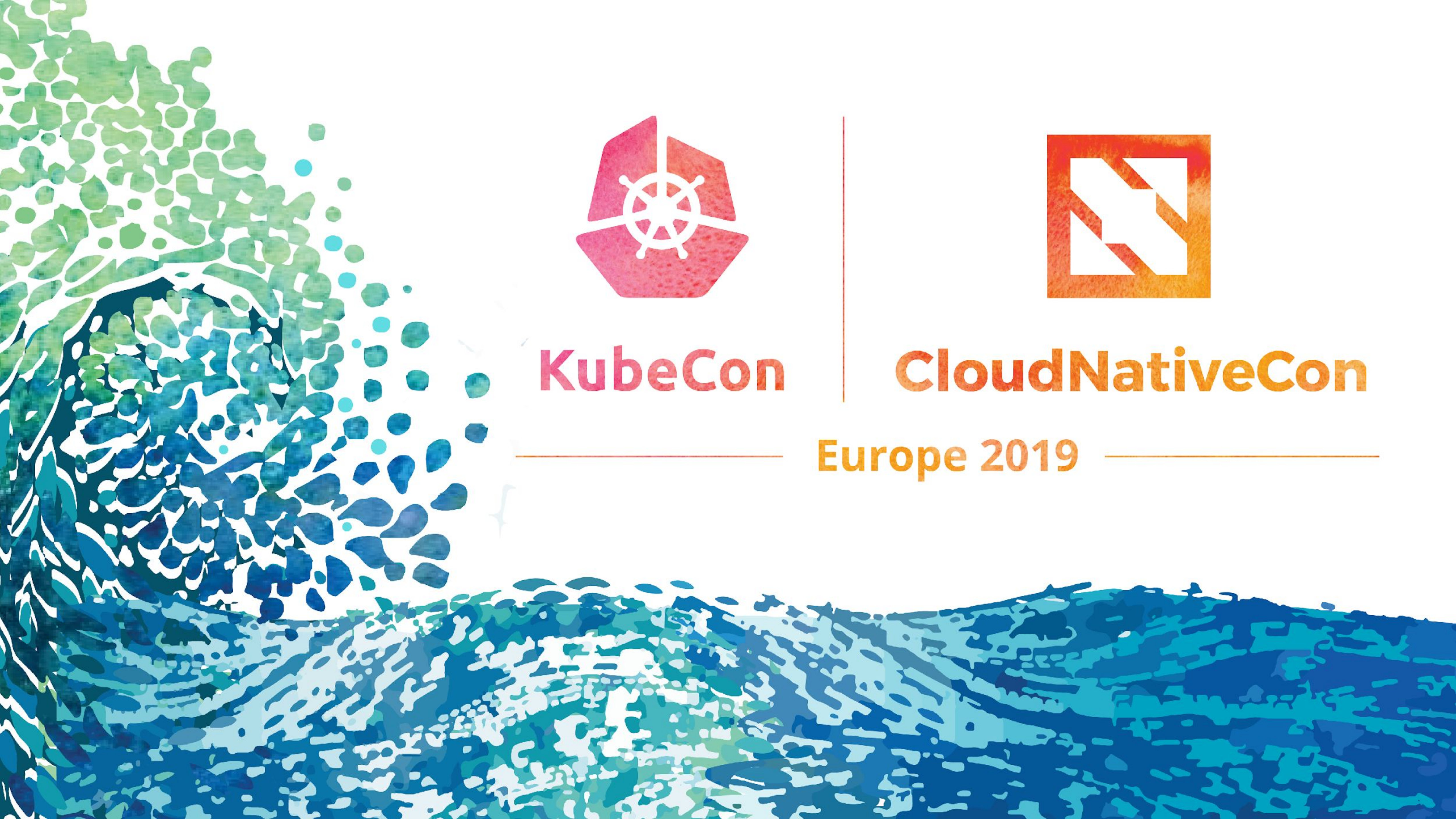


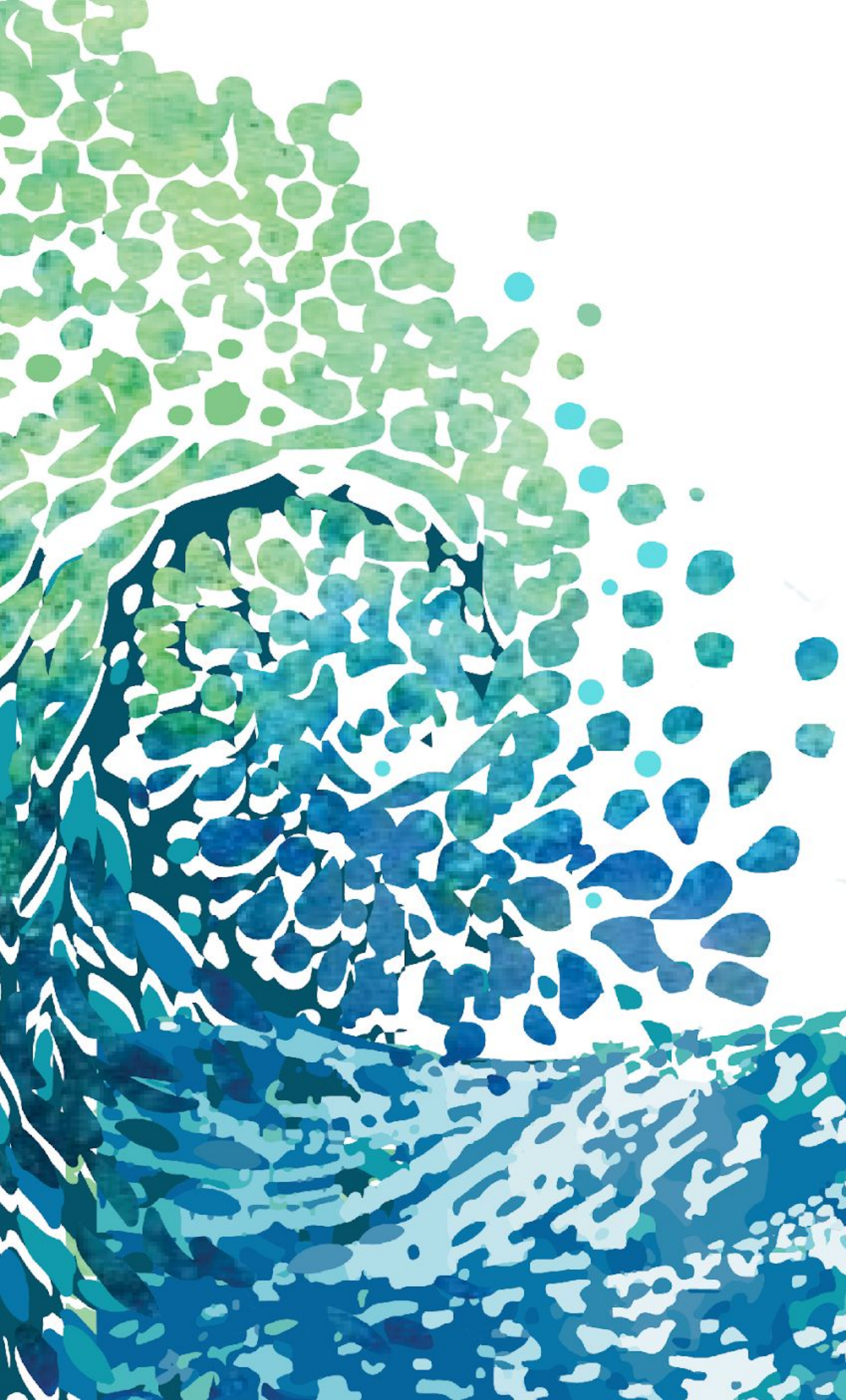
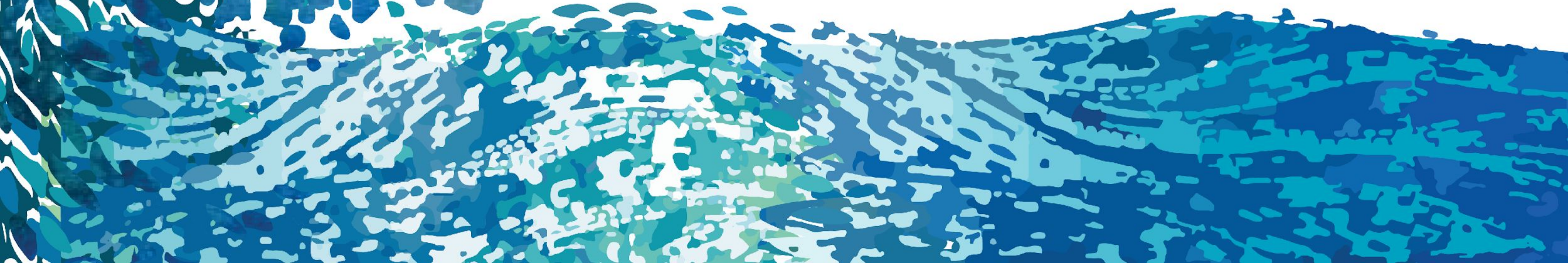
KubeCon | CloudNativeCon

Europe 2019

KubeCon | CloudNativeCon
Europe 2019

Extending Envoy with WebAssembly

Dhi Aurrahman, Tetrate
John Plevyak, Google

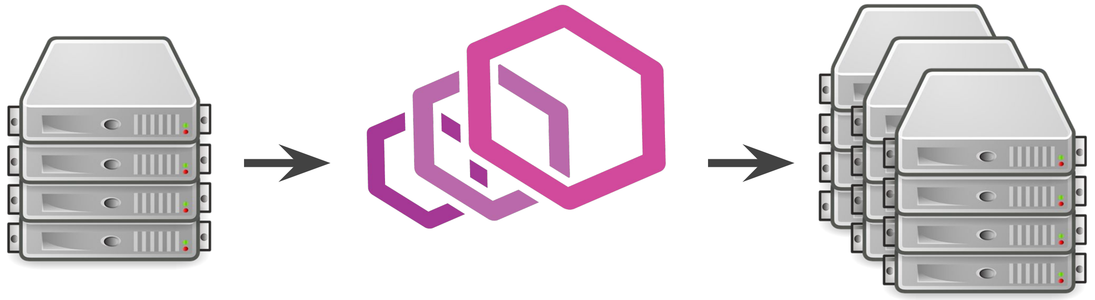
Agenda

- About **Envoy**
- **Envoy** architecture
- **Envoy** extensibility
  Write your own extensions
- Extending **Envoy** with WebAssembly
- Demo
- Q&A

Envoy

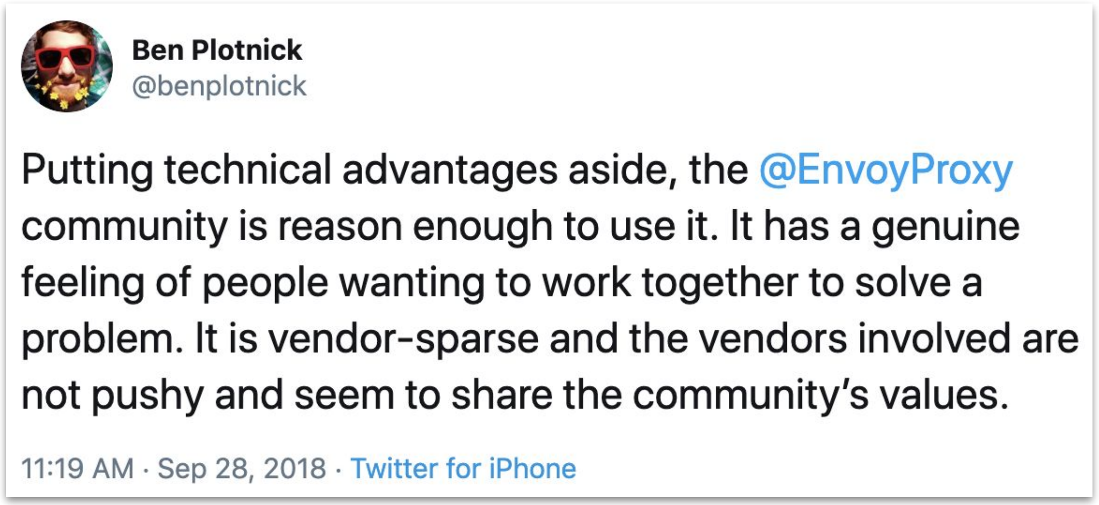
The network should be transparent to applications.

When network and application problems do occur it should be easy to determine the source of the problem.



Envoy

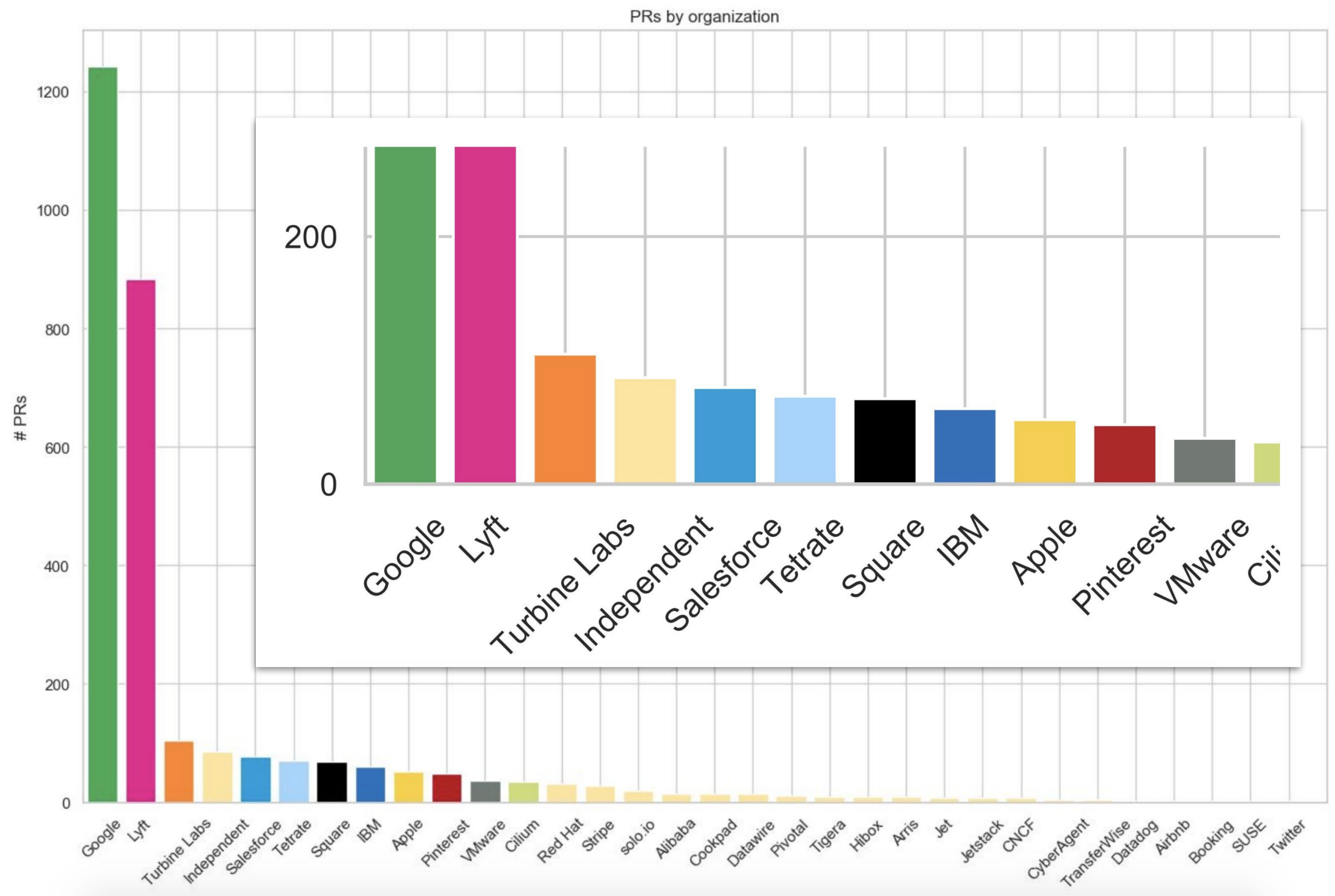
- Quality + velocity
- Extensibility
- Eventually consistent configuration API
- No “open core”/paid premium version. It’s all there.
- **Community, community, community**


Ben Plotnick
@benplotnick

Putting technical advantages aside, the @EnvoyProxy community is reason enough to use it. It has a genuine feeling of people wanting to work together to solve a problem. It is vendor-sparse and the vendors involved are not pushy and seem to share the community’s values.

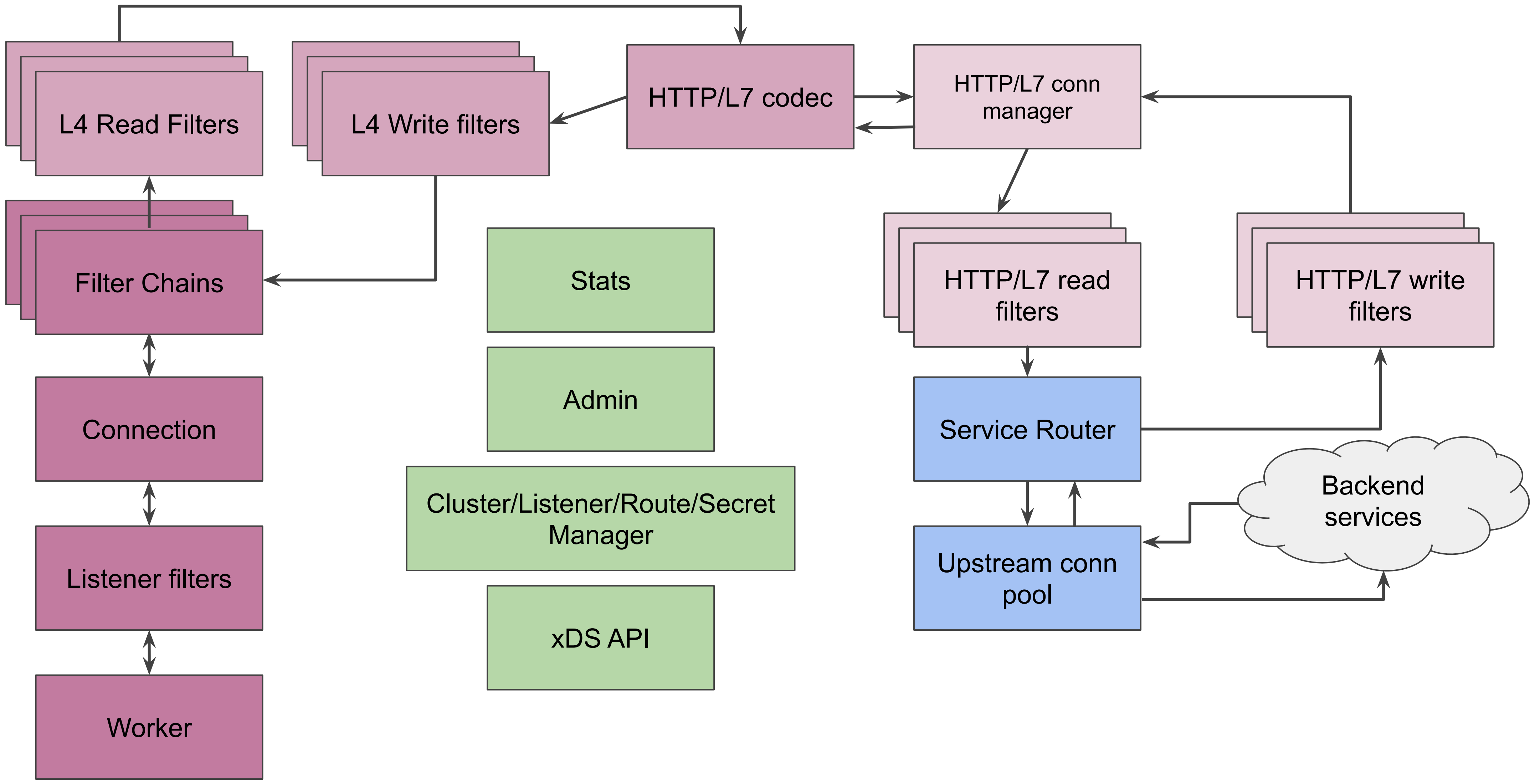
11:19 AM · Sep 28, 2018 · Twitter for iPhone

Who is Envoy?



Who is Envoy?
Harvey Tuch, Google
https://envoyconna18.sched.com/event/HCvf/lightning-talk-who-is-envoy-harvey-tuch-google

Envoy Architecture



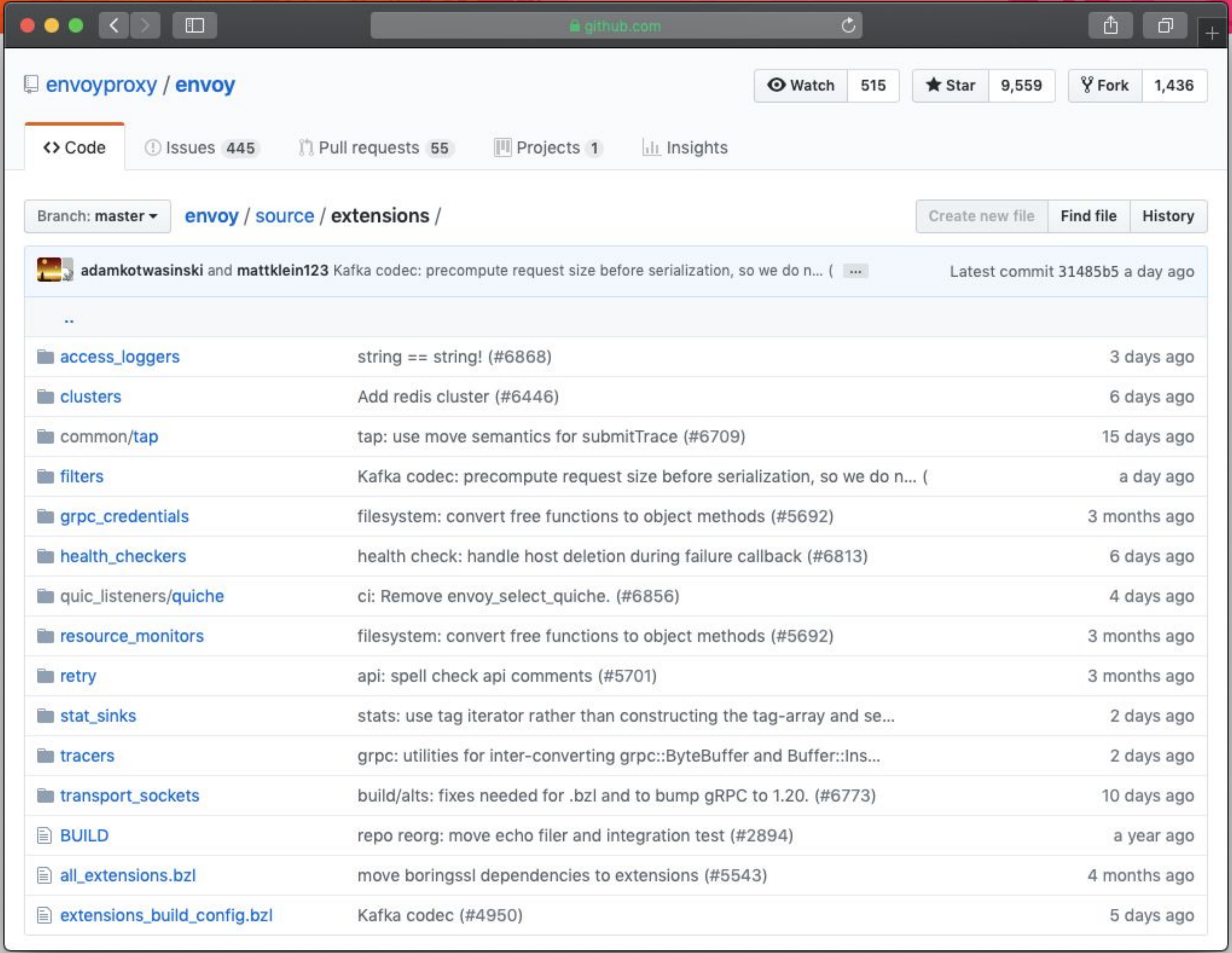
Envoy Extensibility

Envoy is designed to be extensible, with multiple extension points

- L4/L7 filters
- Access loggers
- Tracers
- Health checkers
- Transport sockets
- Retry policy
- Resource monitors
- Stats sink

It's growing...

KubeCon CloudNativeCon Europe 2019



envoy / source / extensions /

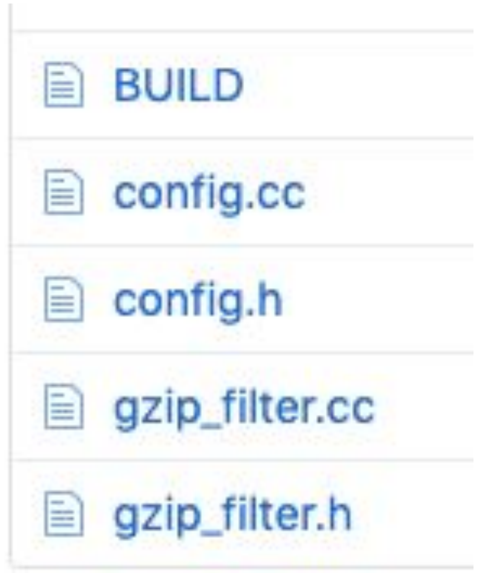
adamkotwasinski and mattklein123 Kafka codec: precompute request size before serialization, so we do n... (... Latest commit 31485b5 a day ago

..

access_loggers	string == string! (#6868)	3 days ago
clusters	Add redis cluster (#6446)	6 days ago
common/tap	tap: use move semantics for submitTrace (#6709)	15 days ago
filters	Kafka codec: precompute request size before serialization, so we do n... (	a day ago
grpc_credentials	filesystem: convert free functions to object methods (#5692)	3 months ago
health_checkers	health check: handle host deletion during failure callback (#6813)	6 days ago
quic_listeners/quiche	ci: Remove envoy_select_quiche. (#6856)	4 days ago
resource_monitors	filesystem: convert free functions to object methods (#5692)	3 months ago
retry	api: spell check api comments (#5701)	3 months ago
stat_sinks	stats: use tag iterator rather than constructing the tag-array and se...	2 days ago
tracers	grpc: utilities for inter-converting grpc::ByteBuffer and Buffer::Ins...	2 days ago
transport_sockets	build/alts: fixes needed for .bzl and to bump gRPC to 1.20. (#6773)	10 days ago
BUILD	repo reorg: move echo filer and integration test (#2894)	a year ago
all_extensions.bzl	move boringssl dependencies to extensions (#5543)	4 months ago
extensions_build_config.bzl	Kafka codec (#4950)	5 days ago

The structure, ingredients

- “Business **logic**”
- **Configuration** for that “filter”


BUILD
config.cc
config.h
gzip_filter.cc
gzip_filter.h

```
message AuthServiceFilter {
 string cluster = 1;
 enum AuthType {
   OAUTH = 0;
   JWT = 1;
 }
 AuthType auth_type = 2;
}
```

Write your own

To write your own extension, currently we can do it using:

- C++
- Lua (limited for HTTP traffic only for now)
    - https://www.envoyproxy.io/docs/envoy/latest/configuration/http_filters/lua_filter

Example: GzipFilter, C++

```
class GzipFilter : public Http::StreamFilter {
public:
 GzipFilter(const GzipFilterConfigSharedPtr& config);


...
// Where the config object is “derived” from the generated protobuf object.
GzipFilterConfig(const envoy::config::filter::http::gzip::v2::Gzip& gzip,
                  const std::string& stats_prefix,
                  Stats::Scope& scope, Runtime::Loader& runtime);
```

Http::StreamFilter

```
// Http::StreamDecoderFilter
 Http::FilterHeadersStatus decodeHeaders(Http::HeaderMap& headers, bool end_stream)
 Http::FilterDataStatus decodeData(Buffer::Instance&, bool)
 Http::FilterTrailersStatus decodeTrailers(Http::HeaderMap&)
 void setDecoderFilterCallbacks(Http::StreamDecoderFilterCallbacks& callbacks)

 // Http::StreamEncoderFilter
 Http::FilterHeadersStatus encode100ContinueHeaders(Http::HeaderMap&)
 Http::FilterHeadersStatus encodeHeaders(Http::HeaderMap& headers, bool end_stream)
 Http::FilterDataStatus encodeData(Buffer::Instance& buffer, bool end_stream)
 Http::FilterTrailersStatus encodeTrailers(Http::HeaderMap&)
 Http::FilterMetadataStatus encodeMetadata(Http::MetadataMap&)
 void setEncoderFilterCallbacks(Http::StreamEncoderFilterCallbacks& callbacks)
```

Core “logic” Implementation

```
Http::FilterHeadersStatus GzipFilter::decodeHeaders(Http::HeaderMap& headers, bool) {
 if (config_->runtime().snapshot().featureEnabled("gzip.filter_enabled", 100) &&
     isAcceptEncodingAllowed(headers)) {
   skip_compression_ = false;
   if (config_->removeAcceptEncodingHeader()) {
     headers.removeAcceptEncoding();
   }
 } else {
   config_->stats().not_compressed_.inc();
 }

 return Http::FilterHeadersStatus::Continue;
}
```

Core “logic” Implementation

```
Http::FilterDataStatus GzipFilter::encodeData(Buffer::Instance& data, bool end_stream) {
 if (!skip_compression_) {
   config_->stats().total_uncompressed_bytes_.add(data.length());
   compressor_.compress(data, end_stream ? Compressor::State::Finish :
Compressor::State::Flush);
   config_->stats().total_compressed_bytes_.add(data.length());
 }
 return Http::FilterDataStatus::Continue;
}
```

A “better” way to extend

Seems like we can do better than this.

WASM?

No, you can't always get what you want. But if you try sometime you find. You get what you need.
~ Mick Jagger

WA

WEBASSEMBLY

Introduction

Why WebAssembly is right for Extensions to Envoy

WebAssembly in Envoy in Summary

- Architecture
- Extension APIs
- VM APIs

Status and Future

Demo

Q&A

Why WebAssembly?

Portable, Sandboxed Machine Code
Web Standard under active development
Good Implementations Available
Broad Language Support
Fast
Safe*
Portable, Dynamically Loadable and Linkable

Use Cases

Consider the Istio (istio.io) Service Mesh

Envoy is in the Data Plane enforcing Istio Policy

WebAssembly Extensions allow:

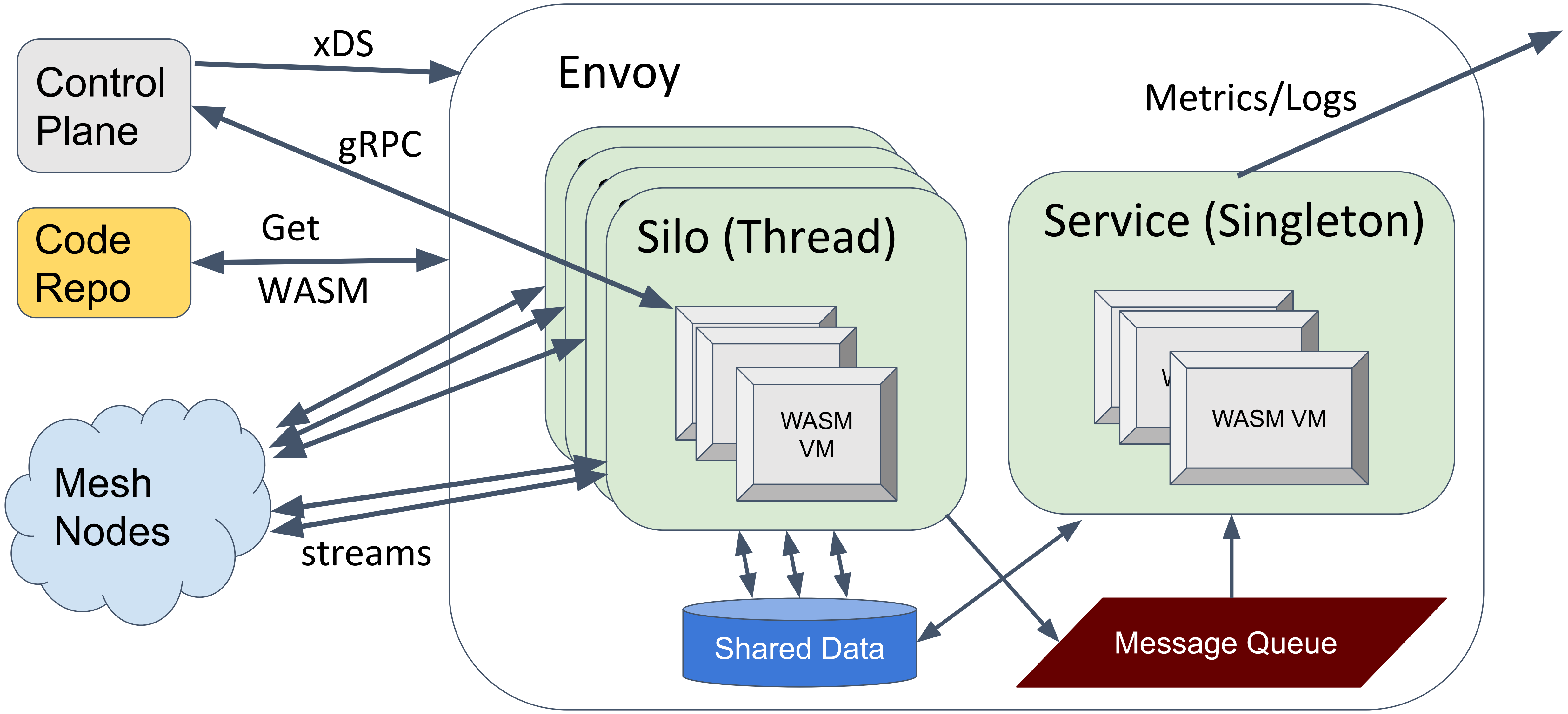
- Avoid changes to Envoy proper
- Avoiding network hops for policy checks
- Avoid restart through Dynamic (Re)loading
- Isolation
- Live A/B testing

WebAssembly in Envoy

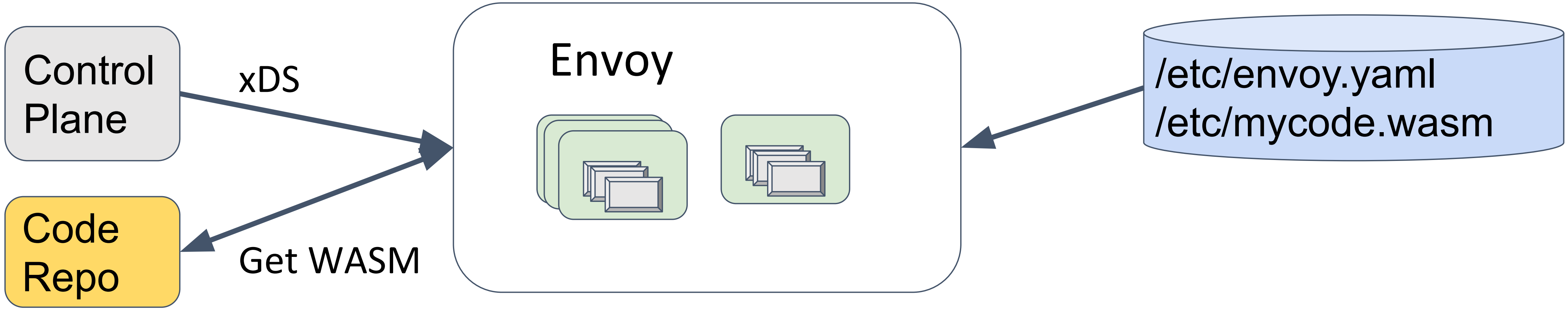
Architecture
General APIs
Extension APIs
VM API
Implementation

https://github.com/envoyproxy/envoy-wasm

Architecture - Overview

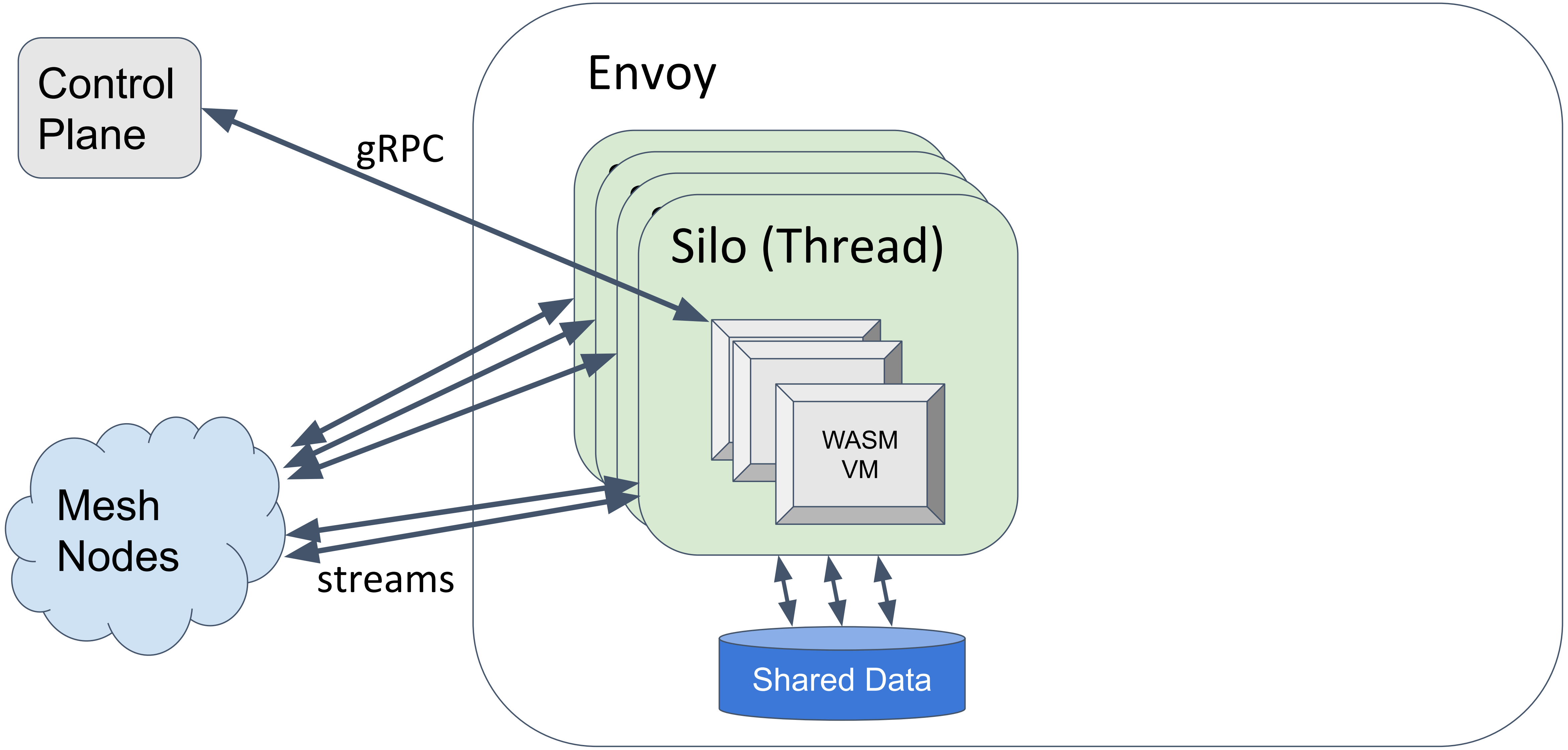


Architecture - Configuration

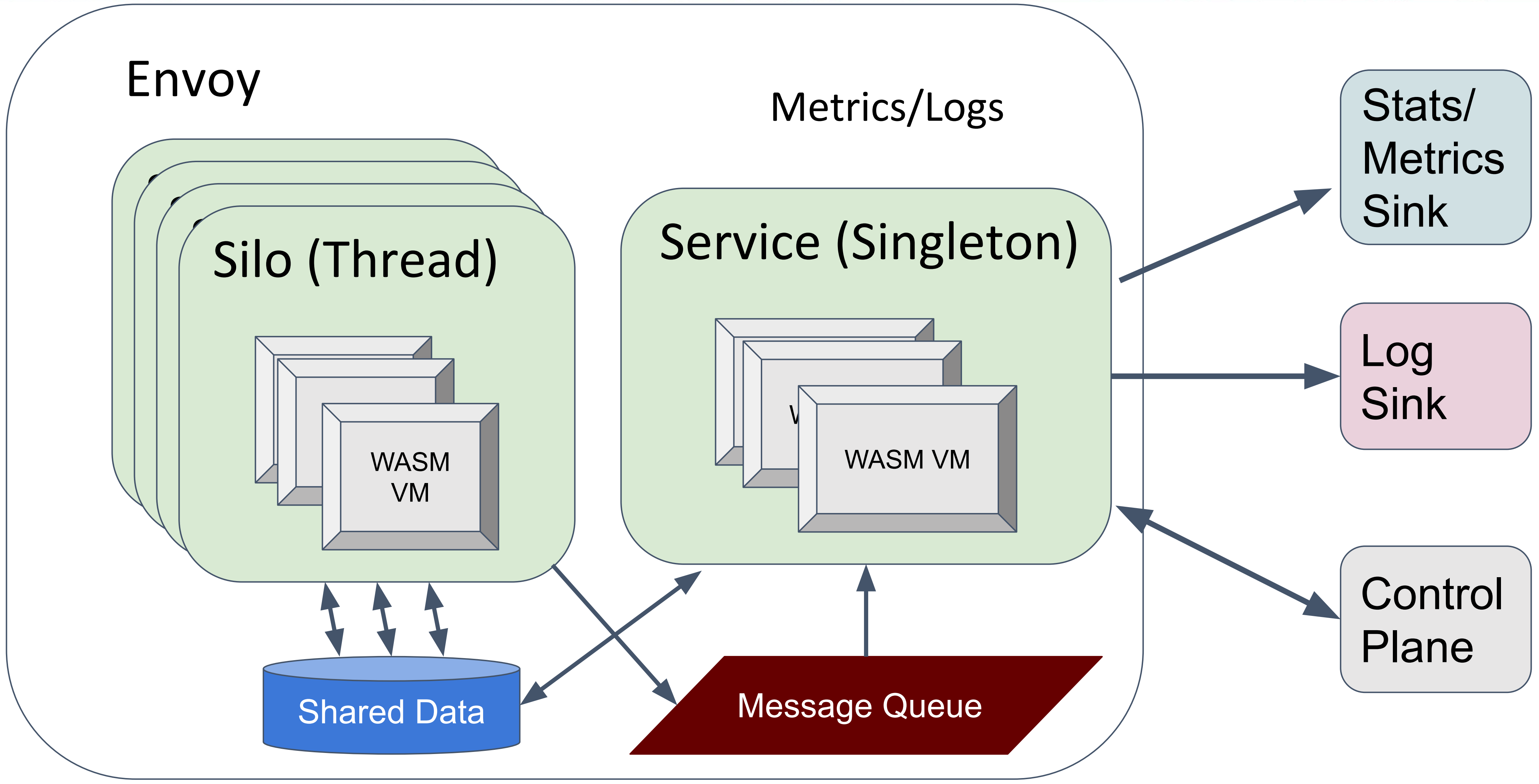


Startup configuration can include extension config and code
Dynamic updates can include config and code URI + signature

Architecture - Extensions



Architecture - Services



General APIs

Timer

Metadata

HTTP Get

gRPC (Envoy and Google style)

Logging

Stats/Metrics

Shared Data

Message Queue

Metrics API Example

Low Level (C) API - calls out of the VM

```
extern "C" uint32_t proxy_defineMetric(
  MetricType type, const char* name_ptr, size_t name_size);
extern "C" void proxy_incrementMetric(
  uint32_t metric_id, int64_t offset);
```

High Level (C++) API - implemented by wrappers in the VM

```
auto c = Counter<std::string, int, bool>::New(
  "test_counter", "string_tag", "int_tag", "bool_tag");
c->increment(1, "test_tag", 7, true);
auto cc = c->resolve("test_tag", 7, true);
cc->increment(1);
```

Extension APIs

HTTP Filters

Log Filters

Services (singletons)

TCP Filters*

Log Sinks*

?

*WIP

HTTP Filter Example

```
class ExampleContext : public Context {
public:
  explicit ExampleContext(uint32_t id) : Context(id) {}
  FilterHeadersStatus onRequestHeaders() override;
  void onLog() override;
};

FilterHeadersStatus ExampleContext::onRequestHeaders() {
  logDebug(std::string("onRequestHeaders ") + std::to_string(id()));
  auto path = getRequestHeader(":path");
  logInfo(std::string("header path ") + std::string(path->view()));
  addRequestHeader("newheader", "newheadervalue");
  replaceRequestHeader("server", "envoy-wasm");
  return FilterHeadersStatus::Continue;
}

void ExampleContext::onLog() {
  auto path = getRequestHeader(":path");
  logWarn("onLog " + std::to_string(id()) + " " + std::string(path->view()));
}
```

VM APIs

Support for Multiple VMs, currently:

- WAVM (https://github.com/WAVM/WAVM)
- V8
- Null Sandbox (use the API, compile directly into Envoy)

Adding a new VM requires:

- registerCallback (for calls from the VM)
- getFunction (for calls to the VM)
- load(), link(), start()
- setMemory/getMemory - copy into and out of the VM

Status

Http Filter, Log Filter and Service Extension APIs done
WAVM, V8, Null Sandbox VM support done
Metrics, logging, Metadata, Shared State done
HTTP Call, gRPC Call done
Event driven stream handling done
Copy-in copy-out data passing done

Future Work

Additional Extension APIs WIP

Additional VMs (cranelift, ...) todo

Coroutines/Threads todo

Zero-copy memory data passing (SharedArrayBuffer) todo

Dynamic Loading and Linking todo

Upstream to Envoy master WIP

Conclusion

WebAssembly is a good way to extend Envoy

- Efficient
- Safe and Secure
- Dynamically upgradeable

Follow the progress at:

- https://github.com/envoyproxy/envoy/issues/4272
- https://github.com/envoyproxy/envoy-wasm

Contribute

- APIs, design docs, PRs, examples and use cases

Demo

Questions

Thank you!

@diorahman
@jplevyak

Backup Slides

Nothing to see here, move along.

Why in Envoy?

Envoy is Efficient and Flexible

Envoy is a good match for Mesh (e.g. Istio)

Envoy is a Platform

- Control Plane Extensions
- Data Plane Extensions
- Reporting Extensions

Extend Envoy without Recompiling

Dynamic Update Envoy Extensions in WebAssembly

Misbehaving Extensions do not take down all of Envoy

Implemenation

Memory Management

- Currently copy-in copy-out
- Simple but not that efficient

VM Tradeoffs

- WAVM allows per-compiling (i.e. in control plane)
- V8 has many miles on it
- Null Sandbox is good for development and debugging

What happens when something goes wrong?

- On WASM SEGV we can fail closed (e.g. for HTTP filters).