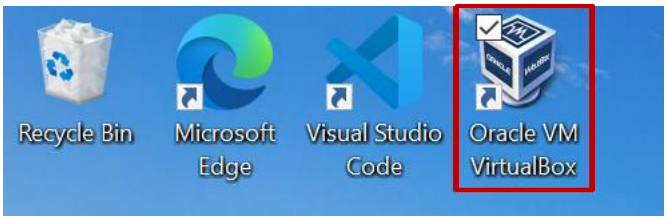
WEEK10 HW2

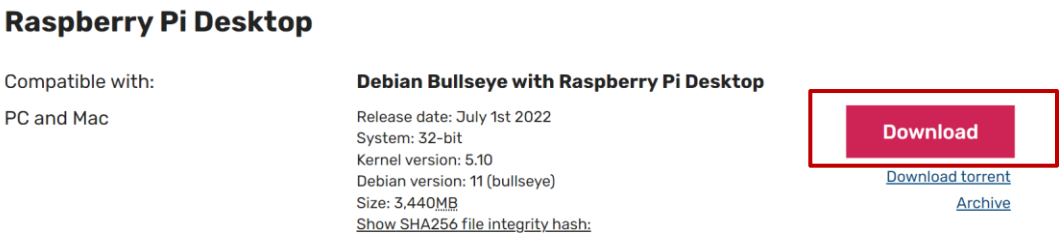
Part1 - Raspberry Pi emulator + VirtualBox + Sense HAT Emulator

Step1. Install VirtualBox on Windows10 by this link.

After installation is done, a shortcut will show up on your desktop.



Step2. Download Raspberry Pi Desktop by this link.

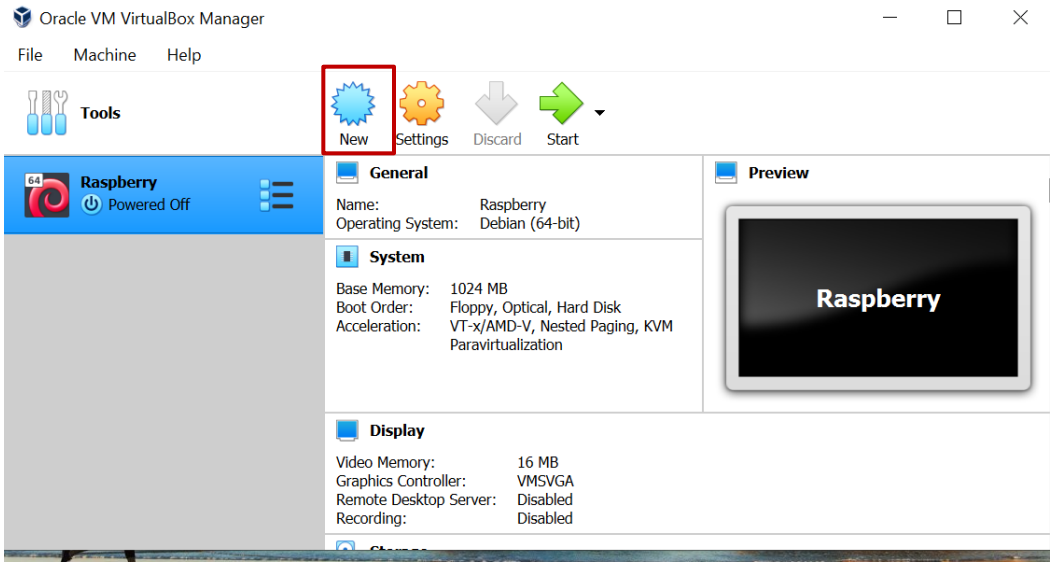


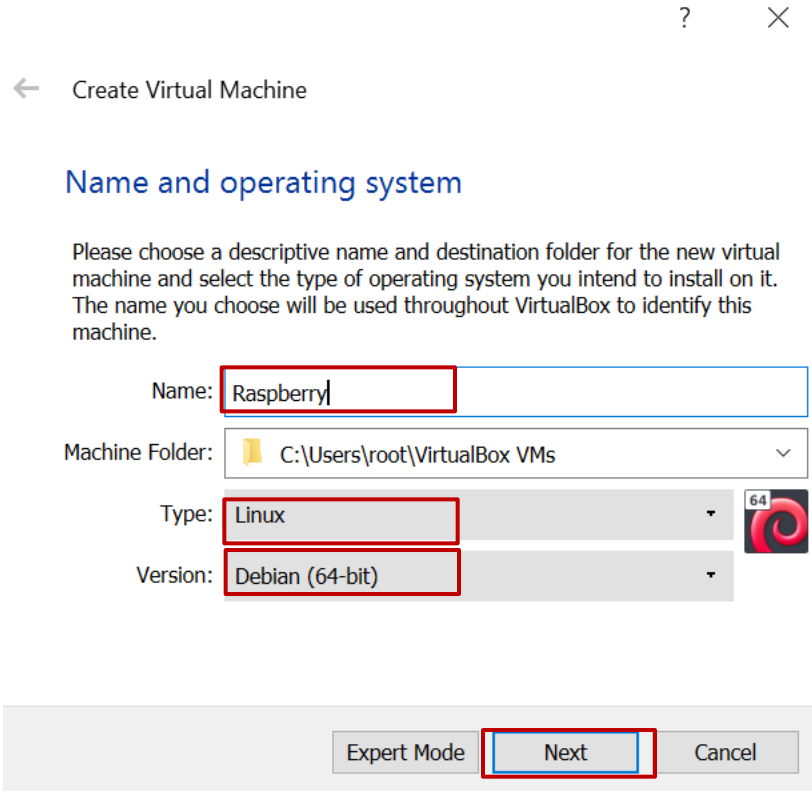
Step3. Enable Intel's VT-x or AMD's AMD-V visualization.

If your notebook model is Lenovo T450S, you may refer to this link to enable this setting.

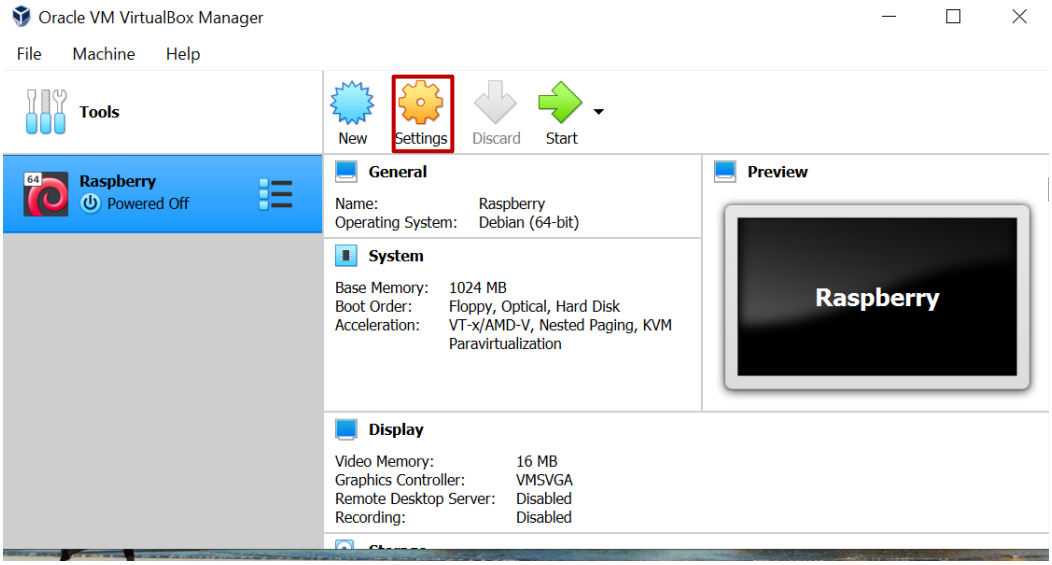
Step4. Create a virtual machine for Raspberry Pi Desktop.

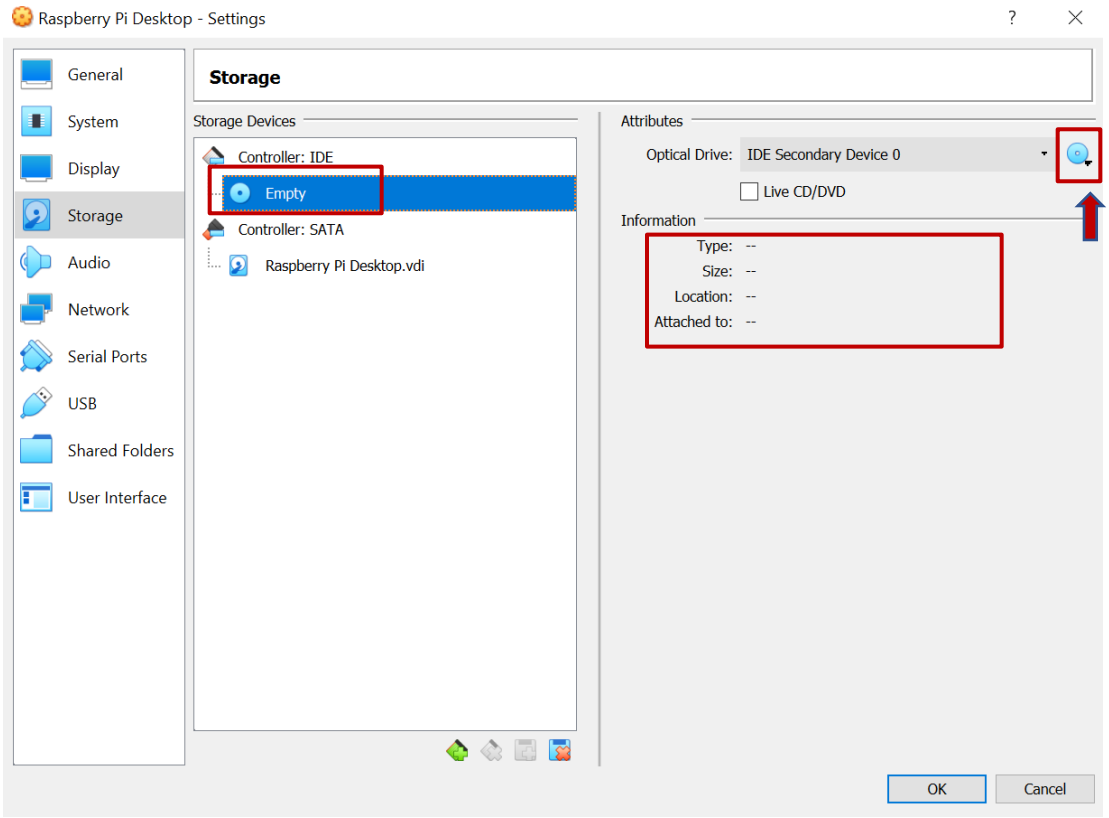
Open VirtualBox, follow the instructions to create a new virtual machine.


Oracle VM VirtualBox Manager
File
Machine
Help
Tools
New
Settings
Discard
Start
Raspberry
Powered Off
General
Name: Raspberry
Operating System: Debian (64-bit)
System
Base Memory: 1024 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM
Paravirtualization
Preview
Raspberry
Display
Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

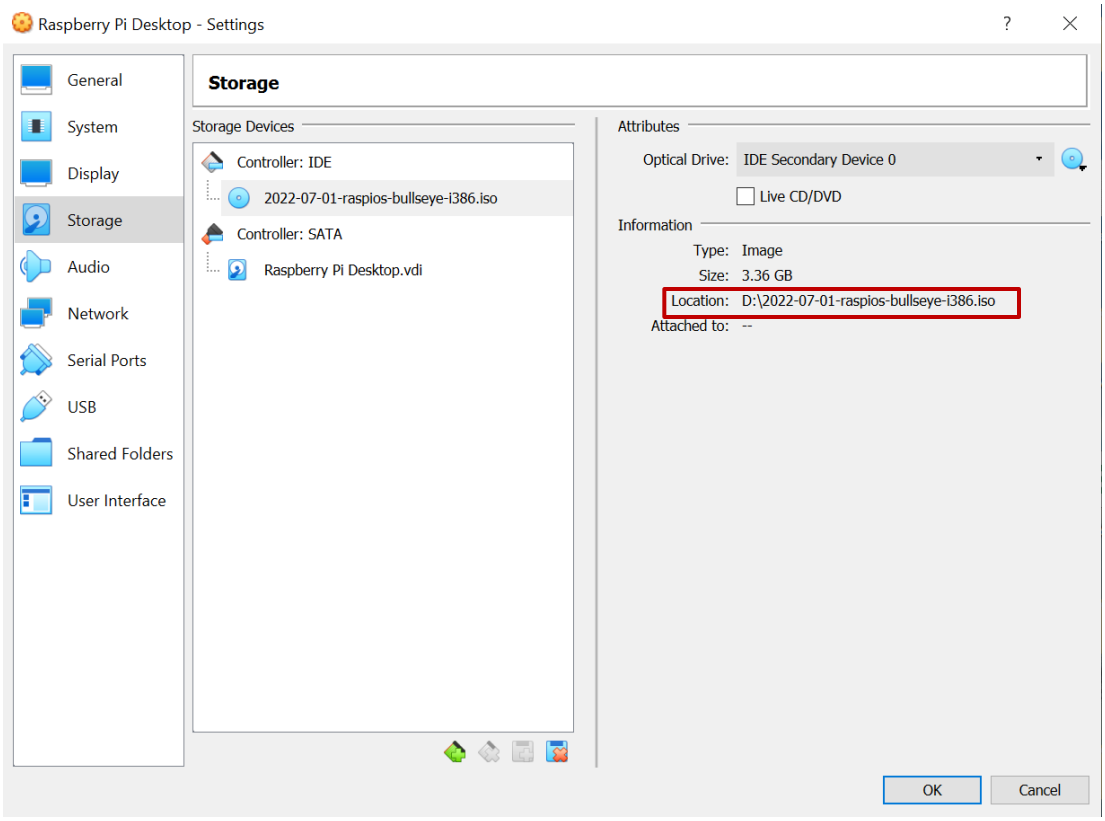

Create Virtual Machine
Name and operating system
Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.
Name: Raspberry
Machine Folder: C:\Users\root\VirtualBox VMs
Type: Linux
Version: Debian (64-bit)
Expert Mode
Next
Cancel

Click the setting → Storage → Controller IDE: Empty

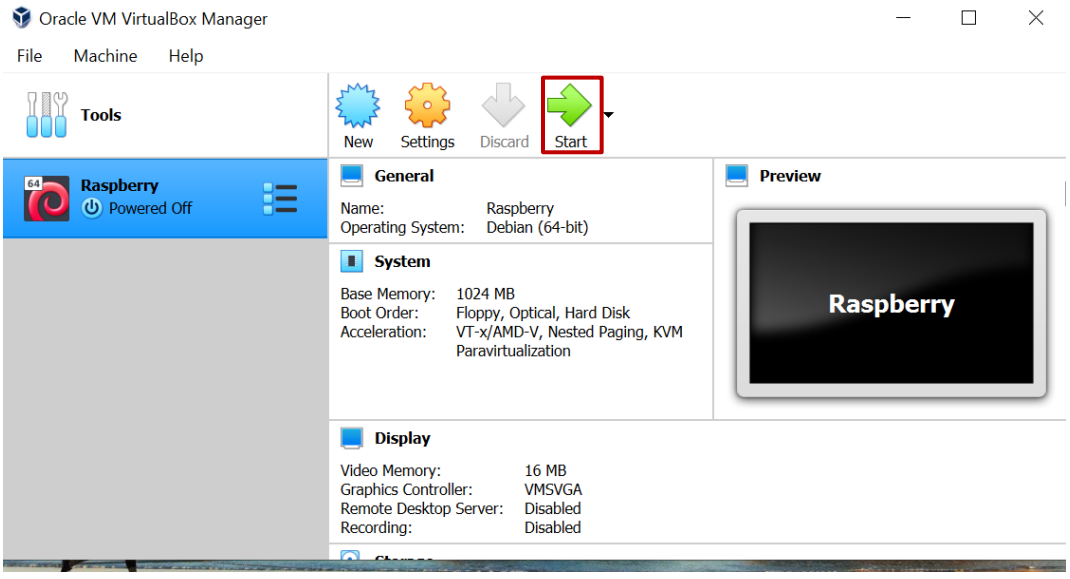


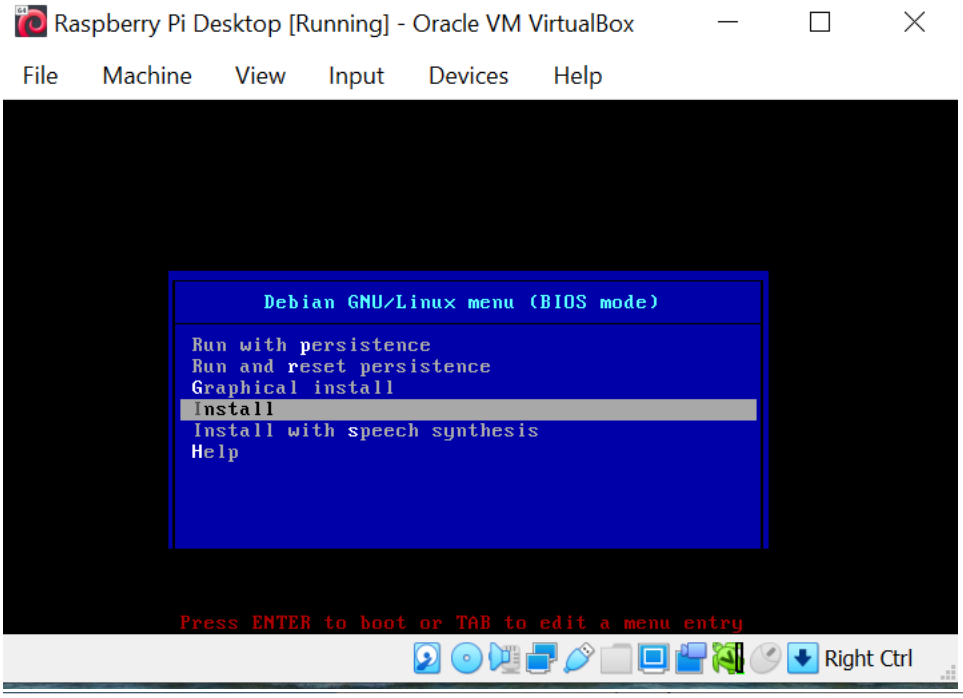


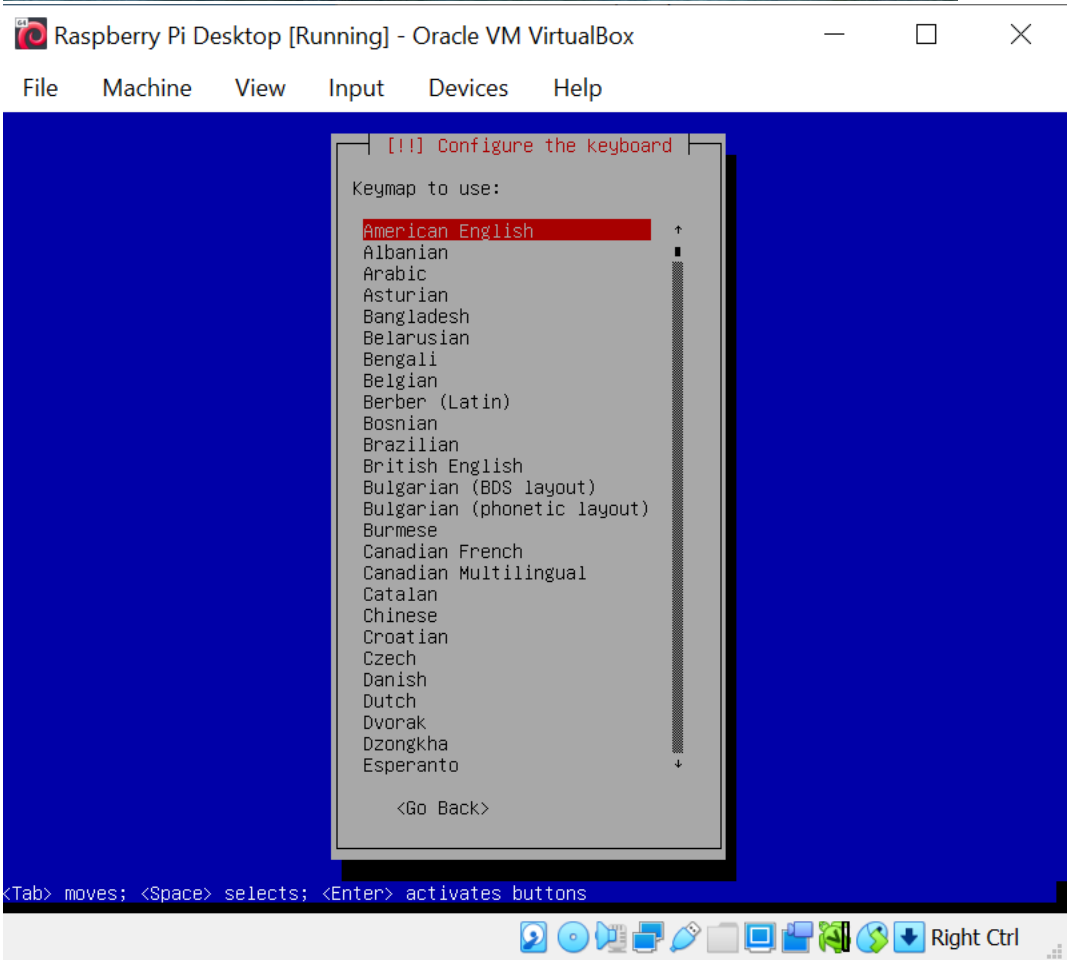
Please choose the *.iso file you downloaded in step2 as the drive.
Then you will see the location information in the below snapshot:

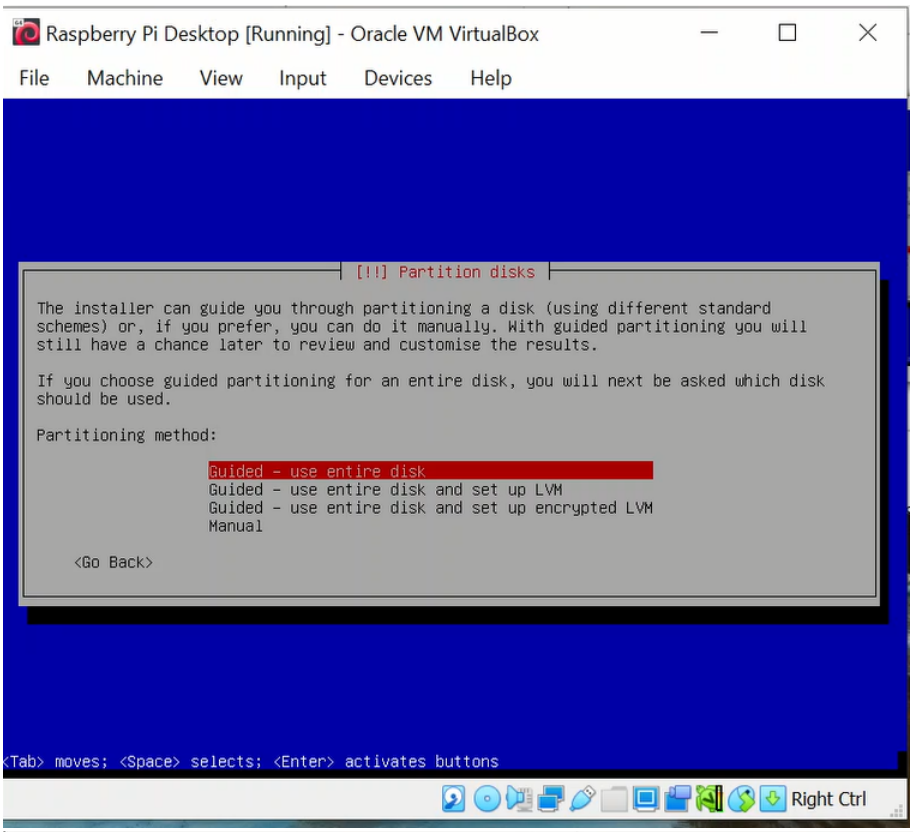


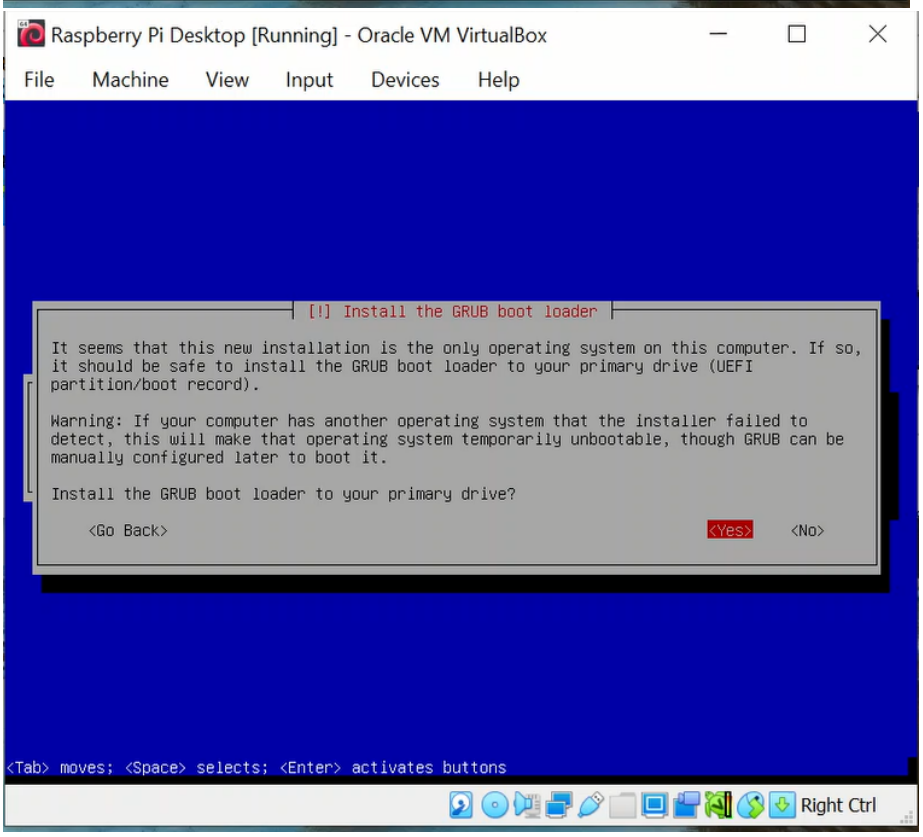
Click Start to install raspberry pi desktop to VirtualBox.

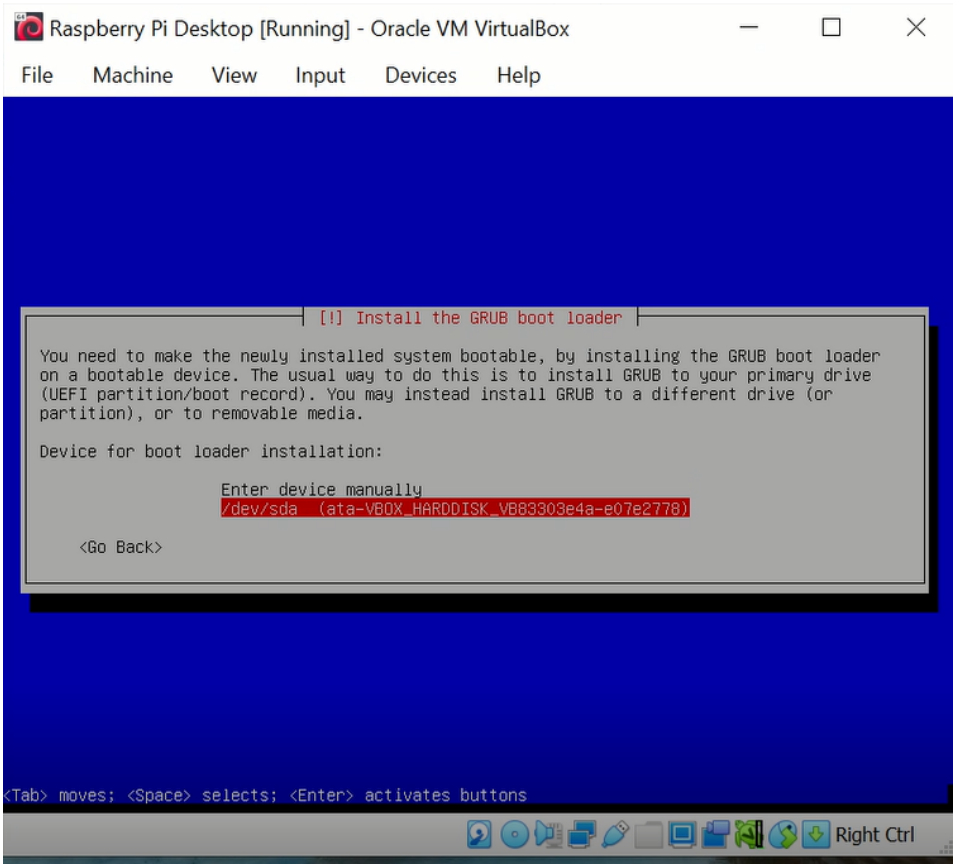


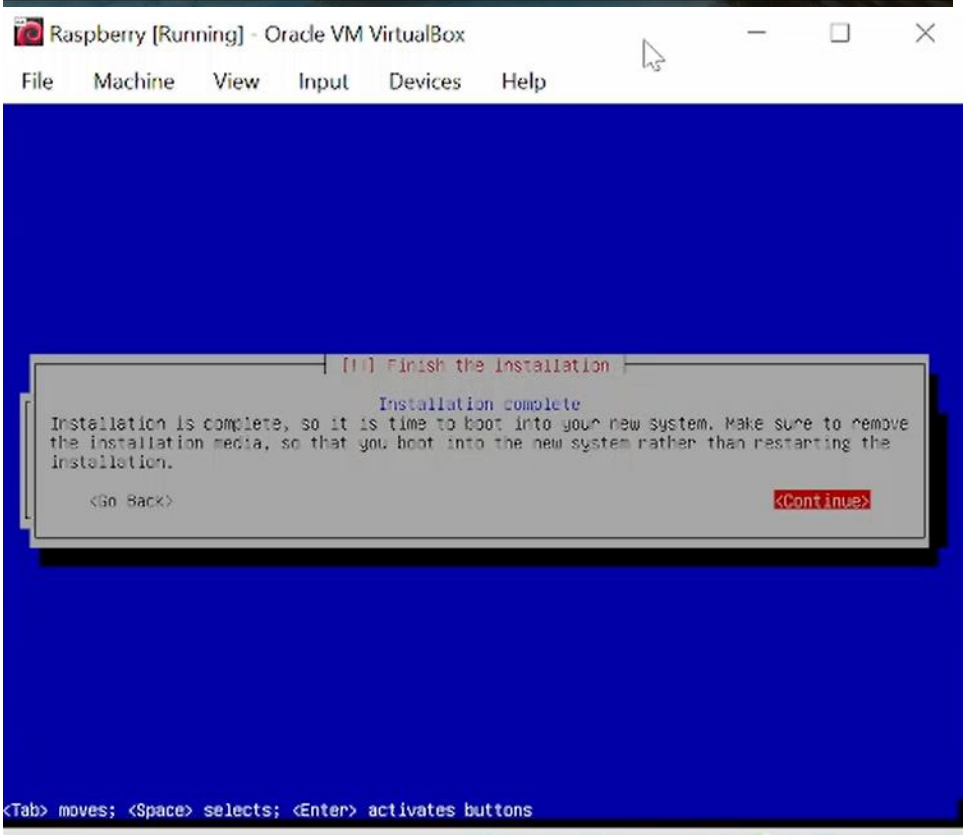

Raspberry Pi Desktop [Running] - Oracle VM VirtualBox
File
Machine
View
Input
Devices
Help
Debian GNU/Linux menu (BIOS mode)
Run with persistence
Run and reset persistence
Graphical install
Install
Install with speech synthesis
Help
Press ENTER to boot or TAB to edit a menu entry
Right Ctrl


Raspberry Pi Desktop [Running] - Oracle VM VirtualBox
File
Machine
View
Input
Devices
Help
[!!] Configure the keyboard
Keymap to use:
American English
Albanian
Arabic
Asturian
Bangladesh
Belarusian
Bengali
Belgian
Berber (Latin)
Bosnian
Brazilian
British English
Bulgarian (BDS layout)
Bulgarian (phonetic layout)
Burmese
Canadian French
Canadian Multilingual
Catalan
Chinese
Croatian
Czech
Danish
Dutch
Dvorak
Dzongkha
Esperanto
<Go Back>
<Tab> moves; <Space> selects; <Enter> activates buttons
Right Ctrl


Raspberry Pi Desktop [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
[!!] Partition disks
The installer can guide you through partitioning a disk (using different standard schemes) or, if you prefer, you can do it manually. With guided partitioning you will still have a chance later to review and customise the results.
If you choose guided partitioning for an entire disk, you will next be asked which disk should be used.
Partitioning method:
Guided - use entire disk
Guided - use entire disk and set up LVM
Guided - use entire disk and set up encrypted LVM
Manual
<Go Back>
<Tab> moves; <Space> selects; <Enter> activates buttons
Right Ctrl

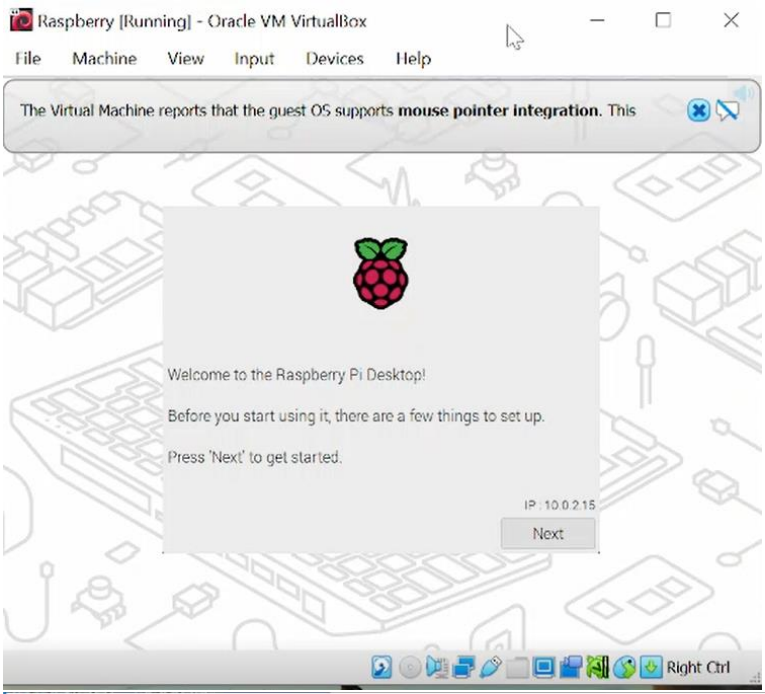

Raspberry Pi Desktop [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
[!] Install the GRUB boot loader
It seems that this new installation is the only operating system on this computer. If so, it should be safe to install the GRUB boot loader to your primary drive (UEFI partition/boot record).
Warning: If your computer has another operating system that the installer failed to detect, this will make that operating system temporarily unbootable, though GRUB can be manually configured later to boot it.
Install the GRUB boot loader to your primary drive?
<Go Back>
<Yes>
<No>
<Tab> moves; <Space> selects; <Enter> activates buttons
Right Ctrl

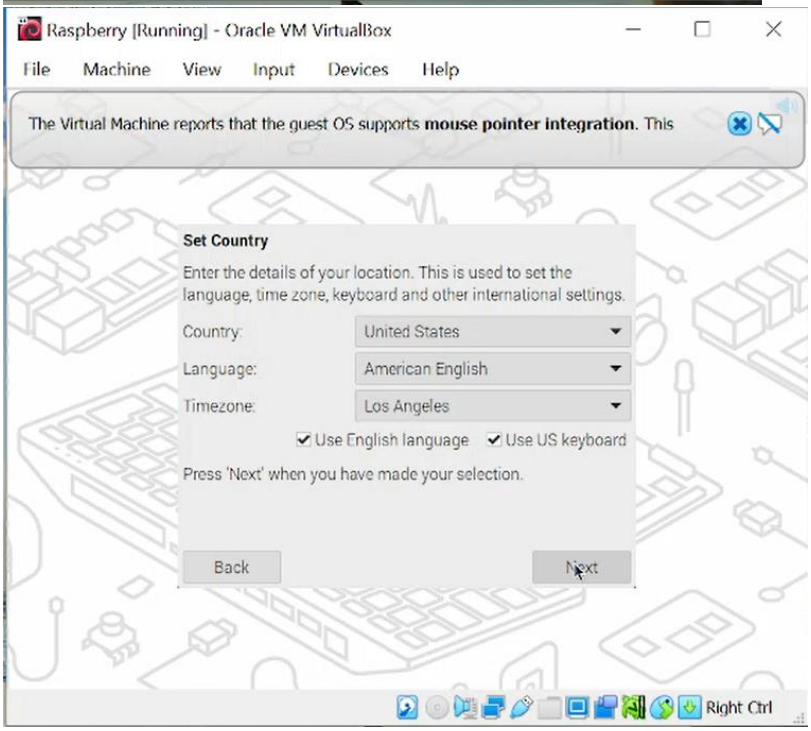

Raspberry Pi Desktop [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
[!] Install the GRUB boot loader
You need to make the newly installed system bootable, by installing the GRUB boot loader on a bootable device. The usual way to do this is to install GRUB to your primary drive (UEFI partition/boot record). You may instead install GRUB to a different drive (or partition), or to removable media.
Device for boot loader installation:
Enter device manually
/dev/sda (ata-VBOX_HARDDISK_VB83303e4a-e07e2778)
<Go Back>
<Tab> moves; <Space> selects; <Enter> activates buttons
Right Ctrl

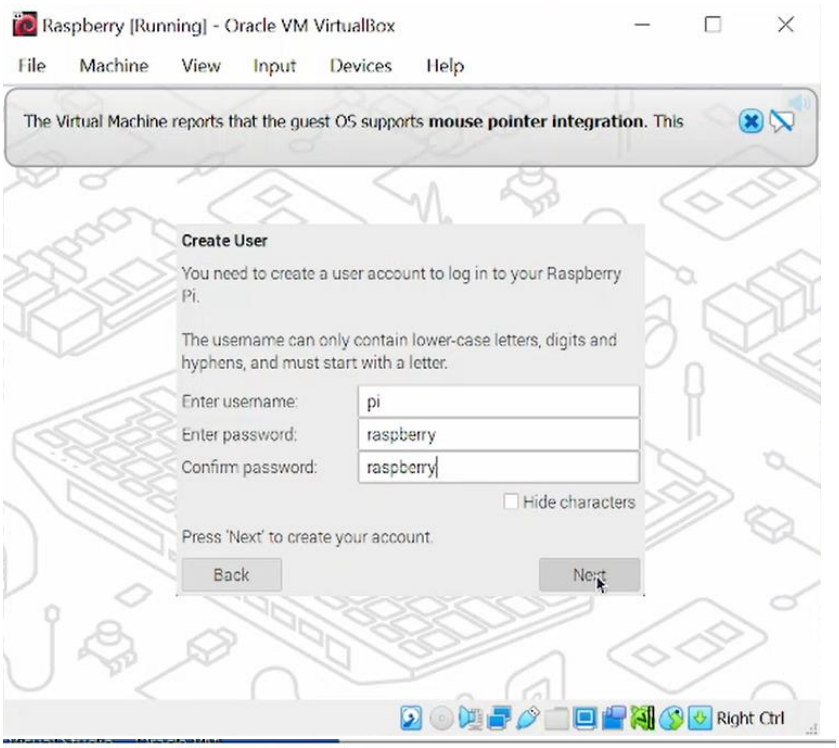

Raspberry [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
[!!] Finish the installation
Installation complete
Installation is complete, so it is time to boot into your new system. Make sure to remove the installation media, so that you boot into the new system rather than restarting the installation.
<Go Back>
<Continue>
<Tab> moves; <Space> selects; <Enter> activates buttons

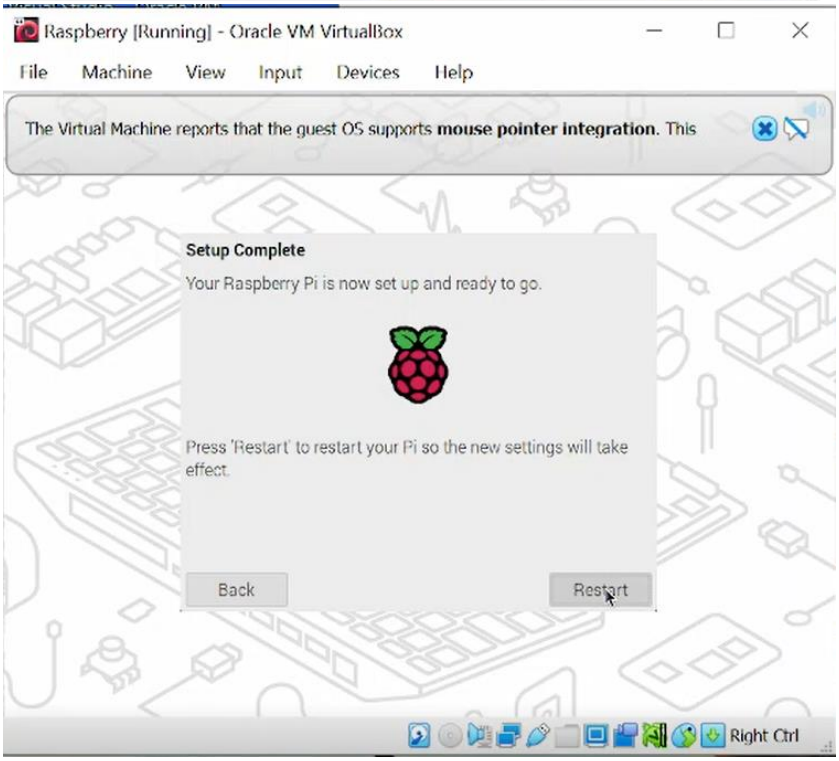
Now, you have installed the raspberry pi desktop sunccessfully.

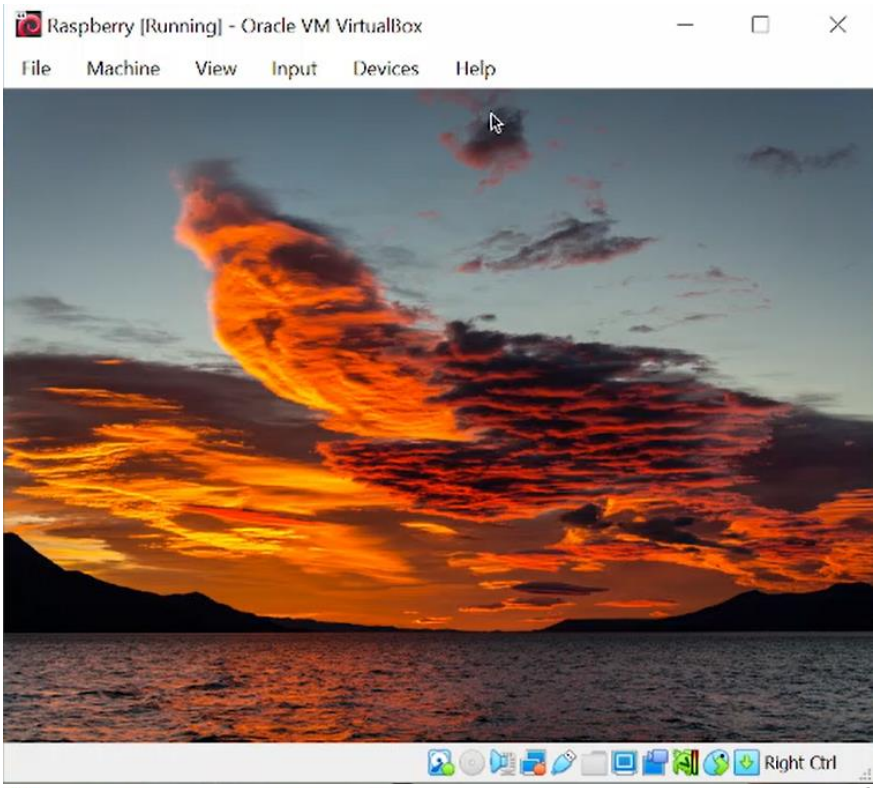
Step5. Configure Raspberry Pi Desktop.

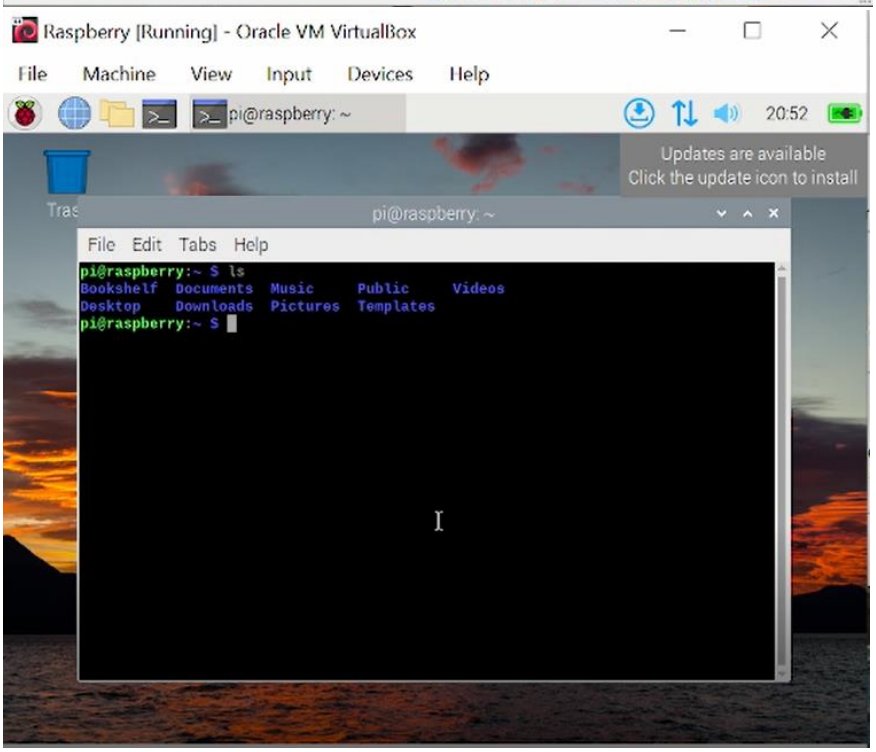




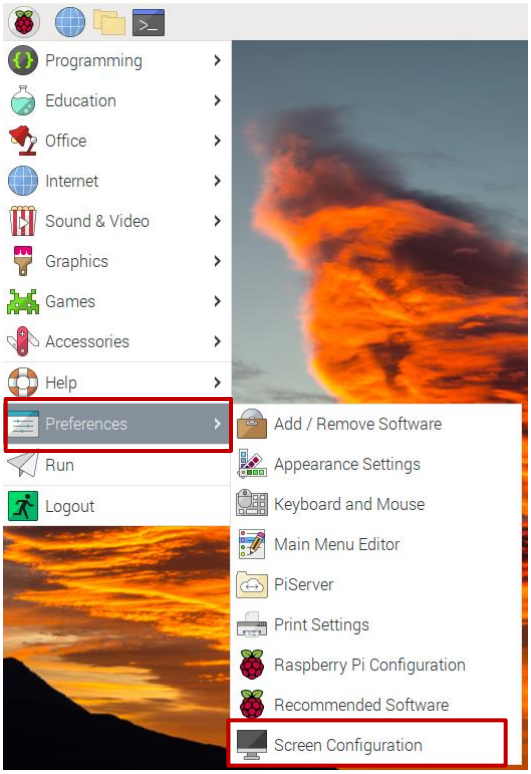

Raspberry [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
The Virtual Machine reports that the guest OS supports mouse pointer integration. This
Create User
You need to create a user account to log in to your Raspberry Pi.
The username can only contain lower-case letters, digits and hyphens, and must start with a letter.
Enter username: pi
Enter password: raspberry
Confirm password: raspberry
Hide characters
Press 'Next' to create your account.
Back
Next
Right Ctrl

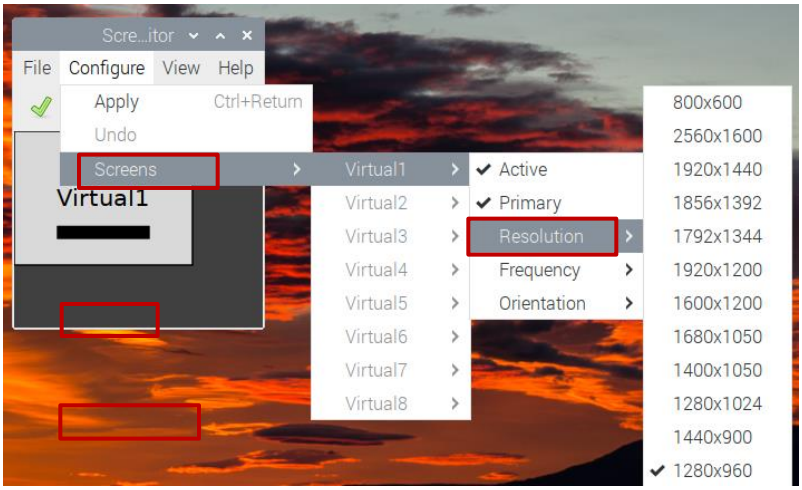

Raspberry [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
The Virtual Machine reports that the guest OS supports mouse pointer integration. This
Setup Complete
Your Raspberry Pi is now set up and ready to go.
Press 'Restart' to restart your Pi so the new settings will take effect.
Back
Restart
Right Ctrl

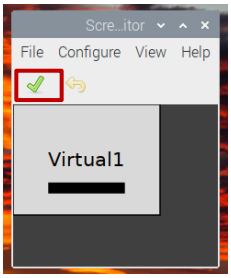

Raspberry [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl


Raspberry [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
pi@raspberry: ~
20:52
Updates are available
Click the update icon to install
pi@raspberry: ~
File Edit Tabs Help
pi@raspberry:~ $ ls
Bookshelf Documents Music Public Videos
Desktop Downloads Pictures Templates
pi@raspberry:~ $

Step6. Configure Resolution for Raspberry Pi Desktop.

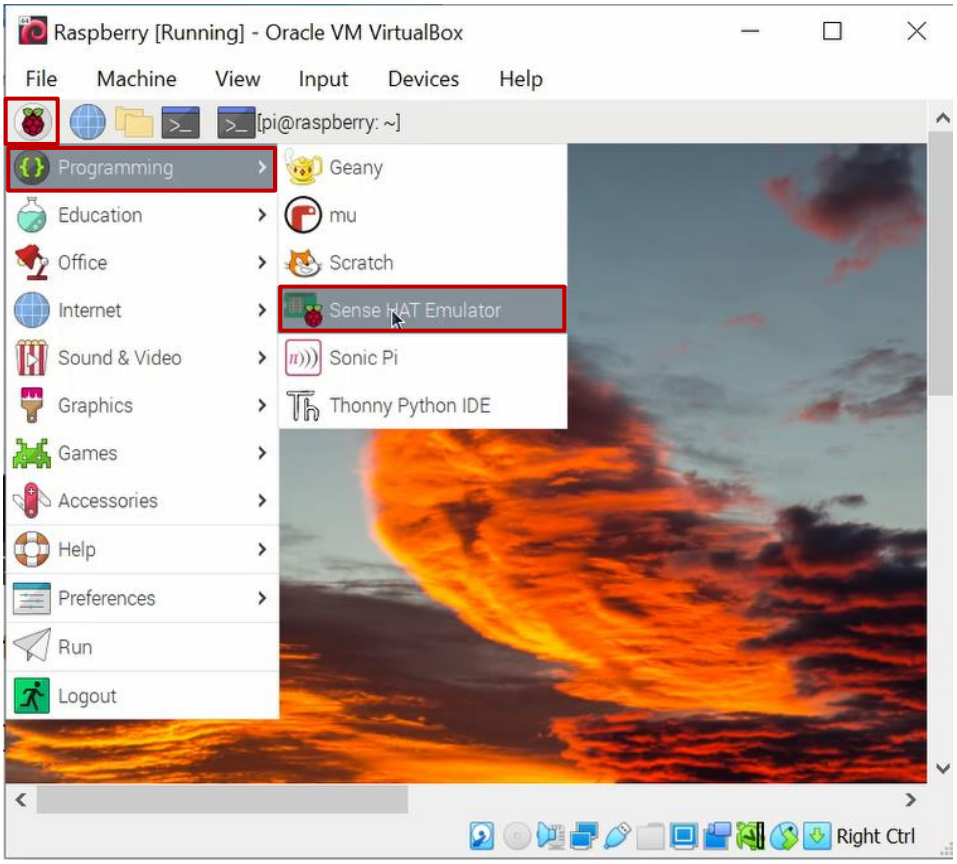






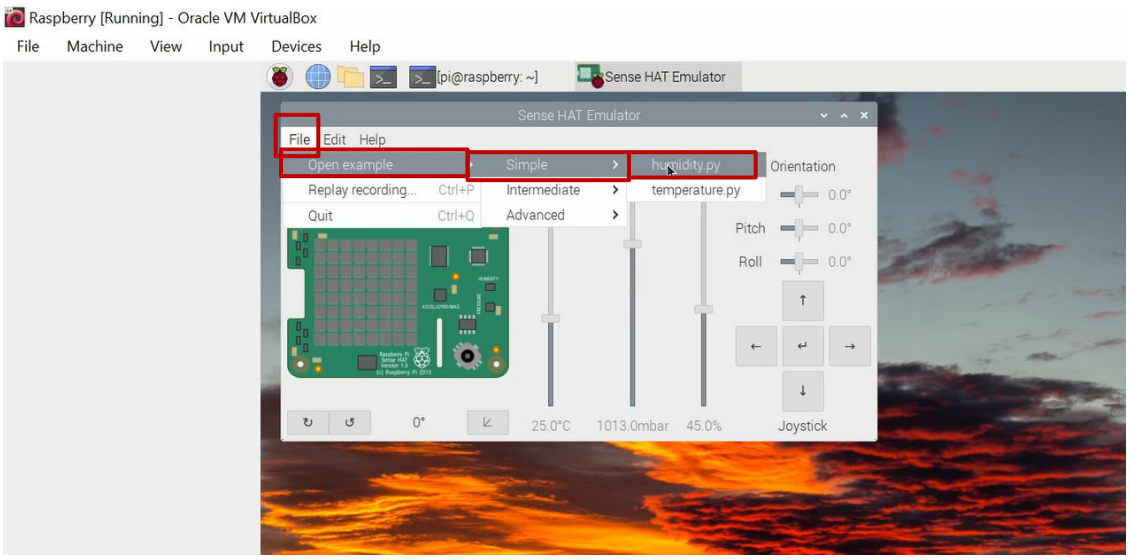
Step7. Run example code to test Sense Hat Emulator.

Start up Sense Hat Emulator

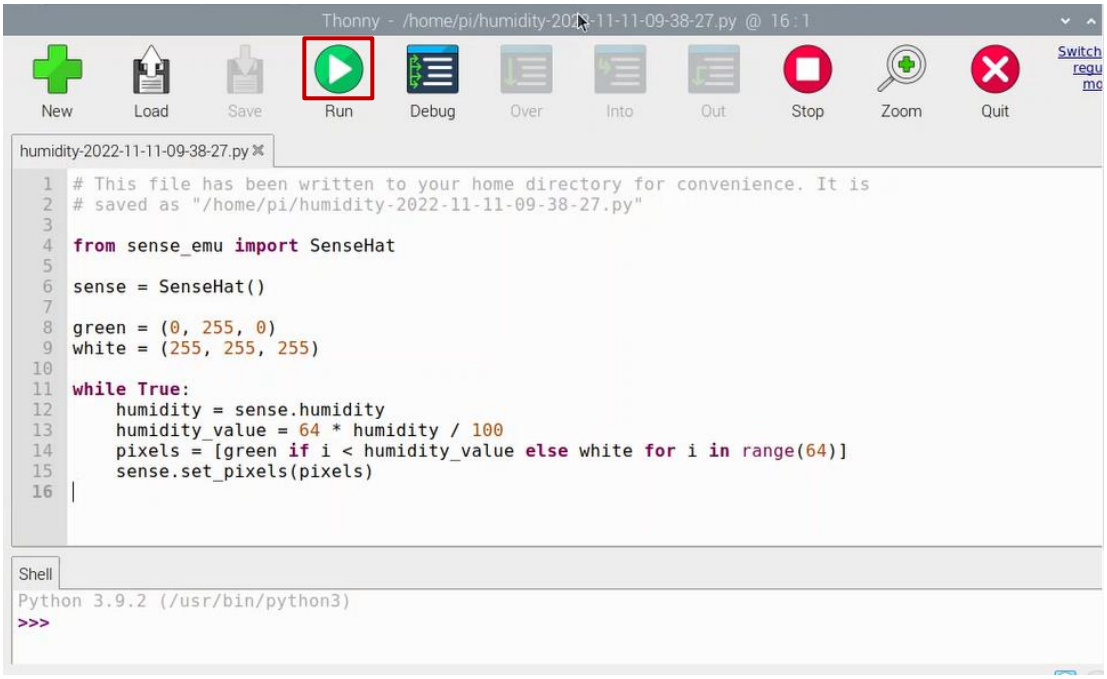


Open the example code, e.g., humidty.py

This program adjusts the number of green and white pixels displayed on the LED, depending on the detected humidity.

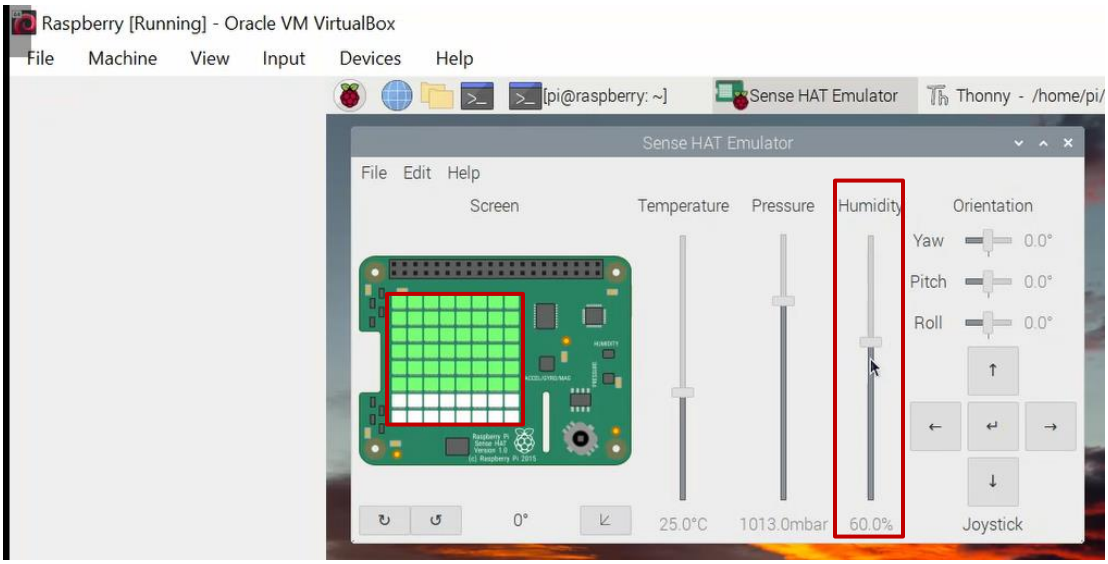


Click the button “Run” to run the program.

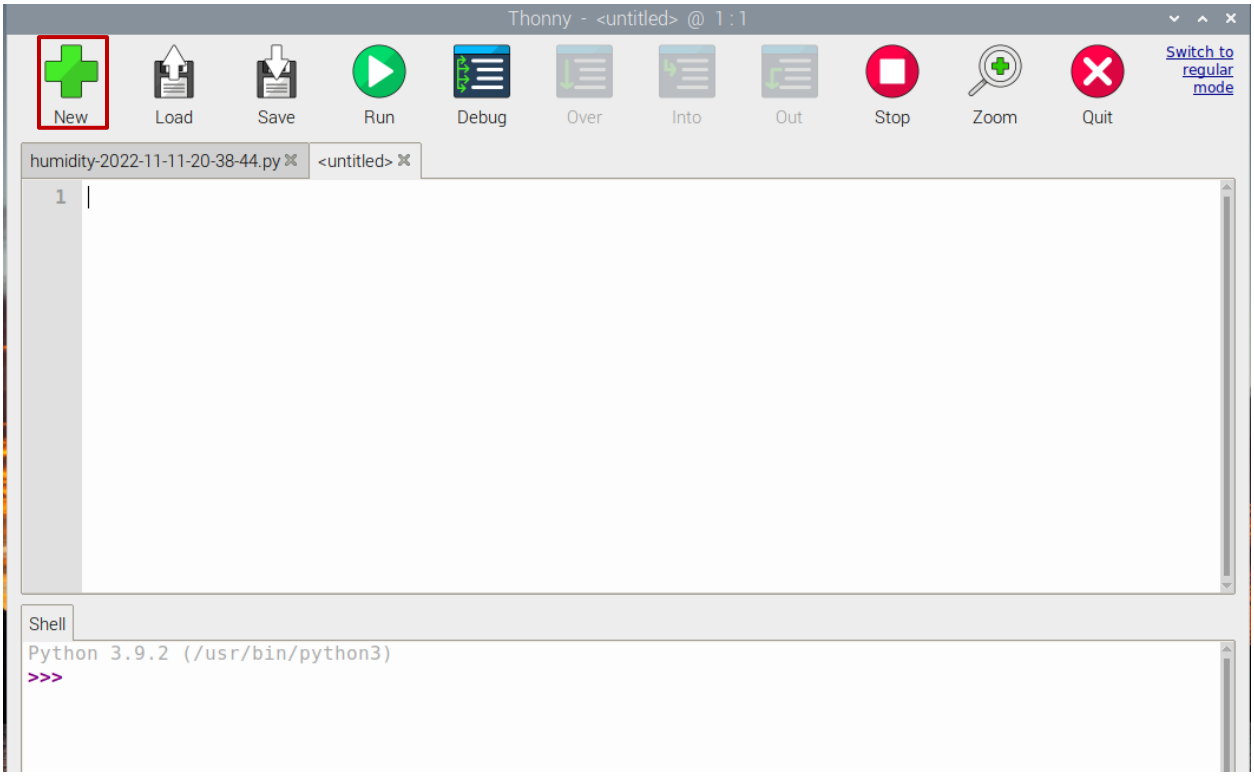


Display the result.

We can see that the numbers of green LEDs will change based on the humidity.

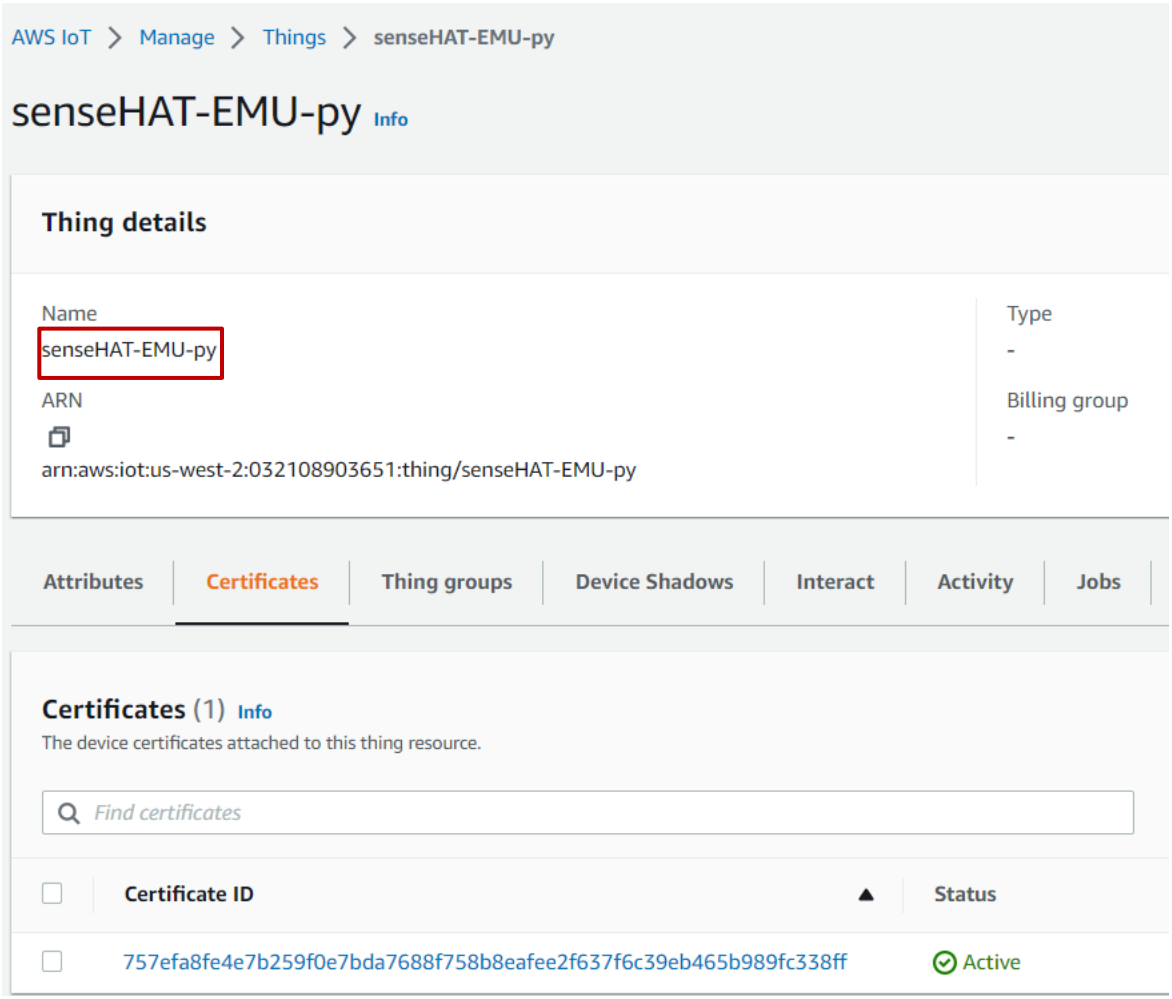


We may also create our own program.



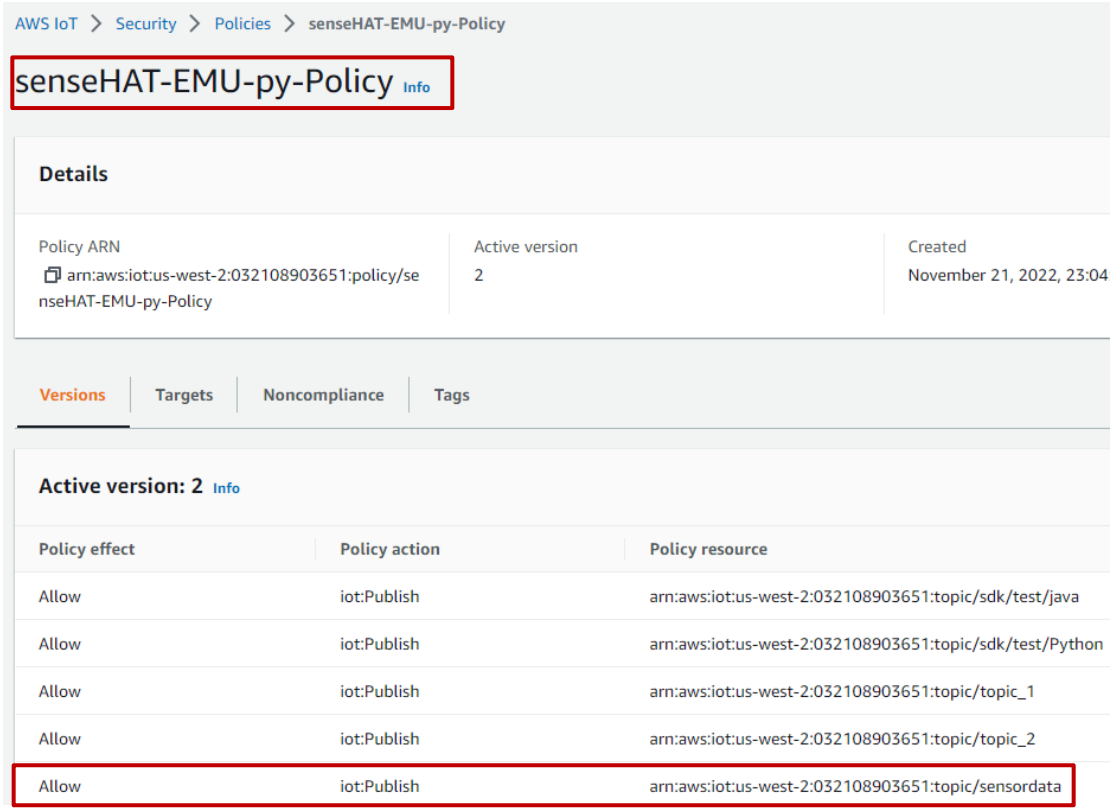
Part2 – Connecting Raspberry Pi emulator + VirtualBox + Sense HAT Emulator to AWS IOT Using Python

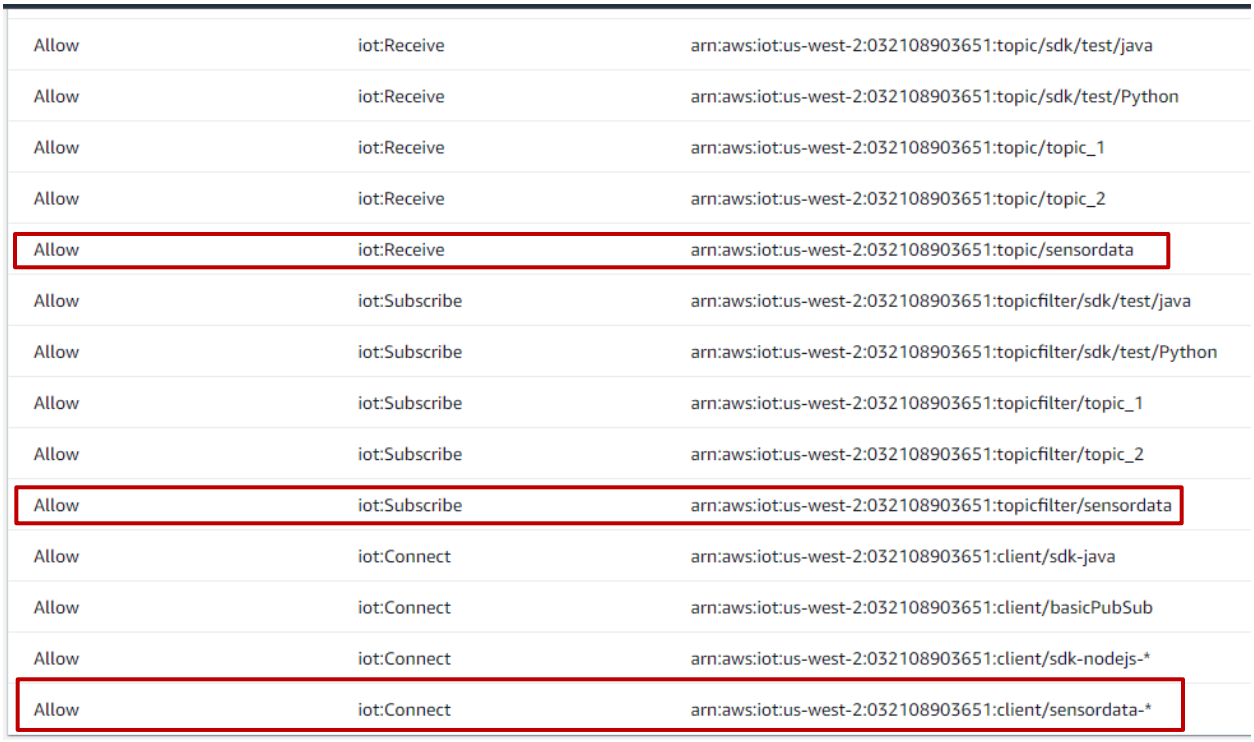
Step1. Walk through AWS IOT → Connect → Connect one device



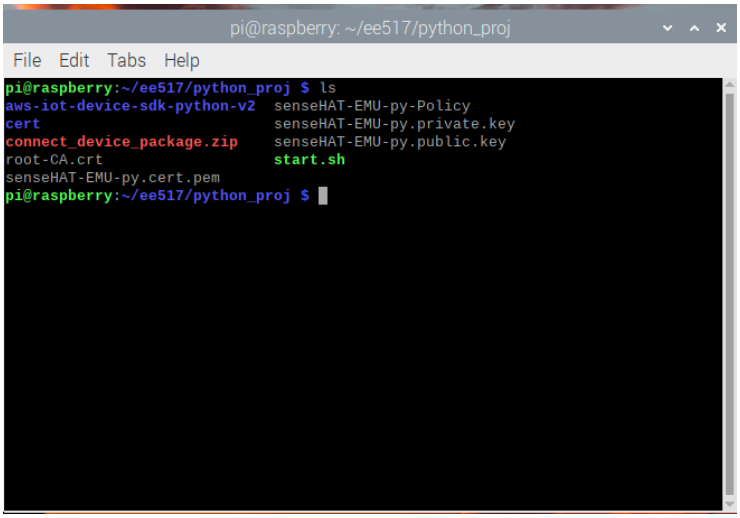
Edit Policy as you need:

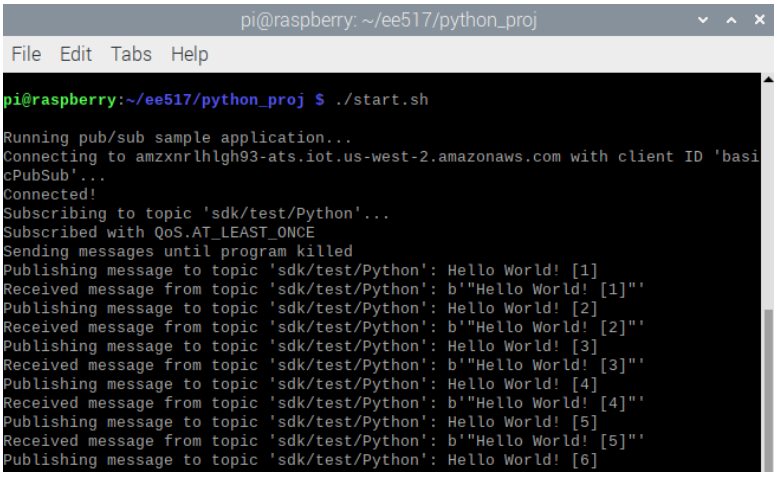
You can add the topic you want to subscribe to, receive or publish as well as the clients you want to connect to.

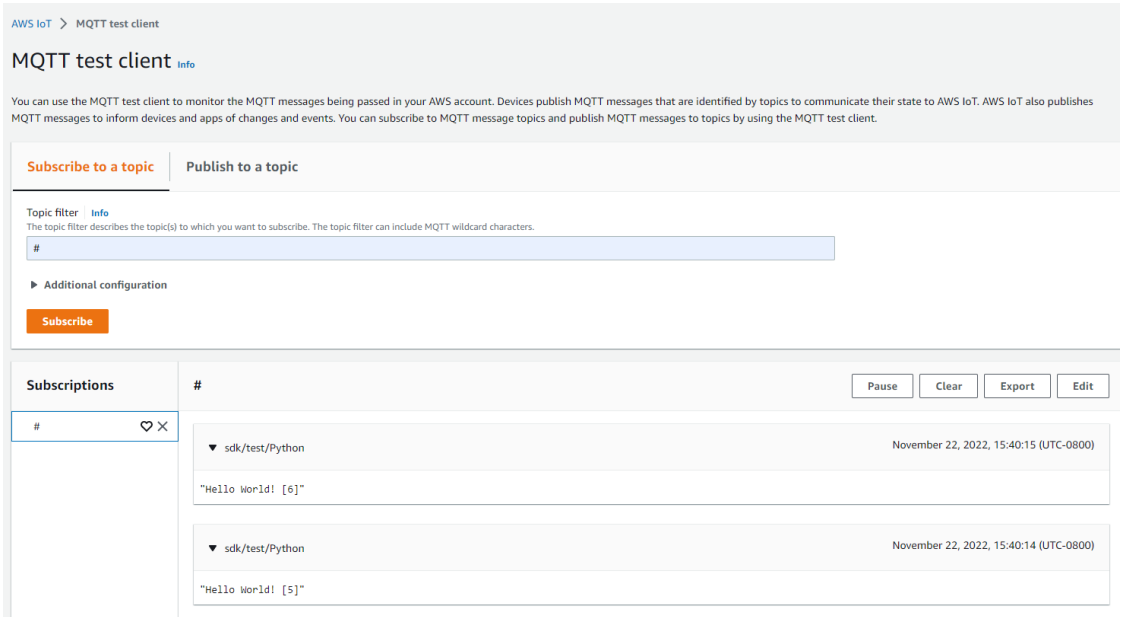




Download connection kit, unzip it, then run start.sh.







Step2. Add codes in sampe/pubsub.py to support Sense HAT Emulator

```
from awscrt import mqtt
import sys
import threading
import time
from uuid import uuid4
import json
from sense_emu import SenseHat
sense = SenseHat()
```

```
# Callback when the subscribed topic receives a message
def on_message_received(topic, payload, dup, qos, retain, **kwargs):
    print("{}".format(payload.decode().replace('"','')))
    #print("Received message from topic '{}': {}".format(topic, payload.decode().replace('"','')))
    global received_count
    received_count += 1
    if received_count == cmdUtils.get_command("count"):
        received_all_event.set()
```

```
# Publish message to server desired number of times.
# This step is skipped if message is blank.
# This step loops forever if count was set to 0.
if message_string:
    if message_count == 0:
        print ("Sending messages until program killed")
    else:
        print ("Sending {} message(s)".format(message_count))

    publish_count = 1
    while (publish_count <= message_count) or (message_count == 0):
        message = "{} {}".format('Temp: ', sense.temperature)
        #message = "{} [{}]".format(message_string, publish_count)
        #print("Publishing message to topic '{}': {}".format(message_topic, message))
        message_json = json.dumps(message)
        mqtt_connection.publish(
            topic=message_topic,
            payload=message_json,
            qos=mqtt.QoS.AT_LEAST_ONCE)
        time.sleep(1)
        publish_count += 1

# Wait for all messages to be received.
# This waits forever if count was set to 0.
if message_count != 0 and not received_all_event.is_set():
    print("Waiting for all messages to be received...")
```

Step3. Run the pubsub.py

```
pi@raspberry:~/ee517/python_proj $ vi aws-iot-device-sdk-python-v2/samples/pubsub_myrun.py
pi@raspberry:~/ee517/python_proj $ python3 aws-iot-device-sdk-python-v2/samples/pubsub_myrun.py --endpoint amzxnrlhlgh93-ats.iot.us-west-2.amazonaws.com --ca_
file root-CA.crt --cert senseHAT-EMU-py.cert.pem --key senseHAT-EMU-py.private.key --client_id sensordata-1 --topic sensordata --count 0
Connecting to amzxnrlhlgh93-ats.iot.us-west-2.amazonaws.com with client ID 'sensordata-1'...
Connected!
Subscribing to topic 'sensordata'...
Subscribed with QoS.AT_LEAST_ONCE
Sending messages until program killed
Temp:  24.984375
Temp:  24.984375
Temp:  24.984375
Temp:  24.984375
Temp:  24.984375
Temp:  24.96875
Temp:  24.9375
Temp:  24.953125
Temp:  24.953125
Temp:  24.984375
Temp:  25.0
Temp:  25.0
Temp:  25.0
Temp:  25.0
Temp:  24.015625
Temp:  21.4375
Temp:  19.1875
Temp:  16.609375
Temp:  14.984375
Temp:  14.984375
Temp:  14.96875
Temp:  19.484375
Temp:  30.46875
Temp:  43.40625
Temp:  56.34375
Temp:  64.125
Temp:  65.09375
```

DONE!!!

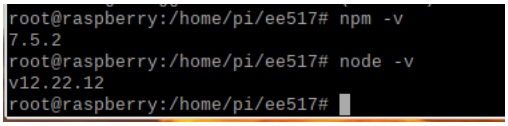
Part2 – Connecting Raspberry Pi emulator + VirtualBox + Sense HAT Emulator to AWS IOT Using Nodejs (NOT COMPLETE due to missing NodeJS package, please skip the following part)

Step1. Install NodeJS

```
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -

sudo apt-get install -y nodejs

node -v

npm -v
```

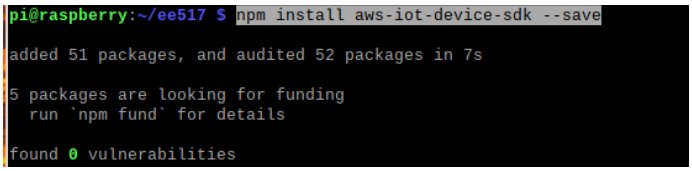
If npm command not found, run the following command to install npm directly:

```
sudo apt-get -f install npm
```



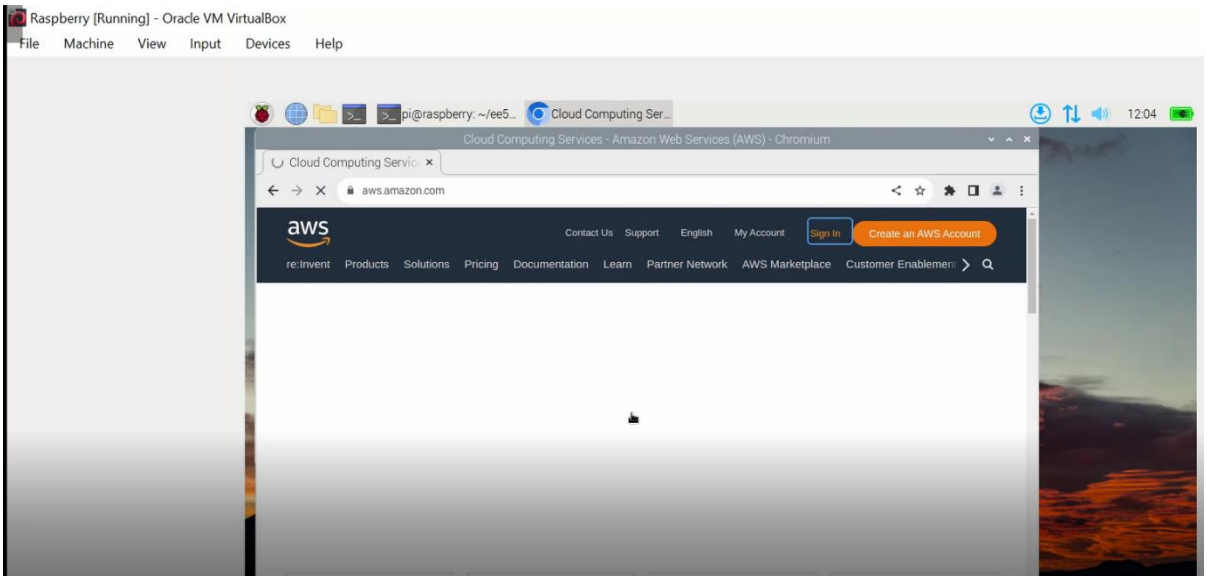
Step2. Install package

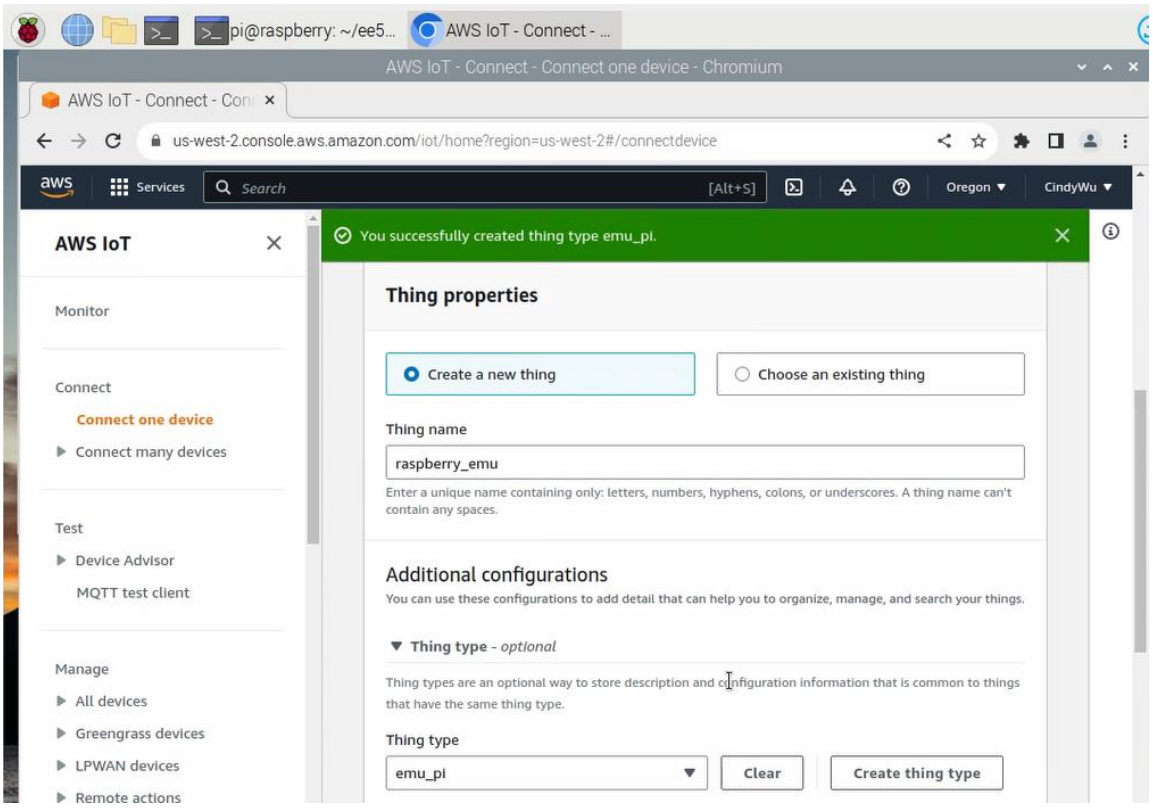
```
npm install aws-iot-device-jdk –save
```



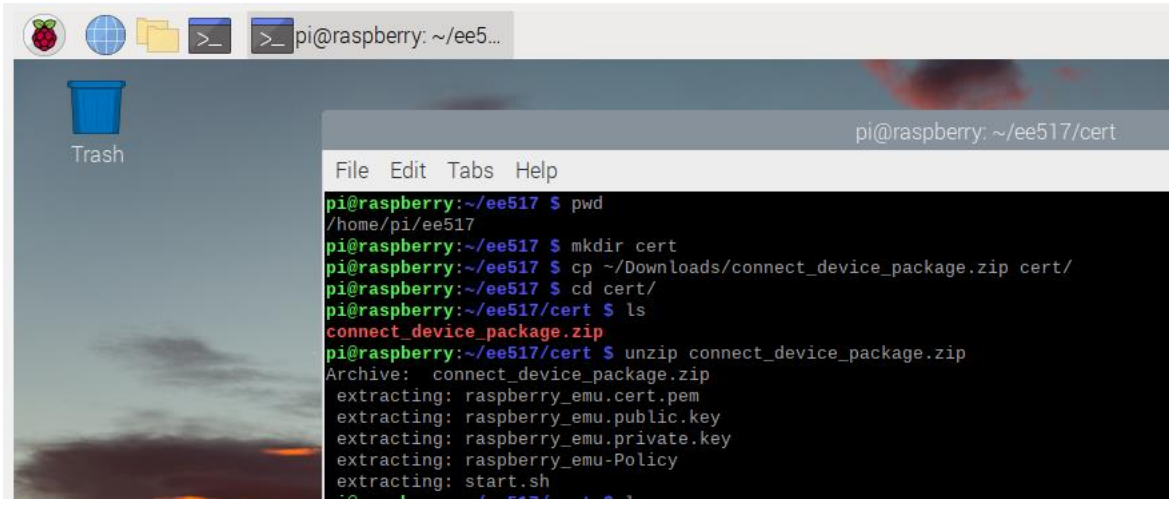
Step3. Open VirtualBox, start virtual machine – raspberry, Login AWS account using integrated browser, Create one IOT device.

Please refer to the previous week9 homework1 for details.





After creating a thing successfully, download the connection kit, upzip it in your working directory.



By default, there is no root-CA.crt, please run the highlighted command (you can find this command in start.sh):

```
pi@raspberry:~/ee517/cert $ ls
connect_device_package.zip  raspberry_emu-Policy       raspberry_emu.public.key
raspberry_emu.cert.pem      raspberry_emu.private.key  start.sh
pi@raspberry:~/ee517/cert $ grep root-CA.crt start.sh
if [ ! -f ./root-CA.crt ]; then
  curl https://www.amazontrust.com/repository/AmazonRootCA1.pem > root-CA.crt
node aws-iot-device-sdk-js-v2/samples/node/pub_sub/dist/index.js --endpoint amzxnrlhlgh93-ats.iot.us-west-2.amazonaws.com
 --key raspberry_emu.private.key --cert raspberry_emu.cert.pem --ca_file root-CA.crt --client_id sdk-nodejs-v2 --topic to
pic_1
pi@raspberry:~/ee517/cert $ curl https://www.amazontrust.com/repository/AmazonRootCA1.pem > root-CA.crt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1188  100  1188    0     0   8671      0 --:--:-- --:--:-- --:--:--  8671
pi@raspberry:~/ee517/cert $ ls
connect_device_package.zip  raspberry_emu-Policy       raspberry_emu.public.key  start.sh
raspberry_emu.cert.pem      raspberry_emu.private.key  root-CA.crt
pi@raspberry:~/ee517/cert $
```

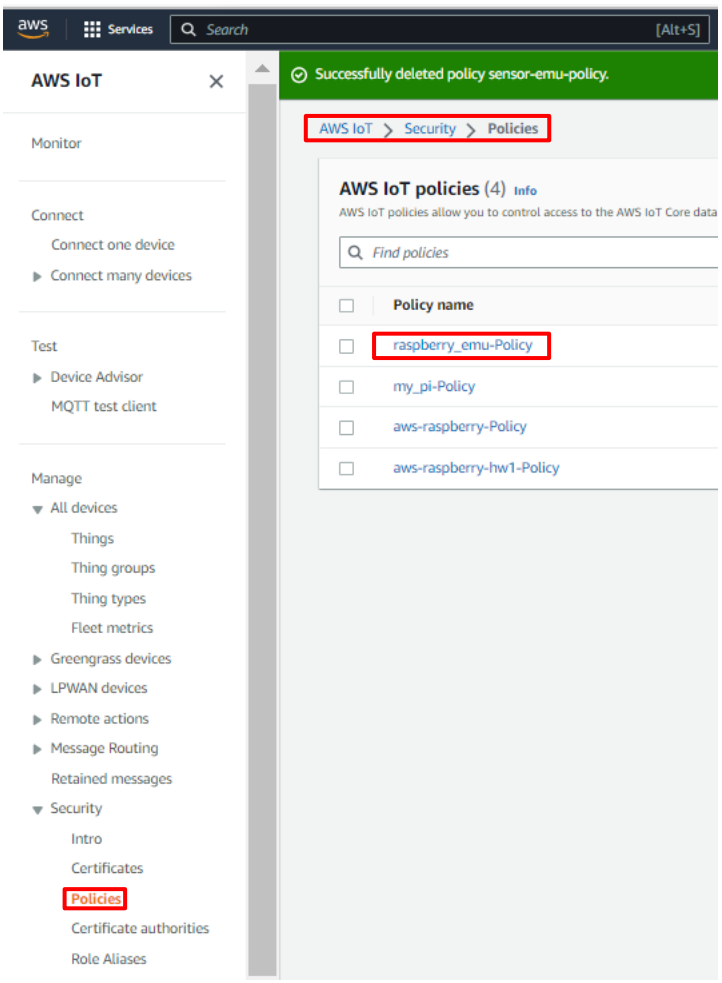
Notice:

In start.sh, you also can find all information we need to use while you want to connect to this AWS IOT device “raspberry-emu” in your *.js program:

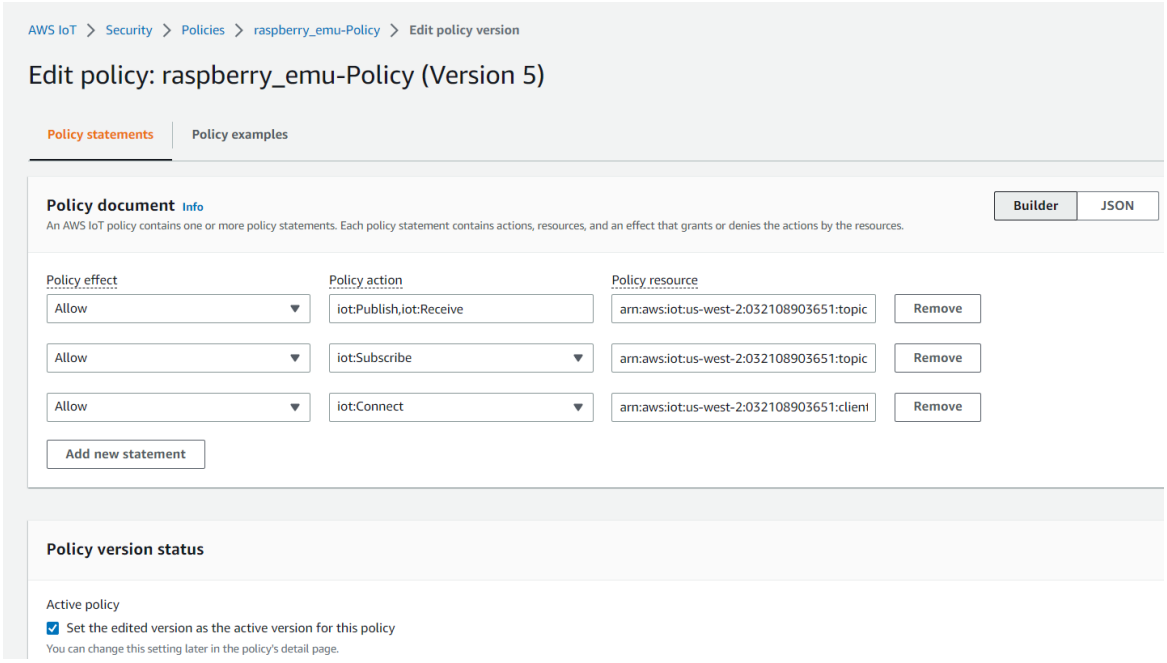
```
pi@raspberry:~/ee517/cert $ grep node start.sh
  cd samples/node/pub_sub
node aws-iot-device-sdk-js-v2/samples/node/pub_sub/dist/index.js --endpoint amzxnrlhlgh93-ats.iot.us-west-2.amazonaws.com
--key raspberry_emu.private.key --cert raspberry_emu.cert.pem --ca_file root-CA.crt --client_id sdk-nodejs-v2 --topic topi
c_1
```

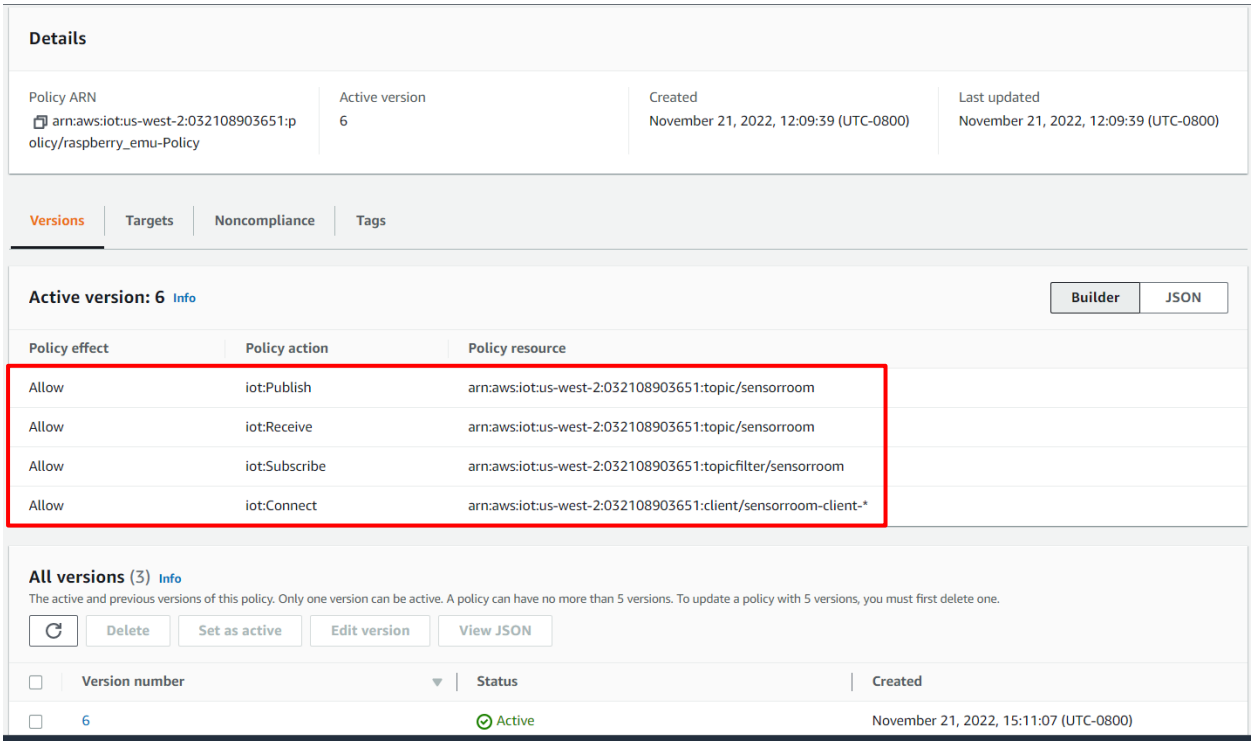
The endpoint/host, key path and cert path are fixed as mentioned above (red part).

Then client id and topic can be edited if you want to use your own name (yellow part).



Edit the policy:





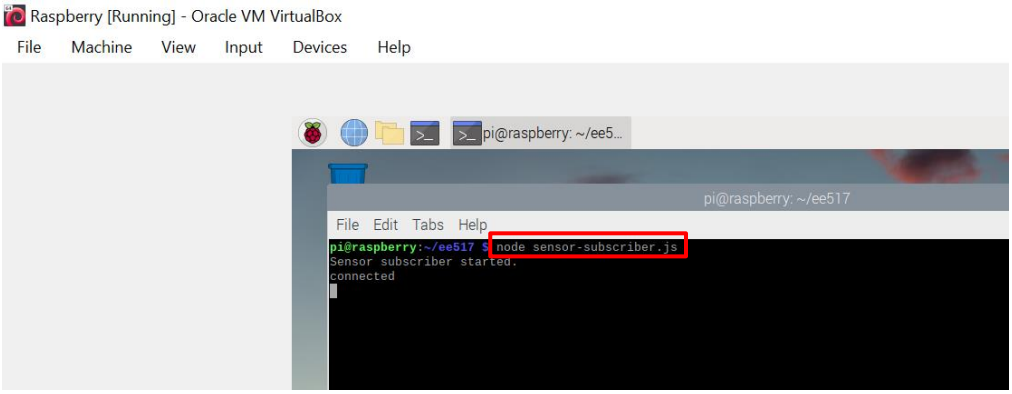
Step4. Create sensor-subscriber.js file by link [Developing a sensor subscriber (sfbu.edu)]

```
File  Edit  Tabs  Help
var awsIot = require('aws-iot-device-sdk')

var device = awsIot.device({
        keyPath: 'cert/raspberry_emu.private.key',
        certPath: 'cert/raspberry_emu.cert.pem',
        caPath: 'cert/root-CA.crt',
        host: 'amzxnrlhlgh93-ats.iot.us-west-2.amazonaws.com',
        clientId: 'sensorroom-client-0',
        region: 'us-west-'
});

device.on('connect', function() {
        console.log('connected');
        device.subscribe('sensorroom');
});

device.on('message', function(topic, payload) {
        console.log('recv:', topic, payload.toString());
});

console.log('Sensor subscriber started.');
~
~
~
```

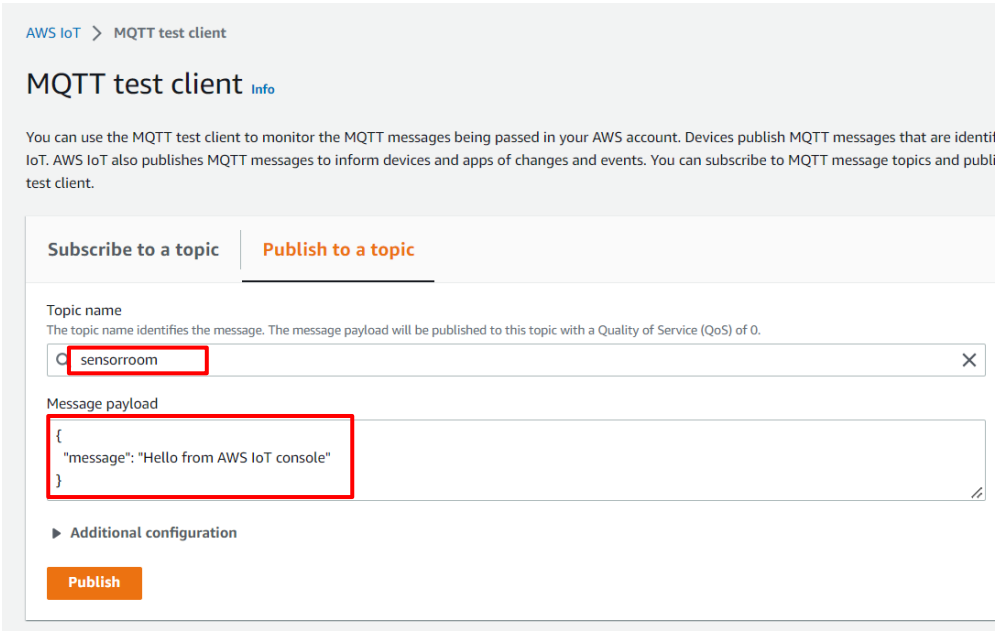
Step5. Run the script sensor-subscriber.js

On the raspberry virtual machine side:

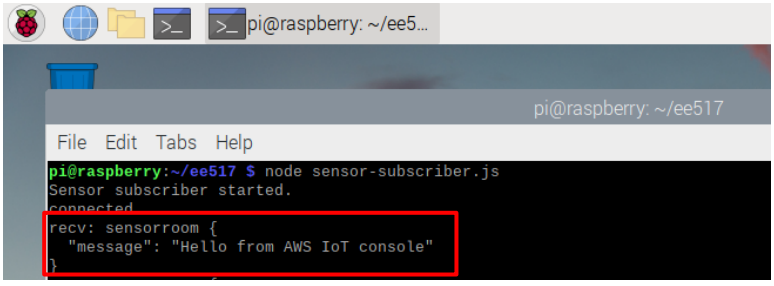
$ node sensor-subscriber.js



On the AWS IOT MQTT client side, publish a message:



On the raspberry virtual machine side, it will display the message as below:



DONE!!!