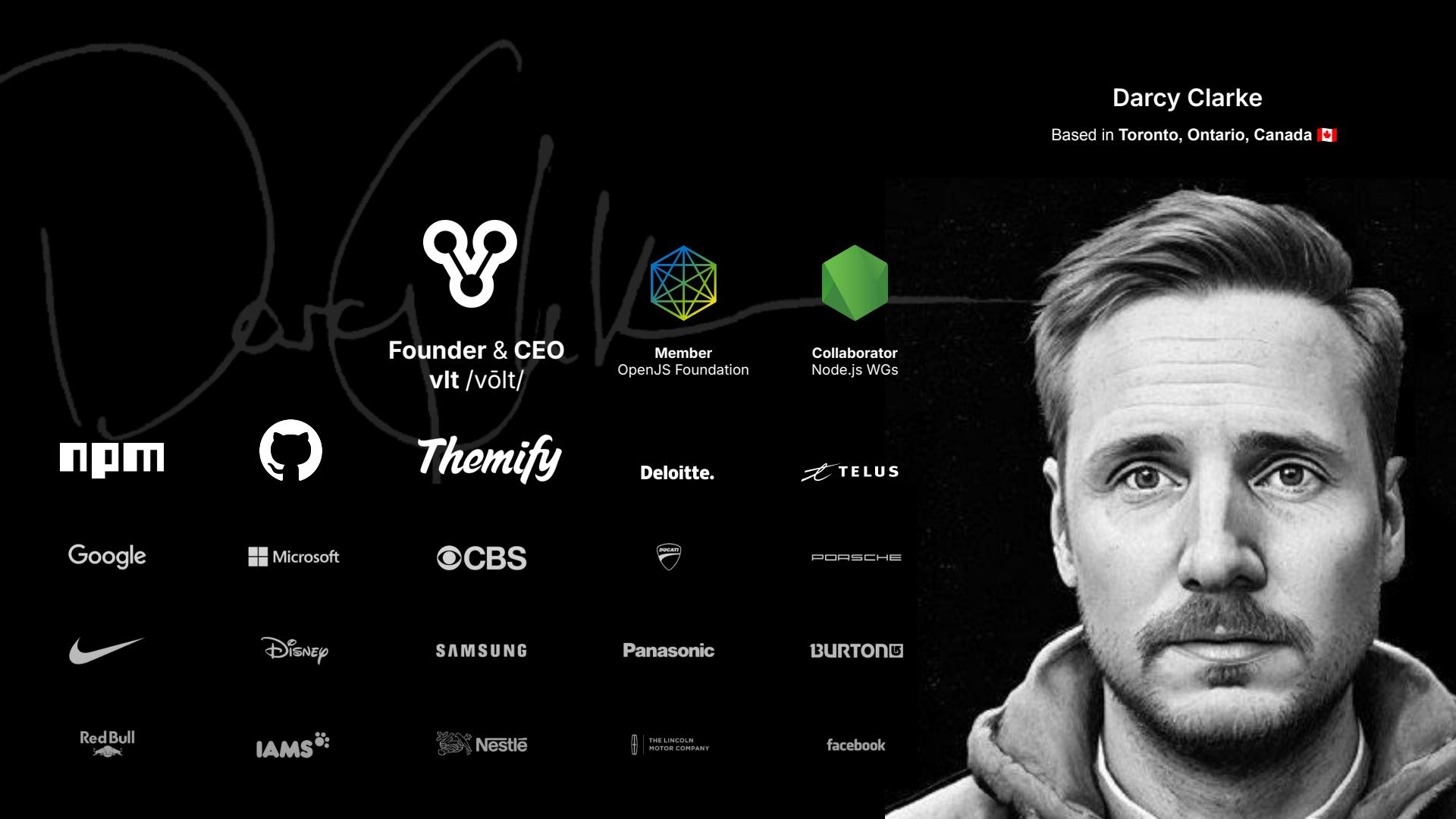


Securing the **JavaScript** Ecosystem with **Reproducibility**

 tinyurl.com/reproduce-2025





Darcy Clarke

Based in Toronto, Ontario, Canada 



Founder & CEO
vlt /vōlt/



Member
OpenJS Foundation



Collaborator
Node.js WGs



Themify

Deloitte.

TELUS

Google

Microsoft

CBS



PORSCHE



Disney

SAMSUNG

Panasonic

BURTON

RedBull

IAMS

Nestlé

THE LINCOLN
MOTOR COMPANY

facebook



 vlt /vōlt/

vlt.sh

The Team



Darcy Clarke

CEO & Founder

Toronto



Isaac Schlueter

Principal Software Engineer

Oakland



Ruy Adorno

Staff Software Engineer

Montreal



Luke Karrys

Staff Software Engineer

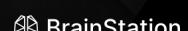
Phoenix



Jason Korol

Design Engineer

Toronto



Jules Pichette

Business & Finance

San Francisco





3+ billion downloads /mo
~2% of all registry traffic

ⓘ ex. **semver**, **tar**, **which**, **ini**, **ssri**, **write-file-atomic**,
hosted-git-info, **make-fetch-happen** & more...

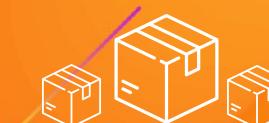
Staff Engineer Manager
July 2019 - December 2022 (~3.5yrs)



Team
Management



Product
Development



Package
Maintenance



Community
Engagement

Securing the **JavaScript** Ecosystem with **Reproducibility**

 tinyurl.com/reproduce-2025

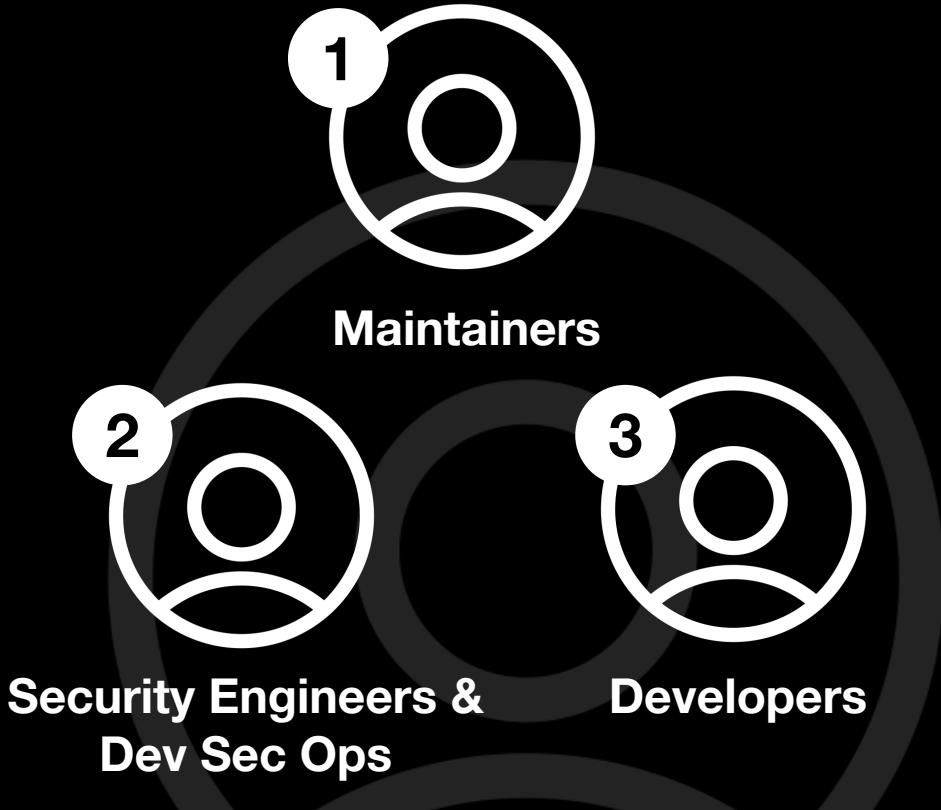


Please give feedback!

Overview

- **Trust**
 - **State of** Our Ecosystem
- **Provenance**
 - **Origins** of Tooling
- **Reproducibility**
 - **New** Tooling

Who Are you?



Why, What & Who we Trust?

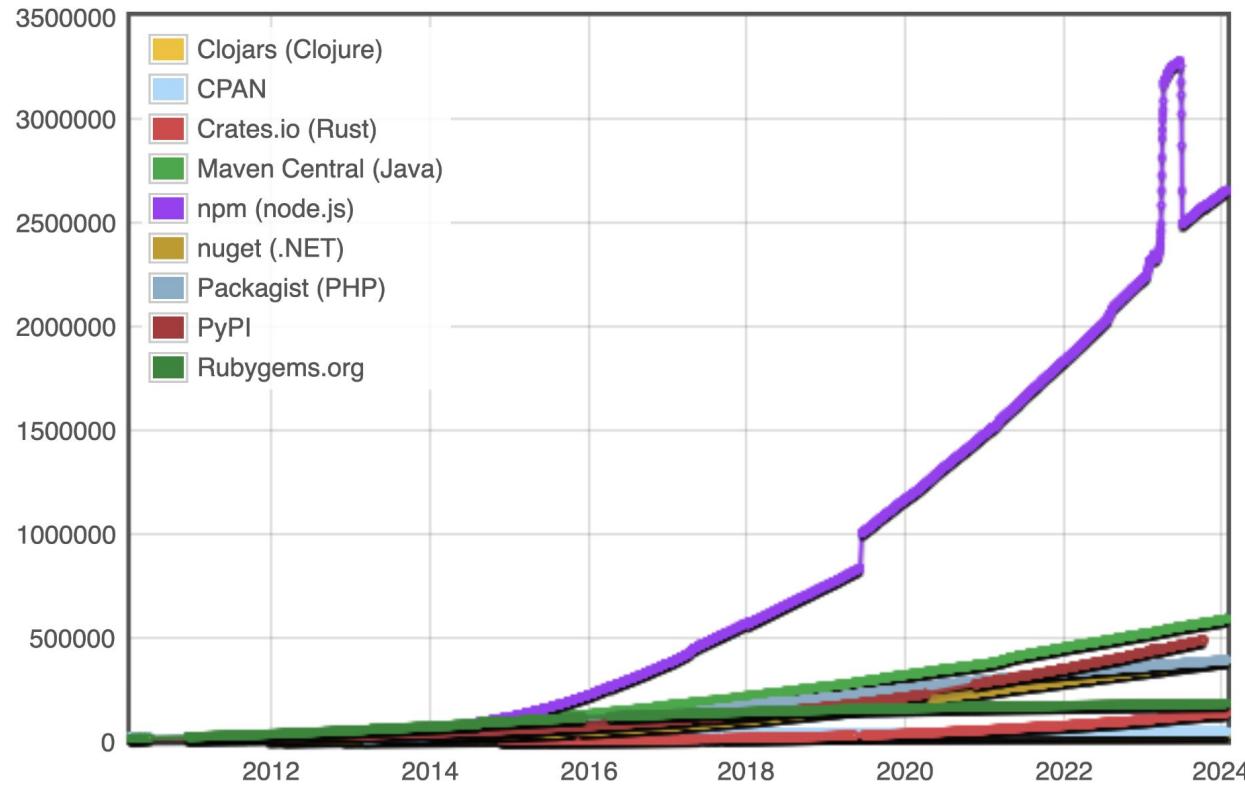


Who Installs Dependencies?

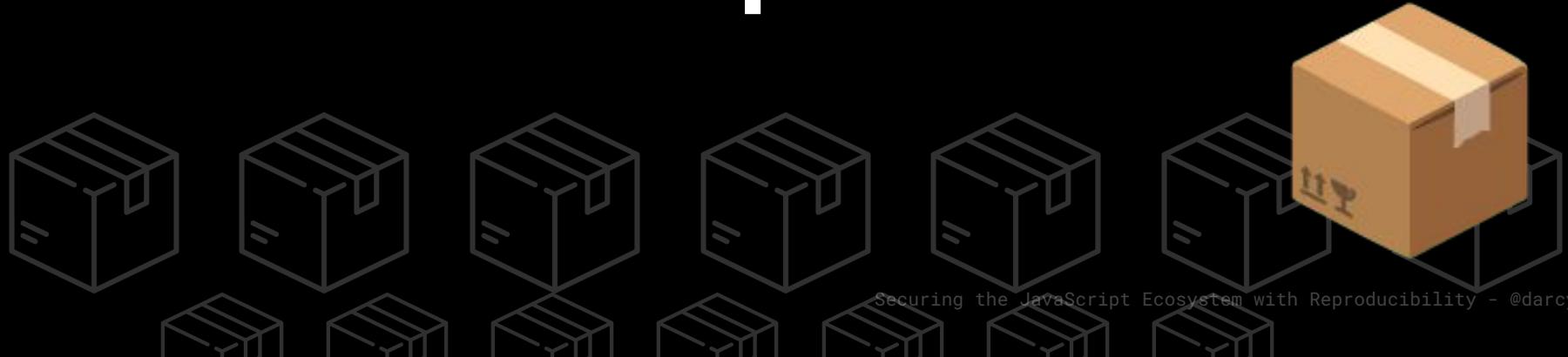
JS + **NPM** >



Module Counts



JavaScript **projects** have a lot of Dependencies

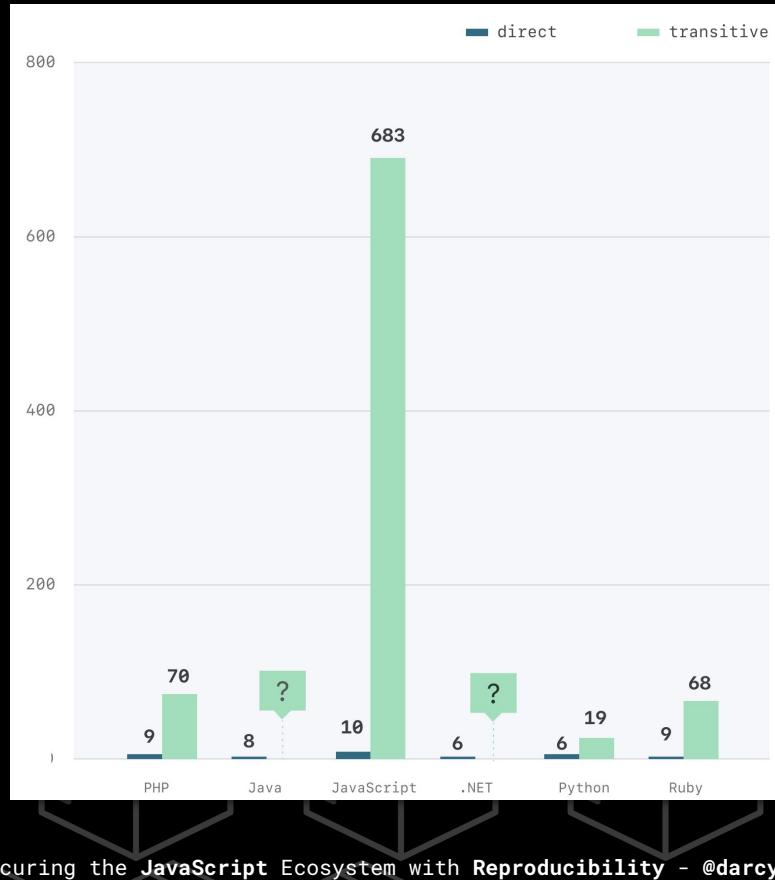
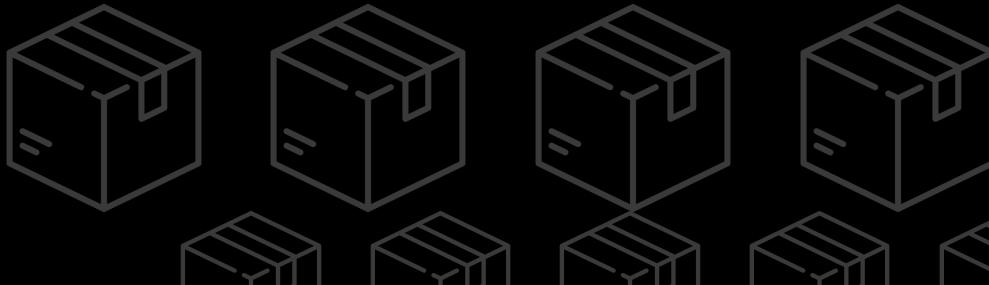


The average project has...

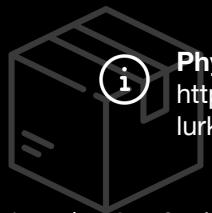
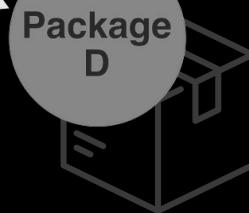
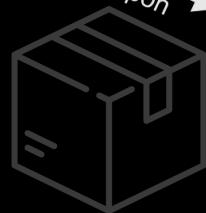
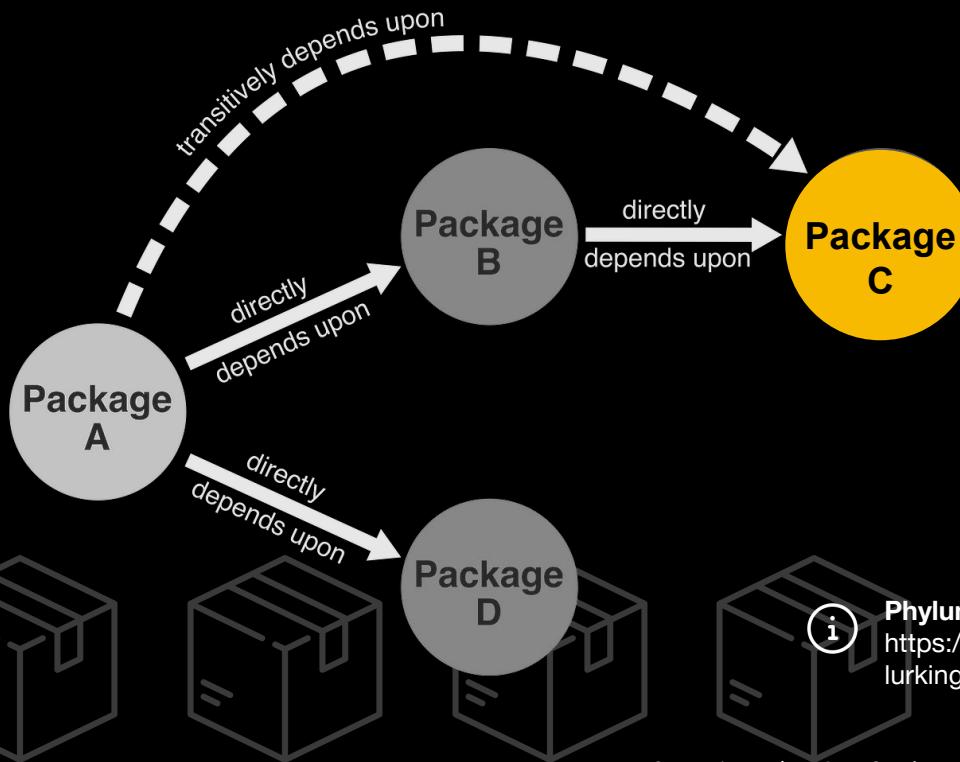
~683 transitive
dependencies

ⓘ GitHub's State of the Octoverse:

<https://octoverse.github.com/2021> & <https://octoverse.github.com/2020>



Transitive Dependencies



Phylum Blog Post
<https://blog.phylum.io/hidden-dependencies-lurking-in-the-software-dependency-network>

“Dependency Hell”

Securing t

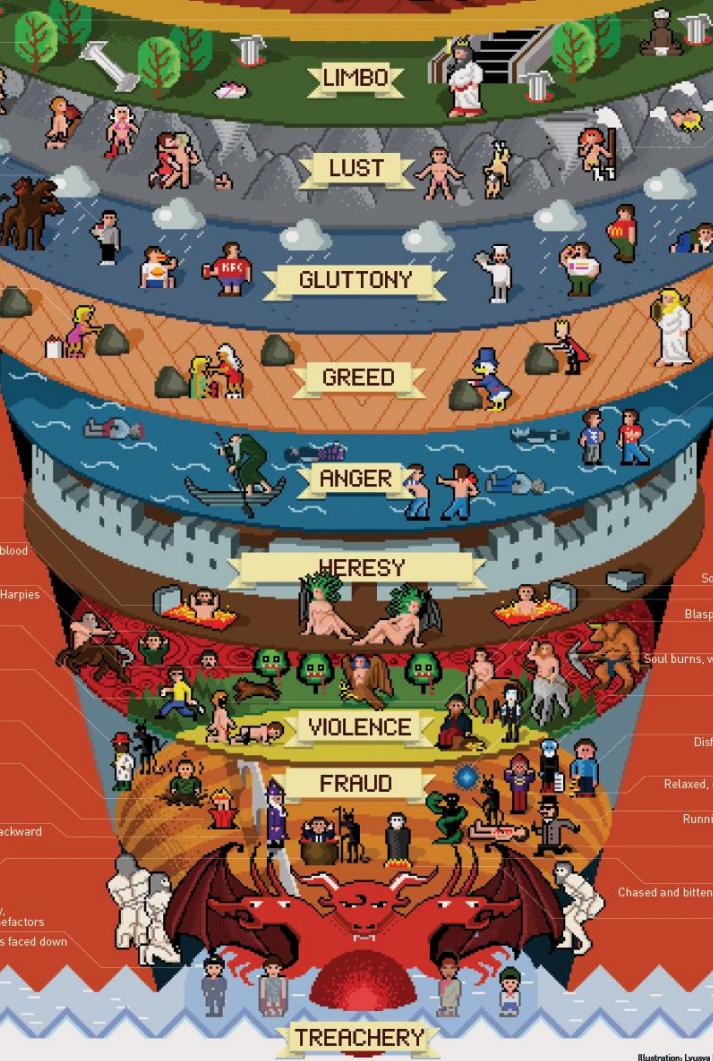


Illustration: Lyuya

The 9 circles of **Dependency Hell**

1. **Limbo - No** dependencies
2. **Lust - Any** dependency
3. **Gluttony - Every** dependency
4. **Greed - Never** contribute
5. **Anger - Protestware**
6. **Heresy - Don't follow spec**
7. **Violence - Malware**
8. **Fraud - Gifts**
9. **Treachery - Rogues**



The 9 circles of Dependency Hell

"Circles of *Incontinence*"
ie. a "lack of self-control"

- 1. Limbo - No dependencies
- 2. Lust - Consuming **any** dependency
- 3. Gluttony - **Everything** is a dependency
- 4. Greed - Consumption w/ **little-to-no** contribution
- 5. Anger - Make **demands** of maintainers or consumers
- 6. Heresy - Don't follow **standards / specs / semver**
- 7. Violence - **Malware / Protestware**
- 8. Fraud - **Grifts / Social Engineering**
- 9. Treachery - **Rogue Actors / Trojans**

Supply-Chain Security **Nightmares** 🤬

Securing t

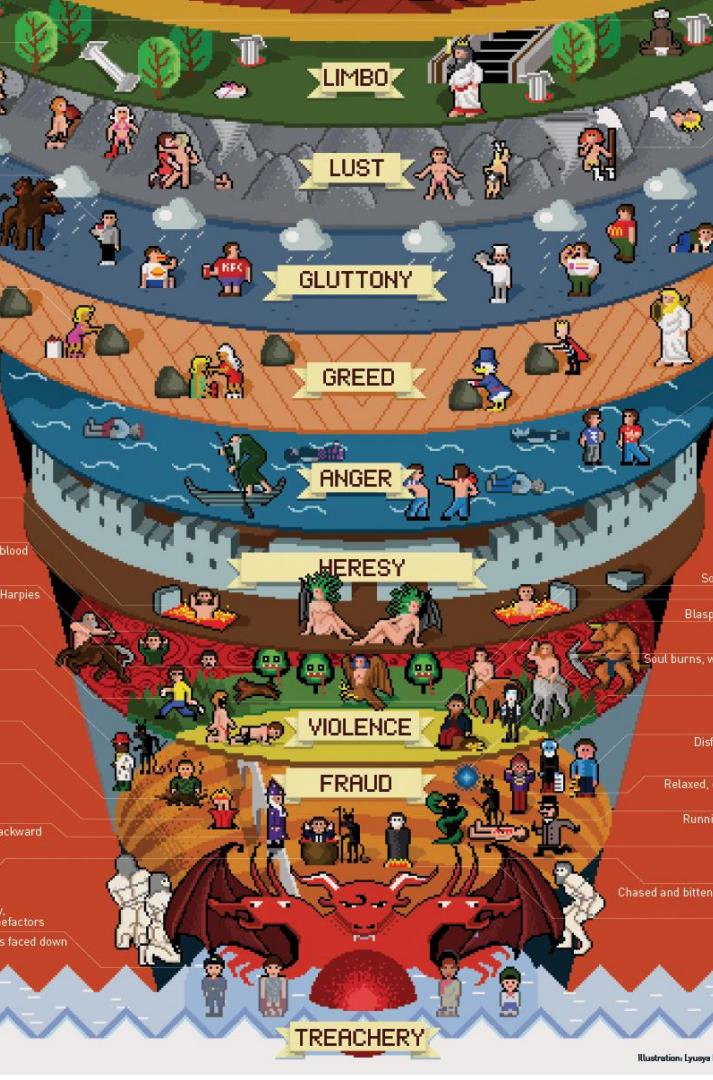
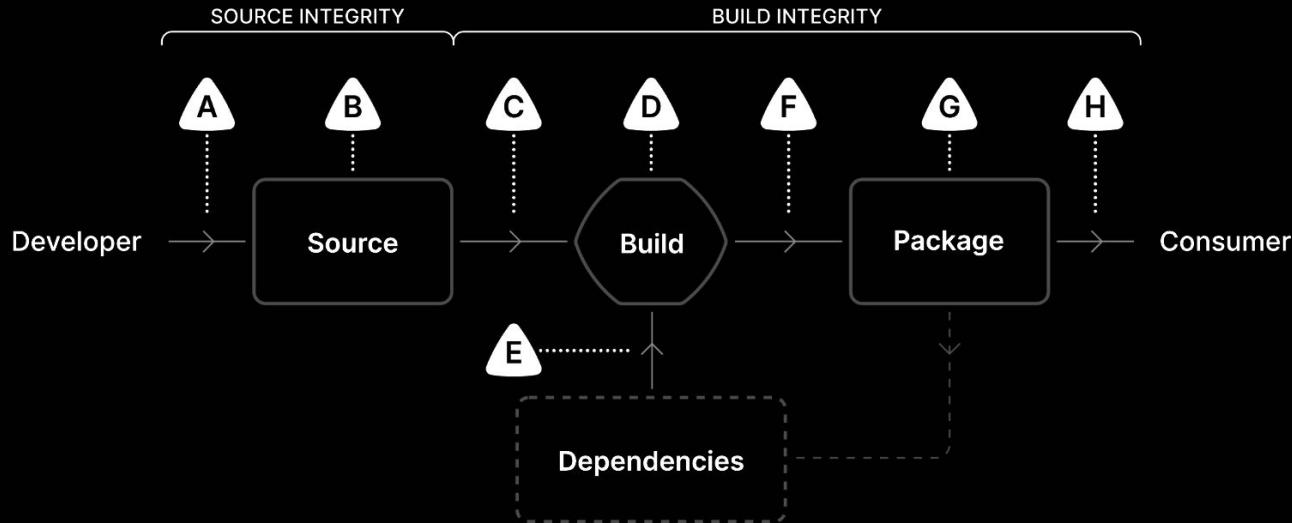


Illustration: Lyuya

Software Supply Chain Model



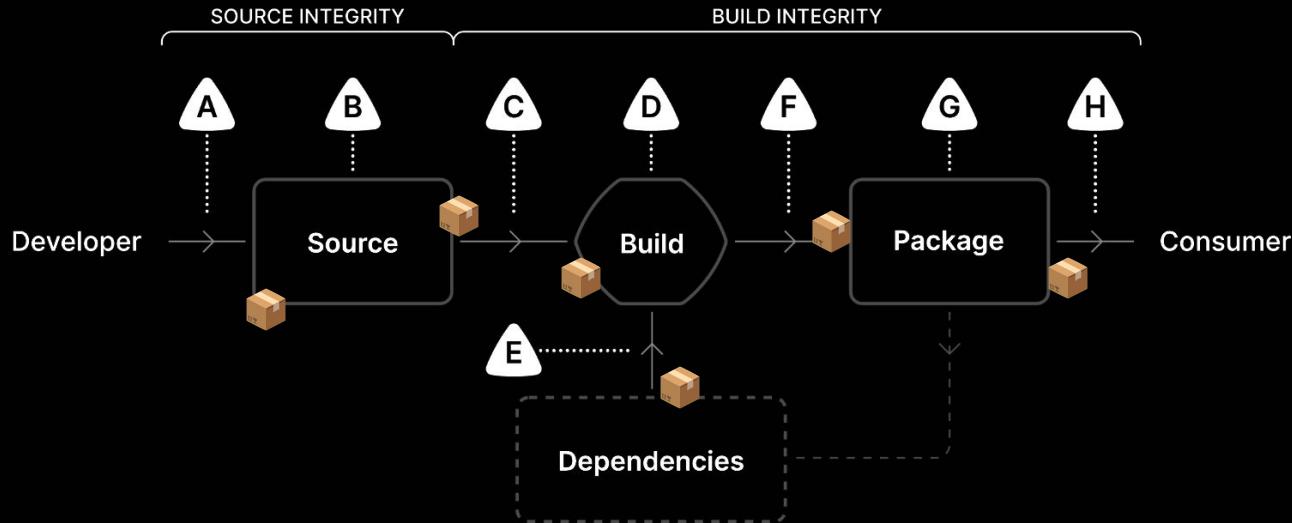
A Submit unauthorized change
B Compromise source repo

C Build from modified source
D Compromise build process

F Upload modified package
G Compromise package repo
H Use compromised package

Supply-chain Levels for Software Artifacts
<https://slsa.dev/>

Software Supply Chain Model



A Submit unauthorized change
B Compromise source repo

C Build from modified source
D Compromise build process
E Use compromised dependency

F Upload modified package
G Compromise package repo
H Use compromised package



Supply-chain Levels for Software Artifacts
<https://slsa.dev/>

JS

What is Provenance?



What is

Provenance?



- **Cryptographic Link Between Source and Artifact**
Provenance is a digitally signed proof that claims a package was built from a specific source repository and build process.
- **Flexible Metadata Attestations**
Provenance allows any arbitrary metadata (e.g., build environment, dependencies, timestamps) to be signed and associated with an artifact, making it highly customizable.
- **Enhanced Software Supply Chain Observability**
By tracing how software was built, provenance provides insights into who built it, where it was built, and how it was built.
- **Trust via Signing, Not Verification**
Provenance mechanisms typically rely on trusted build environments (e.g., GitHub Actions, Buildkite) to sign attestations, but they don't inherently verify the artifact's contents.
- **Focus on Identity, Not Reproducibility**
Provenance ensures that a specific entity (person, organization, or CI system) claims responsibility for a build, but it does not guarantee that the build can be independently reproduced.

Source? Artifact?

Why are there

Multiple Origins?

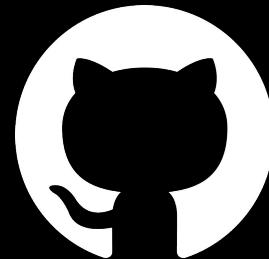


<https://npmjs.com>

“packages”

tar + semver

VS.



<https://github.com>

“repositories”

git



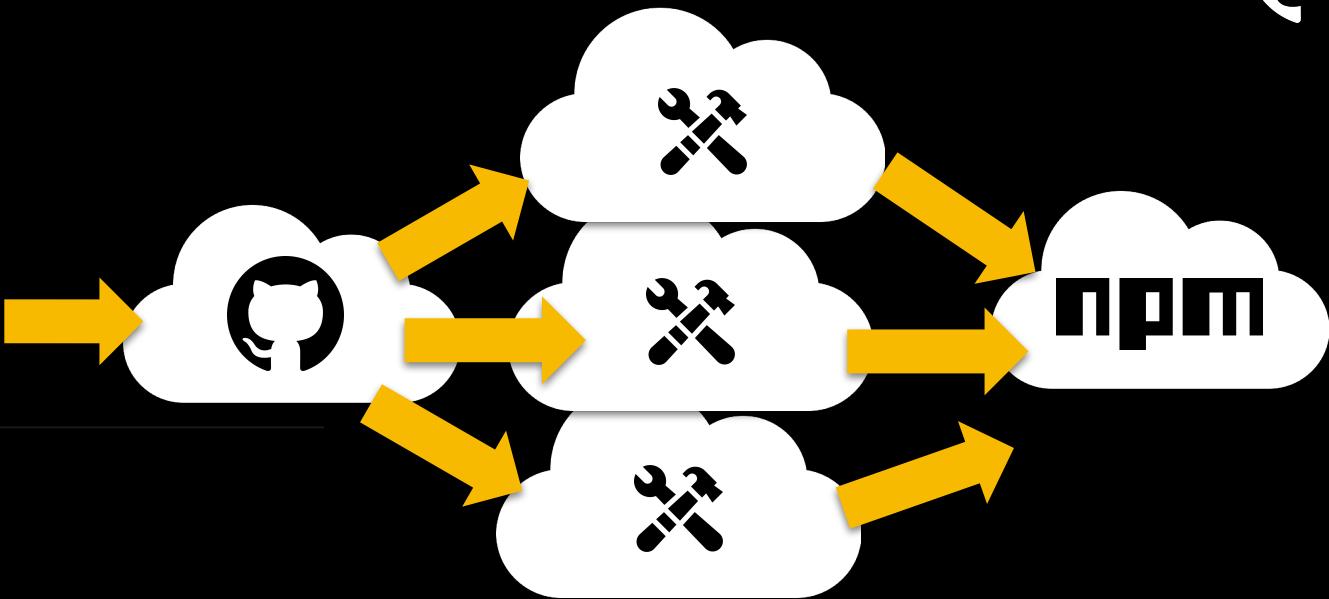


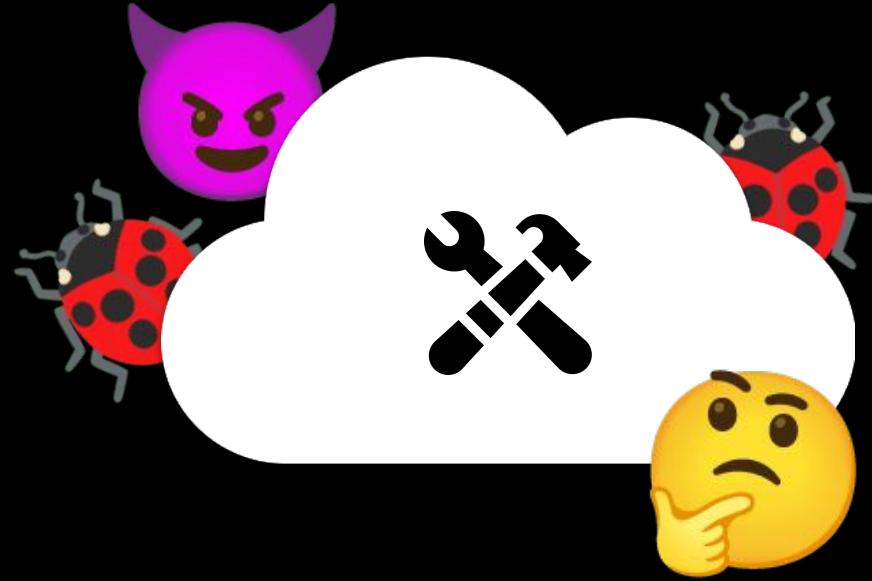
```
→ project  
cat .gitignore  
node_modules
```

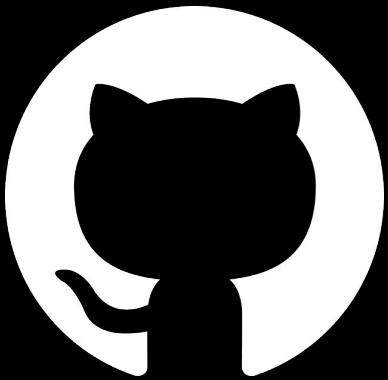
Build Envs End Up **Very Mutable**



```
→ project  
cat .gitignore  
node_modules
```







Securing the JavaScript Ecosystem with Reproducibility - @darcy



<https://npmjs.com/package/react>

vs.



<https://github.com/facebook/react>

Heart Pro Teams Pricing Documentation

npm Search packages

react DT

19.0.0 • Public • Published 3 months ago

Readme [Code](#) Beta

/react/

- cjs/ folder
- LICENSE text/plain
- README.md text/markdown
- compiler-runtime.js application/java
- index.js application/java
- jsx-dev-runtime.js application/java
- jsx-dev-runtime.react-server.js application/java
- jsx-runtime.js application/java

facebook / react Public

Code Issues 772 Pull requests 181 Actions Projects Wiki Security Insights

main 392 Branches 145 Tags Go to file Code

poteto [forgive][ez] Ignore test file (#32477) ebc22ef · 2 hours ago 20,016 Commits

.codesandbox Codesandbox: upgrade to Node.js 18 (#26330) 2 years ago

.github [forgive] Scaffold workspaces (#31917) 9 hours ago

compiler [forgive][ez] Ignore test file (#32477) 2 hours ago

fixtures Add Example of a SwipeRecognizer (#32422) 4 days ago

packages Move ViewTransitions helpers to ReactFiberCommitViewT... 9 hours ago

scripts Include component name in "async/await is not supporte..." 17 hours ago

.editorconfig Remove trim_trailing_whitespace from editorconfig (#314...) 3 months ago

.eslintignore Set and Prettier configs for React Compiler (#29073) 9 months ago

.eslintrc.js UseSwipeTransition when direction changes (#3...) 5 days ago

.git-blame-ignore-revs Create a prettier commit to .git-blame-ignore-revs 7 months ago

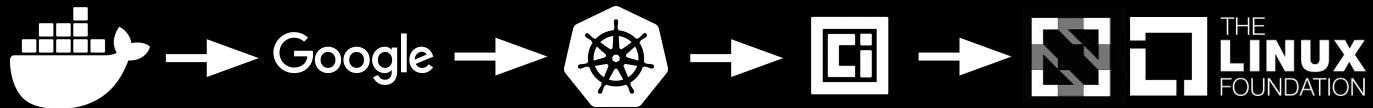
.gitattributes .gitattributes to ensure LF line endings when we should 12 years ago



What is **Sigstore?**



An Origin Story



Attack (March 2020)



Microsoft

vmware®

>200 Companies Affected (2 days later)



Biden Executive Order
#14028 (May 2021)



Raises \$10 Million
(October 2021)



OpenSSF



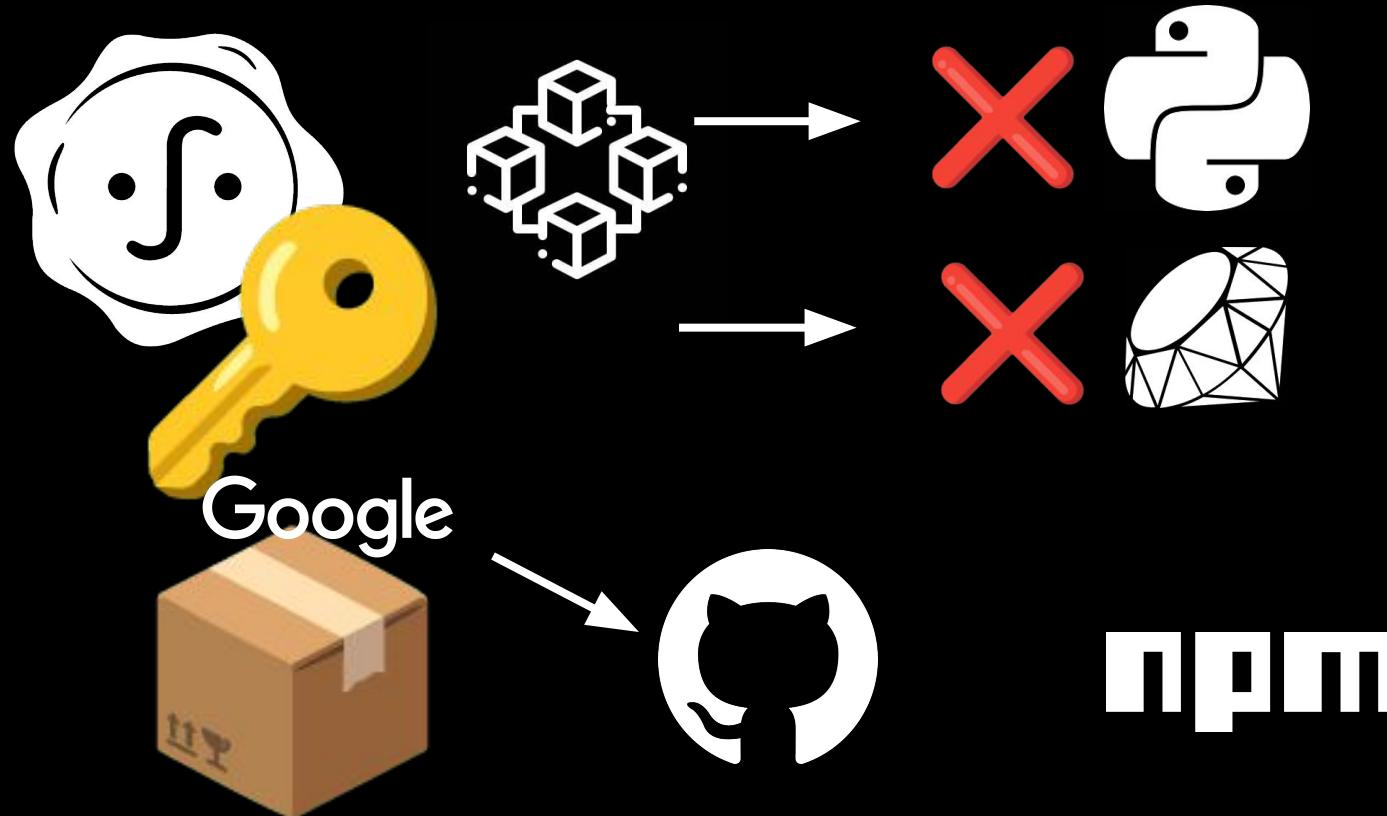
SLSA

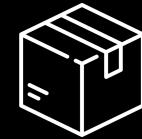
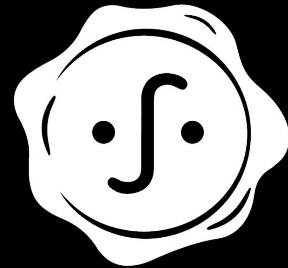
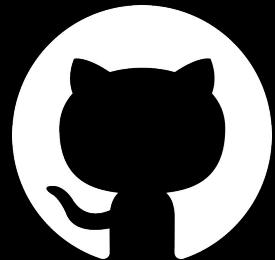


Chainguard

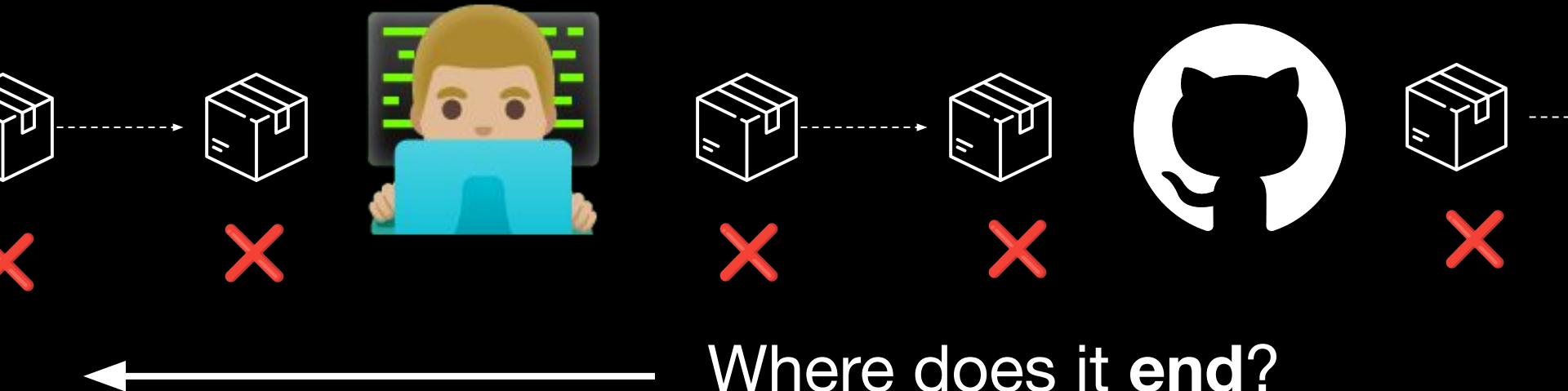


Sigstore



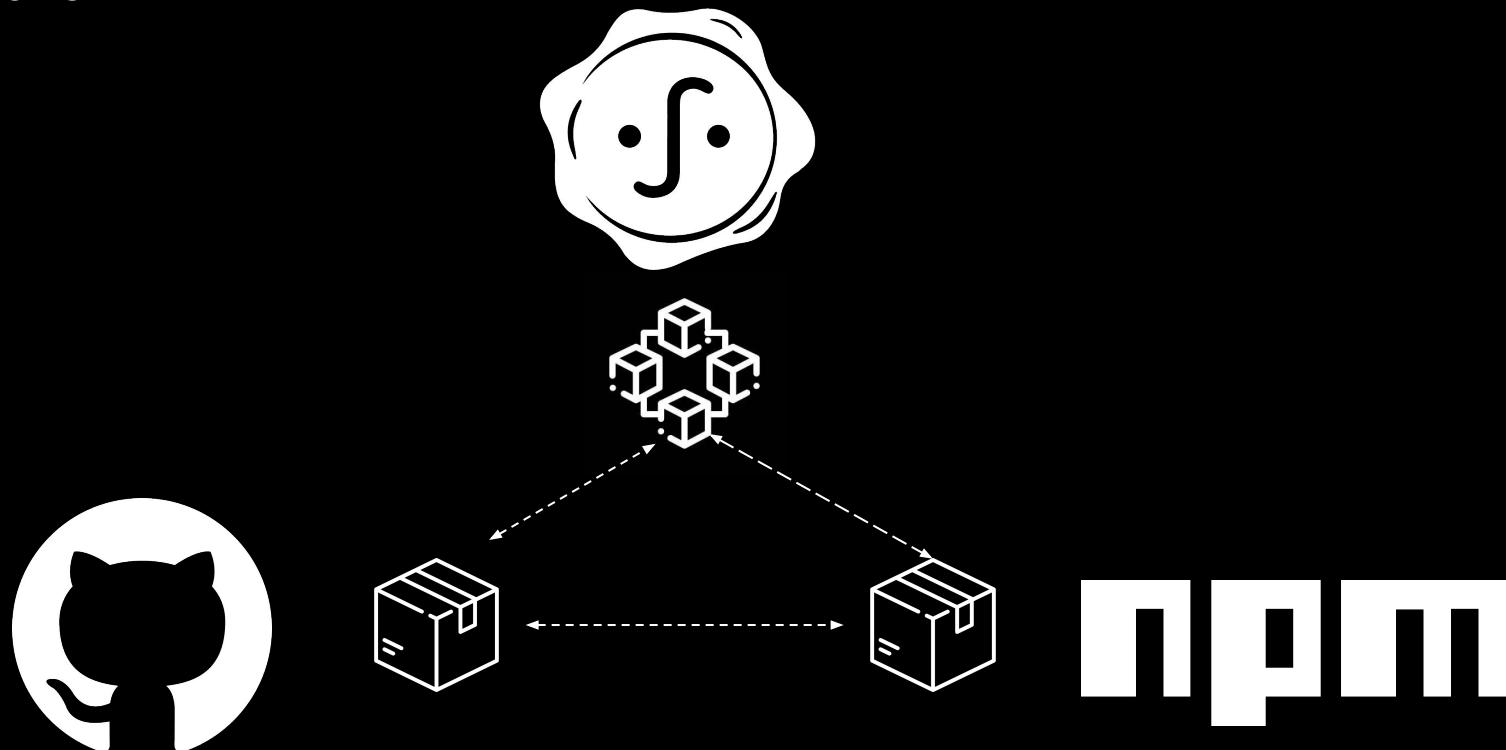


npm

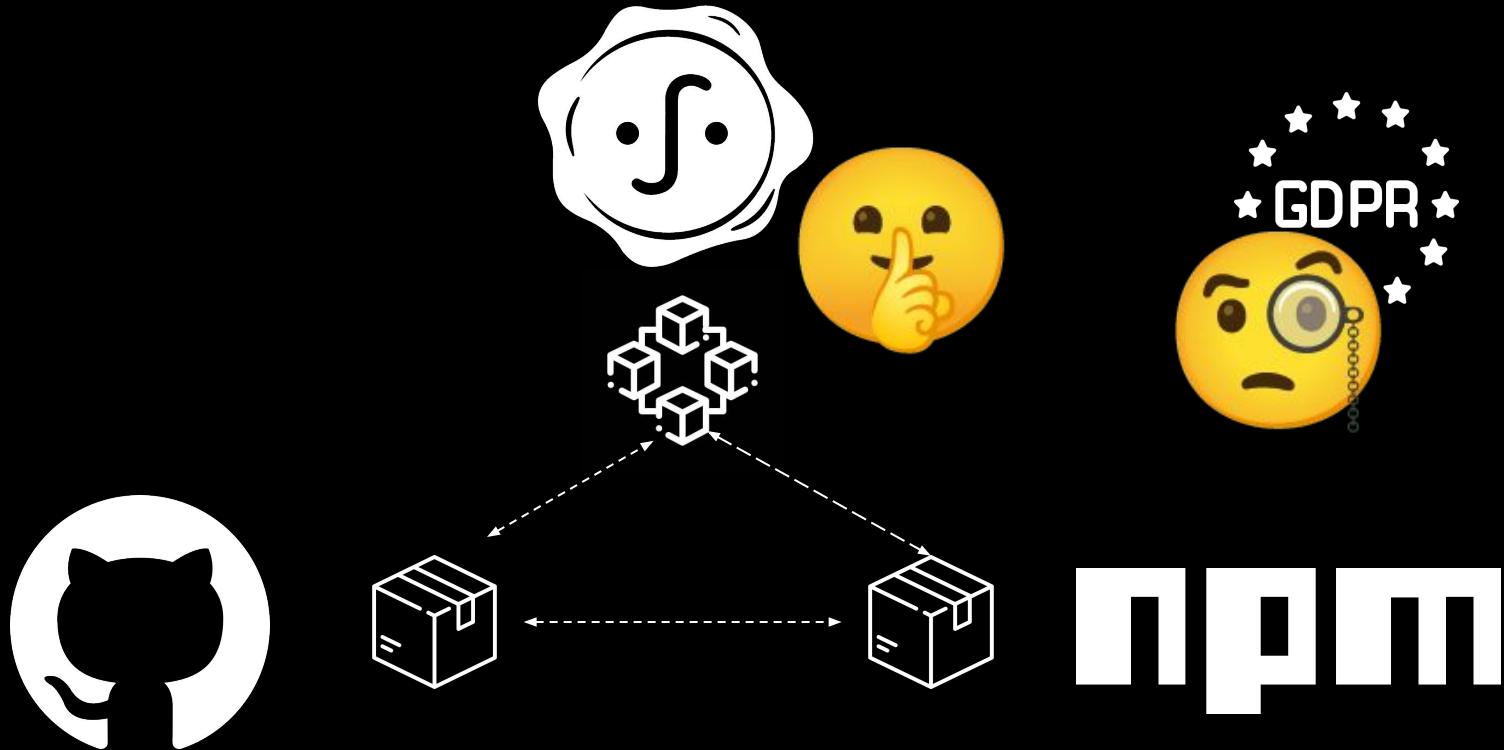


Where does it **end**?

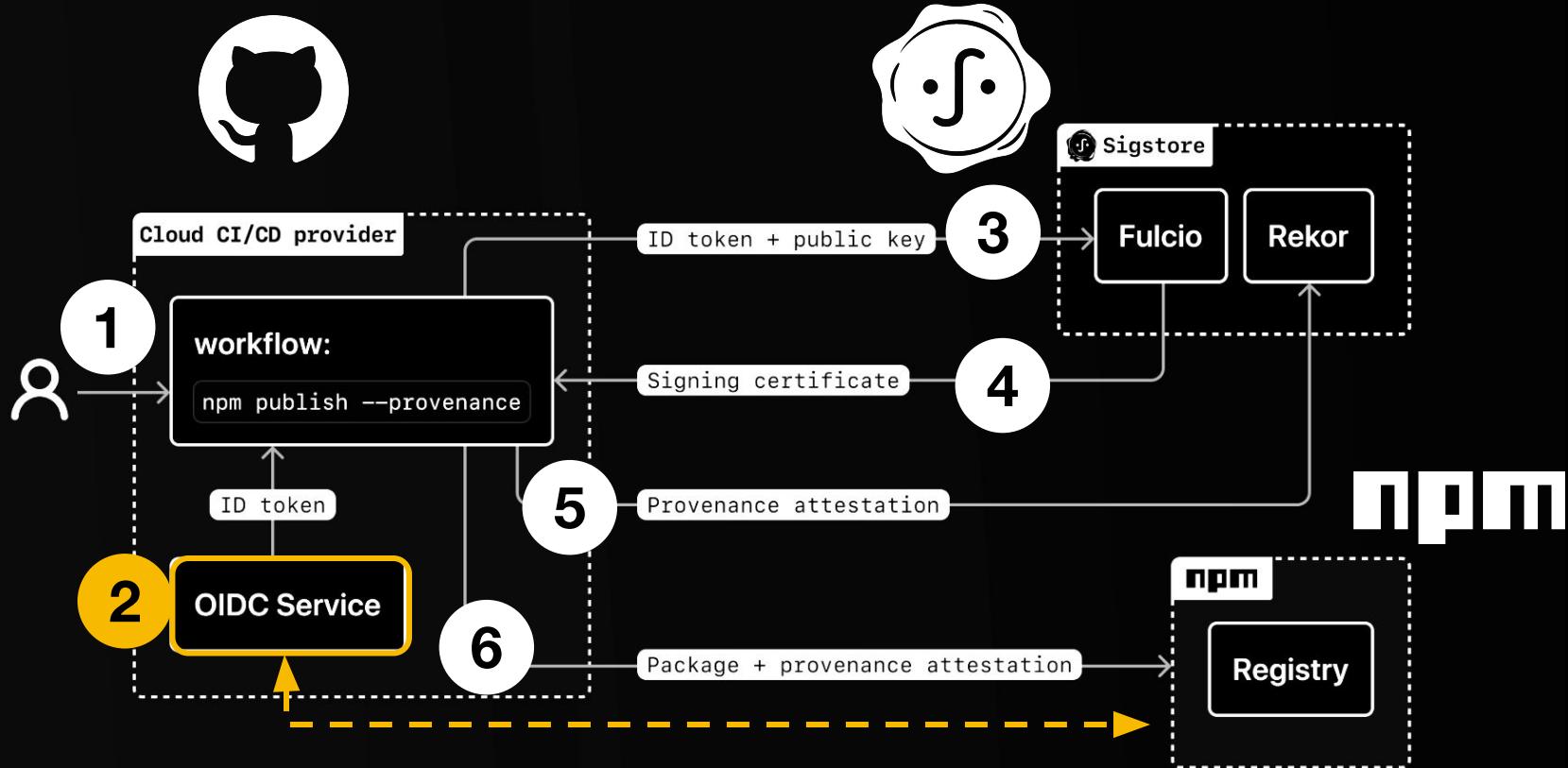
How it works...

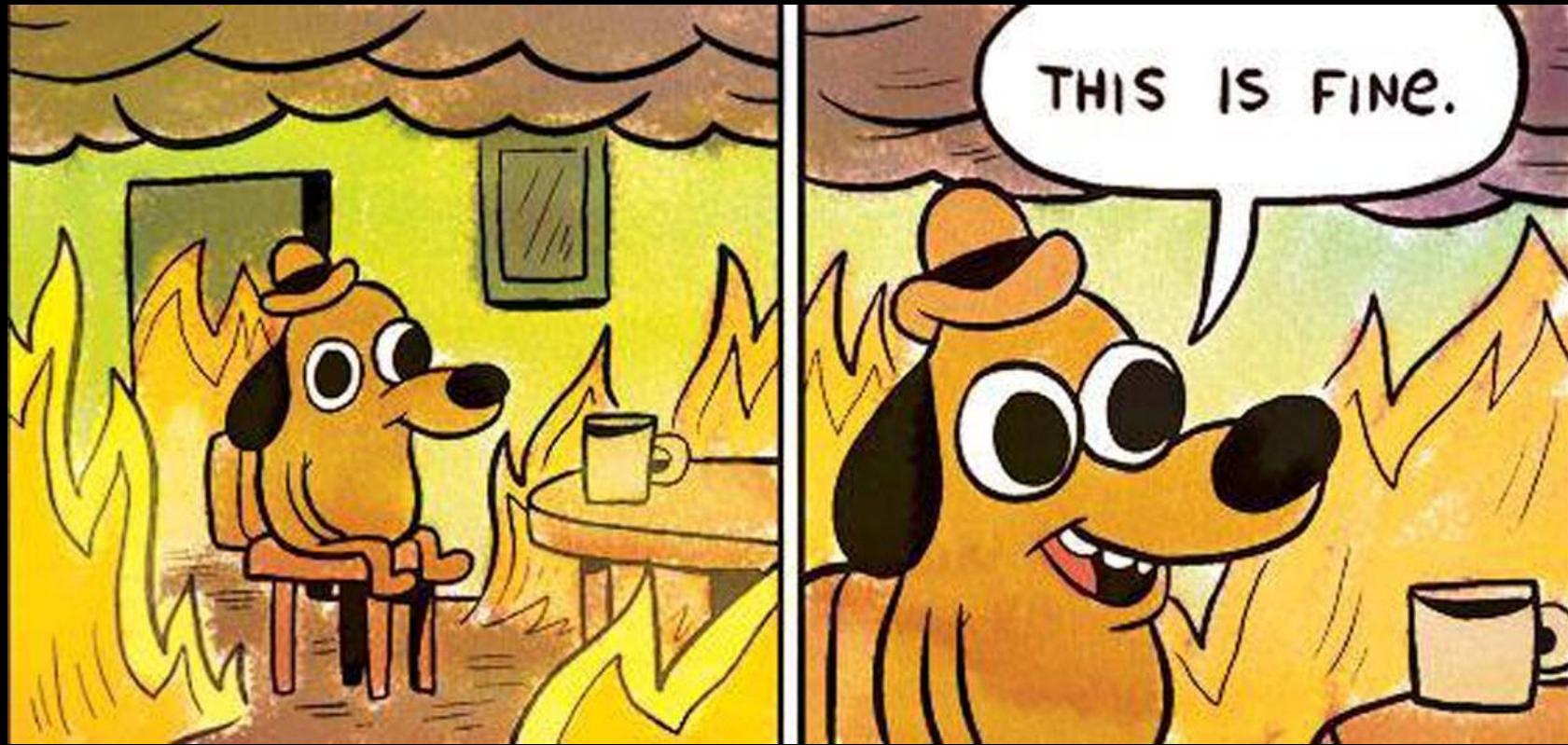


Compliant?

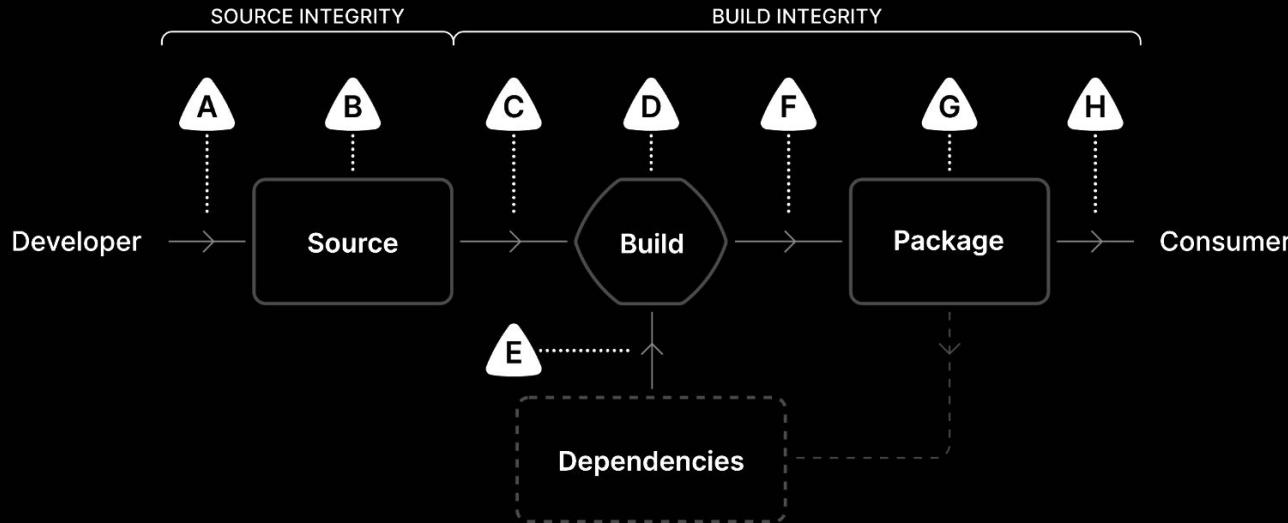


The Architecture





SLSA - Software Supply Chain Model



A Submit unauthorized change

B Compromise source repo

C Build from modified source

D Compromise build process

E Use compromised dependency

F Upload modified package

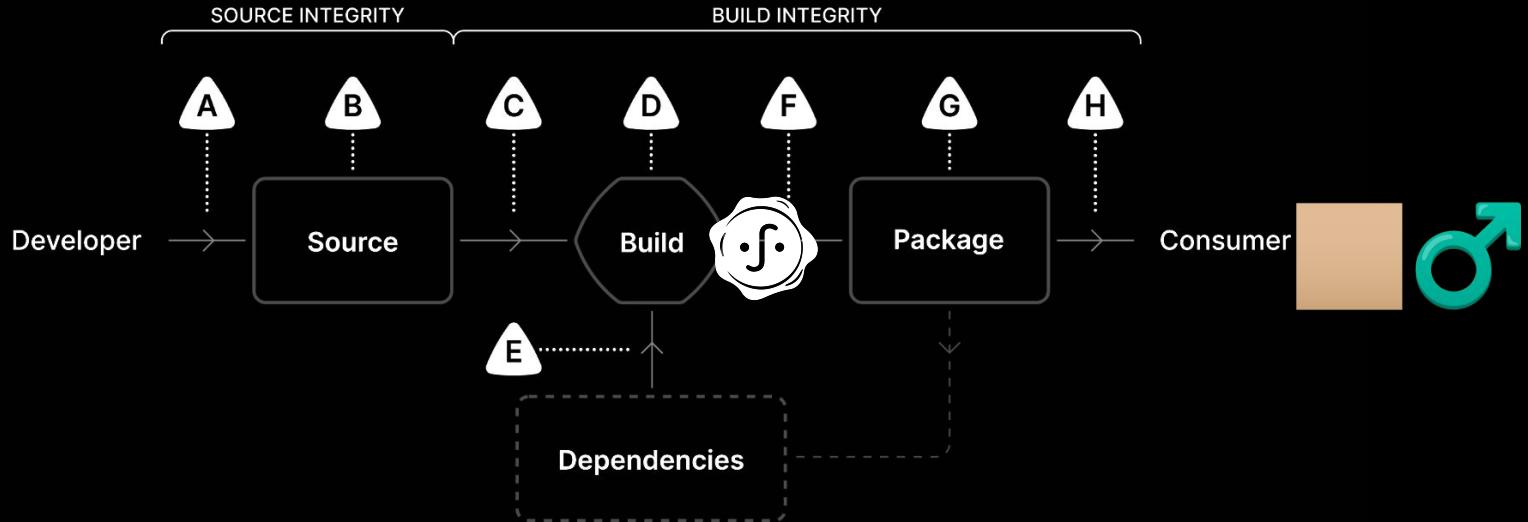
G Compromise package repo

H Use compromised package



Supply-chain Levels for Software Artifacts
<https://slsa.dev/>

SLSA - Software Supply Chain Model



A Submit unauthorized change
B Compromise source repo

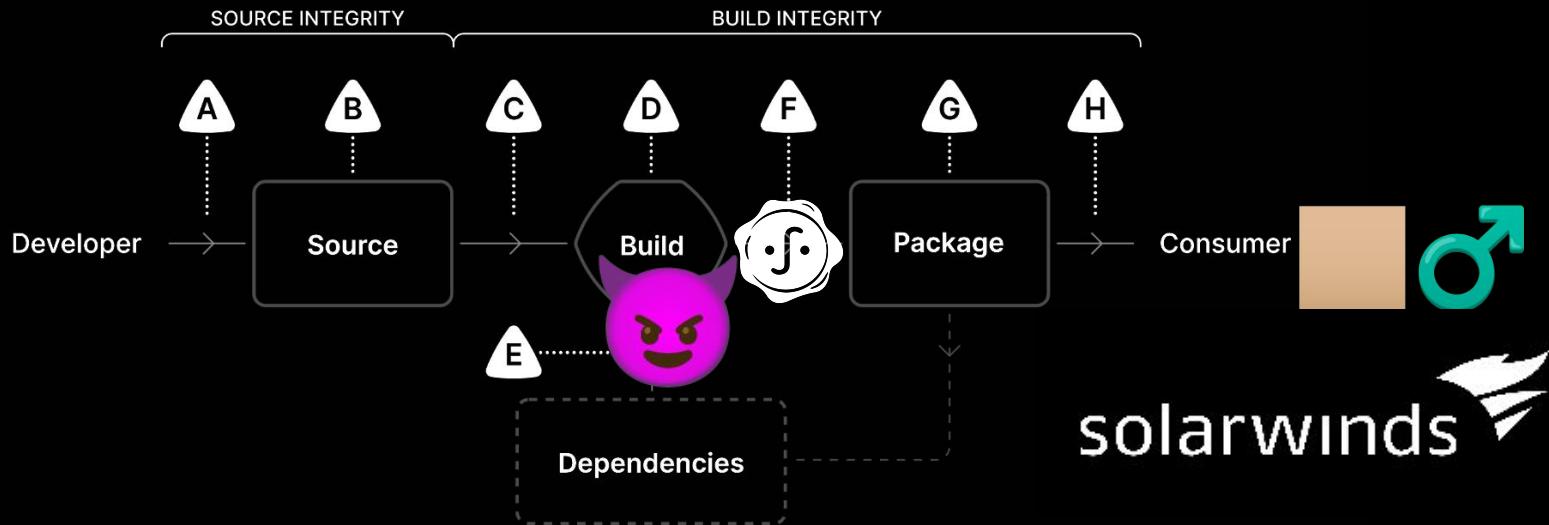
C Build from modified source
D Compromise build process
E Use compromised dependency

F Upload modified package
G Compromise package repo
H Use compromised package



Supply-chain Levels for Software Artifacts
<https://slsa.dev/>

SLSA - Software Supply Chain Model



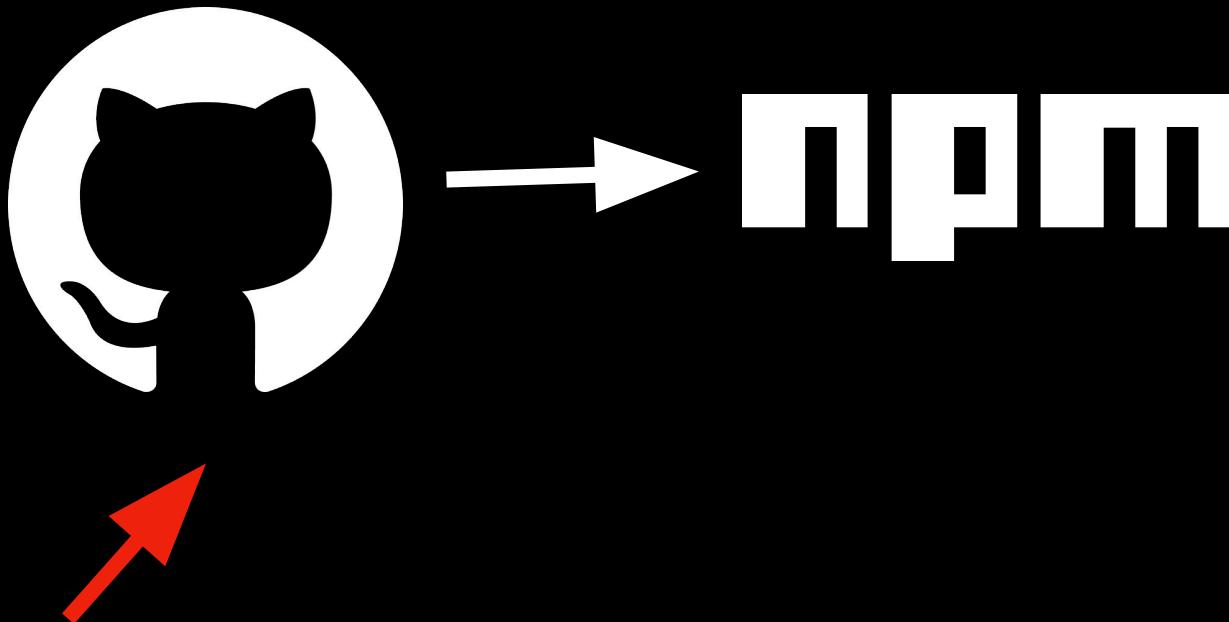
A Submit unauthorized change
B Compromise source repo

C Build from modified source
D Compromise build process
E Use compromised dependency

F Upload modified package
G Compromise package repo
H Use compromised package



Supply-chain Levels for Software Artifacts
<https://slsa.dev/>



Other

Provenance Issues



Forward-facing with **no way to backport**

-  **Miners must change build processes**



Must use cloud envs

- **Ephemeral** build workflows
 - Can be **deleted at anytime**



Ephemeral build environments

- Can be **deleted at anytime**
- Are **deleted after 90 days**

Pro Teams Pricing Documentation

Provenance

Built and signed on
 GitHub Actions
[View build summary](#)

Source Commit github.com/vercel/next.js@3c01e3a
Build File .github/workflows/build_and_deploy.yml
Public Ledger [Transparency log entry](#)

next 
14.2.16 • Public • Published

 [Readme](#)   102,761 Dependents  3,000 VERSIONS


Next.js

MADE BY VERCEL NPM V15.1.7 LICENSE MIT JOIN THE COMMUNITY

Getting Started

Used by some of the world's largest companies, Next.js enables you to create full-stack web applications by extending the latest React features, and integrating powerful Rust-based JavaScript tooling for the fastest builds.

- Visit our [Learn Next.js](#) course to get started with Next.js.
- Visit the [Next.js Showcase](#) to see more sites built with Next.js.

Version **14.2.16** 

Unpacked Size **86.3 MB**

License **MIT**

Total Files **6363**

Issues **2581**

Pull Requests **661**

productivity - @darcy

Pro Teams Pricing Documentation

npm Search

next TS
14.2.16 • Public • Published

Provenance

Built and signed on
 GitHub Actions
[View build summary](#)



Source Commit github.com/vercel/next.js@3c01e3a
Build File .github/workflows/build_and_deploy.yml
Public Ledger [Transparency log entry](#)

 [Readme](#)  [Code](#)  [Dependencies](#)  102,761 Dependents  3,000 VERSIONS


Next.js

MADE BY VERCEL NPM V15.1.7 LICENSE MIT JOIN THE COMMUNITY

Getting Started

Used by some of the world's largest companies, Next.js enables you to create full-stack web applications by extending the latest React features, and integrating powerful Rust-based JavaScript tooling for the fastest builds.

- Visit our [Learn Next.js](#) course to get started with Next.js.
- Visit the [Next.js Showcase](#) to see more sites built with Next.js.

Version	14.2.16 	License	MIT
Unpacked Size	86.3 MB	Total Files	6363
Issues	2581	Pull Requests	661
Productivity	—	@darcy	

vercel / next.js

Code Issues 2.6k Pull requests 660 Discussions Actions

← build-and-deploy v14.2.16 #15100

Summary

Jobs

- build
- stable - x86_64-apple-darwin - nod...
- stable - aarch64-apple-darwin - no...
- stable - x86_64-pc-windows-msvc ...
- stable - i686-pc-windows-msvc - n...
- stable - aarch64-pc-windows-msvc ...
- stable - x86_64-unknown-linux-gnu...
- stable - x86_64-unknown-linux-mus...
- stable - aarch64-unknown-linux-gn...
- stable - aarch64-unknown-linux-mu...
- build-wasm (web)
- build-wasm (nodejs)
- Deploy examples

The logs for this run have expired and are no longer available.

~250k Downloads Last Week
Published ~4 months ago

✓ False Sense of Security 🔒



What is Reproducibility?



What is Reproducibility?



- **Deterministic Builds**

Running the same build process with the same inputs should always produce identical artifacts, regardless of time, location, or environment.



Verifiable Integrity – Anyone should be able to independently rebuild a package from its source code and confirm that it matches the published version, ensuring no tampering or hidden modifications.

- **Source-to-Artifact Transparency** – The package that developers install should be probably linked to the source code it claims to originate from, reducing the risk of backdoors and supply chain attacks.

- **Independence from Trust Assumptions** – Consumers shouldn't have to blindly trust a publisher, CI system, or cryptographic signature—they should be able to verify the package contents themselves



What

Provenance & Reproducibility Don't Do...

Tell you whether a package's
contents are of **good quality**



Steps to Reproduce

1. **Fetch**: package **metadata** from the “trusted” registry
2. **Setup**: published source information to...
 - a. clone repository into temp dir
 - b. change working directory to temp dir
 - c. checkout defined ref (`gitHead` or `HEAD`)
 - d. change working directory to defined `directory` or fallback to `..`
3. **Run**: a predefined strategy of build steps, ex. `npm:*`:
 - a. npm install --no-audit --no-fund --silent
 - b. npm pack --dry-run --json
4. **Validate**: integrity of the built package against the actual integrity



What did we Launch?



The screenshot shows a dark-themed blog post. At the top right is a navigation bar with links for 'Product', 'Docs', 'Blog', and 'Company'. A large, stylized white logo resembling a key or a lock is positioned above the main content area. The main title 'Is your package truly reproducible?' is displayed in a large, bold, white sans-serif font. Below the title is a smaller, lighter text 'vlt /vōlt/'. The date 'Tuesday, February 25, 2025' is visible. The main heading of the post is 'Reproducibility vs. Provenance: Trusting the JavaScript Supply Chain', written in a large, bold, white sans-serif font. Below the heading is a bio for the author, Darcy Clarke (@darcy), with a small profile picture. The bio text reads: 'The security and trustworthiness of the JavaScript package ecosystem has been under scrutiny for years. With growing concerns over software supply chain attacks, the industry has doubled down on provenance: tracking where packages come from, how they're built, and ensuring transparency. But provenance alone isn't enough.' A section titled 'Reproducibility vs. Provenance: Trusting the JavaScript Supply Chain' follows, with a sub-section 'Introduction' and a 'Tags' section containing 'reproducible', 'package', 'security', and 'provenance'. At the bottom, there's a footer with links to 'Previous Article', 'Read Next', 'GitHub', and 'RSS Feed'. The footer also includes the word 'arcy'.

Is your package truly reproducible?

vlt /vōlt/

Tuesday, February 25, 2025

Reproducibility vs. Provenance: Trusting the JavaScript Supply Chain

Darcy Clarke (@darcy)

The security and trustworthiness of the JavaScript package ecosystem has been under scrutiny for years. With growing concerns over software supply chain attacks, the industry has doubled down on provenance: tracking where packages come from, how they're built, and ensuring transparency. But provenance alone isn't enough.

Reproducibility vs. Provenance: Trusting the JavaScript Supply Chain

Introduction

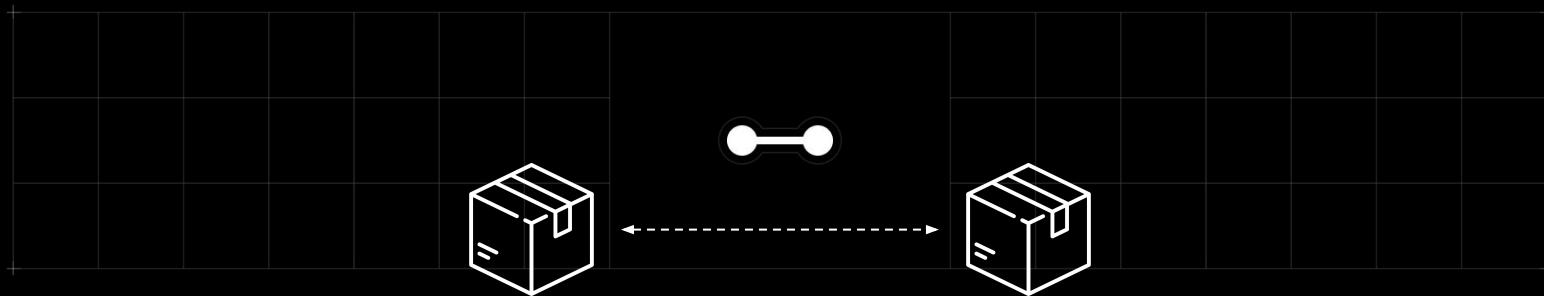
Tags: [reproducible](#), [package](#), [security](#), [provenance](#)

Previous Article Read Next GitHub RSS Feed arcy

Why People Were Excited About Provenance

Introducing **reproduce**

```
$ npx reproduce <pkg>
```



reproduce - Programmatic Usage

```
import reproduce from 'reproduce'

// Basic usage
const result = await reproduce('package-name')

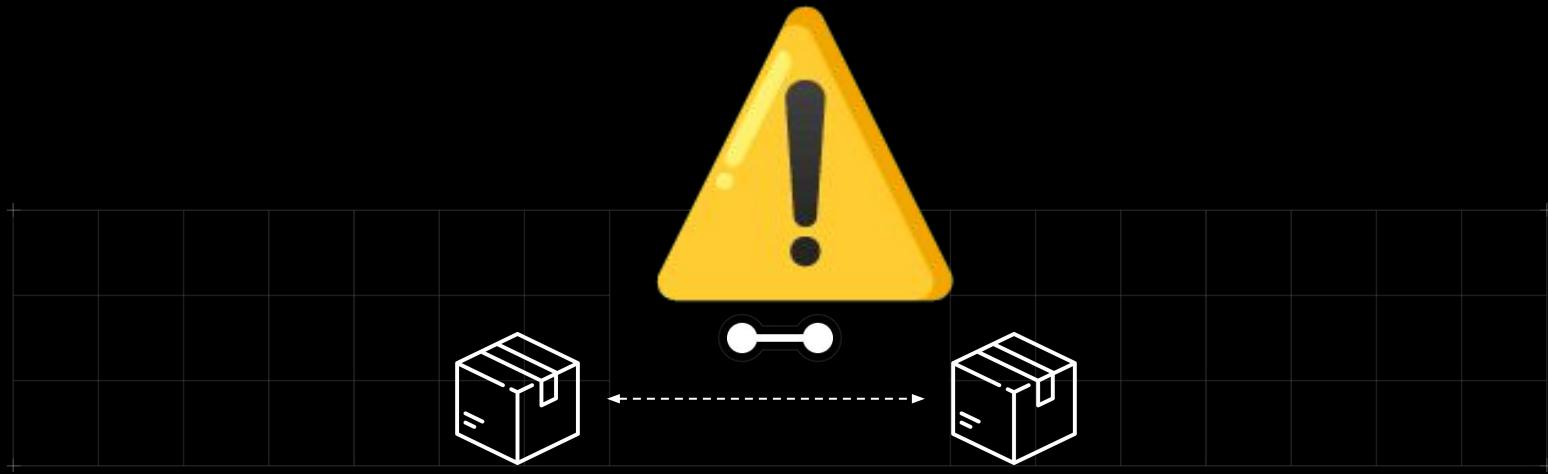
// With custom configuration
const result = await reproduce('package-name', {
  cache: {},
  cacheDir: './custom-cache'
  cacheFile: 'custom-cache.json'
})
```

reproduce - CLI

```
$ npx reproduce semver
# exit code 0 - reproducible

$ npx reproduce esbuild
# exit code 1 - not reproducible

$ npx reproduce semver --json
{
  "reproduceVersion": "0.0.1-pre.1",
  "timestamp": "2025-02-25T11:08:01.729Z",
  "os": "darwin",
  "arch": "arm64",
  "strategy": "npm:10.9.1",
  "reproduced": true,
  "package": {
    "spec": "semver",
    "location": "https://registry.npmjs.org/semver/-/semver-7.7.1.tgz",
    "integrity": "sha512-hlq8tAfn0m/61p4BVRcPzIGr6LKiMwo4VM6dGi6pt4qcRkmNzTcwq6eCEjEh+qXjkMDvPlOFFSGwQjoEa6gyMA=="
}
```



Only supports **1 strategy (npm)** today but more will be added in the future (feel free to make a PR).

What
Does the data look like?

Scanned “*High Impact*” Packages (**5,000**)

>1 Million Weekly Downloads or >500 dependants

5.78%

are **reproducible**

vs.

3.72%

have **provenance**

Key Takeaways

- **Reproducibility > Provenance**
 - Reproducibility is independently verifiable
 - Provenance relies on **more** trust, not **proof**
- **Neither** solves for **quality**
 - Look into static analysis/malware detection tools
(ex. Socket.dev)
- Try to reproduce your favorite npm package today:
\$ npx reproduce <pkg>

Next...

Modernizing JavaScript Supply Chain Security

Tomorrow @ 9:00am (same room)



The screenshot shows the Vit UI interface. On the left is a sidebar with a dashboard, queries, and a list of projects: `@vitpkg/registry`, `apollo-client-devtools`, `d3-pprof`, `gsap`, and `npm`. The main area is titled "Explore" and shows the results of the query `:root > [name="postcss"][version="8.4.49"]`. It displays the "Selected Item" for `postcss v8.4.49`, which has `6,729,473 Weekly Downloads`. The "Manifest" tab shows the package's manifest file:

```
{
  "name": "postcss",
  "version": "8.4.49",
  "description": "Tool for transforming styles with JS plugins",
  "engines": {
    "node": "^10 || ^12 || >14"
  },
  "exports": {
    ".": {
      "require": "./lib/postcss.js",
      "import": "./lib/postcss.mjs"
    },
    "/lib/at-rule": "./lib/at-rule.js",
    "/lib/comment": "./lib/comment.js",
    "/lib/container": "./lib/container.js",
    "/lib/css-syntax-error": "./lib/css-syntax-error.js",
    "/lib/declaration": "./lib/declaration.js",
    "/lib/fromJSON": "./lib/fromJSON.js",
    "/lib/input": "./lib/input.js",
    "/lib/lazy-result": "./lib/lazy-result.js",
    "/lib/no-work-result": "./lib/no-work-result.js",
    "/lib/list": "./lib/list.js",
    "/lib/map-generator": "./lib/map-generator.js",
    "/lib/node": "./lib/node.js",
    "/lib/parse": "./lib/parse.js",
    "/lib/parser": "./lib/parser.js"
}
```

The "Dependencies" section lists `nanoid v3.3.7`, `picocolors v1.1.1`, and `source-map-js v1.2.1`.

Thank you!

Bluesky:

@darcyclarke.me

Twitter / X:

@darcy

GitHub:

@darcyclarke

Website:

darcyclarke.me

Bluesky:

@vltpkg.sh

Twitter / X:

@vltpkg

GitHub:

@vltpkg

Website:

vl.sh



Please give feedback!

“ No man ever steps
in the **same river** ,
twice. ”



Heraclitus of Ephesus (Greek Philosopher)
<https://en.wikipedia.org/wiki/Heraclitus>

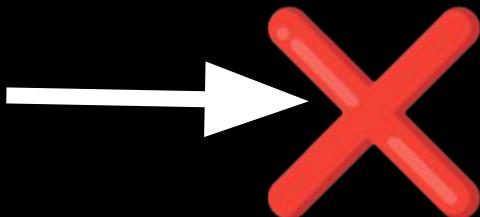


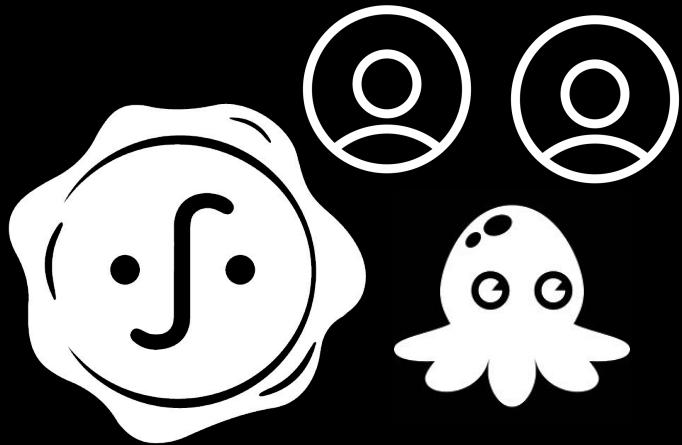


Google



Let's Encrypt





01/2021

10/2021



OPEN SOURCE SECURITY FOUNDATION

