

CLOUD IDENTITY
SUMMIT ‘26

Identity Security Track

Your CI/CD Pipeline Is Over-Privileged: Identity Risks in Modern Cloud Platforms

Mohamed Ali Mellah

Community Event by

Azure Meetup
BONN

Gold Sponsors adesso glueck kanja robopack TOKEN
Silver Sponsor e·on semperis TEAL
Bronze Sponsors MSC



About me

Mohamed Ali El Mellah

Cloud Security Engineer

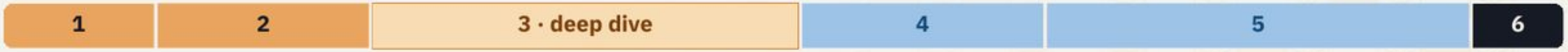
@mella7 · mella7.me · github.com/mella7

MellahMedAli

Link to my Portfolio - https://mella7.me

THE PLAN

From “what is it” to “what do I change on Monday”



1	**Why pipelines are the target**	Why attackers like pipelines so much
2	**Why we lose track of them**	Why identity programs, even good ones, lose track of them
3	**What goes wrong**	what actually goes wrong, authentication versus authorization
4	**How to audit them**	How to find these identities and audit them.
5	**How to fix them**	Controls you own, the deep dive, the whole picture, and response
6	**Wrap-up**	What you should do after this talk, and what all this means now that AI agents are showing up in pipelines

Part 1 of 6

Why pipelines are the target

Which identities attackers pick, and how often it works

1 · Why pipelines are the target

2 · Why we lose track of them

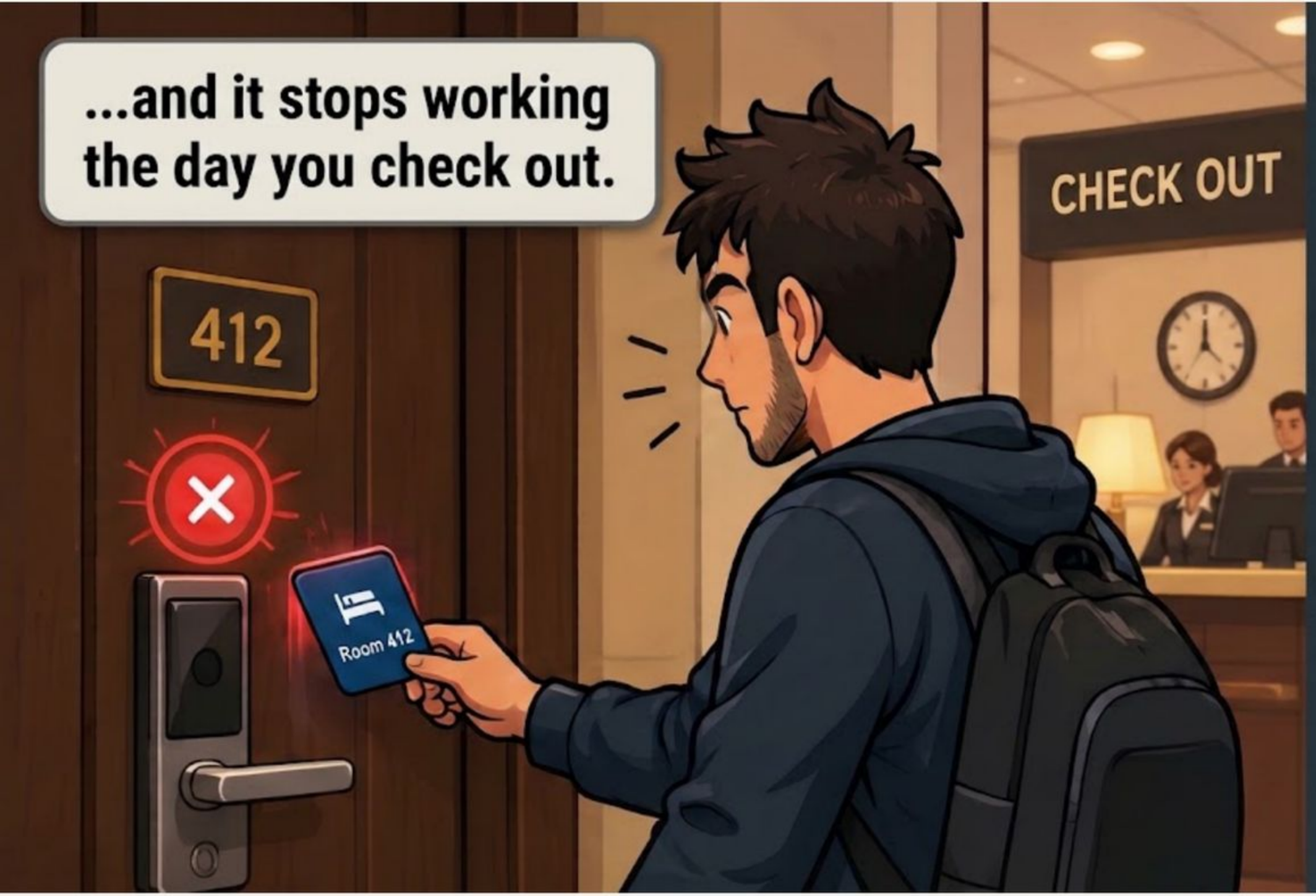
3 · What goes wrong

4 · How to audit them

5 · How to fix them

6 · Wrap-up


When you check into a hotel, your key card opens one room...
GRAND HOTEL
Room 412
PRINCIPLE OF LEAST PRIVILEGE
✓ Specific credentials granted for one-time task
✓ Automatic revocation upon completion
✓ Minimizes security risks



Credential Revocation

- Access terminates automatically on check-out date.
- Check-out process is seamless and automated.
- Ensures room security and guest privacy.

Now imagine the hotel gave you a card that opens every room, the kitchen, and the safe, and it never expires.

MASTER KEY



ALL ROOMS



KITCHEN



SAFE

NEVER EXPIRES

That's what most pipeline credentials look like.





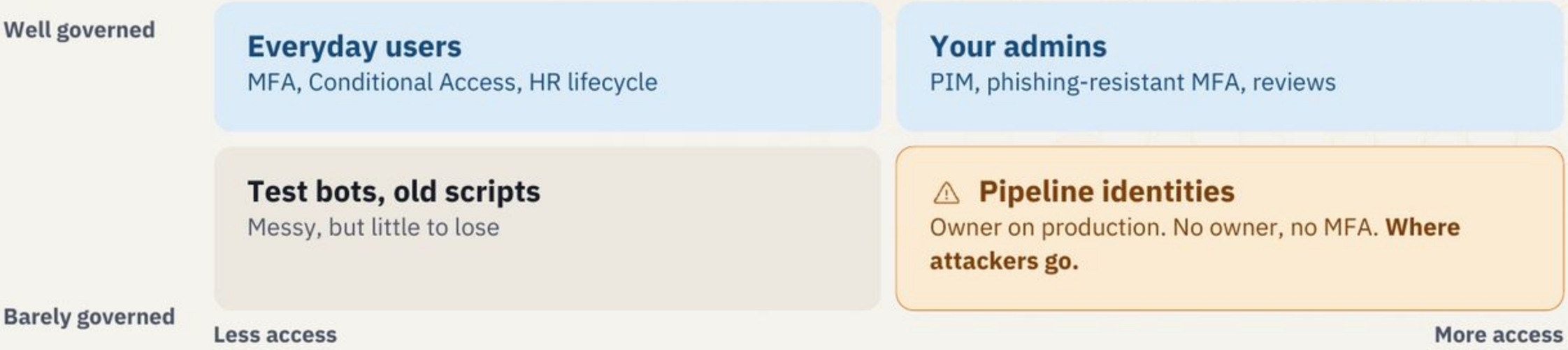
Credential Feature List

Credential Properties:

- ✔ Access to everything
- ✔ Long-lived
- ✔ No expiration
- ✔ No limits

WHICH IDENTITIES GET TARGETED

Attackers aim at the most access with the least governance



WHAT WE MEAN BY NON-HUMAN IDENTITY

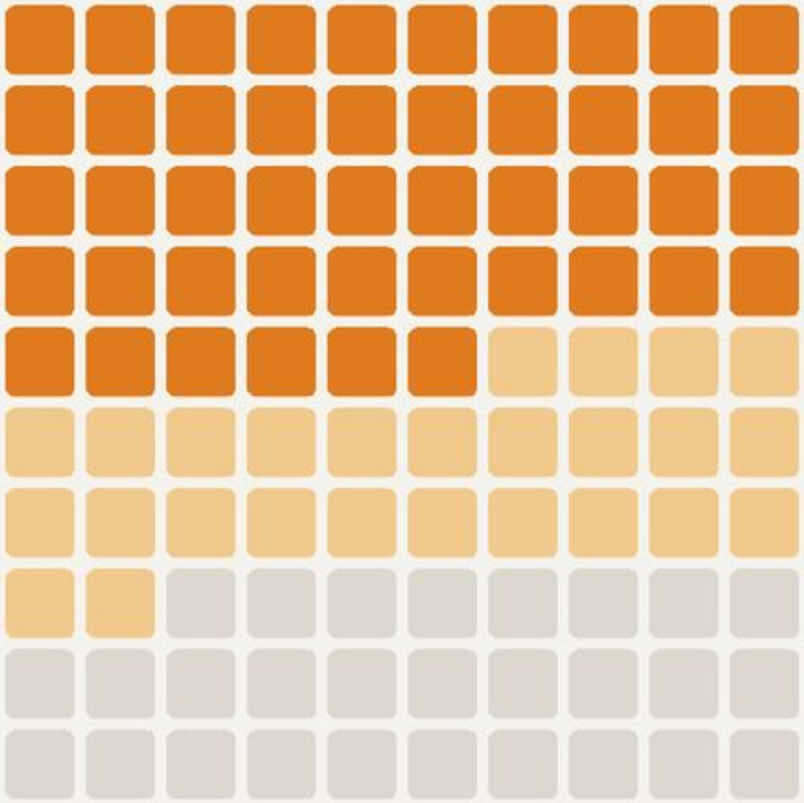
One pipeline, a whole family of them

Identities it uses	How it authenticates	Tools that should see it
Service principal for deploys	Client secret	Entra app registrations
Managed identity on a runner	Certificate	Service principal sign-in logs
GitHub App or job token	Federated OIDC token	Key Vault
A developer’s personal token	Personal access token string	Secret scanning

Created by developers, DevOps and IT admins, **each in a different system.**



We automated deployment. We also automated access to our entire infrastructure.



OUT OF EVERY 100 ORGANIZATIONS

Nearly three in four

46 know they had a breach involving a non-human identity in the past year

26 suspect one, but can’t confirm it

28 report neither

And about 20 non-human identities for every human one

Source: ESG survey of North American organizations, with Oasis Security, 2024. Self-reported.

LEAKED SECRETS DON'T EXPIRE ON THEIR OWN

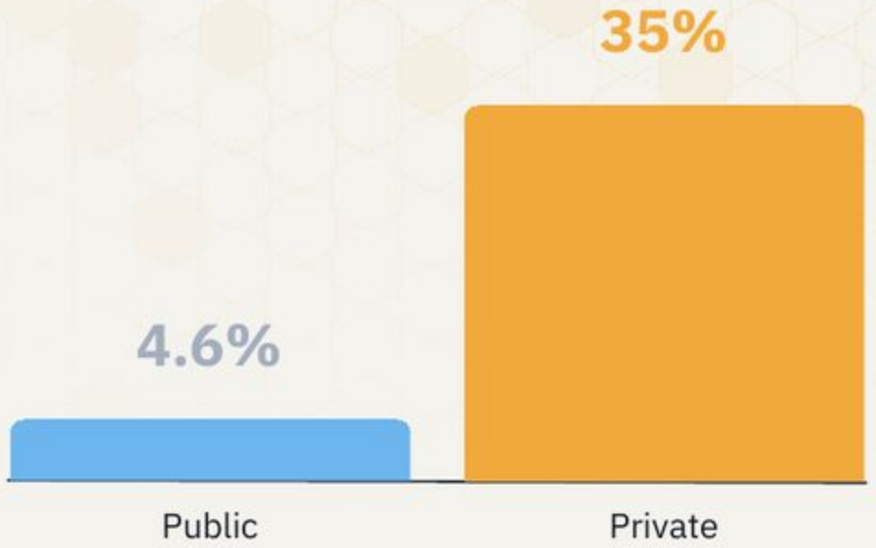
Most leaked secrets still work years later

Secrets leaked in 2022, checked in 2025



23.8M new secrets leaked on public GitHub in 2024, up 25% on the year before

Repositories containing a secret



Source: GitGuardian, The State of Secrets Sprawl 2025 (data from 2024)

Part 2 of 6

Why we lose track of them

An identity program built for people misses machines

✓ 1 · Why pipelines are the target

2 · Why we lose track of them

3 · What goes wrong

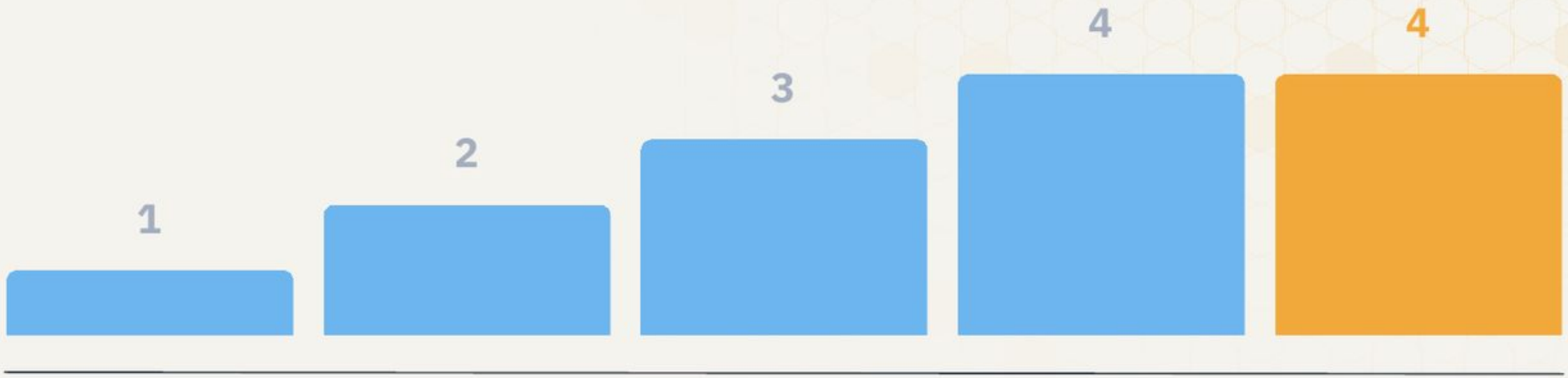
4 · How to audit them

5 · How to fix them

6 · Wrap-up

THE UNGOVERNED JUNGLE

How one repository grows four identities



Day 1
Deploy service principal created

Month 3
Personal token for the package registry

Month 8
Contractor adds a second secret

Year 2
Staging gets its own app registration

Year 3
The creator leaves. All four stay.

WHY THE IAM STACK FALLS SHORT

Two lifecycles, one of them on autopilot

	PROVISION	OWNER	CONTROLS	ACCESS	DEPROVISION
A person Mostly automated	HR-driven, automatic	Employee and a manager	MFA, Conditional Access, PIM	Role-based, reviewed	Disabled on the last day
A pipeline Mostly manual	By hand, when someone needs it	The creator, until they leave	No MFA. Secret valid two years.	Owner, just in case	Never cleaned up

Pipelines need a lifecycle of their own, **not an exception in yours.**

NO SOURCE OF TRUTH

There is no HR system for machines

A PERSON

HR record

↓

Entra ID

One source of truth

GITHUB-DEPLOY-PROD

Entra ID
The app and its roles

GitHub
The secret, who can edit the workflow

Key Vault
Maybe a copy. Maybe not.

A wiki page
Last edited in 2023

Someone's head
They left last year

THE HIDDEN IDENTITY IN MANY PIPELINES

A person’s token doing a machine’s job



The day j.weber leaves
Remove the account and the pipeline breaks. Keep the account alive, and a leaver still holds a key.

The fix
A GitHub App or the job’s own token, scoped to one repo. If a personal token is unavoidable: fine-grained, with an expiry.

CLOUDFLARE · 21 MARCH 2025

No attacker. Just a credential rotation.

Step 1
New credentials created

→

Step 2 · mistake
Deployed to a non-production environment

→

Step 3 · mistake
Old credentials deleted

→

Step 4
Production can't authenticate

67 min
of errors on R2 storage

100%
of write operations failed

~35%
of read operations failed

Rotation is fragile, so teams skip it, so secrets live for years. **For pipelines, remove the secret instead.**

Source: "Cloudflare incident on March 21, 2025", Cloudflare blog.

Part 3 of 6 · deep dive

What goes wrong

The two questions, one concrete attack chain, then the OWASP NHI Top 10

✓ 1 · Why pipelines are the target

✓ 2 · Why we lose track of them

3 · What goes wrong

4 · How to audit them

5 · How to fix them

6 · Wrap-up

THE CORE IDEA · KEEP THESE TWO APART

A pipeline identity has two security questions

TRUST · AUTHENTICATION

Which workflow may obtain credentials?

Decided by the trust / federation config: issuer, audience, and the repo, branch or environment in the token's subject.

then

AUTHORIZATION

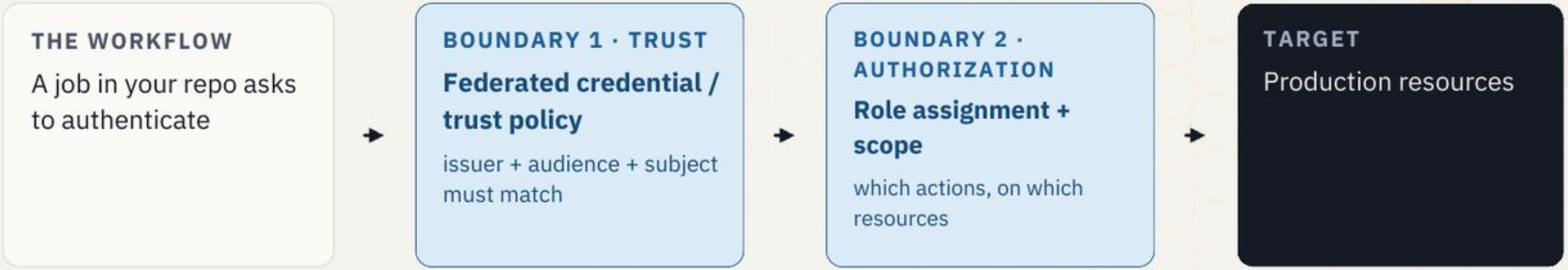
What can the identity then do?

Decided by the permission / role assignment and its scope: which actions, on which resources.

⚠ OIDC federation fixes the first question, not the second. A compromised job with a valid token still does whatever its permissions allow.

THE TWO BOUNDARIES, CONCRETELY

Where trust ends and authorization begins



	Boundary 1 · Trust	Boundary 2 · Authorization
Question	Who may get a credential?	What may that credential do?
In Entra	Federated credential on the app	Role assignment + its scope
In AWS	Role trust policy (AssumeRole)	Permission policy on the role
Gets it wrong by	A subject that matches too much	Owner / wildcard instead of least privilege

Two independent controls. Tightening one does not tighten the other.

PATTERN 01

Static secrets

You think of the settings page. Attackers think of the other seven places.

Eight places a secret hides

Repo & org secrets
the page you already know

CI variables
often unprotected

.env in image layers
baked and shipped

Terraform state
plaintext by default

Build logs
echoed, then stored

Runner caches
and artifacts

base64 kubeconfigs
"encoded" is not "secured"

Git history
committed, then "deleted"

Masking is string matching

```
$ echo $AWS_SECRET_ACCESS_KEY
***
# so encode it first
$ echo $AWS_SECRET_ACCESS_KEY | base64
d2pBbFhVdG5Ga2l2bXc9…  ← not masked
```

Any step can read the environment, files on disk, and — as tj-actions showed — the runner's process memory. Masking hides the exact string in logs. Nothing more.

PATTERN 02

Broad service accounts

One deployer for everything — and "not admin" that's admin with one extra step.

You don't need admin to become admin

The audit says "not admin", so the review passes. These are admin with one extra step.

AWS

`iam:PassRole` + launch a Lambda or EC2 with a stronger role

`iam:CreatePolicyVersion`

GCP

`iam.serviceAccounts.actAs`

`getAccessToken` on a stronger SA

Kubernetes

create pods → mount any Secret in the namespace

bind / escalate on roles

This is where most "least privilege" reviews quietly fail.

PATTERN 03

Unrestricted workflows

Who can trigger it, and every line of code it pulls in to run.

Every uses: line is production access

```
- uses: some-org/action@v4    # a tag: can be moved
- uses: some-org/action@8f4b1c2 # a SHA: can't
- run: curl -s https://x.sh | bash # trust me
```

Tags are mutable — tj-actions proved it. curl | bash is how Codecov reached customers. Caches get poisoned too. A third-party action runs with your prod identity, so review it like a production dependency.

OWASP NON-HUMAN IDENTITIES TOP 10 · 2025

One ordinary pipeline hits most of this list

NHI1 · hit
Improper offboarding
Creator left, identity stayed

NHI2 · hit
Secret leakage
Secret in a build log

NHI3 · hit
Vulnerable third-party NHI
A compromised build action

NHI4
Insecure authentication
Weak or legacy auth flows

NHI5 · hit
Overprivileged NHI
Owner plus Graph write

NHI6 · hit
Insecure cloud deployment config
Static secret in CI/CD

NHI7 · hit
Long-lived secrets
Valid for two years

NHI8 · hit
Environment isolation
One identity for dev and prod

NHI9 · hit
NHI reuse
Same secret in staging

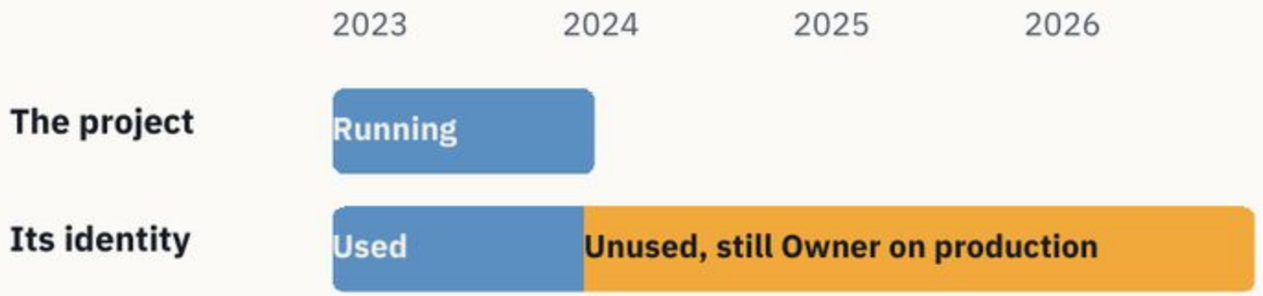
NHI10
Human use of NHI
Engineers using its secret by hand

Source: OWASP Non-Human Identities Top 10, 2025 edition

OWASP NHI1 · 1 OF 10

Improper offboarding

Identities that outlive their purpose. Still active, still privileged.



A zombie identity: nobody uses it, but anyone who finds its secret can.

IN A PIPELINE

The project ends or the creator leaves. The app and its secret stay.

CHECK

Service principals with no sign-in for 90 days, or no owner

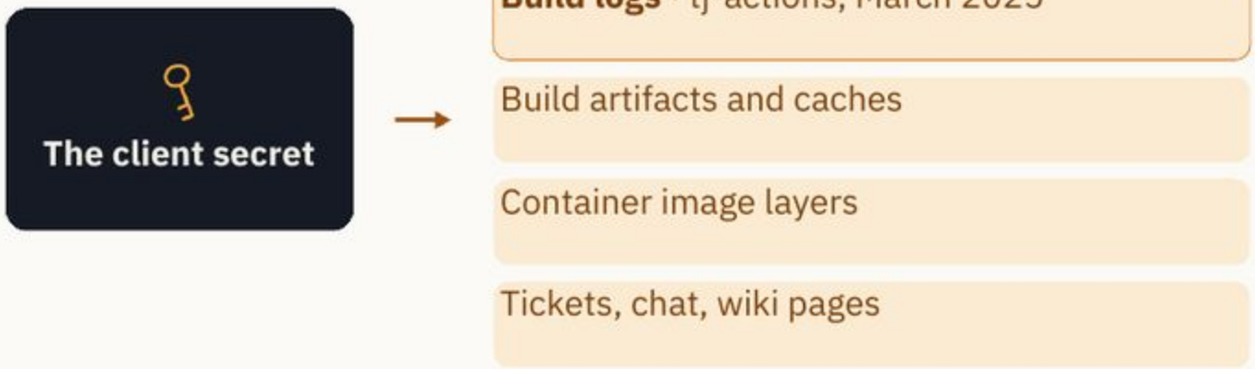
FIX

Retire it with the project. Put identities in the leaver process.

OWASP NHI2 · 2 OF 10

Secret leakage

A credential ends up somewhere it shouldn’t be.



IN A PIPELINE

The secret sits in memory during every run, next to every tool in the job.

CHECK

Secret scanning on repos, images and logs, not just code

FIX

Leave nothing to leak: federation or a managed identity.

OWASP NHI3 · 3 OF 10

Vulnerable third-party NHI

Someone else’s software holds, or can reach, your credentials.



IN A PIPELINE

Every action you pull in runs inside the job, near its credentials.

CHECK

An inventory of third-party actions and integrations, and their reach

FIX

Pin by commit, allow-list actions, keep credentials out of build jobs.

OWASP Non-Human Identities Top 10, 2025 · NHI3:2025 Vulnerable Third-Party NHI · CircleCI security alert, 4 January 2023

OWASP NHI4 · 4 OF 10

Insecure authentication

Weak or legacy ways of proving identity, kept long after better ones exist.

STRONGER

Federated token or managed identity — **Where to be**

Certificate, short-lived, in a vault

Client secret — **Most pipelines**

Shared keys, such as storage account keys

WEAKER

IN A PIPELINE

A client secret, even though the platform supports federation.

CHECK

Apps with password credentials; resources still allowing shared keys

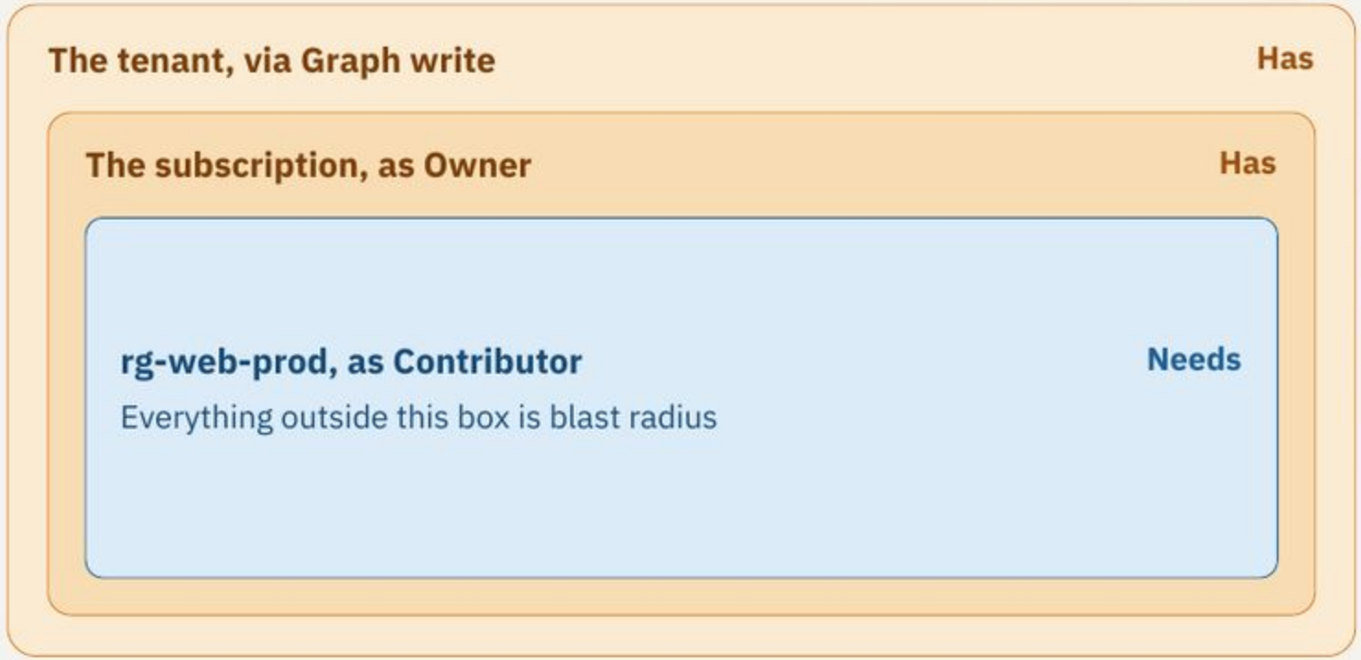
FIX

Move up the ladder. Disable shared-key access where Entra auth works.

OWASP NHI5 · 5 OF 10

Overprivileged NHI

More permission than the job needs, so a small compromise becomes a big one.



IN A PIPELINE

Owner on the subscription “to make it work”, plus a Graph write.

CHECK

Roles above resource-group scope; Graph application permissions

FIX

Contributor on one resource group. No Graph write in the deploy path.

OWASP NHI6 · 6 OF 10

Insecure cloud deployment config

How CI/CD signs in to the cloud: static secrets, or OIDC with loose conditions.

Static credential · risky

`secrets.AZURE_CLIENT_SECRET`

Leaks through config, repos or logs

Federation, loosely scoped · risky

`sub: repo:contoso/web:pull_request`

Any pull request, reviewed or not, can get a token

Federation, tightly scoped · good

`sub: repo:contoso/web:environment:production`

Only approved jobs in the protected environment

IN A PIPELINE

OWASP names CI/CD directly. This one is about your pipeline.

CHECK

Every federated credential subject. Broader than one environment?

FIX

Federate to a protected environment. Review it like a CA policy.

OWASP NHI7 · 7 OF 10

Long-lived secrets

Credentials with far-off or no expiry, which keep working long after a leak.

Toyota T-Connect: an access key in a public GitHub repository

Dec 2017 · published — **Sep 2022 · found**

Almost five years exposed

Nearly 300,000 customer email addresses were reachable.

> A federated token on the same scale: **about an hour**, too thin to draw

IN A PIPELINE

Client secrets made for one or two years, renewed without review.

CHECK

Expiry dates on every app credential. Longer than six months?

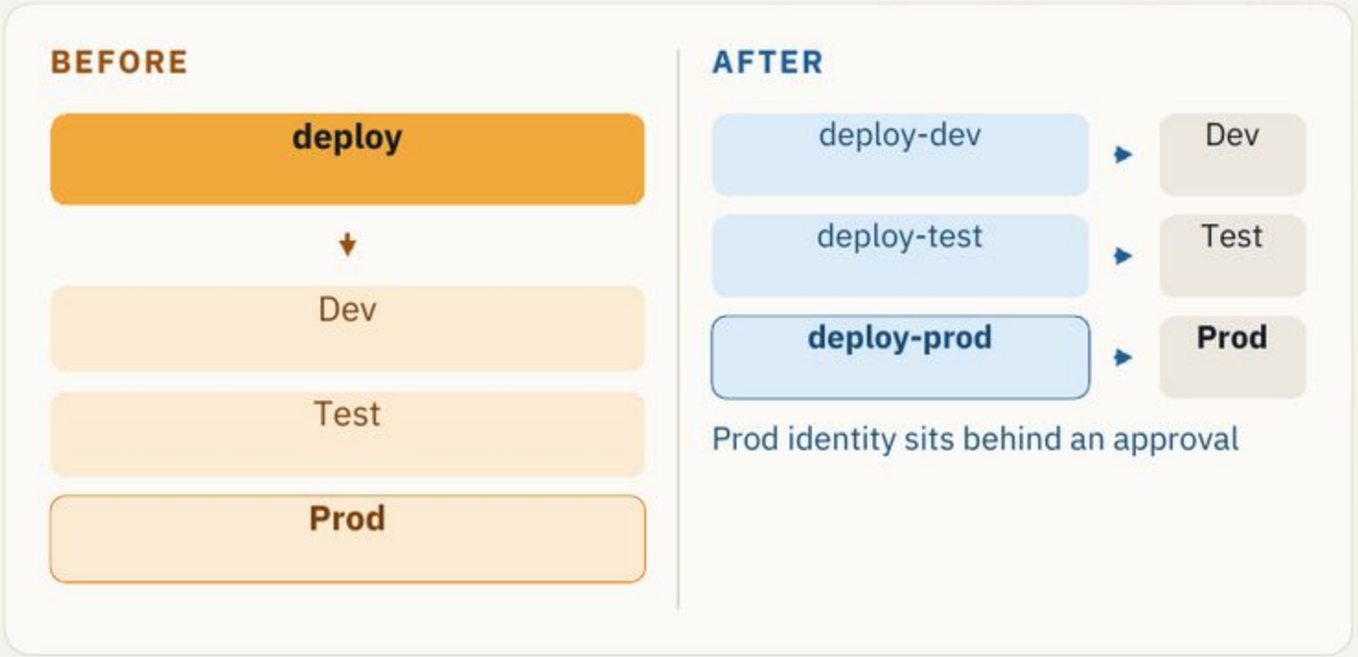
FIX

Short-lived tokens issued at runtime. No secret, no expiry to manage.

OWASP NHI8 · 8 OF 10

Environment isolation

The same identity used across development, test and production.



IN A PIPELINE

One identity deploys everywhere, so a test job holds production access.

CHECK

Service principals with roles in more than one environment

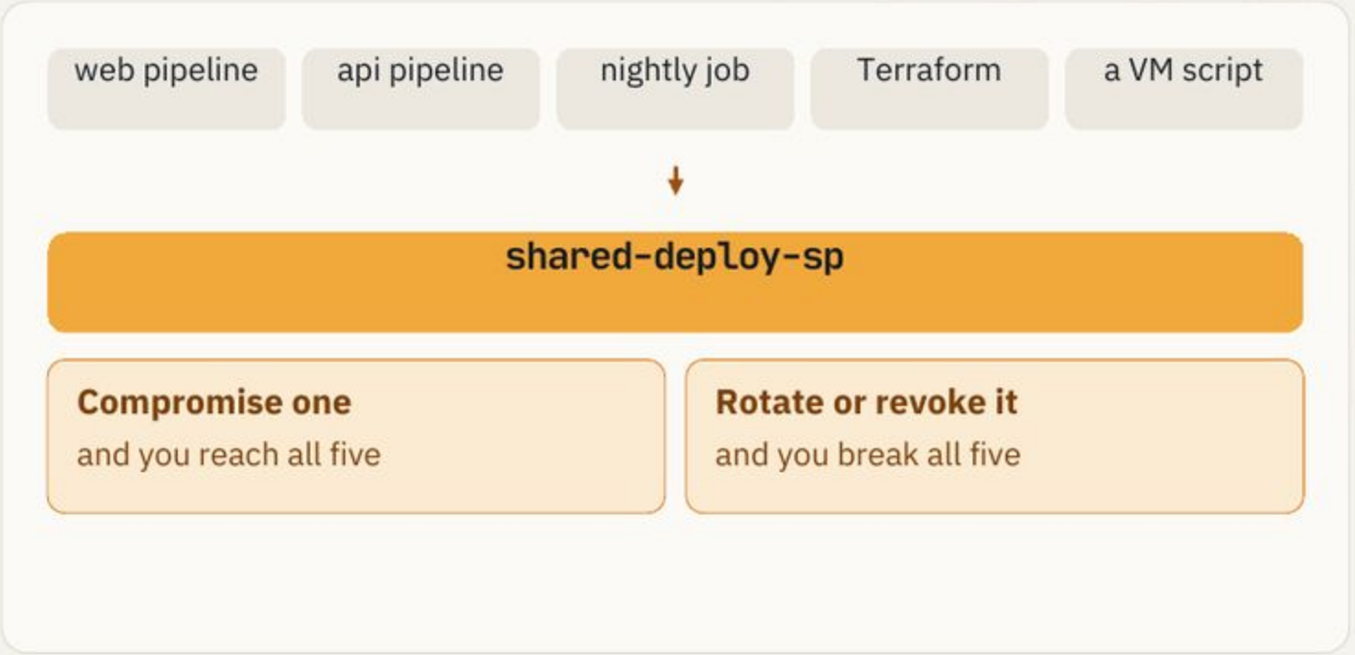
FIX

One identity per environment. Production’s is gated.

OWASP NHI9 · 9 OF 10

NHI reuse

One identity shared by several applications or pipelines.



IN A PIPELINE

A working identity gets copied into the next pipeline, then the next.

CHECK

One service principal signing in from many repos or hosts

FIX

One identity per workload. Creating one is cheap; sharing isn’t.

OWASP NHI10 · 10 OF 10

Human use of NHI

People using a machine identity’s credentials to do things by hand.

Pipeline runner	→	`deploy-prod`	Expected, from known runners
Engineer’s laptop	→	`deploy-prod`	**Skips MFA and PIM**

And the audit log says “deploy-prod”, not the person

A human acted with a machine’s rights, and the trail points at the machine.

IN A PIPELINE

Someone copies the deploy secret to fix production fast from a laptop.

CHECK

Service principal sign-ins from outside your runner network

FIX

No secret to copy. People use their own PIM-eligible accounts.

Part 4 of 6

How to audit them

Find them, then ask five questions

✓ 1 · Why pipelines are the target

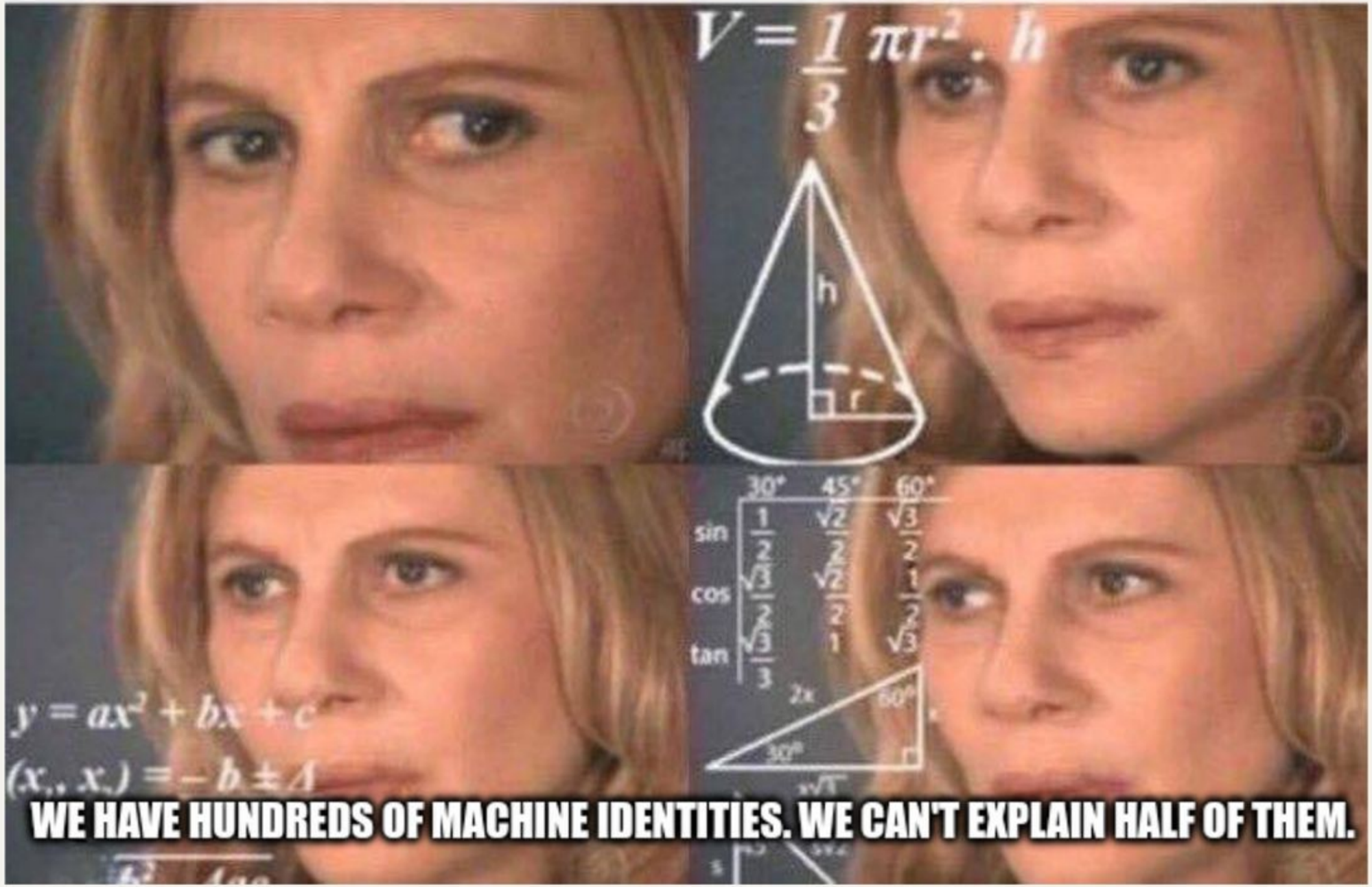
✓ 2 · Why we lose track of them

✓ 3 · What goes wrong

4 · How to audit them

5 · How to fix them

6 · Wrap-up


$V = \frac{1}{3}\pi r^2 . h$
h
r
30°
45°
60°
sin
cos
tan
2x
60°
30°
$y = ax^2 + bx + c$
$(x_1, x_2) = -b \pm \Delta$
WE HAVE HUNDREDS OF MACHINE IDENTITIES. WE CAN'T EXPLAIN HALF OF THEM.

BEFORE THE FIVE W'S: FIND THEM

Narrow hundreds of identities to a short list

All service principals in the tenant

...holding a secret, certificate or federated credential

...with a role on production

...with Owner, or any Graph write

Start here

Azure Resource Graph · broad role assignments

```
authorizationresources
| where type == "microsoft.authorization/roleassignments"
| where properties.principalType == "ServicePrincipal"
| extend scope = tostring(properties.scope)
| where scope !contains "/resourceGroups/"
```

Service principals with a role on a whole subscription or management group

THINK LIKE AN INVESTIGATOR

Five questions for every pipeline identity

Who	What	When	Where	Why
owns it?	can it do?	is it active?	can it be used?	does it exist?

Any answer you can’t give is a task.

“Five W’s” framework: Adam Ochayon, Oasis Security, Identity Management Day 2025

W 1 OF 5 · OWNERSHIP

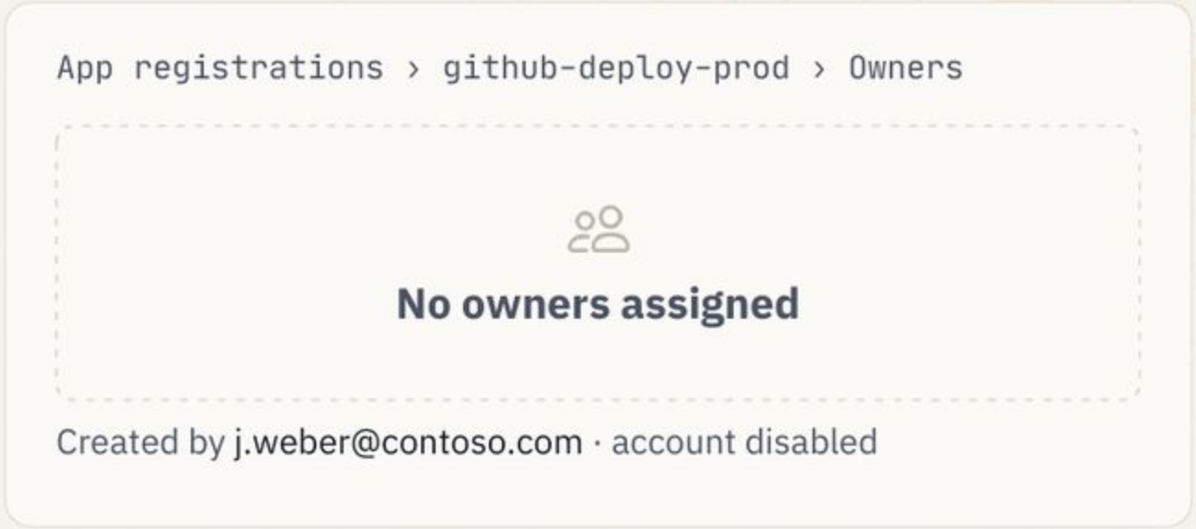
Who

If it misbehaves at 3:47 AM, who gets the call?

Who created it, and are they still here?

Who is responsible for it today?

Is that written down where security can find it?



Fix: a named person, not a team alias, with a review date

Five W’s framework: Adam Ochayon, Oasis Security · applied to a pipeline identity

W 2 OF 5 · PERMISSIONS

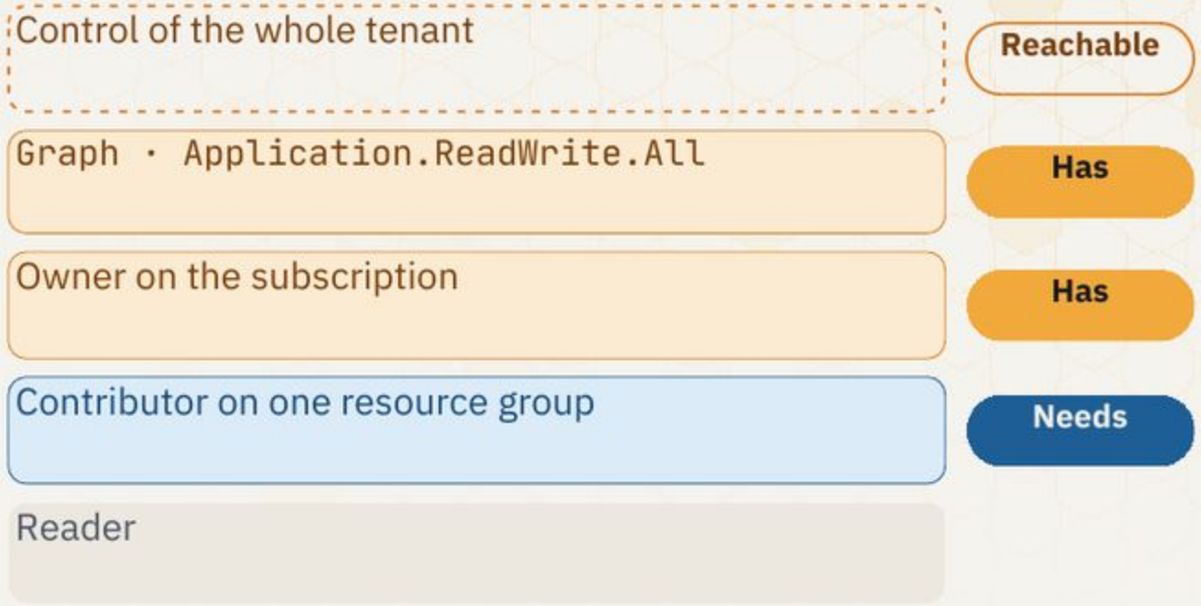
What

What can it do, and what does it actually need?

Which systems and data can it reach?

Does it hold admin or Graph write permissions?

Could it be used to escalate?



The gap between “needs” and “has” is your blast radius.

Five W’s framework: Adam Ochayon, Oasis Security · applied to a pipeline identity

W 3 OF 5 · ACTIVITY

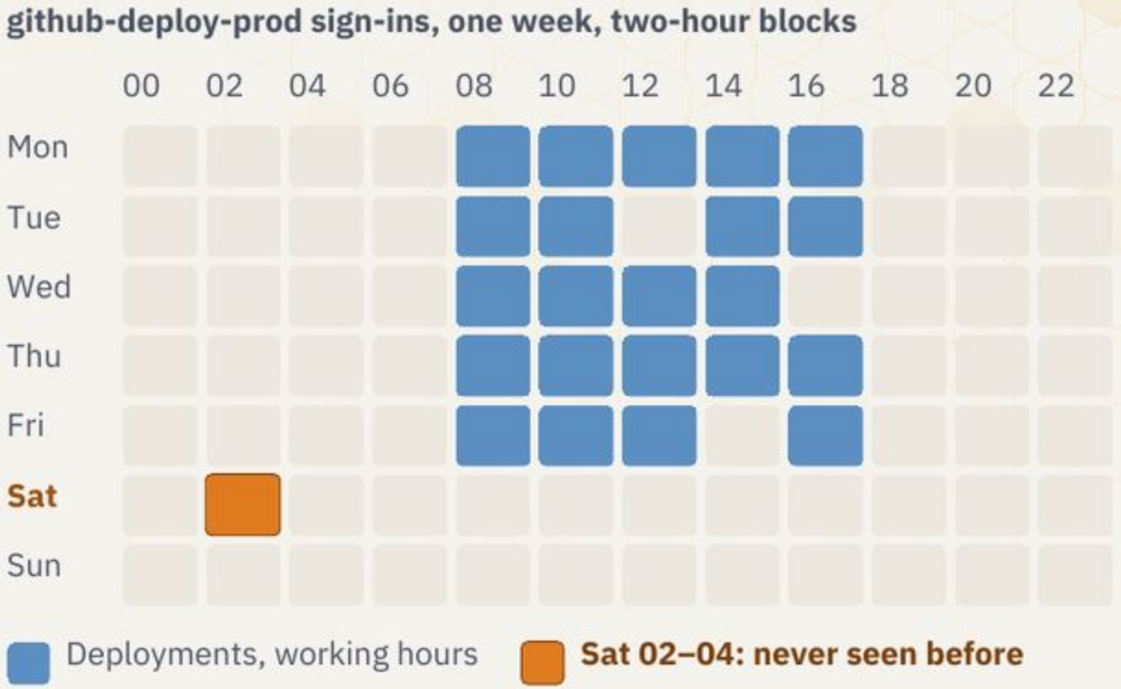
When

When is it active, and does that make sense?

When was it last used?

Only during deployments, or all the time?

Has the pattern changed?



Five W's framework: Adam Ochayon, Oasis Security · illustrative sign-in pattern

W 4 OF 5 · ENVIRONMENT

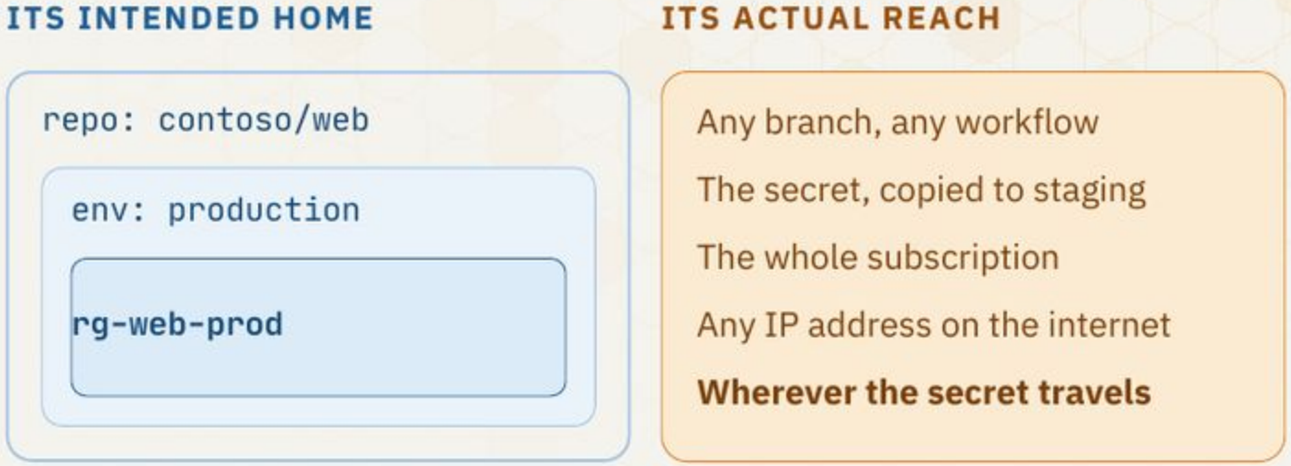
Where

Where should it work, and where does it reach?

Which environment is it meant for?

What breaks if it changes?

Is it tied to one repo and one environment?



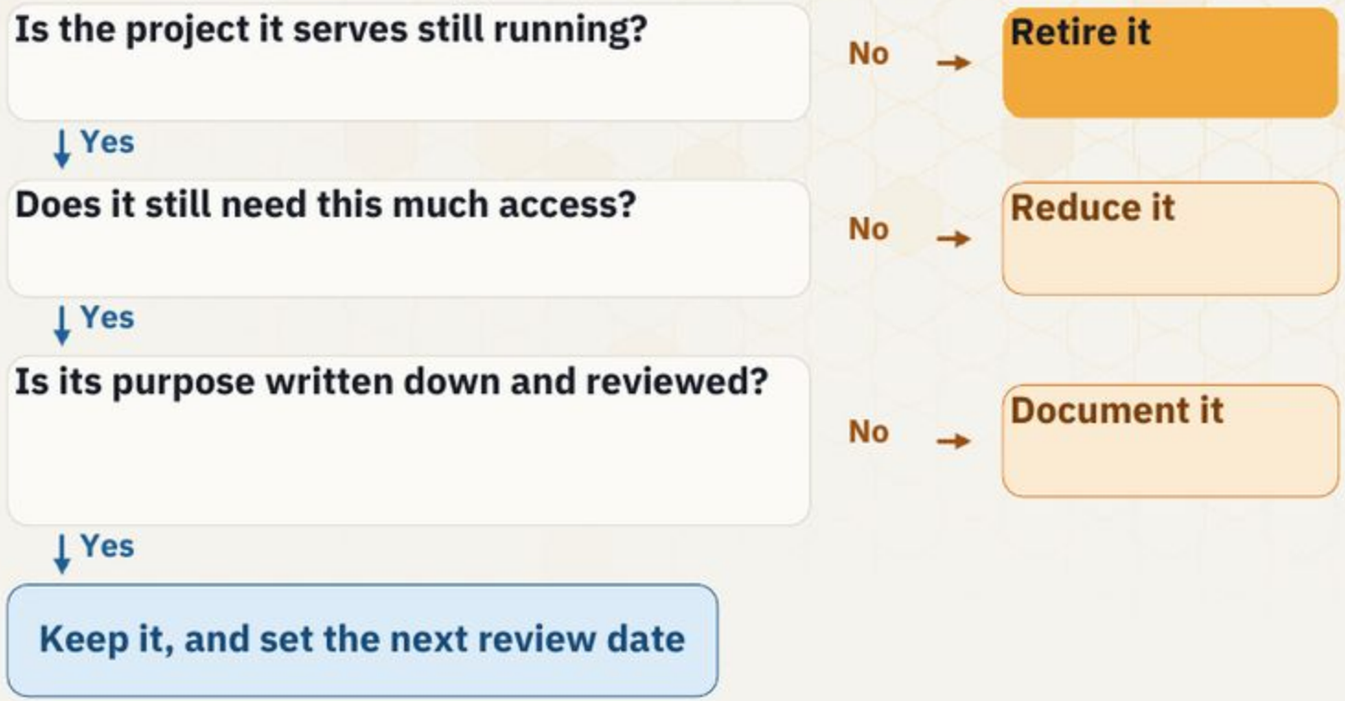
Five W's framework: Adam Ochayon, Oasis Security · applied to a pipeline identity

W 5 OF 5 · PURPOSE

Why

Why does it exist, and is it still needed?

Identities stay behind long after the project they served. The exposure stays in the present.



Five W's framework: Adam Ochayon, Oasis Security · applied to a pipeline identity

Part 5 of 6

How to fix them

Controls you already own, then how they work under the hood

✓ 1 · Why pipelines are the target

✓ 2 · Why we lose track of them

✓ 3 · What goes wrong

✓ 4 · How to audit them

5 · How to fix them

6 · Wrap-up

FROM QUESTIONS TO CONTROLS

Every question has a control you already own

Ask	Best practice	For a pipeline, in Entra and GitHub
Who	Assign a named owner	App registration owners, plus a team tag
What	Enforce least privilege	Resource-group scope, no Graph write in the deploy path
When	Monitor its heartbeat	Service principal sign-in logs, Conditional Access for workload identities
Where	Pin the scope	One federated credential per repo and environment
Why	Prune what isn’t needed	Access reviews for service principals, retire unused apps

Best practices follow Adam Ochayon’s five W’s. Conditional Access and access reviews for workload identities need Workload ID Premium.

CHOOSING THE IDENTITY

Which identity should a pipeline use?

Does the job run on Azure compute you control?

Yes → **Managed identity**
No credential to handle at all

↓ No

Does your CI platform issue OIDC tokens? GitHub, Azure DevOps and GitLab do.

Yes → **Workload identity federation**
Scoped to one repo and environment

↓ No

Last resort: a certificate
In Key Vault, short-lived, with a named owner

~~Client secret in repository settings~~

UNDER THE HOOD · 1 OF 6

Anatomy of the token GitHub gives a job

OIDC ID token · decoded payload (excerpt)

```
{
  "iss": "https://token.actions.githubusercontent.com",
  "aud": "api://AzureADTokenExchange",
  "sub": "repo:contoso/web:environment:production",
  "repository": "contoso/web",
  "ref": "refs/heads/main",
  "environment": "production",
  "job_workflow_ref": "…/deploy.yml@refs/heads/main",
  "actor": "j.weber"
}
```

WHAT ENTRA CHECKS

`iss`
Who signed it. Must be GitHub.

`aud`
Who it’s for. Must be Entra’s exchange.

`sub`
Which job. **This is the security decision.**

The other claims are context for logs and policy.

The exchange, step by step

1
Job

Asks GitHub for an ID token

`id-token: write`

→

2
GitHub

Signs a short-lived JWT with the job's claims

→

3
Job → Entra

Presents the JWT to Entra's token endpoint

→

4
Entra

Checks the signature against GitHub's public keys, then iss, sub, aud

→

5
Entra → Job

Returns an access token for the app. The job calls Azure.

Nothing stored in repository settings

Nothing to rotate, every token is short-lived

Trust = a signature plus three claims

Setting it up: two files, no secrets

In Entra · credential.json

```
{
  "name": "web-prod",
  "issuer": "https://token.actions.githubusercontent.com",
  "subject": "repo:contoso/web:environment:production",
  "audiences": ["api://AzureADTokenExchange"]
}
```

```
$ az ad app federated-credential create \
    --id $APP_ID --parameters credential.json
```

In GitHub · deploy.yml

```
permissions:
  id-token: write
  contents: read
jobs:
  deploy:
    environment: production
    steps:
      - uses: azure/login@v2
        with:
          client-id: ${{ vars.AZURE_CLIENT_ID }}
          tenant-id: ${{ vars.AZURE_TENANT_ID }}
          subscription-id: ${{ vars.AZURE_SUB_ID }}
```

Highlighted lines must agree: **the environment in the workflow is the subject in Entra.** IDs are variables, not secrets.

UNDER THE HOOD · BOUNDARY 2, AUTHORIZATION

The permission policy: what it may then do

WHAT YOU FIND

```
AWS: "Action": "*"
     "Resource": "*"

Entra: Owner on the
       subscription
```

Granted once “to unblock a release”, never walked back.

WHAT IT SHOULD BE

```
AWS: specific actions,
one resource ARN

Entra: built-in role,
scoped to one
resource group
```

The deploy job writes one app. The role should say exactly that.

Scope beats role name

“Contributor” on a tenant is worse than “Owner” on one resource group.

No standing write to identity

Deploy roles never get Graph / directory write. That’s the tenant takeover path.

One identity per environment

Prod, staging, dev each own. A token for one is useless against another.

Illustrative. Refs: AWS “Grant least privilege”; Microsoft “Best practices for Azure RBAC”.

Four workflow settings that shrink the identity

1 · Start with no permissions

```
permissions: {}  # workflow level
id-token: write  # deploy job only
```

Tokens exist only in the job that needs them.

2 · Pin actions to a commit

```
uses: actions/checkout@<full-commit-sha>
# v4, not @v4
```

A moved tag can’t change what runs.

3 · Split build from deploy

```
build:  # no cloud identity at all
deploy: { needs: build, environment: production }
```

Third-party build tools never meet the deploy identity.

4 · Protect the environment

```
Required reviewers: 1
Deployment branches: main only
```

A production token needs a human yes.

Conditional Access for a pipeline identity

Name	**Block deploy identities outside the runner network**
Workload identities	`deploy-prod, deploy-dev`
Target resources	All resources
Locations	Any location, **except** “Self-hosted runners”
Grant	**Block access**
State	Report-only first, then on

KNOW THE LIMITS

- Single-tenant service principals only, not managed identities
- Needs predictable runner IPs
- Add a second policy on service principal risk
- Requires Workload ID Premium

Four signals worth a scheduled query

1 · A service principal signs in from a new IP

```
let known = AADServicePrincipalSignInLogs
| where TimeGenerated between (ago(14d) .. ago(1d))
| distinct ServicePrincipalId, IPAddress;
AADServicePrincipalSignInLogs
| where TimeGenerated > ago(1d)
| join kind=leftanti known on ServicePrincipalId, IPAddress
| project TimeGenerated, ServicePrincipalName, IPAddress, ResultType
```

Pipelines are predictable, so “new” is a strong signal.

2 · A credential added to an app

AuditLogs: “Add service principal credentials”, certificates and secrets changes

3 · A Graph permission granted

AuditLogs: “Add app role assignment to service principal”

4 · A federated credential changed

AuditLogs: “Update application” events touching federated credentials

Federation done badly is a static secret with extra steps

No sub condition: any repo on the platform can assume the role

repo:acme/* : every repo in your org, test repos and forked workflows included

repo:acme/payments : one repo, but any branch and any workflow

`repo:acme/payments:environment:production`

Only this reaches prod. Also match on repository_id, so a deleted and re-created repo name can't inherit the trust.

Get CI out of the cluster

BEFORE · CI pushes



CI can schedule any workload, on any node, as any service account.

AFTER · the cluster pulls



0 cluster credentials

Trade-off: the config repo becomes the crown jewel. Protect it like prod with branch protection, required reviews and signed commits.

WHEN IT GOES WRONG ANYWAY

When a pipeline identity is compromised

1. **Contain**
   Disable the federated credential / detach the role trust. Cheaper than rotating a sprawl of secrets.

2. **Scope**
   Pull the audit log: what did this identity do, and when did it start?

3. **Hunt for persistence**
   Did it create a new credential or identity? That outlives your containment.

4. **Close the link**
   Which boundary failed? Fix that one, add the negative test, then re-enable.

ASK THIS FIRST

“Who owns this identity?”

For a human account the answer is instant. For a pipeline identity it is often a shrug. Decide that **before** the incident.

Federation turns containment from “rotate everything” into “flip one switch”.

MEASURE IT LIKE A PROGRAM

Six numbers to report every month

Pipeline identities with a named owner

100%

Who

Holding Graph write in the deploy path

0

What

Service principal sign-ins reviewed

Weekly

When

Federated credentials broader than one repo and environment

0

Where

Deploying without a stored secret

100%

How it signs in

Unused or unowned apps still active

0

Why

Suggested targets. Report today’s baseline first, then the trend.

Developers don't resist security. They resist friction.

✓

The paved road

A reusable workflow with OIDC already wired in. A Terraform module that scopes a new repo's role in one PR.

⚠

The old way

Needs a ticket and a secret paste. If the secure path is a one-line uses: people migrate themselves.

Track deployment lead time before and after to prove nothing slowed down.

Part 6 of 6

Wrap-up

What comes next, and what to do this week

✓ 1 · Why pipelines are the target

✓ 2 · Why we lose track of them

✓ 3 · What goes wrong

✓ 4 · How to audit them

✓ 5 · How to fix them

6 · Wrap-up

WHAT COMES NEXT

The next non-human identity is already here

</>

Service account

Runs a fixed script

WHO DECIDES WHAT IT DOES?

The script's author

→

Pipeline

Runs whatever the repo says

WHO DECIDES WHAT IT DOES?

Anyone who can change its code

→

AI agent

Decides its own next step

WHO DECIDES WHAT IT DOES?

Anyone who can change its input

Same five questions. The answers just change faster.

WHAT'S NEXT

Now we're putting agents in the pipeline

An AI coding agent or MCP tool running inside a CI job holds the job's token. Same rules apply with a twist: the agent decides *at runtime* what to do with that access.

⚠ The Nx incident already showed malware using local AI CLIs to hunt for secrets.

Do this on Monday

- [ ] List every long-lived cloud key CI holds, with its age
- [ ] Move one prod pipeline to OIDC, delete its key
- [ ] Check every OIDC trust policy for wildcard or missing sub
- [ ] Set permissions: {} as the default in one repo
- [ ] Pin your three most-used third-party actions to SHAs

Do only the first box and you'll know more about your risk than most of the industry.

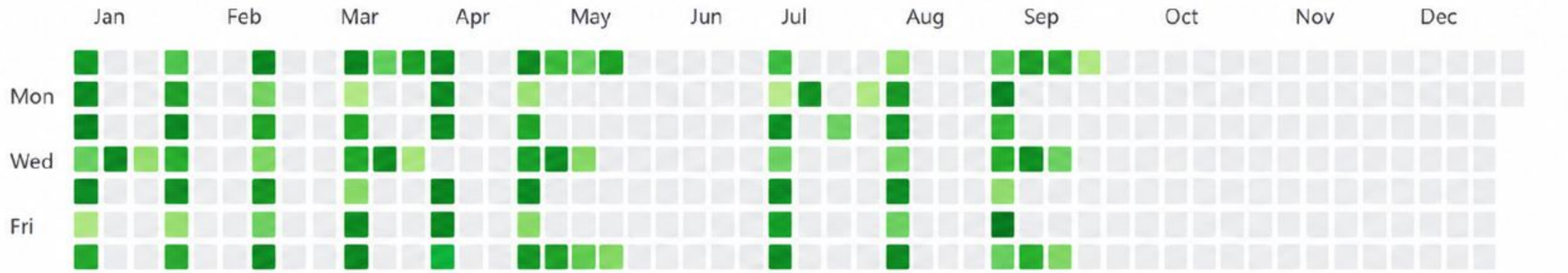
Treat your pipeline like your most privileged employee

Because it is.

https://mella7.me

138283 contributions in 2026

Contribution settings



Learn how we count contributions.

Less More

Contribution activity

2026

January - December 2026

2026

CLOUD IDENTITY SUMMIT '26

Thanks to our sponsors!

Gold Sponsors

adesso

glueck kanja

robopack

TOKEN swiss made software

Silver Sponsors

e·on

semperis

TEAL ALWAYS CHALLENGING IT

Bronze Sponsors

MSC Cyber Guard GmbH



CLOUD IDENTITY SUMMIT ‘26

Ask Me Anything (AMA)

Roundtable discussion and Q&A
on experiences from the field and current trends!

Meet the speakers and exchange
with members of the community!

4:30 PM @Community and Sponsor Area

Community Event by

Azure Meetup
BONN

CLOUD IDENTITY SUMMIT ‘26

Your Feedback is Important!

https://www.identitysummit.cloud/feedback/

Your CI/CD Pipeline Is Over-Privileged: Identity Risks in Modern Cloud Platforms

Community Event by

Azure Meetup

BONN