

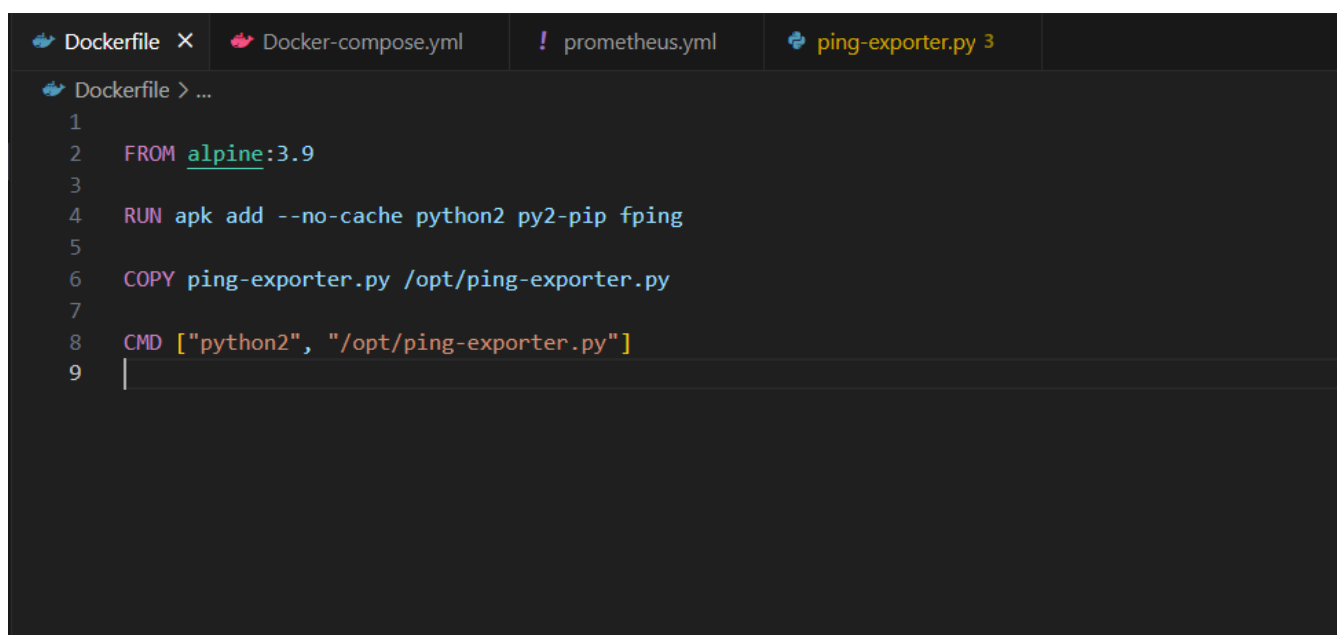
DOCUMENTATION TECHNIQUE : PROJET 3DCT

I- CONSTRUIRE L'EXPORTATEUR

Créons l'image Docker en suivant les instructions fournies par l'équipe de développeurs :

1- CREATION DU FICHIER DOCKERFILE

Nous avons créé un fichier nommé Dockerfile dans le même répertoire que le script ping-exporter.py

A screenshot of a code editor with a dark theme. The top bar shows four tabs: 'Dockerfile' (active), 'Docker-compose.yml', 'prometheus.yml', and 'ping-exporter.py 3'. The 'Dockerfile' tab is selected, and the editor shows the following content:

```
1  
2 FROM alpine:3.9  
3  
4 RUN apk add --no-cache python2 py2-pip fping  
5  
6 COPY ping-exporter.py /opt/ping-exporter.py  
7  
8 CMD ["python2", "/opt/ping-exporter.py"]  
9
```

2- CONSTRUCTION DE L'IMAGE DOCKER

Dans notre terminal, nous avons exécuté la commande suivante pour construire l'image Docker :

docker build -t ping-exporter .

3- EXECUTION AVEC DOCKER COMPOSE

Nous avons créé un fichier Dockercompose.yml pour exécuter Prometheus et l'exportateur ping ensemble :

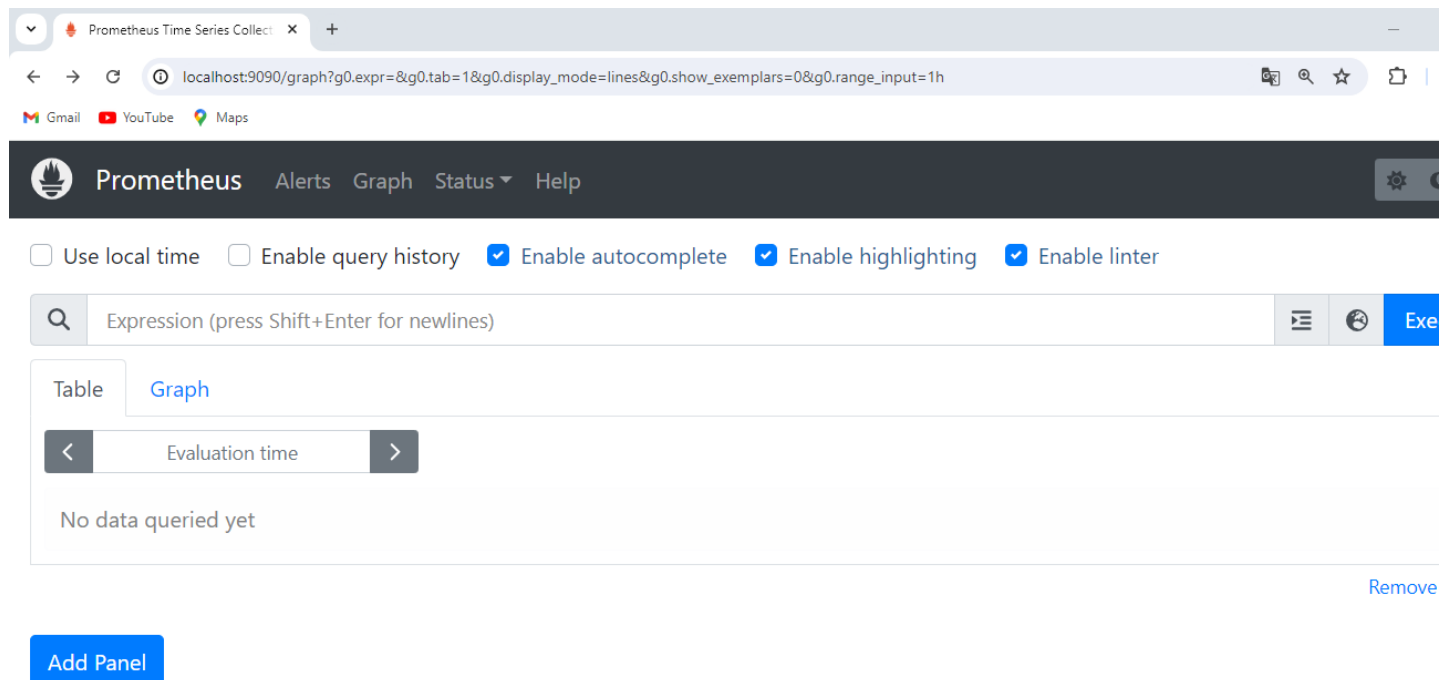
```
Docker-compose.yml X Dockerfile ! prometheus.yml ping-exporter.py 3
Docker-compose.yml > {} services > {} prometheus > [ ] ports
1  version: '3.8'
2
3  services:
4    ping-exporter:
5      build: .
6      environment:
7        - PORT=80
8      ports:
9        - "80:80"
10
11   prometheus:
12     image: prom/prometheus:latest
13     volumes:
14       - ./prometheus.yml:/etc/prometheus/prometheus.yml
15     ports:
16       - "9090:9090"
17
```

4- CONSTRUCTION

Nous avons utilisé la commande **docker-compose up --build** pour construire et démarrer les différents services définis dans notre fichier docker-compose.yml

```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\smart\Desktop\PROJET 3DCT> docker-compose up --build
[+] Building 1.7s (8/8) FINISHED
=> [ping-exporter internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 367B 0.0s
=> [ping-exporter internal] load metadata for docker.io/library/alpine:3.9 1.1s
=> [ping-exporter internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [ping-exporter 1/3] FROM docker.io/library/alpine:3.9@sha256:414e0518bb9228d35e4cd5165567fb91d26c6a214e9c9 0.0s
=> [ping-exporter internal] load build context 0.0s
=> => transferring context: 38B 0.0s
=> CACHED [ping-exporter 2/3] RUN apk add --no-cache python2 py2-pip fping 0.0s
=> CACHED [ping-exporter 3/3] COPY ping-exporter.py /opt/ping-exporter.py 0.0s
=> [ping-exporter] exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:df31aa782044498b4dfd6033b7097dc2ed4a7b78aa24c4e4f386a5491de80cce 0.0s
=> => naming to docker.io/library/projet3dct-ping-exporter 0.0s
[+] Running 3/3
  ✓ Network projet3dct_default Created 0.2s
  ✓ Container projet3dct-prometheus-1 Created 0.4s
  ✓ Container projet3dct-ping-exporter-1 Created 0.4s
Attaching to ping-exporter-1, prometheus-1
ping-exporter-1 | 2024-07-13 19:53:06,194 root INFO Starting server on port 80, use <Ctrl-C> to stop
prometheus-1 | ts=2024-07-13T19:53:06.508Z caller=main.go:589 level=info msg="No time or size retention was set s
```

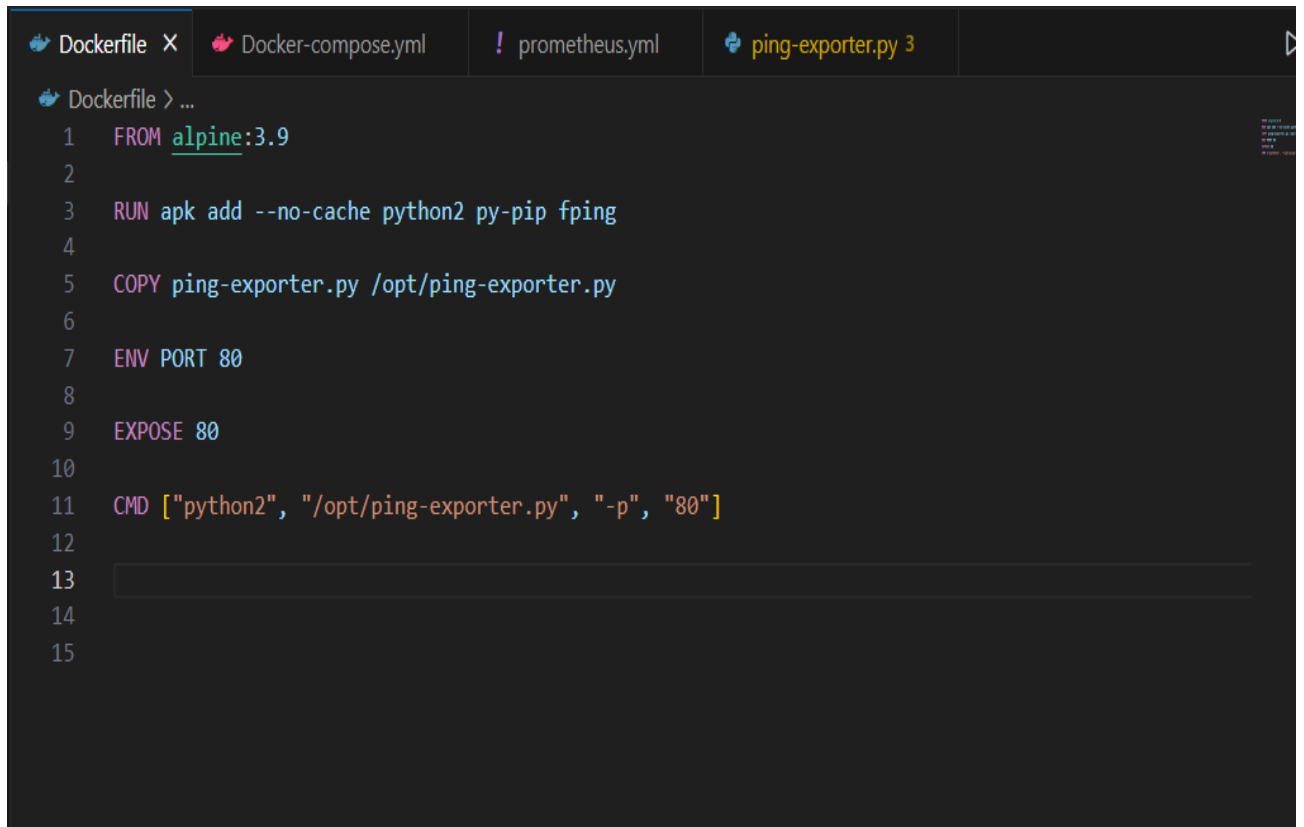
Comme le montre l'image précédente les services ont été créés et en accédant à l'adresse <http://127.0.0.1:9000/> nous pouvons voir l'interface de prometheus :



II- FAIRE DU PORT UNE VARIABLE D'ENVIRONNEMENT

1- MODIFICATION DU DOCKERFILE

Pour faire du port une variable d'environnement nous avons modifier notre docker file comme suit :

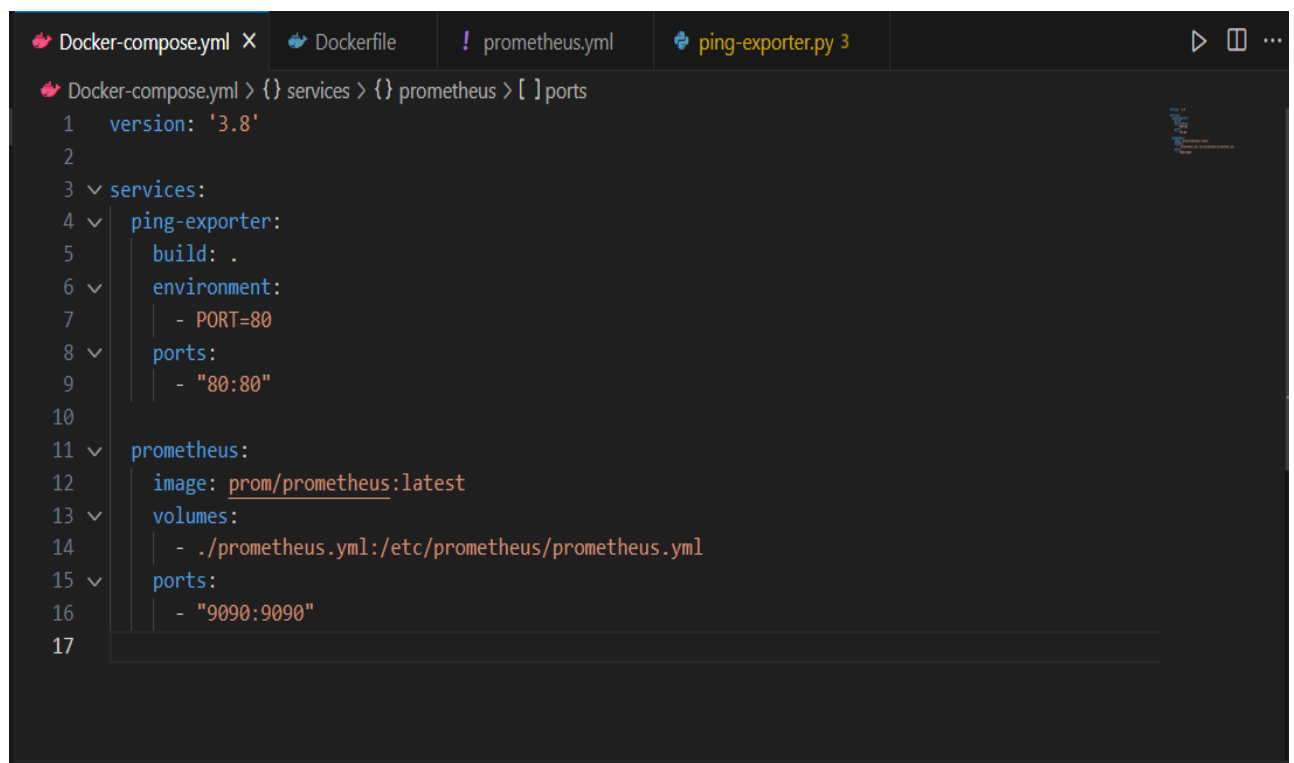


```
Dockerfile X Docker-compose.yml ! prometheus.yml ping-exporter.py 3
Dockerfile > ...
1 FROM alpine:3.9
2
3 RUN apk add --no-cache python2 py-pip fping
4
5 COPY ping-exporter.py /opt/ping-exporter.py
6
7 ENV PORT 80
8
9 EXPOSE 80
10
11 CMD ["python2", "/opt/ping-exporter.py", "-p", "80"]
12
13
14
15
```

Ici

- **ENV PORT 80** définit la variable d'environnement PORT avec la valeur 80
- **EXPOSE 80** Indique que le conteneur écoute sur le port 80
- **CMD ["python2", "/opt/ping-exporter.py", "-p", "80"]**
 - **python2** : Exécute l'interpréteur Python 2.
 - **/opt/ping-exporter.py** : Chemin vers le script Python à exécuter.
 - **-p 80** : Argument passé au script, spécifiant le port à utiliser (80).

Nous avons également modifié notre fichier Docker-compose.yml comme suit :

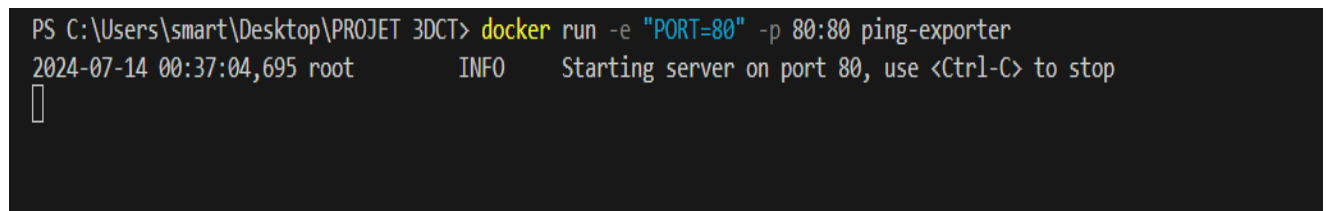


```
Docker-compose.yml X Dockerfile ! prometheus.yml ping-exporter.py 3
Docker-compose.yml > {} services > {} prometheus > [ ] ports
1 version: '3.8'
2
3 services:
4 ping-exporter:
5   build: .
6   environment:
7     - PORT=80
8   ports:
9     - "80:80"
10
11 prometheus:
12   image: prom/prometheus:latest
13   volumes:
14     - ./prometheus.yml:/etc/prometheus/prometheus.yml
15   ports:
16     - "9090:9090"
17
```

environment: - PORT=80 : Définit la variable d'environnement pour le conteneur. Ici, la variable d'environnement PORT est définie avec la valeur 80. Cela configure le conteneur pour qu'il utilise le port 80.

Ensuite nous avons utilisé la commande **docker-compose up** qui crée et démarre le conteneur pour le service exportateur, expose le port 8000 sur l'hôte et le mappe au port 80 du conteneur.

Et enfin nous avons exécuté la commande : **docker run -e "PORT=80" -p 80:80 ping-exporter** et nous avons eu comme résultat :



```
PS C:\Users\smart\Desktop\PROJET 3DCT> docker run -e "PORT=80" -p 80:80 ping-exporter
2024-07-14 00:37:04,695 root INFO Starting server on port 80, use <Ctrl-C> to stop
```

Ce qui montre que le serveur a démarré avec succès sur le port 80 à l'intérieur du conteneur

III- EXECUTEZ L'EXPORTATEUR DEPUIS DOCKERCOMPOSE

fichier dockercompose.yml qui exécutera notre image fraîchement construite avec les propriétés suivantes :

- Nom du service : exportateur
- Variables d'environnement : o PORT égal à 80.
- Ports : port 8000 de votre ordinateur vers le port 80 de votre conteneur.

```
Docker-compose.yml X Dockerfile ! prometheus.yml ping-exporter.py 3
🔥 Docker-compose.yml > {} services > {} exportateur > [ ] ports
1 version: '3.8'
2
3 services:
4 exportateur:
5 image: ping-exporter
6 environment:
7 - PORT=80
8 ports:
9 - "8000:80"
10
```

Par la suite nous avons exécuté la commande docker-compose up

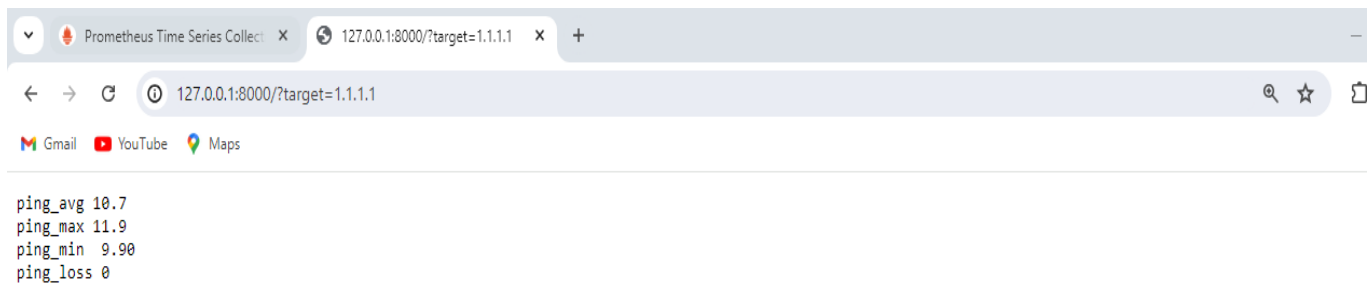
```
PS C:\Users\smart\Desktop\PROJET 3DCT> docker-compose up -d
time="2024-07-14T02:45:00+02:00" level=warning msg="Found orphan containers ([projet3dct-ping-exporter-1 projet3dct-p
rometheus-1]) for this project. If you removed or renamed this service in your compose file, you can run this command
with the --remove-orphans flag to clean it up."
[+] Running 1/1
✔ Container projet3dct-exportateur-1 Started 1.0s
PS C:\Users\smart\Desktop\PROJET 3DCT>
```

Ensuite nous avons vérifié si le conteneur est bien démarré

```
PS C:\Users\smart\Desktop\PROJET 3DCT> docker-compose ps
NAME                IMAGE             COMMAND                  SERVICE    CREATED         STATUS
projet3dct-exportateur-1 ping-exporter     "python2 /opt/ping-e..." exportateur About a minute ago Up About a min
ute 0.0.0.0:8000->80/tcp
PS C:\Users\smart\Desktop\PROJET 3DCT>
```

Nous constatons qu'il est bien démarré.

Pour nous assurer que cela fonctionne, accédons à <http://127.0.0.1:8000/?target=1.1.1.1>

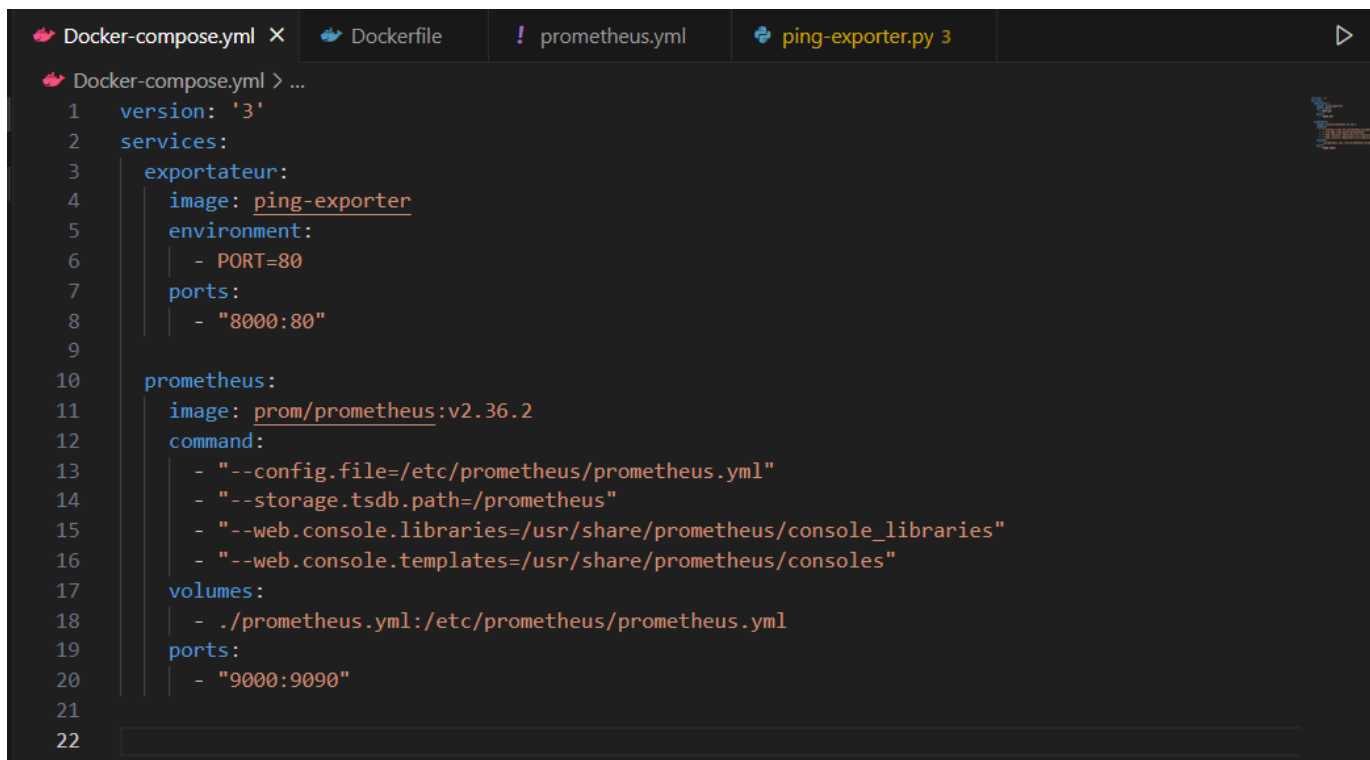


```
ping_avg 10.7
ping_max 11.9
ping_min 9.90
ping_loss 0
```

En saisissant cette adresse dans notre navigateur nous avons belle et bien obtenue les métriques de performance obtenues à partir de la commande du ping

IV- EXECUTION DE PROMETHEUS

1- Ajout des parties manquante du fichier docker-compose.yml



```
Docker-compose.yml > ...
1  version: '3'
2  services:
3    exportateur:
4      image: ping-exporter
5      environment:
6        - PORT=80
7      ports:
8        - "8000:80"
9
10   prometheus:
11     image: prom/prometheus:v2.36.2
12     command:
13       - "--config.file=/etc/prometheus/prometheus.yml"
14       - "--storage.tsdb.path=/prometheus"
15       - "--web.console.libraries=/usr/share/prometheus/console_libraries"
16       - "--web.console.templates=/usr/share/prometheus/consoles"
17     volumes:
18       - ./prometheus.yml:/etc/prometheus/prometheus.yml
19     ports:
20       - "9000:9090"
21
22
```

V- CONFIGURATION PROMETHEUS

```
! prometheus.yml X Docker-compose.yml Dockerfile ping-exporter.py 3
! prometheus.yml > ...
prometheus.json - Prometheus configuration file (prometheus.json)
1 # my global config
2 global:
3   scrape_interval: 15s # By default, scrape targets every 15 seconds.
4   evaluation_interval: 15s # By default, scrape targets every 15 seconds.
5   # scrape_timeout is set to the global default (10s).
6
7   # Attach these labels to any time series or alerts when communicating with
8   # external systems (federation, remote storage, Alertmanager).
9   external_labels:
10     monitor: "my-project"
11
12 # Load and evaluate rules in this file every 'evaluation_interval' seconds.
13 rule_files:
14   - "alert.rules"
15   # - "first.rules"
16   # - "second.rules"
17
18 # alert
19 alerting:
20   alertmanagers:
21     - scheme: http
22       static_configs:
23         - targets:
24           - "alertmanager:9093"
25
```

```
! prometheus.yml X Docker-compose.yml Dockerfile ping-exporter.py 3
! prometheus.yml > {} global
19 alerting:
20   alertmanagers:
21     - scheme: http
22       static_configs:
23         - targets:
24           - "alertmanager:9093"
25
26 # A scrape configuration containing exactly one endpoint to scrape:
27 # Here it's Prometheus itself.
28 scrape_configs:
29   # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
30
31   - job_name: "ping-exporter"
32
33     # Override the global default and scrape targets from this job every 5 seconds.
34     scrape_interval: 15s
35     metrics_path: /
36     params:
37       target: ['8.8.8.8']
38
39     static_configs:
40       - targets: ["exportateur:80"]
41
42
```

ADRESSE DU CONTENEUR : PORT DU CONTENEUR

A été remplacé par exportateur :80

Par la suite en accédant à <http://127.0.0.1:9000> nous avons obtenu le graphe suivant :

