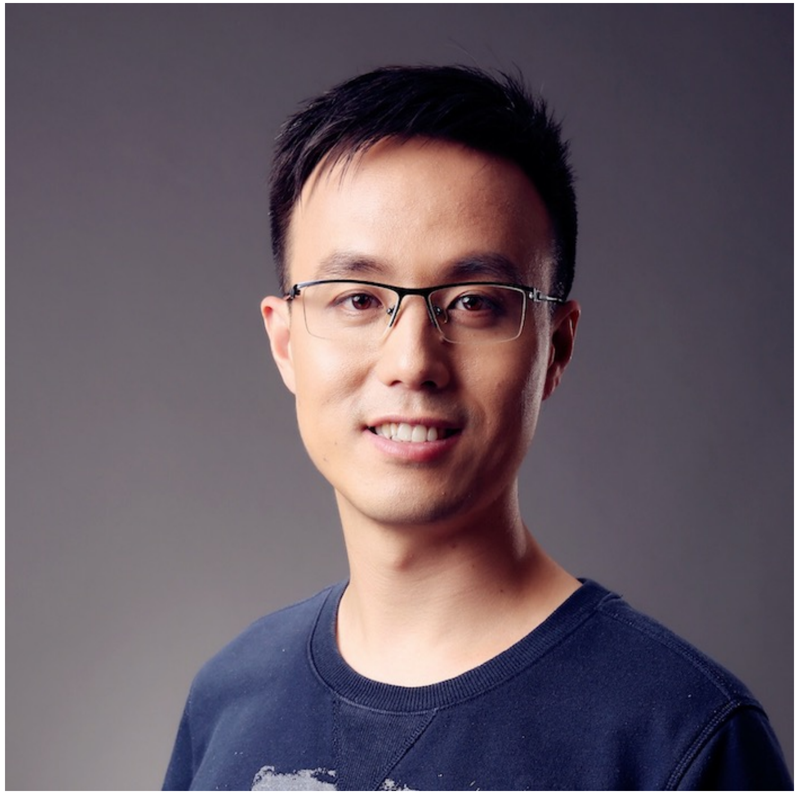
开源发展委员会

2022 CCF中国开源大会
2022 CCF ChinaOSC
开源创新 引领未来

消失的艺术：
云原生中的运行时与Serverless

报告人：宋顺

2022年11月6日



蚂蚁集团高级技术专家
2019 年加入蚂蚁集团，目前专注于云原生、微服务和消息等方向，如 Service Mesh、Application Runtime、Serverless 等

开源爱好者，Apollo Config PMC
https://github.com/nobodyiam

毕业于复旦大学软件工程系
曾就职于携程、大众点评，负责中间件、后台系统等研发工作

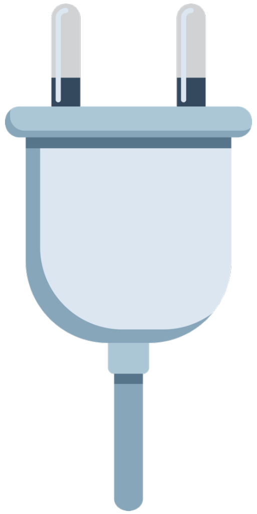
最卓越的技术是那些“消失不见的”技术

The most profound technologies are those that disappear.

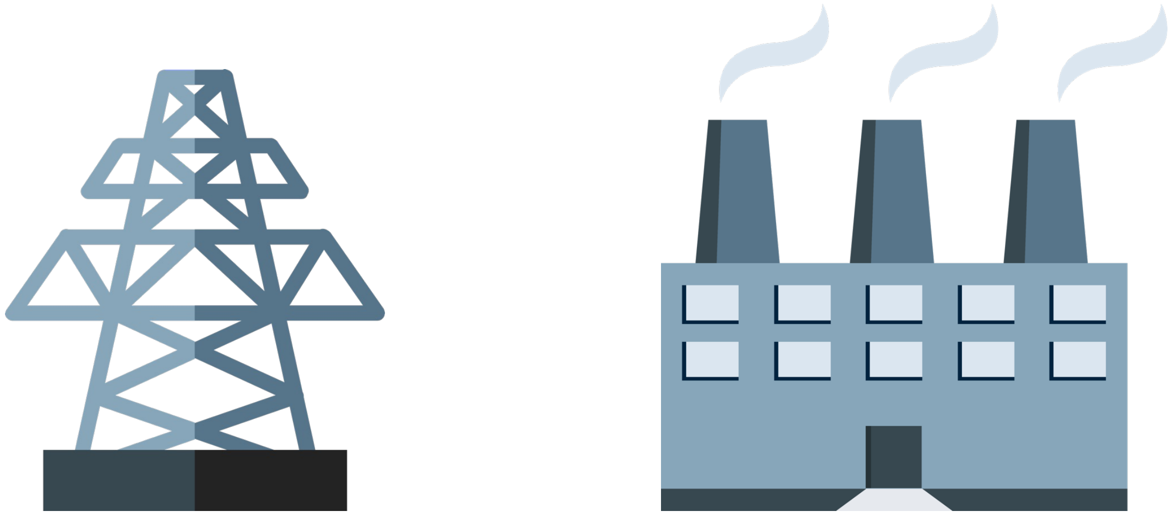
—— Mark Weiser

泛在计算概念的提出者

生活中消失不见的技术

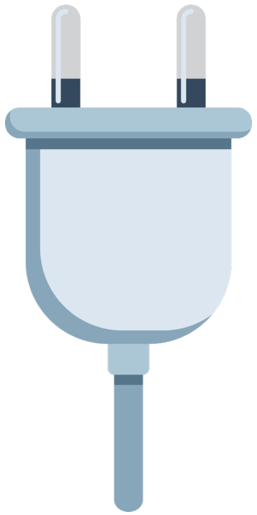


电 - 即插即用



无需感知电力基础设施

生活中消失不见的技术

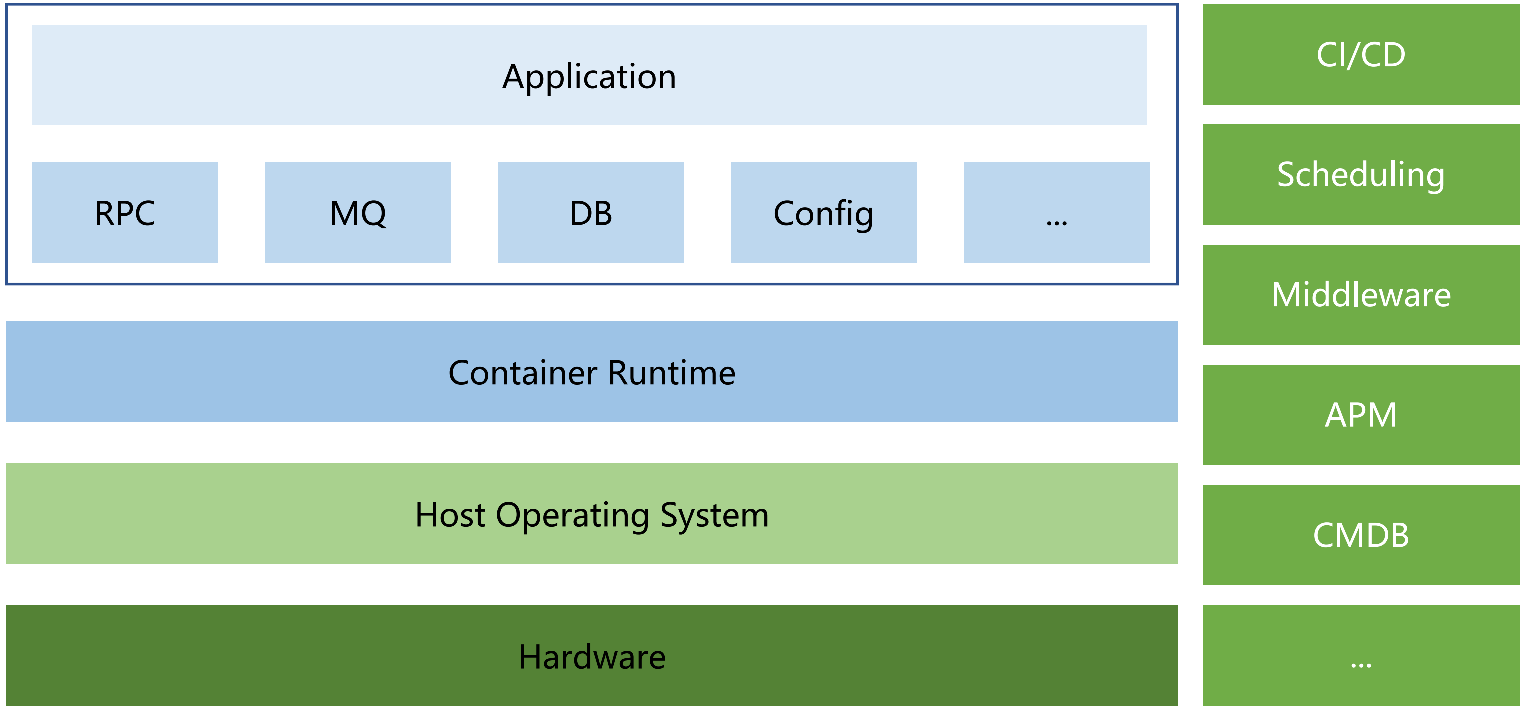


电 - 即插即用

水 – 随开随有

煤 – 随开随用

业务开发与基础设施



研发态负担

- 中间件 SDK 升级
- 多云差异感知
- ...

运维态负担

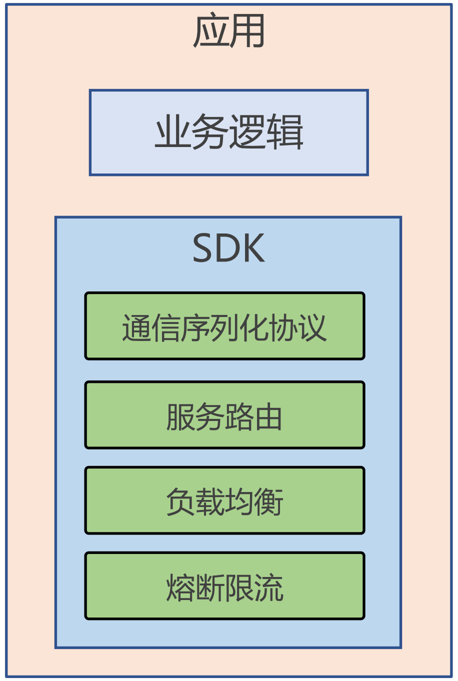
- 发布流程
- 资源申请
- 容量评估
- ...

技术基础设施的存在感越强，研发运维的效率就越低，无法把精力集中在业务创新上

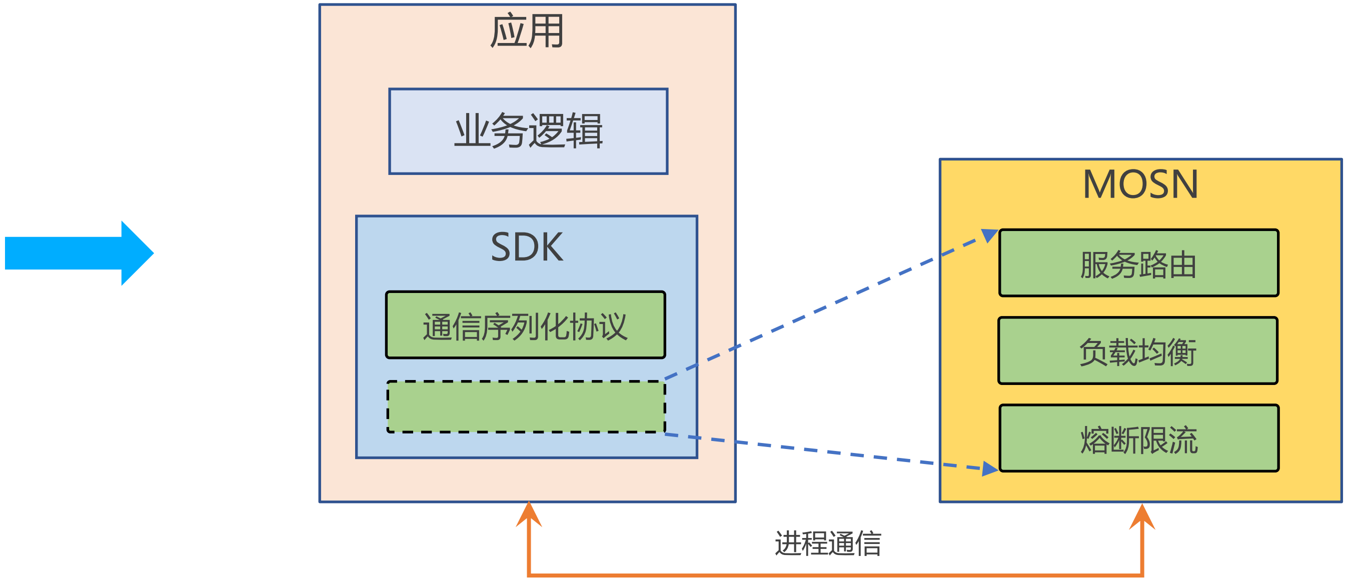
如何让技术基础设施消失不见?

从服务网格到应用运行时

服务网格的实践 - MOSN



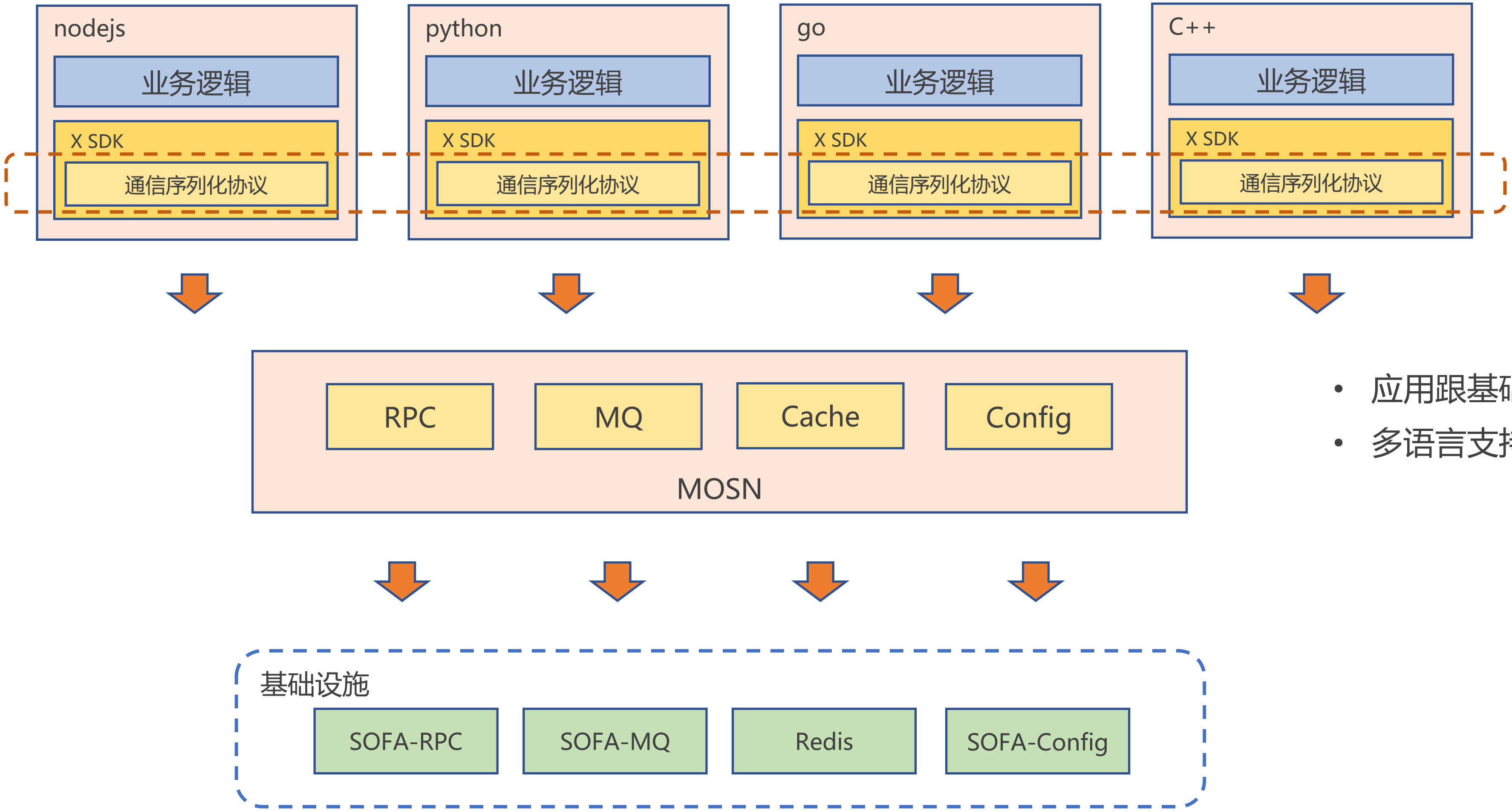
- 升级成本高
- SDK 版本碎片化
- 异构语言治理能力弱



- 业务解耦
- 独立演进、透明升级
- 异构语言治理

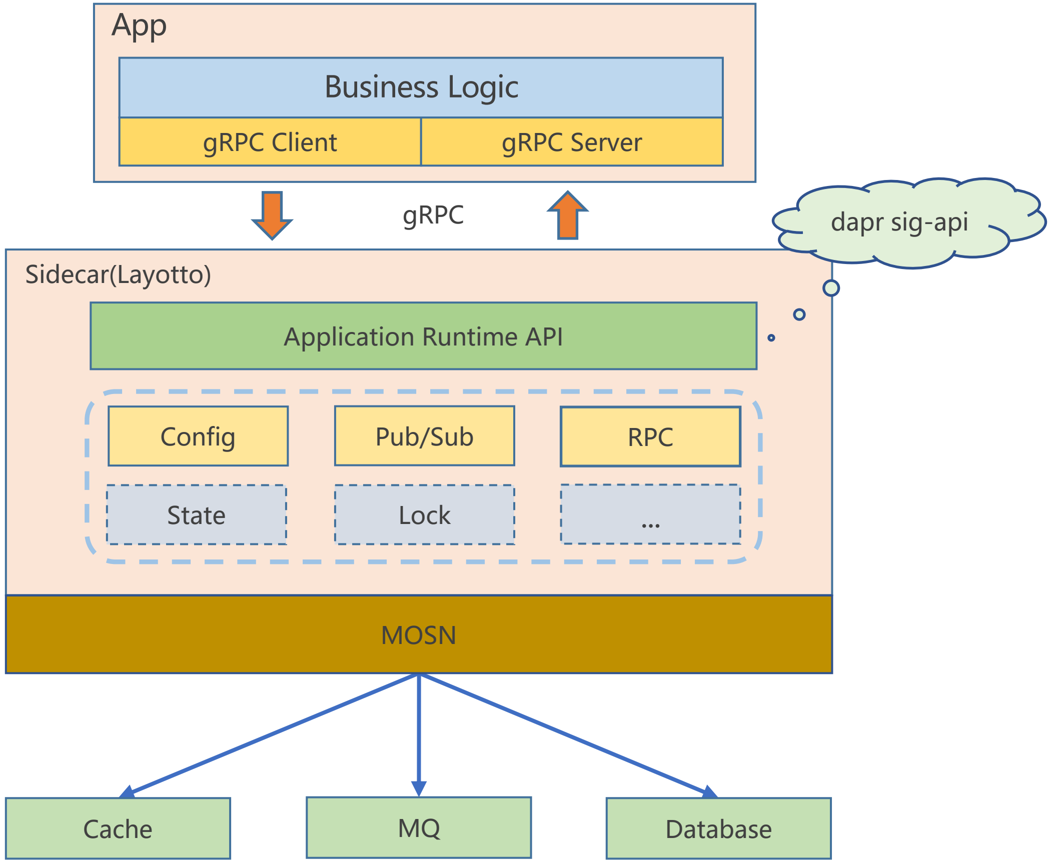
项目地址：https://github.com/mosn/mosn

泛网格化的探索



- 应用跟基础设施强绑定
- 多语言支持成本高

应用运行时的探索 - Layotto

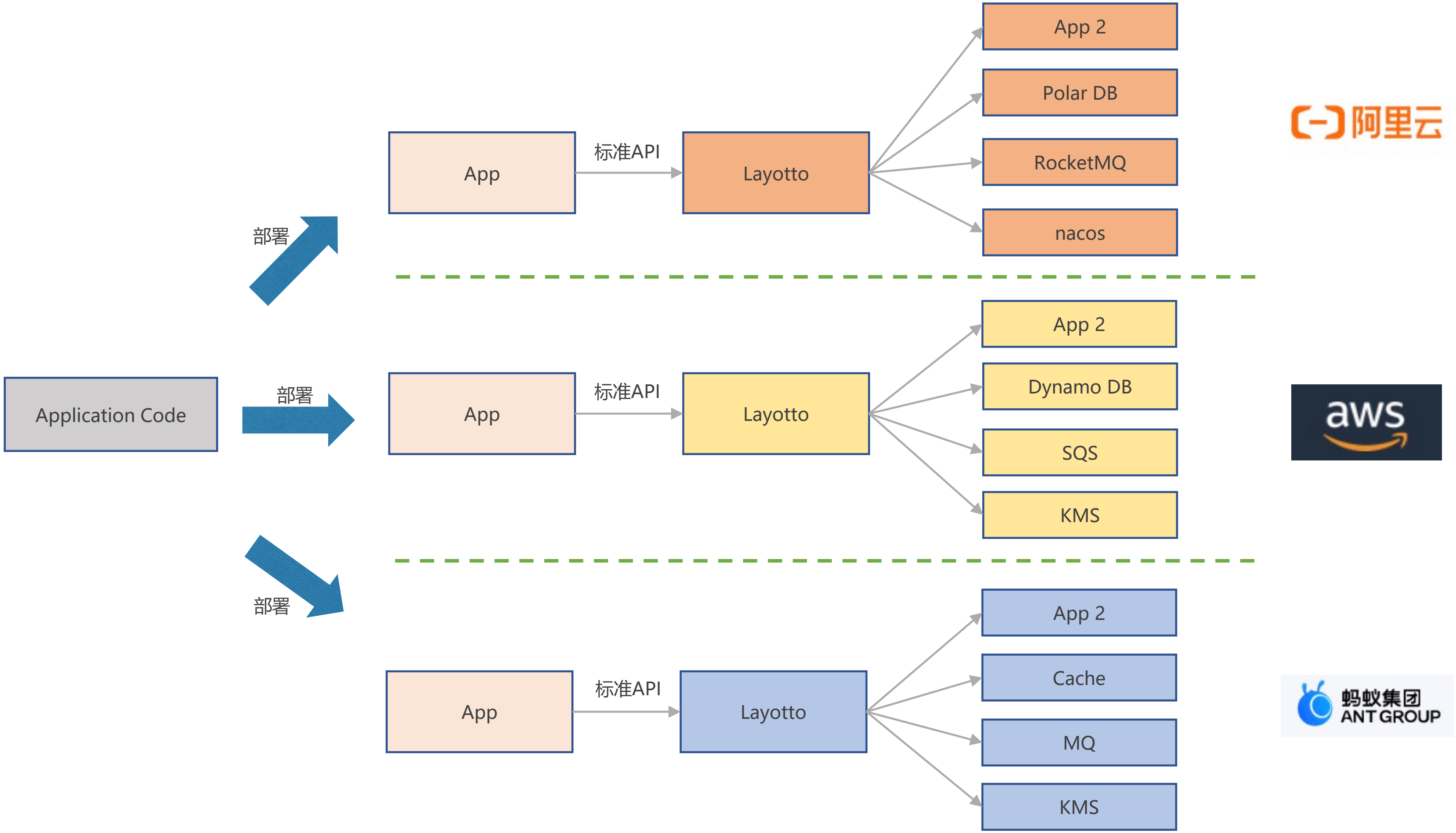


- 抽象分布式能力的标准 API
- 中间件能力下沉到 Layotto 中
- 使用多语言友好的 gRPC 协议通信

```
service Runtime {
  // InvokeService do rpc calls
  rpc InvokeService(InvokeServiceRequest) returns (InvokeResponse) {}

  // Gets the state for a specific key.
  rpc GetState(GetStateRequest) returns (GetStateResponse) {}

  // Saves an array of state objects
  rpc SaveState(SaveStateRequest) returns (google.protobuf.Empty) {}

  // Publishes events to the specific topic.
  rpc PublishEvent(PublishEventRequest) returns (google.protobuf.Empty) {}

  ...
}
```

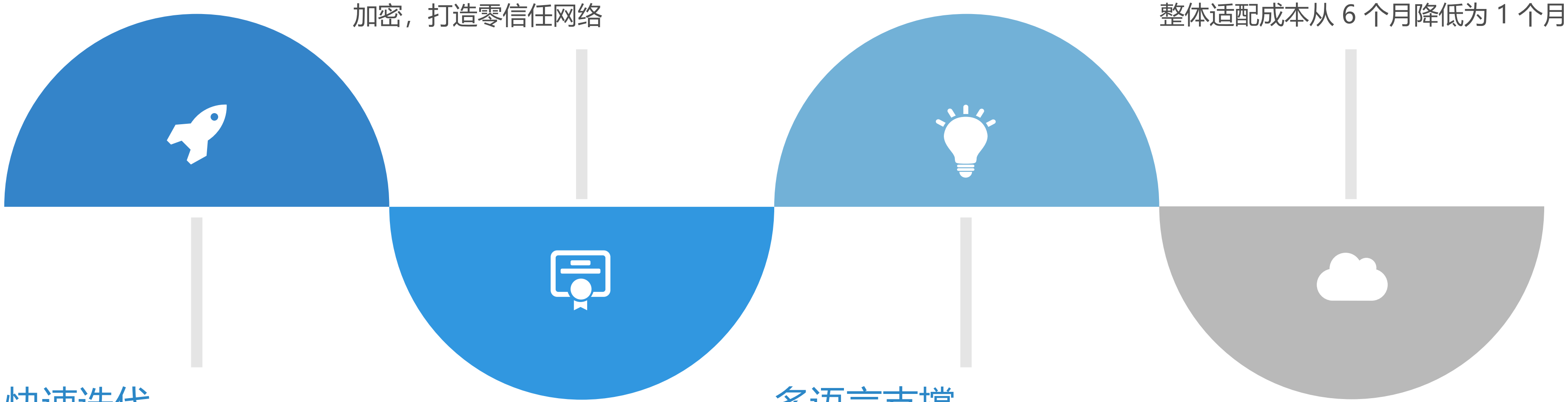
项目地址：https://github.com/mosn/layotto

Layotto 的可移植性



项目地址：https://github.com/mosn/layotto

应用情况



快速迭代

通过 MOSN 实现了基础设施升级能力从 1 ~ 2 次/年提升为 1 ~ 2 次/月，节省每年数千人日的升级成本

安全可信

通过 MOSN 实现了身份认证、服务鉴权、通信加密，打造零信任网络

多语言支撑

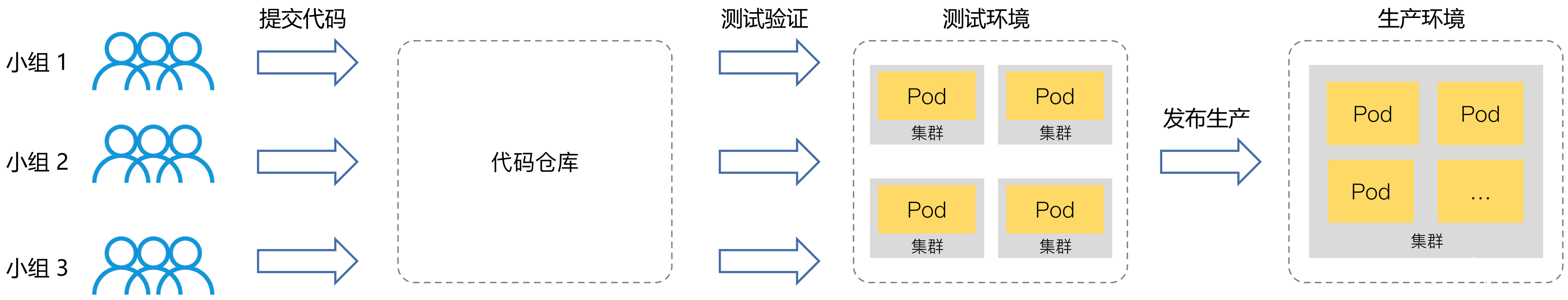
Layotto 接入了 200+ 非 Java 技术栈应用，提供了多语言一致的基础设施能力

多云输出

Layotto 支撑了 SaaS 业务低成本多云输出，整体适配成本从 6 个月降低为 1 个月

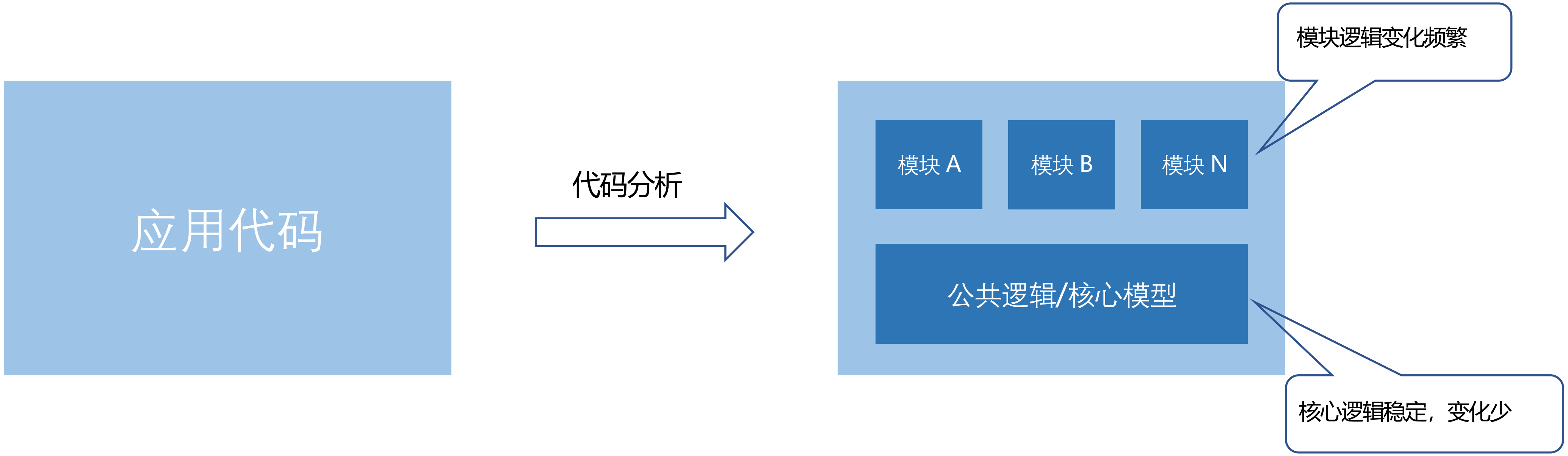
从 Server 到 Serverless

Serverless 的尝试

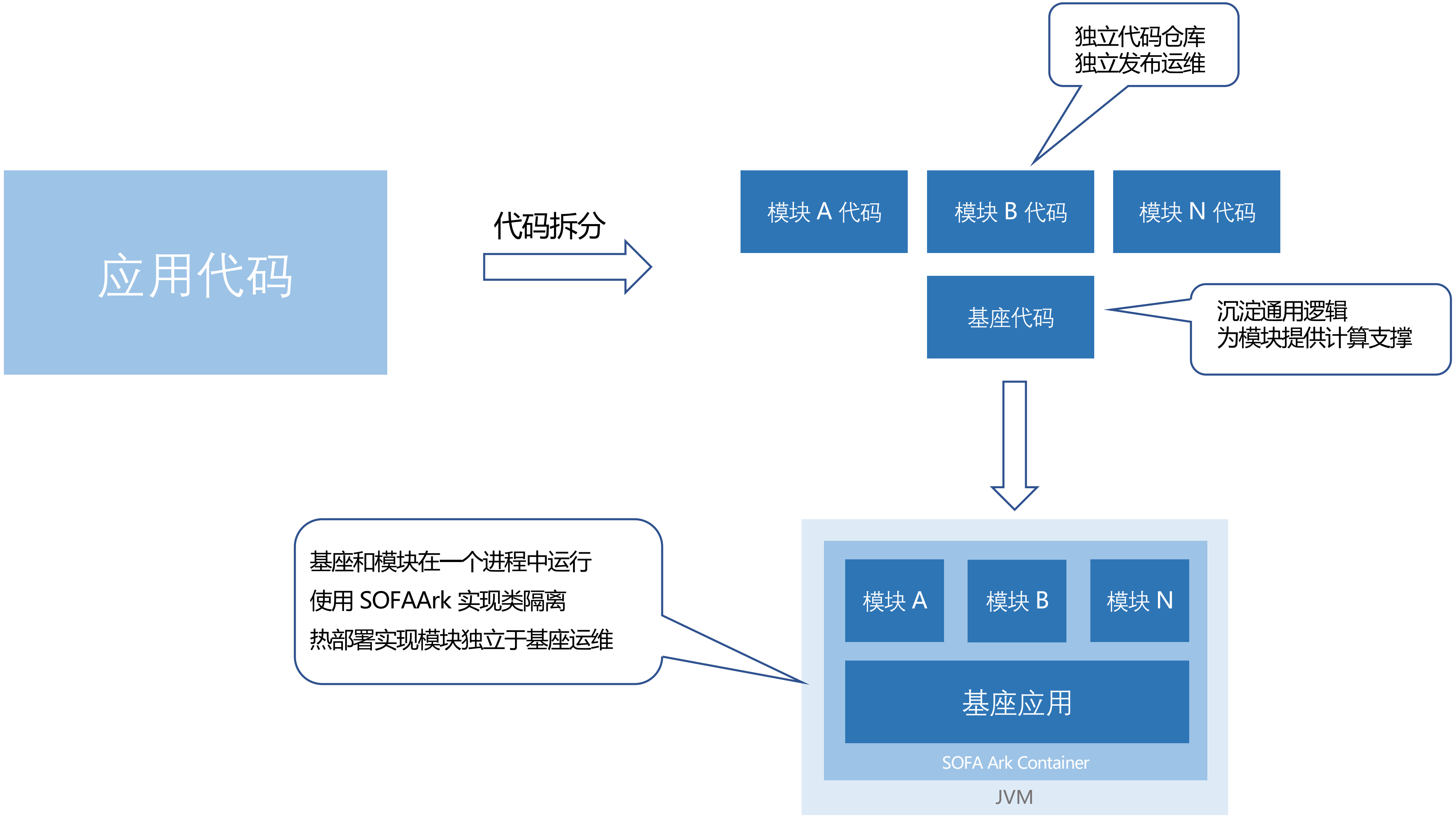


- 应用代码复杂，迭代堵车频发
- 发布流程重，交付周期漫长
- 启动速度慢，应急扩容迟缓

Serverless 的尝试

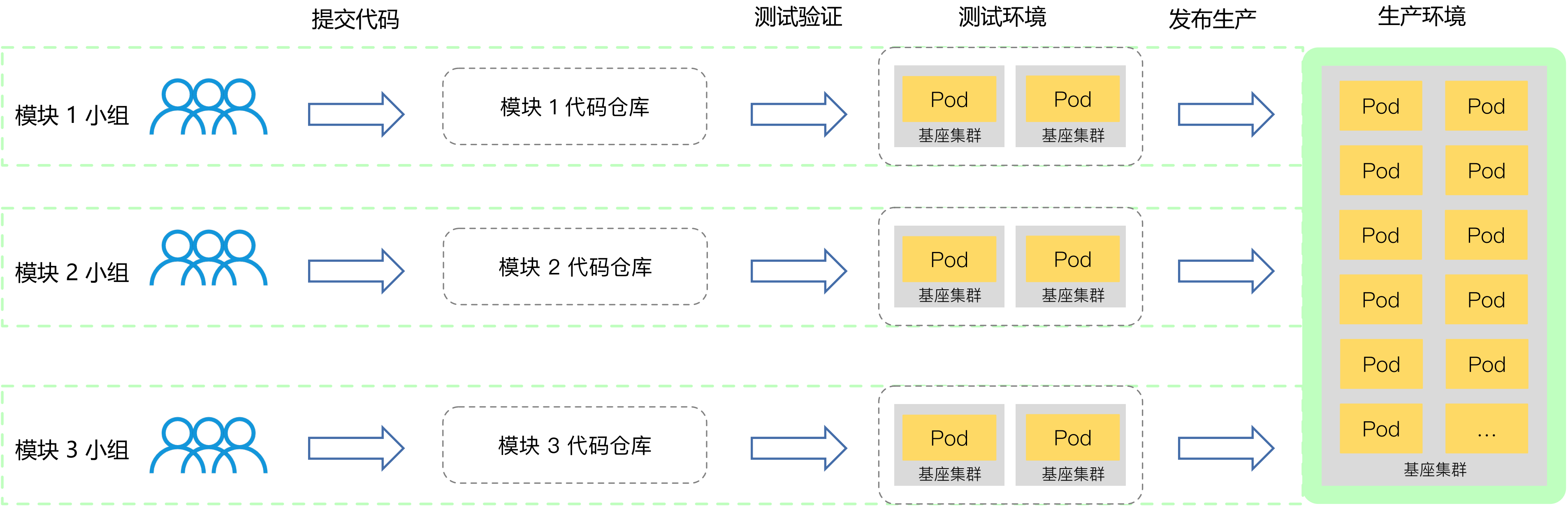


Serverless 的尝试



项目地址：https://github.com/sofastack/sofa-ark

Serverless 的尝试



收益

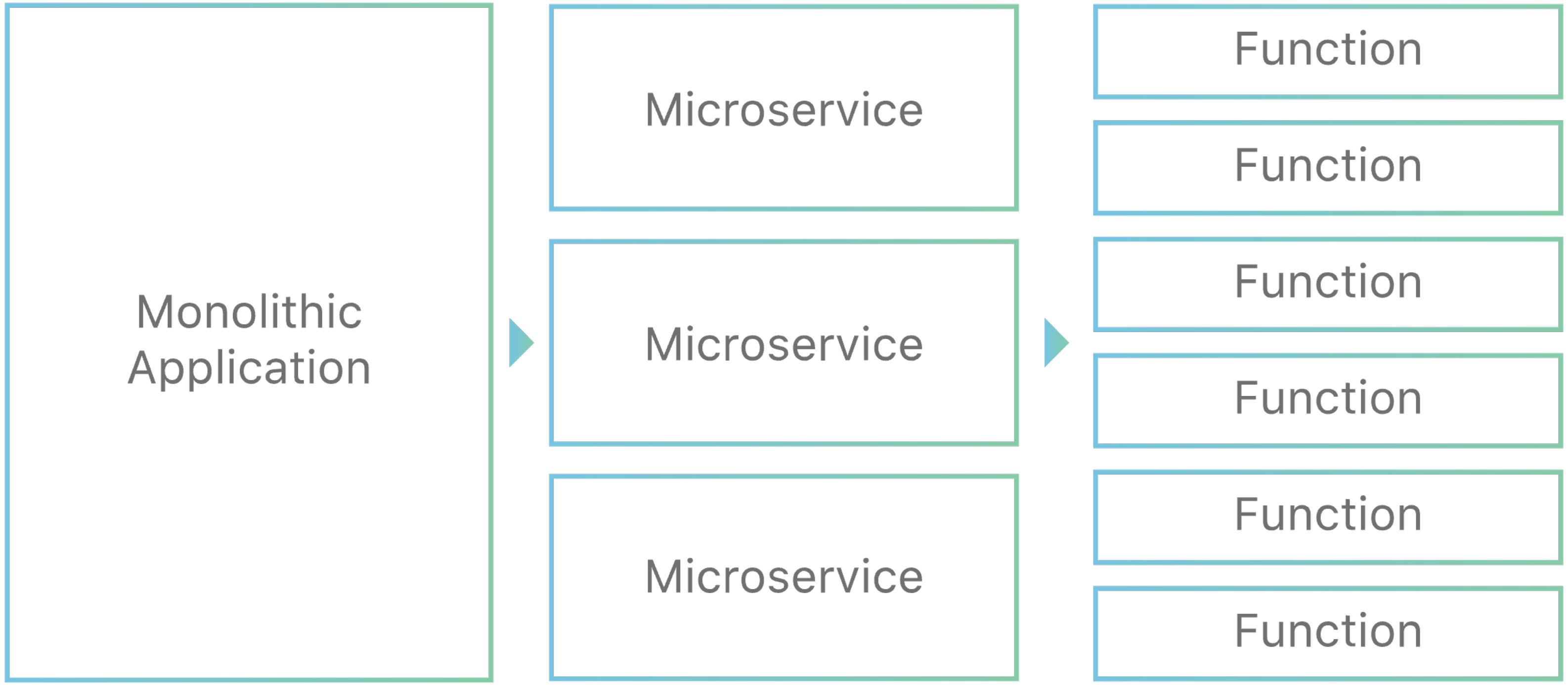
- 对模块开发者实现 Serverless，不用关注资源
- 模块发布时间 10s，具备快速迭代和快速伸缩能力
- 接入了 700+ 应用，支撑了 1W+ 生产研发迭代

不足

- 仅限 Java 应用
- 模块间隔离性较差，会互相影响
- 基座应用没有享受到 Serverless 红利

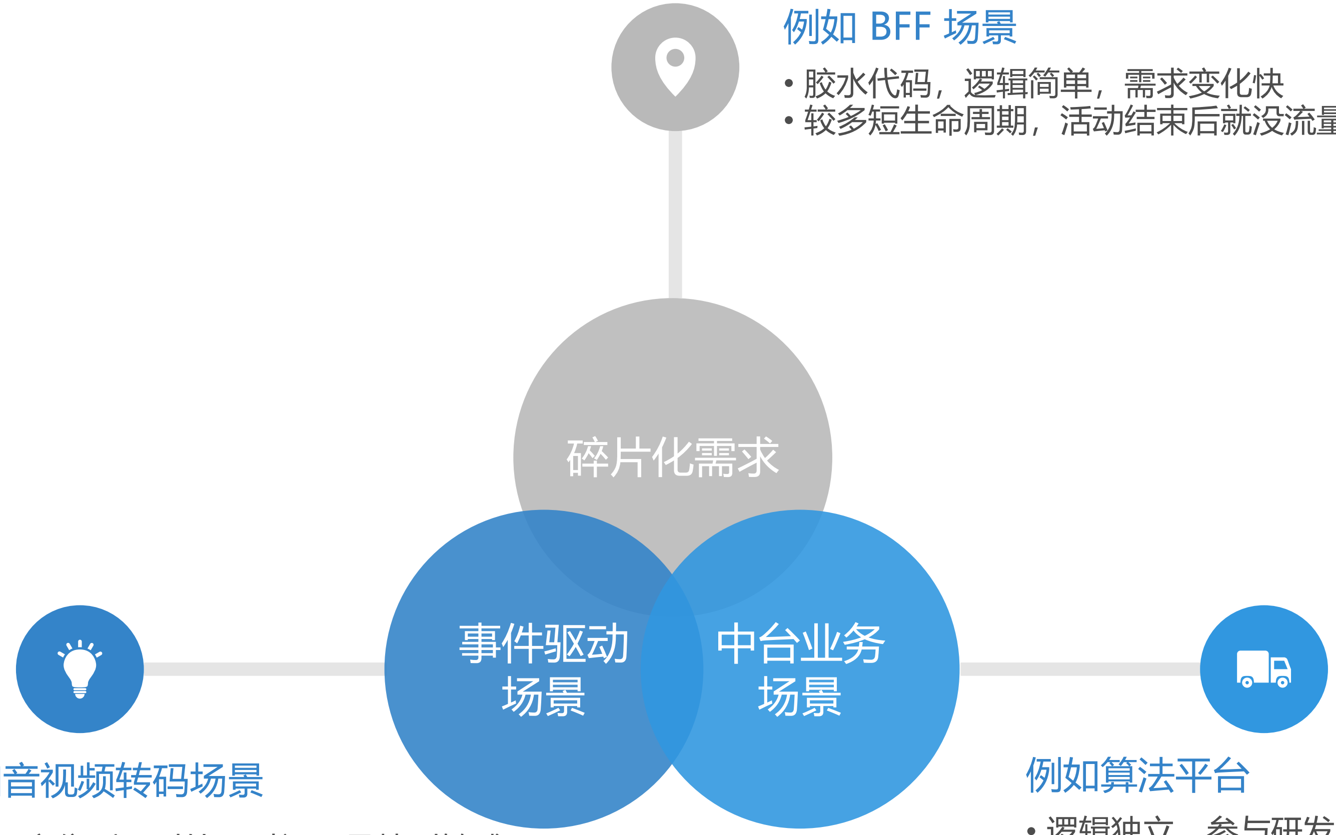
向 FaaS 迈进

应用粒度演变



图片来源：https://www.cloudflare.com/zh-cn/learning/serverless/glossary/function-as-a-service-faas/

轻量化研发场景



碎片化需求

事件驱动场景

中台业务场景

例如 BFF 场景

- 胶水代码，逻辑简单，需求变化快
- 较多短生命周期，活动结束后就没流量了

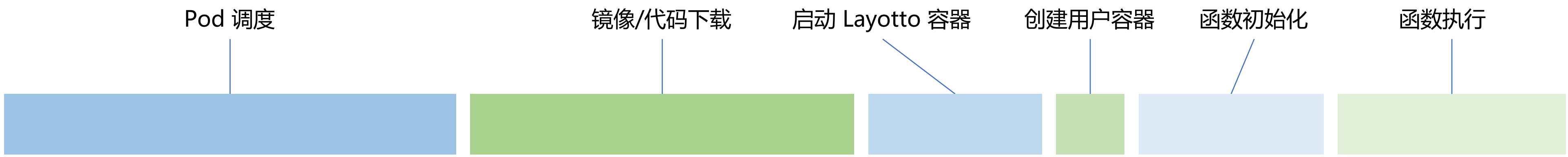
例如音视频转码场景

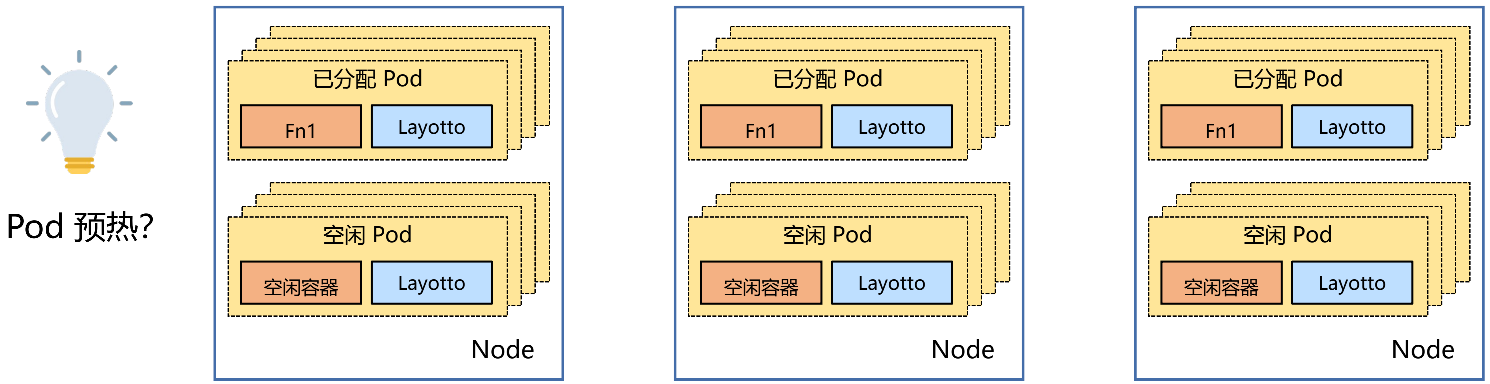
- CPU 密集型，对处理时间不是特别敏感
- 有比较明显的波峰和波谷

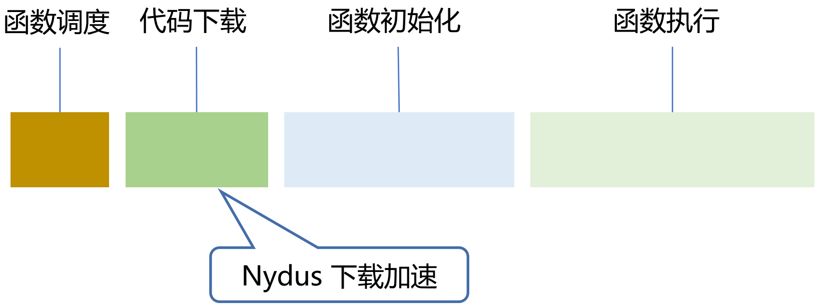
例如算法平台

- 逻辑独立，参与研发人数多
- 代码逻辑不可控，需要更好的隔离能力

如何加快启动速度?

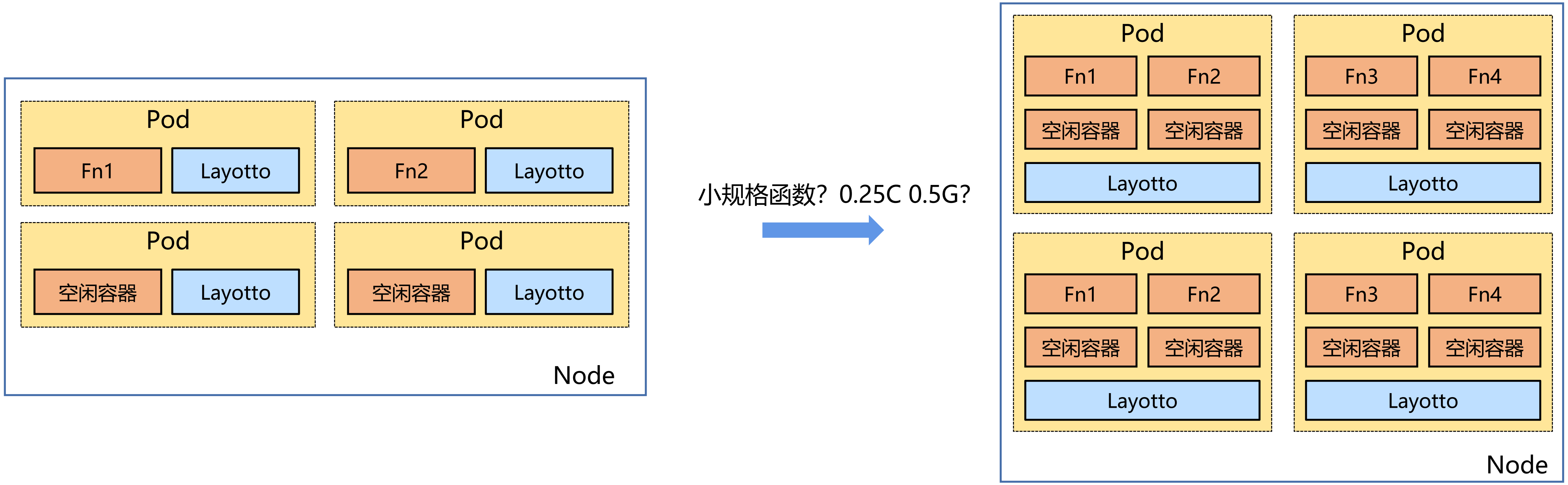




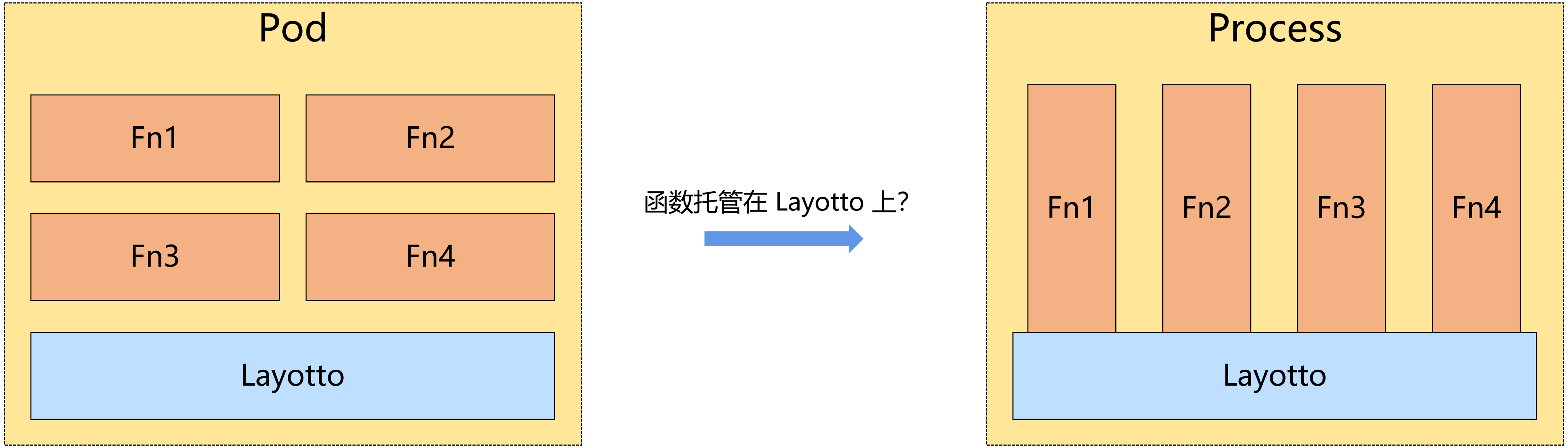


项目地址：https://github.com/dragonflyoss/image-service

降低 Layotto 资源占用



能否更进一步?


Pod
Fn1
Fn2
Fn3
Fn4
Layotto
函数托管在 Layotto 上?
Process
Fn1
Fn2
Fn3
Fn4
Layotto

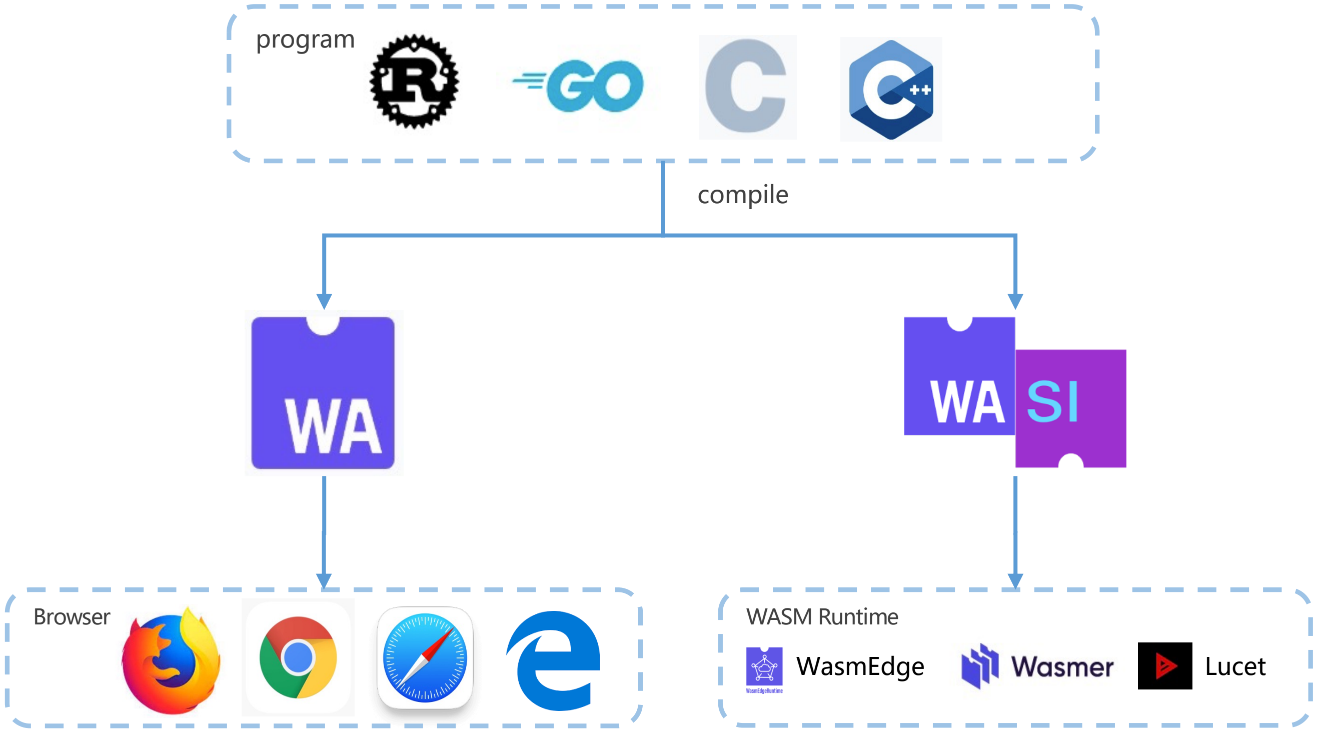
风口浪尖上的 WASM

优势

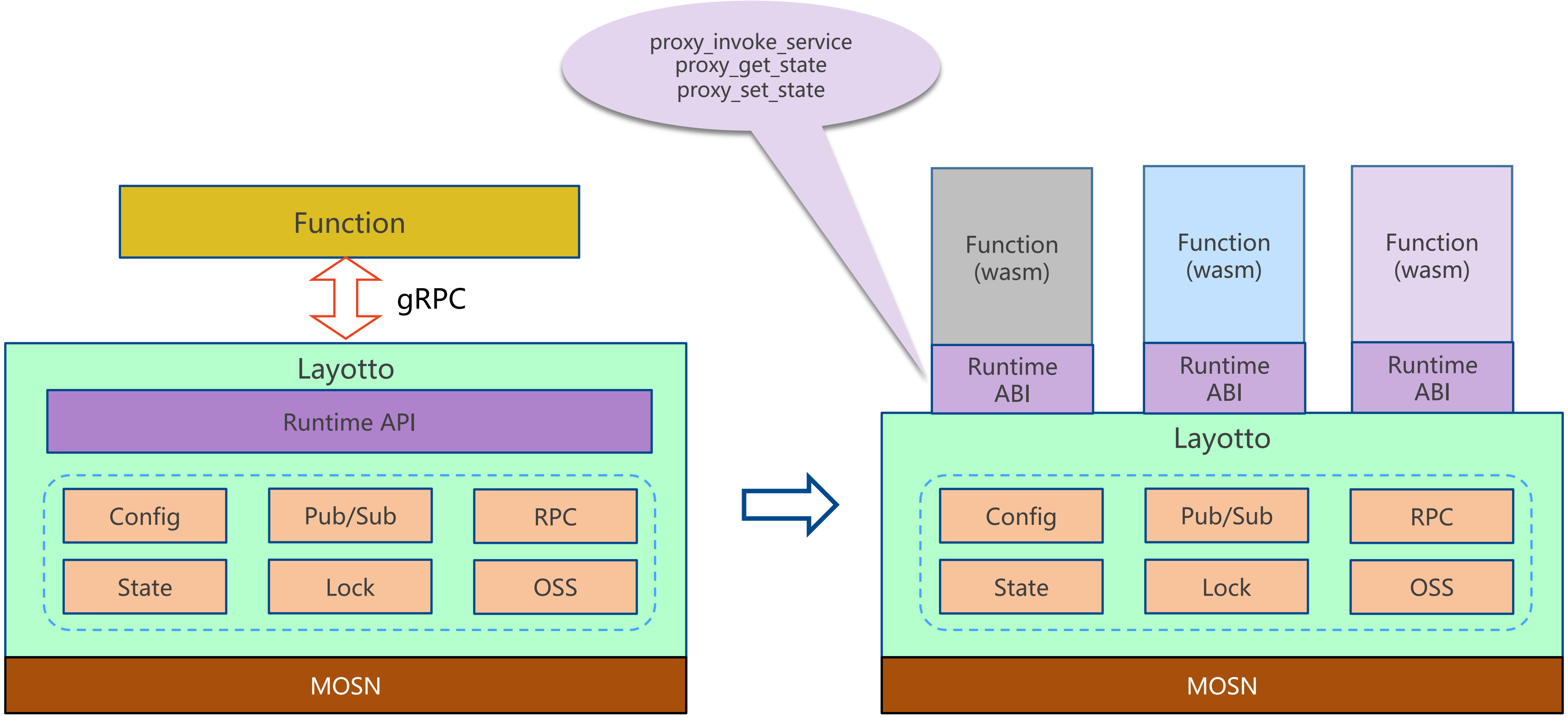
- 性能接近于 native 程序，体积小
- 沙箱执行环境，隔离性好
- 语言无关，跨平台

劣势

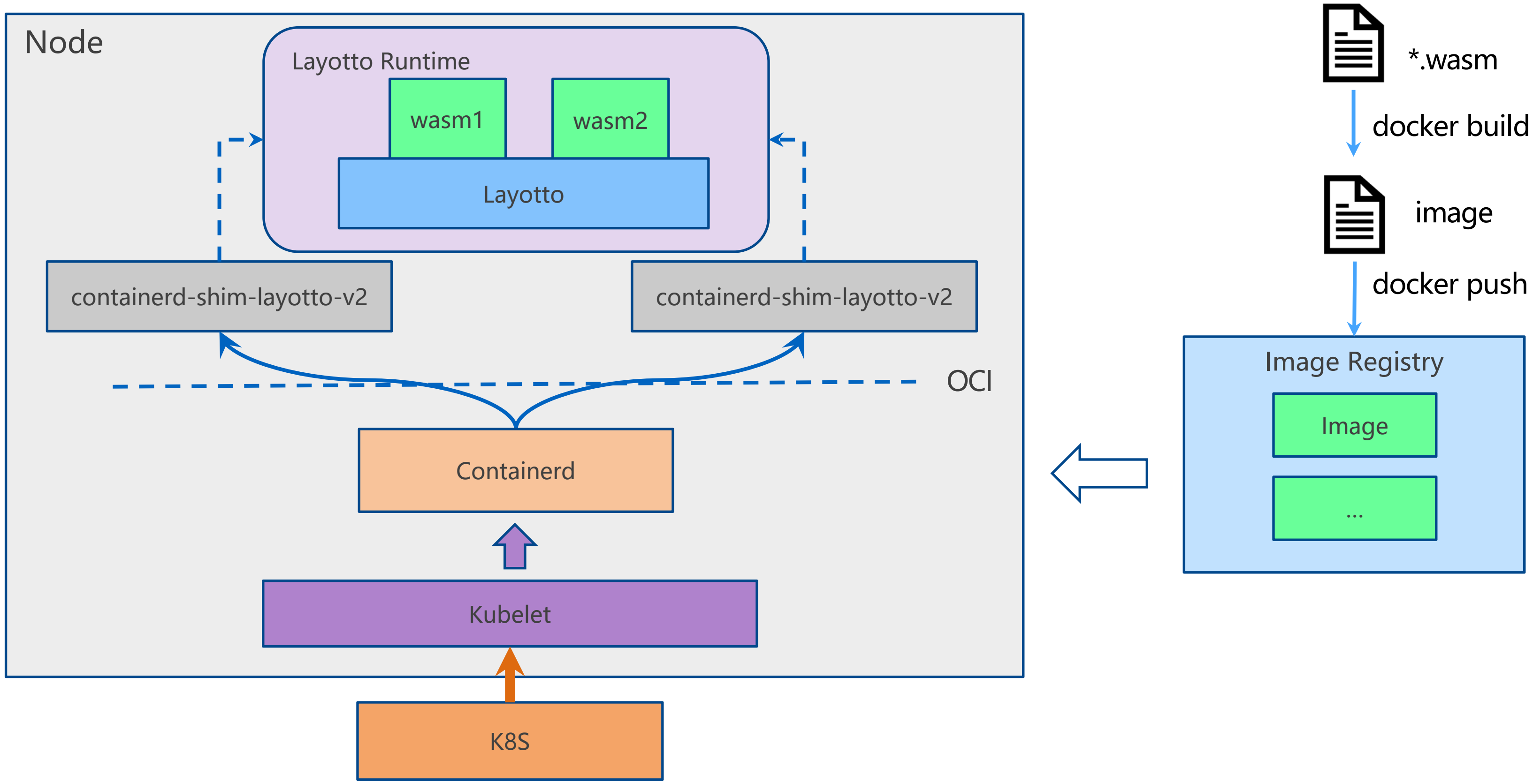
- 各语言支持程度不一
- 生态还不完善，如 debug 能力
- 各运行时能力参差不齐



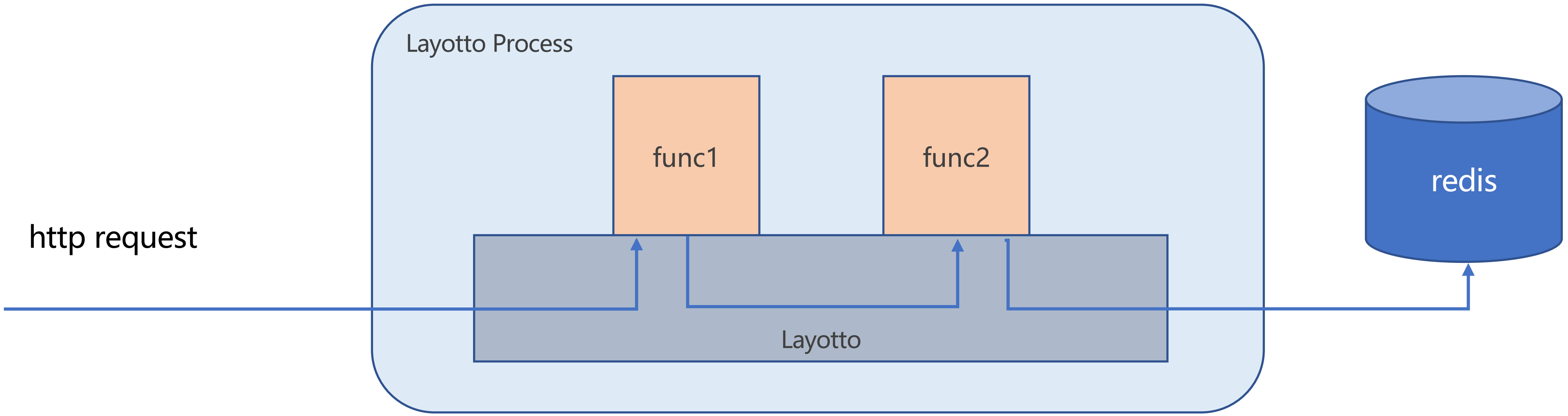
Layotto 托管函数的探索



Layotto 托管函数的探索

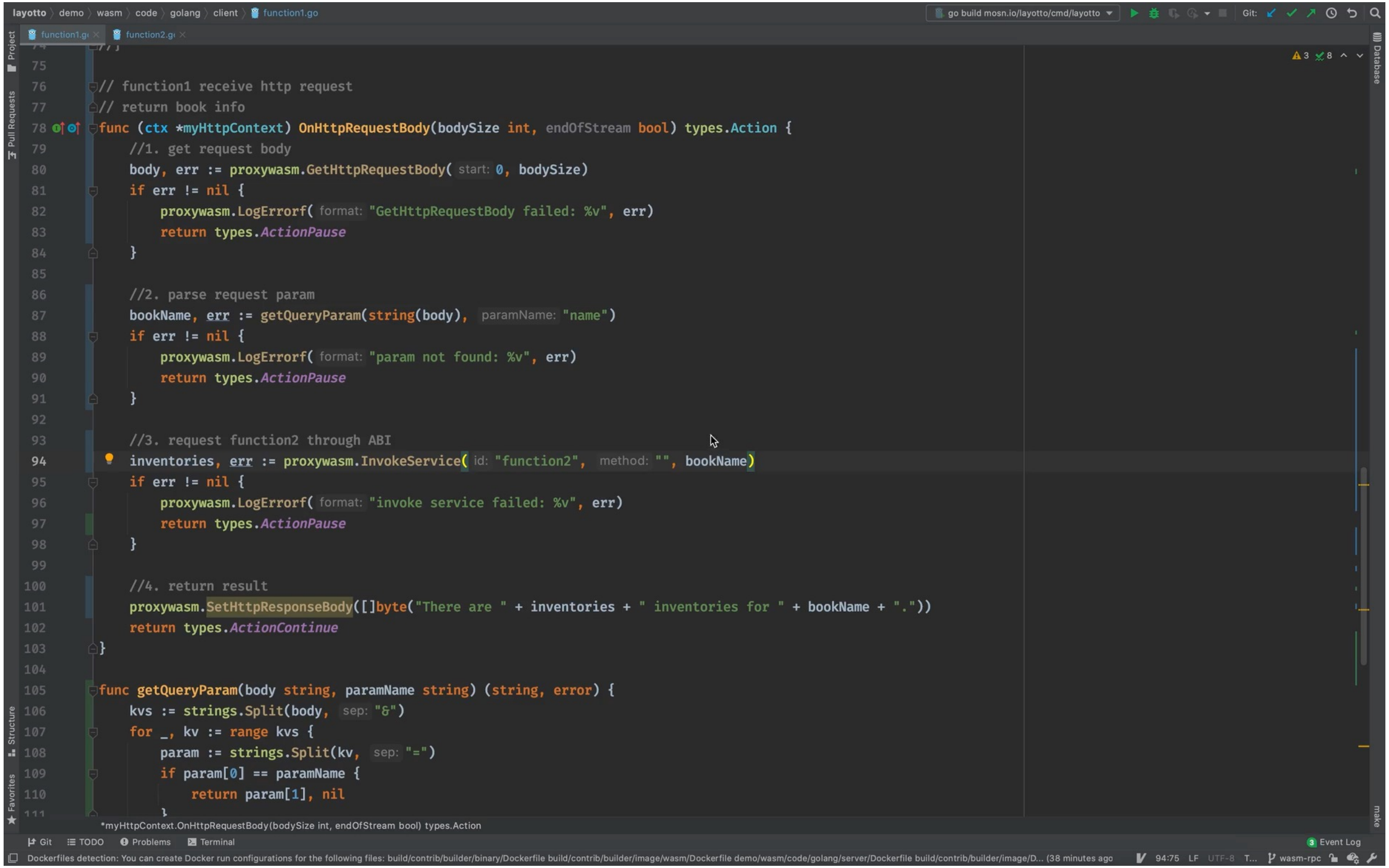


Layotto 托管函数的 PoC



PoC 说明：https://mosn.io/layotto/#/zh/start/faas/start

Layotto 托管函数的 PoC


layotto 〉 demo 〉 wasm 〉 code 〉 golang 〉 client 〉 function1.go
function1.go ×
function2.go ×
go build mosn.io/layotto/cmd/layotto
Git:
Project
Pull Requests
Structure
Favorites
Database
make
3
8

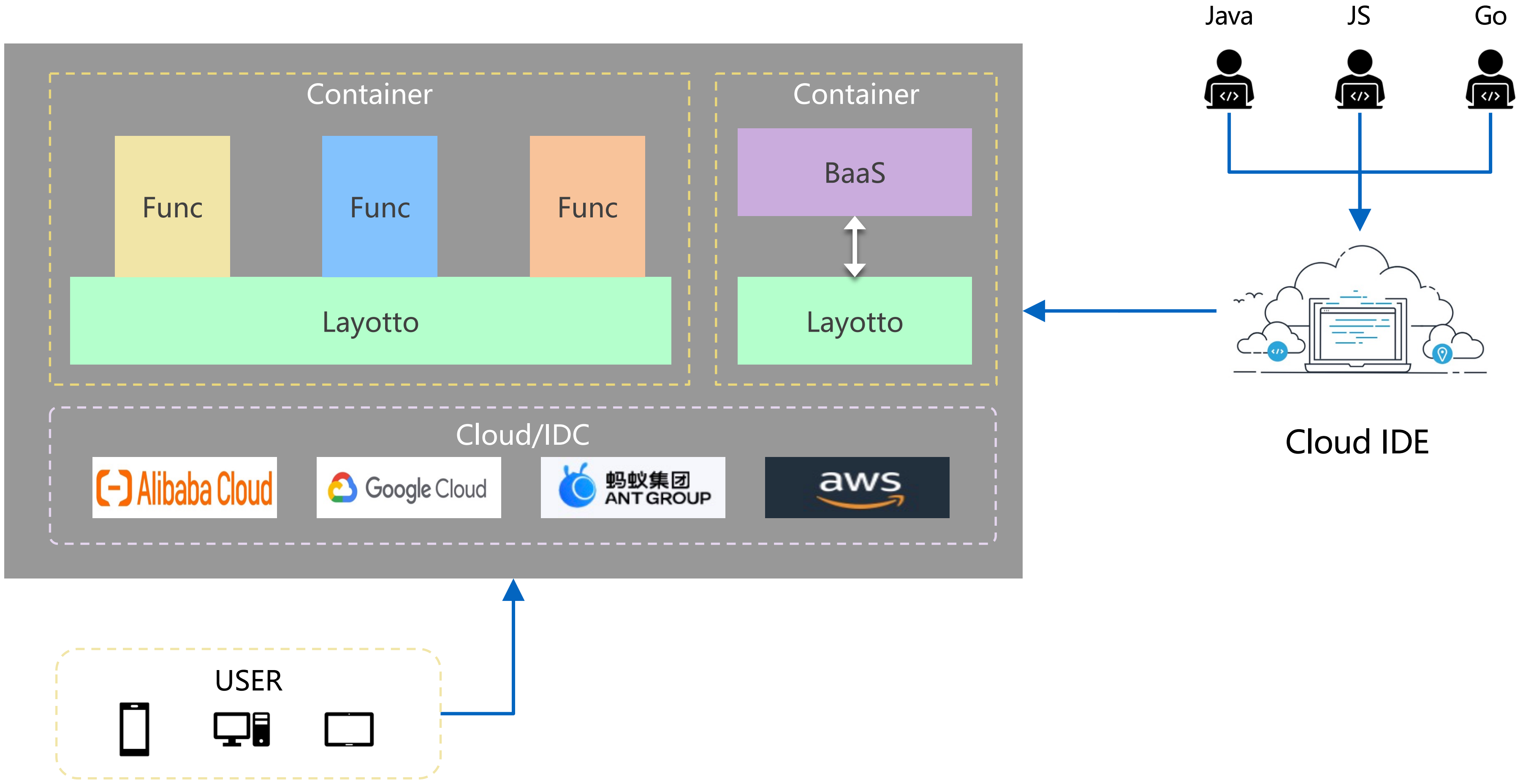
```go
//}

// function1 receive http request
// return book info
func (ctx *myHttpContext) OnHttpRequestBody(bodySize int, endOfStream bool) types.Action {
    //1. get request body
    body, err := proxywasm.GetHttpRequestBody( start: 0, bodySize)
    if err != nil {
        proxywasm.LogErrorf( format: "GetHttpRequestBody failed: %v", err)
        return types.ActionPause
    }

    //2. parse request param
    bookName, err := getQueryParam(string(body),  paramName: "name")
    if err != nil {
        proxywasm.LogErrorf( format: "param not found: %v", err)
        return types.ActionPause
    }

    //3. request function2 through ABI
    inventories, err := proxywasm.InvokeService( id: "function2",  method: "", bookName)
    if err != nil {
        proxywasm.LogErrorf( format: "invoke service failed: %v", err)
        return types.ActionPause
    }

    //4. return result
    proxywasm.SetHttpResponseBody([]byte("There are " + inventories + " inventories for " + bookName + "."))
    return types.ActionContinue
}

func getQueryParam(body string, paramName string) (string, error) {
    kvs := strings.Split(body,  sep: "&")
    for _, kv := range kvs {
        param := strings.Split(kv,  sep: "=")
        if param[0] == paramName {
            return param[1], nil
        }
```

*myHttpContext.OnHttpRequestBody(bodySize int, endOfStream bool) types.Action

Dockerfiles detection: You can create Docker run configurations for the following files: build/contrib/builder/binary/Dockerfile build/contrib/builder/image/wasm/Dockerfile demo/wasm/code/golang/server/Dockerfile build/contrib/builder/image/D... (38 minutes ago

展望未来

技术基础设施“消失不见”


Java
JS
Go
Container
Func
Func
Func
Layotto
Container
BaaS
Layotto
Cloud/IDC
Alibaba Cloud
Google Cloud
蚂蚁集团 ANT GROUP
aws
Cloud IDE
USER

感谢观看

THANK YOU