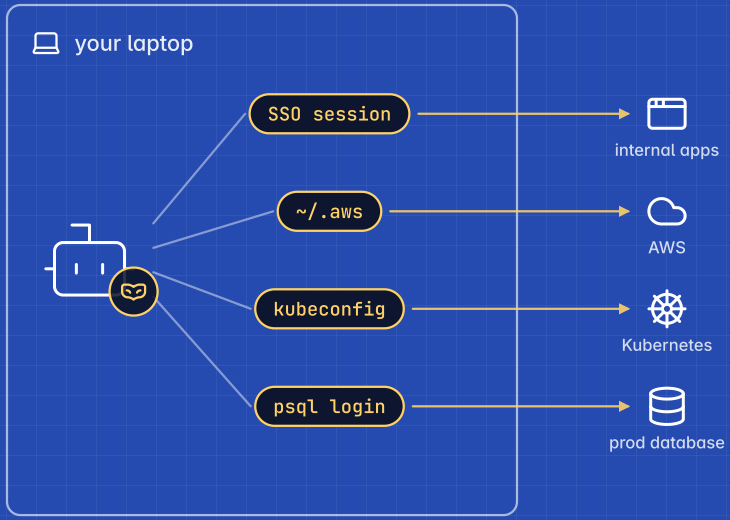
Letting cloud agents loose on our infra (without long-lived secrets)

Aditya Saligrama

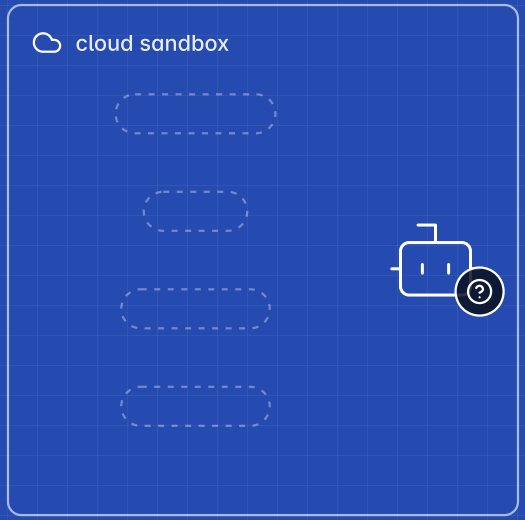
whoami

- Senior Software Engineer @ Formal
- prev @ CLOUDFLARE Verkada
- Lately: helping our cloud agents do more (safely)





The agent is you.



The agent is nobody.

Agent

I don't have database access. Can you run this and paste the results?

```
SELECT o.name, count(w.id) AS workflows
FROM organizations o
JOIN workflows w ON w.org_id = o.id
WHERE w.created_at > now() - interval '30 days'
GROUP BY o.name ORDER BY workflows DESC;
```

You

[Pasted text · 43 lines]

```
name,workflows
acme-corp,118
initech,64
...
```

Agent

Thanks! Can you break that down by week?

```
SELECT date_trunc('week', w.created_at) ...
```

You

[Pasted text · 211 lines]

Agent

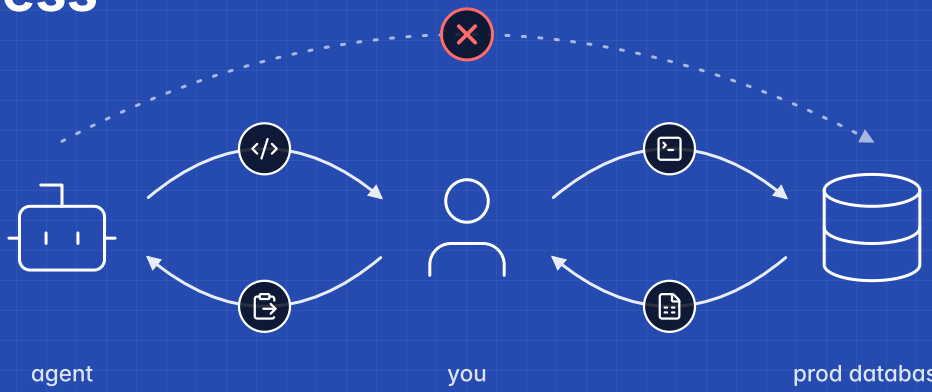
One more query...

What our agents needed access to

Production data	*How many customers adopted this feature?*
Workflow history	*Why is this job slow?*
Cloud accounts	*Does my change actually work on Azure?*

How do you give a cloud agent an identity?

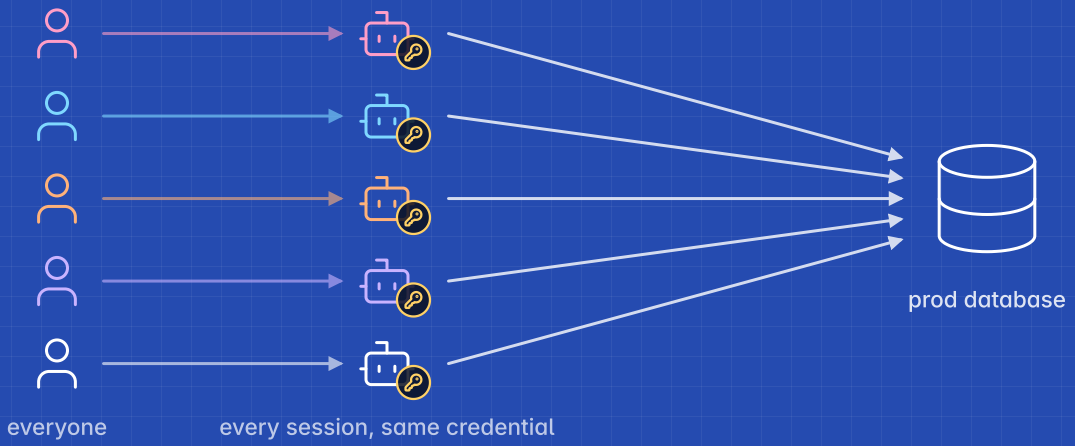
No access



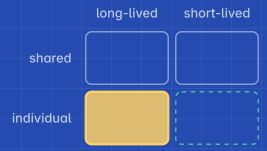

```
audit log: user=aditya
```

Shared key

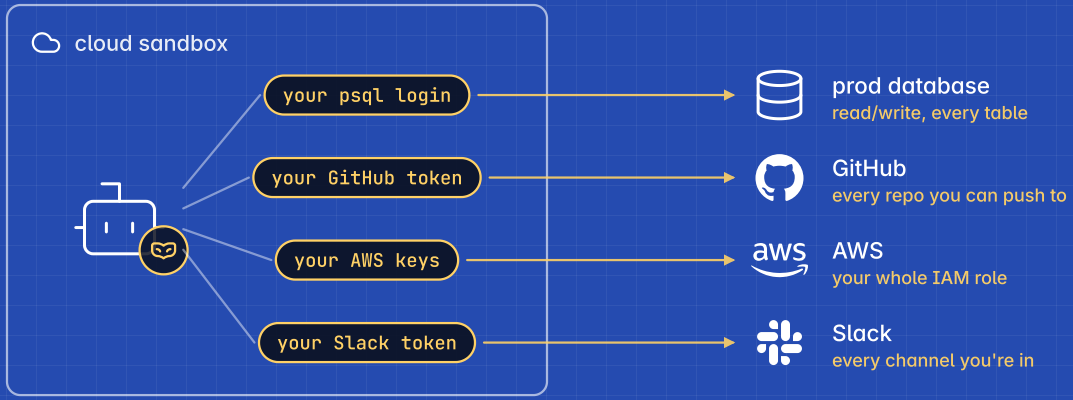





```
audit log: user=agent
```



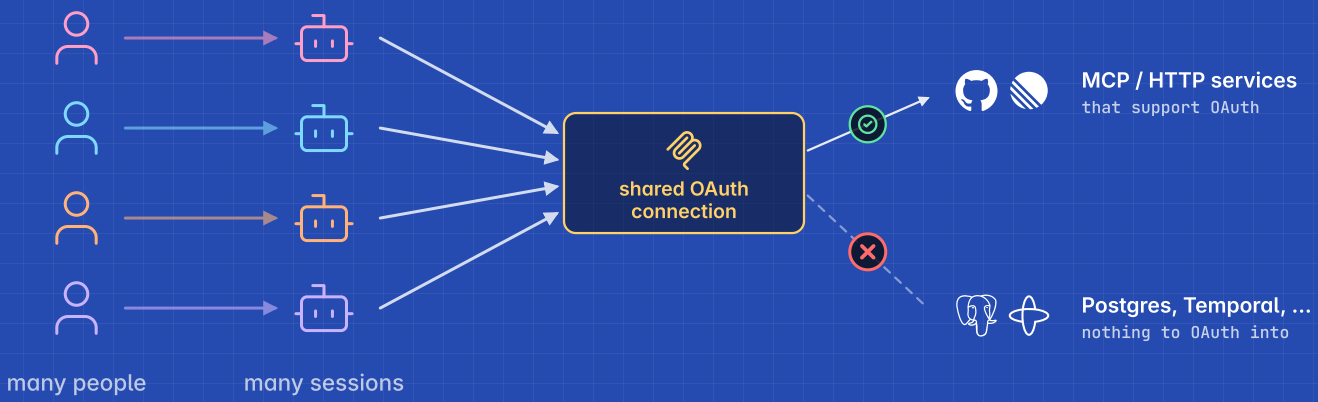
Your tokens in the sandbox




```
audit log: user=aditya (everywhere)
```

Platform identity



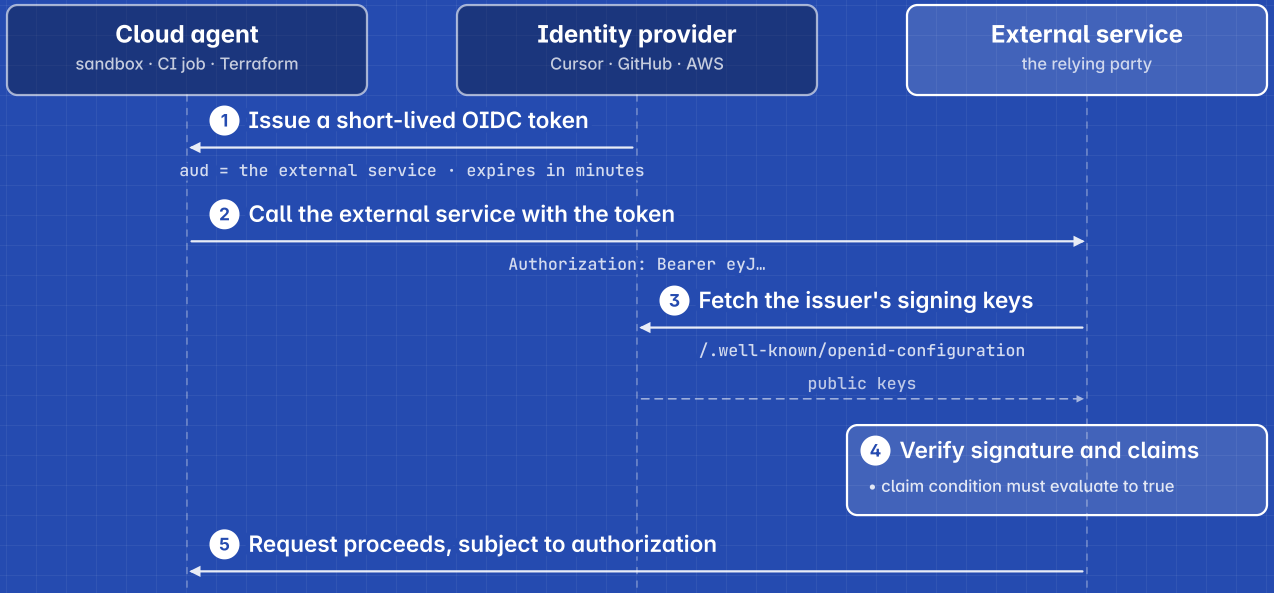



```
audit log: user=agent-platform[bot]
```

Who? For how long?

	Long-lived secret	Short-lived token
Shared identity log says “agent”	 **Shared key** `user=agent`	 **Shared OAuth connection** `user=agent-platform[bot]`
Individual identity log names someone	 **Your tokens in the sandbox** `user=aditya`	 **This session, on behalf of a person** `session bc-a853 · by aditya`

OIDC: federated workload identity



A real agent token

```
{
  "iss": "https://api.cursor.com",
  "aud": "oidc.formal.ai/integrationoidc_01m0…",
  "sub": "user:•••••••••",
  "iat": 1793137488,
  "exp": 1793137788,                    // five minutes later
  "cloud_agent_id": "bc-a8538043-…",
  "owner_email": "aditya@formal.ai",
  "repo_url": "github.com/formalco/monorepo",
  "branch_name": "cursor/oidc-talk-slidev-4a36",
  "team_id": "1723••••",
  "agent_runtime": "managed"
}                                       // + 9 more claims
```

Who can verify the token?

Can verify an OIDC token

- **AWS**
  `AssumeRoleWithWebIdentity`
- **Google Cloud**
  `Workload Identity Federation`
- **Azure**
  `federated identity credentials`
- **Kubernetes**
  `API server OIDC authentication`

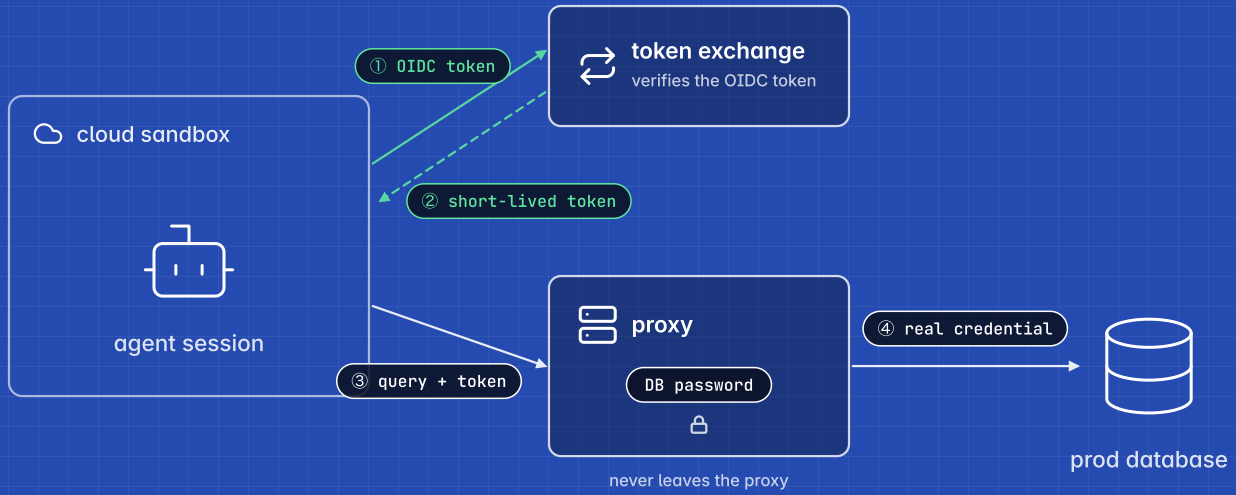
Can't

- **Postgres databases**
  `passwords, certs, cloud IAM tokens`
- **GitHub API**
  `PATs and GitHub App keys`
- **Slack**
  `OAuth bot and user tokens`
- **Temporal Cloud**
  `API keys or mTLS`
- **Internal tools**
  `usually SSO sessions or API keys`

`wif.tax` graded 111 products: 65 accept only long-lived secrets

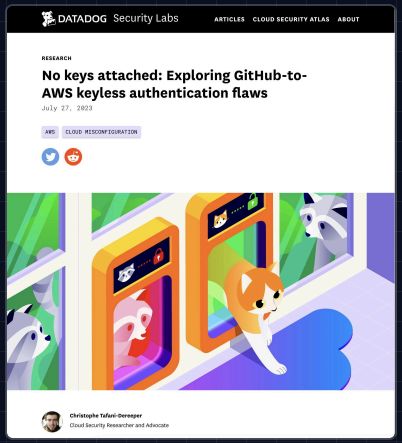
What we built

Proxying static credentials



The agent never holds the database password.

Signatures aren't enough



Skip it: any repo can assume your role.

Datadog Security Labs, July 2023

An issuer signs every customer's tokens with the same key.

```
// who may use this trust at all
claims.team_id == "1723••••"

// which human the session acts for
claims.owner_email
```

Modeling human identity

machine identity



`cloud_agent_id: bc-a8538043…`

`repo_url: …/formalco/monorepo`

`iss: api.cursor.com`

cryptographically verified

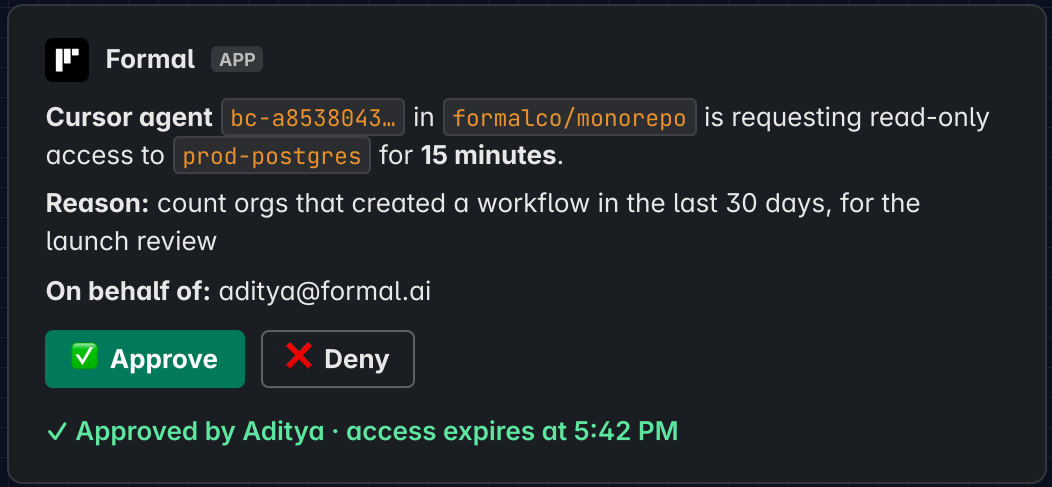
human identity



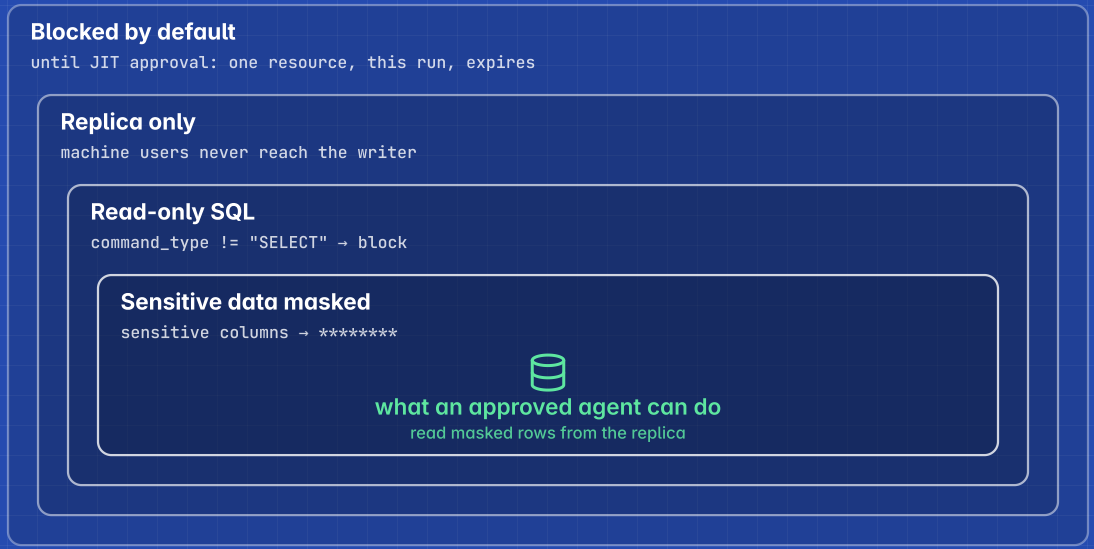
`owner_email: aditya@formal.ai`

asserted by a platform you trust

Extending JIT access to agents



Where the access ends



- Agent

  I need production data for this. Requesting read-only access to the analytics replica for 15 minutes.

- Slack

  approved by aditya · expires 17:12

- Agent

```
SELECT o.name, count(w.id) AS workflows
FROM organizations o
JOIN workflows w ON w.org_id = o.id
WHERE w.created_at > now() - interval '30 days'
GROUP BY o.name ORDER BY workflows DESC;
```

  42 orgs created a workflow in the last 30 days, up from 31 the month before. Top three: acme-corp (118), initech (64), globex (37). I also broke it down by week: adoption doubled after the launch email.

What the logs show

Every query is attributed to the session, the person who launched it, and the approval.

```
16:42:07  prod-postgres  read-only
  query           SELECT o.name, count(w.id) AS workflows …
  machine user    cursor-cloud-agent
  agent session   bc-a8538043…  (formalco/monorepo)
  on behalf of    aditya@formal.ai
  approved by     aditya@formal.ai · expires 17:12
```

Closing the loop: a real AKS cluster



Lessons learned (and open problems)

What surprised us

- Agents thrash when they hit a block
- Read-only is more nuanced than you think

Still unsolved

- No standard claims for agent sessions
- Propagating identity when agents spawn agents

Takeaways

1. Give each agent session its own identity

2. Use OIDC, check the claims, and proxy services that need static credentials

3. Don’t stop at authN: authorize every request with policy

Questions?

Aditya Saligrama · aditya@formal.ai

Read the blog post