在 build.gradle.kts 文件中，通过 classpath 的方式加载插件，可以使用 buildscript 块声明插件依赖，并使用 apply 函数来应用插件。这种方式通常适用于加载一些不支持 plugins 块的插件（例如自定义插件或一些老旧插件）。

示例：通过 classpath 加载和应用插件

以下是通过 classpath 加载插件的完整示例：

项目的 build.gradle.kts（顶层脚本）

```
buildscript {
    repositories {
        google()
        mavenCentral()
    }
    dependencies {
        // 在 classpath 中添加插件依赖
        classpath("com.android.tools.build:gradle:8.1.0")
        classpath("org.jetbrains.kotlin:kotlin-gradle-
plugin:1.9.10")
    }
}
```

模块的 build.gradle.kts（子模块脚本）

```
// 应用插件
apply(plugin = "com.android.application")
apply(plugin = "kotlin-android")

android {
    compileSdk = 34
    defaultConfig {
        applicationId = "com.example.app"
        minSdk = 21
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"
```

```
    }
}

dependencies {
    implementation("org.jetbrains.kotlin:kotlin-
stdlib:1.9.10")
}
```

分解说明

1. buildscript 块

- 用于定义构建脚本中所需的插件依赖。
- 插件通过 classpath 的方式加载，并存储在 Gradle 构建脚本的上下文中。

2. apply(plugin = ...)

- 用于将加载的插件应用到当前模块中。
- 不同于 plugins 块，apply 不支持直接指定版本号，因此插件的版本号必须在 buildscript 中定义。

混合使用 plugins 和 classpath

在现代 Gradle 项目中，你可以混合使用 plugins 块和 classpath 加载的插件。例如：

顶层 build.gradle.kts

```
buildscript {
    repositories {
```

```
        google()
        mavenCentral()
    }
    dependencies {
        classpath("com.android.tools.build:gradle:8.1.0")
    }
}

plugins {
    id("org.jetbrains.kotlin.jvm") version "1.9.10"
}
```

模块 build.gradle.kts

```
apply(plugin = "com.android.application") // 使用
classpath 加载的插件
apply(plugin = "kotlin-android")          // 使用
classpath 加载的插件

android {
    compileSdk = 34
    defaultConfig {
        applicationId = "com.example.app"
        minSdk = 21
        targetSdk = 34
    }
}

dependencies {
    implementation("org.jetbrains.kotlin:kotlin-
stdlib:1.9.10")
}
```

何时使用 classpath 加载插件?

- 插件未发布到 Gradle Plugin Portal（如某些私有或自定义插件）。
- 使用旧版本插件或遗留项目中需要兼容的插件。

- 需要对插件的依赖进行动态配置（例如从特定的 Maven 仓库获取）。

与 plugins 块的对比

- **优点：**
  - 可以加载所有类型的插件（包括未发布到 Gradle Plugin Portal 的插件）。
  - 更灵活，可以在特定仓库中动态寻找依赖。
- **缺点：**
  - 配置相对繁琐，需要手动指定仓库和依赖。
  - 现代插件更推荐使用 plugins 块加载，语法简洁且易于维护。

在现代项目中，除非必须，否则优先使用 plugins 块加载插件。如果插件仅支持 classpath，则按照上述方式加载。