AWS USER GROUP
LJUBLJANA

The Hidden Risk in Mono-Repos

GitHub Actions, OIDC Limited Claim Validation

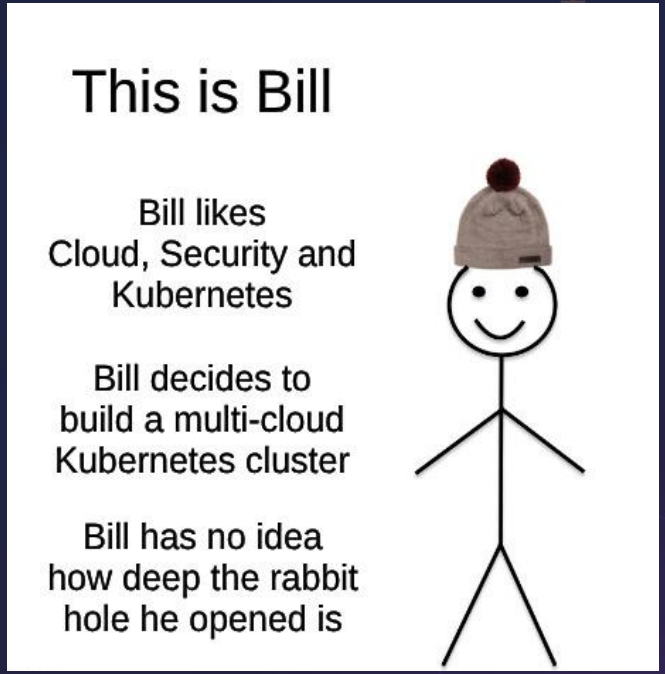
Timotej Petrovčič
SRE @ Aptira

Agenda

1. Motivation & Context
2. The problem
3. Proof of Concept
4. Challenges & Findings
5. Cloud Provider Comparison
6. Potential Attack Vectors
7. Remediations & Improvements

Motivation & Context

- Learn more about the Cloud and Kubernetes
- A bit of a Cybersecurity enthusiast
- “Want to try out multiple cloud providers”
- Need something interesting for a Master’s Thesis



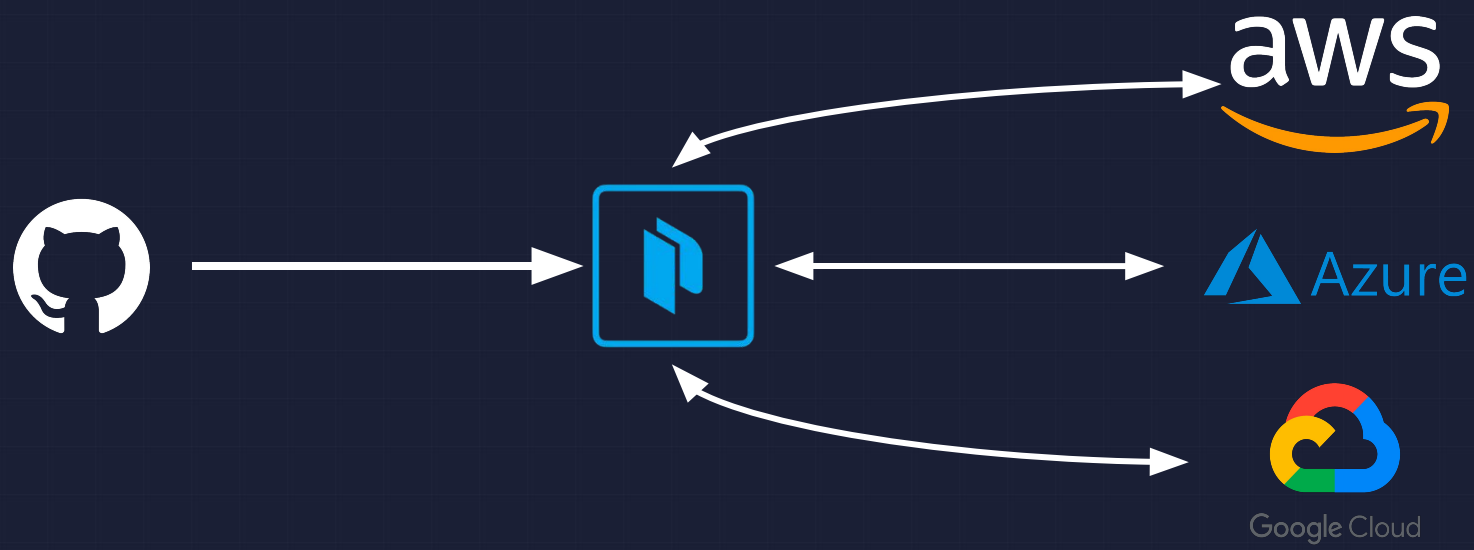
The Problem

- We want a secure Kubernetes distro:
  - Talos OS
    - No SSH
    - Mutual TLS authentication to Kube-API by default
- Budget Friendly Setup (Student at that time)
  - No managed Kubernetes
  - Bare-metal/VM based

The Problem

But how do we distribute OS images to multiple clouds and bare metal?

Golden Image Pipeline

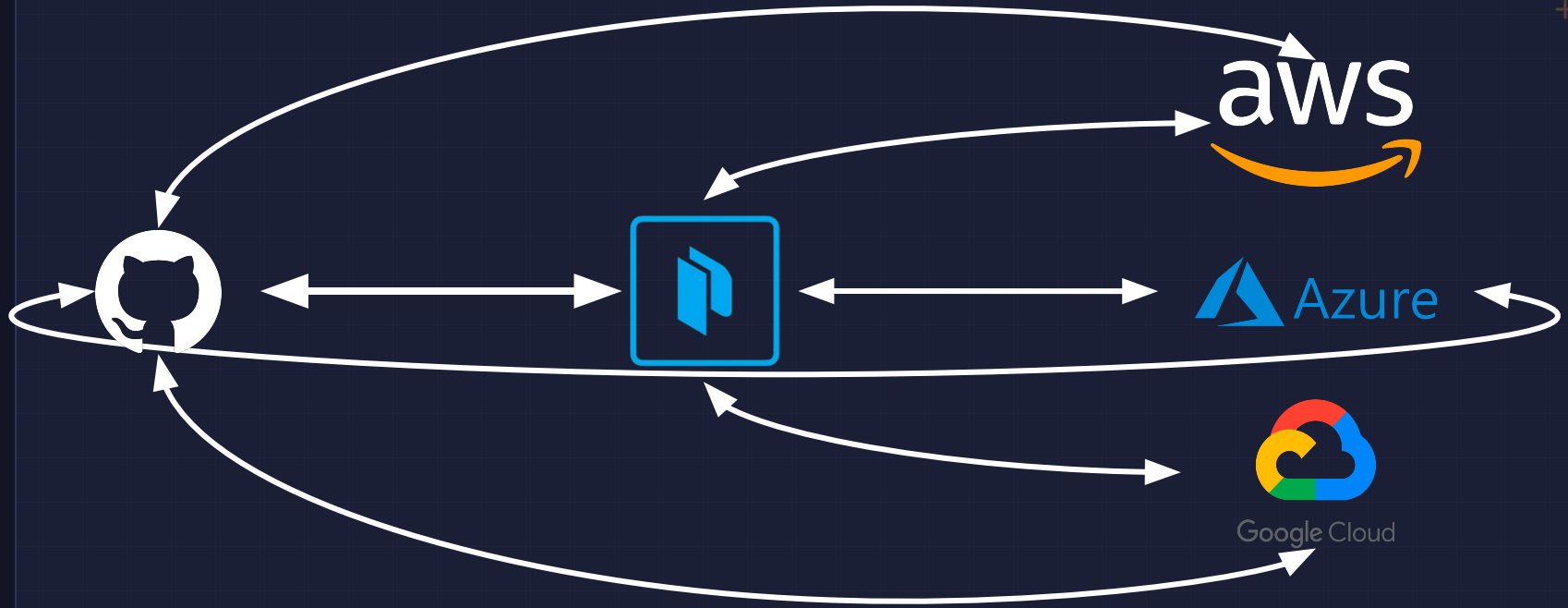


GitHub
Repo / Actions

HCP
Packer

Cloud
Providers

AWS USER GROUP
LJUBLJANA



GitHub
Repo / Actions

HCP
Packer

Cloud
Providers

PoC - GitHub Action Workflow

1. Checkout the repo (actions/checkout@v7)
2. Assume AWS Role via OIDC to access EC2/AMI APIs
3. Build the AMI using a composite action via Packer to AWS
    - Authenticate to HCP Packer via OIDC
    - Build the AMI using EBS Surrogate Pattern - Boot using generic Ubuntu and switch over the root mounted disk to Talos OS once it is built as a secondary volume of the EC2 instance
4. Promote the new HCP Packer Image iteration onto a Production Release Channel

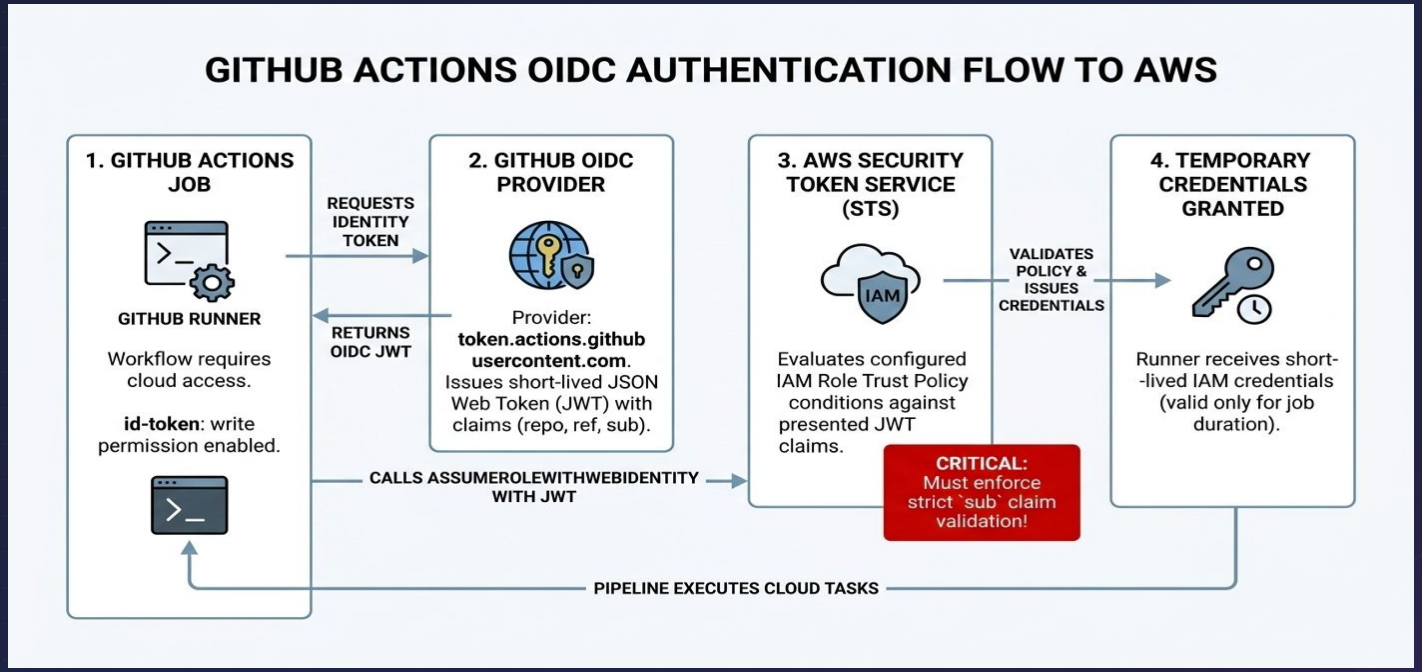
Challenges - GitHub Action Workflow

1. Checkout the repo (actions/checkout@v7) ✅
2. Assume AWS Role via OIDC to access EC2/AMI APIs ⚠️
3. Build the AMI using a composite action via Packer to AWS ✅ 🏔️
    - Authenticate to HCP Packer via OIDC
    - Build the AMI using EBS Surrogate Pattern - Boot using generic Ubuntu and switch over the root mounted disk to Talos OS once it is built as a secondary volume of the EC2 instance
4. Promote the new HCP Packer Image iteration onto a Production Release Channel ✅ 🌋

Findings - OIDC Basics

- **OpenID Connect (OIDC)** — Identity layer built on top of OAuth 2.0
- **Authentication without storing long-lived credentials**
- GitHub issues a **short-lived signed JWT** containing identity and workflow claims
- The cloud provider **validates the JWT** and maps it to a trusted IAM identity/role
- **Trust policies determine which claims are accepted** — e.g. repository, branch, environment, or workflow
- **Security depends on how precisely the cloud provider can scope that trust**

Findings - OIDC Authentication Flow



Findings - Limited Claim Validation

AWS USER GROUP
LJUBLJANA

Default | Amazon Cognito | Login with Amazon | Facebook | Gi

Default lists the standard OIDC claims and how they map to AWS STS condition context keys in AWS. You can use these keys to control access to a role. To do that, compare the **AWS STS condition keys** to the values in the **IdP JWT claim** column. Use this mapping if your IdP is not listed in the tab options.

GitHub Actions workflows and Google are some examples of IdPs that use the default implementation in their OIDC JWT ID token.

AWS STS condition key	IdP JWT claim	Available in session
amr	amr	Yes
aud	azp If no value is set for `azp`, the `aud` condition key maps to the `aud` claim.	Yes
email	email	No
oaud	aud	No
sub	sub	Yes

Login with Amazon | Facebook | GitHub | Google | CircleCI

This tab explains how GitHub Actions maps OIDC claims to AWS STS condition context keys in AWS. You can use these keys to control access to a role. To do that, compare the **AWS STS condition keys** to the values in the **IdP JWT claim** column.

AWS STS condition key	IdP JWT claim	Available in session
actor	actor	No
actor_id	actor_id	No
job_workflow_ref	job_workflow_ref	No
repository	repository	No
repository_id	repository_id	No
repository_owner_id	repository_owner_id	No
workflow	workflow	No
ref	ref	No
environment	environment	No
enterprise_id	enterprise_id	No

Cloud Provider Comparison - OIDC

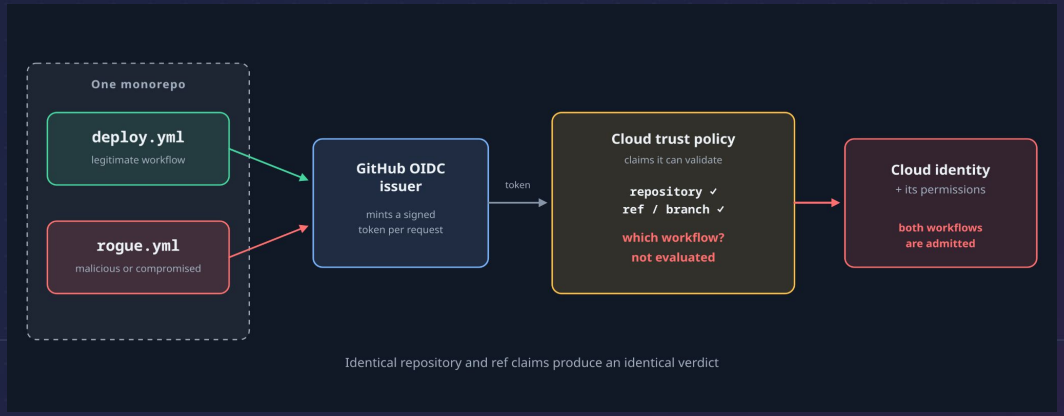
CLOUD PROVIDER	Validated Claims	Custom Claims	Monorepo Workflow Isolation
aws Alibaba Cloud Tencent Cloud	Selected GitHub claims (aud, sub)	❌	❌
Azure	Static Claims (aud, sub) Flexible Claims	⚠️ Limited	✅ Flexible FIC
Google Cloud	**Any Mapped OIDC Claim**	✅	✅ CEL
	Supported GitHub Claims	❌	✅ `workflow` / supported claims

Potential Attack Vectors

Rogue Workflow → Cloud Access

- A malicious or compromised workflow in the same monorepo requests an OIDC token.
- GitHub mints a valid token for it, exactly as it does for the deployment workflow.
- If the cloud trust policy validates only repository- and ref-level claims, the rogue workflow satisfies the same trust conditions as the legitimate one.

Impact: Unauthorized access to the cloud role and its permissions.

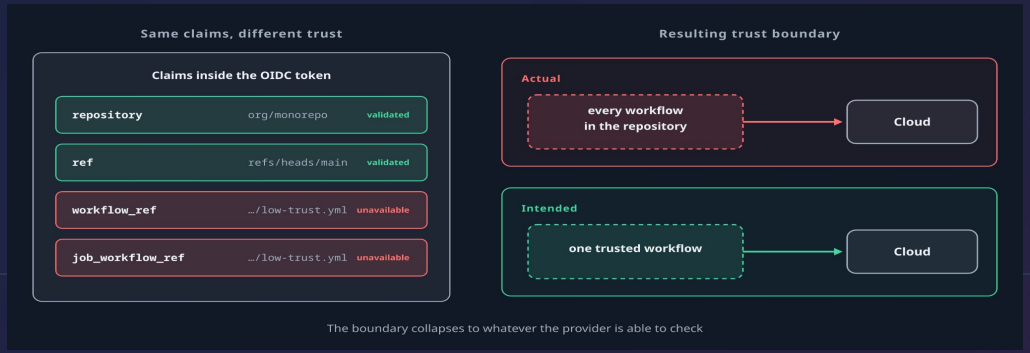


Potential Attack Vectors

Workflow-Level Trust Bypass

- Different workflows share the same repository and branch, but carry very different trust levels.
- Where the provider cannot validate workflow-specific claims such as `workflow_ref` or `job_workflow_ref`, those claims sit in the token unused.
- The effective boundary becomes Repository → Cloud rather than Trusted Workflow → Cloud.

Impact: A lower-trust workflow can reach a higher-trust cloud identity.



Potential Attack Vectors

OIDC Debug / Diagnostic Workflow

- A temporary debugging workflow granted `id-token: write` can request a real, signed OIDC token.
- The token is cryptographically valid; nothing in it marks the request as diagnostic.
- If that workflow falls inside the cloud provider's trust conditions, the debugging mechanism itself becomes an authentication path.

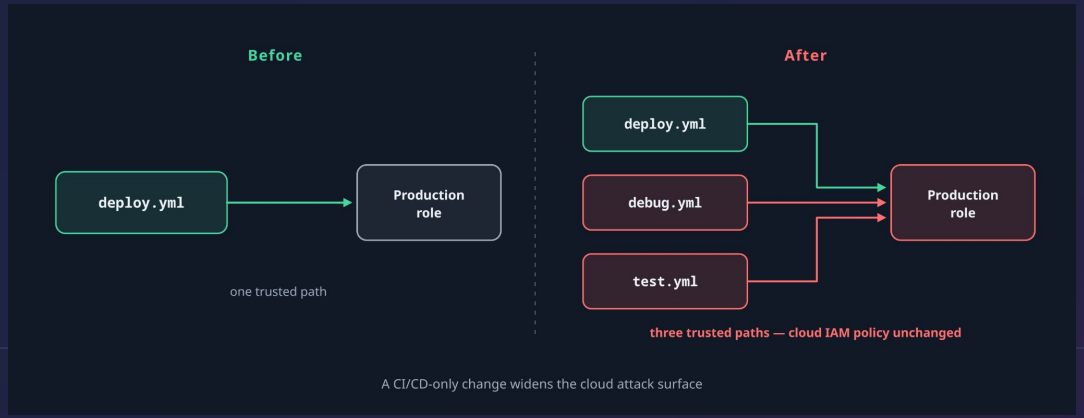
Impact: A seemingly harmless diagnostic workflow can expose production cloud access.

Potential Attack Vectors

Monorepo Trust-Boundary Expansion

- Adding a workflow can silently expand who the cloud trusts.
- One trusted path becomes three, with no change to the cloud IAM policy.
- The risk is sharpest with providers whose claim validation stops at the repository.

Impact: CI/CD changes can expand the cloud attack surface without any change to the cloud IAM policy.



Remediations & Improvements

Repository & Architecture

- **Separate release repositories** — Avoid using one monorepo as the release boundary for multiple cloud environments.
- **Isolate CI/CD trust domains** — Keep production deployment workflows separate from testing, development and experimental workflows.
- **Dedicated testing repositories** — Test GitHub Actions and third-party actions in isolated repositories before introducing them into production CI/CD.

OIDC & Identity

- **Validate workflow-level claims** — Use `job_workflow_ref` / `workflow_ref` where supported.
- **Restrict `id-token: write`** — Grant OIDC permissions only to workflows that require cloud access.
- **Least-privilege cloud roles** — Limit each deployment workflow to the minimum required permissions.

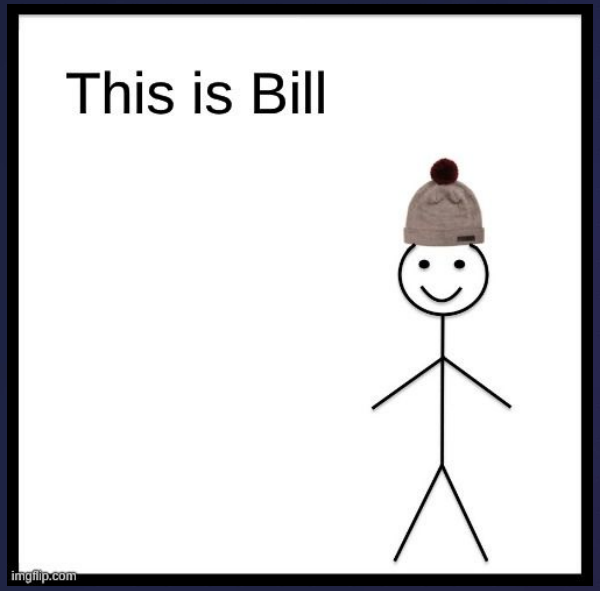
Remediations & Improvements

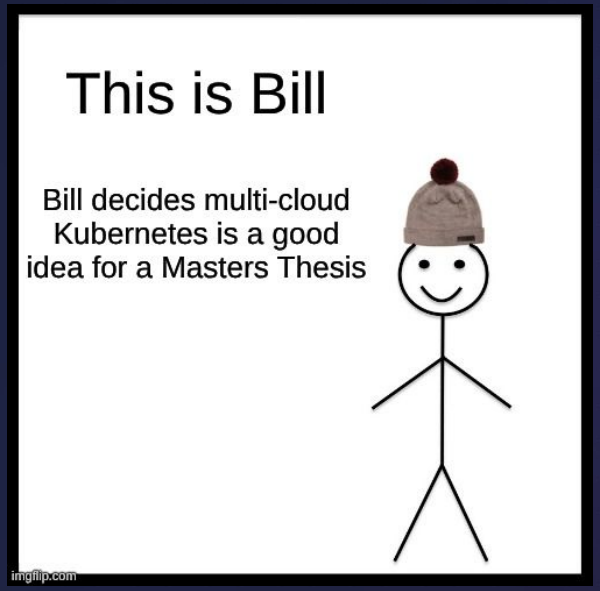
Runtime Visibility & Enforcement

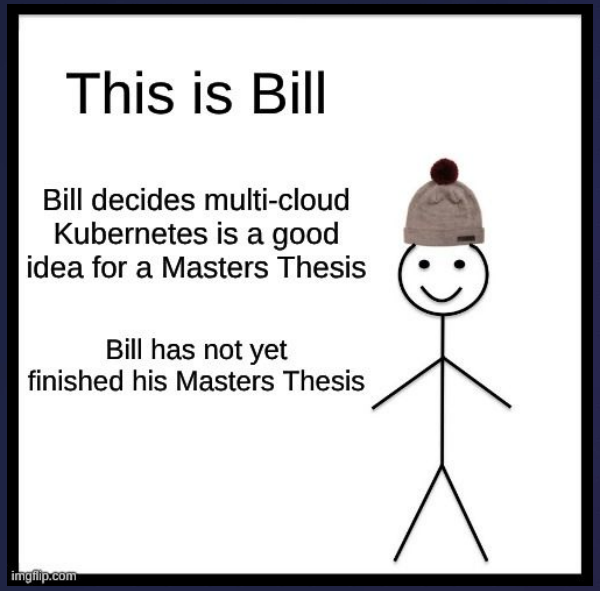
- **Observe GitHub Actions activity** — Use eBPF/Falco-style monitoring to see which processes, APIs and network destinations workflows access.
- **Detect & block unexpected behavior** — Alert or prevent unexpected cloud/API calls, credential access, or network activity from CI runners.

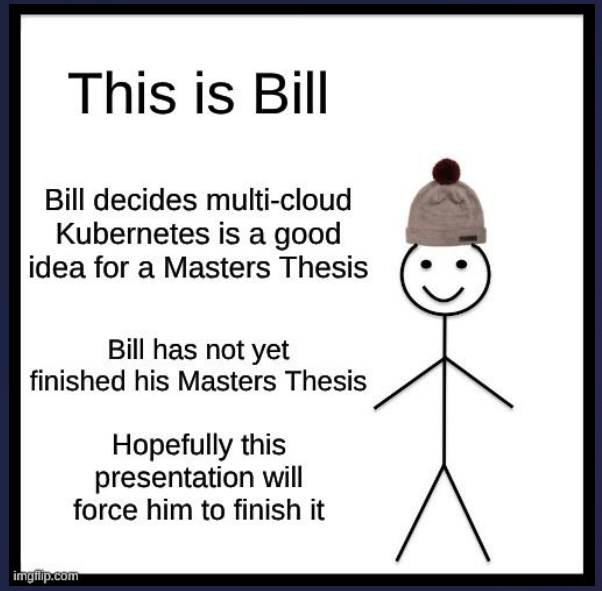
Key principle:

Separate trust domains first; enforce and monitor them at every layer.


This is Bill
imgflip.com


This is Bill
Bill decides multi-cloud Kubernetes is a good idea for a Masters Thesis
imgflip.com


This is Bill
Bill decides multi-cloud Kubernetes is a good idea for a Masters Thesis
Bill has not yet finished his Masters Thesis
imgflip.com



Don’t be like Bill, choose a simpler Masters Thesis

QUESTIONS

THANKS FOR LISTENING