
No excuses for Client Secrets

Use your Cloud Identity for Client Authentication



Arndt Schwenkschuster

Software Engineering & Identity Standards, Defakto Security

Identity enthusiast, SPIFFE Maintainer & IETF Editor

Proudly based in Prague 🇨🇪

What is NHI / non-human identity?

An identity used by **software**, not a human.

Workloads

VMs, Pods, containers, functions

Pipelines

CI/CD jobs, data processing

Service accounts

Scripts, batch jobs, backend integrations, database users

Third-party apps

SaaS integrations, API keys

Devices

IoT, edge nodes, on-prem servers

Cloud Identities

IAM Roles, GCP Service Accounts
Azure Managed Identities

AI agents

Acting autonomously across many tools

Human vs. non-human identities

	HUMAN	NON-HUMAN
Authenticates	Interactively, a few times a day	Continuously, machine to machine
Authentication factors	A few, limited by patience	Endless: OS, hardware, platform and software binding
Consent	Prompts and consent screens	None: nobody is there to click
Lifecycle	Joiner, mover, leaver	Created and destroyed in seconds
Typical credential	Moved past password-only years ago	Still mainly client secrets

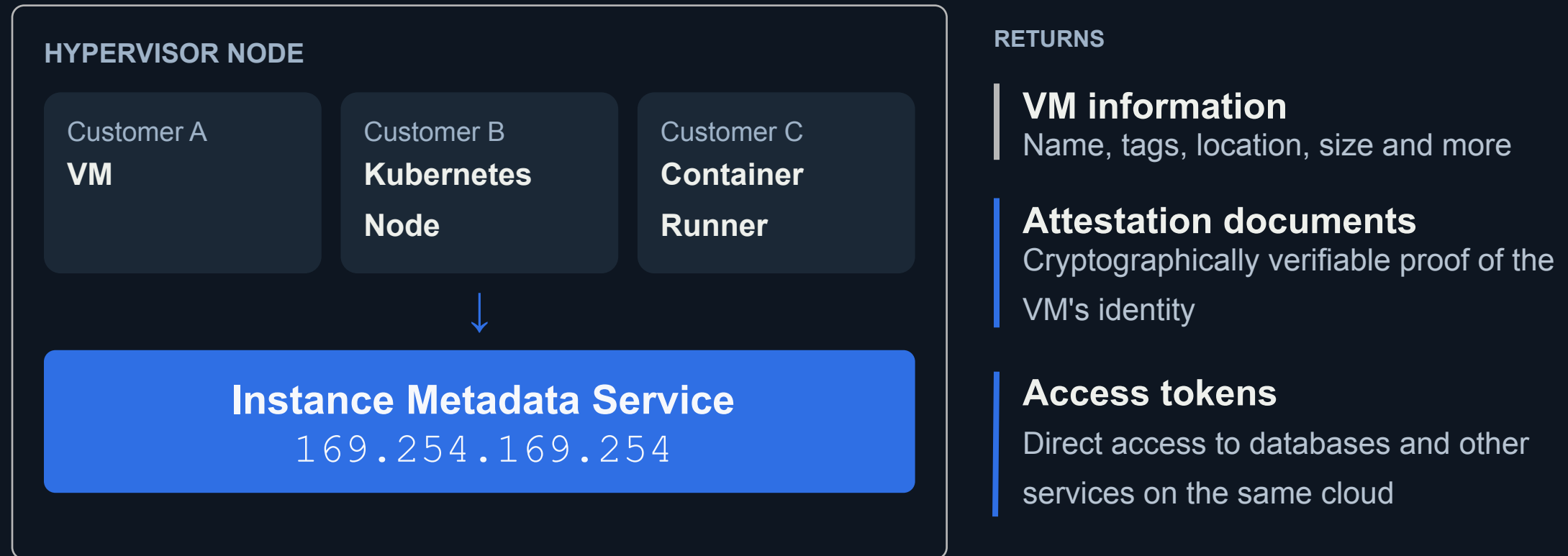
17:1 to **144:1**
conservative estimate high end

Non-human identities per human, depending on the source.

In the agentic AI era, API keys are not just a compliance item. They are a business-critical risk waiting to be misused.

The Instance Metadata Service pattern

Cloud workloads get an NHI by default and for free. Ready to be used.



Every major cloud implements it



Azure Managed Identity

System-assigned and
User-assigned



AWS IAM Roles

Attached to the instance or function



GCP Service Accounts

Attached to the workload

| If you want a common identity layer, including on-premise and other diverse environments



spiffe

Secure Production Identity Framework For Everyone

NHI and OAuth client authentication

METHOD	MECHANISM	ATTRIBUTES	Keycloak
Client secret	The "API key" of OAuth	Easy to use. Long-lived, hard to rotate, replayable, no theft protection, must be provisioned.	☒
Private key JWT	Signed assertion; solves exfiltration and replay	Mitigates replay and better theft protection. Key still long-lived and hard to rotate; same (if not worse) provisioning challenges.	☒
Assertion flow	Accepts externally issued JWTs, e.g. from the Instance Metadata Service	Short-lived and replay protection possible. Trust the issuer, instead of something individually provisioned. Can be used for many other things. SDK support not always guaranteed.	☒
SPIFFE client auth (RFC pending)	SPIFFE-specific authentication		experimental
Attestation-based client auth (RFC pending)	Trusts certificate hierarchies and attestation; built for ecosystems like app install attestation		experimental

Demo

