Google Cloud

Kubernetes DevSecOps

Peek Into

Secure Software Supply Chain

for Kubernetes

Oct 19th, 2022

Shawn Ho, AppMod Specialist

Container Advantages in Security

Containers are short lived and frequently re-deployed; you can constantly be patching.

Containers are intentionally immutable; a modified container is a built-in security alert.

Good security defaults are one line changes; setting secure configurations is easy.

With isolation technologies, you can increase security without adding resources.

Agenda

01 Security Challenges from OSS: 開源軟體對資安的挑戰

02 SDLC & SBOM & SLSA: 社群的努力

03 Software Delivery Shield: 平台部署+持續防護

04 Software Delivery Shield End-to-End 的實現

Agenda

01 Security Challenges from OSS: 開源軟體對資安的挑戰

02 SDLC & SBOM & SLSA: 社群的努力

03 Software Delivery Shield: 平台部署+持續防護

04 Software Delivery Shield在GCP上的實現

Google Cloud

On the rise

Supply Chain Attacks

Increasingly, the software development lifecycle (SDLC) itself has become a vector for attacks.

The recent **Log4Shell**, **SolarWinds**, **Kaseya**, and **Codecov** hacks highlight vulnerable surface areas exposed in the SDLC.

650 %

Surge in OSS supply chain attacks [1]

81 %

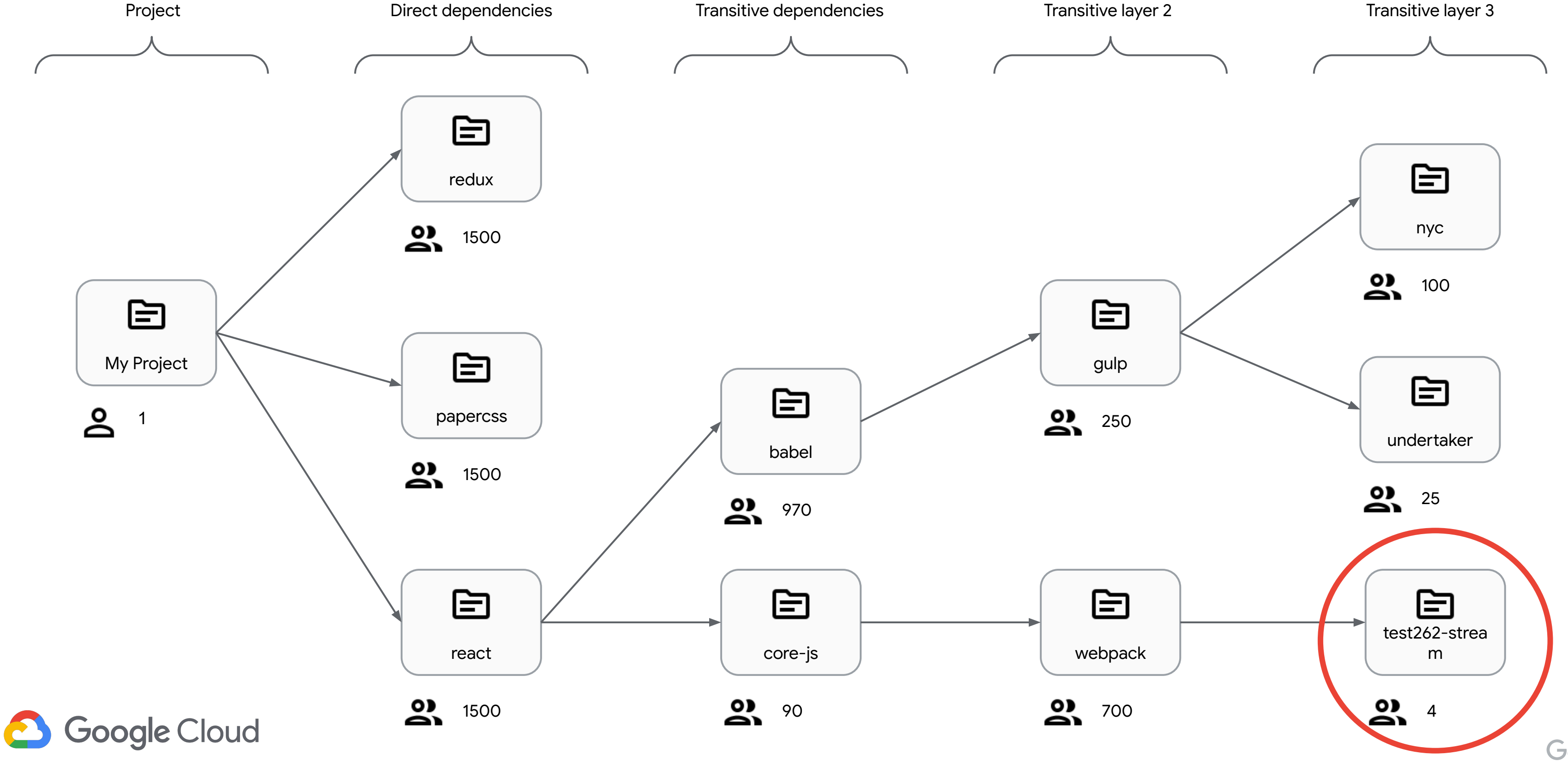
Commercial code bases have OSS vulnerabilities [2]

45 %

of organizations worldwide will have experienced attacks on their software supply chains by 2025 [3]

[1] Sonatype, 2021 - State of the Software Supply Chain
[2] Synopsys, 2022 - Open Source Security and Risk Analysis Report
[3] Gartner, 2021 - How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks

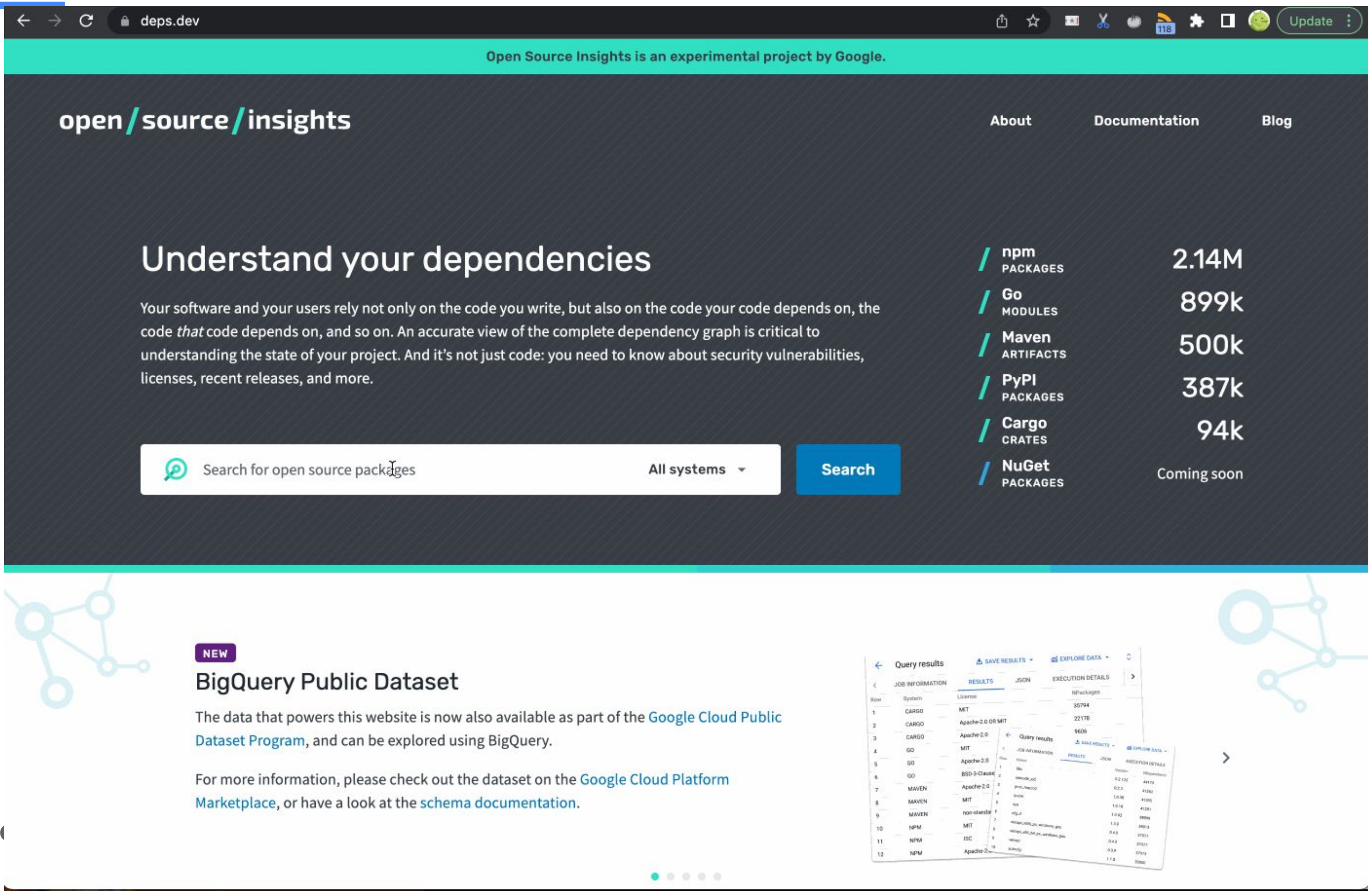
站在巨人肩膀上的代價：開源軟體的高依存性



Google Cloud

Google

deps.dev



Open Source Insights is an experimental project by Google.

open/source/insights

Understand your dependencies

Your software and your users rely not only on the code you write, but also on the code your code depends on, the code *that* code depends on, and so on. An accurate view of the complete dependency graph is critical to understanding the state of your project. And it's not just code: you need to know about security vulnerabilities, licenses, recent releases, and more.

npm PACKAGES	2.14M
Go MODULES	899k
Maven ARTIFACTS	500k
PyPI PACKAGES	387k
Cargo CRATES	94k
NuGet PACKAGES	Coming soon

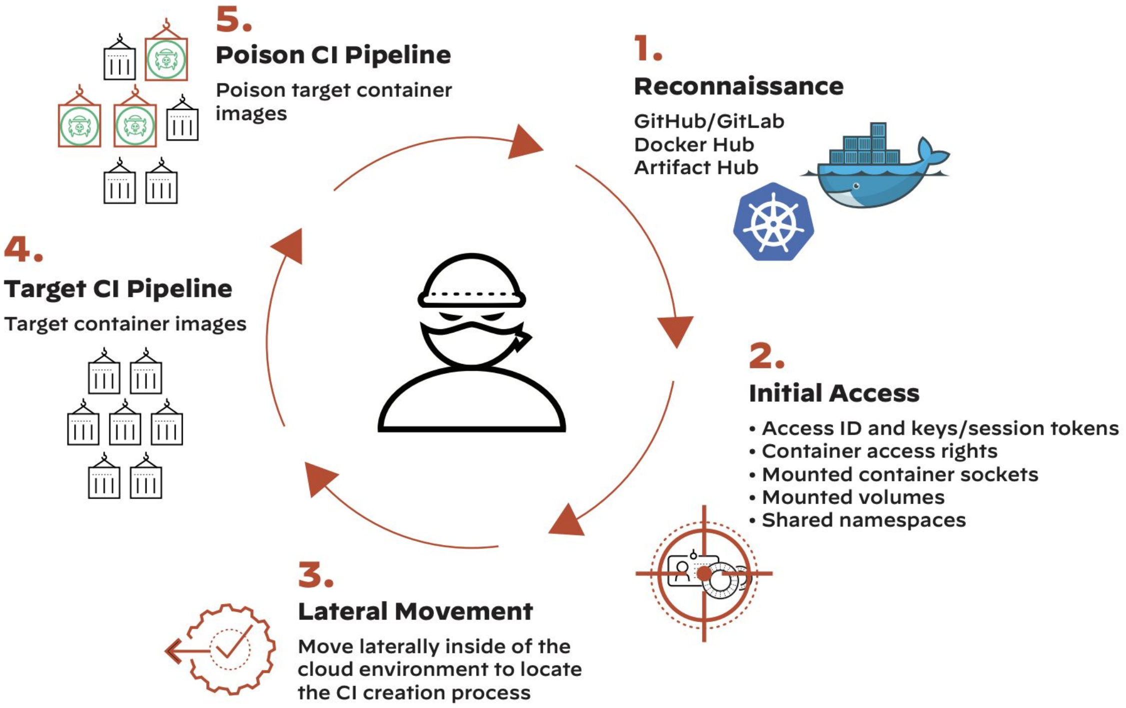
NEW

BigQuery Public Dataset

The data that powers this website is now also available as part of the Google Cloud Public Dataset Program, and can be explored using BigQuery.

For more information, please check out the dataset on the Google Cloud Platform Marketplace, or have a look at the schema documentation.

CI Pipeline 的攻擊手法

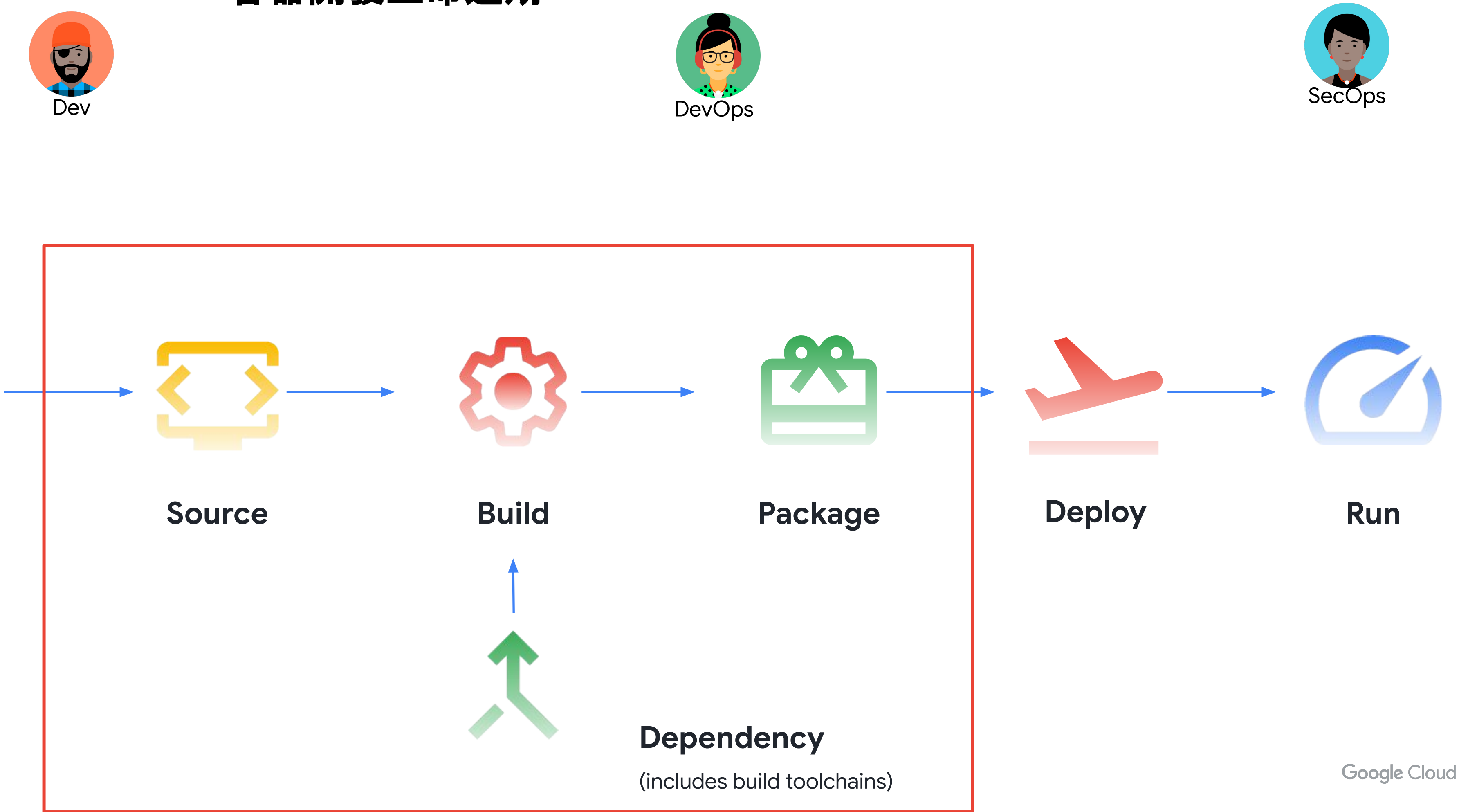


5.
Poison CI Pipeline
Poison target container images
1.
Reconnaissance
GitHub/GitLab
Docker Hub
Artifact Hub
2.
Initial Access
• Access ID and keys/session tokens
• Container access rights
• Mounted container sockets
• Mounted volumes
• Shared namespaces
3.
Lateral Movement
Move laterally inside of the cloud environment to locate the CI creation process
4.
Target CI Pipeline
Target container images

Google Cloud

Source: Palo Alto Network Unit 42 Container Threat Report 2H 2021

容器開發生命週期



Dev

DevOps

SecOps

Source

Build

Package

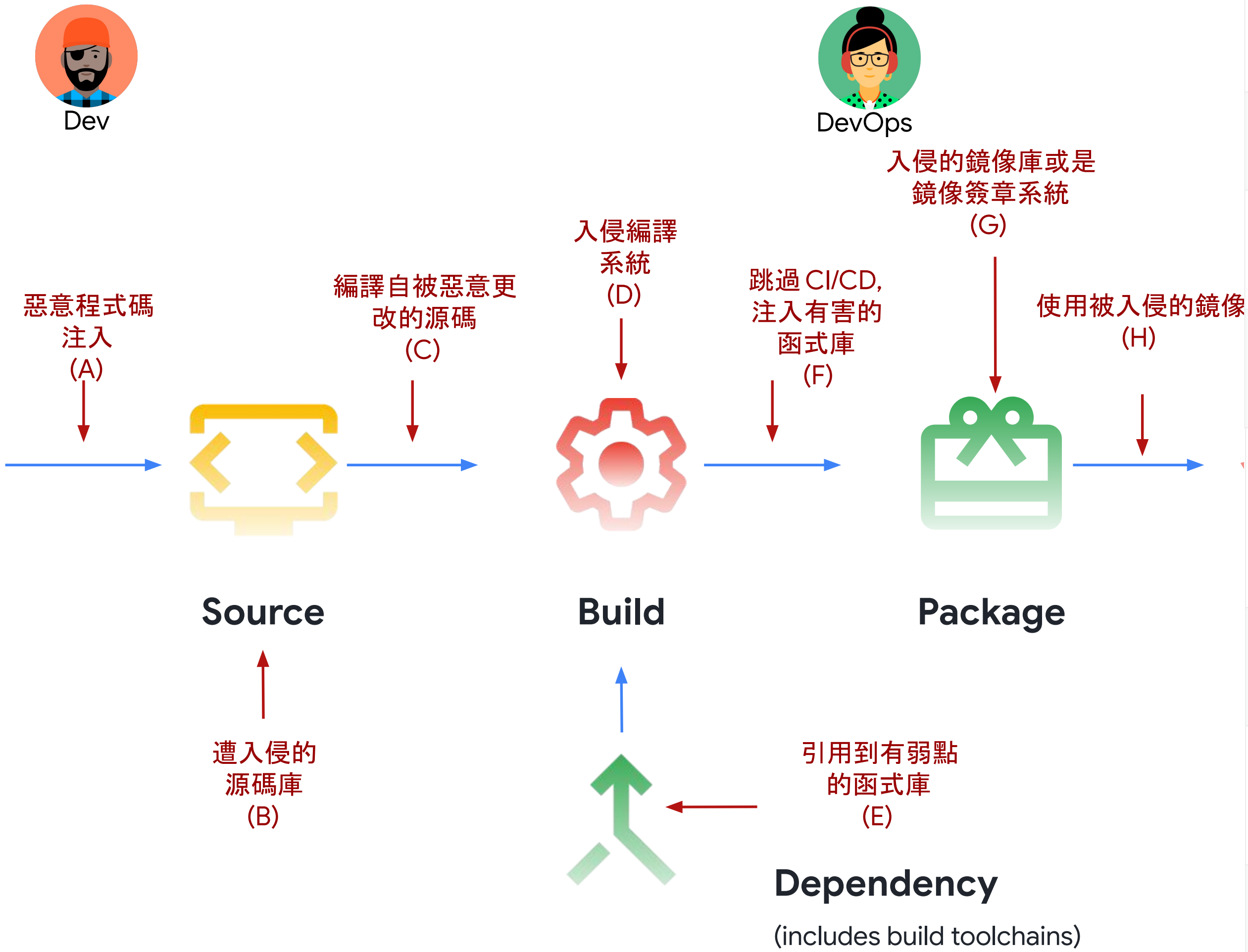
Deploy

Run

Dependency

(includes build toolchains)

攻擊介面 (Attack Vectors - Build)



Dev

DevOps

惡意程式碼注入 (A)

編譯自被惡意更改的源碼 (C)

入侵編譯系統 (D)

跳過 CI/CD, 注入有害的函式庫 (F)

入侵的鏡像庫或是鏡像簽章系統 (G)

使用被入侵的鏡像 (H)

Source

Build

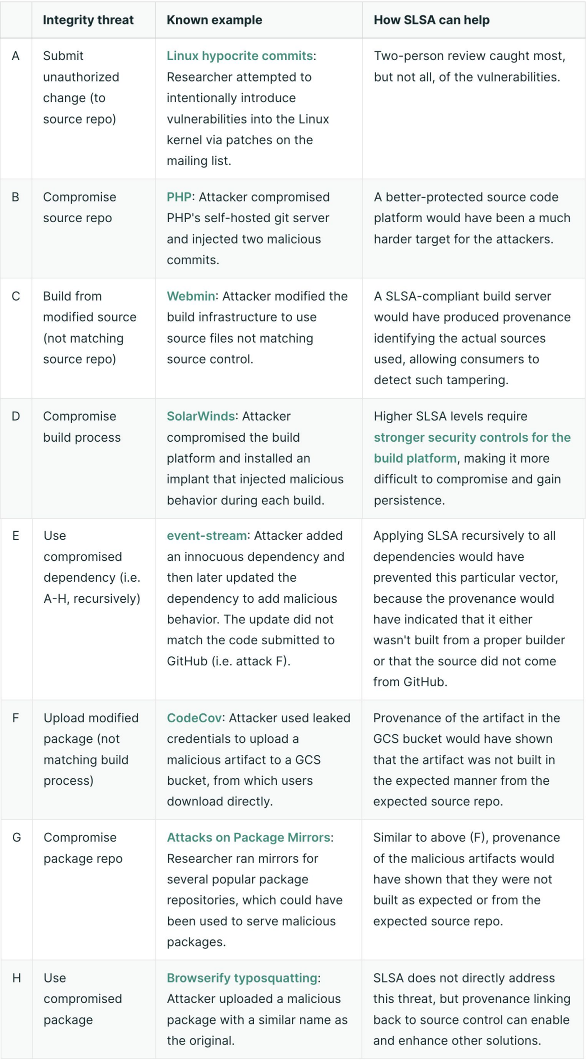
Package

遭入侵的源碼庫 (B)

引用到有弱點的函式庫 (E)

Dependency

(includes build toolchains)



	Integrity threat	Known example	How SLSA can help
A	Submit unauthorized change (to source repo)	Linux hypocrite commits: Researcher attempted to intentionally introduce vulnerabilities into the Linux kernel via patches on the mailing list.	Two-person review caught most, but not all, of the vulnerabilities.
B	Compromise source repo	PHP: Attacker compromised PHP's self-hosted git server and injected two malicious commits.	A better-protected source code platform would have been a much harder target for the attackers.
C	Build from modified source (not matching source repo)	Webmin: Attacker modified the build infrastructure to use source files not matching source control.	A SLSA-compliant build server would have produced provenance identifying the actual sources used, allowing consumers to detect such tampering.
D	Compromise build process	SolarWinds: Attacker compromised the build platform and installed an implant that injected malicious behavior during each build.	Higher SLSA levels require stronger security controls for the build platform, making it more difficult to compromise and gain persistence.
E	Use compromised dependency (i.e. A-H, recursively)	event-stream: Attacker added an innocuous dependency and then later updated the dependency to add malicious behavior. The update did not match the code submitted to GitHub (i.e. attack F).	Applying SLSA recursively to all dependencies would have prevented this particular vector, because the provenance would have indicated that it either wasn't built from a proper builder or that the source did not come from GitHub.
F	Upload modified package (not matching build process)	CodeCov: Attacker used leaked credentials to upload a malicious artifact to a GCS bucket, from which users download directly.	Provenance of the artifact in the GCS bucket would have shown that the artifact was not built in the expected manner from the expected source repo.
G	Compromise package repo	Attacks on Package Mirrors: Researcher ran mirrors for several popular package repositories, which could have been used to serve malicious packages.	Similar to above (F), provenance of the malicious artifacts would have shown that they were not built as expected or from the expected source repo.
H	Use compromised package	Browserify typosquatting: Attacker uploaded a malicious package with a similar name as the original.	SLSA does not directly address this threat, but provenance linking back to source control can enable and enhance other solutions.

Agenda

01 Security Challenges from OSS: 開源軟體對資安的挑戰

02 SDLC & SBOM & SLSA: 社群的努力

03 Software Delivery Shield: 平台部署+持續防護

04 Software Delivery Shield在GCP上的實現

Industry & Government Priority

enisa

EUROPEAN UNION AGENCY FOR CYBERSECURITY

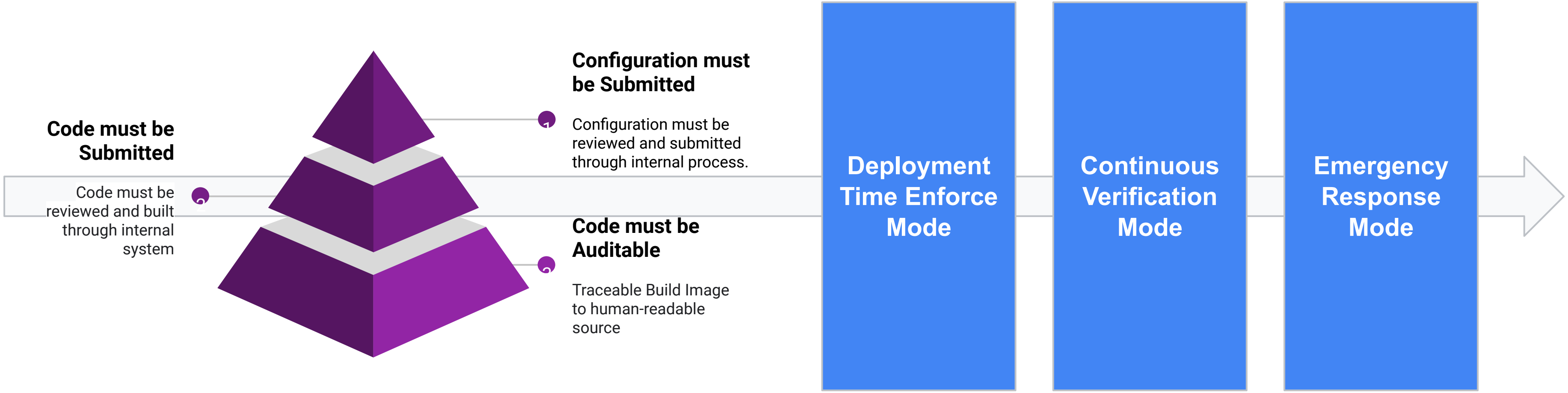
NIST

National Institute of Standards and Technology
U.S. Department of Commerce

The recent **U.S. executive order, European Union Agency for Cybersecurity,** and others are requiring governmental contractors and essential utilities to follow a high standard of **SDLC security** has accelerated the urgency and timeline.

We anticipate these standards to become broad market norms.

Google BAB (Binary Authorization For Borg) Process:



Code must be Submitted

Code must be reviewed and built through internal system

Configuration must be Submitted

Configuration must be reviewed and submitted through internal process.

Code must be Auditable

Traceable Build Image to human-readable source

Deployment Time Enforce Mode

Continuous Verification Mode

Emergency Response Mode

Service Specified Policy (Build)

Enforcement Mode (Run)

Reference

SLSA (“ salsa”) framework

Supply Chain Levels for Software Artifacts

A security framework to secure three main areas involved in software artifact creation:

Build Integrity

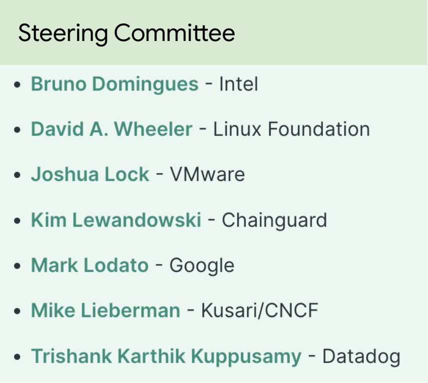
- Modification of code after source control
- Compromised build platforms
- Bypassing CI/CD

Source Integrity

- Available change history
- Code review
- Compromised source control systems

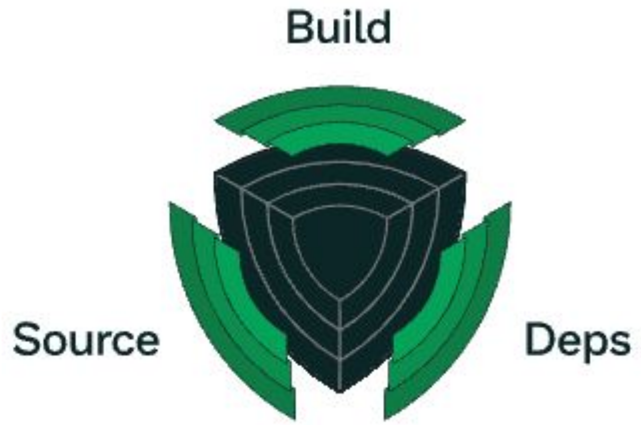
Dependencies

- Applying SLSA checks recursively to dependencies
- Dependency confusion



Steering Committee

- Bruno Domingues - Intel
- David A. Wheeler - Linux Foundation
- Joshua Lock - VMware
- Kim Lewandowski - Chainguard
- Mark Lodato - Google
- Mike Lieberman - Kusari/CNCF
- Trishank Karthik Kuppusamy - Datadog



SLSA Levels

See: slsa.dev

Measure integrity levels for build, source and dependencies

Automation & Provenance

Build must be fully scripted/automated and generate provenance

Version Control & Signed Provenance

Requires using version control and hosted build service that generates authenticated provenance

Non-falsifiable, Ephemeral

Builds are fully trustworthy, with identity attestations of underlying build infrastructure/hardware. Ephemeral builds leave nothing behind.

Hermetic Builds, Review

All build inputs/dependencies are specified upfront with no internet egress during the build. Two-party reviews.

SLSA Levels

See: slsa.dev

Measure integrity levels for build, source and dependencies

~~Automation &~~ Provenance

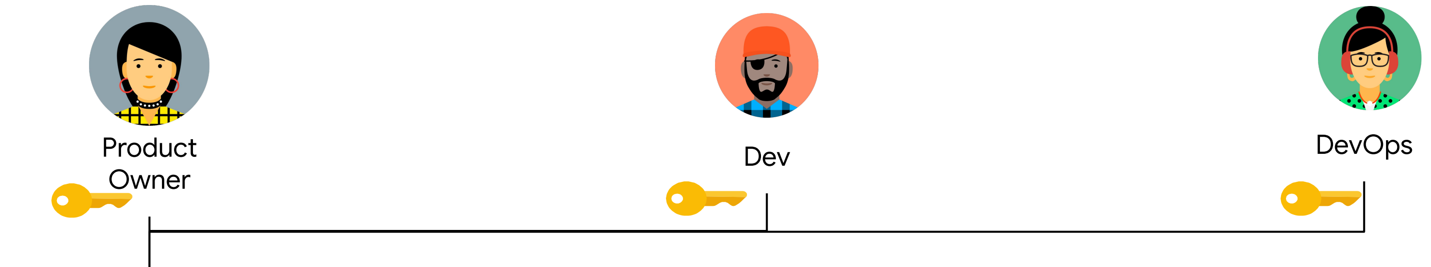
Build must be fully scripted~~/automated~~
and generate provenance

Level 1: Signed Provenance

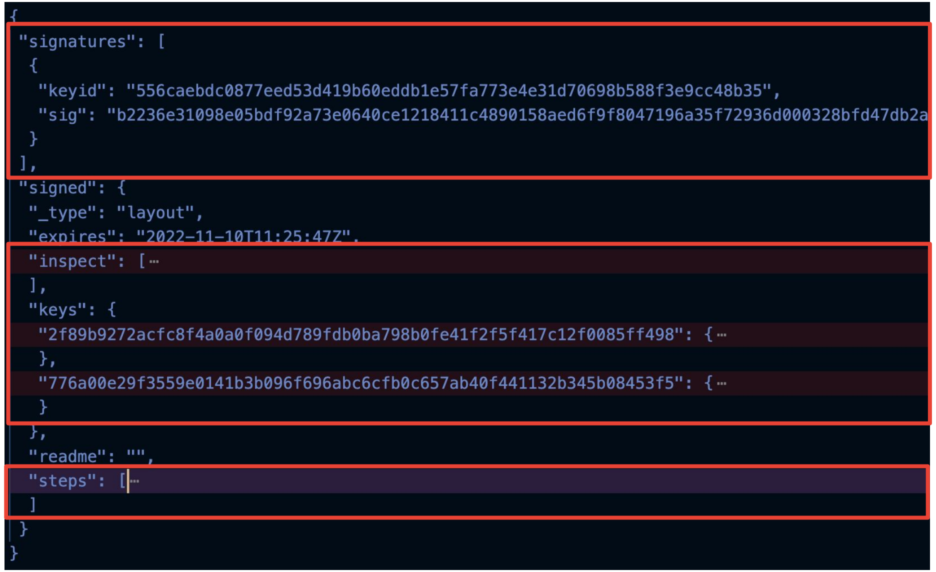
P.O 負責產生Layout

in-toto

A framework to secure the integrity of software supply chains



Product Layout



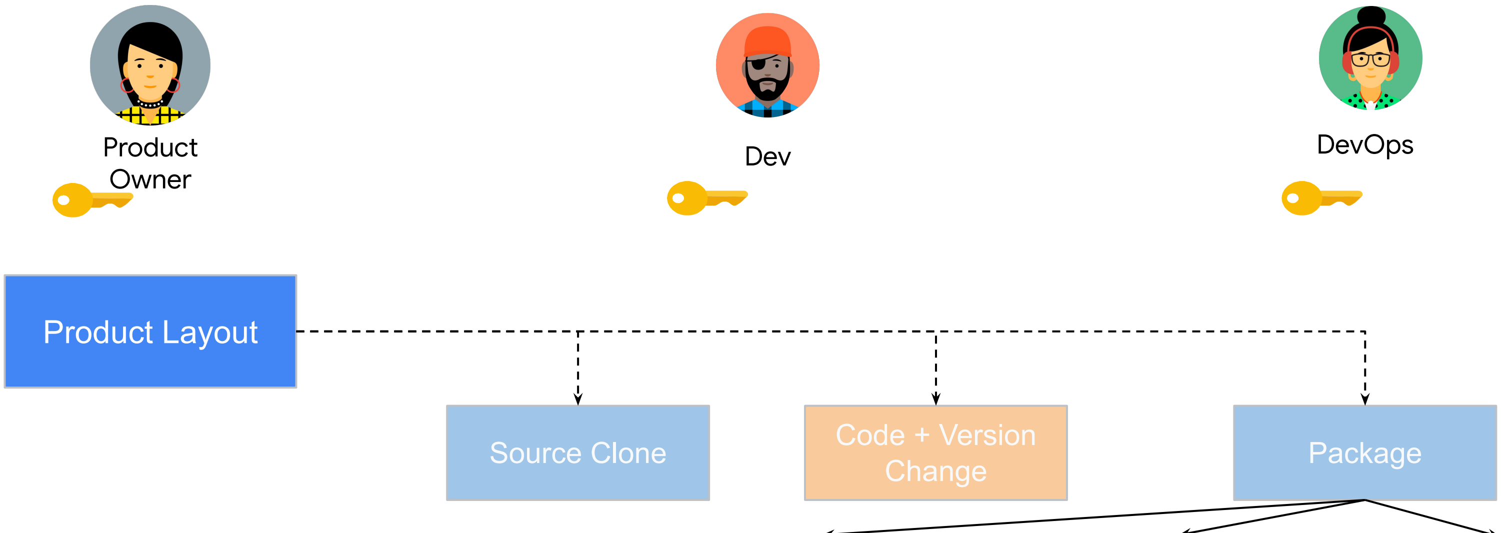
```
{
 "signatures": [
  {
   "keyid": "556caebdc0877eed53d419b60eddb1e57fa773e4e31d70698b588f3e9cc48b35",
   "sig": "b2236e31098e05bdf92a73e0640ce1218411c4890158aed6f9f8047196a35f72936d000328bfd47db2a
  }
 ],
 "signed": {
  "_type": "layout",
  "expires": "2022-11-10T11:25:47Z",
  "inspect": [⋯
  ],
  "keys": {
   "2f89b9272acfc8f4a0a0f094d789fdb0ba798b0fe41f2f5f417c12f0085ff498": {⋯
   },
   "776a00e29f3559e0141b3b096f696abc6cfb0c657ab40f441132b345b08453f5": {⋯
   }
  },
  "readme": "",
  "steps": [⋯
  ]
 }
}
```

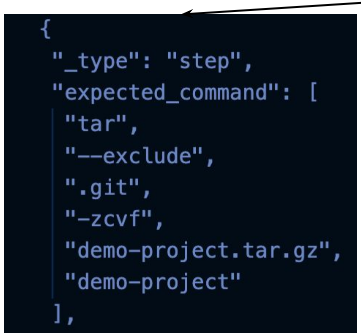
Level 1: Signed Provenance

Layout定義工作流程與對應簽章

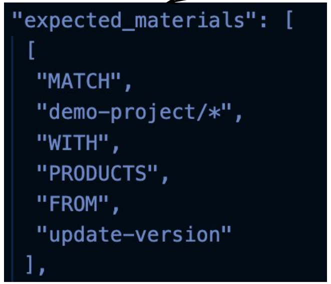
in-toto

A framework to secure the integrity of software supply chains

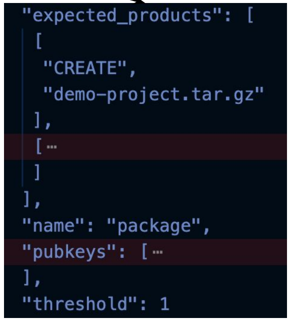




```
{
  "_type": "step",
  "expected_command": [
    "tar",
    "--exclude",
    ".git",
    "-zcvf",
    "demo-project.tar.gz",
    "demo-project"
  ],
```



```
"expected_materials": [
 [
  "MATCH",
  "demo-project/*",
  "WITH",
  "PRODUCTS",
  "FROM",
  "update-version"
 ],
```



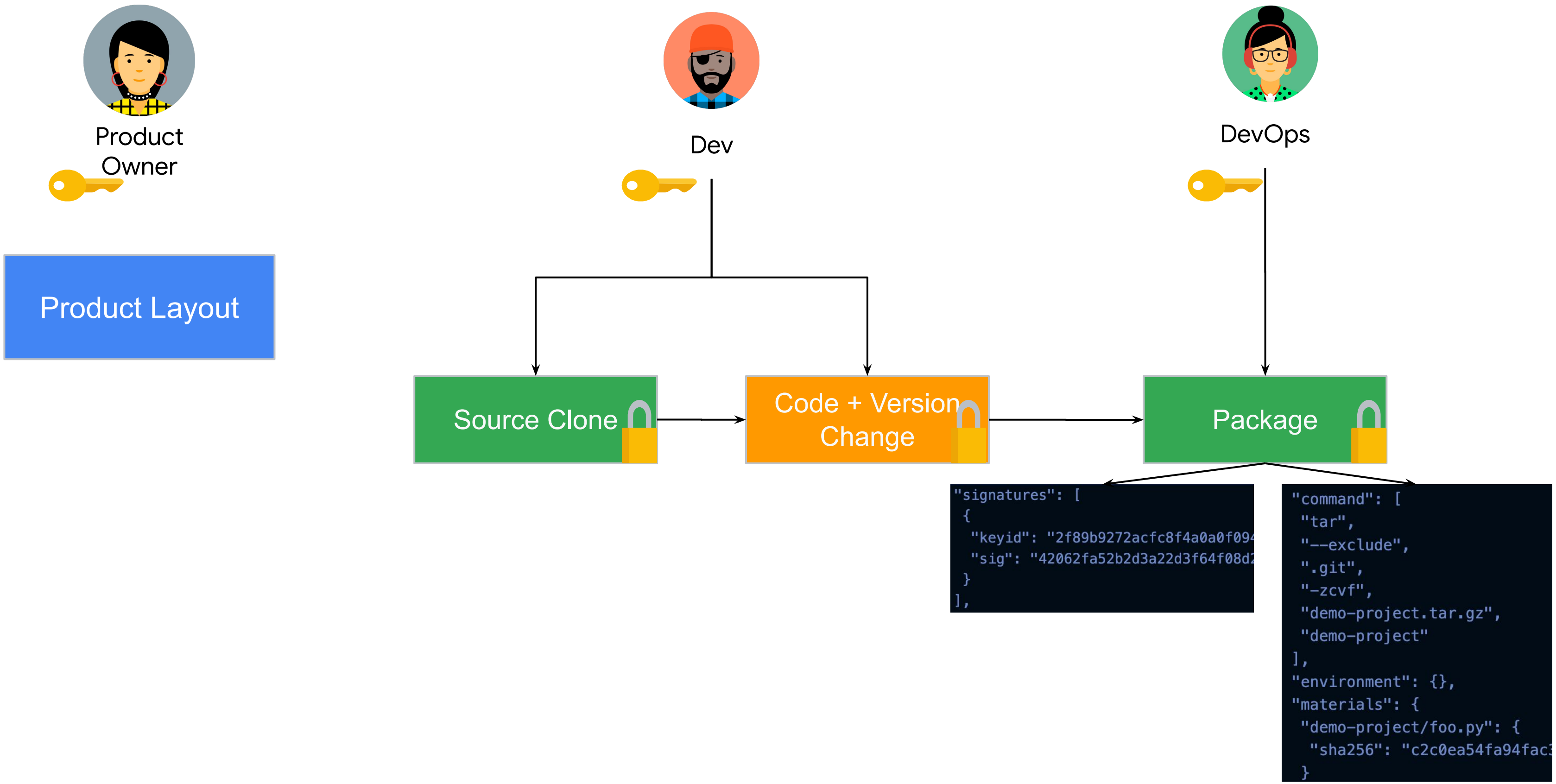
```
"expected_products": [
  [
   "CREATE",
   "demo-project.tar.gz"
  ],
  […
  ]
],
"name": "package",
"pubkeys": […
],
"threshold": 1
```

Level 1: Signed Provenance

執行完成後，由執行者簽章

in-toto

A framework to secure the integrity of software supply chains



```
"signatures": [
 {
  "keyid": "2f89b9272acfc8f4a0a0f094
  "sig": "42062fa52b2d3a22d3f64f08d2
 }
],
```

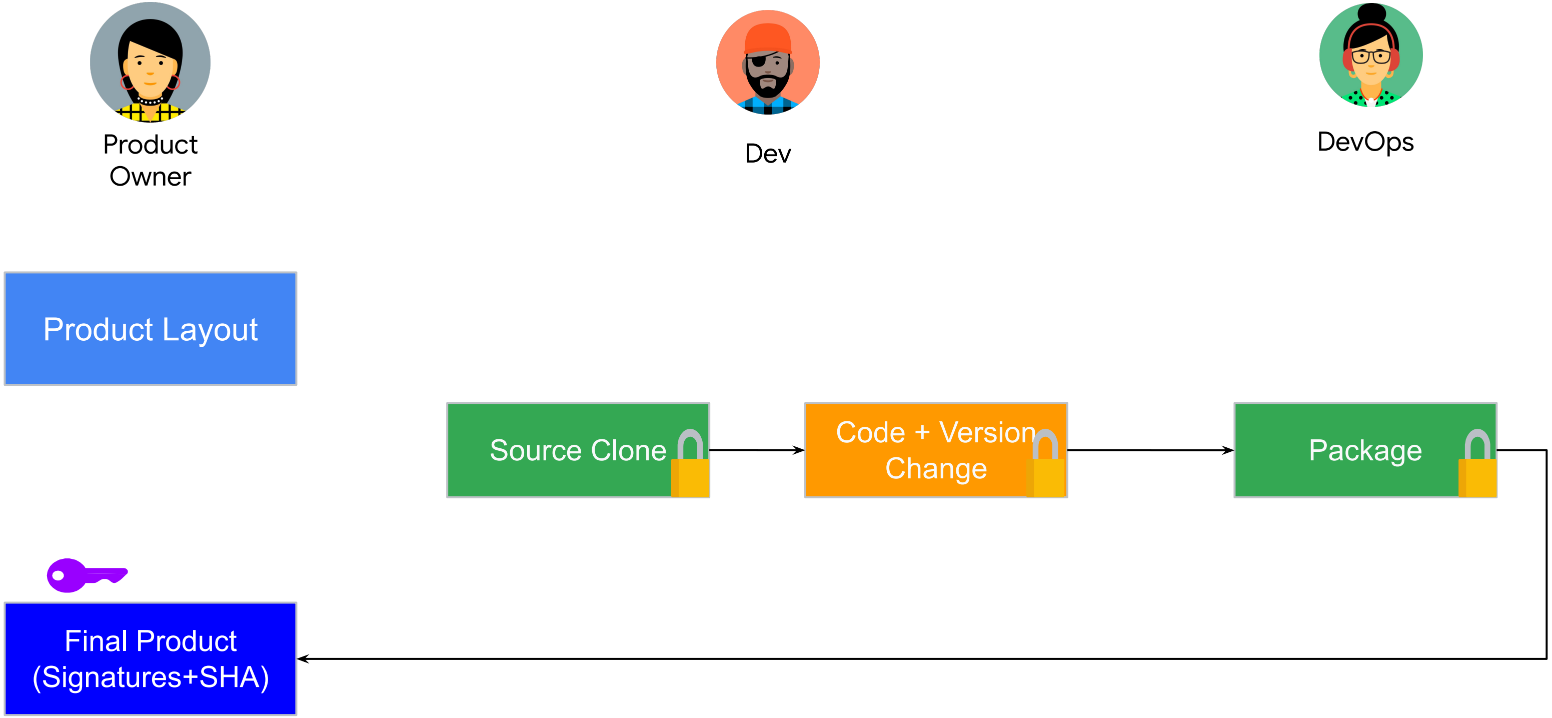
```
"command": [
 "tar",
 "--exclude",
 ".git",
 "-zcvf",
 "demo-project.tar.gz",
 "demo-project"
],
"environment": {},
"materials": {
 "demo-project/foo.py": {
  "sha256": "c2c0ea54fa94fac3
 }
```

Level 1: Signed Provenance

P.O 可依照簽章進行檢查+部署

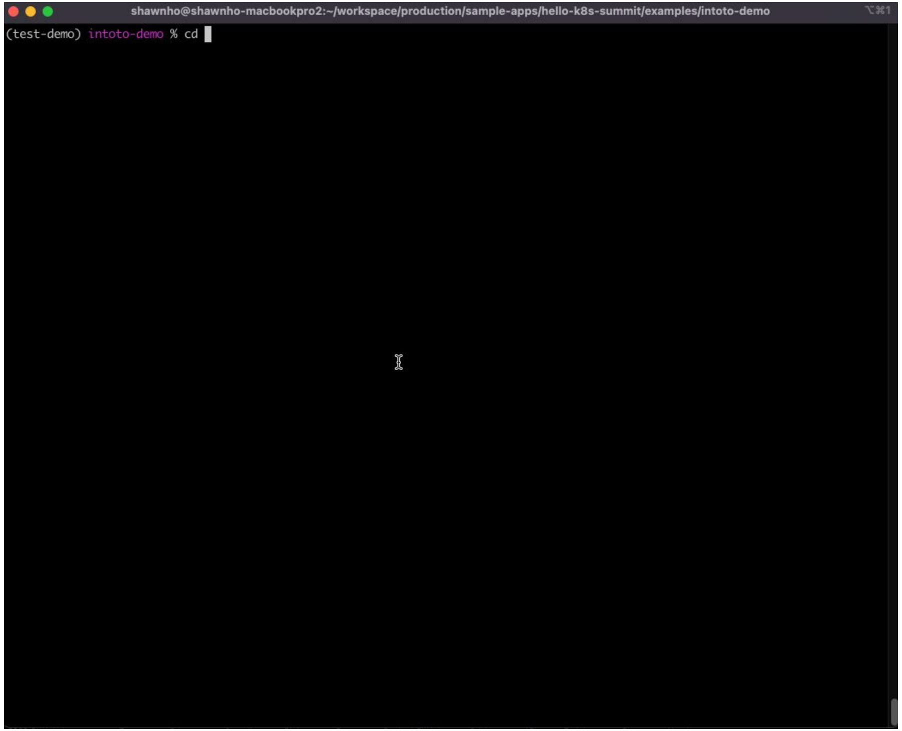
in-toto

A framework to secure the integrity of software supply chains



Demo

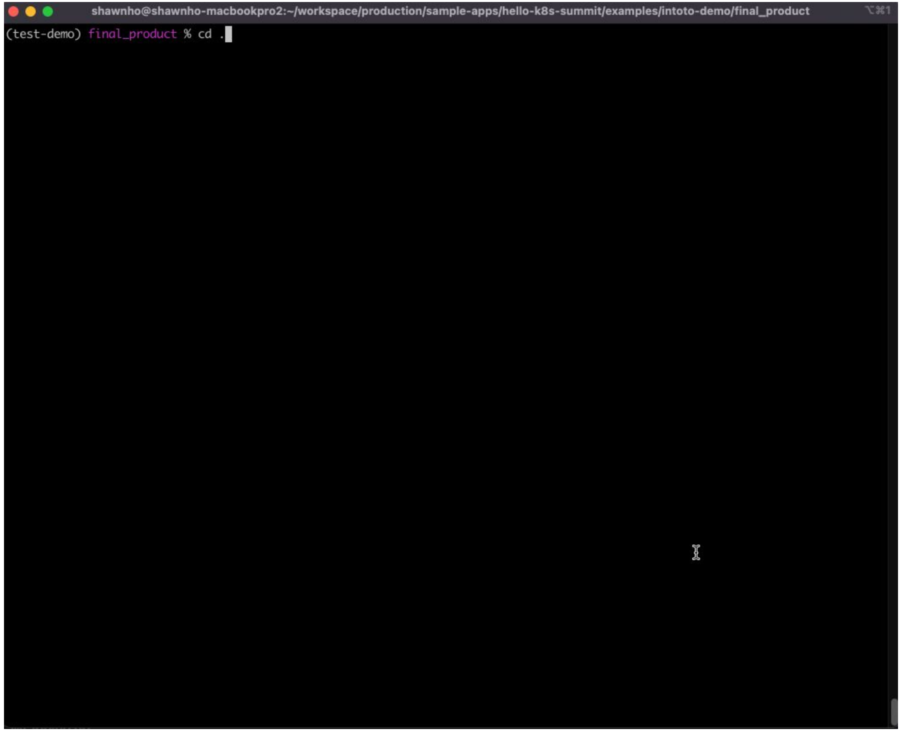
In-toto-verify: Good



shawnho@shawnho-macbookpro2:~/workspace/production/sample-apps/hello-k8s-summit/examples/intoto-demo

```
(test-demo) intoto-demo % cd 
```

In-toto-verify: Tampered



shawnho@shawnho-macbookpro2:~/workspace/production/sample-apps/hello-k8s-summit/examples/intoto-demo/final_product

```
(test-demo) final_product % cd .
```

Agenda

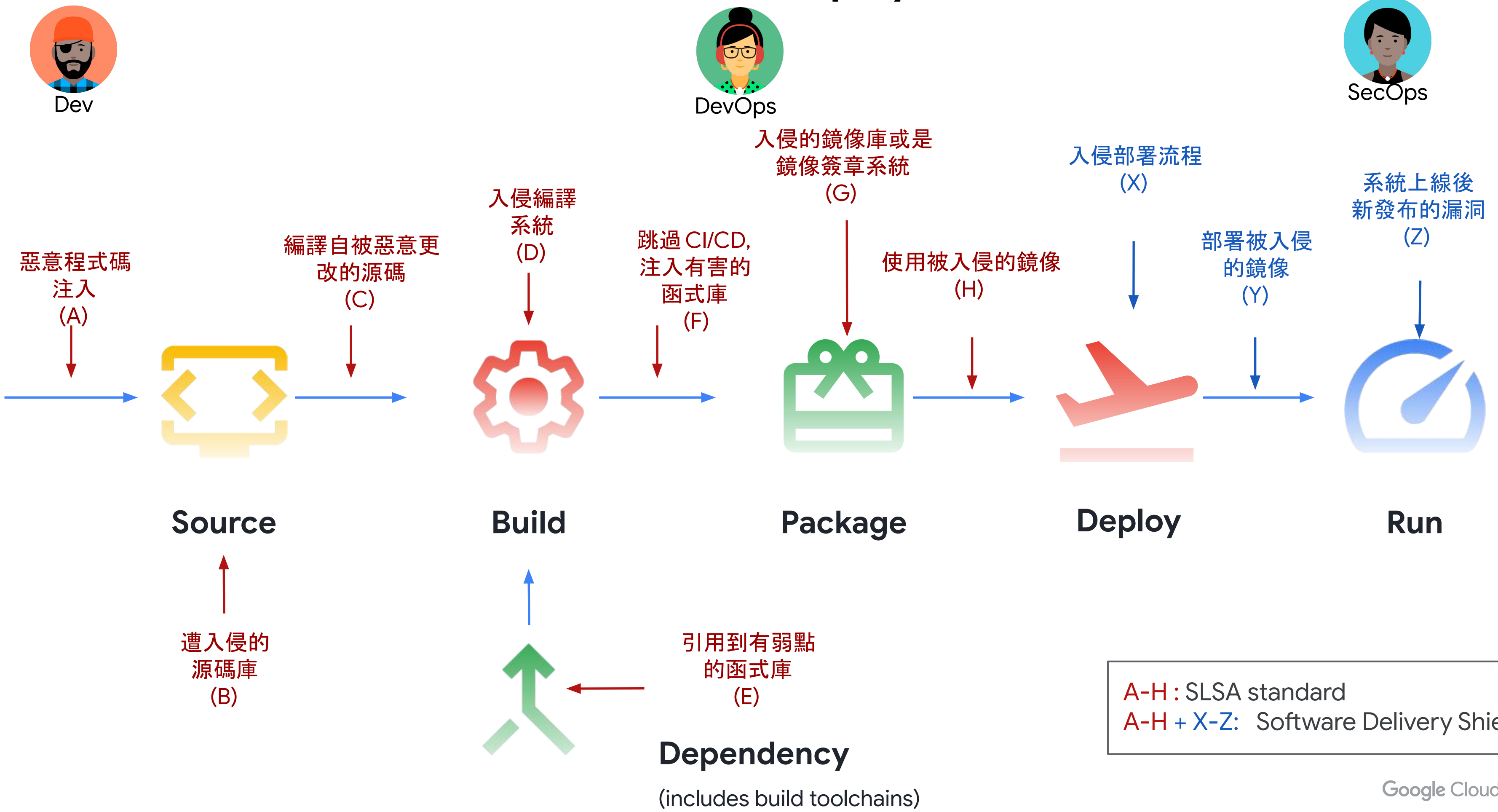
01 Security Challenges from OSS: 開源軟體對資安的挑戰

02 SDLC & SBOM & SLSA: 社群的努力

03 Software Delivery Shield: 平台部署+持續防護

04 Software Delivery Shield在GCP上的實現

攻擊介面 (Attack Vectors) Build + Deploy+Run



A-H : SLSA standard

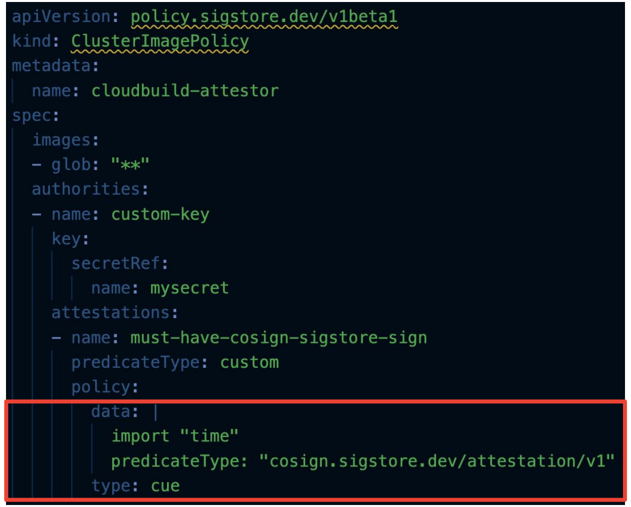
A-H + X-Z: Software Delivery Shield

Google Cloud

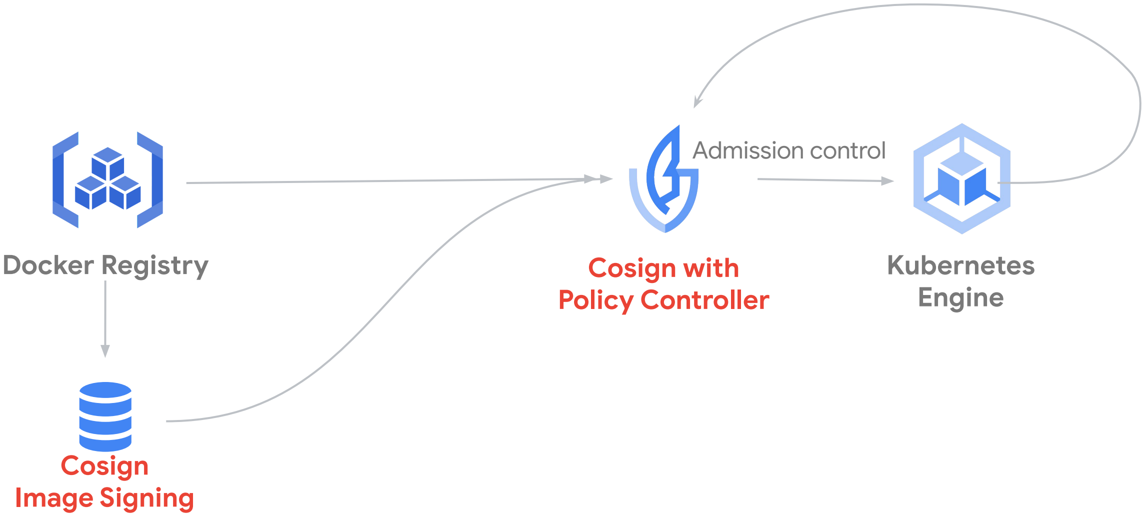
In-Toto概念在K8S的實現:加上K8S平台上的部署政策

In-Toto + Cosign + Policy Controller

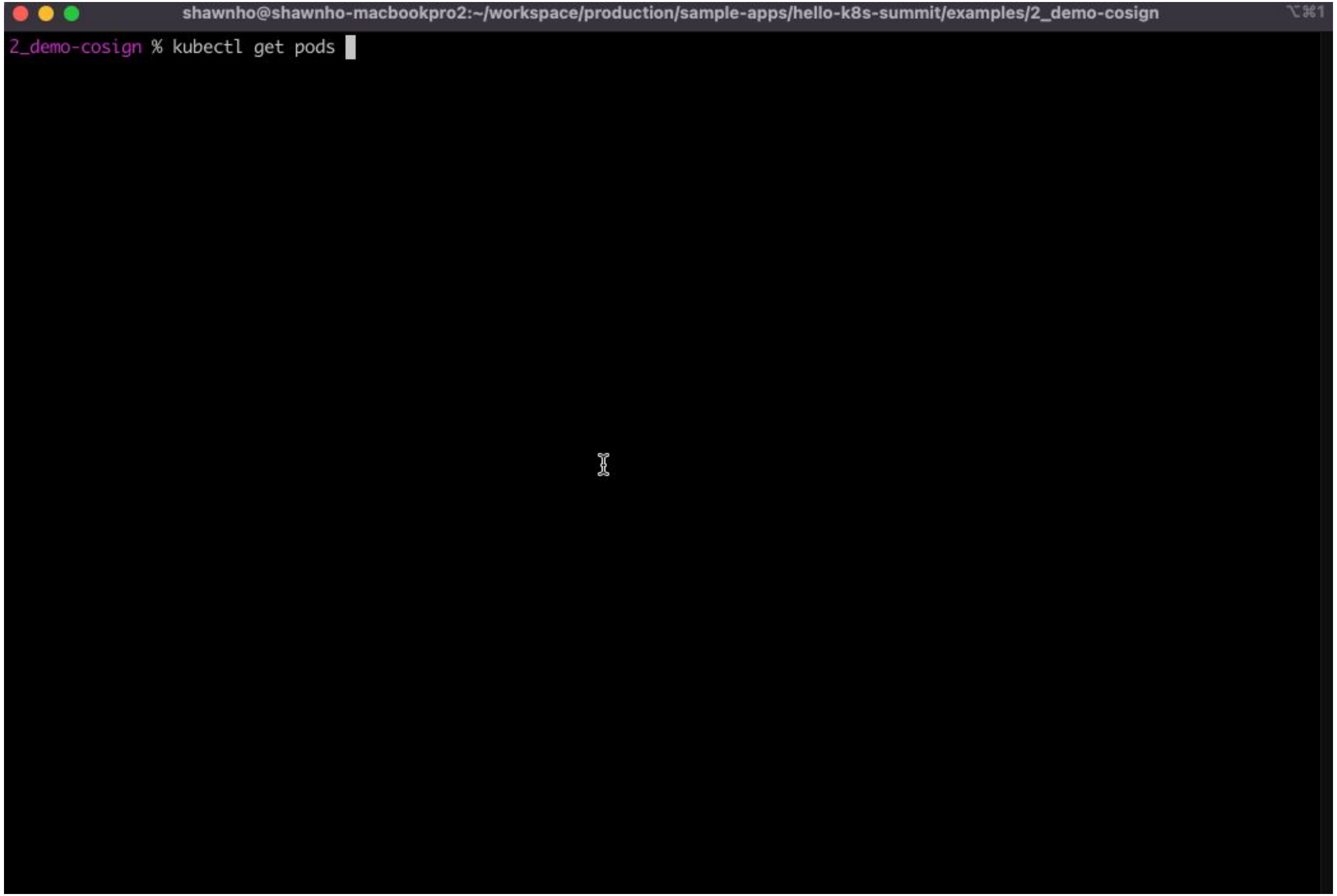
- 安裝Cosign Policy Controller
- 設計ClusterImagePolicy with cue or rego



```
apiVersion: policy.sigstore.dev/v1beta1
kind: ClusterImagePolicy
metadata:
  name: cloudbuild-attestor
spec:
  images:
  - glob: "**"
  authorities:
  - name: custom-key
    key:
      secretRef:
        name: mysecret
    attestations:
    - name: must-have-cosign-sigstore-sign
      predicateType: custom
      policy:
        data: |
          import "time"
          predicateType: "cosign.sigstore.dev/attestation/v1"
        type: cue
```



Demo: From Signing to Deploying



shawnho@shawnho-macbookpro2:~/workspace/production/sample-apps/hello-k8s-summit/examples/2_demo-cosign

```
2_demo-cosign % kubectl get pods
```

準備:

- 展示Policy-Controller
- 標註對象名稱空間policy enabled=true
- 展示對應的ClusterImagePolicy

展示:

- 簽署編譯成功的映像檔
- 部署已帶有簽章的鏡像檔
- 部署未帶有簽章的映像檔

SLSA Security Level 3

See: slsa.dev

Measure integrity levels for build, source and dependencies

1

Automation & Provenance

Build must be fully scripted/automated and generate provenance

2

Version Control & Signed Provenance

Requires using version control and hosted build service that generates authenticated provenance

3

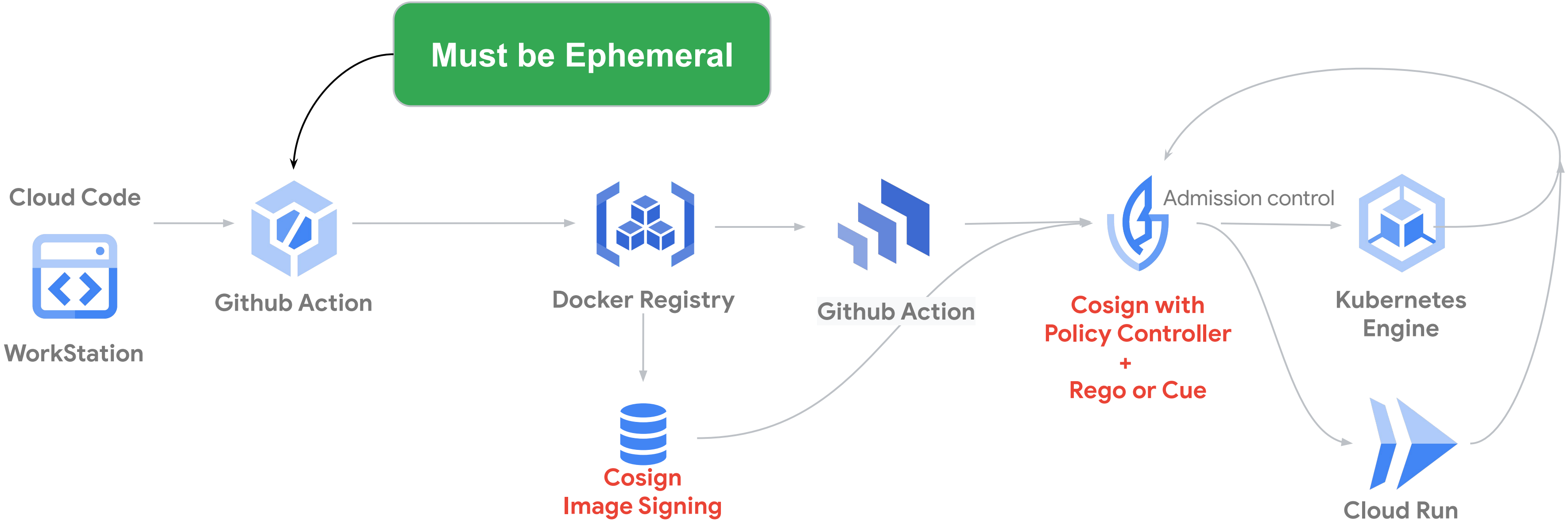
Non-falsifiable, Ephemeral

Builds are fully trustworthy, with identity attestations of underlying build infrastructure/hardware. Ephemeral builds leave nothing behind.

sigstore
cosign

SLSA Security Level 3概念在K8S的實現：

Github Action + Cosign   + Policy Controller



Must be Ephemeral

Cloud Code

WorkStation

Github Action

Docker Registry

Cosign
Image Signing

Github Action

Admission control

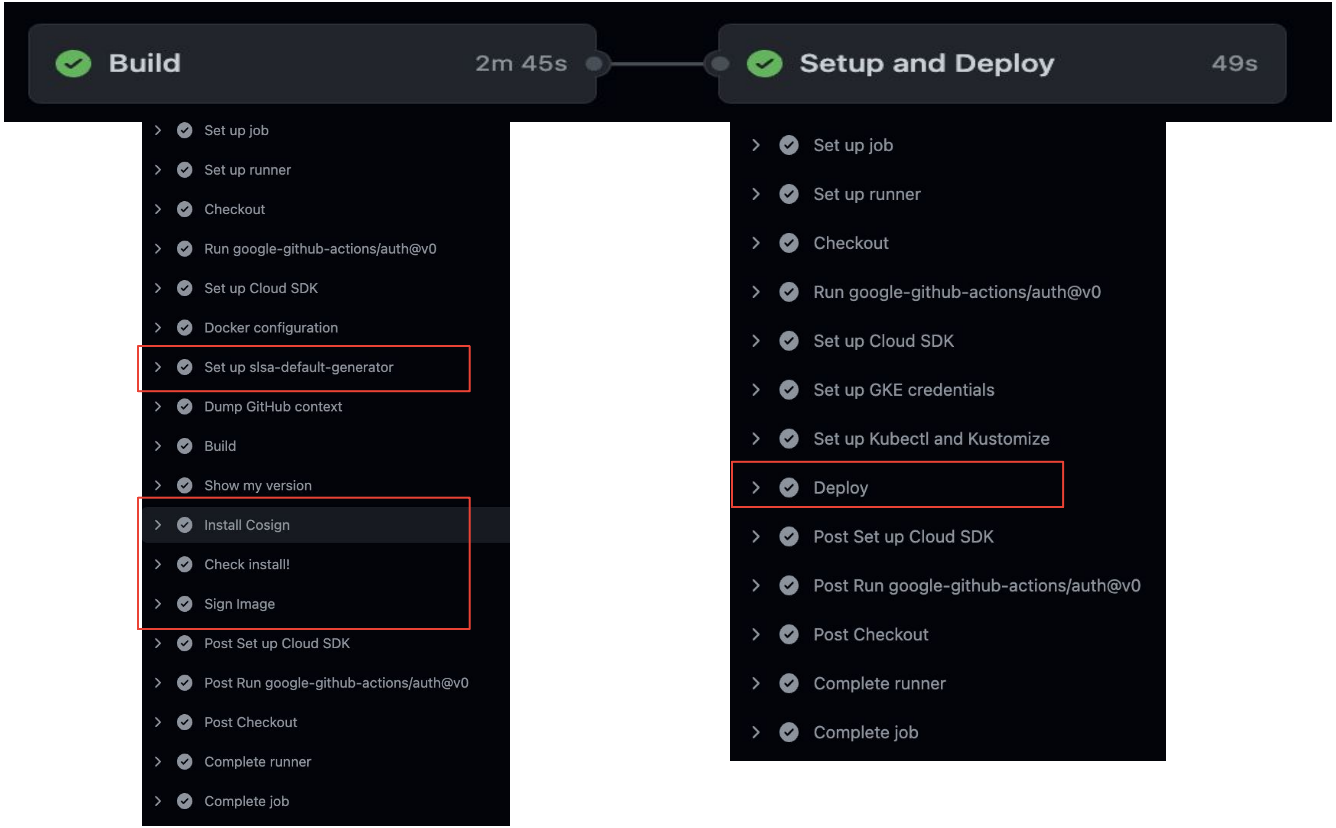
Cosign with
Policy Controller
+
Rego or Cue

Kubernetes
Engine

Cloud Run

Demo: Self-Attested for SLSA Security L3

Github Action + Cosign + Policy Controller



✓ Build 2m 45s

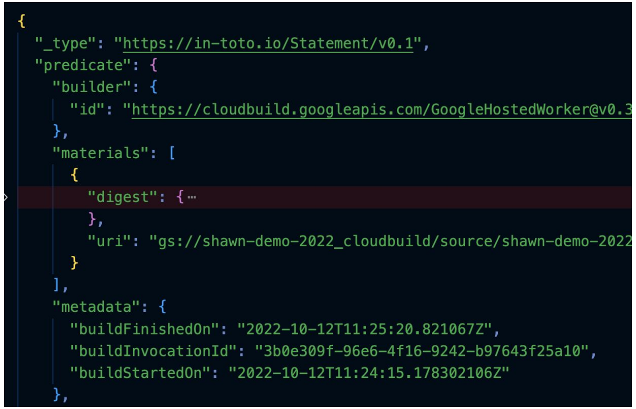
- ✓ Set up job
- ✓ Set up runner
- ✓ Checkout
- ✓ Run google-github-actions/auth@v0
- ✓ Set up Cloud SDK
- ✓ Docker configuration
- ✓ Set up slsa-default-generator
- ✓ Dump GitHub context
- ✓ Build
- ✓ Show my version
- ✓ Install Cosign
- ✓ Check install!
- ✓ Sign Image
- ✓ Post Set up Cloud SDK
- ✓ Post Run google-github-actions/auth@v0
- ✓ Post Checkout
- ✓ Complete runner
- ✓ Complete job

✓ Setup and Deploy 49s

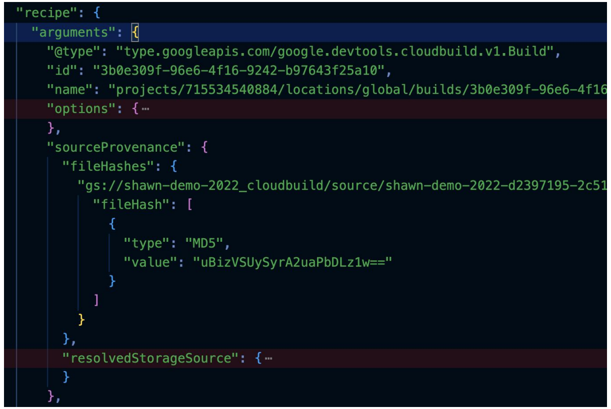
- ✓ Set up job
- ✓ Set up runner
- ✓ Checkout
- ✓ Run google-github-actions/auth@v0
- ✓ Set up Cloud SDK
- ✓ Set up GKE credentials
- ✓ Set up Kubectl and Kustomize
- ✓ Deploy
- ✓ Post Set up Cloud SDK
- ✓ Post Run google-github-actions/auth@v0
- ✓ Post Checkout
- ✓ Complete runner
- ✓ Complete job

CloudBuild: Managed Service for Provenance Provider/Signing

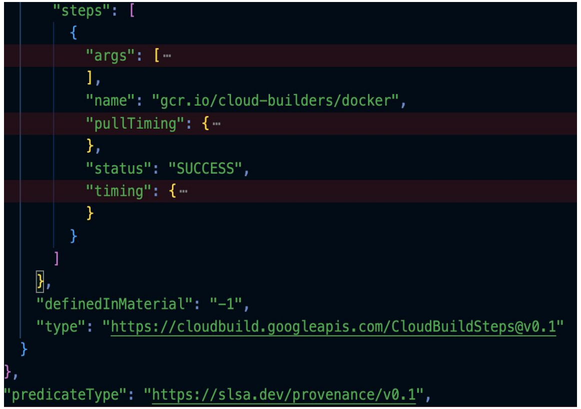
SLSA Security L3 Build Support



```
{
  "_type": "https://in-toto.io/Statement/v0.1",
  "predicate": {
    "builder": {
      "id": "https://cloudbuild.googleapis.com/GoogleHostedWorker@v0.3
    },
    "materials": [
      {
        "digest": {…
        },
        "uri": "gs://shawn-demo-2022_cloudbuild/source/shawn-demo-2022
      }
    ],
    "metadata": {
      "buildFinishedOn": "2022-10-12T11:25:20.821067Z",
      "buildInvocationId": "3b0e309f-96e6-4f16-9242-b97643f25a10",
      "buildStartedOn": "2022-10-12T11:24:15.178302106Z"
    },
```



```
"recipe": {
  "arguments": {
    "@type": "type.googleapis.com/google.devtools.cloudbuild.v1.Build",
    "id": "3b0e309f-96e6-4f16-9242-b97643f25a10",
    "name": "projects/715534540884/locations/global/builds/3b0e309f-96e6-4f16
    "options": {…
    },
    "sourceProvenance": {
      "fileHashes": {
        "gs://shawn-demo-2022_cloudbuild/source/shawn-demo-2022-d2397195-2c51
          "fileHash": [
            {
              "type": "MD5",
              "value": "uBizVSUySyrA2uaPbDLz1w=="
            }
          ]
        }
      },
      "resolvedStorageSource": {…
      }
    },
```



```
      "steps": [
        {
          "args": […
          ],
          "name": "gcr.io/cloud-builders/docker",
          "pullTiming": {…
          },
          "status": "SUCCESS",
          "timing": {…
          }
        }
      ]
    },
    "definedInMaterial": "-1",
    "type": "https://cloudbuild.googleapis.com/CloudBuildSteps@v0.1"
  }
},
"predicateType": "https://slsa.dev/provenance/v0.1",
```

Material: 源碼壓縮檔位置

Metadata: 創建時間

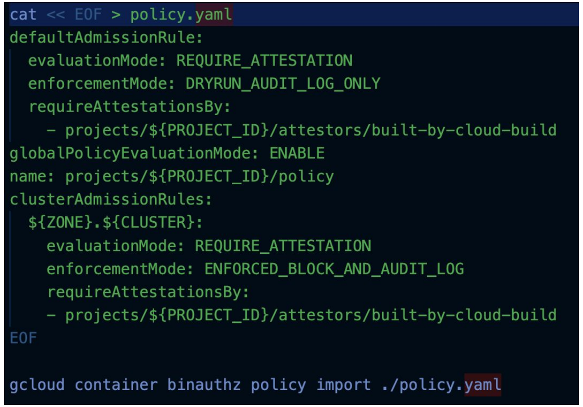
源碼的MD5 Hash

執行步驟

In-Toto概念在GCP的實現:Managed Attester & Verification on GCP

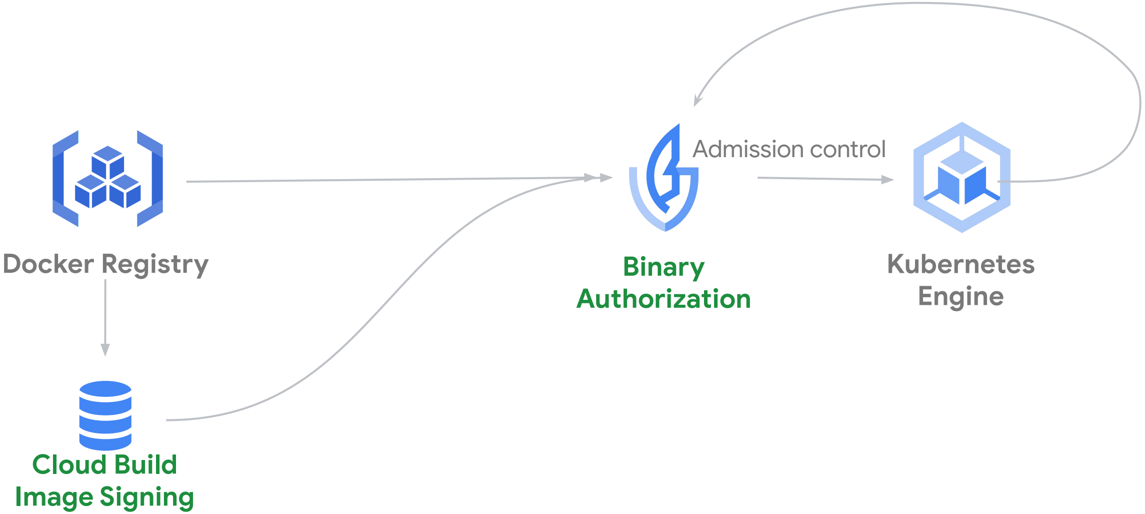
In-Toto + CloudBuild + Binary Authorization

- 安裝Binary Authorization
- 設計Policy:
  - defaultAdmissionRule:
  - clusterAdmissionRule:

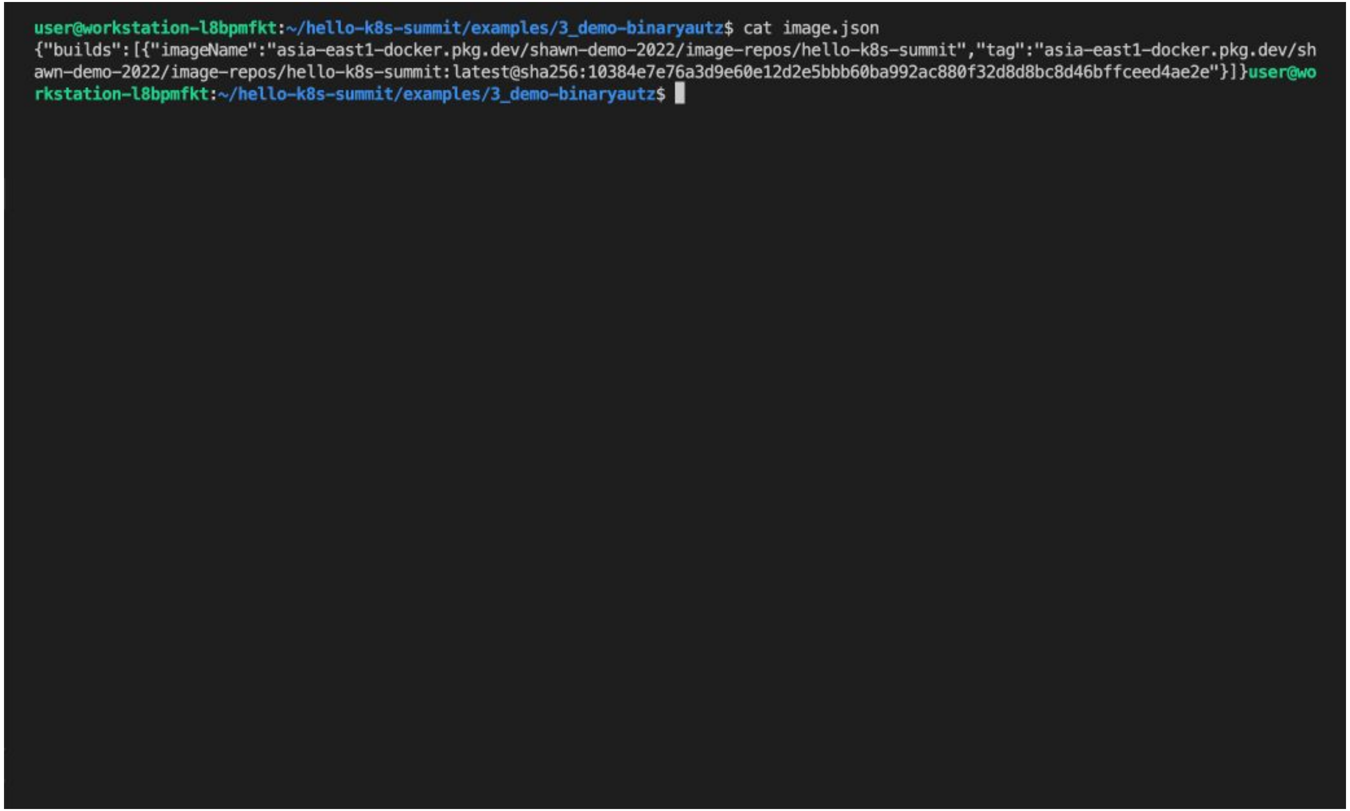


```
cat << EOF > policy.yaml
defaultAdmissionRule:
  evaluationMode: REQUIRE_ATTESTATION
  enforcementMode: DRYRUN_AUDIT_LOG_ONLY
  requireAttestationsBy:
    - projects/${PROJECT_ID}/attestors/built-by-cloud-build
globalPolicyEvaluationMode: ENABLE
name: projects/${PROJECT_ID}/policy
clusterAdmissionRules:
  ${ZONE}.${CLUSTER}:
    evaluationMode: REQUIRE_ATTESTATION
    enforcementMode: ENFORCED_BLOCK_AND_AUDIT_LOG
    requireAttestationsBy:
    - projects/${PROJECT_ID}/attestors/built-by-cloud-build
EOF

gcloud container binauthz policy import ./policy.yaml
```



Demo



```
user@workstation-l8bpmfkt:~/hello-k8s-summit/examples/3_demo-binaryautz$ cat image.json
{"builds":[{"imageName":"asia-east1-docker.pkg.dev/shawn-demo-2022/image-repos/hello-k8s-summit","tag":"asia-east1-docker.pkg.dev/sh
awn-demo-2022/image-repos/hello-k8s-summit:latest@sha256:10384e7e76a3d9e60e12d2e5bbb60ba992ac880f32d8d8bc8d46bffceed4ae2e"}]}user@wo
rkstation-l8bpmfkt:~/hello-k8s-summit/examples/3_demo-binaryautz$
```

準備:

- 設定Binary Authorization Policy
- 使用Skaffold 快速Build出一個帶簽章的版本

展示:

- 使用slsa-verifier驗證Skaffold Build出的鏡像檔案
- 部署已帶有簽章的鏡像檔
- 部署未帶有簽章的映像檔

Agenda

01 Security Challenges from OSS: 開源軟體對資安的挑戰

02 SDLC & SBOM & SLSA: 社群的努力

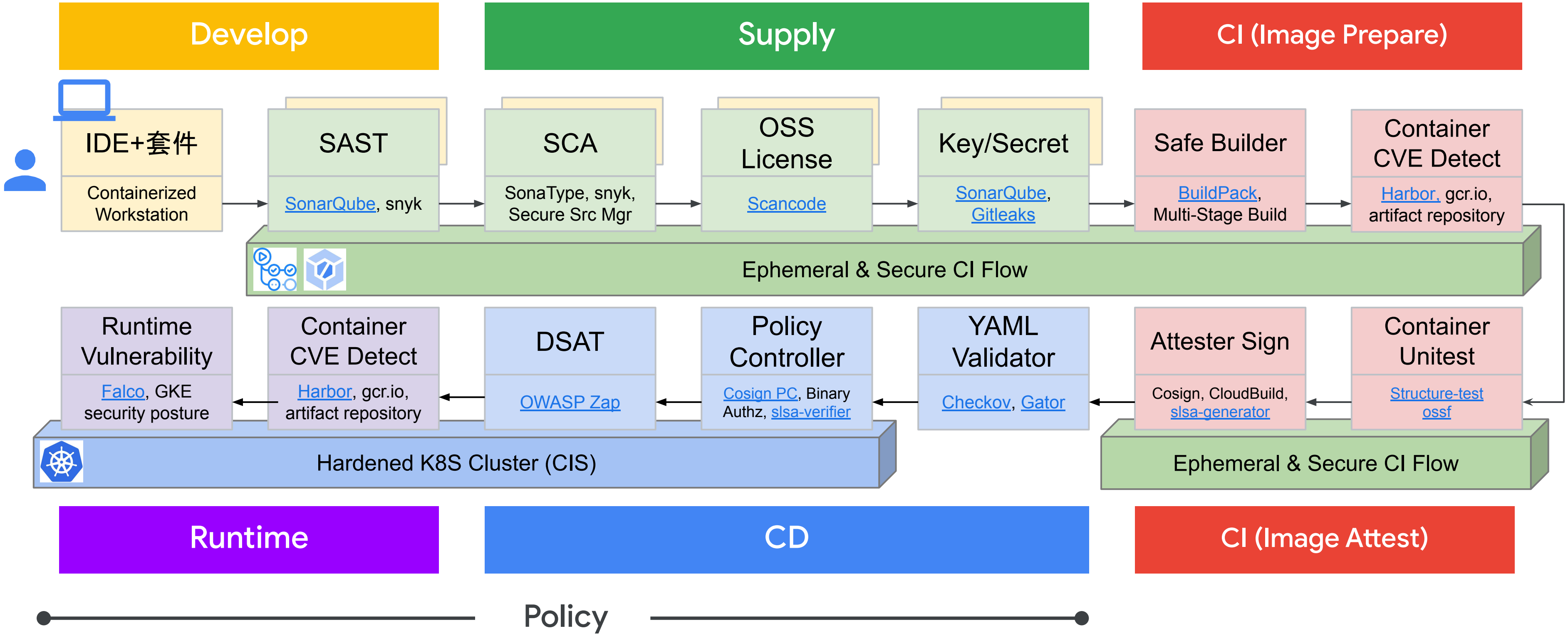
03 Software Delivery Shield: 平台部署+持續防護

04 Software Delivery Shield在GCP上的實現

Google Cloud

End-to-End Software Supply Chain Security Solution

Software Delivery Shield



Develop

Supply

CI (Image Prepare)

IDE+套件
Containerized Workstation

SAST
SonarQube, snyk

SCA
SonaType, snyk, Secure Src Mgr

OSS License
Scancode

Key/Secret
SonarQube, Gitleaks

Safe Builder
BuildPack, Multi-Stage Build

Container CVE Detect
Harbor, gcr.io, artifact repository

Ephemeral & Secure CI Flow

Runtime Vulnerability
Falco, GKE security posture

Container CVE Detect
Harbor, gcr.io, artifact repository

DSAT
OWASP Zap

Policy Controller
Cosign PC, Binary Authz, slsa-verifier

YAML Validator
Checkov, Gator

Attester Sign
Cosign, CloudBuild, slsa-generator

Container Unitest
Structure-test ossf

Hardened K8S Cluster (CIS)

Ephemeral & Secure CI Flow

Runtime

CD

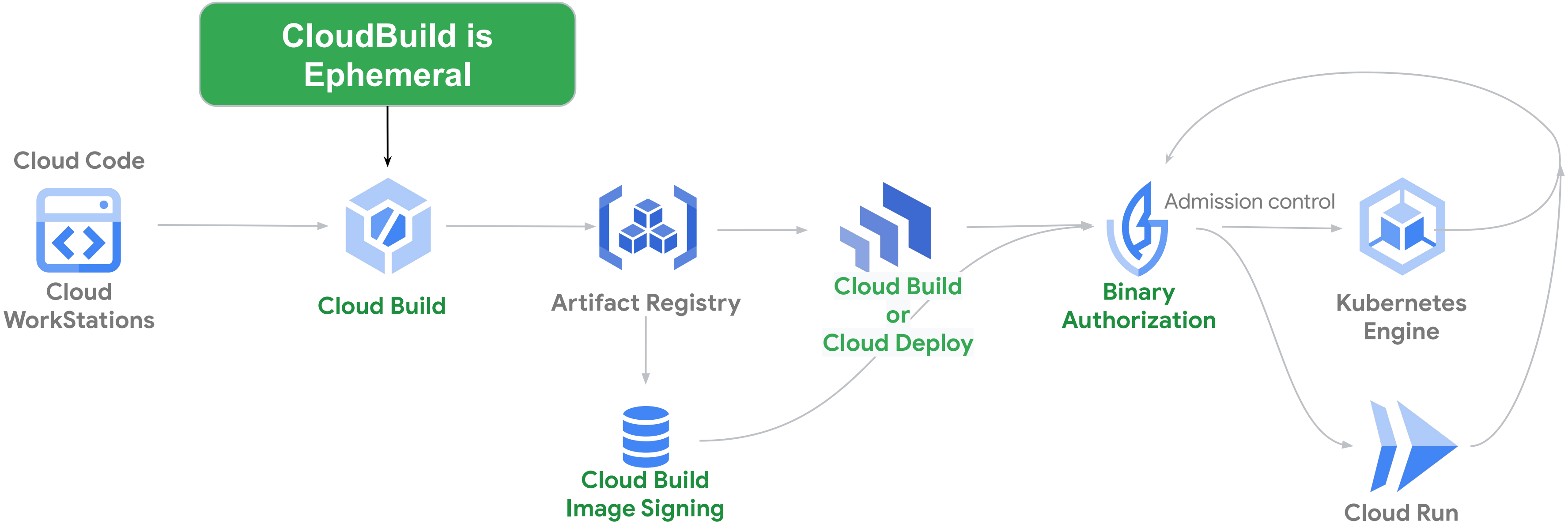
CI (Image Attest)

Policy

Prop

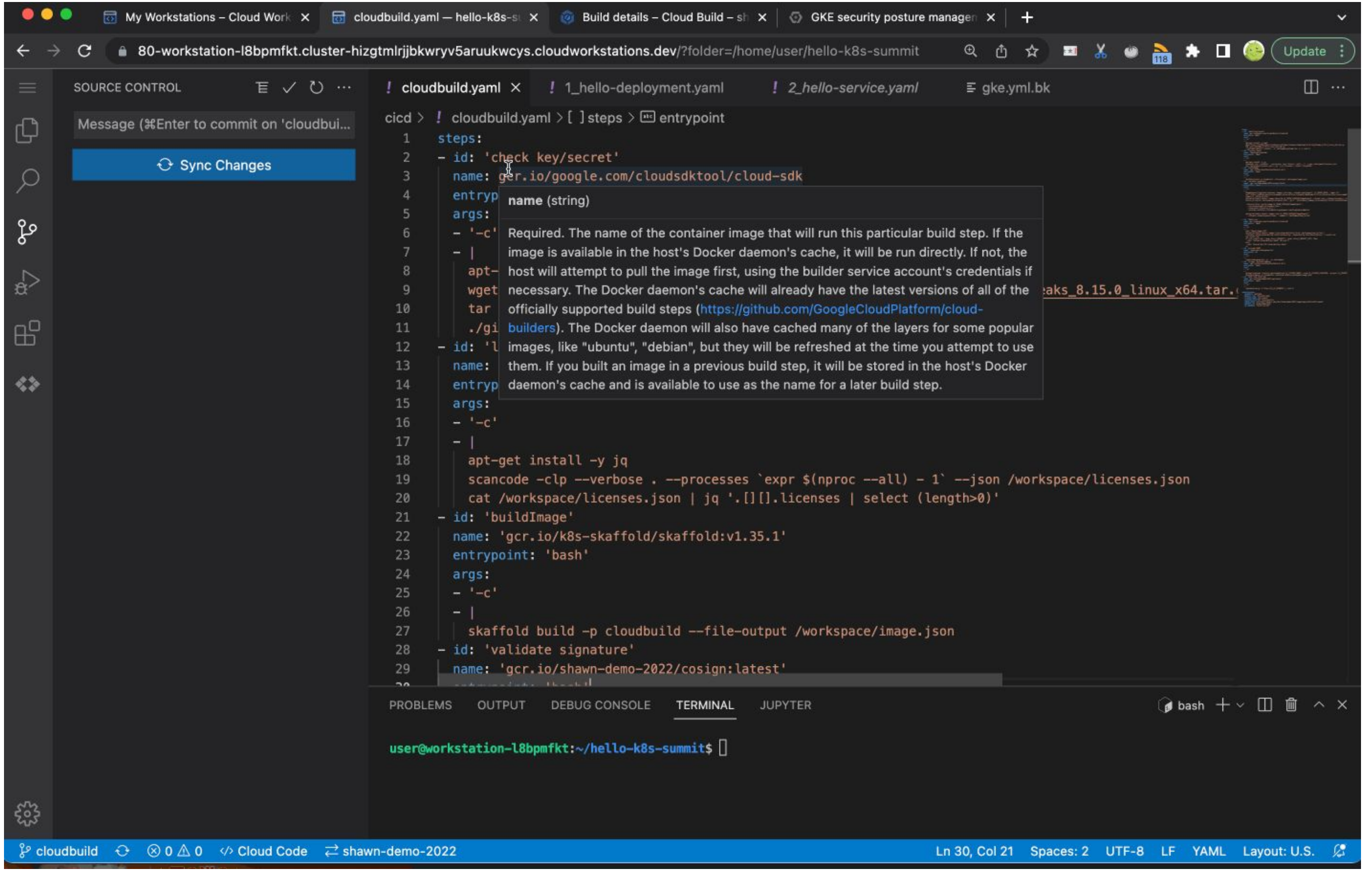
SLSA-Level 3概念在GCP的實現:

Cloud Build + Artifact Registry + Binary Authorization



IDE+套件	OSS License	Key/Secret	Container CVE Detect	Safe Builder	Attester Sign	Policy Controller	DSAT
Containerized Workstation	Scancode	Gitleaks	Artifact Repository	Github Action, CloudBuild	CloudBuild	Binary Authz, slsa-verifier	OWASP Zap

Demo: Secure Delivery Shield 在GCP的實現


My Workstations – Cloud Work
cloudbuild.yaml — hello-k8s-s
Build details – Cloud Build – sh
GKE security posture managem
80-workstation-l8bpmfkt.cluster-hizgtmlrjjbkwryv5aruukwcys.cloudworkstations.dev/?folder=/home/user/hello-k8s-summit
Update
SOURCE CONTROL
Message (⌘Enter to commit on 'cloudbui...
Sync Changes
cloudbuild.yaml
1_hello-deployment.yaml
2_hello-service.yaml
gke.yml.bk
cicd > cloudbuild.yaml > [ ] steps > entrypoint
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
bash
cloudbuild
⊗ 0 ⚠ 0
Cloud Code
shawn-demo-2022
Ln 30, Col 21
Spaces: 2
UTF-8
LF
YAML
Layout: U.S.

```yaml
steps:
- id: 'check key/secret'
  name: gcr.io/google.com/cloudsdktool/cloud-sdk
  entryp
  args:
  - '-c'
  - |
    apt-
    wget                                                      aks_8.15.0_linux_x64.tar.
    tar
    ./gi
- id: 'l
  name:
  entryp
  args:
  - '-c'
  - |
    apt-get install -y jq
    scancode -clp --verbose . --processes `expr $(nproc --all) - 1` --json /workspace/licenses.json
    cat /workspace/licenses.json | jq '.[][].licenses | select (length>0)'
- id: 'buildImage'
  name: 'gcr.io/k8s-skaffold/skaffold:v1.35.1'
  entrypoint: 'bash'
  args:
  - '-c'
  - |
    skaffold build -p cloudbuild --file-output /workspace/image.json
- id: 'validate signature'
  name: 'gcr.io/shawn-demo-2022/cosign:latest'
```

name (string)

Required. The name of the container image that will run this particular build step. If the image is available in the host's Docker daemon's cache, it will be run directly. If not, the host will attempt to pull the image first, using the builder service account's credentials if necessary. The Docker daemon's cache will already have the latest versions of all of the officially supported build steps (https://github.com/GoogleCloudPlatform/cloud-builders). The Docker daemon will also have cached many of the layers for some popular images, like "ubuntu", "debian", but they will be refreshed at the time you attempt to use them. If you built an image in a previous build step, it will be stored in the host's Docker daemon's cache and is available to use as the name for a later build step.

```
user@workstation-l8bpmfkt:~/hello-k8s-summit$ 
```

1. Key/Secret Check
2. OSS License Validation
3. Build Image with CloudBuild
4. Verify Signature with slsa-verifier
5. Verify Image CVEs with Container Analysis API
6. Provide SBOM along with image
7. Deploy Image with Skaffold+Kustomize
8. DSAT Test with ZAP

Road to SLSA Security Level 4

See: slsa.dev

Measure integrity levels for build, source and dependencies

Automation & Provenance

Build must be fully scripted/automated and generate provenance

Version Control & Signed Provenance

Requires using version control and hosted build service that generates authenticated provenance

Non-falsifiable, Ephemeral

Builds are fully trustworthy, with identity attestations of underlying build infrastructure/hardware. Ephemeral builds leave nothing behind.

Hermetic Builds, Review

All build inputs/dependencies are specified upfront with no internet egress during the build. **Two-party reviews.**

Learn more

嘗試 Software Delivery Shield

Check out our Quickstart tutorials to get started with Software Delivery Shield:
cloud.google.com/software-supply-chain-security/docs/sds/overview
bit.ly/SoftwareDeliveryShield

了解更多開發供應鏈上資安強化

To learn more about software supply chain security, visit:
cloud.google.com/software-supply-chain-security

關注以下的社群

- SBOM - The Linux Foundation
- Sigstore - The Linux Foundation
- ScoreCard- Open Security Source Foundation (OSSF)
- SLSA - An Industry Collaboration
- Deps.dev - Open Source Insight Team, powered by Google

Thank you

cloud.google.com/software-supply-chain-security

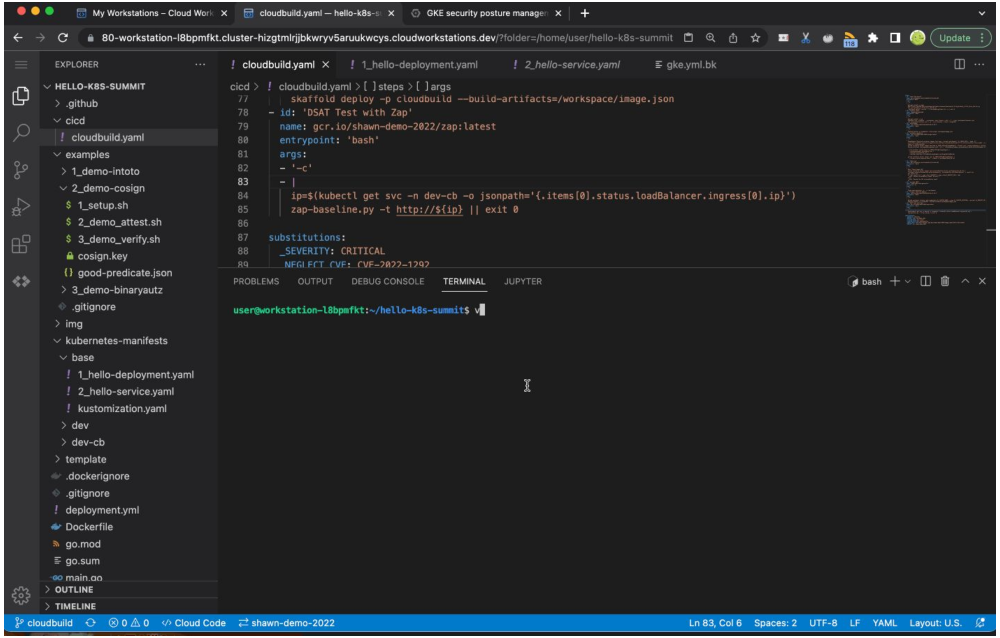
Develop

Software Delivery Shield

Fully managed development environments

Cloud Workstations

- On-demand environments accessible anywhere
- Security policies
- Managed base images
- VPC and VPC-Service controls



```yaml
      skaffold deploy -p cloudbuild --build-artifacts=/workspace/image.json
- id: 'DSAT Test with Zap'
  name: gcr.io/shawn-demo-2022/zap:latest
  entrypoint: 'bash'
  args:
  - '-c'
  - |
    ip=$(kubectl get svc -n dev-cb -o jsonpath='{.items[0].status.loadBalancer.ingress[0].ip}')
    zap-baseline.py -t http://${ip} || exit 0

substitutions:
  _SEVERITY: CRITICAL
   NEGLECT_CVE: CVE-2022-1292
```

```
user@workstation-l8bpmfkt:~/hello-k8s-summit$ v
```

Preview

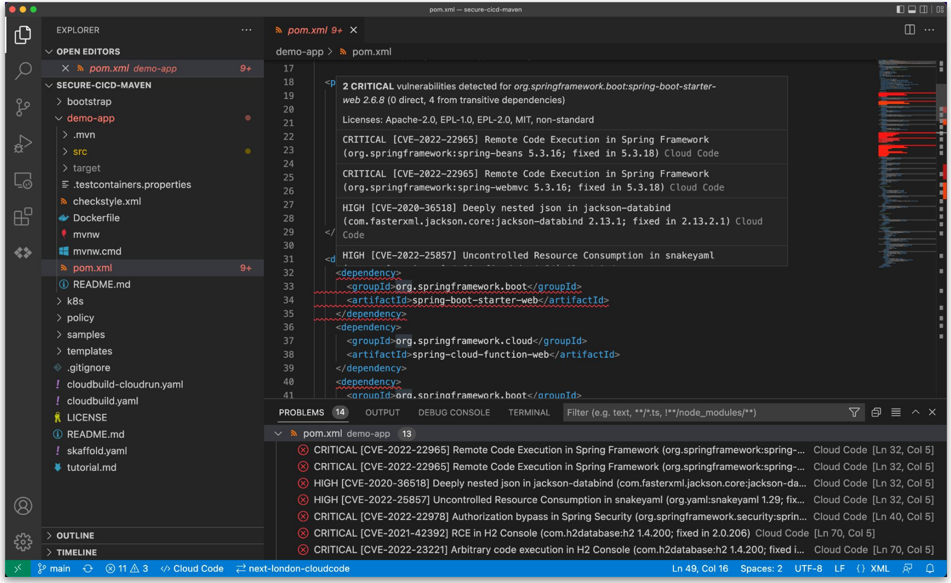
Develop

Software Delivery Shield

Security assistance in the IDE

Cloud Code source protect

- Vulnerability detection as you code
- Support for scanning transitive dependencies
- Dependency license reporting



2 CRITICAL vulnerabilities detected for *org.springframework.boot:spring-boot-starter-web 2.6.8* (0 direct, 4 from transitive dependencies)

Licenses: Apache-2.0, EPL-1.0, EPL-2.0, MIT, non-standard

```
CRITICAL [CVE-2022-22965] Remote Code Execution in Spring Framework (org.springframework:spring-beans 5.3.16; fixed in 5.3.18) Cloud Code
CRITICAL [CVE-2022-22965] Remote Code Execution in Spring Framework (org.springframework:spring-webmvc 5.3.16; fixed in 5.3.18) Cloud Code
HIGH [CVE-2020-36518] Deeply nested json in jackson-databind (com.fasterxml.jackson.core:jackson-databind 2.13.1; fixed in 2.13.2.1) Cloud Code
HIGH [CVE-2022-25857] Uncontrolled Resource Consumption in snakeyaml
```

```xml
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-function-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
```

```
CRITICAL [CVE-2022-22965] Remote Code Execution in Spring Framework (org.springframework:spring-...   Cloud Code [Ln 32, Col 5]
CRITICAL [CVE-2022-22965] Remote Code Execution in Spring Framework (org.springframework:spring-...   Cloud Code [Ln 32, Col 5]
HIGH [CVE-2020-36518] Deeply nested json in jackson-databind (com.fasterxml.jackson.core:jackson-da...   Cloud Code [Ln 32, Col 5]
HIGH [CVE-2022-25857] Uncontrolled Resource Consumption in snakeyaml (org.yaml:snakeyaml 1.29; fix...   Cloud Code [Ln 32, Col 5]
CRITICAL [CVE-2022-22978] Authorization bypass in Spring Security (org.springframework.security:sprin...   Cloud Code [Ln 40, Col 5]
CRITICAL [CVE-2021-42392] RCE in H2 Console (com.h2database:h2 1.4.200; fixed in 2.0.206) Cloud Code [Ln 70, Col 5]
CRITICAL [CVE-2022-23221] Arbitrary code execution in H2 Console (com.h2database:h2 1.4.200; fixed i...   Cloud Code [Ln 70, Col 5]
```

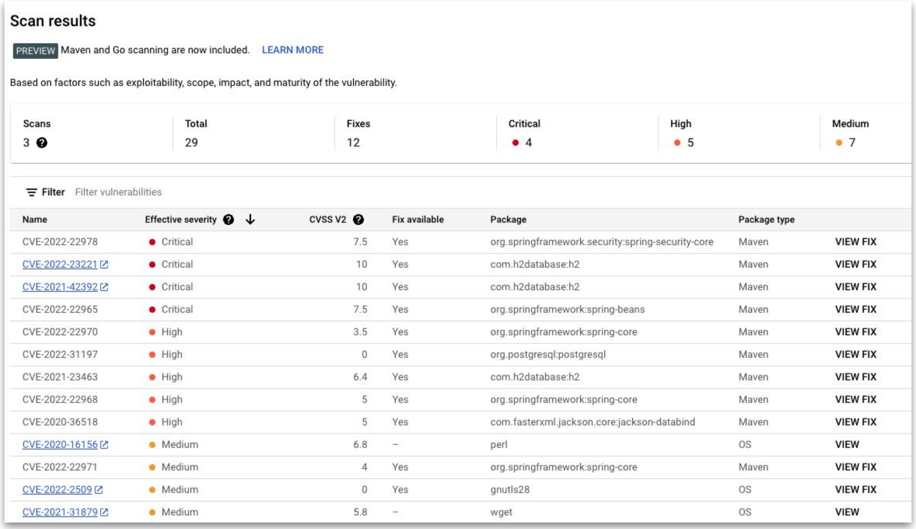
Preview

Improving security of artifacts and dependencies

Artifact Registry & Container Analysis

Assured Open Source Software

- Artifact Registry - Maven virtual and remote repos
- Container Analysis - On-push Maven and Go container scanning and standalone Maven package scanning
- Container Analysis - On-push SBOM dependency list generation for containers
- Assured Open Source Software - 250+ Java and Python packages



Scan results

PREVIEW Maven and Go scanning are now included. LEARN MORE

Based on factors such as exploitability, scope, impact, and maturity of the vulnerability.

Scans	Total	Fixes	Critical	High	Medium
3 ?	29	12	● 4	● 5	● 7

Name	Effective severity ? ↓	CVSS V2 ?	Fix available	Package	Package type	
CVE-2022-22978	● Critical	7.5	Yes	org.springframework.security:spring-security-core	Maven	VIEW FIX
CVE-2022-23221	● Critical	10	Yes	com.h2database:h2	Maven	VIEW FIX
CVE-2021-42392	● Critical	10	Yes	com.h2database:h2	Maven	VIEW FIX
CVE-2022-22965	● Critical	7.5	Yes	org.springframework:spring-beans	Maven	VIEW FIX
CVE-2022-22970	● High	3.5	Yes	org.springframework:spring-core	Maven	VIEW FIX
CVE-2022-31197	● High	0	Yes	org.postgresql:postgresql	Maven	VIEW FIX
CVE-2021-23463	● High	6.4	Yes	com.h2database:h2	Maven	VIEW FIX
CVE-2022-22968	● High	5	Yes	org.springframework:spring-core	Maven	VIEW FIX
CVE-2020-36518	● High	5	Yes	com.fasterxml.jackson.core:jackson-databind	Maven	VIEW FIX
CVE-2020-16156	● Medium	6.8	–	perl	OS	VIEW
CVE-2022-22971	● Medium	4	Yes	org.springframework:spring-core	Maven	VIEW FIX
CVE-2022-2509	● Medium	0	Yes	gnutls28	OS	VIEW FIX
CVE-2021-31879	● Medium	5.8	–	wget	OS	VIEW

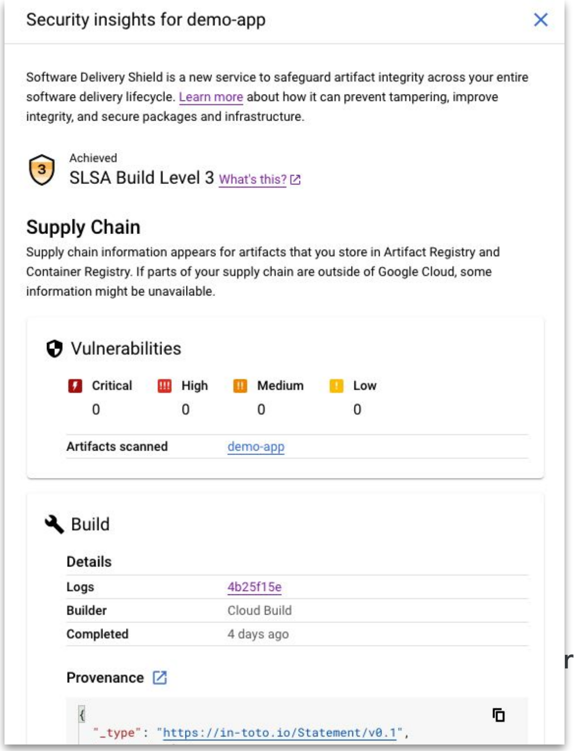
Preview

Software Delivery Shield

Enhance the security of your CI pipelines

Cloud Build

- SLSA Level 3 build support (slsa.dev)
- Build provenance for non-container Java (Maven) and Python packages
- Security insights panel



Security insights for demo-app

Software Delivery Shield is a new service to safeguard artifact integrity across your entire software delivery lifecycle. Learn more about how it can prevent tampering, improve integrity, and secure packages and infrastructure.

Achieved
SLSA Build Level 3 What's this?

Supply Chain

Supply chain information appears for artifacts that you store in Artifact Registry and Container Registry. If parts of your supply chain are outside of Google Cloud, some information might be unavailable.

Vulnerabilities

Critical	High	Medium	Low
0	0	0	0

Artifacts scanned	demo-app

Build

Details	
Logs	4b25f15e
Builder	Cloud Build
Completed	4 days ago

Provenance

```
{
  "_type": "https://in-toto.io/Statement/v0.1",
```

review

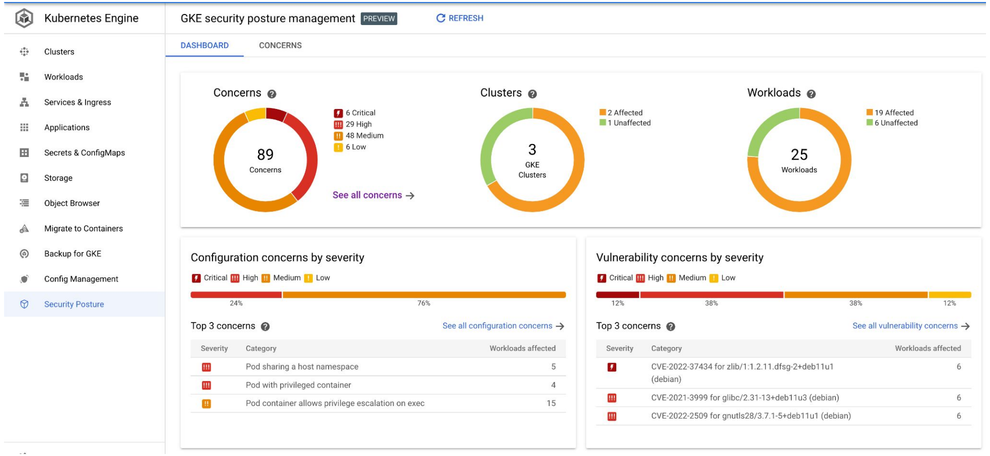
Software Delivery Shield

Security insights at the runtime

GKE security posture

Cloud Run security insights

- GKE continuous runtime vulnerability and workload configuration scanning
- Cloud Run insights into security target levels, service vulnerabilities, and build provenance



Kubernetes Engine

- Clusters
- Workloads
- Services & Ingress
- Applications
- Secrets & ConfigMaps
- Storage
- Object Browser
- Migrate to Containers
- Backup for GKE
- Config Management
- Security Posture

GKE security posture management PREVIEW

REFRESH

DASHBOARD CONCERNS

Concerns

89 Concerns

6 Critical
29 High
48 Medium
6 Low

See all concerns →

Clusters

3 GKE Clusters

2 Affected
1 Unaffected

Workloads

25 Workloads

19 Affected
6 Unaffected

Configuration concerns by severity

Critical High Medium Low

24% 76%

Top 3 concerns

See all configuration concerns →

Severity	Category	Workloads affected
!!!	Pod sharing a host namespace	5
!!!	Pod with privileged container	4
!!	Pod container allows privilege escalation on exec	15

Vulnerability concerns by severity

Critical High Medium Low

12% 38% 38% 12%

Top 3 concerns

See all vulnerability concerns →

Severity	Category	Workloads affected
⚡	CVE-2022-37434 for zlib/1:1.2.11.dfsg-2+deb11u1 (debian)	6
!!!	CVE-2021-3999 for glibc/2.31-13+deb11u3 (debian)	6
!!!	CVE-2022-2509 for gnutls28/3.7.1-5+deb11u1 (debian)	6

Preview

Software Delivery Shield

Cosign to prevent unauthorized images

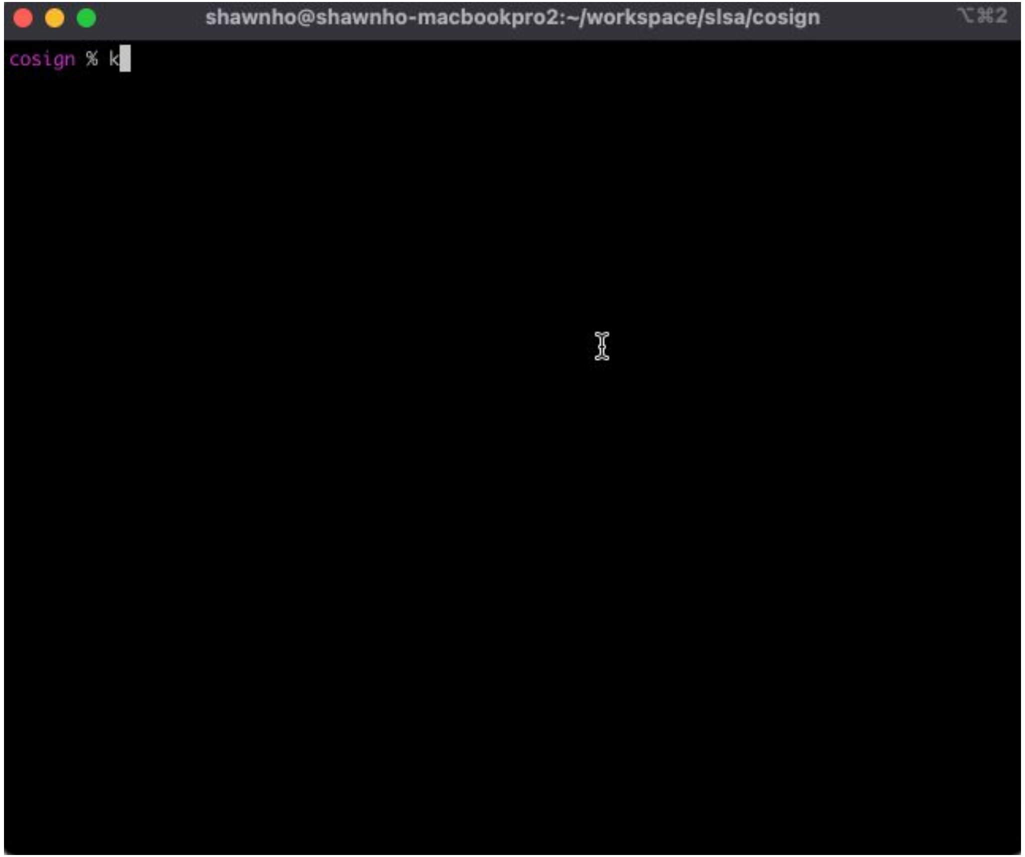
In Kubernetes:

- Install by helm chart

- Use ClusterImagePolicy to replace policy parameter in cosign

- Need to tag namespace with policy.sigstore.dev/include=true to enable admission webhook

- Attestation can be revoked which prevents the image to run on the cluster again.



```
cosign % k
```