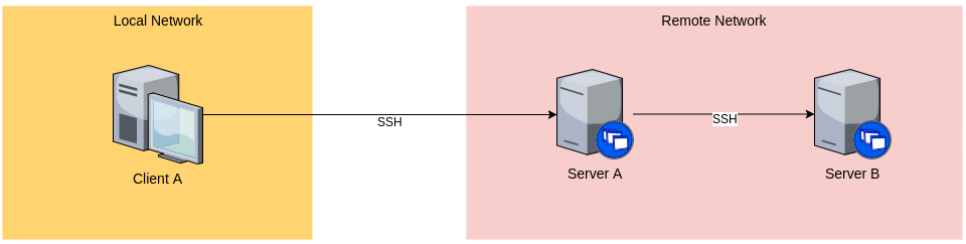
Consider the architecture proposed on the figure below:

Example Network Architecture



The only allowed connection between Client A and Server A is via SSH. The only allowed connection between Server A and Server B is via SSH.

We need to access, from Client A and using HTTP, a service running on port 8000 of Server B.

Notes and constraints:

- There is no direct route from Client A to Server B, and no practical way to build one. Imagine the following scenario as example: Server B belongs to a customer internal datacenter, and we were provided with a VPN that allows access to Server A SSH port only.
- There is another service running on port 8000 of Server A, we must not cause impacts on this one.

Both client and servers run CentOS 7 without X.

We expect you to:

- Provide the set of commands that allow the service to be accessed;
- Provide a document, in english, explaining each of the commands, their possible outputs, including failures, and why you chose them;
- You don't need to be concerned about a production-level solution, users will not be accessing the service in a regular basis. You just need to provide an ad-hoc access (as in the typical scenario: support analyst needs to access web interface for some configuration task);
- However it will be welcome to have a short comment on how you would change the architecture/solution to provide permanent access, in the case users in Local Network start to perform heavy usage of application in Server B.

SSH command description:

The ssh command provides a secure encrypted connection between two hosts.

This connection can also be used for terminal access, file transfers, and for tunneling other applications.

As solution of our challenge it was used to creates an encrypted connection between a local computer (clientA) to access a remote computer ”serverA” that access a another remote computer “serverB” through which you can forwarding the traffic.

Proposed Solution:

- Step 1 - execute the command line on the clientA machine

```
sshpass -p 'userA_password' ssh -g -L ip_clientA:9999:ip_serverA:8001 userA@ip_serverA
```

- Step 2 - execute the command line on the serverA machine

```
sshpass -p 'userB_password' ssh -g -L ip_serverA:8001:ip_serverB:8000 userB@ip_serverB
```

The ideia is create a tunnel forwarding to opens a port 9999 in our clientA local machine that connects through in a port 8001 on the remote serverA that connects on the port 8001 to the remote serverB endding the tunnel.

Commands explanation:

The "sshpass" command is used to set the user password -> sshpass -p user_password;

The “ssh” command combined with –g option is used to enables gateway ports. Remote hosts are allowed to connect to local forwarded ports.

The -L option is used to create a tunel fowarding from port 8001 from serverA to clientA port 9999, the same to step2.

Attention:

Check if the parameter "AllowTcpForwarding" is set to "yes" in /etc/ssh/sshd_config file to both servers (A and B).

The “AllowTcpForwarding” option must be enable on the remote servers to allow port forwarding. By default, forwarding is allowed.

Note:

The network restrictions like firewall rules, connections timeout between servers and unreacheable nodes can generate failure in our proposed challenge.

To provide more security for the solution proposed, we could have uses the public keys just to way to configure authentication without requiring user interaction, because private keys are not transmitted over the encrypted connection.