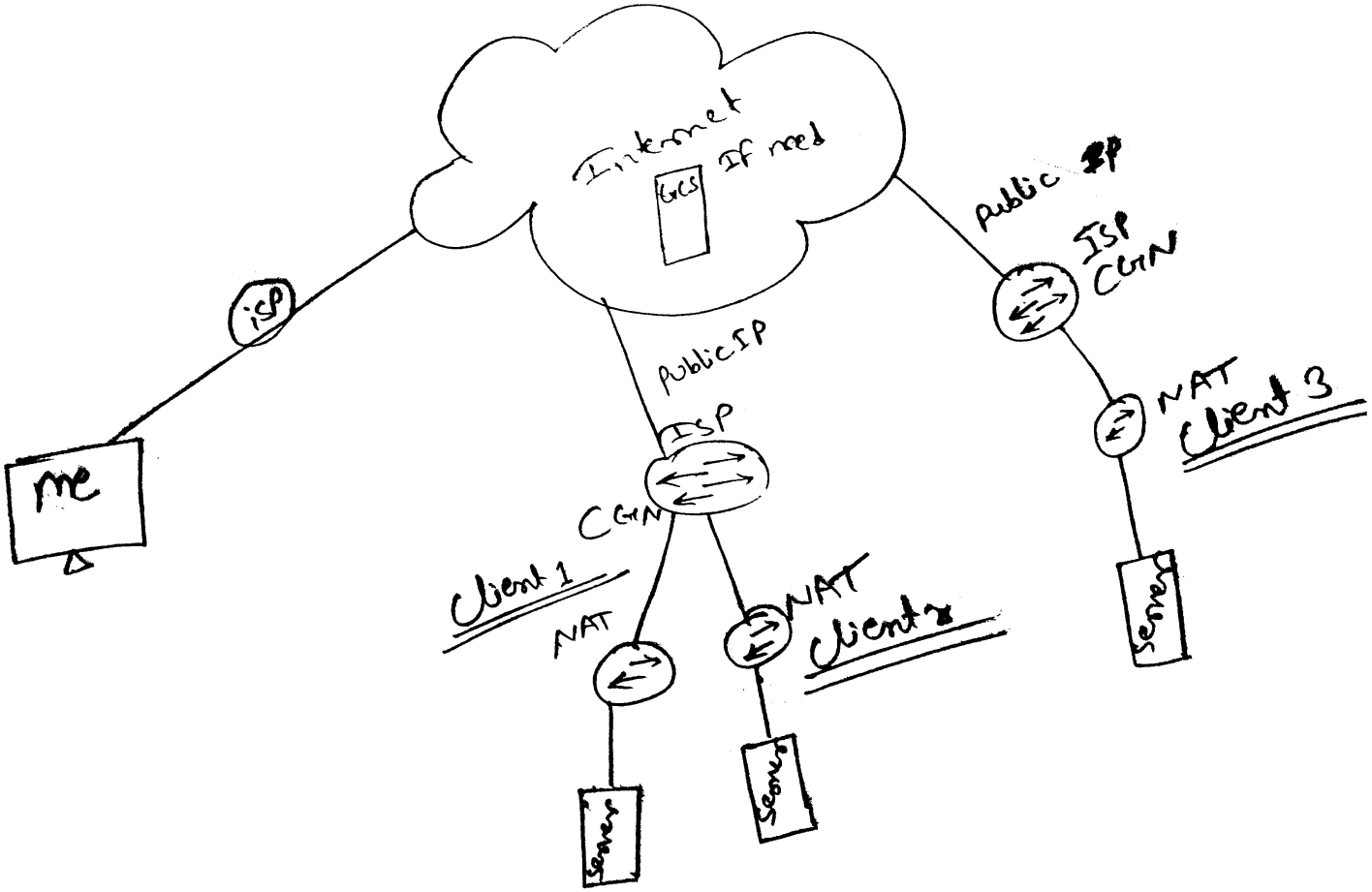
Access remote multiple servers behind NAT [closed]

Ask Question

1

I have a situation where our Ubuntu servers were deployed across multiple remote locations. These servers were behind a carrier grade NAT and also an internal NAT. Now from a central location I want to access it any time I want, but I am also behind a NAT similar to those servers.



When those servers are online I want to access it (through SSH or SSH tunnel). I know I can't access like servers that have a public IP, but maybe somehow I can access it through TeamViewer's working principle. For

this method if I need another public server I can manage it in Google Compute Engine.

networking server ssh remote-access teamviewer Edit tags

share edit reopen delete flag

edited Feb 12 '18 at 6:53

karel
58.5k 13 128 147

asked Feb 12 '18 at 5:21

Sagar Khatiwada
10 1 4

I have full control over servers but might not be able to do networking on clients premises.May be some VPN Stuff.. – Sagar Khatiwada Feb 12 '18 at 5:24

add a comment

1 Answer

active oldest votes

0

If we have access to a server with public IP address we can use it as Gateway. We could achieve this via Reverse SSH Port Forwarding connections. Let's say we have three instances:

- **Public Server**: Here we have an operational SSH server and public IP address (and/or domain name). We are able to connect to this server with public-private key pair, which is not passphrase protected (reference).
- **Private Server**: Here we have an operational SSH server. It is behind a Firewall (NAT, ISP, etc.) and doesn't have a public IP address, but we are able to establish a SSH connection from it to the Public Server, so here we have also SSH client.
- **Client Machine**: Here we (need to) have only SSH client. We are able to establish a SSH connection to the Public Server. **We want to establish a SSH connection from this instance to the Private Server.**

The principal level

At principal level we could apply at least two scenarios.

Scenario 1. Where we don't want to open an additional ports in the Public Server's Firewall:

1. Establish SSH connection with Port Forwarding from the Private Server to the Public Server.
2. Establish SSH connection with Port Forwarding from the Client Machine to the Public Server.
3. Connect to the Private Server from the Client Machine through itself.

Scenario 2. Where we are inclined to open an additional port in the Public Server's Firewall:

1. Establish SSH connection with Port Forwarding from the Private Server to the Public Server.
2. Connect to the Private Server from the Client Machine through the Public Server.

The main advantage of the Scenario 1 is that we don't need to think how secure is our Private Server. The main advantage of the Scenario 2 is that we ommit one step, but in this case we should think about the security of the Private Server, because it becomes public accessible through the forwarded port. In addition, these scenarios could be applied to different port and services, not only SSH, for example to HTTP.

How to apply Scenario 1 within Ubuntu

Establish SSH connection with Port Forwarding from the Private Server to the Public Server

We could do that by the command:

```
ssh user-of-the-public-server@public-server -p 22 -R 2222:127.0.0.1:22 -i ~/.ssh/pass-less/id_
```

- `-p 22` provides the SSH port of the Public Server (it is not mandatory).
- `-i ~/.ssh/pass-less/id_rsa` provides the authentication key file.
- `-R 2222:127.0.0.1:22` means that port `2222` on the (remote) Public Server will be forwarded to the SSH port of the (local) Private Server that, in this case, is `22`.

We can push this connection into the background by adding the options `-fTN` (of the OpenSSH Client – reference). We could use also the tool `autossh` to be sure the connection will be kept alive for a long period of time (reference):

```
autossh user-of-the-public-server@public-server -p 22 -fTN -R 2222:127.0.0.1:22 -i ~/.ssh/pass
```

We can simplify the above command by implementation of the following lines in our `~/.ssh/config` file:

```
Host public-server-reverse
    HostName 100.100.100.100  # this is the IP address or the domain name of the Public Server
    IdentityFile ~/.ssh/pass-less/id_rsa
    User user-of-the-public-server
    Port 22
    RemoteForward 2222 localhost:22
```

In this case the above command will become:

```
autossh public-server-reverse -fTN
```

We can easily automate this task on system reboot by the next Cron job (or we could create service that should be the better approach):

```
@reboot sleep 15 && autossh public-server-reverse -fTN
```

Once this connection is established we can connect to the Private Server from the Public Server, through the reverse tunnel, by the command:

```
ssh user-of-the-private-server@localhost -p 2222    # provide an additional authentication dat
```

Establish SSH connection with Port Forwarding from the Client Machine to the Public Server

We could do that by the command:

```
ssh user-of-the-public-server@public-server -p 22 -fTN -L 1111:127.0.0.1:2222 -i ~/.ssh/pass-l
```

- `-L 1111:127.0.0.1:2222` means that port `1111` on the (local) Client Machine will be forwarded to the (remote) Public Server's port `2222` (that is forwarded to the SSH port of the Private Server). Note we could use `-L 2222:127.0.0.1:2222`.
- `-fTN` these options will push the connection into the background as it is described above.

- We can implement also `autossh` and `~/.ssh/config` file, with directive `RemoteForward` or `LocalForward`.

Connect to the Private Server from the Client Machine through itself

Once the above two steps are implemented, we can connect from the Client Machine to the Public Server by the command:

```
ssh user-of-the-private-server@localhost -p 1111    # provide an additional authentication dat
```

How to apply Scenario 2 within Ubuntu

Establish SSH connection with Port Forwarding from the Private Server to the Public Server

This step is identical as the first one from the Scenario 1, but few additional things should be done.

Open the forwarded port `2222` on the Public Server's Firewall - this is out of the scope of this answer.

Modify `/etc/ssh/sshd_config` of the Public Server and add the next directive, that will allow to open our SSH tunnel to public (don't forget to restart the server: `sudo systemctl restart sshd.service`):

```
GatewayPorts yes
```

Make our SSH tunnel open to public. This step is well described within the question: (How to make ssh tunnel open to public?). According to the answers there we could add the `-g` option to the commands that make a connection between the Private and the Public servers:

```
autossh public-server-reverse -gfTN
```

```
@reboot sleep 15 && autossh public-server-reverse -gfTN
```

Or, alternatively, instead that we could modify the `~/.ssh/config` file in this way:

```
Host public-server-reverse
    HostName 100.100.100.100  # this is the IP address or the domain name of the Public Server
```

```
    IdentityFile ~/.ssh/pass-less/id_rsa
    User user-of-the-public-server
    Port 22
    RemoteForward \*:2222 localhost:22
```

Finally we should `killall autossh` and establish the connection again.

Connect to the Private Server from the Client Machine through the Public Server

Once the above step is performed, we can achieve our goal by the command:

```
ssh user-of-the-private-server@public-server -p 2222    # provide an additional authentication
```

share edit delete flag

edited Feb 17 '18 at 12:05

answered Feb 14 '18 at 20:43

pa4080
13.9k 5 25 64

add a comment