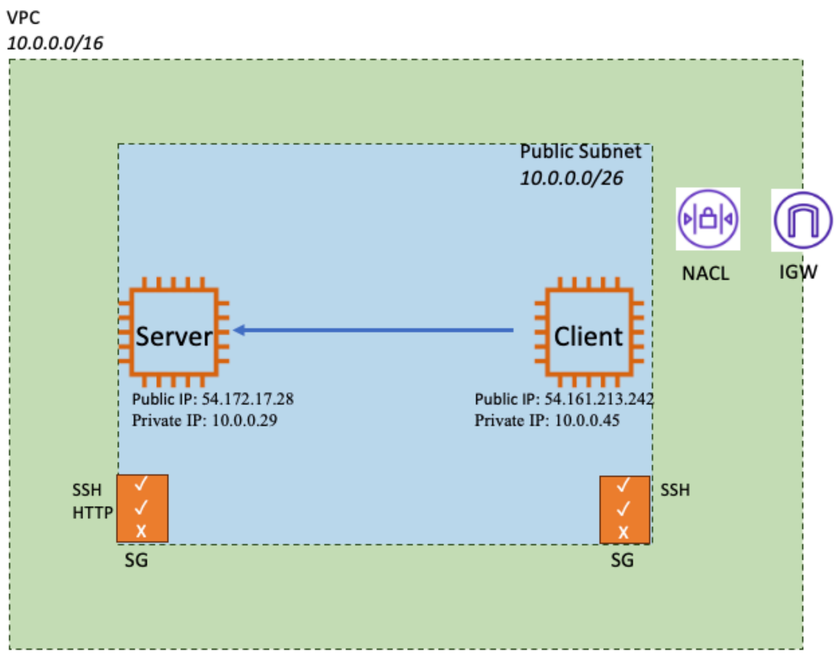
Networking project

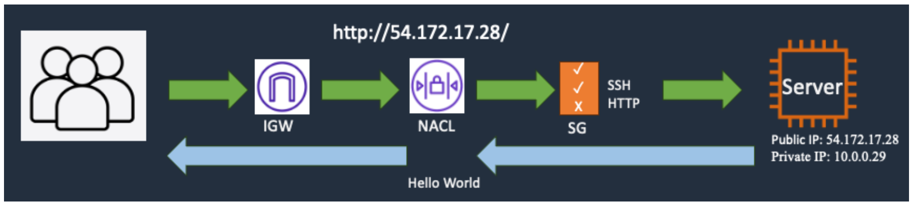
Answer all questions in a text document so you can show your mentor after completing the lab. Your mentor will review your answers and ensure solid connectivity between the two EC2 instances you created. Good luck!

- Launch an EC2 instance in the default VPC. This will be our web-server or server instance.
  - 

  - In the security group rules, ensure the web-server instance allows **ALL HTTP:80 traffic (0.0.0.0/0)**
    - Also add a rule to allow **SSH:22 traffic** from YOUR IP (if you are facing issues SSHing, try allowing 0.0.00.0/0 traffic. This is not good practice, but for the purposes of this lab it's OK).
    - **Do not add inbound rules besides SSH & HTTP**
    - Keep outbound rules ALL 0.0.0.0/0
    - *Don't lose your SSH key!*
- In the user data section [1], paste the following script:

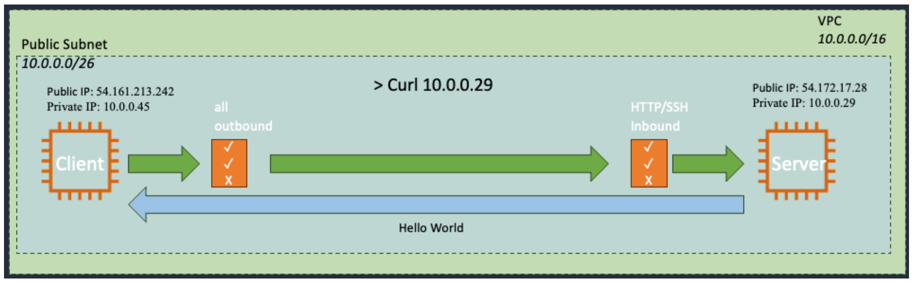
```
#!/bin/bash
yum update -y
yum install -y httpd
echo '<h1>Hello World</h1>' > /var/www/html/index.html
systemctl start httpd
systemctl enable httpd
```

  - The above script simply installs apache (httpd) and creates a simple *Hello World* web page
- Test connectivity to your web-server instance by pasting the public IP of the instance in any web browser
  - ***How does the EC2 instance know its way out to the internet? (Answer this question in a text document)... Hint: Check the Route Table assigned to the subnet of the EC2 instance***
    -



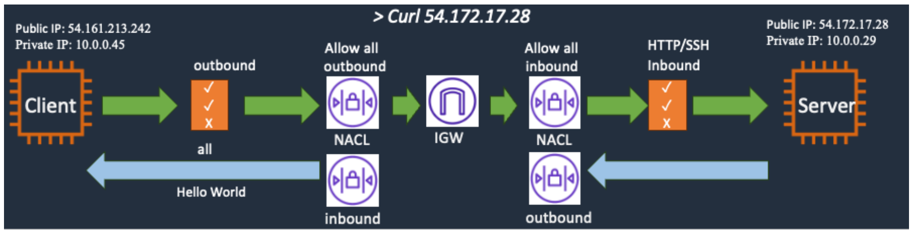
    -
      ```
      From the VPC section, we need to set up an Internet gateway for the
      public subnet. Since the EC2 instance is in the public subnet, it can
      access the Internet. However, the EC2 instance still needs to determine
      which protocols can be accessed. Therefore, we need to set up inbound
      rules in the security group to allow SSH or HTTP access.
      ```
  - If you are not able to connect to your EC2 instance, check the route table... something important is probably missing. Also ensure the IP is public.
- After confirming connectivity to the web-server EC2 instance, launch a second EC2 instance. This second EC2 instance will be our client instance. Like the web-server instance, ensure you allow SSH access into the client instance from your IP. Do not add user data to this client EC2 instance.
  - **Do not add inbound rules besides SSH**
  - Keep outbound rules ALL 0.0.0.0/0
- Now **SSH into the client instance** and test connectivity to the web-server by running the following curl command:

```
curl <private-IP of web-server>
```

  - ***Does connectivity work? If so, what do you see? Did we traverse the internet while making this request?***
    - 

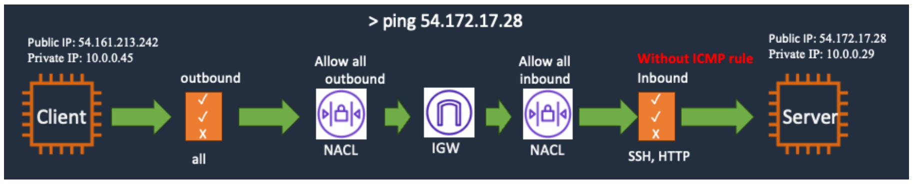
    -
      ```
      The connectivity works. The terminal returns “Hello, World.”
      The traffic does not traverse the internet; instead, it stays
      within the AWS internal network.
      ```
- Run the command again, but this time, use the PUBLIC IP of the web-sever

```
curl <public-IP of web-server>
```

  - ***Is there any difference in response? Did we traverse the internet while making this request?***
    - 

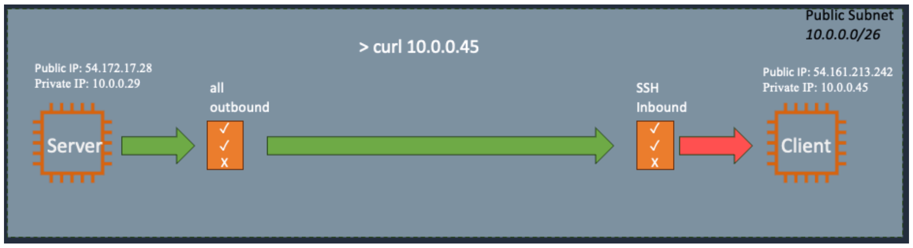
    -
      ```
      Same response, but the traffic does traverse the internet while making
      this request. Despite both instances being in the same subnet, using
      the public IP forces the traffic to go outside the internal AWS network,
      ```

```
making a round trip over the internet before returning to the second
instance. This route involves extra latency and potential security
risks compared to using private IPs.
```

- Now try **pinging** the web-server instance from the client instance
  - `ping <public-IP of web-server>`
  - ***Did pinging the web-server work? Why? Hint: What protocol does PING use?***
    - 

    -
      ```
      Pinging the web server does not work because ping requires the ICMP
      protocol. In the security group of the web server, only SSH & HTTP
      traffic are allowed.
      ```
- Now **SSH into the web-server** EC2 instance (keep your previous SSH session with the client instance open). Try performing a curl to the private-IP of the client Instance

```
curl <private-IP of client instance>
```

  - **What was the response? If connectivity failed, explain the possible cause.**
    - 

    -
      ```
      There is no response because, in the security group of the client
      instance, only SSH traffic is allowed, but the curl command requires
      the HTTP protocol.
      ```

Bonus Points

- *Wireshark is not mandatory for this bonus portion. You can still receive credit by sticking with tcpdump to capture and read the PCAP.*
- If `tcpdump` is not installed on the **web-server** instance, run this command first:
  - `sudo yum install tcpdump`
- Run a packet capture from the web-server instance with the following command

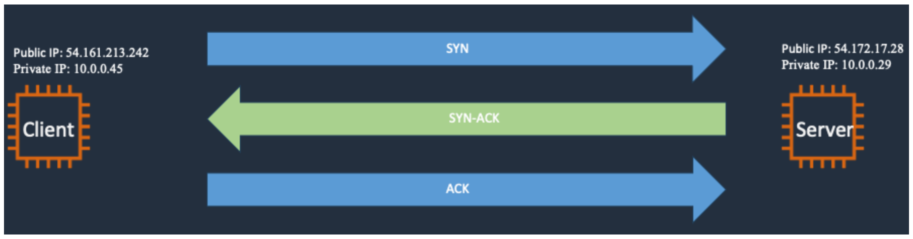
```
sudo tcpdump host <private-ip of client instance> and port 80 -w file.pcap
```

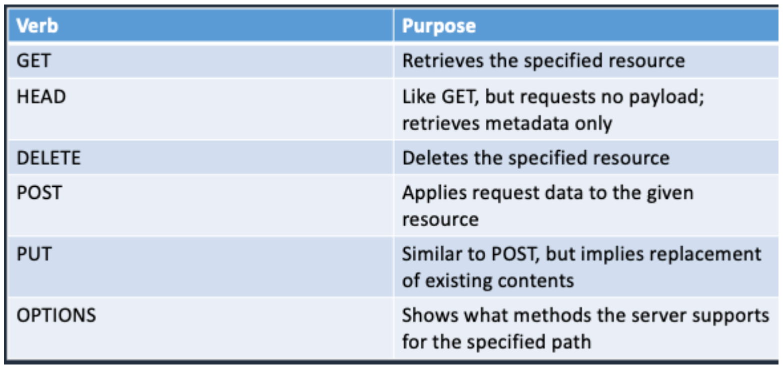
  - The above command runs a packet capture on port 80 of the EC2 instance and captures traffic from the private IP address of the client instance . The output of this capture is written to a PCAP file called file.pcap.
  - While the `tcpdump` is running, go back into your **client EC2 instance** and run a few curl commands to the private IP of the web-server (`curl <public-IP of web-server>`)
  - After running a few curl commands from the client instance, stop the capture on the **web-server** instance by typing ctrl-C. You only need to run a few curl commands to capture the data we need here.
  - Now on the **web-server instance**, you can open this PCAP file by running the command:

```
tcpdump -r file.pcap
```

- However, if you have Wireshark installed on your local machine (install it if you don't have it [2]), you can download the PCAP file from your EC2 instance onto your local machine, then open the file with wireshark
- To download the file from your web-server EC2 instance onto your local machine, run the following command **from your local machine within the directory where your SSH key is stored**

```
scp -i <YOUR SSH KEY> ec2-user@1<public-IP of web-server instance>:/home/ec2-
user/file.pcap /Users/<USERNAME>/Desktop/file.pcap
```

- You may have to modify the `/Users/<USERNAME>/Desktop/file.pcap` portion of the command as this specifies where the PCAP file from EC2 should be installed on your local machine. The example above is for mac and I'm downloading the file to my Desktop.
- Regardless of how you decide to view the packet capture (either through `tcpdump -r` or by downloading to your local machine and viewing with Wireshark), you should be able to verify the following information:
- ***Who initiates the TCP three-way handshake?***
  - 

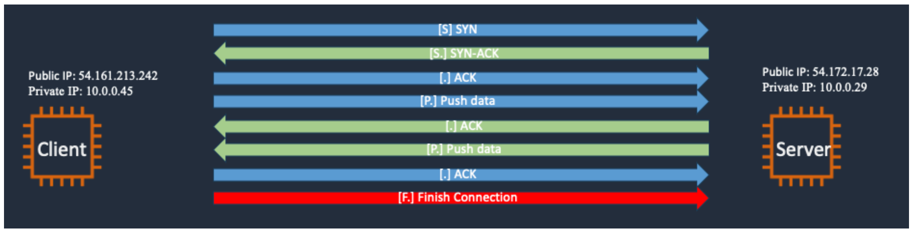
  -
    ```
    Client instance initiates the three-way handshake by sending the first
    SYN packet. This is typical for HTTP requests, where the client starts
    the process to establish a TCP connection.
    ```
- ***What is the HTTP request method?***
  - 

Verb	Purpose
GET	Retrieves the specified resource
HEAD	Like GET, but requests no payload; retrieves metadata only
DELETE	Deletes the specified resource
POST	Applies request data to the given resource
PUT	Similar to POST, but implies replacement of existing contents
OPTIONS	Shows what methods the server supports for the specified path

  -
    ```
    "In its simplest form, HTTP is a client/server, one-request/one-response
    protocol."
     - 2017 Nemeth Evi etal - UNIX and Linux System Administration Handbook[
    ```
- ***How is the connection closed between the two peers? (What TCP flags do you see?)***

  -
    ```
    [ec2-user@ip-10-0-0-29 ~]$ tcpdump -r file.pcap
    reading from file file.pcap, link-type EN10MB (Ethernet), snapshot lengt
    22:27:49.963156 IP 10.0.0.45.56178 > ip-10-0-0-29.ec2.internal.http: Fla
    22:27:49.963372 IP ip-10-0-0-29.ec2.internal.http > 10.0.0.45.56178: Fla
    22:27:49.963757 IP 10.0.0.45.56178 > ip-10-0-0-29.ec2.internal.http: Fla
    22:27:49.963757 IP 10.0.0.45.56178 > ip-10-0-0-29.ec2.internal.http: Fla
    22:27:49.963781 IP ip-10-0-0-29.ec2.internal.http > 10.0.0.45.56178: Fla
    ```

```
22:27:49.964123 IP ip-10-0-0-29.ec2.internal.http > 10.0.0.45.56178: Fla
22:27:49.964541 IP 10.0.0.45.56178 > ip-10-0-0-29.ec2.internal.http: Fla
22:27:49.964707 IP 10.0.0.45.56178 > ip-10-0-0-29.ec2.internal.http: Fla
```

- 

-
  ```
  1): Client send SYN flag to server
      - Client: "I want to connect with you"
  2): Server send back SYN-ACK to client
      - Server: "I'm ready, are you ready?"
  3): Client send ACK to server
      - Client: "Yes, I'm also ready"
  4): Client push data to client
      - Client: "I want this..."
  5): Server send ACK to client
      - Server: "ok, I know what you want now"
  6): Server push data to client
      - Server: "here is the data you request"
  7): Client send ACK to server
      - Client: "thanks, I recieve your data"
  8): Client send finish connection messgae to server
      - Client: "Thanks again, see you next time."

  In step 8), client send a [F.] flag to server, which means
  finish connection.
  ```

- If you're using the `tcpdump -r` command to verify the above information, read the below section:
- `TCPDUMP FLAGS`
  - The main ones are:
    - `[S] = SYN`
    - `[.] = ACK`
    - `[S.] = SYN-ACK`
  - The rest are as follows:
  - `Unskilled = URG = (Not Displayed in Flag Field, Displayed elsewhere)`
  - `Attackers = ACK = (Not Displayed in Flag Field, Displayed elsewhere)`
  - `Pester = PSH = [P] (Push Data)`
  - `Real = RST = [R] (Reset Connection)`
  - `Security = SYN = [S] (Start Connection)`
  - `Folks = FIN = [F] (Finish Connection)`
  - `SYN-ACK = [S.] (SynAcK Packet)`
  - `[.] (No Flag Set)`
  - For more information on reading tcpdump output, check out the following link [3].

- If you completed all the above steps, great job! You now know how to set up TCP connections between two instances, ensure solid connectivity, then perform a packet capture to trace each step of the connection. There are many more ways to dive even deeper into this lab. If you'd like to do so, try creating your OWN VPC instead of using the default VPC. By creating a custom VPC, you'll be responsible for setting up internet access for your servers, assigning public IP's, and setting up the routing. Then you can try setting up VPC Flow Logs and monitoring traffic as it traverses each hop (ENI) within your VPC.

References

---

[1] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html#user-data-launch-instance-wizard
[2] https://www.wireshark.org/
[3] https://opensource.com/article/18/10/introduction-tcpdump