Deploy to Swarm

*Estimated reading time: 4 minutes*

Prerequisites

* Download and install Docker Desktop as described in [Orientation and setup](https://docs.docker.com/get-started/).
* Work through containerizing an application in [Part 2](https://docs.docker.com/get-started/part2/).
* Make sure that Swarm is enabled on your Docker Desktop by typing docker system info, and looking for a message Swarm: active (you might have to scroll up a little).

If Swarm isn’t running, simply type docker swarm init in a shell prompt to set it up.

Introduction

Now that we’ve demonstrated that the individual components of our application run as stand-alone containers and shown how to deploy it using Kubernetes, let’s look at how to arrange for them to be managed by Docker Swarm. Swarm provides many tools for scaling, networking, securing and maintaining your containerized applications, above and beyond the abilities of containers themselves.

In order to validate that our containerized application works well on Swarm, we’ll use Docker Desktop’s built in Swarm environment right on our development machine to deploy our application, before handing it off to run on a full Swarm cluster in production. The Swarm environment created by Docker Desktop is *fully featured*, meaning it has all the Swarm features your app will enjoy on a real cluster, accessible from the convenience of your development machine.

Describe apps using stack files

Swarm never creates individual containers like we did in the previous step of this tutorial. Instead, all Swarm workloads are scheduled as *services*, which are scalable groups of containers with added networking features maintained automatically by Swarm. Furthermore, all Swarm objects can and should be described in manifests called *stack files*. These YAML files describe all the components and configurations of your Swarm app, and can be used to easily create and destroy your app in any Swarm environment.

Let’s write a simple stack file to run and manage our bulletin board. Place the following in a file called bb-stack.yaml:

version: '3.7'

services:

bb-app:

image: bulletinboard:1.0

ports:

- "8000:8080"

In this Swarm YAML file, we have just one object: a service, describing a scalable group of identical containers. In this case, you’ll get just one container (the default), and that container will be based on your bulletinboard:1.0 image created in [Part 2](https://docs.docker.com/get-started/part2/) of the Quickstart tutorial. In addition, We’ve asked Swarm to forward all traffic arriving at port 8000 on our development machine to port 8080 inside our bulletin board container.

**Kubernetes Services and Swarm Services are very different!** Despite the similar name, the two orchestrators mean very different things by the term ‘service’. In Swarm, a service provides both scheduling *and* networking facilities, creating containers and providing tools for routing traffic to them. In Kubernetes, scheduling and networking are handled separately: *deployments* (or other controllers) handle the scheduling of containers as pods, while *services* are responsible only for adding networking features to those pods.

Deploy and check your application

1. Deploy your application to Swarm:
2. docker stack deploy -c bb-stack.yaml demo

If all goes well, Swarm will report creating all your stack objects with no complaints:

Creating network demo\_default

Creating service demo\_bb-app

Notice that in addition to your service, Swarm also creates a Docker network by default to isolate the containers deployed as part of your stack.

1. Make sure everything worked by listing your service:
2. docker service ls

If all has gone well, your service will report with 1/1 of its replicas created:

ID NAME MODE REPLICAS IMAGE PORTS

il7elwunymbs demo\_bb-app replicated 1/1 bulletinboard:1.0 \*:8000->8080/tcp

This indicates 1/1 containers you asked for as part of your services are up and running. Also, we see that port 8000 on your development machine is getting forwarded to port 8080 in your bulletin board container.

1. Open a browser and visit your bulletin board at localhost:8000; you should see your bulletin board, the same as when we ran it as a stand-alone container in Part 2 of the Quickstart tutorial.
2. Once satisfied, tear down your application:
3. docker stack rm demo

Conclusion

At this point, we have successfully used Docker Desktop to deploy our application to a fully-featured Swarm environment on our development machine. We haven’t done much with Swarm yet, but the door is now open: you can begin adding other components to your app and taking advantage of all the features and power of Swarm, right on your own machine.

In addition to deploying to Swarm, we have also described our application as a stack file. This simple text file contains everything we need to create our application in a running state; we can check it into version control and share it with our colleagues, allowing us to distribute our applications to other clusters (like the testing and production clusters that probably come after our development environments) easily.

Swarm and CLI references

Further documentation for all new Swarm objects and CLI commands used in this article are available here:

* [Swarm Services](https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/)
* [Swarm Stacks](https://docs.docker.com/engine/swarm/stack-deploy/)
* [docker stack \*](https://docs.docker.com/engine/reference/commandline/stack/)
* [docker service \*](https://docs.docker.com/engine/reference/commandline/service/)