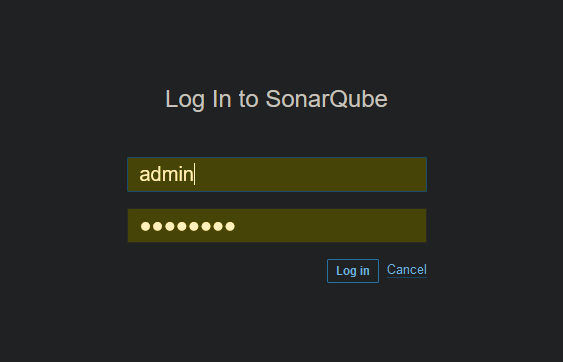
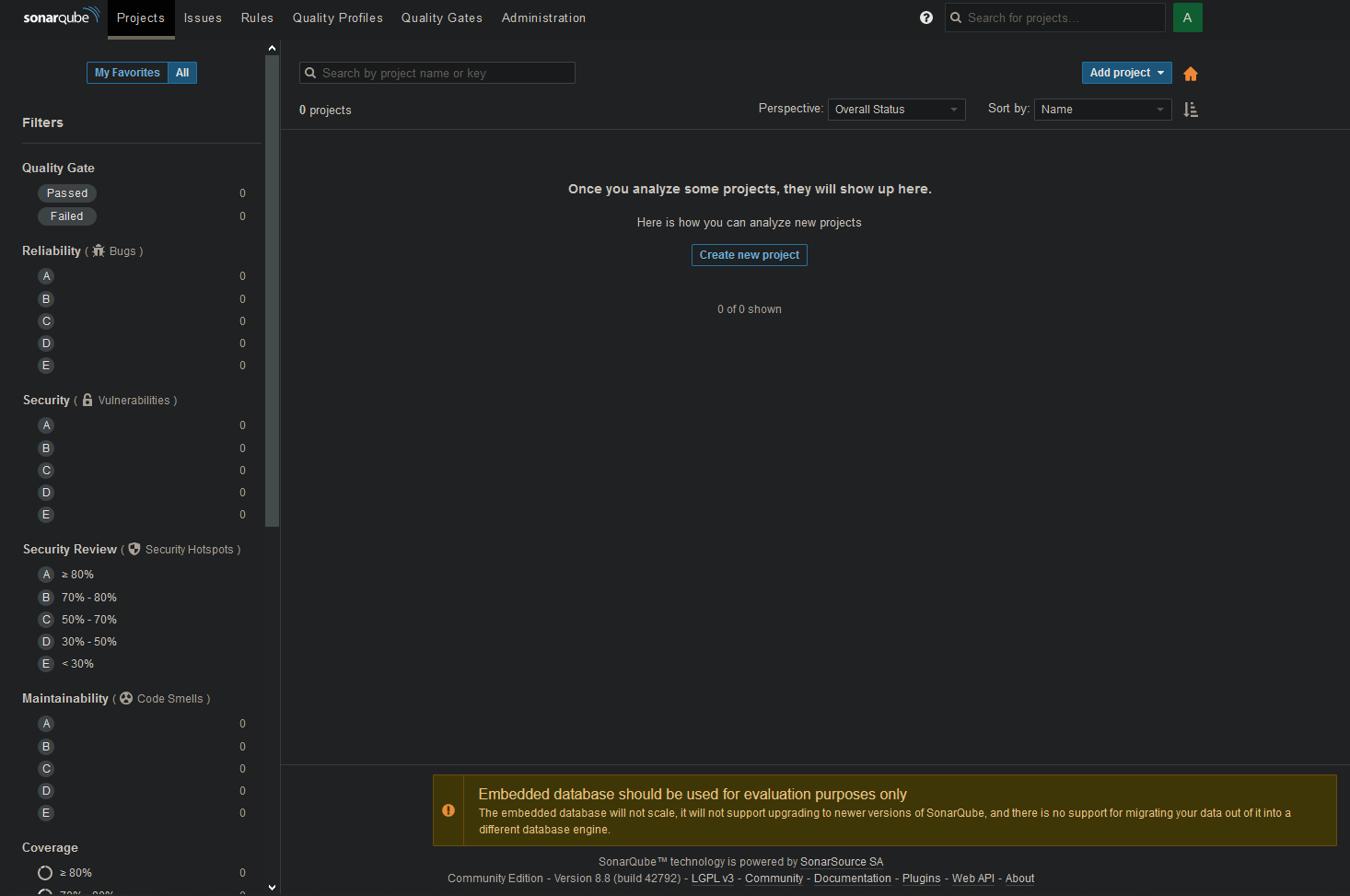
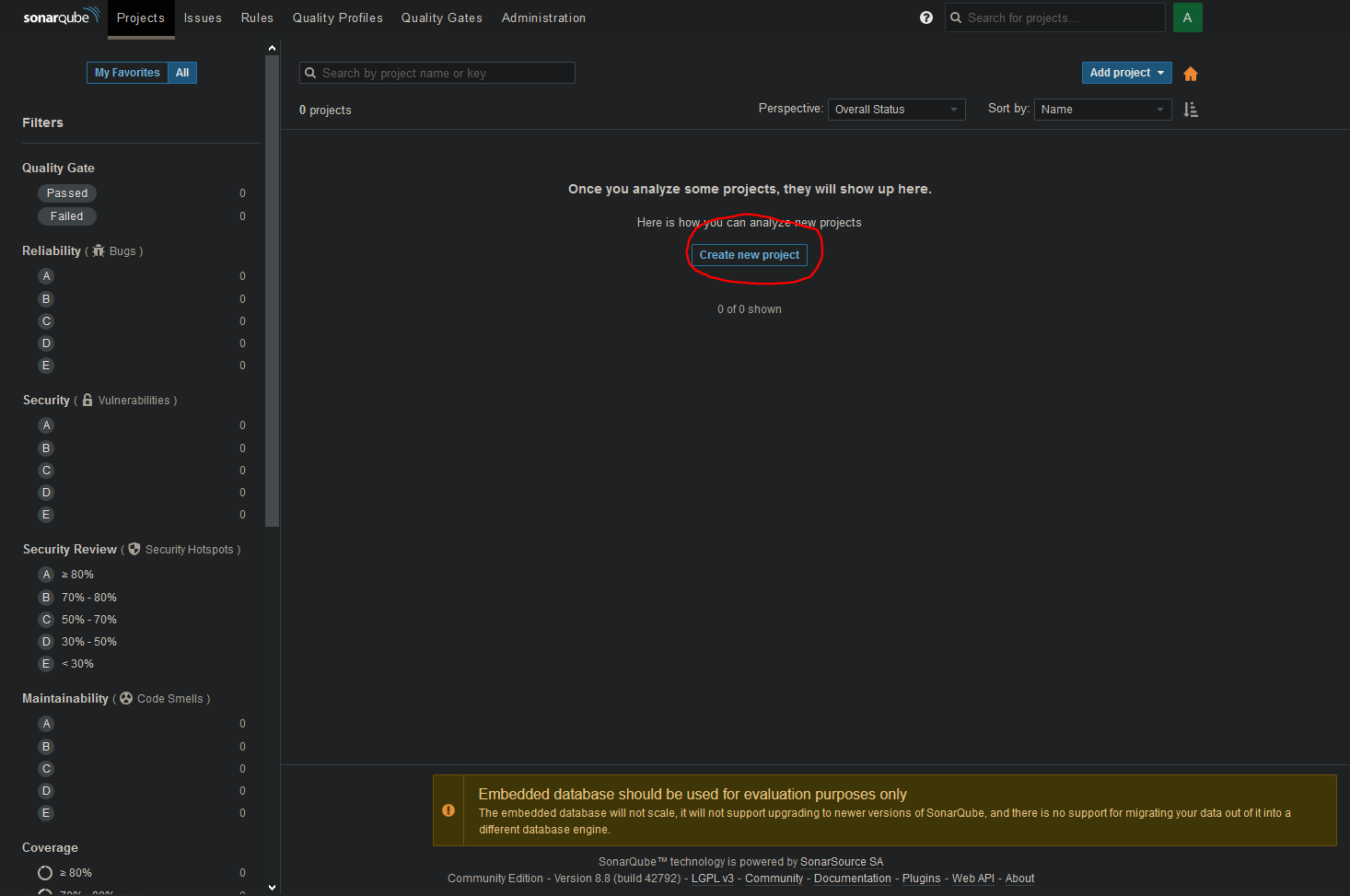
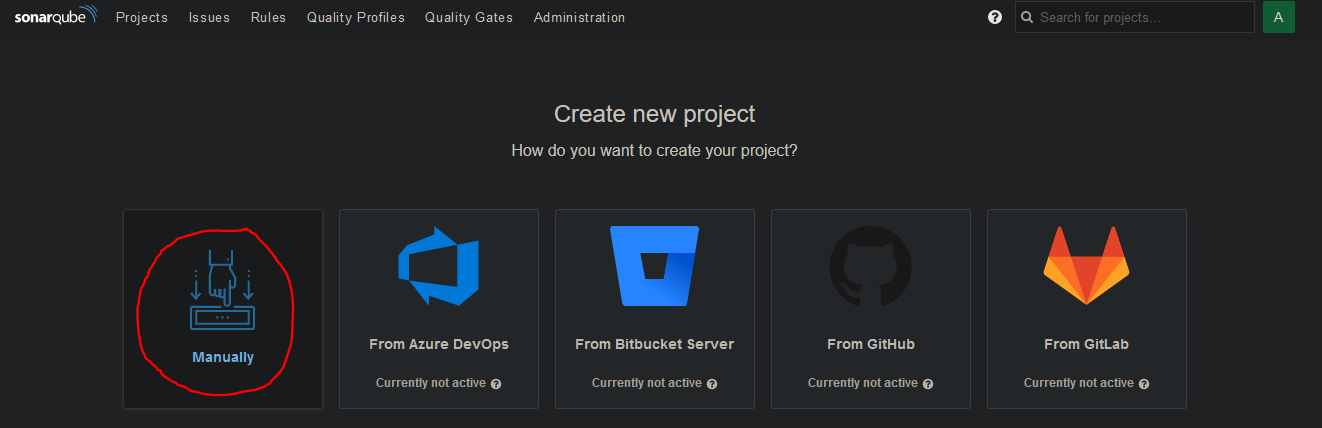
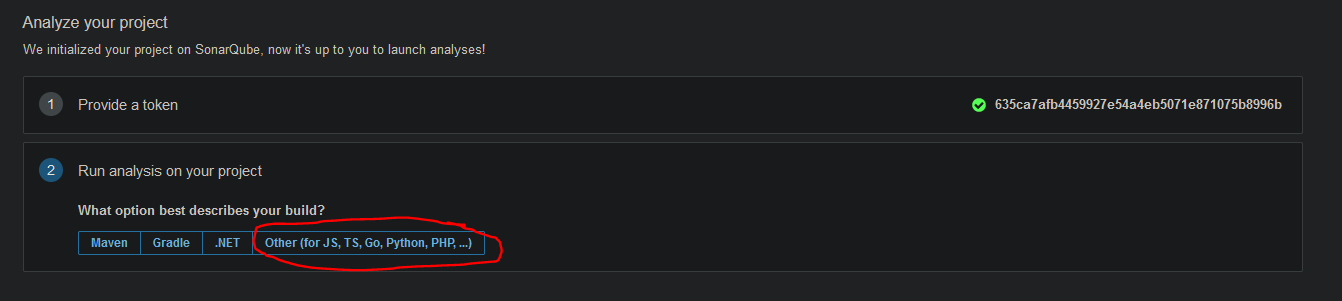
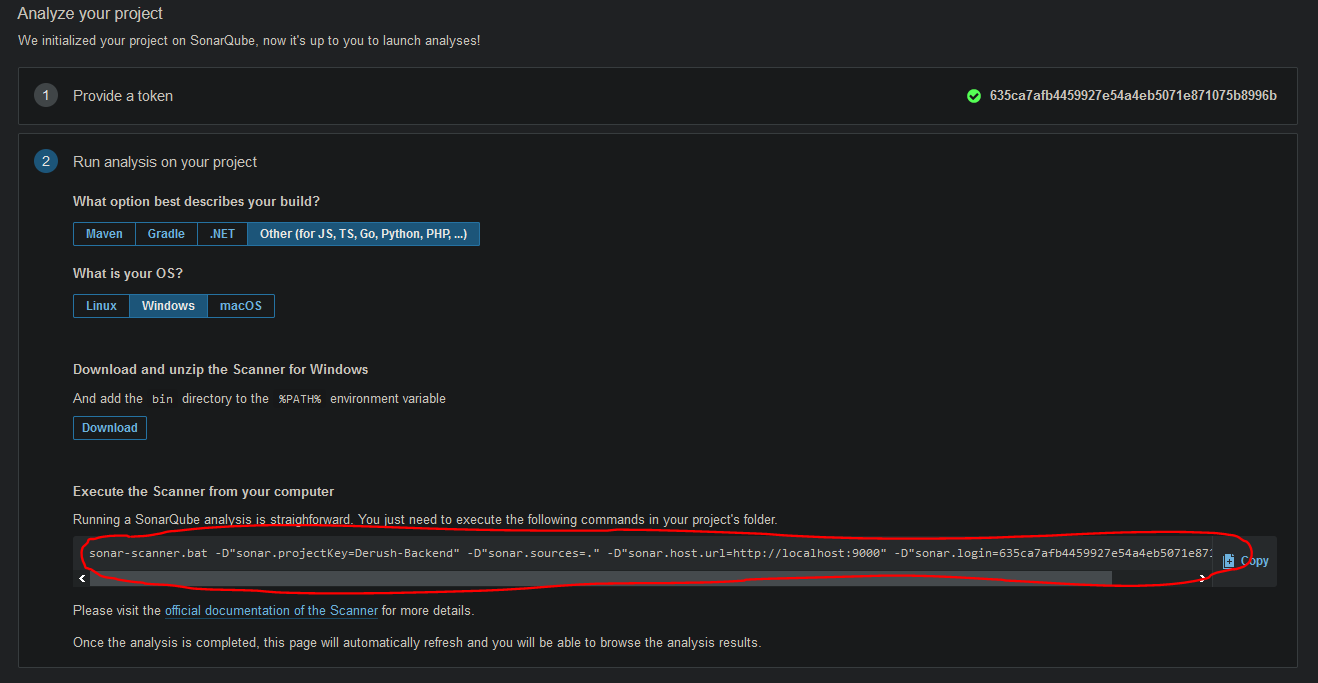
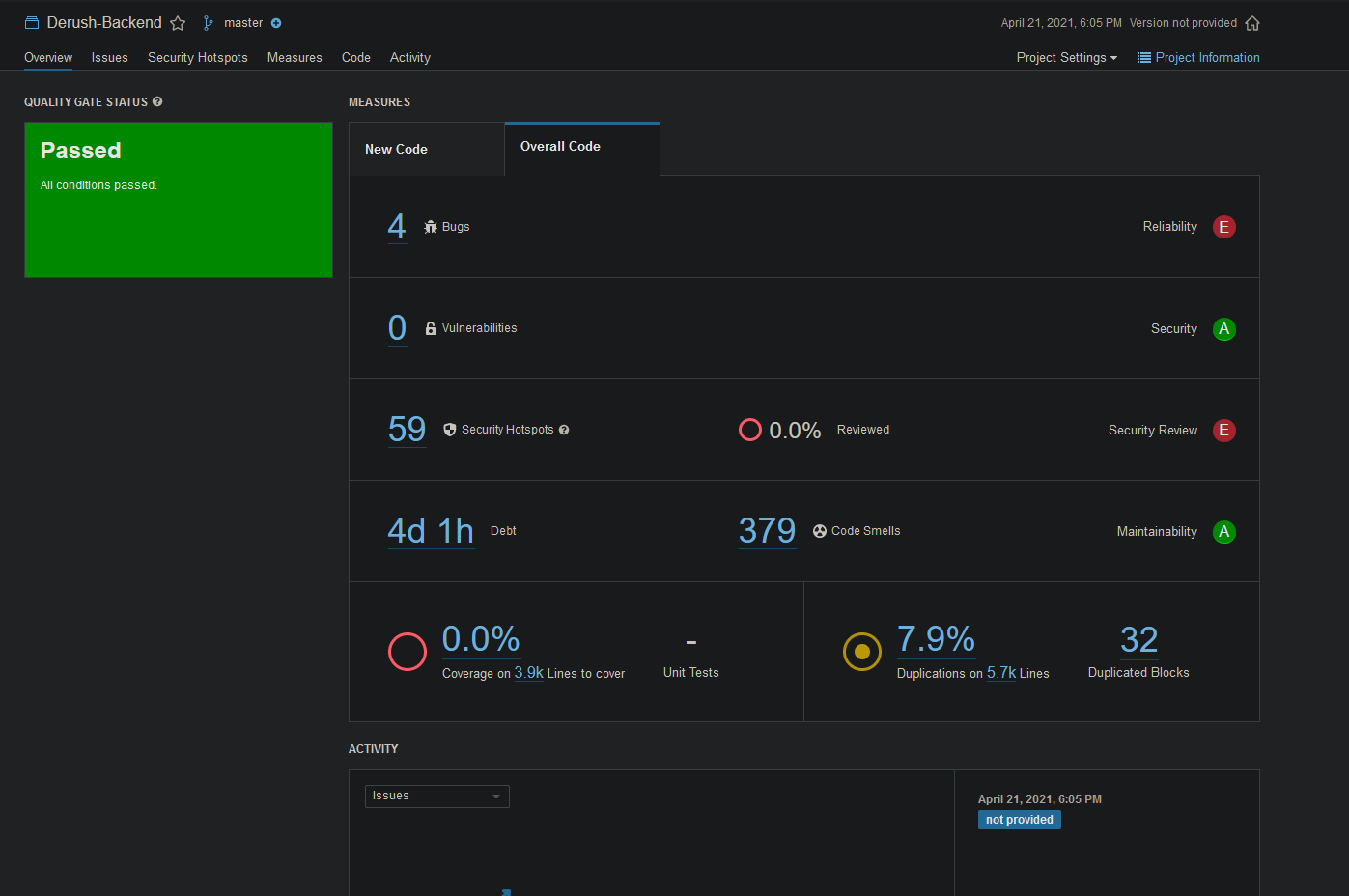
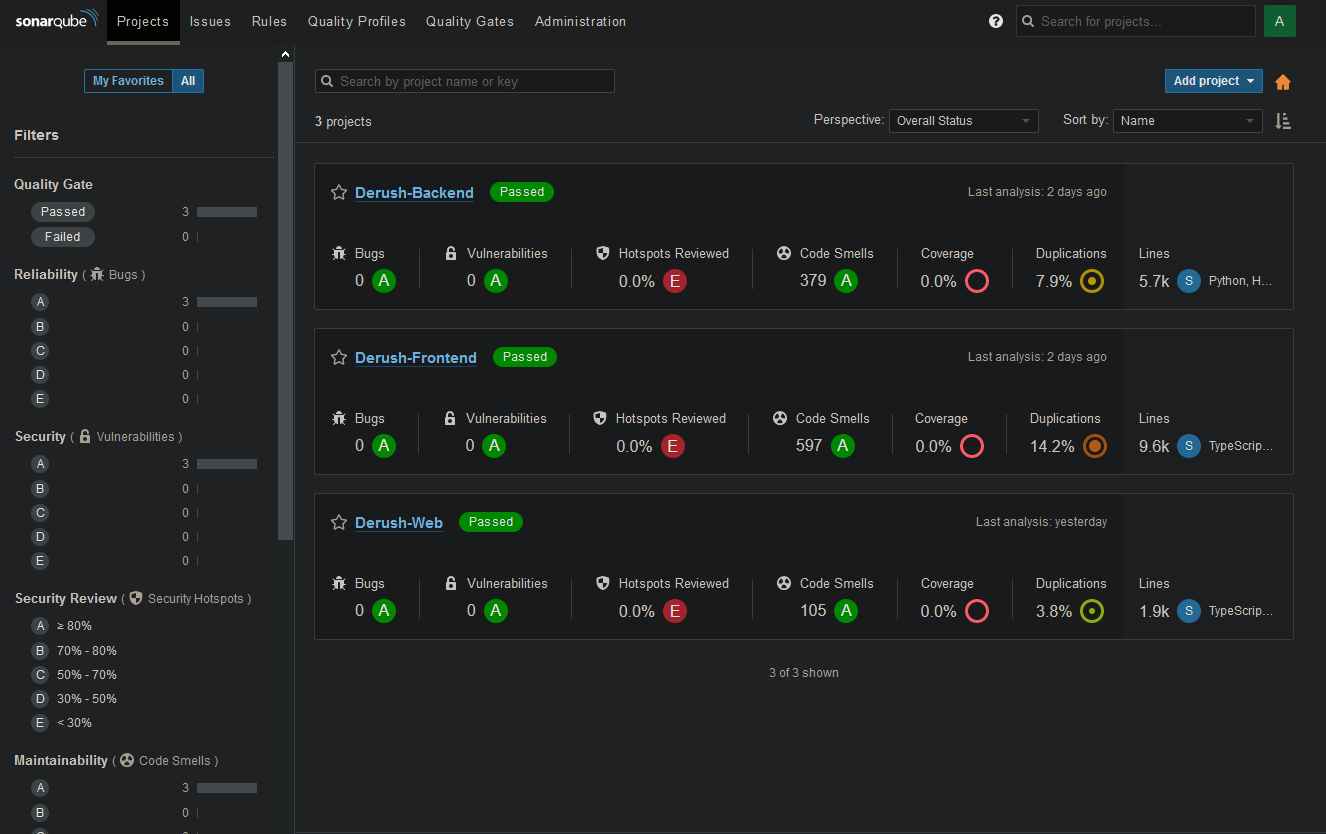
**Installation**

* Download SonarQube at this adress : <https://www.sonarqube.org/downloads/>
* Unzip it in C:\sonarqube (windows) or /opt/sonarqube (macOS)
* Start 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
* Go on <http://localhost:9000/> and you should arrive on this page :
* You can log in with these credentials : login: admin; password: admin
* You should arrive on this page:
* Now you should install sonarScanner: go download it on this page : <https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/>
* Unzip it in the sonarqube folder you created
* Add the "bin" directory to the %PATH% environment variable

**Configuration**

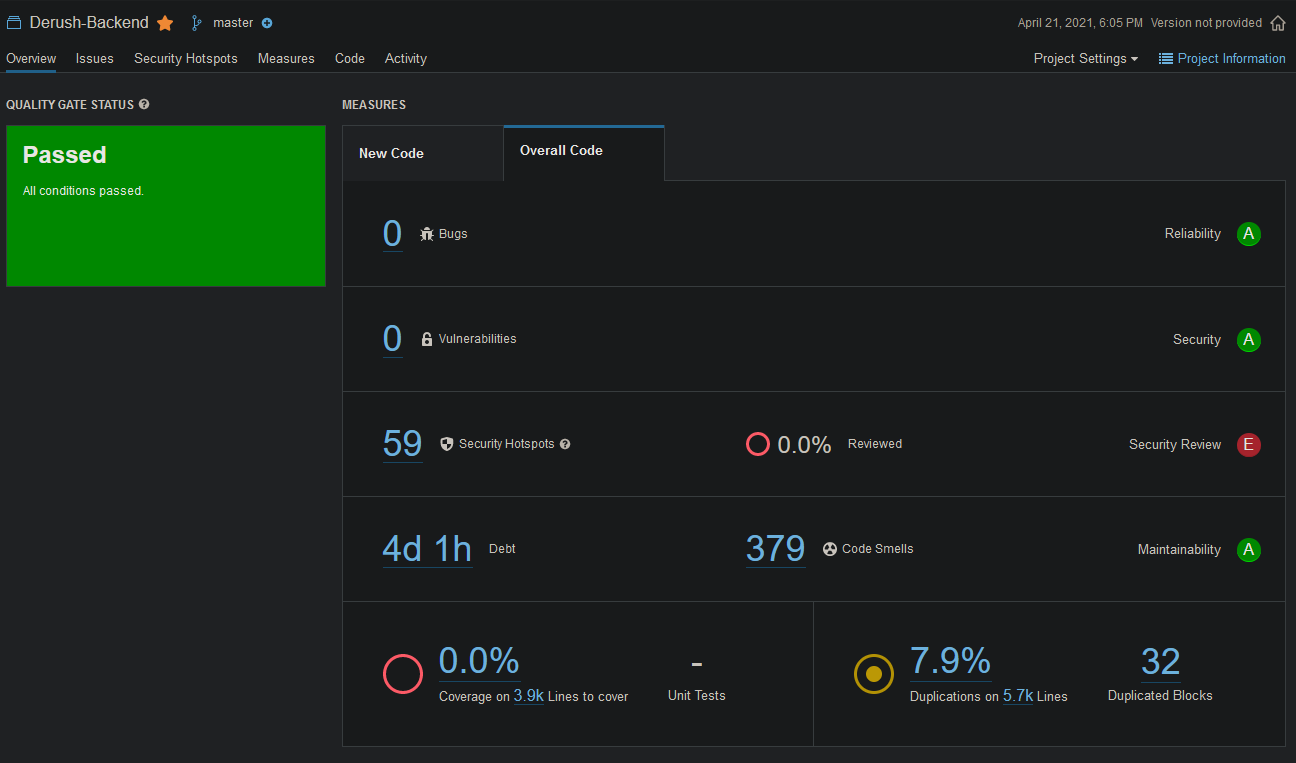
* Make sure your project have a [sonar-project.properties](http://sonar-project.properties) at the root of the application, if not create it with this inside :
* # must be unique in a given SonarQube instance
* sonar.projectKey={projectKey}
* # --- optional properties ---
* # defaults to project key
* #sonar.projectName=My project
* # defaults to 'not provided'
* #sonar.projectVersion=1.0
* # Path is relative to the sonar-project.properties file. Defaults to .
* #sonar.sources=.
* # Encoding of the source code. Default is default system encoding
* #sonar.sourceEncoding=UT
* Note: The file is saved with git but not its contents
* Go to <http://localhost:9000/> and log in
* Click on "Create new project" :
* Then "Manually":
* And the enter a project key and the display name you want. If you didn't have a [sonar-project.properties](http://sonar-project.properties), name the project key as you want and copy it on sonar.projectKey={projectKey}. Then click on "Continue"
* Generate a token with the name you want, click on "Continue" then on "Other":
* Select your OS and run the command shown below in your project's folder:
* Normaly the localhost page should reload and an overall report should appear:
* 

**How to use**

Une fois SonarQube installé, paramétré et que les projets sont importés et analysés, la page d'accueil devrait ressembler à ceci :

On peut voir nos trois projets avec plusieurs paramètres et une note.

On peut cliquer sur un des projets pour avoir un rapport un peu plus détaillé :



On peut voir que ce rapport est divisé en 7 parties dont certaines sont notés (de A à E):

* En haut à gauche (ici Passed en vert), sonarQube dit s'il pense que le code est valide pour un déploiement.
* Ensuite la ligne "Bugs": cela correspond à un problème relativement important dans le code, si cela ne casse pas l'application actuellement ça a des chances de faire casser plus tard. C'est à résoudre en priorité ! Il ne faudrait pas déployer de versions ou de merge de branche qui présentent des Bugs.
* Ensuite la ligne "Vulnerabilities": cela correspond à une faille de sécurité qui peut potentiellement être exploitée. C'est impératif de ne pas en avoir, que ce soit lors d'une mise en prod ou d'une merge de branche.
* Ensuite la ligne "Security Hotspots": cela correspond à une potentielle faille de sécurité qui demande d'être revu manuellement par un dev. Le % à côté correspond au pourcentage de "Hotspots" qui ont été revu manuellement. Il faudrait que le % soit de 100% à chaque mise en prod de nouvelle version.
* Ensuite la ligne "Code Smells": cela correspond à une partie du code qui est fonctionnelle mais qui ne respecte pas les normes de codages. La "Debt" associé correspond à une approximation du temps à passer pour remettre tout le code "à la norme". Il est important d'essayer de coder le plus proprement possible, et de faire remonter ce qui n'est pas bien coder lors de la peer review de code. Il peut être intéressant de régulièrement aller corriger quelques erreurs pour avoir un code plus propre à la fin.
* Ensuite la case "Coverage": cela correspond au % de code couvert par nos tests unitaires associé avec le nombre de TU créés. Dans l'idéal il faudrait que ce soit de 100%.
* Finalement la case "Duplications": cela correspond au % de code dupliqué dans notre projet, associé au nombre de bloc de code que cela correspond. Il faudrait que ce % soit proche (même égal) à 0% et il faudrait revoir les blocs de code dupliqués pour refactor ces endroits.

/!\ SonarQube n'est qu'un outil et n'est pas fiable à 100%, il y a certaines erreurs que le rapport transmet qui n'en sont pas ainsi que d'autres que sonarQube ne voit pas!

**Clean as you code**

Concept de codage proposé par sonarQube, ils ne conseillent pas d'utiliser leur logiciel pour corriger tous les "code smells", c'est trop coûteux en temps et n'apporte pas d'intérêt réel pour l'application.

Ce qu'ils préconnisent est de définir des normes de codage à respecter tout le temps (commenter le code et particulièrement les fonctions, nommer correctement les variables, pas de duplicat de code, TU sur toutes nouvelles fonctions) pour le nouveau code.

Il faut ensuite toujours appliqué cela lors des développement qui en suivent. De plus vu que l'on fait de la peer review de code (lors des merge request) il faut que le code soit "bien coder" pour que la merge request soit accepté. Ne pas hésiter à refuser une merge request si le code n'est pas propre.

Et finalement c'est uniquement si l'on doit modifier du code déjà existant que là on refactor ce qui a été "mal codé" pour le remettre à la norme.

=> Cela permet d'au final avoir un code propre tout en ne perdant pas de temps à recoder tout ce qui a été fait.