RESILIENCE
REALIZED

KubeCon | CloudNativeCon
North America 2021

KubeCon | CloudNativeCon
North America 2021

RESILIENCE REALIZED

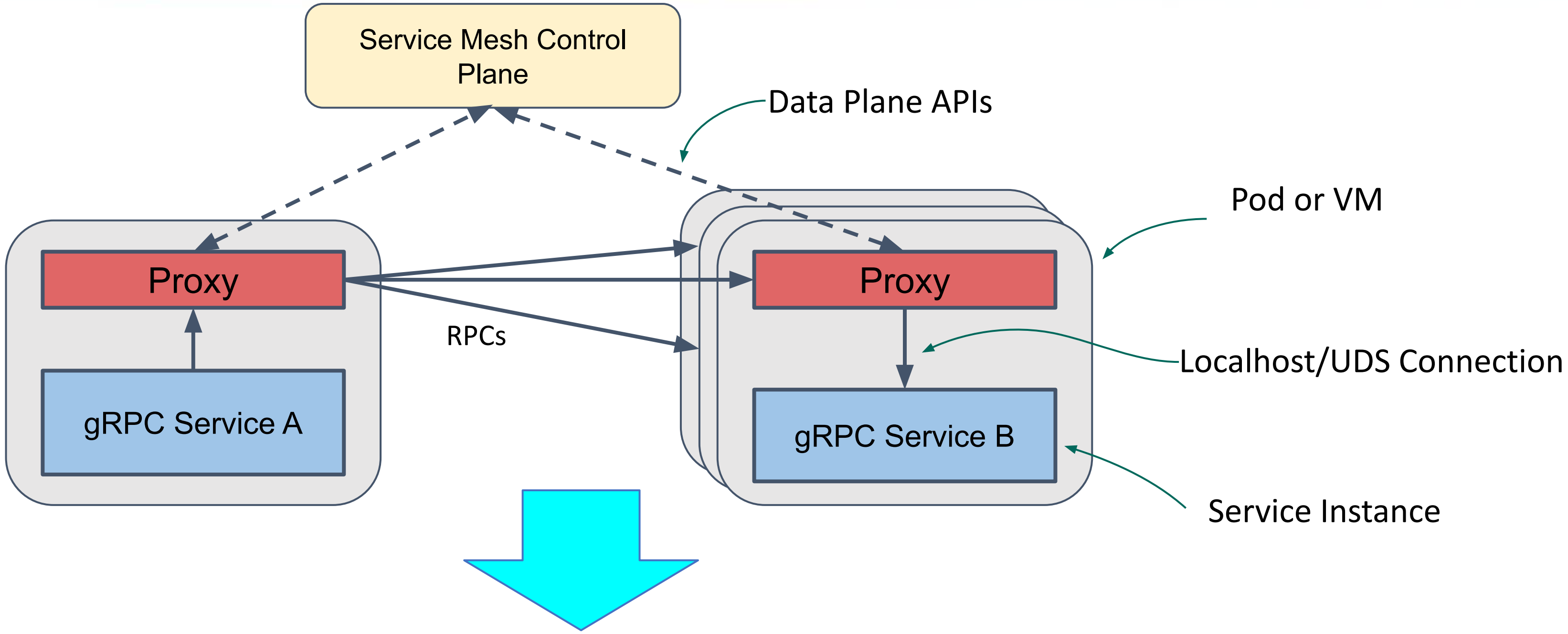
gRPC Proxyless Service Mesh with Security

Sanjay Pujare, Staff SW Engineer, Google Cloud

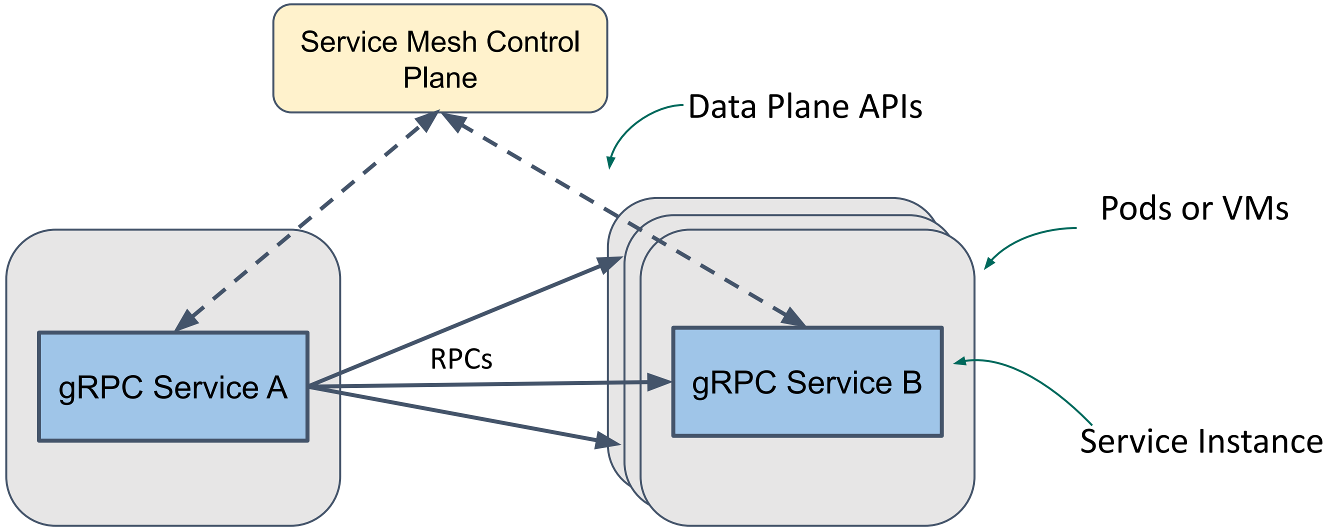
Agenda

- gRPC Proxyless Service Mesh - Intro & Recap
- gRPC PSM Security
- Changes to gRPC for PSM Security
- Importance of Security in a Service Mesh
- xDS Credentials in gRPC Programming API
- Sample Deployment in Google Cloud
- Roadmap & Resources
- Questions

Service Mesh With Proxyless



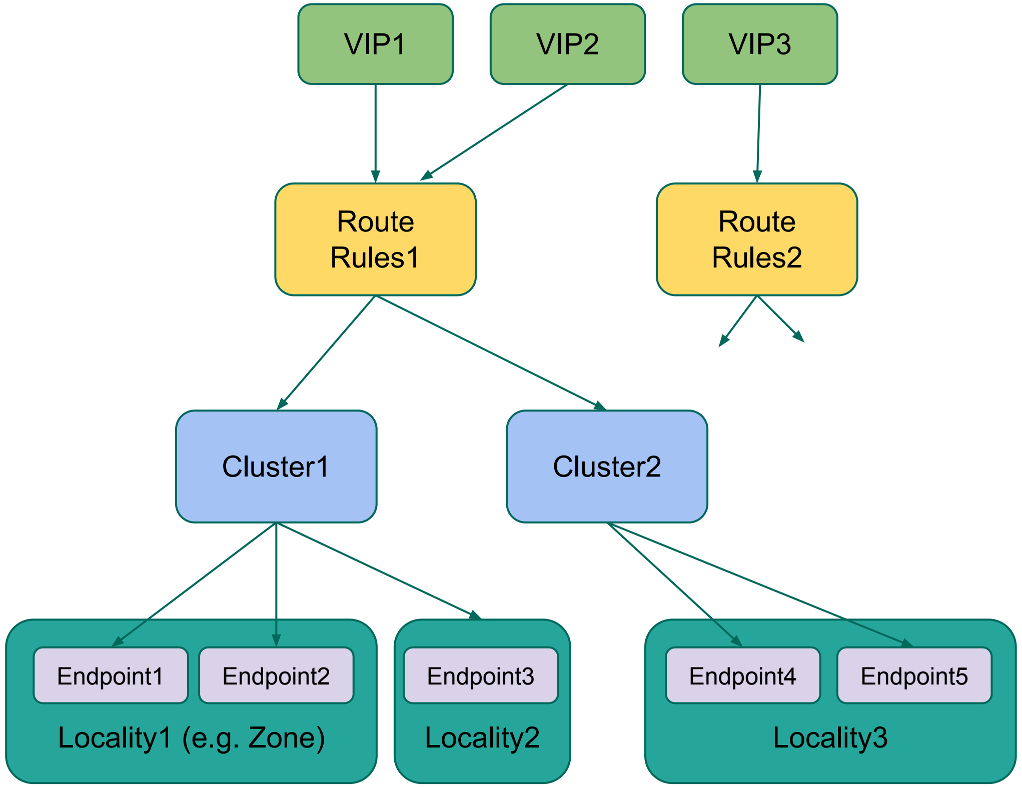
- Proxy used for service mesh policies
- gRPC sends requests to the virtual IP of the service
- Proxy intercepts requests, applies service mesh policies and sends out
- Server proxy receives request, applies policies and forwards to local service



- gRPC client applies service mesh policies from control plane to outbound traffic
- gRPC server similarly applies service mesh policies to incoming traffic
- Services talk to each other directly - without proxies!

Service Mesh With xDS

- xDS Data Plane APIs Developed for popular Envoy proxy
- Open, Extensible & Strong Community Support
- Right choice for gRPC’s Service Mesh implementation!



Listener Discovery Service
Service VIP(IP:Port) configuration

Route Discovery Service
Route matching rules and actions configuration

Cluster Discovery Service
Cluster (Backend Service) configuration

Endpoint Discovery Service
Prioritized and weighted list of localities and endpoints

(x)Discovery Service:

Listener

Route

Cluster

Endpoint etc

Proxyless Service Mesh Journey

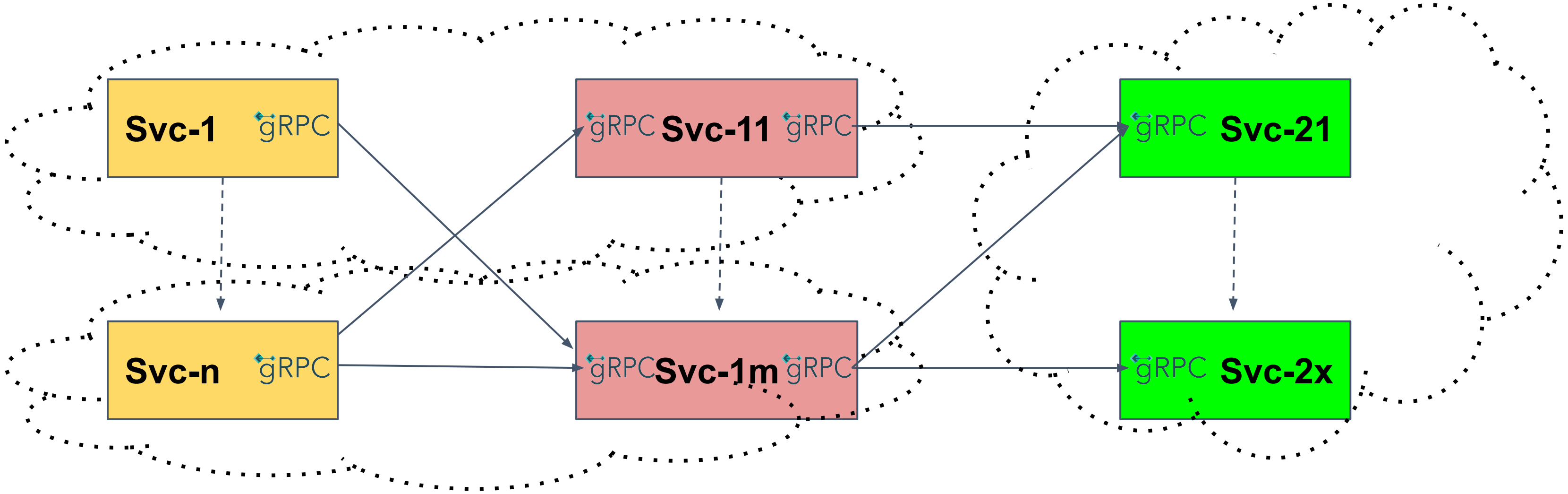
- First release in v1.30.0 in June 2020
  - Service Discovery, Load balancing
- Added various Advanced Traffic Management features
  - v1.31.x in Aug 2020
    - Traffic splitting, Path & header based routing
  - v1.37.x in May 2021
    - Circuit breaking, Fault injection
  - v1.40.x in Aug 2021
    - Retry, Session affinity

gRPC PSM - Current Status

- Previous KubeCon presentation: Service Mesh With GRPC And xDS by Megan Yahya in May’21
  - Covers features upto gRPC v1.37.x
  - Video recording
- Available with Google’s Traffic Director
  - Blog on Circuit Breaking & Fault Injection
  - Blog on Retry & Session Affinity
  - Traffic Director User Guide

Service Mesh Security Importance

- New paradigm of splitting and converting a monolithic application into a mesh of microservices
- In-process calls are now gRPC calls between microservices over the network
- Scaling microservices involves new VMs/clusters/ networks and RPCs crossing these boundaries



Mesh Security Importance…contd…

- Control Plane ties things together: routing, load balancing, service identity authentication and authorization
- Certificate and key updates or rotation: do not burden developers with the toil of cert & key management
- RBAC or Authorization (access control) depends on service identities provided by certificates … logical next step in service mesh security

Service-to-Service Security Today

Using mTLS for Service to Service is a “huge pain” today

- Client & Server Certificates Management
  - create CSR and get them signed or use self-signed
  - exchange root-cert for peer trust store
  - track cert expiry and renew before expiry!
- Code Changes
  - code to load certs and use in the connection
  - security check on top of standard trust verification
- Deployment & Configuration Management
  - Deploy certs on all nodes
  - Periodic replacement on expiry

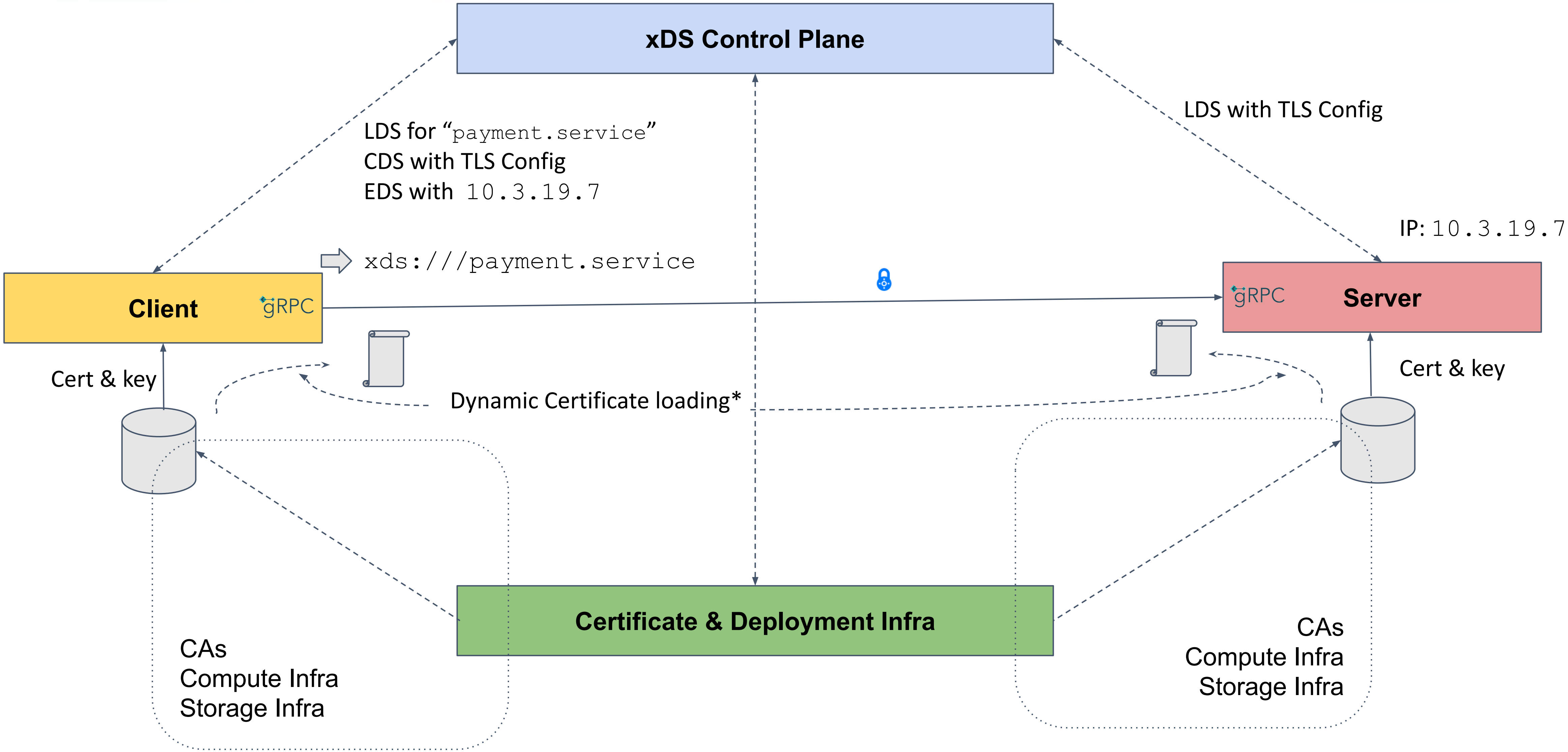
Proxyless Service Mesh Security

- Proxyless gRPC has advanced traffic management features - how about securing the traffic?
- gRPC PSM Security adds service-to-service security
- Transport security (mTLS) for xDS-managed gRPC connections
- mTLS gives you encryption + authentication + server authorization
- How does it work?

PSM Security - How does it work?

- Security Infrastructure provides certificates and keys
- xDS control plane configures a transport_socket with mTLS for a client or server
- Control plane uses the mTLS transport_socket config in CDS (client side) or LDS (server side)
- gRPC takes the provided certs and transport_socket configuration to create mTLS connections
- You have security in the mesh!

PSM Security - Drill Down

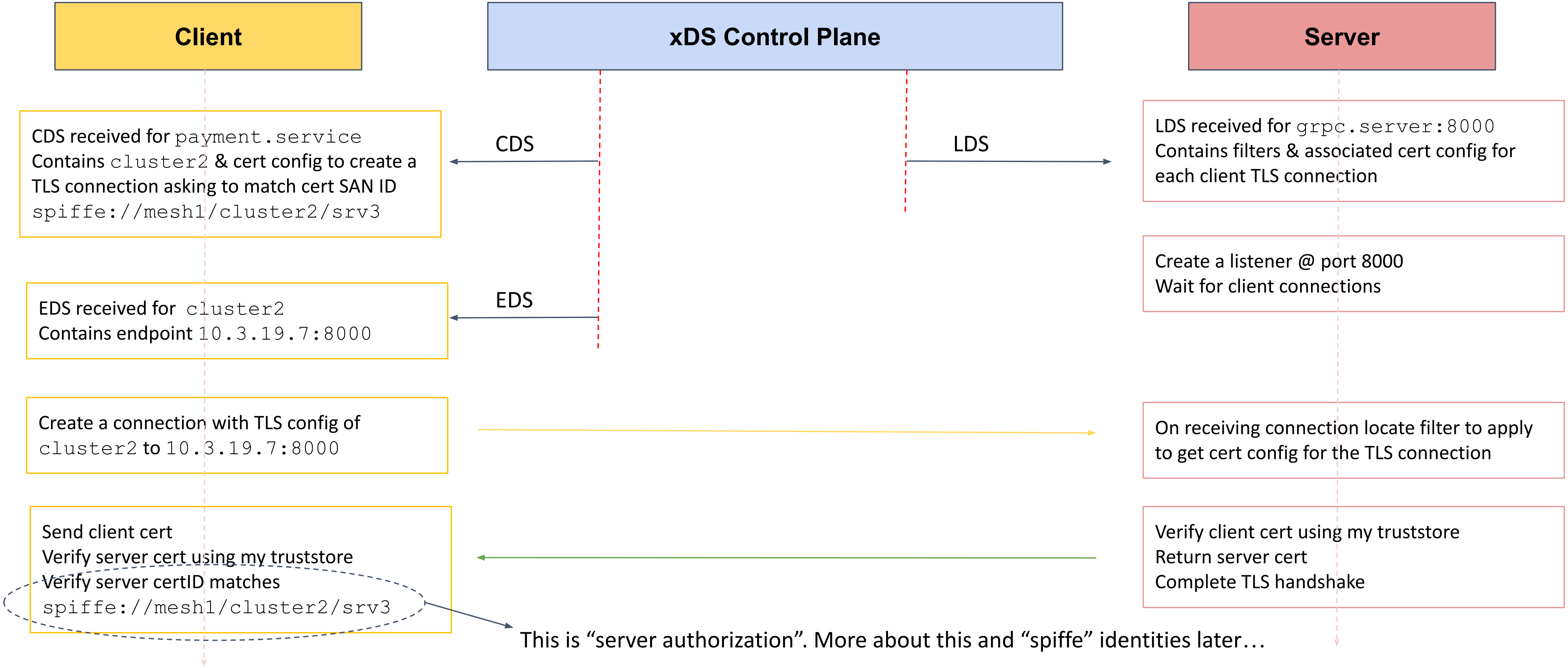


* Certificates can be dynamically updated and are reflected on both client and server side

xDS and mTLS in Client & Server

KubeCon CloudNativeCon North America 2021

IP: `10.3.19.7`
CertID: `spiffe://mesh1/cluster2/srv3`



CDS received for `payment.service`
Contains `cluster2` & cert config to create a TLS connection asking to match cert SAN ID `spiffe://mesh1/cluster2/srv3`

EDS received for `cluster2`
Contains endpoint `10.3.19.7:8000`

Create a connection with TLS config of `cluster2` to `10.3.19.7:8000`

Send client cert
Verify server cert using my truststore
Verify server certID matches `spiffe://mesh1/cluster2/srv3`

LDS received for `grpc.server:8000`
Contains filters & associated cert config for each client TLS connection

Create a listener @ port 8000
Wait for client connections

On receiving connection locate filter to apply to get cert config for the TLS connection

Verify client cert using my truststore
Return server cert
Complete TLS handshake

This is “server authorization”. More about this and “spiffe” identities later…

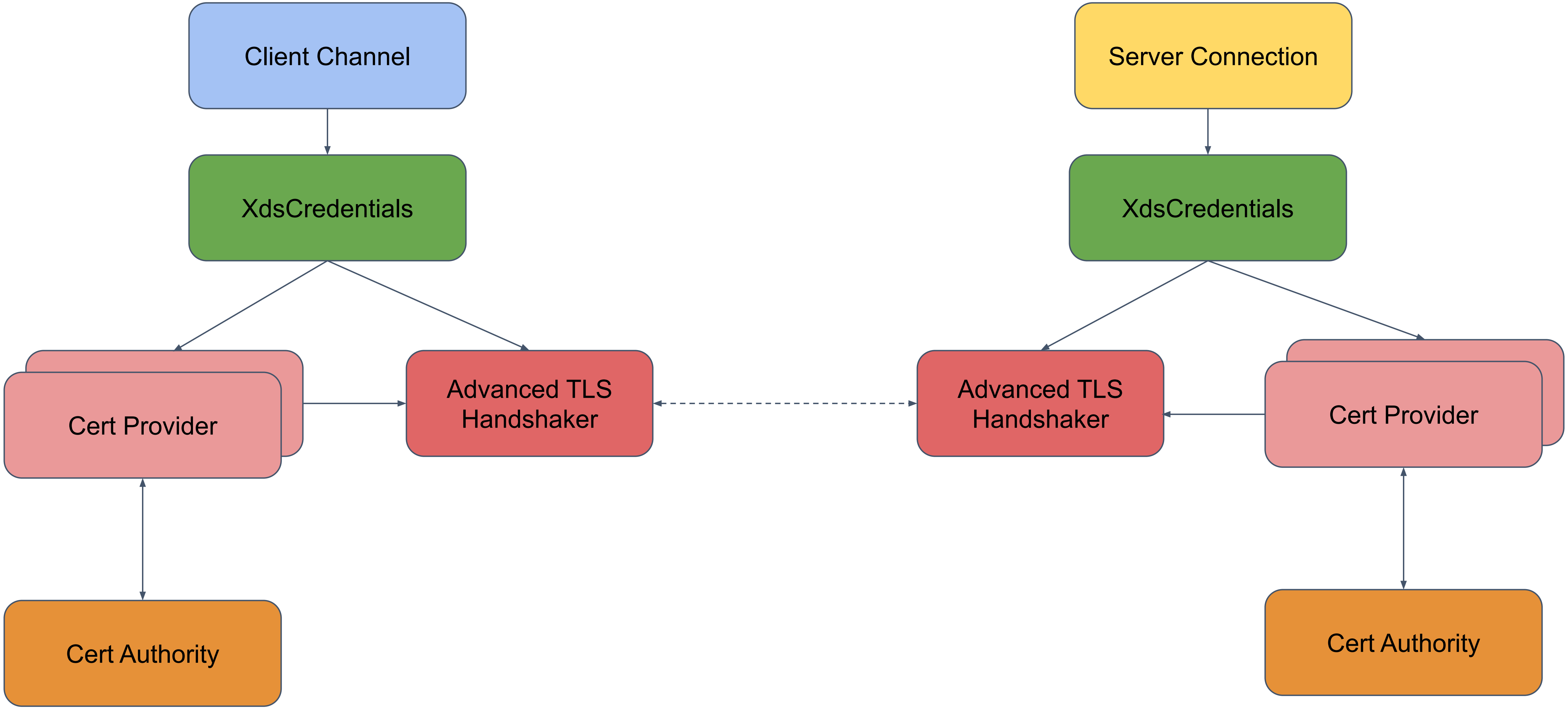
Design and Implementation Details

- Design spec in gRFC A29: xDS-Based Security for gRPC
- gRFC covers:
  - Programming API: what API to call to use the feature?
  - gRPC's implementation of xDS security flow
  - Certificate Provider Plugin framework to provide needed certificates and keys … more later
- Implemented in gRPC Java, Go, C++ and Python
- "Public preview" in May'21 for C++, Python and Java with release 1.37 and in Go with release 1.38

Certificate Provider plugins

- Certificate Provider plugin framework in gRPC
  - extensible framework allows various/custom mechanisms to get certificates
  - plugins loaded and configured locally using bootstrap info
  - xDS only references an “instance” which gRPC interprets using bootstrap
- `file_watcher` plugin in gRPC C++, Go, Java…
  - certs and key watched in the file system
  - updates reflected in channels and servers

Certificate Provider plugins



What’s in the gRPC Library?

- New API to enable programmatic usage
- xDS implementation of transport_socket config
- Certificate Provider Plugin framework in gRPC and its addition to xDS protocols
- Bootstrap file enhancement for Certificate Provider configuration
- `file_watcher` Certificate Provider implementation
- Support dynamic certificate & key updates

Service Identities - SPIFFE Identities

Not a gRPC thing but a service mesh thing:

- Microservices are both client & server: unified identity encoded in certificate used on both sides
- Client performs “server authorization” to ensure the server identity is the one it was expecting to connect to
  - Replaces the “hostname” check in https
- Server can use an “authorization policy” to restrict incoming RPCs based on identities … coming soon!
- SPIFFE service identity `spiffe://<domain>/<identity>`

How Does One Use This in gRPC?

Use “Xds”-Channel and Server credentials.

Java example from A29-xds-tls-security.md#java

XdsChannelCredentials on the channel (client side):

```
ChannelCredentials credentials
           = XdsChannelCredentials.create(InsecureChannelCredentials.create());
ManagedChannel channel = Grpc.newChannelBuilder(target, credentials).build();
```

XdsServerCredentials on the server side:

```
ServerCredentials credentials
      = XdsServerCredentials.create(InsecureServerCredentials.create());
Server server = XdsServerBuilder.forPort(port, credentials)
      .addService(new HostnameGreeter(hostname)).build().start();
```

More about xDS Credentials

- Caller "opts in" to allow use of xDS provided security for a gRPC channel or server by using “`Xds`” credentials
- Caller can use a different credentials in which case xDS provided security is ignored. e.g.

```
    ChannelCredentials credentials = TlsChannelCredentials.create();
    ManagedChannel channel = Grpc.newChannelBuilder(target, credentials).build();
```

even if target is “`xds:///payment.service`” use my TLSCreds

- Fallback credentials: use xDS provided security if present else use my fallback credentials

```
    XdsChannelCredentials.create(TlsChannelCredentials.create())
```

Deploying Your Code

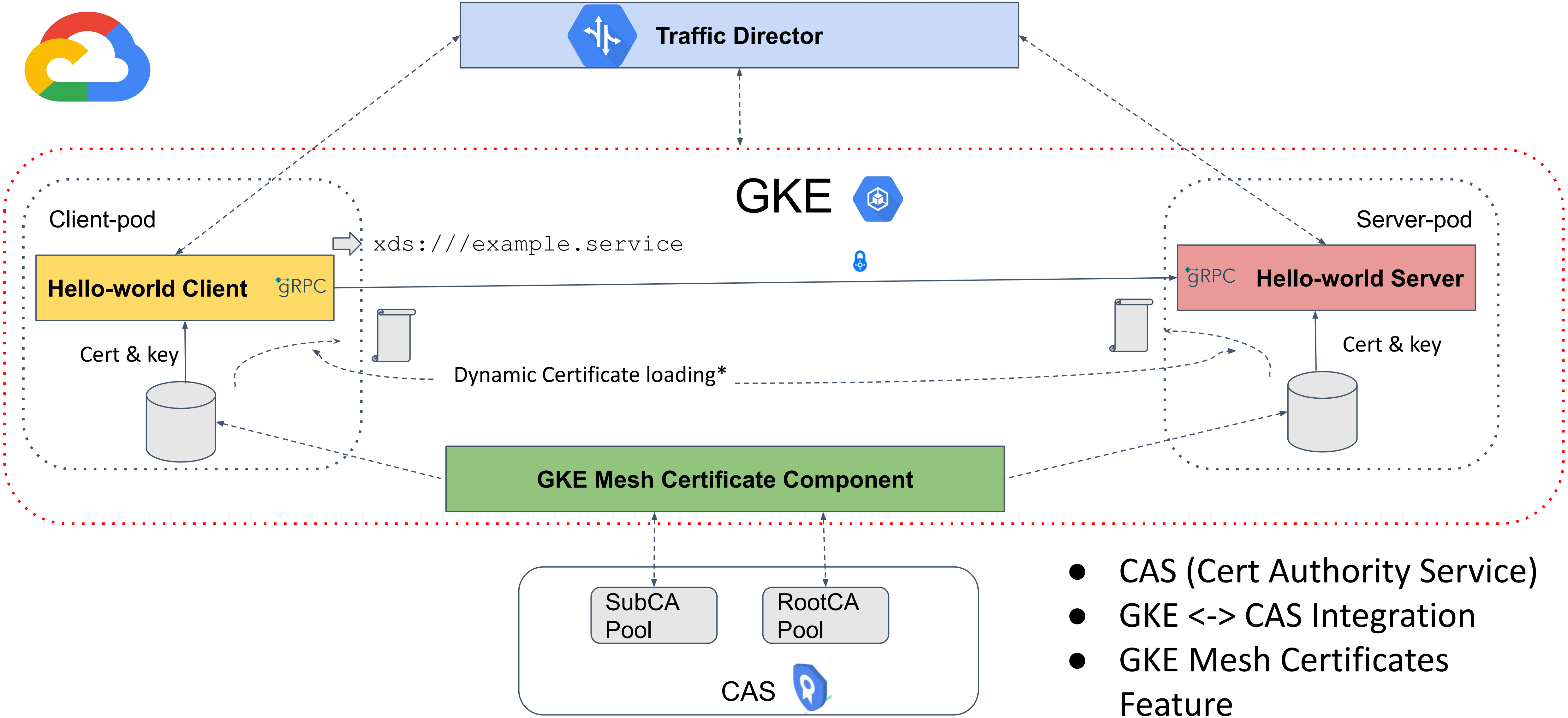
Where can you use your xDS-credentials code?

Use TD Service Security with Proxyless gRPC

What's involved?

- Traffic Director - the xDS control plane
- Certification Authority Service aka CAS - your CA infra
- GKE - deploy your containerized workloads
  - mesh-certificates feature
  - GKE uses the Certification Authority Service aka CAS to get mesh certificates for the pods
- User Guide to try the flow

gRPC PSM Security in Google Cloud



- CAS (Cert Authority Service)
- GKE <-> CAS Integration
- GKE Mesh Certificates Feature
- TD GKE Security Integration

* Certificates are dynamically updated and are reflected on both client and server side

What’s Next?

- xDS Authorization aka xDS RBAC Support : ETA Oct’21
- SPIFFE with federated trust domains i.e. trust bundles
- Configurable Certificate Validator - possibly as part of federation of trust domains
- More Cert Provider plugins…
- Transport_socket extensions e.g. use of handshaker service for handling TLS
- Envoy adopting Cert Provider plugin framework

Resources

- gRFCs
    - A29: xDS-Based Security for gRPC
    - L74: Java Channel and Server Credentials
    - A27: xDS-Based Global Load Balancing
    - A36: xDS-Enabled Servers
    - A41:xDS RBAC Support
- Blog: Security for gRPC Apps with Traffic Director
- Traffic Director service security with proxyless gRPC User Guide

Thank You!

Questions?