NINTH ANNUAL LEADERSHIP EVENT

CYBER SECURITY SUMMIT

Security solutions through collaboration.™

Authenticating Cross-Service Communications in a Multi-Cloud World

Sunil James, Scytale

October 28–30, 2019 | Minneapolis Convention Center

cybersecuritysummit.org | #cybersummitmn


VeriSign®


MANDIANT®
A FireEye® Company


ARBOR®
NETWORKS


amazon
web services


Google

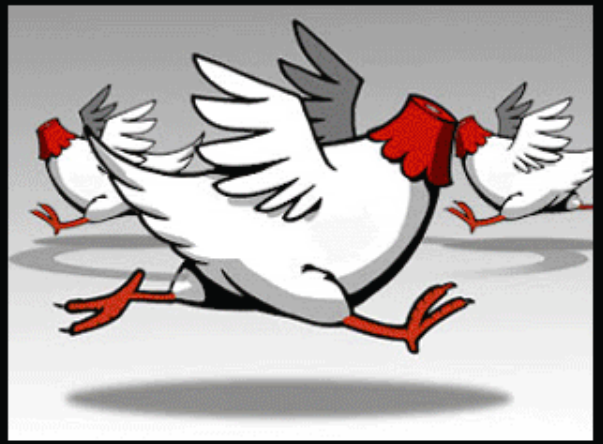
What my friends think I do

What my mom thinks I do

What society thinks I do

What my colleagues think I do

What I think I do



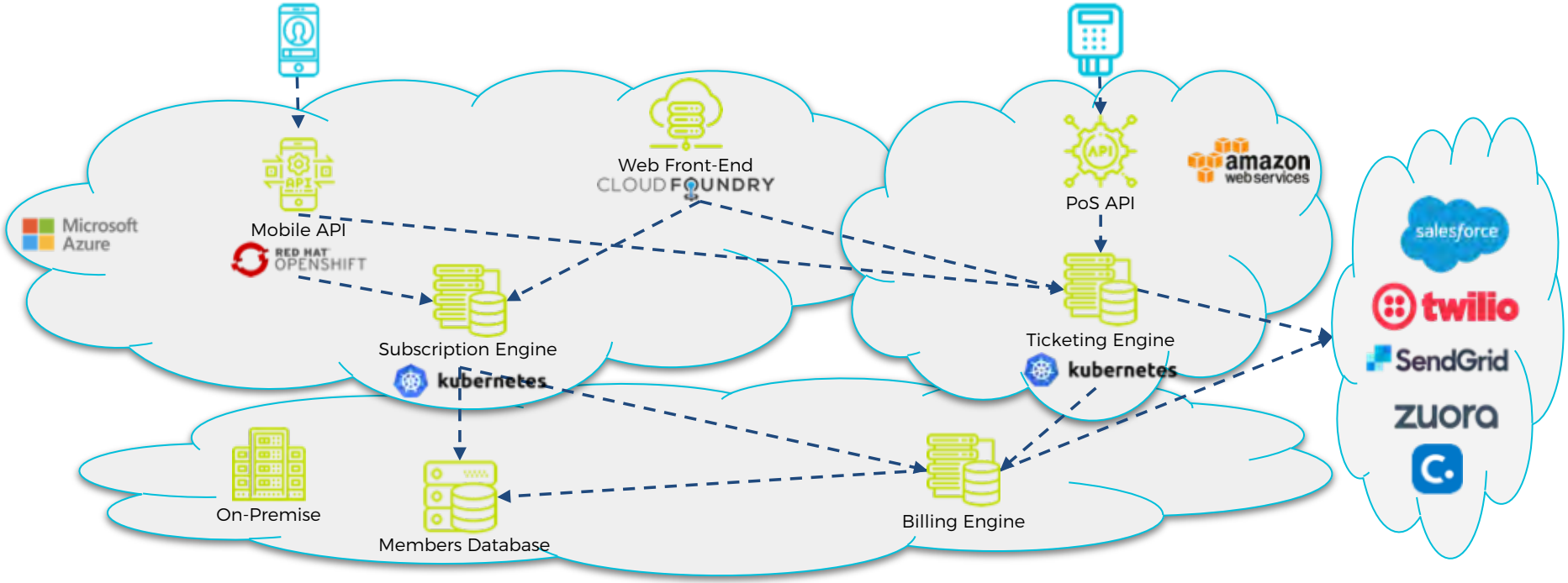
What I actually do

Bessemer
Venture
Partners

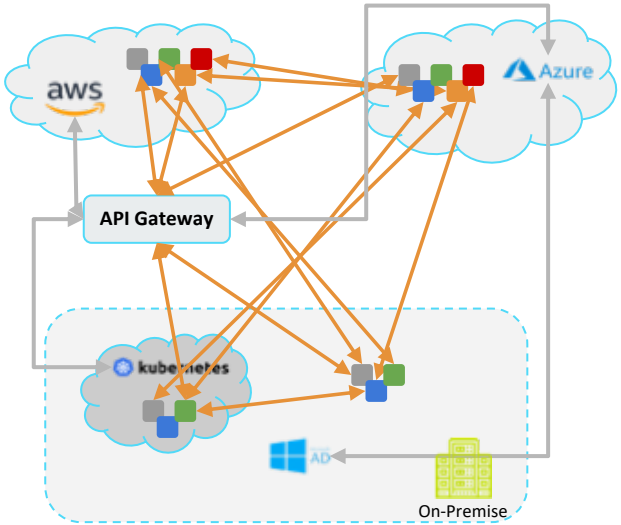
SCYTALE


KEISERIN
GUSTAV
RSVIKTET

Containers enable new architectures.



Service auth’n is complicated.



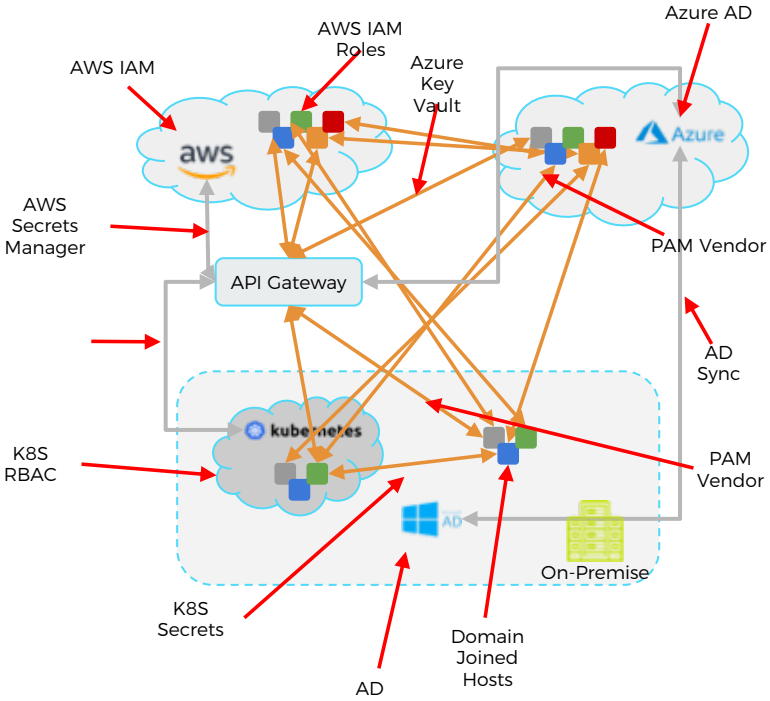
More compute, storage, and network choices for an enterprise means more disjoint identity providers (IdP) for services.

Evolving from a single on-premise IdP like Active Directory means there's no longer one source of truth defining a service.

Cloud "velocity" disappears as developers wait for IdPs to be “glued” together to support existing, risk-approved, multi-week workflows (ID creation, token rotation, ticket management, approvals, etc.).

- Slows time to market.
- Reduces platform adoption.
- Decreases human productivity.
- Introduces new security risks.

Enterprises inconsistently tackle auth'n.

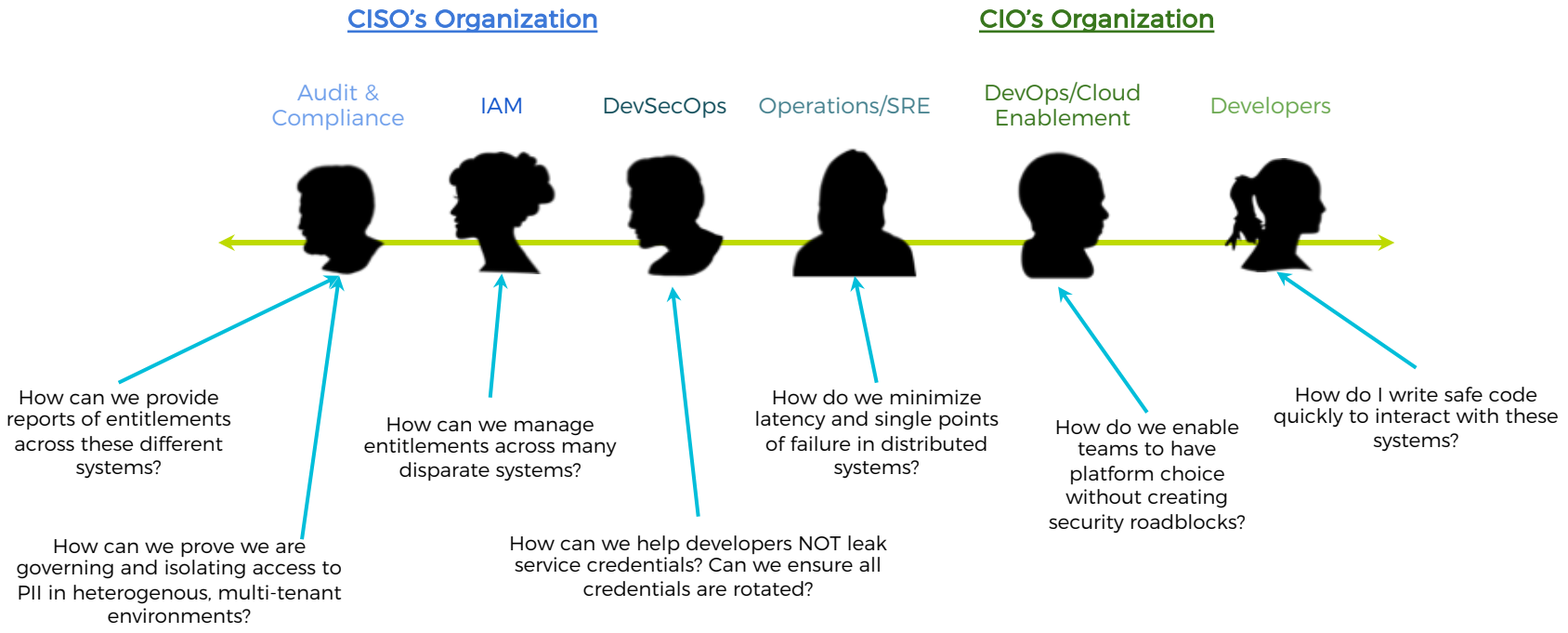


- Solutions are point-to-point and bespoke.
- It's impossible to model application entitlements.
- It's difficult to deliver, rotate, and audit credentials.
- Existing protocols do not scale for elastic or dynamic environments.
- There are no confidentiality guarantees.
- Network trust is still required and must be coordinated independently.

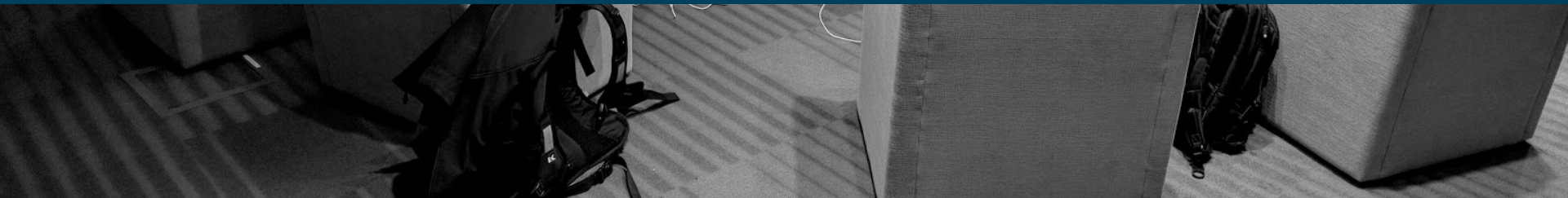
When does this problem surface?

1. Authenticate cloud services with on-premise services.
2. Secure how cloud services access secret stores.
3. Encrypt east-west service traffic within/across trust domains.
4. Reduce our dependency on API gateways to authenticate clients.
5. Deploy an authentication plane that plays nice with middleware.
6. Inject code signing primitives into cross—service authentication.
7. Determine the entitlements of a service chain based on the originating "human."
8. Apply sane and consistent authentication to IoT endpoints & 5G devices.

Differing needs tied to the same problem.



December 6, 2016



Multi-factor *auth'n* for services (not people).

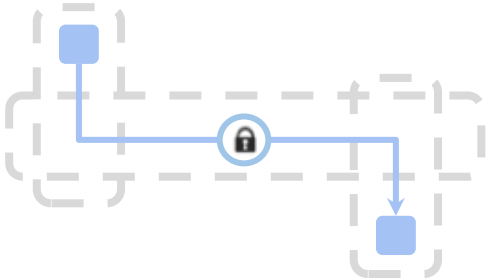
Has it been signed by the CI/CD pipeline?

Is it known to trusted middleware or schedulers?

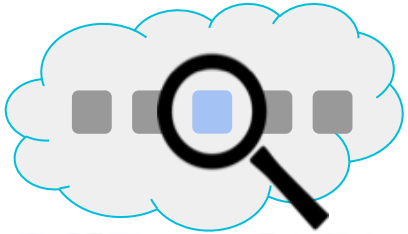
Can we affirm the integrity of the machine it runs upon?

Is the machine a member of a known network or cluster?

Eliminates hard-coded passwords. Well suited for dynamic environments. Improves auditability.



Reduces reliance on the network for integrity.

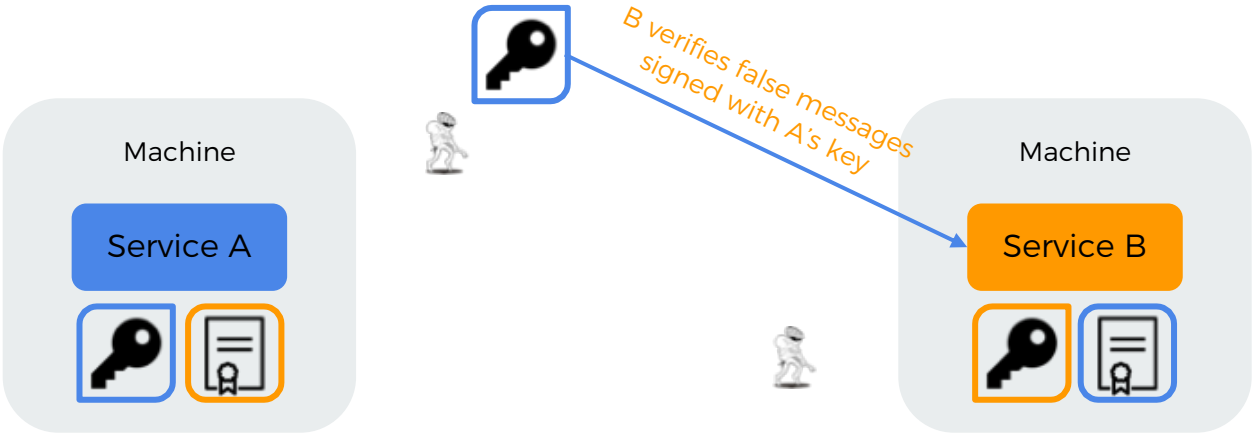


Multi-factor *auth'n* drives zero trust.

For two services to trust each other, they must **attest each other's identity**, and ensure messages between the two **have not been tampered with**. Mutual TLS, established with public and private keys, allows us to do this without necessarily trusting the underlying network.

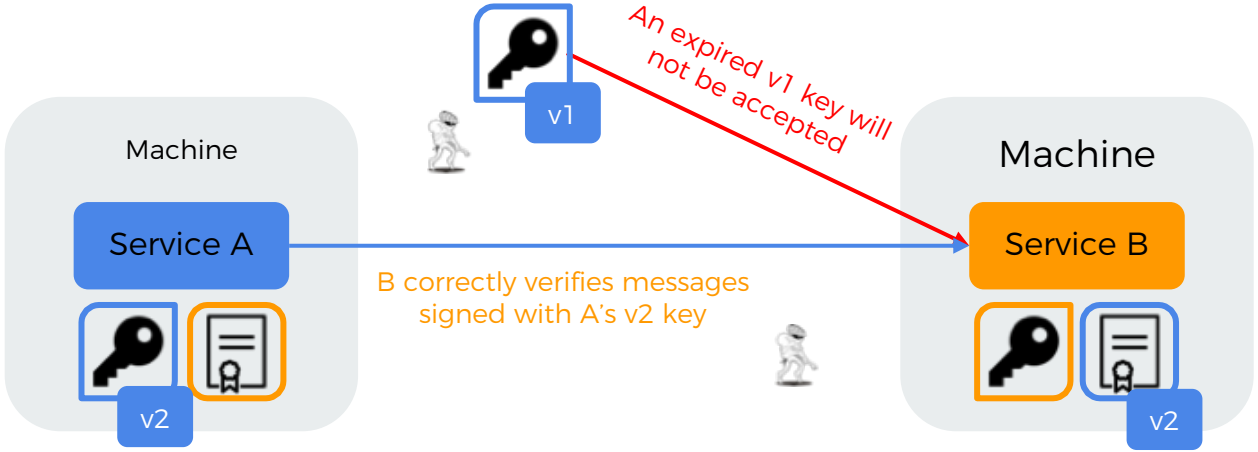
Multi-factor *auth'n* drives zero trust.

But maintaining the integrity of these keys, like any password, becomes critical. If a key for a service is stolen, then an attacker can impersonate that service.



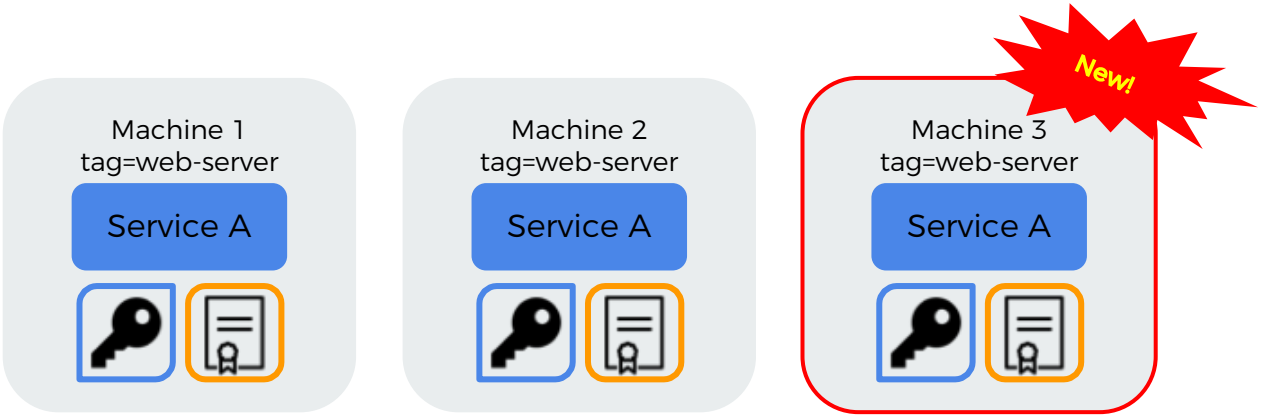
Multi-factor *auth'n* drives zero trust.

A solution to this is **short-lived keys and certificates**. An attacker cannot re-use a stolen key after it expires. Short-lived is typically < one hour. This eliminates a large range of exfiltration attacks.



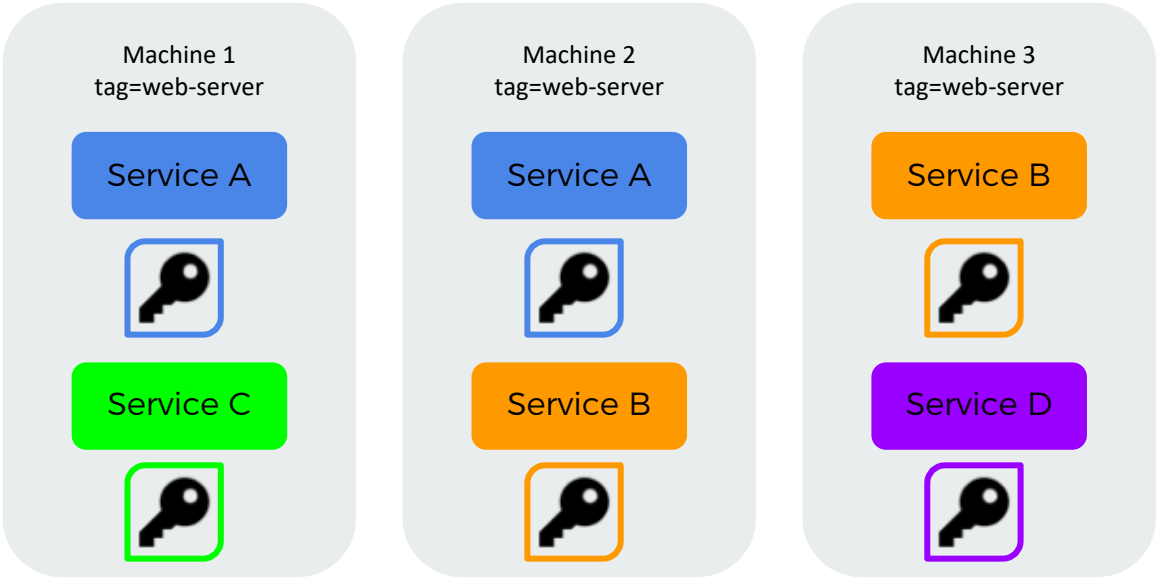
Multi-factor *auth'n* drives zero trust.

But how can we deliver new keys and certificates to every service, every hour? What's more, in dynamic computing environments, machines are arbitrarily created/destroyed based on business demand. Humans would slow down this loop.



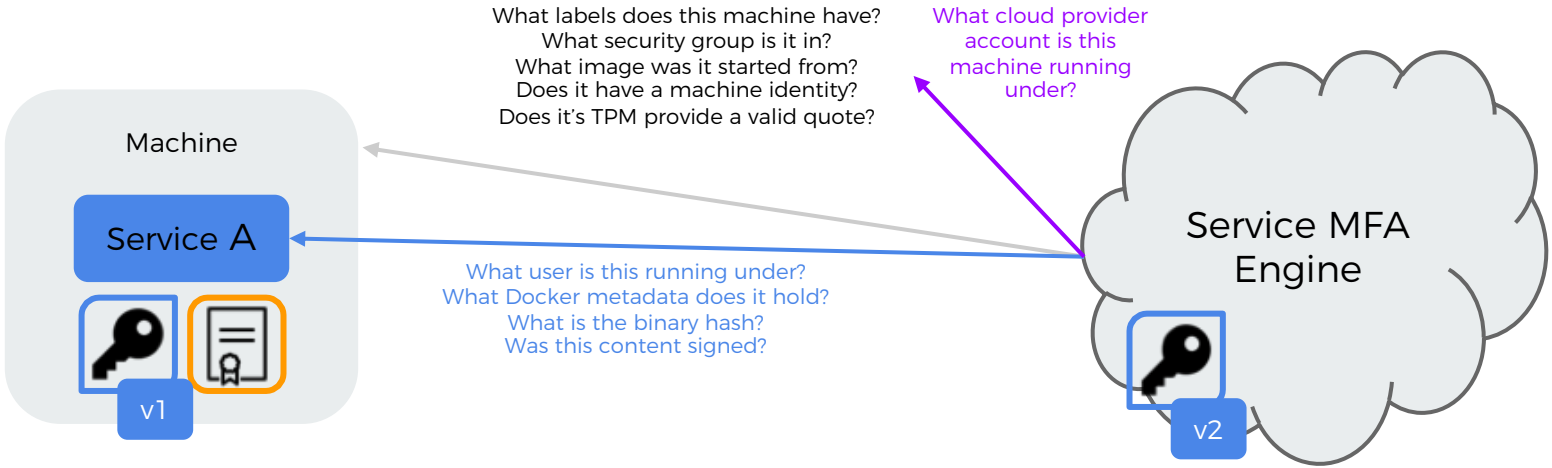
Multi-factor *auth'n* drives zero trust.

In a virtualized environment, multiple services could be balanced across varying machines. It's not feasible for humans to issue unique keys to each service on each machine.



Multi-factor *auth'n* drives zero trust.

Answer: automate delivery of short-lived certificates based on continuous assessment of **many attributes of a service** and its current environment.



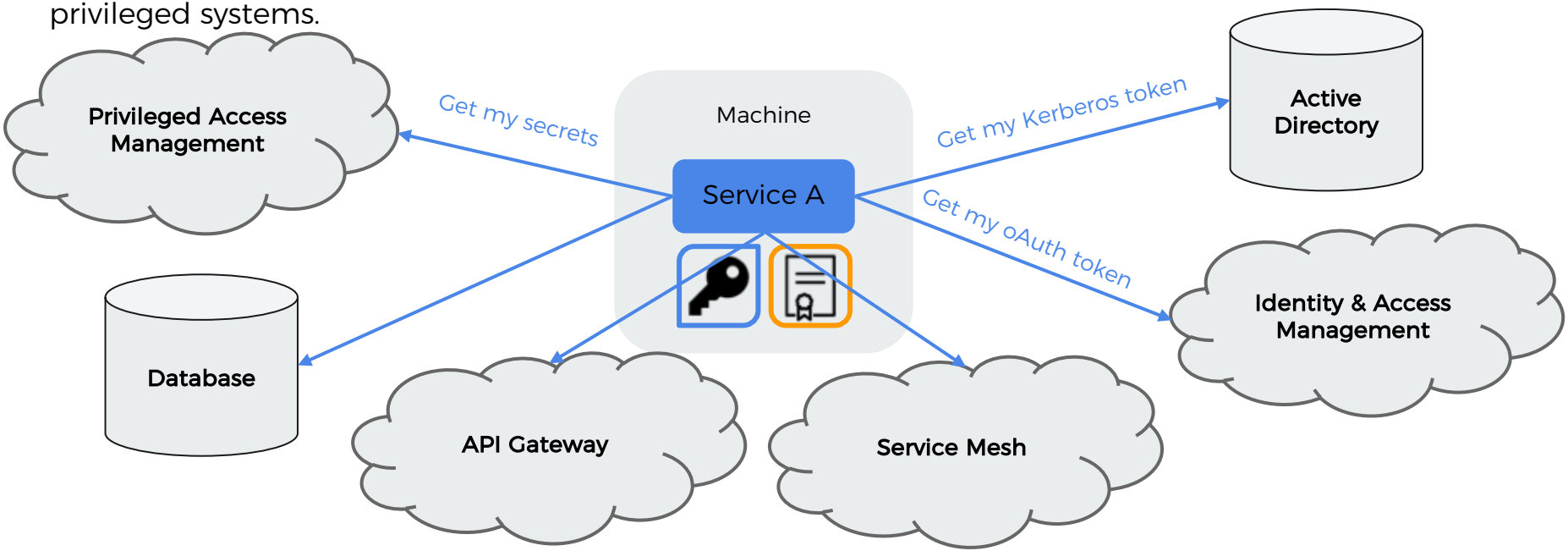
Multi-factor *auth'n* drives zero trust.

With this design, **every service**:

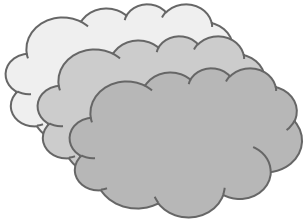
- Is continuously verified for integrity, even during deployment.
- Can be identified when part of an elastic cloud, or dynamically scheduled container environment.
- If verified, can establish trust and encrypt traffic to other services even in a compromised network.
- Holds only short-lived keys and certificates that are protected from exfiltration.

Multi-factor *auth'n* drives zero trust.

This continuously verified, short lived "master key" can then be used to bootstrap trust into other privileged systems.



A viable solution is “flexibly prescriptive.”



Multi-cloud ready

Support on-prem and cloud environments.

Scalable to large workload volumes.

Avoid single points of failure.

Container ready

Support deployment to elastic & dynamically scheduled environments.

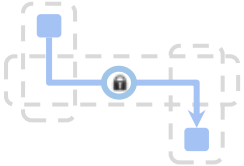
Support audit trails where IPs are ephemeral.

Minimize secret sprawl

Automates credential delivery.

Employs asymmetric crypto to assert identity.

Encourage short-lived, scoped tokens.



Enable defense in depth for mixed mode workloads

Ensure workloads can mutually identify each other specifically and establish an encrypted mTLS tunnel.

Built on open standards

Become a platform for a range of integrations.

SPIFFE defines *passports* for services.

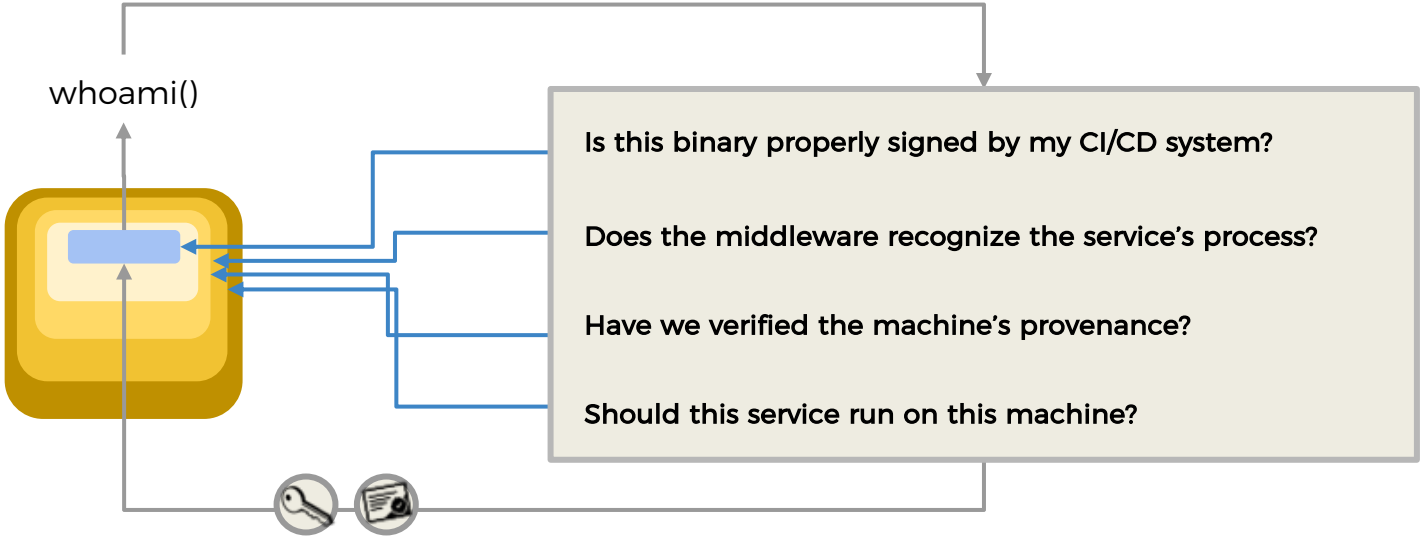
Launched in December 2017.

Accepted into the CNCF in March 2018.

Integrated into multiple cloud-native open-source projects.

Deployed by a growing number of enterprises.

SPIRE *verifies* issued SPIFFE passports.



How does attestation work?

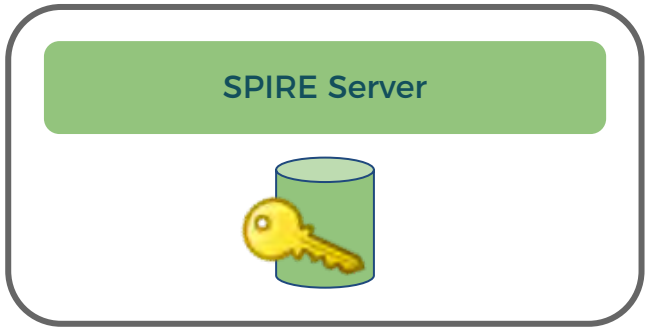
spiffe://acme.com/billing/payments

factor: aws:sg:sg-edcd9784
factor: k8s:ns:payments
factor: k8s:sa:pay-svc
factor: docker:image-id:442ca9

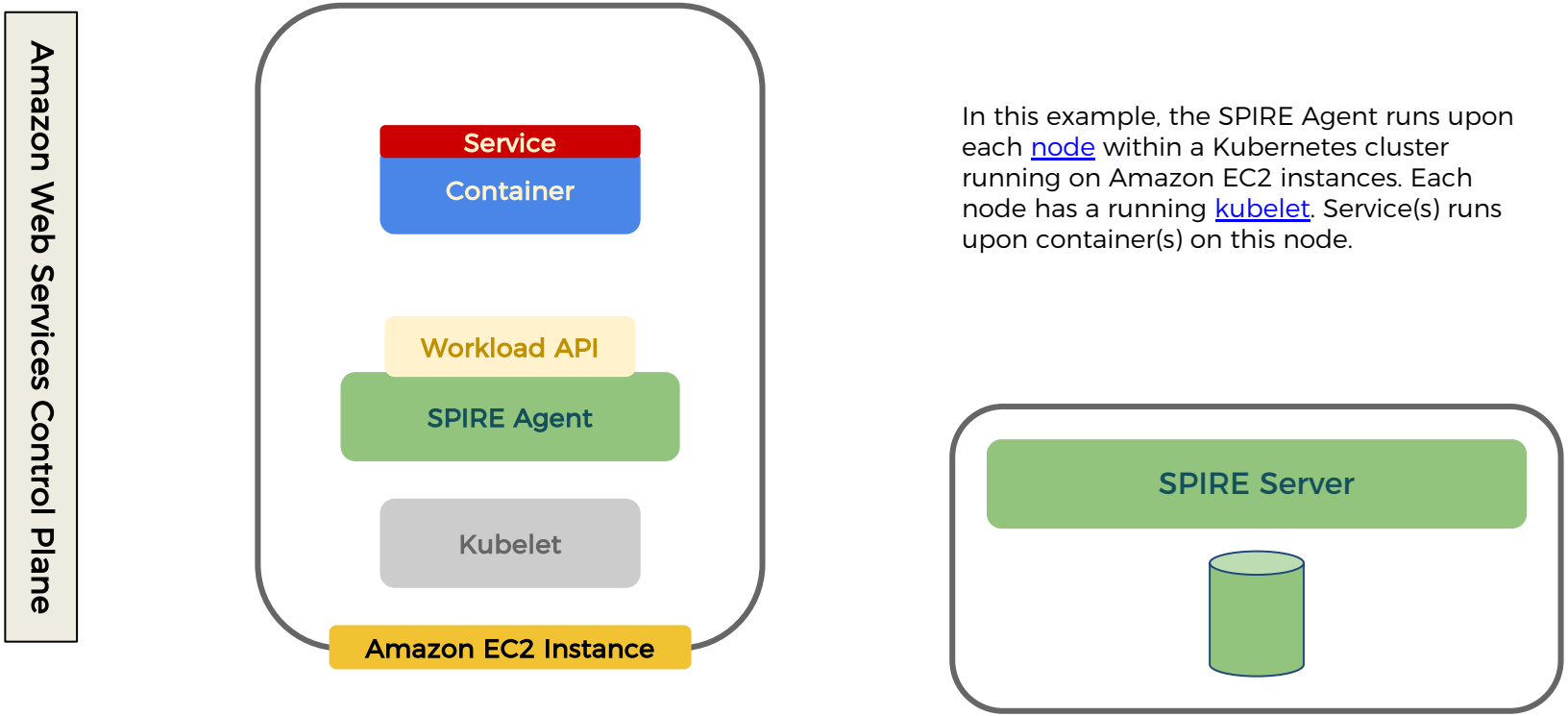
The SPIRE Server (eg.acme.com) maintains a list of:

1) Valid service identities (eg. /billing/payments).

1) Factors that must be matched by a service to be entitled to a valid service identity.

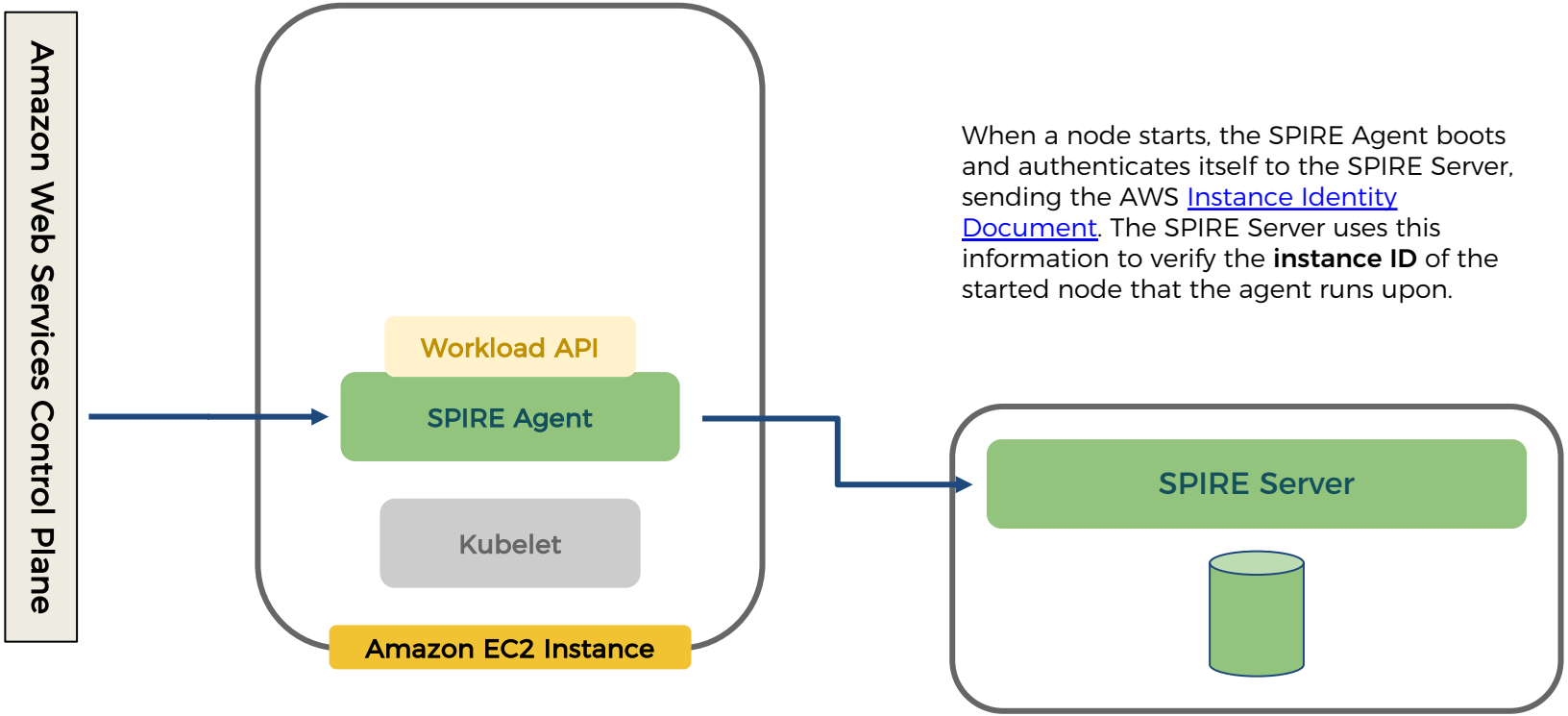


How does attestation work?



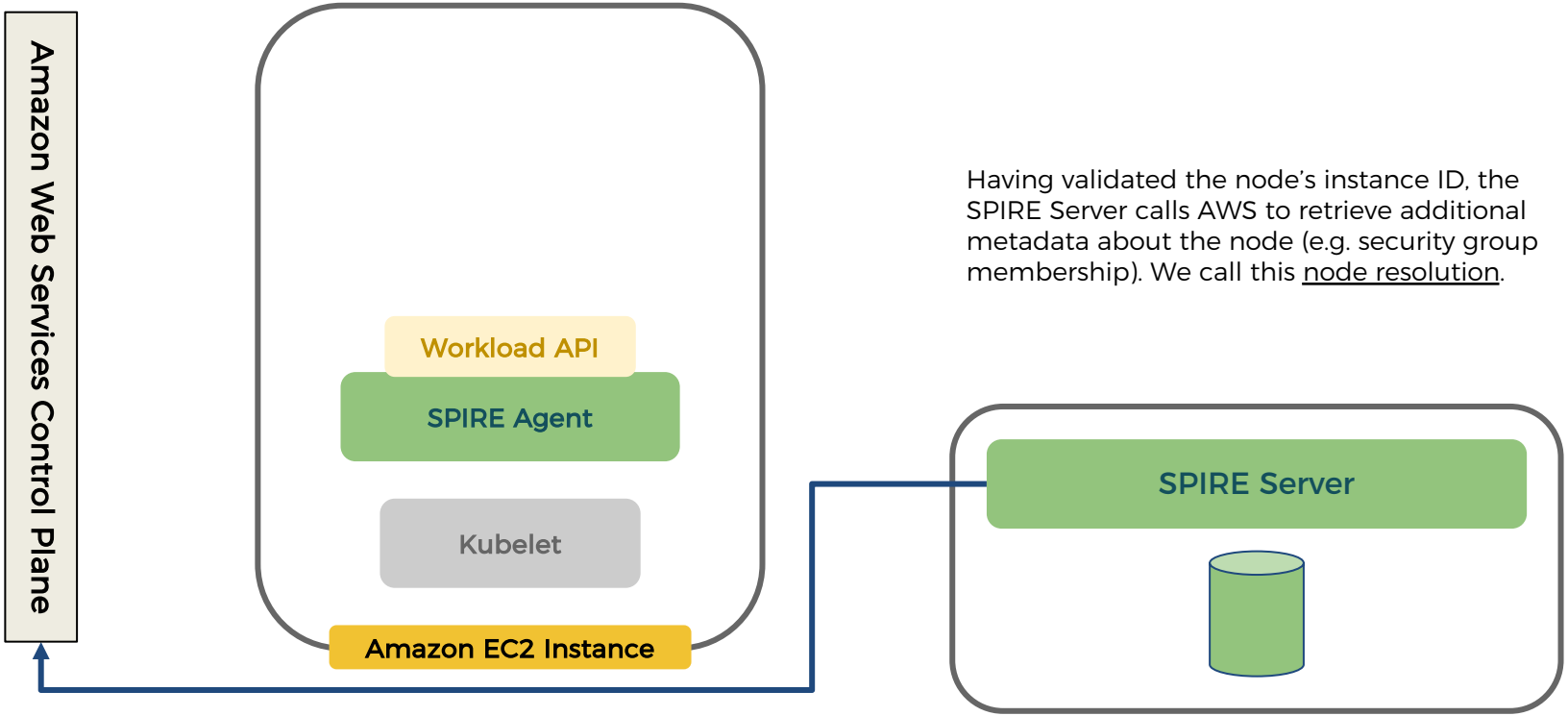
In this example, the SPIRE Agent runs upon each node within a Kubernetes cluster running on Amazon EC2 instances. Each node has a running kubelet. Service(s) runs upon container(s) on this node.

How does attestation work?



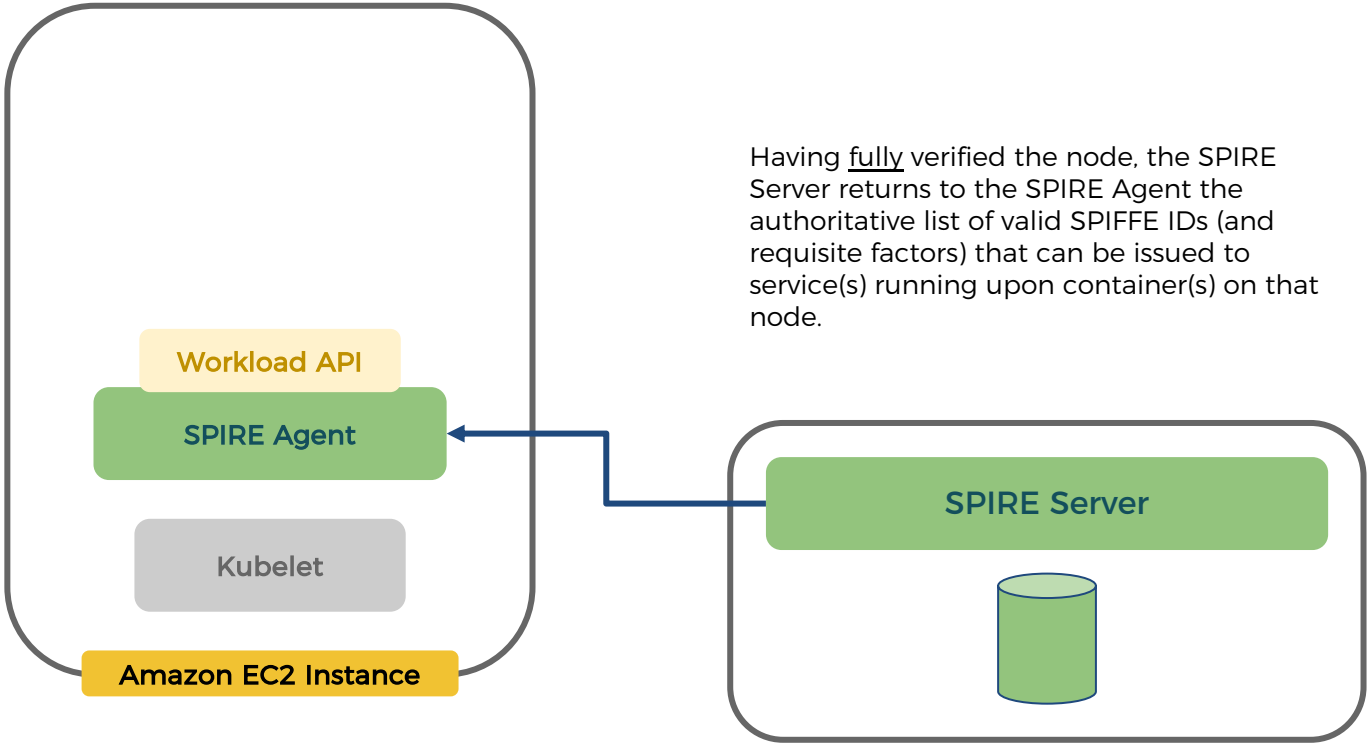
When a node starts, the SPIRE Agent boots and authenticates itself to the SPIRE Server, sending the AWS Instance Identity Document. The SPIRE Server uses this information to verify the **instance ID** of the started node that the agent runs upon.

How does attestation work?



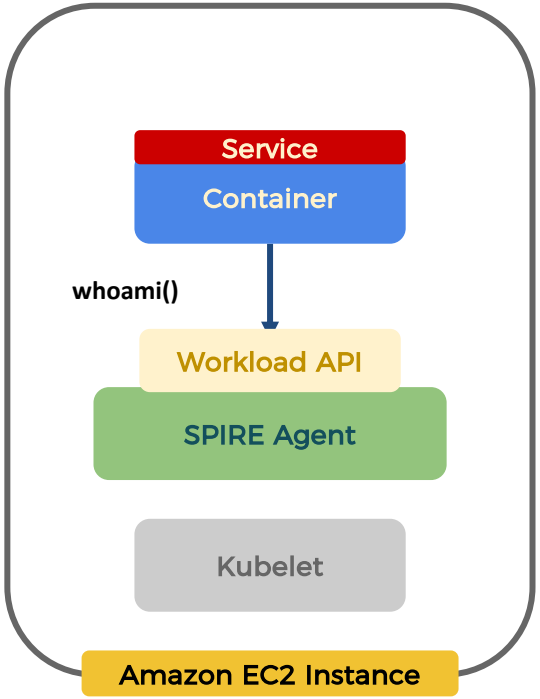
Having validated the node's instance ID, the SPIRE Server calls AWS to retrieve additional metadata about the node (e.g. security group membership). We call this node resolution.

How does attestation work?

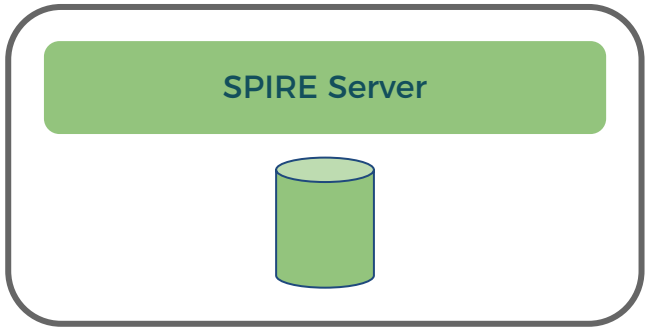


Having fully verified the node, the SPIRE Server returns to the SPIRE Agent the authoritative list of valid SPIFFE IDs (and requisite factors) that can be issued to service(s) running upon container(s) on that node.

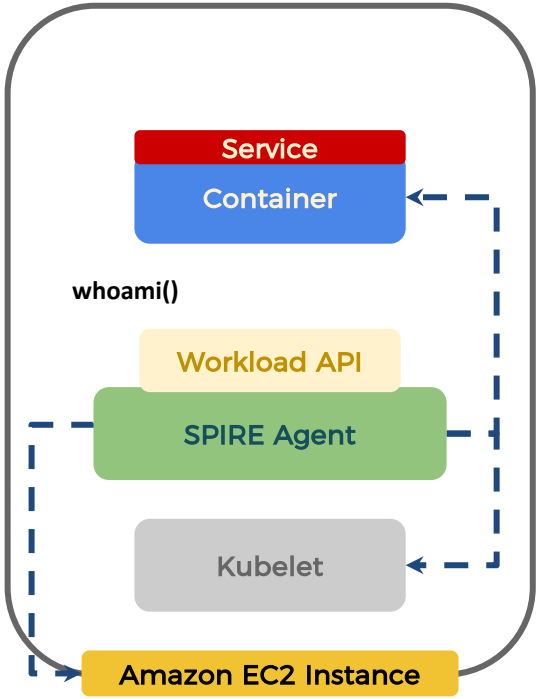
How does attestation work?



When a service launches on a container, it **first** requests its SPIFFE identity from the SPIRE Agent. The service does this via a request to the SPIRE Agent's SPIFFE workload API (exposed as a Unix domain socket).



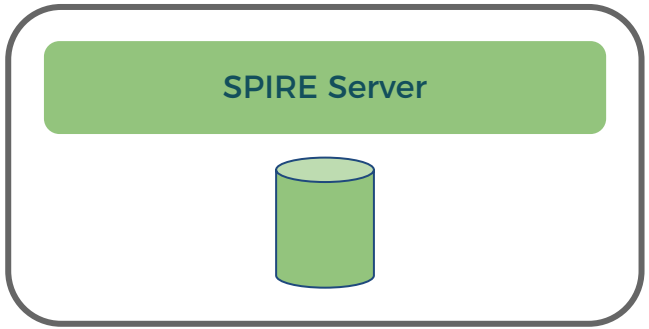
How does attestation work?



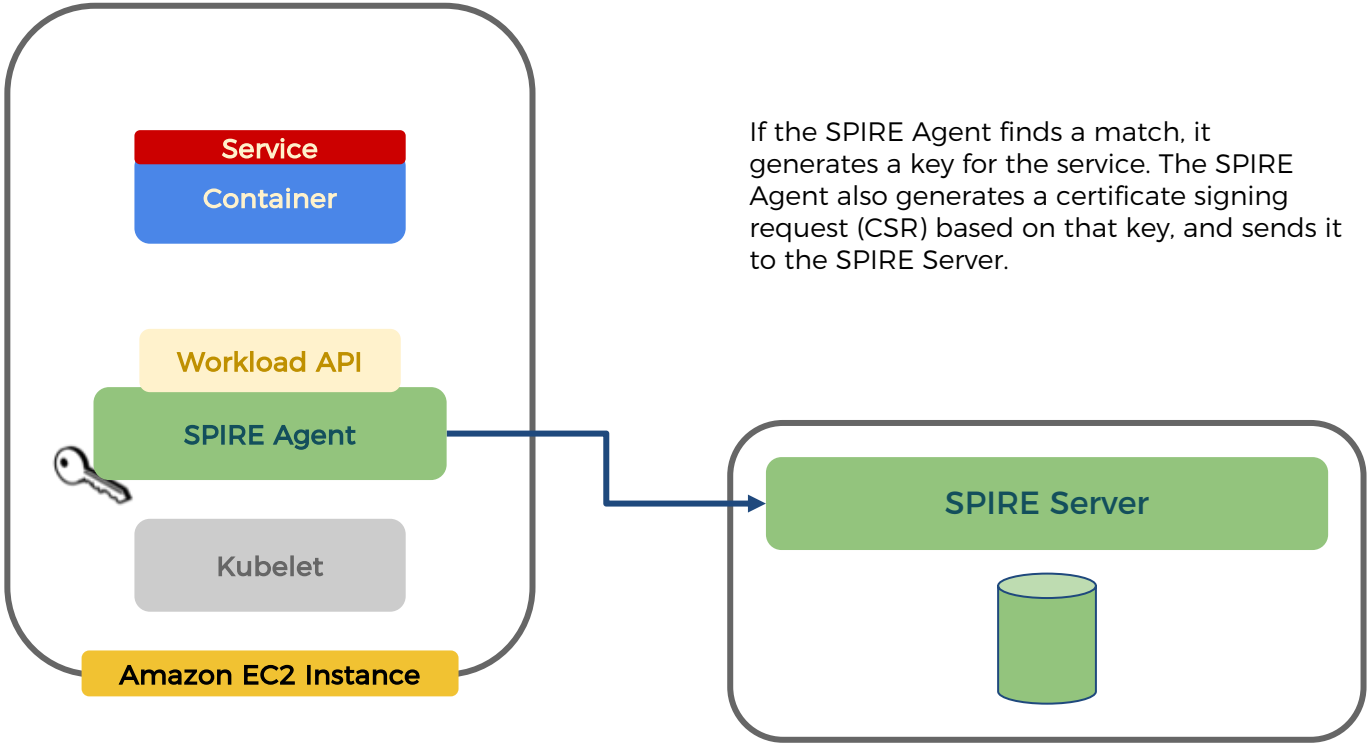
The SPIRE Agent retrieves metadata from the node's kernel about the calling process (service).

The SPIRE Agent also interrogates the kubelet for Kubernetes-specific metadata about the calling process.

The agent correlates the aforementioned against the SPIFFE IDs (and requisite factors) it last retrieved from the SPIRE Server.

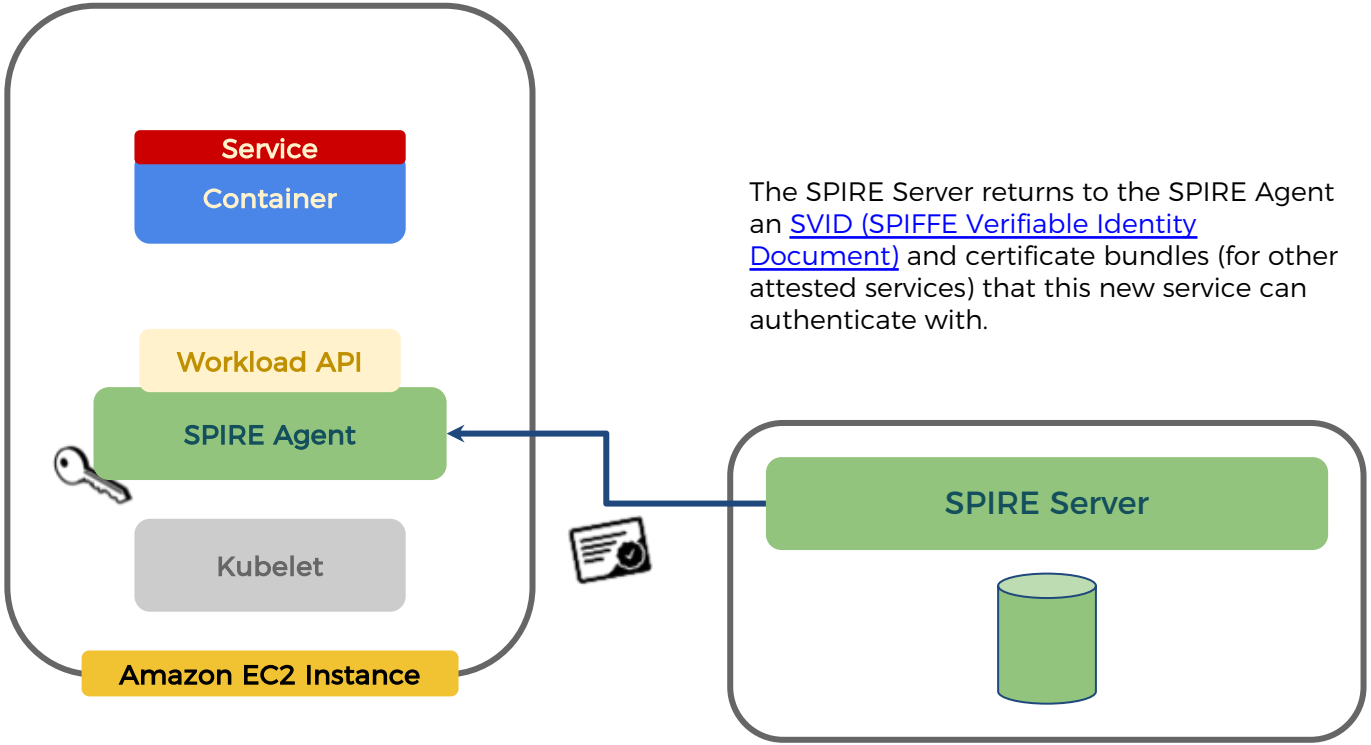


How does attestation work?



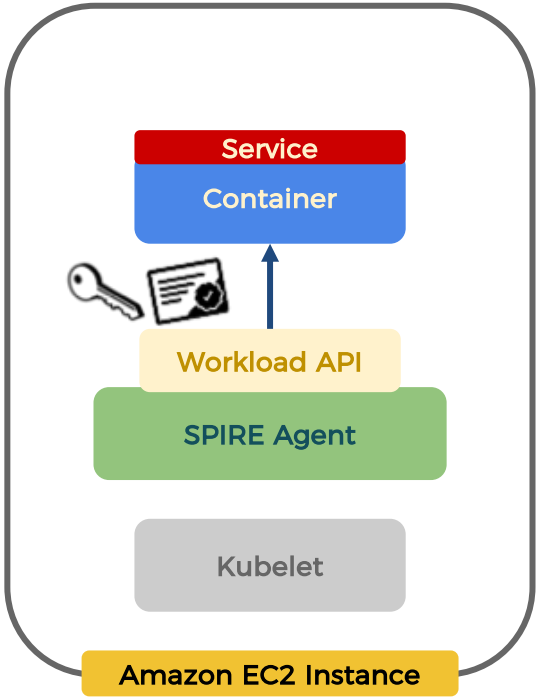
If the SPIRE Agent finds a match, it generates a key for the service. The SPIRE Agent also generates a certificate signing request (CSR) based on that key, and sends it to the SPIRE Server.

How does attestation work?

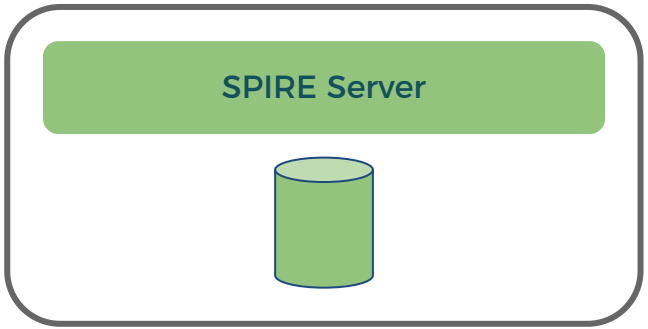


The SPIRE Server returns to the SPIRE Agent an SVID (SPIFFE Verifiable Identity Document) and certificate bundles (for other attested services) that this new service can authenticate with.

How does attestation work?



The SPIRE Agent returns the SVID and certificate bundles to the service.



To summarize, here's what we've done …

- Centrally defined a service "whitelist" along with factors describing each service.
- Attested the fidelity of the infrastructure beneath a running service.
- Attested the fidelity of a new running service.
- Provisioned a globally unique identity and X.509 certificate to the running service.

Now you can start working the problem …

1. Authenticate cloud services with on-premise services.
2. Secure how cloud services access secret stores.
3. Encrypt east-west service traffic within/across trust domains.
4. Reduce our dependency on API gateways to authenticate clients.
5. Deploy an authentication plane that plays nice with middleware.
6. Inject code signing primitives into cross—service authentication.
7. Determine the entitlements of a service chain based on the originating "human."
8. Apply sane and consistent authentication to IoT endpoints & 5G devices.

Questions? Thoughts?

E-Mail: **sunil at scytale dot i o**

Web:

https://www.scytale.io

https://www.spiffe.io