Uber x Security

Tyler Julian, Security Engineer @Uber
Daniel Feldman, Software Engineer @Scytale

May 23, 2019

Uber

Agenda

01 Overview
02 Identity at Uber
03 SPIFFE
04 Case Study
05 Q&A

Identity at Uber

Tyler Julian

About Me

- Authentication
- Distributed Systems
- @Uber
  - Identity & Access Management
  - Trust & Safety
- @21 (acq. by Coinbase)
  - Cryptocurrency Protocol Implementation

Scale

3K+

Unique services.

400K

Running containers to support stateless services.

M

a

Infra

- Deployments in both cloud and on-prem data centers
- RPC with gRPC/HTTP and in-house protocols
- Routing/discovery built in-house
- Orchestration using Mesos, Hadoop, and in-house tools
- Services written in Go, Java, Python, Node.js, and more

Identity Requirements

- Compliance
  - General Data Protection Regulation (GDPR)
  - Sarbanes-Oxley (SOX)

- Trust and Security
  - Reduce assumptions on system behavior (zero trust)
  - Reduce risk of data breach
  - Reduce risk of bad configuration

- Developer Experience
  - Easy to implement and use
  - Integrated with infrastructure

Identity Scope

```
function check(n)
  {   // check if the number n is a prime
    var factor; // if the checked number is not a prime, this
    var c;
    factor = 0;
    // try to divide the checked number by all numbers till it
    for (c=2 ; (c <= Math.sqrt(n)) ; c++)
       {
        if (n%c == 0)    // is n divisible by c ?
           {factor = c;  break}
       }
    return (factor);
  }   // end of check function

function communicate()
  {  // communicate with the user
    var i;       // i is the checked number
    var factor; // if the checked number is not a prime, this
    i = document.primetest.number.value;      // get the check
    // is it a valid input?
    if ((isNaN(i)) || (i <= 0) || (Math.floor(i) != i))
      {alert ("The checked object should be a whole positive n
    else
      {
        factor = check (i);
        if (factor == 0)
          {alert (i + " is a prime")} ;
        else
          {alert (i + " is not a prime, " + i + "=" + factor +
      }
  }     // end of communicate function
```

Users

Riders, drivers, couriers, customer support representatives, managers, engineers, etc.

Machines

Addressable hosts that reside within “Uber” infrastructure.

Workloads

A process that runs application logic for some business purpose.

Workload Identity

- Goal:
  - Uniquely identify a particular program or application

- Control access to:
  - Database credentials
  - Third party API keys
  - Other internal services

- Protect data:
  - Encryption-in-transit
  - Prevent bad actors

SPIFFE

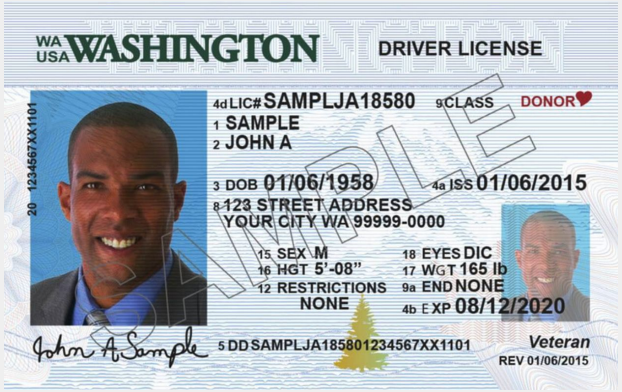
Daniel Feldman

placeholder

Need slides for:

- Intro to SPIFFE (framework for identifying workloads)
- What is an SVID?
- Workload API
- How to actually use Workload API (Proxy?)-- brief
- Selectors (Daniel make this slide)

What is an SVID?



Identity documents are:

Unique

Stable

Verifiable

Attested by a trusted authority

and

github.com/spiffe/spiffe

A set of specifications that cover how a workload should retrieve and use it's identity.

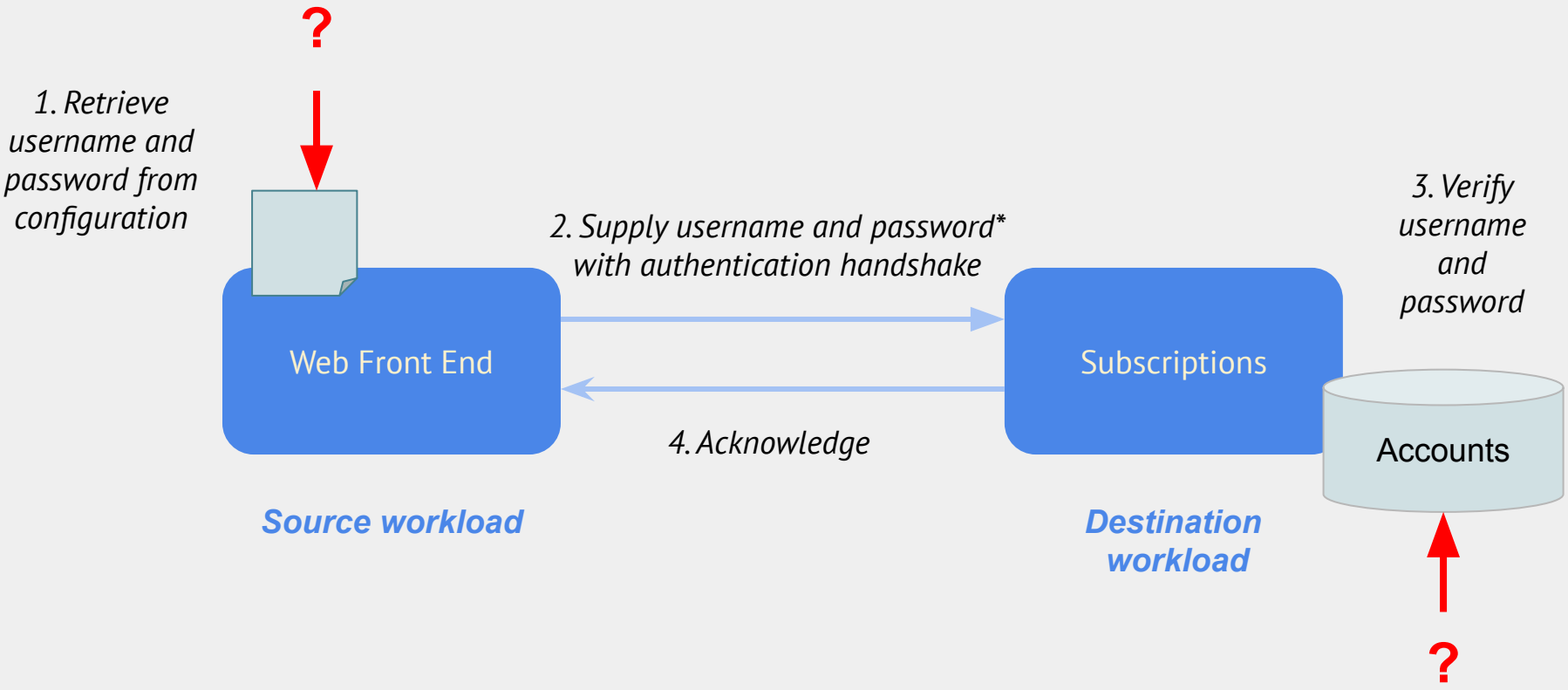
- SPIFFE ID
- SPIFFE Verifiable Identity Documents (SVIDs)
- The SPIFFE Workload API

github.com/spiffe/spire

The SPIFFE Runtime Environment. Open-source software that implements the SPIFFE Workload API for a variety of platforms.

Apache 2.0 license. Independent governance. Highly extensible through plug-ins.

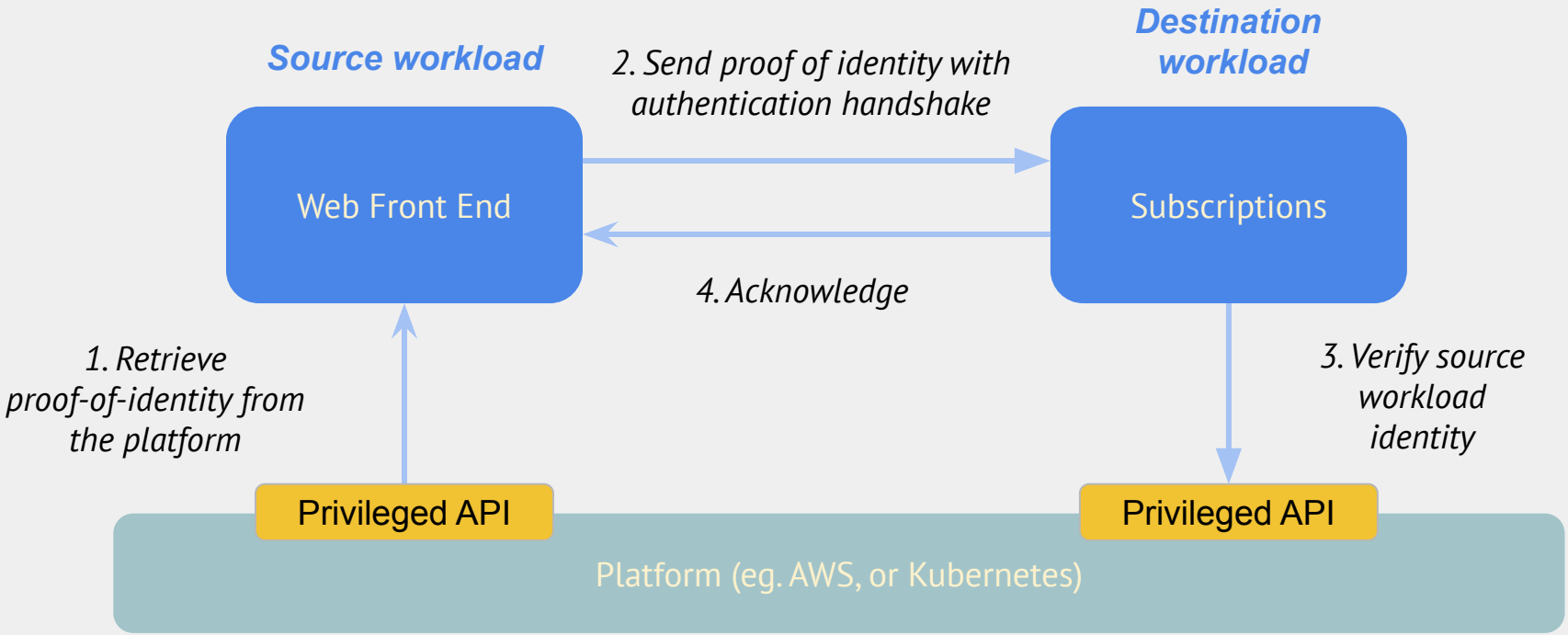
Workload authentication



* Or key/secret, signed nonce etc.

Platform mediated identity

Eg. AWS IAM, Kubernetes Service Accounts



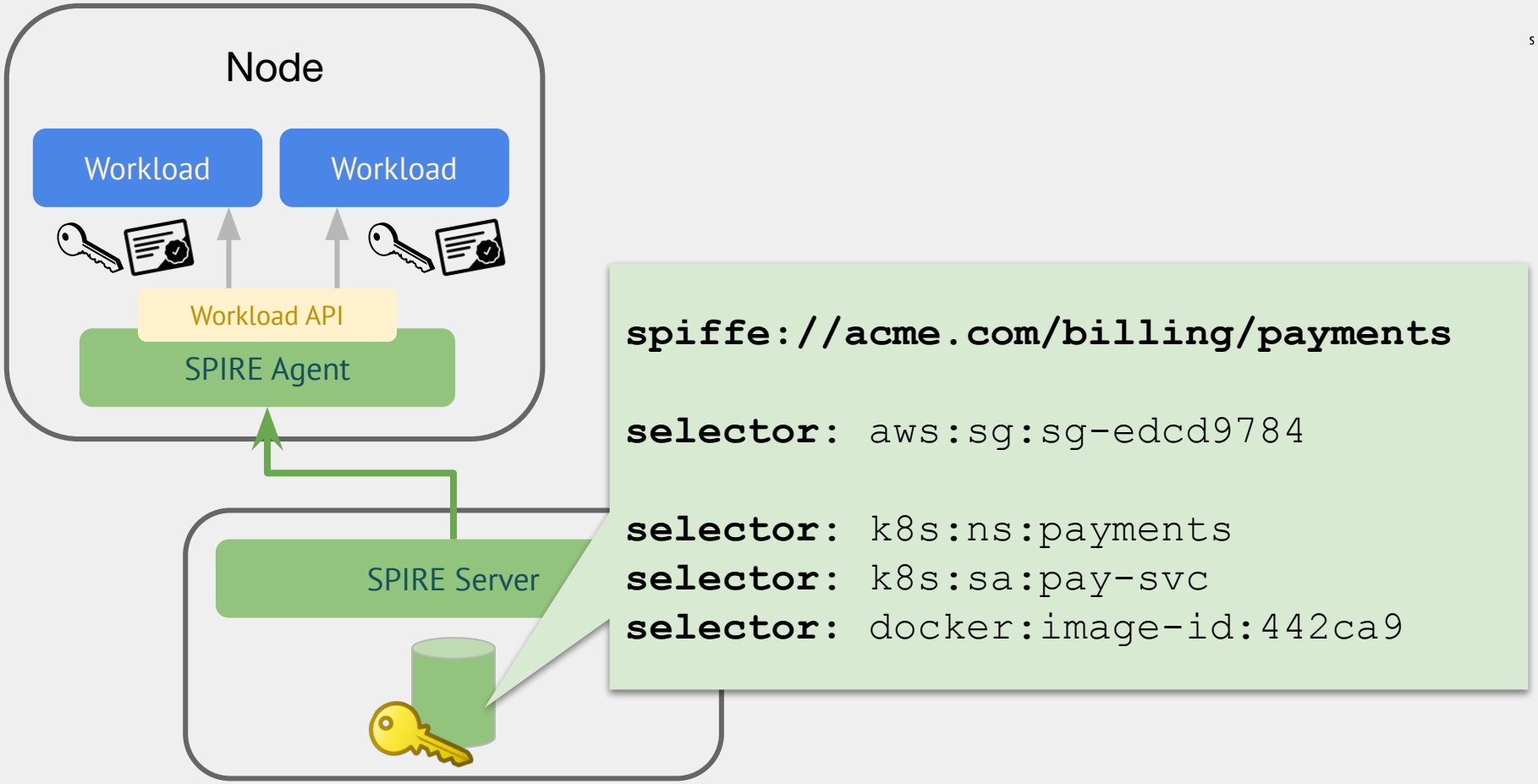
What is an SVID?

spiffe://acme.com/billing/payments

A SPIFFE ID

X.509-SVID describes exactly how to encode a SPIFFE ID in an X.509 certificate

JWT-SVID describes exactly how to encode a SPIFFE ID in an JWT bearer token


Node
Workload
Workload
Workload API
SPIRE Agent
SPIRE Server
spiffe://acme.com/billing/payments
selector: aws:sg:sg-edcd9784
selector: k8s:ns:payments
selector: k8s:sa:pay-svc
selector: docker:image-id:442ca9

Design Goals

SCYTALE

- **Application identity driven.** By building a security model rooted in a strong assertion of application identity, policies and practices become application- and business unit- oriented rather than infrastructure-oriented.

- **Easily adoptable**. Users should be able to leverage Emissary with little or no code change. The system should work well in dynamically orchestrated containerized environments.

- **Federatable**. It should be possible to use these identity mechanisms across business units and even organizations.

- **Reliable**. The single points of failures in the system should be minimized and the system should degrade gracefully when any single point of failure is down.

- **Cloud and Container Ready**. It should be possible to safely extend trust to entities running on to third party cloud providers such as Amazon Web Services and Microsoft Azure, and container orchestrators such as Cloud Foundry and Kubernetes.

Security Goals

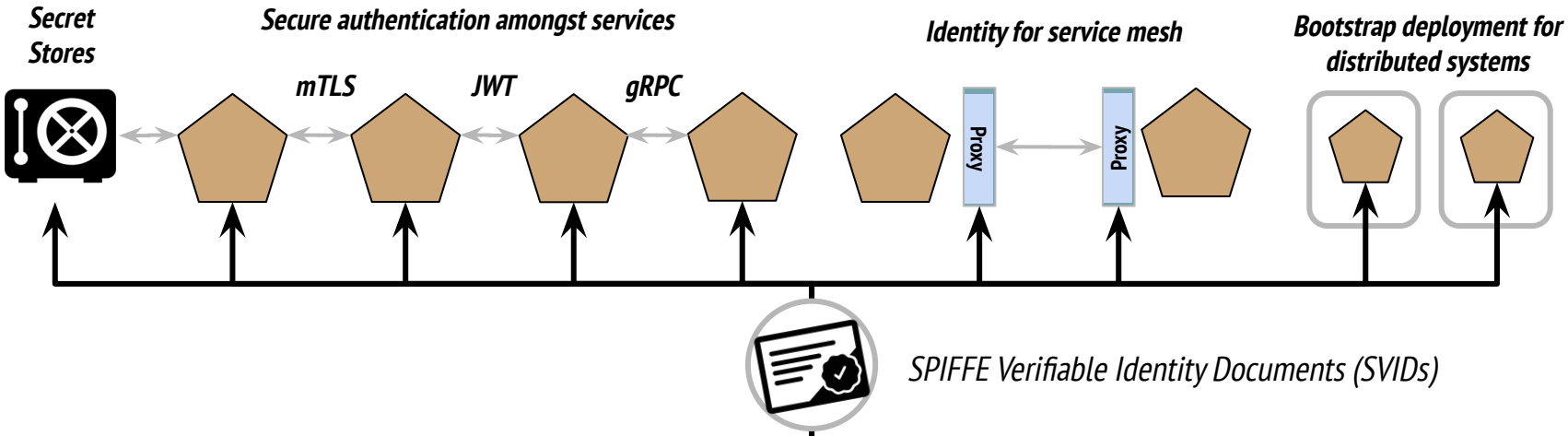
SCYTALE

- **Fully automated and policy driven**. Existing identity (particularly PKI) infrastructure is both complex and often requires “human trust”, which weakens delivery. Emissary is fully automated and should minimize manual key distribution.

- **Minimal Knowledge**. A compromised machine should only expose any secrets for workloads that happen to be running on that machine.

- **Reliable**. The single points of failures in the system should be minimized and the system should degrade gracefully when any SPOF is down. All “steady state” operations shouldn’t have requirements off of a specific node.

- **Scoped trust roots**. There should be no hardcoded, global trust roots as we see in the web browser world.



SPIFFE Workload API

SPIRE

Cloud platform attestation plug-ins

OS attestation plug-ins

Scheduler and PaaS attestation plug-ins

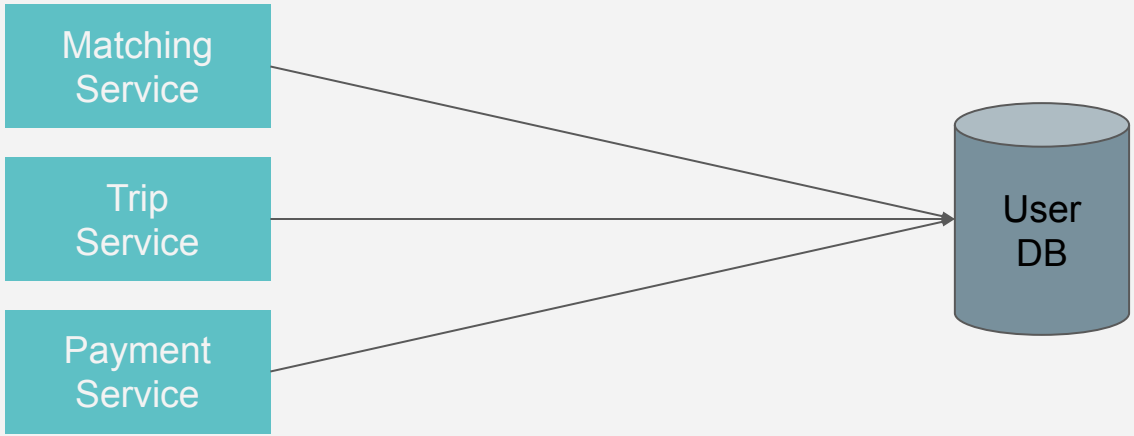
HSM, TPM, Kerberos attestation plug-ins

CA and secret store plug-ins

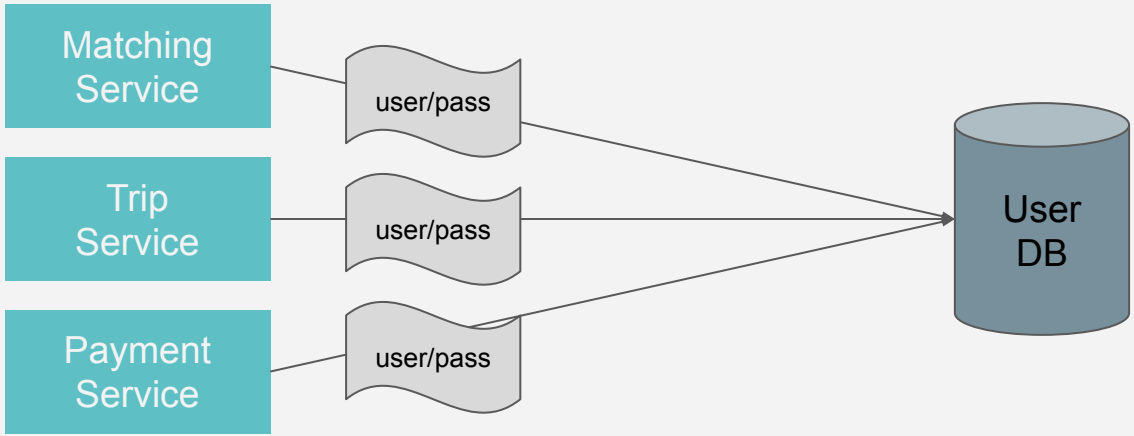
Case Study

Authentication in a Microservice Architecture

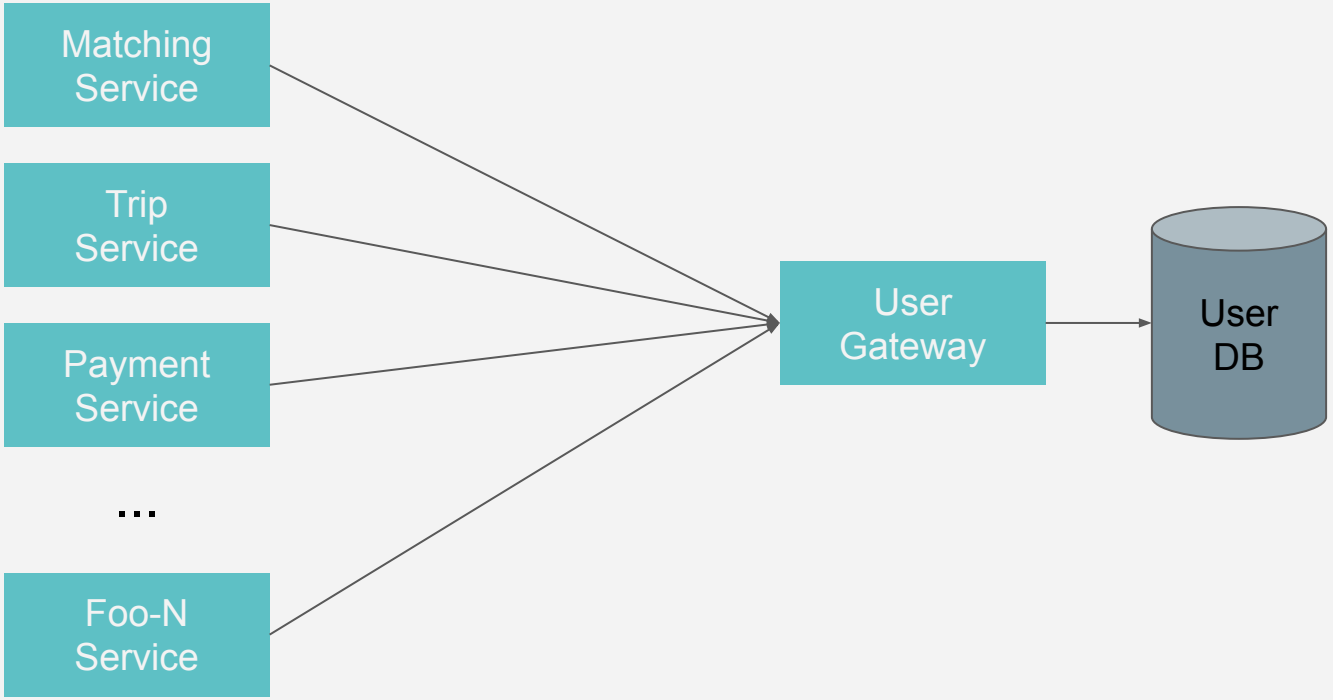
Early: Service to DB (Direct Data Access)



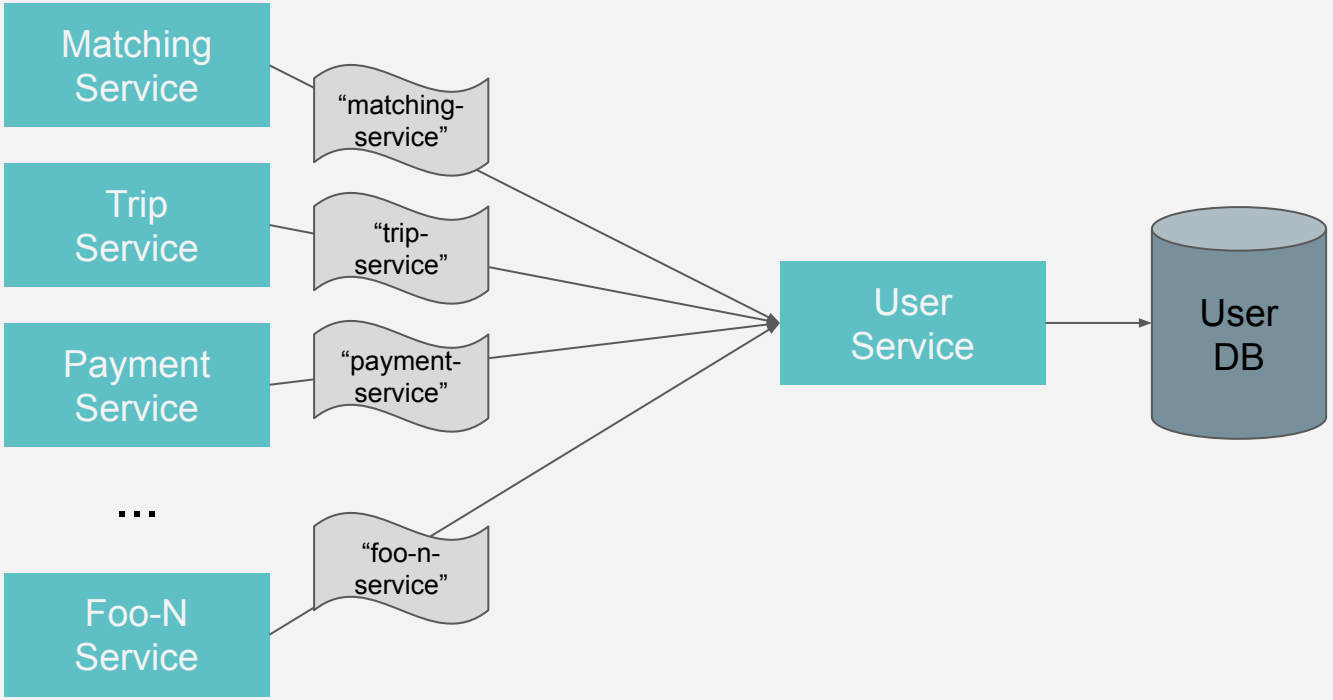
Early: Service to DB (Direct Data Access)



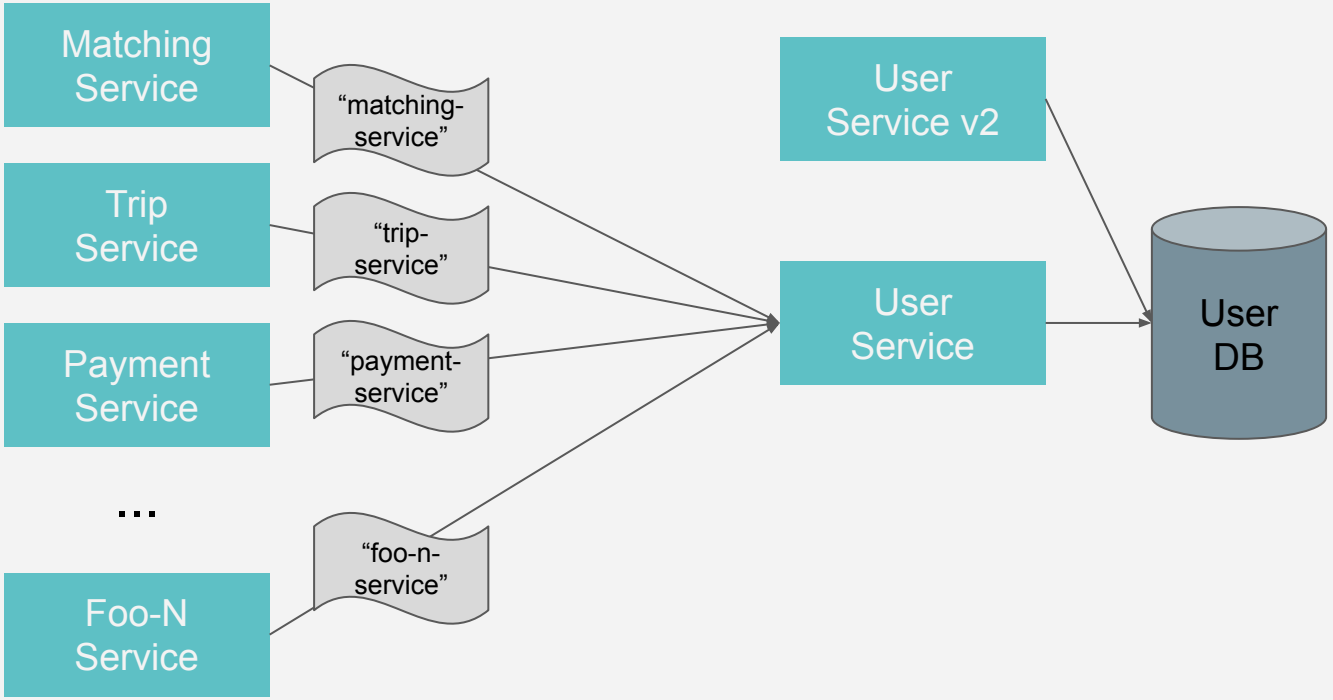
Growth: Service to Gateway (Proxied Data Access)



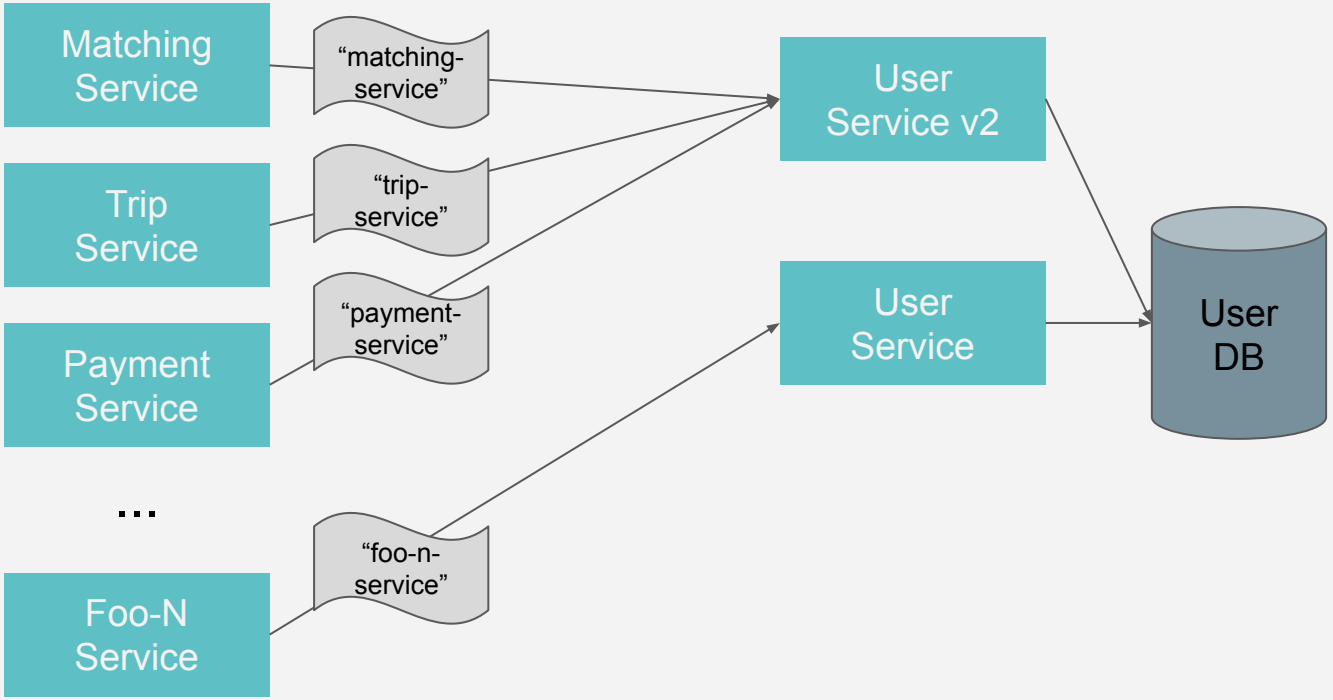
Growth: Service to Gateway (Proxied Data Access)



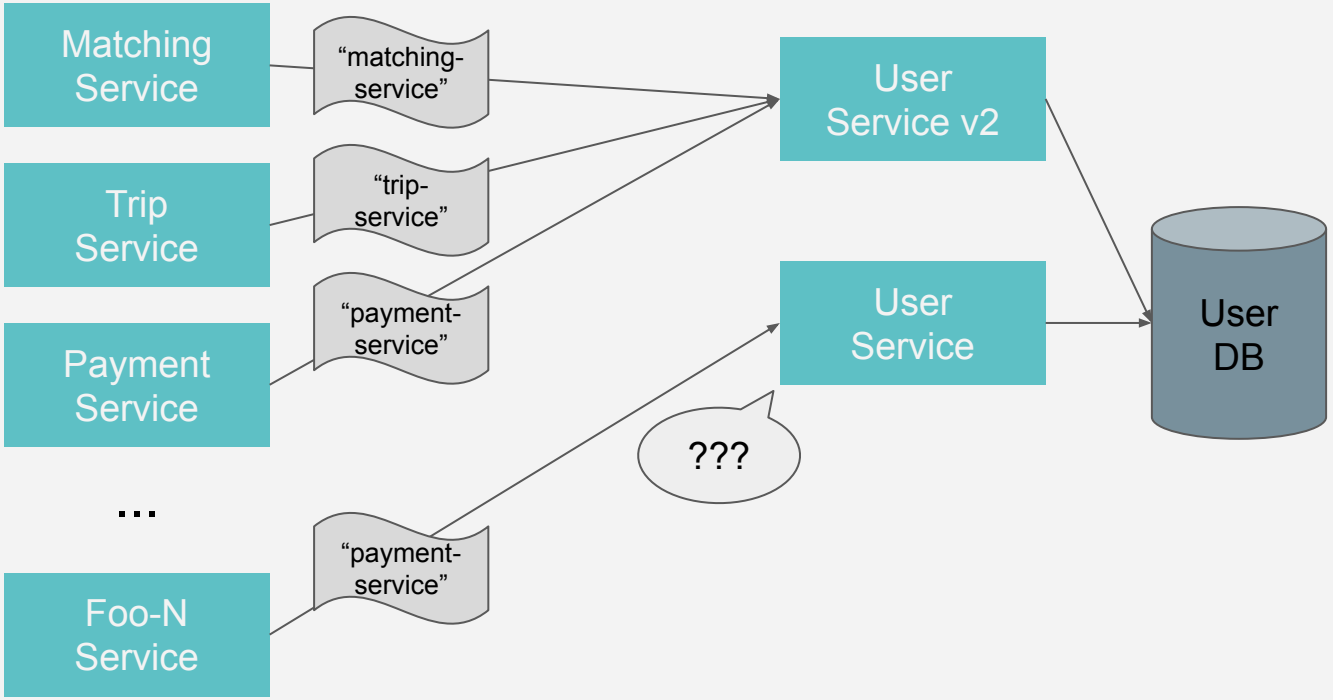
Late: Service to Gateway (Migration)



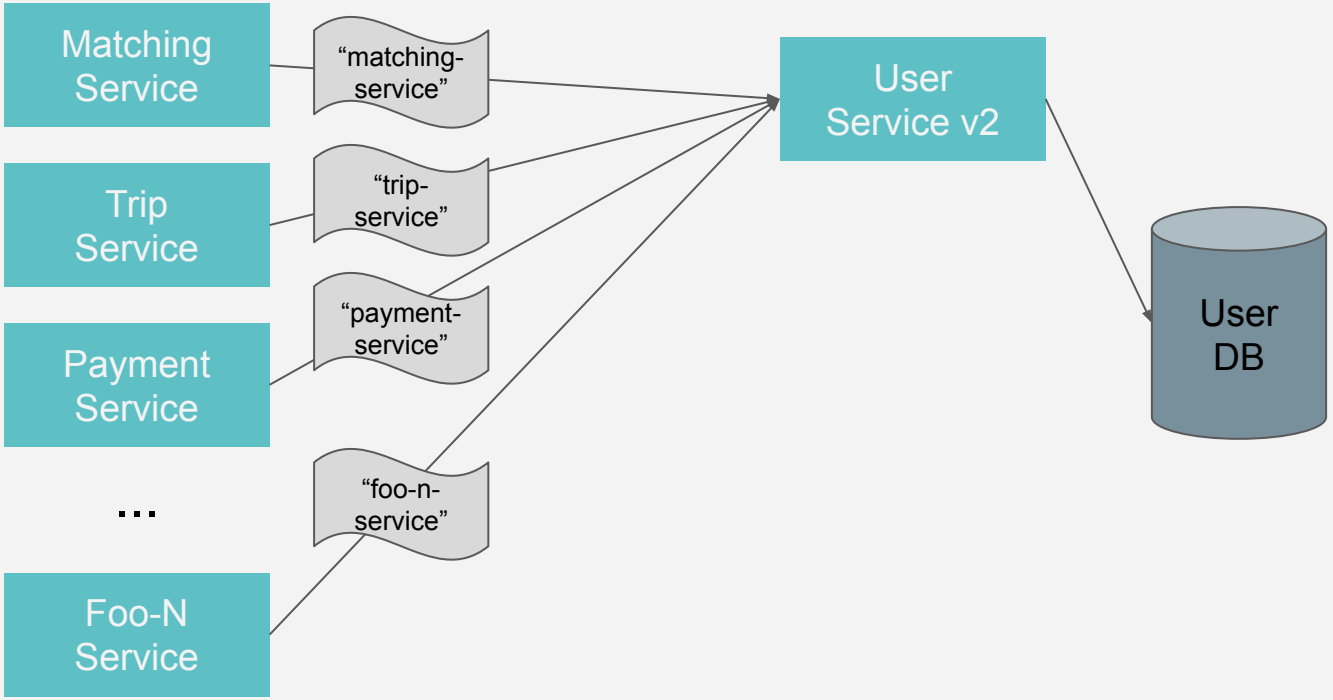
Late: Service to Gateway (Migration)



Late: Service to Gateway (Migration)



Late: Service to Gateway (Migration)



Implementation

- Talk about libraries/sidecars, benefits of encapsulating from application logic, mTLS and JWTs

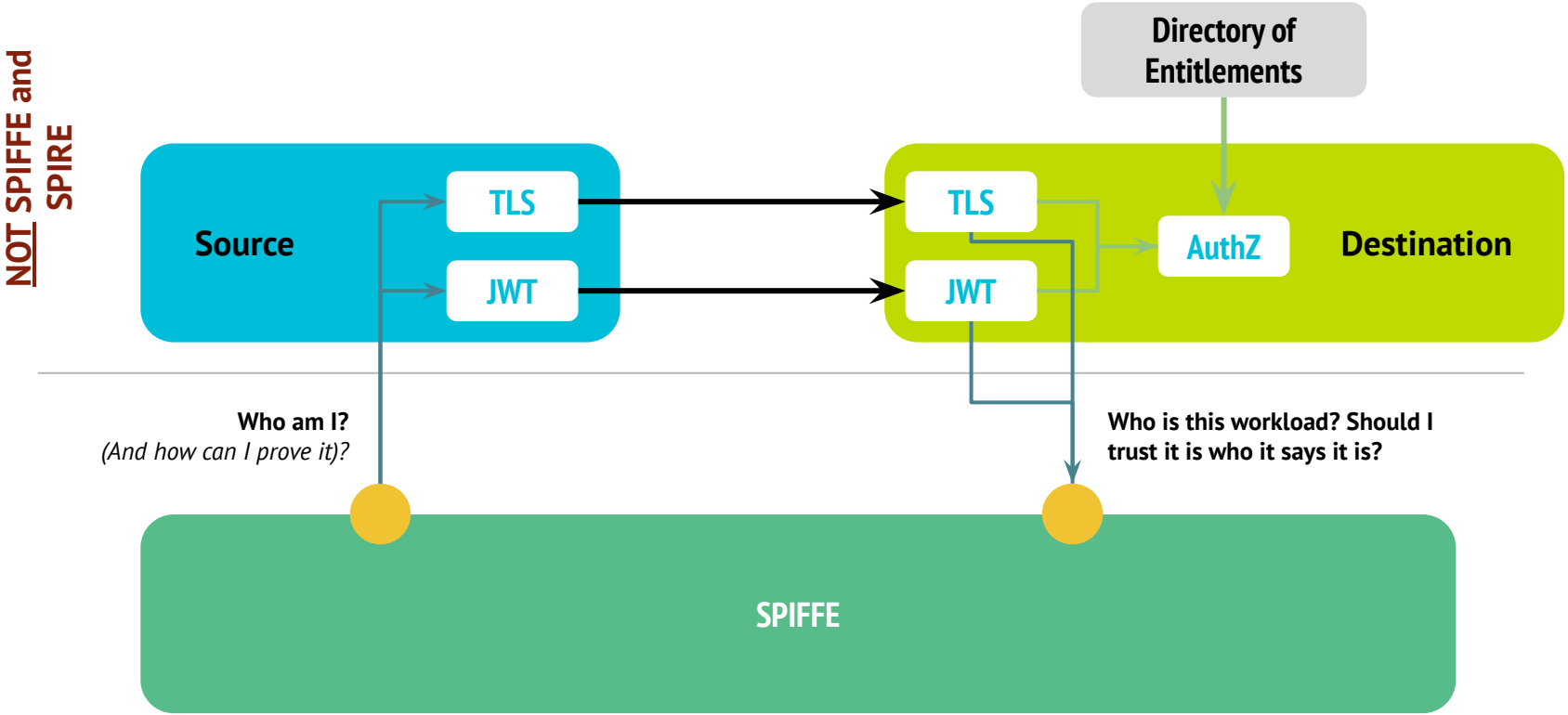
Q&A


KubeCon
CloudNativeCon
North America 2018

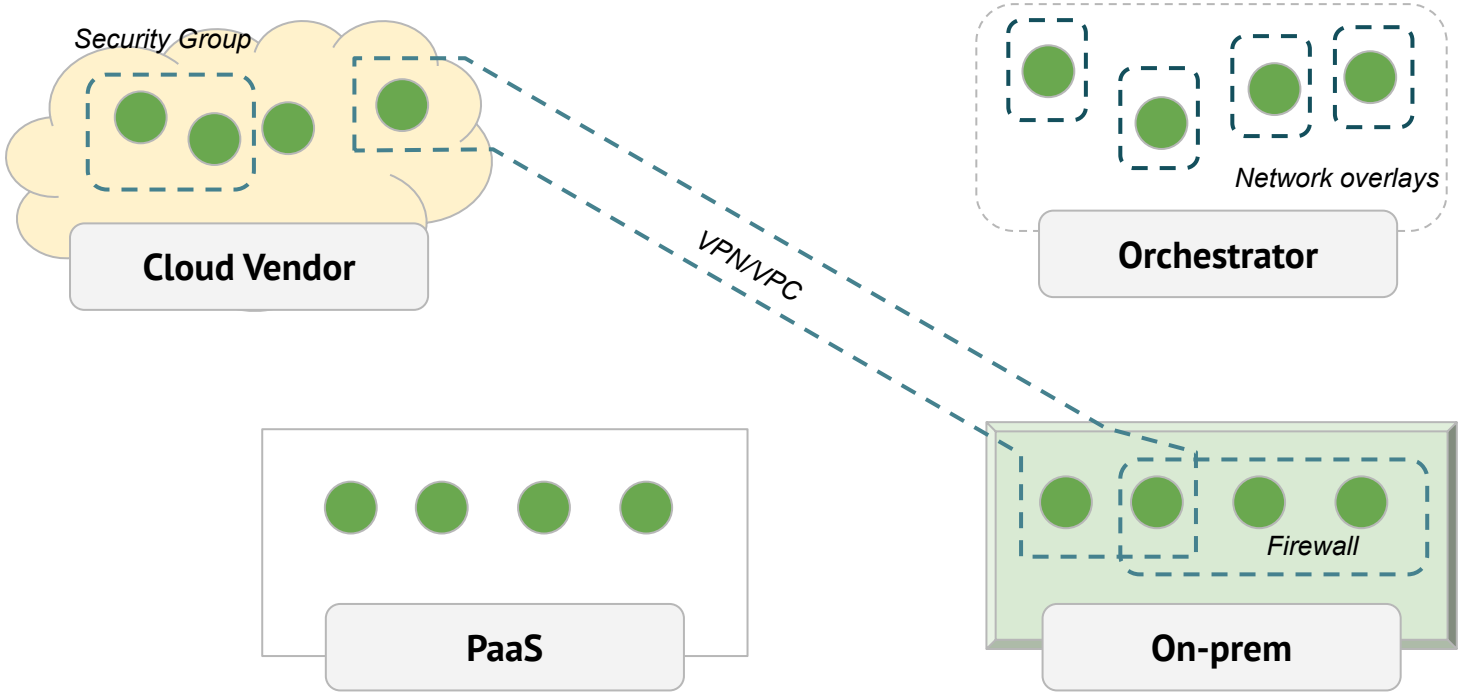

KubeCon
CloudNativeCon
North America 2018

BACKUP SLIDES

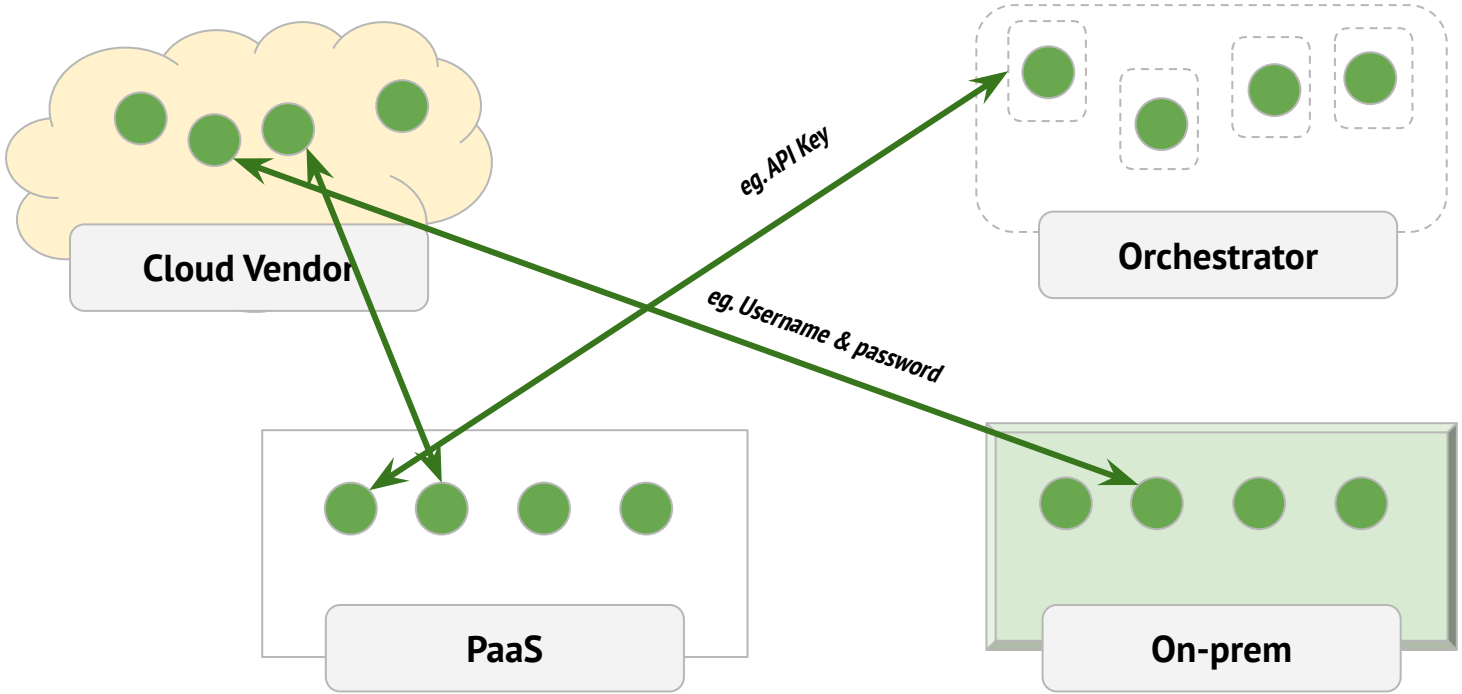
Identity is the *basis for* AuthN and AuthZ



Workload identity? Use the network?

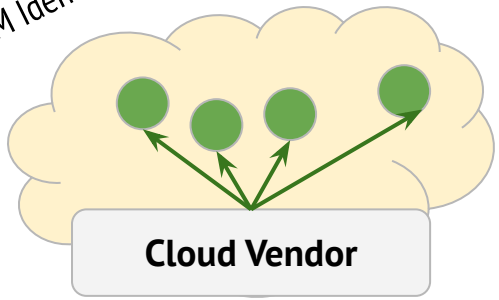


Workload identity? Shared secrets?

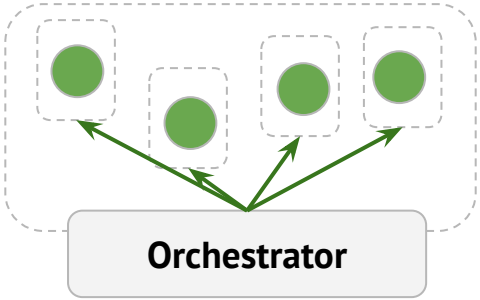


Workload identity? Ask my platform?

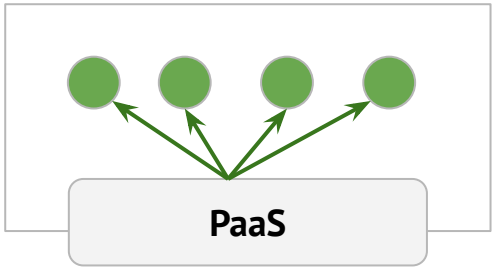
eg. IAM Identities



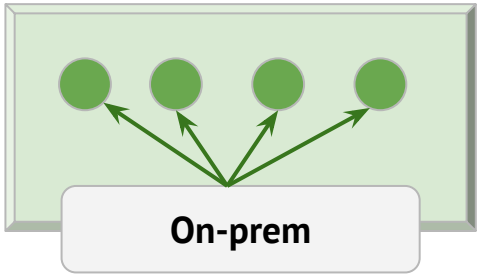
eg. Service accounts



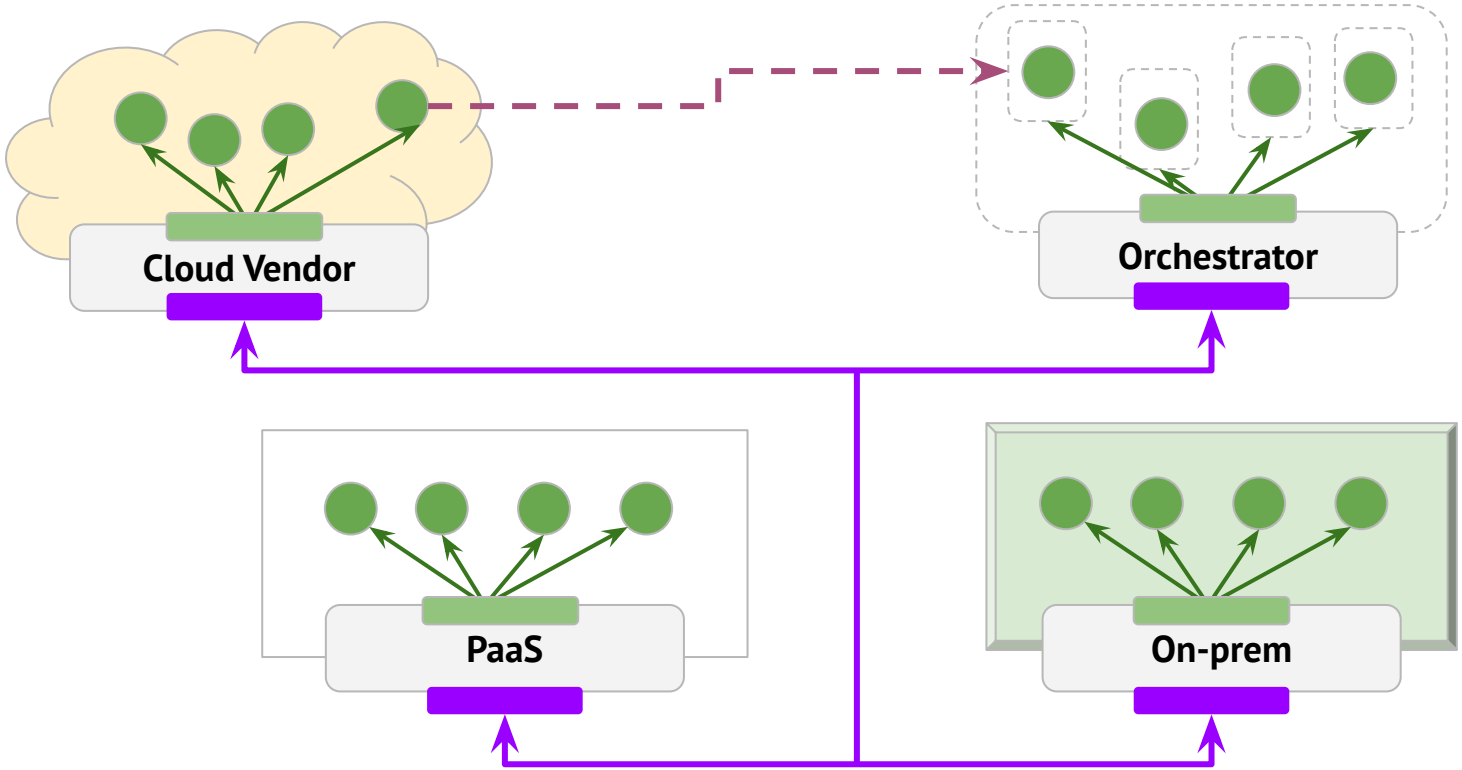
eg. Application IDs



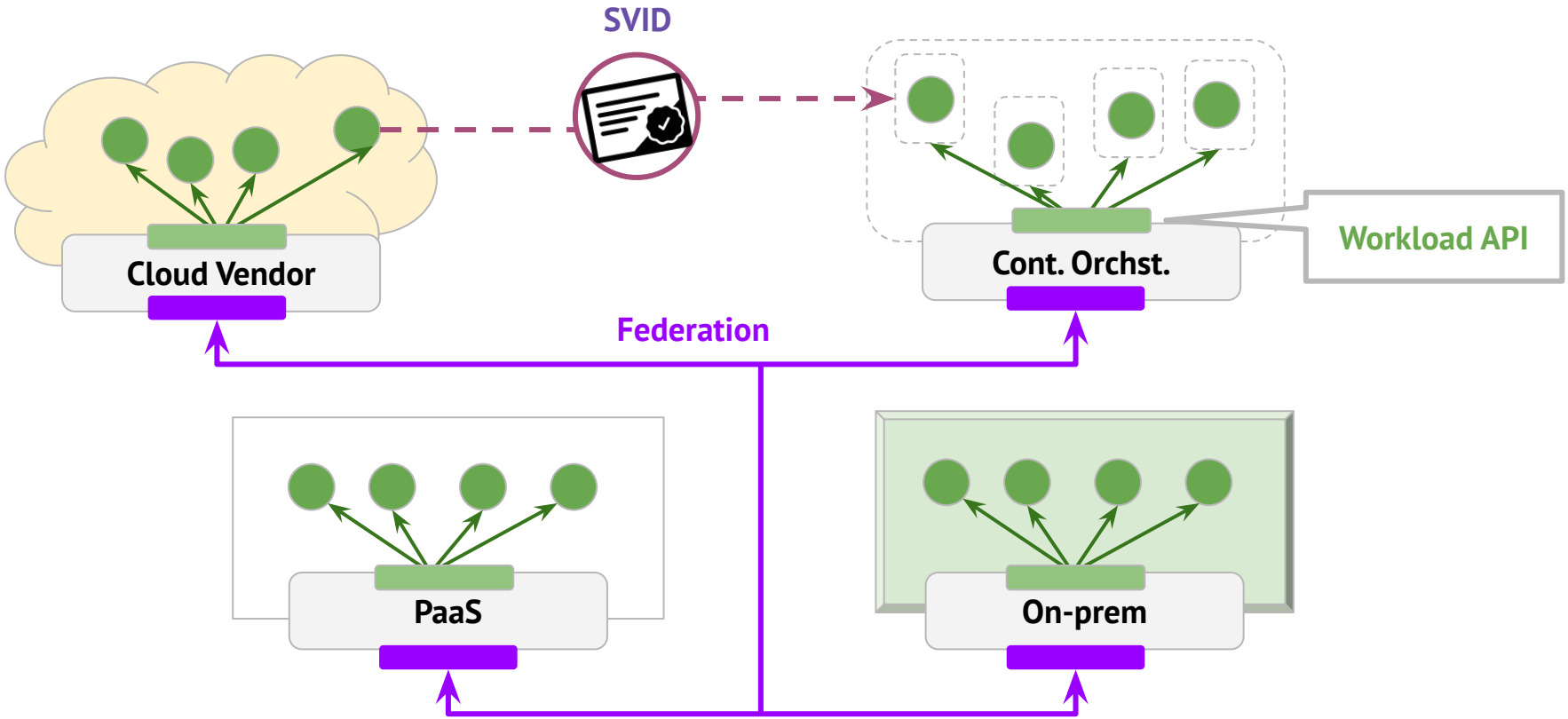
eg. Kerberos Keytabs



SPIFFE: Federated, platform-mediated, vendor neutral identity



SPIFFE: Federated, platform-mediated, vendor neutral identity



SPIFFE Issuers

SPIRE
(Full implementation)

HashiCorp Consul Connect
(Partial implementation)

Istio Citadel
(Partial implementation)

SPIFFE Consumers

HashiCorp Vault
Secret store

Knox
Secret store

Ghostunnel
Proxy

nginx
Web server and proxy

Envoy
Proxy

Your code
Using libraries

Today

A short history of SPIFFE

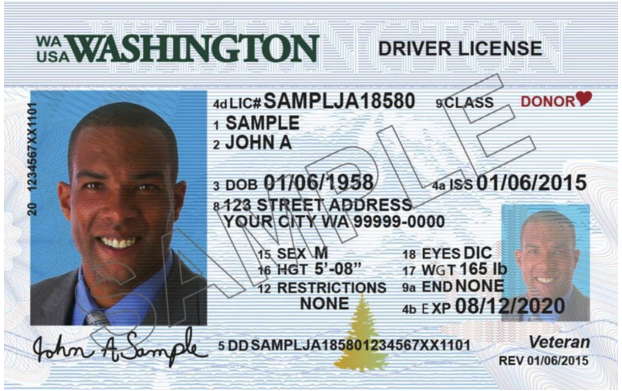
What SPIFFE solves for

SVIDs, Workload API and Federation

How to use SPIFFE

What's Next?

What is an SVID?



Identity documents are:

Unique

Static

Verifiable

Attested by a trusted authority

What is an SVID?

spiffe://acme.com/billing/payments

A SPIFFE ID

X.509-SVID describes exactly how to encode a SPIFFE ID in an X.509 certificate

JWT

JWT-SVID describes exactly how to encode a SPIFFE ID in an JWT bearer token

SPIFFE Verifiable Identity Document

SPIFFE Verifiable Identity Document (SVID)

Trust Bundle

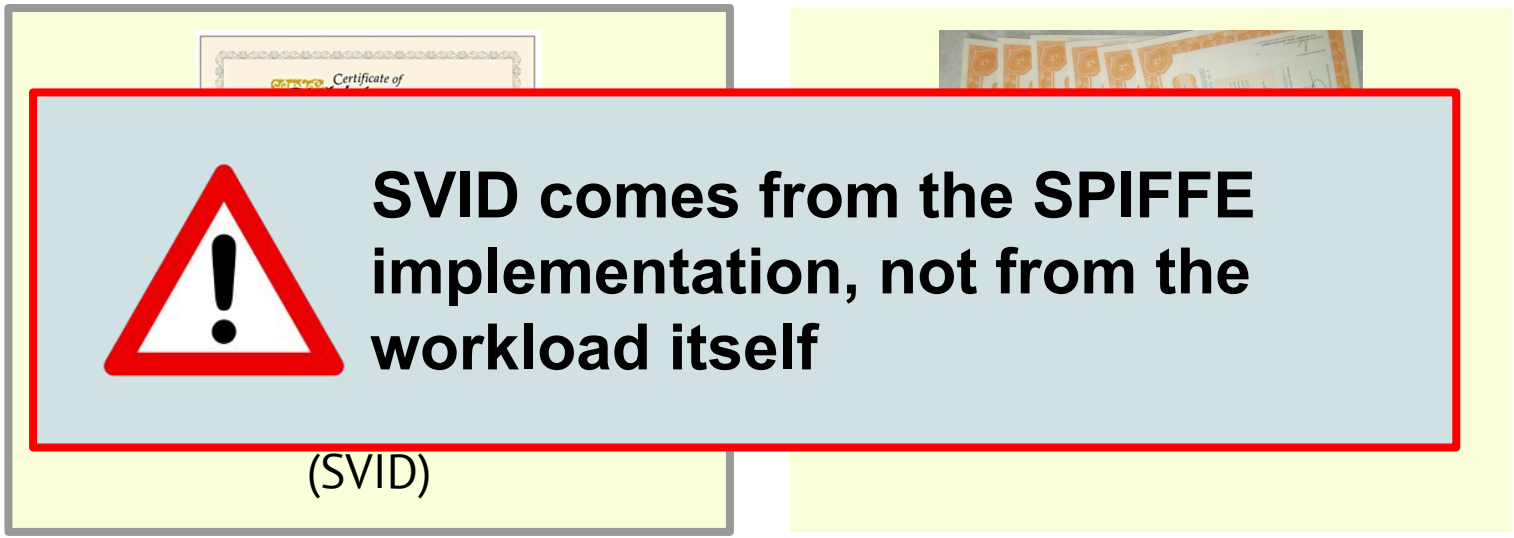
SPIFFE Verifiable Identity Document

SPIFFE
Verifiable Identity Document
(SVID)

Trust Bundle

spiffe://acme.com/billing/payments

SPIFFE Verifiable Identity Document

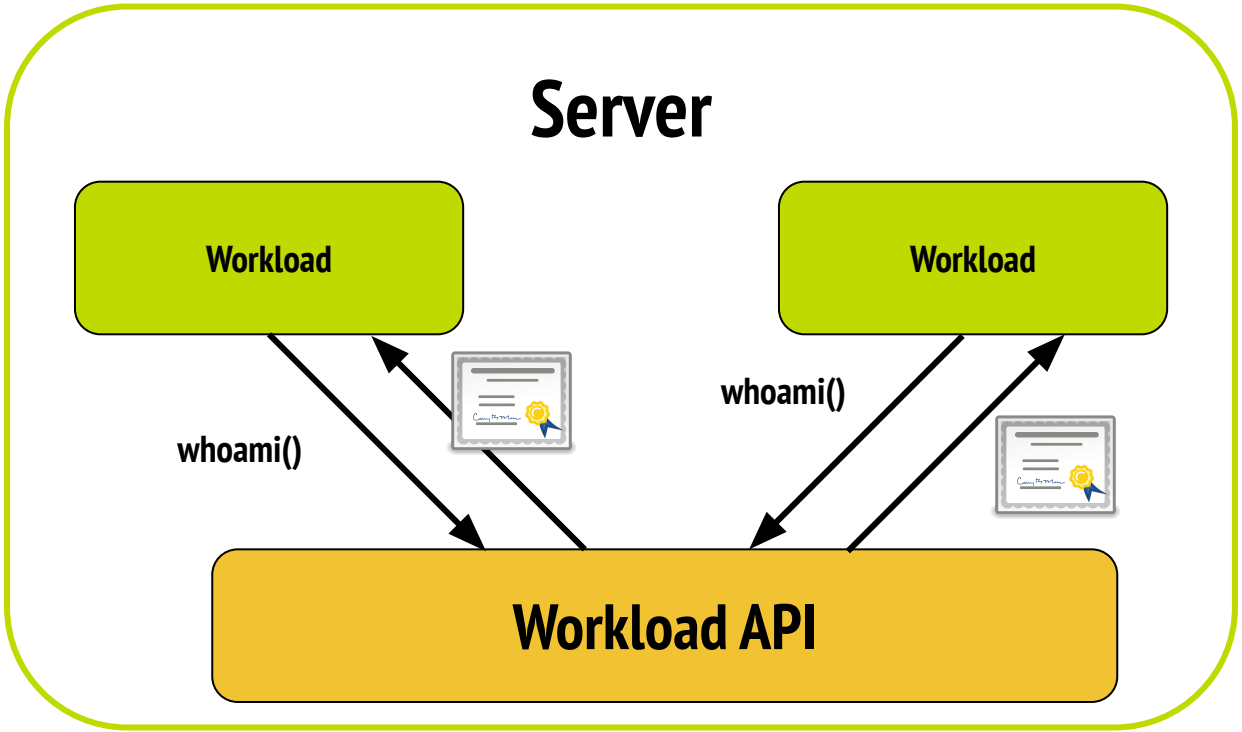


SPIFFE Verifiable Identity Document

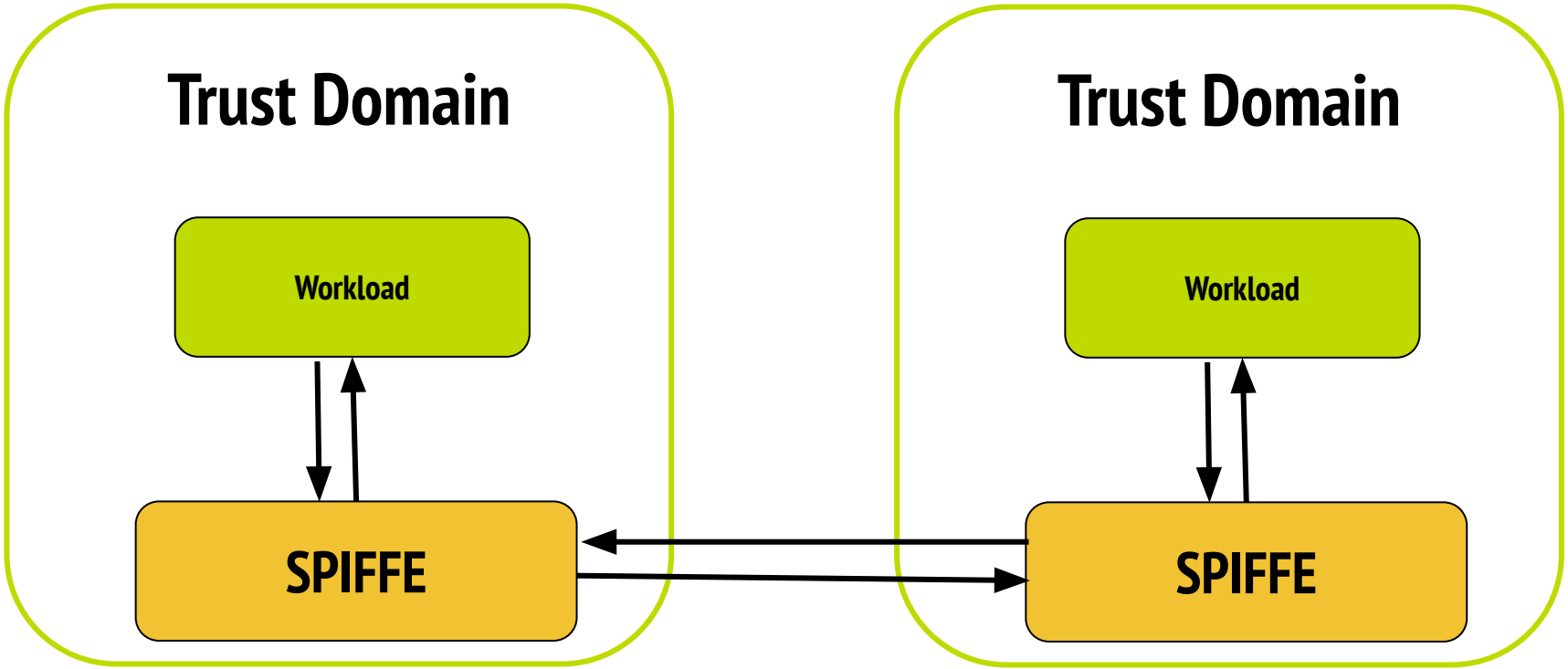
SPIFFE
Verifiable Identity Document
(SVID)

Trust Bundle

SPIFFE Workload API



SPIFFE Federation API



Today

A short history of SPIFFE

What SPIFFE solves for

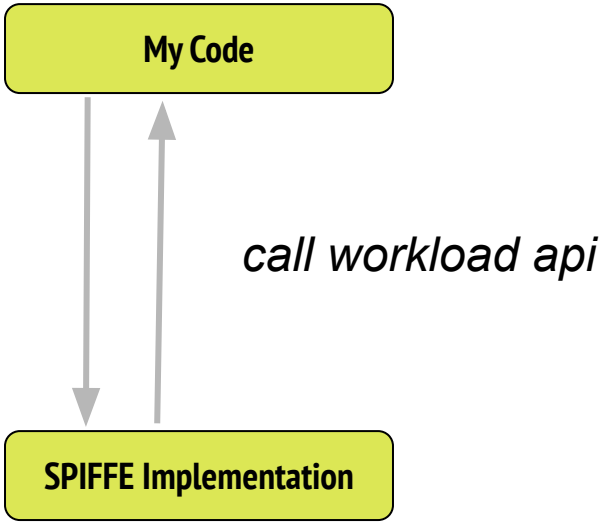
SVIDs, Workload API and Federation

How to use SPIFFE

What's Next?

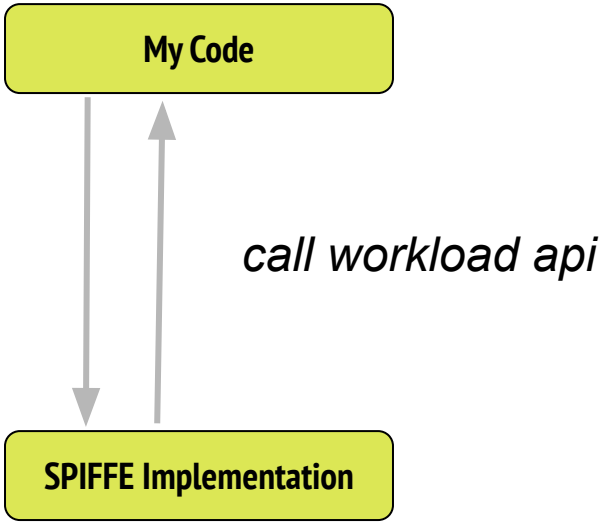
How do I get an SVID?

- My Code
- Library
- Proxy
- spiffe-helper
- Service Mesh



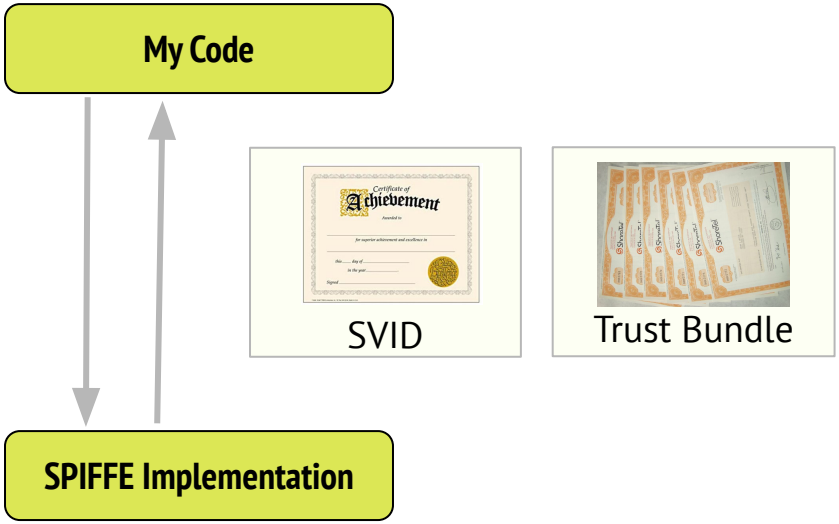
How do I get an SVID?

- My Code
- Library
- Proxy
- spiffe-helper
- Service Mesh



How do I get an SVID?

- My Code
- Library
- Proxy
- spiffe-helper
- Service Mesh



How do I get an SVID?

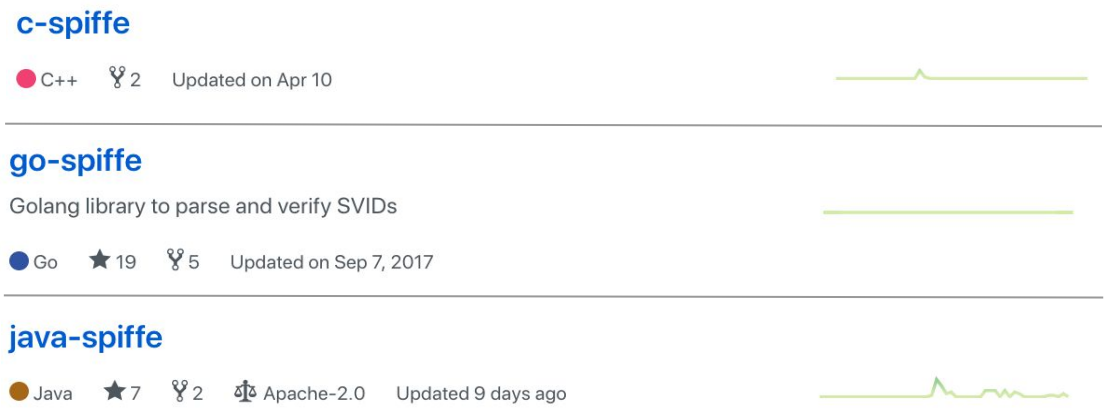
My Code

Library

Proxy

spiffe-helper

Service Mesh



How do I get an SVID?

My Code

Library

Proxy

Service Mesh

c-spiffe

C++ 2 Updated on Apr 10

go-spiffe

Golang library to parse and verify SVIDs

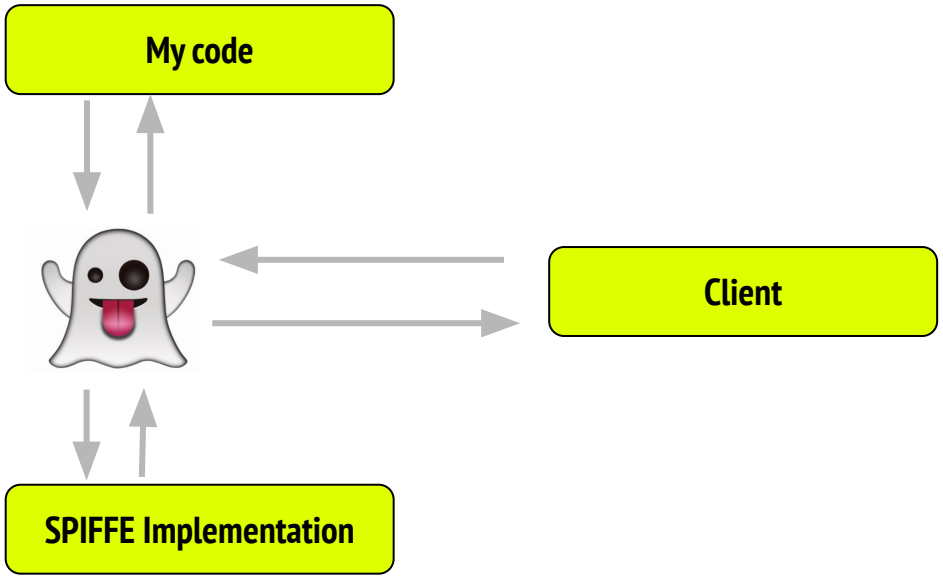
Go 19 5 Updated on Sep 7, 2017

java-spiffe

```
<connection-property name="url">
    jdbc:postgresql://backend:8443/tasks_service?socketFactory=spiffe.provider.SpiffeSocketFactory
</connection-property>
```

How do I get an SVID?

- My Code
- Library
- **Proxy**
- spiffe-helper
- Service Mesh



How do I get an SVID?

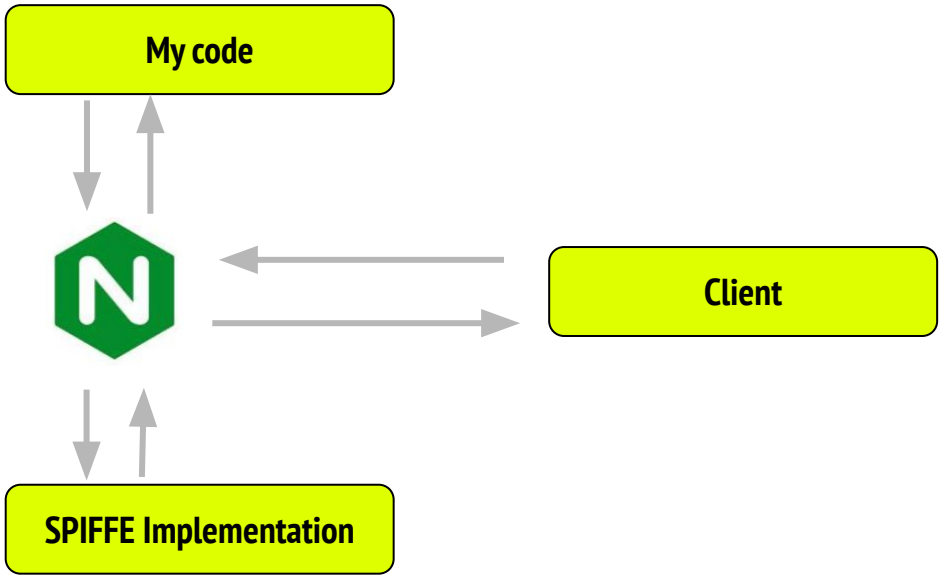
My Code

Library

Proxy

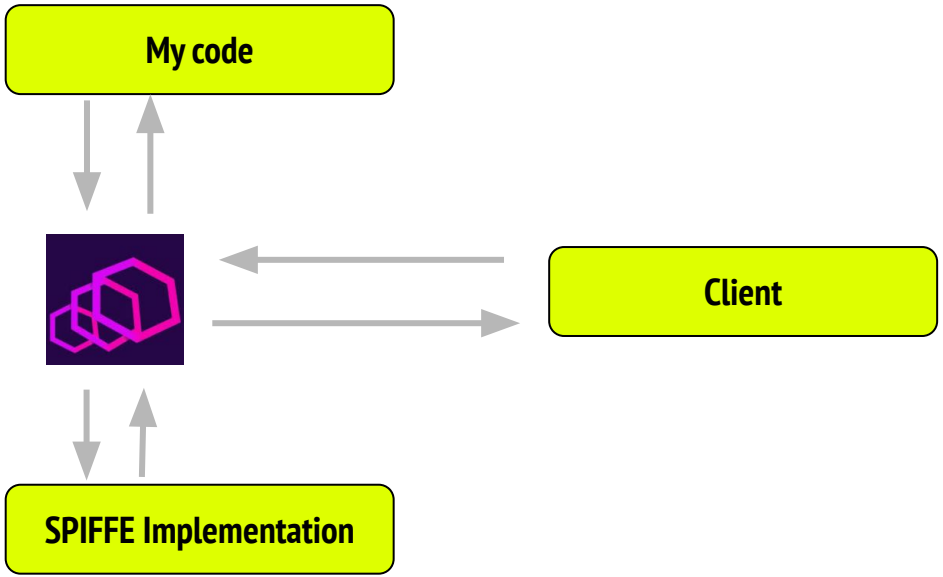
spiffe-helper

Service Mesh



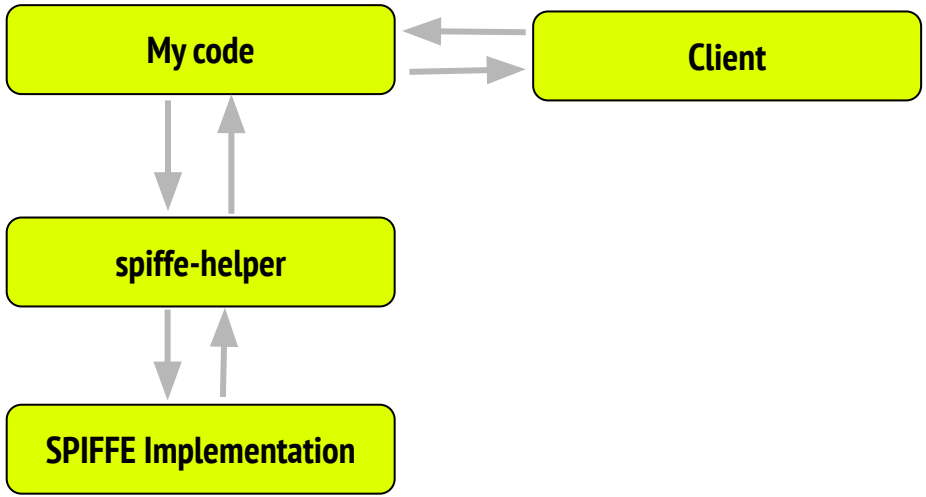
How do I get an SVID?

- My Code
- Library
- **Proxy**
- spiffe-helper
- Service Mesh



How do I get an SVID?

- My Code
- Library
- Proxy
- **spiffe-helper**
- Service Mesh



The Big Idea:

- TLS works really well for encryption
- The problem is creating, managing, and rotating certificates at scale
- We can come up with a standard way to do it automatically

How to do it:

- Restricted form of X.509 certificates:
  - Only certain fields allowed
  - SPIFFE ID in the “SAN” field
  - DN field is not used
- Workload API for services to retrieve their own certificate

Implementations:

SPIFFE IDs

spiffe://acme.com/workload/workload1

Trust Domain

Workload ID

SPIFFE IDs

spiffe://acme.com/workload/workload1

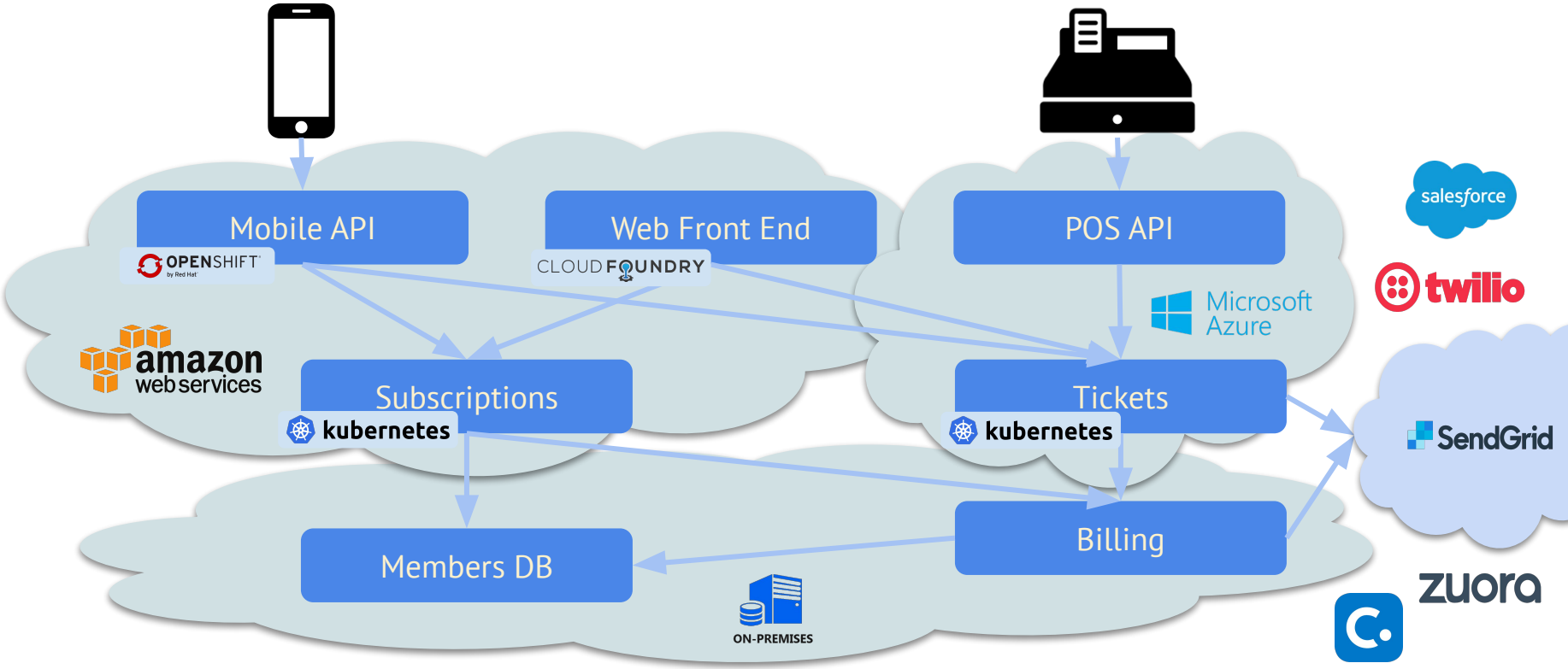
Trust Domain

Workload ID

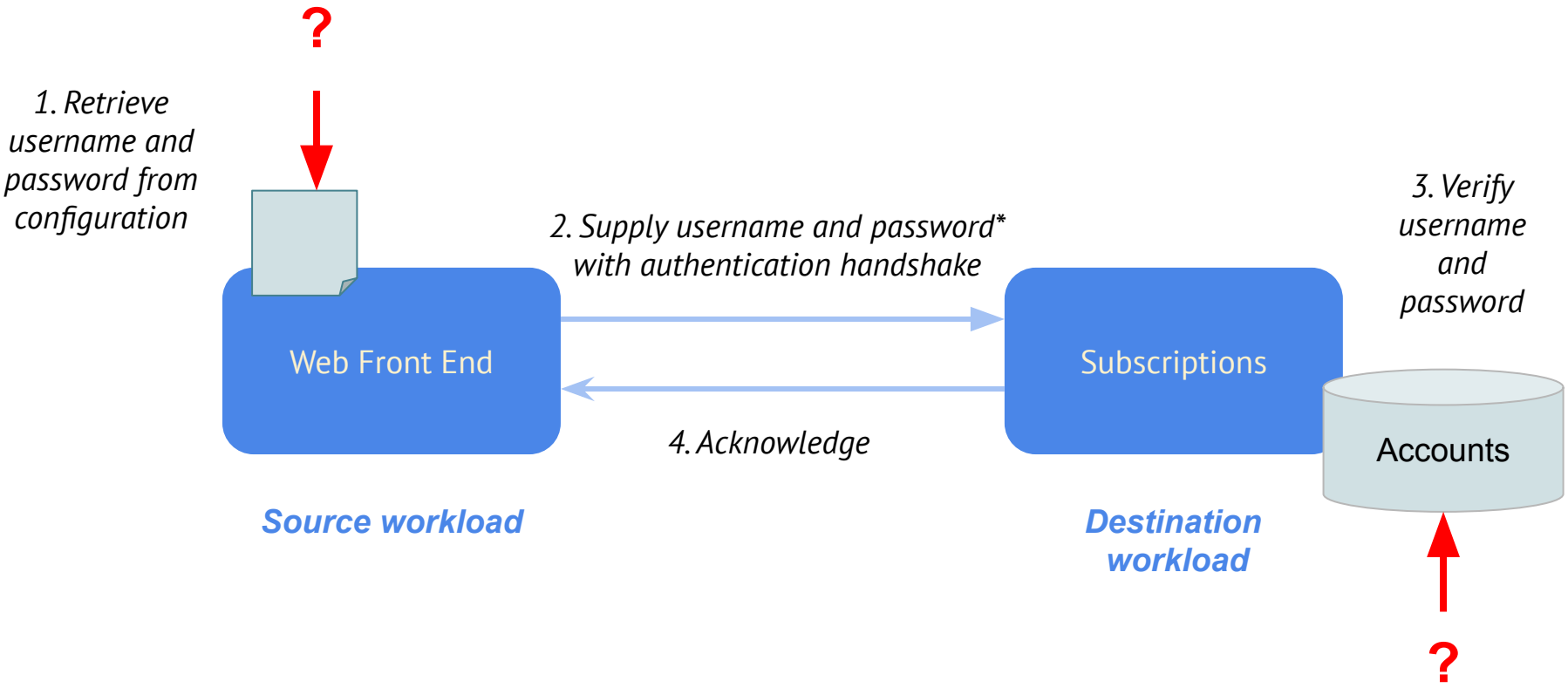
Stored inside the X.509 certificate

Selectors

- What workload am I?
  - Comes from the platform, not the workload itself
  - “Attestation”
- The mappings from platform properties to SPIFFE IDs are selectors


Mobile API
Web Front End
POS API
OPENSHIFT
by Red Hat
CLOUD FOUNDRY
Microsoft
Azure
salesforce
twilio
amazon
web services
Subscriptions
kubernetes
Tickets
kubernetes
SendGrid
Billing
Members DB
ON-PREMISES
zuora

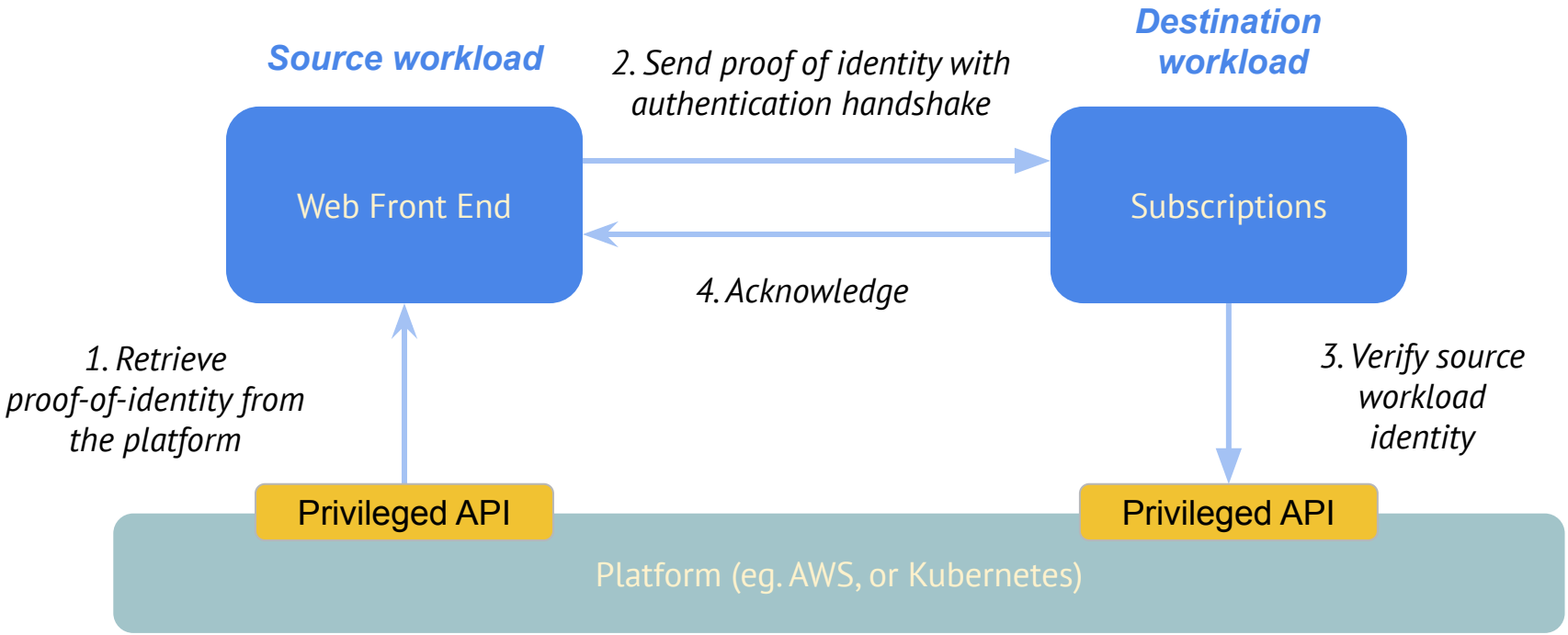
Model 1: Destination workload authentication



* Or key/secret, signed nonce etc.

Model 2: Platform mediated identity

Eg. AWS IAM, Kubernetes Service Accounts



SCYTALE

	Destination workload authentication	Platform mediated identity
API-driven credential rotation and distribution	No	Yes
One identity per workload	No	Yes
No credentials need to be deployed with the workload	No	Yes
Supports trust across different platforms platforms	Yes	No

The Inspiration for SPIFFE and SPIRE

SCYTALE

Google Application Layer Transport Security

"The ALTS trust model has been tailored for cloud-like containerized applications. Identities are bound to entities instead of to a specific server name or host. This trust model facilitates seamless microservice replication, load balancing, and rescheduling across hosts."

"Secure authentication and authorization within Facebook's infrastructure play important roles in protecting people using Facebook's services. Enforcing security while maintaining a flexible and performant infrastructure can be challenging at Facebook's scale, especially in the presence of varying layers of trust among our servers."

"During the startup, access to the long-lived credentials and short-lived credentials are provisioned to each instance.

This credential bootstrap is done by Metatron, *which is a tool at Netflix, which does credential management."*

spiffe

and

SPIRE

github.com/spiffe/spiffe

A set of specifications that cover how a workload should retrieve and use it's identity.

- SPIFFE ID
- SPIFFE Verifiable Identity Documents (SVIDs)
- The SPIFFE Workload API

github.com/spiffe/spire

The SPIFFE Runtime Environment. Open-source software that implements the SPIFFE Workload API for a variety of platforms.

Apache 2.0 license. Independent governance. Highly extensible through plug-ins.

SPIFFE ID

SCYTALE

spiffe://acme.com/billing/payments

Trust Domain

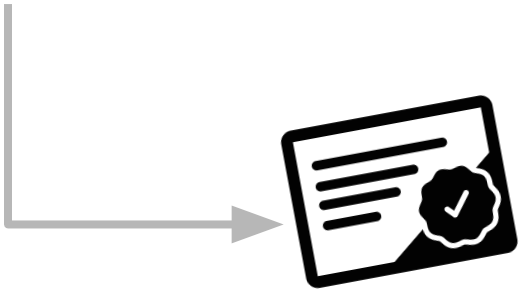
Workload Identifier

SPIFFE Verifiable Identity Document

SCYTALE

spiffe://acme.com/billing/payments

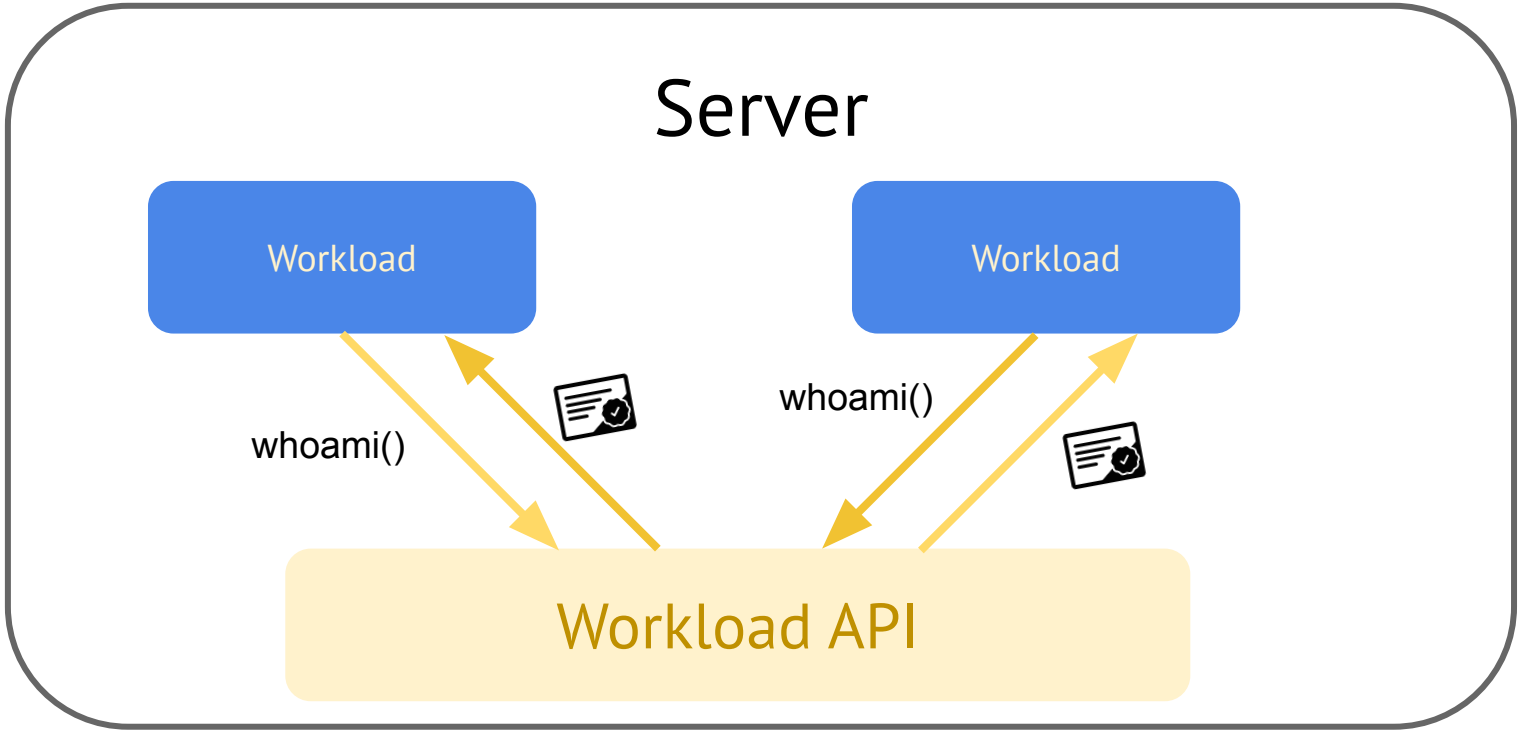
Typically short-lived



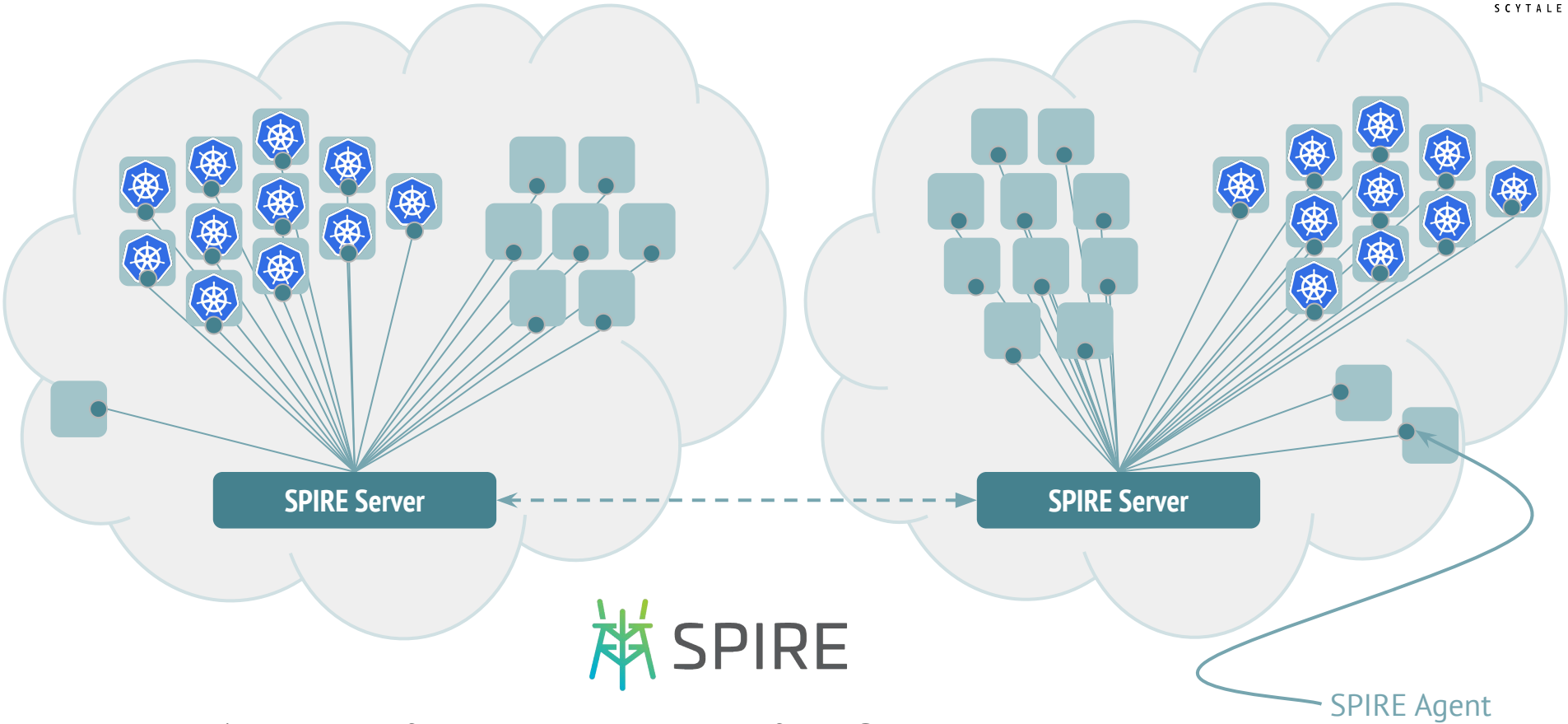
Today only one form of SVID (X509-SVID).
Other document types under consideration
(including JWT-SVID)

SPIFFE Workload API

SCYTALE

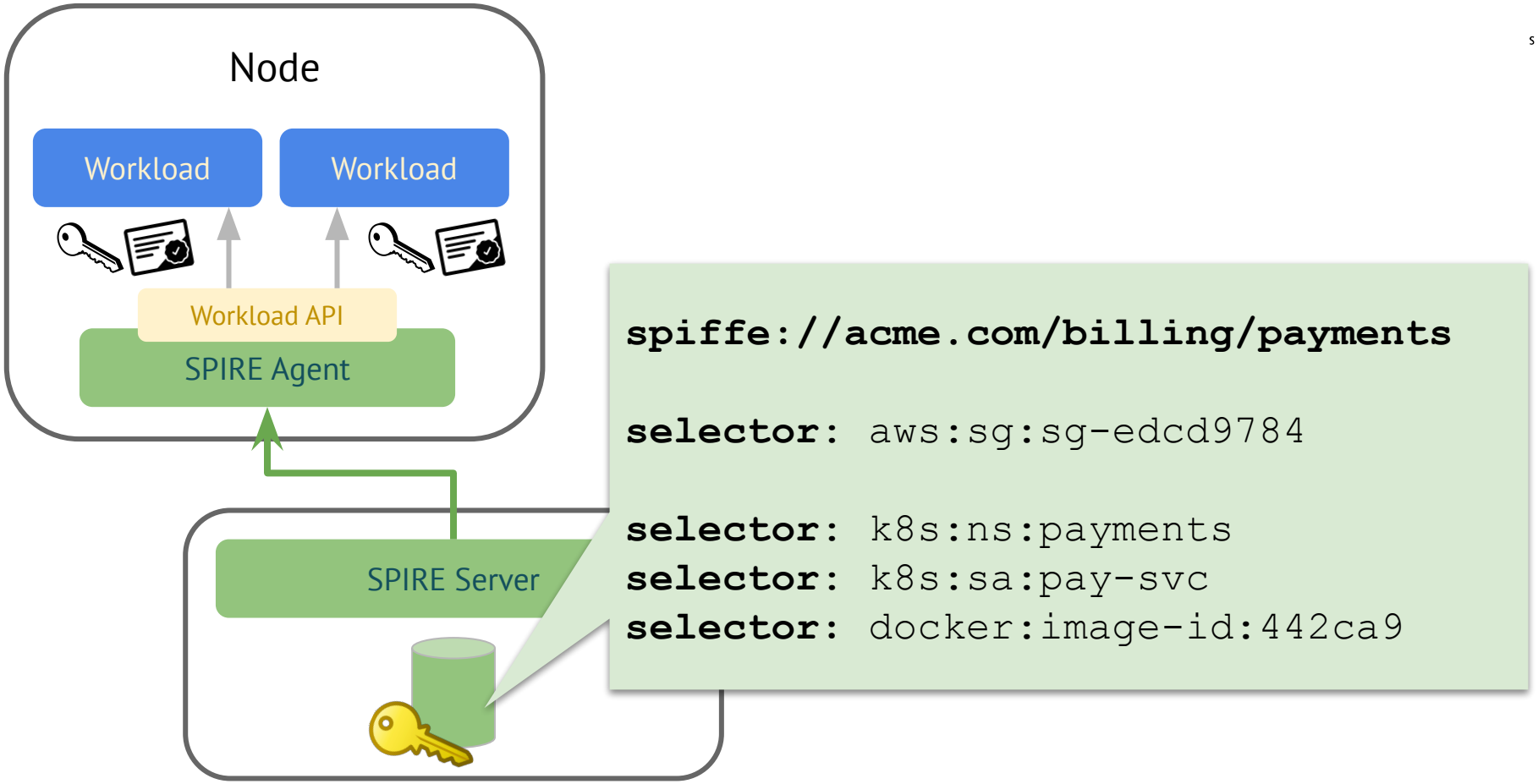


SCYTALE



SPIRE

A cross-platform implementation of the SPIFFE specifications


Node
Workload
Workload
Workload API
SPIRE Agent
SPIRE Server
spiffe://acme.com/billing/payments
selector: aws:sg:sg-edcd9784
selector: k8s:ns:payments
selector: k8s:sa:pay-svc
selector: docker:image-id:442ca9

Design Goals of SPIFFE and SPIRE

SCYTALE

- **Application identity driven.** By building a security model rooted in a strong assertion of application identity, policies and practices become application- and business unit- oriented rather than infrastructure-oriented.

- **Easily adoptable**. Users should be able to leverage Emissary with little or no code change. The system should work well in dynamically orchestrated containerized environments.

- **Federatable**. It should be possible to use these identity mechanisms across business units and even organizations.

- **Reliable**. The single points of failures in the system should be minimized and the system should degrade gracefully when any single point of failure is down.

- **Cloud and Container Ready**. It should be possible to safely extend trust to entities running on to third party cloud providers such as Amazon Web Services and Microsoft Azure, and container orchestrators such as Cloud Foundry and Kubernetes.

Security Goals of SPIFFE and SPIRE

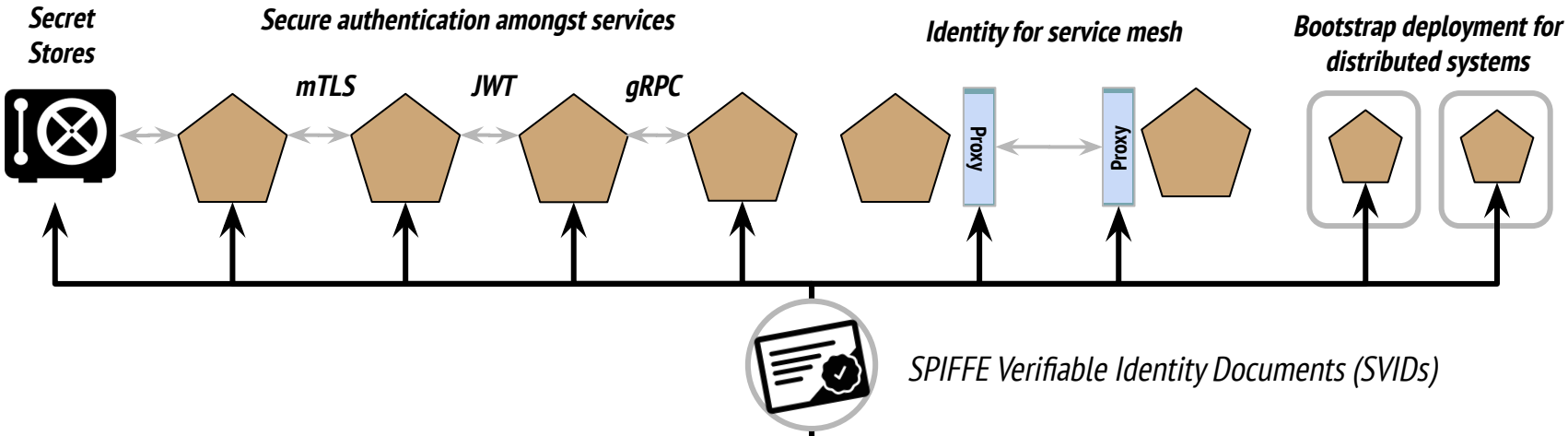
SCYTALE

- **Fully automated and policy driven**. Existing identity (particularly PKI) infrastructure is both complex and often requires “human trust”, which weakens delivery. Emissary is fully automated and should minimize manual key distribution.

- **Minimal Knowledge**. A compromised machine should only expose any secrets for workloads that happen to be running on that machine.

- **Reliable**. The single points of failures in the system should be minimized and the system should degrade gracefully when any SPOF is down. All “steady state” operations shouldn’t have requirements off of a specific node.

- **Scoped trust roots**. There should be no hardcoded, global trust roots as we see in the web browser world.



SPIFFE Workload API

SPIRE

Cloud platform attestation plug-ins

OS attestation plug-ins

Scheduler and PaaS attestation plug-ins

HSM, TPM, Kerberos attestation plug-ins

CA and secret store plug-ins

SCYTALE

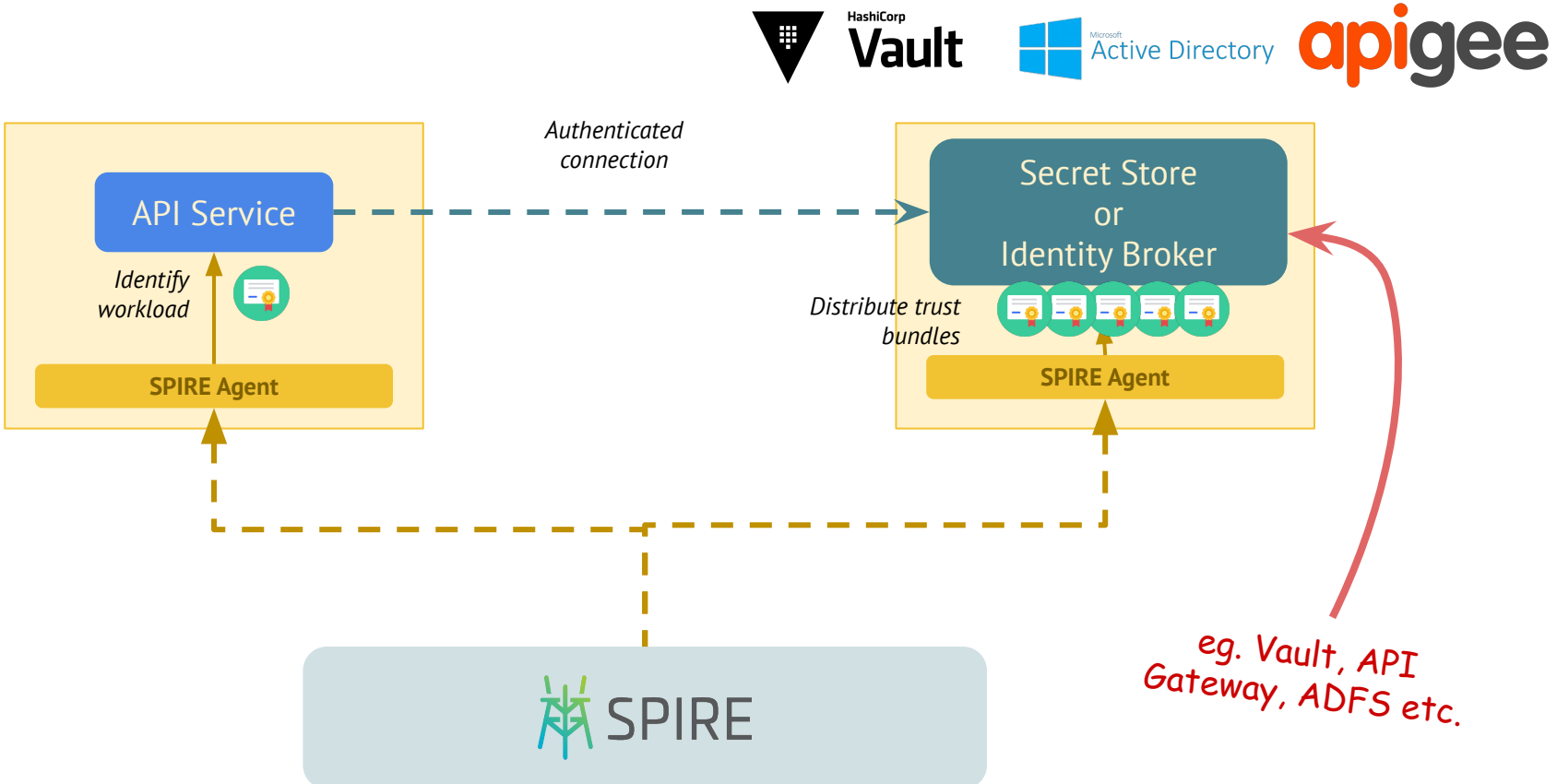
Use cases

How the identity plane becomes the unifying layer for infrastructure

Improving security posture

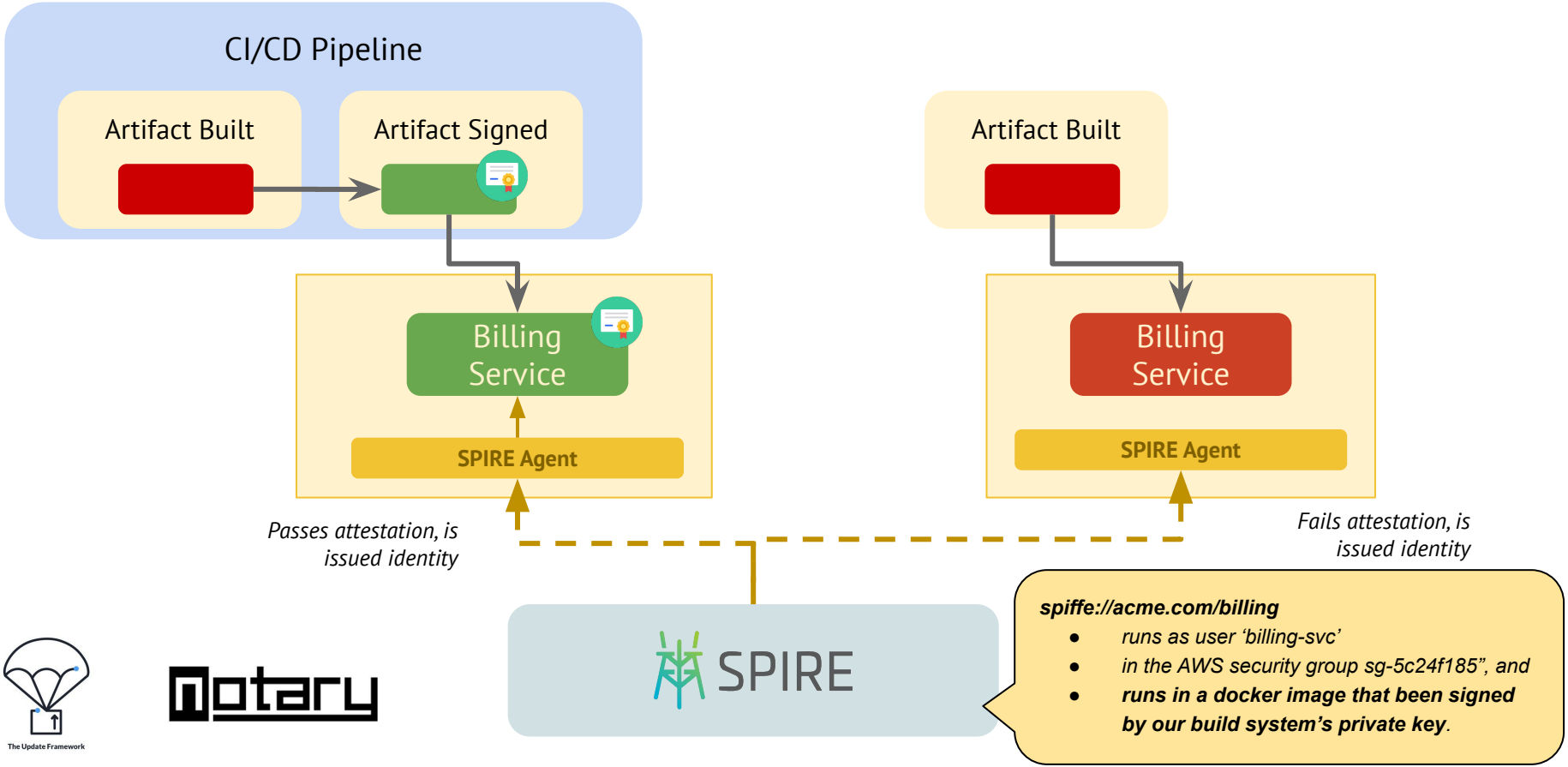
Minimize key leaks with secure introduction

SCYTALE

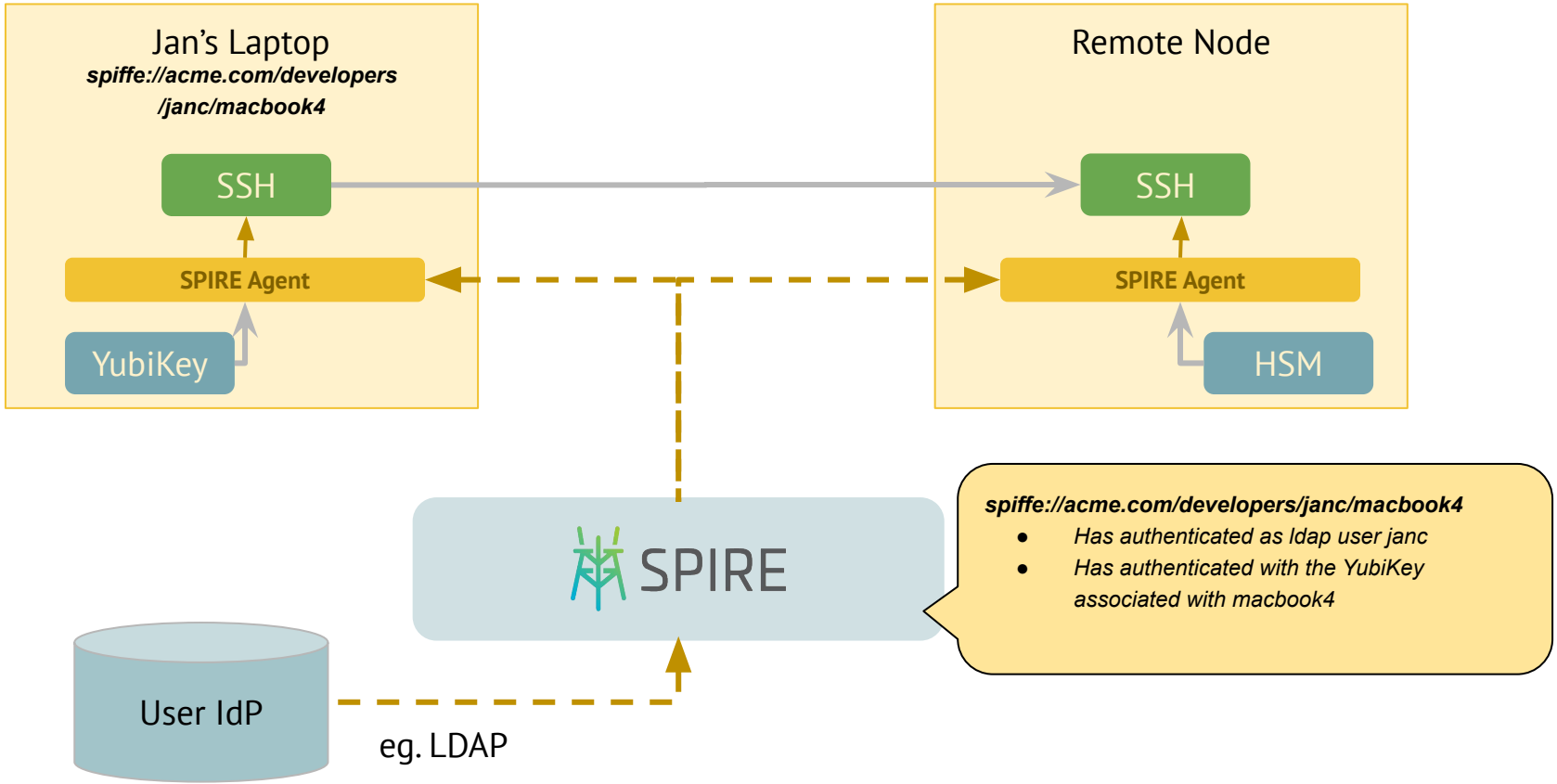


Enforce and verify release pipelines

SCYTALE



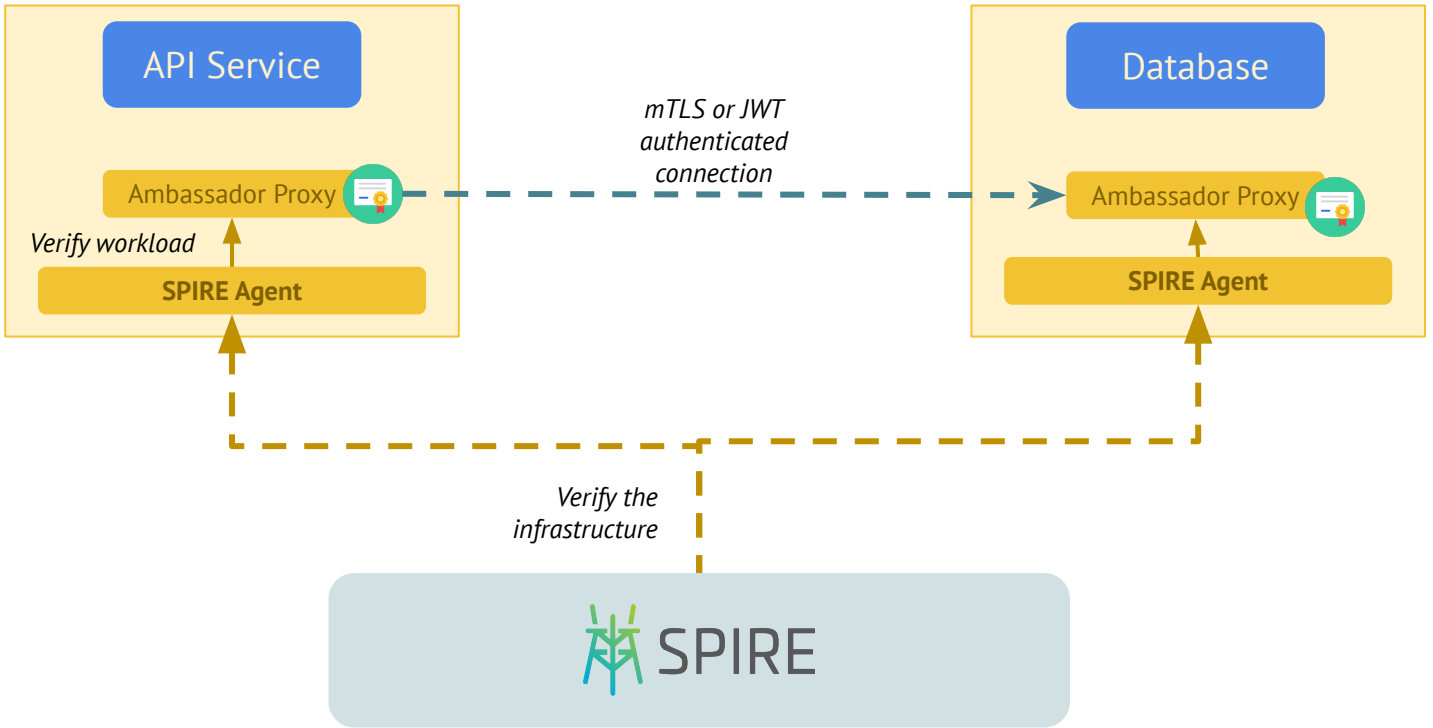
Authenticate developer access (BeyondCorp)



Improving developer efficiency

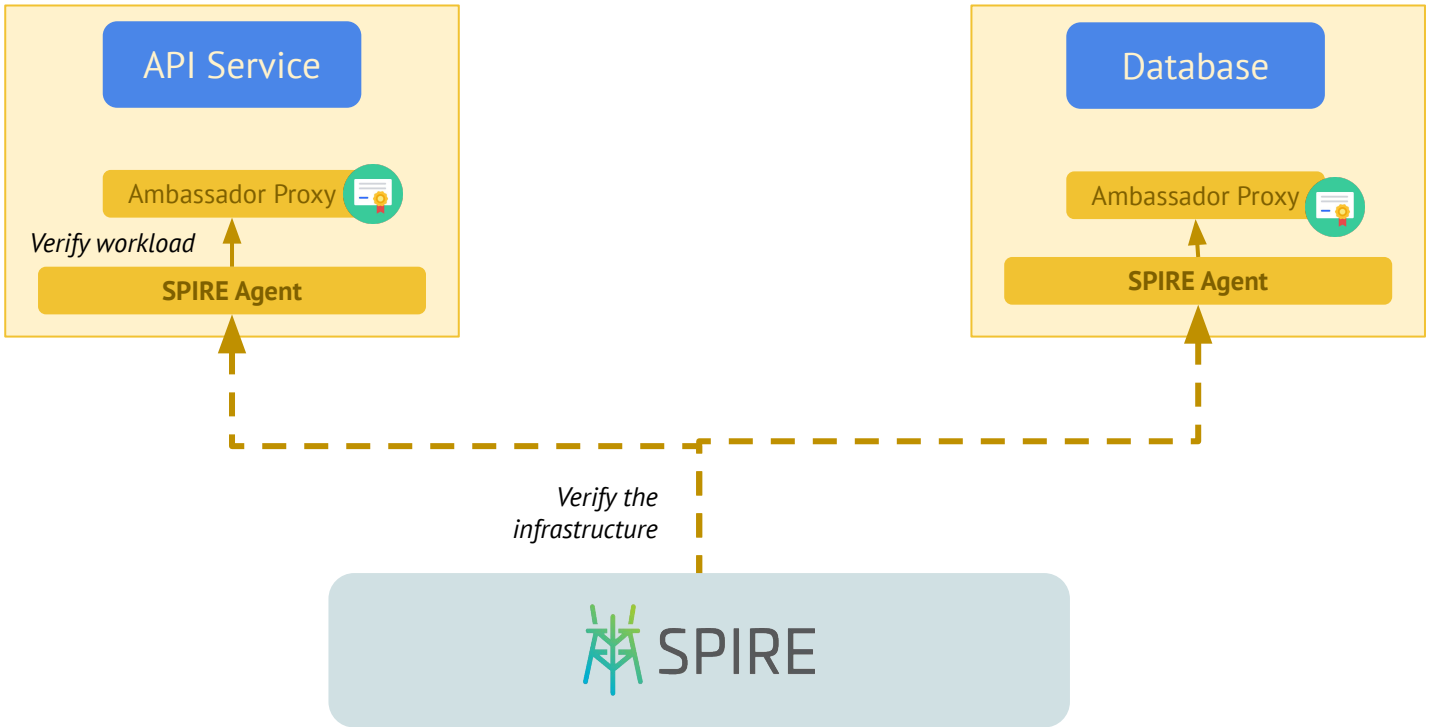
Simplify workload AuthN and AuthZ with Service Mesh

SCYTALE



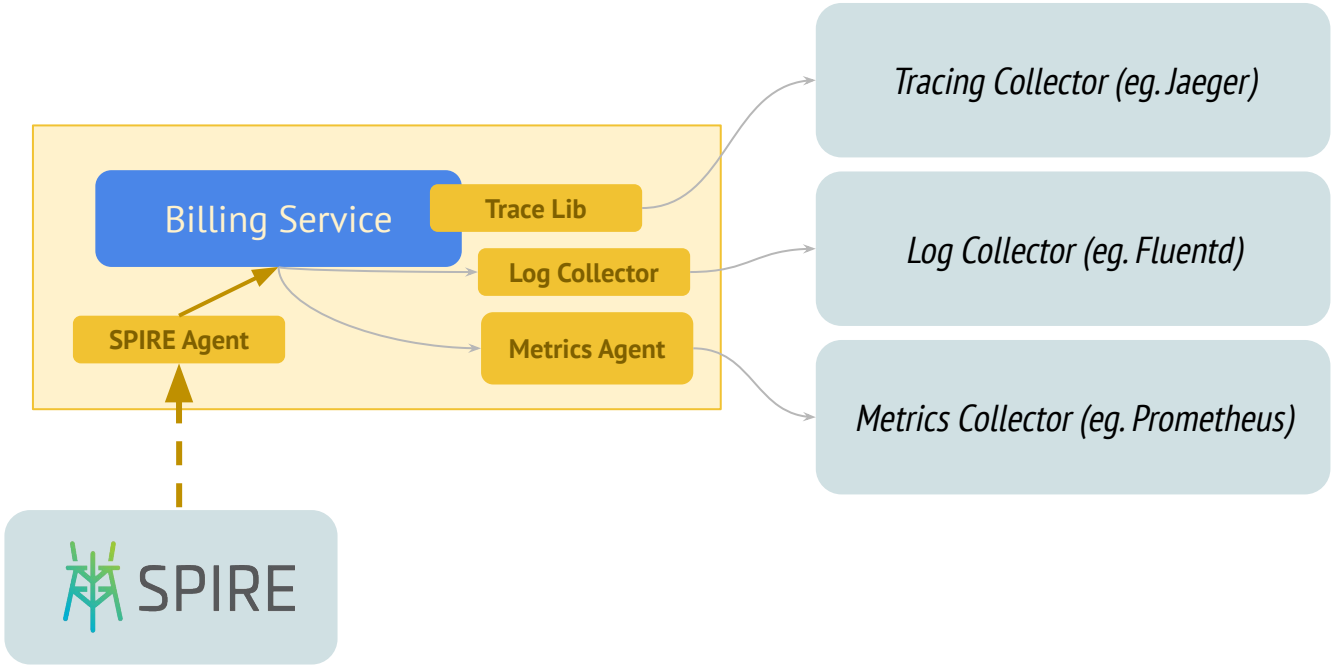
Simplify workload AuthN and AuthZ with Service Mesh

SCYTALE

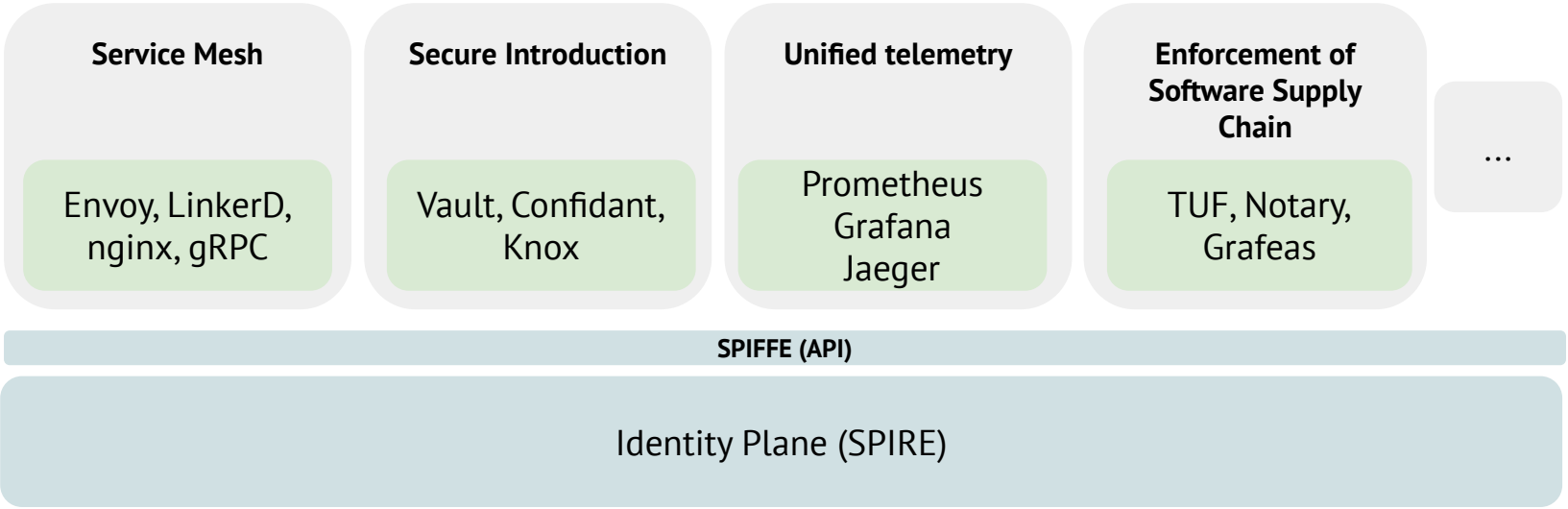


Improving post-incident forensics with unified telemetry

SCYTALE

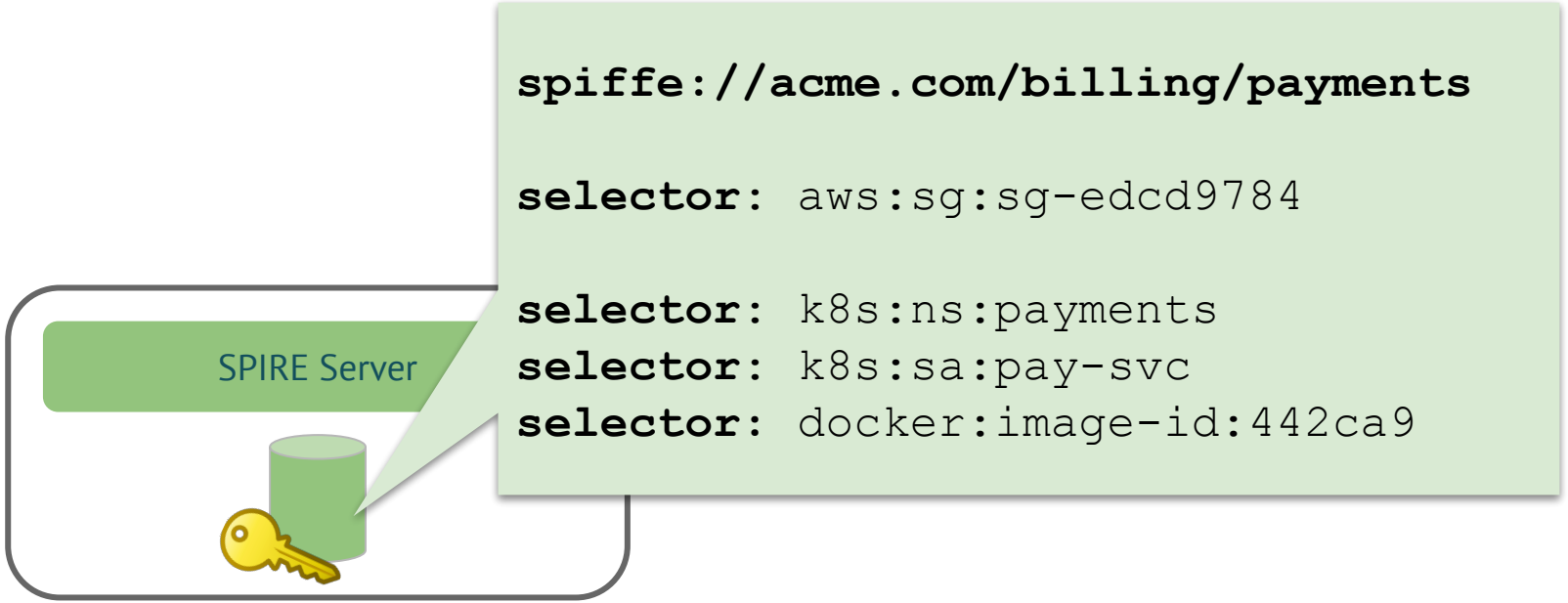


The Identity Plane becomes the unifying layer for infrastructure



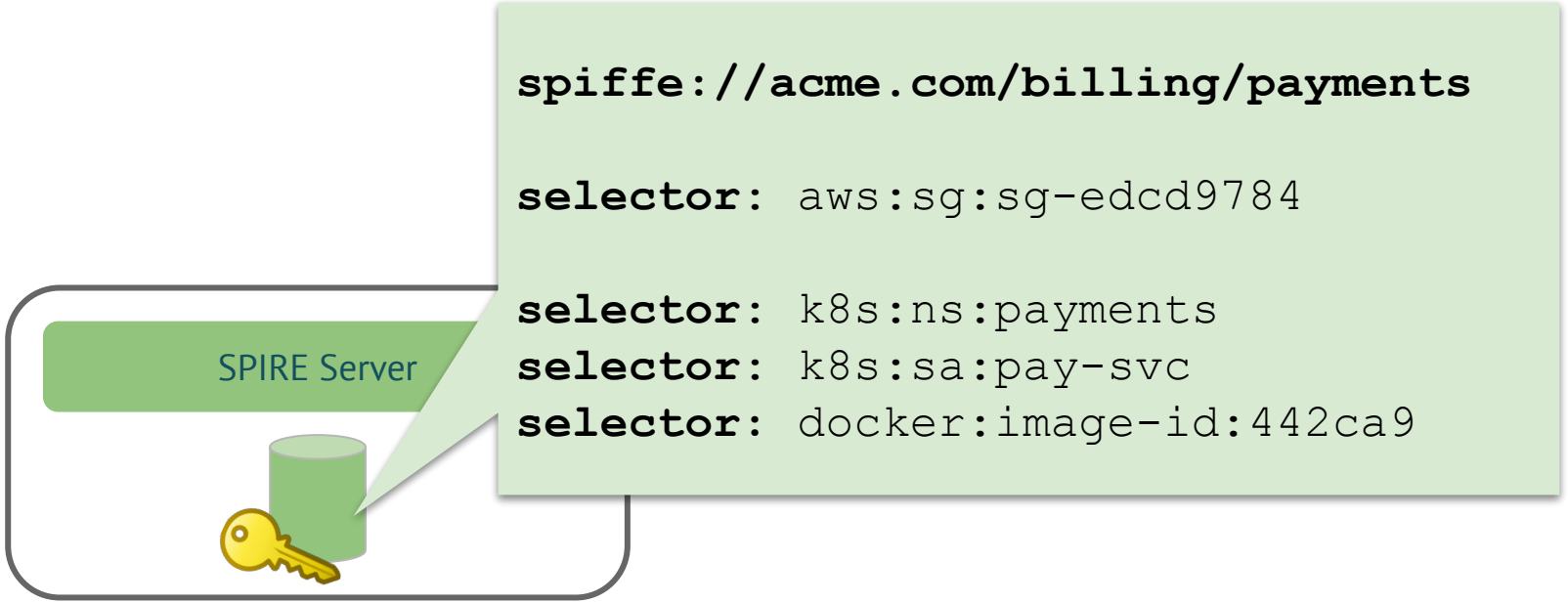
SPIFFE Runtime Environment

SCYTALE

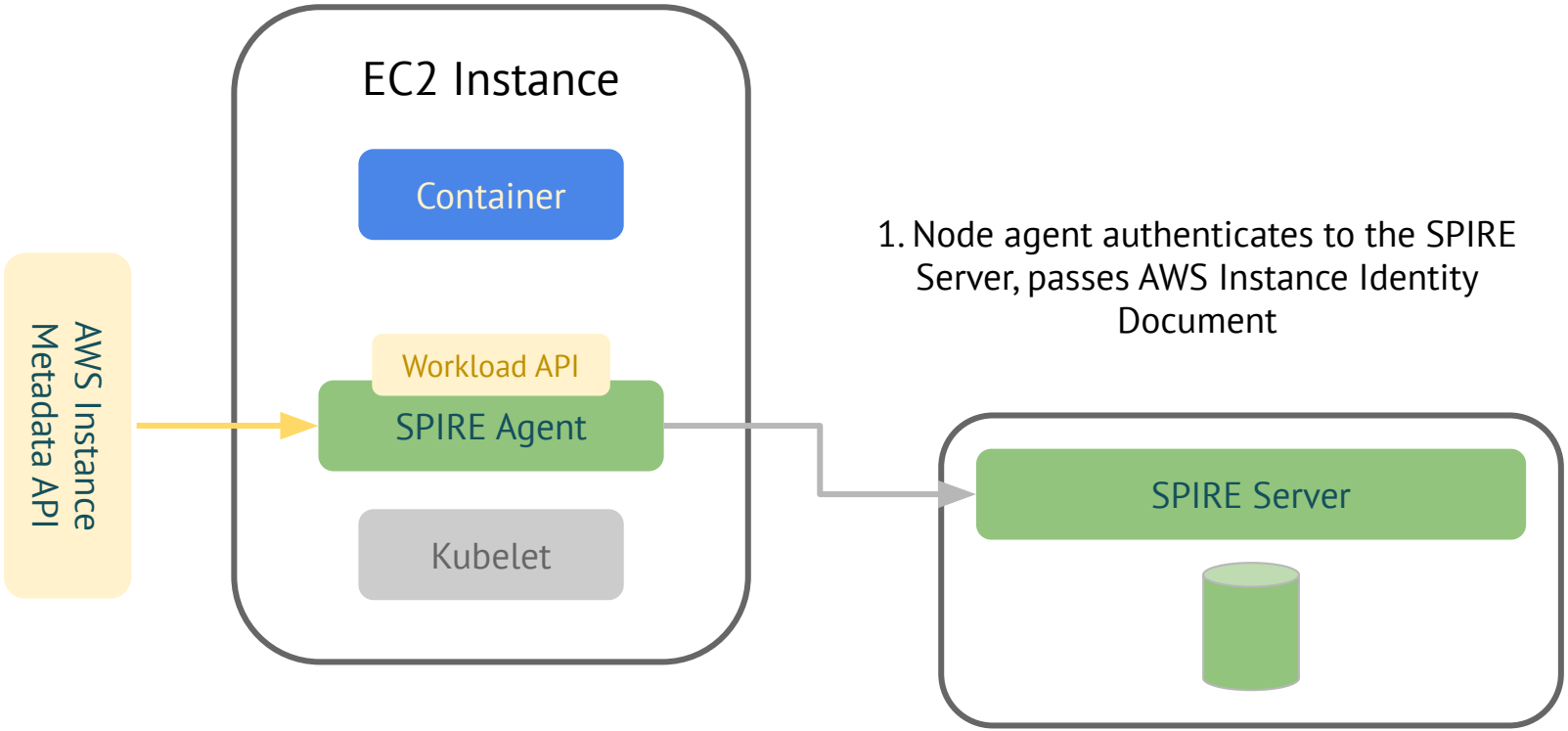


SPIFFE Runtime Environment

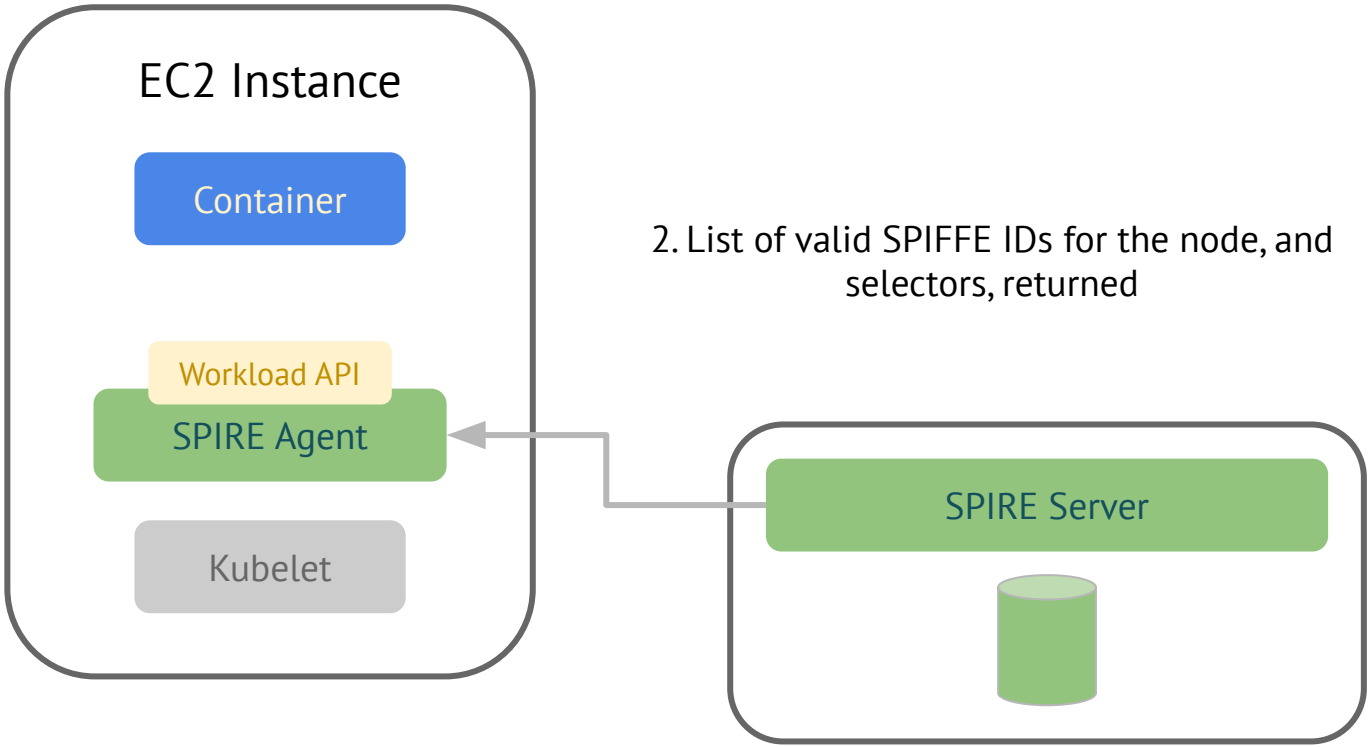
SCYTALE



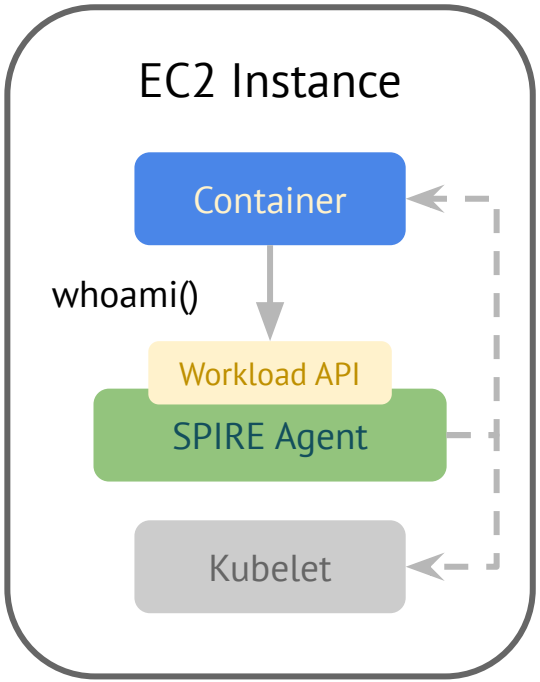
Node attestation



Node attestation

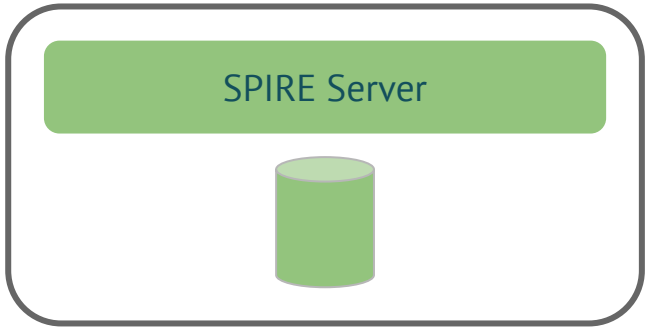


Workload attestation

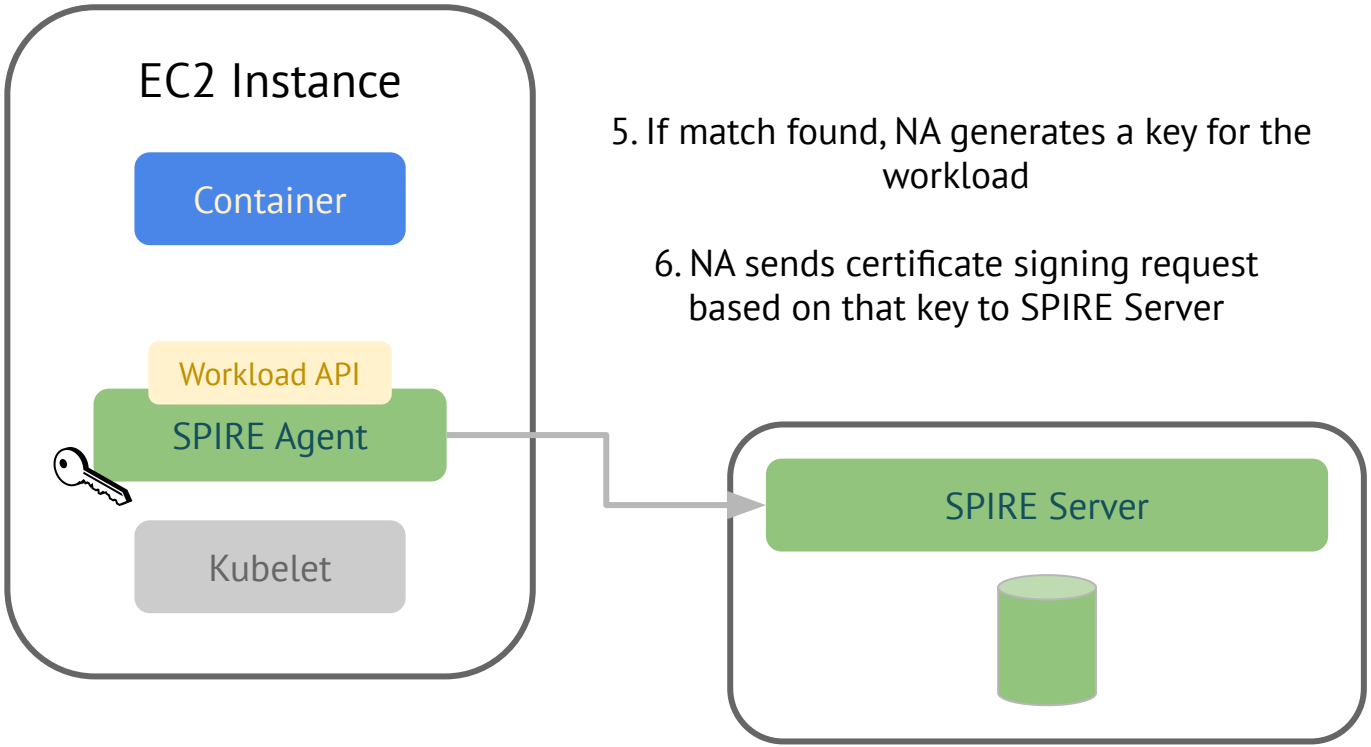


3. Workload requests identity

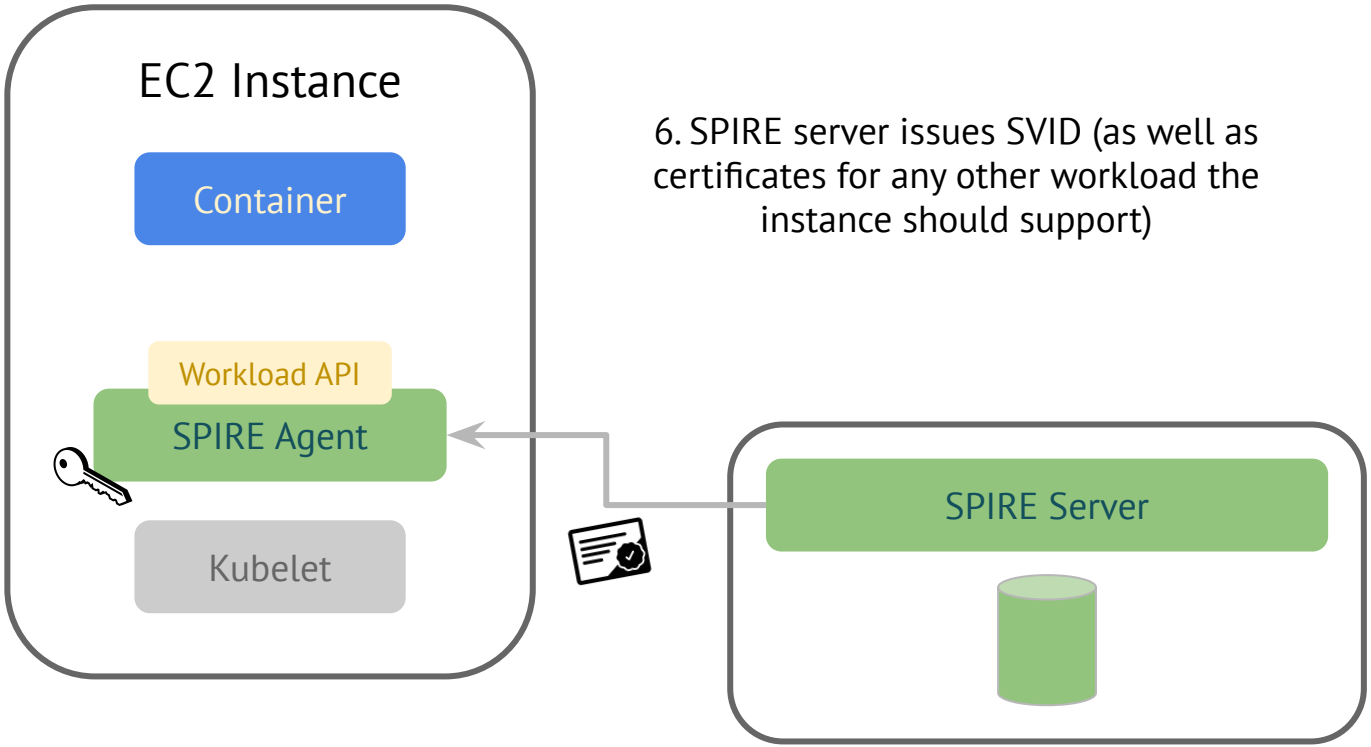
4. Node agent performs an out-of-band check of the workload process metadata, compares to known selectors



SVID Bundle Issuance

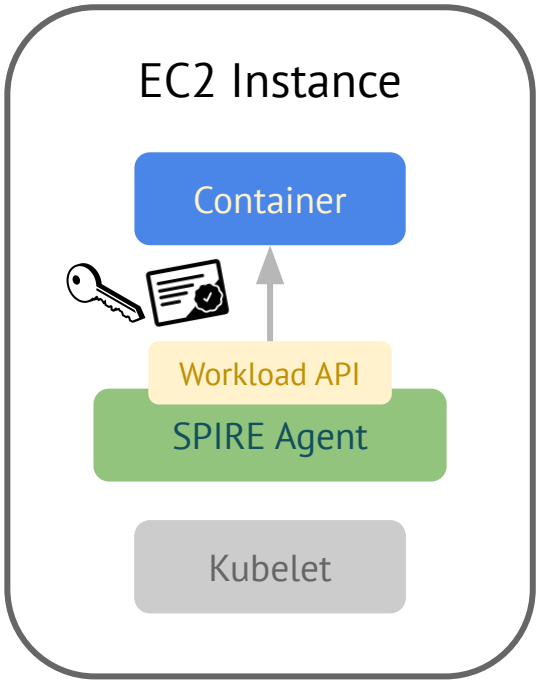


SVID Bundle Issuance



6. SPIRE server issues SVID (as well as certificates for any other workload the instance should support)

SVID Bundle Issuance



7. Certificate bundle returned to the workload