Release 4.2 Upgrade Notes

Multi-language Analysis

Mono-language / Multi-language Analysis Mode

When the `sonar.language` property is set, the behavior remains exactly the same as for versions prior to 4.2: the multi-language analysis mode is not activated. However, as a consequence of the change to multi-language analysis, "language" will no longer be recorded for a project. This means that language will no longer be displayed in the description widget, nor will it be available in the Measures Service for project searches.

Note that the `sonar.language` property doesn't have a default value anymore (was previously set to `java` by default). It means that your Java projects may not explicitly set this property to `java`. Therefore, to keep analyzing in the mono-language mode, make sure to explicitly set this property to `java`.

Multi-language Analysis

1. Do not set the `sonar.language` property.
2. Set the `sonar.sources` property to the parent directory containing all your source code.
3. Run an analysis

Language plugins compatible with multi-language
Note that the following plugins are currently compatible with multi-language:

- CSS 0.1+
- Erlang 0.2+
- Flex 2.0.1+
- Groovy 1.0+
- Java 2.1+
- JavaScript 1.6+
- PHP 2.0+
- PL/SQL 2.6+
- Web 2.2+
- XML 1.1

Converting a Mono-language Project to a Multi-language Project

Let's take as an example a project containing both Java and JavaScript source code. Your SonarQube instance currently contains two different projects: one for the Java source code and one for the JavaScript source code. Optionally, you may also have created a view to aggregate these two projects.

The first step is to choose which one of these two mono-language projects you will convert to a multi-language project. You will lose the history (timeline, false positives, action plans, etc.) on the one that won't get converted to a multi-language project. In this example, we'll choose to convert the Java project to a multi-language project as most of our code (and therefore history) is Java.

The second step is to run another analysis of this Java project the old way (make sure to explicitly set the `sonar.language` property to `java`). This step is mandatory to keep the history on the project.

The third and last step is to remove the `sonar.language` property and set the `sonar.sources` property to the parent directory containing all your source code (Java + JavaScript). You can now run another analysis. You will finally be able to browse your first multi-language project!

SonarQube Java 2.1

SonarQube 4.2 is compatible with SonarQube Java 2.1, and is not backwards-compatible with previous versions of the Java Ecosystem. Conversely, SonarQube Java 2.1 is compatible with SonarQube 4.2, and not with previous versions.

SonarQube Java 2.1 is embedded in SonarQube 4.2, and no longer includes the Checkstyle and PMD plugins. Therefore, you should install those two plugins if you're still using some of their rules.

Technical Accounts

When using an external authentication mechanism like LDAP, the `sonar.security.localUsers` property must be set in *$SONARQUBE_HOME/conf/sonar.properties* to the list of all the technical accounts (comma-separated list). These accounts will not be authenticated against LDAP but against the SonarQube engine. Conversely, all accounts not flagged as local will only be authenticated against the external authentication, although the "fallback" to the SonarQube database authentication will still be available when LDAP, for instance, is unavailable.

By default `admin` is a technical account.

Exclusions

Because the component keys are computed differently in SonarQube 4.2, you might have to update your exclusions. See Narrowing the Focus.

The `sonar.jdbc.schema` property is deprecated

Read Installing to configure SonarQube without using this property.