稀土掘金 | intel

Kubernetes 中应用身份安全实践

张晋涛
API7.ai 技术专家

{Developers Conference}

```
issue(03).open()
title.set('Computer Traveler')
resize(10.9 * 10.9)
```

{Cooperate}

{Communicate}

JUEJIN 2023

张晋涛

APACHE APISIX PMC

KUBERNETES INGRESS-NGINX 维护者

微软 MVP

《Kubernetes 安全原理与实践》作者

『K8S 生态周报』发起人

公众号：MOELOVE

GITHUB ：https://github.com/tao12345666333

BLOG: HTTPS://MOELOVE.INFO/

```
issue(03).open()
title.set('Computer Traveler')
resize(10.9 * 10.9)
```

< Inspire Developers // To Create & Share >

目录

contents

_01

应用身份安全的重要性

{Developers Conference}

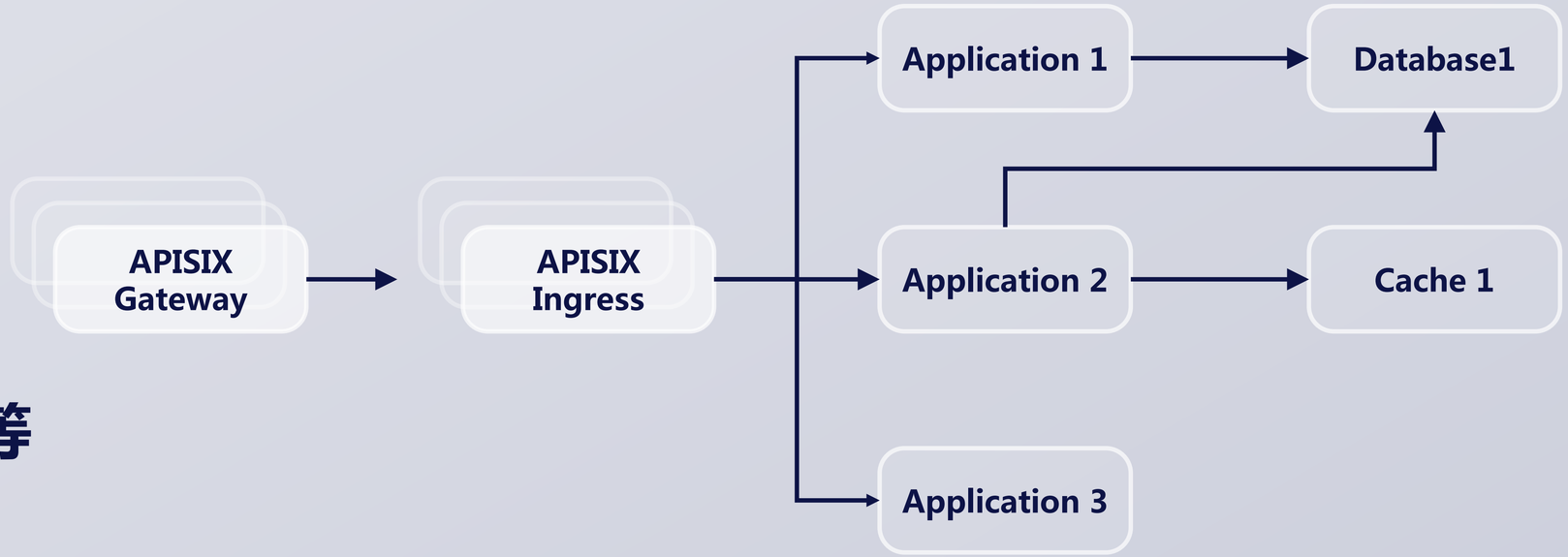
issue(03).open()
title.set('Computer Traveler')
resize(10.9 * 10.9)

{Cooperate}

{Communicate}

Kubernetes 中应用访问链路

- 外层网关
- Ingress
- 应用程序
- 后端数据库 / 缓存等



为什么要重视应用身份安全

多应用间的
安全隔离

数据保护
和加密

防止未授权访问
和越权操作

审计追踪和
恢复能力

_02

零信任安全
如何在 K8s 中构建

{Developers Conference}

issue(03).open()
title.set('Computer Traveler')
resize(10.9 * 10.9)

{Cooperate}

{Communicate}

零信任安全 vs 传统网络

- 传统网络是：城堡和护城河模型
- 通过定义网络边界保护“内网”的安全
- 网络内部安全防护较弱，甚至直接视为安全

- 零信任安全：假设网络内外都会存在安全风险
- 任何网络访问都需要严格的身份验证
- 跨云 / 跨数据中心 / 应用间访问等场景都将获益

如何为应用构建零信任安全

- 此处不讨论：安全访问服务边缘（ SASE ）
- 核心：
  - 标识应用身份（功能 / 部署位置等）
  - 进行认证和鉴权
  - 连接加密（ TLS ）
- 简化：使用 mTLS 可以进行双向验证（服务端 & 客户端）
- 考虑：
  - 身份凭证泄漏，证书轮转
  - 动态 / 大规模环境下自动化处理

SPIRE 是什么

- Secure Production Identity Framework For Everyone (SPIFFE) 规范
- The SPIFFE Runtime Environment (SPIRE) 工具链
- 在各种托管平台上的软件系统之间建立信任

> SPIRE vs RBAC

角色和应用身份是不同的（应用身份粒度更小，更精确）

RBAC 仅支持静态访问决策（无环境动态决策）

RBAC 仅在其上下文中有意义

SPIRE 如何与 K8s 结合

使用 SPIFFEE 标识进行认证

使用 SPIRE 为工作负载颁发证书

使用 SPIRE 管理授权策略

使用 SPIFFEE 标识进行认证

创建适当的 Kubernetes 命名空间和服务账户以部署 SPIRE

将 SPIRE 服务器部署为 Kubernetes statefulset

将 SPIRE Agent 部署为 Kubernetes daemonset

为工作负载配置注册项

通过 SPIFFE 工作负载 API 获取 x509-SVID

使用 SPIFFEE 标识进行认证

> 证书相关：

ca_key_type 和 ca_subject

> 插件相关：

DataStore	NodeAttestor	KeyManager
存储数据	验证器	key 数据

```
server {
  bind_address = "0.0.0.0"
  bind_port = "8081"
  socket_path = "/tmp/spire-server/private/api.sock"
  trust_domain = "example.org"
  data_dir = "/run/spire/data"
  log_level = "DEBUG"
  #AWS requires the use of RSA.  EC cryptography is not supported
  ca_key_type = "rsa-2048"

  ca_subject = {
    country = ["US"],
    organization = ["SPIFFE"],
    common_name = "",
  }
}

plugins {
  DataStore "sql" {
    plugin_data {
      database_type = "sqlite3"
      connection_string = "/run/spire/data/datastore.sqlite3"
    }
  }

  NodeAttestor "k8s_sat" {
    plugin_data {
      clusters = {
        # NOTE: Change this to your cluster name
        "demo-cluster" = {
          use_token_review_api_validation = true
          service_account_allow_list = ["spire:spire-agent"]
        }
      }
    }
  }

  KeyManager "disk" {
    plugin_data {
      keys_path = "/run/spire/data/keys.json"
    }
  }

  Notifier "k8sbundle" {
    plugin_data {
    }
  }
}
```

使用 SPIRE 为工作负载颁发证书

- 部署 SPIRE Server 和 Agent
- 注册 spiffeID
- 应用使用 spiffeID 相互认证
- 创建 SPIRE 服务器策略
- 颁发证书
- 监控审计

```
spire-server entry create \
    -node \
    -spiffeID spiffe://acme.com/web-cluster \
    -selector tag:app:webserver
```

使用 SPIRE 管理授权策略

SPIRE 的 Policy Language （ SPL ）一种基于 JSON 的声明性语言

策略

配置策略

选择器， ID
签发证书

应用策略

将策略文件存储在 SPIRE 服务器的配置目录中或通过使用 SPIRE 服务器 API 来实现

监控策略

确保工作负载只能请求和使用允许的 SPIFFE ID

更新和审计

_03

实现应用身份安全的策略和措施

{Developers Conference}

issue(03).open()
title.set('Computer Traveler')
resize(10.9 * 10.9)

{Cooperate}

{Communicate}

策略

使用 RBAC 进行角色和访问控制（基础）

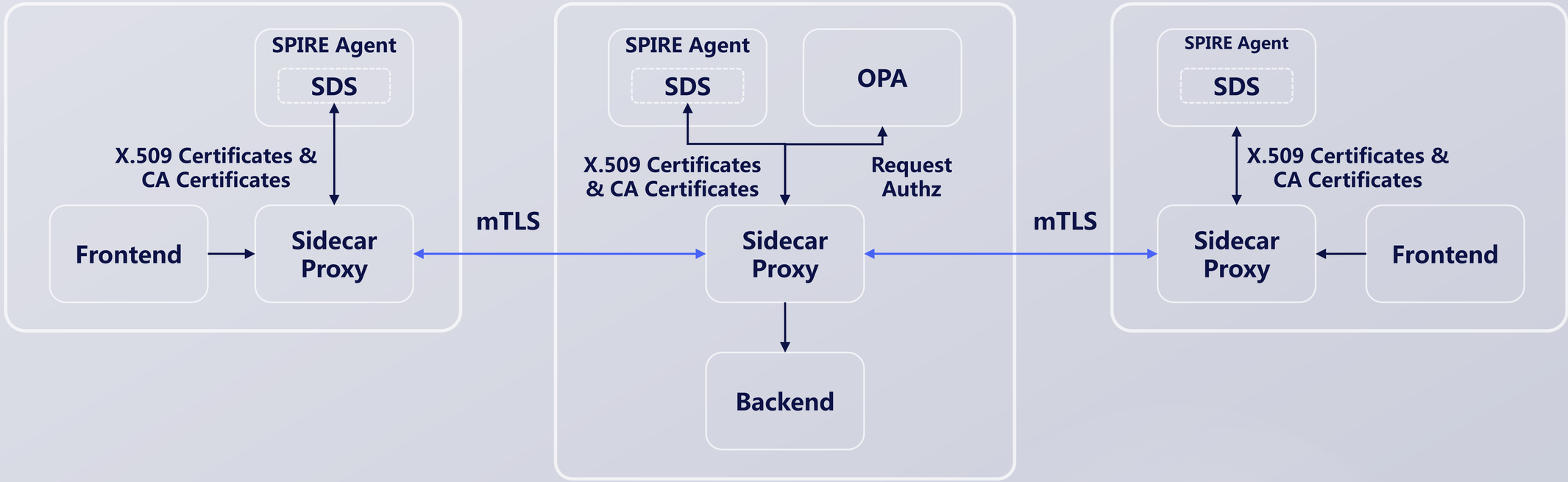
使用 Network Policy 实现网络安全隔离

使用 SPIRE 保护应用间的通信安全

措施

使用 SPIRE 实现身份验证与授权

集成 SPIRE 与 Istio 服务网格



_ 04

总结与展望

{Developers Conference}

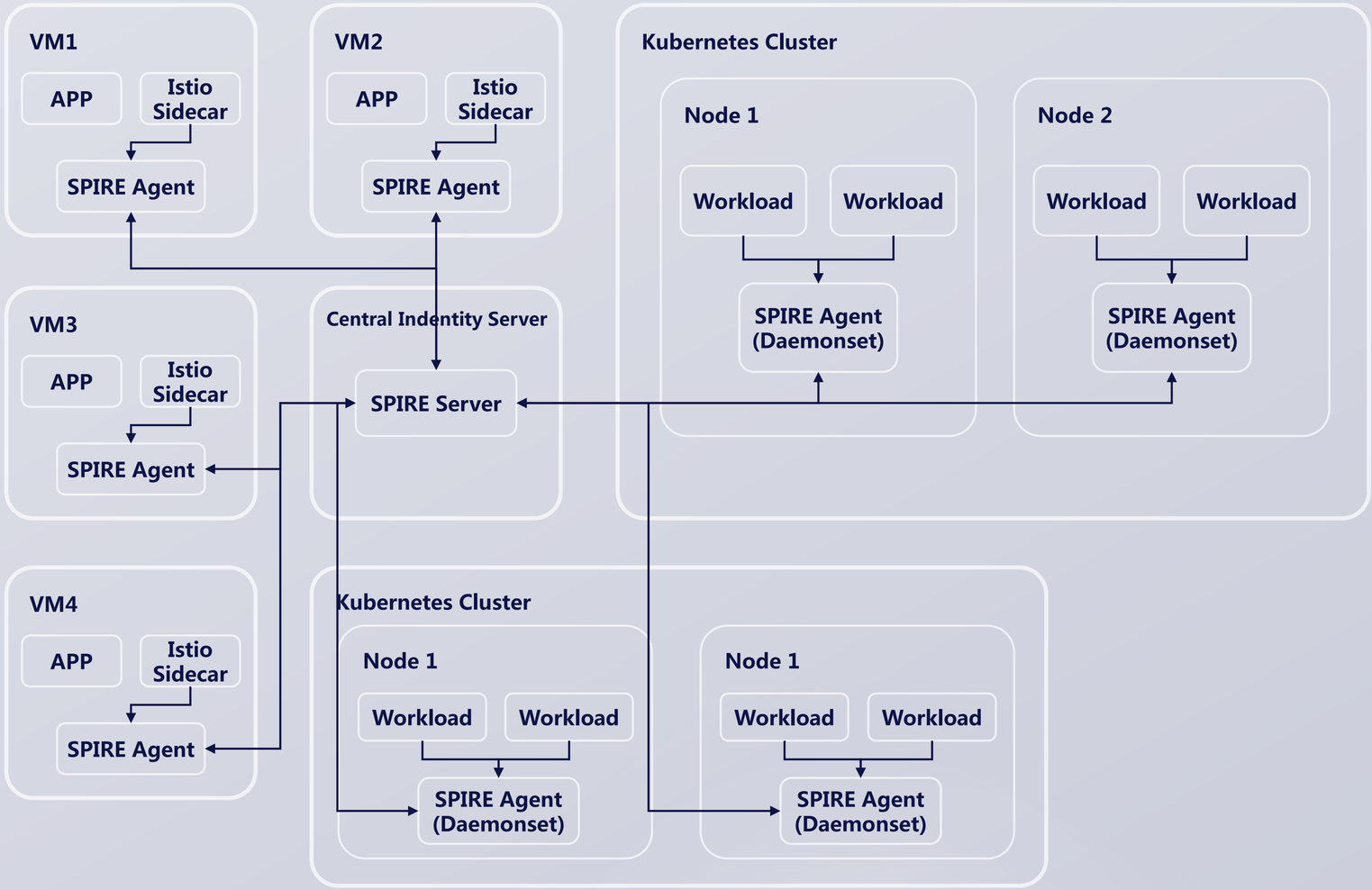
issue(03).open()
title.set('Computer Traveler')
resize(10.9 * 10.9)

{Cooperate}

{Communicate}

总结与展望

- 需开启 mTLS
- Istio v1.14+
- 中间证书轮转（过期 / 升级等）



应用身份安全未来发展趋势及新技术

零信任架构

AI 驱动的身份认证和授权策略

分布式授权管理

API 安全

稀土掘金 | intel

THANKS

{Developers Conference}

```
issue(03).open()
title.set('Computer Traveler')
resize(10.9 * 10.9)
```

{Cooperate}

{Communicate}

JUEJIN 2023