

Docker Compose

Docker Compose é uma ferramenta escrita em Python para definir e rodar aplicações multi-contêineres Docker. Através do Docker Compose, você define a configuração dos serviços da sua aplicação. Em um arquivo `yml`, é descrita a configuração de cada um dos serviços, bem como o relacionamento entre eles. Através de um comando, você pode criar, atualizar ou remover todos os serviços.

Arquivo YML

Para este exemplo, criaremos uma aplicação blog composta de 2 serviços:

- Servidor web: `ghost` ([Ghost](#))
- Banco de dados: `mariadb` ([MariaDB Foundation](#))

Dica: nos arquivos no formato `yml` a formatação é um aspecto importante da validação de sintaxe dele, formate-o de maneira padronizada (quantidade de espaços ou de tabs). Um espaço a mais antes de determinada definição irá causar erro de sintaxe no arquivo. Preste bastante atenção quanto copiar e colar os exemplos.

Crie um diretório chamado `blog` para armazenar a configuração:

```
# acessando o home do usuário
cd ~

# criando o diretório
mkdir blog

# entrando no diretório
cd blog
```

Crie um arquivo chamado `docker-compose.yml` com o seguinte conteúdo:

```
version: '3.0'
services:
  ghost:
    image: ghost:5.87.0
    ports:
```

```
- 2368:2368
restart: always
environment:
  database__client: mysql
  database__connection__host: "ghost-db"
  database__connection__user: "ghostuser"
  database__connection__password: "pass"
  database__connection__database: "ghostdb"
links:
  - ghost-db:ghost-db
depends_on:
  - ghost-db
volumes:
  - ~/blog/ghost:/var/lib/ghost/content
ghost-db:
  image: mariadb:10.3
  ports:
    - 3306:3306
  restart: always
  environment:
    MYSQL_DATABASE: "ghostdb"
    MYSQL_PASSWORD: "pass"
    MYSQL_RANDOM_ROOT_PASSWORD: 'true'
    MYSQL_USER: "ghostuser"
  volumes:
    - ~/blog/ghost-db:/var/lib/mysql
```

Detalhando o arquivo docker-compose.yml

- **version:** versão do docker-compose
- **services:** descrição dos serviços
 - ghost
 - ghost-db
- **image:** imagem utilizada pelo serviço
- **ports:** porta em que o serviço executará (lado esquerdo: host, lado direito: contêiner)
- **restart:** configuração de reinicialização do serviço
- **environment:** variáveis de ambiente dentro do contêiner
- **links:** link um contêiner em um outro serviço através do apelido fornecido
- **depends_on:** monta uma lista de dependências, para que o docker organize qual a ordem desejada de criação dos serviços (Exemplo: cria o banco de dados primeiro, depois

cria o servidor web que precisará se conectar no banco)

- **volumes:** o mapeamento da pasta no host e a respectiva pasta dentro do contêiner (lado esquerdo: host, lado direito: contêiner)

Note que nas configurações do tipo `environment` os mesmos valores passados no serviço de banco de dados são passados no serviço do blog. Você pode customizar esses valores, no entanto não se esqueça de atualizar nos dois serviços.

Quando subir um serviço em produção, não se esquecer de alterar os `valores padrão` das variáveis, especialmente as `credenciais` de acesso (nome de banco, nome do usuário, senha, etc).

Comandos básicos

Para os exemplos abaixo, os comandos foram executados no diretório em que se encontra o arquivo `docker-compose.yml`:

Dica: em algumas instalações mais recentes o comando `docker-compose` pode ser acessado como `docker compose` (note o espaço entre as palavras). Os comandos a seguir deste treinamento serão efetuados sempre como `docker-compose`, se for o caso, faça a respectiva substituição pelo comando com espaço na sua área de desenvolvimento.

Criando a aplicação

Criar a aplicação descrita no arquivo `docker-compose.yml`, a tela exibirá o log:

```
docker-compose up
```

O serviço de banco iniciará, depois que ele estiver disponível a aplicação web irá se conectar no banco e fará a inicialização das tabelas padrão do sistema. Depois de alguns segundos você visualizará os logs cessarem com a rolagem e poderá ver algo como:

```
blog-ghost-1 | [2024-07-03 13:46:47] INFO Ghost booted in 9.523s
blog-ghost-1 | [2024-07-03 13:46:47] INFO Adding offloaded job to the queue
blog-ghost-1 | [2024-07-03 13:46:47] INFO Scheduling job update-check at 1 24 6 * * *. Next run on: Thu Jul 04 2024 06:24:01 GMT+0000 (Coordinated Universal Time)
blog-ghost-1 | [2024-07-03 13:46:47] INFO Running milestone emails job on Fri Jul 05 2024 13:46:47 GMT+0000 (Coordinated Universal Time)
```

Note que o terminal ficará travado neste execução. Os logs dos serviços da stack será exibidos no terminal. Para finalizar, utilize o atalho `CONTROL + C`. A execução da stack será finalizada.

Saída:

```
Gracefully stopping... (press Ctrl+C again to force)
Aborting on container exit...
[+] Stopping 2/2
✓ Container blog-ghost-1    Stopped          0.5s
✓ Container blog-ghost-db-1 Stopped          2.5s
canceled
```

Vamos criar a aplicação novamente, desta vez com a sua execução em background (**-d** Detached mode):

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

Saída:

```
[+] Building 0.0s (0/0)
[+] Running 2/2
✓ Container blog-ghost-db-1 Started          0.4s
✓ Container blog-ghost-1    Started
```

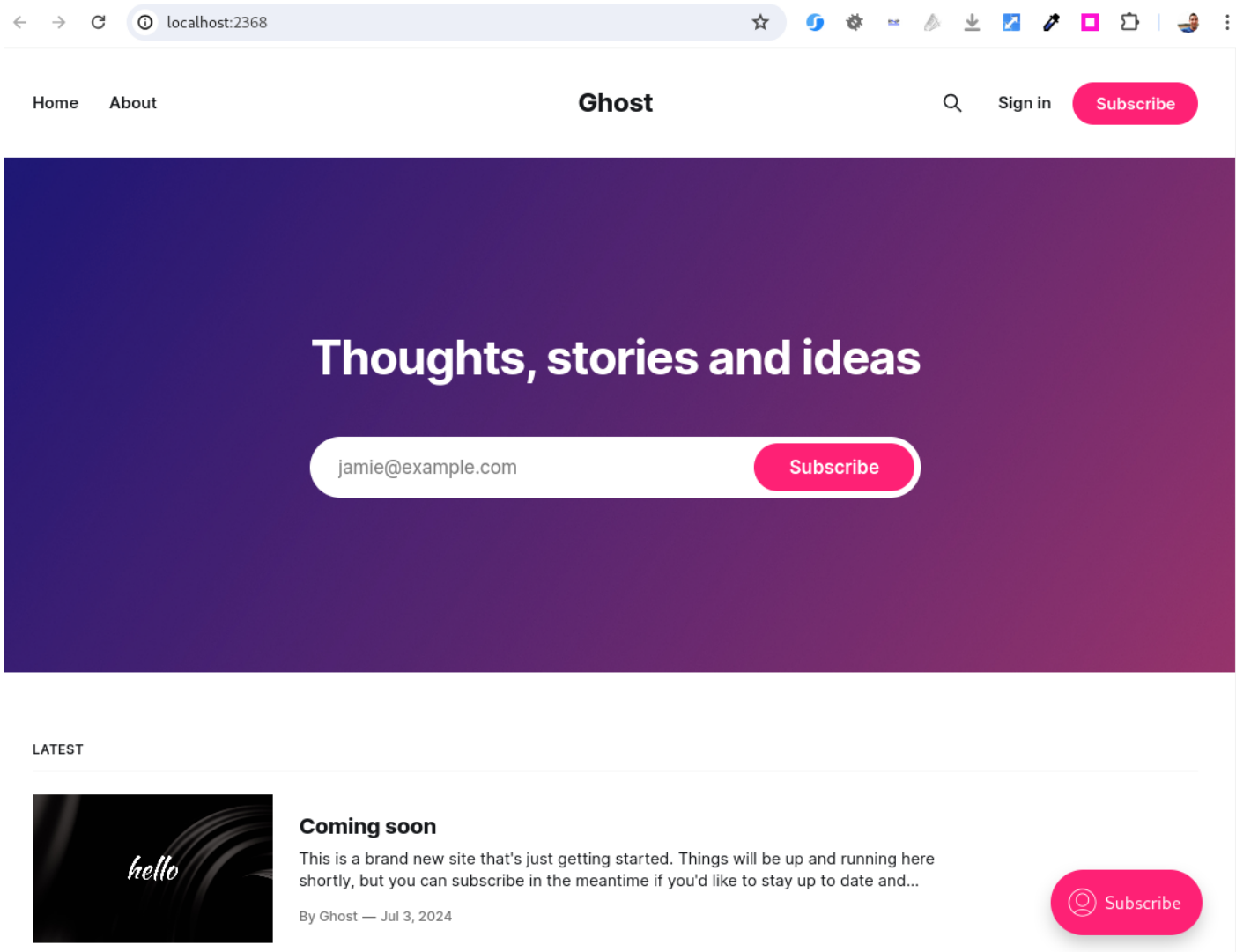
Conferir se a stack está rodando sem problemas:

```
docker-compose ps -a
```

Saída:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
NAMES					
f67c51906df4	ghost:5.87.0	"docker-entrypoint.s..."	5 minutes ago	Up 54 seconds	0.0.0.0:2368->2368/tcp, :::2368->2368/tcp
blog-ghost-1					
7c87994584ea	mariadb:10.3	"docker-entrypoint.s..."	5 minutes ago	Up 54 seconds	0.0.0.0:3306->3306/tcp, :::3306->3306/tcp
blog-ghost-db-1					

- O servidor web pode ser acessado no navegador no seguinte endereço:
<http://localhost:2368>
- O servidor de banco de dados está acessível na porta `3306`



Acessando o serviço de blog hospedado localmente via docker

Visualizar os logs:

```
docker-compose logs
```

Visualizar os logs e travar e tela na exibição neles:

```
docker-compose logs -f
```

Removendo a aplicação

Remover os contêiners da aplicação:

```
docker-compose down
```

Saída:

[+] Running 3/3

- ✓ Container blog-ghost-1 Removed 0.5s
- ✓ Container blog-ghost-db-1 Removed 1.9s
- ✓ Network blog_default Removed

Note que nós fizemos o mapeamento de `volumes` tanto do servidor de banco de dados como da aplicação web do blog ghost, portanto mesmo que você remova a aplicação, quando você criá-la novamente as alterações realizadas serão persistidas.

Conferir os arquivos persistidos no host:

```
ls -la ~/blog
```

Saída:

```
total 20
drwxr-xr-x 4 evertonpaiva evertonpaiva 4096 jul 3 10:46 .
drwxr-xr-x 80 evertonpaiva evertonpaiva 4096 jul 4 14:01 ..
-rw-r--r-- 1 evertonpaiva evertonpaiva 750 jul 3 10:46 docker-compose.yml
drwxr-xr-x 11 evertonpaiva root          4096 jul 3 10:46 ghost
drwxr-xr-x 5      999 systemd-journal 4096 jul 3 11:08 ghost-db
```

- **ghost**: pasta local para os arquivos do servidor web ghost
- **ghost-db**: pasta local para os arquivos do servidor de banco maria-db

Depois de realizarmos nossos teste, caso você deseja remover os arquivos utilizados:

```
sudo rm -rf ~/blog
```

Quando você acessa imagens docker no [Docker Hub](#), geralmente você encontrará informações sobre os principais parâmetros para configuração do serviço, as portas padrão que o serviço é executa e dicamos de como realizar a persistência ou seja, quais pastas deverão ser mapeadas para volumes para não perder os dados quando o contêiner for encerrado.

Ghost

Ghost is a free and open source blogging platform written in JavaScript and distributed under the MIT License, designed to simplify the process of online publishing for individual bloggers as well as online publications.

[wikipedia.org/wiki/Ghost_\(blogging_platform\)](https://wikipedia.org/wiki/Ghost_(blogging_platform))



How to use this image

This will start a Ghost development instance listening on the default Ghost port of 2368.

```
$ docker run -d --name some-ghost -e NODE_ENV=development ghost
```

Custom port

If you'd like to be able to access the instance from the host without the container's IP, standard port mappings can be used:

```
$ docker run -d --name some-ghost -e NODE_ENV=development -e url=http://localhost:3001 -p 3001:
```

If all goes well, you'll be able to access your new site on `http://localhost:3001` and `http://localhost:3001/ghost` to access Ghost Admin (or `http://host-ip:3001` and `http://host-ip:3001/ghost`, respectively).

Página do serviço ghost no Docker Hub como informações de configuração do serviço.

Referências

<https://docs.docker.com/compose/>

Revision #17

Created 2 July 2024 19:48:01 by Éverton de Oliveira Paiva

Updated 5 July 2024 13:01:37 by Éverton de Oliveira Paiva