Introduction to OTK

The Omnifest Toolkit in the Image Builder ecosystem

@supakeen <supakeen@redhat.com>, 2024

Download

https://supakeen.com/slides/introduction-to-otk.pdf

“HTTP request smuggling is a technique for interfering with the way a web site processes sequences of HTTP requests that are received from one or more users.”

https://portswigger.net/web-security/request-smuggling

Image Builder

Image Builder

What is it?

- A team at Red Hat.
- A collection of tools to build images.

Tools

What tools does Image Builder have?

From bottom to top.

- osbuild
- osbuild-composer / images / weldr-client / cockpit-composer
- image-builder-api / image-builder-frontend

osbuild

The Assembler

- Knows nothing about distributions
- Doesn’t do any smart things, doesn’t figure out packages on its own
- Consumes machine readable JSON and produces the same image from the same input every time

osbuild-composer

The Orchestrator

- Has knowledge about distributions and how to turn them into images.
- Provides an API that allows users to start builds.
- Provides a job queue to have multiple builds in parallel.
- Has a bunch of clients that use this API
  - cockpit-composer
  - weldr-client

image-builder-frontend

The Service

- Provides a web service for users to build images.
- Integrated with the Red Hat Console.
- Speaks to the orchestrator to do the builds.

Audiences

Where do we fit?

Not in the footer

- The service is meant for systems administrators that want to build lightly customised images. We give guarantees here that you can’t break the build.
- The orchestrator is meant for systems administrators that want to run Image Builder tools internally for their own CI and/or needs. We give guarantees here that you can’t break the build.
- Osbuild is not to be used directly by users.

Who are we missing?

Distribution Maintainers

What do they want?

- They want to be have full control over how an image is laid out and the steps taken to build this.
- They want to control the definitions themselves.
- They don’t generally need high level customisations directly.
- They want integration in their build systems.

Why not?

What's wrong with the current layers?

- Contributing image definitions means creating at least 3 PRs with their CI in two different languages. More if things need to be exposed in the service.
- Having to run a lot of daemons to build images is generally complicated.

Mirror

A look inwards

- The Image Builder team maintains distribution definitions.
- We control ‘what is RHEL’, ‘what is CentOS’, and ‘what is Fedora’ through our definitions.
- These often lag behind or deviate from what others expect.

Introducing

A new layer

Really, this is the last one!

- **otk** consumes omnifests and turns them into osbuild manifests.
- **images** consumes omnifests and knows how to feed them customisations.
- Specifically **otk** allows users to write higher level YAML where they have control over everything that ends up in a manifest while not having to become machines.
- **otk** is not specific to osbuild by design however it is our first target to be supported.

Decisions

What have we decided?

- **otk** is written in Python, a lowest common denominator language that allows for the most people to be able to contribute.
- **otk** calls external programs for a lot of things. This allows us to port over parts of our Go business logic and Python business logic in a sensible way. It also allows users to quickly extend otk for their use case.
- **otk** uses conventions that are not enforced by code. Conforming to the, for example, customisation convention means that your image can be customised by Image Builder tools.

Status

Where are we at?

- We have probably figured out large parts of the infrastructure.
- You can run otk right now and compile omnifests to manifests.

Status

What is current?

- We are still having an existential crisis about the input format.
- We want to provide a way for customisations to images; preferably in a way that understands our current blueprint format from the higher layers.
- We are integrating with the osbuild-composer layer, initially by being able to pass on our customisations. This would allow us to gradually move our current image definitions to YAML.
- We need to finish up ostree-based images in our examples directory.

Future

Future

What is next?

- We want to make contact with users (CentOS Automotive and Fedora IoT)
- We want to integrate with build systems (koji).
- We want to move some distribution definitions to upstream repositories to figure out the workflow.
- For this we will focus on user friendliness, packaging, and being available there where users need us.
- Remember, we are our own users as well. Please play with it.

Questions

Supakeen on the internet

More supakeen?

- https://supakeen.com/ for a personal homepage.
- @supakeen on Twitter and generally everywhere else.
- supakeen@redhat.com or cmdr@supakeen.com for email contact.
- These slides are at: https://supakeen.com/slides/introduction-to-otk.pdf