SSH Port Forwarding (Tunneling) macOS and Windows

In this example we will forward port ***6102*** to port ***3306*** on the server ***yourserver.com*** using a SSH tunnel

Precondition

You'll need SSH access for your web hosting account. If you are a LivecodeHosting or HostM customer you can contact can request free SSH access. Just submit a ticket.

All other have to check with their providers if SSH access is possible or not.

macOS and Windows 10 have an integrated SSH Client. So that will do, but any other 3rd party SSH-Client can be used also. For Windows there is for example Putty available.

In our example we are using the integrated SSH clients of Windows and macOS.

Establish the SSH tunnel

Run the following command in Windows Cmd.exe or macOS Terminal.app

Variant 1 (tunnel is running in the foreground)

Windows

```
ssh -L 6102:127.0.0.1:3306 youraccountname@yourserver.com [-p portnumber]
```

Mac

```
ssh -L 6102:127.0.0.1:3306 youraccountname@yourserver.com [-p portnumber]
```

If your server is not using the default port 22 for SSH then you have to specify the port also,

e.g. [-p 8228]

The tunnel runs in the foreground and is disconnect either by logging out from the remote server or by closing the terminal/telnet window.

Variant 2 (Background)

Adding the parameter -f forces the command ssh to run in the background, even if the terminal/ telnet window is closed.

Windows

```
ssh –L 6102:127.0.0.1:3306 youraccountname@yourserver.com [–p
portnumber] –f
```

Mac

```
ssh –L 6102:127.0.0.1:3306 youraccountname@yourserver.com [–p
portnumber] –f
```

Regardless if you are using variant 1 or 2, if you connect to the server the first time there is a message/question coming up:

The authenticity of host 'yourserver.com (xxx.xxx.xxx)' can't be established.

ECDSA key fingerprint is SHA256:……some weird characters ;)……..

Are you sure you want to continue connecting (yes/no/[fingerprint])?

Confirm this message/question by typing yes

After that you have to enter your password for your SSH account.

If there is no error message after the password entry, then the tunnel should be established.

If you now want to connect to the remote database server you just connect to

use **127.0.0.1** as the server and according to the above command port **6102**.

So for LC the connection to the DB would look like this

```
put revOpenDatabase("mysql",
"127.0.0.1:6102”,DataBaseName,DataBaseUser ,DataBasePassword,, ,
, ) into tDBID
```

You can even use the computer that established the SSH tunnel as a gateway. This way it is possible to connect from other computers in the same network to the MySQL Database using the computer that established the SSH tunnel.

So instead of 127.0.0.1, use the ip-address of the computer (gateway) Let's say the lan ip address of the computer (gateway) is 192.168.1.12, then you would use this code from an other computer in the same lan

```
put
revOpenDatabase("mysql","192.168.1.12:6102”,DataBaseName,DataBas
eUser ,DataBasePassword,, , , ) into tDBID
```

SSH without entering the password, for example for automated connections

create SSH key

On Windows open the cmd.exe, on macOS Terminal.app.

Run then following command

```
ssh-keygen -t rsa -b 2048
```

Confirm any questions by pressing the RETURN key

After that your personal key will be created.

If you want, you can also enter a password when your are asked for **Passphrase**

This increases the security and make it more difficult to hack your password.

Copy the key to the target server (the server to which you want to connect by SSH)

Now execute the following command to copy you generated key to the remote server

```
ssh-copy-id youraccountname@yourserver.com [-p portnumber]
```

Again, if your server is not using the default port 22 for SSH then you have to specify the port also,

e.g. [-p 8228]

You have to enter your password of your remote SSH account again.

The command copies the key into the folder .ssh im Verzeichnis HOME.

Test the connection (see if we can connect w/o password now)

Run the following command in Windows command shell or macOS Terminal

```
ssh youraccountname@yourserver.com [-p portnumber]
```

Again, if your server is not using the default port 22 for SSH…..;)

The SSH Client should now connect without asking for a password.

If the connection can now be established without the need of entering a password,

we can also establish the SSH tunnel without a password.