容器镜像构建和交付的最佳实践

2022 年 Q4 版

张晋涛

个人介绍

- 张晋涛
- Apache APISIX PMC
- Kubernetes Ingress NGINX maintainer
- Microsoft MVP
- 【K8S 生态周报】维护者
- 公众号： MoeLove
- https://github.com/tao12345666333

目录

CONTENTS

容器镜像是什么

一系列配置文件和归档文件

- 通过 skopeo 下载到本地

```
→  debian-oci skopeo copy docker://index.docker.io/library/debian
dir:./debian
Getting image source signatures
Copying blob a8ca11554fce done
Copying config c31f65dd4c done
Writing manifest to image destination
Storing signatures
→  debian-oci tree debian
debian
├── a8ca11554fce00d9177da2d76307bdc06df7faeb84529755c648ac4886192ed1
├── c31f65dd4cc97f21bd440df4b9b919a99e35a7ede602e8150f2314af1441a312
├── manifest.json
└── version

0 directories, 4 files
```

```
→  debian-oci file debian/a8ca11554fce00d9177da2d76307bdc06df7faeb84529755c648ac4886192ed1
debian/a8ca11554fce00d9177da2d76307bdc06df7faeb84529755c648ac4886192ed1: gzip compressed data, original size modulo
2^32 129330176
→  debian-oci file debian/c31f65dd4cc97f21bd440df4b9b919a99e35a7ede602e8150f2314af1441a312
debian/c31f65dd4cc97f21bd440df4b9b919a99e35a7ede602e8150f2314af1441a312: JSON text data
```

一系列配置文件和归档文件

- 通过 docker save 保存到本地

```
➜  debian-image docker save -o debian.tar debian
➜  debian-image mkdir -p debian
➜  debian-image tar -C debian -xf debian.tar
➜  debian-image tree debian
debian
├── 860b7460ac90e2e79c6e06ccae67c6bcbcaa1abbba22d37602cba3f121486e04
│   ├── json
│   ├── layer.tar
│   └── VERSION
├── c31f65dd4cc97f21bd440df4b9b919a99e35a7ede602e8150f2314af1441a312.json
├── manifest.json
└── repositories

1 directory, 6 files
```

一系列配置文件和归档文件

docker save 是解压后的内容

```
→  debian-oci jq . debian/manifest.json
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
  "config": {
    "mediaType": "application/vnd.docker.container.image.v1+json",
    "size": 1463,
    "digest":
"sha256:c31f65dd4cc97f21bd440df4b9b919a99e35a7ede602e8150f2314af1441a312"
  },
  "layers": [
    {
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
      "size": 55038615,
      "digest":
"sha256:a8ca11554fce00d9177da2d76307bdc06df7faeb84529755c648ac4886192ed1"
    }
  ]
}
```

```
→  debian-oci docker image ls --digests debian
REPOSITORY   TAG       DIGEST                                                                    IMAGE ID
CREATED       SIZE
debian       latest    sha256:3066ef83131c678999ce82e8473e8d017345a30f5573ad3e44f62e5c9c46442b   c31f65dd4cc9   2
weeks ago   124MB
```

一系列配置文件和归档文件

- 内容寻址是其核心
- 容器镜像可以由很多层进行叠加

```
→  debian-oci jq . debian/manifest.json
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
  "config": {
    "mediaType": "application/vnd.docker.container.image.v1+json",
    "size": 1463,
    "digest":
"sha256:c31f65dd4cc97f21bd440df4b9b919a99e35a7ede602e8150f2314af1441a312"
  },
  "layers": [
    {
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
      "size": 55038615,
      "digest":
"sha256:a8ca11554fce00d9177da2d76307bdc06df7faeb84529755c648ac4886192ed1"
    }
  ]
}
```

```
→  debian-oci docker image history debian
IMAGE          CREATED       CREATED BY
  SIZE      COMMENT
c31f65dd4cc9   2 weeks ago   /bin/sh -c #(nop)  CMD ["bash"]
0B
<missing>      2 weeks ago   /bin/sh -c #(nop) ADD file:abc6873c98339a3c4…
124MB
```

如何构建容器镜像

- Dockerfile 是“构建规则”
- docker build 是最常用的命令
- 。。。

如何选择镜像构建工具

- https://github.com/docker/docker
- https://github.com/containers/buildah
- https://github.com/GoogleContainerTools/kaniko
- https://github.com/ko-build/ko
- https://github.com/buildpacks/pack
- https://github.com/moby/buildkit
- https://github.com/genuinetools/img(不维护)
- ...

- 当前基础设施
- 是否需要 deamon 运行
- 通用 / 专用工具
- 是否兼容 Dockerfile

- Buildkit 下一代 Docker 构建引擎
- Docker Desktop 默认启用 BuildKit
- docker 已经将默认的 docker build 切换为了 buildx(为 buildkit 所做的扩展)

- 通过 syntax=docker/dockerfile:1 设置语法
- 增加 here-docs 支持
- 书写更加简单
- 支持 shebang (#!) 比如 python 等语言

```
FROM debian

RUN apt-get update && \
        apt-get upgrade -y && \
        apt-get install -y dnsutils && \
        rm -rf /var/lib/apt/lists/*
```

```
# syntax=docker/dockerfile:1
FROM debian

RUN <<EOF
    apt-get update
    apt-get upgrade -y
    apt-get install -y dnsutils
    rm -rf /var/lib/apt/lists/*
EOF
```

利用缓存提升构建效率

Docker 有构建时缓存，已执行命令可被缓存。 COPY 命令会终止该缓存

```
→  buildkit-demo docker build -t local/debian:bad-style .
[+] Building 37.0s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 223B                                                                                0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => resolve image config for docker.io/docker/dockerfile:1                                                          2.6s
 => CACHED docker-image://docker.io/docker/dockerfile:1@sha256:9ba7531bd80fb0a858632727cf7a112fbfd19b17e94c4e84ced81e24ef1a0dbc  0.0s
 => [internal] load metadata for docker.io/library/debian:latest                                                    0.0s
 => CACHED [1/4] FROM docker.io/library/debian                                                                      0.0s
 => [2/4] RUN apt-get update                                                                                       22.9s
 => [3/4] RUN apt-get install -y dnsutils                                                                          10.2s
 => [4/4] RUN rm -rf /var/lib/apt/lists/*                                                                           0.3s
 => exporting to image                                                                                              0.7s
 => => exporting layers                                                                                             0.7s
 => => writing image sha256:b4931683c0f77f67118b96c6c6048fe4fea6e7aa7b7c42a79616ee6c5f8c7284                        0.0s
 => => naming to docker.io/local/debian:bad-style                                                                   0.0s
→  buildkit-demo docker build -t local/debian:bad-style .
[+] Building 1.3s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 223B                                                                                0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => resolve image config for docker.io/docker/dockerfile:1                                                          1.0s
 => CACHED docker-image://docker.io/docker/dockerfile:1@sha256:9ba7531bd80fb0a858632727cf7a112fbfd19b17e94c4e84ced81e24ef1a0dbc  0.0s
 => [internal] load metadata for docker.io/library/debian:latest                                                    0.0s
 => [1/4] FROM docker.io/library/debian                                                                             0.0s
 => CACHED [2/4] RUN apt-get update                                                                                 0.0s
 => CACHED [3/4] RUN apt-get install -y dnsutils                                                                    0.0s
 => CACHED [4/4] RUN rm -rf /var/lib/apt/lists/*                                                                    0.0s
 => exporting to image                                                                                              0.0s
 => => exporting layers                                                                                             0.0s
 => => writing image sha256:b4931683c0f77f67118b96c6c6048fe4fea6e7aa7b7c42a79616ee6c5f8c7284                        0.0s
 => => naming to docker.io/local/debian:bad-style                                                                   0.0s
→  buildkit-demo
```

```yaml
name: ci
on:
  push:
    branches:
      - 'main'

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v1
      - name: Login to Docker Hub
        uses: docker/login-action@v1
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      - name: Build and push
        uses: docker/build-push-action@v2
        with:
          push: true
          tags: user/app:latest
```

mount=type=bind 可直接从 build context 或者 stage 进行文件操作

mount=type=cache 可进行应用的缓存

```dockerfile
# syntax=docker/dockerfile:1

FROM golang:1.19-alpine AS build
WORKDIR /src
RUN --mount=type=bind,target=. \
  go build -ldflags "-s -w" -o /usr/bin/app .

FROM alpine
COPY --from=build /usr/bin/app /usr/bin/app
CMD ["app"]
```

Buildkit 支持多种 cache： inline， local，registy 和 gha

```yaml
name: ci
on:
  push:
    branches:
      - 'main'

jobs:
  docker:
    runs-on: ubuntu-latest
    steps:
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v1
      - name: Login to Docker Hub
        uses: docker/login-action@v1
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      - name: Build and push
        uses: docker/build-push-action@v2
        with:
          push: true
          tags: user/app:latest
          cache-from: gha
          cache-to: gha
```

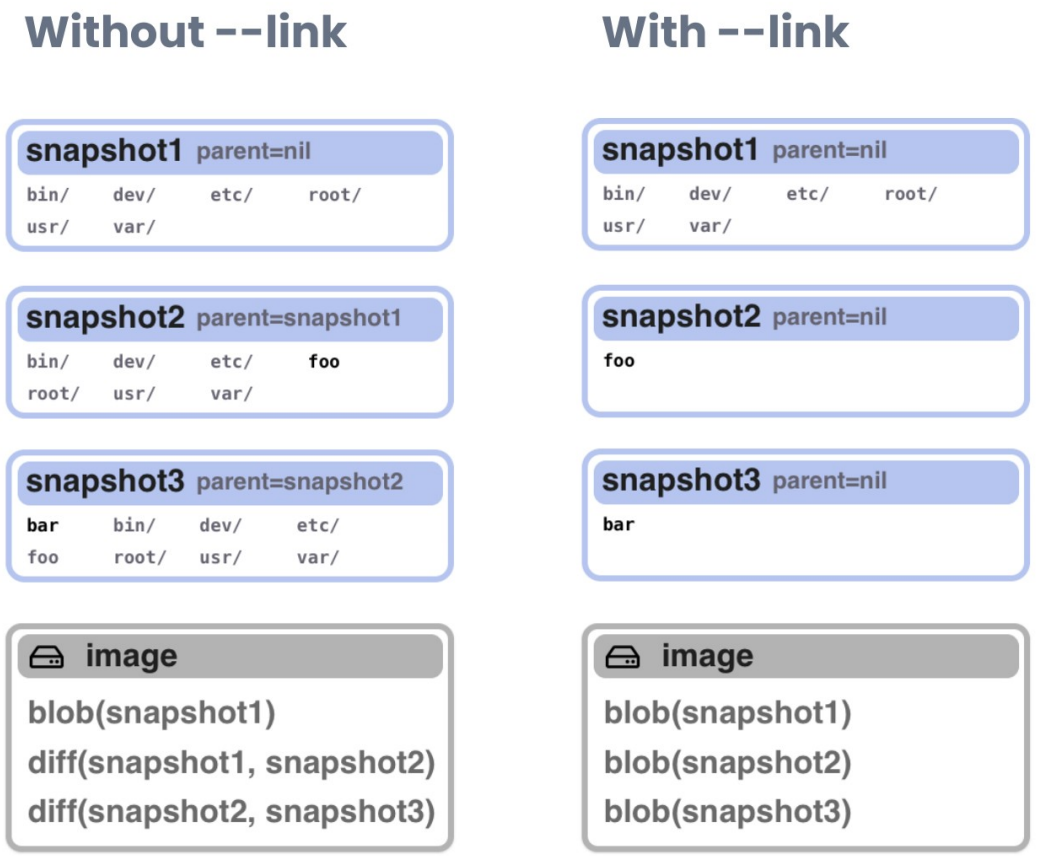
```
#15 importing cache manifest from gha:2200847280167466126
#15 DONE 0.5s

#13 [internal] load build context
#13 transferring context: 1.13kB done
#13 DONE 0.0s

#16 [build 2/3] WORKDIR /src
#16 CACHED

#17 [build 3/3] RUN --mount=type=bind,target=/src   go build
-ldflags "-s -w" -o /usr/bin/app .
#17 CACHED

#18 [stage-1 2/2] COPY --from=build /usr/bin/app /usr/bin/app
#18 CACHED

#19 exporting cache
#19 preparing build cache for export done
#19 DONE 0.4s
```

通过 --link 更高效的利用缓存

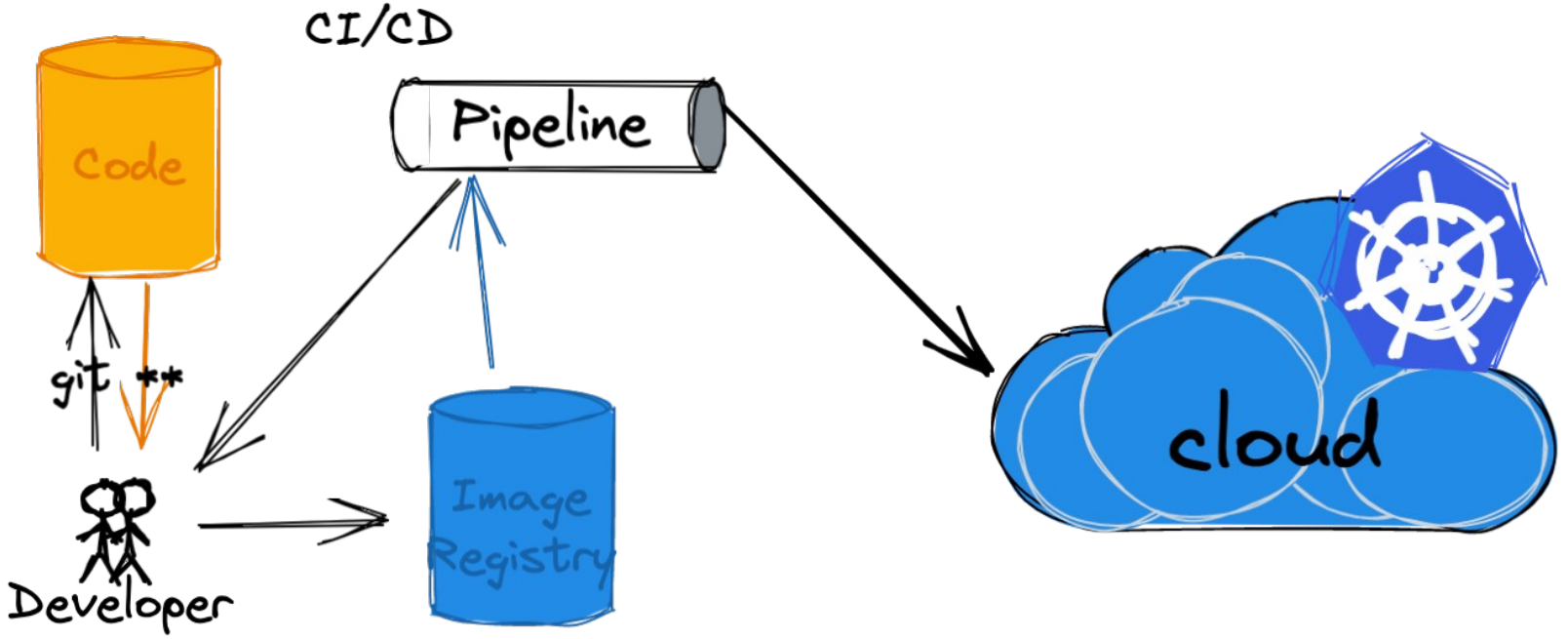
尤其是对多阶段构建

可以 --link 加 --from 配合

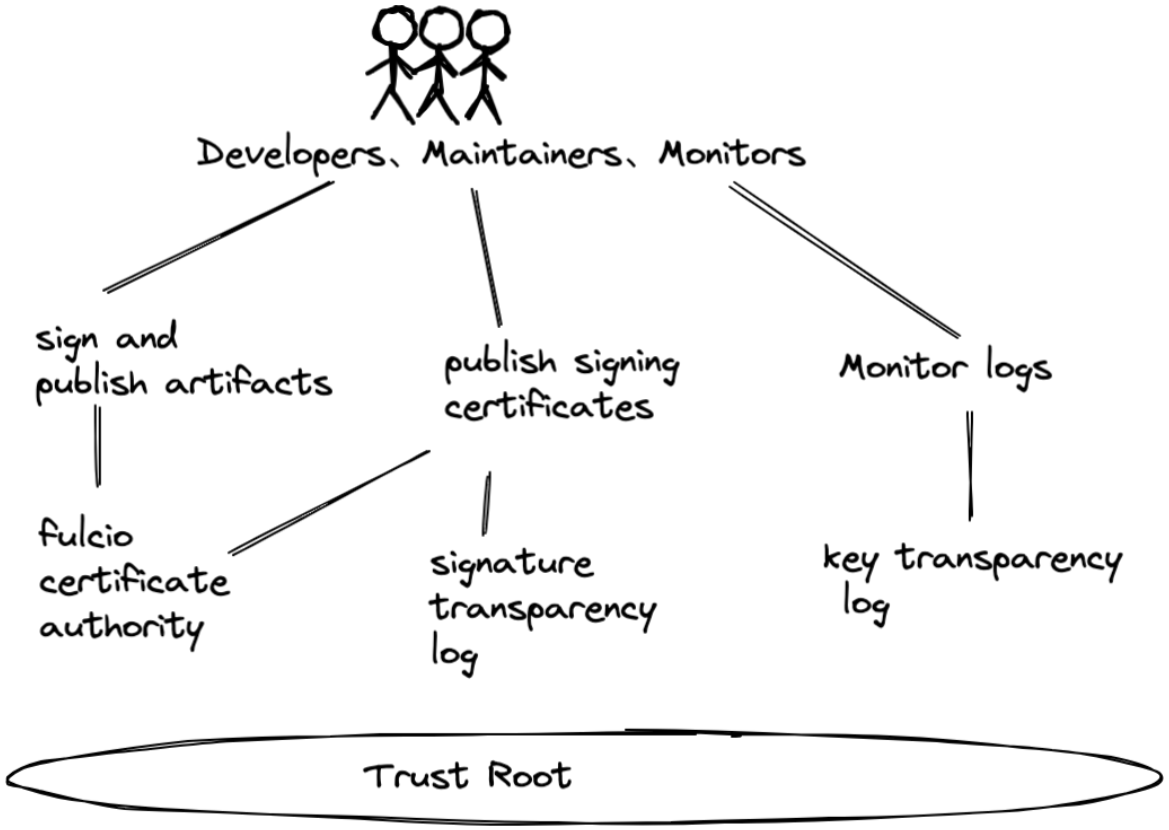
```
FROM alpine

COPY --link foo /

COPY --link bar /
```



如何进行镜像交付


CI/CD
Code
Pipeline
git **
Developer
Image Registry
cloud

使用 sigstore 和 cosign 进行签发校验



Thanks!

https://github.com/tao12345666333