

# 003-总体设计文档

- 1. 引言
  - 1.1. 目的
  - 1.2. 定义和缩写
  - 1.3. 参考和引用
  - 1.4. 总设任务书
- 2. 产品介绍
  - 2.1. 产品名称、型号、代号
  - 2.2. 版本规划
  - 2.3. 对应需求文档
  - 2.4. 系统性能指标
  - 2.5. 关键应用场景
    - 2.5.1 云桥连接不同应用的能力
    - 2.5.2 单个VPC多应用集成
    - 2.5.3 多个VPC应用集成(SCP单集群)
    - 2.5.4 多个VPC应用集成(SCP多集群)
    - 2.5.5 公网多应用集成
    - 2.5.6 云下应用集成(私有云场景)
    - 2.5.7 OpenAPI触发场景
    - 2.5.8 Binlog触发场景
  - 2.6 友商分析和对比
    - 2.6.1 华为ROMA Connect
    - 2.6.2 集简云开放平台
    - 2.6.3 Azure Integration Services
    - 2.6.4 金蝶苍穹集成服务云
    - 2.6.5 竞品能力对比
    - 2.6.6 创新点
  - 2.6. 关键技术及解决途径
    - 2.7.1 SCC访问VPC
- 3. 硬件方案
- 4. 软件方案
  - 4.1. 方案选型
    - 4.1.1. 云桥组件选型
      - 方案一：camunda7
      - 方案二：camunda8
      - 方案三：Activiti7
      - 方案四：Flowable7
      - 方案五：JBPM7.74.1final
      - 方案选择
    - 4.1.2. 云桥网络方案(网络层互通)
      - 方案一：云桥中心端部署在SCC，云桥连接器仅在SCC部署一个
      - 方案二：云桥中心端部署在SCC，云桥连接器每个数据中心部署一个
      - 方案三：云桥中心端部署在SCC，云桥连接器每一个VPC部署一个
      - 方案四：云桥中心端部署在SCC，云桥连接器在VPC和SCC各部署一个
      - 方案选择
    - 4.1.3. 云桥互信方案(TLS层和应用层安全)
      - 方案一：AKSK+openapi\_auth验签
      - 方案二：key-auth
      - 方案三：jwt-auth
      - 方案四：hmac-auth
      - 方案选择
    - 4.1.4. 云桥代理持久化存储方案
      - 方案1：emptyDir
      - 方案2：csi
      - 方案3：nfs
      - 方案4：hostPath
      - 方案5：local
    - 4.1.5. 云桥多租户方案
      - 方案一：基于Identity实现
      - 方案二：基于SCC已有租户体系
    - 4.1.6. 云桥连接器多版本管理方案
    - 4.1.7. 应用密钥管理方案
    - 4.1.8. 云桥引擎导出器方案
  - 4.2. 软件总体架构
    - 4.2.1. 应用架构图
    - 4.2.2. 业务架构图
    - 4.2.3. 总体部署架构图
    - 4.2.4. 技术架构图

- 4.2.5.云桥双活架构
  - 4.3 系统流程
    - 4.3.1.云桥引擎核心实现
      - 4.3.1.1.分布式
      - 4.3.1.2.消息驱动
      - 4.3.1.3.运行时数据和历史数据分离
      - 4.3.1.4.RocksDB
      - 4.3.1.4.源码分析
      - 4.3.1.5.数据不能出VPC→Zeebe必须部署在VPC中，不能在公服(数据会出VPC)
    - 4.3.2.云桥连接器核心实现(应用快速上线)
    - 4.3.3 云桥连接器触发器实现
    - 4.3.4 云桥连接器执行器实现
    - 4.3.5.定时启动
    - 4.3.6.脚本处理器
    - 4.3.7.云桥连接器灰度升级
    - 4.3.8.停用流程
    - 4.3.9.流程字段对应与云桥连接器获取流程
    - 4.3.10.MySQL增量获取记录 实现原理
  - 4.4. 数据结构
    - 4.4.1. 应用模板关键数据结构
    - 4.4.2. 流程关键数据结构
  - 4.5 模块设计
    - 4.5.1云桥管理服务模块
      - 4.5.1.1.模块系统需求
      - 4.5.1.2.模块设计约束
      - 4.5.1.3.接口列表
    - 4.5.2云桥连接器服务模块
      - 4.5.2.1.模块系统需求
      - 4.5.2.2.模块设计约束
      - 4.5.2.3.接口列表
    - 4.5.3.应用模板模块
      - 4.5.3.1.模块系统需求
      - 4.5.3.2.模块设计约束
      - 4.5.3.3.接口列表
    - 4.5.4.监控&告警
      - 4.5.4.1.模块系统需求
      - 4.5.4.2.模块设计约束
      - 4.5.4.3.接口列表
    - 4.5.5.升级
      - 4.5.5.1.模块系统需求
      - 4.5.5.2.模块设计约束
      - 4.5.5.3.接口列表
  - 4.6 风险分析
- 5. 关键特性设计
  - 5.1. 业务出错处理
  - 5.2. 避免业务长时间中断处理
  - 5.3. 硬件故障的可靠性保障措施
  - 5.4. 补救措施
  - 5.5. 维护设计
  - 5.6. 安全性设计
    - 5.6.1. 威胁建模分析
    - 5.6.2. 安全设计
      - 5.6.1.1. 整体安全架构设计
    - 5.6.3. 预使用组件版本合规性情况
    - 5.6.4. 预使用组件版本漏洞情况
  - 5.7. 可靠性设计
  - 5.8. 韧性设计
  - 5.9. 性能设计
    - 5.9.1. 测试方案
  - 5.10. 可监控性设计
  - 5.11. 可调试可测试设计
  - 5.12. 系统隐私设计
  - 5.13. 可重用设计
    - 5.13.1. 技术规范落地计划
    - 5.13.2. 本版本可采用的公用代码
    - 5.13.3. 本版本可产出的公用代码
    - 5.13.4. 系统依赖解耦设计
- 6. 总结
  - 6.1. 与上一个版本的设计变化
    - 6.1.1. 设计变化项1
      - 6.1.1.1. 设计变化原因
      - 6.1.1.2. 设计变化描述

- 6.1.1.3. 设计变化产生的影响
- 6.1.2. 设计变化项2...
- 6.2. 风险问题保证
- 6.3. 尚未解决的问题
- 7. 变更控制
  - 7.1. 变更列表

# 1. 引言

## 1.1. 目的

简要介绍该说明书的目的，和使用该规格书的对象(如项目技术经理等)。

总设的输入：系统需求文档；在文档编写之前，画出一个设计的大体思路，经专家评定后，再动手写，也是一个输入。

总设的输出：模块的划分、模块间的接口、模块间的系统流程

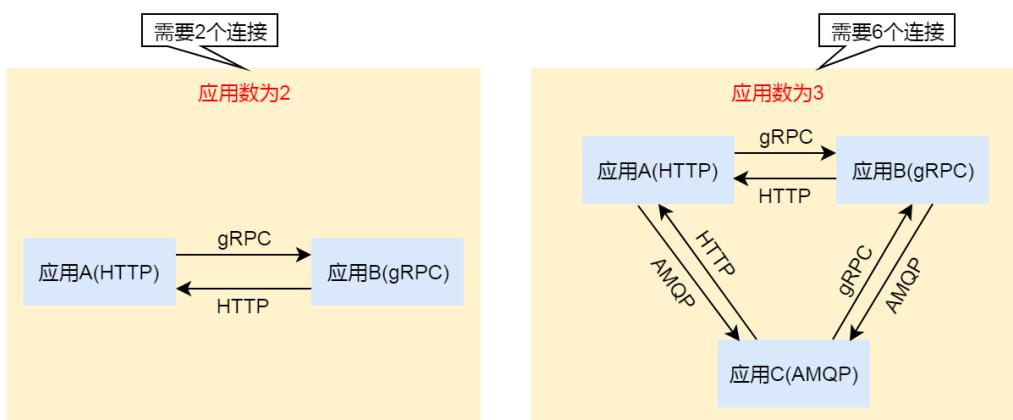
需求文档里面有的需求，都要在总体设计里面有体现。回溯责任时，如果需求文档里面有，总体设计没体现，则是总体设计人员的责任。

在医疗行业，随着数字化转型的推进，越来越多的医疗机构开始搭建自己的业务系统。医疗行业具有特殊的业务需求和复杂的业务流程，需要定制化的系统来支持和管理。然而，由于医疗行业的IT能力相对较弱，大部分机构缺乏业务开发的能力，因此他们通常会与第三方服务提供商合作，以获取定制化的服务，这里的定制化服务包含但不限于API定制和多系统集成。这种合作形式的定制化服务存在以下弊端：

- ①成本高昂，一个API定制的成本以万元为单位
- ②缺乏灵活性，第三方服务提供商进行多系统集成一般采用点对点(Point-to-Point, P2P)集成方式，添加新的集成应用，集成连接呈指数级增长；由于每个应用的交互协议是不相同的，导致在多系统集成中缺乏统一的标准和灵活性，增加了集成的复杂性和难度
- ③无法快速变化，当医疗机构的业务流程发生变化时，需要与服务提供商进行沟通、协商和重新开发，这个过程需要一定的时间和成本
- ④无法进行多云集成，比如有些医疗机构的业务应用一部分在本地，一部分在虚拟专用网络(VPC)中，一部分云端

点对点集成示意图

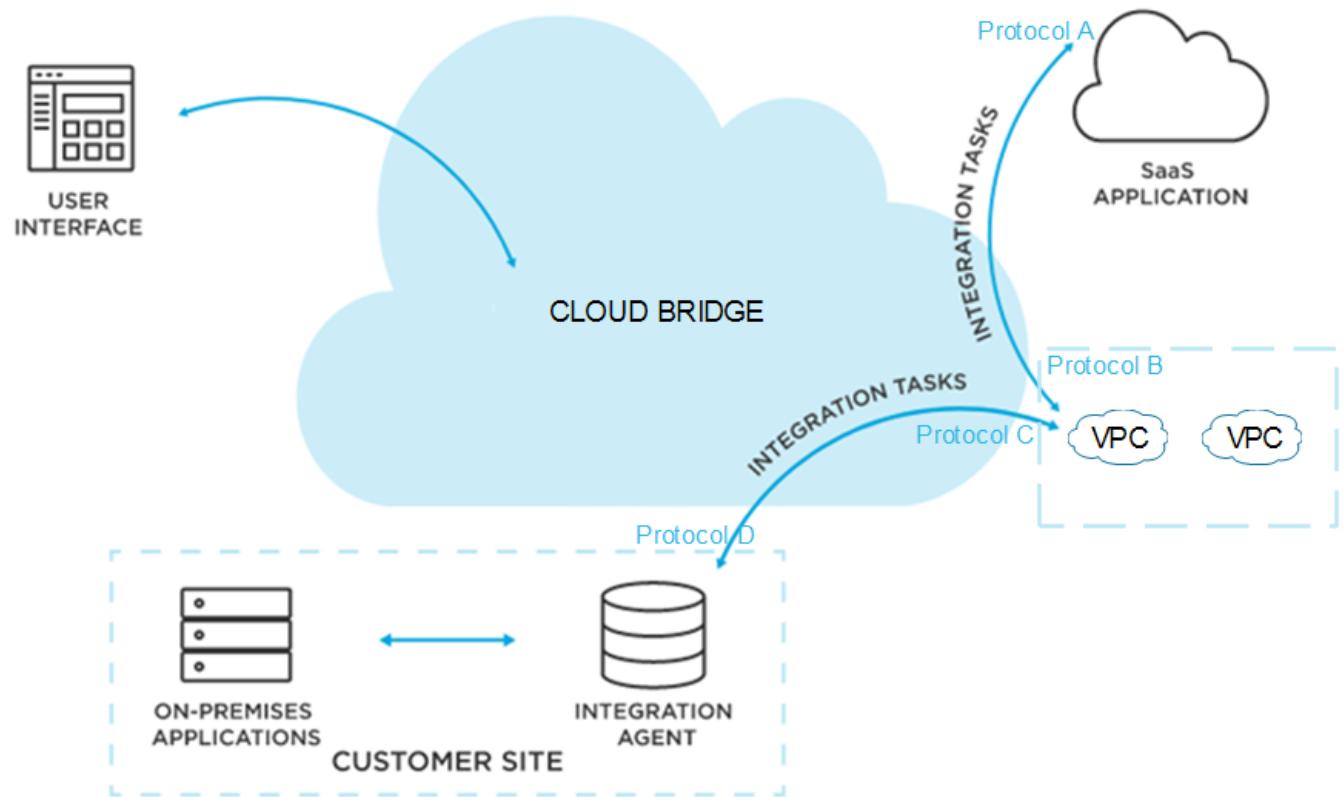
点对点集成的连接数量为： $n(n-1)$



为了解决以上的问题，推出了云桥CloudBridge。云桥CloudBridge是一种云计算服务，提供了一套低代码工具和平台，使医疗机构能够自己开发、执行和管理不同应用程序之间的集成流程或API定制。它支持通过通用的连接器将云端应用、本地应用和虚拟专用网络(VPC)的不同应用层协议的数据源连接起来，实现自动化的多系统集成和API定制，从而提高效率、简化操作并促进数字化转型。

场景描述：自动化的集成流程旨在通过云桥将不同的系统、服务或应用无缝地整合在一起，以实现数据和功能的交互和共享

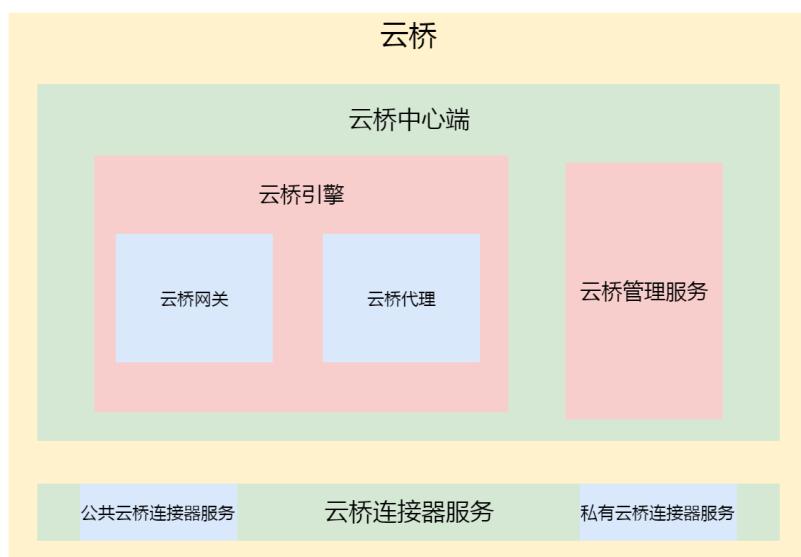
API定制旨在通过连接不同的数据源，实现数据的获取或写入，并通过云桥将这些功能暴露为API，从而实现无需搭建服务端的API开发



## 1.2. 定义和缩写

定义或解释本文档中使用的所有专用名词和缩略语。

图1 云桥概念图



缩写名	描述	备注
SCC	信服云托管云平台	

云桥	CloudBridge(CB) , 包含 <b>云桥中心端</b> <b>CloudBridgeCenter(CBC)</b> 、 <b>云桥连接器服务</b> <b>CloudBridgeConnectorService(CBCS)</b>
云桥中心端	CloudBridgeCenter(CBC) , 包含 <b>云桥引擎</b> <b>CloudBridgeEngine(CBE)</b> 、 <b>云桥管理服务</b> <b>CloudBridgeManagementService(CBMS)</b>
云桥连接器服务	CloudBridgeConnectorService(CBCS) , 分为 <b>公共云桥连接器服务</b> 和 <b>私有云桥连接器服务</b> , 公共云桥连接器服务部署在云上 , 私有云桥连接器服务部署在VPC或云下数据中心
云桥引擎	CloudBridgeEngine(CBE) , 包含 <b>云桥网关</b> <b>CloudBridgeGateway(CBG)</b> 、 <b>云桥代理</b> <b>CloudBridgeBroker(CBB)</b>
云端代理	Skyops , 私有云桥连接器服务的宿主机 , 具备灰度、热升级的功能
PaaS	Platform as a Service(平台即服务) , 是一种云产品 , 它将应用程序基础设施 ( 中间件 ) 功能作为服务提供。PaaS包括iPaaS(集成平台即服务)、aPaaS(应用平台即服务)、apimPaaS(API管理平台即服务)、fPaaS(功能平台即服务)、baPaaS(业务分析平台即服务)、物联网PaaS和dbPaaS(数据库平台即服务)。 PaaS 功能可以以提供商管理或自我管理、多租户或专用的形式提供。
iPaaS	Integration Platform as a Service(集成平台即服务) , 是一套云服务 , 支持集成流程的开发、执行和治理 , 连接单个或跨多个组织内的本地和基于云的流程、服务、应用程序和数据的任意组合。
aPaaS	Application Platform as a Service(应用平台即服务) , 是一种为应用程序服务提供开发和部署环境的云服务。
流程	使用特定的建模语言或标准来描述和定义的 <b>业务流程</b>
流程实例	一次具体的流程执行过程
触发器	启动流程实例的机制。运行在云桥连接器服务中。客户感知到的是“触发应用”。
执行器	一个可重用的构建块 , 用于执行与外部系统的集成。运行在云桥连接器服务中。客户感知到的是“执行应用”。
处理器	BPMN语言的表达 , 常见的有循环处理、逻辑判断等
BPMN	Business Process Model and Notation (业务流程模型与标记语言) , 是一种用于描述业务流程的图形化标准化表示方法。 BPMN提供了一套符号和规则 , 用于描述业务流程中的活动、事件、网关、流程流动、数据对象等元素 , 以及它们之间的关系和交互。

### 1.3. 参考和引用

列出本文档编写过程中参考的其它文档或资料

如果现有规范无法覆盖需求 , 需要组织干系方讨论需求 , 提取新的规范接口并评审 , 参考流程 : <http://code.sangfor.org/sangfor/specification>

交互地址：[https://seerdesignmg.sangfor.com/file/121605707704138?page\\_id=1043%3A065410](https://seerdesignmg.sangfor.com/file/121605707704138?page_id=1043%3A065410)

需求地址：03-系统需求

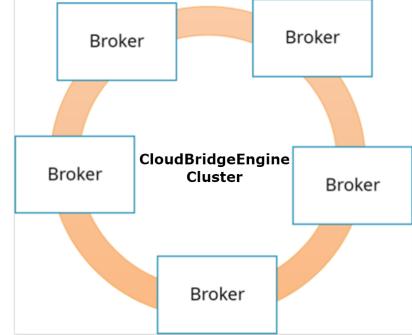
## 1.4. 总设任务书

此总体设计任务书由部分开发经理/产品架构师填写下发，说明设计需要达到的要求

编号	目标项概述	对应的标准要求
1	性能参数要求	<p>云桥管理服务API单个接口耗时小于100ms</p> <p>云桥引擎每秒启动的流程实例支持1000个(3个任务节点)</p> <p>云桥引擎单个任务节点调度小于100ms</p> <p>每个云桥连接器可以同时拉取执行任务100个</p> <p>每个云桥连接器可以同时处理作业数量20个</p> <p>每个流程最多添加50个执行节点</p> <p>每个流程最多添加100个系统变量</p> <p>OpenAPI的超时时间为1分钟，1分钟之内没有返回则报错超时</p>
2	资源开销要求	<p>说明设计需要满足的内存占用、CPU占用、磁盘占用、磁盘操作频繁程度、文件描述符、网络传输数据量资源等要求</p> <p>云桥管理服务 <math>\text{cpu} &lt; 0.2c</math>，内存占用 <math>&lt; 200\text{MB}</math></p> <p>云桥引擎容器一般业务执行场景，<math>\text{cpu} &lt; 1c</math>，内存占用 <math>&lt; 2\text{GB}</math>，引擎磁盘占用 <math>200\text{GB}</math>，导出器(Elasticsearch)磁盘占用 <math>200\text{GB}</math></p> <p>云桥连接器容器一般业务执行场景，<math>\text{cpu} &lt; 0.5c</math>，内存占用 <math>&lt; 500\text{MB}</math></p>
3	稳定性	<p>描述设计需要达到的稳定性要求，比如崩溃恢复时间、最高压力时平均可持续运行时间、需要检测程序是否崩溃出错手段等要求</p> <ul style="list-style-type: none"><li>服务进程：云桥所有容器服务需要配置Readiness(就绪探针)、Liveness(存活探针)探测程序，容器故障需要重新拉起</li></ul>

容器名称	readiness Probe	livenessProbe
CloudBridgeManagementService	periodSeconds : 10s timeoutSeconds : 15s	periodSeconds : 60s timeoutSeconds : 15s
CloudBridgeConnectorService	periodSeconds : 10s timeoutSeconds : 15s	periodSeconds : 30s timeoutSeconds : 15s
CloudBridgeBroker	periodSeconds : 30s timeoutSeconds : 10s	periodSeconds : 30s timeoutSeconds : 30s failureThreshold : 120
CloudBridgeGateway	periodSeconds : 30s timeoutSeconds : 10s	periodSeconds : 30s timeoutSeconds : 30s failureThreshold : 20

- **云桥代理**多节点部署，一个k8s-pod一个节点，每个节点一个分区，statefulset部署形式，节点之间形成一个点对点网络，可以自由横向扩展，支持1-n个节点，一个代理不可用后，该代理的任务会被透明的重新分配到网络中其他代理，单节点故障不会影响集成能力。生产环境建议配置4节点、4分区、1复制因子



- **云桥代理**亲和性方面，优先将pod分散到不同的k8s节点上，以提高系统的稳定性和容错性

```

affinity:
  podAntiAffinity:
    preferredDuringSchedulingIgnoredDuringExecution():
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchExpressions:
              - key: "app.kubernetes.io/component"
                operator: In
                values:
                  - cloudbridge-broker
                    topologyKey: "kubernetes.io/hostname"

```

- **云桥引擎**会处理大量的集成任务，保证在SCC升级期间客户业务正常运行
- **公共云桥连接器服务**部署在云上，滚动升级，升级期间触发器执行器可以正常运行，流程实例任务节点正常流转
- **私有云桥连接器服务**部署在客户VPC中或云下经典网络，滚动升级，升级期间触发器执行器可以正常运行，流程实例任务节点正常流转

```

strategy:
rollingUpdate:
maxSurge(最大额外副本数): 1
maxUnavailable(允许的最大不可用副本数): 0
type: RollingUpdate

```

- **云桥连接器服务**处理完任务后，如果上报云桥引擎出现异常导致上报失败，应该会有重复上报的机制，在设定的时间内(5分钟)全部上报失败，云桥引擎会将该作业重新分配给云桥连接器服务

4	架构要求	<p>描述总体架构设计要达到的要求，比如低耦合、可扩展等方面的具体要求，需要给出具体的要求和建议</p> <ul style="list-style-type: none"> <li>• 云桥管理服务以容器方式运行在标准K8S上，满足云原生开发方式：       <ul style="list-style-type: none"> <li>• 使用HTTP协议，对外暴露统一使用apisix ingress方式注册</li> <li>• 内部服务之间服务发现使用K8S的service和endpoint</li> </ul> </li> <li>• 云桥引擎以容器方式运行在标准K8S上，满足云原生开发方式：       <ul style="list-style-type: none"> <li>• 使用gRPC协议，对外暴露统一使用apisix ingress方式注册</li> <li>• 内部服务之间服务发现使用K8S的service和endpoint</li> </ul> </li> <li>• 低耦合方面       <ul style="list-style-type: none"> <li>• 云桥引擎负责统一的流程任务调度，不感知云桥管理服务和云桥连接器的逻辑</li> <li>• 云桥连接器服务采用标准的应用模型交互协议，不感知数据源的业务逻辑</li> <li>• 云桥连接器服务作为jobworker独立于云桥引擎存在，云桥引擎只负责产生工作；云桥连接器定期请求某种类型的工作，并在执行作业之后为每个工作返回结果</li> </ul> </li> <li>• 可扩展方面       <ul style="list-style-type: none"> <li>• 云桥连接器服务包含触发器和执行器，支持自定义拓展，云桥连接器服务支持分布式管理</li> </ul> </li> </ul>
5	可维护性	<p>提供给维护人员的工具或手段</p> <ul style="list-style-type: none"> <li>• 云桥管理服务相关容器保证其只与业务有关，单独提供debug镜像携带各类调试工具如：网络io、抓包、trace等，使用kubectl debug进行排障</li> <li>• 云桥管理服务日志满足云BG日志规范，日志需要添加错误码，排障码，支持USR1信号触发DEBUG模式</li> <li>• 提供所有流程实例的监控工具，同时支持对失败的流程进行重试、取消等操作</li> <li>• 提供云桥连接器服务的健康状态检测的机制，云桥中心端需要实时感知所有云桥连接器服务的状态(由连接器主动上报)</li> <li>• 服务的配置文件使用configmap方式挂载到pod内部</li> <li>• 接入Prometheus和Grafana来监测集群拓扑的健康度、吞吐量指标、已处理的请求、每秒导出的事件、磁盘和内存使用情况等。</li> </ul>

6	可测试性	<p>说明系统给功能、压力、性能测试及自动化测试提供的机制和方法。比如提供统一的模块内部状态获取及调试的接口，制订调试接口规范。</p> <ul style="list-style-type: none"> <li>• 可调试性           <ul style="list-style-type: none"> <li>• 使用nocalhost作为基于K8S开发调试方式</li> <li>• 云桥管理服务提供cli工具，以快速进行接口调试</li> <li>• 云桥引擎使用标准的gRPC协议，支持grpcurl快速调试，同时可以使用zbctl进行调试</li> </ul> </li> <li>• 可操作性           <ul style="list-style-type: none"> <li>• 云桥管理服务提供API接口，采用swagger2.0和openapi 3.0声明的yaml格式定义</li> <li>• 云桥引擎提供gRPC接口，提供protobuf定义文件（模块内使用）</li> </ul> </li> <li>• 可观察性           <ul style="list-style-type: none"> <li>• 日志规范：日志需要遵守云BG日志规范，调试日志需要有动态开关</li> <li>• 运行状态可视：对于有状态服务，提供API接口获取内部状态</li> </ul> </li> <li>• 可分解性           <ul style="list-style-type: none"> <li>• 模块应该以微服务架构为主，通过标准API进行调用，方便mock测试</li> <li>• 封装对环境的依赖，通过mock及环境模拟机制可以独立运行</li> </ul> </li> <li>• 性能、压力测试           <ul style="list-style-type: none"> <li>• 提供mock形式的云桥连接器服务，结合JMeter批量启动流程实例，用于<a href="#">云桥引擎</a>的压力测试</li> <li>• 提供mock形式的云桥引擎服务，用于<a href="#">云桥连接器服务</a>的压力测试</li> <li>• 可以通过JMeter进行<a href="#">云桥管理服务API</a>压力测试</li> </ul> </li> <li>• 可控制性           <ul style="list-style-type: none"> <li>• 任务节点可以自定义重试次数、重试间隔时间、超时时间。</li> <li>• 针对失败的流程实例提供从失败任务节点进行重新执行的机制</li> <li>• 流程构建时提供连通性检测的功能，前置判断平台到目标应用网络是否可达</li> </ul> </li> <li>• 简单性           <ul style="list-style-type: none"> <li>• 架构统一，云上SaaS场景、云上VPC场景、云下场景采用统一的设计方案</li> <li>• 前后端杜绝定制化编码，可视化建模采用后台返回的模板进行渲染，后续上架新的处理器、触发器、执行器时，无需改动到前端</li> <li>• 避免过多的分支，尽量统一出入口</li> </ul> </li> </ul>
7	可扩展性	<p>说明系统在哪些方面需要保留可扩展的能力，并指明扩展方式</p> <ul style="list-style-type: none"> <li>• 云桥连接器服务提供统一的Inbound和Outbound机制，支持后续自由扩展新的连接器，需要在连接器进程中新增处理方法</li> <li>• 云桥引擎K8S管理，之后后续根据业务量自由横向扩展</li> </ul>

8	兼容性	<p>允许升级配置的版本、允许对接的版本</p> <ul style="list-style-type: none"> <li>• 云桥连接器做多版本管理，云桥管理服务和云桥引擎兼容所有版本的连接器</li> <li>• 云桥升级需要兼容所有客户已经构建好的自动化流程</li> <li>• 云桥支持脚本处理器，支持python、java等常用语言</li> <li>• 后续支持客户自定义触发器或执行器，以插件的形式调用</li> </ul>
9	升级	<p>对升级场景进行额外说明</p> <ul style="list-style-type: none"> <li>• <b>私有云桥连接器服务</b>部署在客户网络中，采用灰度升级的形式，支持热升级，云桥中心端负责将新版本推送至harbor制品仓库，由<b>私有云桥连接器服务</b>宿主机(Skyops)升级服务负责统一升级</li> </ul>
10	安全性	<p>设备具备的防范用户恶意攻击的能力（如无防范，恶意攻击可能导致本设备宕机、拒绝服务）。</p> <ul style="list-style-type: none"> <li>• 业务容器不允许以root权限启动</li> <li>• 部署在客户侧的云桥连接器与云桥中心端通信全部采用TLS加密，并通过互信机制进行认证</li> <li>• 云端代理升级的请求与拉包请求，全部需要经过互信认证</li> <li>• 云桥管理服务对外提供客户服务，所有请求需要通过平台互信认证与权限校验</li> <li>• 公服区只放开VPC访问443端口，统一代理到SCC的waf，需要做负载均衡，防止把一个APISIX打崩</li> <li>• 使用到的开源组件必须通过SDL流程，不允许使用GPL协议组件</li> </ul>
11	可重用性	<p>当前版本可采用的公用代码、可产出的公用代码要求版本经理和开发经理、预研部门一起识别和定义产出。</p> <ul style="list-style-type: none"> <li>• 构建连接器通用的Inbound和OutBound框架，便于后续快速开发新的连接器</li> <li>• 部署在云上客户VPC里的VM后续可以部署其他应用，进行其他活动</li> <li>• 部署在每个SCP的VPC—SCC的公服网络，提供了一个通用的网络通路，后续其他应用同样可以采用这条通路实现VPC内部到SCC的请求</li> <li>• 针对云下数据中心的连接器搭建任务，可以复用之前已经部署的云端代理，不需要重新构建网络通路</li> </ul>
11	可分析定位能力	<ul style="list-style-type: none"> <li>• 所有的异常均有排障码辅助快速定位问题。</li> <li>• 不可出现系统异常等笼统性日志。</li> </ul>
12	其它要求	<p>所有服务不可以主动调用云桥连接器服务，调用必须由云桥连接器服务主动发起</p> <p>云桥连接器主要开发语言：go1.19</p> <p>不可以使用Operate、tasklist、Optimize、Identity组件</p>

## 2. 产品介绍

### 2.1. 产品名称、型号、代号

IPaaS云桥1.0

### 2.2. 版本规划

迭代时间	24.07.30	迭代二
迭代方向	用于促进托管云在医院行业的推广。旨在通过云桥的能力减少医院(特别是二级医院)在接口定制方面的成本，从而使医院选择将业务部署到托管云上或使用托管云提供的云桥服务。	进一步开放连接器的能力，实现符合信服云客户模式的较为全面的多云集成
迭代目标	支持开放通用执行器能力，连接客户云上VPC内部的应用和公网SaaS应用，使客户可以低码构建较为复杂业务流程。有：HTTP、RabbitMQ、Mysql、SQLServer(仅支持模板)  支持开放多种触发器的能力，支持客户进行接口定制，支持自动触发的业务流程。有：OpenAPI、定时器、RabbitMQ、Binlog(仅支持模板)、SQLServer(仅支持模板)  支持通用的处理器的能力，在构建流程时可以自由使用。有：分支判断、循环处理、并行处理、延时、终止、异常捕获  支持统一的管理服务，支持流程管理、模板管理、应用管理、变量管理、运行日志的功能。	支持更多通用的触发器与执行器，支持集成云下场景，可以为4000+私有云客户提供集成服务与API定制的能力，全面吸引客户使用云桥服务
关键价值	客户可以构建SaaS应用和VPC内部应用的集成流程，支持API定制	多云集成

### 2.3. 对应需求文档

列举本总体技术方案对应的需求文档

触点	客户场景

### 2.4. 系统性能指标

类似于以下表格的系统性能指标

参考：[003-总体设计文档](#)

领域	指标	指标说明	备注	是否对外	迭代指标
云桥管理API耗时	<100ms	API业务耗时需小于100ms，除去网关耗时			

云桥引擎每秒触发的任务数	1000个	标准：每个流程有5个任务节点	有背压机制，客户端请求数超过背压引擎会拒绝客户端请求，包括触发新的流程实例。背压是针对所有客户端请求的，连接器请求任务、完成任务、启动新的流程实例，云桥管理服务API启动流程实例均会占用背压。		
云桥引擎任务调度时间	<100ms				
支持云桥连接器进程数量	100个				

## 2.5. 关键应用场景

描述系统需求里定义的整体用户场景，说明哪些是影响系统设计的关键用户场景。在方案选型时需保证方案能满足这些用户场景。

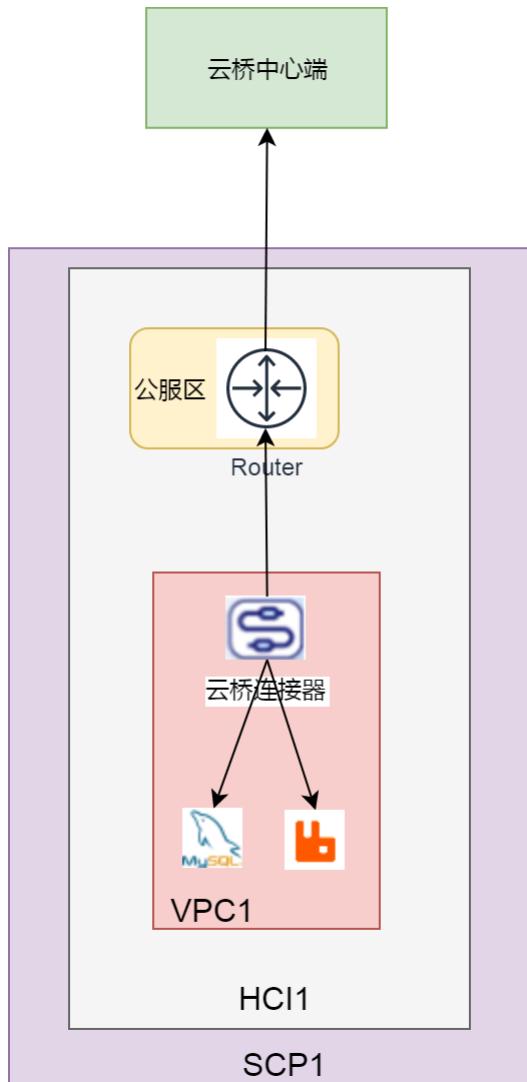
云桥的关键能力是通过通用的云桥连接器连接不同网络场景下的不同应用，以实现多应用集成与API定制，下面将针对不同网络场景下的不同应用的连接展开详细的场景描述：

### 2.5.1 云桥连接不同应用的能力

云桥作为一个集成平台，需要连接不同应用层协议的应用，所以云桥必须具备多协议支持的能力。无论是基于REST、MQTT、AMQP、SOAP等传统的通信协议，还是面向微服务架构的gRPC、GraphQL等新兴协议，云桥都应该能够灵活地适配和处理。通过支持多协议，云桥能够实现不同应用之间的数据交换、功能调用。

### 2.5.2 单个VPC多应用集成

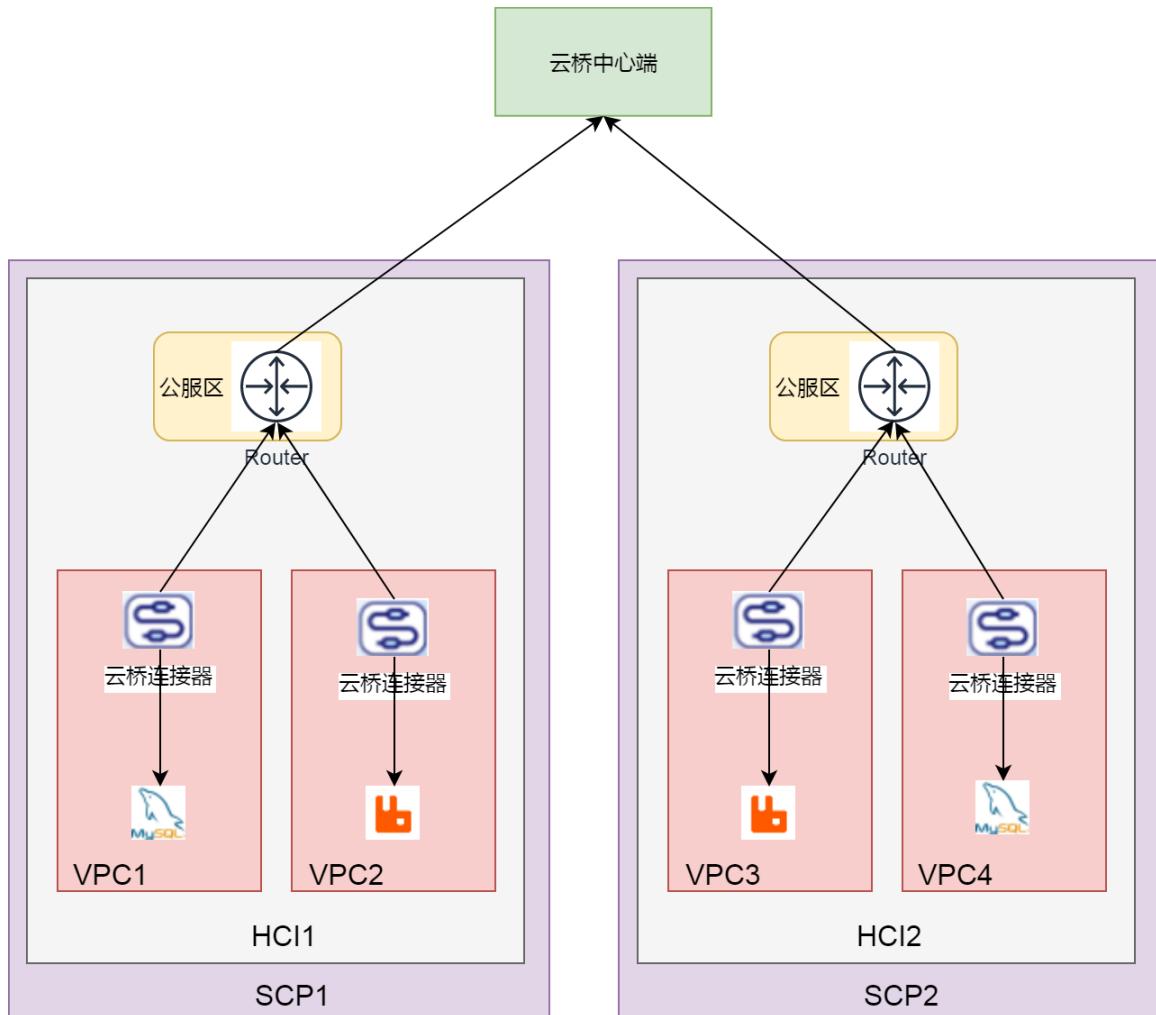
需求场景描述：客户的业务部署在VPC网络中。客户需要实现单个VPC内的多应用集成。**当mysql新增、变更或删除数据时，可以触发写入RabbitMQ的动作。**



要求：需要部署云桥连接器到客户VPC中，VPC所在SCP需要部署一个公共服务网络，公共服务网络中部署一个代理(双网卡)。保证客户VPC和公服代理网络连通，公服代理和SCC网络连通。

### 2.5.3 多个VPC应用集成(SCP单集群)

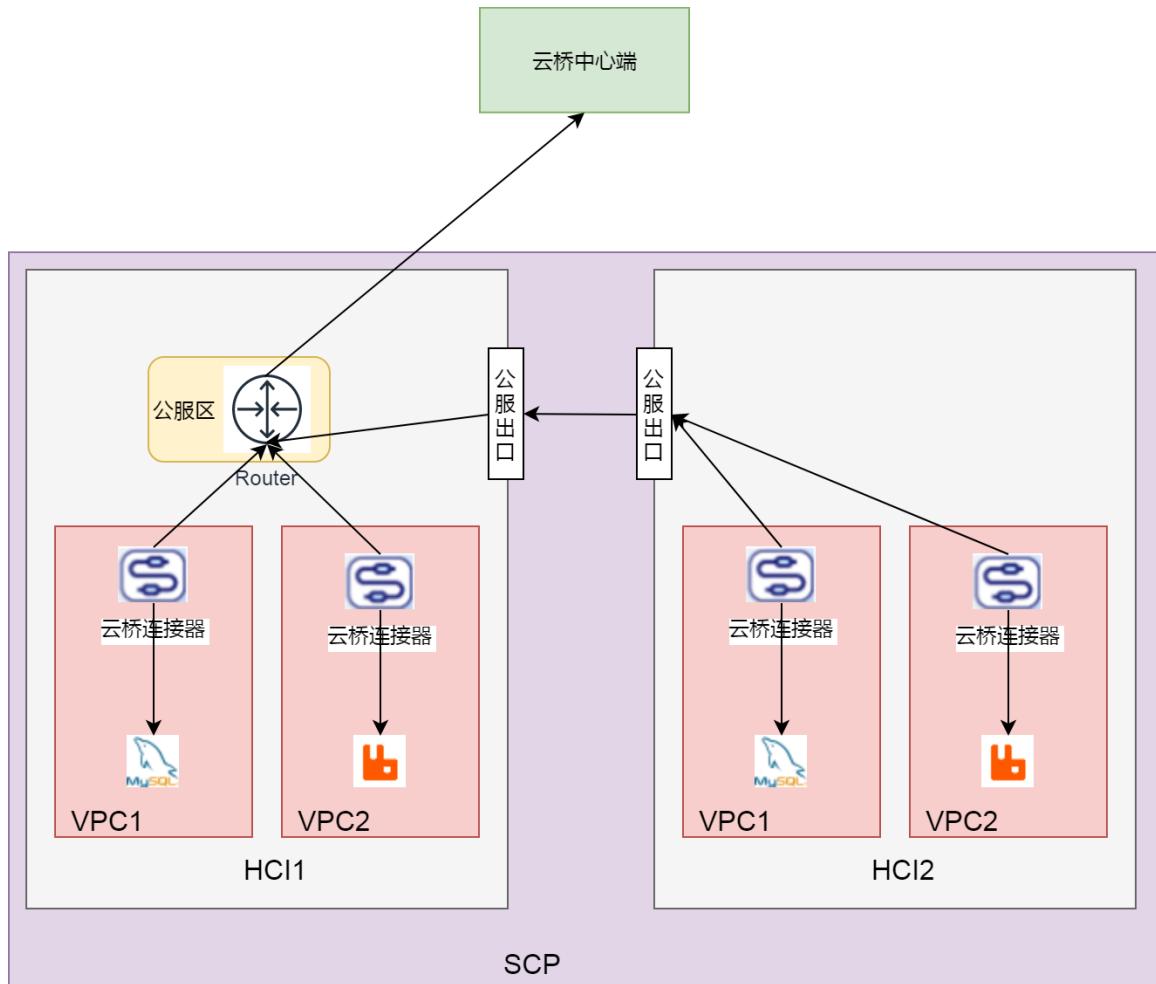
场景描述：客户的业务部署在多个VPC网络中，VPC所在SCP单集群，客户需要实现多个VPC内的多应用集成。



要求：需要部署云桥连接器到客户每个VPC中，每个VPC所在SCP需要部署一个公共服务网络，公共服务网络中部署一个代理(双网卡)。保证客户VPC和公服代理网络连通，公服代理和SCC网络连通。

#### 2.5.4 多个VPC应用集成(SCP多集群)

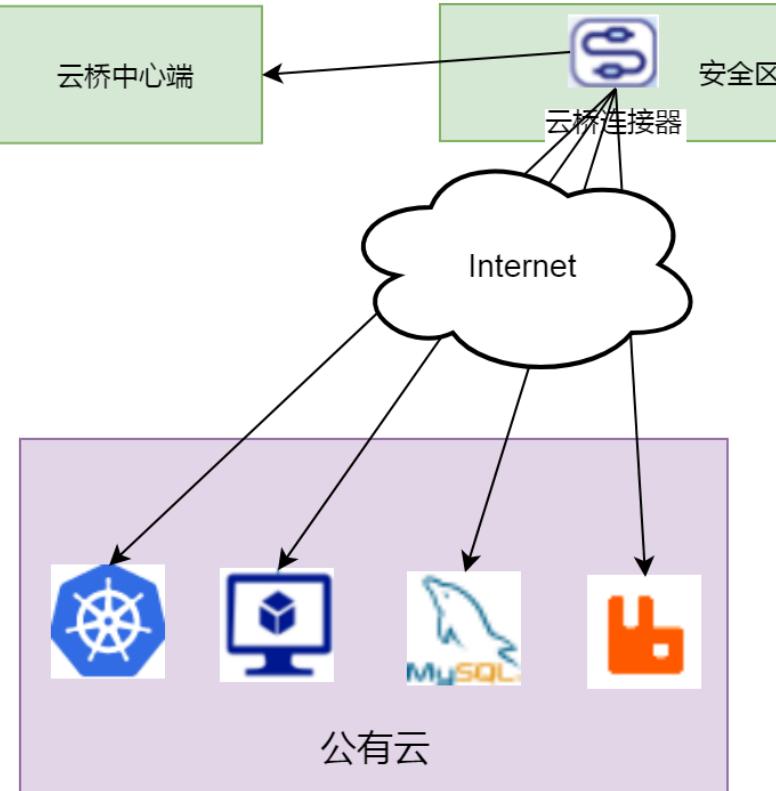
场景描述：客户的业务部署在多个VPC网络中，VPC所在SCP多集群，客户需要实现多个VPC内的多应用集成。



要求：需要部署连接器到每个VPC中，SCP部署一个公共服务网络即可，公共服务网络中部署一个代理(双网卡)。**保证两个集群公服出口相连，公服代理和SCC网络连通。**

## 2.5.5 公网多应用集成

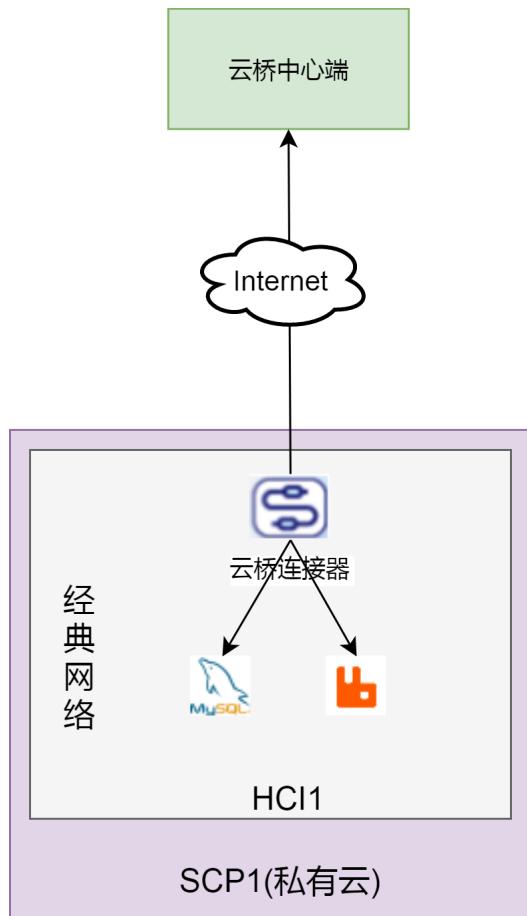
场景描述：客户的业务部署在公网。客户需要实现公网多应用集成。



要求：部署一个云桥连接器到SCC安全区K8S中，为所有客户提供连接公网应用的能力。

## 2.5.6 云下应用集成(私有云场景)

场景描述：客户的业务部署在本地私有云，经典网络下，客户需要实现该网络场景下的多应用集成。



要求：一个集群部署一个云桥连接器服务，集群可以通过Internet连接云桥中心端。

## 2.5.7 OpenAPI触发场景

见云桥管理服务模块设计

## 2.5.8 Binlog触发场景

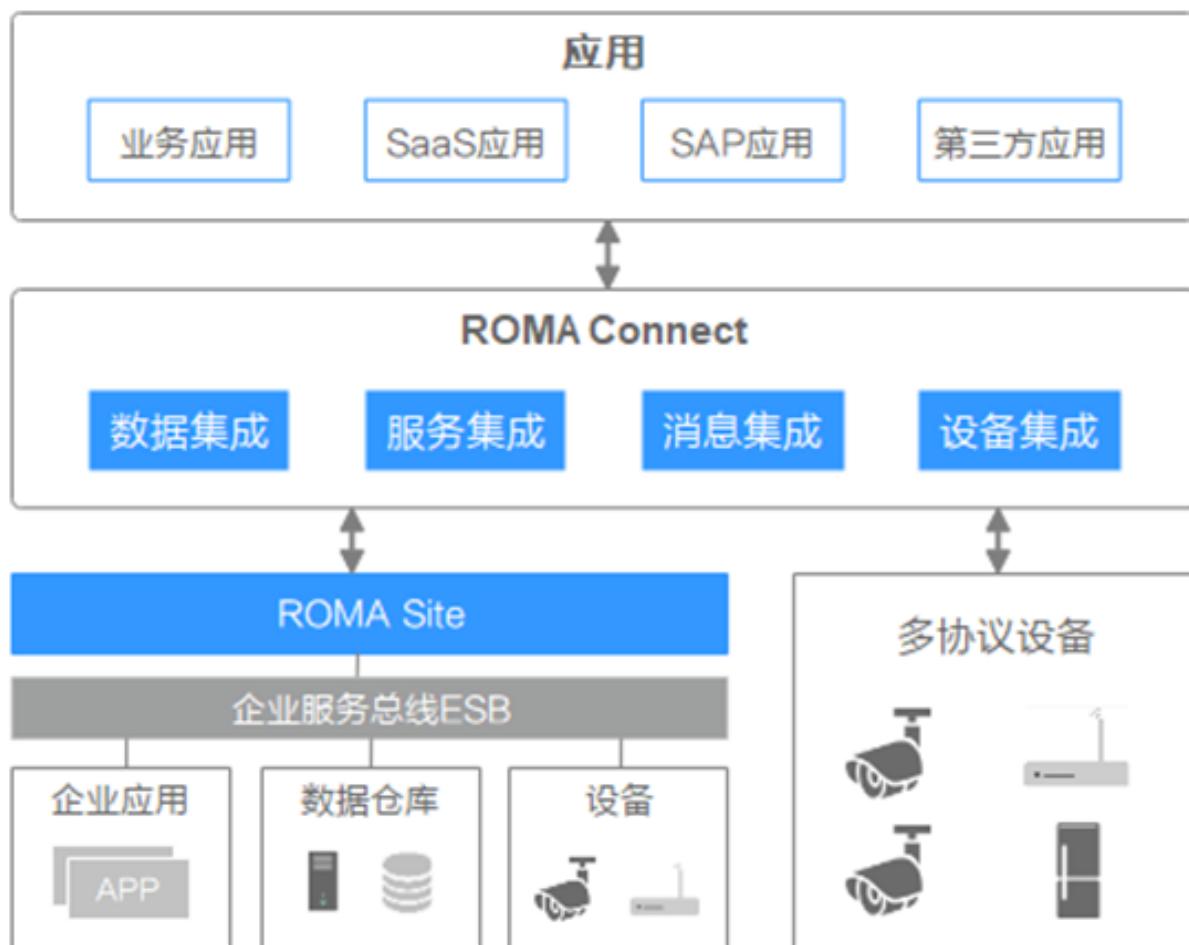
见云桥连接器模块设计

## 2.6 友商分析和对比

### 2.6.1 华为ROMA Connect

华为应用与数据集成平台（ROMA Connect）是一个全栈式的服务平台，聚焦应用和数据连接，提供数据、API、消息和设备集成能力，帮助企业快速联接云上云下，消除数字鸿沟，实现数字化转型。

图：华为ROMA Connect应用架构



ROMA Connect的功能有：

- 数据集成：支持接入多种类型的数据源，并通过数据集成任务实现源端到目标端的数据集成转换
- 服务集成：不同数据源和自定义函数封装成标准的RESTful API，并对外开放
- 消息集成：创建消息Topic，不同系统通过Topic进行对接，发送和接收消息
- 设备集成：在云端定义设备模型和注册设备，设备通过集成SDK接入云端，发送和接收消息

Roma Connect的交互界面：



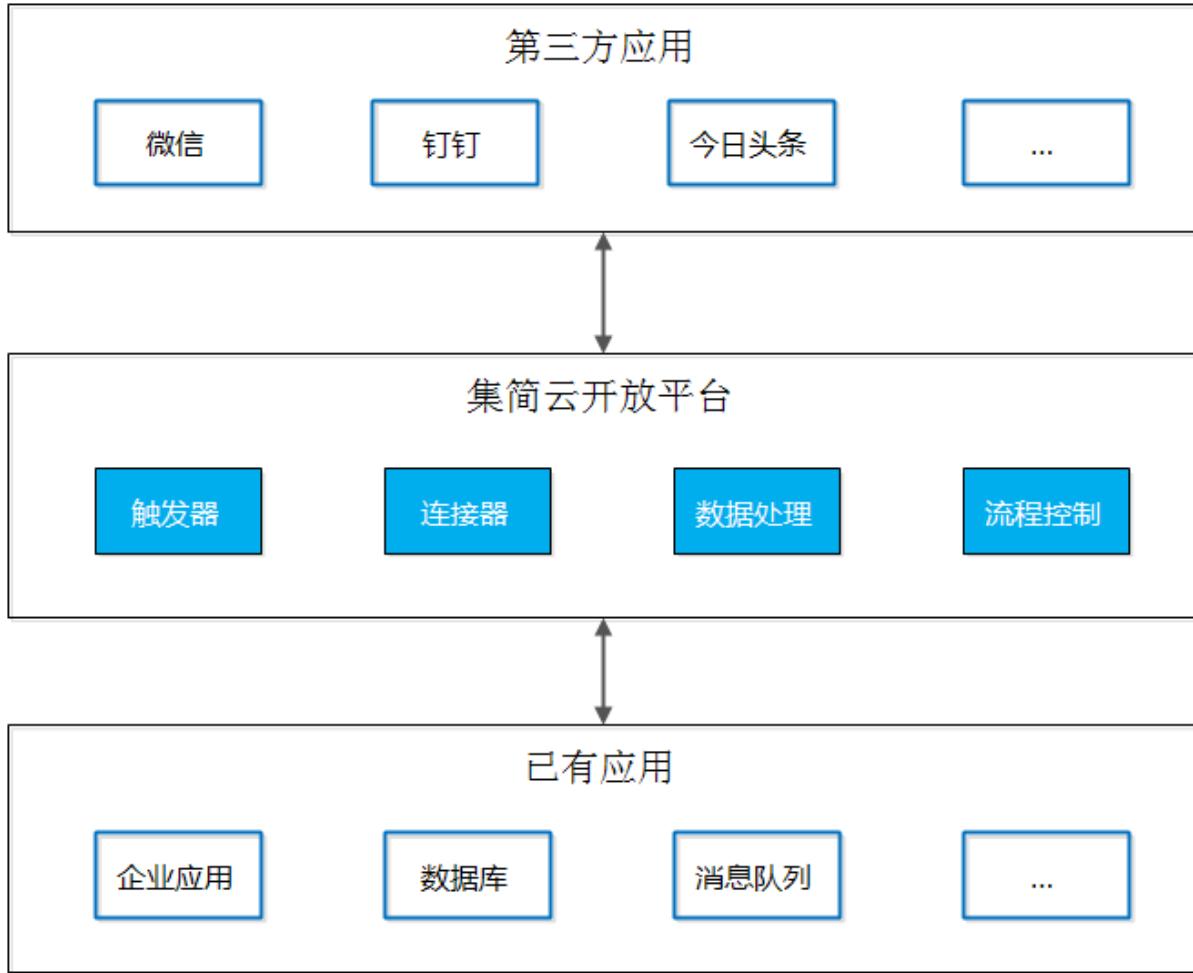
最新版华为ROMA Connect集成平台推出了应用管理能力、多种连接器、多种触发器、多种处理器、可视化流程编排、业务模板管理等功能，可以构建更为灵活的集成功能、业务自动化流程。总结来说，华为Roma是正在迭代的iPaaS平台。

## 2.6.2 集简云开放平台

集简云是一款超级软件连接器，无需开发，无需代码知识就可以轻松打通数百款软件之间的数据连接，构建自动化与智能化的业务流程。通过自动化业务流程，帮助企业节省人工成本。集简云的运行逻辑，可以把它简单的理解为：当“什么”发生时(触发器)，自动执行“什么”动作(连接器)。

目前集简云已经集成了700+款常见应用，可以与企业的各种自建或者第三方业务系统对接，包括客服系统，CRM系统，网站数据分析系统，电子商务系统，物流管理系统，企业数据库，企业API接口等。

图：集简云开放平台应用架构



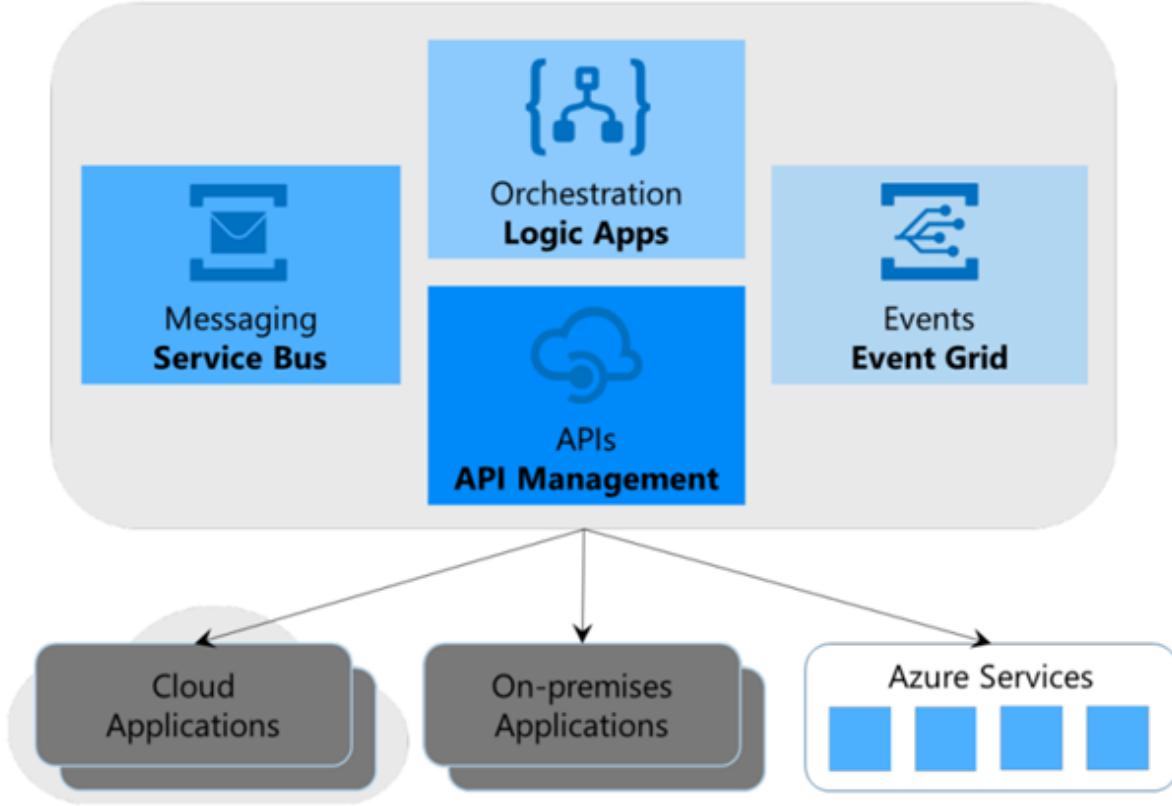
架构分析：

集简云开放平台通过多种触发器、连接器和强大的数据处理、流程控制能力，实现不同应用间的数据同步和自动化流程。基于集简云开放平台能力客户可以快速构建定制化应用。总结来说，集简云是相对完善的iPaaS和aPaaS平台。

### 2.6.3 Azure Integration Services

Azure的集成方案（Azure Integration Services）主要包含以下四个基本组件：

## Azure Integration Services



Azure集成服务的功能有：

- API管理：集成应用API并对其进行管理
- 任务编排：支持对复杂的业务流程进行编排
- 服务总线：支持可靠的消息通信机制(消息队列)
- 事件集成：支持事件发布与订阅(事件总线)

架构分析：

Azure集成服务和老版华为Roma功能较为相似，是一个比较完善的iPaaS和ApimPaaS平台。

### 2.6.4 金蝶苍穹集成服务云

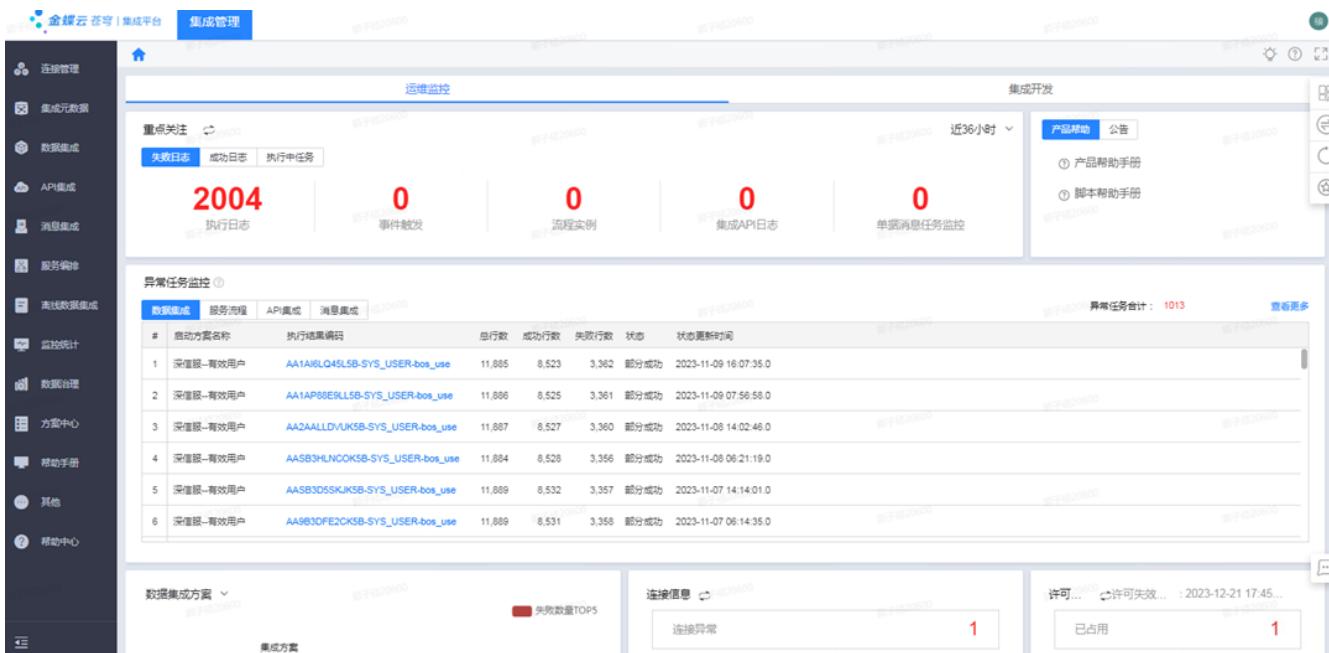
集成服务云是一款采用元数据模型驱动的低代码集成工具，基于系统预置的数据模型、映射、编排、日志、脚本等集成核心引擎，通过可视化配置，快速实现客户灵活多变的动态集成需求，让集成开发简单。平台目前已支持数据集成、消息集成、API集成、服务编排等核心集成功能。

图 苍穹集成云应用架构



苍穹系统有集成的用户界面，支持页面API集成、流程设计、部署、启动等流程，同时支持页面监控与运维，是一个相对较为完善的低代码集成平台。

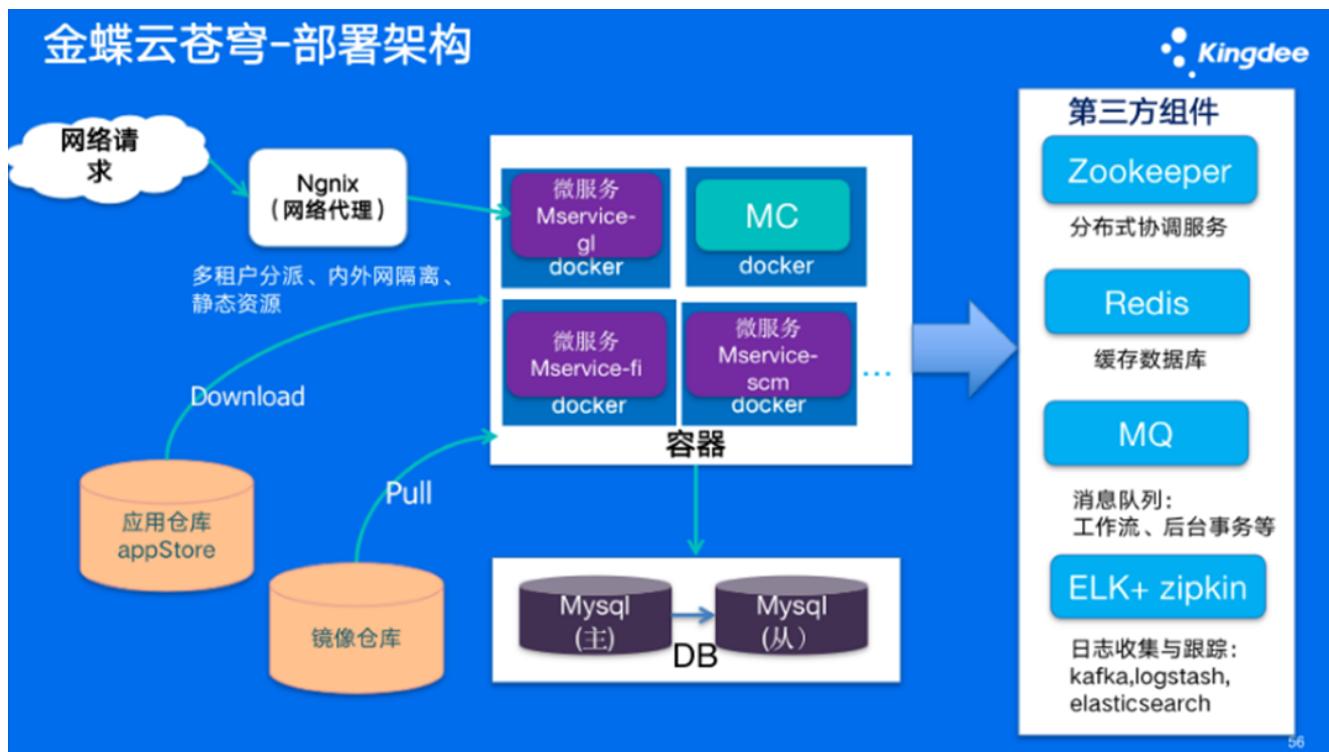
图 苍穹集成平台用户界面



The screenshot shows the Kingdee Integration Platform interface with the following sections:

- 左侧导航栏:** 包含连接管理、集成元数据、数据集成、API集成、消息集成、服务编排、离线数据集成、监控统计、数据治理、方案中心、帮助手册和其他。
- 集成管理 - 集成开发:** 显示了“失败日志”、“成功日志”和“执行中任务”的统计信息（2004条失败日志，0条成功日志，0条执行中任务）。
- 集成管理 - 运维监控:** 显示了“异常任务监控”列表，列出了6条异常任务记录，每条记录包含启动方案名称、执行结果编码、总行数、成功行数、失败行数、状态和状态更新时间。
- 集成管理 - 数据集成方案:** 显示了“失败数量TOP5”和“连接信息”，以及“连接异常”（1个）和“已占用”（1个）的状态。

苍穹部署架构如下，云原生部署，是一个完善的IaaS产品。



架构分析：

金蝶是国内较早研发低代码平台的厂商，苍穹提供了一整套管理平台，可以实现人工任务、脚本任务、UI可视化建模、流程监控等功能。是一个较为完善的IPaaS和APaaS平台。

## 2.6.5 竞品能力对比

分类	功能	华为ROMA	集简云	金蝶苍穹	信服云云桥
流程管理	可视化建模	√	√	√	√
	构建自动化流程	√	√	√	√
	流程模板	√	√	√	√(L2)
	AI生成流程	√	√	✗	√(L2)
	流程监控	√	√	√	√
	变量管理	√	√	√	√
触发器	API触发	√	√	√	√
	定时器触发	√	√	√	√(L2)
	Binlog触发	✗	✗	✗	√
	SQLServer触发	✗	✗	✗	√
	RabbitMQ触发	✗	✗	√	√(L2)
	Kafka触发	√	√	√	√(L2)
执行器	HTTP/S连接器	√	√	√	√(L2)
	数据库连接器	√	√	√	√
	消息队列连接器	√	√	√	√(L2)

	SaaS应用(如企微)	✓	✓	✓	✓(L2)
处理器	分支判断	✓	✓	✓	✓
	循环处理	✓	✓	✓	✓
	并行处理	✓	✓	✓	✓
	延时处理	✓	✓	✓	✓
	数据转换	✓	✓	✓	✓
	脚本处理	✓	✓	✓	✓(L2)
	人工审批	✗	✓	✓	✓(L2)
其他	多云集成	✓	✗	✗	✓(L2)
	连接器快速上架	✓	✓	✗	✓

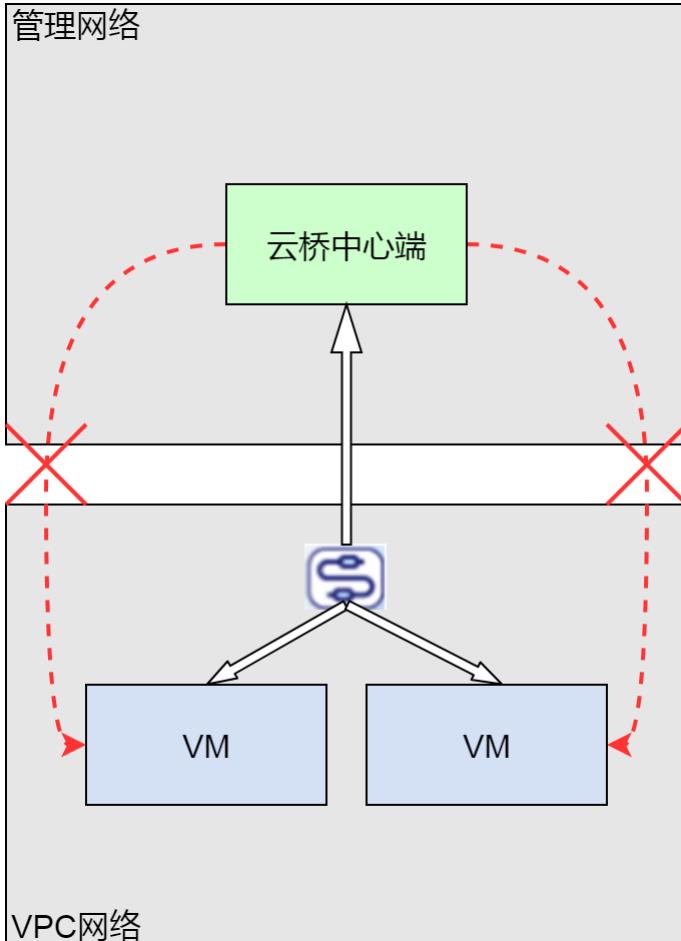
## 2.6.6 创新点

- 1、云桥基于信服云现有的客户场景，提供安全通用的多云集成的能力，可以有效集成公网应用、托管云VPC内应用、私有云应用、自建云应用、各类公有云应用等
- 2、云桥支持更多的触发器与执行器，支持后续对接更多第三方ISV系统，同时支持连接器快速上架
- 3、云桥搭建了客户VPC到SCC的网络通路，后续其他应用可以复用该通路实现业务流程；同时支持SCC任务下发至客户VPC内执行操作。
- 4、云桥后续会实现AI生成流程

## 2.6. 关键技术及解决途径

描述系统所使用的关键技术，比如加速的流缓存算法，这里并不需要描述完整的关键技术，只需要概要说明，但要列举相关资料

### 2.7.1 SCC访问VPC



在VPC网络场景下，云桥中心端处于管理网，客户应用处于VPC网络，云桥中心端无法直接与客户应用进行通信。

解决途径：需要在客户VPC网络中部署一个云桥连接器服务，要求云桥连接器服务可以通过公服网络主动向云桥中心端拉取任务，后主动连接客户的应用执行任务。

### 3. 硬件方案

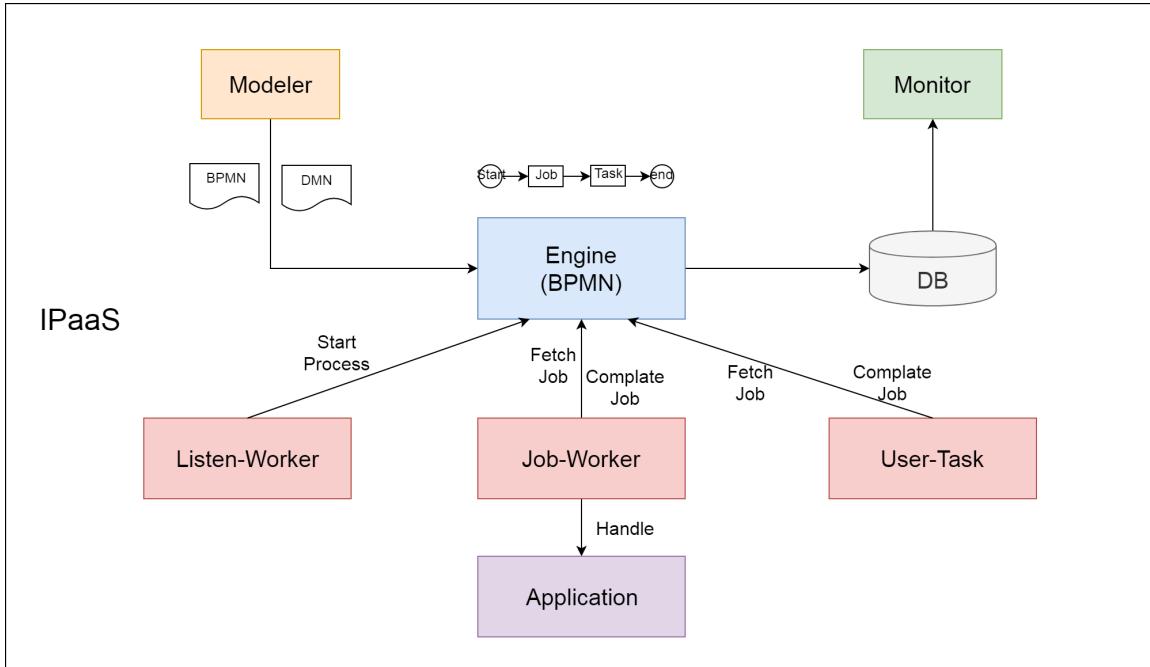
描述系统的硬件构成，描述支持的基本硬件平台以及所采用的其他功能芯片(比如压缩，加密等芯片)。

### 4. 软件方案

#### 4.1. 方案选型

##### 4.1.1. 云桥组件选型

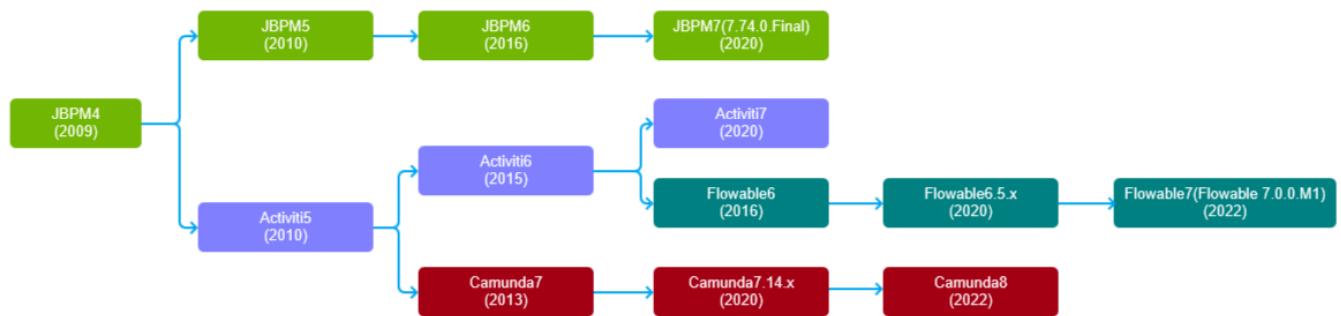
云桥是一个IPaaS集成平台，需要实现通用的流程可视化建模、支持工作流与流程自动化引擎、支持通用的监控与运维平台、支持通用的外部任务执行器功能，整体分层架构如下：



IPaaS中的核心组件是流程引擎，针对流程引擎的选型需求：

- 1、流程引擎Engine需要支持BPMN，支持自动化任务流转
- 2、支持外部Job-Worker的机制，可以将工作执行者服务部署在任意网络位置
- 3、JobWorker需要支持监听(Listen-Worker)和执行(Job-worker)，需要支持人工任务(User-Task)

目前常用的流程引擎有：

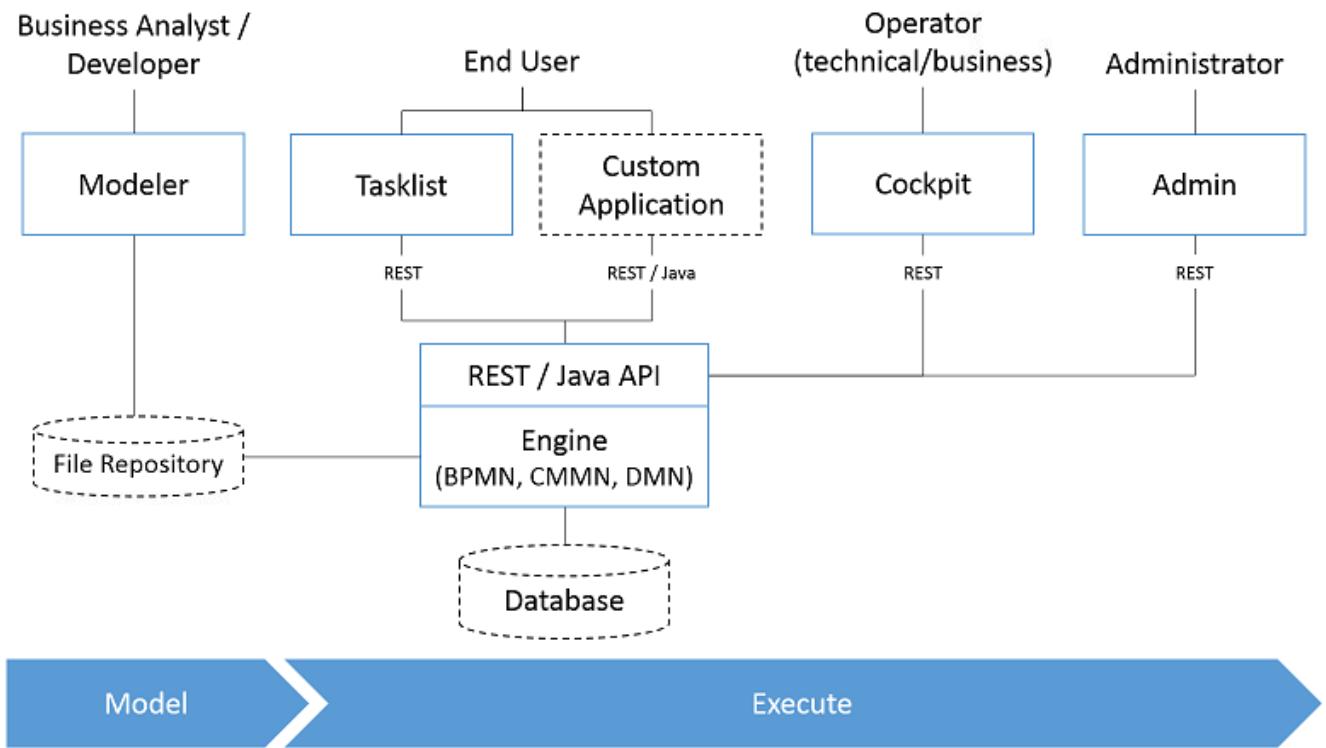


这些流程引擎框架都同宗同源，都是由JBPM4发展而来的。这里根据社区活跃度与功能完整性，每个框架选出1-2个版本来分析，分别为Camunda7、Camunda8、Activiti7、Flowable7、JBPM7.74.0Final。

## 方案一：camunda7

Camunda7基于activiti5发展而来，最新版本Camunda 7.21，基于Java的框架，支持用于工作流和流程自动化的BPMN、用于案例管理的CMMN和用于业务决策管理的DMN。保持每年发布2个小版本的节奏。camunda7在功能方面比flowable、activiti流程引擎强大，性能和稳定性更突出。camunda7的大部分组件开源，基于Apache2.0协议，可免费使用，技术生态较好，程序员上手容易。

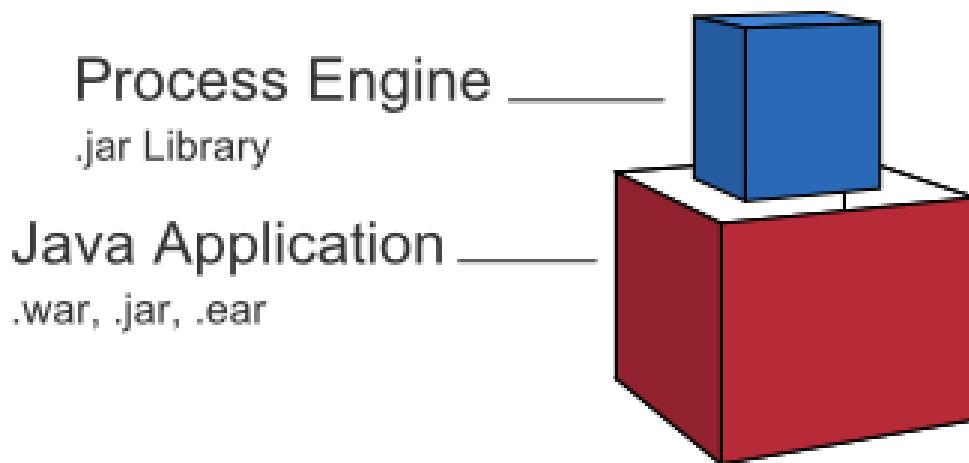
camunda7的架构如下：



Camunda7分为流程设计和流程运行两个阶段，见上图最下方的蓝色大箭头，Model和Execute，按照这两个阶段，Camunda划分为两大部分功能，对应设计阶段的功能有Modeler（流程设计器），对应运行阶段的功能有Engine(流程引擎)、TaskList(用户任务)、Custom Application(第三方应用程序)、Operator(监控与操作)。在Camunda7商业产品中还包括了流程监控预警工具（Optimize）、流程协同设计工具（Cawemo）。Database为流程引擎的数据库系统，目前支持MySQL、Oracle、PostgreSQL、SQL Server、H2等关系型数据库。

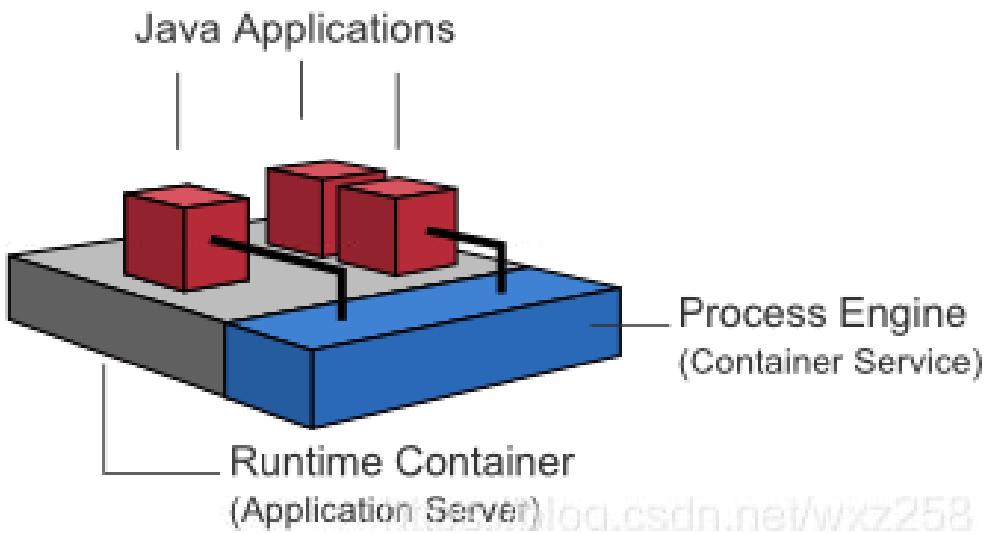
Camunda7是一个灵活的框架，支持嵌入式、分布式、集群等多种部署模式。

1. 嵌入式部署 流程引擎以Jar包方式添加到应用程序中，通过这种方式，可以在应用程序生命周期中轻松启动和停止流程引擎。



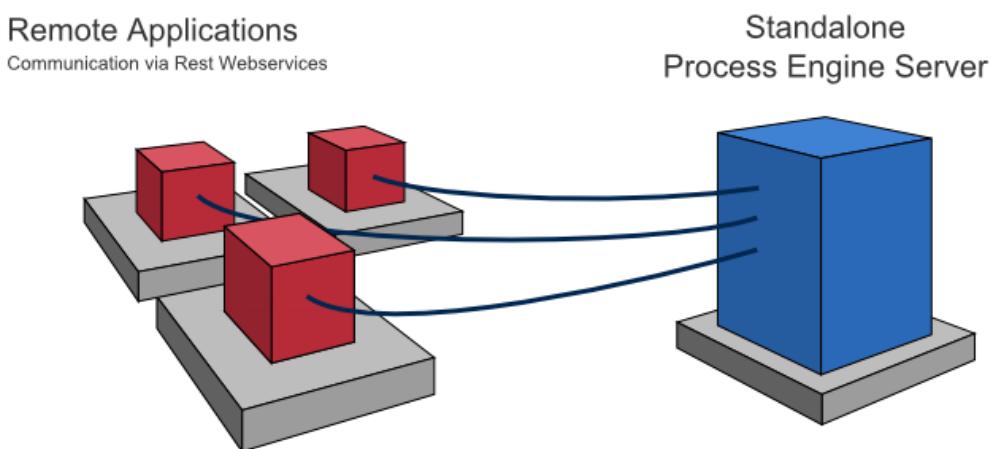
2. 基于web容器启动，多应用共享

流程引擎在运行时容器（Servlet容器、应用程序服务器等）中启动，流程引擎作为容器服务提供，可以由容器内部署的所有应用程序共享。这种方式在实际应用场景中不多见。



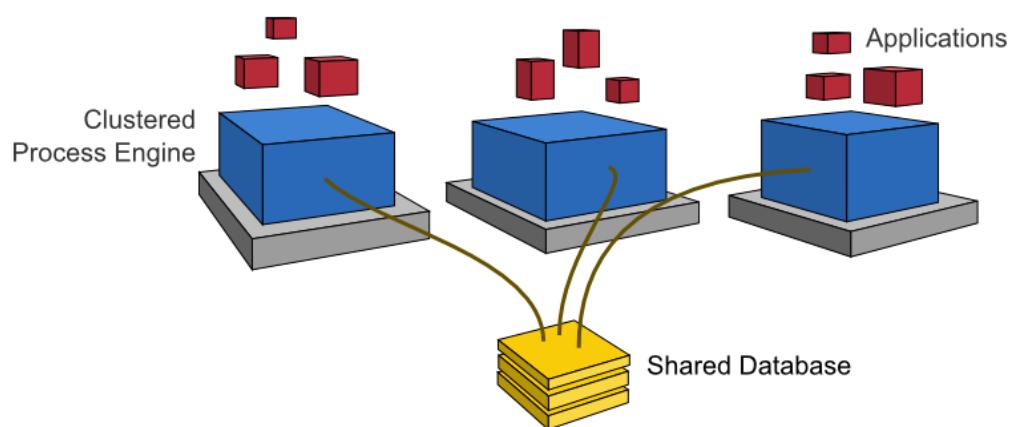
### 3. 独立部署，多应用共享

流程引擎独立部署，通过网络提供服务，网络上运行的不同应用程序可以通过远程通信通道与流程引擎交互，远程访问流程引擎的最简单方法是使用内置的REST服务接口。在企业级流程中心部署架构中，这是一种最常见的部署模式，在现在的微服务部署架构中，也可以采取这种方式。



### 4. 集群部署

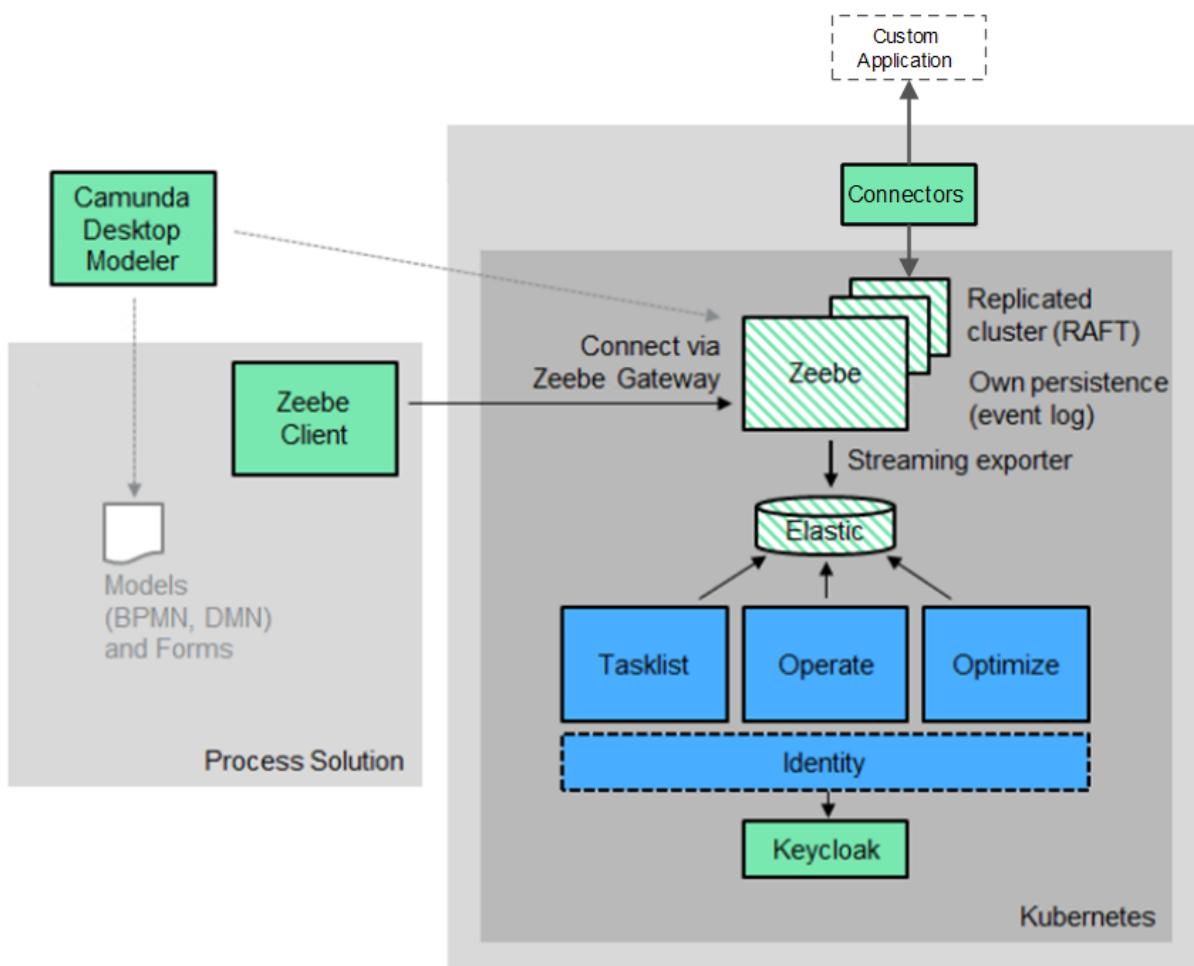
为了提供扩展或故障转移功能，流程引擎可以分布到集群中的不同节点，每个流程引擎实例都必须连接到共享数据库。Camunda7不提供现成的负载均衡功能，可以采用nginx或k8s service等负载均衡软件实现。



## 方案二：camunda8

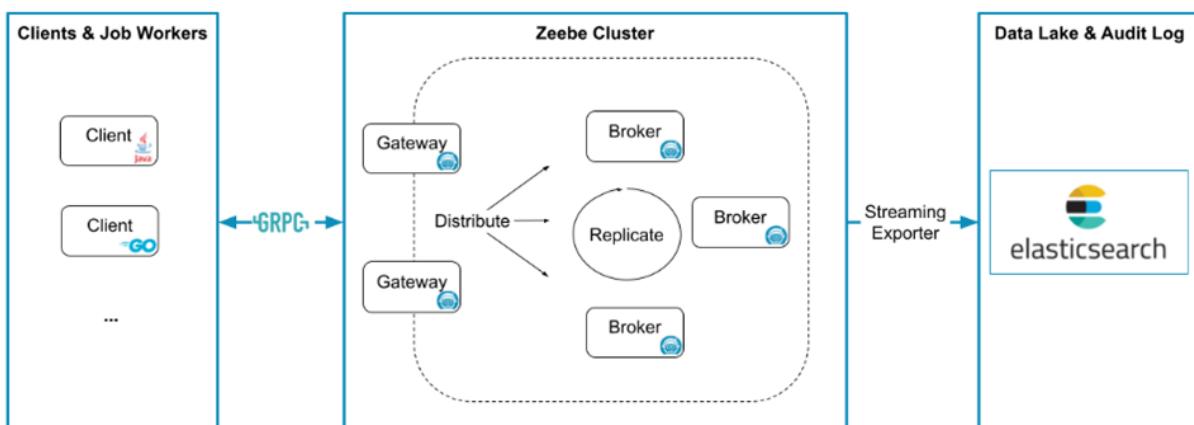
2022年4月，camunda官方发布了Camunda8新版本，Camunda7和Camunda8在技术架构方面有本质区别。Camunda8定位于云架构SaaS模式，基于Zeebe流程引擎内核，采用gRPC API接口技术，不再使用关系型数据库。在开源和商业授权方面，Camunda8有诸多限制，Camunda8仅有Zeebe、modeler、elastic组件是开源的，可以免费使用，其它的组件Camunda Operate、Camunda Tasklist、Camunda Optimize等组件是需要商业授权才能使用。另外，我们从GitHub上看到，只有 Zeebe、modeler 的源码，没有 Camunda Operate、Camunda Tasklist 或 Camunda Optimize 的源码。如果对流程自动化和高并发有显著需求的客户，可以考虑选择camunda 8，有商业授权风险，对技术团队能力要求较高。

camunda8的架构如下：(绿色部分开源可用，蓝色为商业授权可用)



Camunda8同样分为流程设计和流程运行两个阶段，按照这两个阶段，Camunda8划分为两大部分功能，对应设计阶段的功能有Modeler（流程设计器），对应运行阶段的功能有Zeebe(流程引擎)、TaskList(用户任务)、Connectors(连接第三方应用程序)、Operator(监控与操作)、Identity(身份与认证)。Zeebe不再使用关系型数据库，而是采用更高性能的存储方案、支持频繁写入的嵌入式键值存储库-- RocksDB；同时为了优化流程引擎的运行功能，Zeebe不再对外提供查询接口，选择把所有的运行数据导出到Elasticsearch，供其他组件获取数据。官方提供了多种ZeebeClient和通用的Connectors组件，用于执行监听任务和处理任务。

Zeebe的架构分层：

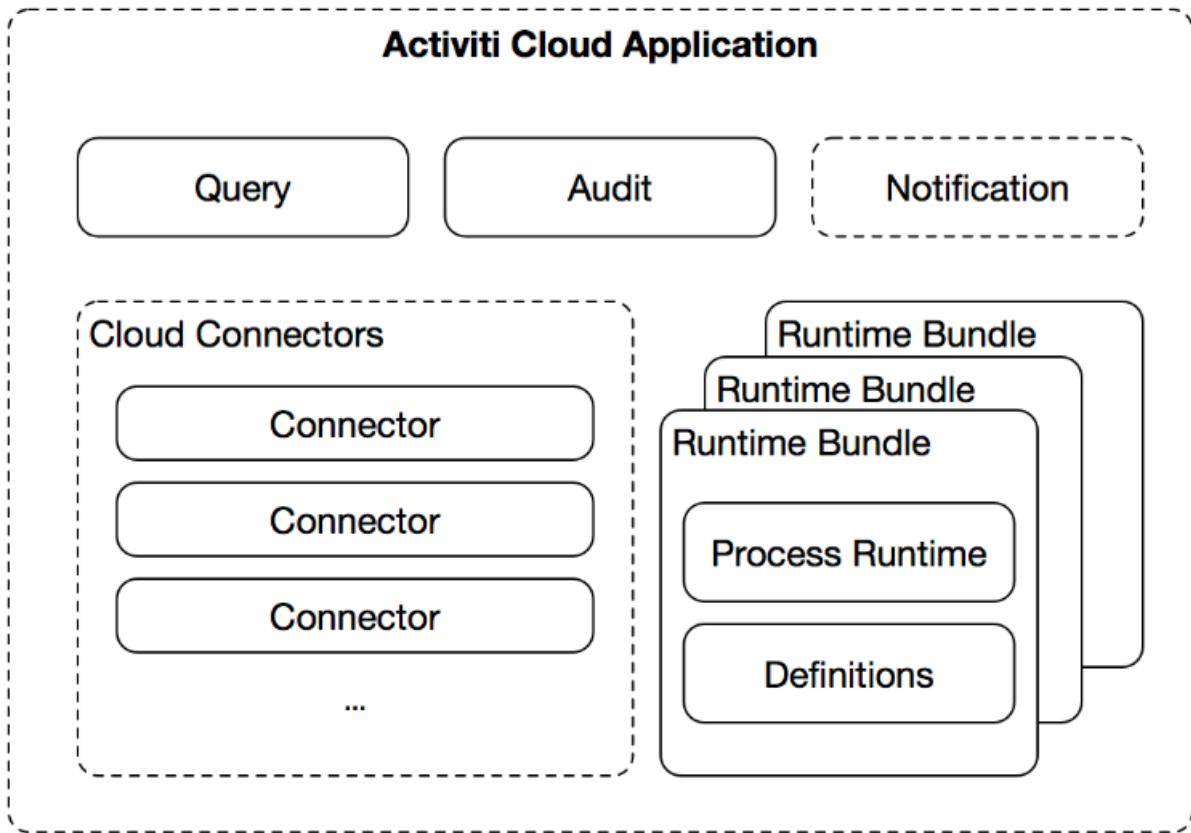


Zeebe支持集群化部署，对外提供gRPC的应用层协议，拥有更好的通信性能。Zeebe-Gateway被视为Zeebe 集群的通信组件，作为Zeebe-Broker的负载均衡器和路由器，负责处理、复制以及基于分区的所有工作

### 方案三：Activiti7

最初，Activiti没有提供外部任务的概念，但随着Activiti 7的发布，引入了新的Activiti Cloud版本，这个版本支持了类似于Camunda的外部任务模式。Activiti Cloud是为微服务架构设计的，并且提供了与外部服务交互的能力，包括外部任务的处理。同时Activiti 7全面拥抱了云原生，可在分布式环境中工作。

Activiti 7的架构如下：



Activiti Cloud 包括 5 个基本构建块④：

- Activiti Cloud Runtime Bundle：为业务流程、任务、决策表提供执行运行时
- Activiti Cloud Connectors：提供与外部系统的双向连接
- Activiti Cloud Query：提供对一个或多个运行时生成的信息的读取访问
- Activiti Cloud Audit：提供审计跟踪功能，从一个或多个运行时收集信息
- Activiti Cloud Notifications Service (GraphQL)：与查询服务结合使用，通知服务通过订阅启用核心构建块并推送有关应用程序状态的通知

以上所有构建块采用REST通信方式。

注④参考：<https://activiti.gitbook.io/activiti-7-developers-guide/components>

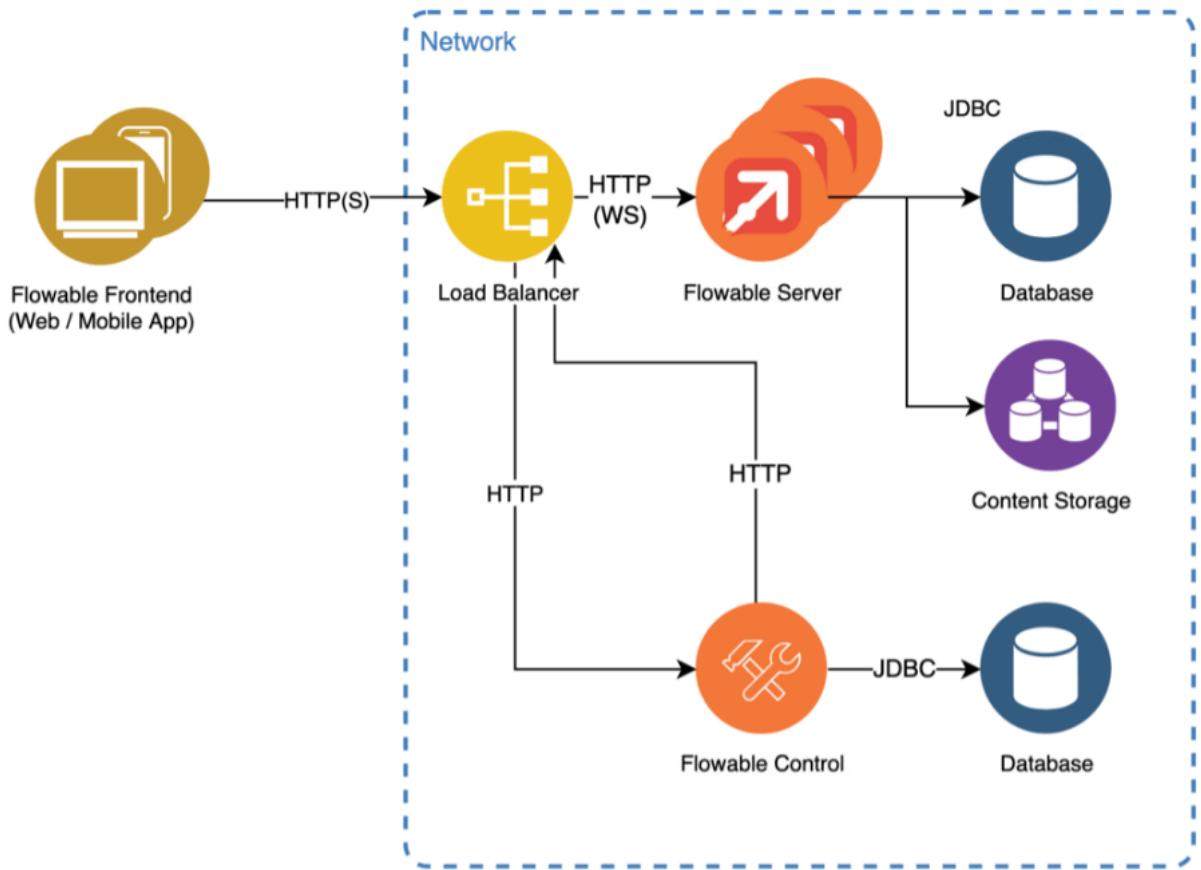
### 方案四：Flowable7

Flowable7是一个使用Java编写的轻量级业务流程引擎。Flowable流程引擎可用于部署BPMN 2.0流程定义（用于定义流程的行业XML标准），创建这些流程定义的流程实例，进行查询，访问运行中或历史的流程实例与相关数据。

Flowable7架构主要分为四部分：工作流引擎、应用程序接口（API）、模型器和任务表单设计器。

- **工作流引擎**：Flowable的核心组件，包括运行时引擎和执行引擎。它管理整个流程的生命周期，监控、控制任务的执行以及记录流程实例的状态等信息。
- **应用程序接口（API）**：根据RESTful风格，提供给外部系统访问Flowable引擎的接口，可以通过编写调用API的客户端程序来使用Flowable引擎服务。
- **模型器**：用于创建和修改流程定义文件，支持基于Web的图形化编辑器。

- 任务表单设计器：用于创建和修改任务表单，支持基于Web的表单设计器。



Flowable7 具有**外部工作任务**，允许您通过使用自己的自定义代码轮询 Flowable 来执行自定义逻辑。外部工作客户端是一组适用于不同编程语言的库，可以更轻松地实现自定义逻辑⑤。目前，Flowable 提供以下外部工作客户端：Java、Python。

注⑤参考：<https://documentation.flowable.com/latest/develop/be/external-worker-client/index.html>

## 方案五：JBPM7.74.1final

JBPM 是一个灵活的业务流程管理 (BPM) 套件。它是轻量级的、完全开源的（根据 Apache License 2.0 分发）并用 Java 编写。它允许您在业务流程和案例的整个生命周期中建模、执行和监控。JBPM 的核心是一个用纯 Java 编写的轻量级、可扩展的工作流引擎，允许您使用最新的 BPMN 2.0 规范执行业务流程。它可以在任何 Java 环境中运行、嵌入到您的应用程序中或作为服务运行。

某些任务类型（例如，脚本任务和标准决策引擎规则任务）在引擎中实现。对于其他任务类型，包括所有自定义任务，当必须执行该任务时，JBPM 引擎将使用工作项处理程序 API 执行调用。引擎外部的代码可以实现这个 API，为实现各种任务提供灵活的机制⑥。

注⑥参考：[https://docs.jbpm.org/7.74.1.Final/jbpm-docs/html\\_single/](https://docs.jbpm.org/7.74.1.Final/jbpm-docs/html_single/)

所以JBPM7.74.0final不支持独立的外部任务客户端，**不符合我们这边的需求**。

## 方案选择

备选方案名称	本方案的优点	本方案的风险和缺点	最终选择
--------	--------	-----------	------

方案一：camunda7	<p>1、大部分组件开源，社区生态较好 2、支持统一表达式语言: Jakarta Expression Language 4.0 standard，可以构建更加灵活的自动化流程</p>	<p>1、camunda7为传统软件架构，不过最新版的7.21已经支持了Docker部署，需要进行云原生改造，需要自行实现负载均衡、横向扩容。 2、目前<b>团队并未对Camunda7进行充分预研，有较大的的设计风险</b> 3、官方未提供通用的外部任务客户端，需要自研 4、没有开箱即用连接器，只有一个java客户端</p>	
方案二：camunda8	<p>1、采用云原生架构，对性能做了多方面优化，性能更好 2、支持统一表达式语言：FEEL 3、目前在SCC已实现，团队有较好的技术储备，IPaaS前期设计也是基于这个组件 4、官方提供了开箱即用连接器（java），同时提供了go客户端 5、社区热度高，并两个月一个版本</p>	<p>1、流程引擎Zeebe基于社区许可证 ZEEBE-COMMUNITY-LICENSE-1.1，可以在生产环境中自用，但是<b>不允许使用Zeebe在云中提供商业工作流服务。</b></p>	√
方案三：Activiti 7	1、社区热度较高	<p>1、没有官方连接器客户端 2、官方文档太过简单，学习成本较大</p>	
方案四：Flowable7		<p>1、没有开箱即用连接器，只有一个java客户端 2、官方文档很乱 3、社区热度较低</p>	
方案五：jbpm 7.74.1final		<p>1、没有官方连接器客户端 2、不支持外置连接器客户端 3、社区热度较低</p>	

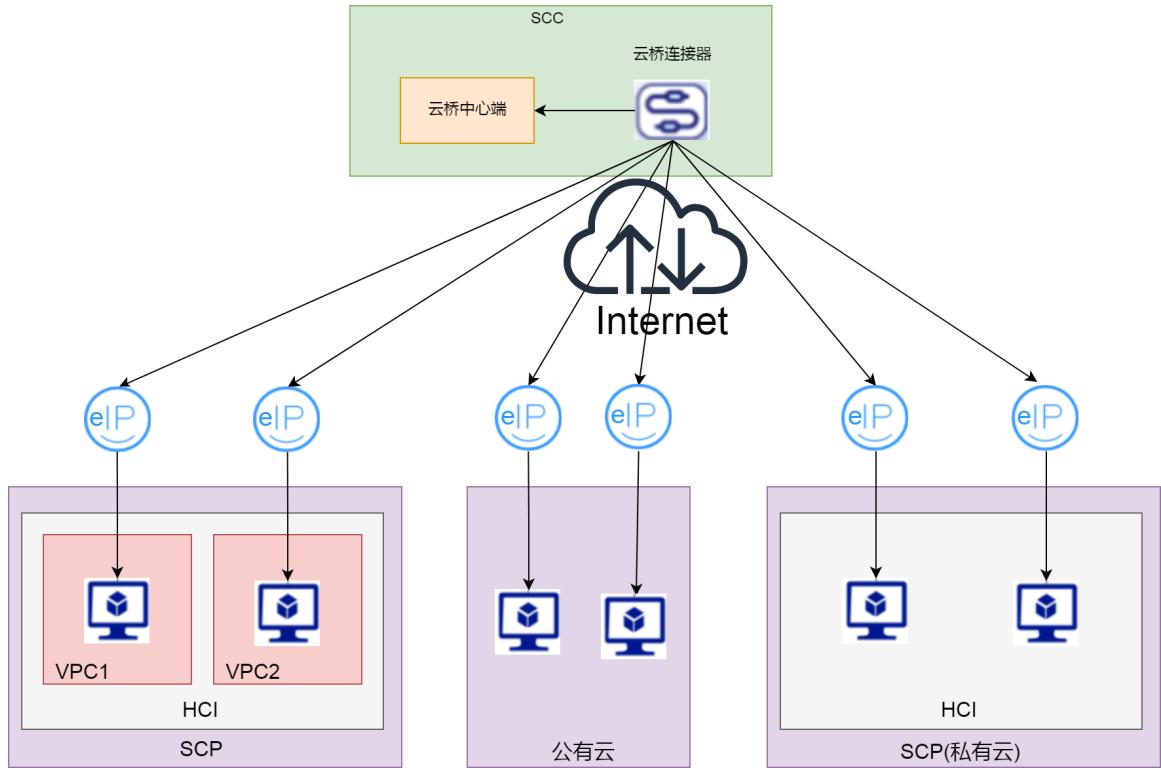
#### 4.1.2. 云桥网络方案(网络层互通)

选型需求：

1、网络方案需要考虑多云集成，有效集成公网应用、托管云VPC内应用、私有云应用、自建云应用、各类公有云应用等

##### 方案一：云桥中心端部署在SCC，云桥连接器仅在SCC部署一个

云桥连接器从云桥中心端获取任务，通过公网弹性IP连接所有应用

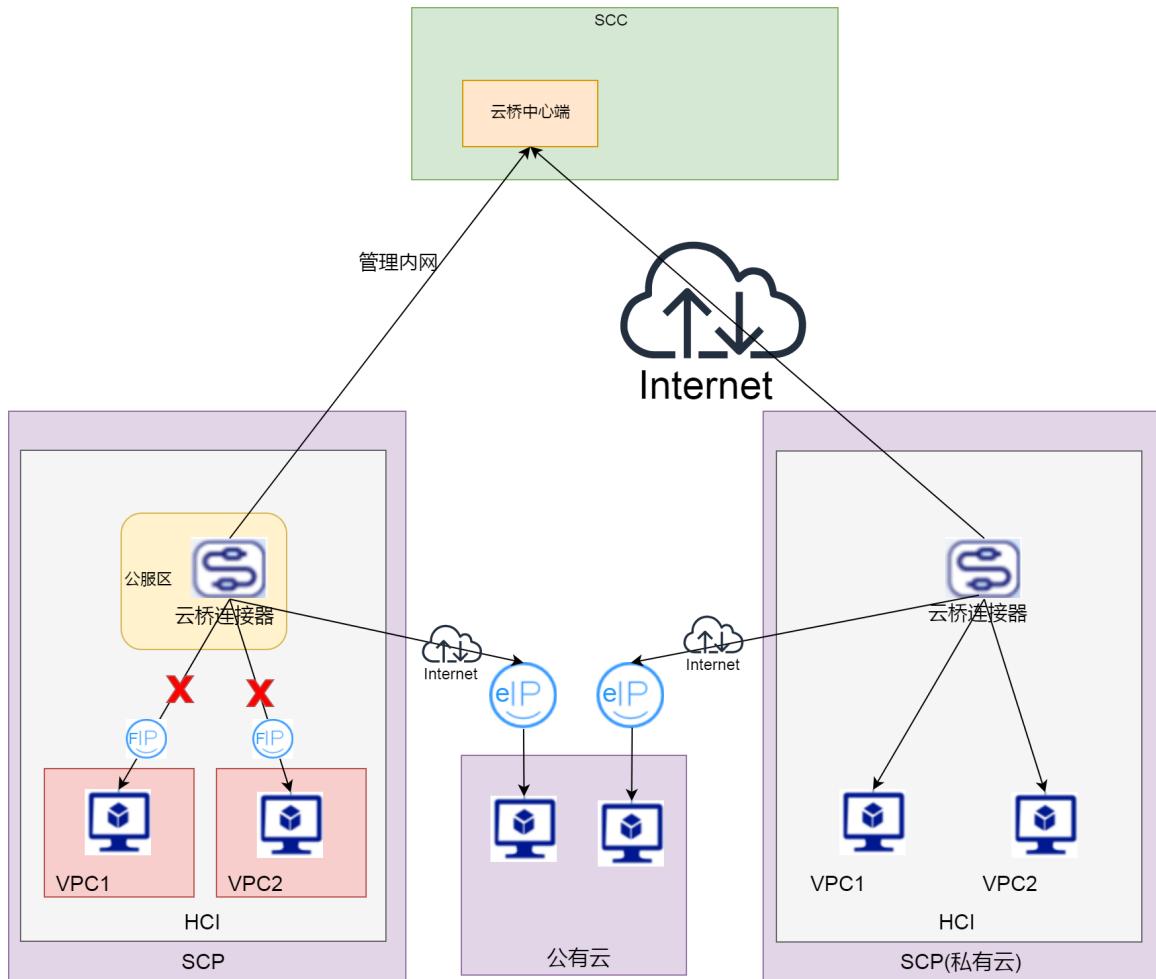


## 方案二：云桥中心端部署在SCC，云桥连接器每个数据中心部署一个

云桥连接器每个数据中心部署一个，托管云负责维护，每个数据中心下的租户共享一个云桥连接器，云桥连接器需要满足客户的VPC集成需求和公网集成需求

托管云VPC场景：云桥连接器部署在每一个数据中心的公服网络中，通过管理内网连接云桥中心端获取任务，通过公服FIP或公网访问所有应用

私有云场景：云桥连接器部署在经典网络中，通过Internet连接云桥中心端获取任务，通过集群管理内网或公网访问所有应用

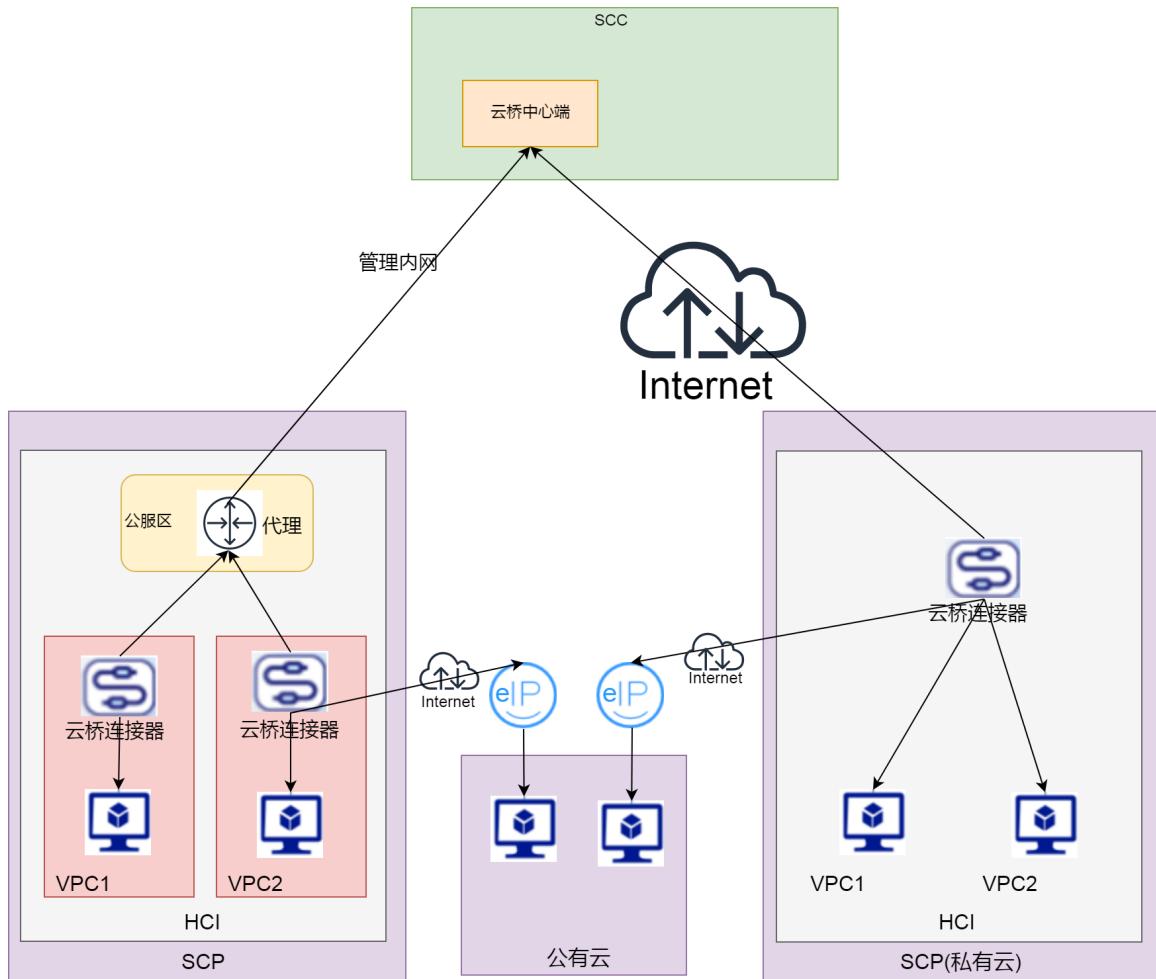


### 方案三：云桥中心端部署在SCC，云桥连接器每一个VPC部署一个

云桥连接器部署在**客户网络中**，托管云负责维护，单个云桥连接器**只为租户本身集成需求负责**，云桥连接器需要满足客户的VPC集成需求和公网集成需求

**托管云VPC场景**：云桥连接器部署在每一个VPC中，通过公服代理走管理内网连接云桥中心端获取任务，通过VPC内部网络或公网访问所有应用

**私有云场景**：云桥连接器部署在经典网络中，通过Internet连接云桥中心端获取任务，通过集群管理内网或公网访问所有应用



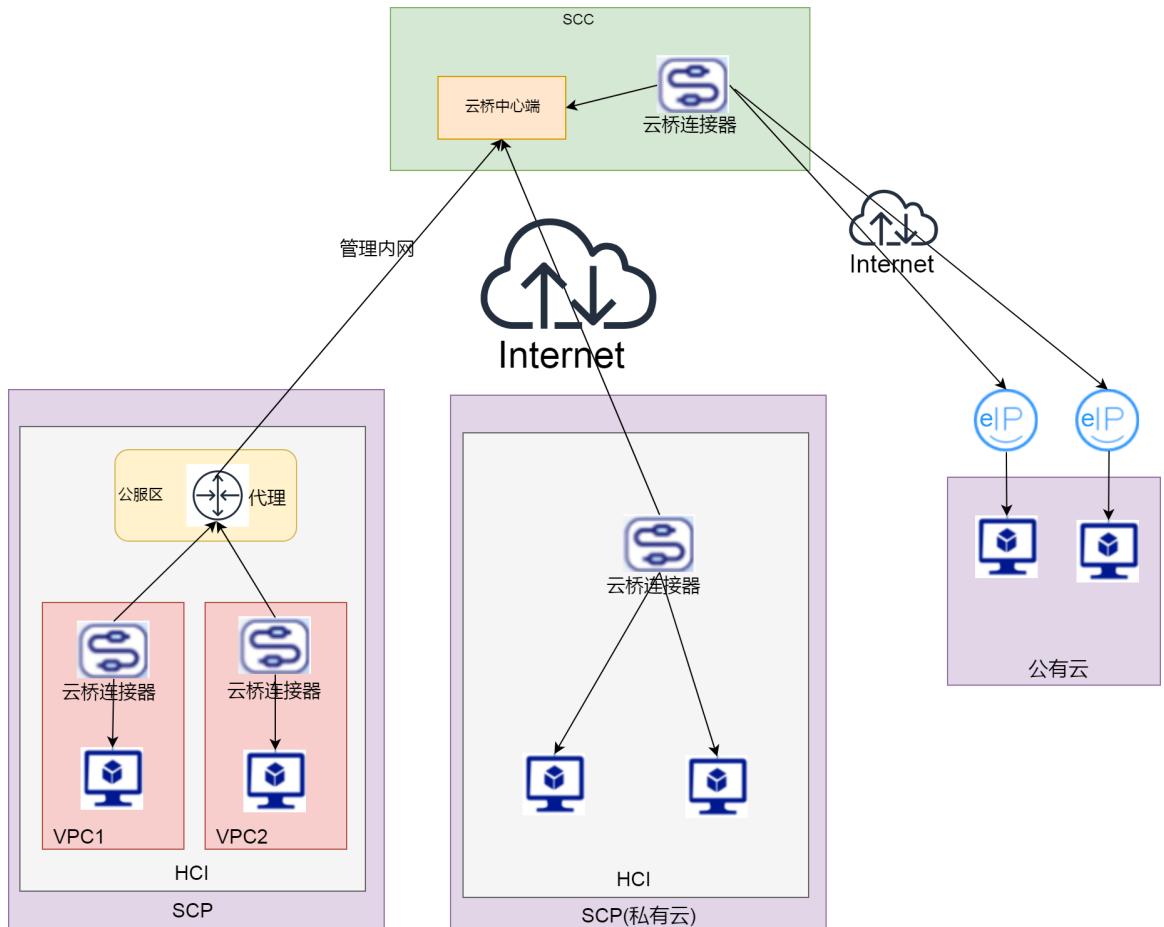
#### 方案四：云桥中心端部署在SCC，云桥连接器在VPC和SCC各部署一个

云桥连接器部署在客户网络中以及在SCC侧，托管云负责维护，VPC内云桥连接器只负责集成VPC内应用，SCC侧云桥连接器负责集成公网应用

托管云VPC场景：云桥连接器部署在每一个VPC中，通过公服代理走管理内网连接云桥中心端获取任务，通过VPC内部网络访问应用

私有云场景：云桥连接器部署在经典网络中，通过Internet连接云桥中心端获取任务，通过集群管理内网访问应用

公网场景：无论是托管云客户还是私有云客户都可以使用SCC侧云桥连接器通过Internet访问公网应用

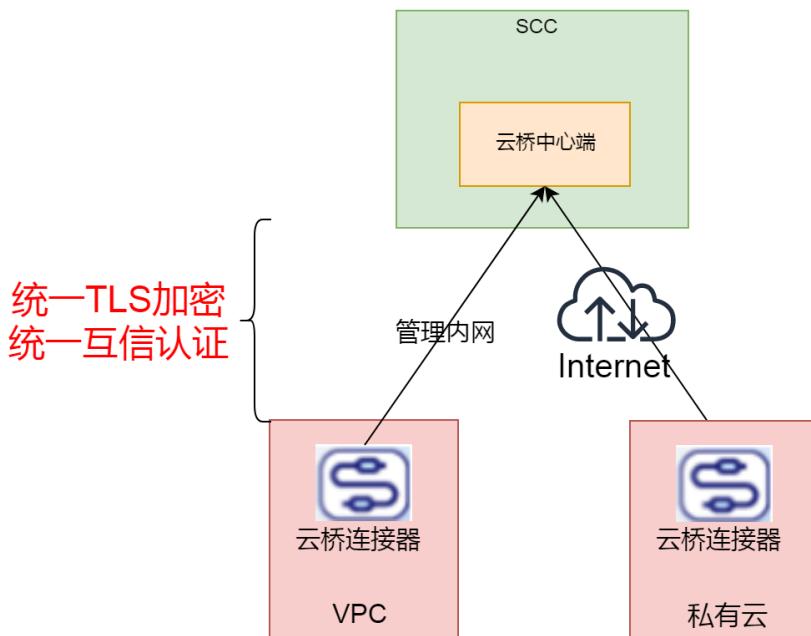


## 方案选择

序号	备选方案名称	本方案的优点	本方案的风险和缺点	最终选择
方案1	云桥中心端部署在SCC，云桥连接器仅在SCC部署一个	1、整体架构简单，维护成本较低，无网络改造成本 2、云桥连接器数量只有一个，整体对性能挑战较小 3、云桥连接器部署在SCC，不用过分要求云桥连接器的资源占用 4、不占用客户的资源	1、所有应用必须通过公网可达， <b>要求客户应用暴露端口到公网</b> ，无法吸引对网络安全要求较高的客户 2、客户需要购买大量公网IP，需要客户调整现有的网络架构，客户直接成本和维护成本都会加大	
方案2	云桥中心端部署在SCC，云桥连接器每个数据中心部署一个	1、架构相对简单，复用现有所有SCP到SCC的网络通路，几乎没有网络改造成本 2、不占用客户的资源	1、现有的数据中心网络安全要求： <b>不允许主动连接VPC网络</b>	

方案3	云桥中心端部署在SCC，云桥连接器每一个VPC部署一个	<p>1、云桥连接器部署在租户网络内部，只需应用暴露端口给VPC内部即可，可以安全进行应用连接</p> <p>2、客户几乎没有网络改造成本</p> <p>3、后续SCC连接VPC内部可以沿用这条安全通路</p>	<p>1、由于会占用客户资源，所以需要进行云桥连接器改造，由Java→Go</p> <p>2、云桥连接器数量较大，会有大量请求从云桥中心端获取任务，需要对云桥中心端进行并发设计</p> <p>3、存在数据中心网络改造成本</p> <p>4、需要考虑以何种形式部署云桥连接器到客户VPC内部</p> <p>5、云桥连接器需要从VPC连接公网应用，会占用客户的网络带宽</p> <p>6、云桥连接器会有冗余线程，比如每个云桥连接器都要启动连接企微的执行器，会有大量的企微执行器线程从云桥中心端获取任务</p>	
方案4	云桥中心端部署在SCC，云桥连接器每一个VPC和SCC均部署一个	<p>1、公共云桥连接器部署在SCC，所有租户公用这些连接器</p> <p>2、VPC内云桥连接器只需要运行需要的连接器线程即可，会减少云桥中心端的并发请求数</p> <p>3、公共连接器和私有连接器分开管理，可以满足更加灵活的业务需求</p>	<p>1、由于会占用客户资源，所以需要进行云桥连接器改造，由Java→Go</p> <p>2、云桥连接器数量较大，会有大量请求从云桥中心端获取任务，需要对云桥中心端进行并发设计</p> <p>3、存在数据中心网络改造成本</p> <p>4、需要考虑以何种形式部署云桥连接器到客户VPC内部</p>	√

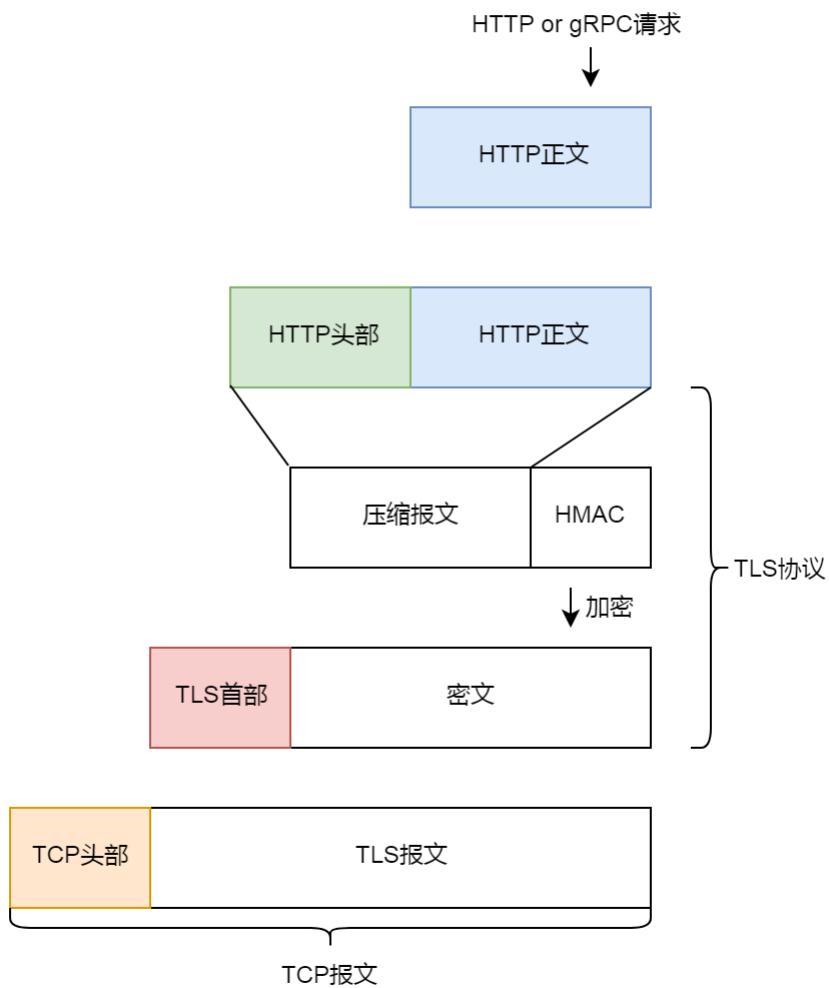
#### 4.1.3. 云桥互信方案(TLS层和应用层安全)



选型需求：

- 1、确保云桥连接器通过管理网络和Internet与云桥中心端通信安全，防止中间人攻击，采用TLS1.3建立安全的通信通道。
- 2、云桥连接器和云桥引擎应用层协议：gRPC(s)，需要设计统一的互信方案，防止身份伪造
- 3、云桥连接器和云桥管理服务应用层协议：HTTP(s)，需要设计统一的互信方案，防止身份伪造
- 4、由于云桥连接器数量较大，所以必须保证互信方案简单有效

#### 统一的TLS加密传输：



#### 统一互信方案：

首先APISIX需要代理云桥连接器到云桥引擎的gRPC请求，添加如下route配置：

```
curl http://10.132.48.88:9180/apisix/admin/routes/cloudbriider -H 'X-API-KEY: 66da8a2f-4aa1-4988-ab9a-fbf65e311fd6' -X PUT -d '
{
  "uri": "/gateway_protocol.Gateway/*",
  "upstream": {"nodes": {"camunda-zeebe-gateway:26500": 1}, "scheme": "grpc", "type": "roundrobin"},
  "plugins": {}
}
```

添加成功后，APISIX支持将/gateway\_protocol.Gateway/\*的gRPC请求转发到camunda-zeebe-gateway:26500服务

plugins字段用于设置该route需要经过的插件，互信认证需要在其中一个插件中完成。

同时需要开启APISIX监听的9080(用于管理口进来的请求)和9081(业务口进来的请求)的http2协议。

```
apisix:  
  enable_ipv6: false  
  node_listen:  
    - port: 9080  
      enable_http2: true  
    - port: 9081  
      enable_http2: true
```

针对应用层安全，有以下多种互信认证的方案

## 方案一：AKSK+openapi\_auth验签

SkyOps已经储存了一份SCC的AKSK，云桥连接器服务在请求SCC时将签名信息放入对应的头部后请求SCC，SCC的Openapi-auth负责验证签名

请求示例：

```
grpcurl -plaintext -proto gateway.proto -d '{"processDefinitionKey": "2251799817793033"}' -k \  
--header 'X-Ca-Timestamp: 1632479334' \  
--header 'X-CA-Key: a54818830bed4ba980d9931280b0d96a' \  
--header 'X-Ca-Nonce: 0bb7f9c81acf4eaa903400f0596dcf6a' \  
--header 'X-Ca-Signature-Headers: X-Ca-Nonce;X-Ca-Timestamp' \  
--header 'X-Ca-Signature: 9619fbb69bleea8e591bc3753602547c95ff6783483dac8d59c949f3512a0563' \  
--header 'X-Ca-Algorithm: hmac-sha256' \  
10.132.48.88:9080 gateway_protocol.Gateway/CreateProcessInstance
```

## 方案二：key-auth

Key-Auth 方案通常用于身份验证，其中客户端需要在请求中携带一个特定的密钥（key）作为身份验证凭据。请求在APISIX中针对该密钥进行验证，密钥信息储存在APISIX-Consumer中。在 gRPC 中，可以通过添加自定义的头部来实现 Key-Auth 方案。

客户端示例：

...

```
package main
```

```
import (  
    "context"  
    "google.golang.org/grpc"  
    "google.golang.org/grpc/metadata"  
    "time"  
)
```

```
func main() {  
    // 创建 gRPC 连接  
    conn, err := grpc.Dial("localhost:50051", grpc.WithInsecure())  
    if err != nil {  
        panic(err)  
    }
```

```

}

defer conn.Close()

// 创建一个新的上下文
md := metadata.Pairs("Authorization", "1234567890")
ctx := metadata.NewOutgoingContext(context.Background(), md)

// 创建 gRPC 客户端
client := pb.NewYourServiceClient(conn)

// 发起 gRPC 请求
response, err := client.YourRPCMethod(ctx, &request)
if err != nil {
    panic(err)
}

// 处理响应
// ...
}

```

```

APISIX配置示例：

①创建consumer

```

curl http://127.0.0.1:9180/apisix/admin/consumers \
-H "X-API-KEY: $admin_key" -X PUT -d '
{
    "username": "test",
    "plugins": {
        "key-auth": {
            "key": "1234567890"
        }
    }
}'

```

②创建route

```

curl http://127.0.0.1:9180/apisix/admin/routes/cloudbrigde -H "X-API-KEY: $ad
{
    "methods": [ "POST" ],
    "uri": "/gateway_protocol.Gateway/*",
    "plugins": {
        "key-auth": {
    }
}
}
```

```

    "header": "Authorization"
}
},
"upstream": {
"type": "roundrobin",
"nodes": {
"camunda-zeebe-gateway:26500": 1
},
},
"schema": "grpc"
}
}
}

```

③请求示例：

```
grpcurl -plaintext -proto gateway.proto -d '{"processDefinitionKey": "2251799817793033"}' -k \
--header 'Authorization: 1234567890' \
10.132.48.88:9080 gateway_protocol.Gateway/CreateProcessInstance
```

### 方案三：jwt-auth

jwt-auth方案通常用于身份验证，被用于在用户登录后生成一个安全的令牌，客户端在后续的请求中携带该令牌以进行身份验证。其中客户端需要通过APISIX提供的public-api申请Token，后续请求携带该Token以完成身份认证。用于申请Token的密钥信息储存在APISIX-Consumer中。

APISIX配置示例：

①创建consumer

```
curl http://127.0.0.1:9180/apisix/admin/consumers \
-H "X-API-KEY: $admin_key" -X PUT -d '
{
  "username": "test",
  "plugins": {
    "jwt-auth": {
      "key": "user-key",
      "secret": "my-secret-key"
    }
  }
}'
```

②创建route

```
curl http://127.0.0.1:9180/apisix/admin/routes/cloudbrigde -H "X-API-KEY: $ad
{
  "methods": ["POST"],
  "uri": "/gateway_protocol.Gateway/*",
  "plugins": {
    "jwt-auth": {}
  },
  "upstream": {
    "type": "roundrobin",
    "nodes": {
      "camunda-zeebe-gateway:26500": 1
    },
  }
},
```

```
    "schema": "grpc"
  }
}'
```

#### ③创建签发 token 的 API

```
curl http://127.0.0.1:9180/apisix/admin/routes/jas \
-H "X-API-KEY: $admin_key" -X PUT -d '
{
  "uri": "/apisix/plugin/jwt/sign",
  "plugins": {
    "public-api": {}
  }
}'
```

#### ④获取Token

```
curl http://127.0.0.1:9080/apisix/plugin/jwt/sign?key=user-key -i
```

#### ⑤请求示例

```
grpcurl -plaintext -proto gateway.proto -d '{"processDefinitionKey": "2251799817793033"}' -k \
--header 'Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJrZXkiOiJ1c2VyLWtleSIsImV4cCI6MTU2NDA1MDgxMX0.Usszh_4VjdXF-TmR5f8cif8mBU7SuefPlpxhH0jbPVI' \
10.132.48.88:9080 gateway_protocol.Gateway/CreateProcessInstance
```

## 方案四 : hmac-auth

HMAC ( Hash-based Message Authentication Code ) 是一种基于哈希函数的消息认证码算法，用于验证消息的完整性和真实性。在网络通信中，HMAC 可以用于身份验证，确保消息在传输过程中没有被篡改。

HMAC 认证通常涉及以下几个步骤：

1. 生成密钥：双方事先共享一个密钥，用于生成和验证 HMAC。
2. 消息摘要：发送方使用密钥和消息内容计算出 HMAC，并将其附加到消息中。
3. 消息传输：发送方将消息和 HMAC 一起发送给接收方。
4. 验证：接收方使用相同的密钥、接收到的消息内容以及接收到的 HMAC 来重新计算 HMAC，并与接收到的 HMAC 进行比较，以验证消息的完整性和真实性。

HMAC 认证可以用于各种网络通信协议，包括 HTTP、WebSocket、gRPC 等。在实际应用中，HMAC 认证通常需要双方事先共享密钥，并且需要对消息内容进行适当的编码和处理，以确保在计算 HMAC 时不会出现歧义或安全漏洞。

APISIX 原生支持 hmac 认证算法，验签过程原理上和 SCC 提供的 openapi\_auth 类似，但是性能要优秀 openapi\_auth 太多。

APISIX 配置示例：

#### ① 创建 consumer

```
curl http://127.0.0.1:9180/apisix/admin/consumers \
-H "X-API-KEY: $admin_key" -X PUT -d '
{
  "username": "jack",
  "plugins": {
    "hmac-auth": {
      "access_key": "user-key",
      "secret_key": "my-secret-key",
      "signed_headers": [ "User-Agent", "Accept-Language", "x-custom-header" ],
      "validate_request_body": true
    }
  }
}'
```

```
}
```

②创建route

```
curl http://127.0.0.1:9180/apisix/admin/routes/cloudbrigde -H "X-API-KEY: $ad

{
  "methods": [ "POST" ],
  "uri": "/gateway_protocol.Gateway/*",
  "plugins": {
    "hmac-auth": {}
  },
  "upstream": {
    "type": "roundrobin",
    "nodes": {
      "camunda-zeebe-gateway:26500": 1
    }
  },
  "schema": "grpc"
}
}'
```

```
import base64
import hashlib
import hmac

secret = bytes('my-secret-key', 'utf-8')
message = bytes("""GET
/index.html
age=36&name=james
user-key
Tue, 19 Jan 2021 11:33:20 GMT
User-Agent:curl/7.29.0
x-custom-a:test
""", 'utf-8')

hash = hmac.new(secret, message, hashlib.sha256)

# to lowercase base64
print(base64.b64encode(hash.digest()))
```

```
grpcurl -plaintext -proto gateway.proto -d '{"processDefinitionKey": "2251799817793033"}' -k \
-H "X-HMAC-SIGNATURE: 8XV1GB7Tq23OJcoz6wjgqTs4ZLxr9DiLoY4PxzScWGYg=" \
-H "X-HMAC-ALGORITHM: hmac-sha256" \
-H "X-HMAC-ACCESS-KEY: user-key" \
-H "Date: Tue, 19 Jan 2021 11:33:20 GMT" \
-H "X-HMAC-SIGNED-HEADERS: User-Agent;x-custom-a" \
-H "x-custom-a: test" \
```

```
10.132.48.88:9080 gateway_protocol.Gateway/CreateProcessInstance
```

## 方案选择

方案选择背景：同时支持HTTP和gRPC应用层协议；由于云桥连接器数量较大，所以必须保证互信方案简单有效，极限情况下需要支持5000并发。

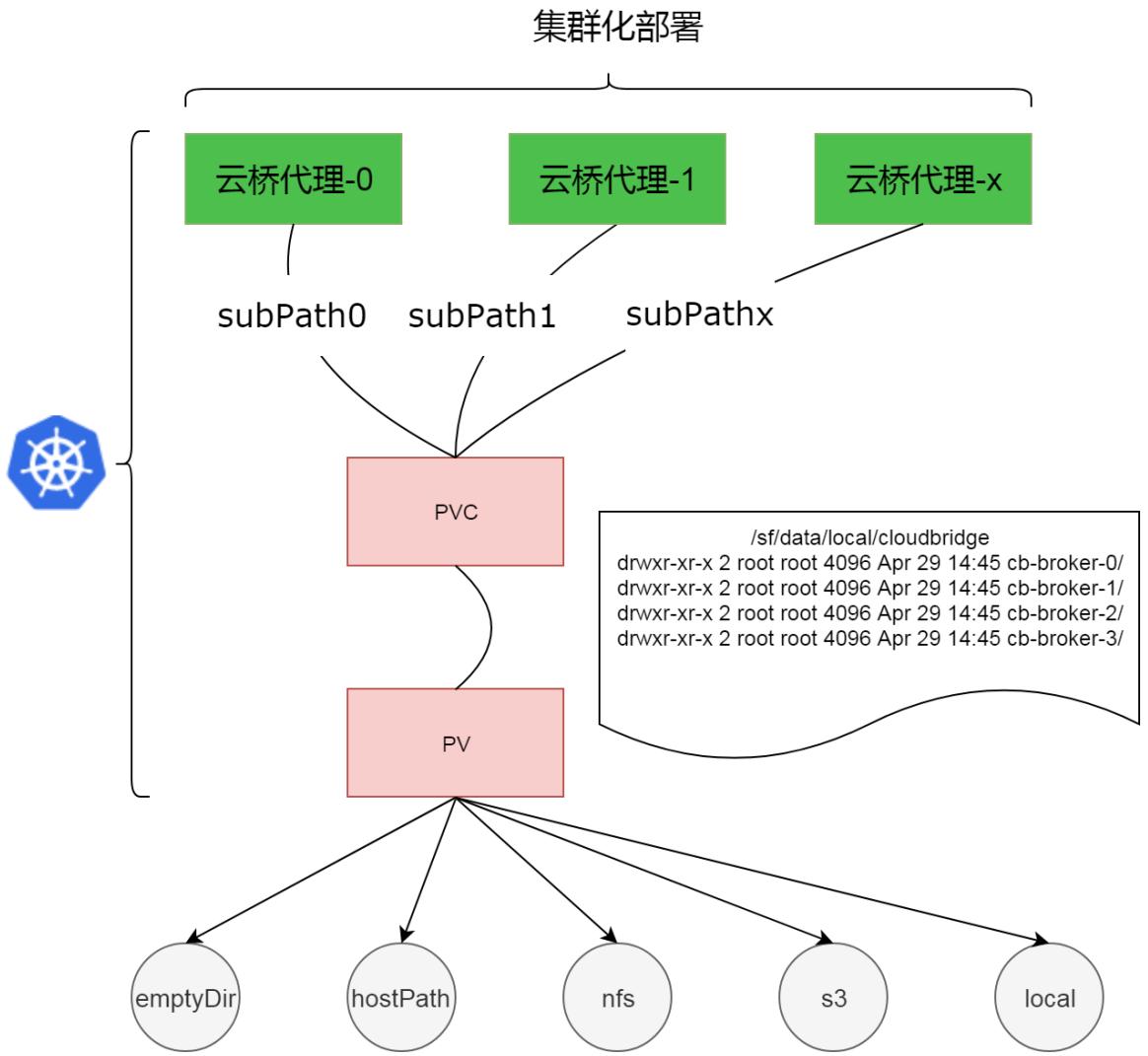
| 序号  | 备选方案名称              | 本方案的优点                                 | 本方案的风险和缺点                                                                                                               | 最终选择 |
|-----|---------------------|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------|------|
| 方案1 | AKSK+openapi_auth验签 | 1、现有网络通路，几乎没有工作量                       | 1、性能较差，生产环境不支持新增过多认证请求<br>2、openapi_auth不但提供验签的功能，还提供了生成后端Token的功能，后者在云桥场景是不需要的<br>3、采用该方案必须优化该插件的性能                    | √    |
| 方案2 | key-auth            | 1、实现较为简单                               | 1、对于云桥这种安全性要求较高的场景，API Key 无法提供足够的安全保障<br>2、存在重放风险<br>3、api-key写死不够灵活                                                   |      |
| 方案3 | jwt-auth            |                                        | 1、对于客户端来说，流程实现较为冗余，所有请求需要先调用颁发Token的接口，维护有效Token给客户端引入额外工作量<br>2、一旦颁发了 JWT，除非过期时间到达，否则无法撤销，可能会增加一定的安全风险<br>3、一般不用于服务间通信 |      |
| 方案4 | hmac-auth           | 1、lua实现，可以实现并发处理<br>2、可以有效防重放，有效防止数据篡改 |                                                                                                                         |      |

hmac-auth需要的access\_keysecret\_key与接入云端代理生成的AKSK一致，不过云端代理的AKSK仅储存在Keystone中和SkyOps文件中，需要在生成云端代理AKSK的时候，生成一份APISIX的Consumer写入etcd，云桥连接器在请求云桥中心端时封装对应的hmac头部，进而请求

采用AKSK+openapi\_auth验签的方式，同时需要解决Openapi\_auth存在的性能问题，具体方案如下：go-runner转lua插件，memcached分库，给Openapi\_auth配置一个单独的memcached缓存系统，验签后的不需要的流程通过头部的形式跳过。

#### 4.1.4. 云桥代理持久化存储方案

云桥代理采用集群化部署的形式，每个云桥代理容器采用本地磁盘来储存持久化数据，数据包含：流程实例数据、任务节点数据等。每个云桥代理的数据各不相同，所以云桥代理严格意义上是一个有状态的服务。



方案选择背景：①持久化数据不丢失②保证每个云桥代理无论在重新部署还是K8S主动杀死再拉起，都可以加载到同一份数据，即使云桥代理实例被K8S调度到其他集群节点。

| 序号+备选方案名称             | 方案介绍                                                                                                                               | 方案优缺点                                                                                                                                                     | 最终选择 |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| <b>方案1 : emptyDir</b> | emptyDir 是 Kubernetes 中的一种卷类型，它在容器之间共享存储空间。emptyDir 的生命周期与 Pod 相关联，当 Pod 被删除时，emptyDir 中的数据也会被清除。emptyDir 适合于临时存储数据，比如容器之间共享文件等场景。 | 1、当 Pod 被删除时，emptyDir 中的数据也会被清除， <b>无法做到持久化数据不丢失</b>                                                                                                      |      |
| <b>方案2: csi</b>       | SCC的csi(Container Storage Interface)目前采用minio作为对象存储服务器                                                                             | 1、目前测试来看，此种挂载方式直接造成了云桥代理的大文件存储出现 <b>性能瓶颈</b> ，原因为这个存储不适合文件内容频繁变更的场景，大文件每次变更1个字节，也会进行整个文件的上传                                                               |      |
| <b>方案3: nfs</b>       | NFS 卷允许将远程的 NFS 文件系统挂载到 Pod 中，提供了跨节点的共享存储功能。适合于需要共享数据的场景，但需要考虑网络 I/O 开销和一致性问题                                                      | 1、NFS作为一种网络文件系统，能够提供跨节点的数据共享和访问，从而为云桥引擎的多节点部署提供了便利<br>2、NFS具有良好的可扩展性和灵活性，能够满足不同规模和需求的存储场景，为系统的持久化存储提供了可靠的基础<br>3、NFS挂载依赖于网络，因此网络延迟或不稳定可能会影响到频繁读写操作的性能和稳定性 | √    |

|                      |                                                                     |                                                                                                                                                  |
|----------------------|---------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>方案4: hostPath</b> | hostPath 允许将集群节点上的文件系统目录直接挂载到 Pod 中                                 | 1、这种类型的卷适合于需要直接访问节点文件系统的场景，但是不具备跨节点的可移植性。使用 hostPath 的话我们就 <b>需要将 Pod 固定在某个节点上</b> ，这样显然就大大降低了应用的容错性                                             |
| <b>方案5: local</b>    | local卷 允许将集群节点上的文件系统目录挂载到 Pod 中。与 hostPath 卷相比，我们可以设置 local卷所在的节点位置 | 1、虽然Kubernetes 调度器会将 Pod调度到 local 卷所在的节点，但是local卷所在的节点位置是固定的，所以也 <b>限制了Pod需要固定在某个节点上</b><br><br>2、当 Pod 从一个节点迁移到另一个节点时，本地卷上的数据不会自动迁移，会导致持久化数据丢失。 |

方案总结：为了确保系统的高可用性，并且实现云桥代理的自由调度，我们选择采用节点无关的网络持久化存储方案NFS。由于NFS实现持久化存储依赖网络，所以我们尽量减少无关文件的同步对网络资源的占用，只同步关键数据。同时采用子目录的形式将云桥代理集群下不同的节点数据进行物理隔离，防止出现数据一致性的问题。

CloudBridgeBroker 部署时增加环境变量，以减少无关数据对网络带宽的占用：

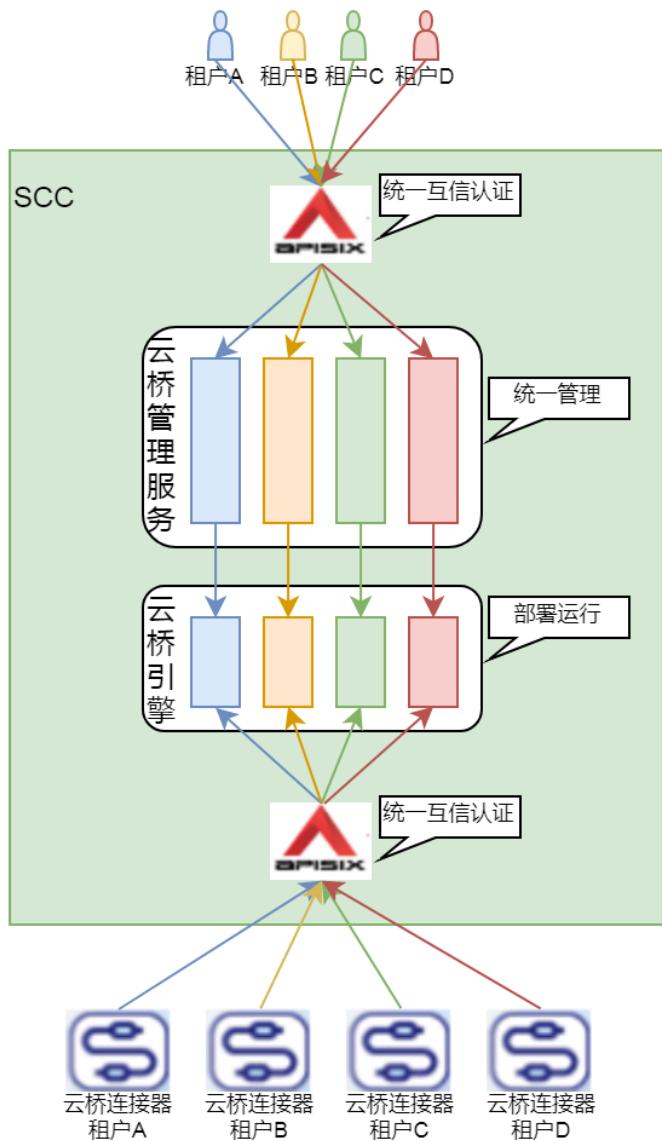
zeebe.broker.data

```
- name: ZEEBE_BROKER_DATA_LOGSEGMENTSIZE
  value: "10MB"
```

#### 4.1.5. 云桥多租户方案

云桥为客户提供SaaS服务，需要满足多租户使用场景，每个租户的数据和流程在逻辑上彼此隔离。这意味着一个租户的工作流程、数据模型和流程配置不会干扰或影响其他租户的操作。每个租户都在同一个云桥引擎集群内独立且安全的空间中运行。同时，每个租户都有自己的云桥连接器，用于从云桥引擎拉取任务并执行任务，我们要确保租户的云桥连接器只可以从云桥引擎拉到自己的任务。

多租户示意图：

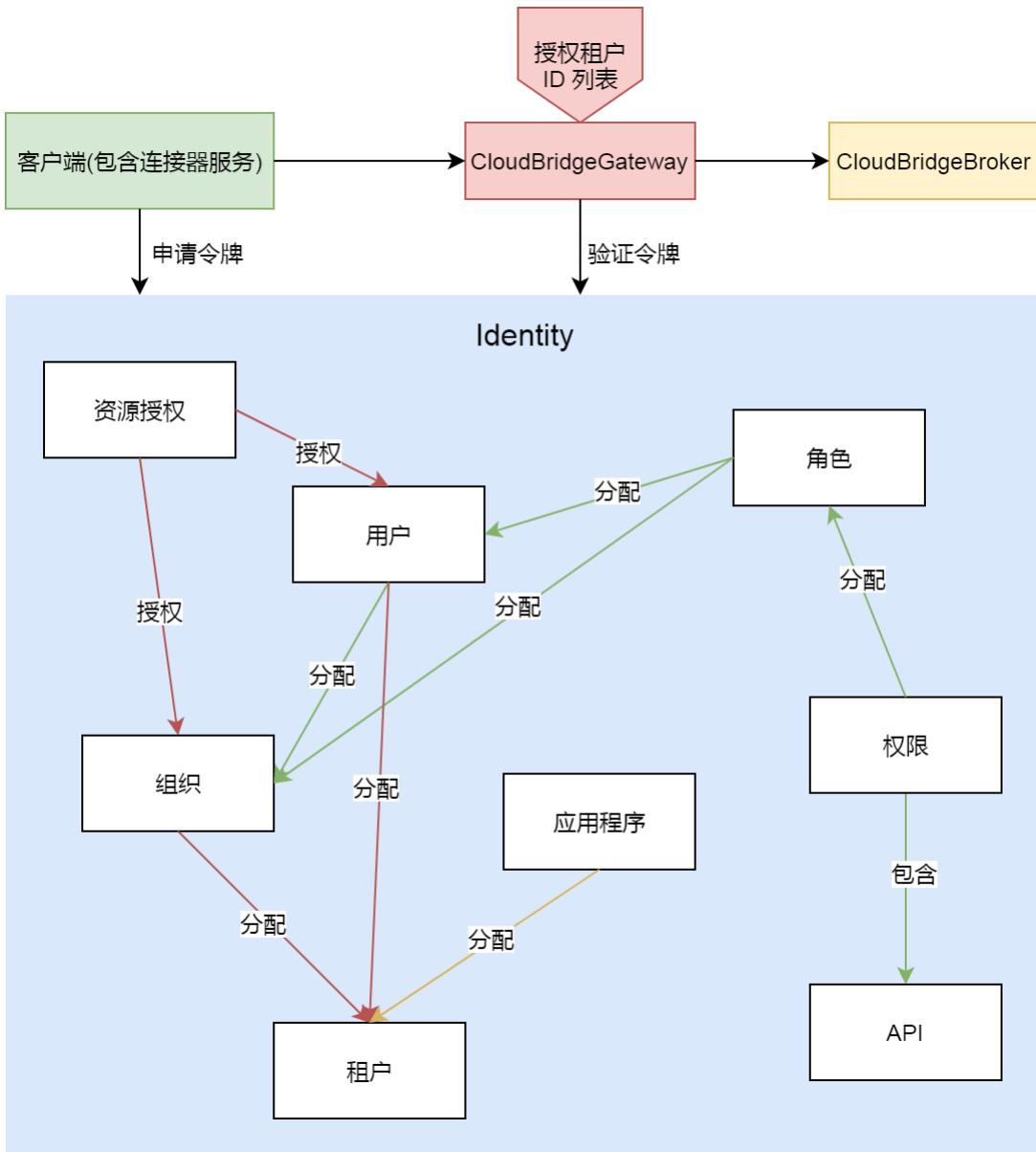


## 方案一：基于Identity实现

Identity是 Camunda8 中负责身份验证和授权的组件，基于Spring Boot。它可以实现：多租户管理、权限管理、角色管理、认证管理等功能。一般和第三方组件Keycloak配合使用，Keycloak负责数据存储。

在部署Identity之后，Identity会在Keycloak中为每个应用预创建用户、密钥信息并分配权限，包含Operate、Optimize、Tasklist、Webmodeler、Connectors，用于以上应用间的认证通信。

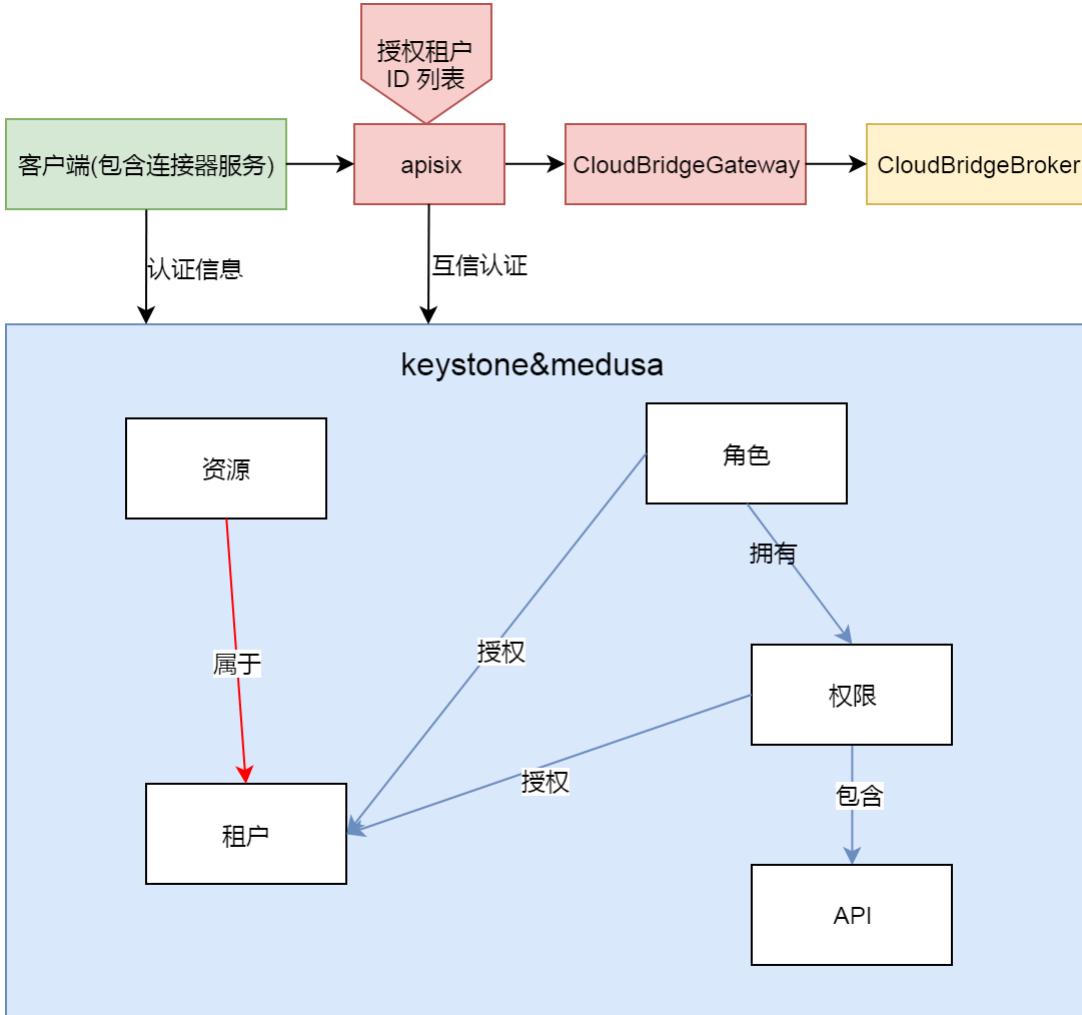
Identity提供了角色管理、权限管理、用户管理、组织管理、租户管理、应用程序管理、API管理、资源授权等功能，各个功能的关联关系以及云桥连接器访问云桥引擎的过程如下图：



作为一个应用程序(部署在同一网络区域的任何位置)，需要先接入Identity，并为该应用程序生成一份client\_id和client\_secret，当该应用程序需要访问另一个应用程序时，需要先向Identity生成请求令牌，进而携带该令牌对另一个应用程序进行访问。在云桥连接器访问资源授权的场景，当请求进入云桥网关(CloudBridgeGateway)时，云桥网关会负责调用Identity获取当前请求者已分配的分配租户ID列表，并在请求转发给 云桥代理(CloudBridgeBroker) 时添加授权租户列表到请求中，进而在云桥代理中根据租户列表进行过滤。

## 方案二：基于SCC已有租户体系

云桥全面接入SCC已经存在的租户认证体系，来实现租户隔离、互信认证、权限管控等功能，体系层级关系以及云桥连接器访问云桥引擎的过程如下：



客户端(包含连接器服务)访问云桥内API或流程资源时需要先经过平台的认证鉴权管道(在APISIX进行)，认证鉴权通过之后添加授权租户列表到请求中，进而在CloudBridgeBroker中根据租户列表进行过滤。

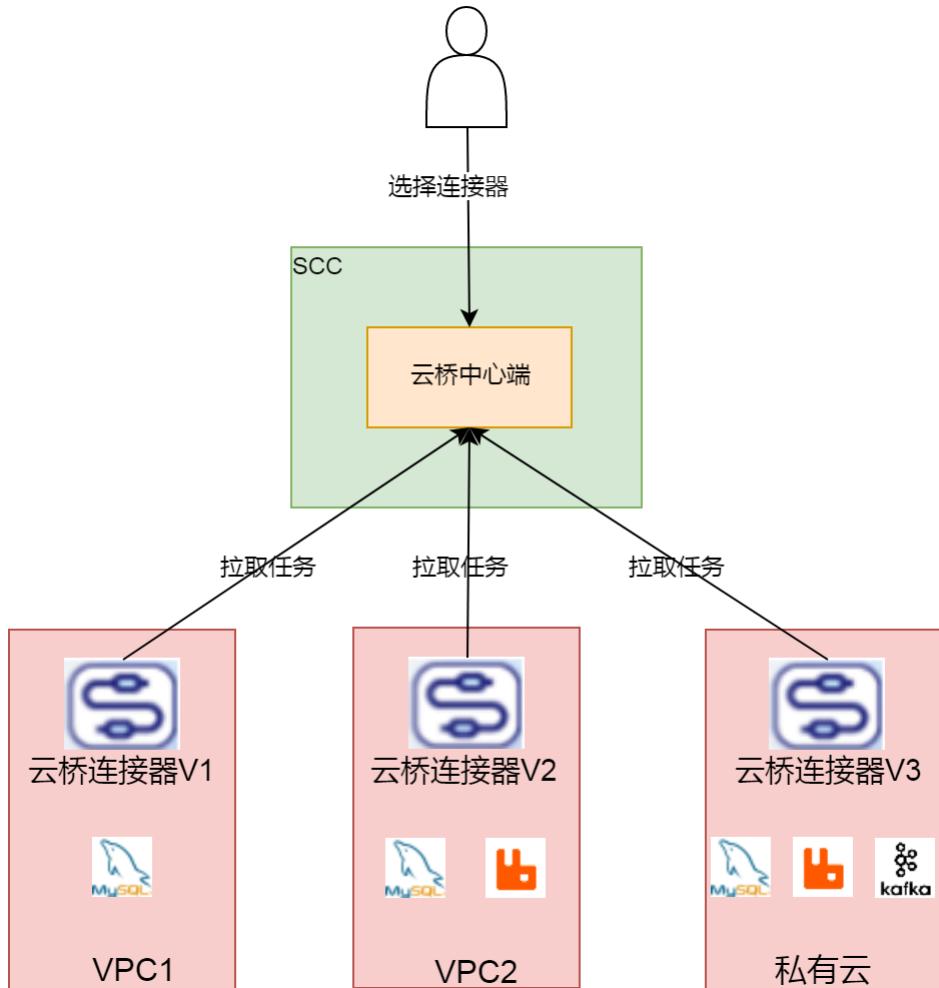
方案选择背景：①实现多租户管理②实现服务间的通信安全③实现安全的访问控制

| 序号  | 备选方案名称       | 本方案优点                                                                                                 | 本方案缺点                                                              | 最终选择 |
|-----|--------------|-------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|------|
| 方案1 | 基于Identity实现 | 1、Identity提供了较为全面的多租户管理能力<br>2、实现RBAC级别的访问控制                                                          | 1、Identity组件许可证不允许部署该组件到生产环境<br>2、需要将SCC用户体系接入Identity，同时构建一整套管理流程 |      |
| 方案2 | 基于SCC已有租户体系  | 1、直接使用SCC用户体系，无需再引入一套新的用户体系，无需考虑用户同步的管理流程<br>2、复用SCC已经提供的认证体系、权限体系，统一在网关层进行，架构统一<br>3、可以实现ABAC级别的访问控制 |                                                                    | √    |

SCCWebAPIOpenApiTLSID流程定义、流程实例、流程作业对应的租户ID，来实现租户隔离的能力；**云桥连接器**使用SCC颁发的互信凭证，实现统一的接入安全，同时互信凭证与租户ID一一对应，在拉取任务、完成任务、启动流程、获取流程模型过程中，均保证只可以操作互信凭证对应的租户的资源。

#### 4.1.6. 云桥连接器多版本管理方案

云桥连接器在云桥架构中，负责通过云桥引擎端拉取对应的任务并进行执行，这个任务包含触发任务和执行任务。根据版本的不同，每个云桥连接器支持的应用是不同的。客户在UI创建流程编辑任务节点时，需要指定该任务节点在哪个云桥连接器中执行，同时需要展示该云桥连接器支持的应用类型，客户只能选择这些应用类型。



#### 4.1.7. 应用密钥管理方案

创建集成流程需要租户输入应用的密码，应用密码存储有以下三个方案：

| 序号  | 备选方案名称                       | 本方案优点                                                                                                                                                                | 本方案缺点                           | 最终选择 |
|-----|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|------|
| 方案1 | 加密储存在SCC，对称加密以AK/SK中的SK作为密钥  | 1、SCC侧：CloudBridge-API和云桥引擎均储存加密的密码，保证安全<br>2、密码传输过程：对称加密的密文、TLS加密，以及互信认证<br>3、客户侧密码存储方式：不储存<br>4、密码使用：云桥连接器服务拿到的是加密密码，用既定的算法解密后使用<br>5、每个租户的SK安全性较高，不会被泄露<br>6、简单直接 | 1、需要连接器上报对应云端代理的AK上来，我们通过SK作为密钥 |      |
| 方案2 | 加密储存在云端代理，对称加密以AK/SK中的SK作为密钥 | 1、SCC侧：SCC储存的密码为uuid的占位符，不感知客户真实密码                                                                                                                                   | 1、需要下发密钥信息到云端代理，存在密钥信息的维护成本     |      |

|     |                                                                     |                                                                                                                                                                  |                                                                          |
|-----|---------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|
|     |                                                                     | <p>2、密码传输过程：传输UUID、TLS加密，以及互信认证</p> <p>3、客户侧密码存储方式：对称加密的密文</p> <p>4、密码使用：云桥连接器服务拿到的是UUID，在使用时获取真实密码</p> <p>5、每个租户的SK安全性较高，不会被泄露</p>                              |                                                                          |
| 方案3 | <p>加密储存在云端代理，非对称加密</p> <p>scc使用云端代理的公钥加密下发，云桥连接器使用密码时，使用自己的私钥解密</p> | <p>1、SCC侧：SCC储存的密码为uuid的占位符，不感知客户真实密码</p> <p>2、密码传输过程：传输UUID、TLS加密，以及互信认证</p> <p>3、客户侧密码存储方式：公钥加密的密文</p> <p>4、密码使用：云桥连接器服务拿到的是UUID，在使用时获取真实密码</p> <p>5、安全性更高</p> | <p>1、需要下发密钥信息到云端代理，存在密钥信息的维护成本</p> <p>2、SCC需要维护所有的云桥连接器的公钥，存在公钥的维护成本</p> |

**结论：**先采用方案一，逐步向方案三演进。

**方案二、三保存密码到云端代理的解决方案：**

- 考虑将密码存储在租户VPC中，具体位置为：VPC内云桥连接器所在云端代理VM的SQLite数据库中
- 用户在UI创建流程A，如果流程中的节点存在密码数据，在创建流程的后端实现中先保存密码到客户的VPC中，保存方式为：启动一个保存密码的流程实例，指定需要保存该密码的VPC位置，保存密码的流程定义如下：



- 对应VPC内的云桥连接器拉取Save Secret对应的job，输入参数是密码数据和一个UUID(SCC负责生成UUID)
- 云桥连接器执行该job，将密码数据(包含UUID、租户信息等)保存在云端代理的sqlite中。
- 创建流程的实现同步等待该流程实例跑完，拿到流程实例的结果是"YES"时，将UUID代替流程中的密码数据，之后保存流程到数据库中
- 对于真正执行上述的流程A的实例时，云桥连接器拉取Job，根据UUID，去sqlite中查询真实密码，执行具体的连接任务。

#### 4.1.8. 云桥引擎导出器方案

Elasticsearch

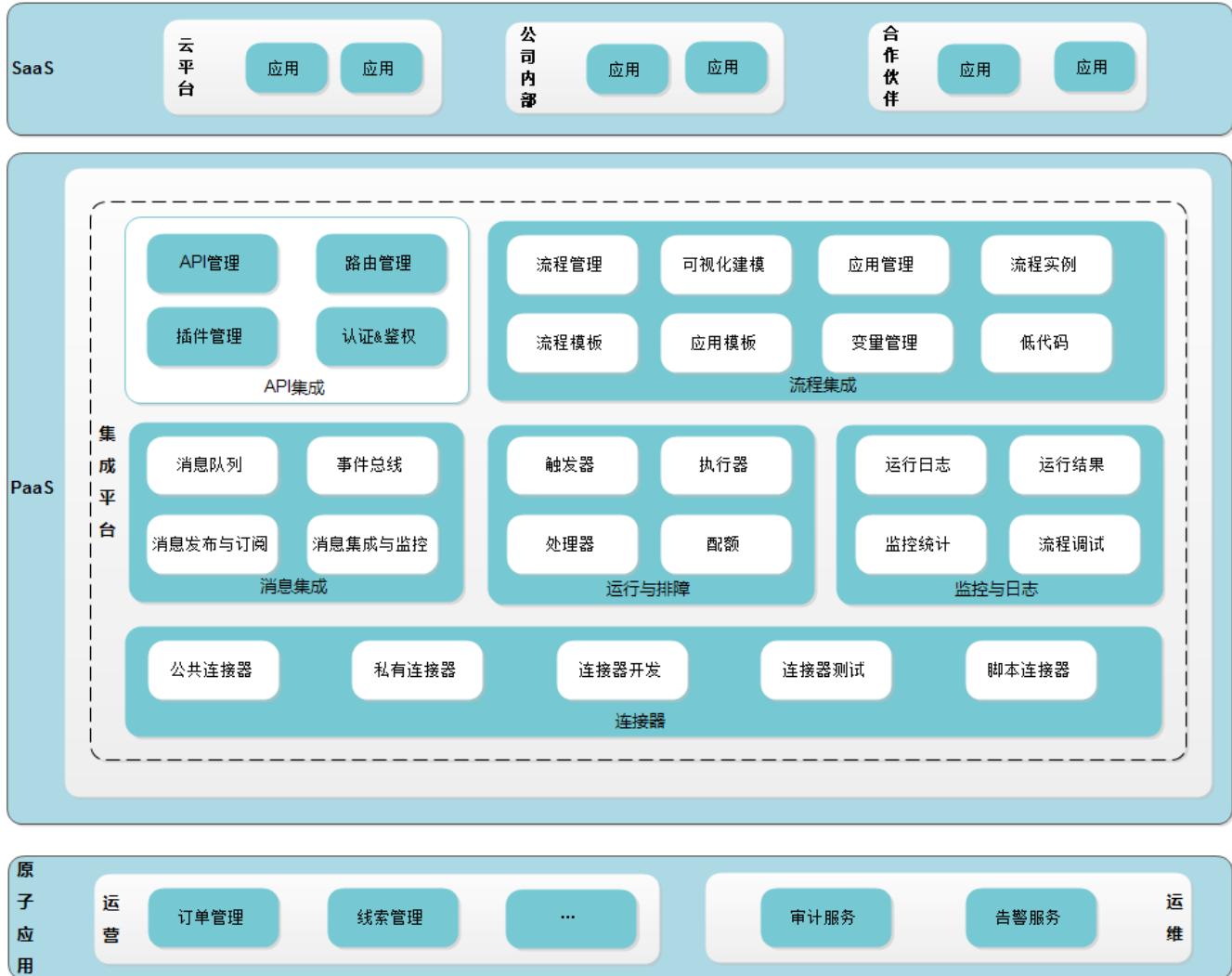
ClickHouse

要考虑：查询的实现以及性能

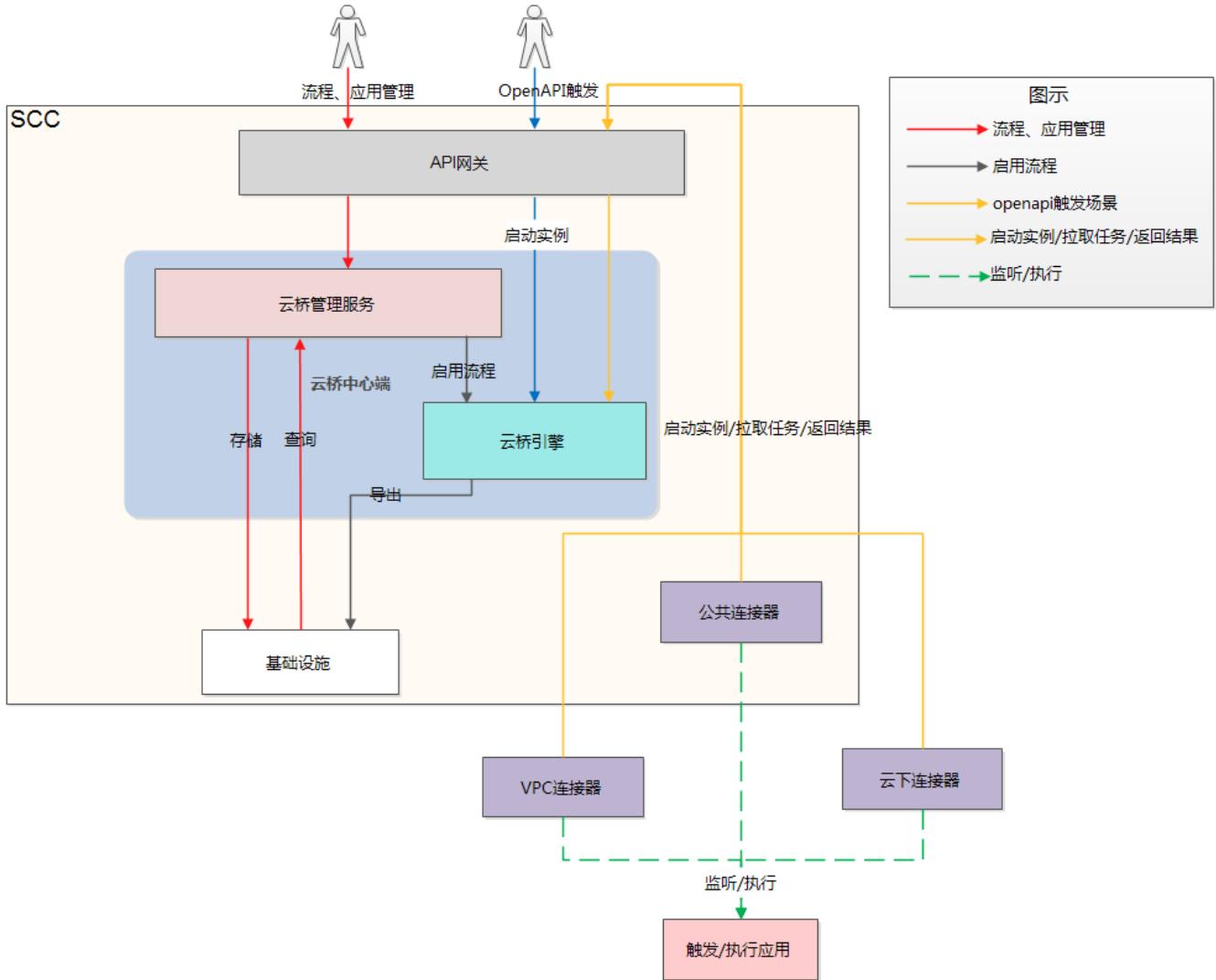
#### 4.2. 软件总体架构

系统的静态视图，重点是系统的软件模块构成，包括模块名、功能说明以及模块之间的层次(依赖)关系。在系统设计之初(在分析系统流程图之前)，我们根据需求和经验初步划分软件功能模块，以及模块的依赖关系和通信模型，然后分析系统流程图，不断的修正系统的模块划分、模块的依赖关系和通信模型，最终得出模块的接口说明文档，而模块以及模块的接口说明文档，是概要设计阶段的工作入口和依据。

## 4.2.1.应用架构图



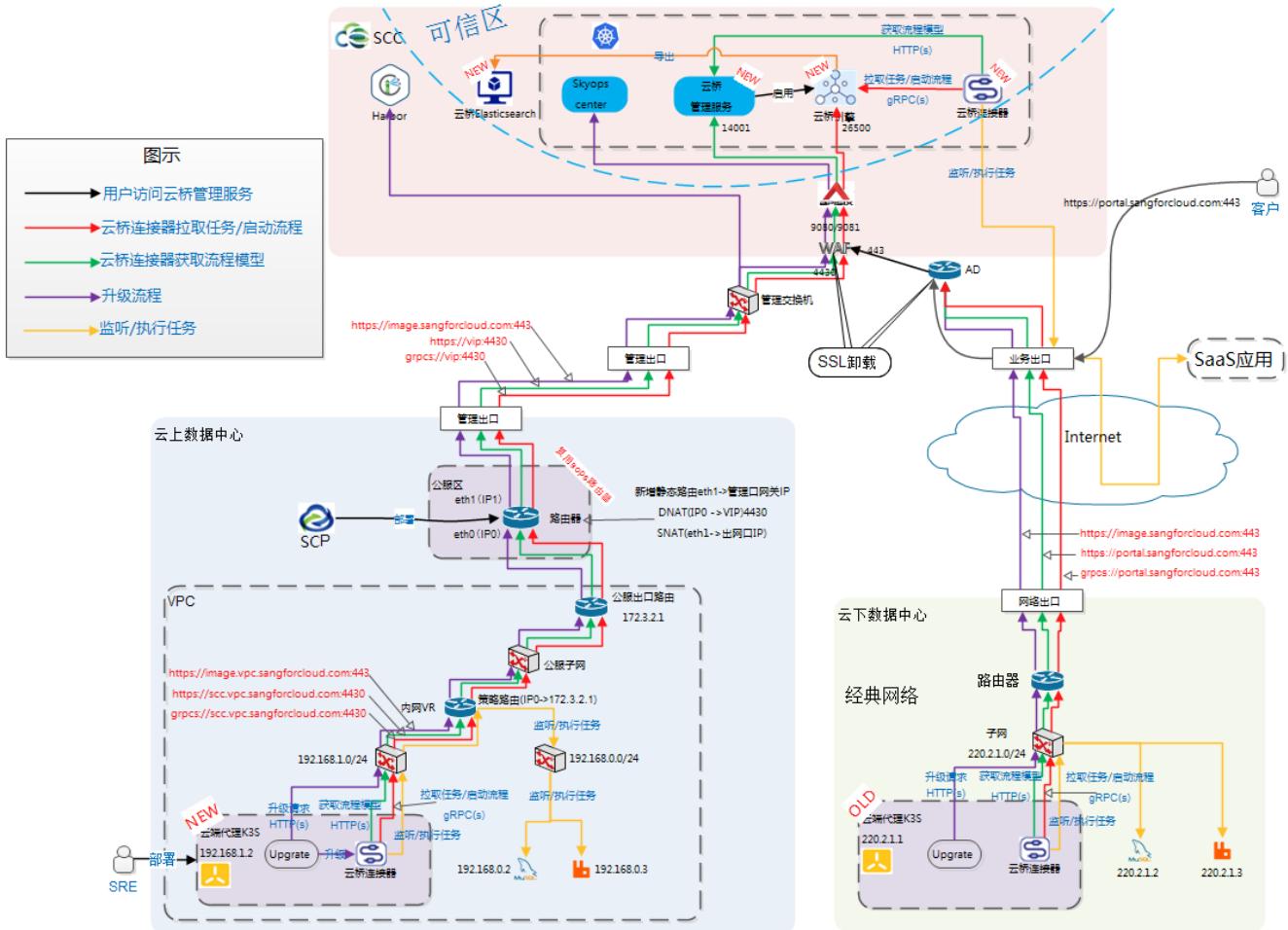
## 4.2.2.业务架构图



#### 架构约束：

- 1、云桥管理服务：整个云桥IaaS的管理平台，对外提供WebAPI和OpenAPI，主要提供模板管理、流程管理、应用管理等能力
- 2、云桥引擎：一种状态机引擎，负责处理复杂的业务逻辑和流程变化，当满足某个转换条件时，会自动触发相应状态转换，并产生相应的任务
- 3、云桥连接器：分为公共连接器、VPC连接器、云下连接器，用于解决不同网络场景下的应用连接；云桥连接器会且仅会做两件事情①从云桥引擎拉取任务并执行任务并返回执行结果②监听某一个应用，一旦达到触发条件，启动流程实例
- 4、API网关：负责统一互信认证，统一SSL卸载

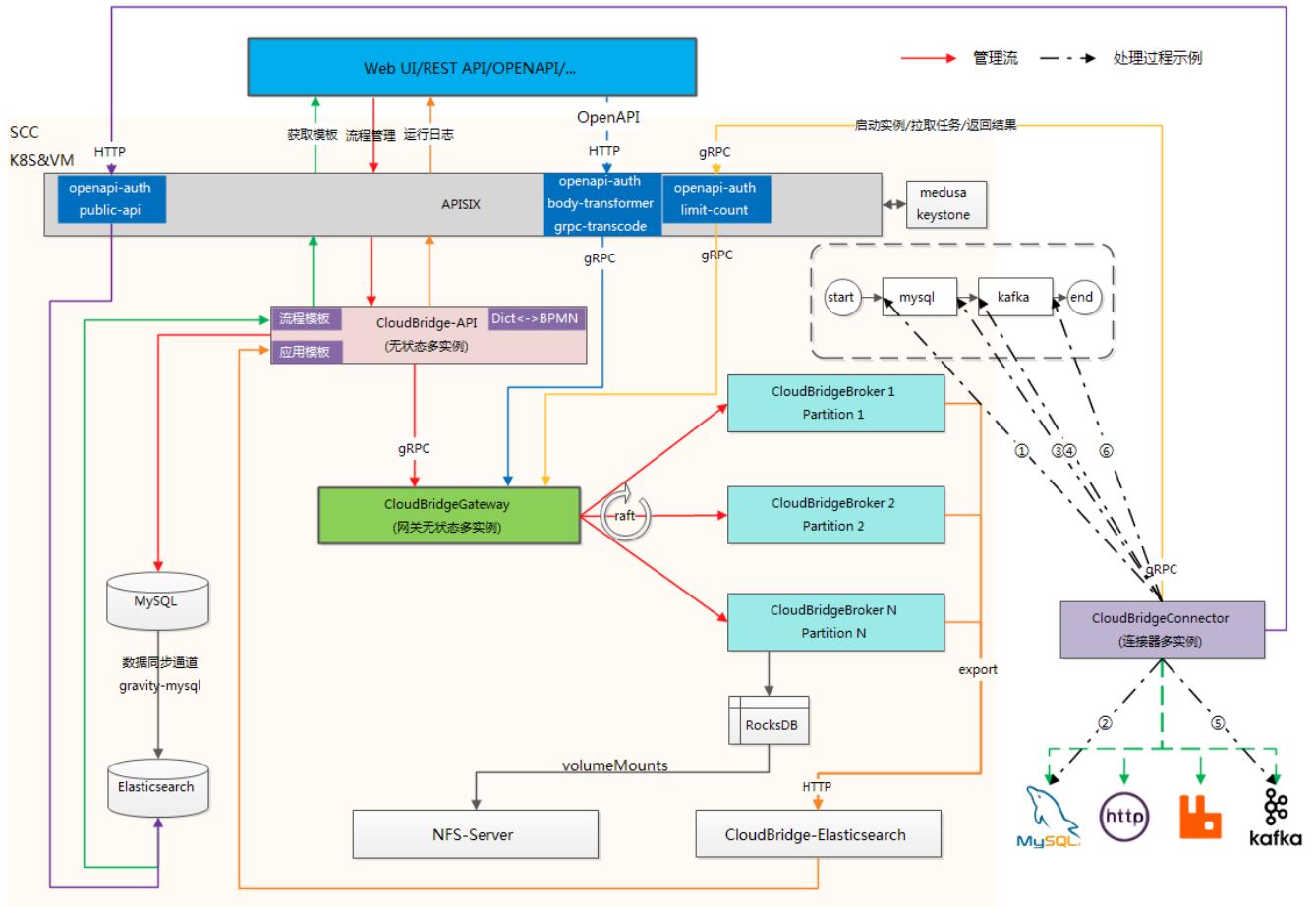
### 4.2.3. 总体部署架构图



#### 架构约束：

- 1、云桥管理服务：整个云桥IPaaS的管理平台，对外提供WebAPI和OpenAPI，为客户提供管理服务，为云桥连接器提供查询功能。
- 2、云桥引擎侧：集群部署，对外暴露gRPC端口，APISIX负责统一互信认证，AD或WAF负责SSL卸载。
- 3、云桥连接器：触发器和执行器，部署在云端代理VM中，K3S管理。云上VPC出向流量，走公服网络，通过管理出口与云桥引擎通信；云下通过云端代理走公网访问云桥引擎。
- 4、所有云桥连接器可以通过云端代理，进行升级管理，skyops-center维护版本信息，harbor作为制品仓库。

#### 4.2.4 技术架构图

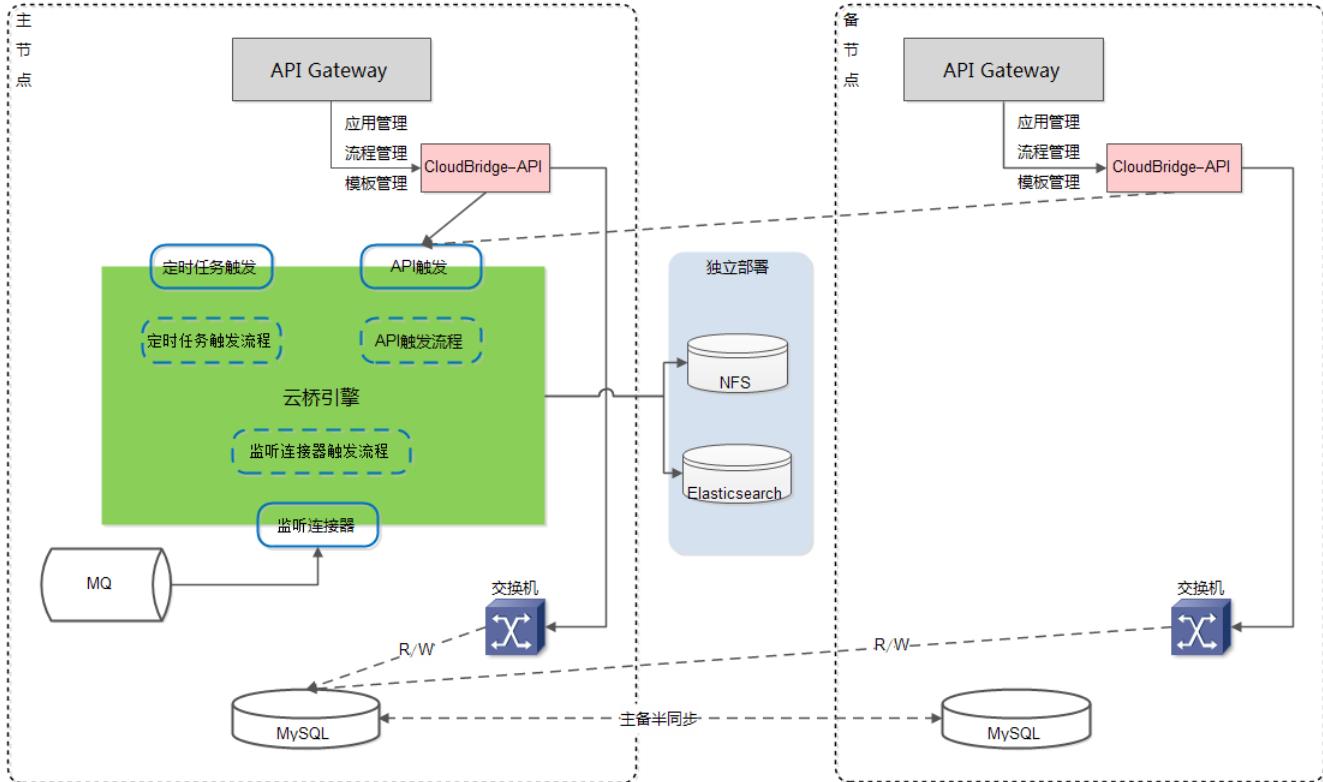


架构约束：

#### 4.2.5.云桥双活架构

SCC生产环境目前为异地双活架构，如果在主备节点分别部署一套云桥引擎，由于两个云桥引擎无法共享数据，将会存在以下影响：

- ①在单个节点的云桥引擎中进行流程部署，需要同步到另一个节点的云桥引擎中
- ②定时任务触发、监听连接器触发的流程会在两个云桥引擎中同时启动，产生重复调用



架构总结：只在主节点k8s集群部署云桥引擎和云桥连接器，但是CloudBridge-API会部署(k8s方式)到主备数据中心上。当主节点整体故障的时候，备节点切换为主节点，这个时候会重新拉起云桥引擎，落盘的数据通过挂载的方式保证一致。云桥连接器可以继续执行任务。

## 4.3 系统流程

系统的动态视图，重点描述系统功能的实现过程。通过各模块的交互，实现所有系统功能。通过分析系统流程图，导出各模块的接口。依据用例或系统功能需求编写系统流程图。每个用例需要对应一个系统流程。每个功能需求点都要描述其实现方案（模块）或者流程。对于规模较大的版本，整体架构只需要定义到子系统一级，系统流程只需要描述子系统之间的交互流程和接口即可。针对每个子系统的设计在子系统概要设计中完成。

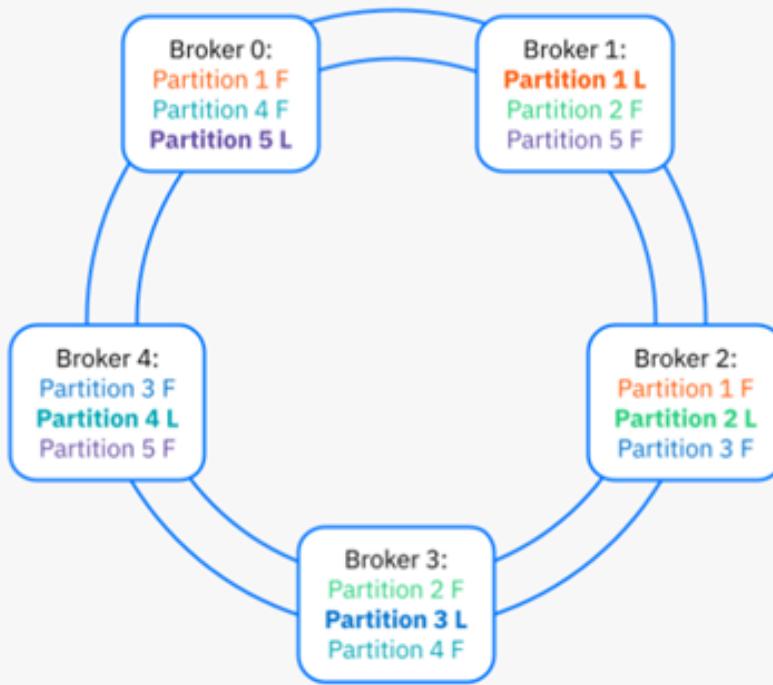
### 4.3.1. 云桥引擎核心实现

云桥引擎能做到高吞吐、高可用的微服务编排，得益于三个关键实现：

#### 4.3.1.1 分布式

云桥引擎内部抽象了一个只追加写的队列(可以类比理解成kafka的topic)，来处理和存储数据。当集群有多个broker节点时，会将队列划分成多个分区(partitions)，分布到各个节点上。每个分区有多个副本(replicas)。在所有的副本中，会根据raft协议选出一个leader，leader负责接收请求和执行所有处理逻辑。其他broker上的副本就是被动的跟随者(passive followers)。当leader不可用时，followers会透明地选出新的leader。

# Partitions (Shards) and Replication using Raft



Example:

- 5 Brokers
- 5 Partitions
- Replication factor 3
- **L = Leader**
- **F = Follower**

## 4.3.1.2.消息驱动

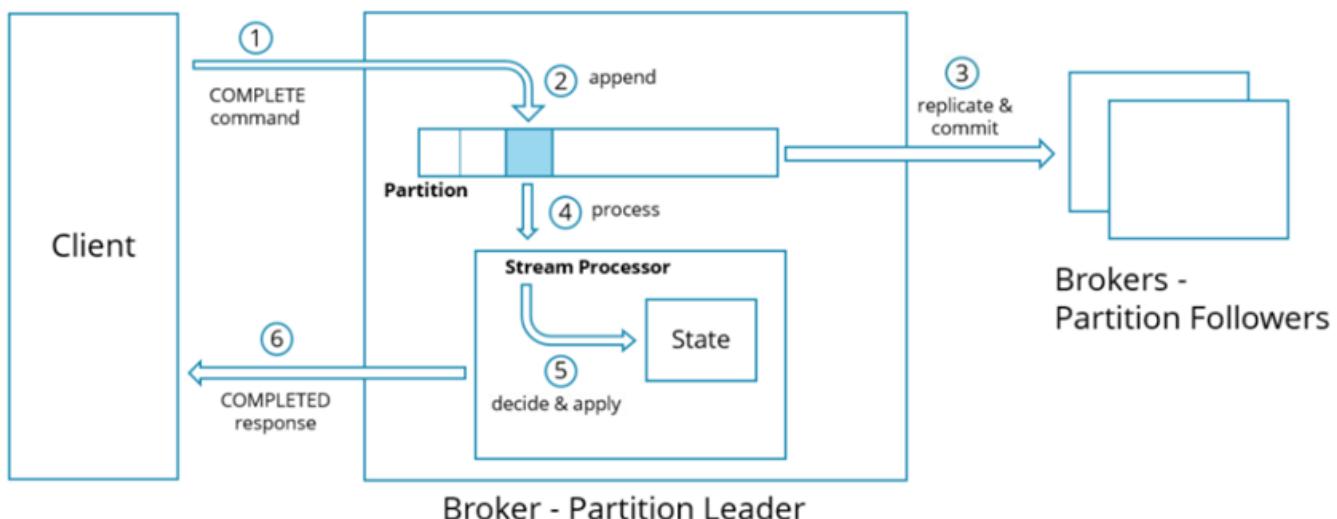
云桥引擎消息驱动架构，体现在两个方面：

①CloudBridgeBroker内部使用队列(即LogStream，只追加写)，异步处理请求

②CloudBridgeConnector和CloudBridgeBroker使用发布订阅的模式交互，当工作流任务状态发生变化，CloudBridgeBroker会发布相应事件。CloudBridgeConnector通过轮询的方式，订阅处理自己相关的事情。

云桥引擎内部实现，其实就是通过记录流(record streams)与流处理器(stream processors)实现的。流处理模型作为一个统一的实现方式，提供：

- 指令协议(command protocol，即请求响应)
- 记录导出(record export / streaming)
- 工作流演算(evaluation, 异步后台任务)



## 4.3.1.3.运行时数据和历史数据分离

当云桥引擎处理任务、工作流或者内部维护时，会产生有序的记录流：

|                                         |                                          |                        |                         |                          |                          |
|-----------------------------------------|------------------------------------------|------------------------|-------------------------|--------------------------|--------------------------|
| 1001:<br>Workflow<br>Instance<br>CREATE | 1002:<br>Workflow<br>Instance<br>CREATED | 1003:<br>Job<br>CREATE | 1004:<br>Job<br>CREATED | 1005:<br>Job<br>COMPLETE | 1006:<br>Job<br>COMPLETE |
| ...                                     |                                          |                        |                         |                          | ...                      |

①运行时数据：云桥引擎的运行时数据存在嵌入式键值数据库RocksDB，来保证性能。RocksDB是一个高性能、持久化的键值存储引擎，它基于Google的LevelDB进行了优化和改进。RocksDB具有快速的写入和读取性能，并且能够有效地处理大规模数据集。RocksDB使用了一些优化技术，如内存管理、压缩和多线程处理，以提高性能和效率。此外，RocksDB还支持持久化存储，确保数据在系统重启或故障恢复后的可靠性。

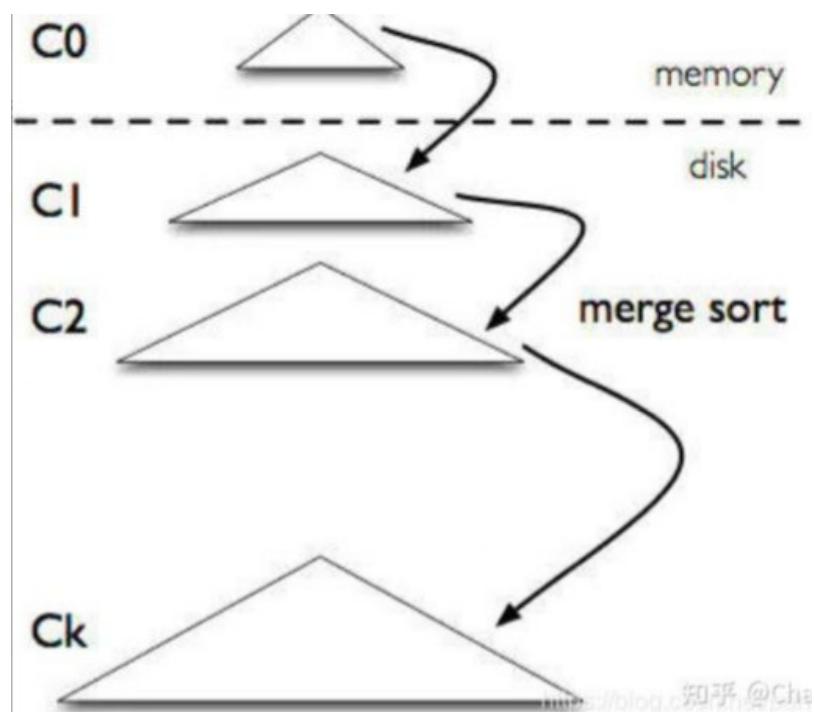
②历史数据：云桥引擎没有直接对外提供查询记录流的接口，而是使用统一的exporter机制，把历史数据推到外部数据仓库中，持久化历史数据，我们采用的外部数据仓库为Elasticsearch。对于集群化部署的云桥引擎来说，每一个分区会且仅会运行一个exporter实例，用于导出历史数据。云桥对外提供的监控数据就是来源于这份导出数据。

#### 4.3.1.4.RocksDB

云桥引擎需要频繁读写运行时数据。而B+树读效率高而写效率差；log型文件操作写效率高而读效率差。在两个解决方案之间做个折中，就引入了log-structured merge tree模型。LSM既有日志型的文件操作，提升写效率，又在每个sstable中排序，保证了查询效率。

LSM-tree 的组成部分，是一个多层次结构，就更一个树一样，上小下大。

首先是内存的 C0 层，保存了所有最近写入的 ( k , v )，这个内存结构是有序的，并且可以随时原地更新，同时支持随时查询。剩下的 C1 到 Ck 层都在磁盘上，每一层都是一个在 key 上有序的结构。

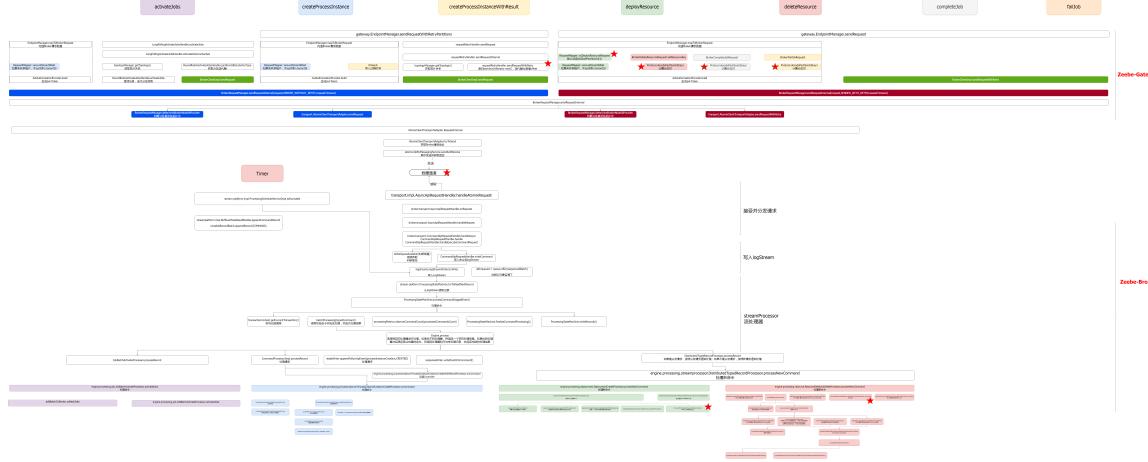


写入流程：追加到写前日志 ( Write Ahead Log，也就是真正写入之前记录的日志 ) 中，接下来加到 C0 层，然后逐层向下合并即compact。

查询流程：在写入流程中可以看到，最新的数据在 C0 层，最老的数据在 Ck 层，所以查询也是先查 C0 层，如果没有要查的 k，再查 C1，逐层查。

#### 4.3.1.4.源码分析

Gateway的处理逻辑：构造Broker接受的请求体；如果开了多租户，塞入租户ID列表；生成一份jwtToken；判断请求该向哪个分区发送。



#### 4.3.1.5. 数据不能出VPC→Zeebe必须部署在VPC中，不能在公服(数据会出VPC)

zeebe-Gateay的作用对外提供gRPC接口以及封装zeebe的请求体，所以Zeebe-Gateway部署在VPC外客户数据还是会出VPC。

所以考虑把Zeebe-gateway和Zeebe和连接器都部署在VPC内。

**方案一：**云桥管理服务在SCC，客户构建流程可以选择VPC内的实例，把流程下发到VPC中，这样流程的流转和执行都在VPC内，VPC和SCC的通信只有下发流程。

**弊端：**占用客户更多资源预计4c8g还不够，需要加2c4g，运行日志需要额外处理，一个流程该不能做多云集成。

**方案二：**沿用现有架构，数据用完销毁，不在中心端储存

**优点：**单个流程可以做多云集成，管理起来较为简单

**方案选择：**建议两个方案都落地，对数据安全要求较高的客户可以自己部署一套云桥，这样数据不会出VPC，但是多云需要客户自己打通网络；对数据安全要求稍低的，将流程部署在云上的引擎中，优势是可以做多云集成，不需要跨VPC打通网络，可以实现公网应用调用。

#### 4.3.2. 云桥连接器核心实现(应用快速上线)

云桥连接器的概念由两部分组成：

- ①业务逻辑由云桥连接器函数实现并由云桥连接器进程执行(4.3.1和4.3.2会详细介绍)
- ②客户建模期间的用户界面是由[应用模板](#)提供的，包含触发器应用模板和执行器应用模板

云桥实现低代码的关键为我们封装好的[应用模板](#)，模板里面声明了使用单个应用需要用户填写的信息，这个信息一定是和云桥连接器函数实现对应的。前端需要根据应用模板动态渲染输入页面，同时后续新增支持的应用时，可以做到快速上架而不需修改前端代码。

## 执行动作配置

X

MySQL – 删除记录  
MySQL\_01

① 选择动作对象    ② 选择执行应用    ③ 配置应用信息    ④ 配置动作参数

MySQL     SQL Server     定时器     OpenAPI

## 执行动作配置

X

MySQL – 删除记录  
MySQL\_01

① 选择动作对象    ② 选择执行应用    ③ 配置应用信息    ④ 配置动作参数

运行位置:  托管云  公共网络

数据库名称:

IP地址:

端口:

用户名:

密码:

连通性测试:

示例:

MySQL执行器模板

```
{  
  "id": "240bdb36-6e43-471e-b1c7-2618bdbdaff2",  
}
```

```

"name": "MySQL",
"description": "MySQL",
"categories": [],
"availability": true,
"action_type": "executor",
"applies_to": "bpmn:Task",
"steps": {
    "steps_details": [
        {
            "key": "app_level_configure",
            "title": "",
            "substeps": [
                {
                    "title": "",
                    "config_items": [
                        {
                            "id": "app-name",
                            "label_display": "",
                            "description": "",
                            "type": "string",
                            "required": true,
                            "default_value": "MySQL_01",
                            "support_expression": "no"
                        },
                        {
                            "id": "mysql-host-address",
                            "label_display": "IP",
                            "description": "",
                            "type": "ipv4",
                            "required": true,
                            "binding": {
                                "type": "cloudbridge:input",
                                "name": "proxyLoc
(databaseConnection).host"
                            },
                            "support_expression": "no"
                        },
                        {
                            "id": "mysql-port",
                            "label_display": "",
                            "description": "",
                            "type": "port",
                            "required": true,
                            "binding": {
                                "type": "cloudbridge:input",
                                "name": "proxyLoc
(databaseConnection).port"
                            },
                            "support_expression": "no"
                        }
                    ]
                }
            ]
        }
    ]
}

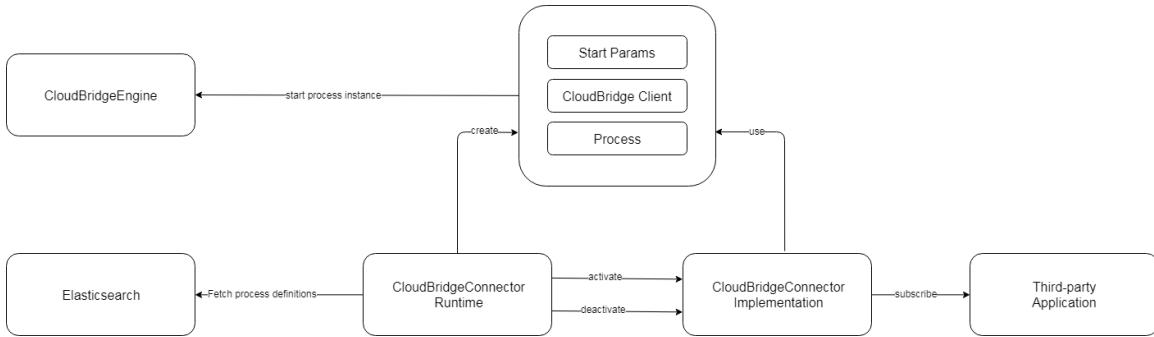
```

}

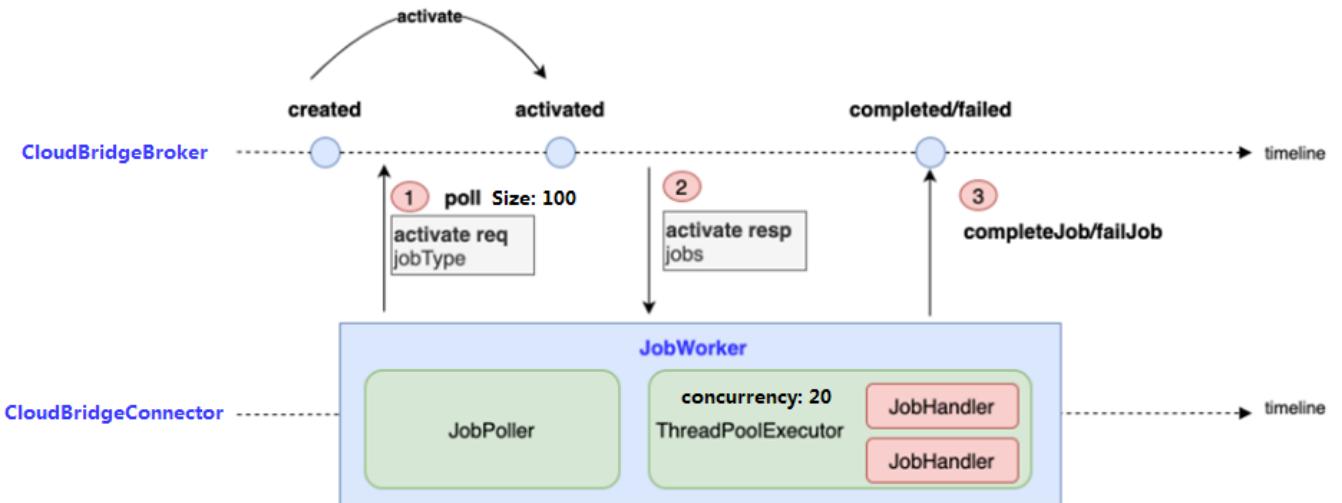
### 4.3.3 云桥连接器触发器实现

云桥连接器能够从外部应用接收数据或消息，从而可以将工作流集成到更广泛的业务流程或系统架构中。

当云桥连接器检测到部署的流程定义中使用了触发器启动的事件时(从SCC中心端获取)，监听消息的触发器代码将被激活，在流程部署期间创建的触发器对象将在整个部署期间保持活动状态，并且被重用于为所有流程实例提供服务，一旦达到触发条件，云桥连接器将会请求云桥引擎触发新的流程实例。如果流程定义被更新或删除，触发器代码将被停用。意味着当重新部署云桥连接器或修改流程时，云桥连接器会重新加载消息驱动的触发器对象。

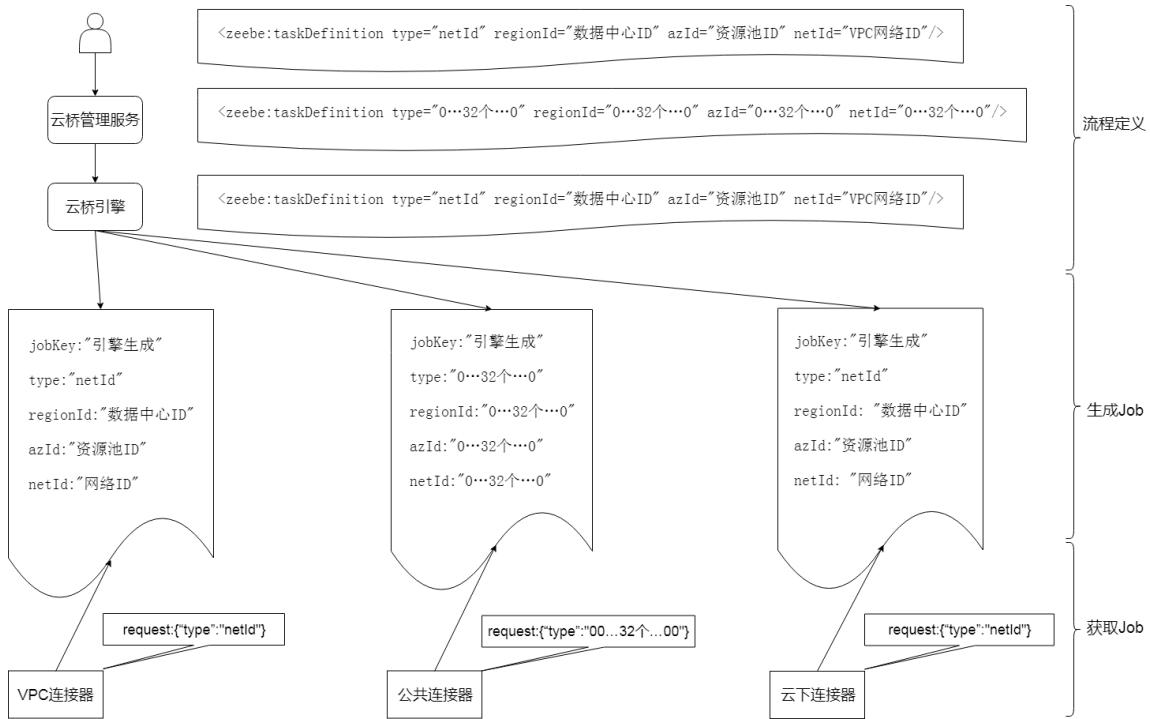


### 4.3.4 云桥连接器执行器实现



客户构建的自动化业务流程中所有外部任务均由云桥连接器负责执行，实现逻辑为：云桥连接器服务在启动时会启动一个线程，轮询云桥引擎获取固定类型的任务，当云桥引擎激活了**对应类型**的任务时，任务被调度到云桥连接器服务进行处理。

这里有几个关键问题需要实现：①保证每个云桥连接器服务只可以获取到自身要执行的任务。



```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: my-network-policy
  namespace: scc
spec:
  podSelector:
  matchLabels:
    app.kubernetes.io/name: connectors
  egress:
  - to:
    - ipBlock:
      cidr: 10.132.48.00/24
  ports:
  - protocol: TCP
    port: 9080
  - protocol: TCP
    port: 9081
  policyTypes:
  - Egress

```

#### 4.3.5. 定时启动

定时启动一般用于解决需要周期启动的重复性任务或定时启动的任务。



当部署一个流程时，云桥引擎会为每个定时启动事件安排一个计时器。当定时器被触发时，会创建一个新的流程实例，并激活相应的定时器启动事件。

定时器分为固定时间定时器、倒计时定时器、循环定时器三种。

- 固定时间定时器

在特定的时间内启动的计时器，如：2012-04-30T12:00:00Z

- 倒计时定时器

设置一个倒数时间，倒计时结束时触发启动，如：P3Y6M4DT12H30M5S



- 循环定时器

重复间隔格式的循环，包含持续时间和重复次数。如果未定义重复次数，则计时器将无限重复，直到被取消。

R5/PT10S: 每 10 秒一次，最多 5 次

R/P1D: 每天，无限循环

0 0 7 \* \* \*: 每天早上七点

注：关于cron表达式的规则可以参考：<https://spring.io/blog/2020/11/10/new-in-spring-5-3-improved-cron-expressions>

#### 4.3.6.脚本处理器

脚本处理器作为执行器运行在公共云桥连接器服务中。

要在 Go 中执行自定义的 Python 代码并获取输入参数和输出结果，包括自定义函数的执行，您可以使用 os/exec 包来调用 Python 解释器并执行相应的脚本或命令，并通过管道（Pipe）来传递输入参数和捕获输出结果。

```
package main

import (
    "fmt"
    "log"
    "os/exec"
)

func main() {
    // Python
    pythonCode := `

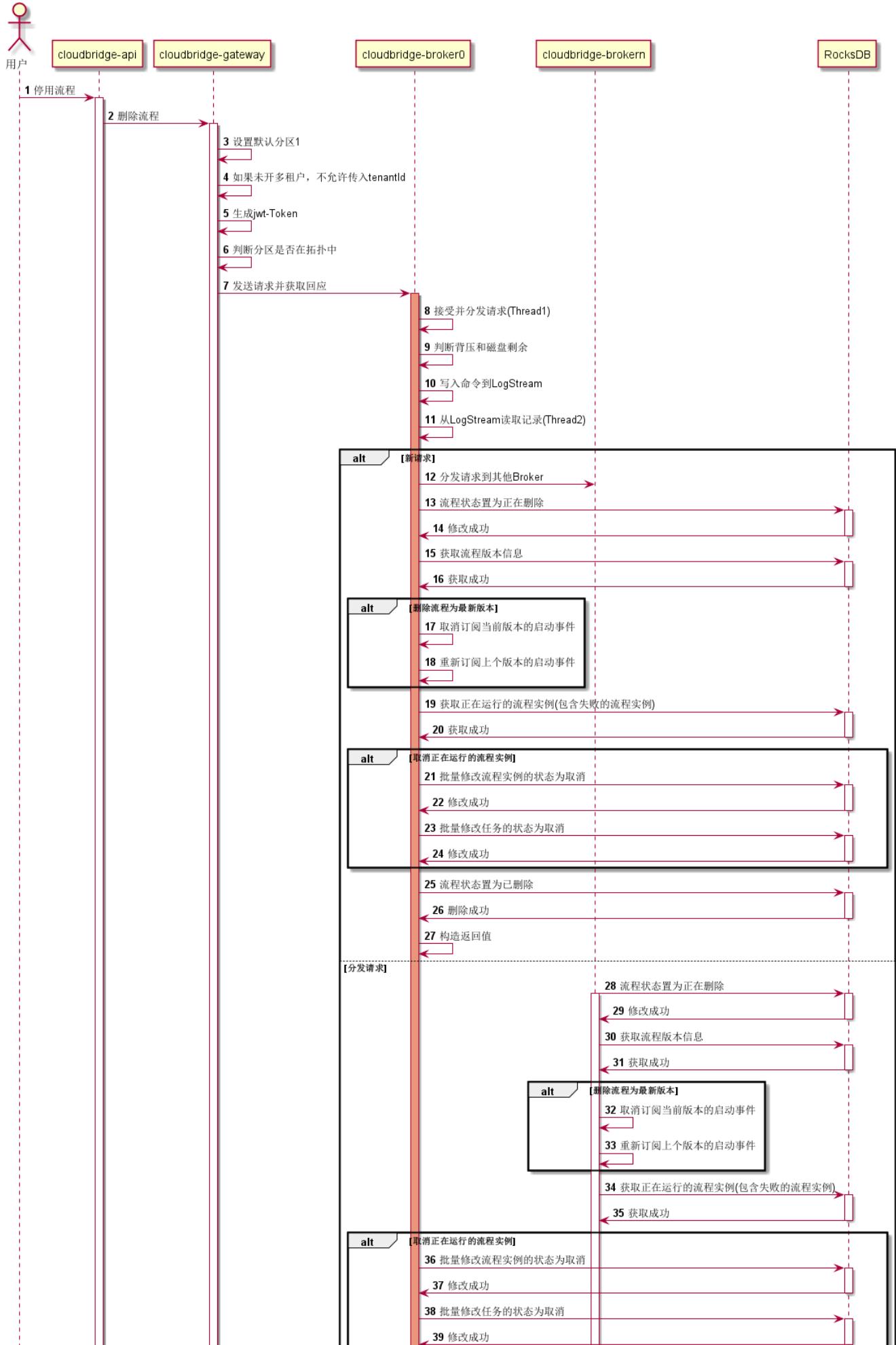
def add_numbers(a, b):
    return a + b

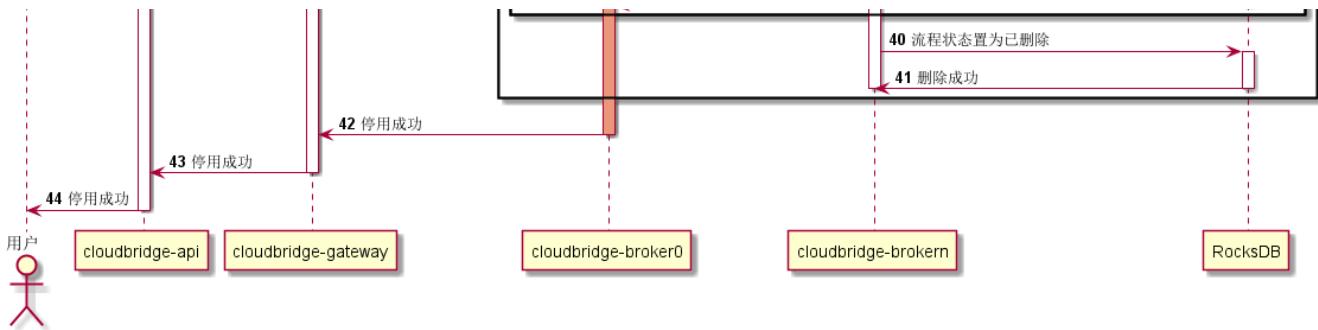
#` + os.Args[1] + " " + os.Args[2]
    cmd := exec.Command("python", "-c", pythonCode)
    var output bytes.Buffer
    cmd.Stdout = &output
    err := cmd.Run()
    if err != nil {
        log.Fatal(err)
    }
    fmt.Println(output.String())
}
```

```
#  
print("The result is:", result)  
  
`  
  
    // Python  
    cmd := exec.Command("python", "-c", pythonCode, "2", "3")  
  
    //  
    outputPipe, err := cmd.StdoutPipe()  
    if err != nil {  
        log.Fatal(err)  
    }  
  
    //  
    err = cmd.Start()  
    if err != nil {  
        log.Fatal(err)  
    }  
  
    //  
    outputBytes, err := ioutil.ReadAll(outputPipe)  
    if err != nil {  
        log.Fatal(err)  
    }  
  
    //  
    err = cmd.Wait()  
    if err != nil {  
        log.Fatal(err)  
    }  
  
    //  
    output := string(outputBytes)  
    fmt.Println(output)  
}
```

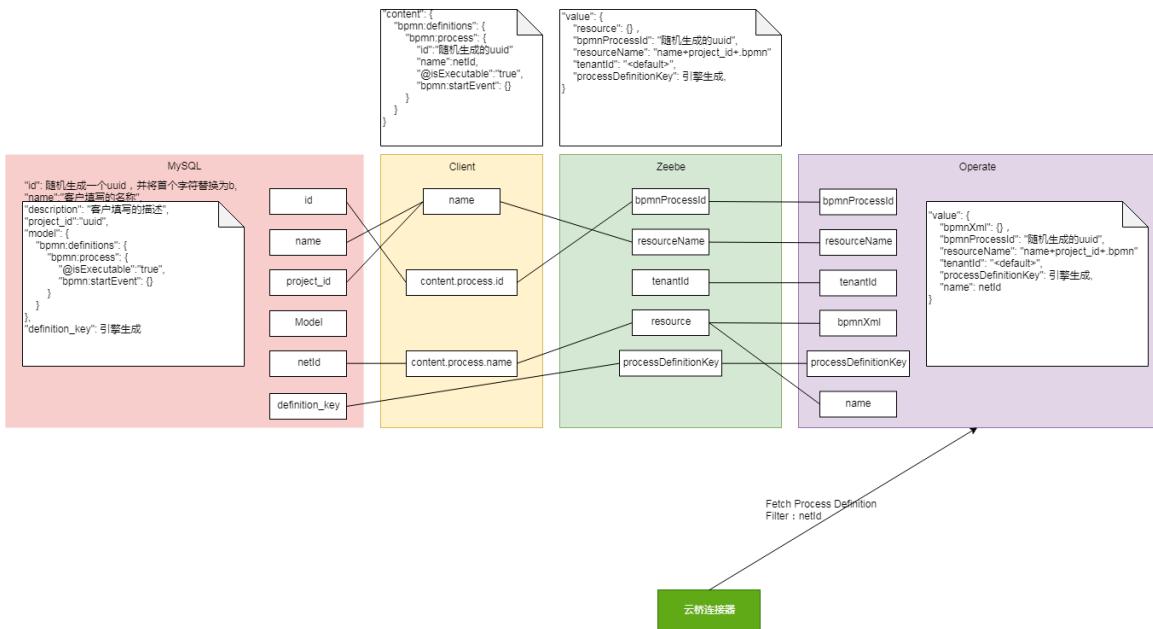
#### 4.3.7.云桥连接器灰度升级

#### 4.3.8.停用流程





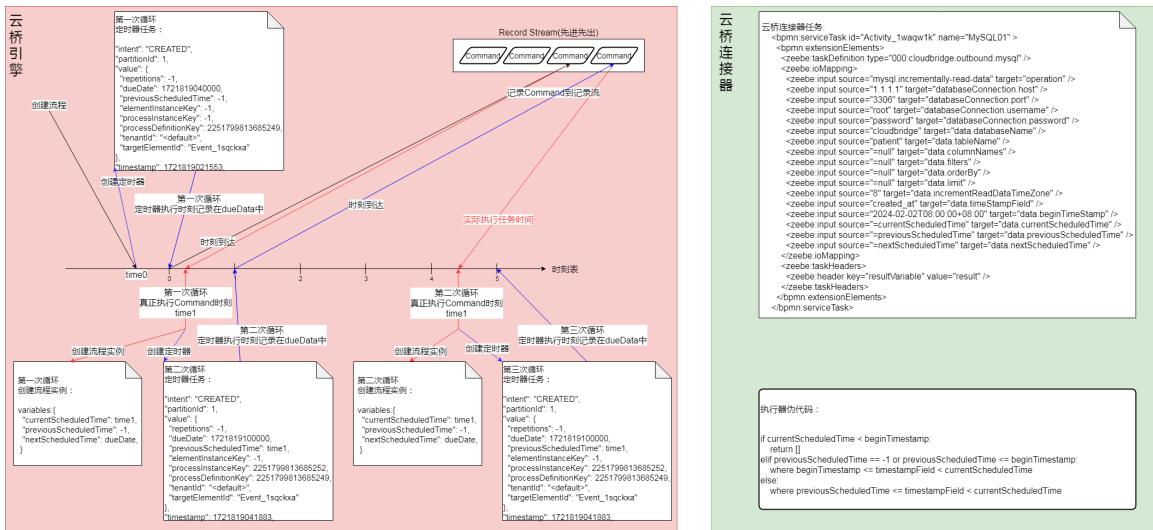
#### 4.3.9.流程字段对应与云桥连接器获取流程



#### 4.3.10.MySQL增量获取记录 实现原理

解决问题：可以针对客户数据库的增量数据进行处理

实现逻辑：“循环触发定时器”可以记录上一次任务执行的时间和当前任务执行的时间，MySQL增量获取记录的执行器可以根据以上时间范围(左闭右开)进行过滤。



## 4.4. 数据结构

描述模块间交互所使用的数据结构，比如数据库，共享内存，文件，等等。统一的消息格式可在此描述，只涉及少数模块的消息格式也可在接口描述时说明。

### 4.4.1. 应用模板关键数据结构

#### MySQL执行器模板

```
{  
    "id": "240bdb36-6e43-471e-b1c7-2618bdbdaff2",  
    "name": "MySQL ",  
    "description": "MySQL ",  
    "categories": [  
        {  
            "id": "uuid",  
            "name": "",  
            "description": ""  
        },  
        {  
            "id": "uuid",  
            "name": "",  
            "description": ""  
        }  
    ],  
    "availability": true,  ()  
    "action_type": "executor", (or)  
    "applies_to": "bpmn:Task",  
    "steps": {  
        "steps_details": [  
            {  
                "key": "app_level_configure",  
                "title": "",  
                "substeps": [  
                    {  
                        "title": "",  
                        "config_items": [  
                            {  
                                "id":  
"app-name",  
  
"label_display": "",  
  
"description": "",  
"type":  
"string",  
  
"required": true,  
  
"default_value": "MySQL_01",  
  
"support_expression": "no"  
                }  
            }  
        ]  
    }  
}
```

```
        "id":  
        "network",  
  
        "label_display": "",  
        "type":  
        "network_radio",  
  
        "choices": [  
  
        {  
  
        "name": "",  
  
        "value": "public_network"  
  
        },  
  
        {  
  
        "name": "",  
  
        "value": "vpc"  
  
        }  
        ]  
    },  
    {  
        "id":  
        "connetcor-type",  
        "type":  
        "hidden",  
  
        "required": true,  
  
        "default_value": "cloud-bridge.outbound.mysql",  
  
        "binding": {  
  
        "type": "cloudbridge:taskDefinition",  
  
        "name": "type"  
        }  
    },  
    {  
        "id":  
        "connetcor-real-type",  
        "type":  
        "hidden",  
  
        "required": true,  
  
        "default_value": "cloud-bridge.outbound.mysql",  
  
        "binding": {  
  
        "type": "cloudbridge:taskDefinition",  
        }
```

```
"name": "realType"
        }
    },
    {
        "id": "type": "hidden",
        "dynamic": true,
        "required": true,
        "default_value": "",
        "binding": {
            "type": "cloudbridge:taskDefinition",
            "name": "siteID"
        }
    },
    {
        "id": "type": "hidden",
        "dynamic": true,
        "required": true,
        "default_value": "",
        "binding": {
            "type": "cloudbridge:taskDefinition",
            "name": "regionID"
        }
    },
    {
        "id": "type": "hidden",
        "dynamic": true,
        "required": true,
        "default_value": "",
        "binding": {
            "type": "cloudbridge:taskDefinition",
            "name": "netID"
        }
    }
}
```

```
"name": "netID"
}
},
{
"hidden",
"dynamic": true,
"required": true,
"default_value": "",
"binding": {
"type": "cloudbridge:taskDefinition",
"name": "azID"
}
},
{
"mysql-host-address",
"label_display": "IP",
"description": "",
"type": "ipv4",
"required": true,
"binding": {
"type": "cloudbridge:input",
"name": "proxyLoc(databaseConnection).host"
},
"support_expression": "no"
},
{
"mysql-port",
"label_display": "",
"description": "",
"port",
"required": true,
"binding": {
```

```
"type": "cloudbridge:input",
"name": "proxyLoc(databaseConnection).port" // proxyLocdocs
},
"support_expression": "no"
},
{
"id":
"mysql-username",
"label_display": "",
"description": "",
"type": "string",
"required": true,
"binding": {
"type": "cloudbridge:input",
"name": "databaseConnection.username"
},
"support_expression": "no"
},
{
"id":
"mysql-password",
"label_display": "",
"description": "",
"type": "password",
"required": true,
"binding": {
"type": "cloudbridge:input",
"name": "databaseConnection.password"
},
"support_expression": "no"
},
{
"id":
"connectivity-detect",
"label_display": "",
"description": "MySQL",
"type": "
```

```
"connectivity_detect_button"
                                }
                            ]
                        }
                    }
                },
                {
                    "key": "app_configure_action",
                    "title": "",
                    "substeps": [
                        {
                            "title": "",
                            "config_items": [
                                {
                                    "id": "actions-choice",
                                    "label_display": "",
                                    "description": "MySQL",
                                    "type": "radio",
                                    "required": "true",
                                    "default_value": "insert",
                                    "choices": [
  {
  "name": "",
  "value": "mysql.insert-data"
  },
  {
  "name": "",
  "value": "mysql.update-data"
  },
  {
  "name": "",
  "value": "mysql.delete-data"
  },
  {
  "name": ""
  }
                                    ]
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}
```

```
"value": "mysql.read-data"
}

        ] ,


"binding": {

"type": "cloudbridge:input",

"name": "operation"
},


"support_expression": "no"
}
]
},


{
    "title": "",
    "config_items": [
        {
            "id":


"action-title-insert",

"label_display": "MySQL-( INSERT )",
"type":


"action_title",


"condition": {


"dependency": {


"id": "actions-choice",


"value": [
    "mysql.insert-data"
]
}
}
}
],


{
    "id":


"action-title-update",

"label_display": "MySQL-( UPDATE )",
"type":


"action_title",


"condition": {


"dependency": {


"id": "actions-choice",


"value": [
]
}
}
}
]
```

```
"mysql.update-data"
]
}

}
}

},
{
"id": "action-title-delete",
"label_display": "MySQL-(DELETE)" ,
"type": "action_title",
"condition": {
"dependency": {
"id": "actions-choice",
"value": [
"mysql.delete-data"
]
}
}
}

},
{
"id": "action-title-query",
"label_display": "MySQL-(QUERY)" ,
"type": "action_title",
"condition": {
"dependency": {
"id": "actions-choice",
"value": [
"mysql.read-data"
]
}
}
}

},
{
"id": "mysql-database-name",

```

```
"label_display": "",  
"description": "",  
"string",  
"required": true,  
"binding": {  
    "type": "cloudbridge:input",  
    "name": "data.databaseName"  
},  
"condition": {  
    "dependency": {  
        "id": "actions-choice",  
        "value": [  
            "mysql.insert-data",  
            "mysql.update-data",  
            "mysql.delete-data",  
            "mysql.read-data"  
        ]  
    }  
},  
"support_expression": "optional",  
"constraints": {  
    "regex": [ ]  
},  
"mysql-table-name",  
"label_display": "",  
"description": "",  
"string",  
"required": true,  
"binding": {  
    "type": "
```

```
"type": "cloudbridge:input",
  "name": "data.tableName"
    } , 

  "condition": {
    "dependency": {
      "id": "actions-choice",
      "value": [
        "mysql.insert-data",
        "mysql.update-data",
        "mysql.delete-data",
        "mysql.read-data"
      ]
    }
  } ,
}

"support_expression": "optional",
  "constraints": {
    "regex": [ ]
  }
} ,
{
  "id": "mysql-update-data",
  "label_display": "",
  "description": "MySQL",
  "type": "type": "json",
  "required": true,
  "binding": {
    "type": "cloudbridge:input",
    "name": "data.updateMap"
  } ,
  "condition": {
    "dependency": {

```

```
"id": "actions-choice",
  "value": [
    "mysql.update-data"
  ]
}

}

"support_expression": "optional",
  "constraints": {
    "regex": []
  }
}

}

{
  "id": "mysql-delete-filters",
  "label_display": "",
  "description": "",
  "type": "type",
  "condition": "condition",
  "required": "true",
  "binding": {
    "type": "cloudbridge:input",
    "name": "data.filters"
  },
  "condition": {
    "dependency": {
      "id": "actions-choice",
      "value": [
        "mysql.delete-data"
      ]
    }
  },
  "support_expression": "required",
  "constraints": {
    "regex": []
  }
},
```

```
        }
    },
    {
        "id": "mysql-update-filters",
        "label_display": "",
        "description": "",
        "condition": "condition",
        "required": true,
        "binding": {
            "type": "cloudbridge:input",
            "name": "data.filters"
        },
        "condition": {
            "dependency": {
                "id": "actions-choice",
                "value": [
                    "mysql.update-data"
                ]
            }
        },
        "support_expression": "optional",
        "constraints": {
            "regex": []
        },
        "id": "mysql-read-filters",
        "label_display": "",
        "description": "",
        "condition": "condition",
        "required": "true",
        "default_value": "=null",
        "type": "type"
    }
}
```



```
"mysql.read-data"

]

}

} ,



"support_expression": "required",

"constraints": {

"regex": [ ]



} ,



"mysql-read-limit",



"label_display": "Limit ",



"description": "null",



"string",



"type": "string",



"required": true,



"default_value": "null",



"binding": {



"type": "cloudbridge:input",



"name": "data.limit"



} ,



"condition": {



dependency": {



"id": "actions-choice",



"value": [



"mysql.read-data"



]



} ,



} ,



"support_expression": "optional",

"constraints": {



"regex": [ ]



} }
```

```
        } ,  
        {  
            "id":  
            "mysql-insert-data-to-insert" ,  
            "lable_display": " " ,  
            "description": " " ,  
            "type":  
            "json" ,  
            "required": true ,  
            "binding": {  
                "type": "cloudbridge:input" ,  
                "name": "data.dataToInsert"  
            } ,  
            "condition": {  
                "dependency": {  
                    "id": "actions-choice" ,  
                    "value": [  
                        "mysql.insert-data"  
                    ]  
                }  
            } ,  
            "support_expression": "required" ,  
            "constraints": {  
                "regex": [ ]  
            } ,  
            {  
                "id":  
                "retries" ,  
                "label_display": " " ,  
                "description": "3" ,  
                "type":  
                "non-negative-integer" ,  
                "required": true ,  
                "default_value": 3 ,  
                "binding": {
```

```
"type": "cloudbridge:taskDefinition",
  "name": "retries"
    },
  "condition": {
    "dependency": {
      "id": "actions-choice",
      "value": [
        "mysql.insert-data",
        "mysql.update-data",
        "mysql.delete-data",
        "mysql.read-data"
      ]
    }
  },
  "support_expression": "no",
  "constraints": {
    "regex": []
  },
  "result-variable",
  "label_display": "",
  "description": "",
  "type": "string",
  "required": false,
  "binding": {
    "type": "cloudbridge:taskHeader",
    "name": "resultVariable"
  },
  "condition": {
    "dependency": {
```

```
"id": "actions-choice",
  "value": [
    "mysql.insert-data",
    "mysql.update-data",
    "mysql.delete-data",
    "mysql.read-data"
  ],
}

},
  },
  "support_expression": "no",
  "constraints": {
    "regex": []
  },
  "hidden-mysql-read-column-names",
  "hidden",
  "required": true,
  "default_value": "=null",
  "binding": {
    "type": "cloudbridge:input",
    "name": "data.columnNames"
  },
  "condition": {
    "dependency": {
      "id": "actions-choice",
      "value": [
        "mysql.read-data"
      ]
    }
  },
  "constraints": {
```

```
"regex": [ ] } } , { "id": "hidden-mysql orderby", "type": "hidden", "required": true, "default_value": "=null", "binding": { "type": "cloudbridge:input", "name": "data.orderBy" } , "condition": { "dependency": { "id": "actions-choice", "value": [ "mysql.update-data", "mysql.delete-data" ] } } , "constraints": { "regex": [ ] } , { "id": "hidden-mysql-limit", "type": "hidden", "required": true, "default_value": "null", "binding": { "type": "cloudbridge:input",
```

```
"name": "data.limit"
        },
        "condition": {
            "dependency": {
                "id": "actions-choice",
                "value": [
                    "mysql.update-data",
                    "mysql.delete-data"
                ]
            }
        },
        "constraints": {
            "regex": []
        }
    },
    {
        "id": "hidden-result-expression",
        "type": "hidden",
        "required": false,
        "binding": {
            "type": "cloudbridge:taskHeader",
            "name": "resultExpression"
        },
        "condition": {
            "dependency": {
                "id": "actions-choice",
                "value": [
                    "mysql.insert-data",
                    "mysql.update-data",
                    "mysql.delete-data",
                    "mysql.read-data"
                ]
            }
        }
    }
]
```

```
        }

    "constraints": {

        "regex": [ ]



    "hidden-error-expression",
    "hidden",
    "required": false,
    "binding": {

        "type": "cloudbridge:taskHeader",
        "name": "errorExpression"
    },
    "condition": {
        "dependency": {
            "id": "actions-choice",
            "value": [
                "mysql.insert-data",
                "mysql.update-data",
                "mysql.delete-data",
                "mysql.read-data"
            ]
        }
    }
},

"constraints": {

    "regex": [ ]
}
]
```

## 4.4.2. 流程关键数据结构

示例:

```
{  
    'bpmn:serviceTask': {  
        '@id': 'Activity_177rx0g',  
        'bpmn:extensionElements': {  
            'cloudbridge:ioMapping': {  
                'cloudbridge:input': [  
                    {  
                        '@source': 'host',  
                        '@target': 'xxx'  
                    },  
                ]  
            },  
            'cloudbridge:taskDefinition': {  
                '@type': 'cloud.bridge:connector-mysql',  
                '@site_id': 'xxx',  
                '@region_id': 'xxx',  
                '@az_id': 'xxx',  
                '@net_id': 'xxx',  
            }  
        }  
    }  
}
```

## 4.5 模块设计

### 4.5.1 云桥管理服务模块

[管理服务设计文档](#)

#### 4.5.1.1. 模块系统需求

#### 4.5.1.2. 模块设计约束

#### 4.5.1.3. 接口列表

### 4.5.2 云桥连接器服务模块

[连接器服务设计文档](#)

#### 4.5.2.1. 模块系统需求

#### 4.5.2.2. 模块设计约束

#### 4.5.2.3. 接口列表

## 4.5.3.应用模板模块

### 连接器模板

#### 4.5.3.1.模块系统需求

#### 4.5.3.2.模块设计约束

#### 4.5.3.3.接口列表

## 4.5.4.监控&告警

#### 4.5.4.1.模块系统需求

#### 4.5.4.2.模块设计约束

#### 4.5.4.3.接口列表

## 4.5.5.升级

#### 4.5.5.1.模块系统需求

#### 4.5.5.2.模块设计约束

#### 4.5.5.3.接口列表

## 4.6 风险分析

基于模块的设计目标，分析哪些点可能存在风险。基于模块的运行场景，分析各个环节可能遇到的问题，面临的风险。

风险分析可采用头脑风暴方法搜集或使用思维导图分析。风险确定后，需要对风险进行排序，确定风险最大，最需要先解决的问题。分析完成后，按风险从大到小排列填写下表。

| 序号 | 风险标题                      | 风险说明                                                                                                                                                                                                                       | 下一步思路                                         |
|----|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| 1  | 云桥引擎历史数据导出到 Elasticsearch | <p>1 ) 需求背景：云桥引擎的流程数据会导出到elasticsearch，为云桥管理服务提供监控的原始数据，这将非常有利于排障与运维。但是生产环境的elasticsearch的性能压力很大，无法承载大量的流程数据导入</p> <p>2 ) 如何处理：由于导出的流程数据均为客户的业务数据，没有必要和SCC的Elasticsearch公用一个，新增一个CloudBridge-Elasticsearch，云桥单独使用</p>      |                                               |
| 2  | 云桥引擎重新部署或升级               | <p>1 ) 背景：由于云桥引擎跑的是客户的业务，所以在任何时刻可能都会有业务在跑，当云桥引擎被重新部署或升级时，可能会导致客户的业务中断。</p> <p>2 ) 如何解决：第一：通过Stateful(有状态)的方式部署云桥引擎，将云桥引擎的流程数据落盘处理，当再次拉起云桥引擎时，可以从磁盘重载流程实例数据，重新执行未完成的任务。第二：设置云桥引擎的helm为static(稳定的)，避免每次SCC升级的时候云桥引擎也跟着升级</p> |                                               |
| 3  | 延迟和周期                     | <p>1 ) 需求：可以通过云桥引擎处理同步任务，并且要求不能有太高的延迟</p>                                                                                                                                                                                  | 不建议使用云桥引擎处理同步的任务。在云桥引擎并发处理的任务数很多时，不能保证同步任务的完成 |

|   |               |                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                       |
|---|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   |               | <p>2 ) 背景 : 一般使用云桥引擎处理相对复杂的异步任务</p> <p>3 ) 影响 : 流程执行的周期时间在很大程度上取决于在这些流程中执行的任务数量 , 工作流引擎本身的操作是可以测量的。在使用 Camunda 8 进行的实验中 , 测量到每个流程节点的处理时间约为 10 毫秒 , 连接器服务任务处理的延迟约为 50 毫秒。因此 , 执行 4 个连接器服务任务会导致 240 毫秒的工作流引擎开销。</p>                                                             | 时间。针对 openapi 触发这种可以设置为同步的流程 , 设置 1min 等待期 , 如果 1 分钟内未返回执行结果 , 报错“超时” , 实际上流程实例还在异步执行的过程中 , 可以在运行日志中查看详情                                                                                                                              |
| 4 | 有效负载          | <p>1) 背景 : 每个流程实例都可以保存一个有效负载 ( 称为流程变量 ) 。所有正在运行的流程实例的有效负载必须由运行时工作流引擎管理 , 并且正在运行和结束的流程实例的所有数据也会导出到 Elasticsearch 。</p> <p>2 ) 要求 : 附加到流程实例的数据 ( 流程变量 ) 影响资源需求。例如 , 如果仅向流程实例添加一两个字符串 ( 需要大约 1 KB 的空间 ) 或包含 1 MB 的完整 JSON 文档 , 则会产生很大的差异。因此 , 在考虑云桥引擎的集群配置时 , 有效负载大小是一个重要因素。</p> | <p>关于有效负载大小有一些一般规则 :</p> <p>每个流程实例的最大变量大小受到限制 , 目前约为 3 MB 。</p> <p>云桥引擎安装的每个分区通常总共可以处理高达 1 GB 的有效负载。较大的有效负载可能会导致处理速度变慢。例如 , 如果运行 100 万个流程实例 , 每个流程实例包含 4 KB 数据 , 则最终会产生 3.9 GB 数据 , 并且至少运行四个分区。</p>                                   |
| 5 | 云桥连接器执行器的超时机制 | <p>1 ) 背景 : 如果云桥连接器执行器在配置的超时时间内未完成或失败 ( 设置了重试次数的话 ) , 云桥引擎会将作业重新分配给另一个云桥连接器。导致两个不同的云桥连接器同时处理过同一工作。</p> <p>2 ) 如何解决 : 第一 : 云桥引擎会保证只会有一个云桥连接器可以完成该作业 , 另一个云桥连接器的 complete 命令会被拒绝。第二 : 要求云桥连接器执行器的实现必须是幂等的</p>                                                                   | <p>云桥连接器返回执行结果设置重试机制 , 重试 3 次。</p> <p>超时时间设置 5min , 在 5min 时云桥连接器未返回执行结果 ( 场景比较极端 ) , 该任务可以被云桥连接器再次激活执行。</p> <p>camunda 之所以这么设计 , 是为了提高容错性 , 即使一个云桥连接器 Worker 发生故障或无法完成作业 , 工作流可以继续进行</p> <p>后续此处修改为 : 未在 5 分钟内返回执行结果 , 统一认为任务失败。</p> |

## 5. 关键特性设计

### 5.1. 业务出错处理

用一览表的方式说明每种可能的出错情况出现时 , 系统输出信息的形式、含意及处理方法。

| 系统流程 | 出错步骤              | 出错可能原因               | 错误处理方式             | 影响 ( 网络中断、网络抖动、性能下降、配置丢失 ) |
|------|-------------------|----------------------|--------------------|----------------------------|
| 通用问题 | VPC 内云桥连接器的任务未被执行 | VPC 内未部署云桥连接器        | 需要 SRE 手动部署        |                            |
|      |                   | VPC 到 SCC 网络不通       | SRE 根据研发操作手册进行网络打通 |                            |
|      | 流程长时间阻塞           | 云桥连接器未准备完成           | 检测连接器组件是否正常        |                            |
| 客户问题 | openapi 触发的流程报错超时 | 流程过长 , 在 1min 内未执行完毕 | 客户自行优化流程 , 去除无关的逻辑 |                            |
|      |                   |                      | 增大云桥引擎的背压数量 , 云    |                            |

|  |               |                 |                                         |  |
|--|---------------|-----------------|-----------------------------------------|--|
|  | 云桥连接器启动流程实例失败 | 触发云桥引擎背压机制      | 桥连接器有等待重试的机制                            |  |
|  | 流程执行失败        | 有可能是执行器连接客户应用失败 | 客户可以设置每一个任务节点的重试次数<br>客户可以在失败的任务节点处重新执行 |  |

## 5.2. 避免业务长时间中断处理

一些可能导致用户的业务长时间中断的问题或者隐患。是“系统出错处理设计”的一个子集。

| 业务中断错误编号 | 受影响客户业务         | 可能导致的异常及其原因 | 恢复和应对措施                                    | 可测试性保证      |
|----------|-----------------|-------------|--------------------------------------------|-------------|
| 1        | 客户自身的某个业务服务异常退出 | OOM或自身BUG等  | 当某一个任务节点请求客户的某一应用服务失败时，可以加入重试机制防止流程实例大面积失败 | 构造某一个应用服务挂掉 |

## 5.3. 硬件故障的可靠性保障措施

描述怎么检查系统中出现的硬件故障，以及出现故障时的业务保障措施。

## 5.4. 补救措施

说明故障出现后可能采取的变通措施，包括：

- a. 后备技术：说明准备采用的后备技术，当原始系统数据万一丢失时启用的副本的建立和启动的技术，例如周期性地把磁盘信息记录到磁带上去就是对于磁盘媒体的一种后备技术；
- b. 降效技术：说明准备采用的后备技术，使用另一个效率稍低的系统或方法来求得所需结果的某些部分，例如一个自动系统的降效技术可以是手工操作和数据的人工记录；
- c. 恢复及再启动技术：说明将使用的恢复再启动技术，使软件从故障点恢复执行或使软件从头开始重新运行的方法。

| 错误编号                     | 措施类型    | 补救措施                 |
|--------------------------|---------|----------------------|
| 客户的流程在某一个任务节点失败，导致整个流程失败 | 恢复再启动技术 | 给客户提供从失败的任务节点恢复执行的功能 |

## 5.5. 维护设计

说明为了系统维护的方便而在程序内部设计中作出的安排，包括在程序中专门安排用于系统的检查与维护的检测点和专用模块。

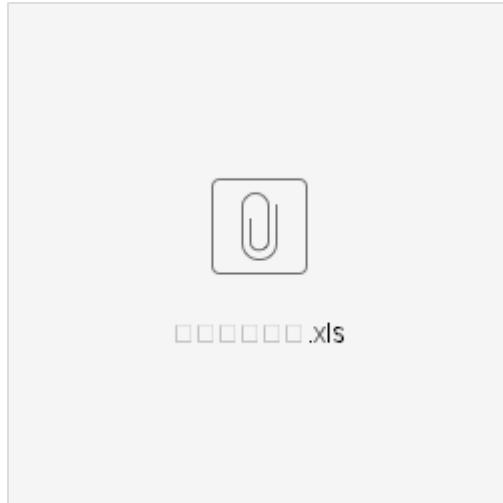
## 5.6. 安全性设计

说明系统需求规格说明书中描述的安全性需求如何实现。如果是由某个模块独立实现，仅需说明具体由哪个模块实现，这个需求应该明确纳入该模块的设计目标书。

### 5.6.1. 威胁建模分析

威胁建模分析，目的是分析该版本总体设计中涉及到的各个模块实体、数据存储、进程服务、开放端口、交互协议等是否存在相关的安全威胁风险，把风险识别出来并填入如下表格。

具体威胁建模分析方法，采用微软的STRIP方法来分析，具体方法请参考如下地址，看不懂的可以找产品线安全接口人咨询，或发邮件给RDSec提供咨询培训。



| 元素           | 威胁分类    | 威胁概述                                                                                                                                     | 威胁场景名称    | 威胁评级 | 威胁分析                                                                                                                                     | 消减方案                                                                                                                                                    | 当前状态 | 测试方法(攻击方法)                                               |
|--------------|---------|------------------------------------------------------------------------------------------------------------------------------------------|-----------|------|------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|------|----------------------------------------------------------|
| 连接器服务/流程引擎   | 篡改T，欺骗S | 攻击者可能会在连接器和流程引擎之间插入自身，使得连接器和流程引擎双方错以为正在直接通信，但是攻击者监控和篡改了双方的所有数据                                                                           | 中间人攻击     | 中    | <p>缺乏加密通信，如果通信双方没有采用加密通信协议，通信内容就会以明文形式传输，容易被攻击者窃取和篡改；</p> <p>如果通信双方没有进行有效的身份验证，攻击者可以冒充其中一方的身份，使得双方认为他们正在直接通信，但实际上所有的通信都经过了攻击者的监控和篡改；</p> | <p>1. 使用加密通信协议：通信双方应该使用TLS1.3 加密通信协议，确保通信内容经过加密</p> <p>2. 沿用托管云中的身份验证机制：通信双方应该使用AKSK认证等身份验证机制</p> <p>3. 签名中有防重放的机制</p>                                  | 已处理  |                                                          |
| 连接器服务/集成管理服务 | 篡改T     | 在创建流程时，攻击者通过在模型参数中输入字符串时注入恶意的SQL指令攻击。在设计不良的程序当中忽略了检查，那么这些注入进去的指令就会被数据库误认为是正常的SQL指令而执行，因此遭到数据泄漏或者数据的破坏。其主要原因是程序没有细致地过滤用户输入的数据，致使非法数据侵入系统。 | SQL注入     | 高    | <p>用户输入的数据未进行参数校验导致产生SQL注入；</p> <p>用户输入验证的不充分，如果应用程序没有对用户输入进行充分的验证和过滤，攻击者更容易成功注入恶意的SQL代码。</p>                                            | <p>1. 输入验证和过滤：应用程序应该对用户输入进行充分的验证和过滤，确保输入的数据符合预期的格式和范围</p> <p>2. 参数化查询，不直接将用户的输入作为SQL代码的一部分</p> <p>3. 最小权限原则，减少被攻击的潜在风险</p>                              | 已处理  | 通过postman测试请求，构造请求参数包含 or '1'='1 sql注入数据，判断是否检测出这个注入问题存在 |
| 连接器服务密码管理    | 信息泄露I   | 租户的密码信息存储租户VPC中的sqlite数据库中，攻击者通过其他的攻击手段进入租户的sqlite中，其中存储的用户密码信息将会泄露，攻击者可以使用这些用户密码信息肆意非法访问租户的其他资源或服务                                      | 数据泄露、密码泄露 | 中    | <p>1. 密码以明文或弱加密形式存储在数据库中，使得攻击者可以轻易获取</p> <p>2. 数据库中存储了过多和密码直接绑定的详细信息，映射关系清晰</p>                                                          | <p>1. 密码存储使用密文存储库，SCC加密，租户的连接器服务解密并使用，即使被攻破，拿到的也是密码的密文</p> <p>2. 只存密码密文和映射key，不存储别的用户信息，不指明这个密码是什么服务或中间件的密码，不提供中间件或服务的访问地址，即使密文被解密，得到的也只是一段没有意义的字符串</p> | 已处理  | 无                                                        |
| 连接器服务/流程引擎   | 欺骗S     | 攻击者模拟连接器满足触发条件                                                                                                                           | 越权操作      | 中    | 1. 系统中缺乏有效身份校验机                                                                                                                          | 1. 资源ID必须是复杂、不可预测                                                                                                                                       | 已处理  | 构造gRPC或HTTP请求，访问                                         |

|                  |       |                                                                                                                       |           |   |                                                              |                                                                                                                                  |     |                                                            |
|------------------|-------|-----------------------------------------------------------------------------------------------------------------------|-----------|---|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|-----|------------------------------------------------------------|
|                  |       | 时创建流程实例，根据流程ID创建别人的流程实例；连接器向流程引擎汇报任务的时候根据jobkey,随意的汇报其他用户的job完成状态；<br>也即：攻击者可以根据某个资源ID，在没有进行有效身份校验的系统中，未经授权地访问其他用户的资源 |           |   | 制，使得攻击者可以绕过访问控制，直接使用资源ID进行访问<br>2.攻击者可以通过猜测或枚举的方式尝试访问其他用户的资源 | 的，增加猜测和枚举的难度<br>2.强制的身份验证方式(AKSK)结合验签算法<br>openapi.auth:(aksk用于验证用户的身份和授予资源访问权限；验签用于验证数据的完整性)<br>3.互信验证<br>4、归属属性校验(防止操作其他租户的资源) |     | 其他租户的流程，如启动其他租户的流程，创建流程实例；结果是到达apisix就会被拦截和报错，无法做到实际操作非法资源 |
| 连接器服务/流程引擎       | 流量攻击  | 攻击者劫持租户的VPC，无限制的访问SCC中心端，导致中心端不可用                                                                                     | 流量攻击DDOS  | 中 | 1.系统没有对连接器的访问做限制，访问控制限制不当                                    | 1.限制连接器在一段时间内(1min)的最大访问次数，超过次数则限制访问<br>2、平台有限流的机制                                                                               | 已处理 | 通过Jmeter并发请求，超过限流设置的请求会被服务端拒绝，状态码:429                      |
| AK/SK泄露          | 信息泄露I | 获取云服务台地址，窃取AK/SK                                                                                                      | 窃取API密钥   | 高 | 1、窃取API密钥                                                    | 1. SK加密存储<br>2、AKSK可以访问的API我们会权限校验进行约束，无法访问没有权限的API<br>3、AKSK有白名单机制，会设置信任的客户端IP，即使泄露也不能在任意网络位置使用<br>4、平台可以快速切断任意AKSK的访问能力        | 已处理 | 1、在不信任的网路位置访问<br>2、访问没有权限的API                              |
| 接口调用             | 篡改T   | 接口调用参数防注入                                                                                                             | SQL注入     | 高 |                                                              | 针对参数做严格类型校验                                                                                                                      | 已处理 | 页面创建流程的时候使用特殊字符参数校验会报错                                     |
| 创建流程：输入密码        | 信息泄露I | 日志中泄漏敏感信息                                                                                                             | 日志中泄漏敏感信息 | 中 | 日志中会输出用户密码，不安全                                               | 日志对密码进行星号*代替                                                                                                                     | 已处理 | 创建流程的时候，选择需要输入密码的应用，然后查看日志                                 |
| 租户调用其他租户的OpenAPI | 提权E   | 租户A请求租户B的URL                                                                                                          | 水平越权      | 高 | 租户可能会调用其他租户的OpenAPI                                          | 进行URL的归属性校验                                                                                                                      | 已处理 | 租户A去调用另一个租户的OpenAPI                                        |

## 5.6.2. 安全设计

根据上一节威胁建模识别出来的风险和建议采取的安全措施，设计相关的安全机制，包括整体的安全架构设计和各个模块需要设计的安全机制，在此章节进行详细描述，比如通信协议要加密，就需要设计相关的加密方案。

### 5.6.1.1. 整体安全架构设计

### 5.6.3. 预使用组件版本合规性情况

使用黑鸭扫描器对预计使用的组件版本进行合规性扫描（<https://200.200.0.207/> 账号权限以及使用方式，请与张笑尘联系），并将扫描结果上传至本处，原则上不允许使用提示告警的组件，如果需要使用，需经过法务胡海斌审批通过，并将分析结果上传至该处，降低后续关联修复的巨额成本。（黑鸭扫描报告同时含有合规性、漏洞情况）

OpenHub给出Zeebe的漏洞报告--无报告的漏洞

## Project Security



### 5.6.4. 预使用组件版本漏洞情况

使用黑鸭扫描器对预计使用的组件版本进行安全性扫描（<https://200.200.0.207/> 账号权限以及使用方式，请与张笑尘联系），并将扫描结果上传至本处，原则上不允许使用含有漏洞的组件版本，如果需要使用，需经过安全经理、主管审批通过，并将分析结果上传至该处，降低后续关联修复的巨额成本。（黑鸭扫描报告同时含有合规性、漏洞情况）

## 5.7. 可靠性设计

可靠性目标定义：

- 1) 可用度的目标定义：比如产品可用度达到5个9/年中断时间少于5分钟
- 2) 可靠性的目标定义：比如故障检测&定位率达到70%，故障自动恢复率达到85%，故障人工恢复时长小于30分钟等等

可靠性系统设计：

进行系统级的FMEA分析，给出系统在冗余设计，故障管理（检测、定位、上报、恢复等）等方面的设计机制，给出系统在防人因差错方面的设计机制；给出系统在故障预测预防方面的实现机制（主要是硬件方面，比如针对硬盘的故障预测，针对内存的故障预测等等）；给出升级不中断业务方面的实现原理（包含升级回退机制）

在SCC升级时，为了保证流程实例可以继续执行&数据不丢失，云桥引擎并不会重新部署，此时流程实例继续保持活动状态。公共云桥连接器采用滚动升级的方式，保证至少有云桥连接器可以执行业务。

## 5.8. 韧性设计

韧性目标：定义系统业务过载控制目标：比如在2倍业务情况下，成功率达到98%；在3倍业务情况下达到95%；在10倍业务情况下，成功率达到93%；产品在被拒绝服务攻击时，处理性能虽然降低，但仍然可以开展业务服务；

韧性设计：给出系统在业务过载保护方面的设计机制，包含但不限于各类业务的优先级设计，各种业务优先的控制实现机制；此外需要确保产品在遇到消耗大量性能攻击时，需要有对应的安全机制，例如：在Apache/Nginx需要设置拒绝服务的超时机制，并且对于同一时间发起大量无效请求的IP进行一段时间的拒绝响应，直至产品能够完全正常响应。

## 5.9. 性能设计

将来会有大量的异步流程下沉到云桥引擎中处理，所以需要评估一下云桥引擎在高负载情况下的性能和稳定性。其作用是模拟系统在实际使用中可能遇到的高负载情况，以便发现系统在负载高峰期间可能出现的问题和瓶颈，并进行优化和改进。

云桥连接器作为执行任务的进程，需要拥有并发处理的功能，需要评估一下在极限场景下，云桥连接器的处理性能

### 5.9.1. 测试方案

分析开发的技术方案，对方案的缺点进行深入分析，主要分析维度是需求满足度、客户场景满足度、性能、稳定性、可靠性维度，

要求：同时要启动多个流程实例，单个流程实例的流程按照50节点、4KB的流程负载来定义。在压力测试过程中，要监测云桥各个组件、elasticsearch的负载情况。

如何实现：

①测试10000流程下云桥服务的静息状态。

②测试10000流程下，并发运行1000流程实例云桥引擎是否达到极限状态。测试每天运行1,000,000流程实例云桥引擎的稳定性。

## 5.10. 可监控性设计

描述模块对监控平台适配相关的设计，是否支持配置监控，如果没有，此处写不涉及即可。

云桥引擎中运行的流程实例大部分为异步进行，提供API查看所有正在运行或已经结束的流程实例，监控流程运行的情况。后续可以接入Prometheus和Grafana来监测云桥引擎集群拓扑的健康度、吞吐量指标、已处理的请求、每秒导出的事件、磁盘和内存使用情况等。

## 5.11. 可调试可测试设计

分析系统实际运行时，容易出现问题、定位问题比较困难。或者测试验证比较困难的逻辑。针对这些逻辑，需要在机制层面考虑怎么快速定位、解决问题，降低调试、测试的难度，同时不对系统运行施加过多负面影响。

| 可测试性    | 说明                                                                                                                                                                  |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 后台CLI命令 | 提供zbctl，方便测试/开发人员进行测试。                                                                                                                                              |
| 接口性能测试  | 所有Python模块的关键接口及函数均使用以下装饰器，开启DEBUG模式下将统计接口耗时：<br><pre>from oslo_utils.timeutils import time_it  @time_it(logger=LOG, enabled=LOG.isEnabledFor(logging.DEBUG))</pre> |
| 性能压测    | 可以通过手动部署流程、并发启动流程实例的方式对云桥引擎进行压力测试。                                                                                                                                  |
| 单元测试    | 新增的Python代码使用单元测试覆盖                                                                                                                                                 |
| API测试   | Python模块使用自动化接口覆盖完成                                                                                                                                                 |
| 业务、场景测试 | 覆盖端到端的自动化测试                                                                                                                                                         |
| 流程实例排障  | 通过统一的API监控和操作运行的流程实例                                                                                                                                                |

此外，由于所有同步或定时任务涉及第三方系统，故可直接通过mock方式构造仿真数据，即通过端到端自动化的方式覆盖所有测试用例，防止改动引发。

针对此设计中关联的BPMN模型，可以通过自动化方式启动流程实例，进行端到端检查，防止改动引发。

## 5.12. 系统隐私设计

隐私目标：定义系统不会出现因自身安全而造成敏感信息泄露的目标：系统在收集用户信息时，不会出现私自收集用户不清楚及用户不同意收集的数据；

隐私设计：给出系统在业务传输过程保护方面的设计机制，包含但不限于各类业务的传输敏感信息时，需要使用安全协议传输，以及对内容进行加密传输；此外需要确保系统在收集信息时，应明确为什么需要收集该信息，还应尝试预判未来是否会使用这些信息用于其他事物，并告知用户是否有这样的计划，提供一份易于阅读的条款或条件的摘要。

## 5.13. 可重用设计

### 5.13.1. 技术规范落地计划

根据产品线的技术规范基线计划，定义本版本需要遵守、实现、新增、修订的技术规范。具体参考流程说明：<http://docs.sangfor.org/pages/viewpage.action?pageId=97130963>

公司已发布的技术规范列表请参考流程中的相关说明。

请填写附件《技术规范基线计划》中的表格，结果以附件的形式保存在本总设文档。此计划需通过技术规范委员会评审，具体要求，请参考上述流程中的相关说明。

### 5.13.2. 本版本可采用的公用代码

通过识别(和预研部门、RDM一起识别)，列出哪些功能点或模块可以使用公用代码。

### 5.13.3. 本版本可产出的公用代码

要考虑产出公用代码，以方便其它产品或版本来重用。这样可以持续提高公司的代码重用率，进而降低研发成本，提高效率。

### 5.13.4. 系统依赖解耦设计

分析系统(或系统内组件)移植到其它软硬件环境，需要考虑的依赖因素。描述降低系统(或系统内组件)对软硬件环境依赖性的设计，即怎么解耦，屏蔽系统(或系统内组件)对底层软硬件环境的依赖。保证系统(或系统内组件)移植到其它软硬件环境时，已有代码逻辑无需修改，或仅需少量修改。

## 6. 总结

### 6.1. 与上一个版本的设计变化

描述本版本与上一个版本之间发生变化的总体设计(该设计涉及到多个模块交互或修改后影响多个模块)。如果这个版本总体设计没有变化，或者是新产品，可不填此表，但需在此明确说明没有变化。

本节可以按章节描述，也可以直接列成一个表格

#### 6.1.1. 设计变化项1

##### 6.1.1.1. 设计变化原因

为什么该设计需要变更？

##### 6.1.1.2. 设计变化描述

描述具体发生变化的内容。

##### 6.1.1.3. 设计变化产生的影响

该设计变化会造成什么影响？

#### 6.1.2. 设计变化项2...

### 6.2. 风险问题保证

风险问题指那些对于达到预定目标没有十足把握的设计问题。

| 风险点                 | 风险预研结果/风险规避措施                    |
|---------------------|----------------------------------|
| 例1：处理速度达100MBPS     | 根据预研文档《xxx》，可以达到要求               |
| 例2：缓存数据耗用内存可能超过800M | 如果超过800M，使用最久未使用淘汰算法将部分数据保存在磁盘上。 |
|                     |                                  |
|                     |                                  |
|                     |                                  |

## 6.3. 尚未解决的问题

说明在总体设计过程中尚未解决而必须在系统完成之前必须解决的各个问题。

如果总体设计阶段没有问题需要遗留到后续阶段完成，本章节可以为空。

## 7. 变更控制

### 7.1. 变更列表

在编码过程中，可能会有一些设计不合理的地方，需要对原设计做一些调整，需要在这里说明变更的章节、内容、原因，并对变更进行评审确认（至少需要经过版本经理/开发经理的确认）。此章节在编码完对设计文档更新后进行评审确认。

需求变更引起的设计变更，在下面加以说明，并对设计变更的结果进行部门内部评审。

| 变更章节 | 变更内容 | 变更原因 | 变更对对老功能、原有设计的影响                              |
|------|------|------|----------------------------------------------|
|      |      |      | 考虑完全这些影响，要和开发经理、架构师、设计人员、老版本的开发人员进行交流，并最后评审。 |
|      |      |      |                                              |
|      |      |      |                                              |
|      |      |      |                                              |
|      |      |      |                                              |