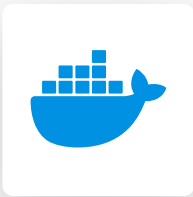


- Secrets
- ABAP
- Apex
- AzureResourceManager
- C
- C#
- C++
- CloudFormation
- COBOL
- CSS
- Dart
- Docker
- Flex
- Go
- HTML
- Java
- JavaScript
- JCL
- Kotlin
- Kubernetes
- Objective C
- PHP
- PL/I
- PL/SQL
- Python
- RPG
- Ruby
- Scala
- Swift
- Terraform
- Text
- TypeScript
- T-SQL
- VB.NET
- VB6
- XML



Docker static code analysis

Unique rules to find Vulnerabilities, Security Hotspots, and Code Smells in your DOCKER code

All rules 44 Vulnerability 4 Bug 4 Security Hotspot 15 Code Smell 21

Tags ▾

Impact ▾

Clean code attribute ▾

Search by name... 🔍

Dockerfile should only have one ENTRYPOINT and CMD instruction	Bug
Access variable which is not available in the current scope	Bug
A space before the equal sign in key-value pair may lead to unintended behavior	Bug
Allowing downgrades to a clear-text protocol is security-sensitive	Security Hotspot
Allowing shell scripts execution during package installation is security-sensitive	Security Hotspot
Using host operating system namespaces is security-sensitive	Security Hotspot
Setting loose POSIX file permissions is security-sensitive	Security Hotspot
Reduce the amount of consecutive RUN instructions	Code Smell
Prefer COPY over ADD for copying local resources	Code Smell
WORKDIR instruction should only be used with absolute path	Code Smell
Too long RUN instruction should be split into multiple lines	Code Smell

Dockerfile should only have one ENTRYPOINT and CMD instruction

Analyze your code

Intentionality - Clear Reliability ⬇

Bug Major ⓘ

The Dockerfile should contain at most one ENTRYPOINT and one CMD instruction, because only the last one will have an effect.

- Why is this an issue?
- How can I fix it?
- More Info

Previous ENTRYPOINT and CMD instructions should be removed to avoid this.

Code examples

Noncompliant code example

```
FROM busybox
ENTRYPOINT ignored_entrypoint param1 param2
ENTRYPOINT effective_entrypoint param1 param2
```

```
CMD ignored_cmd param1 param2
CMD effective_cmd param1 param2
```

Here we have multiple ENTRYPOINT and CMD instructions. The first ENTRYPOINT and the first CMD instructions will have no effect. Although this is valid in Docker, this can lead to confusion and be error-prone, as we may expect each CMD and ENTRYPOINT to have an effect.

Multi-Stage Build:

```
FROM scratch as development
CMD ignored_scratch_cmd param1 param2
CMD effective_scratch_cmd param1 param2
```

```
FROM busybox
CMD ignored_busyBox_cmd param1 param2
CMD effective_busyBox_cmd param1 param2
```

For multi-stage builds we take each stage into account separately. This is because there are valid docker setups, where the file should only be build up to a certain stage. In the example, the developer builds only the first stage as a development environment via `docker build --target development`.

Compliant solution

```
FROM busybox
ENTRYPOINT effective_entrypoint param1 param2
```

```
CMD effective_cmd param1 param2
```

Here we have only one ENTRYPOINT and one CMD instruction. Each of them will be considered by the docker container and have a normal effect, as we can expect.

Multi-Stage Build:

```
FROM scratch as development
CMD effective_scratch_cmd param1 param2
```

```
FROM busybox
CMD effective_busyBox_cmd param1 param2
```

For each stage, we only have one CMD or ENTRYPOINT instruction.

Available In:

sonarlint | sonarcloud | sonarqube

