**CI FOR CSP COMPONENTS**

**Contents**

[**Steps for CSP Job configuration (EAR)** 3](#_Toc457836021)

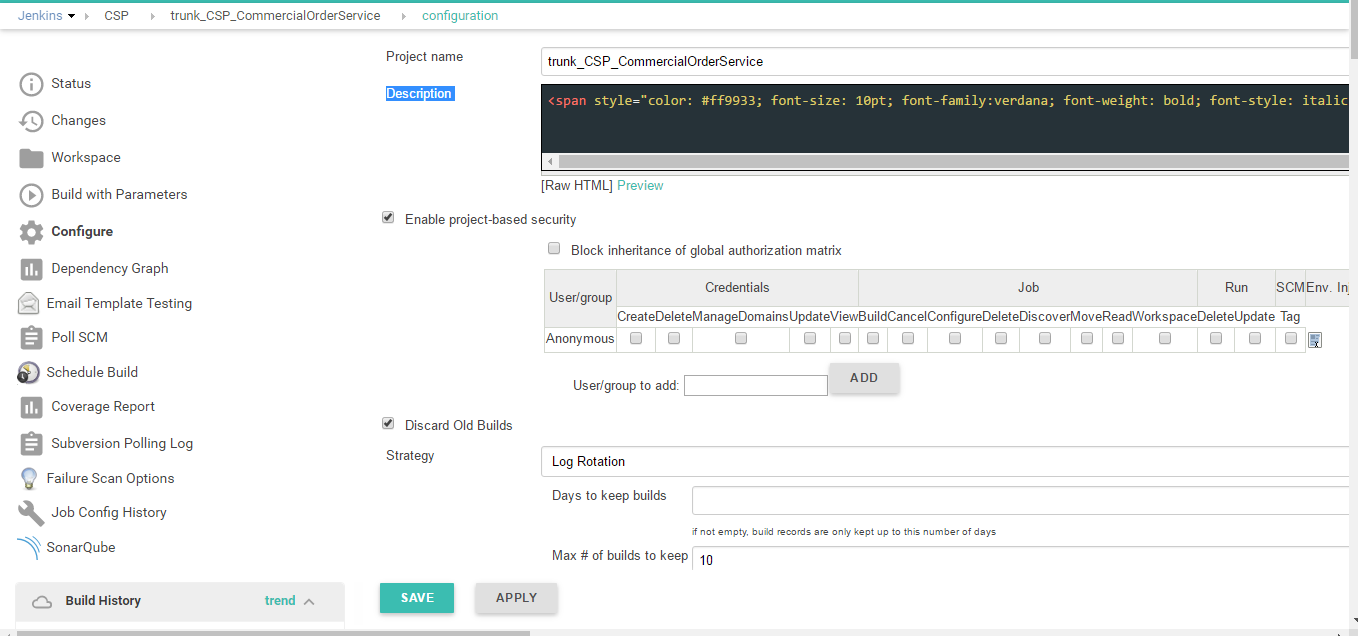
[**Steps for CSP Job configuration (JAR)** 13](#_Toc457836022)

# **Steps for CSP Job configuration (EAR)**

**Step 1**: Click on Configure and fill the require fields.

* **Project name**: Give the component name.
* **Description:** The description is in the HTML form, the HTML code is shown below :

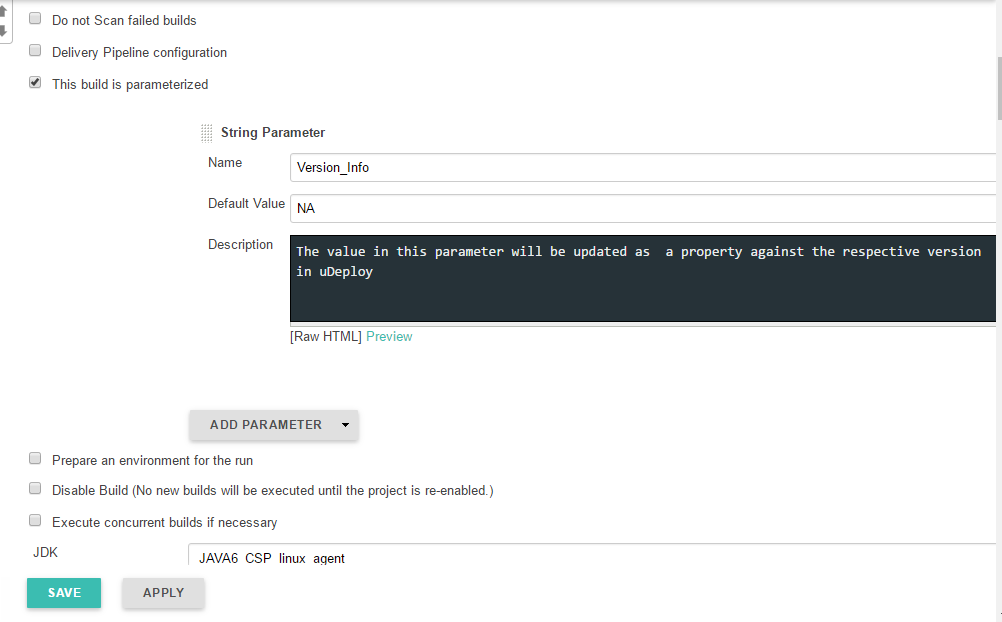
<span style="color: #ff9933; font-size: 10pt; font-family:verdana; font-weight: bold; font-style: italic; ">This job will do get latest,Build,package,Unit Testing,Sonar Analysis,Quality gate based on code metrics and publish the artifacts into IBM udeploy every 30 mins in case of any code changes in <u>CommercialAirtightProxy</u> component(EAR) of CSP application</span>



* Enable the project based security and Discard Old Builds, keep the Max # of builds to 10.

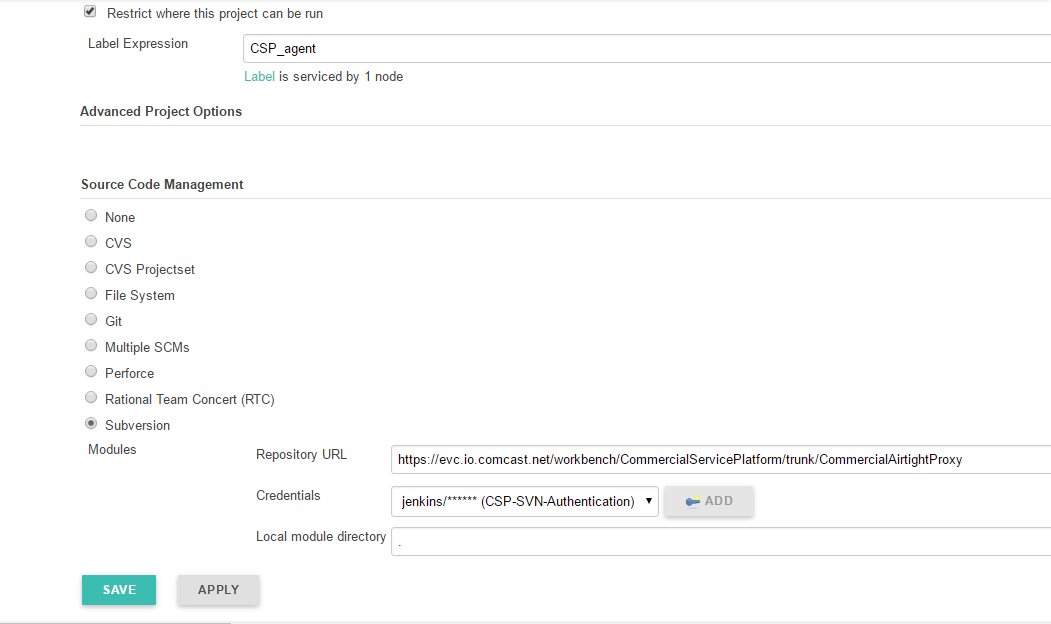
**Step 2:**

* Parameterize the build and Add Parameter (String parameter)
* Name: Give some name to the parameter.
* Default Value: NA.
* Description: Give description for the parameter.
* Keep the JDK as the JAVA6 CSP linux agent as Cloudset support only java version 6 with linux agent.



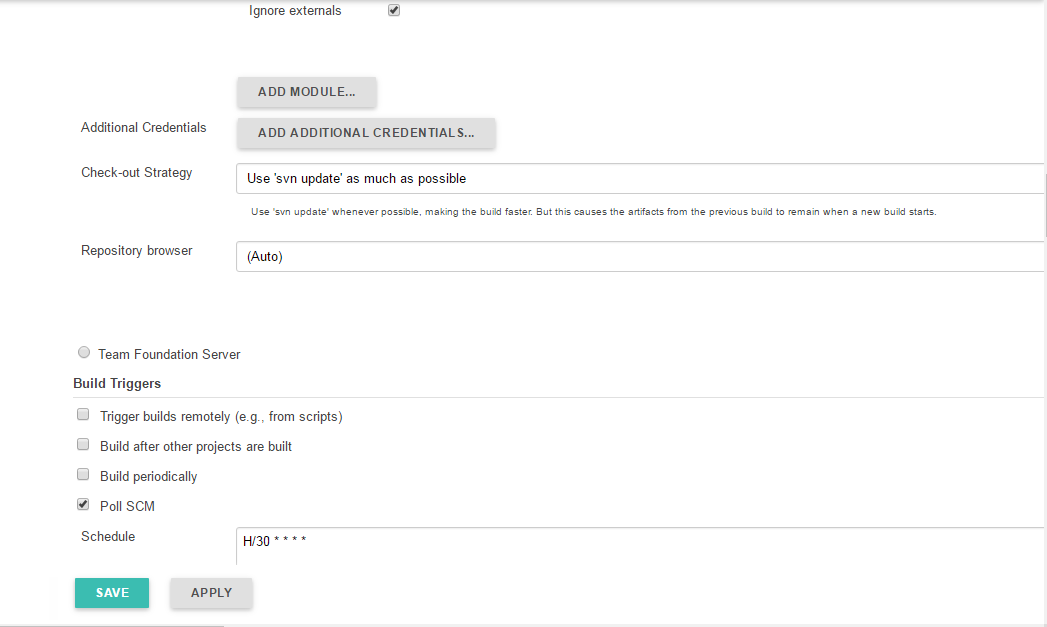
**Step 3:**

* Restrict the Project on the CSP Agent for all the CSP Components as it is an agent created for the components of CSP application.
* **SCM**: Select the Subversion and give the repository URL for the required component.
* **Credentials**: Authenticated User’s ID and password should be provided over here.



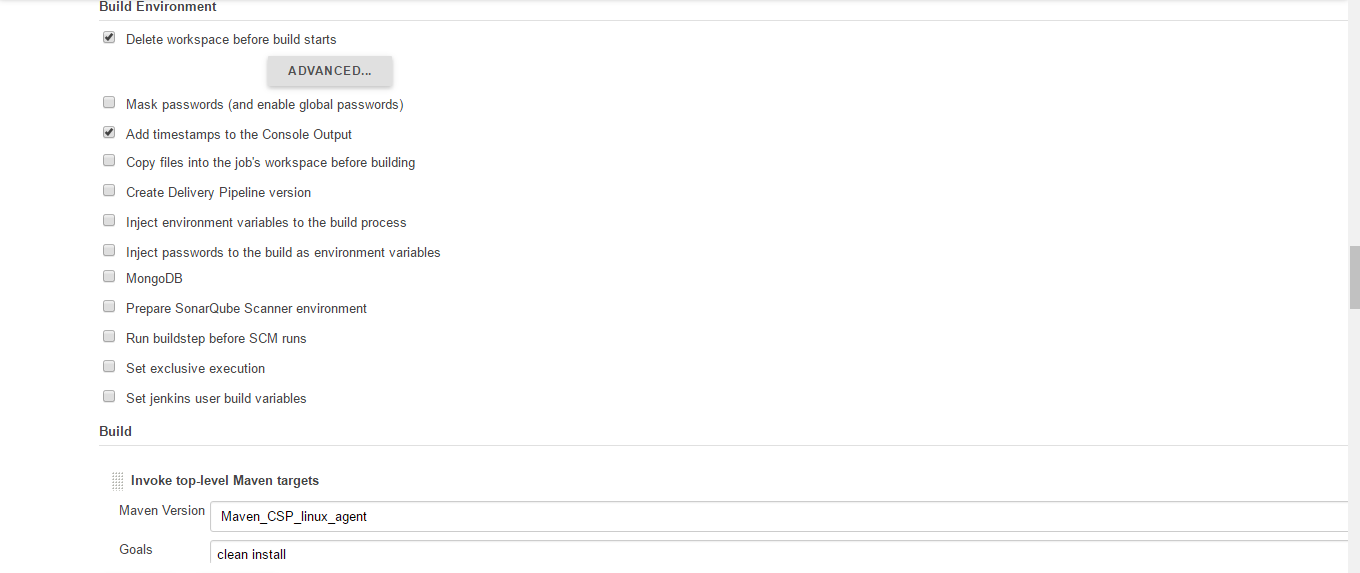
**Step 4:**

* Poll SCM in every 30 minutes, it will poll the SCM at every 30 minutes and build the job itself so that every new change in SCM will be reflected in the job build.



**Step 5:**

* Check the **“Delete the workspace before build start”** check box so that the old workspace does not conflict with the new one and the workspace contain the fresh folders.
* Check the **“Add timestamp to the Console Output”** (wrap the rest of the pipeline script).



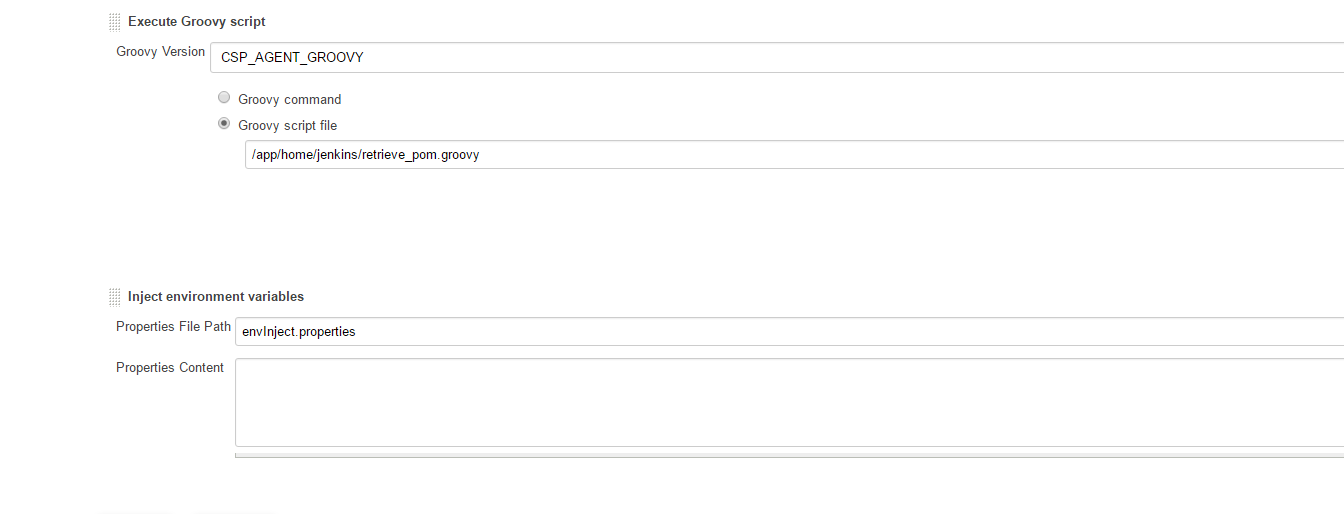
* Add a Build step as **“Invoke top-level Maven target”.**
* **Maven Version**: Select as Maven\_CSP\_linux\_agent (It’s a maven version which supports the CSP application).
* **Goal**: If there are **test cases** in a particular component then provide the **Goal** as clean install but if there are **no test cases** for a particular component then the Goal should have the skip test cases then in the **Goal** provide clean install -Dmaven.test.skip=true.

**Step 6:**

* Add next build step as **Execute Groovy Script.**
* **Groovy Version**: Select the CSP\_ AGENT \_GROOVY.
* **Groovy script file**: /app/home/jenkins/retrieve\_pom.groovy.
* Add next build step as **“Inject environment variables”.**

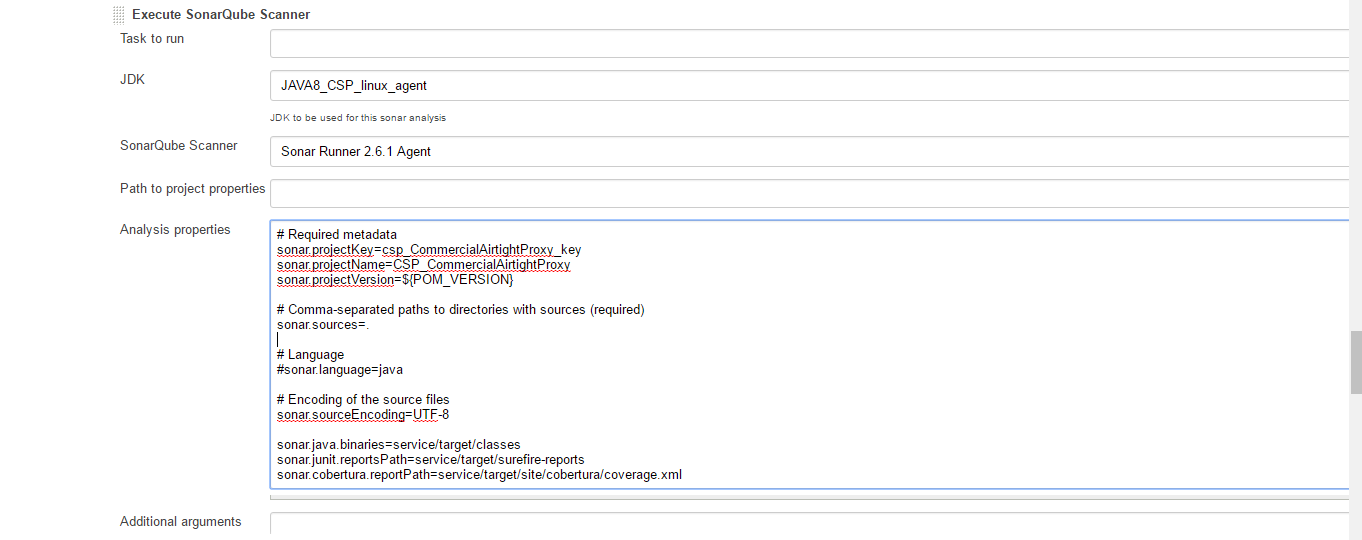
The Groovy script generate some variable from above step, those variable are injected as environment variable with the help of Inject environment variables.

* **Properties File Path**: Provide as envInject.properties.



**Step 7:**

* Add next Build Step as **Execute SonarQube Scanner**.
* **JDK**: Provide as JAVA8\_CSP\_linux\_agent as the sonar supports java version 8 with the CSP linux agent.
* **SonarQube** **Scanner**: SonarRunner 2.6.1 Agent(Sonar-Runner).
* **Analysis properties**: Provide the appropriate Sonar Properties which will contain the projectKey,projectName,projectVersion, sonar java binaries(if the project does not have any test cases) but if the project has test cases then the junit report path and cobetura report path should also be mentioned in the analyses properties along with the projectKey,projectName,projectVersion, sonar java binaries.



**Analysis properties**

# Required metadata

sonar.projectKey=csp\_CommercialAirtightProxy\_key

sonar.projectName=CSP\_CommercialAirtightProxy

sonar.projectVersion=${POM\_VERSION}

# Comma-separated paths to directories with sources (required)

sonar.sources=.

# Language

#sonar.language=java

# Encoding of the source files

sonar.sourceEncoding=UTF-8

sonar.java.binaries=service/target/classes

sonar.junit.reportsPath=service/target/surefire-reports

sonar.cobertura.reportPath=service/target/site/cobertura/coverage.xml

**Step 8:**

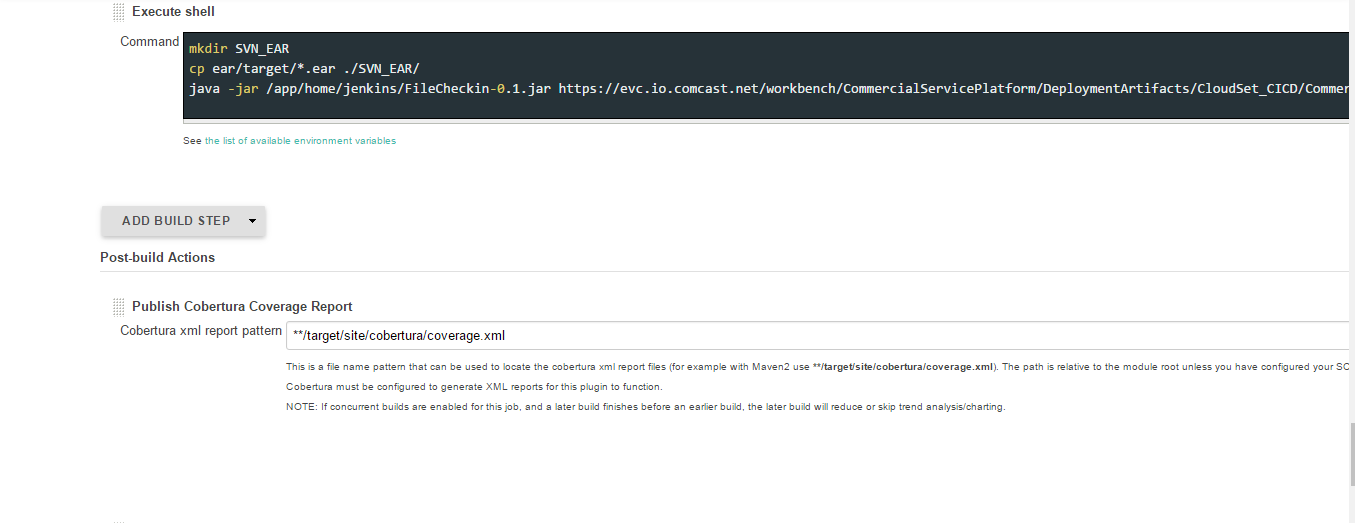
* Add the next build Step as **Execute Shell** and provide the appropriate shell script over it.

mkdir SVN\_EAR

cp ear/target/\*.ear ./SVN\_EAR/

java -jar /app/home/jenkins/FileCheckin-0.1.jar https://evc.io.comcast.net/workbench/CommercialServicePlatform/DeploymentArtifacts/CloudSet\_CICD/CommercialAirtightProxy/${POM\_VERSION}.${BUILD\_ID}/EAR/ ./SVN\_EAR

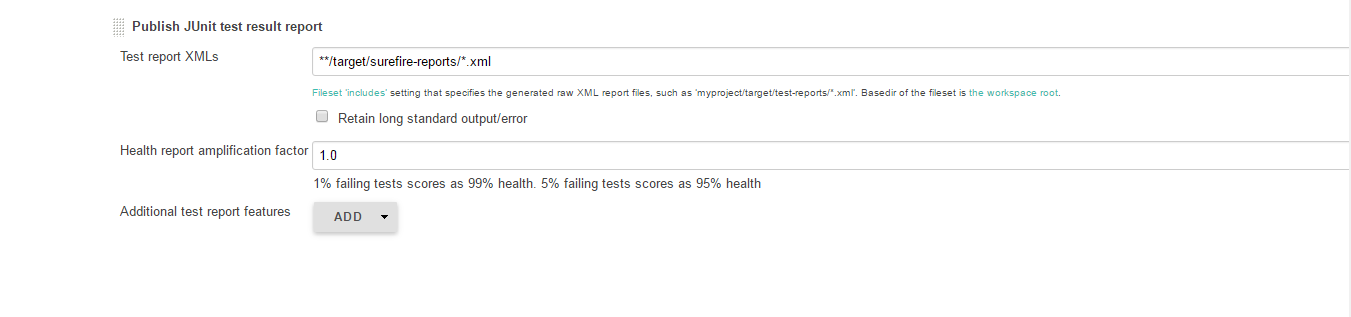
* Execute shell will move the EAR file to the SVN repository (path specified in the script).



* In case the component has test Cases add a post build **step** as **Publish Cobetura Coverage Report (if there are no test cases skip this step).**
* **Cobetura xml report pattern**: \*\*/target/site/cobertura/coverage.xml (it will take the path of the coverage report from the workspace).

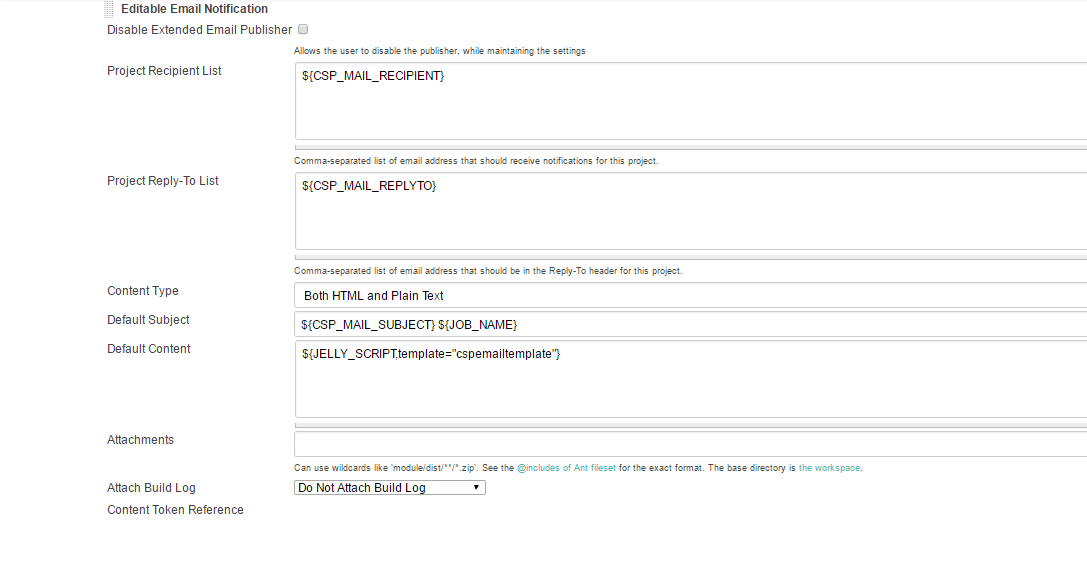
**Step 9:**

* In case the component has test Cases add next **post build** **step** as **Publish Junit test result report (if there are no test cases skip this step).**
* **Test report XMLs:** \*\*/target/surefire-reports/\*.xml (it will take the path of the test report from the workspace).
* In Health report amplification factor, give, 1.0.



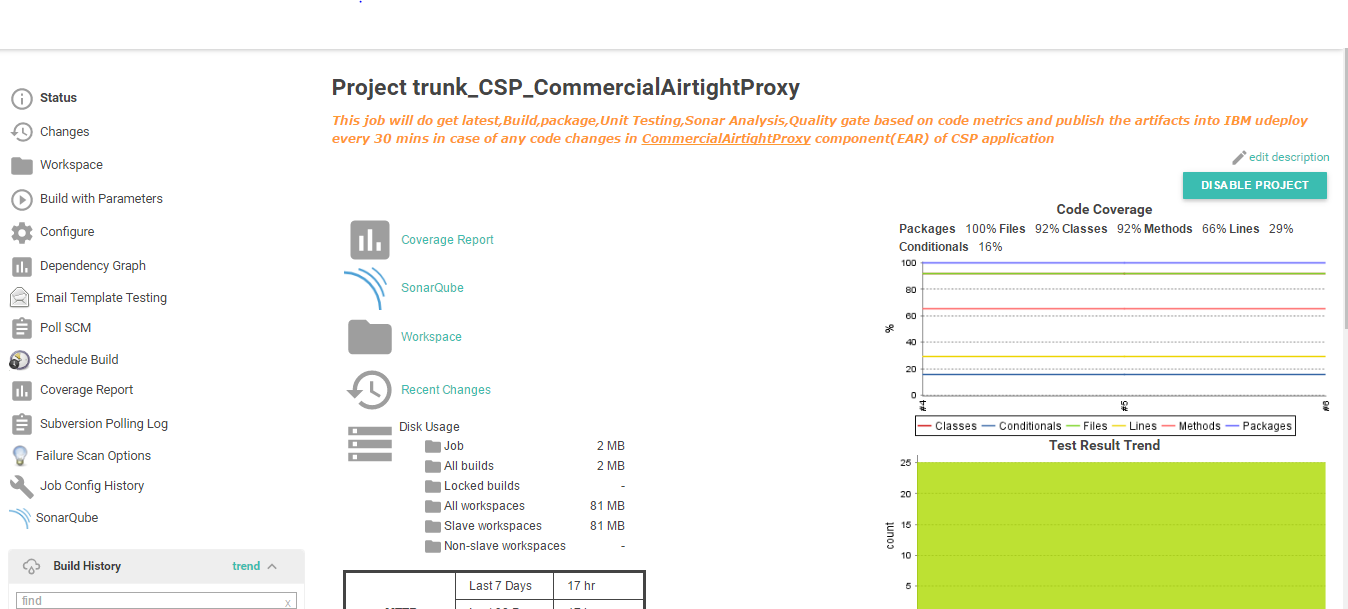
**Step 10:**

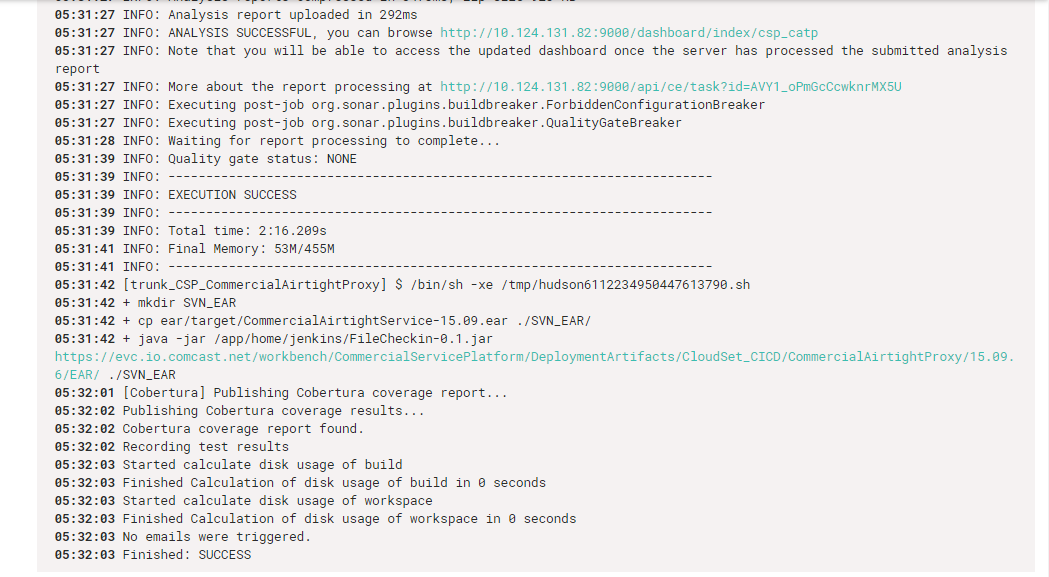
* Add the next **Post Build Step** as **Editable Email Notification**.
* **Post Recipient List**: Specify a single recipient list for each email that is sent (${CSP\_MAIL\_RECIPIENT}).
* **Project Reply-To-List**: ${CSP\_MAIL\_REPLYTO}
* **Content Type**: Select the content type as Both HTML and Plain Text
* **Default Subject**: This will define the subject of the mail sent to the recipient${CSP\_MAIL\_SUBJECT} ${JOB\_NAME}
* **Default Content**: Content of the e-mail ${JELLY\_SCRIPT,template="cspemailtemplate"}
* **Attach Build Log**: Select it as Do Not Attach Build Log.



Step 11:

* Click **Apply** and **Save** the configuration and then **Build with Parameters.**

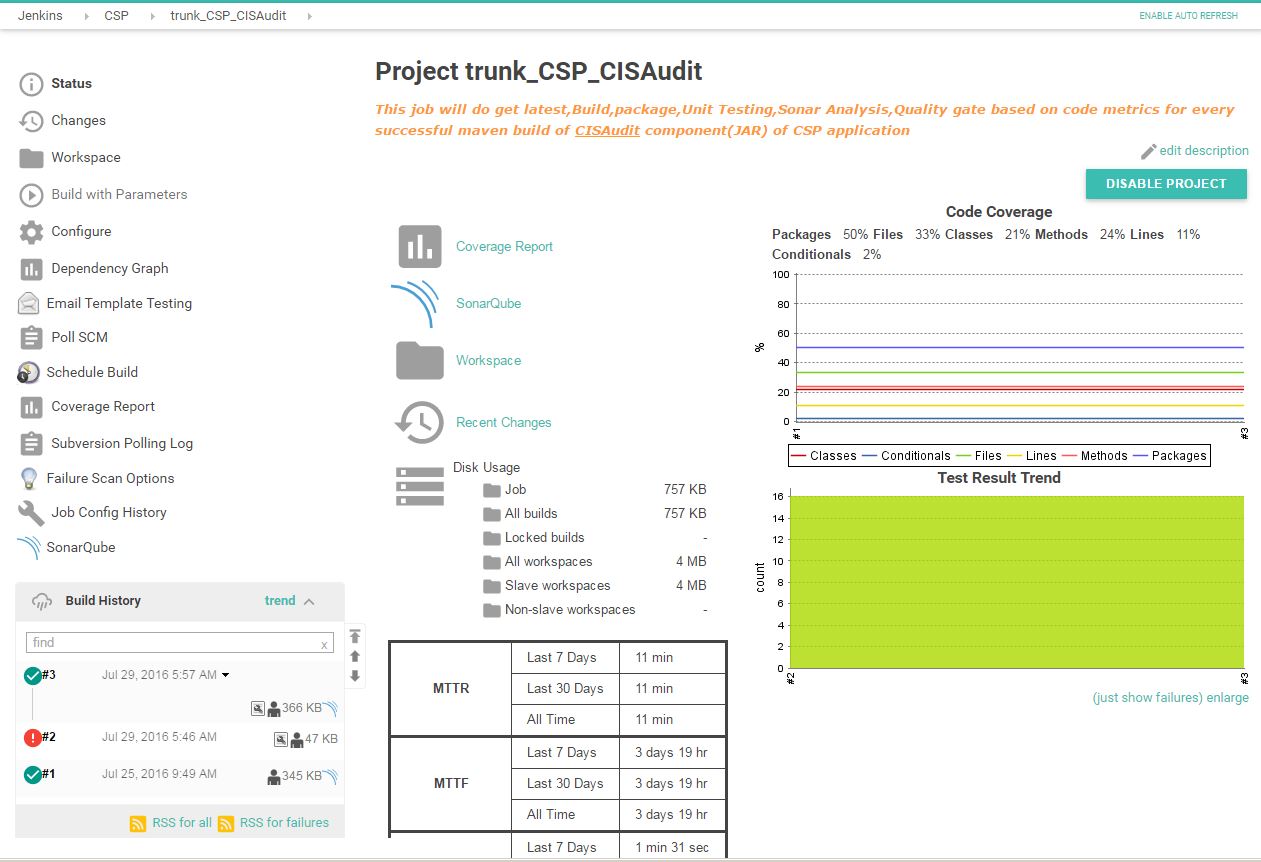




The highlighted path is the path where the EAR file is checked-in and one can access this path to see the EAR File.

# **Steps for CSP Job configuration (JAR)**

**Step 1**-:

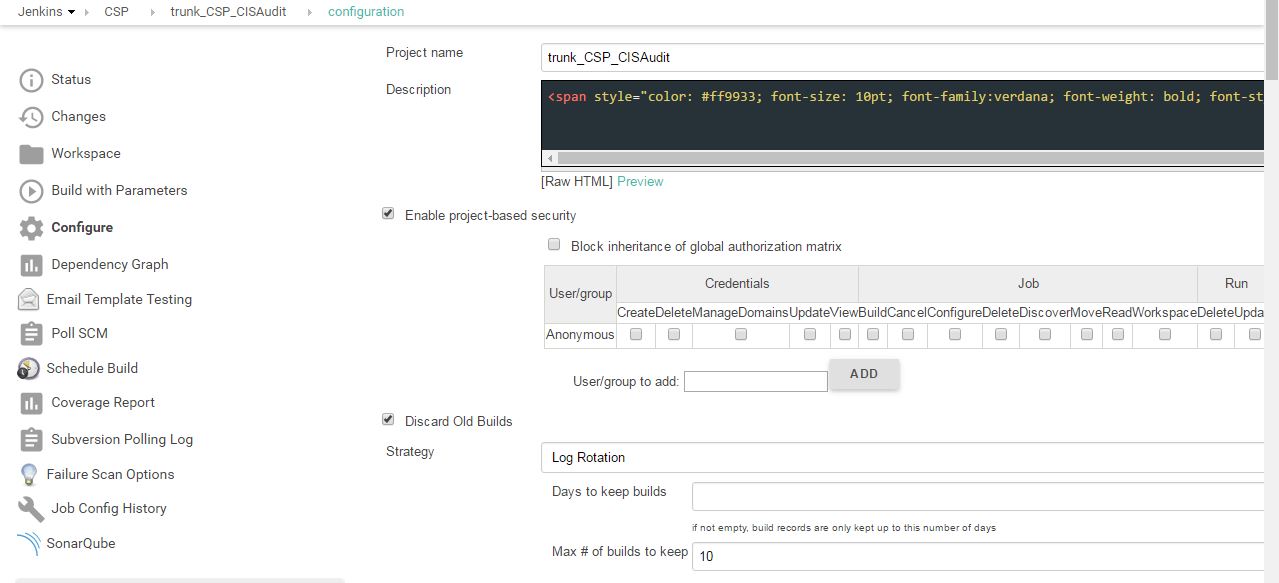
* Go to **Configure** option in Cloudset. 

**Step 2**-:

* Give the **Project Name**.
* Description -: Give the following description

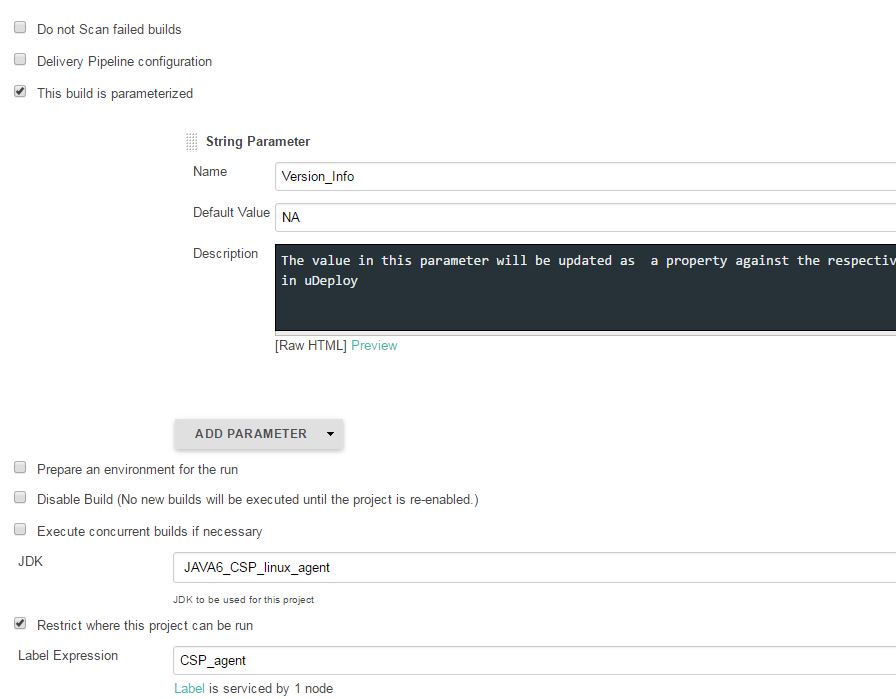
<span style="color: #ff9933; font-size: 10pt; font-family:verdana; font-weight: bold; font-style: italic; ">This job will do get latest,Build,package,Unit Testing,Sonar Analysis,Quality gate based on code metrics for every successful maven build of <u>CISAudit</u> component(JAR) of CSP application</span>

* Tick the **Enable project-based security** and **Discard Old Builds**.



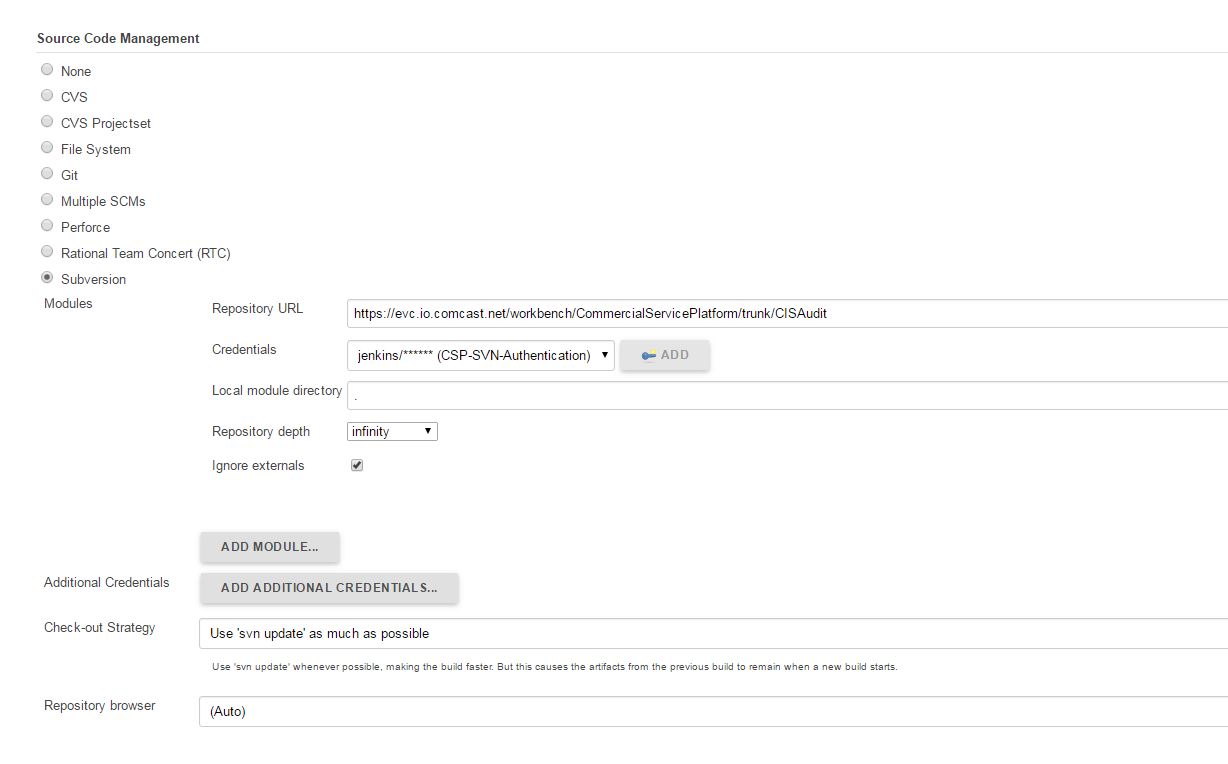
**Step 3**-:

* Parameterize the build and Add Parameter (String parameter)
* Name: Give some name to the parameter.
* Default Value: NA.
* Description: Give description for the parameter.
* Keep the JDK as the JAVA6 CSP linux agent as Cloudset support only java version 6 with linux agent.
* Restrict the project to run on CSP\_agent for all the CSP Components as it is an agent created for the components of CSP application.



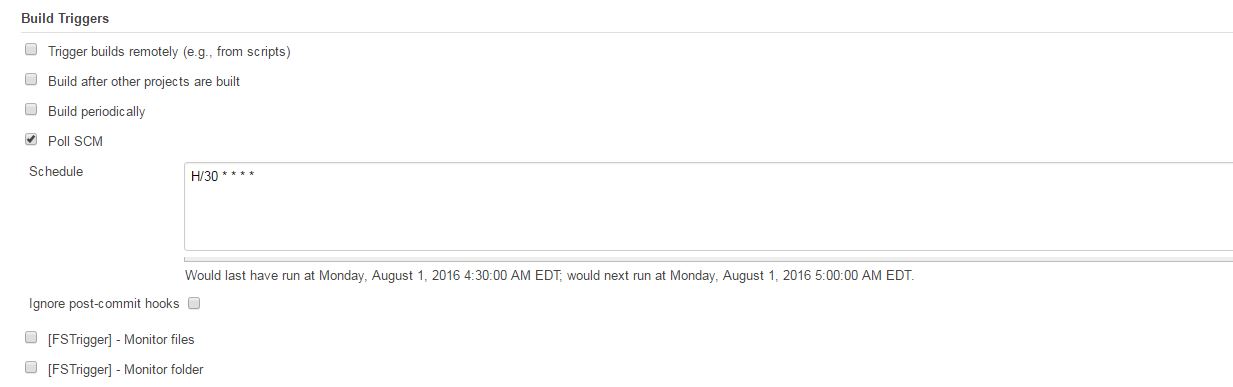
**Step 4**-:

* **SCM**: Select the Subversion and give the repository URL for the required component.
* **Credentials**: Authenticated User’s ID and password should be provided over here.



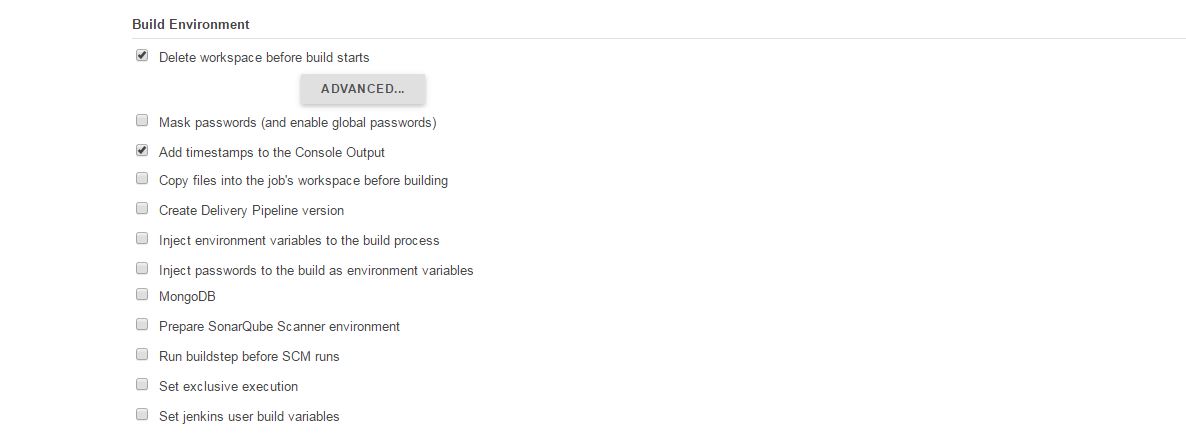
**Step 5**-:

* In **Build Trigger** option, tick **Poll SCM**.
* Poll SCM in every 30 minutes, it will poll the SCM at every 30 minutes and build the job itself so that every new change in SCM will be reflected in the job build.



**Step 6**-:

* In **Build Environment**, Check the **“Delete the workspace before build start”** check box so that the old workspace does not conflict with the new one and the workspace contain the fresh folders.
* Check the **“Add timestamp to the Console Output”** (wrap the rest of the pipeline script).



**Step 7**-:

* On the bottom of page from **Add Build Step**, select **“Invoke top-level Maven target”**.
* Specify **Maven version as** Maven\_CSP\_linux\_agent.
* If the particular component has test cases, then specify the goals as, clean -e install.
* If the particular component has no tests cases, then specify the goals as clean -e install -Dmaven.test.skip=true.



**Step 8**-:

* From **Add Build Step**, select **“Execute Groovy Script”**.
* Specify **Groovy** **Version** as CSP\_AGENT\_Goovy.
* Select **Groovy Script file** as /app/home/jenkins/retrieve\_pom.groovy



**Step 9**-:

* From **Add Build Step**, select **“Inject Environment variables”**.
* Specify **Properties file path as** envInject.properties**.**



**Step 10-:**

* **From Add Build Step,** select **“Execute SonarQube”.**
* **JDK**: Provide as JAVA8\_CSP\_linux\_agent.
* **SonarQube** **Scanner**: SonarRunner 2.6.1 Agent.
* Specify the **Analysis Properties** as shown below.

# Required metadata

sonar.projectKey=csp\_CISAudit\_key

sonar.projectName=CSP\_CISAudit

sonar.projectVersion=${POM\_VERSION}

# Comma-separated paths to directories with sources (required)

sonar.sources=.

# Language

#sonar.language=java

# Encoding of the source files

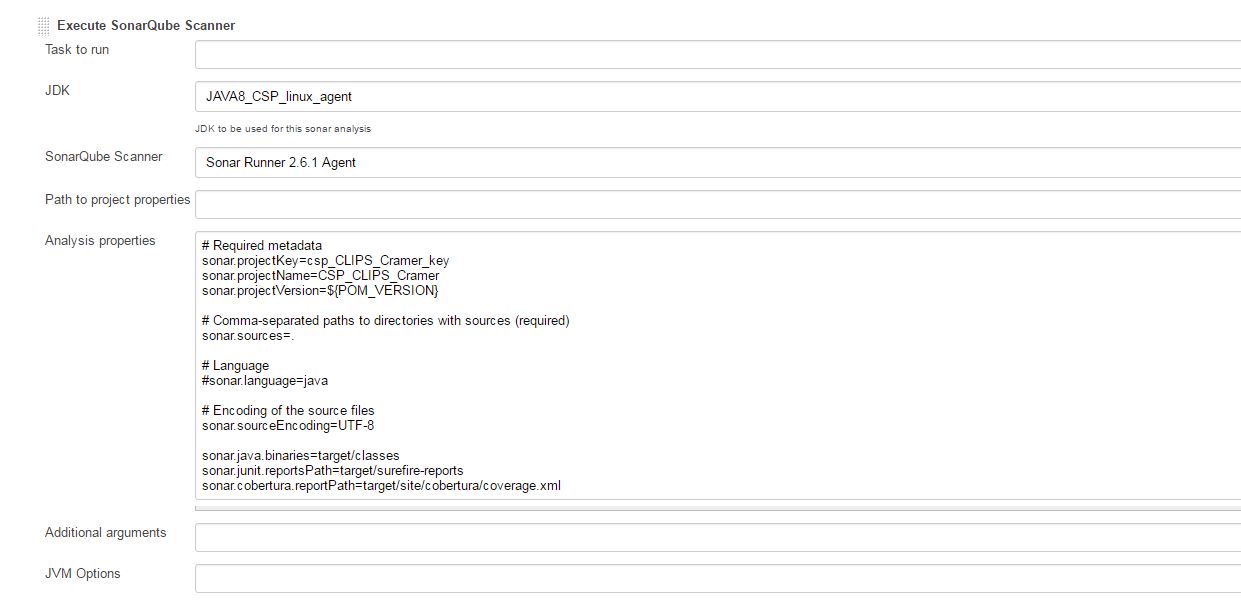
sonar.sourceEncoding=UTF-8

sonar.java.binaries=target/classes

sonar.junit.reportsPath=target/surefire-reports

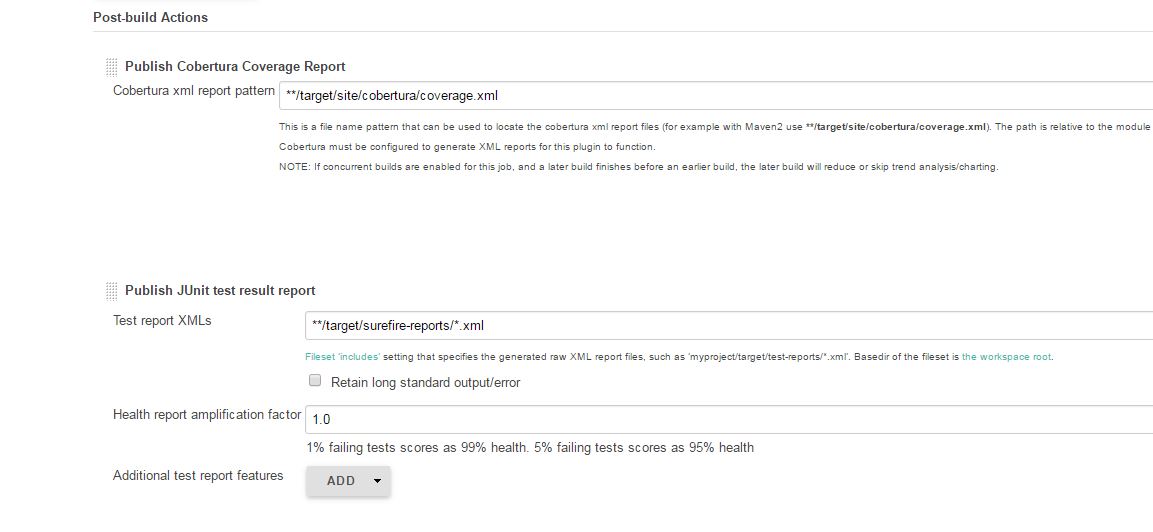
sonar.cobertura.reportPath=target/site/cobertura/coverage.xml

* In case if component has no test cases, then do not specify sonar.junit.reportsPath and sonar.cobertura.reportPath.



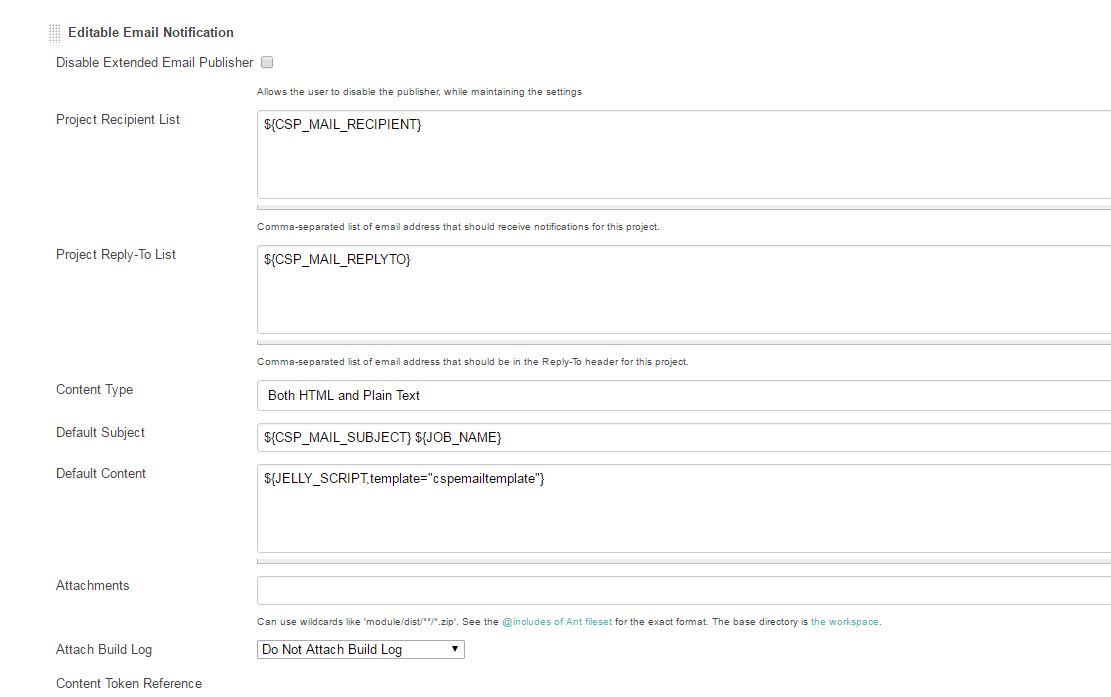
Step 11-:

* From **Add Post Built Action**, select **“Publish Cobertura Coverage Report”**, and **“Publish Junit test result report”**.
* In cobertura xml report pattern, give, \*\*/target/site/cobertura/coverage.xml.
* In text report xml, give, \*\*/target/surefire-reports/\*.xml.
* In Health report amplification factor, give, 1.0.



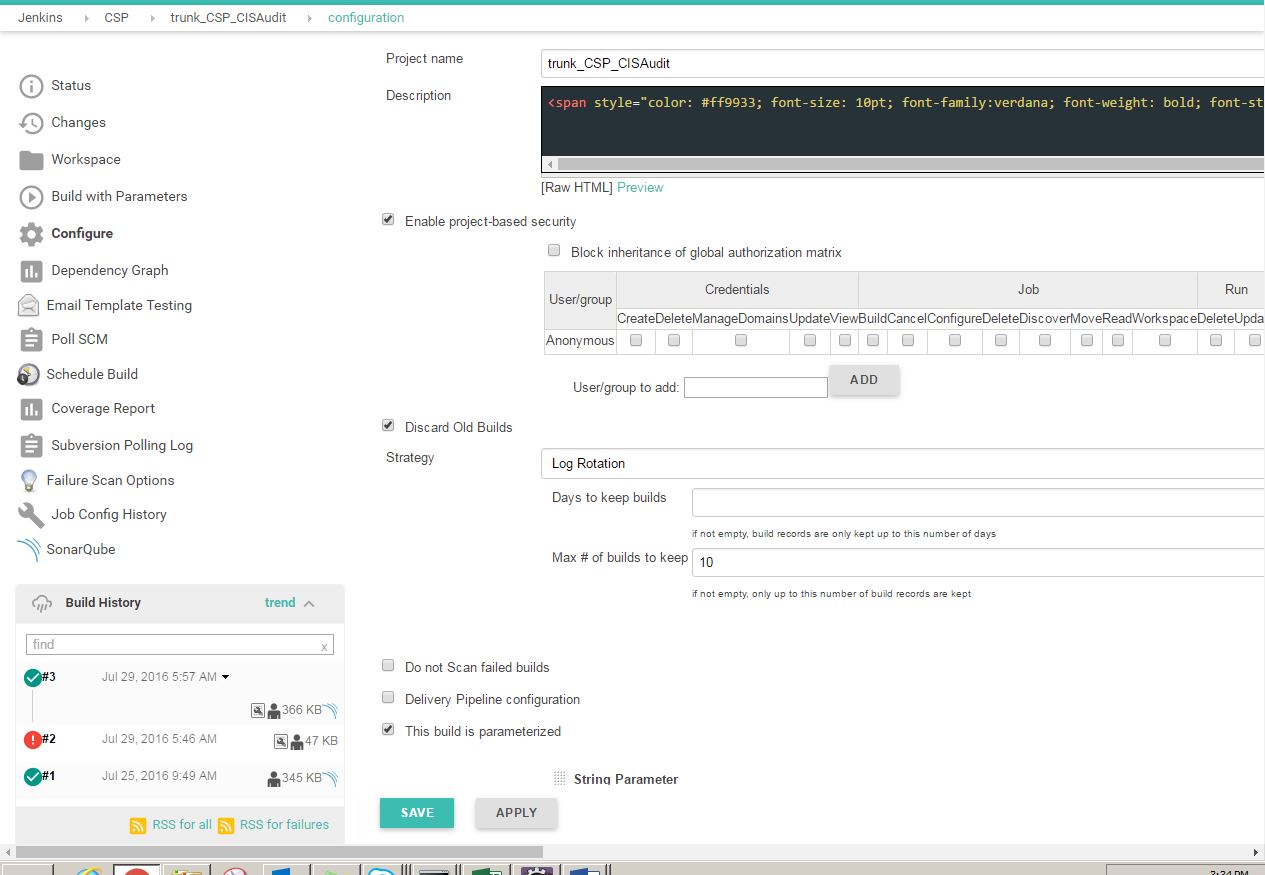
Step 12-:

* From **Add Post Built Action**, select **“Editable Email Notification”**.
* **Post Recipient List**: Specify a single recipient list for each email that is sent (${CSP\_MAIL\_RECIPIENT}).
* **Project Reply-To-List**: ${CSP\_MAIL\_REPLYTO}
* **Content Type**: Select the content type as Both HTML and Plain Text
* **Default Subject**: This will define the subject of the mail sent to the recipient${CSP\_MAIL\_SUBJECT} ${JOB\_NAME}
* **Default Content**: Content of the e-mail ${JELLY\_SCRIPT,template="cspemailtemplate"}
* **Attach Build Log**: Select it as Do Not Attach Build Log.



Step 13-:

* Click **Apply** and **Save** the configuration and then **“Build with Parameters”.**



Step 14-:

* Check the **“Console Output”** of he build, the build is successful. 