|  |
| --- |
|  |
| DevOps – Static code analysis |

|  |
| --- |
|  |

# Introduction

Static code analysis helps to find code violation at early stage of the development. Integrating static code analysis with continuous integration platform enables to monitor code quality continuously.

SonarQube is well known open-source platform for static code analysis. SonarQube has two major components, one **SonarQube server,** which collects analyzed data and renders them in dashboards. **Sonar Scanner** is another component, which works to analyze source code and sends data to Sonar server.

SonarQube has multiple different programming language plugins to understand and scan code quality. Sonar scanner can reside on the same environment where source code is being built.

# Setup Sonar Scanner

* Download Sonar scanner

Download Sonar Scanner for macOS at : <https://sonarsource.bintray.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-3.0.3.778-macosx.zip>

* Expand the downloaded file into the directory of your choice.
* Update the global settings to point to your SonarQube server by editing *<install\_directory>/conf/sonar-scanner.properties*:

#----- Default SonarQube server

sonar.host.url=http://<SONARQUBE\_SERVER\_URL>:9000

* Add the *<install\_directory>/bin* directory to your path.

# Configure required Sonar plugins

For iOS and Android Mobile applications we mainly need Java and Swift plugin respectively to scan source code quality.

Java plugin by default available with SonarQube server whereas Swift is paid plugin hence, it was decided to configure any open-source plugin for Swift applications.

There is third party [Sonar-swift plugin](https://github.com/Backelite/sonar-swift) available which can be used for Swift code analysis. There are multiple other tools required to be installed in order to work this plugin to work with SonarQube. Steps to install all required tools are mentioned at <https://github.com/Backelite/sonar-swift>. It is recommended to follow these steps.

# Sonar project properties

Create sonar project properties file which will be read by Sonar Scanner to execute code analysis. This file should be kept at root directory of the project.

Below is the *sonar-project.properties* file for iOS application



Below are important outline properties.

# Project key will also be used for binary file

sonar.projectKey=AutoLayout

sonar.projectName=AutoLayout

sonar.projectVersion=1.0

# Comment if you have a project with mixed ObjC / Swift

sonar.language=swift

# Configuration to use for your scheme. if you do not specify that the default will be Debug

sonar.swift.appConfiguration=Debug

# Xcode project configuration (.xcodeproj)

# and use the later to specify which project(s) to include in the analysis (comma separated list)

# Specify either xcodeproj or xcodeproj + xcworkspace

sonar.swift.project=AutoLayoutAndSizeClasses.xcodeproj

Below is *sonar-project.properties* for Android application

sonar.web.host=<SONARQUBE\_SERVER>

sonar.web.port=10000

sonar.projectKey=androidtodoapp

sonar.projectName=Android-ToDoApp

sonar.projectVersion=1.0

sonar.sources=app/src

sonar.language=java

sonar.sourceEncoding=UTF-8

# Sonar integration with CI

To achieve continuous code quality analysis, sonar scan step needs to be integrated with continuous integration build build pipeline.

*Jenkinsfile* defines our build pipeline hence below stage should be added to Jenkinsfile pipeline.

stage('Static analysis') {

steps {

dir('AutoLayoutAndSizeClasses') {

echo "Analyse application source code"

sh 'lizard --xml "AutoLayoutAndSizeClasses" > sonar-reports/lizard-report.xml'

withSonarQubeEnv('SonarQube') {

sh '~/DevOps/Tools/sonar-scanner-3.0.3.778-macosx/bin/sonar-scanner'

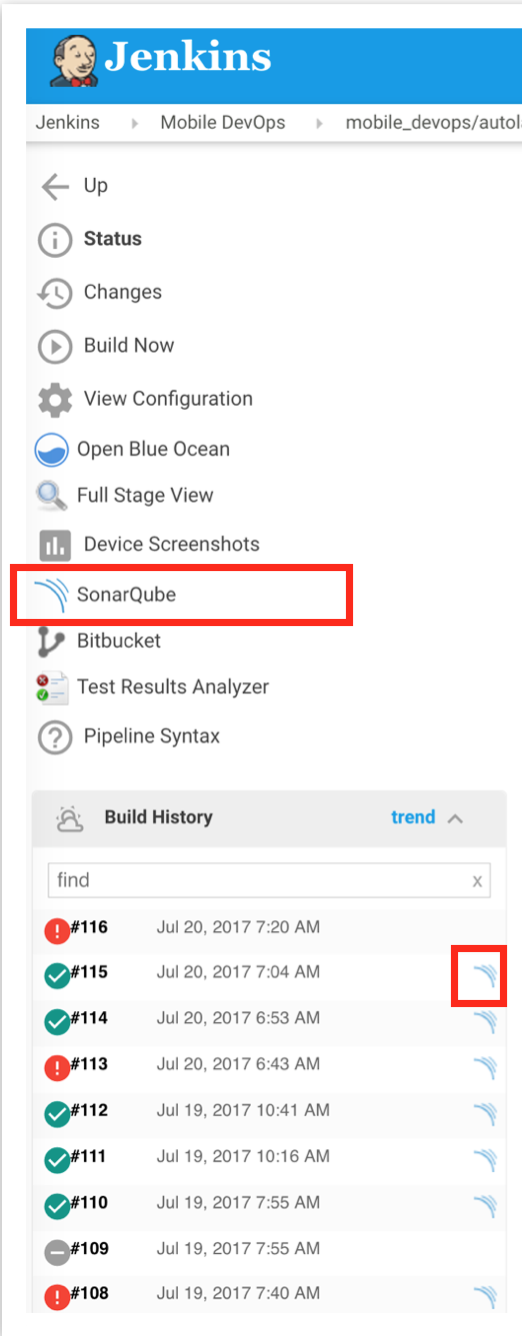
}

}

}

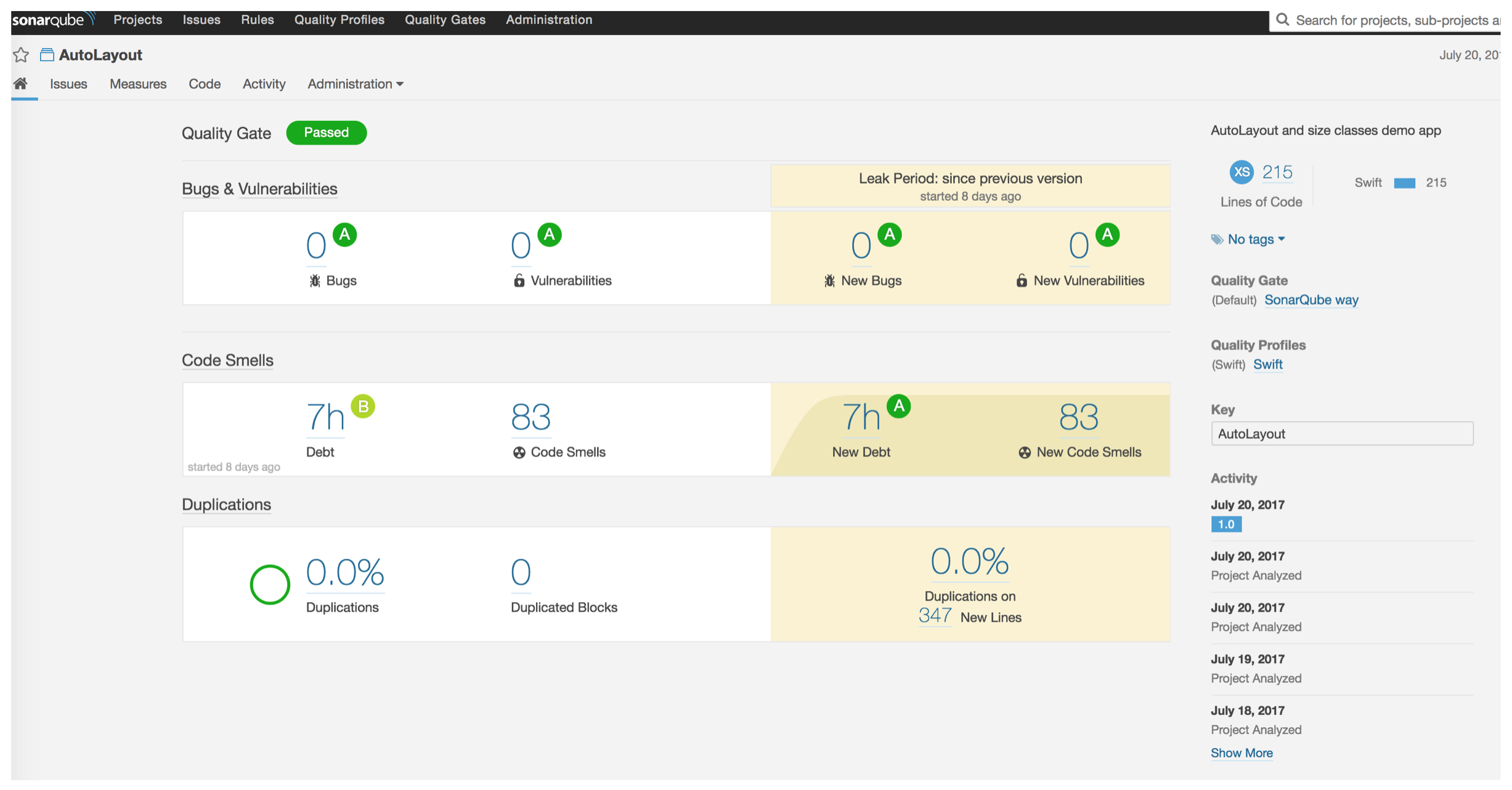
}

Now when pipeline executes, sonarqube static analysis step would be executed and link will be available to directly naviagte to SonarQube server project dashabord.



# SonarQube useful dashboards

## SonarQube dashboard



## Sonar issues details view

