Container craftsmanship from a Pebble to a ROCK
What is a ROCK?
What is a ROCK?

Ever used a *craft tool?
What is a ROCK?

Ever used a *craft tool?

Ever built a ROCK?
What is a ROCK?
What is a ROCK?

“A ROCK is an OCI-compliant **Ubuntu LTS-based** container image with a well-defined and opinionated design that meets the security, stability and reliability requirements from cloud-native software.”
But, why ROCKs?
36. Selection criteria for container images

Which of the following do you value the most when selecting a base image of a container image? Choose 3.

1142 out of 1166 people answered this question (with multiple choice)

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Percentage</th>
<th>Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security — passed vulnerability and malware scanning</td>
<td>56.0%</td>
<td>639</td>
</tr>
<tr>
<td>Stability — long term supported versions</td>
<td>38.8%</td>
<td>443</td>
</tr>
<tr>
<td>Size — lightweight images</td>
<td>38.3%</td>
<td>437</td>
</tr>
<tr>
<td>Compliance — with your company policies or a standard</td>
<td>37.8%</td>
<td>432</td>
</tr>
<tr>
<td>Provenance — getting the image from a trusted publisher</td>
<td>37.3%</td>
<td>426</td>
</tr>
<tr>
<td>Developer experience — frictionless usability of the image, Size — lightweight images, Stability — long term supported versions</td>
<td>27.1%</td>
<td>309</td>
</tr>
<tr>
<td>Ready-to-use — default packages and tools included</td>
<td>22.9%</td>
<td>262</td>
</tr>
<tr>
<td>Base layer — preference for an Alpine, UBI, Ubuntu, etc</td>
<td>21.7%</td>
<td>248</td>
</tr>
<tr>
<td>Price — lowest cost for pulling or running the image</td>
<td>20.1%</td>
<td>230</td>
</tr>
</tbody>
</table>
What sets ROCKs apart?

1. Opinionated and consistent design
   - Predictable entrypoint and built-in metadata

2. User-centric experience
   - It's Ubuntu, it's declarative

3. Security by design
   - Latest & greatest - natively compact and secure
The Design
Pebble?

…seamless orchestration of local service processes as an organised set

1. Consistent UX at runtime
2. Client-server model behind a single binary
3. Declarative service definition, with layering
4. Services are first-class citizens
5. Container-optimized init process
6. Embedded utilities
Wanna try a ROCK?

$ docker pull ubuntu/prometheus:2.46.0-22.04_stable
$ docker pull ubuntu/mlflow:2.1.1-1.0-22.04
$ docker pull ubuntu/alertmanager:0.25.0-22.04_stable
$ docker pull ubuntu/grafana-agent:0.35.2-22.04_stable
...

So...what is **Rockcraft** then?

“Rockcraft is a tool to create ROCKs.”
Why do you need want Rockcraft?

1. From Ubuntu, to Ubuntu
   - built by Canonical and tailored for ROCKs

2. Focus on what matters - the image
   - abstracts the repetitive and boilerplate steps

3. User-defined compartmentalization
   - select the build provider: LXD, host, etc.

4. same declarative language as all craft tools
   - Easy and familiar, for all levels

ISVs
Cloud-native developers
Occasional container users
Dockerfile is to docker build...

...as rockcraft.yaml is to rockcraft
name: my-flask-app
base: ubuntu@20.04
version: "0.1"
summary: A cool Flask ROCK
description:
  A cool ROCK for a Flask server that says hi
license: GPL-3.0
platforms:
  amd64:

services:
  rockserver:
    override: replace
    startup: enabled
    command: python3.8 /usr/bin/app
    on-failure: shutdown

parts:
  python-flask:
    plugin: python
    source: src
    stage-packages:
    - python3.8
name: my-flask-app
base: ubuntu@20.04
version: "0.1"
summary: A cool Flask ROCK
description: |
  A cool ROCK for a Flask server that says hi
license: GPL-3.0
platforms:
  amd64:

services:
  rockserver:
    override: replace
    startup: enabled
    command: python3.8 /usr/bin/app
    on-failure: shutdown

parts:
  python-flask:
    plugin: python
    source: src
    stage-packages:
      - python3.8

What packages are being installed in the ROCK?
name: my-flask-app
base: ubuntu@20.04
version: "0.1"
summary: A cool Flask ROCK
description: 
  A cool ROCK for a Flask server that says hi
license: GPL-3.0
platforms:
  amd64:

services:
  rockserver:
    override: replace
    startup: enabled
    command: python3.8 /usr/bin/app
    on-failure: shutdown

parts:
  python-flask:
    plugin: python
    source: src
    stage-packages:
      - python3.8
rockcraft.yaml sample:

```yaml
name: my-flask-app
base: ubuntu@20.04
version: "0.1"
summary: A cool Flask ROCK
description:
  A cool ROCK for a Flask server that says hi
license: GPL-3.0
platforms:
  amd64:

services:
  rockserver:
    override: replace
    startup: enabled
    command: python3.8 /usr/bin/app
    on-failure: shutdown

parts:
  python-flask:
    plugin: python
    source: src
    stage-packages:
      - python3.8
```

What is the container output on `docker run`?
name: my-flask-app
base: ubuntu@20.04
version: "0.1"
summary: A cool Flask ROCK
description: |
  A cool ROCK for a Flask server that says hi
license: GPL-3.0
platforms:
  amd64:

services:
  rockserver:
    override: replace
    startup: enabled
    command: python3.8 /usr/bin/app
    on-failure: shutdown

parts:
  python-flask:
    plugin: python
    source: src
    stage-packages:
      - python3.8
# Concepts and terminology: workshop cheat sheet

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OCI</strong></td>
<td>The Open Container Initiative.</td>
</tr>
<tr>
<td><strong>OCI image</strong></td>
<td>A container image that follow the OCI image specification (i.e. is OCI compliant).</td>
</tr>
<tr>
<td><strong>OCI archive</strong></td>
<td>A container image in the format of a compressed archive. Rockcraft's default output format.</td>
</tr>
<tr>
<td><strong>(OCI) Entrypoint</strong></td>
<td>The container image entrypoint (e.g. <code>docker inspect &lt;img&gt; -f '{{.Config.Entrypoint}}'</code>).</td>
</tr>
<tr>
<td><strong>(OCI) User</strong></td>
<td>The container user that runs the Entrypoint (e.g. `docker inspect &lt;img&gt; -f('{{.Config.User}}').</td>
</tr>
<tr>
<td><strong>base</strong></td>
<td>The Ubuntu LTS OCI image to be shipped within the ROCK. If the base is bare, then the ROCK won't have any bits from the Ubuntu LTS image within, and an &quot;build-base&quot; needs to be specified.</td>
</tr>
<tr>
<td><strong>build-base</strong></td>
<td>The Ubuntu LTS OCI image where the ROCK's parts will be built.</td>
</tr>
<tr>
<td><strong>part(s)</strong></td>
<td>Rockcraft’s parts.</td>
</tr>
<tr>
<td><strong>craft env vars</strong></td>
<td>A shorthand to refer to the Part’s reserved environment variables (e.g. CRAFT_PART_INSTALL).</td>
</tr>
<tr>
<td><strong>(part) lifecycle</strong></td>
<td>Used to refer to the parts’ lifecycle stages: pull, overlay, build, stage and prime.</td>
</tr>
<tr>
<td><strong>plugin(s)</strong></td>
<td>A part’s plugin. See all the supported plugins in Rockcraft's reference docs for “Part properties”.</td>
</tr>
<tr>
<td><strong>service(s)</strong></td>
<td>The Pebble services to be managed by the ROCK’s Pebble Entrypoint.</td>
</tr>
</tbody>
</table>
Hands-on
Install Rockcraft

How to build a ROCK?

> Inspect and run a ROCK

https://github.com/cjdcordeiro/pebble-to-rock-workshop/tree/main/C.Inspect_and_run_a_rock
> Create a chiselled rock

https://github.com/cjdcordeiro/pebble-to-rock-workshop/tree/main/D.Create_a_chiselled_rock
We want a ROCK that runs an Apache (`apache2`) web server when the container is started.

You can find the Golang application "assistant" in this folder. The objective is to create an Ubuntu ROCK called "challenge-b", that runs the `assistant` binary every time a container is deployed.

Remember the sample? The one with the Flask application...well, let's take it, and rebase it on Ubuntu 22.04 instead, whilst keeping it chiselled and baseless.

References

- Rockcraft docs
- Ubuntu containers
- Workshop: hands-on exercises
- Existing Ubuntu ROCKs (on Docker Hub and ECR)
Feedback

Click Present with Slido or install our Chrome extension to activate this poll while presenting.
Copyrights and attributions

- [CC BY-NC-SA 2.0 DEED] nocturbulous
- https://imgflip.com/i/827kg9