A10 Cloud Manual

This manual will walk through the setup of a virtual machine (VM) hosted on Microsoft Azure cloud. This VM will have:

1. Hosting for firmware binaries & status files used in over the air updates (OTAU)
2. A Node-RED instance for flow-based UI development
3. An MQTT communication broker for sending messages to and from out IoT device

# Table of Contents

[**Table of Contents 1**](#_81brb461c1m6)

[**1. Setting up an Azure VM 3**](#_ikbtoigyhigw)

[1.1 Azure for Students Account 3](#_lu4kxygvqd8x)

[1.2 Refer to Node-RED’s Azure Setup Instructions 4](#_g34u1kbqe2mp)

[1.3 Free Azure VM Setup 4](#_8g8bngnhg2jr)

[1.4 Create a virtual machine - Basics Tab 1 5](#_6inpvplw9ego)

[1.5 Create a virtual machine - Basics Tab 2 6](#_iw6yhvoehq6v)

[1.6 Create a virtual machine - Review + create 7](#_zfo51ummjxag)

[1.8 VM Overview 9](#_kf37swerwp9v)

[1.8 Add inbound security rules 10](#_ipfc09e9ikgc)

[**2. Hosting the OTAU firmware binaries & text file 11**](#_14dab8q9tsbb)

[2.1 SSH from your computer to the Azure VM 12](#_lch6cvvewvwy)

[2.2 Install Apache web server 14](#_hkxh3yiqkj5)

[2.3 File Upload 16](#_58hm3tl09lj2)

[2.4 Firmware: Configure for HTTP 17](#_me8rhneja62)

[**3. Install Node-RED on your VM 17**](#_vbnznco39o68)

[3.1 Add inbound security rule 18](#_75yb5i9hfm9u)

[3.2 Install Node-RED on your VM 18](#_dab89074z7re)

[**4. Node-RED Configuration 21**](#_2b0c5ajmnjf9)

[4.1 Add functionality with nodes 21](#_b2t6adov8rbn)

[4.3 (Optional) Explore an Example UI 23](#_bjl39mcb3t1b)

[4.4 Node Descriptions for Game Simulator 24](#_xkf3jicmnyhs)

[MQTT IN 24](#_sq6h4ztlih53)

[MQTT Out Block 25](#_c8du6uihxren)

[Debug Node 25](#_e4qye2if1ppd)

[JSON Block 26](#_1yf65n3tiuxr)

[Javascript Block (Function) 26](#_eo7fuly1eiqy)

[Front end (UI) Blocks 27](#_cgl6vhgebhz3)

[Deploy 29](#_iou4t6q00uzy)

[Node-RED Dashboard 30](#_z7sen4f0mftk)

[Additional Node-RED Resources 30](#_hediqabqazw5)

[**5. Hosting an MQTT broker 31**](#_49ia53xv9jvr)

[5.1 Connect to Your Azure VM 31](#_e4g7xiwlwm8q)

[5.2 Install Mosquitto MQTT Broker 31](#_72eqf9441c3g)

[5.3 Test the mosquitto installation 31](#_xrzoebvfb1v)

[5.4 Enable Remote Access & Firewall Config 32](#_gvir5yv5ejhm)

[5.5 Restart the Azure VM 33](#_mjd9ybuj8yxw)

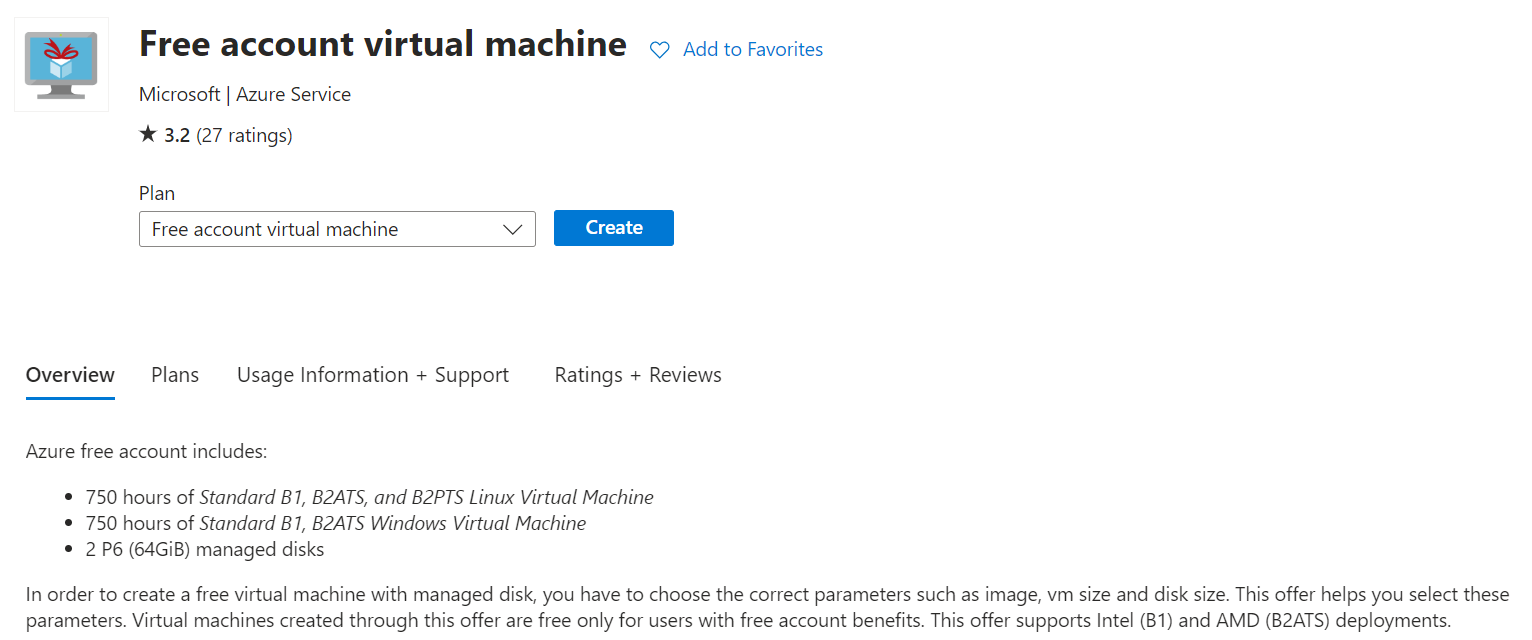
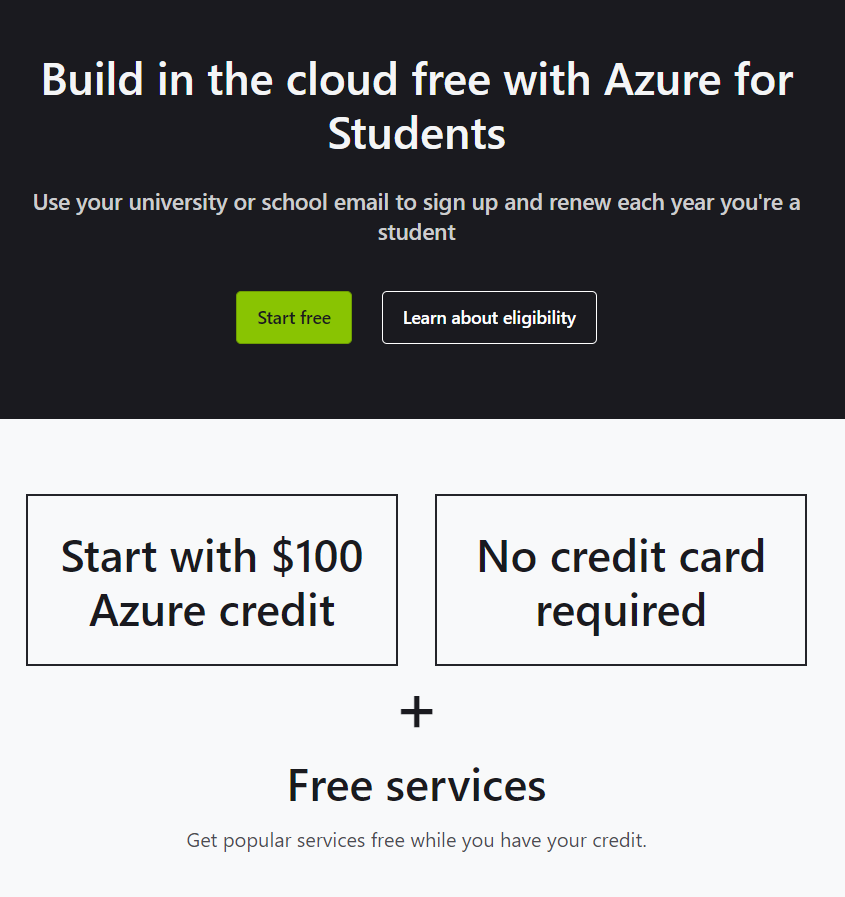
[5.6 Test the MQTT Broker 33](#_o9esn7rza9k6)

[Node Settings 34](#_mye88j3sz4ol)

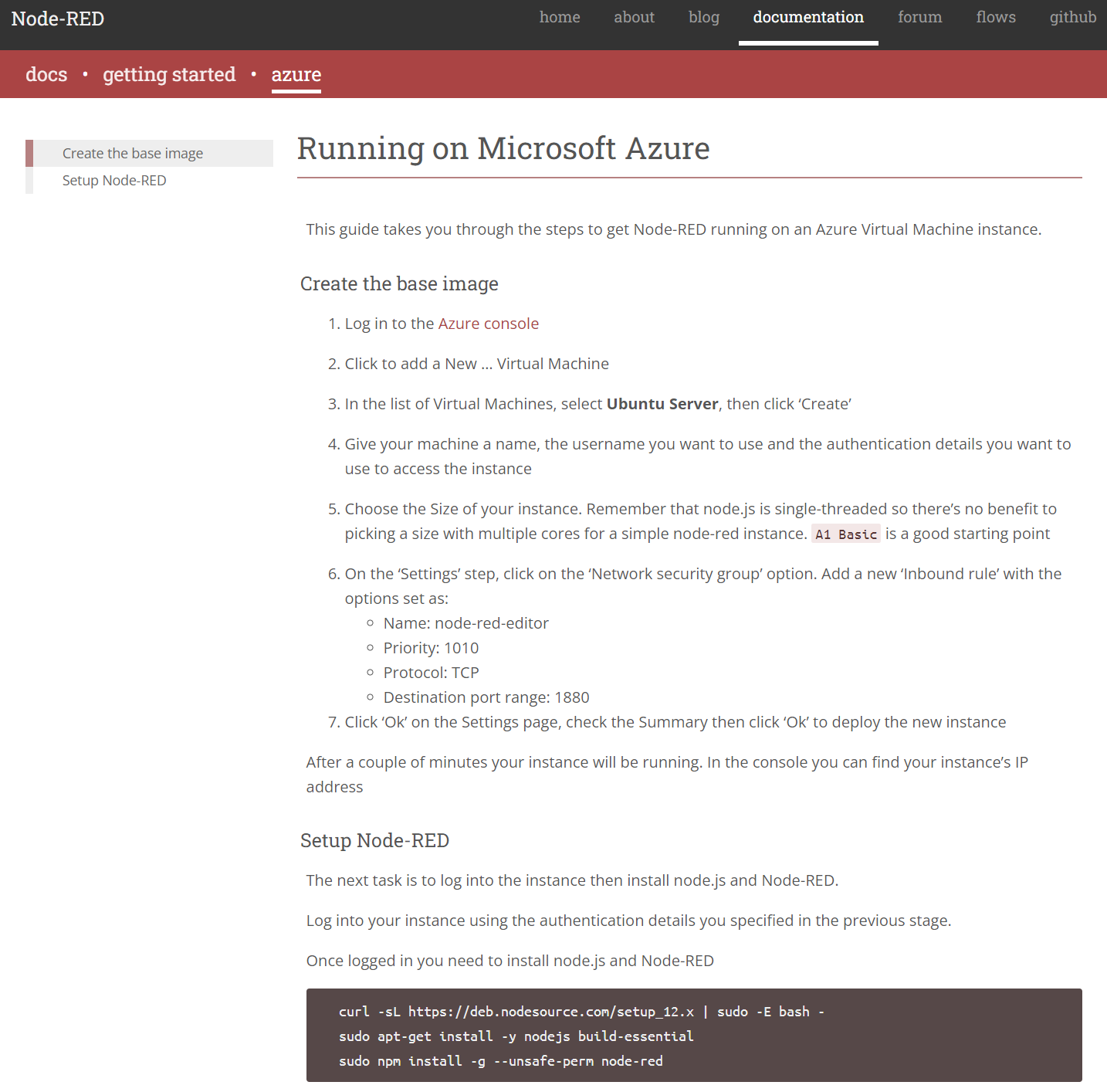
# 

# 1. Setting up an Azure VM

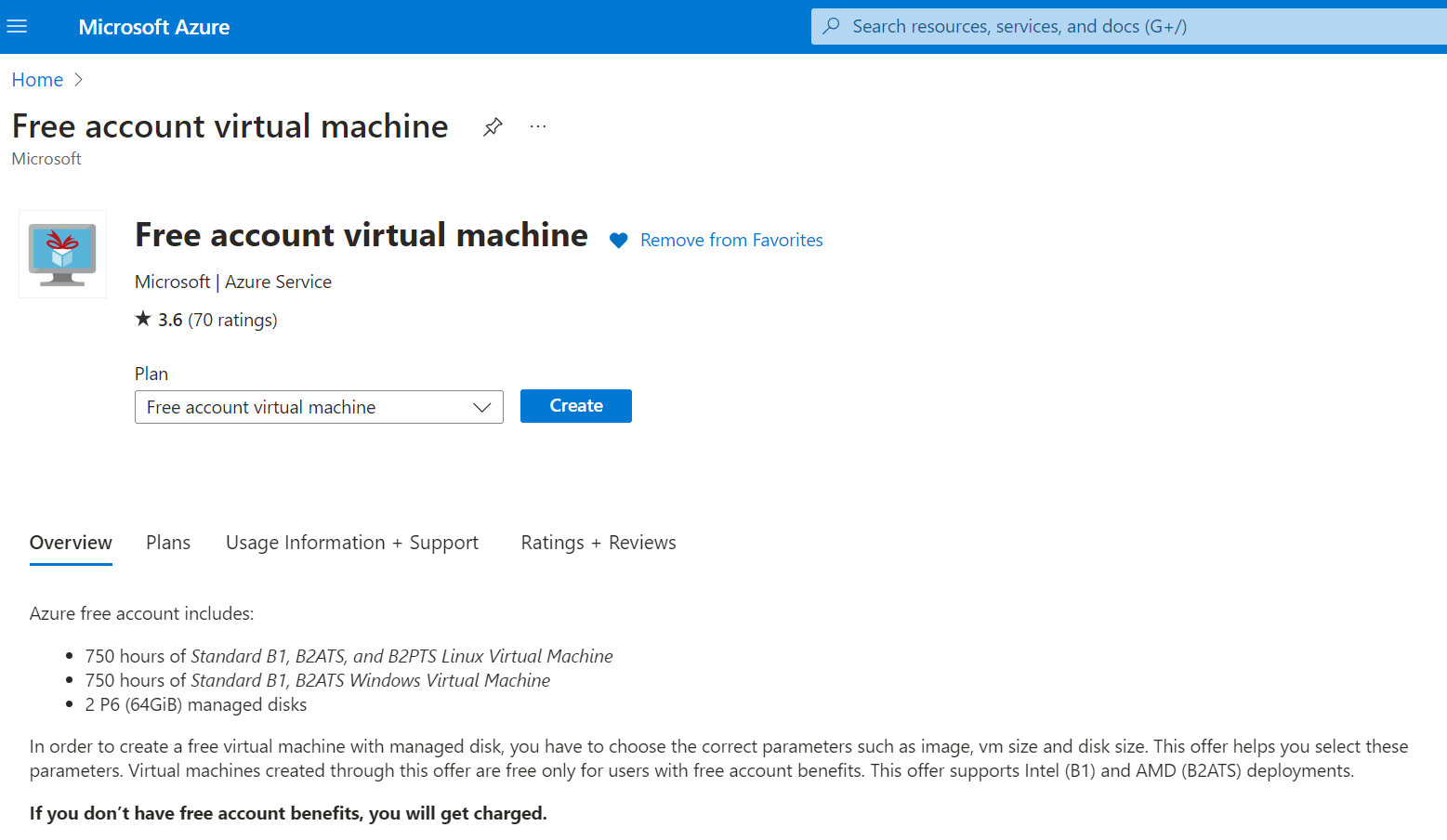
## 1.1 Azure for Students Account

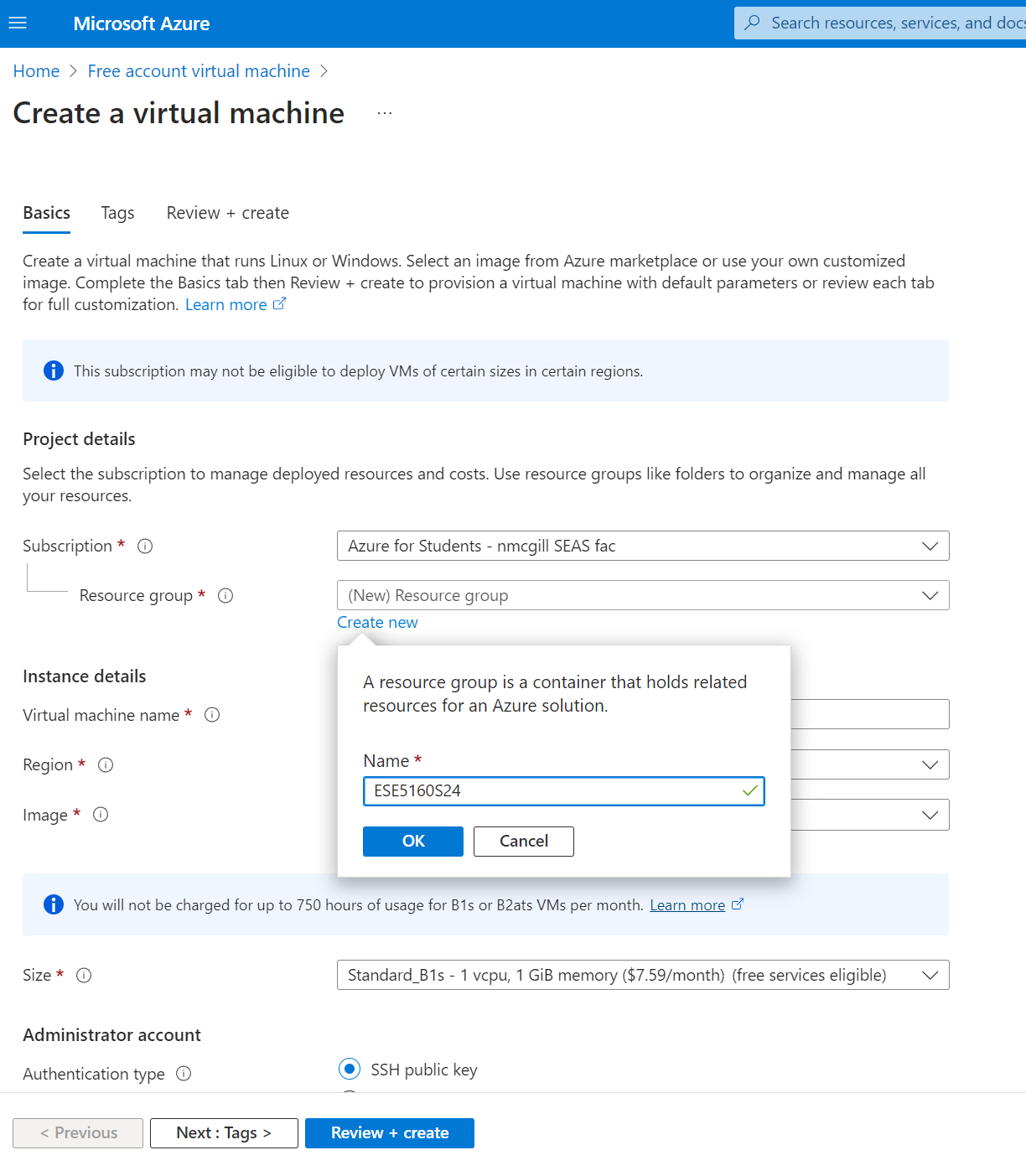
1. Sign up for a free student account with $100 worth of credits [here](https://azure.microsoft.com/en-us/free/students/).
   1. You don’t need to pay for this account and we don’t expect you to pay for it!
   2. Besides these credits, there’s a **Free account virtual machine** that you can use:  
      
   3. Make sure to use your [pennkey]**@upenn.edu** email address (Not the @**seas**.upenn.edu)  
      

## 1.2 Refer to Node-RED’s Azure Setup Instructions

1. Node-RED’s website has some (outdated) instructions for setting up a Node-RED instance on an Azure Virtual Machine [here](https://nodered.org/docs/getting-started/azure).
2. They’re pretty good. Hopefully this manual will fill in the rest.  
   

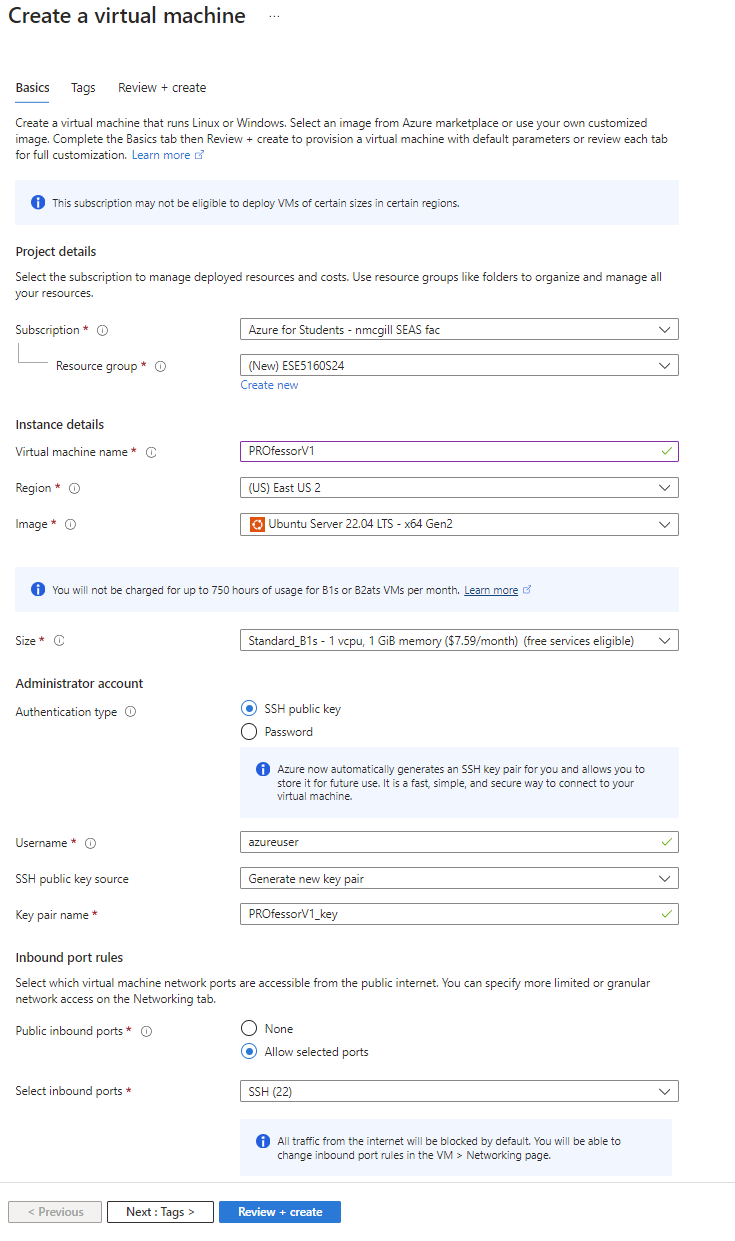
## 1.3 Free Azure VM Setup

Go to [Azure Free account virtual machine](https://azuremarketplace.microsoft.com/en/marketplace/apps/microsoft.freeaccountvirtualmachine?tab=Overview) and create an Azure virtual machine  


Follow the steps to create this free virtual machine as part of your profile.  


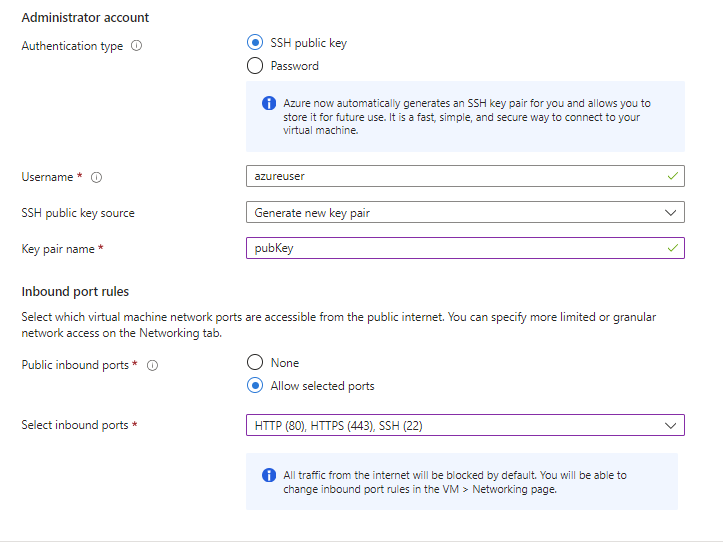
## 1.4 Create a virtual machine - Basics Tab 1

1. Add a Virtual machine name
2. Set the Region to East US
3. Use a Ubuntu Server image
4. **Size** should match the **free services eligible** options.
   1. The **Standard\_B1s - 1 vcpu, 1GiB memory** option was free and will be more than enough for our needs!



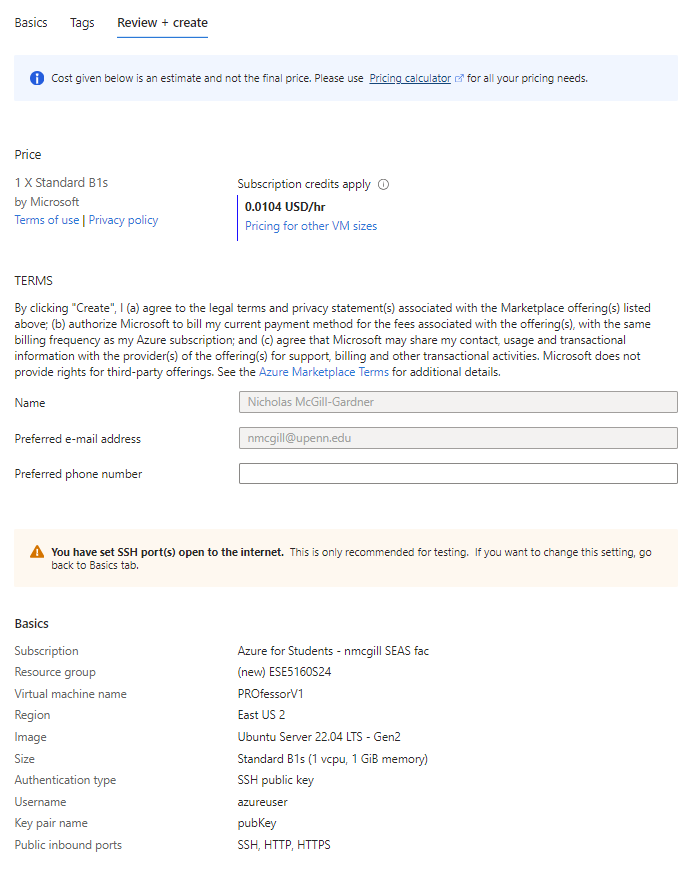
## 1.5 Create a virtual machine - Basics Tab 2

1. Set your **Authentication type**.
2. Set up your preference:
   1. **SSH public key** will make it simple to connect to your instance over an SSH connection
   2. **Username** and **Password** might be easier if you’re not familiar with SSH
   3. Resources:
      1. [Troubleshoot SSH connection issues to an Azure VM](https://learn.microsoft.com/en-us/troubleshoot/azure/virtual-machines/linux/troubleshoot-ssh-connection)
      2. [Reset Password of an Azure Virtual Machine VM | Microsoft Learn](https://learn.microsoft.com/en-us/shows/it-ops-talk/reset-password-of-an-azure-virtual-machine-vm)
3. Allow Public inbound ports for **HTTP (80), HTTPS (443), and SSH (22)**



## 1.6 Create a virtual machine - Review + create

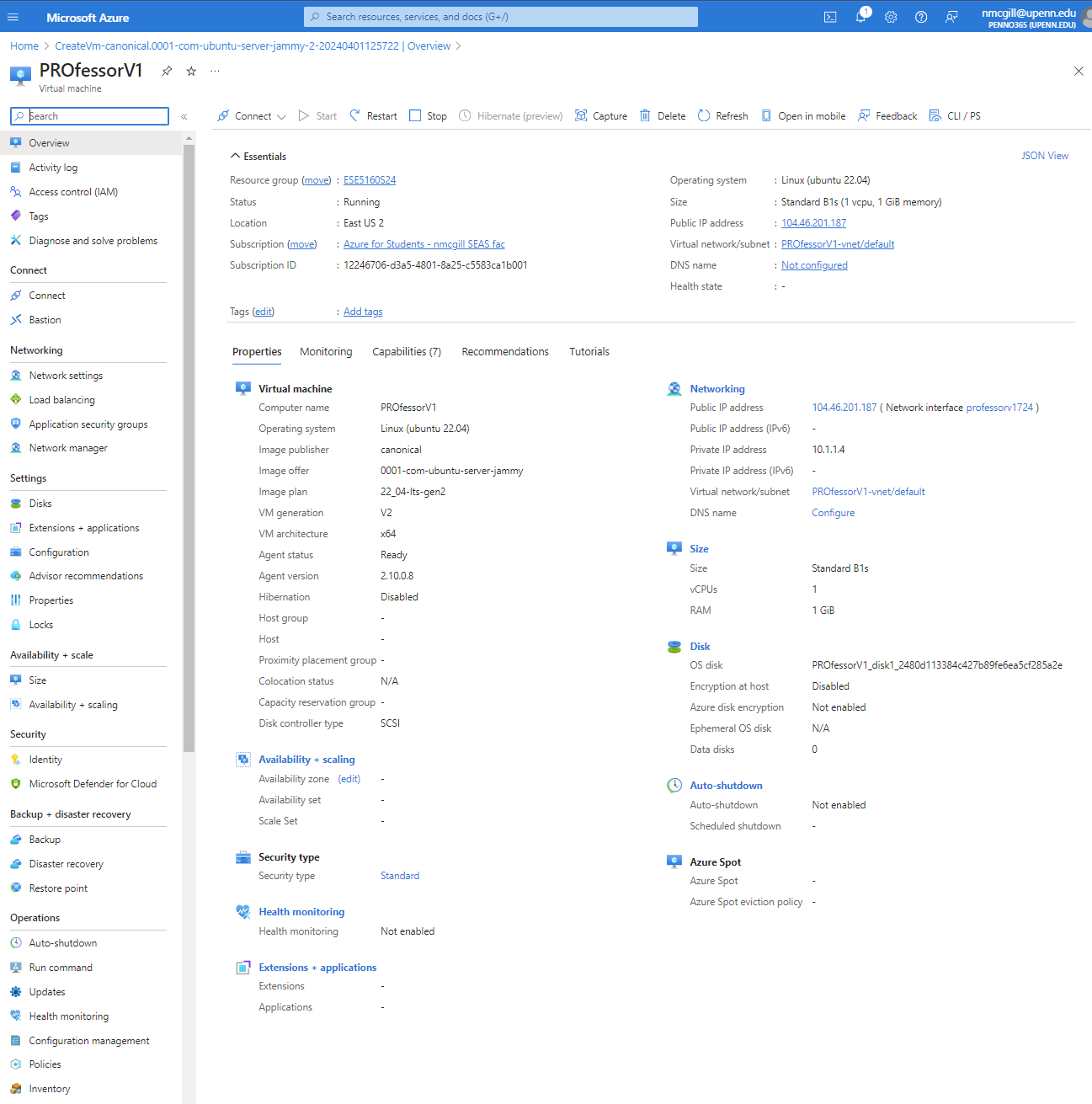
1. You can skip the **Tags** tab and go right on to **Review + create**.
2. You’ll have student account credits to use for this. Again, this should be free for you! If you’ve used all your credits on some other class, you may need to pay for this yourself.



## 1.8 VM Overview

Your deployment will be in progress for a bit, then will be accessible. Congrats! You have started your Virtual Machine instance! Take a look around!

Find your way to the Overview and you can find your public IP address.

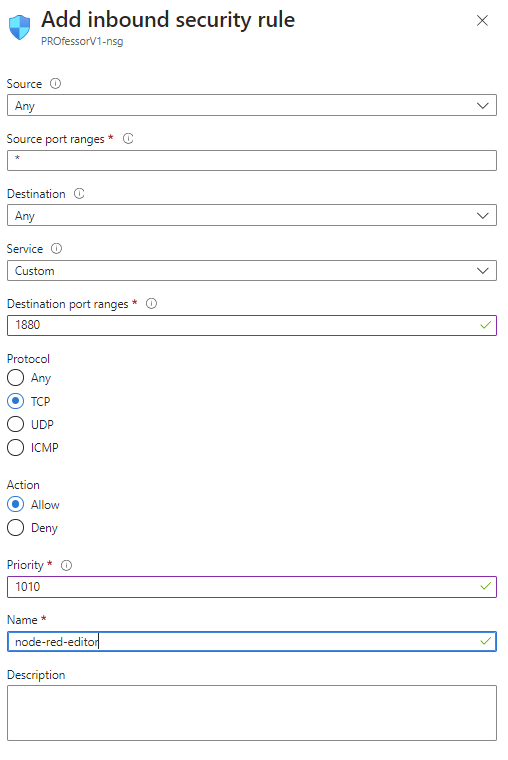


## 

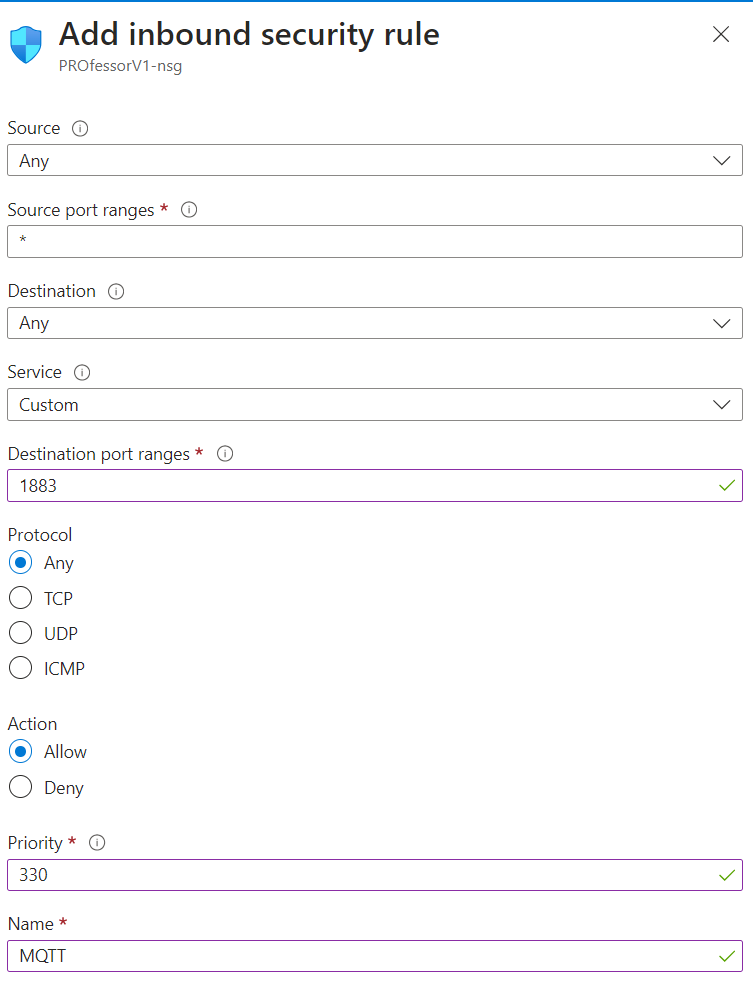
## 1.8 Add inbound security rules

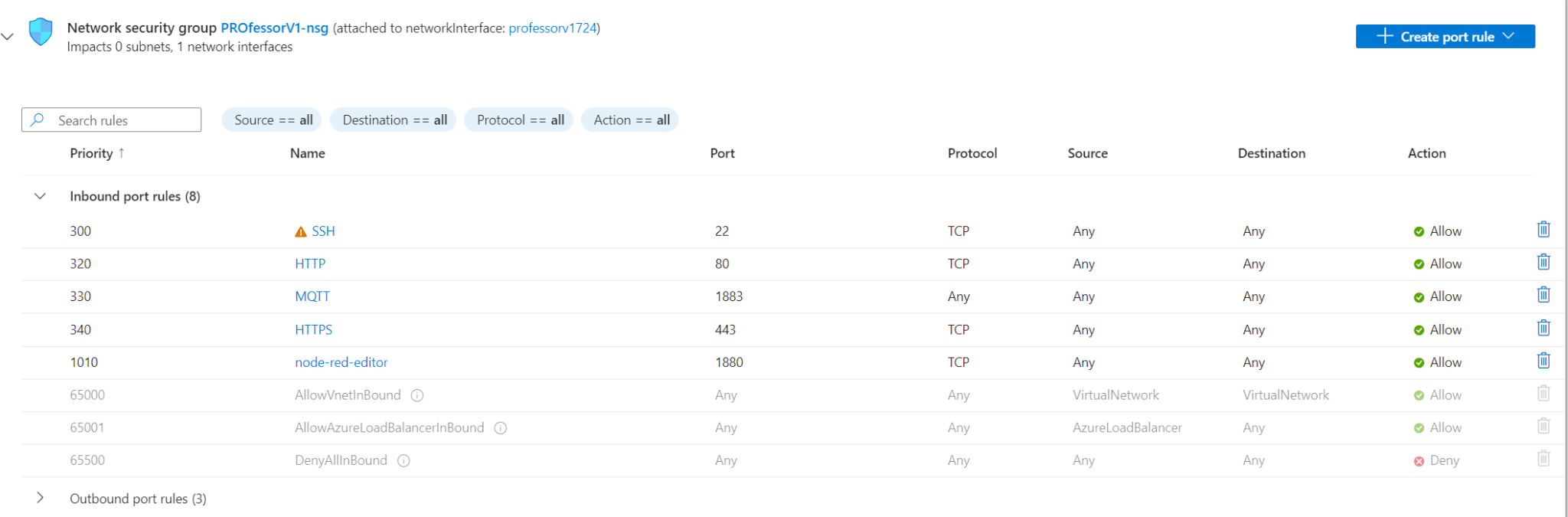
We’ve exposed SSH, HTTP, and HTTPS ports so far for our development. We need to add one more so that the Node-RED editor port is accessible by our web browser.

1. Go to the **Network settings** page.
2. Click on **Create port rule > Inbound port rule.**  Use the settings below to set up the Node-RED editor port, 1880.



1. Click on **Create port rule > Inbound port rule.**  Use the settings below to set up the MQTT broker editor port, 1883.



Now, you’ll see the port rules have been added in the **Rules** panel:  


# 2. Hosting the OTAU firmware binaries & text file

Before you start the following, **make sure you have already set up your Azure Virtual Machine.**

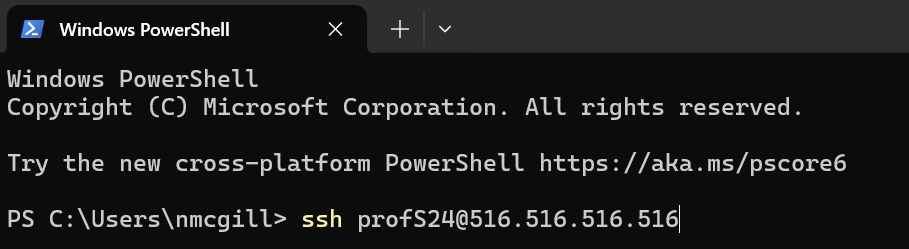
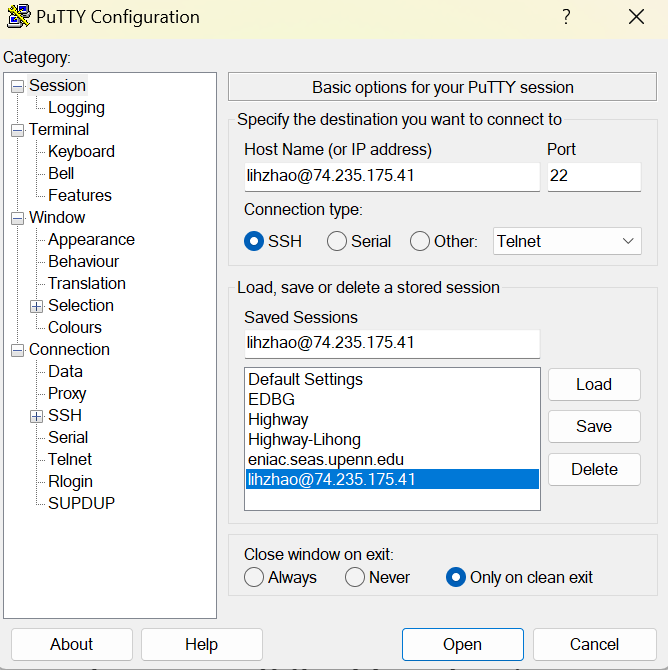
You’re building an Internet-connected device that can execute over the air firmware updates (OTAU). You’ve already built the bootloader - this can load new firmware from the SD card. Now, we want to add the firmware images to an online server so our device can download new images to the SD card.

## 2.1 SSH from your computer to the Azure VM

We’re going to set up some things on the Ubuntu operating system running on our Azure Virtual Machine. To do that, we’ll connect over SSH (secure shell). [This article](https://medium.com/@shrutipal700/hosting-a-website-on-ubuntu-virtual-machine-36598ade82a1) was referenced for these instructions.

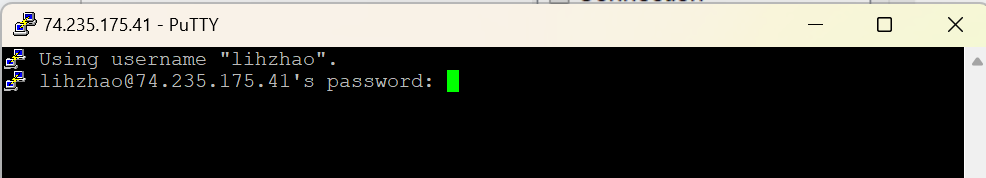
1. Use a program that connects over SSH to servers, such as [Windows PowerShell](https://learn.microsoft.com/en-us/powershell/scripting/what-is-windows-powershell?view=powershell-7.4) or [PuTTY](https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html)
2. **SSH** into your Azure VM using **[username]@[public ip]**

* You set the username through your VM setup.
* The Public IP is on your VM overview page.

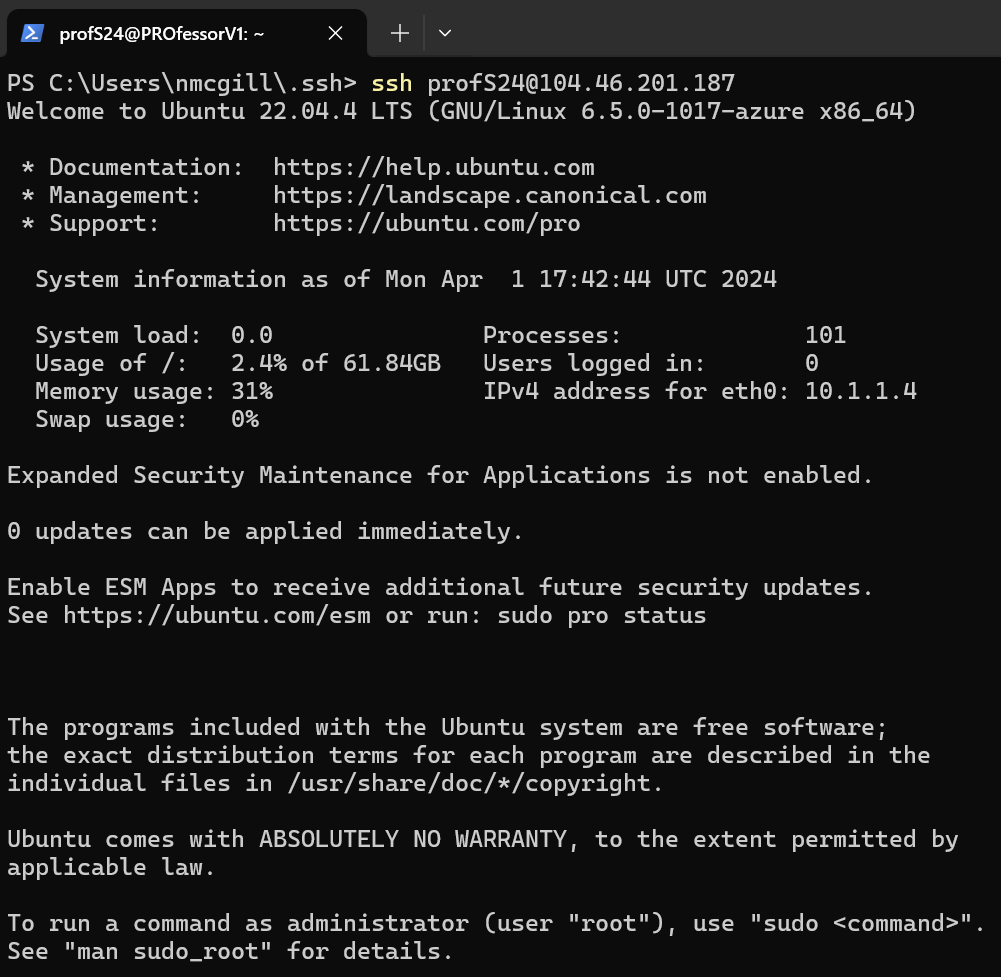


1. If you used SSH keys, you should be able to connect immediately. If you chose username and password, you’ll need to enter those appropriately.

* Note that you may get a “fingerprint” check. This is to ensure you’re connecting to the correct hardware, the Azure VM. You’ll type **yes** if this is true.
* Note that the password will not be displayed, after you have entered your full password, press **Enter** to log in.



After logging in, you will see the following screen.



## 2.2 Install Apache web server

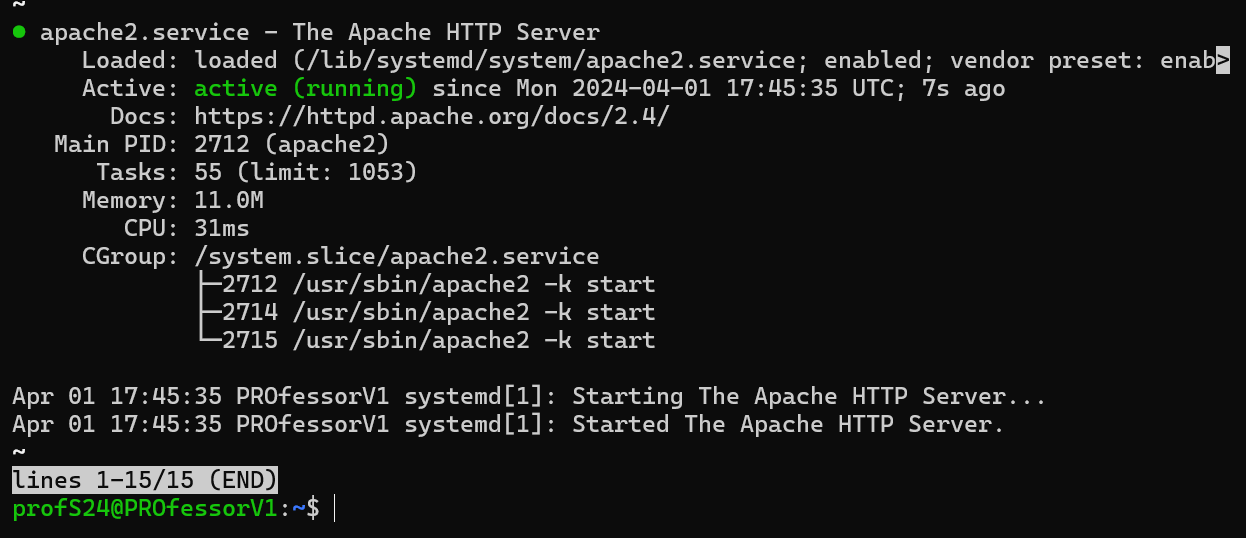
Apache is a popular web server tool. It’s the mechanism by which we can put files (like our firmware images) on a server to be made available publicly online.

1. Run the following commands to update the VM’s packages, install Apache, and then check the status of the apache2 service:

sudo apt update

sudo apt install apache2

sudo systemctl status apache2

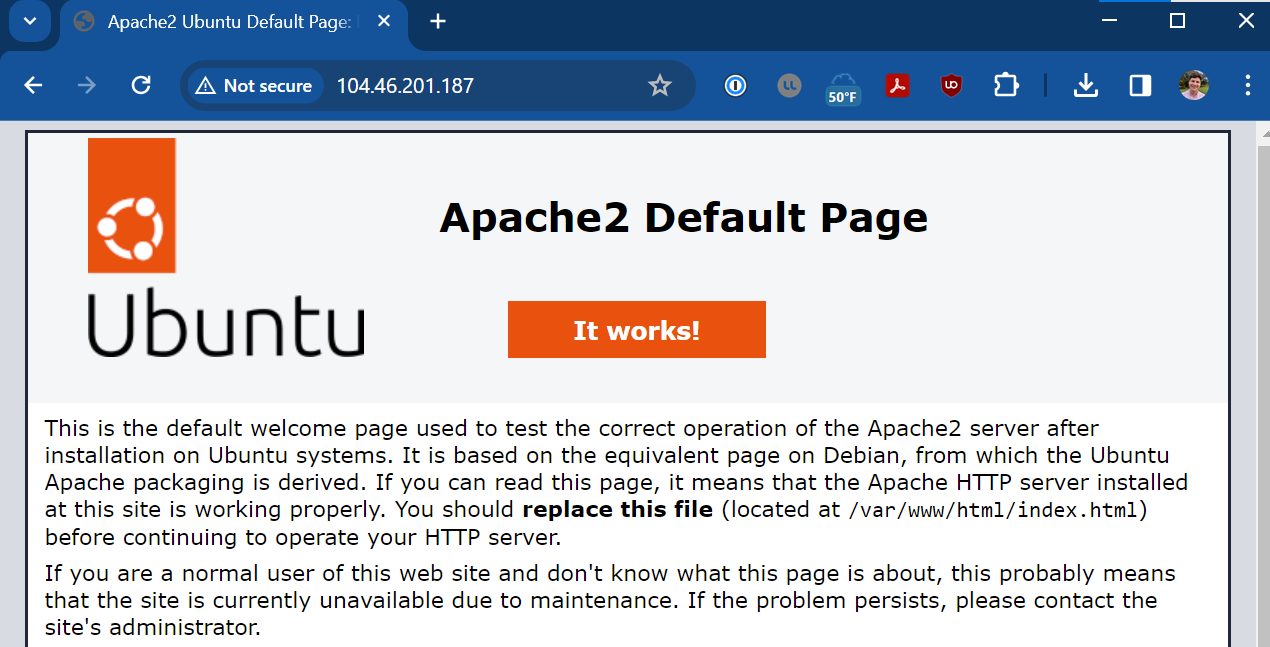
****

1. Check if the Apache webserver is running by accessing your public IP in the web browser:

**http://[public IP]/**

**Make sure you don’t use https!**

If you see the webpage below, the Apache Server is working



If you don’t see this, try running the Apache restart command:

**sudo systemctl restart apache2**

1. The Apache web server makes files in the **/var/www/html/** folder visible to the Internet. By default, you won’t have write access here. That’s a bummer because we need to write/upload files to be downloaded for OTAU!
   1. Run this in your SSH session to the Azure VM:  
      **sudo chown -R $USER:$USER /var/www**

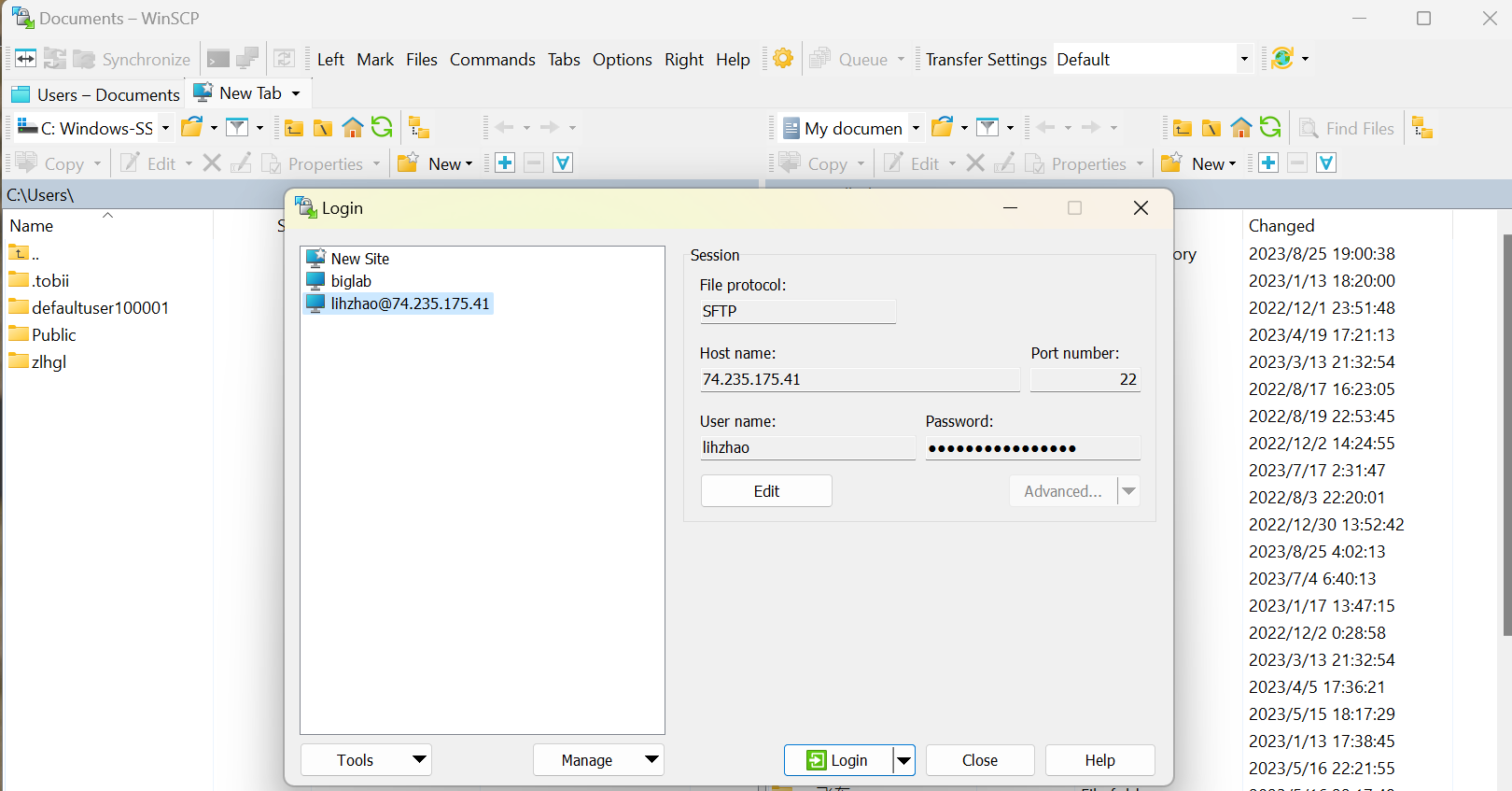
## 2.3 File Upload

These instructions are loosely based on [this webpage](https://winscp.net/eng/docs/guide_microsoft_azure).

1. Upload test files (PDFs, binaries, etc) to the Azure VM in the Apache web server folder.

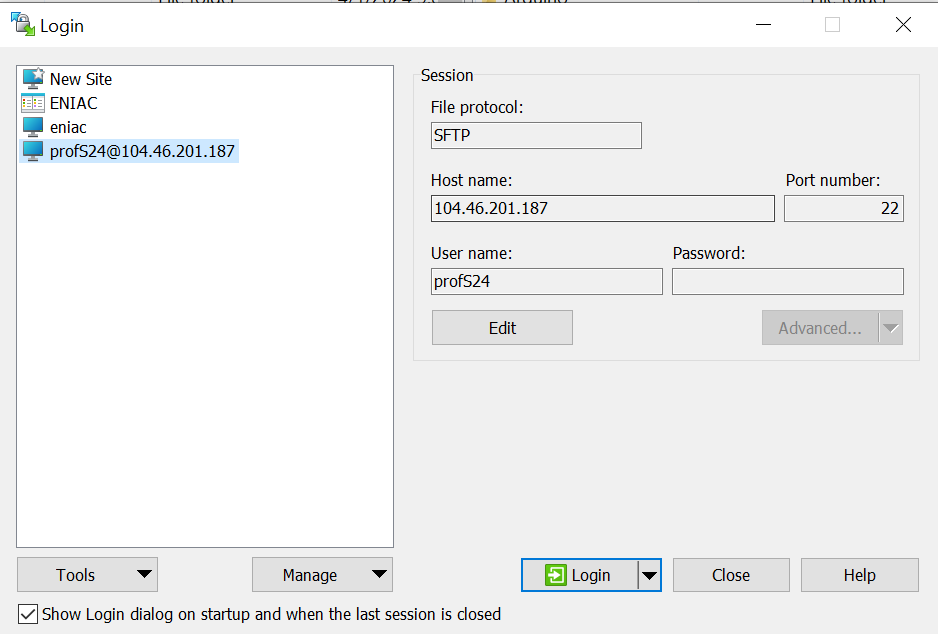
* I used [WinSCP](https://winscp.net/eng/index.php) to do that.

1. Click **New Tab**. A login page will appear.

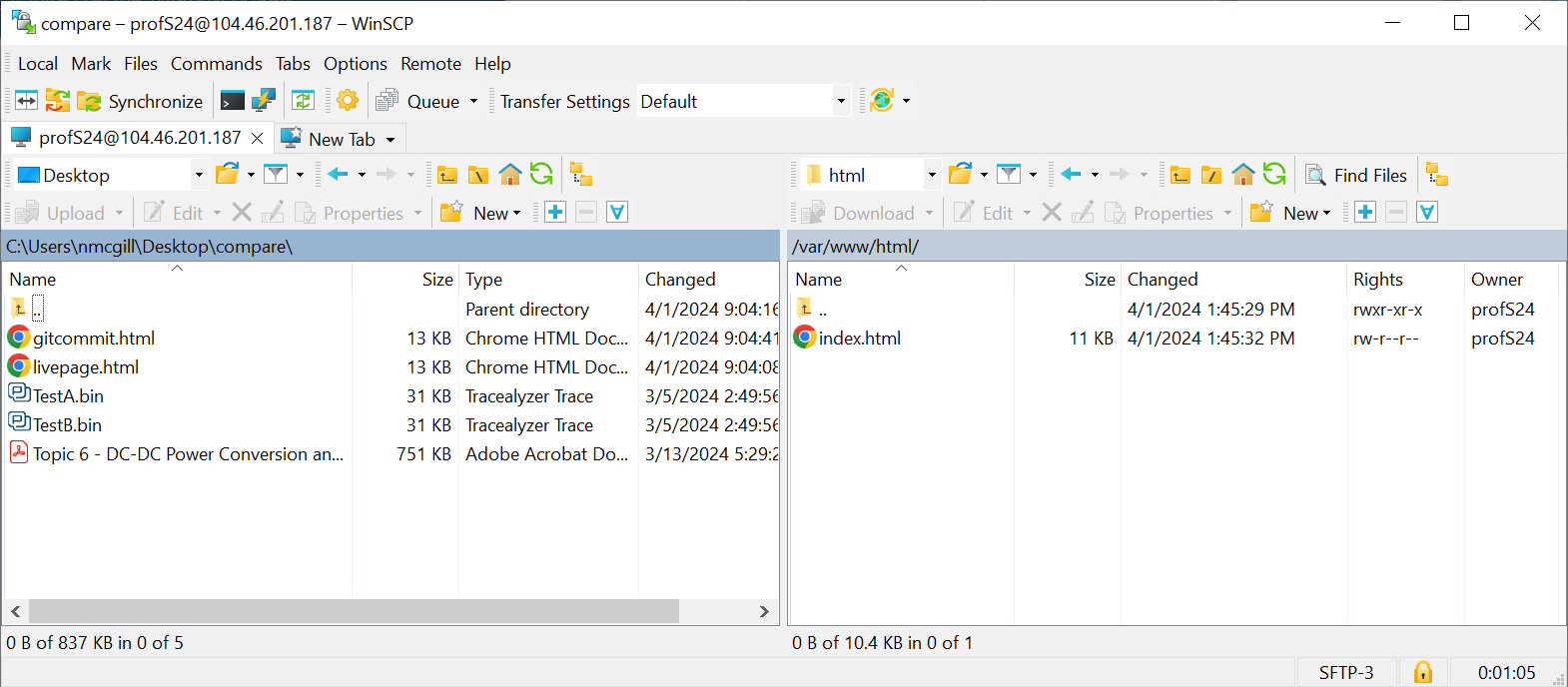


1. For the login information

* File protocol: SFTP
* Host name: [Public IP]
* Port number: 22
* User name: [VM user login]
* Password: [VM user password] (or blank for SSH key, see below)
* If using an SSH key: [Connecting securely to Microsoft Azure service with SFTP or FTPS :: WinSCP](https://winscp.net/eng/docs/guide_microsoft_azure)



1. Then, click **login**, then you’ll see your computer files on the left and files on your Azure VM on the right. You can upload files on your Azure VM by **dragging** and **dropping** files from left to right.



1. Drag a file, link an image of PDF, into the Azure VM server folder: /var/www/html/. Check using the web browser that you can access it. You’ve successfully added publicly accessible files to your server!

## 2.4 Firmware: Configure for HTTP

1. Firmware updates:
   1. Because this solution uses HTTP, not HTTPS, for downloading files, we simply need to comment out the HTTPS related configuration in the function **static void configure\_http\_client(void)**
      1. [Port should be 80](https://www.cloudflare.com/learning/network-layer/what-is-a-computer-port/), TLS should be 0 (this is the default when running **http\_client\_get\_config\_defaults**)
2. Finally, update your download URL in your firmware.

**http://[Public IP]/[File Name]**

Example:

http://74.235.175.41/TestA.bin

With these updates, you should have no problem downloading files from your Azure VM and storing them on your SD Card!

# 3. Install Node-RED on your VM

Before you start the following, **make sure you have already set up your Azure Virtual Machine.**

## 3.1 Add inbound security rule

If you haven’t already done it, review [1.7 Add inbound security rule](#_ipfc09e9ikgc) to grant access for the Node-RED editor port.

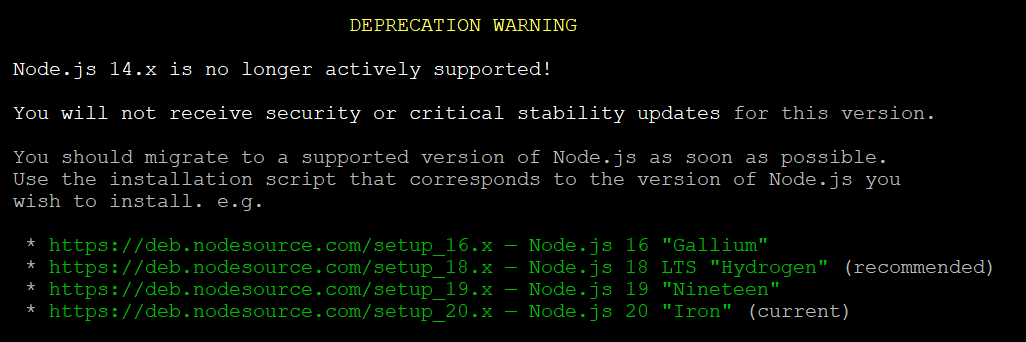
## 3.2 Install Node-RED on your VM

Please refer to [2.1 SSH from your computer to the Azure VM](#_lch6cvvewvwy) if you need a reminder how to connect via SSH to the Azure VM.

1. Log in over SSH to your VM and then run the following command

**curl -sL https://deb.nodesource.com/setup\_20.x | sudo -E bash -**

If the below information comes up, wait around 60s to let the system fetch the newest version. Reference: <https://github.com/nodejs/Release>



Then run:

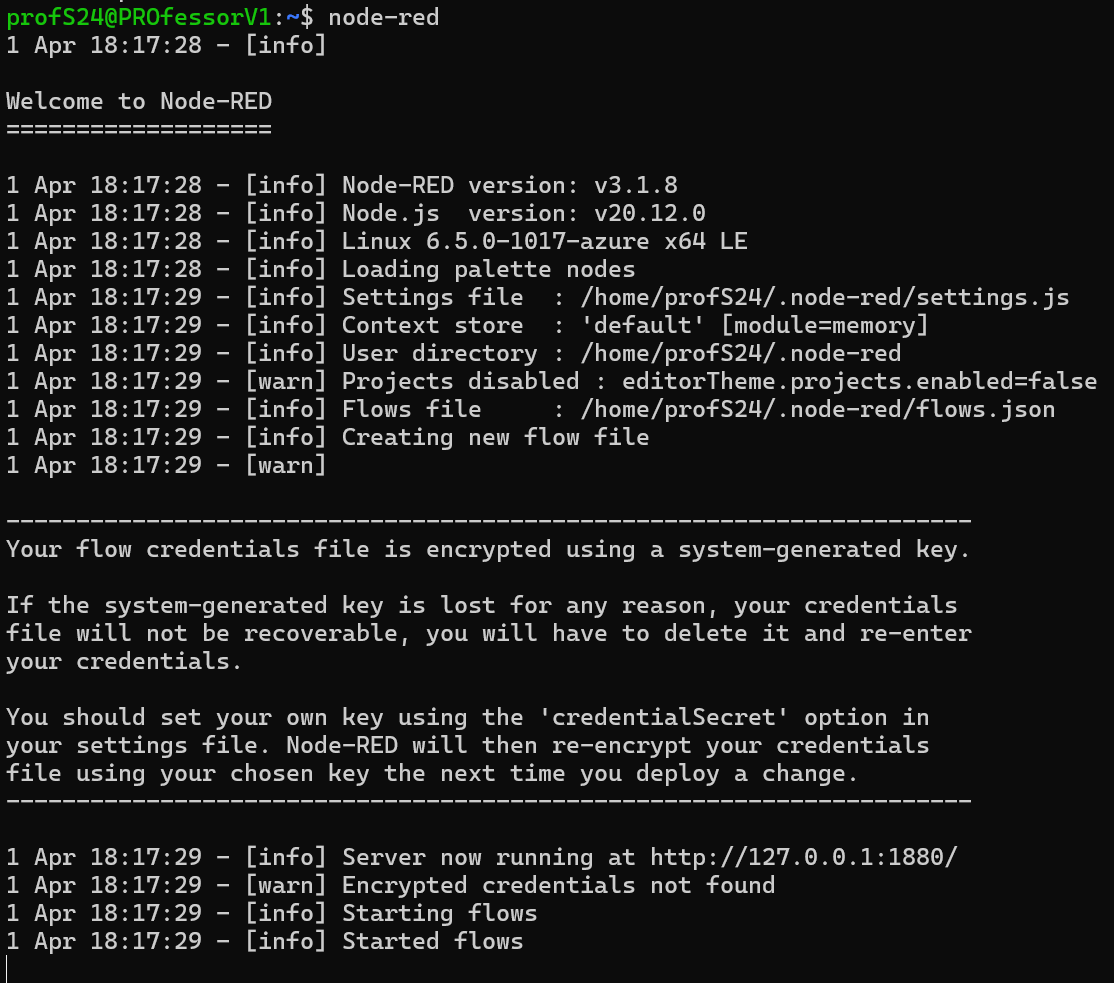
**sudo apt-get install -y nodejs**

Then, run,

**sudo apt-get install -y nodejs build-essential**

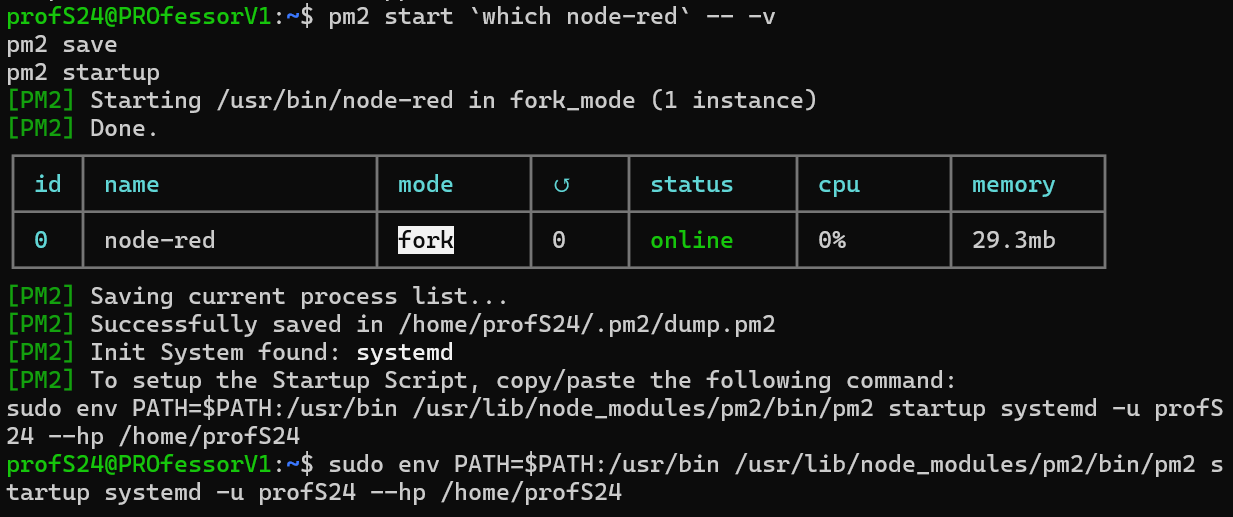
**sudo npm install -g --unsafe-perm node-red  
sudo npm i -g pm2**

1. Boot up Node-RED by entering **node-red** at the terminal.

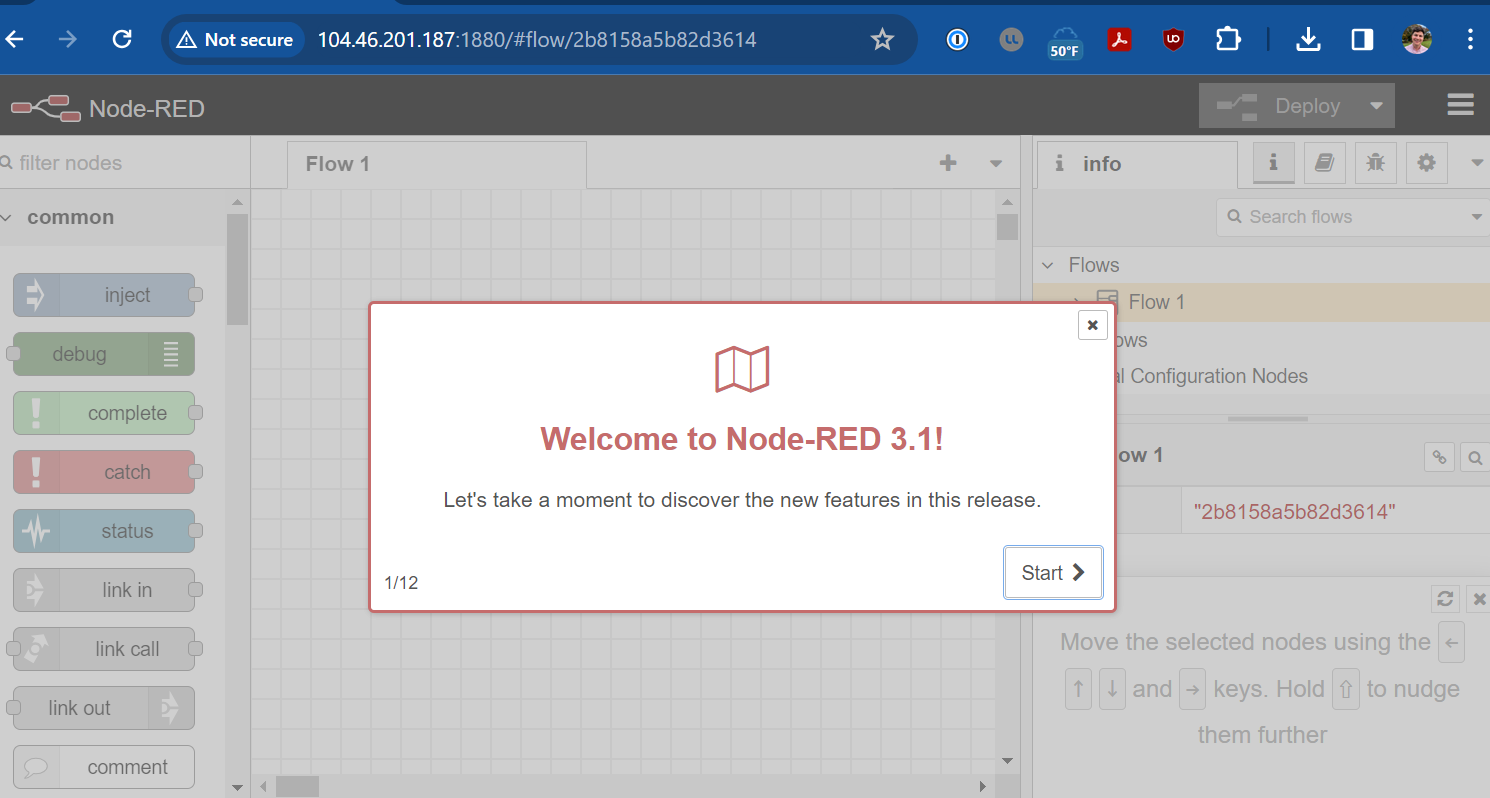


1. We don’t want to type **node-red** every time we boot up our VM. Use the commands below to have Node-RED autostart: **pm2 start `which node-red` -- -v  
   pm2 save  
   pm2 startup**

And then run the command given in the terminal output to setup the Startup Script.

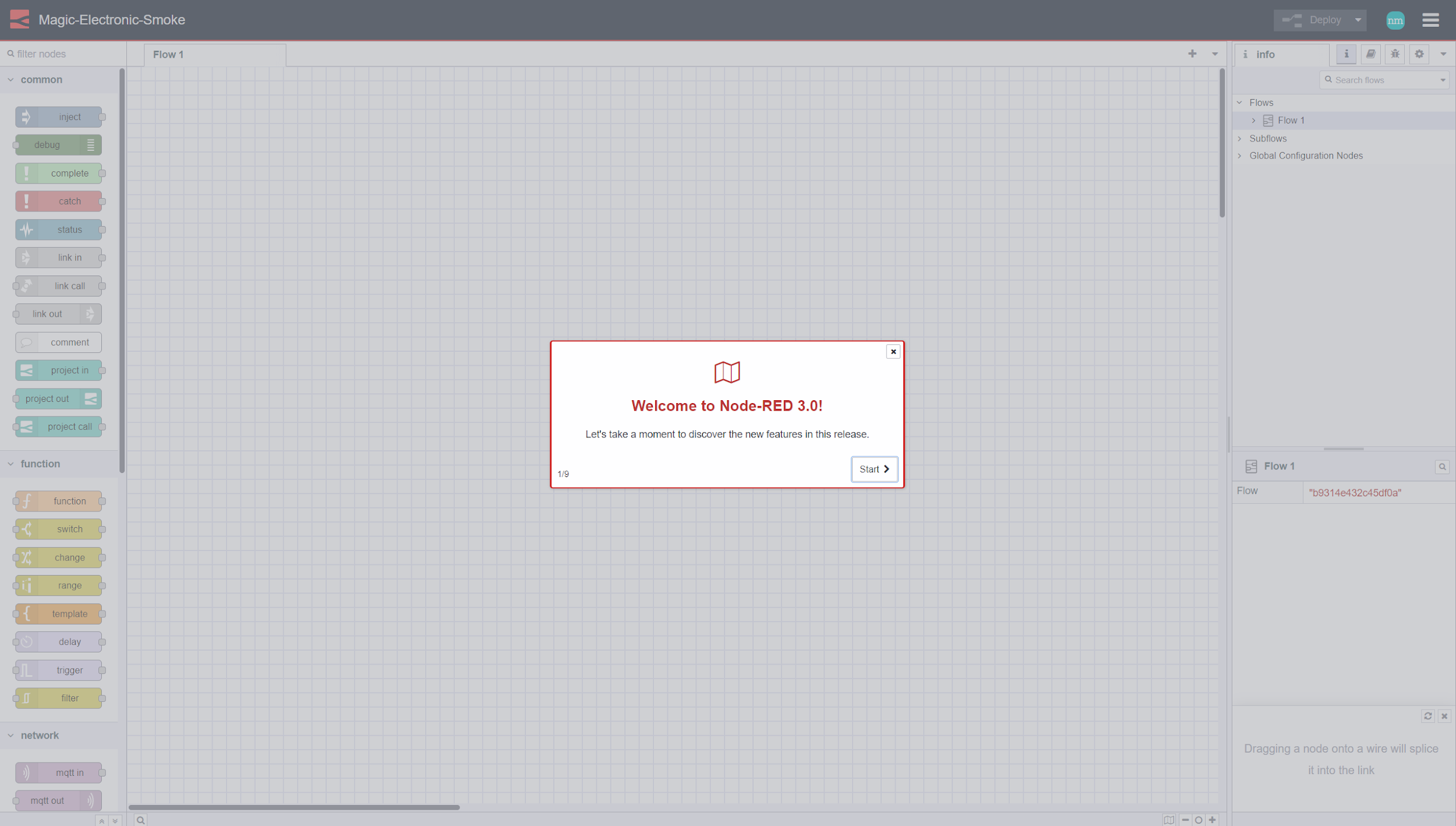
****

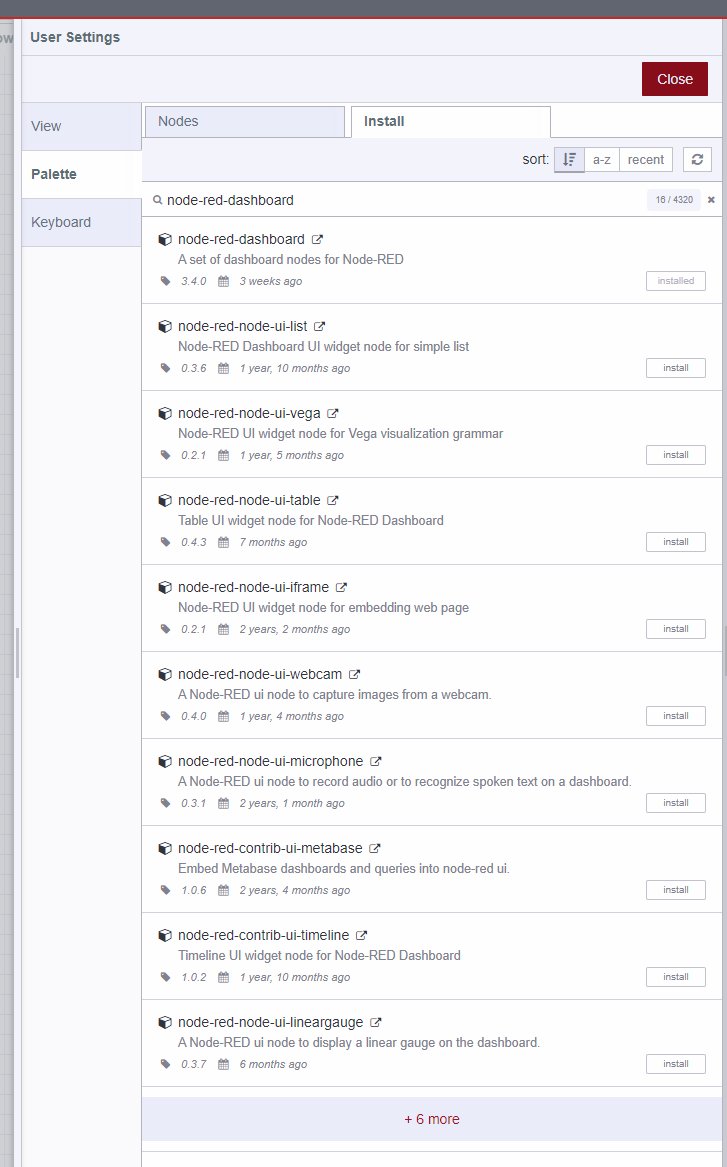
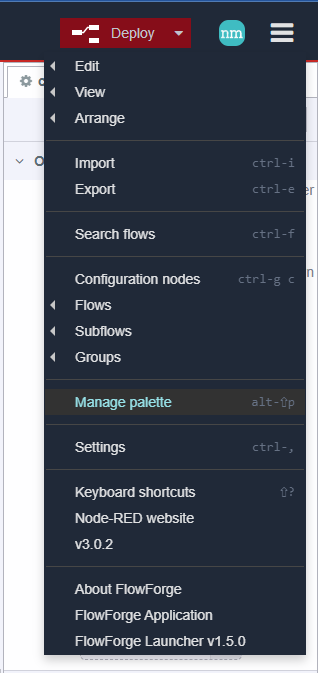
1. Note that Node-RED will be running on the Azure Virtual Machine on some server on the East Coast of the United States. The IP address for the Azure VM is on the overview page. However, you’ll see messages about the Node-RED server running at [**http://127.0.0.1:1880/**](http://127.0.0.1:1880/). This is a special IP address called the **localhost**.
   1. You’ll need to find your **public IP** address of the Azure VM server on your Azure dashboard in order to view your Node-RED instance.
   2. Find that public IP now and access port 1880. You should see something like the following, the editor for Node-RED.

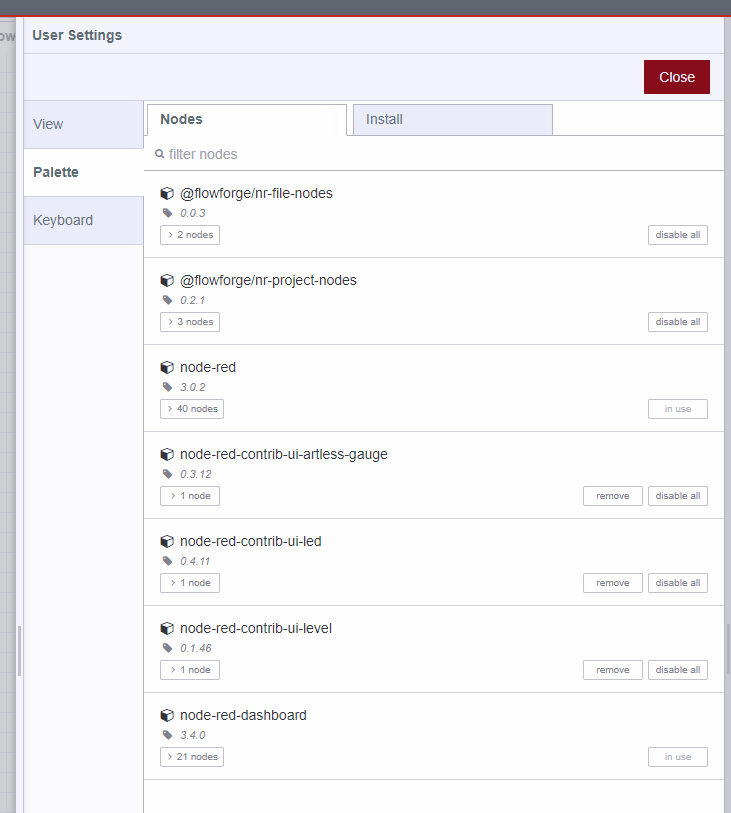


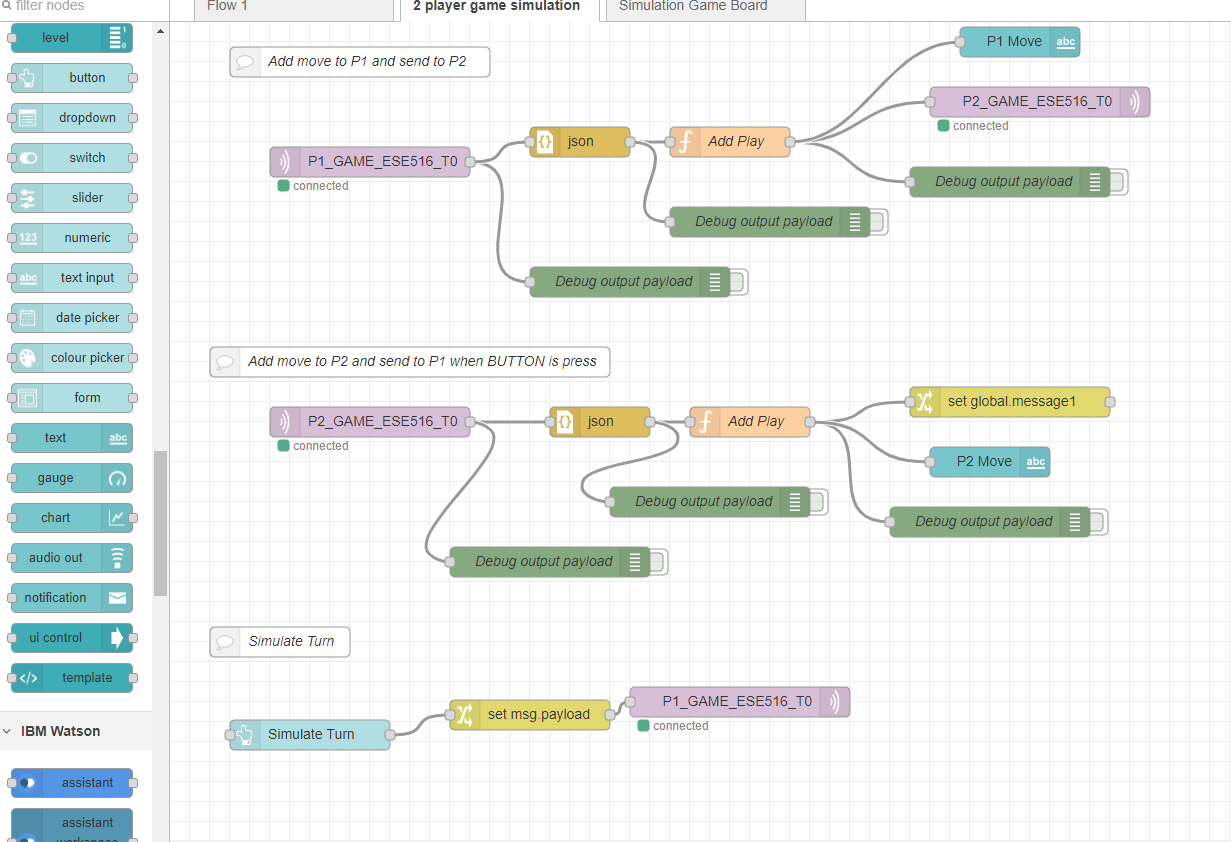
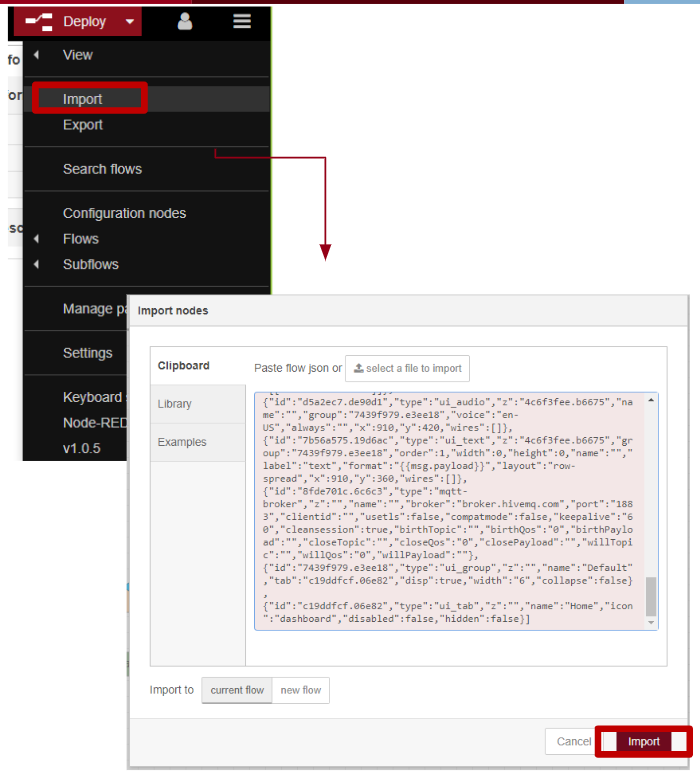
# 4. Node-RED Configuration

## 4.1 Add functionality with nodes

1. First launch will look like this:
2. The Palette Manager installs nodes for Node-RED. We’ll need to install the following nodes for our class:
   1. node-red-dashboard
   2. node-red-contrib-ui-artless-gauge
   3. node-red-contrib-ui-led
   4. node-red-contrib-ui-level



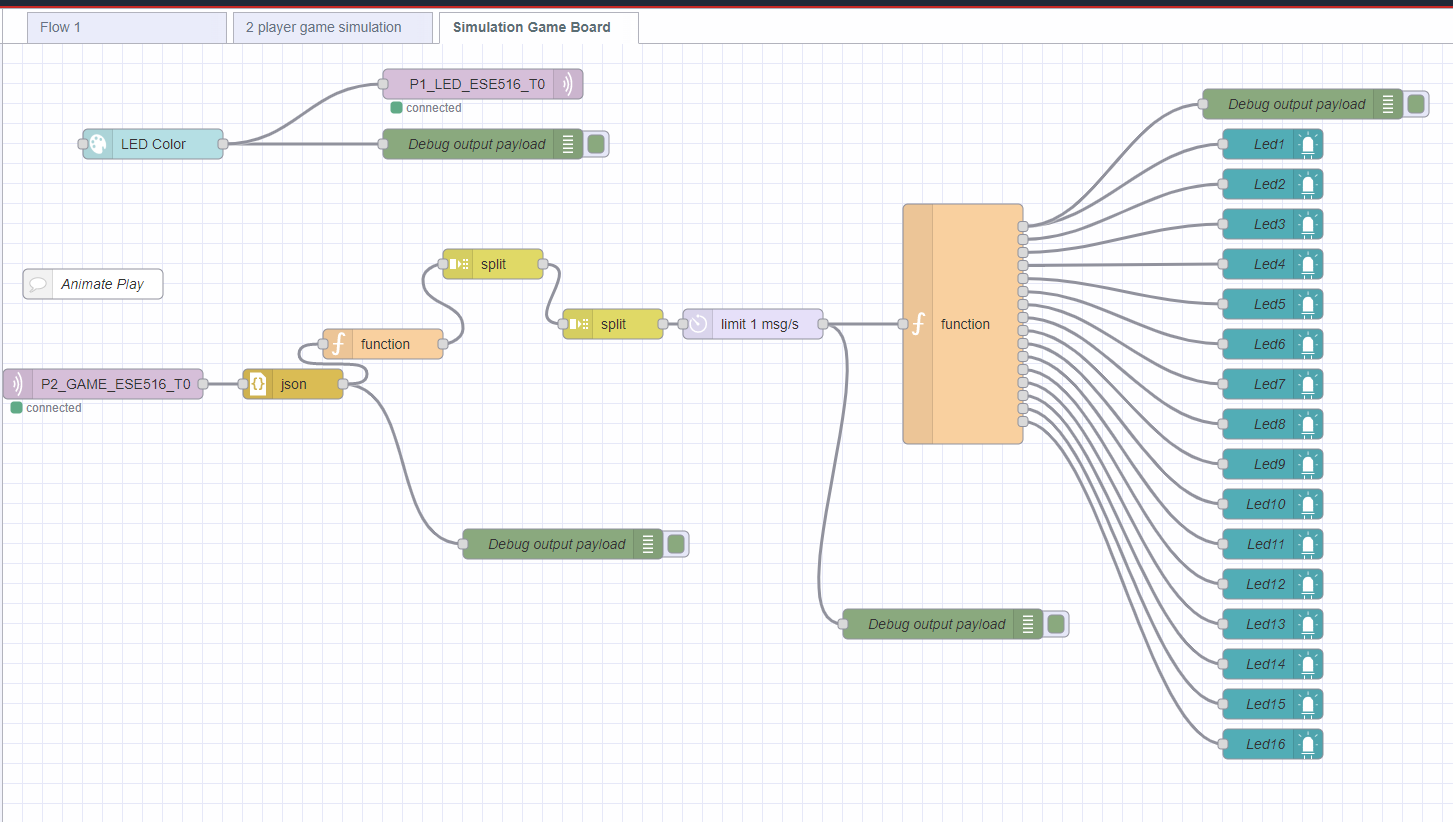


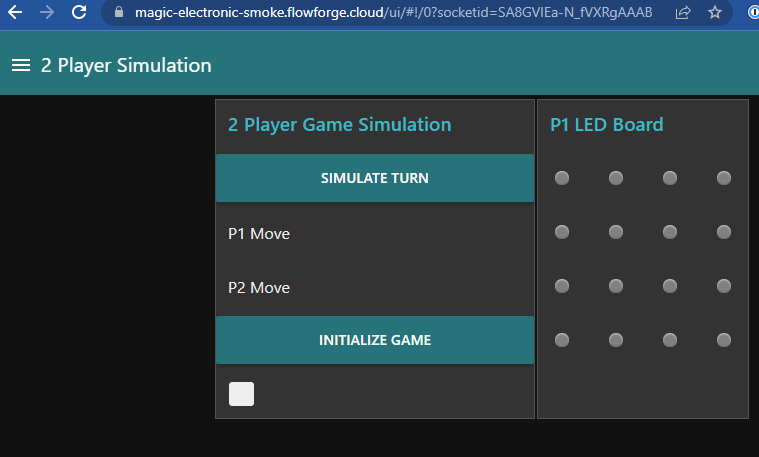


## 4.3 (Optional) Explore an Example UI

If you’d like to get familiar with what you can do in Node-RED, you can import the Simon Says example from a couple years ago. This will also show you how you can backup your work as a JSON file and import it to new Node-RED instances. **If you do this, just make sure you remove any unused parts of the Node-RED interface when you’re done!**

1. The Google Drive Assignments folder has the file **A10G node-red-game-simulator-example.json**. Import this through the Node-RED interface.
2. Take a look at the flows that create the user interface, interface with communications, and make the whole setup work.
3. You can now find your user interface (UI) by entering your instance URL followed by **/ui**
   1. This will vary based on your Azure VM public IP address or hostname!





## 4.4 Node Descriptions for Game Simulator

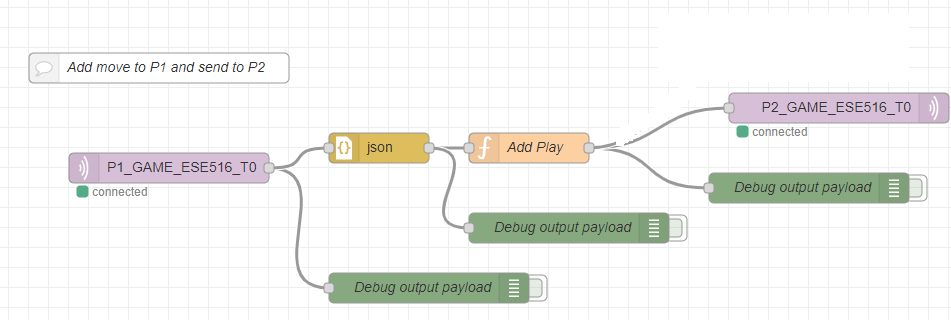
### MQTT In Node

The MQTT IN NODE listens to a command received from a broker. If you double click it you can edit what it listens to! This is connected to an MQTT broker.



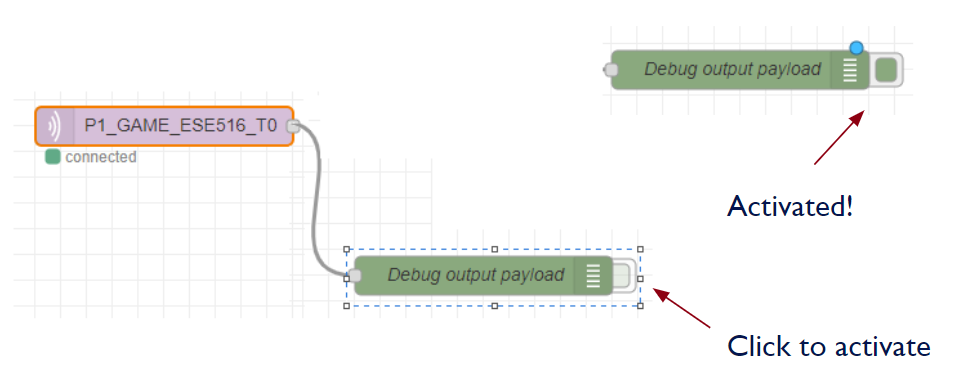
### MQTT Out Node

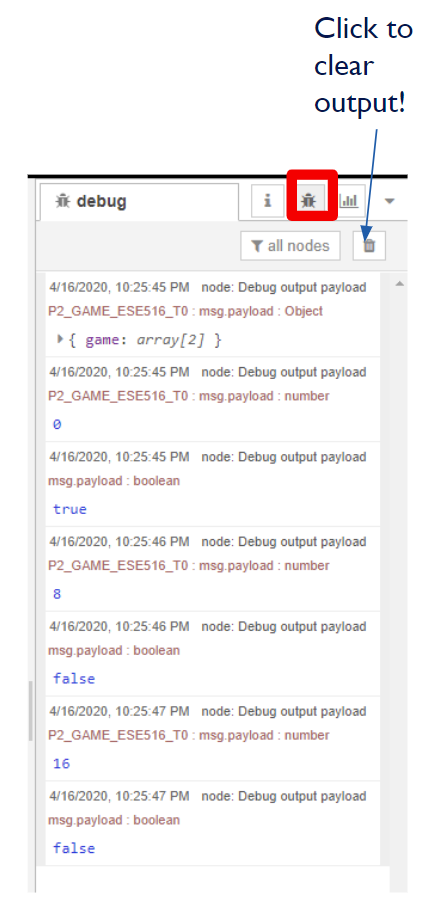
MQTT Out button: Similar to the MQTT in, the MQTT Out will publish data to an MQTT Topic. In this case, we publish to the player 2 (P2\_GAME\_ESE516\_T0).



### Debug Node

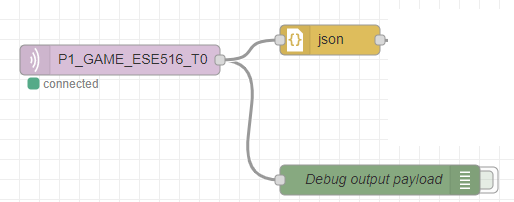
Debug NODE: Connected to this node is a DEBUG NODE. A debug node will print whatever it is connected to on the debug window. You can get to this window on the right screen of the backend view of Node-RED (where you see the nodes). You can click on the green rectangle to activate/deactivate debug output.





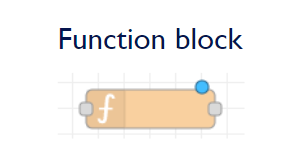
### JSON Block

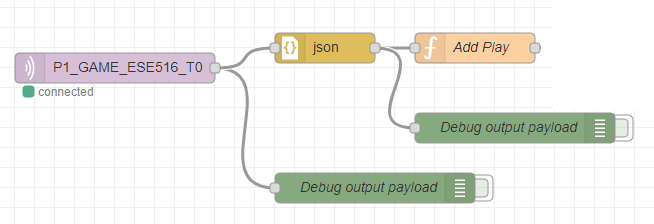
The JSON block converts a JSON string into a Javascript Object. This will allow us to use Javascript on the data later on!

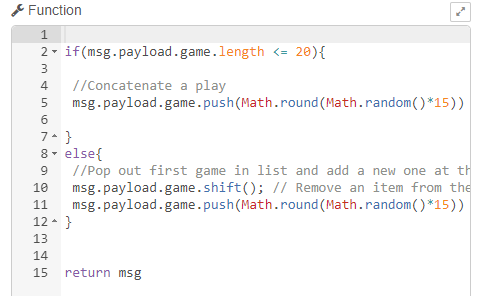


### Javascript Block (Function)

The function block can be added to run Javascript code! Double click on “Add Play” to see its code. In our example, this block is used to add a game play (random number between 0 to 15) to the end of the received play!

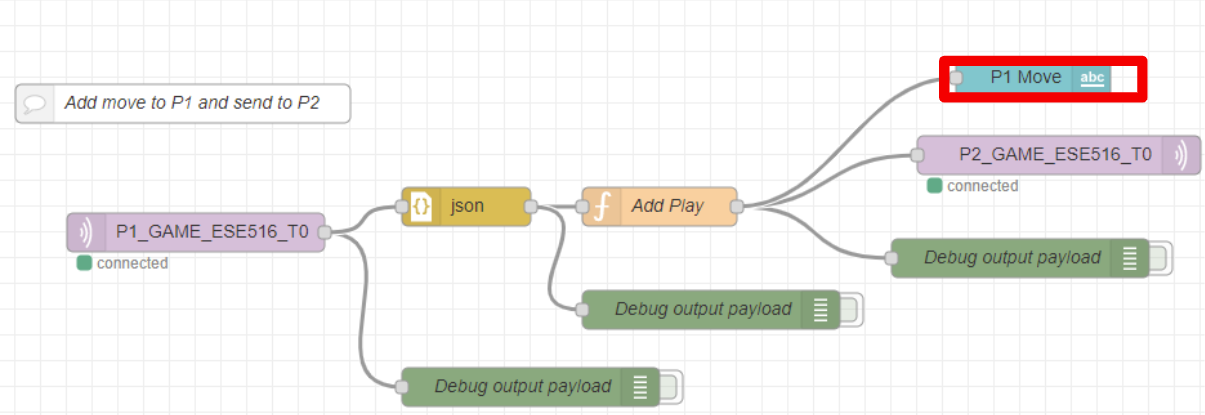
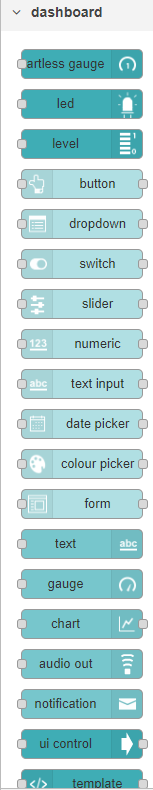
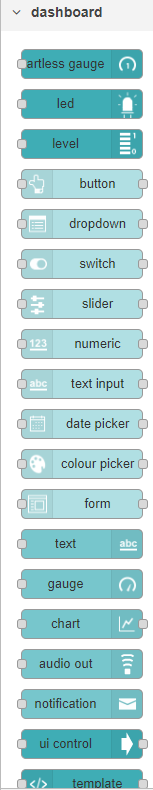




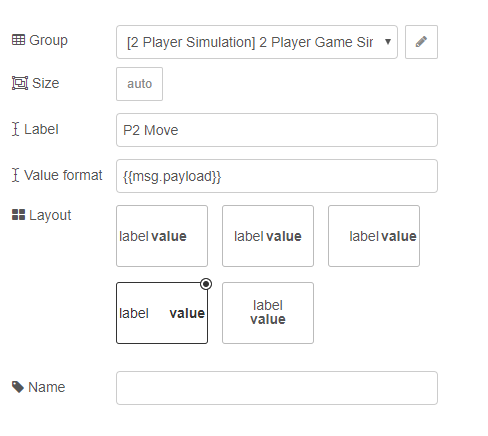


### Front end (UI) Blocks

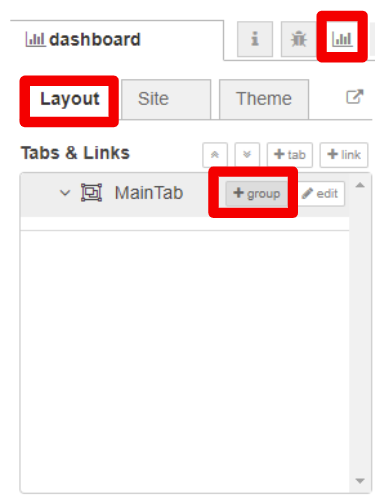
There are going to be special nodes (listed under DASHBOARD) that will allow us to develop a front end GUI. In this example, we use a text label that prints the output to P2.



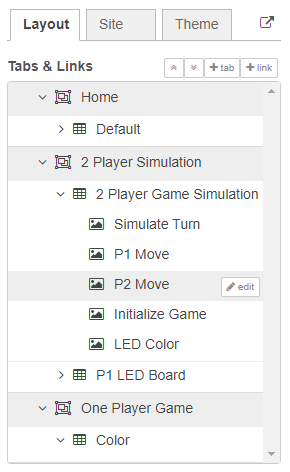
When you double click a dashboard component you can modify how it will look! You can also assign to what group it will be added to (where it will be shown on the UI)



To configure the Layout of Dashboard block, you will need to configure the layout. To do so you can click on the top right layout button. On the layout tab you can add tabs and modify the position of existing DASHBOARD blocks into new groups.



Example of layout of our flow. For example the string we saw, P2 Move, is on the 2 PLAYER GAME SIMULATION Tab.



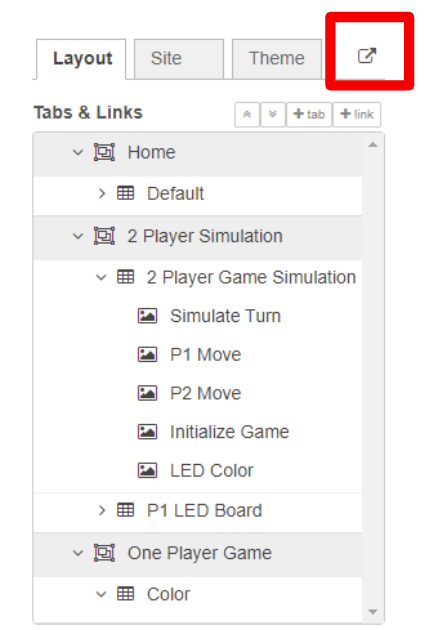
### Deploy

Once you made changes and want to deploy your new code, hit the “Deploy” button on the top right. This will deploy your flow! It will mention any issues that might arise if they are present.



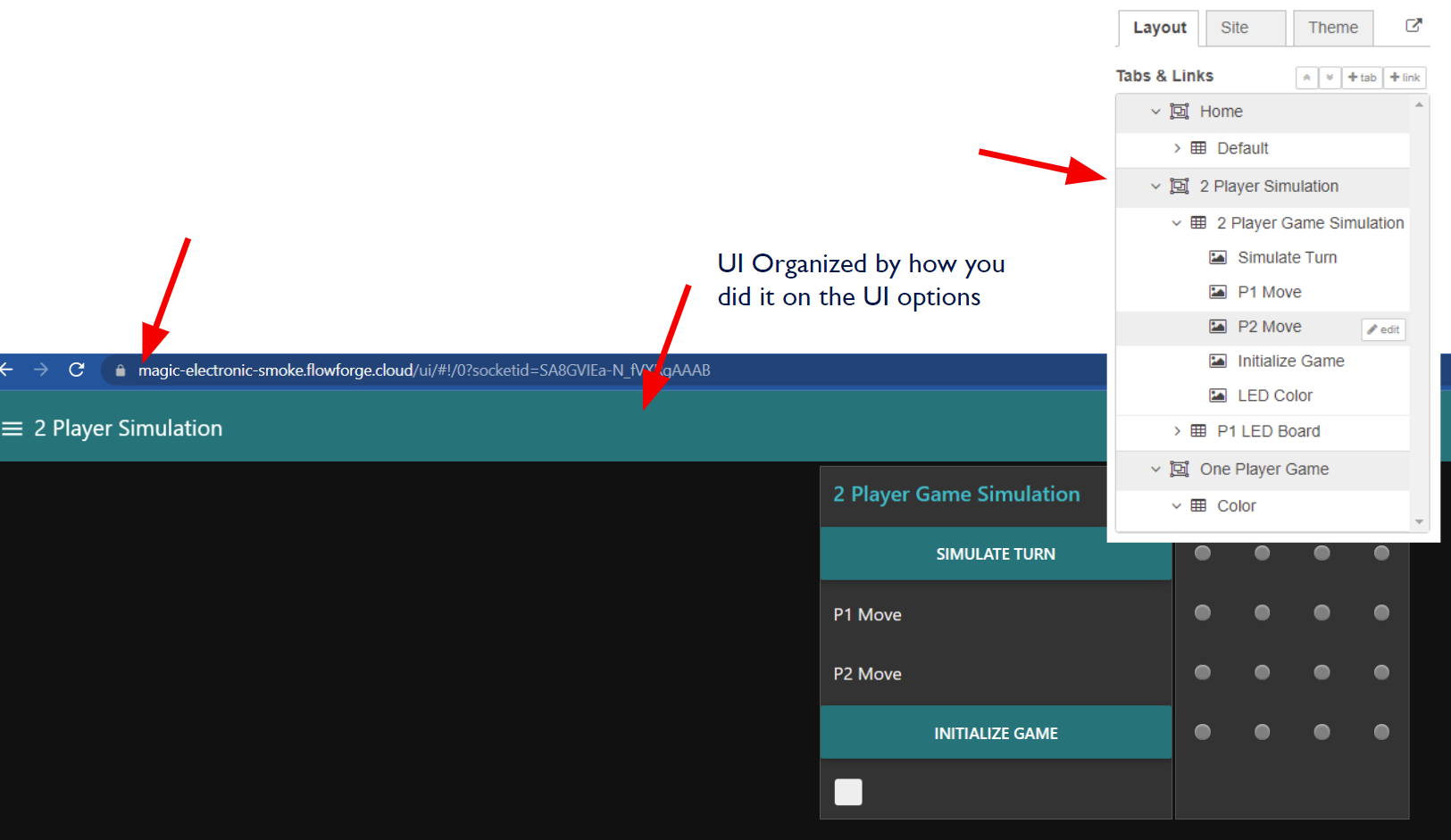


You can open the UI Website by clicking on the link in the Layout table.



### Node-RED Dashboard

Website you can access from anywhere to see your dashboard.



### Additional Node-RED Resources

[Node-RED Docs](https://nodered.org/docs/)

[Node-RED Cookbook](https://cookbook.nodered.org/)

[Node-RED Tutorial](http://noderedguide.com/)

# 

# 5. Hosting an MQTT broker

Thanks to [this webpage](https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/#mosquitto-no-authentication) for helping with this section. Microsoft’s training module [Deploy a private MQTT broker](https://learn.microsoft.com/en-us/training/modules/altair-azure-sphere-deploy-mqtt-broker/) is a good read too. We’ll be hosting our own MQTT broker using the open source [Eclipse Mosquitto Broker](https://mosquitto.org/). Remember, we need an MQTT broker to receive and send messages between MQTT devices - the Node-RED interface, your custom PCBA device, and any other MQTT based services / devices you may be using.

Happily, we’re able to host our own broker with our own settings on the same Azure VM that hosts our firmware binaries and our Node-RED UI!

## 5.1 Connect to Your Azure VM

Start your Azure VM and connect to it using SSH (like in [2.1 SSH from your computer to the Azure VM](#_lch6cvvewvwy)):

**ssh <username>@<public-ip>**

## 5.2 Install Mosquitto MQTT Broker

Run the following commands on your Azure VM to update the package list and upgrade the installed packages:

**sudo apt update && sudo apt upgrade**

Next, install the Mosquitto MQTT broker on your VM with the following command:

**sudo apt install -y mosquitto mosquitto-clients**

Start the Mosquitto service and enable it to start on boot:

**sudo systemctl enable mosquitto.service**

## 5.3 Test the mosquitto installation

Test to make sure that the mosquitto broker is running on your Azure VM. You can do that by using two SSH instances: one instance **subscribes** to a topic, the other instance **publishes** to the topic.

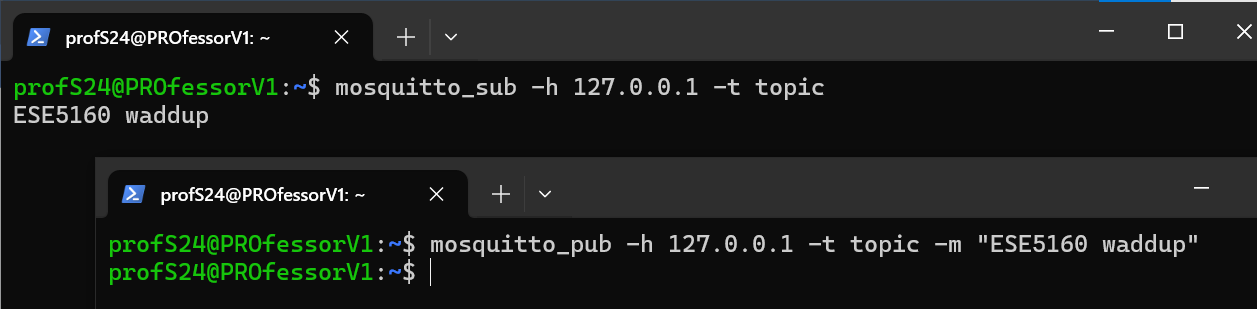
For the first instance, type the following:

**mosquitto\_sub -h 127.0.0.1 -t topic**

For the second instance, type the following:

**mosquitto\_pub -h 127.0.0.1 -t topic -m “ESE5160 testing!”**

Now, see if the **subscribed** instance receives the message. If so, the mosquitto broker is running properly on your VM’s localhost.



If you don’t see this, try running:

**mosquitto -v**

Check for any errors, such as **Address already in use**, a port issue that occurs if you have two instances of mosquitto running at the same time ([solution](https://stackoverflow.com/questions/30207649/address-already-in-use-error-in-mosquitto): kill the existing processes).

## 5.4 Enable Remote Access & Firewall Config

For simplicity, we’ll be enabling remote access with no authentication for your mosquitto MQTT broker. First, let’s open the configuration file in the editor nano ([nano shortcuts here](https://www.nano-editor.org/dist/latest/cheatsheet.html)):

**sudo nano /etc/mosquitto/mosquitto.conf**

Next, modify the configuration file so it **only contains the text below:**

**# Place your local configuration in /etc/mosquitto/conf.d/**

**#**

**# A full description of the configuration file is at**

**# /usr/share/doc/mosquitto/examples/mosquitto.conf.example**

**pid\_file /run/mosquitto/mosquitto.pid**

**persistence true**

**persistence\_location /var/lib/mosquitto/**

**log\_dest file /var/log/mosquitto/mosquitto.log**

**include\_dir /etc/mosquitto/conf.d**

**listener 1883**

**allow\_anonymous true**

Finally, exit and save this file by: **Ctrl+X**, then **Y** and **Enter**.

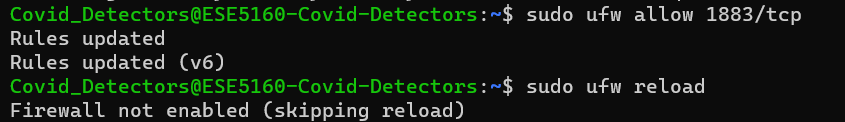
Restart mosquitto with:

**sudo systemctl restart mosquitto**

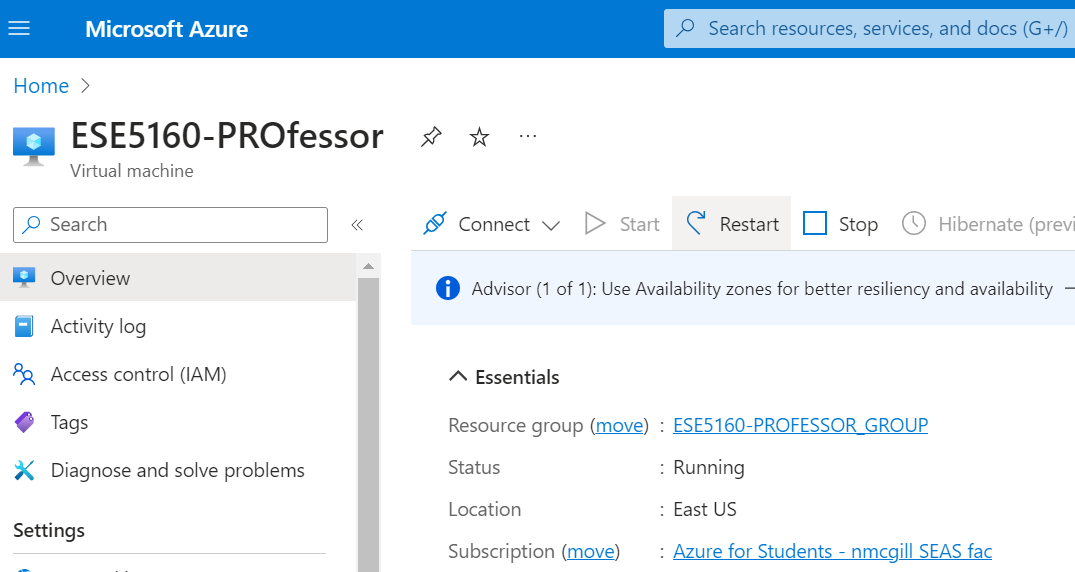
Open the MQTT port (default is 1883) in the VM's firewall so that clients can connect to the MQTT broker:

**sudo ufw allow 1883/tcp**

**sudo ufw reload**



## 5.5 Restart the Azure VM

Go to the Azure dashboard and click **Restart** to reboot your virtual machine. The configuration of your moquitto broker will take effect and it should boot automatically. The Node-RED instance and Apache web server should all start by default on VM boot at this point, too.

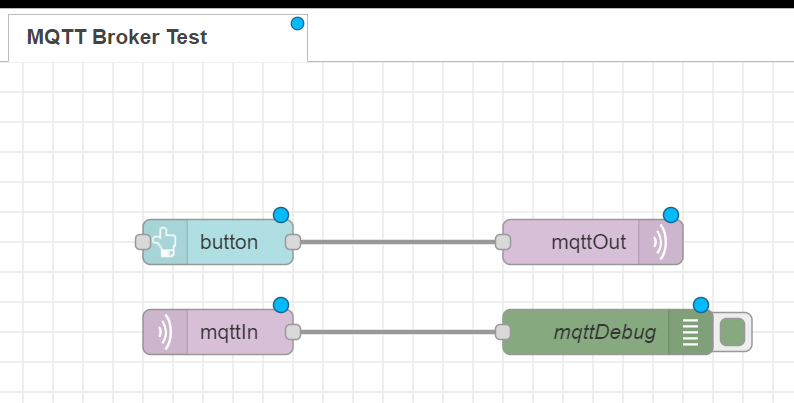
## 5.6 Test the MQTT Broker

Enter the VM public IP address in your web browser with **port 1880**.

Example: http://12.345.678.901:1880/

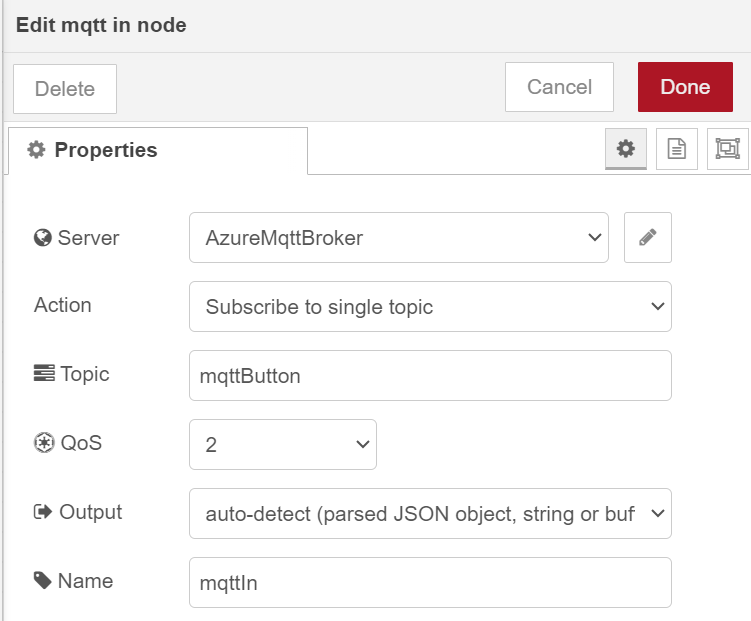
In a new flow, search for and add the following elements:

* Button
* MQTT Out
* MQTT In
* Debug printing

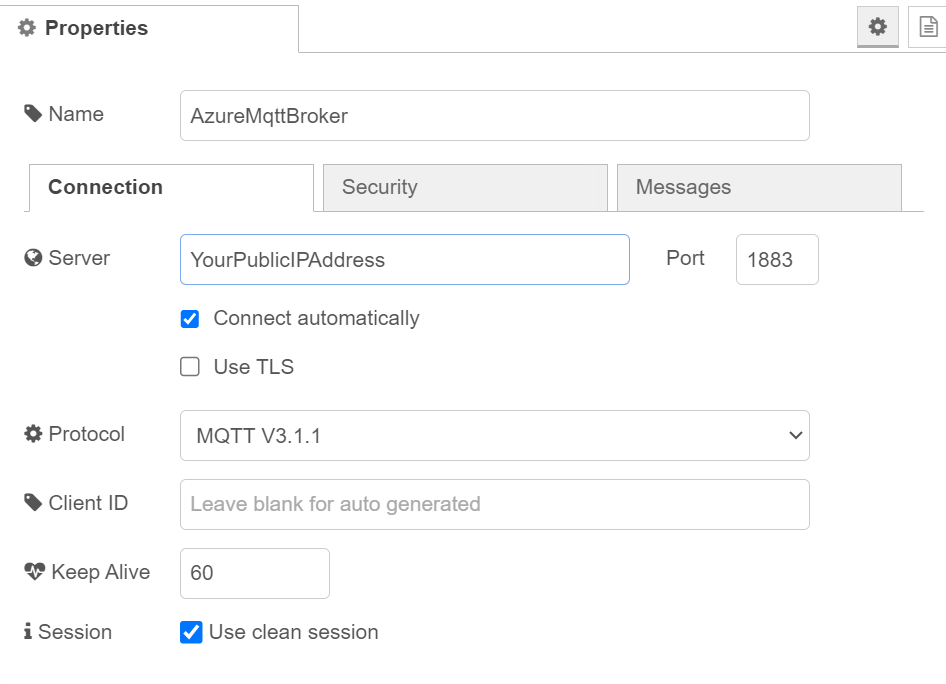


### Node Settings

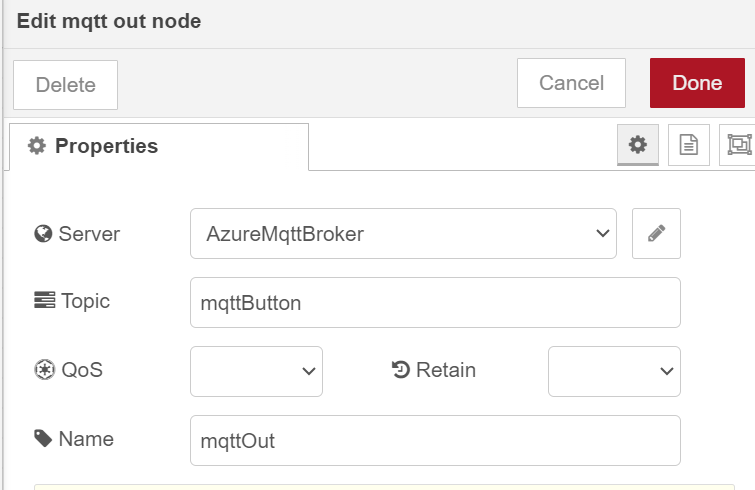
For the **mqttIn** node, set QoS and Topic.



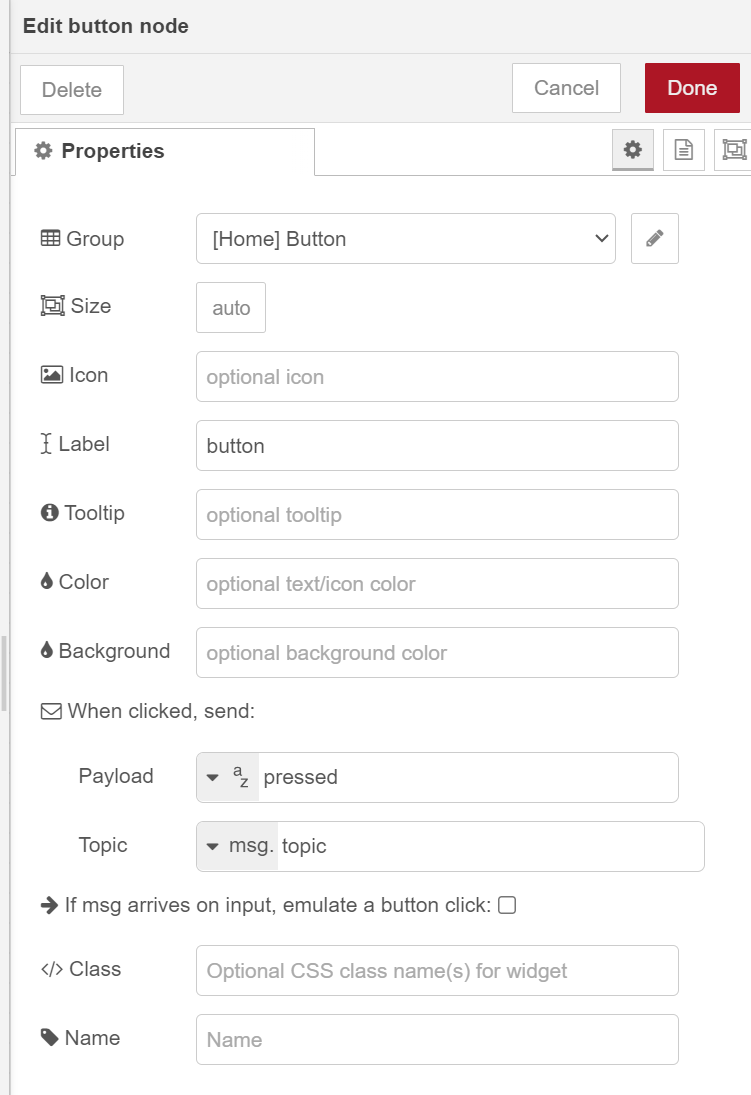
Next, edit the server. Click on pencil button on the right & type your Azure VM public IP as server name:



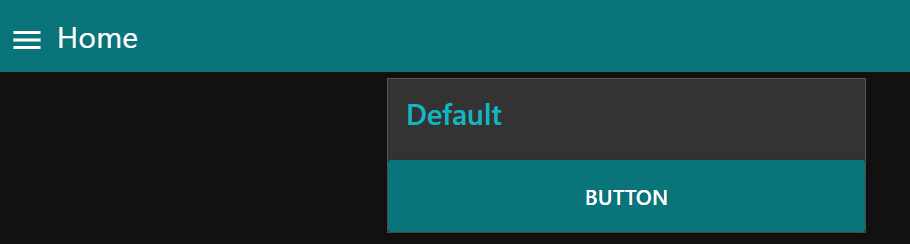
Repeat for the **mqttOut** node.



For the **button**, select the dashboard group you want to add the button element to. Set payload to something. In this case, it sends out **pressed**, a text string:



Now, deploy and launch the dashboard and go to **YourPublicIP:1880/ui**



On clicking the button we can see the messages in the debug window of Node-RED flow

