docker-compose.yml 语法说明

YAML 模板文件语法

> 默认的模板文件是 docker-compose.yml，其中定义的每个服务都必须通过 image 指令指定镜像或 build 指令（需要 Dockerfile）来自动构建。
> 其它大部分指令都跟 docker run 中的类似。
> 如果使用 build 指令，在 Dockerfile 中设置的选项(例如：CMD, EXPOSE, VOLUME, ENV 等) 将会自动被获取，无需在 docker-compose.yml 中再次设置。
> image
> 指定为镜像名称或镜像 ID。如果镜像在本地不存在，Compose 将会尝试拉去这个镜像。
> 例如：

```
image: ubuntu
image: orchardup/postgresql
image: a4bc65fd
```

build

指定 Dockerfile 所在文件夹的路径。 Compose 将会利用它自动构建这个镜像，然后使用这个镜像。
build: /path/to/build/dir

command

覆盖容器启动后默认执行的命令。
command: bundle exec thin -p 3000

links

链接到其它服务中的容器。使用服务名称（同时作为别名）或服务名称：服务别名 （SERVICE:ALIAS） 格式都可以。
links:
- db
- db:database
- redis
使用的别名将会自动在服务容器中的 /etc/hosts 里创建。例如：
172.17.2.186 db
相应的环境变量也将被创建。

external_links

链接到 docker-compose.yml 外部的容器，甚至 并非 Compose 管理的容器。参数格式跟 links 类似。
external_links:
- redis_1
- project_db_1:mysql
- project_db_1:postgresql

ports

暴露端口信息。
使用宿主：容器 （HOST:CONTAINER）格式或者仅仅指定容器的端口（宿主将会随机选择端口）都可以。

ports:

- "3000"
- "8000:8000"
- "127.0.0.1:8001:8001"
注：当使用 HOST:CONTAINER 格式来映射端口时，如果你使用的容器端口小于 60 你可能会得到错误得结果，因为 YAML 将会解析 xx:yy 这种数字格式为 60 进制。所以建议采用字符串格式。

expose

暴露端口，但不映射到宿主机，只被连接的服务访问。
仅可以指定内部端口为参数

```
expose:
- "3000"
- "8000"
```

volumes

卷挂载路径设置。可以设置宿主机路径 （HOST:CONTAINER） 或加上访问模式 （HOST:CONTAINER:ro）。

```
volumes:
- /var/lib/mysql
- cache/:/tmp/cache
- ~/configs:/etc/configs/:ro
```

volumes_from

从另一个服务或容器挂载它的所有卷。

```
volumes_from:
- service_name
- container_name
```

environment

设置环境变量。你可以使用数组或字典两种格式。
只给定名称的变量会自动获取它在 Compose 主机上的值，可以用来防止泄露不必要的数据。

```
environment:
- RACK_ENV=development
- SESSION_SECRET
```

env_file

从文件中获取环境变量，可以为单独的文件路径或列表。
如果通过 docker-compose -f FILE 指定了模板文件，则 env_file 中路径会基于模板文件路径。
如果有变量名称与 environment 指令冲突，则以后者为准。

```
env_file: .env
env_file:
- ./common.env
- ./apps/web.env
- /opt/secrets.env
```

环境变量文件中每一行必须符合格式，支持 # 开头的注释行。

```
# common.env: Set Rails/Rack environment
RACK_ENV=development
```

extends

基于已有的服务进行扩展。例如我们已经有了一个 webapp 服务，模板文件为 common.yml。

```
# common.yml
webapp:
build: ./webapp
environment:
\ - DEBUG=false
\ - SEND_EMAILS=false
```

编写一个新的 development.yml 文件，使用 common.yml 中的 webapp 服务进行扩展。

development.yml

```
web:
extends:
file: common.yml
service: webapp
ports:
\ - "8000:8000"
links:
\ - db
```

environment:
- DEBUG=true
db:
image: postgres
后者会自动继承 common.yml 中的 webapp 服务及相关环节变量。

net

设置网络模式。使用和 docker client 的 --net 参数一样的值。
net: "bridge"
net: "none"
net: "container:[name or id]"
net: "host"

pid

跟主机系统共享进程命名空间。打开该选项的容器可以相互通过进程 ID 来访问和操作。
pid: "host"

dns

配置 DNS 服务器。可以是一个值，也可以是一个列表。
dns: 8.8.8.8
dns:
- 8.8.8.8
- 9.9.9.9

cap_add, cap_drop

添加或放弃容器的 Linux 能力（Capabiliity）。
cap_add:
- ALL
cap_drop:
- NET_ADMIN
- SYS_ADMIN

dns_search

配置 DNS 搜索域。可以是一个值，也可以是一个列表。
dns_search: example.com
dns_search:
- domain1.example.com
\ - domain2.example.com
working_dir, entrypoint, user, hostname, domainname, mem_limit, privileged, restart, stdin_open, tty, cpu_shares
这些都是和 docker run 支持的选项类似。
cpu_shares: 73
working_dir: /code
entrypoint: /code/entrypoint.sh
user: postgresql
hostname: foo
domainname: foo.com
mem_limit: 1000000000
privileged: true
restart: always
stdin_open: true
tty: true

分类: docker

标签: docker-compose.yml

« 上一篇：docker 镜像和容器的批量清理
» 下一篇：docker-compose 工具安装

posted @ 2016-03-23 15:20 freefei 阅读(5953) 评论(0) 编辑 收藏