**TACACS & RADIUS ON AZURE & AWS CLOUD**

Juntao Zhong

ccnp (cisco certified network professional)

DEC. 2018

NEWPORT HIGH SCHOOL, WA

## Purpose

Configure Linux TACACS+ server and Windows RADIUS server on Windows Azure Cloud and on Amazon AWS Cloud (two servers on both clouds, so a total of four servers). Then, bridge local routers into internet and authenticate with AAA protocols.

## Backgroud:

According to Forbes, 83% of enterprise workloads will be in cloud by 2020. Cloud computing has been the trend for enterprise of all sizes.

Compared to traditional servers, cloud servers can be easily scaled. Upgrading/ downgrading the performance and storage of my servers is as simple as a click. In the long run, cloud saves companies money because it reduces expensive hardware purchase and does not require time-consuming software installations.

Cloud is also more secure. Cloud companies like Microsoft and Amazon can implement high-level encryption and routine backups on a large scale. Some problems with hardware: such as a stolen laptop or a virus on PC won’t endanger the data of your company.

After all, Cloud allows users to access the files from wherever whenever, thus, making people’s work more efficient.

TACACS+ and RADIUS are both AAA (Authentication, Authorization and Accounting) protocols that centralizes network device management into one server, expect TACACS+ is newer and more advanced than RADIUS. They both enable centralized authorization of network devices, which increases the security of the network.

## Lab summary

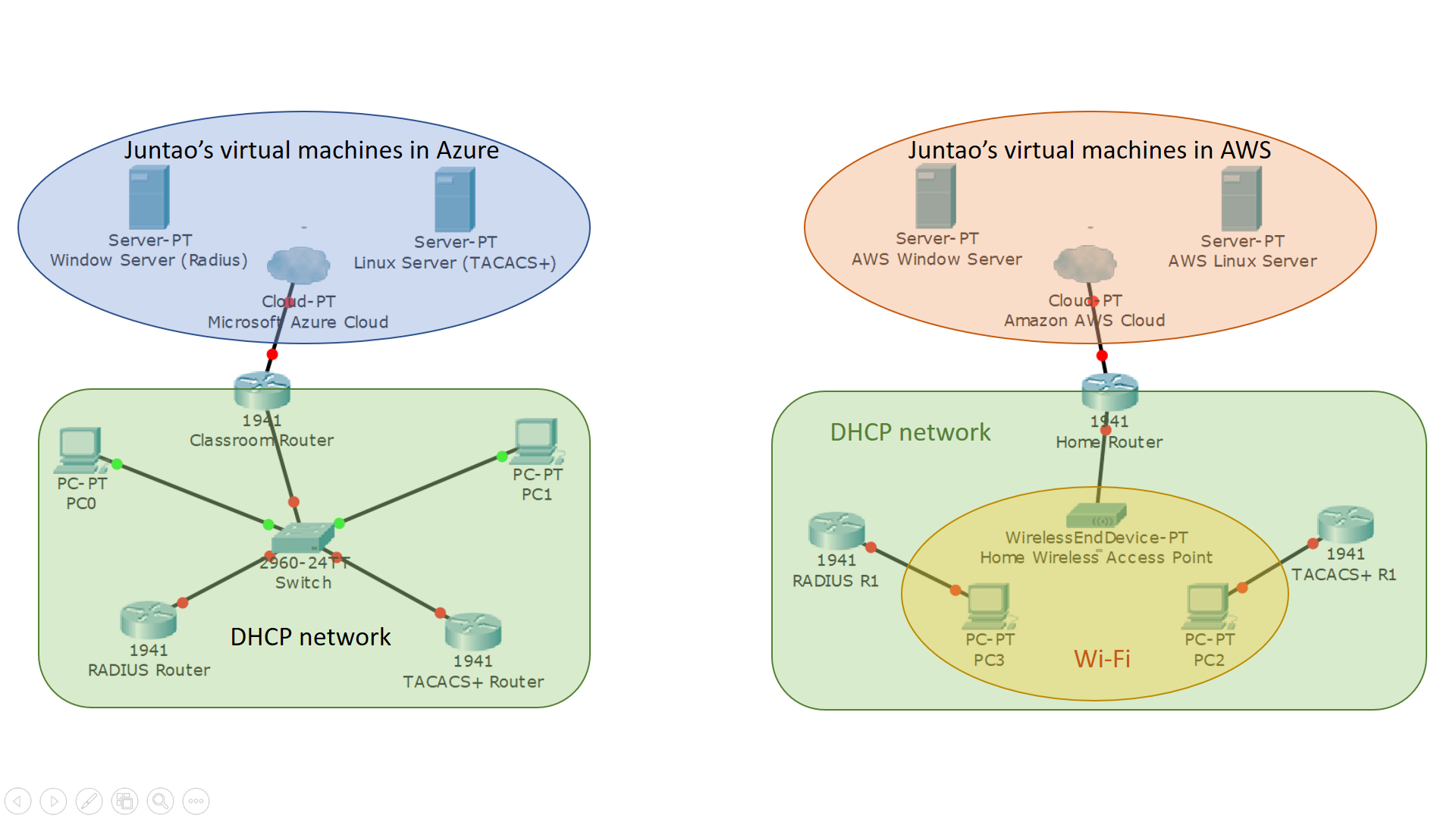
In this lab, I registered an educational account with my “.edu” email from Bellevue College. An educational email has the same function as a professional one, except it already preloaded some money to help start my servers.

I used Ubuntu 16.04 (AWS)/18.04 (Azure) LTS operating system to run TACACS+, and Windows Server 2016 to run RADIUS. I did all my Linux configuration by SSH into my VM’s serial console; and I used RDP (Remote Desktop) to configure Windows Server VM.

The routers at my lab get internet access through bridging. First, my PC connects to the internet through Wi-Fi, just like most home internet devices. Second, I connect my PC with a router through Ethernet cable. Then, I bridge two connections of my PC. In this way, the router wired to my PC becomes a part of the lab internet, and it gets an assigned private DHCP address. No IP addresses was manually configured, all automatically assigned.

After bridging my routers to the internet, setting up AAA services on cloud servers is pretty similar to that on local servers. However, one of the critical step is to allow (open) the AAA service related ports on my VMs.

## Lab topology

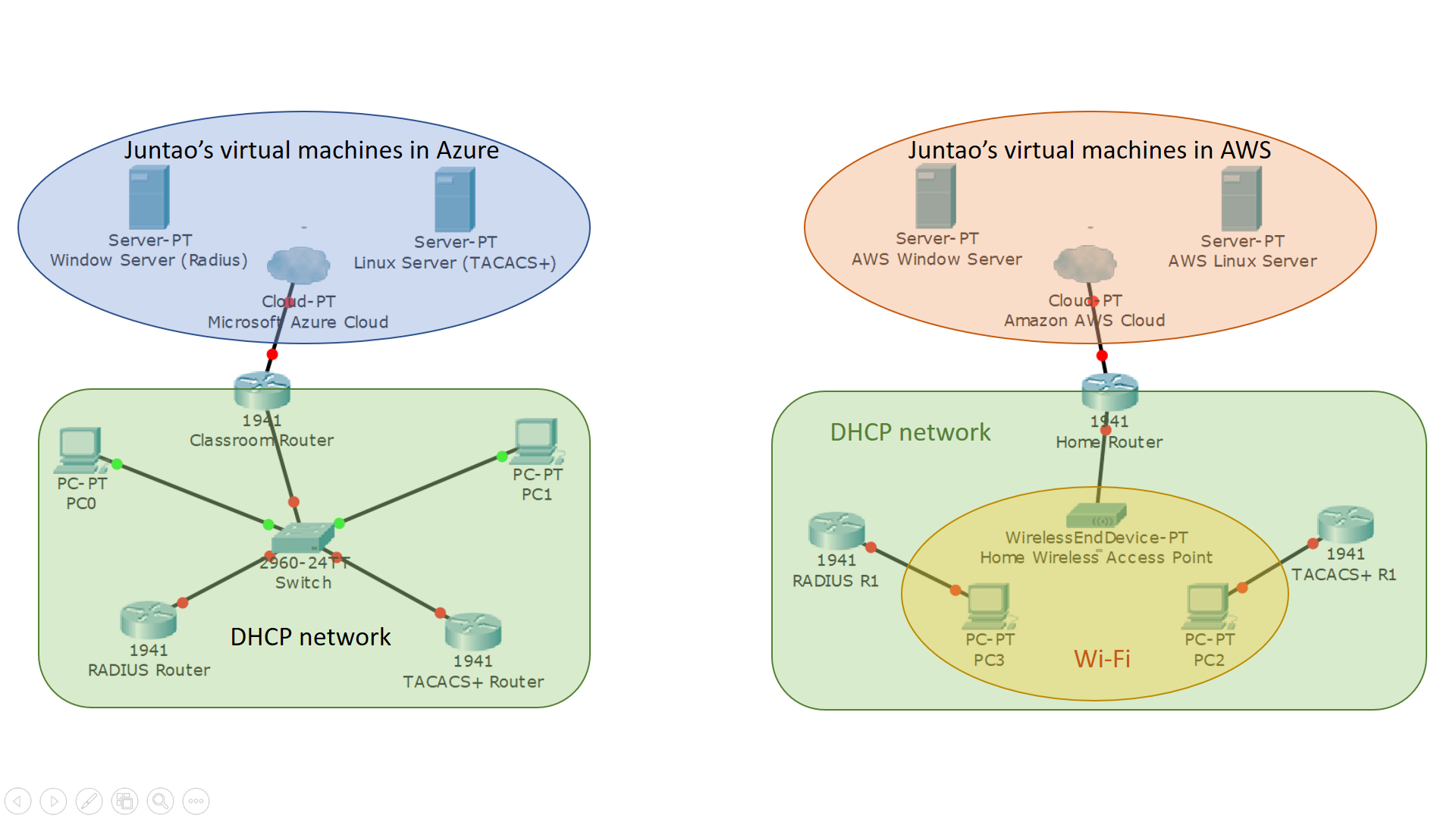


* The servers shown here are logical Virtual Machines (VMs) in cloud.
* Operations system (Linux and Windows) are provided by the cloud companies.

**Original Plan**: Connecting routers to the Internet through PCs.

Routers can’t receive Wi-Fi, so it can only get internet by bridging Wi-Fi connection of PCs.

DHCP network is the network of my lab classroom.



**Alternative Plan**: Connecting routers through a switch.

In my lab network, bridging failed when too many devices (too many students) tried to bridge connections.

So, I connected a LAN port on the classroom router to the switch, and I plugged my PCs and routers into the switch with Ethernet cables.

Without configuring the switch, this method yielded the same result: my lab routers got private DHCP addresses from the Classroom Router.

## Lab commands

Same as lab “Set Up TACACS+ & RADIUS Servers”.

## configurating of routers

### TACACS+ Routers configuration:

**Router(config)#** **hostname R1**

enable password cisco

aaa new-model

aaa authentication login default group tacacs+ local

aaa authorization exec default group tacacs+ if-authenticated

no ip domain lookup

username backup password 0 cisco

interface GigabitEthernet0/1

ip address dhcp

no shutdown

ip tacacs source-interface GigabitEthernet0/1

tacacs-server host 40.87.95.77 key testing123

**Note:** This configuration works for TACACS+ routers connected to both Azure and AWS servers. However, adjust “tacacs-server host x.x.x.x” when servers are power cycled or when switch from Azure to AWS Linux server.

### RADIUS Routers Configuration:

**Router(config)#** **hostname R1**

### enable password cisco

### aaa new-model

### aaa authentication login default group radius local

### aaa authorization exec default group radius if-authenticated

### Username backup password cisco

### aaa session-id common

### interface GigabitEthernet0/1

### ip address dhcp

### no shutdown

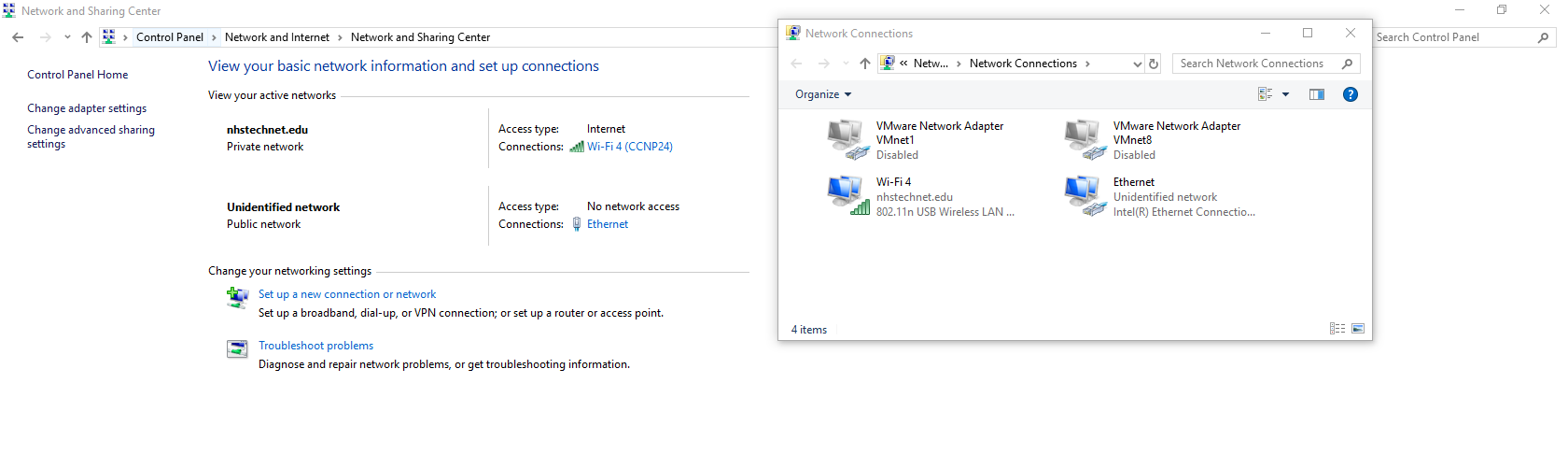
### radius-server host 18.222.211.109 key juntao

**Note:** For the RADIUS router configuration below, these commands also apply to both routers, except that remember to adjust the IP address in “radius-server host x.x.x.x”.

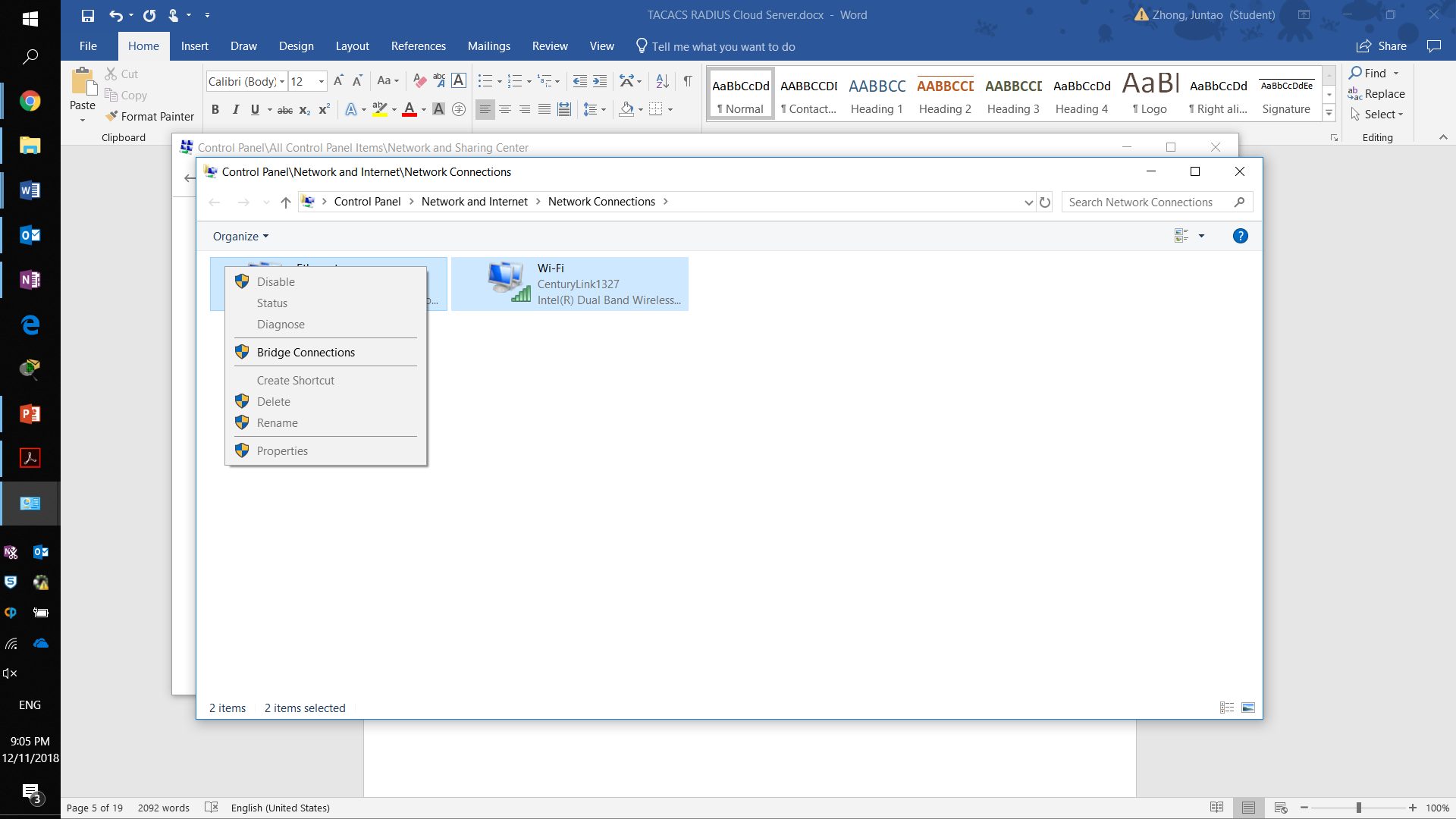
## giving local routers internet access through bridging

### bridging with a pc:

The PC should have normal Wi-Fi connection. At the same time, PC’s Ethernet port is connected to a local router that needs RADIUS/TACACS+ authentication.



Select wireless network and ethernet interface. Right click and select “Bridge Connections”.

A new image please

And two network interfaces are now bridged. A Network Bridge (red box) should show up; the connection should look like a ethernet connection (yellow box).



If there is an error messages, don’t panic, click into the bridge and make sure both wireless and Ethernet are in the bridge.

An image needed

### If bridging does not work:

Use the second topology and connect every device with a switch.

If you are confident about all the configurations, directly connecting internet access router and local router with an ethernet cable will also work. PCs can get internet through its separated wireless access point.

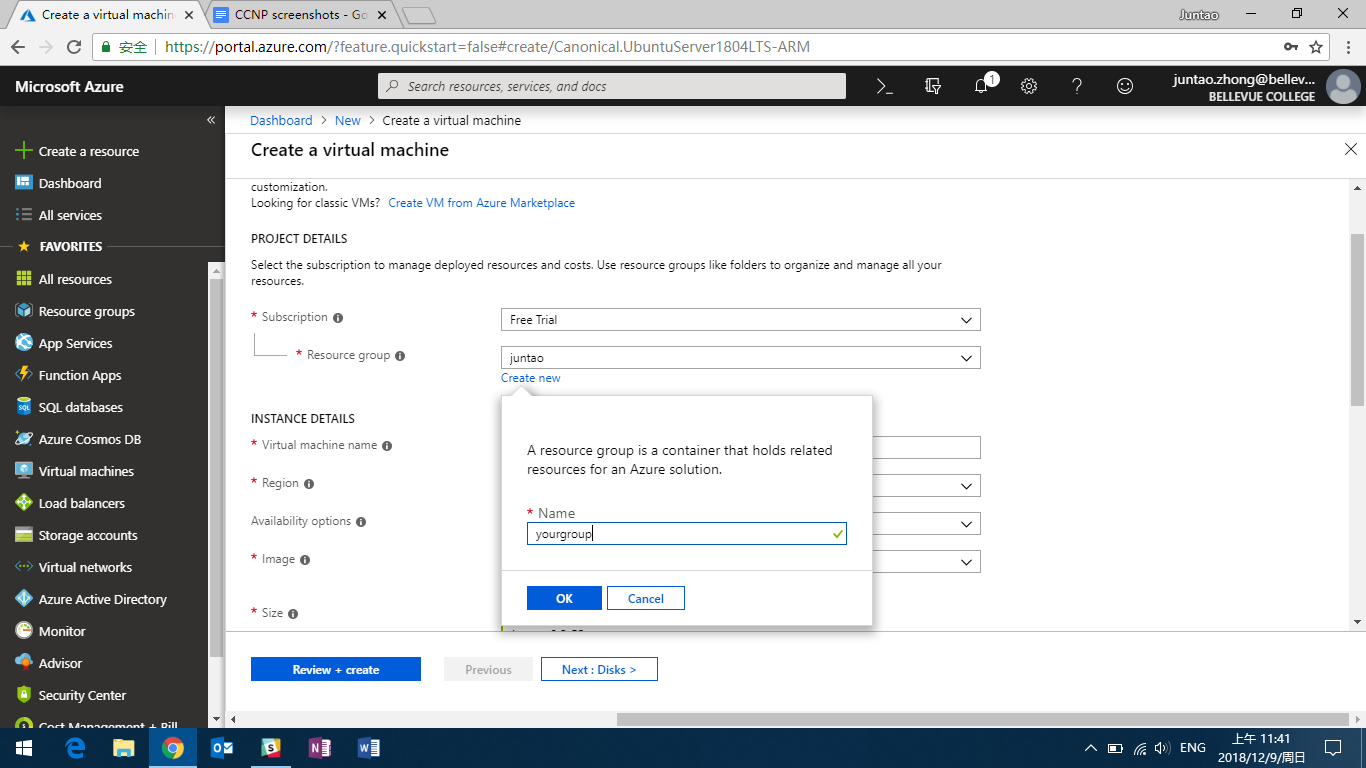
However, for the purpose of this lab, I used PC to bridge whatever possible.

## setting up cloud virtual machines

### Azure linux vm setup:

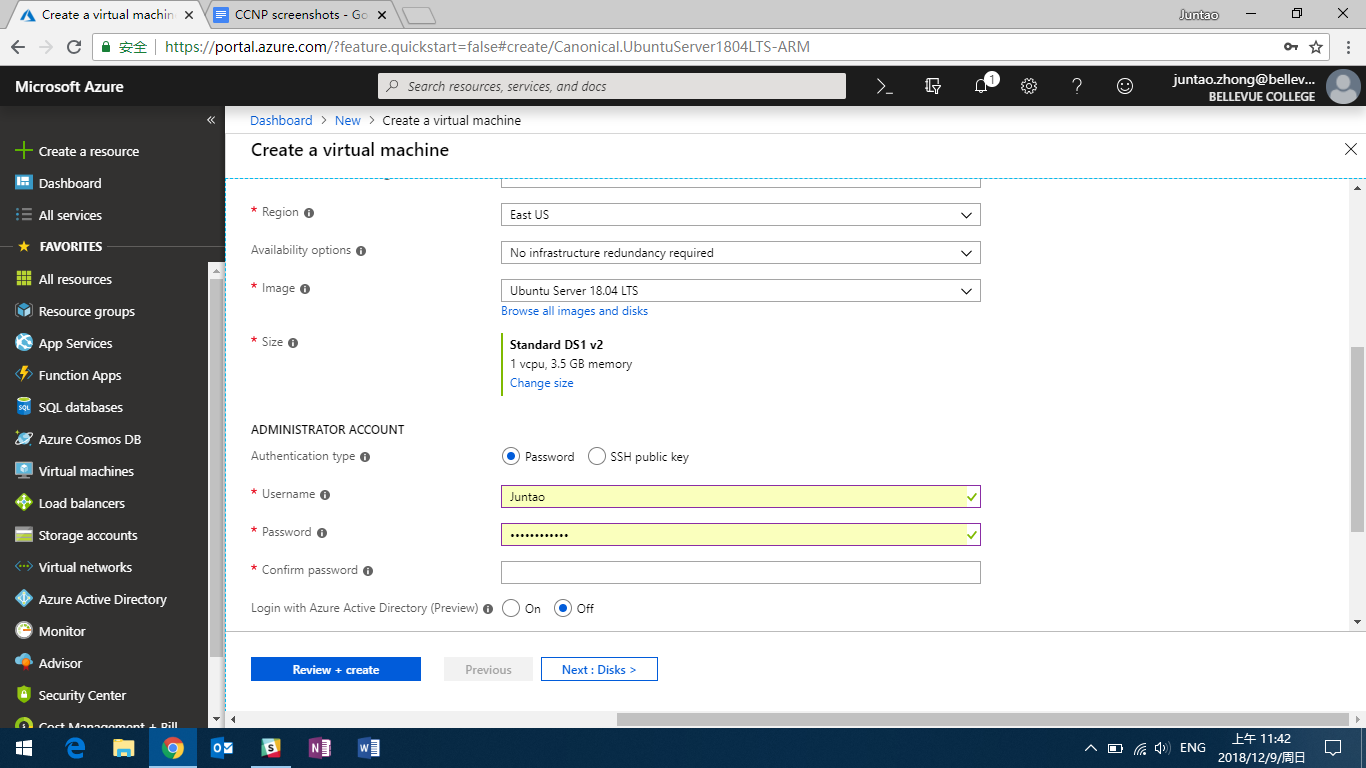
In Azure portal, click “Create a resource” (yellow box). Then select the operating system of your choice (not shown, in this case, Ubuntu 18.04 LTS server). After that the portal brings you into a place to manipulate the basic settings of your machine.

It’s asking for a name for the Resource Group (like the drive that all your programs reside). Type in whatever name you like; it will not appear again.

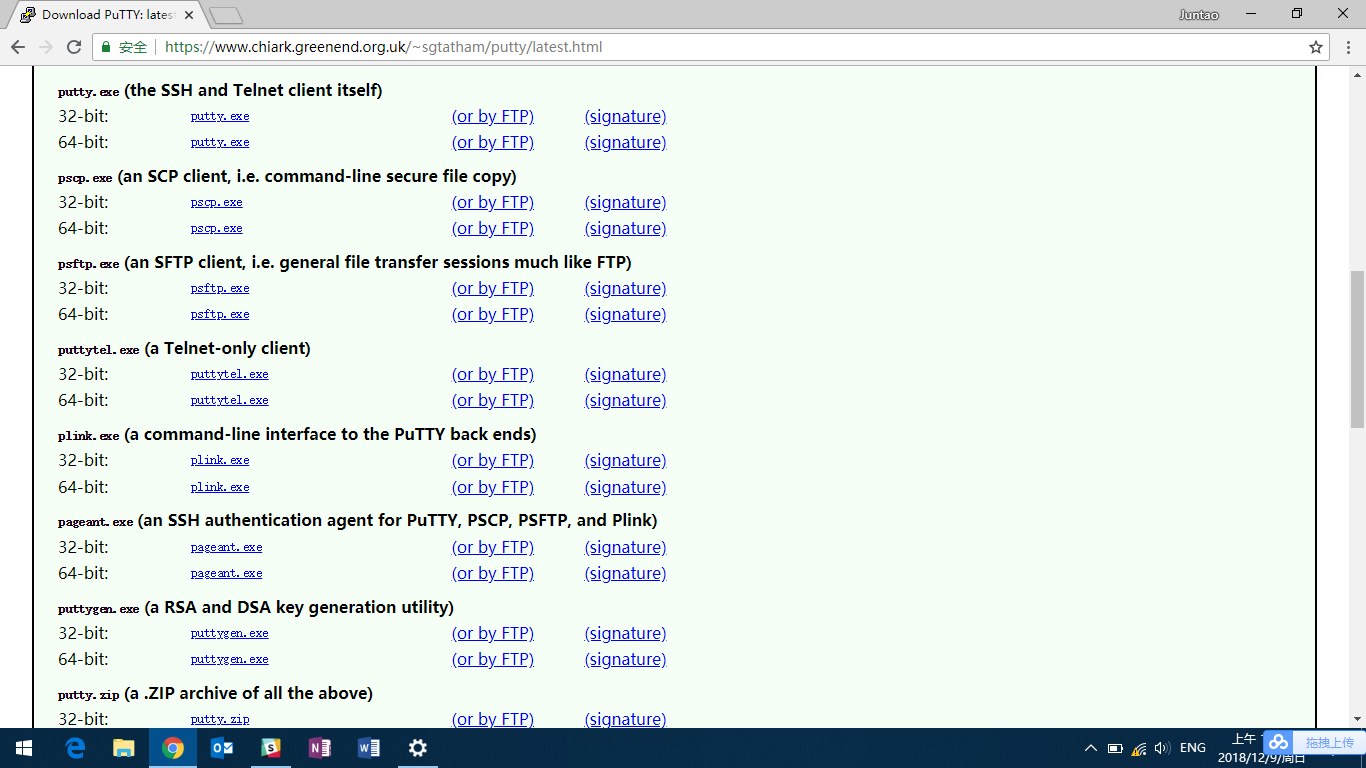


Next, the portal asks for the size/capacity you want for the Virtual Machine. TACACS+ requires very little resource (it’s simply a username + password matching program), so I choose the lowest capacity available to save money.

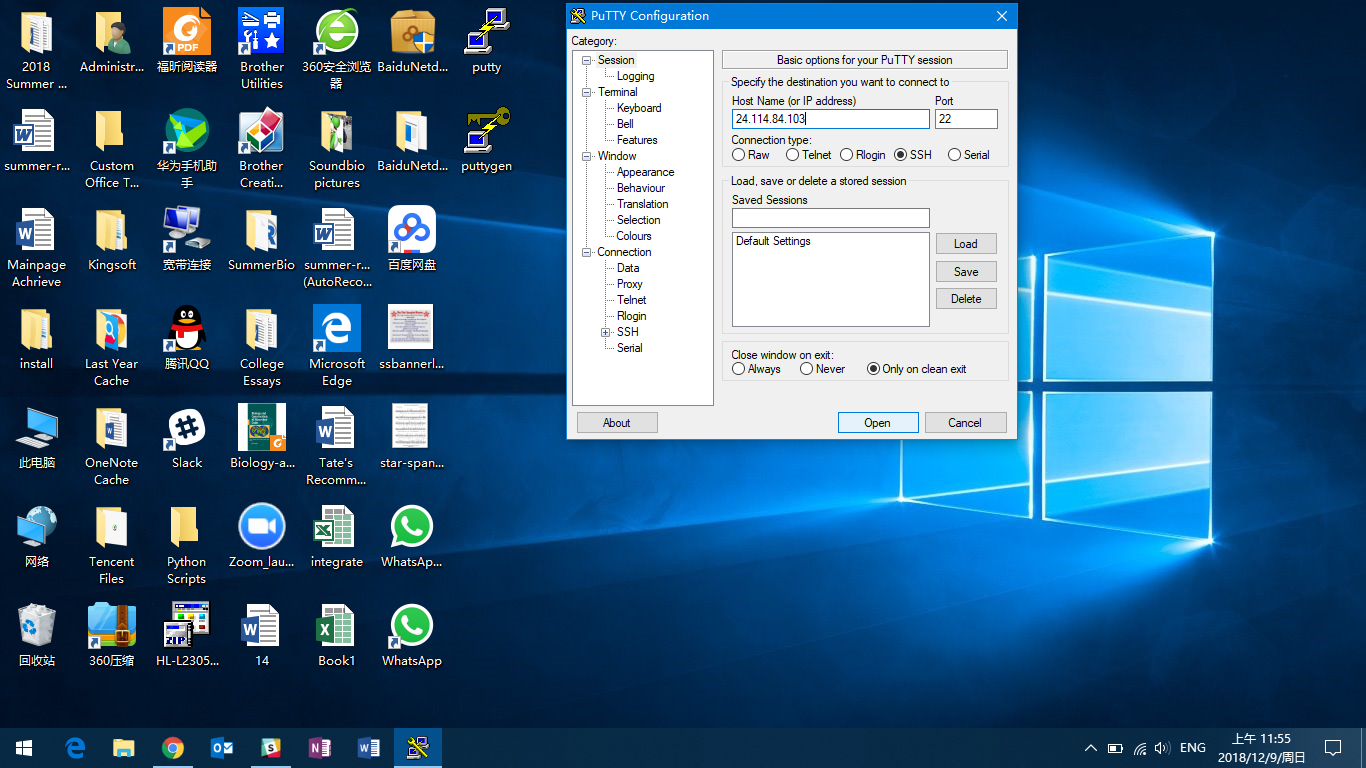
**IMPORTANT (RED BOX):** the default authentication for Administrator Account is “SSH public key”. Selecting such method is ok, but it will make things more complicated. I don’t have a strict security concern in this lab, so I chose to use “username + password” authentication. The authentication set in here are used later in SSH login.

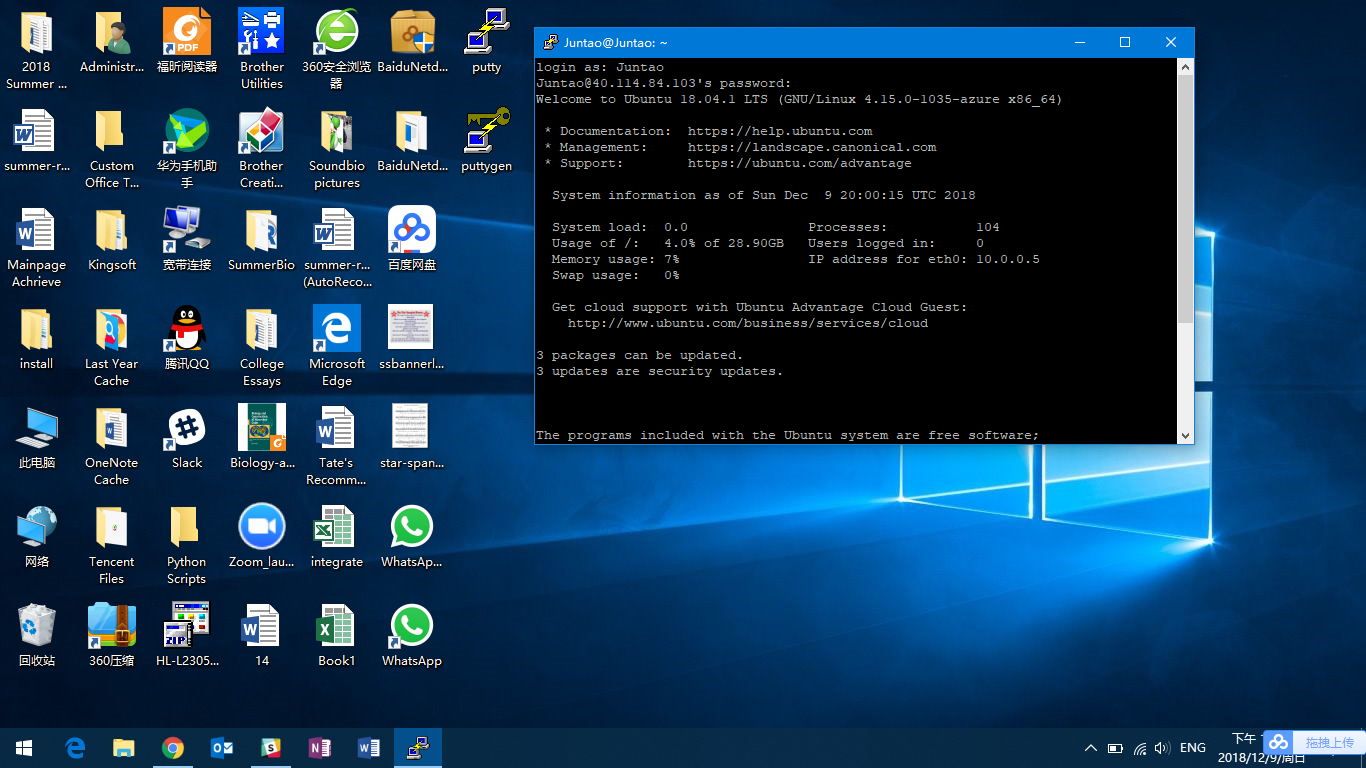


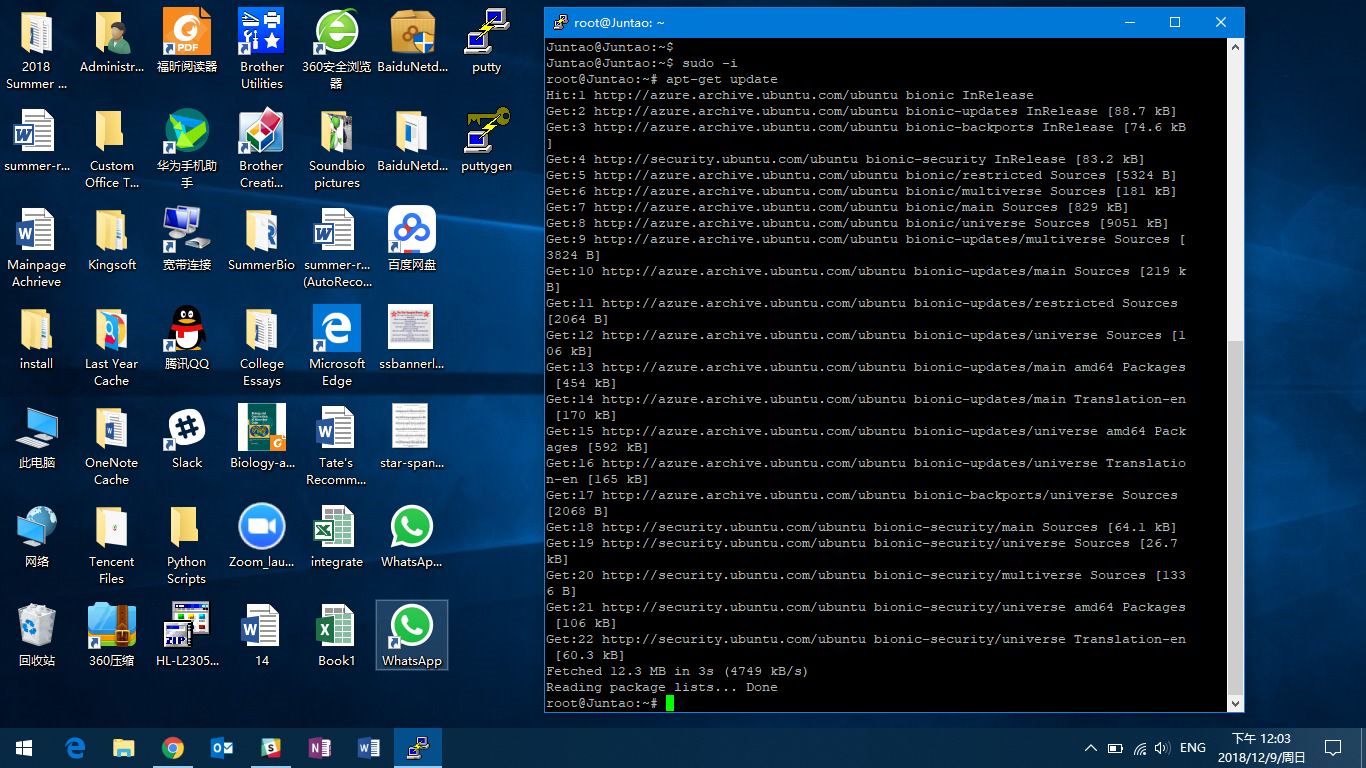
Then, I installed the software tool to SSH (I am demonstrating account setup on my personal PC). I downloaded PuTTY and PuTTYgen (I will use PuTTYgen later in AWS) from Google.



Once the VM is running, it gets a public IP address. So, I used the newly installed PuTTY to SSH my cloud Linux server, using its public IP address. Click open.

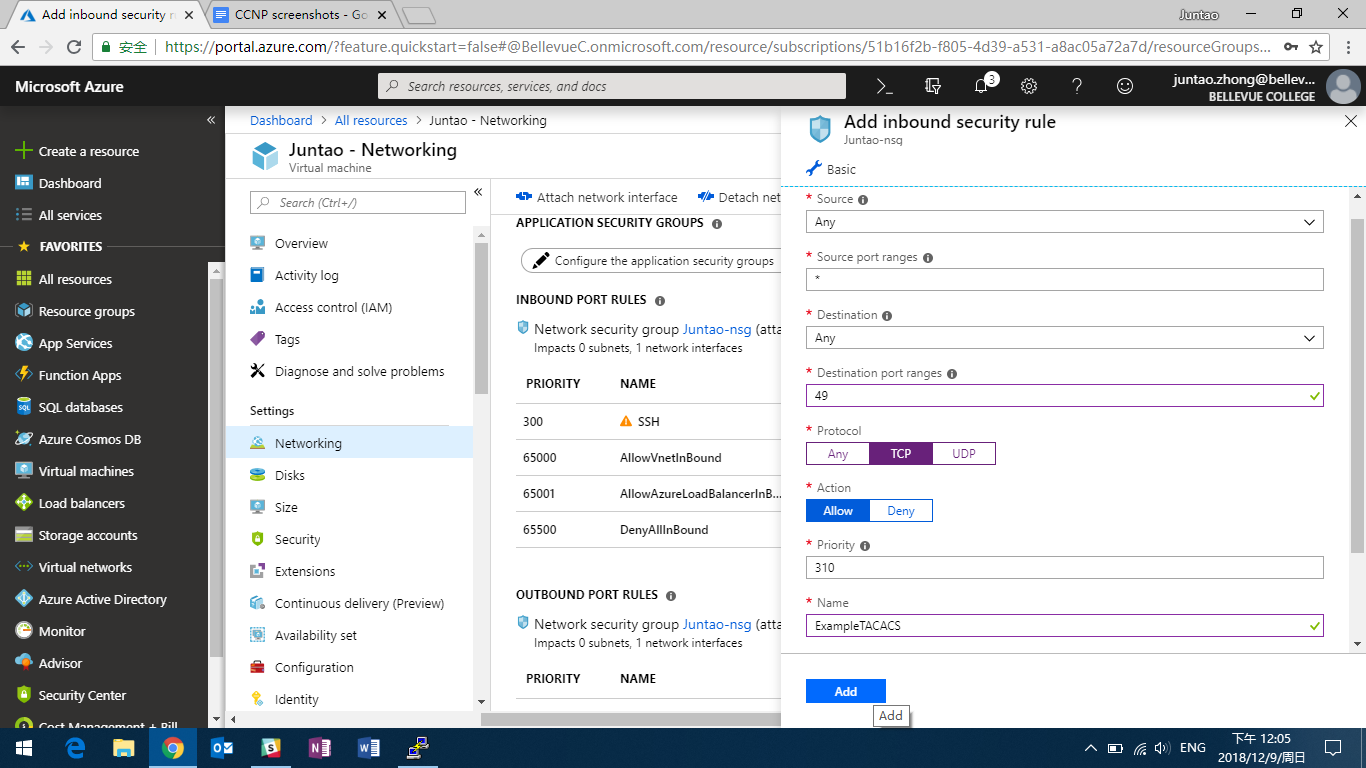






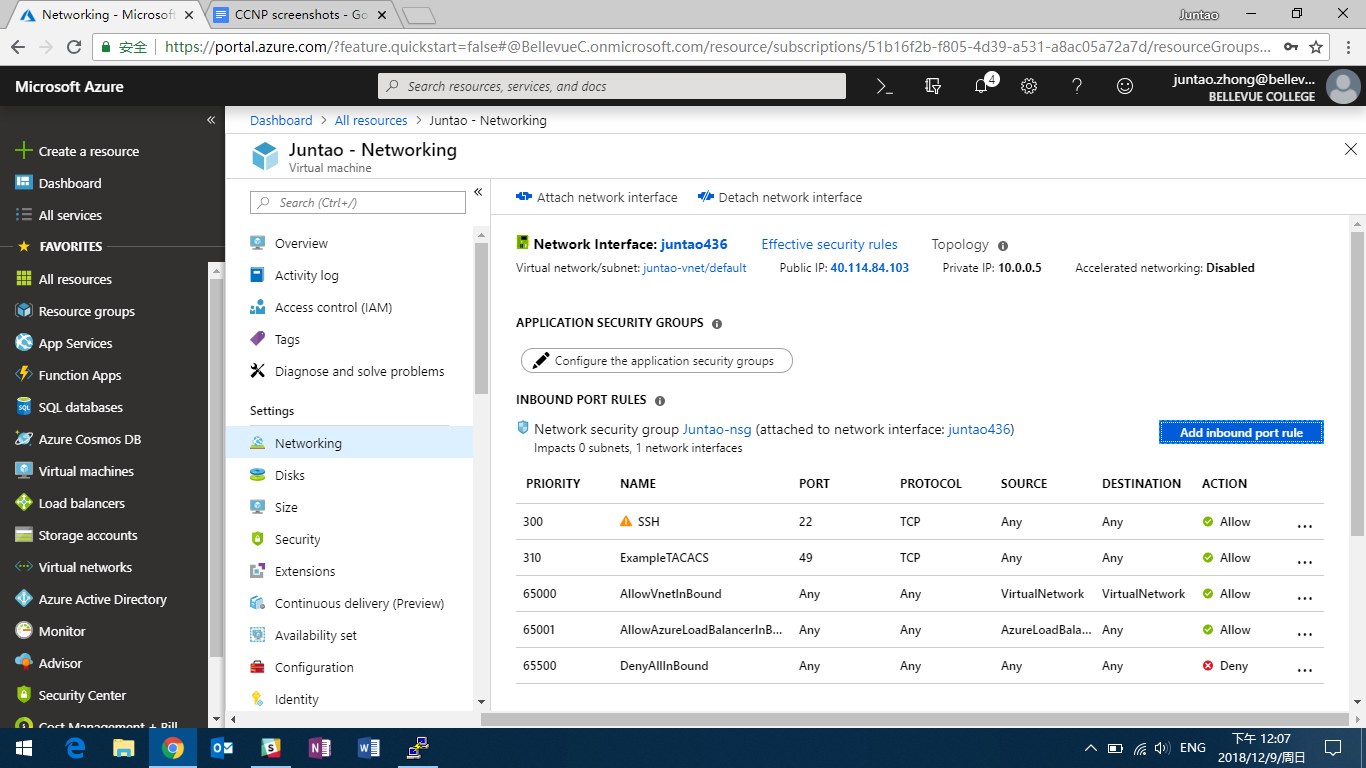
Put in the username and password we just set.

Yeah, we are back to the Linux Server that we are familiar with.



We need to make sure that TCP Port 49 is open for TACACS. TACACS uses port 49, so we need to open port 49 for the TACACS traffic to come in. The default security setting of cloud VMs usually just open SSH and close all others.

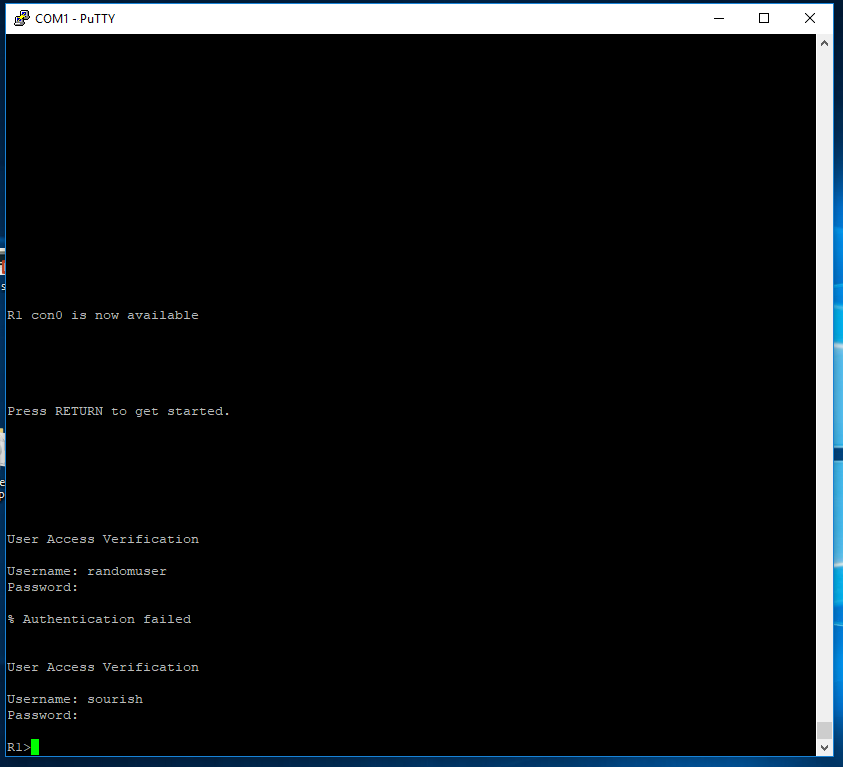
Thus, we need to add inbound port rule to stop blocking TACACS traffic.



After that, we do the usual Linux TACACS+ server configuration (refer to previous lab):

1. **apt-get update**
2. **apt-get install tacacs+**
3. **nano /etc/tacacs+/tac\_plus.conf**. And then, set up the username, password, and privilege level.
4. **Lsof -i : 49.** Check whether the TACACS+ port 49 is running.
5. **service tacacs\_plus restart.**

And it’s done:

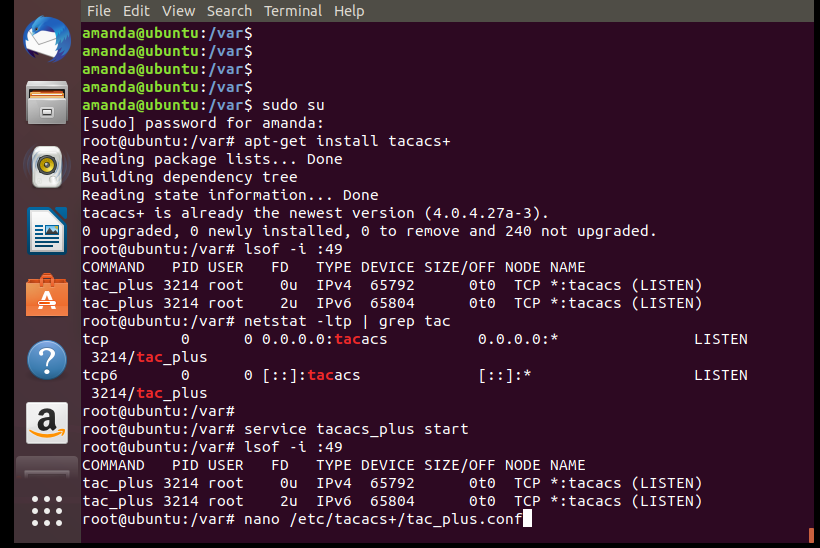


### TACACS+ Routers configuration

### Linux Server Setup:

First, I entered “**sudo -i**” in command prompt to get to Linux root, and issued “**apt-get update**” so that my Linux can download other things from the internet.

Then, I entered “**apt-get install -y gcc make flex bison libwrap0-dev**” and “**apt-get install tacacs+**” to install TACACS+ service in my Linux.

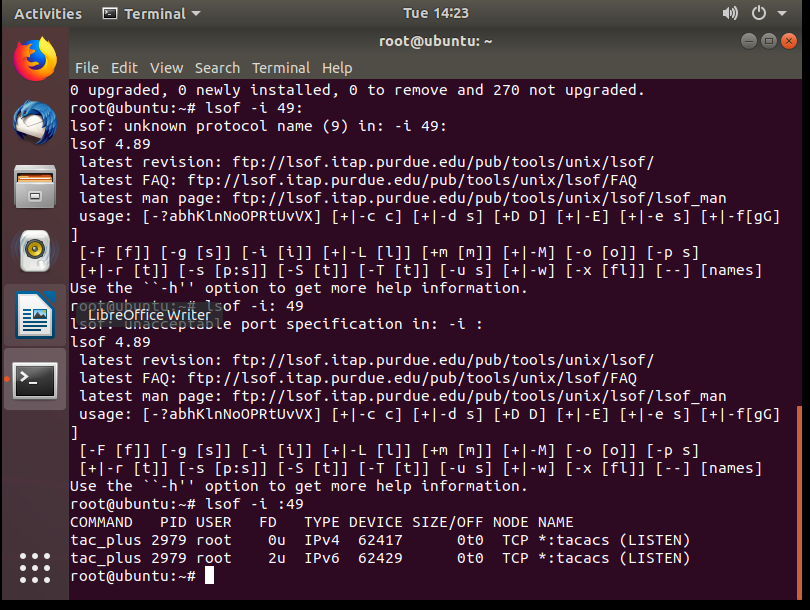


After downloading TACACS+ service, I edited its configuration file section with “**nano …**”to setup TACACS+.



Inside the configuration file, I defined the key (domain) between my server and router, the group with privilege levels and users in such groups.





The last step was to check port 49 with “**lsof -i :49**” to see whether TACACS+ was running. The spelling and space must be exact for it to work. Then, TACACS+ platform was good to go.

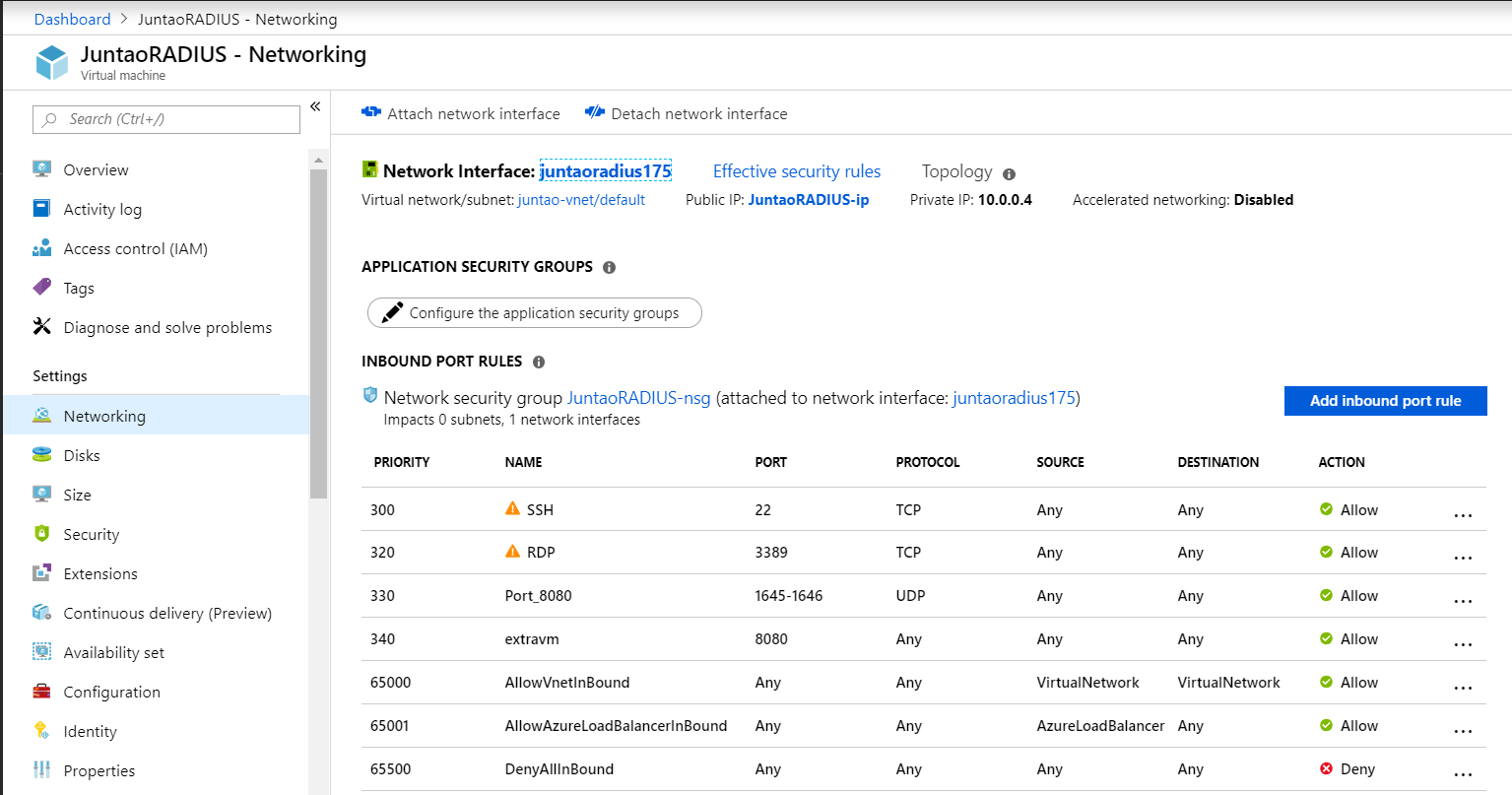
That’s what I saw when I console into my router. If ssh or telnet was configured, the User Access Verification should be similar as well.

### Azure radius vm Setup:

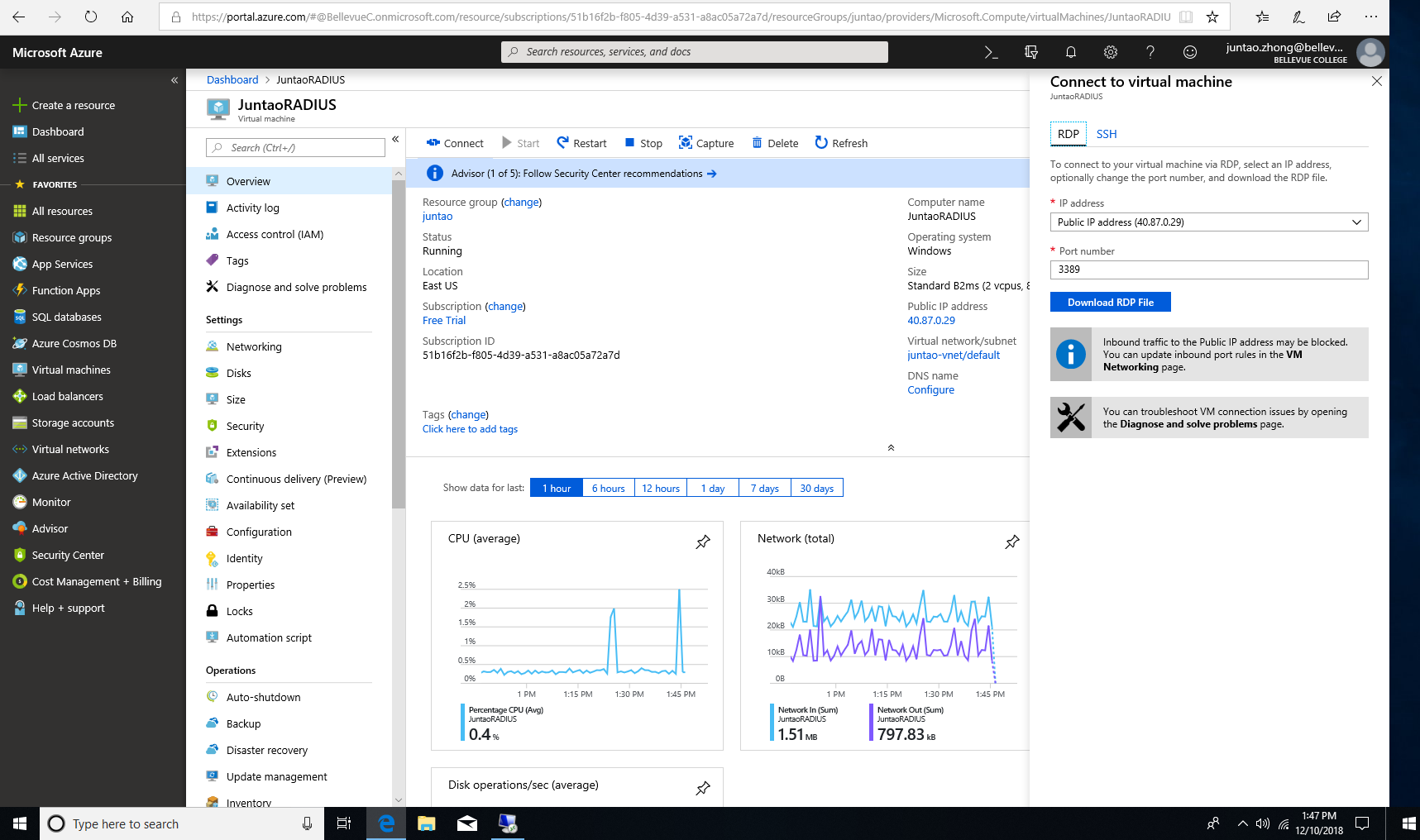
The setup of Windows Server VM on Microsoft Azure is very similar to that of Linux Server on Azure. Since it’s more convenient to configure RADIUS on graphic user interface instead of command lines, we want to remote desktop into our VM.

The default setting of Windows VM blocks **TCP Port 3389** for security reason. **UDP Port 1645-1646** are also closed; but RADIUS needs them to run. So, we need to go to “Networking” and open these three ports. The process is the same as adding inbound rules on Linux VMs that I showed earlier.

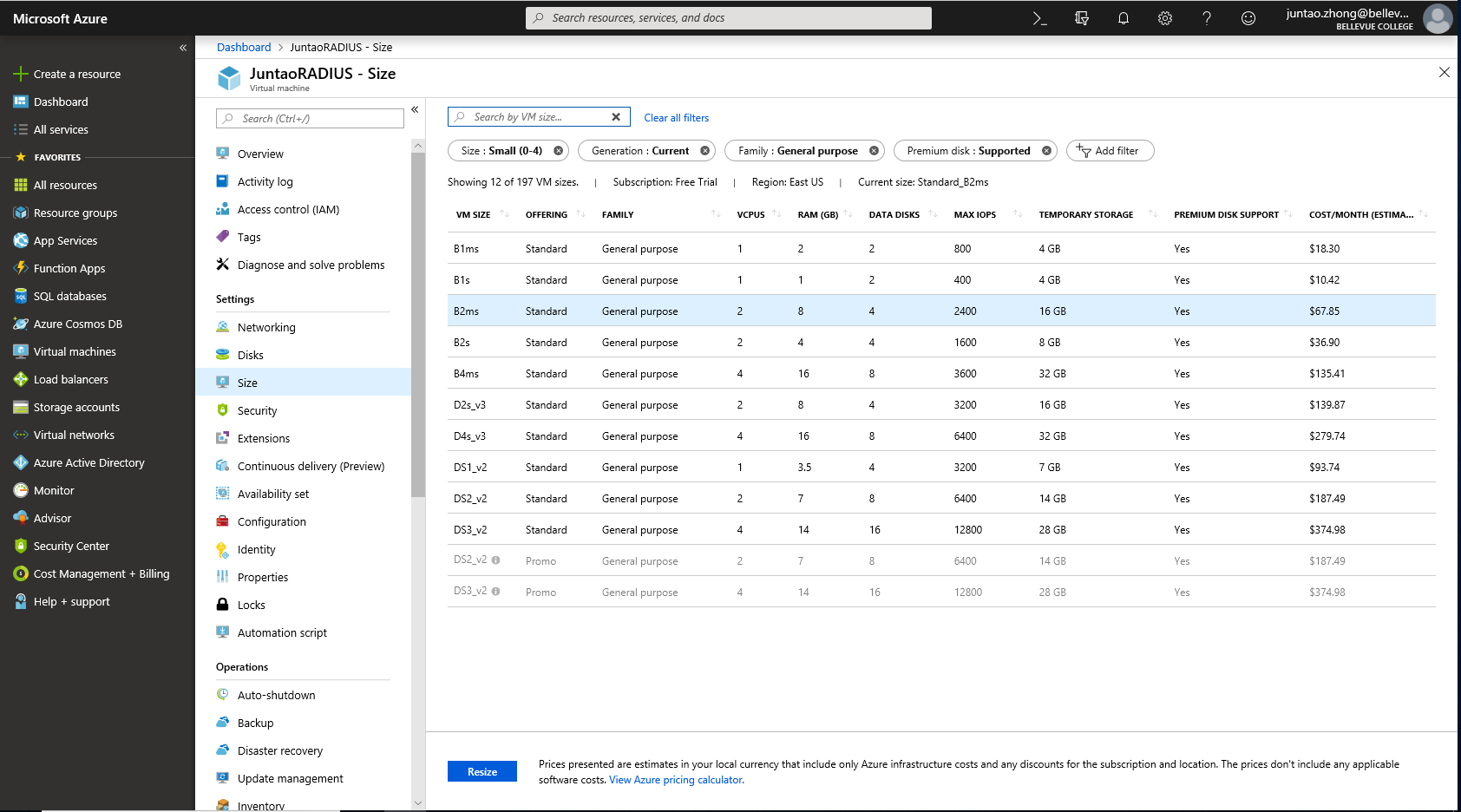
Port 8080 is only needed if you want to remote desktop into multiple VMs, for example working on both Azure and AWS Windows servers at the same time.



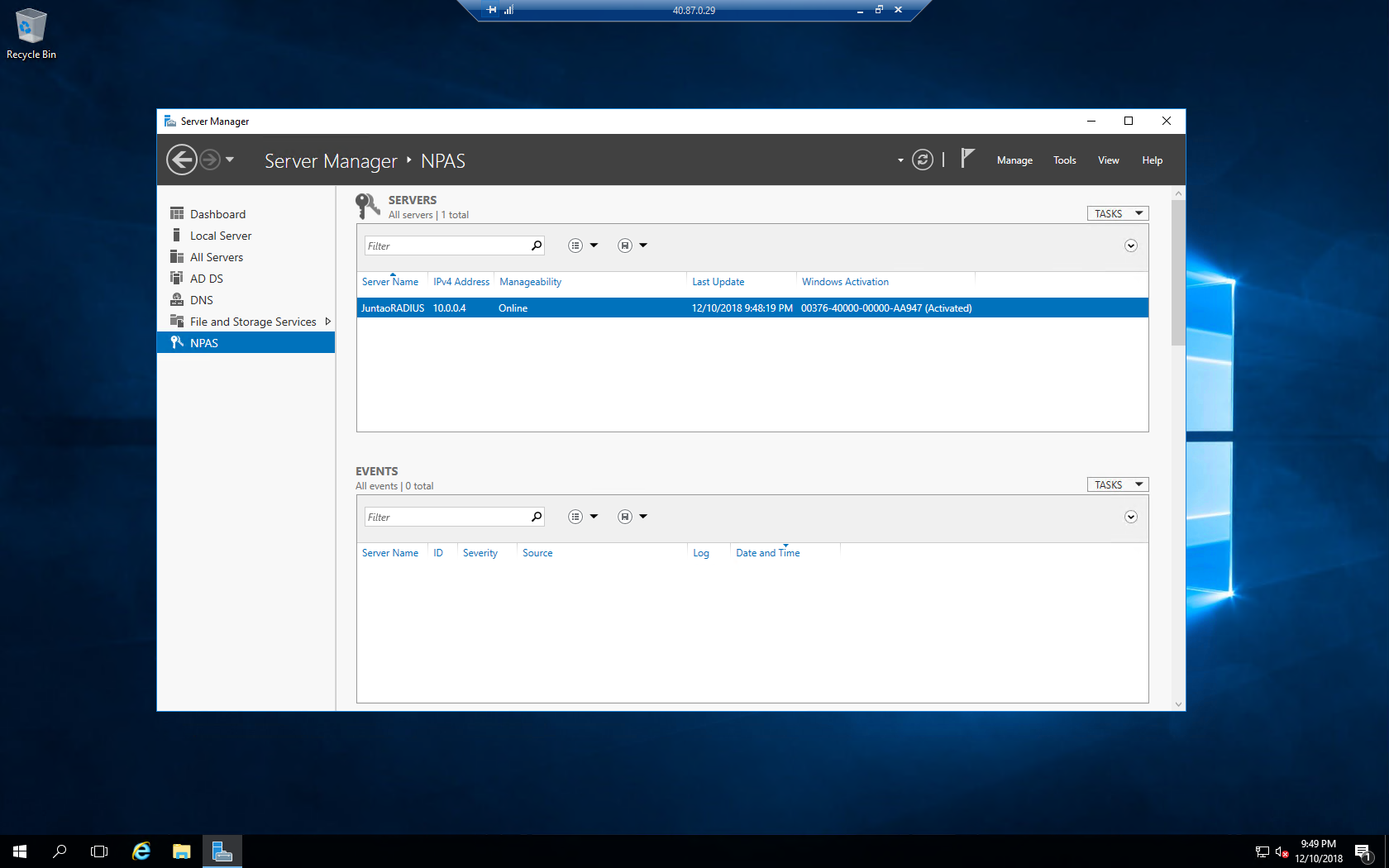
After opening the ports, go to “overview” and click “connect”. Download the RDP files and it will connect you to the VM.



Windows Server takes up a lot of space, so is the RADIUS related programs. 4GB of storage is not enough, so I later switch to 16GB storage. One advantage of cloud VMs is that you can scale it easily. It took less than a minute to upscale my VM.



And we’ve entered our familiar Window Server 2016. Compared to local RADIUS, there are some different steps we need to do, which I will cover in AWS Windows VM.

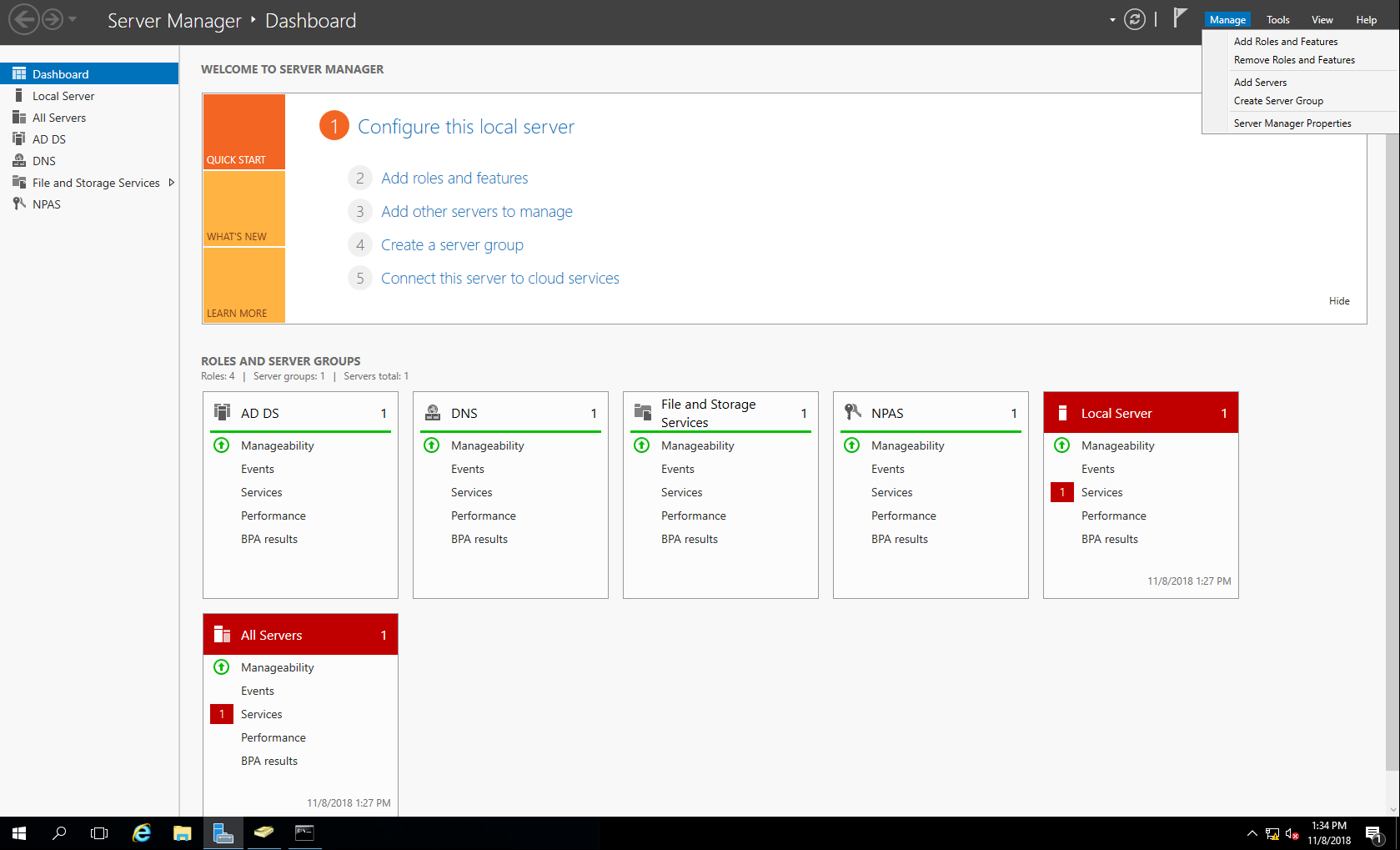


### AWS radius vm Setup:

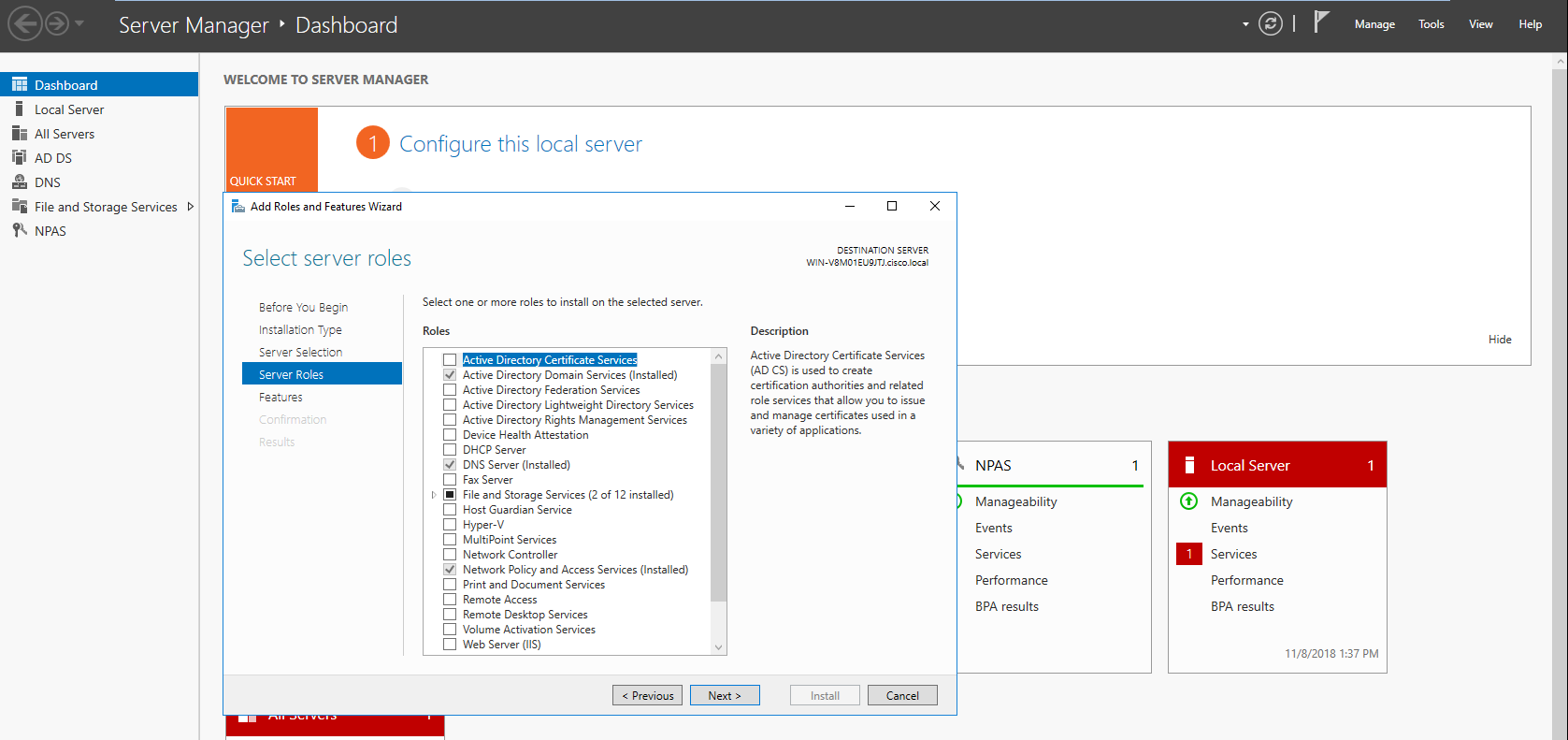
Since Windows has a nice-looking graphic user interface, there are a lot of clicking and screens involved. There are three main components for RADIUS: Active Directory Domain Service (AD DS), Network Policy Server (NPS) and the authentication with routers.

#### **Preparation Step: download the services needed.**

First, I enter the **Dashboard** for Server. Hover over the **Manage** tab and click on “**Add Roles and Features**”. We are going to download AD DS and NPS. Make sure the server connects to Internet.

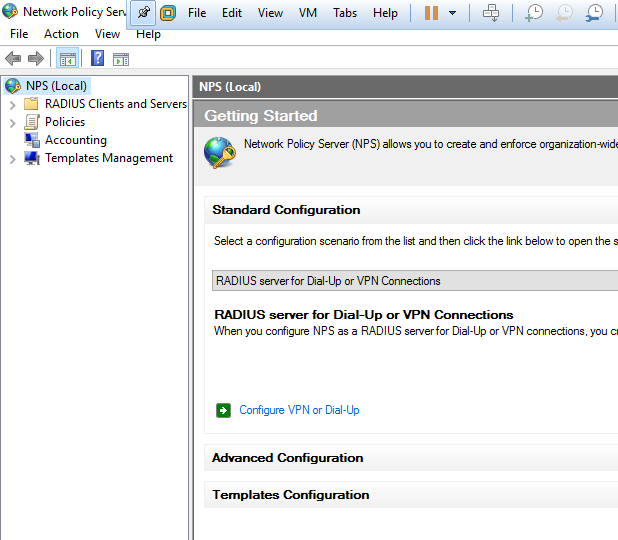


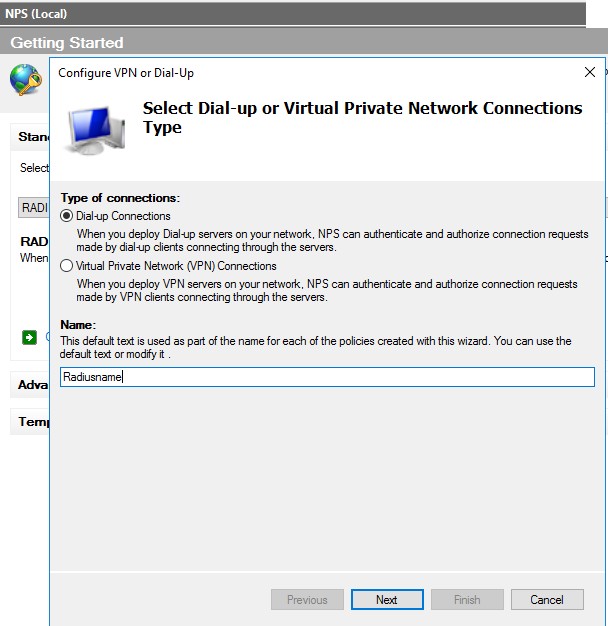
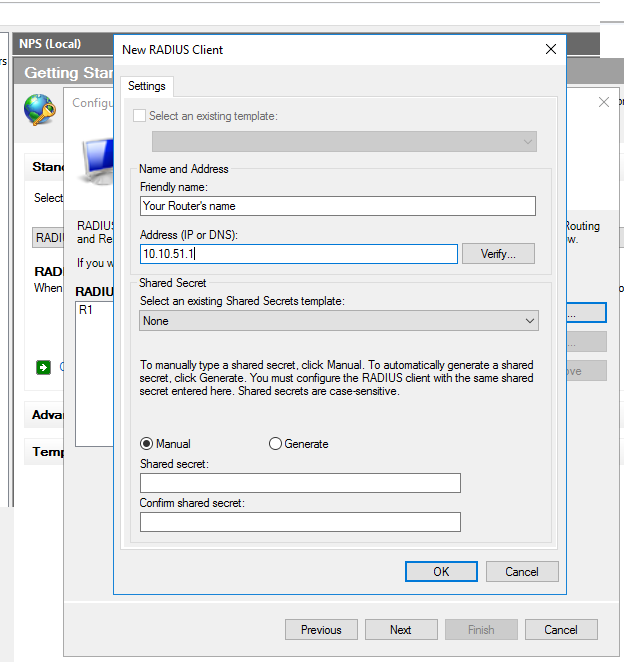
The next step is intuitive: select NPS server and Active Directory Domain to Download. After a successful download, you can find ADDS and NPS from your application list.



#### **Configuring Domain and Authentication in NPS**

First, I created and configured a TACACS+ domain that encompasses all devices under my server’s authentication process.



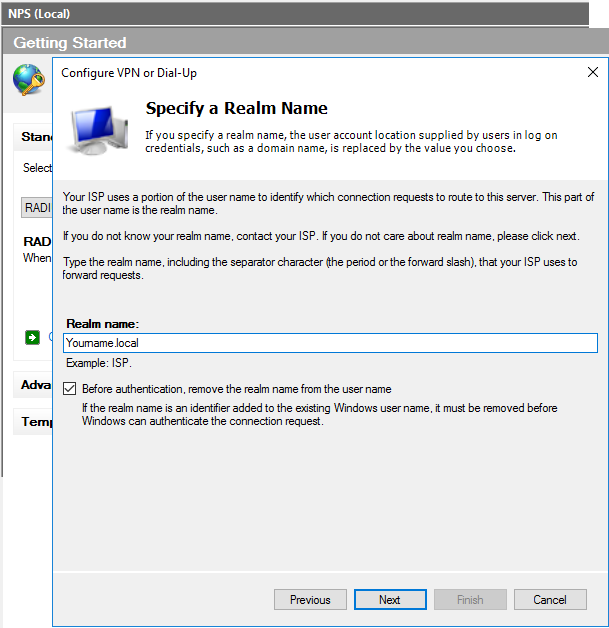


The name should be the name of your RADIUS platform.

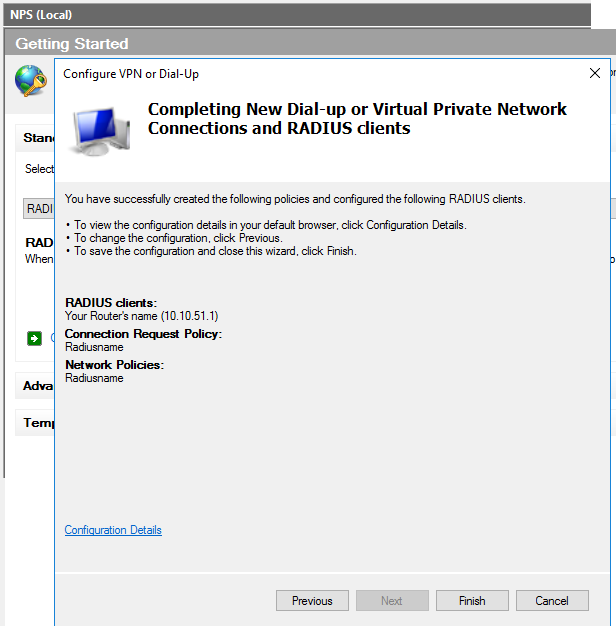
Click “next” after name it. There are a lot of “next” in this GUI, just move forward and follow along with system instruction.

Define the Friendly Name with hostname of the router. Also type in the network that your network devices are belong to.

Share Secret should be the same as the shared key in router configuration. In this lab, I set it to a simple “123456”

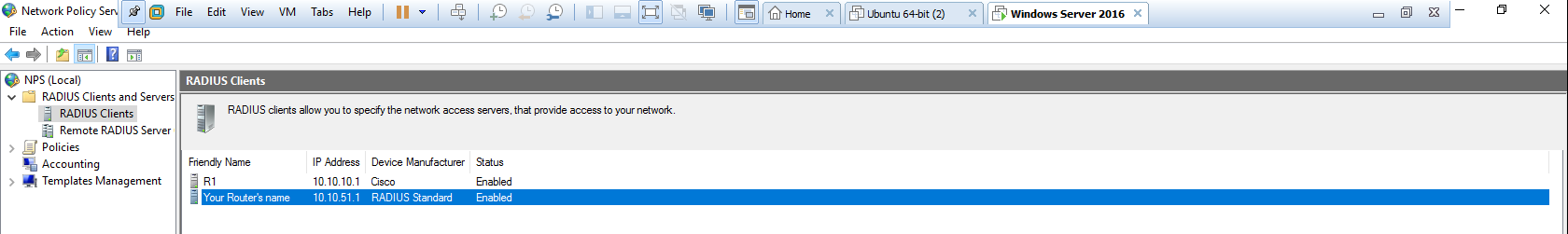


Realm name is important, because I need to use it in future steps. Since I am configuring a local network isolated from the Internet, I used “**local**” as my realm domain. You can choose a name you want before “.**local**”

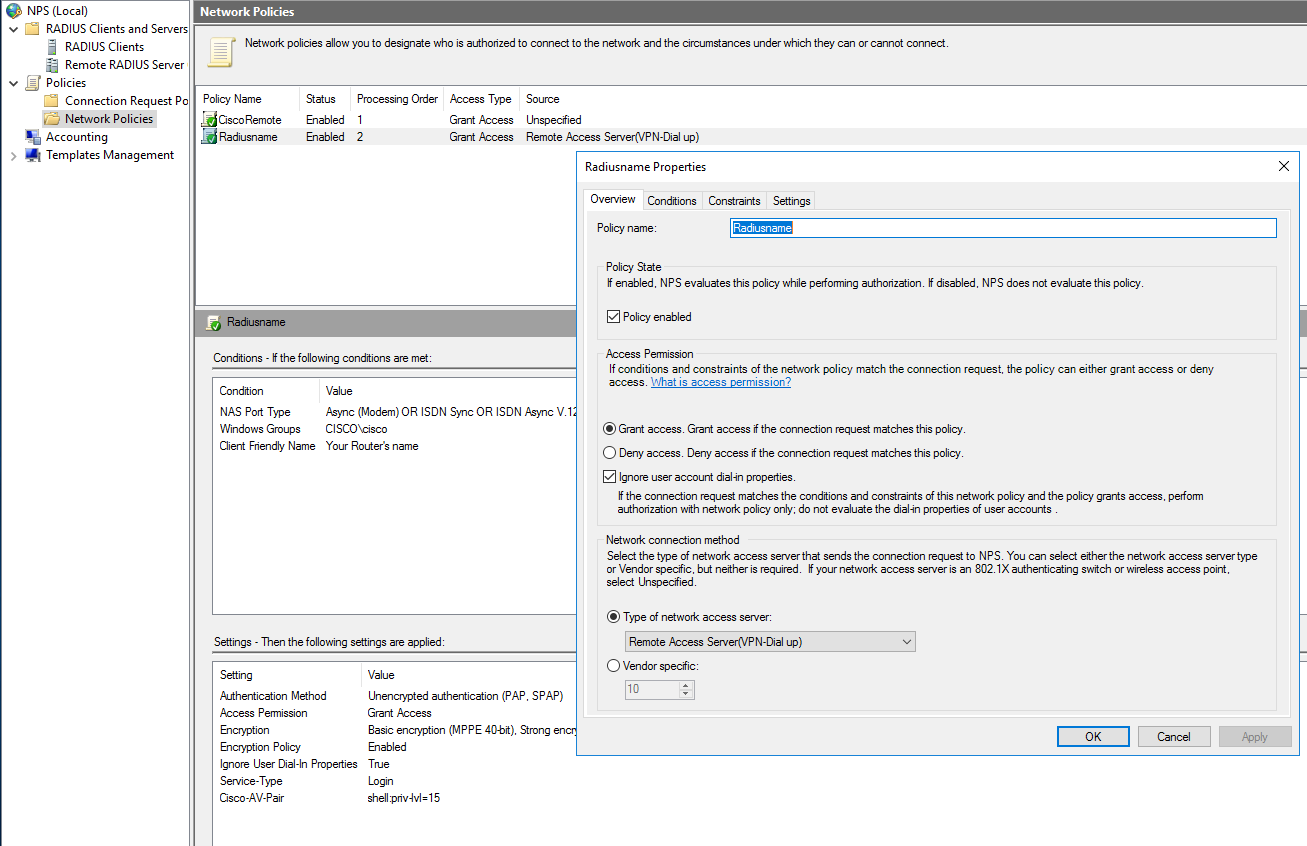


Confirmation step. After you click “Finish”, your RADIUS domain is created.

And here is the domain you just created. “cisco” is my working RADIUS domain that I configured for this project before.



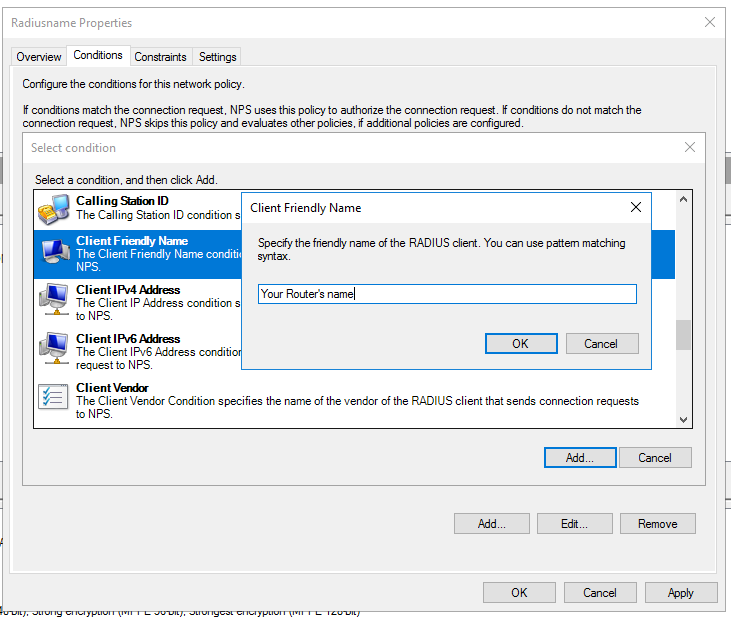
#### **Configuring Privilege and Administrative Network Policy**



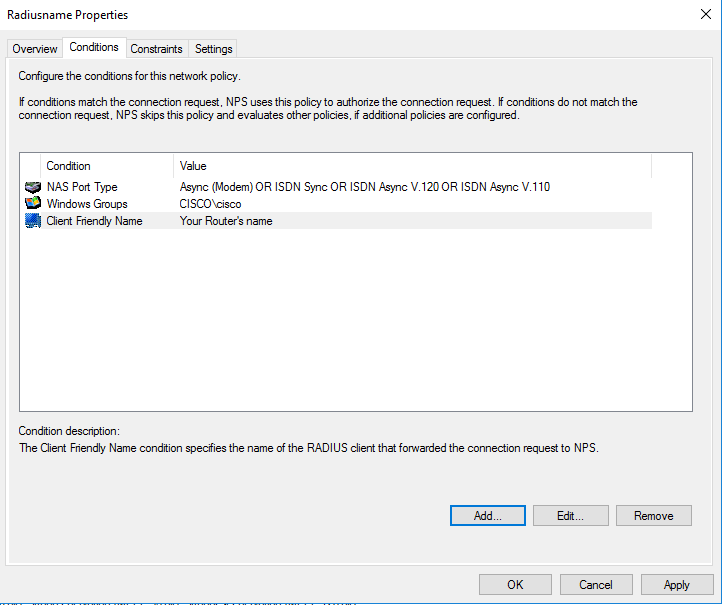
Add your domain into this policy

Next, I needed to configure specific policies in my Radius domain.

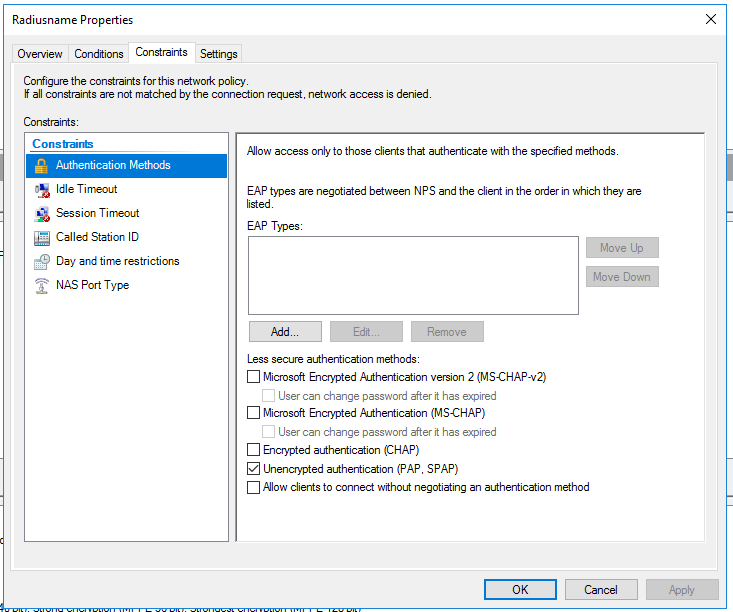
We still operate in NPS but go to “policies” tab and configure.



Add your router as part of this policy

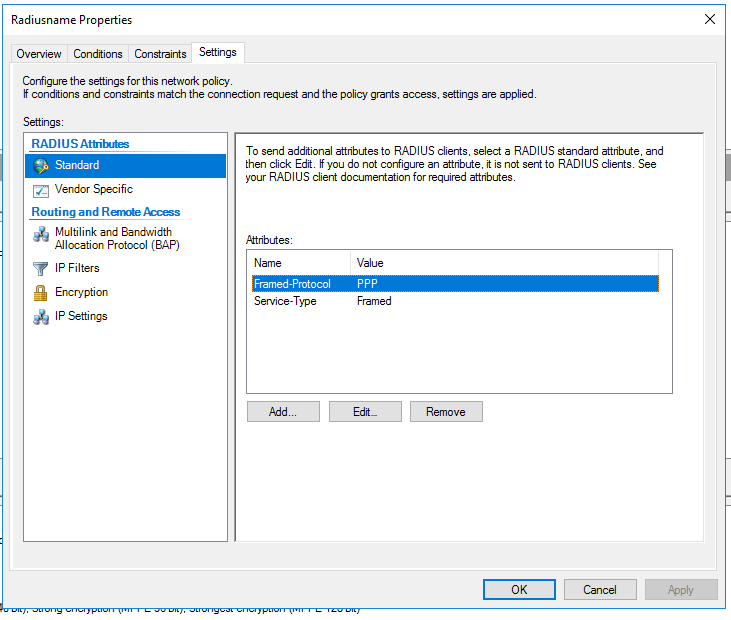


Two added. Confirmed.



Next, I needed to define what privilege/ policies I have in my domain.

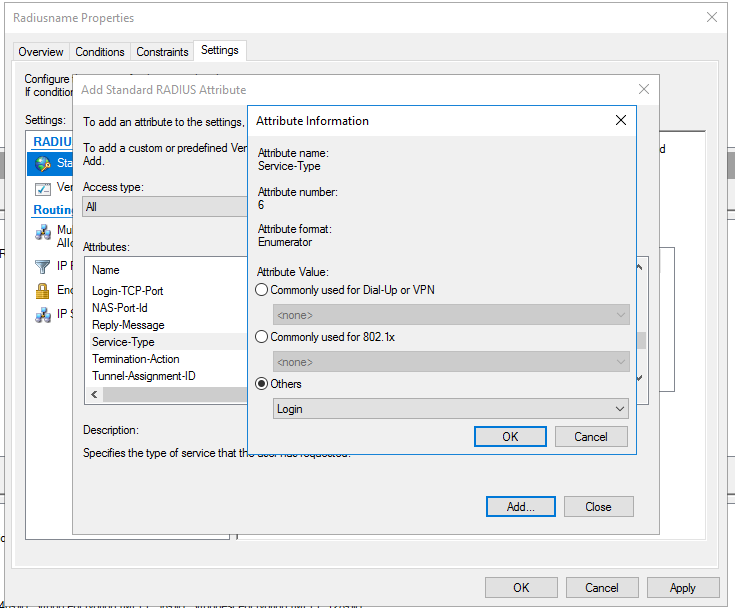
In “Constraints” tab, it showed me the Authentication Method between routers and server. Since it was a practice lab, I used unencrypted authentication.



Then, in “Settings => Standard”, I set up what service type did I want.

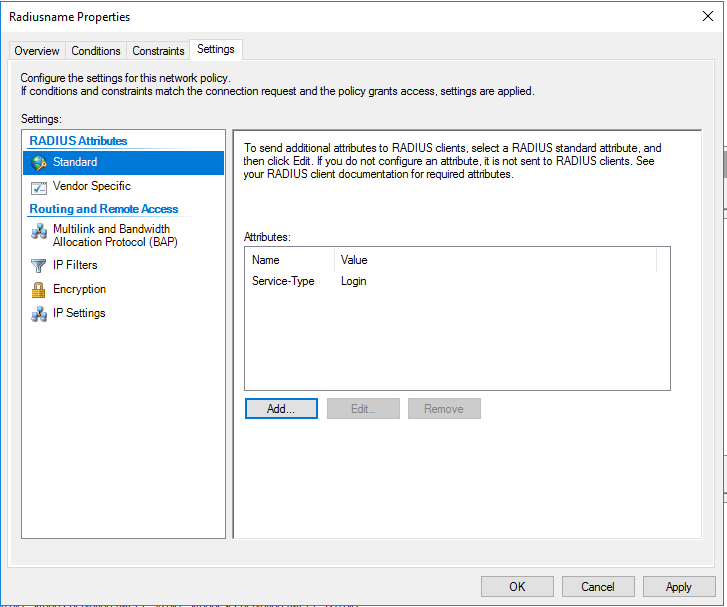
Since I want to login into my RADIUS devices, I wanted only “login” service.

So, I removed two existing Attributes.



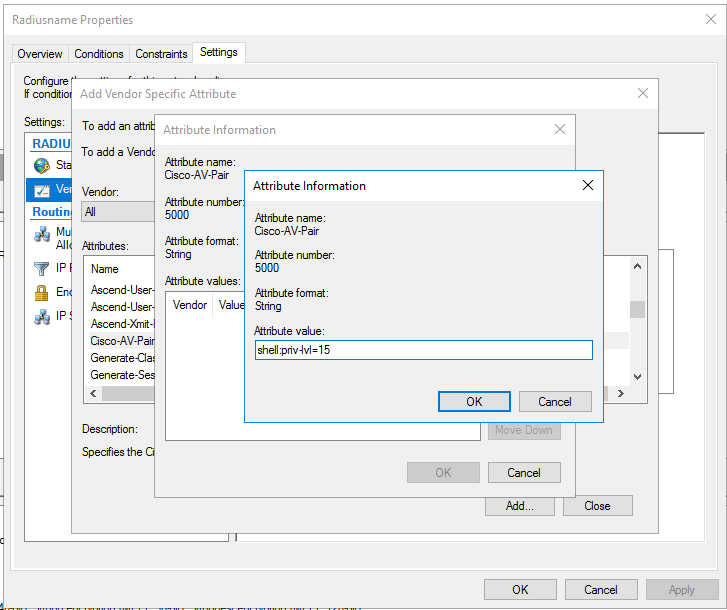
I clicked “Add” to load “login” service.

I clicked “Service-Type” and select “login” from “others”



Confirmed.

Two other original attributes are removed and “login” should be the only attribute.

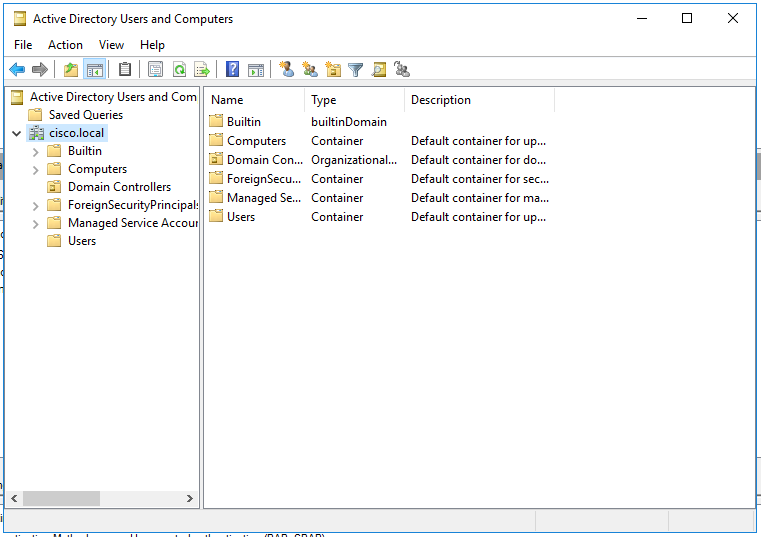


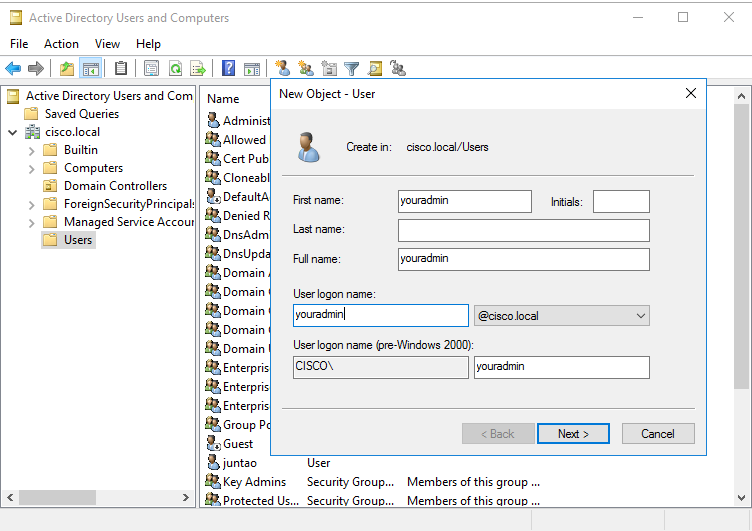
Last step in NPS: configure the privilege level for my domain.

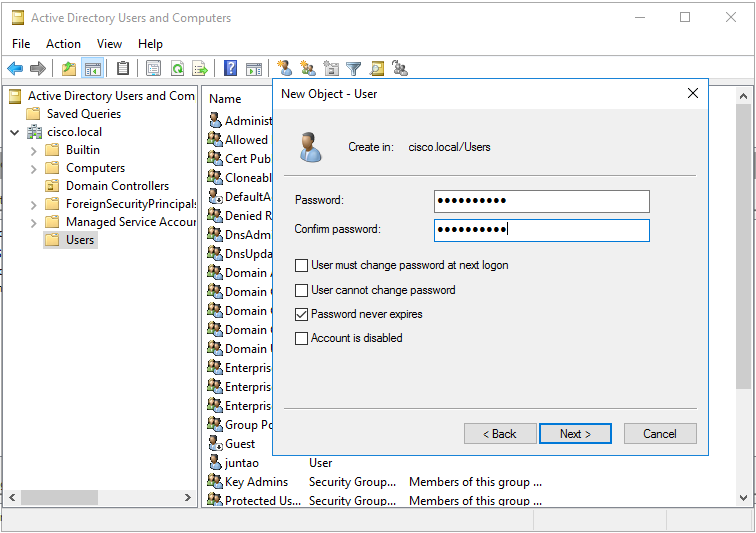
Privilege 15 means that I gave anyone in this domain unreserved privilege to view and write anything.

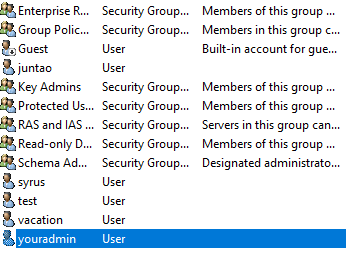
#### **Configuring Active Directory Domain Service (AD DS).**

The next step is to add various users/ administrators into my domain. Open Active Directory Domain Service. Go to your domain (cisco.local) => users and click add users. The password is configured in this process, just follow the system instruction.









User Confirmed.

Last step, add your user into the Your routers’ group.

After that, your router is ready to go. The console authentication is almost the same as TACACS+, so I didn’t show it here.

You can login into your RADIUS authenticated router now.

## Problems encountered

**Linux TACACS+ Administrator directory:**

when I first configure the file for TACACS+ user directory, I thought that I only need to add a user into the directory with username and password. My username and password didn’t work, and I couldn’t log in to my router. However, I later found I must configure a group with defined privilege, and then assign users/admins under that group for TACACS+ to work. Now it makes sense to me because a privilege level must be defined in order for TACACS+ to operate.

**Log Out of Routers, Can’t Log In:**

When I configure first AAA accounting on routers, I didn’t set up my servers correctly, which leads to the failure in authentication. Since authentication failed, I couldn’t log in or do anything with my routers. At first, I had to power cycle my routers to erase my mistake, but that took a lot of time.

To solve this question, I need to configure a local user with:

**R1(config)**# username ***backup*** secret ***#####***

Now, I can log in to and configure my routers even though TACACS+ or RADIUS don’t work. Note that this local user will only be valid when the server is unavailable (for example, unplug the ethernet cable between server and routers).

**Not Sure About the Direction to Go:**

Before, I only configured routers in cisco CLI, or computer in GUI. I was clueless when two distinct interfaces intercepted, so I spent a lot of time following websites that don’t contribute to my project. After extensive research and discussions with my classmates, I learned that basic components of AAA authentication are an active directory, a shared key between routers and server and a defined network policy, any other things are security/ management preferences. After these three essential parts, I’ve built the skeleton of my AAA accounting platform.

**VMware connection issue. Two different modes**

The last problem is not really a problem in actual servers. Because my servers are virtual servers based on the ISO images in VMware, there are two ways that a virtual computer can connect to the physical network: either uses the same ip address as the physical PC, or a unique ip address different from the physical PC. In this lab, my virtual and actual computer each needs an ip address, so I need two ip address for one physical computer.

## Summary

I used configured TACACS+ server on Linux and RADIUS server on Windows 2016. TACACS+ and RADIUS are platforms that centralizes the router login authentication into one server. In this way, I increase the security of my system because I can monitor one server more closely than a lot of individual network devices. There are three essential part of for this platform to run: group policies, user directory and authentication configuration on both routers and servers.

In this project, Linux is much easier for the people who set up the server because all essential parts of the AAA platform are concentrated in a script of codes. Though it’s not easy for a person new to Windows Server to find the right buttons to click, it has a lot of extra features and explicated options; and after all, it has a nice graphic user interface close to PCs that we use every day, which makes it more user friendly in some degree.