# Creating New Gradle Builds

Following this guide, you’ll create a trivial Gradle project, invoke some of the basic Gradle commands, and get a sense of how Gradle manages the project.

## Contents

What you’ll need

Initialize a project

Create a task

Apply a plugin

Explore and debug your build

Next Steps

Help improve this guide

## What you’ll need

* About 11 minutes
* A terminal or IDE application
* A Java Development Kit (JDK), version 1.7 or better (only necessary to run Gradle)
* A Gradle distribution, version 4.7 or better

*Shell commands will shown for Unix-based systems. Windows has analogous commands for each*.

## Initialize a project

First, let’s create a new directory where our project will go.

❯ mkdir basic-demo

❯ cd basic-demo

Now we can use Gradle’s init command to generate a simple project. We will explore everything that is generated so you know exactly what’s going on.

❯ gradle init

Starting a Gradle Daemon (subsequent builds will be faster)

BUILD SUCCESSFUL in 3s

2 actionable tasks: 2 executed

The command should show "BUILD SUCCESSFUL" and generate the following "empty" project. If it doesn’t, please ensure that Gradle is installed properly, and that you have the JAVA\_HOME environment variable set correctly.

This is what Gradle generated for you.

.

├── build.gradle 1.

├── gradle

│ └── wrapper

│ ├── gradle-wrapper.jar 2.

│ └── gradle-wrapper.properties 3.

├── gradlew 4.

├── gradlew.bat 5.

└── settings.gradle 6.

1. Project configuration script for configuring tasks in the current project
2. Gradle Wrapper executable JAR
3. Gradle Wrapper configuration properties
4. Gradle Wrapper script for Unix-based systems
5. Gradle Wrapper script for Windows
6. Settings configuration script for configuring which Projects participate in the build

*gradle init can generate various different types of projects, and even knows how to translate simple pom.xml files to Gradle.*

Boom! Roasted. We could just end the guide here, but chances are you want to know how to use Gradle in this project. Let’s do that.

## Create a task

Gradle provides APIs for creating and configuring tasks through a Groovy or Kotlin-based DSL. A Project includes a collection of Tasks, each of which performs some basic operation.

Gradle comes with a library of tasks that you can configure in your own projects. For example, there is a core type called Copy, which copies files from one location to another. The Copy task is very useful (see the documentation for details), but here, once again, let’s keep it simple. Perform the following steps:

1. Create a directory called src.
2. Add a file called myfile.txt in the src directory. The contents are arbitrary (it can even be empty), but for convenience add the single line Hello, World! to it.
3. Define a task called copy of type Copy (note the capital letter) in the main build file, build.gradle, that copies the src directory to a new directory called dest. (You don’t have to create the dest directory — the task will do it for you.)

task copy(type: Copy, group: "Custom", description: "Copies sources to the dest directory") {

from "src"

into "dest"

}

Here, group and description can be anything you want. You can even omit them, but doing so will also omit them from the tasks report, used later.

Now execute your new copy task:

❯ ./gradlew copy

:copy

BUILD SUCCESSFUL in 0s

1 actionable task: 1 executed

Verify that it worked as expected by checking that there is now a file called myfile.txt in the dest directory, and that its contents match the contents of the same one in the src directory.

# Apply a plugin

Gradle includes a range of plugins, and many, many more are available at the Gradle plugin portal. One of the plugins included with the distribution is the base plugin. Combined with a core type called Zip, you can create a zip archive of your project with a configured name and location.

Add the base plugin to your build.gradle file using the plugins syntax. Be sure to add the plugins {} block at the top of the file.

plugins {

id "base"

}

... rest of the build file ...

Now add a task that creates a zip archive from the src directory.

task zip(type: Zip, group: "Archive", description: "Archives sources in a zip file") {

from "src"

}

The base plugin works with the settings to create an archive file called basic-demo-1.0.zip in the build/distributions folder.

In this case, simply run the new zip task and see that the generated zip file is where you expect.

❯ ./gradlew zip

:zip

BUILD SUCCESSFUL in 0s

1 actionable task: 1 executed