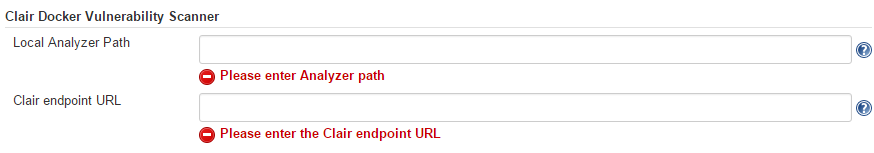
**Documentation for Clair Vulnerabilities Scanner Plugin**

Jenkins configurations:

Below are two fields to be configured at global level Jenkins configurations:  


Below field to be configured at build step of job level Jenkins configurations:



These fields are populated from the **jelly files** that are present under the **src/main/resources** directory  
and help files are present under the same directory.

Command

/home/jenkins/analyze-local-images -endpoint [http://10.242.138.116:6060](http://10.242.138.116:6060/) -my-address 10.242.138.115 tomcat:17

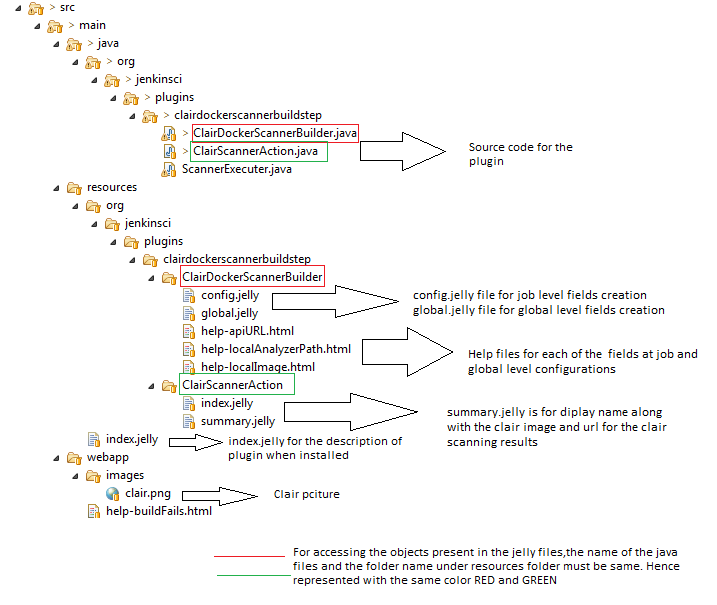
Address of analyzer tool image Clair server address

To copy the war to tomcat webapp folder - docker cp /home/jenkins/workspace/ClairSCM/target/jpetstore.war (id of image):/usr/local/tomcat/webapps/

To commit the image if we make some changes in image - docker commit (id of image) tomcat:17

Image with tag

Below is the folder structure for the Source code of Clair Vulnerabilities Scanner plugin:



Modules for the plugin:

* ClairDockerScannerBuilder.java
* ScannerExecuter.java
* ClairScannerAction.java

**ClariDockerScannerBuilder.java**

public class ClairDockerScannerBuilder extends Builder {

// includes the getters and setters methods for getting the objects declared in the jelly files  
   
   
 **public** **boolean** perform(AbstractBuild build, Launcher launcher, BuildListener listener)

**throws** AbortException, java.lang.InterruptedException {

// perform method is used for “Build now” action of the job.

}

//calls ScannerExecuter class for executing the **docker** command for the scanning of image specified

// calls ClairScannerAction class for displaying the scanning results of the image.

private void archiveArtifacts(AbstractBuild build, Launcher launcher, BuildListener listener)

throws java.lang.InterruptedException {

//archives the artifacts

}

public static final class DescriptorImpl extends BuildStepDescriptor<Builder> {

// To load the data in the global configurations

// On-fly validations for the fields in the configurations. (Made mandatory fields)

// Save the global configurations

}

}

**ScannerExecuter.java**

public class ScannerExecuter {

// Main execution of Docker command for the scanning of image

**public** **static** **int** execute(AbstractBuild build,

Launcher launcher,

BuildListener listener,

String artifactName,

String localAnalyzerPath,

String apiURL,

String localImage){

// Method used for command execution

Callable<String, IOException> task = new Callable<String, IOException>() {

//For getting the IP of the slave on which job is getting executed

}

ArgumentListBuilder args = new ArgumentListBuilder();  
 args.add(localAnalyzerPath, "-endpoint", apiURL, "-my-address", localIP, localImage); // to sum up the command for scanning the image  
 Launcher.ProcStarter ps = launcher.launch();  
 int exitCode = ps.join(); // To execute the image analysis tool

// To copy the output to the workspace

}

}

**ClairScannerAction.java**

public class ClairScannerAction implements Action {

// Mandatory methods required are getIconFileName, getDisplayName, getUrlName  
 public String getIconFileName() {

// For getting the icon of Clair beside the link for the scanning results of each image

}

public String getDisplayName() {

// For displaying the name of the scanning results Ex : “Clair Docker Scanner tomcat”  
 // Display name with image name as suffix

}

public String getUrlName() {

// For the url link for the scanning results of the image

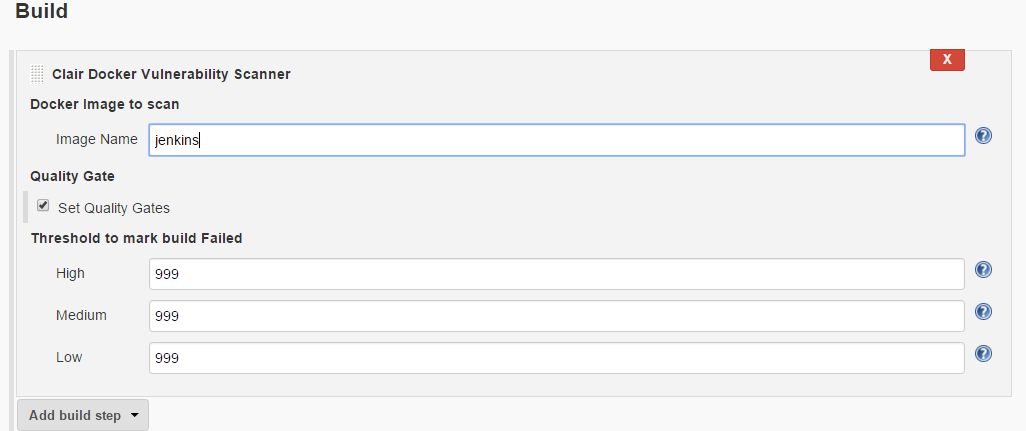
}

}

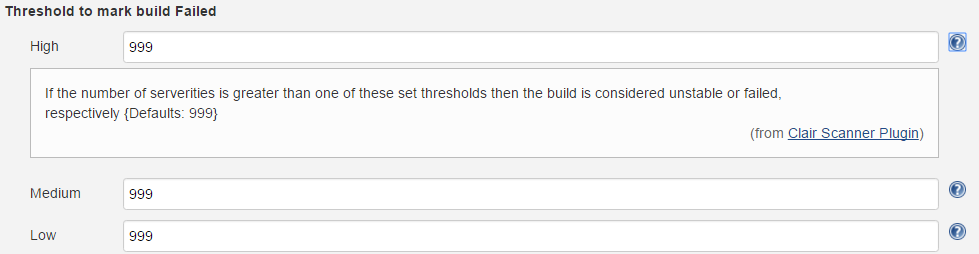
Job Level Configurations:

Pass the image name to scan for vulnerabilities.

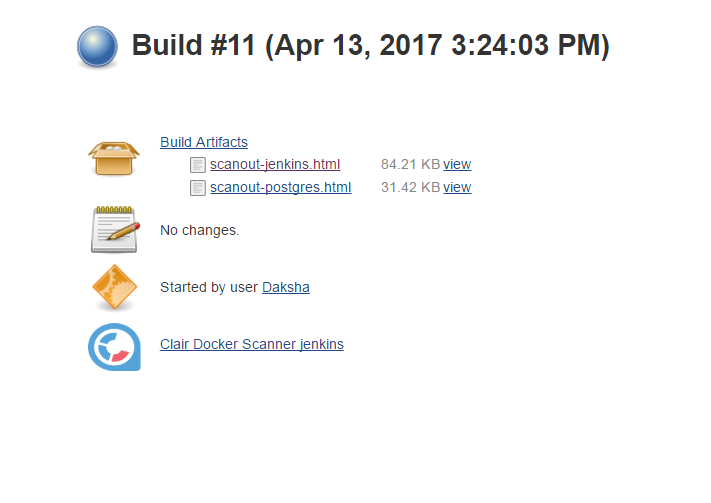
If need to check the quality gate pass the threshold value of vulnerabilities count.

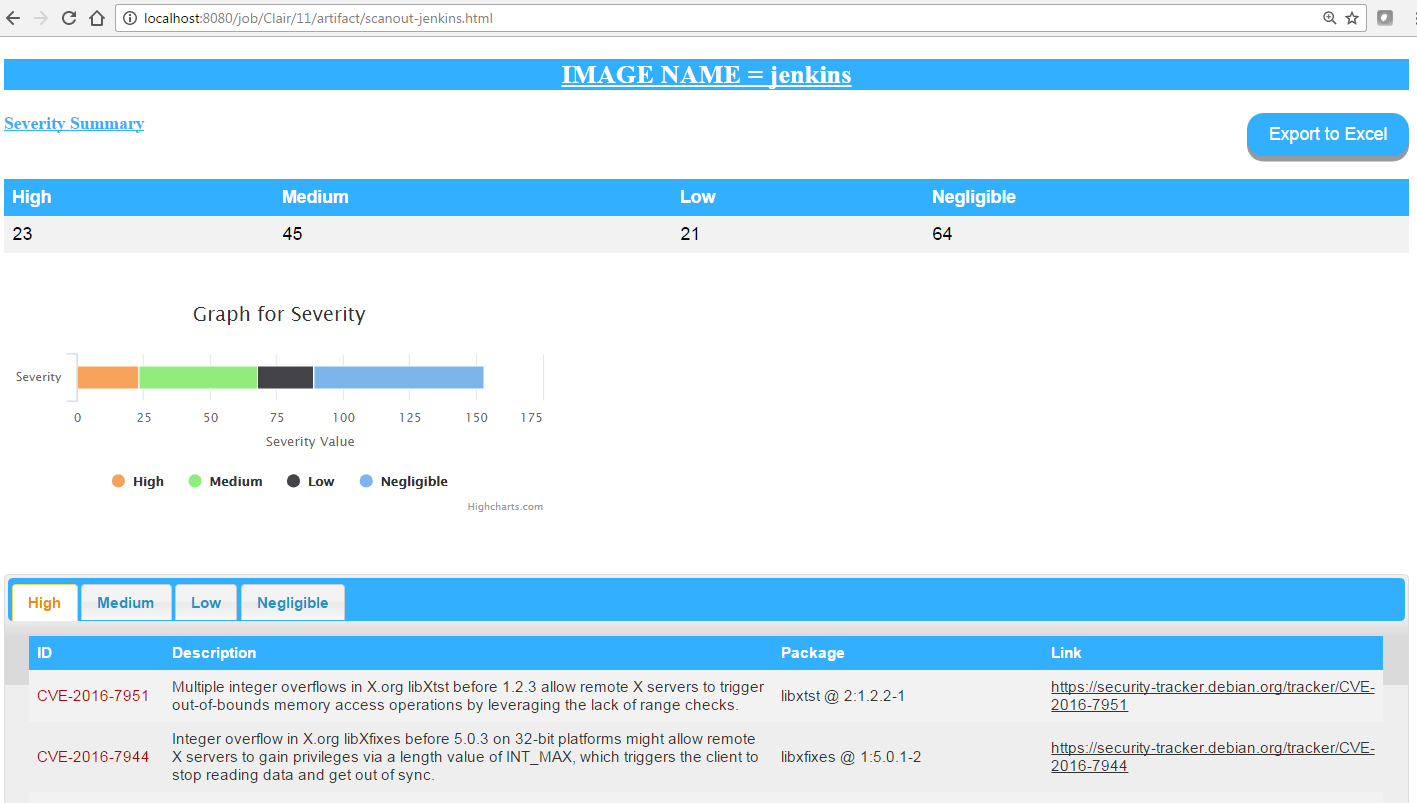


The default value will be 999.



After building the job will get the output as HTML page.





It can be exported to Excel File by clicking on Export to excel button present on page.

