**Java Development Kit (JDK)**

Install [Java Development Kit (JDK) 8](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) or later.

When installing on Windows you also need to set JAVA\_HOME Environment Variable according to your JDK installation path (see [Setting Environment Variables](https://cordova.apache.org/docs/en/latest/guide/platforms/android/index.html#setting-environment-variables))

**Android SDK**

Install [Android Studio](https://developer.android.com/studio/index.html). Detailed installation instructions are on Android's developer site.

**Adding SDK Packages**

After installing the Android SDK, you must also install the packages for whatever [API level](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels) you wish to target. It is recommended that you install the highest SDK version that your version of cordova-android supports (see [Requirements and Support](https://cordova.apache.org/docs/en/latest/guide/platforms/android/index.html#requirements-and-support)).

Open the Android SDK Manager (run android or sdkmanager from the terminal) and make sure the following are installed:

1. Android Platform SDK for your targeted version of Android
2. Android SDK build-tools version 19.1.0 or higher
3. Android Support Repository (found under "Extras")

See Android's documentation on [Installing SDK Packages](https://developer.android.com/studio/intro/update.html) for more details.

**Setting environment variables**

Cordova's CLI tools require some environment variables to be set in order to function correctly. The CLI will attempt to set these variables for you, but in certain cases you may need to set them manually. The following variables should be updated:

1. Set the JAVA\_HOME environment variable to the location of your JDK installation
2. Set the ANDROID\_HOME environment variable to the location of your Android SDK installation
3. It is also recommended that you add the Android SDK's tools, tools/bin, and platform-tools directories to your PATH

**OS X and Linux**

On a Mac or Linux, you can use a text editor to create or modify the ~/.bash\_profile file. To set an environment variable, add a line that uses export like so (substitute the path with your local installation):

export ANDROID\_HOME**=**/Development/android-sdk/

To update your PATH, add a line resembling the following (substitute the paths with your local Android SDK installation's location):

export PATH**=${**PATH**}**:/Development/android-sdk/platform-tools:/Development/android-sdk/tools

Reload your terminal to see this change reflected or run the following command:

$ source ~/.bash\_profile

**Windows**

These steps may vary depending on your installed version of Windows. Close and reopen any command prompt windows after making changes to see them reflected.

1. Click on the **Start** menu in the lower-left corner of the desktop
2. In the search bar, search for **Environment Variables** and select **Edit the system Environment Variables** from the options that appear
3. In the window that appears, click the **Environment Variables** button

**To create a new environment variable:**

1. Click **New...** and enter the variable name and value

**To set your PATH:**

1. Select the **PATH** variable and press **Edit**.
2. Add entries for the relevant locations to the **PATH**. For example (substitute the paths with your local Android SDK installation's location):
3. C:\Development\android-sdk\platform-tools
4. C:\Development\android-sdk\tools

Project Configuration

**Setting up an Emulator**

If you wish to run your Cordova app on an Android emulator, you will first need to create an Android Virtual Device (AVD). See the Android documentation for [managing AVDs](https://developer.android.com/studio/run/managing-avds.html), [configuring the emulator](https://developer.android.com/studio/run/emulator.html#about), and [setting up hardware acceleration](https://developer.android.com/studio/run/emulator-acceleration.html).

Once your AVD is configured correctly, you should be able to deploy your Cordova application to the emulator by running:

$ cordova run --emulator

**Configuring Gradle**

As of **cordova-android@4.0.0**, Cordova for Android projects are built using [Gradle](http://www.gradle.org/). For instructions on building with Ant, refer to older versions of the documentation. Please note that Ant builds are deprecated as of the Android SDK Tools 25.3.0.

**Setting Gradle Properties**

It is possible to configure the Gradle build by setting the values of certain [Gradle properties](https://docs.gradle.org/current/userguide/build_environment.html) that Cordova exposes. The following properties are available to be set:

| **Property** | **Description** |
| --- | --- |
| cdvBuildMultipleApks | If this is set, then multiple APK files will be generated: One per native platform supported by library projects (x86, ARM, etc). This can be important if your project uses large native libraries, which can drastically increase the size of the generated APK. If not set, then a single APK will be generated which can be used on all devices |
| cdvVersionCode | Overrides the versionCode set in AndroidManifest.xml |
| cdvReleaseSigningPropertiesFile | *Default: release-signing.properties* Path to a .properties file that contains signing information for release builds (see [Signing an App](https://cordova.apache.org/docs/en/latest/guide/platforms/android/index.html#signing-an-app)) |
| cdvDebugSigningPropertiesFile | *Default: debug-signing.properties* Path to a .properties file that contains signing information for debug builds (see [Signing an App](https://cordova.apache.org/docs/en/latest/guide/platforms/android/index.html#signing-an-app)). Useful when you need to share a signing key with other developers |
| cdvMinSdkVersion | Overrides the value of minSdkVersion set in AndroidManifest.xml. Useful when creating multiple APKs based on SDK version |
| cdvBuildToolsVersion | Overrides the automatically detected android.buildToolsVersion value |
| cdvCompileSdkVersion | Overrides the automatically detected android.compileSdkVersion value |

You can set these properties in one of four ways:

1. By setting environment variables like so:
2. $ export ORG\_GRADLE\_PROJECT\_cdvMinSdkVersion**=**20
3. $ cordova build android
4. By using the --gradleArg flag in your Cordova build or run commands:
5. $ cordova run android -- --gradleArg**=**-PcdvMinSdkVersion**=**20
6. By placing a file called gradle.properties in your Android platform folder (<your-project>/platforms/android) and setting the properties in it like so:
7. # In <your-project>/platforms/android/gradle.properties
8. cdvMinSdkVersion=20
9. By extending build.gradle via a [build-extras.gradle file](https://cordova.apache.org/docs/en/latest/guide/platforms/android/index.html#extending-buildgradle) and setting the property like so:
10. *// In <your-project>/platforms/android/build-extras.gradle*
11. ext**.**cdvMinSdkVersion **=** 20

The latter two options both involve including an extra file in your Android platform folder. In general, it is discouraged that you edit the contents of this folder because it is easy for those changes to be lost or overwritten. Instead, these two files should be copied from another location into that folder as part of the build command by using the before\_build [hook](https://cordova.apache.org/docs/en/latest/guide/appdev/hooks/index.html).

**Extending build.gradle**

If you need to customize build.gradle, rather than edit it directly, you should create a sibling file named build-extras.gradle. This file will be included by the main build.gradle when present. This file must be placed in the android platform directory (<your-project>/platforms/android), so it is recommended that you copy it over via a script attached to the before\_build [hook](https://cordova.apache.org/docs/en/latest/guide/appdev/hooks/index.html).

Here's an example:

*// Example build-extras.gradle*

*// This file is included at the beginning of `build.gradle`*

ext**.**cdvDebugSigningPropertiesFile **=** '../../android-debug-keys.properties'

*// When set, this function allows code to run at the end of `build.gradle`*

ext**.**postBuildExtras **=** **{**

android**.**buildTypes**.**debug**.**applicationIdSuffix **=** '.debug'

**}**

Note that plugins can also include build-extras.gradle files via:

<framework src="some.gradle" custom="true" type="gradleReference" />

**Setting the Version Code**

To change the [version code](https://developer.android.com/studio/publish/versioning.html) for your app's generated apk, set the android-versionCode attribute in the widget element of your application's [config.xml file](https://cordova.apache.org/docs/en/latest/config_ref/index.html). If the android-versionCode is not set, the version code will be determined using the versionattribute. For example, if the version is MAJOR.MINOR.PATCH:

versionCode = MAJOR \* 10000 + MINOR \* 100 + PATCH

If your application has enabled the cdvBuildMultipleApks Gradle property (see [Setting Gradle Properties](https://cordova.apache.org/docs/en/latest/guide/platforms/android/index.html#setting-gradle-properties)), the version code of your app will also be multiplied by 10 so that the last digit of the code can be used to indicate the architecture the apk was built for. This multiplication will happen regardless of whether the version code is taken from the android-versionCodeattribute or generated using the version. Be aware that some plugins added to your project (including cordova-plugin-crosswalk-webview) may set this Gradle property automatically.

**Please Note:** When updating the android-versionCode property, it is unwise to increment the version code taken from built apks. Instead, you should increment the code based off the value in your config.xml file's android-versionCode attribute. This is because the cdvBuildMultipleApks property causes the version code to be multiplied by 10 in the built apks and thus using that value will cause your next version code to be 100 times the original, etc.