Multi-Tier Security in WasmCloud:

From Developer Constraints to Platform Extensibility

Brooks Townsend

cosmonic

cosmonic

Introduction

- Senior Software Engineer @ Cosmonic
- CNCF wasmCloud maintainer
- Rustacean 🦀
- Demo enthusiast
- Developer Developer Developer

cosmonic

Agenda

- Open Source Security for Platforms
- wasmCloud Introduction
- Basic Demo
- wasmCloud Multi-tiered Security Platform
- Deep Dive Demo
- wasmCloud Secrets & Extensibility
- Secrets Demo

cosmonic

“74% of codebases contained high risk open source vulnerabilities.”

2024 "Open Source Security and Risk Analysis" (OSSRA) report
https://investor.synopsys.com/news/news-details/2024/New-Synopsys-Report-Finds-74-of-Codebases-Contained-High-Risk-Open-Source-Vulnerabilities-Surging-54-Since-Last-Year/default.aspx

cosmonic

Platform Eng. today has a problem

1 **Build**

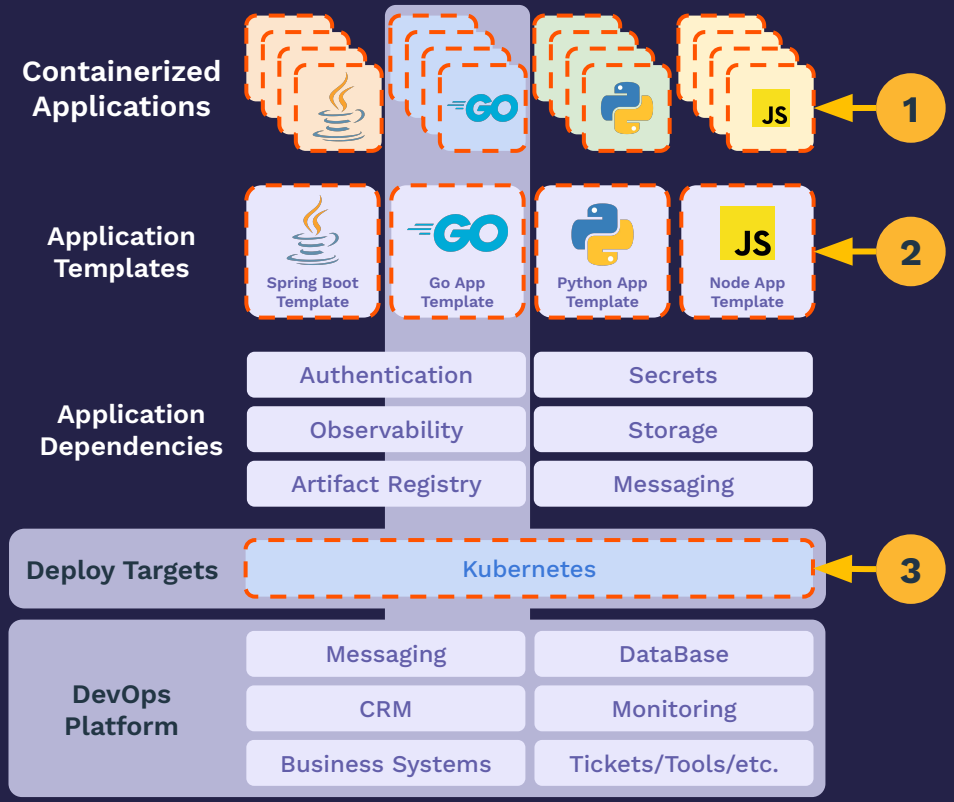
Developers tightly couple application code with OSS libraries, vulnerabilities, & boilerplate.

2 **Compose**

Enterprises integrate apps into ***blessed, required platforms***.

3 **Run**

Platform Engineers operate apps across many unique control planes.



cosmonic

Strategy: Reduce Per App (team) Responsibilities

Containers

Per Application (x1000)

- App Code
- Maintain Dependencies
- Manage Capabilities
- Run Ops

Developer Experience Gap

Per Platform (x1)

- Compute
- Storage
- Observability
- Networking
- Orchestration
- Secrets

WasmCloud

Per Application (x1000)

- App Code

Per Platform (x1)

- Maintain Dependencies
- Compute
- Storage
- Observability
- Manage Capabilities
- Run Ops
- Networking
- Orchestration
- Secrets

Wasm-native Orchestration

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

even your own!

wasmcloud

Build: Components | Wasm-Native Application Platform

Faster Development Cycles

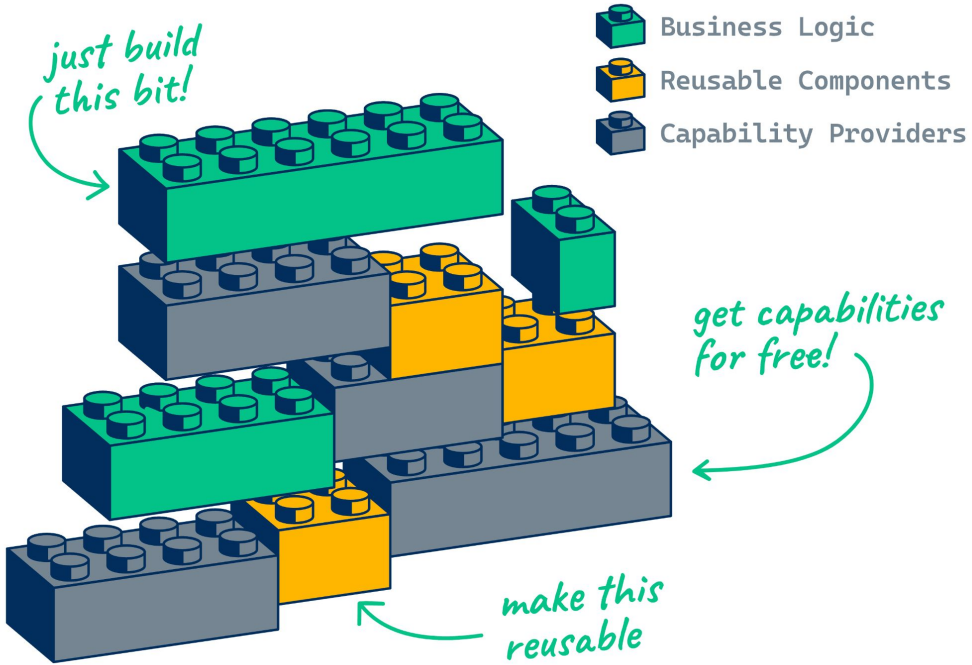
- Make code reuse simple
- Write less boilerplate

Maintainable Apps

- Simple dependency updates

Integrate with Existing Stacks

- Wasm-native works with cloud native



wasmcloud

Compose: Applications with Open Standards

Development Without Lock-In

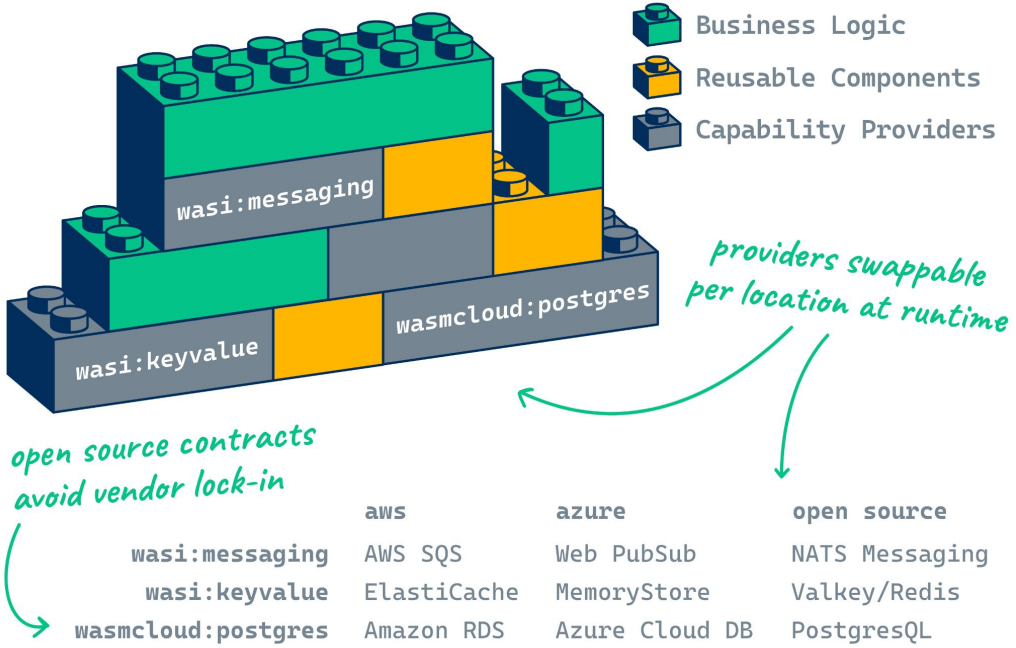
- Interface driven development
- Swap capabilities at runtime

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

wasmcloud

Run: Wasm apps on any device, server or cloud

Scale-to-Zero with Zero Cold Starts

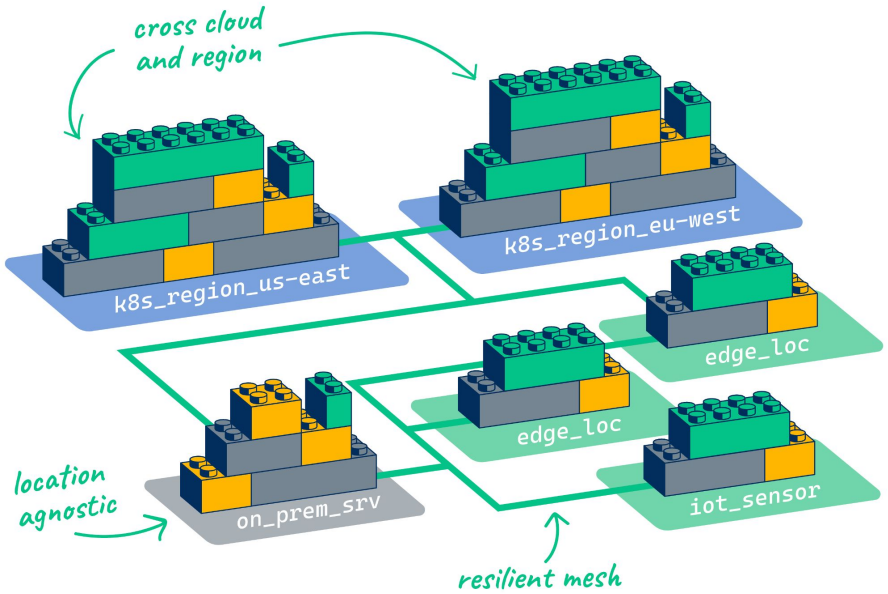
- Sub-millisecond start times
- Vertical auto scaling

Reliable, Fault-Tolerant Apps

- Horizontal scaling with failover
- Capability-level resiliency, reliability, and scalability

Deploy Across Clouds

- Local-first routing
- Cross region, cloud, and edge resiliency



cosmonic

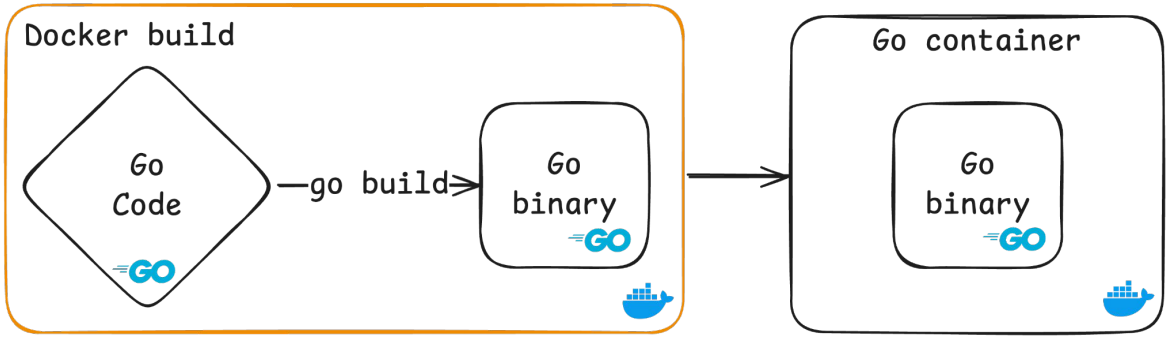
Demo time

Build and run a Wasm component, introspect for runtime behavior

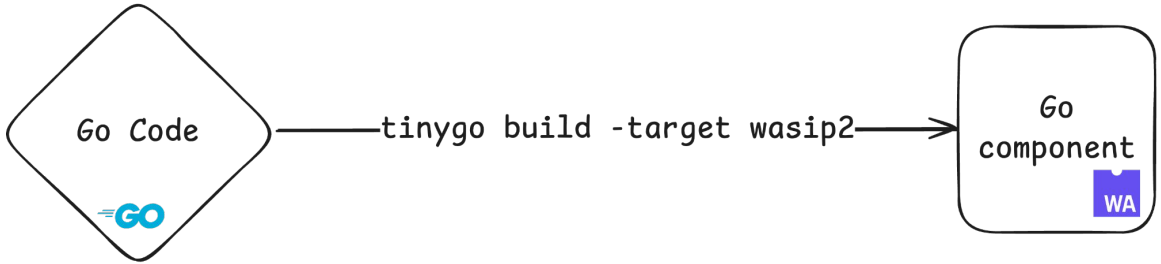
cosmonic

Build

Containerized

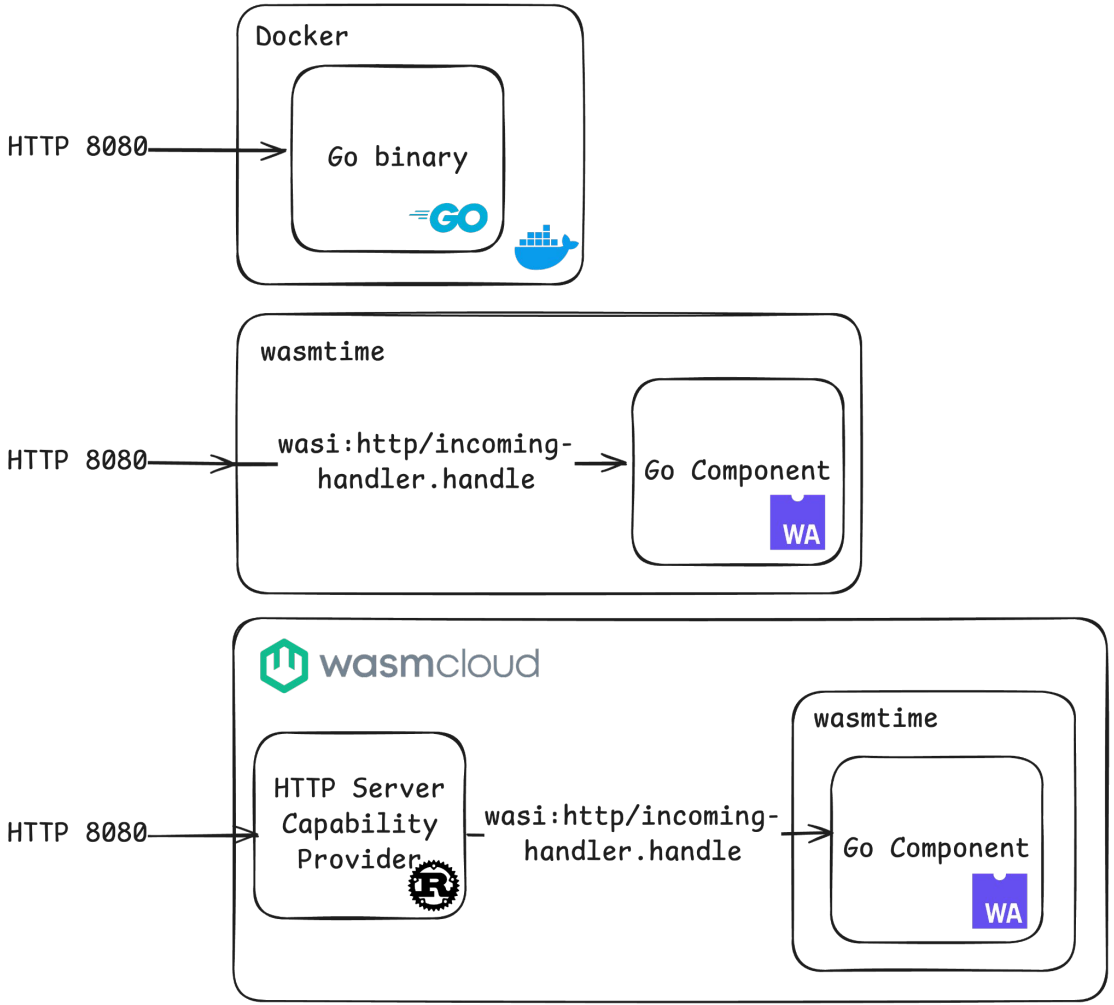


Componentized



cosmonic

Run



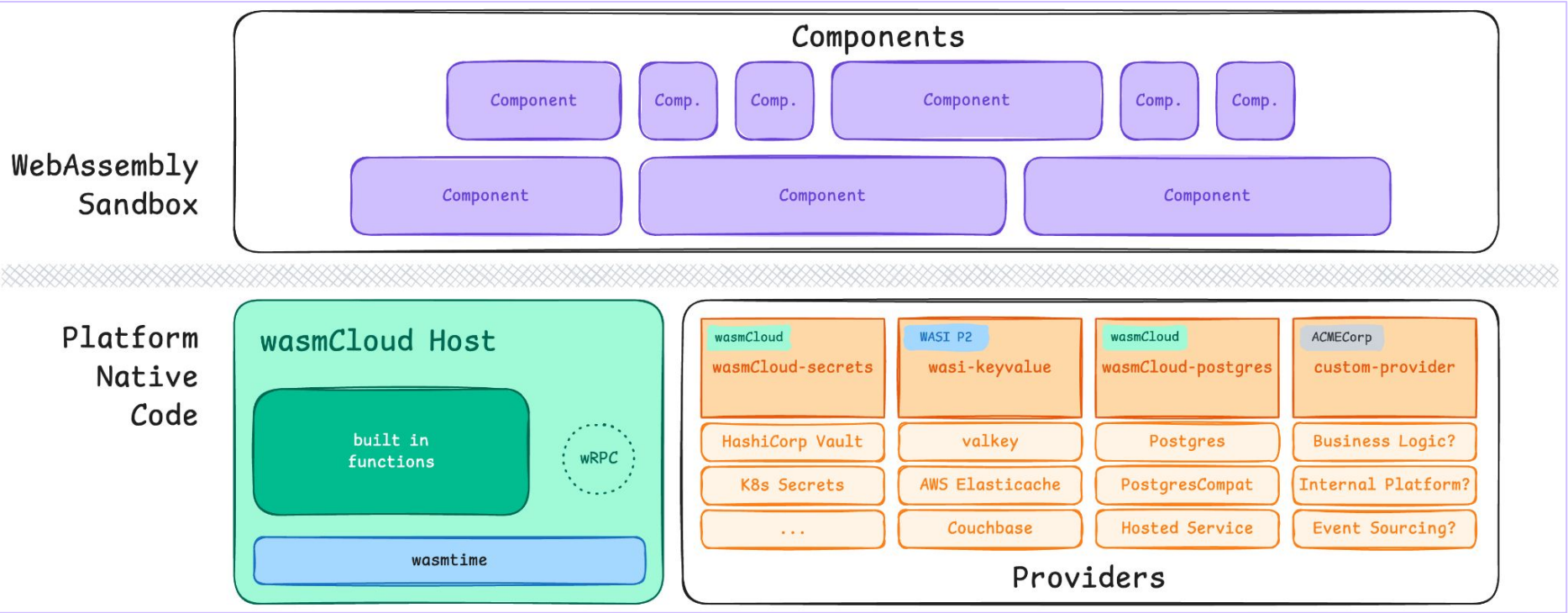
cosmonic

Demo Takeaways

- Wasm is a compile target
- Wasm can be interpreted for runtime behavior at build time
- Capability driven permissions create a deny-by-default sandbox execution environment

cosmonic

Extensibility of the wasmCloud platform



cosmonic

Extensibility of the wasmCloud

```
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;
}
```

WebAssembly Sandbox

Component

Comp.

Comp.

Component

Component

Platform Native Code

wasmCloud Host

built in functions

wRPC

wasmtime

wasmCloud	WASI P2	wasmCloud	ACMECorp
wasmCloud-secrets	wasi-keyvalue	wasmCloud-postgres	custom-provider
HashiCorp Vault	valkey	Postgres	Business Logic?
K8s Secrets	AWS Elasticache	PostgresCompat	Internal Platform?
...	Couchbase	Hosted Service	Event Sourcing?

Providers

cosmonic

Extensibility of the wasmCloud

```
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;
}
```

WebAssembly Sandbox

Component

Comp.

Comp.

Component

Component

Platform Native Code

wasmCloud Host

built in functions

WRPC

wasmtime

wasmCloud	WASI P2	wasmCloud	ACMECorp
wasmCloud-secrets	wasi-keyvalue	wasmCloud-postgres	custom-provider
HashiCorp Vault	valkey	Postgres	Business Logic?
K8s Secrets	AWS Elasticache	PostgresCompat	Internal Platform?
...	Couchbase	Hosted Service	Event Sourcing?

Providers

cosmonic

Demo time

Platform engineers provide capabilities with wasmCloud host plugins, developers don’t know nor care.

cosmonic

Demo takeaways

- Capability driven permissions create a deny-by-default sandbox execution environment
- CNCF wasmCloud as a platform lets developers write features, and platform engineers to provide capabilities at runtime.

cosmonic

Platform Security is like an Onion

Onion

?

Platform Security

?

cosmonic

Platform Security is like an Onion

Onion

They stink

Platform Security

Makes you want to cry

cosmonic

Platform Security is like an Onion

Onion

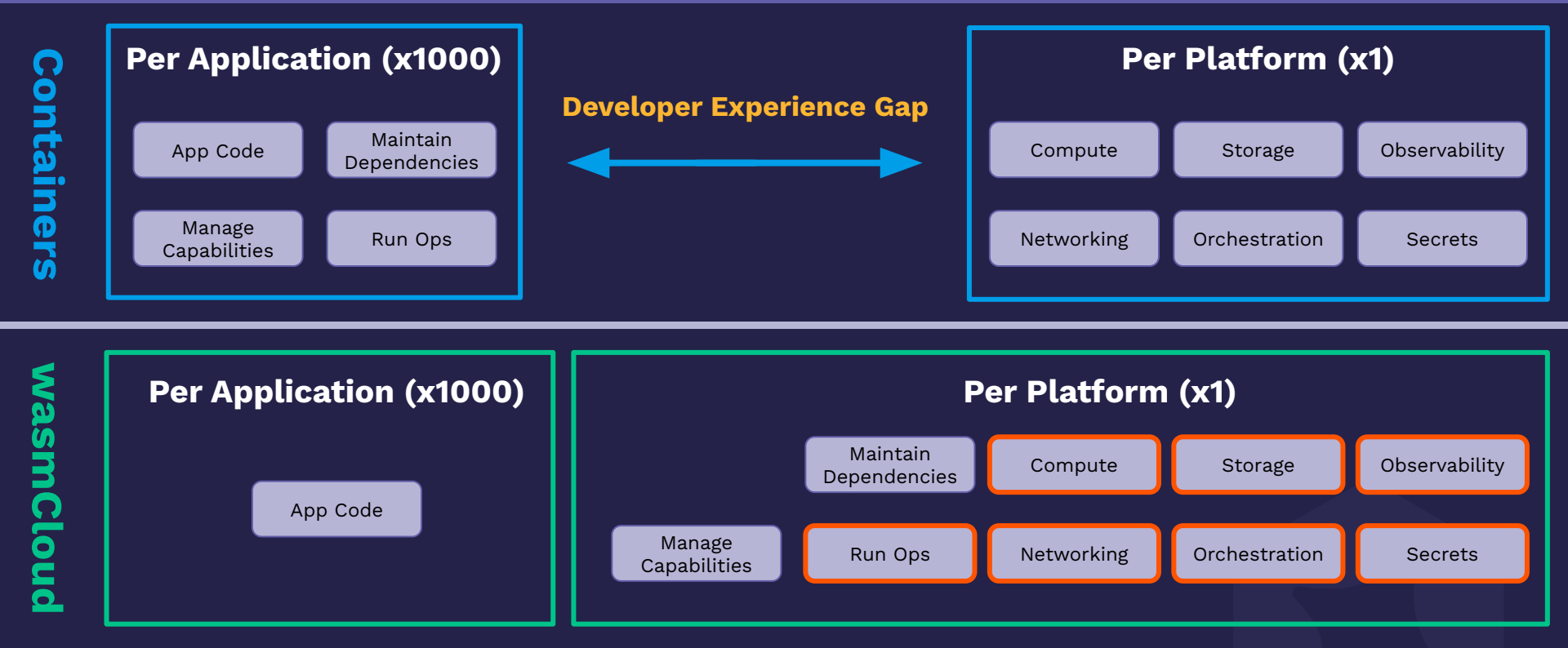
There’s a lot of layers to it

Platform Security

There’s a lot of layers to it

cosmonic

Strategy: Reduce Per App (team) Responsibilities



cosmonic

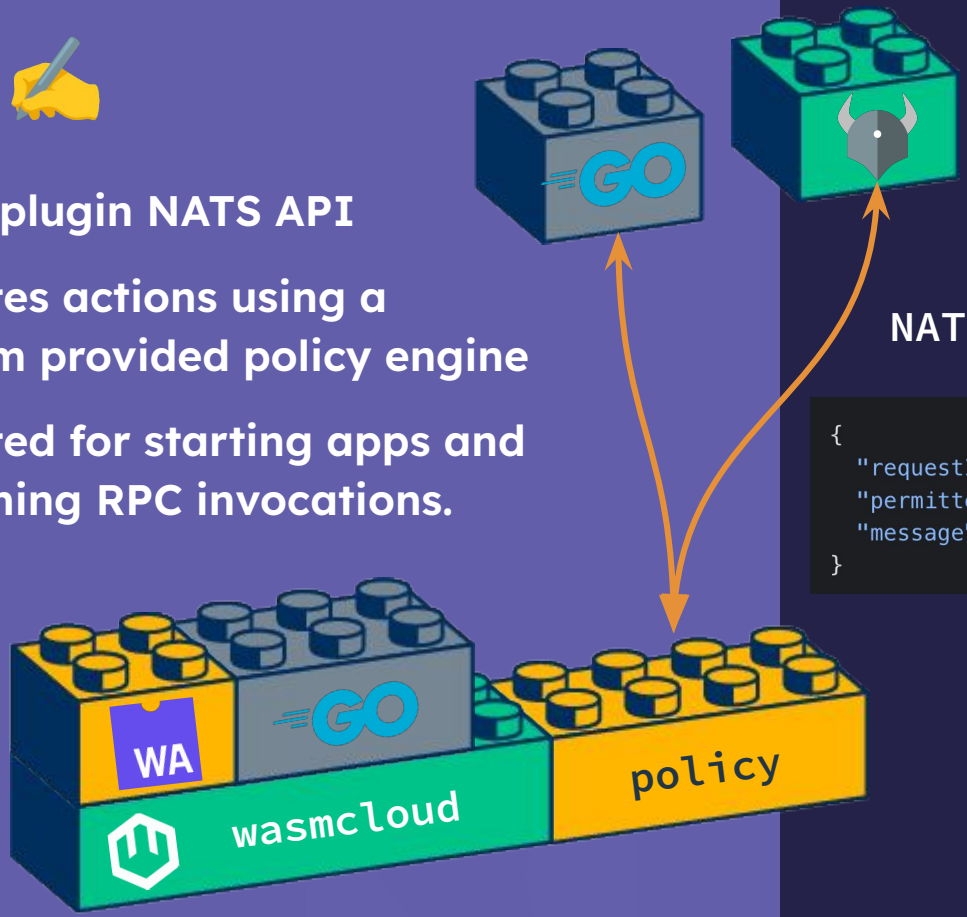
wasmCloud Platform Layers

- Sandbox & Runtime: **wasmtime**
- Application artifact: **WebAssembly Component**
- wasmCloud itself distributed using Wolfi base images
- Multi-tier pluggable security
  - Signed components
  - Fine-grained capability providers
  - **Policy service**
  - **Secrets backends**

cosmonic

Policy ✍️

- Simple plugin NATS API
- Evaluates actions using a platform provided policy engine
- Evaluated for starting apps and performing RPC invocations.




```
{
  "requestId": "... unique ID used for correlation ...",
  "permitted": true|false,
  "message": "... optional detailed message ..."
}
```

cosmonic

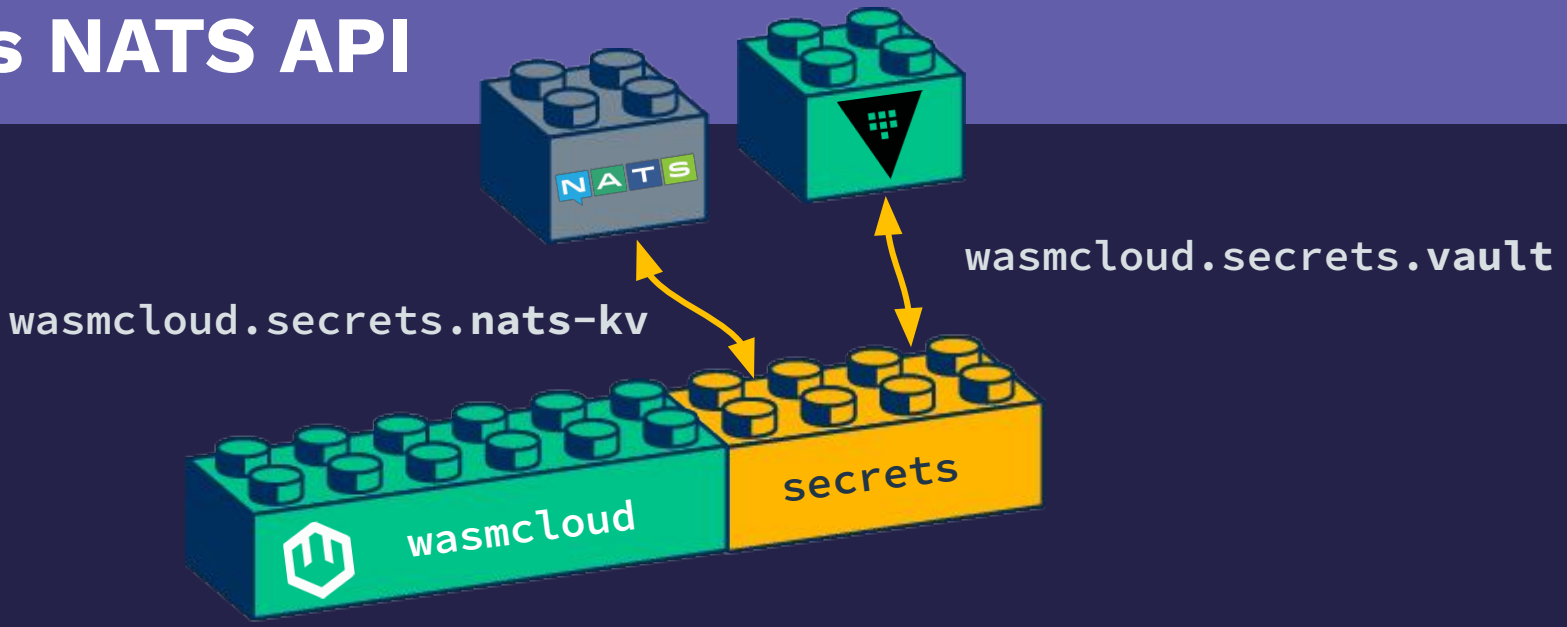
Secrets 🤫

- Extensible NATS API, can be integrated into any secrets service
- Implementations
  - NATS KV
  - Vault
  - Kubernetes Secrets

```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: http-with-secret
  annotations:
    version: v0.0.1
    description: Application with component secret
spec:
  policies:
    - name: vault-policy
      type: policy.secret.wasmcloud.dev/v1alpha1
      properties:
        backend: vault
  components:
    - name: http-component
      type: component
      properties:
        image: ghcr.io/wasmcloud/component-http-hello-world:0.1.0
        secrets:
          - name: api_key
            properties:
              policy: vault-policy
              key: component_api_key
              field: key
              version: v1
```

cosmonic

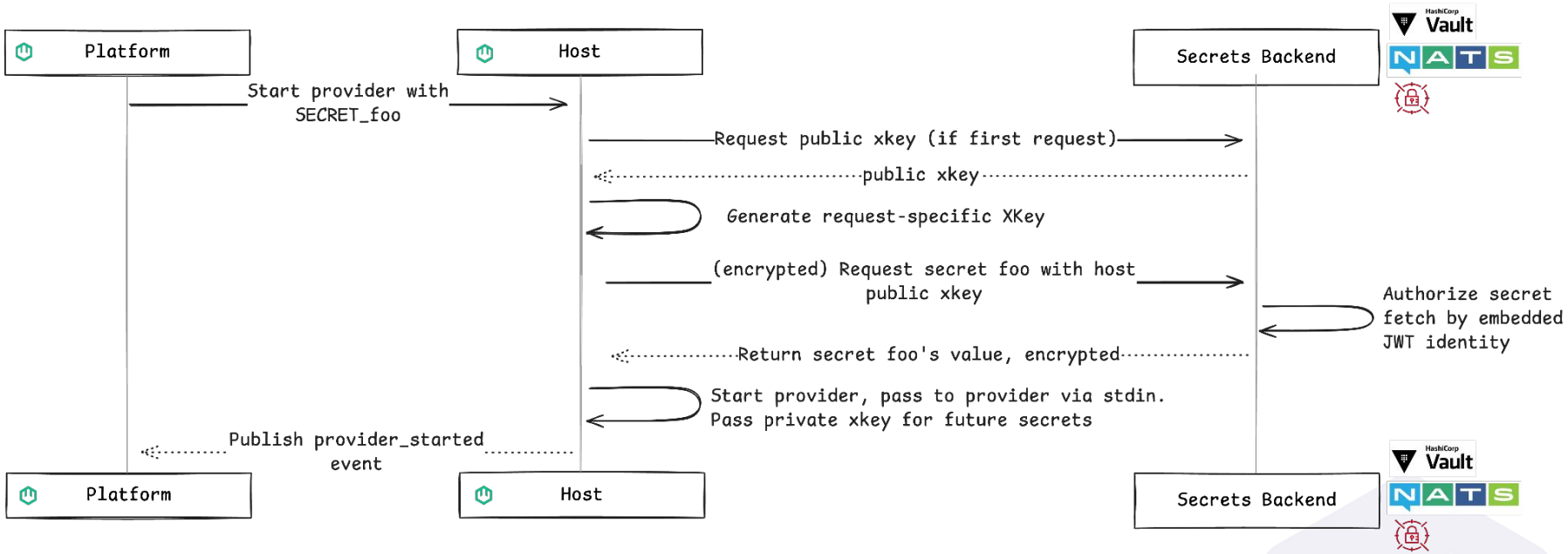
Secrets NATS API



Name	Subject	Return Payload
get	wasmcloud.secrets.nats-kv.v1alpha1.get	Encrypted SecretResponse
server_xkey	wasmcloud.secrets.vault.v1alpha1.server_xkey	Unencrypted String

cosmonic

Requesting secrets secretly



cosmonic

Demo time

Secrets locked away,

“Don’t hardcode,” they always say,

sandbox saves the day.

cosmonic

Demo takeaways

- CNCF wasmCloud has a multi-tiered security model
- Pluggable security is a fantastic way to extend the security of a platform

cosmonic

Looking Forward

- Signing is done with NATS ed25519 NKeys, would be great to implement workload identity by conforming to SPIFFE
- Out-of-the-box net/http support in TinyGo enables direct compilation to Wasm
- Golang wasip2 target support enables direct Go toolchain compilation to Wasm
- Pluggable policy as WebAssembly components would be much more powerful than our existing plugin model (NATS API)

cosmonic

Get Involved!

Read our Docs

wasmcloud.com

Star us on GitHub

github.com/wasmcloud

Join us on Slack

slack.wasmcloud.com