**BUILD DI BOOKKEEPER (MAVEN + SONARCLOUD)**

1) **GitHub**: effettuare una fork del repository BookKeeper. Questo serve a creare una copia del repository sul proprio account GitHub.

2) **Git Bash**: andare sulla directory  
“C:\Users\barba\OneDrive\Desktop\’Work in progress’\’Progetti ISW2’\”  
e inserire il seguente comando:  
“git clone <https://github.com/Fanfarillo/bookkeeper.git>”.  
Questo serve a salvare il repository in locale.

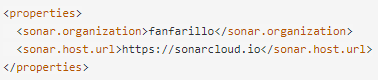
3) **Cmd**: andare sulla directory  
“C:\Users\barba\OneDrive\Desktop\Work in progress\Progetti ISW2\bookkeeper\”  
e inserire il seguente comando:  
“mvn clean package -DskipTests”.  
Questo serve a effettuare la build del progetto di BookKeeper.

4) Nel caso in cui la build dovesse fallire, correggere gli errori. Nel mio caso specifico, ho dovuto aggiungere alla variabile d’ambiente Path il valore  
“C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29910\bin\Hostx64\x64”  
e inserire una nuova variabile di sistema (JAVA\_HOME) con valore  
“C:\Program Files\Java\jdk-15.0.1\_windows-x64\_bin\jdk-15.0.1”.

5) Per evitare di far partire i test e i workflow originali, eliminarli dal proprio repository forkato.

6) **SonarCloud**: creare un nuovo progetto, selezionare “bookkeeper” e generare il SONAR\_TOKEN.

7) Inserire all’interno della cartella “.github/workflows” del repository un file .yml col seguente contenuto:  


8) Modificare il file pom.xml aggiungendovi:  


9) **GitHub**: all’interno della repository bookkeeper, andare su:  
Settings 🡪 Secrets 🡪 Actions 🡪 New repository secret  
E inserire:  
- SONAR\_TOKEN nel campo Name.  
- Il valore generato per SONAR\_TOKEN nel campo value.

**CI SU UN PROPRIO PROGETTO**

1) **Eclipse**: creare un progetto Maven prestando attenzione alle seguenti opzioni:  
 - Create a simple project.  
 - Use default workspace location.  
 - Packaging: jar.

2) Aggiungere le classi Java nel package “src\main\java” del progetto Maven.

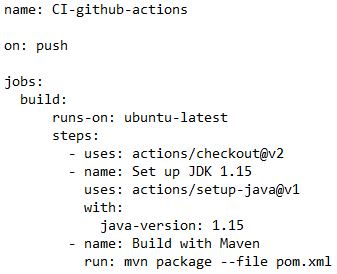
3) Aggiungere al file pom.xml la seguente dicitura, che deve essere indentata rispetto al tag <project>:  
 <properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 </properties>

4) **Cmd**: andare sulla directory  
“C:\Users\barba\eclipse-workspace2\nome-progetto\”  
e inserire il seguente comando:  
“mvn clean package -DskipTests”.  
Questo serve a effettuare la build del proprio progetto.

5) **GitHub**: creare un nuovo repository.

6) **Git Bash**: andare sulla directory  
“C:\Users\barba\eclipse-workspace2\nome-progetto\”  
e inserire i seguenti comandi:  
 - “git init”  
 - “git add .”  
 - “git config –global user.email “[matteo.fanfarillo99@gmail.com](mailto:matteo.fanfarillo99@gmail.com)””  
 - “git config –global user.name “Fanfarillo””  
 - “git remote add origin https://github.com/Fanfarillo/nome-nuovo-repository.git”  
 - “git commit -m “Commento del commit””  
 - “git push -u origin master”

7) **GitHub**: aggiungere la directory “.github/workflows” all’interno di quella già presente nel nuovo repository GithHub (che prende il nome dal progetto Maven).

8) Inserire nella directory “.github/workflows” il file ci.yml, il cui contenuto deve essere:  
  


Dopodiché effettuare il commit su GitHub.  
Questo serve a fare in modo che, a ogni operazione di push (che viene subito dopo il commit), GitHub Actions faccia partire una build del progetto in modo automatico.

Se invece si vuole utilizzare anche SonarCloud, creare un nuovo progetto su SonarCloud e selezionare il repository in questione; dopodiché, all’interno del progetto, andare su Administration 🡪 Analysis Method 🡪 GitHub Actions e seguire le istruzioni fornite (o anche i passi riportati in “Build di Bookkeeper”).

9) **Git Bash**: Rimanendo sulla directory  
“C:\Users\barba\eclipse-workspace2\nome-progetto\”  
Inserire il seguente comando:  
“git pull”.  
Questo serve a riportare in locale gli aggiornamenti appena effettuati su GitHub.

10) Ogni volta che si aggiorna il progetto, andare sulla directory  
“C:\Users\barba\eclipse-workspace2\nome-progetto\”  
e inserire i seguenti comandi:  
 - “git add .”  
 - “git commit -m “Commento del commit””  
 - “git push origin master”