# 使用 ssh 隧道技术安全、高效地连接云端数据库

## 一、服务端安装和配置

### 1.安装 openssh(必须)

开启 powershell

以下命令安装 ssh

Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0

Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0

安装后，启动服务并设为开机自启：

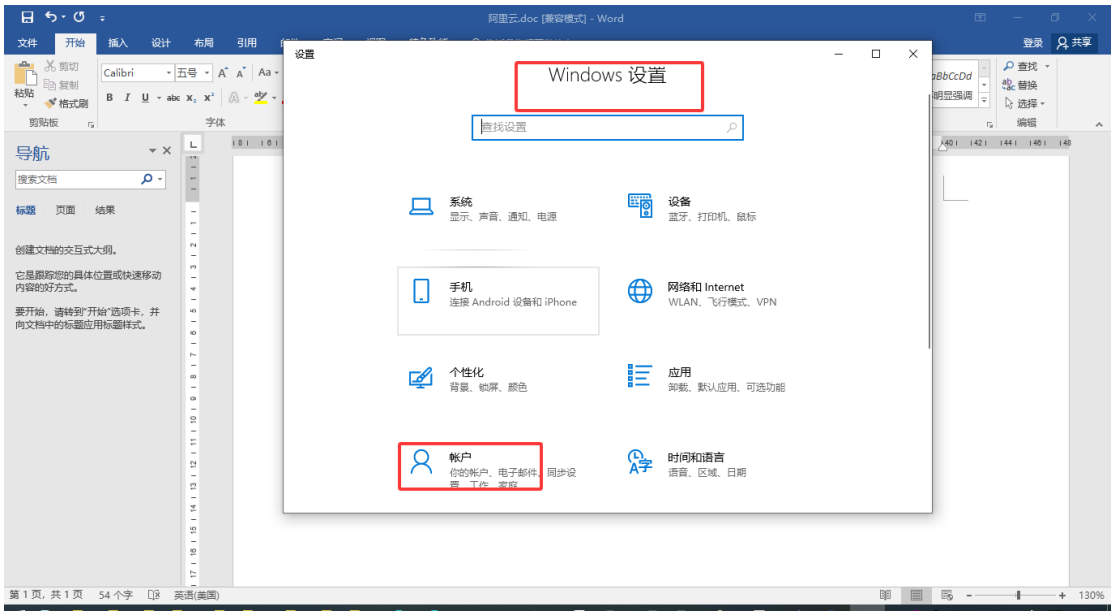
Start-Service sshd

Set-Service -Name sshd -StartupType Automatic

如果 windows server 版本比较低，没有自带 openssh，可从以下 url 下载 openssh 安装文件并安装。

http://8.159.129.248/download/OpenSSH-Win64-v9.8.3.0.msi

### 2.创建用户。为安全起见，不要用 administrator 作为 ssh 登录用户（可选）




建立一个低权限的用户，设置密码。**注意：密码要有一定复杂度，防止被攻击者轻松猜到。**

在 cmd 或 powershell 下测试：

ssh 新建用户名/密码@127.0.0.1

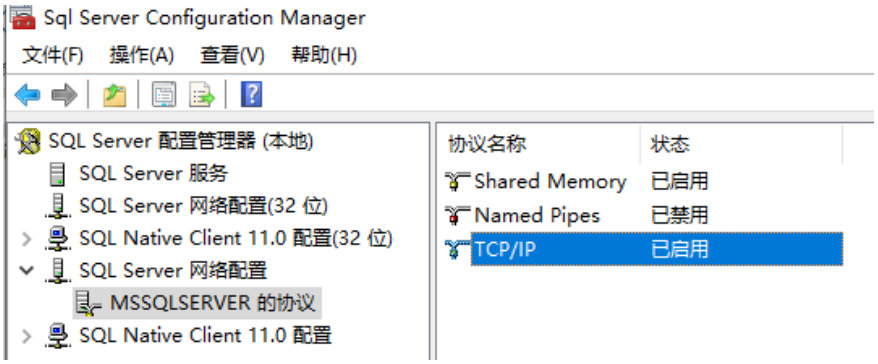
不出意外，应该出现登录提示，或者登录成功。

### 3.安装 ms sqlserver（可选）

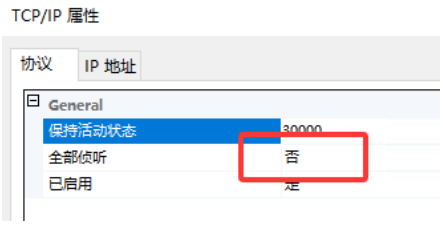
正常安装并配置 sqlserver 服务。

默认情况下，sqlserver 服务会监听所有 IP 和端口。为了安全起见，不要把 sqlserver 服务暴露在公网上，也就是不要在公网上监听 sqlserver 服务。

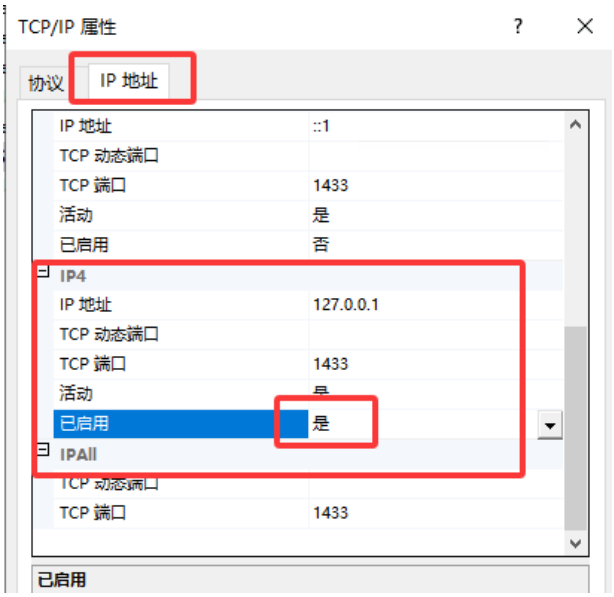
启动 sqlserver 配置管理工具。



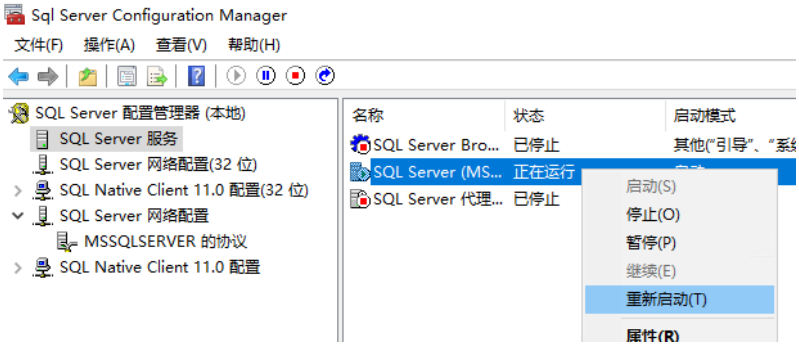

启动配置管理工具，如图所示。鼠标双击 TCP/IP。



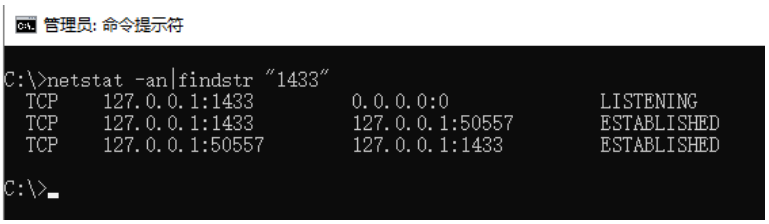

此处如果原值是“是”，请改为“否”。




切换到“IP 地址”，找到 127.0.0.1 这一 IP，把已启用改为“是”。然后确定。




重新启动服务。




在 cmd 或 powershell 下输入：
Netstat –an|findstr “1433”
可以看到，sqlserver 只监听在 127.0.0.1 上，公网上没有监听。

**二、客户端建立 ssh 隧道**

1.标准方法

```
uo_json args; args = create uo_json
args.set("ssh_host", "x.x.x.x"); //你的云服务器 IP
args.set("ssh_port", 22); //ssh 端口，如果没修改默认就是 22，如果修改了，输入修改的端口
args.set("user", "新建用户"); //前文所说的新建的操作系统用户，不要用 administrator
args.set("password", "…."); //对应的操作系统用户密码
args.set("local_host", "127.0.0.1"); //隧道创建后，就是运行 PB 的本电脑，也就是 127.0.0.1
args.set("local_port", 8899); //映射到本机的端口，自定义即可
args.set("remote_host", "127.0.0.1"); //这里的 127.0.0.1，要是指云服务器能访问的 IP，它要把这个 IP 上的端口给投递出来，到你 PB 运行电脑上。这个 IP 也可以是云端服务器内网上的其他 IP，只要云端服务器能直接访问都行。
args.set("remote_port", 1433); //云端服务的端口。这里用 sqlserver 测试，没修改默认就是 1433
//args.set("prikey", "id_rsa"); //可以使用公钥验证，而不是使用用户密码。如果有兴趣，可以 AI 查询如何用公钥登录 ssh。如果指定了密钥，上面 “password”就填 passphrase,即私钥的密码
args.set("compress", true); //启用压缩
//开始创建隧道
sshtunnel = create uo_sshtunnel
if not sshtunnel.sshTunnelOpen(args) then return true
//不出意外，到这里隧道就创建成功了。然后使用 127.0.0.1:8899 登录数据库。
```

2．不暴露登录名和密码的方法

首先，同上要创建参数 json:

```
uo_json args; args = create uo_json
args.set("ssh_host", "x.x.x.x"); //你的云服务器 IP
args.set("ssh_port", 22); //ssh 端口，如果没修改默认就是 22，如果修改了，输入修改的端口
args.set("user", "新建用户"); //前文所说的新建的操作系统用户，不要用 administrator
args.set("password", "…."); //对应的操作系统用户密码
args.set("local_host", "127.0.0.1"); //隧道创建后，就是运行 PB 的本电脑，也就是 127.0.0.1
args.set("local_port", 8899); //映射到本机的端口，自定义即可
args.set("remote_host", "127.0.0.1"); //这里的 127.0.0.1，要是指云服务器能访问的 IP，它要把这个 IP 上的端口给投递出来，到你 PB 运行电脑上。这个 IP 也可以是云端服务器内网上的其他 IP，只要云端服务器能直接访问都行。
args.set("remote_port", 1433); //云端服务的端口。这里用 sqlserver 测试，没修改默认就是 1433
//args.set("prikey", "id_rsa"); //可以使用公钥验证，而不是使用用户密码。如果有兴趣，可以 AI 查询
```

如何用公钥登录 ssh。如果指定了密钥，上面 “password”就填 passphrase,即私钥的密码

```
string loginInfo
loginInfo = sshtunnel.EncryptLoginInfo("pbidea123",args)
msgbox(loginInfo)
```

这里的"pbidea123"是一个自定义的密钥，用于对上述登录信息加密。后续登录时同样需要这个密钥。EncryptLoginInfo 返回一个字符串，这个字符串就是加密的登录信息。**保存备用，以后咱们登录就只用它了，不再需要把操作系统用户名和密码暴露在外。**

有了这个保存的 loginInfo 和加密密钥"pbidea123"，我们可以这样登录：

```
string loginInfo
loginInfo = "/dop6SFjOQSpJhtxe91LT5mImUvoTJit9QK1++l/u+9iX1G0hv+UmJsGWIpphykr0n00N9IZ841mVShomtyWx7mn9jAbQOpA9hfm5F+Ew7AjJDxVBSz9NGPIWP/85DH/"
args = create uo_json
args.set("compress", true);
args.set("loginInfo", loginInfo); //前文所生成的 loginInfo，如果登录信息不变，这个字符串可以一直用。
args.set("password", "pbidea123"); //密钥，
if not sshtunnel.sshTunnelOpen(args) then return true
//不出意外，到这里隧道就创建成功了。然后使用 127.0.0.1:8899 登录数据库。
```

**三、连接到数据库**

**注意：不同 PB 版本支持连接驱动不完全相同，有些比较慢，请根据各自情况使用比较快速的驱动。**

```
SQLCA.DBMS = "OLE DB"
//SQLCA.Database = "sshdb"
SQLCA.LogId = "pbidea"
SQLCA.LogPass = "pbidea"
SQLCA.AutoCommit = False
SQLCA.DBParm = "PROVIDER='SQLOLEDB',DATASOURCE='127.0.0.1,8899'"
connect using sqlca;

if SQLCA.sqlcode = 0 then
    //msgbox("连接成功")
else
    msgbox("错误", "连接失败：" + SQLCA.SQLErrText)
     return
end if
```

如何正确连接到数据库，就不再缀述。只是注意：此时连接用的服务器 IP 用的是 127.0.0.1，而不是云端那个 IP。

**四、综述**

1．由于使用了 ssh 隧道技术，全部数据传输都处于加密状态。

2．可以开启 compress，数据压缩传递，对于大量数据库数据，往往压缩率比较大，大大提高速度。

3．uo_sshtunnel 内置了心跳机制，即使没有操作，没有数据传输，通常也不会定时被路由器或防火墙切断连接，确保不掉线。当然，如果真发生了网络故障，掉线还是在所难免。

4．因为数据库不在公网监听，没有暴露在公网上，大大提高了数据库的安全性。

5．公网上的系统，注意密码要有一定复杂度。