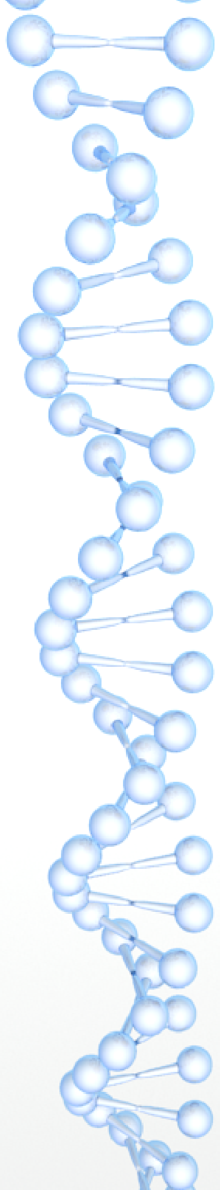
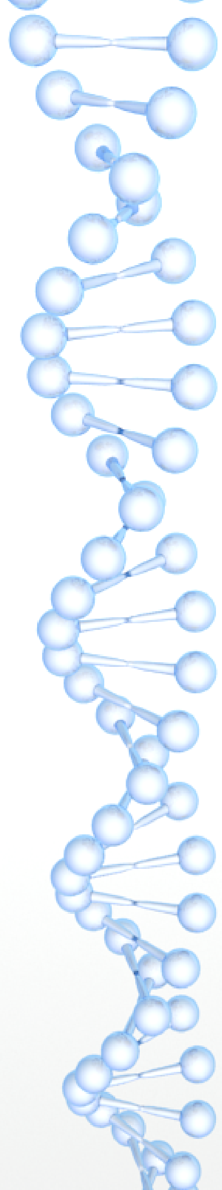


Build Your App from the Command Line

Cont ...

- You can execute all the build tasks available to your Android project using the Gradle wrapper command line tool.
- It's available as a batch file for Windows (***gradlew.bat***) and a shell script for Linux and Mac (***gradlew.sh***), and it's accessible from the root of each project you create with Android Studio.
- To run a task with the wrapper, use one of the following commands:



Cont ...

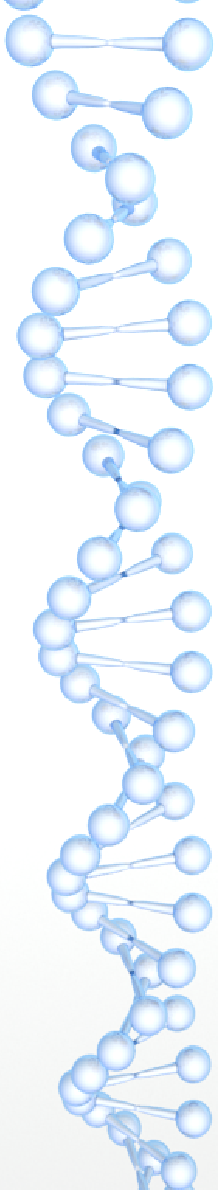
- Windows: ***gradlew task-name***
- Mac/Linux ***./gradlew task-name***
- To see a list of all available build tasks for your project, execute tasks: ***gradlew tasks***

Build types

- By default, there are two build types available for every Android app:
  - one for debugging your app—the **debug build**
  - and one for releasing your app to users—the **release build**.
- The resulting APK from each build must be signed with a certificate before you can install on an emulator or device.
- The debug build is automatically signed with a debug key provided by the SDK tools (it's insecure and you cannot publish this APK to Google Play Store), and the release build must be signed with your own private key.

Cont ...

- If you want to build an APK for release, it's important that you **Sign Your App**.
- You can also define a custom build type in your ***build.gradle*** file and configure it to be signed as a debug build by including ***debuggable true***.



Build a debug APK

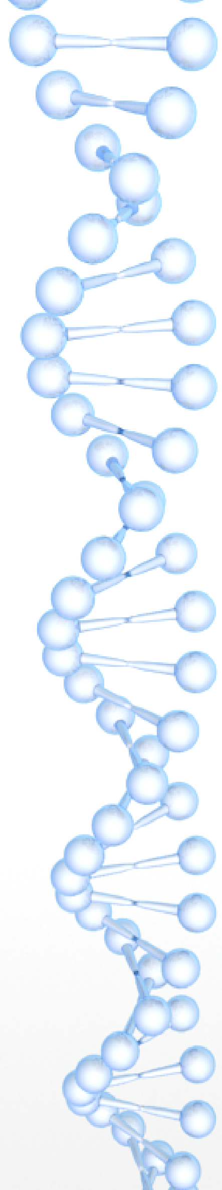
- To build a debug APK, open a command line and navigate to the root of your project directory—from Android Studio, select View > Tool Windows > Terminal.
- To initiate a debug build, invoke the assembleDebug task: **gradlew assembleDebug**
- This creates an APK named module_name-debug.apk in project_name/module_name/build/outputs/apk/.
- The file is already signed with the debug key and aligned with zipalign, so you can immediately install it on a device.

Cont ...

- Or to build the APK and immediately install it on a running emulator or connected device, instead invoke installDebug: **gradlew installDebug**

Build a release APK

- When you're ready to release and distribute your app, you must build a release APK that is signed with your private key



Run your app on the emulator

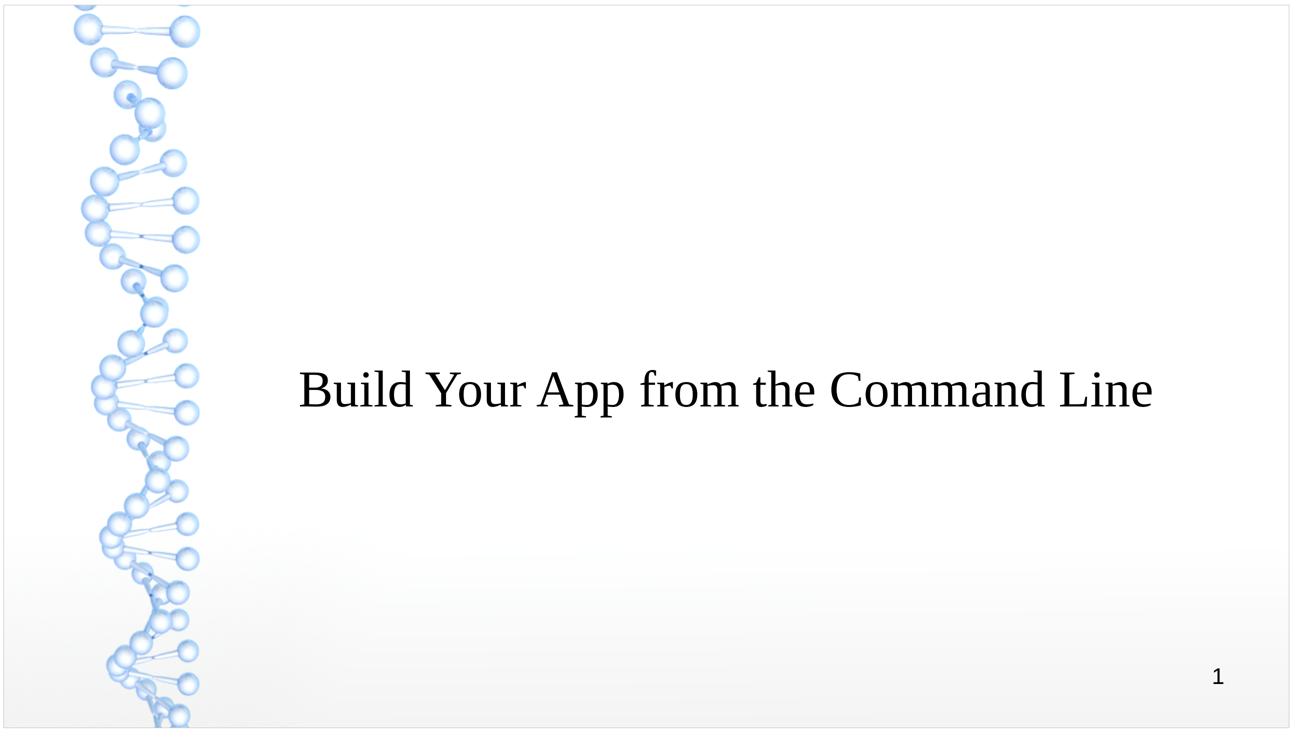
- To use the Android Emulator, you must create an Android Virtual Device (AVD) using Android Studio.
- Once you have an AVD, start the Android Emulator and install your app as follows:
  - In a command line, navigate to android_sdk/tools/ and start the emulator by specifying your AVD: **emulator -avd avd_name.** If you're unsure of the AVD name, execute emulator -list-avds.
  -

Cont ...

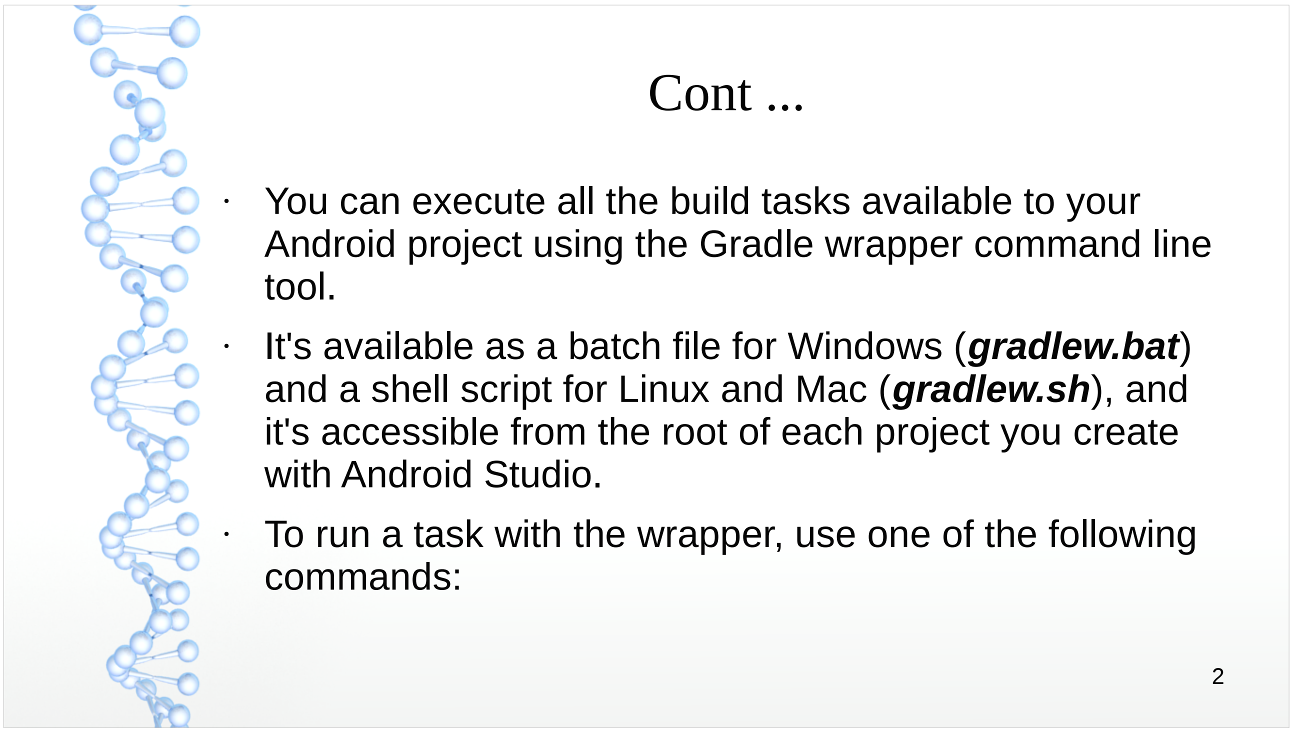
- Now you can install your app using either the Gradle install tasks mentioned above or the adb tool: **adb install path/to/your_app.apk.**
- All built APKs are saved in project_name/module_name/build/outputs/apk/.

Run your app on a device

- Before you can run your app on a device, you must enable USB debugging on your device. You can find the option under Settings > Developer options.
- Once your device is set up and connected via USB, you can install your app using either the Gradle install tasks mentioned above or the adb tool: **adb -d install path/to/your_app.apk**
- All built APKs are saved in project_name/module_name/build/outputs/apk/.

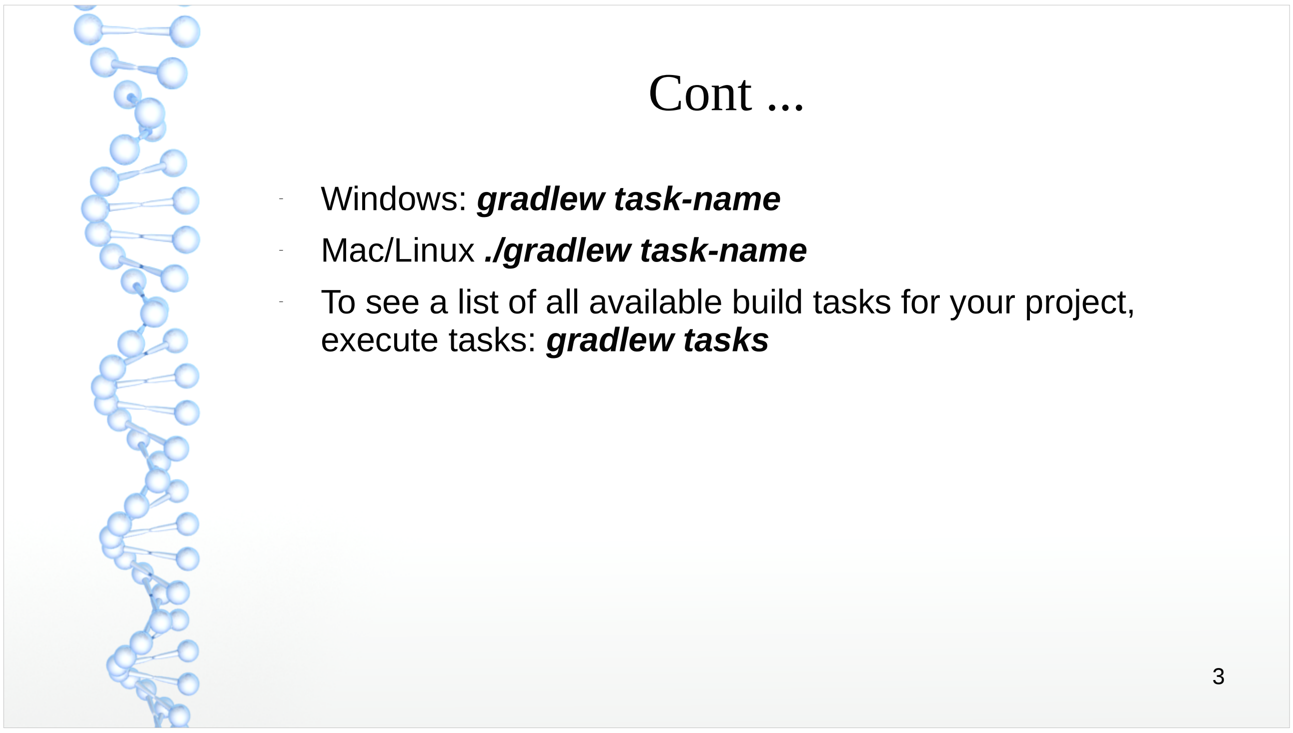


Build Your App from the Command Line



Cont ...

- You can execute all the build tasks available to your Android project using the Gradle wrapper command line tool.
- It's available as a batch file for Windows (***gradlew.bat***) and a shell script for Linux and Mac (***gradlew.sh***), and it's accessible from the root of each project you create with Android Studio.
- To run a task with the wrapper, use one of the following commands:

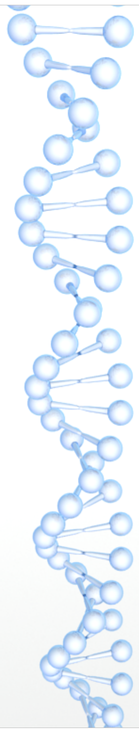


Cont ...

- Windows: ***gradlew task-name***
- Mac/Linux ***./gradlew task-name***
- To see a list of all available build tasks for your project, execute tasks: ***gradlew tasks***

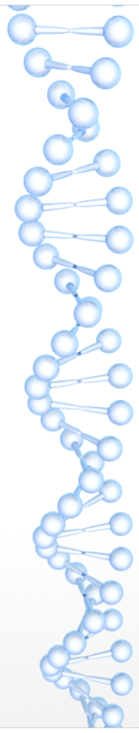
Build types

- By default, there are two build types available for every Android app:
  - one for debugging your app—the **debug build**
  - and one for releasing your app to users—the **release build**.
- The resulting APK from each build must be signed with a certificate before you can install on an emulator or device.
- The debug build is automatically signed with a debug key provided by the SDK tools (it's insecure and you cannot publish this APK to Google Play Store), and the release build must be signed with your own private key.



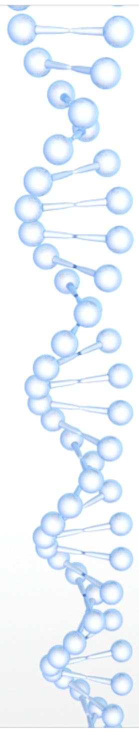
Cont ...

- If you want to build an APK for release, it's important that you **Sign Your App**.
- You can also define a custom build type in your ***build.gradle*** file and configure it to be signed as a debug build by including ***debuggable true***.



Build a debug APK

- To build a debug APK, open a command line and navigate to the root of your project directory—from Android Studio, select View > Tool Windows > Terminal.
- To initiate a debug build, invoke the assembleDebug task: **gradlew assembleDebug**
- This creates an APK named module_name-debug.apk in project_name/module_name/build/outputs/apk/.
- The file is already signed with the debug key and aligned with zipalign, so you can immediately install it on a device.



Cont ...

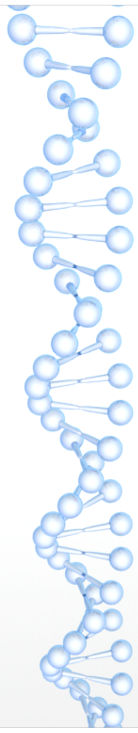
- Or to build the APK and immediately install it on a running emulator or connected device, instead invoke installDebug: **gradlew installDebug**

Build a release APK

- When you're ready to release and distribute your app, you must build a release APK that is signed with your private key

Run your app on the emulator

- To use the Android Emulator, you must create an Android Virtual Device (AVD) using Android Studio.
- Once you have an AVD, start the Android Emulator and install your app as follows:
  - In a command line, navigate to android_sdk/tools/ and start the emulator by specifying your AVD: **emulator -avd avd_name.** If you're unsure of the AVD name, execute emulator -list-avds.



Cont ...

- Now you can install your app using either the Gradle install tasks mentioned above or the adb tool: **adb install path/to/your_app.apk.**
- All built APKs are saved in project_name/module_name/build/outputs/apk/.

Run your app on a device

- Before you can run your app on a device, you must enable USB debugging on your device. You can find the option under Settings > Developer options.
- Once your device is set up and connected via USB, you can install your app using either the Gradle install tasks mentioned above or the adb tool: **adb -d install path/to/your_app.apk**
- All built APKs are saved in project_name/module_name/build/outputs/apk/.