

# 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](https://github.com/vommidapuchinni/selfhealing)

## Project Overview

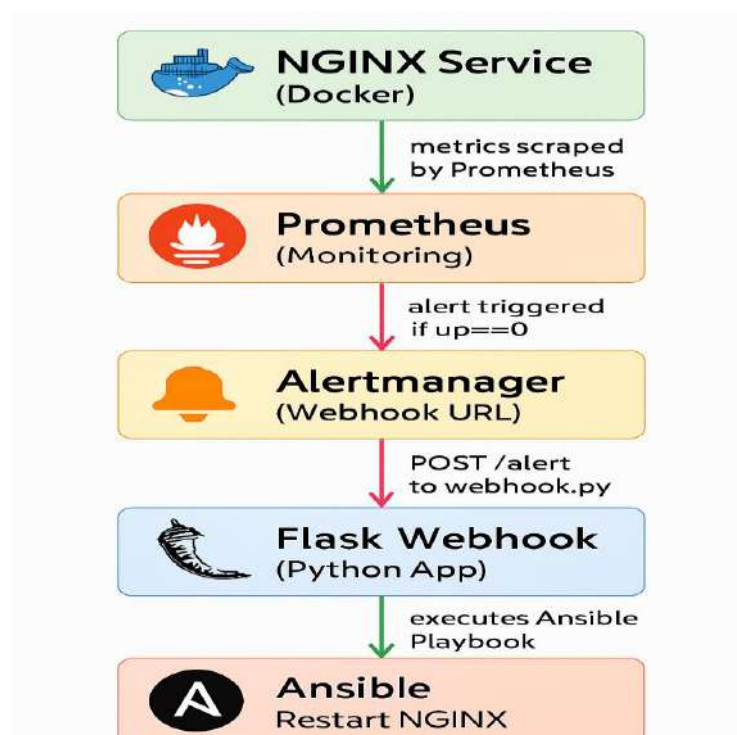
### Objective:

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:

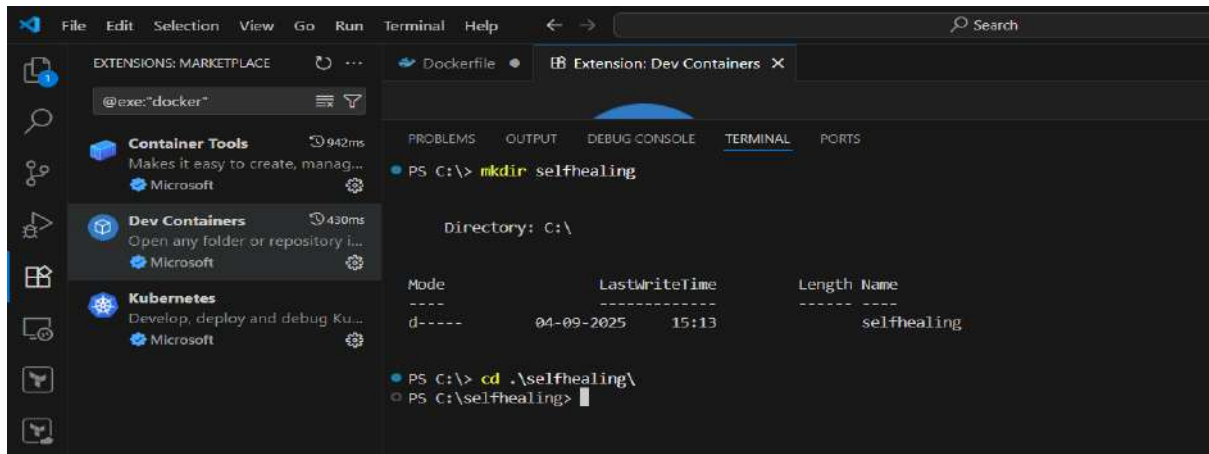
- Prometheus scrapes metrics from NGINX exporter.

- 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

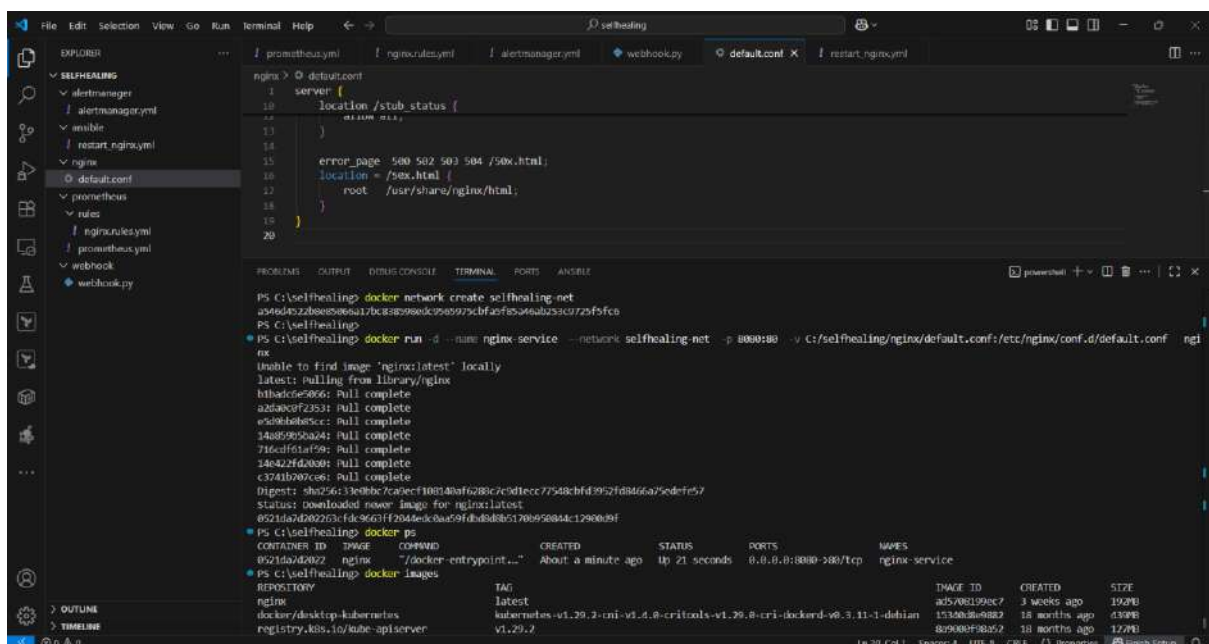
Create a custom network so all containers can communicate.

docker network create selfhealing-net

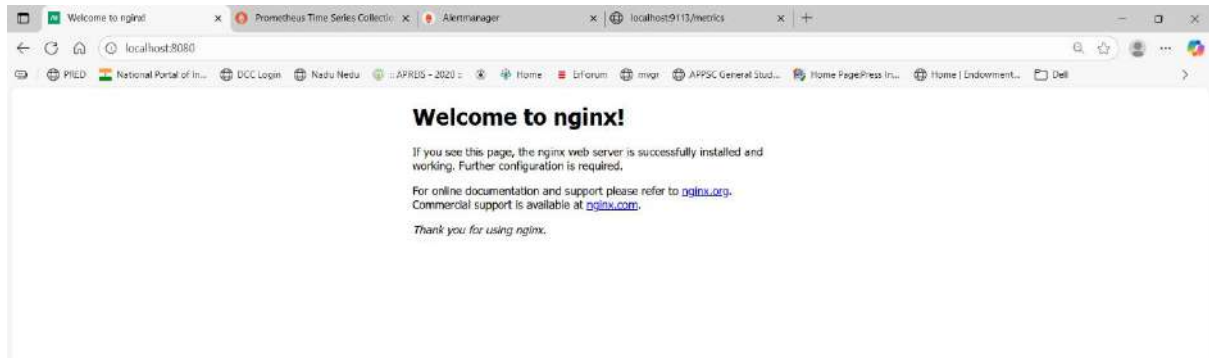
This network will be used by Prometheus, Alertmanager, and NGINX.

### Deploy Sample Service (NGINX)

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

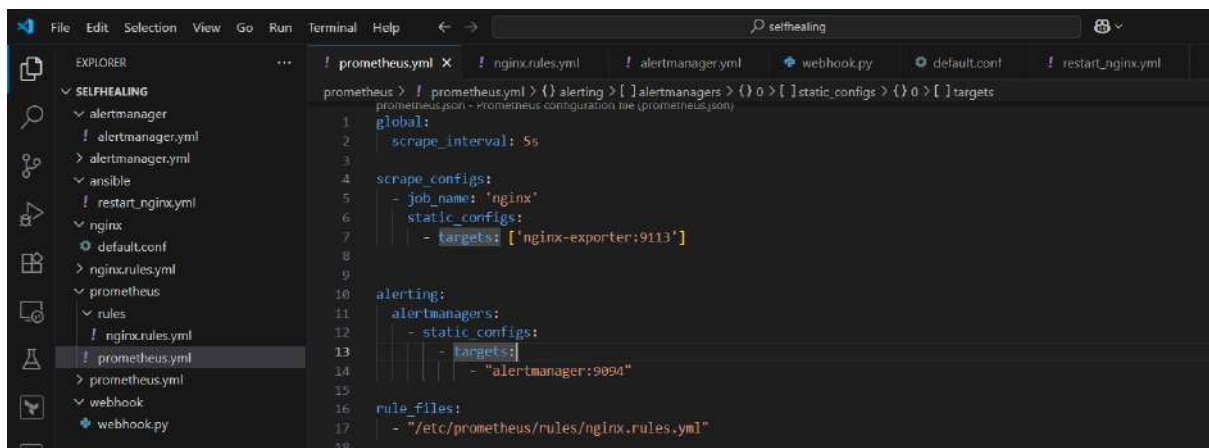


Test it: Open browser → <http://localhost:8080> → Should see NGINX welcome page.



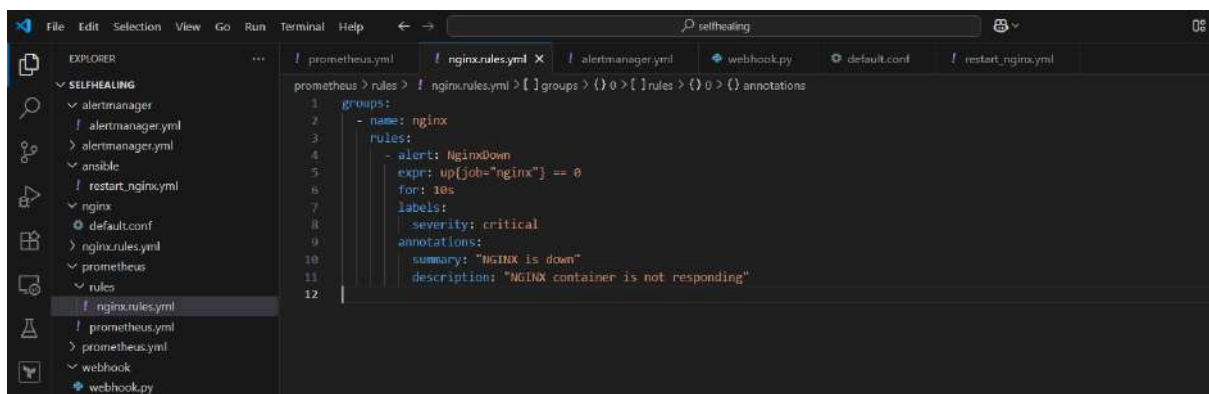
## Prepare Prometheus Configuration

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`



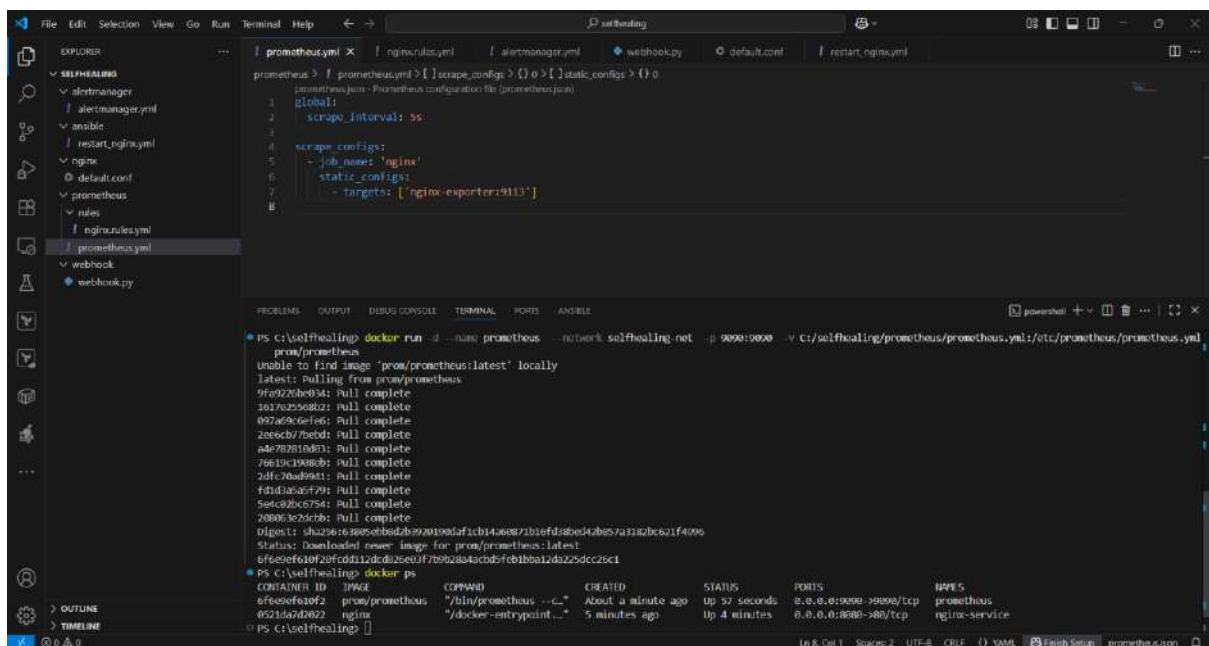
- Defines an alert rule group called nginx.
- Rule NginxDn triggers when `up{job="nginx"} == 0` (NGINX not running).
- Condition must hold for 10 seconds before firing.
- Alert is labeled critical with summary and description for clarity.

## Run Prometheus Container

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

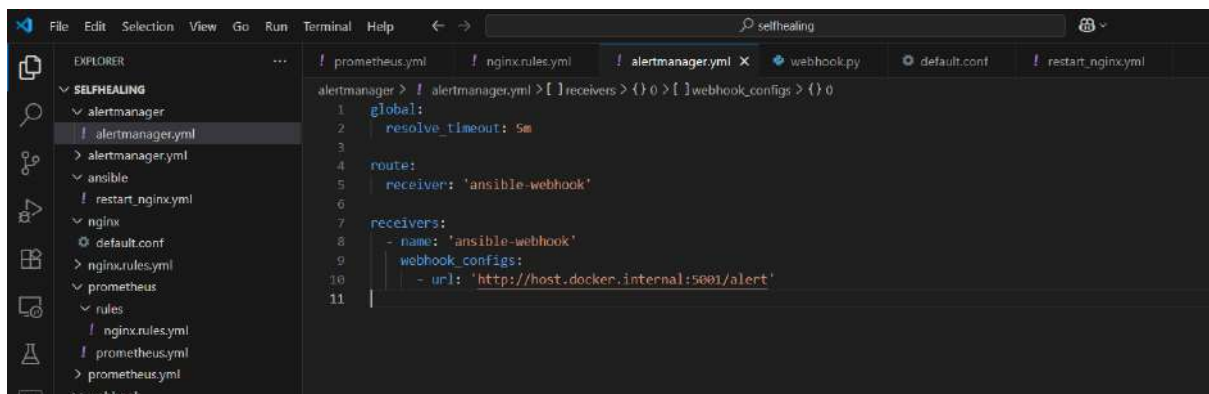
`-v /c/selfhealing/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml`

`-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`

The screenshot shows a VS Code editor with the following files open: `prometheus.yml`, `nginxrules.yml`, `alertmanager.yml`, `webhook.py`, `default.conf`, and `restart_nginx.yml`. The `alertmanager.yml` file is selected and shows the following configuration:

```
1 global:
2   resolve_timeout: 5m
3
4 route:
5   receiver: 'ansible-webhook'
6
7 receivers:
8   - name: 'ansible-webhook'
9     webhook_configs:
10    - url: 'http://host.docker.internal:5001/alert'
11
```

The terminal window shows the following commands and output:

```
PS C:\selfhealing> docker run -d --name alertmanager --network selfhealing-net -p 9094:9093 -v C:\selfhealing\alertmanager\alertmanager.yml:/etc/alertmanager/alertmanager.yml prom/alertmanager:latest
Unable to find image 'prom/alertmanager:latest' locally
latest: Pulling from prom/alertmanager
9fa928b894: Already exists
1017e2908d: Already exists
545c1465415: Pull complete
6fcbf65877c: Pull complete
a0c7f411696f: Pull complete
5759234525c5: Pull complete
4f4fb700f54: Pull complete
Digest: sha256:27a425d5f8156cab1d5c1ba4251ac7e8567746a2483ff264516437a99615ba
Status: Downloaded newer image for prom/alertmanager:latest
5516cf85f0dc8e52ab9789991f370b1d8badf86b7cf6cd2b2e284eth6f

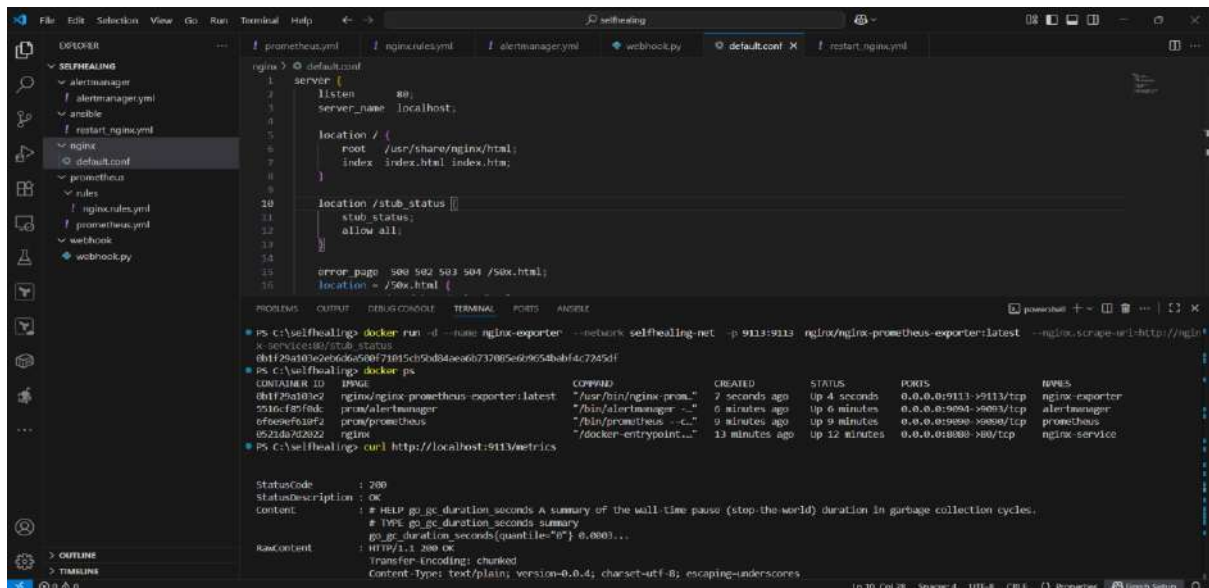
PS C:\selfhealing> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
5516cf85f0dc   prom/alertmanager   "/bin/alertmanager -..."   14 seconds ago   Up 8 seconds   0.0.0.0:9094->9093/tcp             alertmanager
6f6e96f610f2   prom/prometheus     "/bin/prometheus --c..."   3 minutes ago   Up 2 minutes   0.0.0.0:9090->9090/tcp             prometheus
8521da46002    nginx              "/docker-entrypoint..."   7 minutes ago   Up 6 minutes   0.0.0.0:8080->8080/tcp             nginx-service
```

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

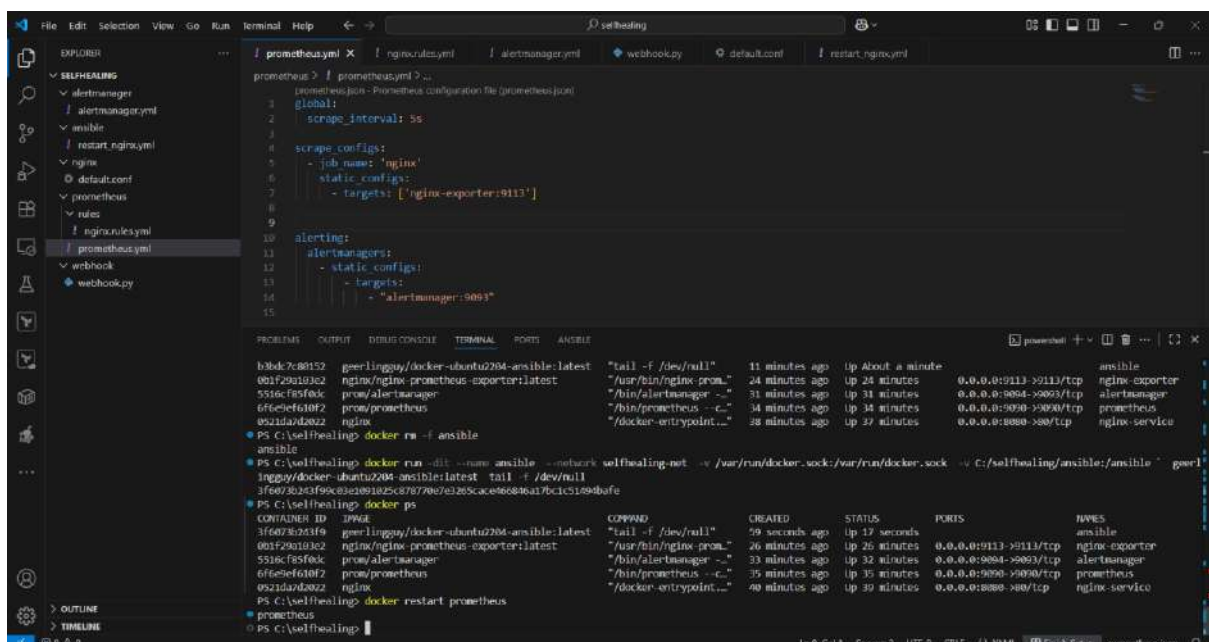
gather\_facts: false

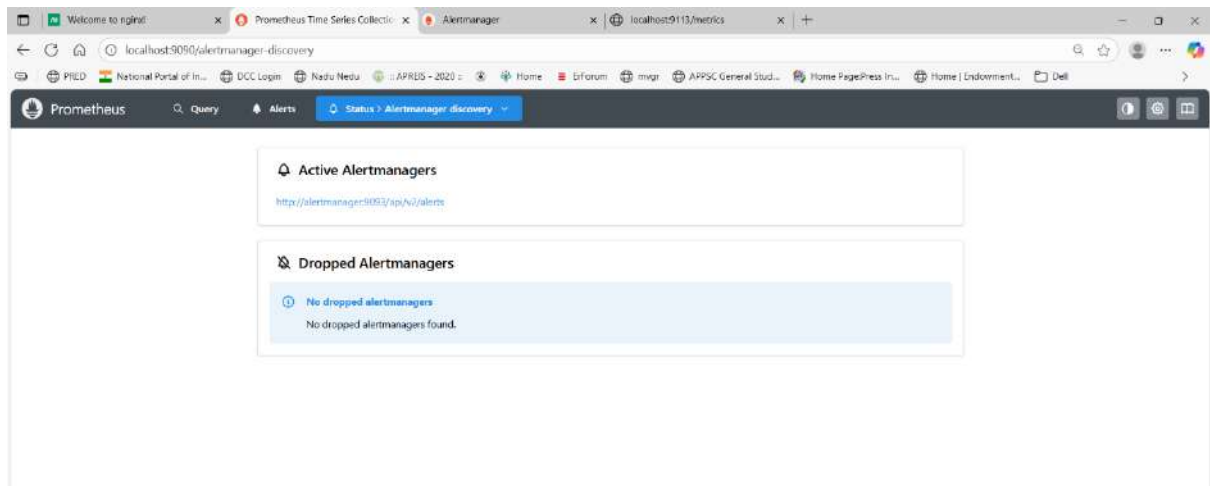
tasks:

- name: Restart NGINX Docker container

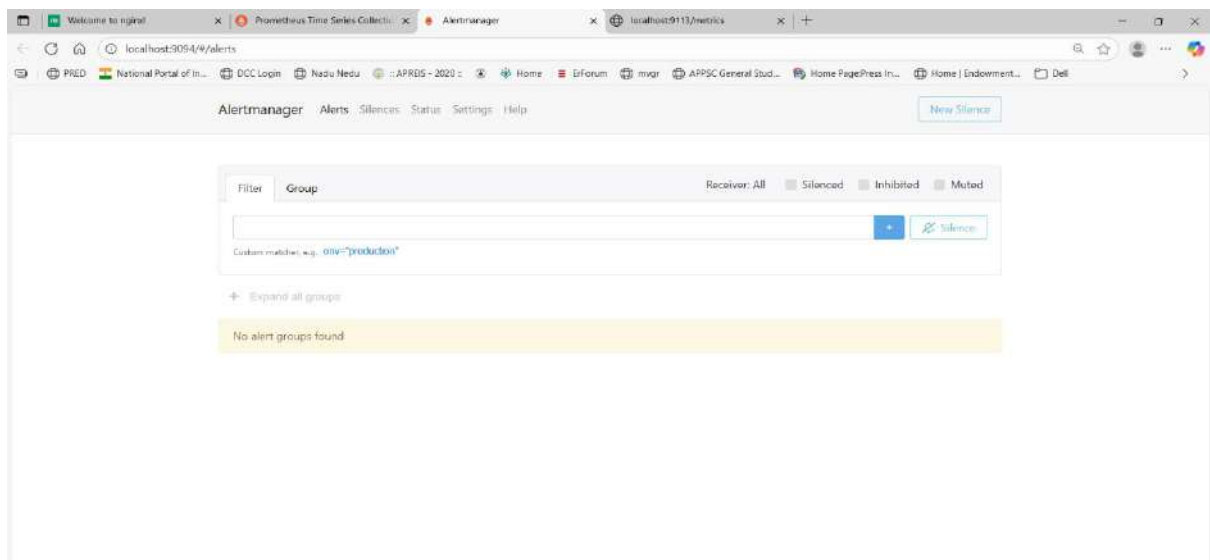
shell: docker restart nginx-service

it will automatically restarts the nginx-service

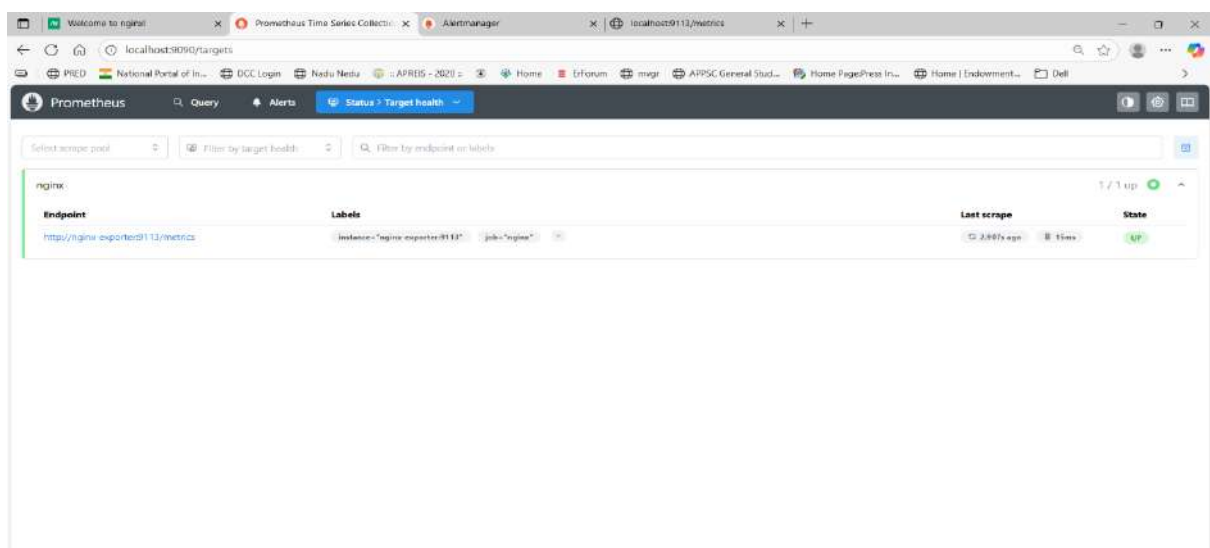




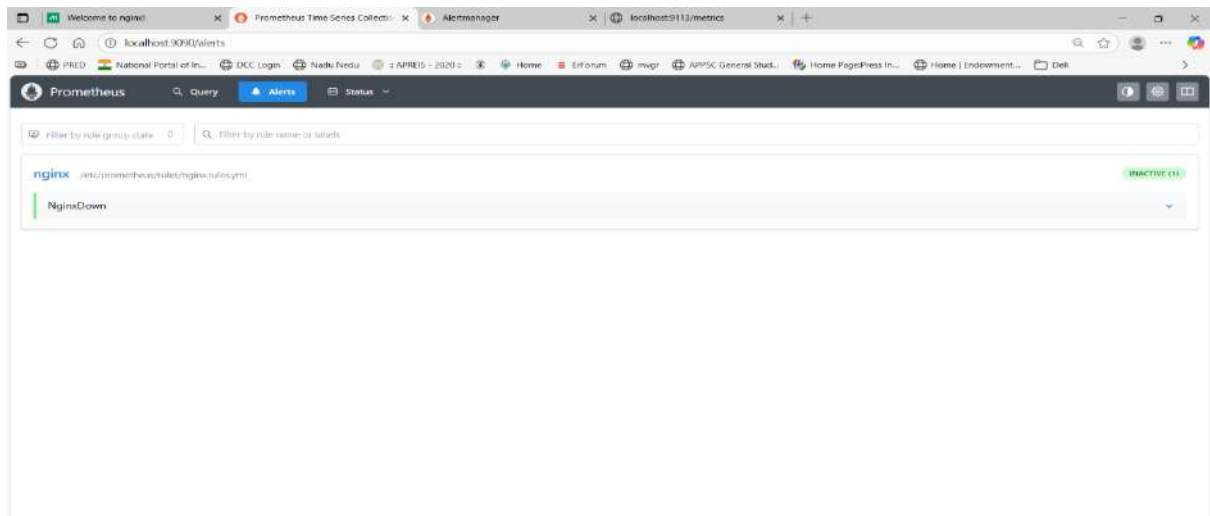
## Alertmanager



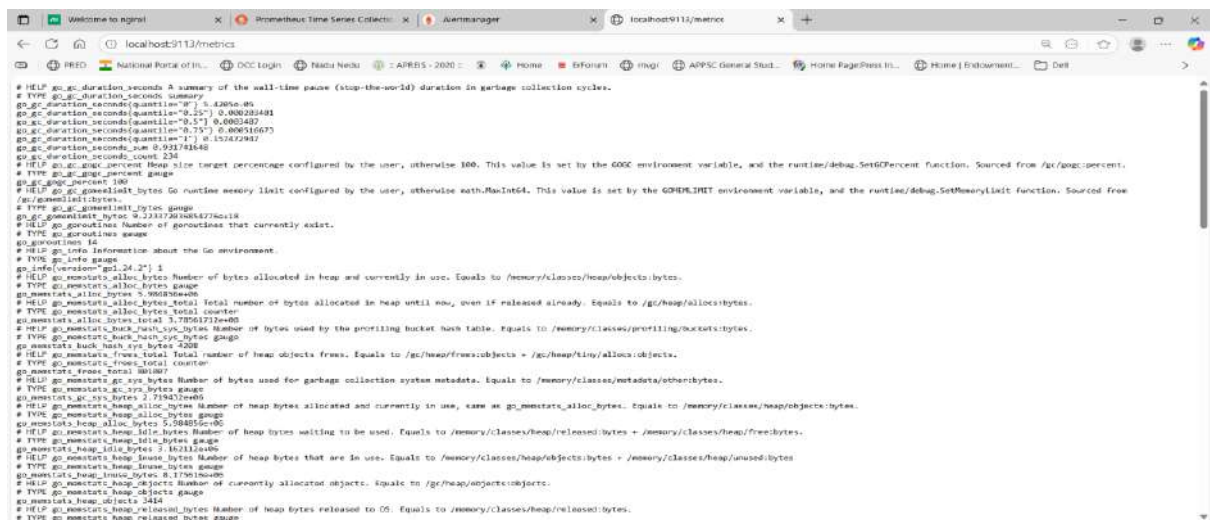
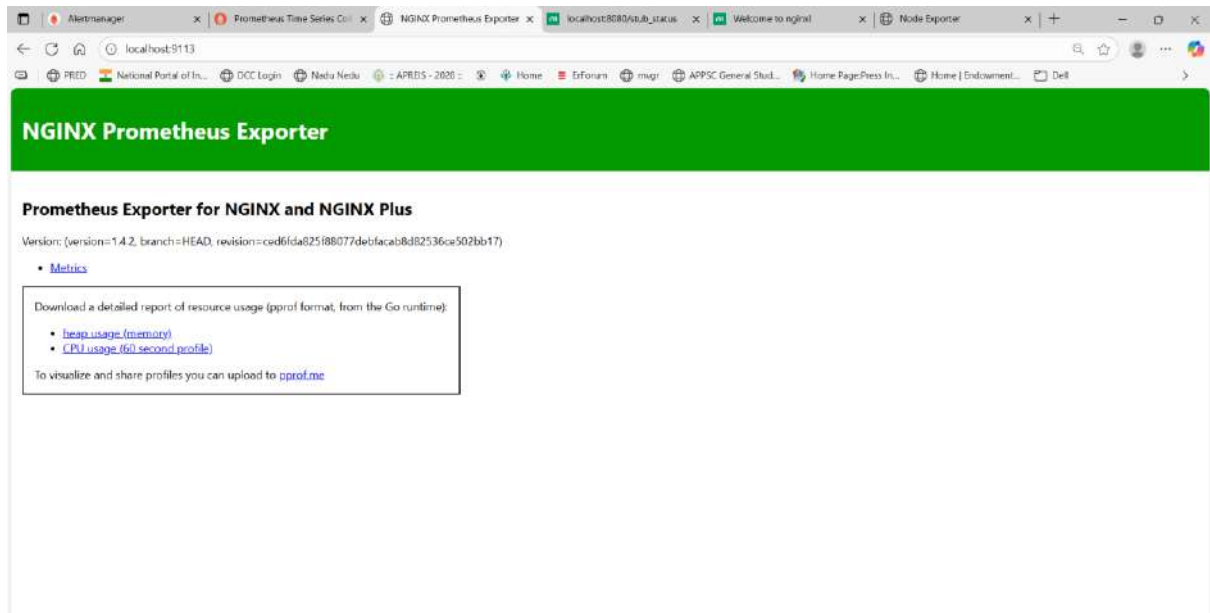
## Prometheus target health



# Alerts



# Nginx metrics





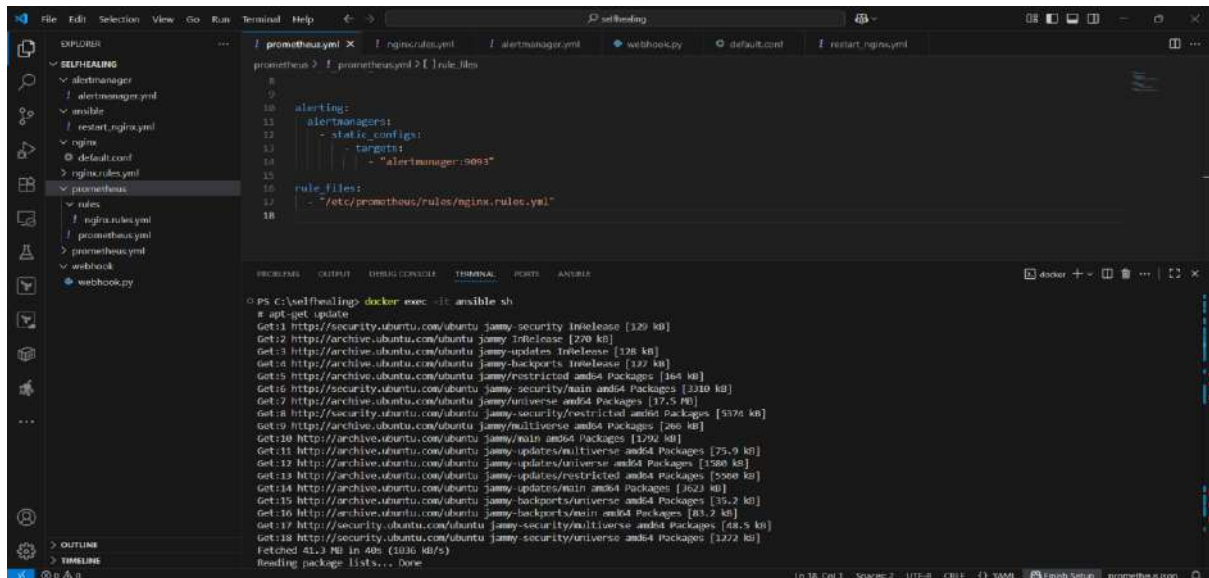
## Install Docker CLI inside the Ansible container

docker exec -it ansible sh

apt-get update

apt-get install docker.io

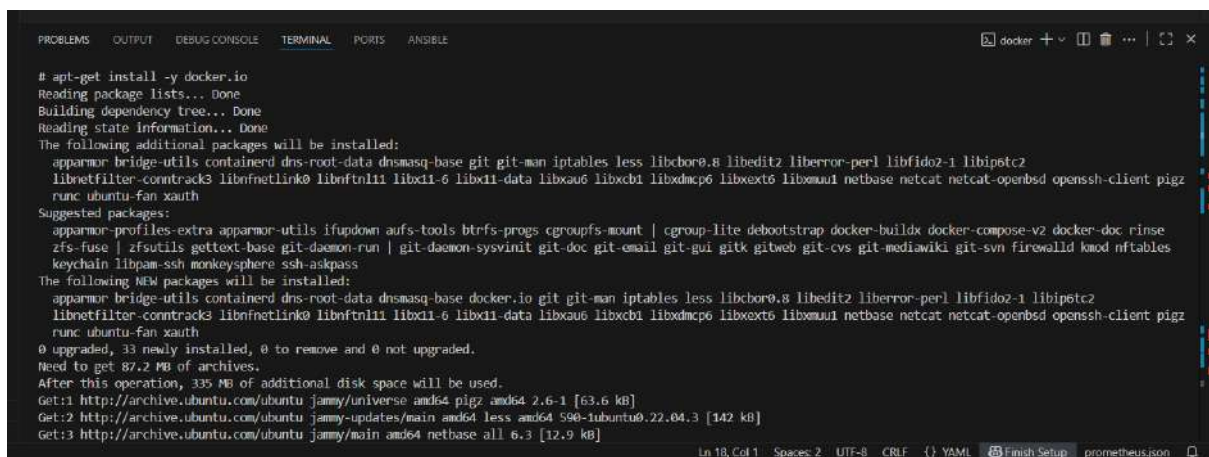
check docker version: docker version



The screenshot shows a VS Code editor with a Prometheus configuration file open. The configuration includes an alerting section with a static\_configs block for targets and a rule\_files block pointing to a local directory. Below the configuration, a terminal window shows the output of the command `docker exec -it ansible sh`. The terminal displays the output of `apt-get update`, showing the process of updating package lists from various Ubuntu repositories.

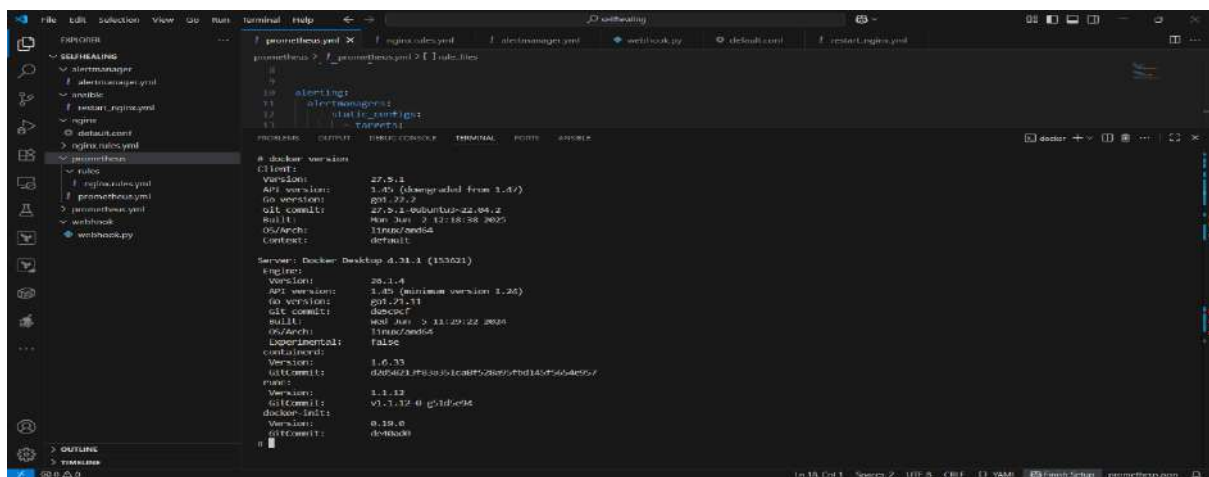
```
prometheus.yml
10 alerting:
11   alertmanagers:
12     - static_configs:
13       - targets:
14         - "alertmanager:9093"
15
16 rule_files:
17   - "/etc/prometheus/rules/nginx.rules.yml"
18
```

```
PS C:\wellbeing> docker exec -it ansible sh
# apt-get update
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [644 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [3310 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [5376 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1294 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [75.9 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1580 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [5080 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [6023 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [35.2 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [83.2 kB]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [68.5 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [1222 kB]
Retrieved 41.2 MB in 40s (1040 kB/s)
Reading package lists... Done
```



The screenshot shows a terminal window in VS Code displaying the output of the command `apt-get install -y docker.io`. The output shows the installation of Docker and its dependencies, including `apparmor`, `bridge-utils`, `containerd`, `dnsmasq-base`, `git`, `git-man`, `iptables`, `less`, `libbcore0.8`, `libedit2`, `liberror-perl`, `libfido2-1`, `libipset2`, `libnetfilter-conntrack3`, `libnetfilter-link`, `libnetfilter-tls`, `libxdmcp6`, `libxext6`, `libxmuu1`, `netbase`, `netcat`, `netcat-openbsd`, `openssh-client`, `pigz`, `runc`, and `ubuntu-fan`. The terminal also shows the suggested packages and the disk space requirements for the installation.

```
# apt-get install -y docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apparmor bridge-utils containerd dnsmasq-base git git-man iptables less libbcore0.8 libedit2 liberror-perl libfido2-1 libipset2
  libnetfilter-conntrack3 libnetfilter-link libnetfilter-tls libxdmcp6 libxext6 libxmuu1 netbase netcat netcat-openbsd openssh-client pigz
  runc ubuntu-fan xauth
Suggested packages:
  apparmor-profiles-extra apparmor-utils ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debotstrap docker-buildx docker-compose-v2 docker-doc rinse
  zfs-fuse | zfsutils gettext-base git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn firewallld kmod nftables
  keychain libpam-ssh monkeysphere ssh-askpass
The following NEW packages will be installed:
  apparmor bridge-utils containerd dnsmasq-base docker.io git git-man iptables less libbcore0.8 libedit2 liberror-perl libfido2-1 libipset2
  libnetfilter-conntrack3 libnetfilter-link libnetfilter-tls libxdmcp6 libxext6 libxmuu1 netbase netcat netcat-openbsd openssh-client pigz
  runc ubuntu-fan xauth
0 upgraded, 33 newly installed, 0 to remove and 0 not upgraded.
Need to get 87.2 MB of archives.
After this operation, 335 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 less amd64 590-1ubuntu0.22.04.3 [142 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 netbase all 6.3 [12.9 kB]
```



The screenshot shows a terminal window in VS Code displaying the output of the command `docker version`. The output shows the Docker client and server versions, along with the engine version and other details.

```
# docker version
Client:
Version: 27.5.1
API version: 1.45 (downgraded from 1.47)
Go version: go1.21.2
Git commit: 27.5.1-ubuntu22.04.2
Built: Mon Jun 12 18:38:20 2023
OS/Arch: linux/amd64
Context: default

Server: Docker Desktop 4.31.1 (153021)
Engine:
Version: 28.1.4
API version: 1.45 (minimum version 1.24)
Go version: go1.21.11
Git commit: d80c06f
Built: Mon Jun 12 18:29:22 2023
OS/Arch: linux/amd64
Experimental: false
containers:
Version: 1.6.3
Git commit: 4056231f33a351c089f52809f9bd145f464d0e7
Name:
Version: 1.1.12
Git commit: v1.1.12.0-g51d0e94
docker-init:
Version: 0.19.0
Git commit: d04bada
```

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

```
PS C:\selfhealing> docker exec -it ansible ansible-playbook /ansible/restart_nginx.yml

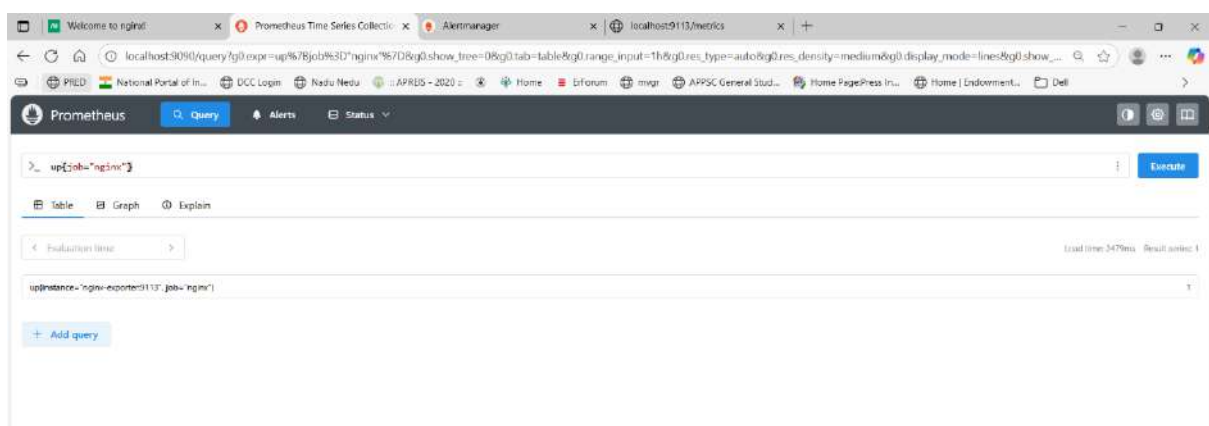
PLAY [localhost] *****

TASK [Restart NGINX Docker container] *****
[WARNING]: Platform linux on host localhost is using the discovered Python interpreter at /usr/bin/python3.10, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.17/reference_appendices/interpreter_discovery.html for more information.
changed: [localhost]

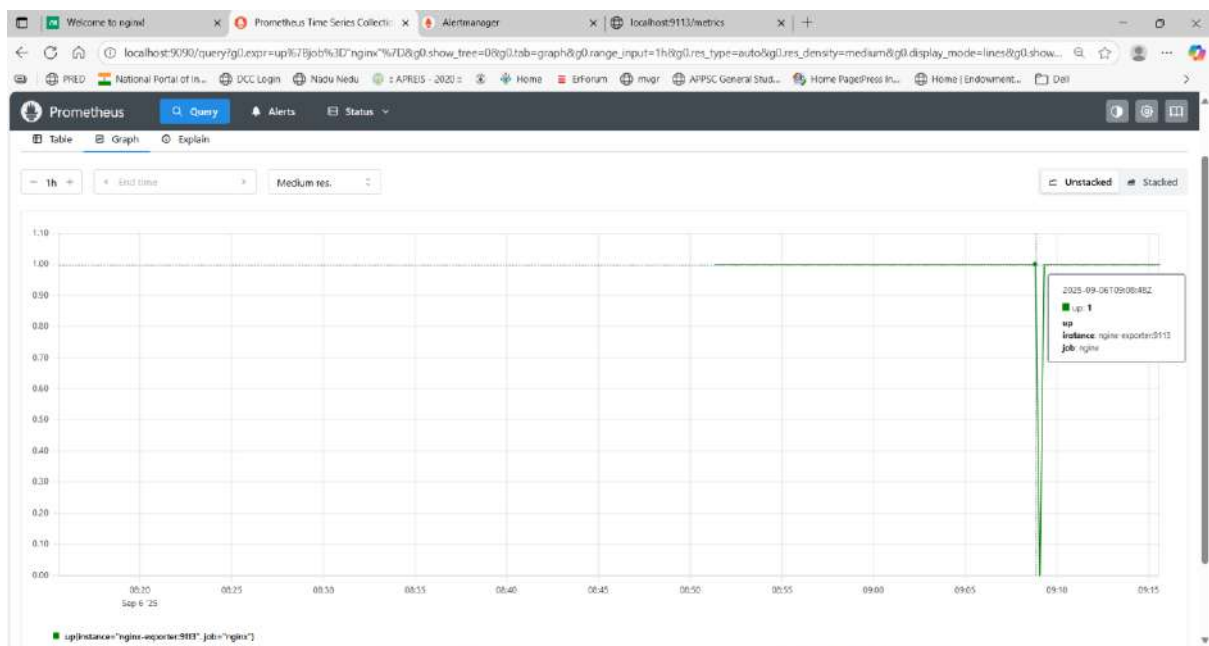
PLAY RECAP *****
localhost                : ok=1  changed=1  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

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>
```

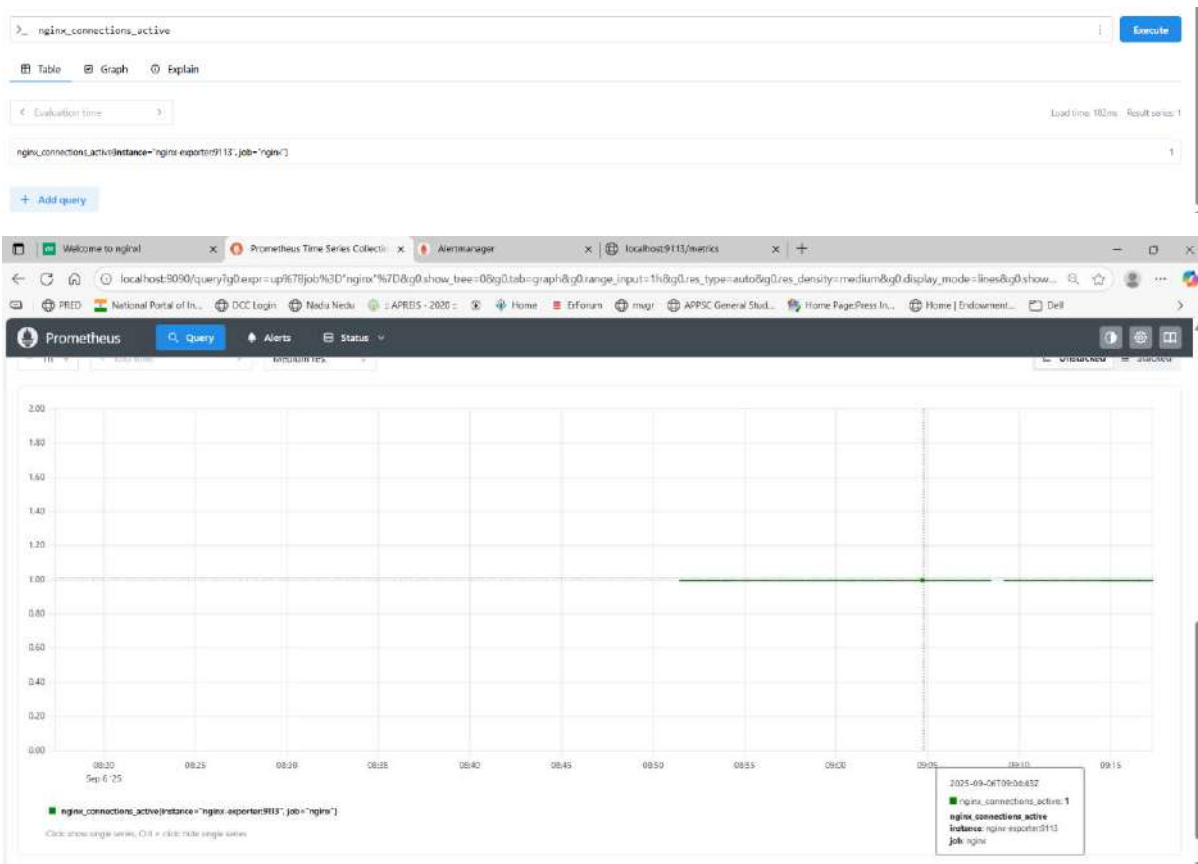
**We see nginx job by executing the query we see table that nginx is running**



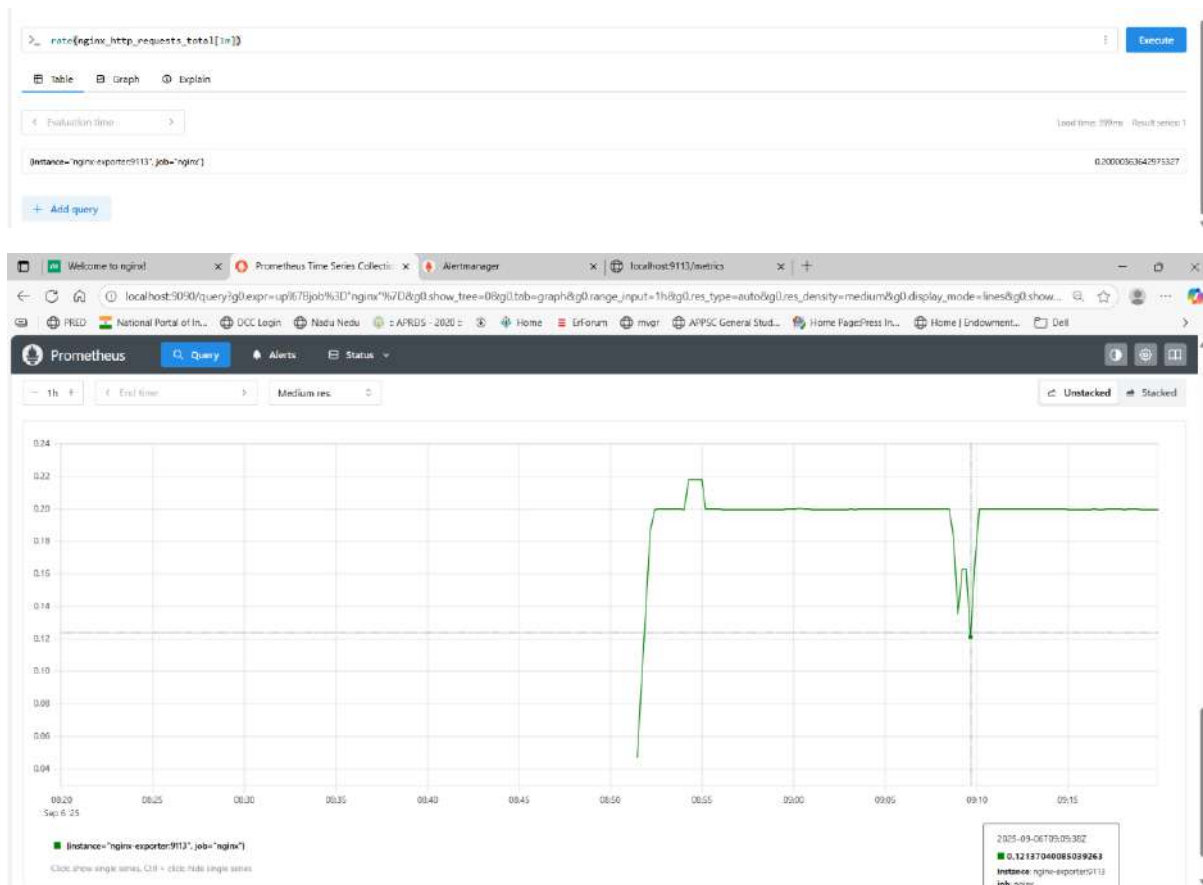
**Graph of that query**



**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`

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

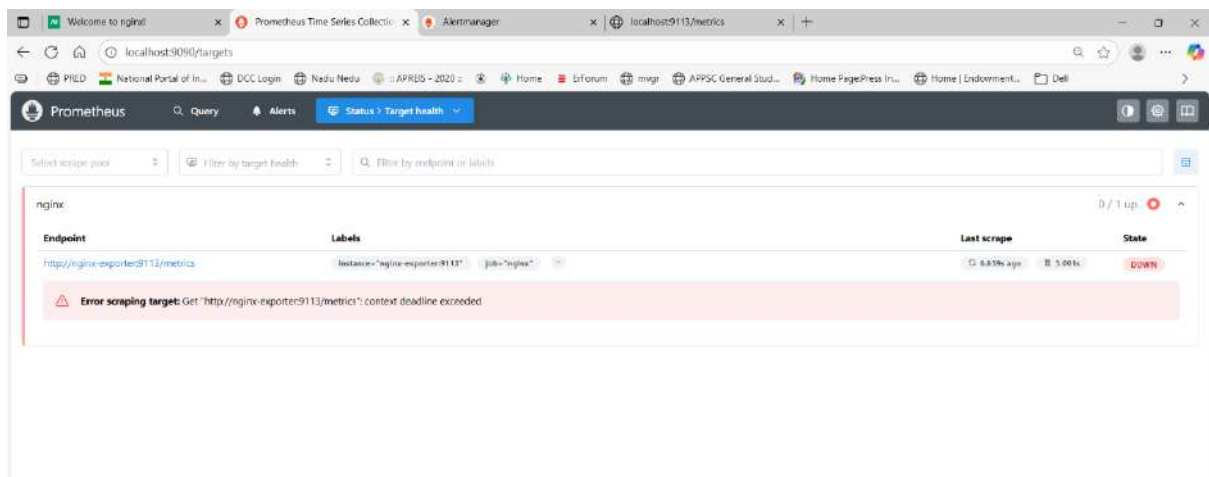
What's next:
Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug ansible
Learn more at https://docs.docker.com/go/debug-cli/

PS C:\selfhealing> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
034348a7b2bb   prom/prometheus                     "/bin/prometheus --c..." 33 minutes ago Up 33 minutes 0.0.0.0:9090->9090/tcp              prometheus
3f6873b243f9   geerlingguy/docker-ubuntu2204-ansible:latest "tail -f /dev/null"       About an hour ago Up About an hour 0.0.0.0:9113->9113/tcp              nginx-exporter
001f29a103e2   nginx/nginx-prometheus-exporter:latest "/usr/bin/nginx-prom..." 2 hours ago   Up 2 hours   0.0.0.0:9094->9093/tcp              alertmanager
5516cf85fddc   prom/alertmanager                   "/bin/alertmanager --..." 2 hours ago   Up 2 hours   0.0.0.0:8880->880/tcp              nginx-service
0521da7d2022   nginx                               "/docker-entrypoint..." 2 hours ago   Up 15 minutes                               nginx-service

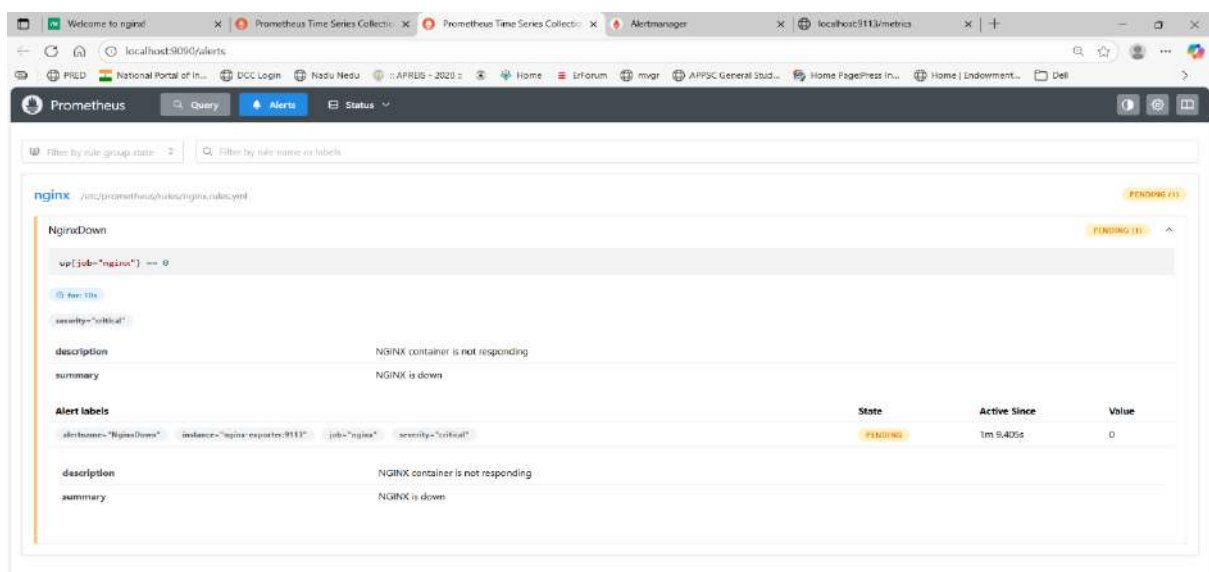
PS C:\selfhealing> docker stop nginx-service
nginx-service
PS C:\selfhealing>
```

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

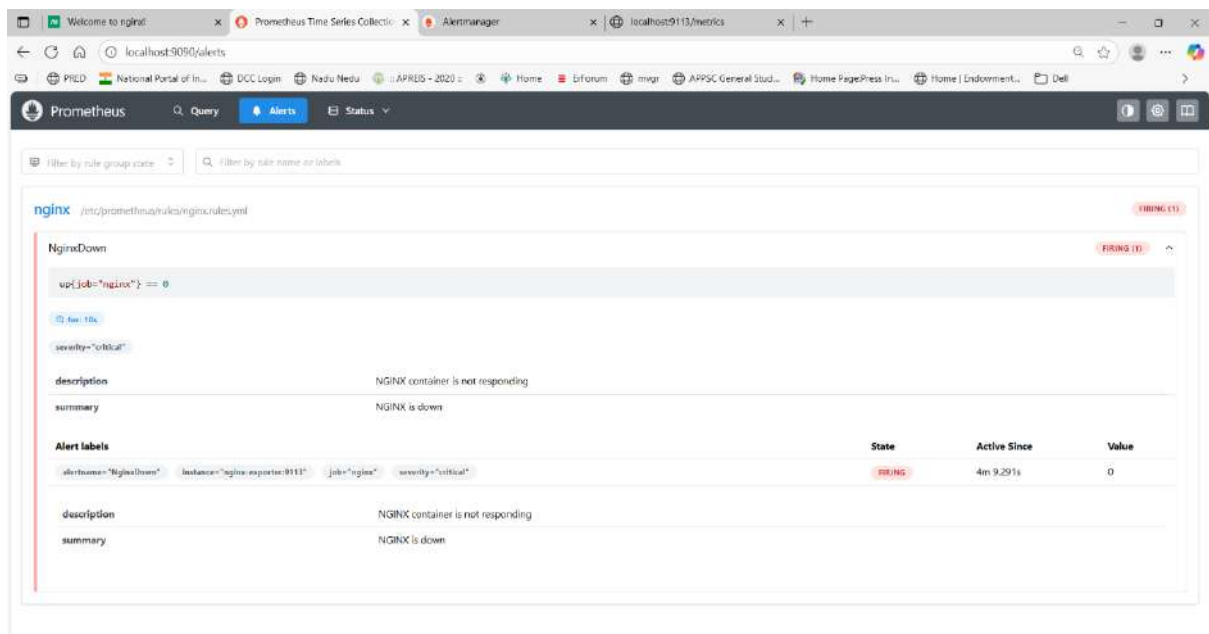
This is Prometheus target health is down because we stopped the nginx



We can see alert is triggered and showing pending



## We see that alert is firing



The screenshot shows the Prometheus Alerts page. The alert 'NginxDown' is in a 'FIRING (1)' state. The alert definition is as follows:

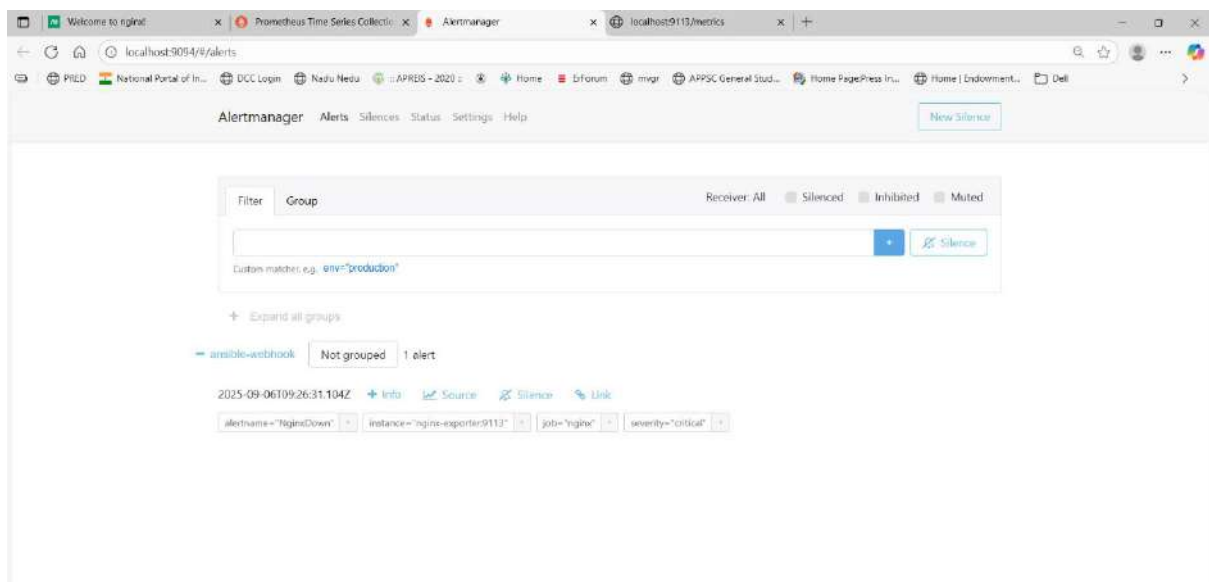
```
up{job="nginx"} == 0
```

Alert labels:

Alert labels	State	Active Since	Value
alertname="NginxDown", instance="nginx-exporter-9113", job="nginx", severity="critical"	FIRING	4m 9.291s	0

Description: NGINX container is not responding  
Summary: NGINX is down

## In Alert manager we see nginx down is triggered



The screenshot shows the Alertmanager Alerts page. The alert 'NginxDown' is triggered. The alert details are as follows:

Filter: Group Receiver: All Silenced Inhibited Muted

Custom matcher, e.g. `env="production"`

Expand all groups

amiable-webhook Not grouped 1 alert

2025-09-06T09:26:31.104Z + Info + Source + Silence + Link

alertname="NginxDown", instance="nginx-exporter-9113", job="nginx", severity="critical"

## We see nginx is not accessed



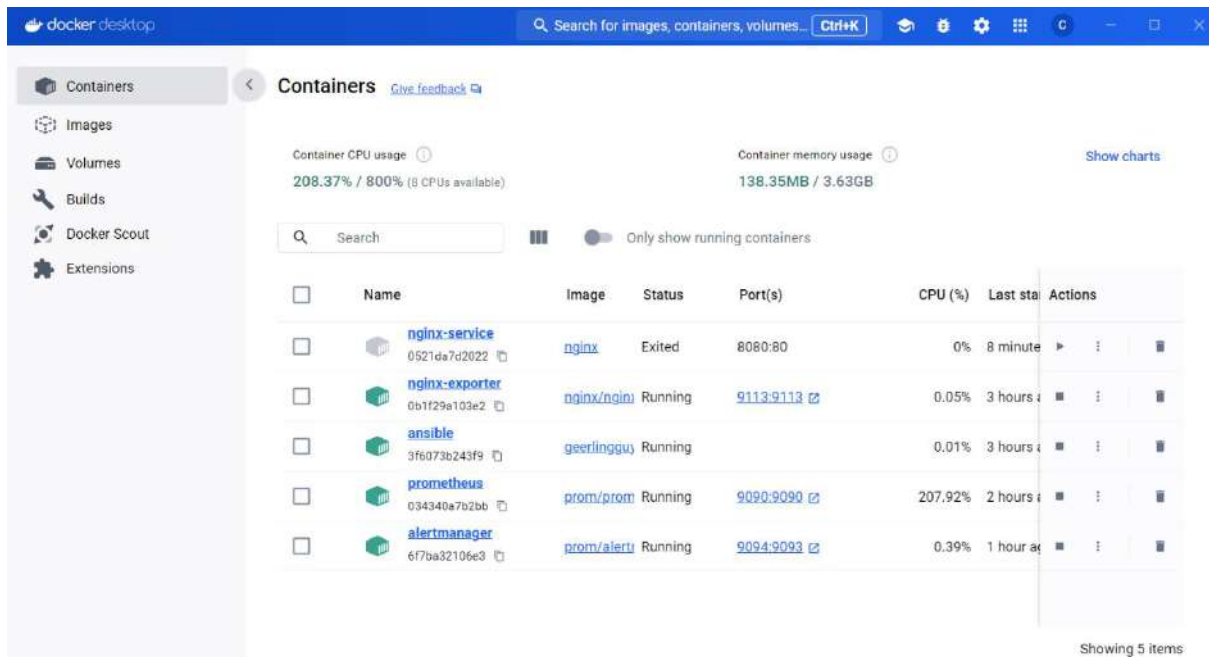
The screenshot shows a web browser with the URL `localhost:8080`. The page displays a 404 Not Found error:

### Access Error: 404 -- Not Found

Cannot locate document: /



## We see nginx is stopped that we see it was exited



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

```

PS C:\selfhealing> python C:\selfhealing\webhook\webhook.py
* Serving Flask app 'webhook'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5001
* Running on http://10.286.243.159:5001
Press CTRL+C to quit
{'receiver': 'ansible-webhook', 'status': 'firing', 'alerts': [{'status': 'firing', 'labels': {'alertname': 'NginxDnwn', 'instance': 'nginx-exporter:9113', 'job': 'nginx', 'severity': 'critical'}, 'annotations': {'description': 'NGINX container is not responding', 'summary': 'NGINX is down', 'startsAt': '2025-09-09T10:09:31.109Z', 'endsAt': '0001-01-01T00:00:00Z', 'generatorURL': 'http://0343f0a7b2bb:9090/graph7g0.expr=up478job43DA22nginx42247D+43DA3D+86g0.tab=1' 'fingerprints': {'09b0ec83a2bf4a'}}], 'group_labels': {}, 'commonLabels': {'alertname': 'NginxDnwn', 'instance': 'nginx-exporter:9113', 'job': 'nginx', 'severity': 'critical'}, 'commonAnnotations': {'description': 'NGINX container is not responding', 'summary': 'NGINX is down', 'externalURL': 'http://6f7ba32106a3:9093', 'version': '4', 'groupKey': '{}:{}'. 'truncatedAlerts': 0}

PLAY [localhost] *****

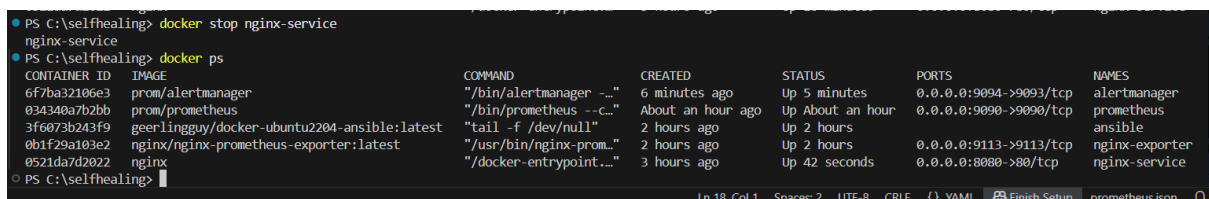
TASK [Restart NGINX Docker container] *****
[WARNING]: Platform linux on host localhost is using the discovered Python
interpreter at /usr/bin/python3.10, but future installation of another Python
interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
changed: [localhost]

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

127.0.0.1 -- [06/Sep/2025 15:41:10] "POST /alert HTTP/1.1" 200 -

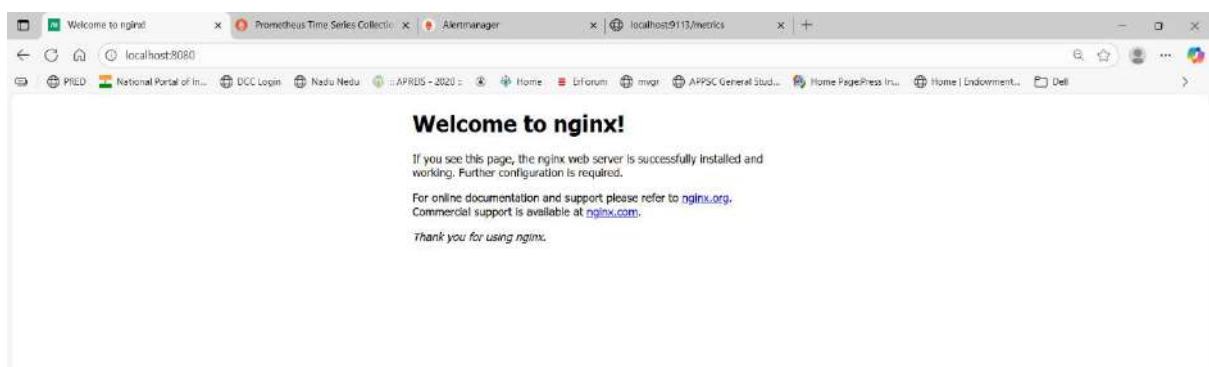
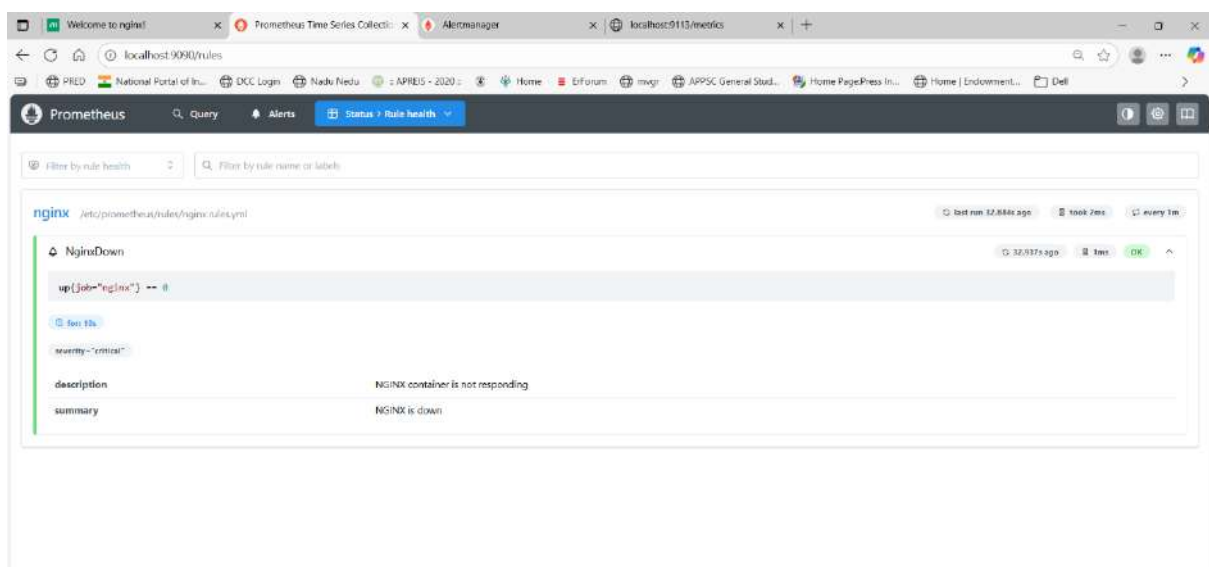
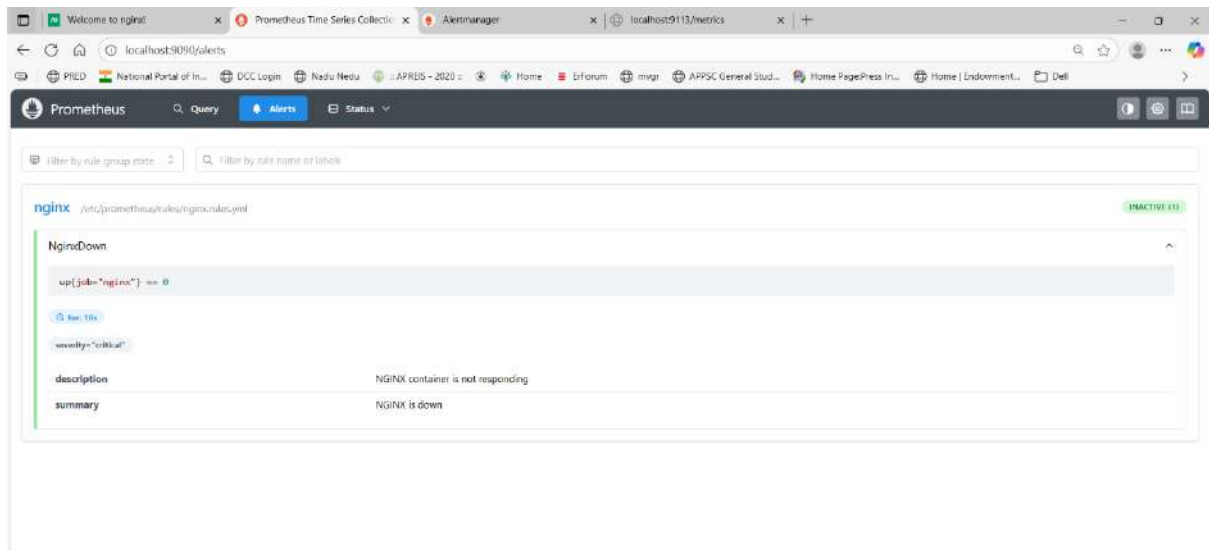
```

**We can see automatically started the nginx-service**

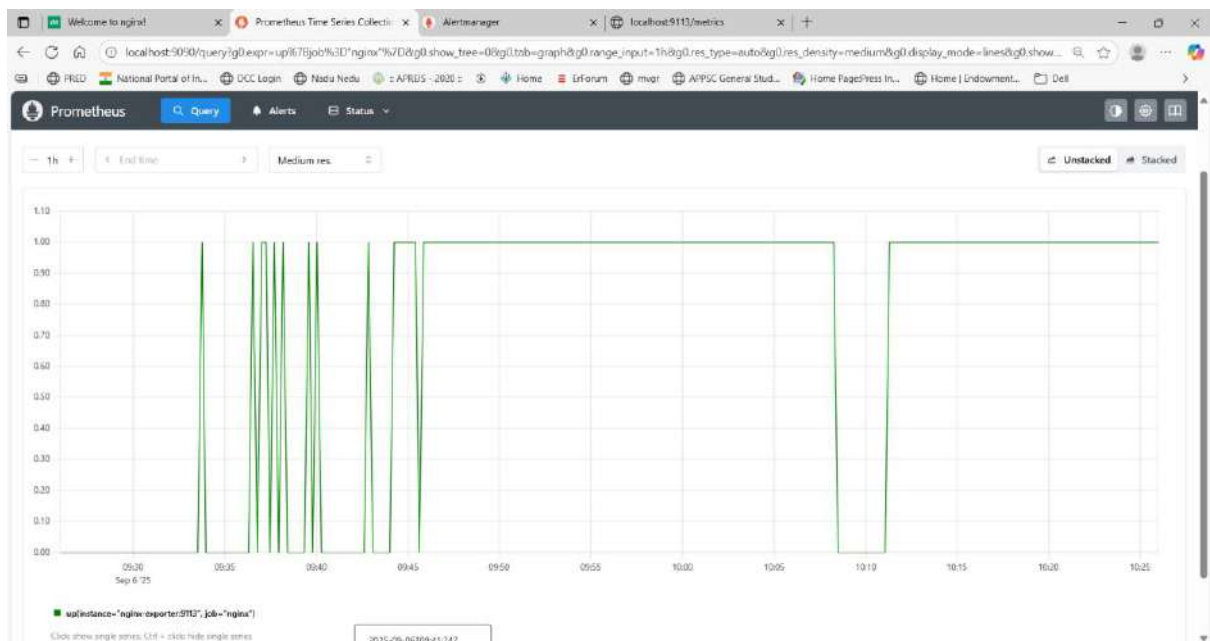
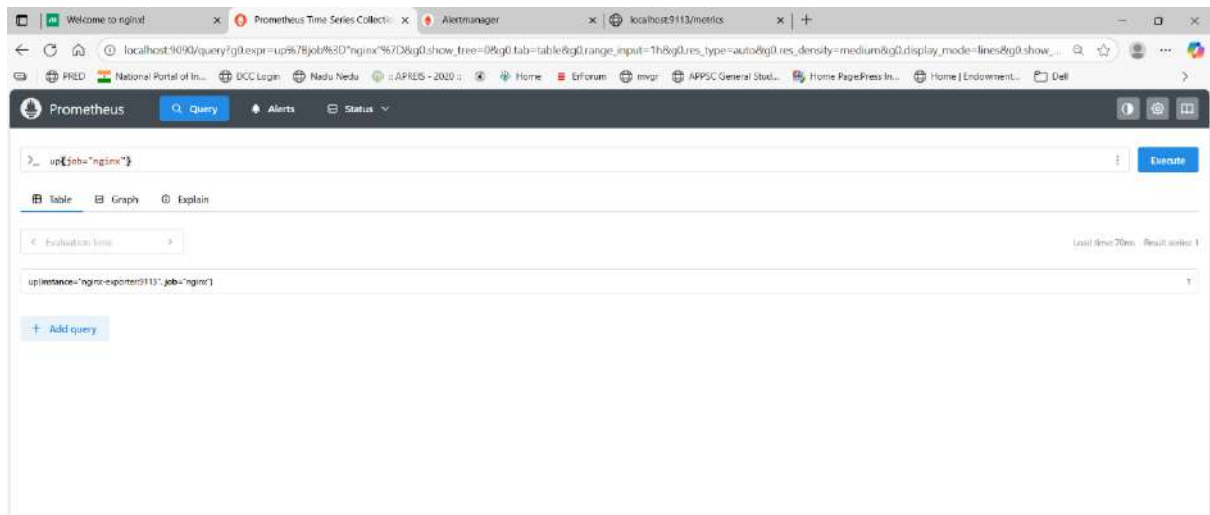


## Alerts firing is stopped because nginx is started

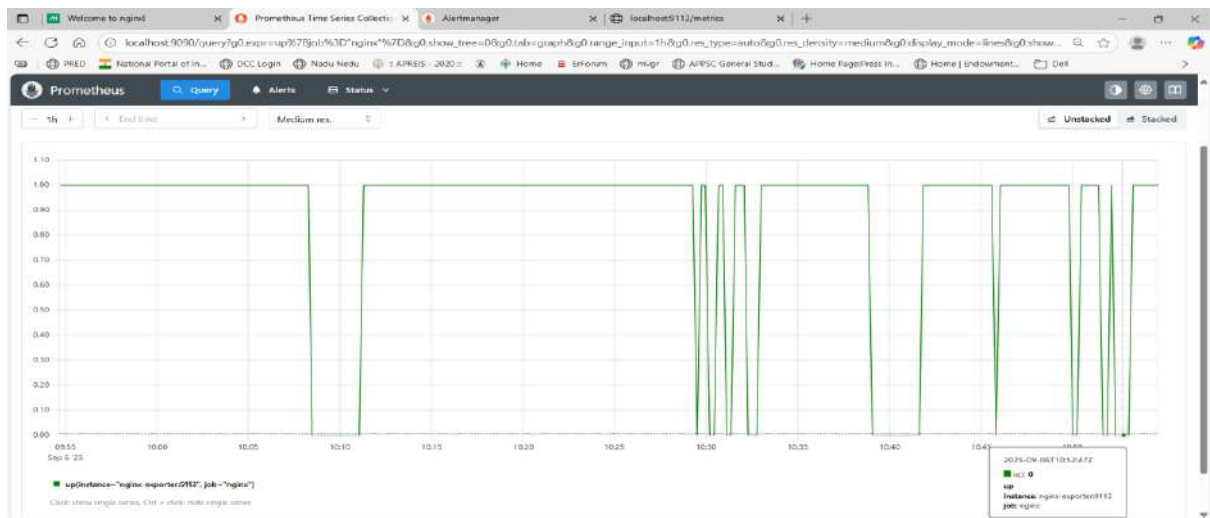


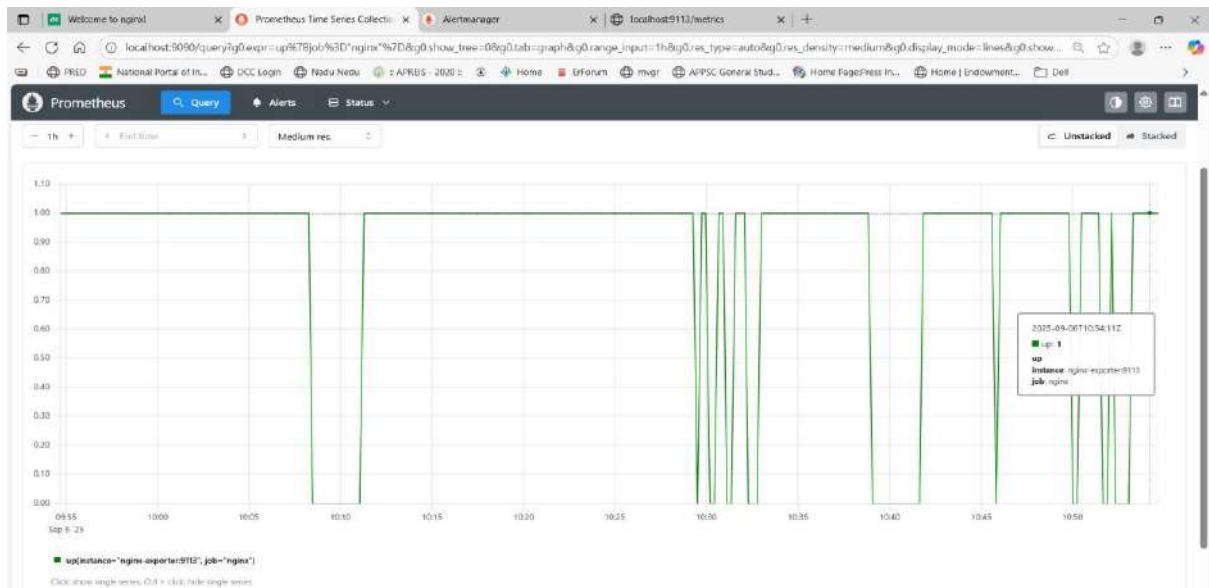


**Query of nginx job up**



We see nginx down means 0





All images are pulled from docker hub

The image shows the Docker Desktop 'Images' tab. The left sidebar contains navigation options: Containers, Images, Volumes, Builds, Docker Scout, and Extensions. The main area shows a list of local images. The table has columns: Name, Tag, Status, Created, Size, and Actions. The images listed are:

Name	Tag	Status	Created	Size	Actions
geerlingguy/docker-ubuntu22	latest	In use	6 days ago	924.02 MB	
nginx	latest	In use	24 days ago	192.38 MB	
prom/prometheus	latest	In use	2 months ago	313.02 MB	
nginx/nginx-prometheus-exp	latest	In use	4 months ago	14.3 MB	
prom/alertmanager	latest	In use	6 months ago	72.3 MB	
docker/desktop-kubernetes	kubernetes-v1.29.2-cni-v1.4.0-	Unused	1 year ago	438.87 MB	

All containers are running in docker desktop

The image shows the Docker Desktop 'Containers' tab. The left sidebar contains navigation options: Containers, Images, Volumes, Builds, Docker Scout, and Extensions. The main area shows a list of running containers. The table has columns: Name, Image, Status, Port(s), CPU (%), Last state, and Actions. The containers listed are:

Name	Image	Status	Port(s)	CPU (%)	Last state	Actions
nginx-service	nginx	Running	8080:80	0.03%	10 seconds	
nginx-exporter	nginx/nginx	Running	9113:9113	0%	3 hours	
ansible	geerlingguy	Running		134.43%	3 hours	
prometheus	prom/prom	Running	9090:9090	0.06%	2 hours	
alertmanager	prom/alert	Running	9094:9093	0.21%	54 minutes	

## 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.