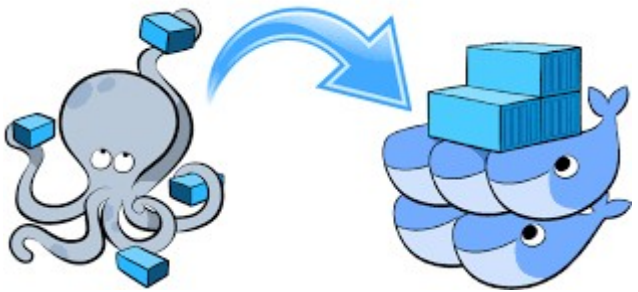
Introduction to docker compose

NobleProg

“a tool for defining and running complex applications with Docker. With Compose, you define a multi-container application in a single file, then spin your application up in a single command which does everything that needs to be done to get it running.”



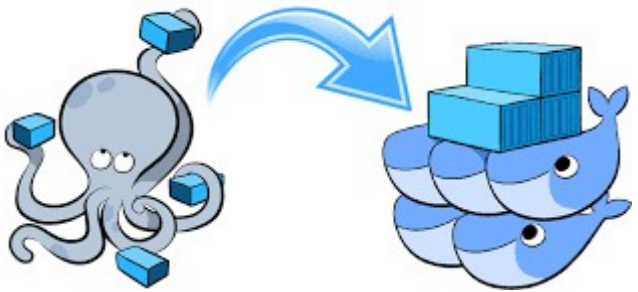
Functionality

The main function of Docker Compose is the creation of microservice architecture, meaning the containers and the links between them. But the tool is capable of much more:

- Building images (if an appropriate Dockerfile is provided)

- Scaling containers running a given service

- Healing, i.e., re-running containers that have stopped



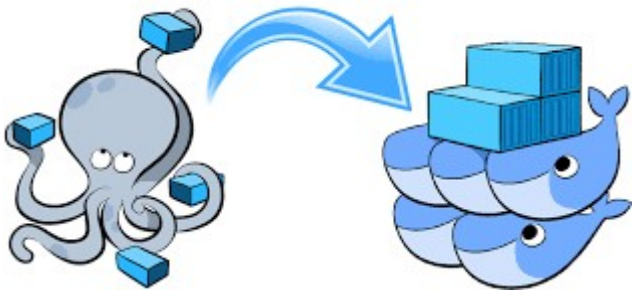
Docker Compose Workflow

There are three steps to using Docker Compose:

1) Define each service in a Dockerfile.

2) Define the services and their relation to each other in the docker-compose.yml file.

3) Use docker-compose up to start the system.



NobleProg

Docker Compose example

```
.
├── commander
│   └── Dockerfile
└── docker-compose.yml
```

```yaml
version: '3'
services:
  redis:
    container_name: redis
    hostname: redis
    image: redis

  redis-commander:
    container_name: redis-commander
    hostname: redis-commander
    image: rediscommander/redis-commander:latest
    build: commander
    restart: always
    environment:
    - REDIS_HOSTS=local:redis:6379
    ports:
    - 8081:8081
```