wasmCloud

基本概念

- actor
  - （面向业务开发）编写业务逻辑的地方
  - 如果涉及到非业务逻辑的公共能力（或中间件能力），通过引用interface的方式来获取能力。
- interface
  - 定义一些业务无关的公共能力接口，比如：keyvalue能力，但是不提供具体的实现。
    - 可以在 interface 中定义允许的操作(operations)，以及各种结构体(structure)。
    - 使用 smithy IDL 进行 interface 的定义，使用 wash 工具来生成对应代码。

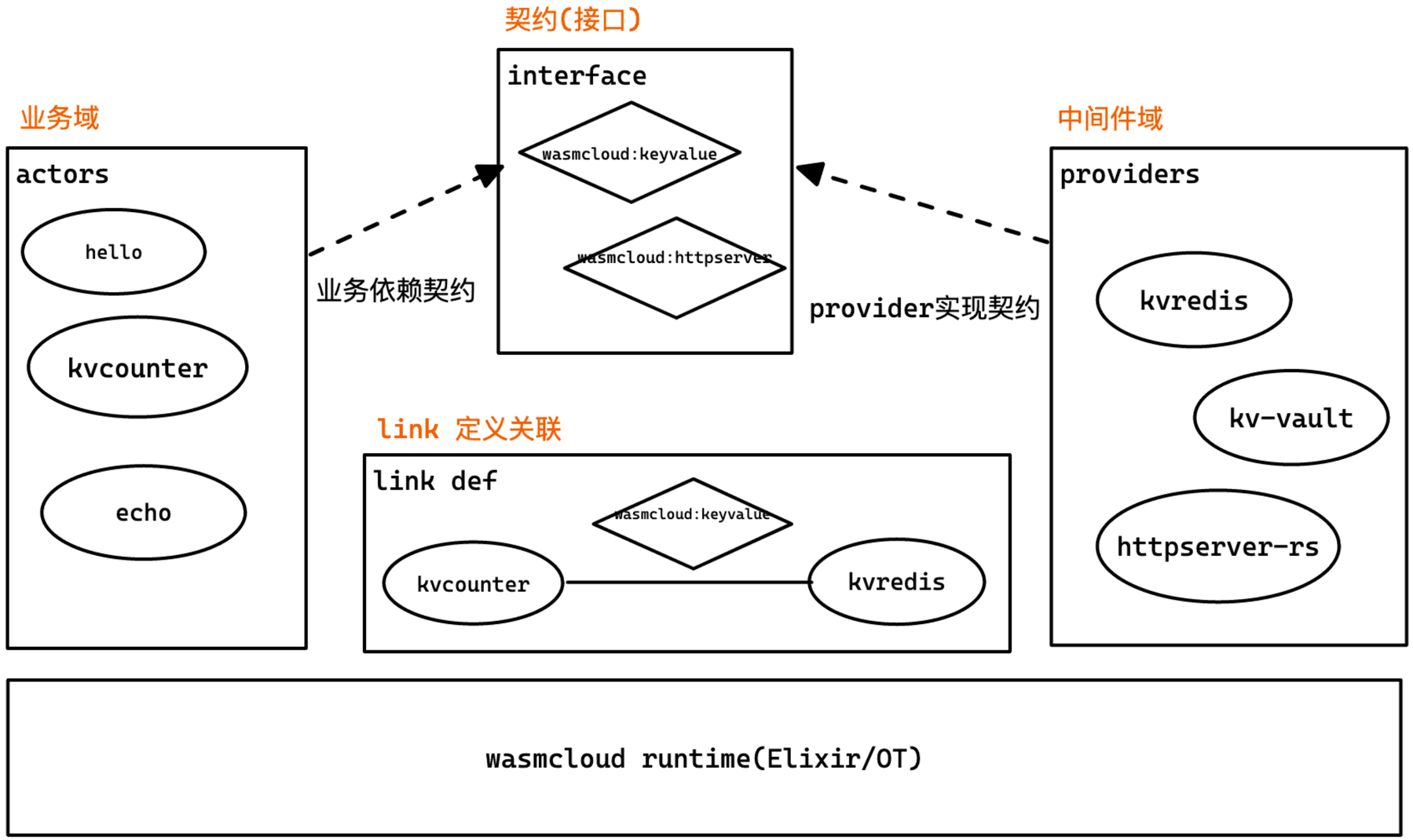
wasmCloud

基本概念

- provider
  - （面向中间件或平台开发）实现interface的能力，比如同样对于keyvalue功能，可以有redis实现，也可以有vault实现。并且可以用不同的技术栈来实现。
- link def
  - （面向业务开发或部署人员）由于actor只是引用了具体的interface，并没有指定哪一个provider，所以在实际部署时，需要关联actor和provider。
- wasm host
  - 用于协调actor,provider之间的分布式通信，同时也提供分布式调度能力。采用 Elixir/OTP 技术栈实现。

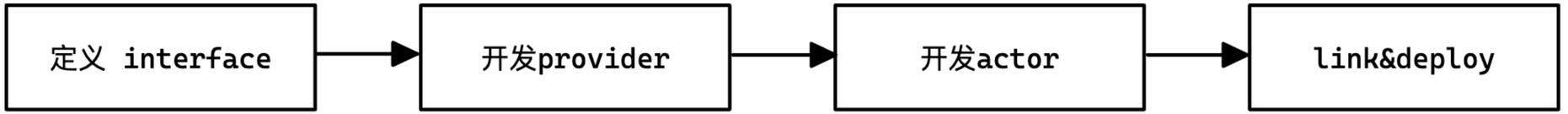
wasmCloud

整体关系图



wasmCloud

开发流程

- 基本步骤
  - 

- 示例
  - 开发一个 kvcounter 示例程序，当用户访问 http://localhost:8080 时，能够正常显示 counter，且每次访问 +1。

wasmCloud

开发流程

- 创建（或使用）interface
  - 在本示例中，我们用到了 kv 功能，所以需要找到一个能够提供 keyvalue 功能的 interface。（可以自建或使用已有的）
  - 编写 interface，即：smithy 文件。首先确定该接口提供的操作，然后是每个操作（operations）涉及到的结构体(structure)：

```
/// the return structure contains exists: true a
/// otherwise the return structure contains exis
@readonly
operation **Get** {
  input: String,
  output: GetResponse,
}

/// Sets the value of a key.
/// expires is an optional number of seconds bef
/// or 0 for no expiration.
operation **Set** {
  input: SetRequest,
}
```

Provider	Contract	OCI Reference & Description
blobstore-fs	`wasmcloud:blobstore`	custom badge inaccessible Blobstore implementation where blobs are local files and containers are folders
blobstore-s3	`wasmcloud:blobstore`	custom badge inaccessible Blobstore implementation with AWS S3
httpserver	`wasmcloud:httpserver`	custom badge inaccessible HTTP web server built with Rust and warp/hyper
httpclient	`wasmcloud:httpclient`	custom badge inaccessible HTTP client built in Rust
redis	`wasmcloud:keyvalue`	custom badge inaccessible Redis-backed key-value implementation
vault	`wasmcloud:keyvalue`	custom badge inaccessible Vault-backed key-value implementation for secrets
nats	`wasmcloud:messaging`	custom badge inaccessible NATS-based message broker
lattice-controller	`wasmcloud:latticecontroller`	custom badge inaccessible Lattice Controller interface
postgres	`wasmcloud:sqldb`	custom badge inaccessible Postgres-based SQL database capability provider

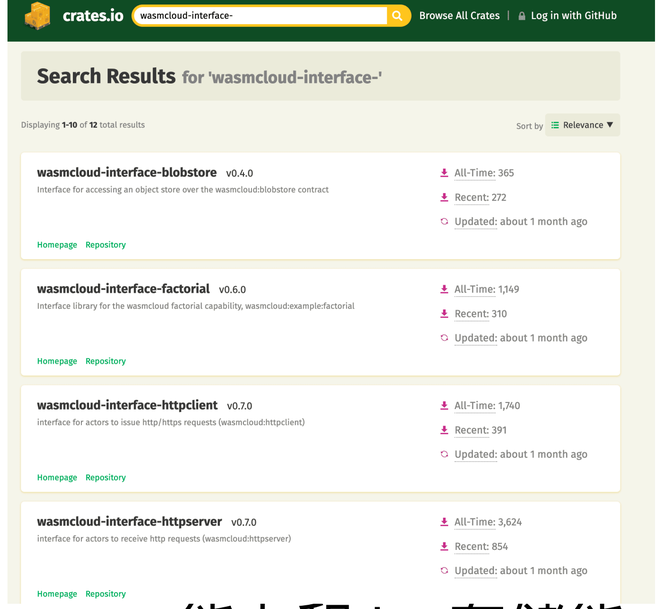
wasmCloud

开发流程

- 创建（或使用）interface
  - 生成代码，发布到对应的仓，方便actor开发人员使用（比如rust就是发布到crate.io）
- 开发provider
  - 基于 redis 实现 interface 中定义的各种 operations
  - 将provider打包为wasm文件，并将其推送到oci仓库，方便使用

wasmCloud

开发流程



- 开发actor
  - kvcounter 依赖了两个 interface，用来提供两个能力：httpserver能力和 kv 存储能力。我们需要声明这两个依赖。
    - 以rust为例
      - 可以在crates.io中查阅已有的interface，然后在cargo.toml中加入依赖的两个包
      - 在actor中调用
  - 将代码打包为.wasm文件，并推送到oci仓库

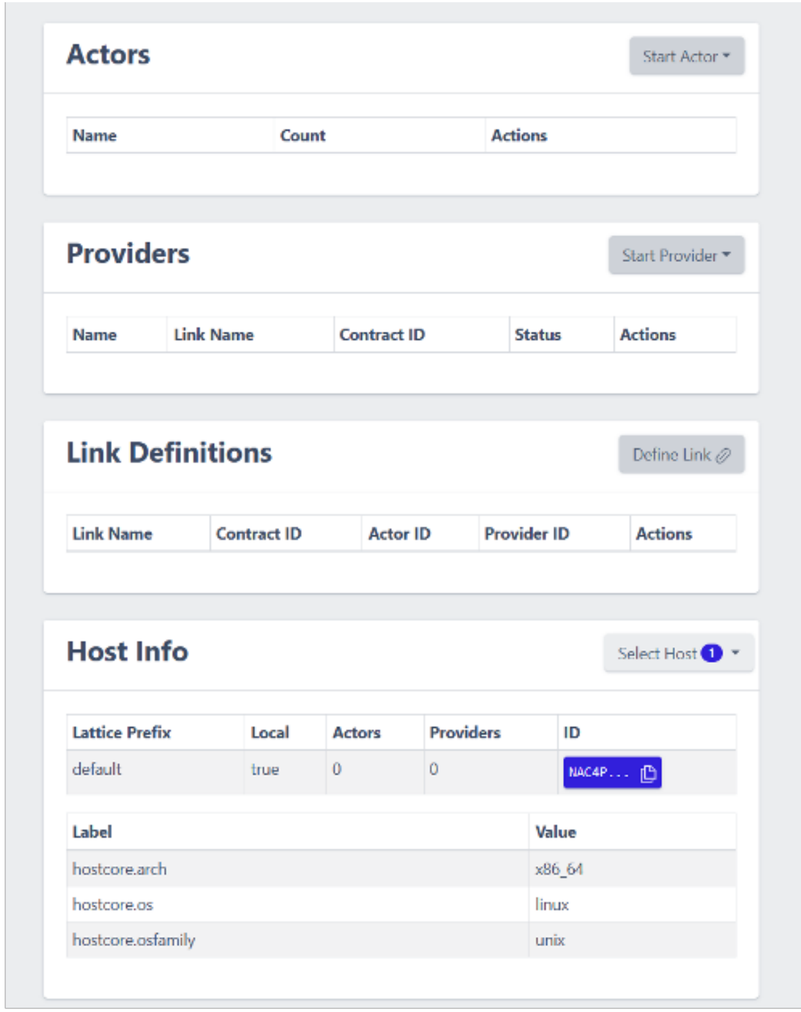
```
[package]
name = "kvcounter"

[dependencies]
**wasmcloud-interface-keyvalue = "0.7.0"**
**wasmcloud-interface-httpserver = "0.6.0"**
```

wasmCloud

开发流程

- 部署
  - 使用 wasmcloud shell(即 wash) 或者 web dashboard 进行部署
  - **start actors**：将 actor 的 wasm 文件上传到 wasmcloud 运行时，或者提供 registry 地址供其下载。
  - **start providers**：将 provider 的包上传到 wasmcloud 运行时，或提供 registry 地址供其下载。
  - 建立 link 关系



wasmCloud

生产案例

- 2021.10 Helping One of Europe’s Largest **Banks** Re-platform with Declarative, Self-healing, **Multi-cloud wasmCloud Clusters** on **Kubernetes**
  - 痛点
    - 微服务架构复杂，需要考虑的因素太多
    - For companies in **regulated industries**, like **financial services or healthcare**, regulators will insist that organizations have a **‘get out’ plan**. Realistically these plans aren't yet real and organizations wouldn't be able to execute them, at least not quickly. They are in place to satisfy the regulations. The reality is that the **cost of moving everything to another cloud provider is prohibitive**.
    - we’re still **locked** into **cloud-provider specific services** — AWS is famous for the number of services it keeps introducing. And every **cloud provider-specific service** we use **locks** us in further.
    - **multi-cloud or multi-location** requires **a higher level abstraction** that sits above the HTTP and TCP IP network that dominates **microservices** today.

wasmCloud

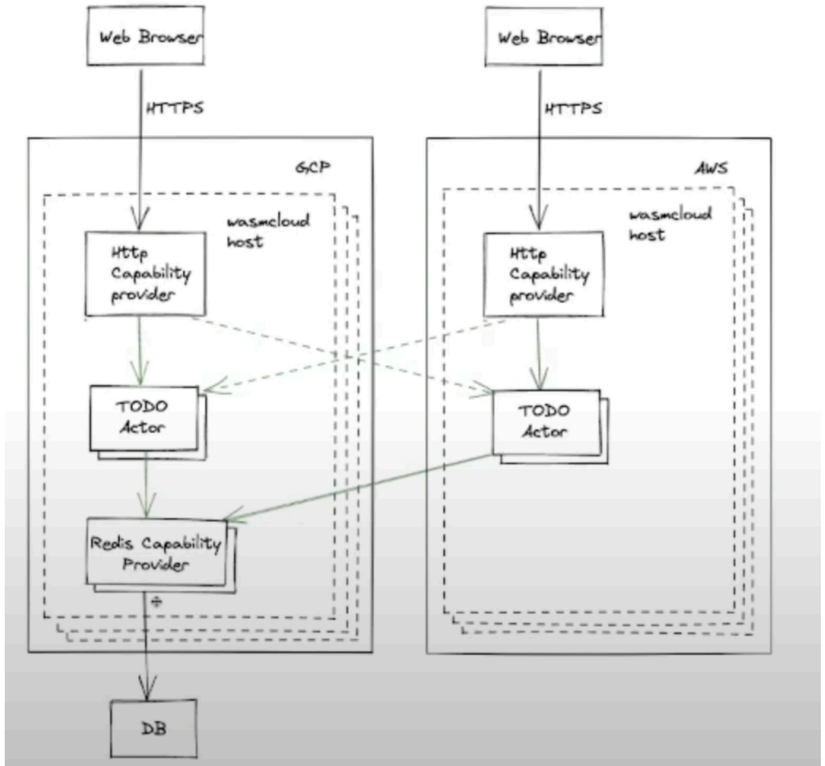
生产案例

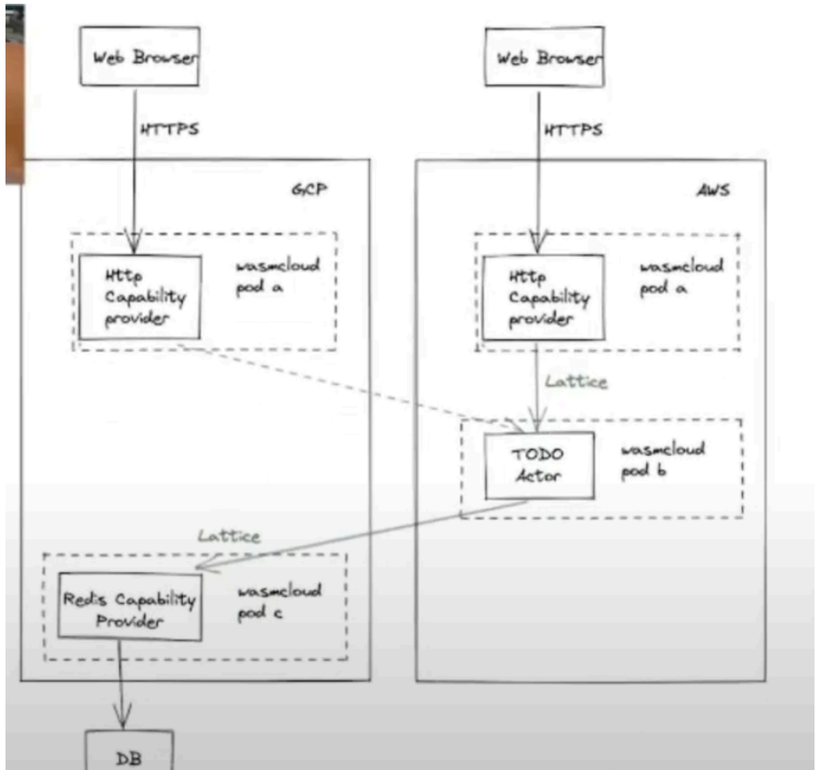
- Helping One of Europe’s Largest **Banks** Re-platform with Declarative, Self-healing, **Multi-cloud wasmCloud Clusters** on **Kubernetes**
  - It’s the first time where developers can have a true **multi-location, cluster** spread **across geographic locations worldwide**. And it's made possible by **WebAssembly** and distributed application runtimes like **wasmCloud** for lightweight, portable workloads and true location independence.
  - This was all in preparation for a world where the bank could **easily** deploy workloads **securely** and **reliably** across on premise and any cloud **without** having to **constantly adjust network topology** to suit.
  - Running on top of NATS, on top of Kubernetes (for now), on top of Cloud.

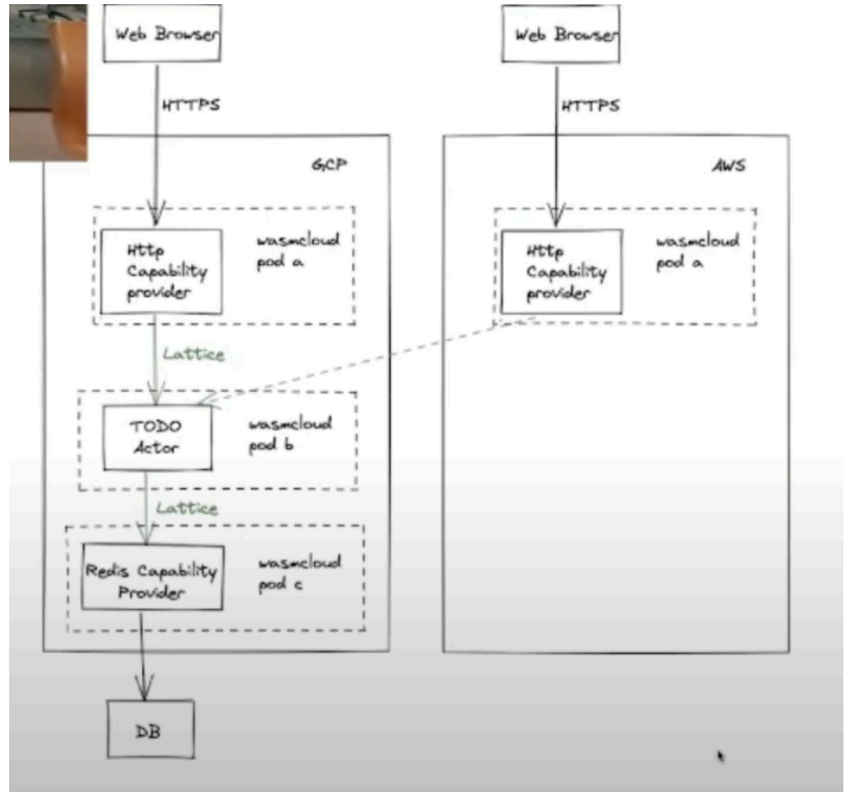
wasmCloud

生产案例

- We saw “work” being transferred transparently between cloud providers (and back again) — totally seamless failover between different geographic locations.

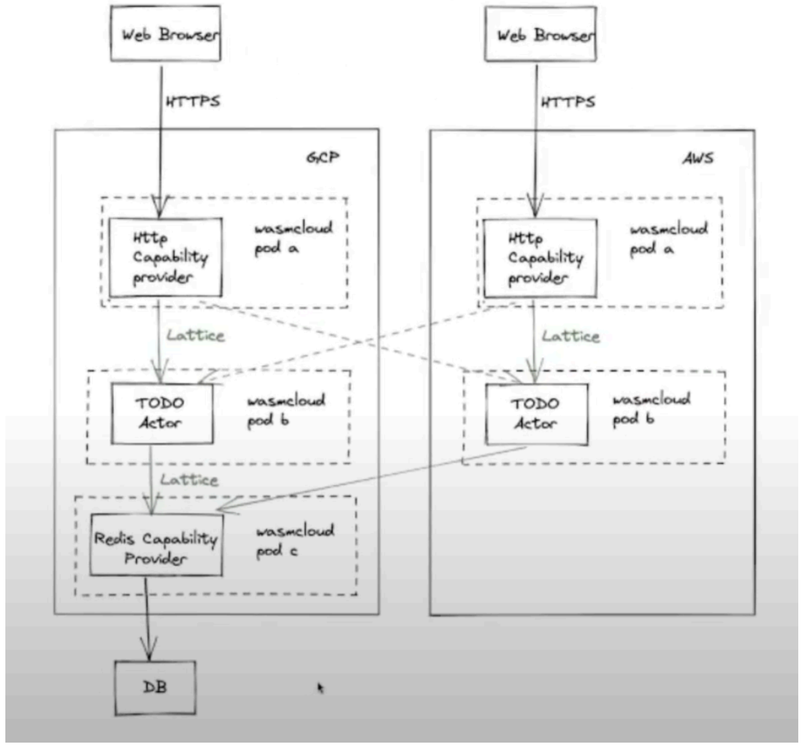


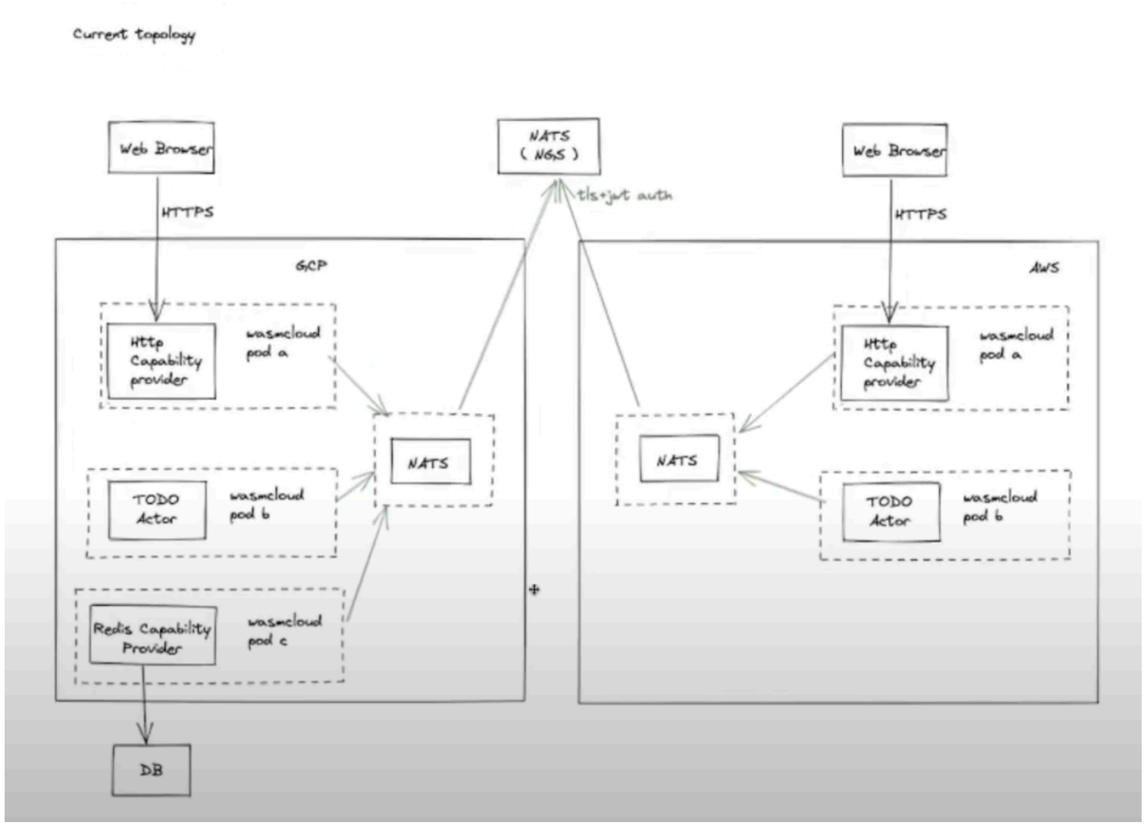




wasmCloud

生产案例



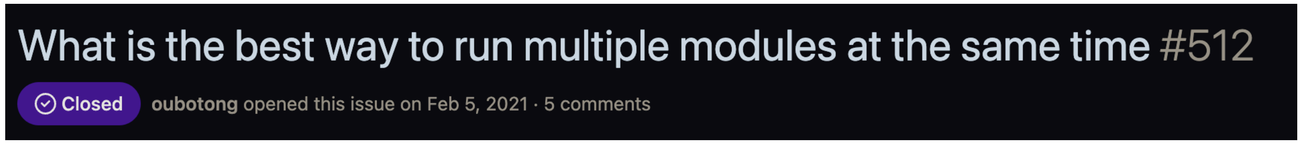


wasmCloud

优点

- 更适合云的微服务架构
- 支持multi-cloud
- 可靠性高
- 安全性高
  - capability模型
- “free from almost all the **boilerplate code** that consumes so much of developers’ energy today”

wasm runtime同时运行多个wasm module

- 

  - WAMR
  - You can create multiple threads through **platform APIs** such as **os_thread_create**, then you can load, instantiate and execute the module in there own threads.
  - the **os_thread_create** API is just a wrapper of the OS's thread interface. The context switch is handled by the OS