Snap Quickstart Workshop

UbuCon Asia 2025

Past iterations

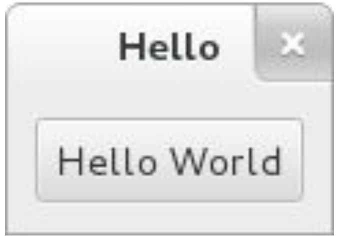
Setup

1. **Install snapd** (not needed on Ubuntu)
   - Install the package: https://snapcraft.io/docs/installing-snapd
   - And add classic snap support:
     ```
     sudo ln -s /var/lib/snapd/snap /snap
     ```
2. **Install Snapcraft**

   -
     ```
     sudo snap install snapcraft --classic
     ```
3. **Install LXD**
   - sudo snap install lxd
   - sudo adduser \`whoami\` lxd

GTK Hello World

Example 1



Building without snap

- Create the file ~/exercises/hello-world-gtk/src/hello-world-gtk.c
- Add the source code from https://www.gtk.org/docs/getting-started/hello-world
- Build the app

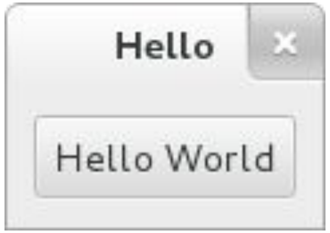
```
$ cd ~/exercises/hello-world-gtk/src
$ sudo apt install libgtk-4-
```

- Run the app

```
$ ./hello-world-gtk
```

- Remove the app

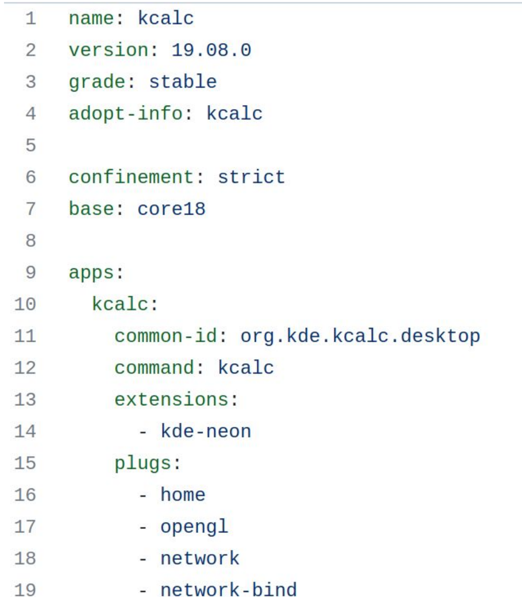
```
$ rm ./hello-world-gtk
```



Creating the snap - snapcraft.yaml

File explaining how to build the app itself and the snap

- **metadata:** Name, version, summary, description, …
- **apps:**
  - How to execute your app
  - What permissions to give your app
- **parts:**
  - How to build the binaries
  - What libraries your app needs



```
name: kcalc
version: 19.08.0
grade: stable
adopt-info: kcalc

confinement: strict
base: core18

apps:
  kcalc:
    common-id: org.kde.kcalc.desktop
    command: kcalc
    extensions:
      - kde-neon
    plugs:
      - home
      - opengl
      - network
      - network-bind
```

First: how to make a snap

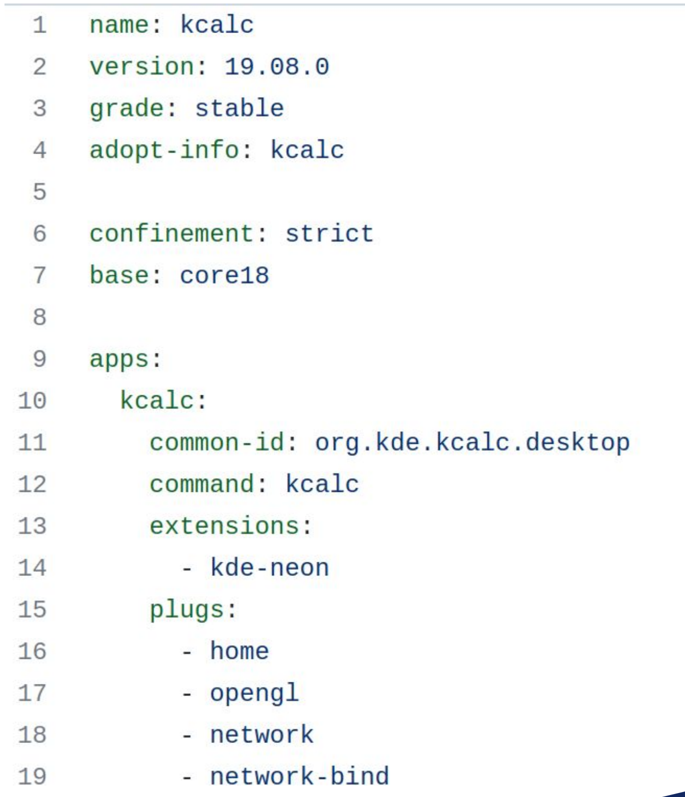
kcalc/snapcraft.yaml

$ snapcraft

kcalc.snap

$ sudo snap install kcalc.snap --dangerous

/snap/kcalc



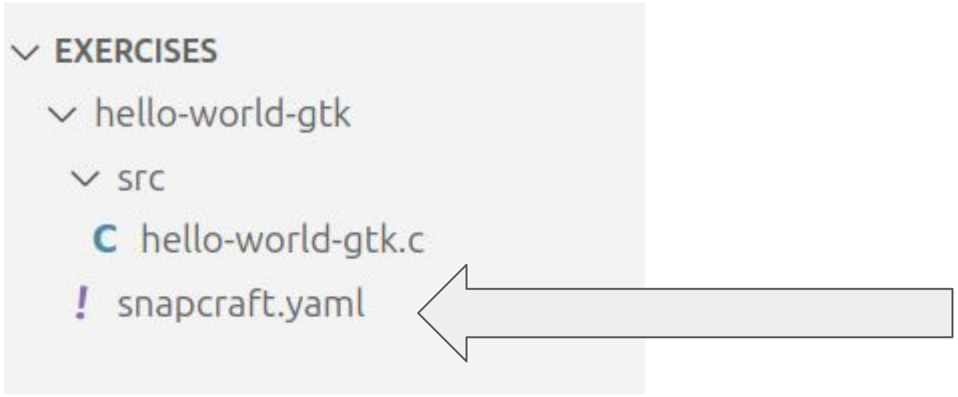
```yaml
name: kcalc
version: 19.08.0
grade: stable
adopt-info: kcalc

confinement: strict
base: core18

apps:
  kcalc:
    common-id: org.kde.kcalc.desktop
    command: kcalc
    extensions:
      - kde-neon
    plugs:
      - home
      - opengl
      - network
      - network-bind
```

gtk-hello-world - snapcraft.yaml

Complete example: https://github.com/snapcrafters/snap-quickstart-workshop

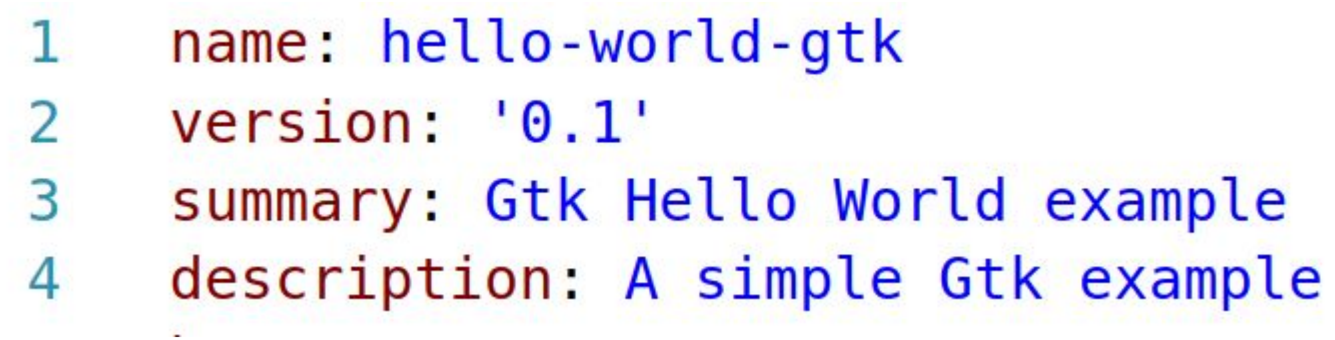


Branding metadata

More info: https://snapcraft.io/docs/snapcraft-yaml-schema

Visible in the Snap Store

- **Name** must be lowercase letters, numbers and hyphen
- **Version** is a string
- **Summary** must be one line
- **Description** can be multiline



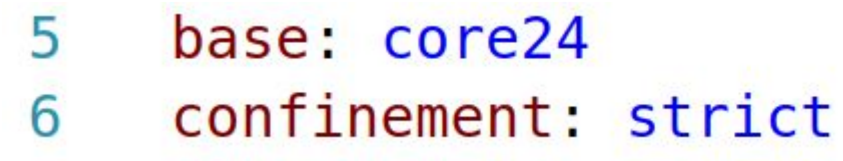
```yaml
name: hello-world-gtk
version: '0.1'
summary: Gtk Hello World example
description: A simple Gtk example
```

Technical metadata

More info: https://snapcraft.io/docs/snapcraft-yaml-schema

Describes the snap's technicalities

- **base** defines the Ubuntu version to use inside the snap
  - "core24": Ubuntu 24.04
  - "core22": Ubuntu 22.04
- **confinement** defines sandbox
  - "devmode" gives all access but logs
  - "strict" for regular sandbox
  - "classic" for no sandbox
- **compression** of the snap package
  - Always use "lzo" for best startup time!



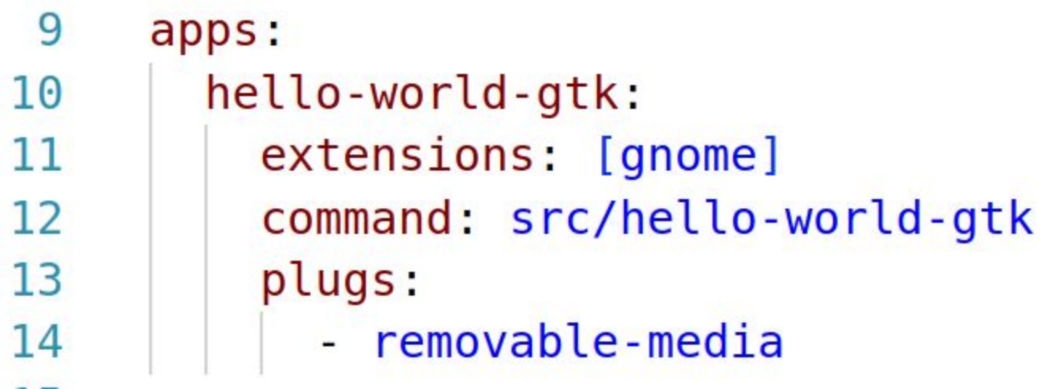
```yaml
base: core24
confinement: strict
```

Apps

More info: https://snapcraft.io/docs/snapcraft-yaml-schema#app-directives

How to execute your apps & what permissions to give

- **extensions** add common functionality
  - *gnome* adds GUI & GTK support
  - *kde* adds GUI & Qt support
- **command** is the path to the binary, relative from snap root
- **plugs** describes the permissions to give your app
  - Supported permissions: https://snapcraft.io/docs/supported-interfaces



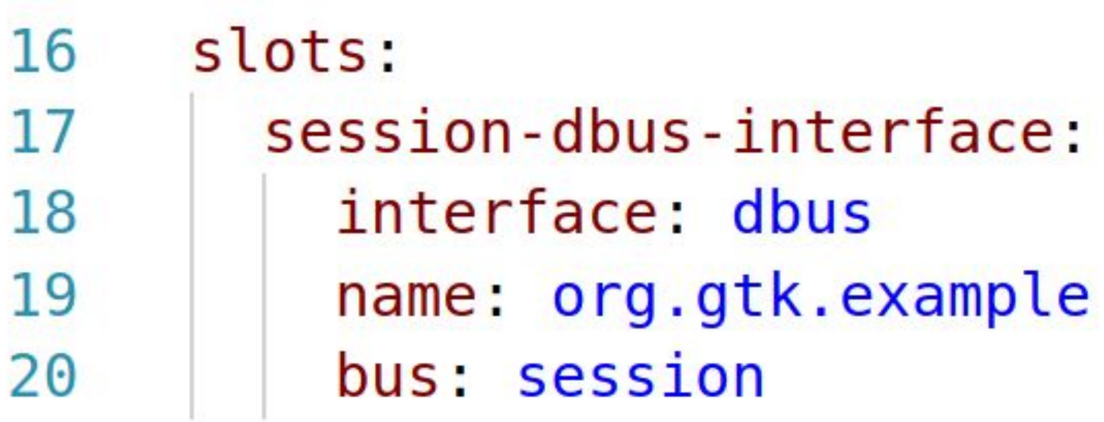
```
apps:
  hello-world-gtk:
    extensions: [gnome]
    command: src/hello-world-gtk
    plugs:
      - removable-media
```

Slots

More info: https://snapcraft.io/docs/dbus-interface

Desktop apps need access to session dbus for common functionality. This needs a declaration in slots.

- **interface**: dbus
- **name**: unique id of the app
- **bus**: session



```
slots:
  session-dbus-interface:
    interface: dbus
    name: org.gtk.example
    bus: session
```

Parts

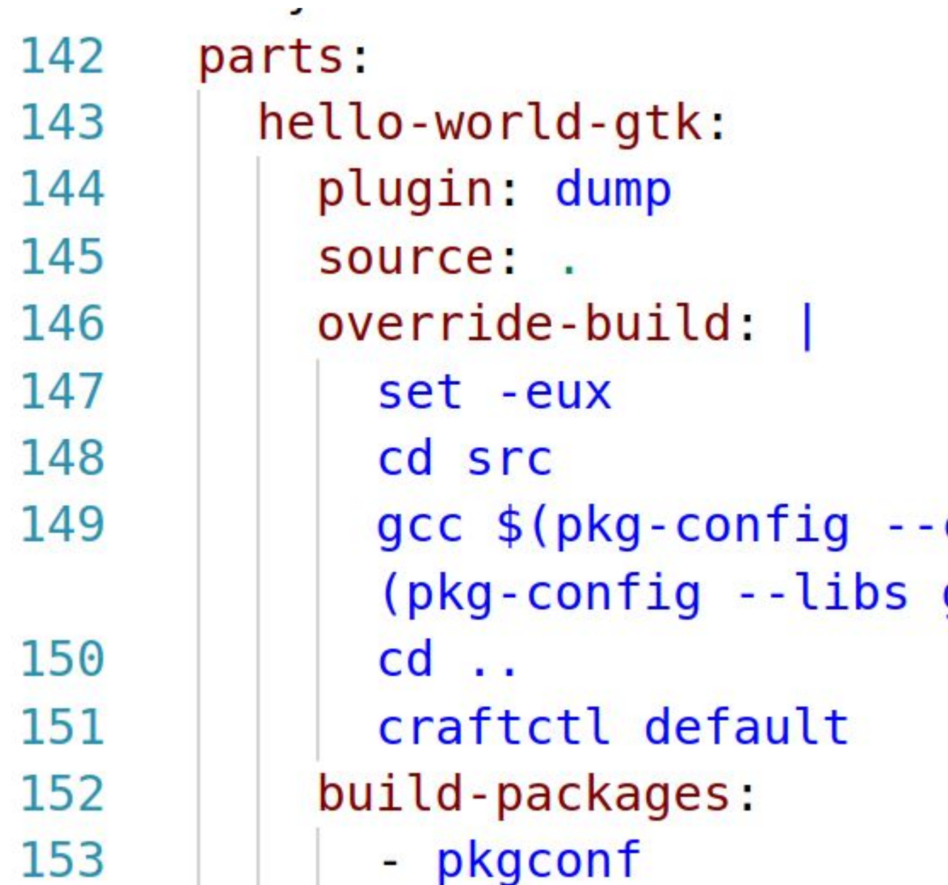
More info: https://snapcraft.io/docs/adding-parts
Deep info: https://snapcraft.io/docs/parts-lifecycle

Describes

- How to compile your app
- What dependencies it needs.

One part for each component that needs to be built separately

- C++
- Python
- Go



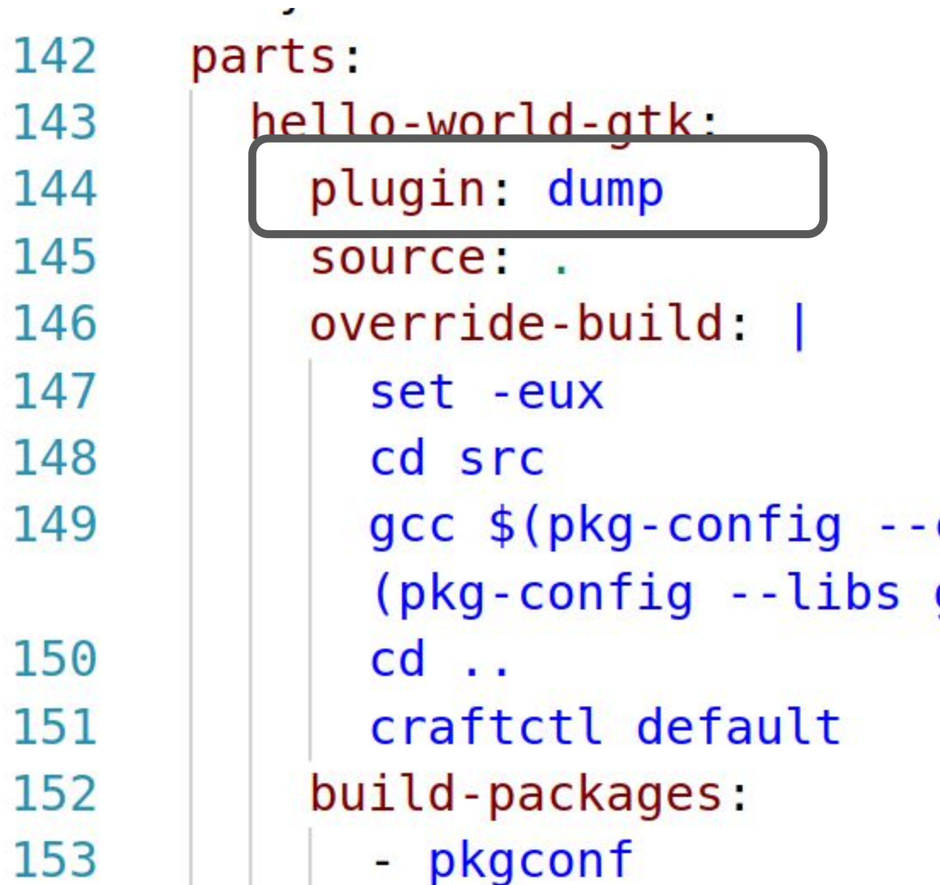
```
parts:
  hello-world-gtk:
    plugin: dump
    source: .
    override-build: |
      set -eux
      cd src
      gcc $(pkg-config --
      (pkg-config --libs
      cd ..
      craftctl default
    build-packages:
      - pkgconf
```

Plugin

More info: https://snapcraft.io/docs/supported-plugins

Which build system to use

- Binary packages or installers
  - dump: copy files (or DIY)
- source code
  - python
  - cmake
  - meson
- DIY with a scripts
  - dump or nil



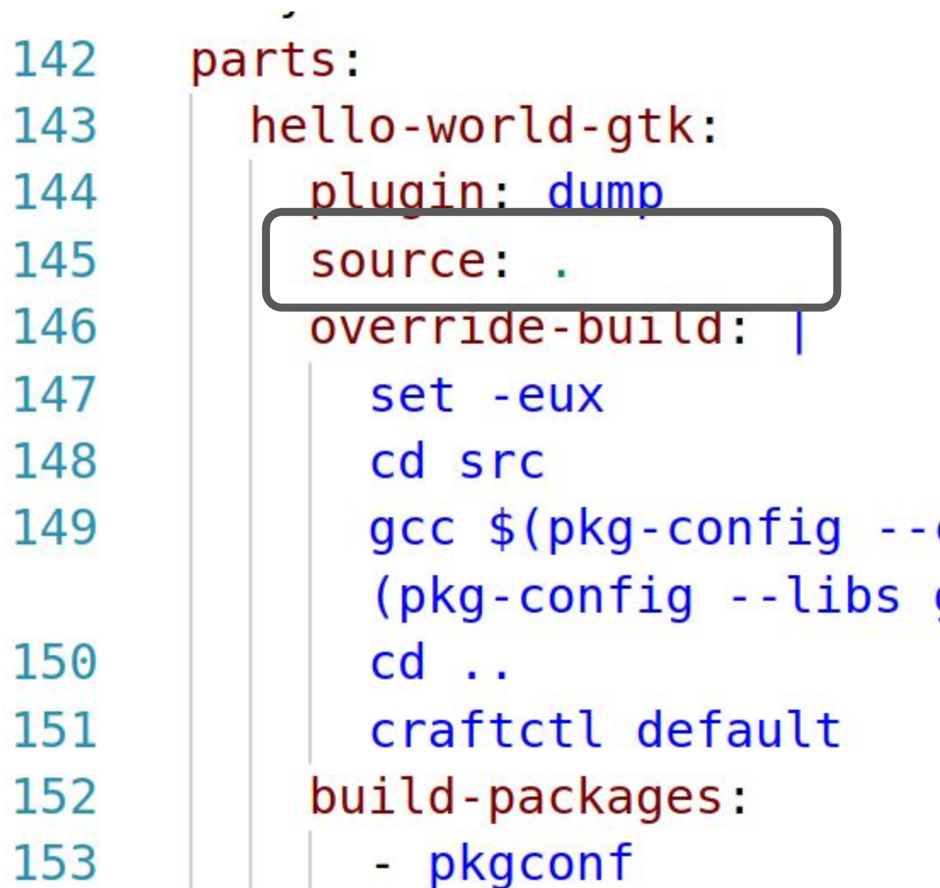
```
parts:
  hello-world-gtk:
    plugin: dump
    source: .
    override-build: |
      set -eux
      cd src
      gcc $(pkg-config --
      (pkg-config --libs
      cd ..
      craftctl default
    build-packages:
      - pkgconf
```

Source

More info: https://snapcraft.io/docs/supported-plugins

Where to get source code or binaries.

- Folder in repo with snapcraft.yaml
- External repo
- Remote file



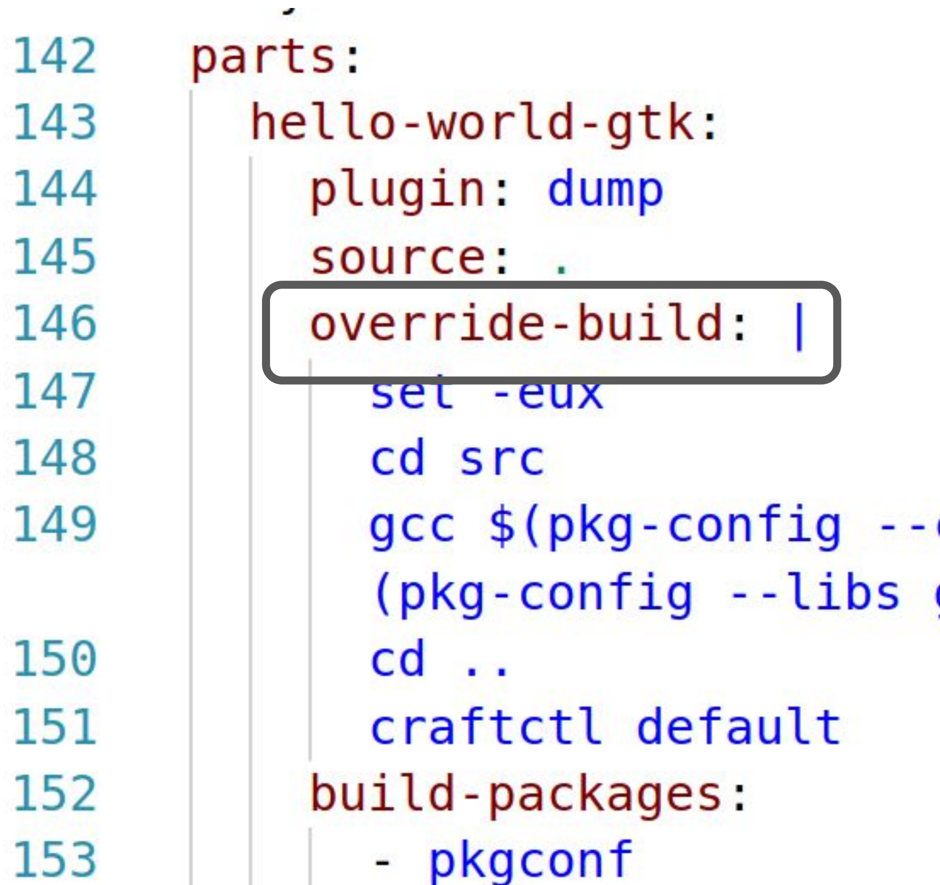
```
parts:
  hello-world-gtk:
    plugin: dump
    source: .
    override-build: |
      set -eux
      cd src
      gcc $(pkg-config --
      (pkg-config --libs
      cd ..
      craftctl default
    build-packages:
      - pkgconf
```

Override-build

More info: https://snapcraft.io/docs/overrides

Change default build logic of plugin with bash script

- call compilers directly
- prepare environment
- run default logic
  - craftctl default



```
parts:
  hello-world-gtk:
    plugin: dump
    source: .
    override-build: |
      set -eux
      cd src
      gcc $(pkg-config --
      (pkg-config --libs
      cd ..
      craftctl default
    build-packages:
      - pkgconf
```

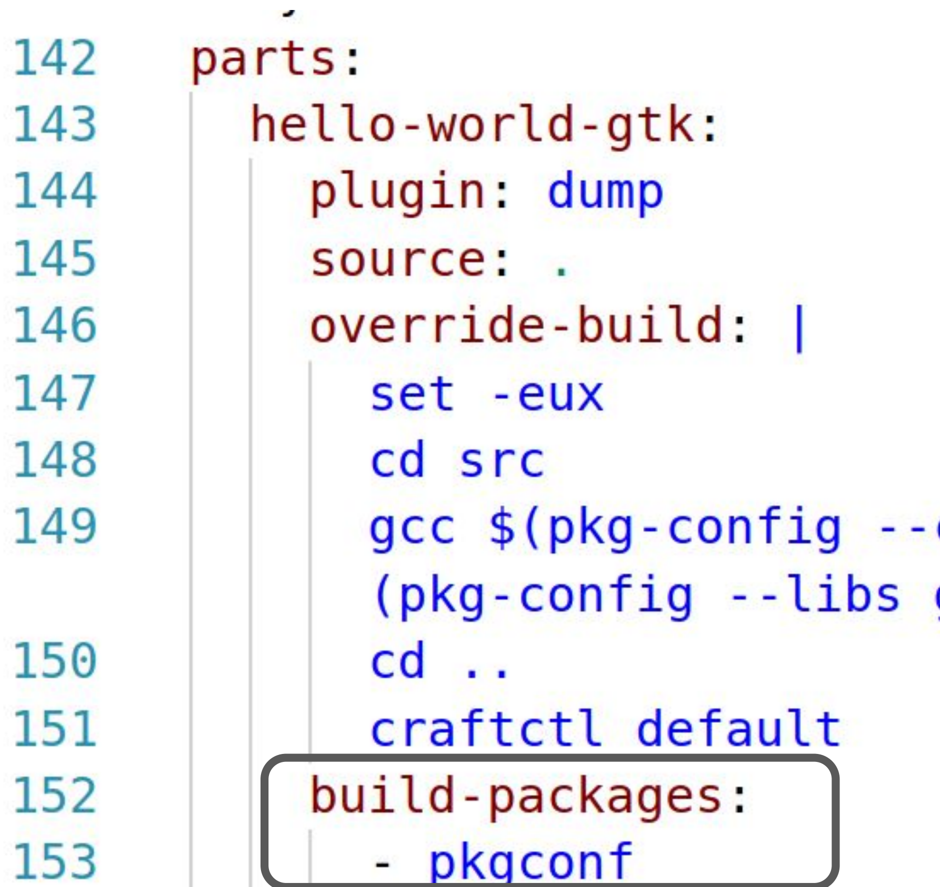
Build-packages

More info: https://snapcraft.io/docs/overrides

Tools needed to build the snap

- Compilers
- SDK libraries
- Build tools

Will not be present in the final snap



```
parts:
  hello-world-gtk:
    plugin: dump
    source: .
    override-build: |
      set -eux
      cd src
      gcc $(pkg-config --
      (pkg-config --libs
      cd ..
      craftctl default
    build-packages:
      - pkgconf
```

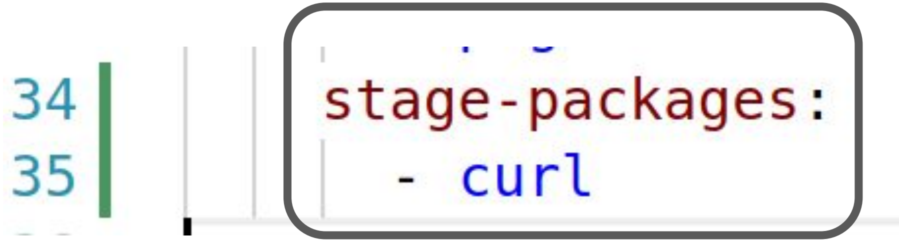
Stage-packages

More info: https://snapcraft.io/docs/overrides

Dependencies needed to run the app.

(and that are not part of an extension)

Will be in the final snap



```
stage-packages:
  - curl
```

Thanks! Questions?

Now it’s your turn!

- Either follow the tutorial: https://github.com/snapcrafters/snap-quickstart-workshop
- Or try snapping your own app and we’ll help you out!
- Or try snapping these snaps may be? Marktext or Marknote

Next steps

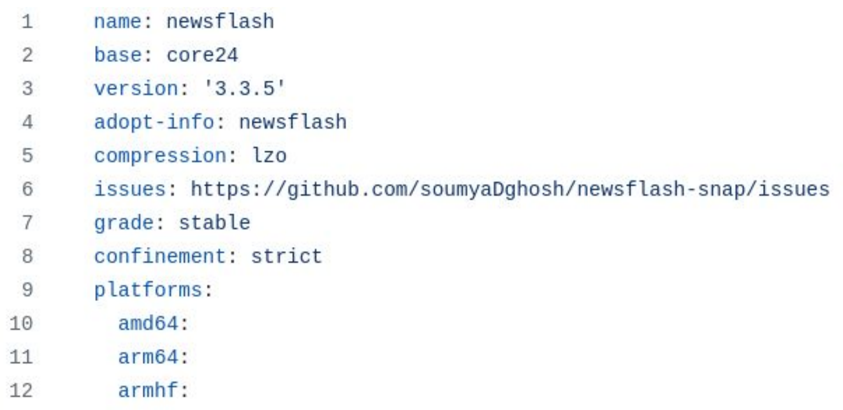
- AppStream via adopt-info
- Specify architectures -> see CI workshop
- Channels and tracks -> see CI workshop
- Building dependencies from source

AppStream as Metadata via adopt-info

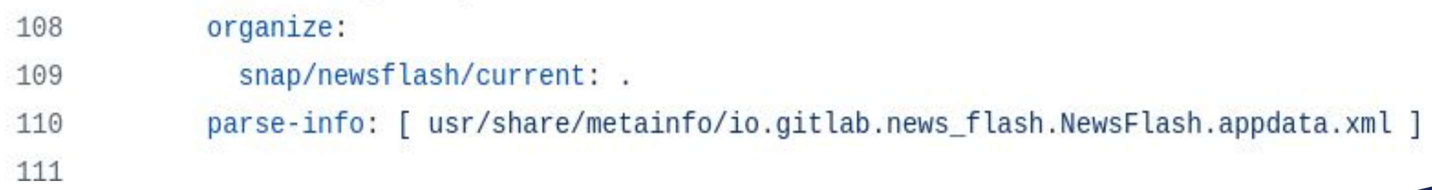
Appstream Metadata => Snap Metadata

Parse the metadata from a part

Let snapcraft know that “this” part will be used as the source of metadata



```yaml
name: newsflash
base: core24
version: '3.3.5'
adopt-info: newsflash
compression: lzo
issues: https://github.com/soumyaDghosh/newsflash-snap/issues
grade: stable
confinement: strict
platforms:
  amd64:
  arm64:
  armhf:
```



```yaml
    organize:
      snap/newsflash/current: .
    parse-info: [ usr/share/metainfo/io.gitlab.news_flash.NewsFlash.appdata.xml ]
```

Architectures…

Snaps can be built on arches that is supported by Ubuntu and the list is huge. Check here to know the list.

In the snap manifest, you should explicitly mention the architecture… But, why? To know that, join the workshop tomorrow on the CI that we use to maintain and publish our snaps by us at 2PM….

Build your dependencies from source

Build your deps in different parts!

- Benefits
  - Latest updates and releases
  - Support for custom patches
  - Everything on your control
- Disadvantages
  - Complexity
  - Miss the CVE checks done by Ubuntu for its archive packages

Things to help you in the process

- Use Gnome/KDE extensions if the libraries are related to this
- Use ffmpeg or webkitgtk shared library snaps, if your app needs them
- Try keeping the files only that your app needs
- Try to keep your snap populated with all the possible metadata you can add