使用 Helm Chart 在 Kubernetes 上部署 Fabric

张海宁、陈家豪、胡辉、尹文开
VMware 中国研发中心

本文读者最好对 Helm，Docker，Kubernetes 比较熟悉，同时对 Fabric 架构有一定了解。

我们之前写的文章介绍了如何在 Kubernetes（K8s）上部署 Fabric，该文在社区里面流传较广，很多朋友都按照我们文章的原理实现了通过 K8s 运维 Fabric 的能力。

随着技术的发展，Kubernetes 上的应用不少都采用 Helm Chart 的形式部署，有逐渐成为标准的趋势。Helm 的优点是自动化程度高，一条 Helm 命令能够替代多条 kubectl 命令。

本文介绍 Fabric 的 Helm Chart 部署方式，可按需灵活配置 solo/Kafka 共识算法以及组织和节点的数目。建议读者先查看先前的文章，了解 Fabric 部署在 Kubernetes 上的步骤。

Helm 简介

Helm 是一个 Kubernetes 的包管理工具，用来简化 Kubernetes 应用的部署和管理。Helm 有三个重要概念：

- Chart：定义了一种可以被部署在 Kubernetes 上的软件包格式 。一个 Chart 包含了可以描述 Kubernetes 相关资源的一组文件。
- Config：用来存储软件的配置信息，同 Chart 一起用来创建 Release。
- Release：是 Chart 的一个运行实例。

Helm 由两个重要的功能组件构成：Helm Client 和 Tiller Server。

- Helm Client 是一个给终端用户使用的命令行工具，主要被用来：开发本地 Chart、管理 Chart 仓库和 Tiller Server 进行交互
- Tiller Server 被安装在 Kubernetes 集群中，它一方面接受 Helm Client 发来的请求，另一方面和 Kubernetes API Server 进行交互，包括响应 Helm Client 请求、组合 Chart 和 Config 创建 Release、将 Chart 安装到 Kubernetes 中并持续追踪状态、升级、卸载已安装的 Chart

使用 Helm 部署 Fabric

1. 安装 Helm
具体步骤请参考官方文档：https://docs.helm.sh/using_helm/#installing-helm

2. 准备 NFS server

在我们的部署方案中需要 NFS sever 来存储 Fabric 所需要的配置信息，所以需要提前准备一个可用的 NFS server，关于 NFS server 在部署中的作用请参考我们另一篇文章：用 Kubernetes 部署超级账本 Fabric 的区块链即服务(1) 。

3. 生成 Fabric 所需的证书文件

在 Fabric 的代码库(https://github.com/hyperledger/fabric)中提供了”cryptogen”工具，通过该工具可以快速、批量地生成证书文件，以供用户、节点等 Fabric 网络中的实体使用。

一般情况下，用户可以在一个 yaml 文件中定义证书的组织结构，例如组织的个数，组织的名称和域名等，然后使用”cryptogen”工具读取该配置文件即可生成相应的证书文件。

一个简单的配置文件 cluster-config.yaml 如下：

```
OrdererOrgs:
 - Name: Orderer
   Domain: example.com
   Template:
     Count: 1
PeerOrgs:
 - Name: Org1
   Domain: org1.example.com
   Template:
     Count: 1
 - Name: Org2
   Domain: org2.example.com
   Template:
     Count: 1
```

其中 OrdererOrgs 和 PeerOrgs 关键字用于区分组织（organization）的类型，两种组织的内部结构如下：

- OrdererOrgs 中定义 orderer 组织的信息，Name 字段定义了组织的名称，Domain 字段定义了组织的域名。同时，通过 template 字段可批量生成 orderer 节点。上述的例子中定义了一个名字为 Orderer，域名为 example.com 的组织，并且它指定 template 中 count 的数值为 1，则在该组织下只有一个 orderer，其 id 为 orderer0。
- PeerOrgs 中定义 peer 组织的信息，上述的例子定义了两个组织，分别为 Org1 和 Org2，对应的域名为 org1.example.com、org2.example.com，与 orderer 类似，每个组织生成了一个 peer，虽然 Org1 中 peer0 和 Org2 中 peer0 的 ID 重复，但是它们不属于同一个组织，因此通过域名很容易就能区分出来。

更多配置信息请读者参考源码[1]中的关于 cryptogen 的描述。

[1] http://github.com/hyperledger/fabric/common/tools/cryptogen/main.go。

4.下载 Chart

通过 git clone 下载 chart：
https://github.com/LordGoodman/fabric-chart
或
https://github.com/hainingzhang/articles

下载后，进入 fabric -chart/目录。

5. 配置 Chart 的 values.yaml

1）通过 consensusType 配置所需要的共识算法：solo 或者 Kafka。Helm 程序安装时根据这个参数安装不同的共识算法。
2）修改组织定义。由于在上面第 3 步中生成的"crypto-config"目录有层级结构，该目录的子目录会根据对应的配置文件命名，因此需要在 values.yaml 中配置相同的信息，以确保 Fabric 节点能够正确找到其证书文件。最简单的方法就是把上述 cluster-config.yaml 的内容复制到 values.yaml。

3） 配置 NFS server 信息，以便在 Kubernetes 环境中读取 Fabric 的配置文件。

配置好的 values.yaml 的内容如下：

```
clusterName: mycluster
# consensus type: kafka, solo
consensusType: solo
nfs:
  ip: 10.192.10.10
  basePath: /opt/share
ordererOrgs:
  - name: orderer
    domain: example.com
    template:
      count: 1
peerOrgs:
  - name: org1
    domain: org1.example.com
    template:
      count: 1
  - name: org2
    domain: org2.example.com
    template:
```

```
        count: 1
```

6. 移动证书文件至 NFS server

在第 3 步中生成的证书文件生成后会存放在目录”crypto-config/”中，为了部署在 Kubernetes 中的 Fabric 节点能够正常获取到证书信息，用户还需要把该目录拷贝至 NFS server 上，这样节点就能通过”persistent volume”的方式访问它所需的证书文件。

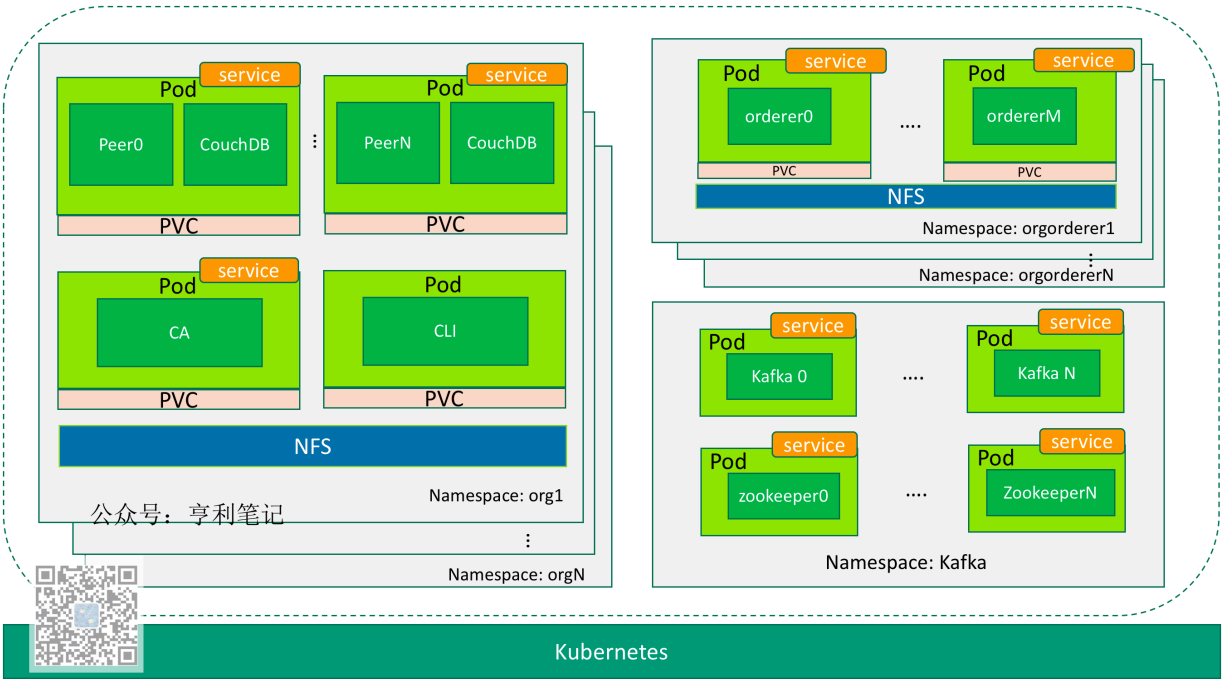
假设 NFS server 中共享的目录为/opt/share，目录 crypto-config 在 NFS 上存放的路径为：/opt/share/<clusterName>/resources/crypto-config，其中 clusterName 用于区分不同 Fabric 集群，根据第 5 步的 values.yaml 文件，这里的 clusterName 为 mycluster。

7. 部署 Chart 到 Kubernetes 中

上述准备完成后，执行以下命令将 Fabric 部署到 Kubernetes 集群中：

```
$ helm install .
```

部署 Fabric 后的架构图如下：



8. 安装原理

Fabric Chart 在设计时可根据 values.yaml 的参数，动态生成部署模板。在实际部署 Fabric 的节点之前，Helm 首先会根据 values.yaml 渲染模版文件，生成部署文件。然后把部署文件推送给 Tiller，最后 Tiller 调用 Kuebrnetes 的 API 完成部署。下面以 Peer 节点的模板文件(`fabric-chart/templates/peer.yaml`)为例，简单讲解模板渲染的工作流程。

```
1. {{- range $peerOrg := $root.Values.peerOrgs}}
```

在模版的定义中，每个 Org 都对应着 Kubernetes 中的一个 namespace，而该 Org 下的每个 Peer 节点都对应着 namespace 下的一个 Pod，因此渲染时最先做的事情是遍历所有 Org。

```
2. {{- $namespace := printf "%s-%s" $orgName $clusterName }}
   {{- $scope := dict "name" $namespace }}
   {{- template "namespace" $scope }}
```

在遍历 Org 的每一次迭代中，为 Org 渲染相应的 namespace，其命名方式为”org.name-clusterName”。

```
3. {{- $name := printf "%s-shared" $namespace }}
   {{- $scope := dict "name" $name "nfsServer" $root.Values.nfs.ip "nfsPath"
   $nfsPath "pvcNamespace" $namespace "pvcName" $sharedPVCName }}
   {{- template "persistentVolume" $scope }}
```

同样是在迭代中，渲染 persistent volume 和 persistent volume claim 以供在该 namespace 下的 Pod 使用。

```
4. {{- range $index := until ($peerOrg.template.count | int) }}
   {{- $name := printf "peer%d" $index }}
   # the deployment of peer
   {{- $scope := dict "name" $name "namespace" $namespace "orgName" $orgName
   "orgDomainName" $orgDomainName "pvc" $sharedPVCName }}
   {{- template "peer.deployment" $scope }}
   # the service of peer
   {{- $name := printf "peer%d" $index }}
   {{- $scope := dict "name" $name "namespace" $namespace }}
   {{- template "peer.service" $scope }}
```

最后根据 peer 所在 Org 的信息渲染 Peer 节点的部署文件。Peer 节点的部署文件由若干个 Kubernetes 资源组成，这些资源涵盖了 deployment、service、ingress 等类型。

模板文件中还定义了 Orderer、CA 和 CLI 等 Fabric 组件的部署，但由于文章篇幅有限便不在这一一列举，具体请读者参考 fabric-chart/templates 下的模板文件。

相关信息：

用 Kubernetes 部署 Fabric（含 Explorer）的工具，请从 VMware Fling 网站下载：
https://labs.vmware.com/flings/blockchain-on-kubernetes

扫码关注公众号：**亨利笔记**，获取更多区块链和云计算等方面的科技文章。

https://github.com/hainingzhang/articles