Docker Compose
and
Docker Swarm

Frank Burkholder

Objectives

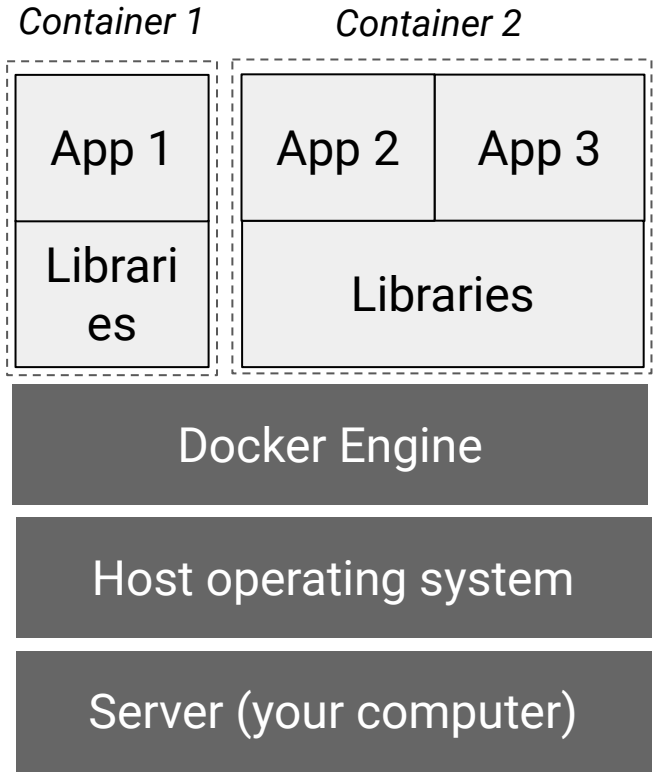
- Docker review
- Docker Compose motivation
- Docker Compose tutorial
- Docker Swarm motivation
- Docker Swarm & Compose walkthrough
- References

Docker review

From founder and CTO of Docker Soloman Hykes, as motivation for Docker:

“Shipping code to the server should not be hard.”

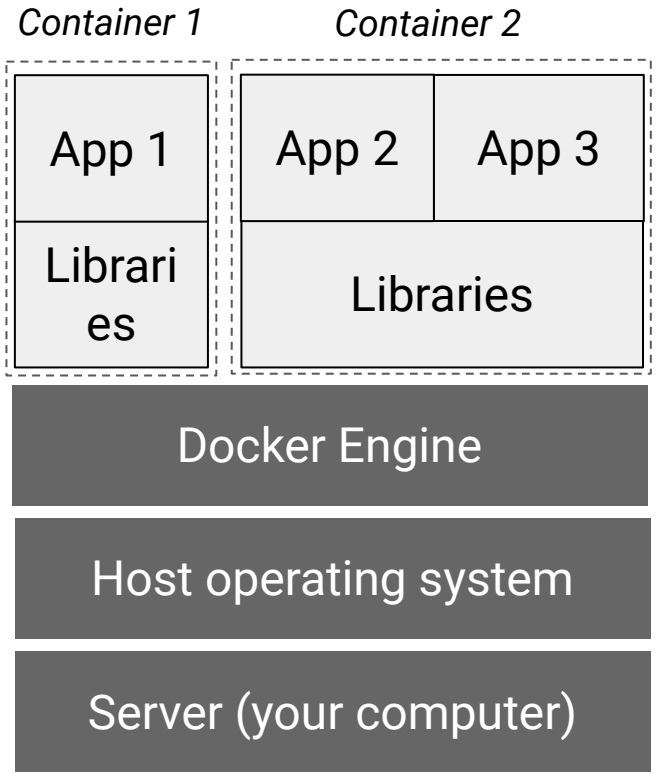
The developer can specify what’s inside the container, and IT just needs to handle the container.



No matter where the container goes, what’s inside will work the same way.

Docker review

- Docker is an open-source project based on Linux containers.
- Docker is a software *containerization* platform.
- A docker *container* is a stand-alone piece of software that includes everything needed to run it.
- Containers are isolated from each other, but can share libraries where possible.
- Containers are created with Linux, but share a *kernel* with almost any type of OS.



A kernel is the central part of the OS.

Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications.

With Compose you use a YAML file to configure your application’s services, then with a single command you create and start all your services.

YAML: YAML Ain’t Markup Language

Docker Compose Use Cases

Development
The ability to run an application in an isolated, consistent environment and interact with it is crucial.

Automated Testing Environment
Automated end-to-end testing requires a representative environment in which to run tests.

Single Host Deployment
You can use Compose to deploy to a remote Docker Engine.

Using Compose at a high level

1. Define your app's environment with a `Dockerfile`.
2. Define the services that make up your app in `docker-compose.yml`
3. Run `$ docker compose up` to run your application.

Docker Compose tutorial

Let's do the "Getting Started" Tutorial in the Docker Compose documentation.

An application is comprised of services, and services defined by containers.

We'll be building an application using Flask and Redis containers.

Docker Compose command-line reference

Docker Compose compose file reference

Scaling your application with Docker Swarm

Docker Compose creates and isolated application on a single host (a single Docker Engine)

What if you wish to create multiple applications to scale your service?

Enter Docker Swarm

In Docker Swarm, you create multiple Docker Engines where each Engine runs your application.

A swarm consists of multiple Docker Engine which run in swarm mode and act as managers (to manage membership and delegation) and workers (which run swarm services). A given Docker Engine can be a manager, a worker, or perform both roles.

Docker Swarm vs Kubernetes

Kubernetes and Docker are two of the major players in container orchestration.

Kubernetes
Kubernetes is an open-source platform created by Google for container deployment operations, scaling up and down, and automation across the clusters of hosts.

Docker Swarm
Docker Swarm is an open-source container orchestration platform and is the native clustering engine for and by Docker. Any software, services, or tools that run with Docker containers run equally well in Swarm. Swarm turns a pool of Docker hosts into a virtual, single host.

Source: Blog Savaram Ravindra

Docker Swarm vs Kubernetes

Go through blog highlights:
https://thenewstack.io/kubernetes-vs-docker-swarm-whats-the-difference/

Conclusion
“Kubernetes supports higher demands with more complexity while Docker Swarm offers a simple solution that is quick to get started with. Docker Swarm has been quite popular among developers who prefer fast deployments and simplicity. Simultaneously, Kubernetes is utilized in production environments by various high profile internet firms running popular services.”

Source: Blog Savaram Ravindra

Docker Swarm vocabulary

A **node** is an instance of the Docker engine participating in the swarm.

- can run one or more locally or in the cloud.
- have roles: **manager** and **worker**

To deploy your application to a swarm, you submit a **service** definition to a **manager** node.

Manager nodes perform the cluster management functions required to maintain the swarm. Manager nodes elect a single leader to conduct orchestration tasks.

Worker nodes receive and execute tasks dispatched from manager nodes.

Docker Swarm services and tasks

A **service** is the definition of the tasks to execute on the manager or worker nodes. (Services defined in the docker-compose.yml file). It is the central structure of the swarm system and the primary root of user interaction with the swarm.

When you create a service, you specify which container image (or application) to use and which commands to execute inside running containers.

A **task** carries a Docker container and the commands to run inside the container. It is the atomic scheduling unit of swarm. Manager nodes assign tasks to worker nodes according to the **number of replicas** set in the service **scale**.

Docker Swarm & Compose walkthrough

Go to Docker labs for beginners, and select 3. Deploying an app to a Swarm

- A front-end web app in Python or ASP.NET Core which lets you vote between two options
- A Redis or NATS queue which collects new votes
- A .NET Core, Java or .NET Core 2.1 worker which consumes votes and stores them in…
- A Postgres or TiDB database backed by a Docker volume
- A Node.js or ASP.NET Core SignalR webapp which shows the results of the voting in real time

References

- Docker Documentation
- Docker Compose command-line reference
- Docker Compose compose file reference
- Docker Swarm command-line reference
- Docker Swarm tutorial

Objectives

- Docker review
- Docker Compose motivation
- Docker Compose tutorial
- Docker Swarm motivation
- Docker Swarm & Compose walkthrough
- References