expected → Confusion between rule *health* and *firing*

  Verified alert conditions and tested with NGINX down scenario.

Conclusion:

This project successfully implemented a self-healing infrastructure where Prometheus monitored the NGINX service, Alertmanager triggered alerts on failures, and Ansible automatically restarted the container. It ensures high availability and reduces downtime through automated recovery.