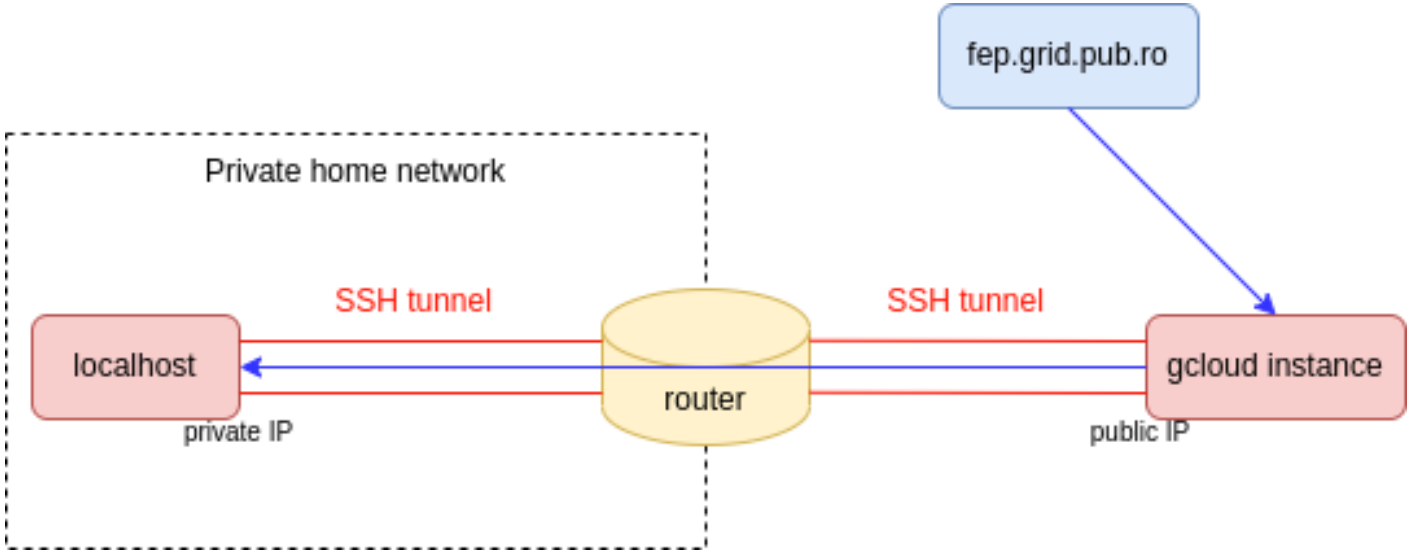
03. [30p] Reverse SSH



Loosely speaking, there are two types of IP addresses:

- **Public**: what your Google Cloud instance has, and what allows you to contact it from anywhere over the Internet.
- **Private**: what your router allocates to your devices (laptop, phone, etc.) in your local network.

Public addresses can be uniquely identified in the Internet. Private addresses can't. Why not, you ask? Well… can you find the IP address assigned to your machine's network interface? Hint: **ip addr show**. That *exact* IP address is shared by millions of other devices, all across the world, in their respective local networks. When you initiate connections from your private network, the router performs a process called Network Address Translation (NAT) [https://avinetworks.com/glossary/network-address-translation/] which uses a single Public IP to represent multiple Private IPs. Unless you have administrative access to said router (too add some custom configurations), clients outside your network will be unable to initiate contact with individual machines inside your network. And most times you don't…

In this exercise, we will set up what is called a reverse SSH tunnel. This tunnel is a persistent two-way communication channel between your computer and the gcloud instance. While this connection is initiated by you (from your private network, to a public server), someone on the other side can piggy back on this channel to initiate connections with you, *in your private network*. The reason for this is that it doesn't target your IP, specifically, in its request. In stead, it sends its request into the gcloud endpoint of the channel and it just so happens that the other endpoint is located on your machine.

To be more specific, the scenario is as follows:

1. You create a SSH channel from your machine to the google cloud instance
2. Then, you connect to fep.grid.pub.ro. SSH-ing from fep back to your computer should be impossible. Imagine your fep instance being you on vacation, trying to access your home computer.
3. To sidestep this problem, you will SSH from fep to your Google Cloud instance, and from Google Cloud to your localhost via the SSH channel.

In the following commands *[localhost]* means that the command should be executed on your computer, *[gcloud]* on the Google Cloud VM, and *[fep]* on fep.grid.pub.ro.

If you are going to SSH from fep.grid.pub.ro to your gcloud instance, you will need to:

1. create a SSH public keypair on your fep account (if not already there).
2. configure the fep public key on your google cloud instance.

If you don't do this, access from fep to your gcloud instance will be denied.

Also, if you want to SSH from gcloud to your localhost via the SSH tunnel, note that you must have a SSH server installed and running.

```
# create a public keypair on fep, if you don't already have one

# create the .ssh directory (if not already there)
[gcloud]$ mkdir -p ~/.ssh

# print out your fep.grid.pub.ro public key and copy it
[fep]$ cat ~/.ssh/id_rsa.pub

# configure your fep.grid.pub.ro public key on gcloud instance
[gcloud]$ vim ~/.ssh/authorized_keys

# install a ssh server on your computer
[localhost]$ sudo apt install openssh-server

# check if the ssh server is running; if not, start it
[localhost]$ systemctl status sshd
[localhost]$ sudo systemctl start sshd
```

And now for the main part:

```
# create a reverse ssh tunnel from your computer to the cloud instance
[localhost]$ ssh -T -N -R 43210:localhost:22 ${GCLOUD_USERNAME}@${GCLOUD_IP}

# show tcp listeners (bound ports, processes, etc.)
[gcloud]$ netstat -tlpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:43210         0.0.0.0:*               LISTEN      1931/sshd: ........
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      448/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      893/sshd: /usr/sbin

# test that the reverse ssh tunnel works
[gcloud]$ ssh ${LOCALHOST_USERNAME}@localhost -p 43210

# connect from fep.grid.pub.ro to your localhost via gcloud instance
[fep]$ ssh -J ${GCLOUD_USERNAME}@${GCLOUD_IP} ${LOCALHOST_USERNAME}@localhost -p 43210
```

Let's take a look at some of the arguments used in these commands:

- `ssh -T -N -R 43210:localhost:22 …`
  - `-T`: do not start a shell on the remote computer (we're not using the connection for that, remember?)
  - `-N`: do not execute one-shot commands either (combined with `-T`, this makes **ssh** do nothing)
  - `-R 43210:localhost:22`: when logged in on the gcloud instance, writing data to port 43210 with itself (i.e.: localhost) intended as the destination, will ensure that the data actually ends up on whoever initiated the tunnel, on port 22 (the SSH server port).
- `netstat -tlpn`
  - `t`: filter for TCP protocol (SSH actually runs *over* TCP)
  - `l`: show ports where processes are listening for new connections
  - `p`: show the PID and name of the program that is listening
  - `n`: use numeric IP addresses in the output
- `ssh -J ${GCLOUD_USERNAME}@${GCLOUD_IP} ${LOCALHOST_USERNAME}@localhost -p 43210`
  - `-J ${GCLOUD_USERNAME}@${GCLOUD_IP}`: create a SSH connection with google cloud as an intermediary hop; it acts like you'd SSH to google cloud, and there you would write another SSH command to your final destination.
  - `${LOCALHOST_USERNAME}@localhost`: *"localhost"* here is relative to the jump point (i.e.: google cloud instance), not to fep.
  - `-p 43210`: in stead of using the default SSH server port 22, pretend that it's in fact running on 43210.

Regarding the output from **netstat**:

- `127.0.0.1:43210`: on port 43210 there is a process listening for connections towards IP *127.0.0.1*. This is synonymous with *localhost*, denoting a loopback interface. In other words, this IP always refers to the current machine.
- `127.0.0.53:53`: this IP also corresponds to a loopback interface. The process listening on port 53 for connections to *127.0.0.53* is a DNS [https://www.cloudflare.com/learning/dns/what-is-dns/] caching server that comes pre-installed on Ubuntu. This is not relevant for this exercise.
- `0.0.0.0:22`: on port 22, there is a SSH server called **sshd**, listening for incoming connection requests. The *0.0.0.0* IP signifies a catch-all. In other words, it doesn't matter if the request's destination IP is that of your Ethernet interface, or your WiFi interface, or your loopback interface (*127.0.0.**). This server will take on all of them.

Reiterating over the last command, since it might still be a bit unclear:

- You are on fep.grid.pub.ro and you SSH to the Google Cloud instance (the `-J` part)
- Once on the Google Cloud instance, it may appear that you're trying to do something utterly insane: you are trying to log in via SSH with your personal computer's username, on the very same Google Cloud instance (that's your localhost at that moment in time), using a SSH server that doesn't run on port 22 (like a normal SSH server) but on port 43210…
- When you initiate that connection you realize that localhost:43210 is in fact a… *wormhole*. localhost is not in fact the Google Cloud instance, but your personal computer. Port 43210 is not in fact port 43210, but port 22. But you already know that, didn't you?