# OpenLDAP 完整部署(含高可用)

### 部署主从模式【TLS】

#### 5.1 证书创建脚本

脚本名为：makecert

#!/bin/bash  
country=CN  
state=Beijing  
locality=Beijing  
org=jops  
email=jops@163.com  
numbits=2048  
ca\_days=36500  
site\_days=36500  
client\_days=36500  
  
target=$1  
ca=  
site=  
client=  
arg\_check=  
if [ "$target" == "ca" ]; then  
if [ "$2" == "" ]; then  
echo argument error  
else  
ca=$2  
arg\_check=ok  
fi  
elif [ "$target" == "site" ]; then  
if [ "$3" == "" ]; then  
echo argument error  
else  
ca=$2  
site=$3  
arg\_check=ok  
fi  
elif [ "$target" == "client" ]; then  
if [ "$3" == "" ]; then  
echo argument error  
else  
ca=$2  
client=$3  
arg\_check=ok  
fi  
fi  
if [ "$arg\_check" != "ok" ]; then  
echo "[make CA]"  
echo " makecert ca CA-FILE-NAME"  
echo "[make site cert/key]"  
echo " makecert site CA-FILE-NAME SITE-DOMAIN-NAME"  
echo "[make client cert/key]"  
echo " makecert client CA-FILE-NAME CLIENT-NAME"  
exit  
fi  
  
if [ "$target" == "ca" ]; then  
echo "creating CA key..."  
openssl genrsa -out "${ca}.key" ${numbits}  
echo "creating CA csr..."  
openssl req -new -sha256 \  
-key "${ca}.key" \  
-out "${ca}.csr" \  
-days ${ca\_days} \  
-subj "/C=${country}/ST=${state}/L=${locality}/O=${org}/OU=${org}/CN=${ca}/emailAddress=${email}"  
  
echo "creating CA cert..."  
openssl x509 -req -sha256 -in "${ca}.csr" -signkey "${ca}.key" -out "${ca}.crt" -days 3650  
  
# echo "creating CA der..."  
#openssl x509 -in "${ca}.crt" -out "${ca}.der" -outform DER  
  
rm -f "${ca}.csr" > /dev/null 2>&1  
elif [ "$target" == "site" ]; then  
echo "creating server key..."  
openssl genrsa -out "${site}.key" ${numbits}  
echo "creating server csr..."  
openssl req -new -sha256 -key "${site}.key" -out "${site}.csr" -days 3650 \  
-subj "/C=${country}/ST=${state}/L=${locality}/O=${org}/OU=${org}/CN=\*.$site/emailAddress=${email}" \  
-config <(cat /etc/pki/tls/openssl.cnf <(printf "\n[SAN]\nsubjectAltName=DNS:\*.$site"))  
  
echo "authorityKeyIdentifier=keyid,issuer  
basicConstraints=CA:FALSE  
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment  
subjectAltName = @alt\_names  
  
[alt\_names]  
DNS.1 = $site" > "/tmp/openssl-site-ext"  
echo "sign server cert..."  
openssl x509 -sha256 \  
-req -in "${site}.csr" \  
-extfile "/tmp/openssl-site-ext" \  
-out "${site}.crt" \  
-CA "${ca}.crt" \  
-CAkey "${ca}.key" \  
-CAcreateserial \  
-days ${site\_days}  
  
rm -f "${site}.csr" > /dev/null 2>&1  
rm -f /tmp/openssl-site-ext  
rm -f .srl > /dev/null 2>&1  
rm -f \*.srl > /dev/null 2>&1  
elif [ "$target" == "client" ]; then  
echo "creating client key..."  
openssl genrsa -out "${client}.key" ${numbits}  
echo "creating client csr..."  
openssl req -new -sha256 -key "${client}.key" -out "${client}.csr" -days 3650 \  
-subj "/C=${country}/ST=${state}/L=${locality}/O=${org}/OU=${org}/CN=${client}/emailAddress=${email}"  
  
echo "extendedKeyUsage=clientAuth" > "/tmp/openssl-client-ext"  
  
echo "sign client cert.."  
openssl x509 -req -sha256 \  
-in "${client}.csr" \  
-extfile "/tmp/openssl-client-ext" \  
-out "${client}.crt" \  
-CA "${ca}.crt" \  
-CAkey "${ca}.key" \  
-CAcreateserial \  
-days ${client\_days}  
# echo "creating client der..."  
# openssl x509 -in "${client}.crt" -out "${client}.der" -outform DER  
  
rm -f "${client}.csr" > /dev/null 2>&1  
rm -f /tmp/openssl-client-ext  
rm -f .srl > /dev/null 2>&1  
rm -f \*.srl > /dev/null 2>&1  
fi

#### 5.2 生成根证书、域名证书、域名私钥

# 生成根证书、域名证书、域名私钥  
  
mkdir -p /data/openldap/{data,config,init,certs} #主  
mkdir -p /data/openldap/{data,config,init} #从  
cd /data/openldap/certs  
chmod +x ./makecert  
./makecert ca root # 创建 ca，会生成文件名root.crt文件  
./makecert site root jops.cn # 用 ca 颁发站点证书，生成key和crt文件  
  
# 同步证书  
scp -r /data/openldap/certs 192.168.91.70:/data/openldap/ #主  
scp -r /data/openldap/certs 192.168.91.71:/data/openldap/ #从

#### 5.3 更新系统的证书

将此证书拷贝【或软连】接至 /etc/ssl/certs/文件夹中 **注意：** 所有版本操作。

cp root.crt /etc/ssl/certs/root.crt

**注意：** 所有版本操作。

update-ca-trust

#### 5.4 部署openldap

创建自定义网络

docker network create --subnet=172.19.0.0/24 appldap

hosts绑定

cat >> /etc/hosts << EOF  
192.168.91.70 ldap1.jops.cn  
192.168.91.71 ldap2.jops.cn  
EOF

openldap-compose-1.yaml

version: "3.5"  
services:  
 openldap:  
 image: "osixia/openldap:latest"  
 container\_name: "ldap1"  
 hostname: ldap1.jops.cn  
 restart: always  
 environment:  
 LDAP\_TLS\_VERIFY\_CLIENT: never  
 LDAP\_ORGANISATION: "jops"  
 LDAP\_DOMAIN: "jops.cn"  
 LDAP\_ADMIN\_PASSWORD: "123456"  
 #定义证书书  
 LDAP\_TLS\_CRT\_FILENAME: "jops.cn.crt"   
 LDAP\_TLS\_KEY\_FILENAME: "jops.cn.key"  
 LDAP\_TLS\_CA\_CRT\_FILENAME: "root.crt"   
 #主从复制  
 LDAP\_REPLICATION\_HOSTS: "#PYTHON2BASH:['ldap://ldap1.jops.cn','ldap://ldap2.jops.cn']"   
 LDAP\_REPLICATION: "true"  
 #定义运行时的hosts配置   
 extra\_hosts:  
 - "ldap1.jops.cn:192.168.91.70"  
 - "ldap2.jops.cn:192.168.91.71"  
 volumes:  
 #- /etc/timezone:/etc/timezone   
 - /etc/localtime:/etc/localtime  
 - /data/openldap/data:/var/lib/ldap  
 - /data/openldap/config:/etc/ldap/slapd.d  
 - /data/openldap/init:/init  
 - /data/openldap/certs:/container/service/slapd/assets/certs  
 ports:  
 - '389:389'  
 - '636:636'  
 networks:   
 - appldap  
  
networks:  
 appldap:  
 driver: bridge

openldap-compose-2.yaml

version: "3.5"  
services:  
 openldap:  
 image: "osixia/openldap:latest"  
 container\_name: "ldap2"  
 hostname: ldap2.jops.cn  
 restart: always  
 environment:  
 LDAP\_TLS\_VERIFY\_CLIENT: never  
 LDAP\_ORGANISATION: "jops"  
 LDAP\_DOMAIN: "jops.cn"  
 LDAP\_ADMIN\_PASSWORD: "123456"  
 #定义证书书  
 LDAP\_TLS\_CRT\_FILENAME: "jops.cn.crt"   
 LDAP\_TLS\_KEY\_FILENAME: "jops.cn.key"  
 LDAP\_TLS\_CA\_CRT\_FILENAME: "root.crt"   
 #主从复制  
 LDAP\_REPLICATION\_HOSTS: "#PYTHON2BASH:['ldap://ldap1.jops.cn','ldap://ldap2.jops.cn']"   
 LDAP\_REPLICATION: "true"  
 #定义运行时的hosts配置   
 extra\_hosts:  
 - "ldap1.jops.cn:192.168.91.70"  
 - "ldap2.jops.cn:192.168.91.71"  
 volumes:  
 #- /etc/timezone:/etc/timezone   
 - /etc/localtime:/etc/localtime  
 - /data/openldap/data:/var/lib/ldap  
 - /data/openldap/config:/etc/ldap/slapd.d  
 - /data/openldap/init:/init  
 - /data/openldap/certs:/container/service/slapd/assets/certs   
 ports:  
 - '389:389'  
 - '636:636'  
 networks:   
 - appldap  
  
networks:  
 appldap:  
 driver: bridge

TLSVerifyClient never # 设置是否验证 client 的身份，其值可以是never/allow/try/demand，

#never 不需要验证 client 端的身份，Client 端只需要有 CA 证书就可以了

#allow Server会要求 client 提供证书，如果 client 端没有提供证书，会话会正常进行 #try Client端提供了证书，但是 Server 端有可能不能校验这个证书，这个证书会被忽略，会话正常进行

#demand Server端需要认证 client 端的身份，Client 端需要有自己的证书和私钥

#### 5.5 部署phpopenldap

phpopenldap-compose.yaml

version: "3.5"  
services:  
 php:  
 image: osixia/phpldapadmin:latest  
 restart: always  
 container\_name: phpopenldap  
 environment:  
 TZ: "Asia/Shanghai"  
 # PHPLDAPADMIN\_HTTPS: "false"  
 LAM\_SKIP\_PRECONFIGURE: "true"  
 LDAP\_DOMAIN: "jops.cn"  
 #PHPLDAPADMIN\_LDAP\_HOSTS: "#PYTHON2BASH:[{'ldap1.jops.cn': [{'server': [{'tls': True}]},{'login': [{'bind\_id': 'cn=admin,dc=jops,dc=cn'}]}]}, {'ldap2.jops.cn': [{'server': [{'tls': True}]},{'login': [{'bind\_id': 'cn=admin,dc=jops,dc=cn'}]}]}]"  
 PHPLDAPADMIN\_LDAP\_HOSTS: "#PYTHON2BASH:['ldap1.jops.cn','ldap2.jops.cn']"  
 PHPLDAPADMIN\_LDAP\_CLIENT\_TLS\_CA\_CRT\_FILENAME: "root.crt"  
 PHPLDAPADMIN\_LDAP\_CLIENT\_TLS\_CRT\_FILENAME: "jops.cn.crt"  
 PHPLDAPADMIN\_LDAP\_CLIENT\_TLS\_KEY\_FILENAME: "jops.cn.key"  
 extra\_hosts:   
 - "ldap1.jops.cn:192.168.91.70"   
 - "ldap2.jops.cn:192.168.91.71"  
 volumes:  
 #- /etc/timezone:/etc/timezone   
 - /etc/localtime:/etc/localtime  
 - /data/openldap/certs:/container/service/ldap-client/assets/certs/  
 ports:  
 # - 80:80  
 - 8443:443

浏览器访问 [https://192.168.91.70:8443](https://link.zhihu.com/?target=https%3A//10.0.0.103%3A8443)

管理员账号: cn=admin,dc=jops,dc=cn

密码: 123456

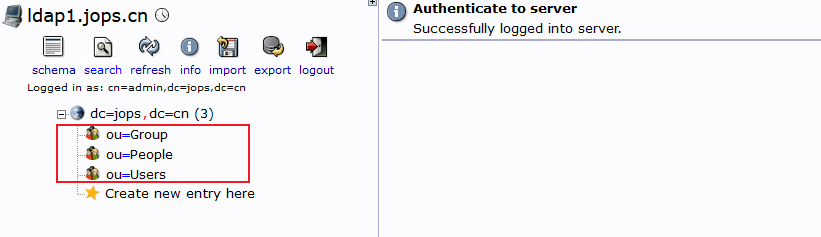
### 关于openLDAP的常用操作

#### 12.1 命令行操作

##### 12.1.1 创建OU

**创建顶级ou**

cd init  
cat > add\_ou.ldif << EOF  
dn: ou=Group,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: Group  
  
dn: ou=People,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: People  
  
dn: ou=Users,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: cn  
EOF  
  
# 执行命令  
ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" -f add\_ou.ldif  
docker exec -it ldap1 ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" -f /init/add\_ou.ldif



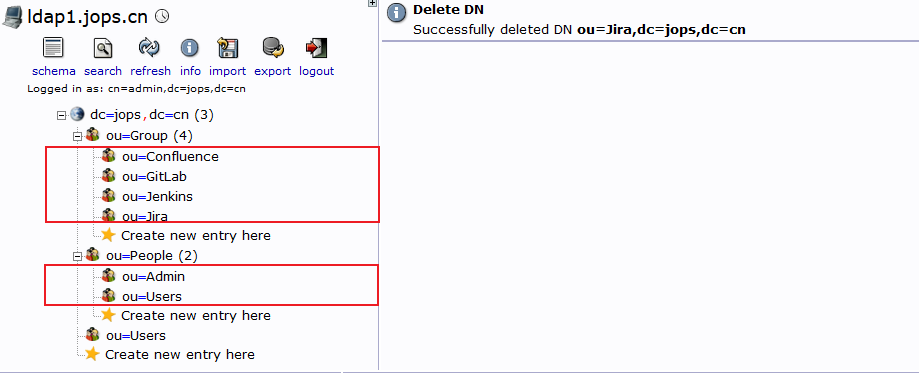
**创建自定义ou【顶级ou】**

cat > add\_custom\_ou.ldif << EOF  
dn: ou=Jenkins,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: Jenkins  
  
dn: ou=GitLab,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: GitLab  
  
dn: ou=Jira,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: Jira  
  
dn: ou=Confluence,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: Confluence  
  
dn: ou=Admin,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: Admin  
  
dn: ou=Users,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: Users  
EOF  
  
# 执行命令  
ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" -f add\_custom\_ou.ldif  
docker exec -it ldap1 ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" -f /init/add\_custom\_ou.ldif

**创建子ou**

分别在Group和People顶级OU下创建子ou

cat > add\_custom\_ou.ldif << EOF  
dn: ou=Jenkins,ou=Group,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: Jenkins  
  
dn: ou=GitLab,ou=Group,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: GitLab  
  
dn: ou=Jira,ou=Group,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: Jira  
  
dn: ou=Confluence,ou=Group,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: Confluence  
  
dn: ou=Admin,ou=People,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: Admin  
  
dn: ou=Users,ou=People,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: Users  
EOF  
  
# 执行命令  
ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" -f add\_custom\_ou.ldif  
docker exec -it ldap1 ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" -f /init/add\_custom\_ou.ldif

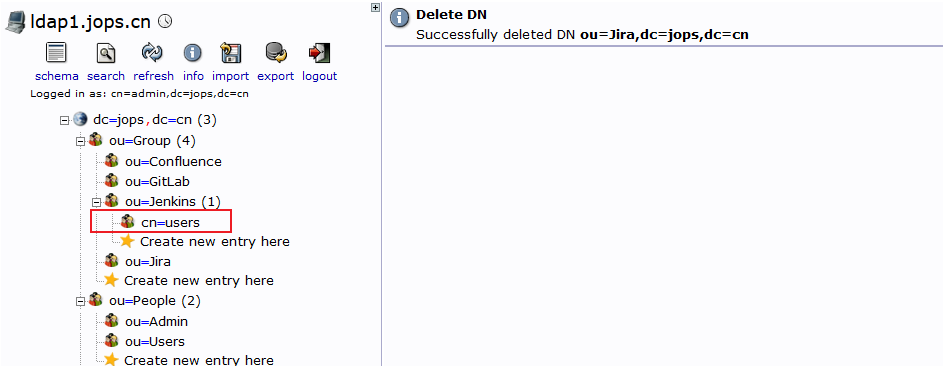


##### 12.1.2 创建组

**添加一个组, 在Jenkins的OU下**

cat > group\_jenkins.ldif << EOF  
dn: cn=users,ou=Jenkins,ou=Group,dc=jops,dc=cn  
objectClass: posixGroup  
objectClass: top  
cn: users  
gidNumber: 4002  
EOF  
  
# 执行命令  
ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" -f group\_jenkins.ldif  
docker exec -it ldap1 ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" -f /init/group\_jenkins.ldif

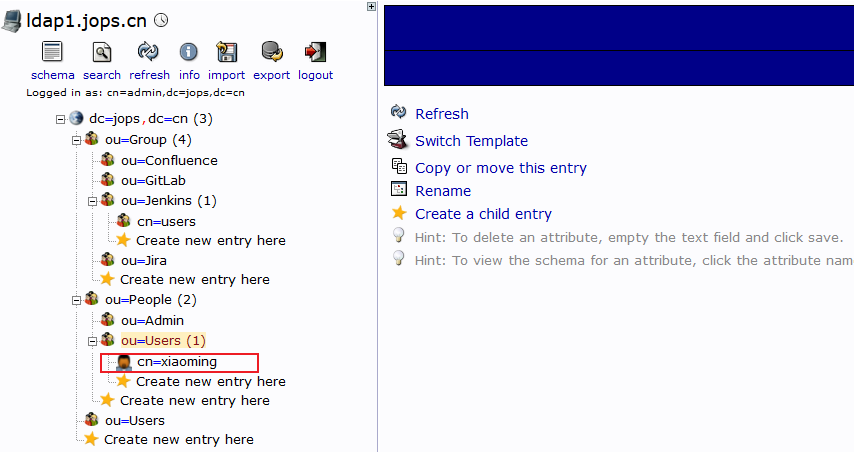
**注意dn的顺序,否则报错**



##### 12.1.3 创建账户

**添加用户小明, 位置在Users的OU下,并绑定到People的用户组Users中**

cat > xiaoming.ldif << EOF  
dn: cn=xiaoming,ou=Users,ou=People,dc=jops,dc=cn  
objectClass: top  
objectClass: inetOrgPerson  
objectClass: posixAccount  
objectClass: shadowAccount  
cn: xiaoming  
sn: xiao  
uid: xiaoming  
userPassword: 123456  
uidNumber: 44001  
gidNumber: 4002  
homeDirectory: /home/users/xiaoming  
mail: xiaoming@test.com.cn  
title: add user xiaoming  
EOF  
  
# 执行命令  
ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" -f xiaoming.ldif  
docker exec -it ldap1 ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" -f /init/xiaoming.ldif



##### 12.1.4 为用户设置密码

ldappasswd -x -h 127.0.0.1 -p 389 -D "cn=admin,dc=jops,dc=cn" -w "123456" "cn=xiaoming,ou=Users,ou=People,dc=jops,dc=cn"  
docker exec -it ldap1 ldappasswd -x -h 127.0.0.1 -p 389 -D "cn=admin,dc=jops,dc=cn" -w "123456" "cn=xiaoming,ou=Users,ou=People,dc=jops,dc=cn"  
  
  
New password: x3N1sfwe # 密码是随机的

##### 12.1.5 搜索

# 搜索全部  
ldapsearch -x -H ldap://127.0.0.1:389 -b "dc=jops,dc=cn" -D "cn=admin,dc=jops,dc=cn" -w "123456"  
  
# 正则匹配  
ldapsearch -x -H ldap://127.0.0.1:389 -b "dc=jops,dc=cn" -D "cn=admin,dc=jops,dc=cn" -w "123456" "cn=xiao\*"  
  
ldapsearch -x -H ldap://127.0.0.1:389 -b "dc=jops,dc=cn" -D "cn=admin,dc=jops,dc=cn" -w "123456" "ou=\*"

##### 12.1.6 删除

删除用户小明

ldapdelete -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" "cn=xiaoming,ou=Users,ou=People,dc=jops,dc=cn"  
---  
docker exec -it ldap1 ldapdelete -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" "cn=xiaoming,ou=Users,ou=People,dc=jops,dc=cn"

删除Jenkins的users组

ldapdelete -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" "cn=users,ou=Jenkins,ou=Group,dc=jops,dc=cn"  
---  
docker exec -it ldap1 ldapdelete -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" "cn=users,ou=Jenkins,ou=Group,dc=jops,dc=cn"

##### 12.1.7 modify

**添加用户小红**

cat > xiaohong.ldif << EOF  
dn: cn=xiaohong,ou=Users,ou=People,dc=jops,dc=cn  
changetype: add  
objectClass: top  
objectClass: inetOrgPerson  
objectClass: posixAccount  
objectClass: shadowAccount  
cn: xiaohong  
sn: xiao  
uid: xiaohong  
userPassword: 123456  
uidNumber: 44002  
gidNumber: 4002  
homeDirectory: /home/users/xiaohong  
mail: xiaohong@test.com.cn  
title: add user xiaohong  
EOF  
  
# 执行命令  
ldapmodify -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" -f xiaohong.ldif

**常用方法**

* 修改密码

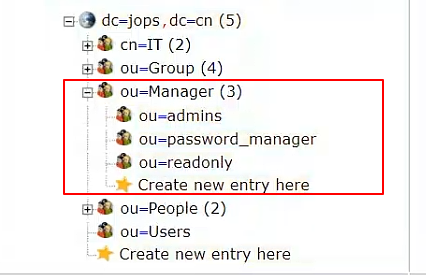
cat > changepwd.ldif << EOF  
dn: cn=xiaohong,ou=Users,ou=People,dc=jops,dc=cn  
changetype: modify  
replace: userPassword  
userPassword: xh1234  
EOF  
  
# 执行命令  
ldapmodify -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w "123456" -f changepwd.ldif

##### 12.1.8 LDAP用户权限配置

**创建管理ou**

创建顶级Manager,在Manager下创建admins(管理),readonly(只读),password\_manager(密码管理)等ou

cat > add\_manager\_ou.ldif << EOF  
dn: ou=Manager,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: Manager  
  
dn: ou=admins,ou=Manager,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: admins  
  
dn: ou=readonly,ou=Manager,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: readonly  
  
dn: ou=password\_manager,ou=Manager,dc=jops,dc=cn  
objectClass: organizationalUnit  
objectClass: top  
ou: password\_manager  
EOF  
  
ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w '123456' -f add\_manager\_ou.ldif  
---  
docker exec ldap1 ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w '123456' -f /init/add\_manager\_ou.ldif



**配置权限**

1 通过查看/etc/ldap/slapd.d/cn=config来确定olcDatabase配置文件, 比如我的是olcDatabase={1}mdb.ldif, 根据olcDatabase={1}mdb.ldif配置文件来确定dn位置, 我的是olcDatabase={1}mdb

2 根据olcDatabase={1}mdb.ldif配置文件, 重新编写访问控制

cat > new-acl.ldif << EOF   
dn: olcDatabase={1}mdb,cn=config  
changetype: modify  
delete: olcAccess  
-  
add: olcAccess  
olcAccess: {0}to \*   
 by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage   
 by \* break  
olcAccess: {1}to attrs=userPassword,shadowLastChange  
 by self write  
 by dn="cn=admin,dc=jops,dc=cn" write  
 by dn.children="ou=admins,ou=Manager,dc=jops,dc=cn" read   
 by dn.children="ou=password\_manager,ou=Manager,dc=jops,dc=cn" write  
 by anonymous auth  
 by \* none  
olcAccess: {2}to \*  
 by self read  
 by dn="cn=admin,dc=jops,dc=cn" write  
 by dn.children="ou=admins,ou=Manager,dc=jops,dc=cn" write  
 by dn.children="ou=password\_manager,ou=Manager,dc=jops,dc=cn" read  
 by dn.children="ou=readonly,ou=Manager,dc=jops,dc=cn" read  
 by \* none  
EOF  
--  
ldapmodify -Y EXTERNAL -H ldapi:/// -f new-acl.ldif  
docker exec ldap1 ldapmodify -Y EXTERNAL -H ldapi:/// -f /init/new-acl.ldif

olcAccess: {1}to attrs=userPassword,shadowLastChange # 对密码属性访问控制

olcAccess: {2}to \* # 对全局属性访问控制(密码除外)

**验证**

分别在几个管理ou下创建对应账户,然后访问ldap,验证权限,ldif配置文件示例

cat > add\_readOnly.ldif << EOF  
dn: cn=readuser,ou=readonly,ou=Manager,dc=jops,dc=cn  
objectClass: simpleSecurityObject  
objectClass: organizationalRole  
description: LDAP read only user  
cn: readuser  
userPassword: 123456  
EOF  
  
ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w '123456' -f add\_readOnly.ldif  
docker exec ldap1 ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w '123456' -f /init/add\_readOnly.ldif  
  
# 登录名: cn=readuser,ou=readonly,ou=Manager,dc=jops,dc=cn 密码: 123456(只能查询，且无法查看密码，即对全局属性访问控制(密码除外))  
  
cat > add\_myadmin.ldif << EOF  
dn: cn=myadmin,ou=admins,ou=Manager,dc=jops,dc=cn  
objectClass: simpleSecurityObject  
objectClass: organizationalRole  
description: LDAP read only user  
cn: myadmin  
userPassword: 123456  
EOF  
  
ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w '123456' -f add\_myadmin.ldif  
docker exec ldap1 ldapadd -x -H ldap://127.0.0.1:389 -D "cn=admin,dc=jops,dc=cn" -w '123456' -f /init/add\_myadmin.ldif  
# 登录名: cn=myadmin,ou=admins,ou=Manager,dc=jops,dc=cn 密码: 123456  
# （拥有管理员权限）

### openldap数据初始化

##### 1、 创建组

cat > "/data/openldap/init/base.ldif" << EOF  
dn: ou=users,dc=jops,dc=cn  
objectClass: organizationalUnit  
ou: users  
  
dn: ou=groups,dc=jops,dc=cn  
objectClass: organizationalUnit  
ou: groups  
  
# 管理员组  
dn: ou=g-admin,ou=groups,dc=jops,dc=cn  
changetype: add  
cn: g-admin  
objectClass: groupOfNames  
objectClass: top  
member: cn=radmin,ou=users,dc=jops,dc=cn  
  
#创建unix组  
dn: cn=unix,ou=groups,dc=jops,dc=cn  
cn: unix  
gidnumber: 10000  
objectclass: posixGroup  
EOF  
  
docker exec ldap1 sh -c "ldapadd -x -H ldap://127.0.0.1:389 -D cn=admin,dc=jops,dc=cn -w 123456 -f /init/base.ldif"  
  
#如果出现如下提示  
ldap\_add: Already exists (68)  
adding new entry "ou=users,dc=jops,dc=cn"  
  
则需先删除users组

##### 2 、创建用户

cat > "/data/openldap/init/adduser.ldif" << EOF  
# 密码 readonly2020  
dn: cn=readonly,dc=jops,dc=cn  
changetype: add  
cn: readonly  
objectClass: inetOrgPerson  
objectClass: top  
sn: readonly  
telephoneNumber: 13000000001  
mail: readonly@jops.cn  
userPassword: readonly2020  
#userPassword: {MD5}DJGL63b7oYOncsZSsb/e7A==  
  
# 密码 test2020  
dn: cn=test,ou=users,dc=jops,dc=cn  
changetype: add  
cn: test  
objectClass: inetOrgPerson  
objectClass: top  
sn: test  
telephoneNumber: 13000000002  
mail: test@jops.cn  
userPassword: {MD5}mLAb4tluXq/vZtslgQfK9A==  
  
# 密码 radmin2020  
dn: cn=radmin,ou=users,dc=jops,dc=cn  
changetype: add  
cn: radmin  
objectClass: inetOrgPerson  
objectClass: top  
sn: radmin  
telephoneNumber: 13000000003  
mail: radmin@jops.cn  
userPassword: {MD5}Wkr/lT7eoTyB27LjGG5BTw==  
  
# 密码 admin2020  
dn: cn=admin,ou=users,dc=jops,dc=cn  
changetype: add  
cn: admin  
objectclass: inetOrgPerson  
objectclass: top  
objectclass: posixAccount  
sn: admin  
userpassword: {MD5}REHl1ws2V5APpX5m20B+Cw==  
#unix用户配置  
gidnumber: 10000  
homedirectory: /home/  
loginshell: /bin/bash  
uid: admin  
uidnumber: 10000  
EOF  
  
docker exec ldap1 sh -c "ldapadd -x -H ldap://127.0.0.1:389 -D cn=admin,dc=jops,dc=cn -w 123456 -f /init/adduser.ldif"

##### 3 、禁止匿名访问

cat > "/data/openldap/init/disable\_anon.ldif" << EOF  
dn: cn=config  
changetype: modify  
add: olcDisallows  
olcDisallows: bind\_anon  
  
dn: cn=config  
changetype: modify  
add: olcRequires  
olcRequires: authc  
  
dn: olcDatabase={-1}frontend,cn=config  
changetype: modify  
add: olcRequires  
olcRequires: authc  
EOF  
  
docker exec ldap1 sh -c "ldapadd -Y EXTERNAL -H ldapi:/// -f /init/disable\_anon.ldif"

##### 4、密码修改策略

cat > "/data/openldap/init/acl.ldif" << EOF  
dn: olcDatabase={1}mdb,cn=config  
changetype: modify  
# 只有自己可以修改密码，不允许匿名访问,允许超级管理员admin修改,允许g-admin组修改  
replace: olcAccess  
olcAccess: {0}to attrs=userPassword   
 by self write   
 by anonymous auth   
 by dn="cn=admin,dc=jops,dc=cn" write  
 by group.exact="cn=g-admin,ou=groups,dc=jops,dc=cn" write   
 by \* none  
# 自己可以修改自己的信息，g-admin组可以修改任何信息，readonly账号可以查看信息  
olcAccess: {1}to \*   
 by self write   
 by dn.exact="cn=readonly,dc=jops,dc=cn" read  
 by group.exact="cn=g-admin,ou=groups,dc=jops,dc=cn" write   
 by \* none  
EOF  
  
docker exec ldap1 sh -c "ldapadd -Y EXTERNAL -H ldapi:/// -f /init/acl.ldif"

##### 5、 ppolicy模块

密码复杂度

**查看是否已加载任何模块**

docker exec ldap1 slapcat -n 0 | grep olcModuleLoad

#配置/加载module模块  
cat > "/data/openldap/init/module.ldif" << EOF  
dn: cn=module,cn=config  
objectClass: olcModuleList  
cn: module  
olcModulepath: /usr/lib/ldap  
olcModuleload: accesslog.la  
olcModuleload: auditlog.la  
olcModuleLoad: ppolicy.la  
#olcModuleload: memberof.la  
EOF  
  
docker exec ldap1 sh -c "ldapadd -Y EXTERNAL -H ldapi:/// -f /init/module.ldif"  
  
#配置默认配置  
cat > "/data/openldap/init/ppolicy\_db.ldif" << EOF  
dn: olcOverlay=ppolicy,olcDatabase={1}mdb,cn=config  
changetype: add  
objectClass: olcConfig  
objectClass: olcOverlayConfig  
objectClass: olcPPolicyConfig  
olcOverlay: ppolicy  
olcPPolicyDefault: cn=default,ou=Policies,dc=jops,dc=cn  
olcPPolicyHashCleartext: TRUE  
olcPPolicyUseLockout: TRUE  
EOF  
  
docker exec ldap1 sh -c "ldapadd -Y EXTERNAL -H ldapi:/// -f /init/ppolicy\_db.ldif"  
  
#创建组  
cat > "/data/openldap/init/ppolicy\_group.ldif" << EOF  
dn: ou=Policies,dc=jops,dc=cn  
objectClass: top  
objectClass: organizationalUnit  
ou: Policies  
EOF  
  
docker exec ldap1 sh -c "ldapadd -x -H ldap://127.0.0.1:389 -D cn=admin,dc=jops,dc=cn -w 123456 -f /init/ppolicy\_group.ldif"  
  
#创建默认密码策略  
cat > "/data/openldap/init/ppolicy\_rulues.ldif" << EOF  
dn: cn=default,ou=Policies,dc=jops,dc=cn  
cn: default  
objectClass: top  
objectClass: device  
objectClass: pwdPolicy  
objectClass: pwdPolicyChecker  
pwdAttribute: 2.5.4.35  
#pwdAttribute: userPassword  
pwdInHistory: 3  
pwdMinLength: 8  
pwdMaxFailure: 3  
pwdFailureCountInterval: 1800  
pwdCheckQuality: 2  
pwdMustChange: TRUE  
pwdGraceAuthNLimit: 0  
pwdMaxAge: 8640000  
pwdExpireWarning: 1209600  
pwdLockoutDuration: 900  
pwdLockout: TRUE  
EOF  
  
docker exec ldap1 sh -c "ldapadd -x -H ldap://127.0.0.1:389 -D cn=admin,dc=jops,dc=cn -w 123456 -f /init/ppolicy\_rulues.ldif"

密码策略属性详解

pwdCheckQuality: 2 #可以使用3个值 0:不检查 1:检查，但是密码不符合也接受 2:检查,密码不符合不接受

pwdAllowUserChange：允许用户修改其密码。

pwdAttribute：pwdPolicy 对象的一个属性，用于识别用户密码。

pwdExpireWarning：密码过期前告警天数。

pwdFailureCountInterval：密码失败后恢复时间。

pwdGraceAuthNLimit：密码过期后不能登入的天数，0 代表禁止登录。

pwdInHistory：开启密码历史记录，用户保证不能和之前设置的密码相同。

pwdLockout：超过定义次数，账号被锁定。

pwdLockoutDuration：密码连接输入错误次数后，账号锁定时间。

pwdMaxAge：密码有效期，到期需要强制修改密码。

pwdMaxFaliure：密码最大失效次数，超过后账号被锁定。

pwdMinAge：密码有效期。

pwdMinLength：密码修改密码时最短的密码长度。

pwdMustChange：用户登录系统后提示修改密码。

pwdSafeModify：是否允许用户修改密码，与 pwdMustChange 共同使用。

pwdLockoutDuration：账号锁定后，不能自动解锁，此时需要管理员干涉。

##### 6、 pqchecker模块

cat > "/data/openldap/init/pqchecker.ldif" << EOF  
dn: cn=default,ou=Policies,dc=jops,dc=cn  
changetype: modify  
add: pwdcheckmodule  
pwdCheckModule: pqchecker.so  
#-  
#add: objectClass  
#objectclass: pwdPolicyChecker  
EOF  
  
docker exec ldap1 sh -c "ldapadd -x -H ldap://127.0.0.1:389 -D cn=admin,dc=jops,dc=cn -w 123456 -f /init/pqchecker.ldif"

**定义用户登录需改密码**

cat << EOF | ldapadd -x -D "cn=admin,dc=jops,dc=cn" -w 123456 -H ldap://127.0.0.1  
dn: uid=chiu,ou=users,dc=jops,dc=cn  
changetype: modify  
replace: pwdReset  
pwdReset: TRUE  
EOF

##### 7、 审核模块audit

cat > "/data/openldap/init/audit.ldif" << EOF  
dn: cn=module{0},cn=config  
changetype: modify  
add: olcModuleLoad  
olcModuleLoad: auditlog  
  
dn: olcOverlay=auditlog,olcDatabase={1}mdb,cn=config  
changetype: add  
objectClass: olcOverlayConfig  
objectClass: olcAuditLogConfig  
olcAuditlogFile: /var/log/slapd/auditlog.log  
  
dn: olcDatabase={1}mdb,cn=config  
changetype: modify  
add: olcAccess  
olcAccess: {0}to attrs=userPassword,shadowLastChange   
 by self write   
 by dn="cn=admin,dc=jops,dc=cn" write   
 by anonymous auth by \* read  
olcAccess: {1}to \*   
 by self write   
 by dn="cn=admin,dc=jops,dc=cn" write  
 by \* read  
EOF  
  
docker exec ldap1 sh -c "ldapadd -Y EXTERNAL -H ldapi:/// -f /init/audit.ldif"

容器审核日志路径：/var/log/slapd/auditlog.log

根据情况做映射

##### 8、sudo模块

cat > "/data/openldap/init/sudo-overlay.ldif" << EOF  
dn: cn=sudo,cn=schema,cn=config  
objectClass: olcSchemaConfig  
cn: sudo  
olcAttributeTypes: {0}( 1.3.6.1.4.1.15953.9.1.1 NAME 'sudoUser' DESC 'User(s  
 ) who may run sudo' EQUALITY caseExactIA5Match SUBSTR caseExactIA5Substrin  
 gsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )  
olcAttributeTypes: {1}( 1.3.6.1.4.1.15953.9.1.2 NAME 'sudoHost' DESC 'Host(s  
 ) who may run sudo' EQUALITY caseExactIA5Match SUBSTR caseExactIA5Substring  
 sMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )  
olcAttributeTypes: {2}( 1.3.6.1.4.1.15953.9.1.3 NAME 'sudoCommand' DESC 'Com  
 mand(s) to be executed by sudo' EQUALITY caseExactIA5Match SYNTAX 1.3.6.1.4  
 .1.1466.115.121.1.26 )  
olcAttributeTypes: {3}( 1.3.6.1.4.1.15953.9.1.4 NAME 'sudoRunAs' DESC 'User(  
 s) impersonated by sudo (deprecated)' EQUALITY caseExactIA5Match SYNTAX 1.3  
 .6.1.4.1.1466.115.121.1.26 )  
olcAttributeTypes: {4}( 1.3.6.1.4.1.15953.9.1.5 NAME 'sudoOption' DESC 'Opti  
 ons(s) followed by sudo' EQUALITY caseExactIA5Match SYNTAX 1.3.6.1.4.1.1466  
 .115.121.1.26 )  
olcAttributeTypes: {5}( 1.3.6.1.4.1.15953.9.1.6 NAME 'sudoRunAsUser' DESC 'U  
 ser(s) impersonated by sudo' EQUALITY caseExactIA5Match SYNTAX 1.3.6.1.4.1.  
 1466.115.121.1.26 )  
olcAttributeTypes: {6}( 1.3.6.1.4.1.15953.9.1.7 NAME 'sudoRunAsGroup' DESC '  
 Group(s) impersonated by sudo' EQUALITY caseExactIA5Match SYNTAX 1.3.6.1.4.  
 1.1466.115.121.1.26 )  
olcAttributeTypes: {7}( 1.3.6.1.4.1.15953.9.1.8 NAME 'sudoNotBefore' DESC 'S  
 tart of time interval for which the entry is valid' EQUALITY generalizedTim  
 eMatch ORDERING generalizedTimeOrderingMatch SYNTAX 1.3.6.1.4.1.1466.115.12  
 1.1.24 )  
olcAttributeTypes: {8}( 1.3.6.1.4.1.15953.9.1.9 NAME 'sudoNotAfter' DESC 'En  
 d of time interval for which the entry is valid' EQUALITY generalizedTimeMa  
 tch ORDERING generalizedTimeOrderingMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1  
 .24 )  
olcAttributeTypes: {9}( 1.3.6.1.4.1.15953.9.1.10 NAME 'sudoOrder' DESC 'an i  
 nteger to order the sudoRole entries' EQUALITY integerMatch ORDERING intege  
 rOrderingMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )  
olcObjectClasses: {0}( 1.3.6.1.4.1.15953.9.2.1 NAME 'sudoRole' DESC 'Sudoer   
 Entries' SUP top STRUCTURAL MUST cn MAY ( sudoUser $ sudoHost $ sudoCommand  
 $ sudoRunAs $ sudoRunAsUser $ sudoRunAsGroup $ sudoOption $ sudoOrder $ su  
 doNotBefore $ sudoNotAfter $ description ) )  
structuralObjectClass: olcSchemaConfig  
EOF  
  
  
cat > "/data/openldap/init/sudo.ldif" << EOF  
dn: ou=SUDOers,dc=jops,dc=cn  
ou: SUDOers  
objectClass: top  
objectClass: organizationalUnit  
  
dn: cn=defaults,ou=SUDOers,dc=jops,dc=cn  
objectClass: sudoRole  
cn: defaults  
sudoOption: requiretty  
sudoOption: !visiblepw  
sudoOption: always\_set\_home  
sudoOption: env\_reset  
sudoOption: env\_keep = "COLORS DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR LS\_COLORS"  
sudoOption: env\_keep += "MAIL PS1 PS2 QTDIR USERNAME LANG LC\_ADDRESS LC\_CTYPE"  
sudoOption: env\_keep += "LC\_COLLATE LC\_IDENTIFICATION LC\_MEASUREMENT LC\_MESSAGES"  
sudoOption: env\_keep += "LC\_MONETARY LC\_NAME LC\_NUMERIC LC\_PAPER LC\_TELEPHONE"  
sudoOption: env\_keep += "LC\_TIME LC\_ALL LANGUAGE LINGUAS \_XKB\_CHARSET XAUTHORITY"  
sudoOption: secure\_path = /sbin:/bin:/usr/sbin:/usr/bin  
#sudoOption: logfile = /var/log/sudo  
EOF  
  
cat > "/data/openldap/init/sudouser.ldif" << EOF  
dn: cn=sudo\_ops\_role,ou=SUDOers,dc=jops,dc=cn  
objectClass: sudoRole  
cn: sudo\_ops\_role  
sudoOption: !authenticate  
sudoRunAsUser: root  
sudoCommand: ALL  
sudoHost: ALL  
sudoUser: 800001  
EOF  
  
docker exec ldap1 sh -c "ldapadd -Y EXTERNAL -H ldapi:/// -f /init/sudo-overlay.ldif"  
docker exec ldap1 sh -c "ldapadd -x -H ldap://127.0.0.1:389 -D cn=admin,dc=jops,dc=cn -w 123456 -f /init/sudo.ldif"  
docker exec ldap1 sh -c "ldapadd -x -H ldap://127.0.0.1:389 -D cn=admin,dc=jops,dc=cn -w 123456 -f /init/sudouser.ldif"

**Sudo 常见的属性有以下几个**

sudoCommand：可执行的二进制命令，如 useradd、userdel、mount、umount 等。

sudoHost：可在哪些机器上执行 sudoCommand 定义的 BASH 命令。

sudoNotAfter：起始时间 sudo 规则匹配。

sudoNotBefore：结束时间 sudo 规则匹配。

sudoOption：定义超过自身权限及切换至其他用户时，是否需要输入当前用户密码。

sudoOrder：sudo 规则执行顺序，其属性是一个整数。

sudoRole：定义的规则。

sudoRunAs：可切换到定义的用户身份下执行 BASH 命令。

sudoRunAsGroup：可切换到定义所属组并具有该组的权限。

sudoRunAsUser：定义可切换至哪些用户下执行命令。

sudoUser：限制哪些用户或哪些组内的成员具有 sudo 相关规则。

**添加 sudo ldif**

首先需要对openldap 添加sudo 管理支持，这一功能是由开源软件sudo提供的，首先通过schema文件添加sudo 的objectClass。 schema文件通常可以通过sudo 的官方源码获得，在Centos 系统中则或被存放在目录“/usr/share/doc/sudo-1.8.19p2"中。

# 首先将schema格式文件转换为 ldif 格式文件  
mkdir /tmp/ldif  
cd /tmp/ldif  
cp /usr/share/doc/sudo-1.8.23/schema.OpenLDAP .  
cat > ./schema\_conv.conf << EOL  
include /tmp/ldif/schema.OpenLDAP  
EOL  
  
slaptest -f ./schema\_conv.conf -F /tmp/ldif/  
cat cn\=config/cn\=schema/cn\=\{0\}schema.ldif > sudoers.ldif  
sed -i "s/dn: cn={0}schema/dn: cn=sudo,cn=schema,cn=config/g" sudoers.ldif  
sed -i "s/cn: {0}schema/cn: sudo/g" sudoers.ldif  
end=`grep -n "structuralObjectClass" sudoers.ldif |awk -F":" '{print $1}'`  
sed -n -i "3,${end}p" sudoers.ldif   
  
# 导入ldif 文件  
ldapadd -Q -Y EXTERNAL -H ldapi:/// -f sudoers.ldif

以下是容器内操作

cp /usr/share/doc/sudo-1.8.23/schema.OpenLDAP /data/openldap/init/  
docker exec -it ldap1 bash  
cd init  
cat > ./schema\_conv.conf << EOL  
include ./schema.OpenLDAP  
EOL  
  
slaptest -f ./schema\_conv.conf -F /tmp/  
cat /tmp/cn\=config/cn\=schema/cn\=\{0\}schema.ldif > sudoers.ldif  
sed -i "s/dn: cn={0}schema/dn: cn=sudo,cn=schema,cn=config/g" sudoers.ldif  
sed -i "s/cn: {0}schema/cn: sudo/g" sudoers.ldif  
end=`grep -n "structuralObjectClass" sudoers.ldif |awk -F":" '{print $1}'`  
sed -n -i "3,${end}p" sudoers.ldif   
# 导入ldif 文件  
ldapadd -Q -Y EXTERNAL -H ldapi:/// -f sudoers.ldif

[OpenLdap添加sudo支持 # 添加 sudo ldif](https://www.nginxbar.com/pages/654431/#%E6%B7%BB%E5%8A%A0-sudo-ldif)

##### 9、memberof模块(不用安装)

默认情况下OpenLDAP的用户组属性是Posixgroup，Posixgroup用户组和用户没有实际的对应关系。如果需要把Posixgroup和user关联起来则需要将用户添加到对应的组中。 通过如上配置可以满足大部分业务场景，但是如果需要通过用户组来查找用户的话，Posixgroup用户组属性，是无法满足要求的。此时需要使用OpenLDAP的groupOfUniqueNames用户组属性

<https://blog.csdn.net/qq_23191379/article/details/106867730>

cat > "/data/openldap/init/memberof\_conf.ldif" << EOF  
#开启memberof支持,记得删除注释  
dn: cn=module{2},cn=config  
cn: modulle{2}  
objectClass: olcModuleList  
objectclass: top  
olcModuleload: memberof.la  
olcModulePath: /usr/lib/ldap  
  
#新增用户支持memberof配置，记得删除注释  
dn: olcOverlay={0}memberof,olcDatabase={1}mdb,cn=config  
objectClass: olcConfig  
objectClass: olcMemberOf  
objectClass: olcOverlayConfig  
objectClass: top  
olcOverlay: memberof  
olcMemberOfDangling: ignore  
olcMemberOfRefInt: TRUE  
olcMemberOfGroupOC: groupOfUniqueNames  
olcMemberOfMemberAD: uniqueMember  
olcMemberOfMemberOfAD: memberOf  
EOF  
  
  
cat > "/data/openldap/init/refint1.ldif" << EOF  
dn: cn=module{2},cn=config  
changetype: modify  
add: olcmoduleload  
olcmoduleload: refint.la  
EOF  
  
cat > "/data/openldap/init/refint2.ldif" << EOF  
dn: olcOverlay=refint,olcDatabase={1}mdb,cn=config  
objectClass: olcConfig  
objectClass: olcOverlayConfig  
objectClass: olcRefintConfig  
objectClass: top  
olcOverlay: refint  
olcRefintAttribute: memberof uniqueMember manager owner  
EOF  
  
docker exec ldap1 sh -c "ldapadd -Y EXTERNAL -H ldapi:/// -f /init/memberof\_conf.ldif"  
docker exec ldap1 sh -c "ldapadd -Y EXTERNAL -H ldapi:/// -f /init/refint1.ldif"  
docker exec ldap1 sh -c "ldapadd -Y EXTERNAL -H ldapi:/// -f /init/refint2.ldif"

查看当前dn下包含cn=config配置列表会显示出已生效的memberof和refint配置

docker exec ldap1 ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b cn=config dn | grep olcOverlay

**创建用户测试**

1.创建一个测试用户ops。ops\_user.ldif文件内容如下

cat > /data/openldap/init/ops\_user.ldif << EOF  
dn: uid=ops,ou=users,dc=jops,dc=cn  
uid: ops  
cn: ops  
objectClass: account  
objectClass: posixAccount  
objectClass: top  
objectClass: shadowAccount  
userPassword: 123456  
shadowLastChange: 19338  
shadowMin: 0  
shadowMax: 30  
shadowWarning: 7  
loginShell: /bin/bash  
uidNumber: 10001  
gidNumber: 10001  
homeDirectory: /home/ops  
EOF  
---  
docker exec ldap1 ldapadd -D "cn=admin,dc=jops,dc=cn" -w 123456 -x -f /init/ops\_user.ldif



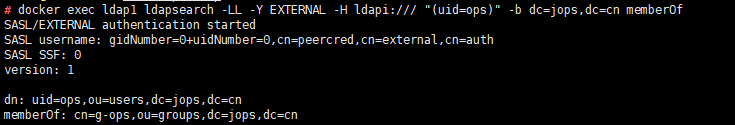
2 创建一个新的groupOfUniqueNames用户组，并把cdsw\_a用户添加到该组

cat > /data/openldap/init/ops\_group.ldif << EOF  
dn: cn=g-ops,ou=groups,dc=jops,dc=cn  
objectClass: groupOfUniqueNames  
cn: g-ops  
uniqueMember: uid=ops,ou=users,dc=jops,dc=cn  
EOF  
---  
docker exec ldap1 ldapadd -D "cn=admin,dc=jops,dc=cn" -w 123456 -x -f /init/ops\_group.ldif



3 通过命令查看用户所属组，命令如下

docker exec ldap1 ldapsearch -LL -Y EXTERNAL -H ldapi:/// "(uid=ops)" -b dc=jops,dc=cn memberOf



其他

#增加用户首次登陆更改密码  
cat > "/data/openldap/init/ppolicy\_changepasswd\_at\_first\_time.ldif" << EOF  
dn: uid=linux\_user1,ou=People,dc=jops,dc=cn  
changetype: modify  
replace: pwdReset  
pwdReset: TRUE  
EOF  
  
#删除该用户登陆更改密码属性  
cat > "/data/openldap/init/ppolicy\_delete\_changepassword.ldif" << EOF  
changetype: modify  
delete: pwdReset  
EOF  
  
  
# 对于服务帐户，不使帐户过期更安全。  
cat > "/data/openldap/init/ppolicy\_1.ldif" << EOF  
dn: cn=servicesaccounts, ou=Policies,dc=jops,dc=cn  
cn: servicesaccounts  
objectClass: top  
objectClass: device  
objectClass: pwdPolicy  
pwdAllowUserChange: TRUE  
pwdAttribute: userPassword  
pwdExpireWarning: 0  
pwdFailureCountInterval: 0  
pwdGraceAuthNLimit: 5  
pwdLockout: FALSE  
pwdLockoutDuration: 0  
pwdInHistory: 0  
pwdMaxAge: 0  
pwdMaxFailure: 0  
pwdMinAge: 0  
pwdMinLength: 15  
pwdMustChange: FALSE  
pwdSafeModify: FALSE  
EOF  
  
  
docker exec ldap sh -c "ldapadd -Y EXTERNAL -H ldapi:/// -f /init/ppolicy\_1.ldif"

[OpenLDAP启用MemberOf](https://wandouduoduo.netlify.app/articles/53f92c3c.html)

##### 10 备份

**10.1、slapcat备份**

mkdir -p /data/openldap/init/backup  
cat >/data/openldap/init/backup/backup.sh <<EOF  
#!/bin/bash  
echo '准备开始备份ldap'  
DATEFORMATTYPE=\$(date +%Y-%m-%d)  
echo \$DATEFORMATTYPE  
  
LDAPSCAT=/usr/sbin/slapcat  
#备份目录  
BACKDIR=/init/backup  
  
docker exec -it ldap1 slapcat -l \${BACKDIR}/backup\_\${DATEFORMATTYPE}.ldif  
EOF  
  
chmod +x /data/openldap/init/backup/backup.sh  
sh /data/openldap/init/backup/backup.sh

删除所有数据的操作

docker exec -it ldap1 ldapdelete -x -D "cn=admin,dc=jops,dc=cn" -w 123456 -r "dc=jops,dc=cn"

恢复数据

slapadd -l /root/openldap.ldif

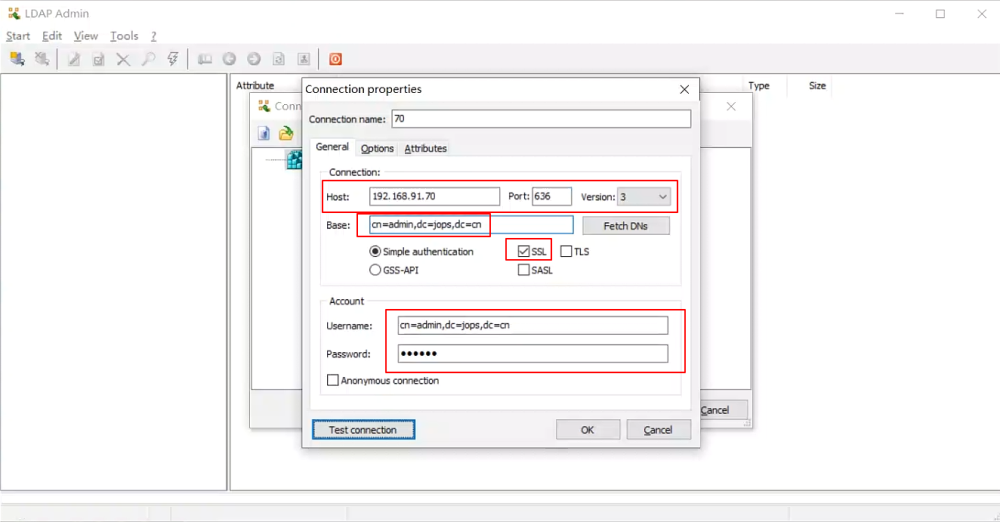
**10.2、整目录备份**

cd /data/openldap  
tar zcvf backup.tar.gz data config init certs

**10.3、phpopenldap进行备份**



#### 验证ssl



### 参考文档

[统一身份认证系统OpenLDAP 完整部署(含高可用)](https://zhuanlan.zhihu.com/p/532447126)

<https://blog.csdn.net/VSON_LIU/article/details/125873183>

[如何在OpenLDAP启用MemberOf](https://cloud.tencent.com/developer/article/1349459)