

# Red Hat Trusted Software Supply Chain

Matching supply chain resiliency with innovation speed

Ilkka Tengvall

Solutions Architect

[ikke@redhat.com](mailto:ikke@redhat.com)

Matrix: #redhat-fin-users:hacklab.fi





When we buy a new car we expect every part supplied to be genuine

...



<https://www.youtube.com/watch?v=3Hip60FA6j8>





... can we say the same about every  
single part of our **software** ?

# 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<sup>1</sup>

## 20%

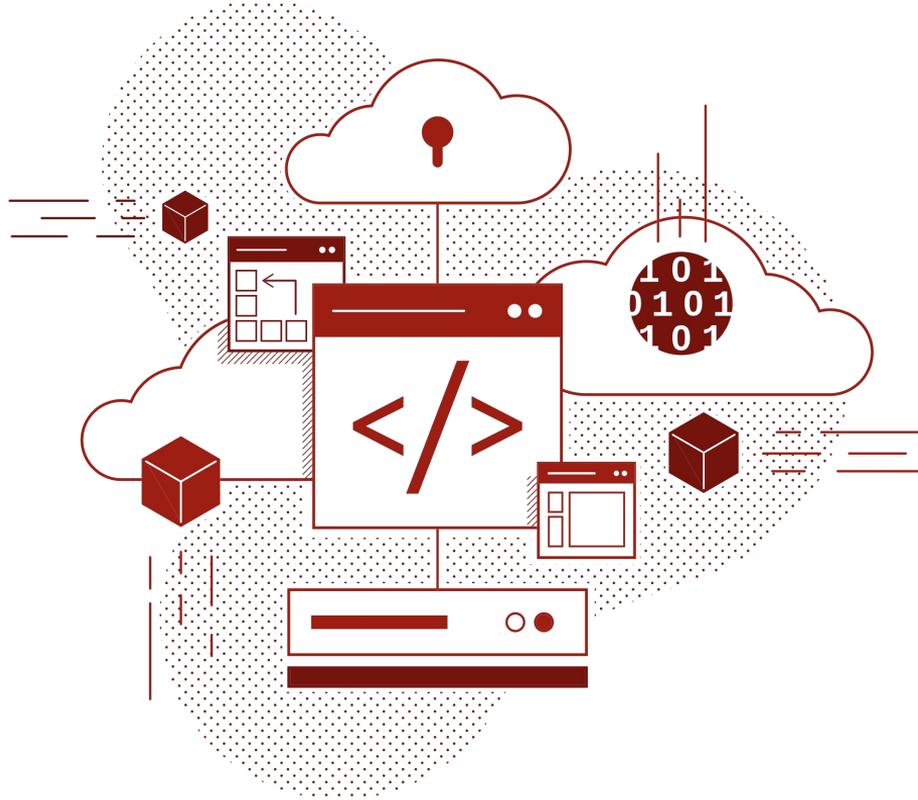
data breaches are due to a  
compromised software  
supply chain<sup>2</sup>

## 78%

have initiatives to  
increase collaboration  
between DevOps and  
Security teams<sup>3</sup>

## 92%

say enterprise open source  
solutions are important as  
their business accelerates to  
the open hybrid cloud<sup>4</sup>



## Growing attack surfaces with new, emerging threats daily

Software supply chain security a critical component to securing data, IP and source code

- Stolen Certificates
- Typosquatting Attack
- Dependency Confusion
- Compromised Build Environment
- Malware preinstalled on devices
- Malicious code in firmware

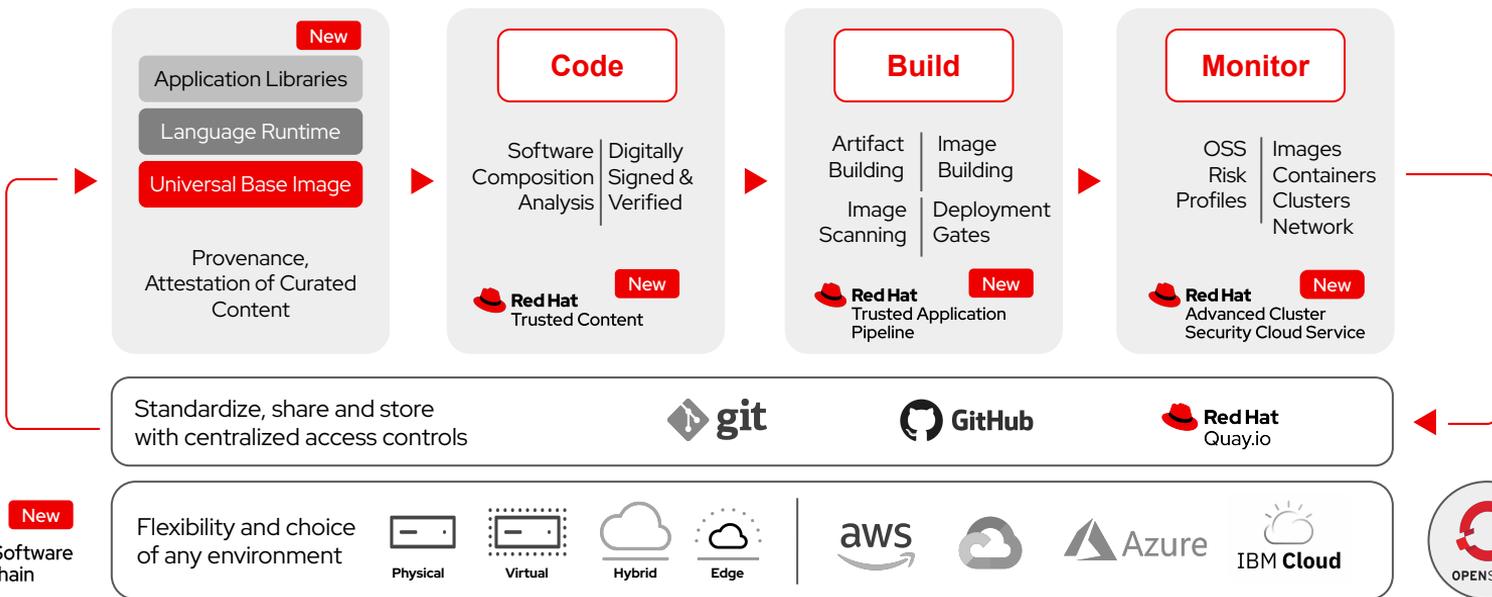
# Red Hat: Providing trusted enterprise open source software for 30+ years



- ▶ All code is cloned in internal repositories.
- ▶ Strong distribution mechanisms with signed packages.
- ▶ Strong safeguards against tampering.
- ▶ Minimal modifications over product lifetimes protects from unwanted and potentially risky upstream code changes.

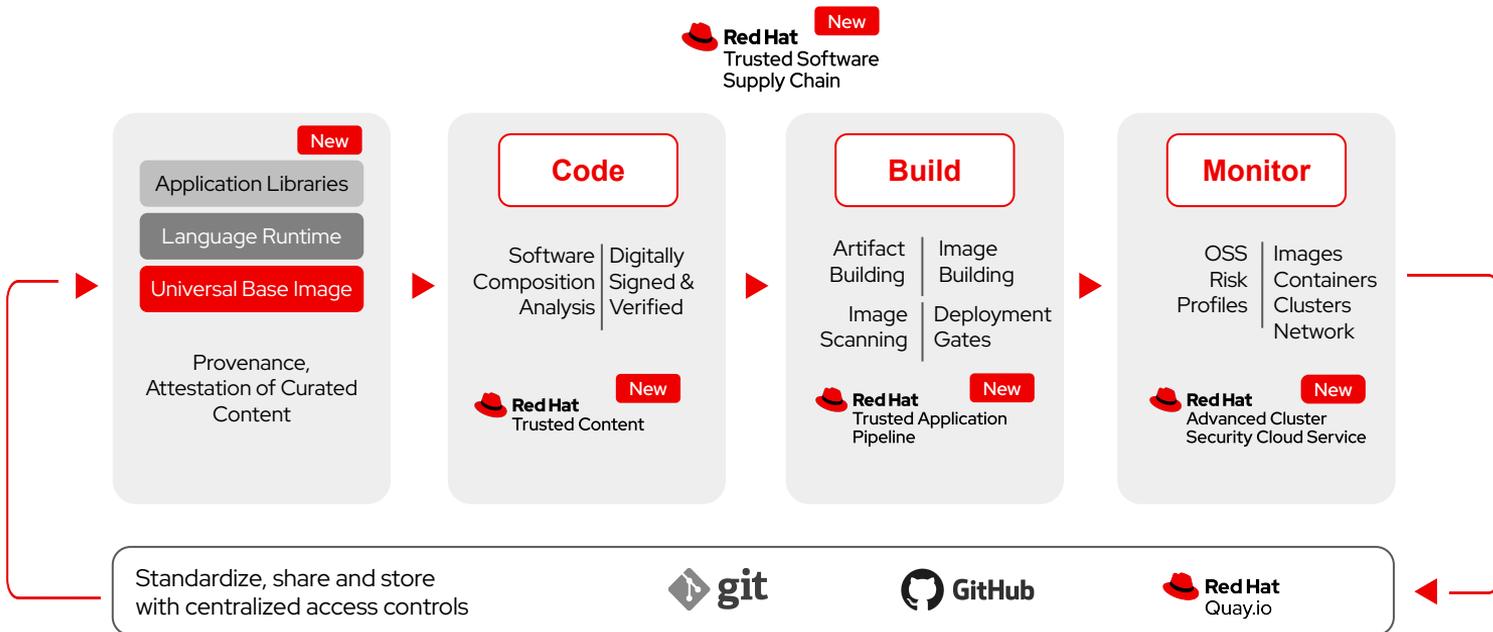
# Code, build, and monitor to a Trusted Software Supply Chain

Delivered as a **cloud service** with integrated security guardrails at every phase of the software development lifecycle



# Code, build, and monitor to a Trusted Software Supply Chain

Delivered as a **cloud service** with integrated security guardrails at every phase of the software development lifecycle



Ulkomaat | Australia

# Nainen tarjosi vierailleen myrkky-sienipiirasta, nyt häntä epäillään kolmen ihmisen kuolemasta

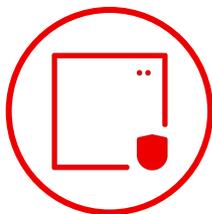
Poliisi tutkii Australiassa kolmen ihmisen kuolemaan johtanutta epäiltyä sienimyrkytystä.

<https://www.hs.fi/ulkomaat/art-2000009775653.html>

# Secure the use of source code and transitive dependencies

Software supply chain security considerations for the software development lifecycle

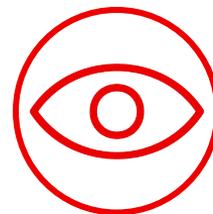
Prevent & identify  
malicious **code**

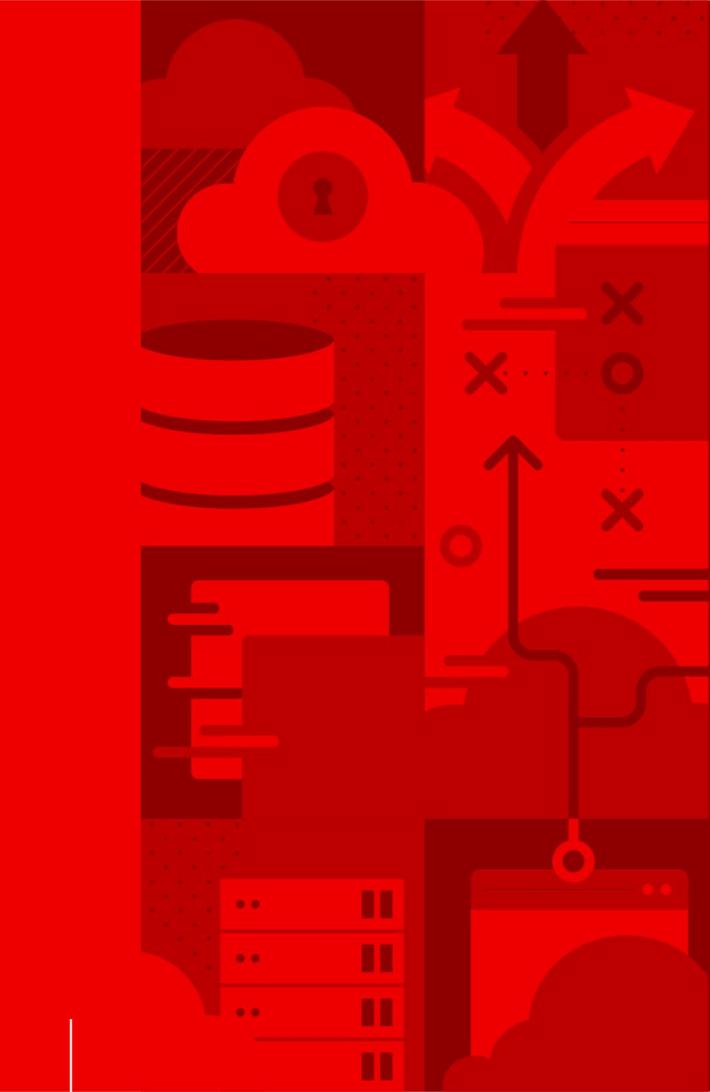


Safeguard **build**  
systems early



Continuously **monitor**  
security at runtime





Secure your  
open source  
code and  
dependencies  
early!

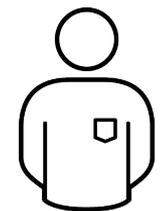
## Terminology

Term	Definition
SLSA	Supply Chain Levels for Software Artifacts SLSA is a set of standards and technical controls you can adopt to improve artifact integrity, and build
SAST	Static Application Security Testing Executed at build time as part of the CI
DAST	Dynamic Application Security Testing Often executed on staging clusters
OWASP	Open Web Application Security Project <a href="#">OWASP Top 10</a>
CVE	Common Vulnerability and Exposures
Provenance	Recording of origin, history and who made the changes
Attestation	Authenticated statement (metadata) about a software artifact or collection of software artifacts
Sigstore	Sigstore empowers software developers to securely sign software artifacts such as release files, container images, binaries, bill of material manifests and more. Signing materials are then stored in a tamper-resistant public log.
SBOM	Software Bill of Materials
SPDX, CycloneDX	Competing solutions for the structure of a SBOM. SPDX lead by Linux Foundation. CycloneDX lead by OWASP.
SCA	Software Composition Analysis

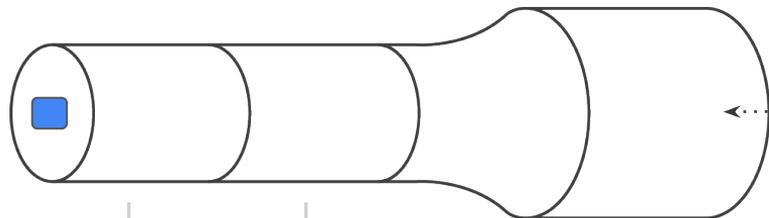
Level 1	Level 2	Level 3
<p>Preventing Mistakes</p> <p>Automated Build Process</p> <p>Generated provenance about source, build process, artifact and dependencies</p>	<p>Preventing tampering after the build</p> <p>Generated, signed and verifiable provenance</p>	<p>Preventing tampering during the build</p> <p>Prevent runs from influencing one another, prevent secret material used to sign provenance from being accessible by the end-user's defined steps</p>



Shift Left



Developer



Development

QA

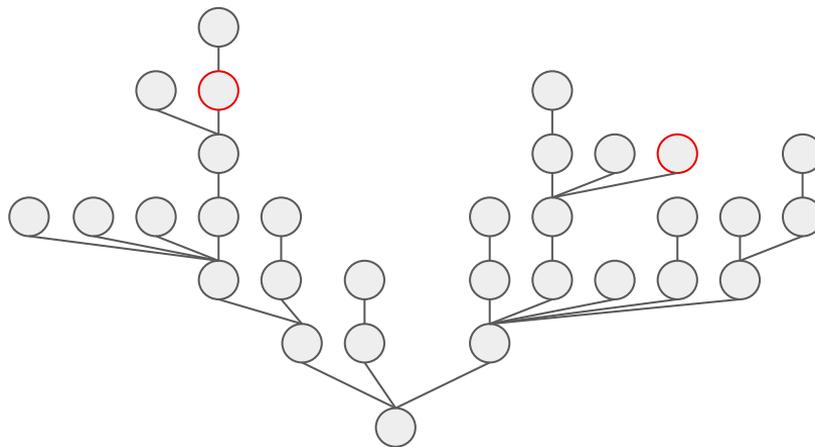
Staging

Production

Router



Users



I need a HTTP library, JSON parser,  
database access, Java runtime, Linux OS



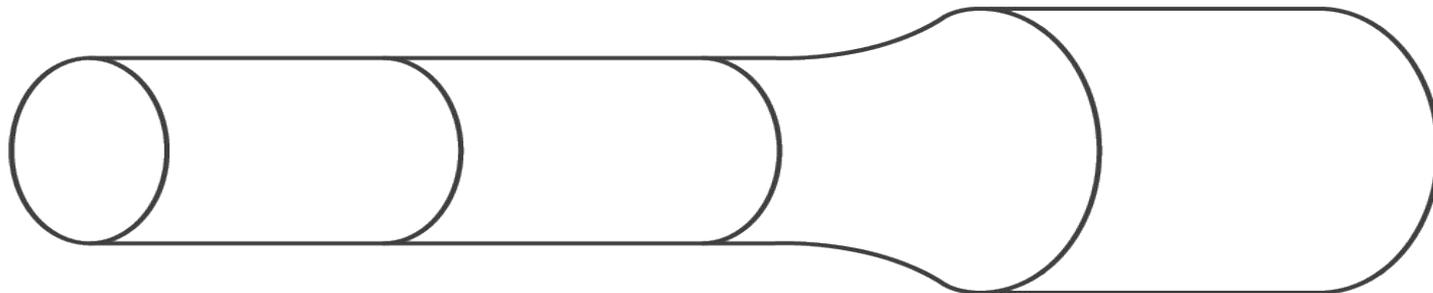
pom.xml  
package.json  
requirements.txt  
Dockerfile

## Spring Boot 2.7.7 Hello World

```

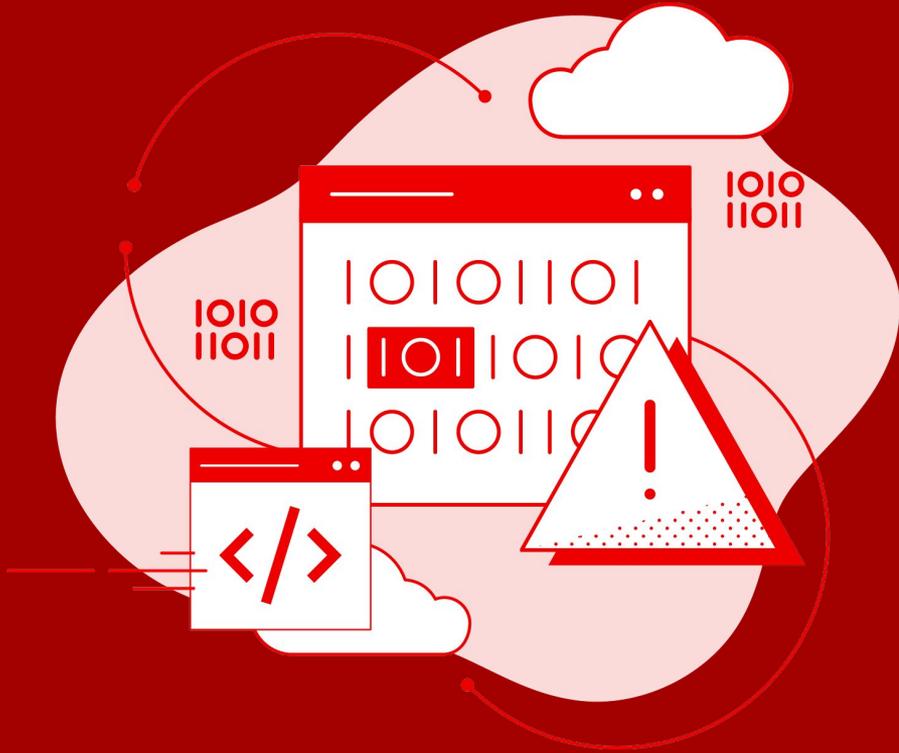
[INFO] com.example.demo:jar:0.0.1-SNAPSHOT
[INFO] +- org.springframework.boot:spring-boot-starter-web:jar:2.7.7:compile
[INFO] | +- org.springframework.boot:spring-boot-starter:jar:2.7.7:compile
[INFO] | | +- org.springframework.boot:spring-boot-starter-logging:jar:2.7.7:compile
[INFO] | | | +- ch.qos.logback:logback-classic:jar:1.2.11:compile
[INFO] | | | | \- ch.qos.logback:logback-core:jar:1.2.11:compile
[INFO] | | +- org.apache.logging.log4j:log4j-to-slf4j:jar:2.17.2:compile
[INFO] | | | \- org.apache.logging.log4j:log4j-api:jar:2.17.2:compile
[INFO] | | \- org.slf4j:jul-to-slf4j:jar:1.7.36:compile
[INFO] | +- jakarta.annotation:jakarta.annotation-api:jar:1.3.5:compile
[INFO] | \- org.yaml:snakeyaml:jar:1.30:compile
[INFO] +- org.springframework.boot:spring-boot-starter-json:jar:2.7.7:compile
[INFO] | +- com.fasterxml.jackson.core:jackson-databind:jar:2.13.4.2:compile
[INFO] | | +- com.fasterxml.jackson.core:jackson-annotations:jar:2.13.4:compile
[INFO] | | \- com.fasterxml.jackson.core:jackson-core:jar:2.13.4:compile
[INFO] | +- com.fasterxml.jackson.datatype:jackson-datatype-jdk8:jar:2.13.4:compile
[INFO] | +- com.fasterxml.jackson.datatype:jackson-datatype-jsr310:jar:2.13.4:compile
[INFO] | \- com.fasterxml.jackson.module:jackson-module-parameter-names:jar:2.13.4:compile
[INFO] +- org.springframework.boot:spring-boot-starter-tomcat:jar:2.7.7:compile
[INFO] | +- org.apache.tomcat.embed:tomcat-embed-core:jar:9.0.70:compile
[INFO] | +- org.apache.tomcat.embed:tomcat-embed-el:jar:9.0.70:compile
[INFO] | \- org.apache.tomcat.embed:tomcat-embed-websocket:jar:9.0.70:compile
[INFO] +- org.springframework:spring-web:jar:5.3.24:compile
[INFO] | \- org.springframework:spring-beans:jar:5.3.24:compile
[INFO] \- org.springframework:spring-webmvc:jar:5.3.24:compile
[INFO] +- org.springframework:spring-aop:jar:5.3.24:compile
[INFO] +- org.springframework:spring-context:jar:5.3.24:compile
[INFO] \- org.springframework:spring-expression:jar:5.3.24:compile
[INFO] +- org.springframework.boot:spring-boot-devtools:jar:2.7.7:compile
[INFO] | +- org.springframework.boot:spring-boot:jar:2.7.7:compile
[INFO] | \- org.springframework.boot:spring-boot-autoconfigure:jar:2.7.7:compile
[INFO] \- org.springframework.boot:spring-boot-starter-test:jar:2.7.7:test
[INFO] +- org.springframework.boot:spring-boot-test:jar:2.7.7:test
[INFO] +- org.springframework.boot:spring-boot-test-autoconfigure:jar:2.7.7:test
[INFO] +- com.jayway.jsonpath:json-path:jar:2.7.0:test
[INFO] | +- net.minidev:json-smart:jar:2.4.8:test
[INFO] | | \- net.minidev:accessors-smart:jar:2.4.8:test
[INFO] | | \- org.ow2.asm:asm:jar:9.1:test
[INFO] | \- org.slf4j:slf4j-api:jar:1.7.36:compile
[INFO] +- jakarta.xml.bind:jakarta.xml.bind-api:jar:2.3.3:test
[INFO] | \- jakarta.activation:jakarta.activation-api:jar:1.2.2:test
[INFO] +- org.assertj:assertj-core:jar:3.22.0:test
[INFO] +- org.hamcrest:hamcrest:jar:2.2:test
[INFO] +- org.junit.jupiter:junit-jupiter:jar:5.8.2:test
[INFO] | +- org.junit.jupiter:junit-jupiter-api:jar:5.8.2:test
[INFO] | | +- org.opentest4j:opentest4j:jar:1.2.0:test
[INFO] | | \- org.junit.platform:junit-platform-commons:jar:1.8.2:test
[INFO] | | \- org.apiguardian:apiguardian-api:jar:1.1.2:test
[INFO] | +- org.junit.jupiter:junit-jupiter-params:jar:5.8.2:test
[INFO] | \- org.junit.jupiter:junit-jupiter-engine:jar:5.8.2:test
[INFO] | \- org.junit.platform:junit-platform-engine:jar:1.8.2:test
[INFO] +- org.mockito:mockito-core:jar:4.5.1:test
[INFO] | +- net.bytebuddy:byte-buddy:jar:1.12.20:test
[INFO] | +- net.bytebuddy:byte-buddy-agent:jar:1.12.20:test
[INFO] | \- org.objenesis:objenesis:jar:3.2:test
[INFO] +- org.mockito:mockito-junit-jupiter:jar:4.5.1:test
[INFO] +- org.skyscreamer:jsonassert:jar:1.5.1:test
[INFO] | \- com.vaadin.external.google:android-json:jar:0.0.20131108.vaadin1:test
[INFO] +- org.springframework:spring-core:jar:5.3.24:compile
[INFO] | \- org.springframework:spring-jcl:jar:5.3.24:compile
[INFO] +- org.springframework:spring-test:jar:5.3.24:test
[INFO] \- org.xmlunit:xmlunit-core:jar:2.9.0:test
[INFO] -----

```



As a managed service, you can be up and running in minutes. Complicated product integrations are handled for you. Upgrades are continuous and seamless.

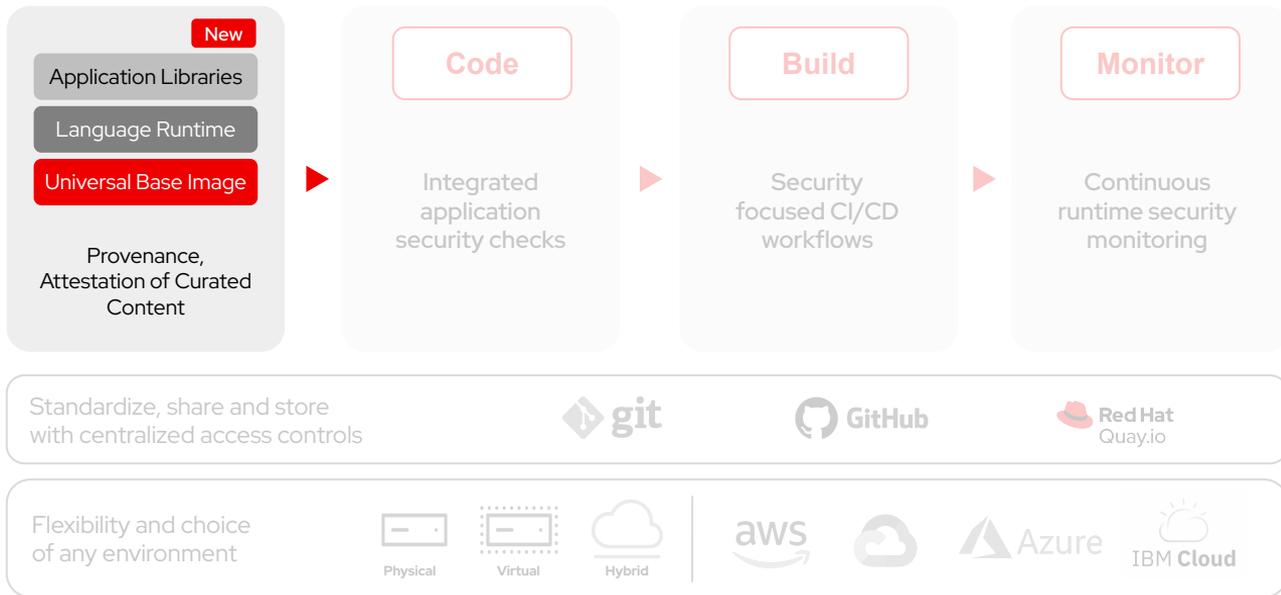
Deliver **securely-built images** to a registry, deploy applications to the cloud or to your on-prem OpenShift cluster with just a few steps.



*Prevent and  
identify malicious  
code*

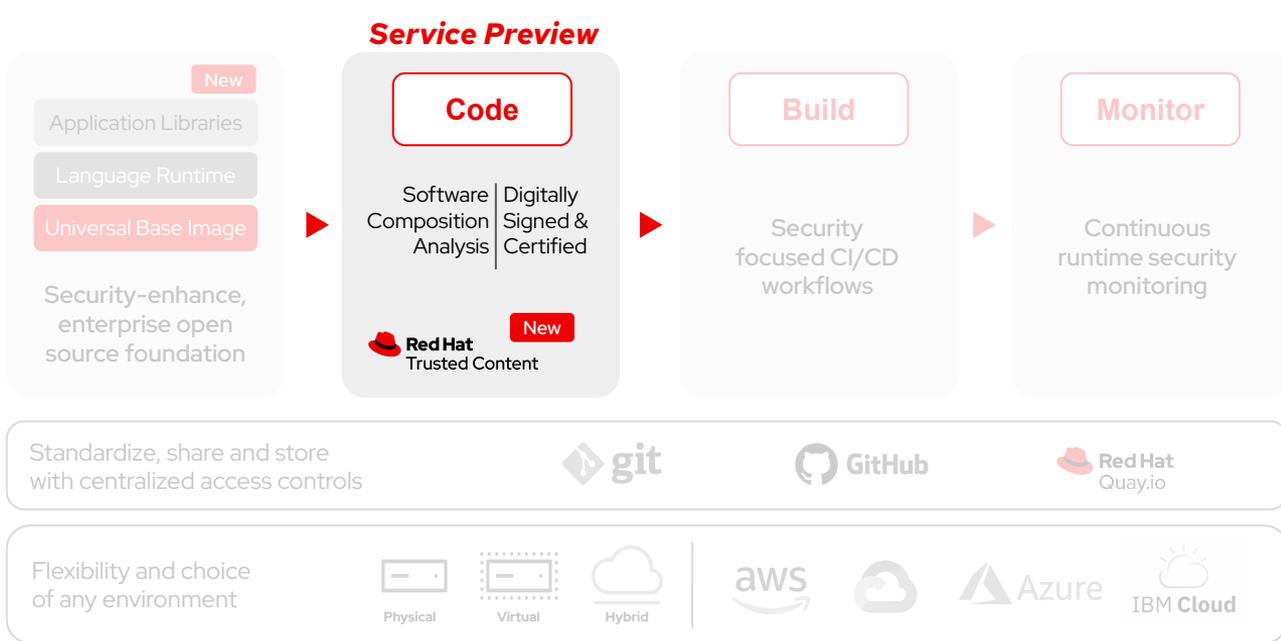
# Security-enhanced, enterprise open source foundation

Build on business resilience that keeps pace with innovation velocity

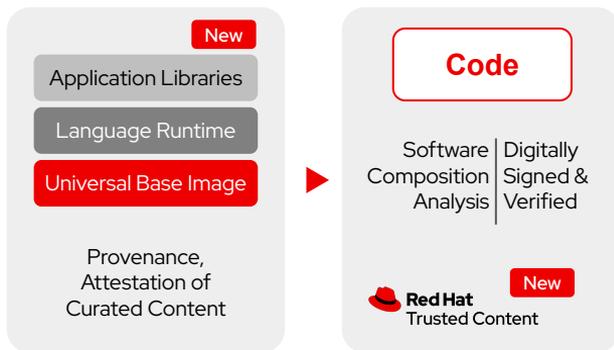


# Integrated application security checks

Catch security issues early to keep and grow user trust



## Code with integrated application security checks



Catch security issues early to  
keep and grow user trust

- ▶ Trusted curated content
- ▶ Automated software composition analysis and dependency analytics
- ▶ Aggregated view with drill down on security health
- ▶ Cryptographic signing and verification

# Leverage tried tested trusted curated content with security best practices at code time

- ▶ 30 years providing trusted images and app libraries that are signed, verified
- ▶ Automate dependency analytics early with plug-ins to popular IDEs
- ▶ Single, shared repository for trusted content, detailed usage information, security issues and recommendations
- ▶ Tamper proof code to verify content from an open, immutable ledger

Dependency Analytics report

**Security issues**

Dependencies with high common vulnerabilities and exposures (CVE) score. Total vulnerabilities: 81  
Vulnerable dependencies: 9

**Dependencies with security issues in your stack**

A list of the dependencies affected with CVE, as well as vulnerability only found while using Snyk's vulnerability database.

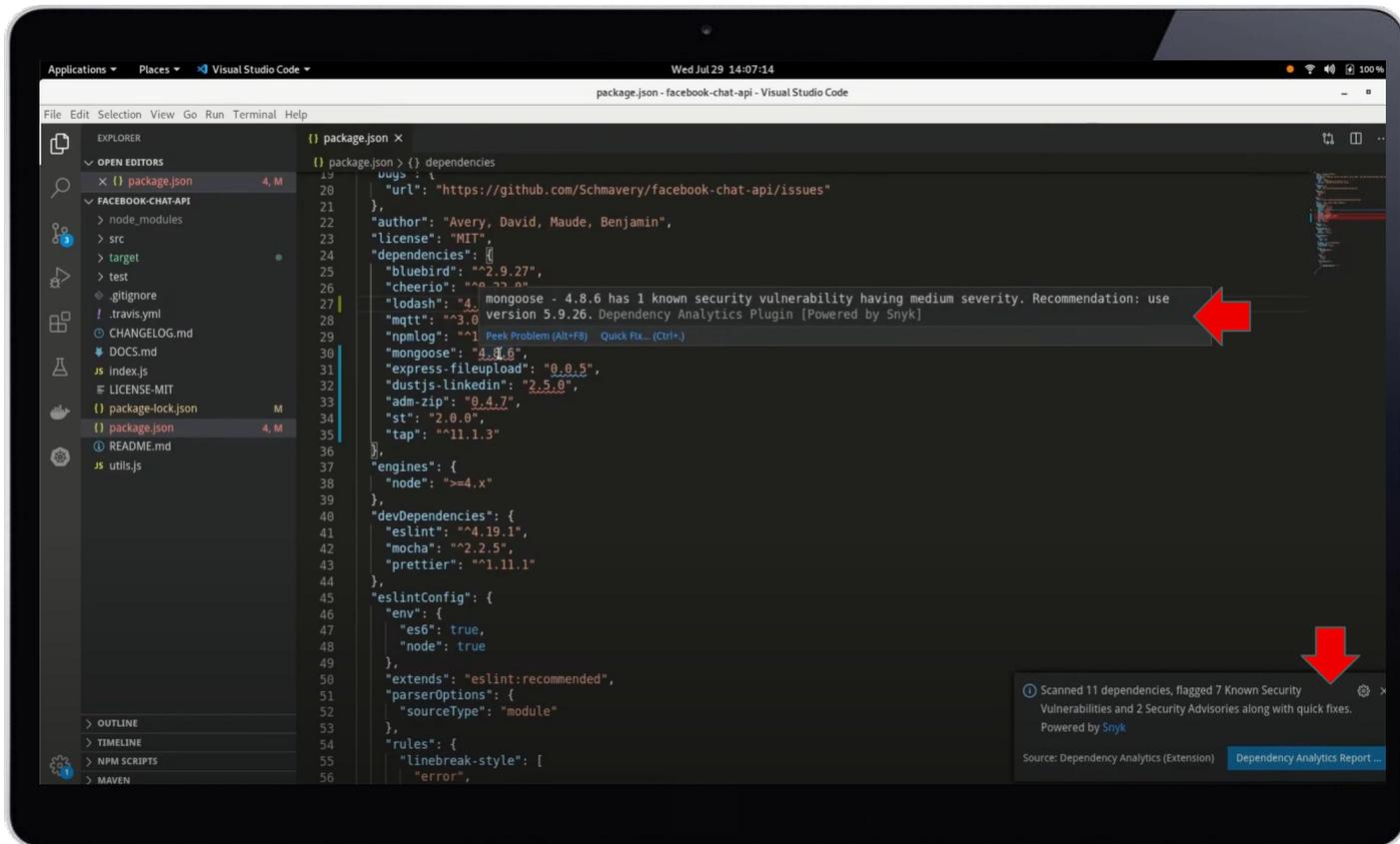
Commonly known vulnerabilities: 80

Dependency	# Direct Vulnerabilities	# Transitive Vulnerabilities	Highest CVSS Score	Highest Severity	Red Hat Remediation Available
org.jboss.nestor-client	1	6	5.1/10	SRNYC-JAVA-DRC-XXXXXX	🛡️
com.github.javafaker:javafaker	7	4	7.5/10	CVE-2022-41725	🛡️

Details of the dependency: com.github.javafaker:javafaker

Severity	Description	CVSS Score	ID	Remediation ?
CRITICAL	Information Exposure	5.1/10	CVE-2022-41678	1.7.3-redhat-001
HIGH	Denial of Service (DoS)	7.5/10	CVE-2022-41725	1.2.3-redhat-003

# Remedy vulnerabilities with Trusted Content



# Analyze and fix security issues from the IDE

**Security issues**

Dependencies with high common vulnerabilities and exposures (CVE) score. Total vulnerabilities: 81

Vulnerable dependencies: 9

**Dependencies with security issues in your stack**

A list of the dependencies affected with CVE, as well as vulnerability only found while using Snyk's vulnerability database.

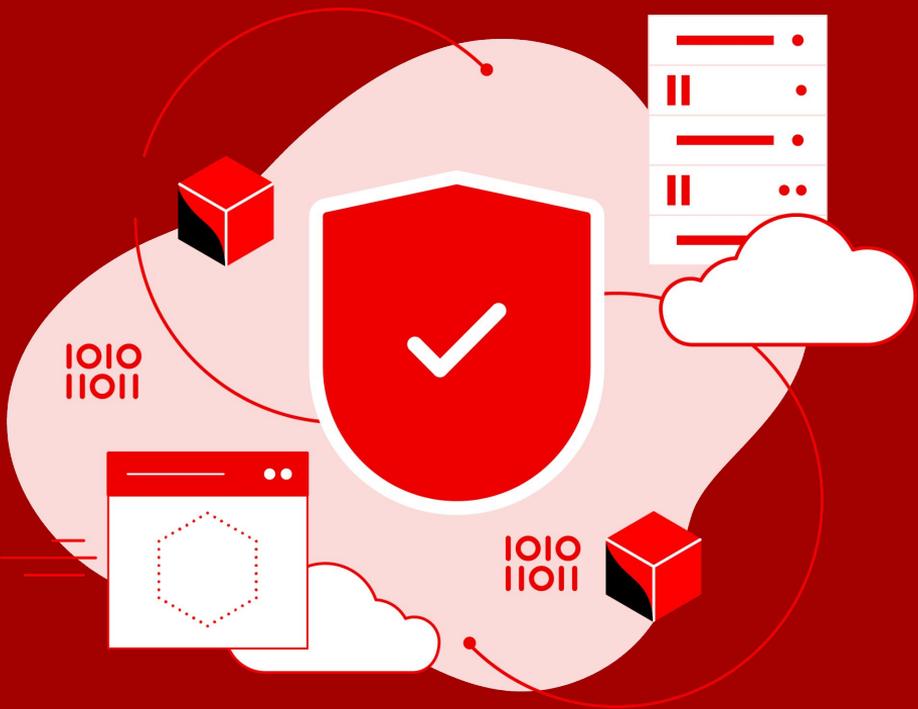
Commonly known vulnerabilities: 80

Dependency	# Direct Vulnerabilities	# Transitive Vulnerabilities	Highest CVSS Score	Highest Severity	Red Hat Remediation Available
org.jboss.resteasy-client	1	6	5.1/10	SNYK-JAVA-ORG-XXXXX	✓
com.github.javafaker:javafaker	2	4	7.5/10	CVE-2022-41725	✓

Details of the dependency: com.github.javafaker:javafaker

Severity	Description	CVSS Score	ID	Remediation ?
Medium	Information Exposure	5.1/10	CVE-2022-41678	12.3-redhat-003
High	Denial of Service (DoS)	7.5/10	CVE-2022-41725	12.3-redhat-003

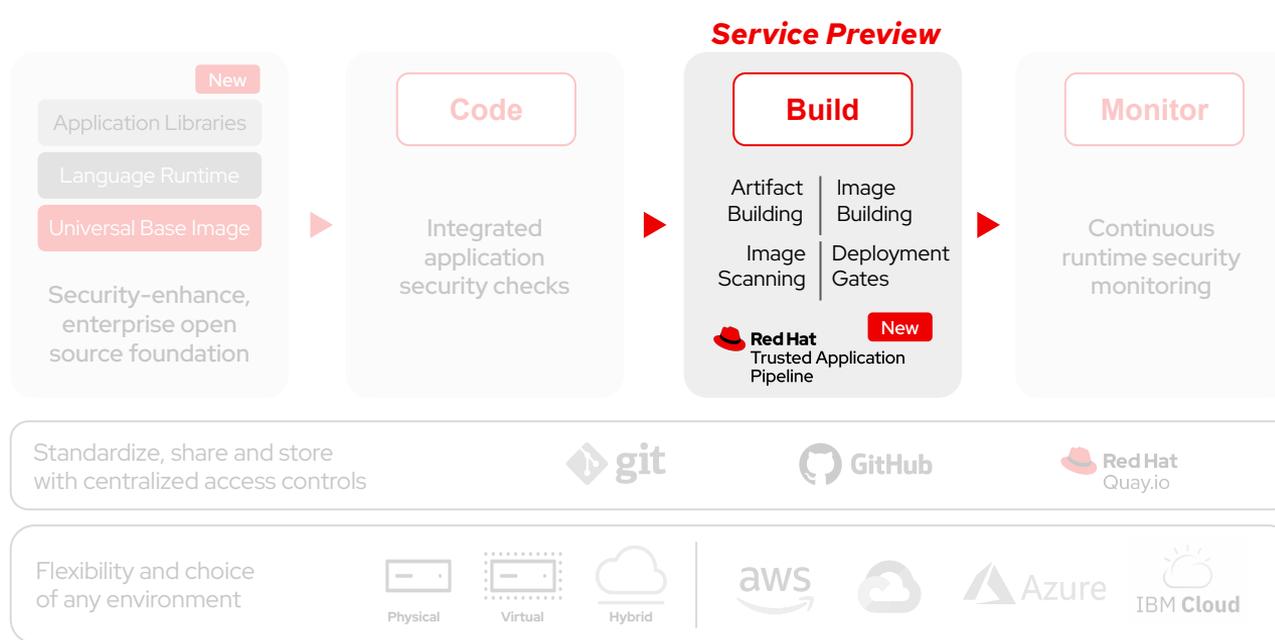
TEMO



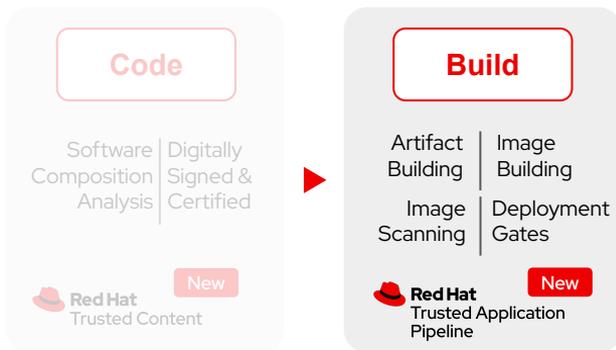
*Safeguard build systems early*

# Security focused CI/CD workflows

Improved productivity, efficiency addresses customer needs faster



## Build with security focused CI/CD workflows



Meet industry compliance while increasing productivity, efficiency

- ▶ Integrated security guardrails across pipelines
- ▶ Auto-generated Software-Bill-of-Materials (SBOM)
- ▶ Attestations and provenance checks
- ▶ Deployment based on policies to a declared state
- ▶ Continuous image vulnerability scanning

# Strengthen the CI/CD pipeline with an automated chain of trust and approval gates

- ▶ Ready to use, customizable pipeline definition for hermetic builds
- ▶ Auto-generated SBOMs in minutes
- ▶ Scan images for vulnerabilities and exposures
- ▶ Continuously deploy via enterprise contract's 43 rules
- ▶ Pre-integrated security guardrails via Tekton Chains

Red Hat Hybrid Cloud Console

Services Favorites

Beta on Samburral Parker

CJ/CD

Overview

Applications

partner-catalog-custom-5dje-on-push-8h5m9 Succeeded

Actions

Details Task runs Logs

Pipeline run details

clone-repository prefetch-dependencies build-container trusted-content sanity-inspect-image sanity-label-check sanity-optional-label-check clair-scan clamav-scan sbom-json-check deprecated-base-image-check sast-snyk-check rowctl-scan

Name partner-catalog-custom-5dje-on-push-8h5m9

Namespace samburral-tenant

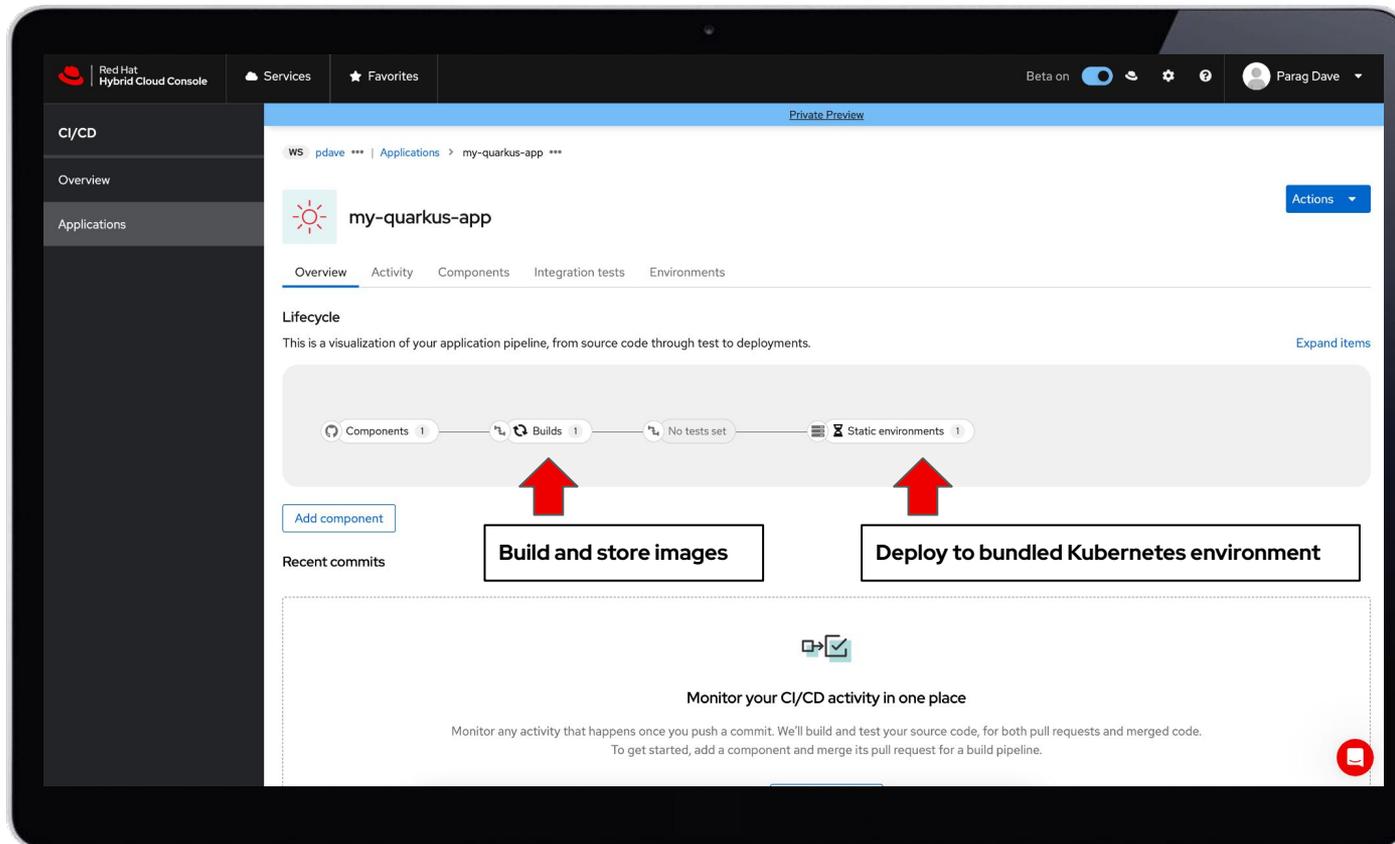
Labels

Status Succeeded

Pipeline partner-catalog-custom-5dje-on-push-8h5m9

Download SBOM

# Automatically run default CI/CD pipeline



# Drill down on pipeline details

The screenshot displays the Red Hat Hybrid Cloud Console interface. The top navigation bar includes the Red Hat logo, 'Red Hat Hybrid Cloud Console', 'Services', 'Favorites', and a user profile for 'Parag Dave'. A 'Beta on' toggle is also visible. The left sidebar shows a navigation menu with 'C/CD', 'Overview', and 'Applications' selected. The main content area is titled 'Private Preview' and shows the details for 'my-quarkus-app'. The 'Activity' tab is active, displaying a table of pipeline runs. A red arrow points to the first row of the table.

WS pdave | Applications > my-quarkus-app

my-quarkus-app

Overview **Activity** Components Integration tests Environments

Activity By

Latest commits Pipeline runs

Filter by name... Status

Name	Started	Vulnerabilities	Duration	Status	Type	Component
<a href="#">devfile-sample-code-with-quarkus-417f-d817s</a>	Just now	-	59 seconds	Running	Build	<a href="#">devfile-sample-code-with-quarkus-417f</a>

# View live pipeline runs in real-time

The screenshot displays the Red Hat Hybrid Cloud Console interface. The top navigation bar includes the Red Hat logo, 'Red Hat Hybrid Cloud Console', 'Services', 'Favorites', 'Beta on' toggle, and a user profile for 'Parag Dave'. The main content area shows a pipeline run for 'devfile-sample-code-with-quarkus-4t7f-d8l7s' in a 'Running' state. The pipeline run details are visualized as a flowchart with the following steps:

- init (Completed)
- clone-repository (Completed)
- prefetch-dependencies (In Progress)
- build-container (Expanded view):
  - build (1 minute 4 seconds)
  - sbom-get
  - analyse-dependencies-java-sbom
  - merge-sboms
  - inject-sbom-and-push
  - upload-sbom
- inspect-image (In Progress)
  - label-check
  - optional-label-check
- clair-scan (In Progress)
- clamav-scan (In Progress)
- sbom-json-check (In Progress)
- deprecated-base-image-check (In Progress)
- sast-snyk-check (In Progress)

Below the flowchart, the pipeline run details are summarized:

Name	devfile-sample-code-with-quarkus-4t7f-d8l7s	Status	Running
Namespace	pdave-tenant	Pipeline	docker-build

33

# Access pipeline task log

The screenshot displays the Red Hat Hybrid Cloud Console interface. The top navigation bar includes the Red Hat logo, 'Red Hat Hybrid Cloud Console', 'Services', 'Favorites', 'Beta on' status, and a user profile for 'Parag Dave'. The main content area shows a breadcrumb trail: 'WS pdave \*\*\* | Applications > my-quarkus-app \*\*\* > Pipeline runs > devfile-sample-code-with-quarkus-4t7f-d8l7s'. Below this, the pipeline run 'devfile-sample-code-with-quarkus-4t7f-d8l7s' is shown as 'Succeeded'. A red arrow points to the 'Logs' tab in the sub-navigation. The 'show-summary' task is selected, and its log content is displayed in a dark-themed window. The log content includes a build summary with repository and image information, and a secret deletion notice.

Red Hat Hybrid Cloud Console

Services Favorites

Beta on

Parag Dave

Private Preview

WS pdave \*\*\* | Applications > my-quarkus-app \*\*\* > Pipeline runs > devfile-sample-code-with-quarkus-4t7f-d8l7s

devfile-sample-code-with-quarkus-4t7f-d8l7s Succeeded

Actions

Details Task runs **Logs**

Download Download all task logs Expand

init

clone-repository

build-container

show-sbom

show-summary

```
show-summary
STEP-APPSTUDIO-SUMMARY

Build Summary:

Build repository: https://github.com/devfile-samples/devfile-sample-code-with-quarkus.git?rev=981e8b32ac180a1389d2b31a
Generated Image is in : quay.io/redhat-user-workloads/pdave-tenant/my-quarkus-app/devfile-sample-code-with-quarkus-4t7f-d8l7s

pipelinerun.tekton.dev/devfile-sample-code-with-quarkus-4t7f-d8l7s annotated
pipelinerun.tekton.dev/devfile-sample-code-with-quarkus-4t7f-d8l7s annotated
End Summary
secret "devfile-sample-code-with-quarkus-4t7f-d8l7s" deleted
```

# Download and share SBOM for the build

The screenshot displays the Red Hat Hybrid Cloud Console interface. The top navigation bar includes 'Red Hat Hybrid Cloud Console', 'Services', 'Favorites', 'Beta on', and a user profile for 'Parag Dave'. The left sidebar shows 'CI/CD' with sub-items 'Overview' and 'Applications'. The main content area is titled 'Pipeline run details' and shows a workflow diagram with steps: 'init', 'clone-repository', 'prefetch-dependencies', 'build-container', and 'inspect-image'. The 'inspect-image' step is expanded to show sub-steps: 'label-check', 'optional-label-check', 'clair-scan', 'clamav-scan', 'sbom-json-check', 'deprecated-base-image-che', and 'sast-snyk-check'. Below the diagram, the pipeline details are listed:

- Name:** devfile-sample-code-with-quarkus-4t7f-d8l7s
- Namespace:** pdave-tenant
- Labels:** appstudio.openshift.io/application=my-quarkus-app, appstudio.openshift.io/component=devfile-sample-code-with-quarkus-4t7f, pipelines.appstudio.openshift.io/type=build (4 more)
- Annotations:**
- Status:** Succeeded
- Pipeline:** docker-build
- Download SBOM:** cosign download sbom quay.io/redhat-user-... (with a 'Copy' button and a red arrow pointing to it)
- Application:** my-quarkus-app
- Vulnerabilities scan:** (with a shield icon)



# Analyze built image for vulnerabilities

The screenshot shows the Red Hat Hybrid Cloud Console interface. The top navigation bar includes the Red Hat logo, 'Red Hat Hybrid Cloud Console', 'Services', 'Favorites', 'Beta on' toggle, and a user profile for 'Parag Dave'. The left sidebar has 'CI/CD' selected, with sub-items 'Overview' and 'Applications'. The main content area displays details for a pipeline run with ID 'devfile-sample-code-with-quarkus-417f-001/s'. The pipeline is in a 'Succeeded' state. Key details include: Namespace: pdave-tenant; Labels: appstudio.openshift.io/application=my-quarkus-app, appstudio.openshift.io/component=devfile-sample-code-with-quarkus-417f, pipelines.appstudio.openshift.io/type=build; Annotations: build.appstudio.openshift.io/image=quay.io/redhat-user-workloads/pdave-tenant/my-quarkus-app/devfil..., build.appstudio.openshift.io/repo=https://github.com/devfile-samples/devfile-sample-code-with-quarkus..., build.appstudio.redhat.com/bundle-quay.io/redhat-appstudio-tekton-catalog/pipeline-docker-build:30bc...; Created at: Apr 13, 2023, 12:42 PM; Duration: 3 minutes 47 seconds. The 'Results' section contains a table with the following data:

Name	Value
IMAGE_URL	quay.io/redhat-user-workloads/pdave-tenant/my-quarkus-app/devfile-sample-code-with-quarkus-417f/build-c9fba-1681411376
IMAGE_DIGEST	sha256:e69da0bef4450801dabf320bb7a0d35b915c2c851c672794d8e1b348b1c8c86a

A red arrow points to the IMAGE\_URL value. The 'Vulnerabilities scan' section shows a status of '-' and a 'Download SBOM' button with the text 'cosign download sbom quay.io/redhat-user-...'. Other sections include 'Application: my-quarkus-app', 'Component: devfile-sample-code-with-quarkus-417f', and 'Source: devfile-samples/devfile-sample-code-with-quarkus'. A 'Related pipelines' section shows '0 pipelines'. A red notification icon is visible in the bottom right corner of the console window.

37

# Drill down on details of vulnerabilities detected

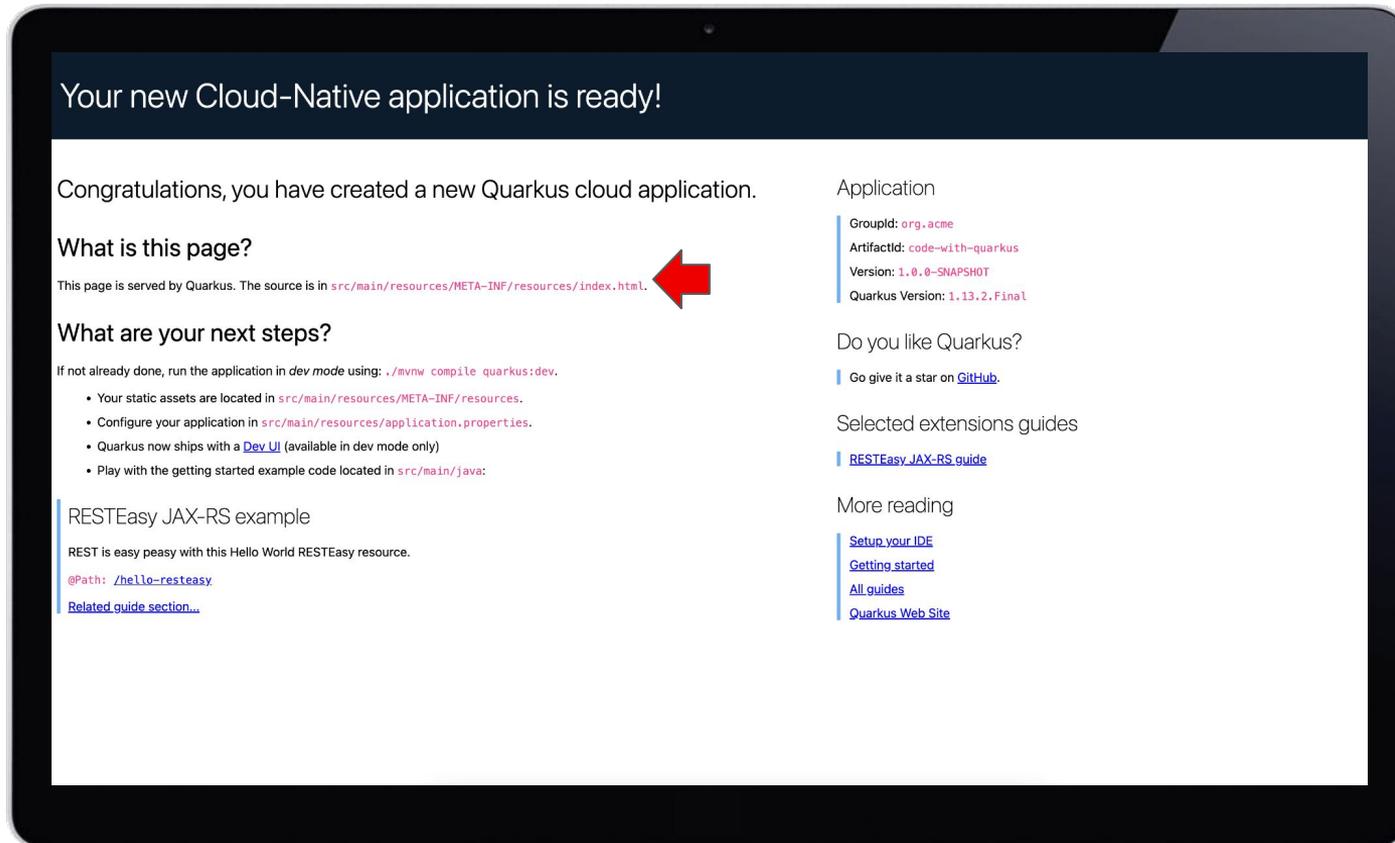
The screenshot shows the Red Hat Quay.io security scanner interface. At the top, there's a navigation bar with 'RED HAT Quay.io', 'EXPLORE', 'TUTORIAL', and 'PRICING'. A search bar and 'SIGN IN' link are on the right. The main content area displays a summary for a specific workload: 'redhat-user-workloads/pdave-tenant/my-quark...' with ID 'e69da0bef445'. A donut chart shows that 16% of vulnerabilities are high-level (3) and 84% are medium-level (16). A summary states: 'Quay Security Scanner has detected 19 vulnerabilities. Patches are available for 18 vulnerabilities.' Below this, a table lists the vulnerabilities. A red arrow points to the first entry in the table, which is expanded to show its description: 'Denial of Service (DoS)'. The table columns are: CVE, SEVERITY, PACKAGE, CURRENT VERSION, FIXED IN VERSION, and INTRODUCED IN LAYER.

CVE	SEVERITY	PACKAGE	CURRENT VERSION	FIXED IN VERSION	INTRODUCED IN LAYER
SNYK-JAVA-IONETTY-1584064	High	io.netty.netty-codec	4.1.49.Final	4.1.68.Final	ENTRYPOINT [ "/deployments/run-java.sh" ]
DESCRIPTION Denial of Service (DoS)					
SNYK-JAVA-IONETTY-1584063	High	io.netty.netty-codec	4.1.49.Final	4.1.68.Final	ENTRYPOINT [ "/deployments/run-java.sh" ]
SNYK-JAVA-IQQUARKUS-3149918	High	io.quarkus.quarkus-vertx-http	1.13.2.Final	2.14.2	ENTRYPOINT [ "/deployments/run-java.sh" ]
SNYK-JAVA-IONETTY-1082236	Medium	io.netty.netty-transport	4.1.49.Final	4.1.59.Final	ENTRYPOINT [ "/deployments/run-java.sh" ]
SNYK-JAVA-IONETTY-1082234	Medium	io.netty.netty-common	4.1.49.Final	4.1.59.Final	ENTRYPOINT [ "/deployments/run-java.sh" ]
SNYK-JAVA-IONETTY-2812456	Medium	io.netty.netty-common	4.1.49.Final	4.1.77.Final	ENTRYPOINT [ "/deployments/run-java.sh" ]
SNYK-JAVA-IQQUARKUS-2331899	Medium	io.quarkus.quarkus-core	1.13.2.Final	2.6.0.CR1	ENTRYPOINT [ "/deployments/run-java.sh" ]

# Access application deployed to a bundled Kubernetes environment

The screenshot displays the Red Hat Hybrid Cloud Console interface. The top navigation bar includes 'Red Hat Hybrid Cloud Console', 'Services', 'Favorites', and a 'Beta on' toggle. The user 'Parag Dave' is logged in. The left sidebar shows 'CI/CD' with sub-items 'Overview' and 'Applications'. The main content area shows the 'my-quarkus-app' page, which is in a 'Succeeded' state. A red arrow points to the application name. A tooltip titled 'Static environments' is visible, explaining that a static environment is a set of compute resources bundled together for development, testing, and staging. Below the application name, there are tabs for 'Overview', 'Activity', 'Components', 'Integration tests', and 'Environments'. The 'Lifecycle' section shows a pipeline visualization with steps: 'Components' (1), 'Builds' (1), 'No tests set', and 'Static environments' (1). The 'Static environments' step is highlighted as 'Succeeded'. Below the pipeline, there is an 'Add component' button and a 'Recent commits' section with a placeholder for monitoring CI/CD activity.

# Share and collaborate on the application from a unique URL



The screenshot shows a web browser window with a dark blue header that reads "Your new Cloud-Native application is ready!". Below the header, the main content area is white. On the left side, there are three sections: "Congratulations, you have created a new Quarkus cloud application.", "What is this page?" with a red arrow pointing to the source code path, and "What are your next steps?" with a list of instructions. On the right side, there are four sections: "Application" with metadata, "Do you like Quarkus?" with a GitHub link, "Selected extensions guides" with a RESTEasy guide link, and "More reading" with links to IDE setup, getting started, all guides, and the Quarkus web site.

## Your new Cloud-Native application is ready!

Congratulations, you have created a new Quarkus cloud application.

### What is this page?

This page is served by Quarkus. The source is in `src/main/resources/META-INF/resources/index.html`.

### What are your next steps?

If not already done, run the application in *dev mode* using: `./mvnw compile quarkus:dev`.

- Your static assets are located in `src/main/resources/META-INF/resources`.
- Configure your application in `src/main/resources/application.properties`.
- Quarkus now ships with a [Dev UI](#) (available in dev mode only)
- Play with the getting started example code located in `src/main/java`:

#### RESTEasy JAX-RS example

REST is easy peasy with this Hello World RESTEasy resource.

@Path: [/hello--resteasy](#)

[Related guide section...](#)

#### Application

GroupId: `org.acme`  
ArtifactId: `code-with-quarkus`  
Version: `1.0.0-SNAPSHOT`  
Quarkus Version: `1.13.2.Final`

#### Do you like Quarkus?

Go give it a star on [GitHub](#).

#### Selected extensions guides

[RESTEasy JAX-RS guide](#)

#### More reading

[Setup your IDE](#)  
[Getting started](#)  
[All guides](#)  
[Quarkus Web Site](#)

# Set approval gates using enterprise contracts available out-of-the-box

The screenshot displays a web interface for a pipeline run. At the top, the breadcrumb navigation shows: WS samburrai \*\*\* | Applications > partner-catalog-ec \*\*\* > Pipeline runs > partner-catalog-ec-p8klr-59mmr. Below this, the pipeline run name 'partner-catalog-ec-p8klr-59mmr' is shown with a green 'Succeeded' status and an 'Actions' dropdown menu. The 'Security' tab is selected, showing a section titled 'Testing apps against Enterprise Contract'. This section explains that Enterprise Contract is used for verifying application snapshots and validating them against a policy, with a link to 'Enterprise Contract Policies'. Below the text is a 'Results' section with a 'Results summary' showing 0 Failed, 0 Warning, and 20 Success. A table lists 20 rules, all of which passed with a 'Success' status. The table has columns for 'Rules', 'Status', 'Message', and 'Component'. A blue circular icon is visible in the bottom right corner of the interface.

WS samburrai \*\*\* | Applications > partner-catalog-ec \*\*\* > Pipeline runs > partner-catalog-ec-p8klr-59mmr

partner-catalog-ec-p8klr-59mmr Succeeded Actions

Details Task runs Logs Security

### Testing apps against Enterprise Contract

Enterprise Contract is a set of tools for verifying the provenance of application snapshots and validating them against a clearly defined policy. The Enterprise Contract policy is defined using the [rego policy language](#) and is described here in [Enterprise Contract Policies](#).

**Results** Results summary

Component Status Filter by rule... Failed 0 Warning 0 Success 20

Rules	Status	Message	Component
> Task bundle references not empty	Success	-	partner-catalog-ec-Olmh
> Tasks defined using bundle references	Success	-	partner-catalog-ec-Olmh
> Known attestation type found	Success	-	partner-catalog-ec-Olmh
> PipelineRun attestation found	Success	-	partner-catalog-ec-Olmh
> Allowed base image registry prefixes list was provided	Success	-	partner-catalog-ec-Olmh
> Base image task result was provided	Success	-	partner-catalog-ec-Olmh
> Base image comes from permitted registry	Success	-	partner-catalog-ec-Olmh
> Attestation signature check passed	Success	-	partner-catalog-ec-Olmh
> Attestation syntax check passed	Success	-	partner-catalog-ec-Olmh
> Image signature check passed	Success	-	partner-catalog-ec-Olmh
> Blocking CVE check	Success	-	partner-catalog-ec-Olmh
> CVE scan results found	Success	-	partner-catalog-ec-Olmh

# Suspicious build activity is automatically blocked from production

The screenshot shows the Red Hat Hybrid Cloud Console interface for a CI/CD pipeline named 'my-quarkus-app'. The pipeline is currently in a 'Succeeded' state. A tooltip for 'Static environments' is displayed, explaining that they are sets of compute resources used for development, testing, and staging. The tooltip lists two environments: 'development' (Succeeded) and 'pre-prod-openshift-aws' (Pending). A red arrow points to the 'Pending' status of the 'pre-prod-openshift-aws' environment. The pipeline lifecycle shows a sequence of steps: Components (2), Builds (2), Tests (2), Static environments (2), No releases set, and No managed environments yet. Below the pipeline, there is a table of recent commits.

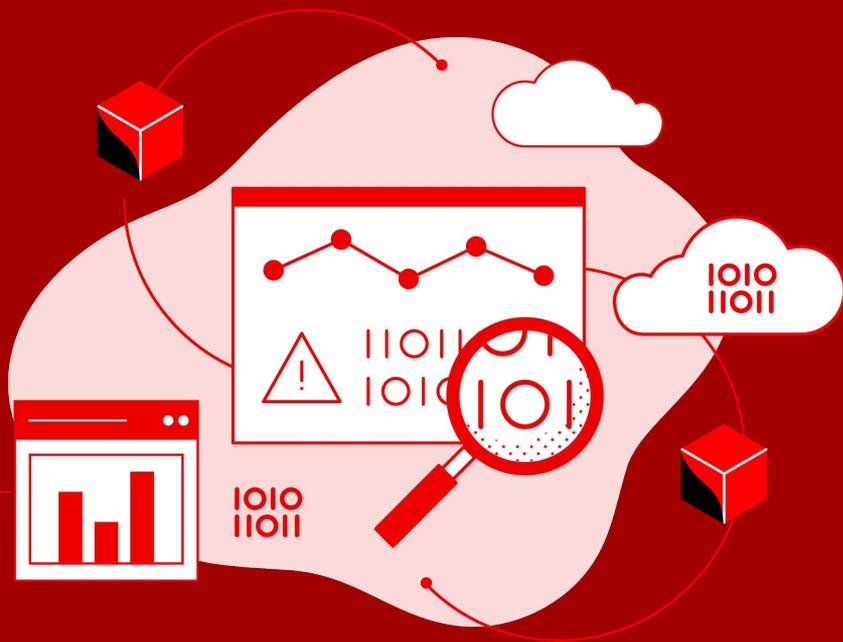
Name	Branch	Component	By user	Latest commit at	Status
<a href="#">Update server.js</a> <a href="#">aa81a5</a>	main	nodejs-rhtap-example-iez8	pdaverh	Apr 14, 2023, 11:43 AM	Succeeded
<a href="#">Update server.js</a> <a href="#">5c0095</a>	main	nodejs-rhtap-example-iez8	pdaverh	Apr 14, 2023, 11:33 AM	Succeeded
<a href="#">Update server.js</a> <a href="#">ea3b3b</a>	main	nodejs-rhtap-example-iez8	pdaverh	Apr 14, 2023, 11:17 AM	Succeeded

# Check details of flagged items in Enterprise Service Contract

The screenshot displays the Red Hat Hybrid Cloud Console interface. The breadcrumb navigation shows: Applications > enterprise-contract > Pipeline runs > enterprise-contract-fmh45-6gwcr. The pipeline run status is 'Succeeded'. The 'verify' task is selected, and its logs are expanded. A red arrow points to the 'verify' task in the task list.

```
violations:
- metadata:
  code: hermetic_build_task.build_task_not_hermetic
  effective_on: "2022-01-01T00:00:00Z"
  msg: Build task was not invoked with hermetic parameter
- metadata:
  code: tasks.missing_required_task
  effective_on: "2022-01-01T00:00:00Z"
  msg: Required task "clair-scan" is missing
- metadata:
  code: tasks.missing_required_task
  effective_on: "2022-01-01T00:00:00Z"
  msg: Required task "clamav-scan" is missing
- metadata:
  code: tasks.missing_required_task
  effective_on: "2022-01-01T00:00:00Z"
  msg: Required task "prefetch-dependencies" is missing
- metadata:
  code: tasks.missing_required_task
  effective_on: "2022-01-01T00:00:00Z"
  msg: Required task "unity-inspect-image" is missing
```

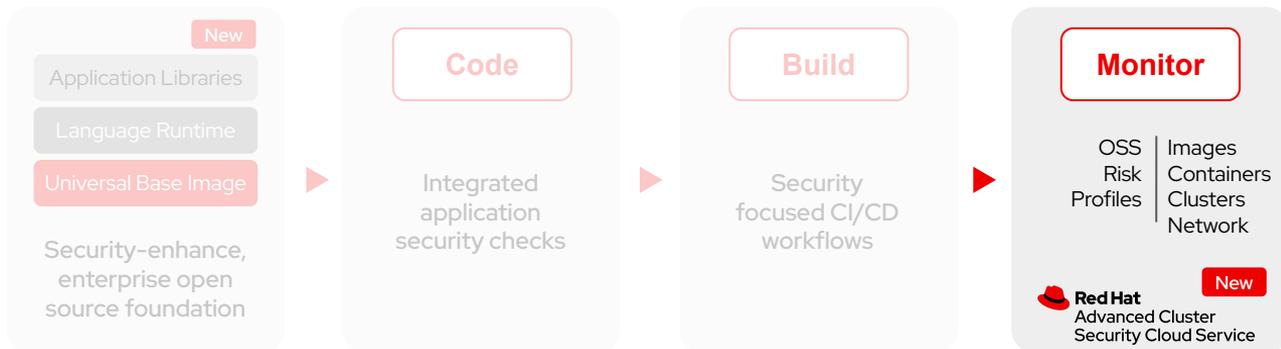
TEMO



*Continuously  
monitor security  
at runtime*

# Continuous security monitoring at runtime

Cut down alert noise, fatigue to eliminate production downtimes



Standardize, share and store with centralized access controls



Flexibility and choice of any environment



Physical



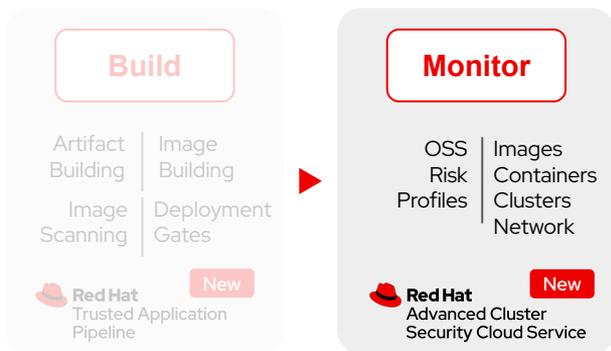
Virtual



Hybrid



## Monitor and identify runtime security incidents

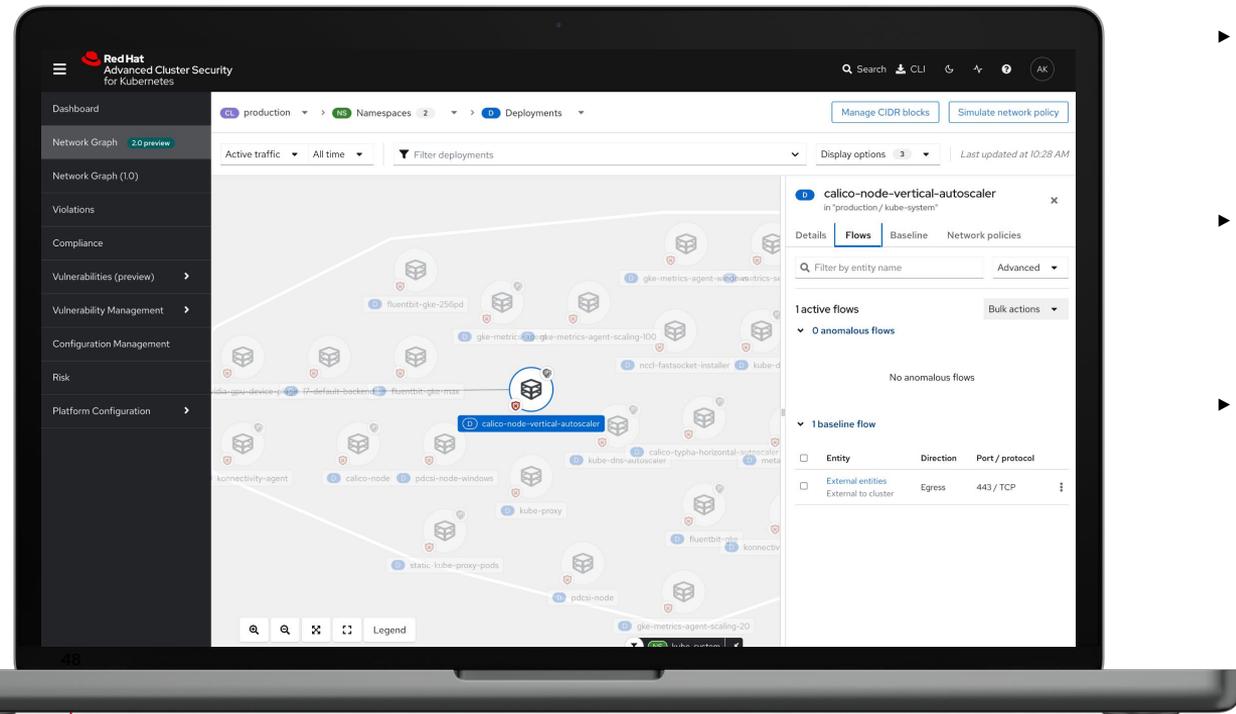


Reduce noise, alert fatigue for shorter time to response

- ▶ Continuous improvement from runtime to build
- ▶ Detect and respond to suspicious activity
- ▶ Runtime vulnerability scanning and management
- ▶ Audit for compliance across hundreds of controls
- ▶ Expedite incident response to reduce down times

# Continuous runtime security and response with minimal false positives

- ▶ Prevent high risk workloads from being deployed or running using OOTB deploy-time and runtime policies
- ▶ Harden workloads by enforcing network policies in accordance with the principles of least privilege
- ▶ Monitor for anomalous behavior indicative of a threat, and configure custom policies and responses, providing feedback to developers



# Continuously monitor for anomalous behaviour at runtime

The screenshot displays the Red Hat Advanced Cluster Security for Kubernetes dashboard. The top navigation bar includes the Red Hat logo, the product name, a search icon, a CLI download icon, a settings icon, a refresh icon, a help icon, and a user profile icon labeled 'BS'. Below the navigation bar, a summary row shows metrics for 1 Cluster, 7 Nodes, 164 Violations, 166 Deployments, 126 Images, and 779 Secrets, with a timestamp 'Last updated 2/20/2023 at 3:50 PM'. The main content area is titled 'Dashboard' and includes a 'Resources' filter set to 'All clusters' and 'All namespaces'. A section titled '164 policy violations by severity' features four horizontal bars representing Low (94), Medium (63), High (6), and Critical (1) counts. Below this, a section for 'Most recent violations with critical severity' lists a violation: 'Iptables Executed in Privileged Container' on node 'ovnkube-node' at 02/20/2023 | 3:50:00PM. The bottom section, 'Images at most risk', contains a table with columns for Images, Risk priority, Critical CVEs, and Important CVEs. A red arrow points to the first row of the table.

Images	Risk priority	Critical CVEs	Important CVEs
redhat-appstudio/user-workload	1	2 fixable	6 fixable
observatorium/token-refresher	2	1 fixable	0 fixable

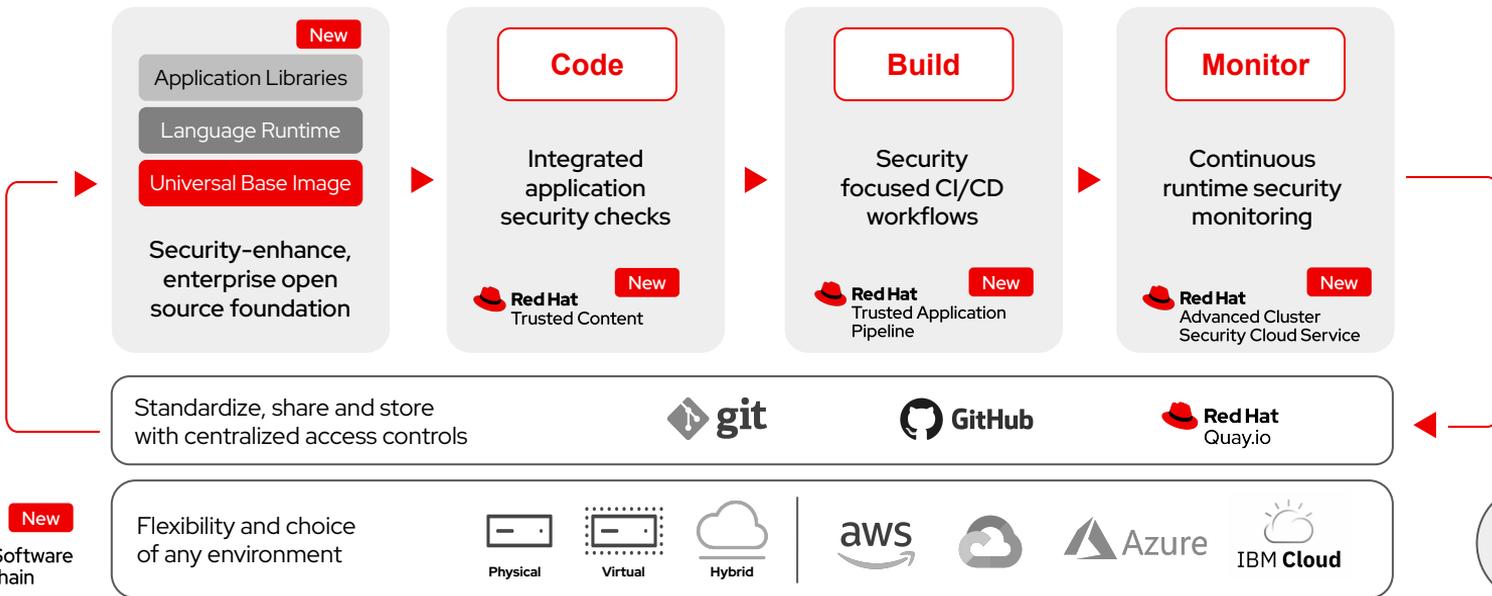
# Harden workloads by enforcing network policies

The screenshot displays the Red Hat Advanced Cluster Security for Kubernetes dashboard. The left sidebar contains navigation options: Dashboard, Network Graph (2.0 preview), Network Graph (1.0), Violations, Compliance, Vulnerabilities (preview), Vulnerability Management, Configuration Management, Risk, and Platform Configuration. The main area shows a network graph with various pods and services. A red arrow points to the 'Network Graph' menu item. The top navigation bar includes 'production' namespace, 'Namespaces 2', and 'Deployments'. The right sidebar shows details for the 'calico-node-vertical-autoscaler' deployment, including tabs for 'Details', 'Flows', 'Baseline', and 'Network policies'. The 'Flows' tab is active, showing 0 anomalous flows and 1 baseline flow. A table below lists the baseline flow details.

Entity	Direction	Port / protocol
External entities External to cluster	Egress	443 / TCP

# Layered security throughout the stack and lifecycle

Achieve business agility while meeting security requirements



51



TEMO

# Thank you

Learn more

[red.ht/trusted](https://red.ht/trusted)

 [linkedin.com/company/red-hat](https://linkedin.com/company/red-hat)

 [youtube.com/user/RedHatVideos](https://youtube.com/user/RedHatVideos)

 [facebook.com/redhatinc](https://facebook.com/redhatinc)

 [twitter.com/RedHat](https://twitter.com/RedHat)