

Bootcamp Java Profesional

Gradle

Javier Ramírez

Software Professional since 2001

Java Developer / Tech Lead

@javaMexico Founder/Co-Leader



@_benek | @javamexico

linktree.com/javamexico



Source Code

- El código fuente de esta clase estará disponible en:
- <https://github.com/benek/Bootcamp-Java-CodigoFacilito>

¿Qué es Gradle?

- Herramienta de construcción genérica
- Nos ayuda a construir proyectos de cualquier tamaño y complejidad
- Provee un lenguaje (DSL) para configurar y declarar lógica de construcción programáticamente
- La herramienta más utilizada para Java, junto con Maven
- Aunque puede construir proyectos en otros lenguajes

Turing complete

- Gradle es considerado “Turing complete”, lo que implica que se puede resolver cualquier problema computacional con esta herramienta
- Es una de las principales diferencias con Maven
- Los “build scripts” son efectivamente “programas” Groovy o Kotlin, y como tal cuentan con estructuras de control de flujo, loops, condiciones, etc...
- Esto puede ser un arma de dos filos

Herramientas similares

- Make
- Ant
- Maven
- Bazel
- Etc...

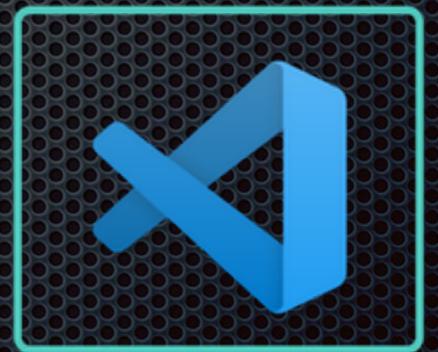
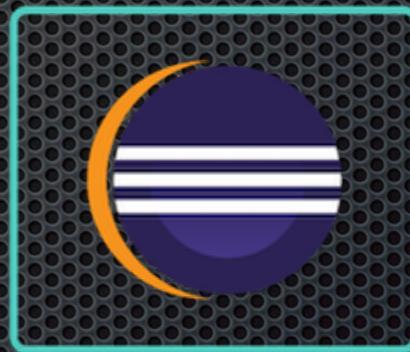
Gradle

- Soportado en:



Gradle

- Puede correr en la CLI y en los principales IDEs



Core Concepts

Gradle Core Concepts

- Project
 - Single Project - root project
 - Multi-Project - root project + subprojects
- Build Script
 - Uno o varios scripts que detallan los pasos para construir un proyecto

Gradle Core Concepts

- Dependency Management
 - Capacidad de una build tool para declarar y obtener recursos o bibliotecas externas a un proyecto
 - Casi cualquier proyecto tiene dependencias externas, Gradle nos ayuda a administrarlas
- Task
 - La unidad básica de trabajo en Gradle
- Plugins

Project Structure

```
project
  └── gradle
      ├── libs.versions.toml
      └── wrapper
          ├── gradle-wrapper.jar
          └── gradle-wrapper.properties
  └── gradlew
  └── gradlew.bat
  └── settings.gradle(.kts)
  └── subproject-a
      ├── build.gradle(.kts)
      └── src
  └── subproject-b
      ├── build.gradle(.kts)
      └── src
```

1

2

3

3

4

5

6

5

6

1. Directory “gradle” para archivos del wrapper y extras
2. Catálogo de versiones para las dependencias del proyecto
3. Scripts del Gradle Wrapper
4. Configuración del nombre del proyecto y subproyectos
5. Build scripts de subproyectos “a” y “b”
6. Código fuente o archivos adicionales para el proyecto

Single y Multi-Project

Single

- `settings.gradle`
- `build.gradle`
 - `tasks`
- `source code`

VS

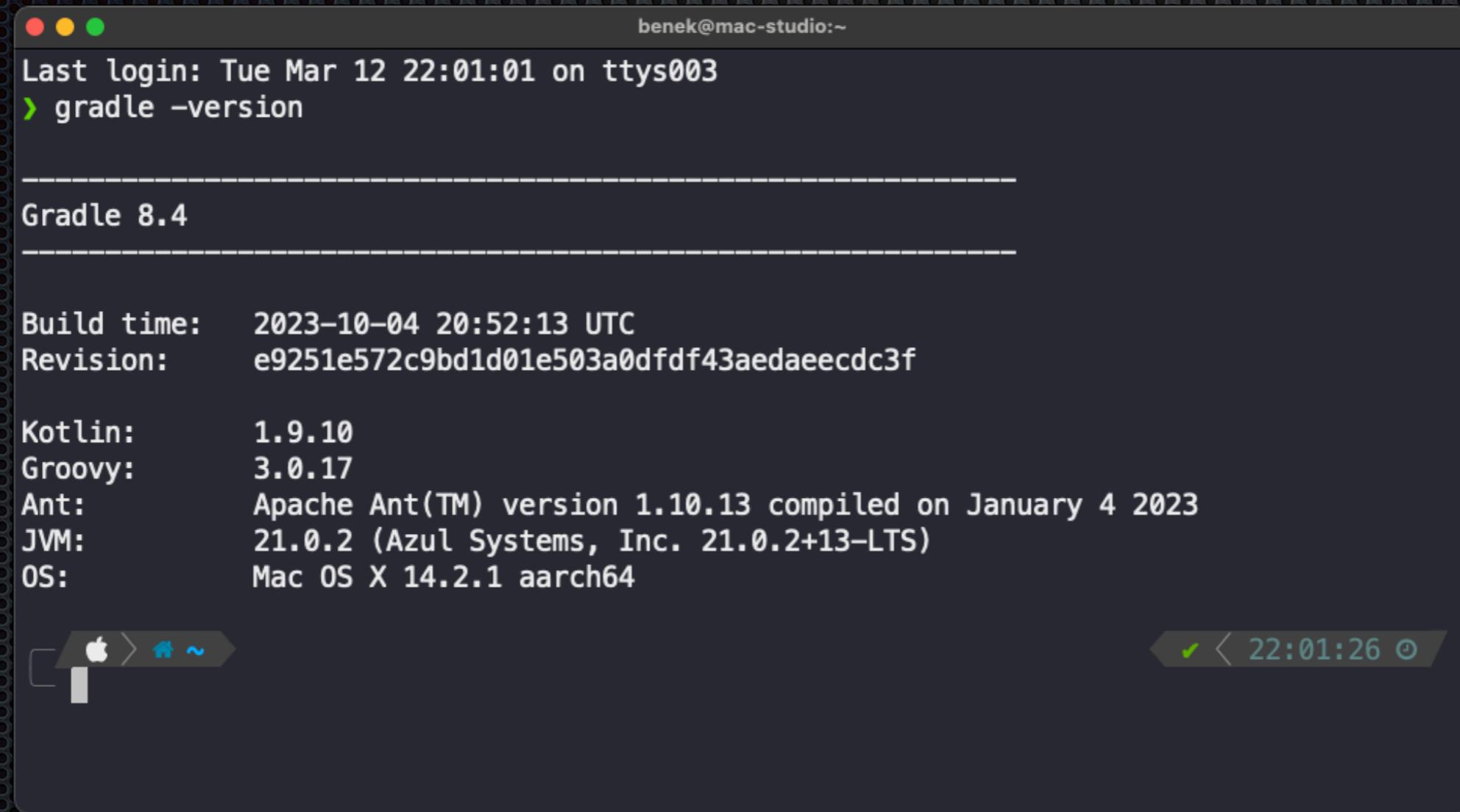
Multi

- `settings.gradle`
- `subproject-a`
 - `build.gradle`
 - `tasks`
 - `source code`
- `Subproject-b`
 - `build.gradle`
 - `tasks`
 - `source code`

Instalación

- Disponible en Linux, MacOS, Windows
- Vía SDKMan, Homebrew o ZIP
- Integrado en IntelliJ IDEA y Android Studio
- Disponible en VSCode mediante extensión

Primer uso



A screenshot of a macOS terminal window titled "benek@mac-studio:~". The window shows the output of the command "gradle -version". The output includes the Gradle version (8.4), build time (2023-10-04 20:52:13 UTC), revision (e9251e572c9bd1d01e503a0dfdf43aedaeecd3f), and various system details like Kotlin (1.9.10), Groovy (3.0.17), Ant (Apache Ant(TM) version 1.10.13 compiled on January 4 2023), JVM (21.0.2 (Azul Systems, Inc. 21.0.2+13-LTS)), and OS (Mac OS X 14.2.1 aarch64). The terminal has a dark theme with light-colored text. The status bar at the bottom right shows the date and time (22:01:26).

```
Last login: Tue Mar 12 22:01:01 on ttys003
> gradle -version

-----
Gradle 8.4

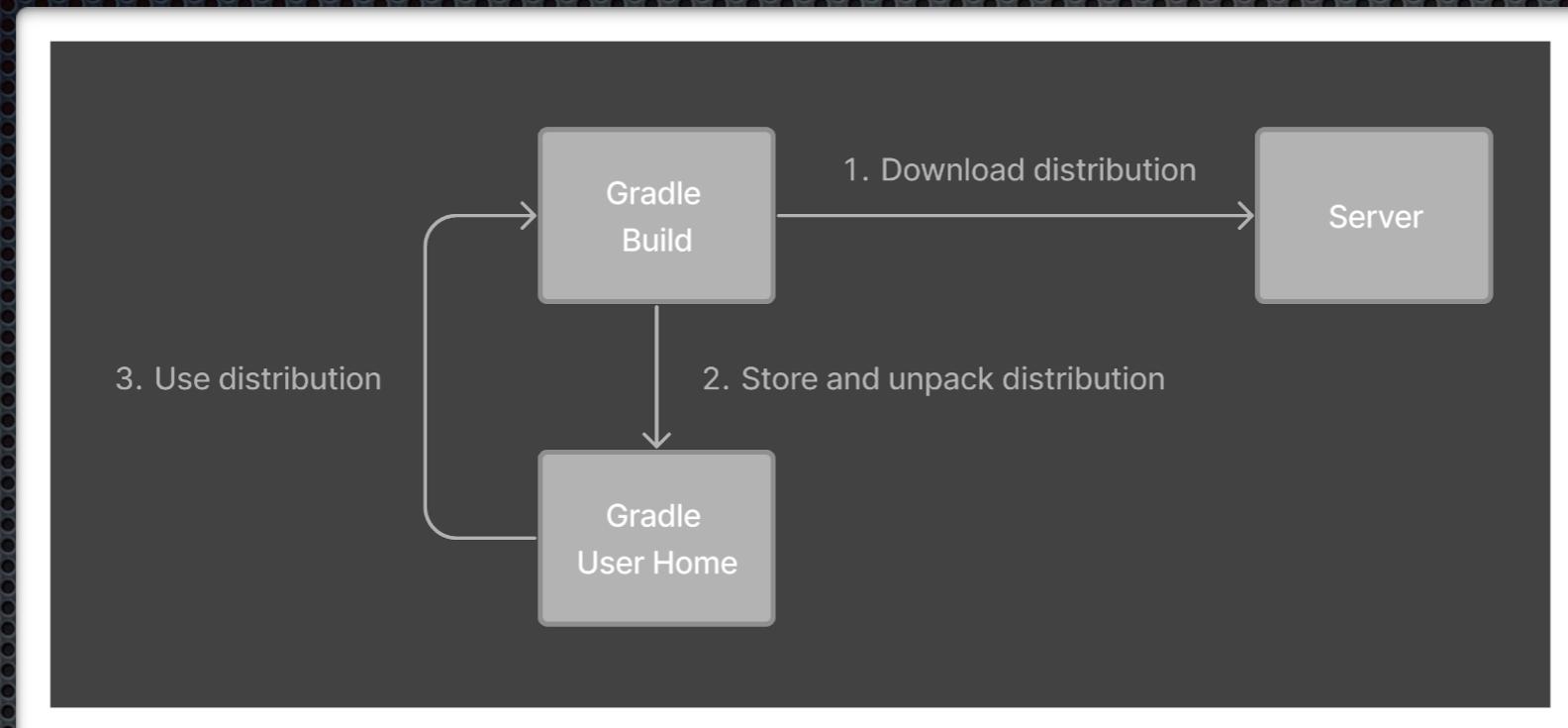
-----
Build time:      2023-10-04 20:52:13 UTC
Revision:        e9251e572c9bd1d01e503a0dfdf43aedaeecd3f

Kotlin:          1.9.10
Groovy:          3.0.17
Ant:             Apache Ant(TM) version 1.10.13 compiled on January 4 2023
JVM:              21.0.2 (Azul Systems, Inc. 21.0.2+13-LTS)
OS:               Mac OS X 14.2.1 aarch64
```

Gradle Wrapper

- Scripts nativo del OS para invocar Gradle, aún si éste no está instalado
- Es el medio recomendado para ejecutar Gradle en un proyecto existente
- `gradlew` para *nix, o `gradlew.bat` para Windows

Gradle Wrapper



- Estandariza la versión de Gradle para el proyecto, evitando que cada usuario tenga una versión diferente
- Simplifica la construcción de un proyecto en cualquier lugar, sin necesidad de instalación previa

Gradle Wrapper

- En lugar de:
 - `gradle build`
- En Windows (PowerShell):
 - `.\gradlew.bat build`
- En Linux o MacOS
 - `./gradlew build`

Gradle Wrapper

```
$ gradlew.bat build
```

```
Downloading https://services.gradle.org/distributions/gradle-5.0-all.zip
```

```
.....  
..
```

```
Unzipping C:\Documents and Settings\User\.gradle\wrapper\dists\gradle-5.0-  
all\ac27o8rbd0ic8ih41or9132mv\gradle-5.0-all.zip to C:\Documents and  
Settings\User\.gradle\wrapper\dists\gradle-5.0-all\ac27o8rbd0ic8ih41or9132mv
```

```
Set executable permissions for: C:\Documents and  
Settings\User\.gradle\wrapper\dists\gradle-5.0-  
all\ac27o8rbd0ic8ih41or9132mv\gradle-5.0\bin\gradle
```

```
BUILD SUCCESSFUL in 12s
```

```
1 actionable task: 1 executed
```

Command-Line Interface

- Estructura de ejecución de comandos:
 - `gradle [task1...]` `[--opcion]`
- O:
 - `gradle` `[--opcion]` `[task1...]`
- Ejemplo:
 - `gradle` `taskName` `--opcion=valor`

Settings Script

- **settings.gradle**
- Opcional para single-project builds
- Requerido para multi-project, ya que en él se definen los subproyectos

Settings Script

- Ejemplo:

```
1 rootProject.name = 'Project Name'  
2  
3 include('sub-project-a')  
4 include('sub-project-b')  
5 include('sub-project-c')  
6 |
```

Build Scripts

- Cada “build” en Gradle consta de al menos un build script
- Se puede escribir en los lenguajes Groovy, o Kotlin
 - Groovy: **build.gradle**
 - Kotlin: **build.gradle.kts**

Build Scripts

- Ejemplo básico:

```
1  plugins {  
2      id 'application'  
3  }  
4  
5  application {  
6      mainClass = 'com.example.Main'  
7  }  
8
```

Plugins

- Gradle es extensible mediante Plugins
- Un plugin puede agregar tasks a un proyecto, o también métodos y propiedades

Plugin types

- Core plugins - Desarrollados y mantenidos por Gradle
 - https://docs.gradle.org/current/userguide/plugin_reference.html#plugin_reference
- Community plugins - Desarrollados por terceros, y disponibles en repositorios remotos como el de Maven o el portal de plugins de Gradle
 - <https://plugins.gradle.org/>
- Local plugins - Custom plugins desarrollados por nosotros

Aplicando plugins

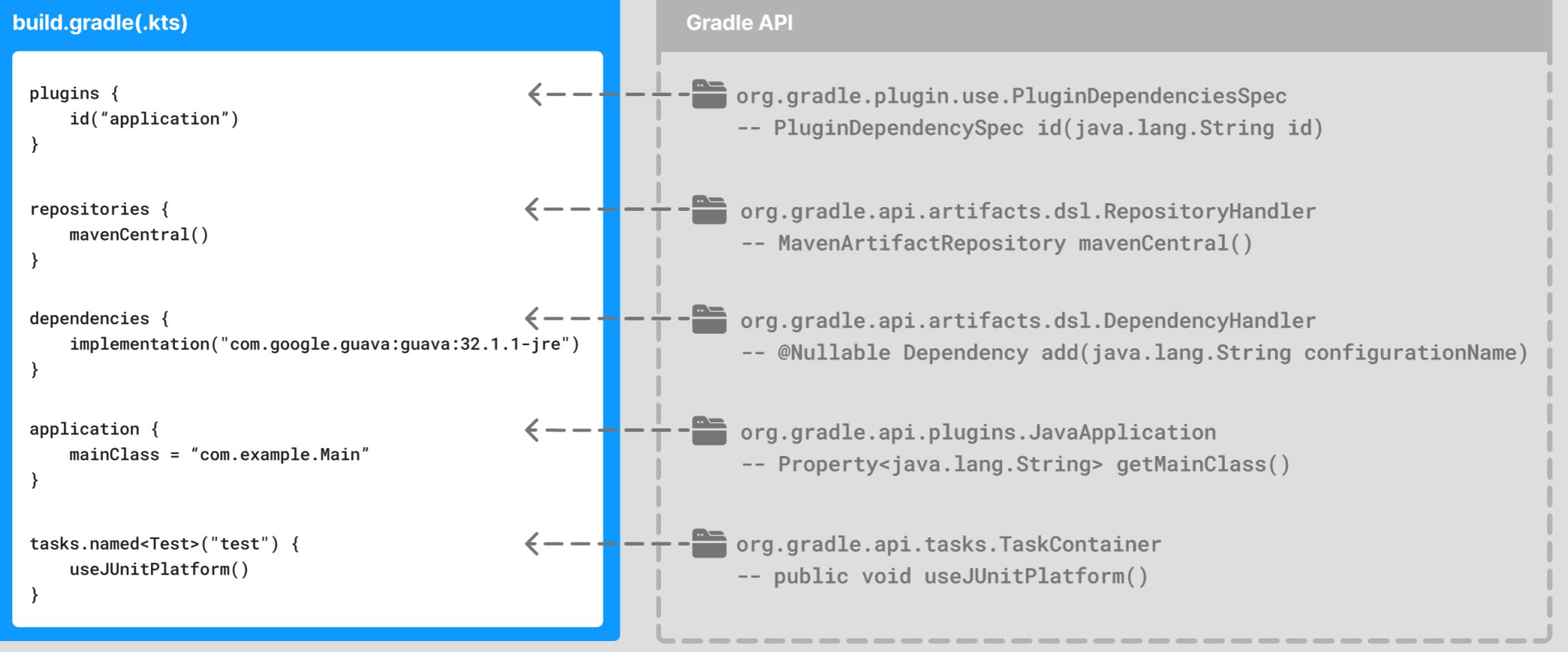
- Por nombre o por id (preferido)

```
8  plugins {  
7      application                      // by name  
6      java                            // by name  
5      id("java")                      // by id - recommended  
4      id("org.jetbrains.kotlin.jvm") version "1.9.0" // by id - recommended  
3  }  
2
```

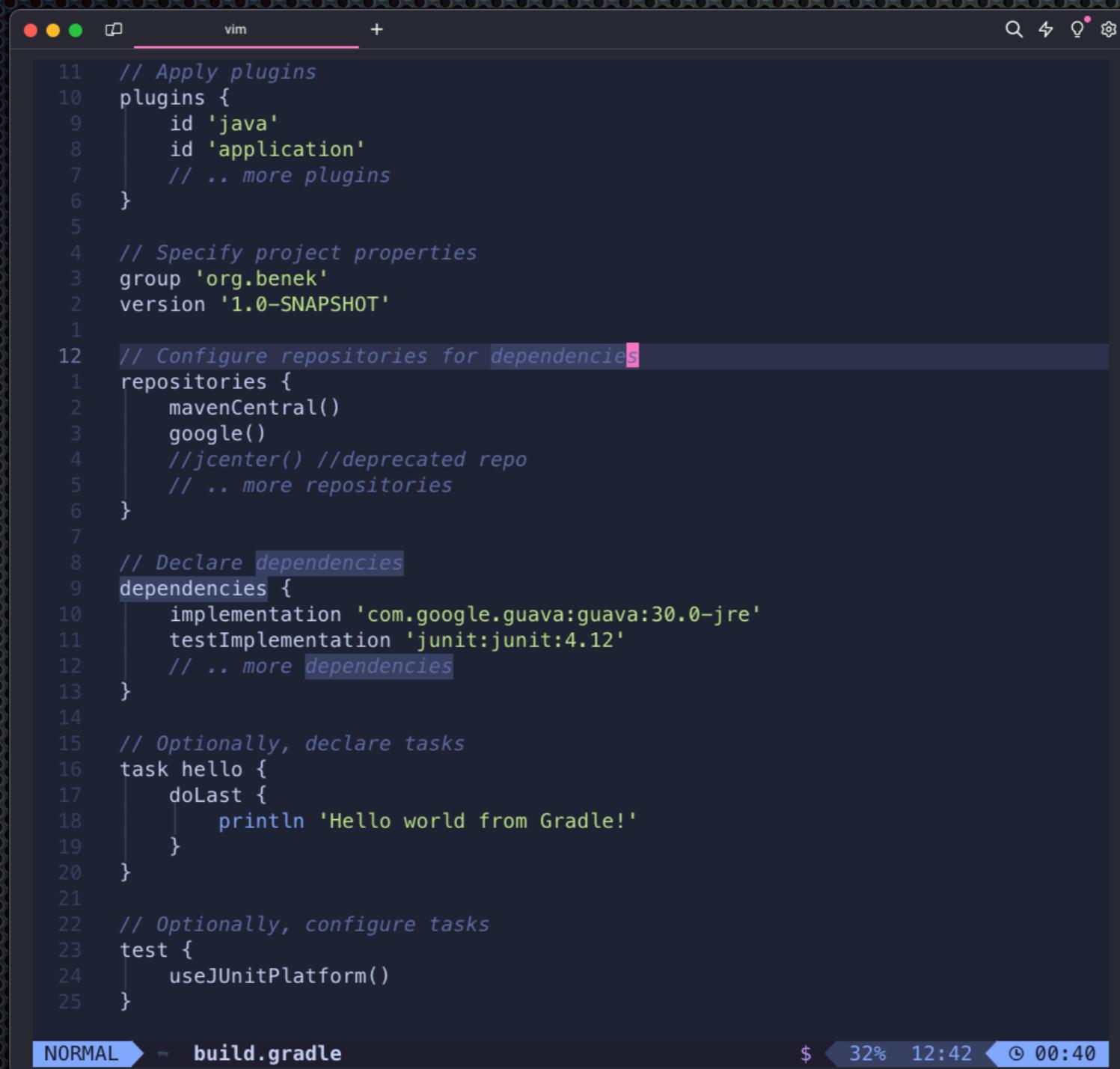
Fase de Inicialización

- Gradle rastrea el ***root project*** y **subproyectos** presentes en el archivo de configuración ***settings.gradle***
- Por cada proyecto encontrado, Gradle busca su correspondiente build script ***build.gradle***, para posteriormente usarlo en la fase de configuración

Estructura de Build Scripts



Estructura de Build Scripts



A screenshot of a vim editor window displaying a build.gradle file. The code is a Gradle build script with the following structure:

```
11 // Apply plugins
10 plugins {
9     id 'java'
8     id 'application'
7     // .. more plugins
6 }
5
4 // Specify project properties
3 group 'org.benek'
2 version '1.0-SNAPSHOT'
1
12 // Configure repositories for dependencies
1 repositories {
2     mavenCentral()
3     google()
4     //jcenter() //deprecated repo
5     // .. more repositories
6 }
7
8 // Declare dependencies
9 dependencies {
10    implementation 'com.google.guava:guava:30.0-jre'
11    testImplementation 'junit:junit:4.12'
12    // .. more dependencies
13 }
14
15 // Optionally, declare tasks
16 task hello {
17     doLast {
18         println 'Hello world from Gradle!'
19     }
20 }
21
22 // Optionally, configure tasks
23 test {
24     useJUnitPlatform()
25 }
```

The vim status bar at the bottom shows: NORMAL ➤ ~ build.gradle \$ 32% 12:42 ⏴ 00:40

The “Project” object

- Cada proyecto y subproyecto detectado por Gradle, se instancia en un objeto “Project” de la API de Gradle
- Éste puede tener varias propiedades asociadas:

Table 1

Name	Type	Description
name	String	Nombre del directorio del proyecto
path	String	Qualified name del proyecto
description	String	Descripción
dependencies	DependencyHandler	Dependency handler del proyecto
repositories	RepositoryHandler	Repository handler del proyecto
layout	ProjectLayout	Ubicaciones importantes del proyecto
group	Object	Grupo
version	Object	Versión

Definir propiedades

- Un plugin puede añadir propiedades y métodos para que nosotros los definamos

```
6 // Apply plugins
5 plugins {
4   id 'java'
3   id 'application'
2   // .. more plugins
1 }
7
```

```
4
3 application {
2   mainClass = 'org.benek.Main'
1 }
43
1
```

Registrar múltiples repositorios

```
18
17 // Configure repositories for dependencies
16 repositories {
15     mavenCentral()
14     google()

13
12     maven {
11         url "https://repo.spring.io/release"
10     }
9     maven {
8         url "https://repository.jboss.org/maven2"
7     }
6     ivy {
5         url "http://repo.mycompany.com/repo"
4     }
3     flatDir {
2         dirs 'lib1', 'lib2'
1     }
29     //jcenter() //deprecated repo
1     // .. more repositories
2 }
3 }
```

Definir repositorio propio

```
15      maven {
14          url "https://repo.spring.io/release"
13      }
12      maven {
11          url "https://repository.jboss.org/maven2"
10      }
9      ivy {
8          url "http://repo.mycompany.com/repo"
7          patternLayout {
6              artifact "[module]/[revision]/[type]/[artifact].[ext]"
5          }
4          credentials {
3              username "user"
2              password "password"
1          }
32      }
1      flatDir {
2          dirs 'lib1', 'lib2'
3      }
```

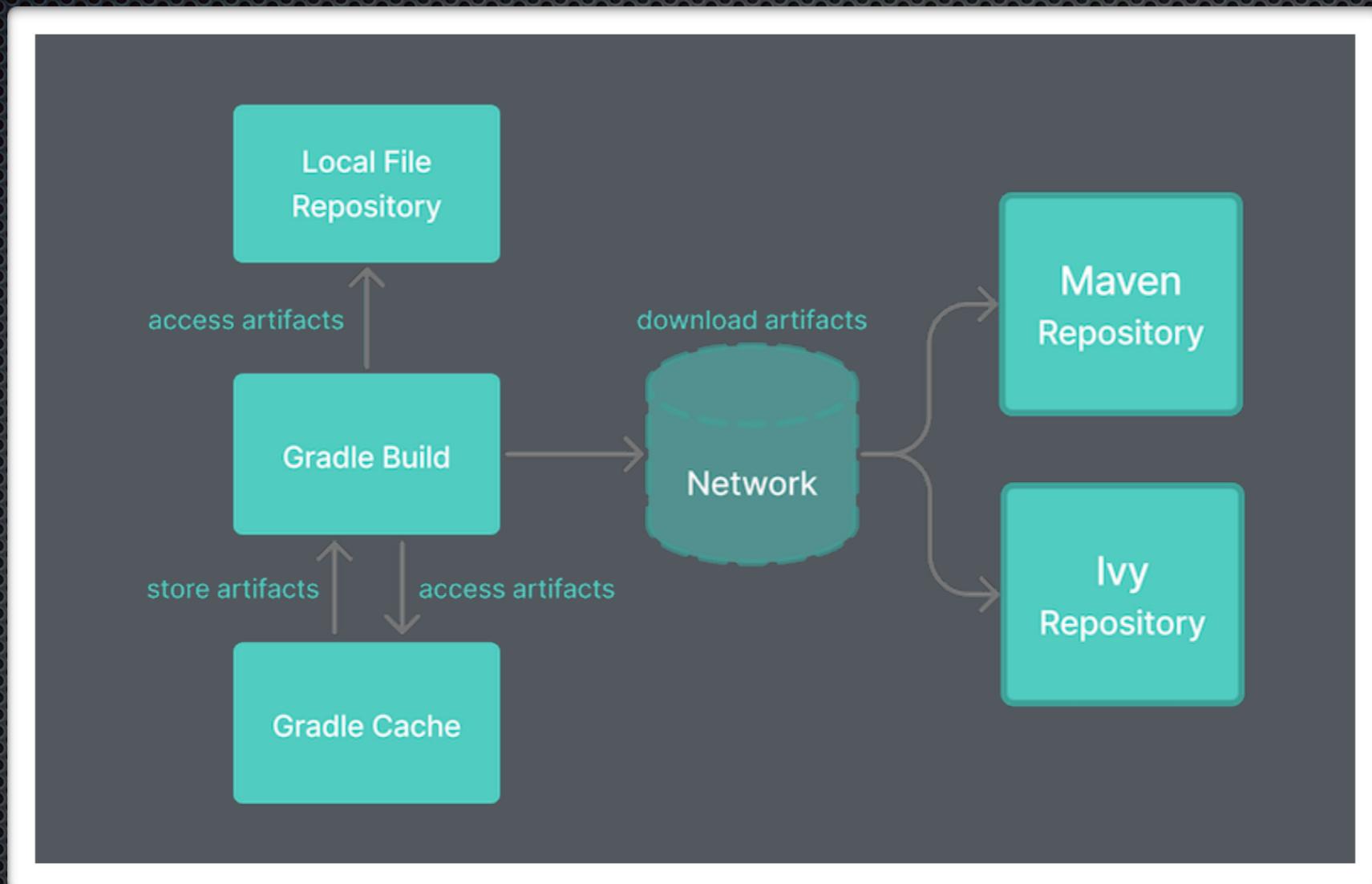
Otros métodos

- SFTP
- AWS S3
- Google Cloud Storage (GCS)

Dependency Management

- Los proyectos de software muy rara vez se desarrollan de manera aislada
- Casi siempre utilizan funcionalidad reutilizable de otras bibliotecas o frameworks
- Una “dependencia” es un apuntador hacia otra pieza de software que se requiere para construir, probar o ejecutar un módulo

Dependency Management



Scopes

implementation

Implementation only dependencies.

compileOnly

Compile time only dependencies, not used at runtime.

compileClasspath extends compileOnly, implementation

Compile classpath, used when compiling source. Used by task compileJava.

annotationProcessor

Annotation processors used during compilation.

runtimeOnly

Runtime only dependencies.

runtimeClasspath extends runtimeOnly, implementation

Runtime classpath contains elements of the implementation, as well as runtime only elements.

testImplementation extends implementation

Implementation only dependencies for tests.

testCompileOnly

Additional dependencies only for compiling tests, not used at runtime.

testCompileClasspath extends testCompileOnly, testImplementation

Test compile classpath, used when compiling test sources. Used by task compileTestJava.

testRuntimeOnly extends runtimeOnly

Runtime only dependencies for running tests.

testRuntimeClasspath extends testRuntimeOnly, testImplementation

Dependency Management

- Definiendo dependencias:

```
10  
9  dependencies {  
8      implementation 'org.springframework.boot:spring-boot-starter-web'  
7      implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
6      runtimeOnly 'com.h2database:h2'  
5      testImplementation('org.springframework.boot:spring-boot-starter-test') {  
4          exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'  
3      }  
2  }  
1  
53
```

Tasks

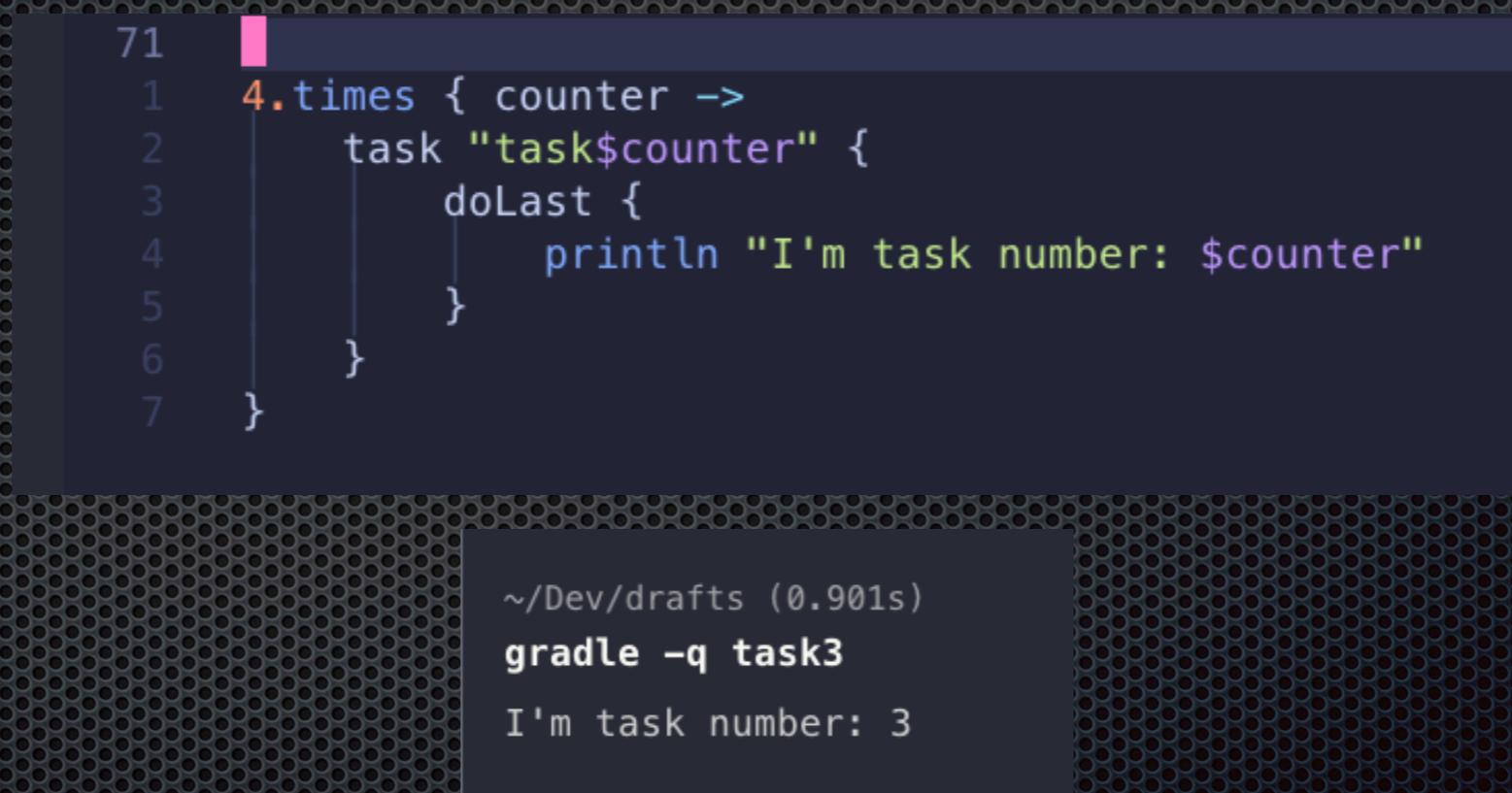
- Las tasks representan una pieza básica de trabajo, como por ejemplo “compilar” clases, “ejecutar” unit tests, o “empaquetar” con ZIP un archivo JAR o WAR
- Con frecuencia usaremos las tasks que ya vienen predefinidas en los plugins que necesitemos
- Aún así, Gradle nos permite crear tasks para nuestros propios propósitos de ser necesario

Tasks

```
19  
18 // Optionally, declare tasks  
17 task hello {  
16     doLast {  
15         println 'Hello world from Gradle!'  
14     }  
13 }  
12  
11 task helloUpper {  
10     doLast {  
9         String defaultName = 'javier benek'  
8         println "Original: $defaultName"  
7         println "New: ${defaultName.toUpperCase()}"  
6     }  
5 }  
4
```

El poder de Groovy y Kotlin

- Al Gradle utilizar Groovy y Kotlin para sus DSL, las capacidades que tenemos dentro de los build scripts son infinitas



```
71
1 4.times { counter ->
2     task "task$counter" {
3         doLast {
4             println "I'm task number: $counter"
5         }
6     }
7 }
```

~/Dev/drafts (0.901s)
gradle -q task3
I'm task number: 3

El poder de Groovy y Kotlin

```
79
1 class UserInfo {
2     String name
3     String email
4 }
5
6 task configure {
7     def user = configure(new UserInfo()) {
8         name = "Isaac Newton"
9         email = "isaac@newton.me"
10    }
11   doLast {
12       println user.name
13       println user.email
14    }
15 }
```

```
~/Dev/drafts (0.905s)
gradle -q configure
Isaac Newton
isaac@newton.me
```

Dos tipos de tasks

- Actionable tasks - tienen una acción concreta asociada, por ejemplo: `compileJava`
- Lifecycle tasks - no representan una acción, sino que funcionan como agrupadores de otras acciones, por ejemplo: `assemble`, o `build`
- Podemos listar las acciones disponibles en un proyecto con la task `tasks`
 - `gradle tasks --all`

Dependencias entre tasks

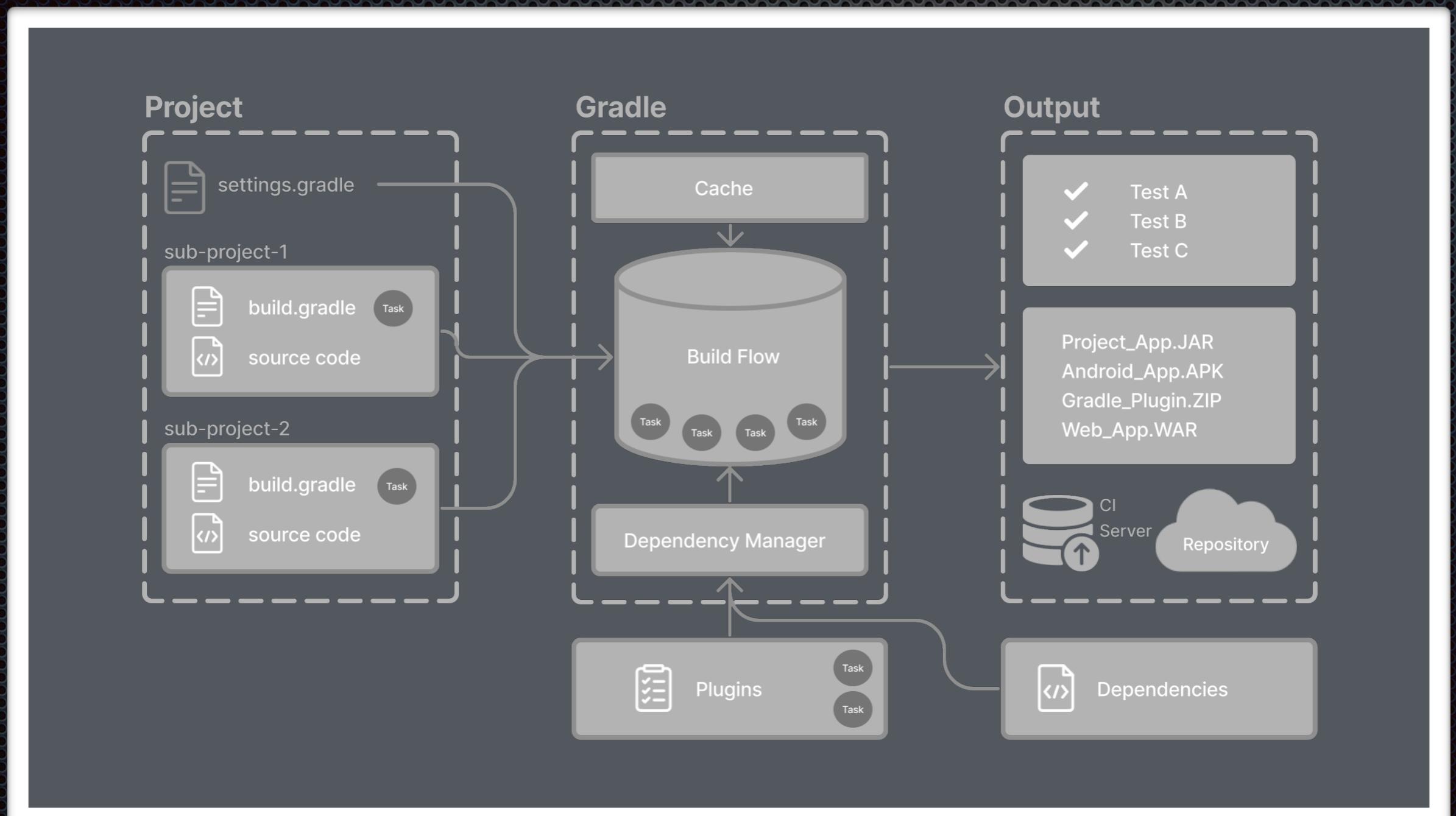
```
79
1 class UserInfo {
2     String name
3     String email
4 }
5
6 task configure {
7     dependsOn tasks.hello
8     def user = configure(new UserInfo()) {
9         name = "Isaac Newton"
10        email = "isaac@newton.me"
11    }
12    doLast {
13        println user.name
14        println user.email
15    }
16 }
```

```
~/Dev/drafts (1.086s)
gradle -q configure
Hello world from Gradle!
Isaac Newton
isaac@newton.me
```

Ejemplo real

```
99  //Docker
100 ext {
101     dockerTag = "benek/${project.name}:${project.version}".toLowerCase()
102     dockerBuildDir = mkdir("${buildDir}/docker")
103 }
104
105 > task prepareDocker(type: Copy, dependsOn: assemble) {
106     description = 'Copia archivos desde src/main/docker y application.jar al directorio temporal del build de Docker.'
107     group = 'Docker'
108
109     from 'src/main/docker'
110     from project.jar
111
112     into dockerBuildDir
113
114     from file('grails-app/assets/images/logo_constancia.png')
115     from file('grails-app/assets/stylesheets/color-admin/css/default/constanciaPDF.css')
116     into file(dockerBuildDir)
117 }
118
```

Recap



Killer features de Gradle

- Wrapper
- Modo continuo
- Gradle init
- Gradle build cache local y remota (*--build-cache*)
- Paralelismo (*--parallel --max-workers=8*)
- Configuration cache (incubation, *--configuration-cache*)
- Extensibilidad

Gradle init

- Plugin para crear un proyecto Gradle
- Similar a la generación de un archetype en Maven, pero de manera guiada y más sencilla

Modo continuo

- Soporte de Gradle para recompilación automática
- “Hot reloading”

Gradle Daemon

- Un “daemon” es un programa que corre en background
- Gradle usa un “daemon” para hacer más eficiente la ejecución de los builds
 - JVM startup time
 - Cache across builds
 - JVM runtime optimization

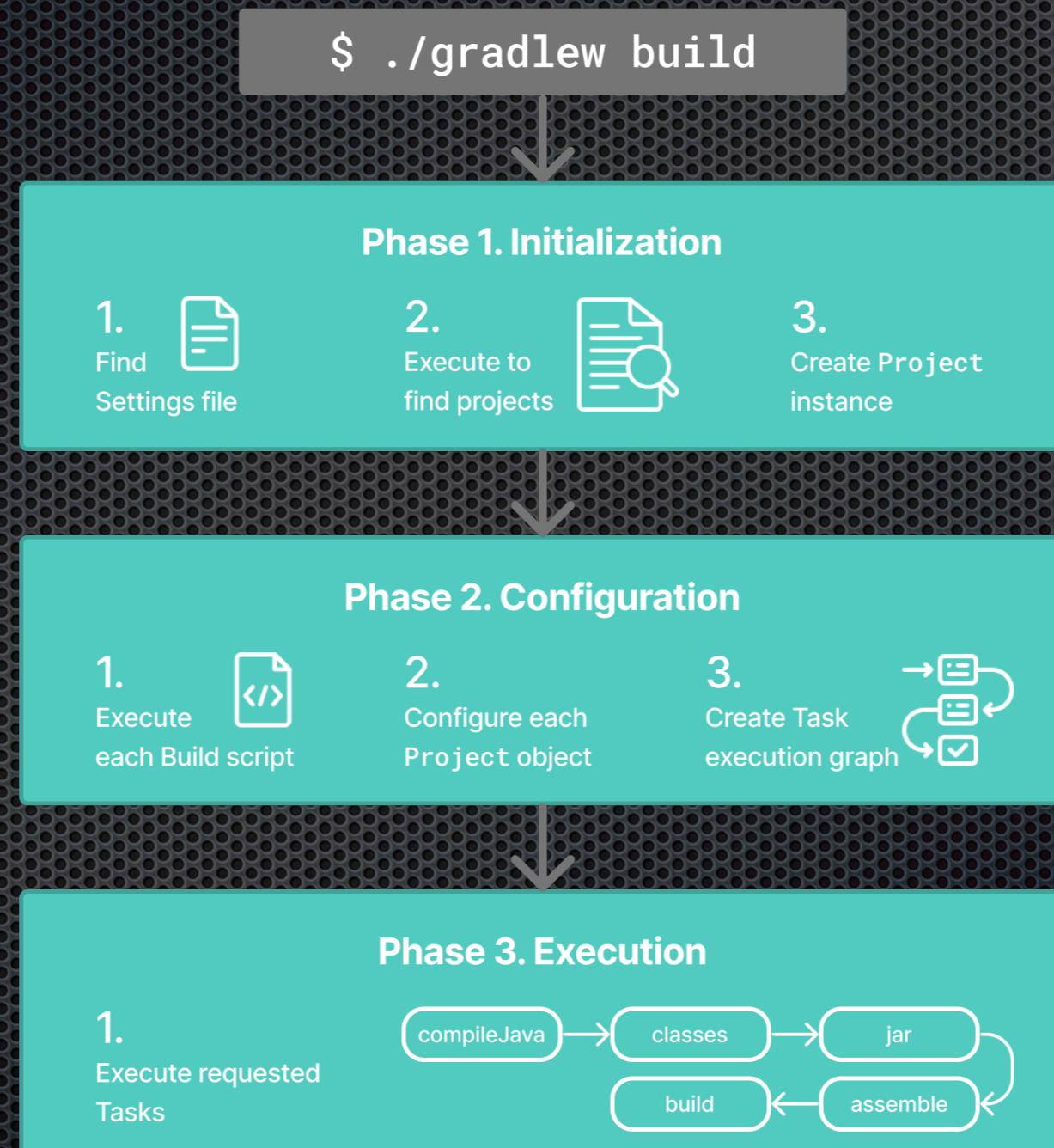
Mejores prácticas

- No solo se trata de construir y de que funcione
- Revisar calidad de código
 - Análisis estático
- Revisar posibles vulnerabilidades en las dependencias

Gradle Enterprise

- Performance insights
- Predictive test selection
- “Build Scan”
- Failure analytics

Build Lifecycle



Q&A