

Game_of_life POC

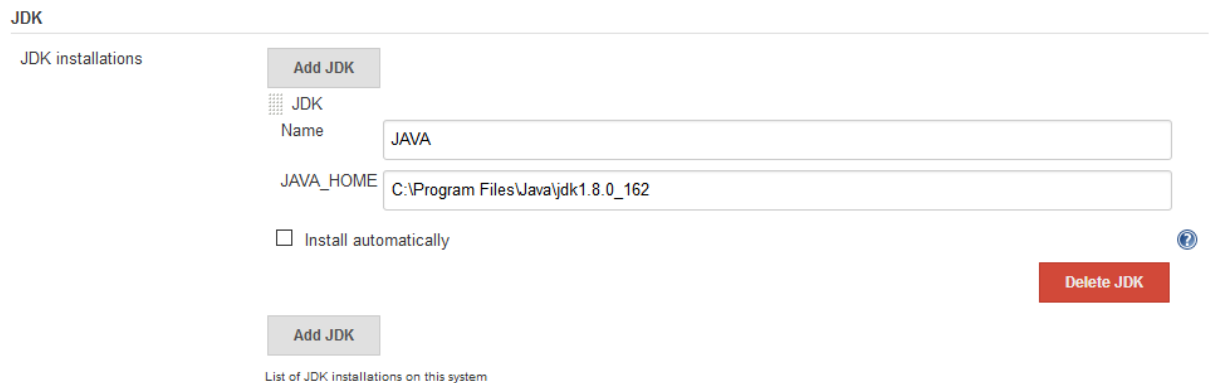
Install Jenkins with necessary packages and configure SonarQube server.

Step-1 In jenkins Install the following plugin,

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

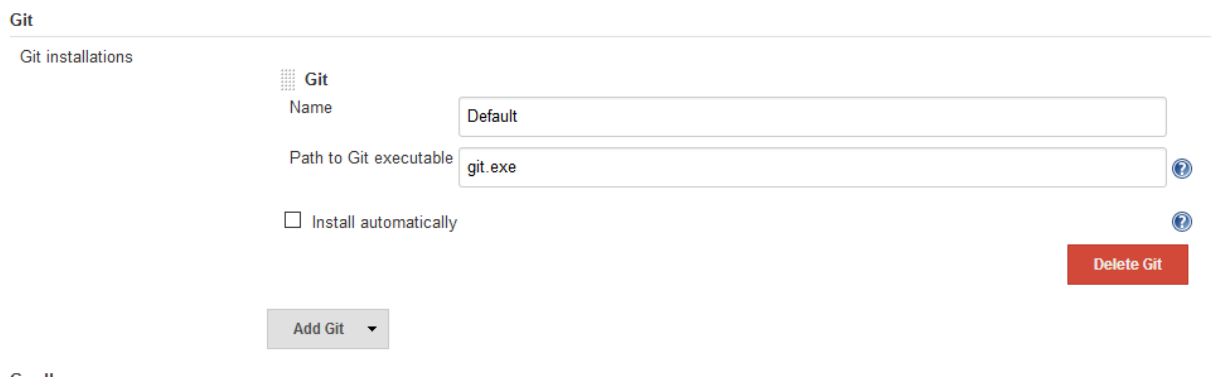
Step-2 Configure In Global Tool Configuration-

- For JAVA_JDK



The screenshot shows the 'JDK' configuration page in Jenkins. It has a sidebar with 'JDK installations' and a main area with an 'Add JDK' button. Below the button is a table with one entry: 'JDK' with 'Name' 'JAVA' and 'JAVA_HOME' 'C:\Program Files\Java\jdk1.8.0_162'. There is an unchecked checkbox for 'Install automatically' and a 'Delete JDK' button. At the bottom, there is another 'Add JDK' button and a link to 'List of JDK installations on this system'.

- For Git



The screenshot shows the 'Git' configuration page in Jenkins. It has a sidebar with 'Git installations' and a main area with an 'Add Git' button. Below the button is a table with one entry: 'Git' with 'Name' 'Default' and 'Path to Git executable' 'git.exe'. There is an unchecked checkbox for 'Install automatically' and a 'Delete Git' button. At the bottom, there is another 'Add Git' button and a link to 'List of Git installations on this system'.

- For Maven

Maven

Maven installations

Add Maven

Maven

Name

MAVEN_HOME

☐ Install automatically

Delete Maven

Add Maven

List of Maven installations on this system

Step-4 Configure SonarQube Server in Jenkins

SonarQube servers

Environment variables

☒ Enable injection of SonarQube server configuration as build environment variables

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name

Server URL

Default is http://localhost:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

Advanced...

Delete SonarQube


Add SonarQube

List of SonarQube installations


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

Enter an item name

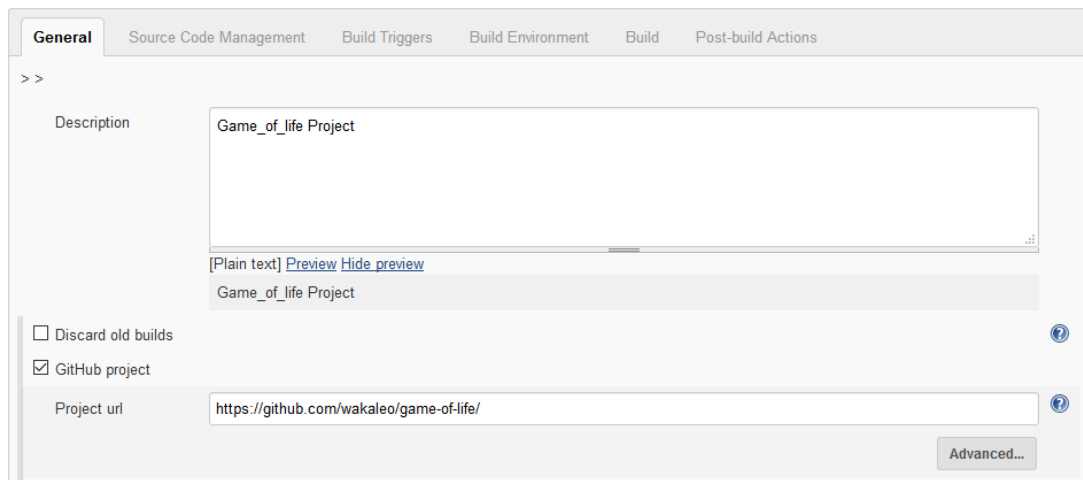
» This field cannot be empty, please enter a valid name

 **Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

 **Maven project**

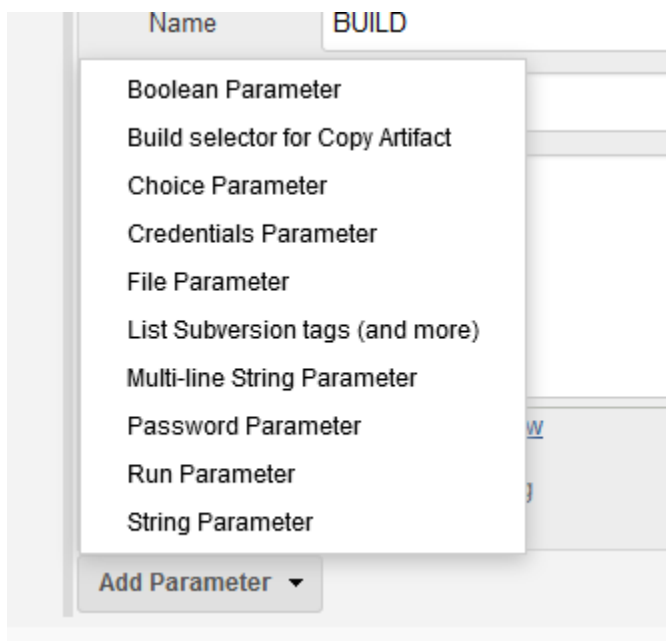
Step-6 Enter the Description, select GitHub project, then enter the repo URL.



The screenshot shows the 'General' tab of a Jenkins configuration page. The 'Description' field contains 'Game_of_life Project'. Below it, a preview shows the same text. The 'Discard old builds' checkbox is unchecked, and the 'GitHub project' checkbox is checked. The 'Project url' field contains 'https://github.com/wakaleo/game-of-life/'. An 'Advanced...' button is visible at the bottom right.

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

Here select which type of parameters you want.



The screenshot shows the 'Add Parameter' dropdown menu in the Jenkins configuration page. The menu is open, displaying a list of parameter types: Boolean Parameter, Build selector for Copy Artifact, Choice Parameter, Credentials Parameter, File Parameter, List Subversion tags (and more), Multi-line String Parameter, Password Parameter, Run Parameter, and String Parameter. The 'Add Parameter' button is visible at the bottom of the menu.

For String based parameter. Select String Parameter.

General Source Code Management Build Triggers Build Environment Build Post-build Actions

☐ This build requires lockable resources

☒ This project is parameterized

String Parameter

Name: BUILD

Default Value:

Description:

[Plain text] [Preview](#)

☐ Trim the string

Add Parameter

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

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

Jenkins game_of_life

Source Code Management

☐ None

☒ Git

Repositories

Repository URL: https://github.com/wakaleo/game-of-life.git

Credentials: - none -

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any'): */master

Add Branch

Repository browser: (Auto)

Additional Behaviours: Add

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.

Build Environment

☒ Delete workspace before build starts

Advanced...

☐ Use secret text(s) or file(s)

☐ Provide Configuration files

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Inspect build log for published Gradle build scans

☒ Prepare SonarQube Scanner environment

☐ Provide Node & npm bin/ folder to PATH

☐ Set GitHub commit status with custom context and message (Must configure upstream job using GHPRB trigger)

☐ Show tests in progress

☐ With Ant

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

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

Build

Invoke top-level Maven targets

Maven Version (Default)

Goals clean install sonar:sonar -Dsonar.host.url=http://localhost:9000 -Dsonar.login=fba1fab4552e7e43655fa5f5647923

Advanced...

Add build step

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

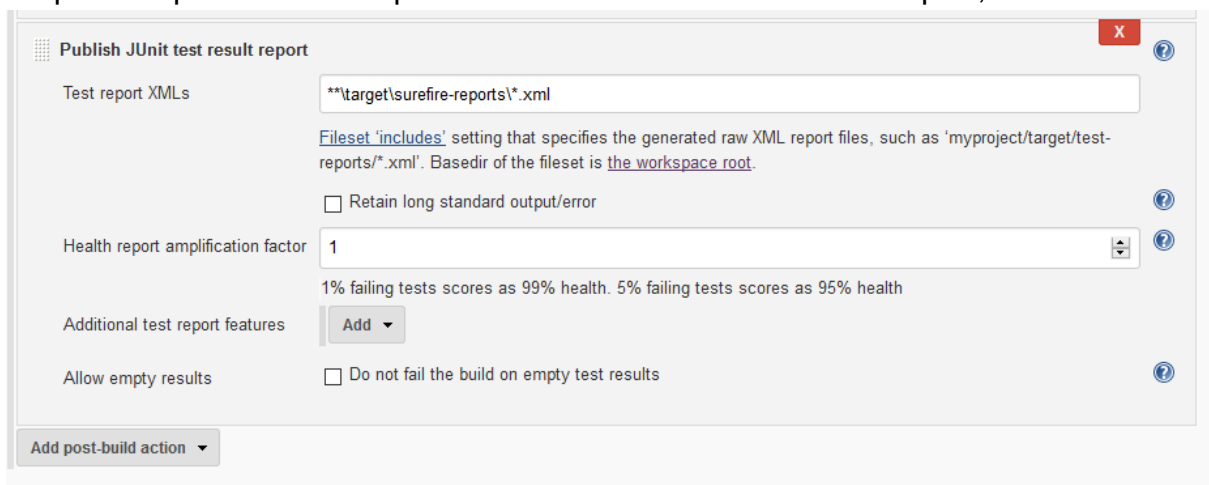
clean install sonar:sonar -Dsonar.host.url=http://localhost:9000 -
Dsonar.login=<sonarqube token>

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,

Build Triggers

☐ Trigger builds remotely (e.g., from scripts) ?

☒ Build after other projects are built ?

Projects to watch

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

☐ Build periodically ?

☐ Build when a change is pushed to TFS/Team Services ?

☐ Build when a change is pushed to a TFS pull request ?

☐ Gerrit event ?

☐ GitHub Pull Request Builder ?

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

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

Build

Copy artifacts from another project X

Project name ?

Which build ?

Limitation Note ?

Artifacts to copy ?

Artifacts not to copy ?

Target directory ?

Parameter filters ?

☒ Flatten directories ☐ Optional ☐ Fingerprint Artifacts ?

Advanced...

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.

The screenshot shows the 'Post-build Actions' configuration in Jenkins. The 'Deploy war/ear to a container' action is selected. The 'WAR/EAR files' field is set to '**/*.war'. The 'Context path' field is empty. Under the 'Containers' section, 'Tomcat 8.x' is selected. The 'Credentials' dropdown is set to 'manager/***** (Tomcat Credentials)'. The 'Tomcat URL' field is set to 'http://localhost:9090'. There is an 'Add Container' button and a 'Deploy on failure' checkbox which is unchecked. At the bottom, there is an 'Add post-build action' dropdown.

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 for 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,

The screenshot shows the Jenkins Jenkins UI for the 'game_of_life' project. The top navigation bar includes the Jenkins logo, a search bar, and user information (ashsid, log out). The left sidebar contains navigation links: Back to Dashboard, Status, Changes, Workspace, Build with Parameters, Delete Project, Configure, GitHub, SonarQube, and Rename. The main content area is titled 'Project game_of_life' and includes links for SonarQube, Workspace, Last Successful Artifacts (gameoflife.war, 3.04 MB), and Recent Changes. Below this, the 'SonarQube Quality Gate' section shows 'gameoflife' with a green 'OK' status and 'server-side processing: Success'. The 'Latest Test Result' section shows '(no failures)'. The 'Downstream Projects' section is partially visible. On the right, there is a 'Test Result Trend' chart showing a count of test results over time, with a 'Disable Project' button and a link to 'add description'. The bottom section shows the 'Build History' with a list of builds (#63, #62, #61, #60, #59) and their timestamps.

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,

☆ [gameoflife](#) Passed

Last analysis: May 15, 2019, 6:01 PM

1 C
Bugs

1 B
Vulnerabilities


7 A
Code Smells

97.4%
Coverage

0.0%
Duplications


1.1k S
XML, Java

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

 [Back to Dashboard](#)

 [Status](#)

 [Changes](#)


 [Workspace](#)

 [Build Now](#)

 [Delete Project](#)

 [Configure](#)

 [Rename](#)

 **Build History** [trend](#)

x

#6

May 15, 2019 6:01 PM

#5

May 15, 2019 5:55 PM

#4

May 15, 2019 5:54 PM

#3

May 15, 2019 5:44 PM

#2

May 15, 2019 5:43 PM

↑

↑

↓

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

/gameoflife	<i>None specified</i>	Game of Life Web Application	true	0	<div>Expire sessions with idle ≥ <input type="text" value="30"/> minutes</div> <div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ <input type="text" value="30"/> minutes</div> <div>Start Stop Reload Undeploy</div>
-----------------------------	-----------------------	------------------------------	------	---	---

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](#)