

# Try Out SonarQube

You've heard about how [SonarQube](#) can help you write cleaner and safer code, and now you're ready to try it out for yourself. This guide shows you how to install a local instance of SonarQube and analyze a project. Installing a local instance gets you up and running quickly, so you can experience SonarQube first hand.

Once you're ready to set up a production instance, take a look at the [Install SonarQube](#) documentation.

## Installing a local instance of SonarQube


You can evaluate SonarQube using a traditional installation with the [zip file](#) or you can spin up a Docker container using one of our [Docker images](#). Click the method you prefer below to expand the installation instructions:

### [From the zip file](#)

1. [Download](#) the SonarQube Community Edition zip file.
2. As a **non-root user**, unzip it, let's say in `C:\sonarqube` or `/opt/sonarqube`.
3. As a **non-root user**, start the SonarQube Server:

```
# On Windows, execute:  
C:\sonarqube\bin\windows-x86-64\StartSonar.bat
```

```
# On other operating systems, as a non-root user execute:  
/opt/sonarqube/bin/[OS]/sonar.sh console
```

 If your instance fails to start, check your [logs](#) to find the cause.

### [From the Docker image](#)

Find the Community Edition Docker image on [Docker Hub](#).

1. Start the server by running:

```
$ docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
```

Once your instance is up and running, Log in to <http://localhost:9000> using System Administrator credentials:

- login: admin
- password: admin

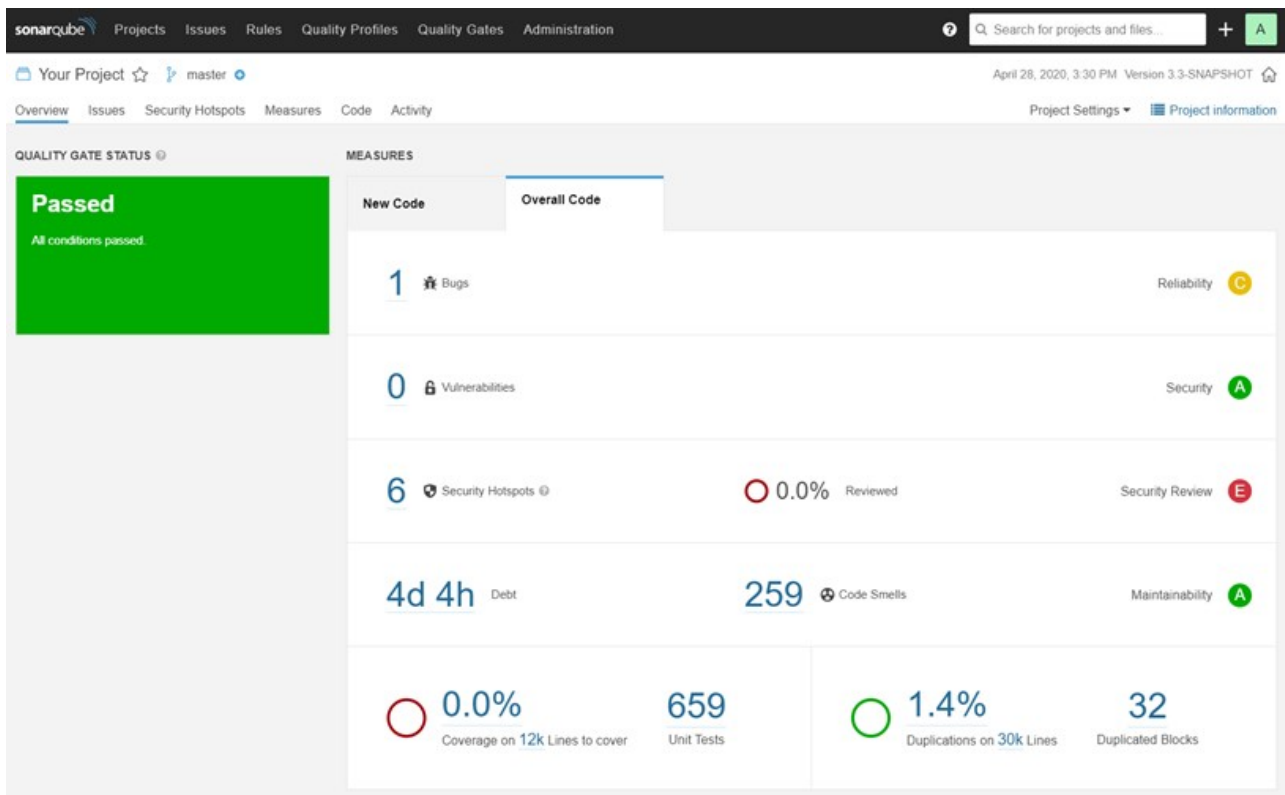
## Analyzing a Project

Now that you're logged in to your local SonarQube instance, let's analyze a project:

1. Click the **Create new project** button.
2. Give your project a **Project key** and a **Display name** and click the **Set Up** button.
3. Under **Provide a token**, select **Generate a token**. Give your token a name, click the **Generate** button, and click **Continue**.

4. Select your project's main language under **Run analysis on your project**, and follow the instructions to analyze your project. Here you'll download and execute a Scanner on your code (if you're using Maven or Gradle, the Scanner is automatically downloaded).

After successfully analyzing your code, you'll see your first analysis on SonarQube:



## Install a production instance

To install a production instance, read the [Requirements](#), and then follow the [Installation Guide](#).

## After the installation

After your server is up and running, you'll need to install one or more [SonarScanners](#) on the machines where analysis will be performed.

# Prerequisites and Overview

## Prerequisite

The only prerequisite for running SonarQube is to have Java (Oracle JRE 11 or OpenJDK 11) installed on your machine.

## Hardware Requirements

1. A small-scale (individual or small team) instance of the SonarQube server requires at least 2GB of RAM to run efficiently and 1GB of free RAM for the OS. If you are installing an instance for a large teams or Enterprise, please consider the additional recommendations below.
2. The amount of disk space you need will depend on how much code you analyze with SonarQube.
3. SonarQube must be installed on hard drives that have excellent read & write performance. Most importantly, the "data" folder houses the Elasticsearch indices on which a huge amount of I/O will be done when the server is up and running. Great read & write hard drive performance will therefore have a great impact on the overall SonarQube server performance.
4. SonarQube does not support 32-bit systems on the server side. SonarQube does, however, support 32-bit systems on the scanner side.

## Enterprise Hardware Recommendations

For large teams or Enterprise-scale installations of SonarQube, additional hardware is required. At the Enterprise level, [monitoring your SonarQube instance](#) is essential and should guide further hardware upgrades as your instance grows. A starting configuration should include at least:

- 8 cores, to allow the main SonarQube platform to run with multiple Compute Engine workers
- 16GB of RAM For additional requirements and recommendations relating to database and Elasticsearch, see [Hardware Recommendations](#).

## Supported Platforms

### Java







SonarQube scanners require version 8 or 11 of the JVM and the SonarQube server requires version 11. Versions beyond Java 11 are not officially supported.

SonarQube is able to analyze any kind of Java source files regardless of the version of Java they comply to.

We recommend using the Critical Patch Update (CPU) releases.



Java	Server	Scanners
Oracle JRE	✔ 11	✔ 11


## Java Server Scanners

	 8	 8
OpenJDK	 11	 11
	 8	 8


## Database


### [PostgreSQL](#)


-  12
-  11
-  10
-  9.3–9.6


 Must be configured to use UTF-8 charset


### [Microsoft SQL Server](#)


 2017 (MSSQL Server 14.0) with bundled Microsoft JDBC driver. Express Edition is supported.

 2016 (MSSQL Server 13.0) with bundled Microsoft JDBC driver. Express Edition is supported.




 2014 (MSSQL Server 12.0) with bundled Microsoft JDBC driver. Express Edition is supported.

 Collation must be case-sensitive (CS) and accent-sensitive (AS) (example: `Latin1_General_CS_AS`)


 `READ_COMMITTED_SNAPSHOT` must be set on the SonarQube database to avoid potential deadlocks under heavy load

 Both Windows authentication (“Integrated Security”) and SQL Server authentication are supported. See the Microsoft SQL Server section in Installing/installation/installing-the-server page for instructions on configuring authentication.

### [Oracle](#)

-  19C
-  18C
-  12C
-  11G
-  XE Editions

 Must be configured to use a UTF8-family charset (see `NLS_CHARACTERSET`)

 The driver `ojdbc14.jar` is not supported

 We recommend using the latest Oracle JDBC driver

 Only the thin mode is supported, not OCI


## Web Browser

To get the full experience SonarQube has to offer, you must enable JavaScript in your browser.

### Browser

Microsoft Internet Explorer  IE 11

Microsoft Edge  Latest

Mozilla Firefox  Latest

Google Chrome  Latest

Opera  Not tested

## Browser

Safari

✔ Latest

# ALM Integrations

## Azure Devops Server

The [SonarScanner for Azure Devops](#) is compatible with TFS 2017 Update 2 and greater

## Bitbucket Server

To add Pull Request analysis to Code Insights in Bitbucket Server, you must be running Bitbucket Server version 5.15+.

## GitHub Enterprise and GitHub.com

To add Pull Request analysis to Checks in GitHub Enterprise, you must be running GitHub Enterprise version 2.15+.

GitHub.com is also supported.

## GitLab Self-Managed and GitLab.com

To add Merge Request Decoration to your Merge Requests in GitLab Self-Managed, you must be running Gitlab Self-Manged 11.7+.

GitLab.com is also supported.

# Platform notes

## Linux

If you're running on Linux, you must ensure that:

- `vm.max_map_count` is greater than or equal to 524288
- `fs.file-max` is greater than or equal to 131072
- the user running SonarQube can open at least 131072 file descriptors
- the user running SonarQube can open at least 8192 threads

You can see the values with the following commands:

```
sysctl vm.max_map_count
sysctl fs.file-max
ulimit -n
ulimit -u
```

You can set them dynamically for the current session by running the following commands as `root`:

```
sysctl -w vm.max_map_count=524288
sysctl -w fs.file-max=131072
ulimit -n 131072
ulimit -u 8192
```

To set these values more permanently, you must update either `/etc/sysctl.d/99-sonarqube.conf` (or `/etc/sysctl.conf` as you wish) to reflect these values.

If the user running SonarQube (`sonarqube` in this example) does not have the permission to have at least 131072 open descriptors, you must insert this line in `/etc/security/limits.d/99-sonarqube.conf` (or `/etc/security/limits.conf` as you wish):

```
sonarqube    -   nofile    131072
sonarqube    -   nproc     8192
```

If you are using `systemd` to start SonarQube, you must specify those limits inside your unit file in the section `[service]` :

```
[Service]
...
LimitNOFILE=131072
LimitNPROC=8192
...
```

## seccomp filter

By default, Elasticsearch uses [seccomp filter](#). On most distribution this feature is activated in the kernel, however on distributions like Red Hat Linux 6 this feature is deactivated. If you are using a distribution without this feature and you cannot upgrade to a newer version with seccomp activated, you have to explicitly deactivate this security layer by updating

`sonar.search.javaAdditionalOpts` in `$SONARQUBEHOME/conf/sonar.properties` :

```
sonar.search.javaAdditionalOpts=-Dbootstrap.system_call_filter=false
```

You can check if seccomp is available on your kernel with:

```
$ grep SECCOMP /boot/config-$(uname -r)
```

If your kernel has seccomp, you will see:

```
CONFIG_HAVE_ARCH_SECCOMP_FILTER=y
CONFIG_SECCOMP_FILTER=y
CONFIG_SECCOMP=y
```

For more detail, see the [Elasticsearch documentation](#).

## Fonts

Generating [Executive Reports](#) requires that fonts be installed on the server hosting SonarQube. On Windows servers, this is a given. However, this is not always the case for Linux servers.

The following should be ensured:

- [Fontconfig](#) is installed on the server hosting SonarQube
- A package of [FreeType](#) fonts is installed on the SonarQube server. The exact packages available will vary by distribution, but a commonly used package is `libfreetype6`

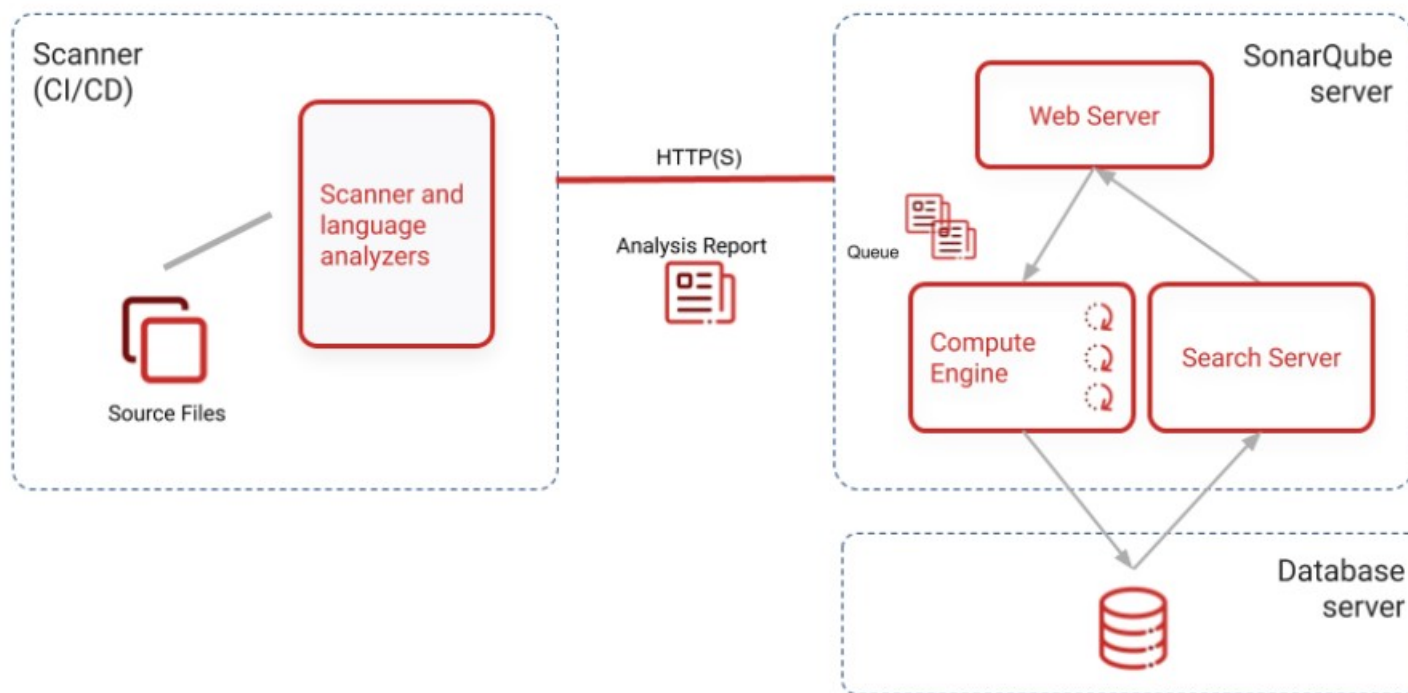
# Install the Server

## Overview

This section describes a single-node SonarQube instance. For details on clustered setup, see [Install the Server as a Cluster](#).

## Instance components

A SonarQube instance comprises three components:



1. The SonarQube server running the following processes:
  - a web server that serves the SonarQube user interface.
  - a search server based on Elasticsearch.
  - the compute engine in charge of processing code analysis reports and saving them in the SonarQube database.
2. The database to store the following:
  - Metrics and issues for code quality and security generated during code scans.
  - The SonarQube instance configuration.
3. One or more scanners running on your build or continuous integration servers to analyze projects.

## Hosts and locations

For optimal performance, the SonarQube server and database should be installed on separate hosts, and the server host should be dedicated. The server and database hosts should be located in the same network.

All hosts must be time-synchronized.

## Installing the database

Several [database engines](#) are supported. Be sure to follow the requirements listed for your database. They are real requirements not recommendations.

Create an empty schema and a `sonarqube` user. Grant this `sonarqube` user permissions to create, update, and delete objects for this schema.

[Microsoft SQL Server](#)

[Oracle](#)

[PostgreSQL](#)

## Installing SonarQube from the ZIP file

First, check the [requirements](#). Then download and unzip the [distribution](#) (do not unzip into a directory starting with a digit).

SonarQube cannot be run as `root` on Unix-based systems, so create a dedicated user account for SonarQube if necessary.

`$SONARQUBE-HOME` (below) refers to the path to the directory where the SonarQube distribution has been unzipped.

## Setting the Access to the Database

Edit `$SONARQUBE-HOME/conf/sonar.properties` to configure the database settings. Templates are available for every supported database. Just uncomment and configure the template you need and comment out the lines dedicated to H2:

```
Example for PostgreSQL
sonar.jdbc.username=sonarqube
sonar.jdbc.password=mypassword
sonar.jdbc.url=jdbc:postgresql://localhost/sonarqube
```

## Adding the JDBC Driver

Drivers for the supported databases (except Oracle) are already provided. Do not replace the provided drivers; they are the only ones supported.

For Oracle, copy the JDBC driver into `$SONARQUBE-HOME/extensions/jdbc-driver/oracle`.

## Configuring the Elasticsearch storage path

By default, Elasticsearch data is stored in `$SONARQUBE-HOME/data`, but this is not recommended for production instances. Instead, you should store this data elsewhere, ideally in a dedicated volume with fast I/O. Beyond maintaining acceptable performance, doing so will also ease the upgrade of SonarQube.

Edit `$SONARQUBE-HOME/conf/sonar.properties` to configure the following settings:

```
sonar.path.data=/var/sonarqube/data
sonar.path.temp=/var/sonarqube/temp
```



The user used to launch SonarQube must have read and write access to those directories.

## Starting the Web Server

The default port is "9000" and the context path is "/". These values can be changed in `$SONARQUBE-HOME/conf/sonar.properties`:

```
sonar.web.host=192.0.0.1
sonar.web.port=80
sonar.web.context=/sonarqube
```

Execute the following script to start the server:

- On Linux: `bin/linux-x86-64/sonar.sh start`
- On macOS: `bin/macosx-universal-64/sonar.sh start`
- On Windows: `bin/windows-x86-64/StartSonar.bat`

You can now browse SonarQube at <http://localhost:9000> (the default System administrator credentials are `admin/admin`).

## Adjusting the Java Installation

If there are multiple versions of Java installed on your server, you may need to explicitly define which version of Java is used.

To change the Java JVM used by SonarQube, edit `$SONARQUBE-HOME/conf/wrapper.conf` and update the following line:

```
wrapper.java.command=/path/to/my/jdk/bin/java
```

## Advanced Installation Features

- Running SonarQube as a Service on [Windows](#) or [Linux](#)
- Running SonarQube [behind a Proxy](#)
- Monitoring and adjusting [Java Process Memory](#)

## Installing SonarQube from the Docker Image

See your SonarQube version below for instructions on installing the server from a Docker image.

### SonarQube 8.2+

Follow these steps for your first installation:

1. Creating the following volumes helps prevent the loss of information when updating to a new version or upgrading to a higher edition:
  - `sonarqube_data` – contains data files, such as the embedded H2 database and Elasticsearch indexes
  - `sonarqube_logs` – contains SonarQube logs about access, web process, CE process, and Elasticsearch
  - `sonarqube_extensions` – will contain any plugins you install and the Oracle JDBC driver if necessary.

Create the volumes with the following commands:

```
$> docker volume create --name sonarcube_data
$> docker volume create --name sonarcube_logs
$> docker volume create --name sonarcube_extensions
```

Make sure you're using [volumes](#) as shown with the above commands, and not [bind mounts](#). Using bind mounts prevents plugins from populating correctly.

2. Drivers for supported databases (except Oracle) are already provided. If you're using an Oracle database, you need to add the JDBC driver to the `sonar_extensions` volume. To do this:

- a. Start the SonarQube container with the embedded H2 database:

```
$ docker run --rm \
  -p 9000:9000 \
  -v sonarcube_extensions:/opt/sonarcube/extensions \
  <image_name>
```

- b. Exit once SonarQube has started properly.

- c. Copy the Oracle JDBC driver into `sonarcube_extensions/jdbc-driver/oracle`.

3. Run the image with your database properties defined using the `-e` environment variable flag:

```
$> docker run -d --name sonarcube \
  -p 9000:9000 \
  -e SONAR_JDBC_URL=... \
  -e SONAR_JDBC_USERNAME=... \
  -e SONAR_JDBC_PASSWORD=... \
  -v sonarcube_data:/opt/sonarcube/data \
  -v sonarcube_extensions:/opt/sonarcube/extensions \
  -v sonarcube_logs:/opt/sonarcube/logs \
  <image_name>
```

For more configuration environment variables, see the [Docker Environment Variables](#).

Use of the environment variables `SONARQUBE_JDBC_USERNAME`, `SONARQUBE_JDBC_PASSWORD`, and `SONARQUBE_JDBC_URL` is deprecated and will stop working in future releases.

### Example Docker Compose configuration

If you're using [Docker Compose](#), use the following example as a reference when configuring your `.yaml` file. Click the heading below to expand the `.yaml` file.

[Docker Compose .yaml file example](#)

## SonarQube 7.9.x LTS

Follow these steps for your first installation:

1. Create volumes `sonarcube_conf`, `sonarcube_data`, `sonarcube_logs`, and `sonarcube_extensions` and start the image with the following command. This will populate all the volumes (copying default plugins, create the Elasticsearch data folder, create

the sonar.properties configuration file). Watch the logs, and, once the container is properly started, you can force-exit (ctrl+c) and proceed to the next step.

```
$ docker run --rm \
  -p 9000:9000 \
  -v sonarqube_conf:/opt/sonarqube/conf \
  -v sonarqube_extensions:/opt/sonarqube/extensions \
  -v sonarqube_logs:/opt/sonarqube/logs \
  -v sonarqube_data:/opt/sonarqube/data \
  <image_name>
```

2. Configure sonar.properties if needed. Please note that due to [SONAR-12501](#), providing sonar.jdbc.url, sonar.jdbc.username, sonar.jdbc.password and sonar.web.javaAdditionalOpts in sonar.properties is not working, and you will need to explicitly define these values in the docker run command with the -e flag.

```
#Example for PostgreSQL
-e sonar.jdbc.url=jdbc:postgresql://localhost/sonarqube
```

Drivers for supported databases (except Oracle) are already provided. Do not replace the provided drivers; they are the only ones supported. For Oracle, you need to copy the JDBC driver into \$SONARQUBE\_HOME/extensions/jdbc-driver/oracle.

3. Run the image with your JDBC username and password :

```
$ docker run -d --name sonarqube \
  -p 9000:9000 \
  -e sonar.jdbc.url=... \
  -e sonar.jdbc.username=... \
  -e sonar.jdbc.password=... \
  -v sonarqube_conf:/opt/sonarqube/conf \
  -v sonarqube_extensions:/opt/sonarqube/extensions \
  -v sonarqube_logs:/opt/sonarqube/logs \
  -v sonarqube_data:/opt/sonarqube/data \
  <image_name>
```

## Next Steps

Once your server is installed and running, you may also want to [Install Plugins](#). Then you're ready to begin [Analyzing Source Code](#).

## Troubleshooting/FAQ

### Failed to connect to the Marketplace via proxy

Double check that settings for proxy are correctly set in \$SONARQUBE\_HOME/conf/sonar.properties. Note that if your proxy username contains a backslash, then it should be escaped - for example username "domain\user" in file should look like:

```
http.proxyUser=domain\\user
```

For some proxies, the exception "java.net.ProtocolException: Server redirected too many times" might mean an incorrect username or password has been configured.

## **Exception `java.lang.RuntimeException`: can not run elasticsearch as root**

SonarQube starts an Elasticsearch process, and the same account that is running SonarQube itself will be used for the Elasticsearch process. Since Elasticsearch cannot be run as `root`, that means SonarQube can't be either. You must choose some other, non-`root` account with which to run SonarQube, preferably an account dedicated to the purpose.

## Overview

Once the SonarQube platform has been installed, you're ready to install a scanner and begin creating projects. To do that, you must install and configure the scanner that is most appropriate for your needs. Do you build with:

- Gradle - [SonarScanner for Gradle](#)
- .NET - [SonarScanner for .NET](#)
- Maven - use the [SonarScanner for Maven](#)
- Jenkins - [SonarScanner for Jenkins](#)
- Azure DevOps - [SonarQube Extension for Azure DevOps](#)
- Ant - [SonarScanner for Ant](#)
- anything else (CLI) - [SonarScanner](#)

SonarQube integrations are supported for popular ALMs: GitHub Enterprise and GitHub.com, BitBucket Server, and Azure Devops Server.

We do not recommend running an antivirus scanner on the machine where a SonarQube analysis runs, it could result in unpredictable behavior.

A project is created in SonarQube automatically on its first analysis. However, if you need to set some configuration on your project before its first analysis, you have the option of provisioning it via Administration options or the + menu item, which is visible to users with project creation rights.

## What does analysis produce?

SonarQube can analyze up to 27 different languages depending on your edition. The outcome of this analysis will be quality measures and issues (instances where coding rules were broken). However, what gets analyzed will vary depending on the language:

- On all languages, "blame" data will automatically be imported from supported SCM providers. [Git and SVN are supported automatically](#). Other providers require additional plugins.
- On all languages, a static analysis of source code is performed (Java files, COBOL programs, etc.)
- A static analysis of compiled code can be performed for certain languages (.class files in Java, .dll files in C#, etc.)

## Will all files be analyzed?

By default, only files that are recognized by your edition of SonarQube are loaded into the project during analysis. For example if you're using SonarQube Community Edition, which includes analysis of Java and JavaScript, but not C++, all `.java` and `.js` files would be loaded, but `.cpp` files would be ignored.

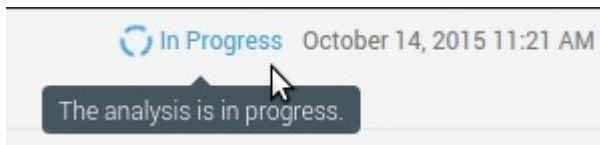
## What about branches and pull requests?

*Developer Edition* adds the ability to analyze your project's [branches](#) and [pull requests](#) as well as the ability to automatically decorate pull requests in some ALM interfaces.

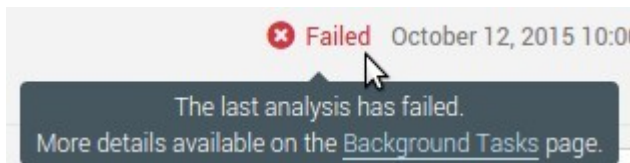
## What happens during analysis?

During analysis, data is requested from the server, the files provided to the analysis are analyzed, and the resulting data is sent back to the server at the end in the form of a report, which is then analyzed asynchronously server-side.

Analysis reports are queued, and processed sequentially, so it is quite possible that for a brief period after your analysis log shows completion, the updated values are not visible in your SonarQube project. However, you will be able to tell what's going on because an icon will be added on the project homepage to the right of the project name. Mouse over it for more detail (and links if you're logged in with the proper permissions).



The icon goes away once processing is complete, but if analysis report processing fails for some reason, the icon changes:



## FAQ

**Q.** Analysis errors out with `java.lang.OutOfMemoryError: GC overhead limit exceeded`. What do I do?

**A.** This means your project is too large or too intricate for the scanner to analyze with the default memory allocation. To fix this you'll want to allocate a larger heap (using `-Xmx[numeric value here]`) to the process running the analysis. Some CI engines may give you an input to specify the necessary values, for instance if you're using a Maven Build Step in a Jenkins job to run analysis. Otherwise, use Java Options to set a higher value. Note that details of setting Java Options are omitted here because they vary depending on the environment.

**Q.** Analysis errors out with `PKIX path building failed`. What do I do? **A.** This error tells you that your SonarQube server is configured with HTTPS and a custom SSL certificate. However, the certificate is not correctly configured in the scanner machine's JVM. This configuration is outside of SonarQube scope. The server certificate is unknown and could not be validated with the provided truststore. You need to add the SonarQube server certificate to the Java truststore. See [Oracle's documentation](#) for more information.

<https://docs.sonarqube.org/latest/analysis/overview/>