Developing & Debugging WebAssembly Filters

Idit Levine & Yuval Kohavi

Idit Levine | Founder & CEO, Solo.io

Yuval Kohavi | Chief Architect, Solo.io

solo.io

Istio Adoption with Gloo Mesh

Crawl

Upstream Istio support (24 X 7)

LTS (N - 3)

FIPS, ARM

Tech Advisory

Walk

Developer portal

API Gateway

Security (EW)

Observability

Zero-trust

Run

Approval Processes

Rollback

Delegation

WASM

Fly

Multi Cluster

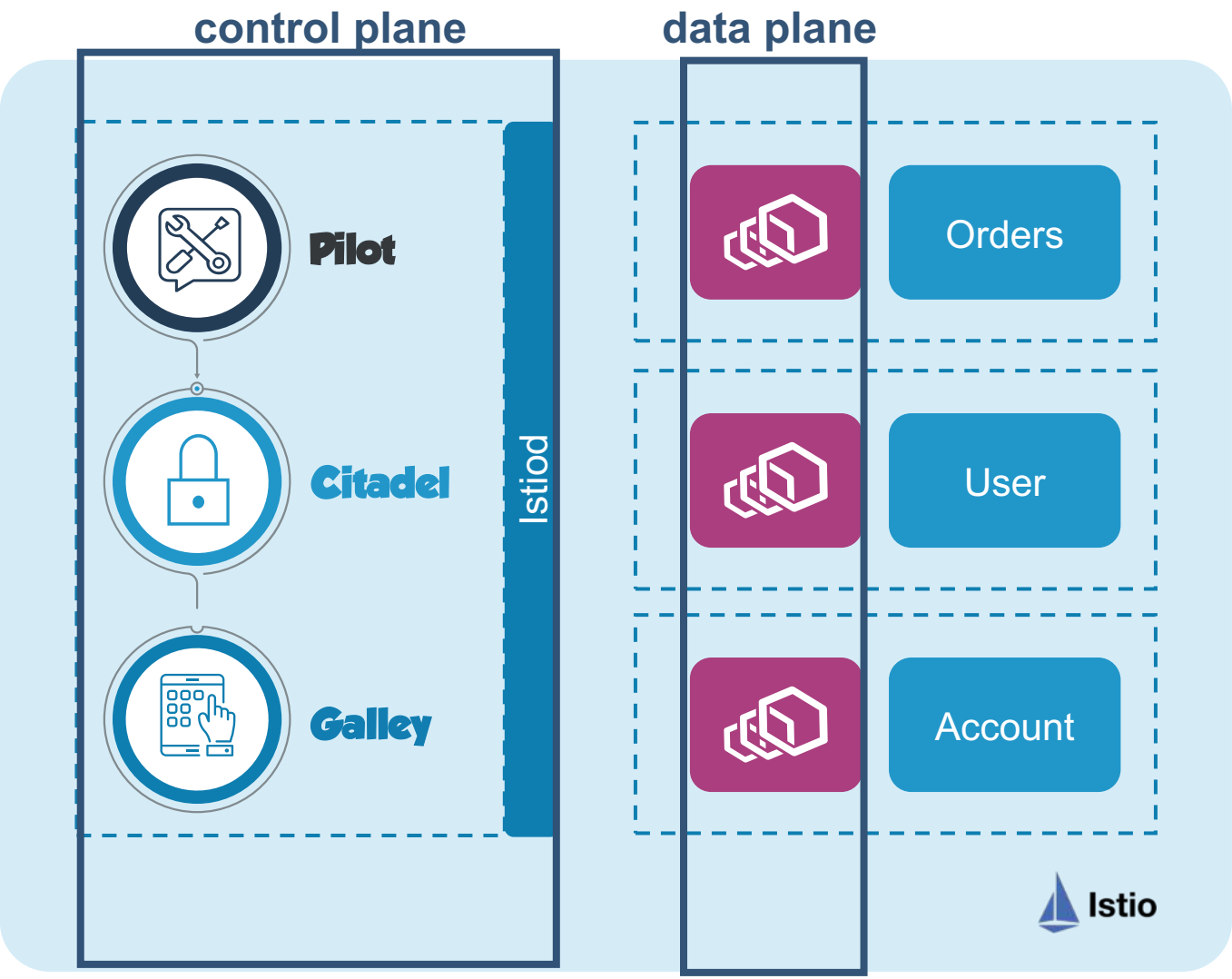
Global Service

Failover

Multi Mesh

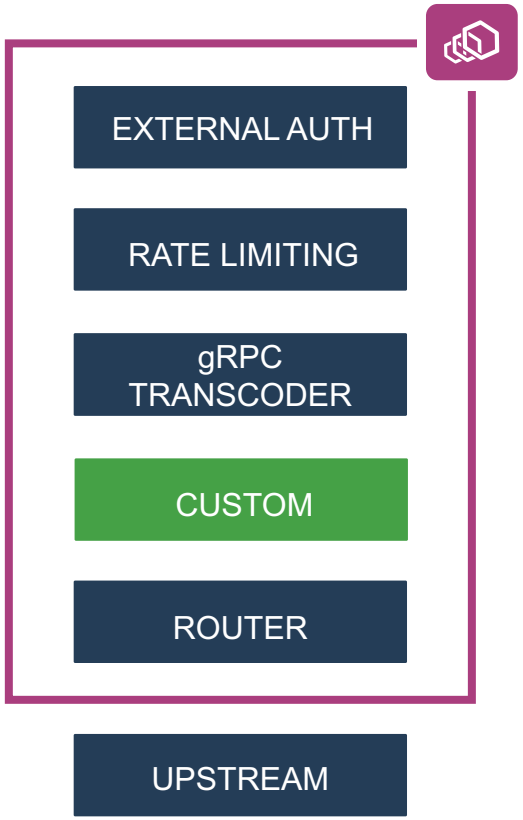
solo.io

Understanding Istio: Control and data planes



solo.io

Extend Envoy Proxy with Filter



Build Custom Envoy Filter

Develop: Envoy Filters are written in C++ Asyc

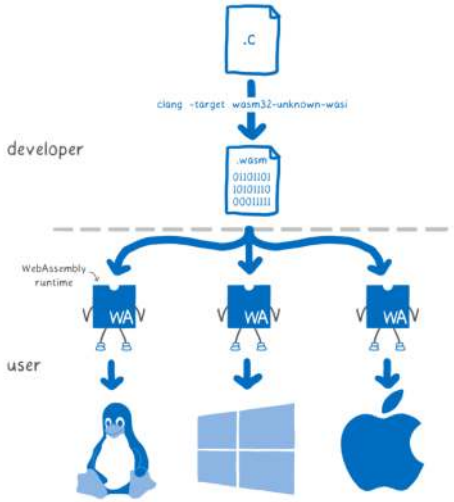
Build: need to recompile and maintain a build of Envoy

Bazel

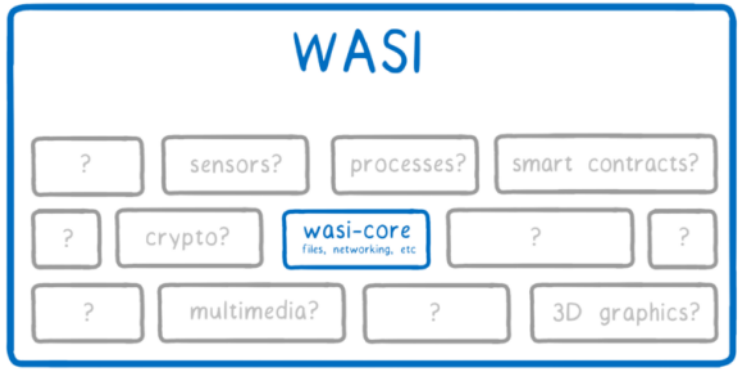
solo.io

Web Assembly

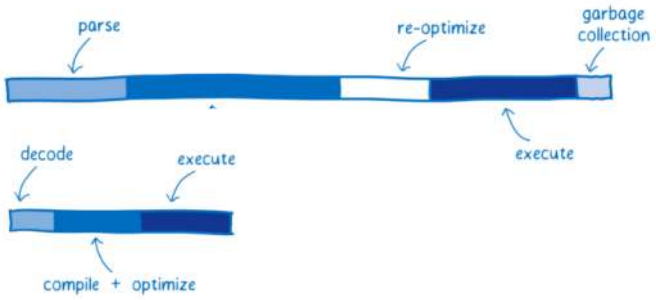
Portable



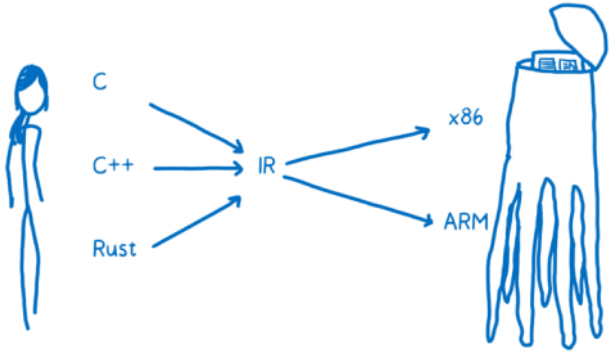
Outside the Web



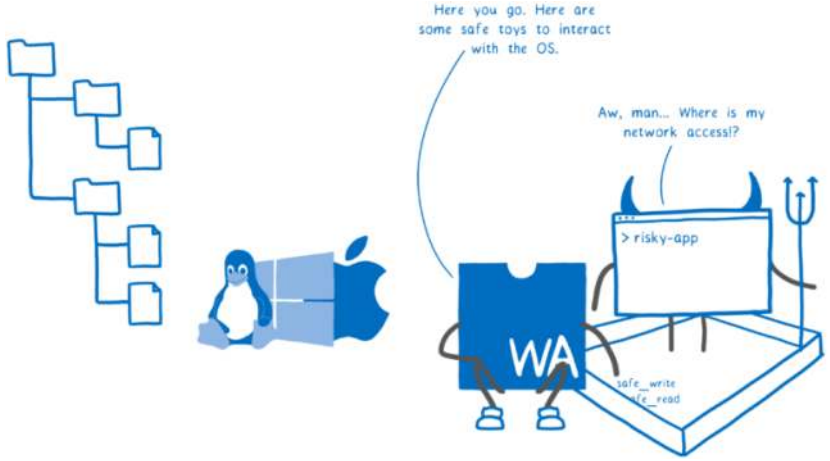
Fast



Any Language

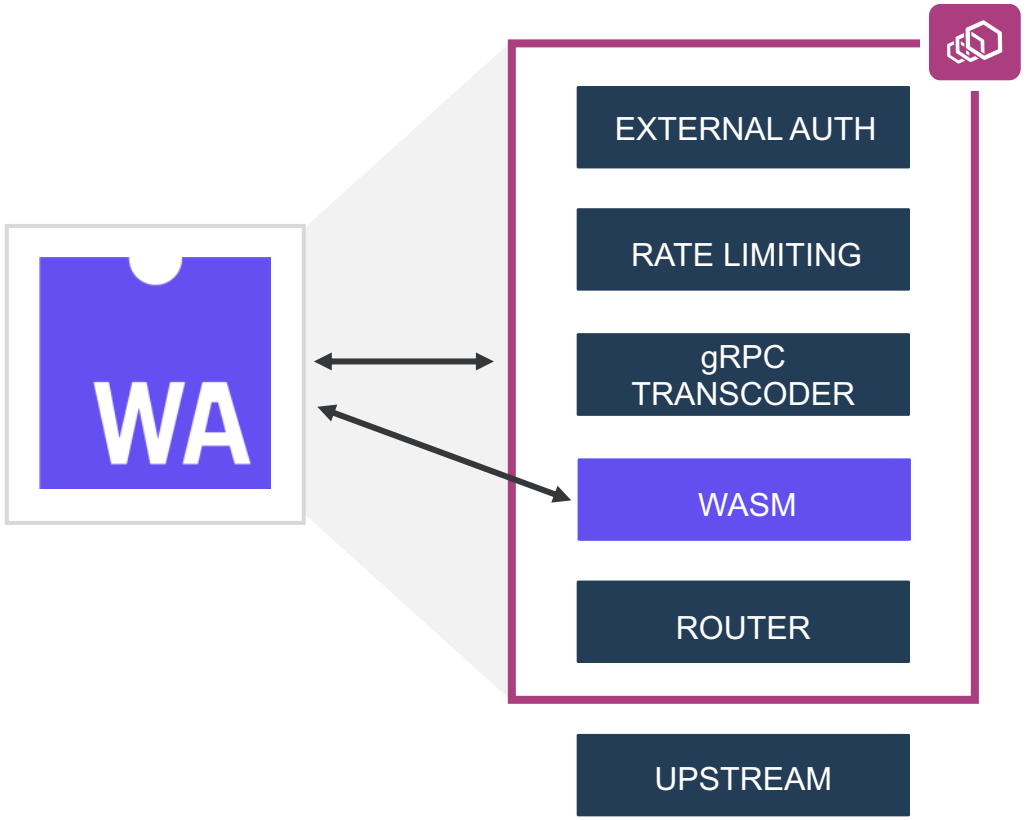


Secure



solo.io

Extend Envoy Proxy with Web Assembly (Wasm)



Why WebAssembly?

Polyglot: Envoy Filters are written in C++ and Wasm expands to any language

Secure and Reliable: Wasm runs in isolated VM, can dynamically update w/o Envoy restarts, no hard dependencies or cascading failures

Speed: Near native performance

Sustainable: Eliminates need to recompile and maintain a build of Envoy

solo.io

User Experience

solo.io

Solomon Hykes @solomonstre · Mar 27, 2019

If WASM+WASI existed in 2008, we wouldn't have needed to created Docker. That's how important it is. Webassembly on the server is the future of computing. A standardized system interface was the missing link. Let's hope WASI is up to the task!

> **Lin Clark** @linclark · Mar 27, 2019
>
> WebAssembly running outside the web has a huge future. And that future gets one giant leap closer today with...
>
> 📢 Announcing WASI: A system interface for running WebAssembly outside the web (and inside it too)
>
> hacks.mozilla.org/2019/03/standa...
>
> Show this thread

34 745 1.7K

solo.io

Technology	LXC	WA WEBASSEMBLY
User Experience	docker	WA WEB AssemblyHub

solo.io

Web Assembly lifecycle

solo.io

Build

ABI: Application Binary Interface

HTTP (L7) extensions

`proxy_on_http_request_headers`

- params:
  - `i32 (uint32_t) context_id`
  - `i32 (size_t) num_headers`
  - `i32 (bool) end_of_stream`
- returns:
  - `i32 (proxy_action_t) next_action`

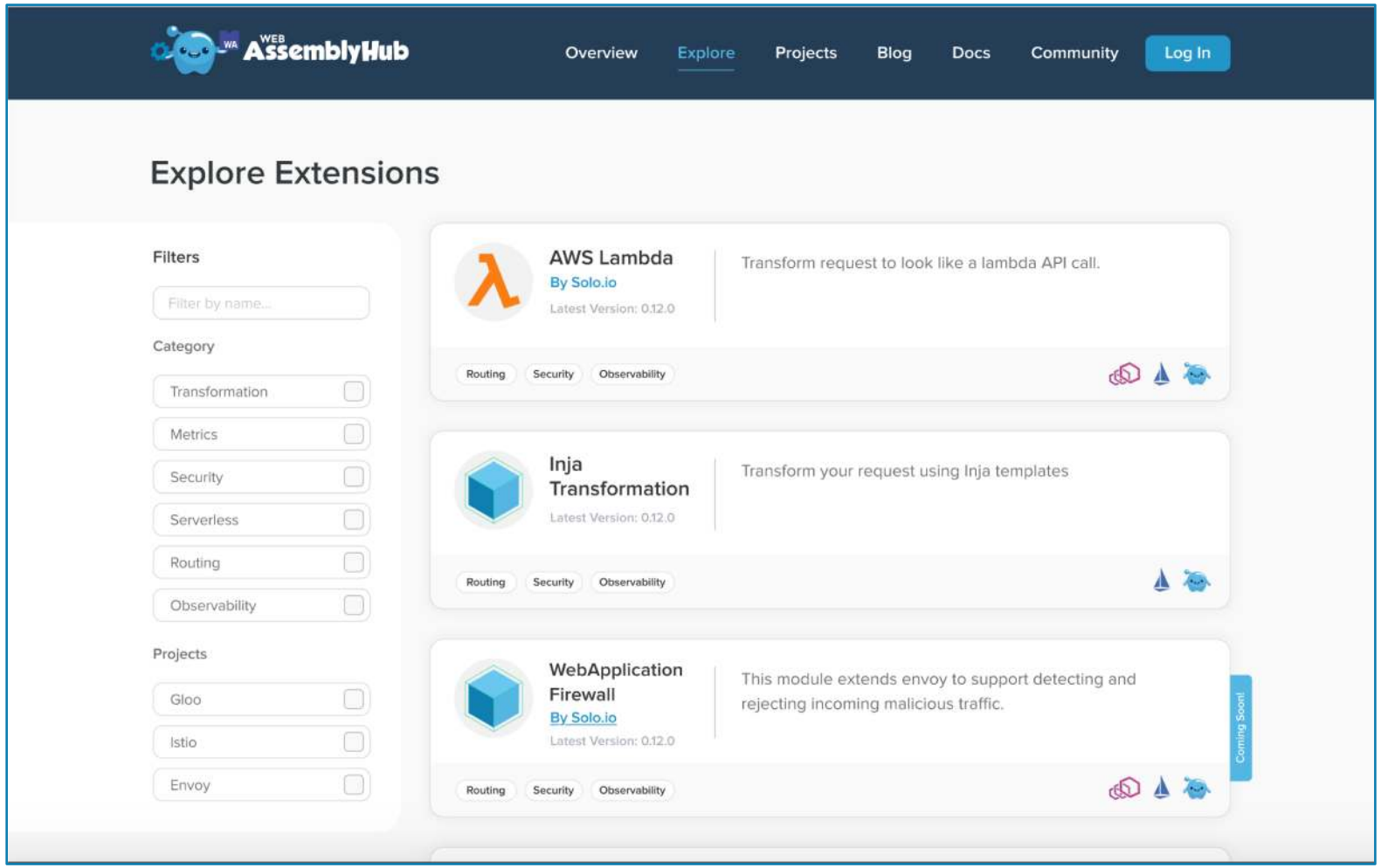
Called when HTTP request headers are received from the client. Headers can be retrieved using `proxy_get_map` and/or `proxy_get_map_value`.

```
> meshctl wasm init addheader-filter --language rust
> meshctl wasm build rust -t
    webassemblyhub.io/yuval/addheader-rust:v1 ./addheader-filter
```


solo.io

Build

Store




```
> meshctl wasm push webassemblyhub.io/yuval/addheader-rust:v1
```


solo.io

Build

Store

WASM Artifact Image Specification

```
[
  {
    "mediaType": "application/vnd.module.wasm.config.v1+json",
    "digest": "sha256:d0a165298ae270c5644be8e9938036a3a7a5191f6be03286c40874d761c18abf",
    "size": 125,
    "annotations": {
      "org.opencontainers.image.title": "runtime-config.json"
    }
  },
  {
    "mediaType": "application/vnd.module.wasm.content.layer.v1+wasm",
    "digest": "sha256:5e82b945b59d03620fb360193753cbd08955e30a658dc51735a0fcbc2163d41c",
    "size": 1043056,
    "annotations": {
      "org.opencontainers.image.title": "filter.wasm"
    }
  }
]
```

The following is the runtime config stored as the `application/vnd.module.wasm.config.v1+json` layer:

```
  "type": "envoy_proxy",
  "abi_version": "v0-541b2c1155fffb15ccde92b8324f3e38f7339ba6",
  "config": {
    "root_ids": [
      "add_header_root_id"
    ]
  }
}
```

Solo.io Proposes OCI Format Extension for WASM

September 14, 2020 containers, kubernetes, oci, OCI specification

by Mike Vizard

Solo.io has launched an initiative to define a format to extend the Open Container Initiative (OCI) image specification to standardize how an image bundles and stores metadata when developers write code using WebAssembly (WASM).

Hollie Ratke
5 months ago

Solo.io Borrows OCI Spec to Bundle WebAssembly Modules

Following last year's release of its WebAssembly Hub, Solo.io has released a proposal for a new WebAssembly (WASM) Open Container Initiative (OCI) image specification that it says will "define how to bundle WASM modules as OCI images to make it easy to build, pull, publish, and execute."

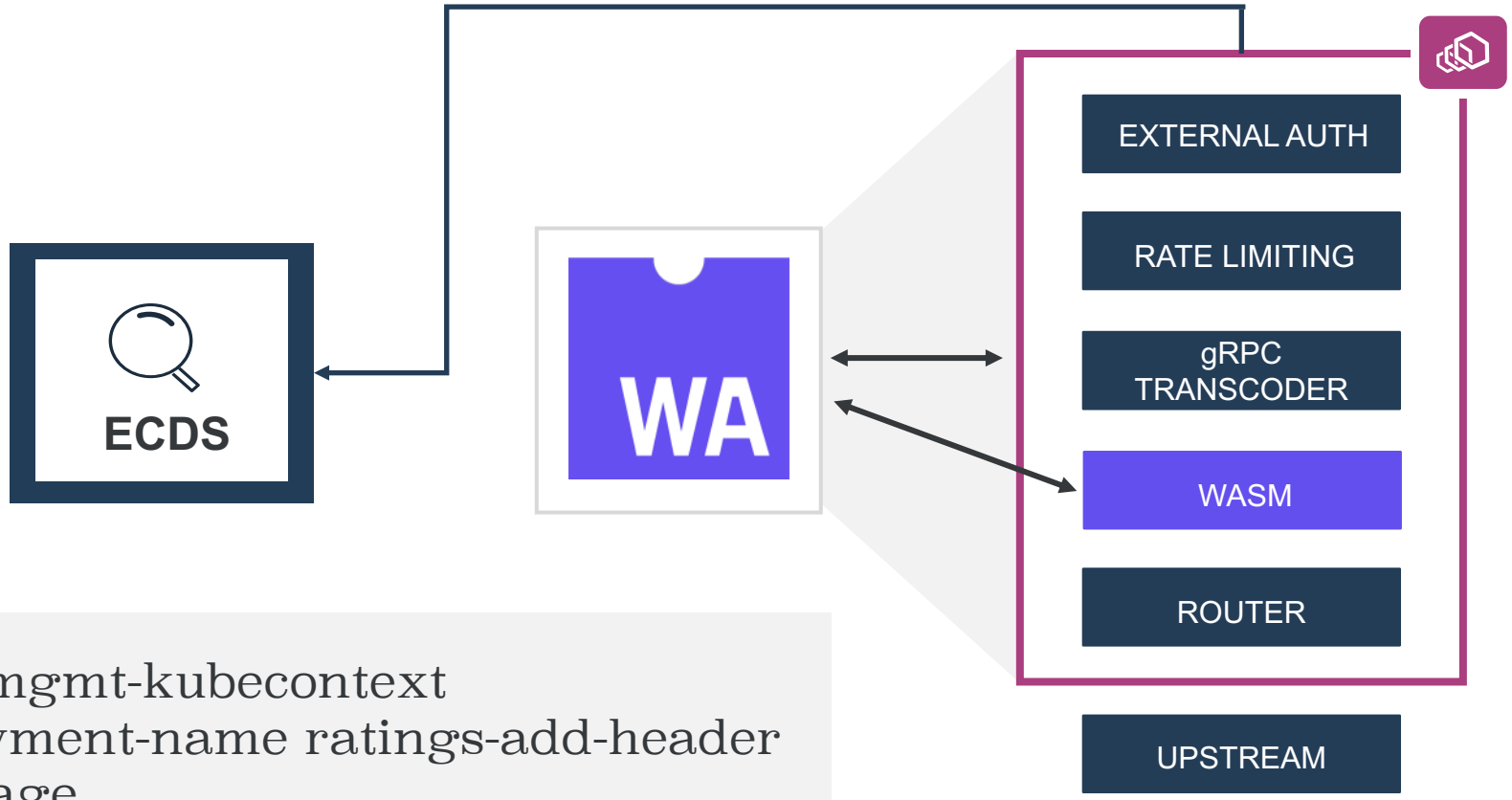
DEVELOPMENT / OPEN SOURCE / SERVICE MESH

Solo.io Borrows OCI Spec to Bundle WebAssembly Modules

17 Sep 2020 11:02am, by Mike Melanson

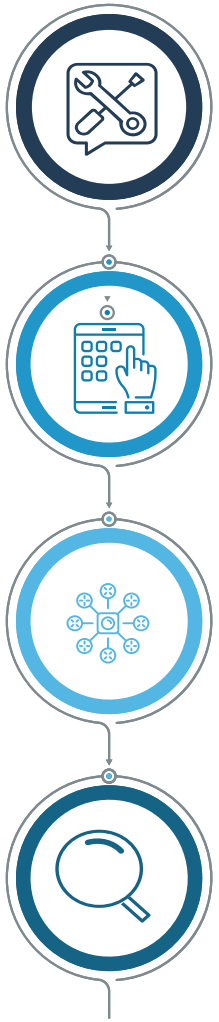
solo.io

Extension Config Discovery Service




```
> meshctl wasm deploy istio --mgmt-kubecontext
    kind-mgmt-cluster --deployment-name ratings-add-header
    --namespace bookinfo --image
    webassemblyhub.io/yuval/addheader-rust:v1
    --cluster mgmt-cluster --labels app=ratings
```


solo.io



Build

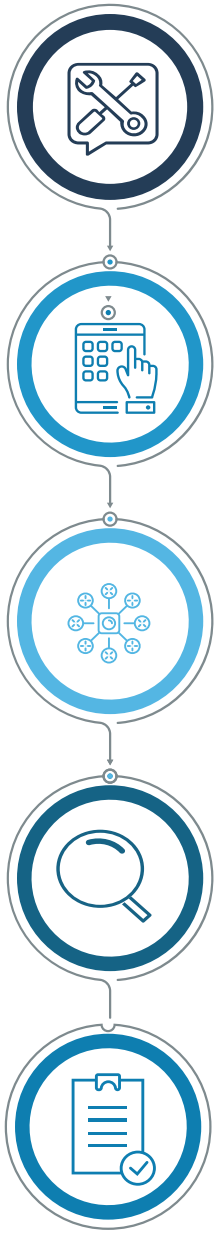
Store

Deploy

Debug


```
meshctl wasm debug workloadSelector
```


solo.io



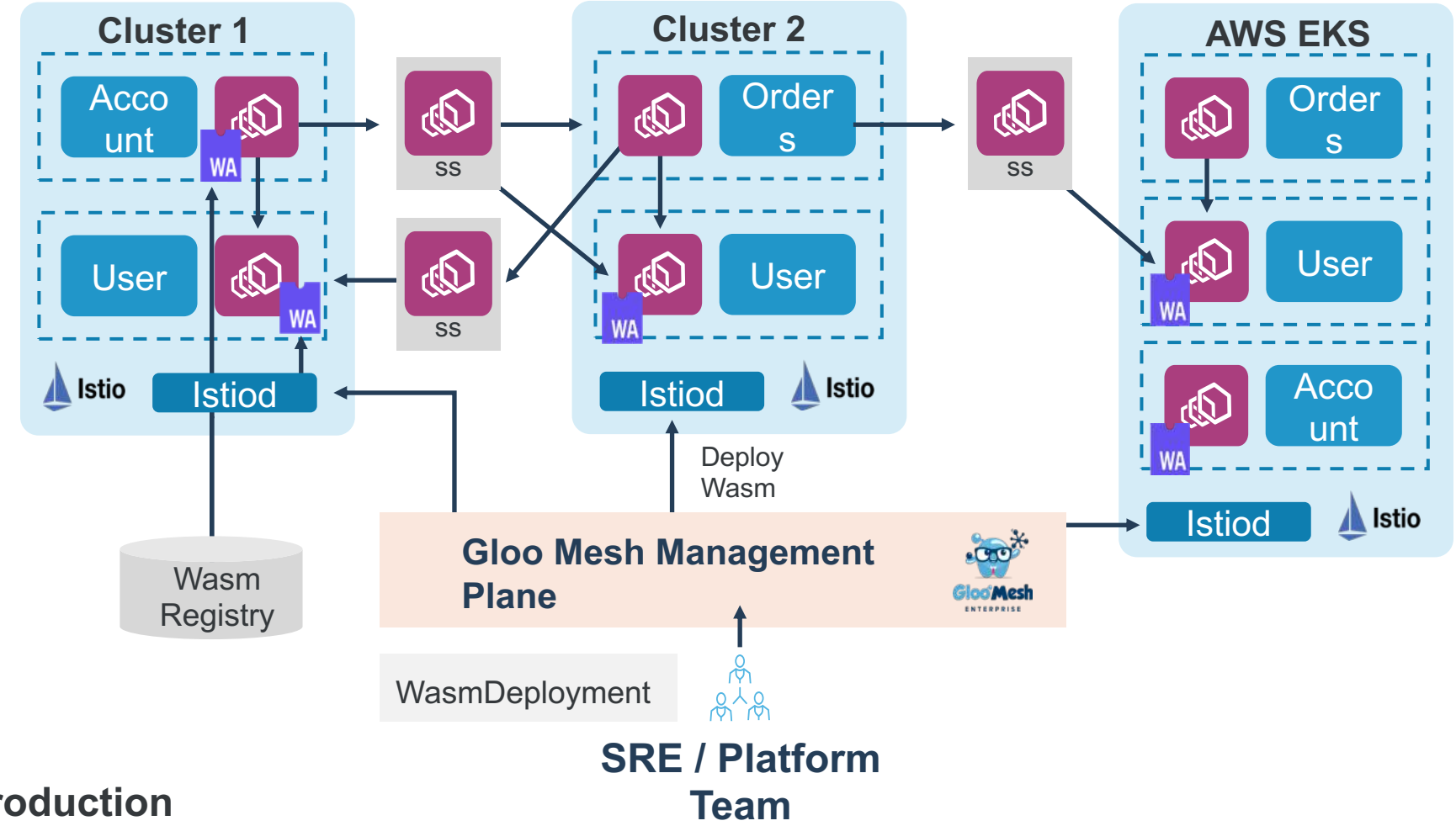
Build

Store

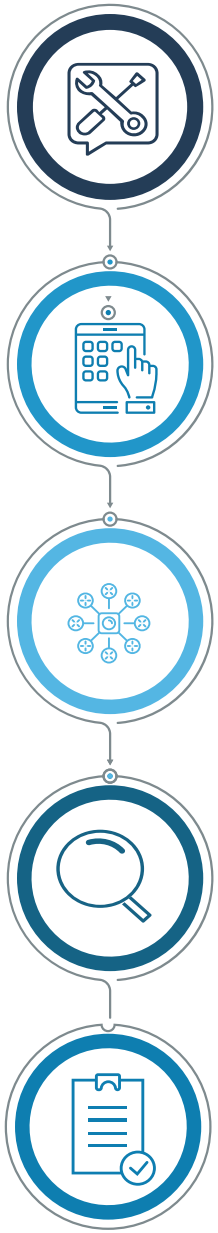
Deploy

Debug

Debug in Production



solo.io



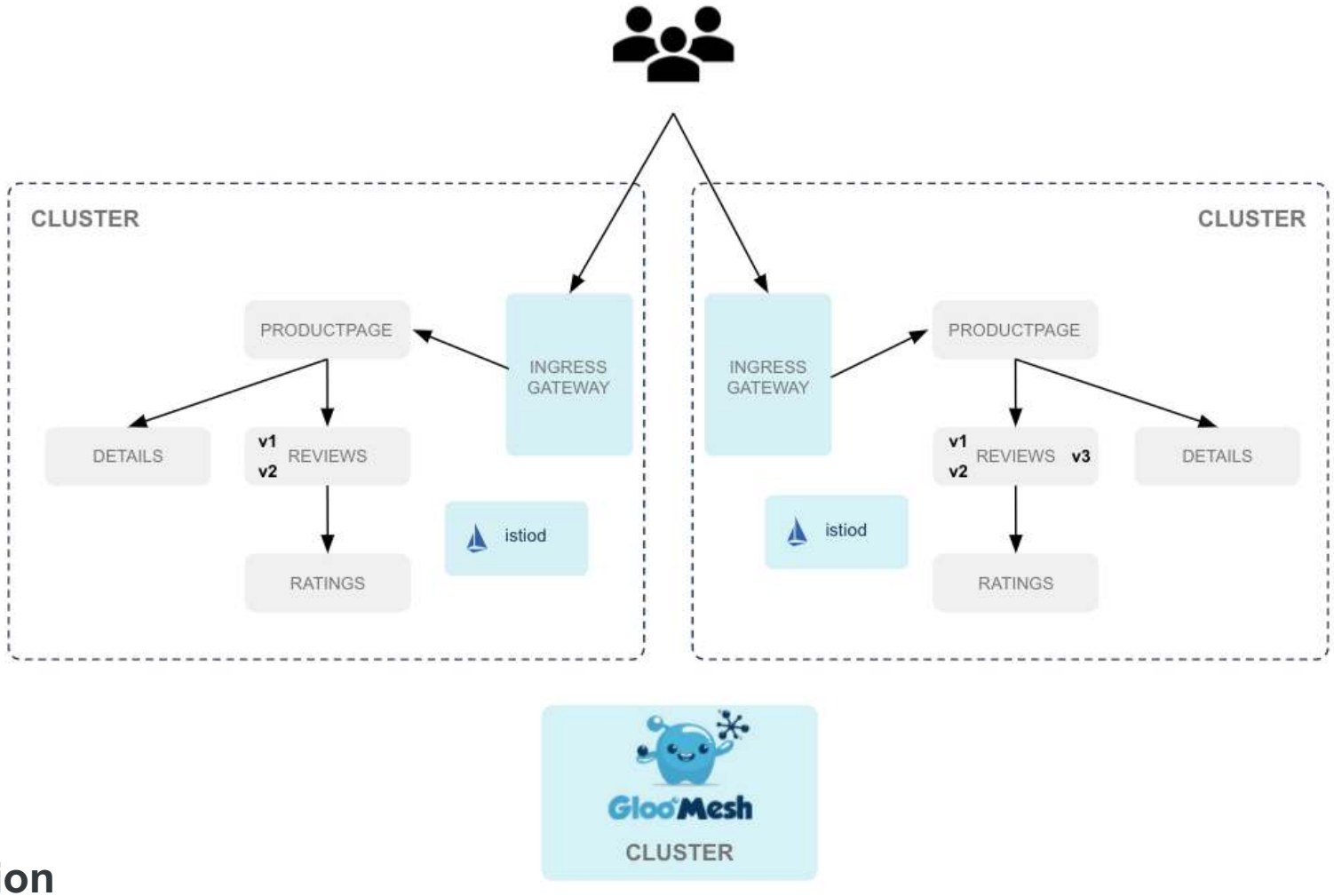
Build

Store

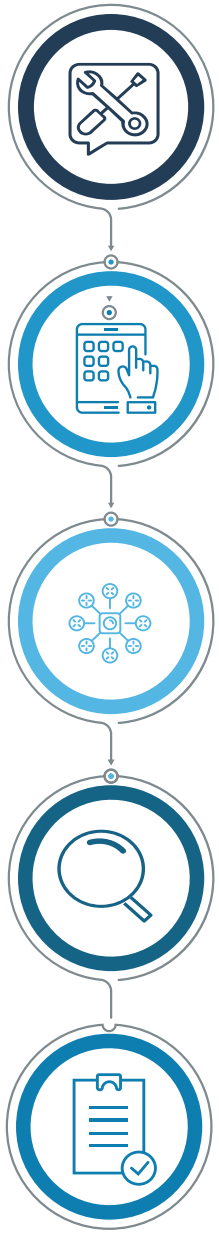
Deploy

Debug

Debug in Production



solo.io



Build

Store

Deploy

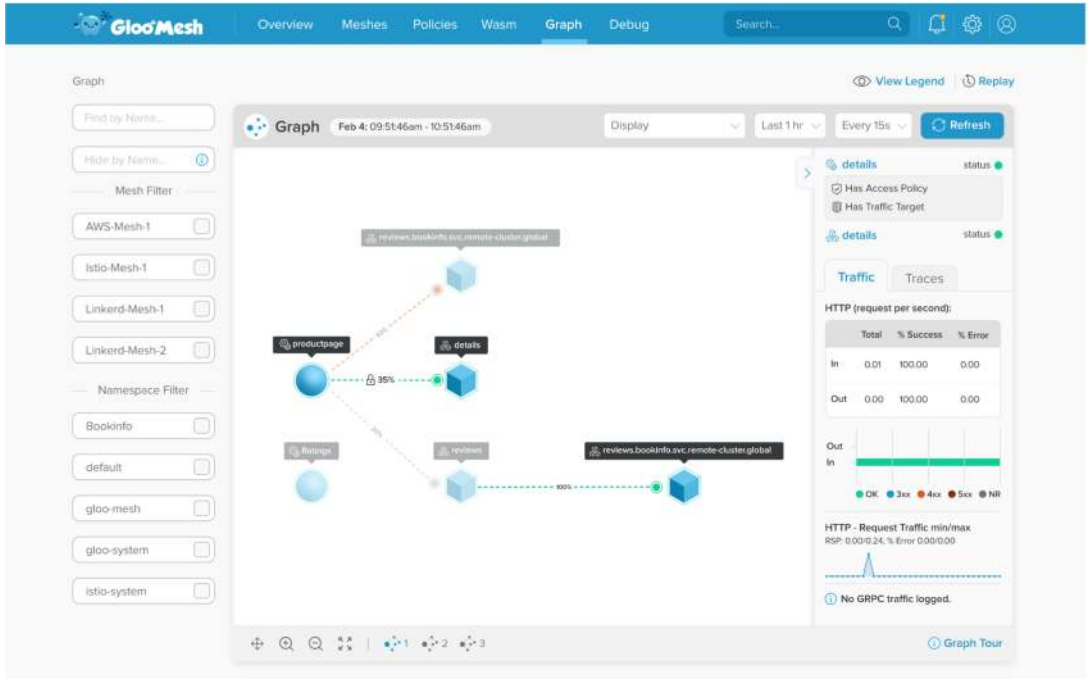
Debug

Debug in Production

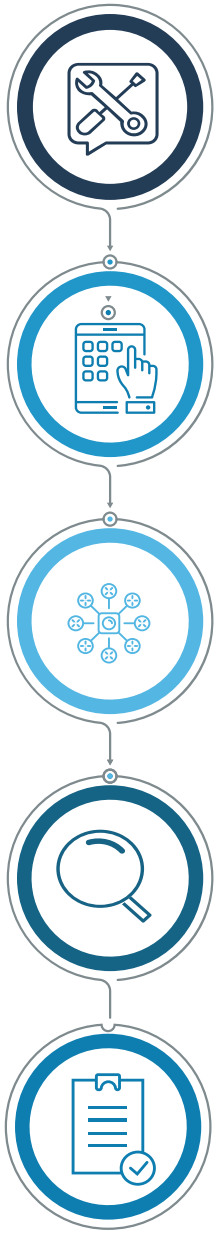
Debug Logs

Access Logs

Metrics



solo.io



Build

Store

Deploy

Debug

Debug in Production

solo.io

Web Assembly Envoy Filter: User Experience

Simplified tooling to bootstrap Wasm modules in Rust, C++, TinyGo, AssemblyScript

Infrastructure to build, push, share, deploy, debug Wasm into Istio service mesh

Wasm Registry

Multi-cluster management, orchestration of Wasm lifecycle

solo.io

- https://solo.io
- https://solo.io/blog
- https://slack.solo.io
- https://gloo.solo.io
- https://envoyproxy.io
- https://istio.io
- https://webassemblyhub.io

solo.io