# Snapcraft Guide (Linux)

This guide provides information on how to package your Electron application for any Snapcraft environment, including the Ubuntu Software Center.

## Background and Requirements

Together with the broader Linux community, Canonical aims to fix many of the common software installation problems with the `snapcraft` project. Snaps are containerized software packages that include required dependencies, auto-update, and work on all major Linux distributions without system modification.

There are three ways to create a `.snap` file:

1. Using `electron-forge` or `electron-builder`, both tools that come with `snap` support out of the box. This is the easiest option.
2. Using `electron-installer-snap`, which takes `electron-packager`'s output.
3. Using an already created `.deb` package.

In some cases, you will need to have the `snapcraft` tool installed. Instructions to install `snapcraft` for your particular distribution are available here.

## Using `electron-installer-snap`

The module works like `electron-winstaller` and similar modules in that its scope is limited to building snap packages. You can install it with:

```
npm install --save-dev electron-installer-snap
```

### Step 1: Package Your Electron Application

Package the application using electron-packager (or a similar tool). Make sure to remove `node_modules` that you don't need in your final application, since any module you don't actually need will increase your application's size.

The output should look roughly like this:

```
.
└── dist
    └── app-linux-x64
        ├── LICENSE
        ├── LICENSES.chromium.html
        ├── content_shell.pak
        ├── app
        ├── icudtl.dat
        ├── libgcrypt.so.11
        ├── libnode.so
        ├── locales
        ├── resources
        ├── v8_context_snapshot.bin
        └── version
```

### Step 2: Running `electron-installer-snap`

From a terminal that has `snapcraft` in its `PATH`, run `electron-installer-snap` with the only required parameter `--src`, which is the location of your packaged Electron application created in the first step.

```
npx electron-installer-snap --src=out/myappname-linux-x64
```

If you have an existing build pipeline, you can use `electron-installer-snap` programmatically. For more information, see the Snapcraft API docs.

```
const snap = require('electron-installer-snap')

snap(options)
  .then(snapPath => console.log(`Created snap at ${snapPath}!`))
```

## Using `snapcraft` with `electron-packager`

### Step 1: Create Sample Snapcraft Project

Create your project directory and add the following to `snap/snapcraft.yaml`:

```
name: electron-packager-hello-world
version: '0.1'
summary: Hello World Electron app
description: |
  Simple Hello World Electron app as an example
base: core18
confinement: strict
grade: stable

apps:
  electron-packager-hello-world:
    command: electron-quick-start/electron-quick-start --no-sandbox
    extensions: [gnome-3-34]
    plugs:
    - browser-support
    - network
    - network-bind
    environment:
      # Correct the TMPDIR path for Chromium Framework/Electron to ensure
      # libappindicator has readable resources.
      TMPDIR: $XDG_RUNTIME_DIR

parts:
  electron-quick-start:
    plugin: nil
    source: https://github.com/electron/electron-quick-start.git
    override-build: |
        npm install electron electron-packager
        npx electron-packager . --overwrite --platform=linux --output=release-build
```

```
--prune=true
        cp -rv ./electron-quick-start-linux-* $SNAPCRAFT_PART_INSTALL/electron-
quick-start
    build-snaps:
    - node/14/stable
    build-packages:
    - unzip
    stage-packages:
    - libnss3
    - libnspr4
```

If you want to apply this example to an existing project:

- Replace `source: https://github.com/electron/electron-quick-start.git` with `source: .`.
- Replace all instances of `electron-quick-start` with your project's name.

### Step 2: Build the snap

```
$ snapcraft

<output snipped>
Snapped electron-packager-hello-world_0.1_amd64.snap
```

### Step 3: Install the snap

```
sudo snap install electron-packager-hello-world_0.1_amd64.snap --dangerous
```

### Step 4: Run the snap

```
electron-packager-hello-world
```

## Using an Existing Debian Package

Snapcraft is capable of taking an existing `.deb` file and turning it into a `.snap` file. The creation of a snap is configured using a `snapcraft.yaml` file that describes the sources, dependencies, description, and other core building blocks.

### Step 1: Create a Debian Package

If you do not already have a `.deb` package, using `electron-installer-snap` might be an easier path to create snap packages. However, multiple solutions for creating Debian packages exist, including `electron-forge`, `electron-builder` or `electron-installer-debian`.

### Step 2: Create a snapcraft.yaml

For more information on the available configuration options, see the documentation on the snapcraft syntax. Let's look at an example:

```
name: myApp
version: '2.0.0'
summary: A little description for the app.
description: |
 You know what? This app is amazing! It does all the things
 for you. Some say it keeps you young, maybe even happy.

grade: stable
confinement: classic

parts:
  slack:
    plugin: dump
    source: my-deb.deb
    source-type: deb
    after:
      - desktop-gtk3
    stage-packages:
      - libasound2
      - libnotify4
      - libnspr4
      - libnss3
      - libpcre3
      - libpulse0
      - libxss1
      - libxtst6
  electron-launch:
    plugin: dump
    source: files/
    prepare: |
      chmod +x bin/electron-launch

apps:
  myApp:
    command: bin/electron-launch $SNAP/usr/lib/myApp/myApp
    desktop: usr/share/applications/myApp.desktop
    # Correct the TMPDIR path for Chromium Framework/Electron to ensure
    # libappindicator has readable resources.
    environment:
      TMPDIR: $XDG_RUNTIME_DIR
```

As you can see, the `snapcraft.yaml` instructs the system to launch a file called `electron-launch`. In this example, it passes information on to the app's binary:

```
#!/bin/sh

exec "$@" --executed-from="$(pwd)" --pid=$$ > /dev/null 2>&1 &
```

Alternatively, if you're building your `snap` with `strict` confinement, you can use the `desktop-launch` command:

```
apps:
  myApp:
    # Correct the TMPDIR path for Chromium Framework/Electron to ensure
    # libappindicator has readable resources.
    command: env TMPDIR=$XDG_RUNTIME_DIR PATH=/usr/local/bin:${PATH}
${SNAP}/bin/desktop-launch $SNAP/myApp/desktop
    desktop: usr/share/applications/desktop.desktop
```