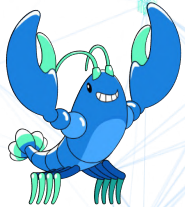
BUOYANT

Creators of LINKERD

Wasm 101 with Linkerd

Bailey Hayes, CTO at Cosmonic

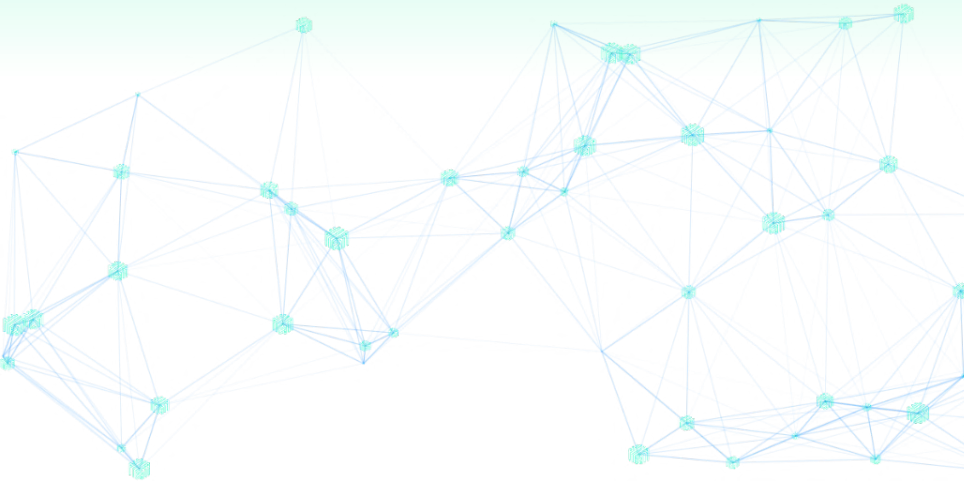
Flynn, Technical Evangelist for Linkerd



@BuoyantIO

buoyant.io

What's on the agenda?



- Intro to WebAssembly (Wasm)
- Intro to wasmCloud
- Quick intro to Linkerd
- Why you should care about this stuff 🙂
- DEMOS!
- Gotchas

BUOYANT

How do you follow along?

- https://github.com/BuoyantIO/service-mesh-academy/tree/main/wasmcloud-and-linkerd
- **For this demo, we'll use Buoyant Enterprise for Linkerd 2.18!**
  - But Linkerd edge-25.4.4 or later will work, too
- I'll be using a `k3d` cluster, but pretty much any cluster that can support LoadBalancer services should work

BUOYANT

How do you follow along?

- **kubectl**
  https://kubernetes.io/docs/tasks/tools/
- **linkerd CLI**
  https://linkerd.io/2/getting-started
- **helm**
  https://helm.sh/docs/intro/quickstart
- **bat**
  https://github.com/sharkdp/bat
- **jq**
  https://github.com/jqlang/jq

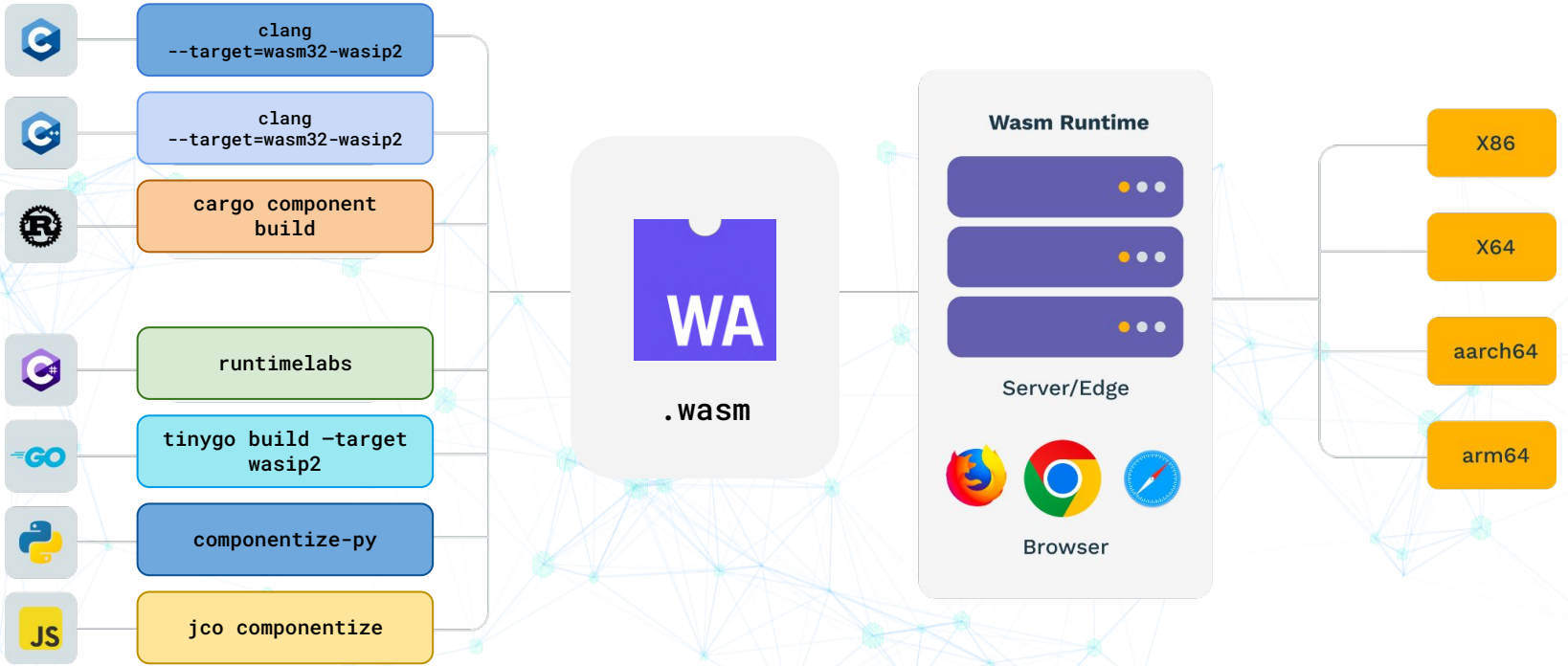
BUOYANT

BUOYANT
SERVICE MESH
ACADEMY

What is Wasm?

What is WebAssembly (Wasm)?

It's a portable compilation target supported by many languages



Why Wasm?

Portable

Portable across architectures, compile to wasm, run anywhere

Zero cold start

Starts in microseconds
Scales to zero by default

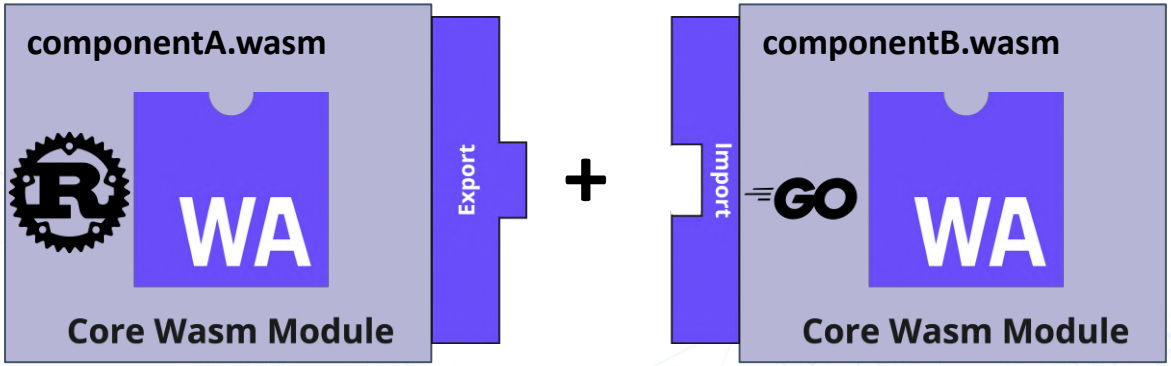
Size

Small in size, typically ranging from 100s KBs to single digit MBs

Capability-based Security model

Unlike containers where you take permissions away, only give capabilities it needs

Wasm components are composable



Language Interoperable

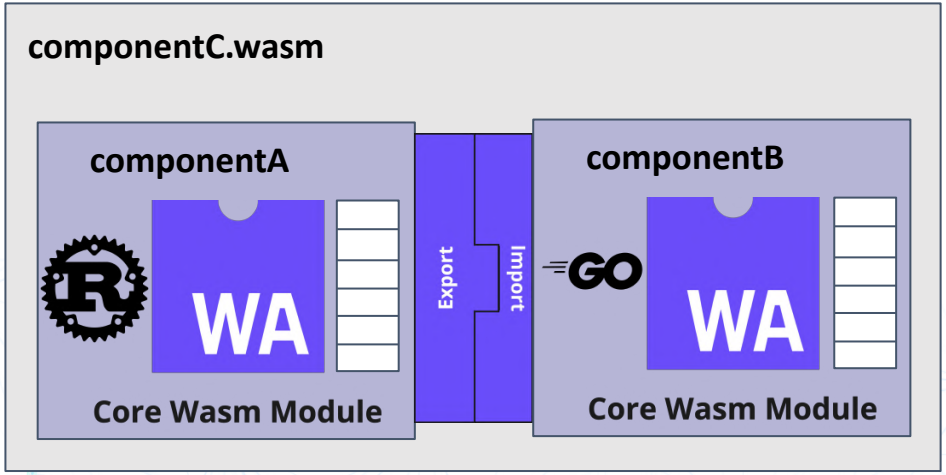
Compose components with any other language

Strictly-defined Interfaces

API Driven design defines boundaries across ecosystem

WIT

With fine-grained sandboxing



Shared-Nothing Linking

Internals including globals and memory are isolated to each component

Interface Driven

Composed components may only communicate via their exports and imports

Fast, Intra Process Execution

Isolation guaranteed via Wasm Runtime allows for cross-component calls to run in ***nanoseconds***

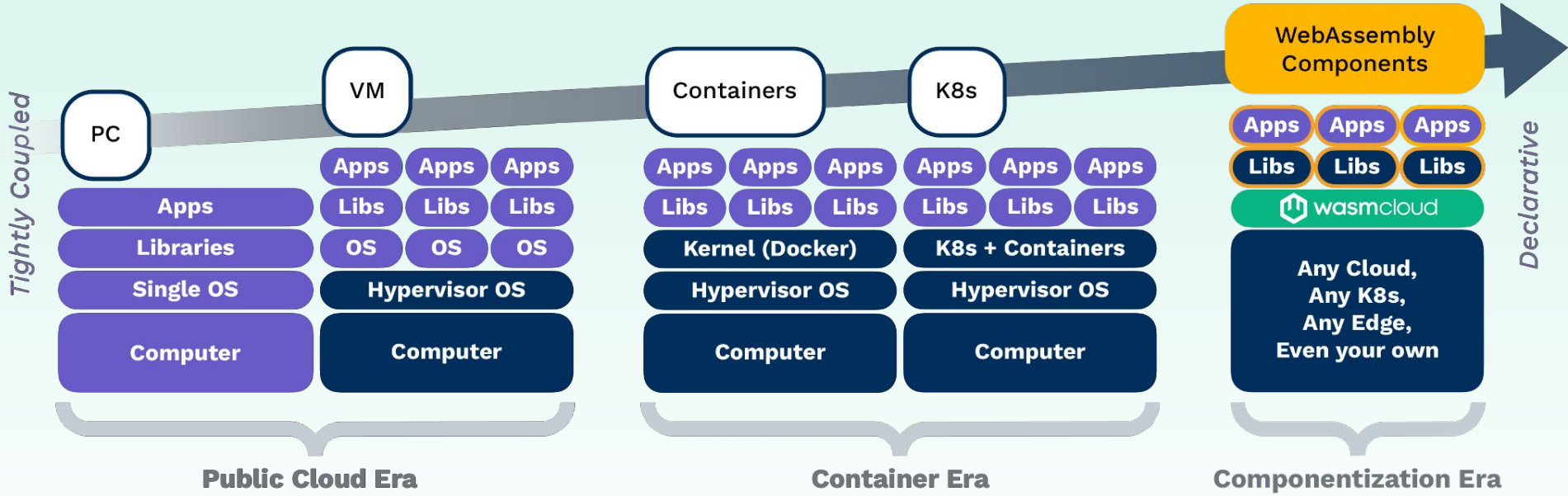
Wasm-native Orchestration

Deploy and manage Wasm applications
on any device, server or cloud

even your own!

wasmcloud

Platform Evolution



Legend: **Developer Provided** **Service Provided** **Componentized**

Problems with Containers

Default Open

Containers deploy into an often unrestricted POSIX environment

Cold Starts, High Cost of Idle Infra

Even highly optimized containers have a cold start greater than a network request - meaning to be available an app must be idle

Often Bloated, Low Density

Containers come in many shapes and sizes - from the very large to the petite

Anchored Dependencies

Portable containers get locked into a specific deployment location via dependencies

App-by-App Maintenance

5000 teams, fixing the same vulnerability, one time

Compose: Applications with Open Standards

Development Without Lock-In

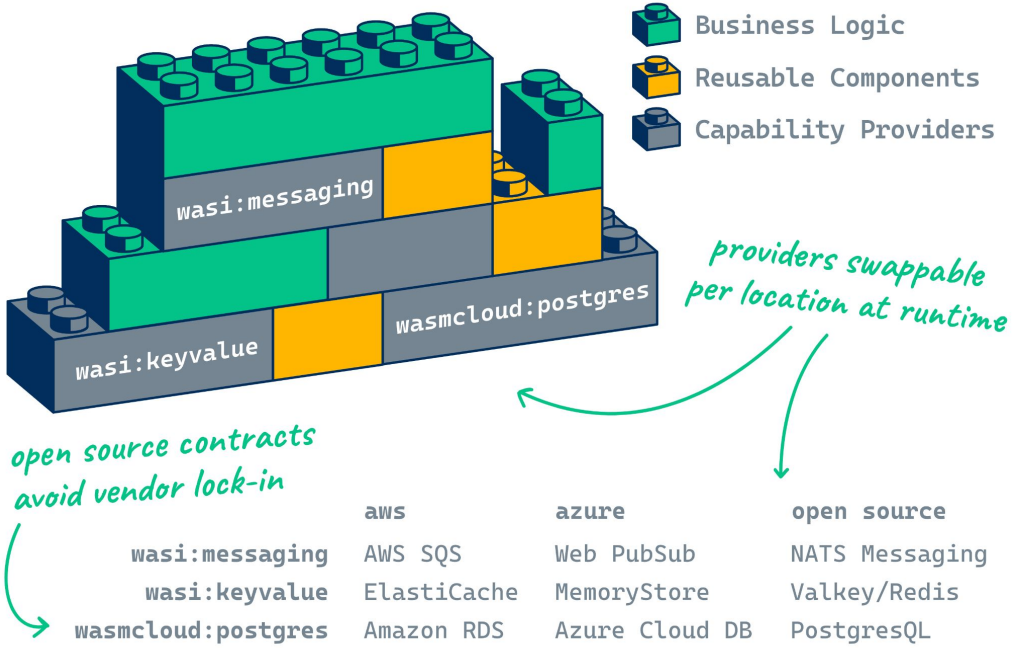
- Interface driven development
- Swap capabilities at runtime

Truly Portable Apps

- Compile once
- Run on any architecture

Custom Capabilities

- Interfaces for native hardware
- Custom-built capabilities



	aws	azure	open source
wasi:messaging	AWS SQS	Web PubSub	NATS Messaging
wasi:keyvalue	ElastiCache	MemoryStore	Valkey/Redis
wasmcloud:postgres	Amazon RDS	Azure Cloud DB	PostgresQL

cosmonic

WebAssembly ~~Systems~~ Interface ^Standard ^s (WASI)

- Started in 2019 as “WASI Snapshot Preview 1”
- Monolithic ABI
- Filesystem, I/O, Random, Clock

WASI Subgroup of WebAssembly Community Group

```
(module
    ;; Import fd_write WASI function which will write
the given io vectors to stdout
    (import "wasi_snapshot_preview1" "fd_write" (func
$fd_write (param i32 i32 i32 i32) (result i32)))
```

cosmonic

WASIP2

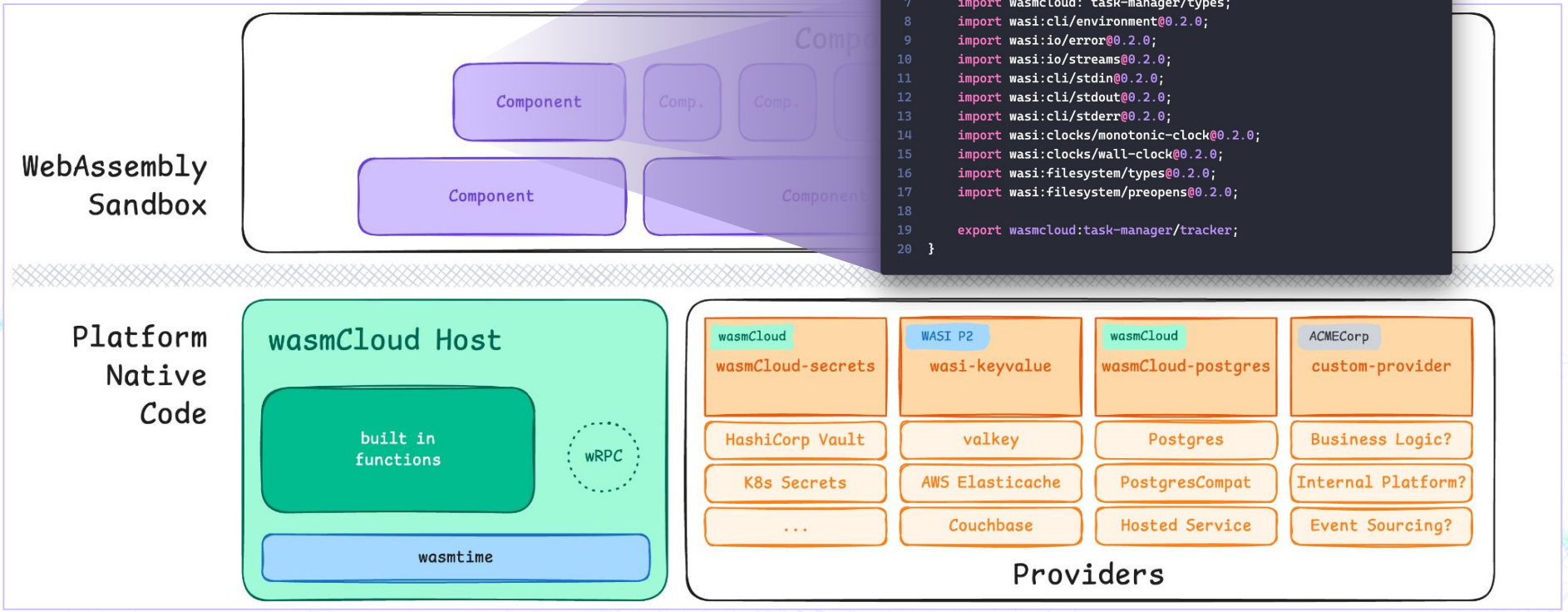
- **Released Jan 2024**
- Modular, versioned interfaces
- Non-breaking releases every two months
- Support in several languages as
  - **`wasm32-wasip2`**
- New Networking Interfaces

API	Repository
Clocks	https://github.com/WebAssembly/wasi-clocks
Random	https://github.com/WebAssembly/wasi-random
Filesystem	https://github.com/WebAssembly/wasi-filesystem
Sockets	https://github.com/WebAssembly/wasi-sockets
CLI	https://github.com/WebAssembly/wasi-cli
HTTP	https://github.com/WebAssembly/wasi-http

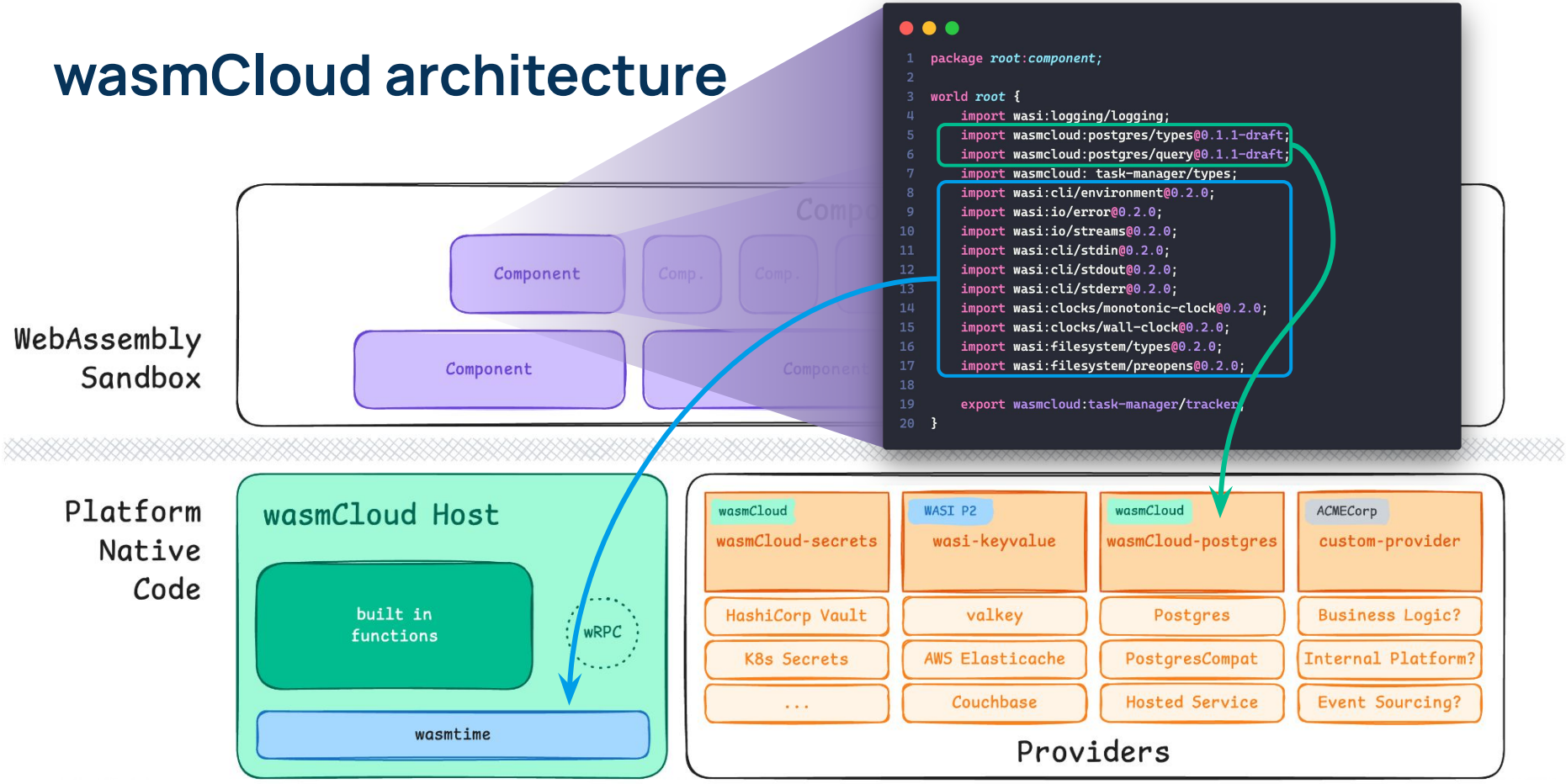
```
(component
  ;; note version and types
  (import "wasi:filesystem/types@0.2.0" "[method]descriptor.write"
    (func $wasi/v0.2.0/types.wasmimport_DescriptorWrite (;29;) (type 29)))
```

wasmCloud architecture

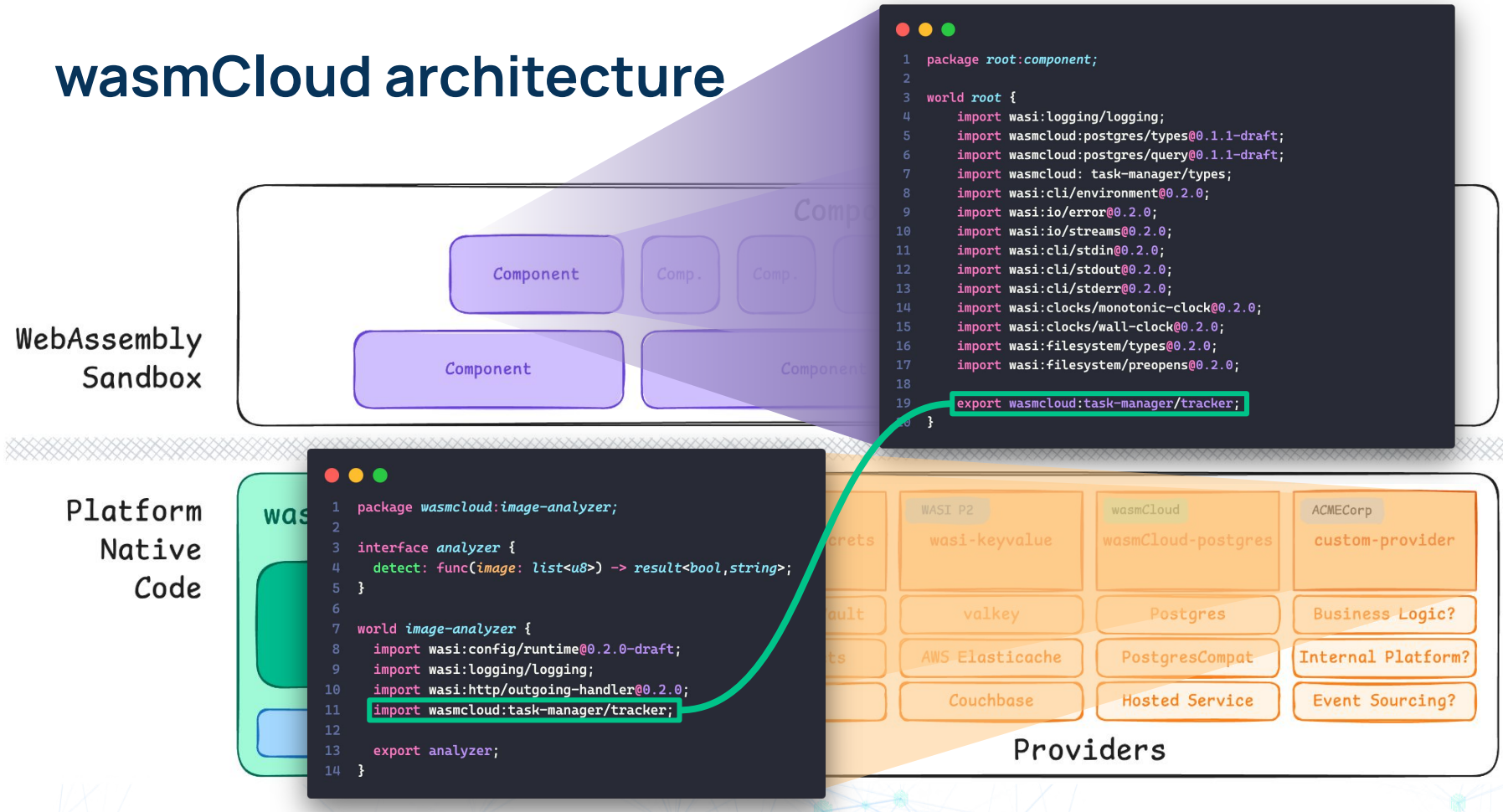
```
package root:component;

world root {
    import wasi:logging/logging;
    import wasmcloud:postgres/types@0.1.1-draft;
    import wasmcloud:postgres/query@0.1.1-draft;
    import wasmcloud: task-manager/types;
    import wasi:cli/environment@0.2.0;
    import wasi:io/error@0.2.0;
    import wasi:io/streams@0.2.0;
    import wasi:cli/stdin@0.2.0;
    import wasi:cli/stdout@0.2.0;
    import wasi:cli/stderr@0.2.0;
    import wasi:clocks/monotonic-clock@0.2.0;
    import wasi:clocks/wall-clock@0.2.0;
    import wasi:filesystem/types@0.2.0;
    import wasi:filesystem/preopens@0.2.0;

    export wasmcloud:task-manager/tracker;
}
```



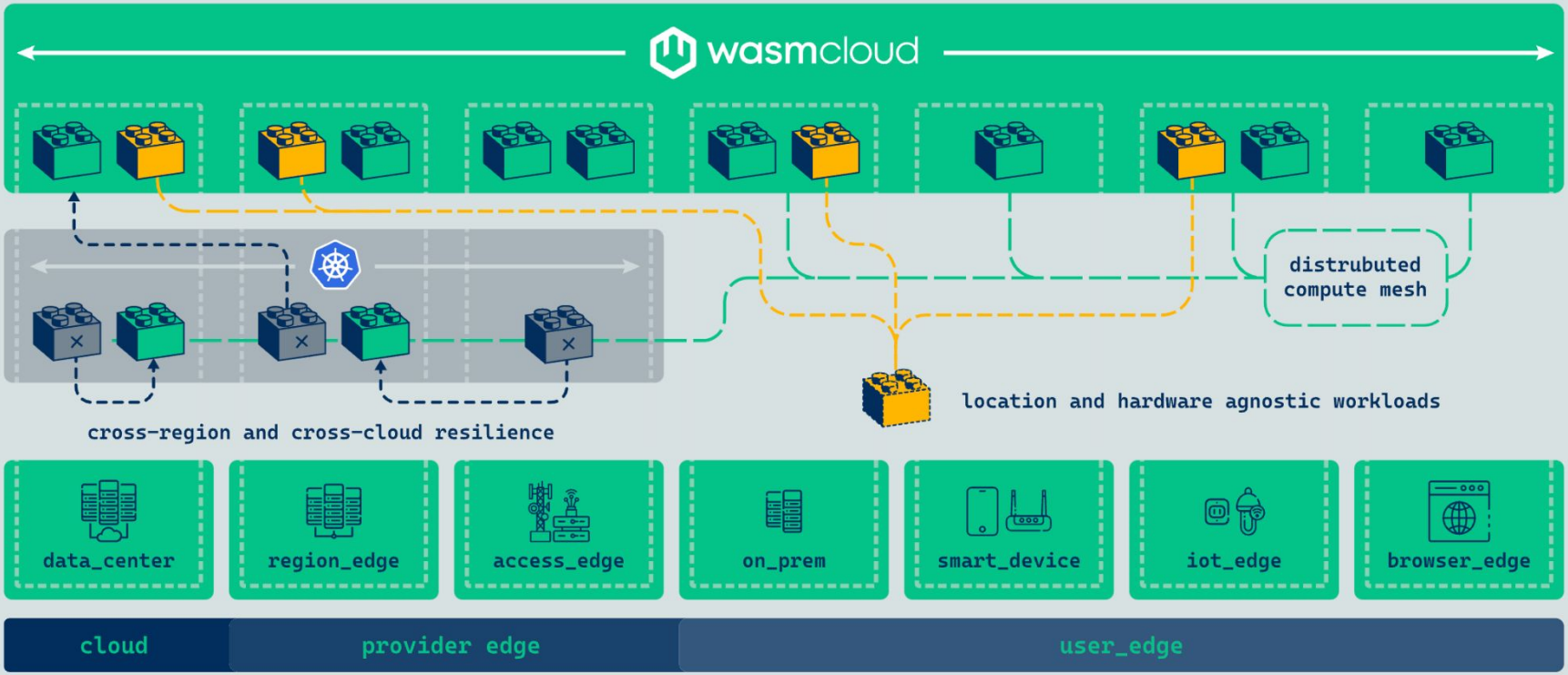
wasmCloud architecture



wasmCloud architecture



wasmcloud


wasmcloud
distrubuted
compute mesh
cross-region and cross-cloud resilience
location and hardware agnostic workloads
data_center
region_edge
access_edge
on_prem
smart_device
iot_edge
browser_edge
cloud
provider edge
user_edge

wasmCloud Operator

- Declarative wasmCloud management via Kubernetes CRDs
  - Wadm! Application Manifests
  - wasmCloud Host Groups
- Service Endpoint integration
- Secrets integration

`wasmCloud/wasmcloud-operator`

Host Groups

Sets up wasmCloud instances for Components & Capabilities hosting

- **WasmCloudHostConfig** CRD
  - Encodes best practices. Ex: NATS leaf
- Managed Kubernetes Deployment Lifecycle
  - Configuration updates
  - Status reporting
- Integrates with manifest validators (kubeconform)

```yaml
apiVersion: k8s.wasmcloud.dev/v1alpha1
kind: WasmCloudHostConfig
metadata:
  name: general
spec:
  lattice: default
  version: "1.0.4"
  hostReplicas: 10
  hostLabels:
    cluster: us-east
    costcenter: engineering
```

Application Manifest

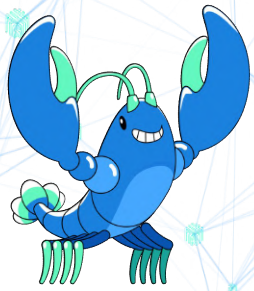
- Based on the Open Application Model specification from oam.dev
- Supports describing the components, providers, links, and configuration that make up an application

```
kubectl apply -f ./wadm.yaml
```

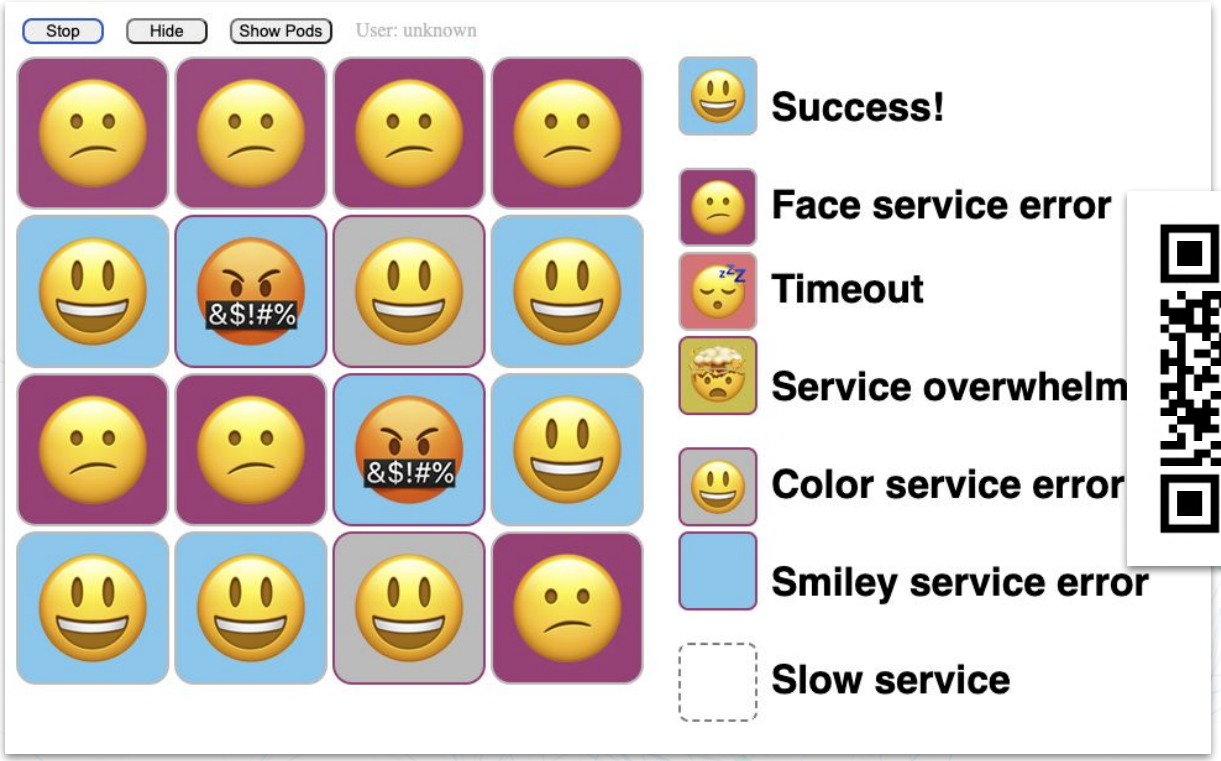

```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: rust-http-hello-world
  annotations:
    version: v0.0.7
    description: "HTTP hello world demo in Rust"
spec:
  components:
    - name: http-hello-world
      type: component
      properties:
        image: wasmcloud.azurecr.io/http-hello-world:0.1.0
        id: helloworld
      traits:
        # Govern the spread/scheduling of the actor
        - type: spreadscaler
          properties:
            replicas: 5000
    # Add a capability provider that mediates HTTP access
    - name: httpserver
      type: capability
      properties:
        image: ghcr.io/wasmcloud/http-server:0.20.0
        id: httpserver
      traits:
        # Link the httpserver with the component above
        - type: link
          properties:
            target: http-hello-world
            namespace: wasi
            package: http
            interfaces: [incoming-handler]
            source_config:
            - name: default-http
              properties:
                address: 0.0.0.0:8080
        - type: daemonscaler
          properties:
            replicas: 1
```

BUOYANT
SERVICE MESH
ACADEMY

DEMO 🤞

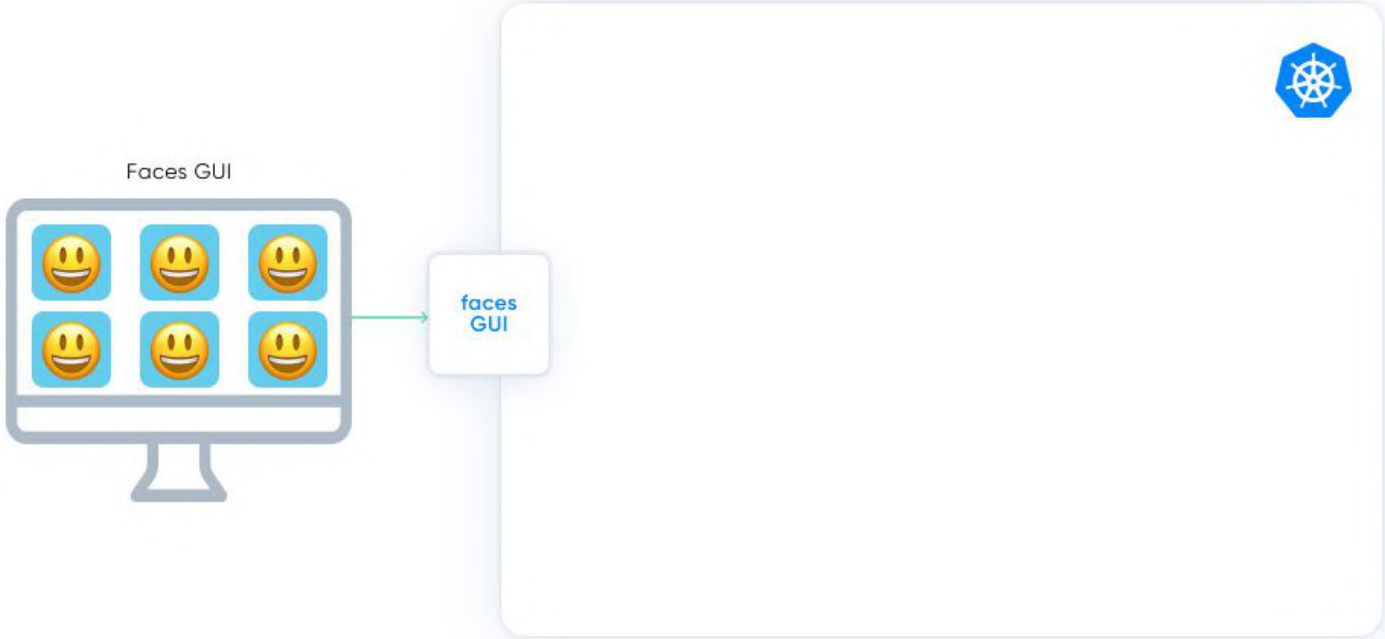


Faces (http://github.com/BuoyantIO/faces-demo)

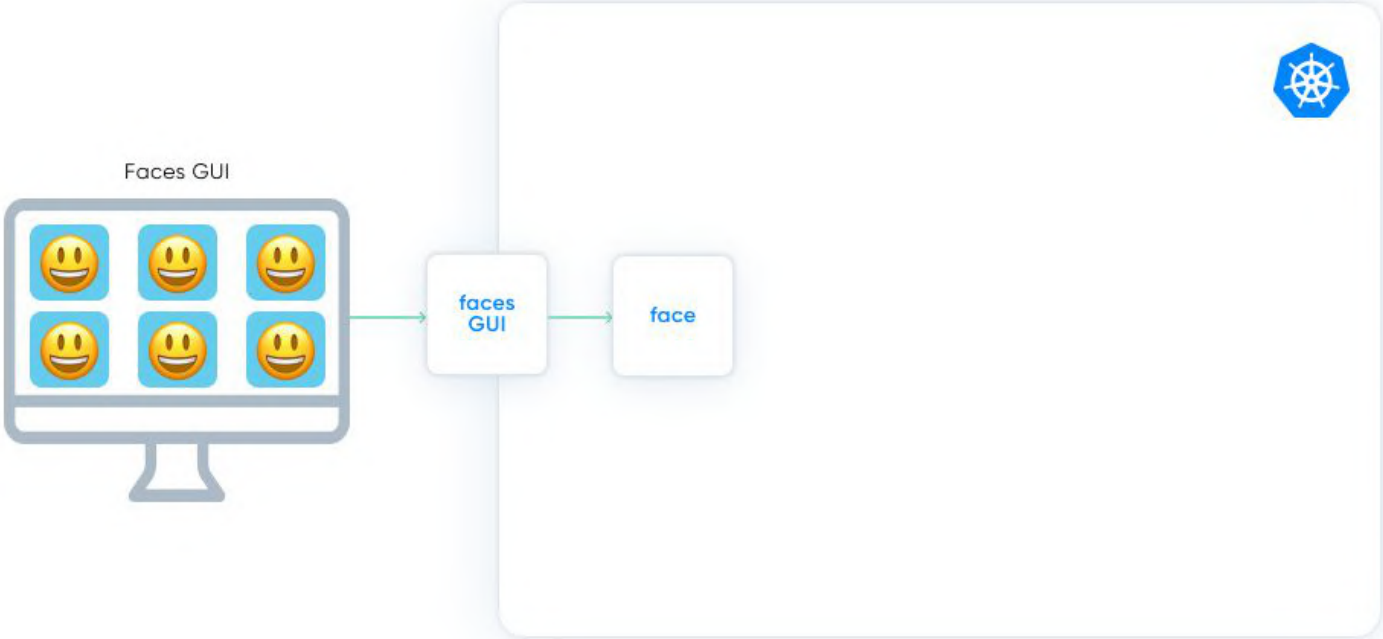


Faces (http://github.com/BuoyantIO/faces-demo)

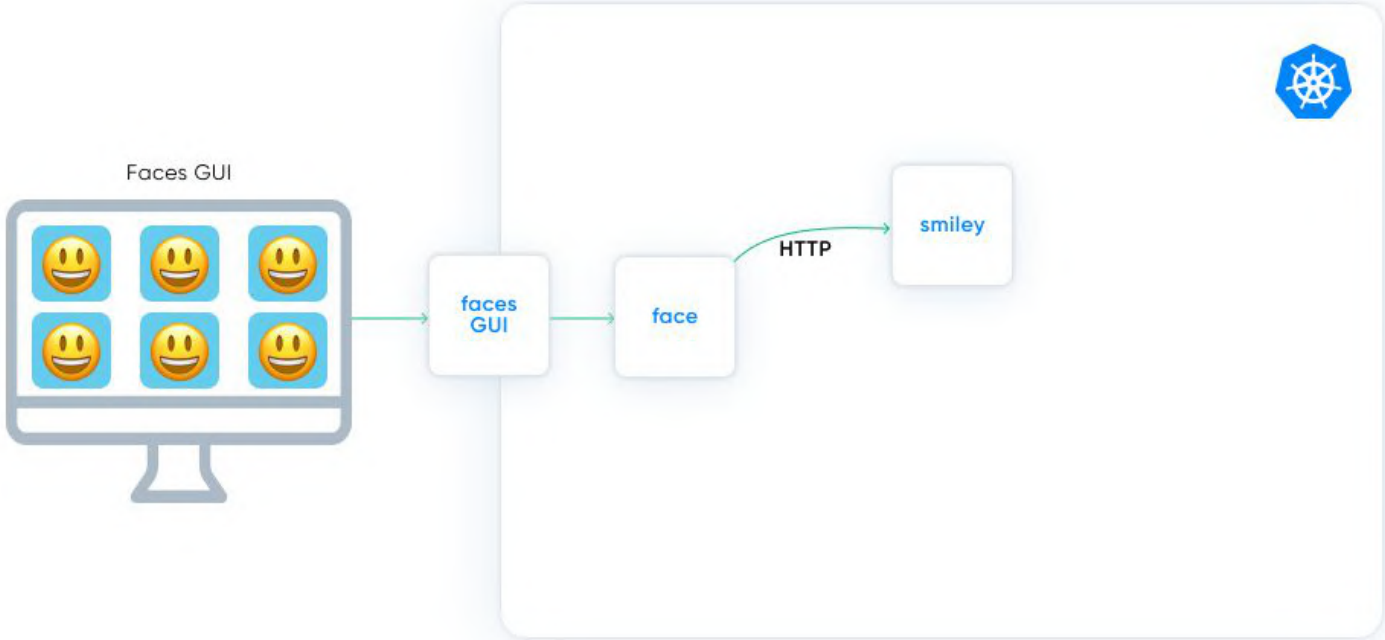
Faces (http://github.com/BuoyantIO/faces-demo)



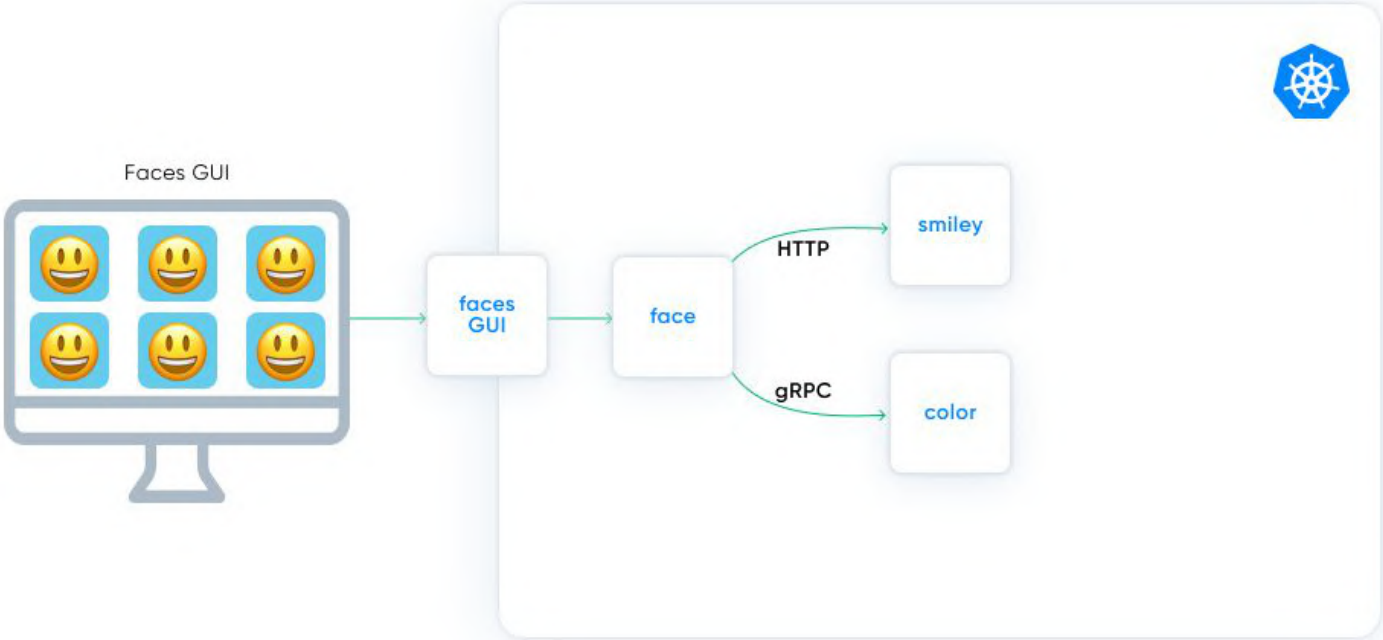
Faces (http://github.com/BuoyantIO/faces-demo)



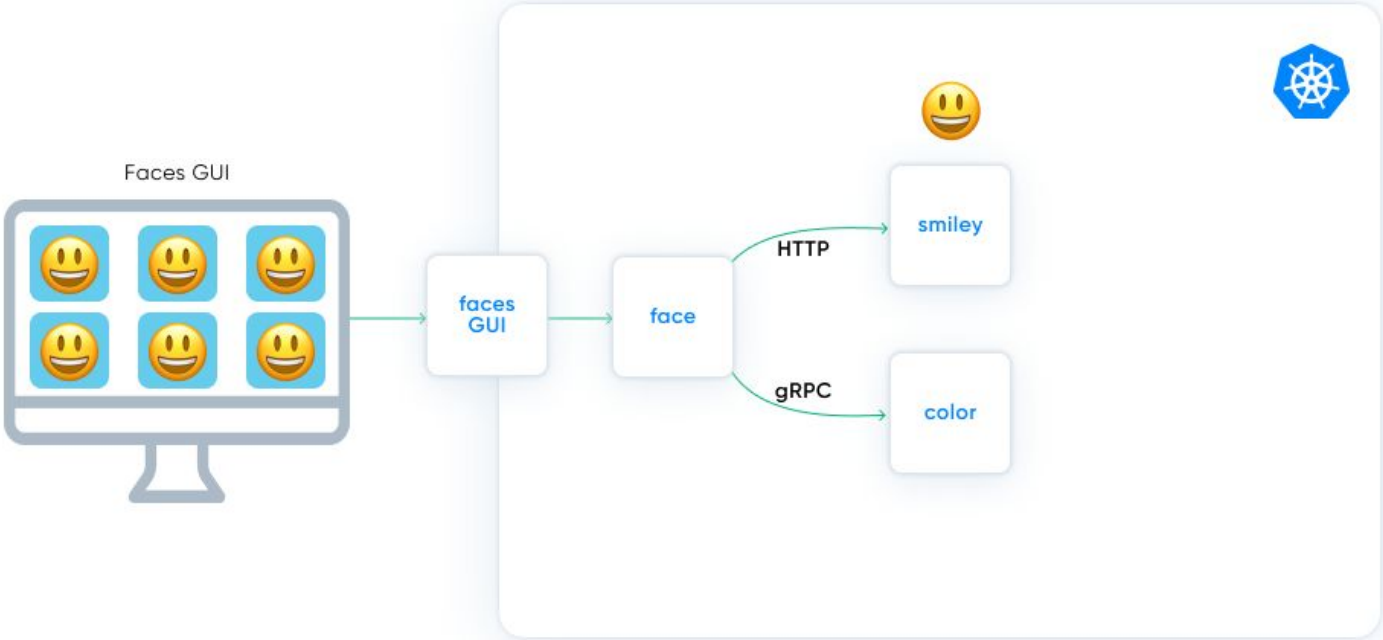
Faces (http://github.com/BuoyantIO/faces-demo)



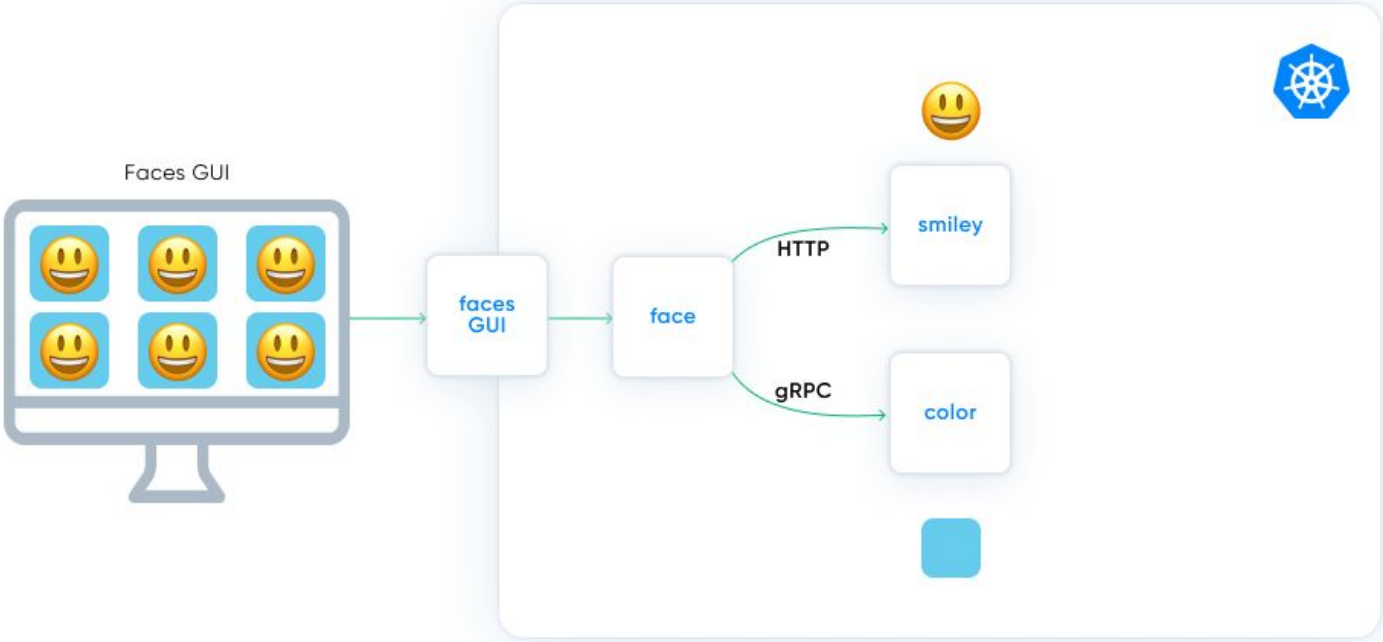
Faces (http://github.com/BuoyantIO/faces-demo)



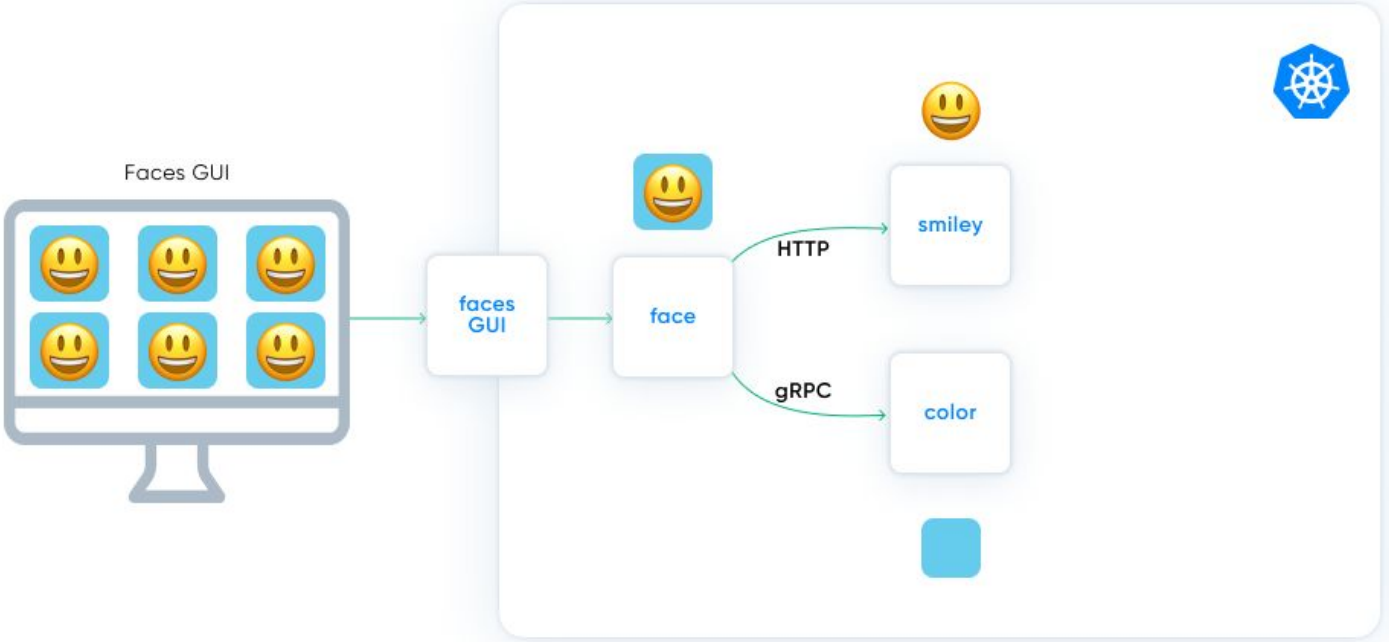
Faces (http://github.com/BuoyantIO/faces-demo)



Faces (http://github.com/BuoyantIO/faces-demo)

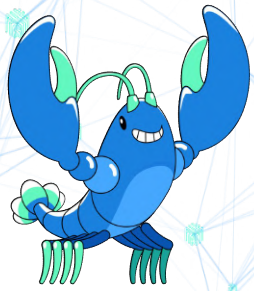


Faces (http://github.com/BuoyantIO/faces-demo)



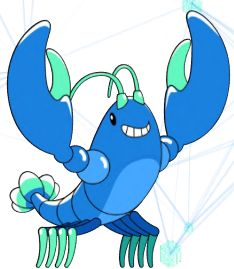
BUOYANT
SERVICE MESH
ACADEMY

DEMO 🤞



BUOYANT
SERVICE MESH
ACADEMY

What is Linkerd?



What is Linkerd?

Linkerd is a **service mesh**.

***service mesh**, n:*

- An infrastructure layer providing security, reliability, and observability at the platform level, uniformly, across an entire application.

What is Linkerd?

Linkerd is a **service mesh**.

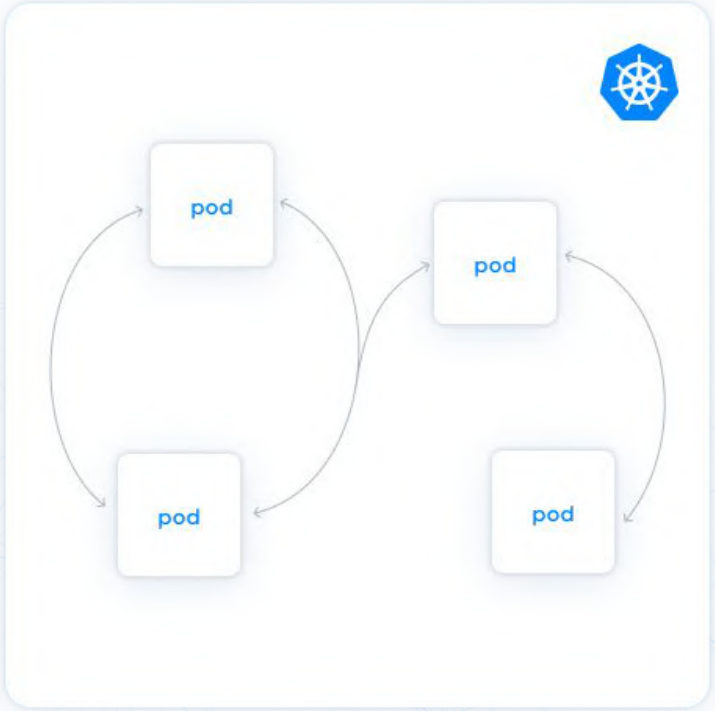
***service mesh**, n:*

- An infrastructure layer providing security, reliability, and observability at the platform level, uniformly, across an entire application.

The Microservices Architecture

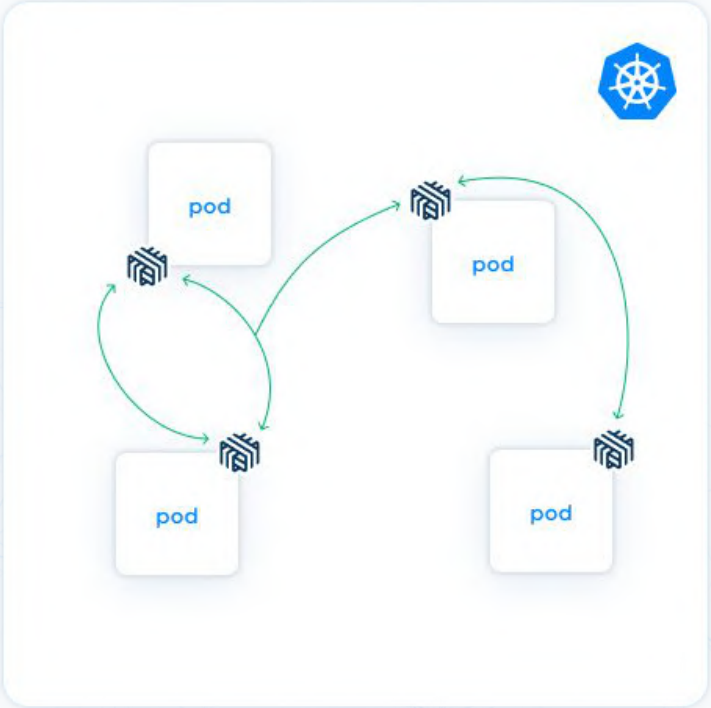
- Microservices communicate over an insecure, unreliable network.
- These are *fundamental characteristics* of the way real networking is built; they cannot be changed.

- Service meshes like Linkerd exist to make this situation better.



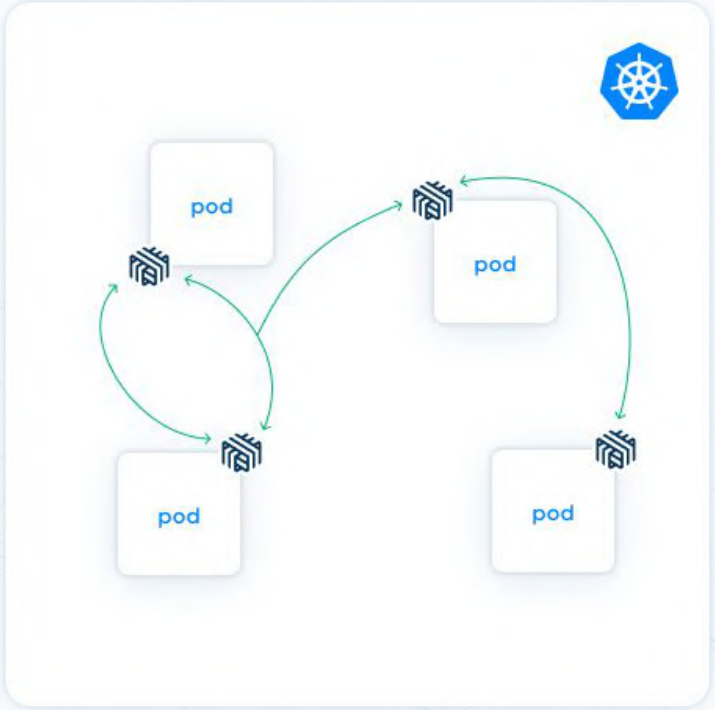
Microservices and the Mesh

- Like most other meshes, Linkerd works by adding a proxy (a *sidecar)* next to each application pod.
- *Unlike* any other mesh, Linkerd uses a purpose-built, lightweight, ultrafast Rust microproxy.
- These microproxies mediate and measure all communications in the mesh, which allows for all the mesh's functionality.



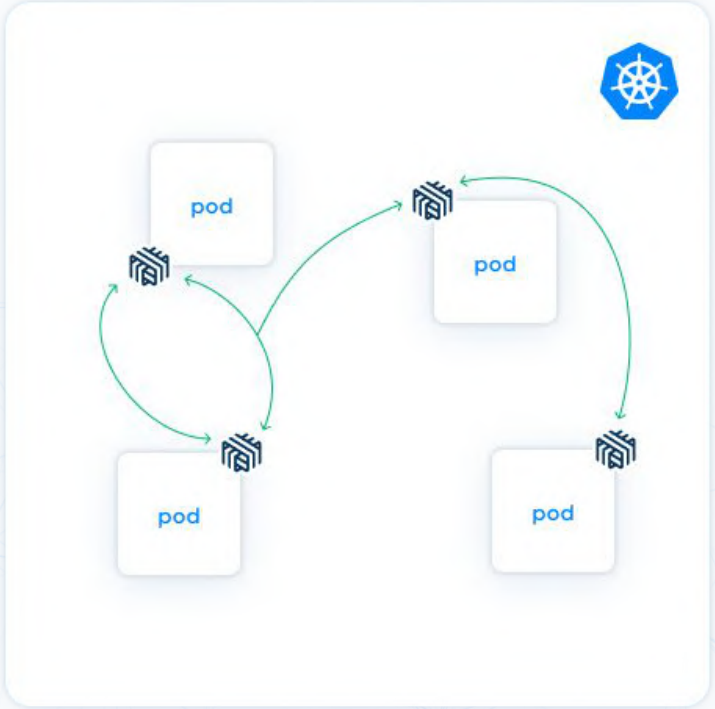
Microservices and the Mesh

- Mediating communications lets Linkerd enforce rules and add capabilities:
  - mTLS
  - advanced load balancing
  - multicluster communication
  - retries, timeouts, etc.



Microservices and the Mesh

- Measuring communications lets Linkerd provide observability:
  - discover and display the actual application call graph
  - measure and publish the golden metrics (request rate, success rate, latency)

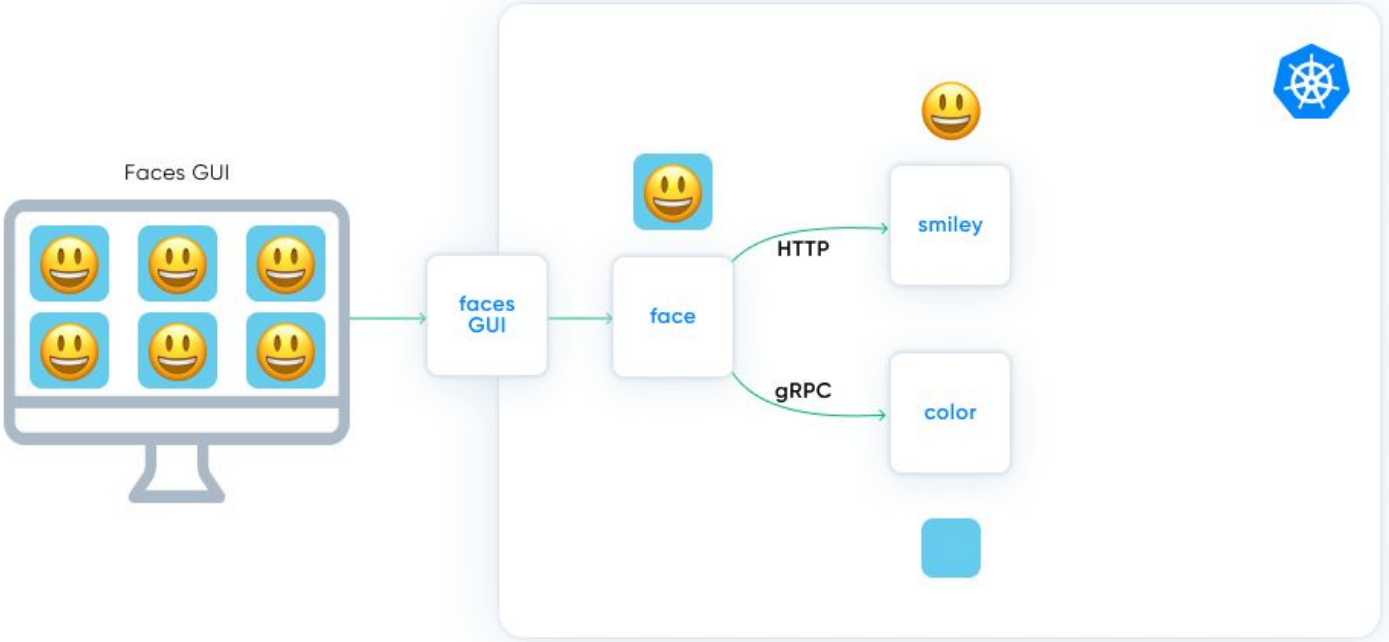


Why is this important?

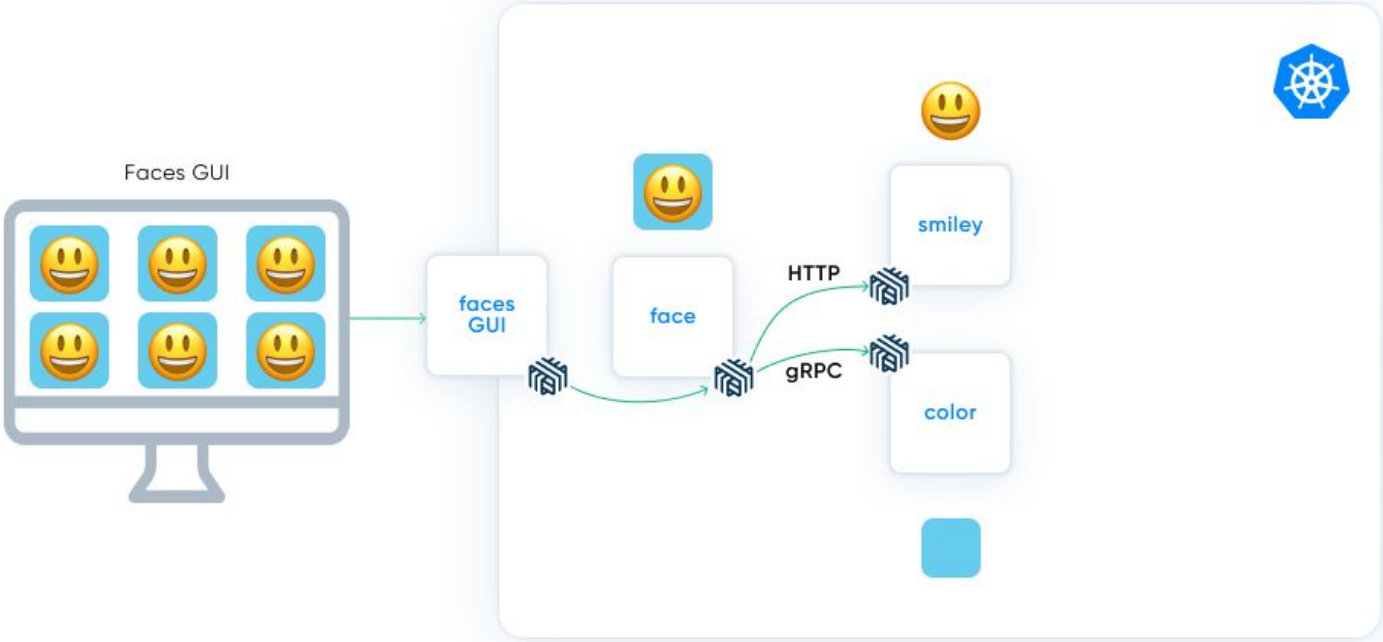
Security, reliability, and observability are not optional.

- You can get them from a mesh.
- You can get them by writing a lot of application code.
- You can't do without them.

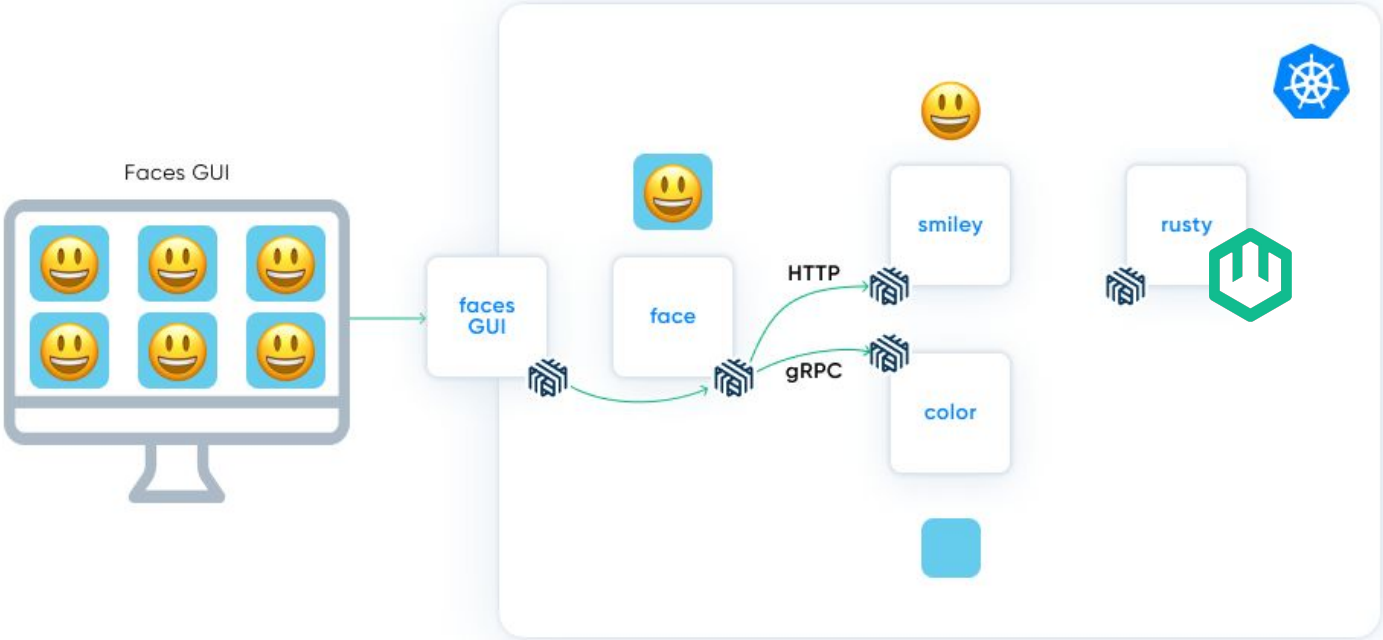
Faces (http://github.com/BuoyantIO/faces-demo)



Faces (http://github.com/BuoyantIO/faces-demo)

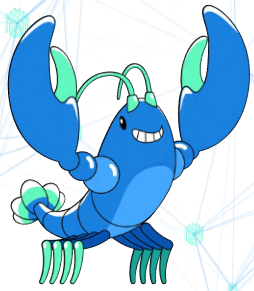


Faces (http://github.com/BuoyantIO/faces-demo)



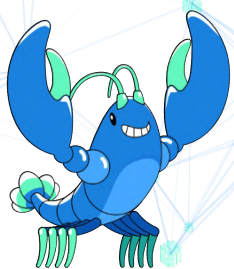
BUOYANT
SERVICE MESH
ACADEMY

DEMO 🤞



BUOYANT
SERVICE MESH
ACADEMY

Gotchas



Gotchas

- The big one: NATS must be marked `opaque`!
  - The good news is that this is *not* a subtle failure if you get it wrong.
- The other one: right now, all the Wasm components in a wasmCloud host share a Linkerd identity.
  - We're working on this. 🙂
- Developing Wasm components in Go is a little rough because `wasip2` isn't yet in the standard library!
  - Working on this too! 🙂
- Wasm apps are *sandboxed*.
  - This is generally a good thing! but you need to understand what the constraints of the sandbox.

Tell us how we can improve!

Your feedback matters!

(We promise it won't take more than a few minutes, and it will help us tremendously — thank you! 🙂)

BUOYANT
Creators of LINKERD

Buoyant Enterprise for LINKERD

Rust-based network security and reliability for modern applications. Built on open source and designed for the enterprise.

- ➔ Zero-trust security and compliance across your entire network
- ➔ Global traffic management and control
- ➔ Full L7 application observability
- ➔ Built for the enterprise

Learn more & try it for free at buoyant.io/**enterprise-linkerd**

BUOYANT
Creators of LINKERD

Updated Courses!

BUOYANT
Creators of LINKERD

Get Certified!

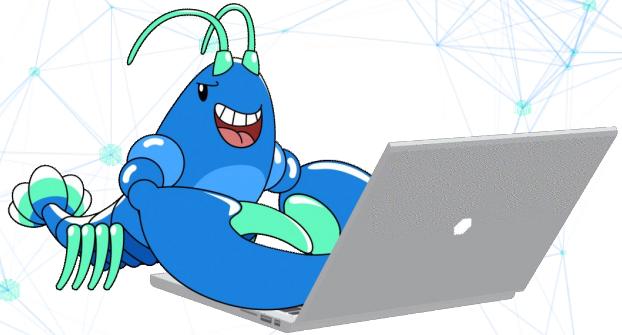
With hands-on LINKERD self-paced courses

- Service Mesh 101
- Linkerd in Production

SERVICE MESH ACADEMY

Up Next on July 17

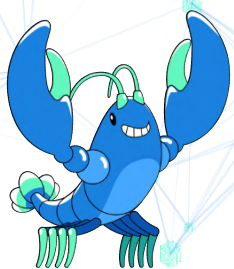
Anti-Complex Multicluster: Federated Services



SIGN UP TODAY!
buoyant.io/sma

BUOYANT
SERVICE MESH
ACADEMY

Q&A



BUOYANT
SERVICE MESH
ACADEMY

Thanks much!

flynn@buoyant.io
@flynn on slack.linkerd.io