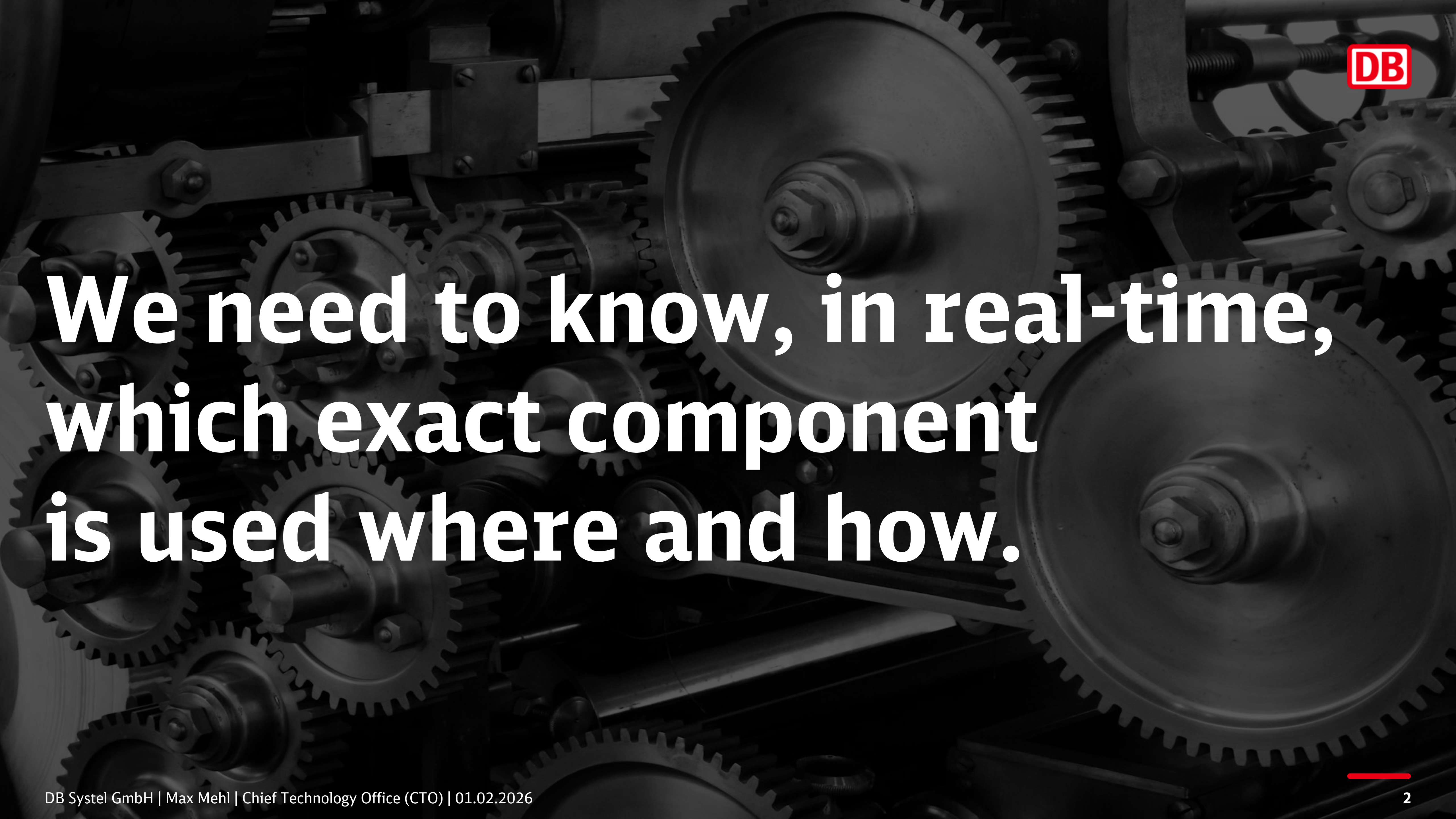
Deutsche Bahn's Approach to Large-Scale SBOM Collection and Use

From Operational Need to Concrete Implementation

DB Systel GmbH | CTO Team | Max Mehl | FOSDEM 2026 | 01.02.2026



We need to know, in real-time, which exact component is used where and how.

Deutsche Bahn’s Business is Trains, not Software

But its IT is equally large

Our Core Business

Transporting people and goods.

- 5,1 million train travelers / day
- 60,970 km of tracks
- 5,700 train stations
- 22,500 trains / day
- 180 million tons of freight / year

Complex Organization

A large and diverse organization keeps our core business running every day.

- 220,000+ employees
- 500+ professions
- Hundreds of subsidiaries

Digitalization is Essential to Scale in the Future

Without IT – and Open Source – no train would be able to run.

- 7,000+ IT applications/services
- 10,000+ IT professionals
- 20,000+ virtual machines
- 40,000+ containers
- 60,000+ repositories
- 100,000+ OSS components

Example: DB Navigator for information and ticketing

The essential entry point for most travelers.

- 23 million users per month
- 170 million travel information requests per month

Transparent Supply Chains: Easier Said than Done

At DB, we have the most diverse sourcing streams for IT

Build software

- For ourselves (services, internal)
- For external customers (you)
- Ranging from operating systems for displays in trains, to services, to apps on your phones

Buy software

- Local
- On-premise
- SaaS
- Bundled in hardware (like trains)

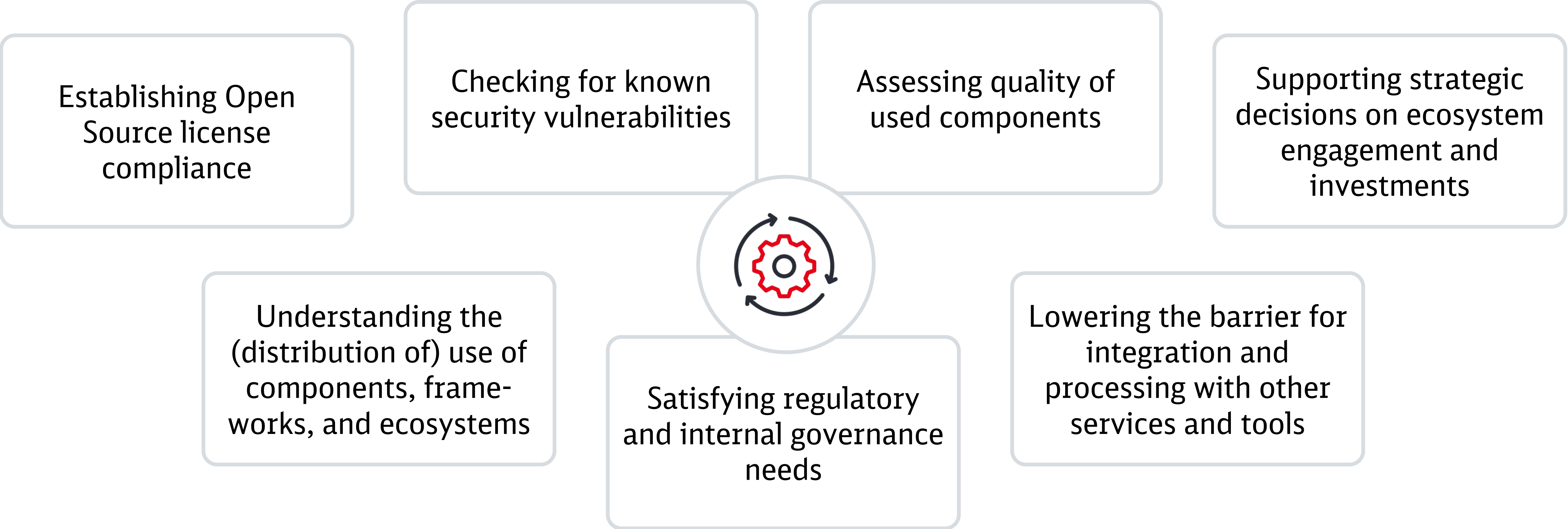
Operate software

- On-premise
- Cloud (VM and containers)
- Edge (embedded)

Which software components are where? And in which state and context?

SBOMs As a Common Methodology to Tackle Challenges

SBOM is not a means by itself, but a standardized method to support several needs



Establishing Open Source license compliance

Checking for known security vulnerabilities

Assessing quality of used components

Supporting strategic decisions on ecosystem engagement and investments

Understanding the (distribution of) use of components, frameworks, and ecosystems

Satisfying regulatory and internal governance needs

Lowering the barrier for integration and processing with other services and tools

SBOMs must become shared infrastructure.

VEX is a Perfect Match for SBOMs

VEX as a perfect match

- Standardized way to make a statement on the status of a known vulnerability detected in one’s supply chain
- Match CVE to component found in an SBOM
- Track status information throughout involved processes and tools, avoid duplicated work for teams
- Allow manufacturers to communicate their interpretation of affection status to us

Reality: integrate a new underlying standard beneath existing processes and tools → challenging in large organizations

To be effective, VEX and SBOMs must be thought together.

Creating an SBOM Strategy and Architecture from Scratch

Challenges

- Size and diversity of the organization
- Various software sourcing models
- DB’s different roles and requirements
- Many stakeholders and user groups
- Preset tools and processes
- Limited resources of teams
- Pressure of time, e.g. by the CRA

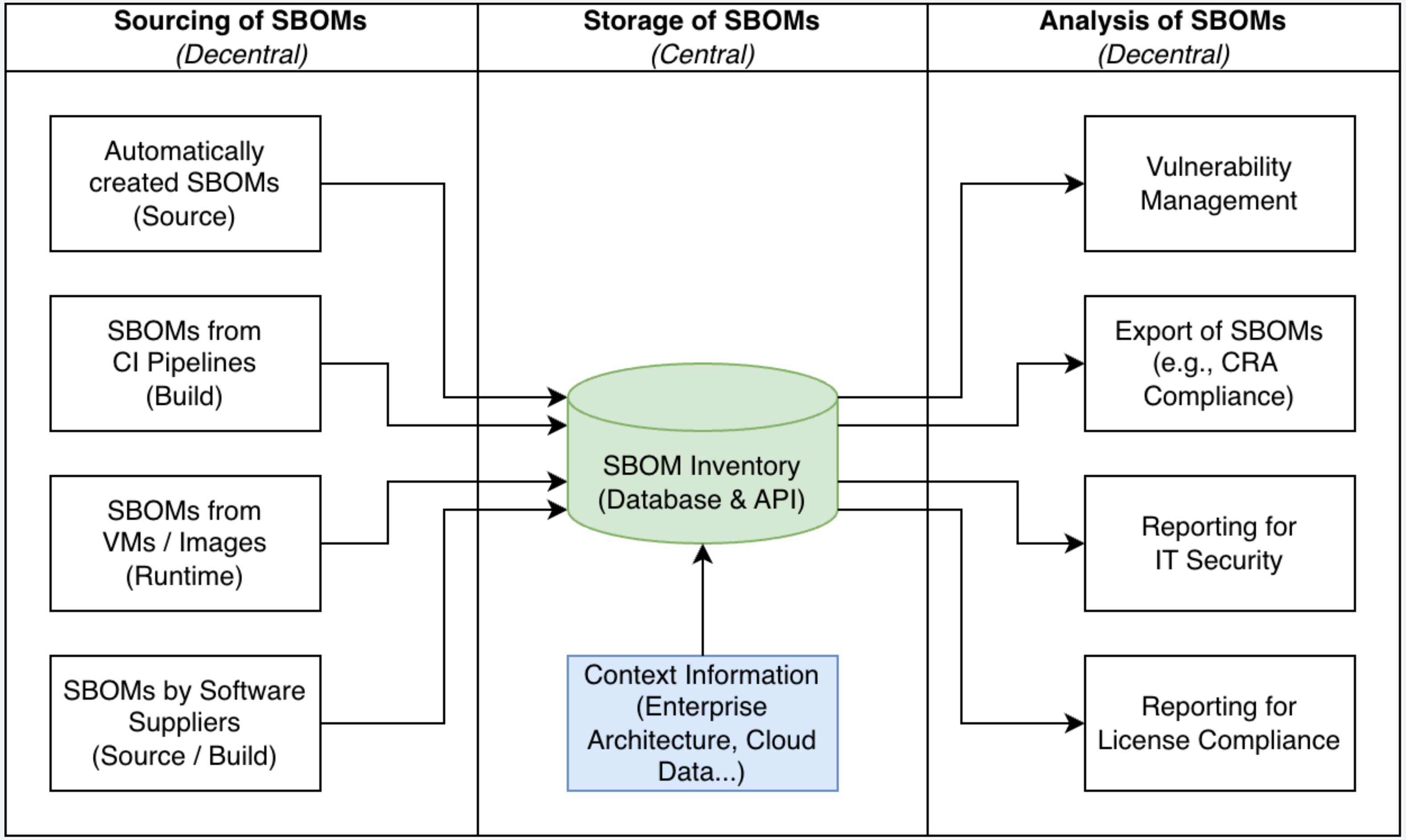
Procedural principles

- Small, interdisciplinary group, consisting of volunteers
- Iterate quickly, gather feedback continuously
- Do not talk in tools, but capabilities
- Focus on existing needs of the organization, not abstract recommendations with all the bells and whistles
- Think big, expect incremental realization
- Document progress and material organization-public

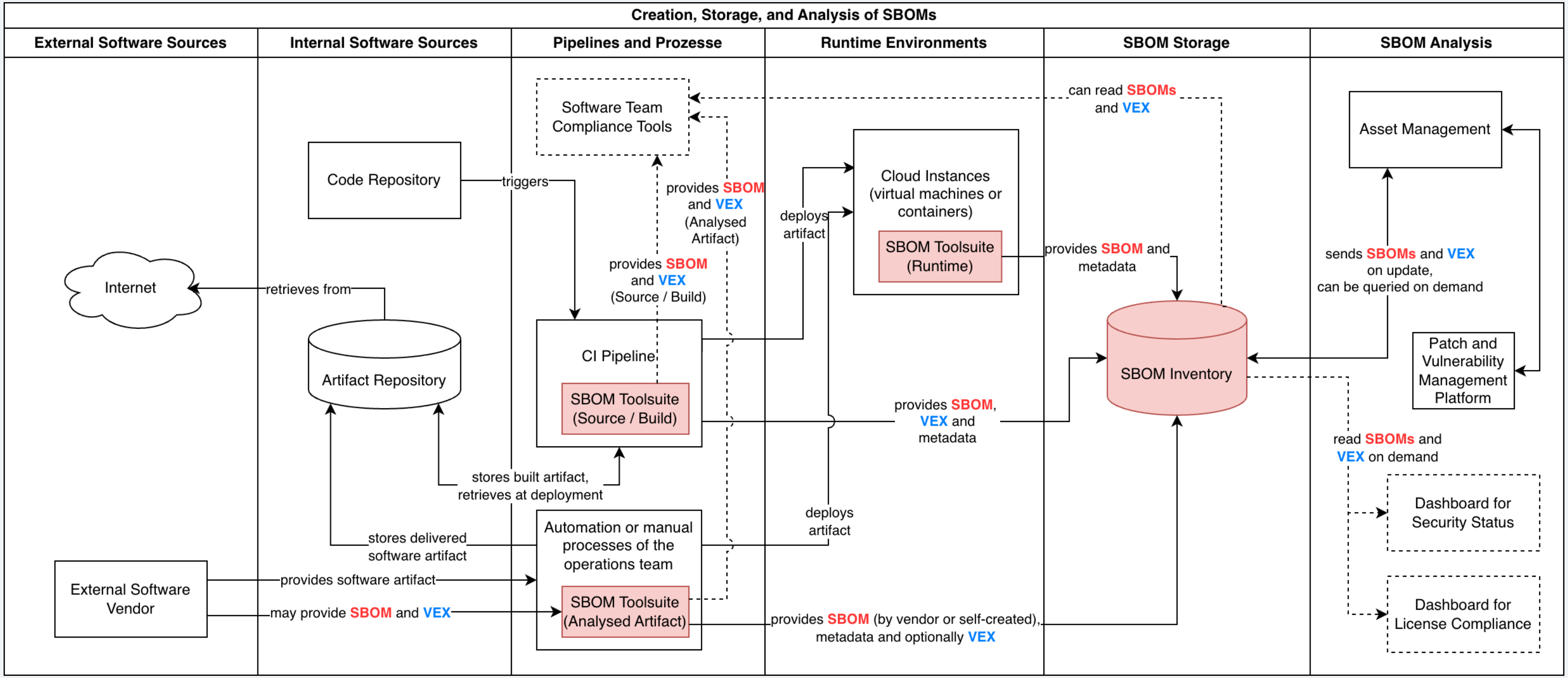
Technical and architectural principles

- Consider all sourcing and SBOM types incl. VEX
- Modularity
- Open standards and interfaces
- Central storage of SBOMs
- Decentral sourcing and analysis of SBOMs

Our Mental Model of SBOM Lifecycle Consists of Three Phases


Sourcing of SBOMs
(Decentral)
Storage of SBOMs
(Central)
Analysis of SBOMs
(Decentral)
Automatically created SBOMs (Source)
SBOMs from CI Pipelines (Build)
SBOMs from VMs / Images (Runtime)
SBOMs by Software Suppliers (Source / Build)
SBOM Inventory (Database & API)
Context Information (Enterprise Architecture, Cloud Data...)
Vulnerability Management
Export of SBOMs (e.g., CRA Compliance)
Reporting for IT Security
Reporting for License Compliance

The SBOM Blueprint is Our Guiding Star



Creation, Storage, and Analysis of SBOMs

External Software Sources | Internal Software Sources | Pipelines and Prozesse | Runtime Environments | SBOM Storage | SBOM Analysis

Internet

External Software Vendor

Code Repository

Artifact Repository

retrieves from

triggers

stores built artifact, retrieves at deployment

stores delivered software artifact

provides software artifact

may provide SBOM and VEX

Software Team Compliance Tools

CI Pipeline

SBOM Toolsuite (Source / Build)

provides SBOM and VEX (Source / Build)

provides SBOM and VEX (Analysed Artifact)

Automation or manual processes of the operations team

SBOM Toolsuite (Analysed Artifact)

deploys artifact

deploys artifact

Cloud Instances (virtual machines or containers)

SBOM Toolsuite (Runtime)

provides SBOM, VEX and metadata

provides SBOM (by vendor or self-created), metadata and optionally VEX

can read SBOMs and VEX

provides SBOM and metadata

SBOM Inventory

Asset Management

Patch and Vulnerability Management Platform

sends SBOMs and VEX on update, can be queried on demand

read SBOMs and VEX on demand

Dashboard for Security Status

Dashboard for License Compliance

Last updated: March 2025

Implementation of Architectural Blueprint by Prioritized Increments

- Given the preconditions, implementation cannot happen overnight
- Prioritization based on identified risks, external requirements, and pragmatism

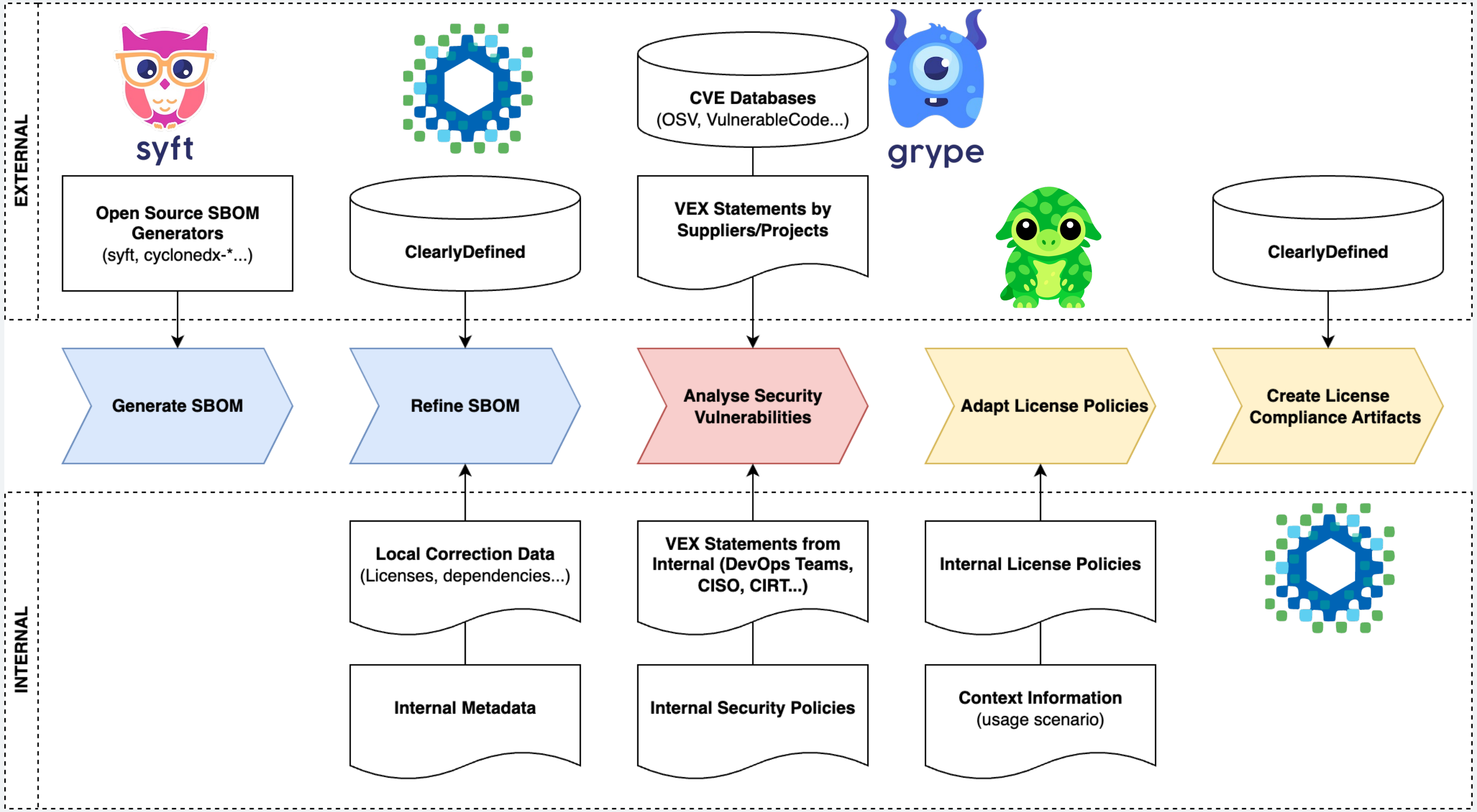
Results

- Focus on Source/Build SBOMs for software developed in-house
- Onboard as many teams as possible
- Low-threshold drop-in solutions for CI pipelines and their templates
- Increase SBOM Quality, especially licenses and metadata → but balance quality vs quantity
- Teams: Integration into compliance portal
- Governance: Enable basic central insights, no shiny dashboards
- Focus on Happy Paths, do not consider all edge cases from the start

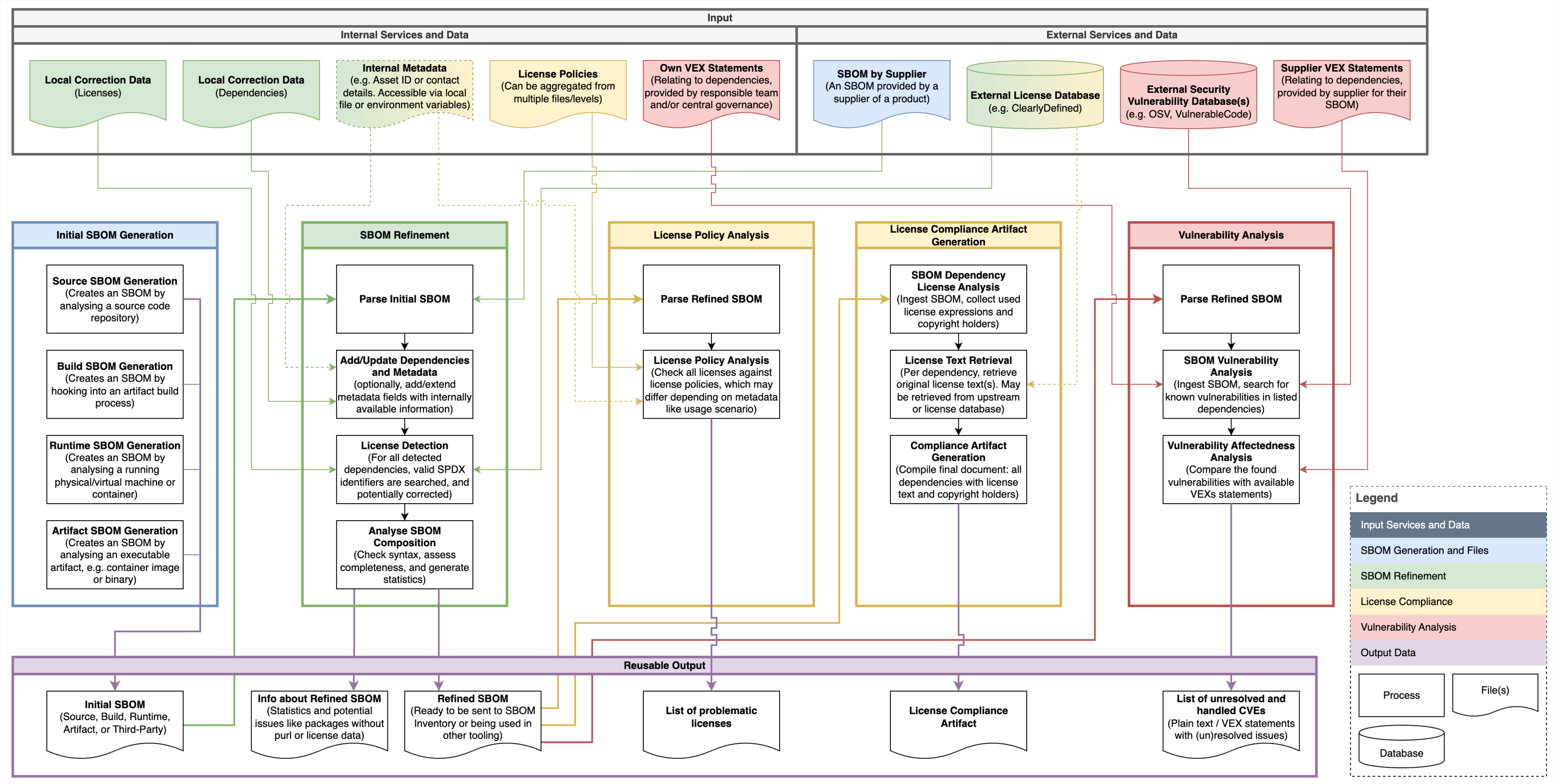
Future Steps and Improvements

- Runtime SBOMs from VMs and containers
- Easier ingestion of SBOMs delivered by vendors
- Support of OT and low-level IT close to hardware

Modular Toolchain to Generate, Enrich, and Analyze SBOMs


EXTERNAL
syft
Open Source SBOM Generators
(syft, cyclonedx-*...)
ClearlyDefined
CVE Databases
(OSV, VulnerableCode...)
grype
VEX Statements by Suppliers/Projects
ClearlyDefined
Generate SBOM
Refine SBOM
Analyse Security Vulnerabilities
Adapt License Policies
Create License Compliance Artifacts
INTERNAL
Local Correction Data
(Licenses, dependencies...)
VEX Statements from Internal (DevOps Teams, CISO, CIRT...)
Internal License Policies
Internal Metadata
Internal Security Policies
Context Information
(usage scenario)

Detailed Look Into SBOM Flow and Interconnected Services



Input

Internal Services and Data

Local Correction Data (Licenses)

Local Correction Data (Dependencies)

Internal Metadata (e.g. Asset ID or contact details. Accessible via local file or environment variables)

License Policies (Can be aggregated from multiple files/levels)

Own VEX Statements (Relating to dependencies, provided by responsible team and/or central governance)

External Services and Data

SBOM by Supplier (An SBOM provided by a supplier of a product)

External License Database (e.g. ClearlyDefined)

External Security Vulnerability Database(s) (e.g. OSV, VulnerableCode)

Supplier VEX Statements (Relating to dependencies, provided by supplier for their SBOM)

Initial SBOM Generation

Source SBOM Generation (Creates an SBOM by analysing a source code repository)

Build SBOM Generation (Creates an SBOM by hooking into an artifact build process)

Runtime SBOM Generation (Creates an SBOM by analysing a running physical/virtual machine or container)

Artifact SBOM Generation (Creates an SBOM by analysing an executable artifact, e.g. container image or binary)

SBOM Refinement

Parse Initial SBOM

Add/Update Dependencies and Metadata (optionally, add/extend metadata fields with internally available information)

License Detection (For all detected dependencies, valid SPDX identifiers are searched, and potentially corrected)

Analyse SBOM Composition (Check syntax, assess completeness, and generate statistics)

License Policy Analysis

Parse Refined SBOM

License Policy Analysis (Check all licenses against license policies, which may differ depending on metadata like usage scenario)

License Compliance Artifact Generation

SBOM Dependency License Analysis (Ingest SBOM, collect used license expressions and copyright holders)

License Text Retrieval (Per dependency, retrieve original license text(s). May be retrieved from upstream or license database)

Compliance Artifact Generation (Compile final document: all dependencies with license text and copyright holders)

Vulnerability Analysis

Parse Refined SBOM

SBOM Vulnerability Analysis (Ingest SBOM, search for known vulnerabilities in listed dependencies)

Vulnerability Affectedness Analysis (Compare the found vulnerabilities with available VEXs statements)

Reusable Output

Initial SBOM (Source, Build, Runtime, Artifact, or Third-Party)

Info about Refined SBOM (Statistics and potential issues like packages without purl or license data)

Refined SBOM (Ready to be sent to SBOM Inventory or being used in other tooling)

List of problematic licenses

License Compliance Artifact

List of unresolved and handled CVEs (Plain text / VEX statements with (un)resolved issues)

Legend

Input Services and Data

SBOM Generation and Files

SBOM Refinement

License Compliance

Vulnerability Analysis

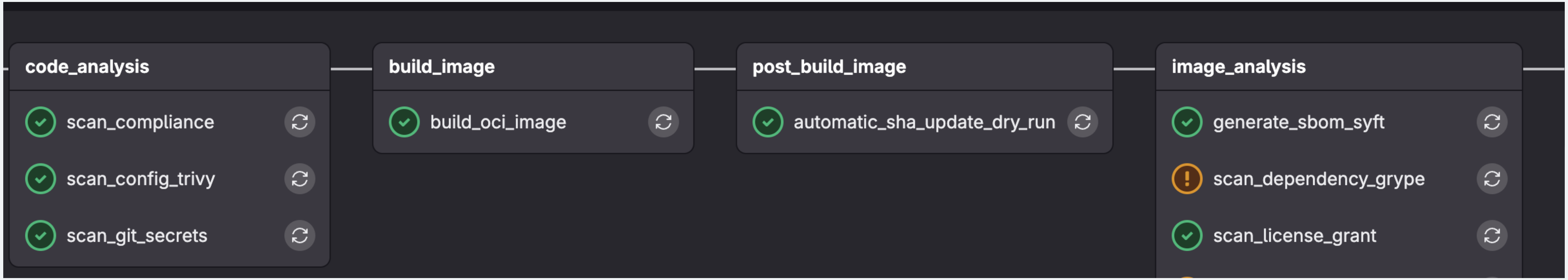
Output Data

Process

File(s)

Database

SBOM Toolsuite Locally and in Pipelines



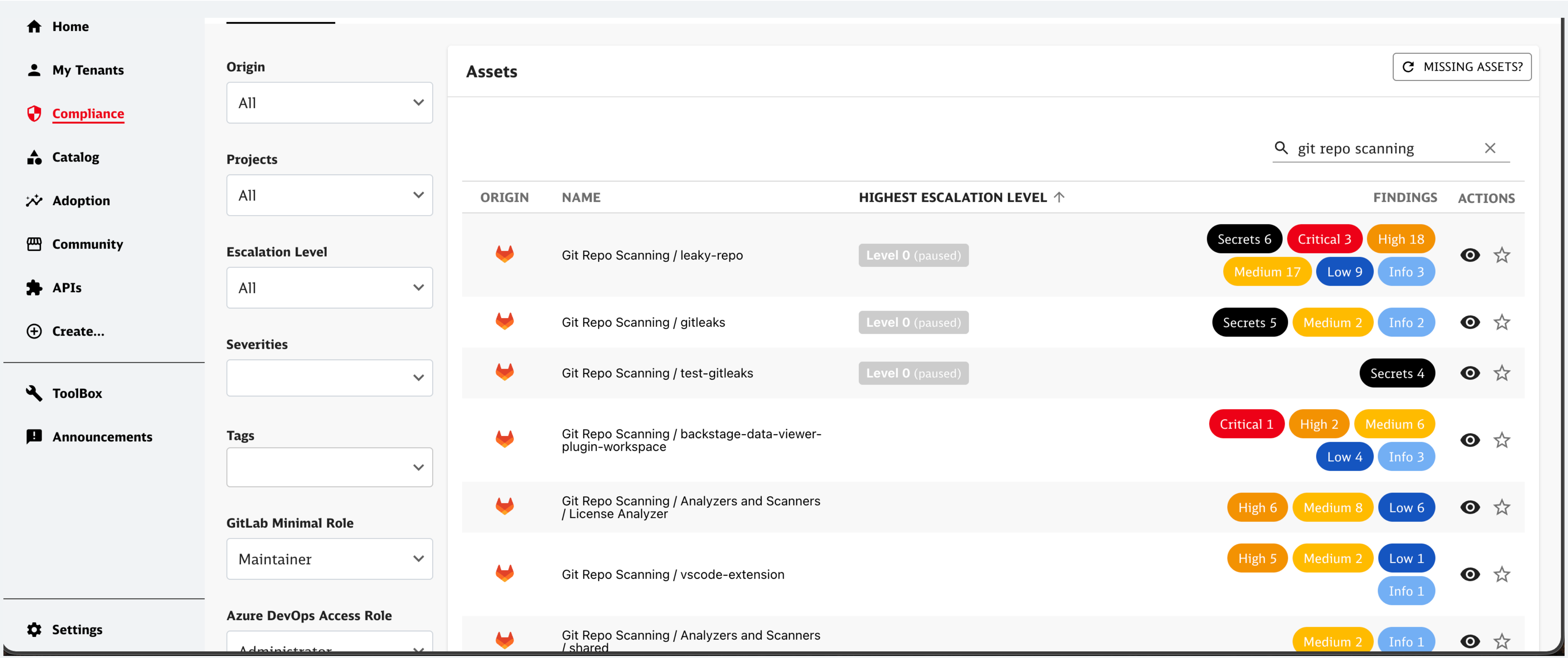
code_analysis
- ✓ scan_compliance
- ✓ scan_config_trivy
- ✓ scan_git_secrets

build_image
- ✓ build_oci_image

post_build_image
- ✓ automatic_sha_update_dry_run

image_analysis
- ✓ generate_sbom_syft
- ! scan_dependency_grype
- ✓ scan_license_grant

Compliance Suite: Point and Click for Teams and Owners



Origin: All

Projects: All

Escalation Level: All

Severities:

Tags:

GitLab Minimal Role: Maintainer

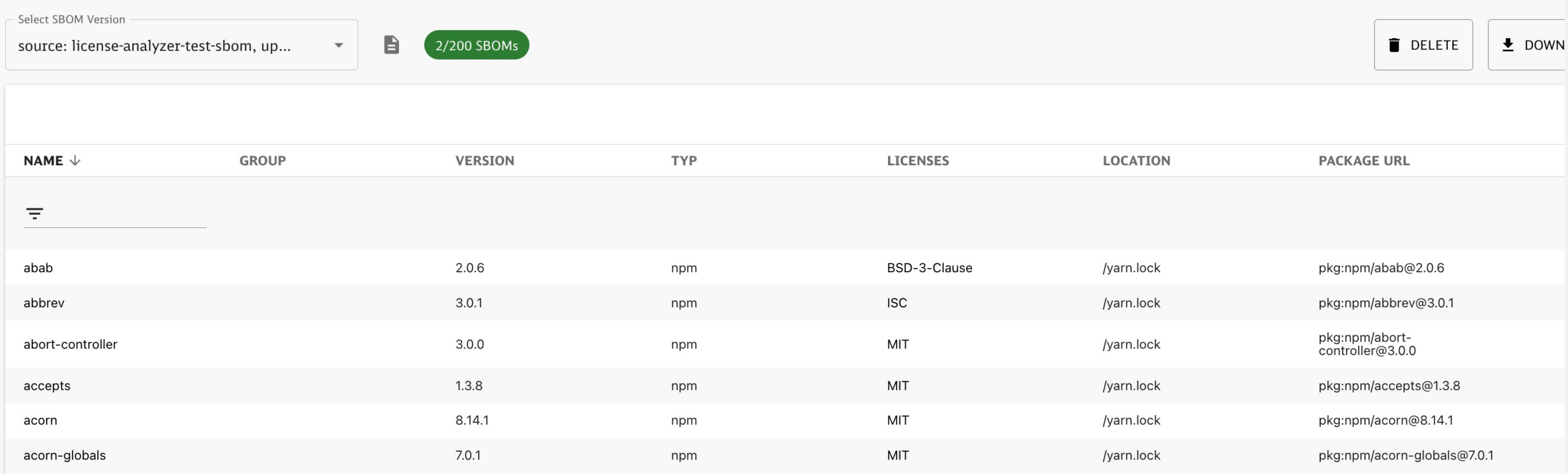
Azure DevOps Access Role: Administrator

Assets

git repo scanning

ORIGIN	NAME	HIGHEST ESCALATION LEVEL ↑	FINDINGS	ACTIONS
	Git Repo Scanning / leaky-repo	Level 0 (paused)	Secrets 6 Critical 3 High 18 Medium 17 Low 9 Info 3	👁 ☆
	Git Repo Scanning / gitleaks	Level 0 (paused)	Secrets 5 Medium 2 Info 2	👁 ☆
	Git Repo Scanning / test-gitleaks	Level 0 (paused)	Secrets 4	👁 ☆
	Git Repo Scanning / backstage-data-viewer-plugin-workspace		Critical 1 High 2 Medium 6 Low 4 Info 3	👁 ☆
	Git Repo Scanning / Analyzers and Scanners / License Analyzer		High 6 Medium 8 Low 6	👁 ☆
	Git Repo Scanning / vscode-extension		High 5 Medium 2 Low 1 Info 1	👁 ☆
	Git Repo Scanning / Analyzers and Scanners / shared		Medium 2 Info 1	👁 ☆

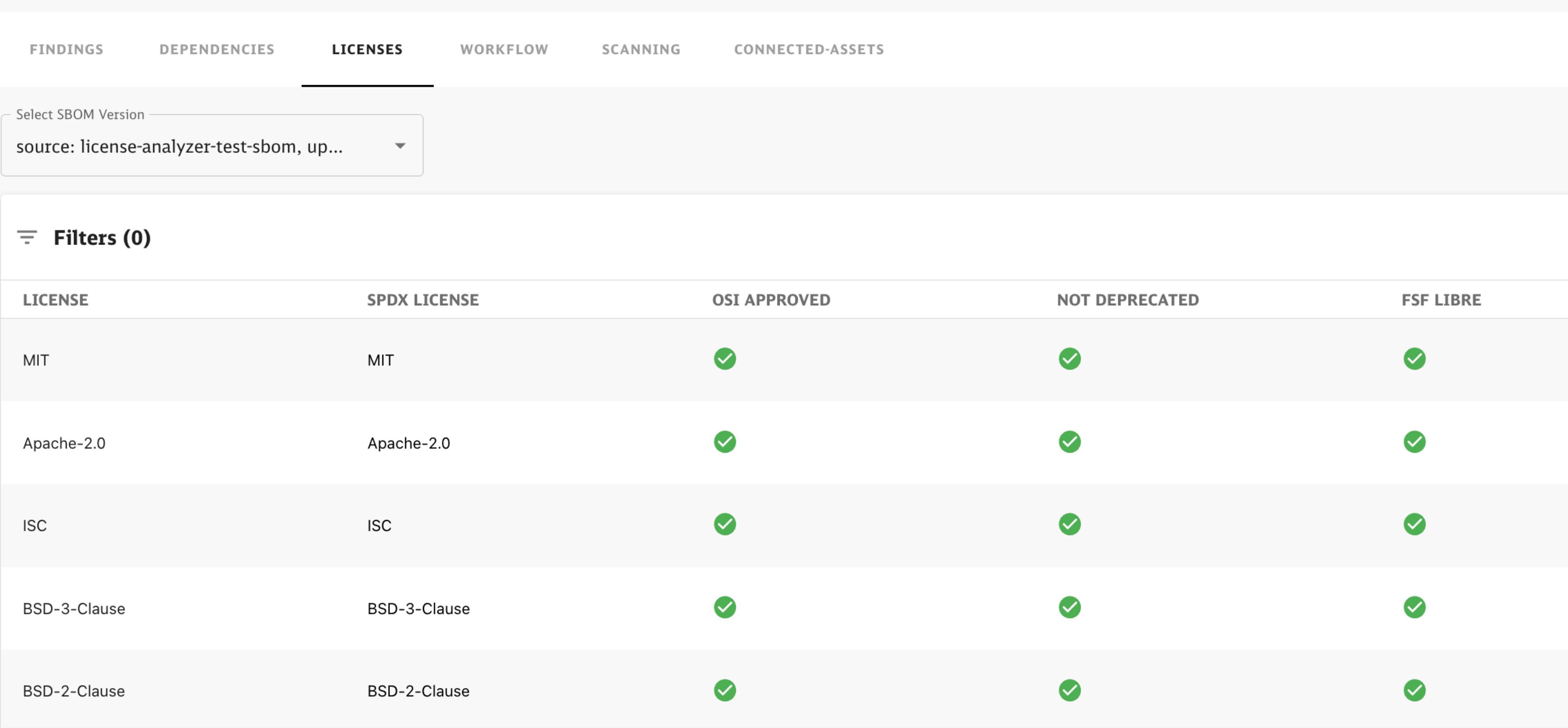
Compliance Suite: Inspect and Verify SBOMs



Select SBOM Version: source: license-analyzer-test-sbom, up...

NAME ↓	GROUP	VERSION	TYP	LICENSES	LOCATION	PACKAGE URL
abab		2.0.6	npm	BSD-3-Clause	/yarn.lock	pkg:npm/abab@2.0.6
abbrev		3.0.1	npm	ISC	/yarn.lock	pkg:npm/abbrev@3.0.1
abort-controller		3.0.0	npm	MIT	/yarn.lock	pkg:npm/abort-controller@3.0.0
accepts		1.3.8	npm	MIT	/yarn.lock	pkg:npm/accepts@1.3.8
acorn		8.14.1	npm	MIT	/yarn.lock	pkg:npm/acorn@8.14.1
acorn-globals		7.0.1	npm	MIT	/yarn.lock	pkg:npm/acorn-globals@7.0.1

Compliance Suite: Investigate Findings

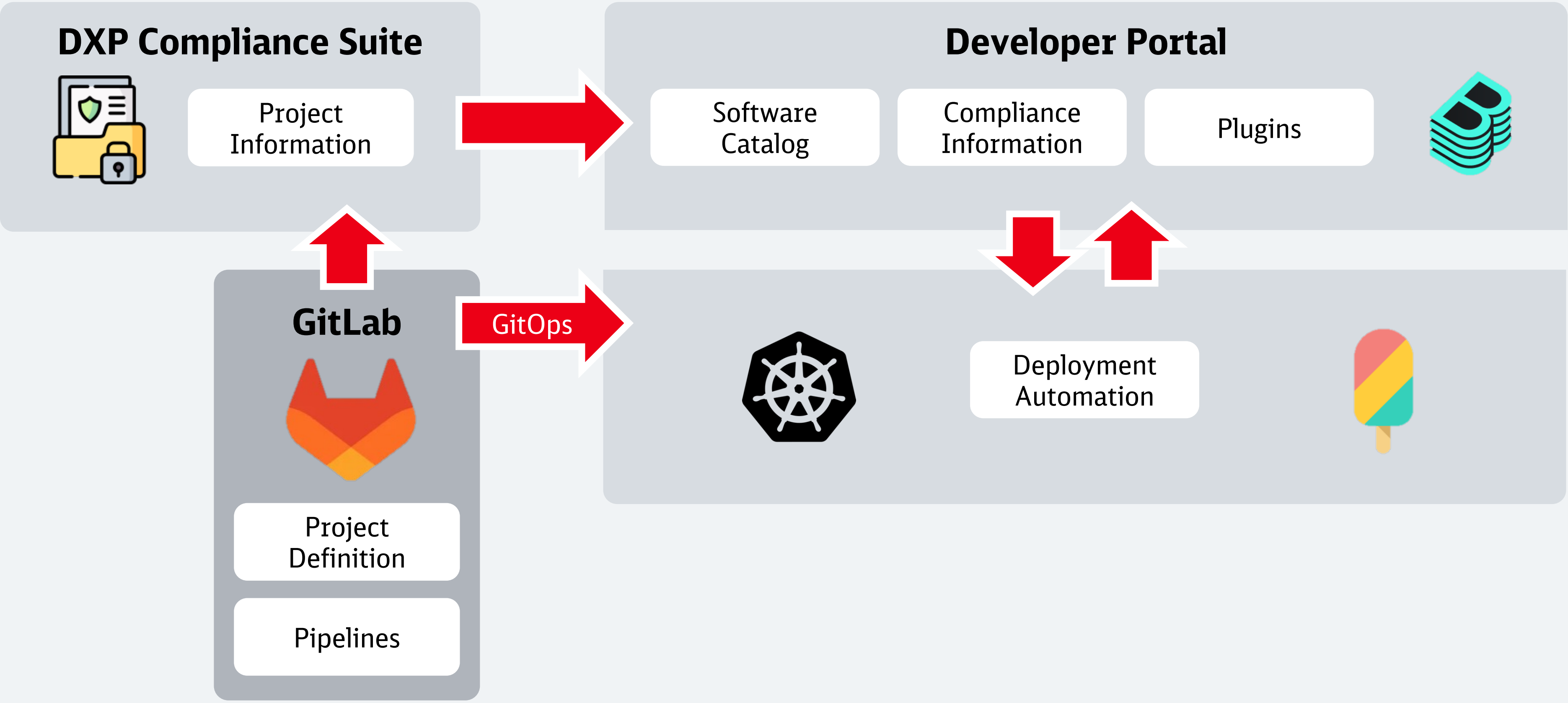


Select SBOM Version
source: license-analyzer-test-sbom, up...

Filters (0)

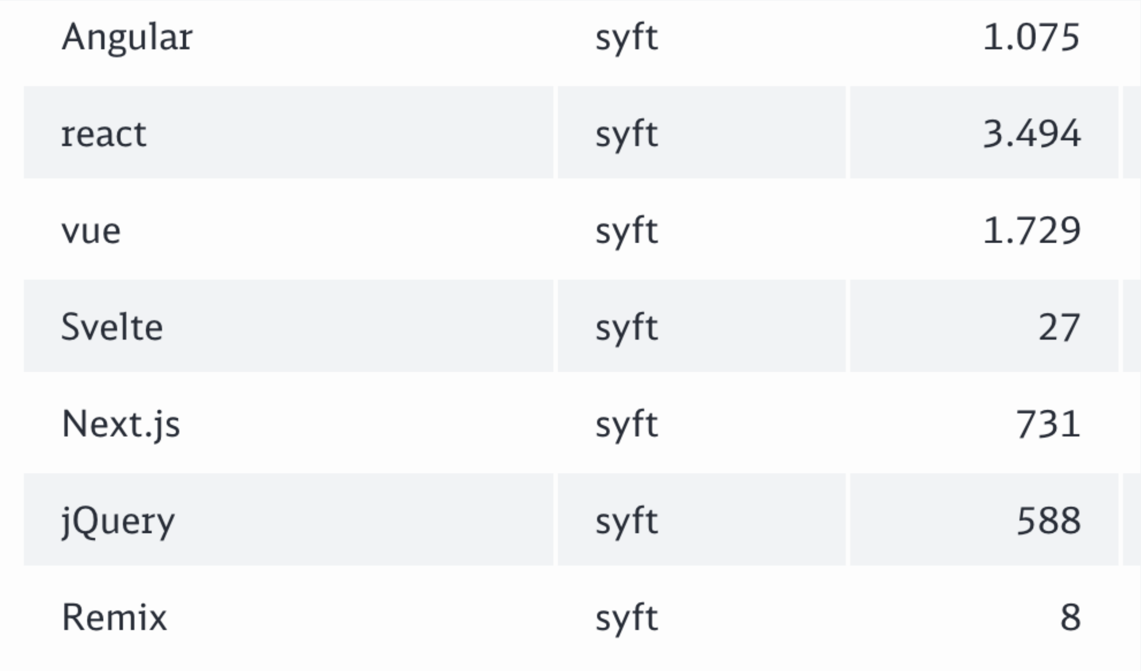
LICENSE	SPDX LICENSE	OSI APPROVED	NOT DEPRECATED	FSF LIBRE
MIT	MIT	✓	✓	✓
Apache-2.0	Apache-2.0	✓	✓	✓
ISC	ISC	✓	✓	✓
BSD-3-Clause	BSD-3-Clause	✓	✓	✓
BSD-2-Clause	BSD-2-Clause	✓	✓	✓

Compliance Suite: Modular Architecture Heavily Based on OSS



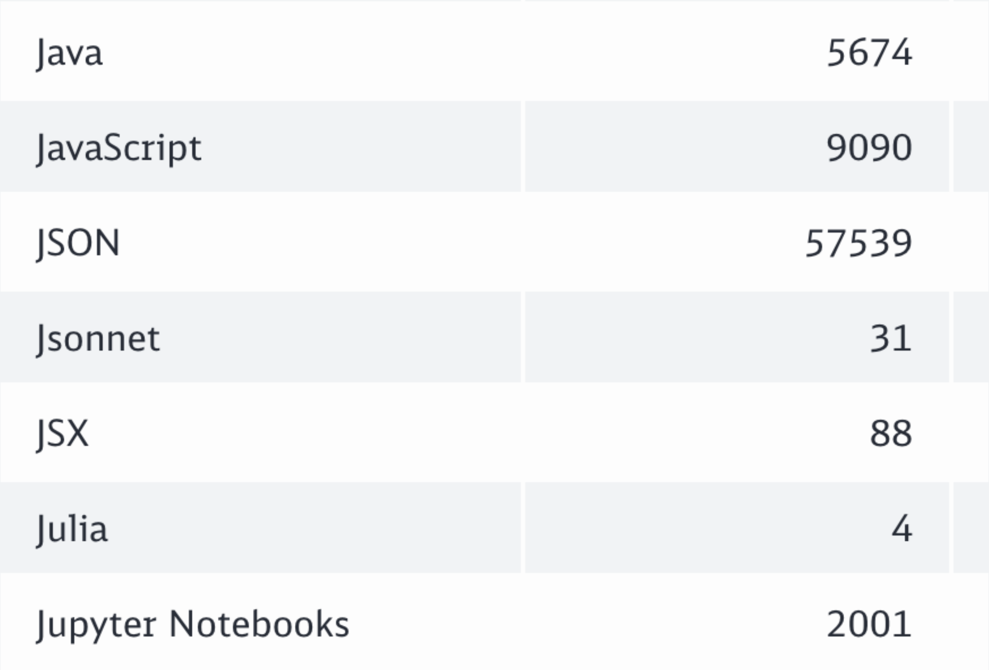
Overall Data Also Supports Technology Evaluation

Frontend Frameworks



Angular	syft	1.075
react	syft	3.494
vue	syft	1.729
Svelte	syft	27
Next.js	syft	731
jQuery	syft	588
Remix	syft	8

Programming Languages



Java	5674
JavaScript	9090
JSON	57539
Jsonnet	31
JSX	88
Julia	4
Jupyter Notebooks	2001

Central Oversight Makes Supply Chain Dimensions Transparent

79,943 SBOMs analyzed from Source and Build stages

1,855 enterprise applications covered by the analyzed SBOMs

104,904 packages in use, most of them Open Source

52,115 internal repositories covered

244 dependencies on average per code project

7.7% of our code projects contain the most-used dependency

Challenge: turn data into actionable items.

Last updated: January 2026

Tools Don’t Integrate Themselves – It’s People

To establish SBOMs and related tools/processes as a core methodology, we need to take all users with us:

- High adoption > perfection
- Pipelines and tools > dashboards
- Automation > manual processes
- Incremental improvements > Big bang release
- User feedback > top-down governance
- Open Source > Inner Source > Blackboxes

Concrete actions

- Heavy use of open source tools to which we contribute upstream
- All development, issue tracking and planning Inner Source, prospectively partly Open Source
- API and automation by default
- Regular open office hours for all users of the related tools and services: see new features, answer questions, provide direct feedback to developers
- Resulting findings are risk-based to not overload teams and help them prioritize

Take-aways and Call to Action

Main take-aways

1. SBOMs are a common methodology, beyond individual needs
2. Think big, implement incrementally
3. Modularity > monoliths
4. Delight your users

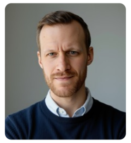
Call to Action

1. Internalize knowledge and skill about such core technologies
2. But collaborate and share in the open
3. Do not reinvent the wheel

Thank you!

Max Mehl
Open Source Governance & Strategy

max.mehl@deutschebahn.com
@mxmehl@mastodon.social
@mxmehl



Missing the strategy part?

Watch the recording of the previous session *“Software Supply Chain Strategy at Deutsche Bahn”*