IN SPACE, NO ONE CAN HEAR YOU SNAP...

DEV TREK THE NEXT GENERATION

Igor Ljubuncic | Developer Advocate | Canonical | Stardate 99068.39
Igor Ljubuncic

- Developer Advocate
- Linux
- Books
- Petrolhead
What are we going to do?

- Talk about snaps, talk about you and me
- Snap, huh, yeah, what is it good for?
- Architecture, syntax
- Publishing to the Snap Store
- ???
- Profit!
“What be snaps?”
-- Niccolo Machiavelli
What be snaps?

- Snaps are self-contained software packages
- Work on all major Linux distributions (40+) without modification
- Simple to create and publish (with reason)
- Safe automatic updates
Traditional methods of software delivery

- Knowledge of Linux internals
- Complex packaging code
- Not transactional
- Unbounded
- Risky
“But why snaps?”

-- Zoolander
Out of the box discovery by millions

- All Ubuntu versions since 14.04
- 40+ Linux distributions
- All software welcome, regardless of license or cost
Click & Install

- One snap to rule them all ...
- Consistent behaviour
- Fewer moving parts
Analytics

- See the growth across geos & measure impact of each release

Past year: [Dropdown]
By version: [Dropdown]
Full control of update lifecycle

- Updates published instantly
- Cryptographically signed
- Risk-based channels
- OS frozen or EOL? Not a problem!
- Updates, 4 times/day

Snap a day (or rather four times a day) keeps the doctor away!
“How does it work?”

-- Friedrich Nietzsche
Snap major components

● Snapd - background service to install and run snaps
● Snap - userspace component of the snapd service, e.g:

  snap install foo

● Snap Store - central, online repository of snap applications
● Snapcraft - command line tool to build and publish snaps
The world of snaps

Snap Store

Snapd

Snap

snapcraft
Snapcraft capabilities

● Command-line tool to build and publish snaps
● Snaps as final artifact
● Snap = single compressed SquashFS filesystem
● Application code + declarative metadata
● Format extension .snap
Where does it all start?

- Build configuration in snapcraft.yaml
- YAML syntax
- Similar to RPM spec file...
Snapcraft.yaml (just an example)

name: wethr
version: "1.4.0"
summary: Command line weather tool.
description:
    Get current weather.
base: core18
apps:
    wethr:
        command: wethr
        plugs:
            - network
parts:
    wethr:
        plugin: nodejs
        source-tag: "v1.4.0"
        source: https://github.com/twobucks/wethr.git
name: wethr
version: "1.4.0"
summary: Command line weather tool.
description:
  Get current weather.
<table>
<thead>
<tr>
<th></th>
<th>Definition</th>
<th>Type</th>
<th>Length</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Name</td>
<td>String</td>
<td>1-40</td>
<td>Unique</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>App version</td>
<td>String</td>
<td>1-32</td>
<td>No semantic meaning</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>Brief</td>
<td>String</td>
<td>1-79</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Full</td>
<td>String</td>
<td>High</td>
<td></td>
</tr>
</tbody>
</table>
Confinement level

- Isolation & security
- Snaps cannot change other snaps
- Snaps cannot change system
- Different confinement levels
- Interfaces
## Confinement level

<table>
<thead>
<tr>
<th></th>
<th>Strict</th>
<th>Devmode</th>
<th>Classic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access to network</strong></td>
<td>N</td>
<td>Y</td>
<td>System</td>
</tr>
<tr>
<td><strong>Access to home dir</strong></td>
<td>N</td>
<td>Y</td>
<td>System</td>
</tr>
<tr>
<td><strong>Access to audio</strong></td>
<td>N</td>
<td>Y</td>
<td>System</td>
</tr>
<tr>
<td><strong>Access to webcam</strong></td>
<td>N</td>
<td>Y</td>
<td>System</td>
</tr>
<tr>
<td><strong>Access to display</strong></td>
<td>N</td>
<td>Y</td>
<td>System</td>
</tr>
<tr>
<td><strong>Used for</strong></td>
<td>Preferred</td>
<td>Troubleshooting</td>
<td>Stopgap measure</td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td>Interfaces override</td>
<td>Requires review</td>
<td></td>
</tr>
</tbody>
</table>

Wethr example - confinement: strict
Application isolation

Read-only file

Enforced confinement

Regularly updated OTA

Signed and authenticated

https://docs.snapcraft.io/core/snapd
Base

- In general, snaps cannot see the root FS on end user systems
- Applications need some location to act as the root FS
- Base - special kind of snap with minimal set of libraries
- Mounted as root FS for applications
- The core18/20 base is recommended

base: core18
Build definition

apps:
  wethr:
    command: wethr
    plugs:
    - network

parts:
  wethr:
    plugin: nodejs
    source-tag: "v1.4.0"
    source: https://github.com/twobucks/wethr.git
Build definition - apps

apps:
  wethr:
    command: wethr
    plugs:
      - network

• apps: defines the application(s) in the snap
• wethr: defines a block for the wethr application
• command: defines the path to executable
• plugs: access resources not available under strict confinement
Build definition - parts

parts:
  wethr:
    plugin: nodejs
    source-tag: "v1.4.0"
    source: https://github.com/twobucks/wethr.git

- parts: sources needed to assemble the app
- plugin: language specific tools
- source-tag: tag for source repositories under version control
- source: URL or path to download for the build
### Snapcraft commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>build</td>
<td>Builds artifacts based on the snapcraft.yaml.</td>
</tr>
<tr>
<td>clean</td>
<td>Remove content - cleans downloads, builds or...</td>
</tr>
<tr>
<td>init</td>
<td>Initializes a snapcraft project.</td>
</tr>
<tr>
<td>push</td>
<td>Pushes a snap to the online snap store.</td>
</tr>
<tr>
<td>register</td>
<td>Registers a snap with the online snap store.</td>
</tr>
<tr>
<td>snap</td>
<td>Create a snaps.</td>
</tr>
</tbody>
</table>

And more ...
Application build process

- Clean environment
- No library dependencies or conflicts

snapcraft
Snap created

- There could be errors - later in the series
- Successful build - <name>.snap
Snap file format

- Unpack snap or mount as loopback device:

  unsquashfs <file>.snap

  mount <file>.snap <mount point> -t squashfs -o loop

  drwxr-xr-x 2 igor igor 4096 Dec  5 12:48 bin/
  -rwxr-xr-x 1 igor igor   61 Dec  5 12:48 command-tqdm.wrapper*
  drwxr-xr-x 3 igor igor 4096 Dec  5 12:48 etc/
  drwxr-xr-x 4 igor igor 4096 Dec  5 12:48 lib/
  drwxr-xr-x 3 igor igor 4096 Dec  5 12:48 meta/
  drwxr-xr-x 3 igor igor 4096 Dec  5 12:48 snap/
  drwxr-xr-x 6 igor igor 4096 Apr 16 2018 usr/
Snap file format

- Concept similar to LD_LIBRARY_PATH
- $SNAP* environment variables
  - $SNAP (install path, RO)
  - $SNAP_DATA (path in /var, RW)
  - $SNAP_USER_DATA (path in /home, RW)
  - And others
Publishing a snap

- Create your dev account
- Register your app’s name
- Release your app
Publishing a snap

snapcraft login

snapcraft push --release=<channel> <file>.snap
Store channels

<track>/<risk>/<branch>

- Track - trade-off between stability and new features
- Risk - multiple supported releases of the same application
- Branch - optional for temporary releases and bug-fixing

--channel=latest/edge
Tracks

- Default = latest
- Minor updates, e.g. 2.0.1, 2.0.2
- Major updates, e.g. 2.1, 2.2
- Long-Term Support, e.g. 3.2, 4.1
Risk

- Most important aspect of channels
- Levels: stable, candidate, beta, and edge
  - `--stable`
  - `--channel=stable`
  - `--channel=latest/stable`
- Users can switch between channels
Upload to the store

- Do not use stable right away
- Staged deployment

```
snapcraft push --release=beta <file>.snap
```

- Automated checks + manual review
- Compelling page
“Questions?”
-- Audience (you)
Thank you!

igor.ljubuncic@canonical.com