SPDX 3 in the Yocto Project

A case study of migrating from SPDX 2 to SPDX 3

Joshua Watt

FOSDEM 2024

February 4th, 2024

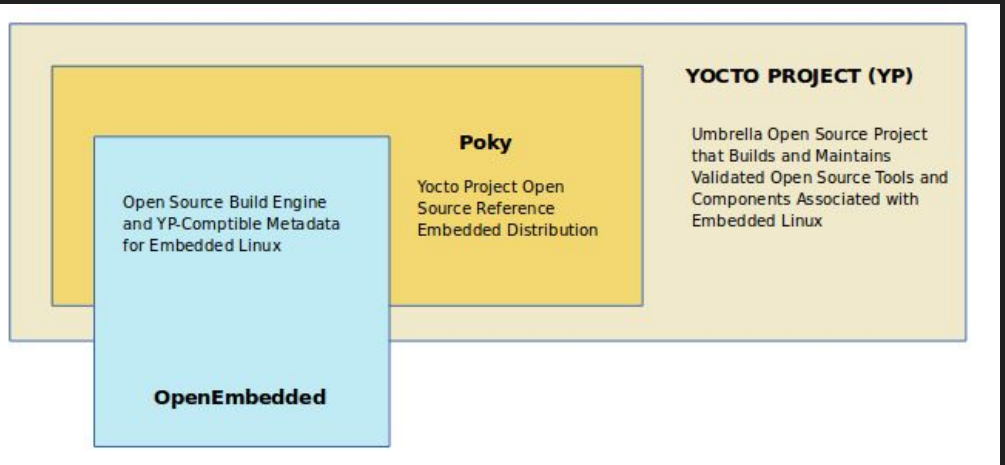
About Me

- Worked at Garmin since 2009
- Using OpenEmbedded & Yocto Project since 2016
- Member of the OpenEmbedded Technical Steering Committee (TSC)
- Joshua.Watt@garmin.com
- JPEWhacker@gmail.com
- IRC (OFTC or libera): JPEW
- X/Twitter: @JPEW_dev
- LinkedIn: joshua-watt-dev

OpenEmbedded and Yocto Project

OpenEmbedded

- Community project
- OpenEmbedded core layer
- Build system (bitbake)



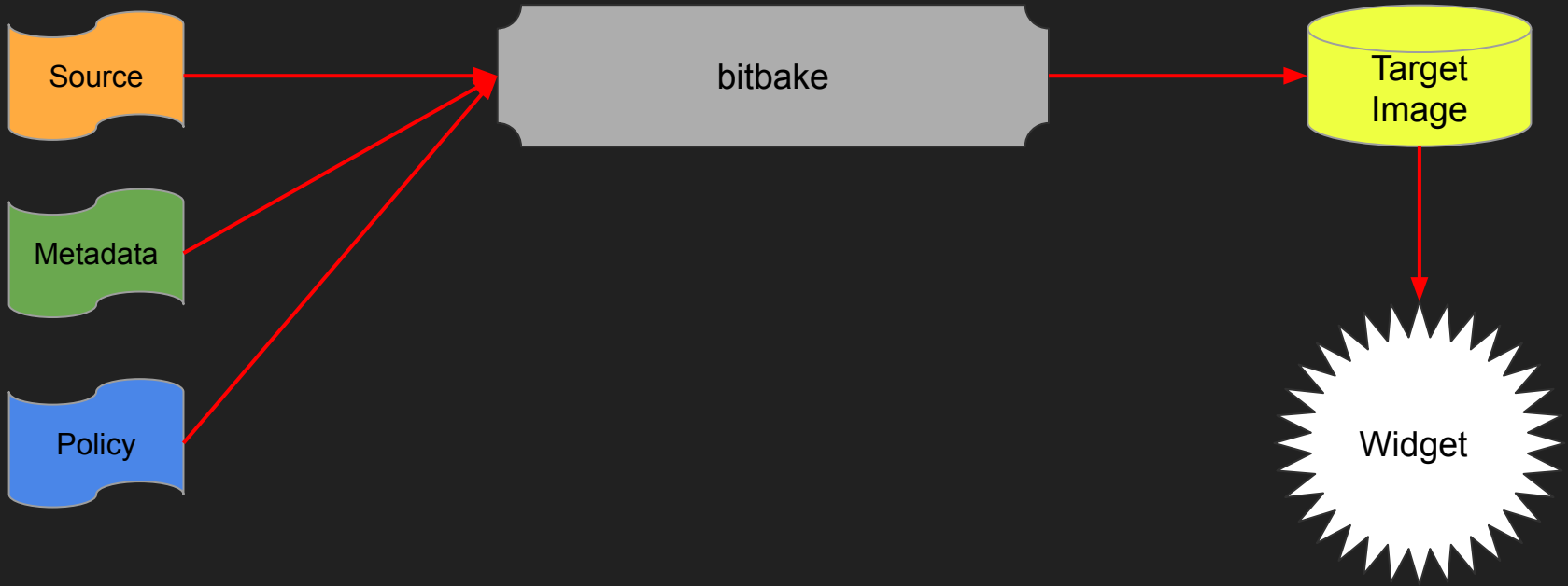
Yocto Project

- Linux Foundation project
- Poky reference distribution
- Runs QA tests
- Manages release schedule
- Provides funding for personnel
- Documentation

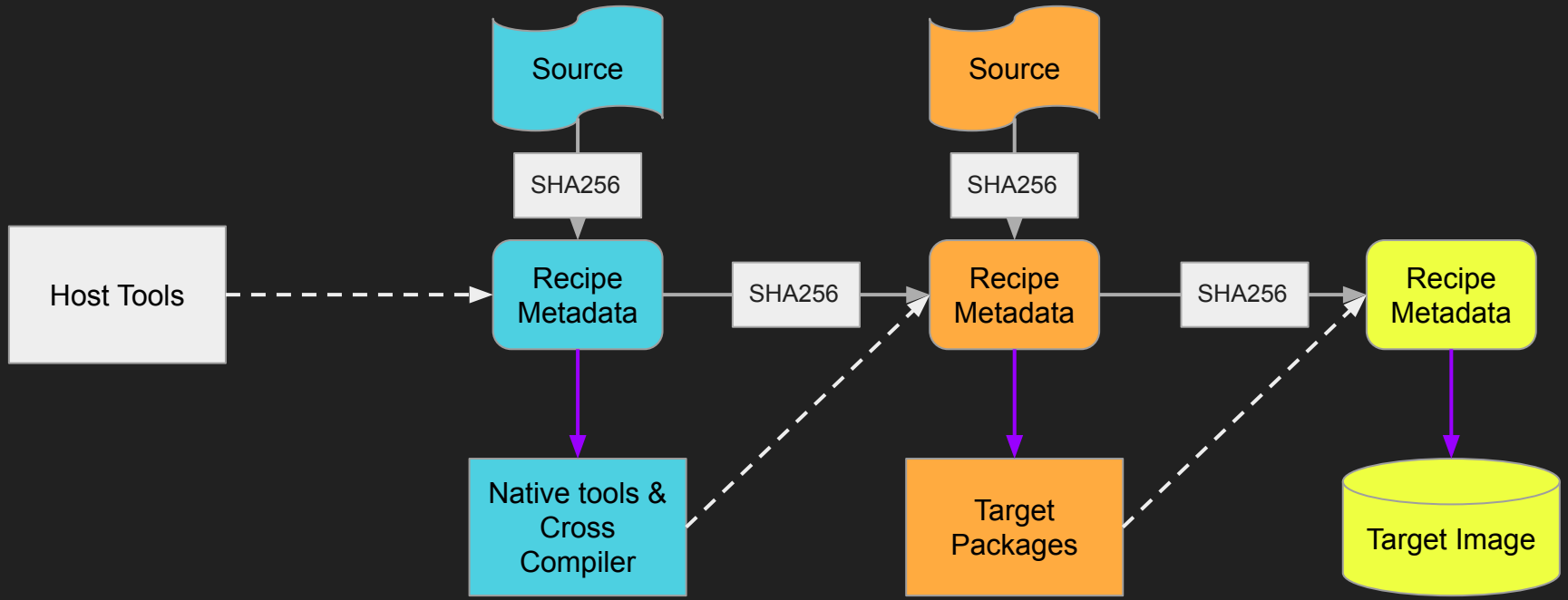
A brief overview of how Yocto works

See https://youtu.be/Q5UQUM6zxVU

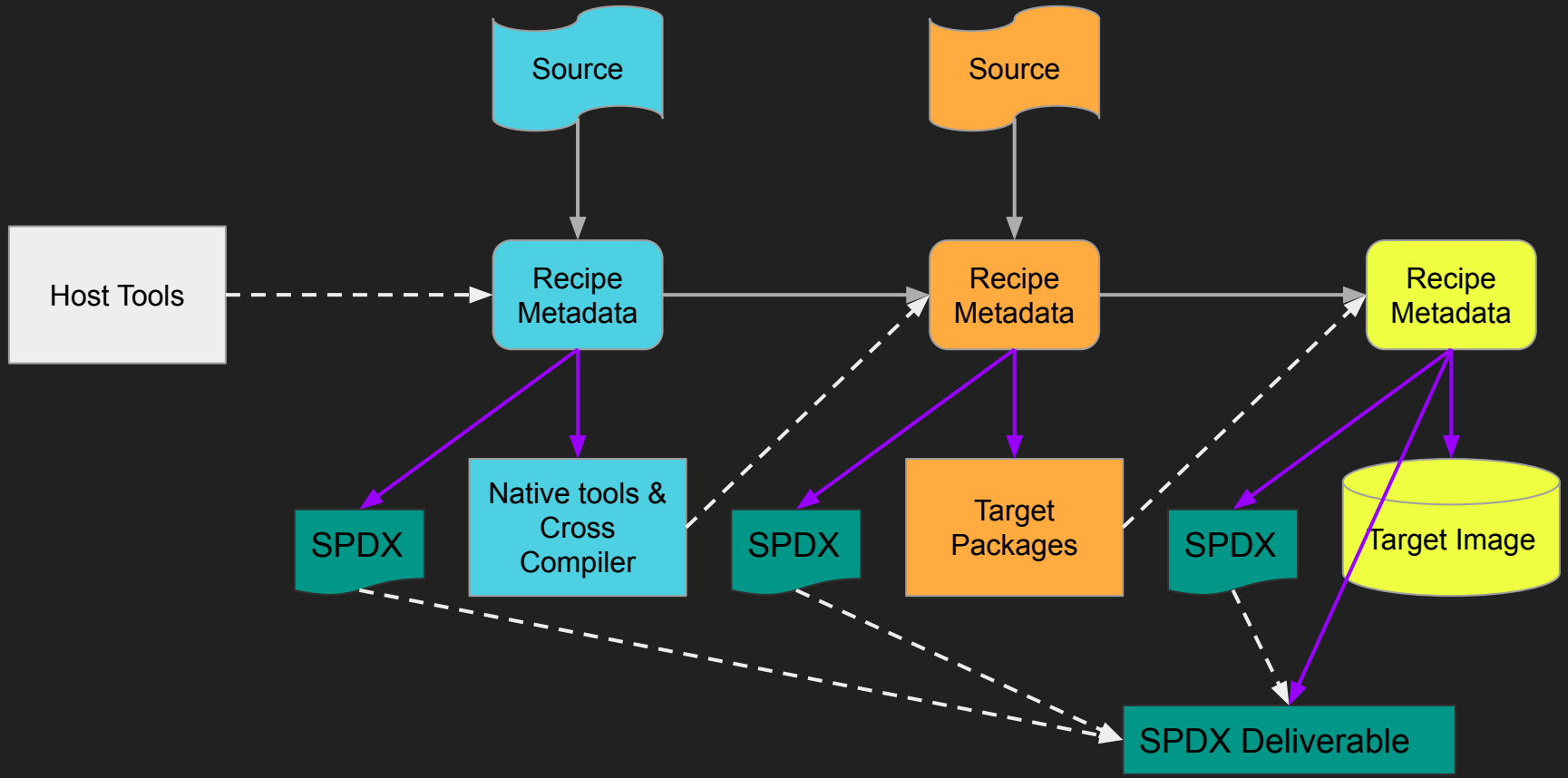
Build Images from Source Code



Simplified Build Flow



SPDX Generation



SPDX 2 Model

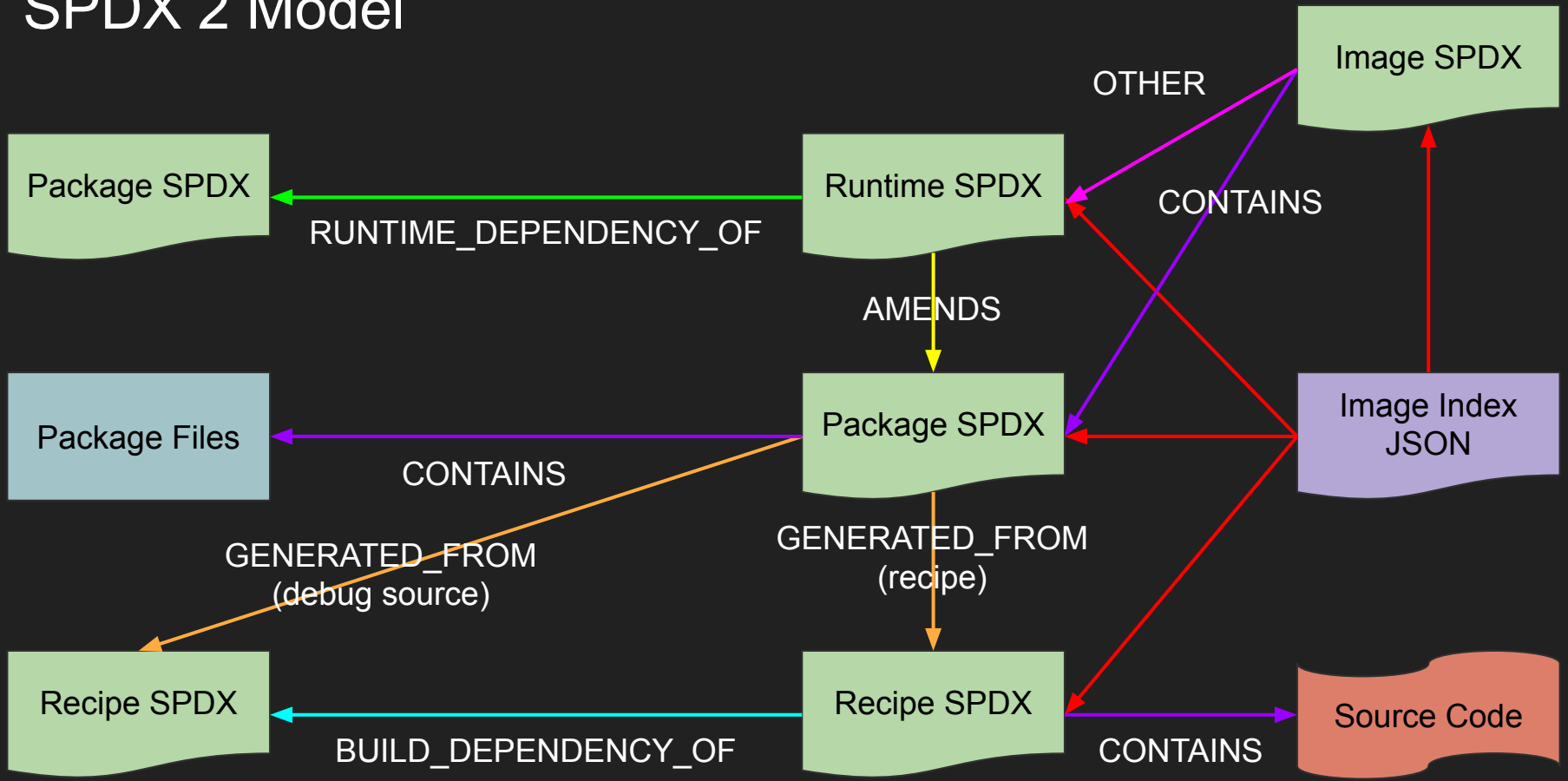
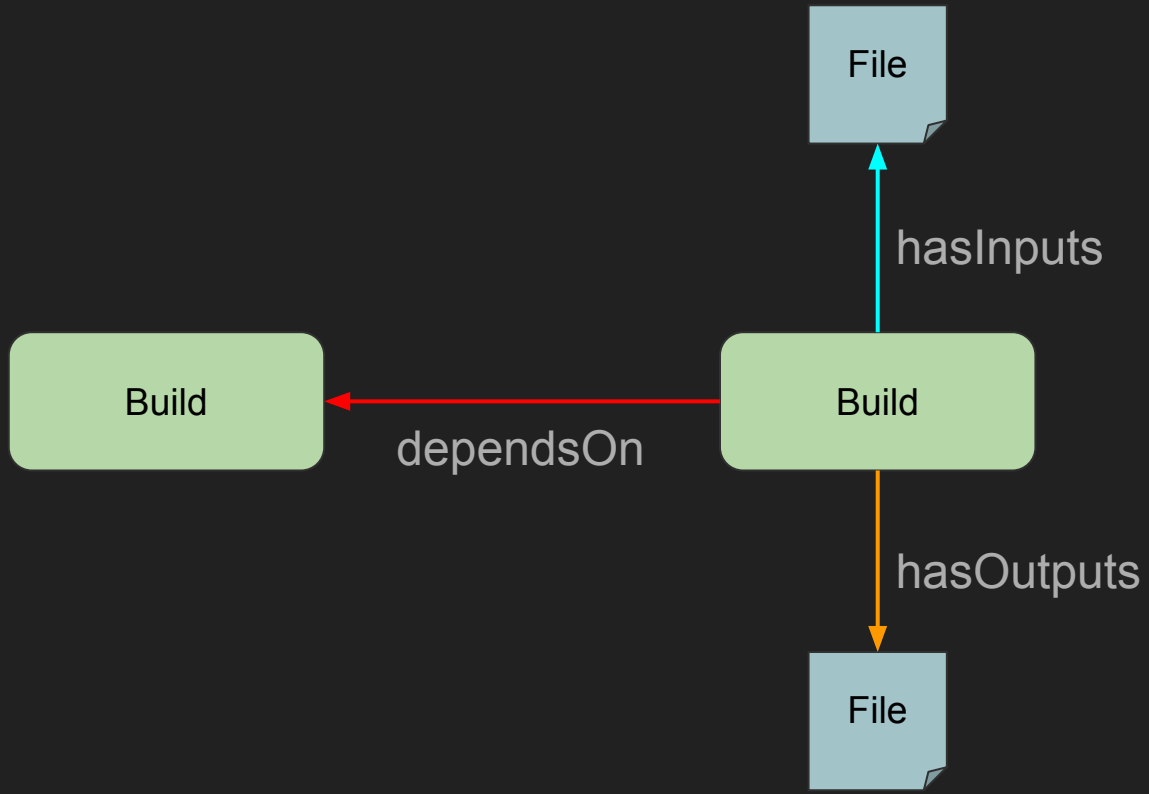


Image SPDX
OTHER
Package SPDX
Runtime SPDX
CONTAINS
RUNTIME_DEPENDENCY_OF
AMENDS
Package Files
Package SPDX
Image Index JSON
CONTAINS
GENERATED_FROM (debug source)
GENERATED_FROM (recipe)
Recipe SPDX
Recipe SPDX
Source Code
BUILD_DEPENDENCY_OF
CONTAINS

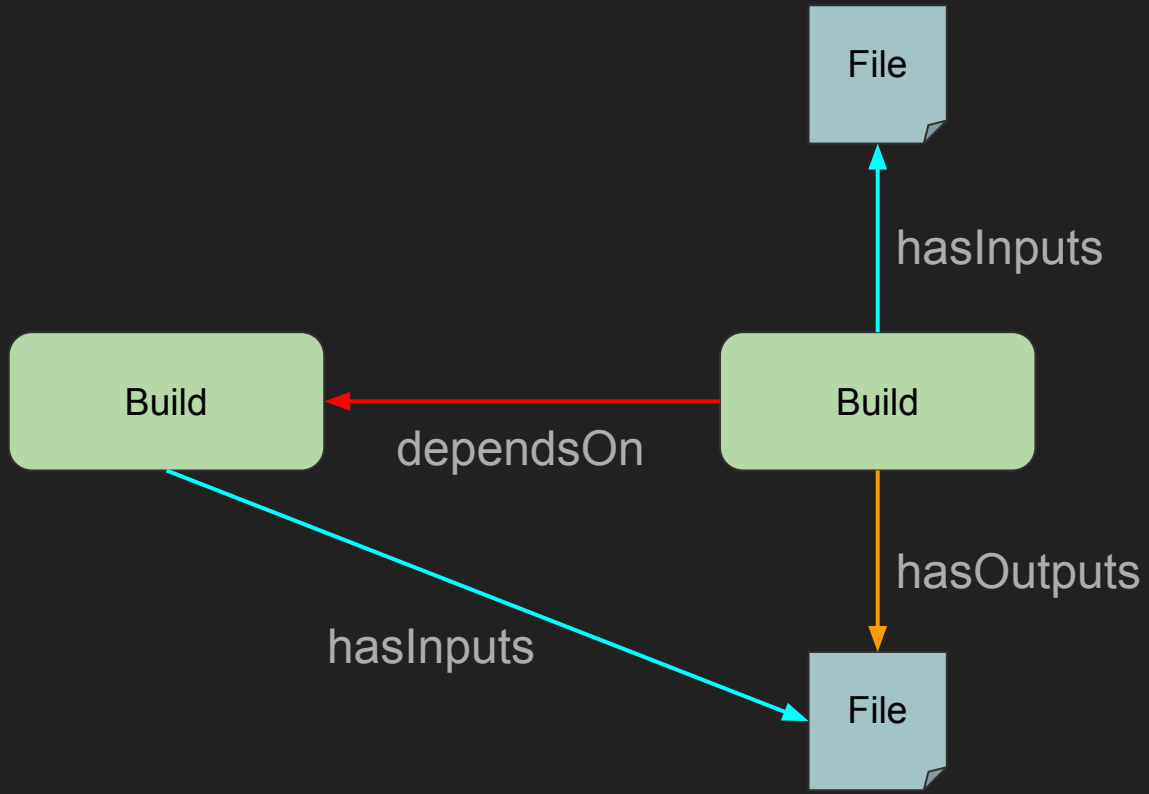
Problem #1: The "Recipe" SPDX Element is a little strange

- It's really describing how some source code was built
- SPDX 2 only has "Packages" as a thing
- SPDX 3 adds a new "Build" element that describes some build process that occurred at a point in time

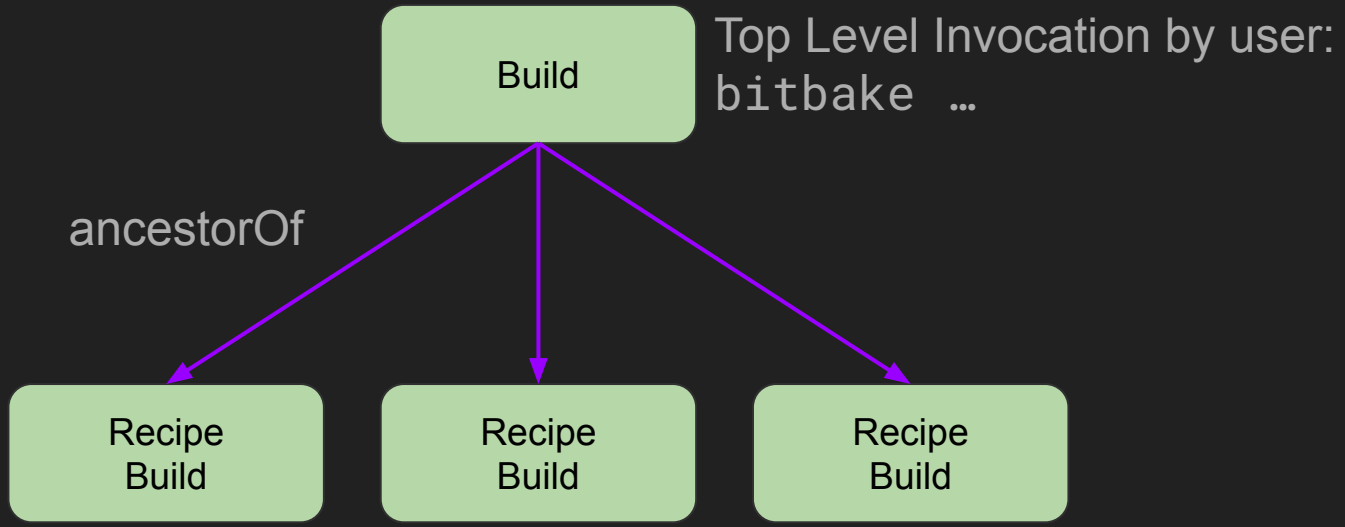
Build Elements



Build Elements



Nested Builds



Build Invocation Information

Three new relationships to describe how a build was invoked:

- **hasHost** - The host where the build was run (potentially the full SPDX document that describes it)
- **invokedBy** - The user or agent that actually invoked the build
- **delegatedTo** - Relates a user to the **invokedBy** user to indicate the build was executed on their behalf (e.g. A build service building on behalf of a user)

Problem #2: SPDXDocument & SPDXID problems

- SPDX (2) IDs are only valid in the scope of a SPDXDocument
- Documents can only be referenced by other documents with accompanying checksum
- This means once you reference an SPDX ID from another document, the containing document is "frozen" in time as any changes will invalidate its checksum
- This is fine for general usage, but really annoying for intermediate processing
- Merging SPDXDocuments is very difficult due to merging disjointed SPDX ID scopes
- "Eh…. just give up and put all the documents in a tarball" ~ Me

SPDX 3 uses Linked Data

- SPDX 3 follows the principles of Linked Data
- Objects *can* have a globally unique spdxid (IRI, URL-ish) which can be referenced by anyone (*mandatory* if the Element can be referenced)
- Since spdxids are globally unique (instead of scoped to a document), linking intermediate documents is *much* easier
- Merging documents is also much easier because there are no namespaces (other than anonymous objects)
- End result: SPDX 3 generation in Yocto produces a single merged JSON-LD SPDX document for the final Target Image

Problem #3: Validation is hard

- We have a lot of documents…. And for some reason we put them all in a tarball?
- 100's of MB of data is too much for most SPDX 2 validation tools. Our SPDX is often larger than the Target Image it describes!

So… Much… Data!

Formal SHACL model

SPDX 3 has a formal SHACL model

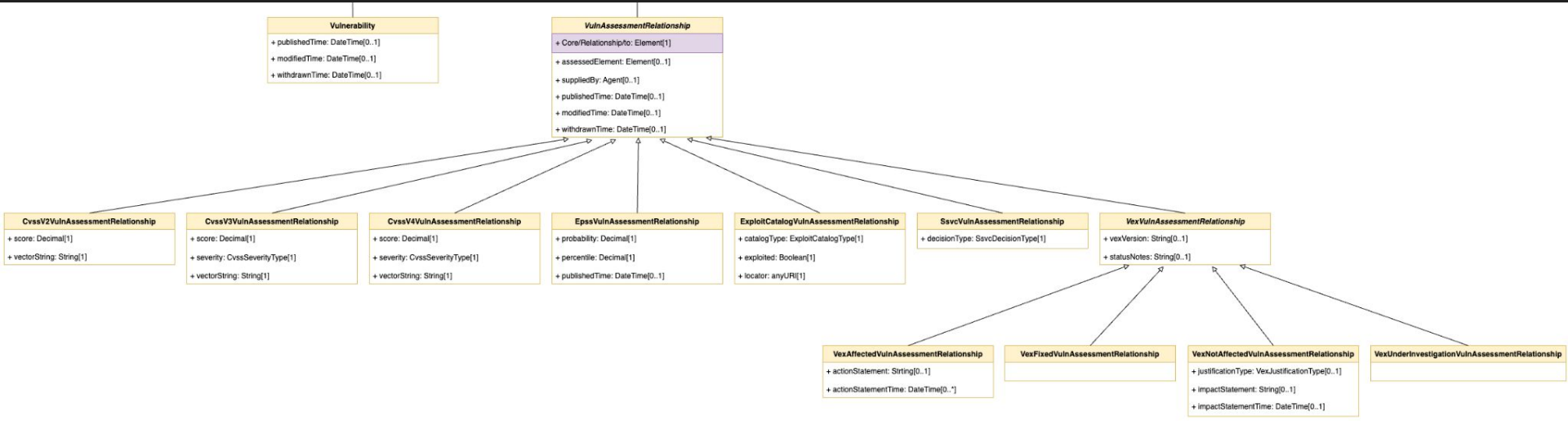
- Simplifies validation (especially for large datasets!)
- Bonus: Automatic generation of language bindings from the model (see https://github.com/JPEWdev/shacl2code)

Problem #4: CVE & Vulnerability tracking

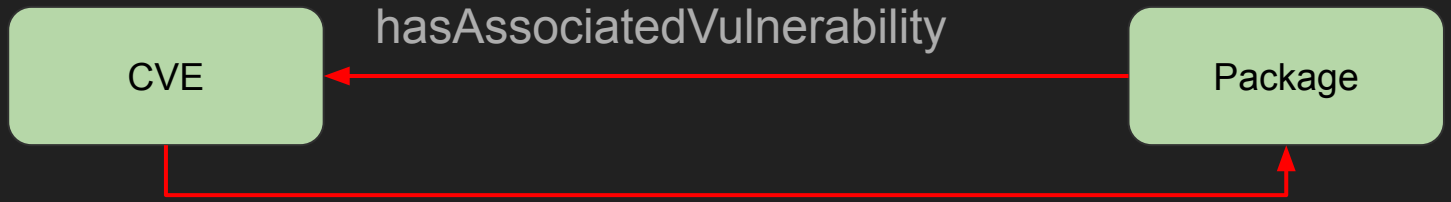
- SPDX 2.2 has no mechanism for reporting how vulnerabilities have been handled for a package
- We implemented this as supplemental information, but it's very rudimentary and not standardized

SPDX 3 Vulnerability reporting

- SPDX 3 implements a complete VEX-compliant encoding for how Vulnerabilities have been addressed
- It's complex, but powerful



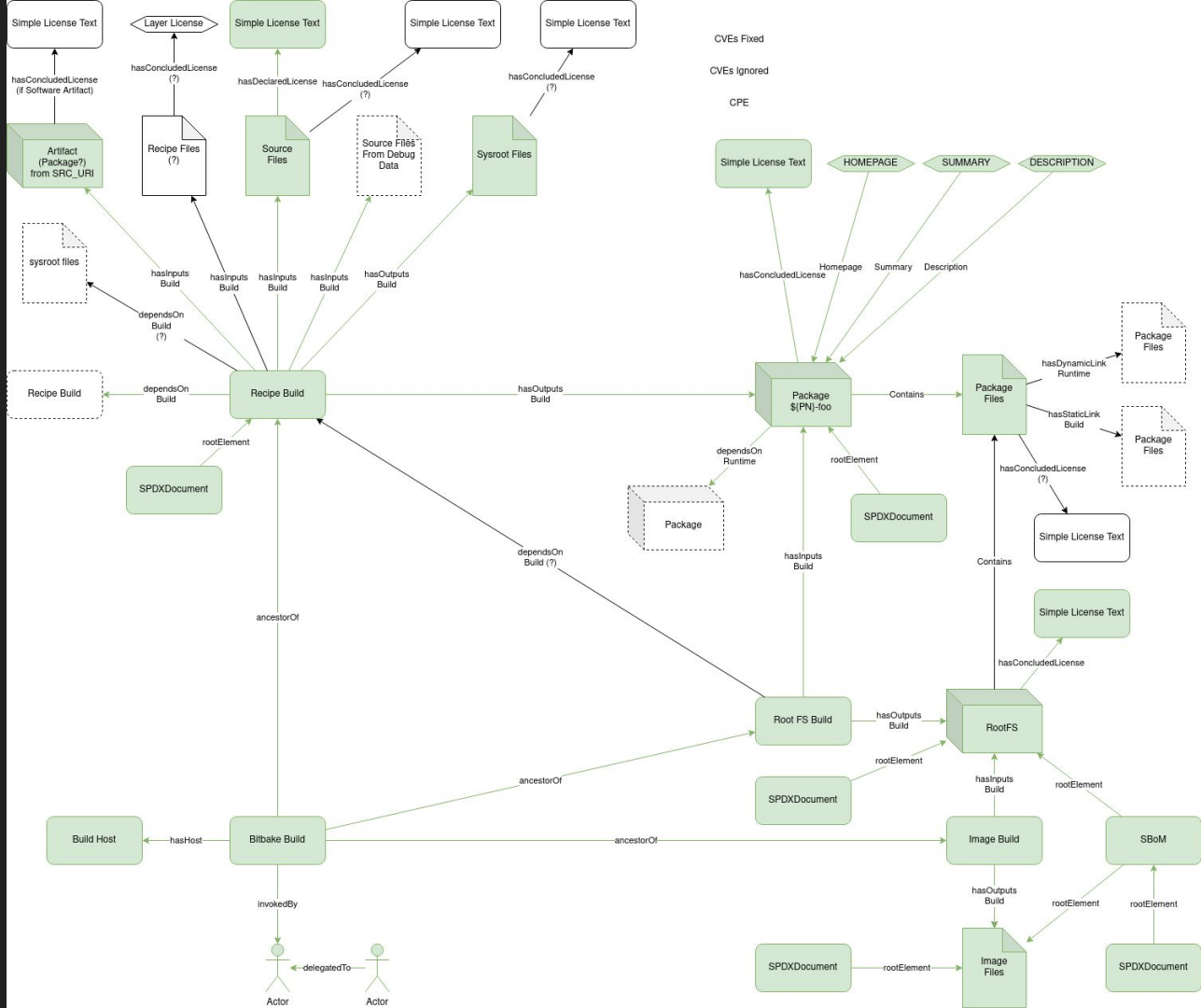
SPDX 3 VEX Example



doesNotAffect
vexVersion: 1.0.0
impactStatement: Code not compiled

Where we are today

SPDX 3 Model



https://drive.google.com/file/d/17EH2NoQrVdFyaYM6ooVCGvCbrlsInOZt/view

SPDX 3 Challenges

- Think of the "zero dependency" users (like bitbake)!
- Do we really need any encodings other than JSON-LD?
- Is `@context` *really* necessary (it's hard to parse without libraries)

Closing Thoughts

- SPDX 3 has a higher "ceiling" of what we can express
- It should also be much easier to link documents together

More information

Other talks that are specifically about SBoM generation in OpenEmbedded

- https://youtu.be/8X5PWa7A6pY
- https://youtu.be/6zms_qGmVqg
- https://youtu.be/h6PRf4zxnR4

Questions?