# Cloud Native Security

## From Development to Runtime

Martin Georgiev
Senior Security Engineer

Anton Sankov
Senior Software Engineer

vmware® Confidential | © VMware, Inc.

# What is Cloud Native?

# What is Cloud Native?

## A modern way to build and run apps

A modern way to **build**, **run** and **operate** scalable applications in dynamic environments such as public, private, and hybrid clouds.

# Cloud Native patterns

Containerised microservices applications

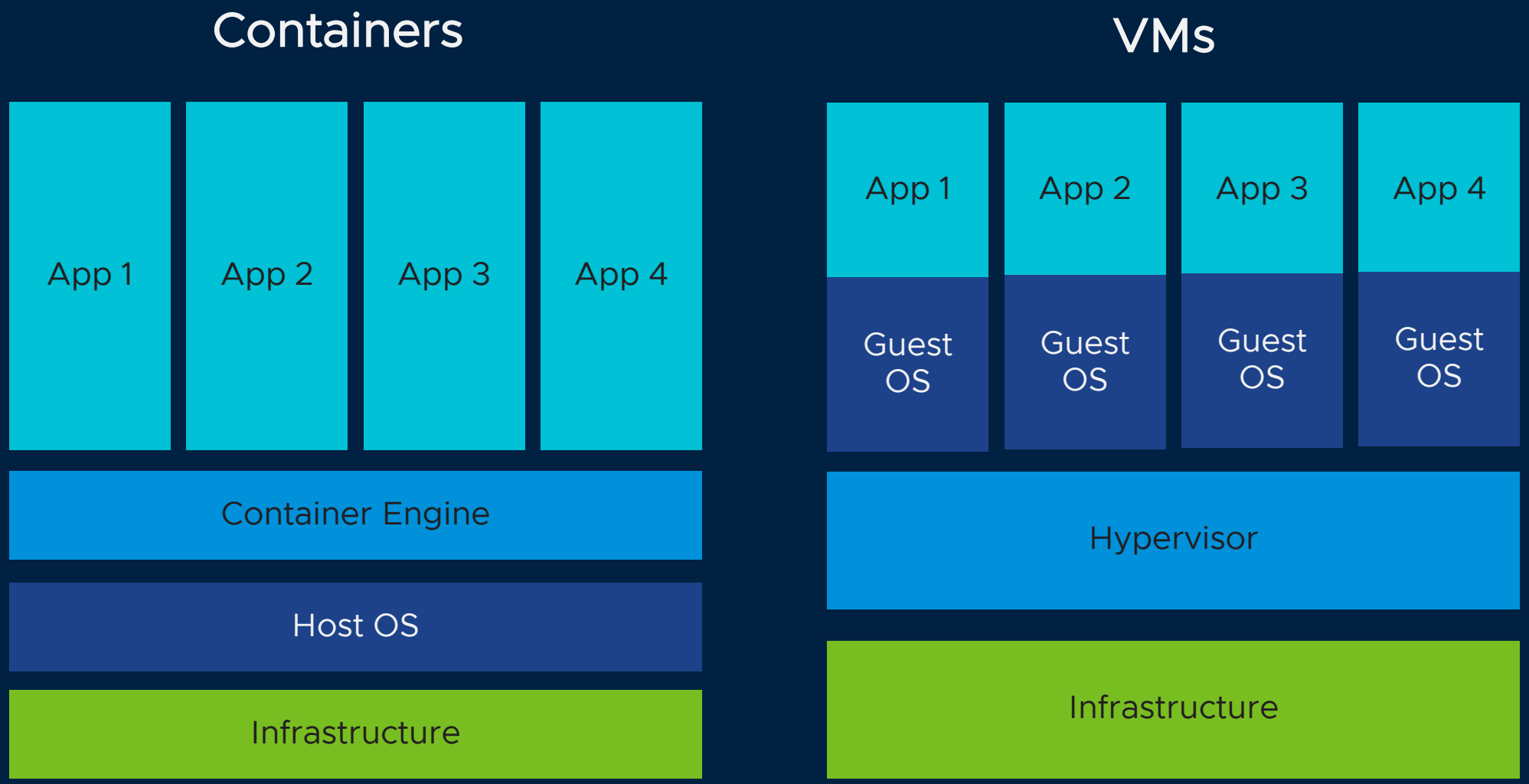
Dev(Sec)Ops teams

GitOps workflow
Declarative infrastructure (as Code)

# Containers

A **lightweight**, **standalone**, **executable** package of software that includes everything needed to **run** an application.

# Containers vs. VMs



# Dev(Sec)Ops

A methodology where **development** and **operations** (and **security**) teams are not different teams, but one team working together.

# Infrastructure as Code / GitOps

## Infrastructure as Code

The process of managing infrastructure by defining it in text files, instead of hardware configuration or manual interaction.

The text files are applied to the infrastructure as code provider that takes care of matching the actual state of the infrastructure with the desired state.

## GitOps

The process of using Git as the source of truth for our operations.

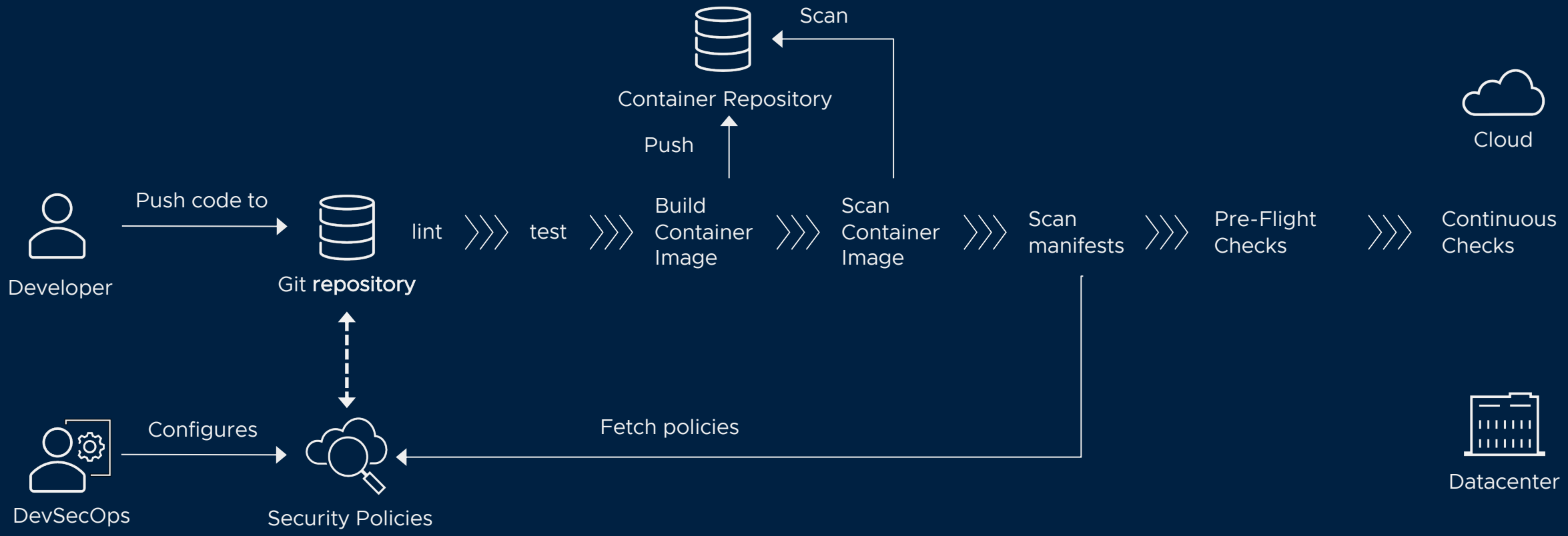
Every change is a Pull Request, every merge runs a CI pipeline that runs a set of actions.

Git is the main platform of collaboration between DevSecOps teams.

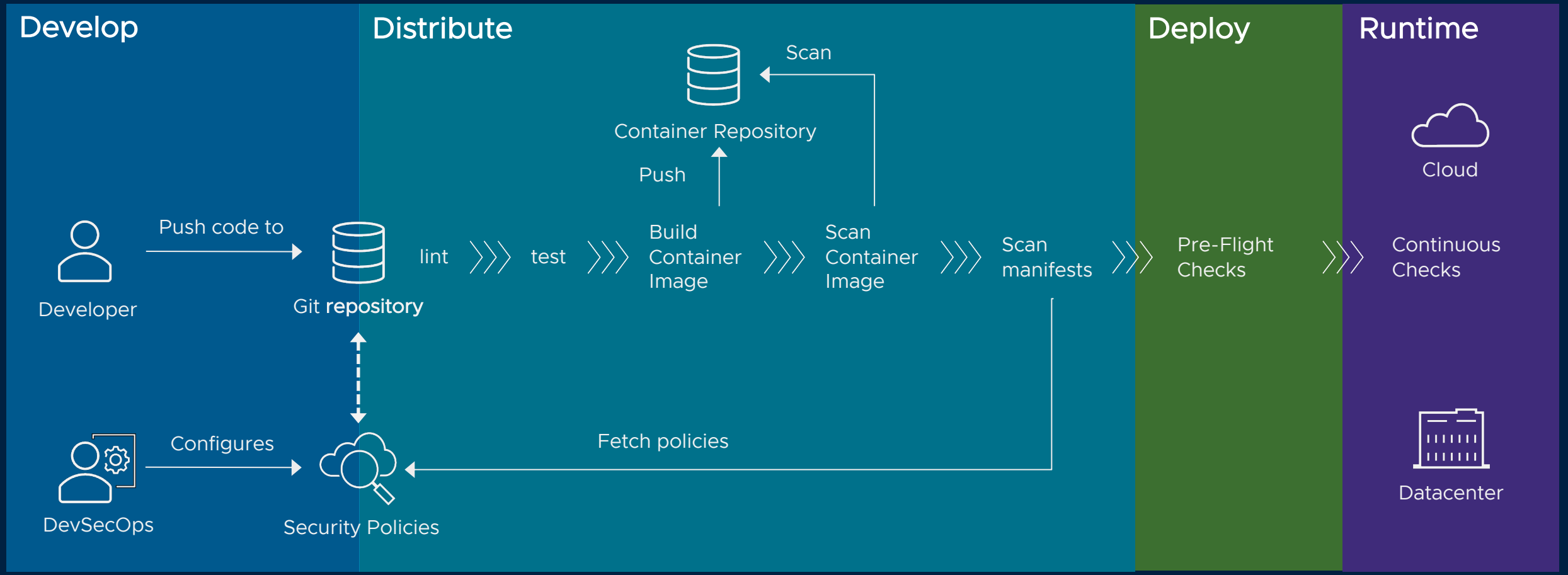
# Security Challenges

- Securing Numerous Services
- Service Communication and Discovery
- Identity and Secret Management
- Complex Infrastructure Management
- State Drift

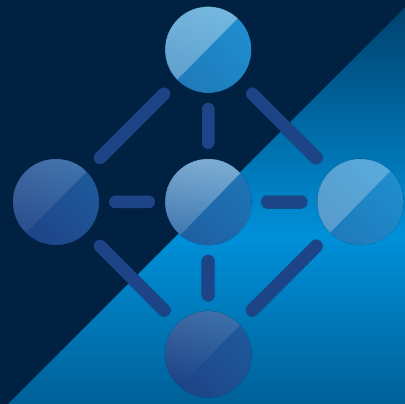
# Cloud Native Continuous Lifecycle



# Cloud Native Continuous Lifecycle



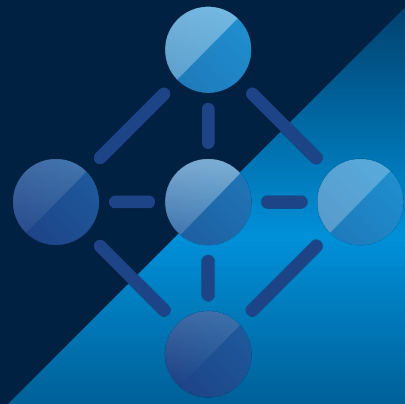
# Develop



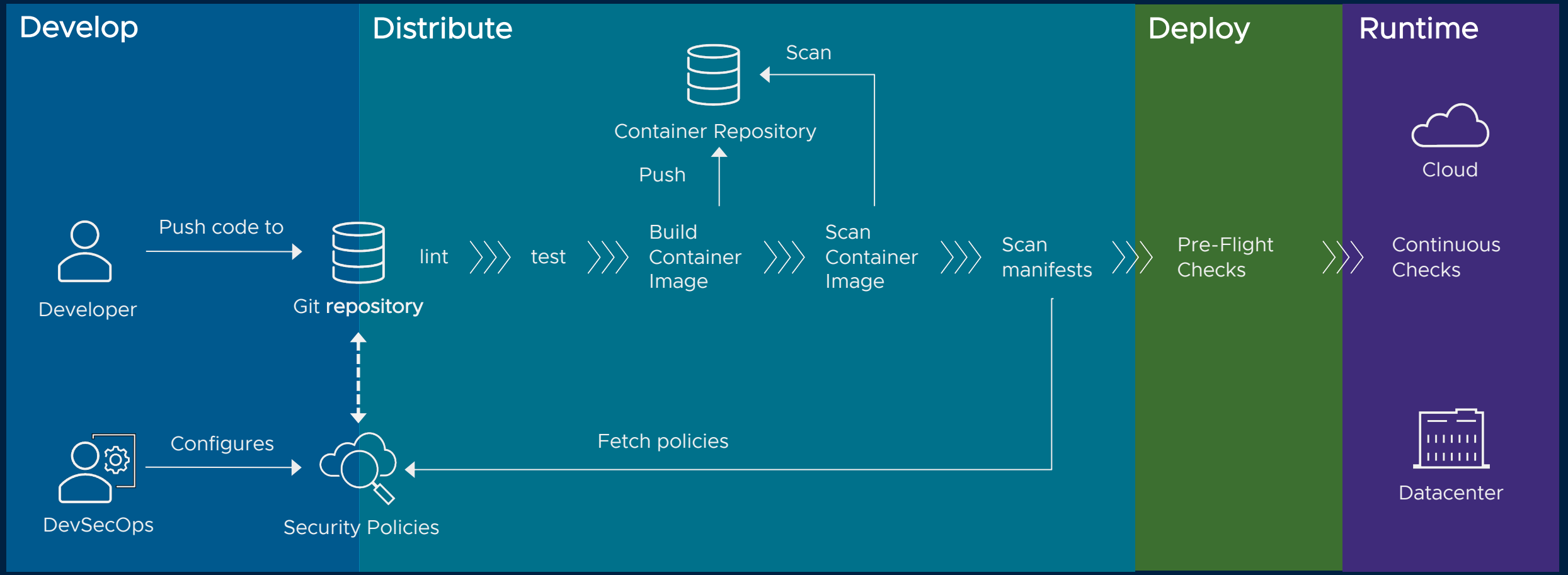
# Develop

- Security Requirements
- Threat Modelling
- Code Reviews

# Distribute



# Cloud Native Continuous Lifecycle



# Distribute: Container Image Scanning

# Distribute. App Evolution.

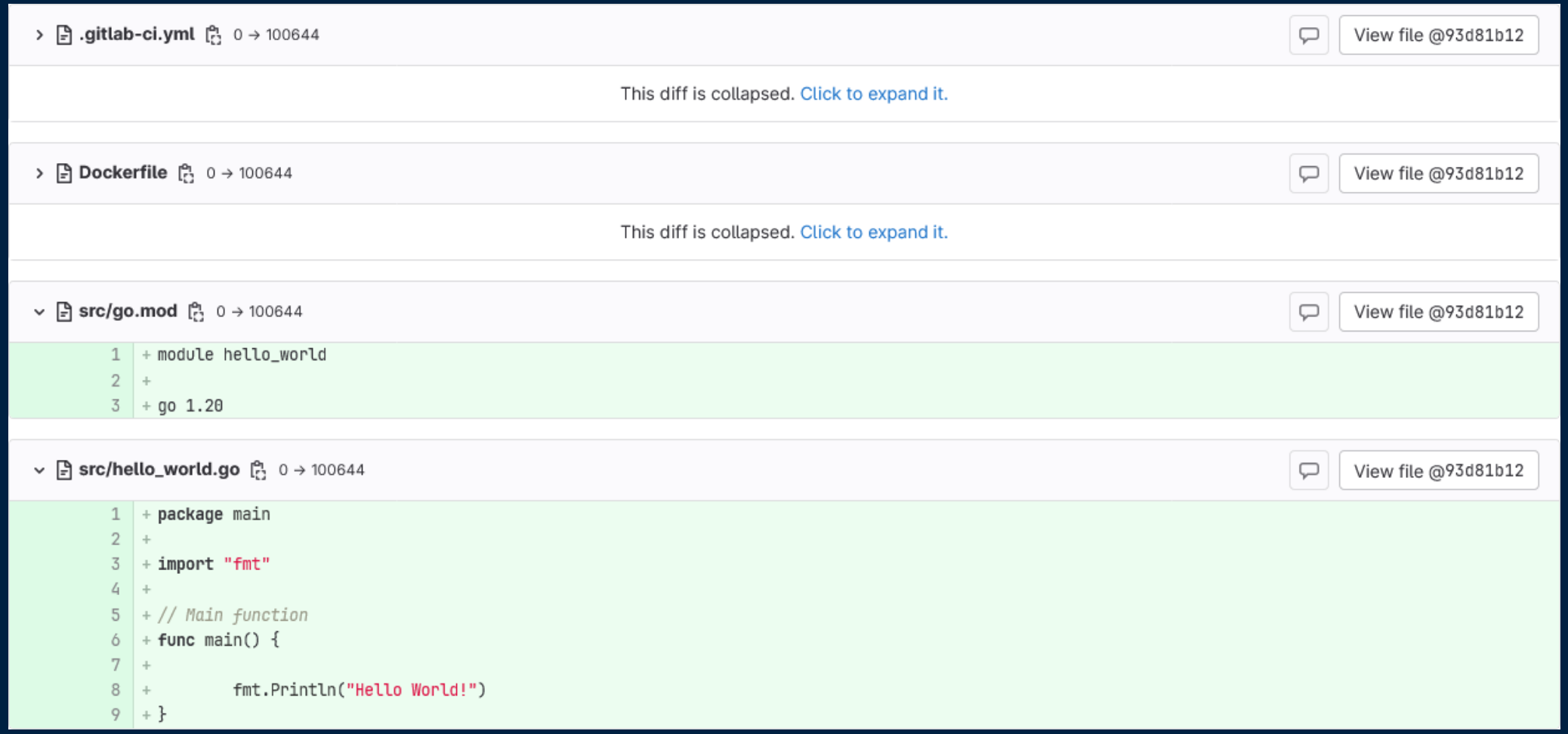
## CI/CD

Demo:

1. App CI/CD (lint, test, build, publish)
2. Add scanning stage
    1. Show scanner (syft + grype)
    2. App CI/CD (lint, test, build, scan, publish)
3. App evolution
    1. A new functionality via new dependency
    2. Scan fails
    3. Fix all issues
    4. Successful build

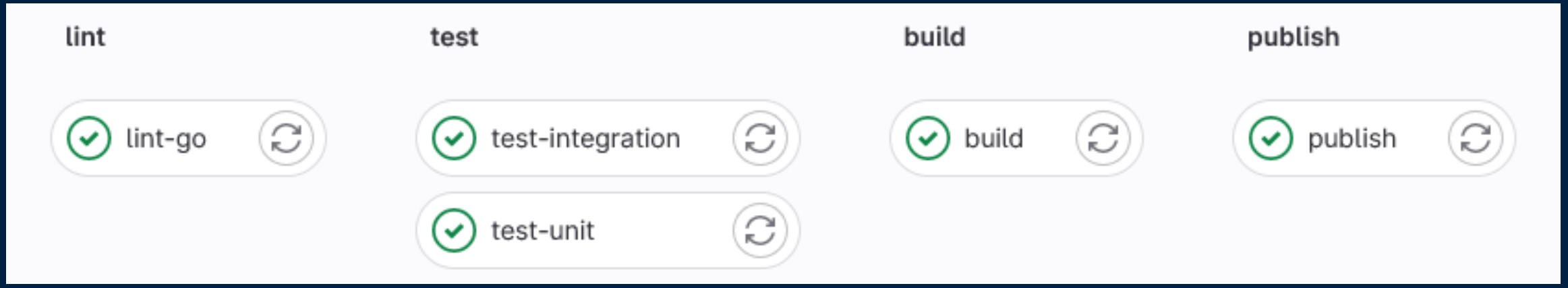
# Distribute. App Evolution.

## Simple App



# Distribute. App Evolution.

## Simple Build Pipeline



# Distribute. App Evolution.

## Prepare scanner

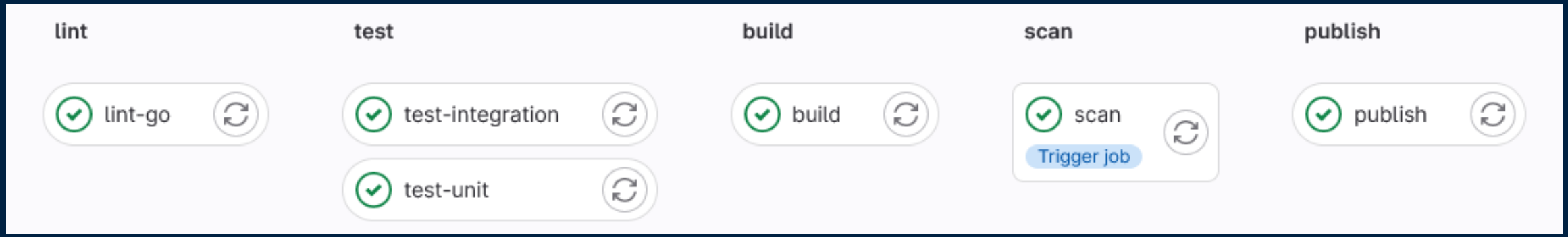
```
# Pull the container image. Make use of caching.
podman pull "${SCAN_IMAGE}"
podman image save "${SCAN_IMAGE}" > image.tar

# Generate Container Software Bill of Materials (SBOM).
syft packages -o json --file "${FILE_SBOM}" docker-archive:image.tar

# Scan the SBOM for vulnerabilities.
grype -o json --file "${FILE_SCAN}" --fail-on "${SEVERITY_FAIL}" sbom:"${FILE_SBOM}"
```

# Distribute. App Evolution.

## Scan stage



# Distribute. App Evolution.

## Add new functionality

**src/go.mod** — View file @07e2ddc0

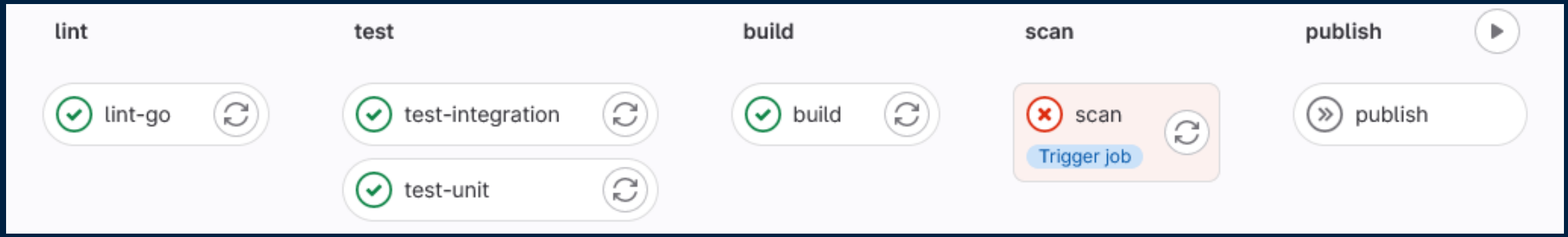
```
1  1   module hello_world
2  2
3  3   go 1.20
   4  +
   5  + require github.com/hashicorp/go-getter v1.5.0
   6  +
```

**src/hello_world.go** — View file @07e2ddc0

```
1  1   package main
2  2
3  3   import "fmt"
   4  + import "github.com/hashicorp/go-getter"
4  5
5  6   // Main function
6  7   func main() {
7  8
8  9           fmt.Println("Hello World!")
  10  +         fmt.Println("Available Decompressors: ", getter.Decompressors)
9  11  }
```

# Distribute. App Evolution.

## Pipeline Status



# Distribute. App Evolution.

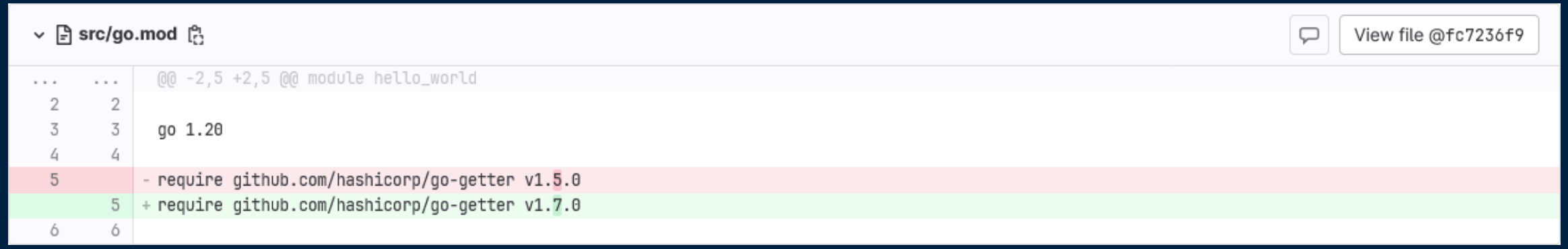
## Scan Status

| ID | Severity | Package Name | Package Version | Fixed In |
|---|---|---|---|---|
| CVE-2022-26945 | Critical | github.com/hashicorp/go-getter | v1.5.0 | |
| CVE-2022-29810 | Medium | github.com/hashicorp/go-getter | v1.5.0 | |
| CVE-2022-30321 | High | github.com/hashicorp/go-getter | v1.5.0 | |
| CVE-2022-30322 | High | github.com/hashicorp/go-getter | v1.5.0 | |
| CVE-2022-30323 | High | github.com/hashicorp/go-getter | v1.5.0 | |
| CVE-2023-0475 | Medium | github.com/hashicorp/go-getter | v1.5.0 | |
| GHSA-27rq-4943-qcwp | Medium | github.com/hashicorp/go-getter | v1.5.0 | 1.5.11 |
| GHSA-28r2-q6m8-9hpx | High | github.com/hashicorp/go-getter | v1.5.0 | 1.6.1 |
| GHSA-39qc-96h7-956f | High | golang.org/x/net | v0.0.0-20190620200207-3b0461eec859 | 0.0.0-20190813141303-74dc4d7220e7 |
| GHSA-69cg-p879-7622 | High | golang.org/x/net | v0.0.0-20190620200207-3b0461eec859 | 0.0.0-20220906165146-f3363e06e74c |
| GHSA-69ch-w2m2-3vjp | High | golang.org/x/text | v0.3.2 | 0.3.8 |
| GHSA-76wf-9vgp-pj7w | Medium | github.com/aws/aws-sdk-go | v1.15.78 | 1.34.0 |
| GHSA-7f33-f4f5-xwgw | Low | github.com/aws/aws-sdk-go | v1.15.78 | 1.34.0 |
| GHSA-83g2-8m93-v3w7 | High | golang.org/x/net | v0.0.0-20190620200207-3b0461eec859 | 0.0.0-20210520170846-37e1c6afe023 |
| GHSA-cjr4-fv6c-f3mv | High | github.com/hashicorp/go-getter | v1.5.0 | 1.6.1 |
| GHSA-f5pg-7wfw-84q9 | Medium | github.com/aws/aws-sdk-go | v1.15.78 | 1.34.0 |
| GHSA-fcgg-rvwg-jv58 | High | github.com/hashicorp/go-getter | v1.5.0 | 1.6.1 |
| GHSA-h86h-8ppg-mxmh | Medium | golang.org/x/net | v0.0.0-20190620200207-3b0461eec859 | 0.0.0-20210428140749-89ef3d95e781 |
| GHSA-hgr8-6h9x-f7q9 | High | golang.org/x/net | v0.0.0-20190620200207-3b0461eec859 | 0.0.0-20190813141303-74dc4d7220e7 |
| GHSA-jpxj-2jvg-6jv9 | Medium | github.com/hashicorp/go-getter | v1.5.0 | 1.7.0 |
| GHSA-p782-xgp4-8hr8 | Medium | golang.org/x/sys | v0.0.0-20190624142023-c5567b49c5d0 | 0.0.0-20220412211240-33da011f77ad |
| GHSA-ppp9-7jff-5vj2 | High | golang.org/x/text | v0.3.2 | 0.3.7 |
| GHSA-vvpx-j8f3-3w6h | High | golang.org/x/net | v0.0.0-20190620200207-3b0461eec859 | 0.7.0 |
| GHSA-x24g-9w7v-vprh | Critical | github.com/hashicorp/go-getter | v1.5.0 | 1.6.1 |

```
+ jq -s '.[0] * .[1]' grype_b8fa427d-eb8b-429d-b69f-fdc2d960c972.json syft_b8fa427d-eb8b-429d-b69f-fdc2d960c972.json
+ exit 1
Running after_script
Running after script...
$ rm image.tar
Cleaning up project directory and file based variables
ERROR: Job failed: exit status 1
```

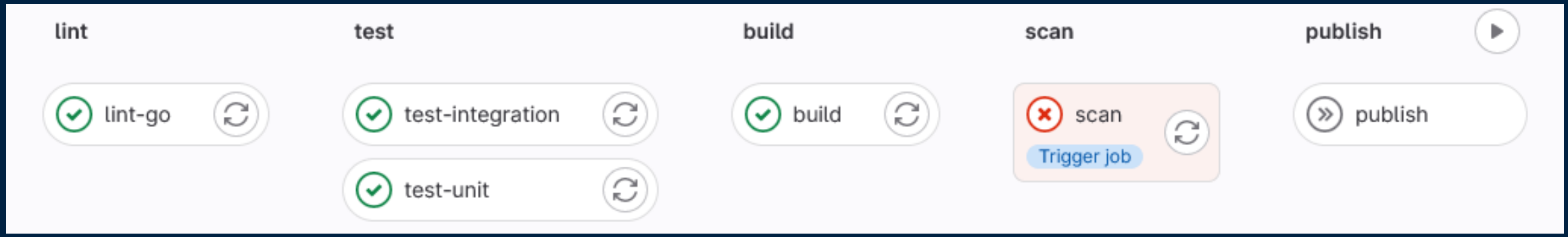
# Distribute. App Evolution.

## Upgrade Library



# Distribute. App Evolution.

## Pipeline Status



# Distribute. App Evolution.

## Scan Status

```
53 ID                  Severity  Package Name      Package Version  Fixed In
54 GHSA-fxg5-wq6x-vr4w  High      golang.org/x/net  v0.1.0           0.1.1-0.20221104162952-702349b0e862
55 GHSA-vvpx-j8f3-3w6h  High      golang.org/x/net  v0.1.0           0.7.0
56 + jq -s '.[0] * .[1]' grype_5fc7d0bf-0f2e-4ef9-b641-e5c7e02a2682.json syft_5fc7d0bf-0f2e-4ef9-b641-e5c7e02a2682.json
57 + exit 1
59 Running after_script
60 Running after script...
61 $ rm image.tar
63 Cleaning up project directory and file based variables
65 ERROR: Job failed: exit status 1
```

# Distribute. App Evolution.

## Add Transitive Dependencies

src/go.mod — View file @b10e0018

```
@@ -4,3 +4,36 @@ go 1.20

 require github.com/hashicorp/go-getter v1.7.0

+require (
+	cloud.google.com/go v0.104.0 // indirect
+	cloud.google.com/go/compute v1.10.0 // indirect
+	cloud.google.com/go/iam v0.5.0 // indirect
+	cloud.google.com/go/storage v1.27.0 // indirect
+	github.com/aws/aws-sdk-go v1.44.122 // indirect
+	github.com/bgentry/go-netrc v0.0.0-20140422174119-9fd32a8b3d3d // indirect
+	github.com/golang/groupcache v0.0.0-20210331224755-41bb18bfe9da // indirect
+	github.com/golang/protobuf v1.5.2 // indirect
+	github.com/google/go-cmp v0.5.9 // indirect
+	github.com/google/uuid v1.3.0 // indirect
+	github.com/googleapis/enterprise-certificate-proxy v0.2.0 // indirect
+	github.com/googleapis/gax-go/v2 v2.6.0 // indirect
+	github.com/hashicorp/go-cleanhttp v0.5.2 // indirect
+	github.com/hashicorp/go-safetemp v1.0.0 // indirect
+	github.com/hashicorp/go-version v1.6.0 // indirect
+	github.com/jmespath/go-jmespath v0.4.0 // indirect
+	github.com/klauspost/compress v1.15.11 // indirect
+	github.com/mitchellh/go-homedir v1.1.0 // indirect
+	github.com/mitchellh/go-testing-interface v1.14.1 // indirect
+	github.com/ulikunitz/xz v0.5.10 // indirect
+	go.opencensus.io v0.23.0 // indirect
+	golang.org/x/net v0.1.0 // indirect
+	golang.org/x/oauth2 v0.1.0 // indirect
+	golang.org/x/sys v0.1.0 // indirect
+	golang.org/x/text v0.4.0 // indirect
+	golang.org/x/xerrors v0.0.0-20220907171357-04be3eba64a2 // indirect
+	google.golang.org/api v0.100.0 // indirect
+	google.golang.org/appengine v1.6.7 // indirect
+	google.golang.org/genproto v0.0.0-20221025140454-527a21cfbd71 // indirect
+	google.golang.org/grpc v1.50.1 // indirect
+	google.golang.org/protobuf v1.28.1 // indirect
+)
```

# Distribute. App Evolution.

## Fix Dependencies

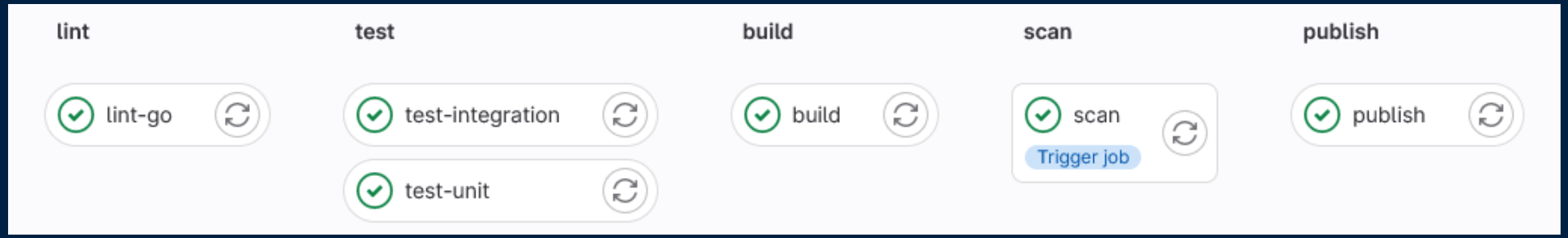
src/go.mod

View file @98317fc2

```
...  ...  @@ -26,7 +26,7 @@ require (
26   26           github.com/mitchellh/go-testing-interface v1.14.1 // indirect
27   27           github.com/ulikunitz/xz v0.5.10 // indirect
28   28           go.opencensus.io v0.23.0 // indirect
29        -       golang.org/x/net v0.1.0 // indirect
     29   +       golang.org/x/net v0.7.0 // indirect
30   30           golang.org/x/oauth2 v0.1.0 // indirect
31   31           golang.org/x/sys v0.1.0 // indirect
32   32           golang.org/x/text v0.4.0 // indirect
...  ...
```

# Distribute. App Evolution.

## Clean Build



# Distribute

## Scanning

- Software Composition Analysis
- Malware Scanning
- Image Hardening
  - Non-root containers
  - Secrets
  - Approved base images

# Distribute

## SBOM Analytics

- Storing SBOMs
- Querying and Analytics

```
search_package log4j 2.14
quay.io/nickmerrett/spring-boot-log4j-vulnerable:0.1
```

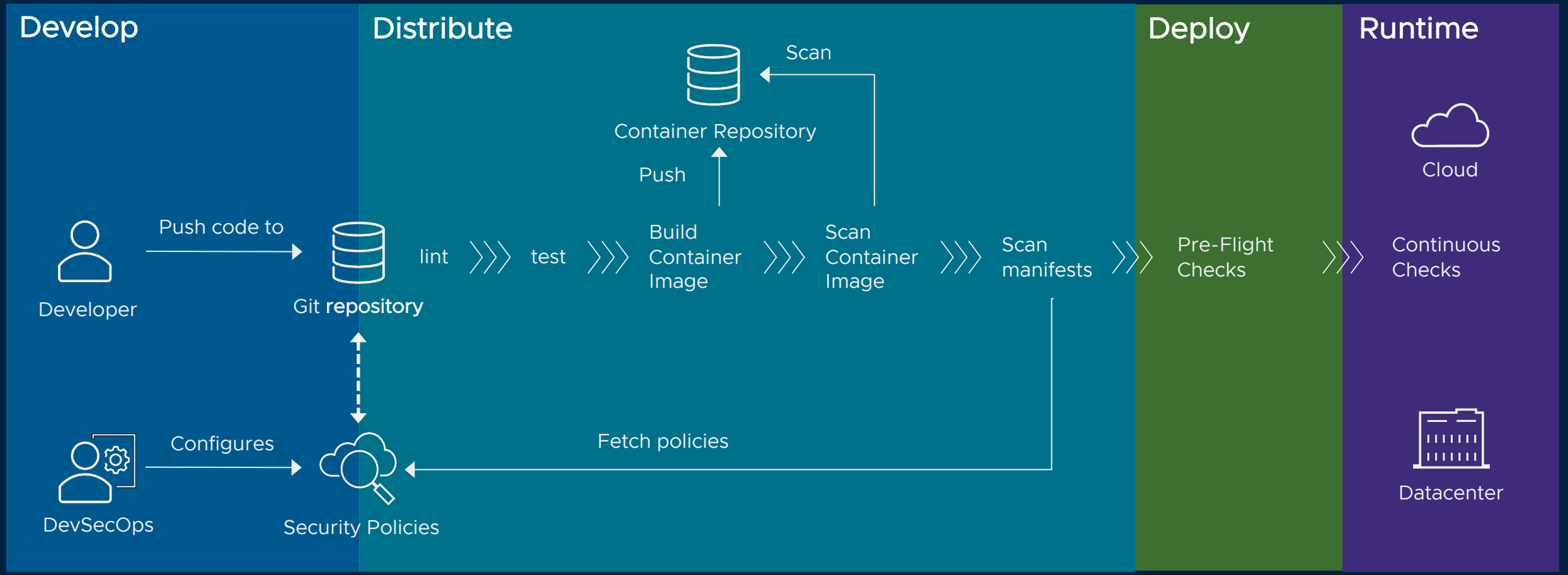
```
search_package go-getter
registry.gitlab.com/cis-workshop/app:98317fc2
registry.gitlab.com/cis-workshop/app:23f4f775
registry.gitlab.com/cis-workshop/app:db036310
docker.io/hashicorp/terraform:latest

search_package go-getter 1.5
registry.gitlab.com/cis-workshop/app:db036310
```

```
search_package openssl 1.1.1
docker.io/library/centos:latest
quay.io/ibm/jetty:11.0-jdk11
```

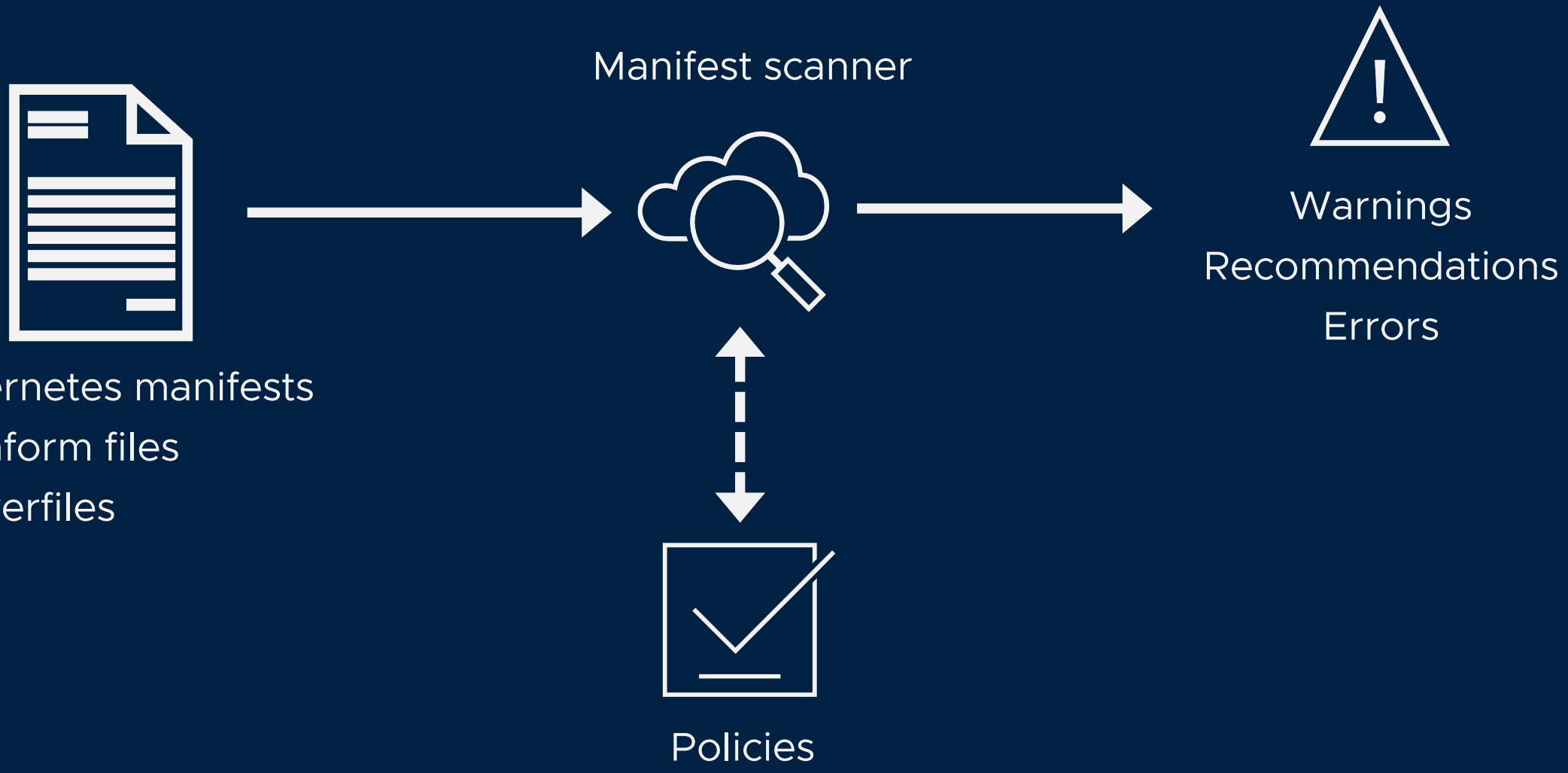
# Distribute: Manifest Scanning

# Cloud Native Continuous Lifecycle



# Distribute

## Manifest Scanning



# Distribute

## Manifest Scanning Example

**Scenario:** Prevent Kubernetes **Deployment** objects that run as **privileged** containers

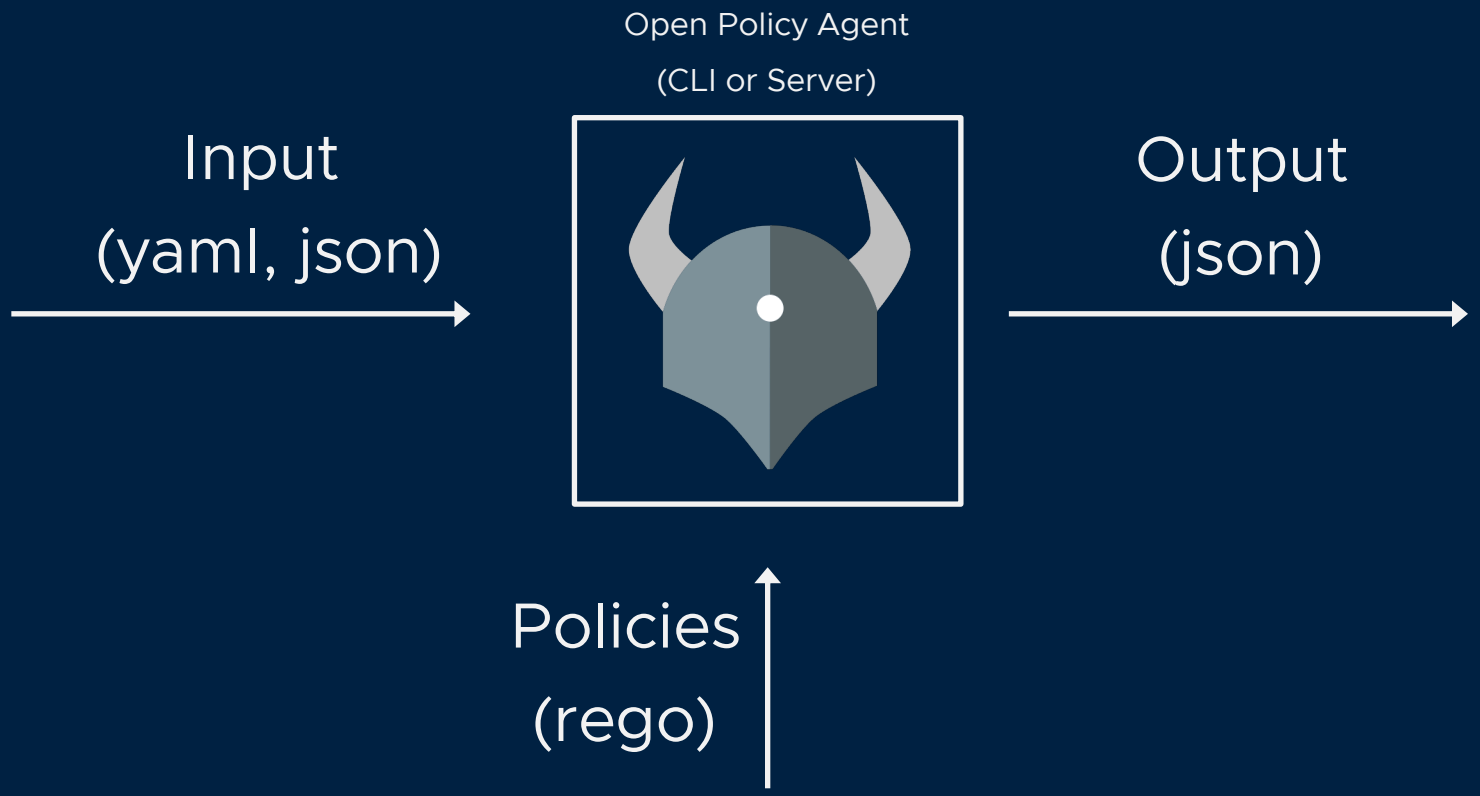
**Solution:** Use **Open Policy Agent** to validate the Kubernetes YAML Manifest

**Reason:** Running privileged containers is a bad practice, because if a privileged container is compromised an attacker can get access to the whole node.

https://www.openpolicyagent.org/
https://play.openpolicyagent.org/

# Open Policy Agent

## An open-source policy agent




https://www.openpolicyagent.org/
https://play.openpolicyagent.org/

# A Crash Course in Rego

## The policy language for OPA

### A simple Rego policy

```
1  package play
2
3  default hello := false
4
5  hello = true {
6      input.message = "world"
7  }
8
```

**INPUT**

```
1  {
2      "message": "world"
3  }
```

**OUTPUT**

```
Found 1 result in 34µs.
1  {
2      "hello": true
3  }
```

**INPUT**

```
1  {
2      "message": "everyone"
3  }
```

**OUTPUT**

```
Found 1 result in 33µs.
1  {
2      "hello": false
3  }
```

# Let’s use OPA to validate Kubernetes YAML files

## Rego policy

```
1  package noprivileged
2
3  violations[{"msg": msg}] {
4    c := input.spec.template.spec.containers[_]
5    c.securityContext.privileged
6    msg := sprintf("Container '%s' is set to run as privileged. This is unsafe.", [c.name])
7  }
```

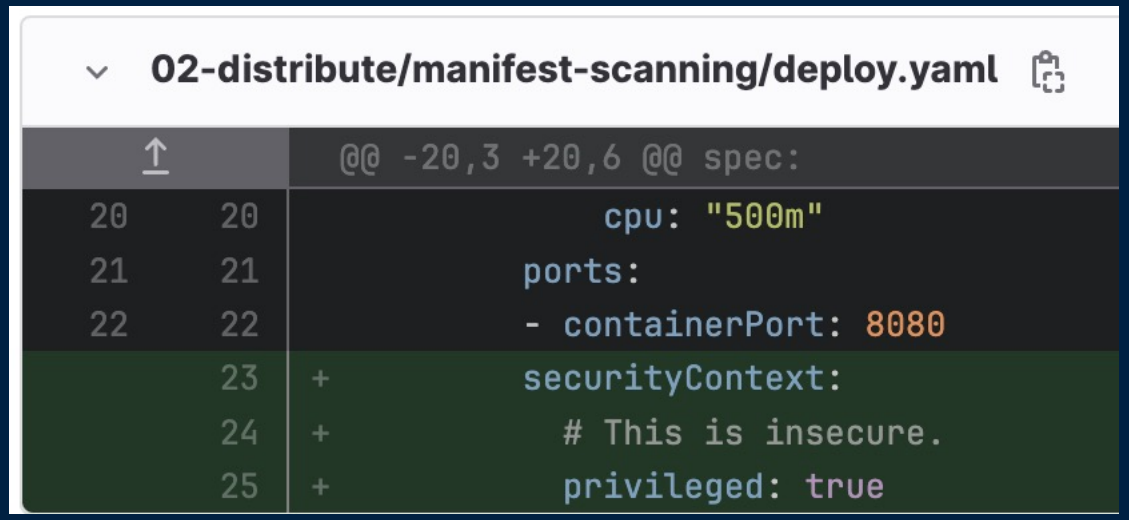
## CI Workflow

```
45 manifest-scan-opa:
46   stage: manifest-scan
47   image:
48     name: registry.gitlab.com/asankov/cloud-native-security:opa
49   script:
50     - cd 02-distribute/manifest-scanning
51     - opa eval -i deploy.yaml -d policy.rego 'data.noprivileged.violations[_].msg' --fail-defined
```

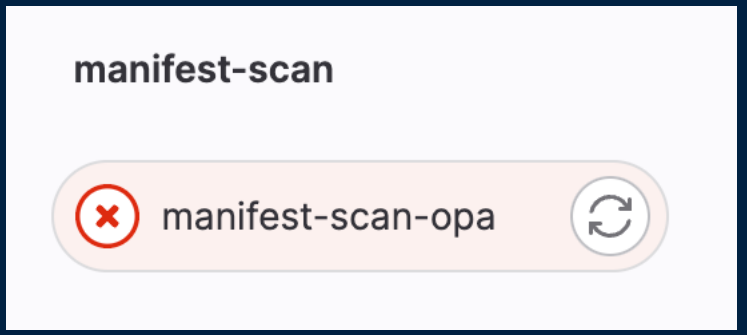
https://gitlab.com/asankov/cloud-native-security/-/blob/main/.gitlab-ci.yml

# Let’s use OPA to validate Kubernetes YAML files

A developer commits a Deployment that violates the policy




But the CI will fail and they will not be able to merge it




https://gitlab.com/asankov/cloud-native-security/-/merge_requests/4

# Let’s use OPA to validate Kubernetes YAML files

## The CI job output

```
$ opa eval -i deploy.yaml -d policy.rego 'data.noprivileged.violations[_].msg' --fail-defined
{
  "result": [
    {
      "expressions": [
        {
          "value": "Container 'my-app' is set to run as privileged. This is unsafe.",
          "text": "data.noprivileged.violations[_].msg",
          "location": {
            "row": 1,
            "col": 1
          }
        }
      ]
    }
  ]
}
Cleaning up project directory and file based variables
ERROR: Job failed: exit code 1
```

https://gitlab.com/asankov/cloud-native-security/-/merge_requests/4

# Distribute

## Manifest Scanning Example

This example uses a **local** policy.

This means that potentially someone could change the policy to fool the pipeline.

To mitigate this we can use **VMware Carbon Black Container Security** that uses policies stored in the cloud.

Developers don’t have access to the policies, so they have no way around them.

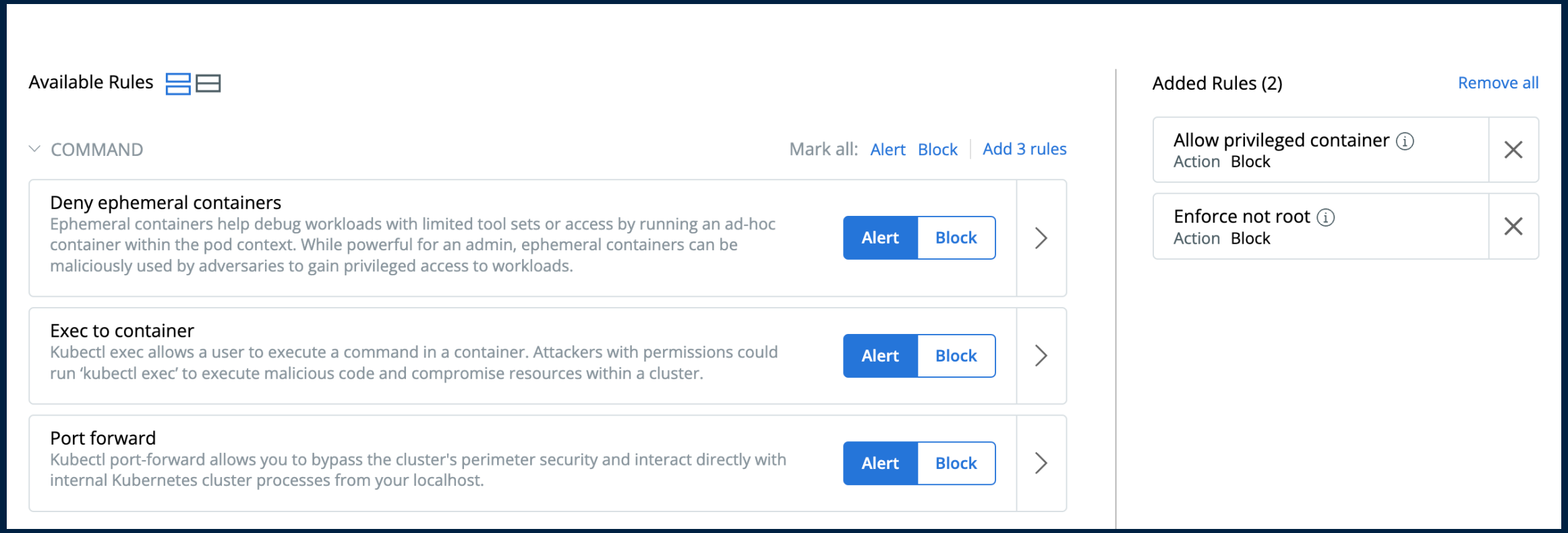
Another benefit of such solution is that we don’t have to write our policies from scratch.

# Distribute

## Manifest Scanning via Carbon Black – creating a policy

# Distribute

## Manifest Scanning via Carbon Black – assigning rules to the policy



# Distribute

## Manifest Scanning via Carbon Black

**Confirm Policy**

**General**

| | |
|---|---|
| Name | asankov-cloud-native-security |
| Scope | asankov-cloud-native-security |
| Include init containers | No |

**Rules**

| RULE ▲ | ACTION | STATUS |
|---|---|---|
| Allow privileged container ⓘ | Block | Enabled |
| Enforce not root ⓘ | Block | Enabled |

# Distribute

## Manifest Scanning via Carbon Black – using the CLI

After configuring the policies we can use the VMware Carbon Black CLI to enforce the policies in our CI.

```
54  manifest-scan-carbon-black:
55    stage: manifest-scan
56    image:
57      name: registry.gitlab.com/asankov/cloud-native-security:cbctl
58    script:
59      - cd 02-distribute/manifest-scanning
60      - cbctl k8s-object validate -f deploy.yaml --org-key $CBCTL_ORG_KEY --build-step gitlab
61        --cb-api-id $CBCTL_API_ID --cb-api-key $CBCTL_API_KEY --saas-url $CBCTL_URL
62
```

https://gitlab.com/asankov/cloud-native-security/-/blob/main/.gitlab-ci.yml

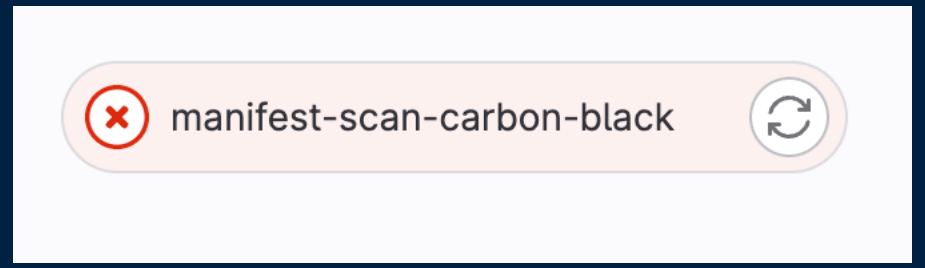
# Let’s use Carbon Black to validate Kubernetes YAML files

A developer commits a Deployment that violates the policy (again)

02-distribute/manifest-scanning/deploy.yaml

```
@@ -23,3 +23,5 @@ spec:
23  23          securityContext:
24  24            # This is secure.
25  25            privileged: false
    26  +         # This is insecure.
    27  +         runAsUser: 0 # 0 is root
```

But the CI will fail (again) and they will not be able to merge it




https://gitlab.com/asankov/cloud-native-security/-/merge_requests/5

# Let’s use Carbon Black to validate Kubernetes YAML files

## The CI job Output

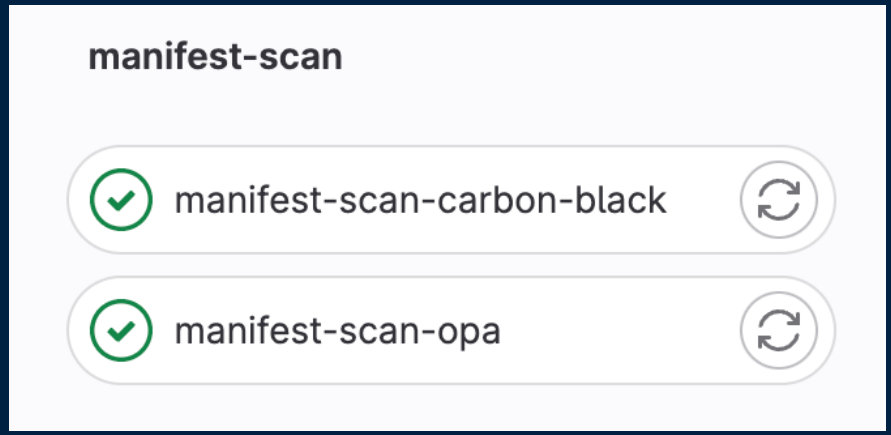
```
$ cbctl k8s-object validate -f deploy.yaml --org-key $CBCTL_ORG_KEY --build-step gitlab --cb-api-id $CBCTL_API_ID --cb-api-key $CBCTL_API_KEY --saas-url $CBCTL_URL
Found 1 violations for policy "asankov-cloud-native-security":
+-----------+------------+--------+-----------------+------+-------------+
| NAMESPACE |    KIND    |  NAME  |       RULE      | RISK |    FILE     |
+-----------+------------+--------+-----------------+------+-------------+
|           | Deployment | my-app | Enforce not root | LOW  | deploy.yaml |
+-----------+------------+--------+-----------------+------+-------------+
Detailed report can be found at
```

https://gitlab.com/asankov/cloud-native-security/-/merge_requests/5

# The developer is forced to fix the policy violations

**02-distribute/manifest-scanning/deploy.yaml**

```
@@ -21,7 +21,7 @@ spec:
          ports:
          - containerPort: 8080
          securityContext:
-           # This is insecure.
-           privileged: true
-           # This is also insecure.
-           runAsUser: 0
+           # This is secure.
+           privileged: false
+           # This is also secure.
+           runAsUser: 1000
```




https://gitlab.com/asankov/cloud-native-security/-/merge_requests/6

# Distribute

## Manifest Scanning

### Open Policy Agent

- policies are stored locally
- rules are written from scratch in Rego
- no predefined rules

### VMware Carbon Black

- policies are stored in the cloud
- there are predefined rules
- ability to write your own rules

# Distribute

## Manifest Scanning Alternatives

Kyverno

https://kyverno.io/

KubeSec

https://kubesec.io/

KubeScape

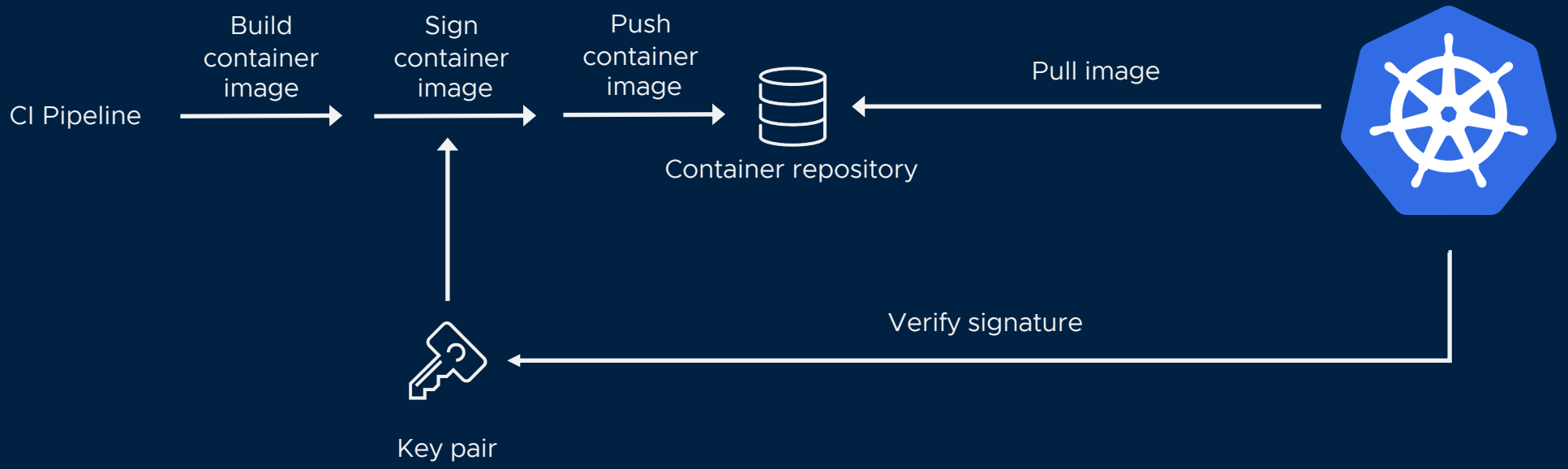
https://github.com/kubescape/kubescape

# Distribute: Artifact Signing

# Distribute

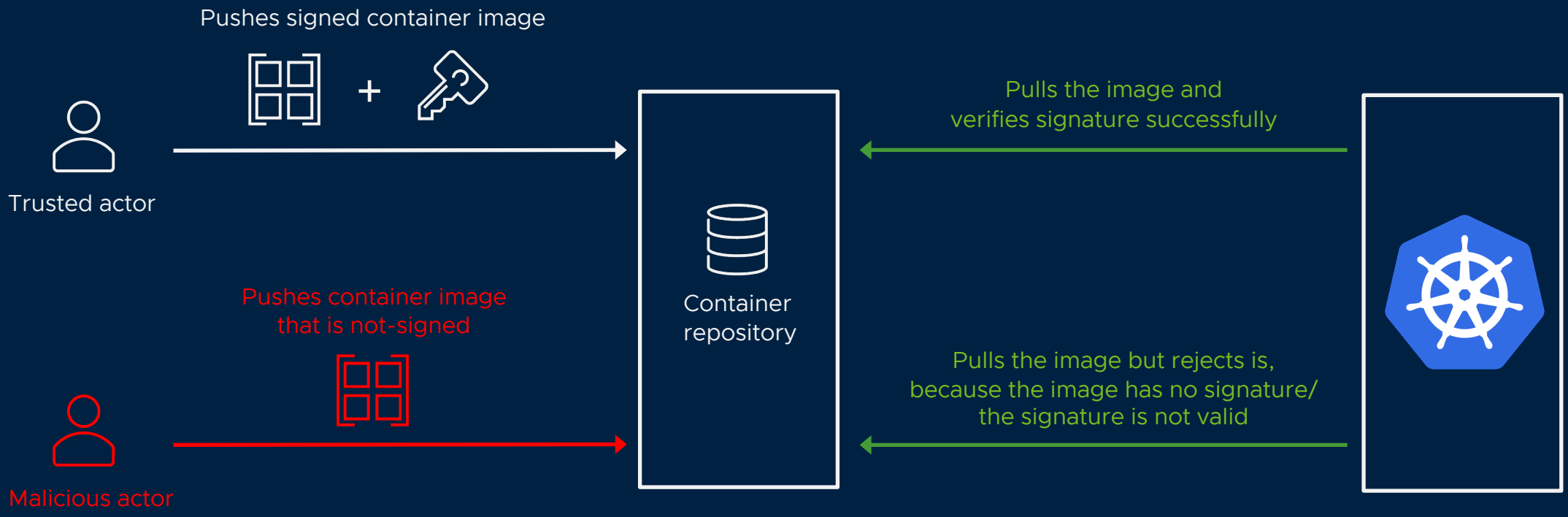
## Artifact Signing

The process of cryptographically signing your artifacts (for example, container images) in a secure way that can later be used to verify that the artifacts are trusted.



# Distribute

## Artifact Signing



# Distribute

## Artifact Signing Example

**Scenario: Sign** container images and verify their signature in the **Deploy** phase

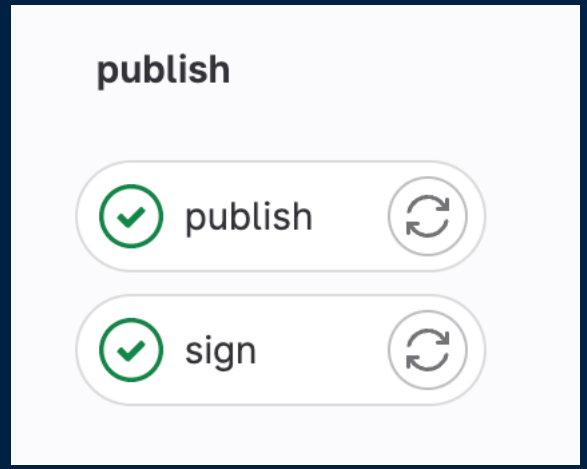
**Solution:** Use **Cosign** to sign and verify container images

**Reason:** Signing container images means we can verify authenticity and integrity. An attacker that managed to infiltrate our container repository will be able to push an image, but not be able to sign it, thus this image will never run in our environment.

# Let's use **Cosign** to sign our container image

## We are adding a new CI Workflow in the **Publish** stage

```
45  sign:
46    stage: publish
47    image:
48      name: registry.gitlab.com/asankov/cloud-native-security:cosign
49    script:
50      - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $CI_REGISTRY
51      - cosign sign --key $COSIGN_KEY -y registry.gitlab.com/asankov/cloud-native-security:cosign
```



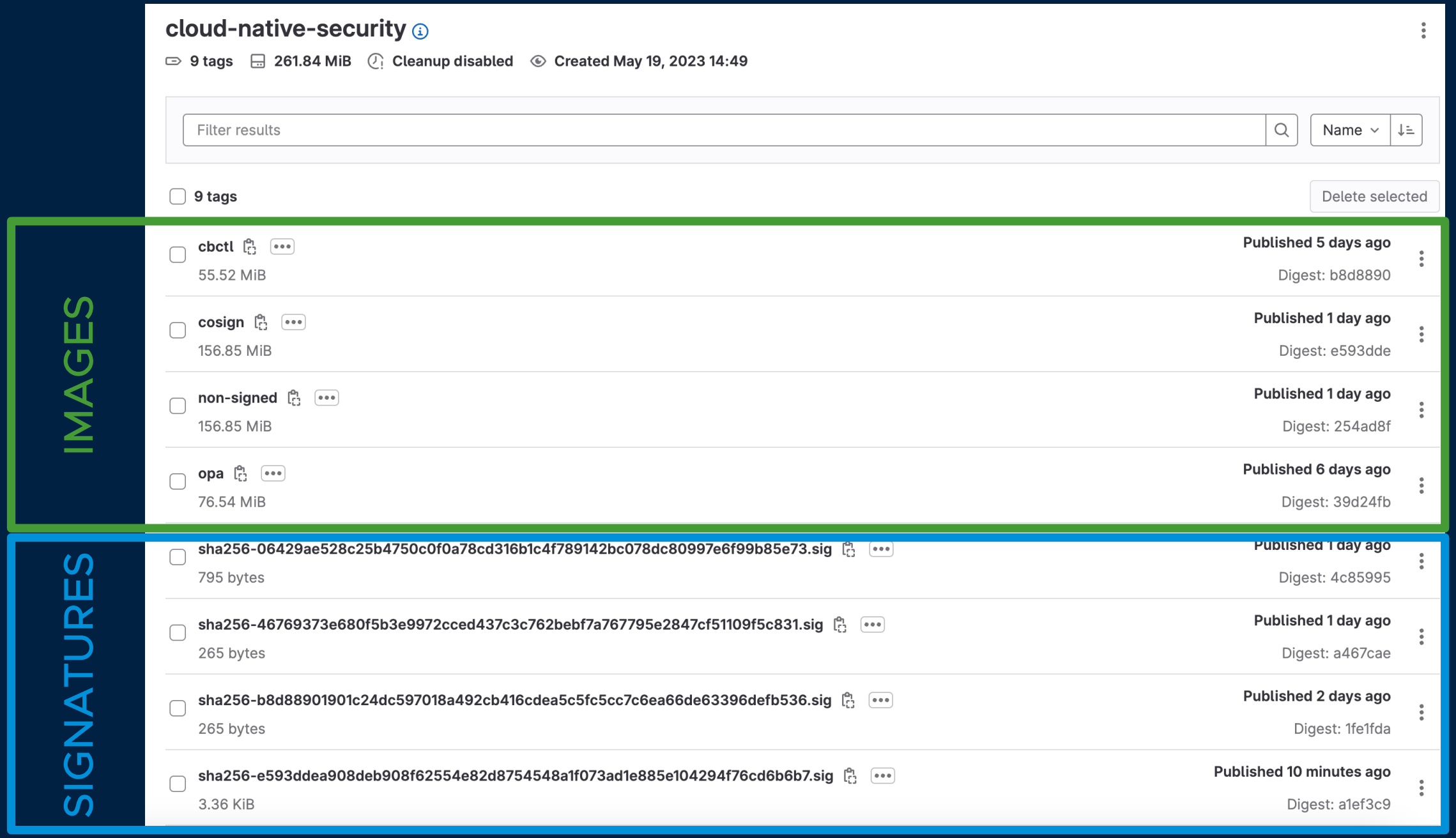

https://gitlab.com/asankov/cloud-native-security/-/blob/main/.gitlab-ci.yml

# Let's use **Cosign** to sign our container image

## CI Output

```
28 $ cosign sign --key $COSIGN_KEY -y registry.gitlab.com/asankov/cloud-native-security:cosign
29 WARNING: Image reference registry.gitlab.com/asankov/cloud-native-security:cosign uses a tag, not a digest, to identify the image to si
   gn.
30     This can lead you to sign a different image than the intended one. Please use a
31     digest (example.com/ubuntu@sha256:abc123...) rather than tag
32     (example.com/ubuntu:latest) for the input to cosign. The ability to refer to
33     images by tag will be removed in a future release.
34        The sigstore service, hosted by sigstore a Series of LF Projects, LLC, is provided pursuant to the Hosted Project Tools Terms of
   Use, available at https://lfprojects.org/policies/hosted-project-tools-terms-of-use/.
35        Note that if your submission includes personal data associated with this signed artifact, it will be part of an immutable recor
   d.
36        This may include the email address associated with the account with which you authenticate your contractual Agreement.
37        This information will be used for signing this artifact and will be stored in public transparency logs and cannot be removed lat
   er, and is subject to the Immutable Record notice at https://lfprojects.org/policies/hosted-project-tools-immutable-records/.
38 By typing 'y', you attest that (1) you are not submitting the personal data of any other person; and (2) you understand and agree to th
   e statement and the Agreement terms at the URLs listed above.
39 tlog entry created with index: 21567535
40 Pushing signature to: registry.gitlab.com/asankov/cloud-native-security
```

https://gitlab.com/asankov/cloud-native-security/-/jobs/4346059534

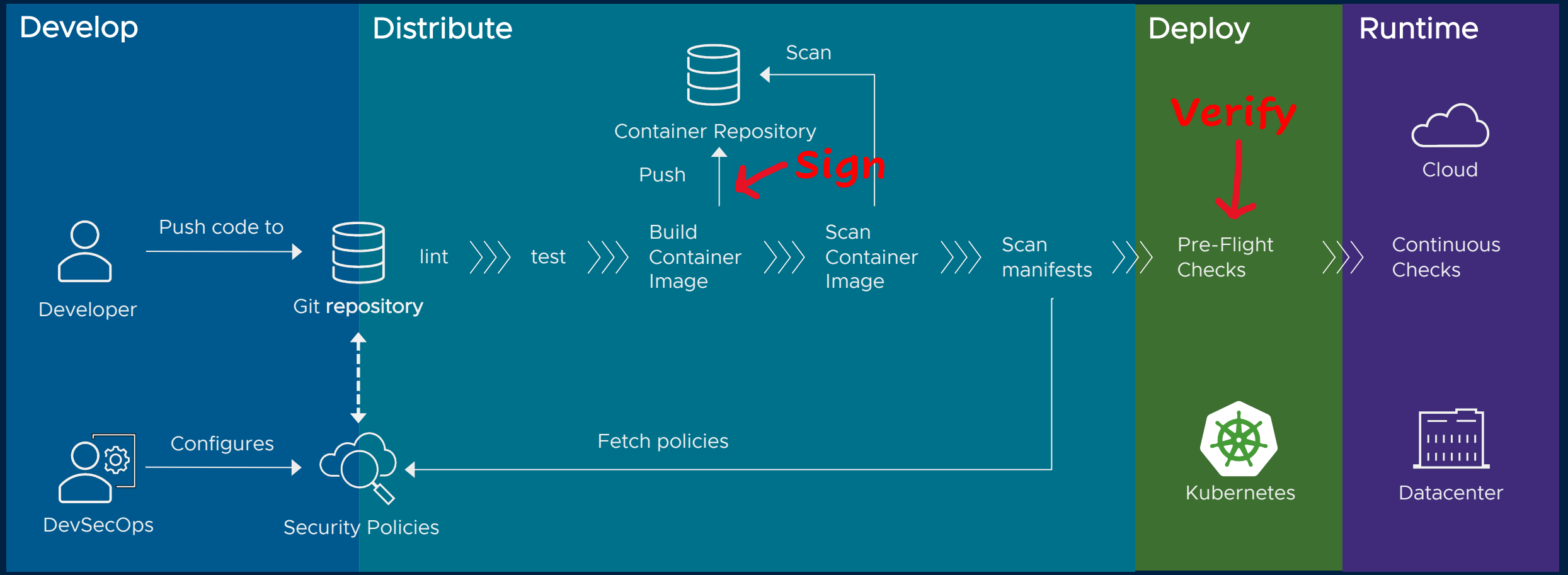

cloud-native-security
9 tags
261.84 MiB
Cleanup disabled
Created May 19, 2023 14:49
Filter results
Name
9 tags
Delete selected
IMAGES
cbctl
55.52 MiB
Published 5 days ago
Digest: b8d8890
cosign
156.85 MiB
Published 1 day ago
Digest: e593dde
non-signed
156.85 MiB
Published 1 day ago
Digest: 254ad8f
opa
76.54 MiB
Published 6 days ago
Digest: 39d24fb
SIGNATURES
sha256-06429ae528c25b4750c0f0a78cd316b1c4f789142bc078dc80997e6f99b85e73.sig
795 bytes
Digest: 4c85995
sha256-46769373e680f5b3e9972cced437c3c762bebf7a767795e2847cf51109f5c831.sig
265 bytes
Published 1 day ago
Digest: a467cae
sha256-b8d88901901c24dc597018a492cb416cdea5c5fc5cc7c6ea66de63396defb536.sig
265 bytes
Published 2 days ago
Digest: 1fe1fda
sha256-e593ddea908deb908f62554e82d8754548a1f073ad1e885e104294f76cd6b6b7.sig
3.36 KiB
Published 10 minutes ago
Digest: a1ef3c9

# Distribute

## Artifact Signing Example

This was the signing part, the next step is to verify the signature during the **Deploy** phase

# Cloud Native Continuous Lifecycle

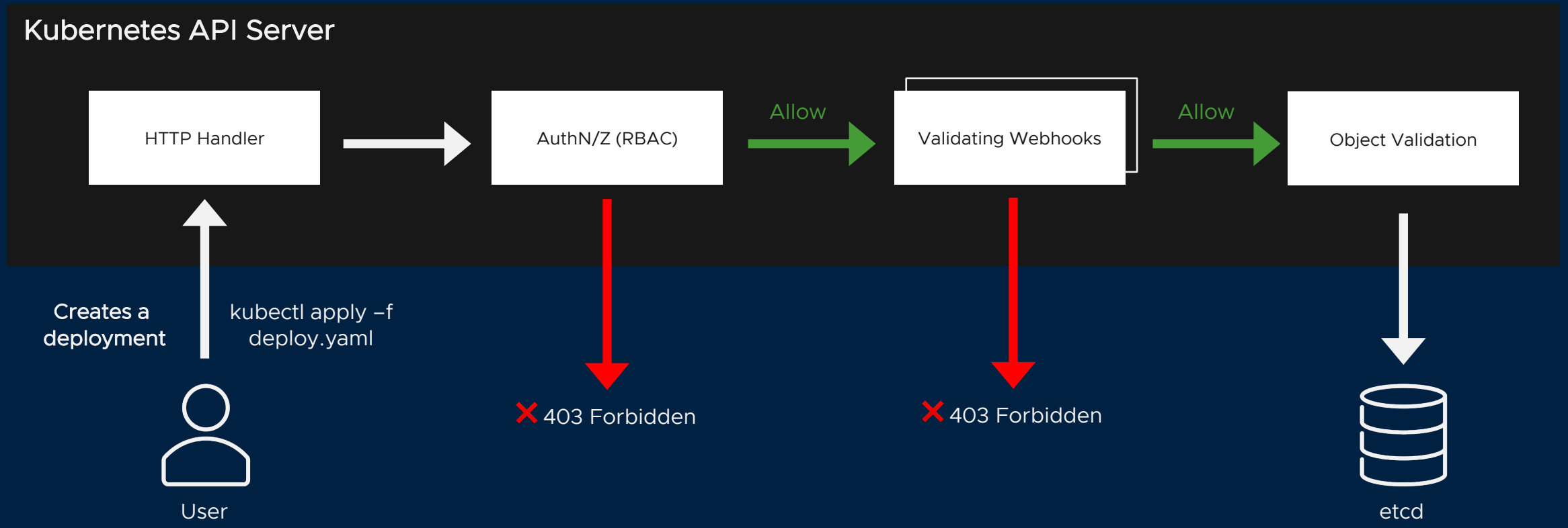


# Enter: Validating Webhooks

- Plugabble mechanism for adding custom verification logic to Kubernetes resources being created/updated
- Can have many of them, Kubernetes calls them in order
- If a validating webhooks denies the request, Kubernetes aborts the operation
- Anyone can write their own

https://kubernetes.io/docs/reference/access-authn-authz/validating-admission-policy/

# Validating Webhooks: Architectural view



# Validating Webhooks and Cosign

We can deploy a Validating Webhook that validates the image signatures.

Luckily, Cosign has already implemented this. We just need to install and configure it.

```
$ helm install policy-controller -n cosign-system sigstore/policy-controller
```

# Validating Webhooks and Cosign

Once we install the Cosign policy controller via Helm, we need to configure it:

```
apiVersion: policy.sigstore.dev/v1alpha1
kind: ClusterImagePolicy
metadata:
  name: cip-key-secret
spec:
  images:
    - glob: "**"
  authorities:
    - key:
        hashAlgorithm: sha256
        data: |
          -----BEGIN PUBLIC KEY-----
          MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEjwTyZK65jaqifzYNpv5D7Glrozx3
          Dnh6wB0ma4qeDrM2zVt8Y20kEXUVje7gMwgltQ2Om1M2+QqkfiU6eKGfMw==
          -----END PUBLIC KEY-----
```

# Validating Webhooks and Cosign

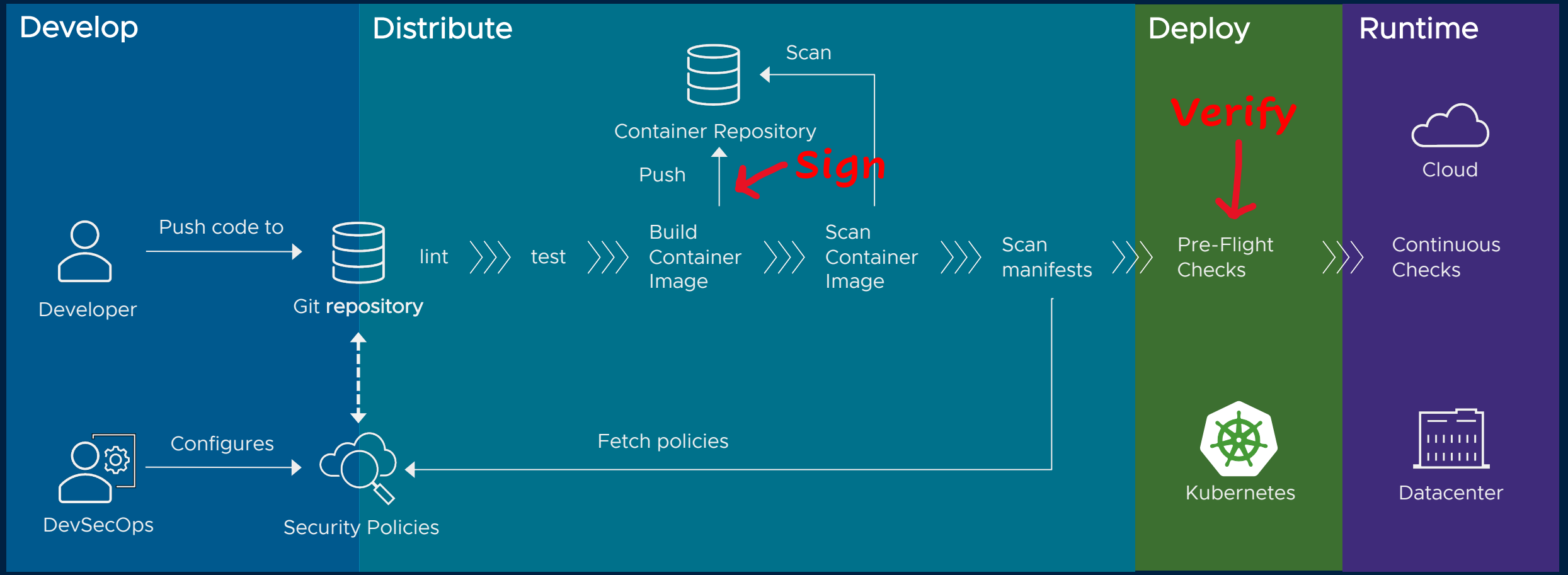
```
16      - name: non-signed-image
17        image: registry.gitlab.com/asankov/cloud-native-security:non-signed
```

```
$ kubectl apply -f non-signed-deploy.yaml
Error from server (BadRequest): error when creating "non-signed-deploy.yaml": admission webhook "policy.sigstore.dev" denied the req
uest: validation failed: failed policy: cip-key-secret: spec.template.spec.containers[0].image
registry.gitlab.com/asankov/cloud-native-security@sha256:254ad8f9fbeb3c346e6eeb5e5f8d16e687e775080213e0e4b8851ac2e8fa787d signature
key validation failed for authority authority-0 for registry.gitlab.com/asankov/cloud-native-security@sha256:254ad8f9fbeb3c346e6eeb5
e5f8d16e687e775080213e0e4b8851ac2e8fa787d: no matching signatures:
```

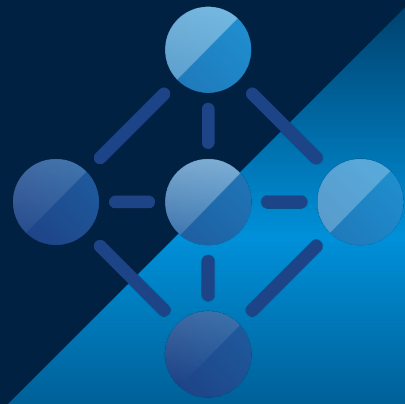
```
16      - name: signed-image
17        image: registry.gitlab.com/asankov/cloud-native-security:signed
```

```
$ kubectl apply -f signed-deploy.yaml
deployment.apps/signed-deploy created
$ kubectl get deployment -n my-secure-ns
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
signed-deploy   0/1     1            0           13s
```

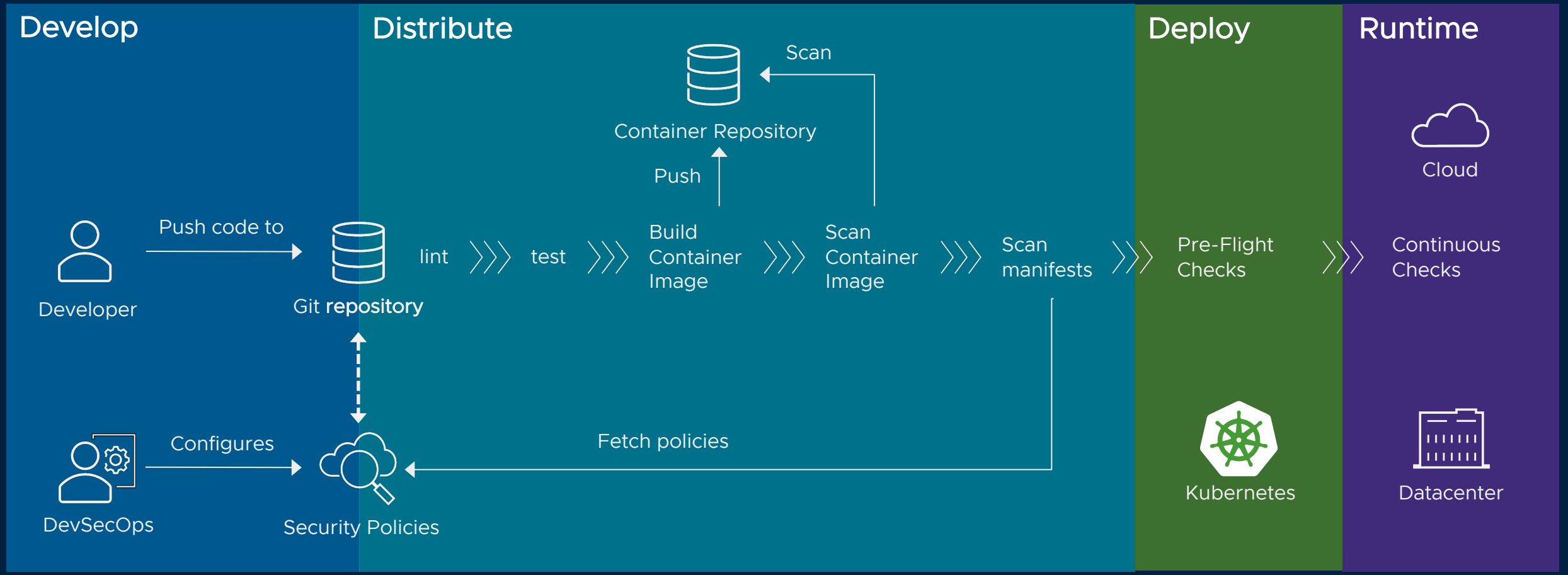
# Cloud Native Continuous Lifecycle



# Runtime



# Cloud Native Continuous Lifecycle



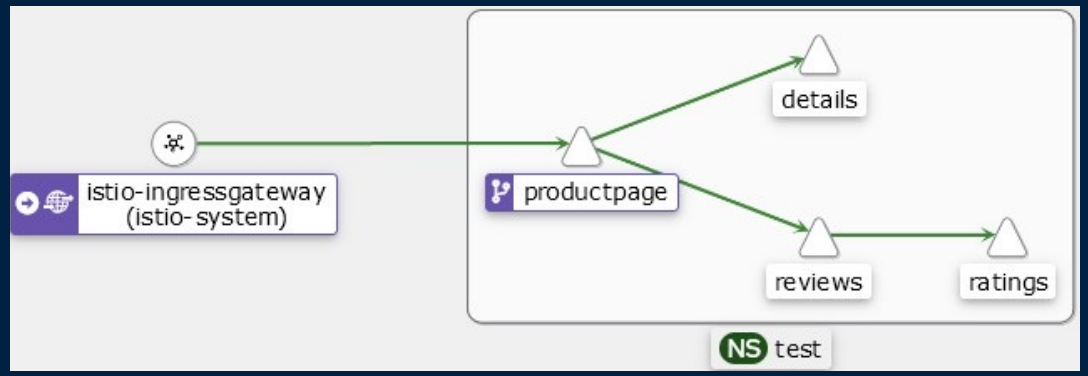
# Runtime

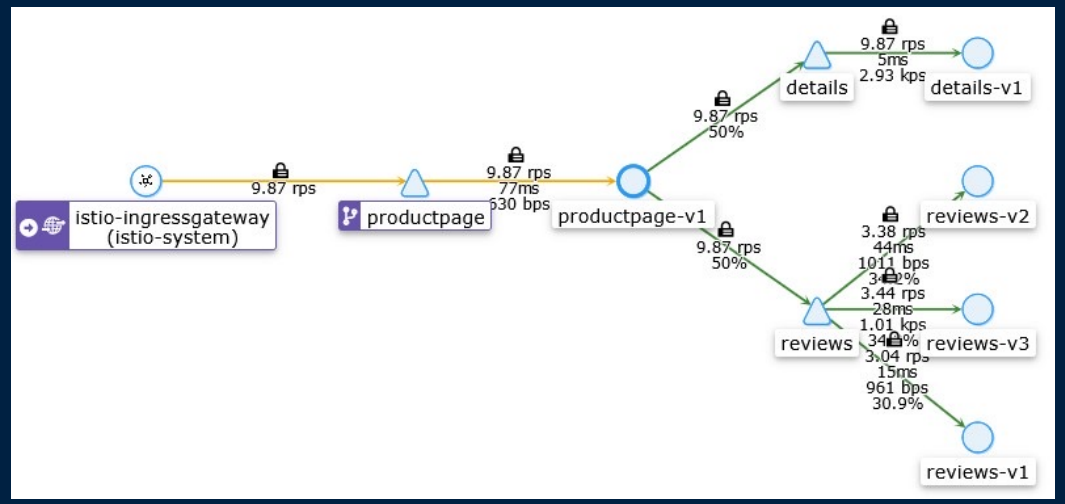
## Service Mesh

- Dedicated Infrastructure Layer
- Service-to-Service Communication (mTLS, AuthZ, AuthN)
- Service Discovery
- Traffic Control (routing, retries, failover, rate limits)
- Observability (metrics, logs, traces)

# Runtime

## Service Mesh



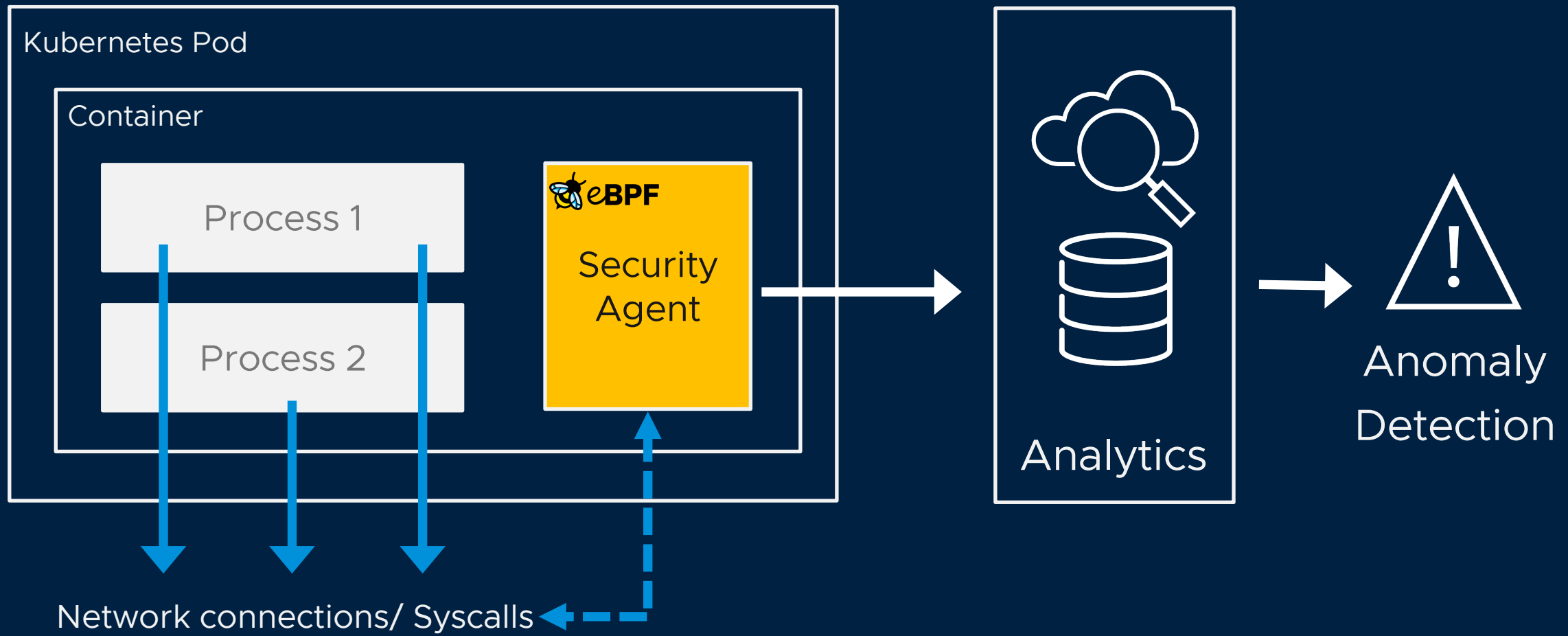

# Runtime

## Secrets Management

- Natively
  - No encryption
  - Encryption by the orchestrator
  - Encryption with external Key Management Service (KMS)
- External Secrets Manager
  - Inject secrets
  - Manage rotation
  - Ad-hoc credentials

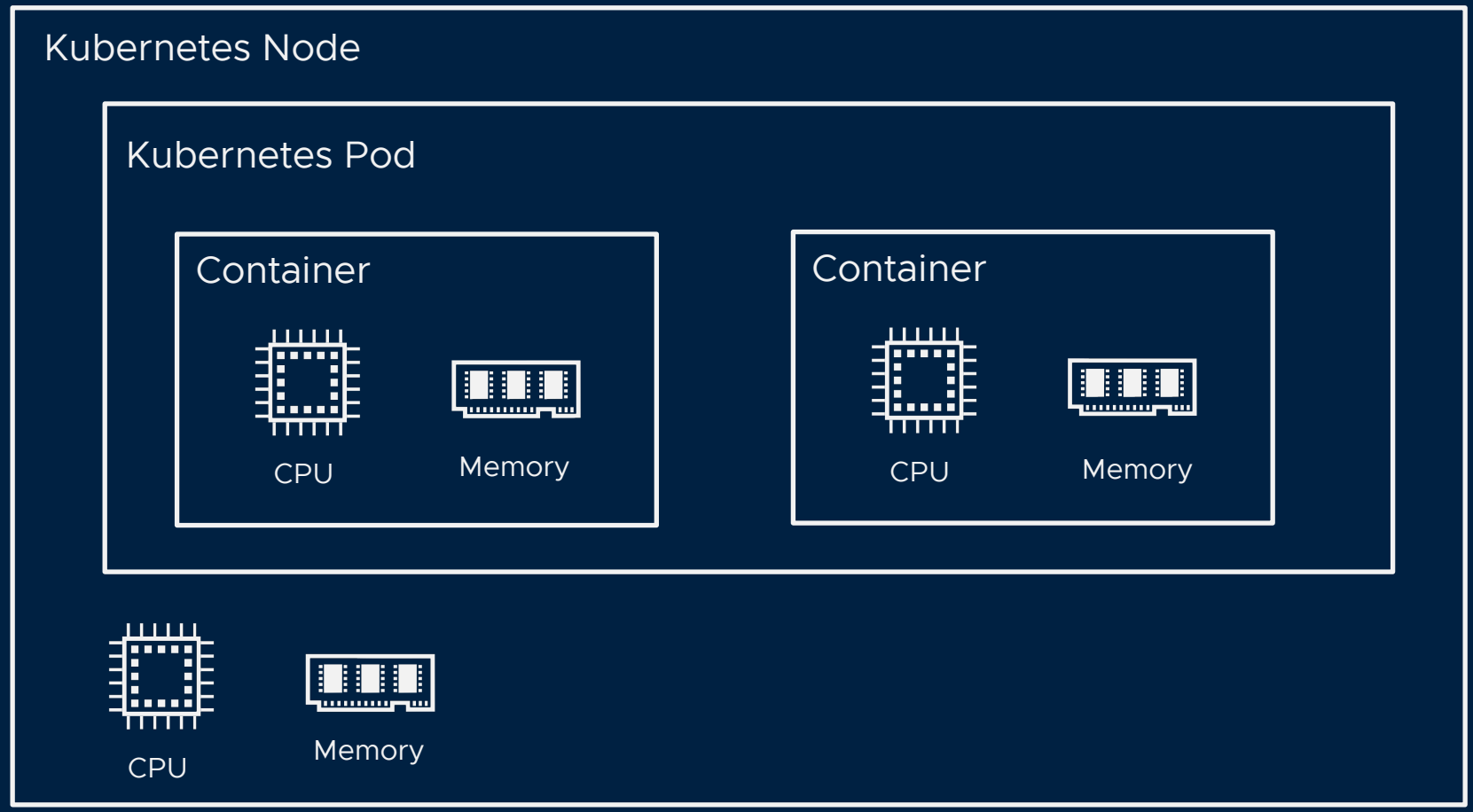
# Runtime

## Container behaviour



# Runtime

## Resource Limits



# Summary

# Thank You

Martin Georgiev
Anton Sankov

vmware® Confidential | © VMware, Inc.