基于Docker的微服务自动化测试

第二天：Docker微服务测试构建篇

> 核心内容 Dockerfile

主要内容

- 通过Dockerfile动手构建自己的镜像
- Dockerfile基本语法和优化方案
- API接口从开发、测试、部署的全过程
- 用pytest对Dockerfile进行冒烟测试

容器是怎么来的?

- 镜像启动而来

镜像是怎么怎么来的?

- 仓库拉取下拉

问题来了:

> 需要某个镜像，但是仓库里没有，怎么办?

自己构建镜像

API接口项目:

1. 接口开发
2. 测试
3. 打包镜像
4. 上线部署

一、 接口开发

1. 创建项目
2. 列出依赖
    - fastapi （框架）

- uvicorn （web server）

3. 编写代码

    1. get接口

```python
@app.get("/")  # GET  /
def index():
    return {"Hello": "World"}


@app.get("/add_by_get")  # GET /add_by_get?a=123&b=456
def add_get(a: int, b: int):
    return {"c": add(a, b)}
```

    2. post

```python
class Itmen(BaseModel):
    a: int
    b: int
    c: int = None  # 可选参数

    def add(self):
        self.c = add(self.a, self.b)


@app.post("/add_by_post")  # POST请求
def add_post(item: Itmen):  # 接口参数
    item.add()
    return item  # 接口的返回值
```

4. 启动项目

```
uvicorn main:app
```

接口地址：http://127.0.0.1:8000/

接口文档：http://127.0.0.1:8000/docs

二、接口测试

第一件事情做申明?

1. 构建client

```python
from fastapi.testclient import TestClient

from main import app

client = TestClient(app)

```

2. 编写用例

```python

def test_add_by_get():
    data = {
        "a": 1111,
        "b": 2222,
    }
    expect_data = {"c": 3333}

    resp = client.get("/add_by_get", params=data)
    assert resp.status_code == 200
    assert resp.json() == expect_data


def test_add_by_get_error():
    data = {
        "a": 1111,
    }

    resp = client.get("/add_by_get", params=data)
    assert resp.status_code == 422
```

三、构建镜像

1. 编写Dockerfile

Dockerfile是文本文件，用来描述镜像构建步骤，使用`docker build`完成真正构建

指令	作用	
FROM	指定基础镜像	
LABEL	添加标签	
RUN	构建时执行命令	
CMD	运行时执行命令	
EXPOSE	申明监听的端口	
ENV	设置环境变量	
ADD	添加（远程）文件到容器	
COPY	复制文件到容器	
ENTRYPOINT	入口点	
VOLUME	申明挂载卷的路径	
USER	指定容器的执行设法你	
WORKDIR	指定容器工作目录	
ONBUILD	作为基础镜像时执行的命令	

2. 编译、构建

`docer build`命令

```
1 docker build -t beifan_api:v1 .
```

-t 给镜像打标签

. 在当前目录寻找Dockerfile文件

3. 运行

`docker run` 命令

```
1 docker run --name my_api --rm -p -d 8080:80  beifan_api:v1
```

```
1 http://ubuntu.local:8080/add_by_get?a=121&b=200
```

四、进行冒烟测试

使用**pytets**自动化的进行测试

1. 列出依赖

- pytest
- docker

2. 定制fixture

1. 让测试用例能有容器可测
2. 让测试用例知道容器ip

3. 冒泡测试

4.经验：

容器就绪，不代表服务（程序启动、数据库启动、程序初始化。。。。。。）就绪

五、部署

部署可用自动化：借助私有仓库、CI/CD

也可以手动

1. 导出镜像

2. 上传

3. 导入镜像

4. 启动容器