15-441/641: Recitation 1

TAs: Nirav Atre, Zili Meng, Hugo Sadok

09/02/2022

1 Setting up an AWS Instance

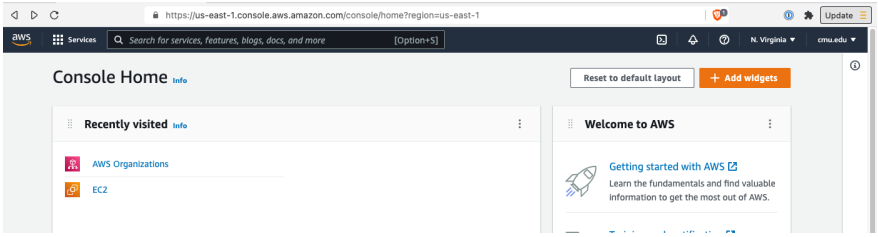
To start, you need an Amazon Web Services (AWS) Account. You should have received an email yesterday with an invitation to create an AWS account.

> **Warning:** A lot of stuff you do on AWS will be charged. Be mindful of launching 100 instances, it's cool but expensive. You are also charged depending on the duration, so leaving a machine running indefinitely will also cost you more credits.

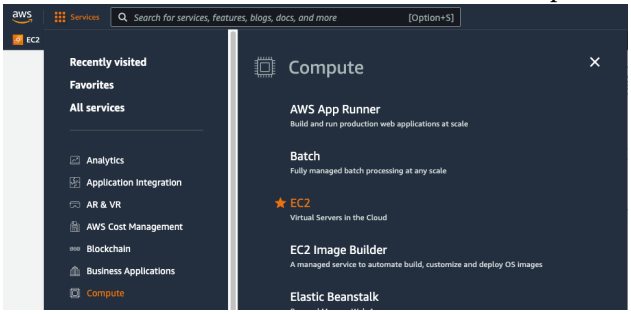
1.1 Launching Instances

AWS allows you to create instances (AWS nomenclature for virtual machines) in many different regions in the world. You may go to `https://aws.amazon.com/about-aws/global-infrastructure/regions_az/` to check the list of AWS regions around the world. Popular Internet services often have instances running in different regions to ensure that different markets can use the services with low latency.

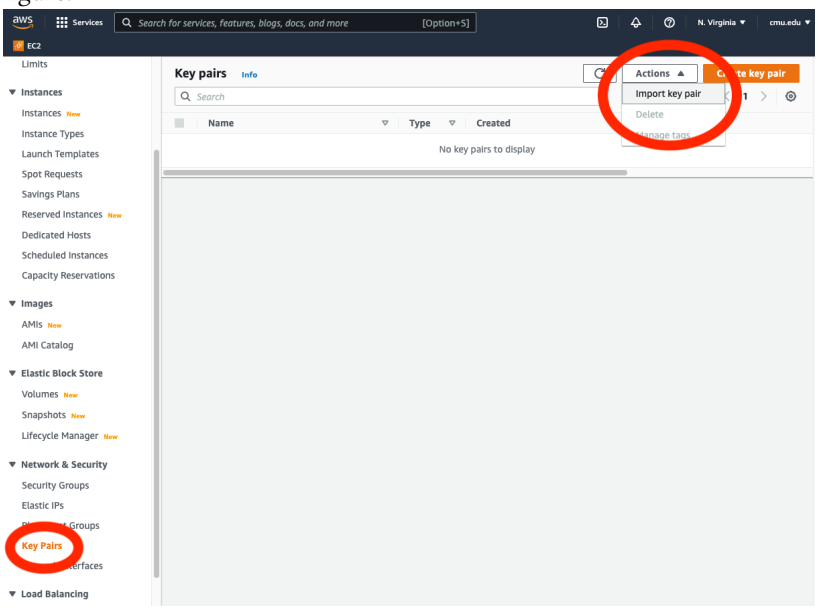
AWS Console. To launch an instance go to your AWS console at `https://us-east-1.console.aws.amazon.com/`. Note that we are accessing the console for the `us-east-1` region. Every region has a separate AWS console and **you need to repeat the process below for every region where you want to launch an instance.** Here is how the AWS console looks like (note that that the region name "N. Virginia" is displayed in the top right corner):



To launch an instance, click on "Services" in the top left corner and select "Compute" and then "EC2."

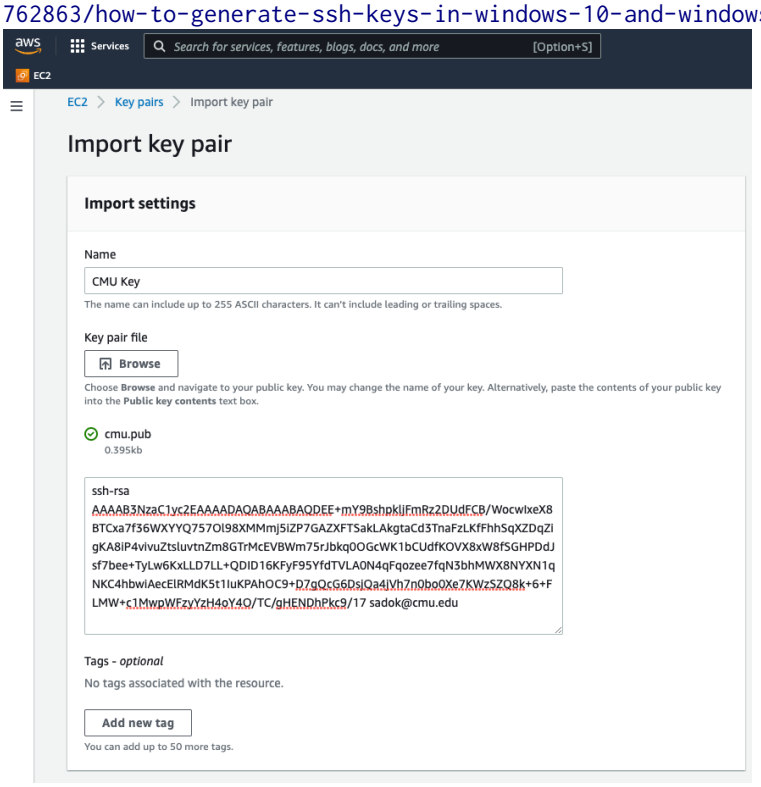


Key pair. Before launching an instance, first make sure you have an SSH key set up. Go to "Key Pairs" (under "Network & Security" in the left side bar). You can either click on "Create key pair" to have AWS create an SSH key for you or you can add an existing SSH key by clicking on "Actions" and then "Import key pair" as shown in the next figure.



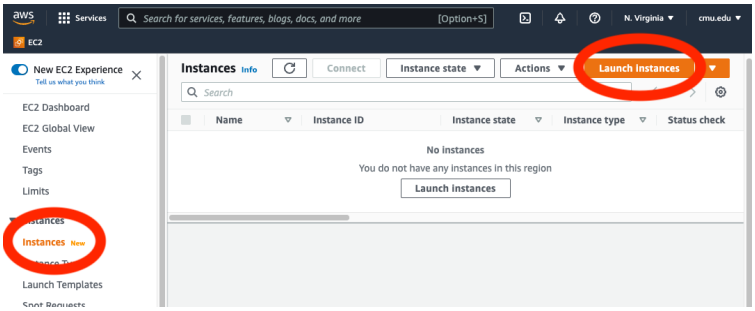
Then, you can either select a file with your public SSH key or paste it in the text box. To finish importing, click on "Import key pair."

Here is also a brief tutorial on how to generate the key pair on your own laptop: https://www.howtogeek.com/762863/how-to-generate-ssh-keys-in-windows-10-and-windows-11/.

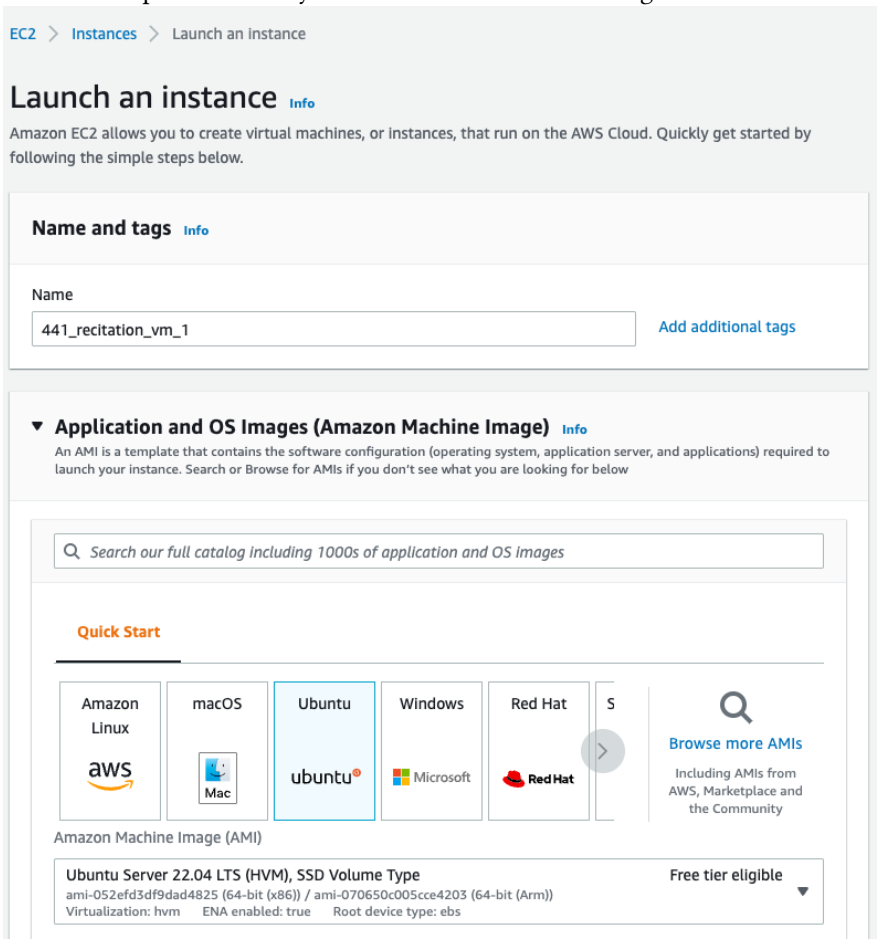


If you create a new key pair, you need to save it to somewhere in your laptop for your future use (later following the connecting to the instance part).

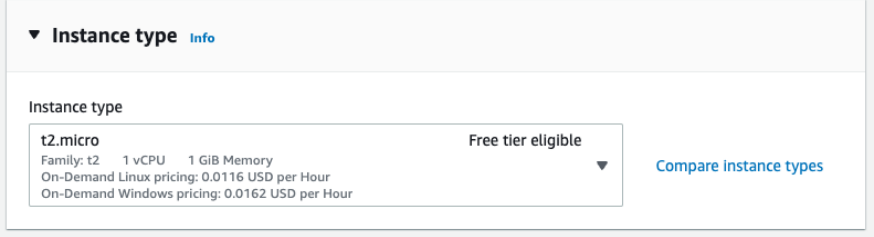
Launching an instance. You are now ready to launch instances. Go to the "Instances" panel in the left side panel and click on "Launch instances."



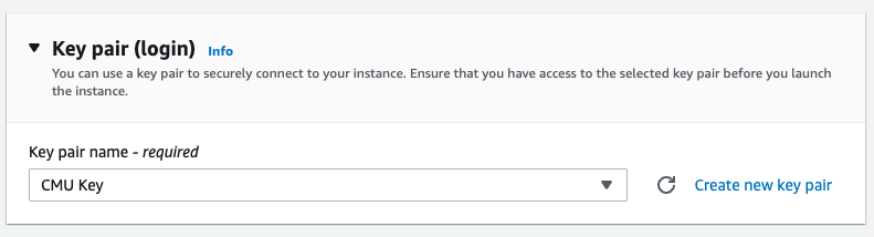
Give a descriptive name to your instance and select an image. We recommend using "Ubuntu Server 22.04 LTS."



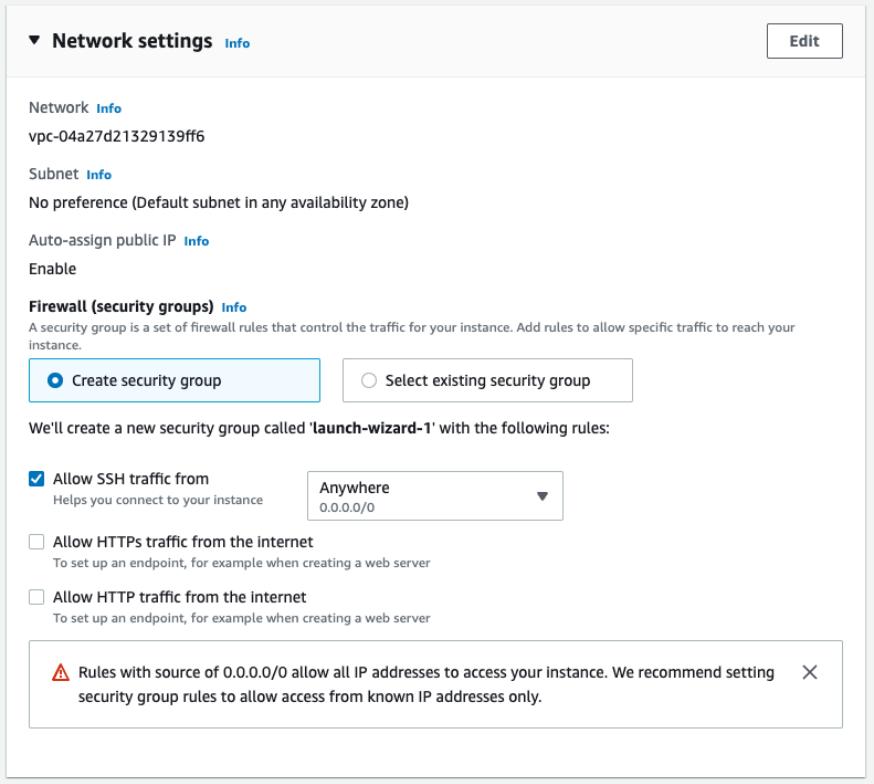
Select an instance type. The instance type determines the amount of resources that will be available to your instance (e.g., CPU cores and memory). **Please use a free-tier elegible instance (e.g., `t2.micro` or `t2.nano`) to prevent overcharging**.



Select the key that you created in the previous step.

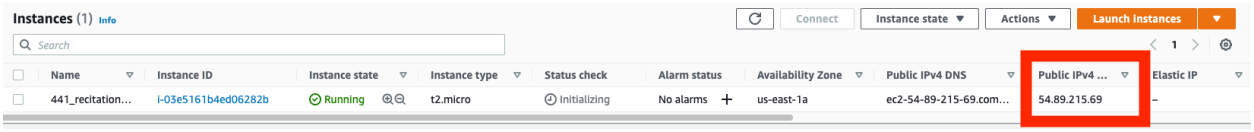


Now we will configure the VM firewall rules. We are going to create a new "security group" and allow SSH traffic from anywhere (`0.0.0.0/0`). This new security group will block traffic to all ports but 22 (the port used by SSH), we will open some ports in a later step.

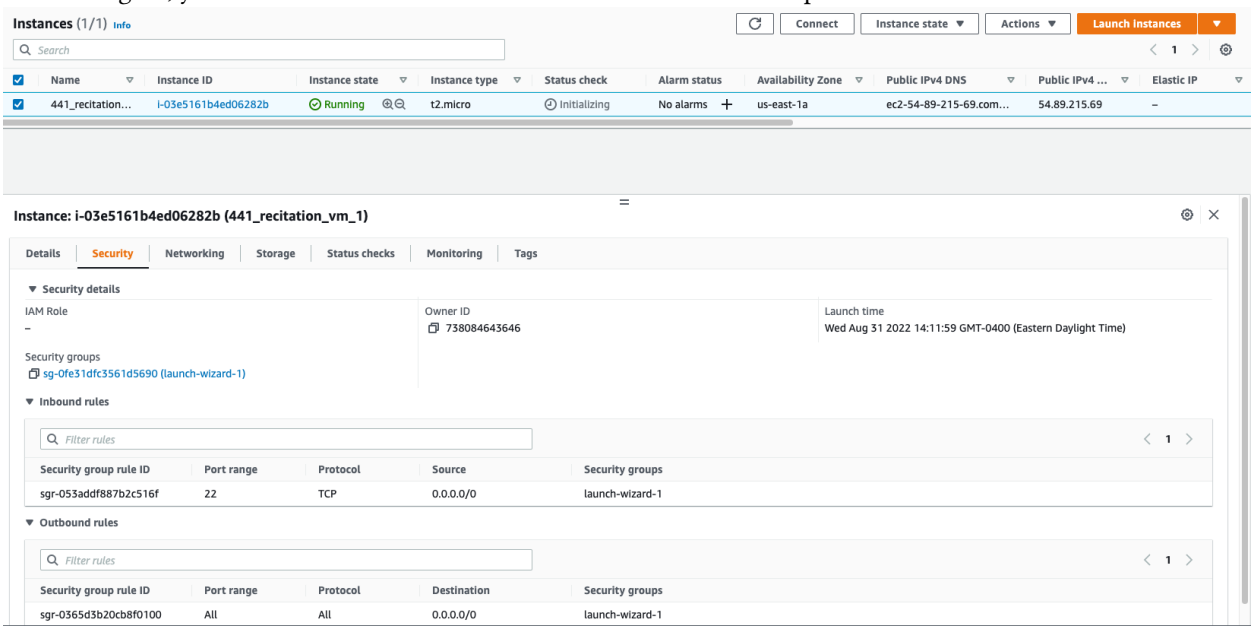


You can leave all the other options unchanged and click on "Launch instance" in the right panel.

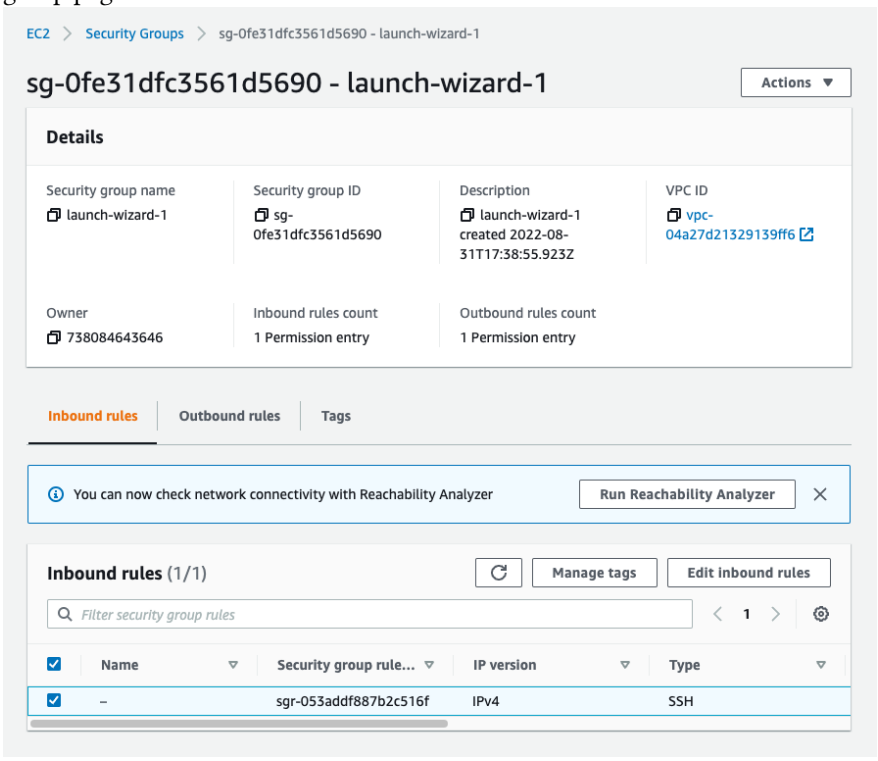
Now if you go back to the Instances panel, you should see your instance listed. It may take a little for it to finish booting. You can also see your instance's public IP. You will use this IP to access your instance using SSH. You may try this now.



Configuring the security group. On AWS a security group is responsible for describing the firewall rules that apply to a given instance. Firewall rules determine which traffic is allowed to enter or leave an instance. For example, when creating the instance, we allowed all SSH traffic, therefore the instance's security group should already include rules to allow inbound SSH traffic. If you click your instance and go to the Security panel in the bottom as shown in the next Figure, you will see that it has an inbound rule to allow traffic to port 22.



We will now add some extra firewall rules to allow our experiments to go through. Start by clicking on the security group (`sg-0fe31dfc3561d5690 (launch-wizard-1)` in the example above). You will be directed to the security group page as shown below. Click on "Edit inbound rules."

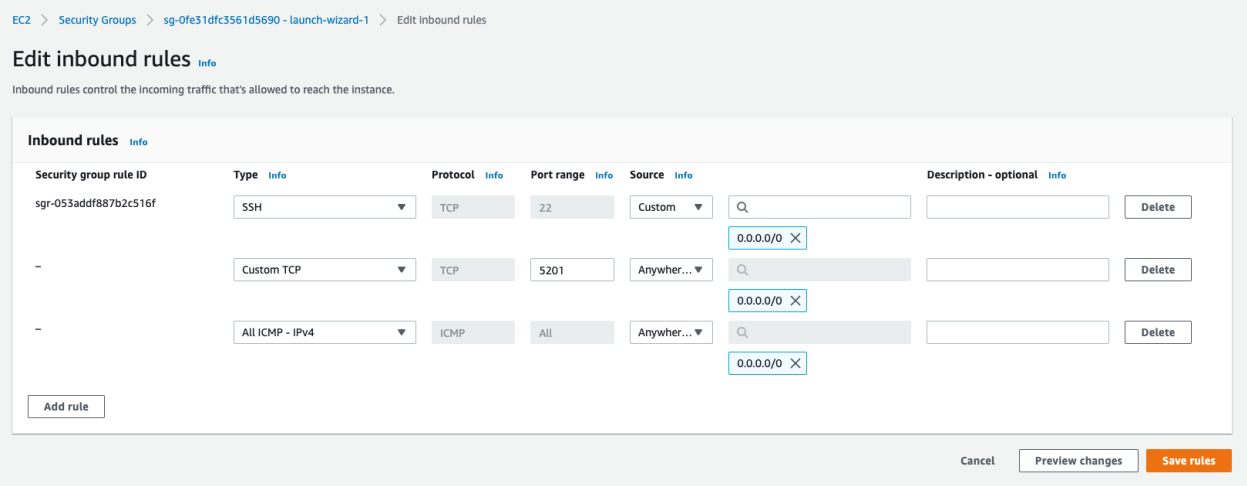


- Click on "Add rule" to add a new rule to allow `iperf3` traffic. Select "Custom TCP" as the Type, "5201" as the

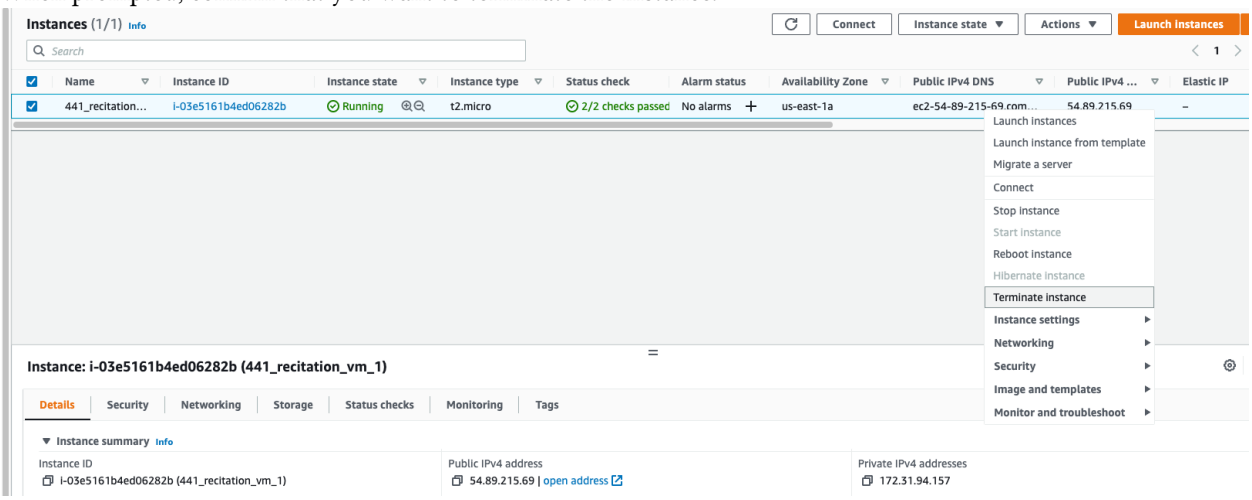
Port range, and “Anywhere-IPv4” as the Source. Note that this will add `0.0.0.0/0` as the source, as we will see later in the class this is a CIDR notation to represent all IPv4 addresses.

- Click again on “Add rule” to allow ICMP traffic (we need to allow ICMP in order to allow the instances to be pinged). Select “All ICMP - IPv4” as the Type and “Anywhere-IPv4” as the Source.

After adding these rules, it should look as follows. Click on “Save rules” to apply your changes.



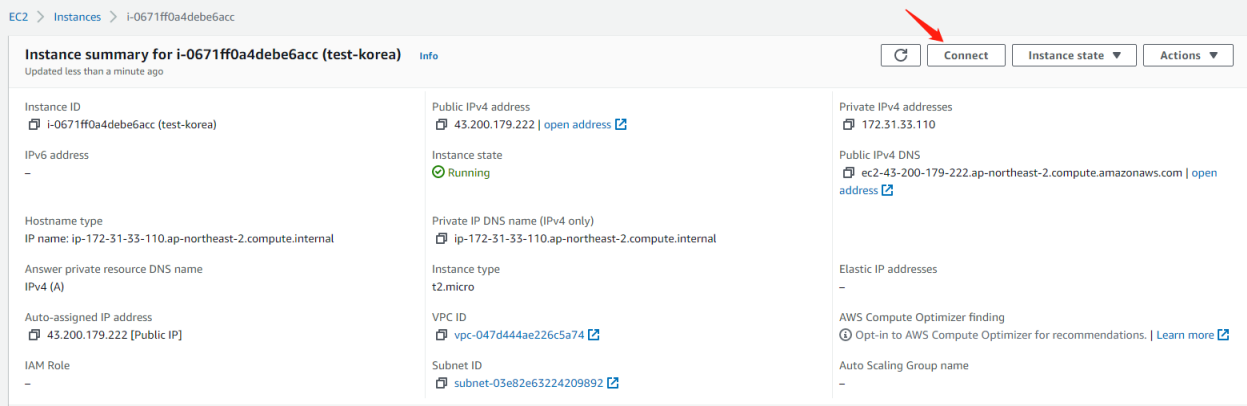
Terminating an instance. Make sure to terminate your instance after you are done. Otherwise the instance will continue to run and drain your credits. To do so, you may right click the instance and select “Terminate instance.” When prompted, confirm that you want to terminate the instance.



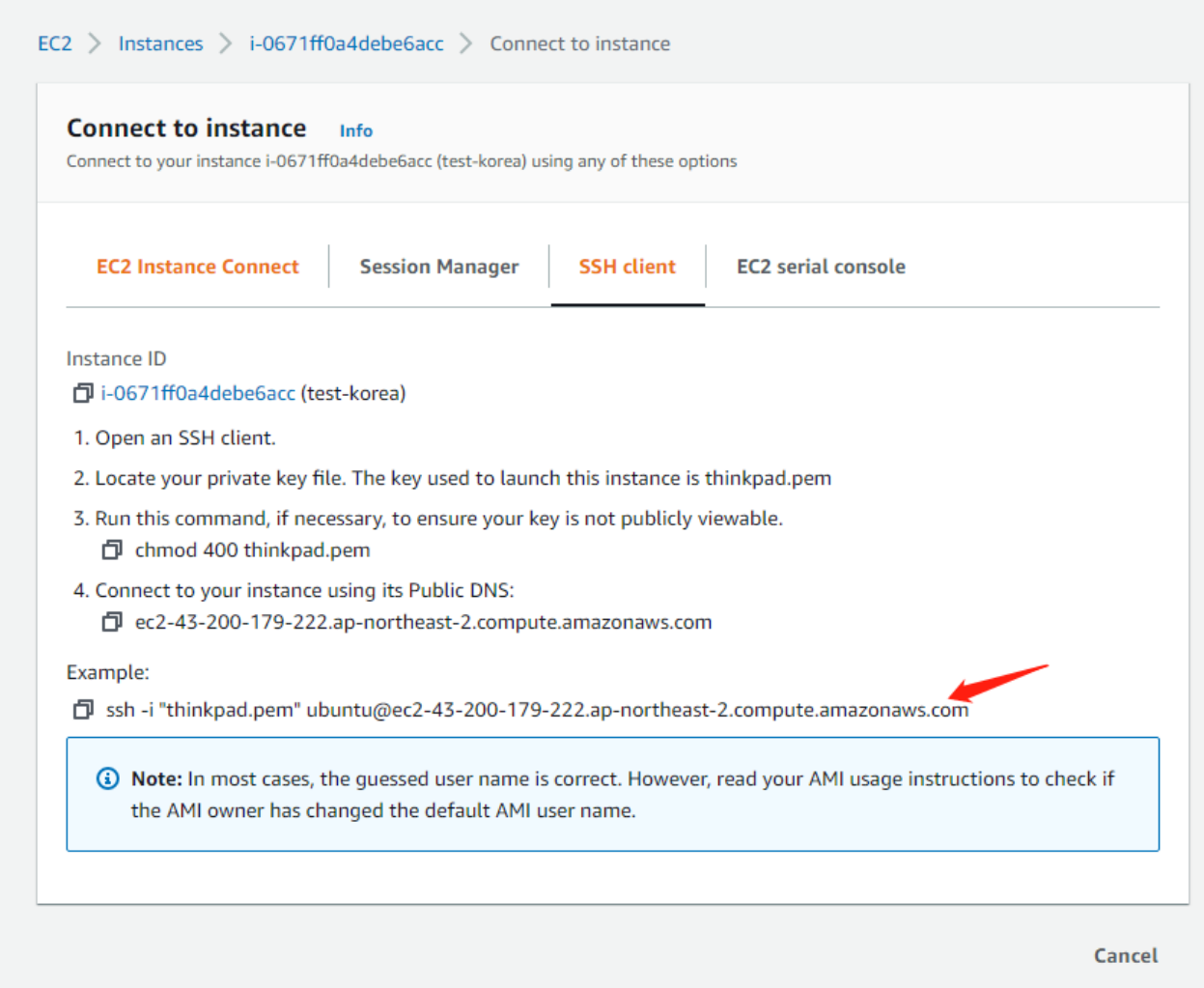
2 Running Experiments

2.1 Connecting to the instance.

Find the public IP address for your instance. On the same page from the last step, click "Connect":



Under the tab of **SSH Client**, find the command to connect to your instance:

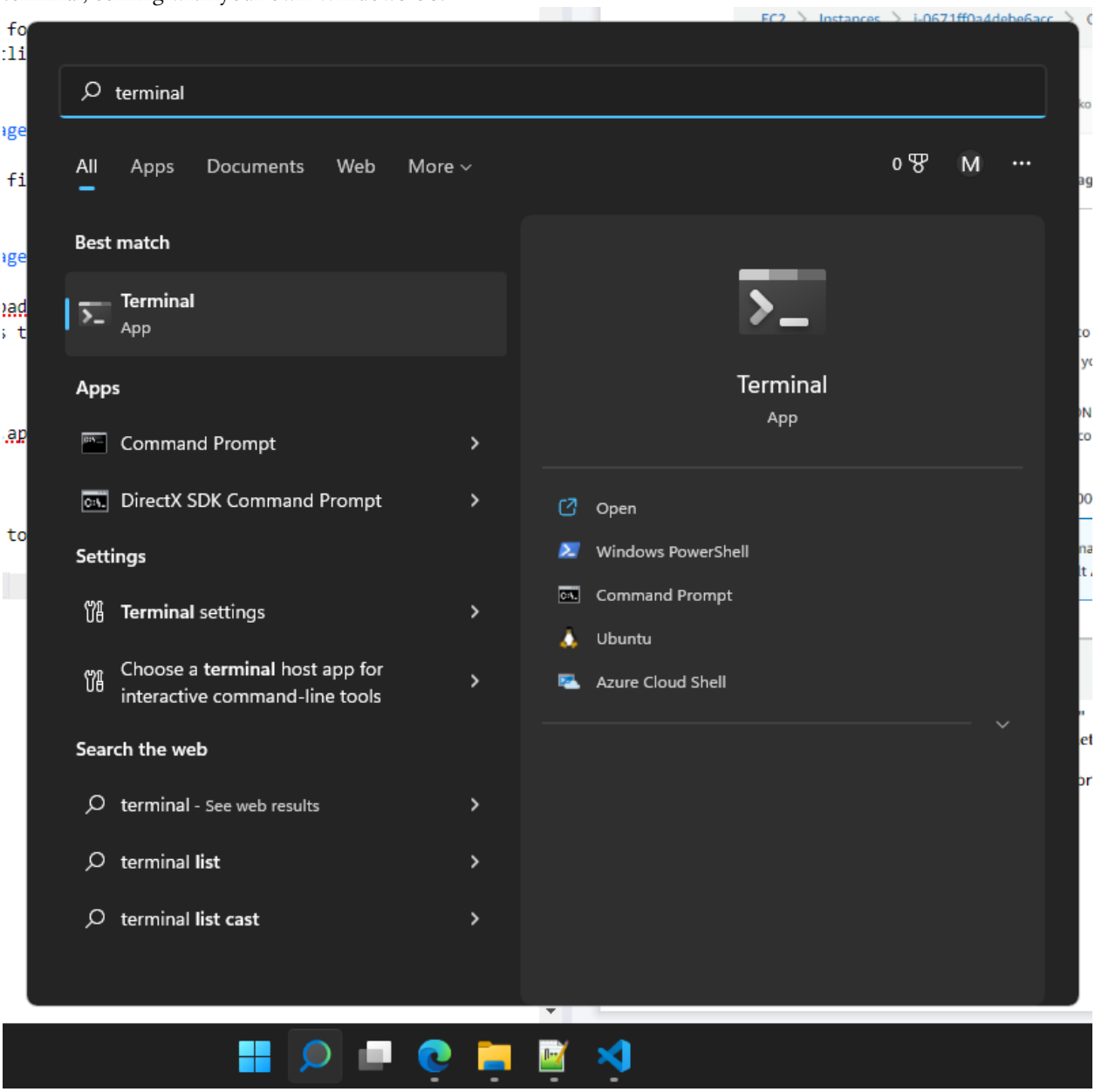


The key parameter `-i "thinkpad.pem"` is the file that you saved in the key pair step. If you upload your own keys to AWS, you may not need the parameter, so the command may be like:

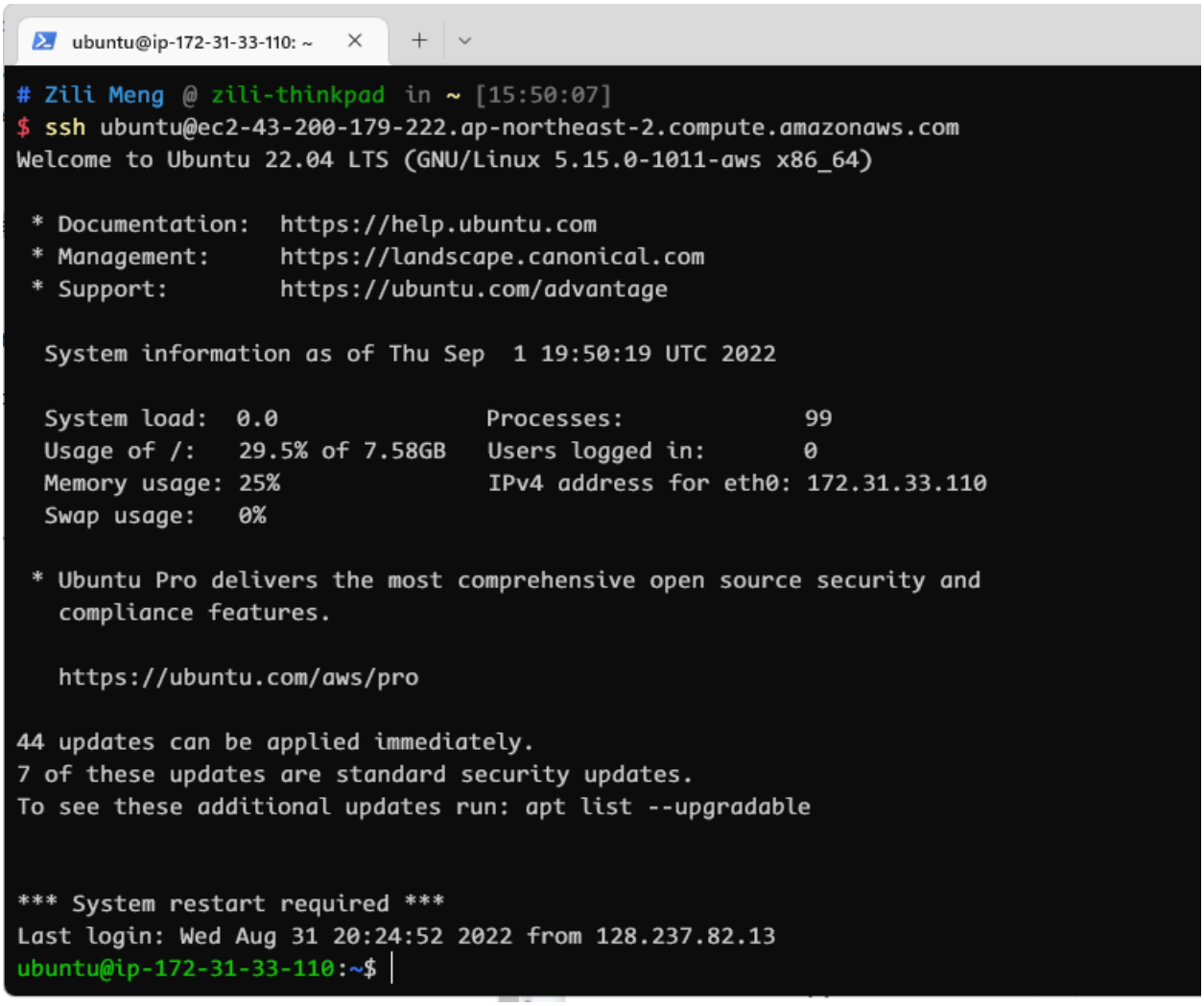
```
ssh ubuntu@ec2-43-200-179-222.ap-northeast-2.compute.amazonaws.com
```

Connect to your instance. Now you need to use your own terminal to connect to the AWS instance. First you need to find the terminal on your own laptop. For example, for Windows users, we recommend you to use the Windows

terminal, coming with your own Windows OS.



Then, type the command starting with `ssh` you copy from the last step. You will have the access to your own instance now!



Note that for P0, you may need to create multiple instances.

2.2 Using `ping`

Using `ping` is relatively easy. `ping` comes with the operating system, so just type `ping` and the IP address or the domain that you want to `ping`, like:

```
ping google.com
ping 8.8.8.8
```

Now you will see the round-trip time (RTT) from the instance where you initiate the ping and the host that you are pinging.

```
ubuntu@ip-172-31-33-110:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=105 time=30.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=105 time=30.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=105 time=30.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=105 time=30.4 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=105 time=30.5 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=105 time=30.5 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=105 time=30.4 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=105 time=30.5 ms
^C
--- 8.8.8.8 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7012ms
rtt min/avg/max/mdev = 30.422/30.469/30.523/0.029 ms
```

From the results, you can see that your instance will try to `ping` the host every second. When you terminate the program, it will report the average and some other statistics of the RTTs. You can try to explore other usages by running `ping --help`.

Note that default `ping` on Ubuntu will not terminate unless you manually terminate it with Ctrl+C.

2.3 Using `iperf3`

`iperf3` requires you to run commands from both sides. If you do not yet have two instances, repeat the process in §1.1 to launch another instance.

On each instance, first you need to install `iperf3` by:

```
sudo apt update && sudo apt install iperf3
```

In the following you need to have different actions on two instances. On **Instance 1**, run:

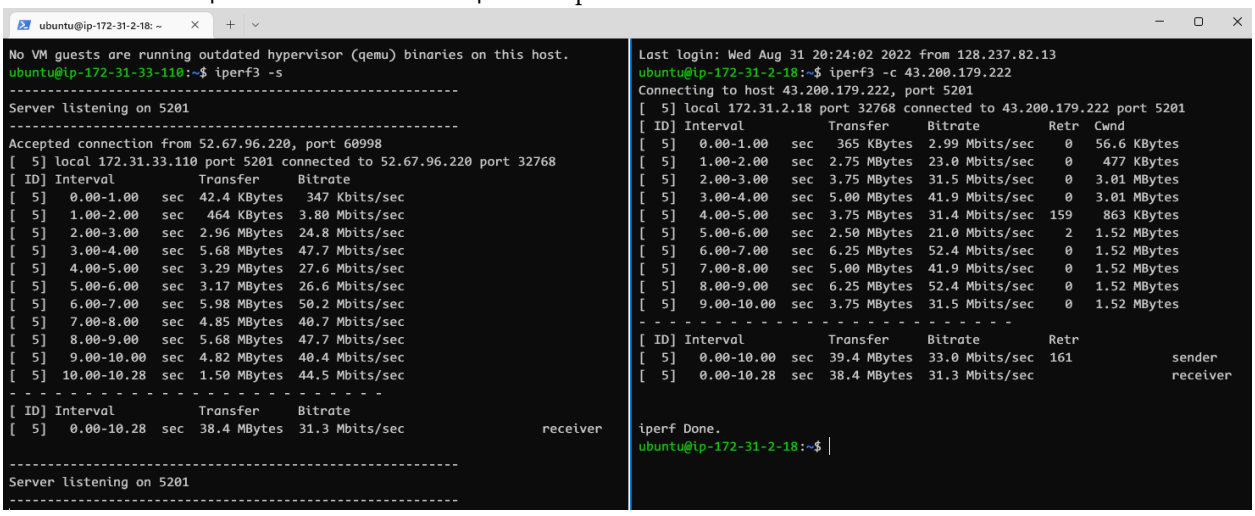
```
iperf3 -s
```

This starts an `iperf3` server.

On **Instance 2**, run:

```
iperf3 -c <server-ip>
```

This starts an `iperf3` client. `<server-ip>` is the public IP address of the **Instance 1**.



You've got it! Now you can see the bandwidth measurement results.

You can try to explore other usages by running `iperf3 --help` (e.g., for UDP traffic).

Now try on the P0! You almost have already had everything set up for P0.