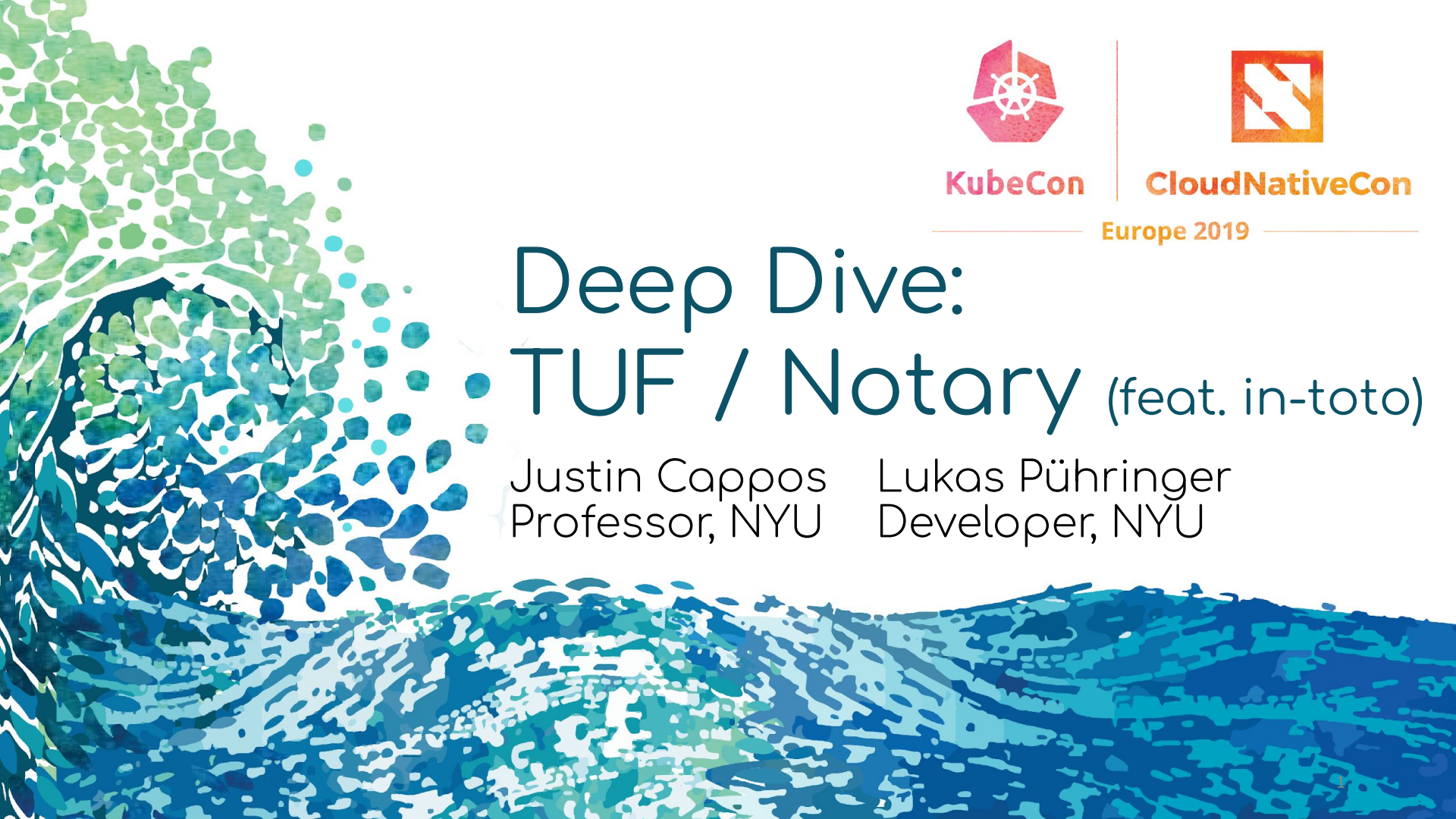


Deep Dive: TUF / Notary (feat. in-toto)

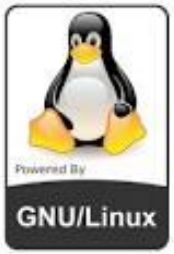
Justin Cappos
Professor, NYU

Lukas Pühringer
Developer, NYU

There exists an attack vector

- When exploited has nearly unlimited privileges
- Traditional defenses are ineffective
- Ubiquitous
- A failed attack often appears benign
- Exploited by the best hackers in the world

Many Victims...



Doesn’t crypto just work?

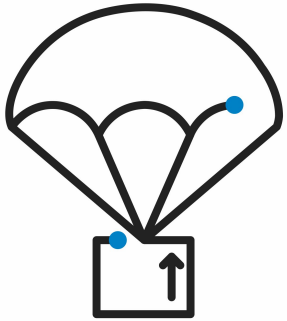
Attack: Compromise

Goal: Arbitrarily change packages

- 2008: **Fedora** signing key may have been compromised
- 2008: Attackers compromise **Red Hat** and sign malicious OpenSSH packages
- 2012: Flame exploits MD5 in **Microsoft** Windows Update
- 2012: **Adobe** revokes stolen certificate used to sign malware
- 2013: Attackers may have stolen SSL certificate for **PHP**
- 2013: Users installed malware signed with stolen **Opera** certificate
- 2014: **npm** RCE bug may have leaked SSL keys
- 2016: Malware gang steals code-signing certificates
- [many recent incidents omitted]
- etc…

- Mechanism: Compromise a package repo
  - including HSMs, signing keys, etc.

Enter TUF!



Be compromise-resilient!
Key management, etc.
Support, don't judge!

TUF Goal: Compromise-Resilience

Goal: Minimize damage from attack

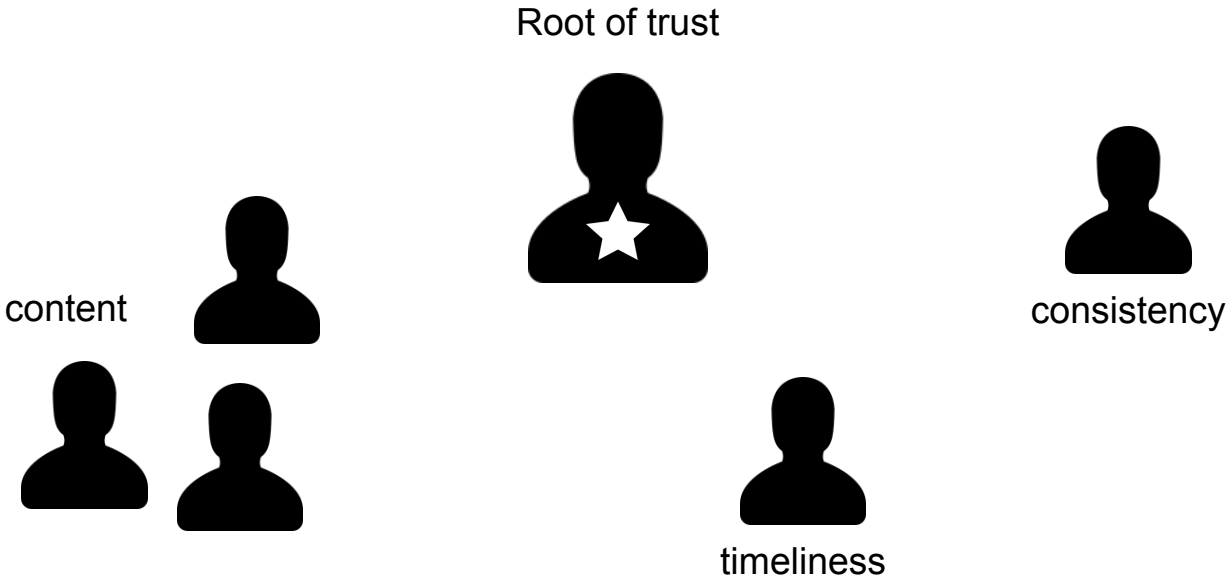
- Minimize likelihood of successful attack
- Contain impact of successful attacks

Explicit & Implicit Revocation

Revocation

Expiration

Responsibility Separation



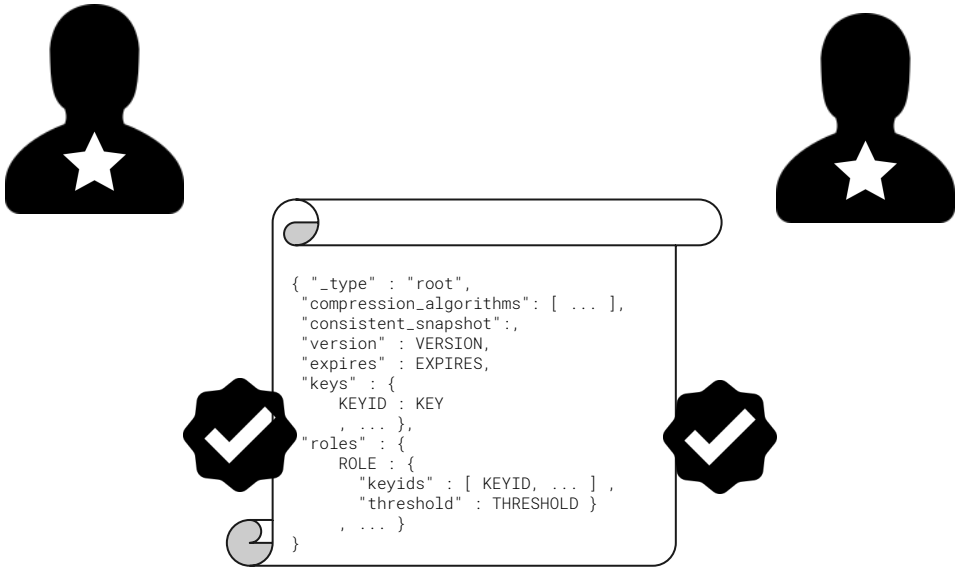
Minimize Risk


```
DAMAGE ~= PROBABILITY x IMPACT
```

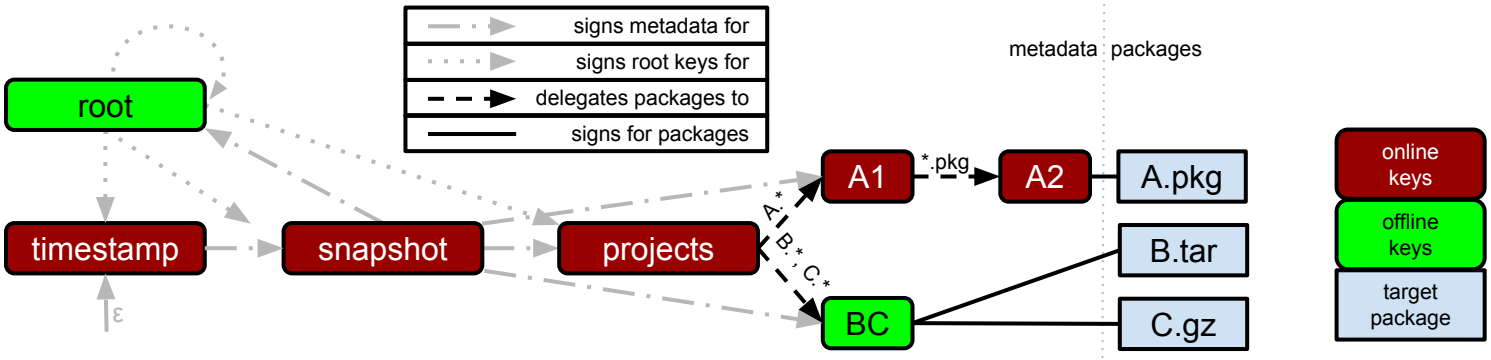
High-impact role? ⟶ Highly secure keys

Online keys? ⟶ Low-impact role

Multi-signature Trust



Roles in TUF



1. **Responsibility separation**.
2. **Multitrust signatures** (a.k.a. two-man rule).
3. **Explicit and implicit revocation of keys**.
4. **Minimizing risk** (with offline keys).

Standardization process (TAPs)

TAP 3 -- multi-role signatures over unequal quorums

TAP 4 -- pinning repository keys

TAP 5 -- split repository location across URLs

TAP 6 -- version numbers in root metadata

TAP 7 -- TUF conformance testing

TAP 8 -- Key rotation / self revocation [1.0]

TAP 9 -- Mandated metadata signing scheme

TAP 10 -- Remove native compression support

In progress -- clearer versioning support, POUF (protocol, operations, usage, and format), partially signed targets [1.0]

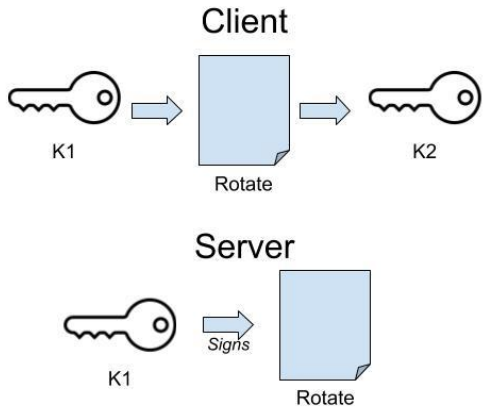
TAP 8: Key Rotation

Problem: A compromised key requires **all** delegators to revoke it

Rotating a key requires **all** delegators to change their delegation

Solution: rotate file points to next valid key

Allow the owner of a key to rotate (or self-revoke) a key



TAP: Managing TUF Versions

Problem: Want to make breaking changes

- Clients and Repos both need to update

Solution: Check the TUF version before an update

- Update client before updating metadata
- Repository stores old “transition” metadata for each major version

Arbitrary changes to TUF formats that can be non-backwards compatible

[Also, standardize on Semantic Versioning (major.minor.patch)]

TAP: POUF

Problem: Meet TUF security goals

- Legacy support
- Interoperate (when desired)

Solution: Record implementation / deployment choices

- It includes the Protocol, Operations, Usage, and Formats (POUF)

Allows implementers to interoperate (conform to the same POUF)

Describes deployments so others can learn

TAP: Handling Partially Signed Metadata

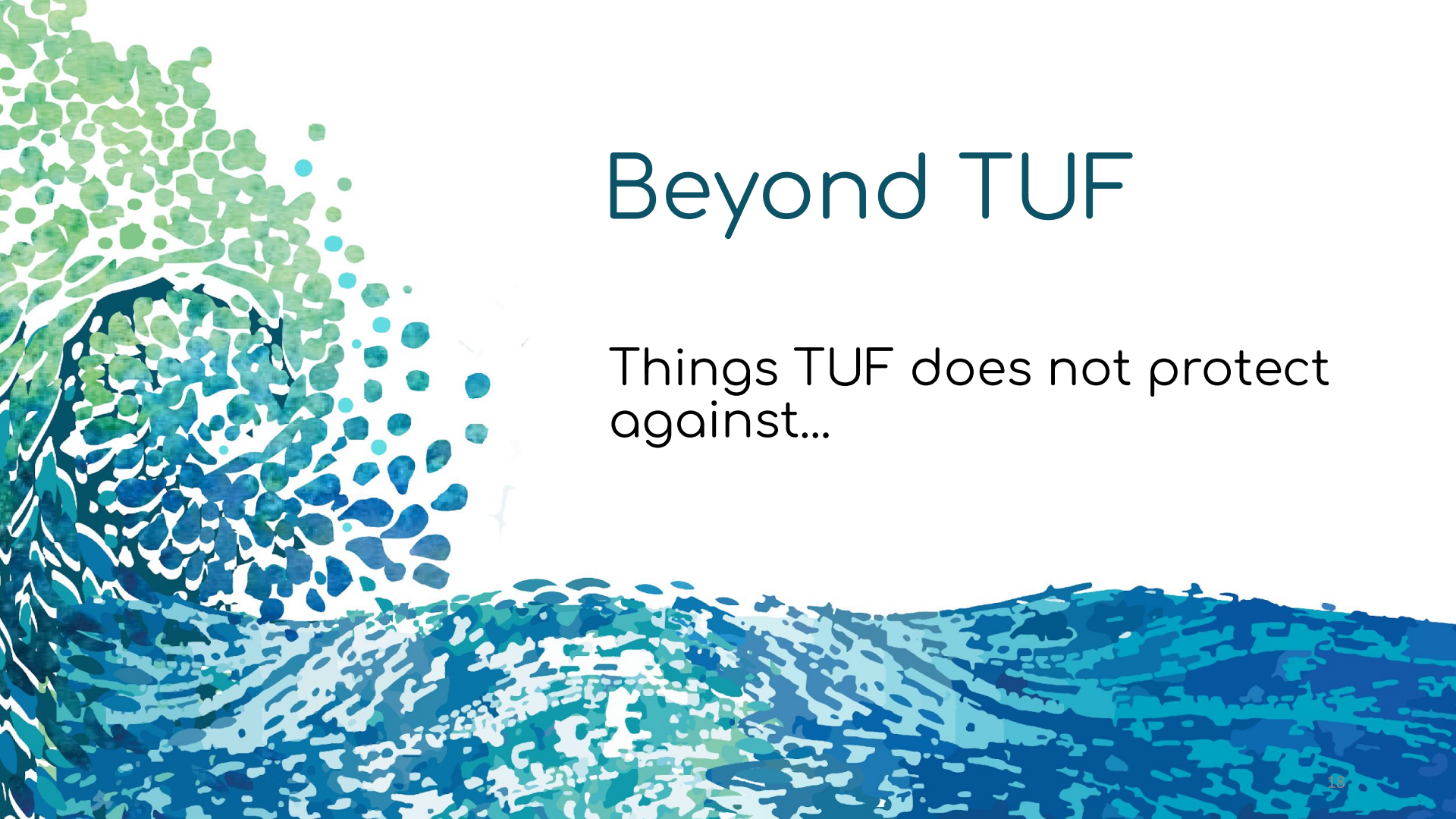
Problem: Threshold signatures aren't atomic

- Store / manage intermediate data

Solution sketch: repo stores metadata before threshold is reached

- Signatures are not over other signatures

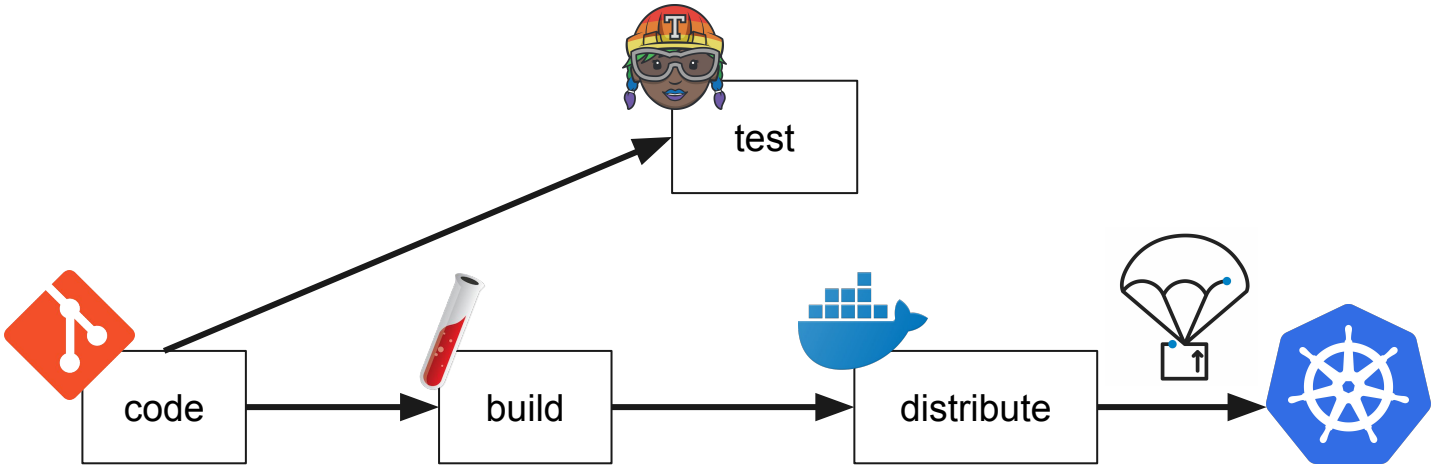
Small functionality add to repo, no TUF client change



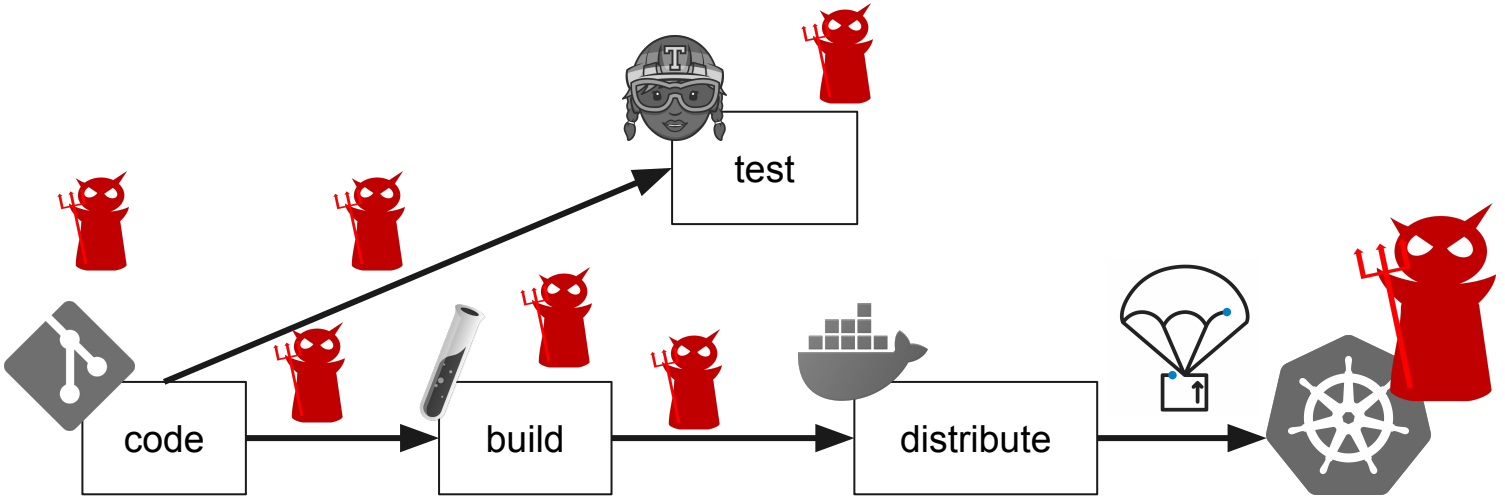
Beyond TUF

Things TUF does not protect against...

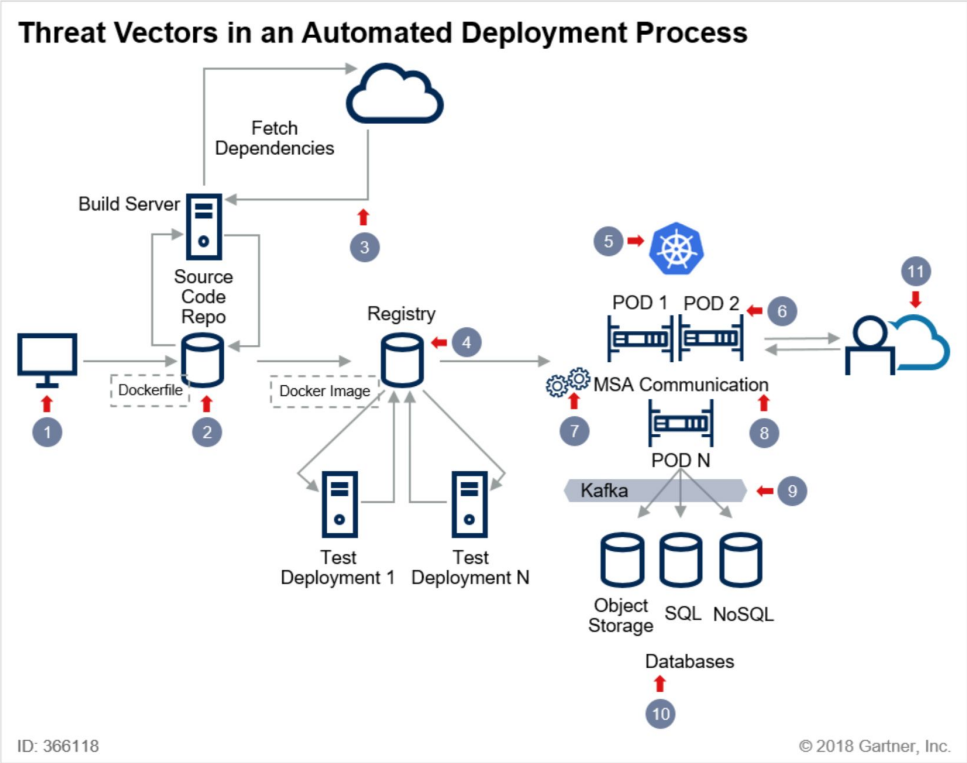
The Software Supply Chain



The Attack Surface



Imagine In The Real World...



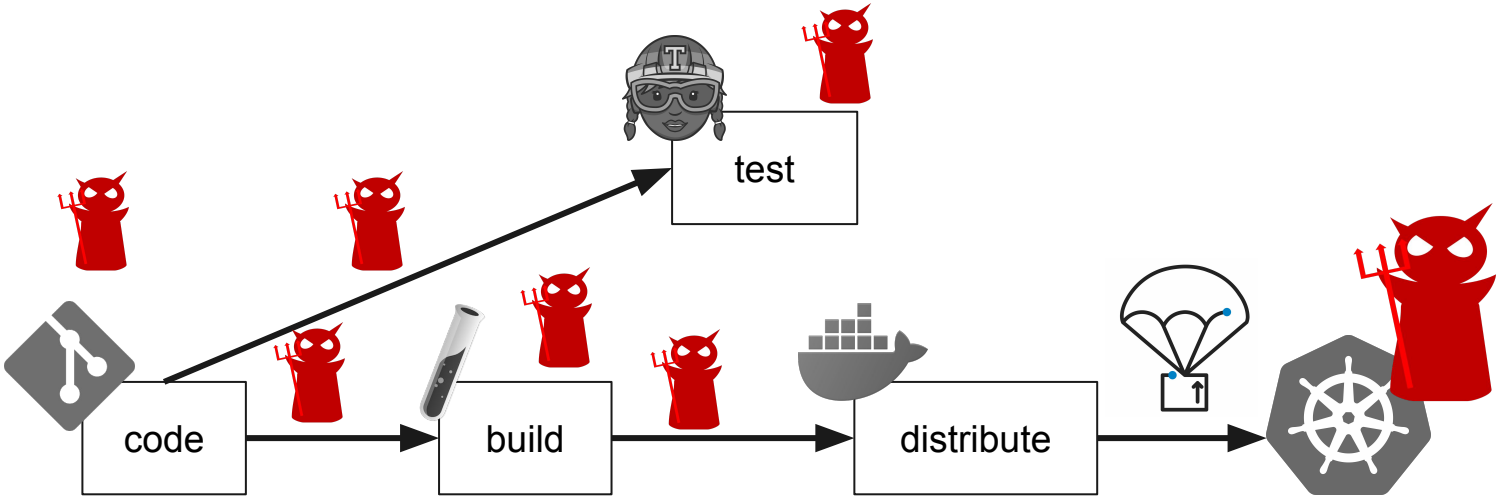
From Gartner, *Container Security — From Image Analysis to Network Segmentation, Options Are Maturing* by Joerg Fritsch and Michael Isbitski, 2018

And It Does Happen ...

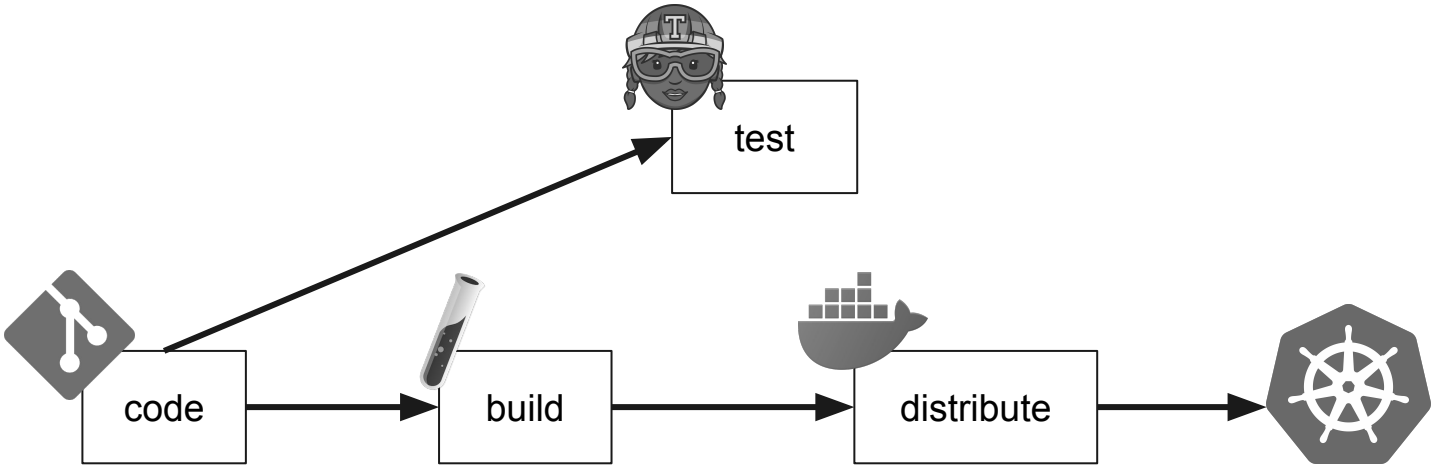
- ...
- ASUS ShadowHammer updater attack, 2019
- NPM “event-stream” hack, 2018
- PyPI “ssh-decorate”, 2018
- NotPetya, 2017
- Kingslayer, 2017
- CCleaner, 2017
- Linux Mint, 2016
- XcodeGhost, 2015
- ...

How Can We Fix This?

For The Sake Of This Talk...

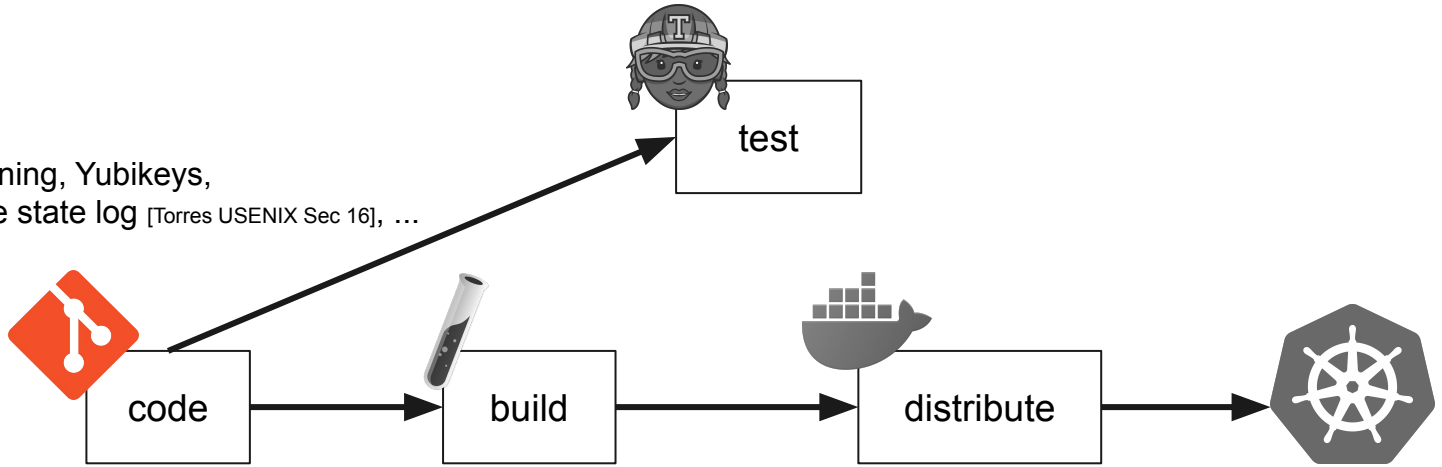


Many Good Point Solutions

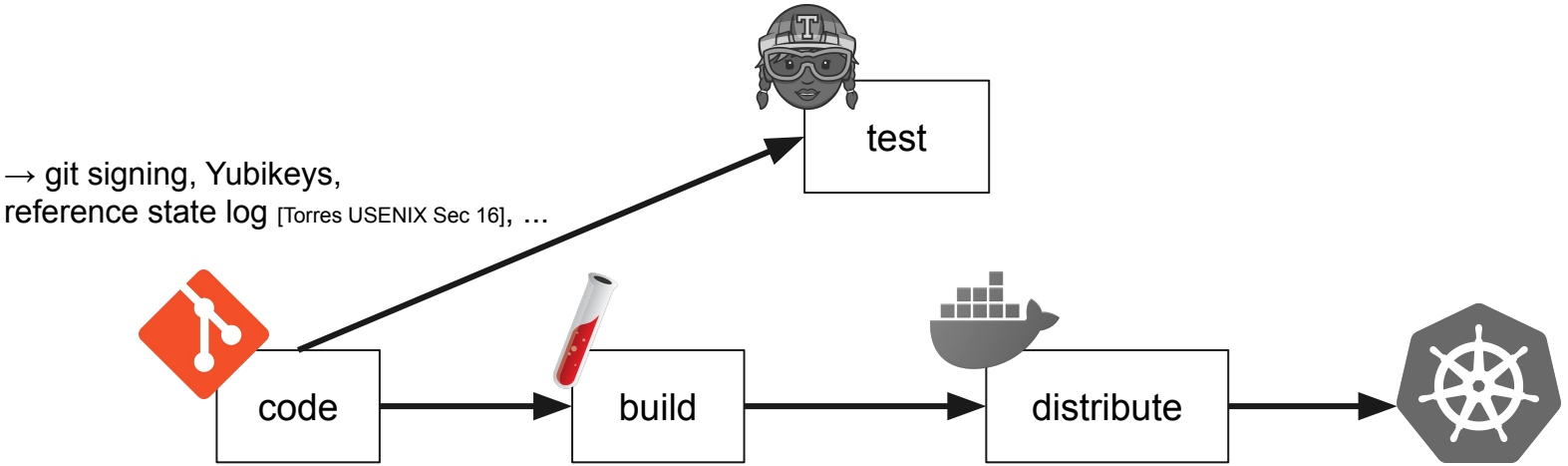


Many Good Point Solutions

→ git signing, Yubikeys,
reference state log [Torres USENIX Sec 16], …

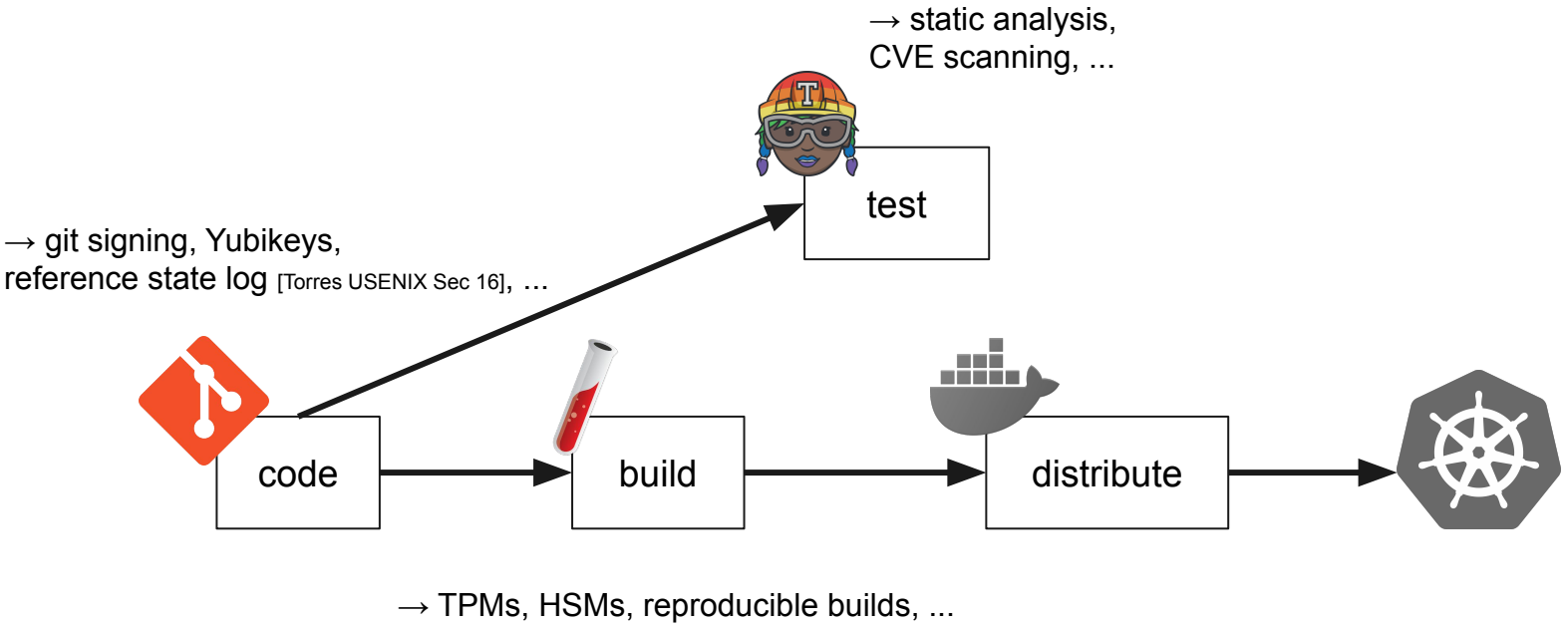


Many Good Point Solutions

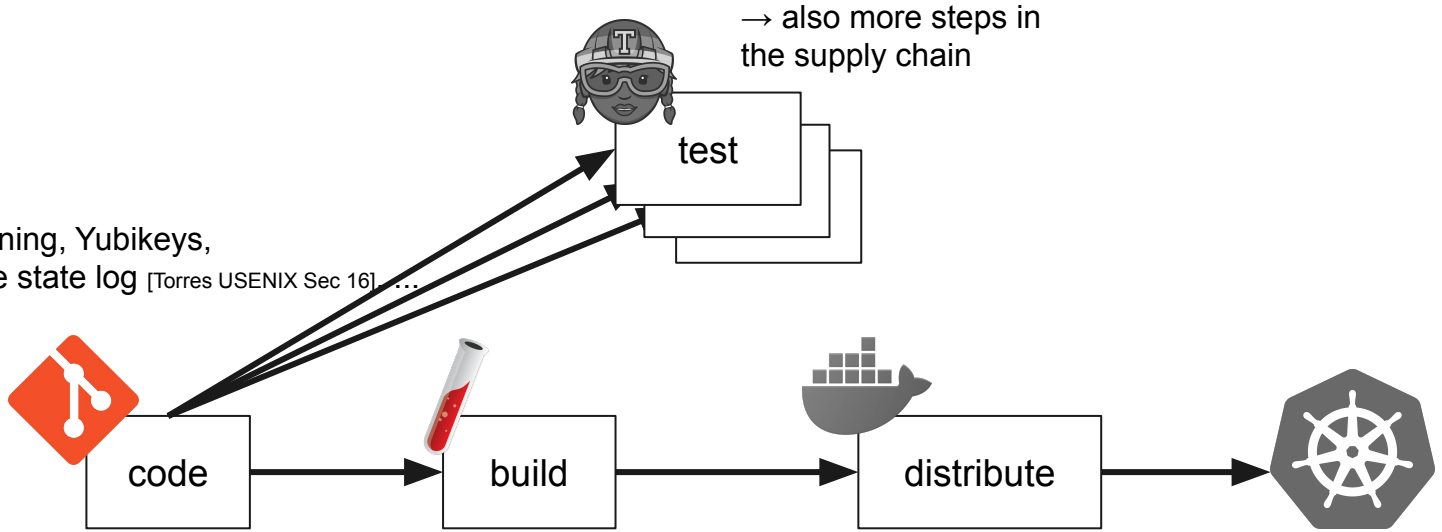


→ TPMs, HSMs, reproducible builds, ...

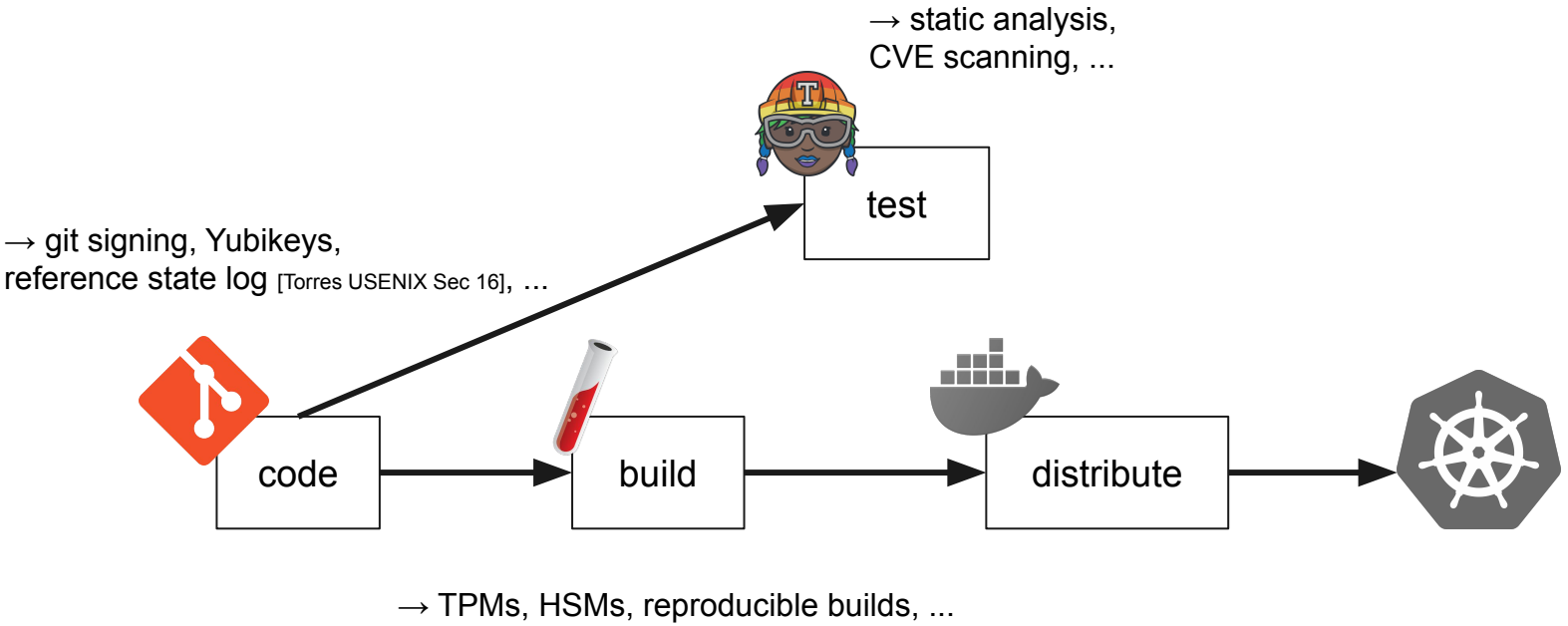
Many Good Point Solutions



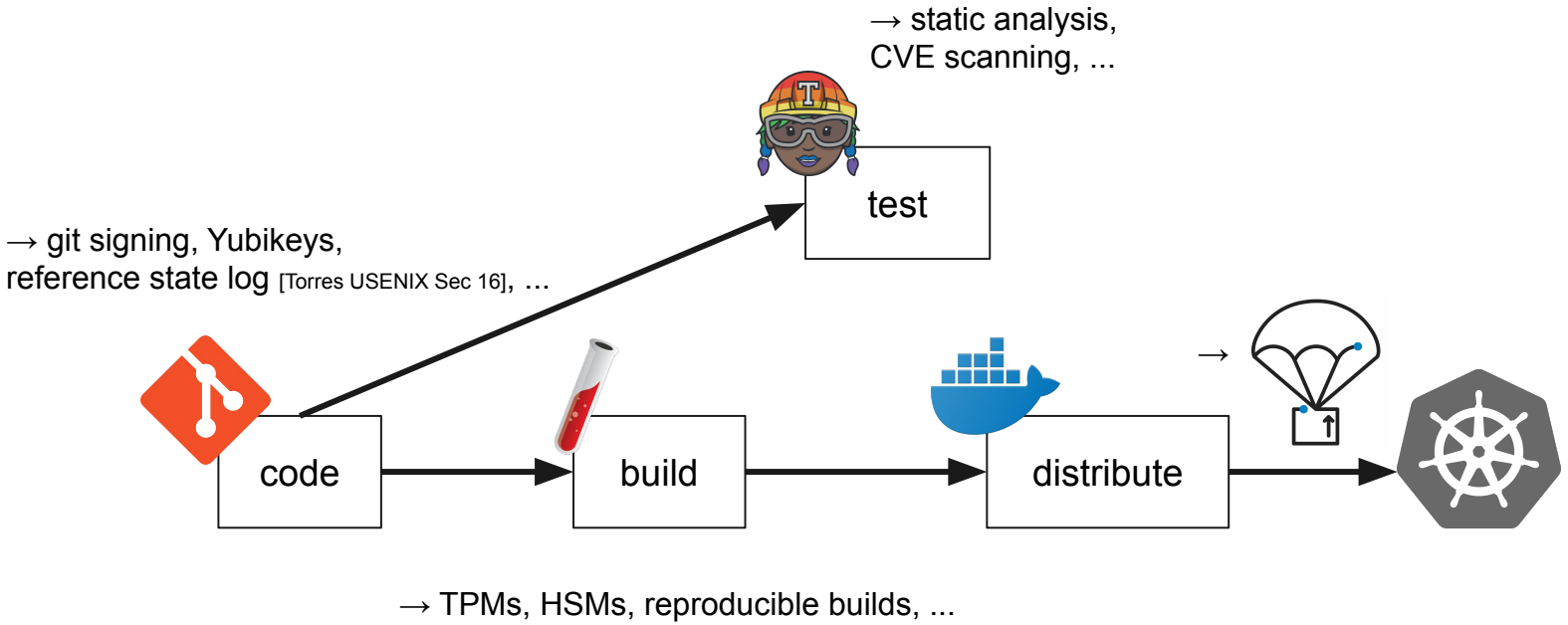
Many Good Point Solutions



Many Good Point Solutions

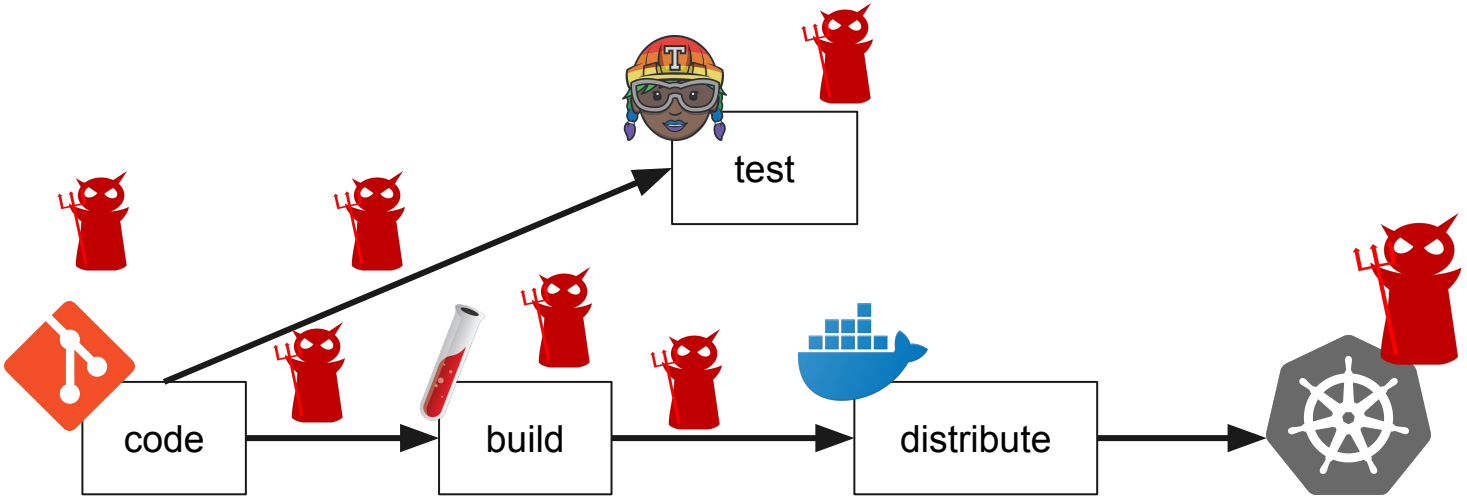


Many Good Point Solutions




Fixed?

Gaps Between Steps? Compliance?



Enter in-toto!

→ Verifiably define the steps of the software supply chain

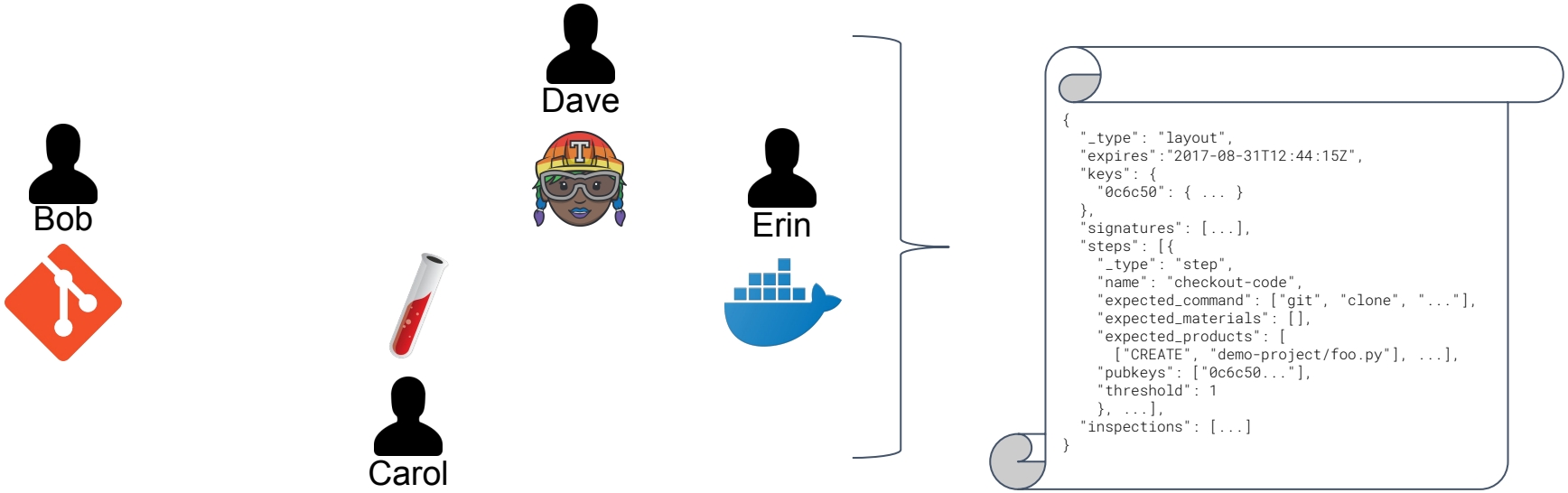
→ Verifiably define the authorized actors

→ Guarantee that everything happens according to definition

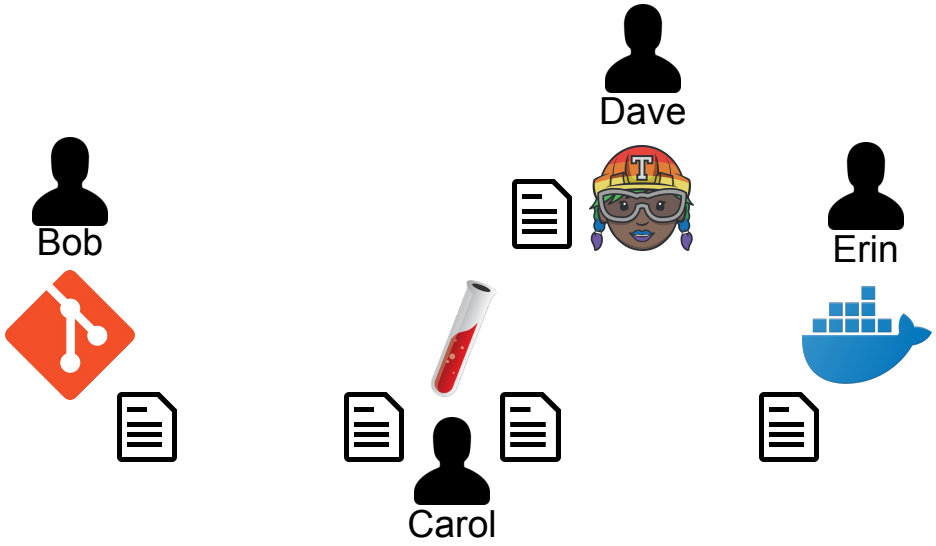
Layout -- Steps

```
{
  "_type": "layout",
  "expires":"2017-08-31T12:44:15Z",
  "keys": {
    "0c6c50": { ... }
  },
  "signatures": [...],
  "steps": [{
    "_type": "step",
    "name": "checkout-code",
    "expected_command": ["git", "clone", "..."],
    "expected_materials": [],
    "expected_products": [
      ["CREATE", "demo-project/foo.py"], ...],
    "pubkeys": ["0c6c50..."],
    "threshold": 1
    }, ...],
  "inspections": [...]
}
```

Layout -- Functionaries

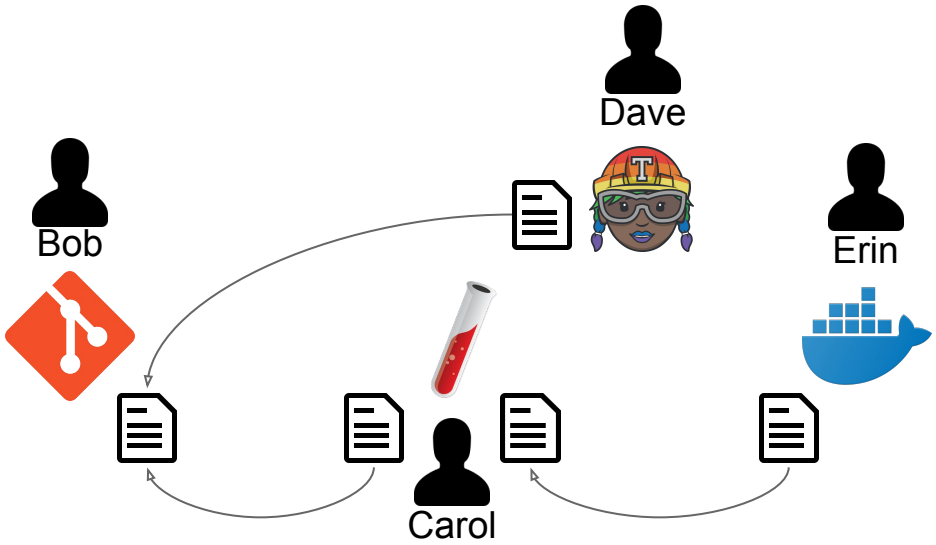


Layout -- Artifacts



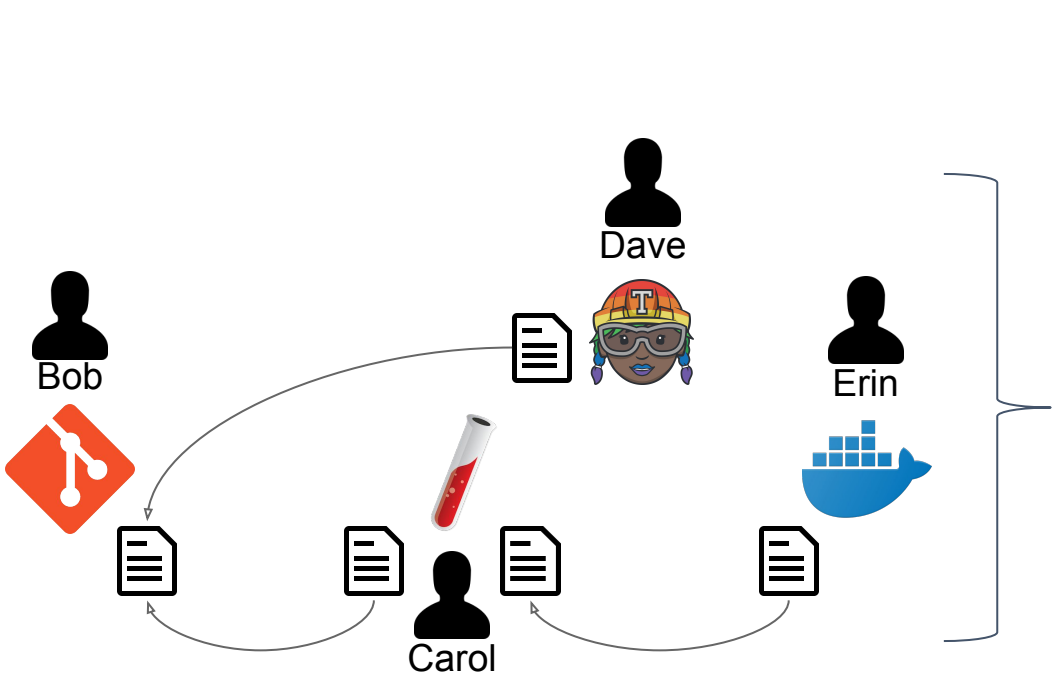

```
{
  "_type": "layout",
  "expires":"2017-08-31T12:44:15Z",
  "keys": {
    "0c6c50": { ... }
  },
  "signatures": [...],
  "steps": [{
    "_type": "step",
    "name": "checkout-code",
    "expected_command": ["git", "clone", "..."],
    "expected_materials": [],
    "expected_products": [
      ["CREATE", "demo-project/foo.py"], ...],
    "pubkeys": ["0c6c50..."],
    "threshold": 1
    }, ...],
  "inspections": [...]
}
```

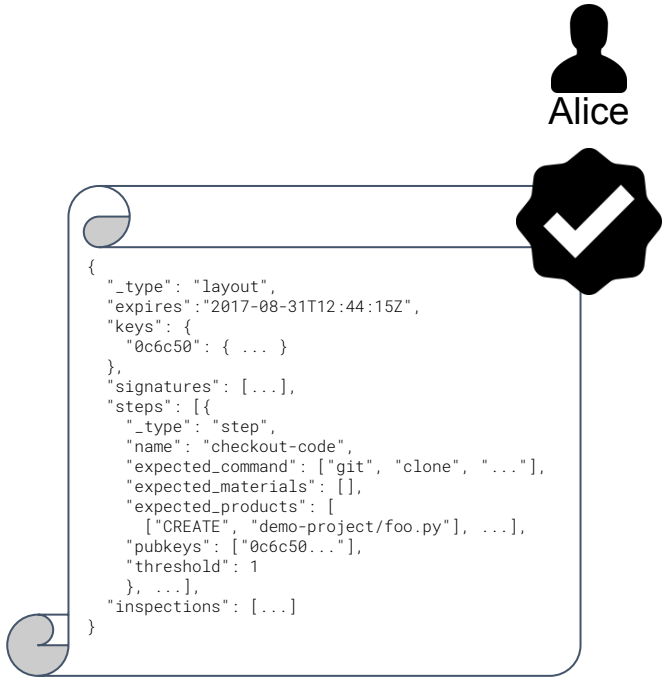
Layout -- Rules




```
{
  "_type": "layout",
  "expires":"2017-08-31T12:44:15Z",
  "keys": {
    "0c6c50": { ... }
  },
  "signatures": [...],
  "steps": [{
    "_type": "step",
    "name": "checkout-code",
    "expected_command": ["git", "clone", "..."],
    "expected_materials": [],
    "expected_products": [
      ["CREATE", "demo-project/foo.py"], ...],
    "pubkeys": ["0c6c50..."],
    "threshold": 1
    }, ...],
  "inspections": [...]
}
```

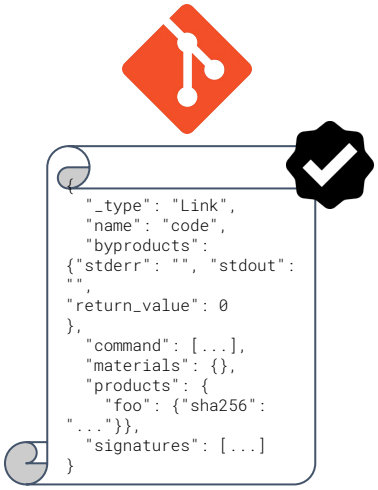
Layout -- Root Of Trust

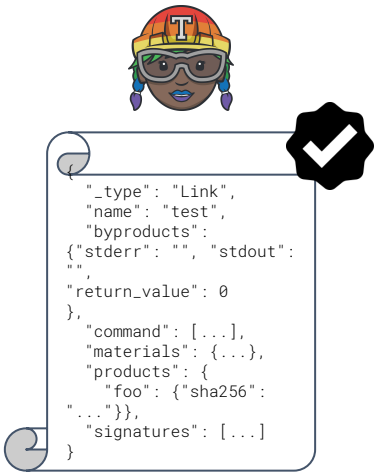


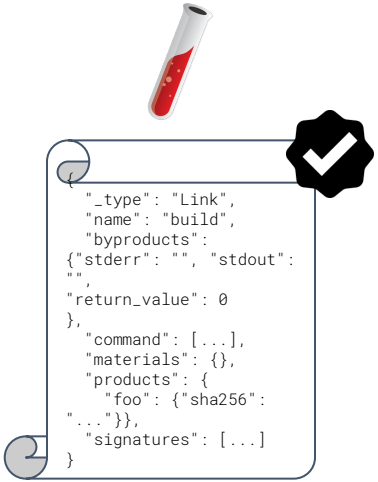


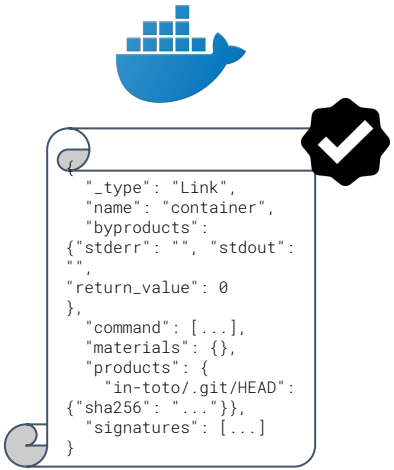
Links -- Signed Step Evidence


```
$ in-toto-run [opts] -- ./do-the-supply-chain-step
```



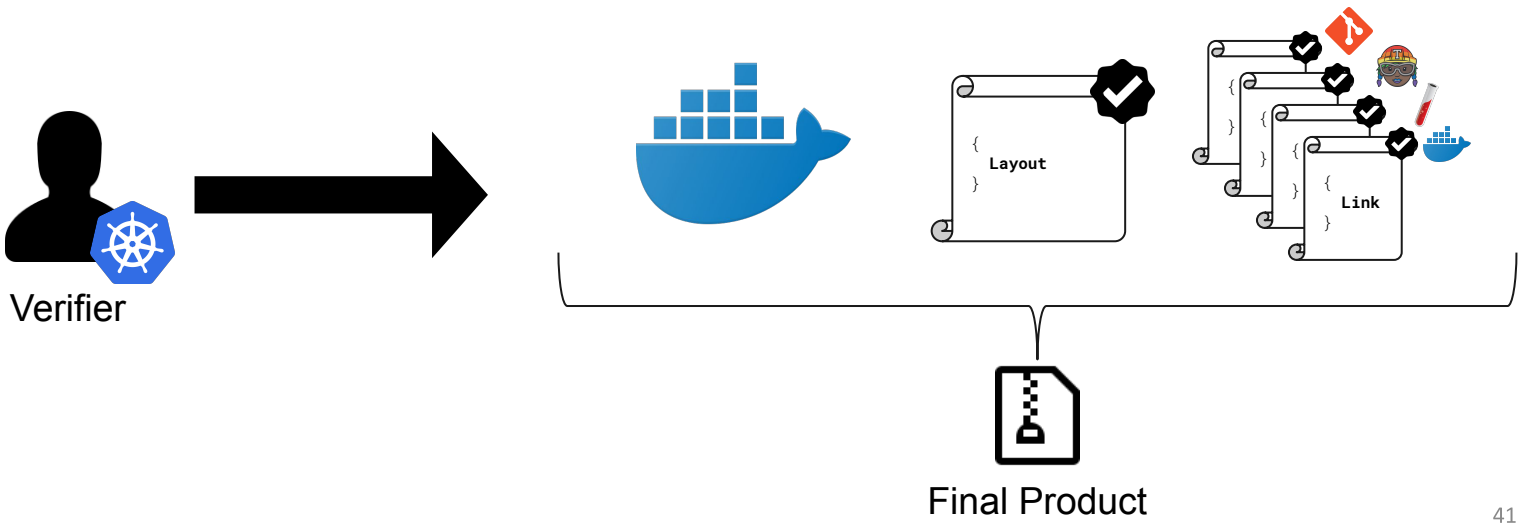






Verification/Admission


```
$ in-toto-verify --layout <layout> --key <pub key>
```



Tooling & Integrations

debian

Reproducible Builds

DATADOG

archlinux

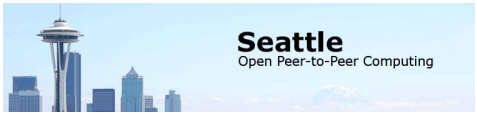
repeatr

SPDX

openSUSE

Java

GOVREADY



Thanks!

Please, do reach out to us.

in-toto.io

jcappos@nyu.edu