HW3 Report

Mai Trinh
COSC 6377 Computer Networks[*], University of Houston

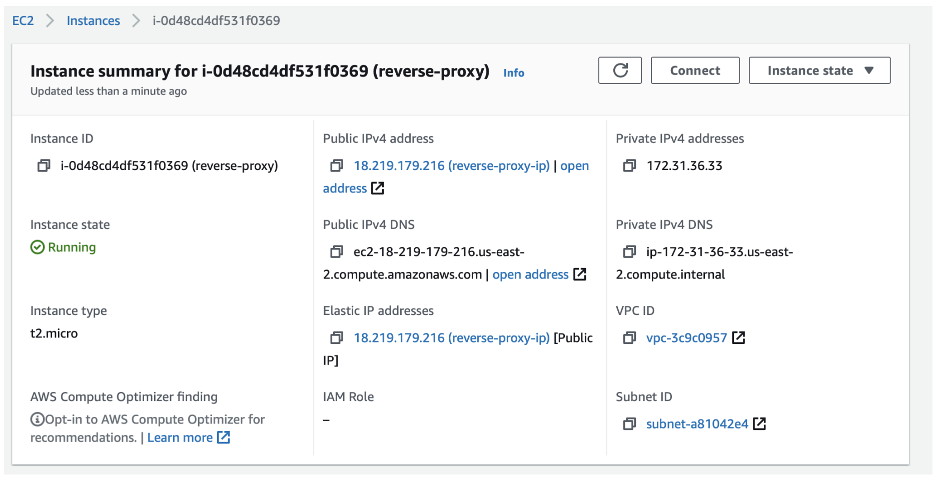
April 30, 2021

1 Reverse Proxy EC2 Instance

1.1 Create and launch EC2 Instance

Follow this guide https://docs.aws.amazon.com/quickstarts/latest/vmlaunch/step-1-launch-instance.html.

Create Elastic IP and assign it to reverse proxy EC2 Instance. By doing this, IP address associated with the instance will not change even when the instance is stopped.

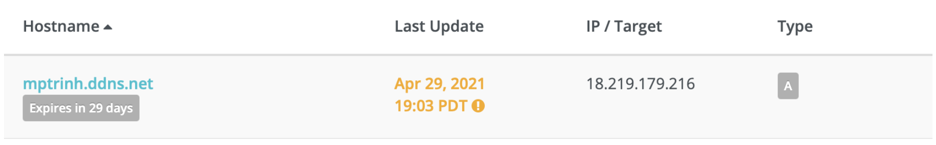


EC2 > Instances > i-0d48cd4df531f0369

Instance summary for i-0d48cd4df531f0369 (reverse-proxy) Info
Updated less than a minute ago

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0d48cd4df531f0369 (reverse-proxy)	18.219.179.216 (reverse-proxy-ip) | open address	172.31.36.33
Instance state	**Public IPv4 DNS**	**Private IPv4 DNS**
Running	ec2-18-219-179-216.us-east-2.compute.amazonaws.com | open address	ip-172-31-36-33.us-east-2.compute.internal
Instance type	**Elastic IP addresses**	**VPC ID**
t2.micro	18.219.179.216 (reverse-proxy-ip) [Public IP]	vpc-3c9c0957
AWS Compute Optimizer finding	**IAM Role**	**Subnet ID**
Opt-in to AWS Compute Optimizer for recommendations. | Learn more	–	subnet-a81042e4

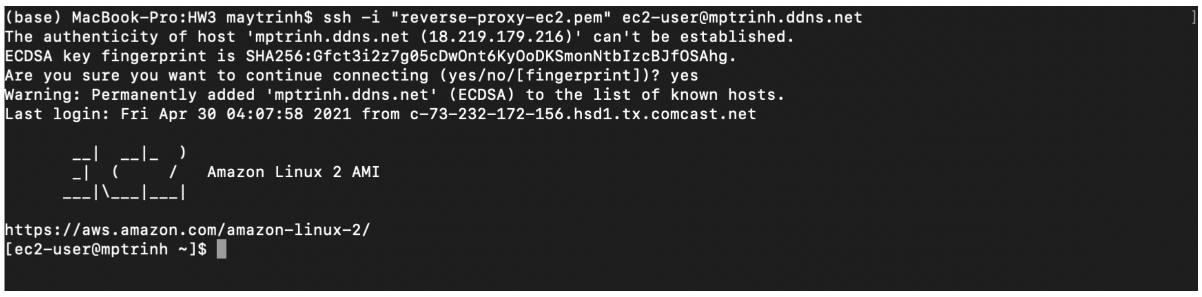
Register a domain and add a A record connecting the IP address of reverse proxy instance to that domain name. I use NoIP.



Hostname ▲	Last Update	IP / Target	Type
mptrinh.ddns.net Expires in 29 days	Apr 29, 2021 19:03 PDT	18.219.179.216	A

To connect to EC2 instance from local machine. Run the below command in the directory where *reverse-proxy-ec2.pem* file is located.

```
ssh -i "reverse-proxy-ec2.pem" ec2-user@mptrinh.ddns.net
```



```
(base) MacBook-Pro:HW3 maytrinh$ ssh -i "reverse-proxy-ec2.pem" ec2-user@mptrinh.ddns.net
The authenticity of host 'mptrinh.ddns.net (18.219.179.216)' can't be established.
ECDSA key fingerprint is SHA256:Gfct3i2z7g05cDwOnt6KyOoDKSmonNtbIzcBJfOSAhg.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'mptrinh.ddns.net' (ECDSA) to the list of known hosts.
Last login: Fri Apr 30 04:07:58 2021 from c-73-232-172-156.hsd1.tx.comcast.net

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@mptrinh ~]$ 
```

1.2 Deploy Python code to AWS EC2 Instance

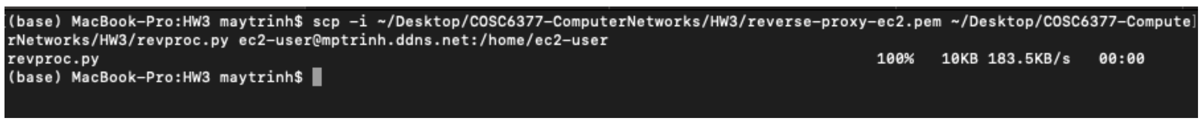
Once the EC2 instance up and running on AWS, to deploy Python code from local machine to the remote instance follow this guide https://praneeth-kandula.medium.com/running-python-scripts-on-an-aws-ec2-instance-8c01f9ee7b2f

There is a slight modify in reverse proxy code where we need to save the server's public IP address in the switch table. Since each server has its own dedicated IP, we need to save the IP address to connect to server later on. Then, run this command below to transfer *revproc.py* file to remote instance.

```
scp -i ~/Desktop/COSC6377-ComputerNetworks/HW3/reverse-proxy-ec2.pem ~/Desktop/COSC6377-
    ComputerNetworks/HW3/revproc.py ec2-user@mptrinh.ddns.net:/home/ec2-user
```

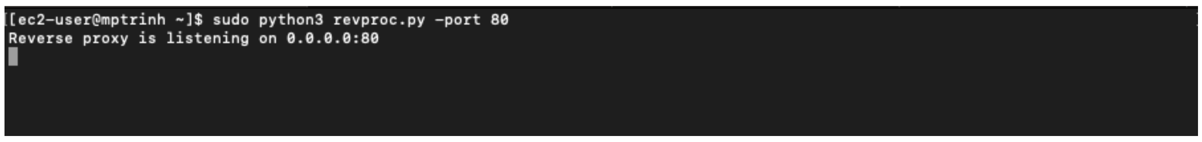
[*]Course assignments

If the transfer is successful, the response is similar to the following:



```
(base) MacBook-Pro:HW3 maytrinh$ scp -i ~/Desktop/COSC6377-ComputerNetworks/HW3/reverse-proxy-ec2.pem ~/Desktop/COSC6377-Compute
rNetworks/HW3/revproc.py ec2-user@mptrinh.ddns.net:/home/ec2-user
revproc.py                                                                      100%   10KB 183.5KB/s   00:00
(base) MacBook-Pro:HW3 maytrinh$ 
```

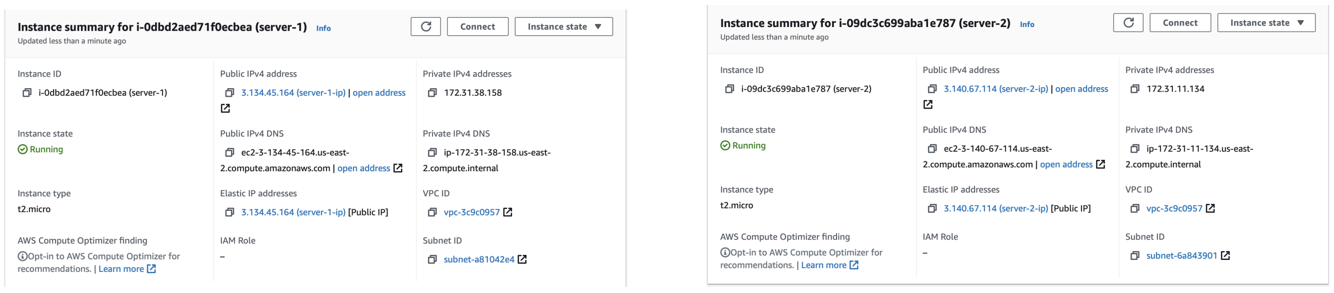
Go to the directory in EC2 containing the file and run the python code. While it looks like the reverse proxy is running on *localhost:80/*, that's on the virtual machine. To access the reverse proxy, we will have to use the instance's public DNS (or hostname).



```
[ec2-user@mptrinh ~]$ sudo python3 revproc.py -port 80
Reverse proxy is listening on 0.0.0.0:80
```

2 Servers EC2 Instances

Create, launch and assign elastic IP addresses for 2 EC2 instances (following the same structure as above, except we do not need to get a human readable domain name for servers).



Instance summary for i-0dbd2aed71f0ecbea (server-1) Info
Updated less than a minute ago

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0dbd2aed71f0ecbea (server-1)	3.134.45.164 (server-1-ip) | open address	172.31.38.158
Instance state	**Public IPv4 DNS**	**Private IPv4 DNS**
Running	ec2-3-134-45-164.us-east-2.compute.amazonaws.com | open address	ip-172-31-38-158.us-east-2.compute.internal
Instance type	**Elastic IP addresses**	**VPC ID**
t2.micro	3.134.45.164 (server-1-ip) [Public IP]	vpc-3c9c0957
AWS Compute Optimizer finding	**IAM Role**	**Subnet ID**
Opt-in to AWS Compute Optimizer for recommendations. | Learn more	–	subnet-a81042e4

Instance summary for i-09dc3c699aba1e787 (server-2) Info
Updated less than a minute ago

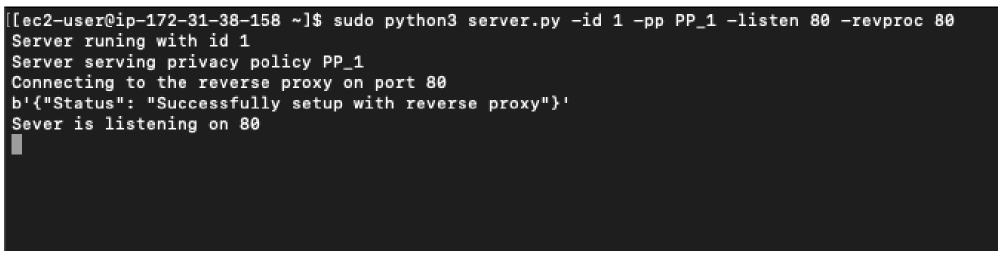
Instance ID	Public IPv4 address	Private IPv4 addresses
i-09dc3c699aba1e787 (server-2)	3.140.67.114 (server-2-ip) | open address	172.31.11.134
Instance state	**Public IPv4 DNS**	**Private IPv4 DNS**
Running	ec2-3-140-67-114.us-east-2.compute.amazonaws.com | open address	ip-172-31-11-134.us-east-2.compute.internal
Instance type	**Elastic IP addresses**	**VPC ID**
t2.micro	3.140.67.114 (server-2-ip) [Public IP]	vpc-3c9c0957
AWS Compute Optimizer finding	**IAM Role**	**Subnet ID**
Opt-in to AWS Compute Optimizer for recommendations. | Learn more	–	subnet-6a843901

Figure 1: EC2 Instance for Server 1 (left) and Server 2 (right)

To correctly run this, we need to slightly modify the code. For Server code, we need to identify the hostname for Reverse Proxy instance. For this, we can either use reverse proxy public IP address or its hostname. When server wants to send setup message to reverse proxy, it will connect to reverse proxy using reverse proxy's host and port. When server listens to new connection, it will use its own host and port.

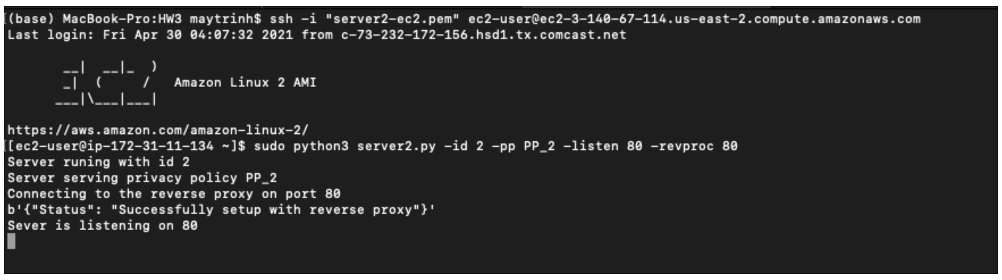
```python
self.revhost = '18.219.179.216' # revproc public IP address
self.revproc = args.revproc #well-known port on which the reverse proxy is running

self.host = '0.0.0.0'
self.port = args.listen #arbitrary non-privileged port

def connect_to_proxy(self):
    self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

    # connect to reverse proxy
    self.socket.connect((self.revhost, self.revproc))
    print("Connecting to the reverse proxy on port", self.revproc)
     ...
     ...
     ...

def listen(self):
    self.serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.serverSocket.bind((self.host, self.port))
    self.serverSocket.listen(1)
    print("Sever is listening on {0}".format(self.port))
    while True:
        conn, addr = self.serverSocket.accept()
         ...
         ...
         ...
```

Deploy the code to server instance like above. Note that we are doing this for 2 servers. We can run the same code but we are connecting to 2 different server instances. When connecting servers to reverse proxy and setting up the switch table, the terminal look like this:



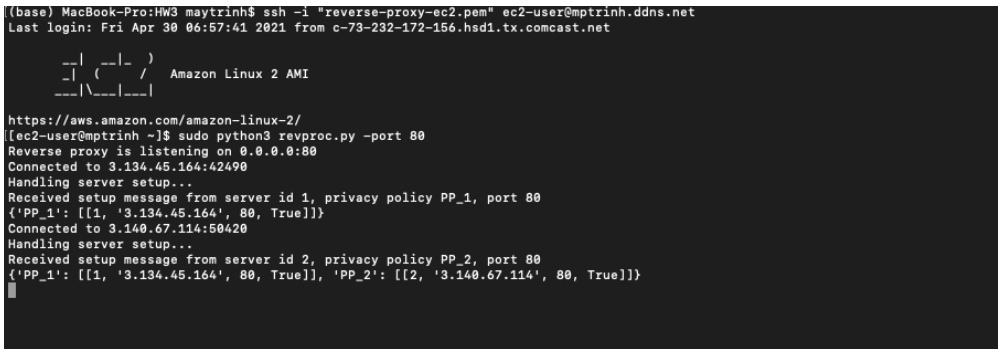
```
[ec2-user@ip-172-31-38-158 ~]$ sudo python3 server.py -id 1 -pp PP_1 -listen 80 -revproc 80
Server runing with id 1
Server serving privacy policy PP_1
Connecting to the reverse proxy on port 80
b'{"Status": "Successfully setup with reverse proxy"}'
Sever is listening on 80
```

Figure 2: Server 1 Instance



```
(base) MacBook-Pro:HW3 maytrinh$ ssh -i "server2-ec2.pem" ec2-user@ec2-3-140-67-114.us-east-2.compute.amazonaws.com
Last login: Fri Apr 30 04:07:32 2021 from c-73-232-172-156.hsd1.tx.comcast.net

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-11-134 ~]$ sudo python3 server2.py -id 2 -pp PP_2 -listen 80 -revproc 80
Server runing with id 2
Server serving privacy policy PP_2
Connecting to the reverse proxy on port 80
b'{"Status": "Successfully setup with reverse proxy"}'
Sever is listening on 80
```

Figure 3: Server 2 Instance



```
(base) MacBook-Pro:HW3 maytrinh$ ssh -i "reverse-proxy-ec2.pem" ec2-user@mptrinh.ddns.net
Last login: Fri Apr 30 06:57:41 2021 from c-73-232-172-156.hsd1.tx.comcast.net

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@mptrinh ~]$ sudo python3 revproc.py -port 80
Reverse proxy is listening on 0.0.0.0:80
Connected to 3.134.45.164:42490
Handling server setup...
Received setup message from server id 1, privacy policy PP_1, port 80
{'PP_1': [[1, '3.134.45.164', 80, True]]}
Connected to 3.140.67.114:50420
Handling server setup...
Received setup message from server id 2, privacy policy PP_2, port 80
{'PP_1': [[1, '3.134.45.164', 80, True]], 'PP_2': [[2, '3.140.67.114', 80, True]]}
```

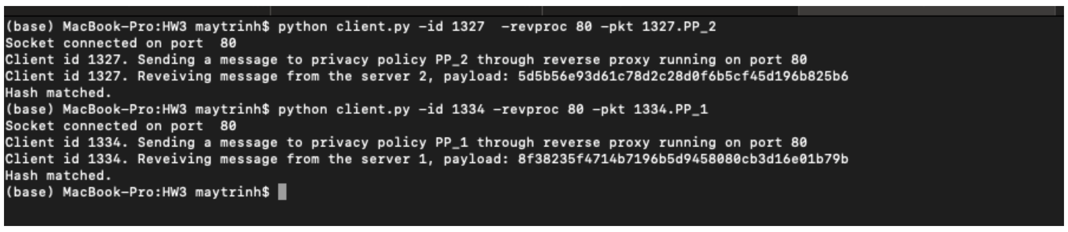
Figure 4: Reverse Proxy Instance

3 Client

Modify client code so that it connects to reverse proxy instance using its hostname.

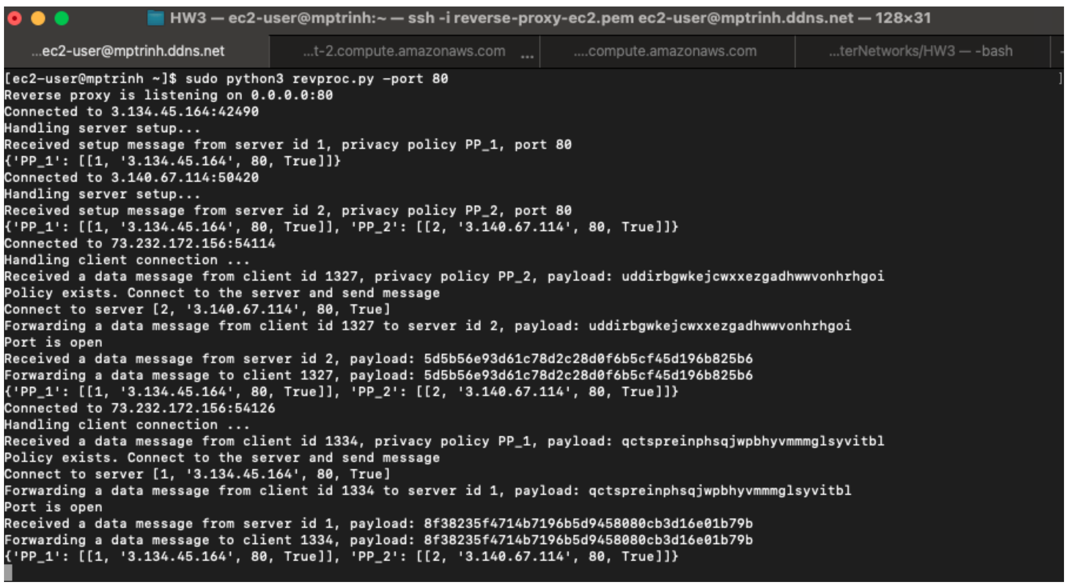
```
self.host = 'mptrinh.ddns.net' #reverse proxy hostname
```

Run the client code from local machine. The messages on the client terminal may look something like this.



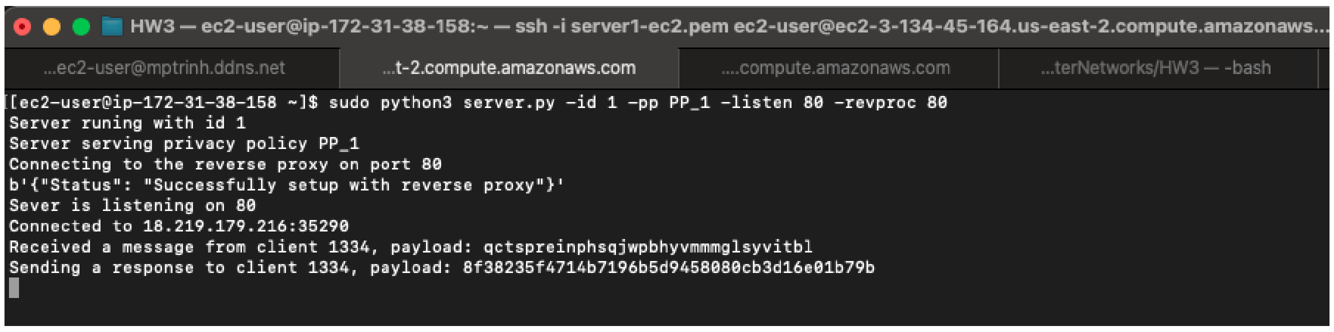
```
(base) MacBook-Pro:HW3 maytrinh$ python client.py -id 1327  -revproc 80 -pkt 1327.PP_2
Socket connected on port  80
Client id 1327. Sending a message to privacy policy PP_2 through reverse proxy running on port 80
Client id 1327. Reveiving message from the server 2, payload: 5d5b56e93d61c78d2c28d0f6b5cf45d196b825b6
Hash matched.
(base) MacBook-Pro:HW3 maytrinh$ python client.py -id 1334 -revproc 80 -pkt 1334.PP_1
Socket connected on port  80
Client id 1334. Sending a message to privacy policy PP_1 through reverse proxy running on port 80
Client id 1334. Reveiving message from the server 1, payload: 8f38235f4714b7196b5d9458080cb3d16e01b79b
Hash matched.
(base) MacBook-Pro:HW3 maytrinh$
```

The message on the Reverse Proxy terminal may look something like this.



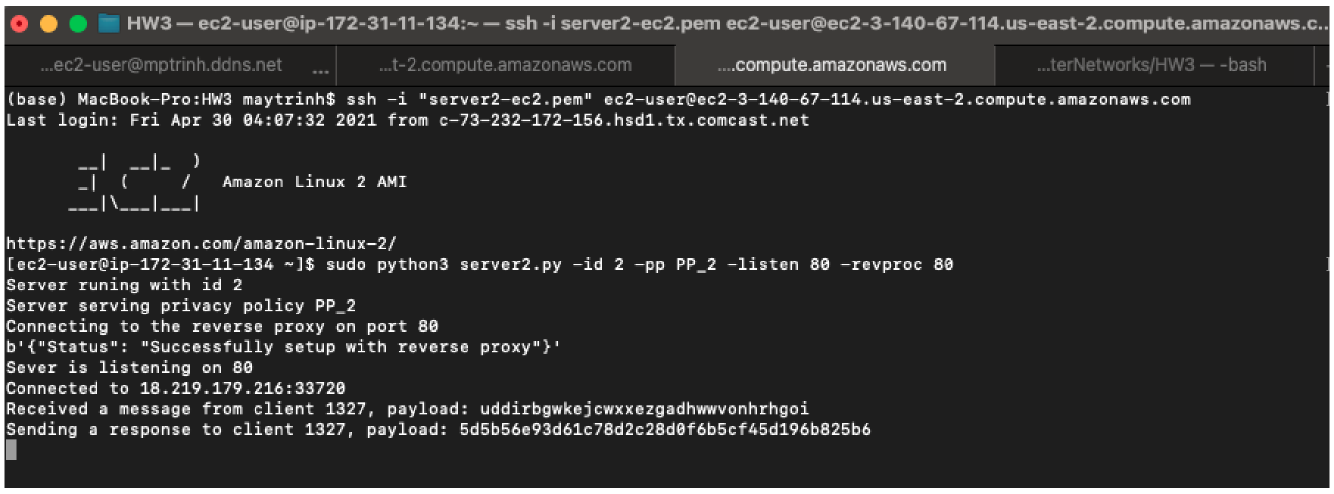
```
HW3 — ec2-user@mptrinh:~ — ssh -i reverse-proxy-ec2.pem ec2-user@mptrinh.ddns.net — 128×31
...ec2-user@mptrinh.ddns.net | ...t-2.compute.amazonaws.com | ....compute.amazonaws.com | ...terNetworks/HW3 — -bash
[ec2-user@mptrinh ~]$ sudo python3 revproc.py -port 80
Reverse proxy is listening on 0.0.0.0:80
Connected to 3.134.45.164:42490
Handling server setup...
Received setup message from server id 1, privacy policy PP_1, port 80
{'PP_1': [[1, '3.134.45.164', 80, True]]}
Connected to 3.140.67.114:50420
Handling server setup...
Received setup message from server id 2, privacy policy PP_2, port 80
{'PP_1': [[1, '3.134.45.164', 80, True]], 'PP_2': [[2, '3.140.67.114', 80, True]]}
Connected to 73.232.172.156:54114
Handling client connection ...
Received a data message from client id 1327, privacy policy PP_2, payload: uddirbgwkejcwxxezgadhwwvonhrhgoi
Policy exists. Connect to the server and send message
Connect to server [2, '3.140.67.114', 80, True]
Forwarding a data message from client id 1327 to server id 2, payload: uddirbgwkejcwxxezgadhwwvonhrhgoi
Port is open
Received a data message from server id 2, payload: 5d5b56e93d61c78d2c28d0f6b5cf45d196b825b6
Forwarding a data message to client 1327, payload: 5d5b56e93d61c78d2c28d0f6b5cf45d196b825b6
{'PP_1': [[1, '3.134.45.164', 80, True]], 'PP_2': [[2, '3.140.67.114', 80, True]]}
Connected to 73.232.172.156:54126
Handling client connection ...
Received a data message from client id 1334, privacy policy PP_1, payload: qctspreinphsqjwpbhyvmmmglsyvitbl
Policy exists. Connect to the server and send message
Connect to server [1, '3.134.45.164', 80, True]
Forwarding a data message from client id 1334 to server id 1, payload: qctspreinphsqjwpbhyvmmmglsyvitbl
Port is open
Received a data message from server id 1, payload: 8f38235f4714b7196b5d9458080cb3d16e01b79b
Forwarding a data message to client 1334, payload: 8f38235f4714b7196b5d9458080cb3d16e01b79b
{'PP_1': [[1, '3.134.45.164', 80, True]], 'PP_2': [[2, '3.140.67.114', 80, True]]}
```

The messages on the servers terminal may look something like this.


HW3 — ec2-user@ip-172-31-38-158:~ — ssh -i server1-ec2.pem ec2-user@ec2-3-134-45-164.us-east-2.compute.amazonaws...
...ec2-user@mptrinh.ddns.net
...t-2.compute.amazonaws.com
....compute.amazonaws.com
...terNetworks/HW3 — -bash

```
[ec2-user@ip-172-31-38-158 ~]$ sudo python3 server.py -id 1 -pp PP_1 -listen 80 -revproc 80
Server runing with id 1
Server serving privacy policy PP_1
Connecting to the reverse proxy on port 80
b'{"Status": "Successfully setup with reverse proxy"}'
Sever is listening on 80
Connected to 18.219.179.216:35290
Received a message from client 1334, payload: qctspreinphsqjwpbhyvmmmglsyvitbl
Sending a response to client 1334, payload: 8f38235f4714b7196b5d9458080cb3d16e01b79b
```

(a) Server 1 Instance


HW3 — ec2-user@ip-172-31-11-134:~ — ssh -i server2-ec2.pem ec2-user@ec2-3-140-67-114.us-east-2.compute.amazonaws.c...
...ec2-user@mptrinh.ddns.net
...t-2.compute.amazonaws.com
....compute.amazonaws.com
...terNetworks/HW3 — -bash

```
(base) MacBook-Pro:HW3 maytrinh$ ssh -i "server2-ec2.pem" ec2-user@ec2-3-140-67-114.us-east-2.compute.amazonaws.com
Last login: Fri Apr 30 04:07:32 2021 from c-73-232-172-156.hsd1.tx.comcast.net

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-11-134 ~]$ sudo python3 server2.py -id 2 -pp PP_2 -listen 80 -revproc 80
Server runing with id 2
Server serving privacy policy PP_2
Connecting to the reverse proxy on port 80
b'{"Status": "Successfully setup with reverse proxy"}'
Sever is listening on 80
Connected to 18.219.179.216:33720
Received a message from client 1327, payload: uddirbgwkejcwxxezgadhwwvonhrhgoi
Sending a response to client 1327, payload: 5d5b56e93d61c78d2c28d0f6b5cf45d196b825b6
```

(b) Server 2 Instance

Figure 5: Servers Instance Running