



下载APP

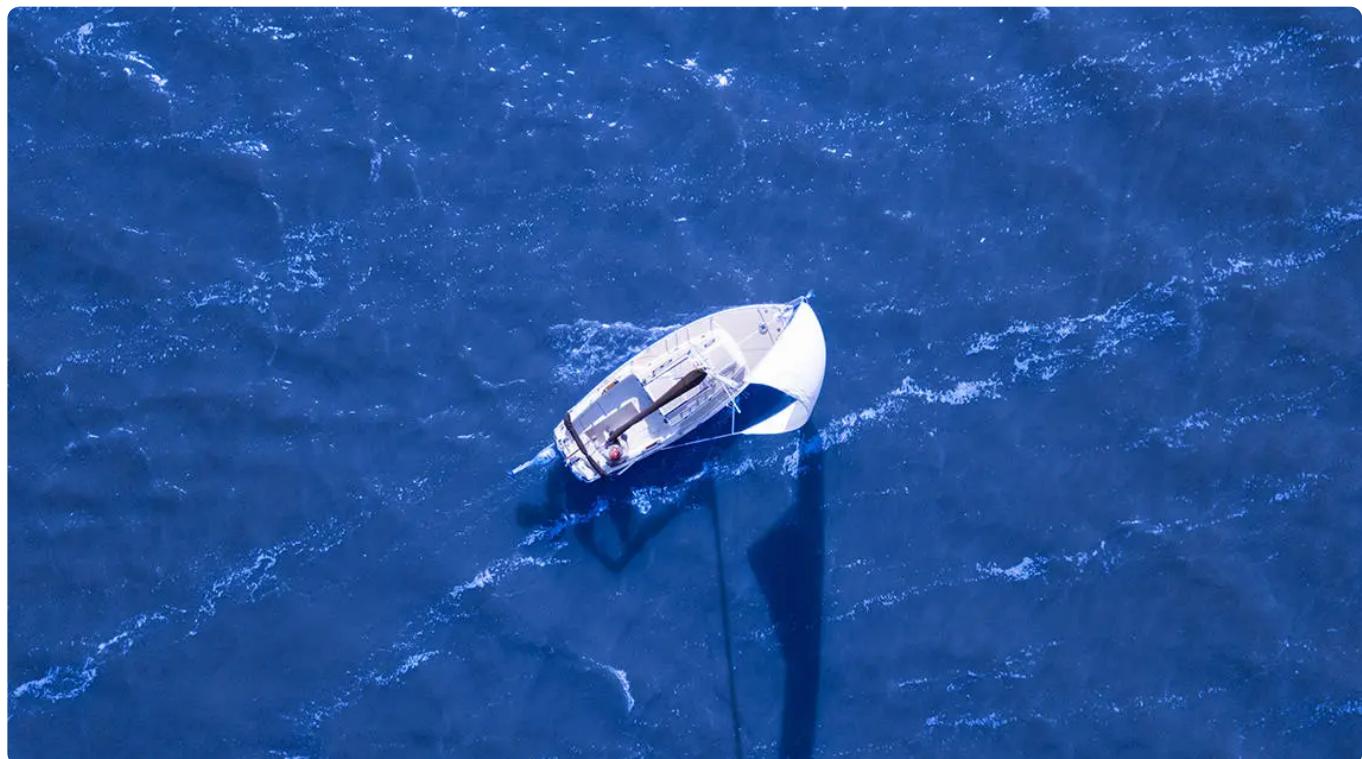


# 加餐 | docker-compose : 单机环境下的容器编排工具

2022-08-15 Chrono 来自北京

《Kubernetes入门实战课》

课程介绍 >



讲述 : Chrono

时长 11:07 大小 10.20M



你好，我是 Chrono。

我们的课程学到了这里，你已经对 Kubernetes 有相当程度的了解了吧。

作为云原生时代的操作系统，Kubernetes 源自 Docker 又超越了 Docker，依靠着它的 master/node 架构，掌控成百上千台的计算节点，然后使用 YAML 语言定义各种 API 对象来编排调度容器，实现了对现代应用的管理。

不过，你有没有觉得，在 Docker 和 Kubernetes 之间，是否还缺了一点什么东西呢？



Kubernetes 的确是非常强大的容器编排平台，但强大的功能也伴随着复杂度和成本的提升，不说那几十个用途各异的 API 对象，单单说把 Kubernetes 运行起来搭建一个小型的



下载APP



运行成本。

显然，在这种简易任务的应用场景里，Kubernetes 就显得有些“笨重”了。即使是“玩具”性质的 minikube、kind，对电脑也有比较高的要求，会“吃”掉不少的计算资源，属于“大材小用”。

那到底有没有这样的工具，既像 Docker 一样轻巧易用，又像 Kubernetes 一样具备容器编排能力呢？

今天我就来介绍 docker-compose，它恰好满足了刚才的需求，是一个在单机环境里轻量级的容器编排工具，填补了 Docker 和 Kubernetes 之间的空白位置。

## 什么是 docker-compose

还是让我们从 Docker 诞生那时讲起。

在 Docker 把容器技术大众化之后，Docker 周边涌现出了数不胜数的扩展、增强产品，其中有一个名字叫“Fig”的小项目格外令人瞩目。

Fig 为 Docker 引入了“容器编排”的概念，使用 YAML 来定义容器的启动参数、先后顺序和依赖关系，让用户不再有 Docker 冗长命令行的烦恼，第一次见识到了“声明式”的威力。

Docker 公司也很快意识到了 Fig 这个小工具的价值，于是就在 2014 年 7 月把它买了下来，集成进 Docker 内部，然后改名成了“docker-compose”。





下载APP



图片来自网络

从这段简短的历史中你可以看到，虽然 docker-compose 也是容器编排技术，也使用 YAML，但它的基因与 Kubernetes 完全不同，走的是 Docker 的技术路线，所以在设计理念和使用方法上有差异就不足为怪了。

docker-compose 自身的定位是管理和运行多个 Docker 容器的工具，很显然，它没有 Kubernetes 那么“宏伟”的目标，只是用来方便用户使用 Docker 而已，所以学习难度比较低，上手容易，很多概念都是与 Docker 命令一一对应的。

但这有时候也会给我们带来困扰，毕竟 docker-compose 和 Kubernetes 同属容器编排领域，用法不一致就容易导致认知冲突、混乱。考虑到这一点，我们在学习 docker-compose 的时候就要把握一个“度”，够用就行，不要太过深究，否则会对 Kubernetes 的学习造成一些不良影响。

## 如何使用 docker-compose

docker-compose 的安装非常简单，它在 GitHub (<https://github.com/docker/compose>) 上提供了多种形式的二进制可执行文件，支持 Windows、macOS、Linux 等操作系统，也支持 x86\_64、arm64 等硬件架构，可以直接下载。



在 Linux 上安装的 Shell 命令我放在这里了，用的是最新的 2.6.1 版本：

复制代码



下载APP



```

5  # apple m1
6  sudo curl -SL https://github.com/docker/compose/releases/download/v2.6.1/docke
7      -o /usr/local/bin/docker-compose
8
9
10 sudo chmod +x /usr/local/bin/docker-compose
    sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

```

安装完成之后，来看一下它的版本号，命令是 `docker-compose version`，用法和 `docker version` 是一样的：

复制代码

```
1 docker-compose version
```

[chrono@K8S ~]\$`docker-compose version`  
Docker Compose version v2.6.1

接下来，我们就要编写 YAML 文件，来管理 Docker 容器了，先用 [第 7 讲](#)里的私有镜像仓库作为示范吧。

`docker-compose` 里管理容器的核心概念是 “**service**” 。注意，它与 Kubernetes 里的 `Service` 虽然名字很像，但却是完全不同的东西。`docker-compose` 里的 “`service`” 就是一个容器化的应用程序，通常是一个后台服务，用 YAML 定义这些容器的参数和相互之间的关系。

如果硬要和 Kubernetes 对比的话，和 “`service`” 最像的 API 对象应该算是 Pod 里的 `container` 了，同样是管理容器运行，但 `docker-compose` 的 “`service`” 又融合了一些 `Service`、`Deployment` 的特性。

下面的这个就是私有镜像仓库 Registry 的 YAML 文件，关键字段就是 “**services**” ， 的 Docker 命令我也列了出来：

复制代码



下载APP



复制代码

```

1 services:
2   registry:
3     image: registry
4     container_name: registry
5     restart: always
6     ports:
7       - 5000:5000

```

把它和 Kubernetes 对比一下，你会发现它和 Pod 定义非常像，“services” 相当于 Pod，而里面的“service”就相当于“spec.containers”：

复制代码

```

1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: nginx-pod
5 spec:
6   restartPolicy: Always
7   containers:
8     - image: nginx:alpine
9       name: nginx
10      ports:
11        - containerPort: 80

```

比如用 `image` 声明镜像，用 `ports` 声明端口，很容易理解，只是在用法上有些不一样，像端口映射用的就还是 Docker 的语法。

由于 docker-compose 的字段定义在官网

( <https://docs.docker.com/compose/compose-file/> ) 上有详细的说明文档，我就不在这里费口舌解释了，你可以自行参考。

需要提醒的是，在 docker-compose 里，每个“service”都有一个自己的名字，它也是这个容器的唯一网络标识，有点类似 Kubernetes 里 Service 域名的作用。



下载APP



复制代码

```
1 docker-compose -f reg-compose.yml up -d
```

```
[chrono@K8S ~]$docker-compose -f reg-compose.yml up -d
[+] Running 2/2
  :: Network chrono_default Created
  :: Container registry Started
```

因为 docker-compose 在底层还是调用的 Docker，所以它启动的容器用 docker ps 也能够看到：

```
[chrono@K8S ~]$docker ps
CONTAINER ID IMAGE COMMAND CREATED
15d619236b86 registry "/entrypoint.sh /etc..." About a minute ago
```

不过，我们用 docker-compose ps 能够看到更多的信息：

```
1 docker-compose -f reg-compose.yml ps
```

复制代码

```
[chrono@K8S ~]$docker-compose -f reg-compose.yml ps
NAME COMMAND SERVICE STATUS
registry "/entrypoint.sh /etc..." registry running
```

下面我们把 Nginx 的镜像改个标签，上传到私有仓库里测试一下：

```
1 docker tag nginx:alpine 127.0.0.1:5000/nginx:v1
2 docker push 127.0.0.1:5000/nginx:v1
```

复制代码



再用 curl 查看一下它的标签列表，就可以看到确实上传成功了：



下载APP



```
[chrono@K8S ~]$curl 127.1:5000/v2/nginx/tags/list
{"name":"nginx","tags":["v1"]}
```

想要停止应用，我们需要使用 `docker-compose down` 命令：

复制代码

```
1 docker-compose -f reg-compose.yml down
```

```
[chrono@K8S ~]$docker-compose -f reg-compose.yml down
[+] Running 2/2
  :: Container registry          Removed
  :: Network chrono_default      Removed
```

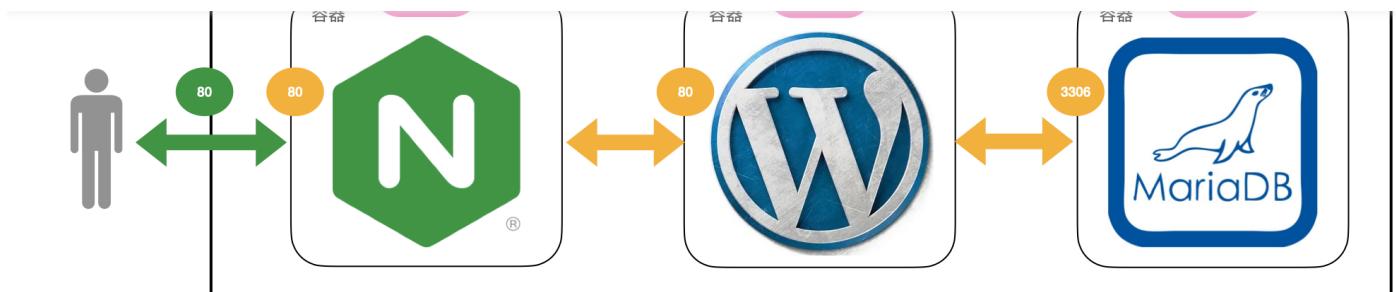
通过这个小例子，我们就成功地把“命令式”的 Docker 操作，转换成了“声明式”的 `docker-compose` 操作，用法与 Kubernetes 十分接近，同时还没有 Kubernetes 那些昂贵的运行成本，在单机环境里可以说是最适合不过了。

## 使用 `docker-compose` 搭建 WordPress 网站

不过，私有镜像仓库 Registry 里只有一个容器，不能体现 `docker-compose` 容器编排的好处，我们再用它来搭建一次 WordPress 网站，深入感受一下。

架构图和第 7 讲还是一样的：



[下载APP](#)

第一步还是定义数据库 MariaDB，环境变量的写法与 Kubernetes 的 ConfigMap 有点类似，但使用的字段是 `environment`，直接定义，不用再“绕一下”：

[复制代码](#)

```

1 services:
2
3   mariadb:
4     image: mariadb:10
5     container_name: mariadb
6     restart: always
7
8     environment:
9       MARIADB_DATABASE: db
10      MARIADB_USER: wp
11      MARIADB_PASSWORD: 123
12      MARIADB_ROOT_PASSWORD: 123

```

我们可以再对比第 7 讲里启动 MariaDB 的 Docker 命令，可以发现 docker-compose 的 YAML 和命令行是非常像的，几乎可以直接照搬：

[复制代码](#)

```

1 docker run -d --rm \
2   --env MARIADB_DATABASE=db \
3   --env MARIADB_USER=wp \
4   --env MARIADB_PASSWORD=123 \
5   --env MARIADB_ROOT_PASSWORD=123 \
6   mariadb:10

```



第二步是定义 WordPress 网站，它也使用 `environment` 来设置环境变量：



下载APP



```

3
4   wordpress:
5     image: wordpress:5
6     container_name: wordpress
7     restart: always
8
9   environment:
10    WORDPRESS_DB_HOST: mariadb #注意这里，数据库的网络标识
11    WORDPRESS_DB_USER: wp
12    WORDPRESS_DB_PASSWORD: 123
13    WORDPRESS_DB_NAME: db
14
15   depends_on:
16     - mariadb

```

不过，因为 docker-compose 会自动把 MariaDB 的名字用做网络标识，所以在连接数据库的时候（字段 WORDPRESS\_DB\_HOST）就不需要手动指定 IP 地址了，直接用“service”的名字 mariadb 就行了。这是 docker-compose 比 Docker 命令要方便的一个地方，和 Kubernetes 的域名机制很像。

WordPress 定义里还有一个**值得注意的是字段 depends\_on，它用来设置容器的依赖关系，指定容器启动的先后顺序**，这在编排由多个容器组成的应用的时候是一个非常便利的特性。

第三步就是定义 Nginx 反向代理了，不过很可惜，docker-compose 里没有 ConfigMap、Secret 这样的概念，要加载配置还是必须用外部文件，无法集成进 YAML。

Nginx 的配置文件和第 7 讲里也差不多，同样的，在 proxy\_pass 指令里不需要写 IP 地址了，直接用 WordPress 的名字就行：

复制代码



```

1 server {
2   listen 80;
3   default_type text/html;
4
5   location / {
6     proxy_http_version 1.1;
7     proxy_set_header Host $host;
8     proxy_pass http://wordpress; #注意这里，网站的网络标识

```



下载APP



然后我们就可以在 YAML 里定义 Nginx 了，加载配置文件用的是 `volumes` 字段，和 Kubernetes 一样，但里面的语法却又是 Docker 的形式：

复制代码

```
1 services:
2 ...
3
4 nginx:
5   image: nginx:alpine
6   container_name: nginx
7   hostname: nginx
8   restart: always
9   ports:
10    - 80:80
11   volumes:
12    - ./wp.conf:/etc/nginx/conf.d/default.conf
13
14 depends_on:
15  - wordpress
```

到这里三个 “service” 就都定义好了，我们用 `docker-compose up -d` 启动网站，记得还是要用 `-f` 参数指定 YAML 文件：

复制代码

```
1 docker-compose -f wp-compose.yml up -d
```

启动之后，用 `docker-compose ps` 来查看状态：





下载APP



```
⋮ Network chrono_default      Created
⋮ Volume "chrono_maria_data" Created
⋮ Container mariadb          Started
⋮ Container wordpress         Started
⋮ Container nginx            Started
[chrono@K8S ~]$
[chrono@K8S ~]$ docker-compose -f wp-compose.yml ps
NAME                  COMMAND           SERVICE
mariadb               "docker-entrypoint.s..." mariadb
nginx                "/docker-entrypoint...." nginx
wordpress             "docker-entrypoint.s..." wordpress
```

我们也可以用 `docker-compose exec` 来进入容器内部，验证一下这几个容器的网络标识是否工作正常：

复制代码

```
1 docker-compose -f wp-compose.yml exec -it nginx sh
```





下载APP



```
PING mariadb (172.22.0.2): 56 data bytes
64 bytes from 172.22.0.2: seq=0 ttl=64 time=0.247 ms
64 bytes from 172.22.0.2: seq=1 ttl=64 time=0.236 ms
^C
--- mariadb ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.236/0.241/0.247 ms
/ #
/ # ping wordpress
PING wordpress (172.22.0.3): 56 data bytes
64 bytes from 172.22.0.3: seq=0 ttl=64 time=0.547 ms
64 bytes from 172.22.0.3: seq=1 ttl=64 time=0.421 ms
^C
--- wordpress ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.421/0.484/0.547 ms
/ #
/ # exit
```

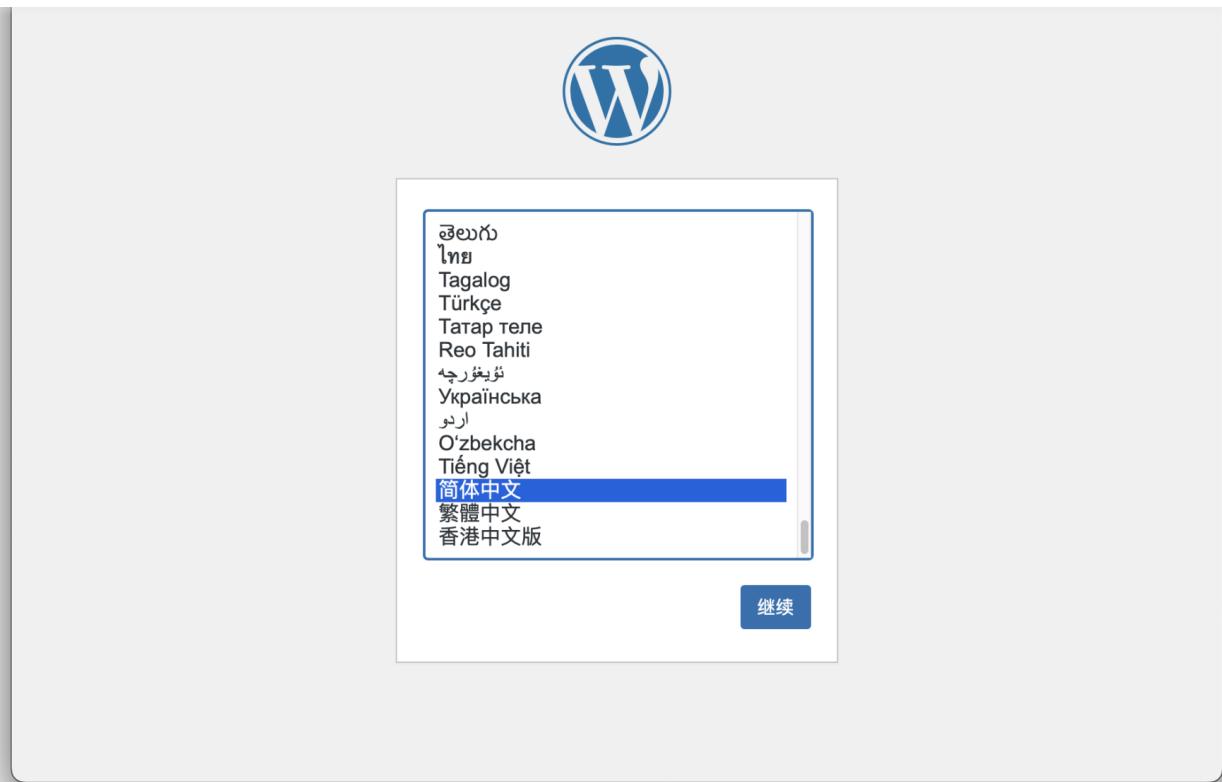
从截图里你可以看到，我们分别 ping 了 mariadb 和 wordpress 这两个服务，网络都是通的，不过它的 IP 地址段用的是 “172.22.0.0/16”，和 Docker 默认的 “172.17.0.0/16” 不一样。

再打开浏览器，输入本机的 “127.0.0.1” 或者是虚拟机的 IP 地址（我这里是 “<http://192.168.10.208>”），就又可以看到熟悉的 WordPress 界面了：





下载APP



## 小结

好了，今天我们暂时离开了 Kubernetes，回头看了一下 Docker 世界里的容器编排工具 docker-compose。

和 Kubernetes 比起来，docker-compose 有它自己的局限性，比如只能用于单机，编排功能比较简单，缺乏运维监控手段等等。但它也有优点：小巧轻便，对软硬件的要求不高，只要有 Docker 就能够运行。

所以虽然 Kubernetes 已经成为了容器编排领域的霸主，但 docker-compose 还是有一定的生存空间，像 GitHub 上就有很多项目提供了 docker-compose YAML 来快速搭建原型或者测试环境，其中的一个典型就是 CNCF Harbor。

对于我们日常工作来说，docker-compose 也是很有用的。如果是只有几个容器的简单用，用 Kubernetes 来运行实在是有种“杀鸡用牛刀”的感觉，而用 Docker 命令、Shell 脚本又很不方便，这就是 docker-compose 出场的时候了，它能够让我们彻底摆脱“命令式”，全面使用“声明式”来操作容器。



下载APP



1. docker-compose 源自 Fig，是专门用来编排 Docker 容器的工具。
2. docker-compose 也使用 YAML 来描述容器，但语法语义更接近 Docker 命令行。
3. docker-compose YAML 里的关键概念是 “service”，它是一个容器化的应用。
4. docker-compose 的命令与 Docker 类似，比较常用的有 up、ps、down，用来启动、查看和停止应用。

另外，docker-compose 里还有不少有用的功能，比如存储卷、自定义网络、特权进程等等，感兴趣的话可以再去看看官网资料。

欢迎留言交流你的学习想法，我们下节课回归正课，下节课见。





下载APP



## 课外小贴士

1. docker-compose (Fig) 最早是用 Python 编写的，2.0 版本后改用 Go 语言重新实现。1.x 的用法与 2.x 略有些不同，最后一个 Python 开发的版本是 1.29.2。
2. docker-compose 还可以安装成 docker 的插件，以子命令的形式使用，也就是“docker compose”（没有中间的横线）。
3. docker-compose 默认 YAML 文件的名字是“compose.yml”，所以如果把 YAML 改成这个名字，就可以省略“-f”参数。
4. 你也许在有的 docker-compose YAML 里看到文件开头有一个“version”字段，它标记了规范的版本，用来实现向后兼容，但现在它已经被废弃了，不建议再使用。
5. 在 Docker 最鼎盛的那个时期，docker-compose 和 Docker Machine、Docker Swarm 曾经被并称为“Docker 三剑客”，如今后两者都已经消失了。

分享给需要的人，Ta订阅超级会员，你最高得 **50** 元

Ta单独购买本课程，你将得 **20** 元



生成海报并分享

赞 2

提建议



下载APP

[上一篇 23 | 视频：中级篇实操总结](#)

## 实例详解

# 6 小时带你搞定 K8s 和容器架构设计原则

不仅让你会用，更让你懂原理



孟凡杰  
前 eBay 资深架构师  
云原生训练营主讲老师

免费报名

## 精选留言 (3)

写留言



Christopher

2022-08-15 来自北京

如果安装成docker compose plugin的形式，即没有中间的横线，harbor安装会有问题，因为检测不到docke-compose 😊

作者回复：所以一般还是用传统的docker-compose的形式，和1.x兼容。



1



Geek\_72ffc9

2022-08-15 来自北京



这个可以用在小型公司生产线上吗？

作者回复：当然可以，它其实就是一个易用性包装。



下载APP



YueShi

2022-08-15 来自北京

```
apt install docker-compose
```

为docker-compose version 1.29.2, build unknown

不是最新版的

作者回复: 从GitHub项目里直接下载二进制文件。

