Game_of_life POC

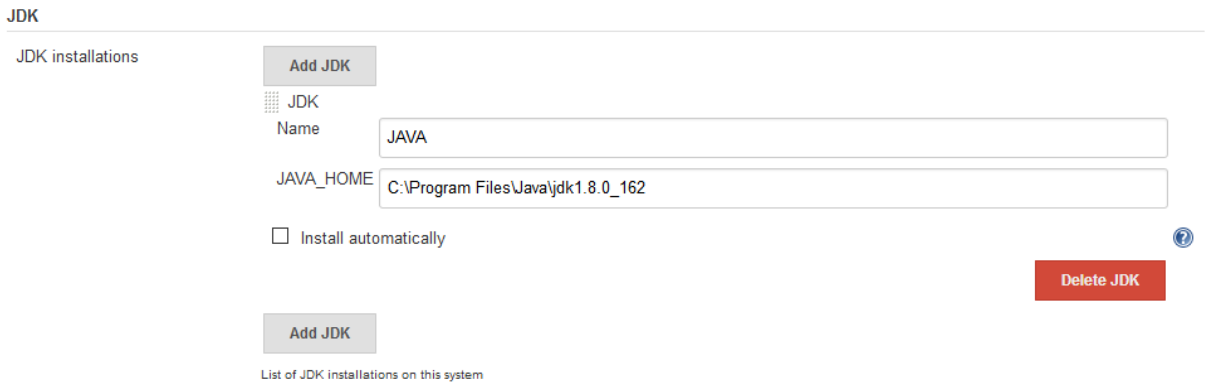
Install Jenkins with necessary packages and configure SonarQube server.

Step-1 In jenkins Install the following plugin,

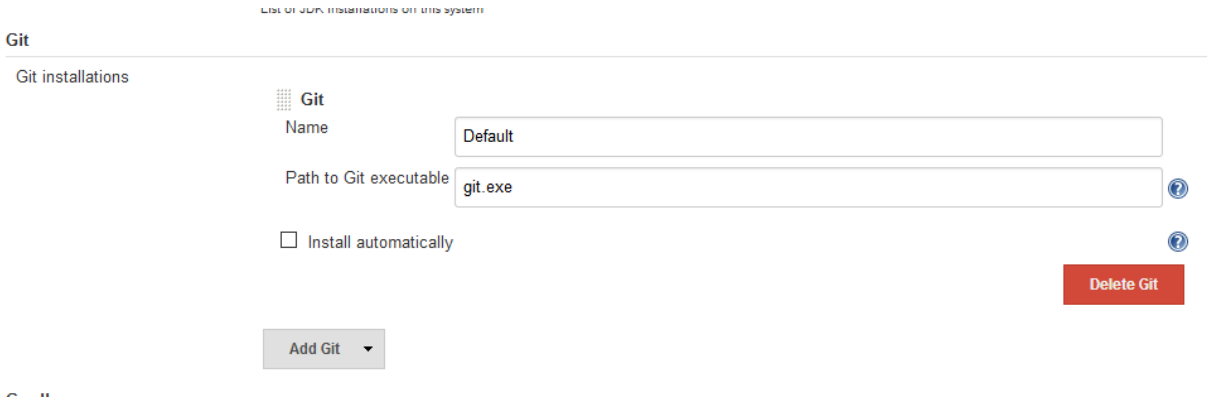
- ✓ Maven Integration
- ✓ Copy Artifact
- ✓ Deploy to container
- ✓ JUnit plugin
- ✓ Pipeline
- ✓ SonarQube Scanner for Jenkins

Step-2 Configure In Global Tool Configuration-

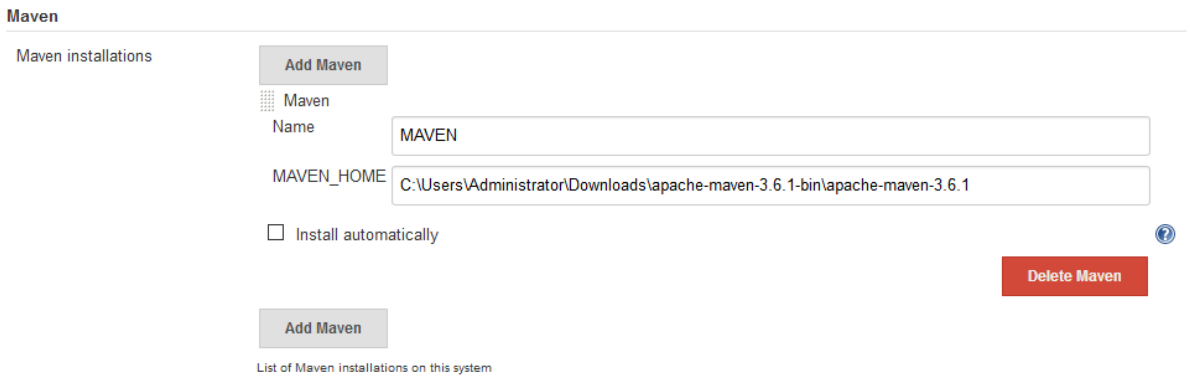
- For JAVA_JDK



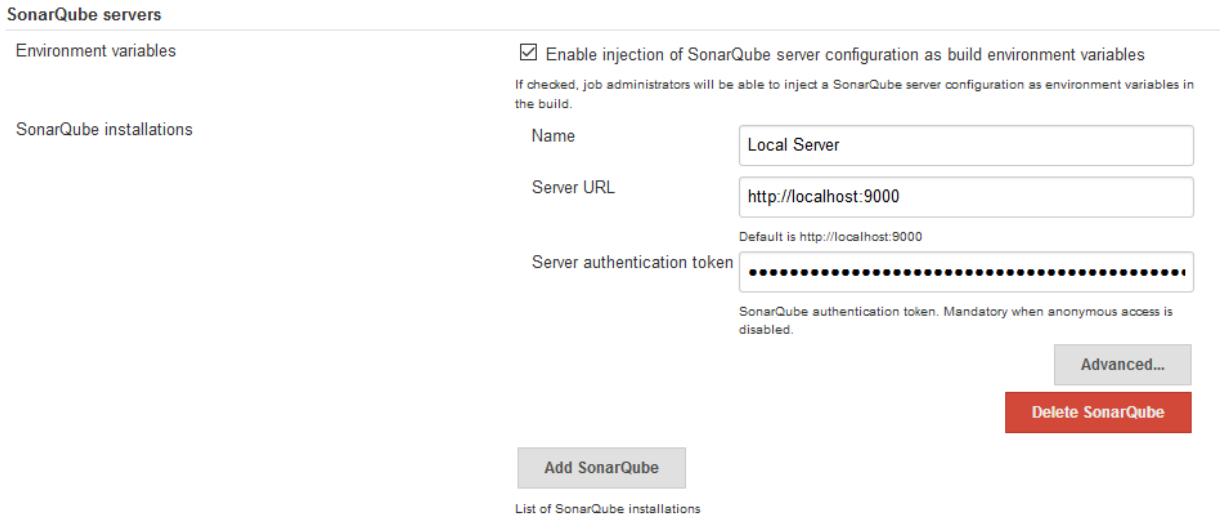
- For Git



- For Maven

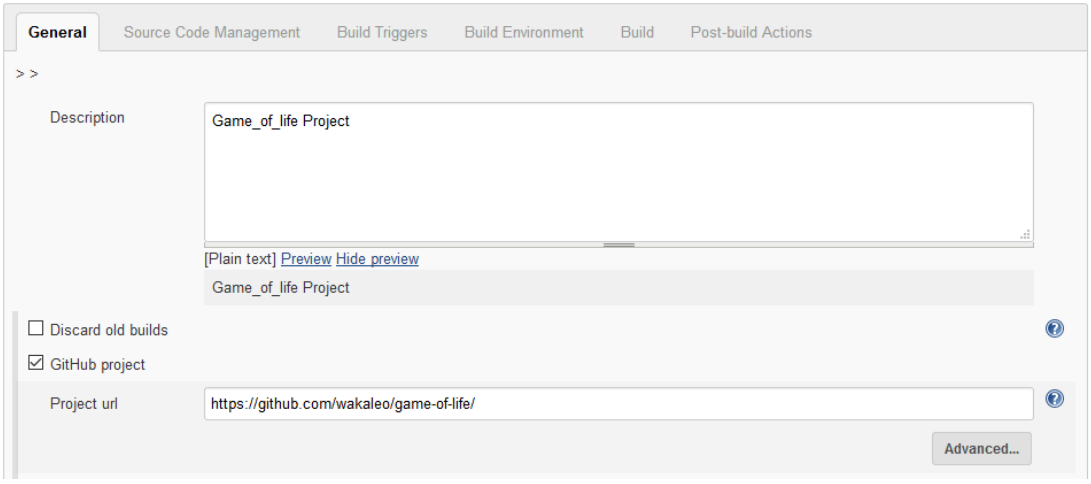


Step-4 Configure SonarQube Server in Jenkins



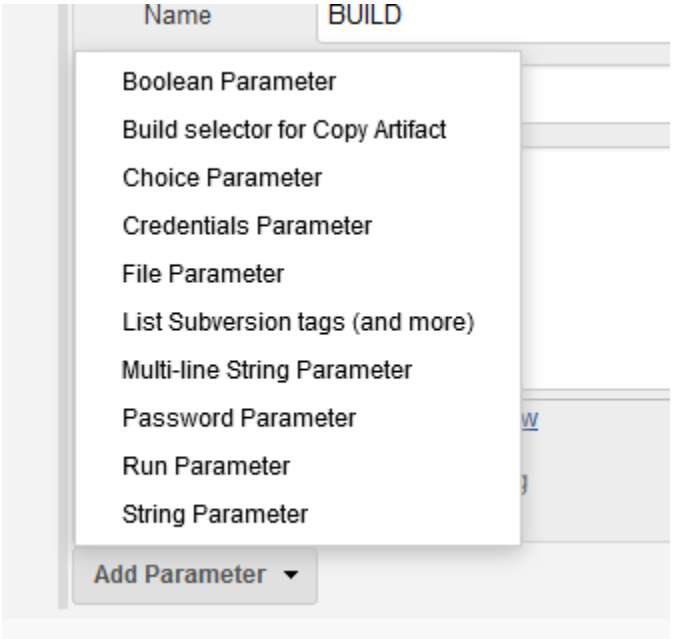
Step-5 Create new item -> Enter the name of project -> select Free Style project

Step-6 Enter the Description, select GitHub project, then enter the repo URL.

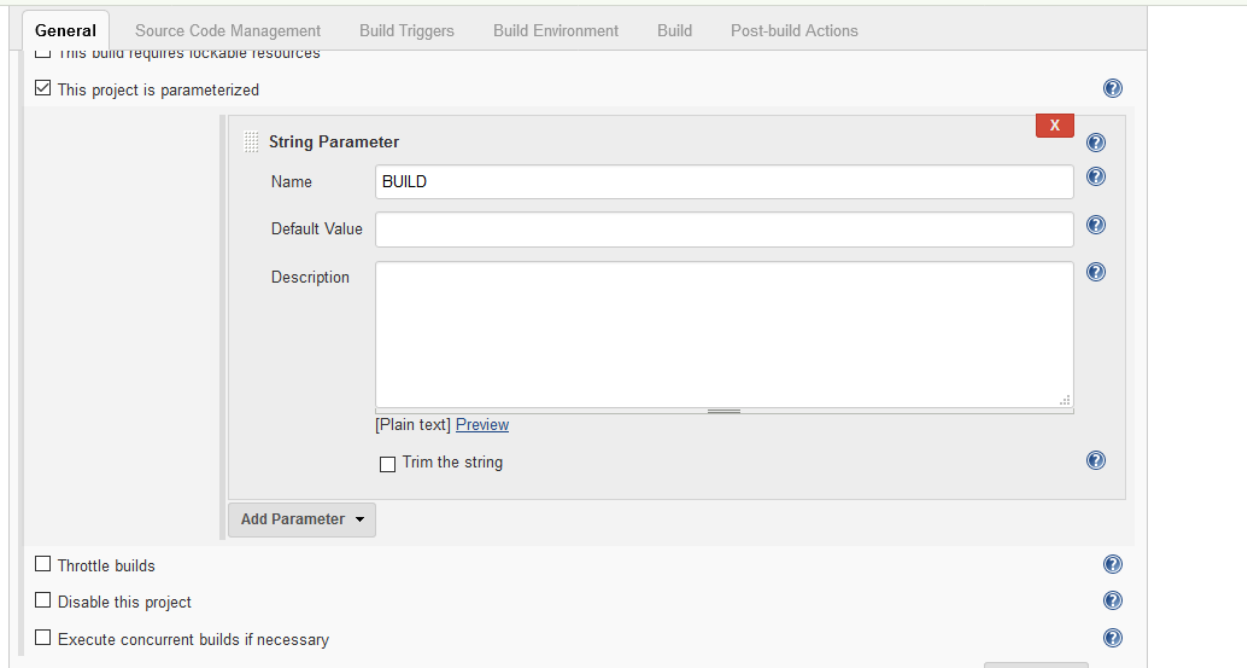


Step-7 To make the build parameterized, select this project is parameterized.

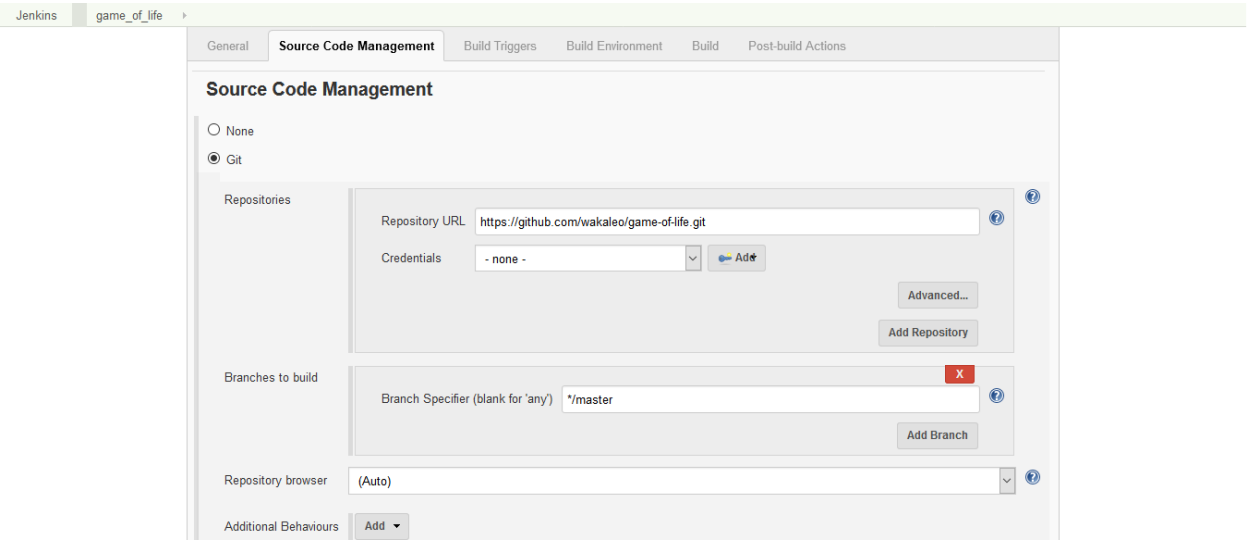
Here select which type of parameters you want.



For String based parameter. Select String Parameter.

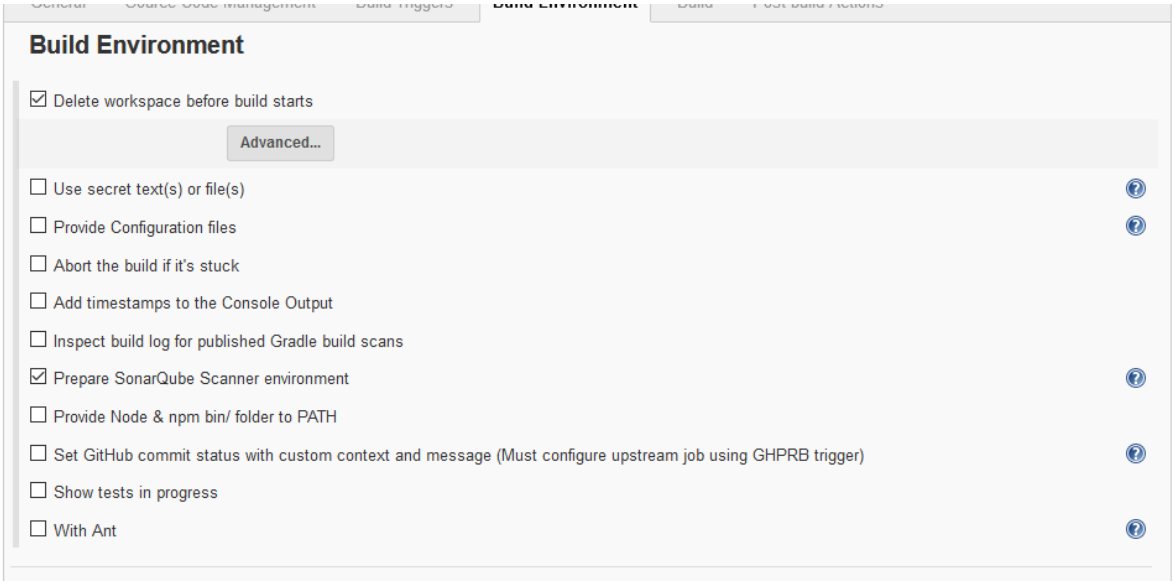


Step-8 In Source Code Management select Git and enter the .git URL



Step-9 In Build Environment, select Delete Workspace before build starts, if you want to delete workspace before build space, this could save the space occupies while building

the project.



To run SonarQube along with job then select Prepare SonarQube scanner environment.

Step-8 In Build select Invoke top-level Maven targets

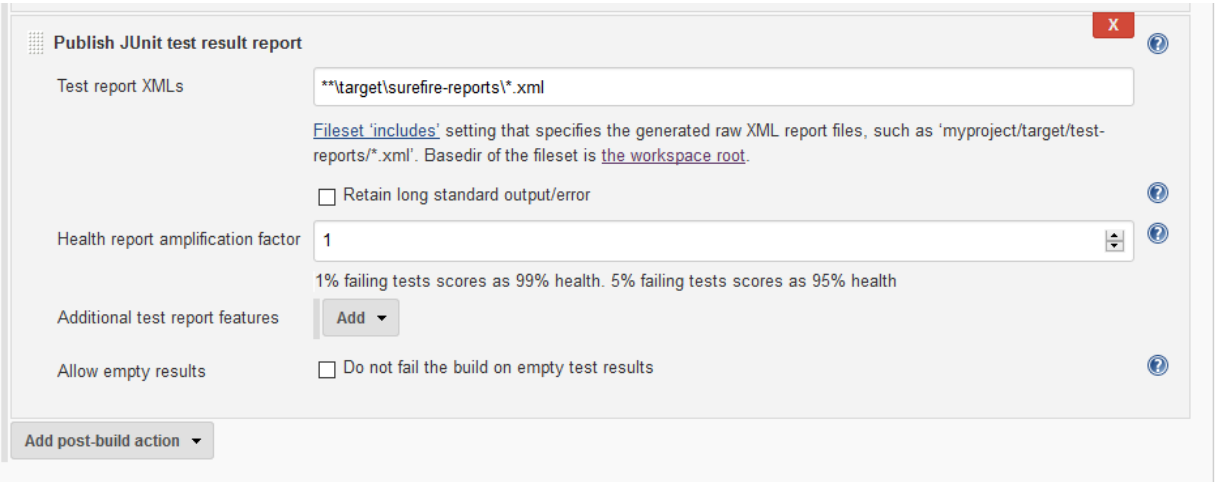
In Goals, enter the maven lifecycle as well as command to integrate SonarQube

clean install sonar:sonar -Dsonar.host.url=http://localhost:9000 -Dsonar.login=<sonarqube token>

Step-10 If you want to deploy in pipeline, then you have to copy that artifact so that same can be used in other project.

In Post-build Actions select Archive the artifacts, and enter the path which you want to archive.

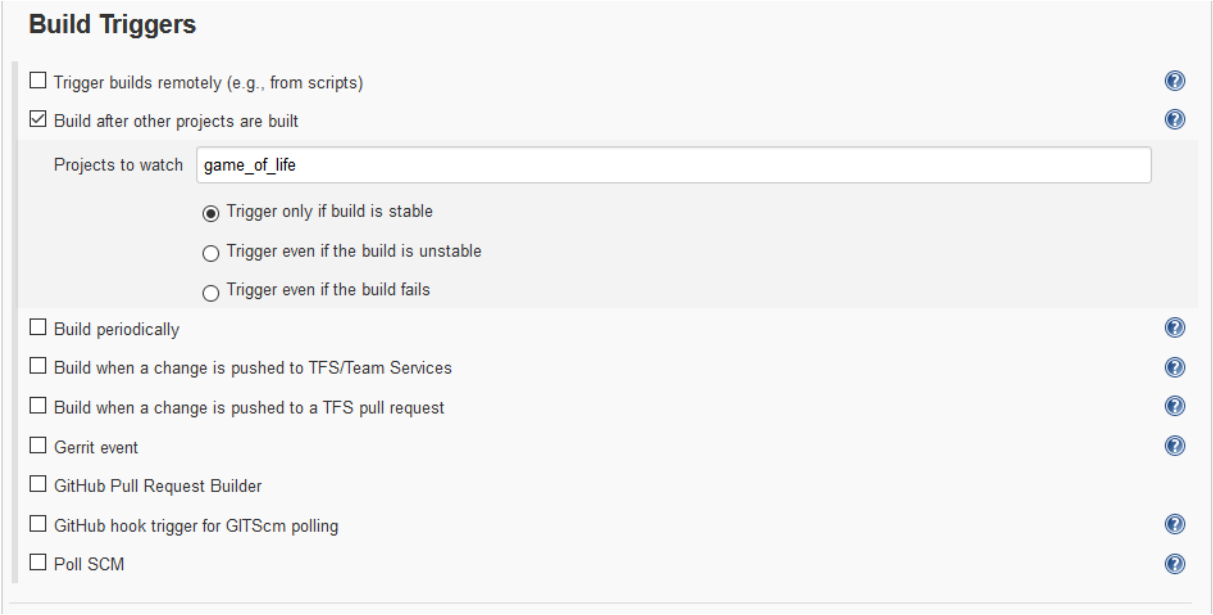
Step-11 To publish JUnit report select Publish JUnit test result report,



Followed by entering the target directory path where .xml is present.

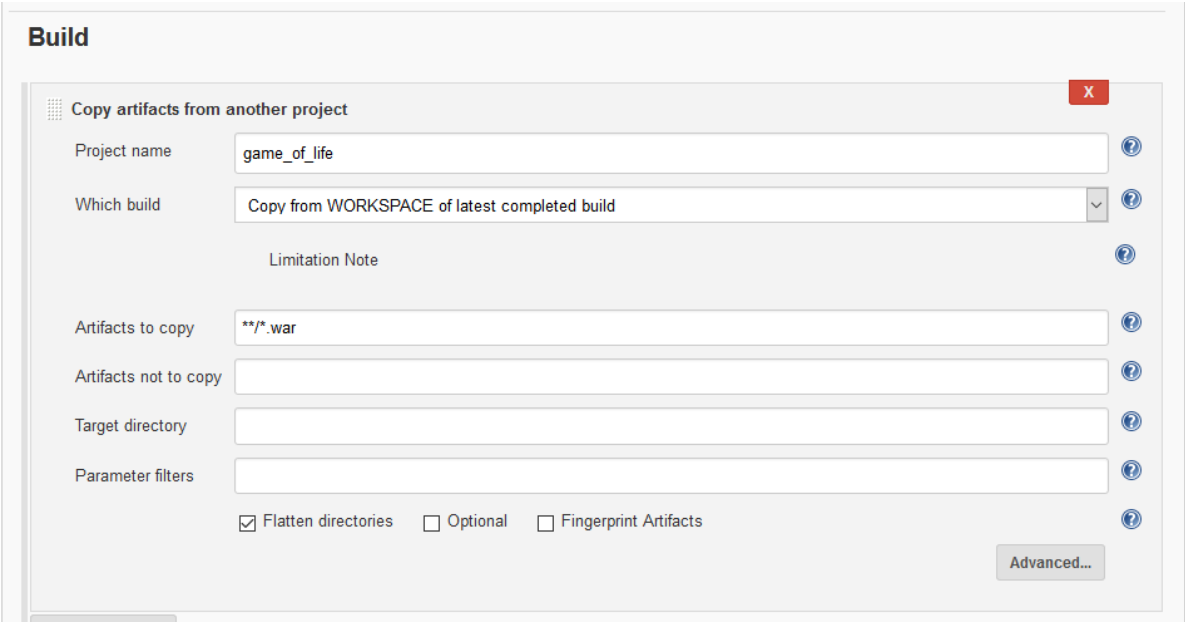
Click Apply and Save.

Step-12 To deploy the project, create another free style project, Since we are going to build a pipeline for project,



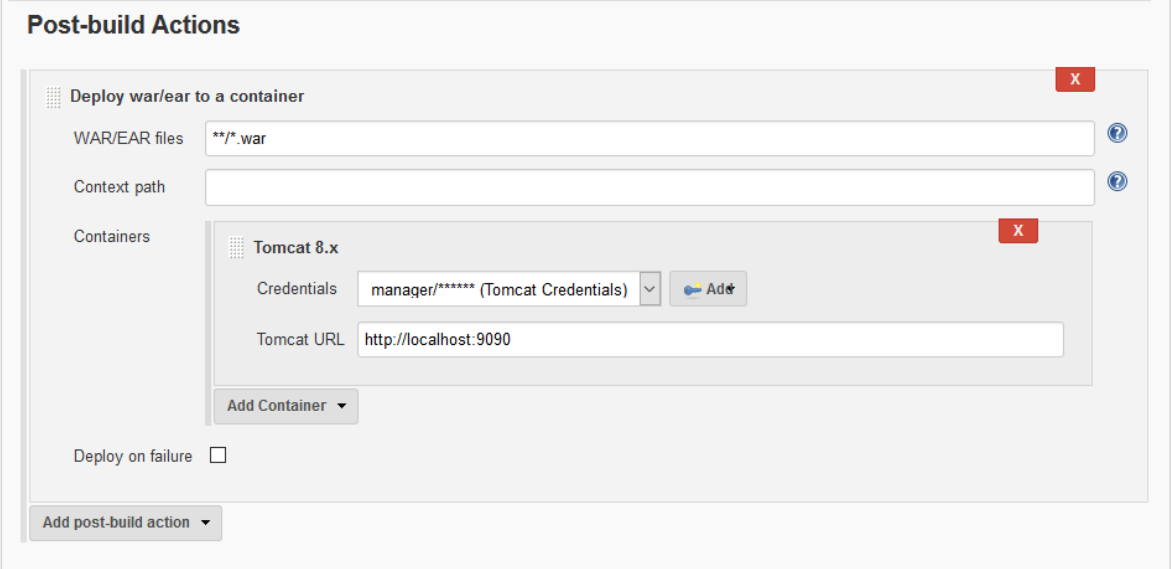
In Build Trigger select Build after other projects are built, select the name of the project which you want the job to run after.

Step-13 Now to deploy the project, we have to copy the artifacts of other project named game_of_life_deploy



Project, in Build, select Copy artifacts from another project, select the Project name, which build, since we want to copy the workspace of other project then we select Copy from WORKSPACE of latest complete build, Artifacts to copy, enter the path where .war is present on other project.

Step-13 Finally, we are going to deploy the project in Tomcat, then in Post Build actions select Deploy war/ear to a container.



Enter the path for .war, in container select the version of Tomcat, here is Tomcat8.x, select the credential for tomcat and enter the url fot the tomcat server. Make sure your tomcat is up and running.

Click Apply and Save.

Step-14 Build the upstream project i.e game_of_life,

It will appear like as shown above. You can see test result clicking Latest Test Result

Test Result

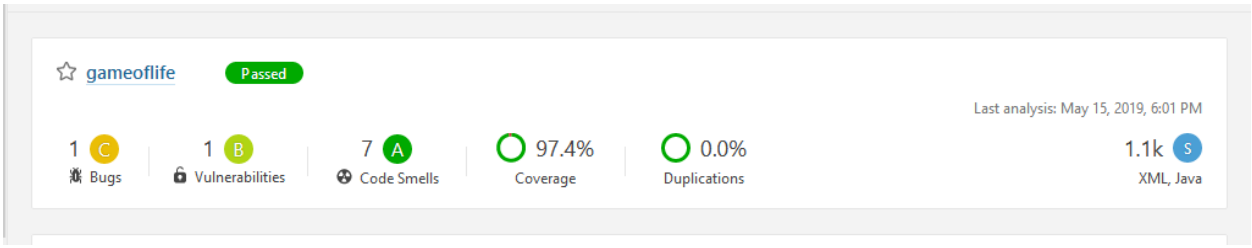
0 failures (±0) , 1 skipped (±0)

60 tests (±0)
Took 6.8 sec.
add description

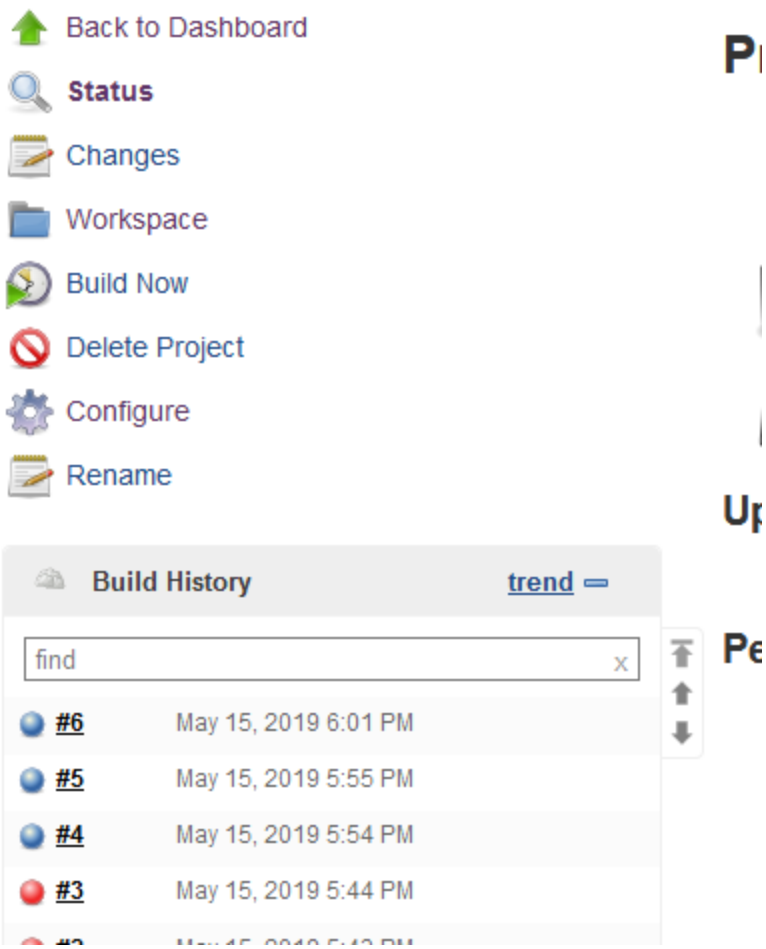
All Tests

Package	Duration	Fail (diff)	Skip (diff)	Pass (diff)	Total (diff)
behavior	95 ms	0	1	1	2
com.wakaleo.gameoflife.domain	0.1 sec	0	0	49	49
com.wakaleo.gameoflife.integration	14 ms	0	0	2	2
com.wakaleo.gameoflife.webtests.controllers	4.1 sec	0	0	7	7

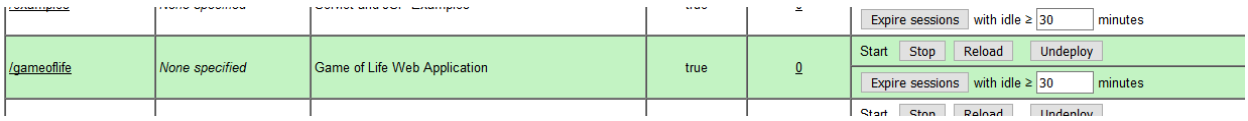
In SonarQube the project will appear,



After first build, it will move to build another project game_of_life_deployment.



After successful build, open tomcat url, go to manager App you will be able to see your build.



This is the final webpage

Welcome to Conway's Game Of Life!

This is a really cool web version of Conway's famous Game Of Life. The Game of Life is a cellular automaton devised by the British mathematician John Horton Conway way back in 1970.

The universe of the Game of Life is an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, live or dead. Every cell interacts with its eight neighbors, which are the cells that are directly horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

- Any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- Any live cell with more than three live neighbours dies, as if by overcrowding.
- Any live cell with two or three live neighbours lives on to the next generation.
- Any dead cell with exactly three live neighbours becomes a live cell.

New Game