# 

# Building a CI/CD Pipeline

# Table Contents

* Overview
* Continuous Integration Set Up
* Creating and scheduling Jobs

1. Git Configuration
2. Maven Configuration
3. Sonar Configuration
4. Nexus Configuration
5. Tomcat Job Configuration
6. Splunk Job Configuration

* Creating Pipeline

# Overview

This tutorial walks through the setup of a CI/CD pipeline, with the following key components:

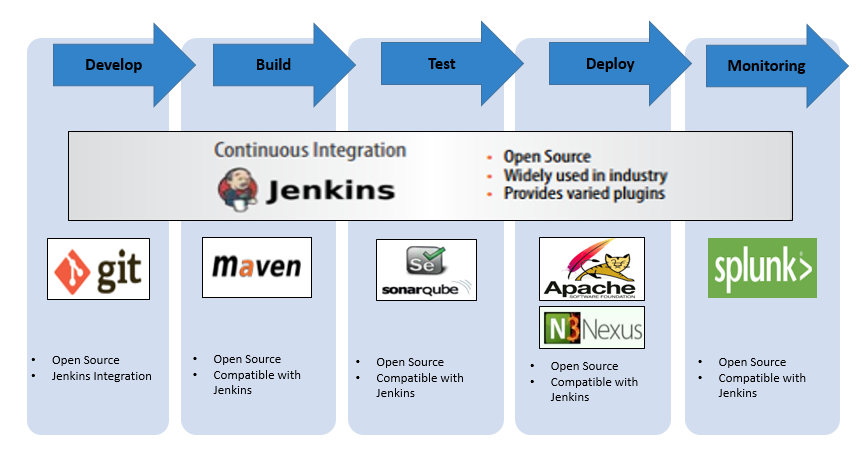
* Jenkins
* GitHub
* Maven
* Sonarqube
* Nexus
* Tomcat
* Splunk

Once configured, Pipeline may be used to automate GitHub project and Jenkins build project creation. While Pipeline only works with GitHub and Jenkins at this time, is written in a way that it is easily pluggable, however, some refactoring of the argument parsing will be necessary as additional plugins are incorporated.

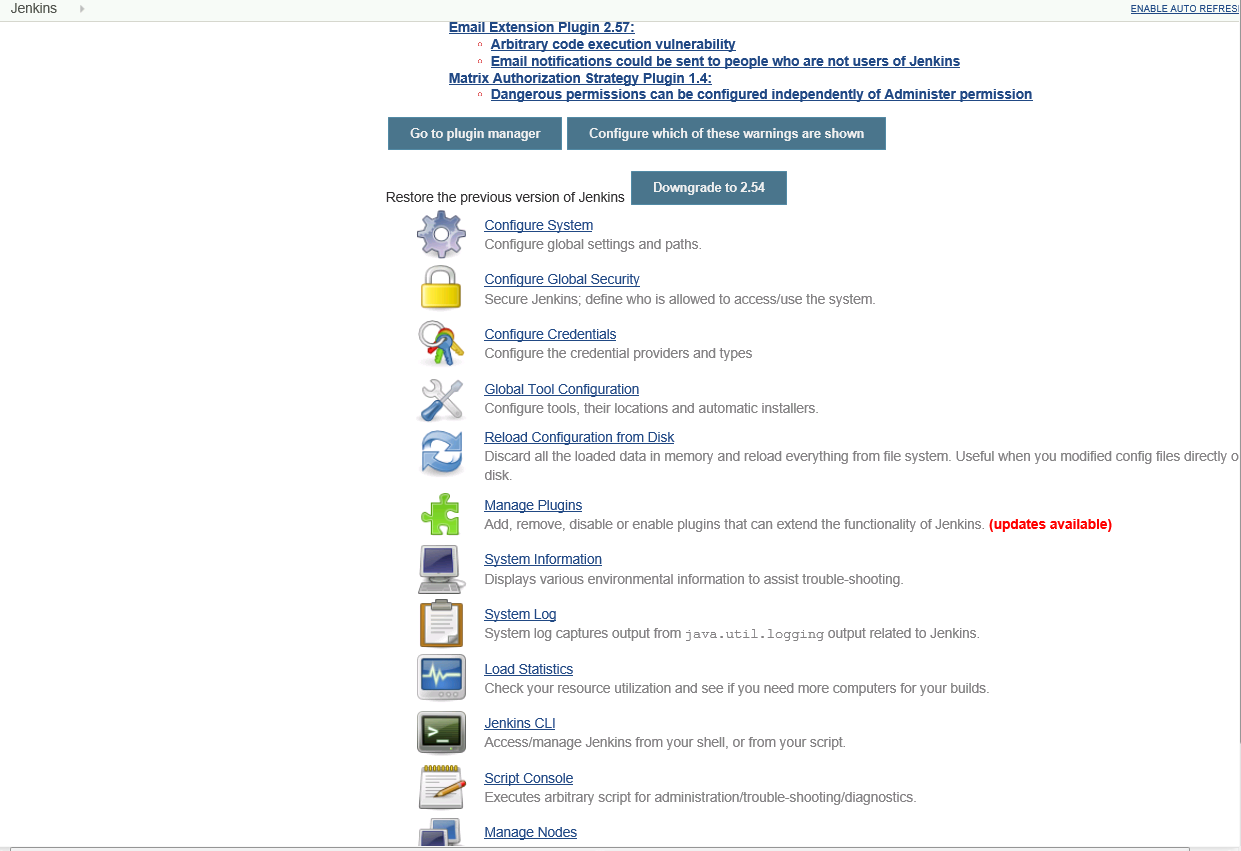
# Continuous Integration Set Up

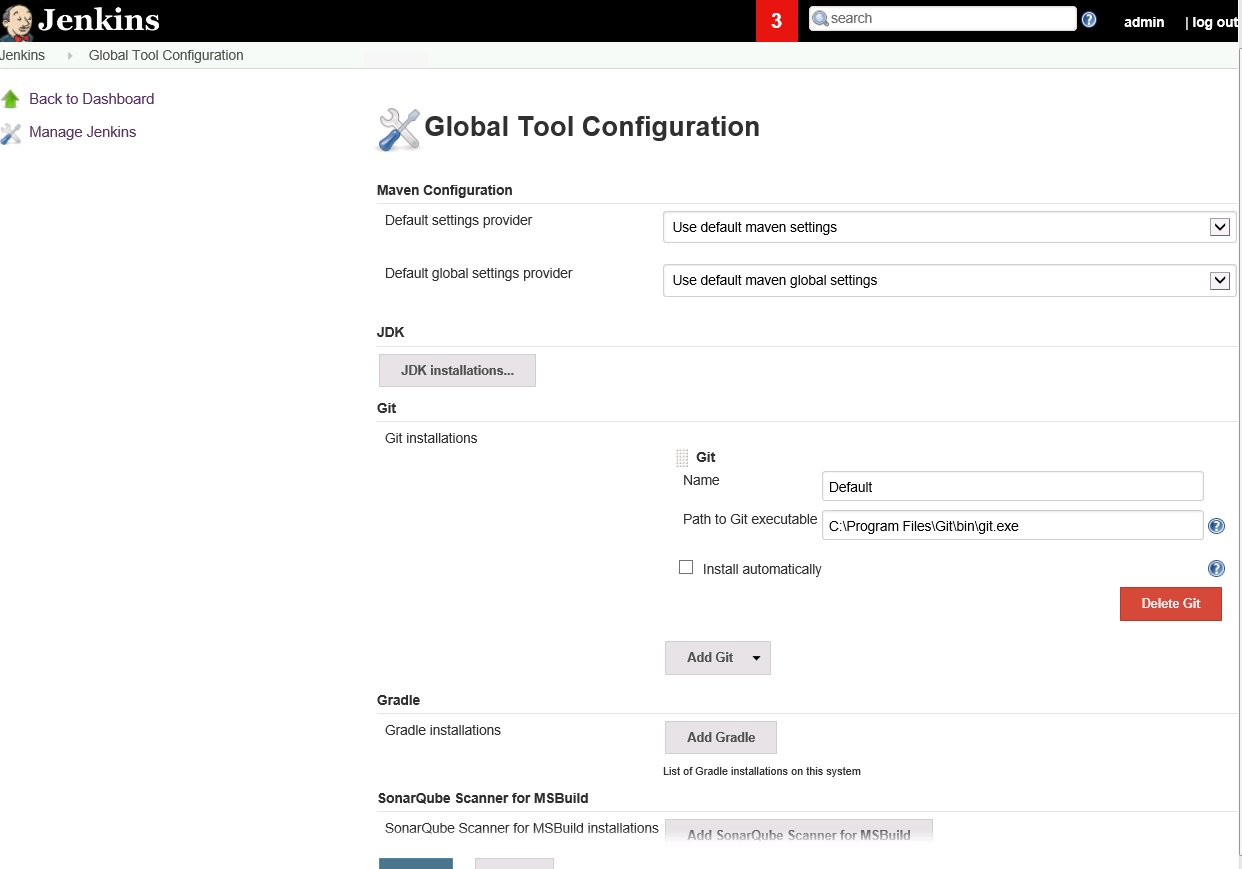
[Continuous integration](http://techbeacon.com/continuous-integration-answer-life-universe-everything) (CI) servers coordinate a wide variety of activities, such as checking out and building new versions of code, running tests, and deploying software. These integrations are handled by automated build and testing systems, which means that your developers are alerted to issues such as code clashes or broken unit tests as early as possible.

The most popular CI tool today is [Jenkins](https://jenkins.io/). Jenkins is an open-source, server-based system that provides a straightforward platform for automating build, testing, integration and supports a wide selection of version control systems.

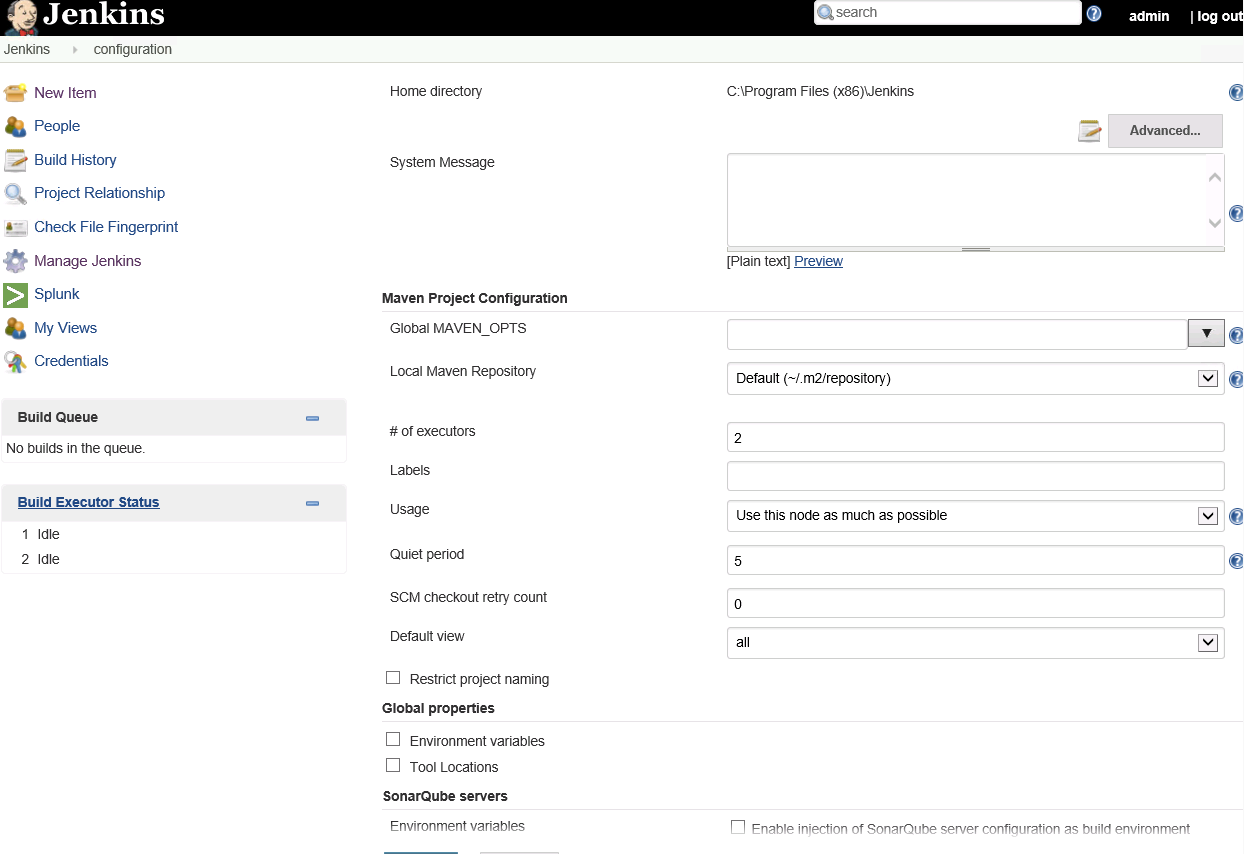


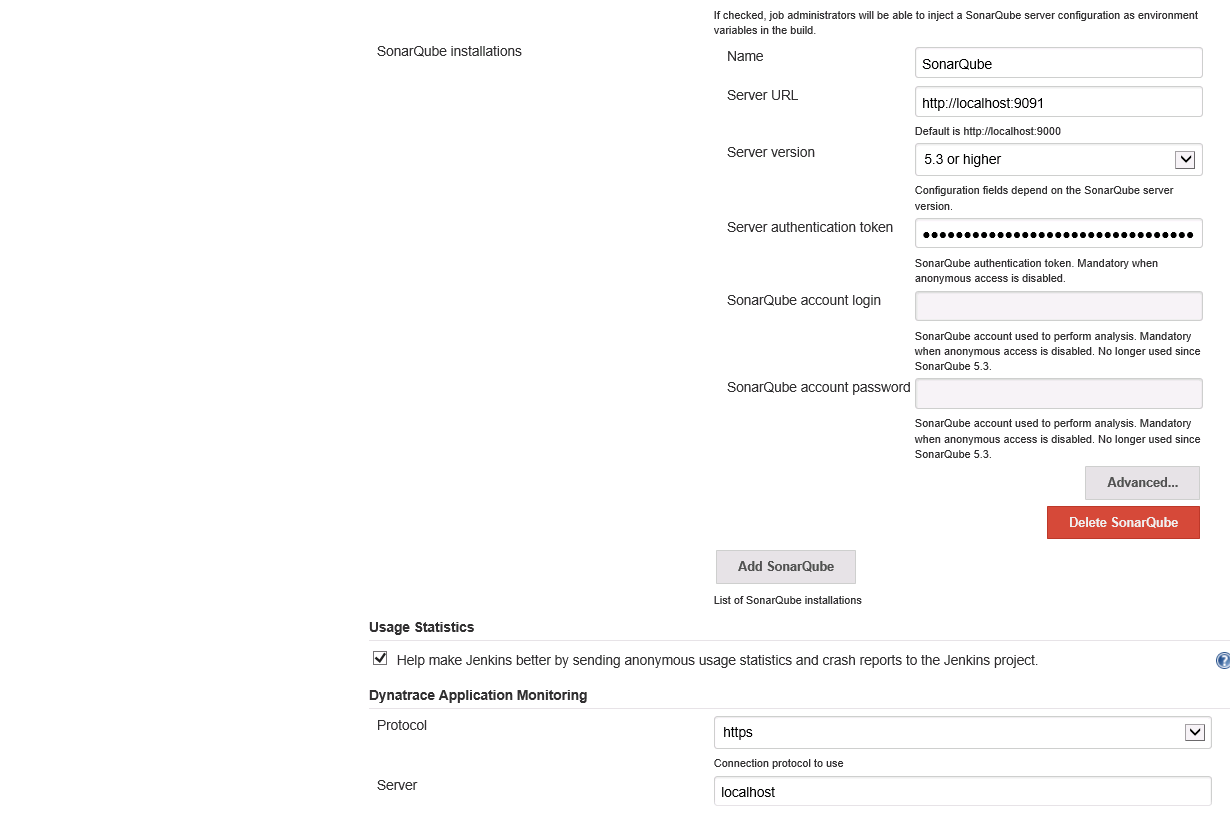
I would like to share some practical guidance for building and deploying applications using Jenkins, including plugins, which support and extend Jenkins capabilities, making them [a vital tool](http://techbeacon.com/top-agile-tools-keep-software-engineers-productive) when building your integration and delivery processes.

* Go to Jenkins Portal. https://localhost:portnumber
* Login to Jenkins with your Credentials.
* Go to Manage Jenkins
* Select “Global Tool Configuration”
* 
* Update the Configurations of GIT, Maven, JAVA, Sonar etc...
* Click Apply and save.



* Go back to Manage Jenkins.
* Select Configuration System
* Update the configuration details of tools like sonar, splunk, Nexus etc...
* Click Apply and save.





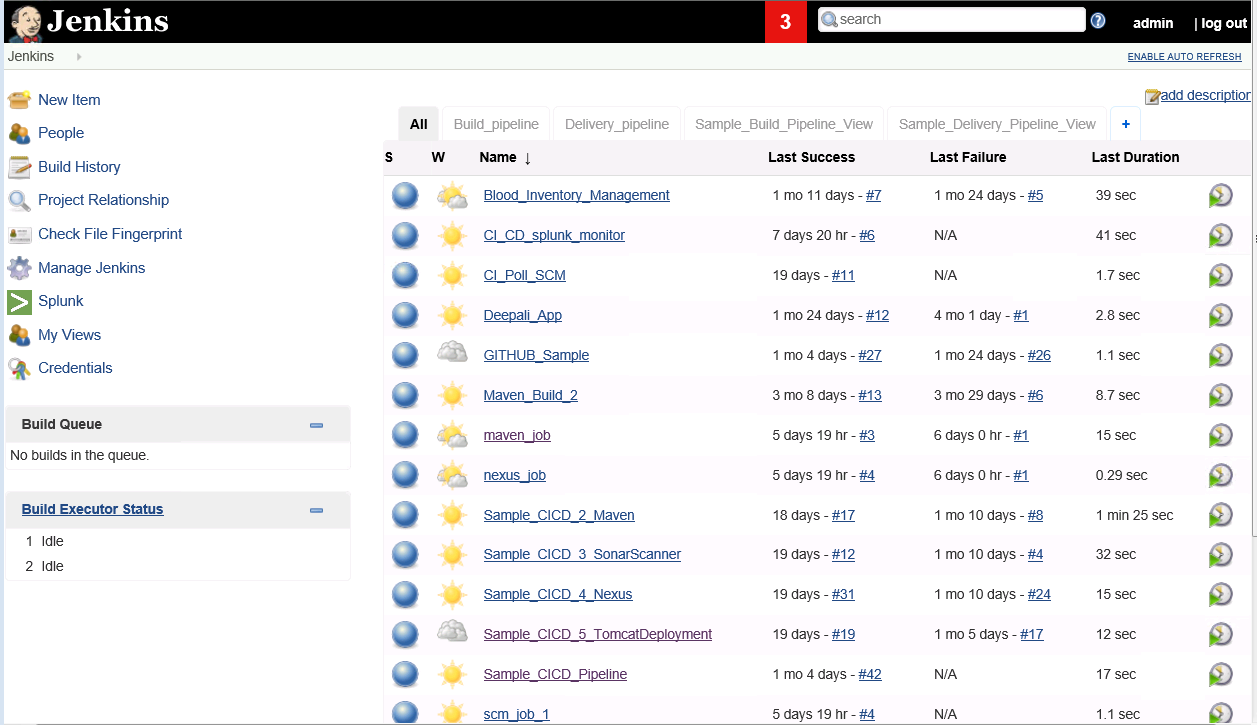
## Creating and scheduling jobs

Jobs are the runnable tasks that are controlled and monitored by Jenkins. Examples of jobs include compiling source code, running tests, provisioning a test environment, deploying, archiving, posting build jobs such as reporting, and executing arbitrary scripts.

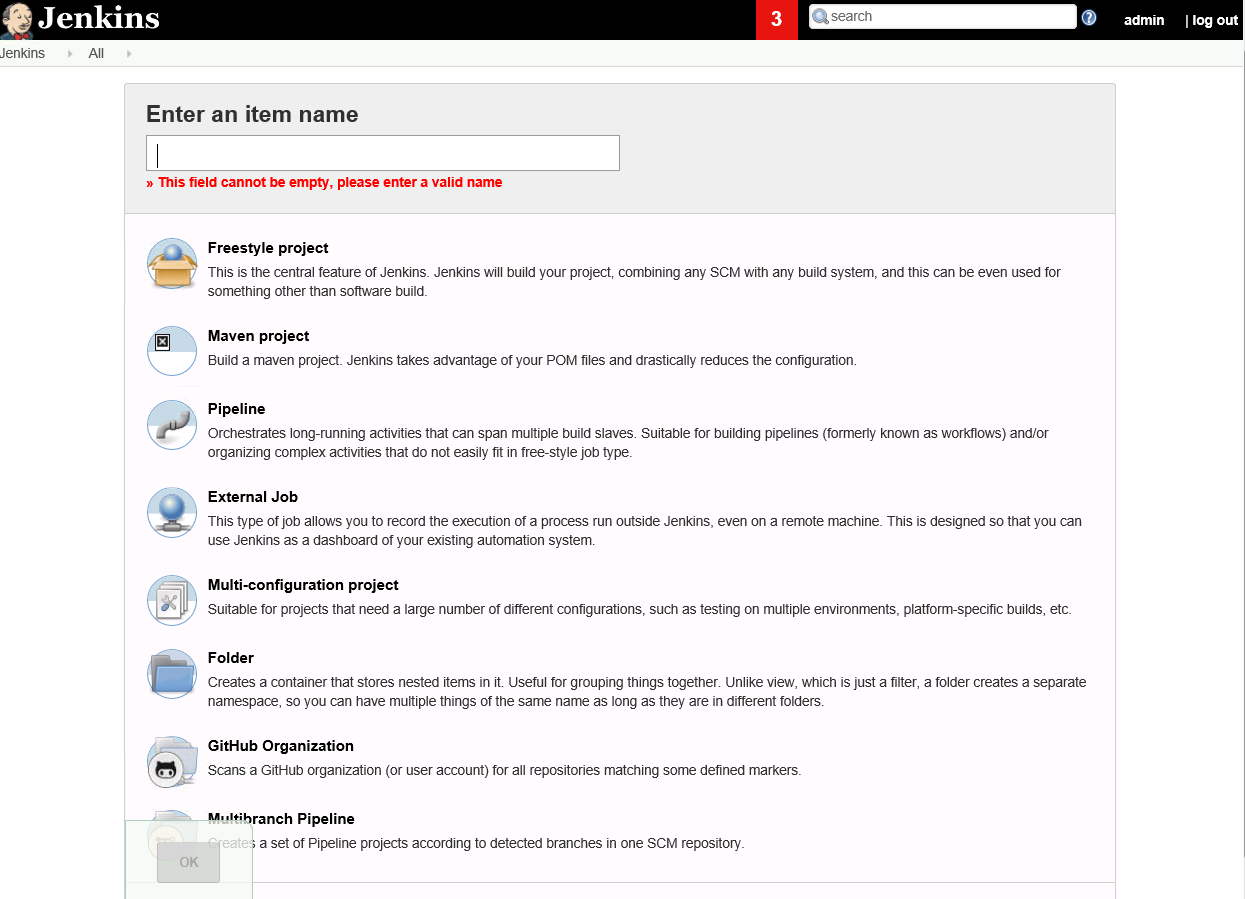
In a few steps, I will create a job that builds a Java project using the [Maven](https://maven.apache.org/) build automation tool. Maven allows developers to automate the process of creating an initial folder structure for a Java application and performs the compilation, testing, and deployment of the final product.

Step-1:

* Go to Jenkins Portal. https://localhost:portnumber
* Login to Jenkins with your Credentials.



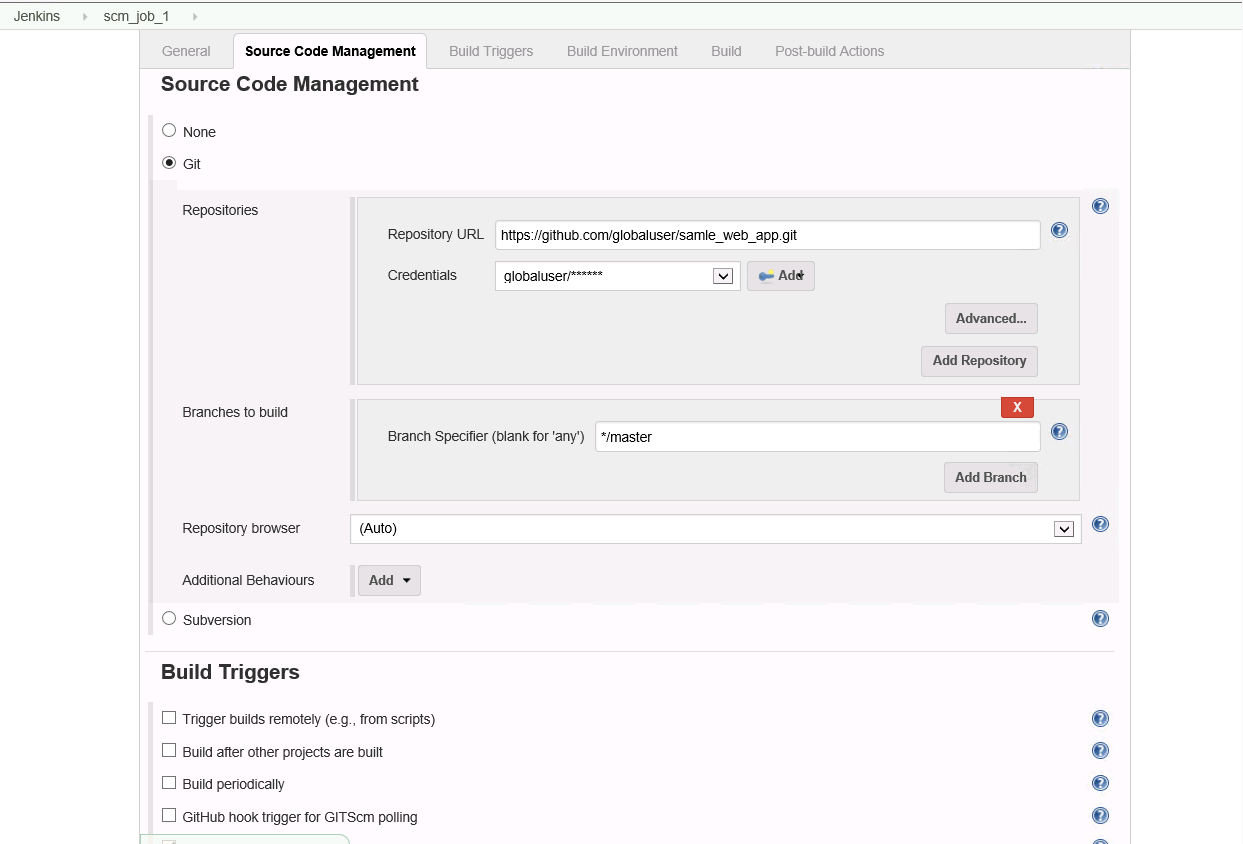
* Go to New Item
* Give your Job name
* Select your job type. Click OK



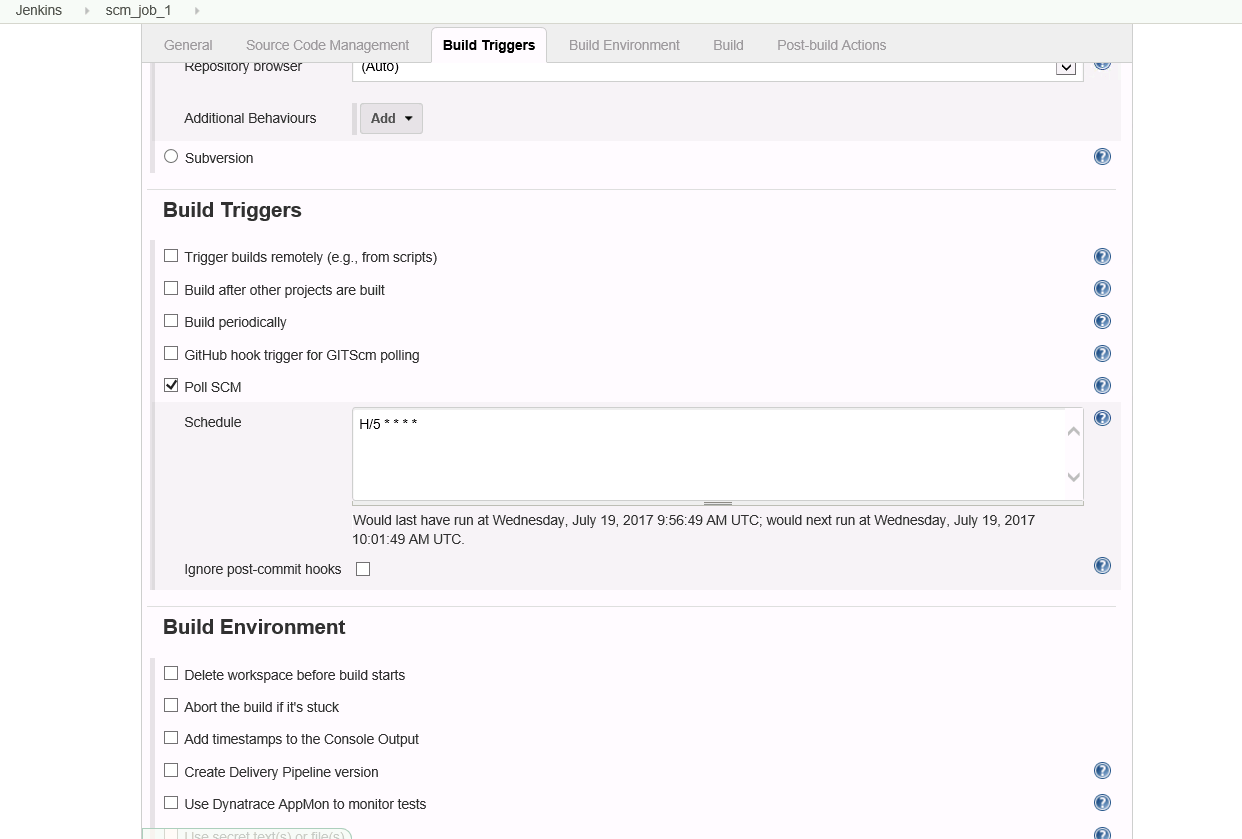
### Git Job configuration

Pull the code from GitHub repository by configuring the job with that git repository URL

Click Apply and SAVE.



In Build triggers, select Poll SCM. Then the job will triggered according to the time given.

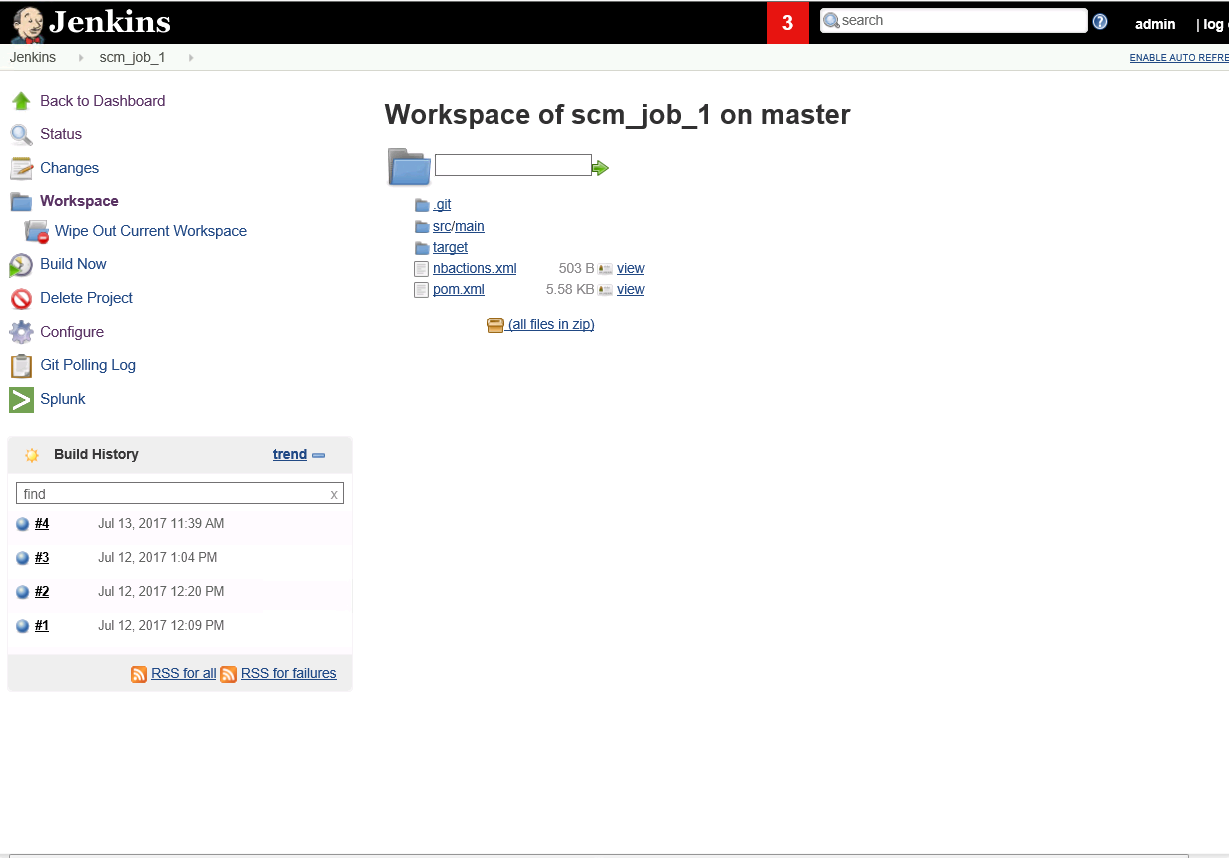


Then Click Build now.

The job will execute.

If the job is successful you will get the data in Job workspace.

The job workspace is looks like as in given below snapshot:-



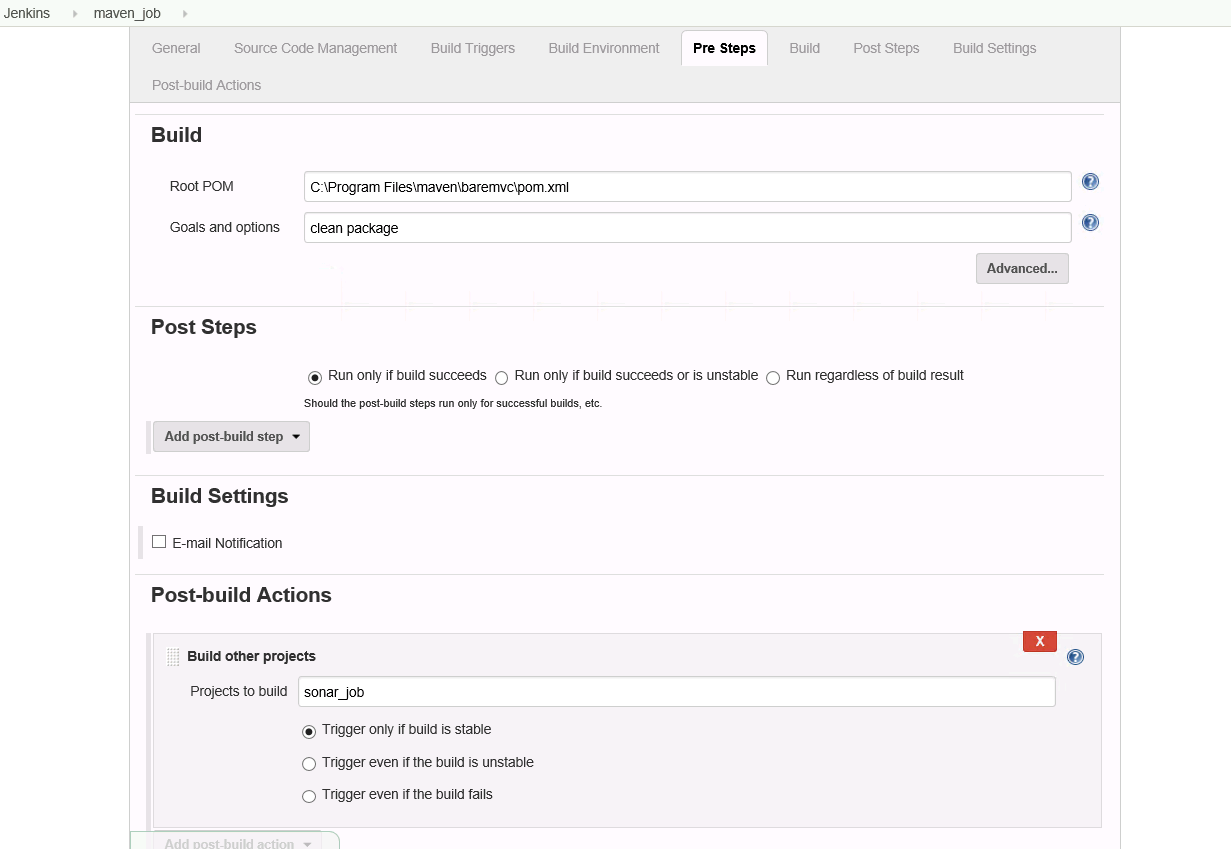
### Configuration for Maven Job:

Maven is using for verification of the code.

Note: When you create your job for Maven , Then you have to create the job select Maven Project

Go to Job configurations

Give your Pom.xml details and goal



### Sonar Job configuration:

Go to Job Configurations

In Build, select Execute sonarqube scanner

Mention the sonar properties as below

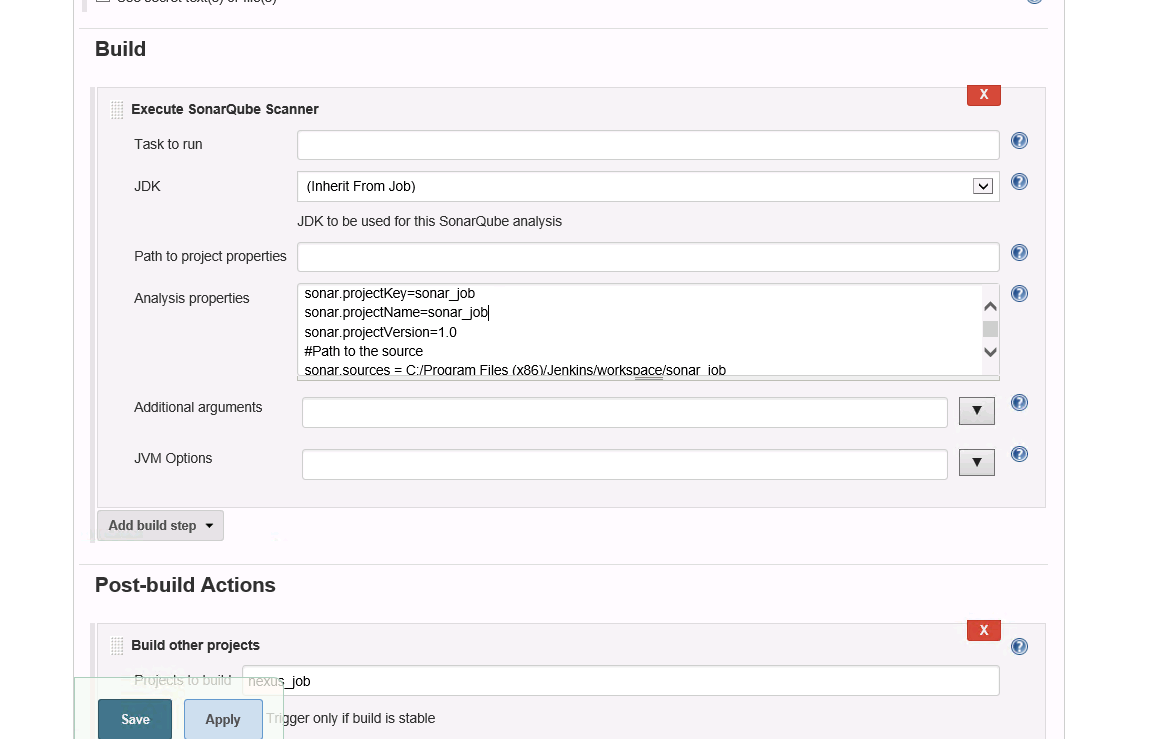
sonar.projectKey=sonar\_job

sonar.projectName=sonar\_job

sonar.projectVersion=1.0

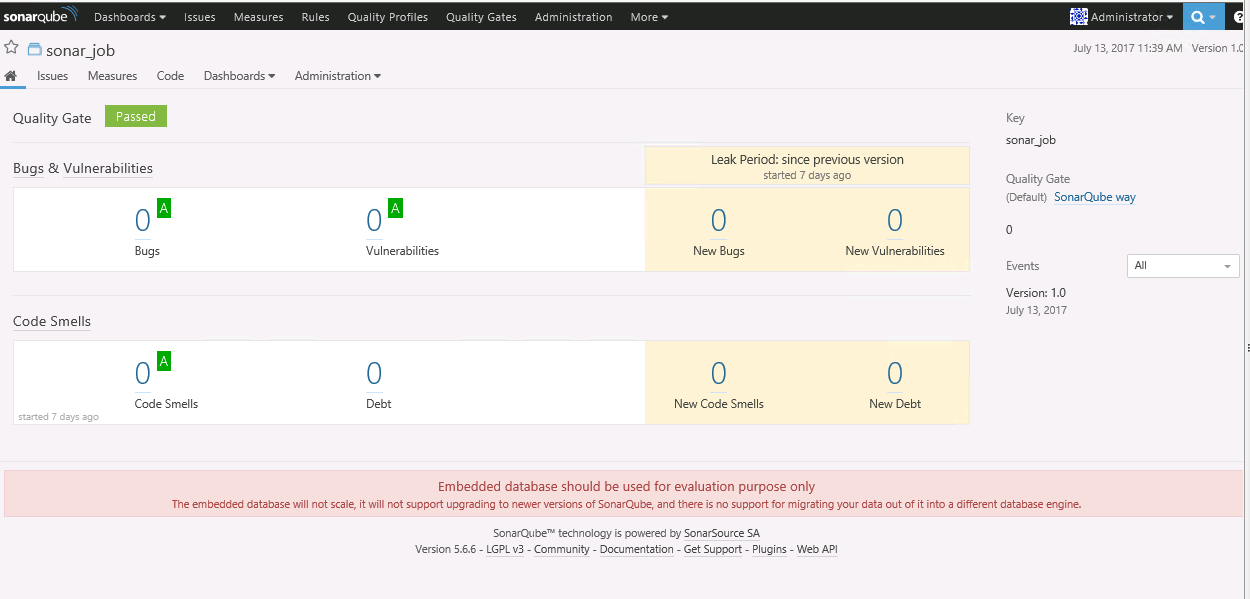
#Path to the source

sonar.sources = C:/Program Files (x86)/Jenkins/workspace/sonar\_job



Click Build now.

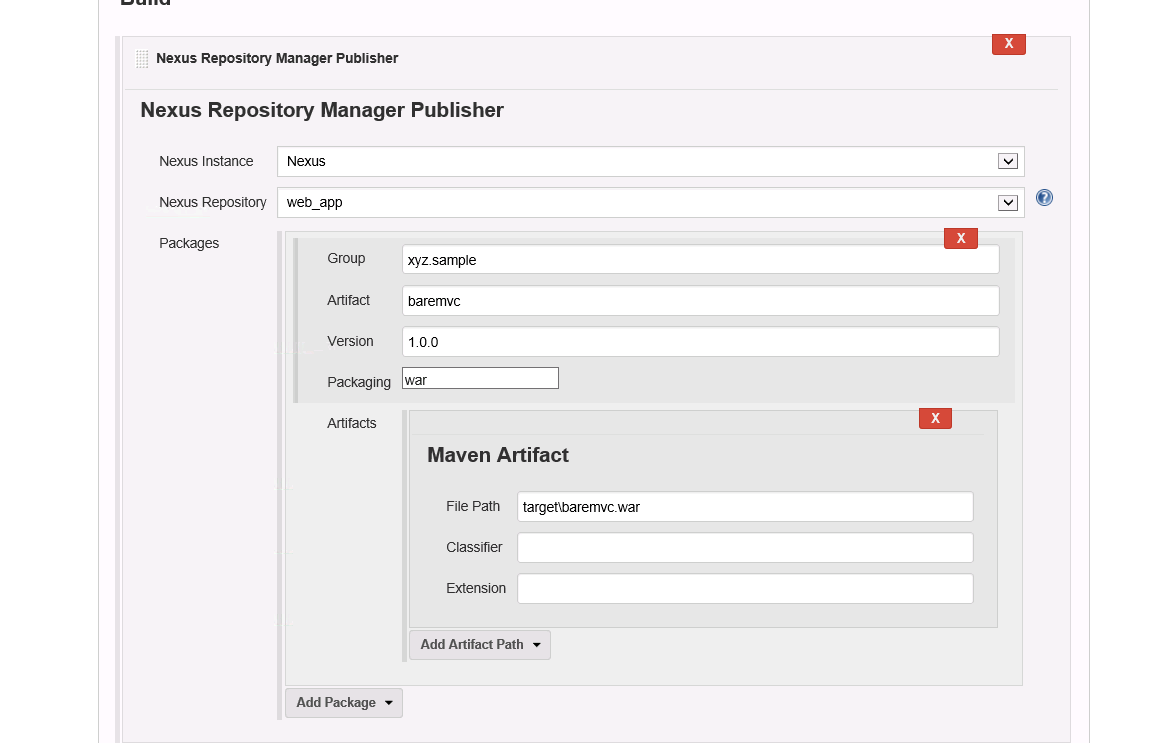
You will see the report in sonar as



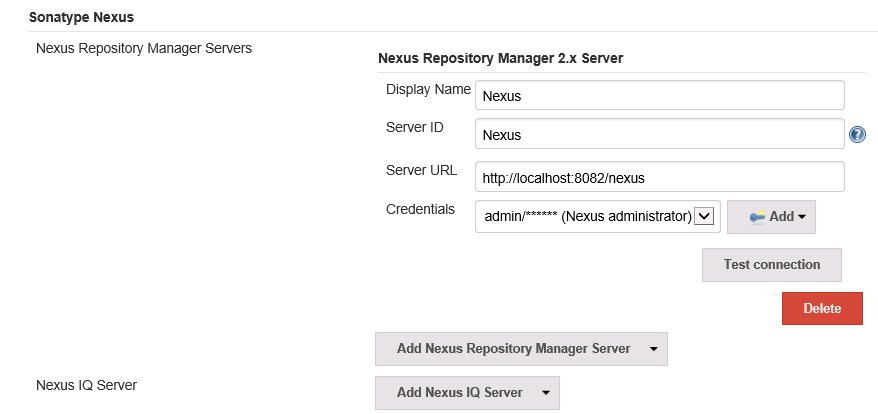
### Nexus Job configuration:

Go to Job configurations

In Build, Select Nexus Repository Manager Publisher

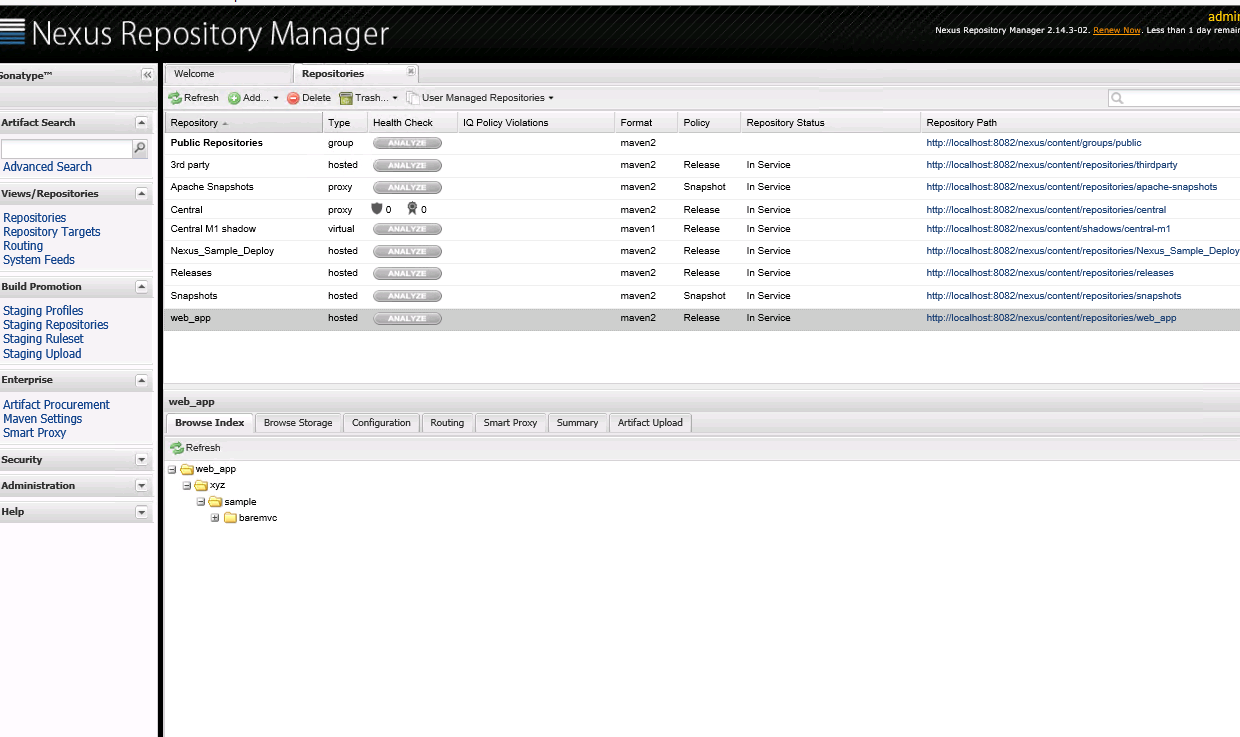


Go to Configuration System and Update with the Nexus Configuration



Click Build now.

The artifact is deployed to Nexus repository. You can see the artifact in Nexus repository.

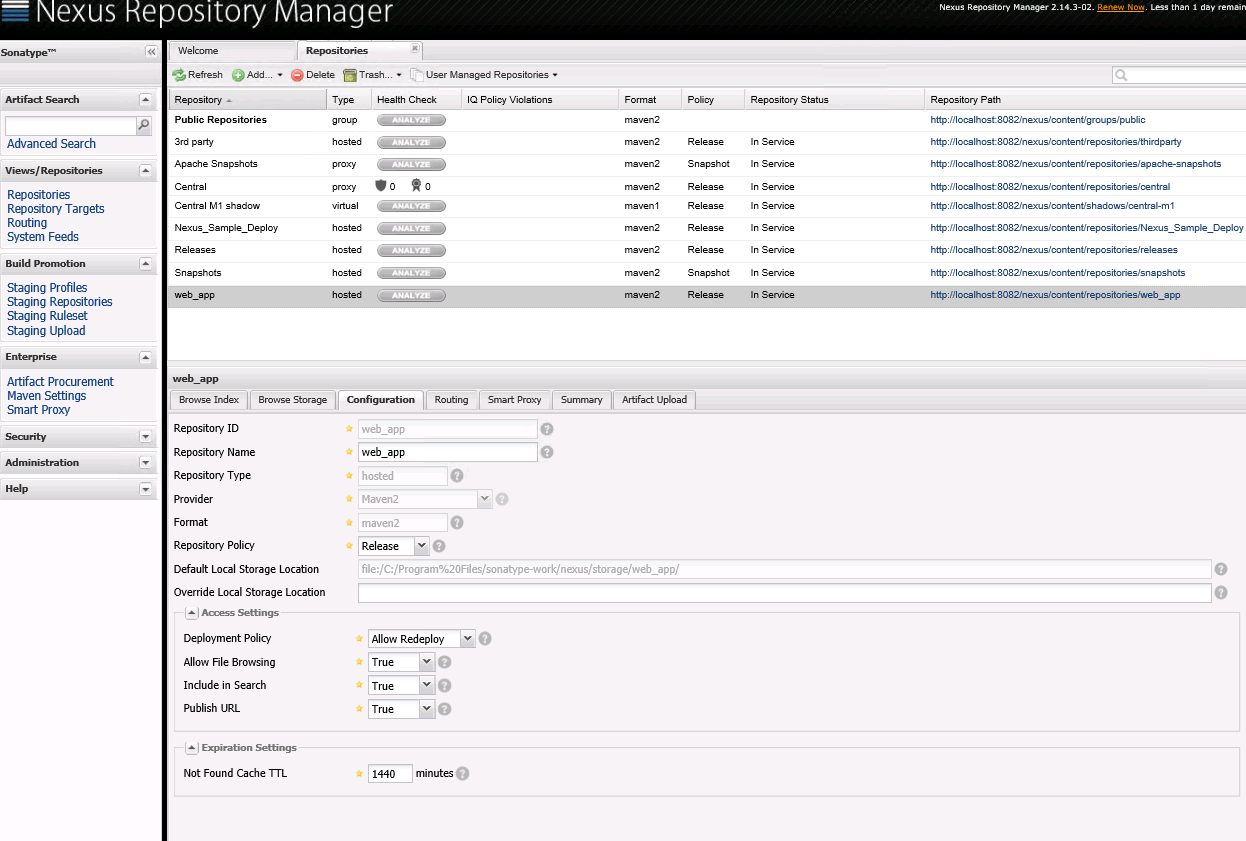


Create Repository in Nexus:

Login to Nexus.

Go to Repositories--🡪 select Add option--🡪 select host/proxy/virtual Repository

Configure Repository as shown



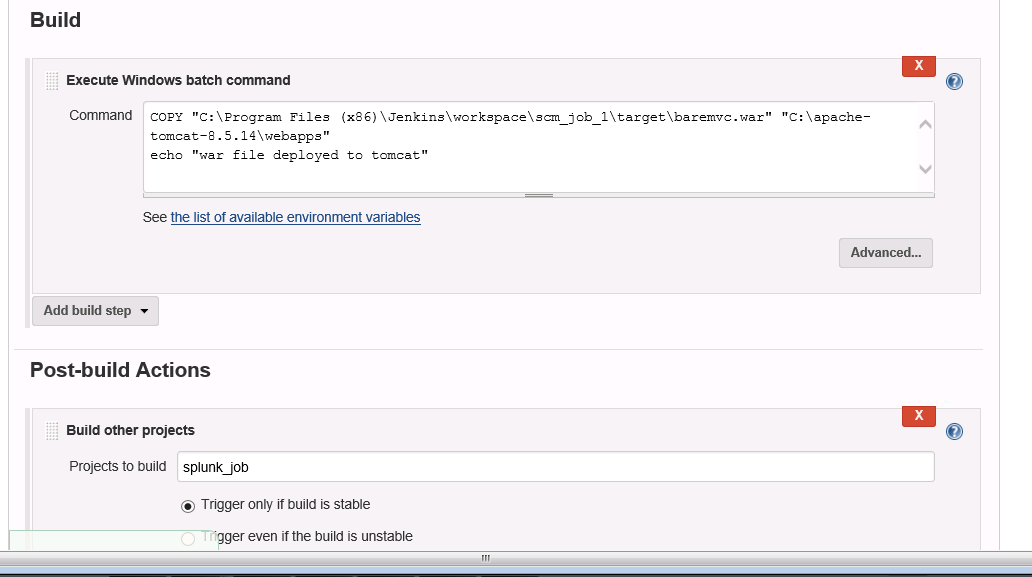
### Tomcat Job configuration:

Go to Job configurations

Option-1:

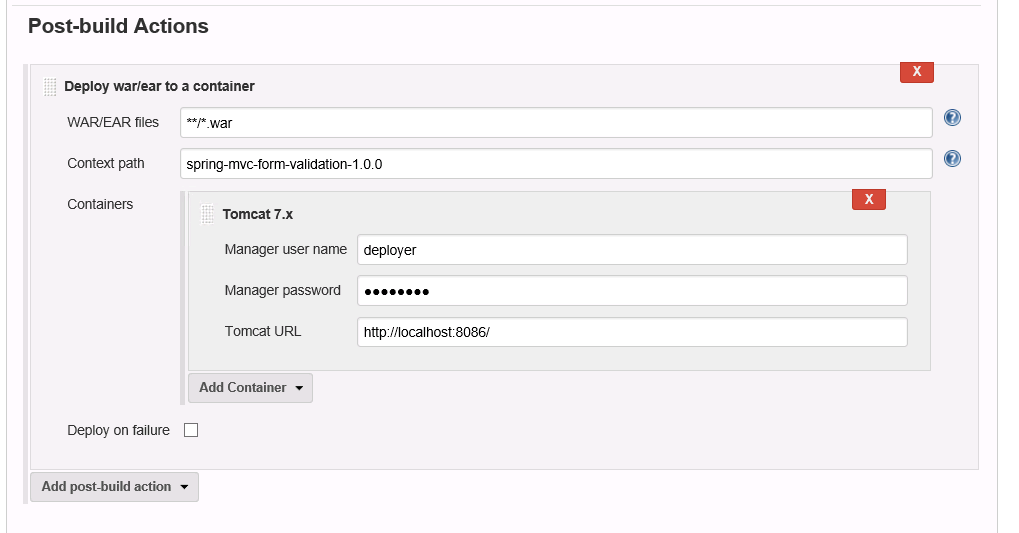
In Build , select Execute Windows Batch command

Give the command to deploy artifacts to Tomcat .

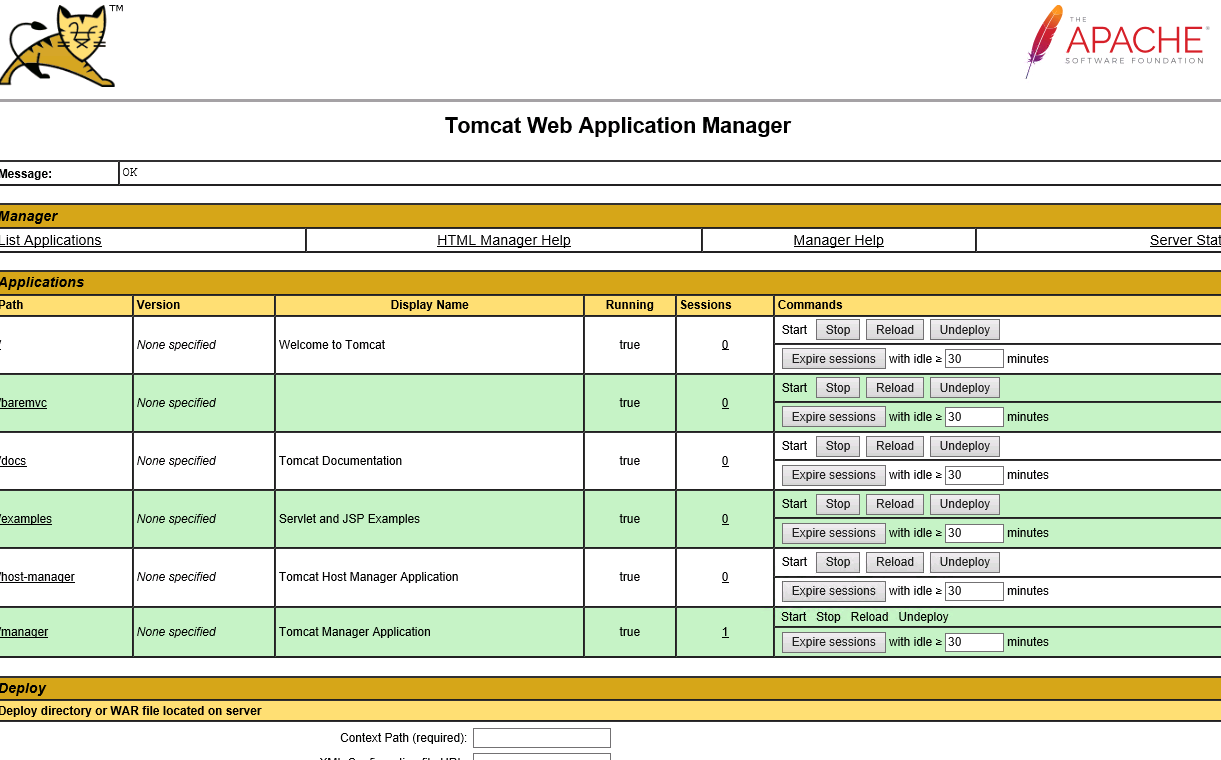


Option-2:

In Post-Build actions, select Deploy to war/ear container



The data deployed to tomcat .You can see that in Tomcat Manager.



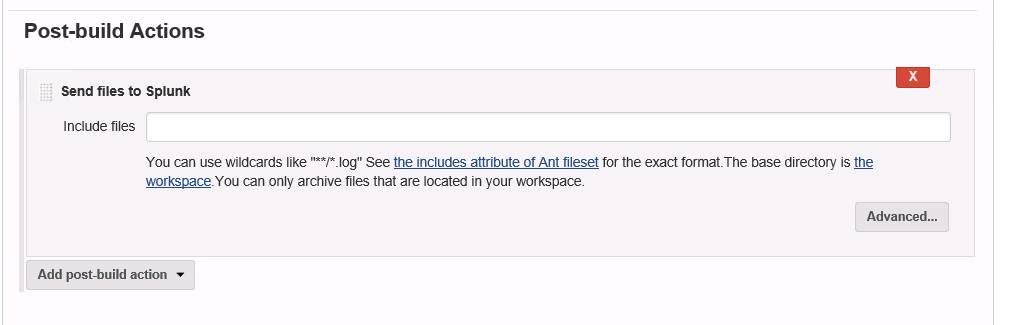
### Splunk Job configuration:

Go to Splunk Portal: <https://localhost:8000>

Create Http Input Token in Splunk.

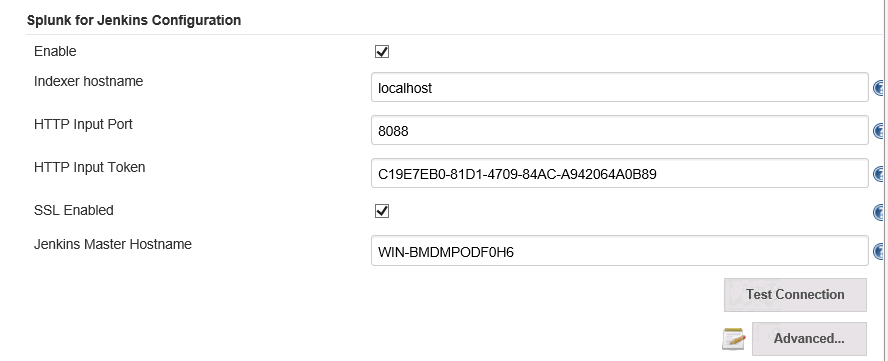
Go to Job configuration:

In Post-build actions, select send files to splunk



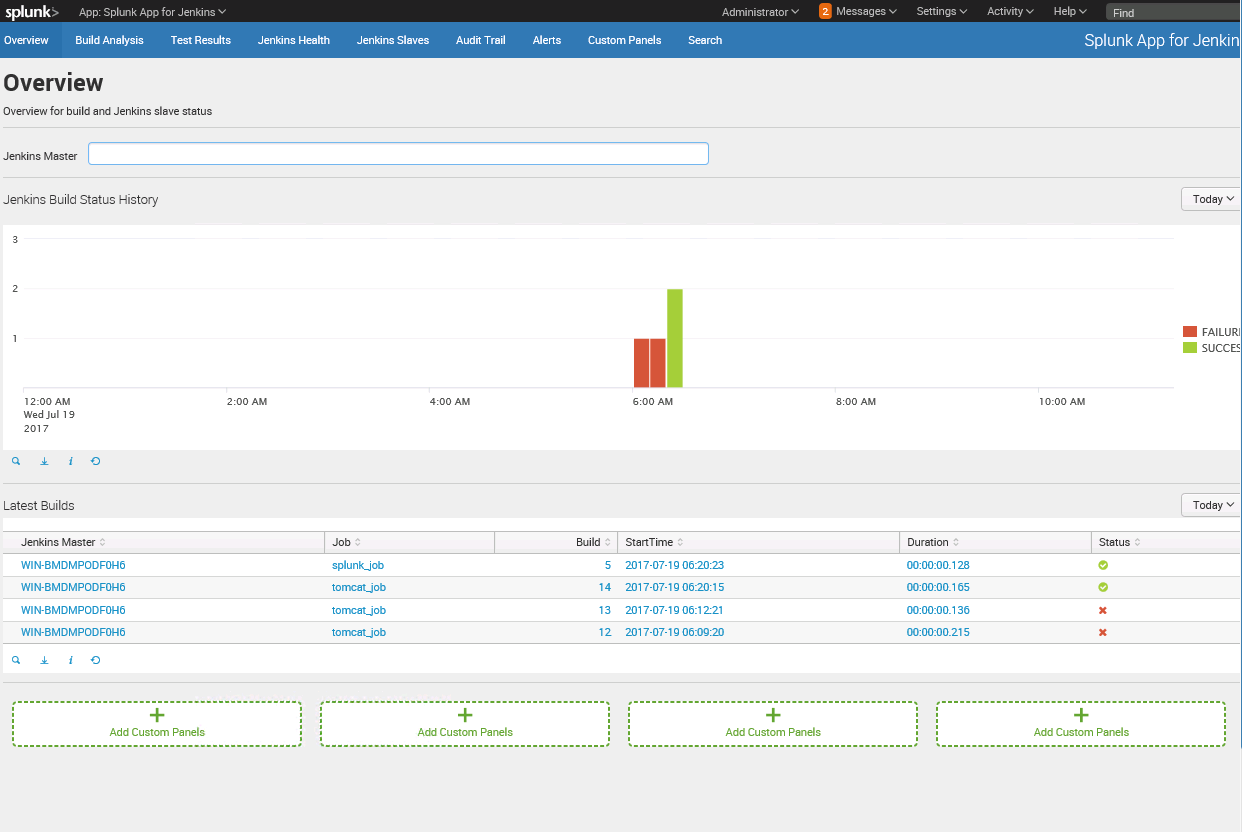
Go to configuration system in Manage Jenkins

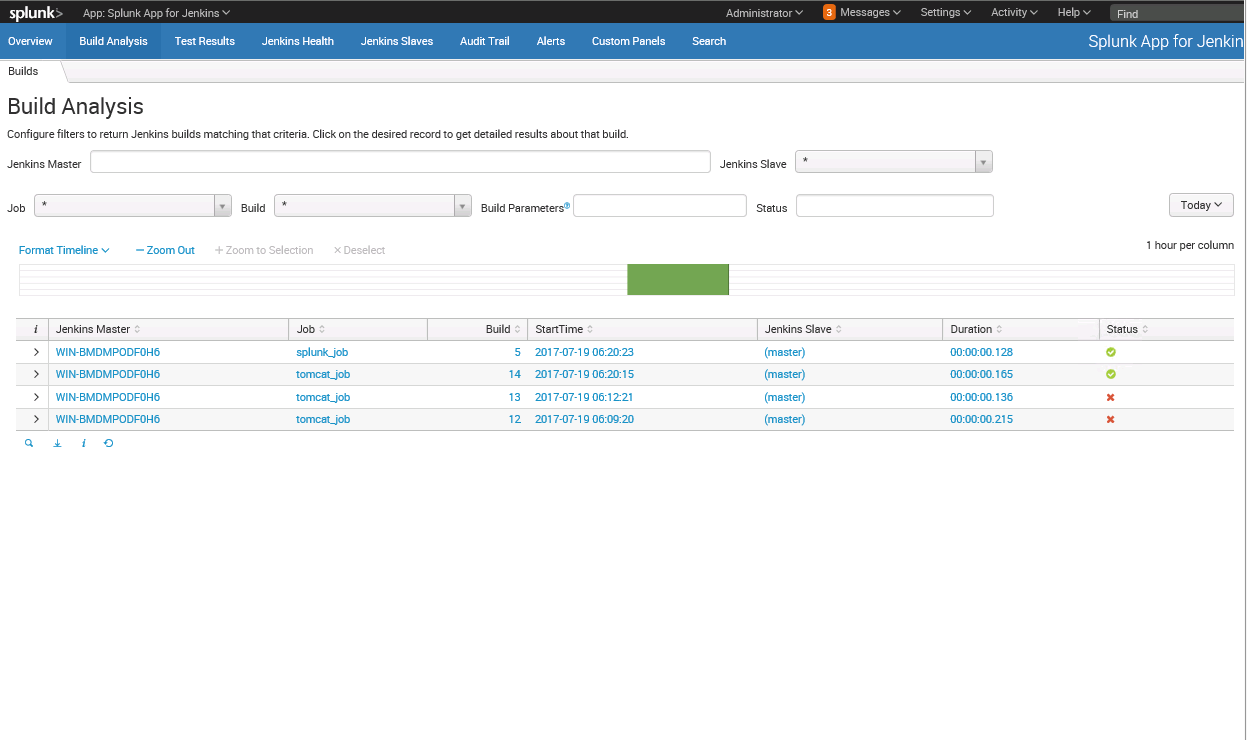
Add the splunk configuration

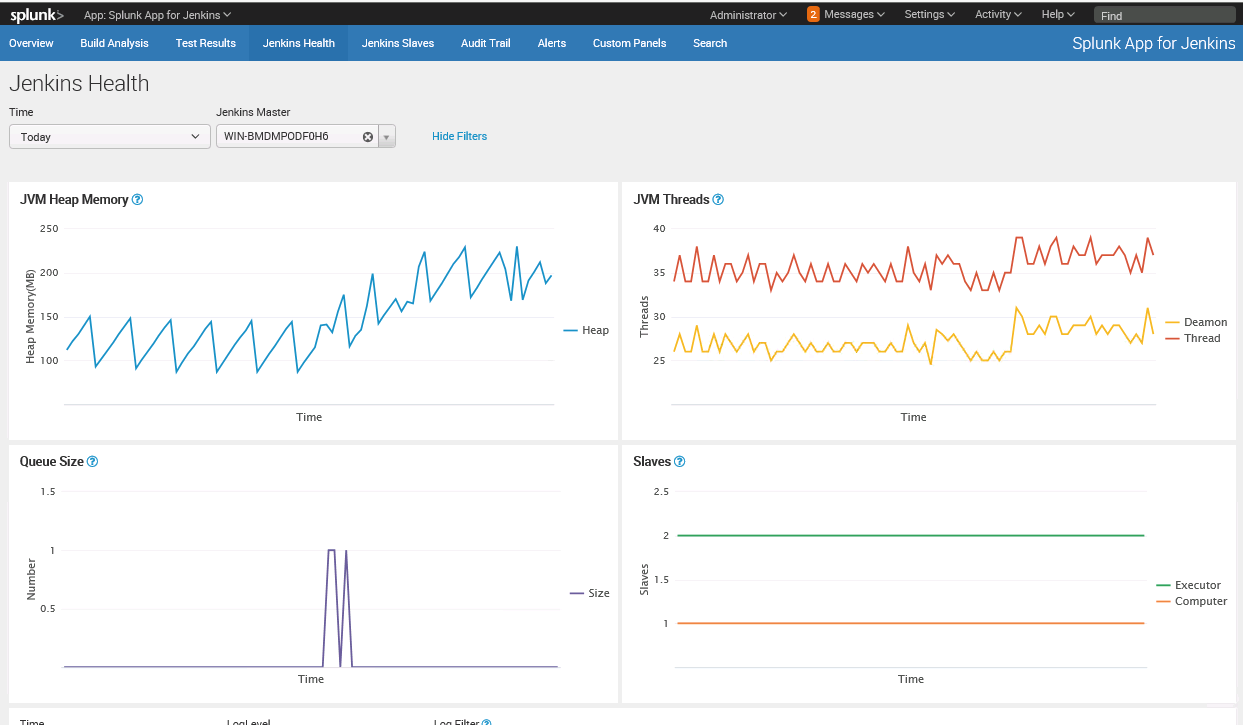


The Job send logs to Splunk.

You can see the Dashbard in Splunk

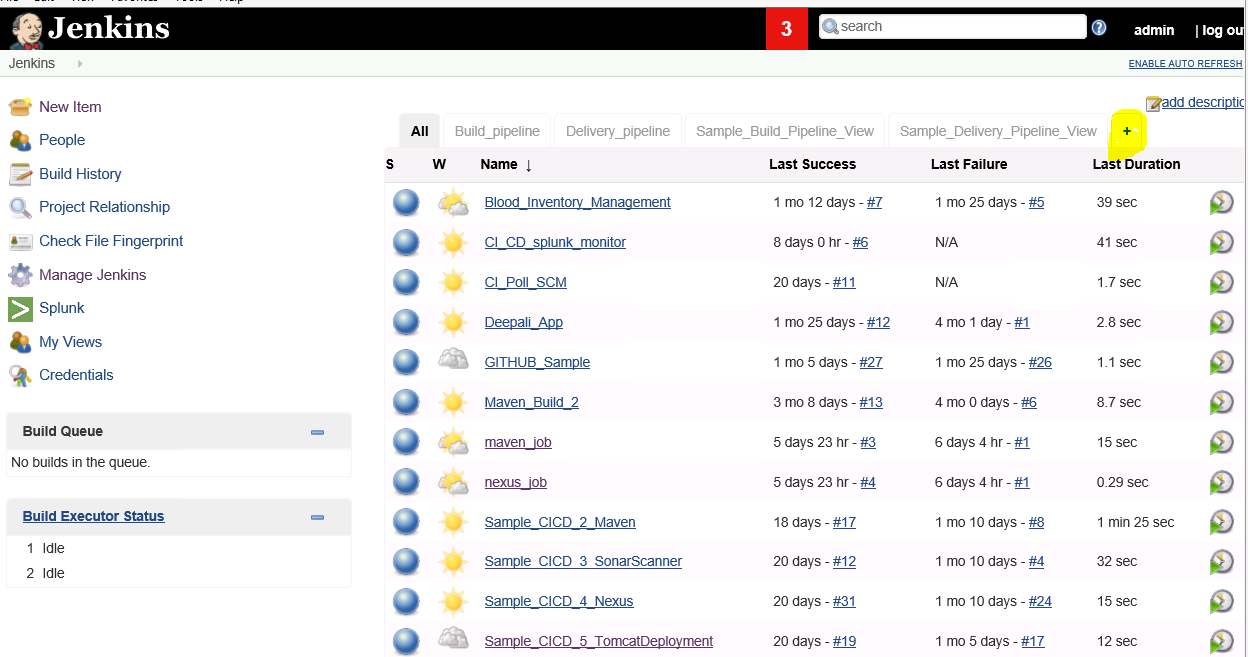






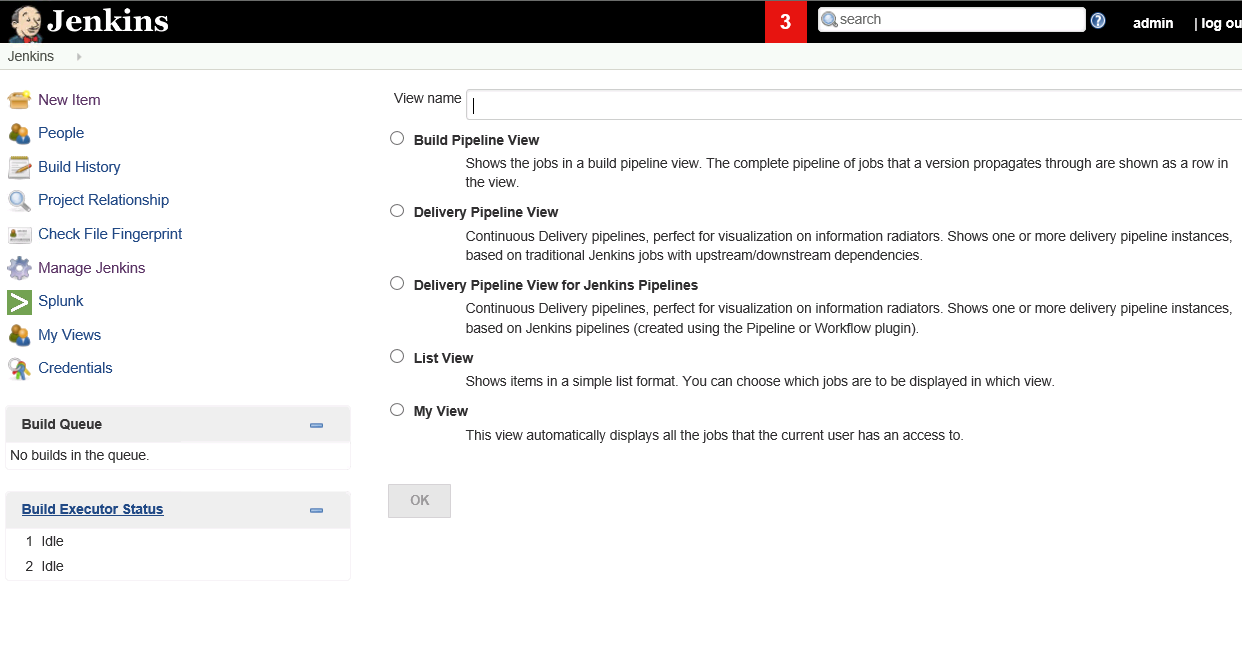
## Creating Pipeline:

Go to Jenkins Home Page and Click on “+” button.



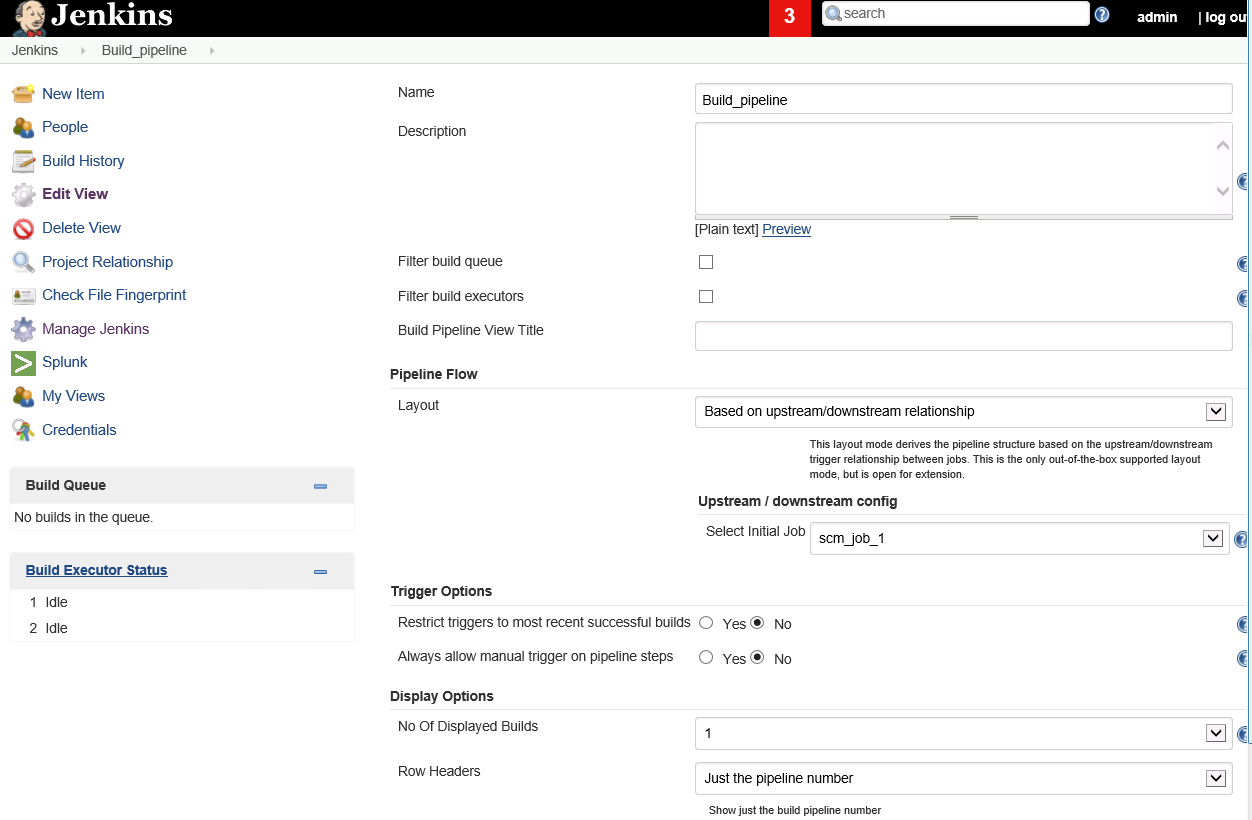
Give your Pipeline name

Select Build Pipeline view

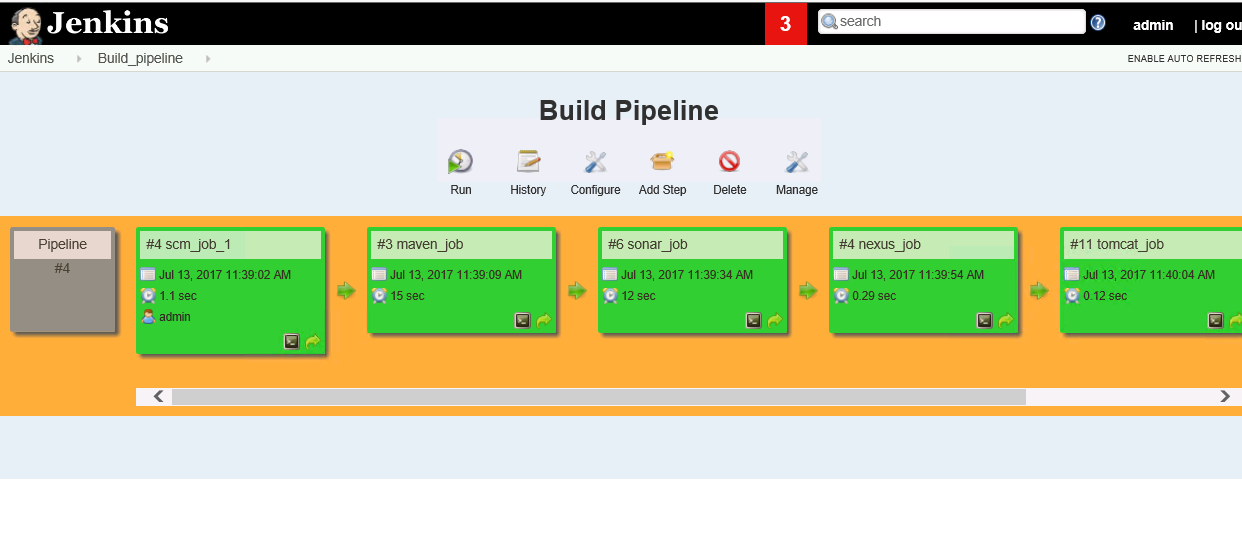


Go to Configurations of Build Pipeline view

Select your first Job in “Pipeline Flow”



The pipeline looks like:



If you select View name is Delivery Pipeline

The Pipeline looks like:

