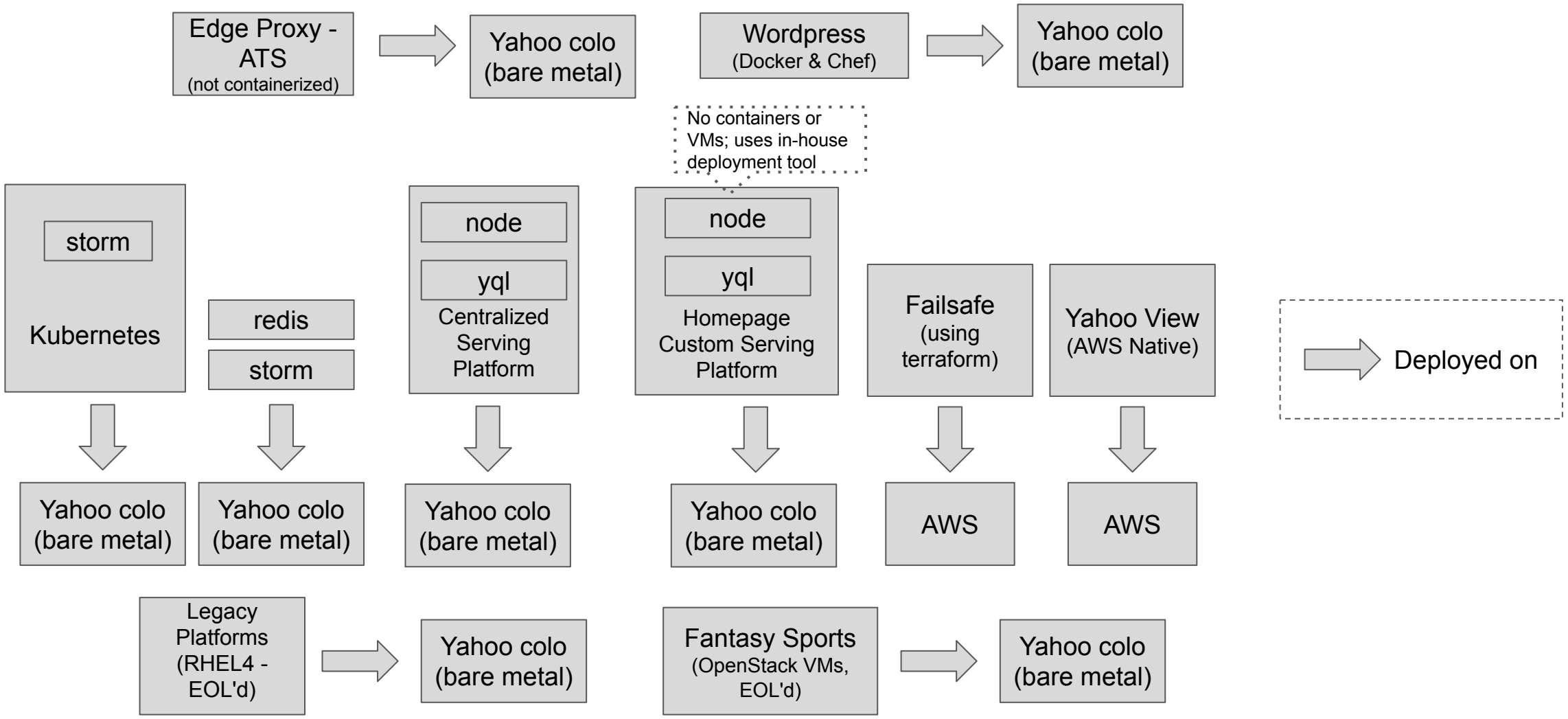
K8s & Istio @ Yahoo

Suresh Visvanathan

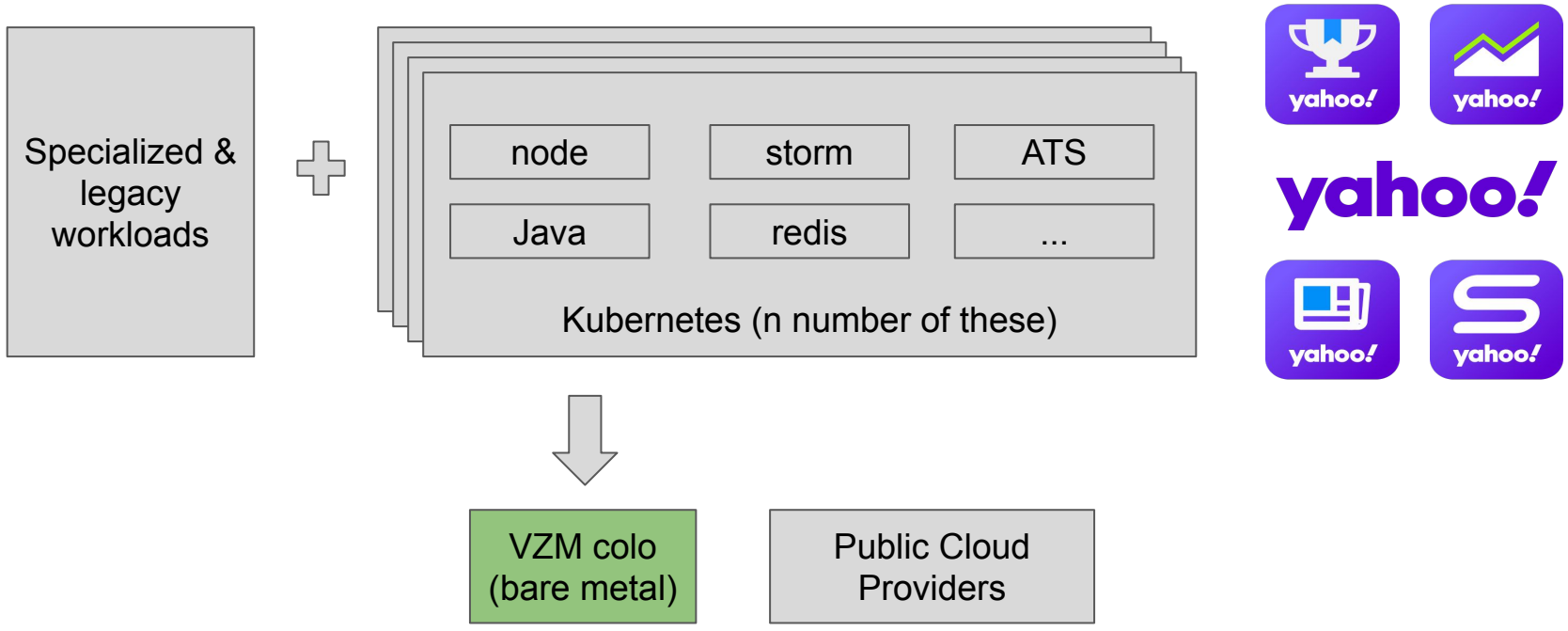
Mrunmayi Dhume

Yahoo Media circa 2017...



… and today!

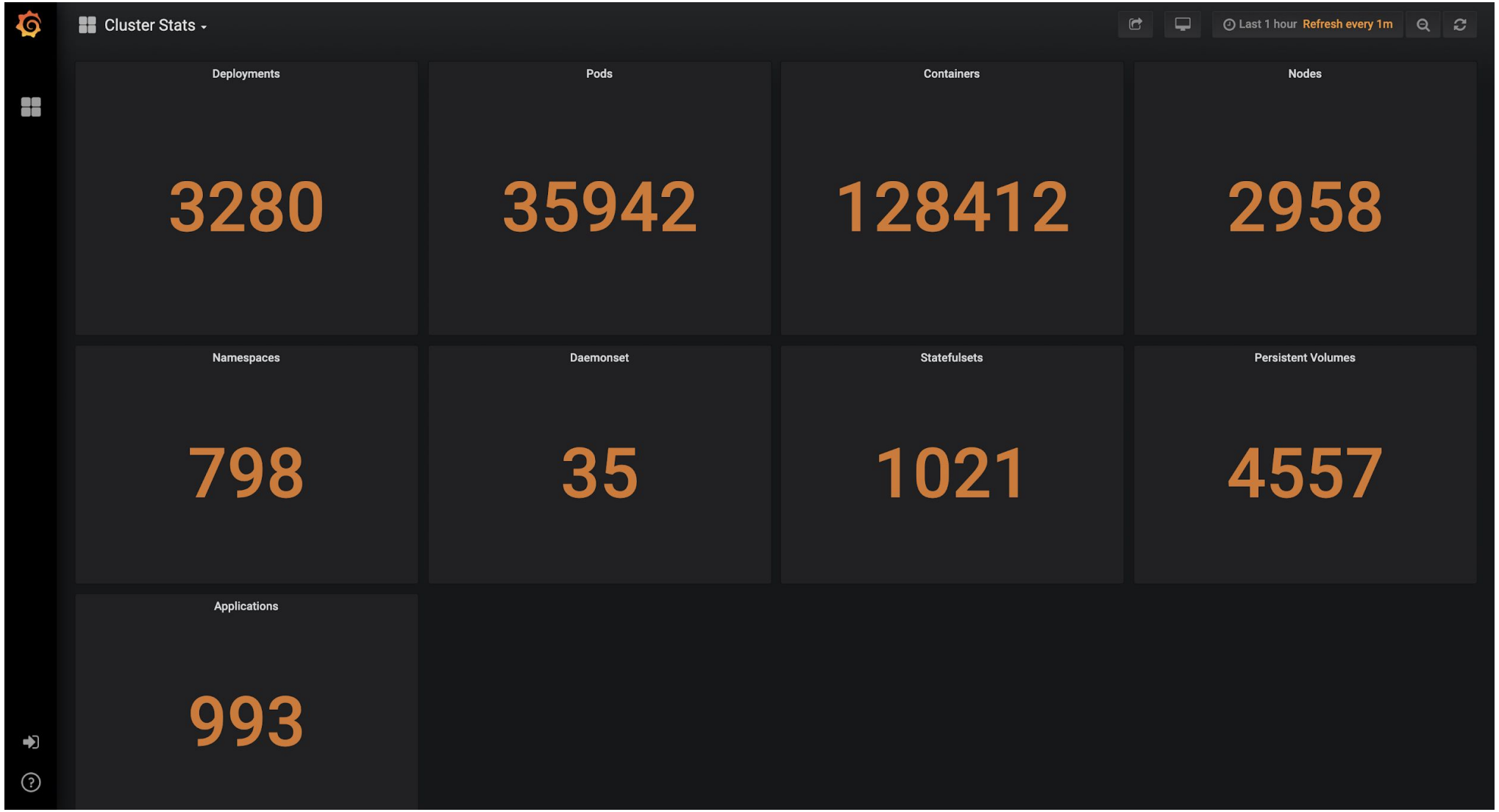
Containerized + K8s — Goal for Verizon Media



Number Crunching

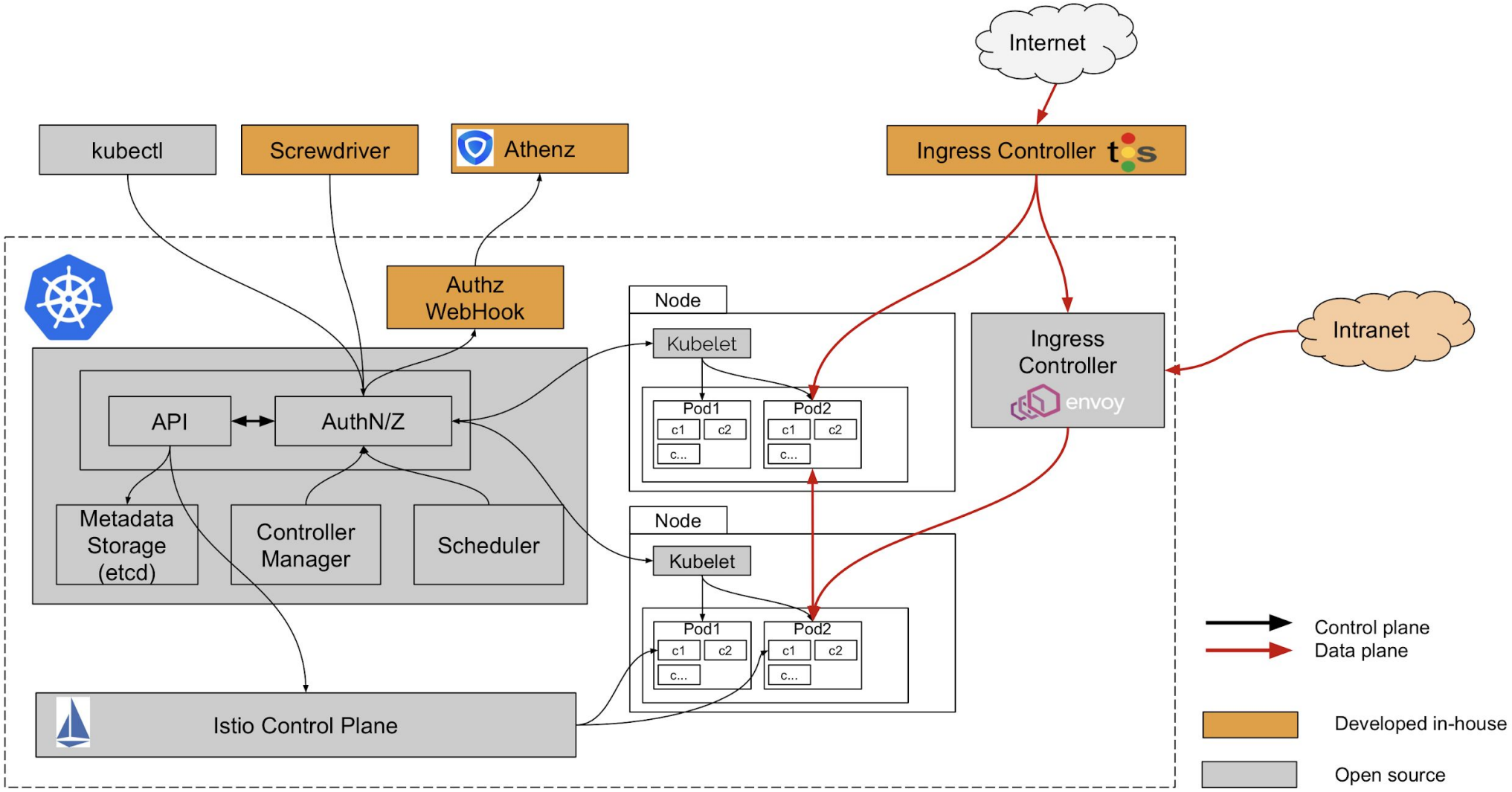
- 990+ apps
- 1K+ stateful apps
- 18 prod grade clusters
- 7 data centers
- 2900+ nodes
- ~1.5M RPS peak on Ingress stack

Key Prod Cluster Statistics



Yahoo K8s Ecosystem

How it all fits together

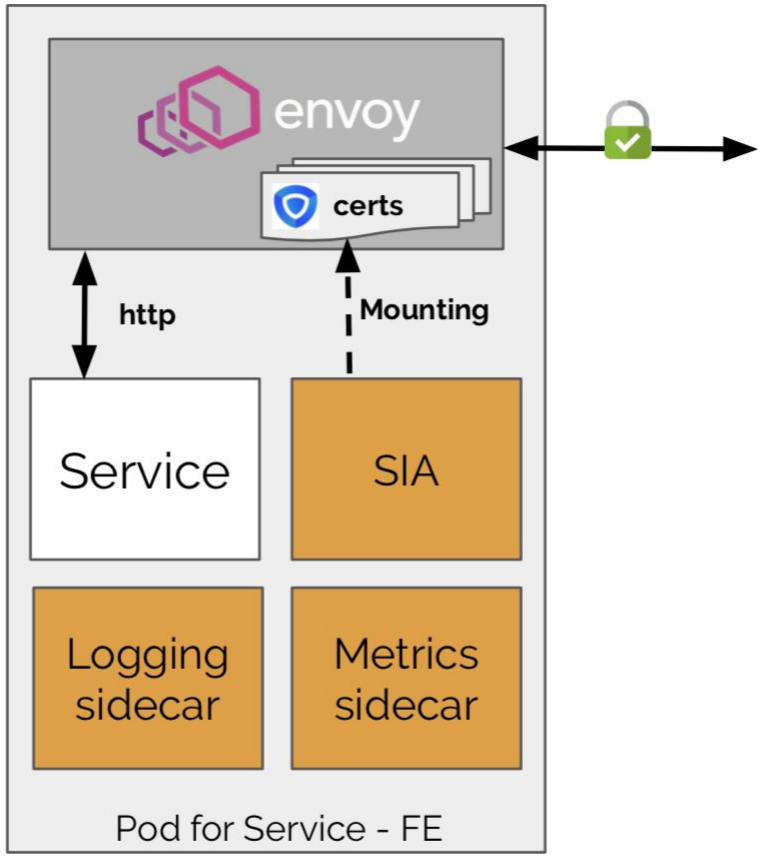


Athenz

“Athenz is an open source platform for X.509 certificate based service authentication and fine grained role based access control in dynamic infrastructures”

https://www.athenz.io/

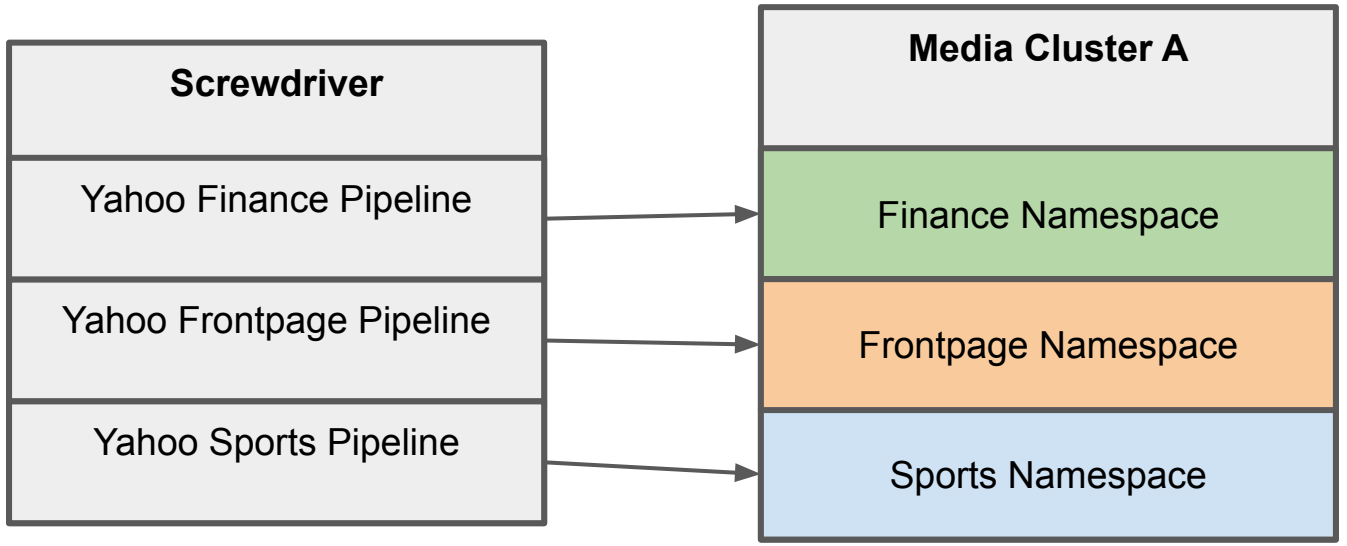
What's in a Pod?



Developed in-house

Auth and Multi-Tenancy

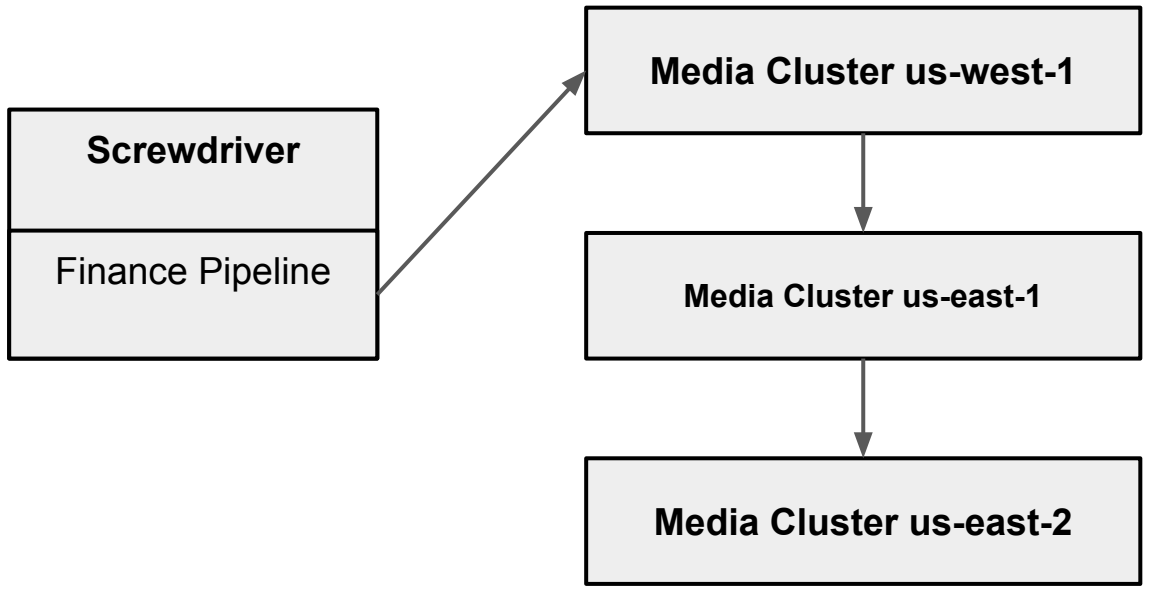
Soft Multi-tenancy



Auth and Multi-Tenancy

Soft Multi-tenancy

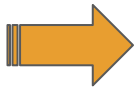
Sequential deployment to multiple clusters across data centers reduces the blast radius



Auth and Multi-Tenancy

Namespace scoped RBAC defined in ATHENZ

K8s Namespace
finance-k8s



Athenz Domain
finance.k8s

Role & Member definition in Athenz

Corresponding Policy definition in Athenz

ALLOW	list	finance.k8s:role.k8s_developer	finance.k8s:pods
ALLOW	get	finance.k8s:role.k8s_developer	finance.k8s:pods
ALLOW	create	finance.k8s:role.k8s_developer	finance.k8s:pods.exec
ALLOW	get	finance.k8s:role.k8s_developer	finance.k8s:pods.log
ALLOW	get	finance.k8s:role.k8s_developer	finance.k8s:deployments
ALLOW	list	finance.k8s:role.k8s_developer	finance.k8s:deployments

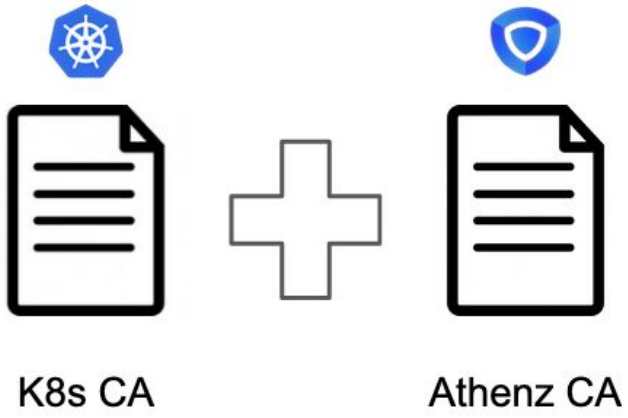
Auth and Multi-Tenancy

User Authorization

Athenz X.509 User certificates

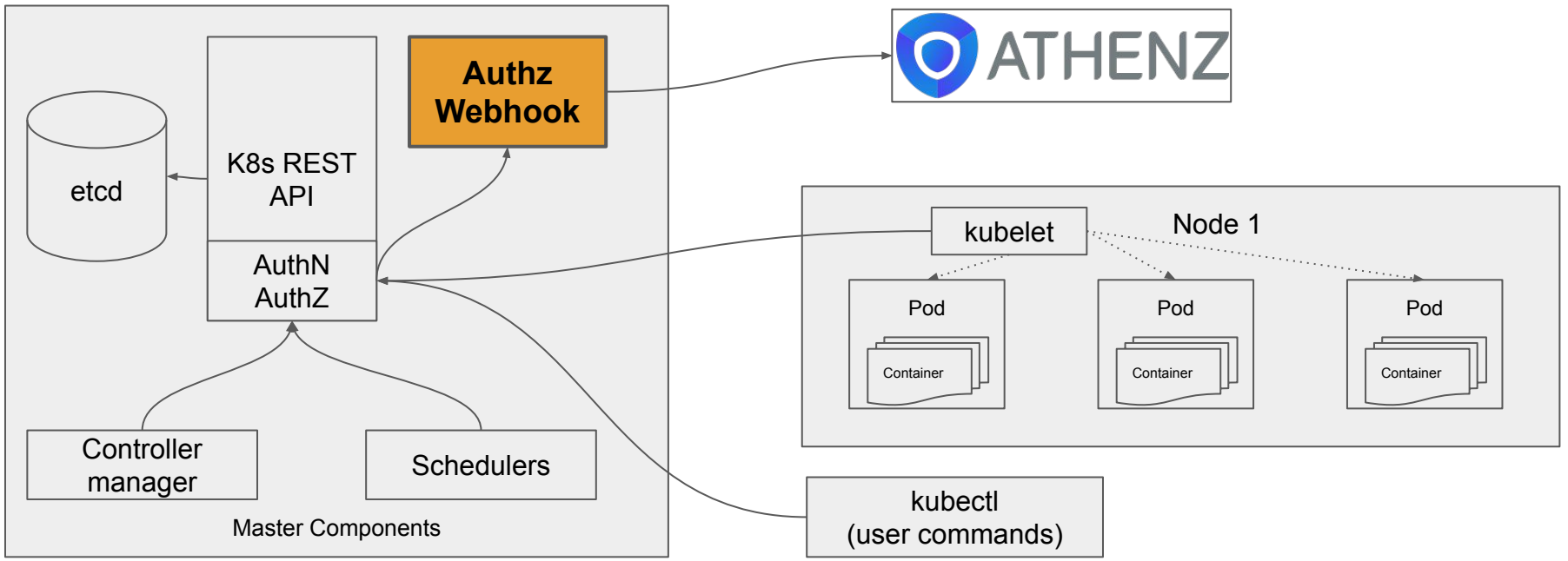
```
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=CA, L=Sunnyvale, O=Oath Inc.,
OU=us-west-2, CN=Athenz AWS CA
        Validity
            Not Before: Nov 12 00:13:54 2019 GMT
            Not After : Nov 12 02:13:54 2019 GMT
        Subject: C=US, O=Oath, OU=Athenz, CN=user.sureshv
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
```

K8s API server Client CA comprises K8s & Athenz CA



Auth and Multi-Tenancy

Authorization webhook



Auth and Multi-Tenancy

Authorization webhook receives a SubjectAccessReview object from K8s API server

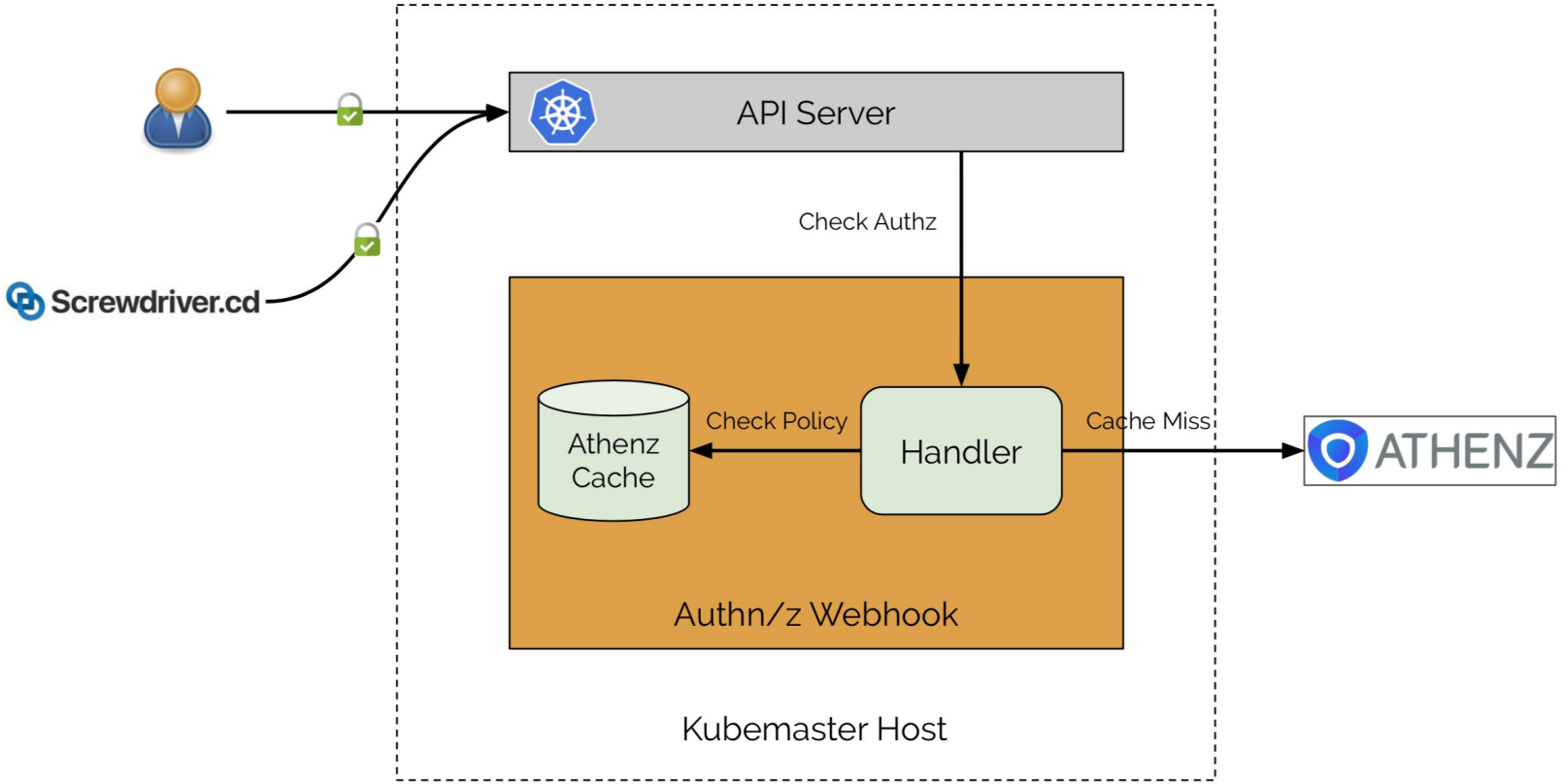
```
{
  "user":"user.sureshv",
  "group":["athenz","system:authenticated"],
  "resourceAttributes":{
    "namespace":"finance-k8s",
    "verb":"get",
    "version":"v1",
    "resource":"deployments"
  }
}
```

It translates and forwards the check to Athenz

`https://athenz:4443/zms/v1/access/get/finance.k8s:deployments?principal=user.sureshv`

Auth and Multi-Tenancy

Authorization webhook workflow

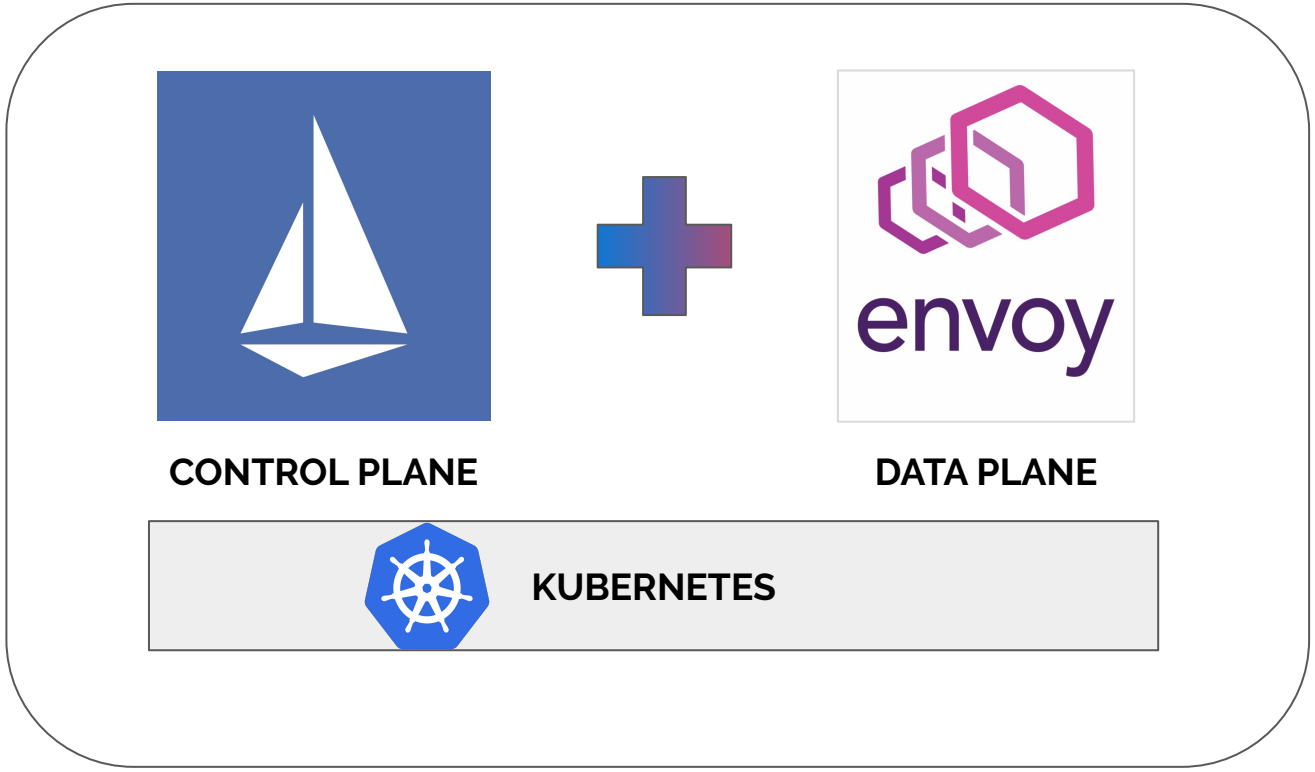


Yahoo Istio Ecosystem

Why Service Mesh?

- Makes the network transparent to application developers
- Language agnostic
- Zero application code change required
- Extensive features (to list a few)
  - Rich protocol support
  - Fine grained tracing and observability
  - Security: Transport level security (TLS).
  - Resiliency - Circuit-breaking, retries and timeouts, fault injection, fault handling, load balancing and failover.

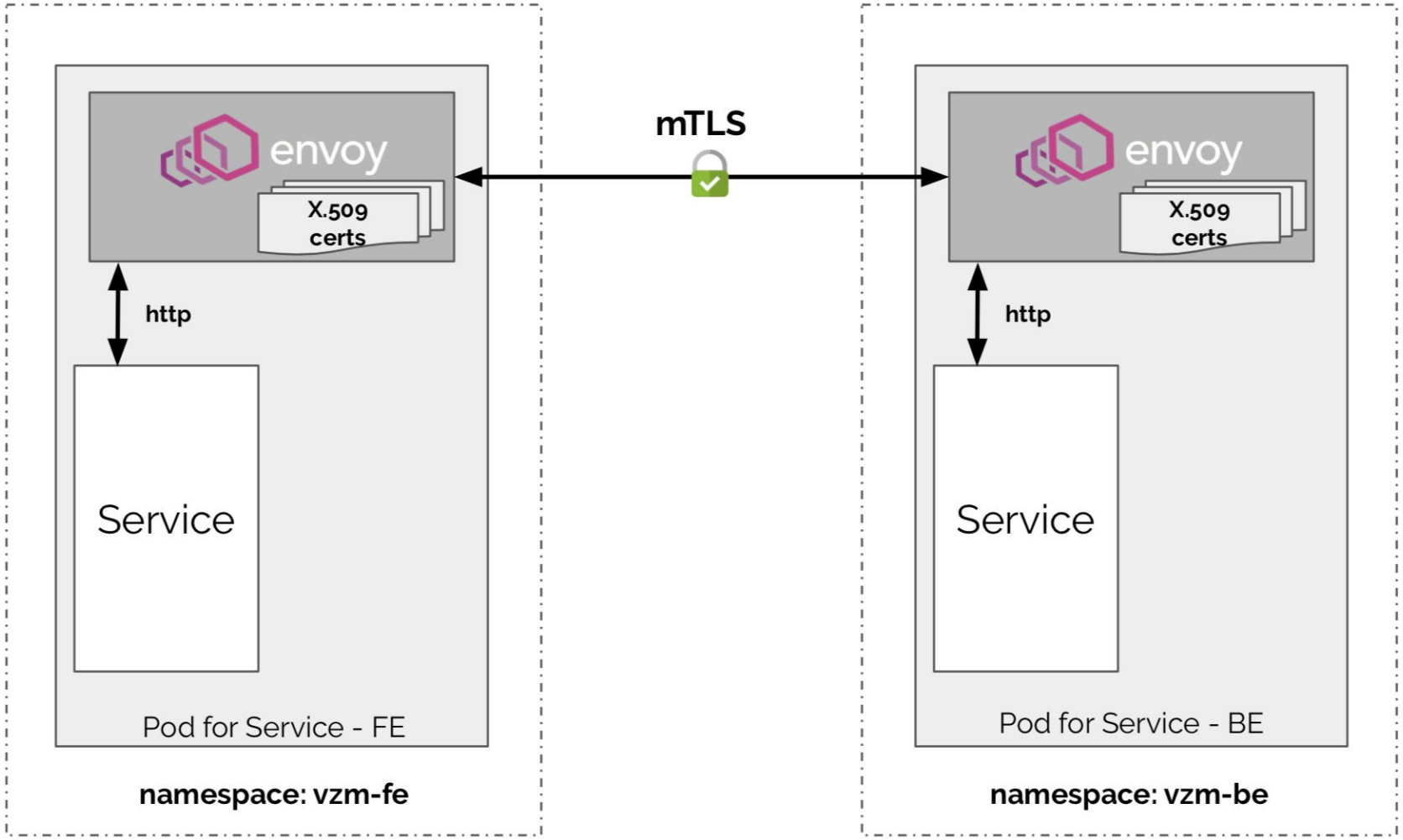
Istio Service Mesh



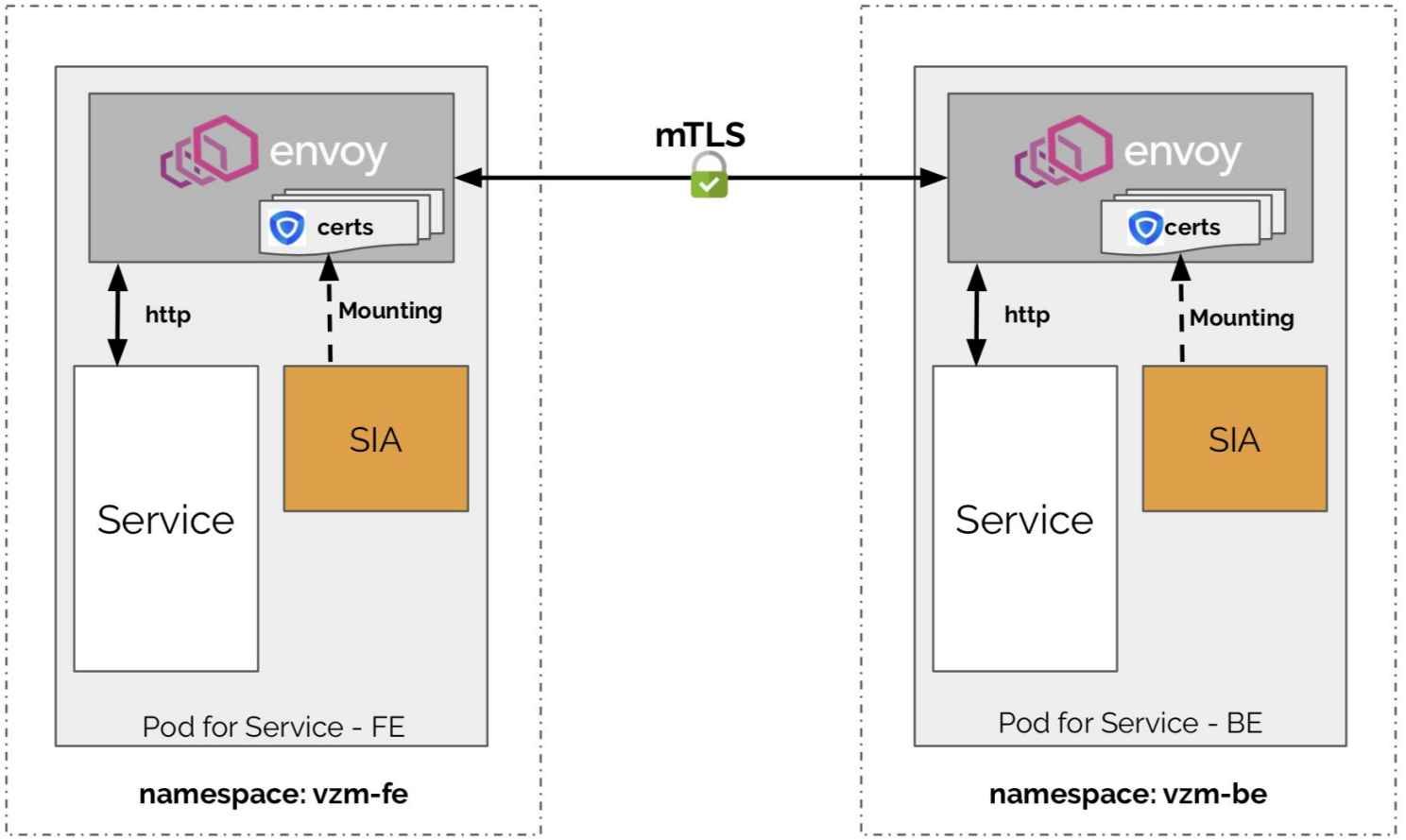
Mutual TLS everywhere

- **Goal:** Secure service to service communication via X.509 certificate based mutually authenticated and encrypted connection
- Leverage Istio and Envoy to handle mTLS
  - Uniform and consistent solution
  - Prevent duplication of code and effort caused by custom solutions from each application

Secure Communication with Istio



mTLS using Istio & Athenz

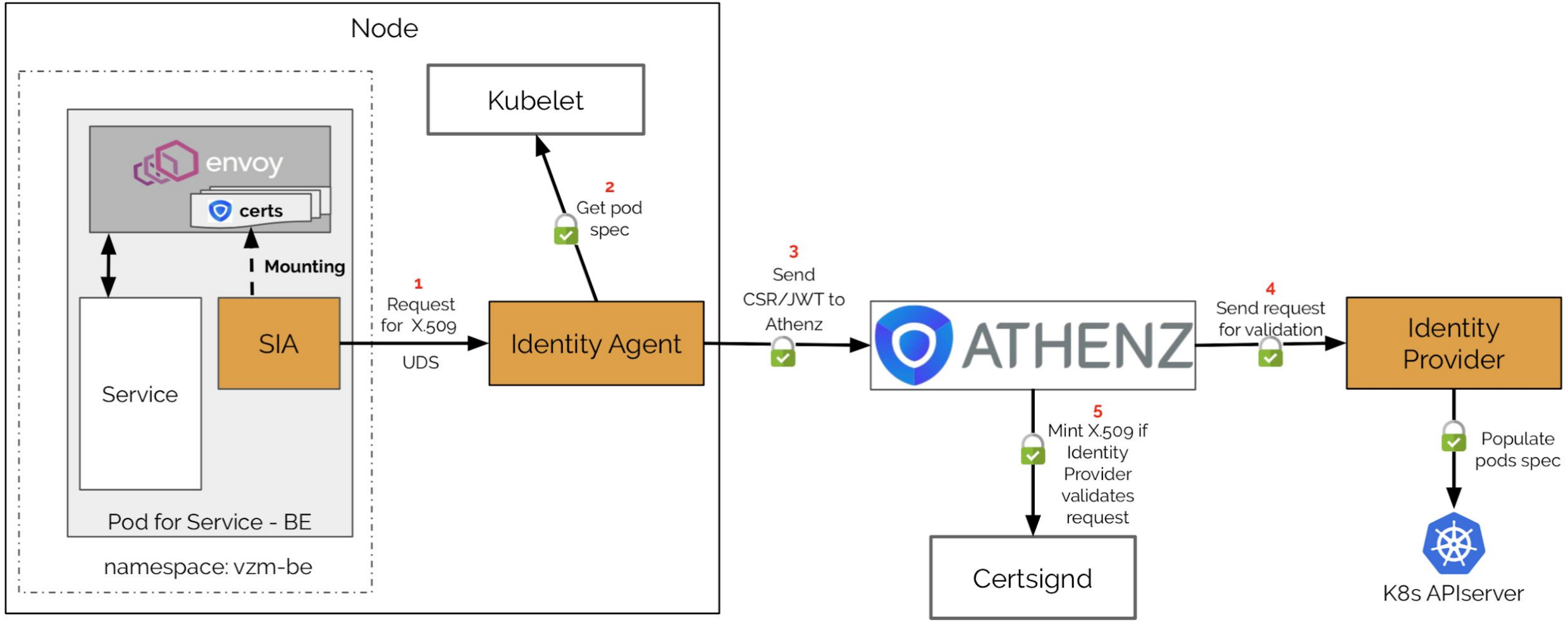


Developed in-house

Yahoo K8s Identity Provider

- Yahoo K8s open-source identity provider system integrated with Athenz does the following:
  - Provide a unique identity (Athenz X.509 certificate) to every workload.
  - Periodic refresh cycle

How it works


Node
Kubelet
envoy
certs
Mounting
2
Get pod spec
1
Request for X.509
UDS
SIA
Service
Identity Agent
Pod for Service - BE
namespace: vzm-be
3
Send CSR/JWT to Athenz
ATHENZ
4
Send request for validation
Identity Provider
5
Mint X.509 if Identity Provider validates request
Certsignd
Populate pods spec
K8s APIserver

Istio mTLS Authentication

- With mTLS an additional Envoy Filter is added for performing authentication
  - Responsible for verifying incoming client certificates
  - Verifies presence of DNS URI field in X.509 certificate
  - Internally sets a `principal` value based on the SPIFFE URI

Athenz X.509 with SPIFFE


```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            c7:ba:6a:bd:d2:ae:1c:75:00:1b:ac:78:1d:68:32:1f
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=CA, L=Sunnyvale, O=Yahoo! Inc., OU=Athenz Tcv, CN=Yahoo Athenz CA
        Validity
            Not Before: Nov 12 02:45:41 2018 GMT
            Not After : Dec 12 03:45:41 2018 GMT
        Subject: C=US, ST=CA, O=Oath Inc., OU=k8s.omega.canary1-tw1.kube-yahoo.identityd, CN=home.sureshv.fe.productpage
```

```
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Client Authentication, TLS Web Server Authentication
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Alternative Name:
                DNS:productpage.home-sureshv-fe.media.yahoo.cloud, DNS:productpage.home-sureshv-fe.canary1-tw1.media.yahoo.cloud,
008cfacc4370.instanceid.athenz.media.yahoo.cloud, IP Address:10.237.132.146, URI:spiffe://home.sureshv.fe/sa/productpage
```

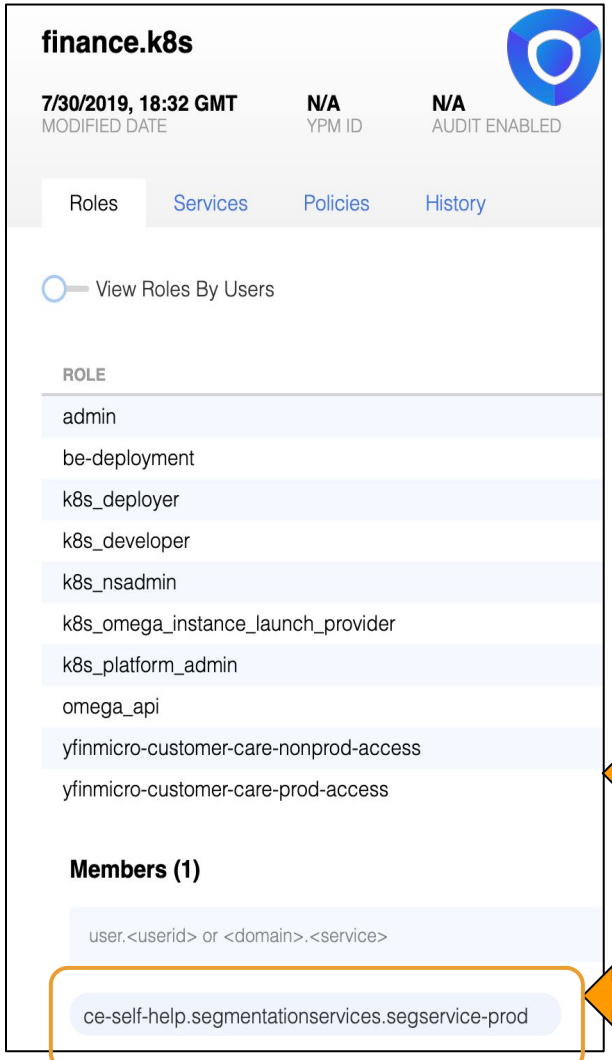
Istio mTLS Authorization

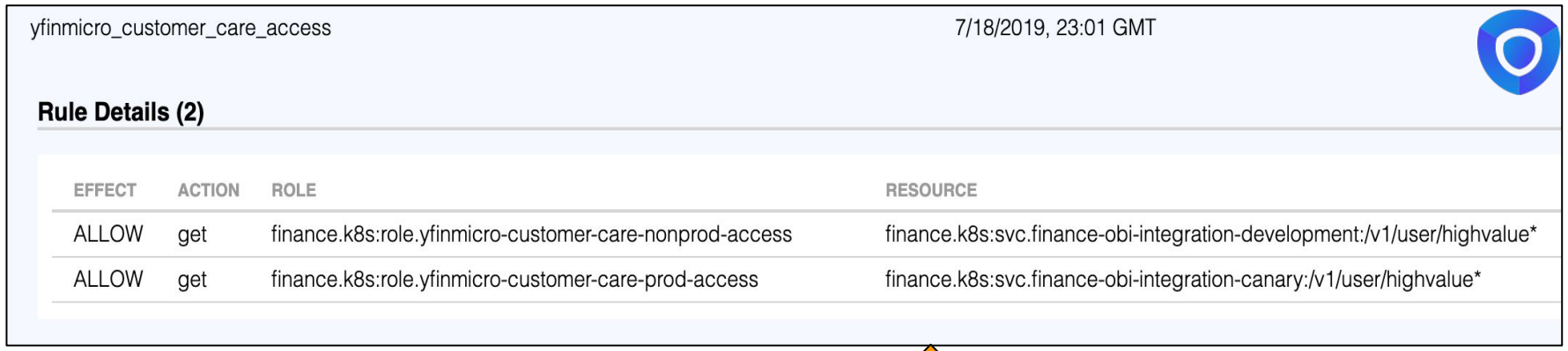
- Istio utilizes the native Envoy RBAC filter
- Envoy RBAC filter enforces authorization by:
  - Extracting SPIFFE URI from X.509 certs to set as the principal
  - Evaluating against the defined policies
  - Returning allowed/denied response

Istio RBAC & Athenz Integration

- In Istio, RBAC definitions are through Custom Resources (CR) - ServiceRole and ServiceRolebinding
- Istio Pilot maps Istio CRs into Envoy RBAC configuration
- **Athenz Istio Authz controller**:
  - Maps Athenz roles and policies into Istio Authz CRs
  - Keeps in sync with Athenz role/policy changes
  - Creates corresponding Istio CRs

Service RBAC definition in Athenz





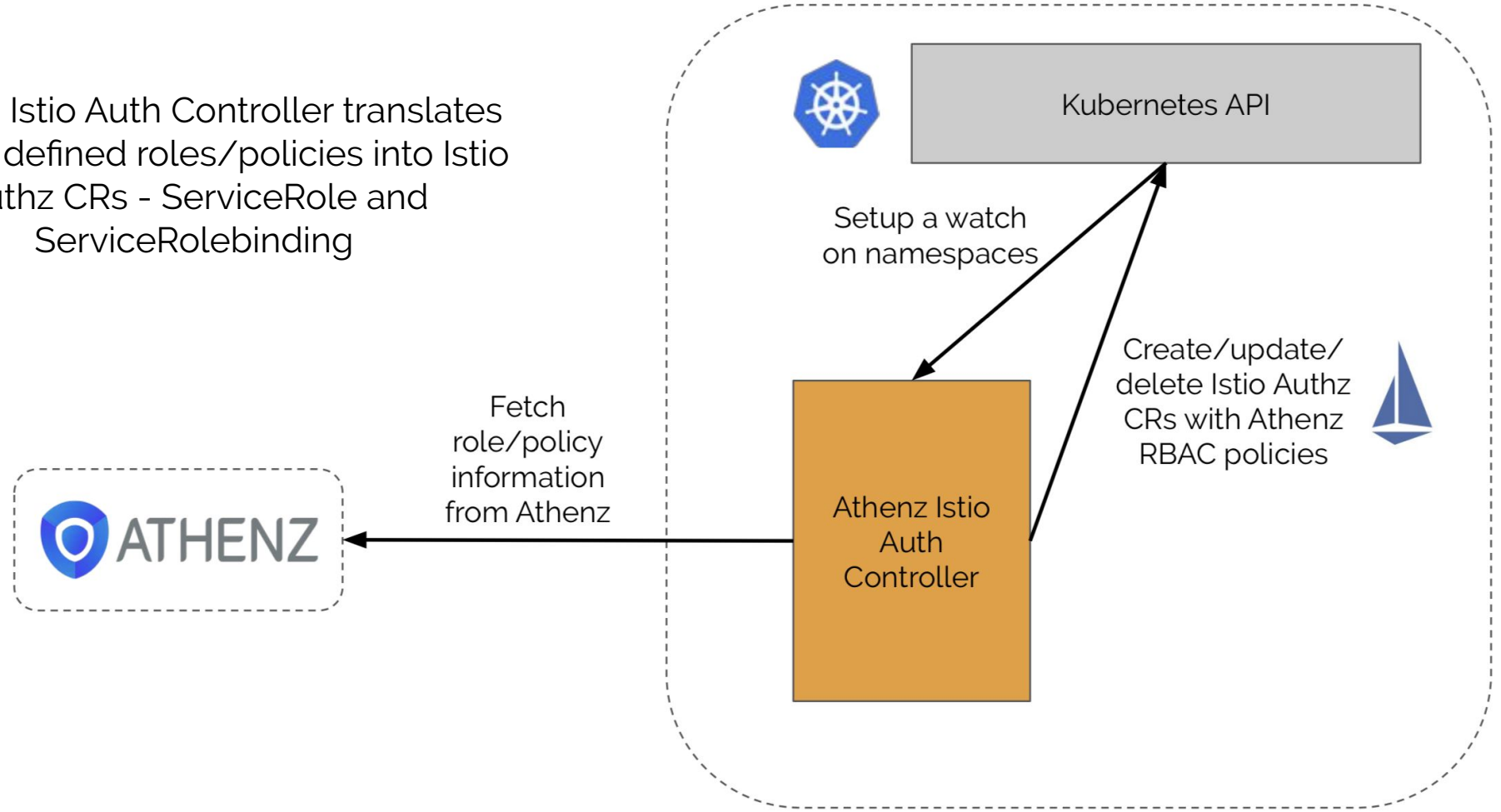
Role definition

Policy definition

Role member

Athenz Istio Authz controller

Athenz Istio Auth Controller translates Athenz defined roles/policies into Istio Authz CRs - ServiceRole and ServiceRolebinding



Yahoo K8s Developer Experience

Design Goals

- Simplicity
  - Easy onboarding
  - No requirement of in-depth K8s knowledge
  - Simple YAML interface
- Flexibility
  - Ability to override default settings
- Stability
  - Comprehensive acceptance tests for every change

Simple YAML Interface

- Settings clearly describe use cases
- No K8s specific domain knowledge required

```
template: nodejs-app:stable
appName: foo-app
maintainer:
foo-team@verizonmedia.com
baseimage: nodejs:10.0.170725-3
namespace: foo-ns
routing:
  ports: 4080
  aliases: xyz.yahoo.com
sidecars:
  logging:
    instance: foo.splunk:443
  metrics:
    namespace: foo
```

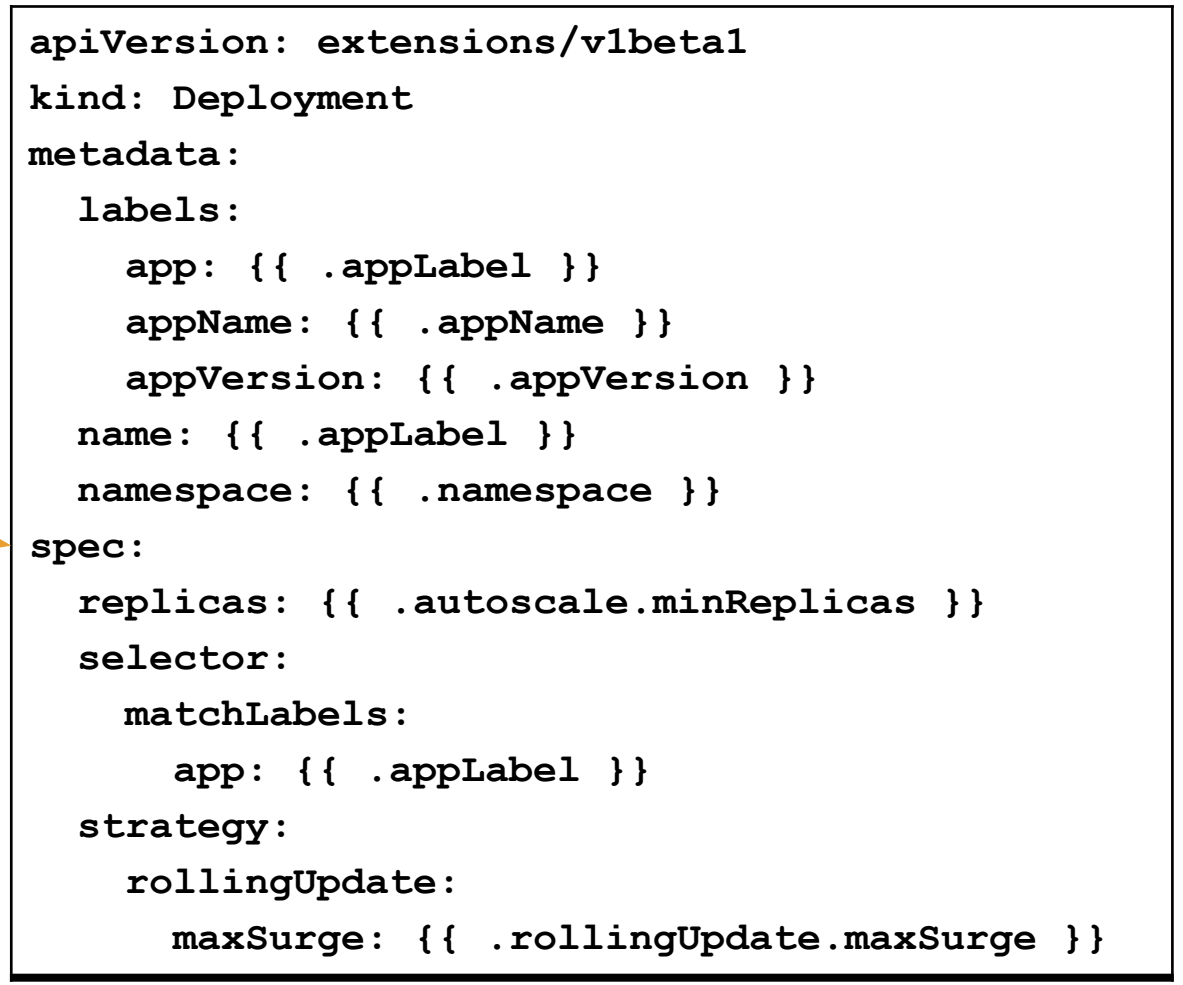
```
jobs:
  stage:
    environment: stage
    colo: west-1
  production-east-1:
    environment: production
    colo: east-1
  production-west-1:
    environment: production
    colo: west-1
```

application.yaml

Standardized templates

- Proprietary templating engine

```
|--- templates
|     |--- autoscale.yaml
|     |--- configmap.yaml
|     |--- deployment.yaml
|     |--- ingress.yaml
|     |--- service.yaml
|     |--- istio
|     |     |--- virtualService.yaml
|     |     |--- destinationRule.yaml
|     |     |--- sidecar.yaml
|--- default.yaml
```



```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  labels:
    app: {{ .appLabel }}
    appName: {{ .appName }}
    appVersion: {{ .appVersion }}
  name: {{ .appLabel }}
  namespace: {{ .namespace }}
spec:
  replicas: {{ .autoscale.minReplicas }}
  selector:
    matchLabels:
      app: {{ .appLabel }}
  strategy:
    rollingUpdate:
      maxSurge: {{ .rollingUpdate.maxSurge }}
```

Generated Artifacts for K8s Deployment

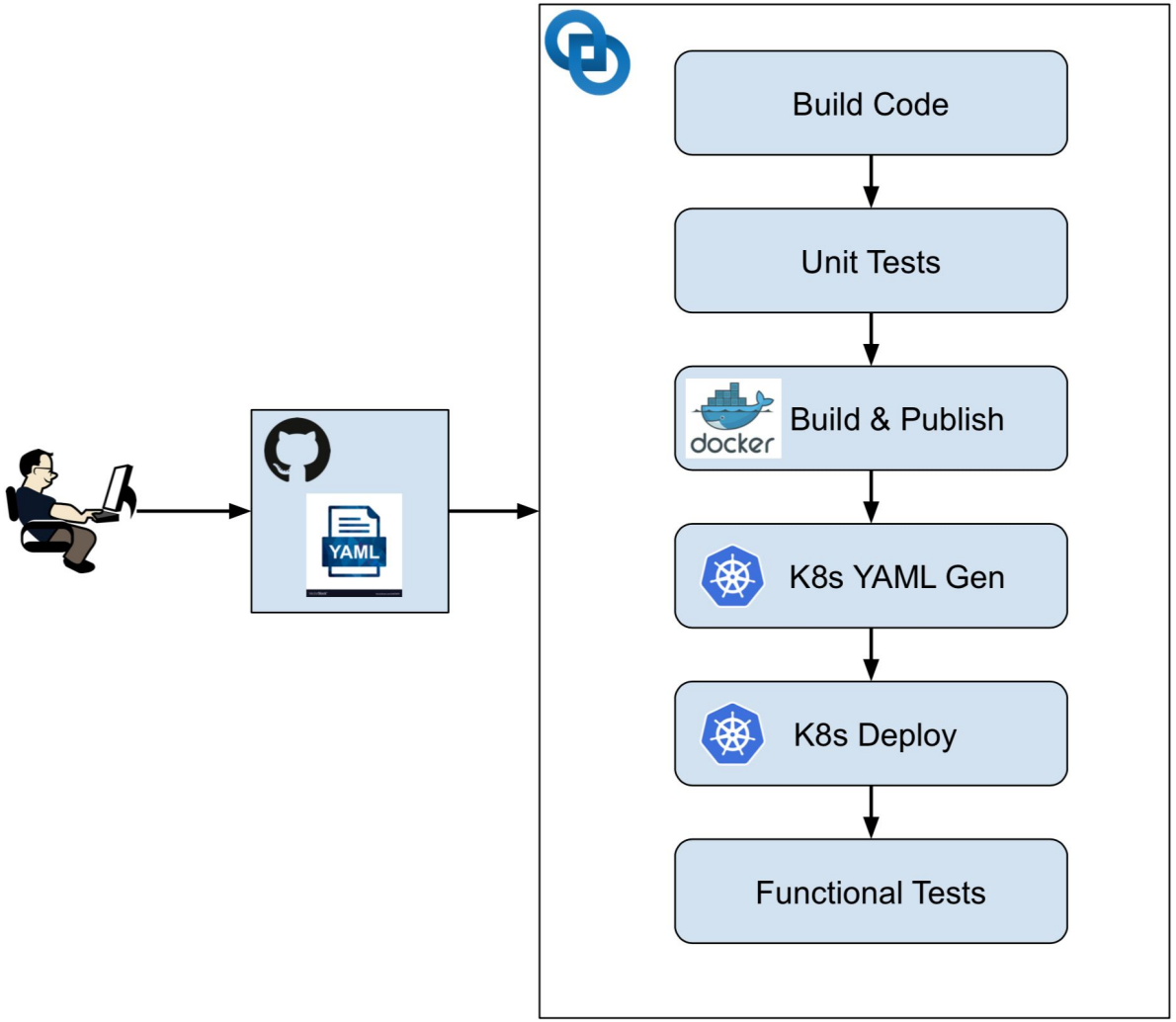
KubeCon CloudNativeCon North America 2019

foo-app-1.0.61.15.tar

```
generated
|--- Dockerfile
|--- stage
|     |--- autoscale.yaml
|     |--- configmap.yaml
|     |--- deployment.yaml
|     |--- ingress.yaml
|     |--- service.yaml
|     |--- istio
|     |     |--- policy.yaml
|     |     |--- virtualService.yaml
|     |     |--- gateway.yaml
|     |     |--- sidecar.yaml
|     |     |--- destinationRule.yaml
|--- production-east-1
|     |--- autoscale.yaml
|     |--- configmap.yaml
|     |--- deployment.yaml
|     |--- ingress.yaml
|     |--- service.yaml
|     |--- istio
|     |     |--- policy.yaml
|     |     |--- virtualService.yaml
|     |     |--- gateway.yaml
|     |     |--- sidecar.yaml
|     |     |--- destinationRule.yaml
```

```
appVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: foo-app--production-east-1
    appVersion: 1.0.61.15
  name: foo-app--production-east-1
  namespace: foo-ns
...
spec:
  replicas: 1
  selector:
    matchLabels:
      app: foo-app--production-east-1
  strategy:
    rollingUpdate:
      maxSurge: 15%
      maxUnavailable: 15%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: foo-app--production-east-1
    spec:
      containers:
      - image: docker.yahoo.com/yahoo-cloud/foo-app-node10:1.0.61.15
...
      - image: docker.yahoo.com/yahoo-cloud/splunk-base
...
```

Standardized CI/CD Pipeline Flow



Sample Developer Use Case

If a developer wants to:

- Deploy a Node.js 10 app
- Routing through Istio Ingress on port 443
- With an Istio sidecar
- With Strict mutual TLS enabled
- Restricting egress traffic to known endpoints

Above requirements can all be summarized in the `application.yaml` defined on the right

```
template: nodejs-app:stable
appName: foo-app
namespace: foo-ns
baseimage: nodejs:10.0.170725-3
routing:
  istio: true
  ports: 443
  aliases: "foo-app.media.yahoo.com"
jobs:
  production-west-1:
    sidecars:
      istio:
        resources:
          limits:
            cpu: "4"
            memory: "500Mi"
        authn: STRICT_MTLS
        outboundTrafficPolicy: REGISTRY_ONLY
        egress:
        - hosts:
            - finance-k8s/*
            - homepage-k8s/*.media.yahoo.com
```

Behind the scenes …


```
appVersion: apps/v1
kind: Deployment
metadata:
  labels:
...
spec:
  template:
    spec:
      containers:
      - image: docker.io/istio/proxyv2
```

```
apiVersion: networking.istio.io/v1alpha3
kind: Sidecar
...
spec:
  workloadSelector:
    labels:
      app: foo-app.foo-ns.svc.yahoo.local
  outboundTrafficPolicy: REGISTRY_ONLY
  egress:
  - hosts:
    - istio-system/*
    - finance-k8s/*
    - homepage-k8s/*.media.yahoo.com
```

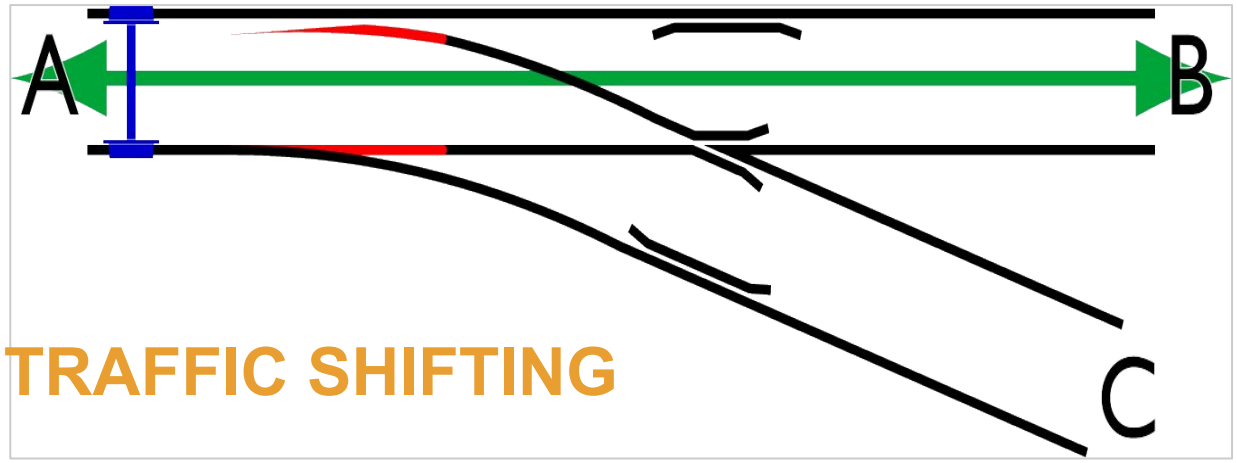
```
apiVersion: authentication.istio.io/v1alpha1
kind: Policy
...
spec:
  targets:
  - name: foo-app.foo-ns.svc.yahoo.local
  peers:
  - mtls:
      mode: STRICT
```

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
...
spec:
  servers:
  - port:
      number: 443
      name: foo-app.foo-ns.svc.yahoo.loca
      protocol: HTTPS
    hosts:
    - foo-app.media.yahoo.com
    - foo-app.foo-ns.svc.yahoo.local
    tls:
      mode: SIMPLE
      serverCertificate: /etc/istio/ingress-certs/tls.crt
      privateKey: /etc/istio/ingress-certs/tls.key
```

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
...
spec:
  host: foo-app.foo-ns.svc.yahoo.local
  trafficPolicy:
    tls:
      mode: MUTUAL
      caCertificates: /etc/certs/root-cer
      clientCertificate: /etc/certs/cert-
      privateKey: /etc/certs/key.pem
```

```
appVersion: networking.istio.io/v1alpha3
kind: VirtualService
...
spec:
  hosts:
  -  foo-app.media.yahoo.com
  -  foo-app.foo-ns.svc.yahoo.local
  http:
  - route:
    - destination:
        port:
          number: 4080
        host: foo-app.foo-ns.svc.yahoo.local
```

More Cool Stuff



Open-Source

Athenz Integrations (https://athenz.io)

- Kubernetes API auth webhook - https://github.com/yahoo/k8s-athenz-webhook
- Identity provider - https://github.com/yahoo/k8s-athenz-identity
- Athenz to Istio RBAC controller - https://github.com/yahoo/k8s-athenz-istio-auth
- Athenz domain cluster cache - https://github.com/yahoo/k8s-athenz-syncer

Policy and Security

- Security benchmarking - https://github.com/yahoo/k8s-sec-check
- Namespace guard admission control - https://github.com/yahoo/k8s-namespace-guard
- Ingress conflict admission control - https://github.com/yahoo/k8s-ingress-claim

Conclusion

- Modernized and standardized overall infrastructure
- Simplified onboarding with custom YAML & templating engine
- Introduced application portability across different platforms
- Reduced development cycles and increased release velocity
- Increased resource utilization
- Enhanced application security
- Provided a uniform interface for doing authn/z

and YES

We ❤ K8s

Thank You

KubeCon CloudNativeCon North America 2019

Contact:

Suresh Visvanathan

Sr. Director, Core Platforms

sureshv@verizonmedia.com

https://www.linkedin.com/in/sureshvisvanathan

@sureshvisvanath

Visit us @ our booth E8