KubeCon | CloudNativeCon

Europe 2022

WELCOME TO VALENCIA

KubeCon CloudNativeCon
Europe 2022

Fun With Continuous Compliance

Ann Wallace, Shopify
Zeal Somani, Google

Agenda

1. Intro, Fun and !Fun
2. What & Why Continuous Compliance
3. Shopify & Continuous Compliance
4. Google & Continuous Compliance
5. Q&A

% who am i

Zeal Somani

Security Solutions Manager

Google

Ann Wallace

@annnwallace

Sr. Manager, Security Eng

Shopify

Intro

Fun and !Fun

The Fun Scale*

Type 1 Fun – is enjoyable while it's happening. Also known as, simply, fun. Good food. Hanging out with friends, KubeCon, margaritas.

Type 2 Fun – is miserable while it's happening, but fun in retrospect. It usually begins with the best intentions, and then things get carried away. Friday deployments, running an ultramarathon.

Type 3 Fun – Not even fun in retrospect. Afterward, you think, "What in the hell was I doing? If I ever come up with another idea that stupid, somebody slap some sense into me." Being on-call for 6 weeks so a co-worker can go on an epic trip. Manual evidence gathering for a yearly audit.

*https://www.rei.com/blog/climb/fun-scale

Compliance != Fun || Compliance == Fun

What & Why Continuous Compliance

The dichotomy…

Developers

Teams responsible for security & compliance often work independently and in silos

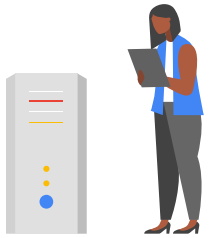
Security Operations

Compliance Analysts

Google Cloud

Developers are often left to interpret arcane compliance requirements that don't always seem relevant to cloud configuration.

Security Operations Teams often work in many tools with noisy alerts, creating fatigue and making it difficult to evaluate the compliance implication of a single alert



Compliance Analysts are operating in an increasingly rapidly growing decentralized environment, disrupting visibility and evidence collection

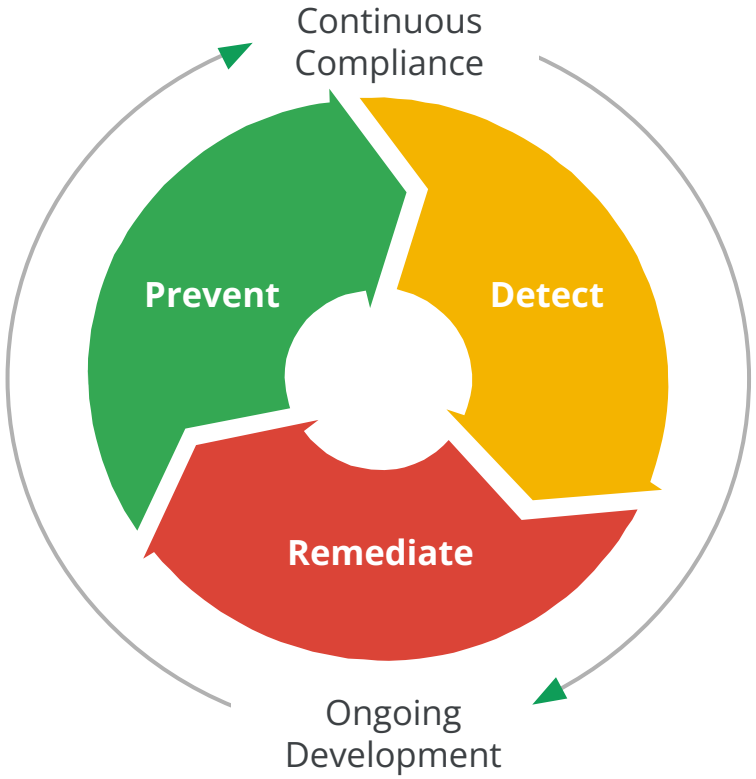
Google Cloud

© 2018 Google LLC. All rights reserved.

We are Headed Towards an Automated Compliance Controls Future

Why Continuous Compliance?

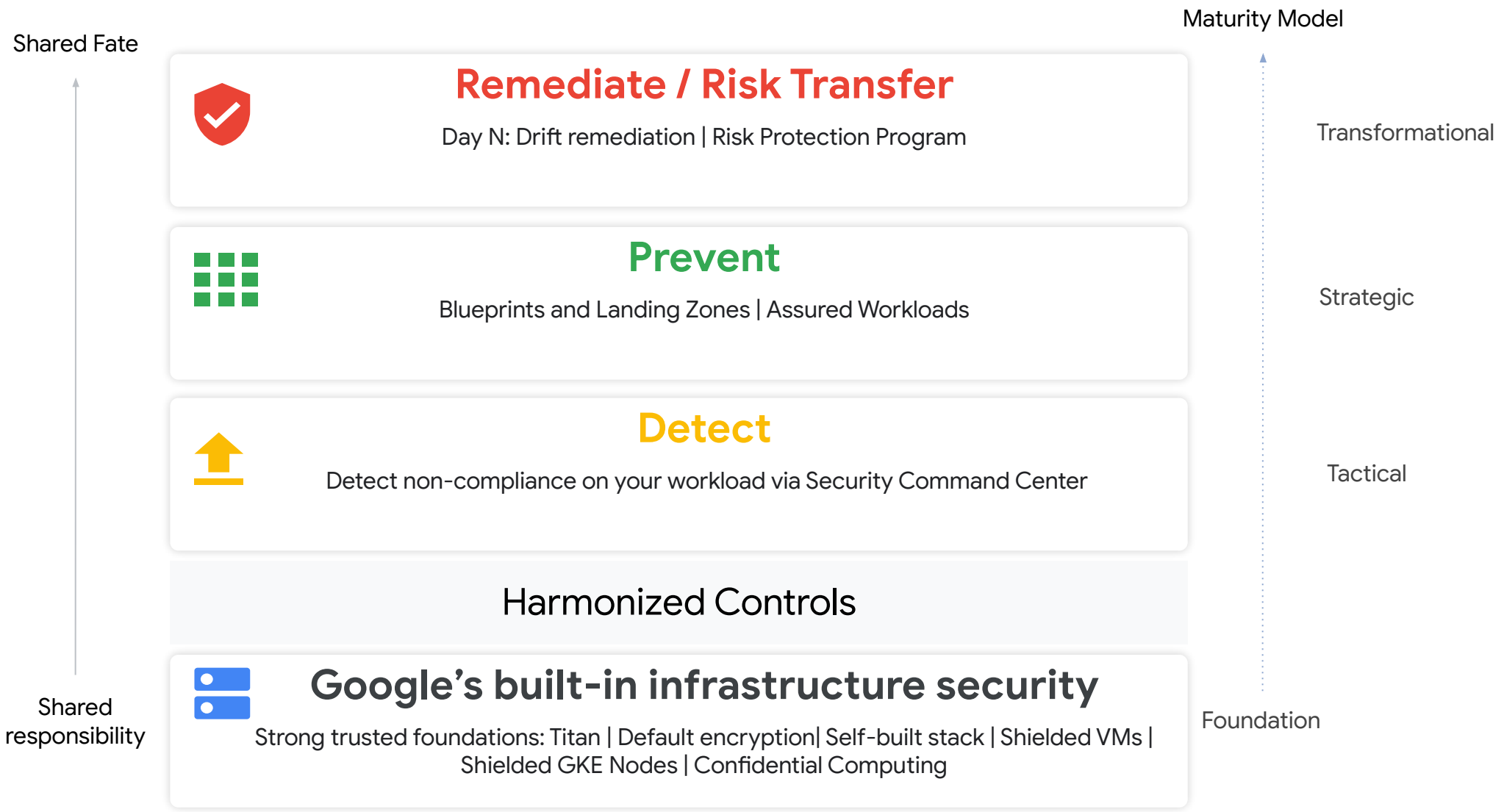
1. Real security and reduced risk
2. Lowered compliance and audit cost
3. Improved engineering experience



Additional Google Cloud resources: Risk and Compliance as Code

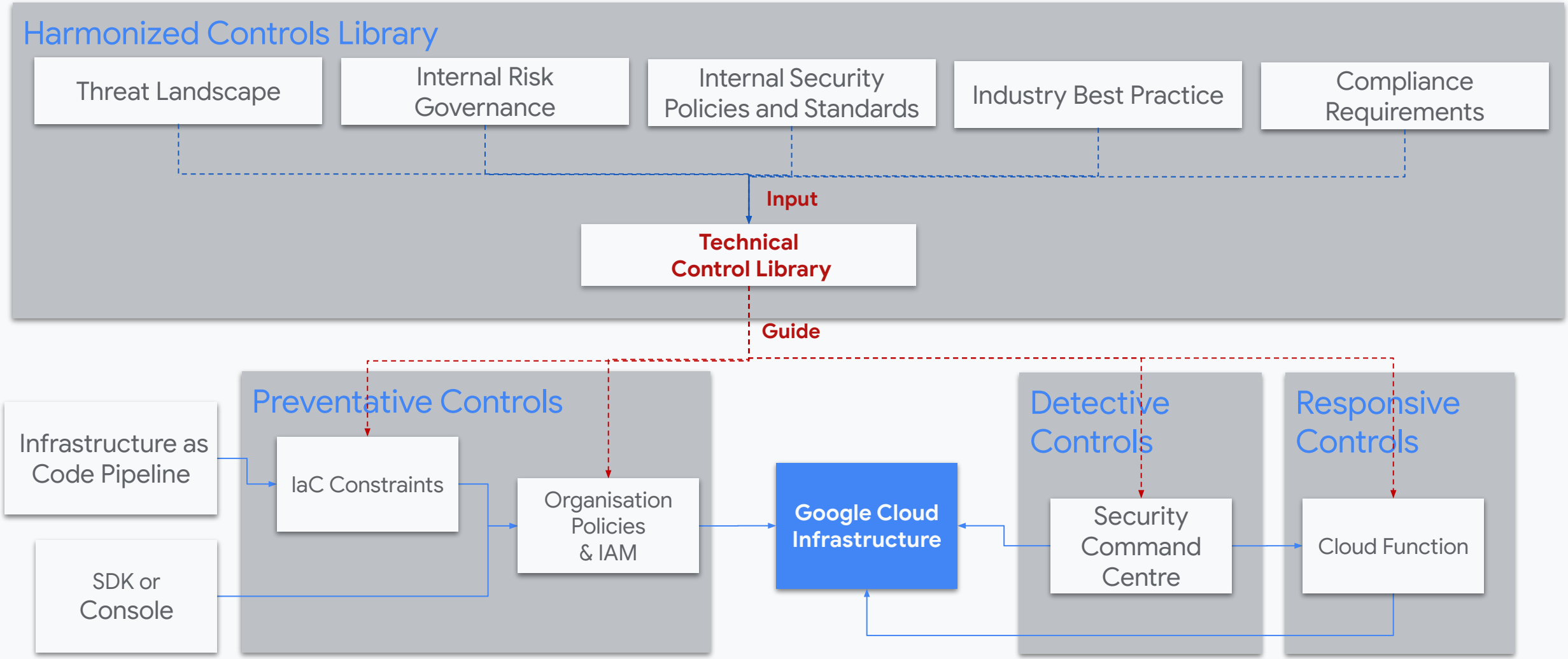
Risk and Compliance as Code (RCaC) on Google Cloud

Proprietary + Confidential



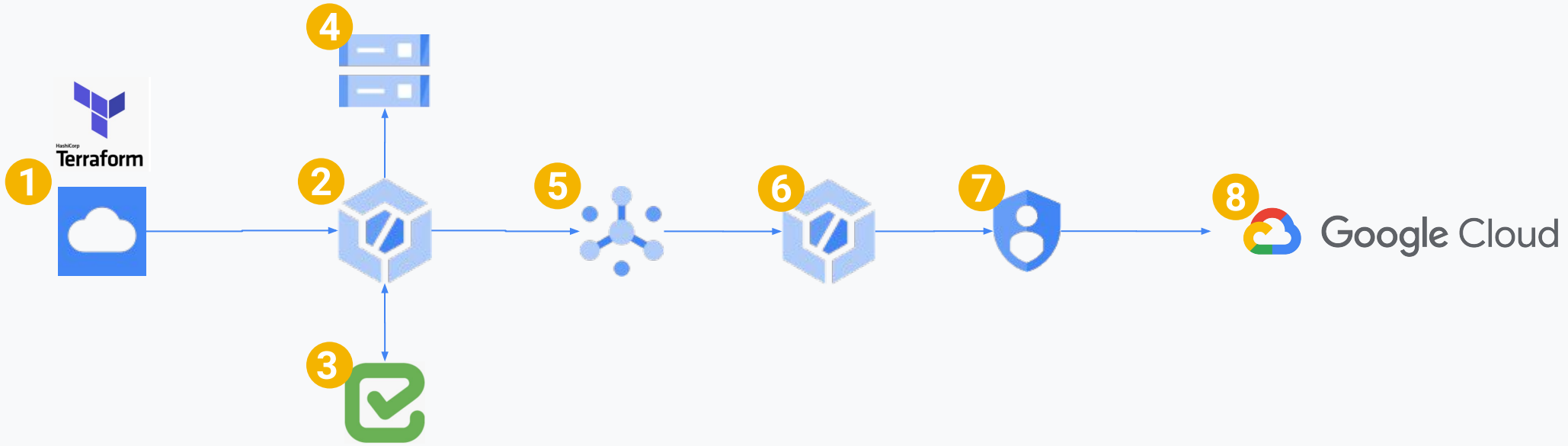
Infrastructure Continuous Compliance Reference Architecture

Proprietary + Confidential



Example - Preventative Controls

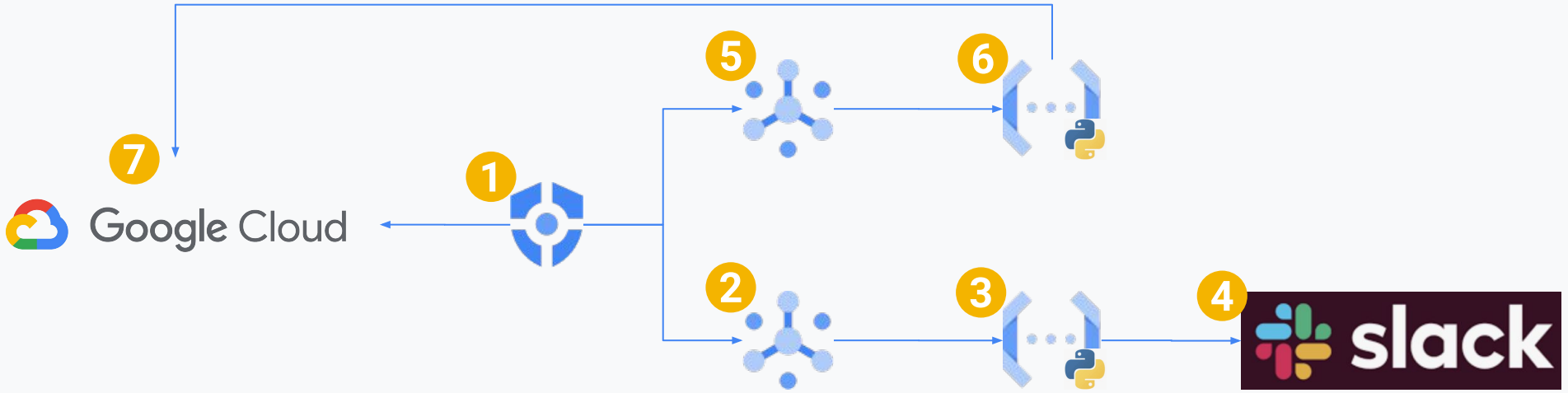
Proprietary + Confidential



Steps	Description
1	Terraform code stored in the Cloud Source Repository, code push to the main branch to trigger the Cloud Build
2, 3, 4, 5	Cloud Build runs Checkmarx KICS, KICS scan results are uploaded to the Cloud Storage; with any Medium or High findings, the build is failed; without Medium or High findings, a success message is pushed to Pub/Sub
6, 7, 8	Cloud Build will run Terraform init and apply; Organisation Policies will apply constraints close to the final GCP APIs; successful Terraform apply will build the GCP infrastructure

Example - Detective and Responsive Controls

Proprietary + Confidential

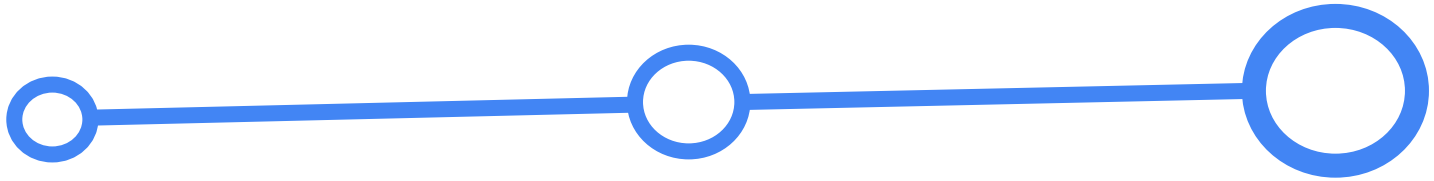


Steps	Description
1	Security Command Centre (SCC) continuous scan the GCP environment against desired state configurations, e.g., CIS Foundation, PCI DSS etc. OPEN_SSH_PORT is used in the demo.
2, 3, 4	Filtered SCC findings (OPEN_SSH_PORT) are pushed to the Pub/Sub, a Cloud Function using Python SDK is triggered and a message is posted to the Slack Channel for SRE or SecOps
5, 6, 7	Filtered SCC findings (OPEN_SSH_PORT) are pushed to the Pub/Sub, a Cloud Function using Python SDK is triggered

Proprietary + Confidential

Maturity Levels

Introduce continuous compliance capabilities over time as your organization matures



	TACTICAL	STRATEGIC	TRANSFORMATIONAL
	Detect	**Detect, Prevent**	**Detect, Prevent, Remediate**
Capabilities	**On-Demand Evidence Collection**	**Preventative controls expressed as code**	**Drift remediation, Cyber Insurance coverage**
Team Alignment	SecOps	Developers, SecOps, Platform engineers	Developers, SecOps, Platform engineers, Compliance Analysts
Skill-sets introduced	Threat detection, Log engineering	Infrastructure-as-Code, Policy as Code	Automate drift correction where applicable Continuous assurance via continuous audit metrics

shopify &

Continuous Compliance

whois shopify

At a Glance

Shopify is a provider of essential internet infrastructure for commerce.

2006
Platform Released

$4.6B
Revenue
(Last 12 months)

10,000
Employees
(December 31, 2021)

Millions
of merchants
— in —
>175 Countries

~$450B
Total sales by merchants
on Shopify
(December 31, 2021)

~600 million
Online shoppers
purchasing from Shopify
merchants in 2021

>40,000
Partners who have referred at least one merchant to Shopify in the last 12 months
(December 31, 2021)

>8,000
Apps in our App Store
(December 31, 2021)

$411M
Paid out in 2021 to partners by Shopify for apps benefiting our merchants

GCP & GKE Stats

250+
GKE Clusters

45k+
K8 Services

25k+
Builds a month

1200+
GCP Projects

Ok we get it **Shopify is doing lots of things**. How does this make **compliance fun**?

shopify

Shopify compliance programs

PCI DSS COMPLIANT

AICPA SOC
aicpa.org/soc4so
SOC for Service Organizations | Service Organizations™

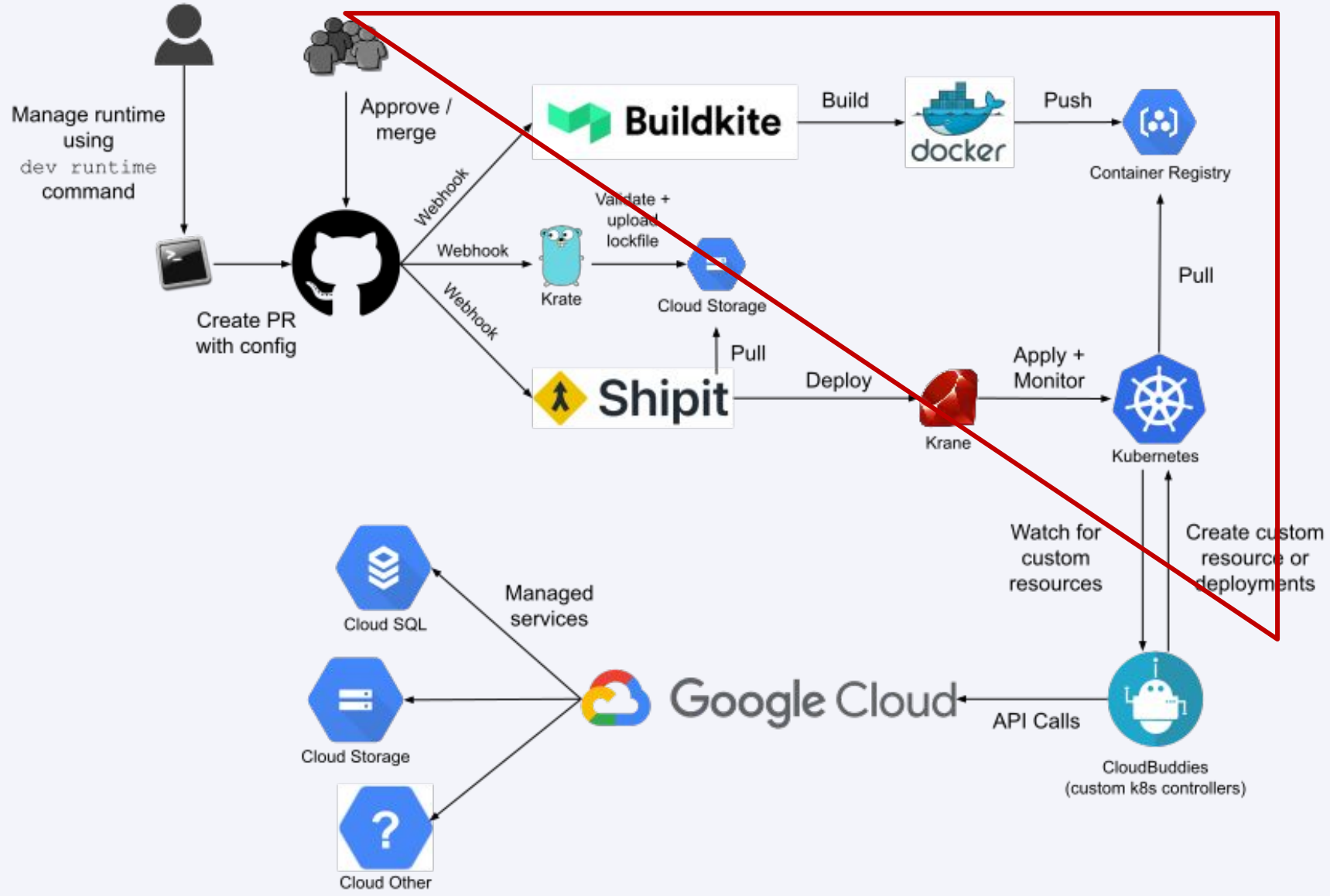
SOC II COMPLIANT

SOX
Sarbanes-Oxley Compliance

SWIFT

We build **~18k**
container images a day

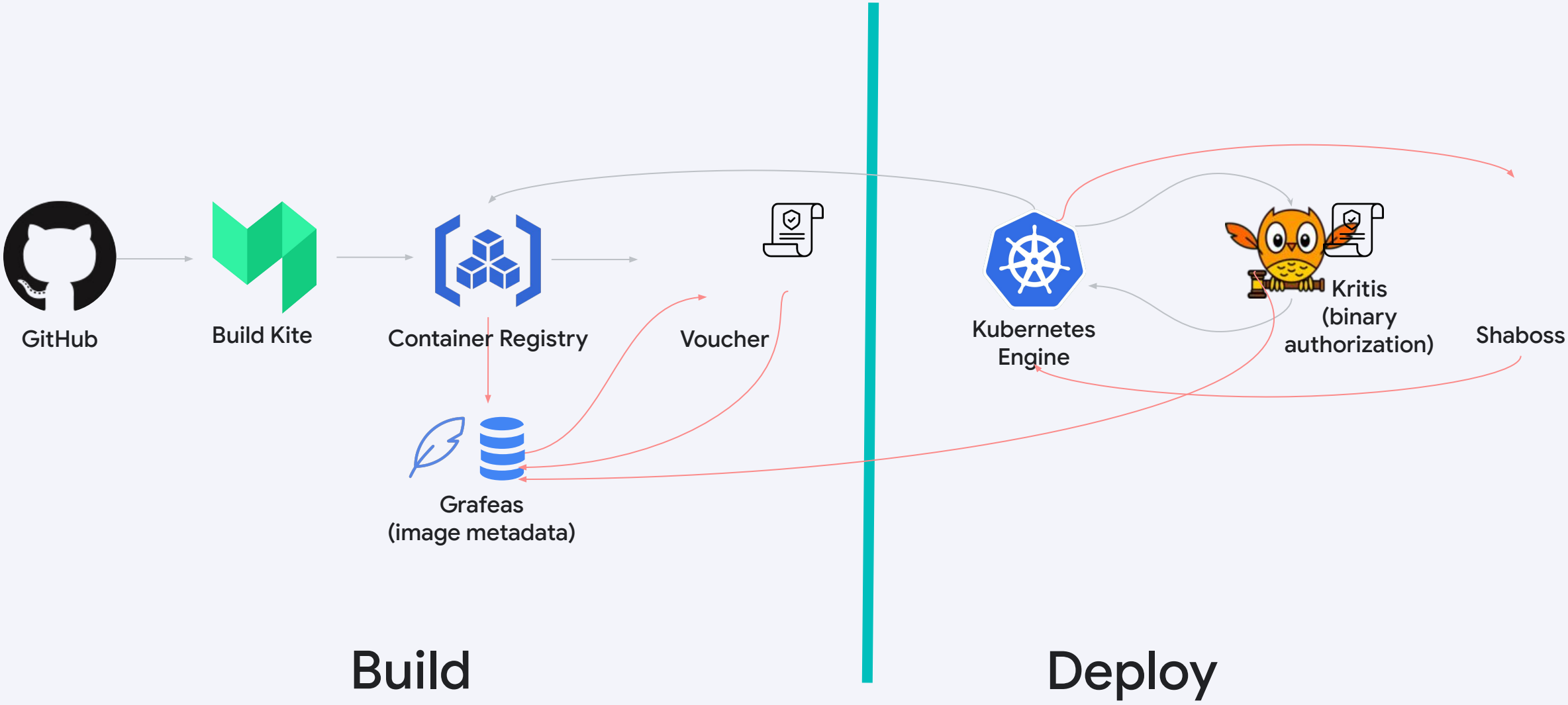
Production Build Pipeline



Here comes the fun part!

shopify

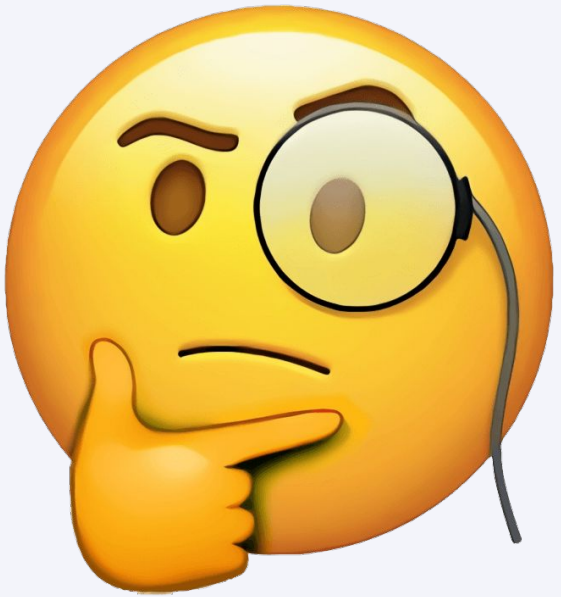
Binary Authorization and Voucher



Voucher out of the box checks

Test Name	Description
`diy`	Can the image be downloaded from our container registry?
`nobody`	Was the image built to run as a user who is not root?
`snakeoil`	Is the image free of known security issues?
`provenance`	Was the image built by us or a trusted system?
`approved`	Did the source code for the image pass all required checks in the code repository?
`is <org_name>`	Did the source for this image come from the passed organization (for example, is_shopify)?

PCI DSS 6.1: Establish a process to identify vulnerabilities using reputable outside sources and assign a risk ranking to newly discovered vulnerabilities



Snakeoil

Test Name	Description
`diy`	Can the image be downloaded from our container registry?
`nobody`	Was the image built to run as a user who is not root?
`snakeoil`	Is the image free of known security issues?
`provenance`	Was the image built by us or a trusted system?
`approved`	Did the source code for the image pass all required checks in the code repository?
`is <org_name>`	Did the source for this image come from the passed organization (for example, is_shopify)?

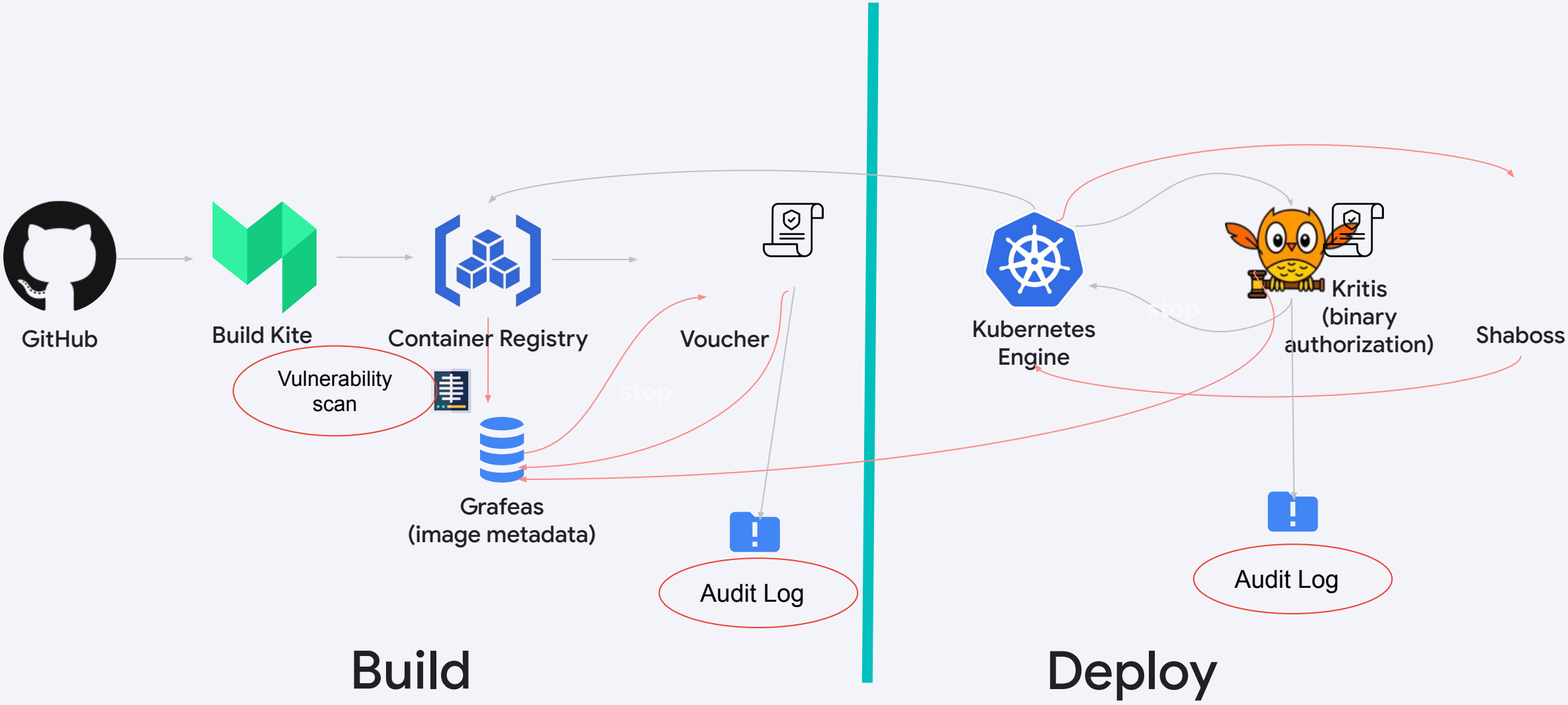
Voucher vulnerability policy

```
dryrun = false
scanner = "metadata"
failon = "high"
metadata_client = "grafeas"
image_project = "<PROJECT_ID>"
binauth_project = "<PROJECT_ID>"
signer = "kms"

[checks]
snakeoil = true

[[kms_keys]]
check = "snakeoil"
path = "projects/<GCP project>/locations/global/keyRings/<key
ring>-keys/cryptoKeys/<key name>/cryptoKeyVersions/<key version>"
algo = "SHA512"
```

Require attestors to sign before deployment

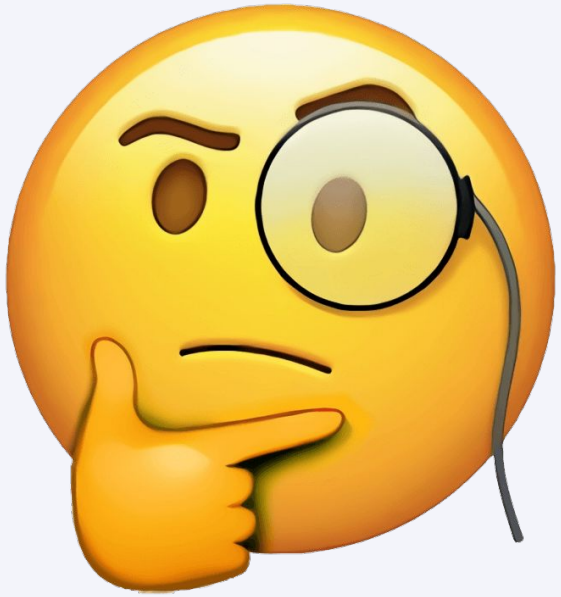
```
name: projects/compliant-project/policy
globalPolicyEvaluationMode: ENABLE
defaultAdmissionRule:
  evaluationMode: REQUIRE_ATTESTATION
  enforcementMode: ENFORCED_BLOCK_AND_AUDIT_LOG
  requireAttestationsBy:
  - projects/compliant-project/attestors/snakeoil
```

Snakeoil flow



See that was fun! Let’s have some more fun

PCI DSS 6.3: Develop internal and external software applications including web-based administrative access to applications in accordance with PCI DSS and based on industry best practices.



Trust, but verify

Test Name	Description
`diy`	Can the image be downloaded from our container registry?
`nobody`	Was the image built to run as a user who is not root?
`snakeoil`	Is the image free of known security issues?
`provenance`	Was the image built by us or a trusted system?
`approved`	Did the source code for the image pass all required checks in the code repository?
`is <org_name>`	Did the source for this image come from the passed organization (for example, is_shopify)?

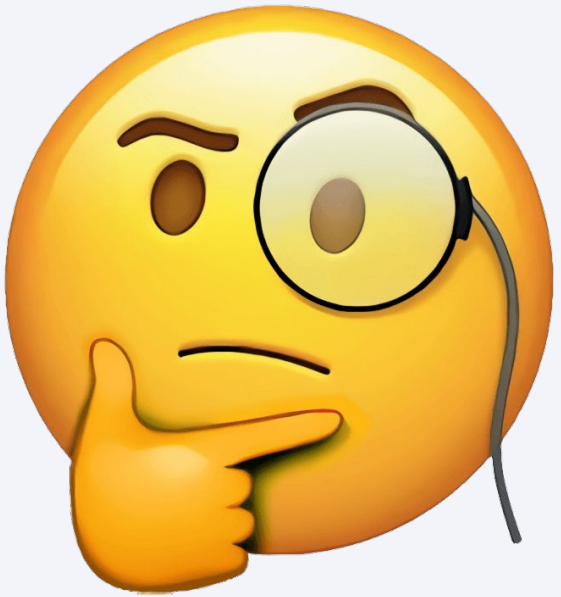
Voucher DIY and Provenance policy

```
[checks]
diy = true
provenance = true

valid_repos = [
    "gcr.io/internal-company-images/",
    "us-docker.pkg.dev/external-company-images/compliance/"
]

trusted_builder_identities = [
    "anners@shopify.com",
    "k8sbuilder.gserviceaccount.com"
]
trusted_projects = [
    "compliance-images"
]
```

Require attestors to sign before deployment

```
name: projects/compliant-project/policy
globalPolicyEvaluationMode: ENABLE
defaultAdmissionRule:
  evaluationMode: REQUIRE_ATTESTATION
  enforcementMode: ENFORCED_BLOCK_AND_AUDIT_LOG
  requireAttestationsBy:
  - projects/compliant-project/attestors/diy
  - projects/compliant-project/attestors/provenance
```

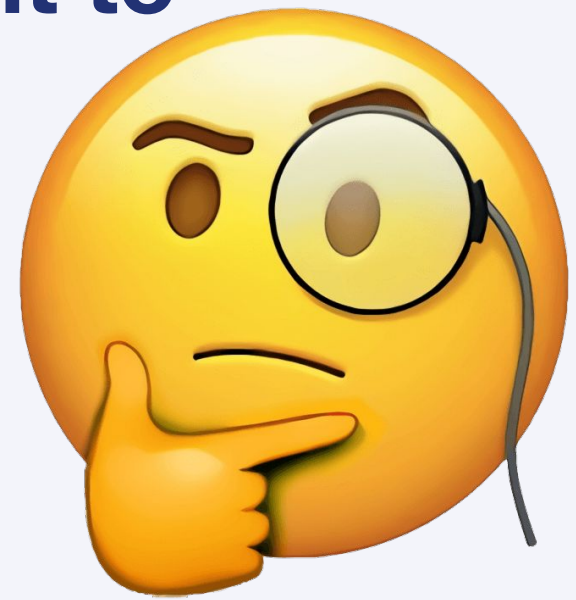
PCI DSS 7.1.2: Restrict access to privileged user IDs to least privileges necessary to perform job responsibilities.



Only Unicorns

Test Name	Description
`diy`	Can the image be downloaded from our container registry?
`nobody`	Was the image built to run as a user who is not root?
`snakeoil`	Is the image free of known security issues?
`provenance`	Was the image built by us or a trusted system?
`approved`	Did the source code for the image pass all required checks in the code repository?
`is <org_name>`	Did the source for this image come from the passed organization (for example, is_shopify)?

SOC 2: Continuous integration tests that cover functionality and security run on changes to relevant applications, prior to deployment to production.



Approved

Test Name	Description
`diy`	Can the image be downloaded from our container registry?
`nobody`	Was the image built to run as a user who is not root?
`snakeoil`	Is the image free of known security issues?
`provenance`	Was the image built by us or a trusted system?
`approved`	Did the source code for the image pass all required checks in the code repository?
`is <org_name>`	Did the source for this image come from the passed organization (for example, is_shopify)?

Let’s have fun together!

Contribute more checks to Voucher

https://github.com/grafeas/voucher/

```go
package examplecheck

import (
        voucher "github.com/grafeas/voucher/v2"
)

// check is a voucher.Check that holds our examplecheck test.
type check struct {
}

// Check if the image described by ImageData is good enough for our purposes.
func (c *check) Check(i voucher.ImageData) (bool, error) {
    ok := isImageGood(i)
        return ok, nil
}

func init() {
        voucher.RegisterCheckFactory("examplecheck", func() voucher.Check {
                return new(check)
        })
}
```

Google & Continuous Compliance

Common asks from our customers

Scale through automation	Enable Continuous Assessment	Decrease compliance paperwork
How to reduce the amount of time required to have a new workload go into production	*How to implement Continuous Compliance and Assurance?*	*How to reduce audit fatigue? How to reduce the total cost of ownership of controls*

This is where OSCAL helps!

How does OSCAL help

OSCAL is not a tool, but it enables tools to share data

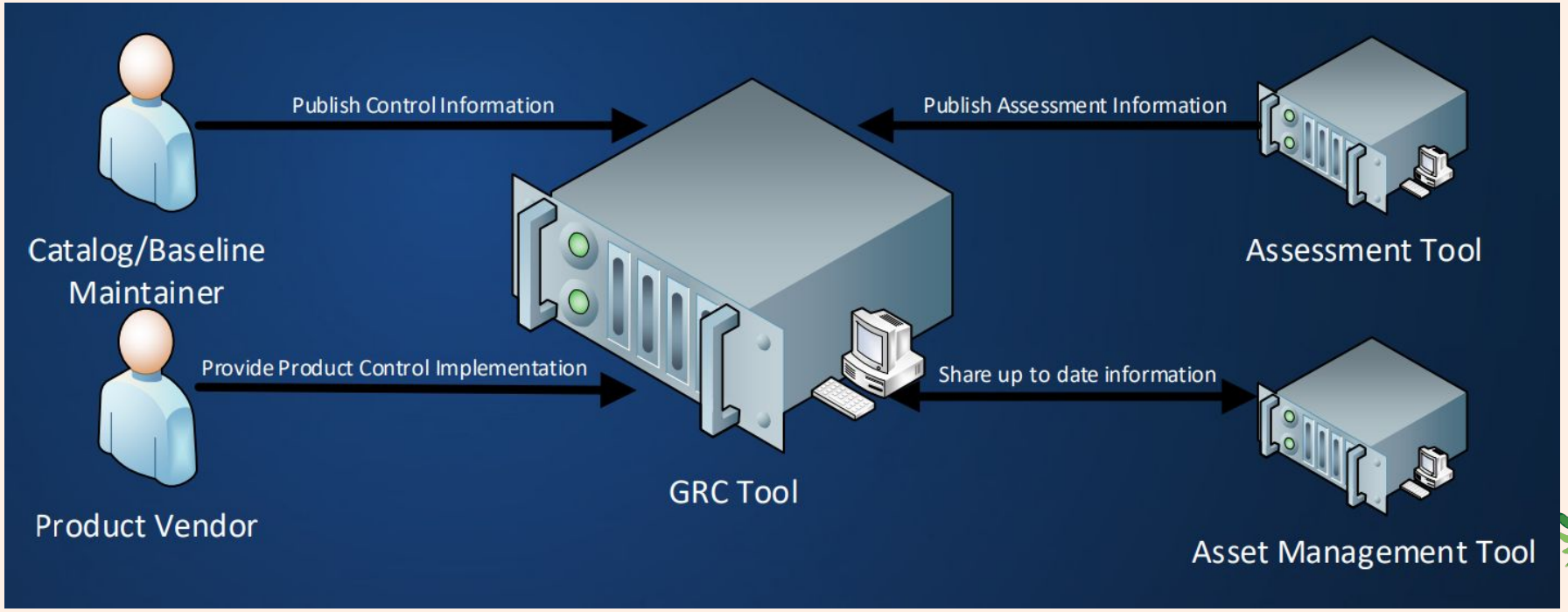
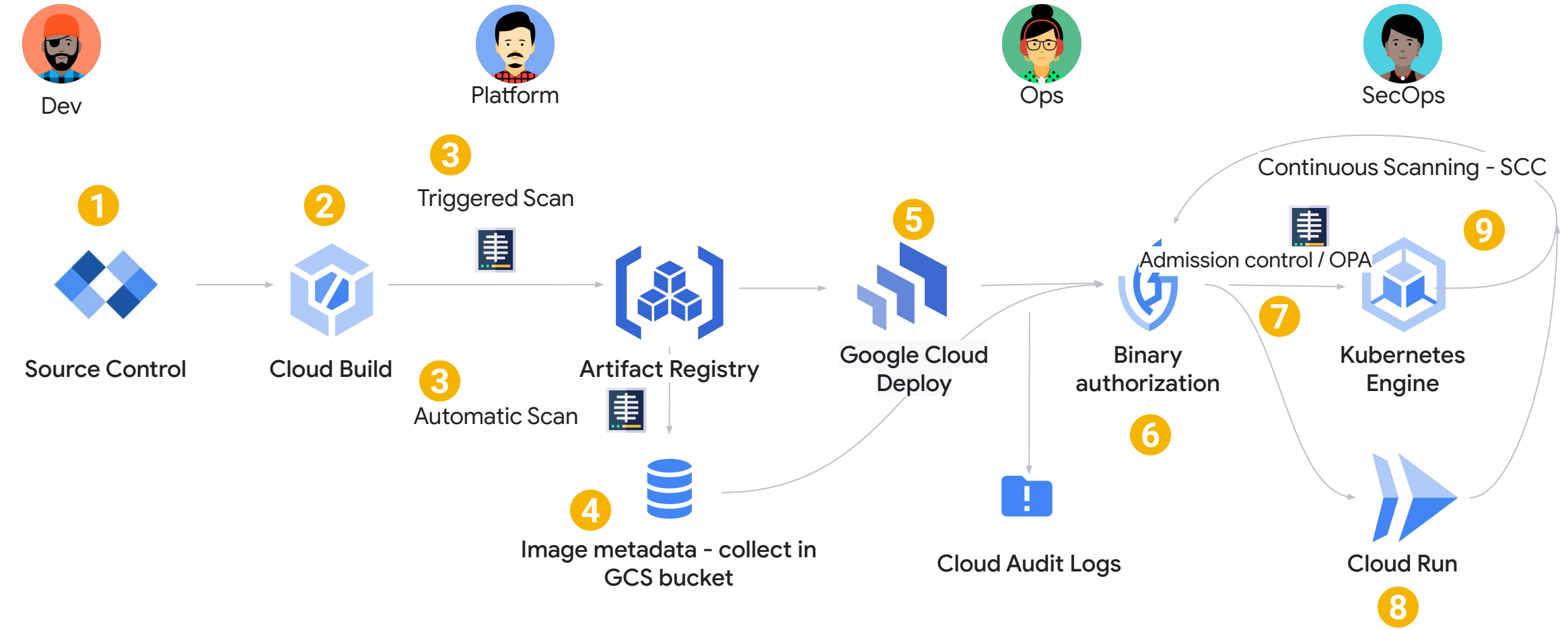


Image source: 2nd OSCAL Workshop Presentation

How will OSCAL work with a CI/CD pipeline

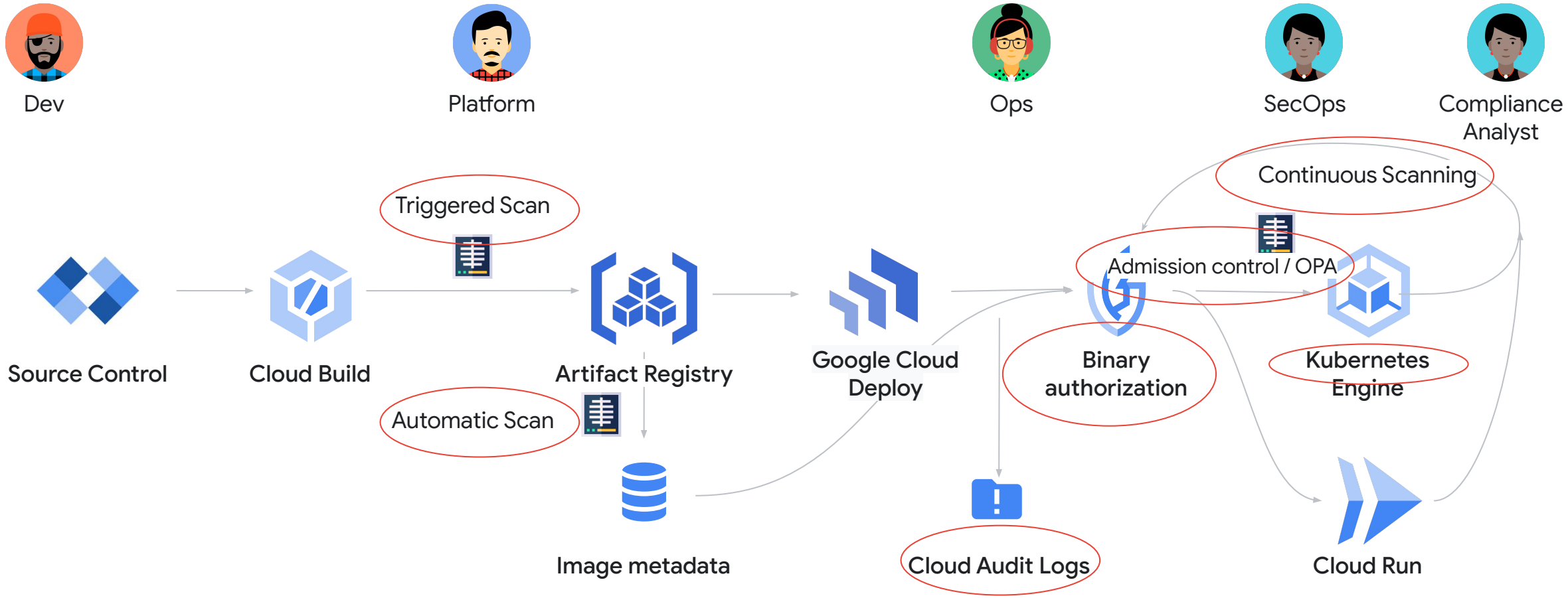
Proprietary + Confidential



Source: Secure Software Supply Chain Blueprint

Proprietary + Confidential

Potential OSCAL hooks…



In order for OSCAL to be effective, it must also integrate with DevOps tools. Integrating with existing tooling, such as Kubernetes configuration files, helps to ensure that controls can be tested with minimal friction. Compliance Trestle has a demo that convers Kubernetes YAML to OSCAL. In addition to looking at configuration before it's deployed to cloud infrastructure, tools such as Chef Inspec can inspect the already deployed environment configuration to populate control catalogs that map back to compliance frameworks.

Call to Action - OSCAL and Kubernetes Policy Working Group

Projects being discussed:

- **Policy Report CRD (custom resource definition) to OSCAL Mappings**
  - Translate from Kubernetes policy reports to OSCAL
  - Automate via command line and PAP (Policy Administration Point) tools
- **Kubernetes Controls Catalog**
  - Mappings from security standards to Kubernetes configurations
- **Policy libraries for controls**

Questions?

Was this type 1, 2, or 3 fun?

Remember to fill out the survey