SOFTWARE VERIFICATION

TEAM [3] PROJECT

CTIP WITH STATIC ANALYSIS

INDEX

REVIEW CTIP

"CONTINUOUS TEST & INTEGRATION PLATFORM"

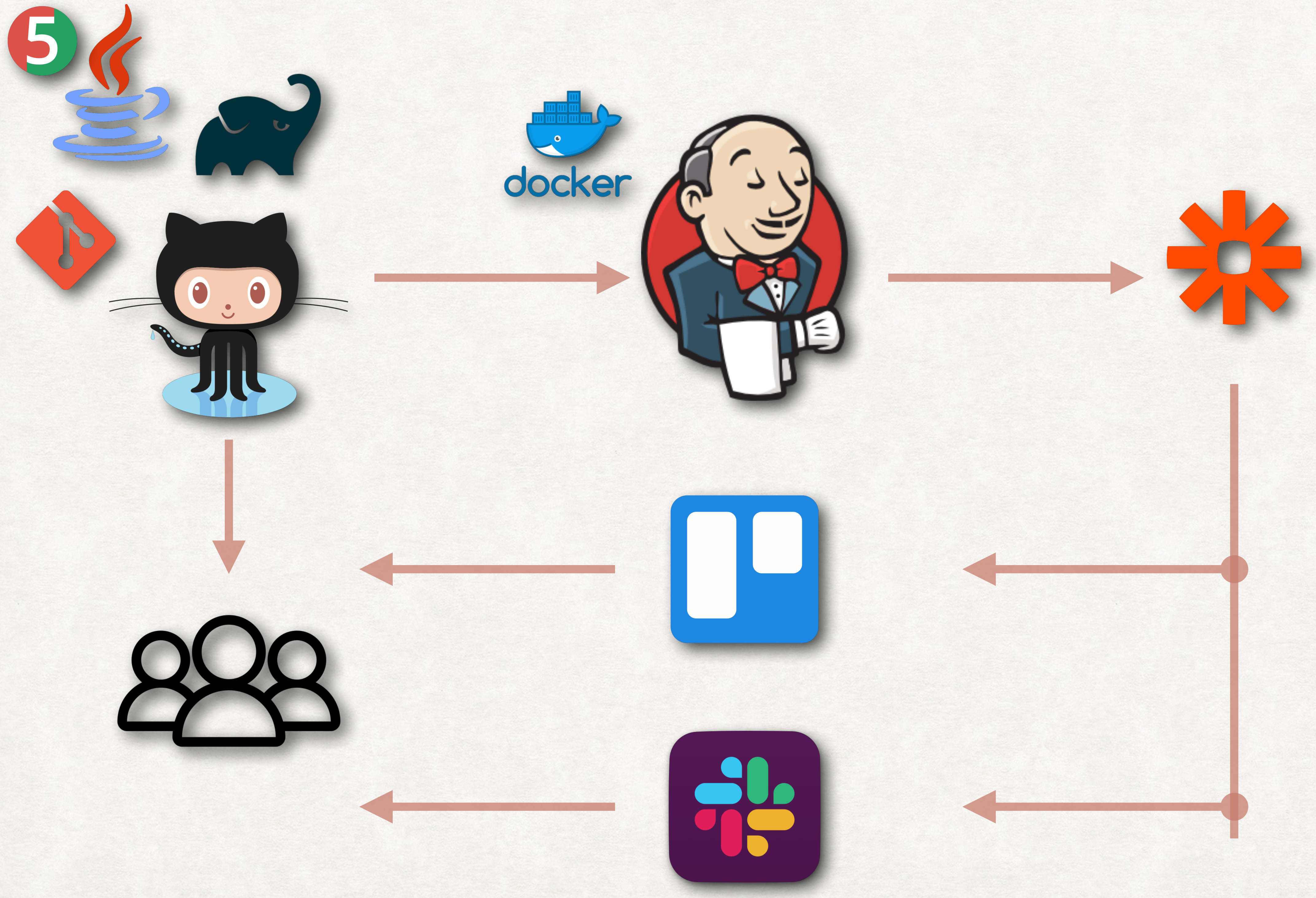
“

WHY DO WE NEED CTIP?

”

CONTINUOUS INTEGRATION

BASED JENKINS



STATIC ANALYSIS TOOL

“WHAT ARE THERE, WHAT ARE THEY DO”

pmd
DON'T SHOOT THE MESSENGER

UNIVERSITY OF MARYLAND
18 56
FindBugs™
because it's easy

checkstyle

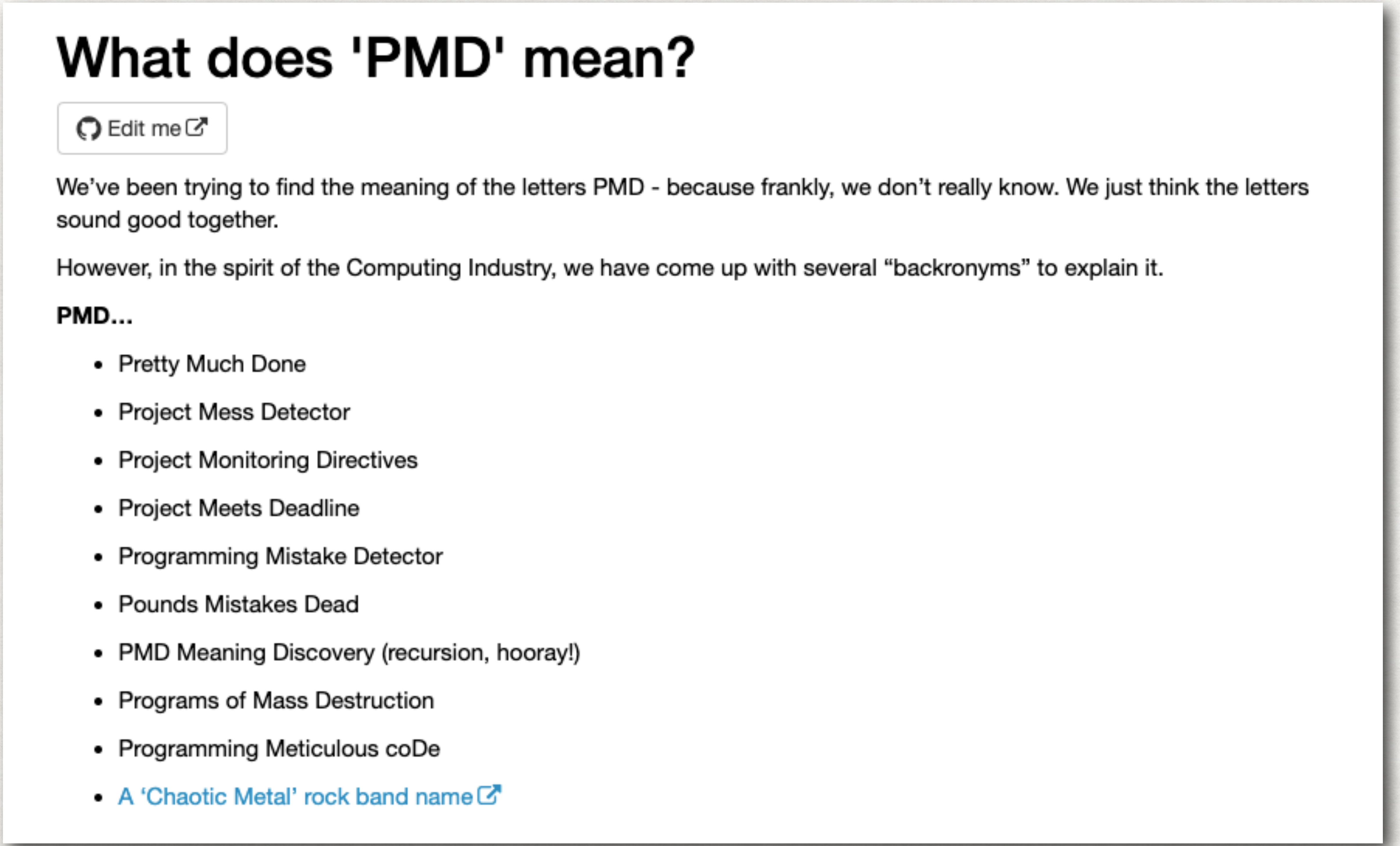
sonarqube

JACOCO
Java Code Coverage

COBERTURA

PMD

STATIC ANALYSIS TOOL



What does 'PMD' mean?

We've been trying to find the meaning of the letters PMD - because frankly, we don't really know. We just think the letters sound good together.

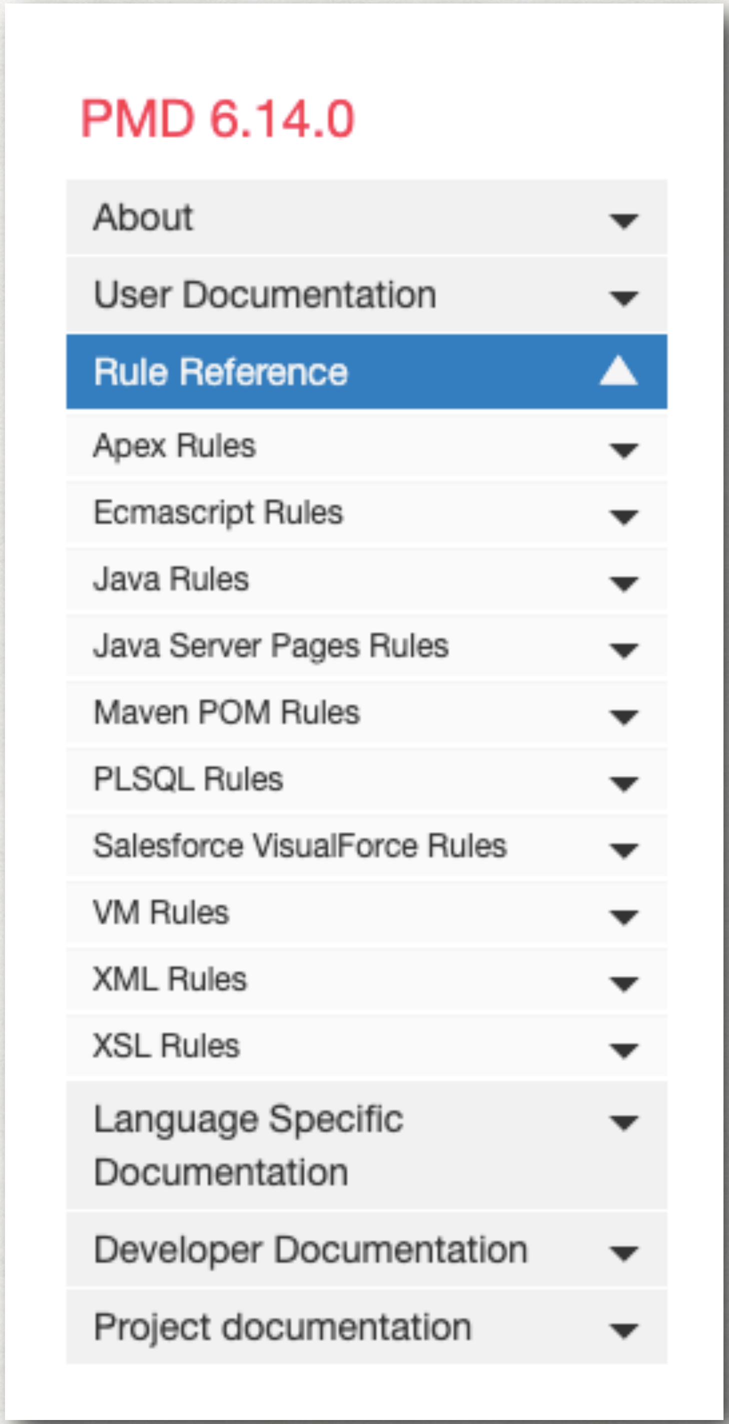
However, in the spirit of the Computing Industry, we have come up with several "backronyms" to explain it.

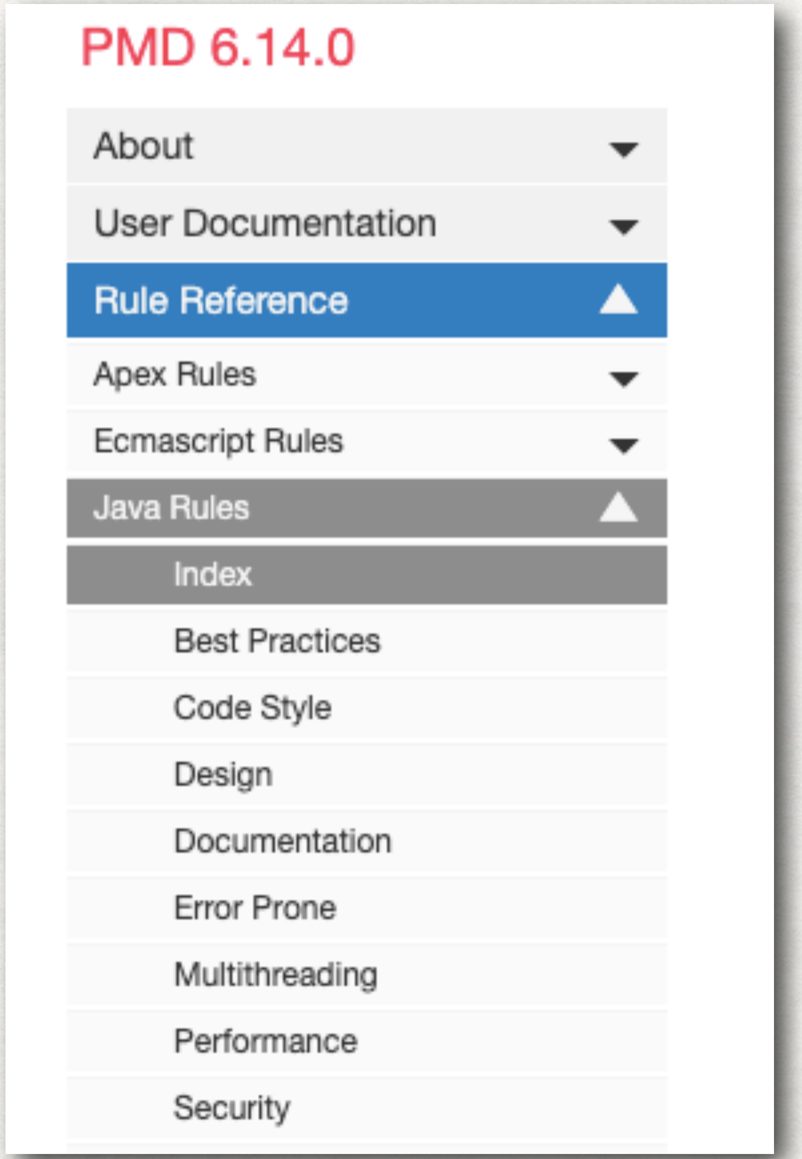
PMD…

- Pretty Much Done
- Project Mess Detector
- Project Monitoring Directives
- Project Meets Deadline
- Programming Mistake Detector
- Pounds Mistakes Dead
- PMD Meaning Discovery (recursion, hooray!)
- Programs of Mass Destruction
- Programming Meticulous coDe
- A 'Chaotic Metal' rock band name

PMD

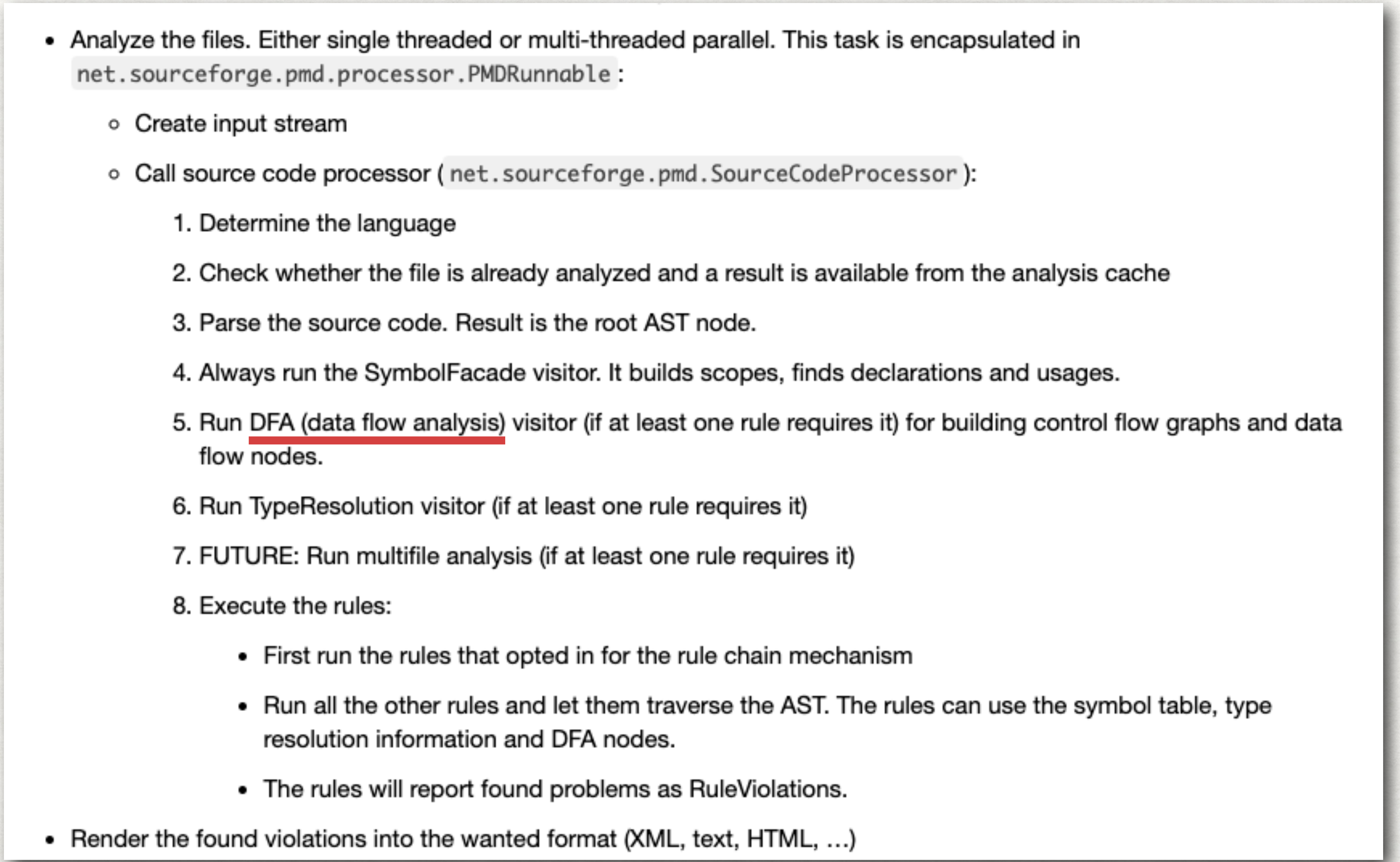
STATIC ANALYSIS TOOL





PMD

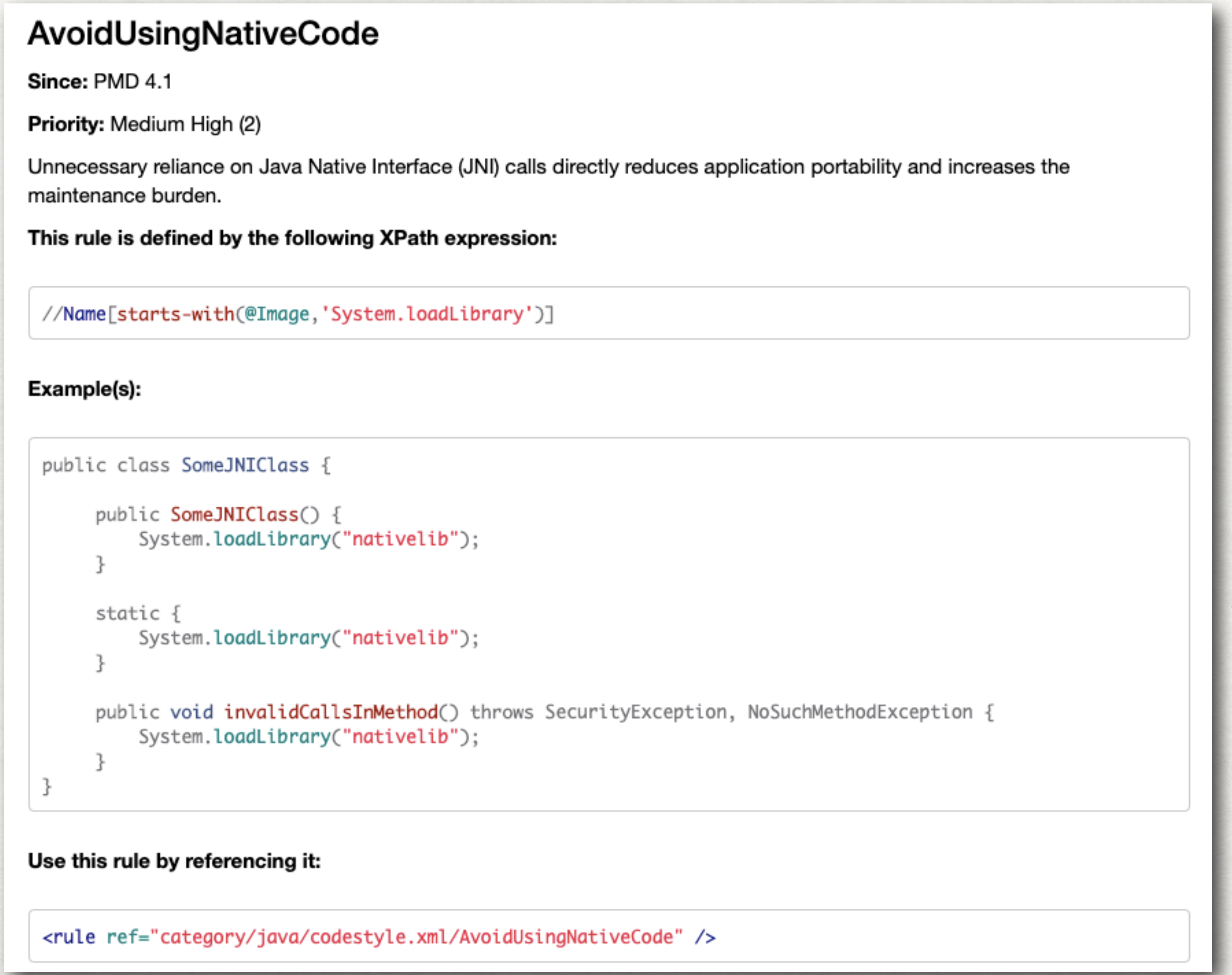
STATIC ANALYSIS TOOL



- Analyze the files. Either single threaded or multi-threaded parallel. This task is encapsulated in `net.sourceforge.pmd.processor.PMDRunnable`:
    - Create input stream
    - Call source code processor (`net.sourceforge.pmd.SourceCodeProcessor`):
        1. Determine the language
        2. Check whether the file is already analyzed and a result is available from the analysis cache
        3. Parse the source code. Result is the root AST node.
        4. Always run the SymbolFacade visitor. It builds scopes, finds declarations and usages.
        5. Run DFA (data flow analysis) visitor (if at least one rule requires it) for building control flow graphs and data flow nodes.
        6. Run TypeResolution visitor (if at least one rule requires it)
        7. FUTURE: Run multifile analysis (if at least one rule requires it)
        8. Execute the rules:
            - First run the rules that opted in for the rule chain mechanism
            - Run all the other rules and let them traverse the AST. The rules can use the symbol table, type resolution information and DFA nodes.
            - The rules will report found problems as RuleViolations.
- Render the found violations into the wanted format (XML, text, HTML, ...)

PMD

STATIC ANALYSIS TOOL



AvoidUsingNativeCode

Since: PMD 4.1

Priority: Medium High (2)

Unnecessary reliance on Java Native Interface (JNI) calls directly reduces application portability and increases the maintenance burden.

This rule is defined by the following XPath expression:

```
//Name[starts-with(@Image,'System.loadLibrary')]
```

Example(s):

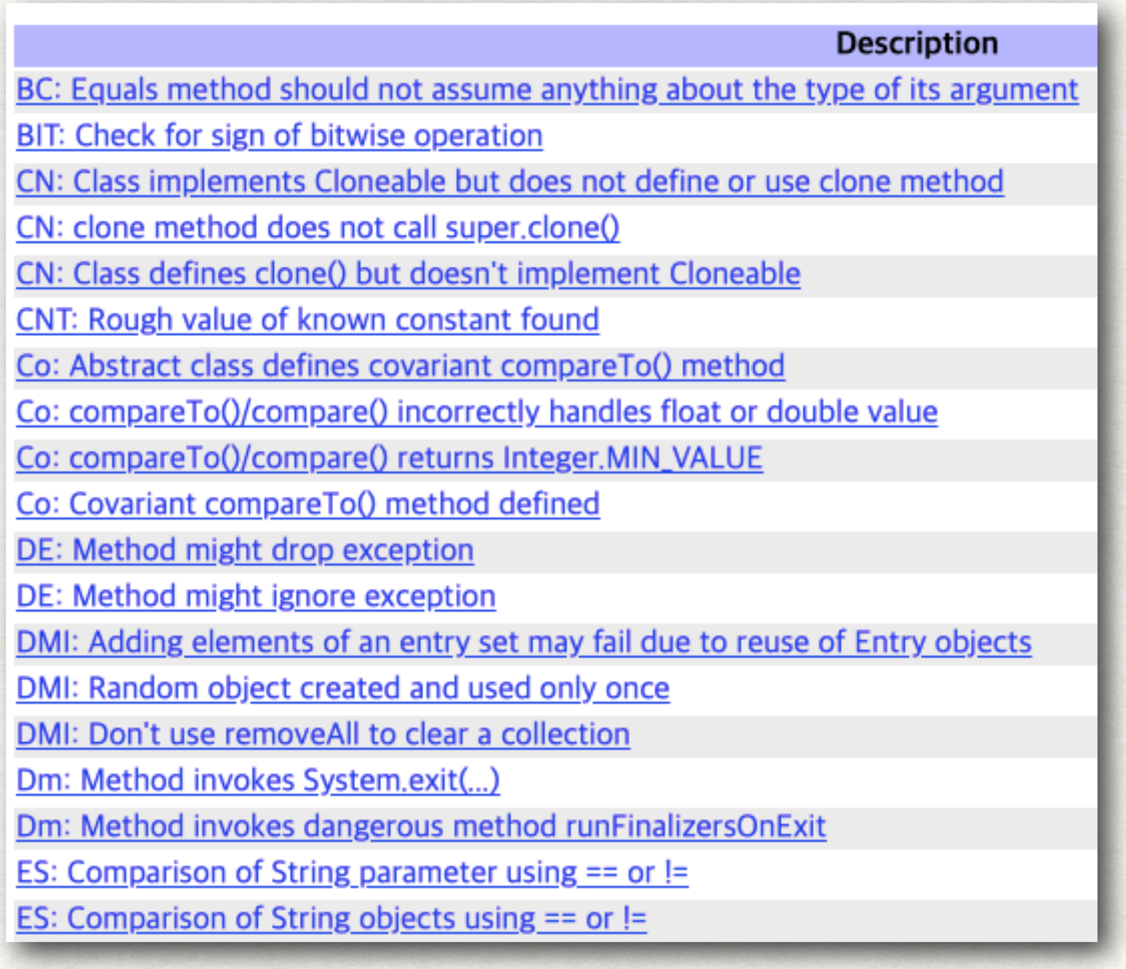
```
public class SomeJNIClass {

     public SomeJNIClass() {
         System.loadLibrary("nativelib");
     }

     static {
         System.loadLibrary("nativelib");
     }

     public void invalidCallsInMethod() throws SecurityException, NoSuchMethodException {
         System.loadLibrary("nativelib");
     }
}
```

Use this rule by referencing it:

```
<rule ref="category/java/codestyle.xml/AvoidUsingNativeCode" />
```

FINDBUGS

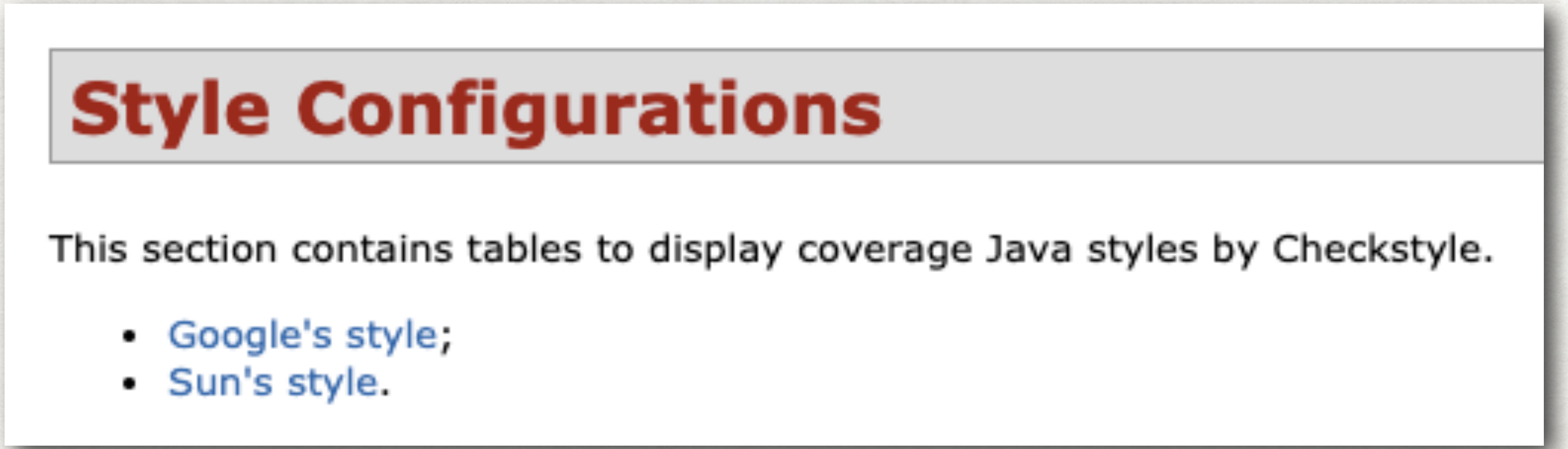
STATIC ANALYSIS TOOL



Description
BC: Equals method should not assume anything about the type of its argument
BIT: Check for sign of bitwise operation
CN: Class implements Cloneable but does not define or use clone method
CN: clone method does not call super.clone()
CN: Class defines clone() but doesn't implement Cloneable
CNT: Rough value of known constant found
Co: Abstract class defines covariant compareTo() method
Co: compareTo()/compare() incorrectly handles float or double value
Co: compareTo()/compare() returns Integer.MIN_VALUE
Co: Covariant compareTo() method defined
DE: Method might drop exception
DE: Method might ignore exception
DMI: Adding elements of an entry set may fail due to reuse of Entry objects
DMI: Random object created and used only once
DMI: Don't use removeAll to clear a collection
Dm: Method invokes System.exit(...)
Dm: Method invokes dangerous method runFinalizersOnExit
ES: Comparison of String parameter using == or !=
ES: Comparison of String objects using == or !=

CHECKSTYLE

STATIC ANALYSIS TOOL



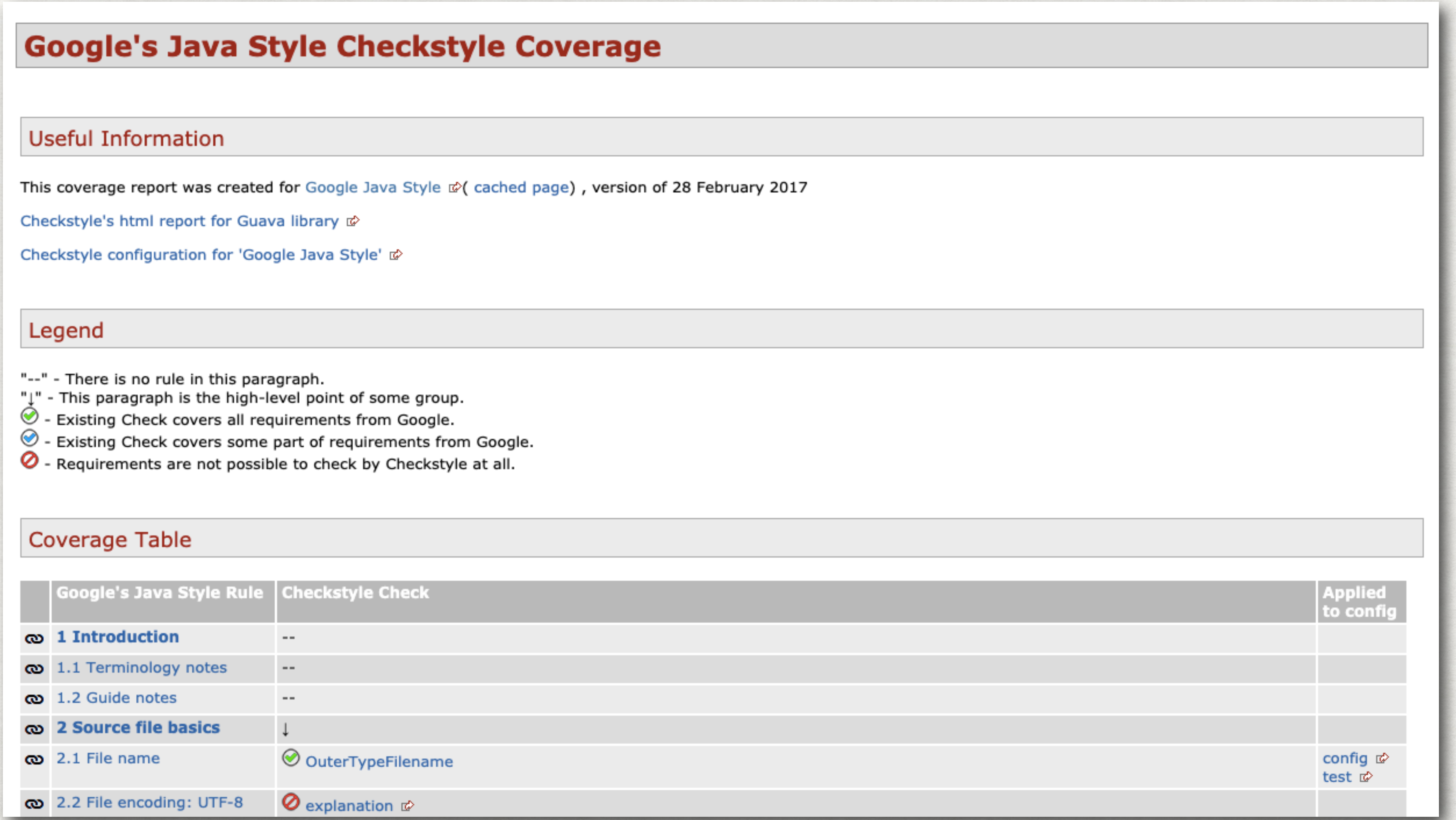
Style Configurations

This section contains tables to display coverage Java styles by Checkstyle.

- Google's style;
- Sun's style.

CHECKSTYLE

STATIC ANALYSIS TOOL



Google's Java Style Checkstyle Coverage

Useful Information

This coverage report was created for Google Java Style (cached page) , version of 28 February 2017

Checkstyle's html report for Guava library

Checkstyle configuration for 'Google Java Style'

Legend

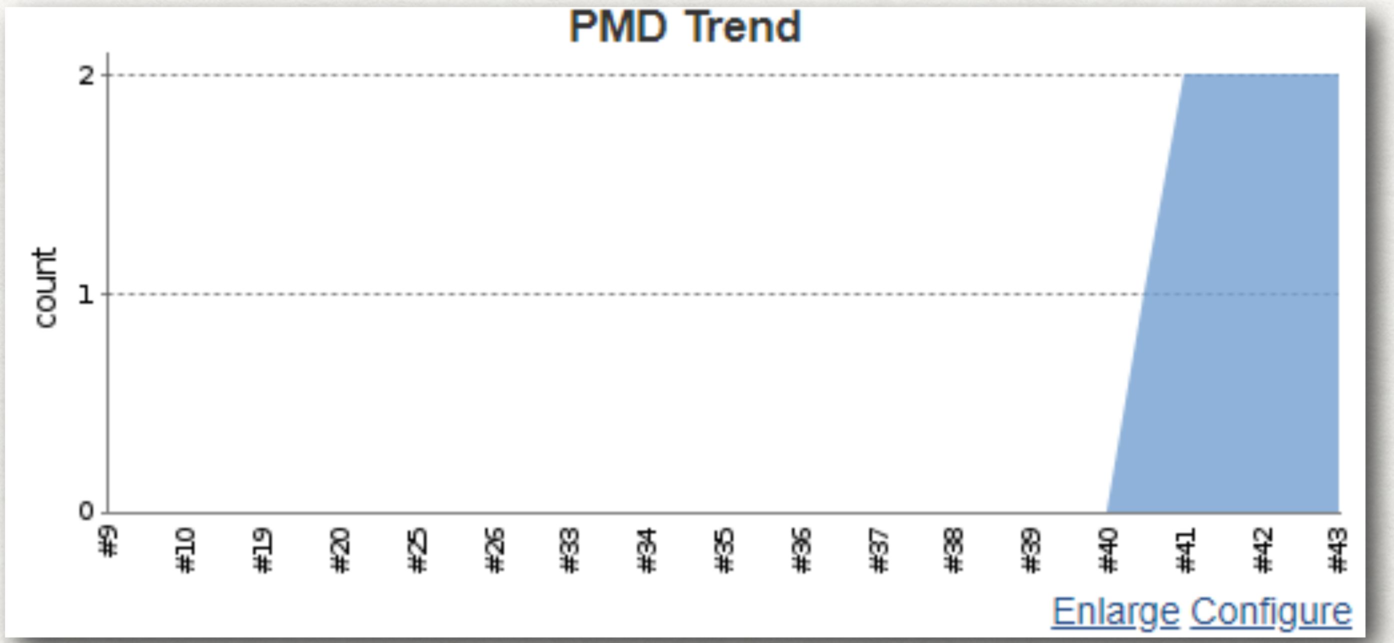
"--" - There is no rule in this paragraph.
"↓" - This paragraph is the high-level point of some group.
✓ - Existing Check covers all requirements from Google.
✓ - Existing Check covers some part of requirements from Google.
⊘ - Requirements are not possible to check by Checkstyle at all.

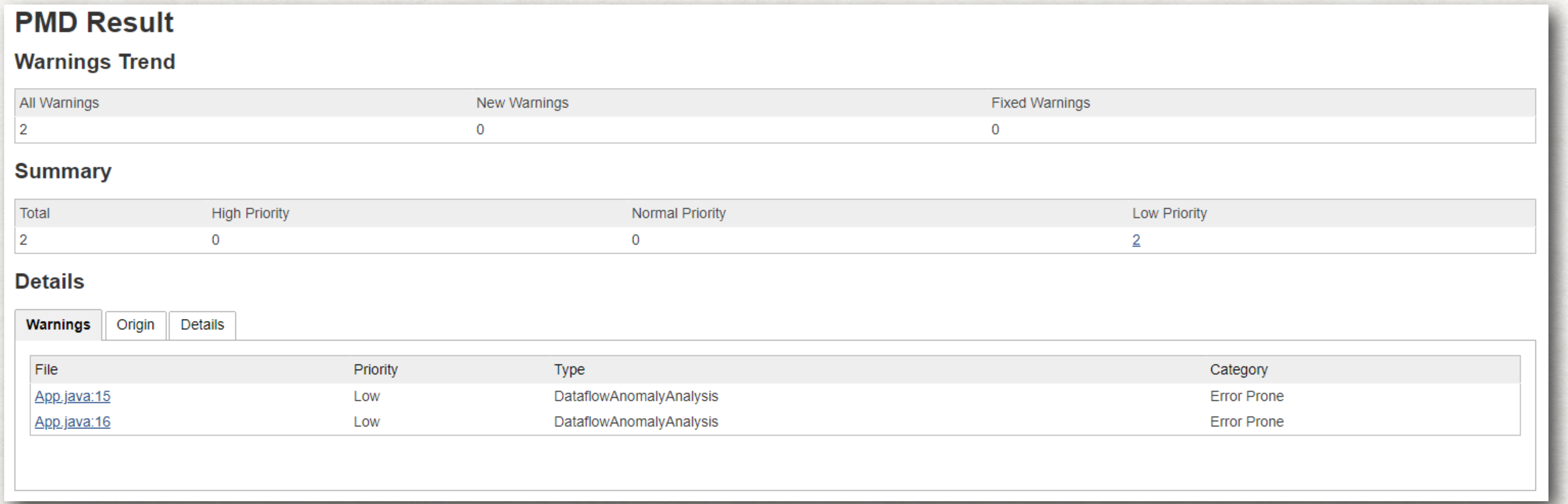
Coverage Table

	Google's Java Style Rule	Checkstyle Check	Applied to config
	1 Introduction	--	
	1.1 Terminology notes	--	
	1.2 Guide notes	--	
	2 Source file basics	↓	
	2.1 File name	✓ OuterTypeFilename	config test
	2.2 File encoding: UTF-8	⊘ explanation	

PMD

STATIC ANALYSIS TOOL





PMD Result

Warnings Trend

All Warnings	New Warnings	Fixed Warnings
2	0	0

Summary

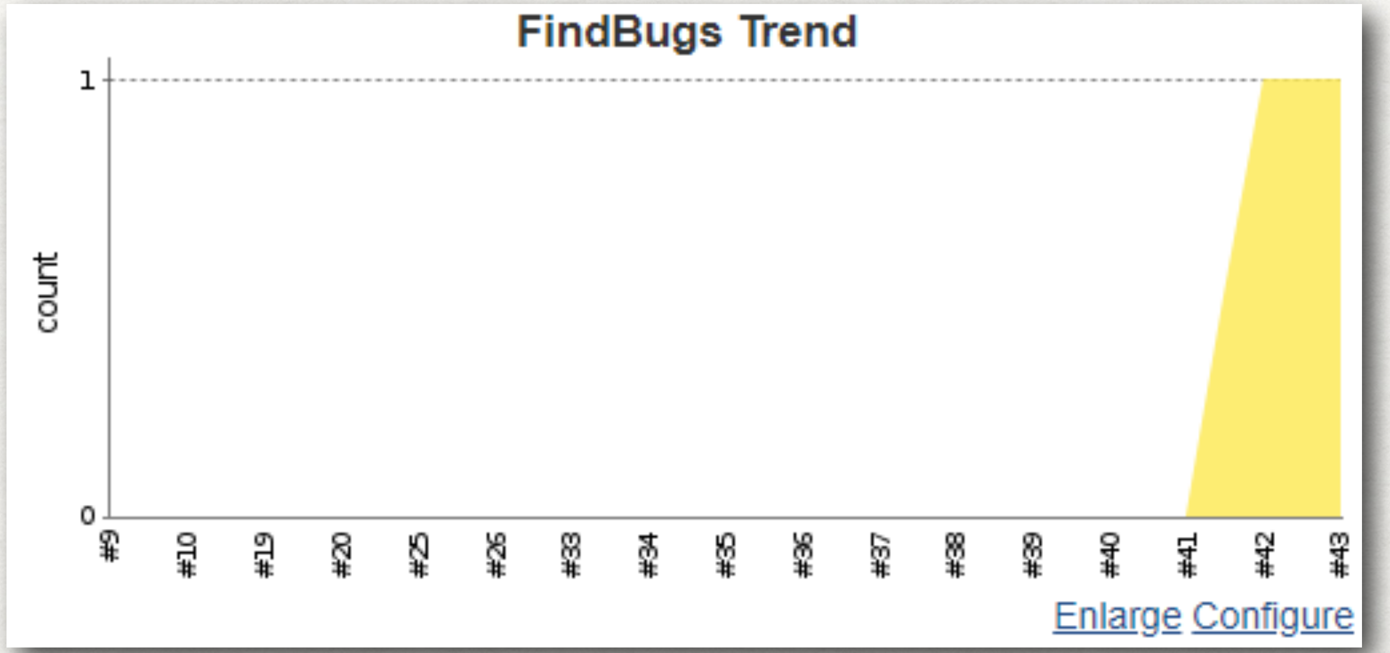
Total	High Priority	Normal Priority	Low Priority
2	0	0	2

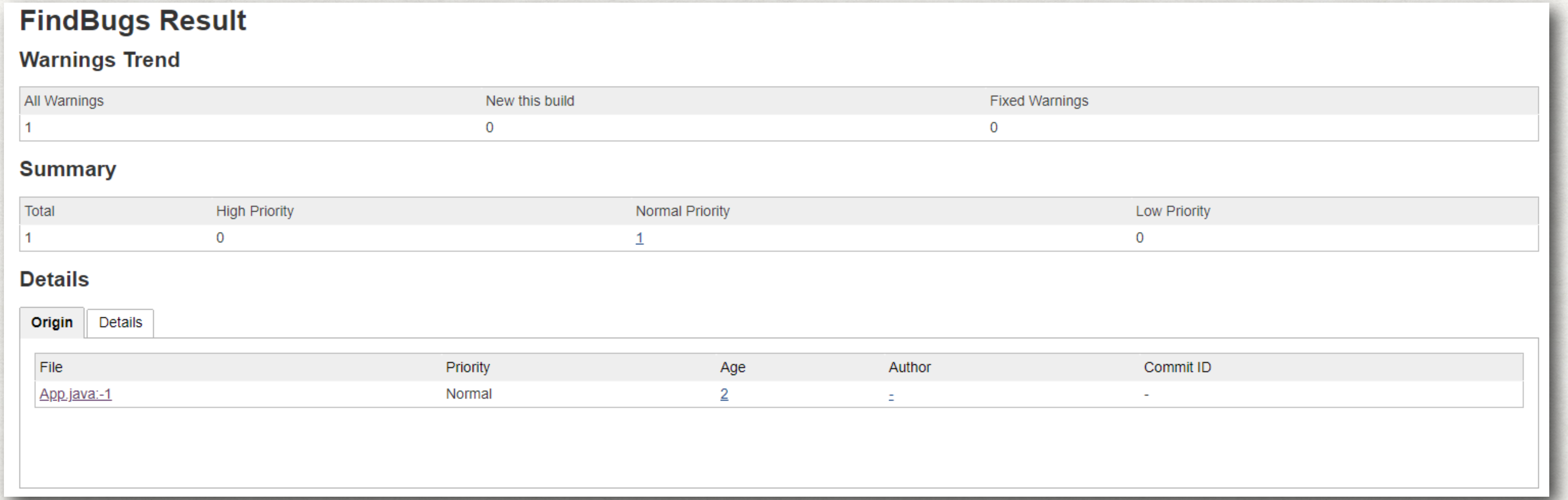
Details

File	Priority	Type	Category
App.java:15	Low	DataflowAnomalyAnalysis	Error Prone
App.java:16	Low	DataflowAnomalyAnalysis	Error Prone

FINDBUGS

STATIC ANALYSIS TOOL





FindBugs Result

Warnings Trend

All Warnings	New this build	Fixed Warnings
1	0	0

Summary

Total	High Priority	Normal Priority	Low Priority
1	0	1	0

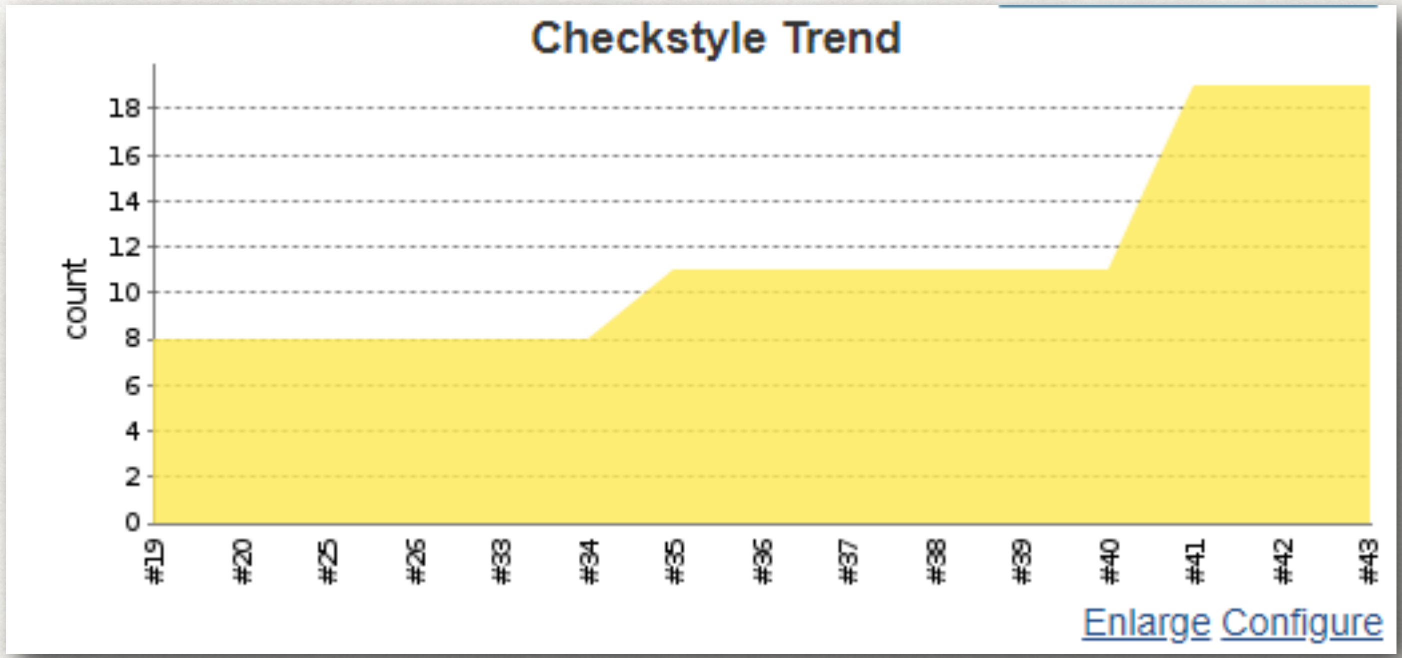
Details

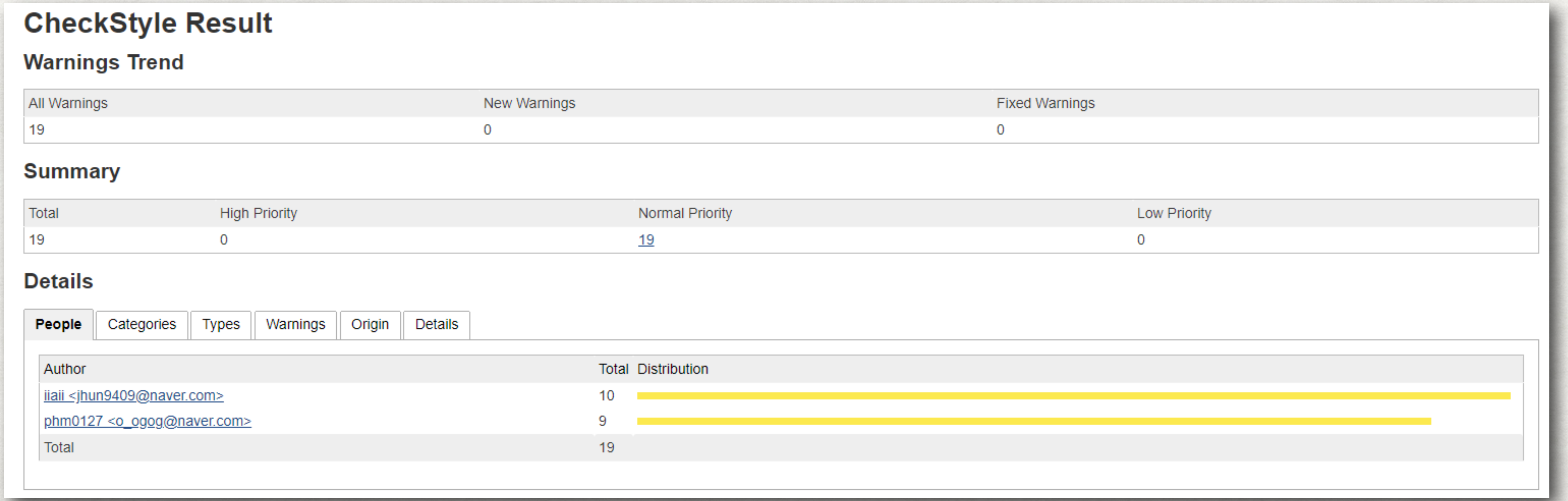
Origin Details

File	Priority	Age	Author	Commit ID
App.java:-1	Normal	2	-	-

CHECKSTYLE

STATIC ANALYSIS TOOL





CheckStyle Result

Warnings Trend

All Warnings	New Warnings	Fixed Warnings
19	0	0

Summary

Total	High Priority	Normal Priority	Low Priority
19	0	19	0

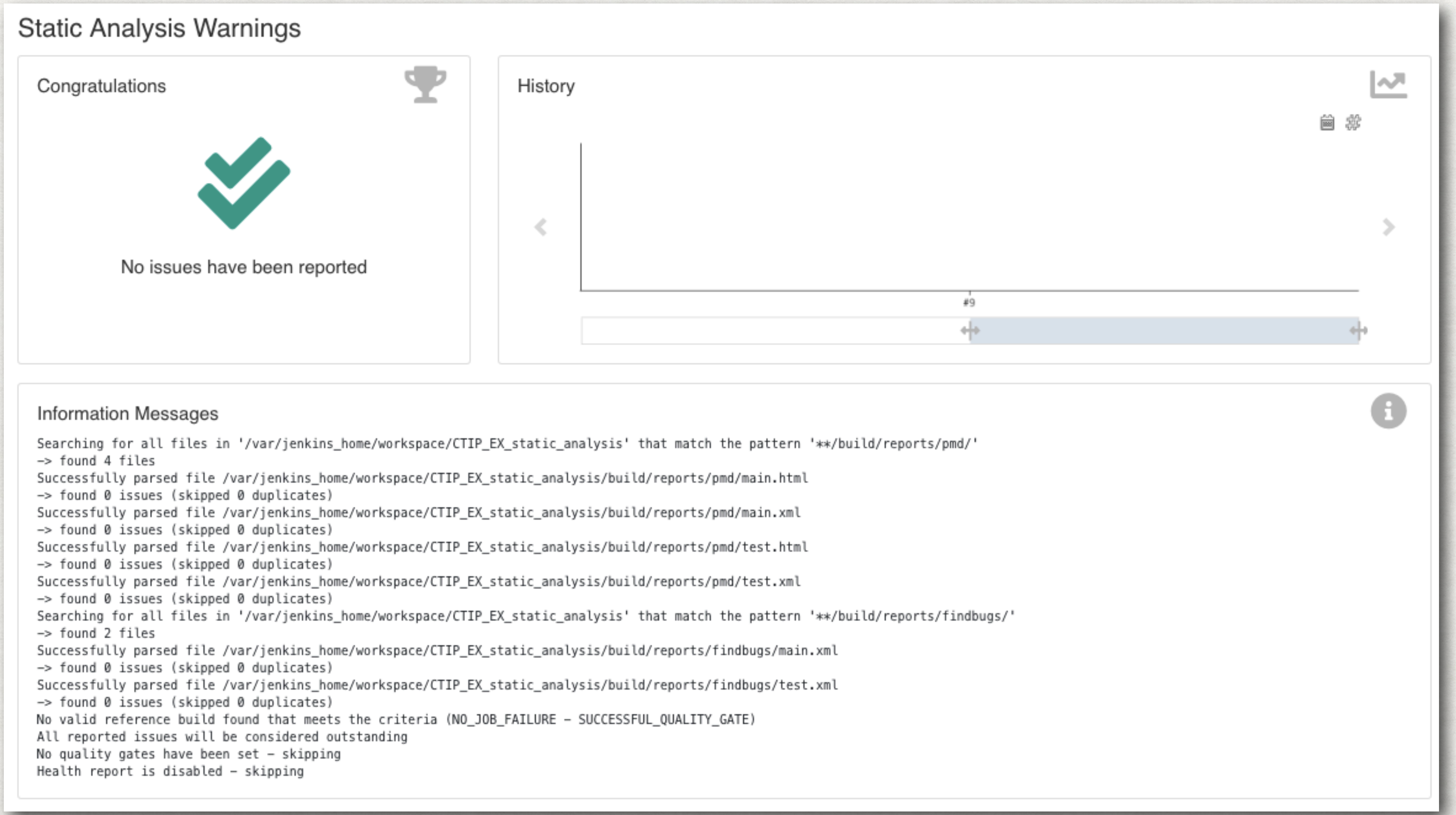
Details

People Categories Types Warnings Origin Details

Author	Total	Distribution
iiaii <jhun9409@naver.com>	10	
phm0127 <o_ogog@naver.com>	9	
Total	19	

CHECKSTYLE

STATIC ANALYSIS TOOL

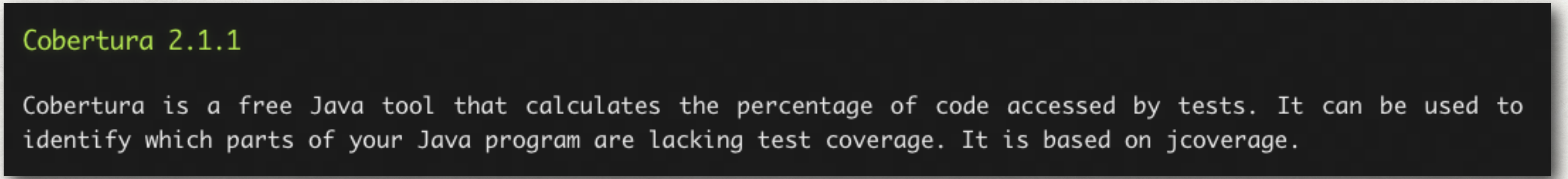


```
Searching for all files in '/var/jenkins_home/workspace/CTIP_EX_static_analysis' that match the pattern '**/build/reports/pmd/'
-> found 4 files
Successfully parsed file /var/jenkins_home/workspace/CTIP_EX_static_analysis/build/reports/pmd/main.html
-> found 0 issues (skipped 0 duplicates)
Successfully parsed file /var/jenkins_home/workspace/CTIP_EX_static_analysis/build/reports/pmd/main.xml
-> found 0 issues (skipped 0 duplicates)
Successfully parsed file /var/jenkins_home/workspace/CTIP_EX_static_analysis/build/reports/pmd/test.html
-> found 0 issues (skipped 0 duplicates)
Successfully parsed file /var/jenkins_home/workspace/CTIP_EX_static_analysis/build/reports/pmd/test.xml
-> found 0 issues (skipped 0 duplicates)
Searching for all files in '/var/jenkins_home/workspace/CTIP_EX_static_analysis' that match the pattern '**/build/reports/findbugs/'
-> found 2 files
Successfully parsed file /var/jenkins_home/workspace/CTIP_EX_static_analysis/build/reports/findbugs/main.xml
-> found 0 issues (skipped 0 duplicates)
Successfully parsed file /var/jenkins_home/workspace/CTIP_EX_static_analysis/build/reports/findbugs/test.xml
-> found 0 issues (skipped 0 duplicates)
No valid reference build found that meets the criteria (NO_JOB_FAILURE - SUCCESSFUL_QUALITY_GATE)
All reported issues will be considered outstanding
No quality gates have been set - skipping
Health report is disabled - skipping
```

COBERTURA

STATIC ANALYSIS TOOL

COBERTURA

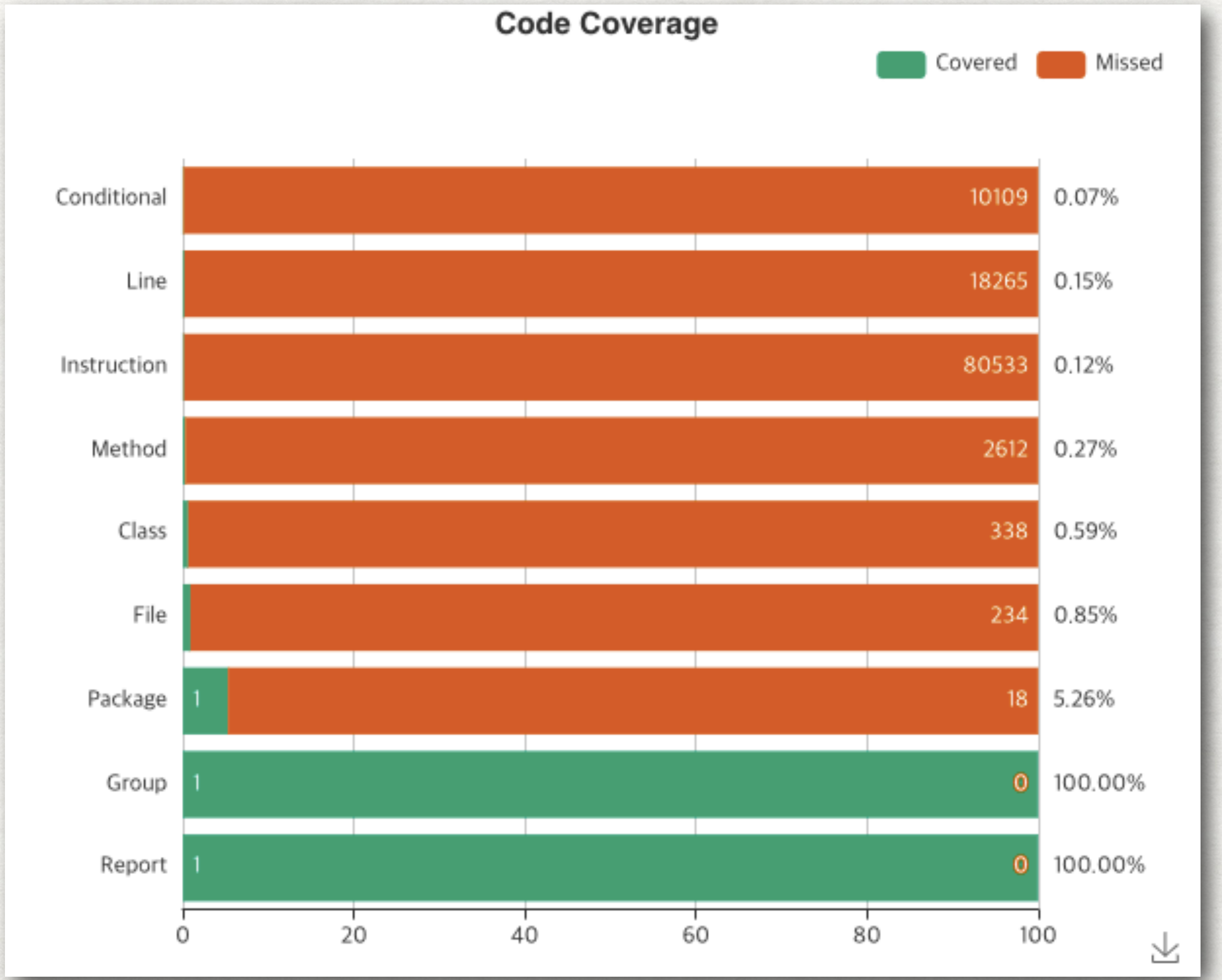


Cobertura 2.1.1

Cobertura is a free Java tool that calculates the percentage of code accessed by tests. It can be used to identify which parts of your Java program are lacking test coverage. It is based on jcoverage.

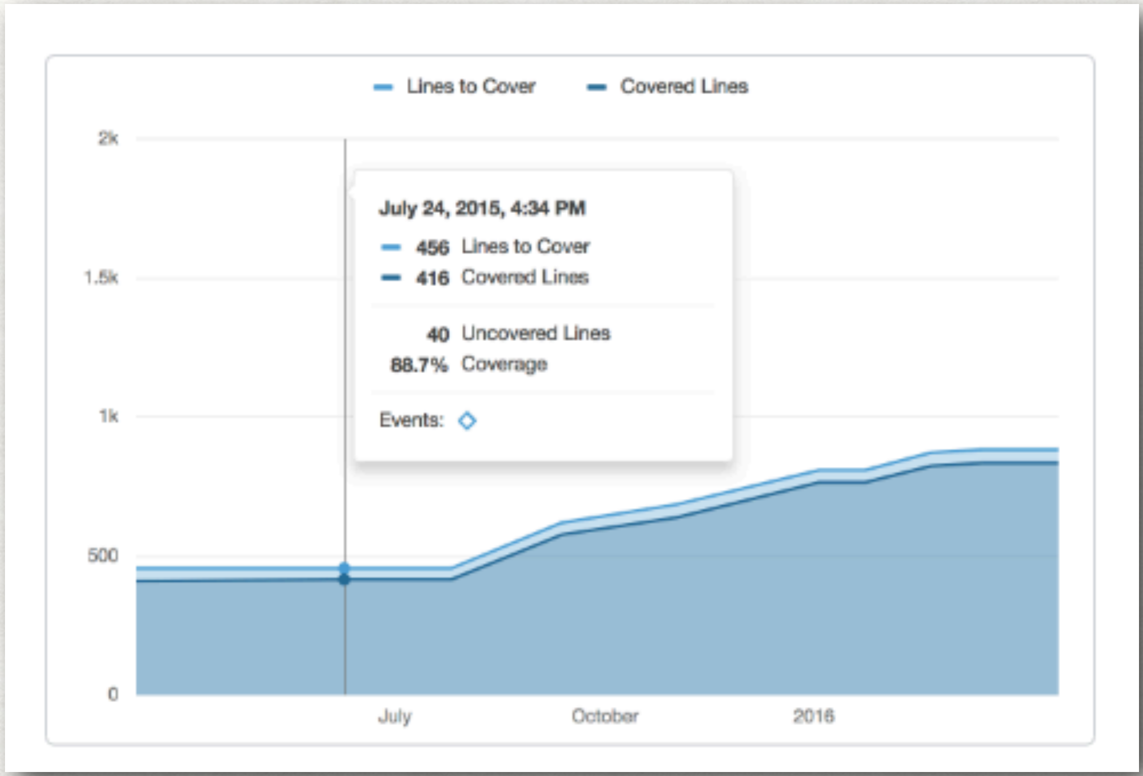
JACOCO

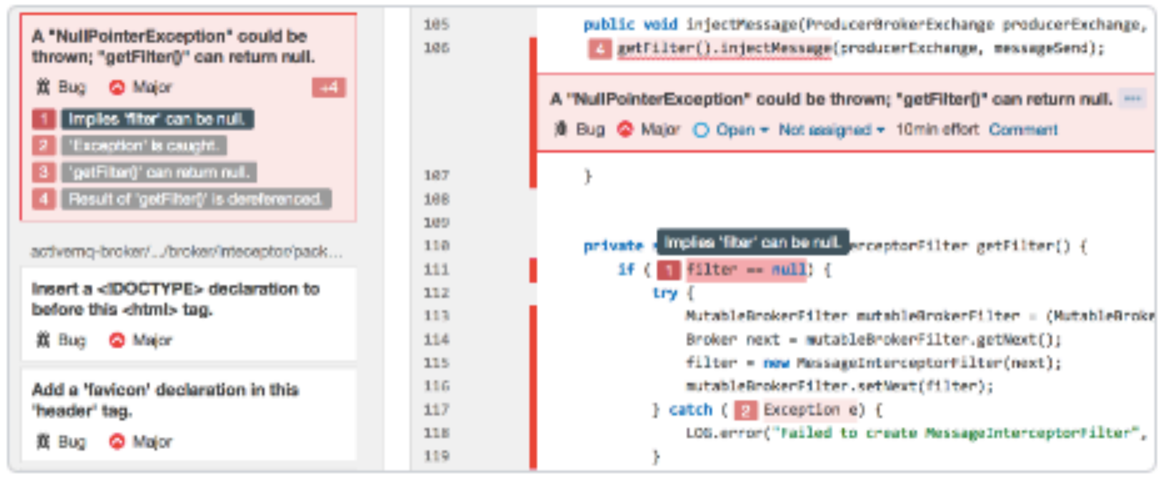
STATIC ANALYSIS TOOL



SONARQUBE

STATIC ANALYSIS TOOL





A "NullPointerException" could be thrown; "getFilter()" can return null.
Bug Major +4
1 Implies 'filter' can be null.
2 'Exception' is caught.
3 'getFilter()' can return null.
4 Result of 'getFilter()' is dereferenced.

activemq-broker/.../broker/inteceptor/pack...

Insert a <!DOCTYPE> declaration to before this <html> tag.
Bug Major

Add a 'favicon' declaration in this 'header' tag.
Bug Major

```
    public void injectMessage(ProducerBrokerExchange producerExchange,
     4 getFilter().injectMessage(producerExchange, messageSend);
```

A "NullPointerException" could be thrown; "getFilter()" can return null.
Bug Major Open Not assigned 10min effort Comment

```
    }


    private  Implies 'filter' can be null. rceptorFilter getFilter() {
        if ( 1 filter == null) {
            try {
                MutableBrokerFilter mutableBrokerFilter = (MutableBroke
                Broker next = mutableBrokerFilter.getNext();
                filter = new MessageInterceptorFilter(next);
                mutableBrokerFilter.setNext(filter);
            } catch ( 2 Exception e) {
                LOG.error("Failed to create MessageInterceptorFilter",
            }
```

Detect Bugs

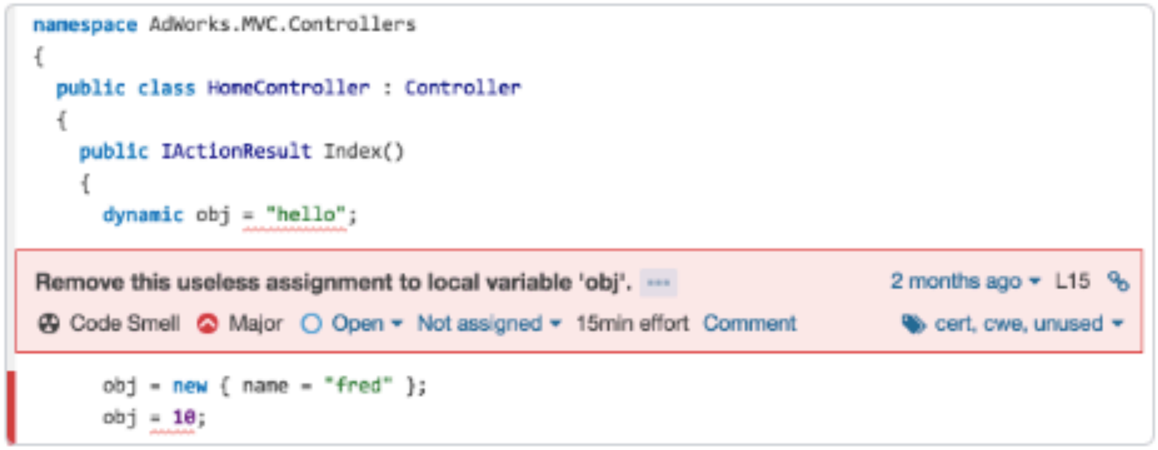
Issues raised by SonarQube are on either demonstrably wrong code, or code that is more likely not giving the intended behavior. Find trickiest bugs navigating easily through the code paths while pointing out issues found in multiple locations.

SONARQUBE

STATIC ANALYSIS TOOL

Code Smells

"Smelly" code does (probably) what it should, but it will be difficult to maintain. In the worst cases, it will be so confusing that maintainers can inadvertently introduce bugs. Examples include duplicated code, uncovered code by unit tests and too complex code.



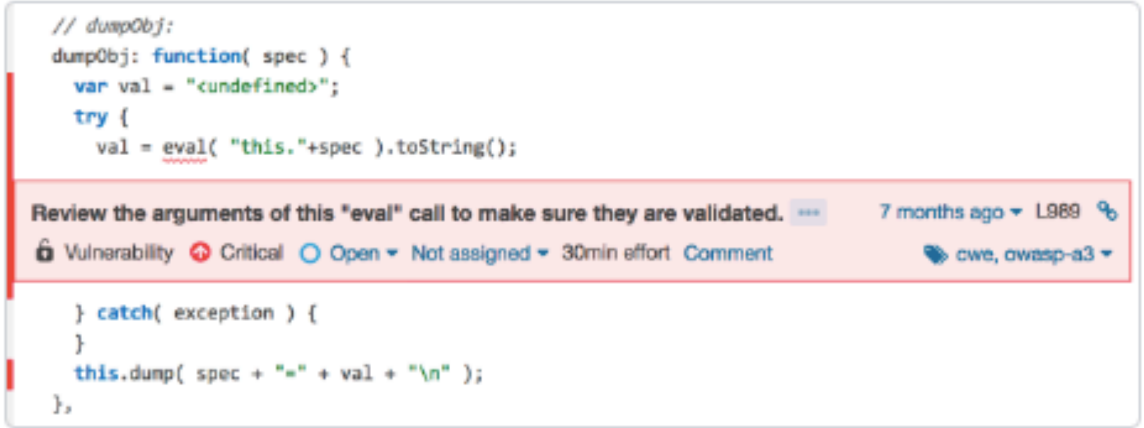
```
namespace AdWorks.MVC.Controllers
{
  public class HomeController : Controller
  {
    public IActionResult Index()
    {
      dynamic obj = "hello";
```

Remove this useless assignment to local variable 'obj'. ··· 2 months ago ▾ L15

Code Smell Major Open ▾ Not assigned ▾ 15min effort Comment cert, cwe, unused ▾

```
      obj = new { name = "fred" };
      obj = 10;
```

Security Vulnerability



```
// dumpObj:
dumpObj: function( spec ) {
  var val = "<undefined>";
  try {
    val = eval( "this."+spec ).toString();
```

Review the arguments of this "eval" call to make sure they are validated. ··· 7 months ago ▾ L989

Vulnerability Critical Open ▾ Not assigned ▾ 30min effort Comment cwe, owasp-a3 ▾

```
  } catch( exception ) {
  }
  this.dump( spec + "=" + val + "\n" );
},
```

It's probably Pollyanna-ish to think you'll never be targeted by hackers. When you are, what vulnerabilities will they find in your system? SonarQube helps you find and track the insecurities in your code. Examples include SQL injection, hard-coded passwords and badly managed errors.

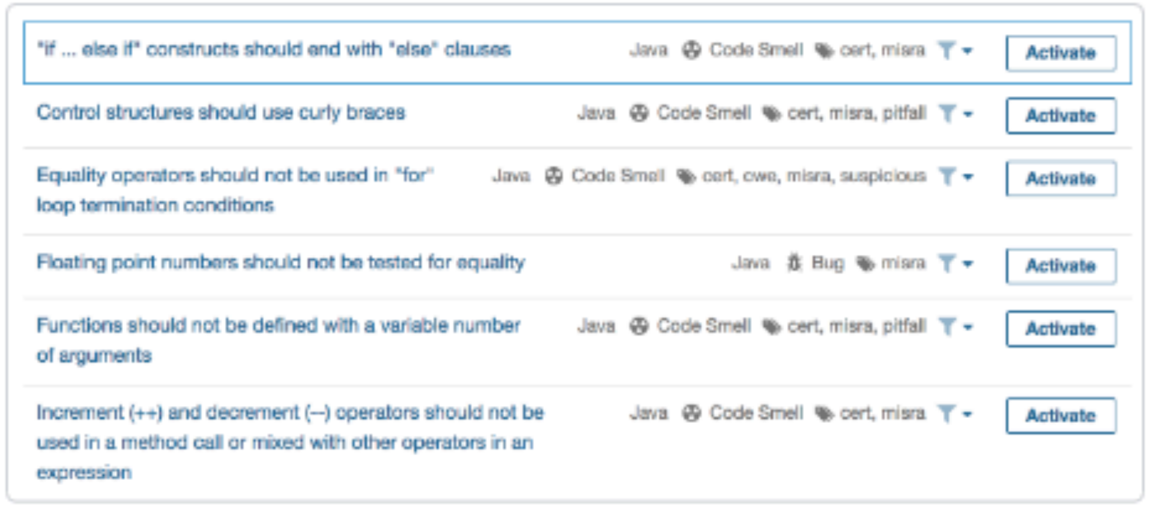
SONARQUBE

STATIC ANALYSIS TOOL

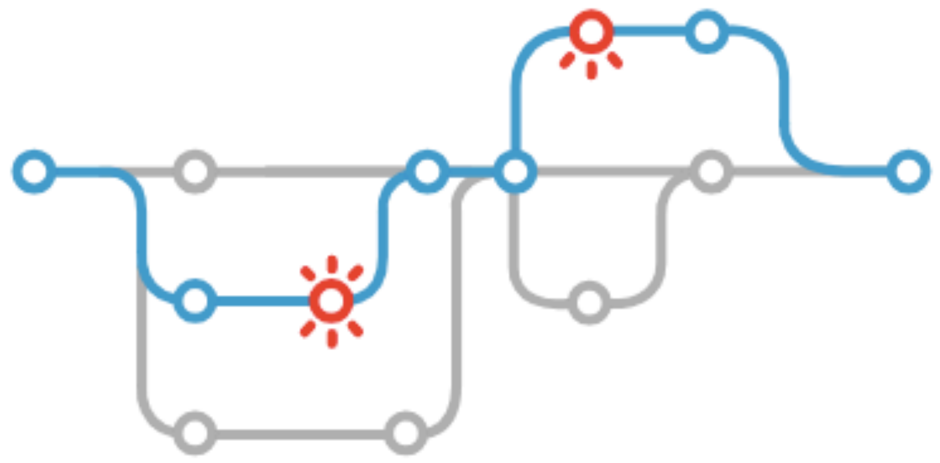
Activate The Rules You Need

SonarQube code analyzers include default Quality Profiles that offer strong value with non-controversial rule sets. The default Quality Profiles will work for most projects, but you can easily tune them to fully match your needs.

The rules page enables to find rules by multiple criteria, alone or in combination. From the search results you can activate or deactivate rules in your Quality Profile.



"if ... else if" constructs should end with "else" clauses	Java	Code Smell	cert, misra	Activate
Control structures should use curly braces	Java	Code Smell	cert, misra, pitfall	Activate
Equality operators should not be used in "for" loop termination conditions	Java	Code Smell	cert, cwe, misra, suspicious	Activate
Floating point numbers should not be tested for equality	Java	Bug	misra	Activate
Functions should not be defined with a variable number of arguments	Java	Code Smell	cert, misra, pitfall	Activate
Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression	Java	Code Smell	cert, misra	Activate



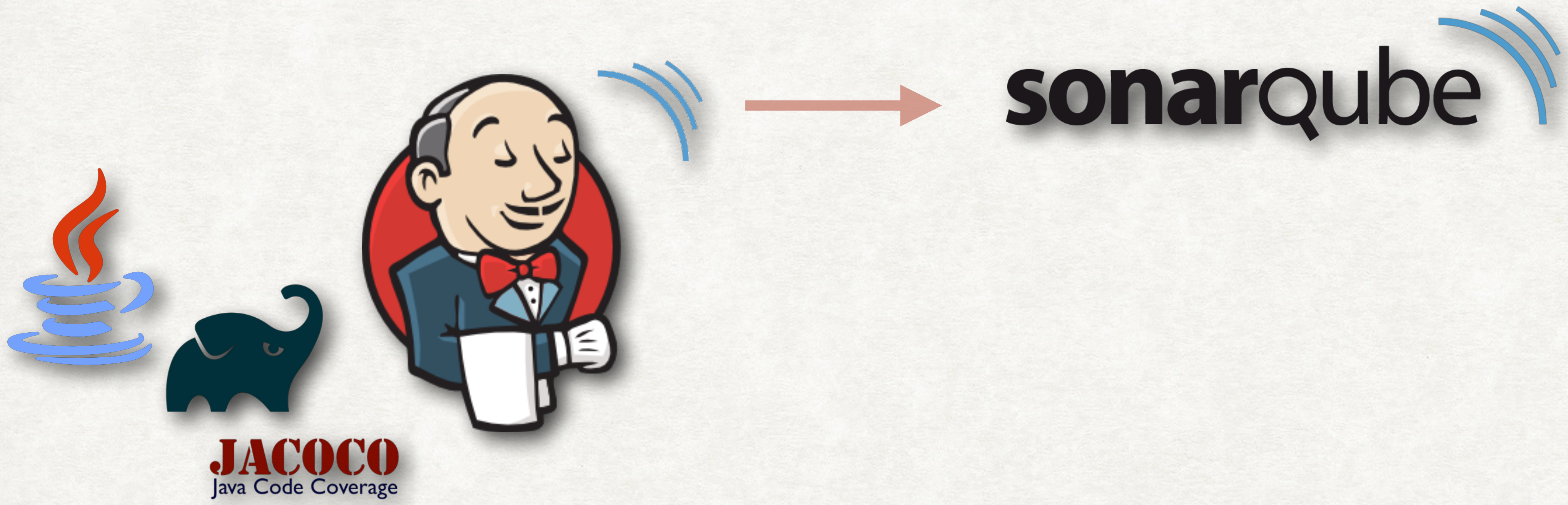
Explore All Execution Paths

SonarQube relies on several path-sensitive dataflow engines and thus code analyzers explore all possible execution paths to spot the trickiest bugs.

Even a simple function containing only 10 different branches might lead to 100 different possible execution paths at runtime. Manually checking that those 100 execution paths are error proof is simply impossible.

SONARQUBE

STATIC ANALYSIS TOOL

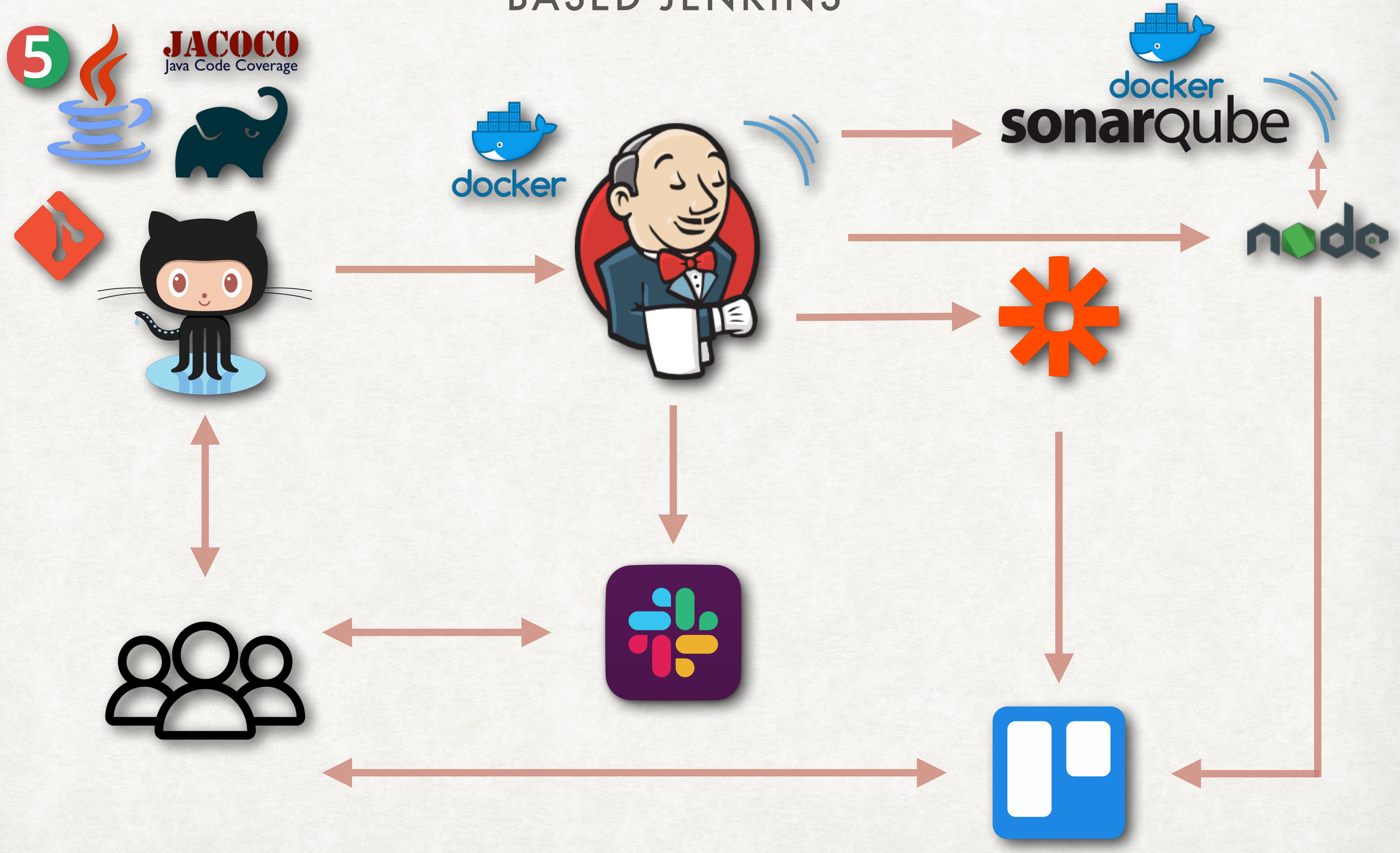


CTIP WITH SAT

"CTIP WITH SONARQUBE"

CONTINUOUS INTEGRATION

BASED JENKINS



“

DEMONSTRATION

”

Q&A

“EVERYTHING”

THANKS