easyROB Deployment Workflow

## Purpose

This document describes how to build and distribute a standalone version of easyROB with its own Python environment for both Windows and Linux systems. It complements the main deployment guide with compatibility notes and clarifications.

# Windows Deployment Workflow

# 1. Developer Workflow

**Step 1**: Prepare conda environment  
- Create and configure your conda environment locally (e.g., robert).  
- Install all required packages (RDKit, Pandas, PySide6, etc.).

**Step 2**: Package environment  
- Use conda-pack to create a portable package:   
 conda activate robert  
 conda pack -o robert\_env\_unpacked.zip  
- This generates robert\_env\_unpacked.zip with the full environment.

**Important:** Once the program has been executed for the first time, conda-unpack adjusts all internal paths inside the unpacked environment. Avoid distributing that already–used folder, always distribute the freshly packed archive to ensure a clean setup on other systems.

**Step 3**: Unpack for distribution  
- Extract the zip into robert\_env\_unpacked (using 7-Zip or PowerShell).  
- The folder robert\_env\_unpacked now contains python.exe, Scripts/, Library/, etc.

**Step 4**: Prepare GUI  
- Keep your GUI project in easyROB/ (with easyrob.py as entry point).

**Step 5**: Launcher script  
- Write run\_easyrob.bat to:  
 \* Clean environment variables.  
 \* Set PATH to robert\_env\_unpacked.  
 \* Run pythonw.exe easyrob.py.  
 \* Execute conda-unpack the first time (using a sentinel .unpack.done).

## 2. Final package to ship to users

The final structure of the distributed package should look as follows:

easyROB\_win/  
├── launcher.vbs  
├── create\_shortcut.vbs  
├── tools/  
│ └── run\_easyrob.bat  
├── easyROB/  
│ ├── easyrob.py  
│ └── icon.png  
└── robert\_env\_unpacked/

This folder contains the standalone version of easyROB for Windows, including a prebuilt Python environment.

## 3. Inside the folder (User View)

The following files and folders are included in the Windows release package:

• launcher.vbs → main launcher (launch “run\_easyrob.bat” without console window).

• create\_shortcut.vbs → optional, creates a desktop shortcut with the official icon.

• tools/run\_easyrob.bat → internal script that manages logging, environment setup and GUI execution.

• easyROB/ → contains the GUI code and program icon.

• robert\_env\_unpacked/ → prebuilt Python environment used by the app.

## 4. How to use

1. Download and unzip easyROB\_win.zip.

2. Run launcher.vbs to start the program (this file internally calls tools/run\_easyrob.bat).

3. (Optional) Run create\_shortcut.vbs once to place an easyROB icon on your desktop for easier access.

4. The log file will be stored in easyrob\_process.log at the root of the folder.

5. No installation or Python setup is required — the bundled environment will be used automatically.

# Ubuntu / Linux Deployment Workflow

## 1. Developer Workflow

**Step 1**: Prepare conda environment

- Create and configure your conda environment locally (e.g., robert). Example:

conda create -n robert python=3.10  
 conda activate robert  
 conda install rdkit pandas pyside6

(Add any additional dependencies your GUI requires and robert.)

**Step 2**: Package environment

- Use conda-pack to create a portable archive:

conda pack -n robert -o robert\_env\_unpacked.tar.gz

Every time you modify the environment, you must re-pack it. Copying an old unpacked folder without re-packing may cause path issues.

**Step 3**: Unpack for distribution

- Extract the archive into a folder named robert\_env\_unpacked:

mkdir robert\_env\_unpacked  
 tar -xzf robert\_env\_unpacked.tar.gz -C robert\_env\_unpacked

The folder now contains bin/python, lib/, and other dependencies ready to use.

**Step 4**: Prepare GUI

- Keep your GUI project inside easyROB/ (with easyrob.py as the entry point).

**Step 5**: Launcher script

- Write run\_easyrob.sh to:

\* Clean environment variables.  
 \* Set PATH and LD\_LIBRARY\_PATH to robert\_env\_unpacked.  
 \* Run python easyrob.py.  
 \* Execute conda-unpack automatically the first time (using a sentinel .unpack.done).

The launcher should also show a small info popup (e.g., using zenity) and log messages to easyrob\_process.log.

## 2. Final package to ship to users

The final structure of the distributed package should look as follows:

easyROB\_linux/  
├── run\_easyrob.sh  
├── create\_desktop\_shortcut.sh  
├── robert\_env\_unpacked/  
└── easyROB/  
 ├── easyrob.py  
 └── icon.png

This folder contains the standalone version of easyROB for Linux, including a prebuilt portable Python environment.

## 3. Inside the folder (User View)

The following files and folders are included in the Linux release package:

• run\_easyrob.sh → main launcher (starts the app using the bundled Python environment).

• create\_desktop\_shortcut.sh → optional, creates a desktop shortcut with the official icon.

• robert\_env\_unpacked/ → prebuilt portable Python environment.

• easyROB/ → contains the GUI code and program icon.

## 3. How to use

1. Download and unzip easyROB\_linux.zip.

2. Open a terminal in the extracted folder and give execution permissions (first time only):

chmod +x run\_easyrob.sh create\_desktop\_shortcut.sh

3. Launch the program by double-clicking run\_easyrob.sh and selecting “Run as a program”, or by executing:

./run\_easyrob.sh

The first launch may take longer as the environment is configured. Subsequent launches are immediate.

4. (Optional) Create a desktop shortcut by running:

./create\_desktop\_shortcut.sh

5. The log file will be stored in easyrob\_process.log at the root of the folder.

6. No installation or Python setup is required — the included environment runs automatically.

# Notes

## Linux Compatibility Notes

The Linux launcher (run\_easyrob.sh) now uses xcb (X11) as the default Qt backend for maximum compatibility across desktop environments. If the system provides a Wayland compositor, the launcher can automatically switch to wayland.  
Both xcb and wayland are graphical backends used by Qt to communicate with the Linux display server — they are **not GPU drivers**, but rather **interfaces that let applications open windows and render graphics** regardless of the underlying hardware. (Essential for GUI operation.)

## Environment Re-Pack Reminder

⚠ Each time you update dependencies in the conda environment, you must recreate the portable archive using `conda-pack` (e.g., `conda pack -n robert -o robert\_env\_unpacked.tar.gz`).

Copying an already unpacked environment without re-packing may cause broken paths or missing binaries during execution.