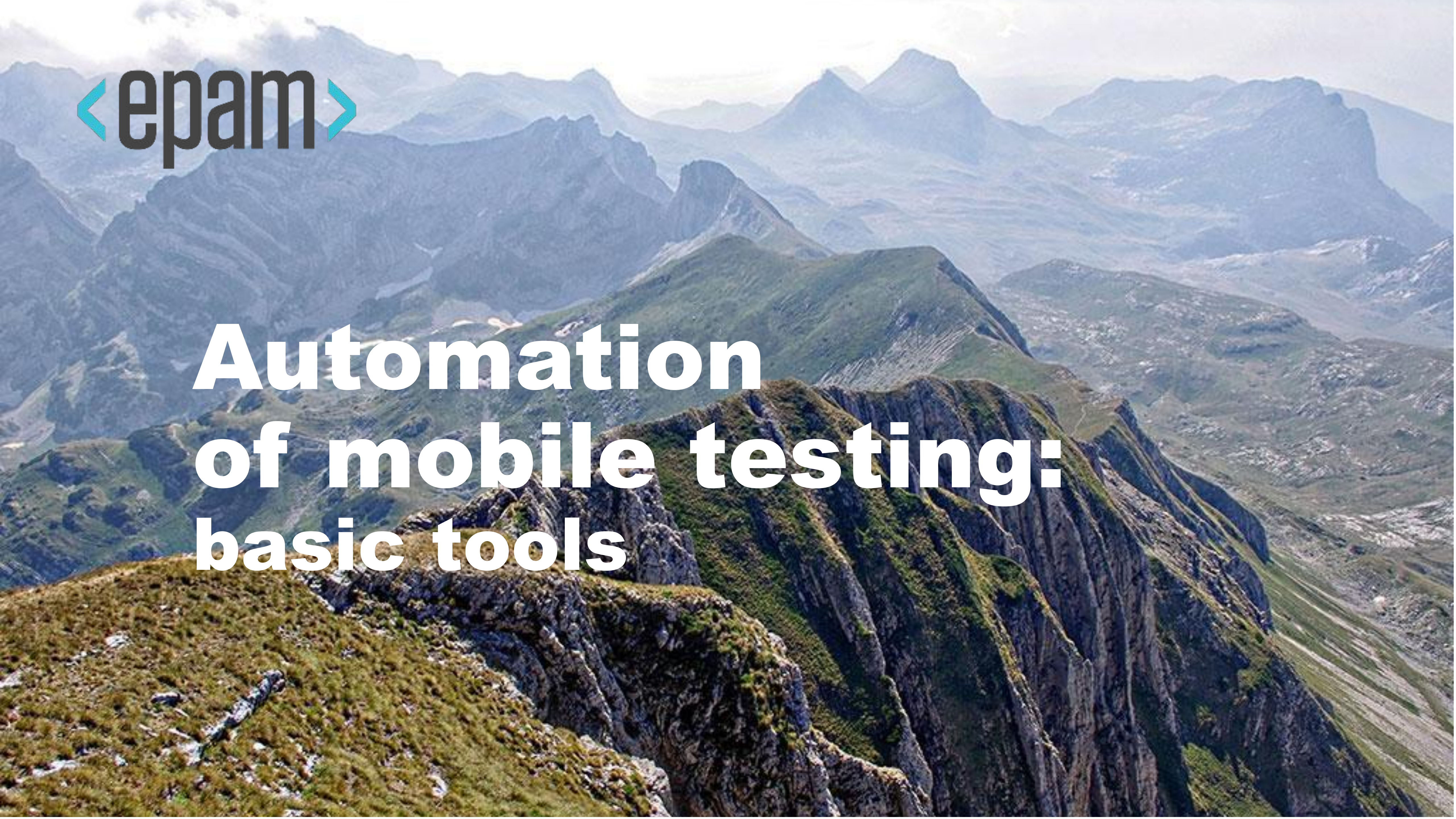


<epam>

Automation of mobile testing: basic tools

Plan

- Basic tools
  - Physical devices, emulators and cloud solutions
  - Appium
- How to write auto-tests
- Basics of Mobile Cloud Services
- Tips and tricks

Physical devices, emulators and cloud solutions

Device vs emulator

Physical device

Pros:

- **Expected user experience**

Contras:

- Expensive
- You need a lot of physical items
- Power and space consumption

Emulator

Pros:

- Cheap
- A lot of parameters can be adjusted: dimensions, RAM, disk space, set of sensors, ...
- No power and space consumption

Contras:

- **Not realistic behaviour**
- Computing consumption
- Performance issues
- Additional software

Device vs emulator: purposes

Emulator:

- Prototyping (GUI, layouts, ...)
- Early stage of auto-tests development

Device:

- Auto-tests finalising and debugging
- Auto-tests run

Mobile cloud services

Mobile cloud services (mobile farms) are the modern approach

They provide developers and testers with remote access to sets of physical devices *for fixed prices*

Remote access to set of emulators can be provided as well for less prices

Mobile platforms

- IOS
  - You have to be a registered Android developer
  - You have to use Apple/Mac environment (Xcode)
- Android
  - You can use free open-source tools on Win/Mac/Linux
  - Occupied most of mobile market at the moment

General environment settings for Android platform

The best way

You need JDK to work with Android development tools. Please use 8th release (9th has some problems yet.)

With the Java SDK ver.8, please install Android Studio **Bundle**.

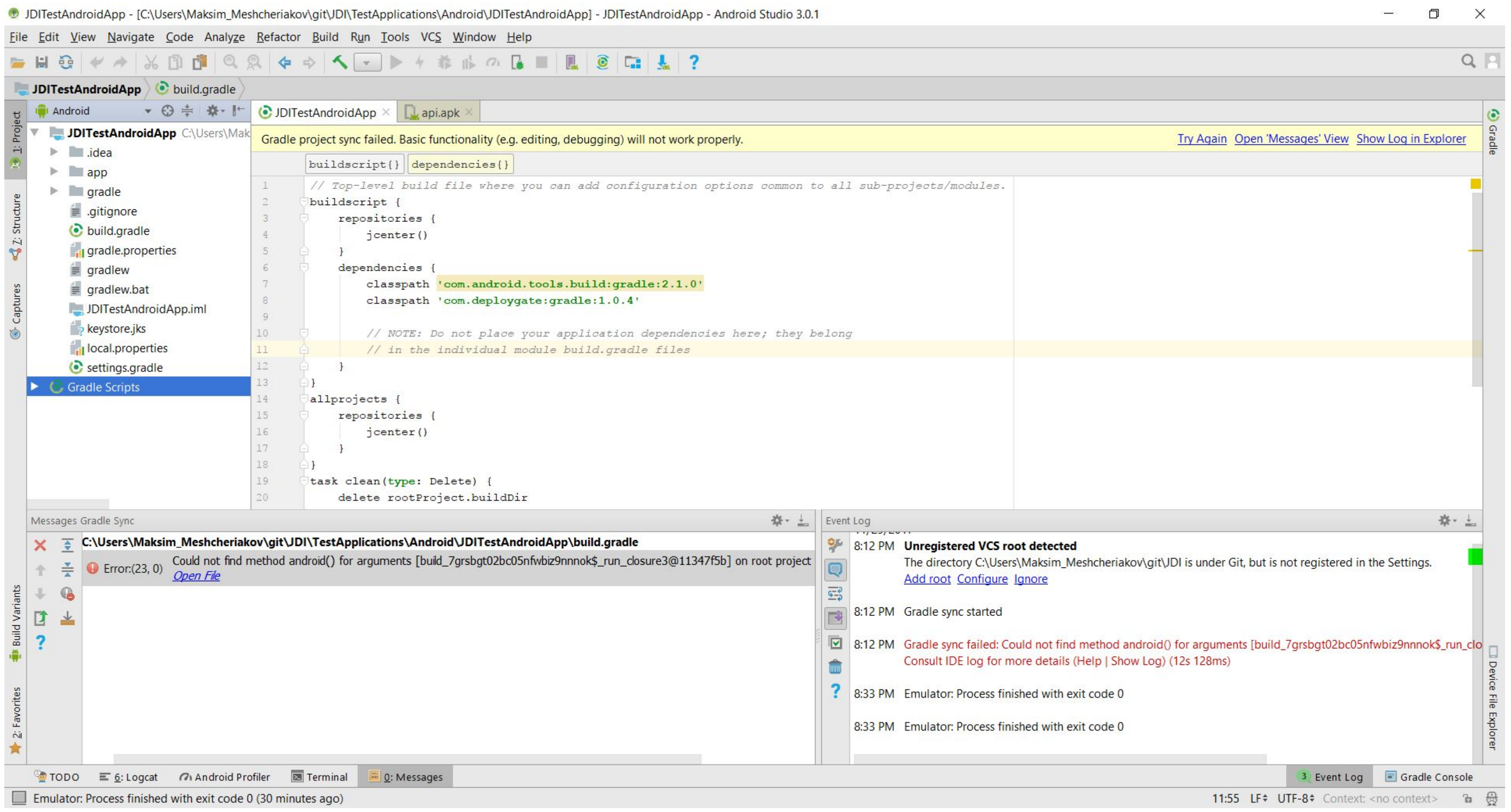
Bundle includes the complete set of all required tools, including Android SDK.

Otherwise, you will have to install and configure several packages by himself.

Android basic toolset

- Android Studio is the common toolset
- Android SDK (includes some CLI tools)
- Android Debug Bridge - ADB
- Android Virtual Device - AVD, and AVD Manager
- Android Device Monitor

Android Studio



JDITestAndroidApp - [C:\Users\Maksim_Meshcheriakov\git\JDI\TestApplications\Android\JDITestAndroidApp] - JDITestAndroidApp - Android Studio 3.0.1

Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not work properly. Try Again Open 'Messages' View Show Log in Explorer

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.0'
        classpath 'com.deploygate:gradle:1.0.4'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}
allprojects {
    repositories {
        jcenter()
    }
}
task clean(type: Delete) {
    delete rootProject.buildDir
```

Messages Gradle Sync

```
C:\Users\Maksim_Meshcheriakov\git\JDI\TestApplications\Android\JDITestAndroidApp\build.gradle
Error:(23, 0)  Could not find method android() for arguments [build_7grsbgt02bc05nfwbiz9nnnok$_run_closure3@11347f5b] on root project
               Open File
```

Event Log

```
8:12 PM  Unregistered VCS root detected
         The directory C:\Users\Maksim_Meshcheriakov\git\JDI is under Git, but is not registered in the Settings.
         Add root  Configure  Ignore

8:12 PM  Gradle sync started

8:12 PM  Gradle sync failed: Could not find method android() for arguments [build_7grsbgt02bc05nfwbiz9nnnok$_run_clo
         Consult IDE log for more details (Help | Show Log) (12s 128ms)

8:33 PM  Emulator: Process finished with exit code 0

8:33 PM  Emulator: Process finished with exit code 0
```

Emulator: Process finished with exit code 0 (30 minutes ago)

Currently, Android Studio is used for most tasks to develop, debug and test Android applications

https://developer.android.com/studio/intro/index.html

Environment variables

- JAVA_HOME = Program Files\Java\jdkXX.YY (actual JDK location)
- ANDROID_HOME = ~\AppData\Local\Android\sdk (actual path to Android SDK)
- PATH = %PATH%, %ANDROID_HOME%\tools, %ANDROID_HOME%\platform-tools

Android emulators setting up

Android Virtual Device

This is emulator of a certain Android device.
NOTE: AVD emulates mobile *hardware* (instead of iOS *simulator*)- first of all, ARM-based processor.

You can create a set of emulator that have different capabilities:

- Dimensions and form-factor
- Display parameters
- API level (Android version)
- RAM and disk space size
- Set of sensors

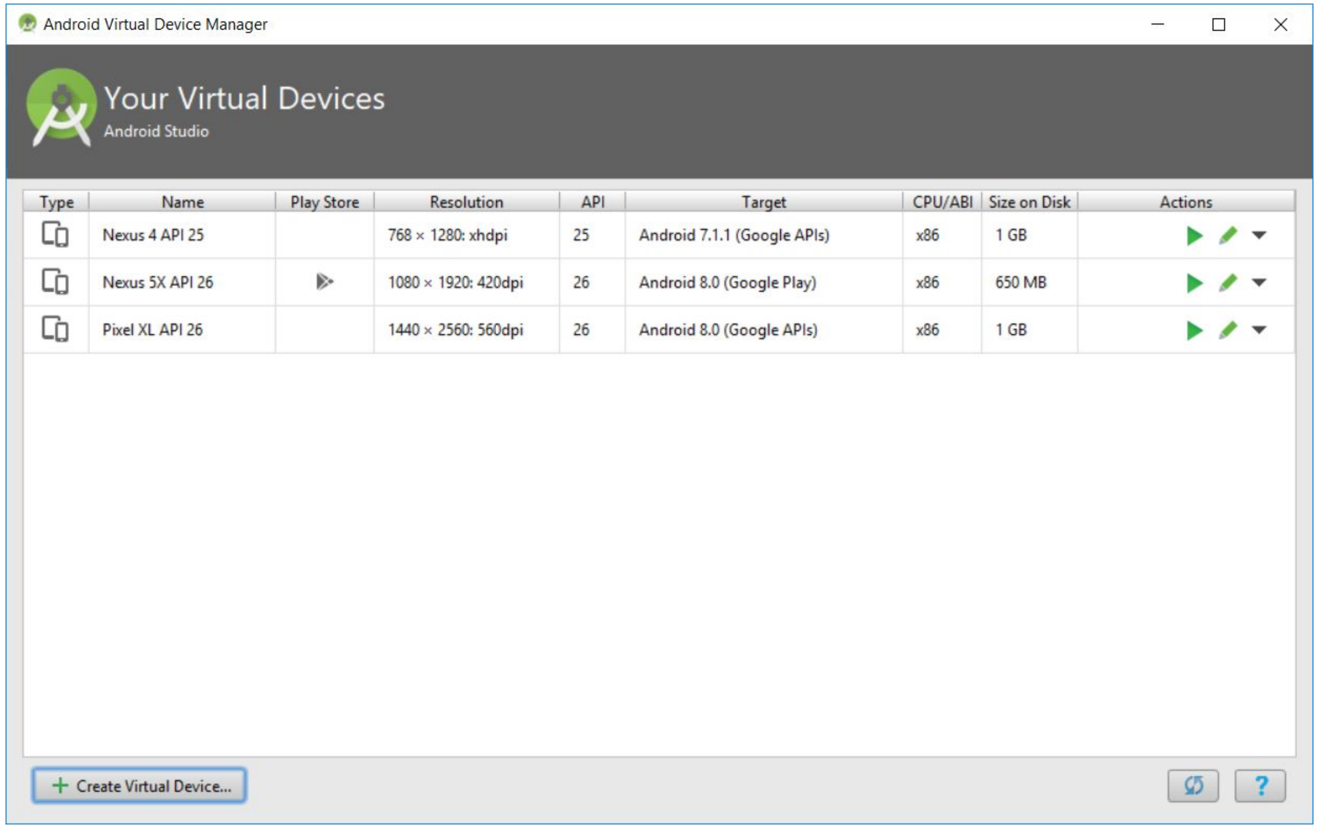
Consist of AVD

- **Hardware profile**: pre-sets of characteristics of a (real) devices. Some profiles include Play Store (indicated). Could be created and/or imported as well
- **System image**: set of software options - certain API version, set of applications
- **Storage area**: dedicated storage area on host computer. It stores the device user data (apps and settings), emulated SD card
- **Skin**: the appearance of a device. The AVD Manager provides some predefined skins. User can define his own skins or use 3d-party ones

Access to AVD manager

To open the AVD Manager in Android Studio, do one of the following:

- Select Tools > Android > AVD Manager
- Click AVD Manager icon in the toolbar



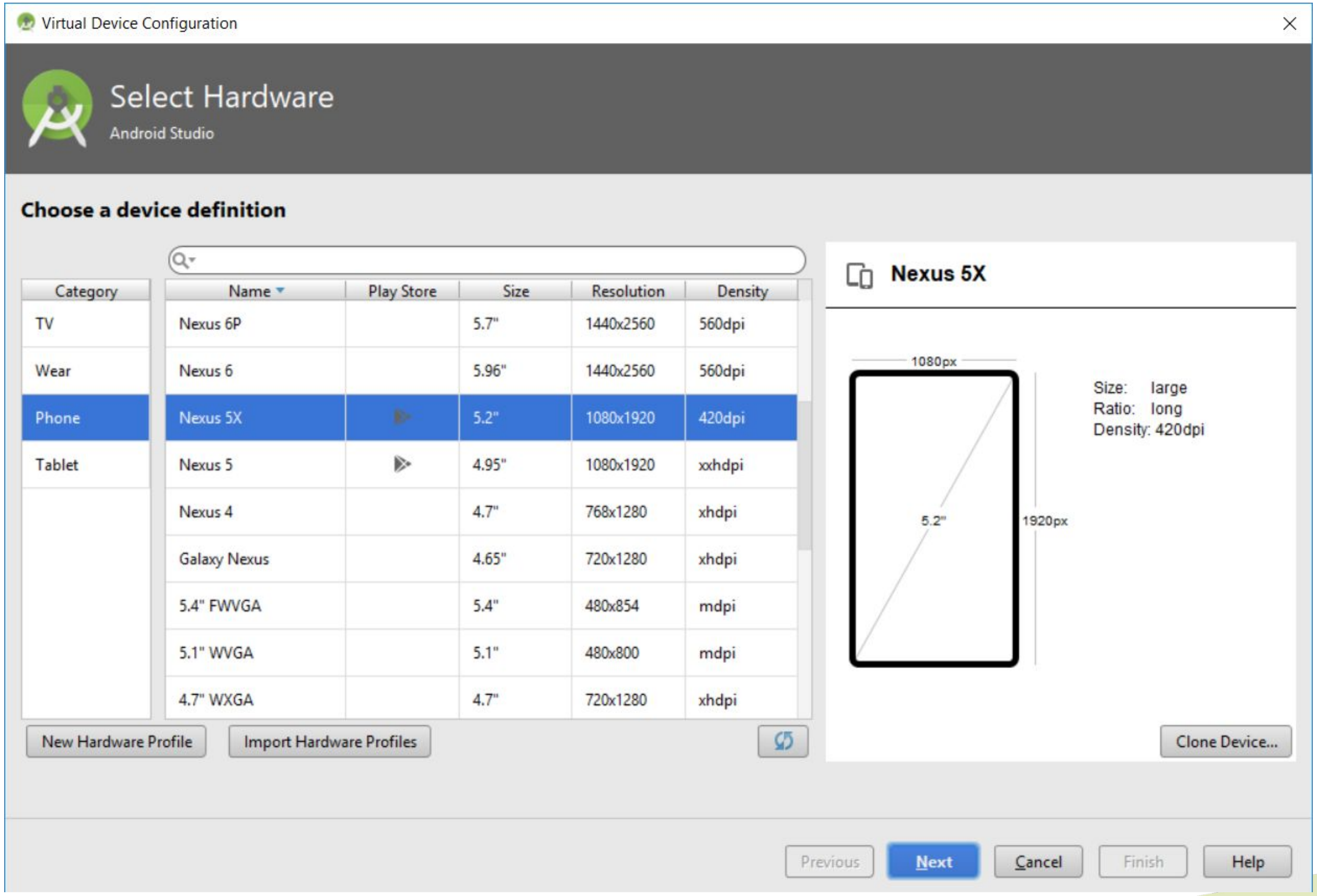
Android Virtual Device Manager

Your Virtual Devices

Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 4 API 25		768 × 1280: xhdpi	25	Android 7.1.1 (Google APIs)	x86	1 GB	
	Nexus 5X API 26		1080 × 1920: 420dpi	26	Android 8.0 (Google Play)	x86	650 MB	
	Pixel XL API 26		1440 × 2560: 560dpi	26	Android 8.0 (Google APIs)	x86	1 GB	

Start to create an AVD



Virtual Device Configuration

Select Hardware

Android Studio

Choose a device definition

Category	Name	Play Store	Size	Resolution	Density
TV	Nexus 6P		5.7"	1440x2560	560dpi
Wear	Nexus 6		5.96"	1440x2560	560dpi
Phone	Nexus 5X	▶	5.2"	1080x1920	420dpi
Tablet	Nexus 5	▶	4.95"	1080x1920	xxhdpi
	Nexus 4		4.7"	768x1280	xhdpi
	Galaxy Nexus		4.65"	720x1280	xhdpi
	5.4" FWVGA		5.4"	480x854	mdpi
	5.1" WVGA		5.1"	480x800	mdpi
	4.7" WXGA		4.7"	720x1280	xhdpi

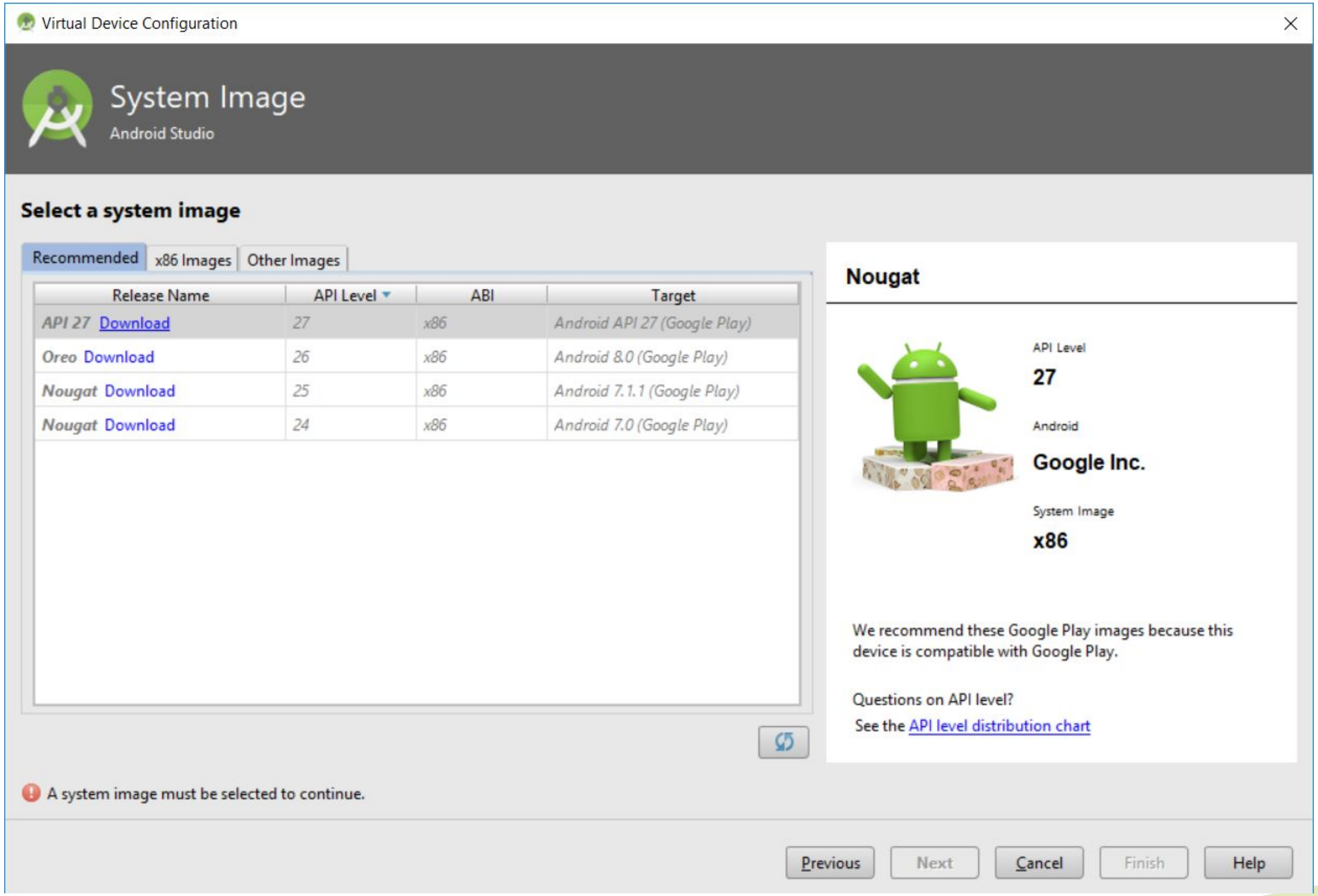
Nexus 5X

Size: large
Ratio: long
Density: 420dpi

Click Create Virtual Device at the bottom of the AVD Manager dialog

The Select Hardware page appears.

Select hardware profile


Virtual Device Configuration

System Image
Android Studio

Select a system image

Recommended | x86 Images | Other Images

Release Name	API Level	ABI	Target
API 27 Download	27	x86	Android API 27 (Google Play)
Oreo Download	26	x86	Android 8.0 (Google Play)
Nougat Download	25	x86	Android 7.1.1 (Google Play)
Nougat Download	24	x86	Android 7.0 (Google Play)

Nougat

API Level
27

Android
Google Inc.

System Image
x86

We recommend these Google Play images because this device is compatible with Google Play.

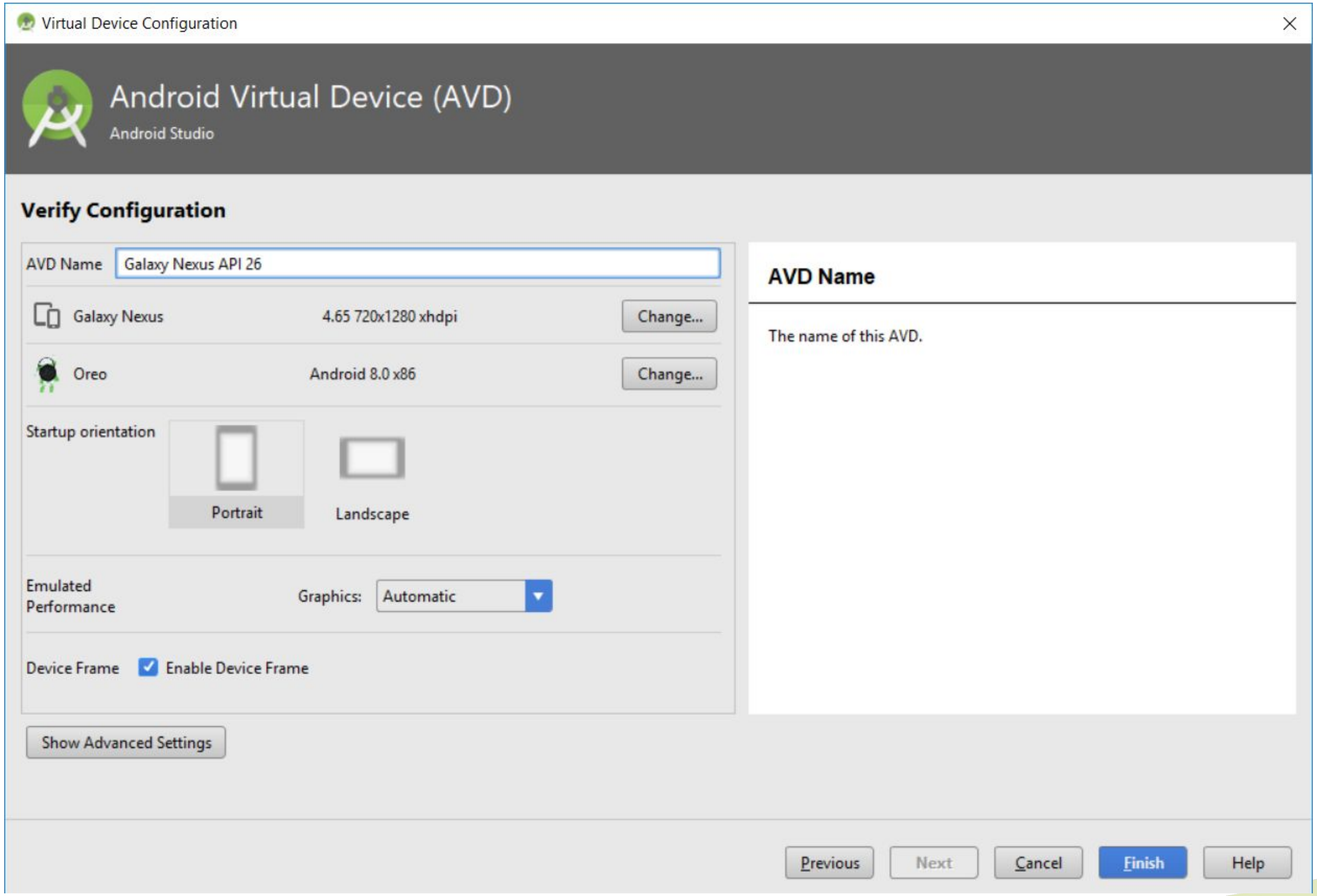
Questions on API level?
See the API level distribution chart

A system image must be selected to continue.

- Select a hardware profile
- Click Next

The System Image page appears.

Verify AVD


Virtual Device Configuration

Android Virtual Device (AVD)
Android Studio

Verify Configuration

AVD Name: Galaxy Nexus API 26

Galaxy Nexus 4.65 720x1280 xhdpi Change...

Oreo Android 8.0 x86 Change...

Startup orientation: Portrait / Landscape

Emulated Performance — Graphics: Automatic

Device Frame: [✓] Enable Device Frame

Show Advanced Settings

AVD Name

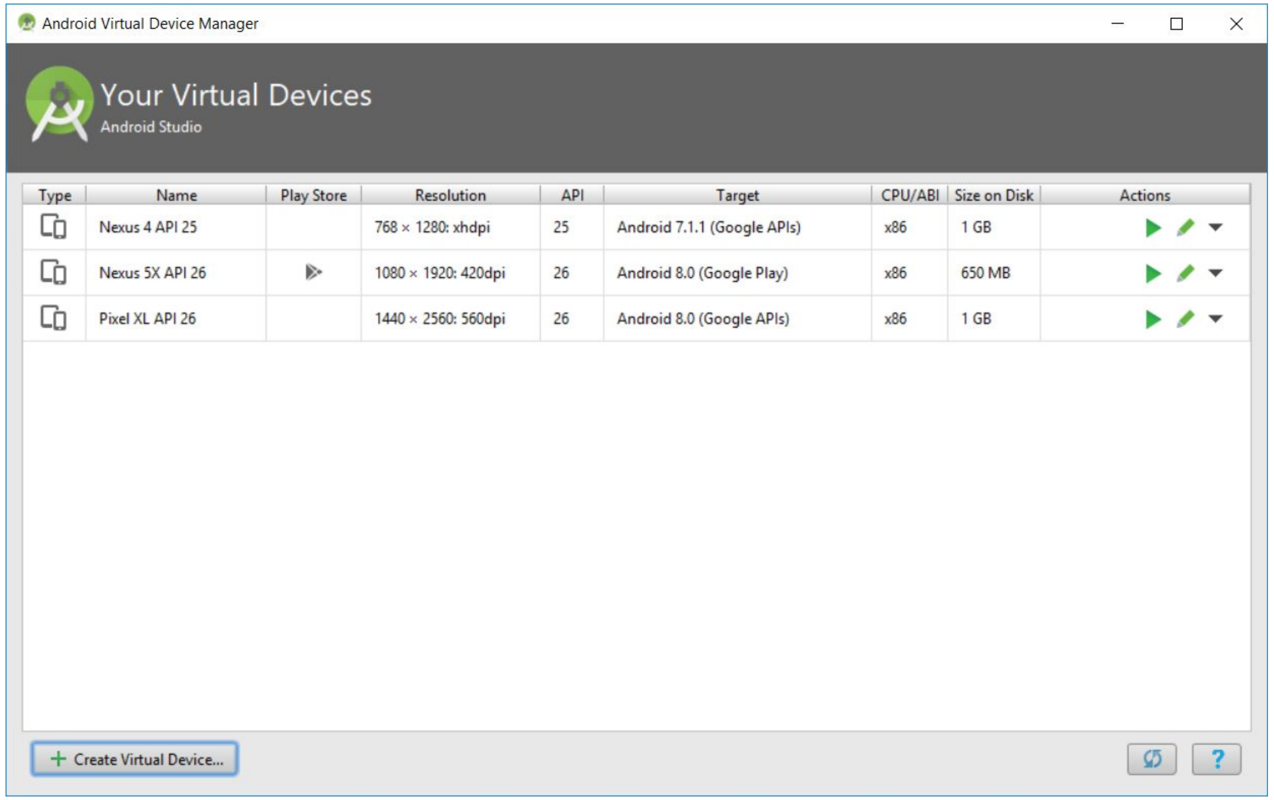
The name of this AVD.

Previous | Next | Cancel | Finish | Help

Verify new AVD and accept (Finish) or adjust its parameters (Previous, Change, Show Advanced Settings)

Editable AVD

- Each existing AVD parameter or feature can be changed and saved for future using
- New changes overwrite default ones of hardware profile and other AVD parts



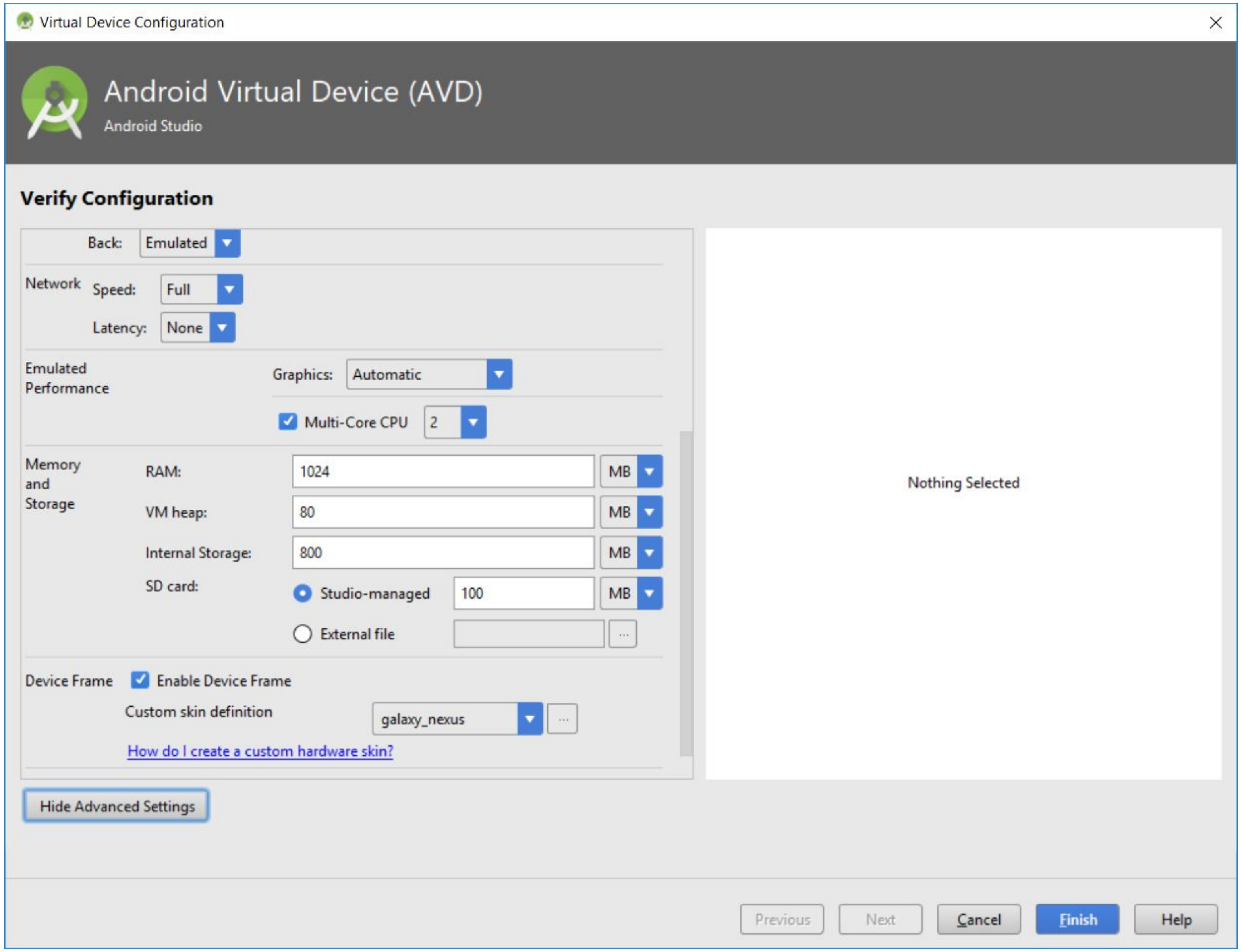
Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 4 API 25		768 × 1280: xhdpi	25	Android 7.1.1 (Google APIs)	x86	1 GB	
	Nexus 5X API 26		1080 × 1920: 420dpi	26	Android 8.0 (Google Play)	x86	650 MB	
	Pixel XL API 26		1440 × 2560: 560dpi	26	Android 8.0 (Google APIs)	x86	1 GB	

- Use “Edit” icon of certain AVD to change required parameters

Advanced settings of AVD


Virtual Device Configuration

Android Virtual Device (AVD)
Android Studio

Verify Configuration

	Back:	Emulated
Network	Speed:	Full
	Latency:	None
Emulated Performance	Graphics:	Automatic
	✓ Multi-Core CPU	2
Memory and Storage	RAM:	1024 MB
	VM heap:	80 MB
	Internal Storage:	800 MB
	SD card:	● Studio-managed 100 MB
		○ External file
Device Frame	✓ Enable Device Frame	
	Custom skin definition	galaxy_nexus

How do I create a custom hardware skin?

Nothing Selected

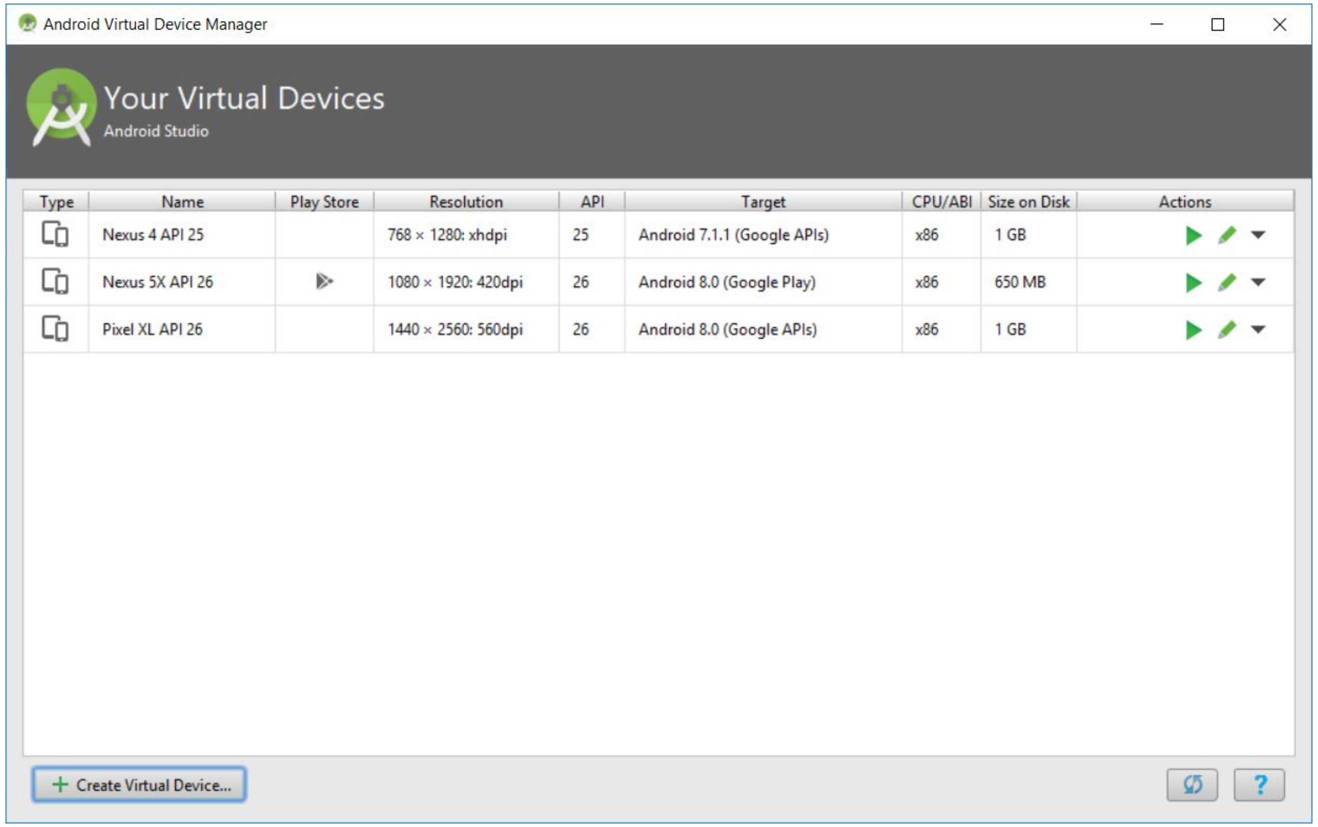
Click

“Show Advanced Settings“

button to get access to more editable settings.

Scroll down to see full list of ones.

Run, stop and wipe an AVD



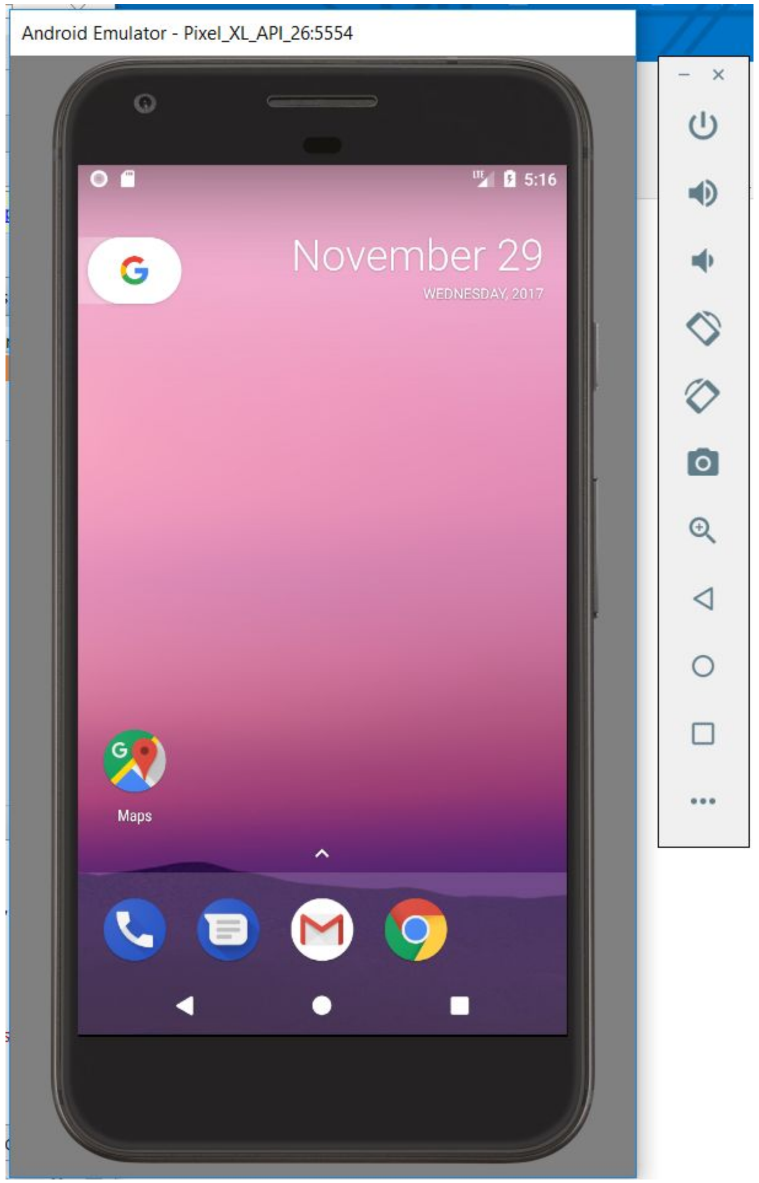
Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 4 API 25		768 × 1280: xhdpi	25	Android 7.1.1 (Google APIs)	x86	1 GB	
	Nexus 5X API 26		1080 × 1920: 420dpi	26	Android 8.0 (Google Play)	x86	650 MB	
	Pixel XL API 26		1440 × 2560: 560dpi	26	Android 8.0 (Google APIs)	x86	1 GB	

- Double-click the required AVD or click Launch ▶ **to run an emulator**
- Right-click an AVD and select Stop, or click Menu ▾ and select Stop **to stop a running emulator**
- Right-click an AVD and select Wipe Data, or click Menu ▾ and select Wipe Data **to clear the data for an emulator**, and return it to the same state as when it was first defined

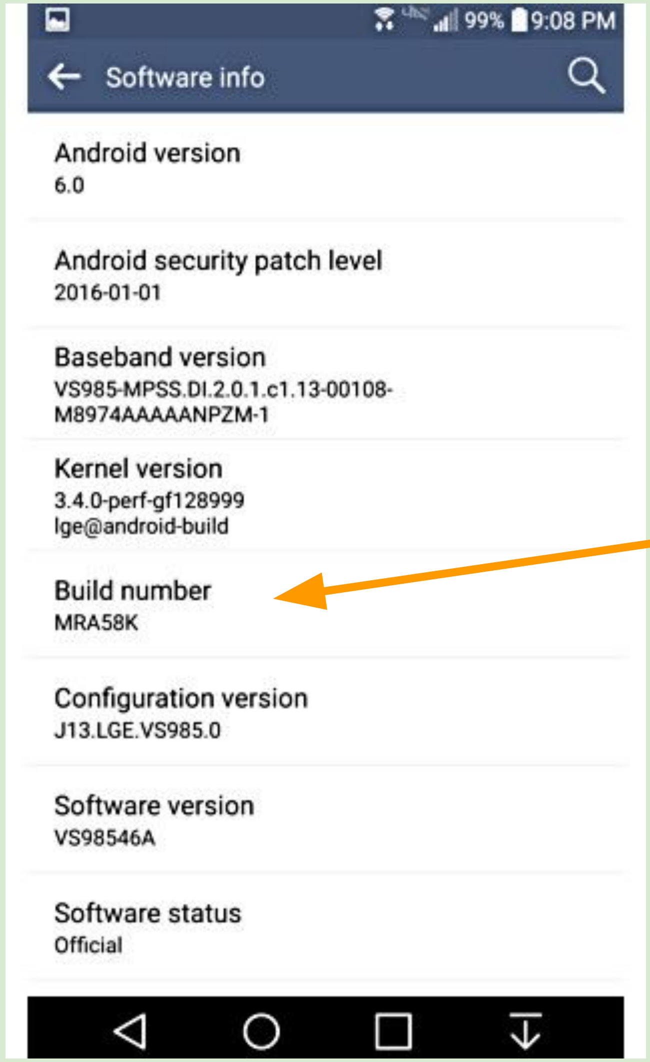
Access to emulator via ADB



- Set up the environment for ADB as described before (if not yet)
- Run AVD instance from Android Studio AVD Manager

Android physical devices setting up

Enable Developer Options


99% 9:08 PM

Software info

Android version
6.0

Android security patch level
2016-01-01

Baseband version
VS985-MPSS.DI.2.0.1.c1.13-00108-M8974AAAAANPZM-1

Kernel version
3.4.0-perf-gf128999
lge@android-build

Build number
MRA58K

Configuration version
J13.LGE.VS985.0

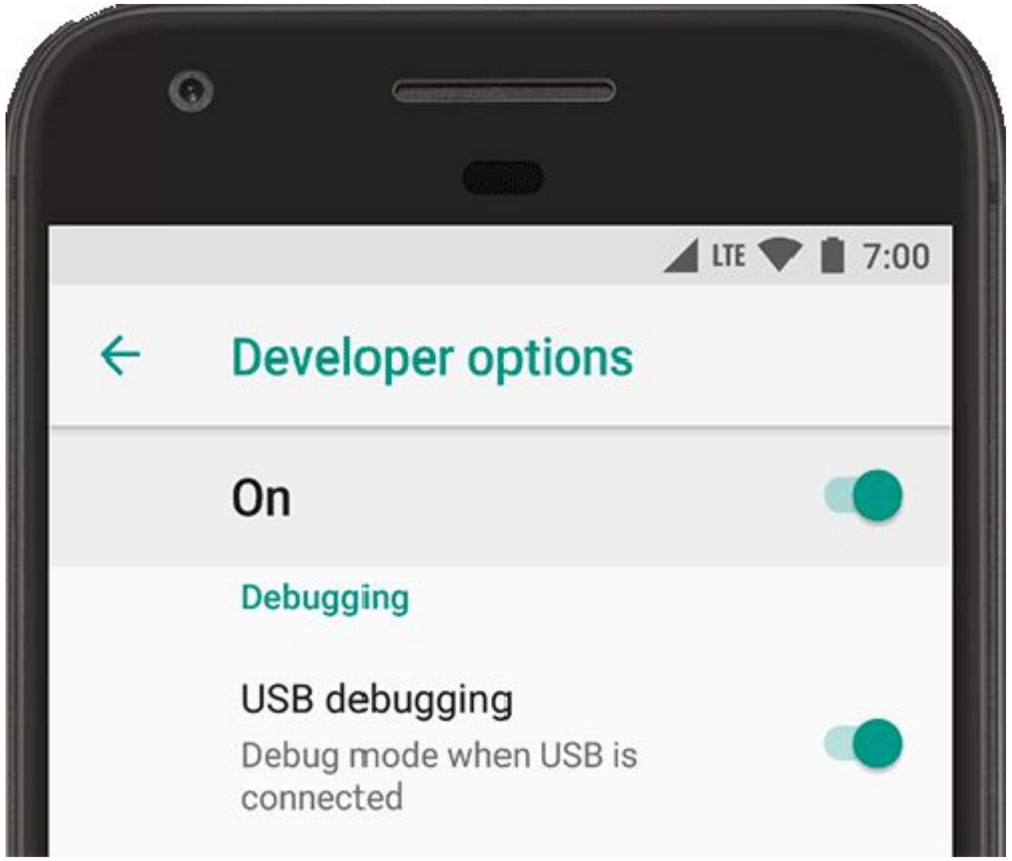
Software version
VS98546A

Software status
Official

If not yet
(starting from ver. 4.2):

- Settings > About device > Software Info
- Press Build Number 7 times

Enable options


7:00

Developer options

On

Debugging

USB debugging

Debug mode when USB is connected

- “Developer options” item appears
- Enable “On”
- Enable “USB debugging” (scroll down a little)
- Set “USB configuration” to MTP
- Full options guide

Check connection

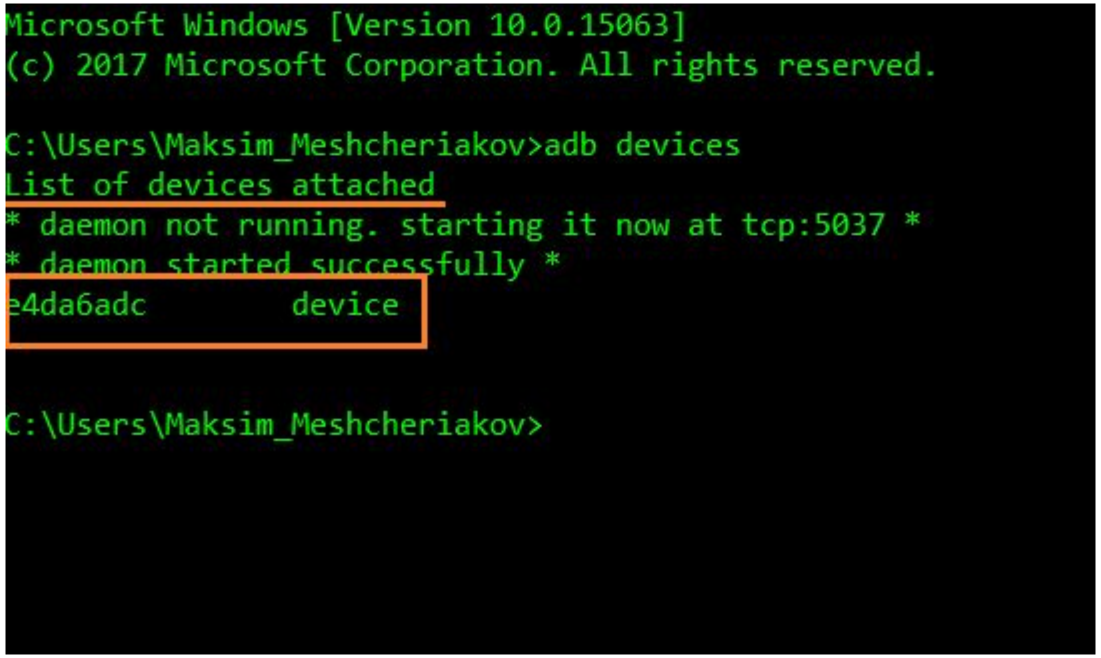
Android Debug Bridge

Use ADB to get access to Android device under testing via USB or WiFi (TCP/IP)

- Install and delete applications
- Add and remove files
- Get logs and dumps
- Get information about state of device and processes

Access via USB

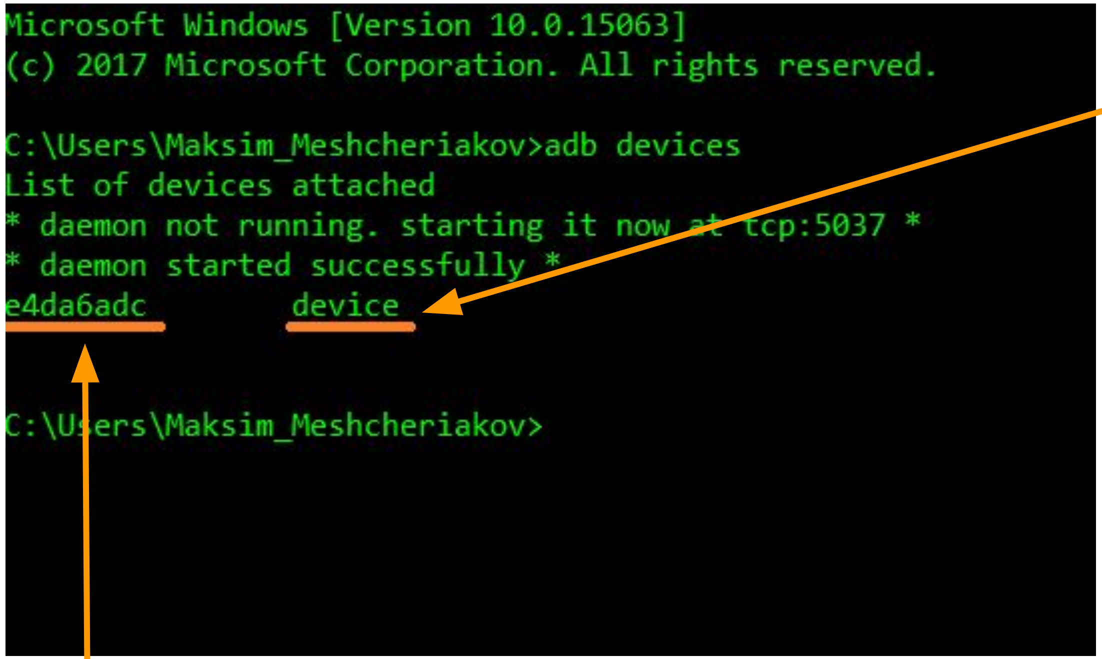
If you have properly configured environment:



```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Maksim_Meshcheriakov>adb devices
List of devices attached
* daemon not running. starting it now at tcp:5037 *
* daemon started successfully *
e4da6adc        device


C:\Users\Maksim_Meshcheriakov>
```

- Connect Android device to computer by USB
- Open command-line terminal
- Use adb devices command to verify connection

Device identification by ADB



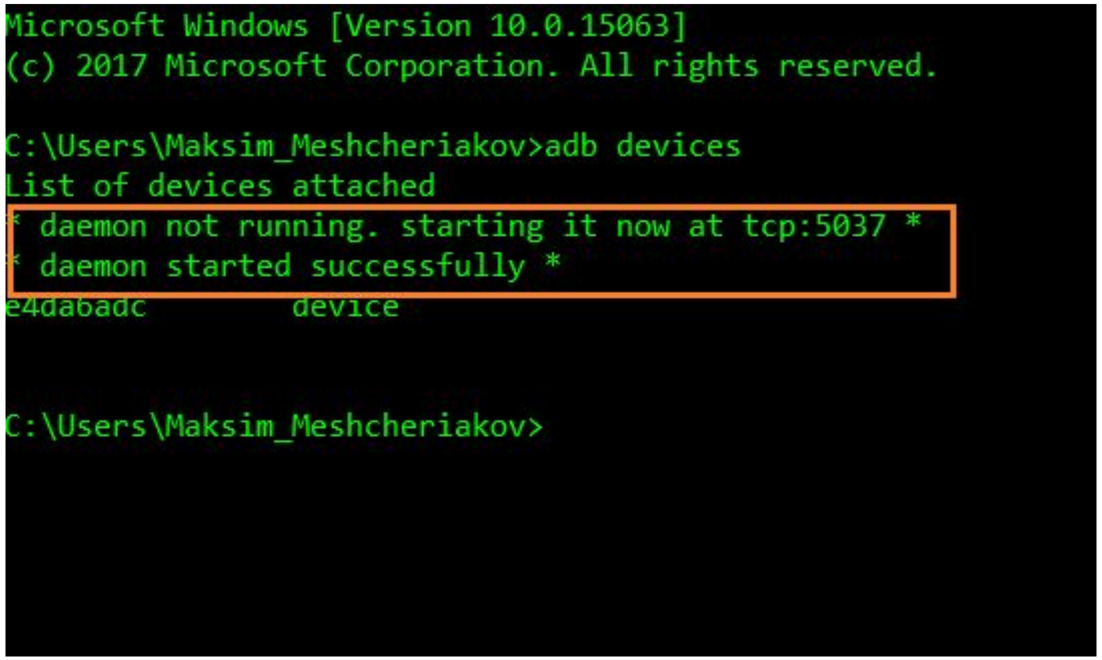
```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Maksim_Meshcheriakov>adb devices
List of devices attached
* daemon not running. starting it now at tcp:5037 *
* daemon started successfully *
e4da6adc        device

C:\Users\Maksim_Meshcheriakov>
```

Device status:

- **Device** - device connected
- **Offline** - device is not connected

Serial number: A string created by adb to uniquely identify the device

ADB is a client-server system

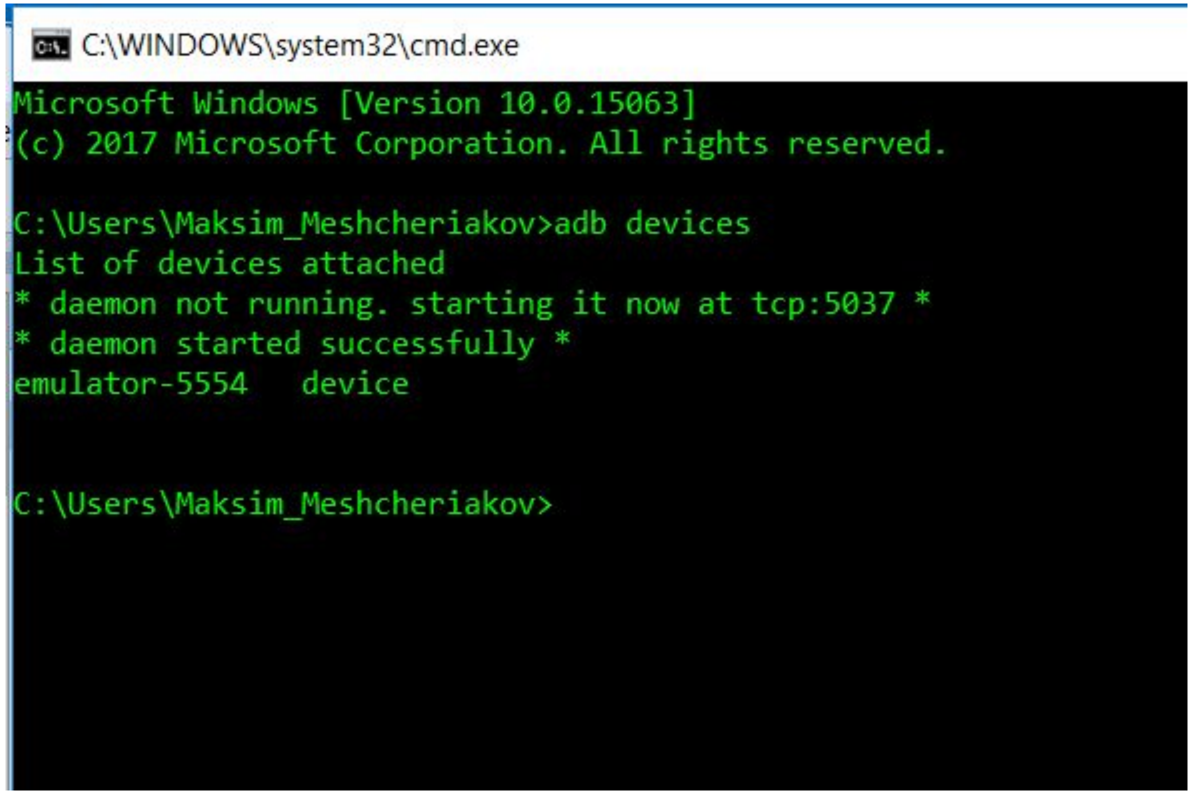


```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Maksim_Meshcheriakov>adb devices
List of devices attached
* daemon not running. starting it now at tcp:5037 *
* daemon started successfully *
e4dababdc        device


C:\Users\Maksim_Meshcheriakov>
```

- **adb kill-server**: for re-initialization of adb if something goes wrong
- **adb start-server**: the adb server start automatically on typing of some adb command

- Daemon **adbd** on the device
- Command-line client

AVD listed by ADB

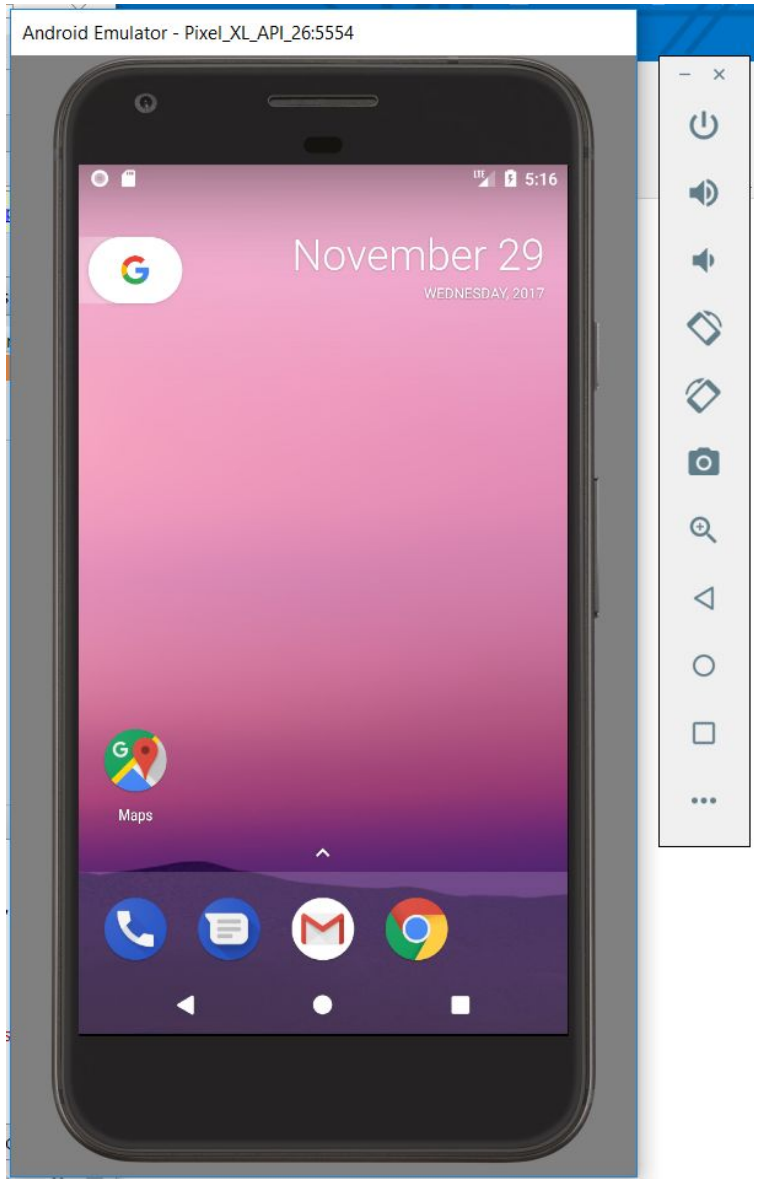


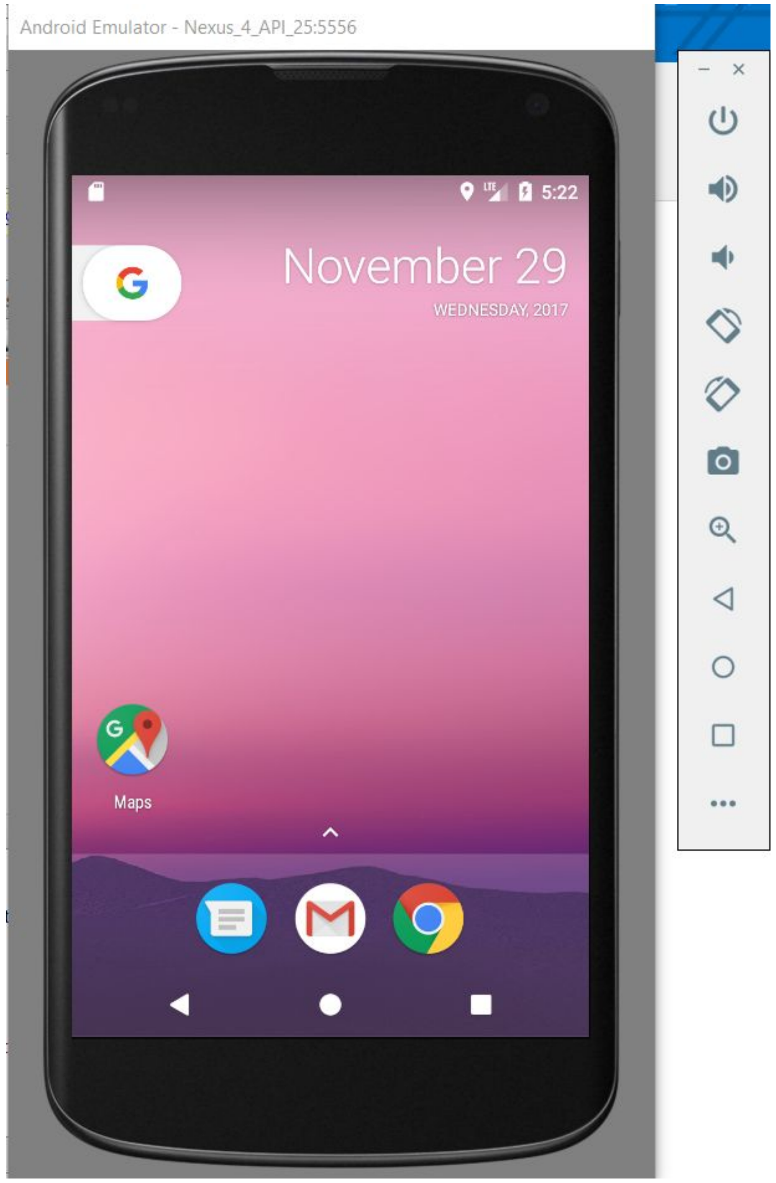
C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Maksim_Meshcheriakov>adb devices
List of devices attached
* daemon not running. starting it now at tcp:5037 *
* daemon started successfully *
emulator-5554   device


C:\Users\Maksim_Meshcheriakov>
```

Type “adb devices” in console

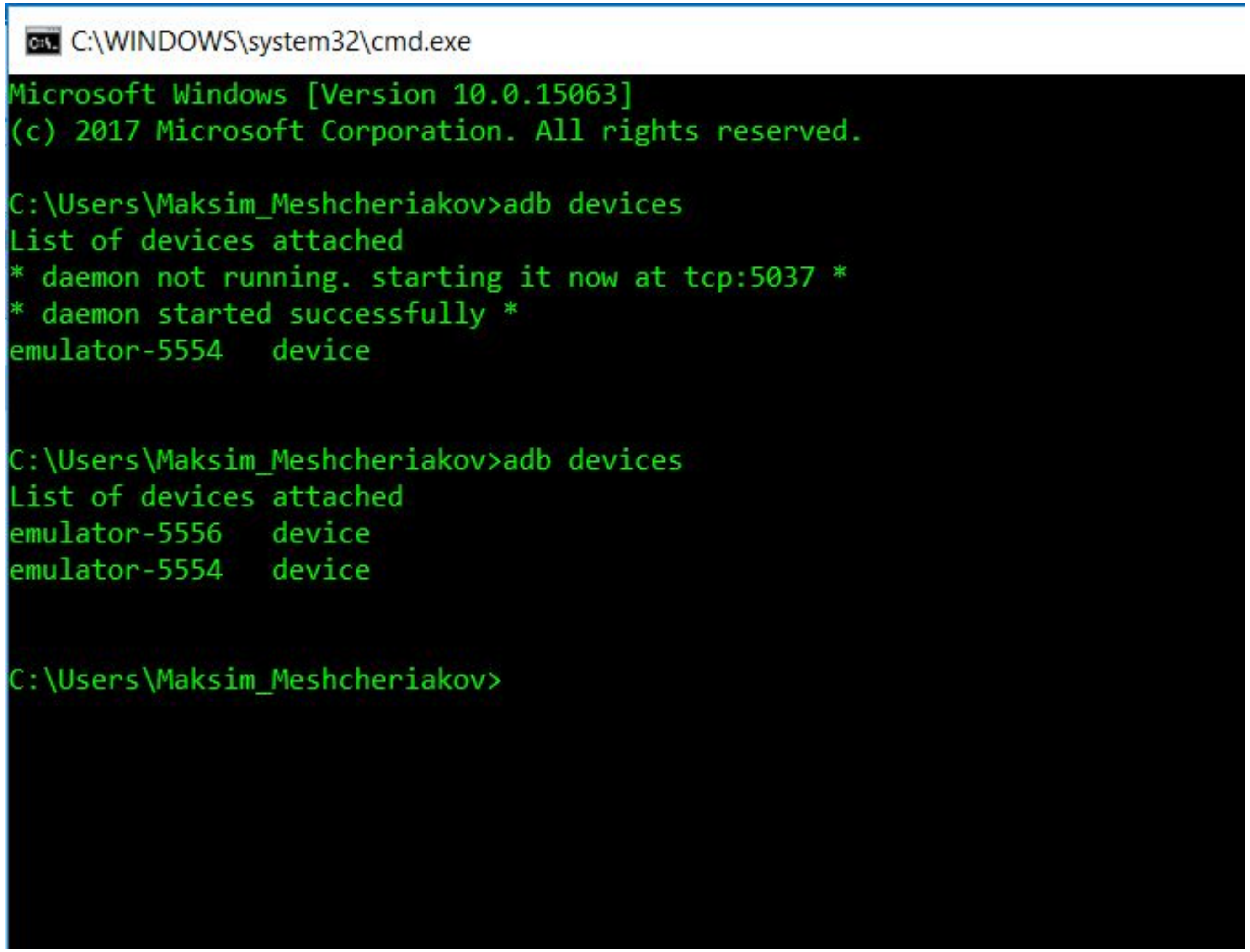
Get another one AVD


Android Emulator - Pixel_XL_API_26:5554


Android Emulator - Nexus_4_API_25:5556

Run another one AVD instance from Android Studio AVD Manager

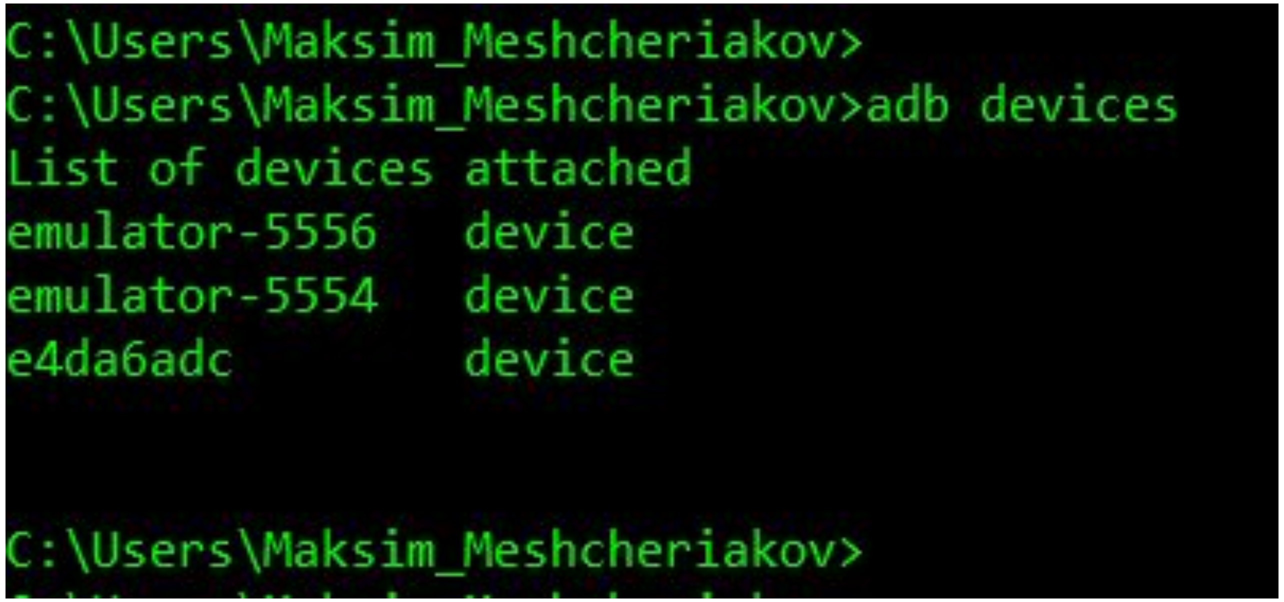
2 AVDs listed by ADB


C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Maksim_Meshcheriakov>adb devices
List of devices attached
* daemon not running. starting it now at tcp:5037 *
* daemon started successfully *
emulator-5554   device


C:\Users\Maksim_Meshcheriakov>adb devices
List of devices attached
emulator-5556   device
emulator-5554   device


C:\Users\Maksim_Meshcheriakov>
```

Type “adb devices” in console once again

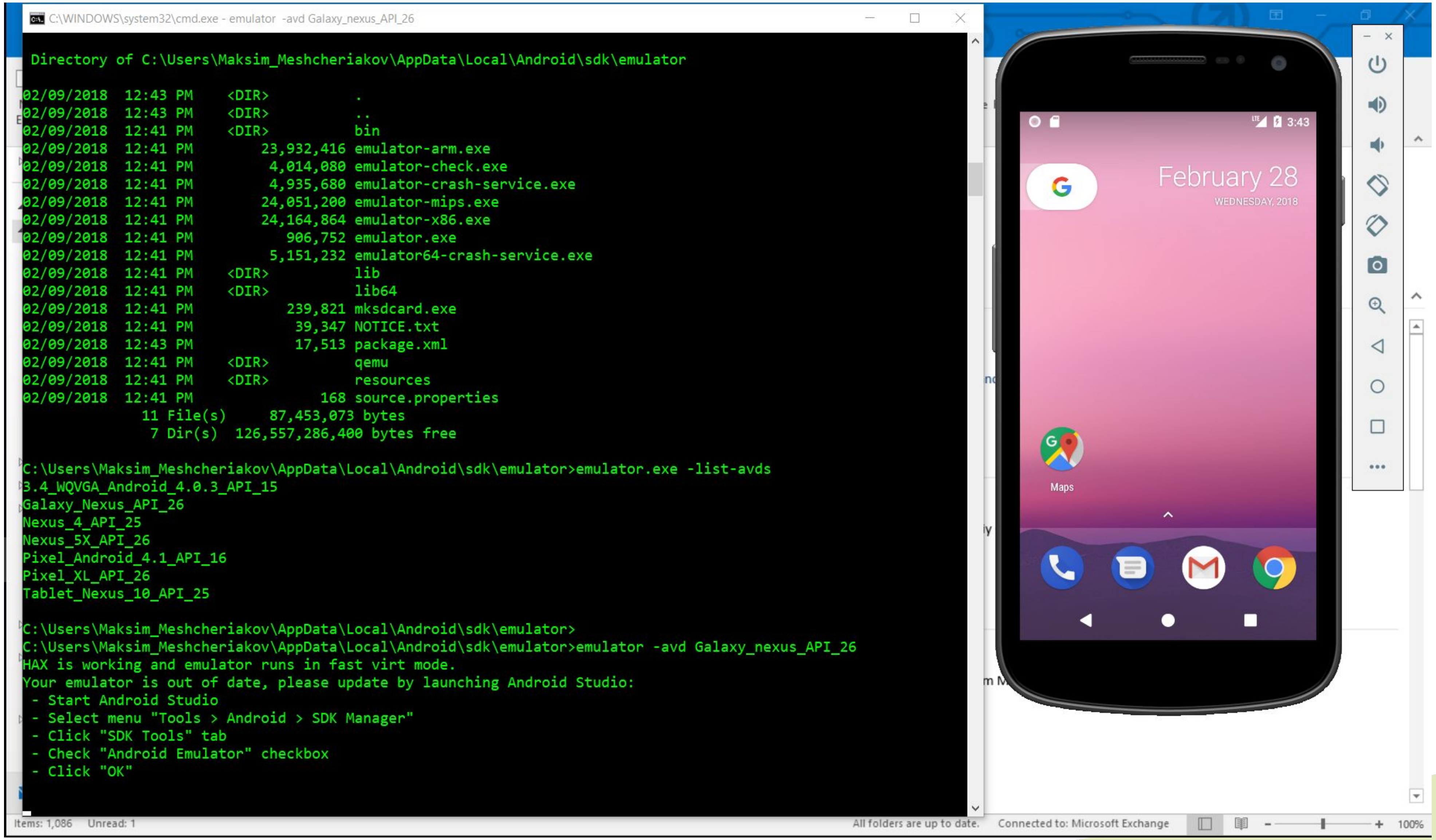
2 emulators and real device



```
C:\Users\Maksim_Meshcheriakov>
C:\Users\Maksim_Meshcheriakov>adb devices
List of devices attached
emulator-5556   device
emulator-5554   device
e4da6adc        device


C:\Users\Maksim_Meshcheriakov>
```

- Connect physical device to USB (do not stop running AVDs)
- Type “adb devices” in console once again

Run AVD via CLI



C:\WINDOWS\system32\cmd.exe - emulator -avd Galaxy_nexus_API_26

```
 Directory of C:\Users\Maksim_Meshcheriakov\AppData\Local\Android\sdk\emulator

02/09/2018  12:43 PM    <DIR>          .
02/09/2018  12:43 PM    <DIR>          ..
02/09/2018  12:41 PM    <DIR>          bin
02/09/2018  12:41 PM        23,932,416 emulator-arm.exe
02/09/2018  12:41 PM         4,014,080 emulator-check.exe
02/09/2018  12:41 PM         4,935,680 emulator-crash-service.exe
02/09/2018  12:41 PM        24,051,200 emulator-mips.exe
02/09/2018  12:41 PM        24,164,864 emulator-x86.exe
02/09/2018  12:41 PM           906,752 emulator.exe
02/09/2018  12:41 PM         5,151,232 emulator64-crash-service.exe
02/09/2018  12:41 PM    <DIR>          lib
02/09/2018  12:41 PM    <DIR>          lib64
02/09/2018  12:41 PM           239,821 mksdcard.exe
02/09/2018  12:41 PM            39,347 NOTICE.txt
02/09/2018  12:43 PM            17,513 package.xml
02/09/2018  12:41 PM    <DIR>          qemu
02/09/2018  12:41 PM    <DIR>          resources
02/09/2018  12:41 PM               168 source.properties
              11 File(s)     87,453,073 bytes
               7 Dir(s)  126,557,286,400 bytes free

C:\Users\Maksim_Meshcheriakov\AppData\Local\Android\sdk\emulator>emulator.exe -list-avds
3.4_WQVGA_Android_4.0.3_API_15
Galaxy_Nexus_API_26
Nexus_4_API_25
Nexus_5X_API_26
Pixel_Android_4.1_API_16
Pixel_XL_API_26
Tablet_Nexus_10_API_25

C:\Users\Maksim_Meshcheriakov\AppData\Local\Android\sdk\emulator>
C:\Users\Maksim_Meshcheriakov\AppData\Local\Android\sdk\emulator>emulator -avd Galaxy_nexus_API_26
HAX is working and emulator runs in fast virt mode.
Your emulator is out of date, please update by launching Android Studio:
 - Start Android Studio
 - Select menu "Tools > Android > SDK Manager"
 - Click "SDK Tools" tab
 - Check "Android Emulator" checkbox
 - Click "OK"
```

CLI for AVD

You can use CLI AVD tools as well:

- tools/bin/avdmanager.bat
  to create and maintain AVD instances
- ANDROID_HOME/emulator
  to run certain AVD instance
  https://developer.android.com/studio/run/emulator-commandline.html

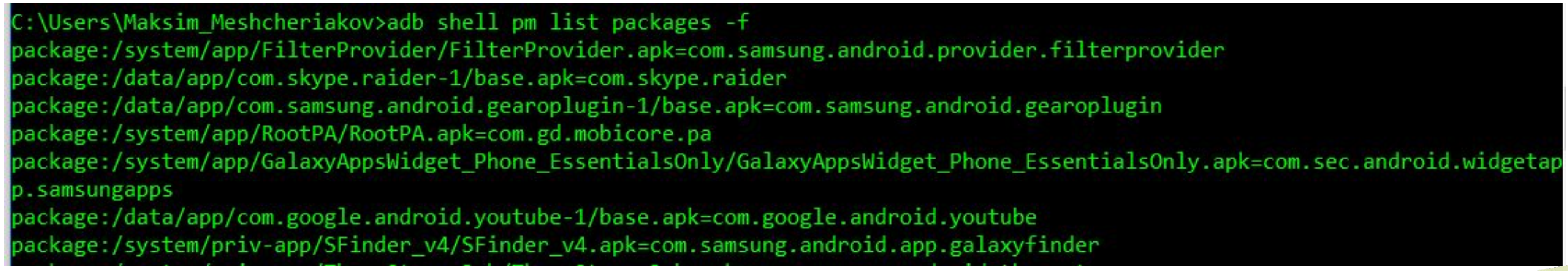
Install and remove application

```
$ adb install path_to_apk
```

```
$ adb uninstall package
```

In case of uninstallation you have to use Java package name instead of .apk filename.

```
$ adb shell pm list packages -f
```



```
C:\Users\Maksim_Meshcheriakov>adb shell pm list packages -f
package:/system/app/FilterProvider/FilterProvider.apk=com.samsung.android.provider.filterprovider
package:/data/app/com.skype.raider-1/base.apk=com.skype.raider
package:/data/app/com.samsung.android.gearoplugin-1/base.apk=com.samsung.android.gearoplugin
package:/system/app/RootPA/RootPA.apk=com.gd.mobicore.pa
package:/system/app/GalaxyAppsWidget_Phone_EssentialsOnly/GalaxyAppsWidget_Phone_EssentialsOnly.apk=com.sec.android.widgetap
p.samsungapps
package:/data/app/com.google.android.youtube-1/base.apk=com.google.android.youtube
package:/system/priv-app/SFinder_v4/SFinder_v4.apk=com.samsung.android.app.galaxyfinder
```

Copy files to/from device

- Push a file to device
```
$ adb push path2local_file path2remote_file
```

- Pull a file from device
```
$ adb pull path2remote_file path2local_file
```

- Example:
```
$ adb push foo.txt /sdcard/foo.txt
```

Get logs with Logcat

Logcat is a command-line tool that dumps a log of system messages, including stack traces when the device throws an error and messages that you have written from your app with the ‘Log’ class.

```
$ adb logcat
```

```
$ adb logcat --help
```

or

```
$ adb shell
> logcat
```

Logcat options

- Default output is ‘stdout’, but you can write output down to required file with -f <*filename*> option
- output filtering:
  Verbose (lowest) / Debug / Info / Warning / Error / Fatal / Silent (highest))
- output formatting with -v <*format*> option

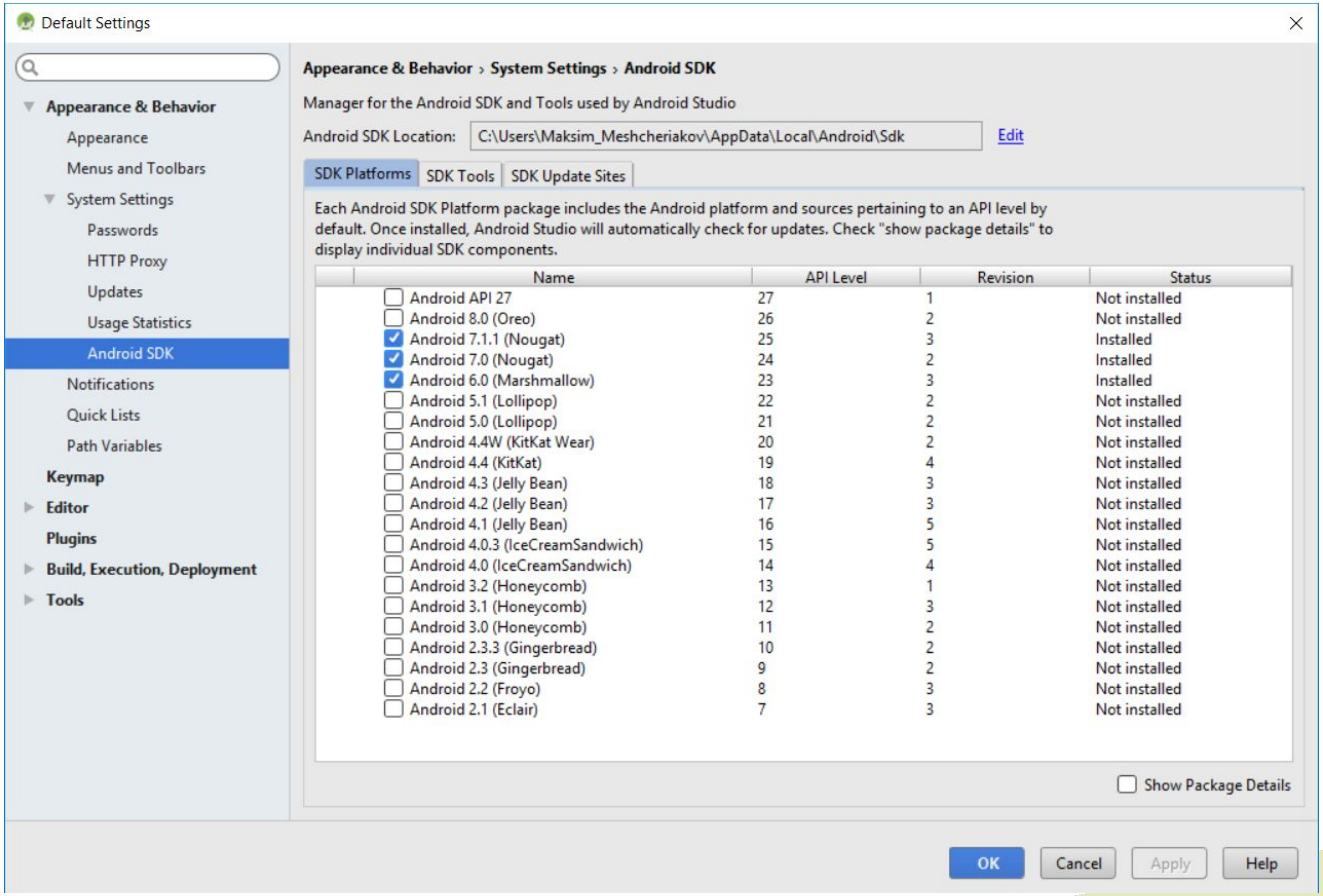
The full syntax description:
https://developer.android.com/studio/command-line/logcat.html$Syntax

Command line tools info

The root of information about Android-related command line tools:

https://developer.android.com/studio/command-line/index.html

Update platforms



Default Settings

- Appearance & Behavior
  - Appearance
  - Menus and Toolbars
  - System Settings
    - Passwords
    - HTTP Proxy
    - Updates
    - Usage Statistics
    - **Android SDK**
  - Notifications
  - Quick Lists
  - Path Variables
- Keymap
- Editor
- Plugins
- Build, Execution, Deployment
- Tools

Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: C:\Users\Maksim_Meshcheriakov\AppData\Local\Android\Sdk Edit

SDK Platforms | SDK Tools | SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

	Name	API Level	Revision	Status
☐	Android API 27	27	1	Not installed
☐	Android 8.0 (Oreo)	26	2	Not installed
☑	Android 7.1.1 (Nougat)	25	3	Installed
☑	Android 7.0 (Nougat)	24	2	Installed
☑	Android 6.0 (Marshmallow)	23	3	Installed
☐	Android 5.1 (Lollipop)	22	2	Not installed
☐	Android 5.0 (Lollipop)	21	2	Not installed
☐	Android 4.4W (KitKat Wear)	20	2	Not installed
☐	Android 4.4 (KitKat)	19	4	Not installed
☐	Android 4.3 (Jelly Bean)	18	3	Not installed
☐	Android 4.2 (Jelly Bean)	17	3	Not installed
☐	Android 4.1 (Jelly Bean)	16	5	Not installed
☐	Android 4.0.3 (IceCreamSandwich)	15	5	Not installed
☐	Android 4.0 (IceCreamSandwich)	14	4	Not installed
☐	Android 3.2 (Honeycomb)	13	1	Not installed
☐	Android 3.1 (Honeycomb)	12	3	Not installed
☐	Android 3.0 (Honeycomb)	11	2	Not installed
☐	Android 2.3.3 (Gingerbread)	10	2	Not installed
☐	Android 2.3 (Gingerbread)	9	2	Not installed
☐	Android 2.2 (Froyo)	8	3	Not installed
☐	Android 2.1 (Eclair)	7	3	Not installed

☐ Show Package Details

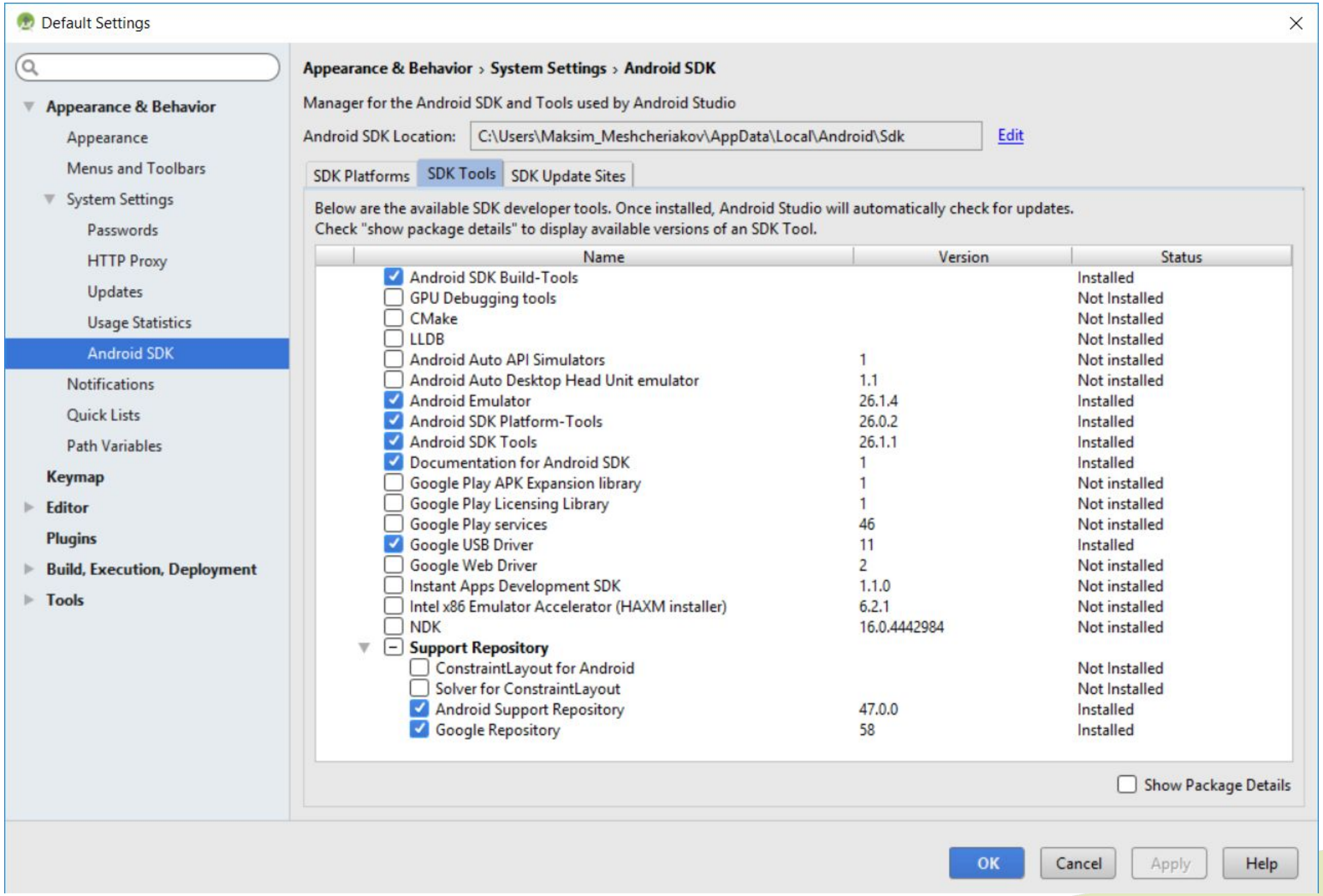
Tools >

Android >

SDK Manager >

SDK Platforms

Update tools



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: C:\Users\Maksim_Meshcheriakov\AppData\Local\Android\Sdk

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
✓ Android SDK Build-Tools		Installed
☐ GPU Debugging tools		Not Installed
☐ CMake		Not Installed
☐ LLDB		Not Installed
☐ Android Auto API Simulators	1	Not installed
☐ Android Auto Desktop Head Unit emulator	1.1	Not installed
✓ Android Emulator	26.1.4	Installed
✓ Android SDK Platform-Tools	26.0.2	Installed
✓ Android SDK Tools	26.1.1	Installed
✓ Documentation for Android SDK	1	Installed
☐ Google Play APK Expansion library	1	Not installed
☐ Google Play Licensing Library	1	Not installed
☐ Google Play services	46	Not installed
✓ Google USB Driver	11	Installed
☐ Google Web Driver	2	Not installed
☐ Instant Apps Development SDK	1.1.0	Not installed
☐ Intel x86 Emulator Accelerator (HAXM installer)	6.2.1	Not installed
☐ NDK	16.0.4442984	Not installed
Support Repository		
☐ ConstraintLayout for Android		Not Installed
☐ Solver for ConstraintLayout		Not Installed
✓ Android Support Repository	47.0.0	Installed
✓ Google Repository	58	Installed

☐ Show Package Details

Tools >

Android >

SDK Manager >

SDK Tools

Appium

Business needs

- EPAM as a global IT service company needs in clear and easy to learn and implement procedure(s) of mobile test automation that will be applicable worldwide
- These procedures should be based on a limited set of tools. These tools should be easy to learn and implement as well
- Engineers can’t learn cute new tools again and again: it’s OK for personal professional development, not to meet business needs

Required features of tool

Important: we are talking about corporate-wide tool for hundreds engineers who are working on hundreds projects with their own peculiarities

- Covers main target mobile platforms
  Android, iOS
- Use the investments made
  knowledge, expertise, processes, infrastructure, software, hardware
- Not expensive
  free open source is preferable
- Easy to learn

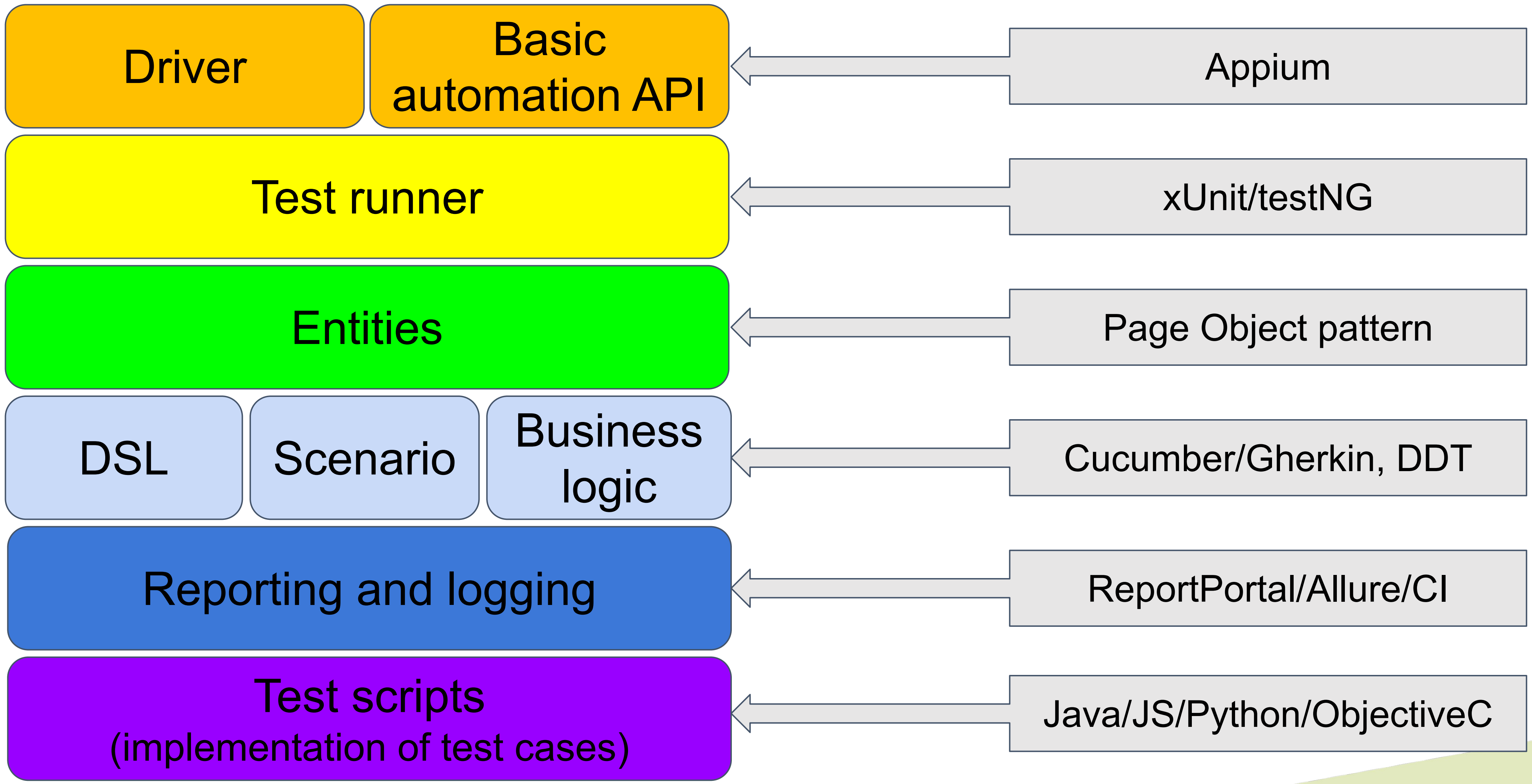
Appium advantages

Capabilities		Selenium	Appium	Espresso	XCTest UI	Calabash
Application Type Support	Mobile Web	Yes	Yes	No	No	No
	Native	No	Yes	Yes	Yes	Yes
	Hybrid	No	Yes	Yes	Yes	Yes
	Desktop Web	Yes	No	No	No	No
Supported Mobile Platforms	Android	Yes	Yes	Yes	No	Yes
	iOS	Yes	Yes	No	Yes	Yes
Supported Context	App Context	N/A	Yes	Yes	Yes	Yes
	Device Context	N/A	Yes	No	Yes	No
Scripting Development Language	Java	Yes	Yes	Yes	No	No
	Python	Yes	Yes	No	No	No
	C#	Yes	Yes	No	No	Yes
	Ruby	Yes	Yes	No	No	Yes
	Java Script	Yes	Yes	No	No	No
	Perl	Yes	Yes	No	No	No
	ObjectiveC/Swift	Yes	Yes	No	Yes	Yes

Appium advantages

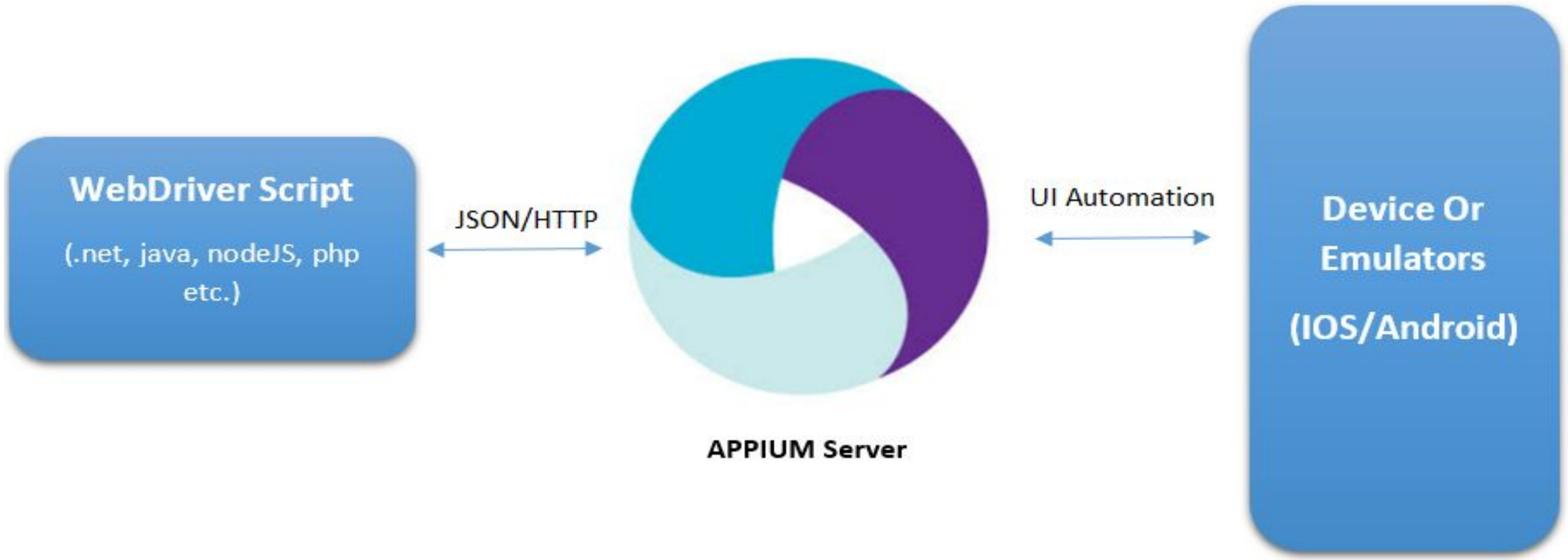
Capabilities		Selenium	Appium	Espresso	XCTest UI	Calabash
Supported Visual Object	*N/A*	*N/A*	*Yes (with extension)*	*No*	*No*	*No*
Integrated with IDEs	Eclipse	*Yes*	*Yes*	*Yes*	*No*	*Yes*
	Android Studio	*Yes*	*Yes*	*Yes*	*No*	*Yes*
	Xcode	*Yes*	*Yes*	*No*	*Yes*	*Yes*
	Visual Studio	*Yes*	*Yes*	*No*	*No*	*Yes*
	IntelliJ IDEA	*Yes*	*Yes*	*Yes*	*No*	*Yes*
Supported by Mobile OS Vendors (no gaps)	*N/A*	*No*	*No*	*Yes (Google)*	*Yes (Apple)*	*No*
Real devices	*N/A*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Simulators/Emulators	*N/A*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Automatic sync between test actions and App UI	*N/A*	*No*	*No*	*Yes*	*No*	*No*
Test Recorder	*N/A*	*No*	*No*	*Yes(Version 2.2+)*	*Yes*	*No*
Community Support	*N/A*	*Active*	*Very Active*	*Google*	*Apple*	*Average*
GitHub Rankings	*N/A*	*787 Stars*	*2444 Stars*	*NA*	*NA (*KIF 3017 Stars)*	*1930 Stars*

Layers of test automation harne



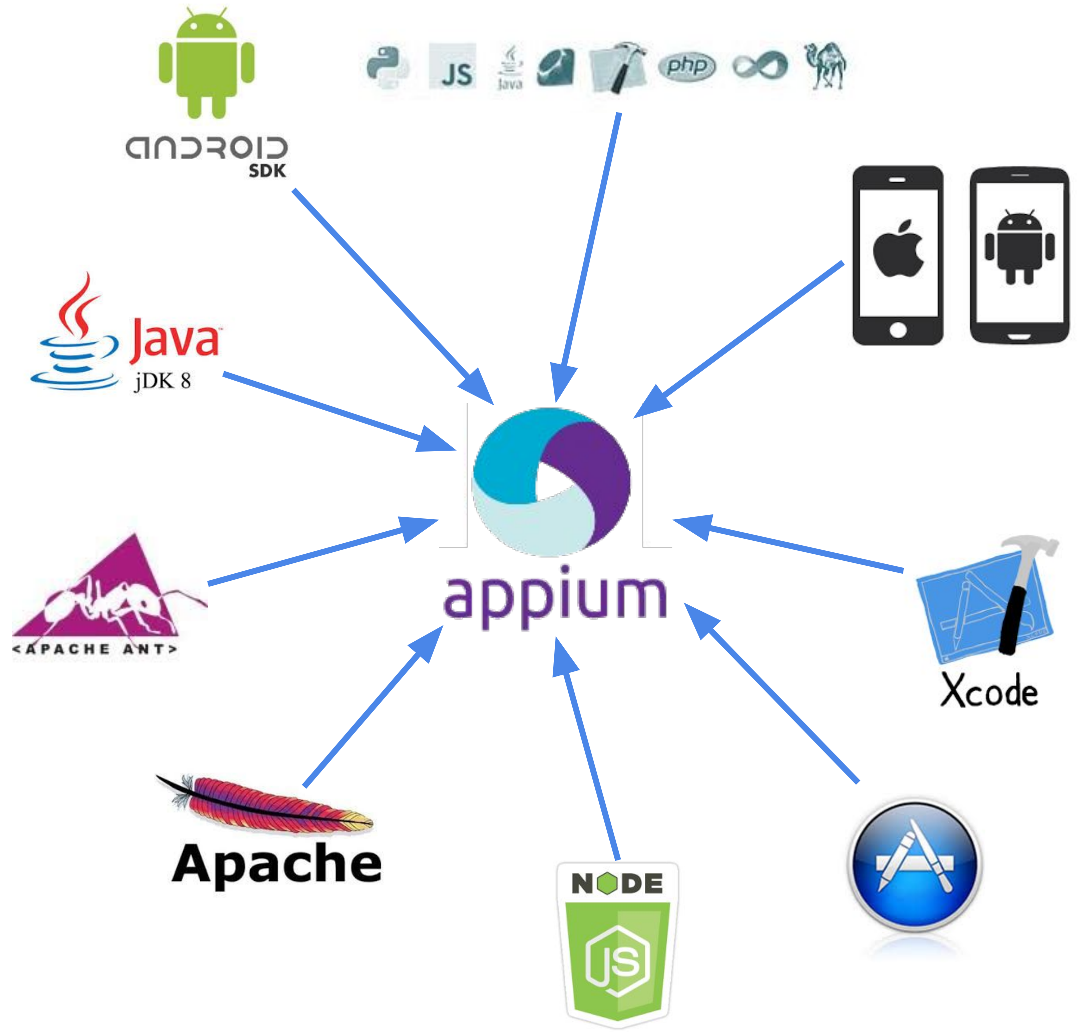
The essence of Appium

Appium server



Appium client: libraries (in Java, Ruby, Python, PHP, JavaScript, and C#) which support Appium's extensions to the WebDriver protocol

Prerequisites



About installation of JDK-8, Android SDK, mobile devices and emulators please refer to module
“General environment settings for Android platform”

Node.js

The most efficient, cross-platform way to use Appium as a node module.

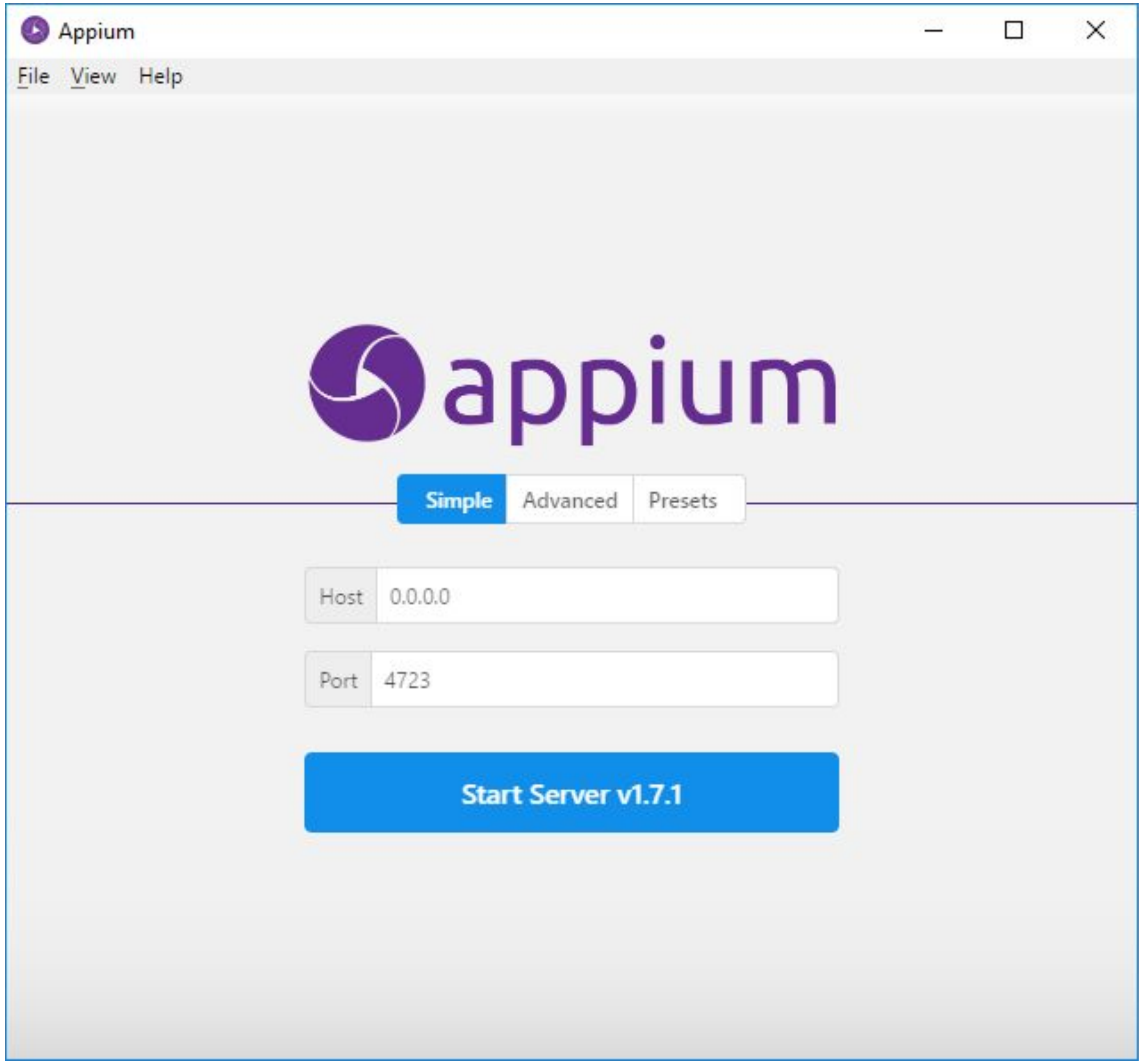
1. Download *Node.js* package suitable for your computer: https://nodejs.org/en/download/, and install it.
2. Use *appium-doctor* to check Appium preconditions:
   a. Install: > npm install -g appium-doctor
   b. Check: > appium-doctor

Appium Desktop installation

Appium *Server* and *Inspector* in desktop GUIs for Mac, Windows, and Linux

1. Download Appium Desktop from here: https://github.com/appium/appium-desktop/releases
2. Short usage instructions (scroll down to text): https://github.com/appium/appium-desktop
3. Install Appium desktop according your system rules
4. Find other Appium-related software packages here: https://github.com/appium
5. Visit appium.io to get more information

Run Appium with default settings

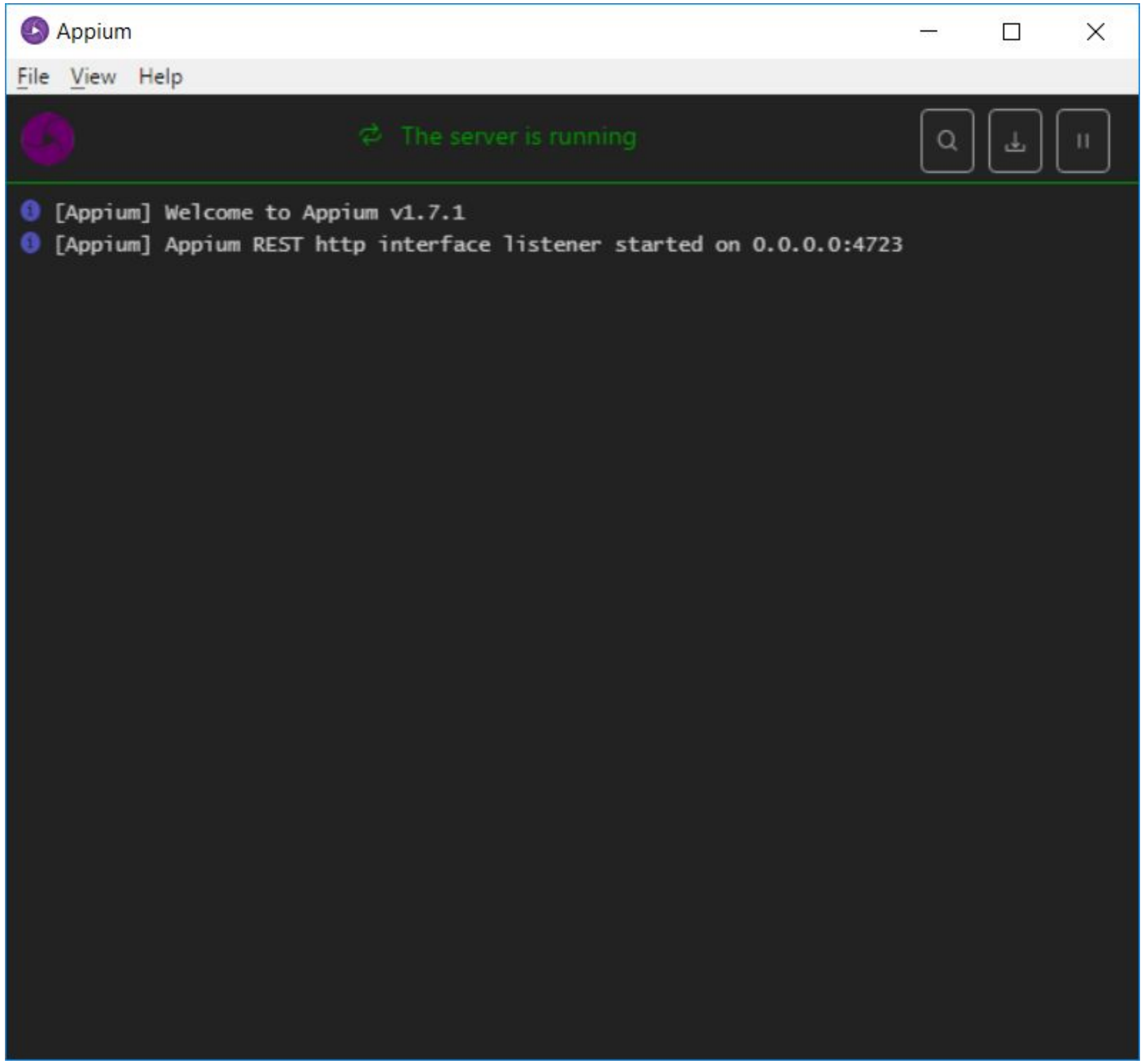


Host 0.0.0.0

Port 4723

1. Start an emulator or attach a device
2. Run Appium DT by clicking on desktop
3. Use default “simple” settings: Appium server will run locally (0.0.0.0:4723)
4. Press “Start Server x.x.x”

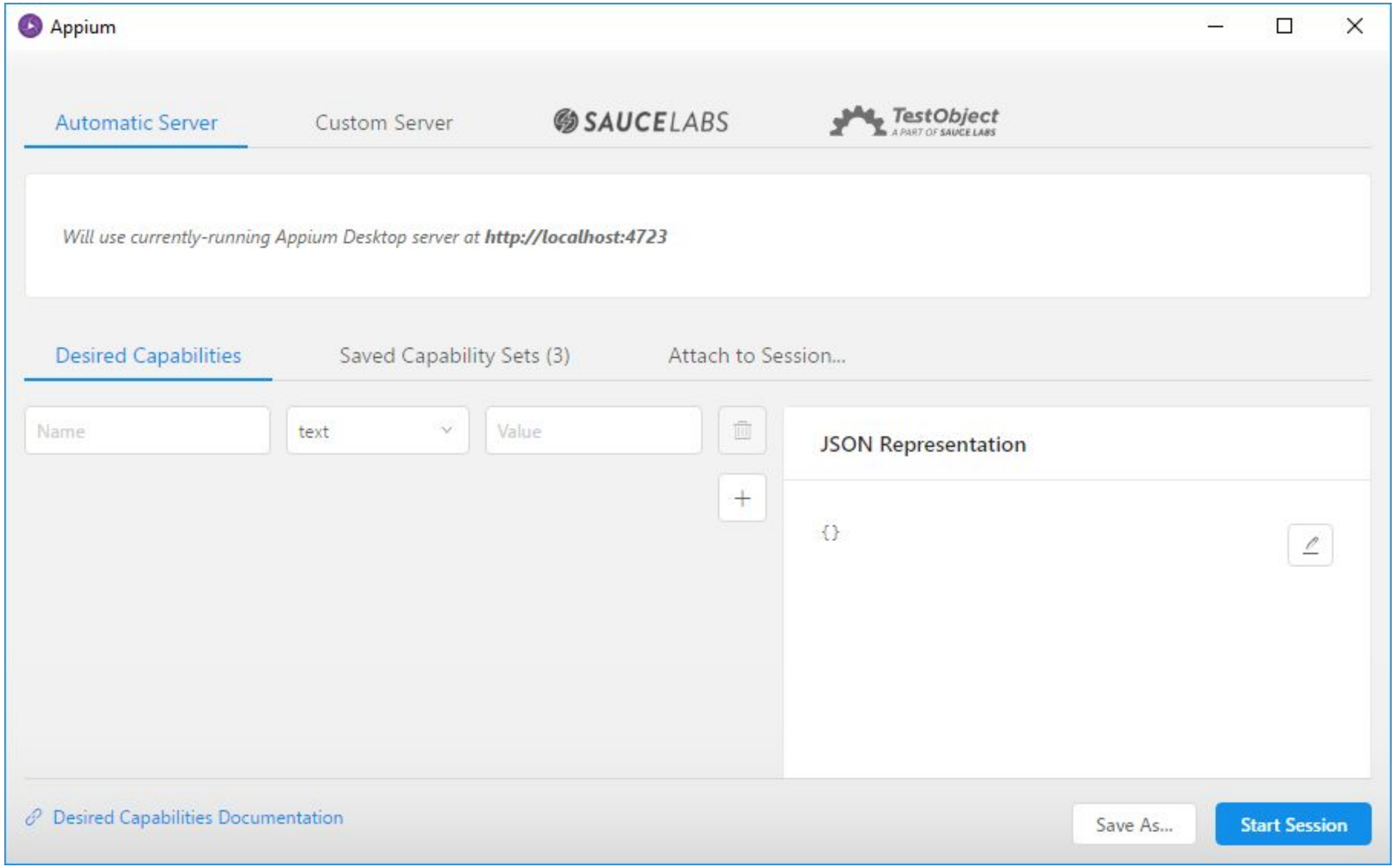
Start Appium Inspector



```
[Appium] Welcome to Appium v1.7.1
[Appium] Appium REST http interface listener started on 0.0.0.0:4723
```

Click 🔍 to start an Inspector session

Default capabilities screen


Appium

Automatic Server

Custom Server

SAUCELABS

TestObject

Will use currently-running Appium Desktop server at ***http://localhost:4723***

Desired Capabilities

Saved Capability Sets (3)

Attach to Session...

Name

text

Value

JSON Representation

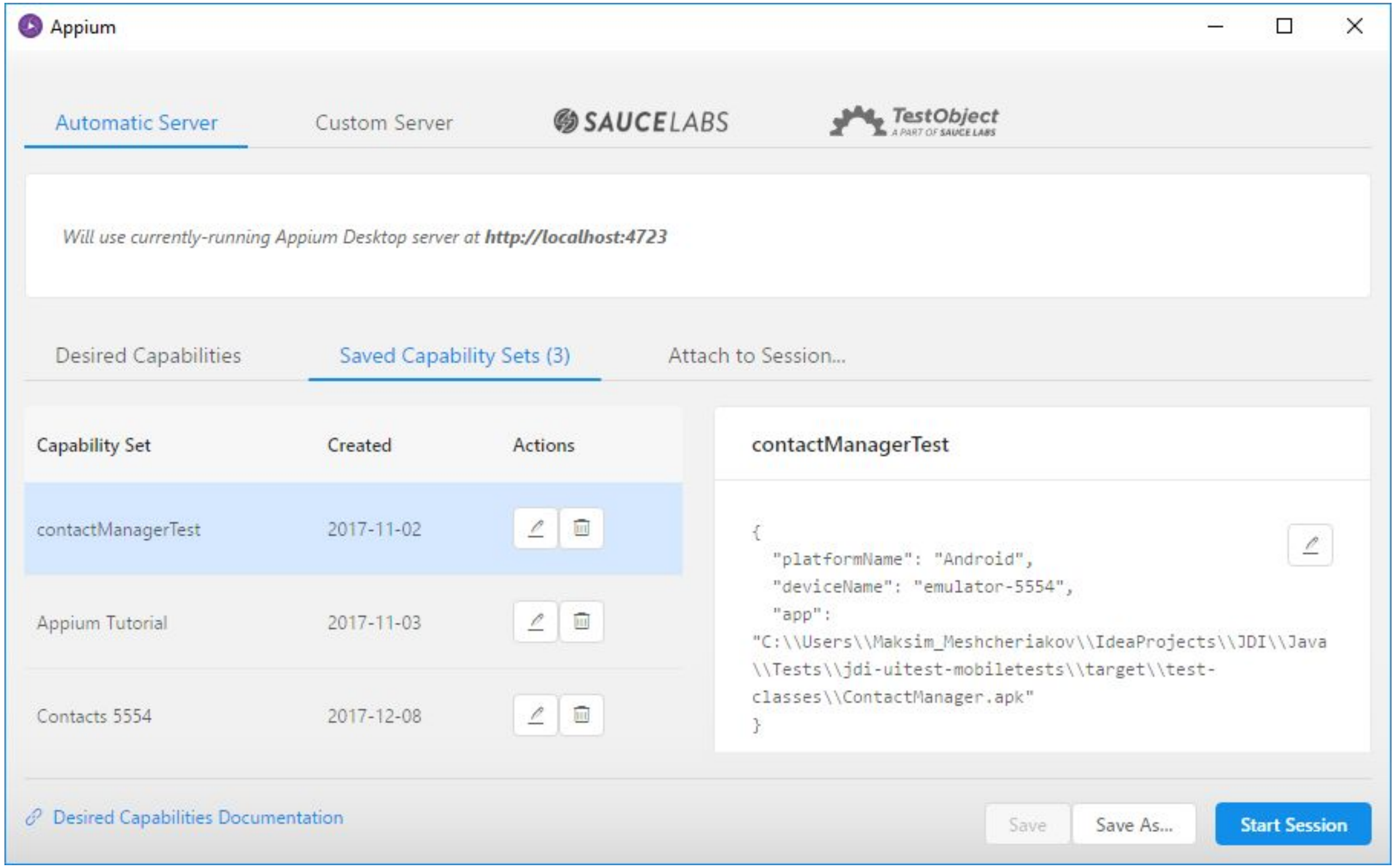
```
{}
```

Desired Capabilities Documentation

Save As...

Start Session

Set of required capabilities



Will use currently-running Appium Desktop server at ***http://localhost:4723***

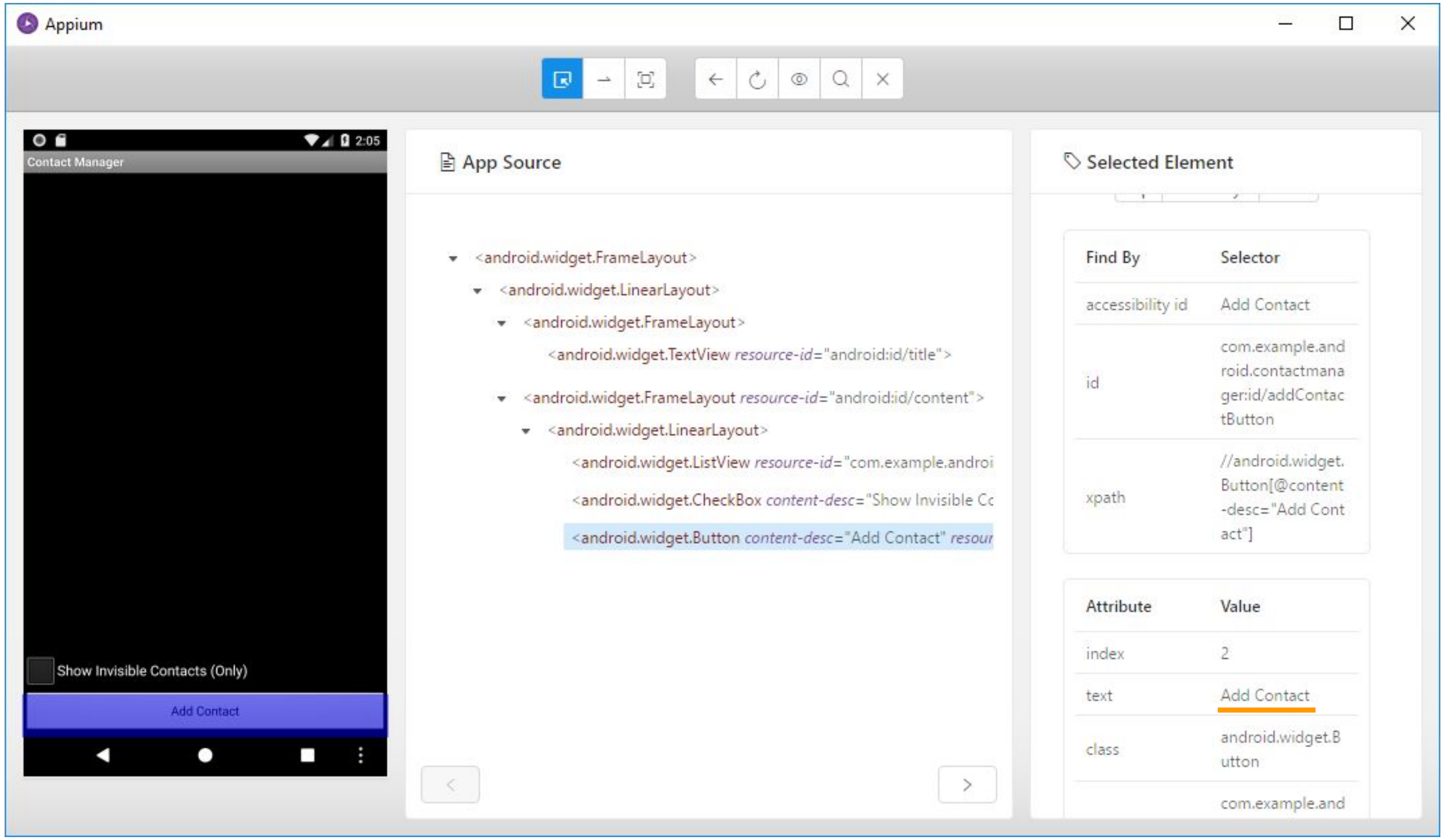
Capability Set	Created	Actions
contactManagerTest	2017-11-02	
Appium Tutorial	2017-11-03	
Contacts 5554	2017-12-08	

contactManagerTest

```
{
  "platformName": "Android",
  "deviceName": "emulator-5554",
  "app":
"C:\\Users\\Maksim_Meshcheriakov\\IdeaProjects\\JDI\\Java
\\Tests\\jdi-uitest-mobiletests\\target\\test-
classes\\ContactManager.apk"
}
```

- Use “Save As” button to store capability set for further usage
- Saved sets will be available
- Click “Start Session” button to run Inspector session

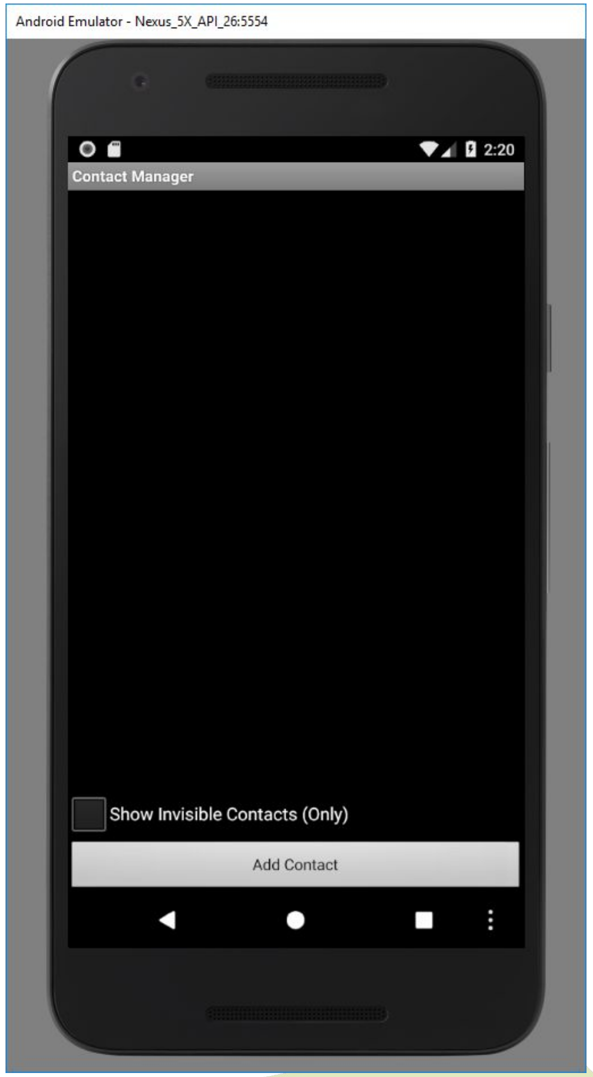
Select desired element



```
<android.widget.FrameLayout>
  <android.widget.LinearLayout>
    <android.widget.FrameLayout>
      <android.widget.TextView resource-id="android:id/title">
    <android.widget.FrameLayout resource-id="android:id/content">
      <android.widget.LinearLayout>
        <android.widget.ListView resource-id="com.example.androi
        <android.widget.CheckBox content-desc="Show Invisible Cc
        <android.widget.Button content-desc="Add Contact" resour
```

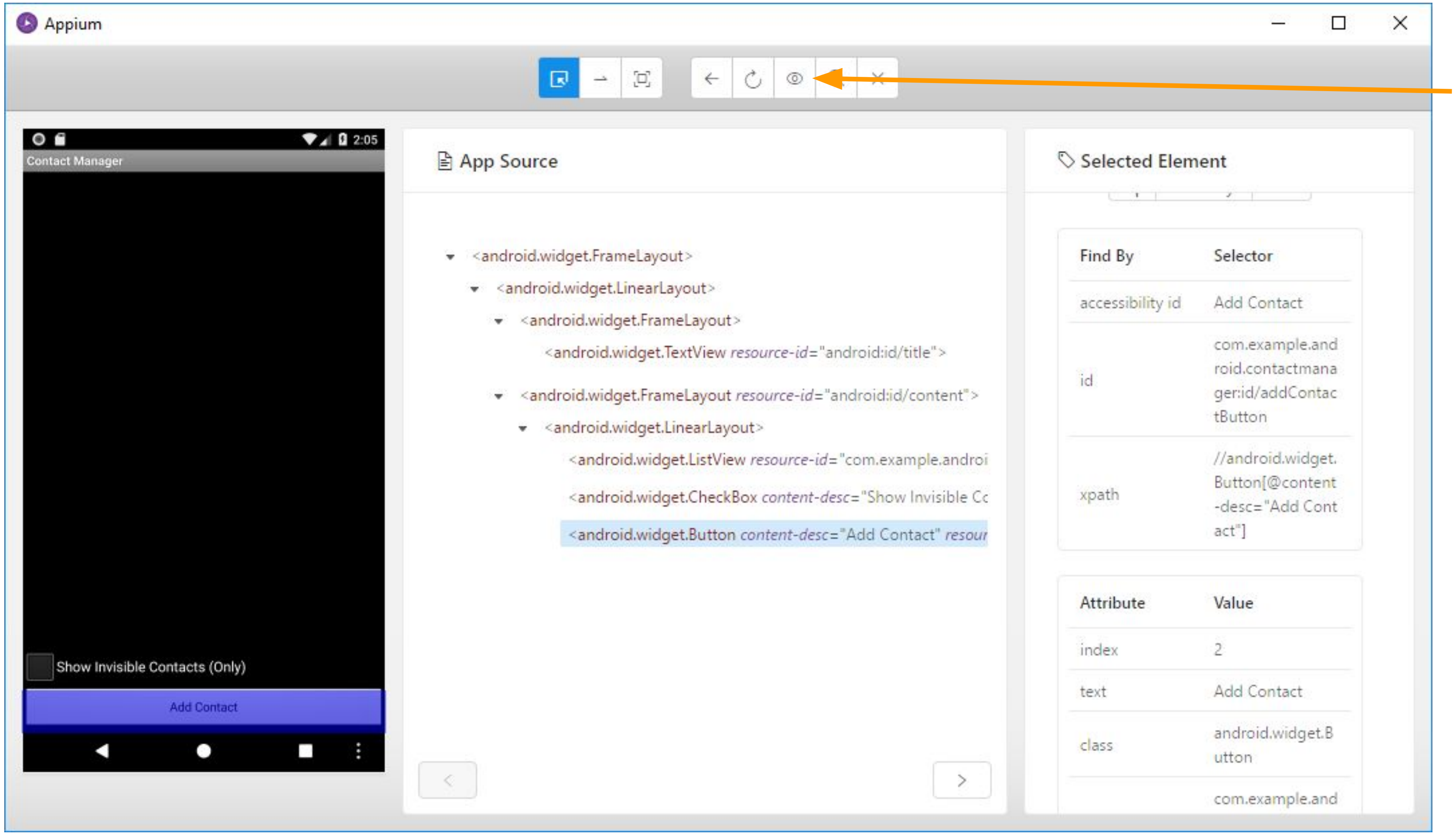
Find By	Selector
accessibility id	Add Contact
id	com.example.android.contactmanager:id/addContactButton
xpath	//android.widget.Button[@content-desc="Add Contact"]

Attribute	Value
index	2
text	Add Contact
class	android.widget.Button
	com.example.and


Android Emulator - Nexus_5X_API_26:5554
2:20
Contact Manager
Show Invisible Contacts (Only)
Add Contact

Appium Inspector tool more convenient than Device Monitor one

Recorder



App Source

```
<android.widget.FrameLayout>
  <android.widget.LinearLayout>
    <android.widget.FrameLayout>
      <android.widget.TextView resource-id="android:id/title">
    <android.widget.FrameLayout resource-id="android:id/content">
      <android.widget.LinearLayout>
        <android.widget.ListView resource-id="com.example.androi
        <android.widget.CheckBox content-desc="Show Invisible Cc
        <android.widget.Button content-desc="Add Contact" resour
```

Selected Element

Find By	Selector
accessibility id	Add Contact
id	com.example.android.contactmanager:id/addContactButton
xpath	//android.widget.Button[@content-desc="Add Contact"]

Attribute	Value
index	2
text	Add Contact
class	android.widget.Button
	com.example.and

Recorder tool:

- not for production code
- help explore Appium API
- demonstration

It is a learning tool, not a robust code generation feature

Plan

- Basic tools
  - Physical devices, emulators and cloud solutions
  - Appium
- How to write auto-tests
- Basics of Mobile Cloud Services
- Tips and tricks

QUESTIONS

maxim.mescheryakov

maksim_meshcheriakov@epam.com