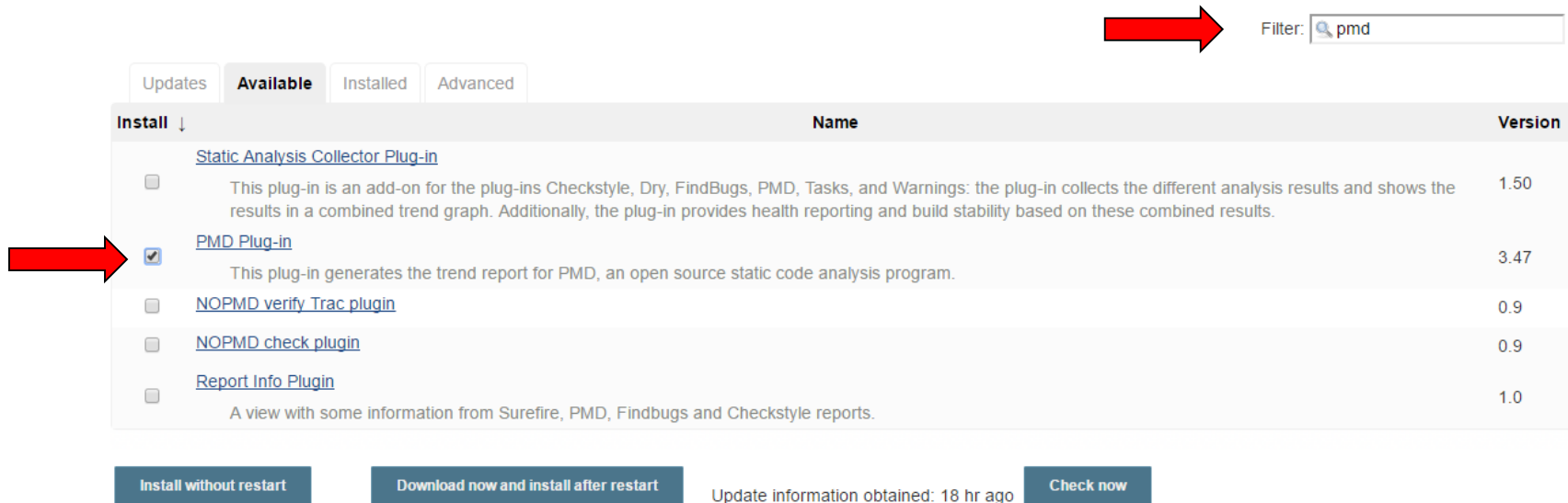


# Jenkins – Static Code Analysis

# Configure Plugins for Code Analysis

- Go to Manage Jenkins → Manage Plugins
- Search for PMD under 'Available' plugins tab
- Install



The screenshot shows the Jenkins Manage Plugins interface. A red arrow points to the search filter 'pmd' in the top right. Below the tabs, a table lists available plugins. A second red arrow points to the 'PMD Plug-in' row, which has its checkbox selected. The table includes columns for 'Name' and 'Version'.

| Install ↓                           | Name   | Version |
|-------------------------------------|--|---------|
| <input type="checkbox"/>            | <a href="#">Static Analysis Collector Plug-in</a><br>This plug-in is an add-on for the plug-ins Checkstyle, Dry, FindBugs, PMD, Tasks, and Warnings: the plug-in collects the different analysis results and shows the results in a combined trend graph. Additionally, the plug-in provides health reporting and build stability based on these combined results. | 1.50    |
| <input checked="" type="checkbox"/> | <a href="#">PMD Plug-in</a><br>This plug-in generates the trend report for PMD, an open source static code analysis program.   | 3.47    |
| <input type="checkbox"/>            | <a href="#">NOPMD verify Trac plugin</a>   | 0.9     |
| <input type="checkbox"/>            | <a href="#">NOPMD check plugin</a>   | 0.9     |
| <input type="checkbox"/>            | <a href="#">Report Info Plugin</a><br>A view with some information from Surefire, PMD, Findbugs and Checkstyle reports.  | 1.0     |

At the bottom, there are buttons for 'Install without restart', 'Download now and install after restart', and 'Check now'. A status message indicates 'Update information obtained: 18 hr ago'.

# Configure Plugins for Code Analysis


- This will also install other dependencies
- Note that Static analysis utilities plugin is also installed. This basically provides a dashboard view, trends graph etc for data generated by PMD and other tools

## Installing Plugins/Upgrades


### Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

### Javadoc Plugin

 Success

### Maven Integration plugin


 Installing



### Static Analysis Utilities

 Pending

### PMD Plug-in

 Pending

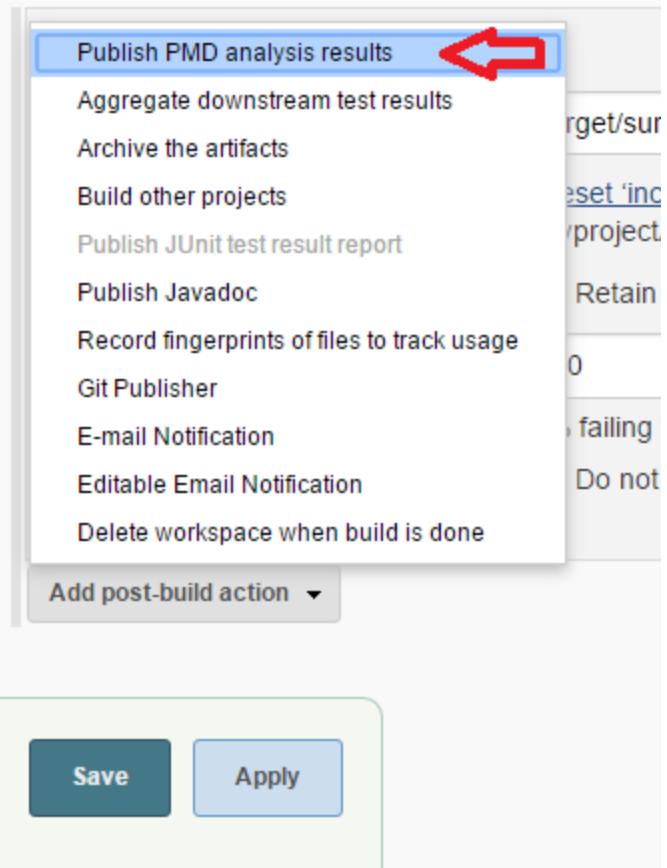
➡ [Go back to the top page](#)  
(you can start using the installed plugins right away)

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

# Configure Jenkins job to run PMD

- Go to Jenkins Dashboard, Select 'mvndemo' build job to configure
- Add Post-Build-Actions
- Click Apply and Save

## Post-build Actions



Publish PMD analysis results

Aggregate downstream test results

Archive the artifacts

Build other projects

Publish JUnit test result report

Publish Javadoc

Record fingerprints of files to track usage

Git Publisher

E-mail Notification

Editable Email Notification

Delete workspace when build is done

Add post-build action ▼

Save Apply

# Configure Jenkins job to run PMD

- You can keep the PMD Results directory empty, since it will be picked up from default location in Jenkins which is [/.jenkins/workspace/mvndemo/target/pmd.xml](#)
- Don't forget to add the goal `pmd:pmd` to the build step

## Post-build Actions



### Publish PMD analysis results



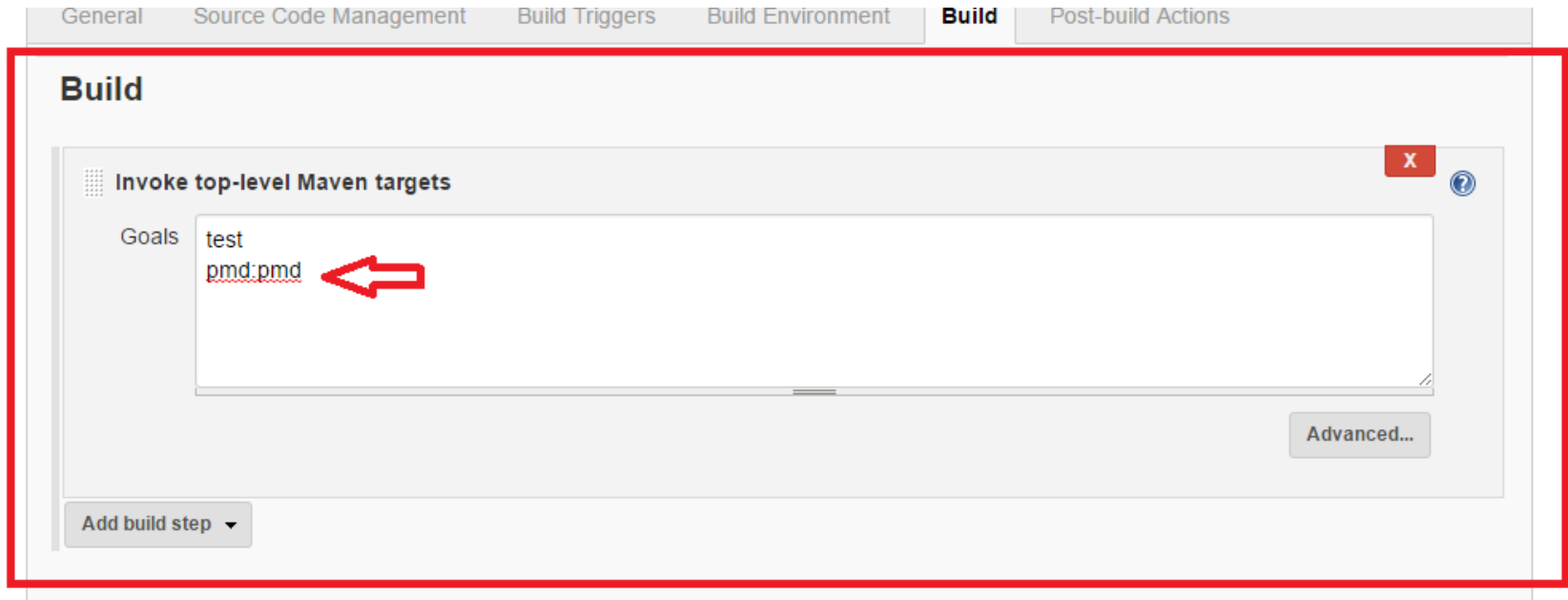
PMD results

[Fileset includes](#) setting that specifies the generated raw PMD XML report files, such as `**/pmd.xml`. Basedir of the fileset is [the workspace root](#). If no value is set, then the default `**/pmd.xml` is used. Be sure not to include any non-report files into this pattern.

Advanced...

# Configure Jenkins job to run PMD contd

- Don't forget to add the goal `pmd:pmd` to the build step



The screenshot displays the Jenkins configuration interface for a job, specifically the 'Build' tab. The 'Invoke top-level Maven targets' step is selected, and its configuration is shown. In the 'Goals' text area, the goals 'test' and 'pmd:pmd' are listed. A red arrow points to 'pmd:pmd', highlighting it. The entire configuration area is enclosed in a red rectangular border. At the bottom left, there is a button labeled 'Add build step' with a dropdown arrow. At the bottom right, there is a button labeled 'Advanced...'. The top of the interface shows tabs for 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'.

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

## Build

Invoke top-level Maven targets

Goals

```
test  
pmd:pmd
```

Advanced...

Add build step ▼

# Configure Maven project for Code Analysis

- Lets configure the project 'mvndemo' to be statically analysed by PMD
- PMD will be run using **maven-pmd-plugin**
- Open **pom.xml** and make the following entry just below <dependencies>
- Note: PMD has various rulesets, we are just going to use **basic** ruleset.

```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-pmd-plugin</artifactId>
      <version>3.7</version>
      <configuration>
        <rulesets>
          <ruleset>/rulesets/basic.xml</ruleset>
        </rulesets>
      </configuration>
    </plugin>
  </plugins>
</reporting>
```

# Configure Maven project for Code Analysis

- Now open Bitbucket Repository and deliberately introduce a violation
- Modify App.java – Here I am introducing a local variable which is unused
- Run the build in Jenkins – PMD should pick this violation



The screenshot shows the Bitbucket web interface for a repository named 'mvndemo'. The left sidebar contains navigation links: Overview, Source (selected), Commits, Branches, Pull requests, Pipelines, Downloads, and Settings. The main content area displays the source code for the file 'mvndemo / src / main / java / com / crud / demo / mvndemo / App.java'. The code is as follows:

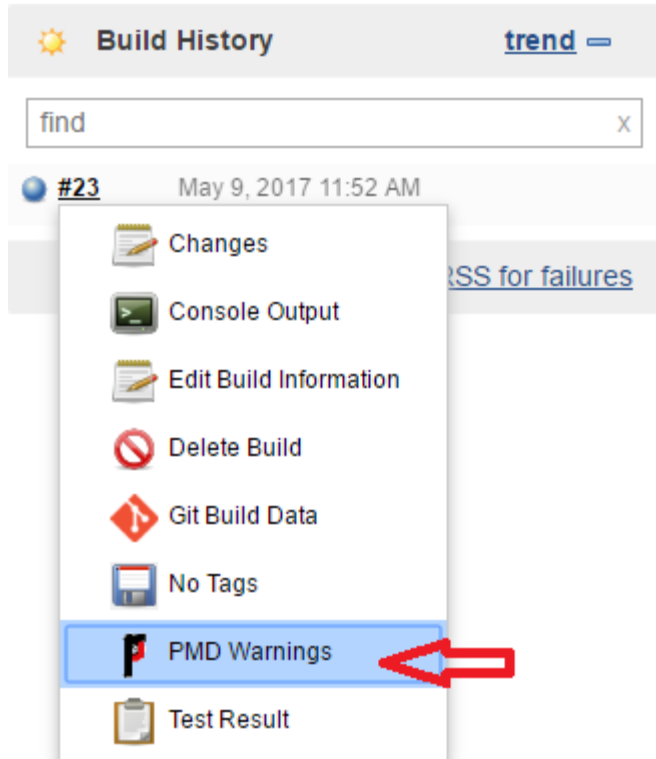
```
1 package com.cruds.demo.mvndemo;
2
3 /**
4  * Hello world!
5  *
6  */
7 public class App
8 {
9     public static void main( String[] args )
10    {
11        System.out.println( "Hello World!" );
12    }
13
14    public String getGreeting()
15    {
16        int x = 10;
17        return "Hello from Maven";
18    }
19
20
21 }
```

A red arrow points to the line `int x = 10;` on line 16, indicating an unused local variable, which is a code quality violation that PMD can detect.



# View Code Analysis Results

- To view PMD Violation Report – Click on PMD Warnings



# View Code Analysis Results contd..

- You can see that PMD has flagged the Unused Local Variable. Click on the link provided to the source file.

## PMD Result

### Warnings Trend


| All Warnings | New Warnings      |
|--------------|-------------------|
| 1            | <a href="#">1</a> |

### Summary

| Total | High Priority | Normal Priority   |
|-------|---------------|-------------------|
| 1     | 0             | <a href="#">1</a> |

### Details

**Details** New

 [App.java:16](#), UnusedLocalVariable, Priority: Normal

**Avoid unused local variables such as 'x'.**  
Detects when a local variable is declared and/or assigned, but not used.

```
public class Foo {  
    public void doSomething() {  
        int i = 5; // Unused  
    }  
}
```

# View Code Analysis Results contd..

- You can see that PMD has flagged the Unused Local Variable and Source code, line number is displayed

## Content of file App.java

```
01 package com.cruds.demo.mvndemo;
02
03 /**
04  * Hello world!
05  *
06  */
07 public class App
08 {
09     public static void main( String[] args )
10     {
11         System.out.println( "Hello World!" );
12     }
13
14     public String getGreeting()
15     {
16         int x = 10;
17         return "Hello from Maven";
18     }
19
20
21 }
```

# Behind the scenes..

- For the curious few who want to know what happened behind the scene, lets browse Jenkins workspace and see for ourselves
- Click on the '[workspace](#)' folder, then click on '[target](#)'
- You can see [5 rules xml](#) files and below it the [pmd.xml](#) which contains the result of analysis
- You can also click on '[site](#)' folder and view the [pmd.html](#)
- You can also view by browsing the folder [.jenkins/workspace/mvndemo/target](#)

Back to Dashboard

Status

Changes

Workspace

Build Now

Delete Project

Configure

PMD Warnings

Build History trend

find X

#23 May 9, 2017 11:52 AM

Project mvndemo

Demo of Maven Build in Jenkins

Workspace

Recent Changes

Latest Test Result (no failures)

Permalinks

Back to Dashboard

Status

Changes

Workspace

Build Now

Delete Project

Configure

PMD Warnings

Build History trend

find X

#23 May 9, 2017 11:52 AM

Workspace of mvndemo on master

target /

classes/com/cruds/demo/mvndemo

maven-status/maven-compiler-plugin

site

surefire-reports

test-classes/com/cruds/demo/mvndemo

java-basic.xml 27.59 KB view

java-empty.xml 10.24 KB view

java-imports.xml 4.94 KB view

java-unnecessary.xml 12.48 KB view

java-unusedcode.xml 2.90 KB view

pmd.xml 593 B view

(all files in zip)

# Lets make more violations in App.java

- Lets import a Class which is not used any where in code, there by violating, Unused Imports rule of PMD
- Make the changes in bitbucket repository
- Kick of the Jenkins build again, we should have 2 violations now

master ▾  mvndemo / src / main / java / com / cruds / demo / mvndemo / App.java

40421b9 10 seconds ago ▾ Full commit

```
1 package com.cruds.demo.mvndemo;
2
3 import java.util.Scanner;
4
5 /**
6  * Hello world!
7  *
8  */
9 public class App
10 {
11     public static void main( String[] args )
12     {
13         System.out.println( "Hello World!" );
14     }
15
16     public String getGreeting()
17     {
18         int x = 10;
19         return "Hello from Maven";
20     }
21
22
23 }
```

# View PMD Results again

- You can also view various information by clicking on other tabs – Types, Warnings etc

## PMD Result

### Warnings Trend

| All Warnings | New Warnings      | Fixed Warnings |
|--------------|-------------------|----------------|
| 2            | <a href="#">1</a> | 0              |

### Summary

| Total | High Priority | Normal Priority   | Low Priority |
|-------|---------------|-------------------|--------------|
| 2     | 0             | <a href="#">2</a> | 0            |

### Details

**Categories** | Types | Warnings | Details | New

| Category                          | Total | Distribution |
|-----------------------------------|-------|--------------|
| <a href="#">Import Statements</a> | 1     | <div></div>  |
| <a href="#">Unused Code</a>       | 1     | <div></div>  |
| Total                             | 2     |              |

# Static Analysis Plugin Dashboard


- Now we have had 2 build and 2 PMD analysis results
- Now we can see the defect trends in project. Click on 'Back to Project' link


Jenkins


mvndemo


#24


PMD Warnings


 Back to Project


 Status


 Changes


 Console Output


 Edit Build Information


 Delete Build

 Git Build Data

 No Tags

 **PMD Warnings**

 Test Result

 Previous Build

## PMD Result

### Warnings Trend

| All Warnings | New Warnings      |
|--------------|-------------------|
| 2            | <a href="#">1</a> |

### Summary

| Total | High Priority | Normal Priority   |
|-------|---------------|-------------------|
| 2     | 0             | <a href="#">2</a> |

### Details

Categories

Types

Warnings

Details

New

| Category                          | Total | Distribution |
|-----------------------------------|-------|--------------|
| <a href="#">Import Statements</a> | 1     | <div></div>  |
| <a href="#">Unused Code</a>       | 1     | <div></div>  |
| Total                             | 2     |              |

# Static Analysis Plugin Dashboard contd

- You can see the PMD Trend has increased
- JUnit Trend has maintained itself, since there were no failures
- These reports are accumulated overtime provides valuable inputs to managers on code quality
- Now lets fix it the bugs and rerun the build





# View results again

- You can see that App.java was edited online in bitbucket
- You can also see git revision number
- PMD warnings 0
- Test results – no failures



## Changes

1. App.java edited online with Bitbucket ([detail](#))



Started by user [Jenkins Admin](#)



**Revision:** edc61599593d80320d906350455a7519c35d6b2c

- `refs/remotes/origin/master`



PMD: 0 warnings from one analysis.

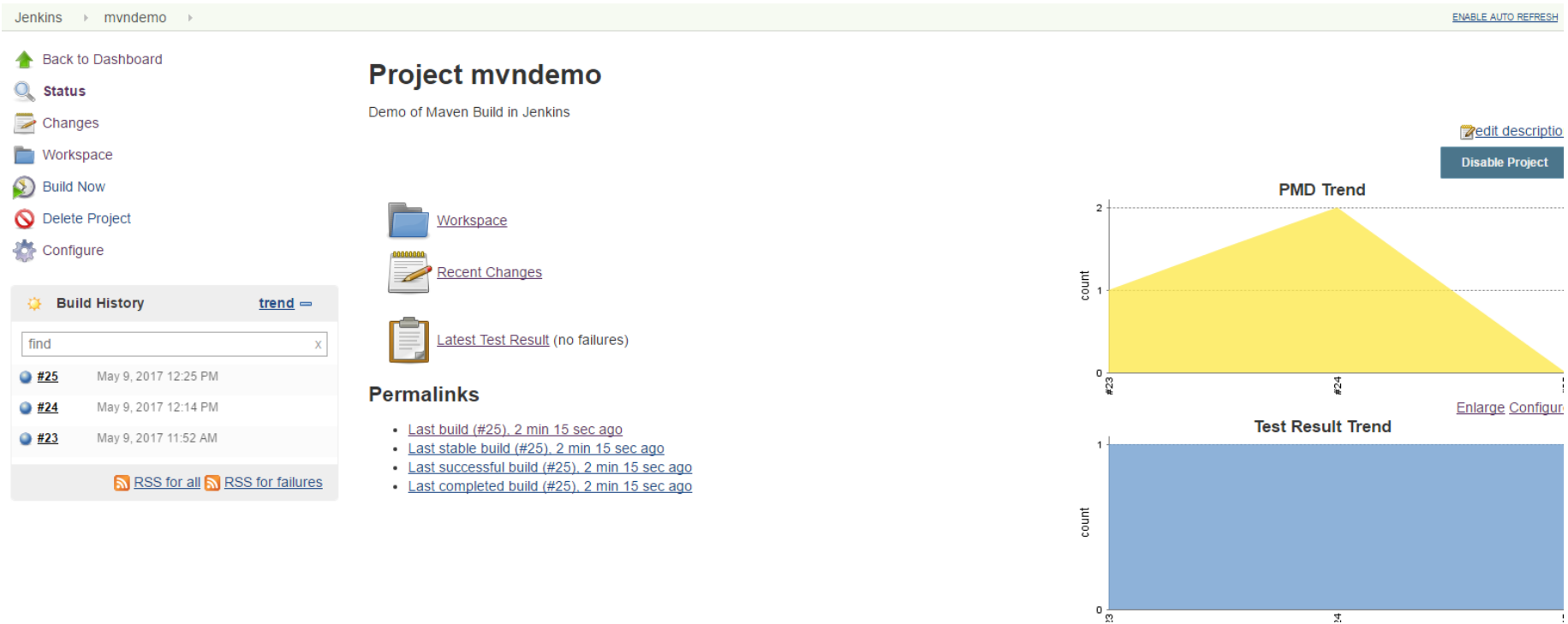
- [2 fixed warnings](#)



[Test Result](#) (no failures)

# Lets see the trend again

- You can see a sharp reduction in PMD trend graph.



# Check Style – Plugin Installation

- Go to Manage Jenkins → Manage Plugins
- Search for Checkstyle
- Select and Install without restart

The screenshot shows the Jenkins Manage Plugins interface. At the top right, a search filter is set to 'checkstyle'. Below the filter, there are tabs for 'Updates', 'Available', 'Installed', and 'Advanced'. The 'Available' tab is selected. A table lists available plugins with columns for 'Name' and 'Version'. The first plugin, 'Checkstyle Plug-in', is selected with a checkbox, indicated by a red arrow. Below the table, there are two buttons: 'Install without restart' (highlighted with a red arrow) and 'Download now and install after restart'. At the bottom right, there is a 'Check now' button and a timestamp 'Update information obtained: 1 hr 19 min ago'.

Filter:

Updates Available Installed Advanced

Install ↓

|                                     | Name   | Version |
|-------------------------------------|--|---------|
| <input checked="" type="checkbox"/> | <a href="#">Checkstyle Plug-in</a><br>This plug-in generates the trend report for Checkstyle, an open source static code analysis program.   | 3.48    |
| <input type="checkbox"/>            | <a href="#">Static Analysis Collector Plug-in</a><br>This plug-in is an add-on for the plug-ins Checkstyle, Dry, FindBugs, PMD, Tasks, and Warnings: the plug-in collects the different analysis results and shows the results in a combined trend graph. Additionally, the plug-in provides health reporting and build stability based on these combined results. | 1.50    |
| <input type="checkbox"/>            | <a href="#">Box UK - JSLint</a><br>Lint JavaScript files using a sensible ruleset - outputs to checkstyle format   | 0.8.2   |
| <input type="checkbox"/>            | <a href="#">Report Info Plugin</a><br>A view with some information from Surefire, PMD, Findbugs and Checkstyle reports.  | 1.0     |

Update information obtained: 1 hr 19 min ago

# Check Style – configure Jenkins job

- Go to Jenkins dashboard, Configure 'mvndemo' job
- Add Post build actions → Publish Checkstyle analysis result

The screenshot shows the Jenkins 'Post-build Actions' configuration page. The 'Publish Checkstyle analysis results' option is selected in the dropdown menu. The configuration includes fields for 'Test report XMLs' (target/surefire-reports/\*.xml), 'Health report amplification factor' (1.0), and a checkbox for 'Retain long standard output/error'. The 'Add post-build action' button is at the bottom left, and 'Save' and 'Apply' buttons are at the bottom right.

General Source Code Management Build Triggers Build Environment Build **Post-build Actions**

**Publish JUnit test result report**

Test report XMLs

[Fileset 'includes'](#) setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/\*.xml'. Basedir of the fileset is [the workspace root](#).

☐ Retain long standard output/error

Health report amplification factor

**Publish Checkstyle analysis results**

Publish PMD analysis results

Aggregate downstream test results

Archive the artifacts

Build other projects

Publish JUnit test result report

Publish Javadoc

Record fingerprints of files to track usage

Git Publisher

E-mail Notification

Editable Email Notification

Delete workspace when build is done

Add post-build action

Save Apply

# Check Style – configure Jenkins job contd..

- Don't forget to add the maven goal **checkstyle:checkstyle** to Build step
- Check style results directory can be default directory

The screenshot displays the Jenkins job configuration interface with the 'Build' tab selected. Two sections are highlighted with red borders:

- Invoke top-level Maven targets:** This section contains a text area for 'Goals' with the following content:

```
test  
pmd:pmd  
checkstyle:checkstyle
```

A red arrow points to the 'checkstyle:checkstyle' goal. Below the text area is an 'Advanced...' button. At the bottom left of this section is an 'Add build step' dropdown.
- Post-build Actions:** This section contains a 'Publish Checkstyle analysis results' step. It has a 'Checkstyle results' text input field. Below the input field is a description: 

[Fileset includes](#) setting that specifies the generated raw CheckStyle XML report files, such as `**/checkstyle-result.xml`. Basedir of the fileset is [the workspace root](#). If no value is set, then the default `**/checkstyle-result.xml` is used. Be sure not to include any non-report files into this pattern.

Below the description is an 'Advanced...' button.

# Check Style – Configure Maven Project POM

- Make an entry for **mvn-checkstyle-plugin** in mvndemo project **pom.xml**
- You can make the changes in Bitbucket repository directly or make changes in Eclipse workspace and do *git push to remote origin*

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-checkstyle-plugin</artifactId>
  <version>2.17</version>
  <reportSets>
    <reportSet>
      <reports>
        <report>checkstyle</report>
      </reports>
    </reportSet>
  </reportSets>
</plugin>
```

# Check Style – Contd..

- After making the changes your **pom.xml** , 'reporting' section should look like this

```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-pmd-plugin</artifactId>
      <version>3.7</version>
      <configuration>
        <rulesets>
          <ruleset>/rulesets/basic.xml</ruleset>
        </rulesets>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-checkstyle-plugin</artifactId>
      <version>2.17</version>
      <reportSets>
        <reportSet>
          <reports>
            <report>checkstyle</report>
          </reports>
        </reportSet>
      </reportSets>
    </plugin>
  </plugins>
</reporting>
```

# View Check Style Report

- Run the build and check the reports
- Click on warnings link



## Changes

1. added checkstyle plugin to pom ([detail](#))



Started by user [Jenkins Admin](#)



**Revision:** 3dd32cdf17f9c7a9661d515c4643a29f57b42fc3

- refs/remotes/origin/master



Checkstyle: [18 warnings](#) from one analysis.

- [18 new warnings](#)



PMD: 0 warnings from one analysis.

- No warnings since build 25.
- New zero warnings highscore: no warnings since yesterday!



[Test Result](#) (no failures)



# View Check Style Report contd







- You will see detailed information
- You can click individual tabs and links to see details

## Checkstyle Warnings - New Warnings

### Summary

| Total | High Priority      | Normal Priority | Low Priority |
|-------|--------------------|-----------------|--------------|
| 18    | <a href="#">18</a> | 0               | 0            |

### Details

| Categories                 | Types | Warnings   | Details |
|----------------------------|-------|--|---------|
| Category                   | Total | Distribution   |         |
| <a href="#">Blocks</a>     | 3     |    |         |
| <a href="#">Checks</a>     | 1     |    |         |
| <a href="#">Design</a>     | 1     |  |         |
| <a href="#">Javadoc</a>    | 3     |  |         |
| <a href="#">Regex</a>      | 5     |  |         |
| <a href="#">Whitespace</a> | 5     |  |         |
| Total                      | 18    |  |         |

# Activity

A decorative header element consisting of a horizontal row of five circles. The first, third, and fifth circles are solid light purple. The second and fourth circles are hollow with a light purple outline.

- Install Find Bugs Plugin
- Configure Maven pom for maven-findbugs plugin
- Configure a job and run build