





































-  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...



Consent flag should be set to avoid manual input

 Code Smell

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

## Instructions should be upper case

Analyze your code

Consistency - Formatted

Maintainability

⬆

 Code Smell

 Major



 convention

Why is this an issue?

How can I fix it?

More Info

While Dockerfile instructions are not case-sensitive, adhering to uppercase conventions for instructions helps enhance clarity and collaboration within development teams. This ensures that instructions are more easily distinguishable from arguments and contributes to effective collaboration.

## What is the potential impact?

In the world of programming, even small details can have significant repercussions. The inconsistent use of uppercase letters in Dockerfile instructions may seem inconsequential at first glance, but it can lead to confusion, misinterpretation, and hinder smooth teamwork.

When Dockerfile instructions are written in a mix of lowercase and uppercase, it becomes challenging to quickly identify and differentiate instructions from arguments. This lack of visual uniformity can make it harder for developers to understand the structure and purpose of the Dockerfile. Teams may spend unnecessary time deciphering the code, which could be better utilized in productive tasks.

Inconsistencies in letter casing can lead to errors that are difficult to diagnose. Imagine spending hours trying to locate a bug only to realize that it resulted from a misinterpretation of an instruction due to inconsistent casing. Such situations can be frustrating and can slow down the development process.

Additionally, modern IDEs provide syntax highlighting to help programmers navigate and comprehend code. However, IDEs can sometimes fail to highlight lowercase or mixed-case Dockerfile keywords properly, as they typically expect instructions to be in uppercase.

In essence, the inconsistent use of uppercase letters in Dockerfile instructions might seem like a minor concern, but its impact reaches beyond aesthetics. It affects comprehension, teamwork, error detection, and the overall development lifecycle.

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](#)

