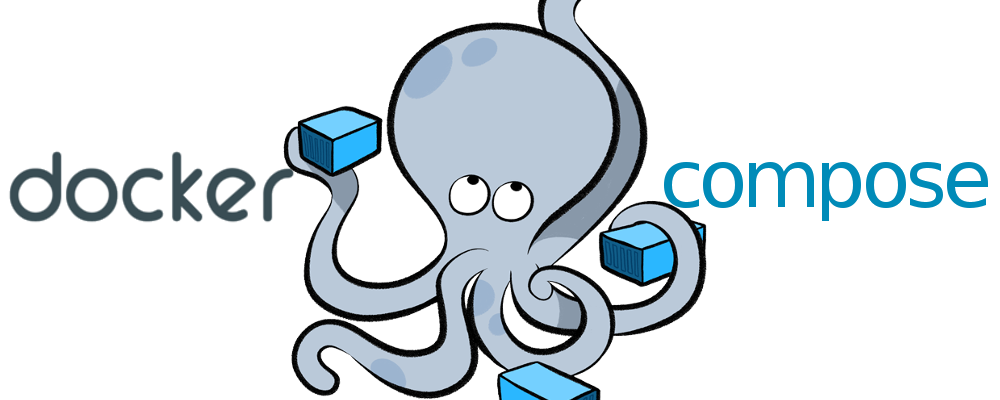
A Practical Introduction to Docker Compose

[](https://hackernoon.com/@faizanbashir?source=post_header_lockup)

[Faizan Bashir](https://hackernoon.com/@faizanbashir)Follow

May 13, 2018



**TL;DR**

Docker containers opened a world of possibilities for the tech community, hassles in setting up new software were decreased unlike old times when a mess was to be sorted by a grievous format, it reduced the time to set up and use new software which eventually played a big part for techies to learn new things, roll it out in a container and scrap it when done. Things became easy, and the best thing its open source anyone and everyone can use it, comes with a little learning curve though.

Out of the myriad possibilities was the possibility of implementing complex technology stacks for our applications, which previously would have been the domain of experts. Today with the help of containers software engineers with sound understanding of the underlying systems can implement a complex stack and why not it’s the need of the hour, the figure of speech “Jack of all trades” got a fancy upgrade; “Master of some” based on the needs of the age. Simply put “T” shaped skills.

The possibility of defining a complex stack in a file and running it with a single command, pretty tempting huh. The guys at Docker Inc. choose to call it Docker compose.



In this article, we will use Docker’s example Voting App and deploy it using Docker compose.

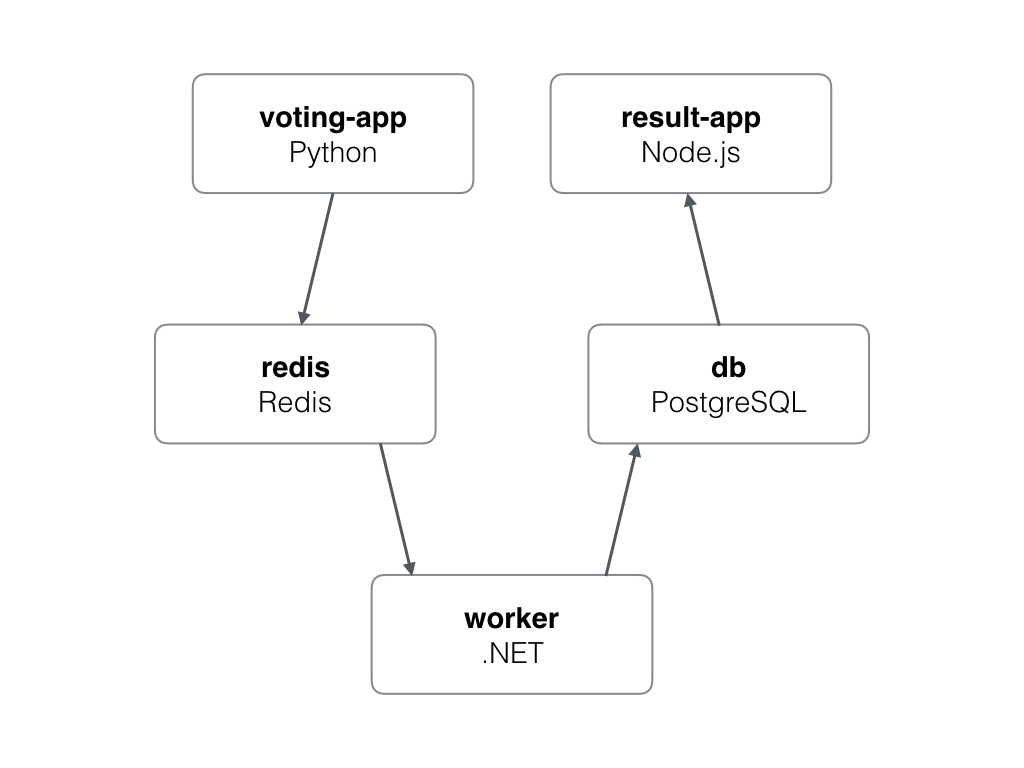
**Docker Compose**

In the words of Docker Inc.

*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 the services from your configuration.*

**The Voting App**

Introducing the most favourite demonstration app for the Docker community “The Voting App”, as if it needs an introduction at all. This is a simple application based on micro-services architecture, consisting of 5 simple services.



Voting app architecture [<https://github.com/docker/example-voting-app>]

1. **Voting-App**: Frontend of the application written in Python, used by users to cast their votes.
2. **Redis**: In-memory database, used as intermediate storage.
3. **Worker**: .Net service, used to fetch votes from Redis and store in Postres database.
4. **DB**: PostgreSQL database, used as database.
5. **Result-App**: Frontend of the application written in Node.js, displays the voting results.

The Voting repo has a file called docker-compose.yml this file contains the configuration for creating the containers, exposing ports, binding volumes and connecting containers through networks required for the voting app to work. Sounds like a lot of pretty long docker run and docker network createcommands otherwise, docker compose allows us to put all of that stuff in a single docker-compose file in [yaml](http://yaml.org/start.html" \t "_blank) format.

Git clone and cd into the voting app repo.

[dockersamples/example-voting-app  
example-voting-app - Example Docker Compose appgithub.com](https://github.com/dockersamples/example-voting-app)

**Compose Time**

With all of our application defined in a single compose file we can take a sigh of relief, chill and simply run the application. The beauty of compose lies in the fact that a single command creates all the services, wires up the networks(literally), mounts all volumes and exposes the ports. Its time to welcome the up command, its performs all of the aforementioned tasks.

$ docker-compose up

After lots of “Pull complete”, hundreds of megabytes and few minutes (maybe more). . .

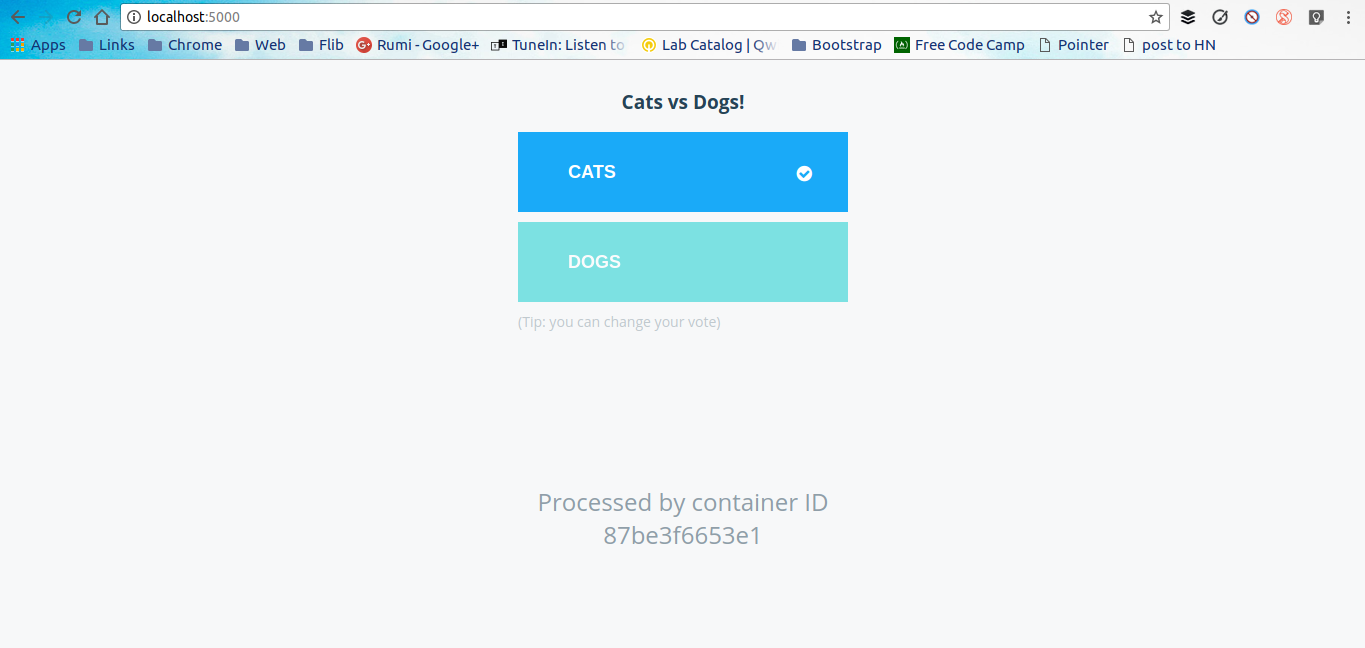
Voila, we have the voting app up and running.

Command docker ps lists all the running containers

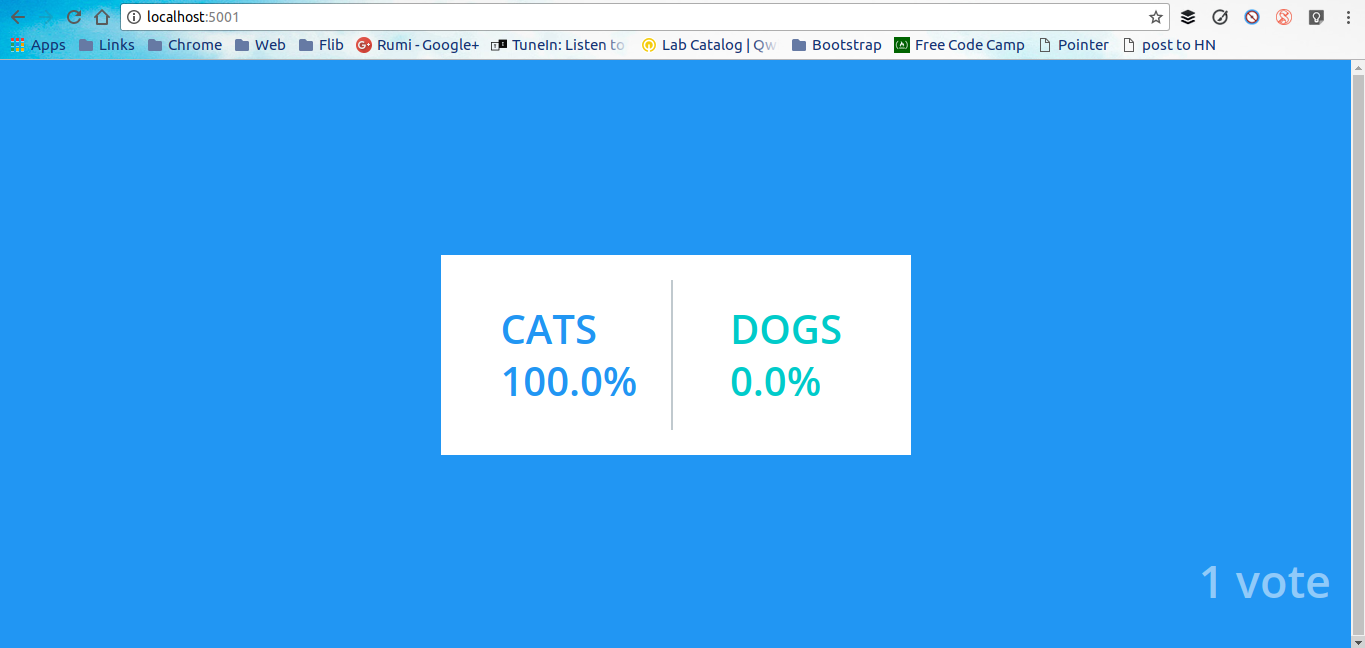
$ docker ps -a --format="table {{.Names}}\t{{.Image}}\t{{.Ports}}"   
NAMES IMAGE PORTS  
voting\_worker\_1 voting\_worker   
db postgres:9.4 5432/tcp  
voting\_vote\_1 voting\_vote 0.0.0.0:5000->80/tcp  
voting\_result\_1 voting\_result 0.0.0.0:5858->5858/tcp, 0.0.0.0:5001->80/tcp  
redis redis:alpine 0.0.0.0:32768->6379/tcp

The above command displays all the running containers, respective images and the exposed port numbers.

The Voting app can be accessed on [http://localhost:5000](http://localhost:5000/)



Likewise the Voting results app can be accessed on [http://localhost:5001](http://localhost:5001/)



Each vote cast on the Voting app is first stored in the Redis in-memory database, the .Net worker service fetches the vote and stores it in the Postgres DB which is accessed by the Node.js frontend.

**Compose Features**

Compose provide the flexibility to use a project name to isolate the environments from each other, the project name is the base name of the directory that contains the project. In our voting app this is signified by the name of the containers voting\_worker\_1 where voting is the base name of the directory. We can set a custom project name using the -p flag followed by the custom name.

Compose preserves all volumes used by the services defined in the compose file, thus no data is lost when the containers are recreated using docker-compose up. Another cool feature is that only the containers which have changed are recreated, the containers whose state did not change remain untouched.

Another cool feature is the support for variables in the compose file, we can define variables in a .env file and use them in the docker-compose file. Here the POSTGRES\_VERSION=9.4 can defined in the environment file or can be defined in the shell. It is used in the compose file in the following manner:

db:  
 image: "postgres:${POSTGRES\_VERSION}"

**Command Cheat Sheet**

Its easy as breeze to start, stop and play around with compose.

$ docker-compose up -d  
$ docker-compose down  
$ docker-compose start  
$ docker-compose stop  
$ docker-compose build  
$ docker-compose logs -f db  
$ docker-compose scale db=4  
$ docker-compose events  
$ docker-compose exec db bash

**Summary**

Docker Compose is a great tool to quickly deploy and scrap containers, the compose file can run seamlessly on any machine installed with docker-compose. Experimentation and learning technologies is just a Compose file away ;).

I hope this article helped in the understanding of Docker Compose. I’d love to hear about how you use Docker Compose in your projects. Clap if it increased your knowledge, help it reach more people.