

Documentation Installation de Docker Swarm

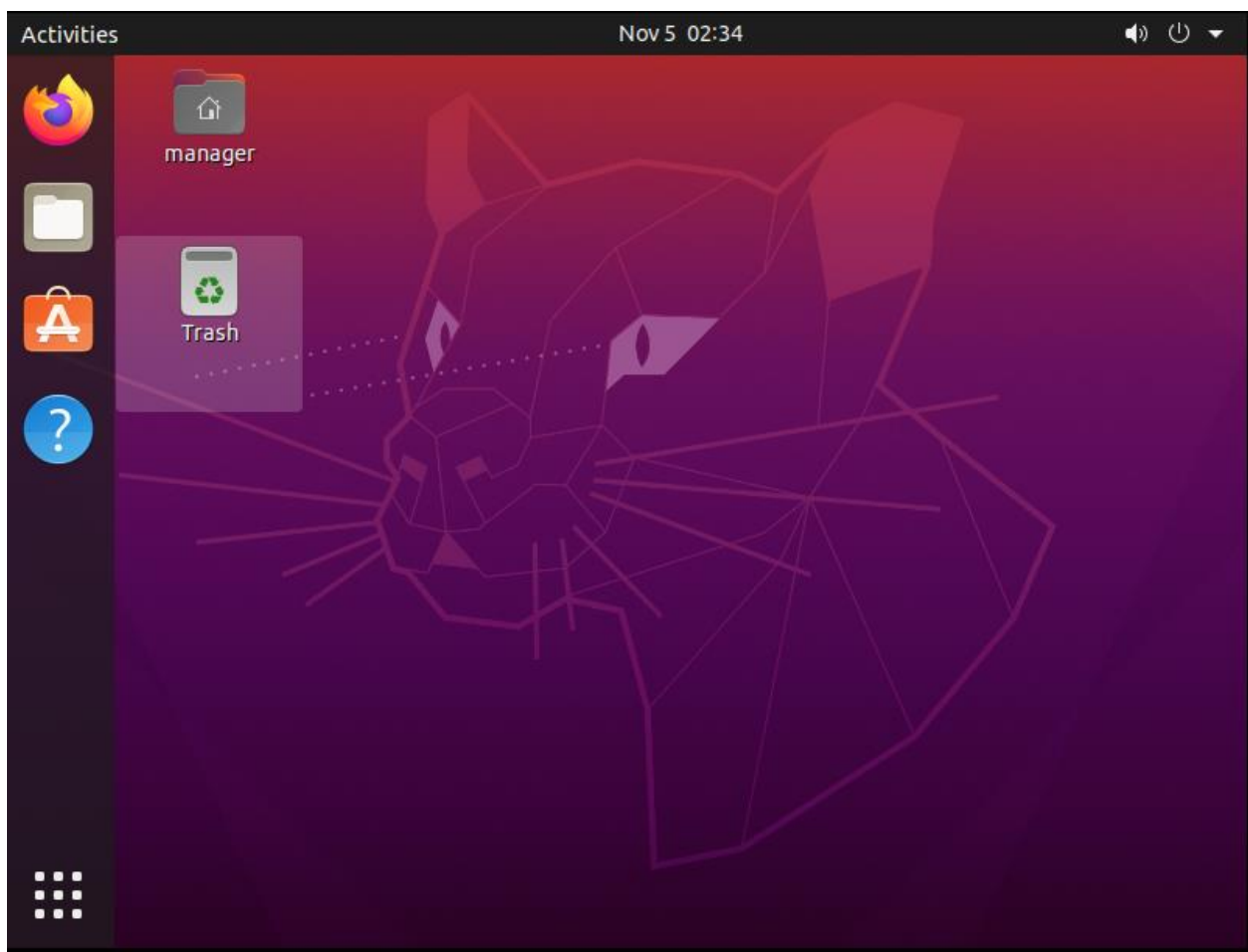
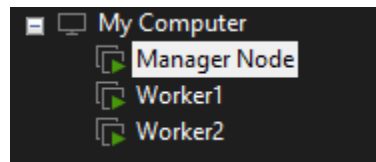
3DOKR

Yoann Chhang

Hugues-Pacôme Stock

Préparation :

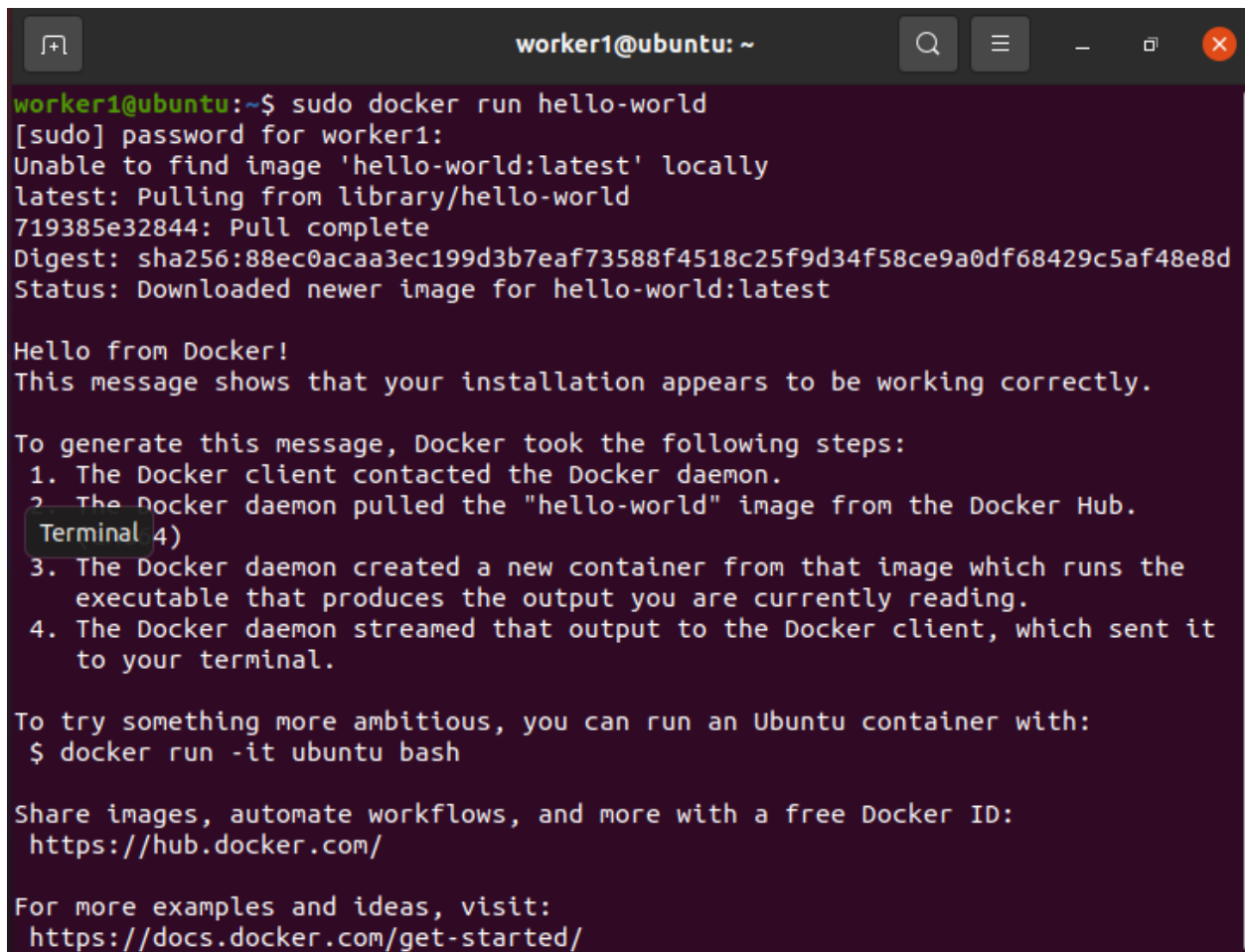
Afin de pouvoir installer un cluster Swarm, il faut avoir multiple machines disponible. Pour cela nous avons donc initialisé trois différentes machine Ubuntu sur VMware.



Chaque machine doit avoir accès à internet et doivent aussi pouvoir communiquer entre elles pour pouvoir faire partie du même cluster. Comme vous pouvez le voir, il y a une machine « manager » et deux « worker ».

Le cluster :

Afin de créer le cluster, il faut au préalable que chaque machine ait Docker installé. Les instructions d'installation se trouvent sur le site officiel de Docker. Après l'installation, nous pouvons donc vérifier que Docker tourne correctement sur nos machines.

A terminal window titled 'worker1@ubuntu: ~' with standard Ubuntu window controls. The terminal shows the command 'sudo docker run hello-world' being executed. It displays the process of pulling the 'hello-world:latest' image from Docker Hub, showing the digest and status. The output of the container is a 'Hello from Docker!' message followed by a confirmation that the installation is working. A list of four steps explains the process: 1. Docker client contacts the daemon, 2. daemon pulls the image from Docker Hub, 3. daemon creates a container from the image, and 4. daemon streams output to the client. It also provides instructions to run an Ubuntu container and links to Docker's documentation and image sharing resources.

```
worker1@ubuntu:~$ sudo docker run hello-world
[sudo] password for worker1:
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:88ec0acaa3ec199d3b7eaf73588f4518c25f9d34f58ce9a0df68429c5af48e8d
Status: Downloaded newer image for hello-world:latest

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.
    Terminal 4)
 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/
```

Ensuite, il faut donc bien sur initialiser le cluster avec notre node manager avec l'adresse IP correspondante. Après l'initialisation, nous pouvons ajouter les autres machines au cluster en tant que nodes workers.

```
manager@ubuntu: ~/Downloads/3DOCKER-main
manager@ubuntu:~/Downloads/3DOCKER-main$ sudo docker swarm init --advertise-addr 192.168.47.128
[sudo] password for manager:
Swarm initialized: current node (d40x7xtg7w30g38dqqqh3f1p) is now a manager.

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

    docker swarm join --token SWMTKN-1-0q7iyuhf23rn2xn1emahqmbi8bky8gjdk4he9hkp
h80urbm20o-aotf1fxd0um0df44t60garunr 192.168.47.128:2377

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

manager@ubuntu:~/Downloads/3DOCKER-main$

worker1@ubuntu:~$ sudo docker swarm join --token SWMTKN-1-0q7iyuhf23rn2xn1emahqmbi8bky8gjdk4he9hkp
h80urbm20o-aotf1fxd0um0df44t60garunr 192.168.47.128:2377
\This node joined a swarm as a worker.
worker1@ubuntu:~$ \

worker2@ubuntu:~$ sudo docker swarm join --token SWMTKN-1-0q7iyuhf23rn2xn1emahqmbi8bky8gjdk4he9hkp
h80urbm20o-aotf1fxd0um0df44t60garunr 192.168.47.128:2377
This node joined a swarm as a worker.
worker2@ubuntu:~$
```

L'application :

Entre temps, j'ai pu importer sur notre node manager le dossier contenant les documents pour les différentes applications et les fichiers Docker qui vont avec (Dockerfile, Docker compose).

La première étape pour déployer des applications sur un cluster est de préparer les images de ces dites applications. Nous devons donc d'abord « build » chaque application une par une pour avoir une image utilisable lors du déploiement.


```

manager@ubuntu:~/Downloads/3DOCKER-main$
manager@ubuntu:~/Downloads/3DOCKER-main$
manager@ubuntu:~/Downloads/3DOCKER-main$ sudo docker build -t vote-app:latest .
/vote/
[+] Building 33.9s (9/9) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 558B                               0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                      0.0s
=> [internal] load metadata for docker.io/library/python:3.11     2.2s
=> [1/4] FROM docker.io/library/python:3.11@sha256:9e00960bde4d9aafdcb 27.1s
=> => resolve docker.io/library/python:3.11@sha256:9e00960bde4d9aafdcbf 0.0s
=> => sha256:99cb81c1d8e4d6fe275c1f5127c770ad86a6428653 2.01kB / 2.01kB 0.0s
=> => sha256:13baa2029dde87a21b87127168a0fb50a007c07d 24.05MB / 24.05MB 2.9s
=> => sha256:9e00960bde4d9aafdcbf2f0fc5b31b15e1824fc795 2.14kB / 2.14kB 0.0s
=> => sha256:8457fd5474e70835e4482983a5662355d892d5f6 49.58MB / 49.58MB 4.9s
=> => sha256:325c5bf4c2f26c11380501bec4b6eef8a3ea35b5 64.13MB / 64.13MB 6.1s
=> => sha256:3f7984adbac453e46f5063ac7193e461419ad446f2 7.53kB / 7.53kB 0.0s
=> => sha256:7e18a660069fd7f87a7a6c49ddb701449bfb9 211.06MB / 211.06MB 14.4s
=> => sha256:98a59f0ffedebe05b4e1ada824cb5389ff0552e77d 6.39MB / 6.39MB 5.9s
=> => extracting sha256:8457fd5474e70835e4482983a5662355d892d5f6f0f90a2 4.4s
=> => sha256:3a5444633a3348f11441df77b3fe52d2315b6ba0 19.79MB / 19.79MB 7.5s
=> => sha256:bbbc9b405dabdd464f2648e63d47bcb6233868b806ef20 244B / 244B 6.6s
=> => sha256:d9992232ef9b080bfff5a001d720504d9fab61a70d5 3.11MB / 3.11MB 7.4s

```

```

manager@ubuntu:~/Downloads/3DOCKER-main$
manager@ubuntu:~/Downloads/3DOCKER-main$ sudo docker build -t result-app:latest
./result/
[+] Building 11.5s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 454B                               0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                      0.0s
=> [internal] load metadata for docker.io/library/node:18         2.5s
=> [internal] load build context                                  0.0s
=> => transferring context: 302.89kB                               0.0s
=> [1/5] FROM docker.io/library/node:18@sha256:7ce8b205d15e30fd395e5fa4 5.2s
=> => resolve docker.io/library/node:18@sha256:7ce8b205d15e30fd395e5fa4 0.0s
=> => sha256:740b4cc8f44c445ced04a02f35262590e02d4d3a 45.89MB / 45.89MB 2.1s
=> => sha256:117e3f0204875685962a88ee77274eb954801b660e 2.00kB / 2.00kB 0.0s
=> => sha256:e1b0a66a57dc941a3d03d3bc859652fbc0084e553e 7.53kB / 7.53kB 0.0s
=> => sha256:c30e0accec6d561f6ec4e8d3ccebdeedbe505923a61 3.37kB / 3.37kB 0.3s
=> => sha256:a3c8fe9d0a3c2ae48aebc6a7034112fd06bd51bef0 2.21MB / 2.21MB 0.5s
=> => sha256:7ce8b205d15e30fd395e5fa4000bcdff595fcff3f43 1.21kB / 1.21kB 0.0s
=> => extracting sha256:c30e0accec6d561f6ec4e8d3ccebdeedbe505923a613afe3 0.0s
=> => sha256:d61a850b2d03a5ba8332fe6a1479476de561a339196b21 450B / 450B 0.7s

```

```

manager@ubuntu:~/Downloads/3DOCKER-main$ sudo docker build -t worker-app:latest
./worker/
[+] Building 9.9s (8/15)                                           docker:default
=> => extracting sha256:0bc8ff246cb8ff91066742f8f7ded40397e7aaaa925200b 6.8s
=> => sha256:3a15a3647c58c15fc183536f7bfab56ea6c76bd3ed6db6 154B / 154B 2.5s
=> => sha256:20fa5f436d27e8ec2798a3781cc08278e1519bb6 10.12MB / 10.12MB 3.1s
=> => sha256:7f70ff820d70543a3a58f64a743dfb15d1e60d84 25.38MB / 25.38MB 4.4s
=> => sha256:59fab9224636a0b076b5ad6d93a6d2fbf358d70 78.64MB / 180.96MB 9.1s
=> => sha256:7a14ac1ce672a28ee49d8e19e037207891fc737f 13.98MB / 13.98MB 5.3s
=> => extracting sha256:5c7a0ff61f6b8430f0383462fbd6e2b1a45b1cd917c0312 0.6s
=> => extracting sha256:711eaa7c8f3b33bee5f2f315cbe0147a4e65c0e3f67e6f9 1.2s
=> => extracting sha256:3a15a3647c58c15fc183536f7bfab56ea6c76bd3ed6db63 0.0s
=> => extracting sha256:20fa5f436d27e8ec2798a3781cc08278e1519bb682a60a5 0.4s
=> => extracting sha256:7f70ff820d70543a3a58f64a743dfb15d1e60d84f7a8399 1.5s
=> [base 1/2] FROM mcr.microsoft.com/dotnet/runtime:7.0@sha256:709d5795 6.6s
=> => resolve mcr.microsoft.com/dotnet/runtime:7.0@sha256:709d5795be92d 0.0s
=> => sha256:709d5795be92db33931ef7485aa9b07ad585473422 1.79kB / 1.79kB 0.0s
=> => sha256:fc024dc1213e67cc9f87965af8249cd75acf9380bd 1.16kB / 1.16kB 0.0s
=> => sha256:e8698d209f83b13c33e2d449904631efd3bfeb73c9 1.93kB / 1.93kB 0.0s
=> => sha256:0bc8ff246cb8ff91066742f8f7ded40397e7aaaa 31.42MB / 31.42MB 2.3s
=> => sha256:5c7a0ff61f6b8430f0383462fbd6e2b1a45b1cd9 14.97MB / 14.97MB 3.4s
=> => sha256:711eaa7c8f3b33bee5f2f315cbe0147a4e65c0e3 32.46MB / 32.46MB 4.1s
=> => sha256:3a15a3647c58c15fc183536f7bfab56ea6c76bd3ed6db6 154B / 154B 2.5s
=> => extracting sha256:0bc8ff246cb8ff91066742f8f7ded40397e7aaaa925200b 2.3s

```

Nous avons maintenant des builds utilisables par la machine maître. Malheureusement, nos machines « worker » n'y ont pas encore accès. Pour rendre les images accessibles, nous devons les envoyer dans un ou plusieurs registres. Nous utilisons donc DockerHub en tant que registre pour rendre nos images accessibles.

yoannchh / 3dokr-result Contains: Image Last pushed: 14 hours ago	Inactive	☆ 0	7	Public
yoannchh / 3dokr-worker Contains: Image Last pushed: 14 hours ago	Inactive	☆ 0	7	Public
yoannchh / 3dokr-vote Contains: Image Last pushed: 14 hours ago	Inactive	☆ 0	7	Public

```
manager@ubuntu:~$ sudo docker login -u yoannchh
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
```

```
manager@ubuntu:~$ sudo docker tag vote-app:latest yoannchh/3dokr-vote:latest
manager@ubuntu:~$ sudo docker tag result-app:latest yoannchh/3dokr-result:latest
manager@ubuntu:~$ sudo docker tag worker-app:latest yoannchh/3dokr-worker:latest
```

```
manager@ubuntu:~$ sudo docker push yoannchh/3dokr-vote
Using default tag: latest
The push refers to repository [docker.io/yoannchh/3dokr-vote]
871ad127157b: Pushing 13.6MB/17.73MB
4697ff0d0b96: Pushed
f5c364566306: Pushed
f6b885c6dd1d: Mounted from library/python
0bf1b99a8791: Mounted from library/python
15f4af8927ad: Mounted from library/python
86e50e0709ee: Mounted from library/python
12b956927ba2: Mounted from library/node
266def75d28e: Mounted from library/node
29e49b59edda: Waiting
1777ac7d307b: Waiting
```

```
manager@ubuntu:~$ sudo docker push yoannchh/3dokr-worker
Using default tag: latest
The push refers to repository [docker.io/yoannchh/3dokr-worker]
be607a184d8f: Pushing 2.538MB/4.415MB
5f70bf18a086: Preparing
ab5582d0b6d1: Pushing 1.536kB
b245ac52f696: Pushing 2.56kB
9e11f0bc1297: Preparing
ef20a2883fd6: Waiting
74c0af6e0227: Waiting
```



```
manager@ubuntu:~$ sudo docker push yoannchh/3dokr-result
Using default tag: latest
The push refers to repository [docker.io/yoannchh/3dokr-result]
3bb7660f4ed7: Pushed
0735c2747d6d: Pushed
05c7843f0d3e: Pushed
1a3c6c533bfb: Pushed
83292216add: Mounted from library/node
820ec460f7e4: Mounted from library/node
c8d266bebfc2: Mounted from library/node
ccd778ffcca1: Mounted from library/node
12b956927ba2: Mounted from yoannchh/3dokr-vote
266def75d28e: Mounted from yoannchh/3dokr-vote
29e49b59edda: Mounted from yoannchh/3dokr-vote
1777ac7d307b: Mounted from yoannchh/3dokr-vote
latest: digest: sha256:cd053e695dc29efeddc2dcb57e4d88b2970103861eaa2685964aefd2
20a66bd0 size: 2840
manager@ubuntu:~$
```

Docker Compose :

Après avoir envoyé nos images vers les registres en ligne sur Docker Hub, Nous devons donc retravailler notre docker compose afin que le fichier puisse fonctionner avec Docker Swarm. Il faut prendre en compte plusieurs options supplémentaires ainsi que la modification d'options déjà présentes.

Il faut d'abord prendre en compte les nouvelles images que nous avons donc mis à disposition sur Docker Hub à travers l'option « image » à la place de l'option « build »

Il faut aussi ajouter l'option « deploy » qui va permettre de configurer certaines options spécifique à un cluster swarm. Ici nous pouvons définir le nombre de réplication mais aussi redéfinir le mode de redémarrage spécifique pour swarm.

Finalement, il faut aussi modifier les réseaux du docker compose en rajoutant une option pour indiquer explicitement que les réseaux sont en « overlay ».


```
services:
  redis:
    image: redis
    ports:
      - "6379:6379"
    deploy:
      replicas: 3
      restart_policy:
        condition: on-failure
    networks:
      - backend
    healthcheck: # On verifie que le service est bien up
      test: ["CMD", "redis-cli", "ping"]
      interval: 5s
      timeout: 5s
      retries: 3

  vote:
    image: yoannchh/3dokr-vote:latest
    ports:
      - "5001:5000"
    depends_on:
      - redis
    deploy:
      replicas: 3
      restart_policy:
        condition: on-failure
    networks:
      - frontend
      - backend
    healthcheck: # On fait une requete sur le service pour verifier qu'il est up
      test: ["CMD", "curl", "-f", "http://localhost:5000"]
      interval: 5s
      timeout: 5s
      retries: 3
```

```
worker:
  image: yoannchh/3dokr-worker:latest
  depends_on:
    - redis
    - postgres
  deploy:
    replicas: 3
    restart_policy:
      condition: on-failure
  networks:
    - backend
  healthcheck: # On verifie que le process n'a pas crash.
    test: ["CMD-SHELL", "pidof dotnet || exit 1"]
    interval: 5s
    timeout: 5s
    retries: 3
```

```
result:
  image: yoannchh/3dokr-result:latest
  ports:
    - "3000:3000"
  depends_on:
    - postgres
  deploy:
    replicas: 3
    restart_policy:
      condition: on-failure
  networks:
    - frontend
    - backend
  healthcheck: # On fait une requete sur le service pour verifier qu'il est up
    test: ["CMD", "curl", "-f", "http://localhost:3000"]
    interval: 5s
    timeout: 5s
    retries: 3
```

```
postgres:
  image: postgres
  ports:
    - "5432:5432"
  environment:
    POSTGRES_USER: postgres
    POSTGRES_PASSWORD: postgres
    POSTGRES_DB: postgres
  deploy:
    replicas: 1
    restart_policy:
      condition: on-failure
  volumes:
    - postgres_data:/var/lib/postgresql/data
  networks:
    - backend
  healthcheck: # On verifie que le service est bien up et ready pour read/write
    test: ["CMD-SHELL", "pg_isready -U postgres"]
    interval: 5s
    timeout: 5s
    retries: 3
```

```
volumes:
  postgres_data:
```

```
networks:
  frontend:
    driver: overlay # Create a custom overlay network for frontend services (vote and result)
  backend:
    driver: overlay # Create a custom overlay network for backend services (redis, postgres, worker)
```

Déploiement :

Maintenant, l'application est prête à être déployée. Nous vérifions quand même que toutes nos nodes sont bien disponibles avant de commencer le déploiement.

```
manager@ubuntu:~/Downloads/3DOCKER-main$ sudo docker node ls
[sudo] password for manager:
ID                HOSTNAME        STATUS        AVAILABILITY        MANAGER STATUS
S ENGINE VERSION
d40x7xtg7w30g38dqqqh3f1p * ubuntu         Ready         Active              Leader
24.0.7
eil5qph84nh7uutw7teyju303 ubuntu         Ready         Active
24.0.7
nnwiwmlc3no1rdku4sm35ec2t ubuntu         Ready         Active
24.0.7
manager@ubuntu:~/Downloads/3DOCKER-main$
```

Avec notre docker compose modifié, nous pouvons initialiser une nouvelle « stack » qui va s'occuper du déploiement et de la répartition de nos services à travers nos différentes nodes.

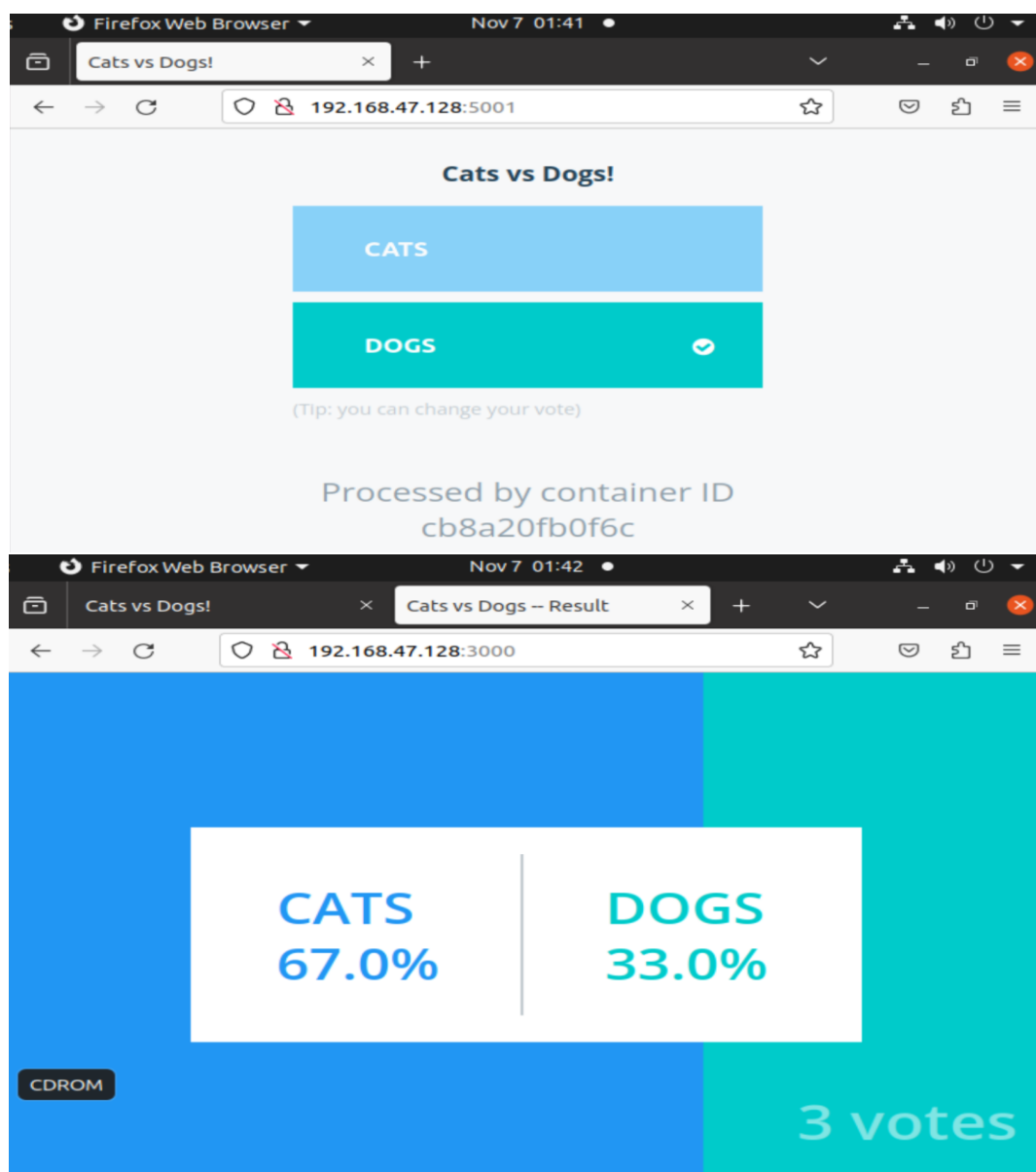
```
manager@ubuntu:~/Downloads/3DOCKER-main$ sudo docker stack deploy -c "docker-co
mpose Swarm.yml" catordog
Creating network catordog_backend
Creating network catordog_frontend
Creating service catordog_result
Creating service catordog_redis
Creating service catordog_vote
Creating service catordog_postgres
Creating service catordog_worker
manager@ubuntu:~/Downloads/3DOCKER-main$
```

```
root@ubuntu: /home/manager/Downloads/3DOCKER-main# docker ps
CONTAINER ID   IMAGE                                PORTS          NAMES                                COMMAND                                CREATED
STATUS        UP
e0ec28759e1b   yoannchh/3dokr-worker:latest        "dotnet Worker.dll"  2 minute
s ago        Up 2 minutes
20817ea85068   postgres:latest                     "docker-entrypoint.s..."  3 minute
s ago        Up 3 minutes  5432/tcp      catordog_postgres.1.ndoswle4ouynu834hqk686mkk
ae770fe32e8a   yoannchh/3dokr-vote:latest          "python app.py"      3 minute
s ago        Up 3 minutes  5000/tcp      catordog_vote.2.rqu89trkowl5isq63y8rmzrbv
2ee0ee335b45   yoannchh/3dokr-result:latest        "docker-entrypoint.s..."  3 minute
s ago        Up 3 minutes  3000/tcp      catordog_result.3.51swrb5woc5pf8169ec4v0ea5
bc98cbcf768a   redis:latest                         "docker-entrypoint.s..."  3 minute
s ago        Up 3 minutes  6379/tcp      catordog_redis.2.q4tlbpa3biy0t1x5t9xpitcnr
root@ubuntu: /home/manager/Downloads/3DOCKER-main#
```

```
root@ubuntu: /home/worker1# docker ps
CONTAINER ID   IMAGE                                COMMAND                                CREATED
STATUS        UP
e987ea997281   yoannchh/3dokr-worker:latest        "dotnet Worker.dll"  29 secon
ds ago       Up 27 seconds
d8a1fac93b47   yoannchh/3dokr-vote:latest          "python app.py"      48 secon
ds ago       Up 42 seconds  5000/tcp      catordog_vote.3.7eyxykhy7ltl2gwit2thifdon
8f23ad39189e   yoannchh/3dokr-result:latest        "docker-entrypoint.s..."  50 secon
ds ago       Up 43 seconds  3000/tcp      catordog_result.1.iw9u6z376o6m47y8f3rq6jmu
43c06f0844ba   redis:latest                         "docker-entrypoint.s..."  51 secon
ds ago       Up 44 seconds  6379/tcp      catordog_redis.1.nj29ayhwpwbqlkj3pgkxf99u
root@ubuntu: /home/worker1#
```

```
root@ubuntu: /home/worker2# docker ps
CONTAINER ID   IMAGE                                PORTS          NAMES                                COMMAND                                CREATED
7a317e0a930c   yoannchh/3dokr-worker:latest        5000/tcp       catordog_worker.1.jf6r44mmjkg3f6eirur1gxim3  "dotnet Worker.dll"                  20 seconds ago
cb8a20fb0f6c   yoannchh/3dokr-vote:latest          5000/tcp       catordog_vote.1.m4ihoc01195uqrf6day5mh4a5  "python app.py"                      38 seconds ago
7e7ac18c38ce   redis:latest                        6379/tcp       catordog_redis.3.3tsvzqy18ka39heksolq4kz4a  "docker-entrypoint.s..."           41 seconds ago
5d381f30a5dc   yoannchh/3dokr-result:latest        3000/tcp       catordog_result.2.o27js04wwwtzzsn3bp4p0rp5k  "docker-entrypoint.s..."           42 seconds ago
root@ubuntu: /home/worker2#
```

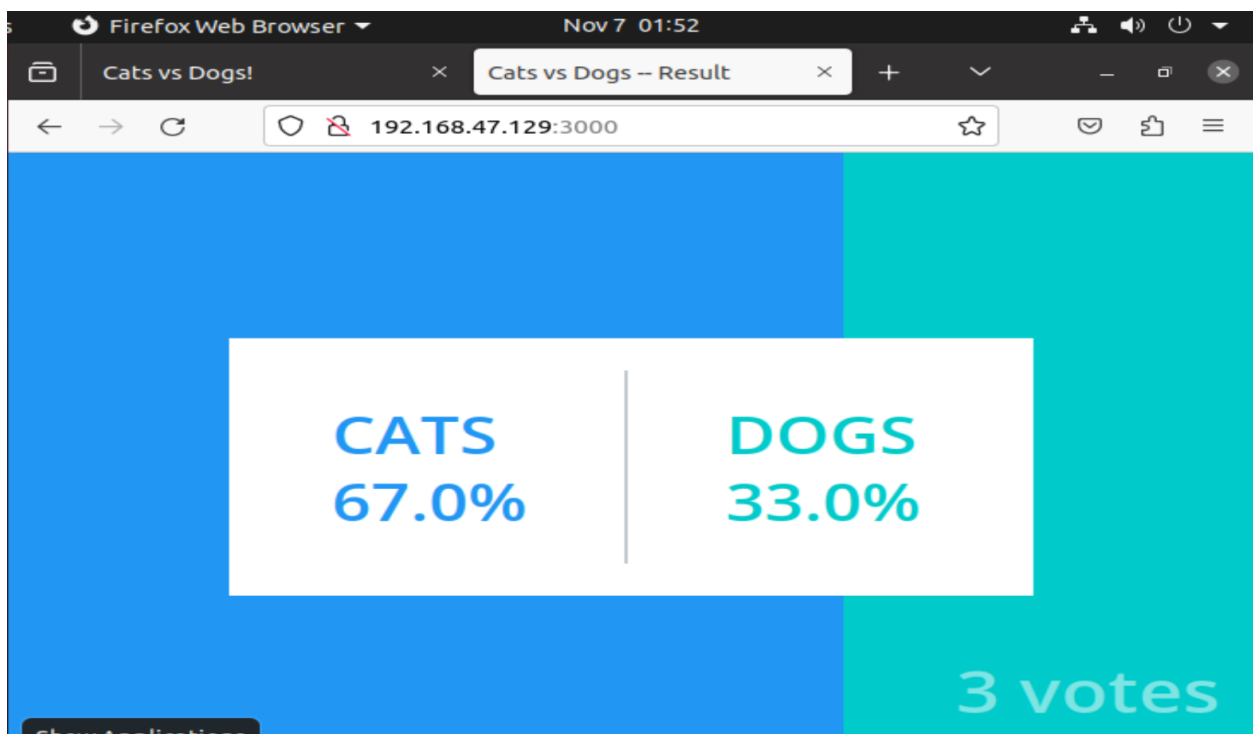
Les services sont bien déployés à travers nos nodes et nos sites sont bien accessibles et fonctionnels. Après avoir voté sur chaque machine cela donne :

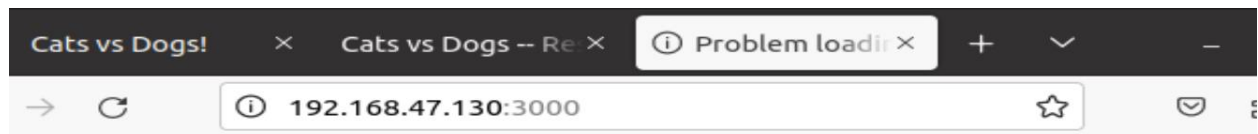


High availability :

Même après que une node est indisponible, il est possible d'accéder au sites à travers les multiples IPs des différentes nodes disponibles.

```
root@ubuntu:/home/manager/Downloads/3DOCKER-main# docker node update --availability pause ofherxj701wzkddlc5tz38o7
ofherxj701wzkddlc5tz38o7
root@ubuntu:/home/manager/Downloads/3DOCKER-main# docker node ls
ID                               HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
1xpfimyikle998ckrs5z80fws *    ubuntu   Ready   Active        Leader
ofherxj701wzkddlc5tz38o7       ubuntu   Ready   Pause
twjdyvmbvphagxvjmykjchiz4      ubuntu   Ready   Active
```





Unable to connect

Firefox can't establish a connection to the server at 192.168.47.130:3000.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

OM

[Try Again](#)

