
# Supply Chain Security, why should I care?

# Magnus Eklund

magnus-eklund
magnuseklund
Red Hat Developer magnus-eklund

**$ whoami**

**Specialist Solution Architect**, OpenShift team at Red Hat

**Main focus** on software supply chain security, platform engineering and application development.
~7 years at Red Hat, background as a java developer and software architect. Started as a middleware consultant at Red Hat

**Software supply chain security** - enabling developers and increasing security posture without impacting developer productivity. Make it more easy to do the right thing and introduce security early.

# Why software supply chain security?

## Software supply chain attacks: a matter of when, not if

Ransom paid but a mere fraction to the overall downtime and recovery costs of a data breach

**742%**

average annual increase in software supply chain attacks over the past 3 years[1]

**6 out of 7**

project vulnerabilities come from transitive dependencies[1]

**1 in 5**

data breaches are due to a software supply chain compromise[3]

**45%**

say software is released without going through security checks and/or testing[4].

[1] State of the Software Supply Chain | [3] Cost of a Data Breach 2022 - IBM Report | [4]Walk the line: GitOps and Shift left security - ESG Report |

# Securing Software Supply Chain – Why?

## Recognize that open source software has eaten the world

Security of open source software has to be a fundamental, ongoing aspect of the SDLC

**2 out of 3** organizations are already using OSS to augment internal development of new applications [1]

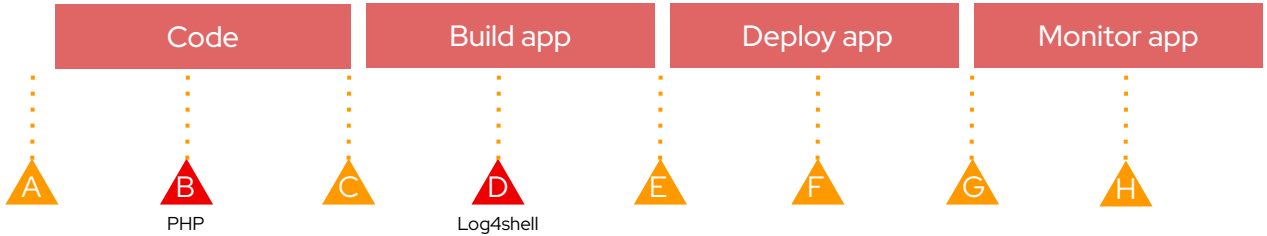
**600$^{est}$** number of open source components in any given software, in codebases that are widely open source based[2]

**90+%** of codebases contain open source components with no development activity or security fixes in two years[3]

[1] Better Together: DevOps and Open Source Go Hand in Hand - IDC Perspective, 2022 | [2] Open Source Security and Risk Analysis Report 2023 - Synopsys Report | [2] Open Source Security and Risk Analysis Report 2023

# The Rise of Software Supply Chain Attacks

## Real attacks in every stage - public, disruptive and costly




- **A** Submit unauthorized change
- **B** Compromise source repo
- **C** Build from modified source
- **D** Use compromised dependency
- **E** Compromise build process
- **F** Compromise package registry
- **G** Upload modified package
- **H** Use compromised package

Interesting fact: 13% of downloads of Log4j remained vulnerable 3 years later despite the availability of a fixed version.

State of the Software Supply Chain

SLSA Threat Overview

# The Supply Chain Security space is relatively young

## Recent activities have highlighted its importance

*"The Cyber Resilience Act is the first ever EU-wide legislation of its kind. It introduces common cybersecurity rules for manufacturers and developers of products with digital elements, covering both hardware and software/.../throughout its lifecycle"*

### DevSecOps Movement

Evolution of the DevOps movement which also includes a security component where there is increased involvement from security teams and methodologies

### Government Regulations

Recent actions by governments across the world have began to mandate certain steps be implemented in order to utilize software produced or utilized from external sources

### Initiatives to Drive Increased Security

Organizations are looking for additional methods for securing the content they produce and use

Sources:

- Cyber Resilience Act: MEPs adopt plans to boost security of digital products | News | European Parliament
- Cyber Resilience Act - Questions and Answers

# Challenges in Software Supply Chain Security

**Internal Tampering Risks:** Within an organization, unauthorized changes to code or build processes can introduce vulnerabilities or malicious elements into the software

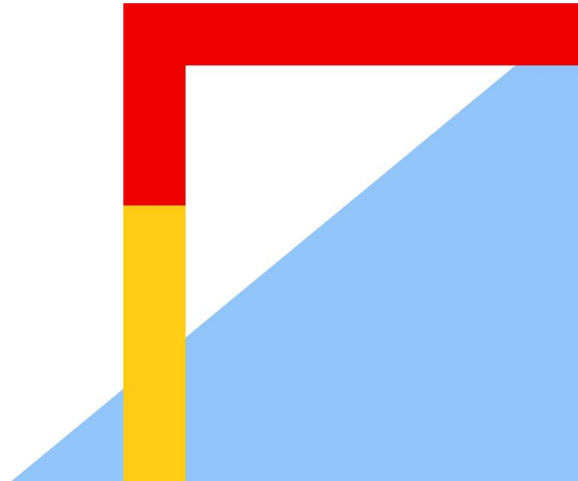
**Complexity of dependencies:** Modern software relies on numerous external and internal components. Managing and securing each dependency throughout the development process is challenging.

**Integrity Verification of the software:** Ensuring that every component remains unaltered and trustworthy across all stages of the supply chain requires robust verification mechanisms

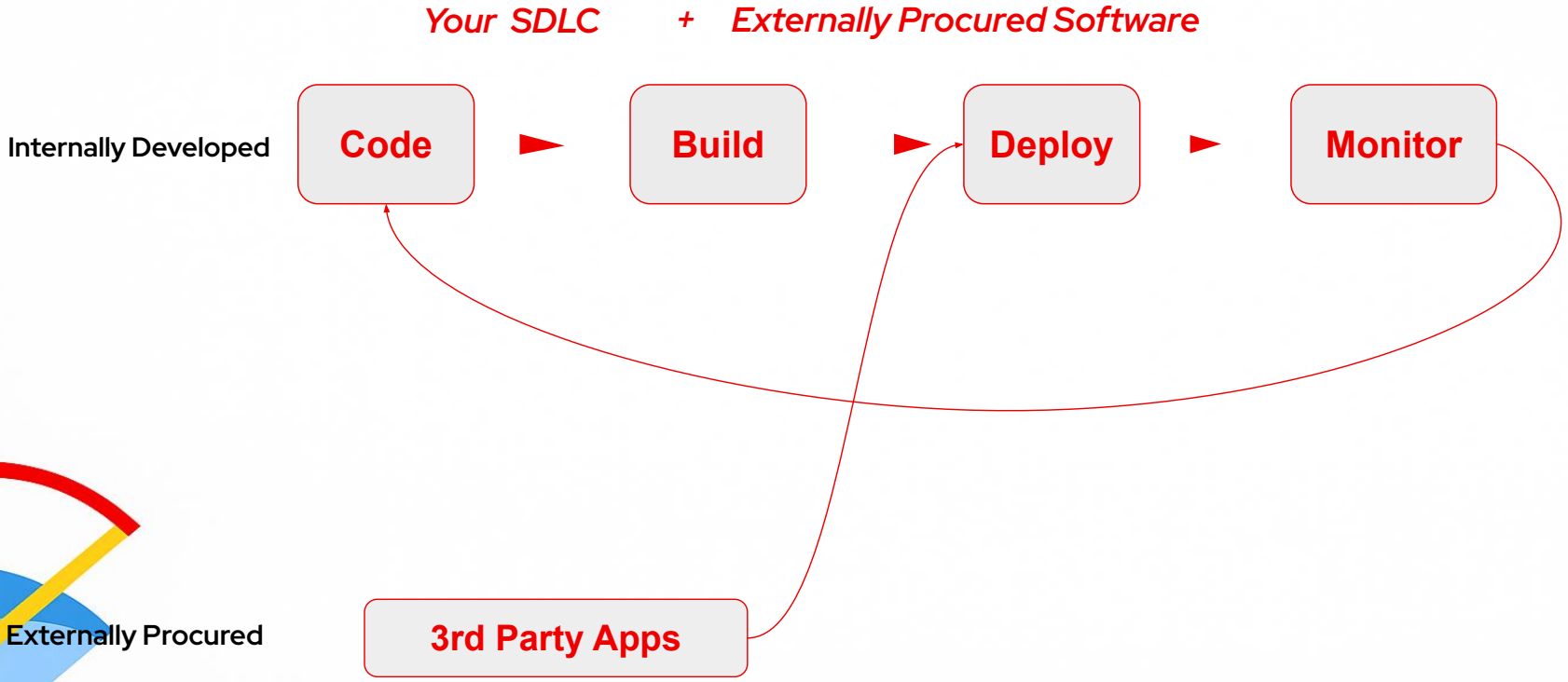
**Lack of Transparency in component origins:** Without clear visibility into both external sources and internal processes, detecting and preventing security issues becomes difficult.

**Insufficient Controls in the SDLF:** If the systems and tools used to assemble software aren't secure, they become potential targets for attackers aiming to compromise the final product,

# Securing the Software Supply Chain - how?

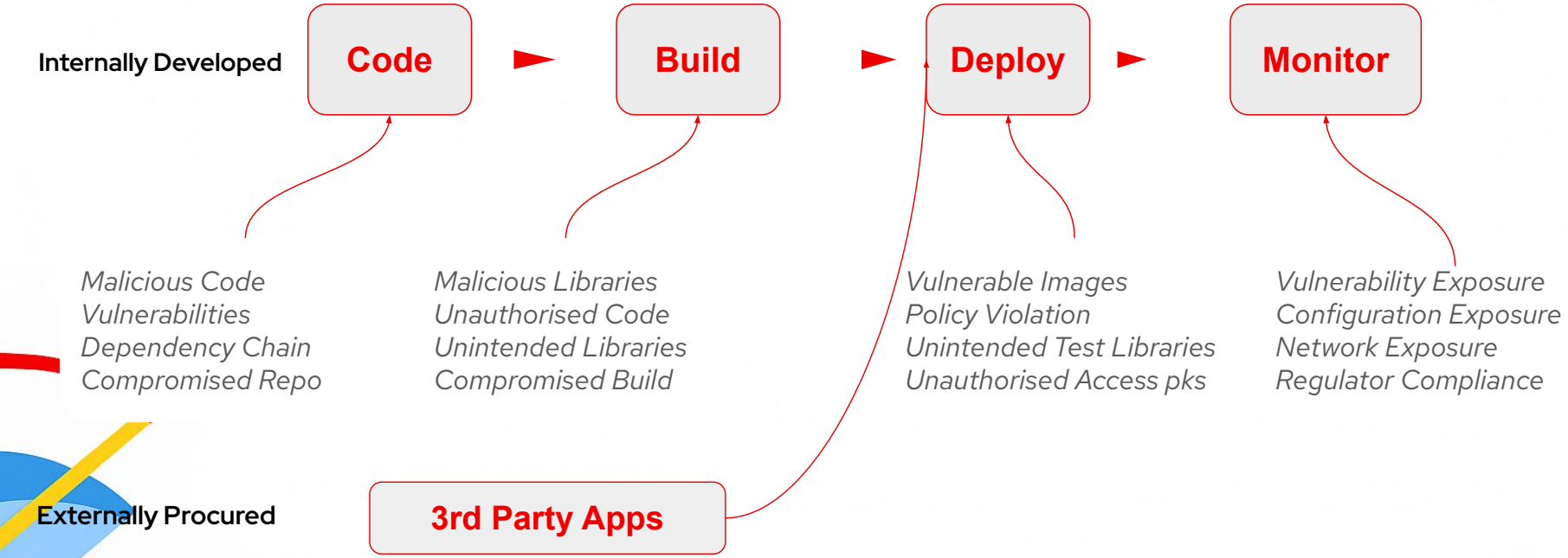


# The Software Supply Chain

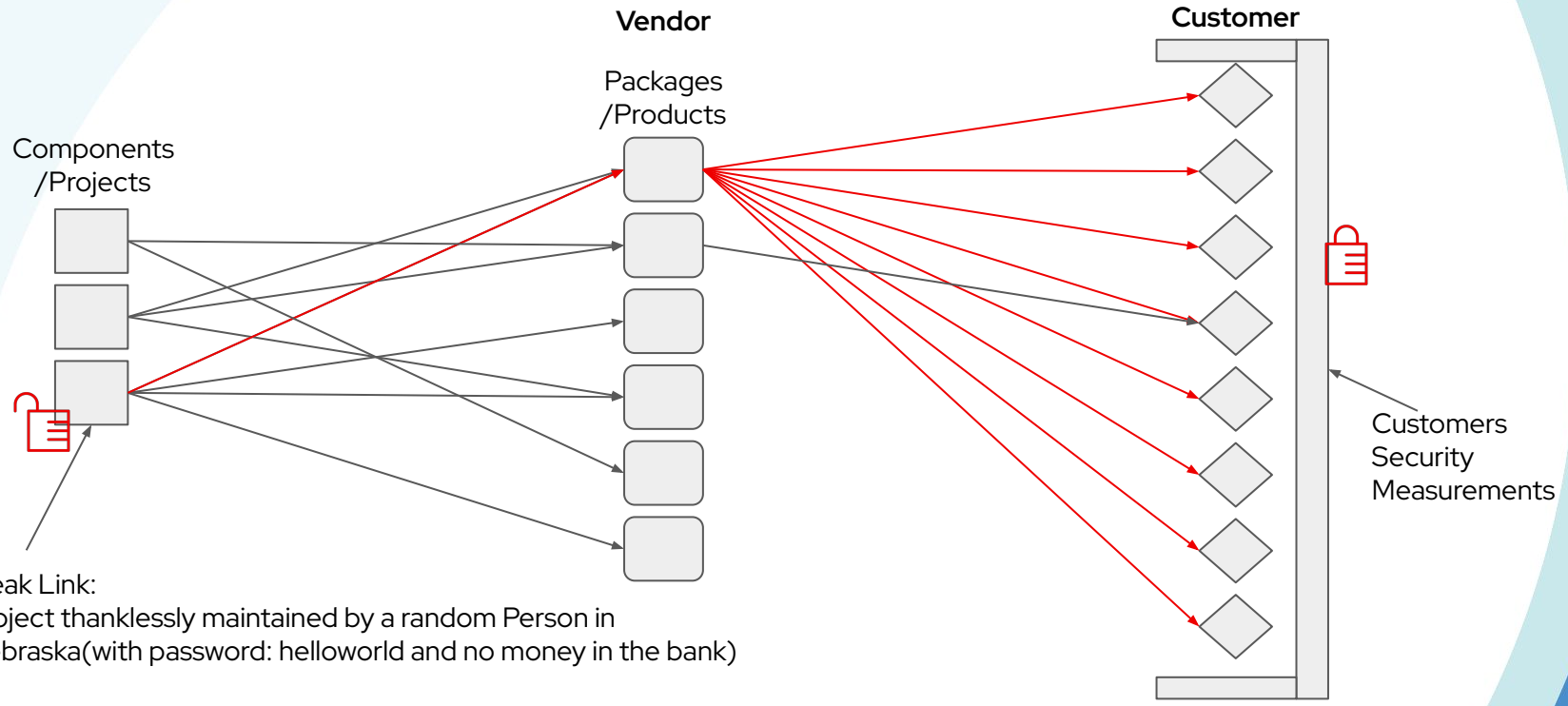


# Potential Attack Vectors

***Multiple threats and Large attack Surface***



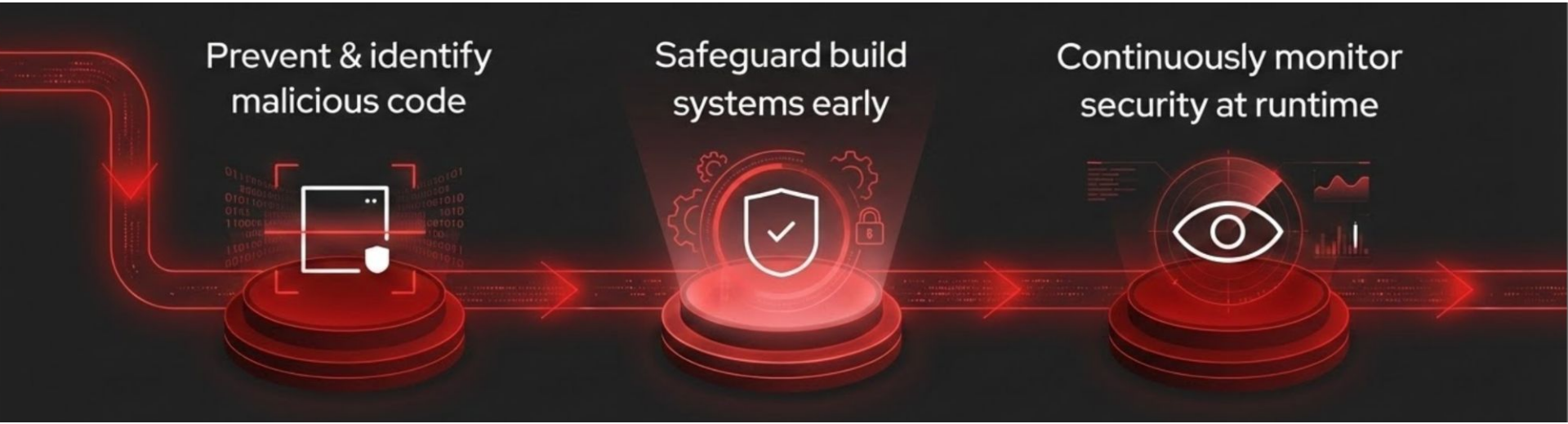
# Why is the Supply Chain interesting for attackers?




High utilization > High reward

# Securing the software supply chain

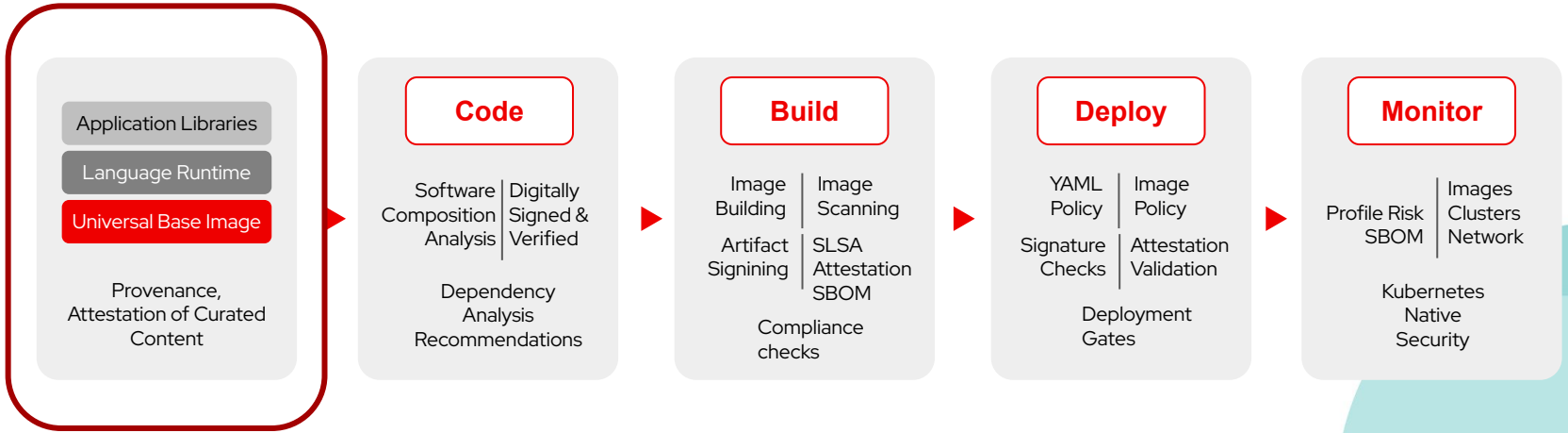
Software supply chain security considerations for the software development lifecycle




Get control and understanding, by adding new capabilities

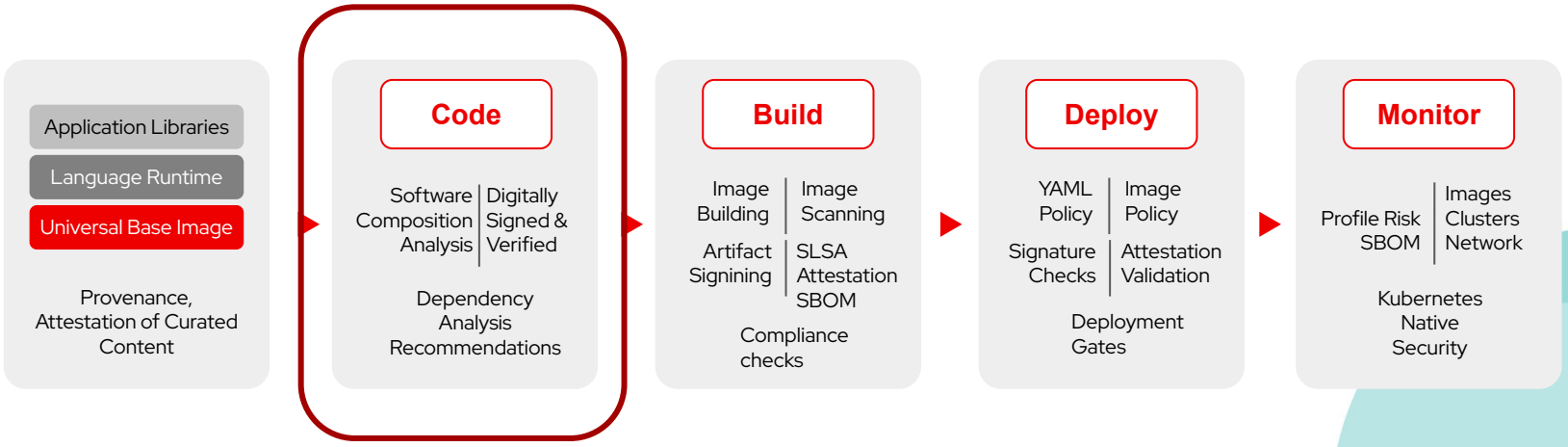
# Start by using Trusted Content

- Your code should be based on trusted images, libraries and runtimes
- Store provided SBOMs (Software Bills of Materials) and attestations, so you know “what’s in the box”, just in case.



# Give your developers the right tools

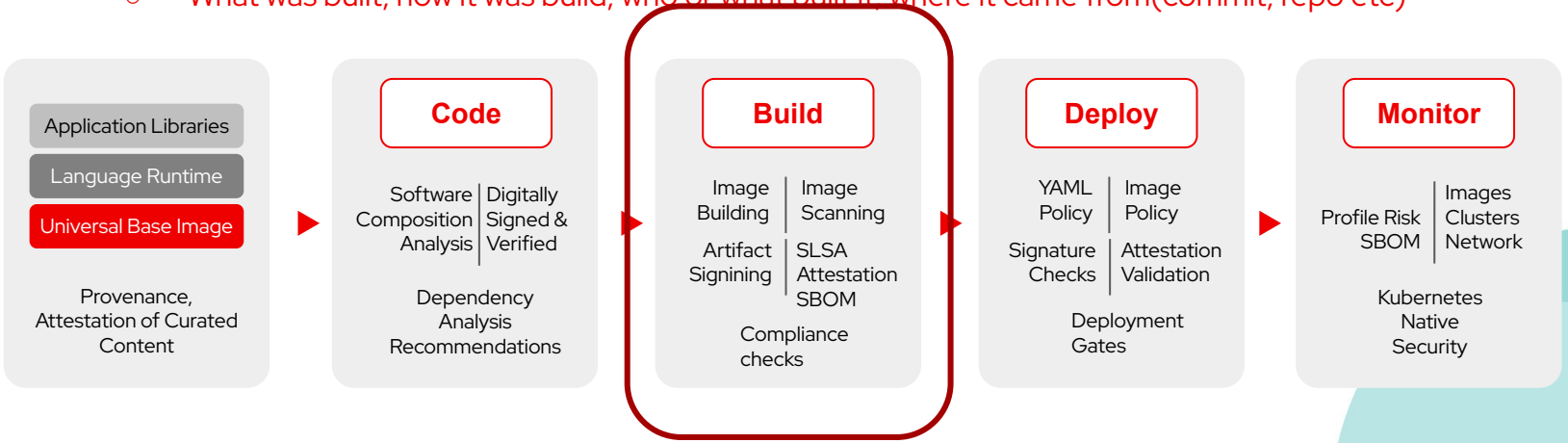
- Provide automated composition & dependency analysis and recommendations
- Make signing any component (Source Code, YAML & config files, Infrastructure-as-Code) easy, without cumbersome additional steps
- Provide developers with curated templates (based on Trusted Content) to create new components
- Give developers confidence in their code security and used dependencies, ***but don't get in their way. *)***




*) Developers should do what they love and do best - thrill your customers with the next great application. They shouldn't have to spend their time with additional security tooling that "gets in their way" and slows down their overall performance.

# Augment and secure your build process (CI)

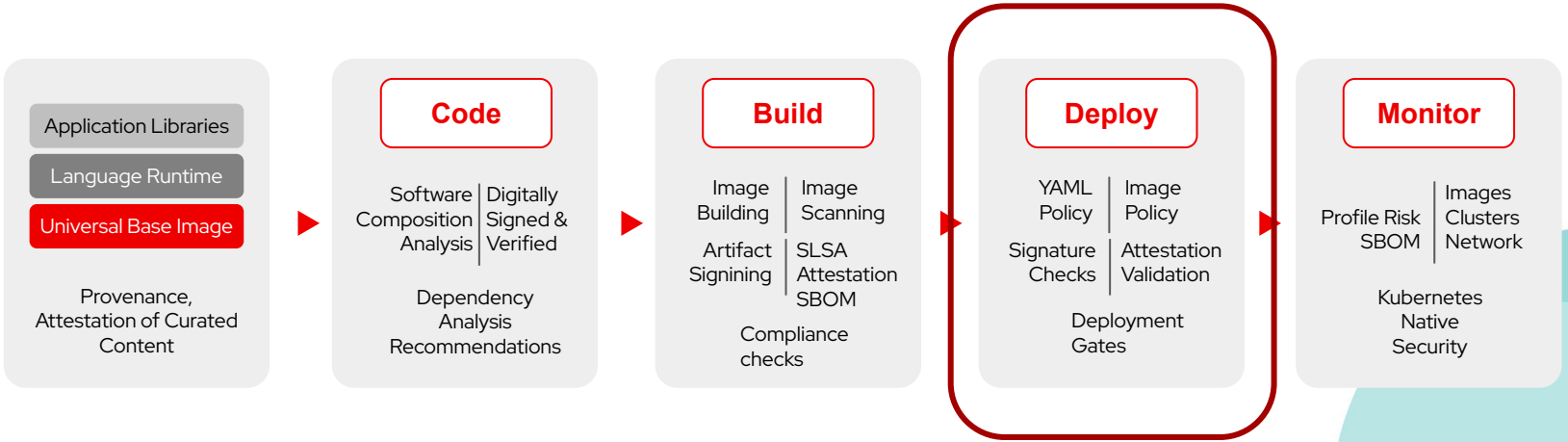
- Only build from signed and verified source code
- Only use signed and verified (base-)images, add image scanning
- Automatically sign your built artifacts (images, binaries,...)
- Create SBOMs in the build process so you know what is used where (including dependencies)
- Attest the integrity of your artifacts and build pipeline (SLSA [1])
  - What was built, how it was build, who or what built it, where it came from(commit, repo etc)




[1] SLSA ("Supply-chain Levels for Software Artifacts") Specification: https://slsa.dev

# Augment and secure your deployment process (CD)

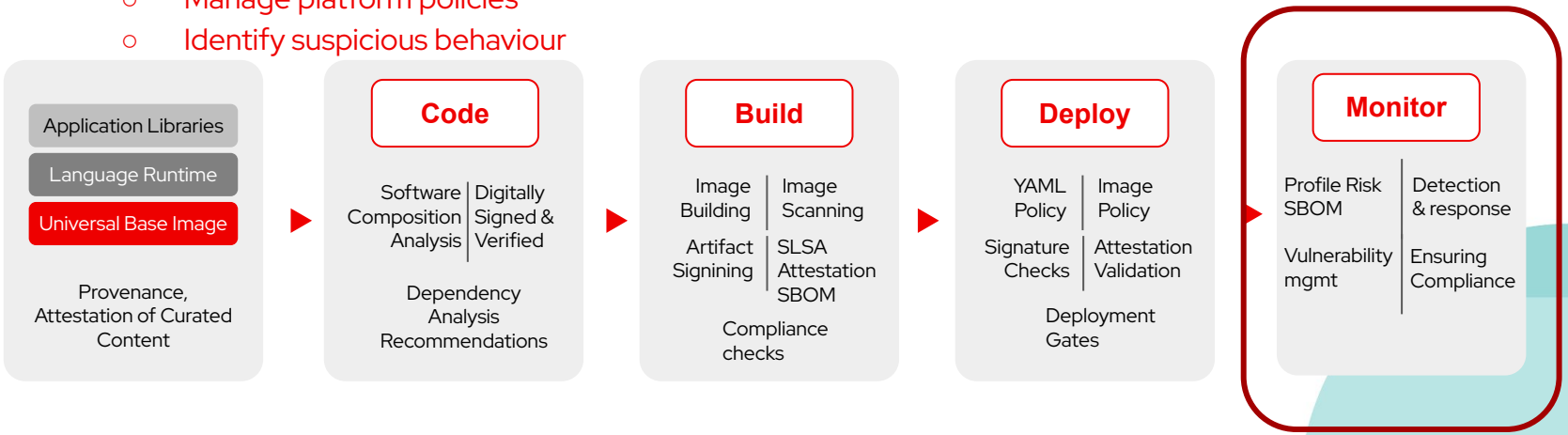
- Automatically check & verify artifacts and attestations as part of your pipeline
- Only deploy signed and verified artifacts
- Attest the same signature (the same artifact) has been promoted across environments
- Run policy checks - verify compliance towards internal policies




[1] SLSA ("Supply-chain Levels for Software Artifacts") Specification: https://slsa.dev

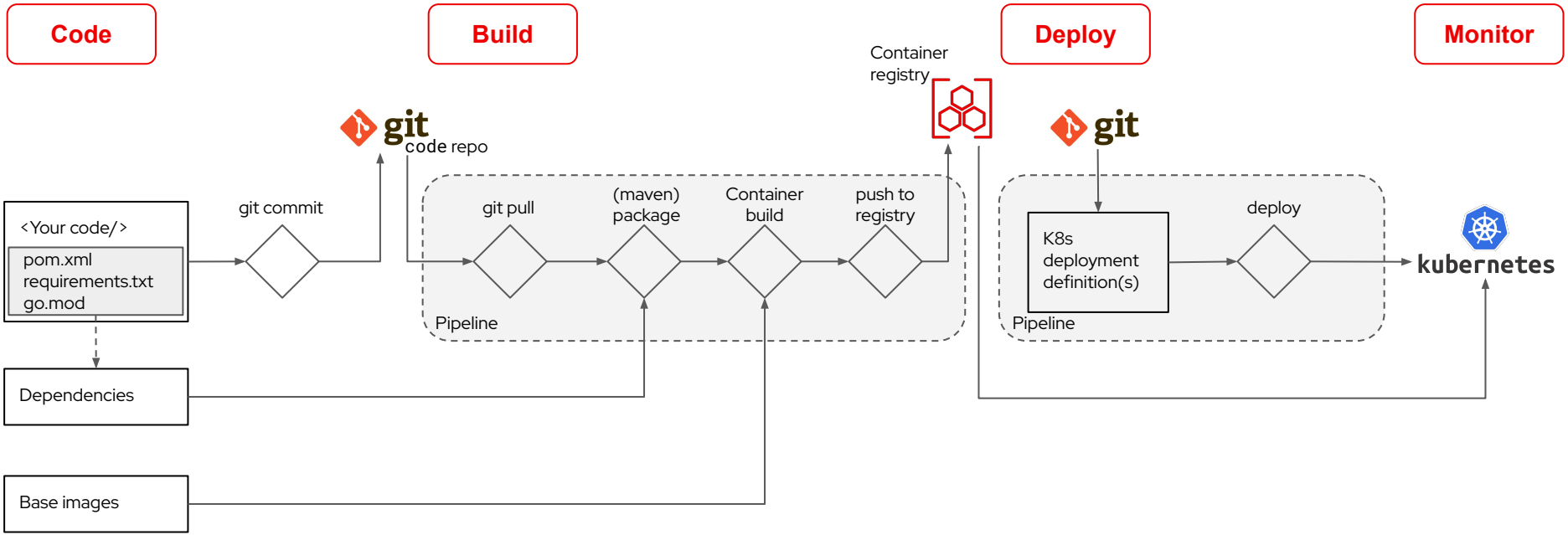
# Manage your Security Posture and monitor your platform

- Manage your own and 3rd party SBOMs and VEX-Files [2]
  - In case of a CVE, know the impact and blast radius across your applications
  - Know your security posture, risks and report
  - Manage your risk profile over time
- Monitor your platform security
  - Manage platform policies
  - Identify suspicious behaviour

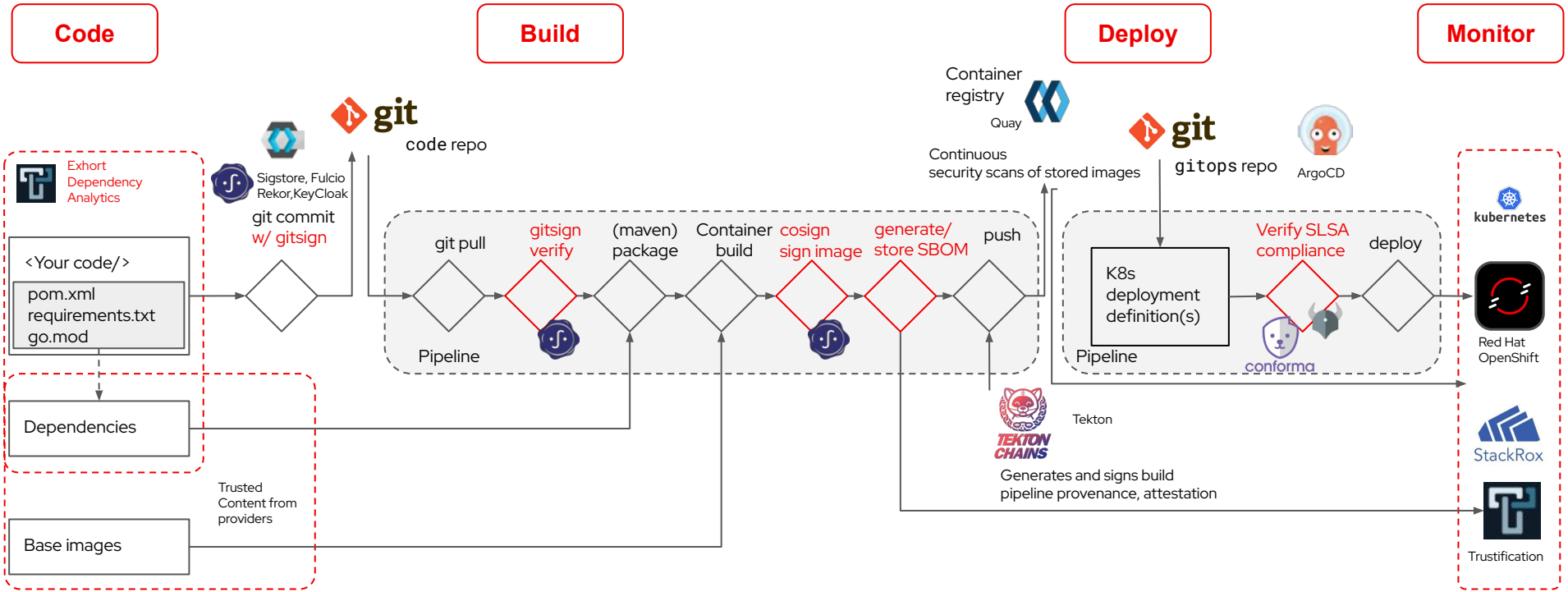



[2] VEX (Vulnerability Exploitability eXchange) is a standardized format by which vendors can publish vulnerability information pertaining their products.

# A generic development process

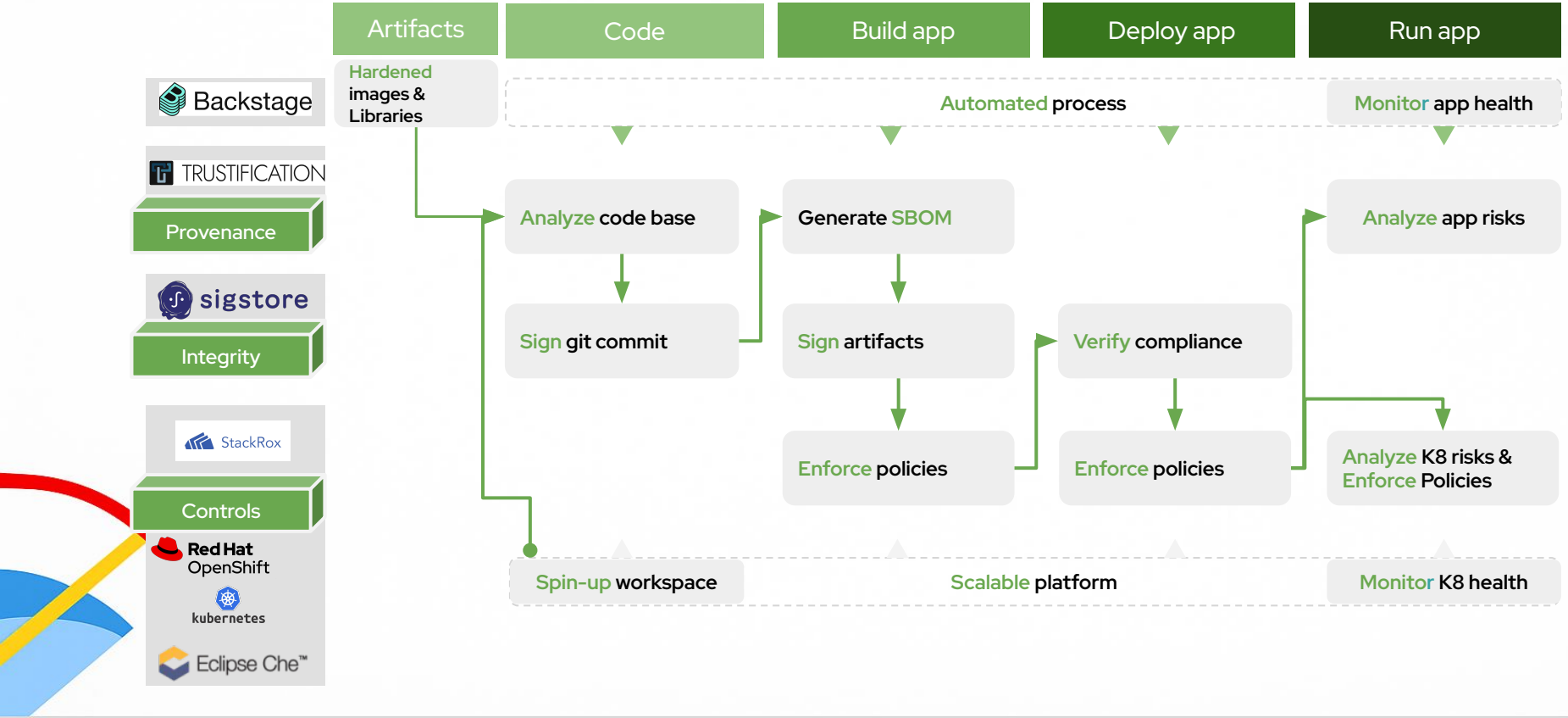


# A security-augmented development process



SLSA Lvl 3 ✅

# Achieve Compliance with a Trusted Software Supply Chain



# Demo

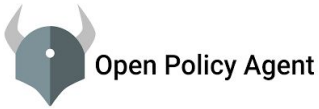
# Augment and secure your deployment process

IExamples of foundations and projects to use for securing SDLC - from end to end















# What's next ?

*We are at an inflection point:*

- AI can now **find, explain, and fix vulnerabilities**
- SDLC is becoming **agent-driven from agent-assisted**
- The **agent ecosystem itself is a new attack surface**

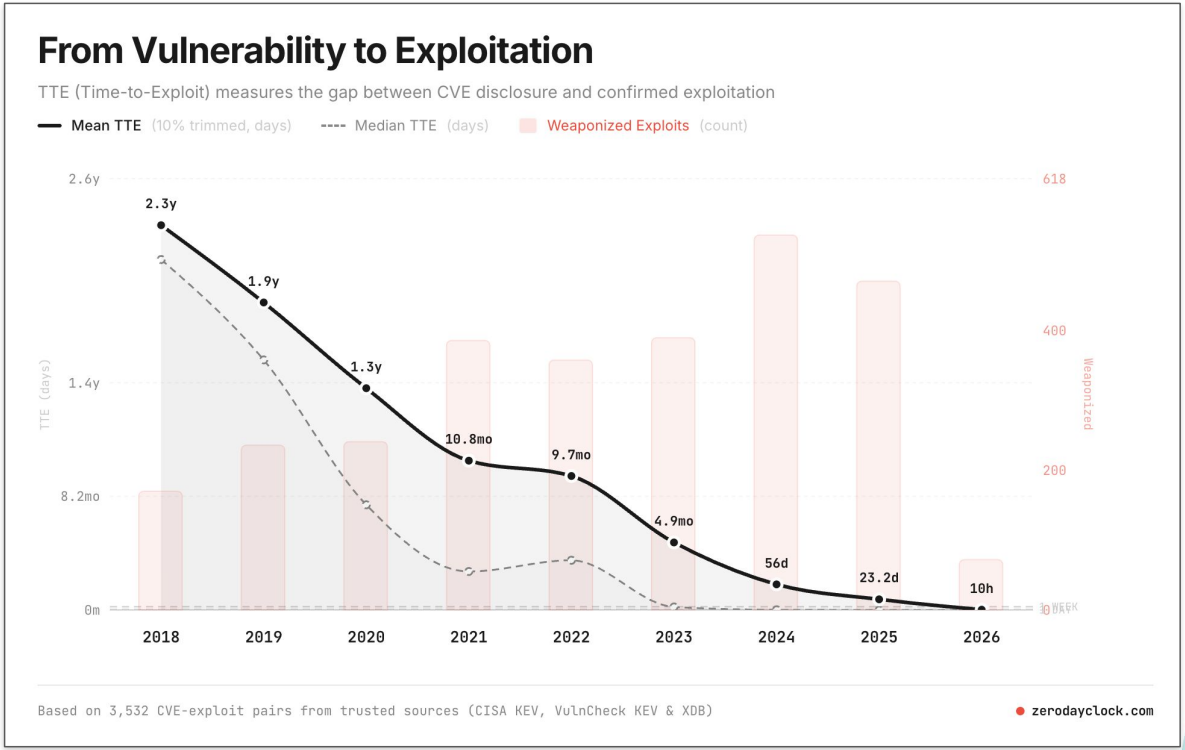
**Implication:**

- 10x code velocity → 10x risk propagation
- **Security must operate at machine speed or fail - Agents as attack vector**
- Security must evolve from static validation → continuous, intelligent governance

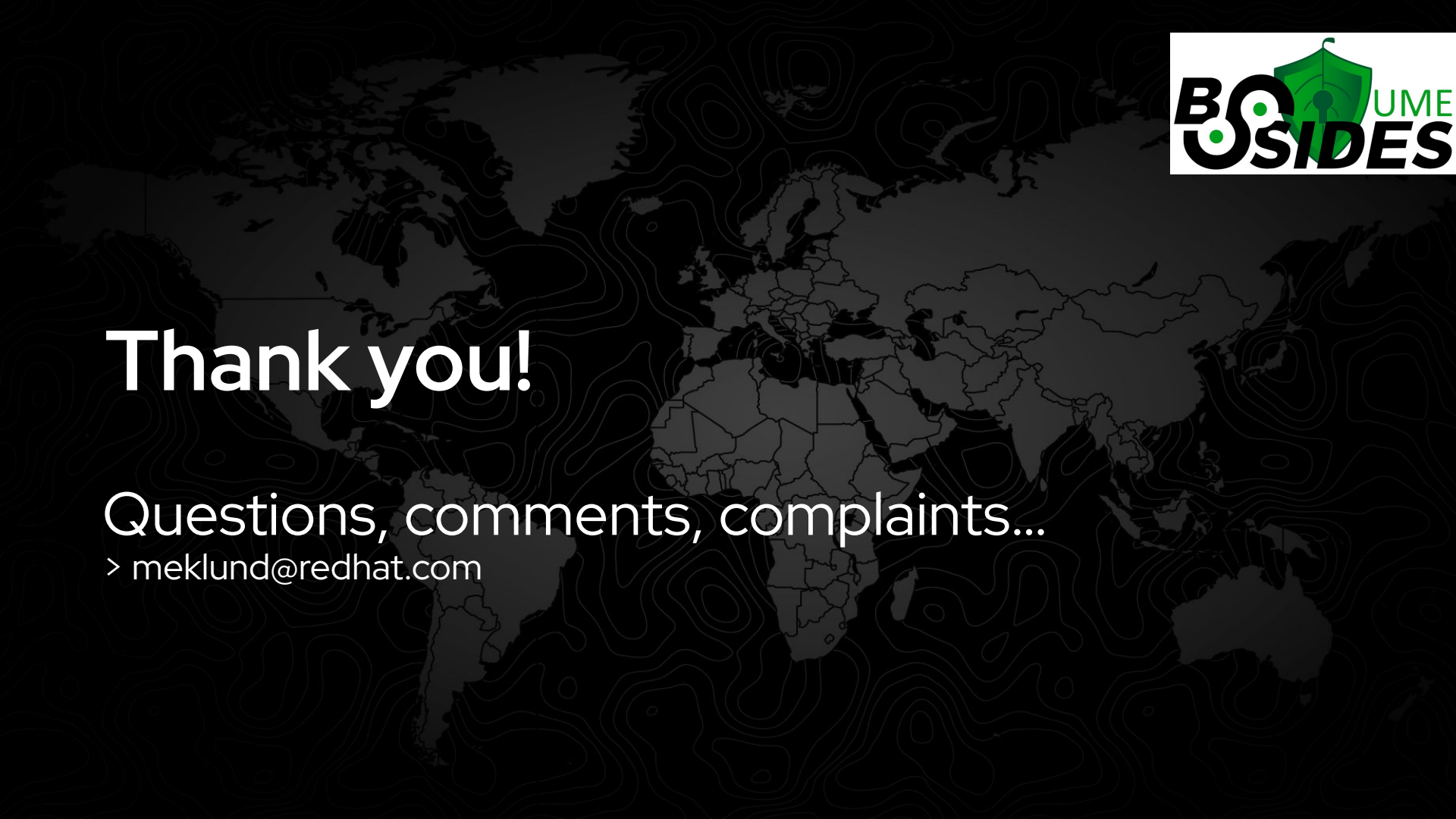
**Conclusion:**

The market is converging on agentic development plus agentic security.
Trust, provenance, and policy are becoming more important, not less.

# Shrinking window to address Supply Chain




Source: The "AI Vulnerability Storm": Building a Mythos-ready Security Program, Zero Day Clock



# Thank you!

Questions, comments, complaints...

> meklund@redhat.com