云智未来 9th
第九届中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2017

使用Kubernetes部署超级账本Fabric

张海宁（Henry Zhang)
VMware中国研发现先进技术中心技术总监

自我介绍

- VMware中国研发先进技术中心首席架构师、技术总监
- Harbor开源企业级容器Registry项目创始人
- Cloud Foundry中国社区最早技术布道师之一
- Hyperledger Cello项目贡献者
- 《区块链技术指南》、《软件定义存储》作者之一

公众号：亨利笔记

《区块链技术指南》

《软件定义存储》

议程

1 超级账本项目概览

2 Kubernetes架构简介

3 用Kubernetes部署Fabric

4 总结

超级账本项目概览

商用区块链的要求

多方共享数据
访问权限控制

用代码描述业务
可验证和签名确认

交易具有合适的可见性
交易需认证身份

多方共同认可交易
满足需求的吞吐量

公有链的不足之处

- 比特币、以太坊等公有链项目，不能满足商用的需求
  - 无保密性(Confidentiality)
  - 无法溯源(Provenance)
  - 确认时间长(Slow confirmation)
  - 无最终性(Finality)
  - 吞吐量低(Throughput)
  - 软件许可(license)
  - 极客主导
- 需要新的解决方案

超级账本项目（Hyperledger)

- Linux基金会于2015年12月成立超级账本项目
- 30个创始成员
  - 科技巨头（IBM、Intel、思科等）
  - 金融大鳄（摩根大通、富国银行、荷兰银行等）
  - 专注区块链的公司（R3，ConsenSys等）
- 目前已经超过120个成员
- 150+ 贡献者
- 8000+ commits

超级账本成员

Premier Member

General Member

超级账本目标

- 基于区块链的企业级分布式账本技术(DLT)
- 用于构建各种行业的商业应用平台
- 模块化、性能和可靠性
- 提供商业友好的许可(Apache V2.0)

区块链项目对比

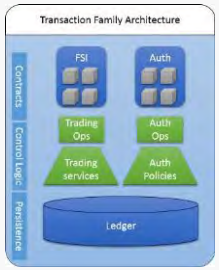
	Hyperledger (Fabric)	Bitcoin	Ethereum
项目定位	通用联盟链平台	数字货币系统	通用公有链平台
管理方式	Linux基金会	社区	社区（众筹）
货币	无	BTC 比特币	Ether 以太币
挖矿	无	有	有
状态数据方式	键值数据、文档数据	交易数据	帐号数据
共识网络	PBFT等	PoW	PoW, PoS
网络	公开或私有	公开	公开
隐私性	有	无	无
智能合约	Go, Java等多种开发语言	无	Solidity

超级账本项目生命周期

- 多个子项目并存
- 每个子项目可有5个阶段

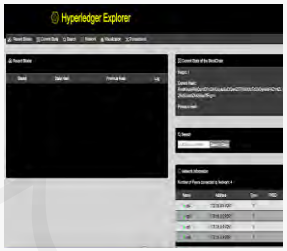
超级账本子项目

Fabric



Sawtooth Lake

Iroha



Blockchain Explorer

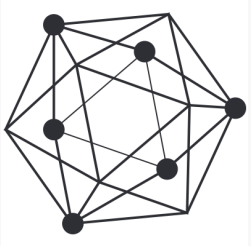
Cello

Burrow

Indy

Composer

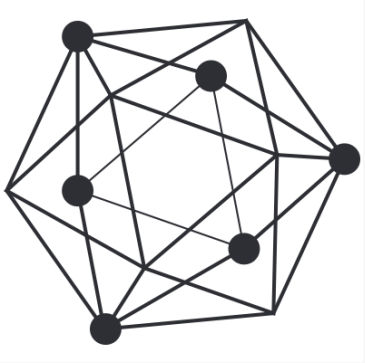
Fabric



- 2015年12月开源
- 主体由IBM的OBC(Open Blockchain)开源代码转化过来
- 增加了DAH和Blockstream两家公司的代码
- 项目以Go语言为主
- 90+贡献者
- 5000+commits

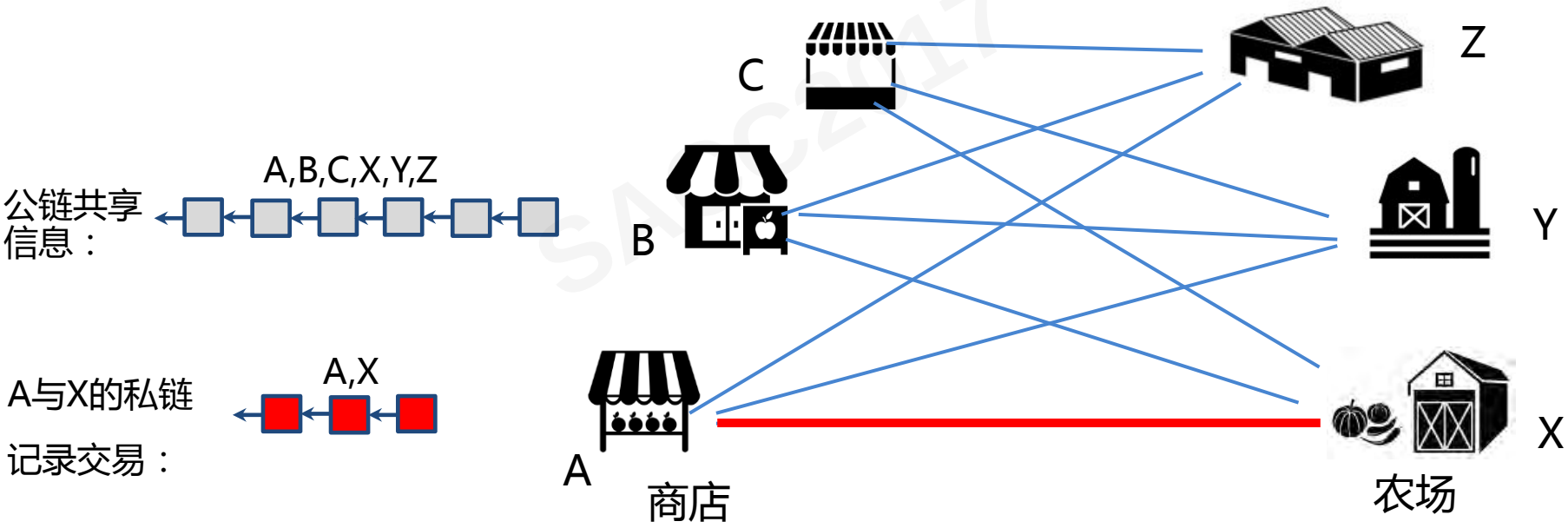
Hyperledger Fabric 1.0 特点

- 提供了交易的机密性
- 权限管理和控制
- 分离了共识和记账职能
- 节点数动态伸缩
- 吞吐量有望提升
- 可升级的智能合约（chaincode）
- 成员服务是高可用



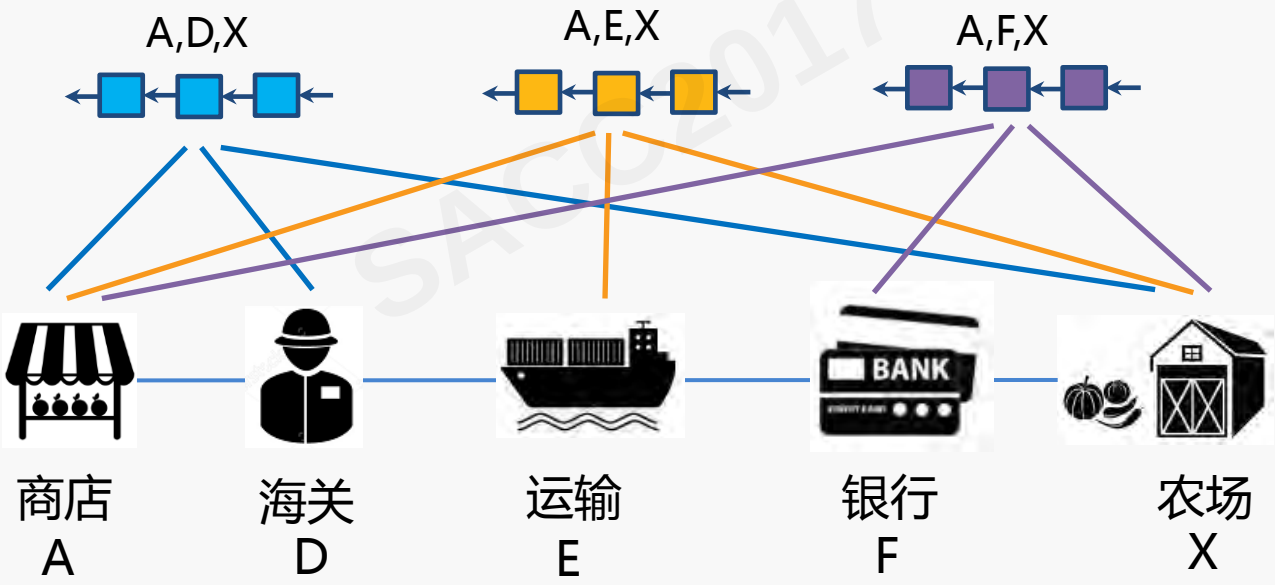
供应链场景(1)

- 公共链：
  - 共享公开信息（如商品种类、报价等）
- 私有链
  - 私密的交易信息(如A购买X的产品）

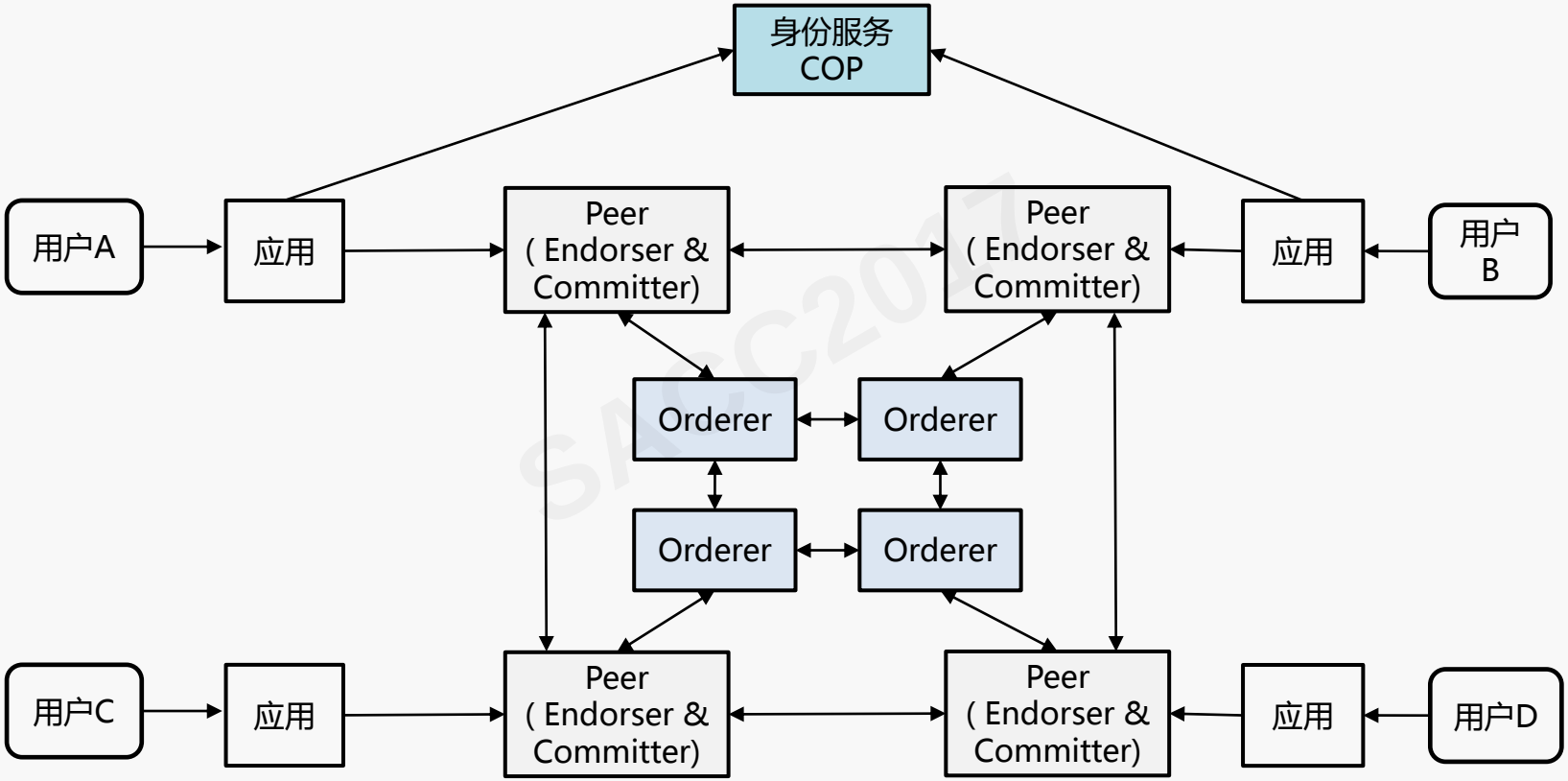


供应链场景(2)

- 不同群体之间构建不同的私链（和账本）
- 互相独立，分别记账



Fabric v1.0部署方式



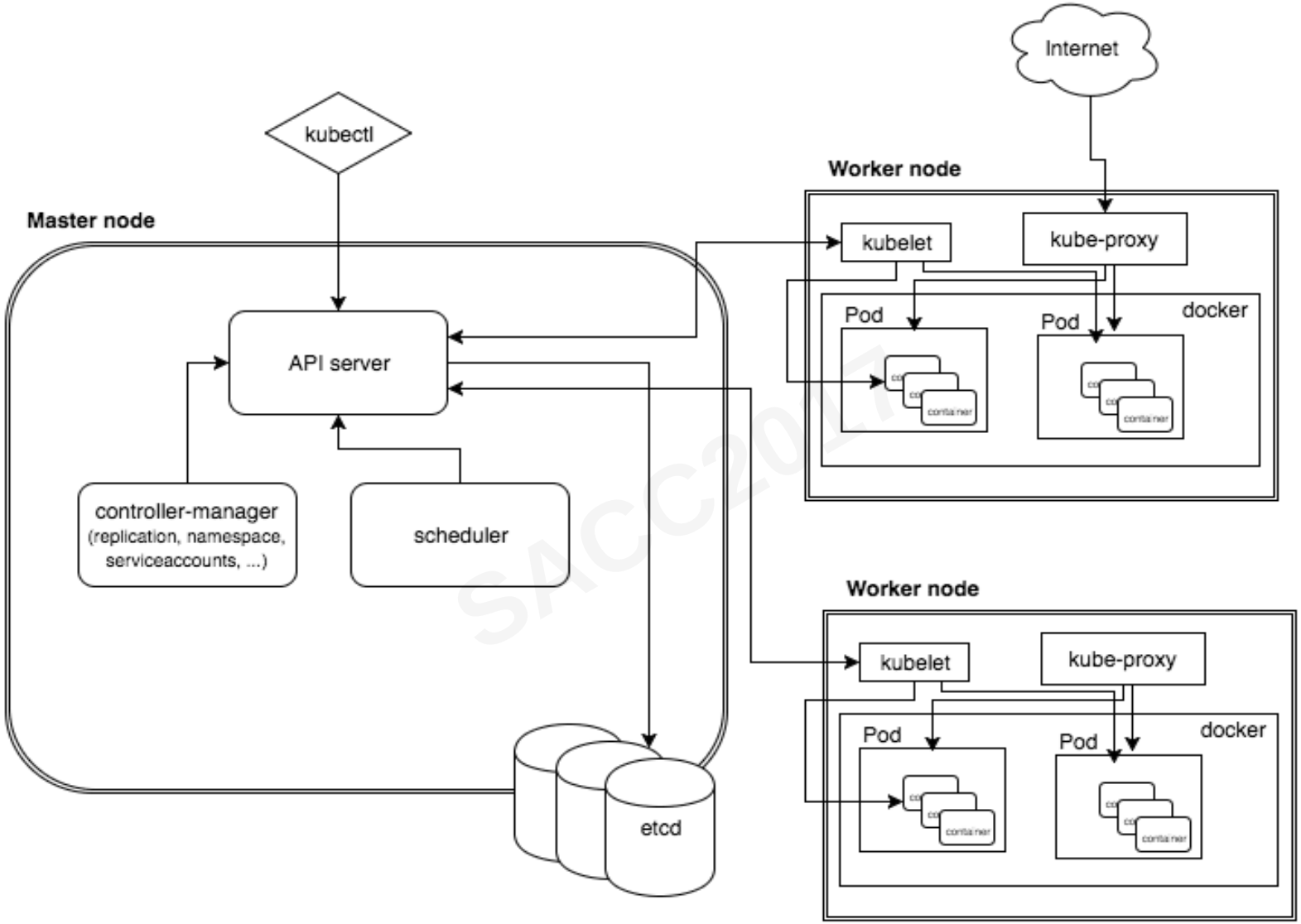
为简明起见，部分箭头未标注

Kubernetes架构简介

Kubernetes (K8s)项目

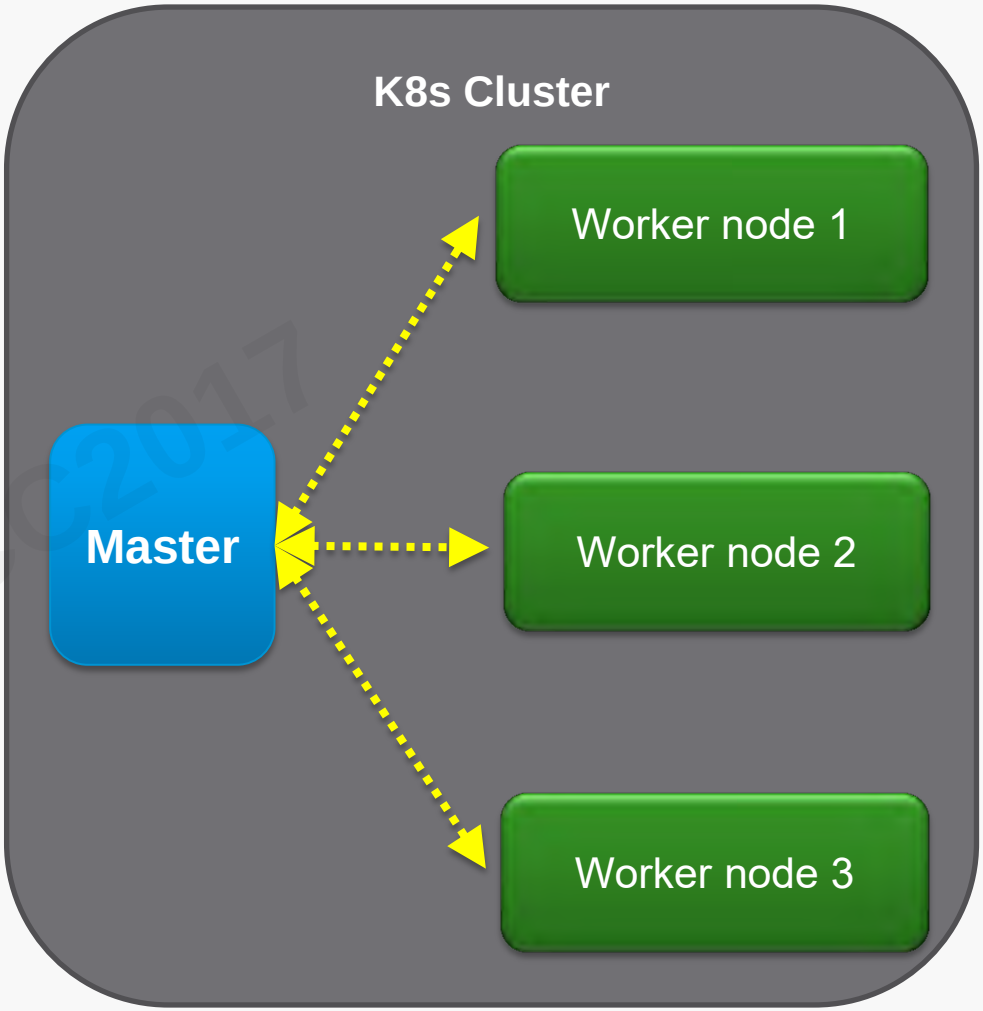
- 开源的容器管理平台
- 自动化编排和部署
- 自动化扩展
- 优化资源使用
- 运维容器化应用
- CaaS，介于PaaS和IaaS之间

Kubernetes 架构



K8s 集群模型

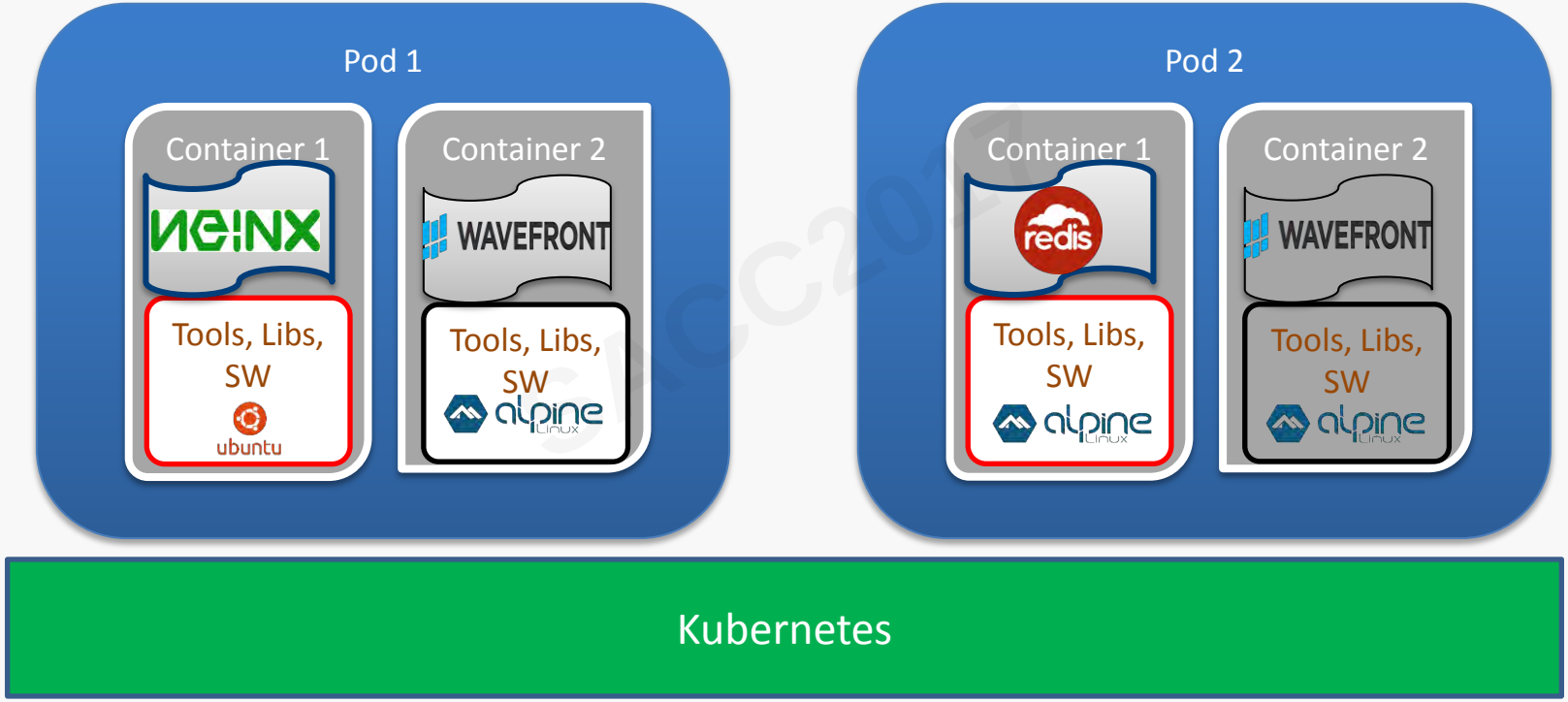
- 一个或多个主节点 (master)
- 一个或多个工作节点(worker)
- 命名空间（Namespaces）
  - 用于命名分隔资源的逻辑组



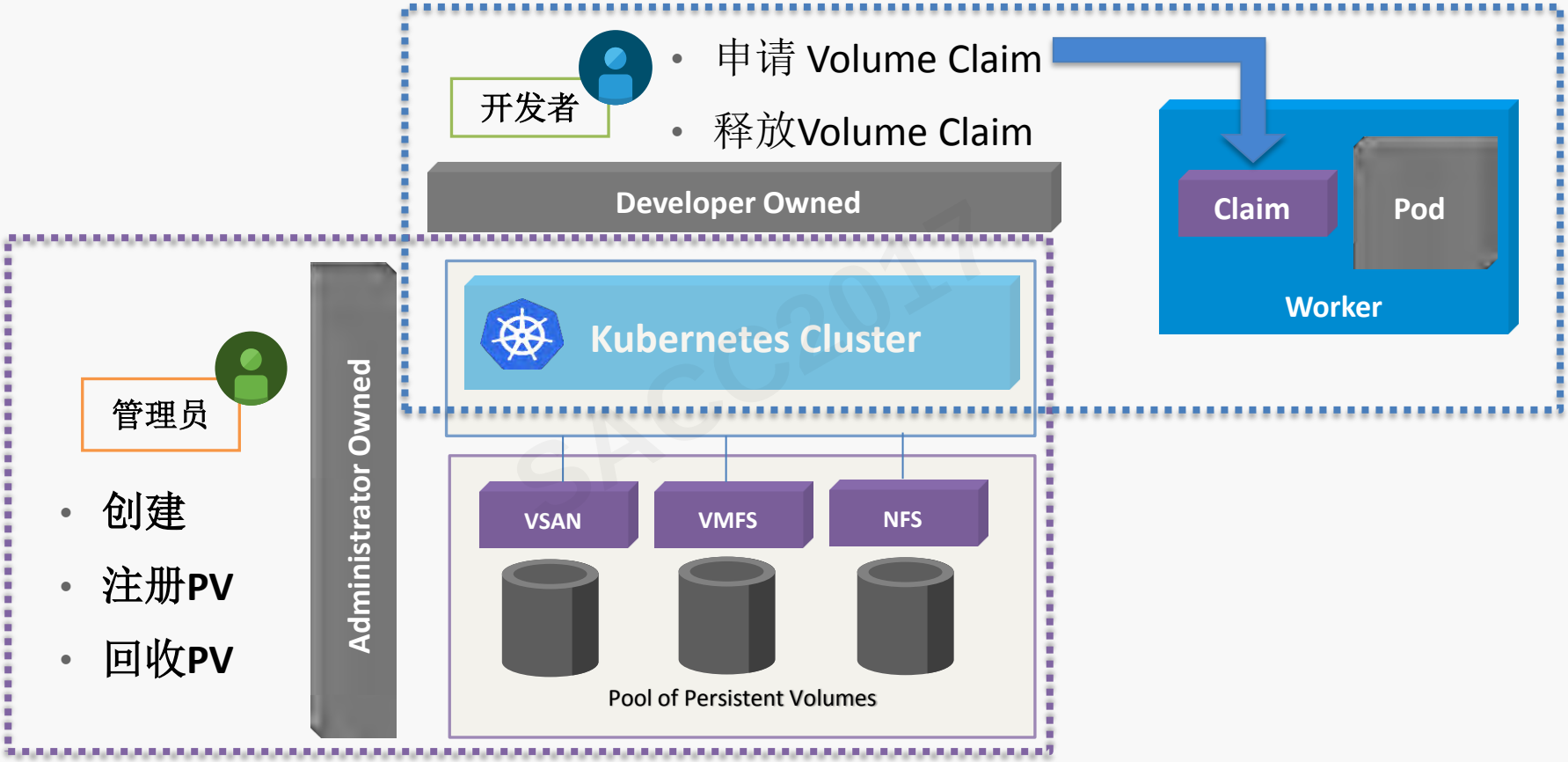
Pod的概念

- Pod是K8s中一个或多个容器组成的部署单位
- 容器共享一个IP地址和端口空间，互相之间用localhost访问
- 容器间还共享数据卷Volumes
- 有点类似虚拟机中的多个进程

Pod部署在K8s中

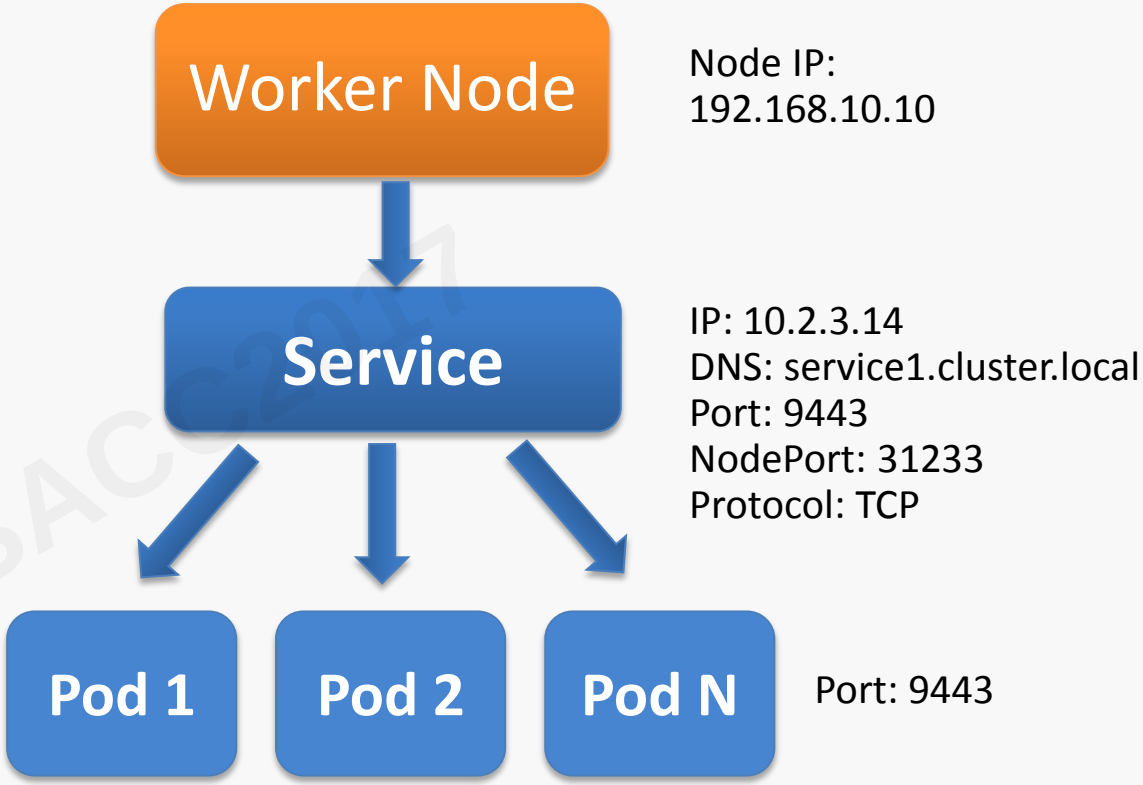

Pod 1
Container 1
NGINX
Tools, Libs, SW
ubuntu
Container 2
WAVEFRONT
Tools, Libs, SW
alpine Linux
Pod 2
Container 1
redis
Tools, Libs, SW
alpine Linux
Container 2
WAVEFRONT
Tools, Libs, SW
alpine Linux
Kubernetes

Kubernetes 的存储： Persistent Volume Claim



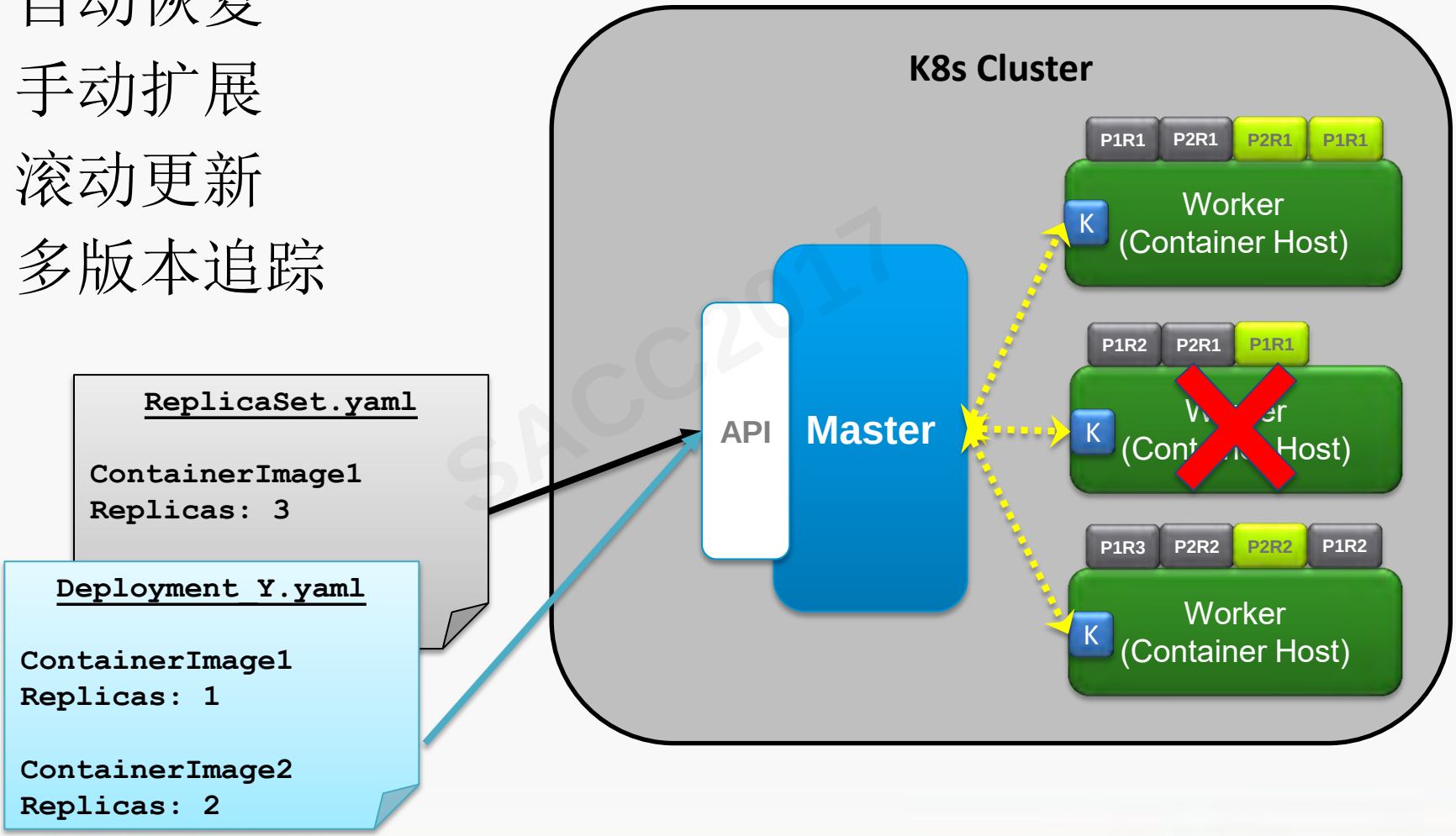
Kubernetes的Services

- 服务类型
  - ClusterIP
  - NodePort
  - LoadBalancer
- 服务发现
  - DNS
  - 环境变量



复制控制器Replication Controller

- 自动恢复
- 手动扩展
- 滚动更新
- 多版本追踪



在Kubernetes中部署Fabric

机遇与挑战

- Fabric的应用面临几个问题：
  - 大量配置文件，繁琐且容易出错。
  - 开发人员无法专注于应用开发。
  - 基于Fabric身份管理的设计，网络中节点增减的步骤繁多。
  - 节点的状态不易监控，节点宕机后需手动重启。

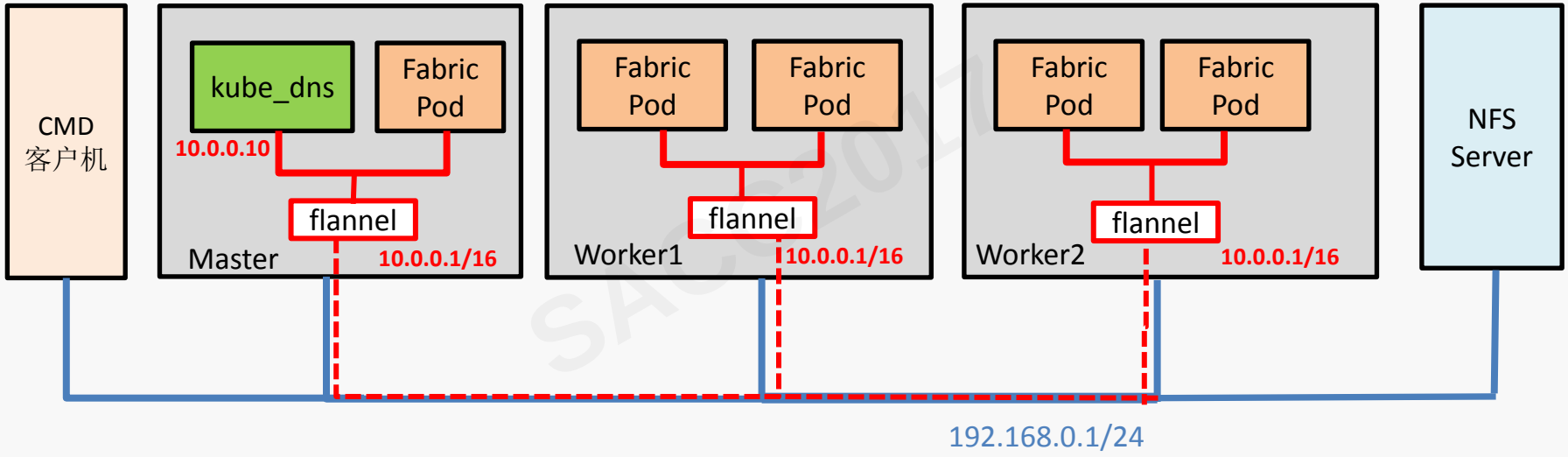
- 降低使用门槛，提高易用性成为区块链应用落地的首要条件。

为什么采用Kubernetes

- Fabric特点
  - 组件都封装成容器，很方便部署在容器平台上
  - 需要灵活地配置和调整

- Kubernetes优势
  - 面向微服务架构的容器平台，扩展方便
  - 提供高可用、监控管理、自动化运维等能力
  - 具备多租户的能力，可运行多个互相隔离的Fabric实例

Kubernetes部署Fabric的网络拓扑



架构 - 网络 (1)

- Kubernetes集群包含一个overlay网络（flannel），容器（Pod）都接入到这个网络。
- K8s的namespace与Fabric的organization做映射，org通过域名进行区分。
- 采用namespace分隔各个组织的组件，配上网络策略来实现多租户的能力。

架构 - 网络 (2)

- Chaincode容器的不在Kubernetes管理范围内，因此worker都需对docker daemon进行DNS配置。

```
"--dns=10.0.0.10 --dns=192.168.0.1  --dns-search \
 default.svc.cluster.local --dns-search \
 svc.cluster.local --dns-opt ndots:2 --dns-opt \
 timeout:2 --dns-opt attempts:2 "
```

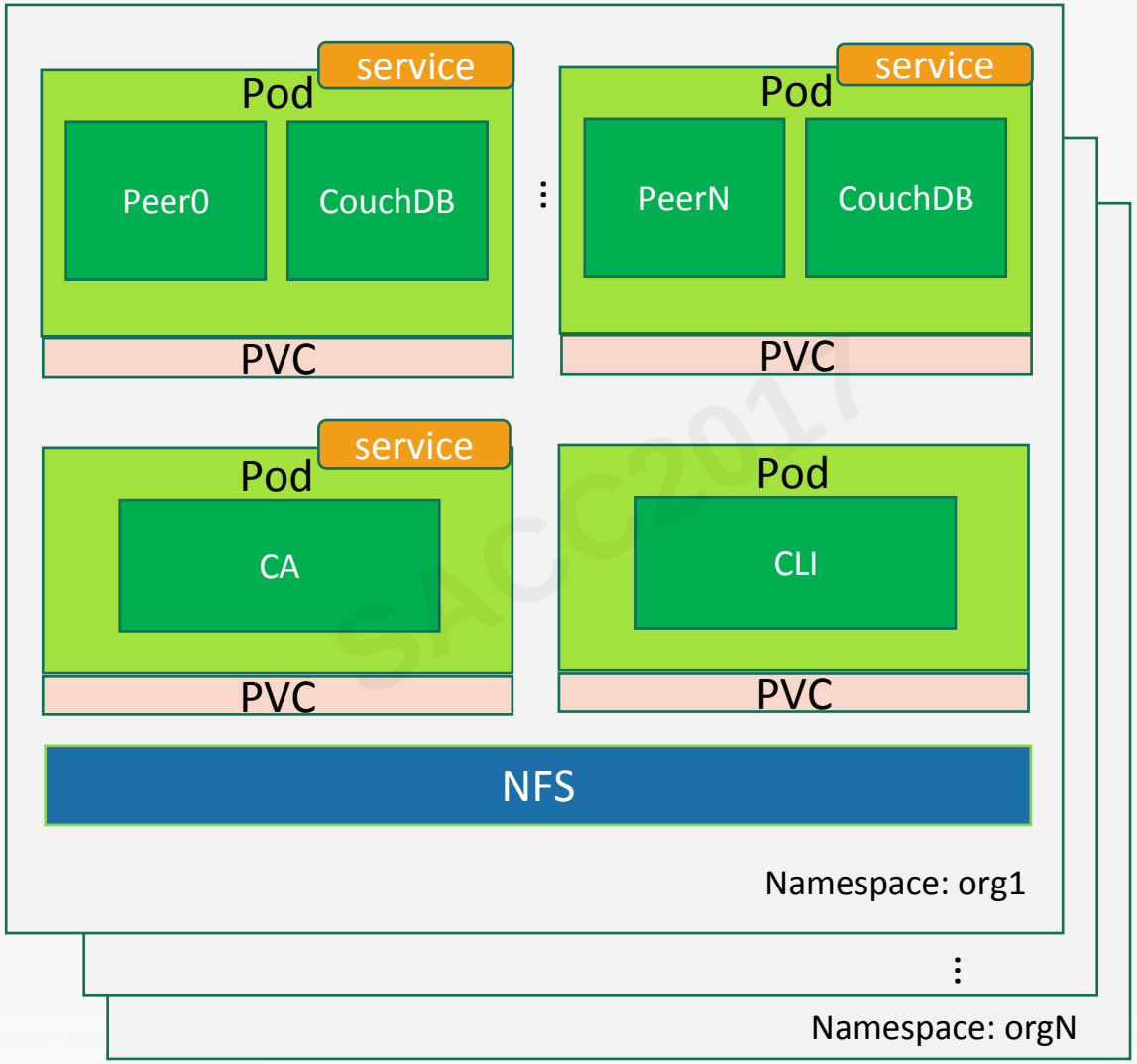
架构 – 共享存储

- 集中生成和存放配置文件，按需导入Pod。
- 支持Pod在各个worker之间的迁移。
- K8s的PV和PVC可以确保每个Fabric的节点只能看到所需要的文件。

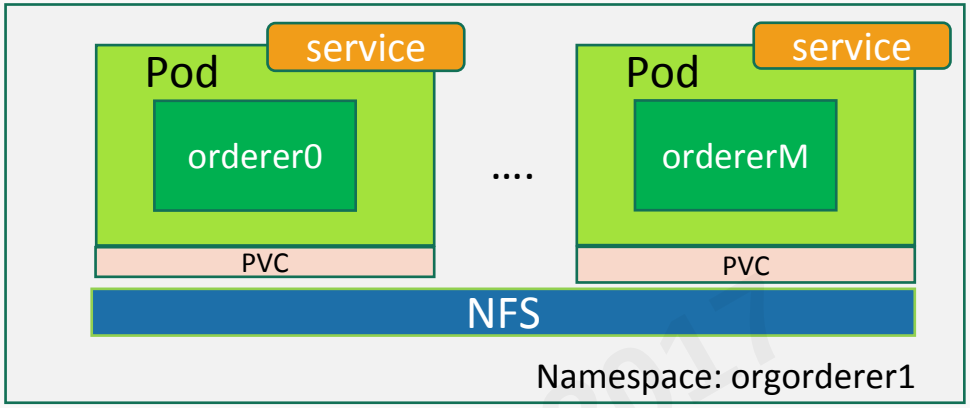
架构 – Fabric组件映射成Pod

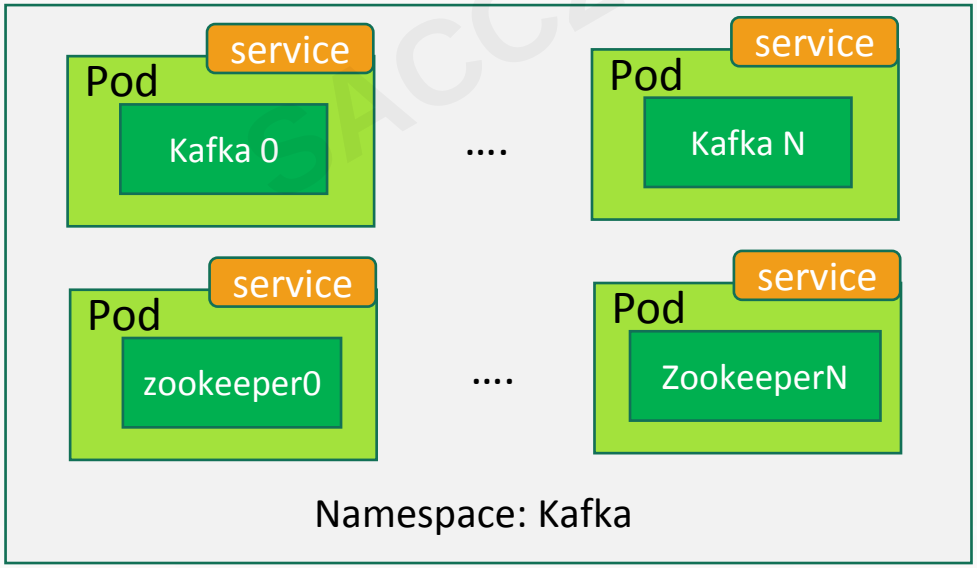
- Peer Pod：包括Fabric peer，couchDB（可选），　代表每个组织的peer节点
- CA Server Pod：　Fabric CA Server
- CLI Pod：　（可选）提供命令行工具的环境，方便操作本组织的节点
- Orderer Pod：运行Orderer节点
- Kafka Pod：运行kafka节点
- Zookeeper Pod: 运行zookeeper节点

用namespace分隔各个组织的Pod

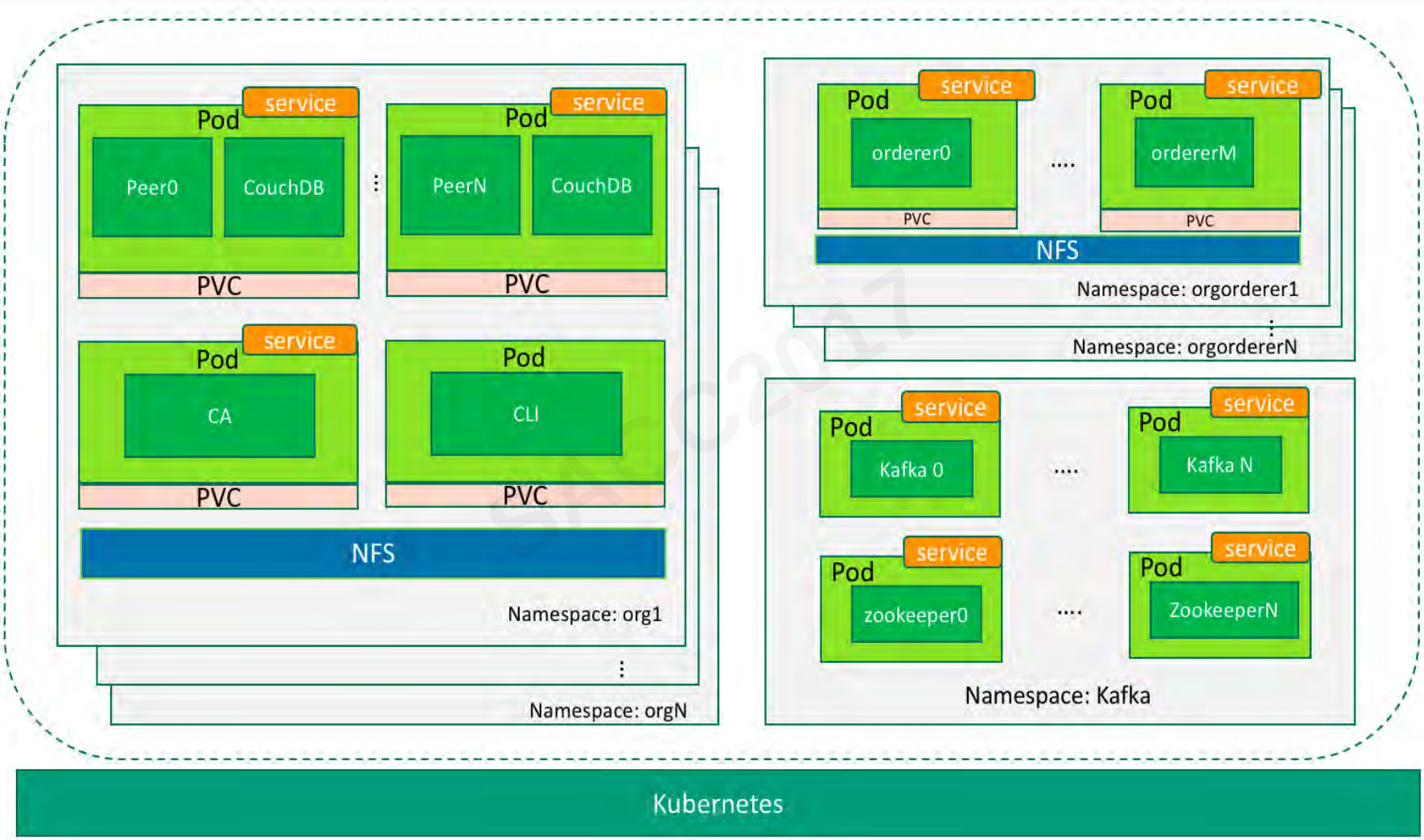


Orderer和Kafka Pod





整体架构


service
Pod
Peer0
CouchDB
PVC
service
Pod
PeerN
CouchDB
PVC
service
Pod
CA
PVC
Pod
CLI
PVC
NFS
Namespace: org1
Namespace: orgN
service
Pod
orderer0
PVC
....
service
Pod
ordererM
PVC
NFS
Namespace: orgorderer1
Namespace: orgordererN
service
Pod
Kafka 0
....
service
Pod
Kafka N
service
Pod
zookeeper0
....
service
Pod
ZookeeperN
Namespace: Kafka
Kubernetes

Service的外部调用

- 在K8s集群外能访问到Fabric中的各个服务
- CA、peer和Orderer的service类型定义为NodePort，
- 端口映射规则如下(N和M的范围分别为N>=1,M>=0):
  - 组织orgN端口范围：`30000+(N-1)*100 ~ 30000+(N)*100-1`
  - CA服务的映射关系：`ca.orgN:7054 -> worker:30000+(N-1)*100`
  - 每个peer需要映射7051和7052两个端口，映射关系如下:

```
peerM.orgN:7051 -> worker:30000+(N-1)*100 + 2 * M + 1
peerM.orgN:7052 -> worker:30000+(N-1)*100 + 2 * M + 2
```

  - ordererN的映射关系为：`ordererN:7050 -> worker:23700+N`

部署- 生成Fabric配置文件

通过cryptogen工具生成证书。cryptogen工具根据cluster-config.yaml来生成证书，并按一定目录存放这些证书:

```
OrdererOrgs:
 - Name: Orderer
   Domain: orgorderer1
   Template:
     Count: 1

PeerOrgs:
 - Name: Org1
   Domain: org1
   Template:
     Count: 1
```

配置文件 → cryptogen → 目录结构

```
crypto-config
    |--- ordererOrganizations
    |        |--- orgorderer1
    |            |--- msp
    |            |---  ca
    |                |--- tlsca
    |            |--- users
    |            |--- orderers
    |            |--- orderer0.orgorderer1
    |              |--- msp
    |              |--- tls
    |
    |--- peerOrganizations
            |--- org1
                |--- msp
                |--- ca
                |--- tlsca
                |--- users
                |--- peers
                |--- peer0.org1
                  |--- msp
                  |--- tls
```

部署 – 生成Pod、namespace配置

- 每个节点需要相应的配置文件。
- 通过模板自动生成各个节点的配置文件。
- 遍历目录结构修改模板，并把修改后的文件放置到相应的目录下，
- 例如org1目录

```
|--- org1
    |--- msp
    |--- ca
    |--- tlsca
    |--- users
    |--- peers
    |--- peer0.org1
```

生成部署文件前

脚本

生成部署文件后

```
|--- org1
    |--- msp
    |--- ca
    |--- tlsca
    |--- users
    |--- peers
    |--- peer0.org1
    |--- org1-ca.yaml
    |--- org1-cli.yaml
    |--- org1-namespace.yaml
```

启动集群

- 已经生成一套相对完整的启动文件，放置在共享存储NFS上
- 通过PV和PVC控制容器对文件的访问权限。
- 启动集群时按照一定顺序启动。

以org1为例：

- 根据定义namespace的yaml文件创建namespace.
- 根据定义ca的yaml文件，创建CA pod 和 service.
- 根据定义cli的yaml文件，创建CLI pod 和 service.
- 遍历org1/peers的子目录找出定义peer的yaml文件，创建peer pod和service.

使用 Fabric Cluster

- 以org1为例，查看namespace为 org1下的所有Pod：

```
$ kubectl get pods -namespaces org1
NAME                                READY     STATUS    RESTARTS   AGE
ca-2708682628-qpz64                 1/1       Running   0          2h
cli-2586364563-vclmr                1/1       Running   0          2h
peer0-org2-3143546256-9prph         2/2       Running   0          2h
peer1-org2-110343575-06pvc          2/2       Running   0          2h
```

- 进入cli-2586364563-vclmr Pod：

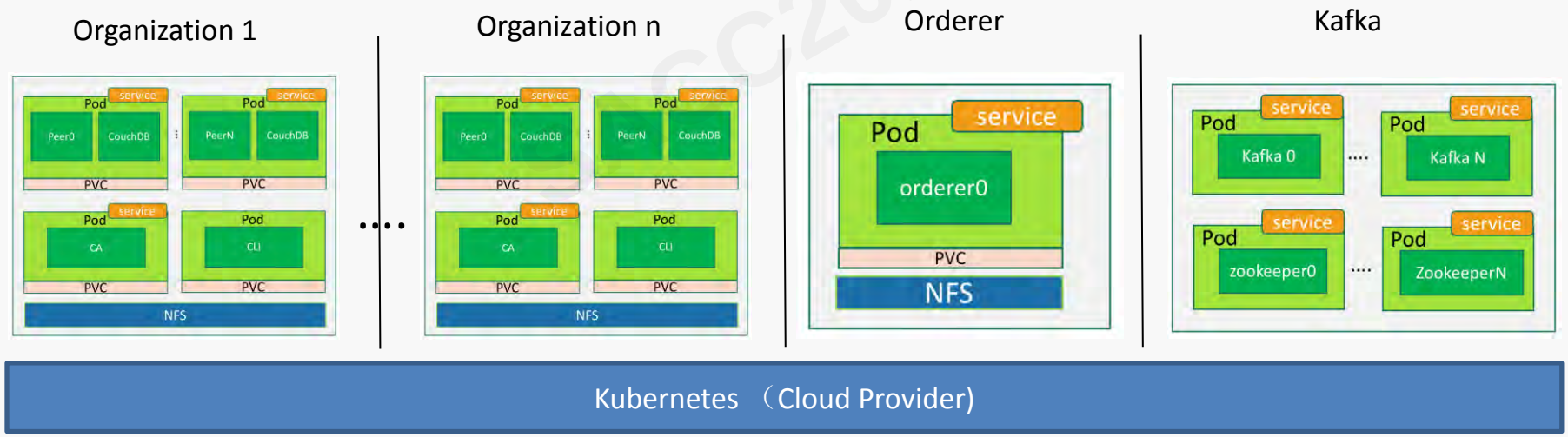
```
$ kubectl exec -it cli-2586364563-vclmr bash --namespace=org1
```

- 执行Fabric命令，如创建channel：

```
$ peer channel create -o orderer0.orgorderer1:7050 \
-c mychannel -f ./channel-artifacts/channel.tx
```

Fabric 区块链即服务（BaaS）云端部署场景

- 同一个Kubernetes集群实现区块链即服务（BaaS）
- 采用网络隔离多租户



总结

- 基于Kubernetes容器云平台初步实现BaaS的基础部署步骤。
- 在此之上，增加更多的区块链层运维管理功能，图形化运维界面，使得开发人员投入更多的精力到应用的业务逻辑上。

- 详细文档和代码：

https://github.com/hainingzhang/article

公众号：亨利笔记

THANKS