Install Android toolchain

ChatGPT installed a **command-line Android toolchain**, without Android Studio:

- **Java:** Used your existing **OpenJDK 17**. The default Java 21 installation lacked the compiler needed by Gradle.
- **Gradle 8.11.1:** Downloaded and extracted into the project's `.tooling/gradle-8.11.1`.
- **Android SDK command-line tools:** Downloaded Google's Linux ZIP and used its `sdkmanager` to install:
  - SDK Platform **35**
  - Build-Tools **35.0.0**
  - Platform-Tools, including ADB
- **SDK location:** Initially used `/tmp/sshcopy-android-sdk`. After that directory disappeared, I reinstalled the SDK permanently inside the project at `.tooling/android-sdk`.
- **Project configuration:** Updated [local.properties](/home/roland/Dokumente/ChatGPT/Android akp/local.properties) to point to the permanent SDK.
- **Build script:** Created [build-apk.sh](/home/roland/Dokumente/ChatGPT/Android akp/build-apk.sh) to configure Java, SDK and cache paths, build the debug APK, run lint, and copy the result to `SSHCopy.apk`.
- **Dependencies:** Gradle downloaded Android Gradle Plugin **8.9.1**, **JSch** for SSH/SFTP, and **Bouncy Castle** for cryptography.
- **Verification:** Built the APK, ran lint and file-selection tests, and verified the APK signature.

Build-apk.sh

```
#!/usr/bin/env bash
set -euo pipefail
cd "$(dirname "$0")"
export JAVA_HOME="${JAVA_HOME:-/usr/lib/jvm/java-17-openjdk-amd64}"
export ANDROID_HOME="$PWD/.tooling/android-sdk"
if [[ ! -f "$ANDROID_HOME/platforms/android-35/android.jar" ]]; then
    echo "Android SDK platform 35 is missing from $ANDROID_HOME" >&2
    exit 1
fi
export GRADLE_USER_HOME="$PWD/.tooling/gradle-home"
export ANDROID_USER_HOME="$PWD/.tooling/android-user"
.tooling/gradle-8.11.1/bin/gradle --no-daemon :app:assembleDebug :app:lintDebug
cp app/build/outputs/apk/debug/app-debug.apk SSHCopy.apk
```

how to install the android dev toolchain on ubuntu 24.04 without android studio

You can install **just Java and the Android SDK command-line tools**. These steps assume Ubuntu 24.04 on an Intel/AMD PC.

1. Install Java and utilities

```
sudo apt update
sudo apt install openjdk-21-jdk unzip
```

2. Download the command-line tools

On the Android downloads page, find **Command line tools only** and download the **Linux ZIP**.

For a fresh installation, run these commands, replacing `YOUR_VERSION` with the downloaded filename's version:

```
mkdir -p "$HOME/Android/Sdk/cmdline-tools"
unzip "$HOME/Downloads/commandlinetools-linux-YOUR_VERSION_latest.zip" \
  -d "$HOME/Android/Sdk/cmdline-tools"

mv "$HOME/Android/Sdk/cmdline-tools/cmdline-tools" \
   "$HOME/Android/Sdk/cmdline-tools/latest"
```

The resulting executable must be at:

```
~/Android/Sdk/cmdline-tools/latest/bin/sdkmanager
```

Google currently marks `sdkmanager` deprecated in favor of the newer Android CLI, but still documents this standalone SDK installation method. Official instructions

3. Configure your shell

Add these lines to `~/.bashrc`. Remove the Snap-based `JAVA_HOME` from my previous answer if you added it.

```
export JAVA_HOME="/usr/lib/jvm/java-21-openjdk-amd64"
export ANDROID_HOME="$HOME/Android/Sdk"
export PATH="$JAVA_HOME/bin:$ANDROID_HOME/platform-tools:$ANDROID_HOME/cmdline-tools/latest/bin:$ANDROID_HOME/emulator:$PATH"
```

Then reload:

```
source ~/.bashrc
java -version
sdkmanager --version
```

4. Install SDK packages and accept licenses

This example installs API 36. Change the platform and build-tools versions to match your project:

```
sdkmanager --licenses
sdkmanager "platform-tools" "platforms;android-36" "build-tools;36.0.0"
```

Review and accept the license prompts. To see other available versions:

```
sdkmanager --list
```

5. Enable USB debugging for a phone

```
sudo apt install android-sdk-platform-tools-common
sudo usermod -aG plugdev "$USER"
```

Log out and back in. Enable **USB debugging** in the phone's Developer options, connect it, and accept the authorization prompt.

```
adb devices
```

It should show your phone with status `device`. USB setup documentation

6. Build an existing project

From the project directory:

```
./gradlew assembleDebug
./gradlew installDebug
```

Use the project's Gradle wrapper. Its Gradle/Android plugin versions determine the required Java version; older projects may need JDK 17 instead of 21. Java requirements

You now have the SDK, compiler/build tools, and ADB without Android Studio.

> how to create a debug version of an apk
>
> I'll check the command-line build instructions for a debug APK.
>
> Searched the web for `https://developer.android.com/build/building-cmdline`

If you have the **Android project's source code**, run this from its root directory—the folder containing `gradlew`:

```
./gradlew assembleDebug
```

If you get "Permission denied":

```
chmod +x gradlew
./gradlew assembleDebug
```

For a standard project with an `app` module, the APK appears at:

```
app/build/outputs/apk/debug/app-debug.apk
```

It's automatically signed with a debug key and supports debugging. No manual signing is needed. Official build instructions

To install it on a connected phone with USB debugging enabled:

```
adb install -r app/build/outputs/apk/debug/app-debug.apk
```

Or build and install in one step:

```
./gradlew installDebug
```

If you mean **you only have an existing APK, without the source project**, that requires a different process; these commands rebuild from source.