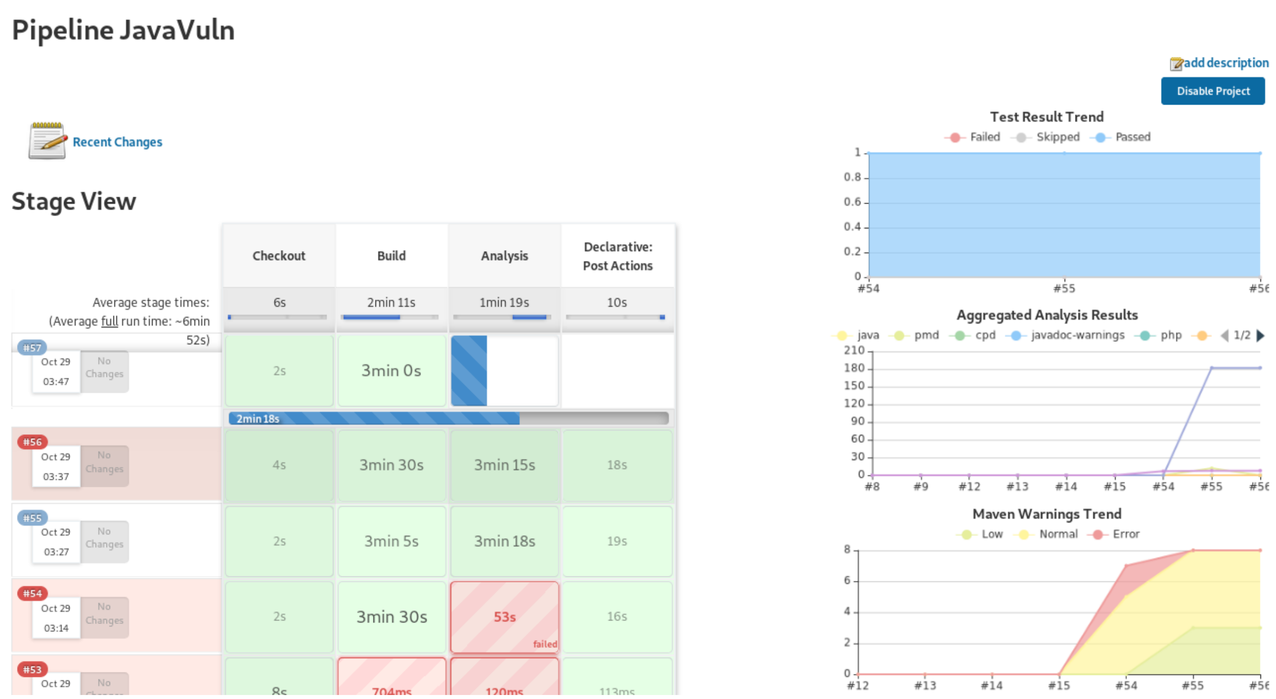
Jenkins Warnings Next Generation Plugin

Overview



Pipeline JavaVuln

Stage View

	Checkout	Build	Analysis	Declarative: Post Actions
Average stage times: (Average full run time: ~6min 52s)	6s	2min 11s	1min 19s	10s
#57 Oct 29 03:47 No Changes	2s	3min 0s		
	2min 18s			
#56 Oct 29 03:37 No Changes	4s	3min 30s	3min 15s	18s
#55 Oct 29 03:27 No Changes	2s	3min 5s	3min 18s	19s
#54 Oct 29 03:14 No Changes	2s	3min 30s	53s failed	16s
#53 Oct 29 No Changes	8s	704ms	120ms	113ms

In this lab, you are going to learn how to use Jenkins Warnings Next Generation Plugin to generate static code analysis report.

The Jenkins Warnings Next Generation Plugin collects compiler warnings or issues reported by static analysis tools and visualizes the results. It has built-in support for numerous static analysis tools (including several compilers),

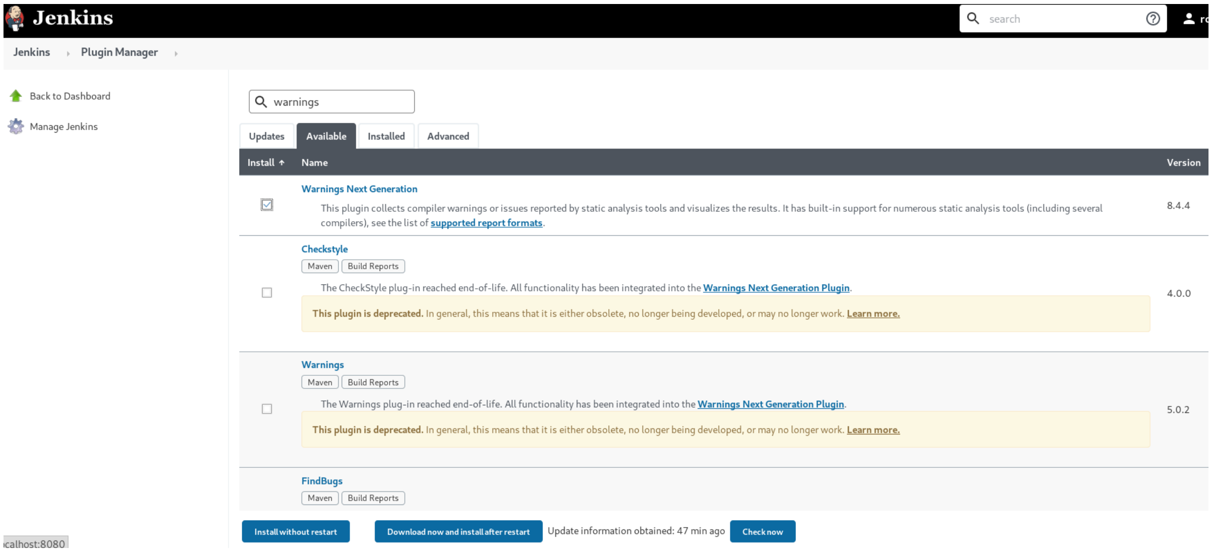
Outcomes

Upon completion of this session, you should be able to

- Use Jenkins Warnings Next Generation Plugin to analysis source code
- Select the suitable SAST for your team project
- Start incorporating Jenkins Pipeline with Static Code Analysis into your team project

1: Installation

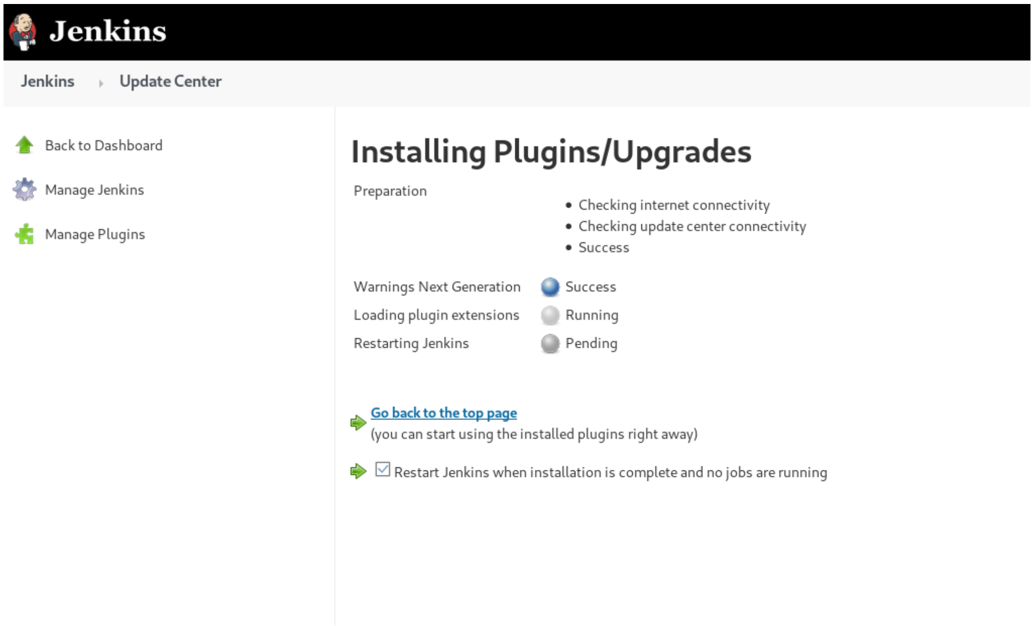
This lab is based on the instruction https://github.com/jenkinsci/warnings-ng-plugin/blob/master/doc/Documentation.md and the Vulnado - Intentionally Vulnerable Java Application https://github.com/ScaleSec/vulnado , but it also requires different docker SAST image / software to be installed before you can incorporate Jenkins Pipeline.



Install ↑	Name	Version
☑	**Warnings Next Generation** This plugin collects compiler warnings or issues reported by static analysis tools and visualizes the results. It has built-in support for numerous static analysis tools (including several compilers), see the list of supported report formats.	8.4.4
□	**Checkstyle** Maven Build Reports The CheckStyle plug-in reached end-of-life. All functionality has been integrated into the Warnings Next Generation Plugin. **This plugin is deprecated.** In general, this means that it is either obsolete, no longer being developed, or may no longer work. Learn more.	4.0.0
□	**Warnings** Maven Build Reports The Warnings plug-in reached end-of-life. All functionality has been integrated into the Warnings Next Generation Plugin. **This plugin is deprecated.** In general, this means that it is either obsolete, no longer being developed, or may no longer work. Learn more.	5.0.2
	FindBugs Maven Build Reports	

Update information obtained: 47 min ago

1. Install the **Warnings Next Generation Plugin** under the Plugin Manager



Installing Plugins/Upgrades

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Warnings Next Generation ● Success

Loading plugin extensions ● Running

Restarting Jenkins ● Pending

Go back to the top page
(you can start using the installed plugins right away)

☑ Restart Jenkins when installation is complete and no jobs are running

2. Restart Jenkins

2. Install and Configure Maven in Jenkins

In this step, we are going to install and configure Maven in Jenkins. For this, we have to download the Maven binary from the official website. At the time of writing, the Apache Maven version is 3.6.3.

The guide is assuming that you are running Jenkins through Docker, so to install Maven binary, you need to follow the below-mentioned steps:

```
$ docker exec -it jenkins-container /bin/bash

$ cd /var/jenkins_home

$ curl http://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz --output apache-maven-3.6.3-bin.tar.gz

$ tar -xvzf apache-maven-3.6.3-bin.tar.gz && cd apache-maven-3.6.3
```

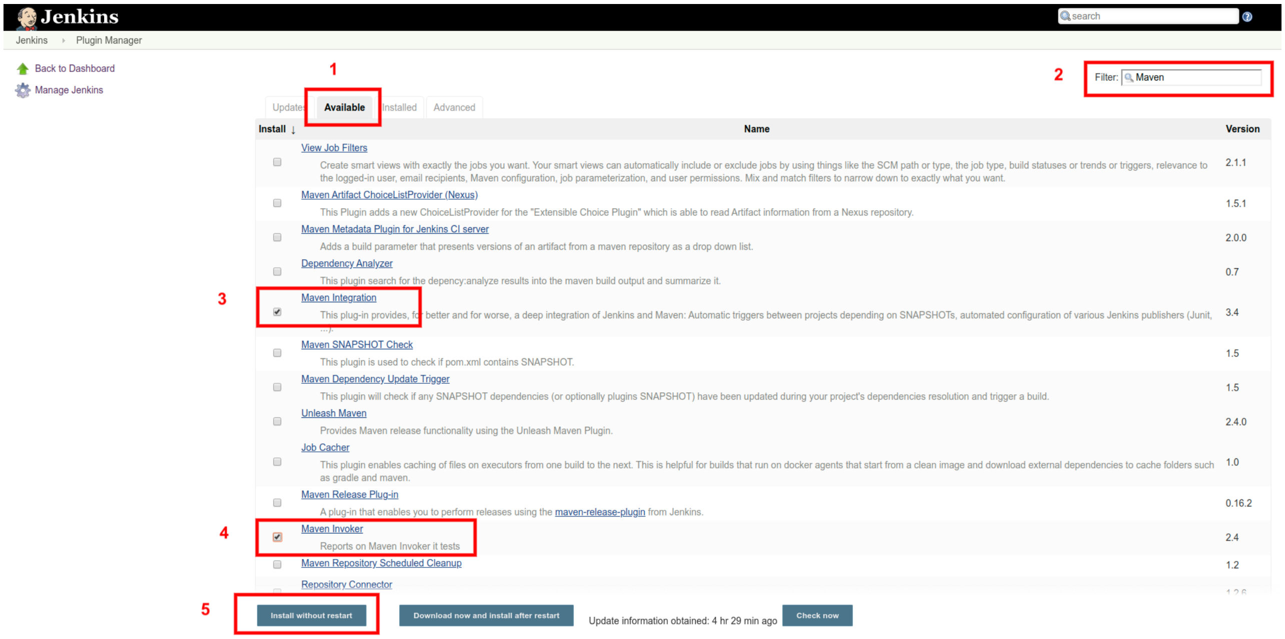
Now as you have downloaded the required binary, copy the path for later use:

```
$ pwd

/var/jenkins_home/apache-maven-3.6.3
```

Install Maven Plugins in Jenkins

To install Maven plugins. Go to *Dashboard > Manage Jenkins > Manage Plugins > Available* and search for *Maven*, as shown below:



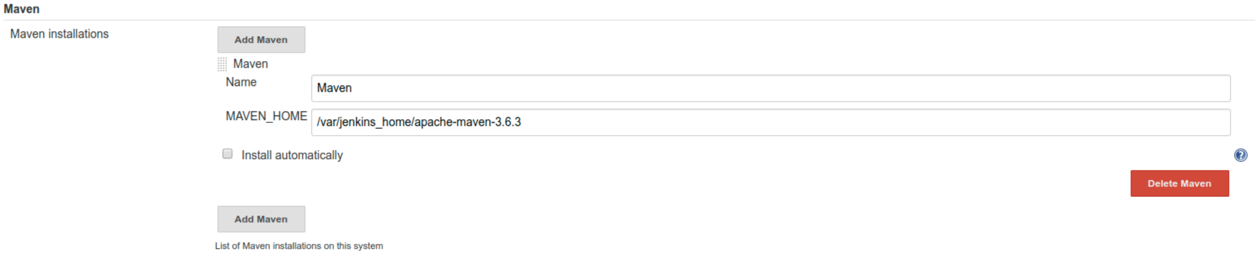
Filter: Maven

Install ↓	Name	Version
□	View Job Filters Create smart views with exactly the jobs you want. Your smart views can automatically include or exclude jobs by using things like the SCM path or type, the job type, build statuses or trends or triggers, relevance to the logged-in user, email recipients, Maven configuration, job parameterization, and user permissions. Mix and match filters to narrow down to exactly what you want.	2.1.1
□	Maven Artifact ChoiceListProvider (Nexus) This Plugin adds a new ChoiceListProvider for the "Extensible Choice Plugin" which is able to read Artifact information from a Nexus repository.	1.5.1
□	Maven Metadata Plugin for Jenkins CI server Adds a build parameter that presents versions of an artifact from a maven repository as a drop down list.	2.0.0
□	Dependency Analyzer This plugin search for the depency:analyze results into the maven build output and summarize it.	0.7
☑	Maven Integration This plug-in provides, fo better and for worse, a deep integration of Jenkins and Maven: Automatic triggers between projects depending on SNAPSHOTs, automated configuration of various Jenkins publishers (Junit, ...).	3.4
□	Maven SNAPSHOT Check This plugin is used to check if pom.xml contains SNAPSHOT.	1.5
□	Maven Dependency Update Trigger This plugin will check if any SNAPSHOT dependencies (or optionally plugins SNAPSHOT) have been updated during your project's dependencies resolution and trigger a build.	1.5
□	Unleash Maven Provides Maven release functionality using the Unleash Maven Plugin.	2.4.0
□	Job Cacher This plugin enables caching of files on executors from one build to the next. This is helpful for builds that run on docker agents that start from a clean image and download external dependencies to cache folders such as gradle and maven.	1.0
□	Maven Release Plug-in A plug-in that enables you to perform releases using the maven-release-plugin from Jenkins.	0.16.2
☑	Maven Invoker Reports on Maven Invoker it tests	2.4
□	Maven Repository Scheduled Cleanup	1.2
□	Repository Connector	1.2.6

Update information obtained: 4 hr 29 min ago

On the next page, along with *Maven Integration* and *Maven Invoker*, you will see some additional dependencies getting installed, which are required for both plugins to work.

Now, let’s set the Maven path which you copied from the Jenkins's Container. Go to *Dashboard > Manage Jenkins > Global Tool Configuration* and find *Maven* to set the extracted binary home directory path, which is, /var/jenkins_home/apache-maven-3.6.3



Maven

Maven installations

Maven

Name: Maven

MAVEN_HOME: /var/jenkins_home/apache-maven-3.6.3

□ Install automatically

List of Maven installations on this system

Save the setting, following which your Maven application is all set to go.

3: Configuration

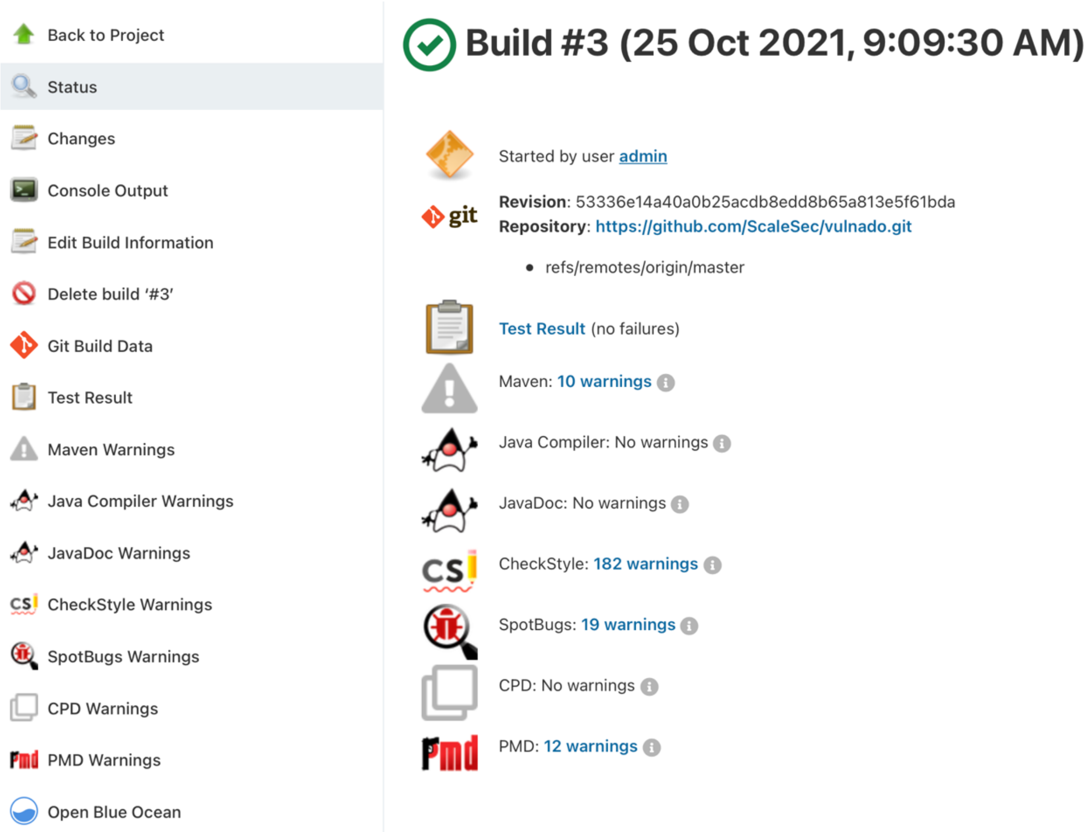
1. Create a new pipeline project, and use the following Jenkinsfile in your pipeline:

```
pipeline {
    agent any
    stages {
        stage ('Checkout') {
            steps {
                git branch:'master', url: 'https://github.com/ScaleSec/vulnado.git'
            }
        }

        stage ('Build') {
            steps {
                sh '/var/jenkins_home/apache-maven-3.6.3/bin/mvn --batch-mode -V -U -e clean
verify -Dsurefire.useFile=false -Dmaven.test.failure.ignore'
            }
        }

        stage ('Analysis') {
            steps {
                sh '/var/jenkins_home/apache-maven-3.6.3/bin/mvn --batch-mode -V -U -e
checkstyle:checkstyle pmd:pmd pmd:cpd findbugs:findbugs'
            }
        }
    }
    post {
        always {
            junit testResults: '**/target/surefire-reports/TEST-*.xml'
            recordIssues enabledForFailure: true, tools: [mavenConsole(), java(), javaDoc()]
            recordIssues enabledForFailure: true, tool: checkStyle()
            recordIssues enabledForFailure: true, tool: spotBugs(pattern:
'**/target/findbugsXml.xml')
            recordIssues enabledForFailure: true, tool: cpd(pattern: '**/target/cpd.xml')
            recordIssues enabledForFailure: true, tool: pmdParser(pattern: '**/target/pmd.xml')
        }
    }
}
```

2. Please note that **Warnings Next Generation Plugin** support different languages by applying different SAST, please refer to https://github.com/jenkinsci/warnings-ng-plugin/blob/master/SUPPORTED-FORMATS.md and include the languages that you are using.



- Back to Project
- Status
- Changes
- Console Output
- Edit Build Information
- Delete build '#3'
- Git Build Data
- Test Result
- Maven Warnings
- Java Compiler Warnings
- JavaDoc Warnings
- CheckStyle Warnings
- SpotBugs Warnings
- CPD Warnings
- PMD Warnings
- Open Blue Ocean

Build #3 (25 Oct 2021, 9:09:30 AM)

Started by user admin

Revision: 53336e14a40a0b25acdb8edd8b65a813e5f61bda
Repository: https://github.com/ScaleSec/vulnado.git

- refs/remotes/origin/master

Test Result (no failures)

Maven: 10 warnings

Java Compiler: No warnings

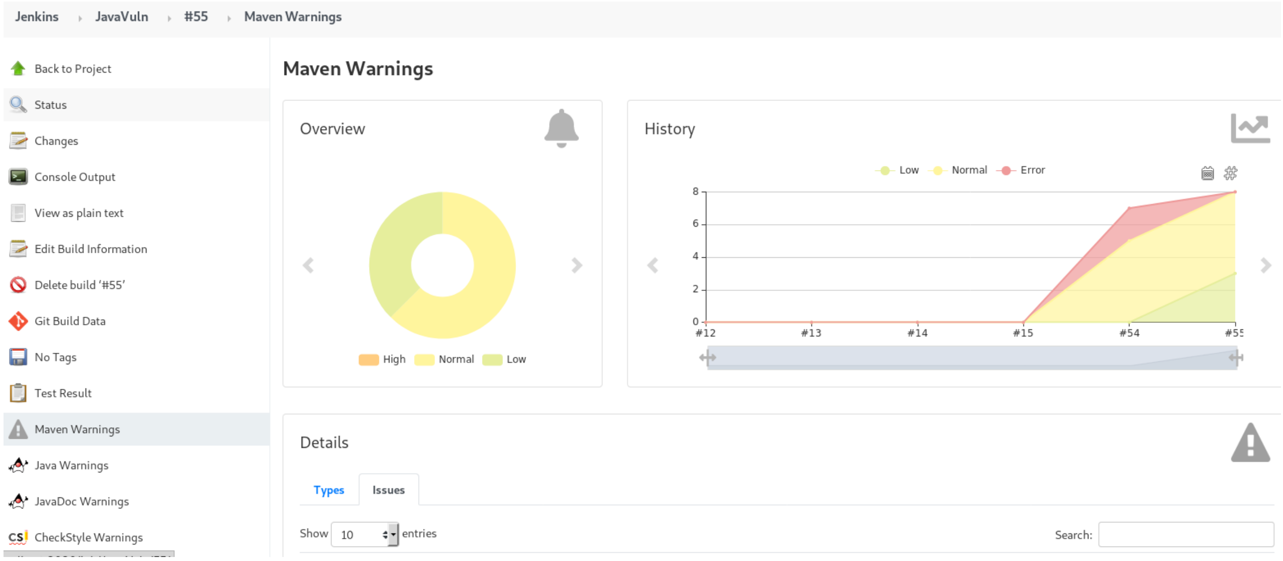
JavaDoc: No warnings

CheckStyle: 182 warnings

SpotBugs: 19 warnings

CPD: No warnings

PMD: 12 warnings



Jenkins › JavaVuln › #55 › Maven Warnings

- Back to Project
- Status
- Changes
- Console Output
- View as plain text
- Edit Build Information
- Delete build '#55'
- Git Build Data
- No Tags
- Test Result
- Maven Warnings
- Java Warnings
- JavaDoc Warnings
- CheckStyle Warnings

Maven Warnings

Overview

History

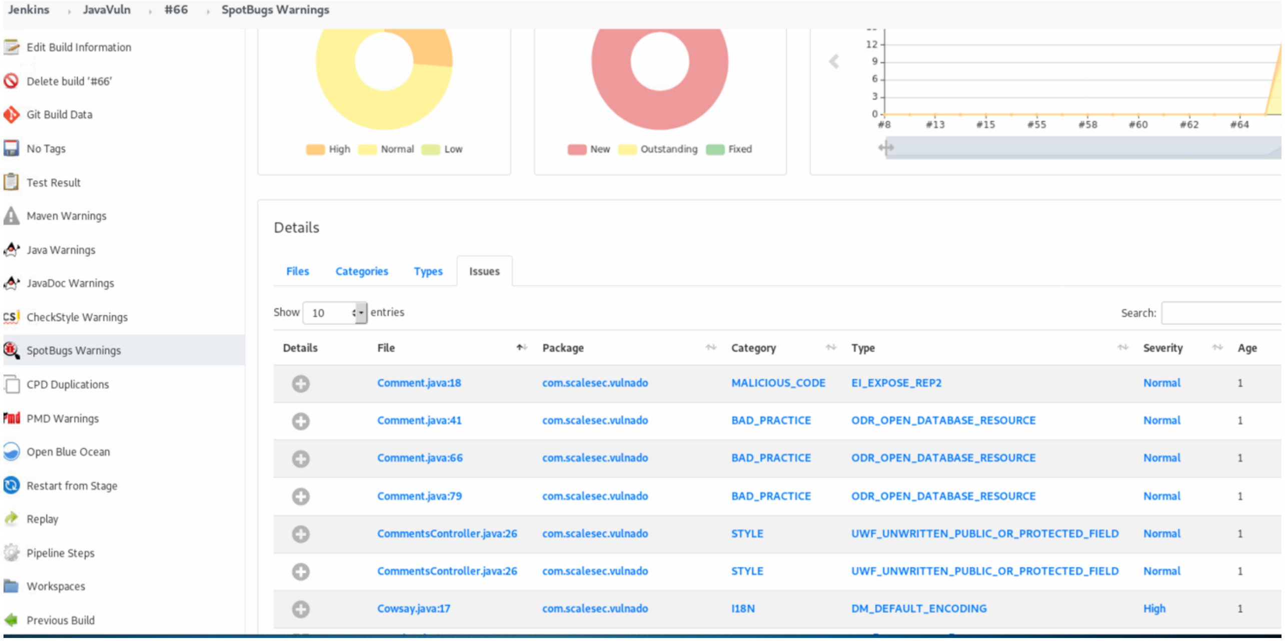
Details

Types | Issues

Show 10 entries

Search:

3. You may check the Warnings from different tools in the build information



Details	File	Package	Category	Type	Severity	Age
⊕	Comment.java:18	com.scalesec.vulnado	MALICIOUS_CODE	EI_EXPOSE_REP2	Normal	1
⊕	Comment.java:41	com.scalesec.vulnado	BAD_PRACTICE	ODR_OPEN_DATABASE_RESOURCE	Normal	1
⊕	Comment.java:66	com.scalesec.vulnado	BAD_PRACTICE	ODR_OPEN_DATABASE_RESOURCE	Normal	1
⊕	Comment.java:79	com.scalesec.vulnado	BAD_PRACTICE	ODR_OPEN_DATABASE_RESOURCE	Normal	1
⊕	CommentsController.java:26	com.scalesec.vulnado	STYLE	UWF_UNWRITTEN_PUBLIC_OR_PROTECTED_FIELD	Normal	1
⊕	CommentsController.java:26	com.scalesec.vulnado	STYLE	UWF_UNWRITTEN_PUBLIC_OR_PROTECTED_FIELD	Normal	1
⊕	Cowsay.java:17	com.scalesec.vulnado	I18N	DM_DEFAULT_ENCODING	High	1

4: Reference

https://github.com/jenkinsci/warnings-ng-plugin/blob/master/doc/Documentation.md#configuration
https://appfleet.com/blog/ci-dc-pipeline-using-jenkins-git-and-maven/

END OF DOCUMENT