

# 2

## Gradle

# Objectives

After completing this lesson, you should be able to do the following:

- An Introduction to Gradle
- Installing Gradle
- Gradle Build
- Gradle Dependencies
- Gradle Repository
- Projects & Tasks





# Introduction to Gradle

- **Gradle** is an open source **build automation** tool that is based on the concept of **Apache Maven** and **Apache Ant**. It is capable of building almost any type of software.
- It is designed for the multi-project build, which can be quite large. It introduces a **Java and Groovy-based DSL(Domain Specific Language)** instead of XML (Extensible Markup Language) for declaring the project configuration.
- It uses a DAG (Directed Acyclic Graph) to define the order of executing the task.

- Gradle offers an elastic model that can help the development lifecycle from compiling and packaging code for web and mobile applications. It provides support for the **building**, **testing**, and **deploying software** on different platforms.
- It has been developed for building automation on many languages and platforms, including Java, Scala, Android, C / C ++, and Groovy. Gradle provides integration with several development tools and servers, including Eclipse, IntelliJ, Jenkins, and Android Studio.
- Gradle is used by large projects such as **Spring Projects**, **Hibernate Projects**, and **Grails Projects**.

- Gradle was initially released in 2007, and it is stably released on November 18, 2019 (latest version 6.0.1). Gradle has taken the advantages of both Ant and Maven and remove the drawbacks of both.

# What is a Build Tool?

- Build tools are **programs that are used to automate the creation of executable** applications from source code.
- The building process involves compiling, linking, and **packaging the code into a useful or executable form.**

# Projects and Tasks in Gradle

- Gradle describes everything on the basis of **projects** and **tasks**.
- Every **Gradle build** contains one or more projects, and these projects contain some tasks.



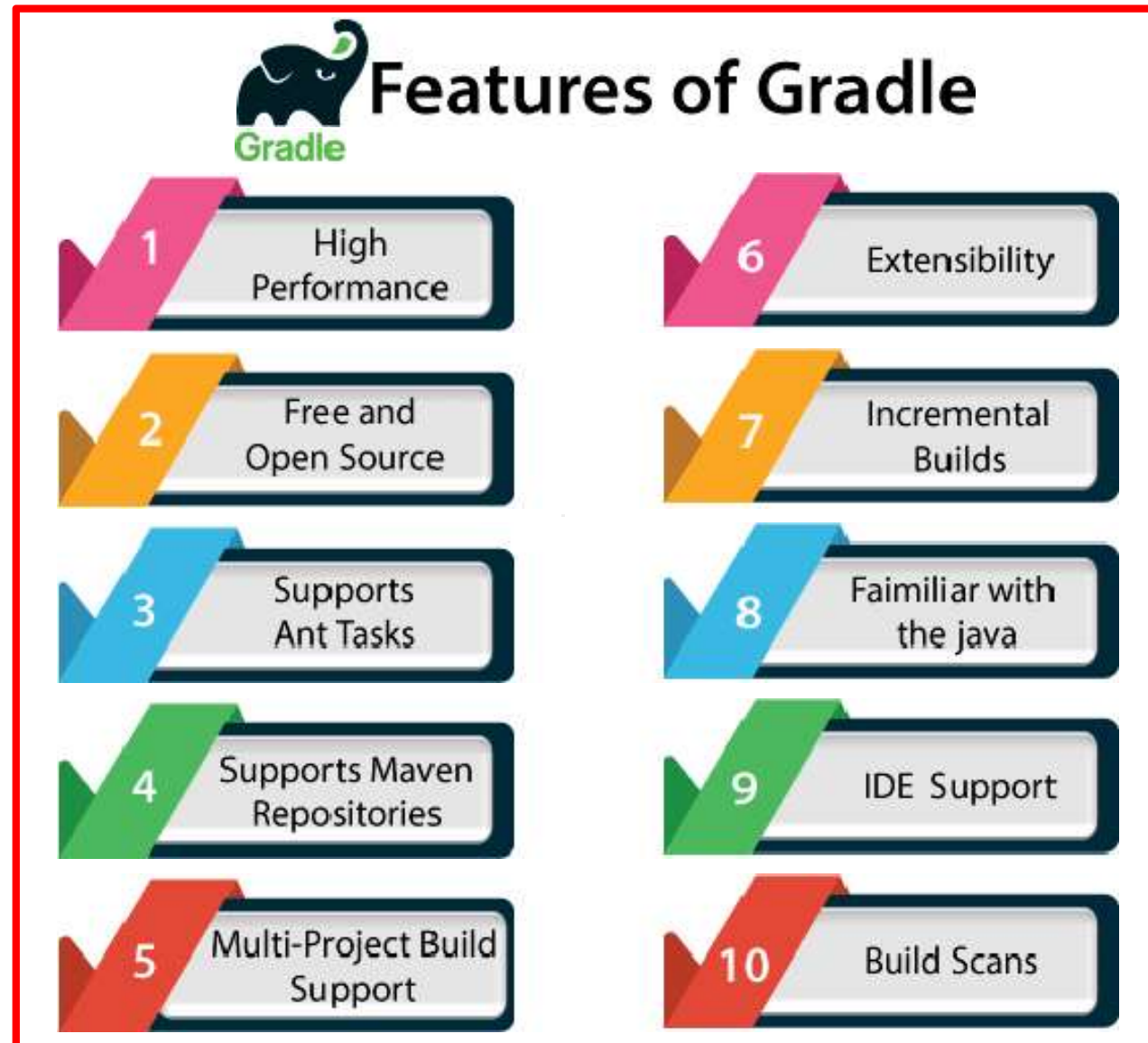
# Gradle Projects

- In Gradle, A project represents a library JAR or a web application. It may also represent a distribution ZIP, which is assembled from the JARs produced by other projects.
- A project could be deploying your application to staging or production environments. Each project in Gradle is made up of one or more tasks.

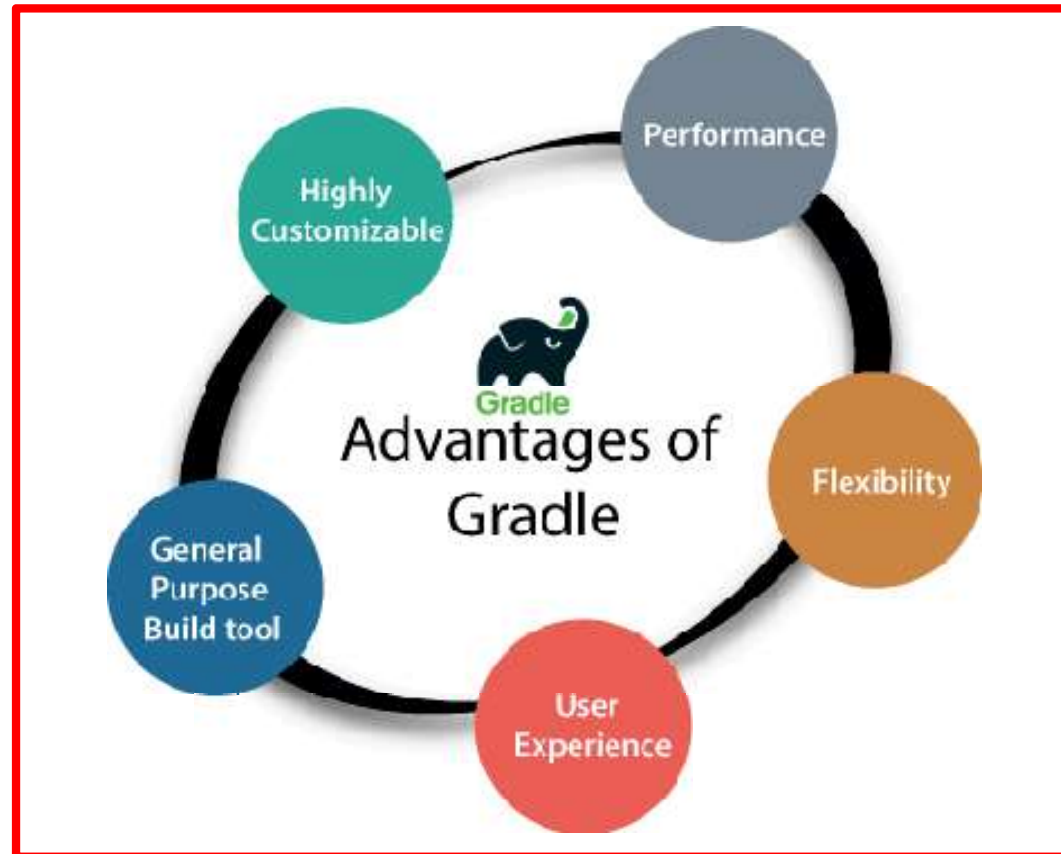
# Gradle Tasks

- In Gradle, Task is a single piece of work that a build performs.
- For example, it could **compile classes**, **create a JAR**, **Generate Javadoc**, and **publish some archives** to a repository and more.

# Features of Gradle



# Advantages of Gradle



# Why Gradle?

- Gradle is a modern build tool that comes up thinking about the challenges we have faced on other tools like ANT and Maven.
- The build tool should help us accomplish the goal of automating the project. Therefore we should not compromise on maintainability, usability, flexibility, extendibility, or performance.
- It is developed to **overcome the drawbacks of Maven and Ant** and supports a wide range of IDEs.
- It features **lots of plug-ins** that can be written on our prediction. Also, it can be used for large projects such as Spring projects, Hibernate projects, and Grails projects. So it may be the right choice for us to choose Gradle as our build tool.



# Installation of Gradle

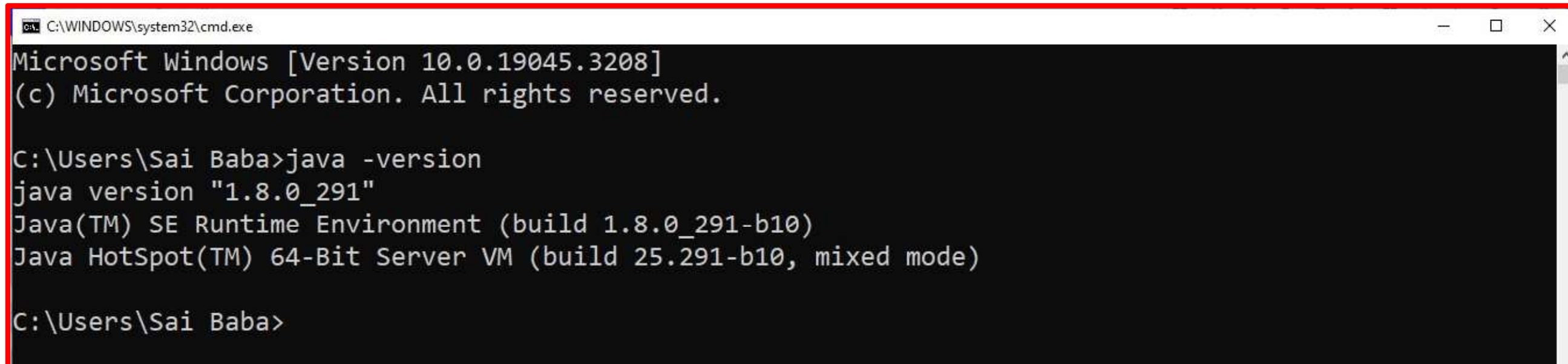
- Gradle is a build automation tool that combines the best of Maven and Ant. It is a potent and customizable tool.
- It also uses a sleek Groovy DSL instead of the XML file of Maven and Ant. Let's understand how to install Gradle.

# Gradle Prerequisites

- Gradle is a Java based tool, so **Java 8** or higher version is required on a computer to run it.
- Before installing Gradle, make sure you have the Java SDK 8 or higher version installed. It runs on all major operating systems.
- We do not need to install groovy because **Gradle has its groovy** library.



To verify Java Installation, run the Java -version command:

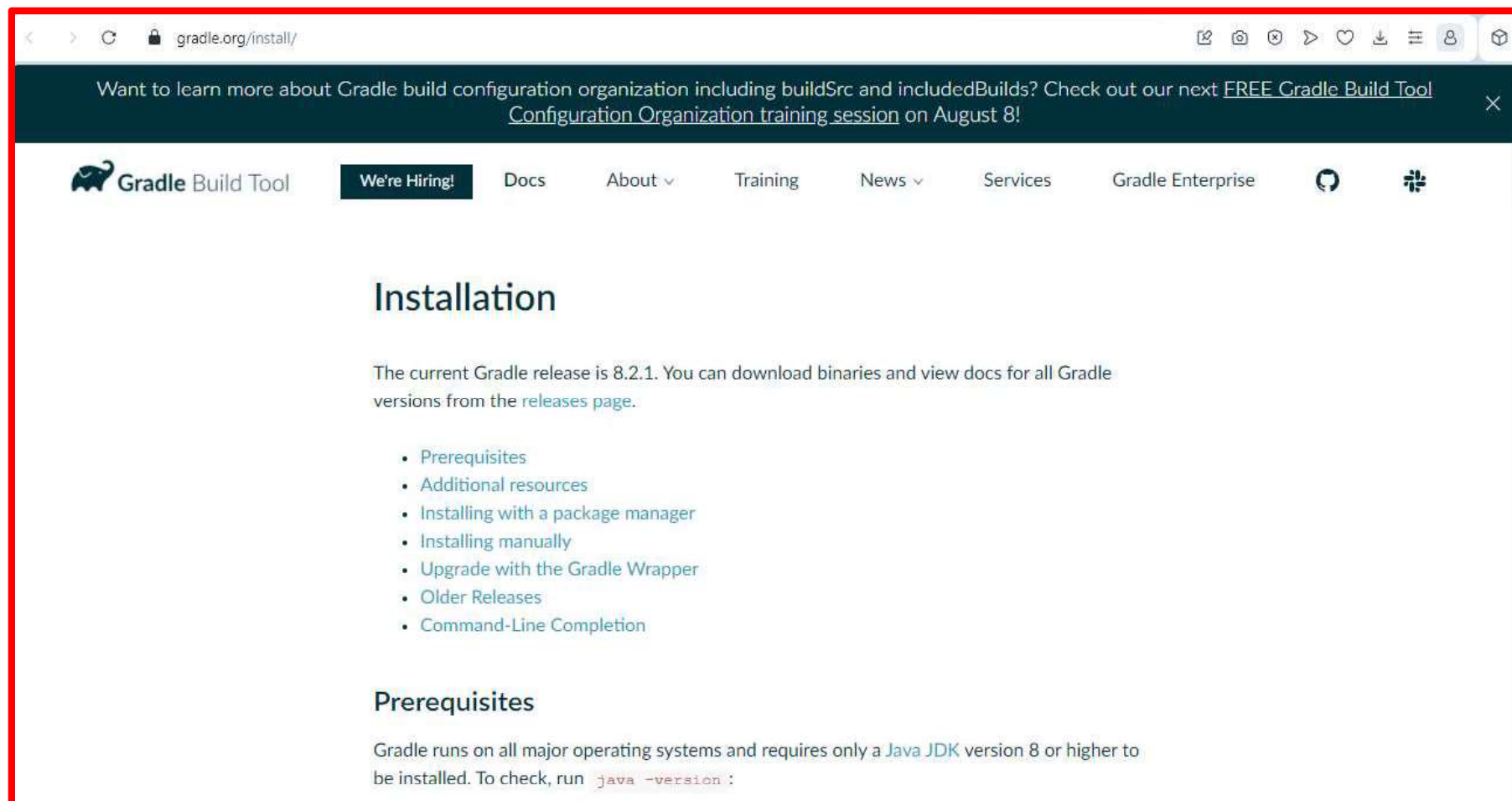
A screenshot of a Windows Command Prompt window. The title bar shows 'C:\WINDOWS\system32\cmd.exe'. The window contains the following text: 'Microsoft Windows [Version 10.0.19045.3208]', '(c) Microsoft Corporation. All rights reserved.', 'C:\Users\Sai Baba>java -version', 'java version "1.8.0\_291"', 'Java(TM) SE Runtime Environment (build 1.8.0\_291-b10)', 'Java HotSpot(TM) 64-Bit Server VM (build 25.291-b10, mixed mode)', and 'C:\Users\Sai Baba>'.

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sai Baba>java -version
java version "1.8.0_291"
Java(TM) SE Runtime Environment (build 1.8.0_291-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.291-b10, mixed mode)

C:\Users\Sai Baba>
```

- **Step-1:** Visit the [official download page](https://gradle.org/install/) of the Gradle. Under the **Installing manually** option, select the **Download** option given on the step1. Consider the below snap of the page: [ <https://gradle.org/install/> ]



- **Step2:** In this step, we will select the Gradle version and package. The **latest version** of the Gradle is **v8.2.1**. Select the package type **binary-only** or **complete** option. The recommended option is binary-only because the binary package contains all the required files for the installation.



gradle.org/next-steps/

Gradle Build Tool

We're Hiring! Docs About Training News

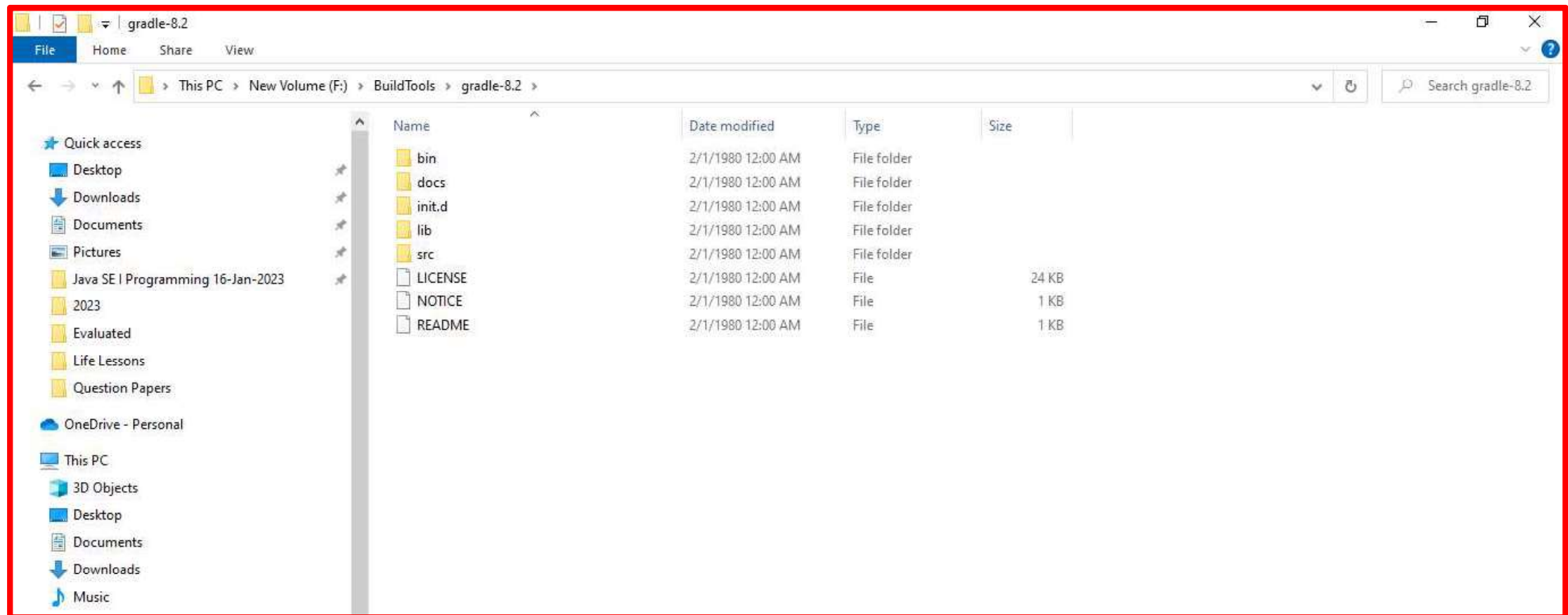
gradle-8.2-all.zip 65%  
2 secs left (119 MB of 184 MB, 23.9 MB/s)

# Thank you for downloading Gradle!

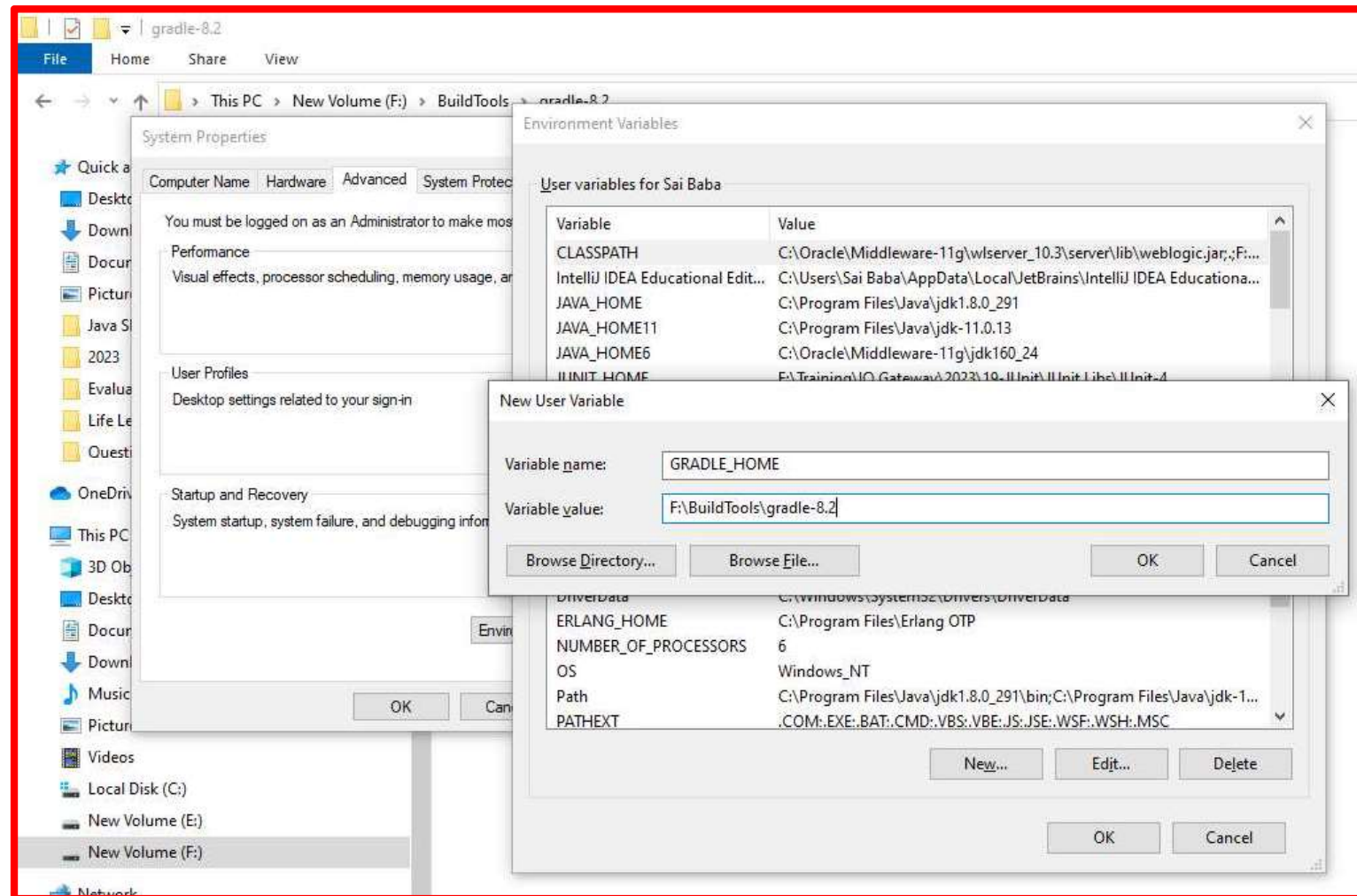
[› Verify SHA-256 checksum](#) [› Installation instructions](#) [› View Documentation](#)

Your download should start shortly. [If it doesn't, use the direct link.](#)

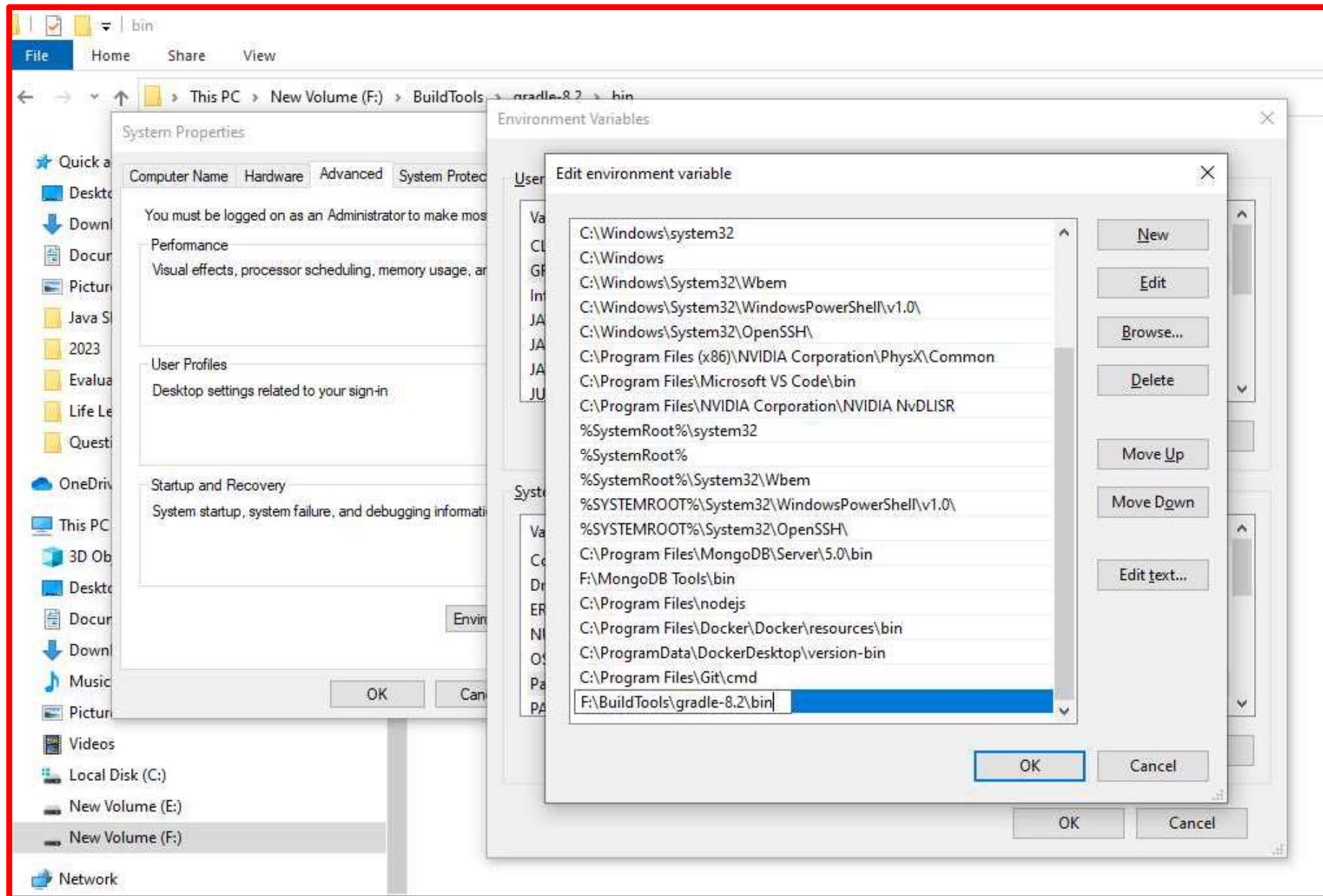
- **Step3:** Extract the zip file to a location on which we would like to install, copy the path from here for environment set up. Consider the below image:



- **Step4:** The next step is to set the environment variable for the system. To set the environment variable, open **This PC** and navigate to **System Properties**.

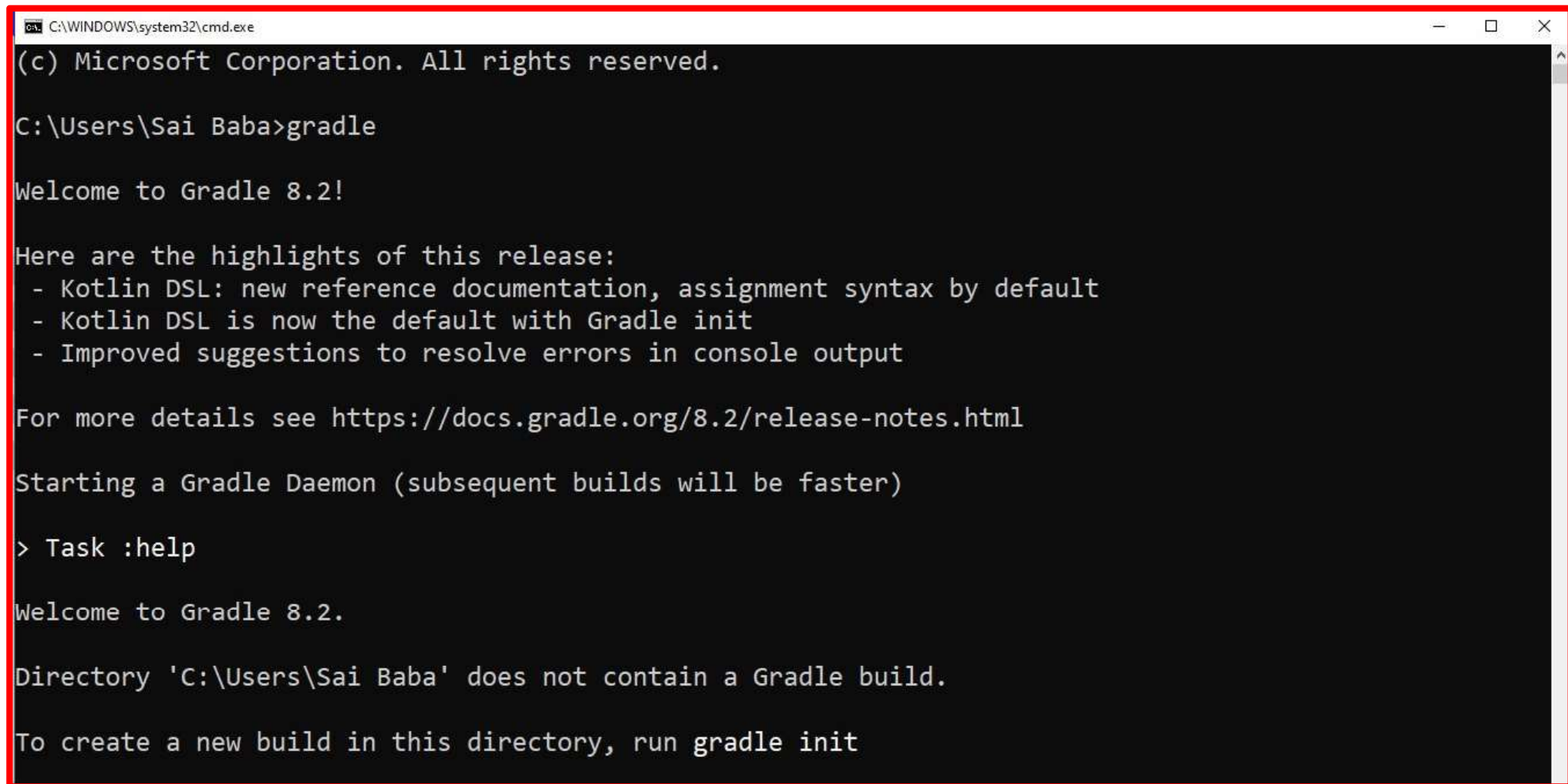






# Gradle Command Line

- The **gradle** command will provide the highlights of the latest release version and start a daemon process to configure the Gradle with the system.

A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window has a black background with white text. The text shows the execution of the 'gradle' command from the directory 'C:\Users\Sai Baba'. The output includes a welcome message for Gradle 8.2, a list of release highlights (Kotlin DSL documentation, Kotlin DSL as default, and improved error suggestions), a link to the release notes, and a message stating that the directory does not contain a Gradle build and suggesting to run 'gradle init' to create a new build.

```
C:\WINDOWS\system32\cmd.exe
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sai Baba>gradle

Welcome to Gradle 8.2!

Here are the highlights of this release:
- Kotlin DSL: new reference documentation, assignment syntax by default
- Kotlin DSL is now the default with Gradle init
- Improved suggestions to resolve errors in console output

For more details see https://docs.gradle.org/8.2/release-notes.html

Starting a Gradle Daemon (subsequent builds will be faster)

> Task :help

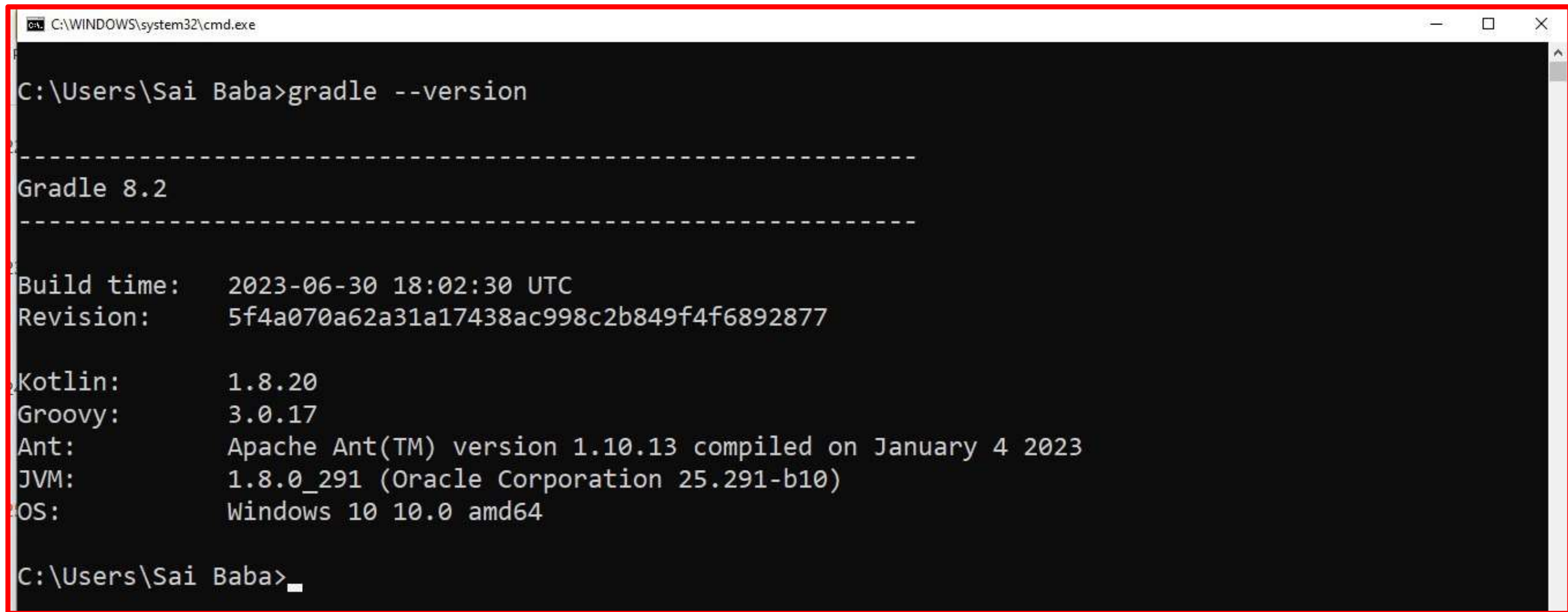
Welcome to Gradle 8.2.

Directory 'C:\Users\Sai Baba' does not contain a Gradle build.

To create a new build in this directory, run gradle init
```



- To check the version of the Gradle, run the **gradle -version** command. It will display the version of Gradle, Kotlin, Groovy, Ant, JVM, and OS.

A screenshot of a Windows Command Prompt window. The title bar shows 'C:\WINDOWS\system32\cmd.exe'. The command prompt shows the user 'Sai Baba' at the 'C:\Users\Sai Baba' directory. The command 'gradle --version' has been entered and executed. The output displays the Gradle version (8.2) separated by dashed lines, followed by build time and revision information. Below this, it lists the versions of Kotlin (1.8.20), Groovy (3.0.17), Ant (Apache Ant(TM) version 1.10.13), JVM (1.8.0\_291), and the OS (Windows 10 10.0 amd64).

```
C:\WINDOWS\system32\cmd.exe
C:\Users\Sai Baba>gradle --version

-----
Gradle 8.2
-----

Build time:   2023-06-30 18:02:30 UTC
Revision:     5f4a070a62a31a17438ac998c2b849f4f6892877

Kotlin:       1.8.20
Groovy:       3.0.17
Ant:          Apache Ant(TM) version 1.10.13 compiled on January 4 2023
JVM:          1.8.0_291 (Oracle Corporation 25.291-b10)
OS:           Windows 10 10.0 amd64

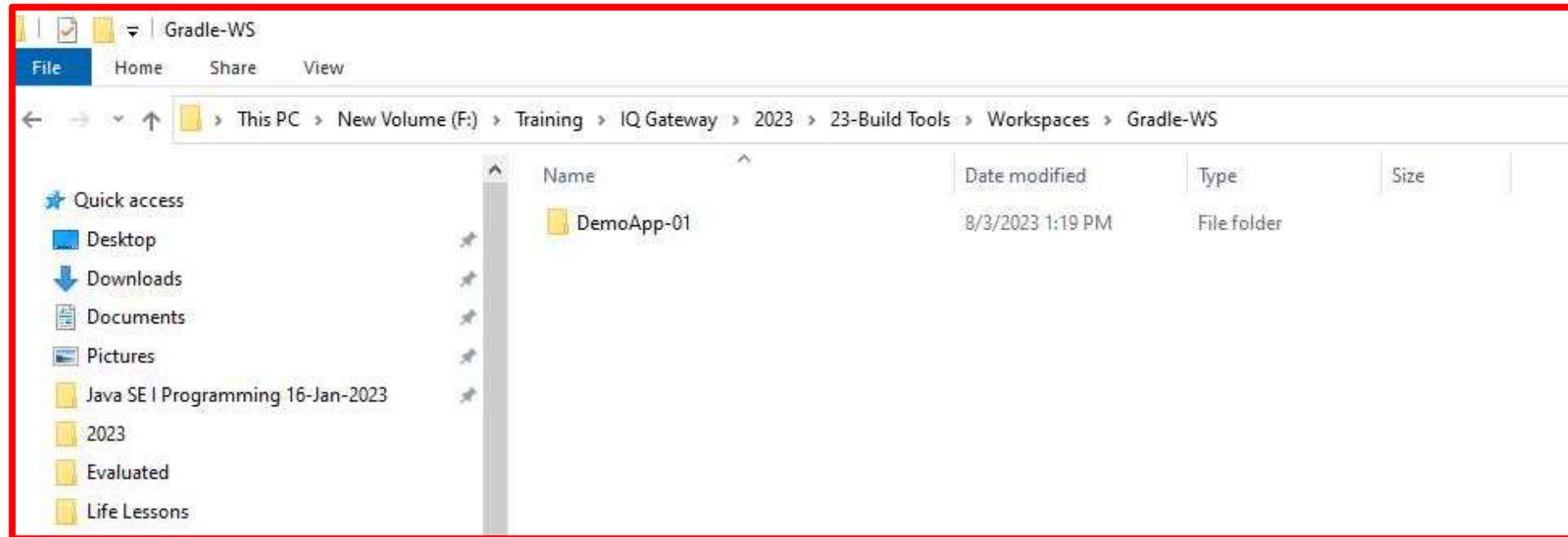
C:\Users\Sai Baba>
```



# Gradle Build

- The **Gradle build** is a process of creating a Gradle project. When we run a gradle command, it will look for a file called **build.gradle** in the current directory.
- This file is also called **the Gradle build script**. The build configuration, tasks, and plugins are described in this file. The build script describes a project and its tasks.

## Step 1: Create a directory in workspace



## Step2: Initialize a Gradle project

- To generate a Gradle project, run the **gradle init** command. It will generate a simple project. With this project, we will explore and understand everything that is generated.
- When we run the gradle init command, it will ask for some basic requirements. First, it will ask the **type of project** that we want to create. It will give four options:

```
1: basic  
2: application  
3: library  
4: Gradle plugin
```

```
C:\Windows\System32\cmd.exe - gradle init

Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.

F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>gradle init

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

Next, it will ask for DSL. There are two options that are available for DSL:

```
C:\Windows\System32\cmd.exe - gradle init
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.

F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>gradle init

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

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

- Next, it will ask for the **project name**. Type the project name and press Enter key. It will take a while to build a project. After the successful execution of the project, we will get a message **BUILD SUCCESSFUL**.

```
C:\Windows\System32\cmd.exe - gradle init
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.

F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>gradle init

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

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

Project name (default: DemoApp-01): First_Gradle
```



```
C:\Windows\System32\cmd.exe
3: library
4: Gradle plugin
Enter selection (default: basic) [1..4] 1

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

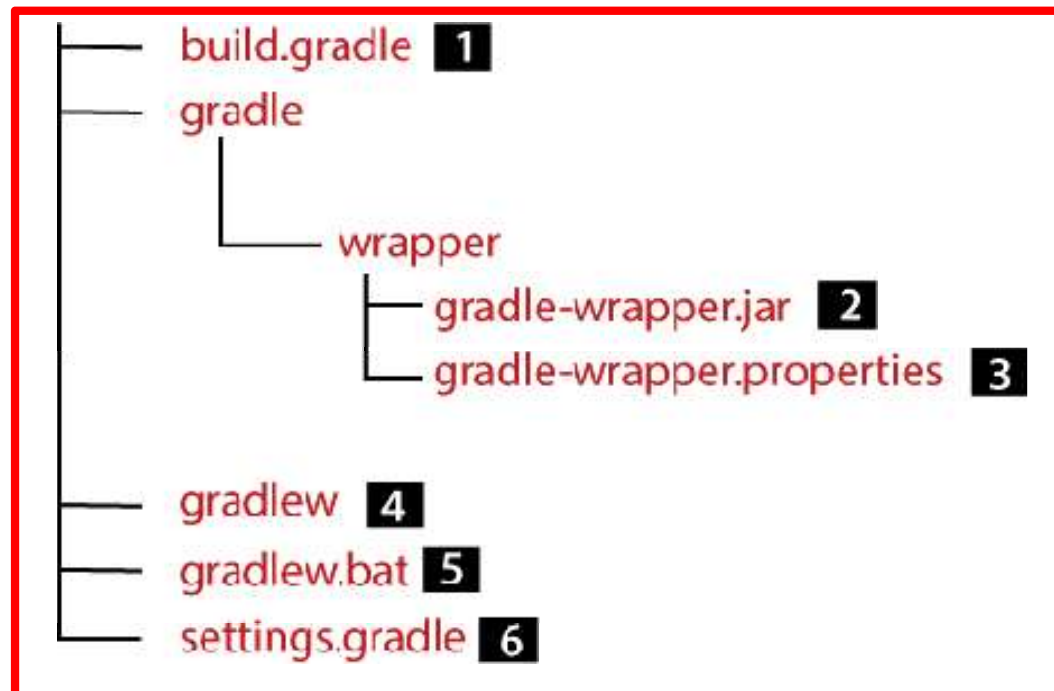
Project name (default: DemoApp-01): First_Gradle

Generate build using new APIs and behavior (some features may change in the next minor release)? (default: no) [yes, no] yes

> Task :init
To learn more about Gradle by exploring our Samples at https://docs.gradle.org/8.2/samples

BUILD SUCCESSFUL in 50s
2 actionable tasks: 2 executed
F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>
```

- Now we have successfully created a Gradle project. Now, what will happen to our specified directory? Consider the below structure of the Gradle project.



It is the default structure of a Gradle project. Gradle will generate the following things for us:

1. The **gradle** file is build script for configuring the current project.
2. An **executable JAR** file is used as a Gradle wrapper.
3. **Configuration properties** for Gradle Wrapper.
4. The **gradlew** is a Gradle wrapper script for UNIX based OS.
5. The **bat** is the Gradle Wrapper script for Windows.
6. **The settings script** for configuring the Gradle build.

### ➤ **Step3: Create a task**

- Gradle supports APIs for creating and managing tasks through a Groovy-based DSL or Kotlin-based DSL. Every project contains a collection of tasks for some basic operation.
- Gradle supports a library of tasks that configure the project. For example, there is a Copy task, which copies files from one location to another. The Copy task is one of the most used tasks In Gradle.
- To use the Copy task in build script, follow the below process.

- **Step1:** Create a directory called src.

```
C:\Windows\System32\cmd.exe

F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>mkdir src

F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>
```

- **Step2:** Add a file called **myfile.txt** in the src directory. Add the single line "Hello, World!" to it, also, we can leave it empty.

```
C:\Windows\System32\cmd.exe

F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>mkdir src

F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>echo testFile1.txt
testFile1.txt

F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>
```

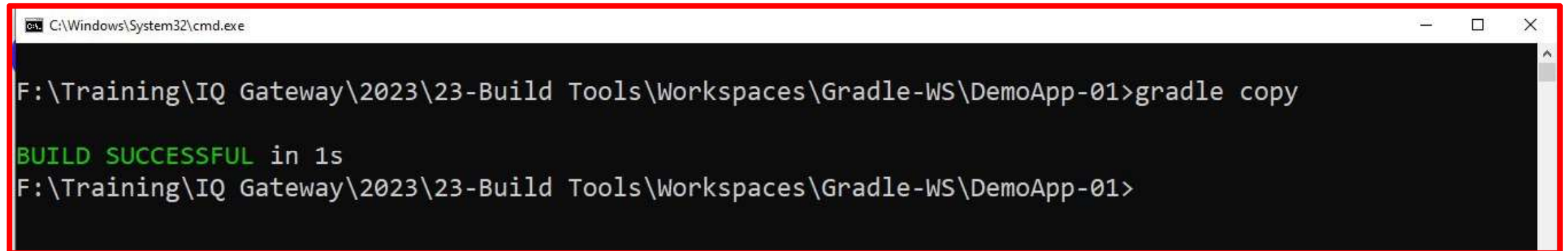
- Define a task called Copy in **build.gradle** file. It will copy the src directory to a new directory called dest. We don't have to create the dest directory; the Copy task will do it for us.

```

1  /*
2  * This file was generated by the Gradle 'init' task.
3  *
4  * This is a general purpose Gradle build.
5  * To learn more about Gradle by exploring our Samples at
6  * https://docs.gradle.org/8.2/samples
7  * This project uses @Incubating APIs which are subject to change.
8  */
9  task copy(type: Copy, group: "Custom", description: "The sources are copied to dest
10 directory") {
11     from "src"
12     into "dest"
13 }

```

- Now execute our new copy task:

A screenshot of a Windows command prompt window titled "C:\Windows\System32\cmd.exe". The window has a black background with white text. The command prompt shows the current directory as "F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01" and the command "gradle copy" has been executed. The output is "BUILD SUCCESSFUL in 1s" in green text. The prompt is now "F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>".

```
C:\Windows\System32\cmd.exe
F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>gradle copy
BUILD SUCCESSFUL in 1s
F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>
```



## Display the Information of the Gradle project

- To understand the structure, dependencies and debugging problems of a build, Gradle provides many built-in features that display information on a project.
- Following are some basic commands to display the information of the project:
- **Listing projects**
- In Gradle, all the sub-projects of a project in the workspace can be listed in a hierarchy. To do so, run the below command from the root directory of the project.

```
gradle -q projects
```



```
F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>gradle -q projects
```

```
-----  
Root project 'First_Gradle'  
-----
```

```
Root project 'First_Gradle'
```

```
No sub-projects
```

```
To see a list of the tasks of a project, run gradle <project-path>:tasks
```

```
For example, try running gradle :tasks
```

```
F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>
```

## ➤ Listing Tasks

- Gradle allows us to list all the essential tasks of the project. To list the task, run the below command:

```
gradle -q tasks
```

```
C:\Windows\System32\cmd.exe

F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>gradle -q tasks

-----
Tasks runnable from root project 'First_Gradle'
-----

Build Setup tasks
-----
init - Initializes a new Gradle build.
wrapper - Generates Gradle wrapper files.

Custom tasks
-----
copy - The sources are copied to dest directory

Help tasks
-----
buildEnvironment - Displays all buildscript dependencies declared in root project 'First_Gradle'.
dependencies - Displays all dependencies declared in root project 'First_Gradle'.
dependencyInsight - Displays the insight into a specific dependency in root project 'First_Gradle'.
help - Displays a help message.
javaToolchains - Displays the detected java toolchains.
outgoingVariants - Displays the outgoing variants of root project 'First_Gradle'.
projects - Displays the sub-projects of root project 'First_Gradle'.
```

To list all the tasks of the project, run the below command:

```
gradle tasks -all
```

To display more details about a task, run the below command:

```
gradle help --task
```

### **Listing Dependencies**

In Gradle, we can list the dependencies which are broken down by the configuration. To list the dependencies, run the below command:

```
gradle -q dependencies
```



# Gradle Dependencies

- Gradle build script describes a process of building projects. Most of the projects are not self-contained. They need some files to compile and test the source files. For example, to use Hibernate, we must include some Hibernate JARs in the classpath. Gradle uses some unique script to manage the dependencies, which needs to be downloaded.
- The dependencies are used to assist a task, such as required JAR files of the project and external JARs. Every dependency is applied to a specified scope. For example, dependencies are used to compile the source code, and some will be available at runtime. Gradle signifies the scope of a dependency with the help of a Configuration, and a unique name can recognize every configuration. Most of the Gradle plugins support pre-defined configuration for the projects.

- Gradle considers the outcomes of building and publishing the projects. Publishing is based on the task that we define. It can copy the files to a local directory or upload them to a remote Maven or Ivy repository. We can use these files from another project in the same multi-project build. The process of publishing a task is called publication.

# Dependency configurations

- Dependency configuration is a set of dependencies and artifacts. Following are three main tasks of configuration:
  1. Declaring dependencies
  2. Resolving dependencies
  3. Exposing artifacts for consumptions



# Declaring Dependency

- Dependency is an essential part of any project. We must declare a dependency to use it. Dependency configuration is a process of defining a set of dependencies. This feature is used to declare external dependencies, which we want to download from the web.

```
apply plugin: 'java.'  
repositories {  
    mavenCentral()  
}  
dependencies {  
    compile group: 'org.hibernate', name: 'hibernate-core', version: '3.6.7.Final'  
    testCompile group: 'junit', name: 'junit', version: '4.+'  
}
```

A dependency can be used on different phases of the project. These phases can be:

- **Compile:** At compile time, we will use the dependencies that are required to compile the production source of the project.
- **Runtime:** These dependencies are used at runtime by production classes. By default, it also contains the compile-time dependencies.
- **Test Compile:** These dependencies are required to compile the test source of the project. It also contains the compiled production classes and the compile-time dependencies.
- **Test Runtime:** These dependencies are required to run the tests. It also contains runtime and test compile dependencies.

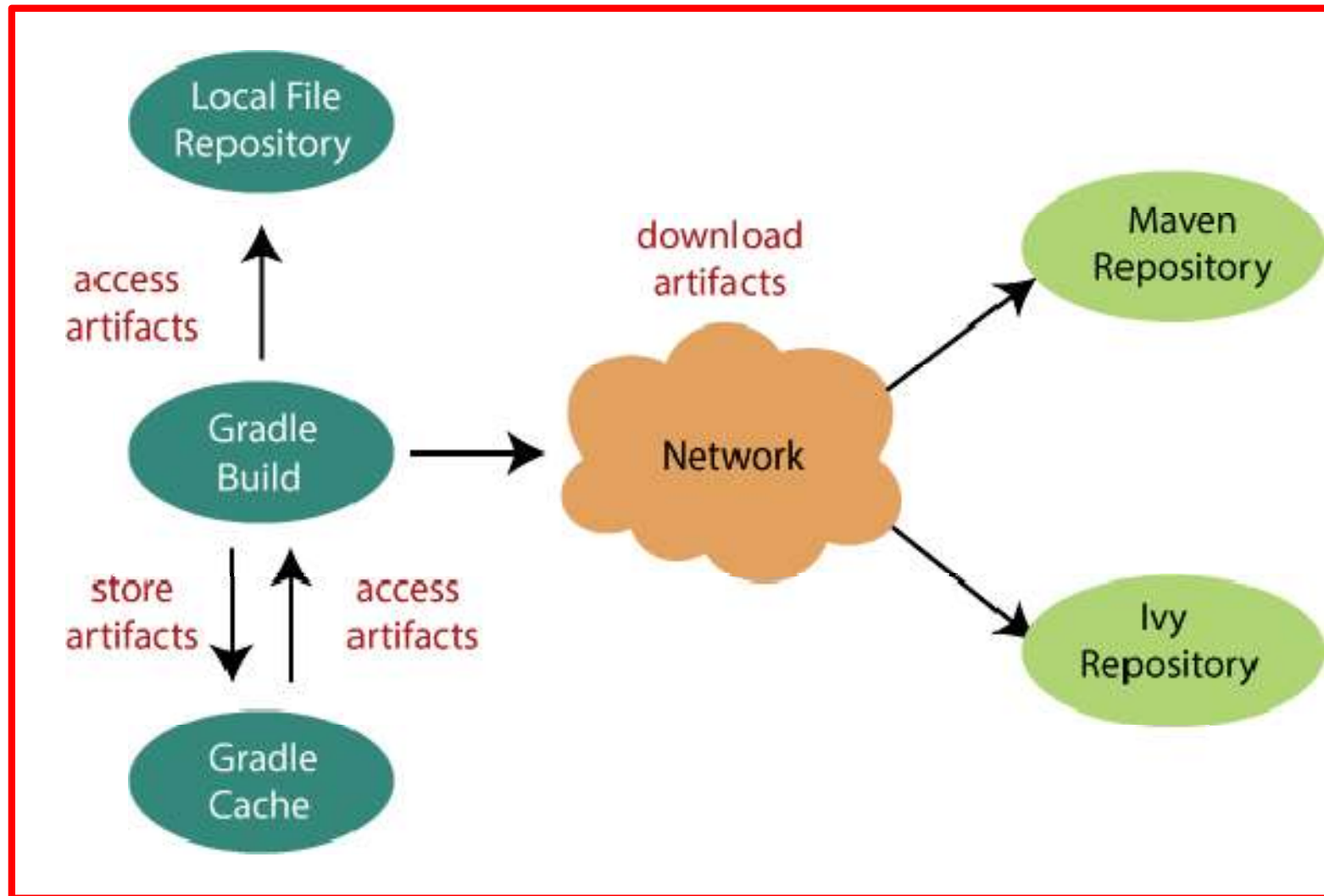
# Resolving Dependencies

- The plugin uses the configuration and gets input for the tasks defined. For example, the Gradle project uses the Spring Web Framework jar file, which must be downloaded from Maven Central.

## Exposing Artifacts for Consumption

- The plugin uses configurations that are used to define artifacts for other projects consumption.

# Dependency Management





# Gradle Projects & Tasks

# Gradle Tasks

- In Gradle, Task is a single unit of work that a build performs. **These tasks can be compiling classes, creating a JAR, Generating Javadoc, and publishing some archives** to a repository and more. It can be considered as a single atomic piece of work for a build process.
- Gradle's strength lies in his extensible model; tasks are the heart of this model. A task is a self-contained unit by which the Gradle functions. The essence of a task is its action.
- For example, we can declare a task to compile Java source code or copy some data from one directory to another directory. A task can perform some action in isolation, but we can also declare dependencies on other tasks. A task can also define its input and the output of the file from which it reads and writes. This allows Gradle to determine whether a task needs to be done.

# Gradle Projects

- In Gradle, A project displays a library JAR or a web application. It may also serve a distribution ZIP, which is assembled from the JARs produced by other projects.
- A project could be deploying our application to staging or production environments. Each project in Gradle is made up of one or more tasks.



- In Gradle, a project is a collection of one or more tasks. A project represents a library JAR or a web application. It may also serve a distribution ZIP, which is assembled from the JARs of different projects.
- A project could be deploying our application to staging or production environments. A project could be deploying our application to the staging or production environment. Every Gradle build contains one or more projects. We can relate it with an example of a building; it can have any numbers of floors.

# Types of Tasks

There are two types of tasks in Gradle. They are as following:

- Default task
- Custom task

# Default Tasks

- Default tasks are predefined tasks of Gradle. We can define it in our projects. Gradle let us define one or more default tasks that are executed if no other tasks are specified.
- We can list the default tasks by running the gradle task command. Consider the below output:

```
C:\Windows\System32\cmd.exe

F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>gradle task

> Task :tasks

-----
Tasks runnable from root project 'First_Gradle'
-----

Build Setup tasks
-----
init - Initializes a new Gradle build.
wrapper - Generates Gradle wrapper files.

Custom tasks
-----
copy - The sources are copied to dest directory

Help tasks
-----
buildEnvironment - Displays all buildscript dependencies declared in root project 'First_Gradle'.
dependencies - Displays all dependencies declared in root project 'First_Gradle'.
dependencyInsight - Displays the insight into a specific dependency in root project 'First_Gradle'.
help - Displays a help message.
javaToolchains - Displays the detected java toolchains.
```

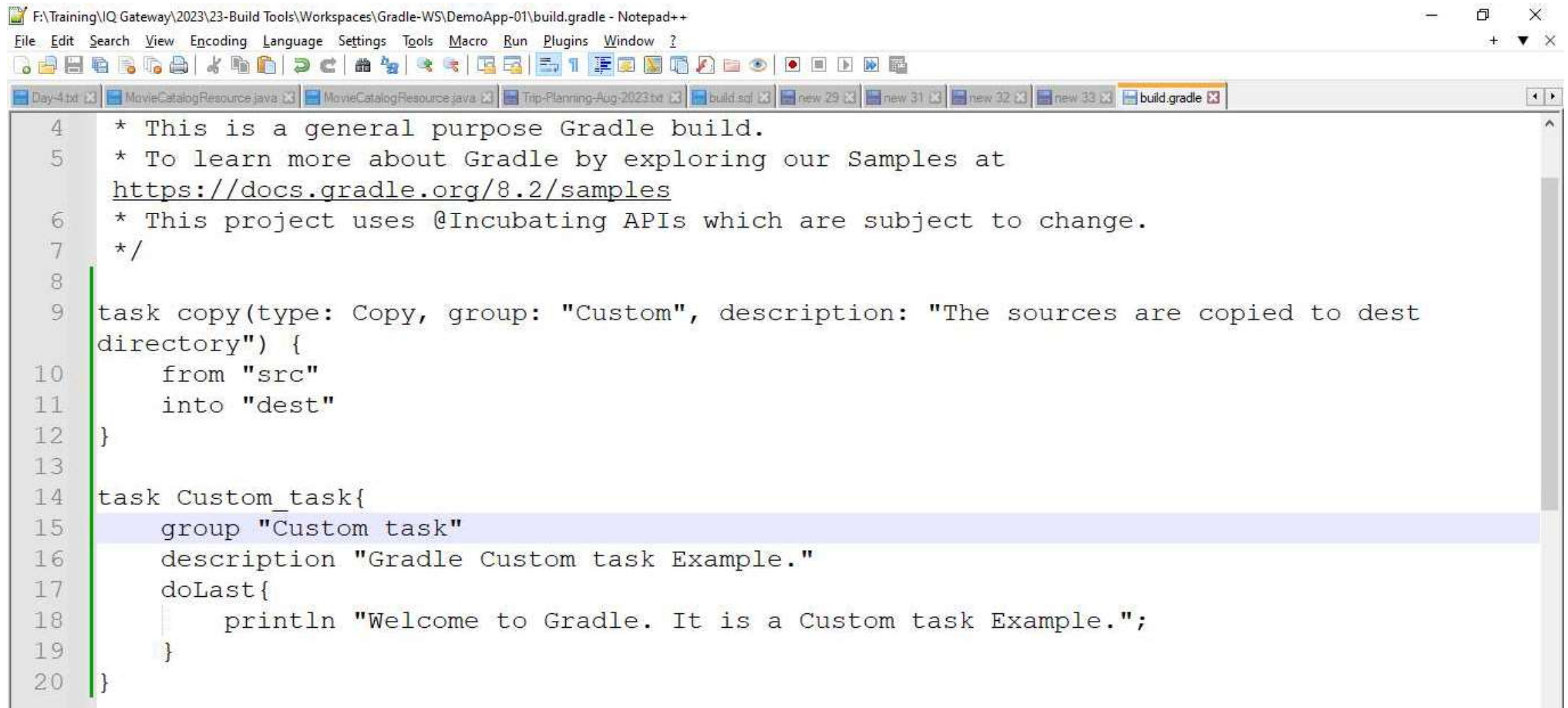
# Custom tasks

- Gradle allows us to create tasks; these tasks are called custom tasks. Custom tasks are user-defined tasks that are built to perform some specific work.

Syntax:

```
Task task_name{  
    group "-----group_name for task-----"  
    description '-----description of the task-----'  
    doLast{  
        -----code for execution-----  
        -----  
        -----  
    }  
}
```

## Add task code in build.gradle file.



```
4  * This is a general purpose Gradle build.
5  * To learn more about Gradle by exploring our Samples at
   https://docs.gradle.org/8.2/samples
6  * This project uses @Incubating APIs which are subject to change.
7  */
8
9  task copy(type: Copy, group: "Custom", description: "The sources are copied to dest
   directory") {
10     from "src"
11     into "dest"
12 }
13
14 task Custom_task{
15     group "Custom task"
16     description "Gradle Custom task Example."
17     doLast{
18         println "Welcome to Gradle. It is a Custom task Example.";
19     }
20 }
```

```
C:\Windows\System32\cmd.exe

F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>gradle tasks

> Task :tasks

-----
Tasks runnable from root project 'First_Gradle'
-----

Build Setup tasks
-----
init - Initializes a new Gradle build.
wrapper - Generates Gradle wrapper files.

Custom tasks
-----
copy - The sources are copied to dest directory

Custom task tasks
-----
Custom_task - Gradle Custom task Example.

Help tasks
-----
buildEnvironment - Displays all buildscript dependencies declared in root project 'First_Gradle'.
```



```
C:\Windows\System32\cmd.exe

F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>gradle Custom_task

> Task :Custom_task
Welcome to Gradle. It is a Custom task Example.

BUILD SUCCESSFUL in 717ms
1 actionable task: 1 executed
F:\Training\IQ Gateway\2023\23-Build Tools\Workspaces\Gradle-WS\DemoApp-01>
```



## Adding Dependencies to Tasks

- You can make a task dependent on another task and that means, when one task is done then only other task will begin.
- Each task is differentiated with the task name. The collection of task names is referred by its tasks collection. To refer to a task in another project, you should use path of the project as a prefix to the respective task name.

```
task taskX << {  
    println 'taskX'  
}  
task taskY(dependsOn: 'taskX') << {  
    println "taskY"  
}
```

```
task taskY << {  
    println 'taskY'  
}  
task taskX << {  
    println 'taskX'  
}  
taskY.dependsOn taskX
```

```
C:\> gradle -q taskY
```

```
task taskX << {  
    println 'taskX'  
}  
taskX.dependsOn {  
    tasks.findAll {  
        task → task.name.startsWith('lib')  
    }  
}  
task lib1 << {  
    println 'lib1'  
}  
task lib2 << {  
    println 'lib2'  
}  
task notALib << {  
    println 'notALib'  
}
```

```
C:\> gradle -q taskX
```

## Adding a Description

```
task copy(type: Copy) {  
    description 'Copies the resource directory to the target directory.'  
    from 'resources'  
    into 'target'  
    include('**/*.txt', '**/*.xml', '**/*.properties')  
    println("description applied")  
}
```

# Summary

In this lesson, you should have learned how to:

- An Introduction to Gradle
- Installing Gradle
- Gradle Build
- Gradle Dependencies
- Gradle Repository
- Projects & Tasks

