《Service Mesh 实战》

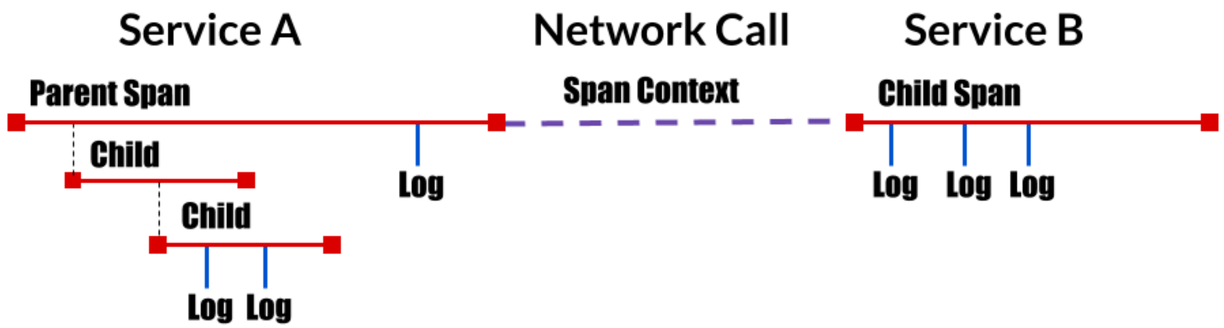
扫码试看/订阅

《Service Mesh 实战》视频课程

3.16 使用 Jaeger 对应用进行分布式追踪

分布式追踪概念

- 分析和监控应用的监控方法
- 查找故障点、分析性能问题
- 起源于 Google 的 Dapper
- OpenTracing：
  - API 规范、框架、库的组合



什么是 Jaeger

- 开源、端到端的分布式追踪系统
- 针对复杂的分布式系统，对业务链路进行监控和问题排查

distributed transaction monitoring

performance and latency optimization

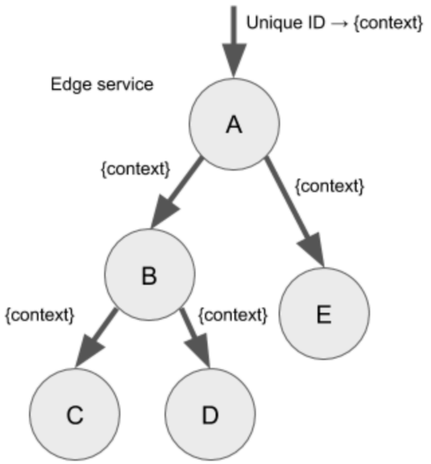
root cause analysis

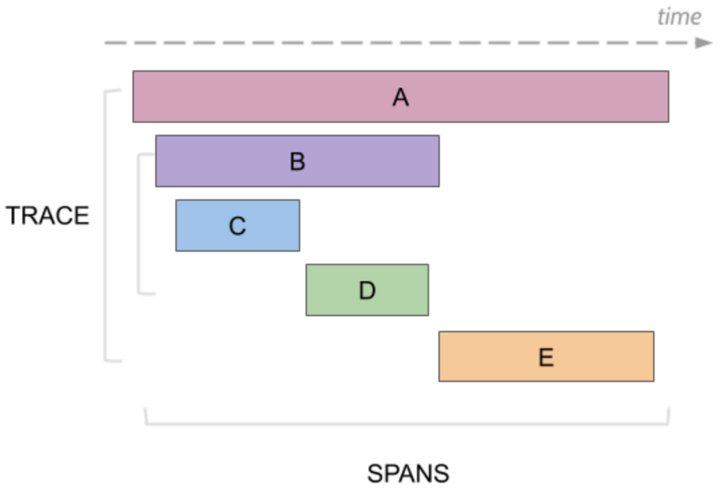
service dependency analysis

distributed context propagation

术语

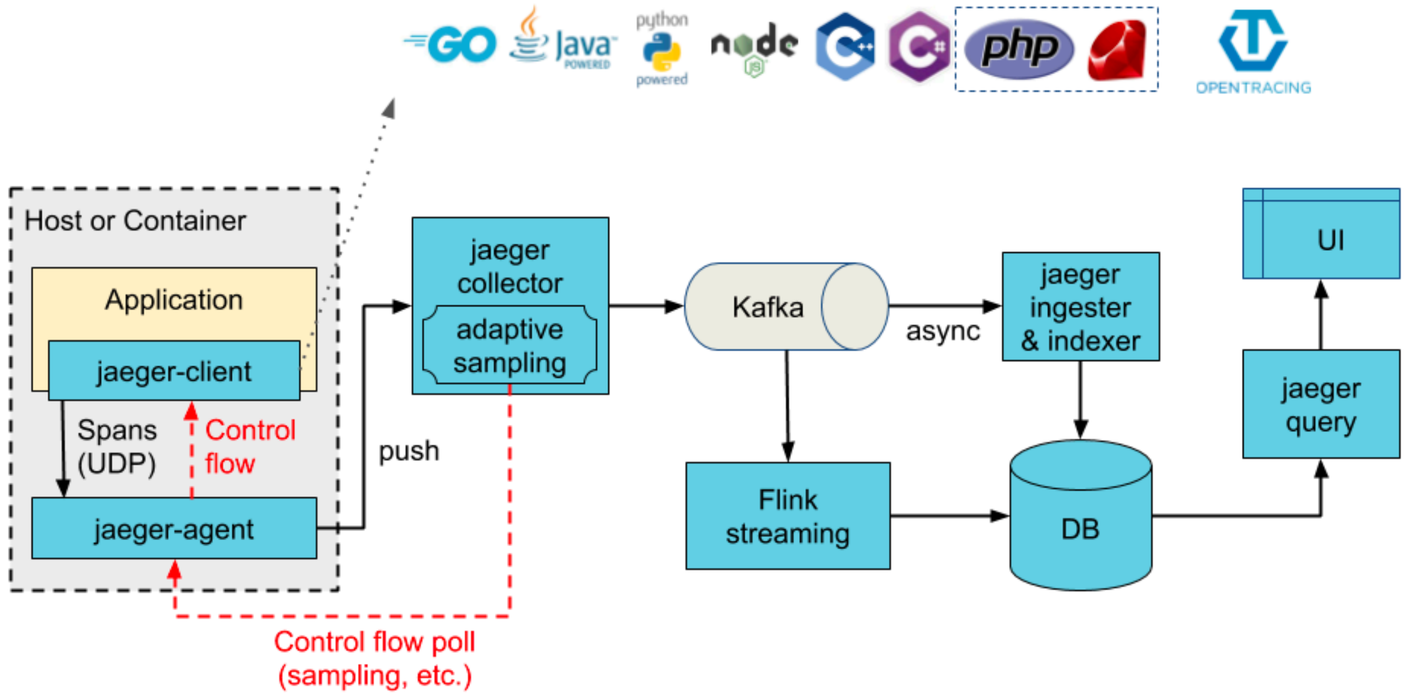
- **Span:**
  - 逻辑单元
  - 有操作名、执行时间
  - 嵌套、有序、因果关系
- **Trace:**
  - 数据/执行路径
  - Span 的组合





Jaeger 架构

- 组件
  - Client libraries
  - Agent
  - Collector
  - Query
  - Ingester



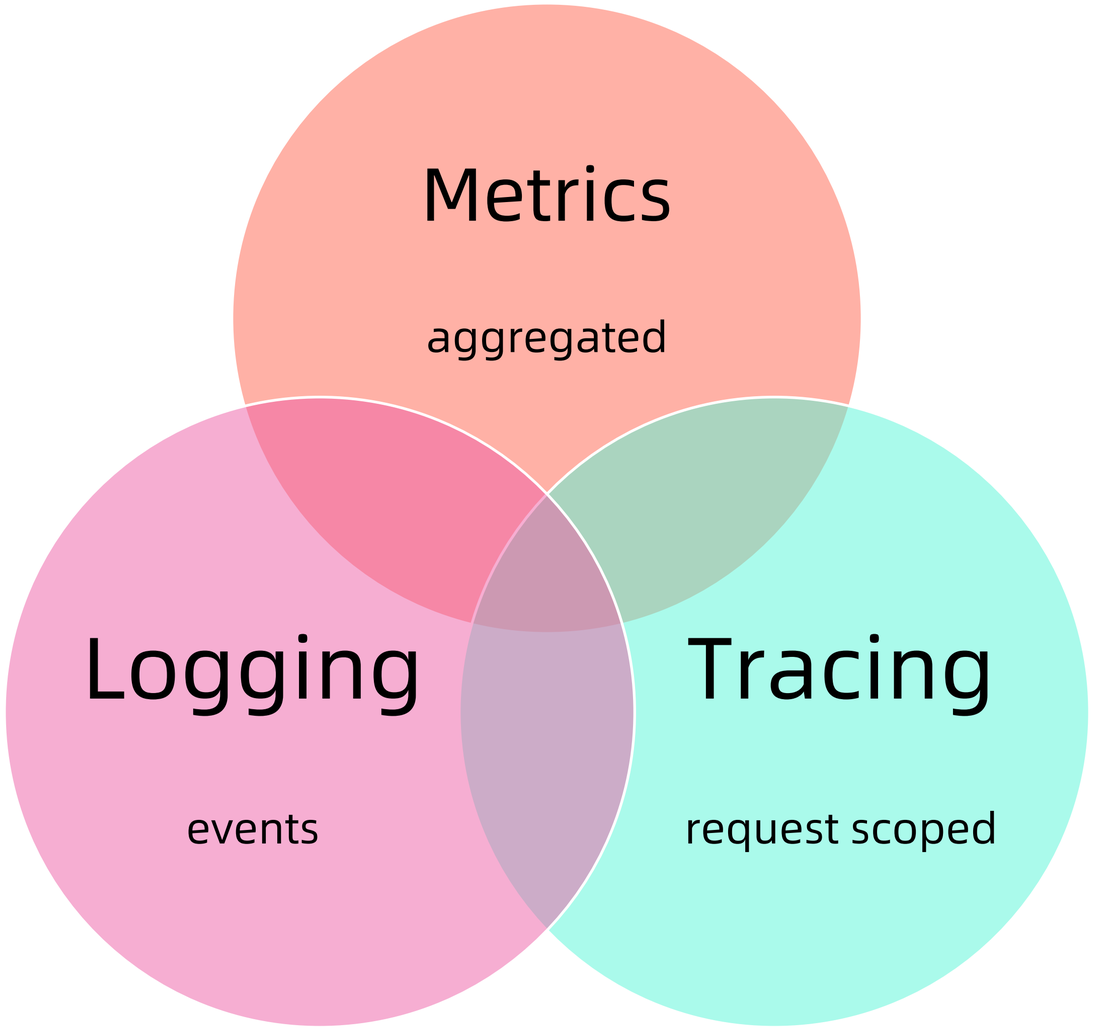
演示

- 确认 Jaeger 安装
  - --set values.tracing.enabled=true
  - --set values.global.tracer.zipkin.address = <jaeger-collector-service>.<jaeger -collector-namespace>:9411
- 打开 Jaeger UI
- 访问 bookinfo 生成 trace 数据

课后练习

- 添加一个延迟，查看 tracing 信息

可观察性小结


Metrics
aggregated
Logging
events
Tracing
request scoped

3.17 配置 TLS 安全网关

Istio 1.5 的安全更新

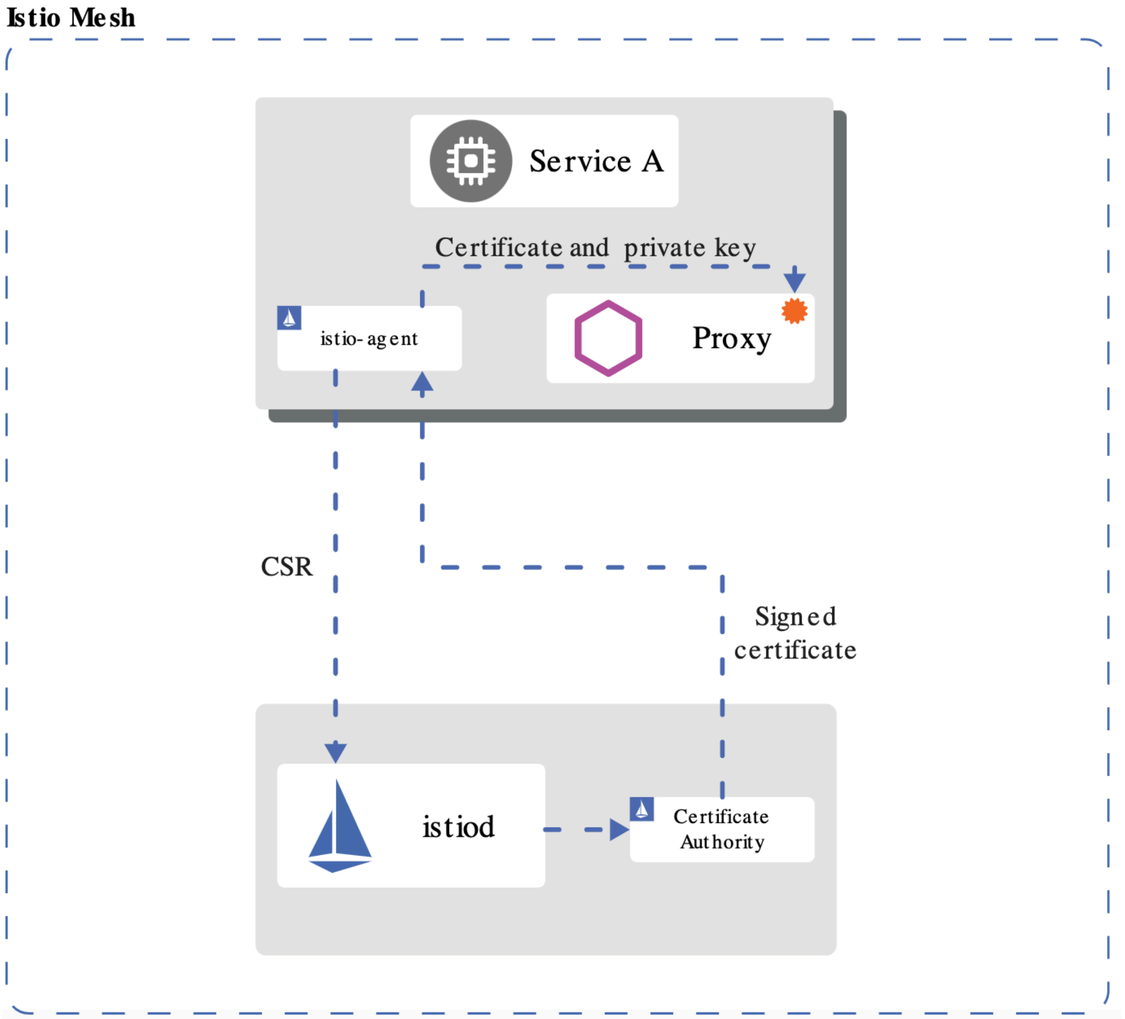
- SDS （安全发现服务）趋于稳定、默认开启
- 对等认证和请求认证配置分离
- 自动 mTLS 从 alpha 变为 beta，默认开启
- Node agent 和 Pilot agent 合并， 简化 Pod 安全策略的配置
- 支持 first-party-jwt （ServiceAccountToken） 作为 third-party-jwt 的备用
- ...

任务：配置基于 SDS 的安全网关

- 任务说明
  - 配置安全网关，为外部提供 HTTPS 访问方式
- 任务目标
  - 学习配置全局自动的 TLS、mTLS
  - 了解 SDS 及工作原理

安全发现服务（SDS）

- 身份和证书管理
- 实现安全配置自动化
- 中心化 SDS Server
- 优点：
  - 无需挂载 secret 卷
  - 动态更新证书，无需重启
  - 可监视多个证书密钥对

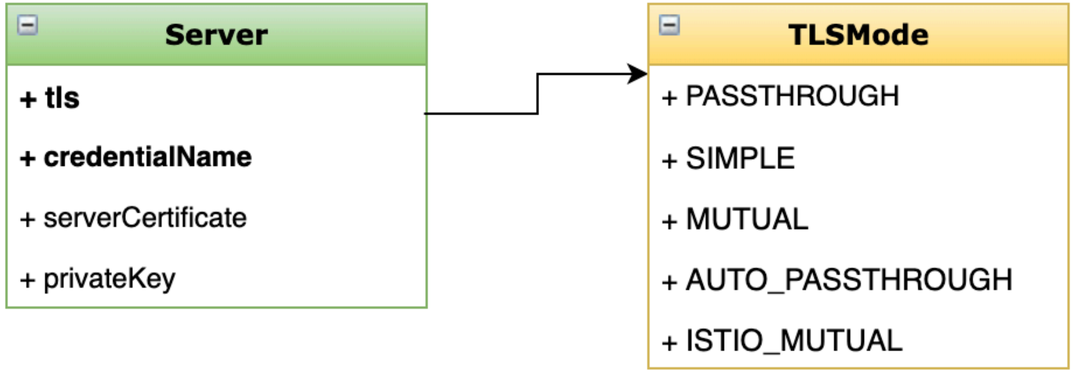


演示

- 确认 curl 命令的编译包
- 生成证书、密钥
- 配置 TLS 网关和路由
- curl 测试

配置分析

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: mygateway
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 443
      name: https
      protocol: HTTPS
    tls:
      mode: SIMPLE
      credentialName: httpbin-credential
    hosts:
    - httpbin.example.com
```



课后练习

- 为网关配置多个 hosts，并配置不同的证书

3.18 为应用设置不同级别的双向 TLS

认证策略

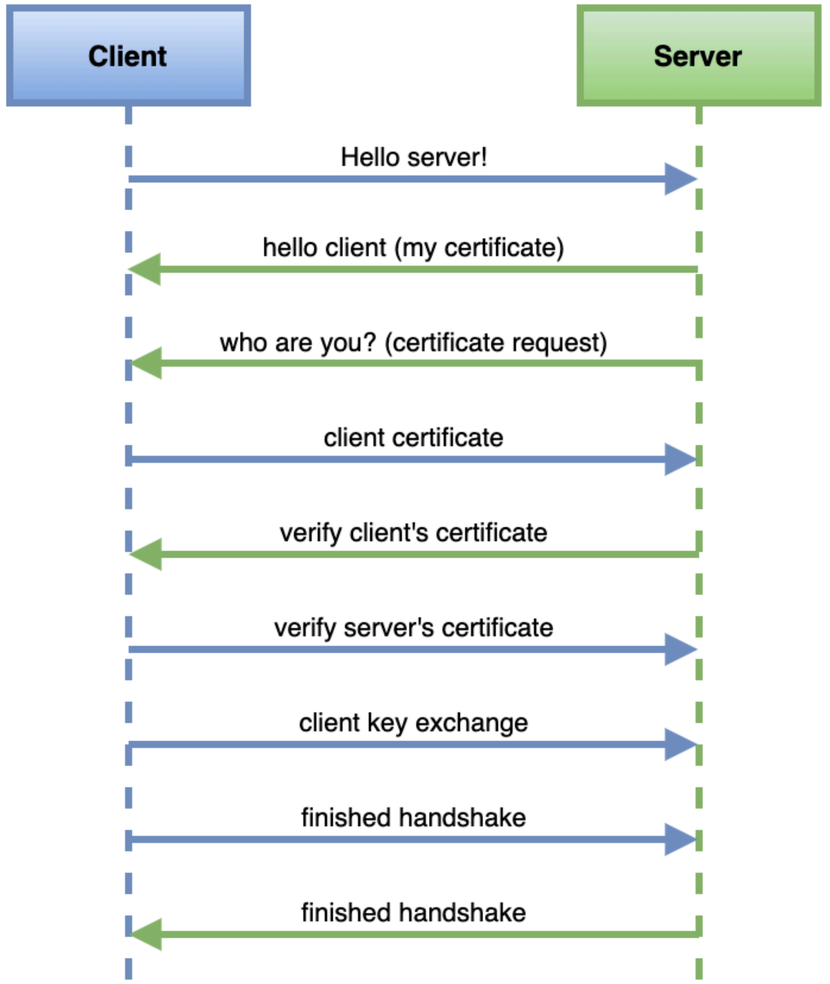
- 认证策略的分类
  - 对等认证（PeerAuthentication）
  - 请求认证（RequestAuthentication)
- 认证策略范围
  - 网格
  - 命名空间
  - 特定服务
- 优先级：最窄原则

任务：开启 mTLS

- 任务说明
  - 为网格内的服务开启自动 mTLS
- 任务目标
  - 学会配置不同级别的 mTLS 策略
  - 理解对等认证的应用场景

mTLS 工作原理

- TLS：客户端根据服务端证书验证其身份
- mTLS：客户端、服务端彼此都验证对方身份



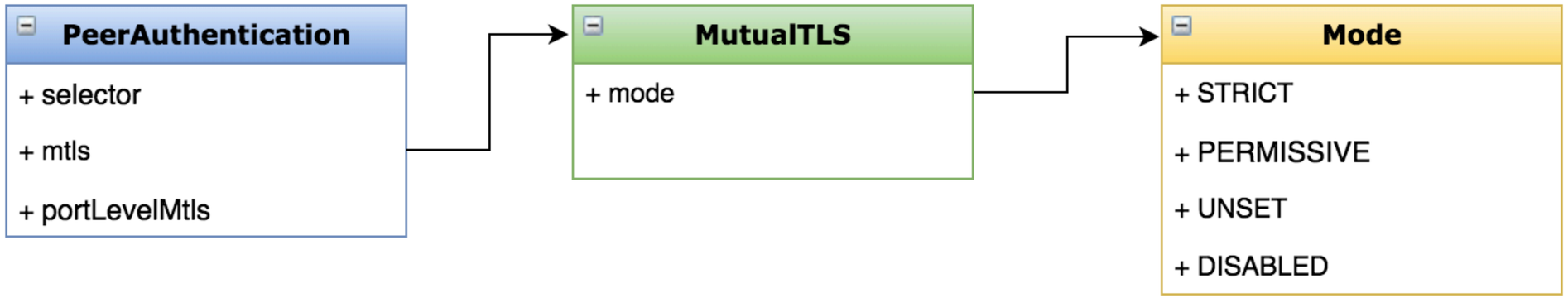
mTLS 握手

演示

- 准备测试用的客户端（sleep）和服务端（httpbin）
- 添加对等认证策略
- 测试基于 namespace 的 mTLS 策略

配置分析

```yaml
apiVersion: "security.istio.io/v1beta1"
kind: "PeerAuthentication"
metadata:
  name: "httpbin"
  namespace: "default"
spec:
  selector:
    matchLabels:
      app: httpbin
  mtls:
    mode: STRICT
```



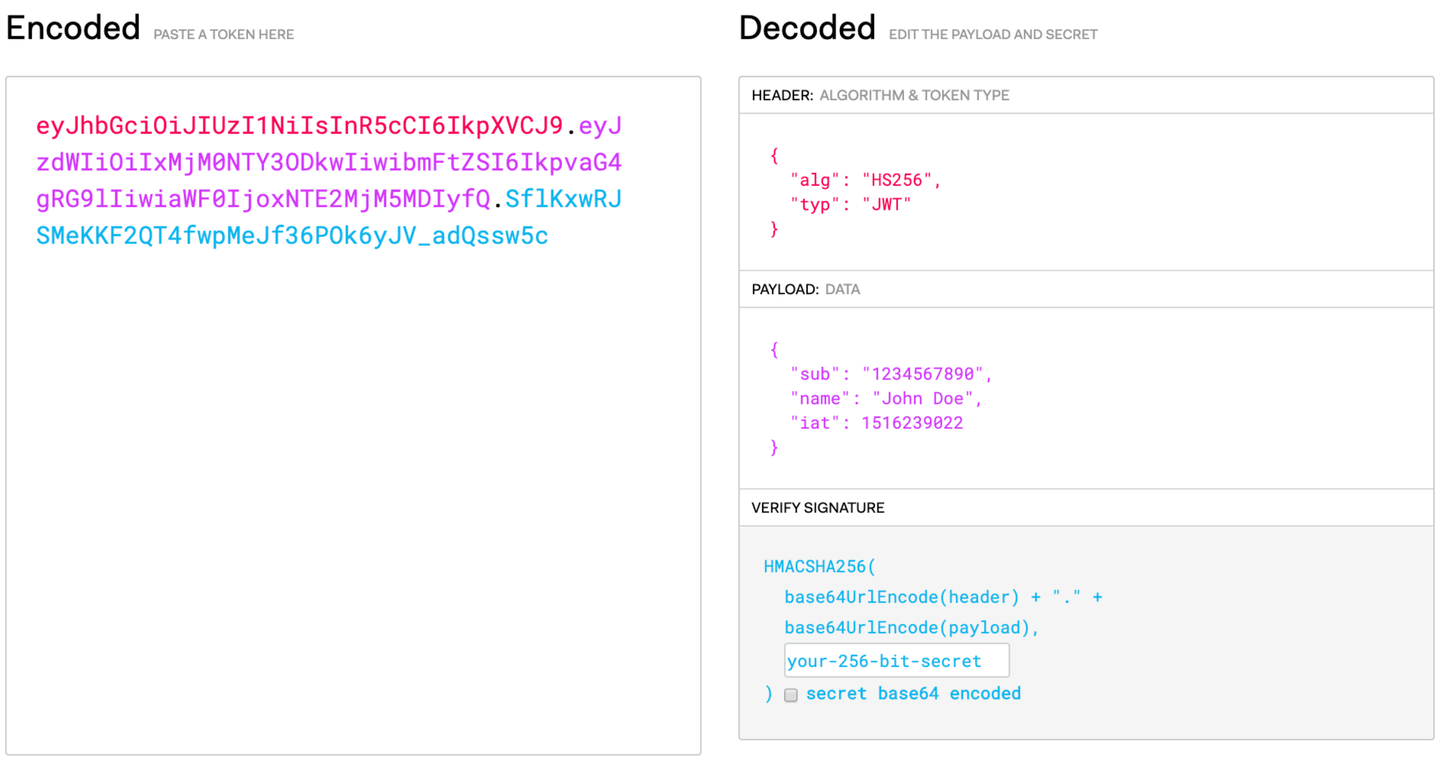
课后练习

- 配置一个服务级别的对等认证

3.19 如何实现 JWT 身份认证与授权？

什么是 JWT

- JSON Web Token
- 以 JSON 格式传递信息
- 应用场景
  - 授权
  - 信息交换
- 组成部分
  - Header、payload、signature



任务：添加 JWT 的授权策略

- 任务说明
  - 实现基于 JWT 的授权访问
- 任务目标
  - 学会配置 JWT 的认证与授权
  - 了解授权策略的配置选项

演示

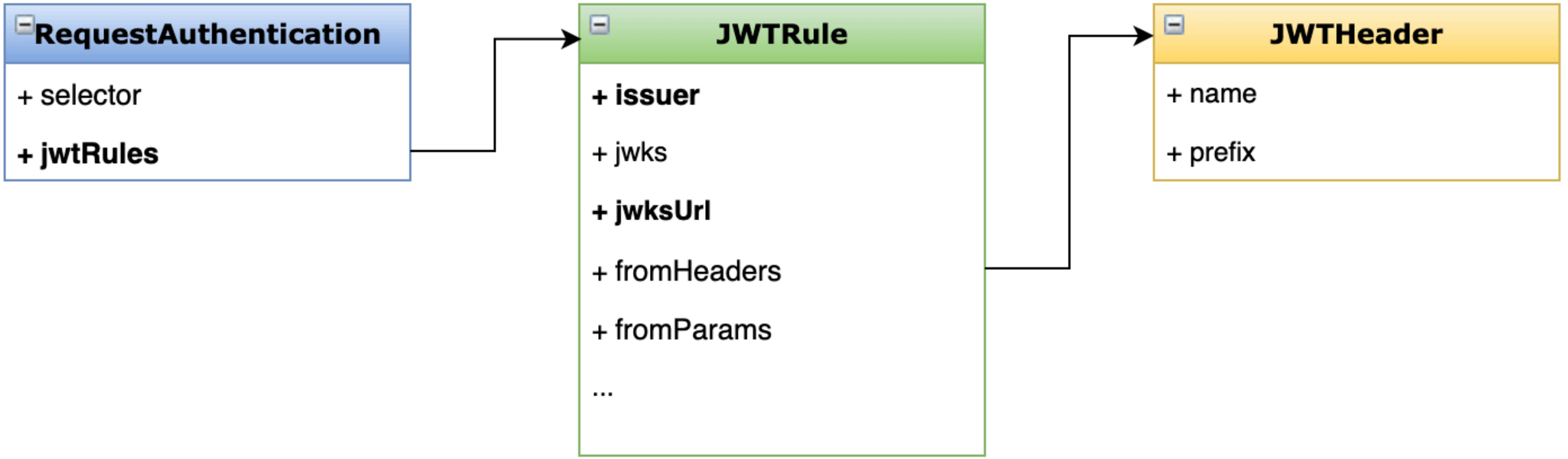
- 创建测试服务
- 配置基于 JWT 的认证策略
- 配置 JWT 的授权策略
- 验证

配置分析

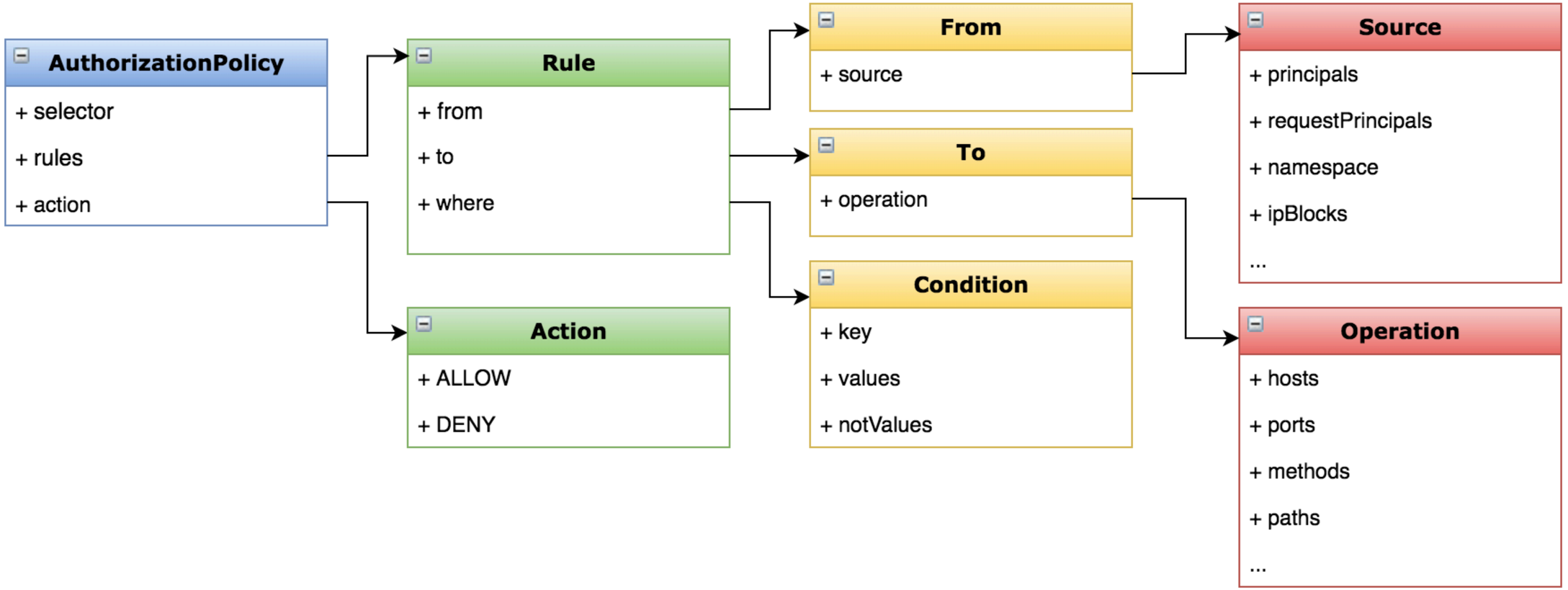
```yaml
apiVersion: "security.istio.io/v1beta1"
kind: "RequestAuthentication"
metadata:
  name: "jwt-example"
  namespace: testjwt
spec:
  selector:
    matchLabels:
      app: httpbin
  jwtRules:
  - issuer: "testing@secure.istio.io"
    jwksUri: "https://raw.githubusercont
```

```yaml
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: require-jwt
  namespace: testjwt
spec:
  selector:
    matchLabels:
      app: httpbin
  action: ALLOW
  rules:
  - from:
    - source:
        requestPrincipals: ["testing@se
```

请求认证配置


RequestAuthentication
+ selector
+ jwtRules
JWTRule
+ issuer
+ jwks
+ jwksUrl
+ fromHeaders
+ fromParams
...
JWTHeader
+ name
+ prefix

授权策略配置


AuthorizationPolicy
+ selector
+ rules
+ action
Rule
+ from
+ to
+ where
Action
+ ALLOW
+ DENY
From
+ source
To
+ operation
Condition
+ key
+ values
+ notValues
Source
+ principals
+ requestPrincipals
+ namespace
+ ipBlocks
...
Operation
+ hosts
+ ports
+ methods
+ paths
...

课后练习

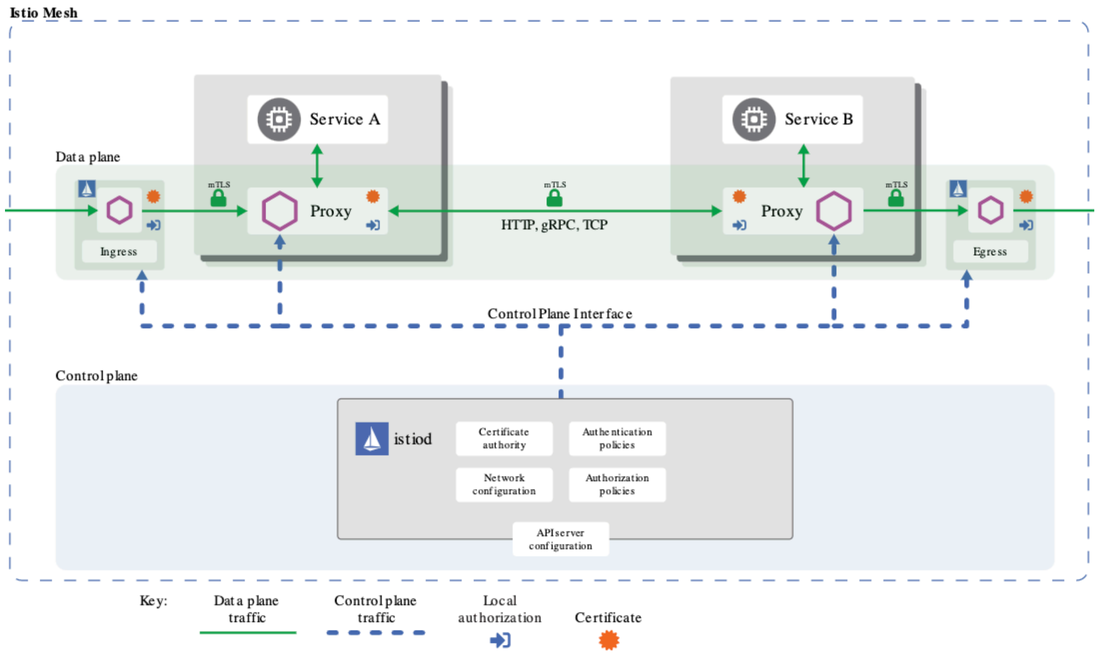
- 实现对特定请求来源的授权
- 提示：在授权规则（Rule.condition）中添加 request.headers 的条件判断
- 条件参考链接：
  https://istio.io/docs/reference/config/security/conditions/

安全功能总结

- 认证（authentication）： you are who you say you are
- 授权（authorization）： you can do what you want to do
- 验证（validation）： input is correct

Istio 的安全机制

- 透明的安全层
- CA：密钥和证书管理
- API Server：认证、授权策略分发
- Envoy：服务间安全通信（认证、加密）



扫码试看/订阅

《Service Mesh 实战》视频课程