Deploy a stack to a swarm

Estimated reading time: 7 minutes

When running Docker Engine in swarm mode, you can use `docker stack deploy` to deploy a complete application stack to the swarm. The `deploy` command accepts a stack description in the form of a Compose file (/compose/compose-file/compose-file-v3/).

The `docker stack deploy` command supports any Compose file of version "3.0" or above. If you have an older version, see the upgrade guide (/compose/compose-file/compose-versioning/#upgrading).

To run through this tutorial, you need:

1. A Docker Engine running in swarm mode (/engine/swarm/swarm-mode/). If you're not familiar with swarm mode, you might want to read Swarm mode key concepts (/engine/swarm/key-concepts/) and How services work (/engine/swarm/how-swarm-mode-works/services/).

   > **Note**
   >
   > If you're trying things out on a local development environment, you can put your engine into swarm mode with `docker swarm init` .
   >
   > If you've already got a multi-node swarm running, keep in mind that all `docker stack` and `docker service` commands must be run from a manager node.

2. A current version of Docker Compose (/compose/install/).

Set up a Docker registry

Because a swarm consists of multiple Docker Engines, a registry is required to distribute images to all of them. You can use the Docker Hub (https://hub.docker.com) or maintain your own. Here's how to create a throwaway registry, which you can discard afterward.

1. Start the registry as a service on your swarm:

```
$ docker service create --name registry --publish published=5000,target=5000 registry:2
```

2. Check its status with `docker service ls`:

```
$ docker service ls

ID            NAME      REPLICAS  IMAGE                                                                          CO
l7791tpuwkco  registry  1/1       registry:2@sha256:1152291c7f93a4ea2ddc95e46d142c31e743b6dd70e194af9e6ebe530f782c17
```

   Once it reads `1/1` under `REPLICAS`, it's running. If it reads `0/1`, it's probably still pulling the image.

3. Check that it's working with `curl`:

```
$ curl http://localhost:5000/v2/

{}
```

Create the example application

The app used in this guide is based on the hit counter app in the Get started with Docker Compose (/compose/gettingstarted/) guide. It consists of a Python app which maintains a counter in a Redis instance and increments the counter whenever you visit it.

1. Create a directory for the project:

```
$ mkdir stackdemo
$ cd stackdemo
```

2. Create a file called `app.py` in the project directory and paste this in:

```
from flask import Flask
from redis import Redis

app = Flask(__name__)
redis = Redis(host='redis', port=6379)

@app.route('/')
def hello():
    count = redis.incr('hits')
    return 'Hello World! I have been seen {} times.\n'.format(count)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000, debug=True)
```

3. Create a file called `requirements.txt` and paste these two lines in:

```
flask
redis
```

4. Create a file called `Dockerfile` and paste this in:

```
FROM python:3.4-alpine
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

5. Create a file called `docker-compose.yml` and paste this in:

```
version: "3.9"

services:
  web:
    image: 127.0.0.1:5000/stackdemo
    build: .
    ports:
      - "8000:8000"
  redis:
    image: redis:alpine
```

The image for the web app is built using the Dockerfile defined above. It's also tagged with `127.0.0.1:5000` - the address of the registry created earlier. This is important when distributing the app to the swarm.

Test the app with Compose

1. Start the app with `docker-compose up` . This builds the web app image, pulls the Redis image if you don't already have it, and creates two containers.

   You see a warning about the Engine being in swarm mode. This is because Compose doesn't take advantage of swarm mode, and deploys everything to a single node. You can safely ignore this.

```
$ docker-compose up -d

WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in
a swarm. All containers are scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Creating network "stackdemo_default" with the default driver
Building web
...(build output)...
Creating stackdemo_redis_1
Creating stackdemo_web_1
```

2. Check that the app is running with `docker-compose ps`:

```
$ docker-compose ps

      Name                     Command               State           Ports
-------------------------------------------------------------------------------------
stackdemo_redis_1   docker-entrypoint.sh redis ...   Up      6379/tcp
stackdemo_web_1     python app.py                    Up      0.0.0.0:8000->8000/tcp
```

You can test the app with `curl`:

```
$ curl http://localhost:8000
Hello World! I have been seen 1 times.

$ curl http://localhost:8000
Hello World! I have been seen 2 times.

$ curl http://localhost:8000
Hello World! I have been seen 3 times.
```

3. Bring the app down:

```
$ docker-compose down --volumes

Stopping stackdemo_web_1 ... done
Stopping stackdemo_redis_1 ... done
Removing stackdemo_web_1 ... done
Removing stackdemo_redis_1 ... done
Removing network stackdemo_default
```

Push the generated image to the registry

To distribute the web app's image across the swarm, it needs to be pushed to the registry you set up earlier. With Compose, this is very simple:

```
$ docker-compose push

Pushing web (127.0.0.1:5000/stackdemo:latest)...
The push refers to a repository [127.0.0.1:5000/stackdemo]
5b5a49501a76: Pushed
be44185ce609: Pushed
bd7330a79bcf: Pushed
c9fc143a069a: Pushed
011b303988d2: Pushed
latest: digest: sha256:a81840ebf5ac24b42c1c676cbda3b2cb144580ee347c07e1bc80e35e5ca76507 size: 1372
```

The stack is now ready to be deployed.

Deploy the stack to the swarm

1. Create the stack with `docker stack deploy` :

```
$ docker stack deploy --compose-file docker-compose.yml stackdemo

Ignoring unsupported options: build

Creating network stackdemo_default
Creating service stackdemo_web
Creating service stackdemo_redis
```

The last argument is a name for the stack. Each network, volume and service name is prefixed with the stack name.

2. Check that it's running with `docker stack services stackdemo`:

```
$ docker stack services stackdemo

ID            NAME             MODE        REPLICAS  IMAGE
orvjk2263y1p  stackdemo_redis  replicated  1/1       redis:3.2-alpine@sha256:f1ed3708f538b537eb9c2a7dd50dc90a706f7debd7e
s1nf0xy8t1un  stackdemo_web    replicated  1/1       127.0.0.1:5000/stackdemo@sha256:adb070e0805d04ba2f92c724298370b7a4e
```

Once it's running, you should see `1/1` under `REPLICAS` for both services. This might take some time if you have a multi-node swarm, as images need to be pulled.

As before, you can test the app with `curl`:

```
$ curl http://localhost:8000
Hello World! I have been seen 1 times.

$ curl http://localhost:8000
Hello World! I have been seen 2 times.

$ curl http://localhost:8000
Hello World! I have been seen 3 times.
```

Thanks to Docker's built-in routing mesh, you can access any node in the swarm on port 8000 and get routed to the app:

```
$ curl http://address-of-other-node:8000
Hello World! I have been seen 4 times.
```

3. Bring the stack down with `docker stack rm`:

```
$ docker stack rm stackdemo

Removing service stackdemo_web
Removing service stackdemo_redis
Removing network stackdemo_default
```

4. Bring the registry down with `docker service rm`:

```
$ docker service rm registry
```

5. If you're just testing things out on a local machine and want to bring your Docker Engine out of swarm mode, use `docker swarm leave`:

```
$ docker swarm leave --force

Node left the swarm.
```

guide (/search/?q=guide), swarm mode (/search/?q=swarm mode), composefile (/search/?q=composefile), stack (/search/?q=stack), compose (/search/?q=compose), deploy (/search/?q=deploy)