





































-  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...



Environment variables should not be unset on a different layer than they were set

 Code Smell

Expanded filenames should not become options

 Code Smell

Double quote to prevent globbing and word splitting

 Code Smell

Instructions should be upper case

 Code Smell

Allowing non-root users to modify resources copied to an image is security-sensitive

 Security Hotspot

Automatically installing recommended packages is security-sensitive

 Security Hotspot

Running containers as a privileged user is security-sensitive

 Security Hotspot

Delivering code in production with debug features activated is security-sensitive

 Security Hotspot

Use ADD instruction to retrieve remote resources

 Code Smell

Arguments in long RUN instructions should be sorted

 Code Smell

Track uses of "TODO" tags

 Code Smell

## Environment variables should not be unset on a different layer than they were set

Analyze your code

Consistency - Conventional

Maintainability ⬆

 Code Smell

 Major ?

Setting an environment variable using the `ENV` instruction creates a new layer in the Docker image. The variable is then persisted for all subsequent build stages and is also present in the resulting image. Calling `RUN unset <env-variable>` unsets the variable only for this particular layer, but it is still possible to dump the environment variable from the previous layer.

Why is this an issue?

How can I fix it?

More Info

### Documentation

- [ENV - Dockerfile reference](#)
- [ENV - Best practices for writing Dockerfiles](#)

Available In:

**sonarlint**



**sonarcloud**



**sonarqube**



© 2008-2024 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE, and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

Sonar helps developers write Clean Code.  
[Privacy Policy](#) | [Cookie Policy](#)

