Uniform Workload Identity Everywhere: SPIRE Integrations and Extensibility

Ryan Turner - Software Engineer @ Uber

Common Integration Challenges

- Using multiple environments - public and/or private clouds
- Proprietary tooling and infrastructure
- Mix of legacy and cloud-native applications
- Enforcing uniform authentication across all RPCs

SPIFFE + SPIRE can provide consistent, strong identity and meet all these use cases

Agenda

- SPIFFE + SPIRE Overview
- SPIRE Integrations
- Extensibility Points in SPIRE
- Downstream Integrations
- Q+A

SPIFFE Overview

SPIFFE - Secure Production Identity Framework for Everyone

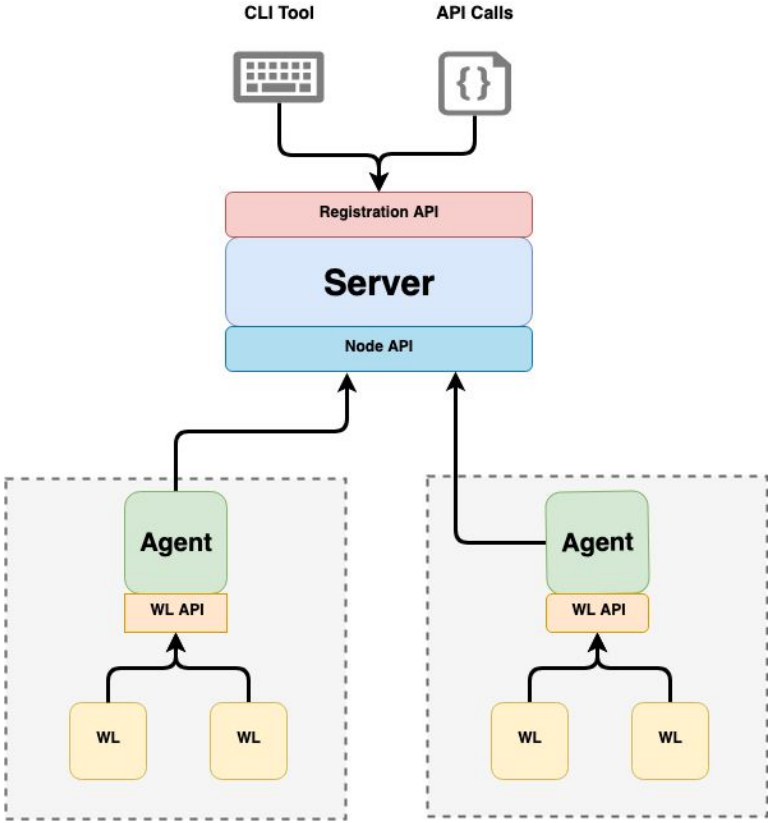
- **SPIFFE ID** - Identifier standard
  - URI format: `spiffe://trust-domain/path`
- **SVID** - Identity document standard
  - SVID - SPIFFE Verifiable Identity Document
  - Supported document types:
    - X.509
    - JWT
- **Workload API** - Specification for issuing/retrieving SVIDs

SPIRE Overview

SPIRE - SPIFFE Runtime Environment

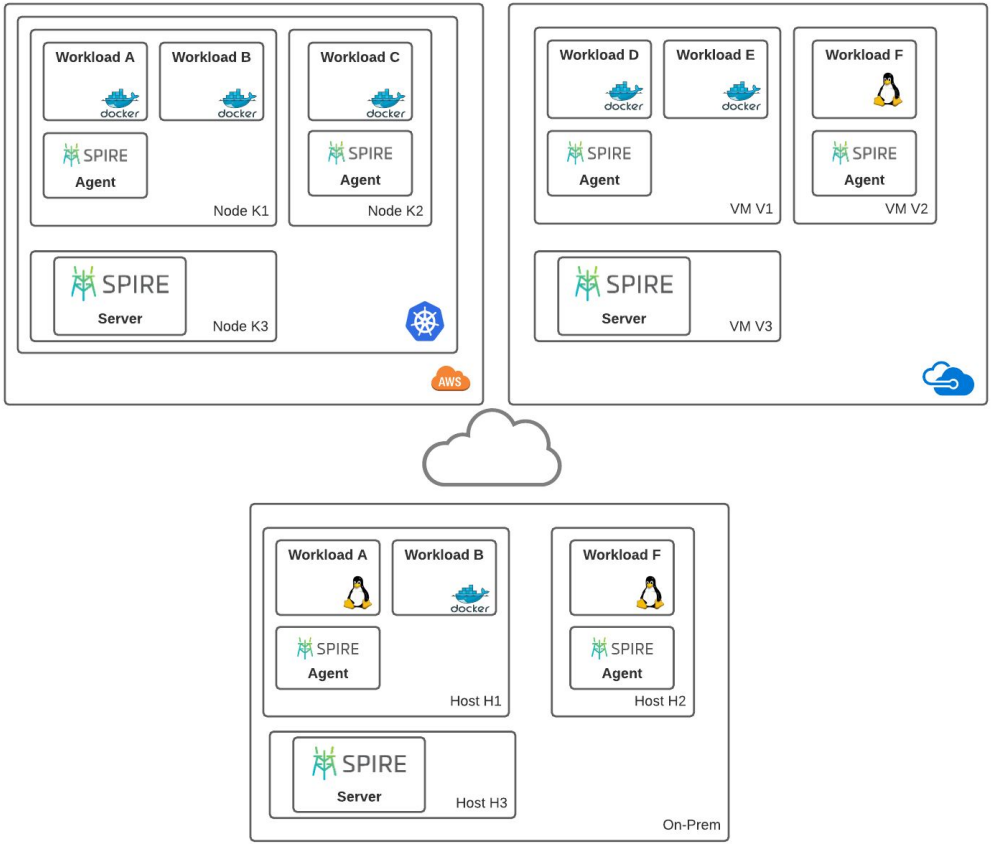
- Open-source implementation of SPIFFE specification
- Control plane for identity distribution/rotation
- Scalable distributed system

Architecture



WL = Workload

SPIRE in Complex Environment



Types of Integrations

- Operations of SPIRE
  - Getting SPIRE up and running
  - Controlling select functionality and security properties
- Consumption of SPIRE-issued identity
  - Downstream integrations
  - Simplifying propagation of SVIDs
  - Using SVID as authentication material for external domains

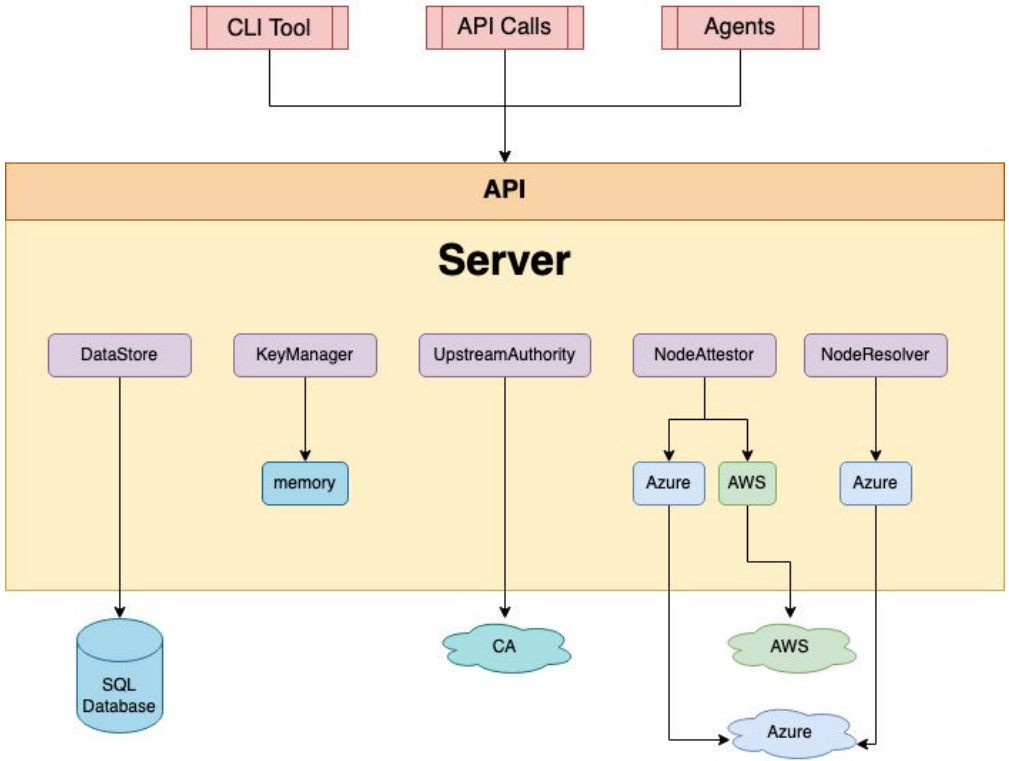
Operations Integrations

- Linking identity chain of trust to existing PKI
- Host identity <-> SPIRE
- Host categorization (node alias)
  - Labeling hosts to scope distribution of identity
- Key management
  - Controlling how SPIRE manages its private keys
- Workload identification (attestation)
  - Querying runtime attributes of workload
- Event hooks
  - Triggering downstream processes in external systems

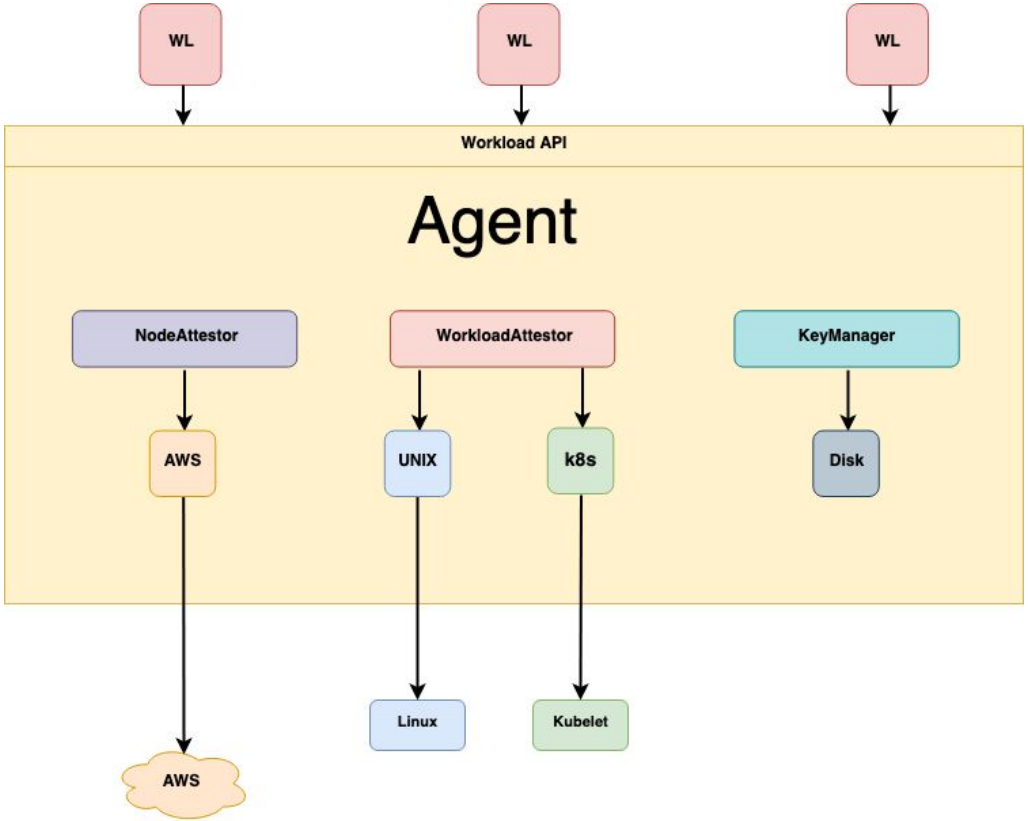
SPIRE Plugin Framework

- Plugin interfaces defined as Protocol Buffers
- Built-in plugins loaded in-process
- External plugins provided as binaries launched by SPIRE
- Communication from SPIRE core to plugins over gRPC
- Based on open source Hashicorp go-plugin project

Server Plugin Types



Agent Plugin Types



UpstreamAuthority

Applies to: **Server**

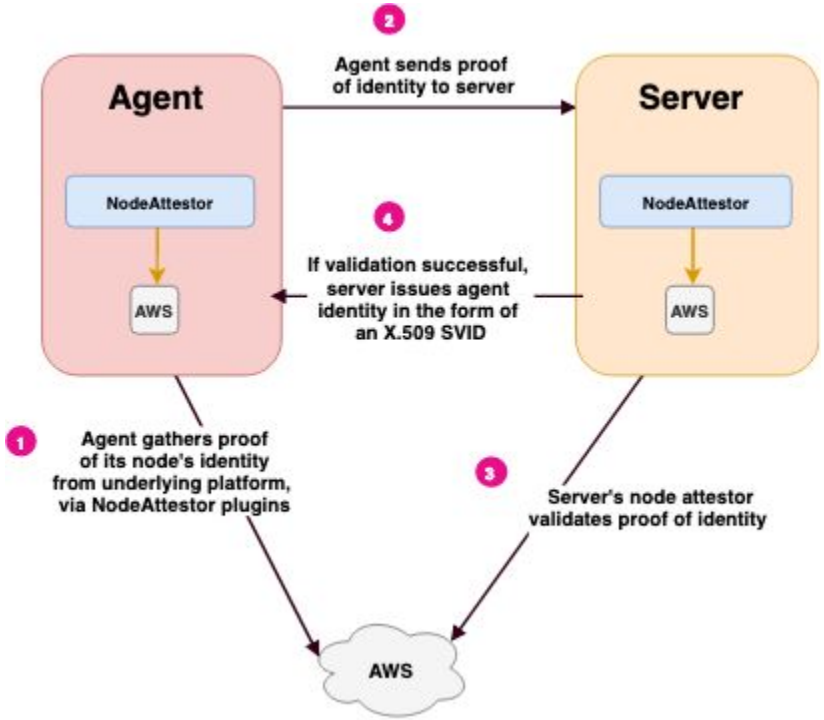
- Synchronizes upstream PKI chain/keys with SPIRE
- Handles CSRs for SPIRE CA
- Optionally accepts SPIRE JWT signing keys
- Available built-in plugins
  - disk
  - aws_pca
  - awssecret
  - vault
  - spire

NodeAttestor

Applies to: **Server**, **Agent**

- Authenticates a node (physical or virtual) in the infrastructure
- Challenge-response protocol
- Defines bridge of trust between host identity system and SPIRE
- Built-in plugins:
  - aws_iid
  - azure_msi
  - gcp_iit
  - join_token
  - k8s_psat
  - sshpop
  - x509pop

Example NodeAttestor: AWS

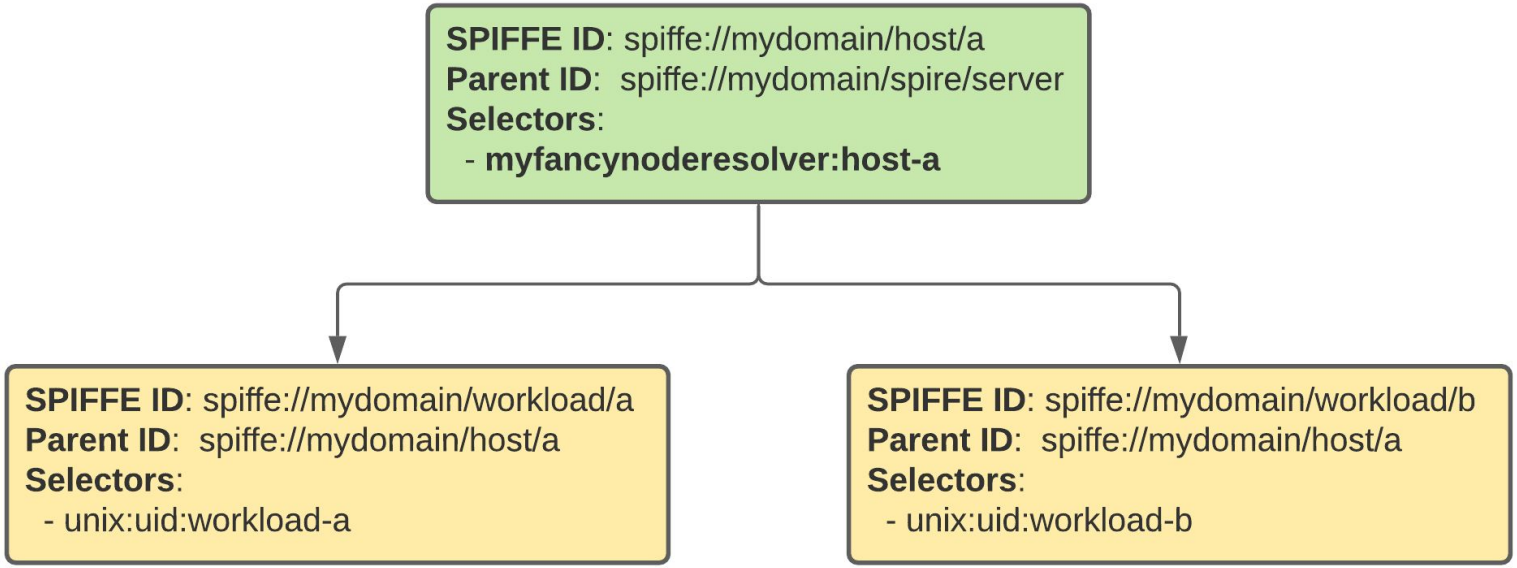


NodeResolver

Applies to: **Server**

- Expands the selector set for an attested node
  - Selectors can be based on host metadata or be static
- Enables distribution of identities to more finely-grained subsets of hosts
  - Alias registration entries matching node selectors can be used to group workload registrations

Grouping Registrations (by Host)

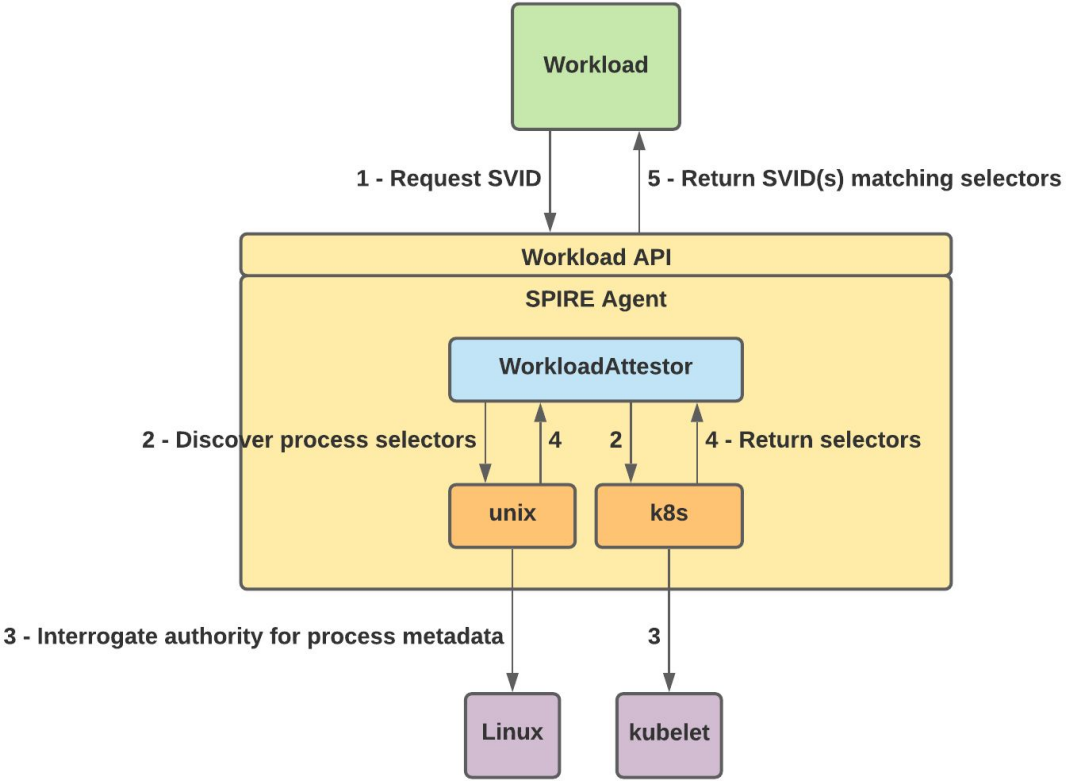


WorkloadAttestor

Applies to: **Agent**

- Interrogates trusted system for attributes of process
- Matches workload metadata to selectors of identity registrations
- Example authorities: OS kernel, orchestration platform
- Built-in plugins:
  - docker
  - k8s
  - unix

WorkloadAttestor Flow



KeyManager

Applies to: **Server**, **Agent**

- Handles private key operations and storage
  - Private key generation
  - Computes digital signatures of data
- Built-in plugins:
  - disk
  - memory

Notifier

Applies to: **Server**

- Publishes notifications of events
- Currently only hooked up to trust bundle events
- Built-in plugins:
  - gcs_bundle
  - k8sbundle

Implementing SPIRE Plugins

- Plugin interfaces defined in `proto/spire/{agent,server}/*`
- Implement respective plugin interface
- Add HCL config stanza for respective component(s) (Server and/or Agent)
  - Example for custom NodeAttestor plugin called "mynodeattestor:

```
NodeAttestor "mynodeattestor" {
  plugin_cmd = "/path/to/plugin-binary"
  plugin_checksum = "<SHA256 of binary>"
  plugin_data = {
    # custom plugin data goes here
  }
}
```

Downstream Integrations

- Envoy
  - mTLS using X.509 SVIDs
  - SPIRE Workload API implements Envoy SDS
- OIDC Federation
  - Authenticate to external services with SVIDs
  - Example using a JWT-SVID to invoke AWS APIs

Extensions under Consideration

- Agentless mode
  - Enables serverless use cases
- Integration with Apache data projects
  - Exchanging Kerberos user identities for SPIFFE identities

Conclusion

- Extensibility of SPIRE => identity to complex environments
- Native integrations simplify usage of SPIFFE for authentication
- Plugin model enables internal proprietary extensions

SPIFFE/SPIRE Community

- Website: https://spiffe.io
- GitHub
  - SPIFFE: https://github.com/spiffe/spiffe
  - SPIRE: https://github.com/spiffe/spire
- Slack: https://slack.spiffe.io
- Twitter: https://twitter.com/SPIFFEio


KEEP CLOUD NATIVE
EVERYWHERE
KubeCon
CloudNativeCon
North America 2020
Virtual
HELM
KV

Reference

- Extending SPIRE
- SPIRE Server and Agent plugin types
- Plugin configuration
- hashicorp/go-plugin
- SPIRE Examples