<Course Name>   <Lesson Name>

Instructor Notes:

Add instructor notes here.

DevOps

Lesson 04-Sonar(SonarQube)

<Course Name> <Lesson Name>

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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

2

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.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. 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".

<Course Name>   <Lesson Name>

Instructor Notes:

Add instructor notes here.

5.1: introduction of Sonar
Features of Sonar

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

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!

<Course Name>   <Lesson Name>

Instructor Notes:

Add instructor notes
here.

5.1: introduction of Sonar
Features of Sonar

DevOps Integration
Build Systems
For dynamic languages like JavaScript, PHP, Python, … executing an analysis
is as easy as feeding SonarQube with a bunch of source files. But for
languages like Java, C#, C, C++ and Objective-C, there is simply no way to
provide accurate results without being part of the build. That's why built-in
integrations are provided for MSBuild, Maven, Gradle, Ant, and Makefiles.
CI Engines
Native integrations with build systems let you easily schedule the execution of
an analysis from all CI engines: Jenkins, VSTS, TFS, Travis-CI… Don't worry
if your CI engine isn't listed here, integration effort will be minimum.
Pass/Fail Notification
Once an analysis is done, a report is sent to the SonarQube server to be
integrated. At the end of this integration, a standard webhook mechanism lets
you notify any external system to do whatever you want: trigger an alarm,
update a wallboard, notify a chat room.
Full Web API
As part of the overall development ecosystem, the SonarQube Web API can
be used to automatically provision a SonarQube project, feed a BI tool, monitor
SonarQube, etc. Morever the list and definition of all the Web API is built in
SonarQube.
Promotion Pipelines
Using webhooks, SonarQube can be integrated as a promotion step in your
delivery pipelines. This way, you can make sure that only artifacts that pass
the Quality Gate will be released and deployed to production.

Instructor Notes:

Add instructor notes here.

5.1: introduction of Sonar
Features of Sonar

Centralized Quality
All projects in one place
Getting everyone on a team on the same page about quality is hard enough. What happens when you expand the scope to a department or an entire organization? SonarQube enables you to centralize and scale a single vision of code quality.
Shared rulesets
SonarQube offers a central place to view and define the rules used during analysis of projects. These rulesets are organized in quality profiles. Every member of the organization can see which rules are applied to their project. Every project administrator can choose which quality profile is used for the project.
Unified Quality Gate
SonarQube provides out-of-the-box a default Quality Gate focusing on the Leak concept. This means that the same requirements will be applied across the board to every project - greenfield and legacy; in-house, out-source and off-shore.
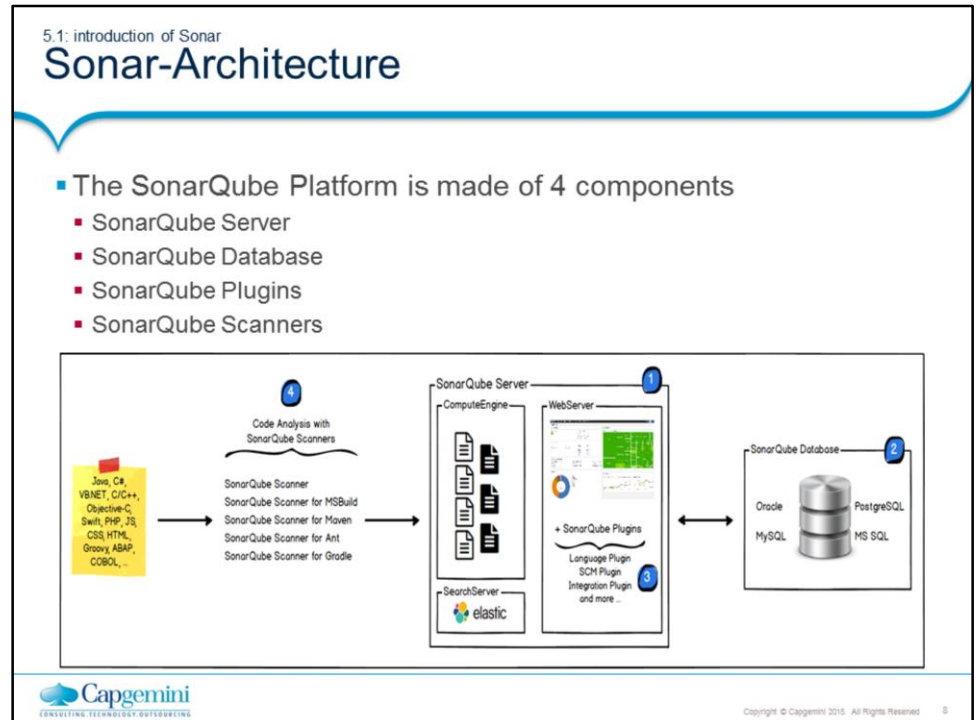Cross projects services
Most services are available cross projects. For example, as a developer you can use the issues service to get all new issues assigned to you - across projects - so that you can concentrate on your work. As a technical lead, the projects page lists all your favorite projects and lets you explore them on different axes.
Support 20+ Langauges java, C, C++, Cobol, Flex, C#, PHP, Phython ,Swift Objective C ,web,ABAP

Instructor Notes:

How sonar works give idea.

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.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

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.)
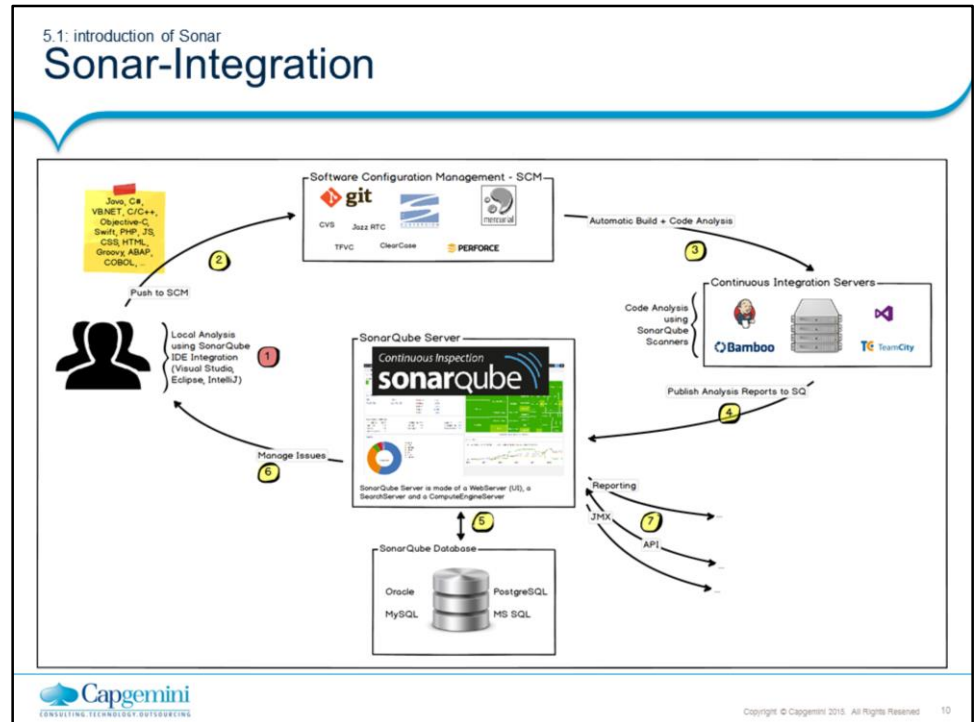> the quality snapshots of projects, views, etc.

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**.

<Course Name>   <Lesson Name>

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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. 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/

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved   12

Add the notes here.

<Course Name>   <Lesson Name>

Instructor Notes:

Add instructor notes
here.

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.

<Course Name>   <Lesson Name>

Instructor Notes:

Add instructor notes here.

5.2: Analyzing Java code with Sonar
Analyzing Java with Sonar

- First Project run on http://localhost:9000

sonarqube   Projects   Issues   Rules   Quality Profiles   Quality Gates   Log in

sonarqube

1

0 Bugs
0 Vulnerabilities
11 Code Smells

Projects Analyzed

Works on localhost port 9000

Keep your code clean by fixing the leak

By fixing new issues as they appear in code, you create and maintain a clean code base. Even on legacy projects, focusing on keeping new code clean will eventually yield a code base you can be proud of.

SonarQube Quality Model

Bugs

Bugs track code that is demonstrably wrong or highly likely to yield unexpected behavior.

Vulnerabilities

Vulnerabilities are raised on code that is potentially vulnerable to exploitation by hackers.

Code Smells

Understanding the Leak Period

The leak metaphor and the default Quality Gate are based on the leak period - the recent period against which you're tracking issues. For some previous_version makes the most sense, for others the last 30 days is a good option.

Project Analyzed is One

Code have Analysis problem

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

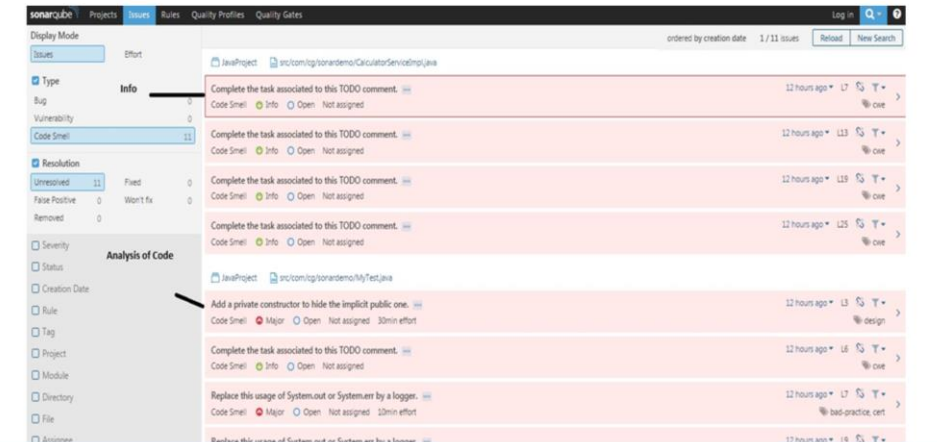Copyright © Capgemini 2015. All Rights Reserved   15

Add the notes here.

Instructor Notes:

Give Demo example

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.
.

<Course Name>   <Lesson Name>

Instructor Notes:

Add instructor notes here.

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.

<Course Name>   <Lesson Name>

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.
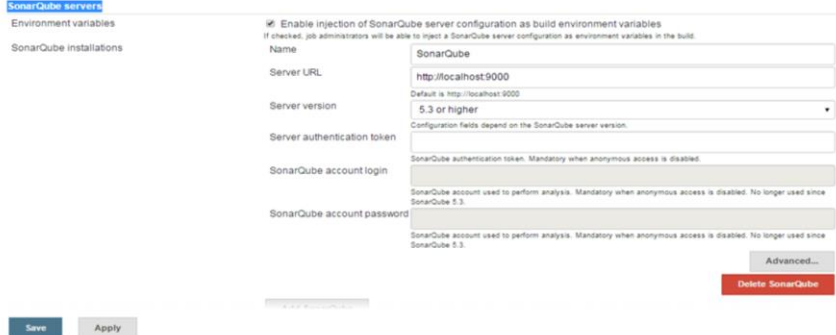
Instructor Notes:

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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

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.

<Course Name>   <Lesson Name>

Instructor Notes:

Give Demo example

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

Build Environment

☑ Prepare SonarQube Scanner environment   ❓

Pre Steps

[Add pre-build step ▾]   [Pre Steps]

Build

Root POM   DemoOne/pom.xml   ❓
Goals and options   clean package   ❓
                                                                        [Advanced...]

Post Steps

○ Run only if build succeeds ○ Run only if build succeeds or is unstable ◉ Run regardless of build result
Should the post-build steps run only for successful builds, etc.

[Add post-build step ▾]

Build Settings
[Save] [Apply]

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved   20

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.

<Course Name>   <Lesson Name>

Instructor Notes:

Give Demo example

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

- Clicking on SonarQube & analyzing the code

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.
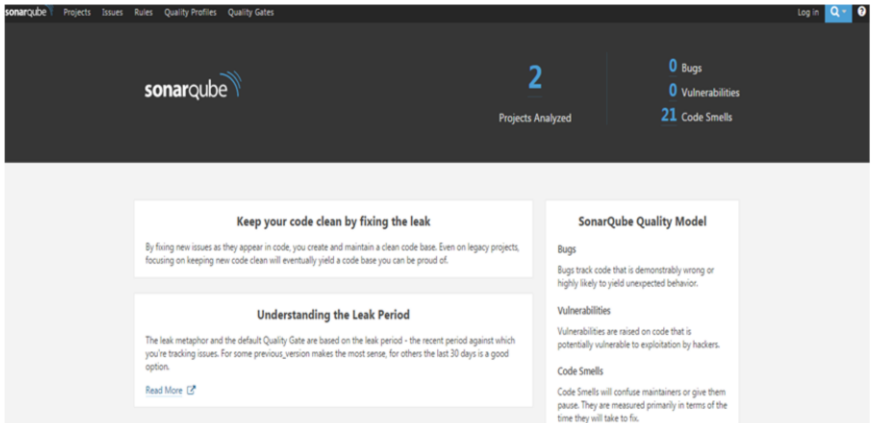
Instructor Notes:

Give Demo example

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

- Open http://localhost:9000/

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

Demo

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved   24

Add the notes here.

Instructor Notes:

Add instructor notes here.

Lab

- Lab 03

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved   25

Add the notes here.

<Course Name>   <Lesson Name>

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

Summary

Add the notes here.

<Course Name>   <Lesson Name>

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.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Add the notes here.