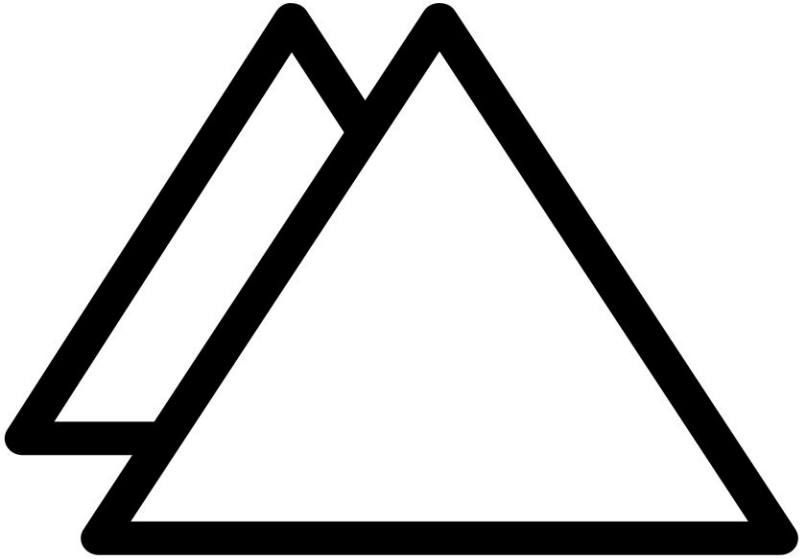
Throw Away Your Passwords

Trusting Workload Identity

controlplane

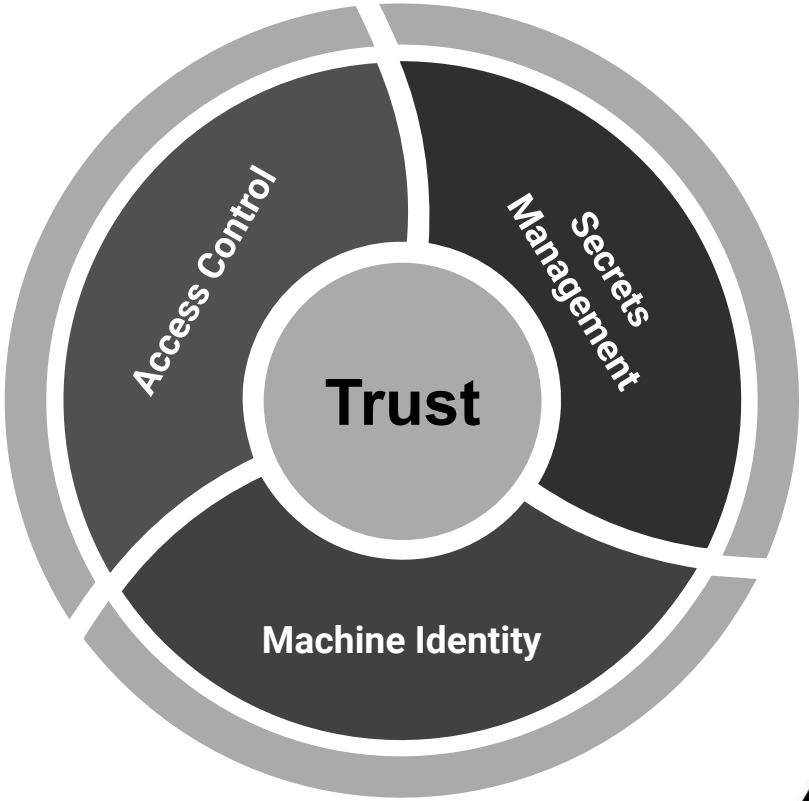
Ric Featherstone
(Cloud Native) Engineer

@controlplaneio



controlplane

- **Demystify** Machine Identity
- Discuss **historical issues**
- Solve the **"bottom turtle"** trust bootstrap quandary
- **Acquire** a Workload Identity or Secret Zero
- **Appraise** Open Source implementations and technologies
- Strive for a world in which passwords and static keys are replaced with **dynamic credentials** and **hardware roots of trust**



controlplane

Ye Olde Ways

- Location Based
  - Firewalls
  - X.509 IP SAN
- Secrets Management
  - Username and Passwords
  - API Keys
- Authentication Mechanisms
  - PKI
  - Kerberos

controlplane

Dynamic Infrastructure

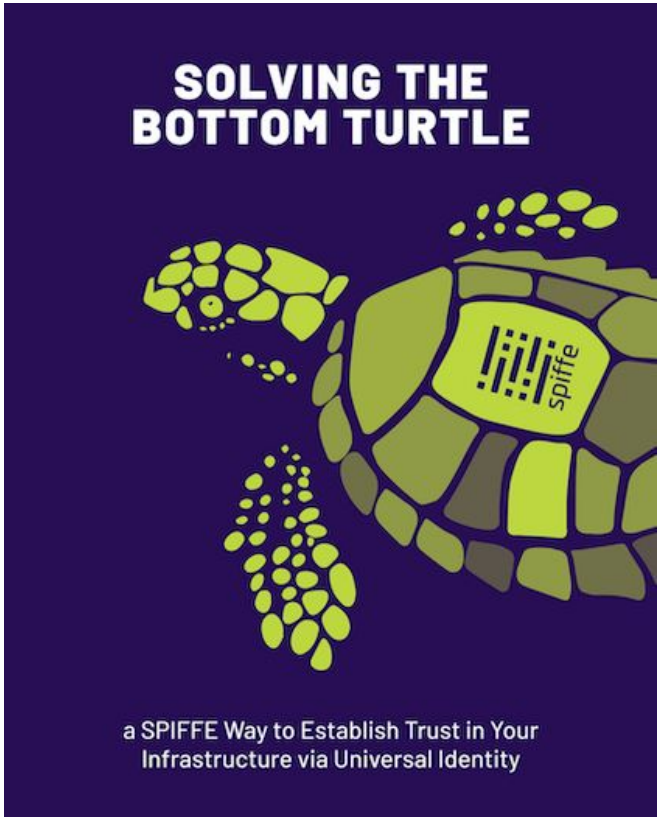
controlplane

Secret Zero

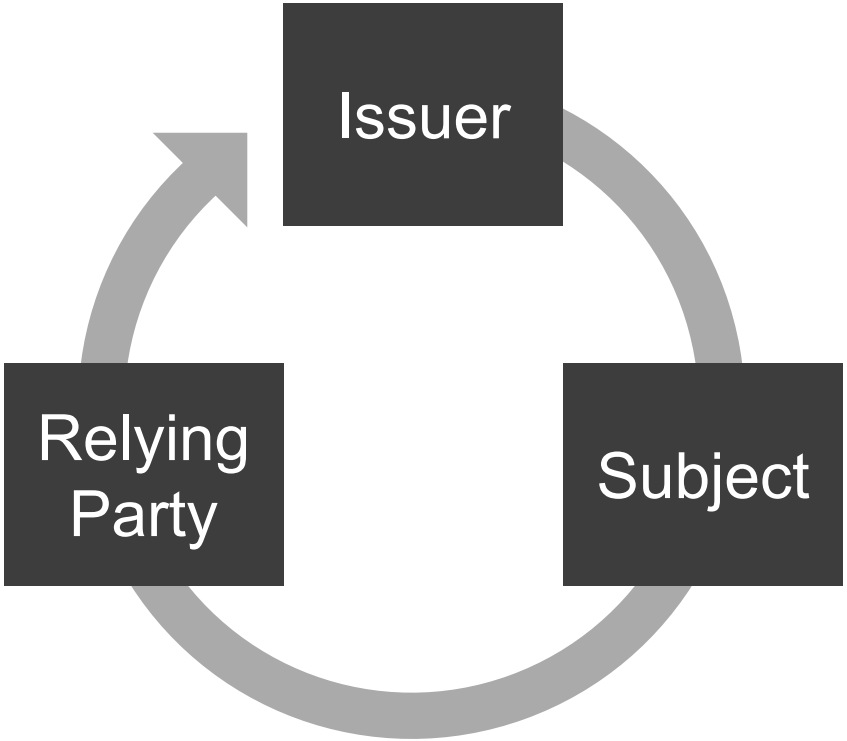
Protecting access to one secret results in a new secret you need to protect

“turtles all the way down”

https://spiffe.io/book/



controlplane

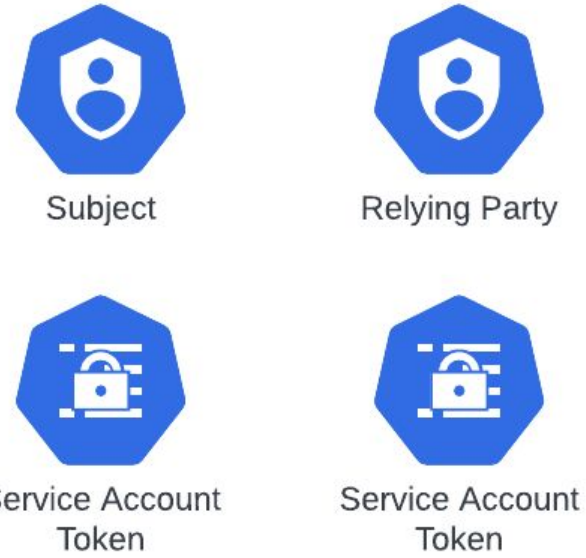


controlplane

This **Secret Zero** sounds good, where can I get one?

```
curl https://API_SERVER/apis/authentication.k8s.io/v1/tokenreviews
     -H "Authorization: Bearer RELYING_PARTY_TOKEN"
     -H \'Content-Type: application/json; charset=utf-8\'
     -d @data/token-review.json


{
  "apiVersion": "authentication.k8s.io/v1",
  "kind": "TokenReview",
  "spec": {
    "token": "SUBJECT_TOKEN"
  }
}
```



controlplane

```
{
  "authenticated": true,
  "user": {
    "username": "system:serviceaccount:default:example-1",
    "uid": "fa8450cf-00bd-411a-9a34-b86253076633",
    "groups": [
      "system:serviceaccounts",
      "system:serviceaccounts:default",
      "system:authenticated"
    ],
    "extra": {
      "authentication.kubernetes.io/pod-name": [
        "example-1-7c78d55679-9d5jn"
      ],
      "authentication.kubernetes.io/pod-uid": [
        "084ae167-5d1a-4dd0-80ff-4026c9232683"
      ]
    }
  },
  "audiences": [
    "https://container.googleapis.com/v1/projects/***/locations/europe-west2/
workload-identity"
  ]
}
```


Token Review API

Requirements

- Access to the API Server
- Permission

```
rules:
- apiGroups:
  - authentication.k8s.io
  resources:
  - tokenreviews
  verbs:
  - create
```

Vault

controlplane

Don’t use **long lived** Service Account tokens

Bound Service Account Tokens

```
"name": "token",
"projected": {
    "sources": [
        {
            "serviceAccountToken": {
                "audience": "demo",
                "expirationSeconds": 900,
                "path": "token"
            }
        }
    ]
}
```

- Projected Volume
- Limited Scope
  - **Audience** Bound
  - **Time** Bound

controlplane

So now we have a **Secrets Management** problem again

Service Account Issuer Discovery

- OpenID Connect (**OIDC**) Discovery Compatible
  - Discovery Document
  - JSON Web Key Set (JWKS)
- Vanilla Kubernetes
  - API Service Access
  - RBAC Permissions
- Managed Kubernetes
  - Public Endpoint
  - No Authorization

```
rules:
- nonResourceURLs:
  - /.well-known/openid-configuration
  - /openid/v1/jwks (vanilla)
  verbs:
  - get
```

controlplane

What is **in** a JWT?

What is in a JWT?

HEADER . PAYLOAD . SIGNATURE

```
jq -R 'split(".") | .[1] | @base64d | fromjson' <<< $TOKEN
```

controlplane

```
HEADER
{
  "alg": "RS256",
  "kid": "9vwrVontDoFY8qW28rmQmC8HNyHb5lD9qJ3gJFfa1ZE"
}
PAYLOAD
{
  "aud": [
    "demo"
  ],
  "exp": 1652180920,
  "iat": 1652180020,
  "iss": "https://container.googleapis.com/v1/projects/***/locations/europe-west2/clusters/workload-identity",
  "kubernetes.io": {
    "namespace": "default",
    "pod": {
      "name": "example-1-7c78d55679-pw8kv",
      "uid": "b6f5a8d1-ce11-46cd-a8a5-1453ab141ce9"
    },
    "serviceaccount": {
      "name": "example-1",
      "uid": "771af7f6-5ba0-4223-990f-6d7775358534"
    }
  },
  "nbf": 1652180020,
  "sub": "system:serviceaccount:default:example-1"
}
SIGNATURE
"qFeKznKPGyE3yZJEfZ35VQ1T77g1vIKi_lx0TftTkJlQVlyAO-3EPG7homwWwKqJyxegu9fnlXQmKFS2MJBy85NxiEvuyS0z7QAvuP4j
55x-WFn6ww1yGZreLpQiOmjWfvfvautSHTRsf8MBs9XaM0kEV-tJtIAOfEibhCucVzHDaJXh8jdqXcy2DHcrMWZeKL9nXGKZV4neSHdJw
A0572WFNpkH0qXMoPzbwMye9-s3l0cLPzhzY8pBe46Nb90fi8qzely0m6hr3_BazQZ_YpYgZZC-gMnZSAZIVyKooZSpYkHbTME0u5vJSQ
workload-identity $
```

What does that **look** like?

```
PAYLOAD
{
  "aud": [
    "demo"
  ],
  "exp": 1652353441,
  "iat": 1652352541,
  "iss": "https://container.googleapis.com/v1/projects/***/locations/europe-west2/clusters/workload-identity",
  "kubernetes.io": {
    "namespace": "default",
    "pod": {
      "name": "example-2-89fbdfcbb-bg44q",
      "uid": "878269db-c170-4465-8e8a-abe49c608c64"
    },
    "serviceaccount": {
      "name": "example-2",
      "uid": "46913dbf-1f0e-4a38-a8da-a32c4a3b41d3"
    }
  },
  "nbf": 1652352541,
  "sub": "system:serviceaccount:default:example-2"
}
workload-identity $
```

```
DISCOVERY DOCUMENT
{
  "issuer": "https://container.googleapis.com/v1/projects/***/locations/europe-west2/clusters/workload-identity",
  "jwks_uri": "https://container.googleapis.com/v1/projects/***/locations/europe-west2/clusters/workload-identity/jwks",
  "response_types_supported": [
    "id_token"
  ],
  "subject_types_supported": [
    "public"
  ],
  "id_token_signing_alg_values_supported": [
    "RS256"
  ],
  "claims_supported": [
    "iss",
    "sub",
    "kubernetes.io"
  ],
  "grant_types": [
    "urn:kubernetes:grant_type:programmatic_authorization"
  ]
}
workload-identity $
```

JWKS


```
{
  "keys": [
    {
      "kty": "RSA",
      "alg": "RS256",
      "use": "sig",
      "kid": "LYx5ReZr4IZ73ROI3pZopqPHXZ-AWQnVwlwkU1zTABs",
      "n": "gSXTqNC-DEGyBCq9Ey45WL1Ll89Yf8zgk-RrkonSYck7o-O-46sJ8dp4QZTLXpMBFzI1psQ4E5GkyQLca4NlLalf1A9dMAQnljHVTHeNDV3ABlcyF6Y9T-0yCe_Va2KFeSSUvE3BqkleRClkgyHlup7B7GXyCjkj62wtC8oZBfWJfWEj6-JzmFCKfD01DdwFA7hkeVFqvGbRuChDfuH7maKnUCvDieNZoUOKkL6NQqARKfU1wylJwK8rqDMWn37i4NdHOW0HBwXIDanIlUpD6A4D_R1Z6-yWypBI5ba-wOr_ZZvudPyXrGdgUOm2o6-gL0bhcPLd5VN4GfnnwOPh8w",
      "e": "AQAB"
    }
  ]
}
```


workload-identity $

```
PAYLOAD
{
  "aud": [
    "demo"
  ],
  "exp": 1652353670,
  "iat": 1652352770,
  "iss": "https://oidc.eks.eu-west-2.amazonaws.com/id/BF292F000B51B2A40D716C45519EB46E",
  "kubernetes.io": {
    "namespace": "default",
    "pod": {
      "name": "example-2-89fbdfcbb-lhrg6",
      "uid": "bcff40d4-1953-4b7a-b985-23e679745619"
    },
    "serviceaccount": {
      "name": "example-2",
      "uid": "71966509-2466-4721-9514-7d5574a2711c"
    }
  },
  "nbf": 1652352770,
  "sub": "system:serviceaccount:default:example-2"
}
workload-identity $
```

```
DISCOVERY DOCUMENT
{
  "issuer": "https://oidc.eks.eu-west-2.amazonaws.com/id/BF292F000B51B2A40D716C45519EB46E",
  "jwks_uri": "https://oidc.eks.eu-west-2.amazonaws.com/id/BF292F000B51B2A40D716C45519EB46E/keys",
  "authorization_endpoint": "urn:kubernetes:programmatic_authorization",
  "response_types_supported": [
    "id_token"
  ],
  "subject_types_supported": [
    "public"
  ],
  "claims_supported": [
    "sub",
    "iss"
  ],
  "id_token_signing_alg_values_supported": [
    "RS256"
  ]
}
workload-identity $
```

JWKS


```
{
  "keys": [
    {
      "kty": "RSA",
      "e": "AQAB",
      "use": "sig",
      "kid": "afe102b87b96fa2ad6ae37de27b52a3509f3482a",
      "alg": "RS256",
      "n": "gwfNbUq2oQAl89Dga72223Q9QE1yPskMC2HhPQ0ZdvCGmIqM2uGvZUUIc1SfrDeJRF2BwNTujaSi0JRTqLpyKOuq
pgGHcQygKFxr3J-0kLmotBRWUvpAz4u4wpancOa_02ZKR-bzoivqgK_qpTncvzoiOkgDxQVU37i5clOK_HLJvNTh-q_SDnU_xESc
CNZ-pY8vgiJfEbBQ3-nnhyciJy9jJuG5aWMUEJLj5TTgNSx2ElwPmVrXwAlstpLVDsHz3RuIcwBSmLE4wrfFWW87Z8HEnTuOh7wz
iDZJkzdNVEp-biC7DBjetSD7mZp_1M1mtF8uqVs3Q_mcjsAUvtCZXQ"
    }
  ]
}
```


workload-identity $

```
PAYLOAD
{
  "aud": [
    "demo"
  ],
  "exp": 1652353802,
  "iat": 1652352902,
  "iss": "https://oidc.prod-aks.azure.com/2dc26668-2e05-4bec-91ab-551d4660a786/",
  "kubernetes.io": {
    "namespace": "default",
    "pod": {
      "name": "example-2-89fbdfcbb-h7dhz",
      "uid": "3144dfd8-5234-454c-98b0-a045c3c8a305"
    },
    "serviceaccount": {
      "name": "example-2",
      "uid": "eb18c7ed-1665-4a46-a25e-a04d82bf7894"
    }
  },
  "nbf": 1652352902,
  "sub": "system:serviceaccount:default:example-2"
}
workload-identity $
```

```
DISCOVERY DOCUMENT
{
  "issuer": "https://oidc.prod-aks.azure.com/2dc26668-2e05-4bec-91ab-551d4660a786/",
  "jwks_uri": "https://oidc.prod-aks.azure.com/2dc26668-2e05-4bec-91ab-551d4660a786/openid/v1/jwks",
  "response_types_supported": [
    "id_token"
  ],
  "subject_types_supported": [
    "public"
  ],
  "id_token_signing_alg_values_supported": [
    "RS256"
  ]
}
workload-identity $
```

JWKS


```
{
  "keys": [
    {
      "use": "sig",
      "kty": "RSA",
      "kid": "BM25Mx8UjZEzXUciDU61ecYjWq5qq9Ch4HnWEyP5Dcw",
      "alg": "RS256",
      "n": "yIpi-g_1PQbeLuugwoagzOUX5dJVtVYhPvD33DEaX_BghMSwvm52DJLUC1fVK_2X-1YS5OWKBkXKUhY2UxGgeb8J
3LXObE9ZPSMt59by7cUqw3P3fKBb8vl0vNjb-fevXAVefXdVz5b6tUe_pi47vvvP3SwKolblU8Em47cAKZOzR1bRHXo6vFLR5v0k
UNSOlnzgAbsh5vAdsDvN3eDsNi_IUSNTZydE6bg-KkHSJrJ9gGCBjaykRMLkJh9KL-l1Dcw9AwnQe9xeOQQZHq5CCOriCL89uaR6
4PVBZNVo2wg0PnLpY207nAvhI1WSU7Xp9JLA6hSyqXGbWvWIEa9d95bnv6o1WcpJeDQuA4Gbta8cRC-MP9_51uCPBHs5KJBUkVH4
T8_FDIFIobZRa1tUMOL8gnGlgl1bJIL4um3GLl7AtVnu0bFME4vuVePEIevvftFEgYiUWsyaa7EHWU5EEUpnaCYJJsi_S7Moe6Lg
UmQpZUzLyhjpD75gYHFvtL734DzS8sKqrGGB_ZxTTeC6HmQ_QfAFM7wXZzv-Gei4ZxncQxWIvhfsNMOxnQodiwcSAuvsg7Sinezy
akfv4rvu9XJHY1p__irZL077vN_zVg-ZSmMiVBLWa0jUzfKacTPVXZ16zO57jKwWVS9FneVvOVjQF9UWwyEKrWowO9pbZ7c",
      "e": "AQAB"
    }
  ]
}
```


workload-identity $

What can I **do** with that then?

Web Identity Federation

- **Cloud** Identity and Access Management (**IAM**)
  - GCP
  - AWS
  - Azure (Preview)
- Configure
  - OIDC Provider
  - Role Mappings

controlplane

```
decoded = io.jwt.decode(token)
discovery = http.send({
    "url": concat("", [decoded[1].iss, "/.well-known/openid-configuration"]),
    "method": "GET"
}).body

jwks = http.send({
    "url": discovery.jwks_uri,
    "method": "GET"
}).raw_body

verified = io.jwt.verify_rs256(token,jwks)
subject = decoded[1].sub
audience = decoded[1].aud

workload-identity $
```

Open Policy Agent

controlplane

Your **Identity Provider** does not have to be Kubernetes

```
gcloud iam workload-identity-pools providers create-oidc "github-provider" \
  --location="global" \
  --workload-identity-pool="demo-pool" \
  --display-name="GitHub provider" \
  --attribute-mapping="google.subject=assertion.sub,attribute.actor=assertion.actor,attribute.aud=as
sertion.aud,attribute.repository=assertion.repository" \
  --issuer-uri="https://token.actions.githubusercontent.com"

gcloud iam service-accounts add-iam-policy-binding "demosa@***.iam.gserviceaccount.com" \
  --role="roles/iam.workloadIdentityUser" \
  --member="principalSet://iam.googleapis.com/$pool_name/attribute.repository/demo-org/demo-repo"


workload-identity $
```

controlplane

```
- id: 'auth'
  name: 'Authenticate to Google Cloud'
  uses: 'google-github-actions/auth@v0.4.0'
  with:
    token_format: 'id_token'
    workload_identity_provider: 'projects/***/locations/global/workloadIdentityPools/demo-pool/providers/github-provider'
    service_account: 'demosa@***.gserviceaccount.com'
    id_token_audience: demo

- name: 'Whoami'
  run: echo '${{ steps.auth.outputs.id_token }}' | jq -R 'split(".") | .[1] | @base64d | fromjson'

workload-identity $
```

controlplane

google-auth

succeeded 2 days ago in 4s

- Set up job
- Run actions/checkout@v3
- Authenticate to Google Cloud
- Whoami

```
1  ▶ Run echo '***' | jq -R 'split(".") | .[1] | @base64d | fromjson'
12 {
13   "aud": "demo",
14   "azp": "108524896999291150114",
15   "exp": 1651436309,
16   "iat": 1651432709,
17   "iss": "https://accounts.google.com",
18   "sub": "108524896999291150114"
19 }
```

- Post Run actions/checkout@v3
- Complete job

controlplane

What if I **need** more than that?

spiffe

SPIRE

Secure Production Identity Framework for Everyone (SPIFFE)

SPIFFE Identity

- spiffe://**acme.com**/billing/payments

SPIFFE Verifiable Identity Document (**SVID**)

- **JWT** SVID
- **X.509** SVID
- **Trust Bundle**

Workload Endpoint and Workload API

- Unauthenticated Identity Bootstrapping

SPIFFE Runtime Environment (SPIRE)

- Implementation of the SPIFFE standards
- **Attestation**
  - Nodes
  - Workloads
- **Registration** based on selectors (metadata)

controlplane

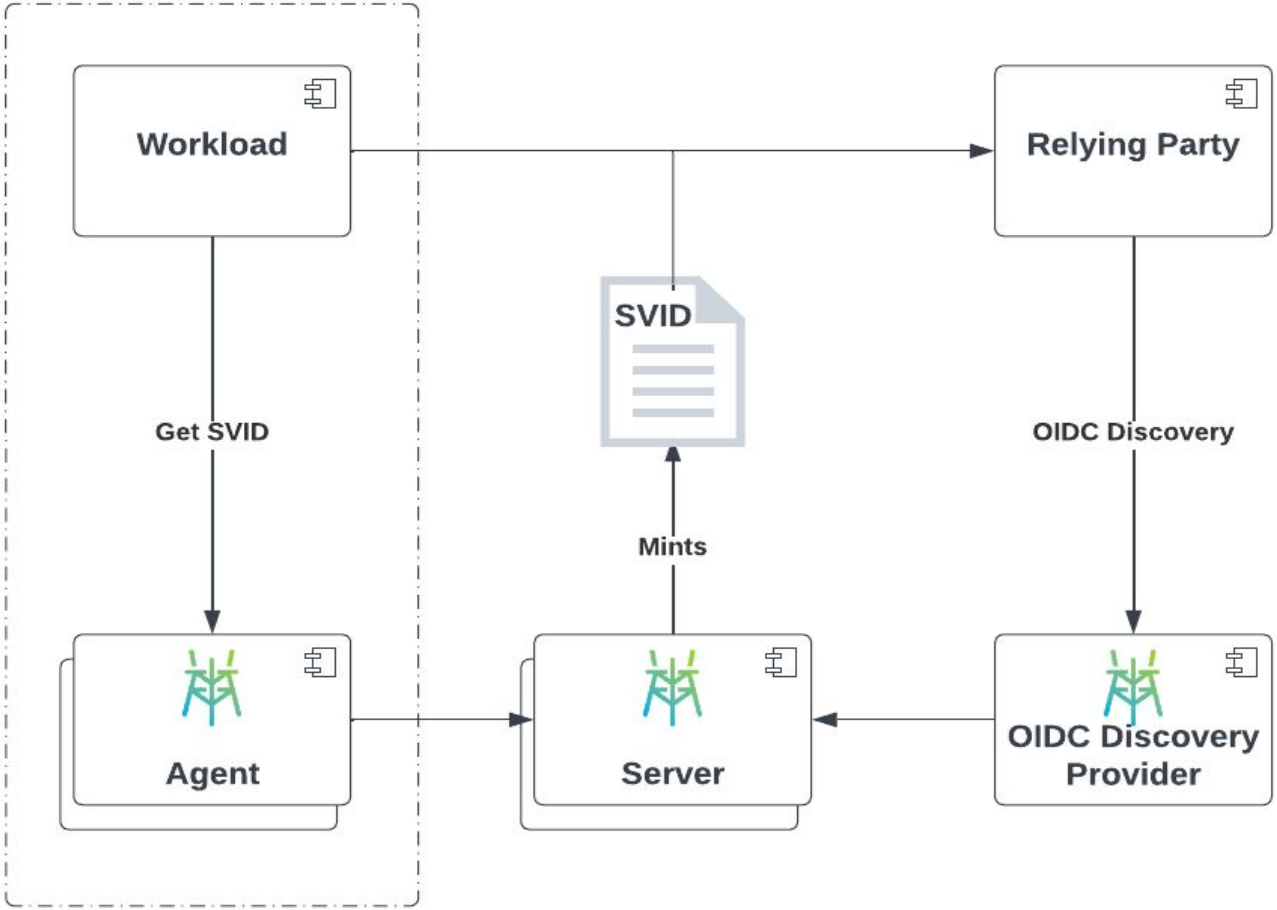
Node Attestation

- join_token
- x509pop
- sshpop
- k8s_sat
- k8s_psat
- gcp_iit
- aws_iid
- azure_msi

Workload Attestation

- unix
- docker
- k8s

controlplane



controlplane

```
HEADER
{
  "alg": "RS256",
  "kid": "k23CZK9ew7L2baDbbvgs1aJzjXSTneS7",
  "typ": "JWT"
}
PAYLOAD
{
  "aud": [
    "demo"
  ],
  "exp": 1652181606,
  "iat": 1652181306,
  "iss": "https://oidc.ricfeatherstone.com",
  "sub": "spiffe://ricfeatherstone.com/example-3"
}
SIGNATURE
"JwdNZvD6cZ5wWL7og10Y8XTB5SwLUP-4KBlpjy16xpQrKf1KfBIAA49l0CvGxt4iYWzr1OgLcDT286ezlRfi9O0SIUdInar5cQO5acoDuXEYKoSQeWbdz4
pJ1vahLO5L1XSA8A2dEUpcobFZmQAd9fYnbTc3pimjUOEgp2U0mZkWuk_xQbwGBApyVIj7f5BsF-IwGJLrXeHq2acYySFVgDYsjr0edHF5shDiL0ZxQ4aU9
cBed3ufXzrLlkbComn1y5HF8CPNlXBvfu7bbEIcK2psUlszcb8XFxVWC15Fx_EgloNNjHbXByDlIlR6n_t7Pd1s1xpMKrwD2aLyYxj4bw"
workload-identity $
```

```
DISCOVERY DOCUMENT
{
  "issuer": "https://oidc.ricfeatherstone.com",
  "jwks_uri": "https://oidc.ricfeatherstone.com/keys",
  "authorization_endpoint": "",
  "response_types_supported": [
    "id_token"
  ],
  "subject_types_supported": [],
  "id_token_signing_alg_values_supported": [
    "RS256",
    "ES256",
    "ES384"
  ]
}
workload-identity $
```

JWKS


```
{
  "keys": [
    {
      "use": "sig",
      "kty": "RSA",
      "kid": "ag8gAREwfjqL1JBxXDwiXRJgpMQuiPlX",
      "n": "ycnvlGNMWj-NBlNtCaKyfkMNwjeKwOYviWJ7rY-qWP_XeSJgx9zJMYG3YOrCLOmBOx9o6m6RZlqhzP52OXObIHwr_Za-4S-L7jYlMj7aF86
s9nj6hRNhXmnIGNPY3cGzRFbMkvZ0oq16csbRMF5XGpaRxq1FKW7gZ3u6Vh4NQ3C-iPqu2ZQI-piHsMrzK1F-qJcoAhHk2ixED0qSs6LVwuS8wAJEPajkSg
NZhu4k-i84D9JKpg3-ay6PBHd5auWA8xfRdHt3x-_I5VD0vVkZnn0fwYNVJqAfNPbvQG2RxdhQDFxa0IQsrE6avGA-Fe3ySsxWqsluU71sH-kG8D71Ow",
      "e": "AQAB"
    }
  ]
}
```


workload-identity $

What about those **X.509** SVIDs?

```
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: CN = ricfeatherstone.com
    Validity
        Not Before: May 10 11:48:58 2022 GMT
        Not After : May 10 12:49:08 2022 GMT
    Subject: C = US, O = SPIRE
    Subject Public Key Info:
        Public Key Algorithm: id-ecPublicKey
            Public-Key: (256 bit)
            pub:
                04:e8:e6:af:91:d4:27:ca:79:a7:0e:f0:42:cc:64:
                e4:dc:0b:46:7c:2a:e3:85:a3:cf:cf:c3:96:6e:1c:
                7d:11:b5:50:ab:d8:17:ea:36:cd:28:e0:ce:7b:21:
                b9:24:7c:d4:05:e4:42:b6:54:54:22:b6:65:33:34:
                a5:eb:e0:6c:e6
            ASN1 OID: prime256v1
            NIST CURVE: P-256
    X509v3 extensions:
        X509v3 Key Usage: critical
            Digital Signature, Key Encipherment, Key Agreement
        X509v3 Extended Key Usage:
            TLS Web Server Authentication, TLS Web Client Authentication
        X509v3 Basic Constraints: critical
            CA:FALSE
        X509v3 Subject Key Identifier:
            64:7C:A9:2E:6C:F3:6B:A7:62:26:01:98:31:29:19:10:B2:DA:16:86
        X509v3 Authority Key Identifier:
            keyid:41:E3:BD:6A:B6:C4:08:D5:54:39:8A:AB:36:F1:1F:26:E0:EA:12:81

        X509v3 Subject Alternative Name:
            URI:spiffe://ricfeatherstone.com/example-3
Signature Algorithm: sha256WithRSAEncryption
     0b:dc:c2:3f:05:0d:4d:30:35:e1:3c:1f:e7:02:af:ff:fb:09:
```

You said something about **Hardware Root of Trust**?

Keylime

TPM

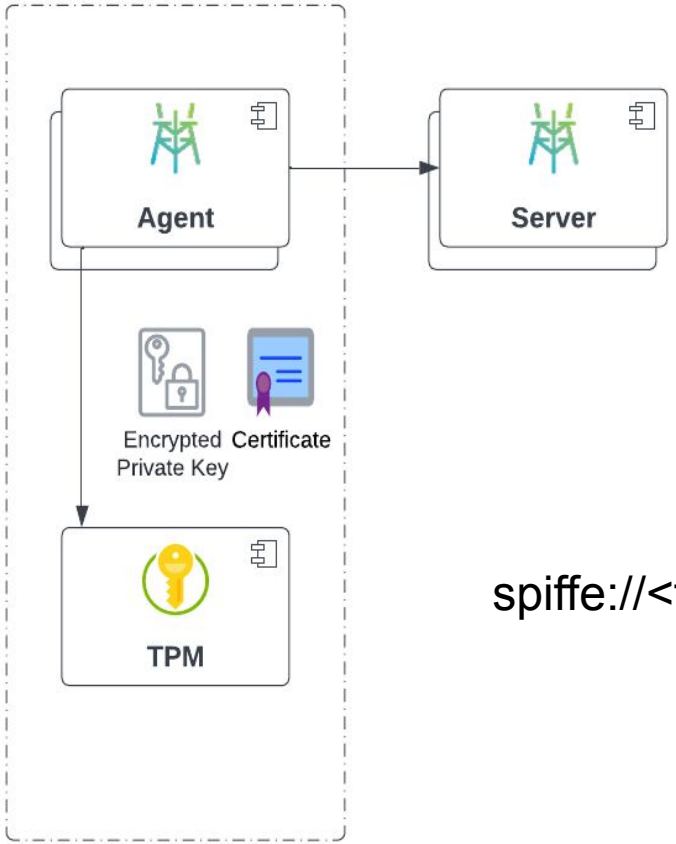
- Hardware, Firmware or Virtual **Crypto-processing** Device
- **Endorsement Key** (EK) burned in a manufacturing time
- Generated Keys cannot leave the TPM **Unencrypted**
- **Platform Configuration Registers** (PCR)

Keylime

- **Remote Attestation**
- Linux **Integrity Measurement Architecture** (IMA)
- **Continuous Verification**
- Linked to **Attestation Key** (AK) of a **specific** machine

https://keylime.dev/

controlplane



TPM DevID Attestation

- Like x509_pop attestation
- Private Key encrypted by TPM
- Physically bound to specific machine

spiffe://<trust domain>/spire/agent/tpm_devid/<fingerprint>

https://www.youtube.com/watch?v=SSdWeDf02TM

controlplane

Takeaways

- You might already have a **Workload Identity** that satisfies your use case
- Does your Kubernetes cluster work as your **Identity Provider**?
- If you are using **Service Account Tokens** for Workload Identity
  - Make sure they are **Bound Service Account Tokens**
  - Always use a **Secure Channel** for communication
- **OIDC Discovery** and **Web Identity Federation** cover a number of use cases
- Think about the boundaries of your **Trust Domains**
- Consider **SPIFFE** and **SPIRE**
- Investigate **TPM**s and **Keylime**

controlplane