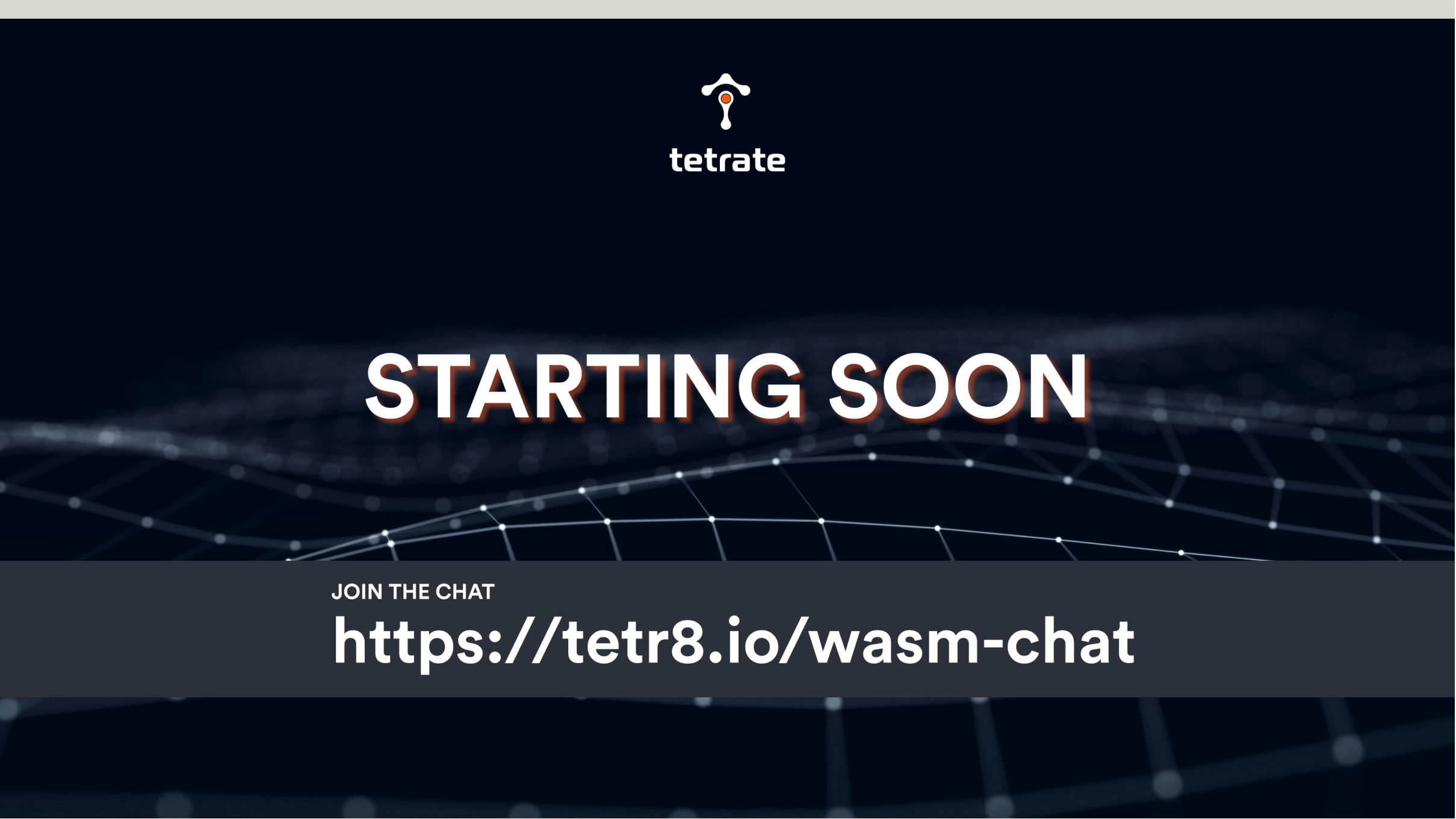

tetrate
STARTING SOON
JOIN THE CHAT
https://tetr8.io/wasm-chat

Welcome

- Duration: 90 minutes
- Labs: https://tetratelabs.github.io/wasm-workshop/
- Questions/chat: https://tetr8.io/wasm-chat

Agenda

1. Intro: What is WebAssembly (Wasm)?
2. Wasm in Envoy
3. Proxy-wasm & SDKs
4. Wasm in Istio - WasmPlugin
5. Q&A

Introduction to WebAssembly (Wasm)

WebAssembly (Wasm)

- Portable binary, open standard
- Isolated from the host and executed in a sandbox environment
  - Virtual machine (VM)
- Communicates to host via an API

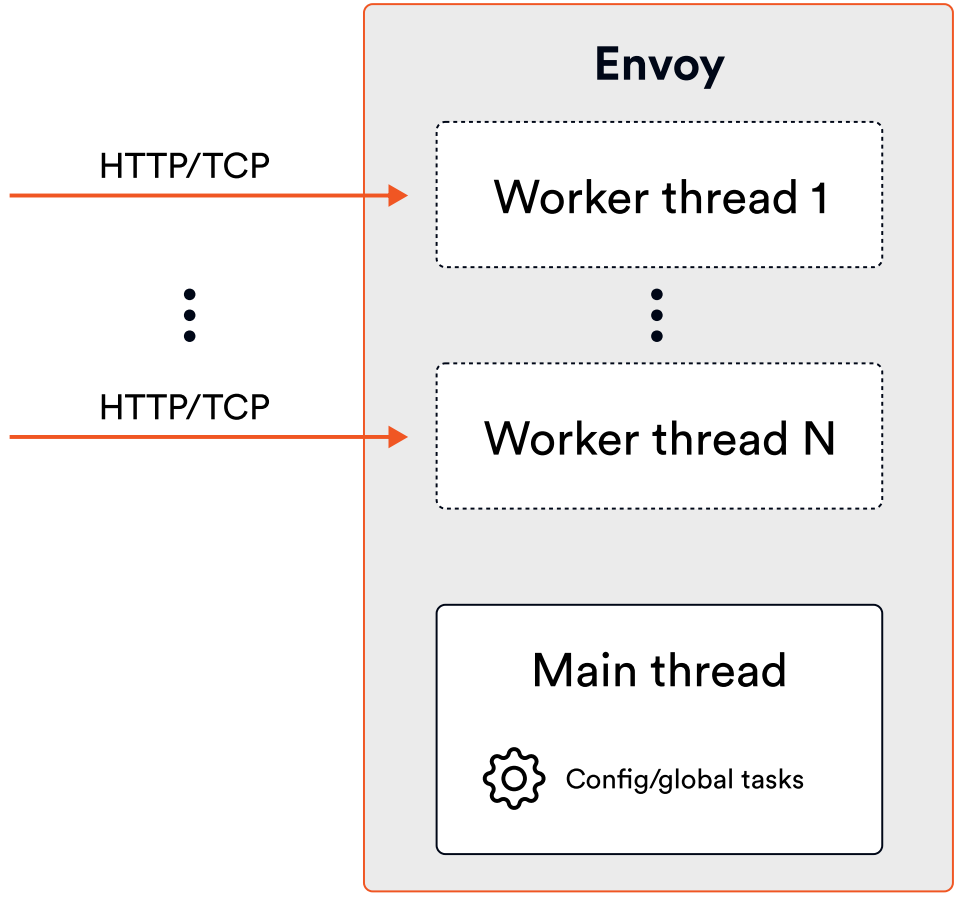
Wasm in Envoy and Istio

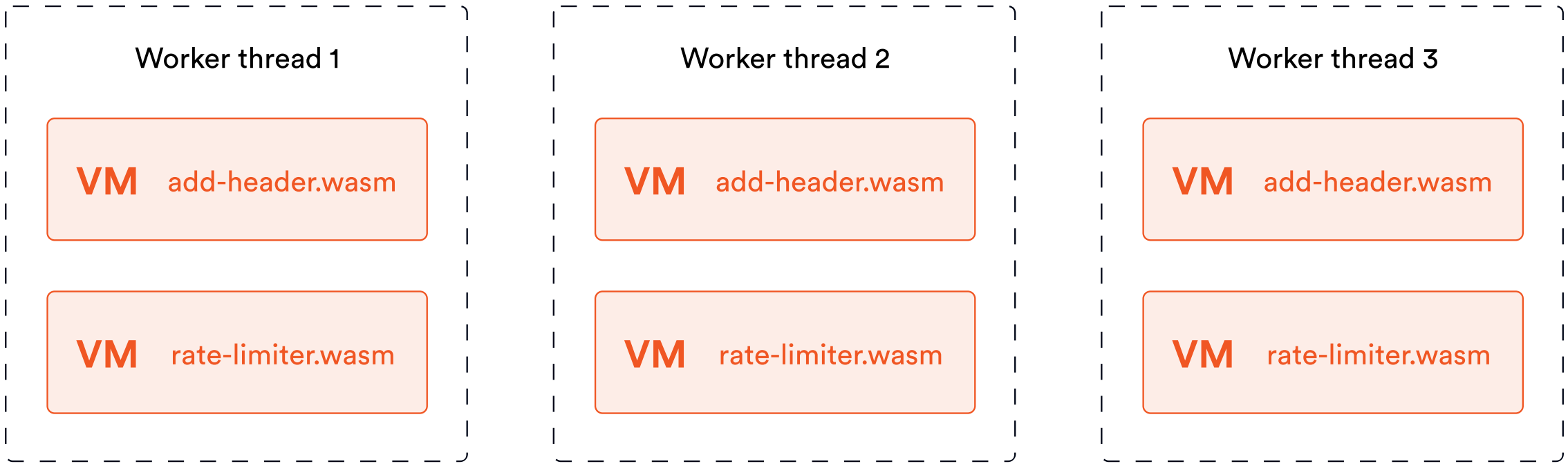
Extending Envoy

- Native C++ filters
  - Recompile Envoy
- Lua script filters
  - Inline with config or in a separate file
- Wasm filters
  - Filter in a separate module, pulled from OCI registry

Wasm in Envoy

- Subset of a V8 VM
  - Used in Chrome and Node.js
- Multi-threaded model
  - Main + worker threads
- Threads are independent
- Wasm VM = loaded .wasm module

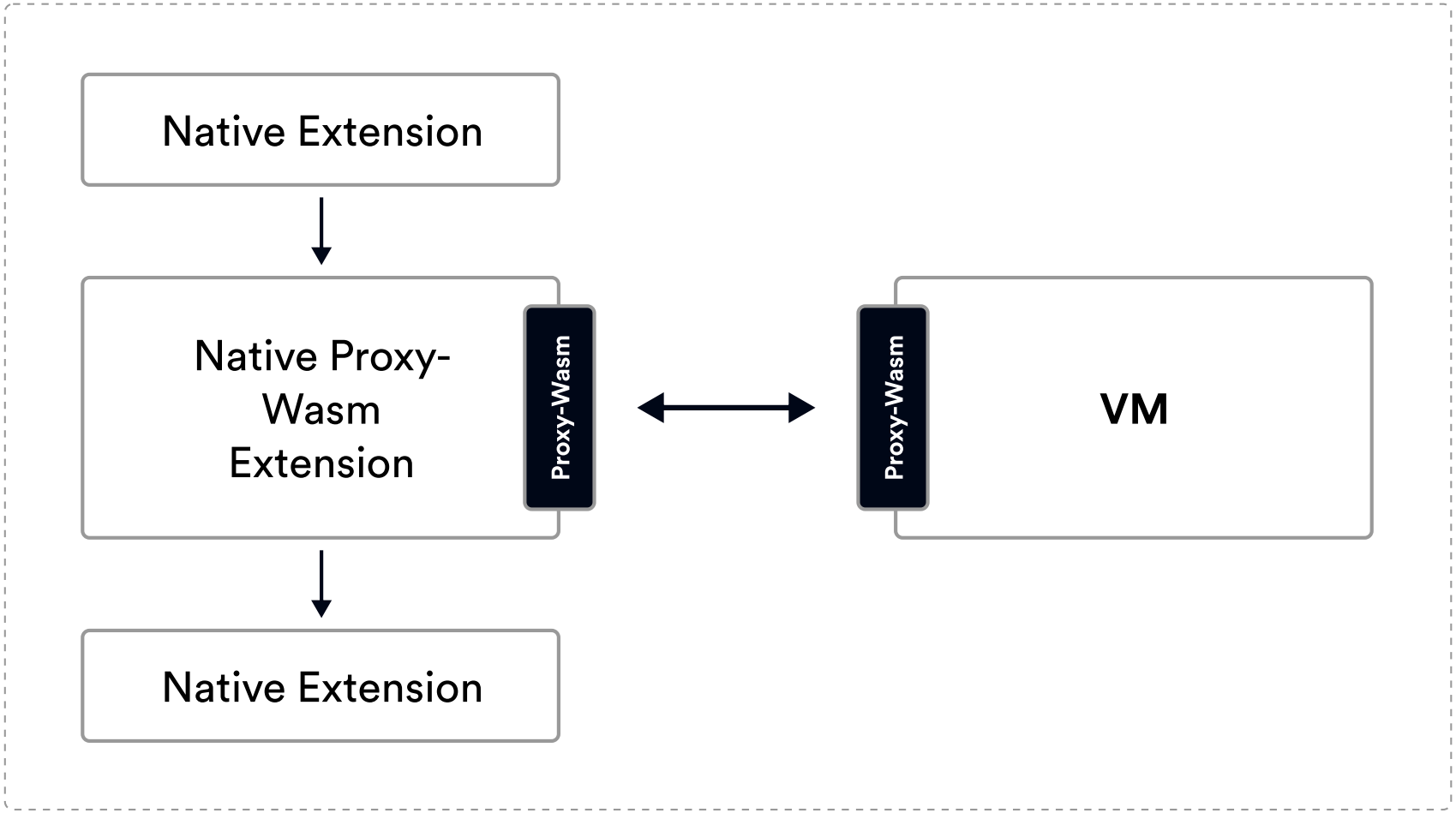



Worker thread 1
VM add-header.wasm
VM rate-limiter.wasm
Worker thread 2
VM add-header.wasm
VM rate-limiter.wasm
Worker thread 3
VM add-header.wasm
VM rate-limiter.wasm

Proxy-Wasm ABI

- ABI (Application Binary Interface) standard
  - Functions and callbacks: `GetHttpResponseBody`, `SendHttpResponse`, `GetSharedData`, ...
- Proxy agnostic APIs
  - Makes Wasm filters portable
- Low-level

Worker Thread



Extension types

- Extension on HTTP/TCP path
  - Run per each worker thread
- Singleton (Wasm service)
  - Run per Envoy instance

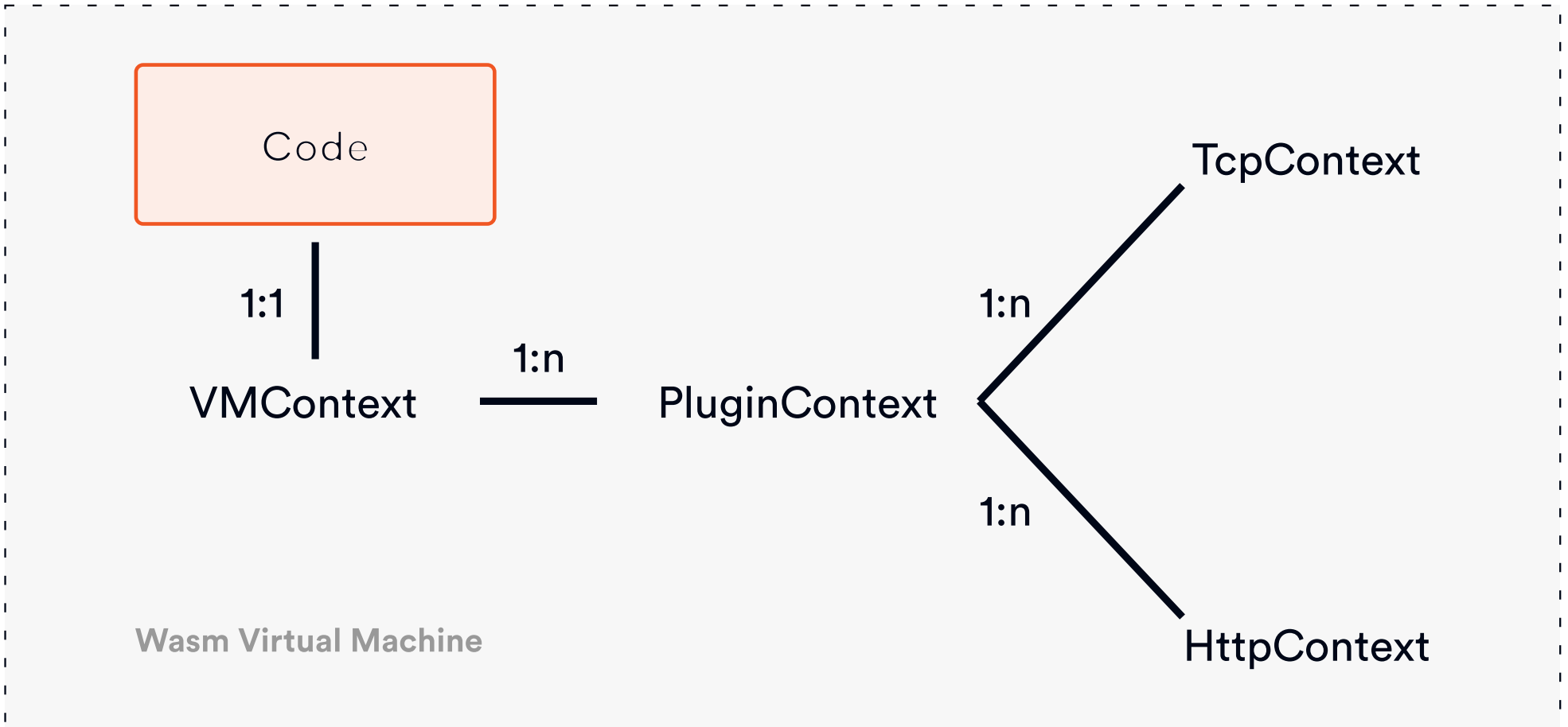
Proxy-Wasm and Go SDK

Proxy-Wasm SDKs

- TinyGo SDK
- AssemblyScript SDK
- C++ SDK
- Rust SDK
- Zig SDK

Proxy-Wasm Go SDK

- Powered by TinyGo
  - Supports a subset of standard Go packages
- Concepts:
  - Contexts
  - Hostcall API
  - Entrypoint


Code
1:1
VMContext
1:n
PluginContext
1:n
TcpContext
1:n
HttpContext
Wasm Virtual Machine

VMContext

```
type VMContext interface {
  OnVMStart(vmConfigurationSize int) OnVMStartStatus
  NewPluginContext(contextID uint32) PluginContext
}
```

PluginContext

```
type PluginContext interface {
  OnPluginStart(pluginConfigurationSize int) OnPluginStartStatus
  OnPluginDone() bool

  OnQueueReady(queueID uint32)
  OnTick()

  NewTcpContext(contextID uint32) TcpContext
  NewHttpContext(contextID uint32) HttpContext
}
```

Hostcall API

- Ways to interact with Envoy proxy
- Methods for:
  - Reading configuration
  - Setting up shared queue & performing queue operations
  - Dispatching HTTP calls
  - Retrieving headers, trailers, body, ...

Entrypoint

```
func main() {
  proxywasm.SetVMContext(&myVMContext{})
}

type myVMContext struct { .... }

var _ types.VMContext = &myVMContext{}
```

How to get started?

DEVELOPMENT ENVIRONMENT

- Code
- SDK (Proxy-Wasm Go SDK)
- Compiler (TinyGo)

- Envoy proxy (func-e CLI)
- Configuration

Lab: Minimal Wasm extension

HTTP/TCP manipulation and configuration values

HTTP/TCP manipulation

- HTTP:
  - `GetHttp[Request|Response][Headers|Body|Trailers]`
  - `ReplaceHttp[Request|Response][Headers|Trailers]`
  - `RemoveHttp[Request|Response][Headers|Trailers]`
  - `Add[Request|Response][Header|Trailer]`
- TCP:
  - `[Get|Append|Prepend|Replace][Downstream/Upstream]Data`

Configuration

- Plugin and VM configuration
  - `GetVMConfiguration` to retrieve `vm_config.configuration`
  - `GetPluginConfiguration` to retrieve `config.configuration`
- Available during `OnVMStart` and `OnPluginStart` only

Other functionality

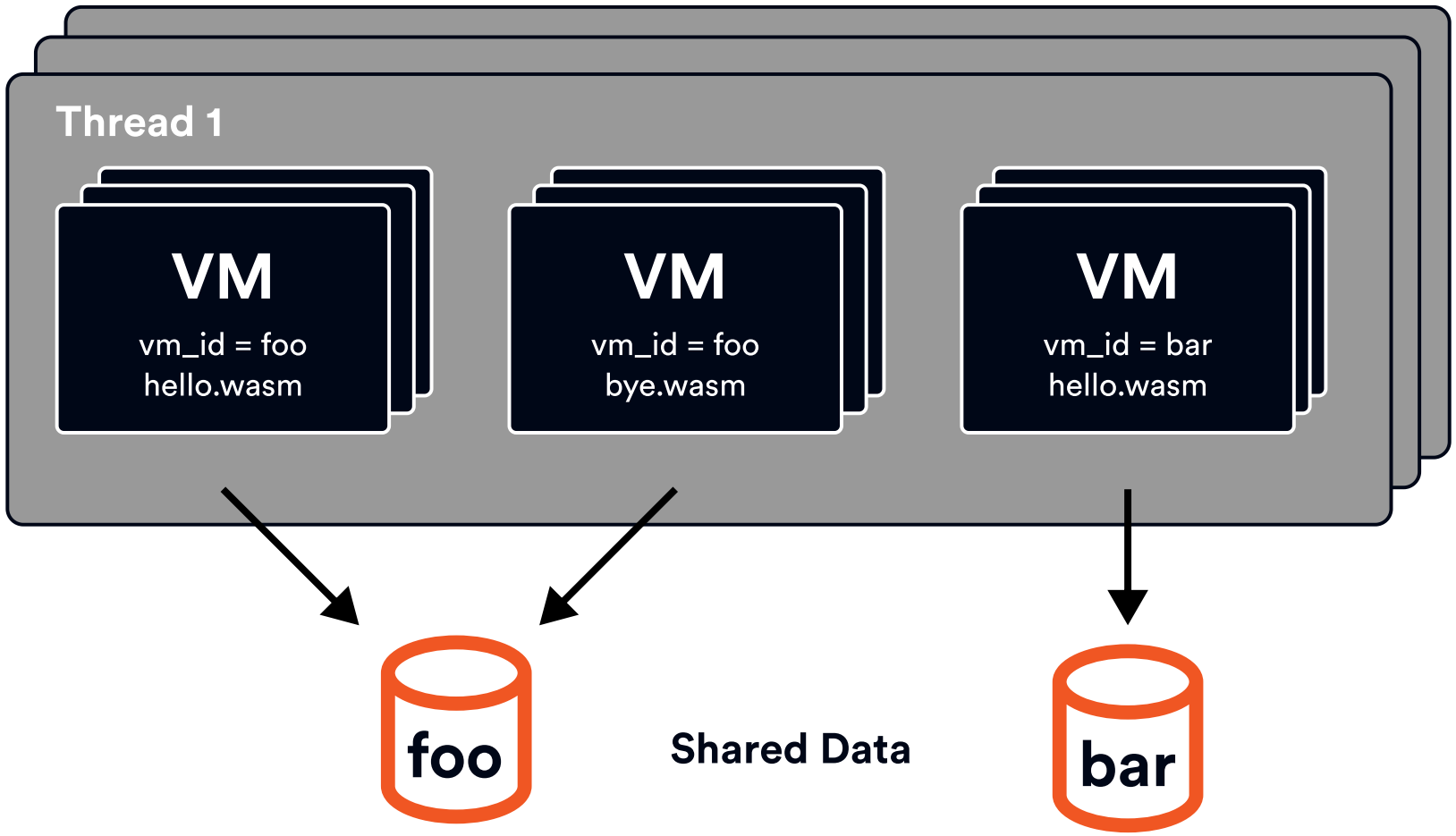
- Sending an HTTP response (`SendHttpResponse`)
- Making HTTP requests to clusters (`DispatchHttpCall`)
- Getting/setting property/metadata from Envoy
  - Envoy implements multiple attributes that can be retrieved using this call

Lab: Configuration & headers

Sharing data

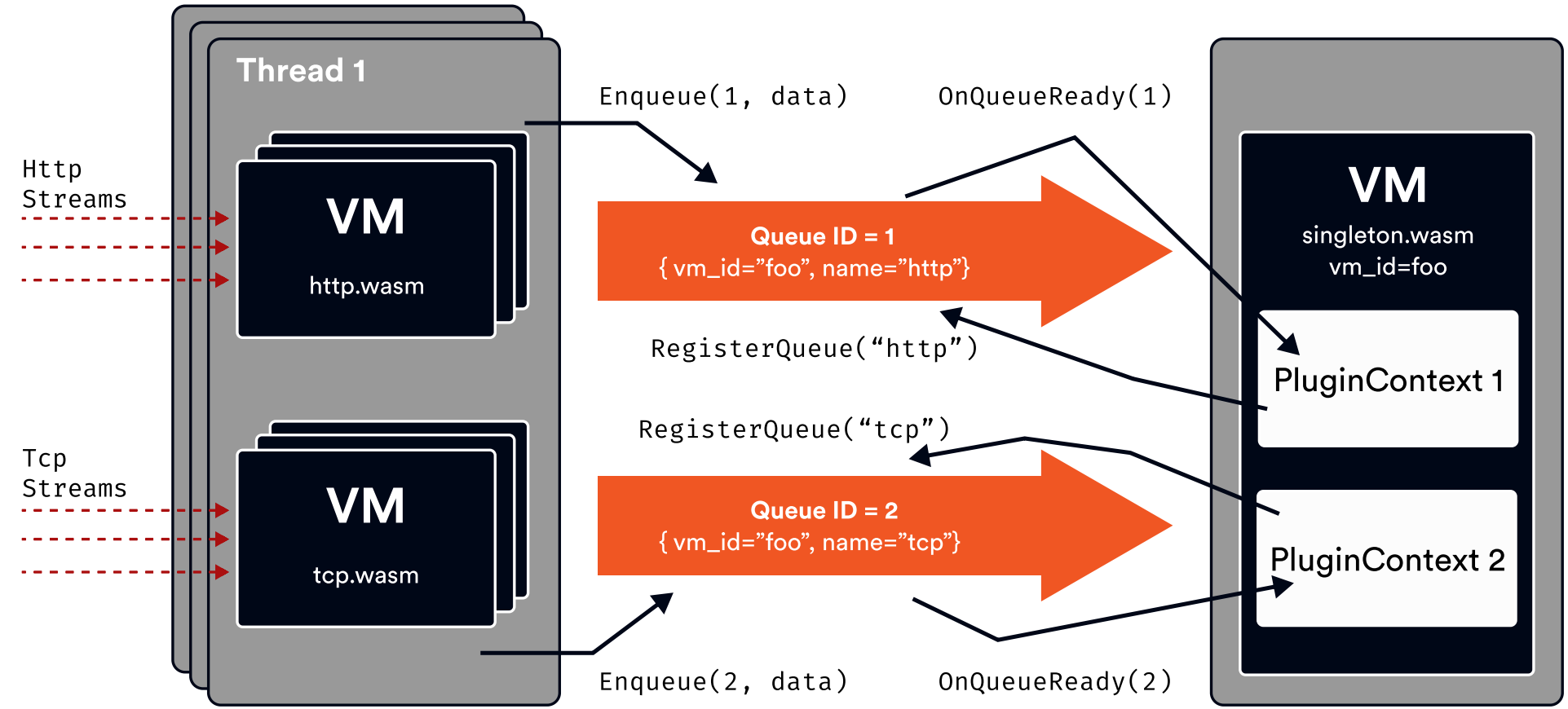
Sharing data

- Store key/value pairs
  - `SetSharedData(string, []byte, uint32)` and `GetSharedData(string) : ([]byte, unint32, error)`
- Shared across VMs with the same `vm_id`


Thread 1
VM
vm_id = foo
hello.wasm
VM
vm_id = foo
bye.wasm
VM
vm_id = bar
hello.wasm
foo
Shared Data
bar

Lab: Sharing data

Using a queue


Thread 1
Http
Streams
VM
http.wasm
Tcp
Streams
VM
tcp.wasm
Enqueue(1, data)
OnQueueReady(1)
Queue ID = 1
{ vm_id="foo", name="http"}
RegisterQueue("http")
RegisterQueue("tcp")
Queue ID = 2
{ vm_id="foo", name="tcp"}
Enqueue(2, data)
OnQueueReady(2)
VM
singleton.wasm
vm_id=foo
PluginContext 1
PluginContext 2

Metrics

Metrics

- Counters, histograms, gauge
- `Define[Counter|Histogram|Gauge]Metric`
- Functions to increment, add, record the values

Lab: Adding metrics

Deploy and run Wasm plugins

Wasm and Istio history

- Istio 1.4: no way to deploy/run Wasm plugins
  - Istio maintains its Envoy fork
  - Mixer for extensibility (auth policies & telemetry)
  - Inefficient

Wasm and Envoy history

- Knowledge of C++ required
  - Building extensions with the Envoy binary
  - Updating all instances (not trivial)
- RBAC and JWT filters are already in Envoy

Enter WebAssembly

- Wasm support in Envoy started in 2018
- Features included in Istio 1.5
  - New extensibility model using Wasm (no more Mixer)
  - Stats and metadata exchange as Wasm plugins
  - EnvoyFilter resource
- ABI & C++, Rust, and AssemblyScript SDKs

EnvoyFilter resource

- Very light (almost 0) abstraction over Envoy configuration
  - You have to understand Envoy
- Loading the .wasm files was painful
  - Load from HTTP - fetch failures

Improvements and path to WasmPlugin

- Istio 1.9: Istio agent intercepts CRD and fetches binaries
  - Still uses EnvoyFilters
- Istio 1.12: WasmPlugin API
  - No need for EnvoyFilter anymore!

WasmPlugin resource

- Specify workload selectors
- Host Wasm binary in OCI registry (or HTTP)

```yaml
apiVersion: extensions.istio.io/v1alpha1
kind: WasmPlugin
metadata:
  name: hello-world-wasm
  namespace: default
spec:
  selector:
    labels:
      app: hello-world
  url: oci://my-registry/tetrate/hello-world:v1
  pluginConfig:
    greeting: hello
    something: anything
```

Development workflow

1. Write your extension in your language
2. Build the `.wasm` file
3. Build the OCI image (`docker build`)
4. Push the OCI image (`docker push`)
5. Deploy WasmPlugin resource

Lab: WasmPlugin

What's next?

- Image pull secrets
- 1st class Wasm image cache support in Istio

Questions?

Upcoming Events

- Istio 0 to 60 workshops (February 17th)
  - PST: https://www.tetrate.io/istio-workshop-americas/
  - IST: https://www.tetrate.io/istio-workshop-asia-pacific/

Thank you!

RESOURCES

- Tetrate Academy: https://academy.tetrate.io
  - Envoy & Istio Fundamentals Course
  - Certified Istio Administrator
- Istio weekly: https://www.youtube.com/c/tetrate
- Labs: https://tetratelabs.github.io/wasm-workshop/