DAPR and Wasm; a Symbiosis for Polyglot Application Development

Sven Pfennig & Christoph Voigt
Liquid Reply GmbH

Agenda

- Meet dapr
- Meet Wasm
- Better together
- Blueprints for your use case

Who is talking?

Sven Pfennig

Principal Consultant Software Engineering at Liquid Reply

Working on cloud native application development for hybrid- and multi-cloud environments.

Tech lead WG-Wasm (TAG Runtime)

#wg-wasm @ CNCF Slack

@0xe282b0

@0xe282b0@hachyderm.io

Who is talking?

KubeCon CloudNativeCon Europe 2024

Christoph Voigt

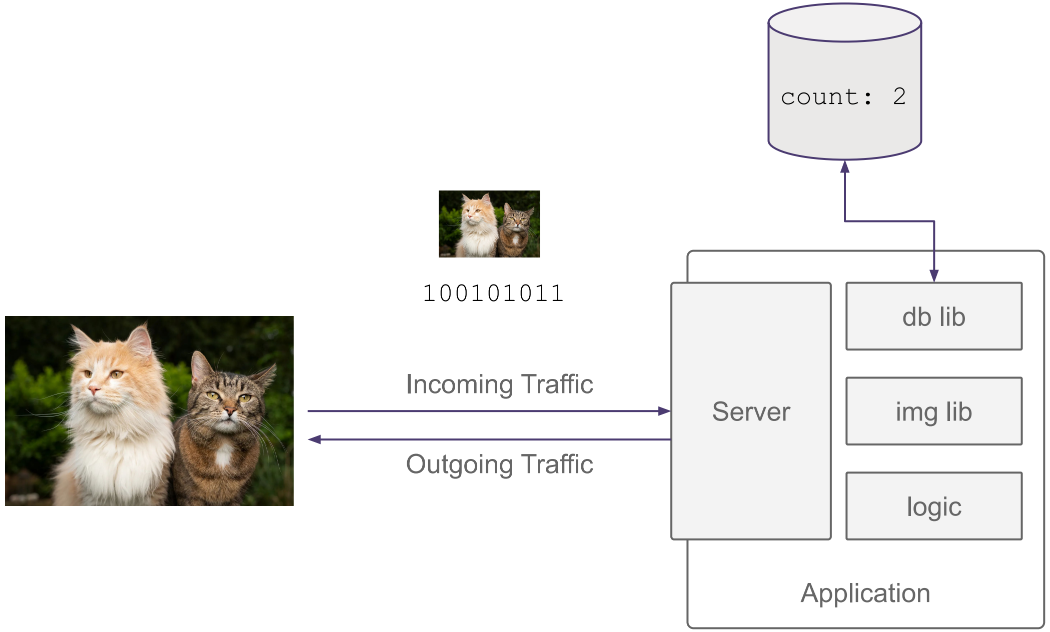
Co-Founder and developer of Liquid Reply

Software Engineering background, having a focus on Cloud Native Infrastructure- and Application-Architectures

#wg-wasm @ CNCF Slack

@vogti

@cv@hachyderm.io



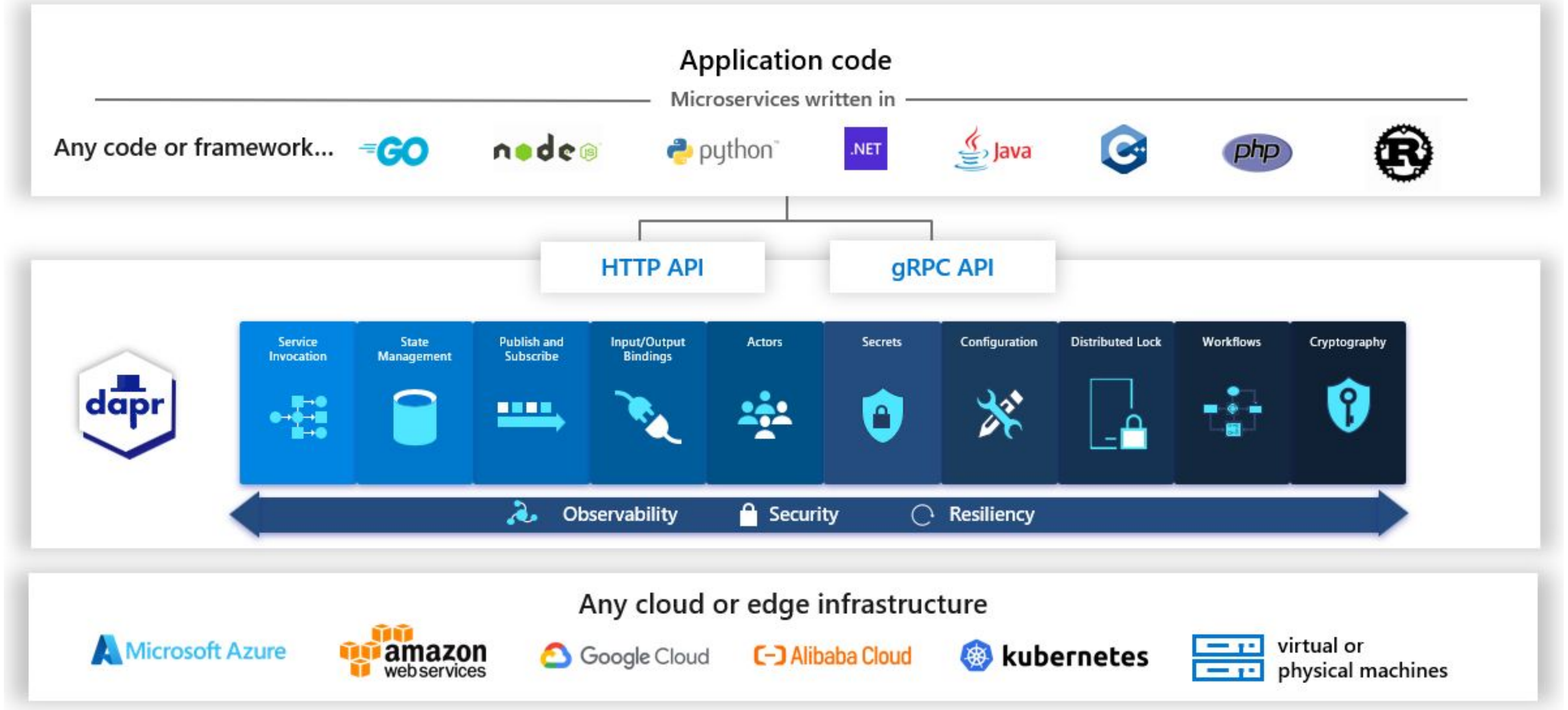
What if…

I want to change my database?

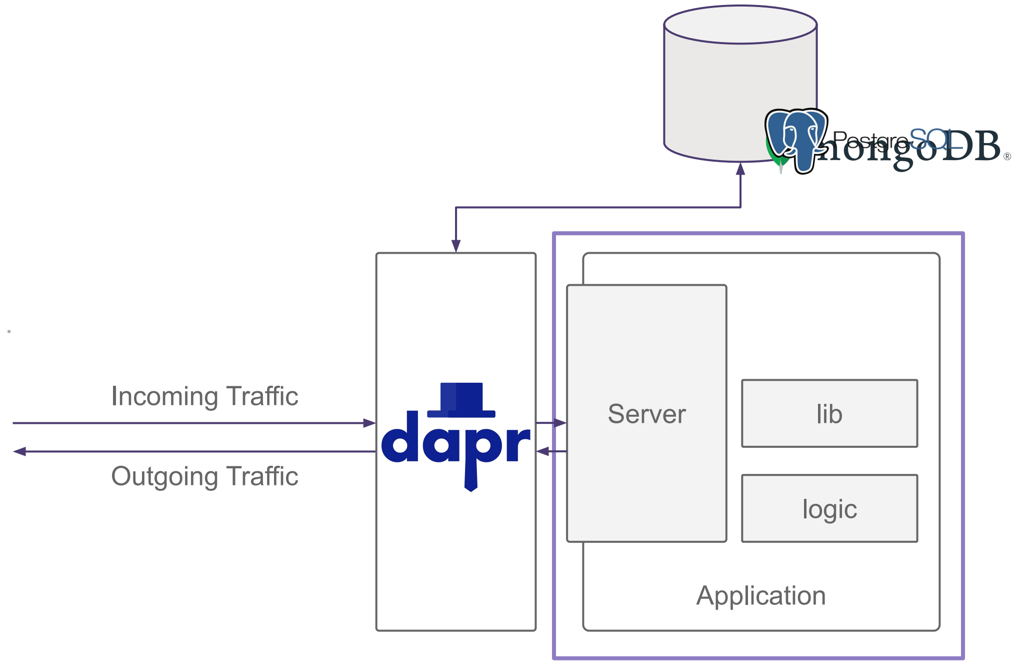
I need additional authentication?

I want to write a similar app?

Introducing Dapr



Adding Dapr to the Equation



Benefits

✅ I can change my database

✅ Dapr can handle auth for me

❓ I want to write a similar app?

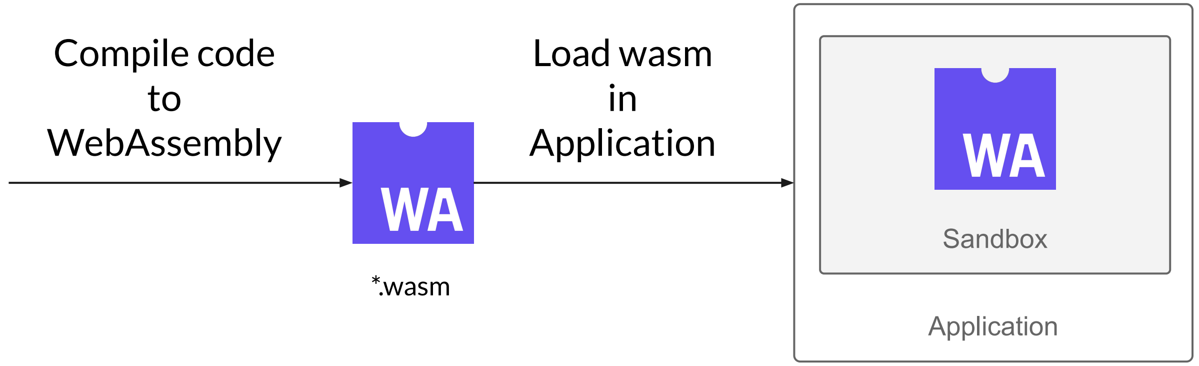
What is WebAssembly

Is a compilation target & low-level binary instruction format

Portable	➡️	makes no architectural assumptions
Safe	➡️	code is validated and executes in a memory-safe, sandboxed environment
Fast	➡️	executes with near native code performance
Language independent	➡️	does not privilege any particular language, programming model, or object model

How does WebAssembly work conceptually?

```
int add_one(int x) {
  return x+1;
}
```



Dapr & Wasm have common Goals

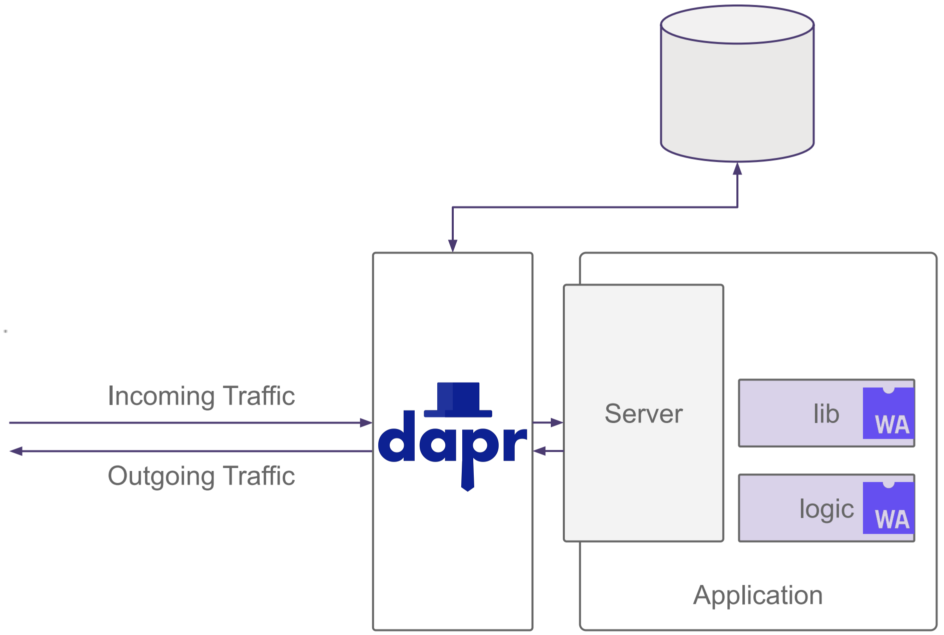
- Avoid Boilerplate, make code reusable
- Low Overhead
- Separation of Concerns
- Improve Security
- Freedom of Choice (Language & Tooling)
- **BUT the way how they implement it are on different layers**

Why would you want to run Wasm & Dapr together?

- We want to run WebAssembly because
  - Runtime Security
  - Tiny Footprint
  - Free choice of language
  - Platform independence

- Interfaces to connect to 150+ cloud services including databases, message brokers, etc.
- Observability
- Authentication (OAuth2, OIDC)
- Rate limiting and concurrency control
- Identity and access control
- …

Adding Wasm to the Equation



Benefits

✅ I can change my database

✅ Dapr can handle auth for me

✅ I want to write a similar app?

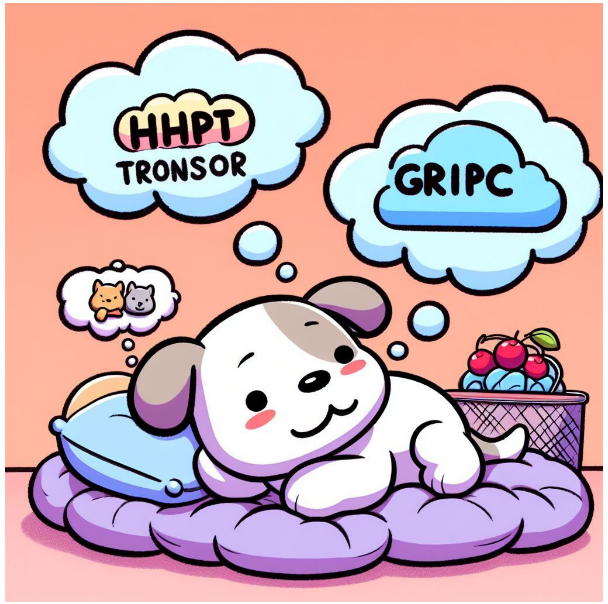
How to connect WebAssembly & Dapr?

Heterogeneous use cases

- Architecture: FaaS, Microservice, Event driven
- Runtimes: WasmTime, WasmEdge, Wazero, Wasmer and more…
- Languages: Rust, Go, JavaScript, Python, Zig, …
- Invocation rates: Most important for the type of integration

Lot of choices available

- Dapr/Wasm Integration on Kubernetes; a lot of parameters:
  - Sidecar / Daemonset
  - HTTP/GRPC
  - Standalone Runtime/ Embedded SDK
  - Vertical / Horizontal Pod Autoscaler
  - …



A Blueprint

Considerations:

- What use case should you consider this blueprint for

Prerequisites:

- What do you need to apply the blueprint

Variations:

- How to apply and adapt the blueprint

Limitations:

- When to not use this blueprint

Wazero HTTP Middleware

KubeCon CloudNativeCon Europe 2024

Considerations:

- Wazero is already included
- Hosting pure functions
  - Transformation
  - Validation
- Low/Medium invocation rates

Prerequisites:

- Dapr deployment
- Function compiled to Wasm
- A way to provision the wasm module

Variations:

1. Wasm in container at buildtime
2. Mount a volume
3. Download during startup

Alternative:

- Customize dapr behavior without recompiling

Limitations:

- No side effects in function

Microservice on Standalone Wasm Runtime

Considerations:

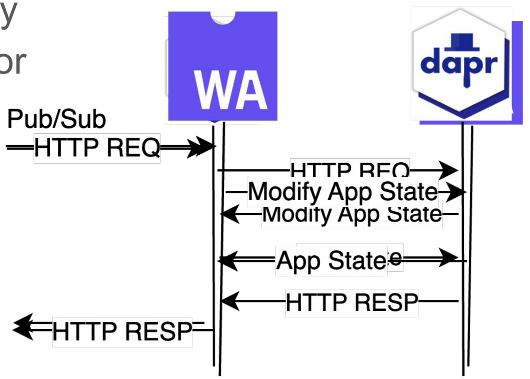
- Microservices are a common architectural pattern
- Almost all Wasm runtimes have HTTP support
- Dapr provides state management support
- Bindings and pub/sub can be used as triggers
- High invocation rates, scalable deployments

Prerequisites:

- Dapr deployment
- Wasm runtime
- Support for HTTP client/servrer

Variations:

1. Access to state management, output bindings, service invocation, …
2. Get triggered by Input bindings or pub/sub.



Limitations:

- Non HTTP connections

Embedded Wasm Runtime

Considerations:

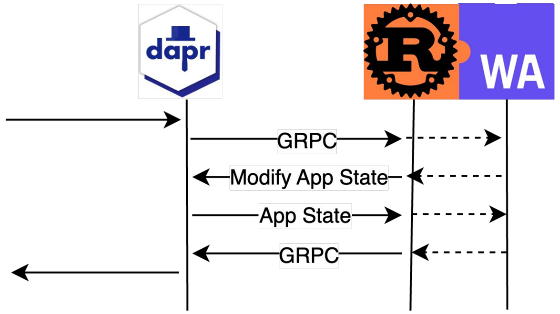
- HTTP calls have quite some overhead
- Some use cases like event streaming need really high throughput.
- GRPC implementations are not common in the Wasm ecosystems
- Wasm is a great plugin system
- Host function calls that map to GRPC increase the throughput dramatically

Prerequisites:

- Runtime with SDK (almost all)
- Supported host language
- Some development effort

Variations:

1. Rust host language, pass trough calls to dapr
2. Reduce talks with local state and caching



Limitations:

- Additional development effort

Summary

- Does Dapr + Wasm live up to its promises?
  - You can start with almost no boilerplate code and increase complexity if needed
  - Observability which is actually hard to implement in Wasm applications
  - Flexible variants can be adapted to the actual use case
- Good for…
  - Cloud native architectures
- Not good for…
  - highly specialized protocols e.g. UDP based
  - Systems programming

Where to go from here

KubeCon CloudNativeCon Europe 2024

More on Dapr & WebAssembly

- Rust an WebAssembly (Michael Young, Secondstate)
  - https://www.manning.com/liveprojectseries/rust-and-webassembly-ser
- wasm-dapr template project
  - https://github.com/second-state/dapr-wasm
- Dapr with WebAssembly Course
  - stay tuned… (and follow our socials)

More on WebAssembly & Kubernetes

- Course on WebAssembly from Kubesimplify (Saiyam Pathak & Rishit Dagli)
  - https://www.youtube.com/watch?v=eYekV2Do0YU
  - or search for "**Kubesimplify Wasm**" on YouTube
- **Spinkube**
  - https://www.spinkube.dev/

@vogti

@cv@hachyderm.io

@0xe282b0

@0xe282b0@hachyderm.io

@LiquidReply

Feedback welcome: