ex_control_plane: A Control Plane Framework for EnvoyProxy in Elixir

André Graf

Elixir Meetup Basel @ Helvetia Insurance (23.09.2025)

Who am I?

- Solution Architect at Helvetia Insurance since 2020
- Engineer & operate the Helvetia API platform (from API gateways to GitOps enabled selfservices to API portal)
- On the Beam since 2009

Enter Envoy

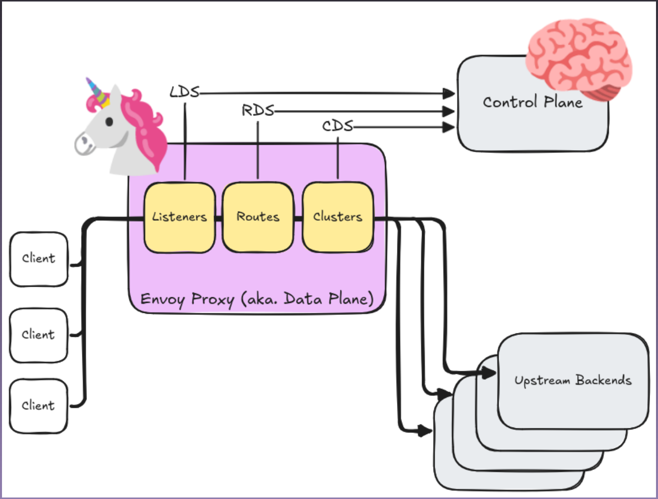
- **High-performance proxy** built for cloud-native systems
- Originally created at Lyft, now a **CNCF graduated project**
- Works as a **data plane**: handles traffic between services
- Designed for **modern architectures**:
    - Service Mesh (e.g. Istio, Consul)
    - API Gateway & edge proxy
    - Sidecar proxy for microservices

Enter Envoy

Key Features

- **L4/L7 proxy**: understands both TCP and HTTP
- **Dynamic configuration** via xDS APIs (not config files)
- **Observability**: detailed metrics, tracing, logging
- **Extensibility**: Lua, WASM, experimental Go & Rust, External Processor

High-level Control Plane



Today’s Control Planes

Most control planes for Envoy are written in Go

- CNCF ecosystem defaults to Go
- Strong ecosystem: gRPC, Kubernetes client libs, tooling
- The `go_control_plane` is the official dataplane API implementation[1]

[1] https://github.com/envoyproxy/go-control-plane

Why Elixir fits Control Planes

“Scaling control planes requires skills and tools for distributed systems.” — Matt Klein, creator of Envoy[2]

[2] https://mattklein123.dev/2020/03/15/2020-03-14-on-the-state-of-envoy-proxy-control-planes/

Why Elixir fits Control Planes

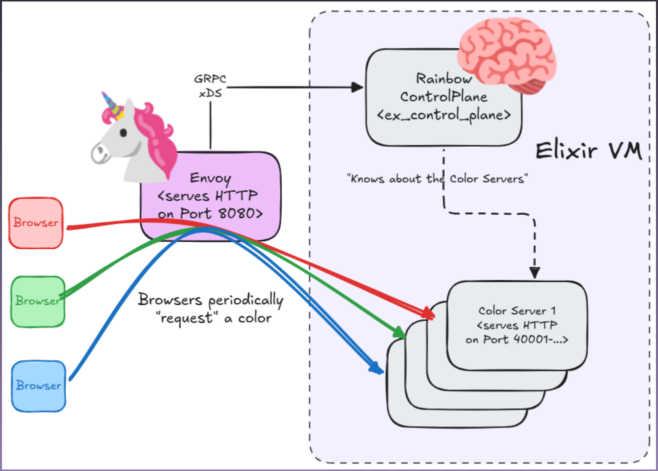
- **Massive concurrency**: handle thousands of Envoy xDS streams with lightweight processes
- **Fault tolerance**: supervisors keep the system resilient under failure
- **Built-in distribution**: message passing and clustering are part of the VM
- **Hot code upgrades**: upgrade the system without a downtime
- **Expressive state**: Envoy configs are just data, easily modeled, transformed, piped in Elixir

Why Elixir fits Control Planes

Relating to What You Know

- Phoenix handles *thousands of live client connections*
- LiveView updates all those clients dynamically

The Demo Playground



GRPC xDS
Rainbow ControlPlane <ex_control_plane>
Elixir VM
"Knows about the Color Servers"
Envoy <serves HTTP on Port 8080>
Browser
Browser
Browser
Browsers periodically "request" a color
Color Server 1 <serves HTTP on Port 40001-...>

ex_control_plane

- Uses Aggregated Discovery Services (ADS)
  - Resource-level state-of-the-world
- Supported resources (via ADS)
  - listeners
  - clusters
  - secrets
  - route configurations
  - scoped route configurations
- ETS based config cache
- Simple AWS S3 based config snapshots (cold start from last good state)
- Implement `ExControlPlane.Adapter` behaviour (3 callbacks)

Thank You

- Slides & demo code: rainbow_plane
- ex_control_plane on GitHub: proxyconf/ex_control_plane
- Questions?