

Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.

Lesson Objectives



Introduction of Sonar

- Architecture
- Integration

Analyzing the Java code with Sonar


Integrating Jenkins with Sonar

Analyzing Maven, Java Code with Sonar

Instructor Notes:

Add instructor notes here.

5.1: introduction of Sonar
Sonar



Sonar is an open source platform used by development teams to manage source code quality. Sonar has been made with a main objective in mind: make code quality management accessible to everyone with minimal effort.

SonarQube (formerly known as *Sonar*) is an open source tool suite to measure and analyze the quality of source code. It is written in Java but is able to analyze code in about 20 different programming languages. Code analysis may be started manually by executing a so-called sonar runner but SonarQube's full potential is especially revealed when used in combination with continuous integration such as a Jenkins server.

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

3

Add the notes here.

Instructor Notes:

Add instructor notes here.

5.1: introduction of Sonar Why Code Analyzer tool



Why we are using SonarQube(Code analyzer tool)

Code quality analysis helps to make your code:

- less error-prone
- more sustainable
- more reliable
- more readable
- more welcoming to new contributors

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

4

Code quality analysis mainly relies on a set of tools that look at your code and give you hints. The most famous tools are Findbugs, PMD, Checkstyle but also code coverage tools such as Jacoco. JDT itself provides very powerful quality checks, but there are not enabled by default. You should go to Error/Warnings in preferences and replace all "ignore" by "Warning".

Instructor Notes:

Add instructor notes here.

5.1: introduction of Sonar Features of Sonar



- Write clean Code
- DevOps Integration
- Centralize Quality
- Support 20+ languages

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

5

Write clean code

Overall health

Your project home page shows where you stand in terms of quality in a glimpse of an eye. This main page also shows you an immediate sense of the good results achieved over time.

Focus on the Leak

The water leak paradigm is a simple yet powerful way to manage code quality: quality of new - changed and added - code should be put under control before anything else. Once that Leak is under control, code quality will start improving mechanically. In SonarQube, the Leak is a built-in concept that you can't miss. Once you've had a look at this yellow area on the left of your project home page, you will always remain focused on it to not miss any new issues.

Enforce Quality Gate

With SonarQube, a developer has everything at hand to take ownership of the quality of his code. To fully enforce a code quality practice across all teams, you need to set up a Quality Gate. This core concept of SonarQube is a set of requirements that tells whether or not a new version of a project can go into production. SonarQube's default Quality Gate checks what happened on the Leak period and fails if your new code got worse in this period.

Analyze pull requests

Once you have SonarQube in place, you will quickly want to make sure you add as few issues as possible to your code base. To shorten the feedback loop so you don't have to wait for new analyses to be available on SonarQube, you can set up the analysis of your pull requests. Analyses will be run on your feature branches without being pushed to SonarQube, giving you the opportunity to fix issues before they ever reach SonarQube!

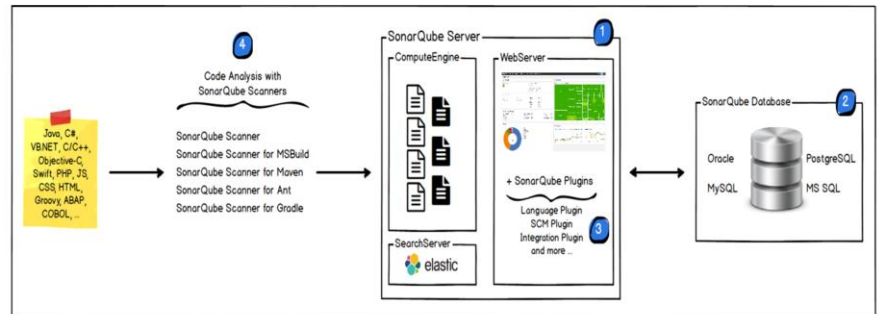
Instructor Notes:

How sonar works give idea.

5.1: introduction of Sonar Sonar-Architecture

The SonarQube Platform is made of 4 components

- SonarQube Server
- SonarQube Database
- SonarQube Plugins
- SonarQube Scanners



Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

8

The SonarQube Platform is made of 4 components:

One **SonarQube Server** starting 3 main processes:

- a **Web Server** for developers, managers to browse quality snapshots and configure the SonarQube instance
- a **Search Server** based on Elasticsearch to back searches from the UI
- a **Compute Engine Server** in charge of processing code analysis reports and saving them in the SonarQube Database

One **SonarQube Database** to store:

- the configuration of the SonarQube instance (security, plugins settings, etc.)
- the quality snapshots of projects, views, etc.

Multiple **SonarQube Plugins** installed on the server, possibly including language, SCM, integration, authentication, and governance plugins

One or more **SonarQube Scanners** running on your Build / Continuous Integration Servers to analyze projects

Instructor Notes:

Add instructor notes here.

5.1: introduction of Sonar Sonar-Integration



The following schema shows how SonarQube integrates with other ALM tools where the various components of SonarQube are used.

- Developers code in their IDEs and use SonarLint to run local analysis.
- Developers push their code into their favorite SCM : git, SVN, TFVC, ...
- The Continuous Integration Server triggers an automatic build, and the execution of the SonarQube Scanner required to run the SonarQube analysis.
- The analysis report is sent to the SonarQube Server for processing.
- SonarQube Server processes and stores the analysis report results in the SonarQube Database, and displays the results in the UI.
- Developers review, comment, challenge their issues to manage and reduce their Technical Debt through the SonarQube UI.
- Managers receive Reports from the analysis.
- Ops use APIs to automate configuration and extract data from SonarQube.
- Ops use JMX to monitor SonarQube Server.

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

9

The SonarQube Platform is made of 4 components:

One **SonarQube Server** starting 3 main processes:

a **Web Server** for developers, managers to browse quality snapshots and configure the SonarQube instance

a **Search Server** based on Elastic search to back searches from the UI

a **Compute Engine Server** in charge of processing code analysis reports and saving them in the SonarQube Database

One **SonarQube Database** to store:

the configuration of the SonarQube instance (security, plugins settings, etc.)

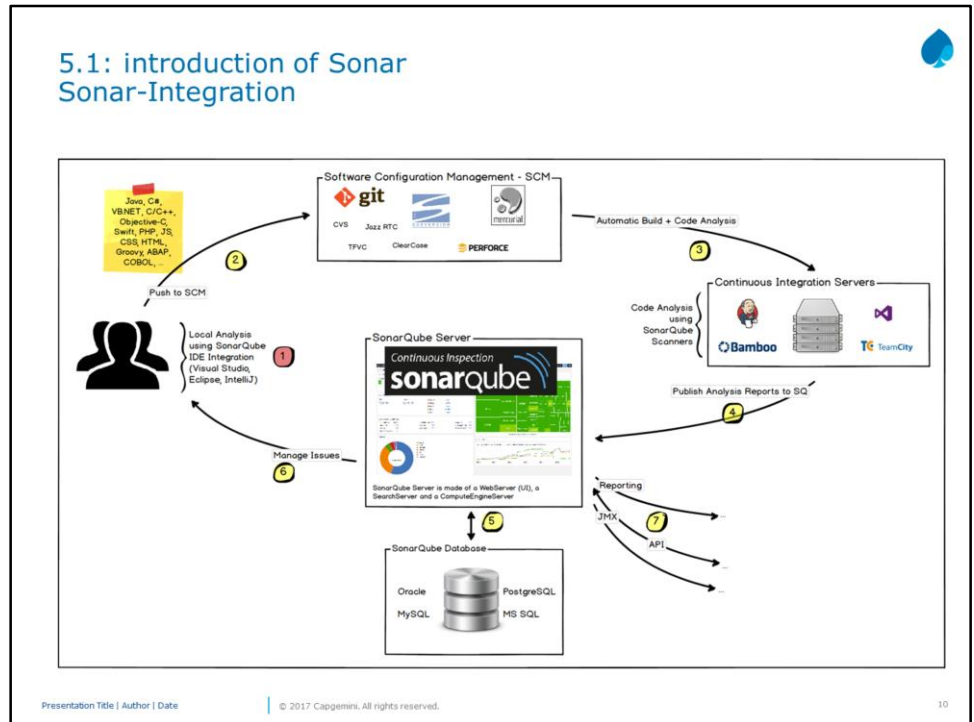
the quality snapshots of projects, views, etc.

Multiple **SonarQube Plugins** installed on the server, possibly including language, SCM, integration, authentication, and governance plugins

One or more **SonarQube Scanners** running on your Build / Continuous Integration Servers to analyze projects

Instructor Notes:

Give idea of flow

**About Machines and Locations**

The SonarQube Platform cannot have more than one SonarQube Server and one SonarQube Database.

For optimal performance, each component (server, database, scanners) should be installed on a separate machine, and the server machine should be dedicated.

SonarQube Scanners scale by adding machines.

All machines must be time synchronized.

The SonarQube Server and the SonarQube Database must be located in the same network

SonarQube Scanners don't need to be on the same network as the SonarQube Server.

There is **no communication** between **SonarQube Scanners** and the **SonarQube Database**.

Instructor Notes:

Add instructor notes here.

5.1: introduction of Sonar Sonar-Rules



254 rules written – identify atleast 10 rules

- "equals(Object obj)" and "hashCode()" should be overridden in pairs
- "final" classes should not have "protected" members
- "for" loop incrementers should modify the variable being tested in the loop's stop condition
- "Iterator.hasNext()" should not call "Iterator.next()"
- "Iterator.next()" methods should throw "NoSuchElementException"
- "main" should not "throw" anything
- "NullPointerException" should not be caught
- "entrySet()" should be iterated when both the key and value are needed

We can see all the rules in Sonar Dashboard

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

11

About Machines and Locations

The SonarQube Platform cannot have more than one SonarQube Server and one SonarQube Database.

For optimal performance, each component (server, database, scanners) should be installed on a separate machine, and the server machine should be dedicated.

SonarQube Scanners scale by adding machines.

All machines must be time synchronized.

The SonarQube Server and the SonarQube Database must be located in the same network

SonarQube Scanners don't need to be on the same network as the SonarQube Server.

There is **no communication** between **SonarQube Scanners** and the **SonarQube Database**.

Instructor Notes:

Give Demo example

5.1: introduction of Sonar Sonar Installation



Sonar is easy to install & use .

Download Sonar -Sonarqube-x.xx & Sonar-scanner-x.xx:

- <https://www.sonarqube.org/downloads/>

Sonar can be installed in different ways:

- As a standalone application
- Windows Service

For starting sonar server use -StartSonar.bat

For stopping sonar server use – StopSonar.bat

Once sonar is started, the sonar dash board can be accessed by giving the following link in the browser

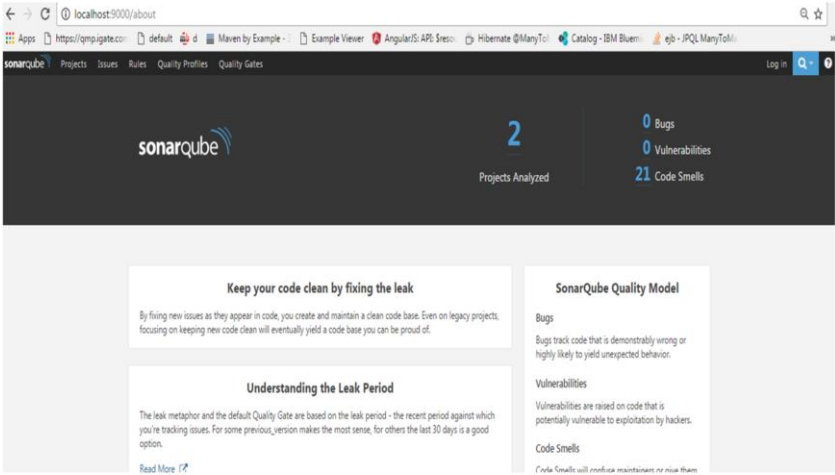
<http://localhost:9000/>

Add the notes here.

Instructor Notes:

Add instructor notes here.

5.1: introduction of Sonar
Sonar Installation



The screenshot displays the SonarQube web application interface. At the top, the title '5.1: introduction of Sonar' and 'Sonar Installation' are visible. Below the title, a navigation bar includes links for 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', and 'Quality Gates'. The main content area features a dark header with the SonarQube logo and a summary of project analysis: '2 Projects Analyzed', '0 Bugs', '0 Vulnerabilities', and '21 Code Smells'. Below this, three informational boxes are present: 'Keep your code clean by fixing the leak', 'Understanding the Leak Period', and 'SonarQube Quality Model'. The 'SonarQube Quality Model' box lists 'Bugs', 'Vulnerabilities', and 'Code Smells' with brief descriptions. The footer contains the text 'Presentation Title | Author | Date' and '© 2017 Capgemini. All rights reserved.' along with a page number '13'.

Add the notes here.

Instructor Notes:

Add instructor notes here.

5.2: Analyzing Java code with Sonar

Analyzing Java with Sonar



Integrating Java program with SonarQube

- Create a Java Project
- Add description of your project in sonar-scanner-x.xx->conf ->sonar-scanner.Properties
 - sonar.projectKey=JavaProject
 - sonar.projectName=JavaProject
 - sonar.projectVersion=1.0
 - sonar.sources=C:/DevOps/Training/JavaProject/src/com/cg/sonardemo
- Run Sonar server by using command **StartSonar.bat**
- Go to project folder & run command **sonar-scanner.bat**
- Open <http://localhost:9000/> & we can see code is analyzing

Add the notes here.

Instructor Notes:

Add instructor notes here.

5.2: Analyzing Java code with Sonar

Analyzing Java with Sonar

First Project run on <http://localhost:9000>

The screenshot shows the SonarQube web interface in a browser window. The address bar shows 'localhost:9000/about'. The SonarQube logo is in the top left. The main header shows '1 Projects Analyzed', '0 Bugs', '0 Vulnerabilities', and '11 Code Smells'. Below this, there are several informational cards: 'Works on localhost port 9000', 'Keep your code clean by fixing the leak', 'Understanding the Leak Period', 'SonarQube Quality Model', and 'Code have Analysis problem'. The footer contains 'Presentation Title | Author | Date' and '© 2017 Capgemini. All rights reserved.'.

Add the notes here.

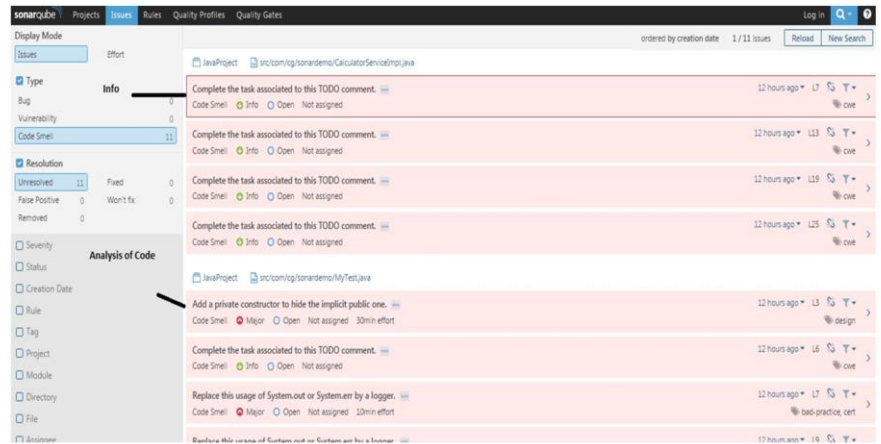
Instructor Notes:

Give Demo example

5.2: Analyzing Java code with Sonar

Analyzing Java with Sonar

Select code smell after log in ,you will get all kind of major and minor problems



Each issue has one of five severities:

BLOCKER

Bug with a high probability to impact the behavior of the application in production: memory leak, unclosed JDBC connection, The code **MUST** be immediately fixed.

CRITICAL

Either a bug with a low probability to impact the behavior of the application in production or an issue which represents a security flaw: empty catch block, SQL injection, ... The code **MUST** be immediately reviewed.

MAJOR

Quality flaw which can highly impact the developer productivity: uncovered piece of code, duplicated blocks, unused parameters, ...

MINOR

Quality flaw which can slightly impact the developer productivity: lines should not be too long, "switch" statements should have at least 3 cases, ...

INFO

Neither a bug nor a quality flaw, just a finding.

Technical Review

Confirm, False Positive, Won't Fix, Change Severity, and Resolve fall into this category, which presumes an initial review of an issue to verify its validity. Assume it's time to review the technical debt added in the last review period - whether that's a day, a week, or an entire sprint. You go through each new issue and do one:

Confirm - By confirming an issue, you're basically saying "Yep, that's a problem." Doing so moves it out of "Open" status to "Confirmed".

False Positive - Looking at the issue in context, you realize that for whatever reason, this issue isn't actually an issue, erm... "problem." It's not actually a problem. So you mark it False Positive and move on. Requires Administer Issues permission on the project.

Won't Fix - Looking at the issue in context, you realize that while it's a valid issue it's not one that actually needs fixing. In other words, it represents accepted technical debt. So you mark it Won't Fix and move on. Requires Administer Issues permission on the project.

Change Severity - This is the middle ground between the first two options. Yes, it's a problem, but it's not as bad a problem as the rule's default severity makes it out to be. Or perhaps it's actually far worse. Either way, you adjust the severity of the issue to bring it in line with what you feel it deserves. The marker in the drilldown will change to show the new severity immediately, but the change won't be reflected in your issue counts until after the next analysis. Requires Administer Issues permission on the project.

Resolve - If you think you've fixed an open issue, you can Resolve it. If you're right, the next analysis will move it to closed status. If you're wrong, its status will go to re-opened.

Instructor Notes:

Add instructor notes here.

5.2: Analyzing Java code with Sonar

Analyzing Java with Sonar

Rules to analyze Java Code

The screenshot shows the SonarQube interface for managing rules. The left pane allows filtering rules by language and type. The main area displays a list of rules with their descriptions, applicable languages, and associated tags. A red arrow points to the 'Rules' menu item in the top navigation bar.

Click on the top "Rules" menu item to enter the world of rules. By default, you will see all the available rules, with the ability to narrow the selection based on search criteria in the left pane:

Language: the language to which a rule applies.

Type: Bug, Vulnerability or Code Smell rules

Tag: it is possible to add tags to rules in order to classify them and to help discover them more easily.

Repository: the engine that contributes rules to SonarQube.

Default Severity: the original severity of the rule - as defined by the plugin that contributes this rule.

Status: rules can have 3 different statuses:

Beta: The rule has been recently implemented and we haven't gotten enough feedback from users yet, so there may be false positives or false negatives.

Deprecated: The rule should no longer be used because a similar, but more powerful and accurate rule exists.

Ready: The rule is ready to be used in production.

Available Since: date when a rule was first added on the SonarQube instance. This is useful to list all the new rules since the last upgrade of a plugin for instance.

Template: display rule templates that allow to create custom rules (see later on this page).

Quality Profile: inclusion in or exclusion from a specific profile

To see the details of a rule, either click on it, or use the right arrow key. Along with basic open data, you'll also be able to see which, if any, profiles it's active in and how many open issues have been raised with it.

The 2 following actions are available only if you have the right permissions ("Administer Quality Profiles and Gates"):

Add/Remove Tags:

It is possible to add existing tags on a rule, or to create new ones (just enter a new name while typing in the text field).

Note that some rules have built-in tags that you cannot remove - they are provided by the plugins which contribute the rules.

Extend Description:

Extending rule descriptions is useful to let users know how your organization is using a particular rule for instance or to give more insight on a rule.

Note that the extension will be available to non-admin users as a normal part of the rule details.

Instructor Notes:

Add instructor notes here.

5.3: Integrating Jenkin with Sonar Sonar Jenkin Integration

Download SonarQube Plugin in Jenkins

Go to Manage Jenkin->Configure System->Go to SonarQube servers->check on Enable injection of SonarQube->add Server name & server URL

Click on the top "Rules" menu item to enter the world of rules. By default, you will see all the available rules, with the ability to narrow the selection based on search criteria in the left pane:

Language: the language to which a rule applies.

Type: Bug, Vulnerability or Code Smell rules

Tag: it is possible to add tags to rules in order to classify them and to help discover them more easily.

Repository: the engine that contributes rules to SonarQube.

Default Severity: the original severity of the rule - as defined by the plugin that contributes this rule.

Status: rules can have 3 different statuses:

Beta: The rule has been recently implemented and we haven't gotten enough feedback from users yet, so there may be false positives or false negatives.

Deprecated: The rule should no longer be used because a similar, but more powerful and accurate rule exists.

Ready: The rule is ready to be used in production.

Available Since: date when a rule was first added on the SonarQube instance. This is useful to list all the new rules since the last upgrade of a plugin for instance.

Template: display rule templates that allow to create custom rules (see later on this page).

Quality Profile: inclusion in or exclusion from a specific profile

To see the details of a rule, either click on it, or use the right arrow key. Along with basic rule data, you'll also be able to see which, if any, profiles it's active in and how many open issues have been raised with it.

The 2 following actions are available only if you have the right permissions ("Administer Quality Profiles and Gates"):

Add/Remove Tags:

It is possible to add existing tags on a rule, or to create new ones (just enter a new name while typing in the text field).

Note that some rules have built-in tags that you cannot remove - they are provided by the plugins which contribute the rules.

Extend Description:

Extending rule descriptions is useful to let users know how your organization is using a particular rule for instance or to give more insight on a rule.

Note that the extension will be available to non-admin users as a normal part of the rule details.

Instructor Notes:

Add instructor notes here.

5.4: Analyzing Maven code ,Jenkin with Sonar Sonar,Maven,Git & Jenkins Integration



Create New item->Enter item name->Select Maven Project->Ok
Give Git Repository link, in build environment check prepare sonarqube scanner environment
Give path of pom.xml of your project & then select post build action as sonarqube analysis with maven
Then apply & Build now
We can see in console output build success and failure
Analyze in SonarQube

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

19

Click on the top "Rules" menu item to enter the world of rules. By default, you will see all the available rules, with the ability to narrow the selection based on search criteria in the left pane:

Language: the language to which a rule applies.

Type: Bug, Vulnerability or Code Smell rules

Tag: it is possible to add tags to rules in order to classify them and to help discover them more easily.

Repository: the engine that contributes rules to SonarQube.

Default Severity: the original severity of the rule - as defined by the plugin that contributes this rule.

Status: rules can have 3 different statuses:

Beta: The rule has been recently implemented and we haven't gotten enough feedback from users yet, so there may be false positives or false negatives.

Deprecated: The rule should no longer be used because a similar, but more powerful and accurate rule exists.

Ready: The rule is ready to be used in production.

Available Since: date when a rule was first added on the SonarQube instance. This is useful to list all the new rules since the last upgrade of a plugin for instance.

Template: display rule templates that allow to create custom rules (see later on this page).

Quality Profile: inclusion in or exclusion from a specific profile

To see the details of a rule, either click on it, or use the right arrow key. Along with basic rule data, you'll also be able to see which, if any, profiles it's active in and how many open issues have been raised with it.

The 2 following actions are available only if you have the right permissions ("Administer Quality Profiles and Gates"):

Add/Remove Tags:

It is possible to add existing tags on a rule, or to create new ones (just enter a new name while typing in the text field).

Note that some rules have built-in tags that you cannot remove - they are provided by the plugins which contribute the rules.

Extend Description:

Extending rule descriptions is useful to let users know how your organization is using a particular rule for instance or to give more insight on a rule.

Note that the extension will be available to non-admin users as a normal part of the rule details.

Instructor Notes:

Give Demo example

Click on the top "Rules" menu item to enter the world of rules. By default, you will see all the available rules, with the ability to narrow the selection based on search criteria in the left pane:

Language: the language to which a rule applies.

Type: Bug, Vulnerability or Code Smell rules

Tag: it is possible to add tags to rules in order to classify them and to help discover them more easily.

Repository: the engine that contributes rules to SonarQube.

Default Severity: the original severity of the rule - as defined by the plugin that contributes this rule.

Status: rules can have 3 different statuses:

Beta: The rule has been recently implemented and we haven't gotten enough feedback from users yet, so there may be false positives or false negatives.

Deprecated: The rule should no longer be used because a similar, but more powerful and accurate rule exists.

Ready: The rule is ready to be used in production.

Available Since: date when a rule was first added on the SonarQube instance. This is useful to list all the new rules since the last upgrade of a plugin for instance.

Template: display rule templates that allow to create custom rules (see later on this page).

Quality Profile: inclusion in or exclusion from a specific profile

To see the details of a rule, either click on it, or use the right arrow key. Along with basic rule data, you'll also be able to see which, if any, profiles it's active in and how many open issues have been raised with it.

The 2 following actions are available only if you have the right permissions ("Administer Quality Profiles and Gates"):

Add/Remove Tags:

It is possible to add existing tags on a rule, or to create new ones (just enter a new name while typing in the text field).

Note that some rules have built-in tags that you cannot remove - they are provided by the plugins which contribute the rules.

Extend Description:

Extending rule descriptions is useful to let users know how your organization is using a particular rule for instance or to give more insight on a rule.

Note that the extension will be available to non-admin users as a normal part of the rule details.

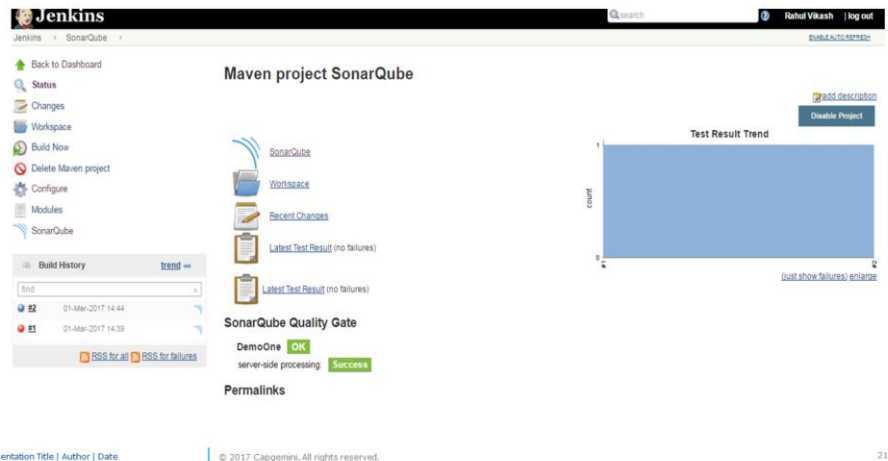
Instructor Notes:

Give Demo example

5.4: Analyzing Maven code ,Jenkin with Sonar Sonar,Maven,Git & Jenkins Integration

After successful completion, sonarqube analysis can be checked.

- Click on sonarqube or Open <http://localhost:9000/> & code analyzing is seen.



Click on the top "Rules" menu item to enter the world of rules. By default, you will see all the available rules, with the ability to narrow the selection based on search criteria in the left pane:

Language: the language to which a rule applies.

Type: Bug, Vulnerability or Code Smell rules

Tag: it is possible to add tags to rules in order to classify them and to help discover them more easily.

Repository: the engine that contributes rules to SonarQube.

Default Severity: the original severity of the rule - as defined by the plugin that contributes this rule.

Status: rules can have 3 different statuses:

Beta: The rule has been recently implemented and we haven't gotten enough feedback from users yet, so there may be false positives or false negatives.

Deprecated: The rule should no longer be used because a similar, but more powerful and accurate rule exists.

Ready: The rule is ready to be used in production.

Available Since: date when a rule was first added on the SonarQube instance. This is useful to list all the new rules since the last upgrade of a plugin for instance.

Template: display rule templates that allow to create custom rules (see later on this page).

Quality Profile: inclusion in or exclusion from a specific profile

To see the details of a rule, either click on it, or use the right arrow key. Along with basic rule data, you'll also be able to see which, if any, profiles it's active in and how many open issues have been raised with it.

The 2 following actions are available only if you have the right permissions ("Administer Quality Profiles and Gates"):

Add/Remove Tags:

It is possible to add existing tags on a rule, or to create new ones (just enter a new name while typing in the text field).

Note that some rules have built-in tags that you cannot remove - they are provided by the plugins which contribute the rules.

Extend Description:

Extending rule descriptions is useful to let users know how your organization is using a particular rule for instance or to give more insight on a rule.

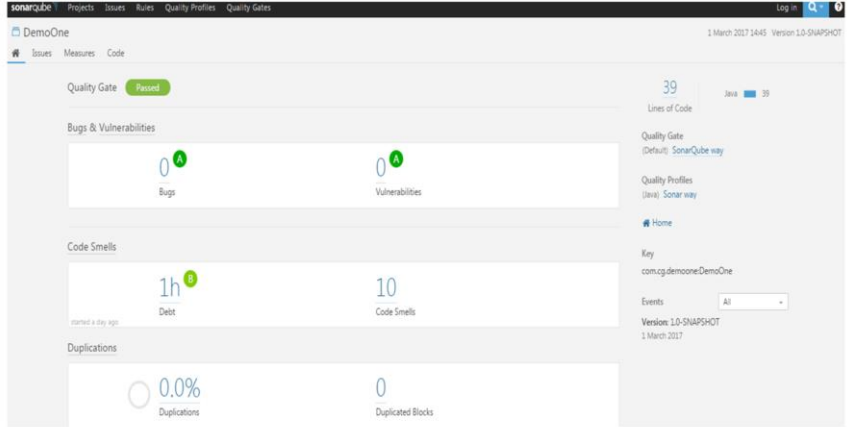
Note that the extension will be available to non-admin users as a normal part of the rule details.

Instructor Notes:

Give Demo example

5.4: Analyzing Maven code ,Jenkin with Sonar Sonar,Maven,Git & Jenkins Integration

Clicking on SonarQube & analyzing the code



The screenshot displays the SonarQube web interface for a project named 'DemoOne'. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, and Quality Gates. The main dashboard shows a 'Quality Gate' status of 'Passed'. Below this, there are sections for 'Bugs & Vulnerabilities', 'Code Smells', and 'Duplications'. The 'Bugs & Vulnerabilities' section shows 0 bugs and 0 vulnerabilities. The 'Code Smells' section shows 1h (1 hour) of debt and 10 code smells. The 'Duplications' section shows 0.0% duplications and 0 duplicated blocks. On the right side, there is a sidebar with 'Lines of Code' (39), 'Quality Gate' (Default: SonarQube way), 'Quality Profiles' (Java: Sonar way), and a 'Key' field with the value 'com.cgdemoone:DemoOne'. The bottom of the interface shows the version '1.0-SNAPSHOT' and the date '1 March 2017'.

Click on the top "Rules" menu item to enter the world of rules. By default, you will see all the available rules, with the ability to narrow the selection based on search criteria in the left pane:

Language: the language to which a rule applies.

Type: Bug, Vulnerability or Code Smell rules

Tag: it is possible to add tags to rules in order to classify them and to help discover them more easily.

Repository: the engine that contributes rules to SonarQube.

Default Severity: the original severity of the rule - as defined by the plugin that contributes this rule.

Status: rules can have 3 different statuses:

Beta: The rule has been recently implemented and we haven't gotten enough feedback from users yet, so there may be false positives or false negatives.

Deprecated: The rule should no longer be used because a similar, but more powerful and accurate rule exists.

Ready: The rule is ready to be used in production.

Available Since: date when a rule was first added on the SonarQube instance. This is useful to list all the new rules since the last upgrade of a plugin for instance.

Template: display rule templates that allow to create custom rules (see later on this page).

Quality Profile: inclusion in or exclusion from a specific profile

To see the details of a rule, either click on it, or use the right arrow key. Along with basic rule data, you'll also be able to see which, if any, profiles it's active in and how many open issues have been raised with it.

The 2 following actions are available only if you have the right permissions ("Administer Quality Profiles and Gates"):

Add/Remove Tags:

It is possible to add existing tags on a rule, or to create new ones (just enter a new name while typing in the text field).

Note that some rules have built-in tags that you cannot remove - they are provided by the plugins which contribute the rules.

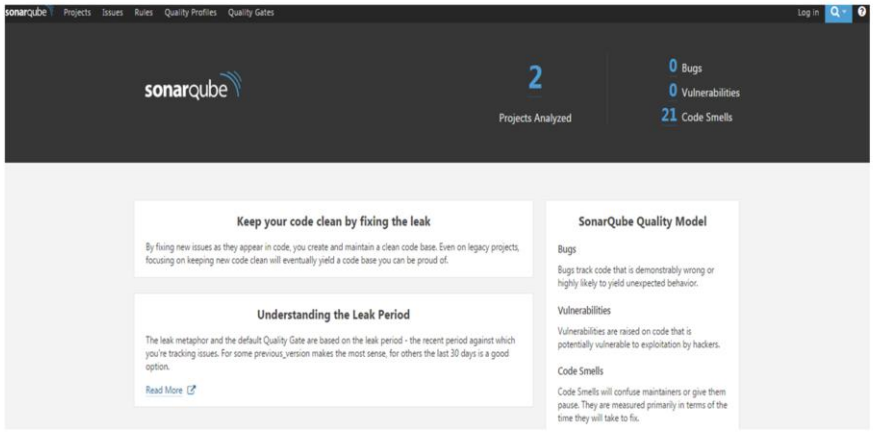
Extend Description:

Extending rule descriptions is useful to let users know how your organization is using a particular rule for instance or to give more insight on a rule.

Note that the extension will be available to non-admin users as a normal part of the rule details.

Instructor Notes:

Give Demo example



5.4: Analyzing Maven code ,Jenkin with Sonar
Sonar,Maven,Git & Jenkins Integration

Open <http://localhost:9000/>

The screenshot shows the SonarQube web interface. At the top, there's a navigation bar with links for Projects, Issues, Rules, Quality Profiles, and Quality Gates. The main dashboard displays the SonarQube logo, a large number '2' indicating 'Projects Analyzed', and counts for '0 Bugs', '0 Vulnerabilities', and '21 Code Smells'. Below this, there are three informational cards: 'Keep your code clean by fixing the leak', 'Understanding the Leak Period', and 'SonarQube Quality Model'. The footer shows 'Presentation Title | Author | Date' and '© 2017 Capgemini. All rights reserved.' with a page number '23'.

Click on the top "Rules" menu item to enter the world of rules. By default, you will see all the available rules, with the ability to narrow the selection based on search criteria in the left pane:

Language: the language to which a rule applies.

Type: Bug, Vulnerability or Code Smell rules

Tag: it is possible to add tags to rules in order to classify them and to help discover them more easily.

Repository: the engine that contributes rules to SonarQube.

Default Severity: the original severity of the rule - as defined by the plugin that contributes this rule.

Status: rules can have 3 different statuses:

Beta: The rule has been recently implemented and we haven't gotten enough feedback from users yet, so there may be false positives or false negatives.

Deprecated: The rule should no longer be used because a similar, but more powerful and accurate rule exists.

Ready: The rule is ready to be used in production.

Available Since: date when a rule was first added on the SonarQube instance. This is useful to list all the new rules since the last upgrade of a plugin for instance.

Template: display rule templates that allow to create custom rules (see later on this page).

Quality Profile: inclusion in or exclusion from a specific profile

To see the details of a rule, either click on it, or use the right arrow key. Along with basic rule data, you'll also be able to see which, if any, profiles it's active in and how many open issues have been raised with it.

The 2 following actions are available only if you have the right permissions ("Administer Quality Profiles and Gates"):

Add/Remove Tags:

It is possible to add existing tags on a rule, or to create new ones (just enter a new name while typing in the text field).

Note that some rules have built-in tags that you cannot remove - they are provided by the plugins which contribute the rules.


Extend Description:

Extending rule descriptions is useful to let users know how your organization is using a particular rule for instance or to give more insight on a rule.

Note that the extension will be available to non-admin users as a normal part of the rule details.

Instructor Notes:

Add instructor notes here.



Demo

Analyze Java code with Sonar
Jenkins Maven Git integration & analyzing with sonar

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

24

Add the notes here.

Instructor Notes:

Add instructor notes here.

Lab

Lab 03

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

25

Add the notes here.

Instructor Notes:

Add instructor notes here.

Summary



Sonar is an open source platform used by development teams to manage source code quality. Sonar has been developed with a main objective in mind: make code quality management accessible to everyone with minimal effort.

Working with code analyzing tool with Maven Jenkins, Git

Add the notes here.

Instructor Notes:

Q1.All of above.

Q2. Sonarqube

Q3StartSonar.bat

Review Question



SonarQube platform is made of components, choose the correct one

- Database
- plugins
- Server
- All of above

_____ plugin needs to be downloaded for Jenkins and sonar integration.

■ _____ command is used to run Sonar software.

Add the notes here.