

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

MAECKELBERG Julien

9 janvier 2026

## Table des matières

<b>1</b>	<b>Objectif</b>	<b>2</b>
<b>2</b>	<b>Structure du projet</b>	<b>2</b>
<b>3</b>	<b>Configuration initiale (première fois uniquement)</b>	<b>2</b>
3.1	Copier le projet dans le système Linux . . . . .	2
3.2	Installation des dépendances système . . . . .	3
3.3	Vérification des installations . . . . .	3
3.4	Création de l'environnement virtuel . . . . .	3
3.5	Installation des dépendances Python . . . . .	4
<b>4</b>	<b>Première compilation</b>	<b>4</b>
<b>5</b>	<b>Recompilations suivantes</b>	<b>5</b>
<b>6</b>	<b>Récupération de l'APK</b>	<b>6</b>

# 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:\Users\maeck\Desktop\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       (fichier HTML)  
    noseConeShape.py        (module Python)  
    presplash.png           (ecran de démarrage)  
    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

```
1 mkdir -p ~/projets/vulcain  
2 cp -r /mnt/c/Users/maeck/Desktop/vulcain/* ~/projets/vulcain/  
3 cd ~/projets/vulcain  
4 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.

```
1 sudo apt update
2 sudo apt install -y \
3     openjdk-17-jdk \
4     python3.10-venv \
5     git \
6     cmake \
7     build-essential \
8     pkg-config \
9     autoconf \
10    automake \
11    libtool \
12    zip \
13    unzip \
14    adb
```

## 3.3 Vérification des installations

### ✓ PREMIÈRE FOIS UNIQUEMENT

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

```
1 java -version
2 javac -version
3 git --version
4 adb version
5 cmake --version
6 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 :

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

### i Information

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

```
1 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 :

```
1 python -m pip install --upgrade pip
2 pip install --upgrade pip setuptools wheel
3 pip install "Cython==0.29.36" "buildozer==1.5.0"
4 pip install appdirs colorama jinja2 cython
5 pip install packaging toml
6 pip install build
7 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 :

```
1 buildozer android clean
```

Puis lancez la compilation :

```
1 buildozer android debug
2 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
3 cd .buildozer/android/platform/build-arm64-v8a-armeabi-v7a/dists/
vulcain
4 ./gradlew assembleDebug
5 cd ~/projets/vulcain
```

#### i Information

Buildozer téléchargera automatiquement :

- Android SDK
- Android NDK
- Outils Gradle
- Bibliothèques 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 :

```
1 cd ~/projets/vulcain
2
```

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

```
1 source .venv/bin/activate
2
```

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

```
1 cp -r /mnt/c/Users/maeck/Desktop/Projet/* ~/projets/vulcain/
2
```

#### 4. Nettoyer :

```
1 buildozer android clean
2
```

#### 5. Compiler :

```
1 buildozer android debug
2 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
3 cd .buildozer/android/platform/build-arm64-v8a_armeabi-v7a/dists/
  vulcain
4 ./gradlew assembleDebug
5 cd ~/projets/vulcain
6
```

#### 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/bin/
```

Le fichier s'appelle généralement :

```
vulcain-0.1-arm64-v8a-debug.apk
```

Vous pouvez le copier vers Windows avec :

```
1 cp bin/*.apk /mnt/c/Users/maeck/Desktop/
```