SUPPLYCHAIN SECURITYCON @ OPEN SOURCE SUMMIT JAPAN

THE LINUX FOUNDATION

The Telemetry of Trust, Using Attestations to Secure Your SDLC w/ Open Source Tools

Jesse Sanford - Software Architect, Autodesk
Jagadish Ramidi - Security Software Engineer, Autodesk

Jesse Sanford

Software Architect, Autodesk

Jesse is a lifelong software engineer focused on site reliability and Infosec. Currently architecting the juncture of platform engineering and security/compliance for Autodesk's Developer Enablement team. He regularly contributes to open source and frequently speaks about his work utilizing it at Autodesk and in the community. When not in front of a computer, he is a backpacker, sailor and continuously delivering parent of two young daughters.

Jagadish Ramidi

Software Engineer, Autodesk

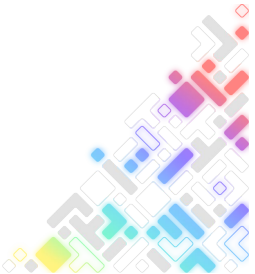
Works as a security software engineer at Autodesk focusing on software composition analysis and supply chain security.

Agenda

- Level set on S3C (Secure Software Supply Chain)
- S3C at Autodesk
- Telling the SDLC “story”
- Attestations!
- Demo
- How to get started and where we can go

Quick level set on SSCS

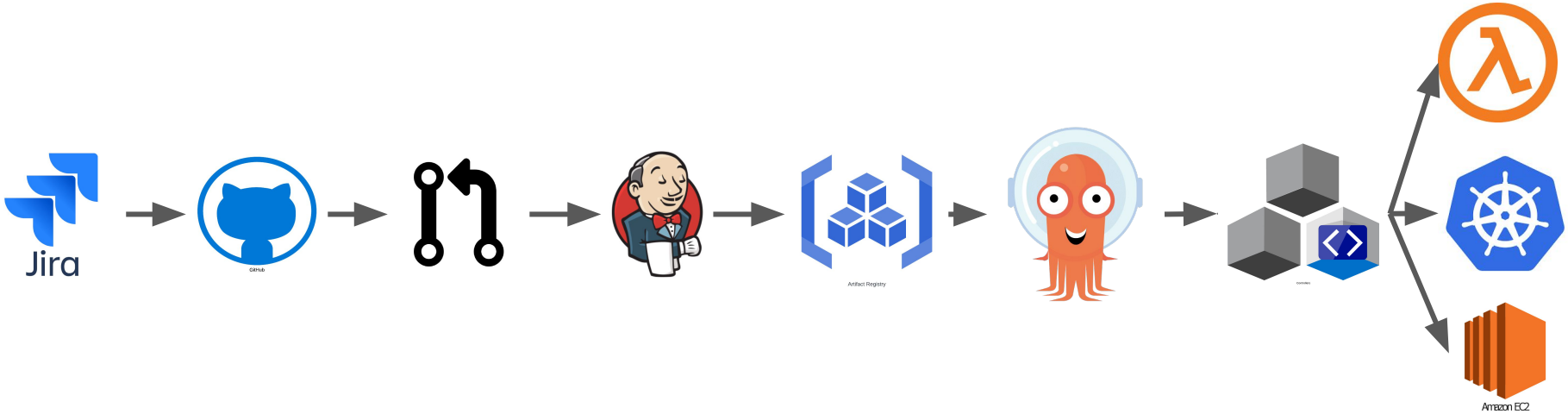
- Progress over past few years is good
  - Standards: (SBOM, SLSA 1.X, SSDF 1.X, IETF SCITT)
  - Public Good Services: Sigstore, Alpha-Omega project
  - Major SCM as a service provenance generation (Github, Gitlab)
  - Sigstore for package managers and major projects (K8s, NPM, Homebrew)
- But adoption in the enterprise is slow
  - Technology inertia (Legacy / brown field)
  - Compliance regime deltas

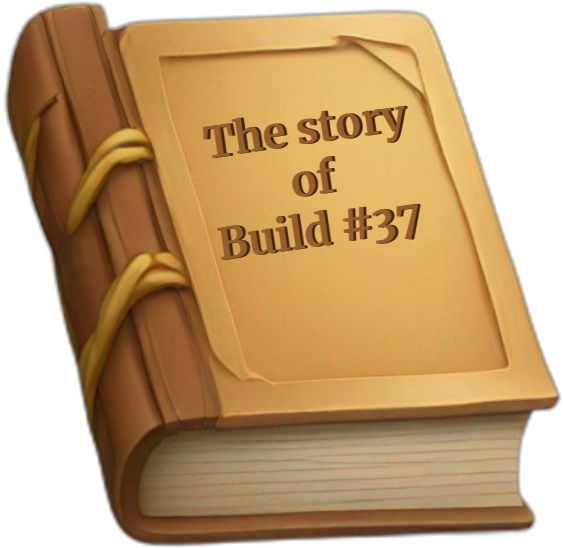


Autodesk

- Autodesk also has its own technology inertia
- Vigorous M&A
- How can we wrap all that diversity?
  - Need something adaptable
  - Can encompass the complete SDLC (from intent to runtime)

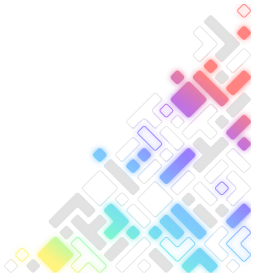
Autodesk SDLC overview



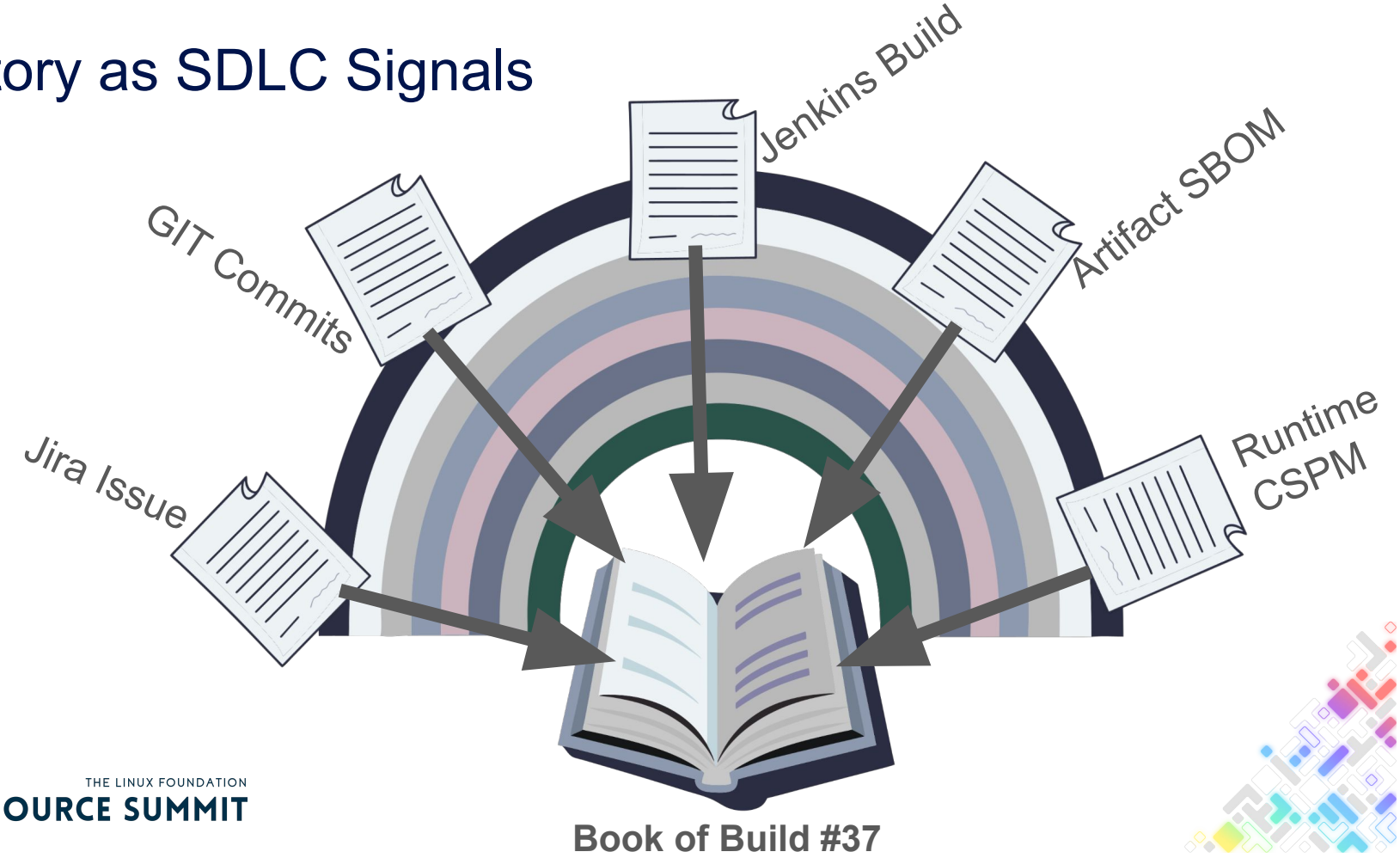

The story
of
Build #37

Telling the SDLC “story”

- For a particular build, let’s tell it’s story
- Every build is it’s own book
- Each phase of the SDLC is another “chapter”
- Each step in a phase like (CI) is another page
- Each page can have multiple “statements”

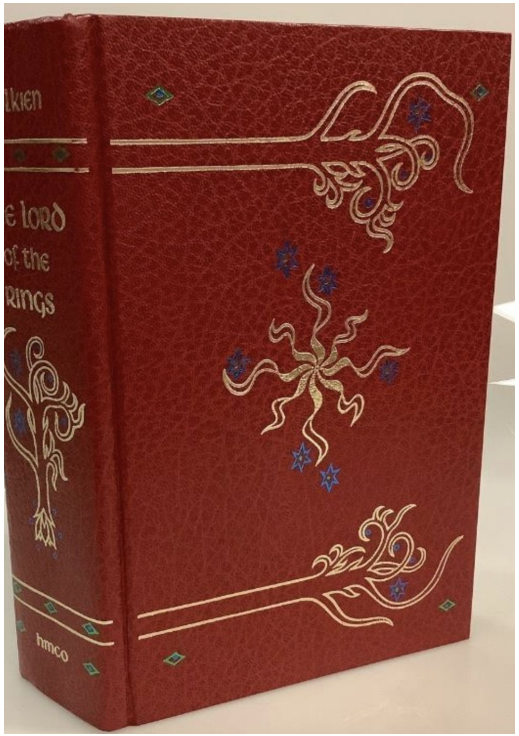


The Story as SDLC Signals

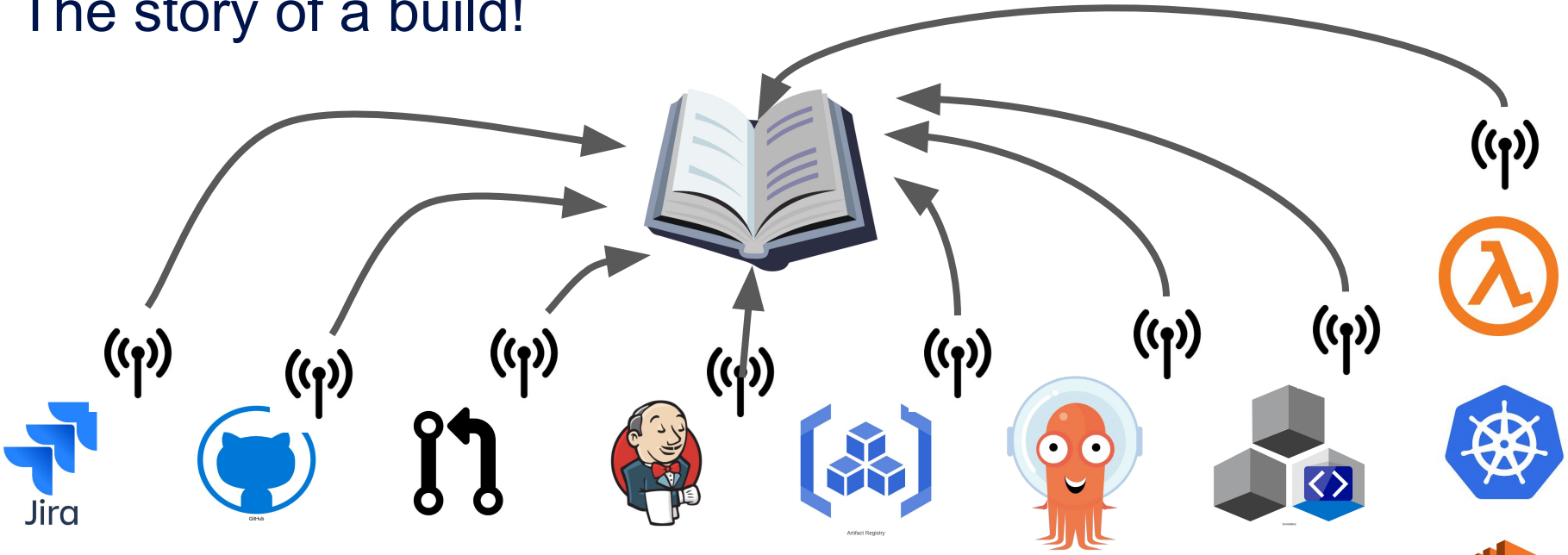


Telling the SDLC “story”

- We can stop the story if we don’t like it
- We can skim the book
  - Start with the pages that make book “trusted” faster
  - Then embellish to make it sound even better



The story of a build!



Signals of Trust

- A snapshot of SDLC
- Each signal is a statement about something
  - Something we expect to happen
  - But it must be instrumented!
- The statements are “attested” to
  - Just like testifying to a statement in a court

The “Trust Telemetry” of the Story

- We can start to judge builds against criteria
  - Specify through policies
  - Policies for each phase or even each step
  - Manual review when things look strange?

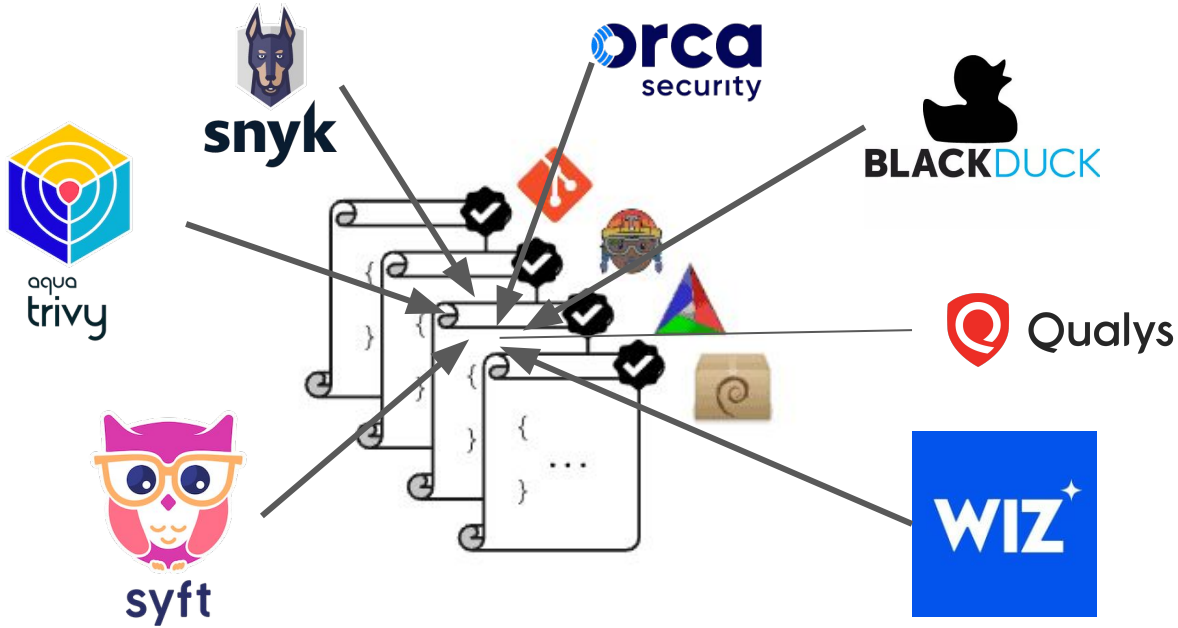
Daily Trust Telemetry

Daily Trust Telemetry

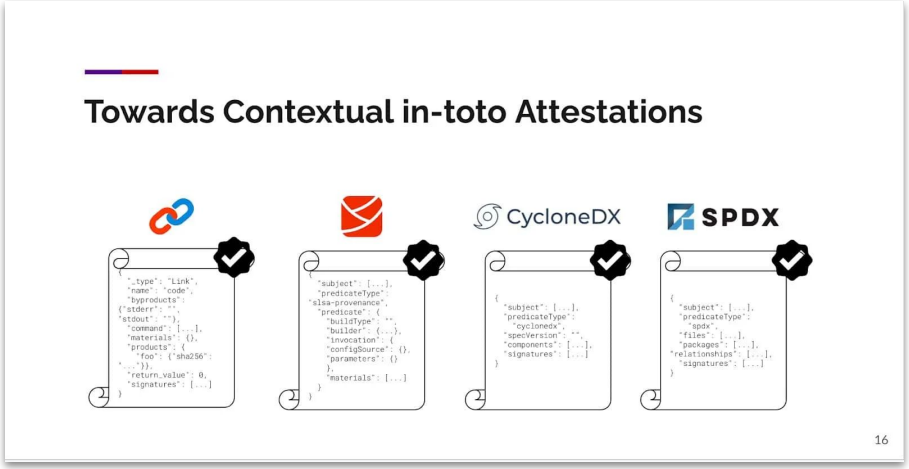
Attestations provide the context

- Policy decisions are smarter with more context
  - More context brings picture into focus
- Individually, attestations provide partial trust
- In aggregate, stronger than the sum of their parts

Attestations bridge the diversity



Attestations bridge the diversity

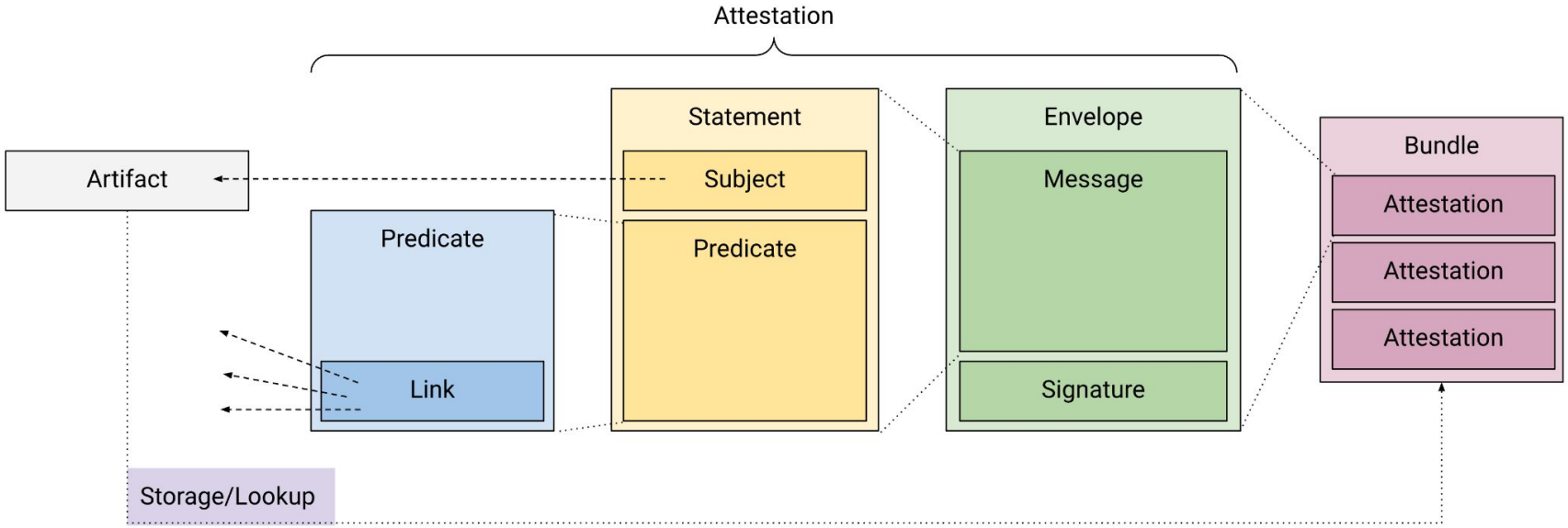


- in-toto's attestation framework is the "API of DevSecOps"
- Blog post: https://www.cncf.io/blog/2023/08/17/unleashing-in-toto-the-api-of-devsecops/
- Kubecon EU 2023 Preso: https://www.youtube.com/watch?v=ezV_oWBPqKw

Focus on Attestations

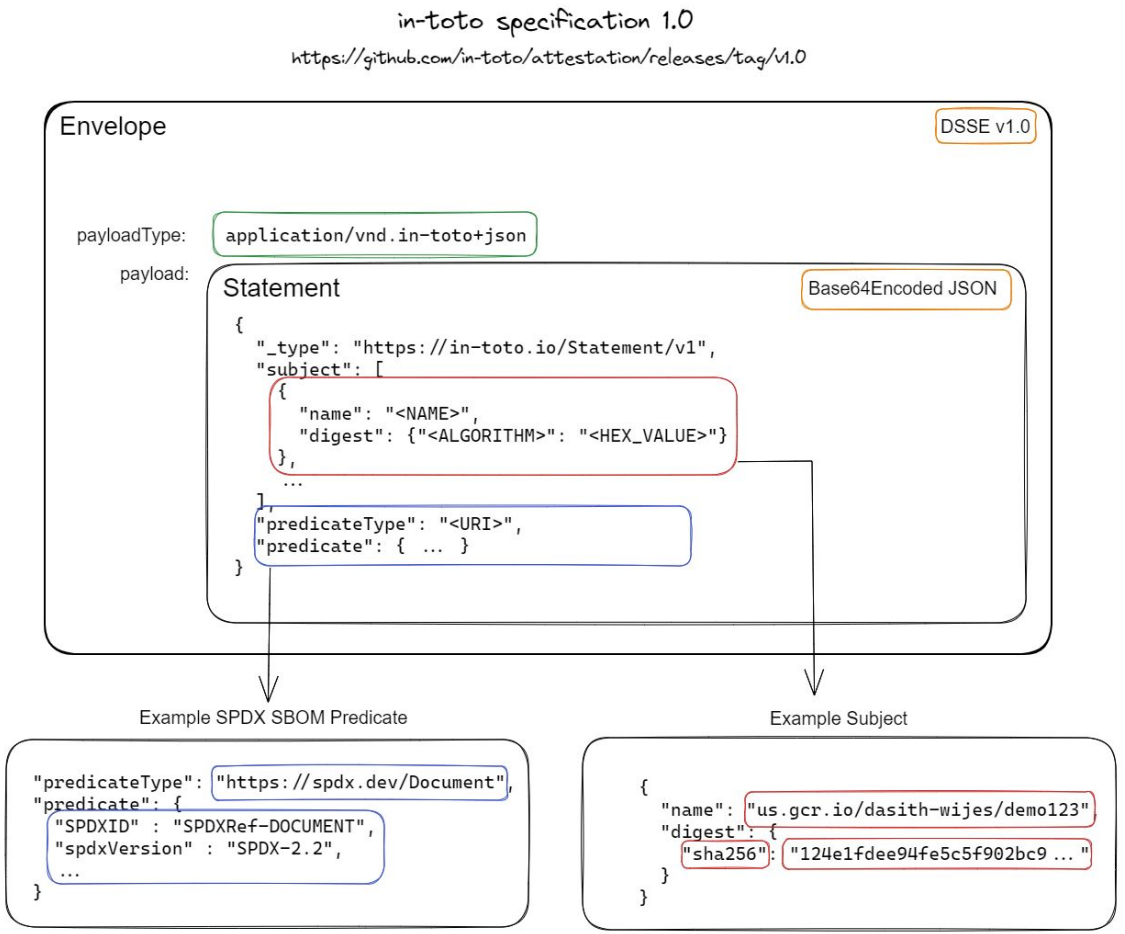
- “Claims” about a process
- Metadata
- Time stamped
- Persistent
- Verifiable
- Examples:
  - in-toto attestations
  - SCITT statements

Focus on attestations ctd



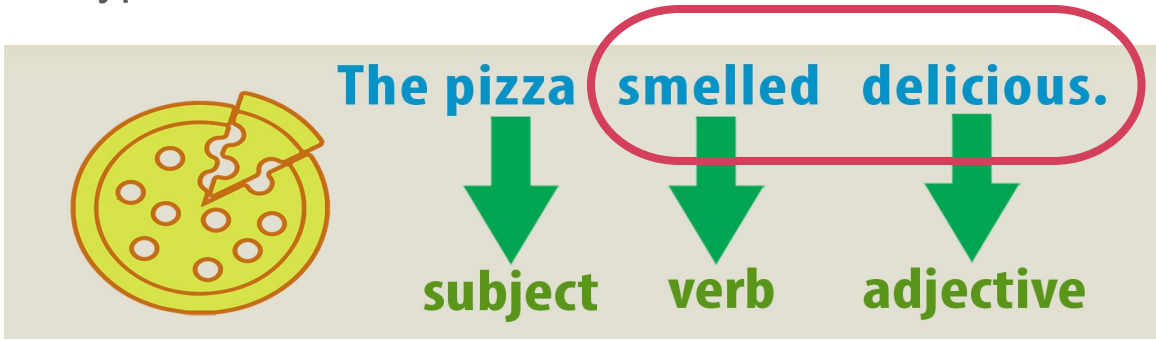
The in-toto Attestation

- Predicate: Defines an action taken
- Statement: Binds the action to the subject of the action.
- Envelope: Enables verification and ultimately trust.



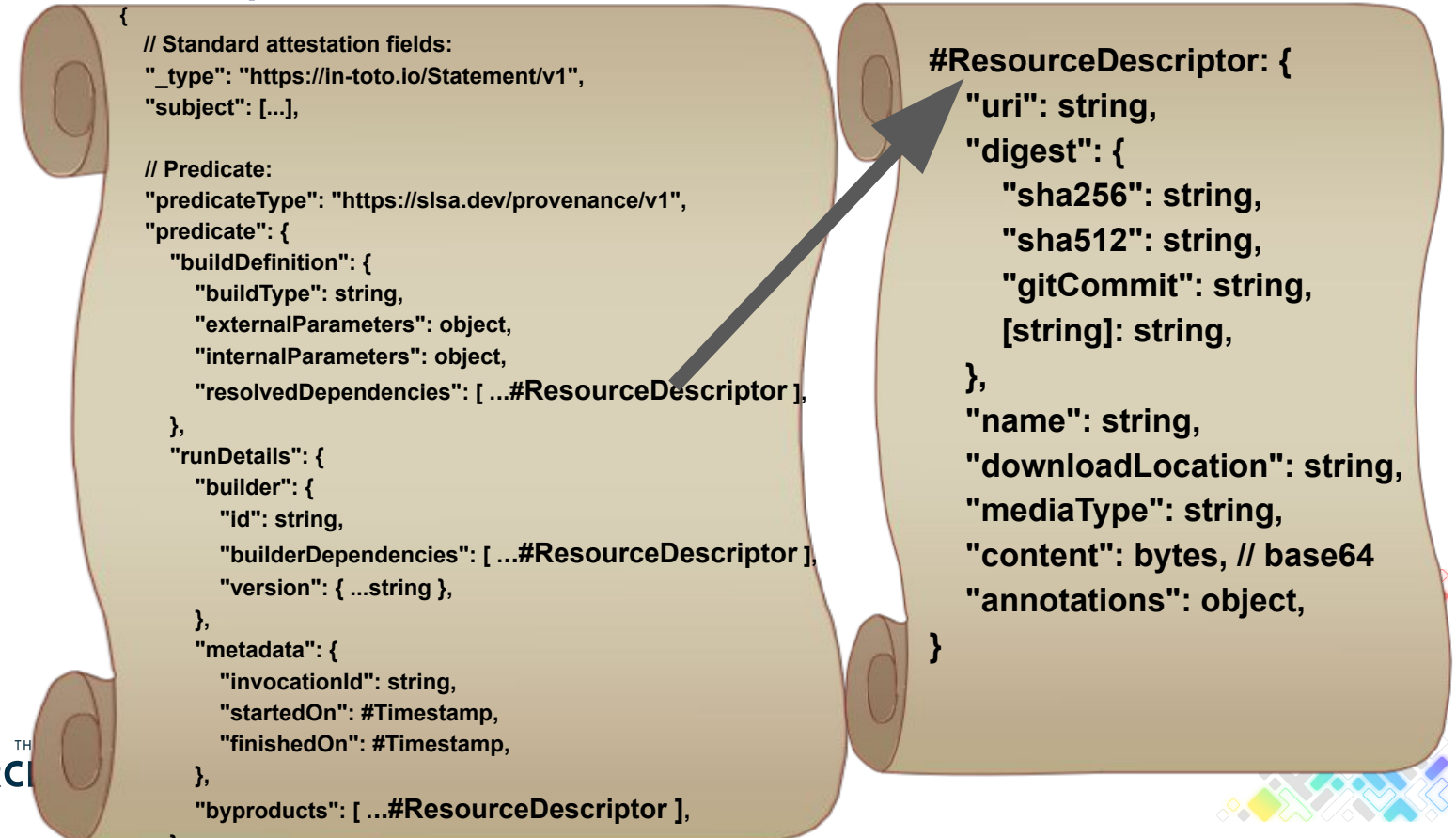
Predicates = the “types” of actions

- “The software was scanned”
- Many common predicates or “types”
  - SBOM
  - SLSA Provenance
  - Test results
  - Runtime Traces
- Custom predicates
  - Escape hatch
  - Semantics are hard



Let's look at example attestations

SLSA Provenance:



Other interesting ones…

SBOM:

```
{
  // Standard attestation fields:
  "_type": "https://in-toto.io/Statement/v0.1",
  "subject": [{ ... }],

  // Predicate:

"predicateType":"https://cyclonedx.org/bom/v1.
4",
  "predicate": {
    "bomFormat": "CycloneDX",
    "specVersion": "1.4",
    "serialNumber":
"urn:uuid:3e671687-395b-41f5-a30f-a58921a69b7
9",
    "version": 1,
    "components": [
       {
       "type": "library",
       "name": "acme-library",
       "version": "1.0.0"
       }
    ]
    ...
  }
}
```

Vuln Scan:

```
{
  "_type": "https://in-toto.io/Statement/v0.1",
  "subject": [
    {
      ...
    }
  ],
  // Predicate:
  "predicateType":
"https://cosign.sigstore.dev/attestation/vuln/
v1",
  "predicate": {
    "invocation": {
      "parameters": [],
      // [ "--format=json", "--skip-db-update" ]
      "uri": "",
      //
https://github.com/developer-guy/alpine/acti
ons/runs/1071875574
      "event_id": "",
      // 1071875574
      "builder.id": "" …
```

Generating Attestations

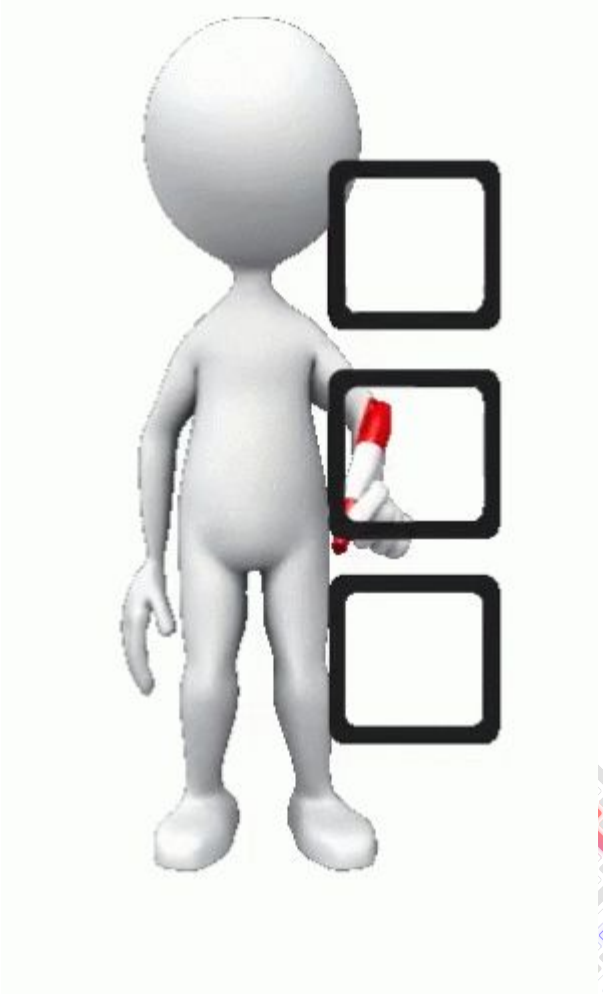
A few tools can create attestations:

- In-toto suite (in-toto-run, in-toto-record, in-toto-sign)
  - Requires Layouts
  - Creates Link files
- Witness
  - Does not need layouts
  - Automated data collection
  - Leverages in-toto attestations
- cosign
  - Part of the sigstore suite
  - Can take in attestation as json
  - Store with container image in OCI registry
- slsa-attestor
  - Specifically for slsa provenance

Verifying Attestations

And their verifying pairs:

- in-toto-verify
  - Verifies layouts and link files
- witness verify command
  - Verfies against witness (rego) policies
- cosign verify
  - Validates attestation signatures
  - Can validate with cue or rego policies
- slsa-verifier
  - Verifies slsa provenance attestations
  - Limited to provenance specifics
    - Builder id
    - Src code repo info

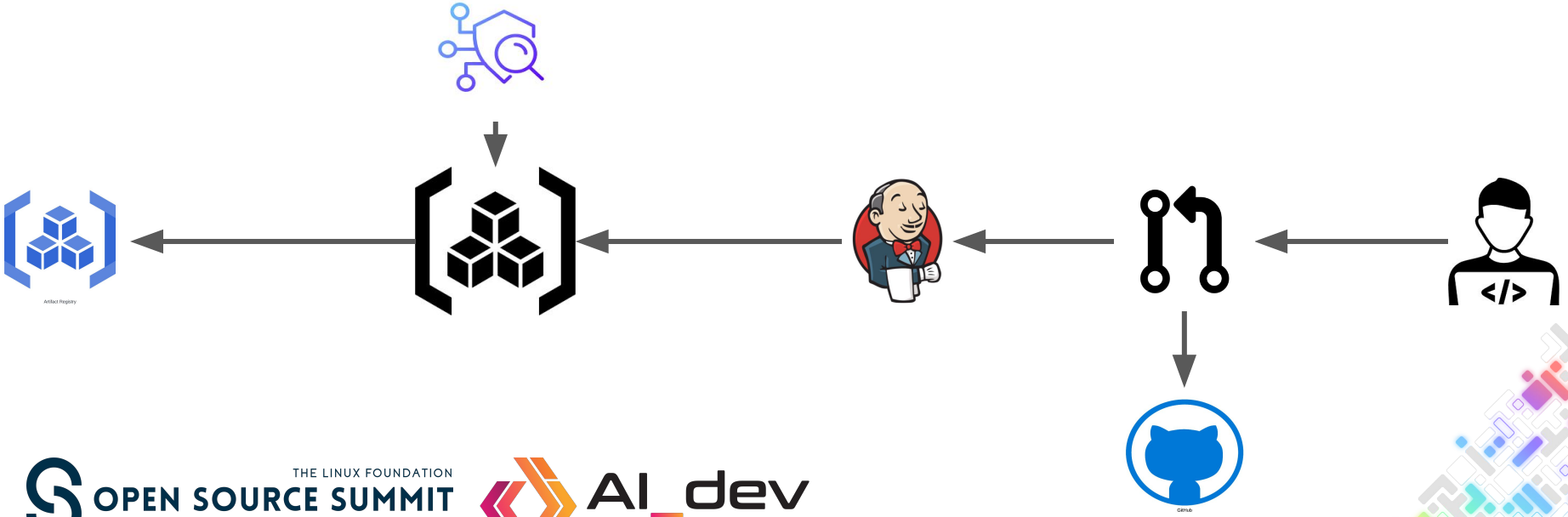


Spotlight on Witness

- Allows us to declare policies independently of build pipeline design
- Automated Data collection to reduce developer friction.
- Allows separation of concerns from software developers and policy owners
    - Important when security and compliance teams need ownership over policy

But where are they stored?

- Archivista is a graph and storage service for in-toto attestations.
- Enables the discovery and retrieval of attestations for software artifacts.



Now let’s see an attestation get generated

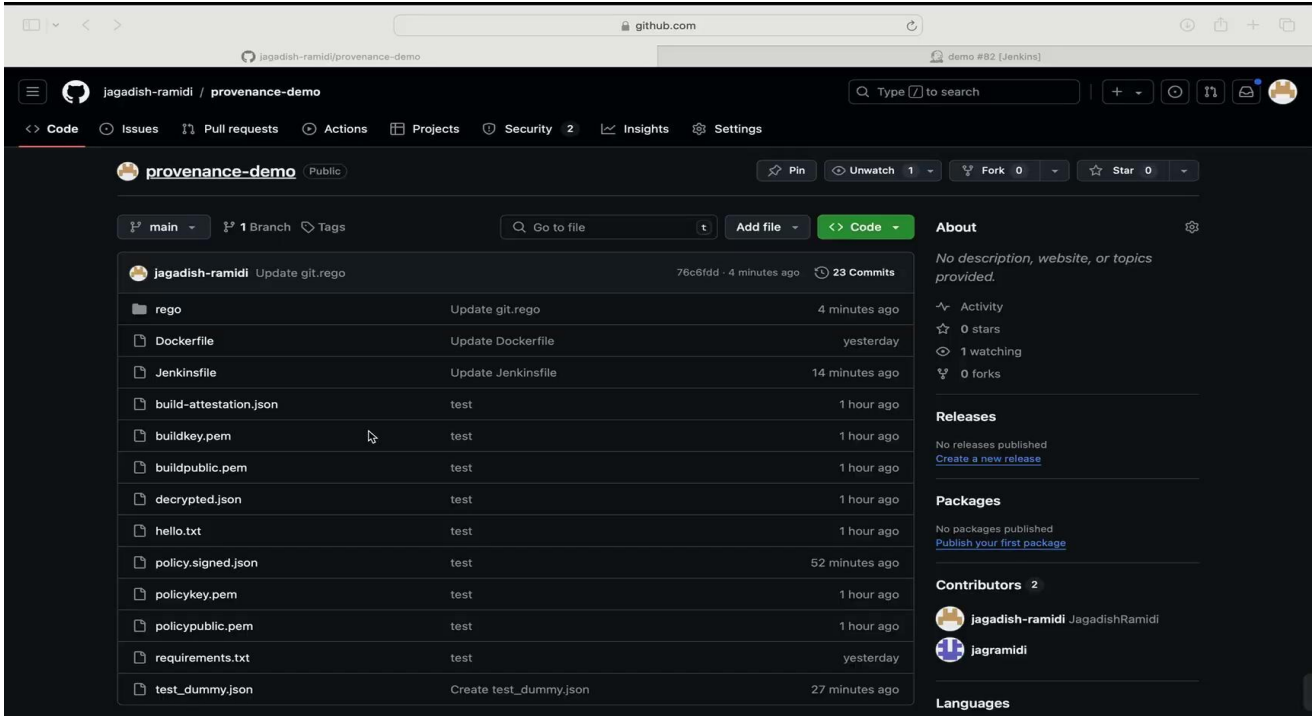
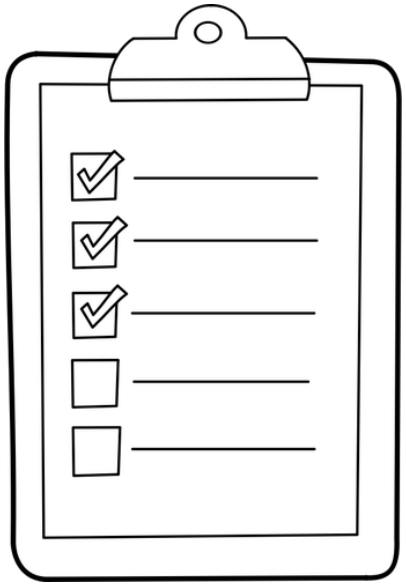


Diagram of the build SDLC as attestations in graphql viz

How to get started

- Start by accounting for your entire SDLC process
  - Threat modeling helps!
- Find the places that you can instrument
- Sort by the ones that are most helpful in telling the “story”
- Instrument and write policy to enforce expectations
- Repeat

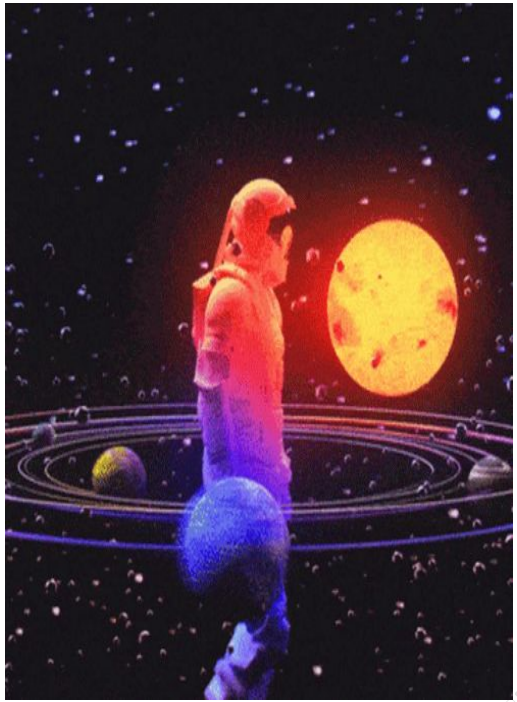


Where can we take this?

- Centralized policy evaluation
  - More on this next…
- AI/ML
  - Trained on data lake
  - Can augmented with software quality/incident data
  - Enable earlier “trust” decisions
  - Alerting?
  - Automated Remediation?

Centralized Policy Decisions

- Central data lake w/ graph API is foundation
- Evaluation of policies can be centralized into an API
- Policy evaluation invocation becomes pluggable
- Managing policies in diverse environment easier
- Enables the separation of concerns
-



Thank you!

Slides/demo and supporting links:
https://github.com/jessesanford/scscon-japan-2024

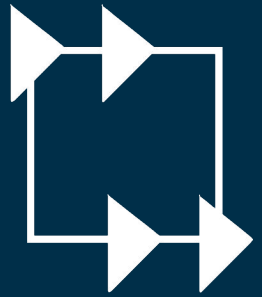
Follow us on linkedin!

/in/jessesanford

/in/jagadishramidi/

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT JAPAN



SUPPLYCHAIN SECURITYCON