docker

# Module Six

Best Practices:

- Governance
- Containers
- SDLC

# Repositories and Tags

# Docker Hub: Team Best Practices

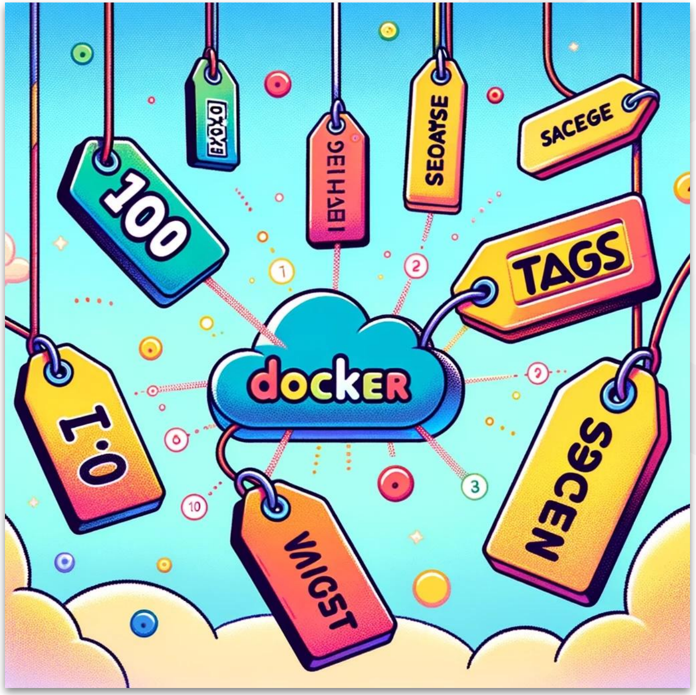
- **Manage Permissions**
  - Organizations / Teams
  - Manage Users and Repos
- **Access Control**
  - Assign Roles / Permissions
- **Shared Repositories**
  - Encourage Collaboration
  - Document Process

# Docker Hub: Repository Best Practices

- Follow a Naming Convention
  - Consistency and Clarity are Key
  - Avoid Non-Standard Names
- Public vs Private Repos
  - Open Source Project
  - Internal and Confidential
- Repository Management
  - Prune Unused Repositories
  - Review Security

# Docker Hub: Tag Best Practices

- Use to Manage Images
  - Consistency and Clarity are Key
  - Helps with Rollback
  - Helps with Image Selection
- Use Semantic Versioning
  - Helps Track Scope of Changes
  - Major.Minor.Patch
- Define a Tagging Strategy
  - Document and Enforce
- Use Additional Tags as Required
  - Images Can Have 1 to n Tags



docker

## Docker Labels

# Docker Labels

**Key-value pairs attached to Docker objects**

- Image / Containers
- Adds Additional Metadata

**Facilitate**

- Organization of Images
- Filtering of Images
- Provide Information

**Best Practices**

- Consistency
- Documentation

## Adding Labels

### Dockerfile:

```
FROM ubuntu
LABEL version="1.0" description="Our application" maintainer="admin@example.com"
```

### CLI:

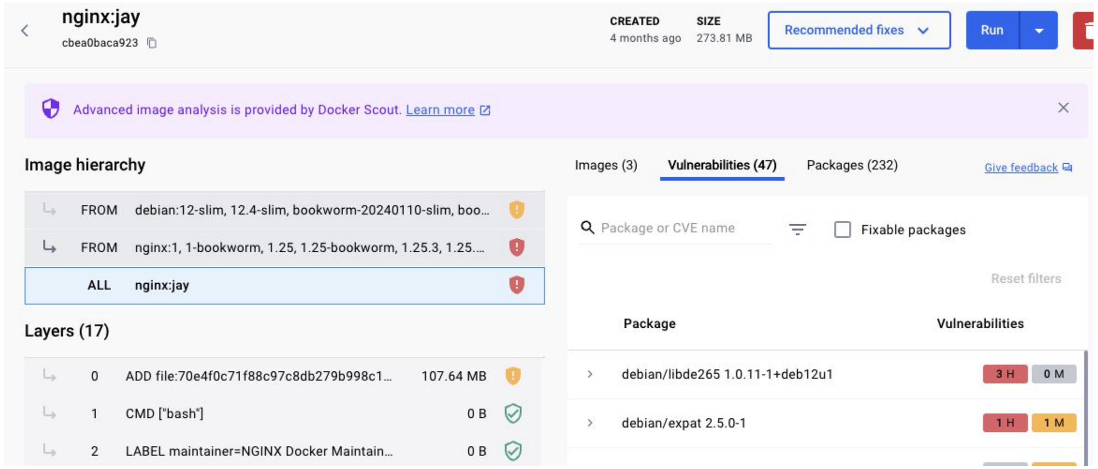
```
$ docker build --label "version=1.0" --label "description=Our application" .
```

# Viewing Labels

## CLI:

```
$ docker inspect --format='{{json .Config.Labels}}' <image-name>
```

## GUI:



nginx:jay
cbea0baca923

CREATED 4 months ago
SIZE 273.81 MB

Advanced image analysis is provided by Docker Scout. Learn more

Image hierarchy

| | |
|---|---|
| FROM | debian:12-slim, 12.4-slim, bookworm-20240110-slim, boo... |
| FROM | nginx:1, 1-bookworm, 1.25, 1.25-bookworm, 1.25.3, 1.25.... |
| ALL | nginx:jay |

Layers (17)

| | | |
|---|---|---|
| 0 | ADD file:70e4f0c71f88c97c8db279b998c1... | 107.64 MB |
| 1 | CMD ["bash"] | 0 B |
| 2 | LABEL maintainer=NGINX Docker Maintain... | 0 B |

| Package | Vulnerabilities |
|---|---|
| debian/libde265 1.0.11-1+deb12u1 | 3 H 0 M |
| debian/expat 2.5.0-1 | 1 H 1 M |

docker

## Docker Image Governance

## Governance

in Docker image creation and configuration

### Importance of governance

- Consistency
- Security
- Standardization

### Key principles of governance

- Version Control
- Documentation
- Testing
- Collaboration / InnerSource

# Best practices

for creating and configuring Docker images

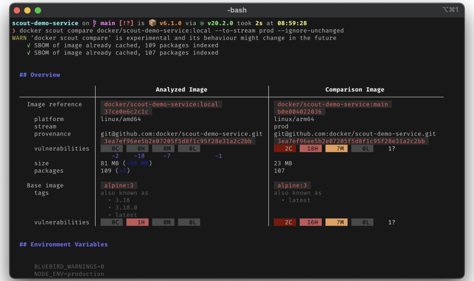
- Image Optimization
- Secure Configuration
- Lifecycle Management

# Security considerations

in Docker image governance

## Vulnerability Scanning

docker scout



```
-bash                                                                          ⌥⌘1
scout-demo-service on  main [!?] is 📦 v6.1.0 via  v20.2.0 took 2s at 08:59:28
❯ docker scout compare docker/scout-demo-service:local --to-stream prod --ignore-unchanged
WARN 'docker scout compare' is experimental and its behaviour might change in the future
    √ SBOM of image already cached, 109 packages indexed
    √ SBOM of image already cached, 107 packages indexed


  ## Overview

                          │              Analyzed Image                │           Comparison Image
  ────────────────────────┼────────────────────────────────────────────┼──────────────────────────────────────────────
    Image reference       │  docker/scout-demo-service:local           │  docker/scout-demo-service:main
                          │  37ce0e6c2c1c                              │  b0e004022036
      platform            │  linux/amd64                               │  linux/arm64
      stream              │                                            │  prod
      provenance          │  git@github.com:docker/scout-demo-service.git │ git@github.com:docker/scout-demo-service.git
                          │  3ea7ef96ee5b2e07205f5d8f1c95f28e31a2c2bb  │  3ea7ef96ee5b2e07205f5d8f1c95f28e31a2c2bb
      vulnerabilities     │    0C     0H     0M     0L                 │    2C    18H     7M     0L     1?
                          │    -2    -18     -7           -1           │
      size                │  81 MB (+59 MB)                            │  23 MB
      packages            │  109 (+2)                                  │  107
                          │                                            │
    Base image            │  alpine:3                                  │  alpine:3
      tags                │  also known as                             │  also known as
                          │    • 3.18                                  │    • latest
                          │    • 3.18.0                                │
                          │    • latest                                │
      vulnerabilities     │    0C     1H     0M     0L                 │    2C    16H     7M     0L     1?


  ## Environment Variables


      BLUEBIRD_WARNINGS=0
      NODE_ENV=production
```

## Access Control

Docker Administration

Settings Management

Enhanced Container Isolation

Registry Access Management

Image Access Management

## Compliance Checks

Policy Evaluation with

docker scout

https://docs.docker.com/scout/policy

## Conclusion and key takeaways

### Establish Clear Policies

Develop and communicate clear policies regarding Docker image creation, configuration, and usage within the organization.

### Regular Auditing

Implement regular audits and reviews of Docker images to ensure compliance and security.

### Continuous Auditing?

### Continuous Improvement

Emphasize continuous improvement in Docker image governance processes by learning from incidents and feedback.

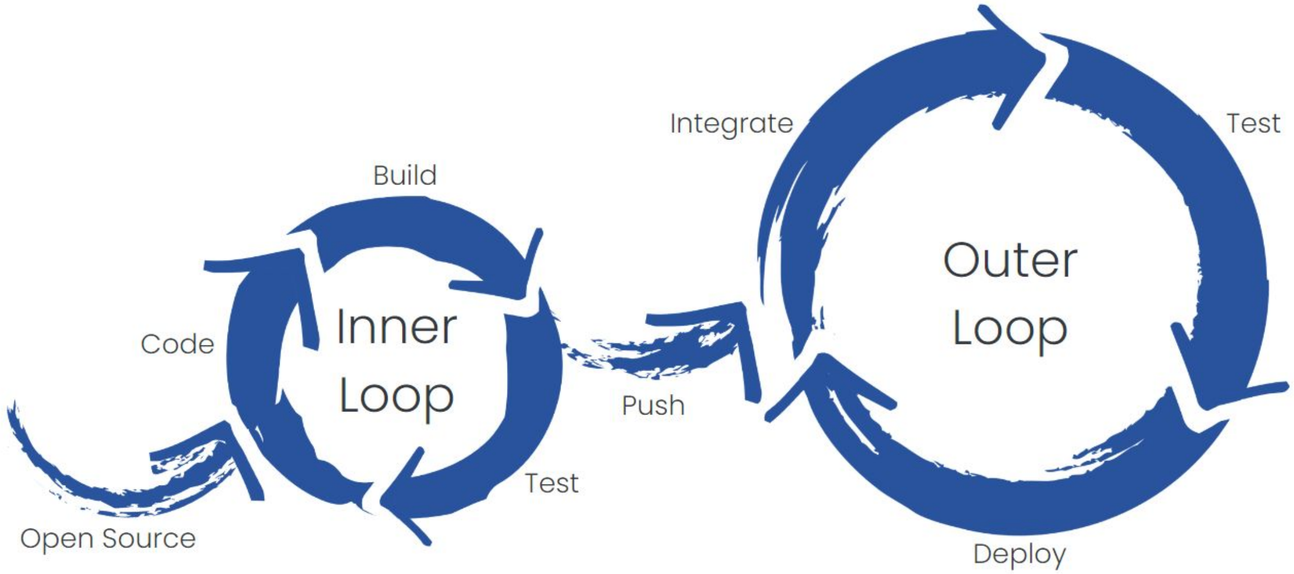
### Educate Teams

Provide ongoing education and training to teams involved in the creation and management of Docker images.

docker

## Docker in the SDLC

## SDLC Overview



# Dependencies Best Practices

Have a standardized process for accepting/using base images and other dependencies

Have a trusted software supply chain strategy that complies with your internal standards and external requirements

## Code Best Practices

Have a clear onramp process for teams that want to containerize their applications

Standardize container tooling configuration across teams/organizations

Ensure the developer environment is both secure and productive

Reduce the number of tools/steps the developers have to use/take to accomplish tasks so they actually can "shift left"

Make the inner loop process as fast as possible so developers are more productive

Determine where dev tools will be deployed (host, container)

Determine when hybrid or remote development makes sense

Code
Inner Loop
Open Source

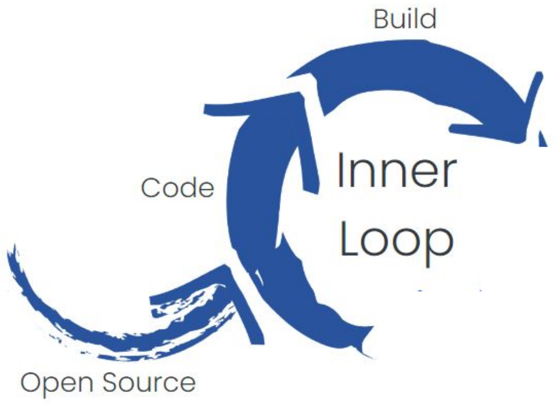
## Build Best Practices

Developer images should have all the tools they need to work (debugging, etc)

Production images should be minimal size, built fast, and highly secure

Utilize the many Docker build features as you grow your build maturity (caching, layers, multi-stage, multi-architecture, Bake (build orchestration), Build Cloud)

Developer build time is wasted Developer time



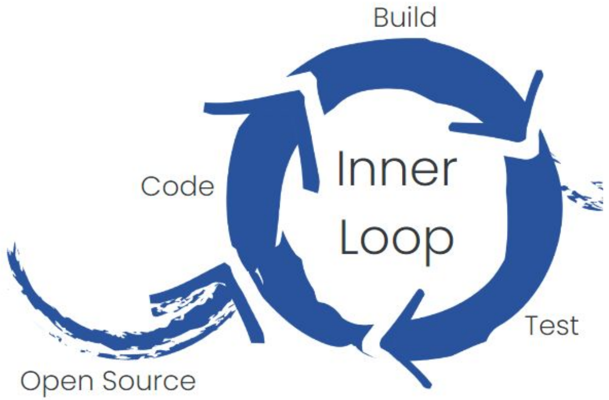
## Test Best Practices

Determine which testing developers will be responsible for (unit, security/policy, integration, etc)

Have a standard way for developers to create full dev environments

Determine how shared test resources (data sets, queues, environments) will be used

Make the inner loop spin as fast as possible (faster feedback means better productivity)

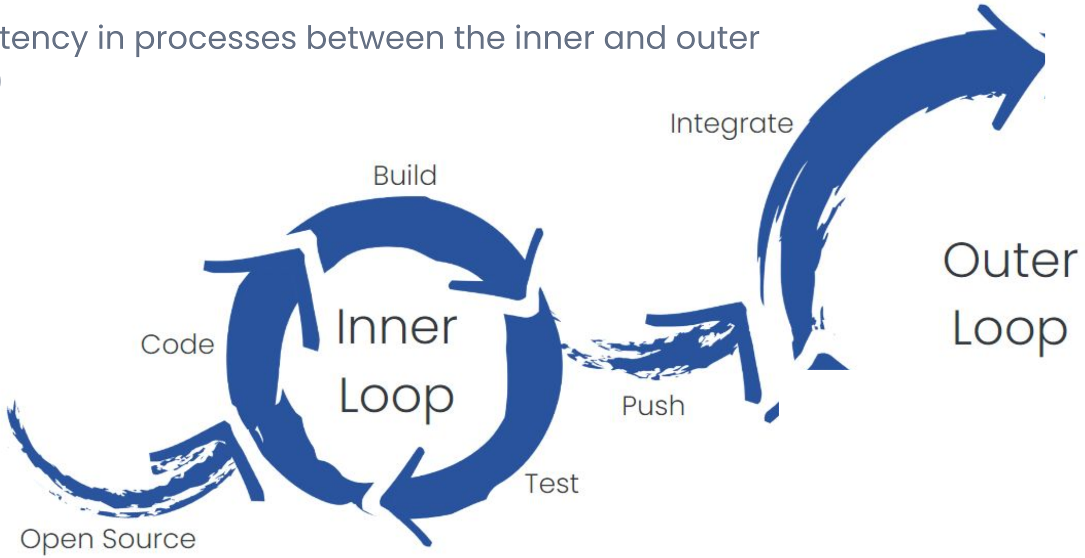


# Integrate Best Practices

Optimized your pipelines for working with Docker

Ensure builds are quick and small

Ensure consistency in processes between the inner and outer loops (builds)

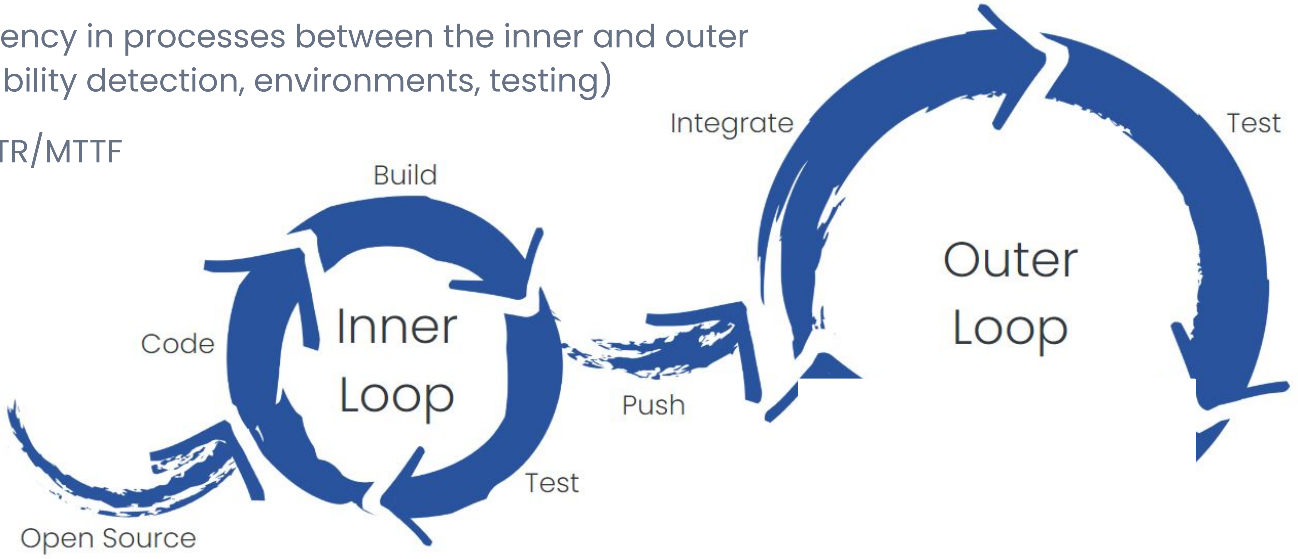


# Test Best Practices

Determine what types of testing you will do as part of your CI process

Ensure consistency in processes between the inner and outer loops (vulnerability detection, environments, testing)

Testing for MTTR/MTTF

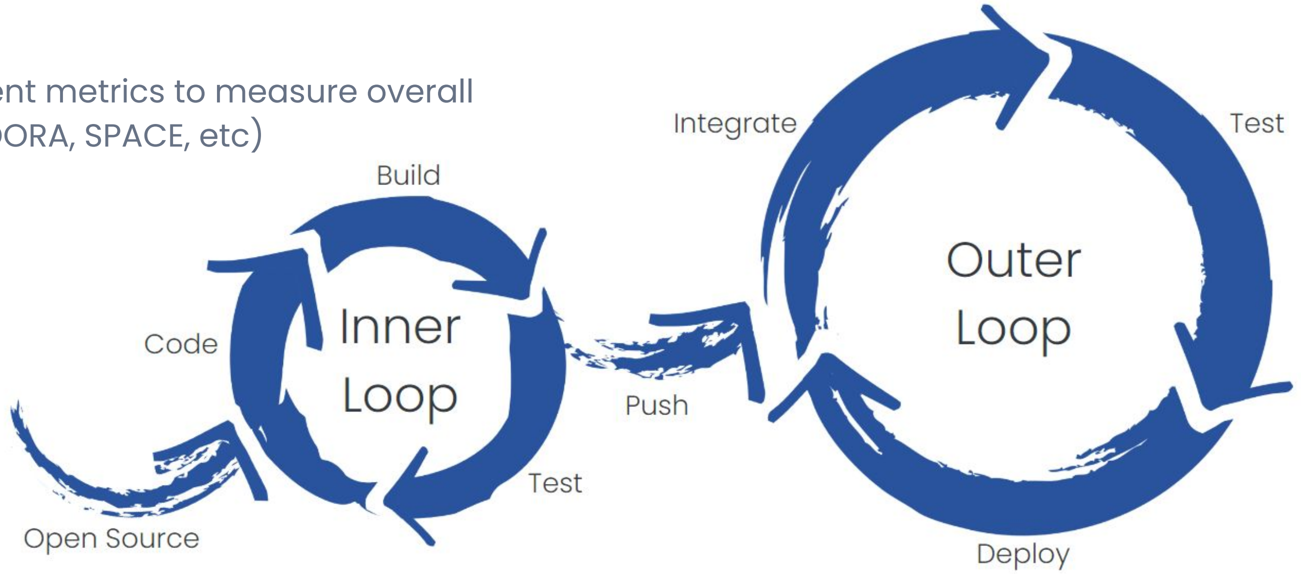


# Deploy Best Practices

Have clear processes for rollbacks

Utilize advanced deployment techniques like blue/green, canary

Use deployment metrics to measure overall productivity(DORA, SPACE, etc)



docker

## Container Best Practices

## The HEALTHCHECK Directive

- Provided in Dockerfile
- Use to Detect
  - Application crashes
  - Dependency failures
  - Resource limitations
  - Misconfigurations
- Configuration
  - Command
  - Interval
  - Timeout
  - Start delay
  - Retries

# The HEALTHCHECK Directive: What it Looks Like

```
$ cat Dockerfile
FROM nginx:latest
HEALTHCHECK --interval=30s --timeout=3s \
  CMD curl -f http://localhost/ || exit 1
EXPOSE 80

$ docker image build -t nginx:latest .

$ docker run --name=nginx-proxy -d --health-cmd='stat /etc/nginx/nginx.conf || exit 1' \
        Nginx:latest

$ docker inspect --format='{{json .Config.Healthcheck}}' nginx-proxy

{
  "Test": [
    "CMD-SHELL",
    "stat /etc/nginx/nginx.conf || exit 1"
  ],
  "Interval": 30000000000,
  "Timeout": 3000000000
}
```

# The HEALTHCHECK Directive: Testing

```
$ docker exec nginx-proxy rm /etc/nginx/nginx.conf

$ docker inspect --format='{{json .State.Health}}' nginx-proxy | jq
{
  "Status": "unhealthy",
  "FailingStreak": 3,
  "Log": [
    {
      "Start": "2024-03-19T22:13:25.915145969Z",
      "End": "2024-03-19T22:13:25.935736635Z",
      "ExitCode": 0,
      "Output": "  File: /etc/nginx/nginx.conf\n  Size: 648       \tBlocks: 8          IO Block:
4096   regular file\nDevice: 0,306\tInode: 2146441     Links: 1\nAccess: (0644/-rw-r--r--)  Uid: (
0/    root)   Gid: (    0/    root)\nAccess: 2023-10-24 16:10:31.000000000 +0000\nModify:
2023-10-24 16:10:31.000000000 +0000\nChange: 2024-01-25 19:58:12.249971010 +0000\n Birth:
2024-01-25 19:58:12.249971010 +0000\n"
    },
    {
      "Start": "2024-03-19T22:14:25.970379552Z",
      "End": "2024-03-19T22:14:26.010667552Z",
      "ExitCode": 1,
      "Output": "stat: cannot statx '/etc/nginx/nginx.conf': No such file or directory\n"
    },
← SNIP →
```

## Docker Logging

- Multiple Configuration Points
- Have a Plan
  - What are you interested in?
  - Is this only of local interest?
  - Should we aggregate?
- Be Tidy
  - Configure log rotation
  - Purge logs as needed
- Be Mindful
  - Honor notification levels
  - Does this need to be a log message?

## Start with OTEL

- Tracing
  - Span Data
  - Context Propagation
- Metrics
  - Metric Collection
  - Metric Export
- Logs
  - Structured Logging
- Context Management
  - Context APIs
- Integration and Instrumentation
  - Auto-Instrumentation
  - Manual Instrumentation
- Exporting and Forwarding Data
  - Pluggable Exporters
  - Protocol and Format Support
- Vendor Neutrality
  - Interoperability

docker

## OCI Label Recommendations

## Container (Software) Specifications

- Docker created the Open Container Initiative in June 2015
- Currently owned by the Linux Foundation
- Currently defines three specifications
  - `image-spec` - defines image structures and manifests
  - `runtime-spec` - defines how to run OCI images
  - `distribution-spec` - defines the API protocol to push, pull, and discover content

# OCI Recommended Labels

| Label | Content |
|---|---|
| org.opencontainers.image.created | The date and time on which the image was built (string, RFC 3339 date-time). |
| org.opencontainers.image.authors | Contact details of the people or organization responsible for the image (freeform string). |
| org.opencontainers.image.url | URL to find more information on the image (string). |
| org.opencontainers.image.documentation | URL to get documentation on the image (string). |
| org.opencontainers.image.source | URL to the source code for building the image (string). |
| org.opencontainers.image.version | Version of the packaged software (string). |
| org.opencontainers.image.revision | Source control revision identifier for the image (string). |
| org.opencontainers.image.vendor | Name of the distributing entity, organization, or individual (string). |
| org.opencontainers.image.licenses | License(s) under which contained software is distributed (string, SPDX License List). |
| org.opencontainers.image.ref.name | Name of the reference for a target (string). |
| org.opencontainers.image.title | Human-readable title of the image (string). |
| org.opencontainers.image.description | Human-readable description of the software packaged in the image (string). |

docker

## Questions and Answers