华为云Flomesh服务包使用手册

# 软件概述

Flomesh 是针对云边计算环境设计的服务网格，采用 osm 作为控制平面，采用 pipy 作为数据平面，具有高性能、低资源、简单、易用、易扩展、广泛兼容（支持x86/arm64/龙芯/RISC-V）的特点。

## 软件特点

### 兼容性

Flomesh支持x86、arm64、龙芯、RISC-V等计算架构，具有广泛的兼容性。

### 高性能低资源

Flomesh控制面实现简单，使用简单；数据面 sidecar初始内存10M左右，合适高密度部署;执行文件和容器镜像均低于15M，满足“快速部署、快速回收”的需求。 在 cgroup CPU=1000m的情况下，提供行业内最大的吞吐量TPS，最小的处理延迟，最丰富的指标采集能力。

### 易用性

Flomesh架构简单，sidecar proxy可以独立运行，独立调试。提供GUI控制台，支持多租户。

组件简单，易于使用。提供图形用户界面，方面管理和维护。

### 自主知识产权

Flomesh为全自主、自研开发，拥有完整知识产权；可以开发特定行业“专属版本”。控制面和数据面完全自主可控。

### 扩展性

Flomesh的控制平面充分支持SMI规范，SMI规范提供服务网格的标准定义；数据平面Pipy作为可编程的代理，无论从协议层面还是网格能力的扩展，都具备深度定制能力。

## 软件资源需求

| **键名** | **类型** | **默认值** | **描述** |
| --- | --- | --- | --- |
| osm.injector.resource | object | {"limits":{"cpu":"0.5","memory":"64M"},"requests":{"cpu":"0.3","memory":"64M"}} | Sidecar 注入器的容器资源参数。 |
| osm.osmBootstrap.resource | object | {"limits":{"cpu":"0.5","memory":"128M"},"requests":{"cpu":"0.3","memory":"128M"}} | Flomesh引导程序的容器资源参数。 |
| osm.osmController.resource | object | {"limits":{"cpu":"1.5","memory":"1G"},"requests":{"cpu":"0.5","memory":"128M"}} | Flomesh控制器的容器资源参数。 |
| osm.prometheus.resources | object | {"limits":{"cpu":"1","memory":"2G"},"requests":{"cpu":"0.5","memory":"512M"}} | Prometheus 的容器资源参数。 |

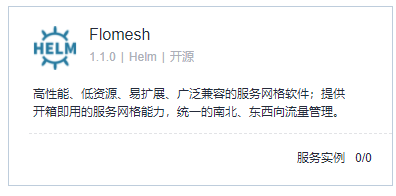
# 软件安装指南

## 安装前准备

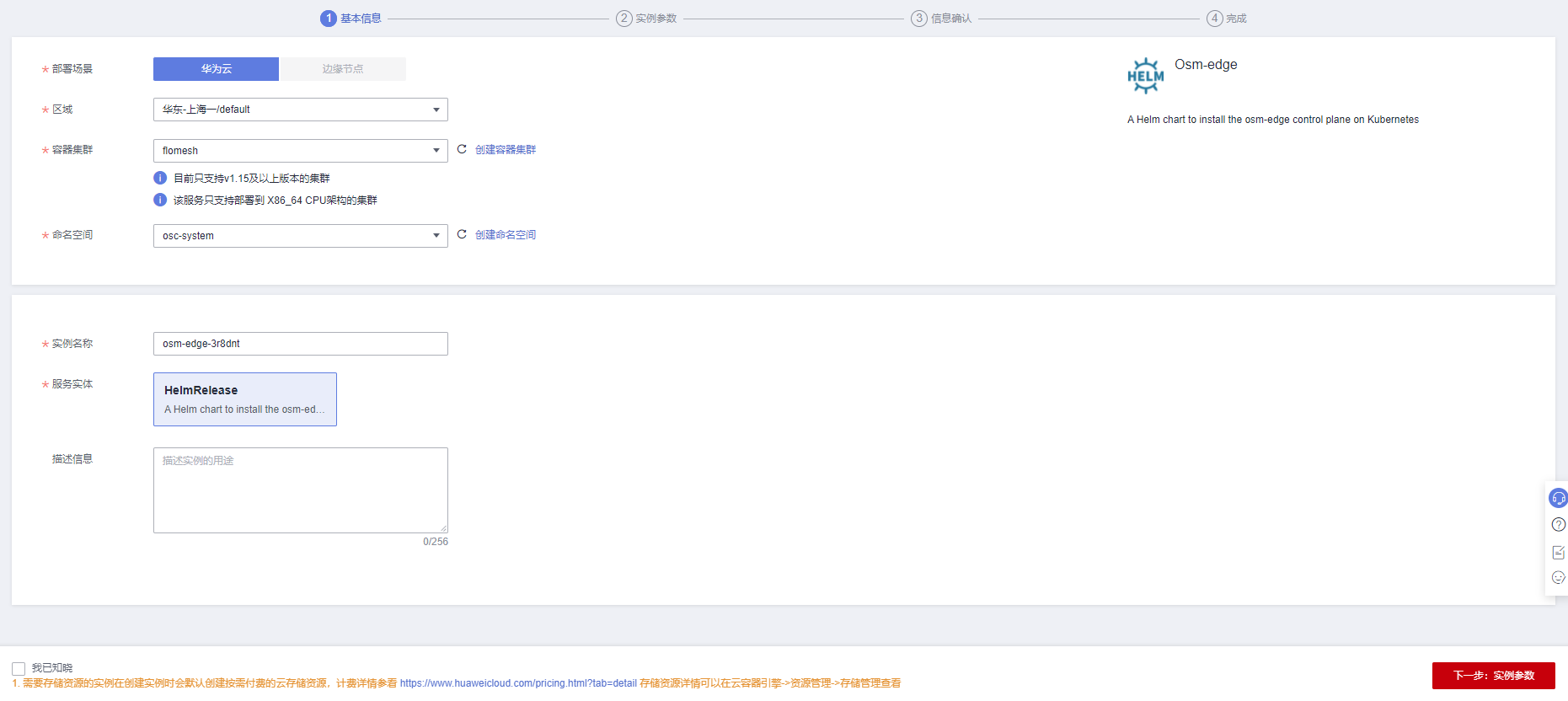
华为云CCE集群具备可用计算节点，要求计算节点能够访问公网，用于拉取Flomesh公网镜像。

## 安装步骤

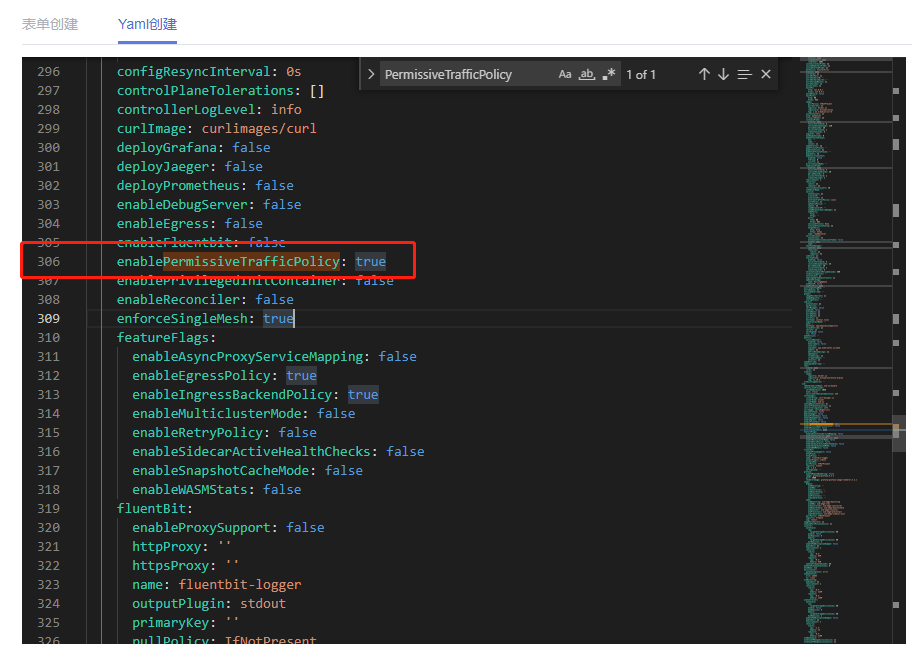
进入云原生服务中心OSC，在“服务目录”页面，选择“Flomesh”，点击“创建实例”。



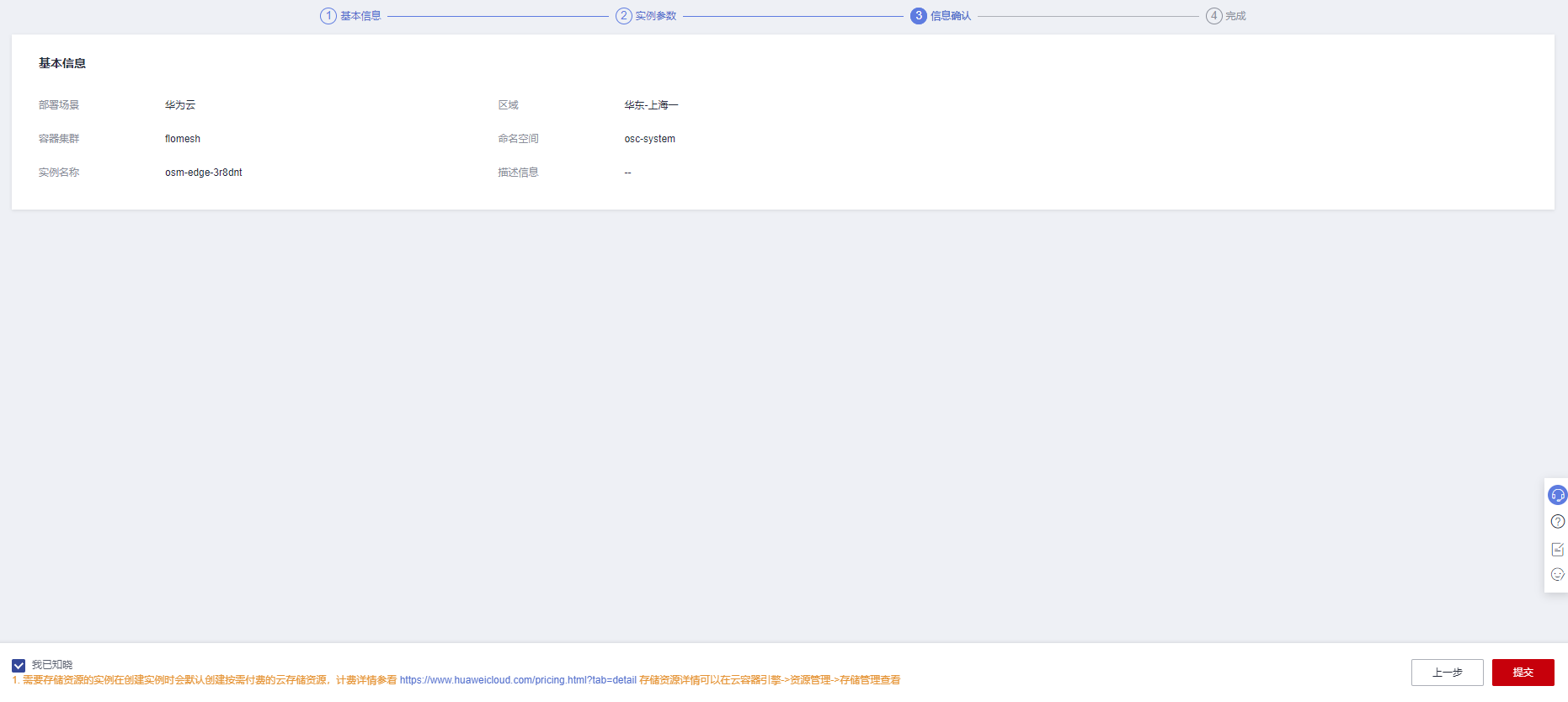
选择对应的区域、集群和命名空间，点击“下一步：实例参数”



修改实例参数，在yaml文件中，修改第306行为：enablePermissiveTrafficPolicy:ture



点击“下一步：信息确认”，点击提交。



实例创建需要等待一段时间。

创建完成后，在“我的实例”栏目下可看到刚刚创建的实例。



# 软件CLI终端准备

调用Flomesh服务网格服务，需要一台能够访问华为云CCE集群的终端，并安装kubectl和Osm命令行工具。kubectl工具安装配置参照华为云CCE集群相关手册。

下载并安装 osm-edge 命令行工具：

system=$(uname -s | tr [:upper:] [:lower:])

arch=$(dpkg --print-architecture)

release=v1.1.0

curl -L https://github.com/flomesh-io/osm-edge/releases/download/${release}/osm-edge-${release}-${system}-${arch}.tar.gz | tar -vxzf -

./${system}-${arch}/osm version

cp ./${system}-${arch}/osm /usr/local/bin/

# 应用迁入

## 应用迁入要求

应用迁入Flomesh前提需要满足如下条件：

*Security Contexts*

*Do not run applications with the user ID (UID) value of 1500. This is reserved for the Pipy proxy sidecar container injected into pods by osm-edge’s sidecar injector.*

*If security context runAsNonRoot is set to true at the pod level, a runAsUser value must be provided either for the pod or for each container.*

*For example:*

|  |
| --- |
| *securityContext:*  *runAsNonRoot: true*  *runAsUser: 1200* |

*If the UID is omitted, application containers may attempt to run as root user by default, causing conflict with the pod’s security context.*

*Additional capabilities are not required.*

*Note: the osm-edge init container is programmed to run as root and add capability NET\_ADMIN as it requires these security contexts to finish scheduling. These values are not changed by application security contexts.*

*Ports*

*Do not use the following ports as they are used by the Pipy sidecar.*

| **Port** | **Description** |
| --- | --- |
| 15000 | Pipy Admin Port |
| 15001 | Pipy Outbound Listener Port |
| 15003 | Pipy Inbound Listener Port |
| 15010 | Pipy Prometheus Inbound Listener Port |

## 迁入命名空间

将 Kubernetes 命名空间迁入 Flomesh：

如果要将命名空间下的所有应用都纳入 Flomesh网格管理，执行 osm namespace add 命令：

osm namespace add <namespace> --mesh-name <mesh-name>

### 注意：移除命名空间

要从 Flomesh服务网格中移除命名空间，可以使用命令 osm namespace remove：

osm namespace remove <namespace>

请注意： osm namespace remove 命令指示告诉 Flomesh停止对命名空间中的 sidecar 代理配置应用更新。并不会移除 sidecar 代理。这意味着可以继续使用现有的代理配置，但是不会收到 Flomesh控制平面的更新。如果想要移除所有 pod 的代理，在使用 CLI 将命名空间从 Flomesh网格移除后重新安装所有 pod 负载。

## 演示应用准备

### 部署演示应用并迁入Flomesh

为了让用户更好的快速上手Flomesh，本手册指导如何部署演示应用并迁入Flomesh服务网格，并完成后续Flomesh相关功能演示。

演示应用包括了如下服务：

* bookbuyer 是一个 HTTP 客户端，它发送请求给 bookstore。这个流量是允许的。
* bookthief 是一个 HTTP 客户端，很像 bookbuyer，也会发送 HTTP 请求给 bookstore。这个流量应该被阻止。
* bookstore 是一个服务器，负责对 HTTP 请求给与响应。同时，该服务器也是一个客户端，发送请求给 bookwarehouse 服务。这个流量是被允许的。
* bookwarehouse 是一个服务器，应该只对 bookstore 做出响应。bookbuyer 和 bookthief 都应该被其阻止。
* mysql 是一个 MySQL 数据库，只有 bookwarehouse 可以访问。

使用如下命令部署这些服务：

kubectl create namespace bookstore

kubectl create namespace bookbuyer

kubectl create namespace bookthief

kubectl create namespace bookwarehouse

osm namespace add bookstore bookbuyer bookthief bookwarehouse

kubectl apply -f https://raw.githubusercontent.com/flomesh-io/osm-edge-docs/main/manifests/apps/bookbuyer.yaml

kubectl apply -f https://raw.githubusercontent.com/flomesh-io/osm-edge-docs/main/manifests/apps/bookthief.yaml

kubectl apply -f https://raw.githubusercontent.com/flomesh-io/osm-edge-docs/main/manifests/apps/bookstore.yaml

kubectl apply -f https://raw.githubusercontent.com/flomesh-io/osm-edge-docs/main/manifests/apps/bookwarehouse.yaml

kubectl apply -f <https://raw.githubusercontent.com/flomesh-io/osm-edge-docs/main/manifests/apps/mysql.yaml>

### 暴露服务访问方式

把每个服务的GUI端口对外暴露，这样用浏览器我们可以访问这些端口，观察演示的现象。

git clone https://github.com/flomesh-io/osm-edge.git -b main

cd osm-edge

cp .env.example .env

./scripts/port-forward-all.sh #可以忽略错误信息

在一个浏览器中，打开下面的 URL：

注意：如果需要从宿主机访问，需要将 localhost 替换成虚拟机的 IP 地址；或者在宿主机上运行 port-forward-all.sh 脚本。

http://localhost:8080 - bookbuyer

http://localhost:8083 - bookthief

http://localhost:8084 - bookstore

# 访问控制

通过上面的安装的演示应用，所有的服务都是没有访问控制的（宽松流量模式），或者说所有的访问都是允许的。通过在浏览器中观察每个服务的页面数量增长可以看到没有访问控制时候的情况：

在 bookbuyer、bookthief UI 中的计数分别对应了购买和盗窃的书籍数量，而在 bookstore-v1 中这些都应该在增加，在 bookstore UI 中的对于书籍销售的计数也应该在增加。

接下来演示通过禁用宽松流量策略模式，拒绝对 bookstore 服务的访问：

kubectl patch meshconfig osm-mesh-config -n osm-system -p '{"spec":{"traffic":{"enablePermissiveTrafficPolicyMode":false}}}' --type=merge

此时会发现计数将不再增加。

现在执行下面的命令，放行 bookbuyer 对 bookstore 的访问：

kubectl apply -f <https://raw.githubusercontent.com/flomesh-io/osm-edge-docs/main/manifests/access/traffic-access-v1.yaml>

这里再去查看 bookbuyer 和 bookstore UI，会发现计数恢复增加，而 bookthief UI 的计数仍然停止。

通过访问控制，我们成功阻止 bookthief 从 bookstore 盗窃书籍，而正常的购买不受影响。

# 灰度发布

## 部署灰度版本

部署 bookstore-v2：

kubectl apply -f https://raw.githubusercontent.com/flomesh-io/osm-edge-docs/main/manifests/apps/bookstore-v2.yaml

停止并重启 port-forward脚本。

./scripts/port-forward-all.sh

在浏览器中访问 bookstore-v2：http://localhost:8082 - bookstore-v2

## 灰度发布配置

初始化流量拆分策略：

kubectl apply -f <https://raw.githubusercontent.com/flomesh-io/osm-edge-docs/main/manifests/split/traffic-split-v1.yaml>

调整 bookstore-v2 的权重，将 50% 的流量放行到 bookstore-v2 中。

kubectl apply -f <https://raw.githubusercontent.com/flomesh-io/osm-edge-docs/main/manifests/split/traffic-split-50-50.yaml>

将所有流量导入bookstore-v2

kubectl apply -f <https://raw.githubusercontent.com/flomesh-io/osm-edge-docs/main/manifests/split/traffic-split-v2.yaml>

# 监控

使用下面的命令开启命名空间下的 metrics 采集，否则前面创建的 Pod 产生的 metrics 并不会被采集：

osm metrics enable --namespace "bookstore,bookbuyer,bookthief,bookwarehouse"

在执行了端口转发脚本之后，在浏览器中打开 http://localhost:3000 可以访问已经安装的 Grafana，默认的用户名和密码分别为 admin、admin。

# 链路追踪

在浏览器中输入 http://localhost:16686/search 可访问 Jaeger 的仪表板，仪表板中可以查询服务相关的 tracing 信息，并展示服务拓扑。

# 卸载Flomesh

osm uninstall mesh