

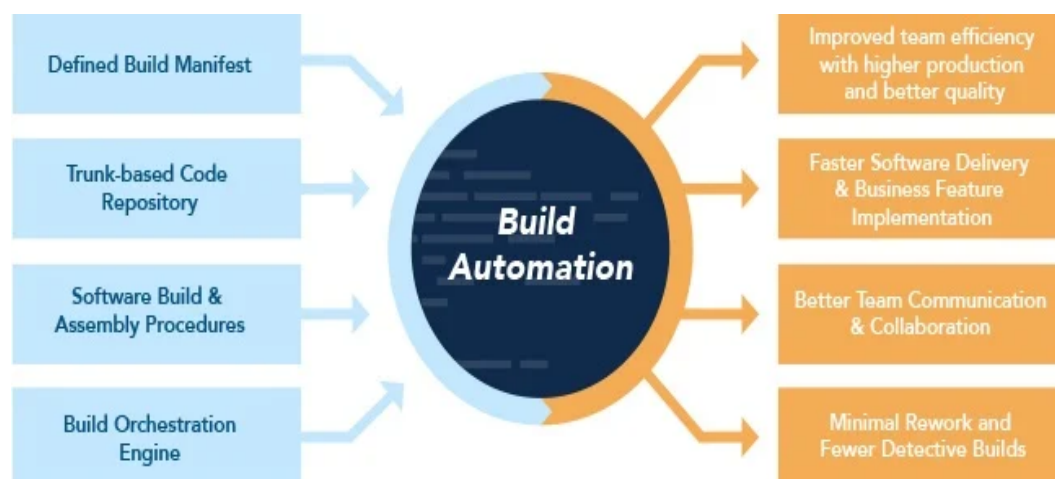
Gradle

Build Automation Tool	1
Why Gradle?	2
How many companies use Gradle	2
Key features of Gradle	3
Gradle vs. Maven	3
How to install Gradle in the machine	4
Demo on creating a Java project using the command-line interface (CLI)	5
Demo on adding custom task	7
Demo on creating Maven Project	8
Demo on creating Gradle Project	9
Demo on migrating from Maven to Gradle	10
Demo on Generating test report in HTML	10
Demo on generating scan report	11

Build Automation Tool

A build automation tool is a software tool or framework designed to automate the process of building, compiling, and packaging software applications. Build automation tools typically provide features such as dependency management, compilation, testing, packaging, and deployment.

These tools can automatically perform tasks such as fetching dependencies, compiling source code, running tests, generating documentation, and creating deployable artifacts. They often integrate with version control systems and continuous integration/continuous delivery (CI/CD) pipelines to ensure consistent and reliable builds.



Some popular build automation tools include:

1. Apache Maven
2. Gradle
3. Jenkins
4. Apache Ant
5. circleci

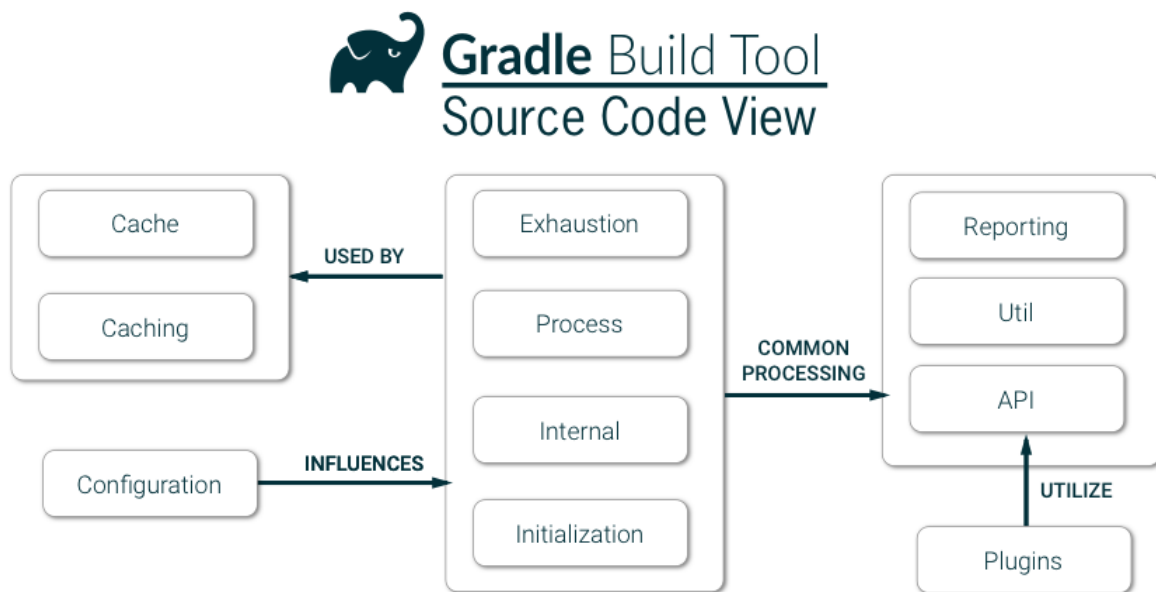
Why Gradle?

1. Flexible configuration using a concise and readable DSL.
2. Support for multiple programming languages, accommodating diverse technology stacks.
3. Robust dependency management, simplifying the handling of external libraries.
4. Incremental builds that speed up development iterations by rebuilding only what has changed.
5. Rich ecosystem of community-contributed plugins for extended functionality.
6. Gradle Wrapper for consistent Gradle version usage across environments.
7. Seamless integration with popular IDEs and CI/CD tools, ensuring smooth development and automation processes.

How many companies use Gradle

- 1194 companies reportedly use Gradle in their tech stacks
- Including Netflix, Udemy, and CRED.

Source Code View



Key features of Gradle

- Multi-project builds
- Gradle is the first build integration tool
- Ease of migration
- Free open source
- Different ways to manage your builds

Gradle vs. Maven

Gradle and Maven are both build tools with some primary distinctions on some fundamental grounds:

SI No	Features	Gradle	Maven
1	Language	Domain Specific Language - Groovy, Kotlin	XML
2	Graph	Yes - Directive Acyclic Graph	No - Linear Cyclic Graph
3	Build Cache	Yes	No
4	Highly Customizable	yes(Domain Specific Language) Write a Task for each phase	No - XML scope is limited

How to install Gradle in the machine

You can install the Gradle build tool on Linux or Windows.

Step 1 Download the latest Gradle distribution, <https://gradle.org/releases/> download from binary-only.

Step 2 Unpack the distribution.

1. Linux users
 - a. Unzip the distribution zip file in the directory of your choosing, e.g.:
 - b. `➤ mkdir /opt/gradle`
 - c. `➤ unzip -d /opt/gradle gradle-8.1.1-bin.zip`
 - d. `➤ ls /opt/gradle/gradle-8.1.1`
 - e. LICENSE NOTICE bin README init.d lib media
2. Microsoft Windows users
 - a. Create a new directory C:\Gradle with File Explorer.
 - b. unZIP archive to expose the content. Drag the content folder gradle-8.1.1 to your newly created C:\Gradle folder.

Step 3. Configure system environment

To run Gradle, the path to the unpacked files from the Gradle website needs to be on your terminal's path. The steps to do this are different for each operating system.

1. Linux users

Configure your PATH environment variable to include the bin directory of the unzipped distribution, e.g.:

```
➤ export PATH=$PATH:/opt/gradle/gradle-8.1.1/bin
```

Alternatively, you could also add the environment variable GRADLE_HOME and point this to the unzipped distribution. Instead of adding a specific version of Gradle to your PATH, you can add \$GRADLE_HOME/bin to your PATH. When upgrading to a different version of Gradle, just change the GRADLE_HOME environment variable.

2. Microsoft Windows users

In File Explorer right-click on the This PC (or Computer) icon, then click Properties → Advanced System Settings → Environment Variables.

Under System Variables select Path, then click Edit. Add an entry for C:\Gradle\gradle-8.1.1\bin. Click OK to save.

Alternatively, you could also add the environment variable GRADLE_HOME and point this to the unzipped distribution. Instead of adding a specific version of Gradle to your Path, you can add %GRADLE_HOME%/bin to your Path. When upgrading to a different

version of Gradle, just change the GRADLE_HOME environment variable.

Demo on creating a Java project using the command-line interface (CLI)

Create a Java project using the command-line interface (CLI) using gradle.

Steps

1. Go to the C://Gradle
2. Create a folder known as javaproject1
3. Open a terminal or command prompt.



4. Choose a directory where you want to create your Java project.
5. Navigate to the chosen directory using the cd command. For example, if you want to create the project in the Documents directory on Windows, you can use the following command:
`cd Documents`
6. Create a new directory for your Java project. You can name it anything you like. For example, let's name it MyJavaProject.
`mkdir MyJavaProject`
7. Move into the newly created project directory:
`cd MyJavaProject`
8. To initialize the project, run gradle init in the terminal. Choose project type 2 (application) and implementation language 3 (Java). Press enter for the default build script DSL and accept the remaining default values.

```

C:\Users\M.M\Documents\javaproject1>gradle init

Select type of project to generate:
 1: basic
 2: application
 3: library
 4: Gradle plugin
Enter selection (default: basic) [1..4] 2

Select implementation language:
 1: C++
 2: Groovy
 3: Java
 4: Kotlin
 5: Scala
 6: Swift
Enter selection (default: Java) [1..6] 3

Generate multiple subprojects for application? (default: no) [yes, no]

Select build script DSL:
 1: Groovy
 2: Kotlin
Enter selection (default: Groovy) [1..2] 1

```

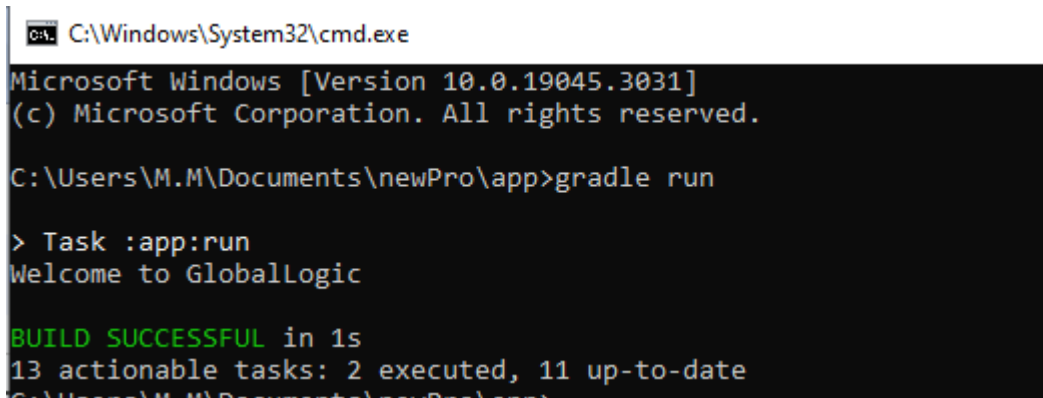
```

1  /*
2   * This Java source file was generated by the Gradle 'init' task.
3   */
4  package com.gl.myProj1.app;
5
6  import com.gl.myProj1.list.LinkedList;
7
8  import static com.gl.myProj1.utilities.StringUtils.join;
9  import static com.gl.myProj1.utilities.StringUtils.split;
10 import static com.gl.myProj1.app.MessageUtils.getMessage;
11
12 import org.apache.commons.text.WordUtils;
13
14 public class App {
15     public static void main(String[] args) {
16
17         System.out.println("Welcome to GlobalLogic");
18     }
19 }
20

```

9. To run the java application use to following command:
gradle run

Below is the output obtained from the command "gradle run"



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3031]
(c) Microsoft Corporation. All rights reserved.

C:\Users\M.M\Documents\newPro\app>gradle run

> Task :app:run
Welcome to GlobalLogic

BUILD SUCCESSFUL in 1s
13 actionable tasks: 2 executed, 11 up-to-date

```

Demo on adding custom task

Add a task in Gradle to create a custom task during the build phase.

To add a custom task in Gradle during the build phase, you can modify your build.gradle file as follows:

1. Open your build.gradle file in a text editor.
2. Inside the tasks block, define your custom task using the task keyword.
For example, let's create a custom task called customTask:

```

tasks.register('customTask') {
doLast {
    // Task actions go here
    println 'Executing custom task!'
}
}

```

In the above code, doLast is a closure where you define the actions to be performed when the task is executed. In this case, it simply prints a message.

3. Save the build.gradle file.
Now, when you run the build command (gradle build), the custom task will be executed as part of the build process.

You can run the custom task individually using the gradle <taskName> command, where <taskName> is the name you provided for your custom task. In this case, you would run gradle customTask to execute the customTask.

Demo on creating Maven Project

Using IDE how to create a Maven Project

To create a Maven project using an Integrated Development Environment (IDE) like Eclipse or IntelliJ IDEA, you can follow these general steps:

Using Eclipse:

1. Open Eclipse IDE.
2. Go to "File" -> "New" -> "Other".
3. In the "New" dialog, expand the "Maven" folder, select "Maven Project", and click "Next".
4. Select the option "Create a simple project (skip archetype selection)" and click "Next".
5. Provide the Group Id and Artifact Id for your Maven project. These values are used to uniquely identify your project. For example:
 - Group Id: com.example
 - Artifact Id: my-maven-project
 - Click "Finish".
6. Eclipse will create the Maven project structure with a pom.xml file and a default directory structure (src/main/java, src/test/java, etc.).
7. You can start adding Java source code to the project by creating packages and classes in the appropriate directories within src/main/java.

Using IntelliJ IDEA:

1. Open IntelliJ IDEA IDE.
2. Go to "File" -> "New" -> "Project".
3. In the "New Project" dialog, select "Maven" from the left menu.
4. Choose the option "Create from archetype" and click the "Add Archetype" button.
5. In the "Add Archetype" dialog, fill in the following details:
 - Group Id: org.apache.maven.archetypes
 - Artifact Id: maven-archetype-quickstart
 - Version: (Leave it as default)
 - Repository: (Leave it as default)
 - Click "OK".
6. Back in the "New Project" dialog, select the recently added archetype and click "Next".
7. Provide the Group Id and Artifact Id for your Maven project. These values are used to uniquely identify your project. For example:
 - Group Id: com.example
 - Artifact Id: my-maven-project
 - Click "Next".
8. Choose the project location and click "Finish".
9. IntelliJ IDEA will create the Maven project structure with a pom.xml file and a default directory structure (src/main/java, src/test/java, etc.).
10. You can start adding Java source code to the project by creating packages and classes in the appropriate directories within src/main/java.

Demo on creating Gradle Project

Using IDE how to create a Gradle Project

To create a Gradle project using an Integrated Development Environment (IDE) like Eclipse or IntelliJ IDEA, you can follow these general steps:

Using Eclipse:

1. Open Eclipse IDE.
2. Click on "File" in the menu bar and select "New" and then "Other".
3. In the "Select a wizard" window, expand the "Gradle" folder and choose "Gradle Project". Click "Next".
4. In the "New Gradle Project" window, enter the project details such as the project name, location, and build script DSL (Groovy or Kotlin).
5. Click "Next" to proceed.
6. In the "New Gradle Project" window, select the desired options for your project:
7. Select the desired Java version for your project.
8. Click "Finish" to create the Gradle project.
9. Eclipse will generate the Gradle project structure and import it into the workspace.
10. You can now work with your Gradle project in Eclipse, including managing dependencies, running tasks, and building the project using Gradle.
11. Note: Ensure that you have the Gradle Buildship plugin installed in your Eclipse IDE. You can install it by going to "Help" -> "Eclipse Marketplace" and searching for "Gradle Buildship". Once installed, restart Eclipse to enable Gradle support.

Using IntelliJ IDEA:

1. Launch IntelliJ IDEA and select "Create New Project" on the Welcome screen. If you already have a project open, you can also go to "File" > "New" > "Project".
2. In the "New Project" window, choose "Gradle" from the left sidebar. Ensure that the "Java" option is selected under "Additional Libraries and Frameworks". Click on "Next".
3. Specify the project location and click "Next".
4. Configure the project settings:
 - Set the "Group Id" and "Artifact Id" to uniquely identify your project.
 - Select the desired "Java version" for your project.
 - Choose the desired "Gradle version" (or leave the default).
 - Click "Next".
5. Select the desired project template. You can choose "Empty Project" or any other template based on your requirements. Click "Next".
6. Configure project dependencies:
 - Add any required libraries or frameworks by clicking on the "+" icon and selecting the desired dependencies.
 - Click "Next".
7. Specify the project name and location for the IntelliJ project files. Click "Finish" to create the project.

8. IntelliJ IDEA will create the Gradle project structure and download the necessary dependencies specified in your build.gradle file.

Demo on migrating from Maven to Gradle

Maven to Gradle Migration for a Spring Project

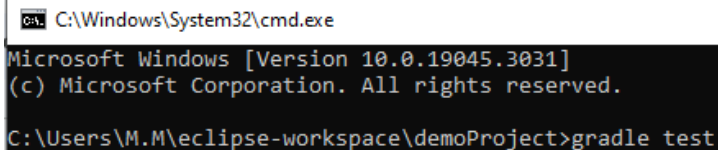
To convert Maven to Gradle, the only step is to run `gradle init` in the directory containing the POM. This will convert the Maven build to a Gradle build, generating settings. gradle file and one or more builds. gradle files.

Demo on Generating test report in HTML

To Generate Test Report in HTML

To generate a test report in HTML using a Gradle project, you can follow these steps:

1. Use the terminal to execute the command **gradle test**.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3031]
(c) Microsoft Corporation. All rights reserved.

C:\Users\M.M\eclipse-workspace\demoProject>gradle test
```

2. This will automatically generate the test report in the directory **build/reports/tests/test/index**.
3. Open the generated **index.html** file, and it will display the test report in an HTML format, providing a comprehensive overview of the test results.

Test Summary

5 tests	0 failures	0 ignored	0.027s duration	100% successful
------------	---------------	--------------	--------------------	--------------------

Packages	Classes
----------	---------

Package	Tests	Failures	Ignored	Duration	Success rate
demoProject	5	0	0	0.027s	100%

Demo on generating scan report

To generate Scan report using gradlew build --scan you can follow the below steps:

1. To create a directory, use the below command:
mkdir Build_Scan
2. To create a Gradle project, use the below command:
gradle init
3. Execute the below command to publish the build scan:
gradlew build --scan
4. The above command will create a build scan. Consider the below snap of output:

```

C:\Windows\System32\cmd.exe

C:\Users\M.M\IdeaProjects\demoBuildScan>gradlew build --scan

Welcome to Gradle 8.0!

For more details see https://docs.gradle.org/8.0/release-notes.html

Starting a Gradle Daemon, 1 incompatible Daemon could not be reused, use --status for details

BUILD SUCCESSFUL in 21s
2 actionable tasks: 2 executed

Publishing a build scan to scans.gradle.com requires accepting the Gradle Terms of Service defined at https://gradle.com
Gradle Terms of Service accepted.

Publishing build scan...CUTTING [16s]
https://gradle.com/s/vimjt4dlnujpc

```

Now, we have successfully published the Gradle project. From the above output, the green link is used to access the build scan.

5. Access the Build Scan online
6. To access the build scan online, follow the generated link and it will ask to activate the build scan. Enter the email id and click on the Go option. Consider the below image:

Activate your Build Scan™

An email will be sent to the specified address with a link to your Build Scan.

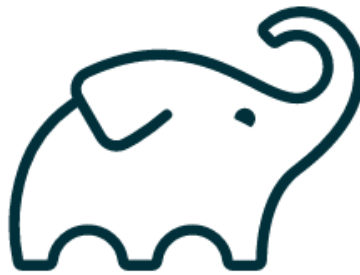
Email ☒ Remember me

By entering your email, you agree to the [Terms of Service](#) and [Privacy Policy](#).

☐ Subscribe to Gradle newsletters

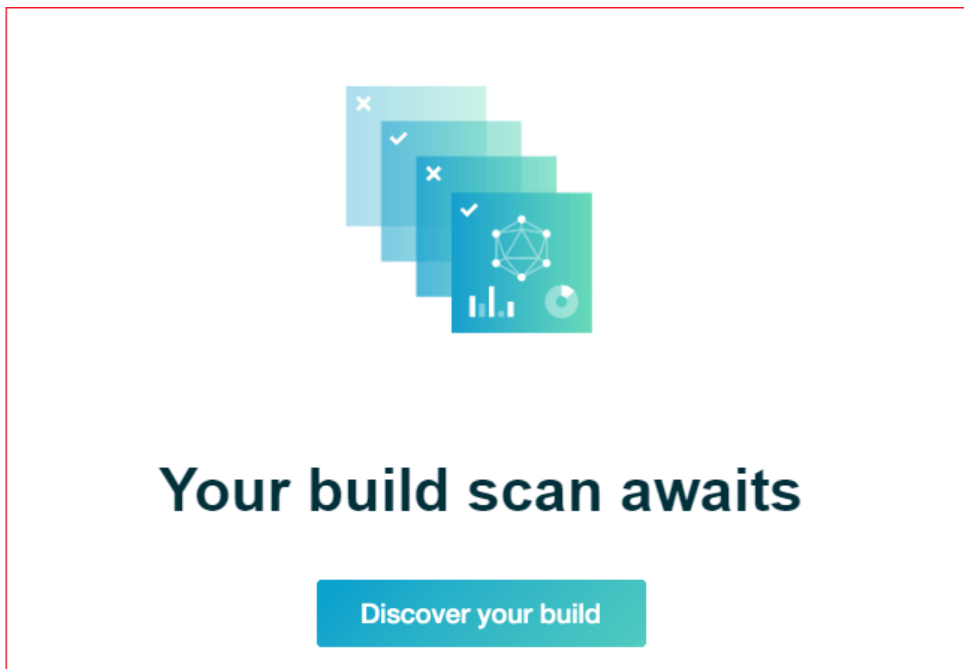
7. It will send a link to our email id for activating the build scan, follow that link. We can also delete the scans from the given option Delete this build scan, in the above screenshot.

Check your inbox

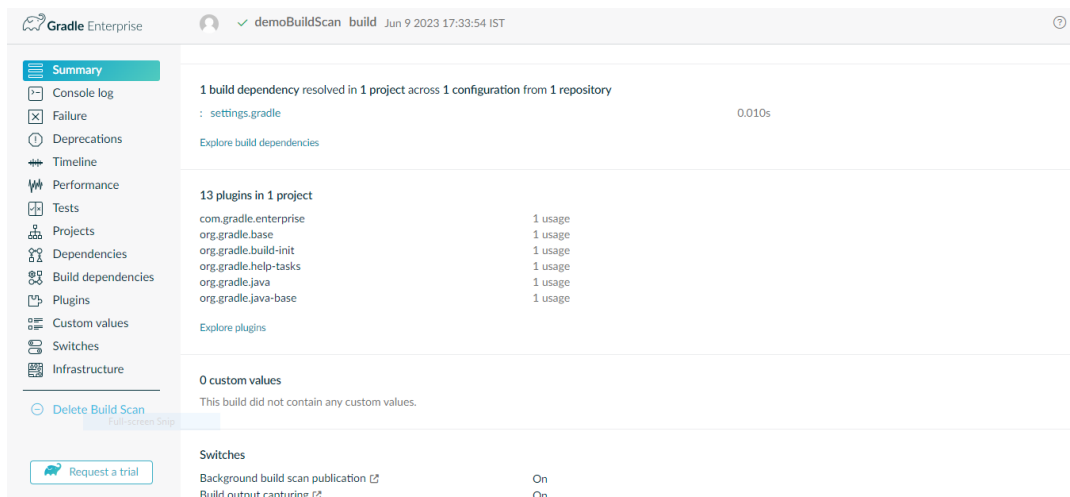


Follow the link in the email to activate your build scan.

8. Click on the Discover your build option to explore the created build.



9. It will open our created build. It will look like the below image:



We can explore all the information about the created build scan. It includes console log, failure points, timeline, performance, tests, projects, dependencies, plugins, switches, and infrastructures. It also contains the execution time, the required time at each stage, and the result of any test.