Software Supply Chain Security - Why should I care ?

Red Hat Trusted Software Supply Chain (TSSC)

Markus Nagel
Principal Technical Marketing Manager
Hybrid Platforms Business Unit (HPBU)
mnagel@redhat.com

Red Hat

Agenda

1. Software Supply Chain Security - in Layman's Terms
2. Attacking the Software Supply Chain
3. From a Software Supply Chain to a TRUSTED Software Supply Chain
4. Red Hat Trusted Software Supply Chain - to the rescue!
5. Quick Demo
6. Q&A / Discussion

Understanding Software Supply Chain Security

Red Hat

Imagine you're a chef, preparing a complex meal for your guests

- The meal is complex, requires many ingredients and isn't exactly a one-pot stew

- Ingredients are fresh and safe
- From trusted sources
- Have a certificate of origin
- ...and also an ingredients list themselves


Ingredients: Enriched Corn Meal (Corn Meal, Ferrous Sulfate, Niacin, Thiamin Mononitrate, Riboflavin, Folic Acid), Vegetable Oil (Corn, Canola, and/or Sunflower Oil), Cheese Seasoning (Whey, Cheddar Cheese [Milk, Cheese Cultures, Salt, Enzymes], Canola Oil, Maltodextrin [Made from Corn], Natural and Artificial Flavors, Salt, Whey Protein Concentrate, Monosodium Glutamate, Lactic Acid, Citric Acid, Artificial Color [Yellow 6]), and Salt.
CONTAINS MILK INGREDIENTS

- Your kitchen is clean, so are your pots & pans
- You trust your kitchen helpers
- Nobody tampers with your ingredients
- Nobody tampers with the meal when it goes from the pot to the oven, then to the plates, then to the table

Imagine you're a chef, preparing a complex meal for your guests

You want to verify the quality and safety of every ingredient and every step in your cooking process to ensure the final meal is delicious and safe to eat...

...and your patrons are enjoying what left your kitchen.

In the software world,...

- You are building (and maintaining) complex software products
- Containing internal and external dependencies, like open-source libraries

- Applications are going through a chain of tasks from code to build (assembly) and finally deployment

In the software world,...

- You are building (and maintaining) complex software products
- Containing internal and external dependencies, like open-source libraries
- Can you trust your “ingredients”?

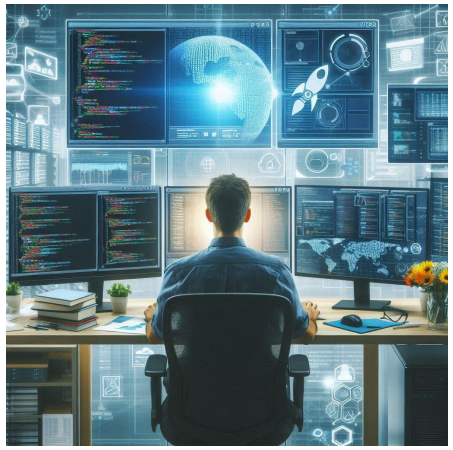
- Applications are going through a chain of tasks from code to build (assembly) and finally deployment
- Can you trust your “kitchen helpers”?
- Is your “kitchen” clean?
- Does nobody tamper with your “meal” (your artifacts), the “ingredients” (dependencies, code) or your pots & pans (CI/CD tasks)?
- Can you vouch for the integrity of what’s on the plate? Is it what left your pot?

In the software world,...

If any part of the development process is compromised, malicious code could be introduced, or critical assets could be altered without detection.

Verifying the integrity of assets across the entire supply chain—from development to deployment—is critical to maintain the overall security of the software.

Challenges in **Software Supply Chain Security**



1. **Complexity of Ingredients (Dependencies)**: Modern software relies on numerous external and internal components, similar to a recipe requiring many ingredients. Managing and securing each dependency throughout the development process is challenging.
2. **Internal Tampering Risks in the Kitchen**: Within an organization, unauthorized changes to code or build processes can introduce vulnerabilities or malicious elements into the software, just as someone tampering with ingredients in the kitchen can ruin the dish.
3. **Integrity Verification of the Meal**: Ensuring that every component remains unaltered and trustworthy across all stages of the supply chain requires robust verification mechanisms, like taste-testing at different stages.
4. **Lack of Transparency in Ingredient Origins**: Without clear visibility into both external sources and internal processes, detecting and preventing security issues becomes difficult.
5. **Insufficient Controls in the Cooking Process**: If the systems and tools used to assemble software aren't secure, they become potential targets for attackers aiming to compromise the final product, just as an unsecure kitchen might lead to contaminated food.

Addressing these Challenges

Implementing Software Bill of Materials (SBOMs):

- **What is an SBOM?** An SBOM is like a detailed ingredients list for your software, documenting every component and dependency used, including those developed internally.
- **Why is it important?** If a vulnerability is found or if tampering is suspected, an SBOM allows you to quickly identify affected components and take corrective action.
- **How it helps:** It enhances transparency and traceability across both external and internal supply chains, aiding in risk management and integrity verification.


```
{
    "bom-ref": "pkg:maven/org.apache.logging.log4j/log4j-to-slf4j@2.11.2?package-id=30e582d320883b6c",
    "type": "library",
    "group": "org.apache.logging.log4j",
    "name": "log4j-to-slf4j",
    "version": "2.11.2",
    "licenses": [
        {
            "license": {
                "name": "https://www.apache.org/licenses/LICENSE-2.0.txt"
            }
        }
    ],
    "cpe": "cpe:2.3:a:apache:log4j-to-slf4j:2.11.2:*:*:*:*:*:*:*",
    "purl": "pkg:maven/org.apache.logging.log4j/log4j-to-slf4j@2.11.2",
    "externalReferences": [
        {
            "url": "",
            "hashes": [
                {
                    "alg": "SHA-1",
                    "content": "6d37bf7b046c0ce2669f26b99365a2cfa45c4c18"
                }
            ],
            "type": "build-meta"
        }
    ],
    "properties": [
```


Red Hat

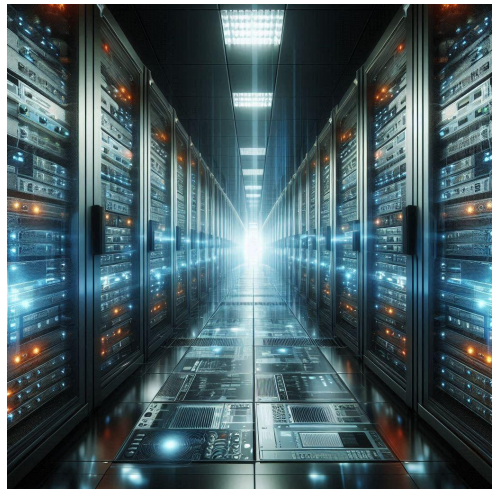
Addressing these Challenges

Adopting SLSA Provenance and Levels:

- **What is SLSA?** *Supply Chain Levels for Software Artifacts (SLSA)* [1] is a security framework that provides guidelines to protect the **software build process** from tampering and ensure artifact integrity.
- **Understanding Provenance:** In this context, provenance refers to metadata that records the origin, history, and build process of software components, both external and internal.
- **SLSA Levels Explained**:
  - **Level 1 (Basic Provenance):** The build process is fully scripted and automated, reducing manual errors and making the process more transparent.
  - **Level 2 (Authenticated Provenance):** Build metadata is cryptographically signed, ensuring that provenance data is authentic and hasn't been altered.
  - **Level 3 (Certified Provenance):** Builds occur in a tamper-resistant environment, protecting against unauthorized changes during the build process.
  - **Level 4 (Highest Integrity):** Implements rigorous security measures, including two-person reviews, hermetic builds (self-contained with no external dependencies), and reproducible builds (allowing others to produce the same build independently).
- **How it helps:** Following SLSA levels strengthens both external and internal supply chain security, making it difficult for attackers to introduce malicious code at any point in the process.

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

Addressing these Challenges



Securing the Internal Build Environment:

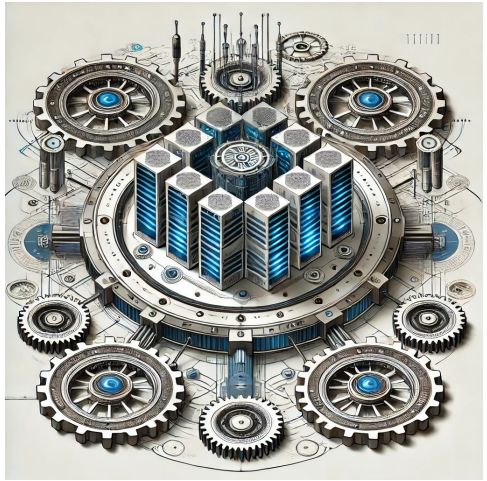
- **Isolated Build Systems:** Use secure, isolated environments for building and assembling software to prevent unauthorized access or tampering.
- **Access Controls and Authentication:** Implement strict access controls, ensuring that only authorized personnel can modify build processes or code repositories.
- **Monitoring and Logging:** Continuously monitor build environments and maintain detailed logs to detect any suspicious activities or changes.

Addressing these Challenges

End-to-End Integrity Verification:

- **Cryptographic Hashing:** Use cryptographic hashes to verify that components remain unchanged throughout the build and deployment process.
- **Digital Signatures:** Sign code and artifacts to authenticate their origin and integrity at every stage of the supply chain.
- **Automated Testing and Validation:** Incorporate security checks and automated tests within the build pipeline to detect anomalies or unauthorized modifications promptly.

Addressing these Challenges



Establishing Trust and Verification Mechanisms:

- **Immutable Infrastructure:** Use infrastructure that prevents changes once deployed, reducing the risk of tampering after deployment.
- **Secure Artifact Repositories:** Store build artifacts in secure repositories with controlled access and integrity verification mechanisms.
- **Chain of Custody Documentation:** Maintain records of all individuals and processes that interact with each component, enhancing traceability and accountability.

Why it matters

Why Supply Chain Security Matters

- **Preventing Tampering:** Protects against unauthorized modifications to code or build processes, whether malicious or accidental—just as kitchen security prevents food tampering.
- **Ensuring Integrity:** Verifies that all components and artifacts remain unaltered from development through deployment, similar to ensuring a dish is prepared correctly from start to finish.
- **Compliance and Trust:** Meets regulatory requirements and builds trust with customers by demonstrating a commitment to security and quality.
- **Reducing Risk:** Minimizes the potential for security breaches that could lead to data loss, financial damage, or reputational harm, much like avoiding food poisoning or customer dissatisfaction.

Software Supply Chain Attacks

Red Hat

Increased security risks for application development

Growing concerns that open source is vulnerable to exploit and attack

742%

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

84%

of open source scanned codebases contained at least one vulnerability[2]

31%

of organizations are not monitoring these open source dependencies[3]

The increase in the number of applications, systems and environments causes organizations to have larger exposure to malicious attacks and compels them to look for strategies that reduce the surface area of attack.

[1] State of the Software Supply Chain | [2] Open Source Security and Risk Analysis

A quick search reveals (11-March-25)

We need to strengthen our Security Posture End-to-End

Notable Recent Attacks

SolarTrade Incident (2025)

A logistics management platform used by over 500 global retailers was compromised when attackers injected malicious code into a routine software update. This breach led to unauthorized access to customer payment information and disrupted supply chain operations for months[1].

MedTech Vulnerability Exploitation (2025)

Hackers targeted a major medical device manufacturer's firmware update system, injecting malware into pacemakers and insulin pumps. This attack raised significant concerns over patient safety and regulatory compliance[1].

GreenGrid Cyberattack (2025)

A renewable energy supply chain provider was breached through a third-party supplier's compromised API, leading to grid disruptions in multiple regions. This incident highlighted the critical nature of securing energy infrastructure[1].

XZ Utils Backdoor (March 2024)

A sophisticated backdoor was implanted in XZ Utils, a popular compression utility widely used in Linux distributions. This attack could have potentially compromised millions of SSH servers globally[2].

CrowdStrike Outage (July 2024)

A flawed update to CrowdStrike's cloud-based security software triggered malfunctions in 8.5 million Microsoft Windows devices, causing the largest global IT outage in history. This incident disrupted airlines, banks, healthcare systems, broadcasters, and payment infrastructure worldwide, resulting in estimated losses of $5 billion[1][2]. The outage highlighted the systemic risks of supply chain dependencies and the potential for cascading failures when critical providers experience issues.

3CX Attack (Early 2024)

The desktop applications of 3CX, a widely-used communications software provider, were compromised in a supply chain attack. The malicious code was signed with valid 3CX certificates, suggesting a compromised build environment. This allowed attackers to execute malicious activities within victims' environments[4].

JetBrains Vulnerability

A critical vulnerability in JetBrains TeamCity servers affected over 3,000 on-premises servers out of an estimated 30,000 JetBrains customers. This vulnerability potentially enabled remote code execution and administrative control, posing significant risks to organizations relying on JetBrains software[4].

Software Supply Chain Attacks and pending legislation

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

- Government willingness to enforce and fine executives ignoring SSC
  - SEC fines SolarWinds and CISO for concealing vulnerabilities
  - Log4J vulnerability
- US Executive Order on Improving the Nation's Cybersecurity
- US Executive Order 14017 - America's Supply Chains
- US Executive order 14018 Improving the Nation's Cybersecurity
- EU Network and Information Security 2 Directive
- EU CRA (Cyber Resilience Act)

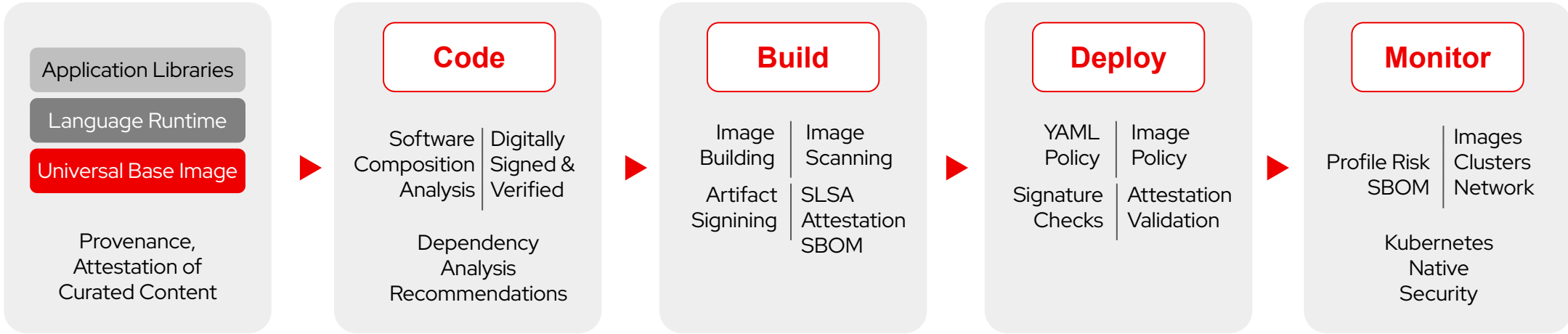
What is a TRUSTED Software Supply Chain

Red Hat

The Software Supply Chain

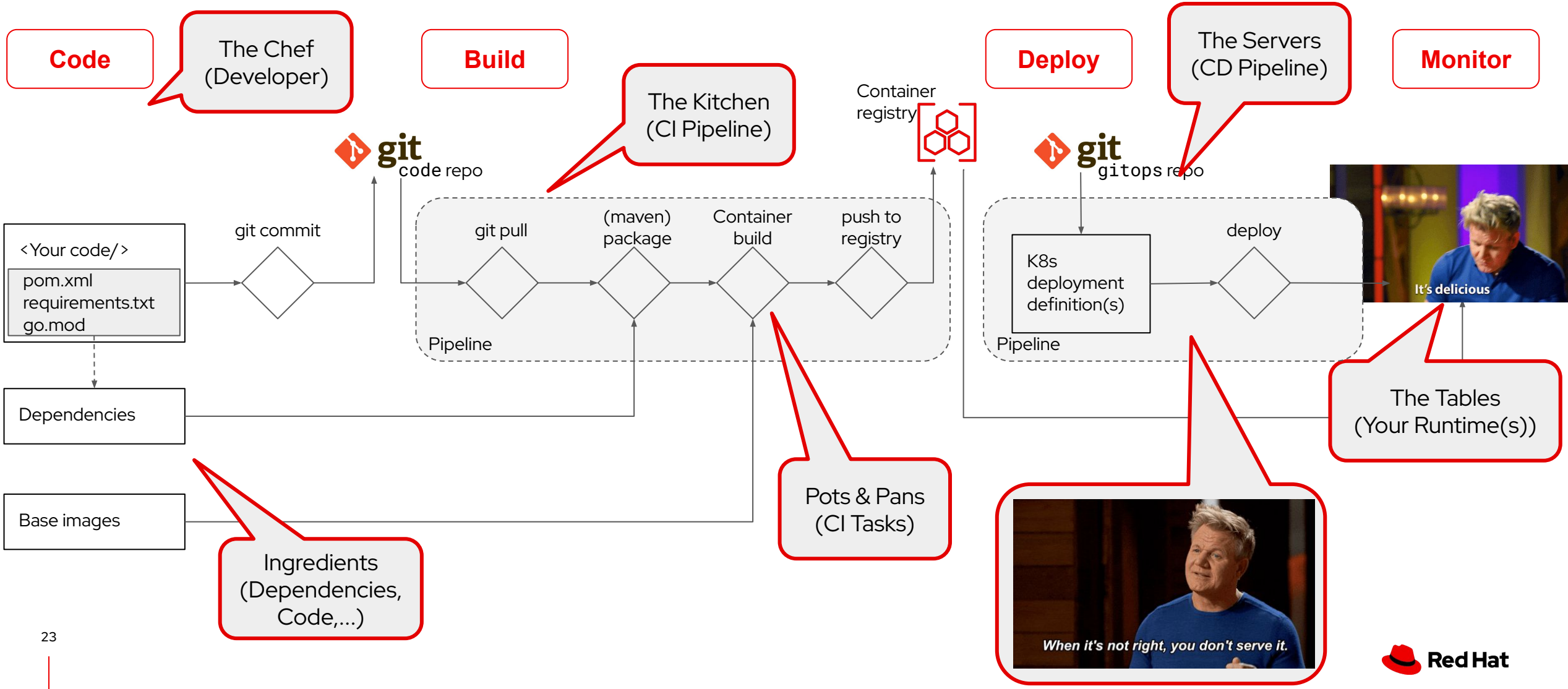
Red Hat

The Software Supply Chain



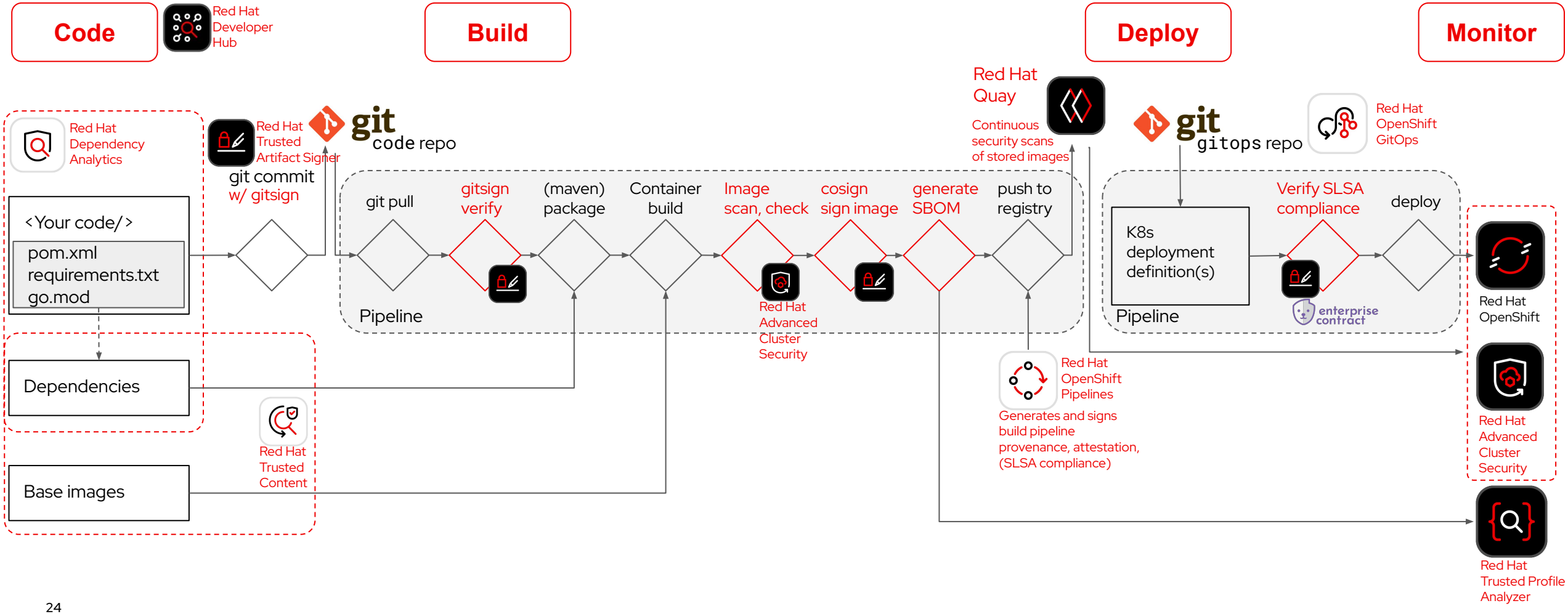
Red Hat

A generic development process



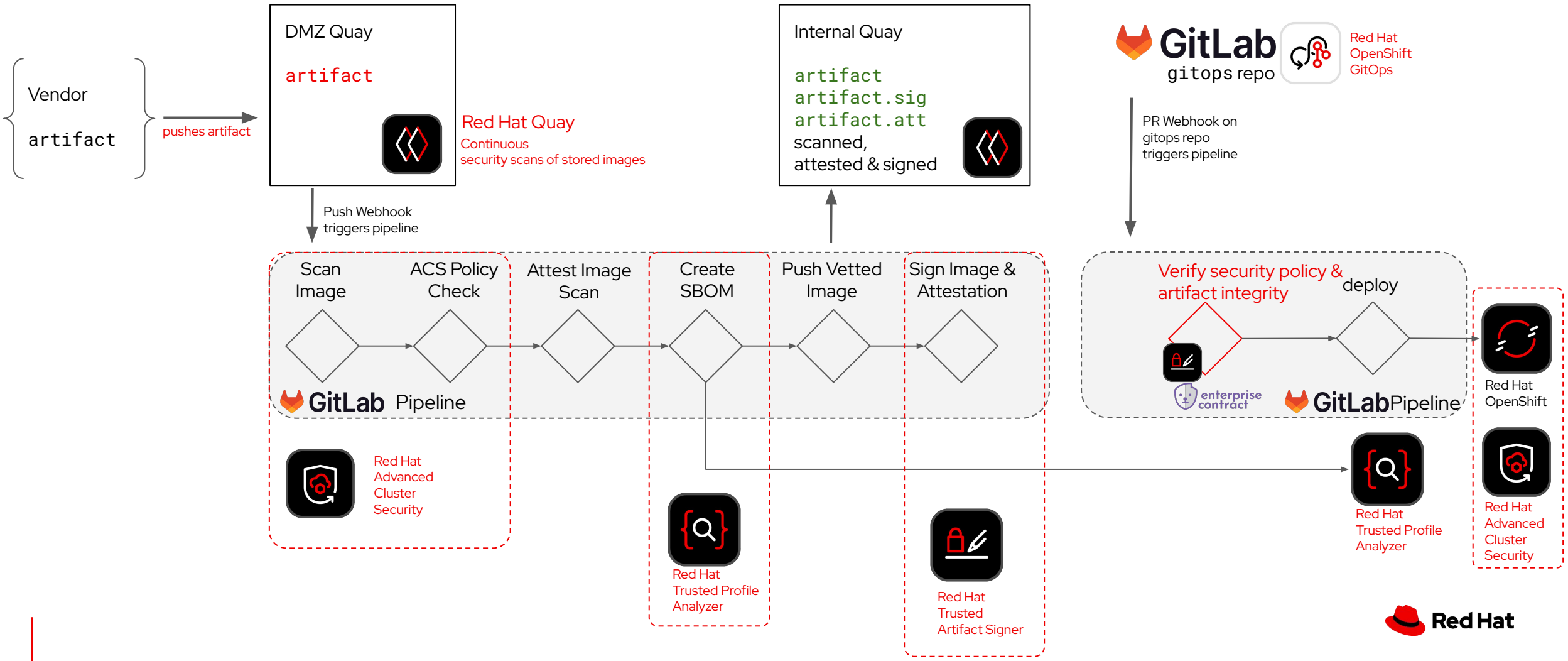
Red Hat

A security-augmented development process

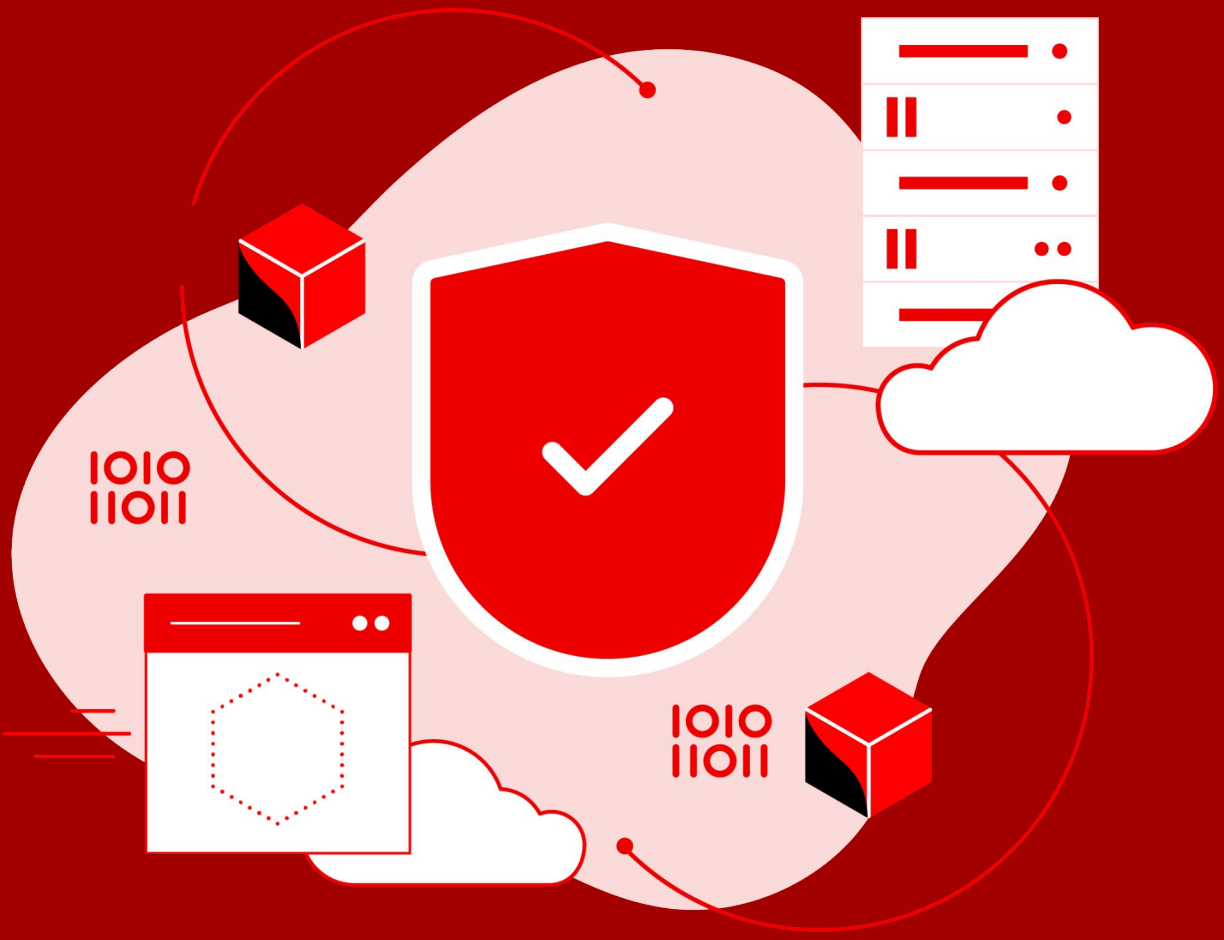


Red Hat

Securely Consuming Third-party artifacts in the SDLC



Red Hat



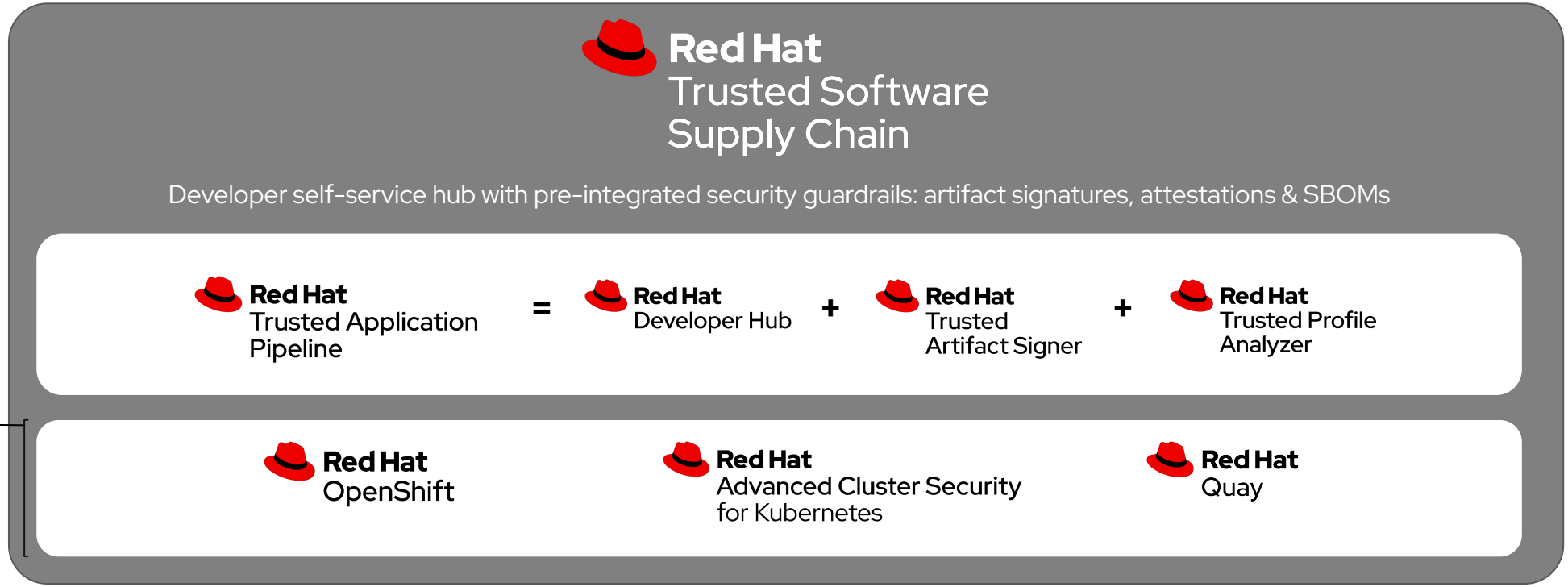
Workshop

Red Hat

Demo

Accelerate Innovation that Safeguards User Trust

Delivered with integrated security guardrails at every phase of the software development lifecycle



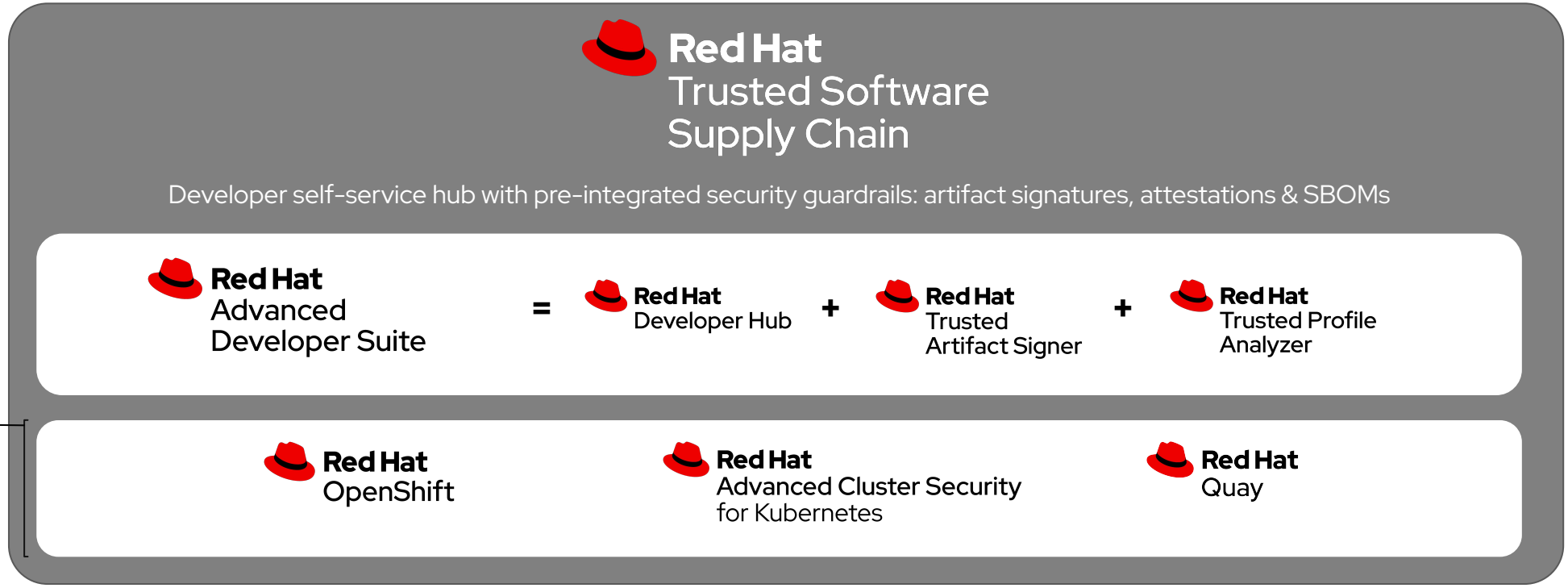
Included in
Red Hat OpenShift Platform Plus
but also available separately

**Red Hat Trusted Application Pipeline is a single product SKU*
Includes Red Hat Developer Hub, Red Hat Trusted Artifact Signer, Red Hat Trusted Profile Analyzer capabilities and a multi-product installer

Red Hat

Accelerate Innovation that Safeguards User Trust

Delivered with integrated security guardrails at every phase of the software development lifecycle



Included in
Red Hat OpenShift Platform Plus
but also available separately

**Red Hat Advanced Developer Suite is a single product SKU*
Includes Red Hat Developer Hub, Red Hat Trusted Artifact Signer, Red Hat Trusted Profile Analyzer capabilities and a multi-product installer

Red Hat

Red Hat Open Hybrid Cloud Platform

Announced at Red Hat Summit '25

Red Hat OpenShift Platform Plus

Advanced Management & Security

Multicluster Management | Cluster Security Global Registry | Cluster Data Management

Red Hat Advanced Developer Suite

Advanced Development Capabilities

Internal Development Portal | Secure Software Delivery | Developer Tools

Red Hat OpenShift

Foundational Application Platform Capabilities

Service Mesh | Serverless | Builds | Pipelines | GitOps |Tracing | Log Management | Cost Management

Red Hat OpenShift Kubernetes Engine

Kubernetes

Run Containers and Virtual Machines

Red Hat OpenShift Virtualization Engine

Red Hat Enterprise Linux

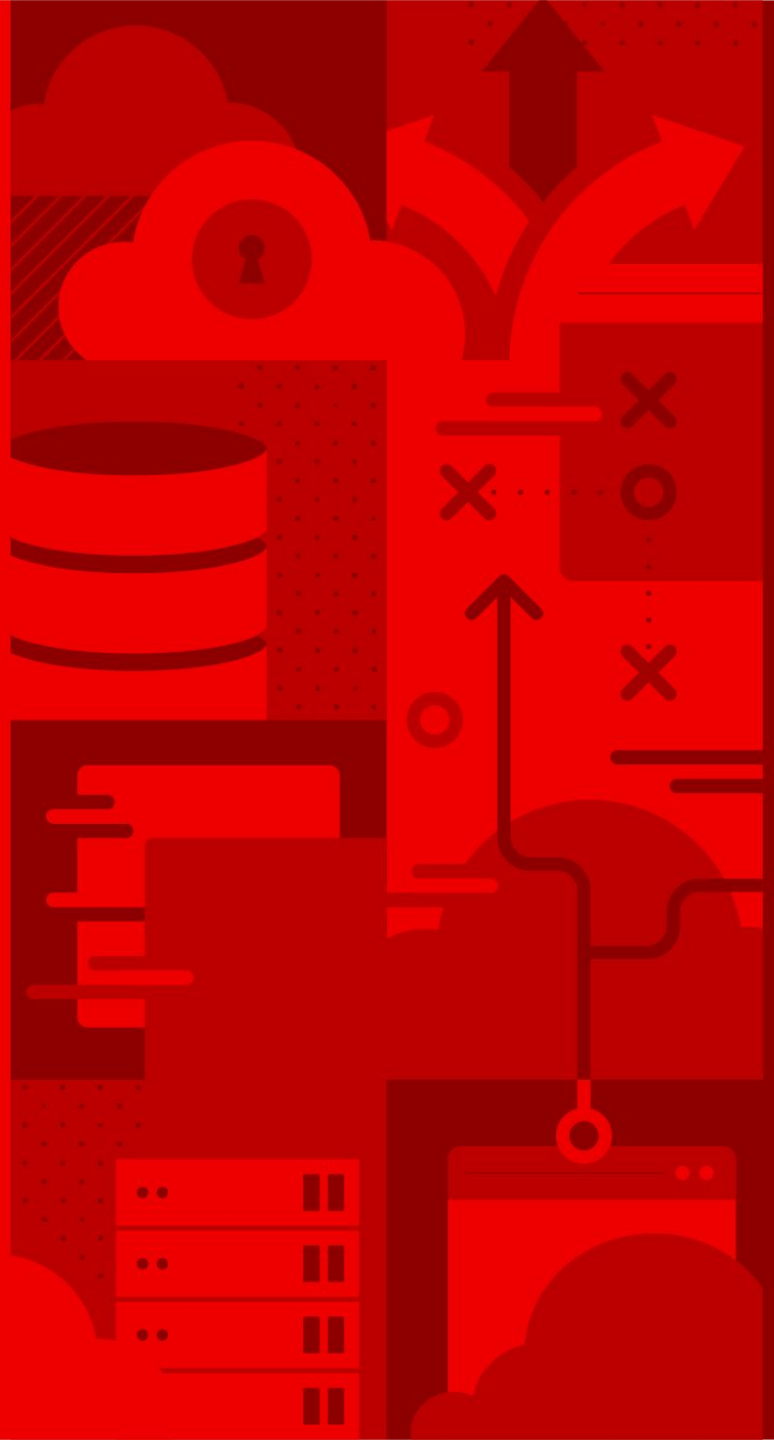
Linux

host operating system

OpenShift Cloud Services

- Red Hat OpenShift Service on AWS
- Azure Red Hat OpenShift
- OpenShift Dedicated
- Red Hat OpenShift on IBM Cloud

Physical — Virtual — Private cloud — Public cloud — Edge



Questions and feedback?

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat

Deeper Dive Slides

Red Hat

Trusted Artifact Signer (TAS)

Red Hat

Keyless Signing? Huh?

"Traditional" Signing & Verification

- Has been around "for ages"
- Public/private key pairs are generated, then distributed
- Signer uses private key (and needs to remember the PK password 🫣 🙄) to sign an artifact
- Verifier needs the artifact and the public key to verify signature

Challenges

- **Identity:** How do you know the person signing the artifact is who they say they are?
- **Key management:** How do you keep the private key secure so it can't be lost or stolen? How do you make the public key easily accessible for users, but also protect it from tampering by a malicious attacker?
- **Key revocation:** If the keypair is compromised, how do you distribute new keys in a way that convinces users of your legitimacy and that you're not an attacker?

Result: Well known but rarely used at scale in an enterprise environment.

Keyless Signing? Huh?

“Keyless” Signing & Verification

- The Sigstore project
  - Backed by the Open Source Security Foundation (OpenSSF) under the Linux Foundation
  - Contributions from Google, **Red Hat**, Chainguard, GitHub and Purdue University
  - Goal: Simplify signing and verification
  - Also operates a public-good, non-profit service to improve the open source software supply chain (transparency log is publicly accessible).

Benefits

Your artifact is:

- **Signed**: By using a Sigstore client (Cosign/gitsign).
- **Associated**: With an identity through our certificate authority (Fulcio).
- **Witnessed**: By recording the signing information in a permanent transparency log (Rekor).

Keyless Signing? Huh?

“Keyless” Signing & Verification

- The Sigstore project
  - Backed by the Open Source Security Foundation (OpenSSF) under the Linux Foundation
  - Contributions from Google, **Red Hat**, Chainguard, GitHub and Purdue University
  - Goal: Simplify signing and verification
  - Also operates a public-good, non-profit service to improve the open source software supply chain (transparency log is publicly accessible).

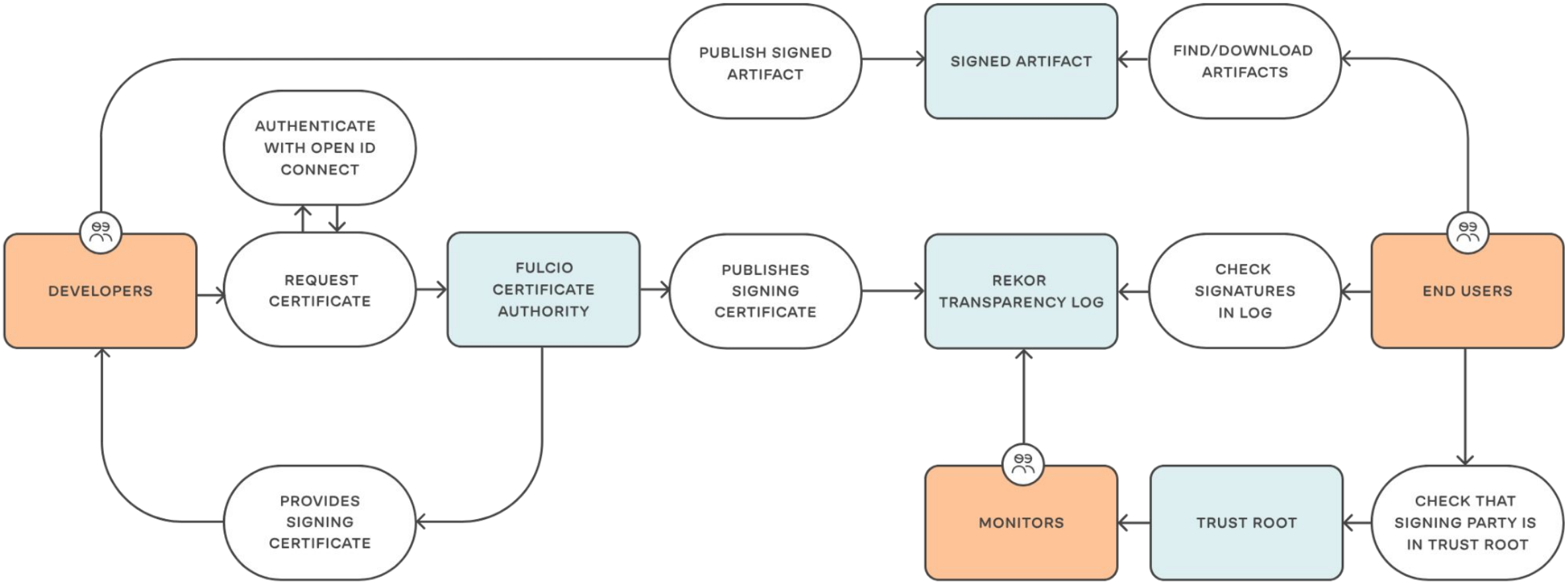
“Keyless” or “ephemeral key” signing:

- Using the transparency log for signature verification rather than keys.

Convenience: Convenient tooling, easy container signing, bypass the difficult problem of key management and rotation.

Security: With Sigstore, the artifact is not just signed; it’s signed with an ephemeral key, associated with a known identity, and publicly auditable.

Keyless Signing? A-ha!



Red Hat

Keyless Signing? A-ha!

Cosign

For signing and verification of artifacts and containers, with storage in an Open Container Initiative (OCI) registry, making signatures and in-toto/SLSA attestations invisible infrastructure.

Rekor

Append-only, auditable transparency log service, Rekor records signed metadata to a ledger that can be queried, but can’t be tampered with.

OpenID Connect

An identity layer that checks if you're who you say you are. It lets clients request and receive information about authenticated sessions and users.

Fulcio

Code-signing certificate authority, issuing short-lived certificates to an authenticated identity and publishing them to a certificate transparency log.

Trust root

The foundation for trust underpinning Sigstore utilizes TUF (The Update Framework).

Key Functions of the TUF Root in Sigstore:

- **Key Management:** It securely manages root keys and delegated keys for signing.
- **Metadata Distribution:** It provides signed metadata that clients use to verify the authenticity and integrity of software artifacts.
- **Security Enhancements:** It enables key rotation, revocation, and delegation, which are crucial for maintaining long-term security.