Fabric CA 应用与配置

简介

Fabric CA项目是超级账本Fabric内的MemberService组件, 对网络内各个实体的身份证书的管理, 主要实现:

- 负责Fabric网络内所有实体(Identity)的身份管理, 包括身份的注册、注销等
- 负责证书管理, 包括ECerts(身份证书)、TCerts(交易证书)等的发放和注销
- 服务端支持基于客户端命令行的RESTful API的交互方式

Fabric CA采用Go语言进行编写

基本组件

Fabric CA采用了典型的C/S架构， 目前包含两个基本组件， 分别实现服务端功能和客户端功能

- 服务端: fabric-ca-server实现核心的PKI(Public Key Infrastructure: 公钥基础设施)服务功能， 支持多种数据库后台(包括SQlite3、MySQL、PostgreSQL等), 并支持集成LDAP用为用户注册管理功能
- 客户端: fabric-ca-client封装了服务端的RESTful API, 提供访问服务端的命令, 供用户与服务端进行交互

  RESTful: 不是什么新技术, 只是一种风格

安装服务端与客户端

安装Go1.10+

设置GOPATH环境变量

安装libtool 与 libltdl-dev 依赖包

```
$ sudo apt update
$ sudo apt install libtool libltdl-dev
```

安装

安装服务端与客户端二进制命令到$GOPATH/bin目录下

```
$ go get -u github.com/hyperledger/fabric-ca/cmd/...
```

切换至源码目录下：

```
$ cd $GOPATH/src/github.com/hyperledger/fabric-ca/
```

使用make命令编译：

```
$ make fabric-ca-server
$ make fabric-ca-client
```

生成 `bin` 目录, 目录中包含 `fabric-ca-client` 与 `fabric-ca-server` 两个可执行文件

设置环境变量

```
$ export PATH=$GOPATH/src/github.com/hyperledger/fabric-ca/bin:$PATH
```

初始化

返回至用户目录

```
$ cd ~
$ mkdir fabric-ca
$ cd fabric-ca
```

fabric-ca启动:

1. 使用init进行初始化
2. 使用start启动

初始化

```
$ fabric-ca-server init -b admin:pass
```

生成配置文件到至当前目录

- fabric-ca-server-config.yaml：默认配置文件
- ca-cert.pem：PEM格式的CA证书文件, 自签名
- fabric-ca-server.db： 存放数据的sqlite数据库
- msp/keystore/：路径下存放个人身份的私钥文件(_sk文件), 对应签名证书

快速启动

快速启动并初始化一个fabric-ca-server服务

```
$ fabric-ca-server start -b admin:pass
```

-b: 提供注册用户的名称与密码, 如果没有使用LDAP, 这个选项为必需. 默认的配置文件的名称为fabric-ca-server-config.yaml

> 如果之前没有执行初始化命令, 则启动过程中会自动先进行初始化操作. 即从主配置目录搜索相关证书和配置文件, 如果不存在则会自动生成

RESTful API

在打开的新终端中输入以下命令获取指定CA服务的基本信息.

```
$ curl -X POST -d '{"caname":"test_ca"}'
http://localhost:7054/api/v1/cainfo
```

> 如要获取默认CA服务的基本信息, 可以不带任何参数, 如:
>
> ```
> $ curl -X POST http://localhost:7054/api/v1/cainfo
> ```

参数说明:

-X: 指定提交请求时的请求方式

-d: 指定提交请求时的参数

服务端命令剖析

fabric-ca-server命令主要负责启动一个CA服务, 包括init和start两个子命令

服务端配置文件解析

fabric-ca-server-config.yaml配置文件包括通用配置, TLS配置, CA配置, 注册管理配置, 数据库配置, LDAP配置, 组织结构配置, 签名, 证书申请等几部分

```
version: 1.1.1-snapshot-e656889
port: 7054              # 指定服务的监听端口
debug: false            # 是否启用DEBUG模式，输出更多的调试信息上
crlsizelimit: 512000

# 是否在服务端启用TLS,如果启用TLS后进行身份验证的证书和签名的私钥
tls:
  enabled: false        # 是否启用TLS，默认不启用
  certfile:         # TLS证书文件
  keyfile:          # TLS密钥文件
  clientauth:    # 客户端验证配置
    type: noclientcert      # 默认不进行身份验证
```

```
    certfiles:       # 进行客户端身份验证时，信任的证书文件列表

# 包括实例的名称、签名私钥文件、身份验证证书和证书链文件乖；这些私钥和证书文件会用来作为生成ECert、TCert的根证书
ca:
  name:          # CA服务名称. 可以支持多个服务
  keyfile:       # 密钥文件(默认: ca-key.pem)
  certfile:      # 证书文件(默认: ca-cert.pem)
  chainfile:     # 证书链文件(默认: chain-cert.pem)

crl:
  expiry: 24h

# 当fabric-ca-server自身提供用户的注册管理时使用，此情况下需要禁用LDAP功能，否则fabric-ca-server将会把注册管理数据转发到LDAP进行查询
registry:
  # 允许同一个用户名和密码进行enrollment的最大次数，-1为无限制，0为不支持登记
  maxenrollments: -1
  identities:   # 注册的实体信息，可以进行enroll. 只有当LDAP未启用时起作用
      - name: admin
        pass: pass
        type: client
        affiliation: ""
        attrs:
           hf.Registrar.Roles: "peer,orderer,client,user"
           hf.Registrar.DelegateRoles: "peer,orderer,client,user"
           hf.Revoker: true
           hf.IntermediateCA: true       # 该id是否是一个中间层的CA
           hf.GenCRL: true
           hf.Registrar.Attributes: "*"
           hf.AffiliationMgr: true

# 数据库支持SQLite3、MySQL、Postgres. 默认为SQLite3类型的本地数据库. 如果要配置集群，则需要选用MySQL或Postgres后端数据库，并在前端部署负载均衡器(如Nginx或HAProxy)
db:
```

```
  type: sqlite3
  datasource: fabric-ca-server.db       # SQLite3文件路径
  tls:
      enabled: false    # 是否启用TLS来连接到数据库
      certfiles:        # PEM格式的数据库服务器的TLS根证书，可以指定多个，用逗号隔开
      client:
        certfile:       # PEM格式的客户端证书文件
        keyfile:        # PEM格式的客户端证书私钥文件

# 配置使用远端的LDAP来进行注册管理，认证enrollment的用户和密码，并获取用户属性信息．此时，服务端将按照指定的usrfilter从LDAP获取对应的用户，利用其唯一识别名(distinguidhed name)和给定的密码进行验证．
# 当LDAP功能启用时，registry中的配置将被忽略
ldap:
   enabled: false   # 是否启用LDAP，默认不启用
   url: ldap://<adminDN>:<adminPassword>@<host>:<port>/<base>   # LDAP的服务地址
   tls:
      certfiles:    # PEM格式的LDAP服务器的TLS根证书，可以为多个，用逗号隔开
      client:
         certfile:  # PEM格式的客户端证书文件
         keyfile:   # PEM格式的客户端证书私钥文件
   attribute:
      names: ['uid','member']
      converters:
         - name:
           value:
      maps:
         groups:
            - name:
              value:

# 组织结构配置
affiliations:
   org1:
      - department1
```

```
      - department2
   org2:
      - department1

# 签发证书相关的配置包括签名方法、证书超时时间等. fabric-ca-server可以作为用户证书的签发CA(默认情况下), 也可以作为根CA来进一步支持其它中间CA
signing:
    default:    # 默认情况下,用于签署Ecert
      usage:    # 所签发证书的KeyUsage extension域
        - digital signature
      expiry: 8760h
    profiles:   # 不同的签发配置
      ca:   # 签署中间层CA证书时的配置模板
         usage:
           - cert sign  # 所签发证书的KeyUsage extension域
           - crl sign
         expiry: 43800h
         caconstraint:
           isca: true
           maxpathlen: 0    # 限制该中间层CA无法进一步签署中间层CA
      tls:
         usage:
            - signing
            - key encipherment
            - server auth
            - client auth
            - key agreement
         expiry: 8760h

# CA自身证书的申请请求配置. 当CA作为根证书服务时, 将基于请求生成一个自签名的证书; 当CA作为中间证书服务时, 将请求发送给上层的根证书进行签署
csr:
   cn: fabric-ca-server     # 建议与服务器名一致
   names:
      - C: US
        ST: "North Carolina"
        L:
        O: Hyperledger
```

```
        OU: Fabric
   hosts:
     - kevin-hf
     - localhost
   ca:      # 配置后会加入到证书的扩展字段
      expiry: 131400h       # 超时时间
      pathlength: 1         # 允许产生的中间证书的深度

# 配置所选择的加密库
bccsp:
    default: SW
    sw:
        hash: SHA2
        security: 256
        filekeystore:
            keystore: msp/keystore      # 存放密钥文件的路径

# 自动创建除了默认CA外的多个CA实例，如ca1、ca2等
cacount:

# 可以指定多个CA配置文件路径，每个配置文件会启动一个CA服务,注意不同配置文件之间需要避免出现冲突(如服务端口、TLS证书等)
cafiles:


# 当CA作为中间层CA服务时的相关配置．包括父CA的地址和名称、登记信息、TLS配置等.
# 注意：当intermediate.parentserver.url非空时，意味着本CA是中间层CA服务,否则为根CA服务
intermediate:
  parentserver:     # 父CA相关信息
    url:
    caname:

  enrollment:       # 在父CA侧的登记信息
    hosts:          # 证书主机名列表
    profile:        # 签发所用的profile
    label:          # HSM操作中的标签信息
```

```
  tls:      # TLS相关配置
    certfiles:      # 信任的根CA证书
    client:         # 客户端验证启用时的相关文件
      certfile:
      keyfile:
```

与服务器端进行交互

可以采用包括RESTful API在内的多种方式与Fabric-CA服务端进行交互. 其中最方便的方式是通过客户端工具 fabric-ca-client

登记用户

注册管理员需要以管理员身份使用CA Client连接到CA Server，并生成相应的文件

```
$ export PATH=$GOPATH/src/github.com/hyperledger/fabric-ca/bin:$PATH
$ fabric-ca-client enroll -u http://admin:pass@localhost:7054
```

> 如果生成的配置文件不是在当前目录下, 则运行登记命令后可能会产生如下错误
>
> ```
> Error: Response from server: Error Code: 20 - Authorization
> failure
> 解决方式：返回至目录下重新启动服务
> $ cd ~
> $ fabric-ca-server start -b admin:pass
> ```

-u: 进行连接的fabric-ca-server服务地址, 默认为"http://localhost:7054"

该命令访问本地Fabric CA服务, 采用默认的admin用户进行登记. 默认情况下会在用户目录下的.fabric-ca-clien子目录下创建默认的配置文件 `fabric-ca-clien-config.yaml` 和 `msp` 子目录(包括签发的证书文件)

> 可以使用 `$ tree .fabric-ca-client/` 命令查看结构

注册用户

登记后的用户身份可以采用如下命令来注册一个新的用户:

```
$ fabric-ca-client register --id.name kevin --id.type user --
id.affiliation org1.department1 --id.attrs
'"hf.Registrar.Roles=peer,user"' --id.attrs 'hf.Revoker=true'
```

执行后输出:

```
2018/05/02 14:39:28 [INFO] Configuration file location:
/home/kevin/.fabric-ca-client/fabric-ca-client-config.yaml
Password: KDwrXkAFENWW
```

> **命令执行成功后返回该新注册用户的密码**
>
> **如果想使用指定的密码, 在命令中添加选项 --id.secret password 即可**

可以再次使用enroll命令，给kevin这个用户生成msp的私钥和证书

```
$ fabric-ca-client enroll -u
http://kevin:KDwrXkAFENWW@localhost:7054 -M kevinca
```

> -M: 指定生成证书存放目录MSP的路径, 默认为"msp"
>
> 命令执行成功后会在 `.fabric-ca-client` 目录下生成指定的userca目录, 在此目录下生成msp的私钥和证书
>
> 可使用tree使用查看
>
> ```
> $ tree .fabric-ca-client/
> ```
>
> 输出内容如下:
>
> ```
> .fabric-ca-client/
> ├── fabric-ca-client-config.yaml
> ├── kevinca
> │   ├── cacerts
> ```

```
│   │   └── localhost-7054.pem
│   ├── intermediatecerts
│   │   └── localhost-7054.pem
│   ├── keystore
│   │   ├──
8c7e4d893af9b9a5907299097edd69c9adee743f4421f81a2d73ed55a874545e
_sk
│   │   └──
ec63821399143fc422e01596eca622e8c658c0ded8e5c189804180256c40c19e
_sk
│   └── signcerts
│       └── cert.pem
├── msp
    ├── cacerts
    │   └── localhost-7054.pem
    ├── intermediatecerts
    │   └── localhost-7054.pem
    ├── keystore
    │   └──
eaa009831dc406950766cc1df104f6fb3ed4b456faa5f476dc41db817a873d2a
_sk
    └── signcerts
        └── cert.pem
```

登记节点

登记Peer或Orderer节点的操作与登记用户身份类似. 可以通过-M指定本地MSP的根路径来在其下存放证书文件

注册节点:

```
$ fabric-ca-client register --id.name peer0 --id.type peer --
id.affiliation org1.department1 --id.secret peer0pw
```

登记节点

```
$ fabric-ca-client enroll -u http://peer0:peer0pw@localhost:7054 -M
peer0
```

客户端命令剖析

fabric-ca-client命令可以与服务端进行交互, 包括五个子命令:

- register: 注册用户实体
- enroll: 登录获取ECert
- getcacert: 获取CA服务的证书链
- reenroll: 再次登录
- revoke: 吊销签发的实体证书

这些命令都是通过服务端的RESTful接口来进行操作的

enroll命令

向服务端申请签发ECert证书并将文件保存至本地

```
$ fabric-ca-client enroll -u http://admin:pass@localhost:7054
```

getcacert命令

向服务端申请根证书信息并保存至本地主配置目录的msp/cacerts路径下

```
$ fabric-ca-client getcacert -u http://admin:pass@localhost:7054
```

证书命名格式为: 服务器主机名-CA实例名.pem

reenroll命令

利用本地配置信息再次执行enroll过程, 生成新的签名证书材料

```
$ fabric-ca-client reenroll
```

register命令

执行注册新用户实体的客户端必须已经通过登记认证, 并且拥有足够的权限(所注册用户的hf.Registrar.Roles和affiliation都不能超出调用者属性)来进行注册

```
$ fabric-ca-client register --id.name jack --id.type user --
id.affiliation org1.department1 --id.attrs
'"hf.Registrar.Roles=peer,user"' --id.attrs 'hf.Revoker=true' --
id.secret jackpw
```

revoke命令

吊销指定的证书或指定实体相关的所有证书. 执行revoke命令的客户端身份必须拥有足够的权限(hf.Revoker为true, 并且被吊销者机构不能超出吊销者机构的范围)

```
$ fabric-ca-client revoke -e "jack" -r "affiliationchange"
```

-e: 指定吊销用户

-r: 指定吊销原因

输出内容如下

```
2018/05/02 17:55:44 [INFO] Configuration file location: .fabric-ca-
client/fabric-ca-client-config.yaml
2018/05/02 17:55:44 [INFO] Sucessfully revoked certificates: []
```

查看AKI和序列号

AKI: 公钥标识号, 代表了对该证书进行签发机构的身份

查看根证书的AKI与序列号信息:

```
$ openssl x509 -in .fabric-ca-client/msp/signcerts/cert.pem -text -
noout
```

输出内容如下:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:  # 序列号

74:48:88:33:70:1a:01:a0:ad:32:29:6e:c5:ab:5a:fa:3b:91:25:a4
   ......
        X509v3 extensions:
           ......
            X509v3 Authority Key Identifier:    # keyid后面的内容就是
AKI

keyid:45:B1:50:B6:CD:8A:8D:C5:9B:9E:5F:75:15:47:D6:C0:AD:75:FE:71


    ......
```

单独获取AKI

```
$ openssl x509 -in .fabric-ca-client/msp/signcerts/cert.pem -text -
noout | awk '/keyid/ {gsub (/ *keyid:|:/,"",$1);print tolower($0)}'
```

输出内容如下:

```
45b150b6cd8a8dc59b9e5f751547d6c0ad75fe71
```

单独获取序列号

```
$ openssl x509 -in .fabric-ca-client/msp/signcerts/cert.pem -serial -
noout | cut -d "=" -f 2
```

输出内容如下:

```
74488833701A01A0AD32296EC5AB5AFA3B9125A4
```