

# Installation

The Blueriq Model Analyzer consists out of three components: Blueriq Sonar Plugin, Blueriq Quality Gate, Blueriq Sonar Scanner. Only the Blueriq Sonar Plugin and the Blueriq Quality Gate have to installed before being analyzing your Blueriq Models. The Blueriq Sonar Scanner is used to execute the analysis and therefor has to be made available on the environment from where you want to execute the analysis of you Blueriq Models.

- [Installation requirements](#)
- [Blueriq Sonar Plugin](#)
  - [Installing Sonarqube](#)
  - [Installing Blueriq Sonar Plugin in Sonarqube](#)
  - [Running SonarQube](#)
  - [Logging in in SonarQube](#)
  - [Adding a license](#)
- [Blueriq Quality Gate](#)
  - [Adding Blueriq Quality Gate](#)
- [Blueriq Sonar Scanner](#)
  - [Where to place the Sonar Scanner](#)
- [Performing an Analysis](#)
- [Enforcing the Quality Gate in a Jenkins pipeline](#)

## Installation requirements

All of the components require the Java to be available on the environment from where the components are ran. You can either install Java with a Java Installer, which makes Java globally available on your environment, or you can extract Java to a specific directory. This installation guide will provide information for both options. Make sure you download the correct Java version (32/64bit). See the [Blueriq Model Analyzer Platform Support](#) page which version of Java is support for your Blueriq Model Analyzer version.

## Blueriq Sonar Plugin

The Blueriq Sonar Plugin is a plugin for SonarQube, which performs the analysis on your Blueriq Models created in the Blueriq Studio. The Blueriq Sonar Plugin needs to be placed inside a SonarQube instance. This installation guide will provide information on how to install SonarQube and how to place the Blueriq Sonar Plugin in the correct directory of Sonarqube so it becomes available.

## Installing Sonarqube

Download SonarQube from <https://www.sonarqube.org/downloads/>.

Select the supported version mentioned in [Blueriq Model Analyzer Platform Support](#), and store it on your machine. Unzip the downloaded zip-file to the destination directory of your choice. The contents of the unzipped directory should look like this:

Name	Date modified	Type	Size
bin	05/09/2023 07:34	File folder	
conf	05/09/2023 07:34	File folder	
data	05/09/2023 07:34	File folder	
elasticsearch	05/09/2023 07:41	File folder	
extensions	05/09/2023 07:34	File folder	
lib	05/09/2023 07:41	File folder	
logs	05/09/2023 07:34	File folder	
temp	05/09/2023 07:34	File folder	
web	05/09/2023 07:42	File folder	
COPYING	05/09/2023 07:34	File	8 KB
dependency-license.json	05/09/2023 07:36	JSON File	76 KB

### Configuring Java

If you do not have Java globally available on your system, you will need to set the path to the installation directory of Java. This can be done by editing the wrapper.conf file which is located in the conf directory of your SonarQube installation directory.

Below is an example of how the wrapper.conf should look when you Java installation directory is located here: C:\Program Files\Java\jdk-11.0.2

#### wrapper.conf

```
# Path to JVM executable. By default it must be available in PATH.
# Can be an absolute path, for example:
#wrapper.java.command=/path/to/my/jdk/bin/java
wrapper.java.command=C:/Program Files/Java/jdk-11.0.2/bin/java

...
```

## Installing Blueriq Sonar Plugin in Sonarqube

The Blueriq Sonar Plugin called bma-sonar-plugin.jar needs to be placed inside the installation directory of SonarQube. Navigate to the subdirectory extensions/plugins inside the installation directory of SonarQube, and copy the bma-sonar-plugin.jar into the subdirectory.



Only copy the bma-sonar-plugin.jar to extensions/plugins. If any non SonarQube plugin jars are placed into the extensions/plugins directory, Sonarqube will not start.

## Running SonarQube

When the Blueriq Sonar Plugin is in place, start the SonarQube service by running the StartSonar.bat in the subdirectory of the bin directory corresponding to your operating system. For instance /sonarqube/bin/windows-x86-x64

## Logging in in SonarQube

To proceed you have to login in SonarQube, which by default is available at localhost:9000. An Administrator can manage tokens on a user's behalf via **Administration > Security > Users**

When installing SonarQube, a default user with Administer System permission is created automatically:

- Login: admin
- Password: admin

## Adding a license

In SonarQube, go to Administration Blueriq, and put the content of your license.aql file into the license textbox. It is also possible to include the license key in the application.properties of the BMA (see also: [Configuring Sonar Scanner](#)).

## Blueriq Quality Gate

The Blueriq Quality Gate is a SonarQube Quality Gate, more information about quality gates can be found [here](#).

## Adding Blueriq Quality Gate

The Blueriq Quality Gate called bma-qualitygate-installer.jar can be runned by using java -jar bma-qualitygate-installer.jar. When running this command you will be prompted with several questions.

Prompt	Input
hostname	The url to your SonarQube instance. This should be either yoururl.here[:PORT] or http[s]://yoururl.here[:PORT]/
username	A username for a user with the <b>Administer Quality Gates</b> permission. (default: admin)
password	The password for the above mentioned user. (default: admin)
Clean install? (y/n)	Decide whether to delete the current quality gate and overwrite it with a new one. If the answer is not "Y", the script will only install missing conditions.

```
Hostname:
localhost:9000
username:
admin
password:

Clean install? (y/n)
y
```

#### command line arguments

It is possible to pass command line arguments into the `bma-qualitygate-installer.jar` for information see [Quality Gate Installer](#).

During the installation of the Blueriq Quality Gate you will be informed of the progress. After the Quality Gate installation is finished, you can verify that the Blueriq Quality Gate is present in SonarQube. This can be done by navigating to the Quality Gates tab in SonarQube.

## Blueriq Sonar Scanner

The Blueriq Sonar Scanner is a SonarQube Scanner, which is a tool to analyze your projects. For Blueriq we have introduced our own Blueriq Sonar Scanner, due to implications we cannot provide an analysis using the default SonarQube Scanners. The Blueriq Sonar Scanner is a Spring Boot Java Command Line Interface runner, which can be used to execute the analysis of your Blueriq Models.

## Where to place the Sonar Scanner

Commonly a quality analysis is performed on Continuous Integration(CI) Servers like for instance Jenkins. With older versions of the Blueriq Model Analyzer it was possible to use the Maven Sonar Scanner to execute an analysis, and therefore only a connection to a central maven repository had to be setup. From the Blueriq Model Analyzer version 3 or higher you will need to place the Blueriq Sonar Scanner called `bma-sonar-scanner.jar` on the environment you wish to execute the analysis from.

For example if you want to execute an analysis on your CI Server, you will need to place the `bma-sonar-scanner.jar` somewhere on the system of your CI Server. If you want to execute an analysis on locally on your own environment, you will need to place the `bma-sonar-scanner.jar` somewhere on your local environment.

## Performing an Analysis

See the [User documentation](#) on how to execute an analysis of your Blueriq Models.

## Enforcing the Quality Gate in a Jenkins pipeline

To enforce the Quality Gate in Jenkins you can use the [SonarQube Scanner](#) plugin. To configure the BMA you can follow the provided example configuration of this plugin. The only way the BMA configuration deviates from this example is in the way it triggers the analysis.

Instead of using the `"sh "mvn clean package sonar:sonar""` command, use the `"java -jar <location on disk>\bma-sonar-scanner.jar"` command to trigger the analysis.

It may occur that after the analysis stage, SonarQube needs some time to process the results while Jenkins already proceeds to the Quality Gate stage. It could therefore be useful to add a wait step.

Example pipeline:

```
pipeline {
  agent none
  stages {
    stage("BMA SonarQube analysis") {
      agent any
      steps {
        withSonarQubeEnv('My SonarQube Server') {
          java -Dspring.config.location=file:C:\bma\application.properties -jar C:\bma\bma-sonar-scanner.jar
        }
      }
    }
    stage("Quality Gate") {
      steps {
        timeout(time: 1, unit: 'HOURS') {
          waitForQualityGate abortPipeline: true
        }
      }
    }
  }
}
```