Docker CE user manual

Contents

1. Introduction to docker CE

Docker Community Edition (Docker CE) is ideal for developers and small teams looking to get started with Docker and experimenting with container-based apps. Docker CE is available on many platforms, from desktop to cloud to server. Docker CE is available for macOS and Windows and provides a native experience to help you focus on learning Docker. You can build and share containers and automate the development pipeline all from a single environment.

Docker CE gives you the option to run **stable** or **edge** builds.

- **Stable** builds are released once per quarter.
- **Edge** builds are released once per month.

For more information about Docker CE, see Docker Community Edition.

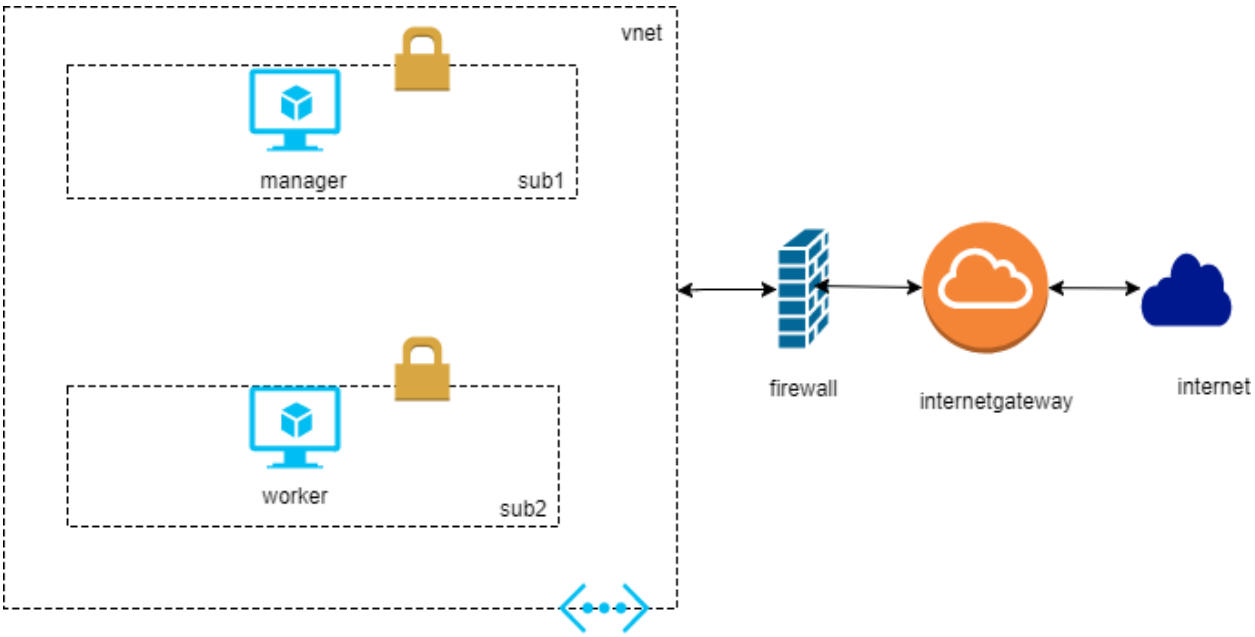
2. About this test drive

In this test drive, the user will be guided on how to create a **Docker swarm** and use it to deploy containerized voting application.

In Docker swarm mode, we will set up a swarm, that contains a Manager node and a worker node.

You will be guided step by step to perform all the above tasks in detail.

3. Architecture diagram



3.1 Description

The test drive is provisioned with two virtual machines, one is a manager node and other is a worker node. User will initialize swarm in the Manager node and then join the worker node to the swarm.

4. Use case

Deploying a sample voting app on a Docker swarm which has one manager node and one worker node in it.

Deploying the voting application as set of services on Docker swarm.

5. Information on how to access nodes

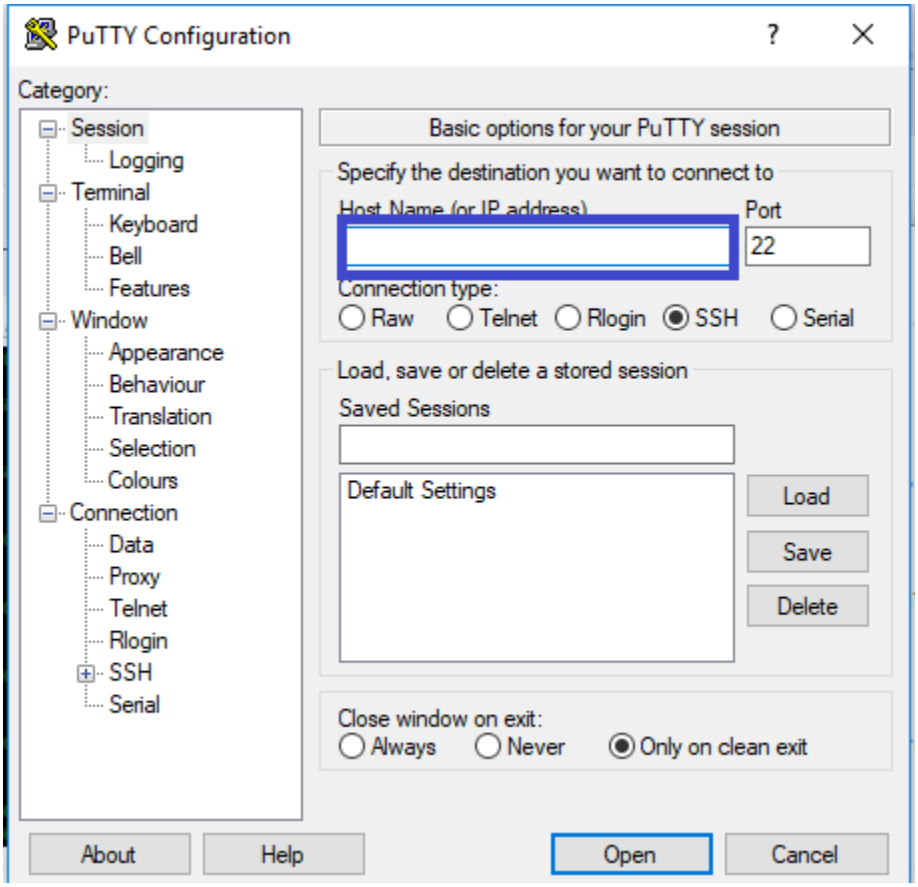
The following screenshot shows about the public and private IP's of manager and worker nodes.

It also shows the credentials to login into the nodes.

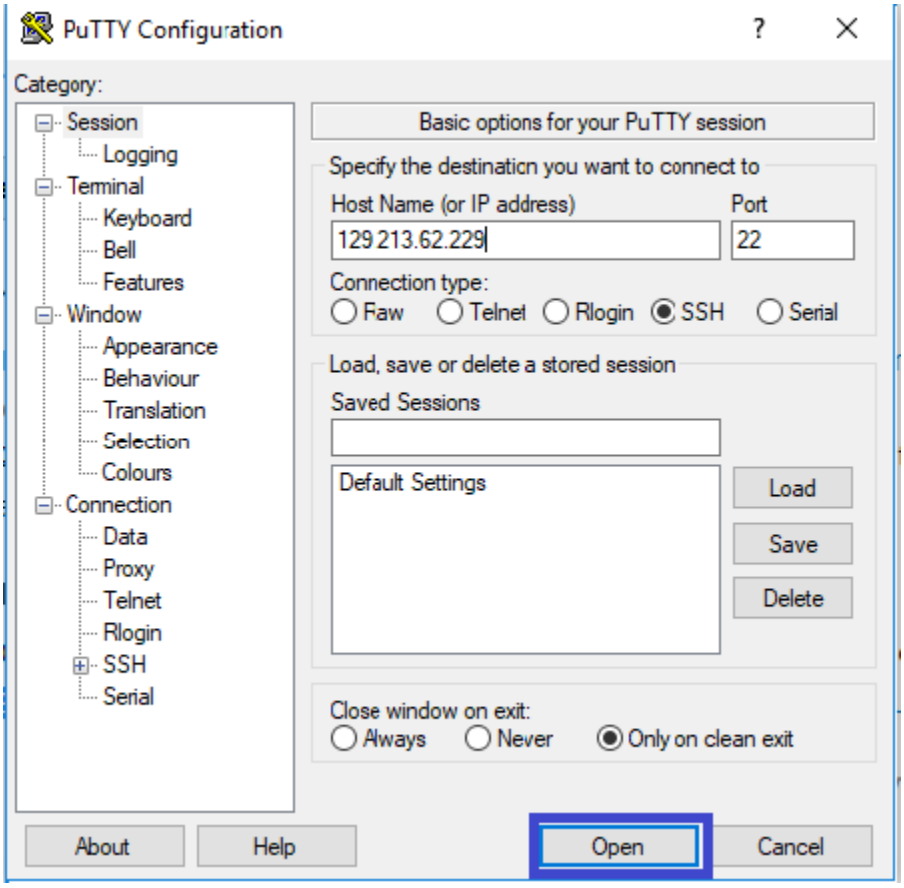
```
Outputs:

Manager_InstancePrivateIP = [
    10.2.3.2
]
Manager_InstancePublicIP = [
    129.146.81.177
]
Worker_01_InstancePrivateIP = [
    10.2.2.2
]
Worker_01_InstancePublicIP = [
    129.146.71.42
]
admin-password = Docker2017
admin-username = docker
```

Log in to the Manager node (an Ubuntu Virtual Machine) using an SSH client such as PuTTy.

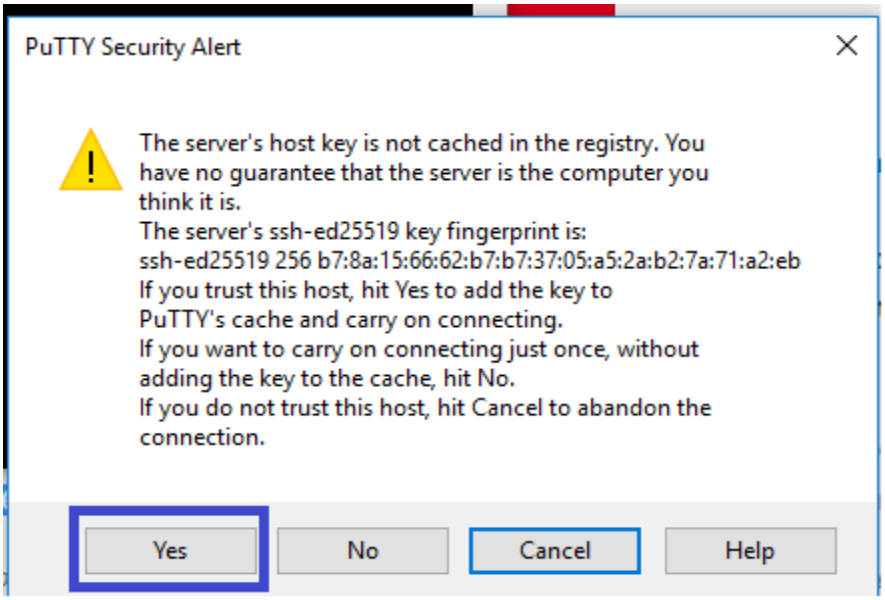
Here you need to enter the public ip of the manager node.



After entering the public IP then click on open button.



Then it shows a security alert, click on yes to proceed.

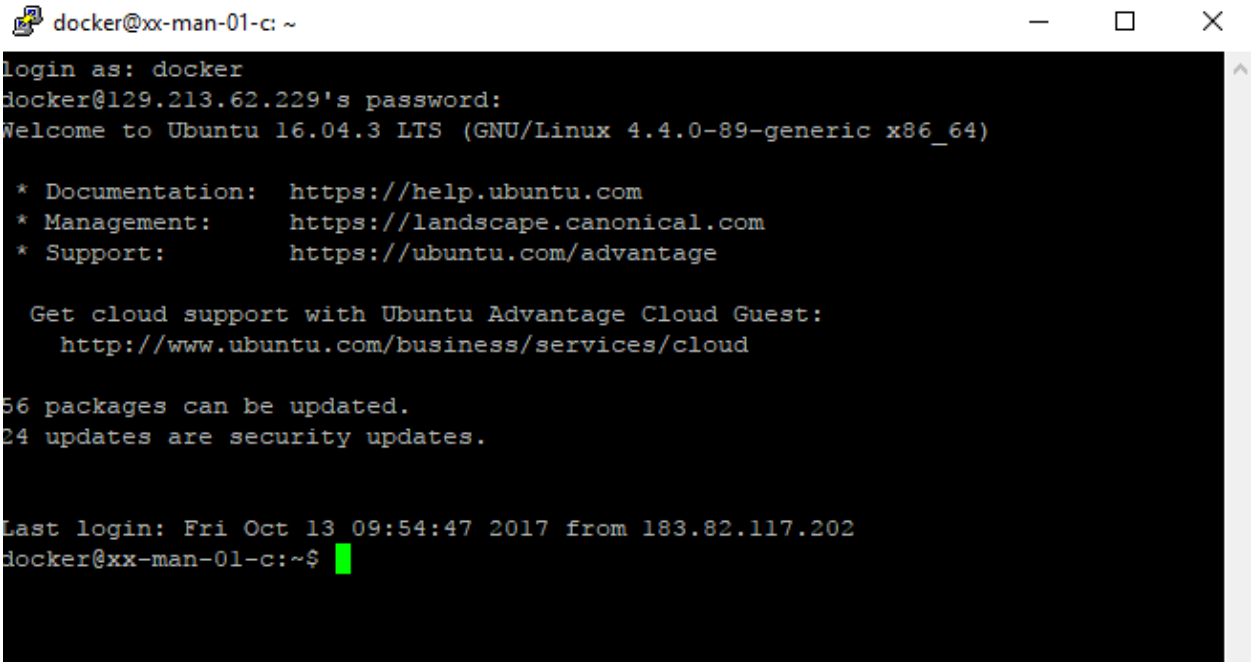


click on yes

Now you will provide the login credentials i.e. and hit enter.

Username: docker

Password: Docker2017



Now you have successfully logged in into the Manager node

6. Creating docker swarm

6.1 Configure the Manager Node to Run in Swarm Mode

Run the following command to create a new swarm and make the VM run as a Manager node in Docker swarm:

$ docker swarm init --advertise-addr <Manager-node-public-ip>

Ex: docker swarm init --advertise-addr 129.146.81.177

The output will be generated as follows which shows the command to be executed in the worker node so that it joins the docker swarm.

```
docker@dce-man-01-c:~$ docker swarm init --advertise-addr 129.146.81.247
Swarm initialized: current node (t7leeudqamguqj7ghmk74qhpt) is now a manager.

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

    docker swarm join --token SWMTKN-1-02n6manxwxzips4j2ijtbt0716ix1rkm5op3c3ubb1wcxbmec3-5nhelii2076efxbakwkqziq8r 129.146.81.247:2377

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

docker@dce-man-01-c:~$
```

6.2 Configure the Worker Node to Run in Swarm Mode

Log in to the worker node (an Ubuntu Virtual Machine) using an SSH client such as PuTTy, using worker node's public IP address provided.

Follow the above login procedure for worker node with worker public ip.

After opening the console of worker node run the command which is displayed after you run the ***docker swarm init*** in manager node to join the worker node to the swarm manager.

After performing the command, the output will be shown as follows.

```
docker@dce-wor-01-b:~$ docker swarm join --token SWMTKN-1-5mphp1kro4luwv8v95artfeowdsbtq5jdzufvw3rkhx71gqp4j-b91ttwxpp44e141vph0ubeewg 10.2.3.3:2377
This node joined a swarm as a worker.
docker@dce-wor-01-b:~$
```

The node joined a swarm as a worker

7. Building docker images

To build Docker images, we require the application files. To do this, we will clone a sample Voting Application repository into the Manager Node VM.

Again, open console of the manager node.

To clone the voting application repository run the following command:

$ git clone https://github.com/ashwinse/example-voting-app.git

This downloads the voting repository into the Manager node. Now you can build vote, result and worker images of the sample voting application.

```
docker@dce-man-01-c:~$ cd ~
docker@dce-man-01-c:~$ git clone https://github.com/dockersamples/example-voting-app.git
Cloning into 'example-voting-app'...
remote: Counting objects: 377, done.
remote: Total 377 (delta 0), reused 0 (delta 0), pack-reused 377
Receiving objects: 100% (377/377), 204.57 KiB | 0 bytes/s, done.
Resolving deltas: 100% (133/133), done.
Checking connectivity... done.
docker@dce-man-01-c:~$
```

7.1 Build image for vote

Navigate to the 'Vote' directory of the cloned repository and run the following commands:

$ cd example-voting-app/vote

$ docker build -t vote .

This will build the Docker image for the vote component of the Voting app.

```
docker@dce-man-01-c:~$ ls
docker-ce-install.sh  enable-password-auth.sh  example-voting-app  remote-exec.log
docker@dce-man-01-c:~$ cd example-voting-app/
docker@dce-man-01-c:~/example-voting-app$
```

```
docker@dce-man-01-c:~/example-voting-app$ cd vote
docker@dce-man-01-c:~/example-voting-app/vote$
```

Now run the following command to build the docker image for vote component of the voting application.

$ docker build -t vote .

```
docker@dce-man-01-c:~/example-voting-app/vote$ docker build -t vote .
Sending build context to Docker daemon  12.29kB
Step 1/7 : FROM python:2.7-alpine
2.7-alpine: Pulling from library/python
90f4dba627d6: Pull complete
a615e2cf13bb: Pull complete
21bec36dca7a: Pull complete
87e78cdc890d: Pull complete
Digest: sha256:2b1b7a67f8e93ba2fada53a189ab7b50c1cd117879ec5028996503cbf577b3d4
Status: Downloaded newer image for python:2.7-alpine
 ---> 9b06bbaac1c7
Step 2/7 : WORKDIR /app
 ---> 6a77f8cc4ff7
Removing intermediate container 1e512e1e32e5
Step 3/7 : ADD requirements.txt /app/requirements.txt
 ---> 3c9245bb6cbf
Step 4/7 : RUN pip install -r requirements.txt
 ---> Running in 54008a60d1d9
Collecting Flask (from -r requirements.txt (line 1))
  Downloading Flask-0.12.2-py2.py3-none-any.whl (83kB)
Collecting Redis (from -r requirements.txt (line 2))
  Downloading redis-2.10.6-py2.py3-none-any.whl (64kB)
Collecting gunicorn (from -r requirements.txt (line 3))
  Downloading gunicorn-19.7.1-py2.py3-none-any.whl (111kB)
Collecting itsdangerous>=0.21 (from Flask->-r requirements.txt (line 1))
  Downloading itsdangerous-0.24.tar.gz (46kB)
Collecting click>=2.0 (from Flask->-r requirements.txt (line 1))
  Downloading click-6.7-py2.py3-none-any.whl (71kB)
Collecting Jinja2>=2.4 (from Flask->-r requirements.txt (line 1))
```

7.2 Build image for result

Navigate to the 'result' directory of the cloned repository and run the following commands:

```
docker@dce-man-01-c:~/example-voting-app/vote$ cd ..
docker@dce-man-01-c:~/example-voting-app$ cd result
docker@dce-man-01-c:~/example-voting-app/result$
```

$ cd example-voting-app/result

$ docker build -t result .

This will build the Docker image for the result component of the voting application.

Now build the result image.

```
docker@dce-man-01-c:~/example-voting-app/result$ docker build -t result .
Sending build context to Docker daemon  195.1kB
Step 1/11 : FROM node:5.11.0-slim
5.11.0-slim: Pulling from library/node
8b87079b7a06: Extracting [==============================================>   ]  48.76MB/51.36MB
a3ed95caeb02: Download complete
1bb8eaf3d643: Download complete
5674f5dccbc4: Download complete
96a79bcf8a3b: Download complete
```

7.3 Build image for worker

Navigate to the 'Vote' directory of the cloned repository and run the following commands:

$ cd example-voting-app/worker

$ docker build -t worker .

This will build the Docker image for the worker component of the Voting app.

```
docker@dce-man-01-c:~/example-voting-app/result$ cd ..
docker@dce-man-01-c:~/example-voting-app$ cd worker
docker@dce-man-01-c:~/example-voting-app/worker$ docker build -t worker .
Sending build context to Docker daemon  20.48kB
Step 1/5 : FROM microsoft/dotnet:1.1.1-sdk
1.1.1-sdk: Pulling from microsoft/dotnet
10a267c67f42: Extracting [==============>                                  ]  15.04MB/52.58MB
fb5937da9414: Download complete
9021b2326a1e: Download complete
c63131473568: Download complete
a4274048307f: Downloading [========================>                        ]  65.39MB/132.8MB
61820b027a34: Downloading [================>                                  ]  62.13MB/187.9MB
```

7.4 Verifying the images

You can run this following command to view the list of images present in your manager node:

$ docker images ls

```
docker@dce-man-01-c:~/example-voting-app$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED              SIZE
worker              latest              065e86c642eb        About a minute ago   962MB
result              latest              6f1ac09015de        2 minutes ago        229MB
vote                latest              3ca23336d248        4 minutes ago        85.1MB
python              2.7-alpine          9b06bbaac1c7        3 weeks ago          72.3MB
microsoft/dotnet    1.1.1-sdk           a97efbca0c48        5 months ago         879MB
node                5.11.0-slim         cb888ea932ad        17 months ago        207MB
docker@dce-man-01-c:~/example-voting-app$
```

8. Deploying a Docker Stack File in the Docker Swarm

Navigate to the example voting app repository which you cloned earlier from the manager node:

$ cd example-voting-app/

$ ls

```
architecture.png  dockercloud.yml  docker-compose-javaworker.yml  docker-compose-simple.yml  docker-compose.yml  docker-stack.yml  LICENSE  MAINTAINERS  README.md  result  vote  worker
```

Edit the 'docker-stack.yml' file and change the image value under result, vote and worker sections of this file as highlighted below to edit the file press **insert** button to go into editing mode in the manager node console.

$ sudo vim docker-stack.yml

```
version: "3"
services:

  redis:
    image: redis:alpine
    ports:
      - "6379"
    networks:
      - frontend
    deploy:
      replicas: 1
      placement:
        constraints: [node.role == manager]
      update_config:
        parallelism: 2
        delay: 10s
      restart_policy:
        condition: on-failure
  db:
    image: postgres:9.4
    volumes:
      - db-data:/var/lib/postgresql/data
    networks:
      - backend
    deploy:
      placement:
        constraints: [node.role == manager]
  vote:
    image: vote
    ports:
      - 5000:80
    networks:
      - frontend
    depends_on:
      - redis
    deploy:
      replicas: 1
      update_config:
        parallelism: 2
```

```yaml
      restart_policy:
        condition: on-failure
  result:
    image: result
    ports:
      - 5001:80
    networks:
      - backend
    depends_on:
      - db
    deploy:
      replicas: 1
      update_config:
        parallelism: 2
        delay: 10s
      restart_policy:
        condition: on-failure
  worker:
    image: worker
    networks:
      - frontend
      - backend
    deploy:
      mode: replicated
      replicas: 1
      labels: [APP=VOTING]
      restart_policy:
        condition: on-failure
        delay: 10s
        max_attempts: 3
        window: 120s
      placement:
        constraints: [node.role == manager]

  visualizer:
    image: dockersamples/visualizer:stable
    ports:
      - "8080:8080"
    stop_grace_period: 1m30s
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    deploy:
      placement:
        constraints: [node.role == manager]
networks:
```

```
  frontend:
  backend:
volumes:
  db-data:
```

Save the file by hitting **Esc** and type **:wq!** . Now you are ready to deploy the stack file as services in the Docker swarm.
Run the following command to deploy the voting app stack:

$ docker stack deploy --compose-file docker-stack.yml votingapp

```
Creating network votingapp_backend
Creating network votingapp_frontend
Creating network votingapp_default
Creating service votingapp_vote
Creating service votingapp_result
Creating service votingapp_worker
Creating service votingapp_visualizer
Creating service votingapp_redis
Creating service votingapp_db
```

You have successfully deployed the example voting application as services in the Docker swarm.

You can view the list of services running in the Docker swarm by running the following command:

$ docker services ls

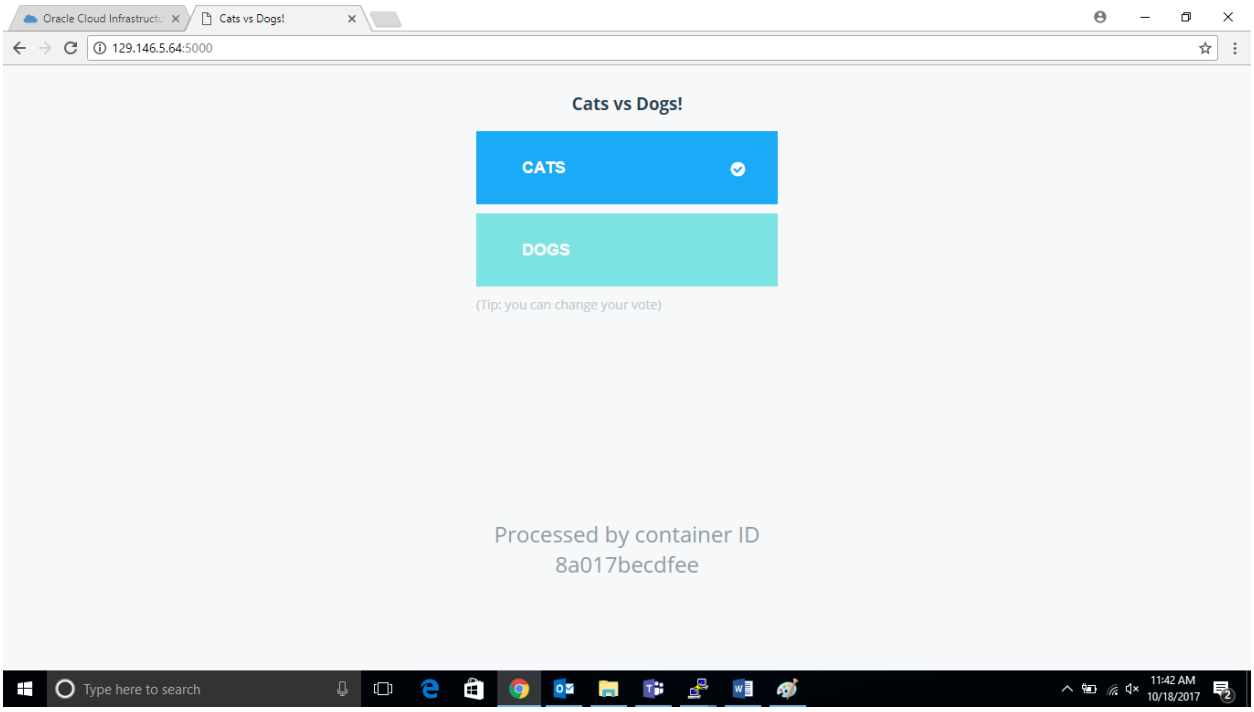
You will see similar output as shown below:

```
ID             NAME                   MODE         REPLICAS   IMAGE
6j45d0xmuhqg   votingapp_redis        replicated   2/2        redis:alpine
ebchhu8ijjks   votingapp_db           replicated   1/1        postgres:9.4
hyuo7kvue9bb   votingapp_vote         replicated   2/2        vote
nkn6k9zjfg1b   votingapp_worker       replicated   1/1        worker
plwvqnpkbhkx   votingapp_result       replicated   1/1        result
w3cuofvkakv0   votingapp_visualizer   replicated   1/1        dockersamples/visualizer:st
able
```

Now you can visit the voting site by using the manager node IP address provided at port 5000. Enter it in a web browser, as shown below:

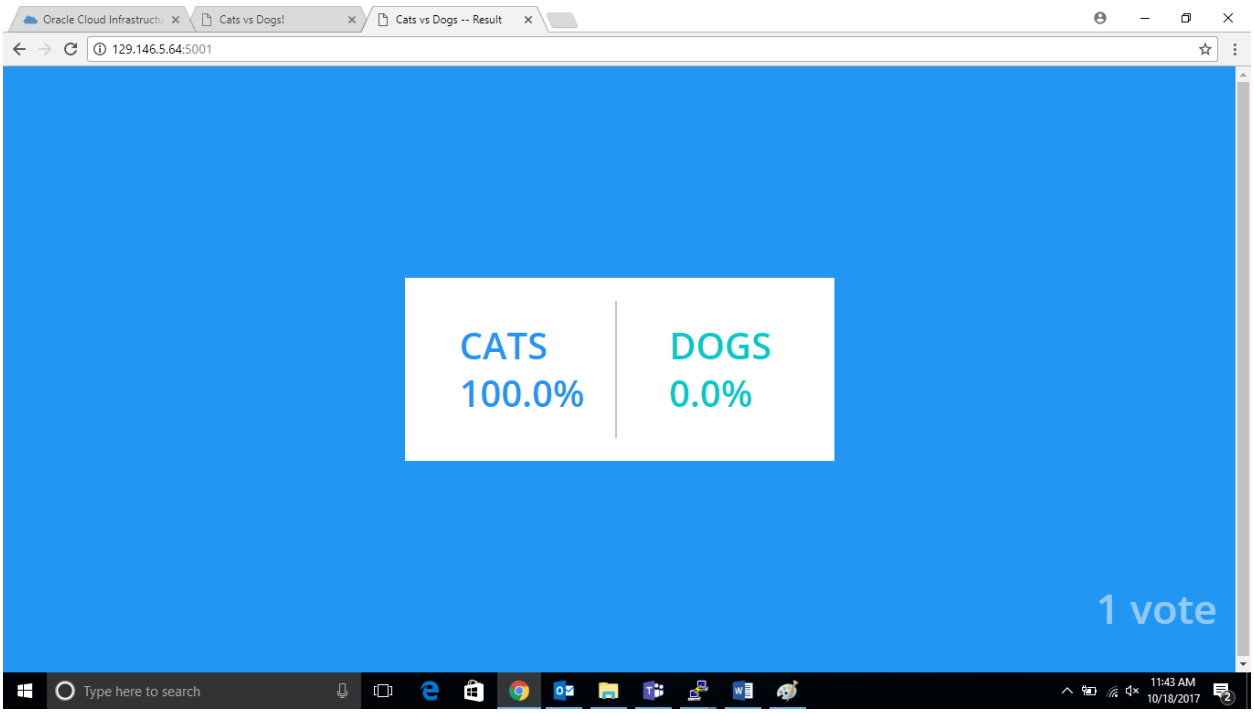
<manager node public IP>:5000

The manager node public IP is provided to you in **Access information** section of the test drive page, and via email.



You can visit the result site using the manager node IP address at port **5001** in the browser. Enter the address below to view it:

<manager node public IP>:5001



You have successfully created and deployed the example voting application as a set of services on Docker Swarm. You should now be more familiar with what Docker swarm mode is, and how to create and configure nodes in the Docker swarm. This Docker technology is the Community Edition, and is therefore "freeware."

Thank you for taking this Test drive, we hope you got the hang of Docker CE.