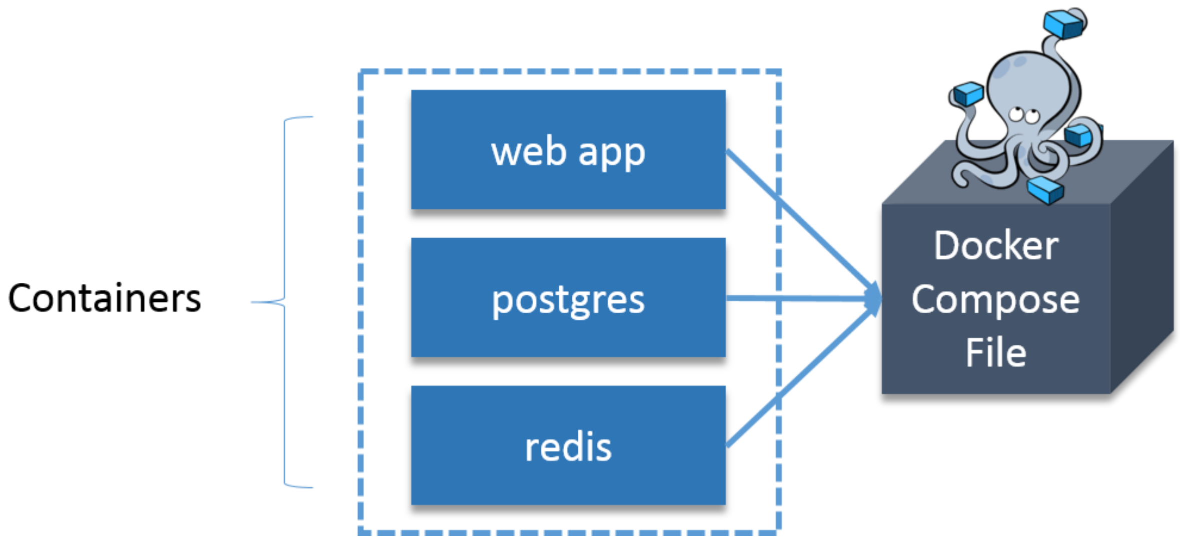
Docker Compose: It is a tool for defining and running multi-container Docker applications. With Docker Compose, you can use a Compose file to configure your application's services. Then, using a single command, you can create and start all the services from your configuration.

Suppose you have multiple applications in various containers and all those containers are linked together. So, you don't want to execute each of those containers one by one. But, you want to run those containers with a single command. That's where Docker Compose comes in to the picture. With it you can run multiple applications in various containers with a single command. i.e. docker-compose up.

Example: Imagine you have different containers, one running a web app, another running a postgres and another running redis, in a YAML file. That is called docker compose file, from there you can run these containers with a single command.



Suppose you want to publish a blog, for that you will use CMS (Content Management System), and wordpress is the most widely used CMS. Basically, you need one container for WordPress and you need one more container as MySQL for back end, that MySQL container should be linked to the wordpress container. We also need one more container for Php Myadmin that will be linked to MySQL database, basically, it is used to access MySQL database.

How about I execute the above stated example practically.

Steps involved:

1. **Install Docker Compose:**
2. **Install WordPress:** We'll be using the official WordPress and MariaDB Docker images.
3. **Install MariaDB:** It is one of the most popular database servers in the world. It's made by the original developers of MySQL. MariaDB is developed as open source software and as a relational database it provides an SQL interface for accessing data.
4. **Install PhpMyAdmin:** It is a free software tool written in PHP, intended to handle the administration of MySQL over the Web.
5. **Create The WordPress Site:**

Install Docker Compose:

Install Python Pip first:

```
sudo apt-get install python-pip
```

Install Docker Compose:

```
sudo pip install docker-compose
```

Install WordPress:

Create a wordpress directory:

```
mkdir wordpress
```

Enter this wordpress directory:

```
cd wordpress/
```

In this directory create a Docker Compose YAML file, then edit it using gedit:

```
sudo gedit docker-compose.yml
```

Paste the below lines of code in that yaml file:

```
wordpress:

image: wordpress

links:

- wordpress_db:mysql

ports:

- 8080:80

wordpress_db:

image: mariadb

environment:

MYSQL_ROOT_PASSWORD: XXXXX

phpmyadmin:

image: corbinu/docker-phpmyadmin

links:

- wordpress_db:mysql

ports:

```

```
- 8181:80

environment:

MYSQL_USERNAME: root

MYSQL_ROOT_PASSWORD:XXXXX
```

```
wordpress_db:
...
 environment:
    MYSQL_ROOT_PASSWORD: XXXX
...
```

This will set an environment variable inside the wordpress_db container called MYSQL_ROOT_PASSWORD with your desired password. The MariaDB Docker image is configured to check for this environment variable when it starts up and will take care of setting up the DB with a root account with the password defined as MYSQL_ROOT_PASSWORD.

```
wordpress:
...
 ports:
    - 8080:80
...
```

The first port number is the port number on the host, and the second port number is the port inside the container. So, this configuration forwards requests on port 8080 of the host to the default web server port 80 inside the container.

```
phpmyadmin:
  image: corbinu/docker-phpmyadmin
  links:
    - wordpress_db:mysql
  ports:
    - 8181:80
  environment:
    MYSQL_USERNAME: root
    MYSQL_ROOT_PASSWORD: XXXX
```

This grabs docker-phpmyadmin by community member corbinu, links it to our wordpress_db container with the name mysql (meaning from inside the phpmyadmin container references to the hostname mysql will be forwarded to our wordpress_db container), exposes its port 80 on port 8181 of the host system, and finally sets a couple of environment variables with our MariaDB username and password. This image does not automatically grab the MYSQL_ROOT_PASSWORD environment variable from the wordpress_dbcontainer's environment, the way the wordpress image does. We actually have to copy the MYSQL_ROOT_PASSWORD: XXXX line from the wordpress_db container, and set the username to root.

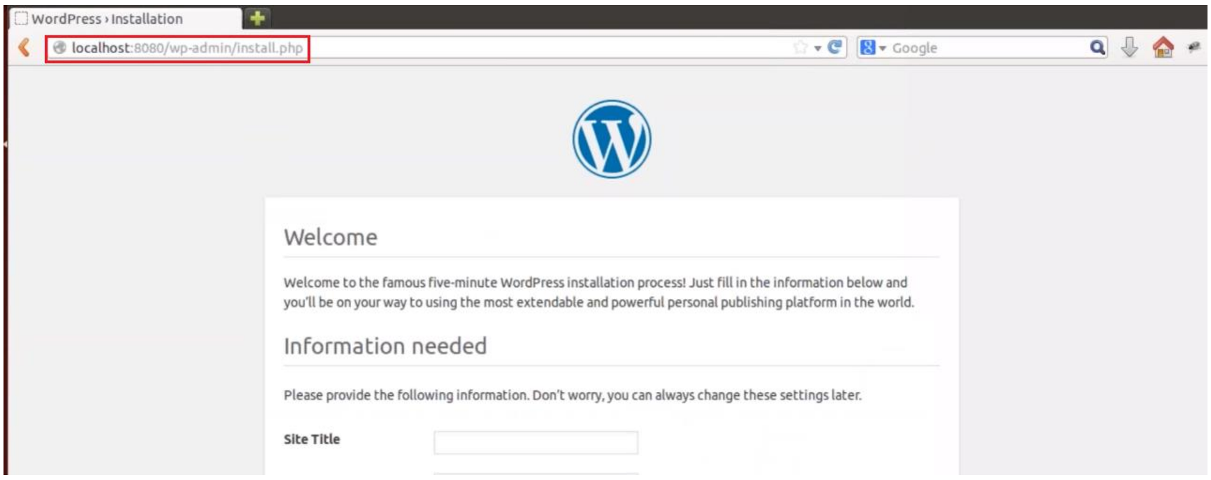
Now start the application group:

```
docker-compose up -d
```

That's all you have to do. You can add as many containers as you like this way, and link them all up in any way you please.

Now, in the browser go to port 8080, using your public IP or host name, as shown below:

```
localhost:8080
```



Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title