**Creating a Gradle Project in IntelliJ IDEA**

**Step 1: Open IntelliJ IDEA and Create a New Project**

1. Click on "New Project".

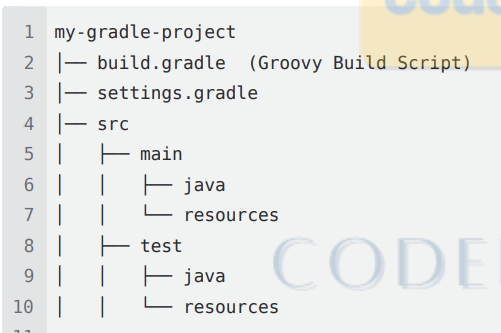
2. Select "Gradle" (under Java/Kotlin).

3. Choose Groovy or Kotlin DSL (Domain Specific Language) for the build script.

4. Set the Group ID (e.g., com.example ).

5. Click Finish.

**Step 2: Understanding Project Structure**

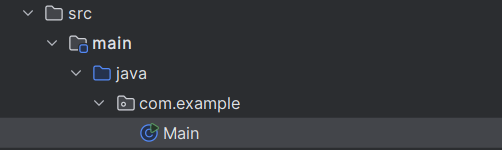


**Build and Run a Simple Java Application**

**Step 1: Modify build.gradle (Groovy DSL)**

plugins **{** id 'application'  
**}**repositories **{** mavenCentral()  
**}**dependencies **{** testImplementation 'org.junit.jupiter:junit-jupiter:5.8.1'  
**}**task copyWebsite(type: Copy) **{** from 'src/main/resources'  
 into 'docs'  
**}**tasks.named('run')**{** dependsOn 'copyWebsite'  
**}**application **{** mainClass = 'com.example.Main'  
**}**jar **{** manifest **{** attributes 'Main-Class': 'com.example.Main' // This tells Java where to start execution  
 **}  
}**

**Step 2: Create Main.java in src/main/java/com/example**

****

**package com.example;  
  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("Hello, World!");  
 }  
}**

**Step 3: Build and Run the Project**

**In IntelliJ IDEA, open the Gradle tool window (View → Tool Windows → Gradle).**

**Click Tasks > application > run .**

**Or run from terminal:**

**gradle run**

**󾠲 Hosting a Static Website on GitHub Pages**

**Step 1: Create a /docs Directory Create docs inside the root folder (not in src ).**

**Add your HTML, CSS, and images inside /docs .**

**Step 2: Modify build.gradle to Copy Website Files (This is optional)**

**task copyWebsite(type: Copy) {  
 from 'src/main/resources'  
 into 'docs'  
}**

**Step 3: Commit and Push to GitHub**

**1.git add .**

**2 git commit -m "Deploy website using Gradle"**

**3 git push origin main**

**Step 4: Enable GitHub Pages Go to GitHub Repo → Settings → Pages. Select the /docs folder as the source.**

**Your website will be hosted at**

****

**Testing the Website using Selenium & TestNG in IntelliJ IDEA Step 1: Add Selenium & TestNG Dependencies in build.gradle**

**Step 1: Add Selenium & TestNG Dependencies in build.gradle**

**dependencies {  
 testImplementation 'org.junit.jupiter:junit-jupiter:5.8.1'  
}**

**test {**

**useTestNG()**

**}**

**Step 2: Write a Test Script ( src/test/java/org/test/WebpageTest.java )**

**import static org.testng.Assert.assertTrue;**

**public class WebpageTest {**

**private static WebDriver driver;**

**@BeforeTest**

**public void openBrowser() throws InterruptedException {**

**driver = new ChromeDriver();**

**driver.manage().window().maximize();**

**Thread.sleep(2000);**

**driver.get("https://sauravsarkar-codersarcade.github.io/CA-GRADLE/");**

**}**

**@Test 24 public void titleValidationTest(){**

**String actualTitle = driver.getTitle();**

**String expectedTitle = "Tripillar Solutions";**

**Assert.assertEquals(actualTitle, expectedTitle);**

**assertTrue(true, "Title should contain 'Tripillar'");**

**}**

**@AfterTest**

**public void closeBrowser() throws InterruptedException {**

**Thread.sleep(1000);**

**driver.quit();**

**}**

**}**

**Step 3: Run the Tests Open the Gradle tool window in IntelliJ.**

**Click Tasks > verification > test . “Recommended“ Or**

**run from terminal:**

**gradle test // Fails sometimes due to terminal issues**

**Packaging a Gradle Project as a JAR**

**Step 1: Modify build.gradle for JAR Packaging**

**application {  
 mainClass = 'com.example.Main'  
}  
jar {  
 manifest {  
 attributes 'Main-Class': 'com.example.Main' // This tells Java where to start execution  
 }  
}**

**Step 2: Build and Package the JAR**

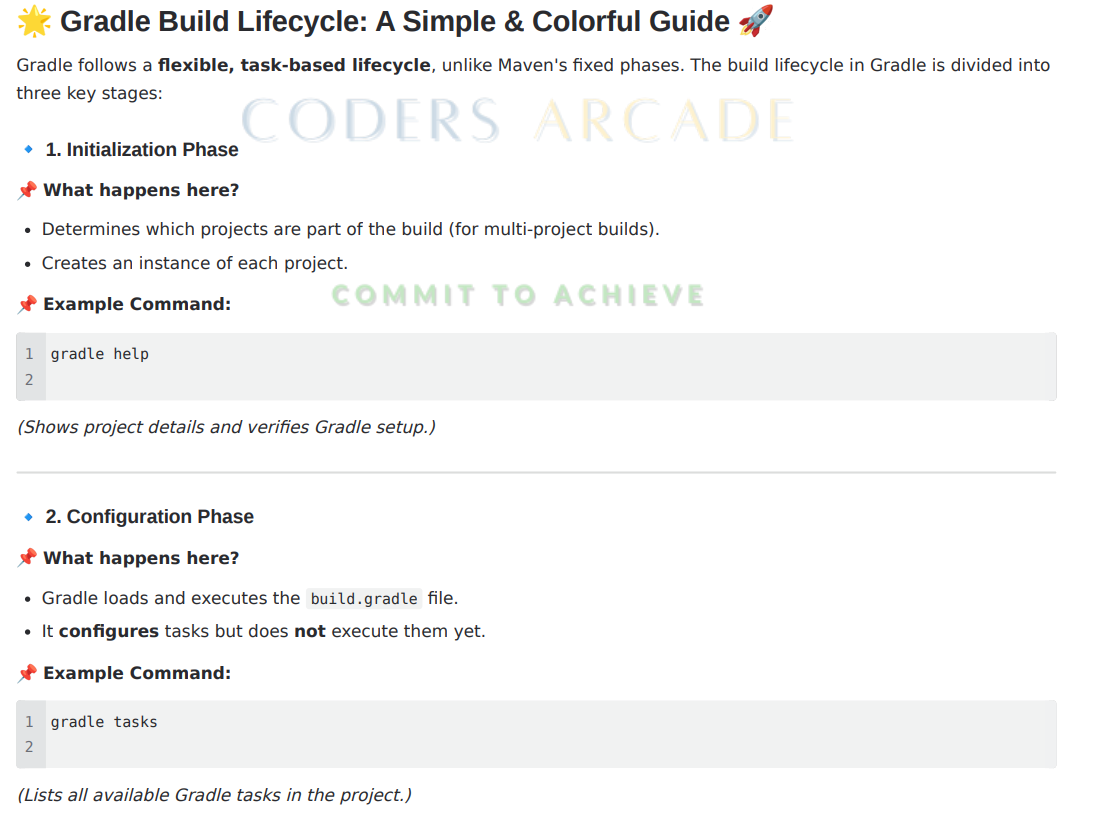
**gradle jar**

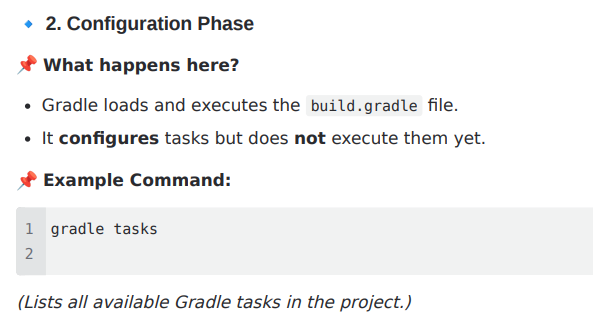
**Step 3: Run the JAR**

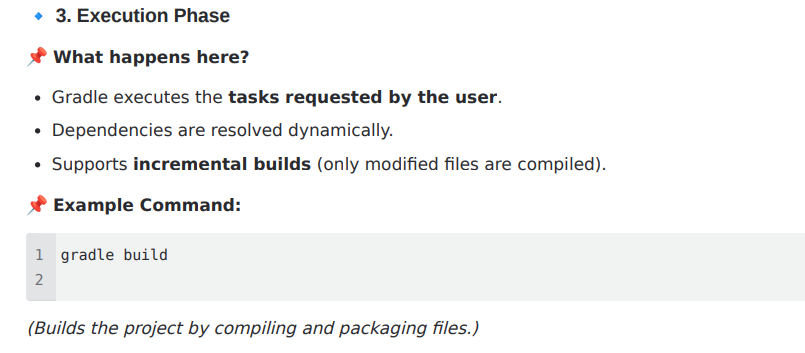
****

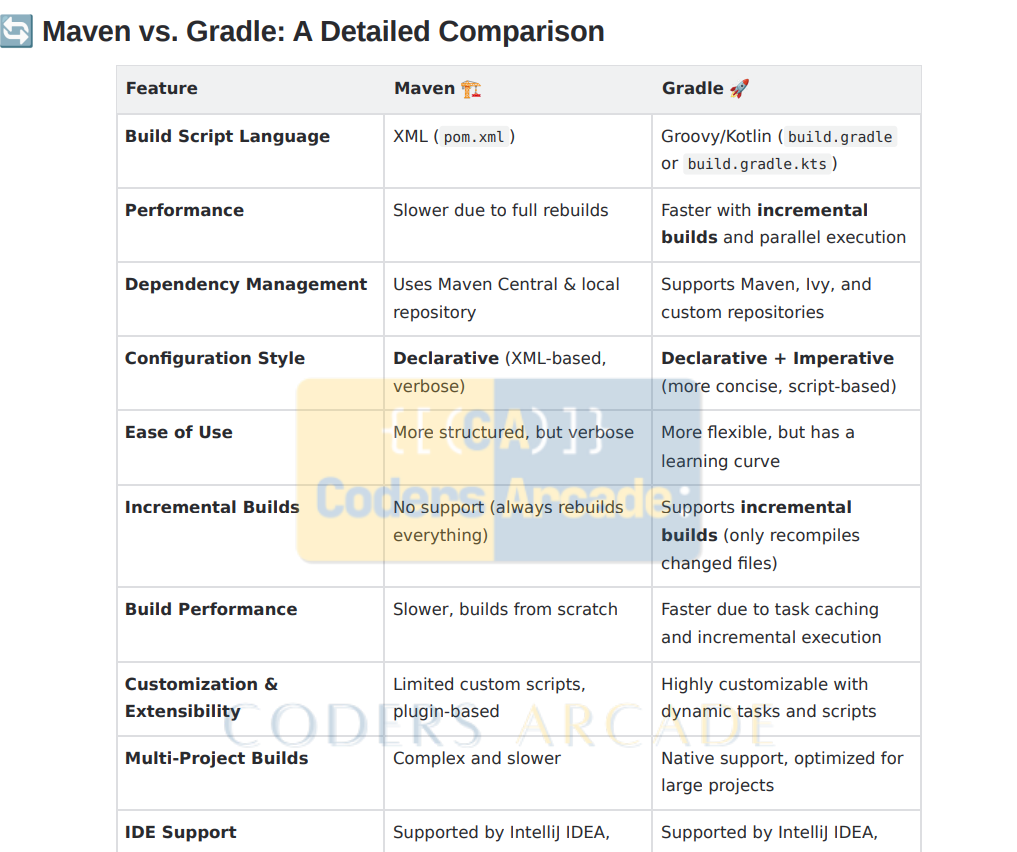
**Expected output:**

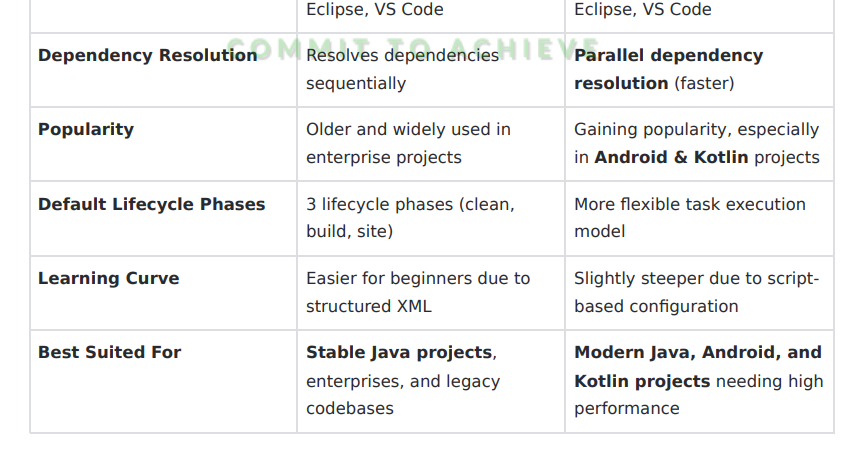
**Hello from Gradle!**

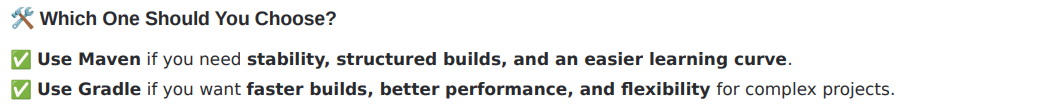
****

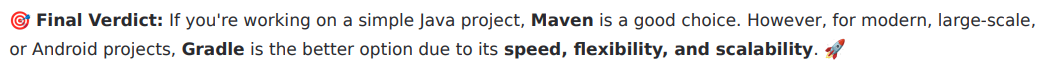
****

****

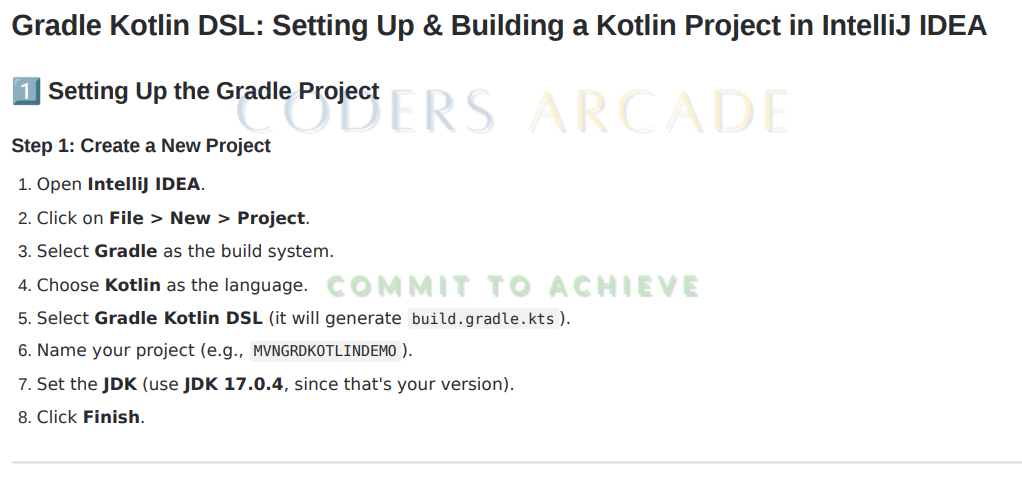
****

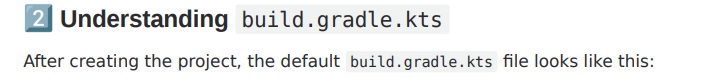
****

****

****

**CA - Experiment 2 Part 2 : GRADLE KOTLIN DSL WORKFLOW || IntelliJ Idea**

****

****

**import org.jetbrains.kotlin.gradle.tasks.KotlinCompile**

**plugins {**

**kotlin("jvm") version "1.8.10" // Use latest stable Kotlin version 5 application**

**}**

**group = "org.example"**

**version = "1.0-SNAPSHOT"**

**repositories {**

**mavenCentral()**

**}**

**dependencies {**

**implementation(kotlin("stdlib")) // Kotlin Standard Library**

**testImplementation("org.junit.jupiter:junit-jupiter-api:5.8.2") testRuntimeOnly("org.junit.jupiter:junit-jupiter-engine:5.8.2")**

**}**

**tasks.test {**

**useJUnitPlatform() }**

**tasks.withType {**

**kotlinOptions.jvmTarget = "17" // Match with your JDK version**

**} application { mainClass.set("MainKt") // Update this if using a package**

**}**

**Creating the Main Kotlin File Now, create your Main.kt file inside src/main/kotlin/ . If you're using a package (e.g., org.example ), it should look like**

**package org.example**

**fun main() {**

**println("Hello, Gradle with Kotlin DSL!")**

**}**

**Building and Running the Project Build the Project**

**./gradlew build**

**Run the Project**

**./gradlew run**

**Packaging as a JAR**

**To run the project without IntelliJ, we need a JAR file.**

**Step 1: Create a Fat (Uber) JAR Modify build.gradle.kts :**

**tasks.register("fatJar")**

**{ archiveClassifier.set("all")**

**duplicatesStrategy = DuplicatesStrategy.EXCLUDE**

**manifest {**

**attributes["Main-Class"] = "MainKt"**

**}**

**from(configurations.runtimeClasspath.get().map { if (it.isDirectory) it else zipTree(it) })**

**with(tasks.jar.get() as CopySpec)**

**}**

**Step 2: Build the Fat JAR**

**./gradlew fatJar**

**Step 3: Run the Fat JAR**

**java -jar build/libs/MVNGRDKOTLINDEMO-1.0-SNAPSHOT-all.jar**

**Additional Features Adding Custom Gradle Tasks Example: A simple task that prints "Hello, Gradle!"**

**tasks.register("hello") {**

**doLast {**

**println("Hello, Gradle!")**

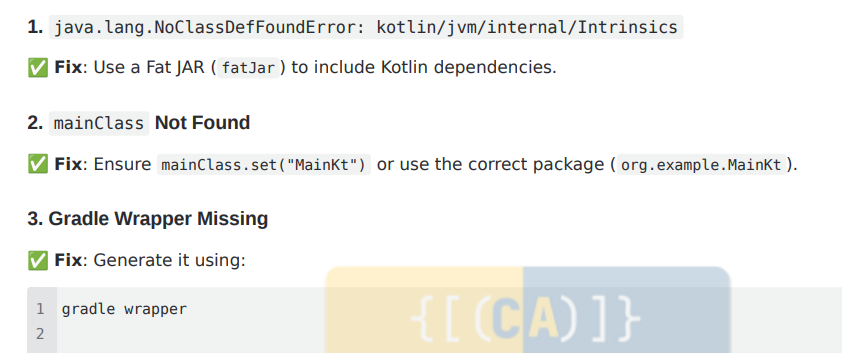
**}**

**}**

**Run it with:**

**.** **/gradlew hello**

**Troubleshooting & Fixes**

****