Configure your build

⇒ The **Android build system** compiles:

① app resources

② Source code

and ③ package them into APKs.

↳ You can test, deploy, sign & distribute the apk file.

⇒ Android studio uses **Gradle**, to automate & manage the build process.

⇒ The **Android plugin for Gradle** works with the build toolkit to provide process & Configuration settings that are specific to building & testing Android application.

⇒ Gradle & Android plugin run independently of Android Studio.

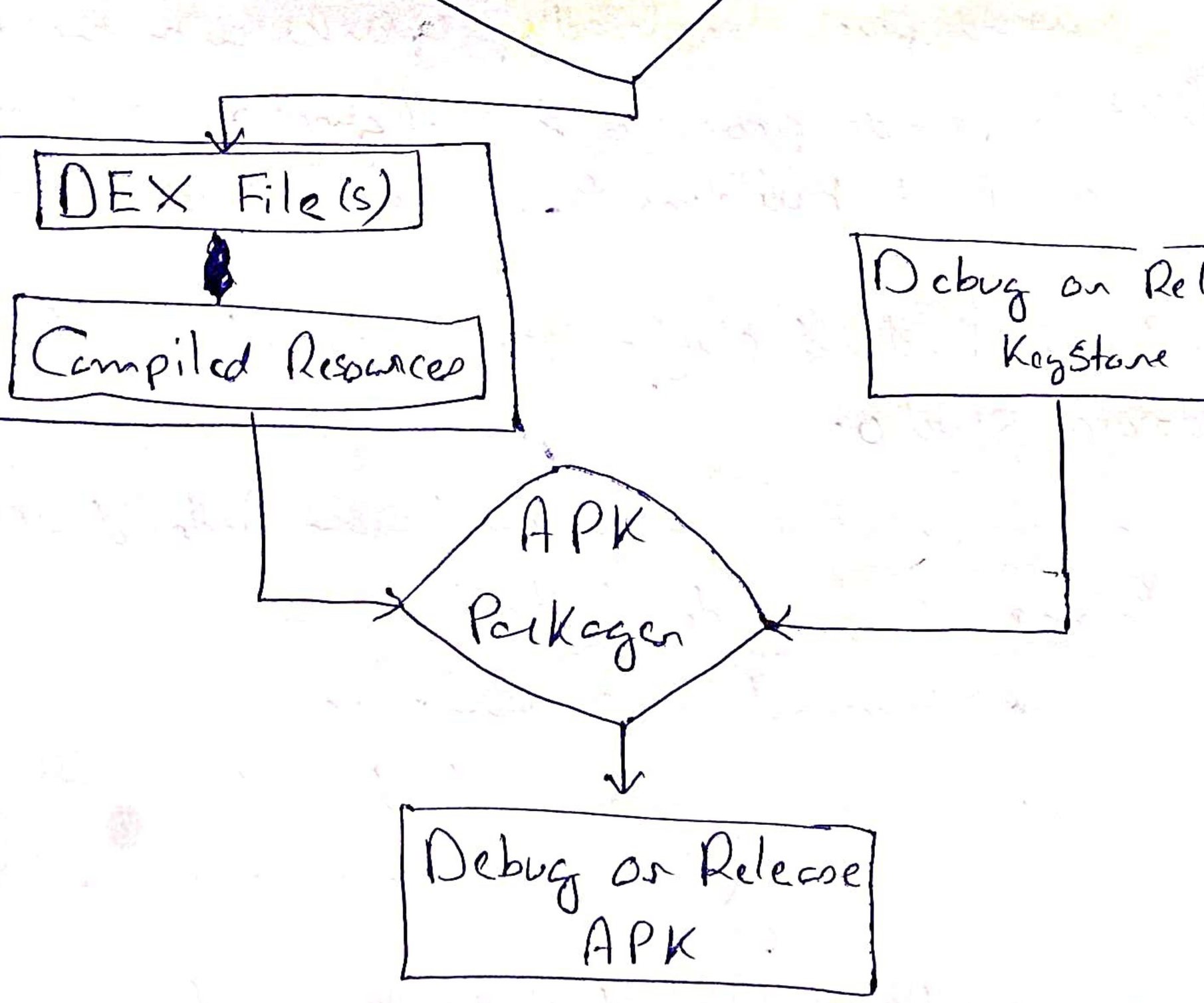
↳ This means that you can build your Android apps from emiting Android Studio

- Command line on your machine.
- machine where android Studio is not installed. (Such as continuous integration Servers)

⇒ Because Gradle & Android plugin run independently from Android Studio, you need to update the build tool Separately.

★ The build process

⇒ The build process involves many tools and processes that convert your project into an Android Application Package (APK).



DEX (Dalvik Executable) file

↳ Include the bytecode that runs on Android devices.

★ Custom build configurations

⇒ Gradle and the Android plugin help you configure the following aspects of your build:

Build types { Exple debug & release build }

- Define certain properties that Gradle uses when Building and packaging your app.
- Typically configured for different stages of your development lifecycle.

Product flavors

- Represent different versions of your app that you may release to users, such as free & paid versions of your app.

Build variants

- Cross product of a build type & product flavor.
- It is the configuration Gradle uses to build your app.

Manifest entries

- You can specify value of some properties of the manifest file in the built variant configuration.
  - These build values override the existing values in the manifest file.

Dependencies

→ The build system manage project dependencies from your local filesystem & from remote repositories.

→ Prevents you from having to manually search, download, and copy binary packages of your dependencies into your project directory.

Signing

→ The build System enables you to specify signing setting in the build configuration, and it can automatically sign your APKs during the build process.

Code & Resource Shrinking

→ Applies the appropriate set of rules to shrink your code & resources using its built-in shrinking tools, such as R8.

Multiple APK support

→ Enables you to automatically build different APKs that each contain only the code and resources needed for a specific screen density or Application Binary Interface (ABI).

* Build Configuration files

⇒ Creating custom build configurations requires you to make changes to one or more build configuration files, or `build.gradle` files.

→ { Uses Domain specific Language (DSL) to describe & manipulate the build logic using Groovy }

Groovy → { Dynamic Language for the Java Virtual Machine (JVM) }

⇒ You dont need to know Groovy to start configuring your build because the Android plugin for gradle introduces most of the DSL elements you need.

⇒ When Starting a new project, Android Studio automatically creates some of these files for you:

```
MyApp/
├→ build.gradle
├→ Settings.gradle
└→ app/
    ├→ build.gradle
    ├→ build/
    ├→ libs/
    └→ src/
        └→ main/
            ├→ Java/
            │   └→ com.exaple.myapp/
            ├→ res/
            │   ├→ drawable/
            │   └→ layout
            └→ AndroidManifest.xml
```

⇒ There are a few Gradle build configuration files that are a part of the standard project structure for an Android app.

(#) The Gradle Setting file { settings.gradle }

⇒ Tells Gradle which modules it should include when building your app.

⇒ For most projects, the file is simple and only includes the following:

```
include: ':app'
```

(#) The top-level build file { build.gradle }

⇒ Defines build configurations that apply to all modules in your project.

⇒ By default, it uses the buildscript block to define the Gradle repositories & dependencies that are common to all modules in the project.

⇒ You should not include dependencies for your module here.

```
buildscript {
    repositories {
        google()
        jcenter()
    }
    dependencies {
        Classpath "com.android.tools.build:gradle:4.1.2"
    }
}
```

⇒ The **repositories block** configures the repositories gradle uses to search on download the dependencies.

↳ You can also use local repositories or define your own remote repositories.

⇒ The **dependencies block** configures the dependencies Gradle needs to use to build your project.

{Android Plugin}

```
allProjects {
    repositories {
        google()
        jcenter()
    }
}
```

⇒ The **allprojects block** is where you configure the repositories & dependencies used by all modules in your project.

{3rd party plugins or libraries}

⊛ Configure project-wide properties

⇒ For android project that includes multiple modules, it may be useful to define certain properties at the project level & share them across all the modules.

⇒ You can do this by adding **ext block** in the top-level build.gradle file.

{Extra properties}

```
ext {
    compileSdkVersion = 28
}
```

⇒ To access these properties from a module in the module's build.gradle file:

`rootProject.ext.<Property_name>`

⊕ The module-level build files {build.gradle}

⇒ Allows you to configure build settings for the specific module it is located in.

```
apply plugin: 'com.android.application'
```

↳ Applies the Android plugin for Gradle to this build & makes the **android block** available to specify Android-specific build options.

```
android {
```

↳ The android block is where you configure all your Android-specific build options.

```
compileSdkVersion 28
```

↳ Specifies the **Android API level** Gradle should use to compile your app.

```
buildToolsVersion "29.0.2"
```

↳ Specifies the version of the SDK build tools, CLI utilities, and compiler that Gradle should use to build your app.

↳ You need to download the build tools using the **SDK manager**.

defaultConfig {

↳ Encapsulates default settings & entries for all build variants.

applicationId 'com.example.myapp'

↳ Uniquely identifies the package for publishing.

minSdkVersion 15

↳ Minimum API level required to run the app

targetSdkVersion 28

↳ API level used to test the app

versionCode 1

↳ Version number of your app

versionName "1.0"

↳ Defines a user-friendly version name for your app.

}

buildTypes {

↳ This is where you can configure multiple build types.

↳ By default, the build system defines two build types:

- debug
- release

}

flavorDimension "tier"

→ If you declare product flavors, you must declare flavor dimensions and assign each flavor to a flavor dimension.

```
productFlavors {
    free {
        dimension "tier"
        applicationId 'com.example.mygpp.free'
    }
    paid {
        dimension "tier"
        applicationId 'com.example.mygpp.paid'
    }
}
```

→ Here you can configure multiple product flavors.
→ Product flavors are optional, and the build system does not create them by default.

```
splits {
    ...
}
```

→ Here you can configure different APK builds that each contain only code & resources for a supported screen density or ABI.

}

→ End of android block

```
dependencies {
    implementation project(":lib")
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation fileTree(dir: 'libs', include: ['*.jar'])
}
```

↳ Specifies dependencies required to build only the module itself.

⑧ Gradle properties files

⇒ Gradle also includes two properties files located in your root project directory, that you can use to specify settings for the Gradle build toolkit itself.

① gradle.properties

⇒ This is where you configure project wide Gradle settings.

② local.properties

⇒ Configures local environment properties for the build system.

- ndk.dir
- sdk.dir
- cmake.dir