



下载APP



46 | 如何制作Docker镜像？

2021-09-11 孔令飞

《Go 语言项目开发实战》

课程介绍 >



讲述：孔令飞

时长 15:12 大小 13.94M



你好，我是孔令飞。

要落地云原生架构，其中的一个核心点是通过容器来部署我们的应用。如果要使用容器来部署应用，那么制作应用的 Docker 镜像就是我们绕不开的关键一步。今天，我就来详细介绍下如何制作 Docker 镜像。

在这一讲中，我会先讲解下 Docker 镜像的构建原理和方式，然后介绍 Dockerfile 的指令，以及如何编写 Dockerfile 文件。最后，介绍下编写 Dockerfile 文件时要遵循的一些最佳实践。



Docker 镜像的构建原理和方式

首先，我们来看下 Docker 镜像构建的原理和方式。

我们可以用多种方式来构建一个 Docker 镜像，最常用的有两种：

通过 `docker commit` 命令，基于一个已存在的容器构建出镜像。

编写 Dockerfile 文件，并使用 `docker build` 命令来构建镜像。

上面这两种方法中，镜像构建的底层原理是相同的，都是通过下面 3 个步骤来构建镜像：

1. 基于原镜像，启动一个 Docker 容器。
2. 在容器中进行一些操作，例如执行命令、安装文件等。由这些操作产生的文件变更都会被记录在容器的存储层中。
3. 将容器存储层的变更 `commit` 到新的镜像层中，并添加到原镜像上。

下面，我们来具体讲解这两种构建 Docker 镜像的方式。

通过 `docker commit` 命令构建镜像

我们可以通过 `docker commit` 来构建一个镜像，命令的格式为 `docker commit [选项] [<仓库名>[:<标签>]]`。

下图中，我们通过 4 个步骤构建了 Docker 镜像

`ccr.ccs.tencentyun.com/marmotedu/iam-apiserver-amd64:test`：

```
[colin@dev iam]$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
48d1dbb89a7f   ccr.ccs.tencentyun.com/marmotedu/iam-apiserver-amd64:latest  "/opt/iam/bin/iam-ap...  5 minutes ago  Up 5 minutes
pensive_lichterman
[colin@dev iam]$ docker pause 48d1dbb89a7f
48d1dbb89a7f
[colin@dev iam]$ docker commit 48d1dbb89a7f ccr.ccs.tencentyun.com/marmotedu/iam-apiserver-amd64:test
sha256:f34935cad37e0e34453d35b06ece9d64268792f45f07d09a73af3e16ab9d8c64
[colin@dev iam]$ docker images ccr.ccs.tencentyun.com/marmotedu/iam-apiserver-amd64:test
REPOSITORY          TAG                 IMAGE ID            CREATED          SIZE
ccr.ccs.tencentyun.com/marmotedu/iam-apiserver-amd64   test              f34935cad37e      12 seconds ago  34.5MB
[colin@dev iam]$ docker unpause 48d1dbb89a7f
48d1dbb89a7f
```

具体步骤如下：

1. 执行 `docker ps` 获取需要构建镜像的容器 ID `48d1dbb89a7f`。
2. 执行 `docker pause 48d1dbb89a7f` 暂停 `48d1dbb89a7f` 容器的运行。

3. 执行`docker commit 48d1dbb89a7f`

`ccr.ccs.tencentyun.com/marmotedu/iam-apiserver-amd64:test`，基于容器 ID `48d1dbb89a7f`构建 Docker 镜像。

4. 执行`docker images ccr.ccs.tencentyun.com/marmotedu/iam-apiserver-amd64:test`，查看镜像是否成功构建。

这种镜像构建方式通常用在下面两个场景中：

构建临时的测试镜像；

容器被入侵后，使用`docker commit`，基于被入侵的容器构建镜像，从而保留现场，方便以后追溯。

除了这两种场景，我不建议你使用`docker commit`来构建生产现网环境的镜像。我这么说的主要原因有两个：

使用`docker commit`构建的镜像包含了编译构建、安装软件，以及程序运行产生的大量无用文件，这会导致镜像体积很大，非常臃肿。

使用`docker commit`构建的镜像会丢失掉所有对该镜像的操作历史，无法还原镜像的构建过程，不利于镜像的维护。

下面，我们再来看看如何使用Dockerfile来构建镜像。

通过Dockerfile来构建镜像

在实际开发中，使用Dockerfile来构建是最常用，也最标准的镜像构建方法。

Dockerfile是 Docker 用来构建镜像的文本文件，里面包含了一系列用来构建镜像的指令。

`docker build`命令会读取Dockerfile的内容，并将Dockerfile的内容发送给 Docker 引擎，最终 Docker 引擎会解析Dockerfile中的每一条指令，构建出需要的镜像。

`docker build`的命令格式为`docker build [OPTIONS] PATH | URL | -`。PATH、URL、-指出了构建镜像的上下文（context），context 中包含了构建镜像需要的Dockerfile文件和其他文件。默认情况下，Docker 构建引擎会查找 context 中名为Dockerfile的文件，但你可以通过-f，--file选项，手动指定Dockerfile文件。例如：

[复制代码](#)

```
1 $ docker build -f Dockerfile -t ccr.ccs.tencentyun.com/marmotedu/iam-apiserve
```

使用 Dockerfile 构建镜像，本质上也是通过镜像创建容器，并在容器中执行相应的指令，然后停止容器，提交存储层的文件变更。和用`docker commit`构建镜像的方式相比，它有三个好处：

Dockerfile 包含了镜像制作的完整操作流程，其他开发者可以通过 Dockerfile 了解并复现制作过程。

Dockerfile 中的每一条指令都会创建新的镜像层，这些镜像可以被 Docker Daemon 缓存。再次制作镜像时，Docker 会尽量复用缓存的镜像层（using cache），而不是重新逐层构建，这样可以节省时间和磁盘空间。

Dockerfile 的操作流程可以通过`docker image history [镜像名称]`查询，方便开发者查看变更记录。

这里，我们通过一个示例，来详细介绍下通过Dockerfile构建镜像的流程。

首先，我们需要编写一个Dockerfile文件。下面是 iam-apiserver 的 [Dockerfile](#) 文件内容：


[复制代码](#)

```
1 FROM centos:centos8
2 MAINTAINER Lingfei Kong <colin404@foxmail.com>
3
4 RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
5 RUN echo "Asia/Shanghai" > /etc/timezone
6
7 WORKDIR /opt/iam
8 COPY iam-apiserver /opt/iam/bin/
9
```

```
10 ENTRYPOINT ["/opt/iam/bin/iam-apiserver"]
```

这里选择centos:centos8作为基础镜像，是因为centos:centos8镜像中包含了基本的排障工具，例如vi、cat、curl、mkdir、cp等工具。

接着，执行docker build命令来构建镜像：

 复制代码

```
1 $ docker build -f Dockerfile -t ccr.ccs.tencentyun.com/marmotedu/iam-apiserver
```

执行docker build后的构建流程为：

第一步，docker build会将 context 中的文件打包传给 Docker daemon。如果 context 中有.dockerignore文件，则会从上传列表中删除满足.dockerignore规则的文件。

这里有个例外，如果.dockerignore文件中有.dockerignore或者Dockerfile，docker build命令在排除文件时会忽略掉这两个文件。如果指定了镜像的 tag，还会对 repository 和 tag 进行验证。

第二步，docker build命令向 Docker server 发送 HTTP 请求，请求 Docker server 构建镜像，请求中包含了需要的 context 信息。

第三步，Docker server 接收到构建请求之后，会执行以下流程来构建镜像：

1. 创建一个临时目录，并将 context 中的文件解压到该目录下。
2. 读取并解析 Dockerfile，遍历其中的指令，根据命令类型分发到不同的模块去执行。
3. Docker 构建引擎为每一条指令创建一个临时容器，在临时容器中执行指令，然后 commit 容器，生成一个新的镜像层。
4. 最后，将所有指令构建出的镜像层合并，形成 build 的最后结果。最后一次 commit 生成的镜像 ID 就是最终的镜像 ID。

为了提高构建效率，docker build默认会缓存已有的镜像层。如果构建镜像时发现某个镜像层已经被缓存，就会直接使用该缓存镜像，而不用重新构建。如果不希望使用缓存的镜像，可以在执行docker build命令时，指定--no-cache=true参数。

Docker 匹配缓存镜像的规则为：遍历缓存中的基础镜像及其子镜像，检查这些镜像的构建指令是否和当前指令完全一致，如果不一样，则说明缓存不匹配。对于ADD、COPY指令，还会根据文件的校验和（checksum）来判断添加到镜像中的文件是否相同，如果不相同，则说明缓存不匹配。

这里要注意，缓存匹配检查不会检查容器中的文件。比如，当使用RUN apt-get -y update命令更新了容器中的文件时，缓存策略并不会检查这些文件，来判断缓存是否匹配。

最后，我们可以通过docker history命令来查看镜像的构建历史，如下图所示：

```
[colin@dev iam]$ docker image history ccr.ccs.tencentyun.com/marmotedu/iam-apiserver-amd64:test
```


IMAGE	CREATED	CREATED BY	SIZE	COMMENT
f34935cad37e	5 hours ago		8.57kB	
7cda0b3483ae	2 weeks ago	/bin/sh -c #(nop) ENTRYPOINT ["/opt/iam/bin...	0B	
b626e2307fc0	2 weeks ago	/bin/sh -c #(nop) COPY file:2bf17185429458ef...	29MB	
e6dc0d1c1b7c	2 weeks ago	/bin/sh -c #(nop) WORKDIR /opt/iam	0B	
50d9f675255b	2 weeks ago	/bin/sh -c echo "Asia/Shanghai" > /etc/timez...	14B	
34651bd9dcb9	2 weeks ago	/bin/sh -c ln -sf /usr/share/zoneinfo/Asia/S...	33B	
aaf0c508836e	2 weeks ago	/bin/sh -c #(nop) MAINTAINER Lingfei Kong <...	0B	
e7b300aee9f9	3 months ago	/bin/sh -c #(nop) CMD ["/bin/sh"]	0B	
<missing>	3 months ago	/bin/sh -c #(nop) ADD file:c5377eaa926bf412d...	5.58MB	

其他制作镜像方式

上面介绍的是两种最常用的镜像构建方式，还有一些其他的镜像创建方式，这里我简单介绍两种。

1. 通过docker save和docker load命令构建

docker save用来将镜像保存为一个tar文件，docker load用来将tar格式的镜像文件加载到当前机器上，例如：

 复制代码

```
1 # 在 A 机器上执行，并将 nginx-v1.0.0.tar.gz 复制到 B 机器
2 $ docker save nginx | gzip > nginx-v1.0.0.tar.gz
3
4 # 在 B 机器上执行
5 $ docker load -i nginx-v1.0.0.tar.gz
```


通过上面的命令，我们就在机器 B 上创建了nginx镜像。

2. 通过docker export和docker import命令构建

我们先通过docker export 保存镜像，再通过docker import 加载镜像，具体命令如下：

[复制代码](#)

```
1 # 在 A 机器上执行，并将 nginx-v1.0.0.tar.gz 复制到 B 机器
2 $ docker export nginx > nginx-v1.0.0.tar.gz
3
4 # 在 B 机器上执行
5 $ docker import - nginx:v1.0.0 nginx-v1.0.0.tar.gz
```

通过docker export导出的镜像和通过docker save保存的镜像相比，会丢失掉所有的镜像构建历史。在实际生产环境中，我不建议你通过docker save和docker export这两种方式来创建镜像。我比较推荐的方式是：在 A 机器上将镜像 push 到镜像仓库，在 B 机器上从镜像仓库 pull 该镜像。

Dockerfile 指令介绍

上面，我介绍了一些与 Docker 镜像构建有关的基础知识。在实际生产环境中，我们标准的做法是通过 Dockerfile 来构建镜像，这就要求你会编写 Dockerfile 文件。接下来，我就详细介绍下如何编写 Dockerfile 文件。

Dockerfile 指令的基本格式如下：

[复制代码](#)

```
1 # Comment
2 INSTRUCTION arguments
```

INSTRUCTION是指令，不区分大小写，但我的建议是指令都大写，这样可以与参数进行区分。Dockerfile 中，以 # 开头的行是注释，而在其他位置出现的 # 会被当成参数，例如：

```
1 # Comment
2 RUN echo 'hello world # dockerfile'
```

一个 Dockerfile 文件中包含了多条指令，这些指令可以分为 5 类。

定义基础镜像的指令：**FROM**；

定义镜像维护者的指令：**MAINTAINER**（可选）；

定义镜像构建过程的指令：**COPY**、**ADD**、**RUN**、**USER**、**WORKDIR**、**ARG**、**ENV**、**VOLUME**、**ONBUILD**；

定义容器启动时执行命令的指令：**CMD**、**ENTRYPOINT**；

其他指令：**EXPOSE**、**HEALTHCHECK**、**STOPSIGNAL**。

其中，加粗的指令是编写 Dockerfile 时经常用到的指令，需要你重点了解下。我把这些常用 Dockerfile 指令的介绍放在了 GitHub 上，你可以看看这个 [🔗 Dockerfile 指令详解](#)。

下面是一个 Dockerfile 示例：

```
1 # 第一行必须指定构建该镜像所基于的容器镜像
2 FROM centos:centos8
3
4 # 维护者信息
5 MAINTAINER Lingfei Kong <colin404@foxmail.com>
6
7 # 镜像的操作指令
8 RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
9 RUN echo "Asia/Shanghai" > /etc/timezone
10 WORKDIR /opt/iam
11 COPY iam-apiserver /opt/iam/bin/
12
13 # 容器启动时执行指令
14 ENTRYPOINT ["/opt/iam/bin/iam-apiserver"]
```


Docker 会顺序解释并执行 Dockerfile 中的指令，并且第一条指令必须是 FROM，FROM 用来指定构建镜像的基础镜像。接下来，一般会指定镜像维护者的信息。后面是镜像操作的指令，最后会通过 CMD 或者 ENTRYPOINT 来指定容器启动的命令和参数。

Dockerfile 最佳实践

上面我介绍了 Dockerfile 的指令，但在编写 Dockerfile 时，只知道这些指令是不够的，还不能编写一个合格的 Dockerfile。我们还需要遵循一些编写 Dockerfile 的最佳实践。这里，我总结了一份编写 Dockerfile 的最佳实践清单，你可以参考。

1. 建议所有的 Dockerfile 指令大写，这样做可以很好地跟在镜像内执行的指令区分开来。
2. 在选择基础镜像时，尽量选择官方的镜像，并在满足要求的情况下，尽量选择体积小的镜像。目前，Linux 镜像大小有以下关系：`busybox < debian < centos < ubuntu`。最好确保同一个项目中使用一个统一的基础镜像。如无特殊需求，可以选择使用 `debian:jessie` 或者 `alpine`。
3. 在构建镜像时，删除不需要的文件，只安装需要的文件，保持镜像干净、轻量。
4. 使用更少的层，把相关的内容放到一个层，并使用换行符进行分割。这样可以进一步减小镜像的体积，也方便查看镜像历史。
5. 不要在 Dockerfile 中修改文件的权限。因为如果修改文件的权限，Docker 在构建时会重新复制一份，这会导致镜像体积越来越大。
6. 给镜像打上标签，标签可以帮助你理解镜像的功能，例如：`docker build -t="nginx:3.0-onbuild"`。
7. FROM 指令应该包含 tag，例如使用 `FROM debian:jessie`，而不是 `FROM debian`。
8. 充分利用缓存。Docker 构建引擎会顺序执行 Dockerfile 中的指令，而且一旦缓存失效，后续命令将不能使用缓存。为了有效地利用缓存，需要尽量将所有的 Dockerfile 文件中相同的部分都放在前面，而将不同的部分放在后面。
9. 优先使用 COPY 而非 ADD 指令。和 ADD 相比，COPY 功能简单，而且也够用。ADD 可变的行会导致该指令的行为不清晰，不利于后期维护和理解。
10. 推荐将 CMD 和 ENTRYPOINT 指令结合使用，使用 `exec` 格式的 ENTRYPOINT 指令设置固定的默认命令和参数，然后使用 CMD 指令设置可变的参数。
11. 尽量使用 Dockerfile 共享镜像。通过共享 Dockerfile，可以使开发者明确知道 Docker 镜像的构建过程，并且可以将 Dockerfile 文件加入版本控制，跟踪起来。
12. 使用 `.dockerignore` 忽略构建镜像时非必需的文件。忽略无用的文件，可以提高构建速度。

13. 使用多阶段构建。多阶段构建可以大幅减小最终镜像的体积。例如，COPY指令中可能包含一些安装包，安装完成之后这些内容就废弃掉。下面是一个简单的多阶段构建示例：

 复制代码

```
1 FROM golang:1.11-alpine AS build
2
3 # 安装依赖包
4 RUN go get github.com/golang/mock/mockgen
5
6 # 复制源码并执行build，此处当文件有变化会产生新的一层镜像层
7 COPY . /go/src/iam/
8 RUN go build -o /bin/iam
9
10 # 缩小到一层镜像
11 FROM busybox
12 COPY --from=build /bin/iam /bin/iam
13 ENTRYPOINT ["/bin/iam"]
14 CMD ["--help"]
```

总结

如果你想使用 Docker 容器来部署应用，那么就需要制作 Docker 镜像。今天，我介绍了如何制作 Docker 镜像。

你可以使用这两种方式来构建 Docker 镜像：

通过 `docker commit` 命令，基于一个已存在的容器构建出镜像。

通过编写 Dockerfile 文件，并使用 `docker build` 命令来构建镜像。

这两种方法中，镜像构建的底层原理是相同的：

1. 基于原镜像启动一个 Docker 容器。
2. 在容器中进行一些操作，例如执行命令、安装文件等，由这些操作产生的文件变更都会被记录在容器的存储层中。
3. 将容器存储层的变更 `commit` 到新的镜像层中，并添加到原镜像上。

此外，我们还可以使用 `docker save / docker load` 和 `docker export / docker import` 来复制 Docker 镜像。

在实际生产环境中，我们标准的做法是通过 Dockerfile 来构建镜像。使用 Dockerfile 构建镜像，就需要你编写 Dockerfile 文件。Dockerfile 支持多个指令，这些指令可以分为 5 类，对指令的具体介绍你可以再返回复习一遍。

另外，我们在构建 Docker 镜像时，也要遵循一些最佳实践，具体你可以参考我给你总结的最佳实践清单。

课后练习

1. 思考下，为什么在编写 Dockerfile 时，“把相关的内容放到一个层，使用换行符 \ 进行分割”可以减小镜像的体积？
2. 尝试一下，为你正在开发的应用编写 Dockerfile 文件，并成功构建出 Docker 镜像。

欢迎你在留言区与我交流讨论，我们下一讲见。

分享给需要的人，Ta订阅后你可得 **24 元现金奖励**

 赞 2  提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 45 | 基于Kubernetes的云原生架构设计

下一篇 47 | 如何编写Kubernetes资源定义文件？

更多学习推荐

175 道 Go 工程师 大厂常考面试题

限量免费领取 



精选留言 (2)

 写留言



jxlwqq

2021-09-13

自荐一个dockerfile的写法：

```Dockerfile

# 多阶段构建：提升构建速度，减少镜像大小

...

展开 



 4



随风而过

2021-09-12

官方文档中最佳实践有介绍，RUN, COPY, ADD 三个指令会创建层，其他指令会创建一个中间镜像，并且不会影响镜像大小。这样我们说的指令合并也就是以这三个指令为主。当然了docker history查看构建历史与镜像大小，更为易读和简约

展开 



 4