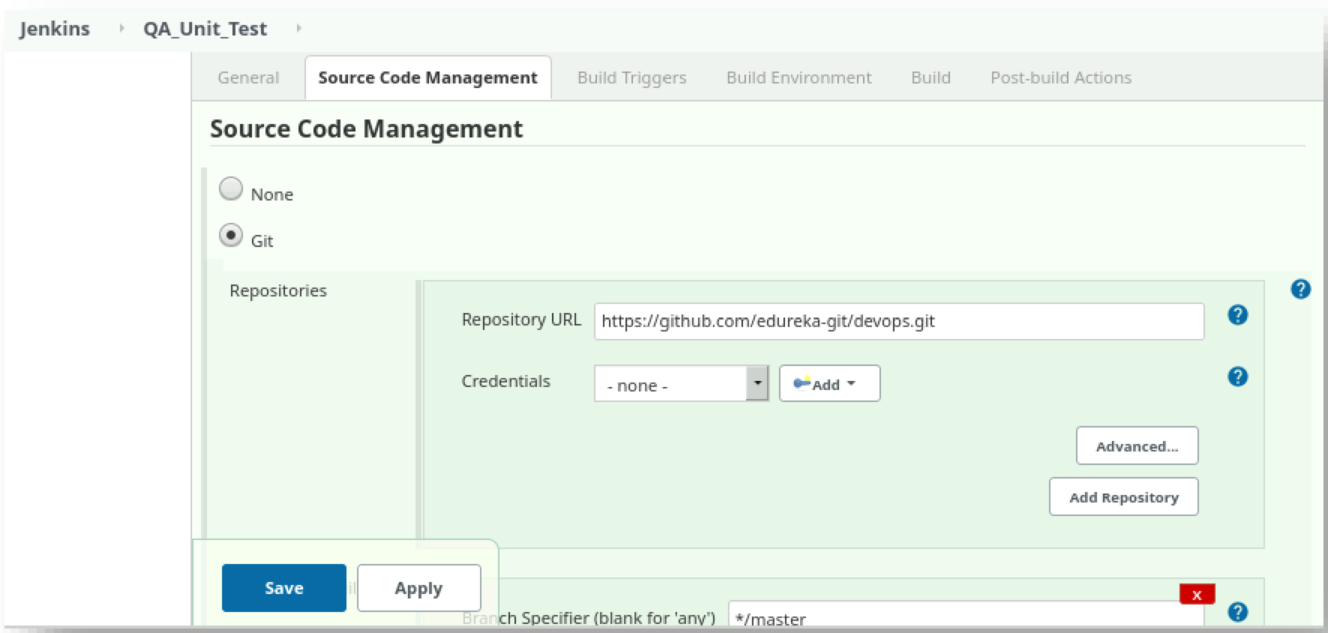
Module 4: Continuous Integration With Jenkins

Assignment - 1 Solution

Assignment Solution

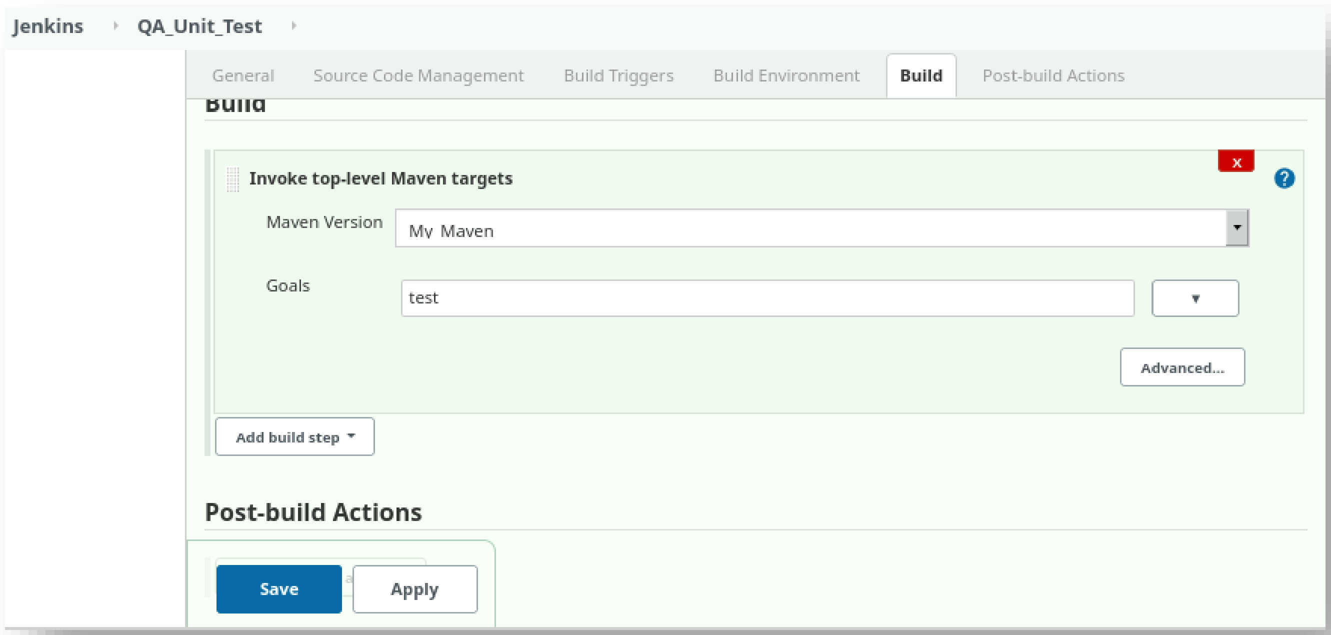
- Add a task QA_TEST that is driven from job DEVELOPER_CODE_REVIEW and performs unit testing

  Step 1: In your VM's browser, navigate to the Jenkins Home page
  Step 2: Click on the "New Item"
  Step 3: Give Item Name as QA_UNIT_TEST and select freestyle project.
  Click OK
  You will be redirected to the configuration Page of QA_UNIT_TEST.
  Step 4: In Source code management, select the Git radio button and add your repository link in the **Repository URL** field.



Repository URL: https://github.com/edureka-git/devops.git
Credentials: - none -
Branch Specifier (blank for 'any'): */master

Step 5: In Build, enter “test” in the **Goals and options** field.



Maven Version: My Maven

Goals: test

Click Save.

Step 6: Navigate to the **Build trigger** tab in the Configuration option of QA_UNIT_TEST.

Step 7: Check the “**build after other projects are built**” checkbox

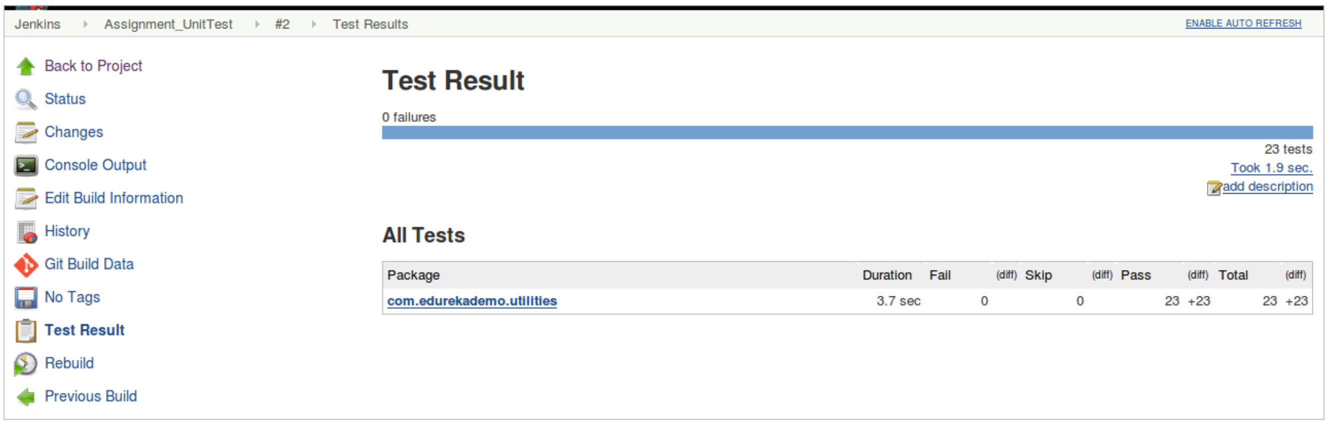
Step 8: Select the “DEVELOPER_CODE_REVIEW” project that you have built earlier

Step 9: Click on Save

Step 10: Navigate to the **Post-build action** in the DEVELOPER_CODE_REVIEW configuration settings and write “QA_TEST” in the **Build other Projects** field. Apply and Save.

Step 11: Build the project by clicking on Build Now.
After the successful build, your project icon will turn blue.

Step 12: Click on the Latest Test Result option to check the result of the Unit Test.



Test Result

0 failures

23 tests

Took 1.9 sec.

All Tests

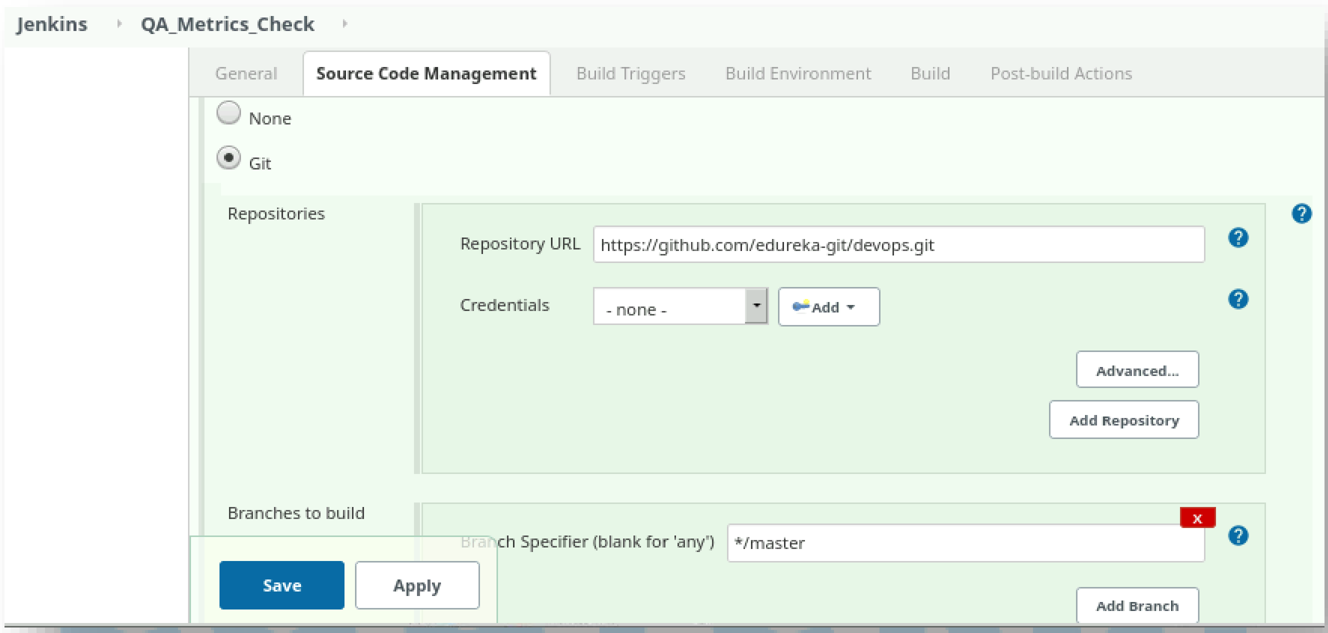
Package	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
com.edurekademo.utilities	3.7 sec	0		0		23	+23	23	+23

- Create a freestyle project with the name QA_ METRICS_CHECK in Jenkins to check the test cases.

Perform Step 1 and 2 as mention in the question 1 solution

Step 3: Give the item name as QA_ METRICS_CHECK and select Freestyle Project.

Step 4: In Source code management, select the Git radio button and add your repository link in the **Repository URL** field.


Jenkins › QA_Metrics_Check ›

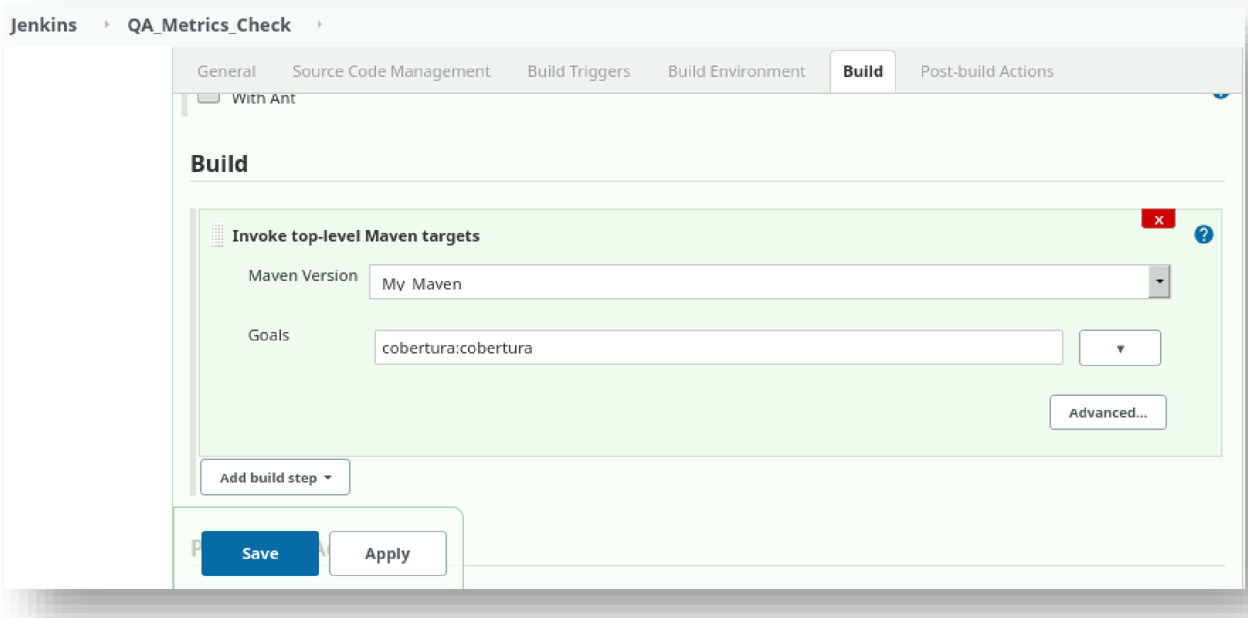
General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

None
Git

Repositories
Repository URL: https://github.com/edureka-git/devops.git
Credentials: - none -

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

Step 5: In Build, enter “cobertura: cobertura” in the **Goals and options** field.


Jenkins › QA_Metrics_Check ›

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

With Ant

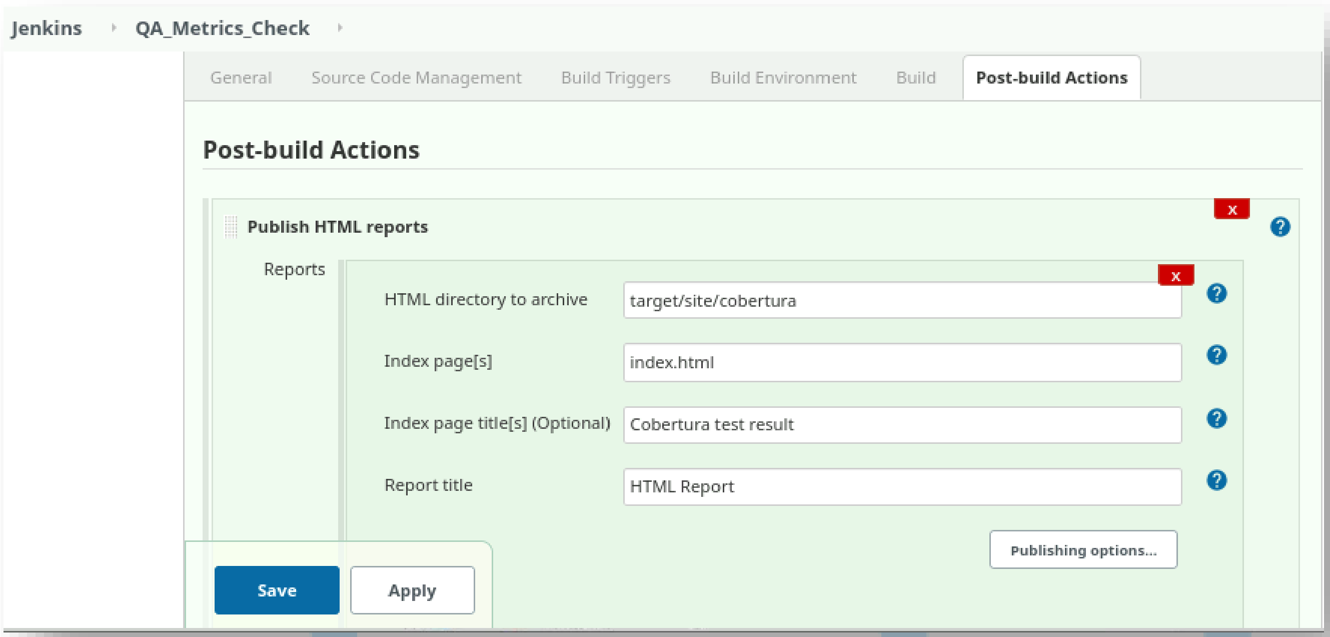
Build

Invoke top-level Maven targets
Maven Version: My Maven
Goals: cobertura:cobertura

Step 6: Select **Publish HTML Reports** from the Post Build Action menu.

Write “target/site/cobertura” in the **HTML directive to archive** field.

In **Report Title**, write Cobertura Test Result.



Post-build Actions

Publish HTML reports

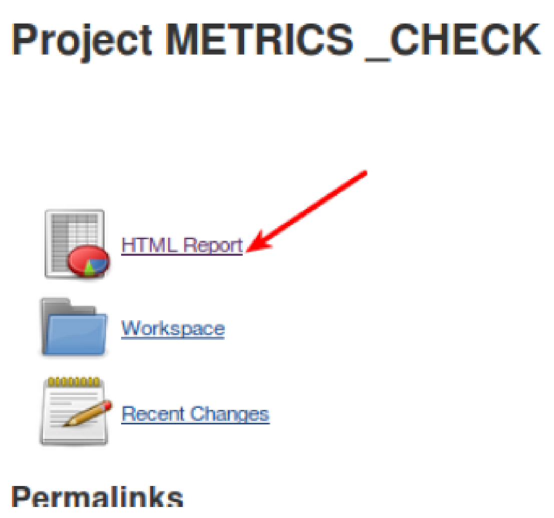
Reports

HTML directory to archive	target/site/cobertura
Index page[s]	index.html
Index page title[s] (Optional)	Cobertura test result
Report title	HTML Report

Click Save

Step 7: Build the project by clicking on Build Now.
After the successful build, your project icon will turn blue.

Step 8: Once it is a success, you can download the HTML reports from the below location :



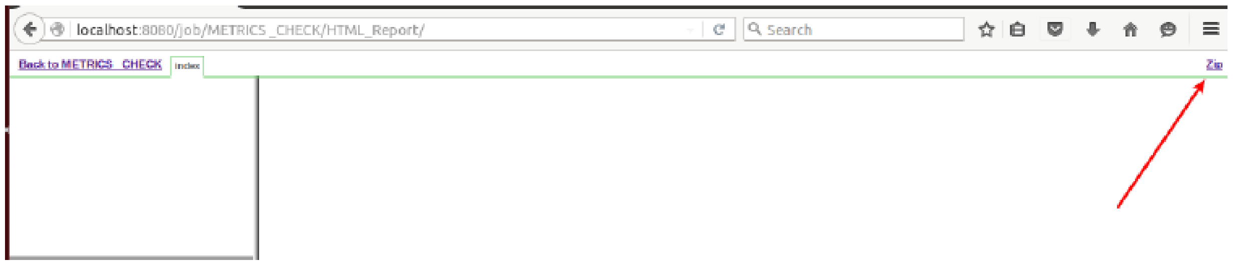
Project METRICS _CHECK

HTML Report

Workspace

Recent Changes

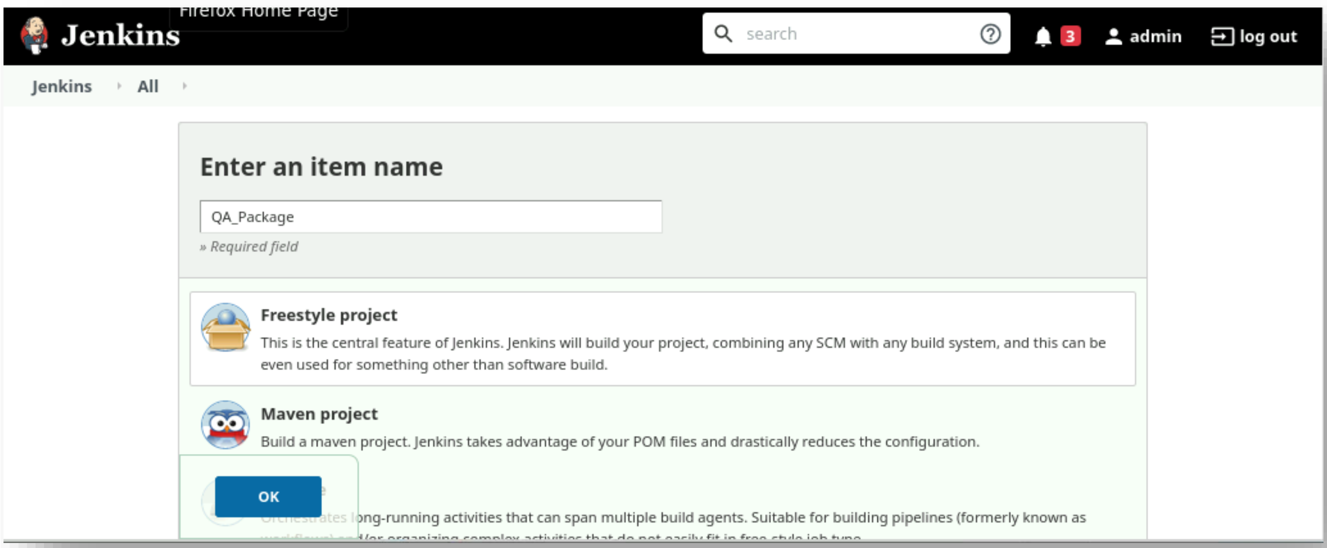
Permalinks



localhost:8080/job/METRICS _CHECK/HTML_Report/

- Create a freestyle project with the name QA_ PACKAGE in Jenkins to create an executable jar/war file.

Step1: Create a new Freestyle Project named QA_ PACKAGE from the Jenkins Dashboard.



Enter an item name

QA_Package

» Required field

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.

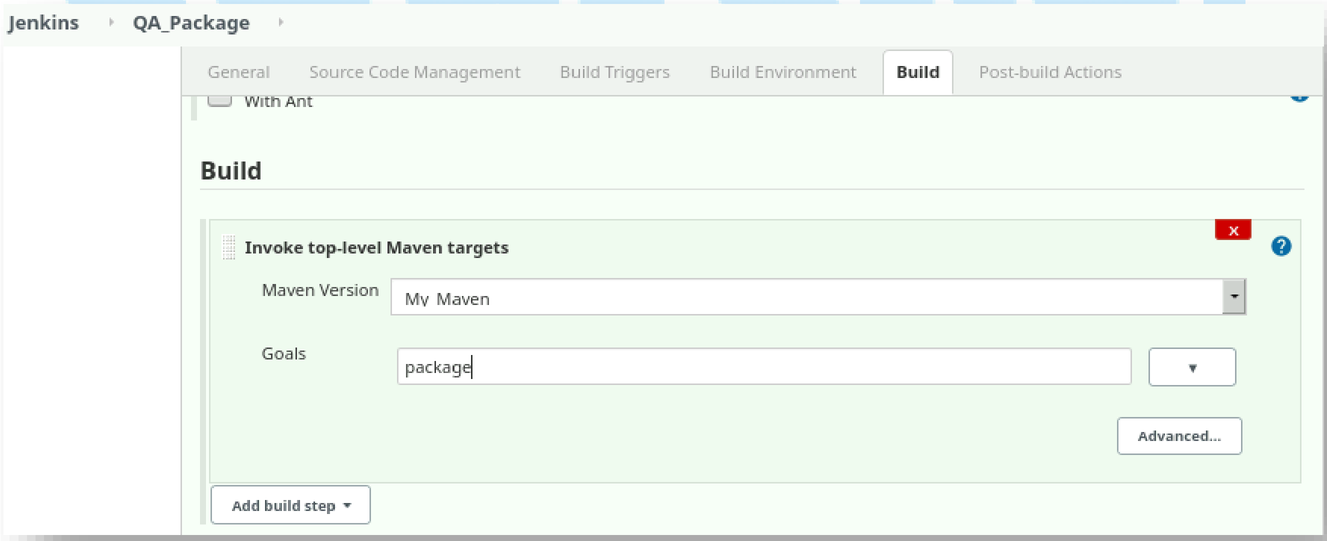
Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as

Navigate to Configuration Settings of QA_ PACKAGE.

Enter your Git repository link under **Source Code Management** Tab

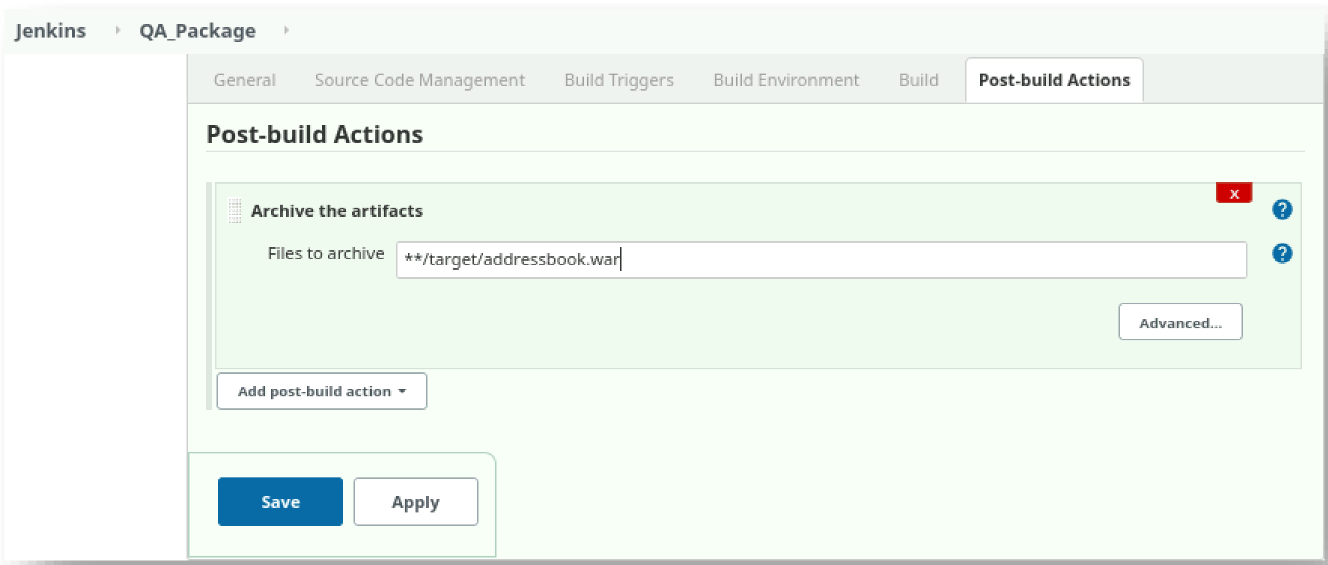


Build

Invoke top-level Maven targets

Maven Version: My Maven

Goals: package

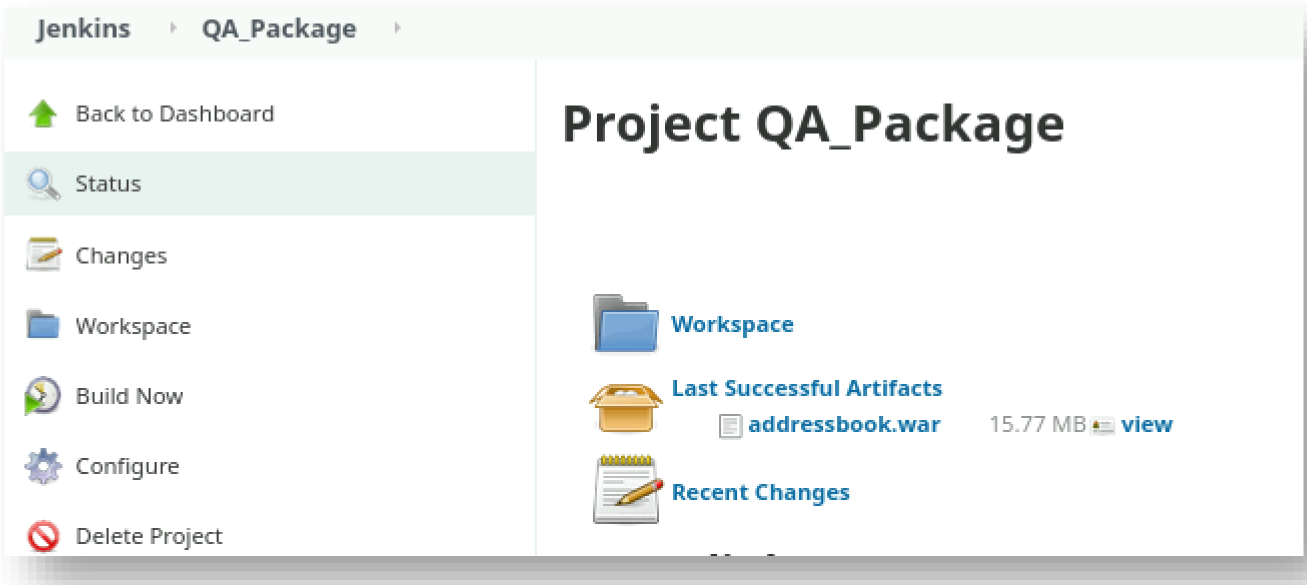


Then navigate to the **Post Build Actions**, Select **Archive the artifacts** and write "**/target/addressbook.war"

This build will make a .war file and show it once it is a success:

Apply and Save

Build QA_ PACKAGE.



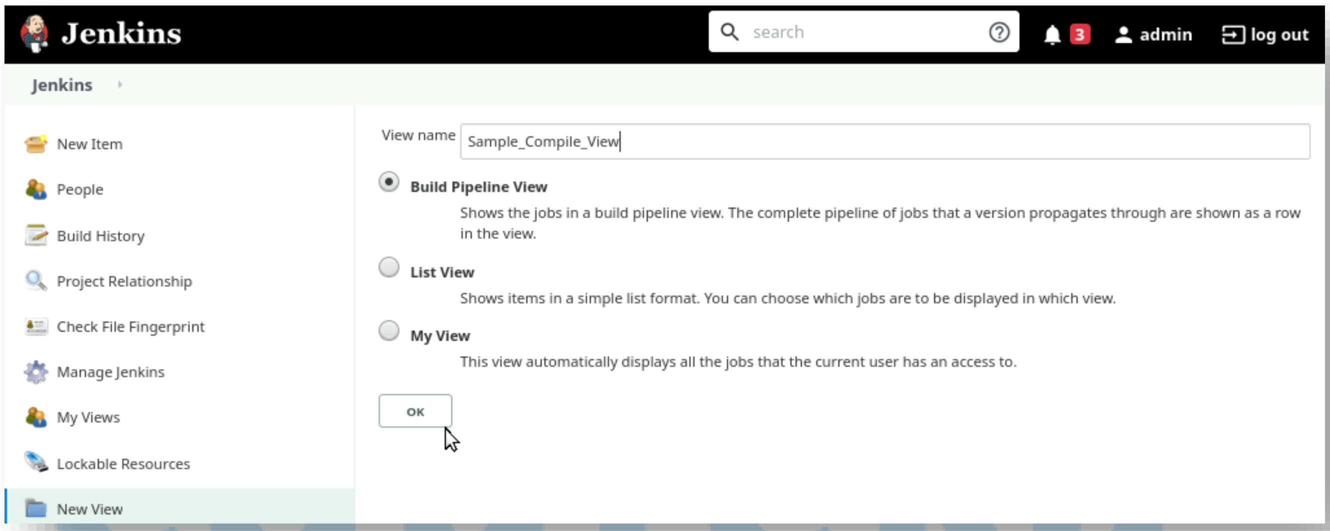
- Create a pipeline named SAMPLE_COMPILE_VIEW with **Build Pipeline View** option**,** select DEVELOPER_COMPILE project under layout section, and run the pipeline to check the console output

Step 1: Go to Jenkins Dashboard and click on the "+" button. That button is for adding a view.

You will be redirected to the following screen.

Step 2: Give View name as SAMPLE_COMPILE_VIEW

Step 3: Select the build Pipeline View radio button and press OK.



View name: Sample_Compile_View

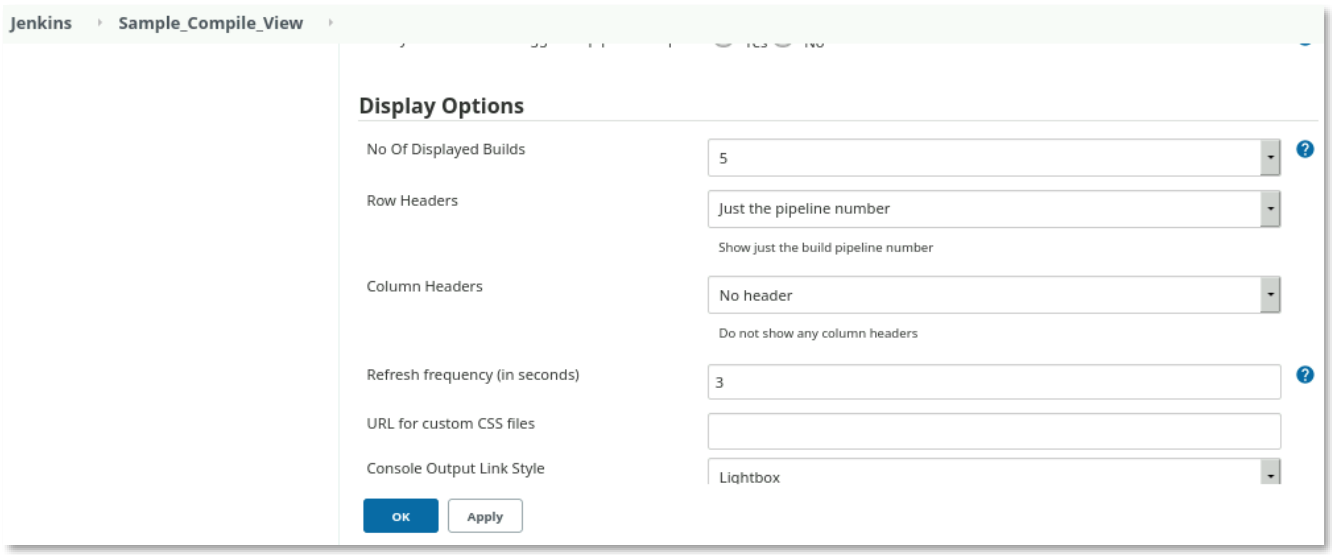
[✓ Build Pipeline View]
Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.

List View
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

My View
This view automatically displays all the jobs that the current user has an access to.

OK

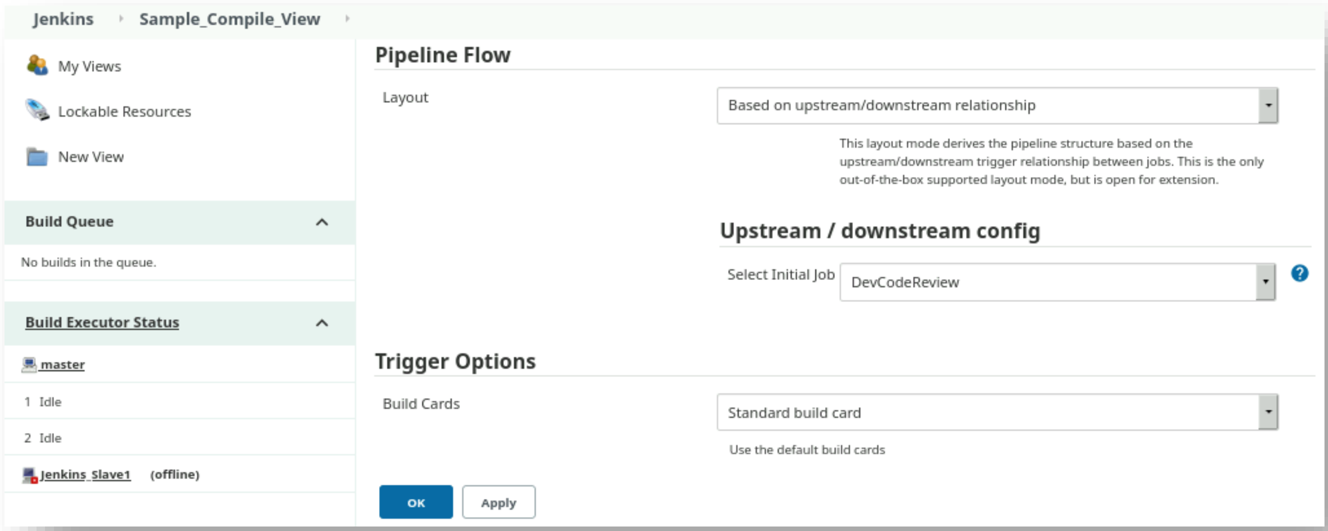
Step 4: You will be redirected to the Pipeline Configuration screen. Select 5 for the **Number of Displayed Builds.**



Display Options

No Of Displayed Builds	5
Row Headers	Just the pipeline number Show just the build pipeline number
Column Headers	No header Do not show any column headers
Refresh frequency (in seconds)	3
URL for custom CSS files	
Console Output Link Style	Lightbox

Step 5: In Select Initial Job, select DEVELOPER_COMPILE from the drop-down.


Jenkins › Sample_Compile_View ›

- My Views
- Lockable Resources
- New View

Build Queue

No builds in the queue.

Build Executor Status

master
1 Idle
2 Idle
Jenkins_Slave1 (offline)

Pipeline Flow

Layout: Based on upstream/downstream relationship

This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs. This is the only out-of-the-box supported layout mode, but is open for extension.

Upstream / downstream config

Select Initial Job: DevCodeReview

Trigger Options

Build Cards: Standard build card

Use the default build cards

Step 6: Press Apply and OK.

- The pipelines can also be extended to running web tests and load tests. Explain how you would do the same using Jenkins

  Answer: Selenium can be tested in multiple ways. These are:

Solution	Complexity	Pros	Cons
1. Selenium on one machine - headless browsers. Run the Selenium tests directly on your Jenkins workstation. Use headless browsers to save time and at least approximate GUI behavior.	Low	• Tests take much less time to run (compared to real browsers) • No need to update browsers - no real browsers used • Easier to scale, fewer system resources used • Open source	• Not testing real browsers and platforms • Still increases build time compared to unit testing
2. Selenium on one machine - real browsers. Run Selenium tests directly on your Jenkins workstation against real browsers. Build takes longer, but you'll be able to test real browser reactions.	Medium	• Testing real platforms • Open source	• Tests take a long time to run • Difficult to scale • Limited browser coverage - only what you have installed on Jenkins workstation • Need to install, update, maintain browsers
3. Selenium Grid.	High	• Tests take much less	• Need to update

Run a battery of Selenium tests quickly and on a wide range of browser/OS combinations by investing in infrastructure. Set up Selenium Grid to distribute test execution across multiple nodes.		time to run (compared to Selenium on one machine) • Testing real platforms (unlike headless testing) • Ability to support more browsers and operating systems • Open source	browsers • Difficult to scale further - need to add more machines and configure them • High cost of hardware and ongoing maintenance • Browser coverage still limited to the number of computers / operating systems you can physically set up. There are more than 700 OS/browser combinations in use today.
4. Sauce Labs Jenkins plugin. Outsource the infrastructure to Sauce Labs, run tests quickly on over 700 browser/OS combinations without disrupting your CI process and without installing and maintaining browsers locally.	Low	• Tests take much less time to run • Full browser coverage • Testing real platforms • Ability to scale effortlessly • Browsers are updated for you	• Commercial service - beyond the free plan limit, priced per minute. (Sauce is free for open source projects.)

Depending on what option you choose, the solution will differ. For Load testing, the same principle applies. You cannot run it on the same machine as Jenkins is located. It has to be spun off into slave where JMeter is installed, and tests run against same.