Improving Snap maintenance:

Automating Snap updates on new upstream releases

An **interactive** workshop

Till Kamppeter <till.kamppeter@gmail.com>

Jesús Soto <jesus.soto@canonical.com>

Ubuntu Summit - November 2023

CANONICAL ubuntu

Introduction

The method described in this workshop was originally developed by **Heather Ellsworth**. Thanks a lot to her for this great work.

It was originally planned that she is co-hosting this workshop, but unfortunately, she has left Canonical.

Heather Ellsworth

Introduction

Please download these slides!

- You find them on the **Ubuntu Summit web site**, find this workshop on the **timetable**, open its **description page** and go to "**Presentation Materials**" at the bottom.
  https://events.canonical.com/event/31/contributions/217/
- This way you can **browse the slides in your own pace** while setting up and doing the exercises.
- And you can **copy and paste** command lines and example code.
- You can click the numerous links.
- You can also read the **advanced topics** which will not get necessarily presented here.
- **May the source (and these slides) always be with you!**

Introduction

What you will learn

- GitHub **actions** and **workflows**
- Workflow to **auto-update your** `snapcraft.yaml` if there is a **new release** in the upstream project of one of its parts
- **Auto-building/uploading** your Snap to the Snap Store
- **Development** branch and **stable** branch

What you need to know

- **Snap packaging** basics (esp. `snapcraft.yaml`)
- Basics in using the **GIT** version control system

Introduction

Why do I organize and host Snap workshops on conferences?

On this conference there are many free software **app developers** and also **potential snappers** in the community.

- **Upstream should snap**: Free software projects snap their apps by themselves.
  - They know the ins and outs and the quirks of their own apps
  - They can most easily adapt their apps for snapping
  - The one a user could trust most is the creator of the app
- If upstream does not do it we need **volunteers in the community** for app snapping projects, like **snapcrafters**
  - Well-known projects like snapcrafters have a high degree of trustworthiness, too.

Your application everywhere, just in a Snap!

Setup

Setup

This is a **workshop**, interactive, so you will try everything on **your laptop**!

You need

- Access to the **conference Wi-Fi** (SSID: **Ubuntu-Summit** Password: **Latvia2023**)
- A **web browser**
- A **text editor**
- **GIT command line utilities** (usually the "**git**" package)

Accounts (you should already have these, as experienced developer/bug reporter)

- **GitHub** - https://github.com/ To host the snapping (snapcraft.yaml, …)
- **Launchpad** - https://launchpad.net/ To setup auto-build/upload

Setup

Please **fork the sample repositories** to your GitHub account (so that you can **push** to them, **tag** them, and change their **settings**):

- https://github.com/tillkamppeter/versioning-example-app-1
- https://github.com/tillkamppeter/versioning-example-app-2
- https://github.com/tillkamppeter/versioning-example-app-3
- https://github.com/tillkamppeter/snap-automation-exercise

Clone your forks of the 4 repositories (via SSH) to your local machine for local editing:

- git clone …

Edit `snapcraft.yaml` to use your forks of the apps

Your application everywhere, just in a Snap!

What is this all about?

What is this all about?

- Imagine **you are snapping** an application, **app-1**.
- Your Snap is built from the **app-1**'s upstream source code but also needs the code of two additional repositories (libraries, command line utilities, …), **app-2**, and **app-3**.
- The **3 upstream sources** are loaded and built by **3 parts** in the `snapcraft.yaml` of **snap-automation-exercise**.
- Now imagine that the **developers** of **app-1**, **app-2**, and **app-3** are **very busy** developing and **release frequently** …
- … and **you are very busy** with a lot of things …
- … and your users complain that **your Snap is not up-to-date with upstream** …

-> We will auto-update the Snap on new upstream releases!

What is this all about?

Requirements

- **Your snapping** (`snapcraft.yaml` and auxiliary files) must be **hosted on GitHub**
  - GitHub allows **automatic tasks** on a repository via **workflows**
  - All needed **scripting** is available on GitHub
- The **upstream sources** must be **GIT** repositories (most are nowadays)
  - Each **release** has to get **tagged** with the version number

Our example has everything in GitHub repositories and therefore fulfills the requirements.

What is this all about?

Have a look into the **`snapcraft.yaml`** file of **snap-automation-exercise**

- There are 3 parts, named **app-1**, **app-2**, **app-3**. You have forked their upstream source repositories.

```
parts:
  app-1:
     source: https://github.com/tillkamppeter/versioning-example-app-1
     source-type: git
```

What is this all about?

- As **app-1** is your actual application, the **Snap's version number** is taken from **app-1**

```
name: snap-automation-example
adopt-info: app-1
[...]
parts:
  app-1:
    source: https://github.com/tillkamppeter/versioning-example-app-1
    source-type: git
    plugin: make
    override-pull: |
      craftctl default
      craftctl set version=$(git describe --tags --abbrev=10)
```

What is this all about?

- Update their **URLs** in **`snapcraft.yaml`** to use **your forks**, **commit/push** the change

```
parts:
  app-1:
    source: https://github.com/YOU/versioning-example-app-1
```

- If your laptop has **`snapcraft`** installed you can **build** and **install** this Snap. Running it simply displays the versions of your Snap and the components.

Your application everywhere, just in a Snap!

GitHub Automation

GitHub Automation

GitHub workflows

- Defines a **task to be done automatically**
  - On **each commit** or **pull request** (like CI tests, test builds, …)
  - **Time-based** (cron job)
  - **Manually** triggered in the GitHub web interface
- One `*.yml` file per workflow in `.github/workflows/` of repository
  - Trigger rules (`on:`)
  - **Tasks** to do (`jobs:`, `steps:`) as **shell scripts** (`run: |`) or as **GitHub actions** (`uses:`)
  - **GitHub actions** are tasks defined in **other GitHub repositories**
- **Logging** in web interface, under "**Actions**"
- **E-mail notification** on failure

GitHub Automation

GitHub actions

- Kind of **library functions** for **GitHub workflows**
- Defined by the `action.yml` file in a GitHub repository.
- File defines **exactly 1 action** => Usually an action is defined by **one dedicated GitHub repo**.
- When calling an action one can define a **branch** or **tag** => **More than 1 action** from 1 repo possible via **branches** or **tags**.
- Many **standard actions** under https://github.com/actions …
- … **but everyone could define their own actions**

Your application everywhere, just in a Snap!

Let's create our workflow!

Let's create our workflow!

- Create **`.github/workflows/auto-update.yml`**, just copy from **gnome-text-editor** (this file is always the same)
  - https://github.com/ubuntu/gnome-text-editor/blob/stable/.github/workflows/auto-update.yml

```
name: Push new tag update to stable branch
on:
  schedule:
    # Daily for now
    - cron: '9 7 * * *'
  workflow_dispatch:
jobs:
  update-snapcraft-yaml:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout this repo
        uses: actions/checkout@v3
      - name: Run desktop-snaps action
        uses: ubuntu/desktop-snaps@stable
        with:
          token: ${{ secrets.GITHUB_TOKEN }}
          repo: ${{ github.repository }}
```

Let's create our workflow!

Action required on each new upstream release …

… but we cannot trigger a workflow by external repositories

- Therefore we run it simply **every 24 hours**
- Plus option for **manual triggering** via GitHub web interface

```
on:
  schedule:
    # Daily for now
    - cron: '9 7 * * *'
  workflow_dispatch:
```

Let's create our workflow!

Define what to do:

- We run on **current Ubuntu**, scripting was created on Ubuntu …
- **Steps:**
  - **Download the repo** using a standard action from GitHub
  - Check upstream versions and **update** `snapcraft.yaml` with **our own action**
- `uses: org/repo@tag` or `uses: org/repo@branch`
  - Repo/branch needs `action.yml` file to describe action
- `with:` for parameter list
  - `repo:` Our repository for the script to act on
  - `token:` Authorizes the script to modify our repository

Let's create our workflow!

Our Snap updater action:

- Called by `uses: ubuntu/desktop-snaps@stable`
  => "stable" branch of https://github.com/ubuntu/desktop-snaps/
- **action.yaml** describes what has to be done
  - Defines **input and output parameters**
  - Allows running **shell scripts** or **other actions**, like a workflow
- Action calls script `updatesnap/updatesnapyaml.py` for actual task:
  - **Finds latest release tag** of source of each part in `snapcraft.yaml`
  - **Compares** with tag used in `snapcraft.yaml` and **updates**
- In case of a change, the action **commits** and **pushes** the change

Your application everywhere, just in a Snap!

Edit your snapcraft.yaml

Edit your snapcraft.yaml

Required to make a part getting auto-updated:

- **Load source** of stable releases **from GitHub repo**, not from tarball
- Select **version by release tag** of upstream repository
- Add **`source-depth: 1`** to mark for auto-update (and loads more quickly)

```
parts:
  app-1:
    source: https://github.com/tillkamppeter/versioning-example-app-1
    source-type: git
    source-tag: '0.1.4'
    source-depth: 1
```

Edit your snapcraft.yaml

Optional, to fine-tune the updating:

- **No update to a new generation**, often incompatible changes (API, …)
- **Skip** versions like **1.91**, **2.99**, … these are usually betas
- More control directives available, like **`format: "pixman-%M.%m.%R"`**

```
parts:
  app-1:
    [...]
    source-depth: 1
# ext:updatesnap
#   version-format:
#     lower-than: '2'
#     no-9x-revisions: true
```

Your application everywhere, just in a Snap!

Let's go!

Let's go!

Allow the workflow to push changes:

- In the GitHub web interface select "**Settings**" tab
- "**Actions**" -> "**General**" (Left side bar, under "**Code and Automation**")
- Set "**Actions permissions**" to "**Allow all actions and reusable workflows**"
- Set "**Workflow permissions**" to "**Read and write permissions**"

Let's go!

Test it:

- "**Release**" a new `VERSION` of one or more of app-1, app-2, app-3
  - **Edit the executable** to display `VERSION`
  - `git commit; git tag VERSION; git push; git push --tags`
  - Make sure `VERSION` is newer than the current one
- **Within 24 hours** your `snapcraft.yaml` should get **auto-updated**
- This workshop is only 1 hour, so let us **trigger manually**:
  - In the GitHub web interface of the Snap go to "**Actions**" tab
  - On the left, select "**Push new tag update to the stable branch**"
  - Click "**Run workflow**" button and choose the branch
  - You get a new entry in the list, click it and **see the logs**
  - Check the `snapcraft.yaml` in your repo

Your application everywhere, just in a Snap!

Auto-update right into the Snap Store!

Auto-update right into the Snap Store!

Two methods of building your Snap on every GitHub commit:

1. Directly in the **Snap Store**
   - Go to the **Snap Store publisher web interface**: https://snapcraft.io/snaps
   - Select your Snap
   - Click "**Builds**" tab: https://snapcraft.io/YOUR-SNAP/builds
   - Follow instructions

   Advantage:

   - No **Launchpad** interaction/account needed

   Disadvantage:

   - Only commits to **master** branch uploaded into **Edge** channel

Auto-update right into the Snap Store!

2. **Control via Launchpad**

   - Go to https://launchpad.net/
   - Select "**Register a project**" (on the right)
   - Fill in the forms (2 pages)
   - On the project page go to "**Code**" tab
   - Click "**Configure Code**"
   - Select "**GIT**" at the top
   - Under "**Link or import an existing repository**" select "**Import a Git repository hosted somewhere else**"
   - Enter your "https://github.com/…" repository URL
   - **Now each commit is automatically mirrored to the Launchpad GIT**

Auto-update right into the Snap Store!

- Under "**Branches**" you see your GitHub repo's branches
- Select the one you are auto-updating with our workflow
- Under "**Related snap packages**" click "**Create snap package**"
- Fill the field "**Snap recipe name:**"
- Choose the desired architectures under "**Processors:**"
- Mark "**Automatically build when branch changes**" to get a build on each push
- Optionally mark "**Automatically upload to store**" and choose destination channel under "**Store channels**", drop-down "**Risk**".
- Submit via "**Create Snap package**", no other settings needed.

Auto-update right into the Snap Store!

Advantages:

More configurability

One Snap build/upload recipe for each branch

Upload into desired Snap Store channel

Optionally just building, without upload

Auto-update right into the Snap Store!

At Ubuntu we do (ex.: https://github.com/ubuntu/gnome-calculator/)

- "**stable**" branch for releases
  - Default branch
  - Update automation on upstream releases
  - **Auto-upload into "candidate"** channel in Snap Store
  - Promotion into "stable" after manual test
- "**edge**" branch for development
  - **Auto-upload into "edge"** channel in Snap Store
  - Merge changes into stable branch as needed

Your application everywhere, just in a Snap!

Summary

Summary

Steps for full update automation

- Add `.github/workflows/auto-update.yml`
  - Copy from gnome-text-editor or similar
- Edit `snapcraft.yaml`
  - Source code needs to be downloaded from **GIT**
  - `source-tag:` ... to select release tag
  - `source-depth: 1` to mark as to be auto-updated
  - Additional parameters via `# ext:updatesnap`
- **Allow** the workflow to **push changes** to your GIT repository
- Set **Launchpad project GIT to auto-sync from your GitHub repo** and set this GIT to **auto-build and -upload Snap**

Your application everywhere, just in a Snap!

Get the **perfect snapper** – More info/Links

More info/links:

- Ubuntu blog by **Heather Ellsworth** and **Sergio Costas Rodríguez** about the Snap updating automation described in this workshop:
  - https://ubuntu.com/blog/improving-snap-maintenance-with-automation
- Real-live example Snaps using update automation:
  - https://github.com/ubuntu/gnome-text-editor
  - https://github.com/ubuntu/gnome-calculator

More info/links:

- **Getting started** with snapping apps, with GNOME desktop apps as example, in the workshop "**Your app everywhere, just in a Snap**" (links to slides and exercises/examples)
  - https://events.canonical.com/event/35/contributions/291/
- **More Snap magic**, not only for daemons, in the "**Daemon Snapper's workshop**" (links to slides and exercises/examples)
  - https://events.canonical.com/event/2/contributions/42/
- Workshop GNOME app Snap example from Olivier Tilloy, each commit in this GIT repository is one step of the snapcraft.yaml development:
  - https://git.launchpad.net/~osomon/+git/secrets-snap/log/?h=main
- Want to snap something cute? Qt/KDE apps? Jesús' talk from Akademy 2023:
  - https://github.com/jssotomdz/qt-snaps

More info/links:

- Ubuntu blogs from **Oliver Smith** about optimizing performance of Snaps:
  - https://ubuntu.com/blog/how-are-we-improving-firefox-snap-performance-part-1
  - https://ubuntu.com/blog/how-are-we-improving-firefox-snap-performance-part-2
  - https://ubuntu.com/blog/improving-firefox-snap-performance-part-3
  - https://ubuntu.com/blog/firefox-snap-updates-and-upgrades
- And to know why we all are snapping like hell (all-Snap **Ubuntu Core Desktop**):
  - https://ubuntu.com/blog/ubuntu-core-an-immutable-linux-desktop
- Want to watch some snappy videos? Here we go:
  - https://www.youtube.com/watch?v=TfB6QwR2GYg
  - https://www.youtube.com/watch?v=ido6kGmSHWI

More info/links:

- And at OpenPrinting we are also snappy:
  - https://snapcraft.io/publisher/openprinting
  - http://www.openprinting.org/
  - https://openprinting.github.io/about-us/
  - https://openprinting.github.io/news/
  - https://github.com/OpenPrinting