Comment installer Docker Swarm sur Ubuntu 22.04

Docker Swarm est une orchestration de conteneurs construite sur Docker Engine. Il vous permet de créer et de déployer un cluster de nœuds Docker sur plusieurs serveurs. Docker Swarm simplifie le déploiement de votre application conteneurisée en service. Il fournit un moyen simple et simple de gérer et d'orchestrer les conteneurs.

Docker Swarm offre un haut niveau d'applications disponibles. Dans Docker Swarm, vous pouvez exécuter une seule application ou un seul service sur plusieurs nœuds, appelés « nœuds de travail ». Et il existe également un nœud appelé « Swarm Manager », qui est la gestion et l'orchestration centrales de Docker Swarm.

Dans ce didacticiel, nous allons vous montrer étape par étape comment installer le logiciel Docker Swarm sur les serveurs Ubuntu 22.04.

Conditions préalables

Pour compléter ce guide, assurez-vous d'être équipé des éléments suivants :

- 3 serveurs Ubuntu 22.04 - Un sera utilisé comme Swarm Master/Manager et deux serveurs seront utilisés comme nœuds de travail.
- Un utilisateur non root avec les privilèges d'administrateur sudo.

Configuration des systèmes

Avant d'installer Docker et de configurer Docker Swarm, vous devez préparer vos systèmes en effectuant les tâches suivantes :

- Ports ouverts pour Docker Swarm : vous devez ouvrir certains ports utilisés par Docker Swarm sur tous vos serveurs. Ceci peut être réalisé via UFW (Uncomplicated Firewall).
- Ajout du registre Docker : vous utiliserez le registre Docker officiel pour installer Docker Engine sur tous vos serveurs.

Ports ouverts pour Docker Swarm

Dans la section suivante, vous ouvrirez le port **22** pour SSH, puis les ports **2377**,**7946**, et **4789** pour Docker Swarm via UFW (Uncomplicated Firewall) sur le Swarm Master/Manager et les Swarm Nodes. L'UFW est installé par défaut, mais pas encore démarré.

Tout d'abord, exécutez la commande ufw ci-dessous pour ajouter le profil d'application OpenSSH et ouvrir le port SSH 22 par défaut. Ensuite, démarrez et activez UFW.

```
sudo ufw allow OpenSSH
sudo ufw enable
```

Tapez y pour continuer et vous devriez obtenir le résultat **Le pare-feu est actif et activé au démarrage du système**.

Exécutez maintenant la commande ci-dessous pour ouvrir les ports qui seront utilisés par les services de votre déploiement Swarm. Dans ce cas, vous allouerez des ports entre 30 000:35 000 pour les services.

```
sudo ufw allow 30000:35000/tcp
```

Ensuite, exécutez la commande suivante pour ouvrir les ports pour Docker Swarm.

```
for ports in 2377/tcp 7946/tcp 7946/udp 4789/udp
do
sudo ufw allow $ports
done
```

```
root@manager:~#
root@manager:~# sudo ufw allow 30000:35000/tcp
Rule added
Rule added (v6)
root@manager:~#
root@manager:~# for ports in 2377/tcp 7946/tcp 7946/udp 4789/udp
do
sudo ufw allow $ports
done
Rule added
Rule added (v6)
Rule added
Rule added (v6)
Rule added
Rule added (v6)
Rule added
Rule added (v6)
root@manager:~#
root@manager:~# sudo ufw reload
Firewall reloaded
root@manager:~#
```

Enfin, rechargez UFW et vérifiez l'état d'UFW en exécutant la commande ci-dessous.

```
sudo ufw reload
sudo ufw status
```

Vous devriez voir le profil d'application OpenSSH et les ports pour Docker Swarm, notamment 2377, 7946 et 4789, activés sur UFW.

```
root@manager:~#
root@manager:~# sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
30000:35000/tcp            ALLOW       Anywhere
2377/tcp                   ALLOW       Anywhere
7946/tcp                   ALLOW       Anywhere
7946/udp                   ALLOW       Anywhere
4789/udp                   ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
30000:35000/tcp (v6)       ALLOW       Anywhere (v6)
2377/tcp (v6)              ALLOW       Anywhere (v6)
7946/tcp (v6)              ALLOW       Anywhere (v6)
7946/udp (v6)              ALLOW       Anywhere (v6)
4789/udp (v6)              ALLOW       Anywhere (v6)
```

Ajout du registre Docker

Après avoir configuré l'UFW, vous devez ajouter le registre Docker officiel à vos serveurs. Vous utiliserez le registre Docker officiel pour installer Docker Engine.

Exécutez la commande ci-dessous pour installer certains packages de base sur vos serveurs Ubuntu.

```
sudo apt install apt-transport-https ca-certificates curl gnupg lsb-release -y
```

```
root@manager:~#
root@manager:~# sudo apt install apt-transport-https ca-certificates curl gnupg lsb-release -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
lsb-release is already the newest version (11.1.0ubuntu4).
lsb-release set to manually installed.
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.14).
curl set to manually installed.
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 14 not upgraded.
Need to get 1510 B of archives.
After this operation, 169 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-transport-https all 2.4.10 [1510 B]
```

Ensuite, exécutez la commande ci-dessous pour ajouter la clé et le *registre* Docker GPG à vos systèmes.

```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

```
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
root@manager:~#
root@manager:~# sudo install -m 0755 -d /etc/apt/keyrings
root@manager:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
root@manager:~# sudo chmod a+r /etc/apt/keyrings/docker.gpg
root@manager:~#
root@manager:~# echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
root@manager:~#
root@manager:~#
```

Enfin, mettez à jour et actualisez votre registre Ubuntu sur chaque serveur en exécutant la commande ci-dessous.

```
sudo apt update
```

Vous devriez voir le registre Docker récupéré pendant le processus.

```
root@manager:~#
root@manager:~# sudo apt update
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Installation du moteur Docker

Après avoir préparé vos systèmes Ubuntu, vous installerez Docker Engine sur ces serveurs.

Installez Docker Engine sur vos systèmes Ubuntu à l'aide de la commande ci-dessous. Saisissez y pour confirmer l'installation.

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

```
root@manager:~#
root@manager:~# sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7
  slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 14 not upgraded.
Need to get 114 MB of archives.
After this operation, 409 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Une fois l'installation terminée, exécutez la commande systemctl ci-dessous pour vérifier le service Docker et vous assurer que le service est en cours d'exécution.

```
sudo systemctl is-enabled docker
sudo systemctl status docker
```

Dans le résultat suivant, vous devriez voir que le service Docker est en cours d'exécution et activé.

```
root@manager:~#
root@manager:~# sudo systemctl is-enabled docker
enabled
root@manager:~# sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 2213 (dockerd)
      Tasks: 10
     Memory: 26.5M
        CPU: 1.420s
```

(Facultatif) : Autoriser les non-roots à exécuter les conteneurs Docker

Si vous déployez Docker Swarm et exécutez des conteneurs à l'aide d'un utilisateur non root, vous devez ajouter votre utilisateur au groupe Docker afin que l'utilisateur puisse exécuter la commande Docker et exécuter des conteneurs.

Exécutez la commande usermod ci-dessous pour ajouter votre utilisateur actuel au groupe Docker.

```
sudo usermod -aG docker username
```

Connectez-vous maintenant en tant qu'utilisateur non root et exécutez la commande docker ci-dessous pour vérifier votre configuration.

```
su - username
docker run hello-world
```

Si la configuration réussit, vous devriez pouvoir exécuter le conteneur hello-world et obtenir le résultat suivant :

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Création d'un essaim Docker

Maintenant que vous avez installé Docker Engine, vous êtes prêt à créer et démarrer Docker à l'aide de vos serveurs Ubuntu. Dans cet exemple, nous utiliserons un serveur comme Swarm Master/Manager et deux serveurs comme Worker Nodes.

Effectuez les tâches suivantes pour configurer Docker Swarm :

- Initialisation du mode Swarm sur le Master/Manager.
- Ajout de nœuds de travail au Docker Swarm.

Commençons.

Initialisation du mode Swarm sur le maître/gestionnaire

Pour initialiser Docker Swarm, exécutez la commande docker swarm init ci-dessous. Le paramètre supplémentaire --advertise-addr liera le Docker Swarm à l'adresse IP spécifique, et le paramètre --default-addr-pool détermine l'adresse IP interne des conteneurs exécutés sur le Swarm.

Dans cet exemple, le mode Docker Swarm se liera à l'adresse IP**192.168.5.30** et le pool d'adresses IP pour les conteneurs est **10.20.0.0/16**.

```
docker swarm init --advertise-addr 192.168.5.30 --default-addr-pool 10.20.0.0/16
```

Si le processus d'initialisation réussit, la sortie suivante s'affichera. Dans la sortie, vous devriez voir le jeton généré pour l'ajout de nœuds à Docker Swarm.

```
root@manager:~#
root@manager:~# docker swarm init --advertise-addr 192.168.5.30 --default-addr-pool 10.20.0.0/16
Swarm initialized: current node (zbjeko5o32kwfrg179m77gf6c) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-0i6kbe2oek1iw19jfpvd2j5l0dhfmssz4w505aeihx7ouz8wqc-2dbk7cnmo12uunj53eywwnqr7 192.168.5.30:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

root@manager:~#
```

Ensuite, exécutez la commande suivante pour vérifier l'état du mode Swarm.

```
docker info
```

Si le mode Swarm est activé, vous devriez obtenir la sortie Swarm: actif avec les détails du NodeID et l'état du Manager et du Node.

```
Swarm: active
 NodeID: zbjeko5o32kwfrg179m77gf6c
 Is Manager: true
 ClusterID: kc796s677x31hen0apqxq55yu
 Managers: 1
 Nodes: 1
 Default Address Pool: 10.20.0.0/16
 SubnetSize: 24
 Data Path Port: 4789
 Orchestration:
  Task History Retention Limit: 5
 Raft:
```

Enfin, exécutez la commande ci-dessous pour vérifier la liste des nœuds sur Docker Swarm.

```
docker node ls
```

À ce stade, un seul nœud est disponible sur votre Docker Swarm, qui est le Swarm Master/Manager.

```
root@manager:~#
root@manager:~# docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
zbjeko5o32kwfrg179m77gf6c *   manager    Ready     Active         Leader           24.0.7
root@manager:~#
root@manager:~#
```

Ajout de nœuds de travail à Docker Swarm

Une fois le Docker Swarm initialisé, vous pouvez ajouter des nœuds de travail à votre Docker Swarm.

Tout d'abord, exécutez la commande ci-dessous pour afficher le jeton généré pour le nœud de travail.

```
docker swarm join-token worker
```

Vous devriez voir les instructions pour ajouter des nœuds de travail, qui incluent le jeton.

```
root@manager:~#
root@manager:~# docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-0i6kbe2oek1iw19jfpvd2j5l0dhfmssz4w505aeihx7ouz8wqc-2dbk7cnmo12uunj53eywwnqr7 192.168.5.30:2377

root@manager:~#
```

Maintenant, passez au nœud de travail et ajoutez le nœud de travail au Docker Swarm en exécutant la commande docker swarm join ci-dessous.

```
docker swarm join --token SWMTKN-1-0i6kbe2oek1iw19jfpvd2j5l0dhfmssz4w505aeihx7ouz8wqc-2dbk7cnmo12uunj53eywwnqr7 192.168.5.30:2377
```

Lorsque le processus réussit, le résultat « Ce nœud a rejoint un essaim en tant que travailleur » sera imprimé sur votre écran.

```
root@worker1:~#
root@worker1:~# docker swarm join --token SWMTKN-1-0i6kbe2oek1iw19jfpvd2j5l0dhfmssz4w505aeihx7ouz8wqc-2dbk7cnmo12uunj53eywwnqr7 192.168.5.30:2377
This node joined a swarm as a worker.
root@worker1:~#
```

```
root@worker2:~#
root@worker2:~# docker swarm join --token SWMTKN-1-0i6kbe2oek1iw19jfpvd2j5l0dhfmssz4w505aeihx7ouz8wqc-2dbk7cnmo12uunj53eywwnqr7 192.168.5.30:2377
This node joined a swarm as a worker.
root@worker2:~#
root@worker2:~#
```

Ensuite, revenez au Swarm Master/Manager et exécutez la commande ci-dessous pour vérifier la liste des nœuds disponibles.

```
docker node ls
```

Si tout se passe bien, il y aura trois serveurs disponibles sur le Docker Swarm, 1 sur le Swarm Manager et 2 nœuds de travail avec le statut Prêt et la disponibilité est en cours d'exécution.

```
root@manager:~# docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
zbjeko5o32kwfrg179m77gf6c *   manager    Ready     Active         Leader           24.0.7
vgxk4zi9ax1fy6zuduv7ttx5e     worker1    Ready     Active                          24.0.7
mcomf26ep5emaaykgrndi7fiq     worker2    Ready     Active                          24.0.7
root@manager:~#
root@manager:~#
```

Exécution du service dans Docker Swarm

À ce stade, vous avez créé avec succès Docker Swarm avec trois serveurs Ubuntu. Vous allez maintenant apprendre à déployer votre application dans Docker Swarm, appelé service. Un service est une image de votre application de microservice et il peut s'agir d'un serveur HTTP, d'un serveur de base de données ou d'autres applications.

Dans cet exemple, vous déployerez un service HTTP simple avec une image Nginx.

Exécutez la commande ci-dessous pour créer un nouveau service sur votre Swarm. Dans cet exemple, vous allez créer un nouveau service testnginx avec 1 réplica, exposer le port 30001 et baser l'image nginx:alpine.

```
docker service create --replicas 1 --name test-nginx -p 30001:80 nginx:alpine
```

```
root@manager:~#
root@manager:~# docker service create --replicas 1 --name test-nginx -p 30001:80 nginx:alpine
ok540djumirwxu0tz5n90tf9c
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
root@manager:~#
```

Maintenant, vérifiez les détails du service test-nginx à l'aide de la commande ci-dessous.

```
docker service inspect test-nginx
docker service inspect --pretty test-nginx
```

Vous devriez voir des informations détaillées sur le service test-nginx comme celles-ci.

```
root@manager:~#
root@manager:~# docker service inspect --pretty test-nginx

ID:             ok540djumirwxu0tz5n90tf9c
Name:           test-nginx
Service Mode:   Replicated
 Replicas:      1
Placement:
UpdateConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Update order:      stop-first
RollbackConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Rollback order:    stop-first
ContainerSpec:
 Image:         nginx:alpine@sha256:db353d0f0c479c91bd15e01fc68ed0
 Init:          false
Resources:
Endpoint Mode:  vip
Ports:
 PublishedPort = 30001
  Protocol = tcp
  TargetPort = 80
  PublishMode = ingress
```

Ensuite, vérifiez la liste des services Docker dans Docker Swarm à l'aide de la commande ci-dessous.

```
docker service ls
docker service ps test-nginx
```

En cas de succès, vous devriez voir le service test-nginx s'exécuter sur le gestionnaire NODE avec 1 réplique et le port exposé 30001.

```
root@manager:~#
root@manager:~# docker service ls
ID             NAME         MODE         REPLICAS   IMAGE          PORTS
ok540djumirw   test-nginx   replicated   1/1        nginx:alpine   *:30001->80/tcp
root@manager:~#
root@manager:~# docker service ps test-nginx
ID             NAME           IMAGE          NODE      DESIRED STATE   CURRENT STATE           ERROR     PORTS
ux1xer1drx7v   test-nginx.1   nginx:alpine   manager   Running         Running 2 minutes ago
root@manager:~#
root@manager:~#
```

Enfin, accédez au service test-nginx via l'adresse IP de l'hôte avec le port 30001 à l'aide de la commande curl ci-dessous.

```
curl 192.168.5.30:30001
curl -I 192.168.5.30:30001
```

Vous devriez voir la page de code source index.html et les détails des en-têtes HTTP.

```
root@manager:~#
root@manager:~# curl 192.168.5.30:30001
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Service de mise à l'échelle dans Docker Swarm

Maintenant que vous avez déployé le service Docker, la tâche suivante consiste à faire évoluer le service sur Docker Swarm. Cela créera la réplication souhaitée de vos services sur les nœuds de travail.

Pour faire évoluer un service, exécutez la commande docker service scale ci-dessous. Dans ce cas, vous mettrez à l'échelle le service test-nginx sur 3 réplicas.

```
docker service scale test-nginx=3
```

```
root@manager:~#
root@manager:~# docker service scale test-nginx=3
test-nginx scaled to 3
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
root@manager:~#
root@manager:~#
```

Exécutez maintenant la commande ci-dessous pour vérifier le service test-nginx. Si le service test-nginx a été mis à l'échelle, vous devriez voir deux autres services créés par Docker et exécutés sur les deux nœuds de travail.

```
docker service ps test-nginx
```

```
root@manager:~#
root@manager:~# docker service ps test-nginx
ID             NAME           IMAGE          NODE      DESIRED STATE   CURRENT STATE                ERROR     PORTS
ux1xer1drx7v   test-nginx.1   nginx:alpine   manager   Running         Running 8 minutes ago
atie45g6gira   test-nginx.2   nginx:alpine   worker2   Running         Running 2 minutes ago
nkycwa15dcnj   test-nginx.3   nginx:alpine   worker1   Running         Running about a minute ago
root@manager:~#
```

Accédez au terminal du nœud de travail et exécutez la commande suivante pour vous assurer que le service test-nginx est en cours d'exécution.

```
docker ps
curl 192.168.5.31:30001
```

Si tout se passe bien, vous devriez voir le conteneur test-nginx.RANDOM-STRING avec le statut Up sur chacun des nœuds de travail.

```
root@worker1:~#
root@worker1:~# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
ffed9923a2c0   nginx:alpine   "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes   80/tcp    test-nginx.3.nkycwa15dcnjey5ge31xps5mn
root@worker1:~#
root@worker1:~# curl 192.168.5.31:30001
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@worker1:~#
```

```
root@worker2:~#
root@worker2:~# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
51ad4e1f40e0   nginx:alpine   "/docker-entrypoint.…"   3 minutes ago   Up 3 minutes   80/tcp    test-nginx.2.atie45g6giraaabenvuntlody
root@worker2:~#
root@worker2:~# curl 192.168.5.32:30001
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@worker2:~#
```

Suppression d'un service dans Docker Swarm

Pour nettoyer votre environnement, vous supprimerez le service test-nginx de Docker Swarm et supprimerez l'image docker nginx:alpine.

Supprimez le service test-nginx et vérifiez la liste des services disponibles sur Docker Swarm à l'aide de la commande suivante.

```
docker service rm test-nginx
docker service ps
```

Supprimez maintenant l'image nginx:alpine et vérifiez la liste des images téléchargées pour chaque serveur à l'aide de la commande ci-dessous.

```
docker rmi nginx:alpine
docker images
```

Conclusion

Pour conclure, vous avez maintenant installé avec succès Docker Swarm sur Ubuntu 22.04, étape par étape. Vous avez déployé Docker Swarm avec trois serveurs Ubuntu et appris à déployer, mettre à l'échelle et supprimer des applications ou des services sur Docker Swarm.