Fiche Complète : Génération d'un APK Kivy sous WSL2 (Ubuntu)

MAECKELBERG Julien

9 janvier 2026

Table des matières

1 Objectif

Compiler un projet Kivy/Python (fichiers `.py` et `.kv`) en APK Android fonctionnel depuis Ubuntu sous WSL2.

Attention

Important : Nous partons du principe que le projet se trouve dans le dossier "Projet" au chemin suivant : $C:\backslash Users\backslash maeck\backslash Desktop\backslash Projet$. Adaptez la suite en fonction de votre cas.

2 Structure du projet

Structure de votre projet :

```
C:\Users\maeck\Desktop\Projet\
            buildozer.spec          (fichier de configuration)
            fusee_test.jpg          (image)
            guide_1.jpg             (image)
            guide_2.jpg             (image)
            icon.png                (image)
            main.py                 (fichier principal Python)
            manifest_pmi.html       (fichierfichier HTML)
            noseConeShape.py        (module Python)
            presplash.png           (ecran de demarrage)
            shapeExtraction.py      (module Python)
            spockApi.py             (module Python)
            vulcain.kv              (fichier interface Kivy)
```

3 Configuration initiale (première fois uniquement)

3.1 Copier le projet dans le système Linux

Attention

Important : Ne compilez jamais dans `/mnt/c/...`. Travaillez toujours dans votre répertoire Ubuntu local (`/home/...`).

PREMIÈRE FOIS UNIQUEMENT

```
mkdir -p ~/projets/vulcain
cp -r /mnt/c/Users/maeck/Desktop/vulcain/* ~/projets/vulcain/
cd ~/projets/vulcain
ls
```

Vous devez voir :

```
buildozer.spec  fusee_test.jpg  guide_1.jpg  guide_2.jpg  icon.png
main.py  manifest_pmi.html  noseConeShape.py  presplash.png
shapeExtraction.py  spockApi.py  vulcain.kv
```

3.2 Installation des dépendances système

PREMIÈRE FOIS UNIQUEMENT

Ces outils sont nécessaires à la compilation Android.

```
sudo apt update
sudo apt install -y \
    openjdk-17-jdk \
    python3.10-venv \
    git \
    cmake \
    build-essential \
    pkg-config \
    autoconf \
    automake \
    libtool \
    zip \
    unzip \
    adb
```

3.3 Vérification des installations

PREMIÈRE FOIS UNIQUEMENT

Vérifiez les versions (elles doivent exister et ne pas afficher d'erreur) :

```
java -version
javac -version
git --version
adb version
cmake --version
python3.10 --version
```

Résultat attendu :

```
openjdk version "17..."
javac 17...
git version 2.x.x
Android Debug Bridge version 1.0.41
cmake version 3.x.x
Python 3.10.x
```

3.4 Création de l'environnement virtuel

PREMIÈRE FOIS UNIQUEMENT

Créer l'environnement virtuel :

```
python3.10 -m venv .venv
```

Information

À chaque session (première fois et recompilations), vous devrez activer l'environnement virtuel :

```
source .venv/bin/activate
```

Le terminal doit afficher : `(.venv)`

3.5 Installation des dépendances Python

PREMIÈRE FOIS UNIQUEMENT

Installez les outils nécessaires à Buildozer et à la compilation :

```
python -m pip install --upgrade pip
pip install --upgrade pip setuptools wheel
pip install "Cython==0.29.36" "buildozer==1.5.0"
pip install appdirs colorama jinja2 cython
pip install packaging toml
pip install build
pip install python-for-android
```

Note : Ces versions sont stables et compatibles avec Python 3.10.

4 Première compilation

PREMIÈRE FOIS UNIQUEMENT

Avant toute première compilation, effectuez un nettoyage :

```
buildozer android clean
```

Puis lancez la compilation :

```
buildozer android debug
sed -i 's|</application>|    <provider android:name="androidx.core.
    content.FileProvider" android:authorities="${applicationId}.
    fileprovider" android:exported="false" android:grantUriPermissions="
    true"><meta-data android:name="android.support.FILE_PROVIDER_PATHS"
    android:resource="@xml/file_paths" /></provider>\n    </application
    >|' .buildozer/android/platform/build-arm64-v8a_armeabi-v7a/dists/
    vulcain/src/main/AndroidManifest.xml
cd .buildozer/android/platform/build-arm64-v8a_armeabi-v7a/dists/
    vulcain
./gradlew assembleDebug
cd ~/projets/vulcain
```

Information

Buildozer téléchargera automatiquement :

- Android SDK
- Android NDK
- Outils Gradle
- Librairies Python-for-Android

Cette étape peut durer **30 minutes à 2 heures** la première fois.

5 Recompilations suivantes

RECOMPILATIONS SUIVANTES

Après avoir modifié votre code, voici les étapes pour recompiler :

1. **Naviguer vers le projet :**

```
cd ~/projets/vulcain

```

2. **Activer l'environnement virtuel :**

```
source .venv/bin/activate

```

3. **Copier les fichiers modifiés (si nécessaire) :**

```
cp -r /mnt/c/Users/maeck/Desktop/Projet/* ~/projets/vulcain/

```

4. **Nettoyer :**

```
buildozer android clean

```

5. **Compiler :**

```
buildozer android debug
sed -i 's|</application>|    <provider android:name="androidx.core.content.FileProvider" android:authorities="${applicationId}.fileprovider" android:exported="false" android:grantUriPermissions="true"><meta-data android:name="android.support.FILE_PROVIDER_PATHS" android:resource="@xml/file_paths" /></provider>\n    </application>|' .buildozer/android/platform/build-arm64-v8a_armeabi-v7a/dists/vulcain/src/main/AndroidManifest.xml
cd .buildozer/android/platform/build-arm64-v8a_armeabi-v7a/dists/vulcain
./gradlew assembleDebug
cd ~/projets/vulcain

```

Information

Les recompilations sont beaucoup plus rapides (30 minutes environ) car les dépendances sont déjà téléchargées.

6 Récupération de l'APK

Information

Après une compilation réussie, l'APK se trouve dans :

```
~/projets/vulcain/.buildozer/android/platform/build-arm64-v8a_armeabi-
    v7a/dists/vulcain/build/outputs/apk/debug
```

Le fichier s'appelle généralement :

```
vulcain-debug.apk
```

Vous pouvez le copier vers Windows avec :

```
1 cp .buildozer/android/platform/build-arm64-v8a_armeabi-v7a/dists/
    vulcain/build/outputs/apk/debug/*.apk /mnt/c/Users/maeck/Desktop/
```