

# How to - End to End Jenkins integration with Gradle

## 1.1 Create a Jenkins job

1. Create new freestyle project
2. Under SourceCode select GIT

The screenshot shows the 'Source Code Management' configuration page in Jenkins. The 'Git' radio button is selected. Under 'Repositories', the 'Repository URL' is set to 'http://novartis.devops.altimetrik.io:7990/scm/nov/gradle-demo.git' and the 'Credentials' are set to 'admin/\*\*\*\*\* (Credential for logging into BitBucket)'. There are buttons for 'Advanced...', 'Add Repository', and 'Add Branch'. Under 'Branches to build', the 'Branch Specifier (blank for \'any\')' is set to '\*/master'. There is a red 'X' icon and a button for 'Add Branch'. Under 'Repository browser', the dropdown is set to '(Auto)'. At the bottom, there is an 'Additional Behaviours' section with an 'Add' button.

3. Create a "Build step" in the section "Build" by selecting "Invoke Gradle build script".

The screenshot shows the 'Build' configuration page in Jenkins. The 'Invoke Gradle script' step is selected. Under 'Invoke Gradle', the 'Gradle Version' is set to 'gradle6'. Under 'Use Gradle Wrapper', the 'Tasks' field is set to 'build sonarqube upload'. There is a red 'X' icon and a button for 'Advanced...'. At the bottom, there is an 'Add build step' button.

CMD : gradle build

Output: Build will successfully executed along with test cases

```
[test-gradle] $ /var/jenkins_home/tools/hudson.plugins.gradle.GradleInstallation/gradle6/bin/gradle build
Starting a Gradle Daemon (subsequent builds will be faster)
> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE
> Task :jar UP-TO-DATE
> Task :startScripts UP-TO-DATE
> Task :distTar UP-TO-DATE
> Task :distZip UP-TO-DATE
> Task :assemble UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test UP-TO-DATE
> Task :check UP-TO-DATE
> Task :build UP-TO-DATE
```

## 1.2 Integrate sonar

### Run Sonar with Build.gradle file

#### 1. Activate the SonarQube plugin in your build

To get Sonar working in Gradle you need to apply the sonar plugin, like this:

```
#####
build.gradle

plugins {
    id "org.sonarqube" version "2.7"
}

#####
```

#### 2. Adding code coverage to our build

If we want to add code coverage to such a project we need to add jacoco in the version corresponding to the jacoco-gradle-plugin to our libs in build.gradle

```
#####

build.gradle

plugins {

id 'jacoco'

}

#####
```

### 3.Configure the Scanner

Installation is automatic, but certain global properties should still be configured.Be aware that the scanner uses system properties so all properties should be prefixed by `systemProp`.

```
#####

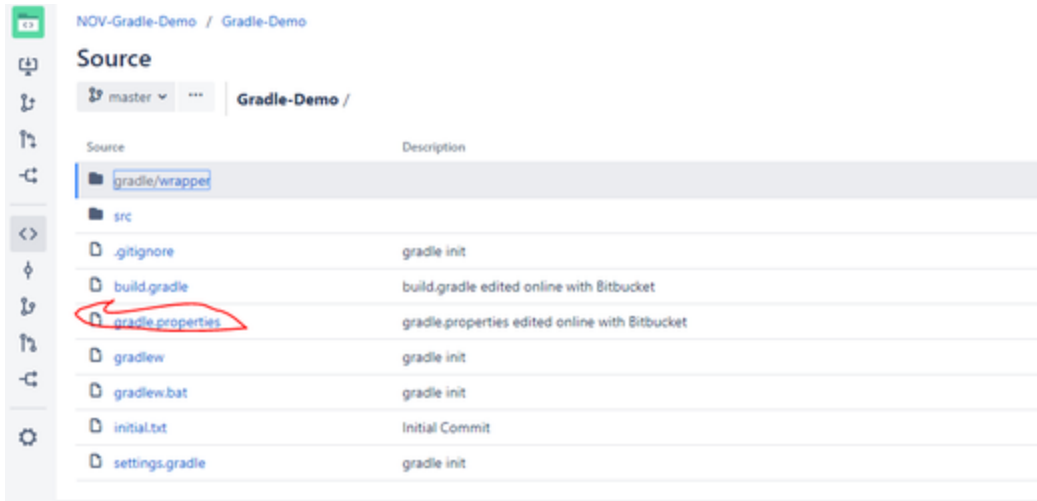
project root directory global.properties

systemProp.sonar.host.url=http://novartis.devops.altimetrik.io:9000/
systemProp.sonar.sourceEncoding=UTF-8
systemProp.sonar.forceAuthentication=true

#----- Token generated from an account with 'publish analysis'
permission
systemProp.sonar.login=<token>

#####
```

under project root directory global.properties



#### 4. Configure analysis properties

The SonarQube plugin leverages information contained in Gradle's object model to provide smart defaults for many of the standard SonarQube properties. The defaults are summarized in the tables below.

##### Gradle defaults for standard SonarQube properties

Property	Gradle default
<code>sonar.projectKey</code>	<code>"\$project.group:\$project.name"</code>
<code>sonar.projectName</code>	<code>project.name</code>
<code>sonar.projectDescription</code>	<code>project.description</code>
<code>sonar.projectVersion</code>	<code>project.version</code>
<code>sonar.projectBaseDir</code>	<code>project.projectDir</code>
<code>sonar.working.directory</code>	<code>"\$project.buildDir/sonar"</code>

##### Additional defaults when `java-base` plugin is applied

Property	Gradle default
<code>sonar.java.source</code>	<code>project.sourceCompatibility</code>
<code>sonar.java.target</code>	<code>project.targetCompatibility</code>

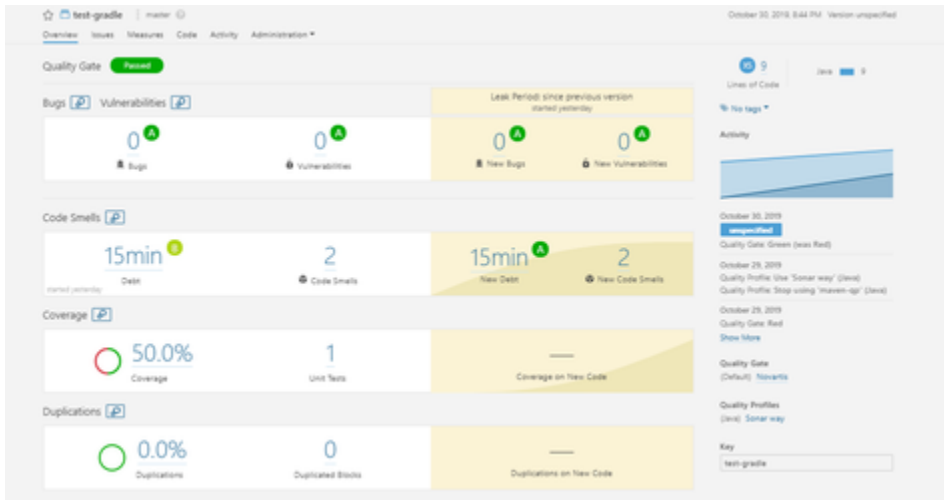
##### Additional defaults when `java` plugin is applied

Property	Gradle default
<code>sonar.sources</code>	<code>sourceSets.main.allSource.srcDirs</code> (filtered to only include existing directories)



```
[test-gradle] $ /var/jenkins_home/tools/hudson.plugins.gradle.GradleInstallation/gradle6/bin/gradle build sonarqube
Starting a Gradle Daemon (subsequent builds will be faster)
> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE
> Task :jar UP-TO-DATE
> Task :startScripts UP-TO-DATE
> Task :distTar UP-TO-DATE
> Task :distZip UP-TO-DATE
> Task :assemble UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test UP-TO-DATE
> Task :check UP-TO-DATE
> Task :build UP-TO-DATE
> Task :sonarqube
```

Go to sonar dashboard



### 1.3 Integrate with nexus

#### Run Nexus with Build.gradle file

##### 1. Activate the nexus plugin in your build

To get nexus working in Gradle you need to apply the maven plugin, like this:

```
#####  
To push artifacts in to nexus build.gradle  
  
plugins {  
  
    id 'maven'  
  
}  
  
#####
```

## 2. Configure properties in build file

build.gradle file that configures a build to publish artifacts to a novartis-maven repository

```
#####

build.gradle

repositories {
    maven {
        url
"http://novartis.devops.altimetrik.io:8082/repository/novartis-maven/"
    }
}

uploadArchives {
    repositories {
        mavenDeployer {
            repository(url:
"http://novartis.devops.altimetrik.io:8082/repository/novartis-maven/")
{
                authentication(userName: "admin", password: "password123")
            }

            pom.version = "1.0-SNAPSHOT"
            pom.artifactId = "gradle-demo"
            pom.groupId = "com.demo"
        }
    }
}

#####
```

### 3. Pushing jars into Nexus Repo

To upload artifacts from this project, run "gradle upload"

```
#####
To push artifacts

[gradle upload]
#####
```



## Build

Invoke Gradle script

Invoke Gradle

Gradle Version

Use Gradle Wrapper

Tasks

Advanced...

Add build step

output:

```
[test-gradle] $ /var/jenkins_home/tools/hudson.plugins.gradle.GradleInstallation/gradle6/bin/gradle build sonarqube upload
Starting a Gradle Daemon (subsequent builds will be faster)
> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE
> Task :jar UP-TO-DATE
> Task :startScripts UP-TO-DATE
> Task :distTar UP-TO-DATE
> Task :distZip UP-TO-DATE
> Task :assemble UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test UP-TO-DATE
> Task :check UP-TO-DATE
> Task :build UP-TO-DATE
> Task :sonarqube
> Task :uploadArchives
```

Go to nexus repo

<http://novartis.devops.altimetrik.io:8082/>

Sonatype Nexus Repository Manager  
OSS 3.15.2-01

Browse

Search

Custom

Maven

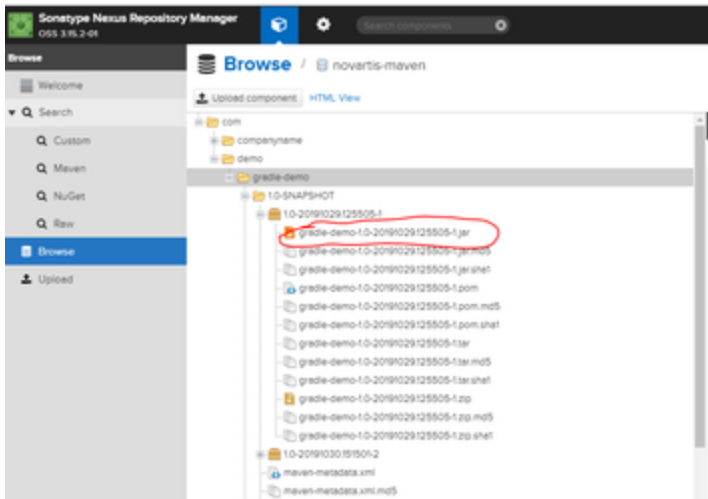
NuGet

Raw

Browse

Upload

Name ↑	Type	Format	Status	URL	Health check
maven-central	proxy	maven2	Online - Ready to Connect	copy	Analyze
maven-public	group	maven2	Online	copy	
maven-releases	hosted	maven2	Online	copy	
maven-snapshots	hosted	maven2	Online	copy	
Nexus_Repo	hosted	maven2	Online	copy	
<b>novartis-maven</b>	hosted	maven2	Online	copy	
nuget-group	group	nuget	Online	copy	
nuget-hosted	hosted	nuget	Online	copy	
nuget.org-proxy	proxy	nuget	Online - Ready to Connect	copy	Analyze
py-project	hosted	maven2	Online	copy	
Python-Project	hosted	raw	Online	copy	
R-project	hosted	raw	Online	copy	



#### 1.4 Pipeline As a code

*Pipeline as Code* describes a set of features that allow Jenkins users to define pipelined job processes with code, stored and versioned in a source repository. These features allow Jenkins to discover, manage, and run jobs for multiple source repositories and branches.

To use *Pipeline as Code*, projects must contain a file named `Jenkinsfile` in the repository root, which contains a "Pipeline script."

#### Jenkinsfile

The `Jenkinsfile` should contain a Pipeline script, specifying the steps to execute the job.

```
#####
Jenkinsfile

node {
stage("Poll SCM ") {
    git credentialsId: 'BitCred', url:
'http://novartis.devops.altimetrik.io:7990/scm/nov/gradle-demo.git'
}
stage("gradle build stage ") {
    sh "${tool name: 'gradle6', type: 'gradle'}/bin/gradle
build"

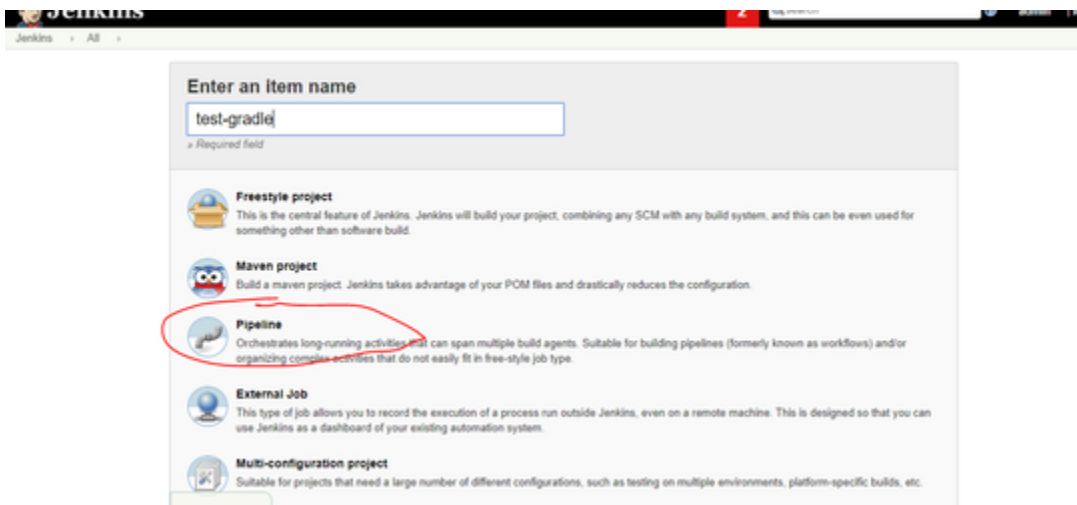
}

stage("gradle sonar ") {
    sh "${tool name: 'gradle6', type: 'gradle'}/bin/gradle
sonarqube"

}
stage("publish artifact") {
    sh "${tool name: 'gradle6', type: 'gradle'}/bin/gradle upload"
}
}
#####
```

## 1.5 Create pipeline job:

Goto Jenkins dashboard > Newitem> pipeline



Select pipeline as a script with scm

Definition Pipeline script from SCM

SCM Git

Repositories

Repository URL http://novartis.devops.altimetrix.io:7990/scm/novi/gradle

Credentials admin\*\*\*\*\* (Credential for logging into ...)

Add Advanced... Add Repository

Branches to build

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

Add Branch

Repository browser (Auto)

Additional Behaviours Add

Save Apply Script Path Jenkinsfile

Run build pipeline

Jenkins - test-pipeline

## Pipeline test-pipeline

Full project name: POCtest-pipeline

Recent Changes

### Stage View

Average stage times:  
(Average full run time: ~10s)

	Poll SCM	gradle build stage	gradle sonar	publish artifact
<span>172ms</span>	172ms	6s	9s	3s
<span>172ms</span>	172ms	6s	9s	3s
<span>1749</span>				
<span>1749</span>				
<span>1749</span>				
<span>1749</span>				

**Build History** Stand

find

- 1720 Oct 31, 2019 12:20 PM
- 1719 Oct 31, 2019 12:19 PM
- 1718 Oct 31, 2019 12:17 PM
- 1717 Oct 31, 2019 12:10 PM
- 1716 Oct 31, 2019 11:48 AM