Software Supply Chain Management with the Yocto Project

Joshua Watt

Open Source Summit

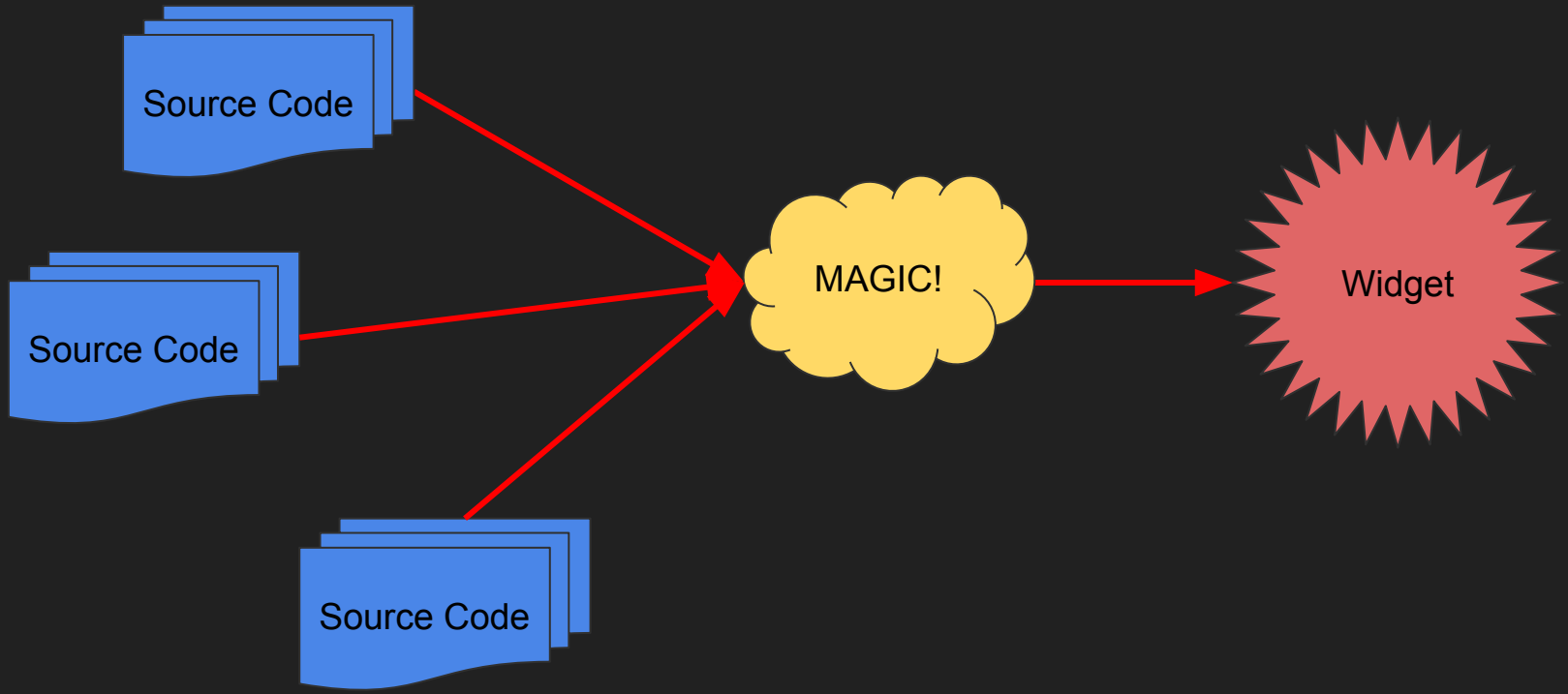
May 20, 2026

About Me

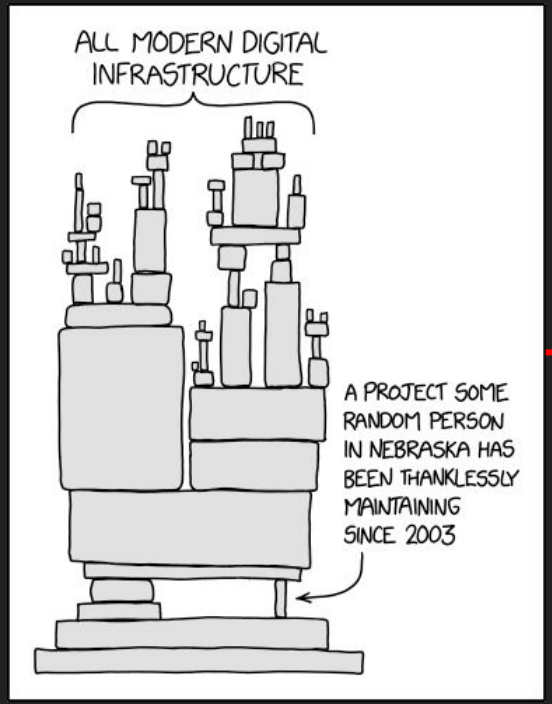
- Worked at Garmin since 2009
- Using OpenEmbedded & Yocto Project since 2016
- Member of the OpenEmbedded Technical Steering Committee (TSC)
- Member of Yocto TSC
- Member of the SPDX Technical Team
- Joshua.Watt@garmin.com
- JPEWhacker@gmail.com
- IRC (OFTC or libera): JPEW
- X/Twitter: @JPEW_dev
-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

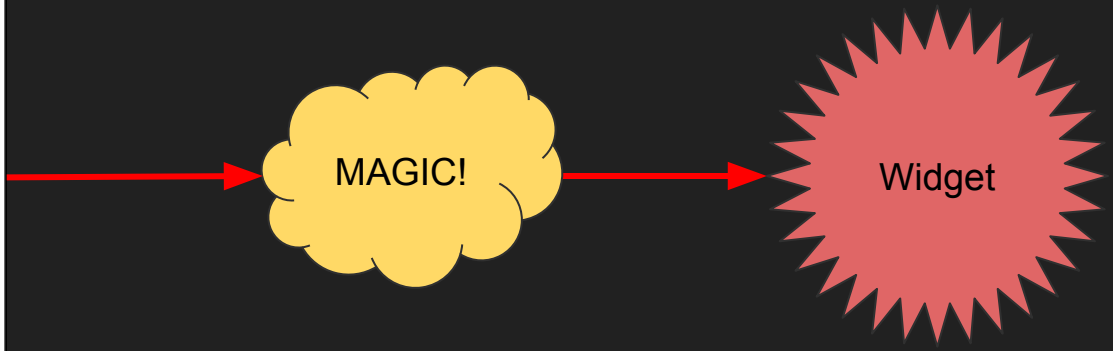
Software Supply Chain

Modern Software Supply Chains

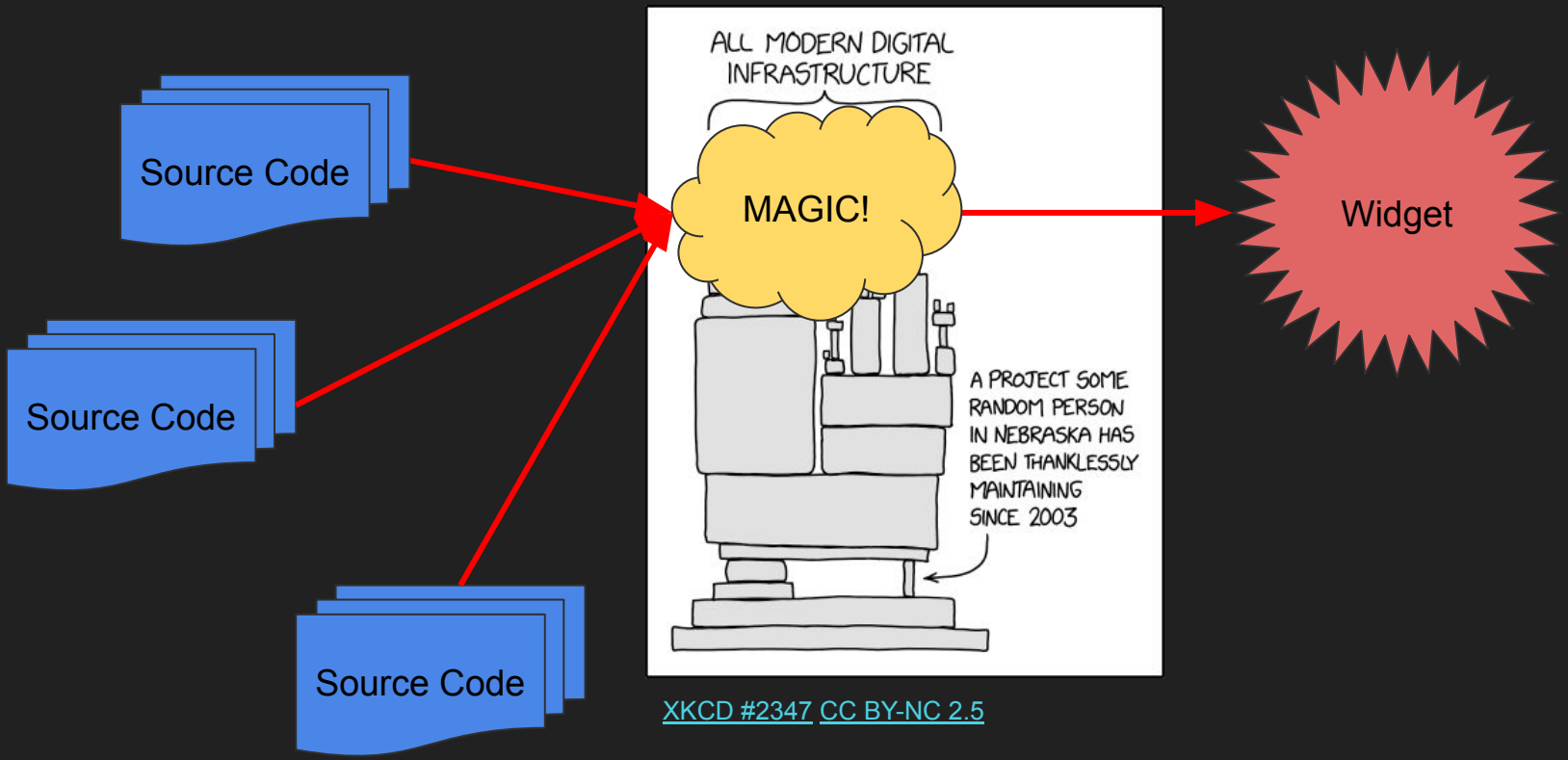


Modern Software Supply Chains





Modern Software Supply Chains



XKCD #2347 CC BY-NC 2.5

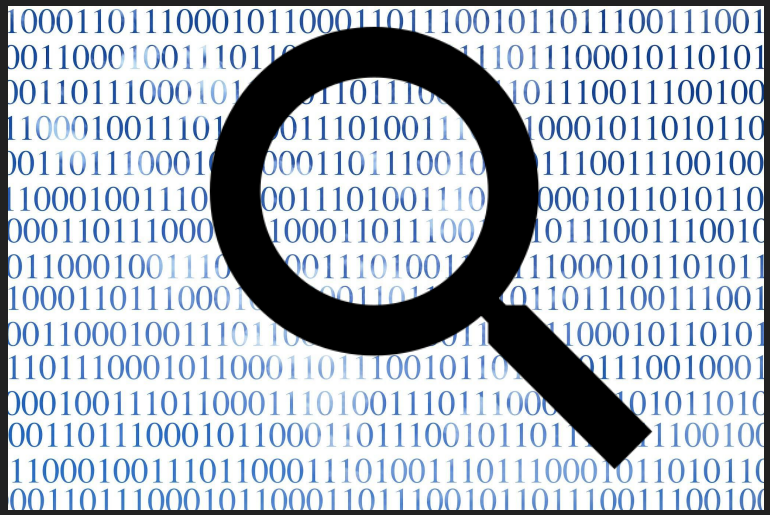
Regulatory Agencies have taken notice

- United States Executive Order 14028
- EU Cyber Resilience Act

What's in my Software?

- Software is often not easily observable in its final form
- What's in my Software?
  - Where did it come from?
  - What version is it?
- Am I complying with Software Licenses?
- Has it been tampered with?
- Is it vulnerable to exploits?
- **Can deliverables be traced back to their code?**

What's really in here?



The relationship of a Software Supply Chain and SBoMs

- Everyone has a Software Supply Chain
  - It might be good, bad, or ugly
- SBoMs are one way of providing visibility into the supply chain and reasoning about it

Standardized Format for the Exchange of Information

Regulatory Agencies

Internal Compliance

Downstream 3rd Parties

Standardized Format for the Exchange of Inform

Do you have any vulnerabilities?

Regulatory

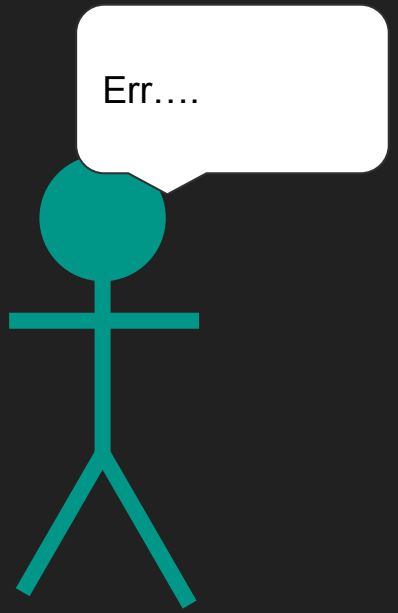
What Licenses are we using?

Internal

What's in the software you gave us?

Downstream 3rd Parties

Standardized Format for the Exchange of Inform



Do you have any vulnerabilities?

Regulatory

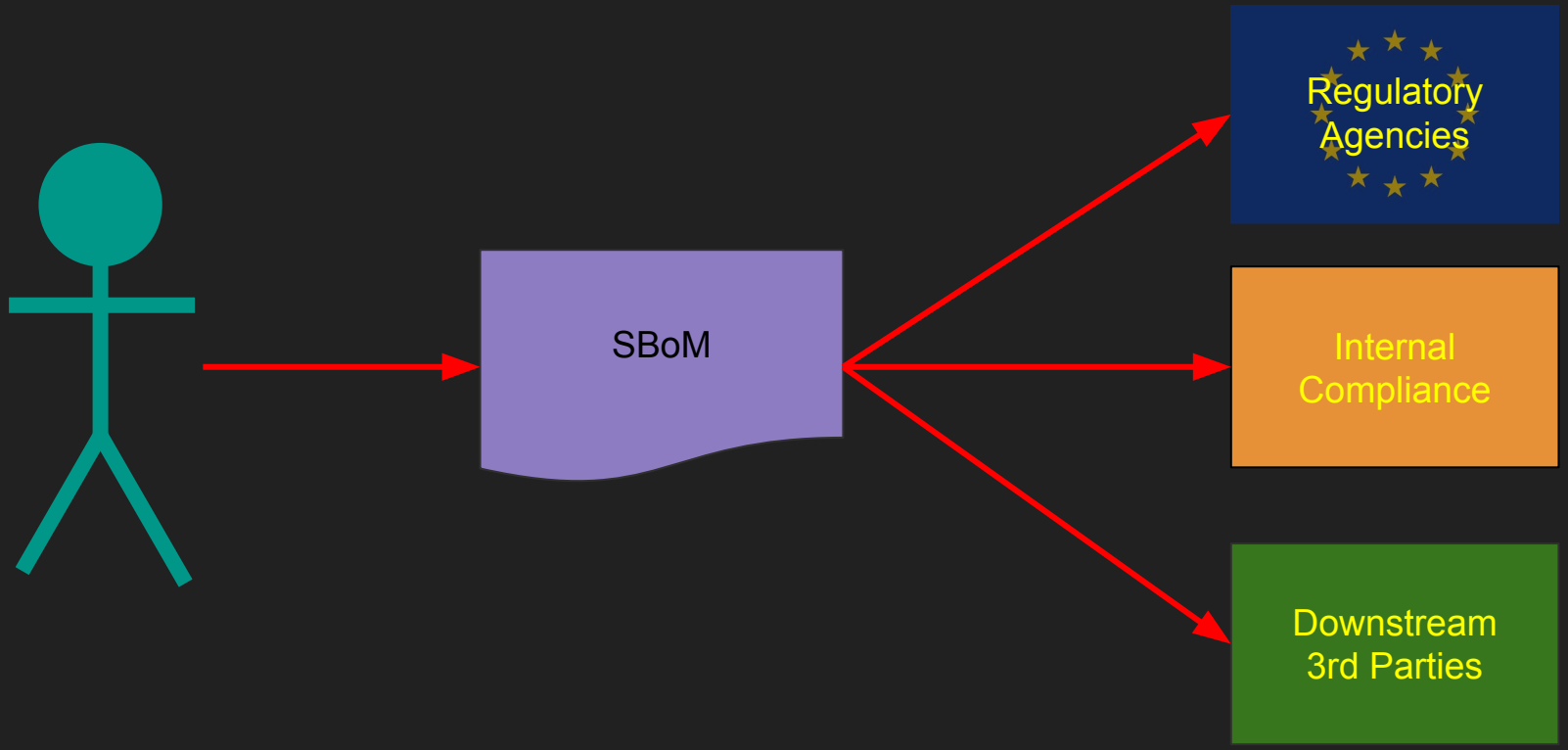
What Licenses are we using?

Internal

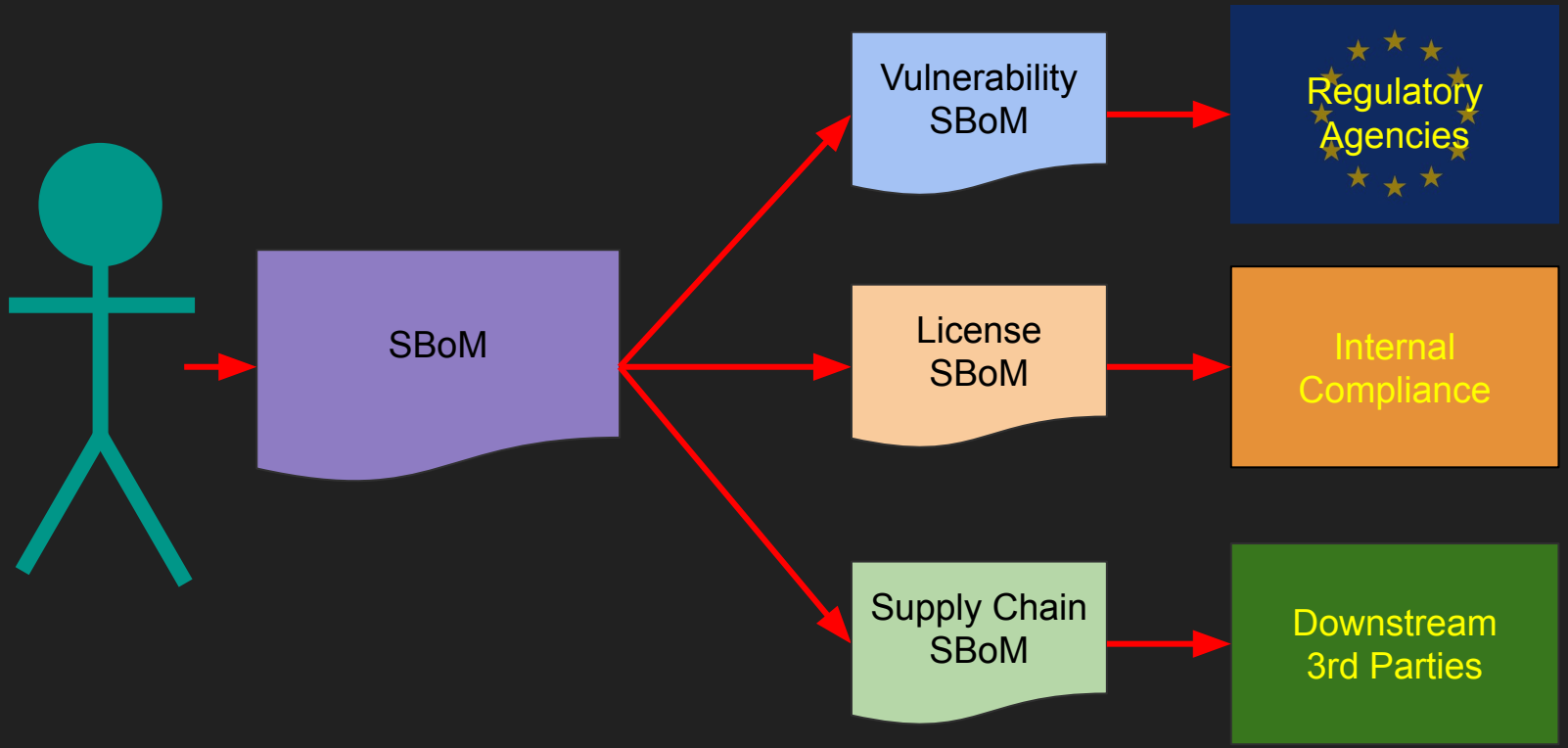
What's in the software you gave us?

Downstream 3rd Parties

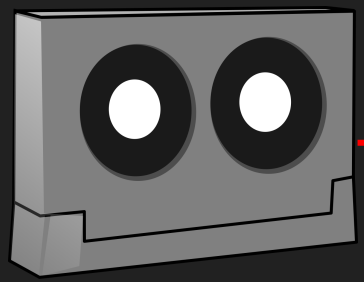
Standardized Format for the Exchange of Information



Standardized Format for the Exchange of Information



Standardized Format for the Exchange of Information



KSEltar, CC BY-SA 3.0 via Wikimedia Commons

SBoM

Vulnerability SBoM

License SBoM

Supply Chain SBoM

Regulatory Agencies

Internal Compliance

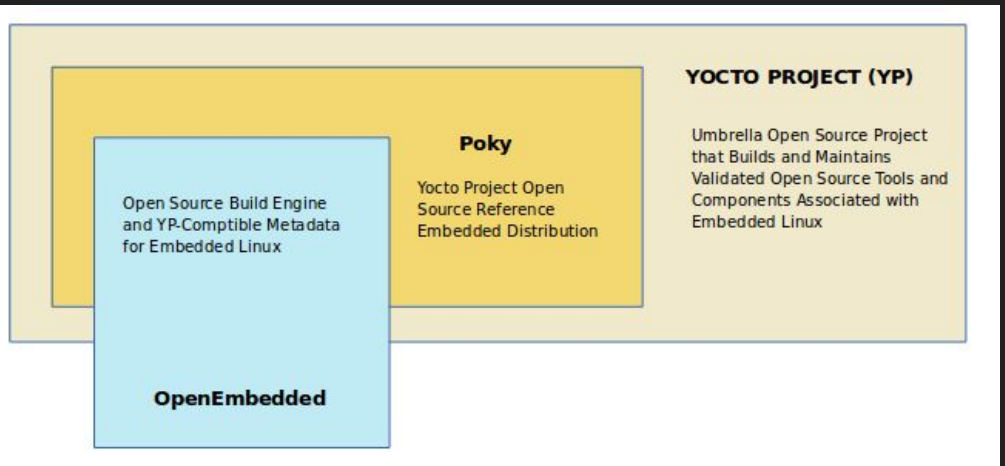
Downstream 3rd Parties

OpenEmbedded Build Flow

Yocto Project and OpenEmbedded

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

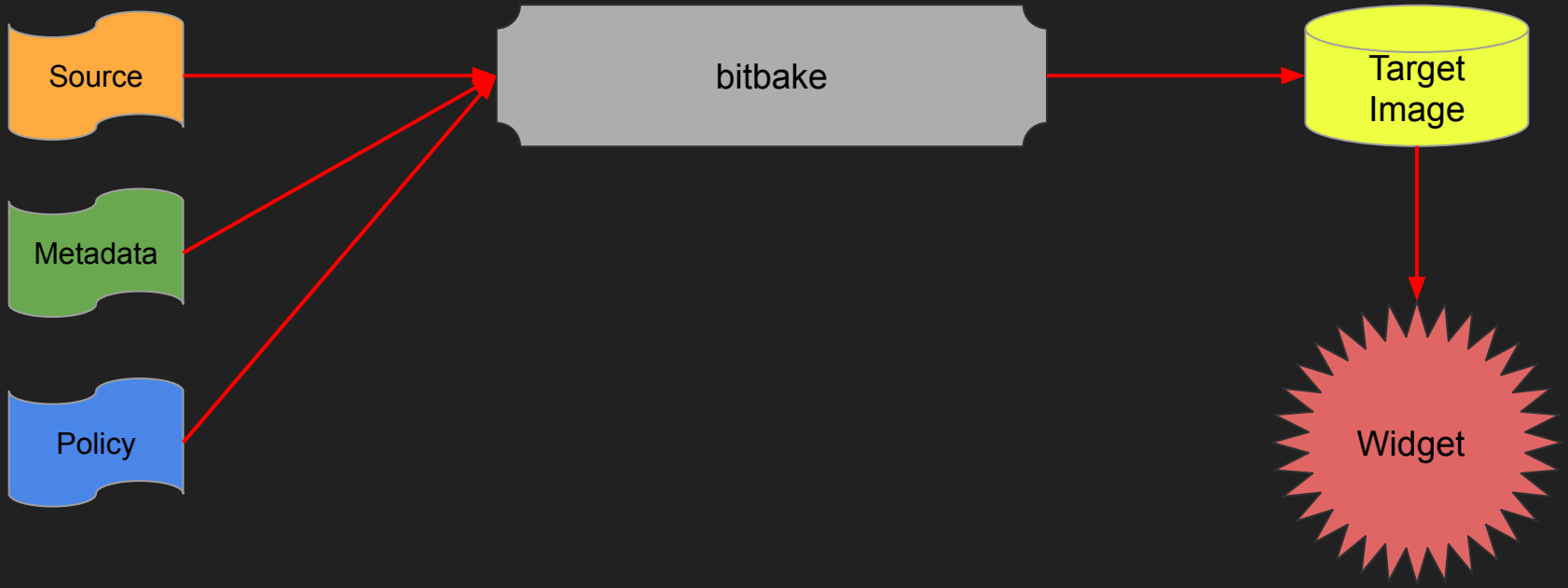
Relationship with other projects

AUTOMOTIVE GRADE LINUX

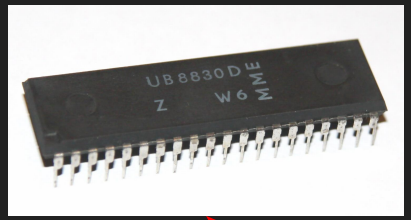
...

yocto PROJECT

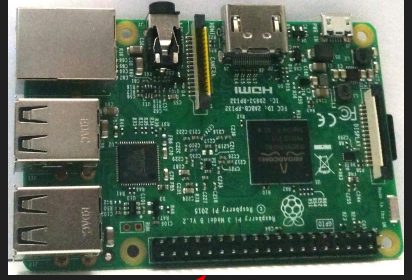
Build Images from Source Code



Images



Procolotor, CC BY 3.0 via Wikimedia Commons



QEMU

Target Image

SDK

eSDK

buildtools

IPK

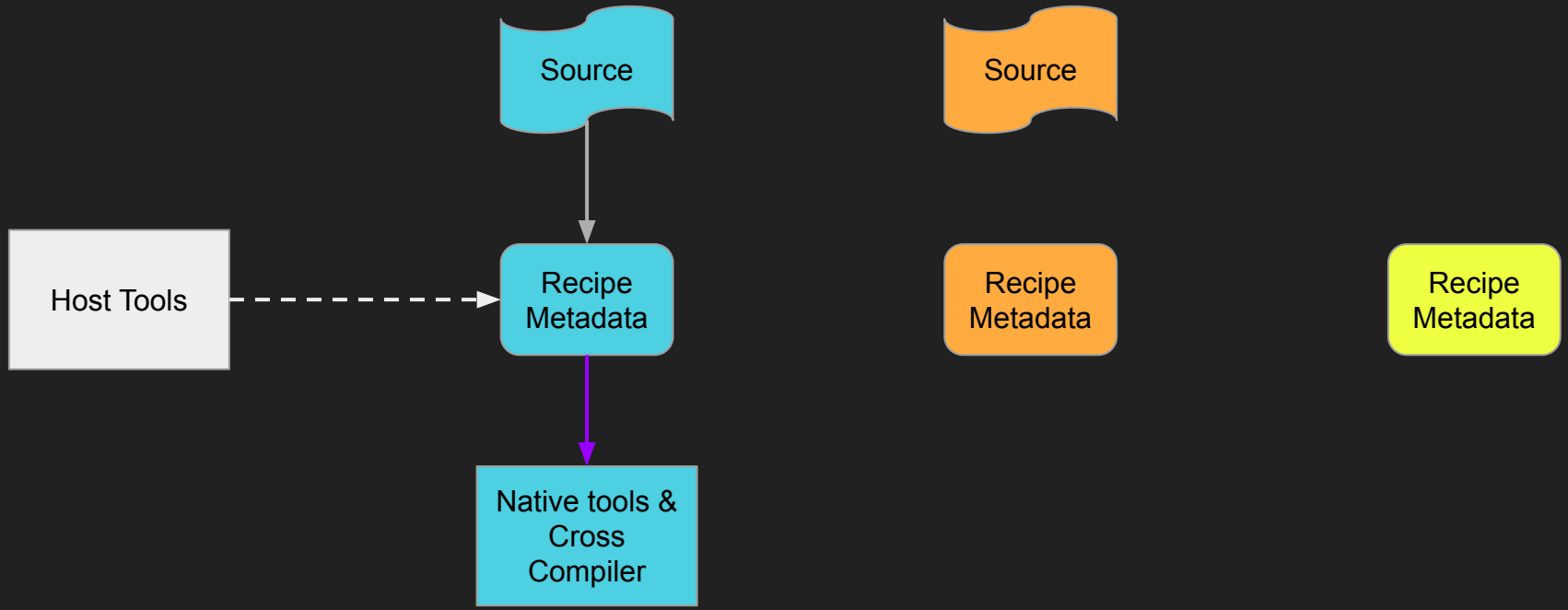
DEB

RPM

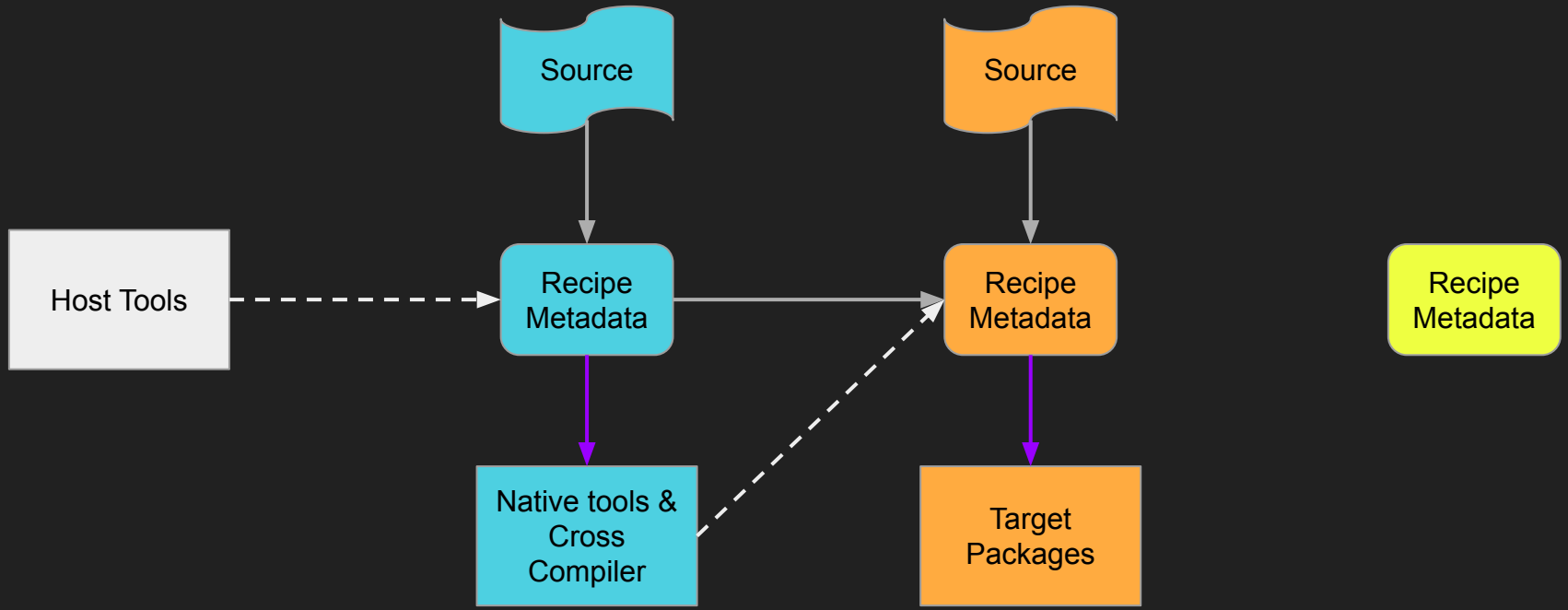
OPEN CONTAINER INITIATIVE

Simplified Build Flow

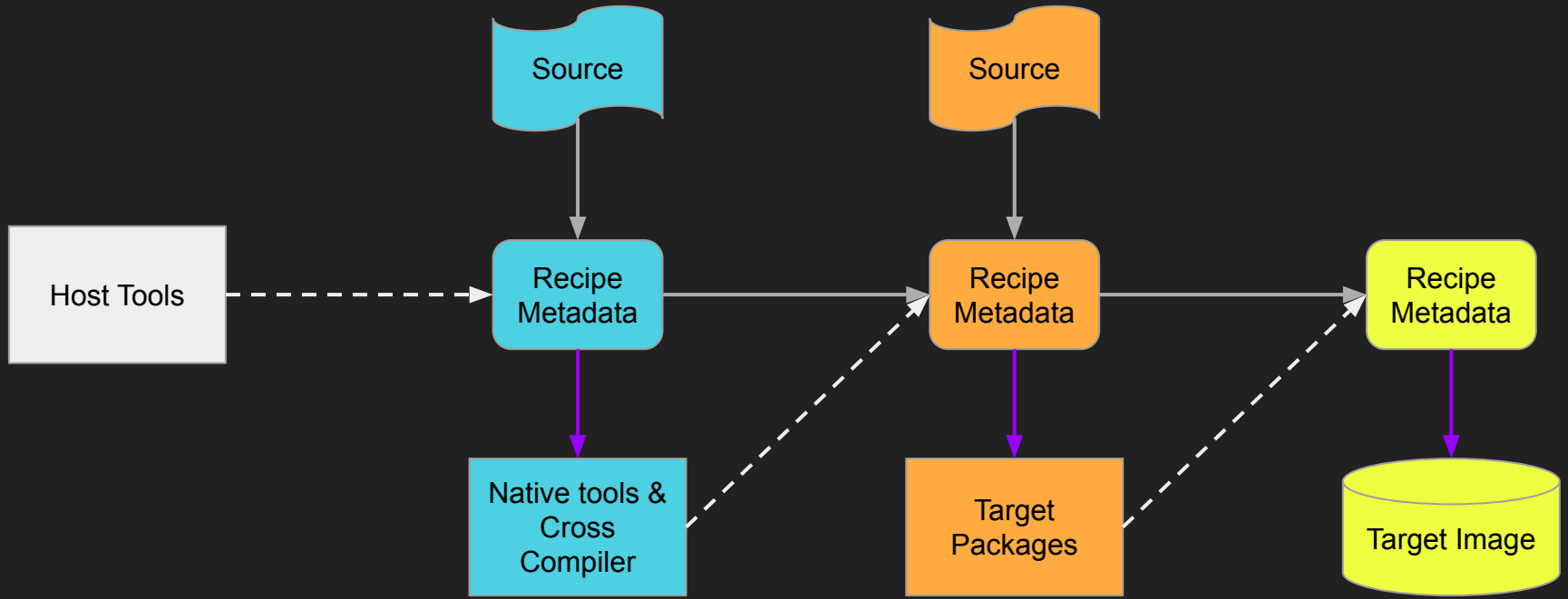
Simplified Build Flow



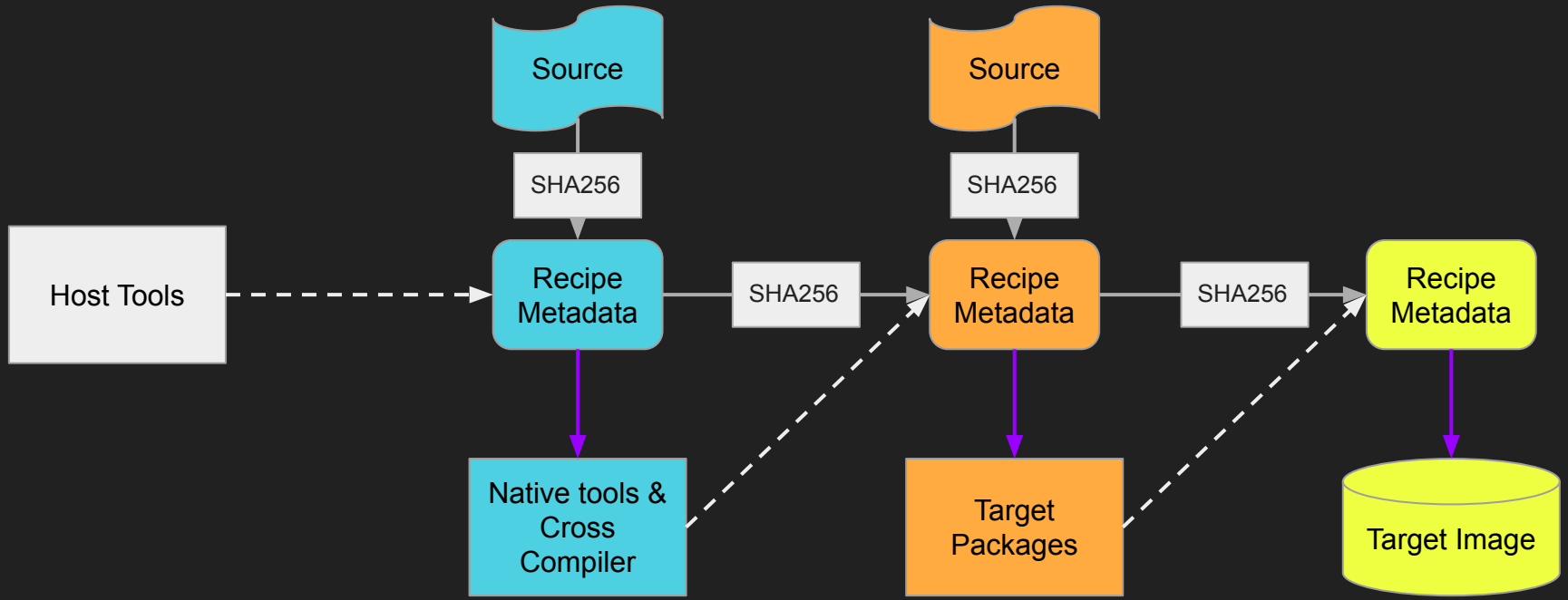
Simplified Build Flow



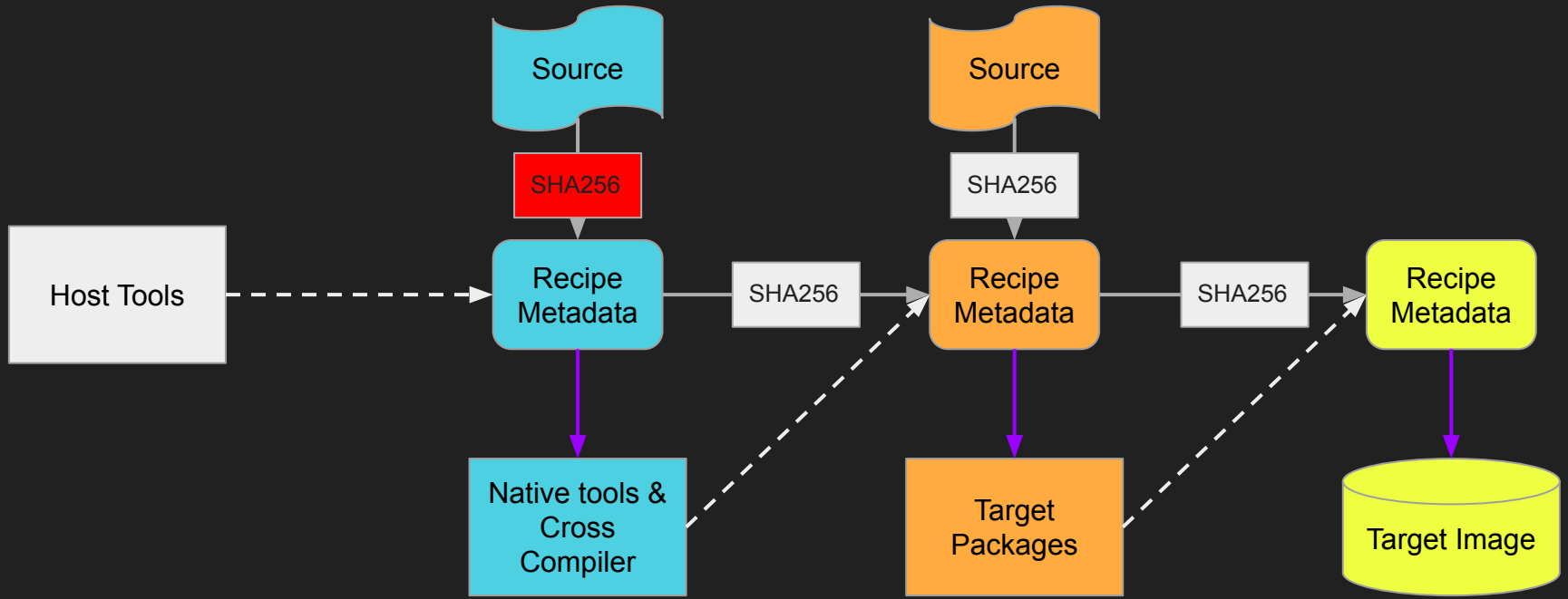
Simplified Build Flow



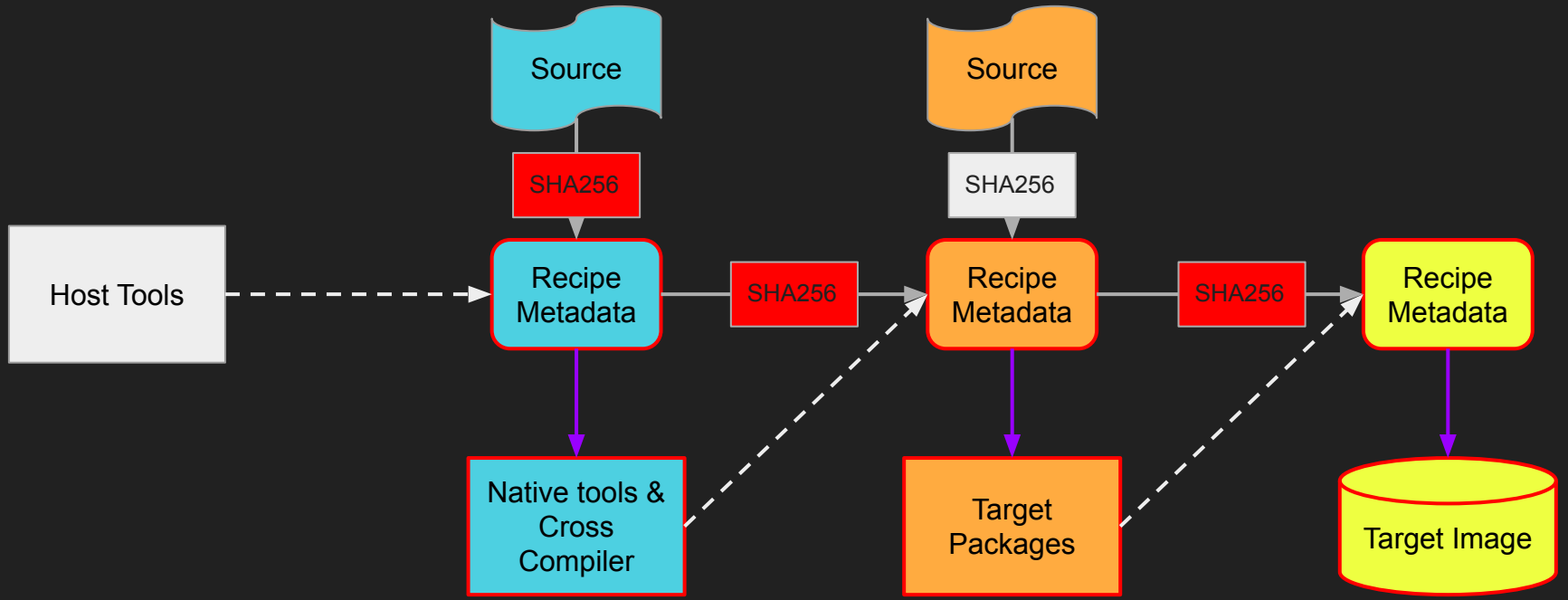
Simplified Build Flow



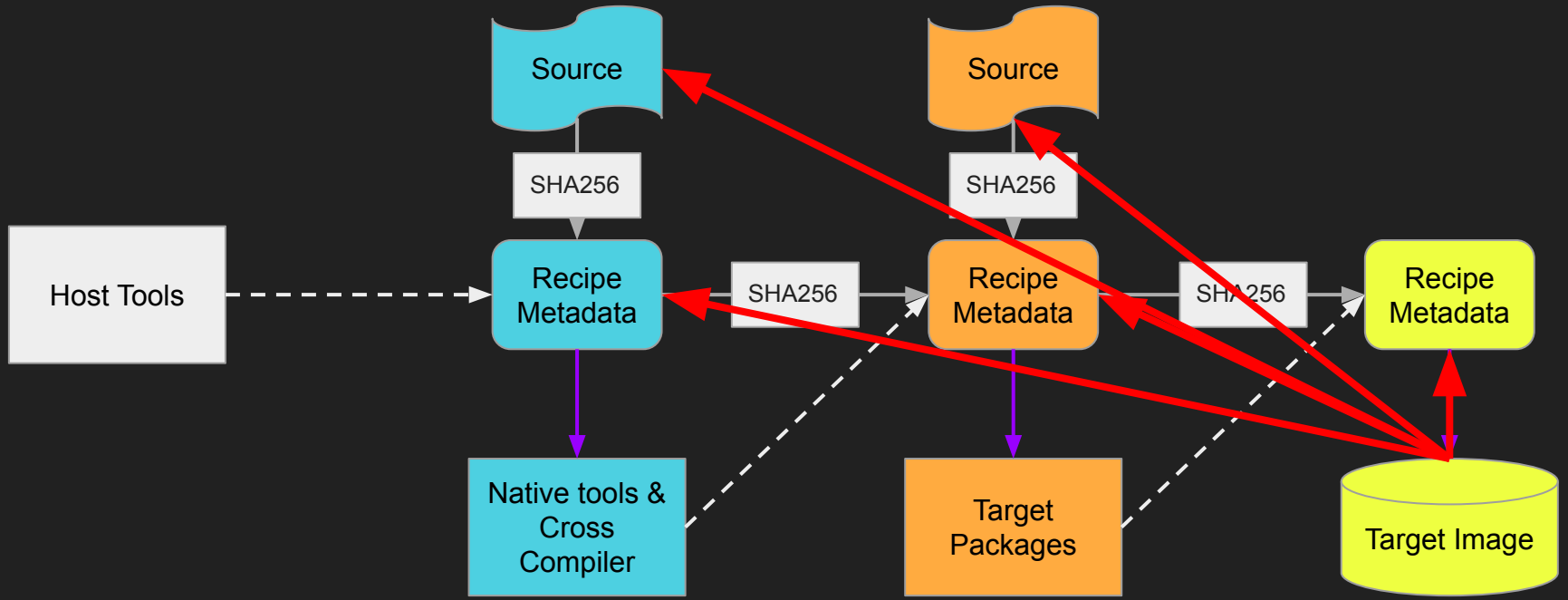
Simplified Build Flow



Simplified Build Flow

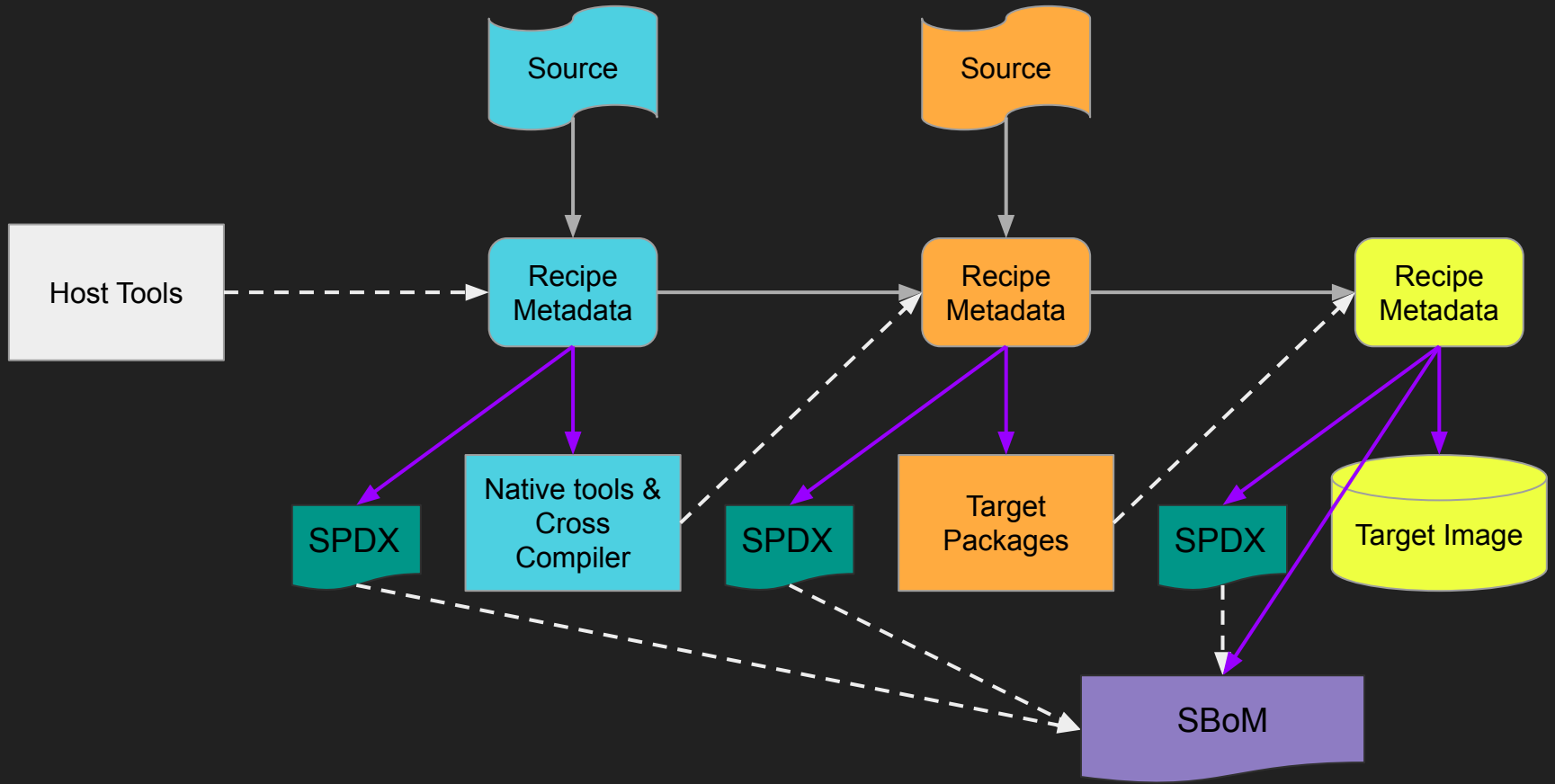


Simplified Build Flow



Traces the target image back to the code (and metadata)

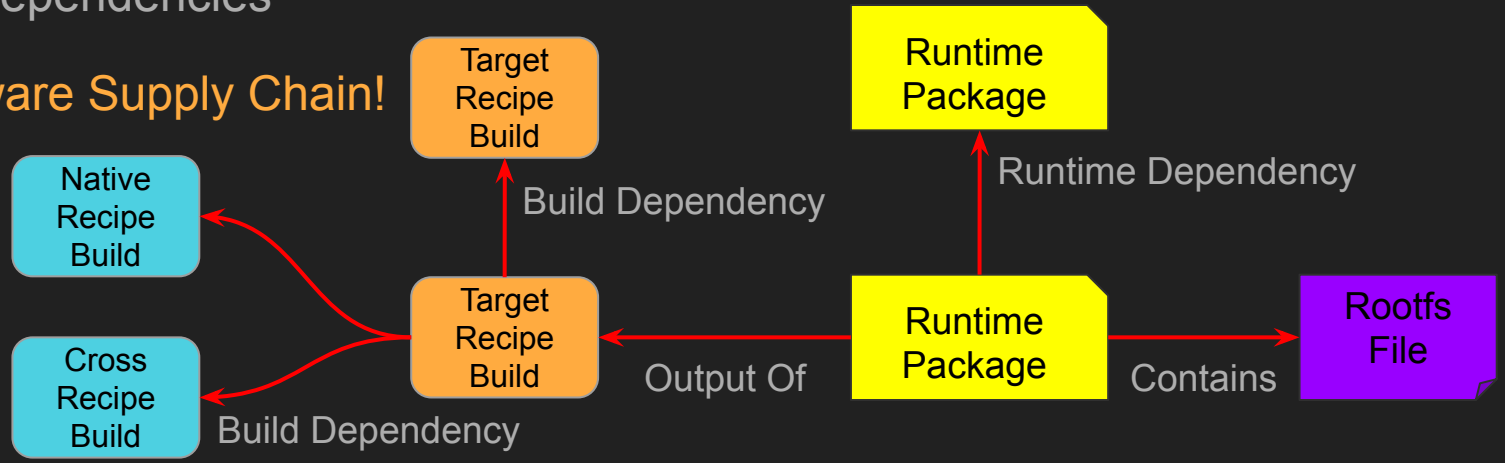
SPDX Generation



Extensive Dependency Tracking

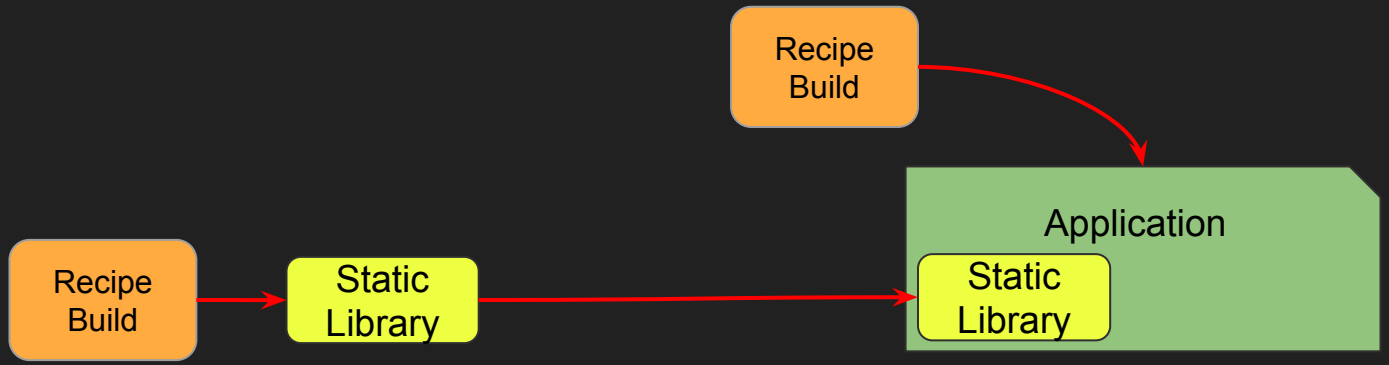
- Build Time Dependencies
  - Regular ("target") build dependencies
  - Cross Compile Tooling (i.e. "cross" recipes)
  - Host Tool Dependencies (i.e "native" recipes")
  - Buildtools tarball can extend this even further!
- Runtime Dependencies

Excellent Software Supply Chain!

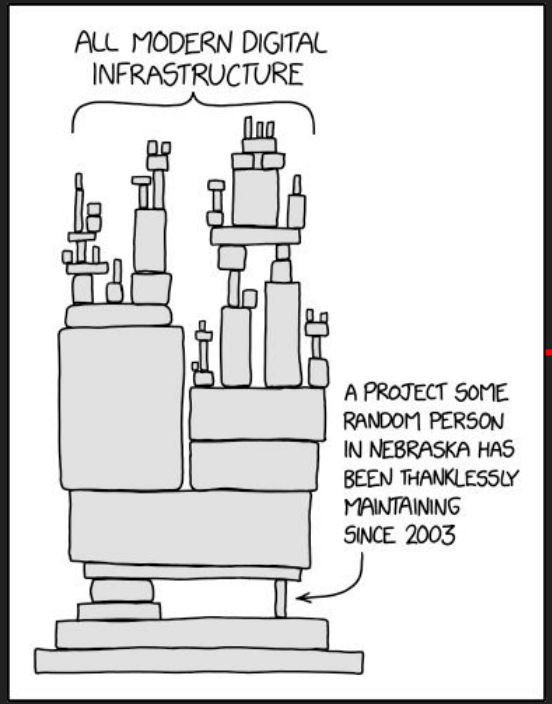


Tracking of Static Library

- We can (and do) track static library sources embedded in binaries back to the original recipe (and source code) that produced them
- This is possible because we have all the source code *and* the debug data at the time of SPDX generation
- This is fairly unique for build systems



Modern Software Supply Chains



MAGIC!

Widget

About SBoMs…

- Yocto produces SPDX 3.0 SBoMs
- Hours of SBoMs: My SBoM YouTube playlist

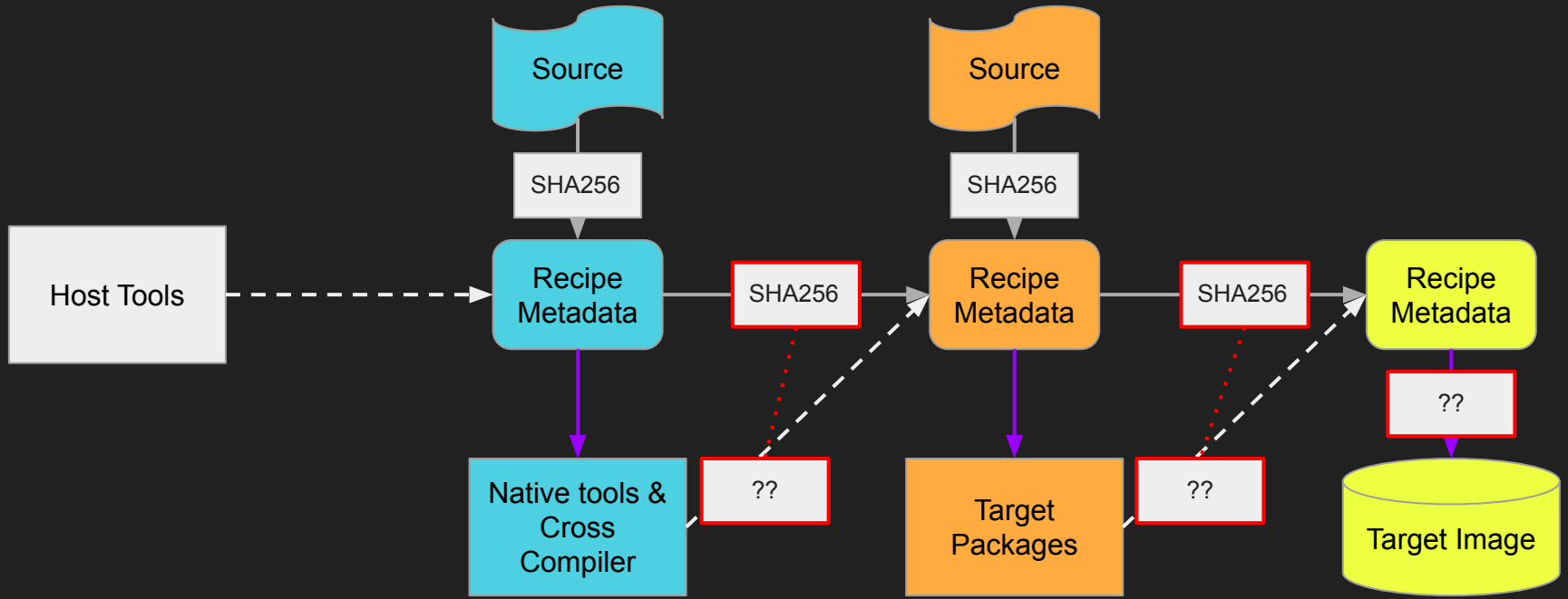
Reproducible Builds

Why do we need reproducible builds?

- Resist attack
  - What binaries need more scrutiny?
- Compiler Trust
  - Diverse Double-Compilation (David A. Wheeler) requires reproducible builds
- Quality Assurance
  - Rare timing bugs, race conditions, locale dependencies
- Smaller Binary Differences
  - Better delta updates
- Increased Development Speed
  - No need to rebuild if nothing has changed

https://reproducible-builds.org/docs/buy-in/

Binary output should associate with recipe hashes



Reproducibility Testing

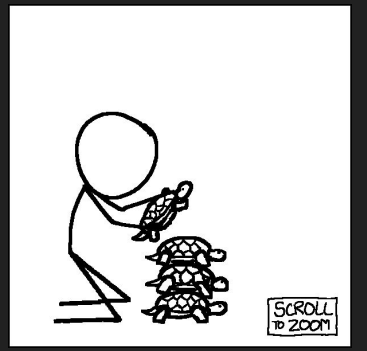
- Yocto Autobuilder tests regularly for regressions
- https://www.yoctoproject.org/reproducible-build-results/
- ~12,000 *target* packages
- 3 Package formats (ipk, deb, rpm)
- Multiple build hosts (Fedora, Ubuntu, CentOS, Debian)
  - Ensures cross-host builds are reproducible!
- Automatic diffoscope HTML output for packages that are not reproducible

Extending Quality Assurance Test

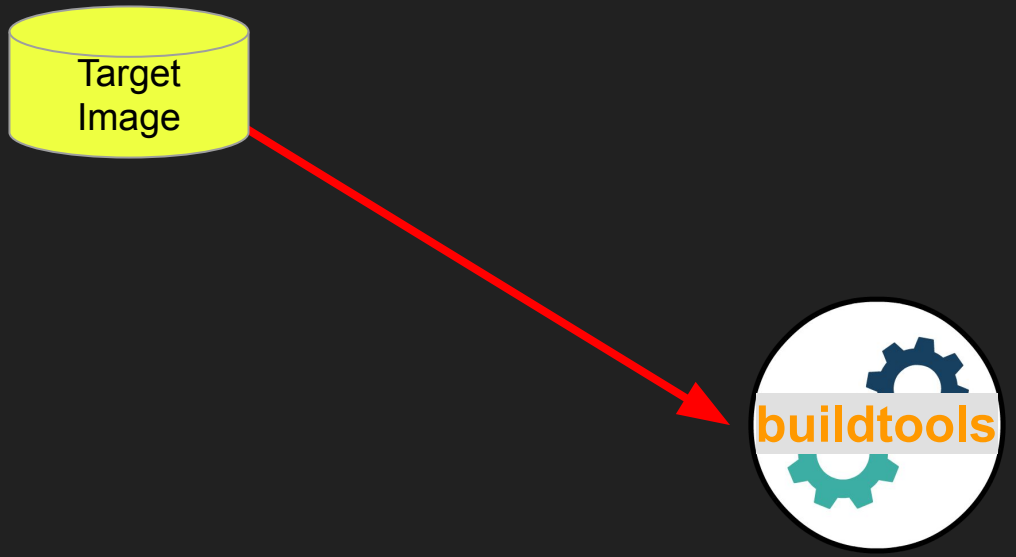
- The QA test for reproducibility is designed to be easy to extend and run for testing your own images:

```
$ cat lib/oeqa/selftest/cases/myreproducible.py
from oeqa.selftest.cases.reproducible import ReproducibleTests

class MyReproTests(ReproducibleTests):
    images = [‘my-image’]



$ oe-selftest -r myreproducible
```

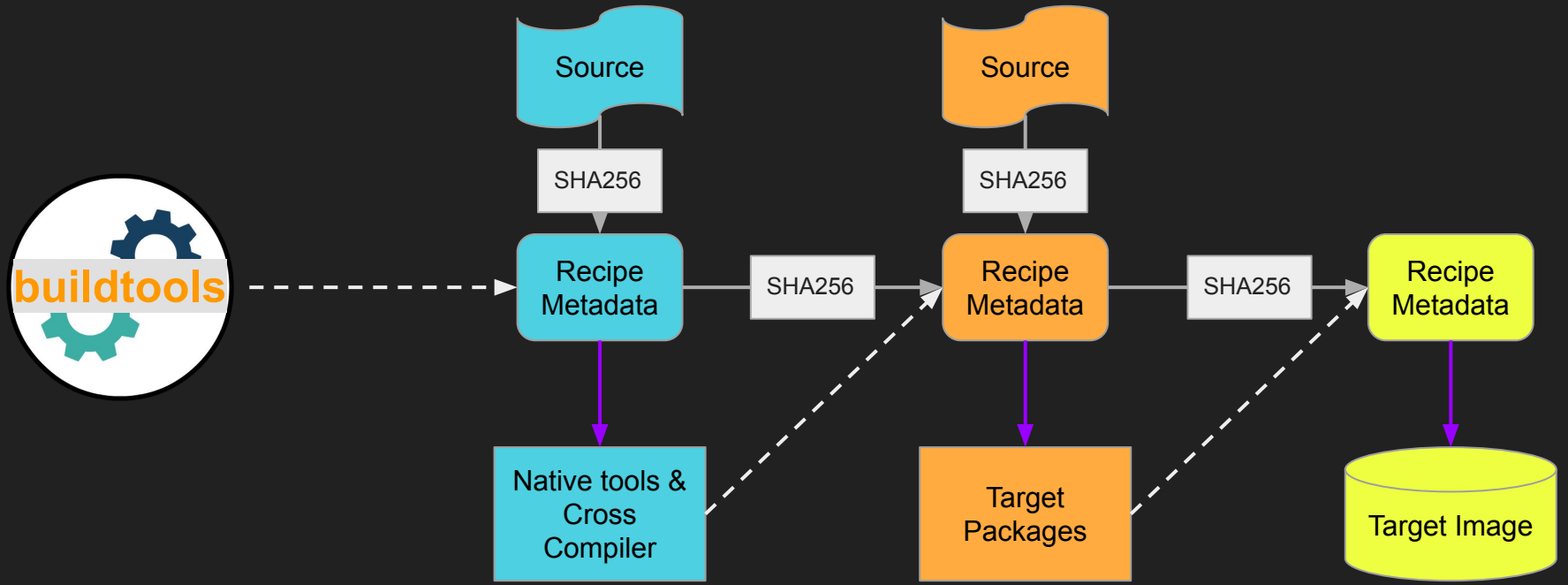
Buildtools Tarball
"It's SBoMs all the way down"



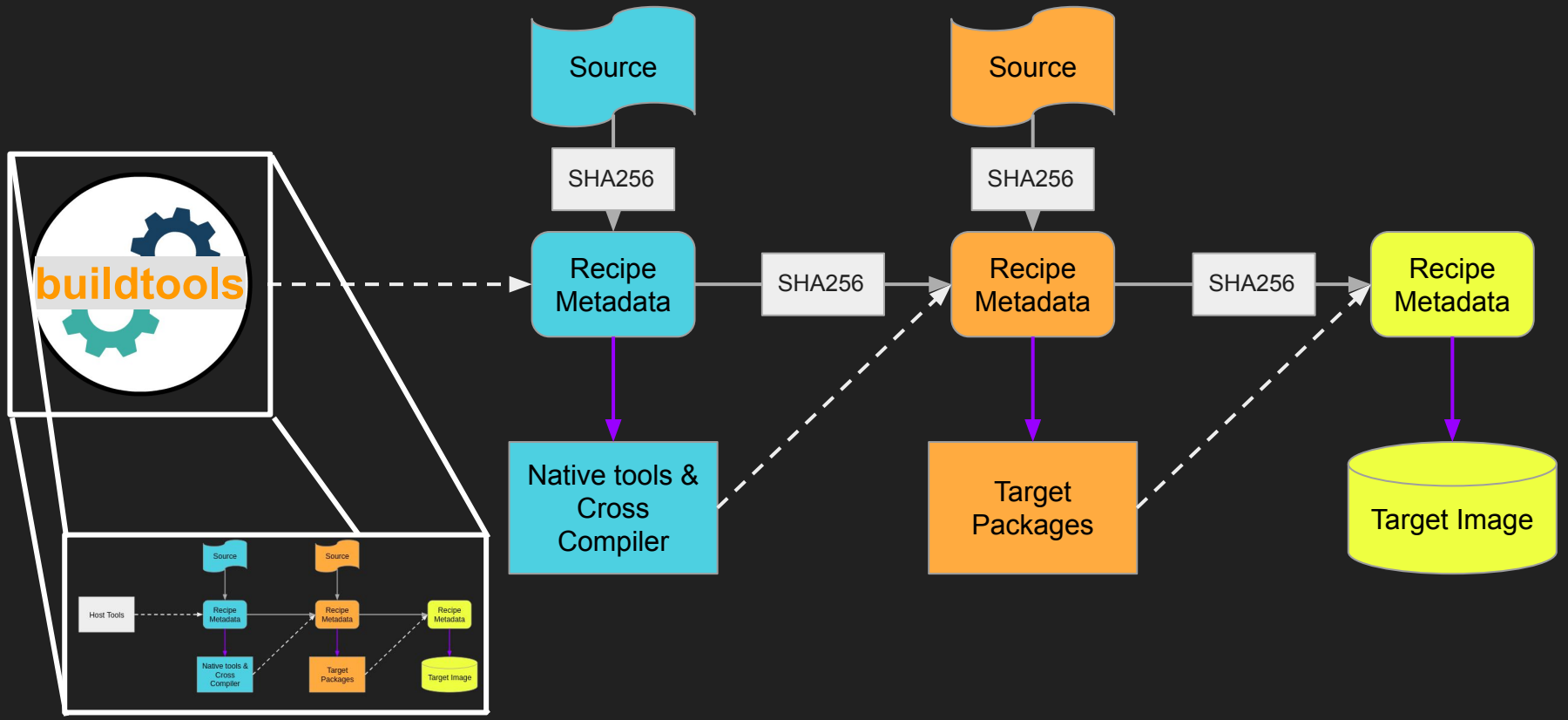
Images



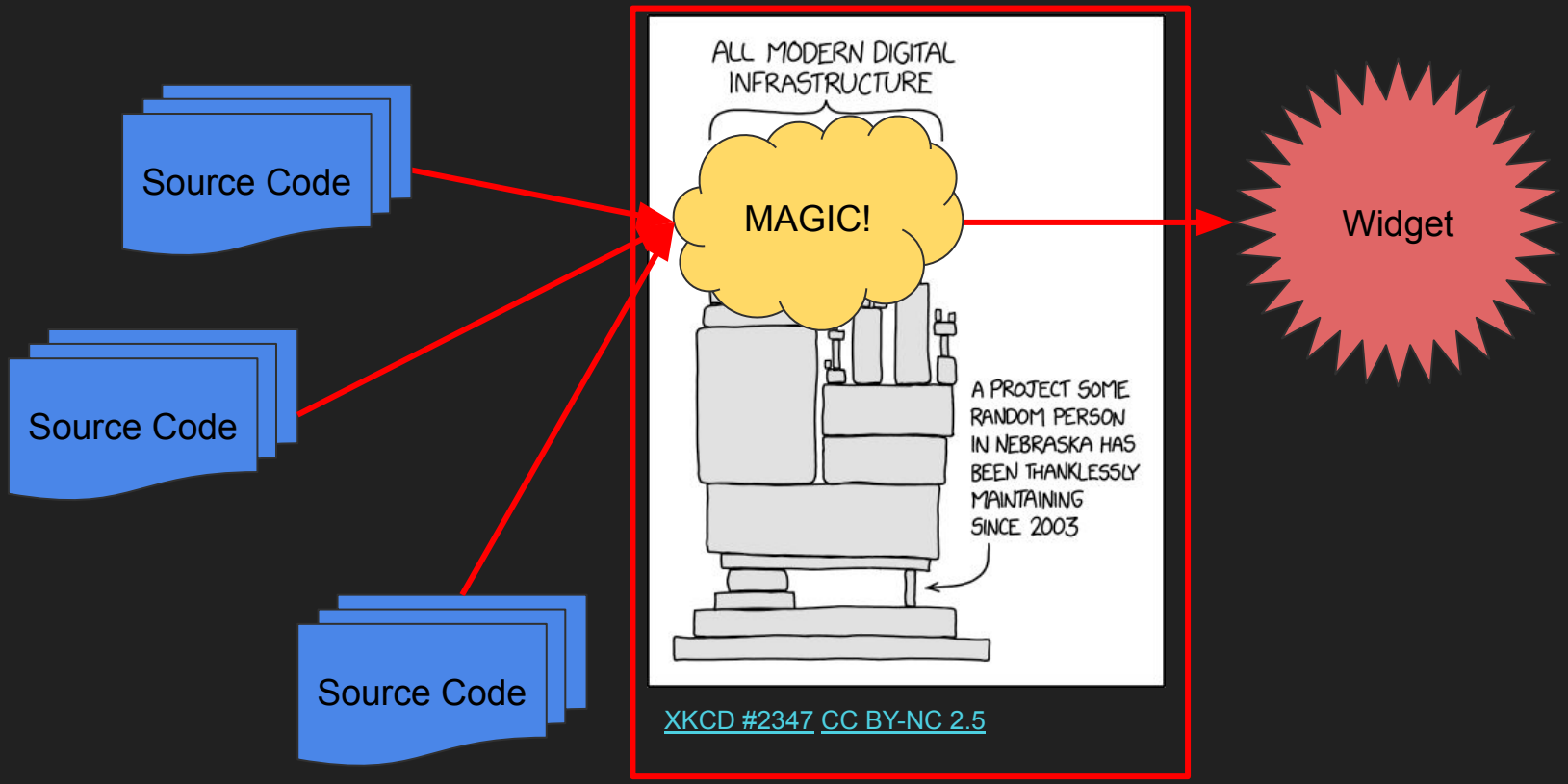
Buildtools replaces Host tools



Buildtools replaces Host tools



Modern Software Supply Chains



Getting Involved

- Yocto
  - Libera IRC
    - #oe
    - #yocto
  - https://www.yoctoproject.org/community/get-involved/
- SPDX
  - https://spdx.dev/engage/participate/

Questions?