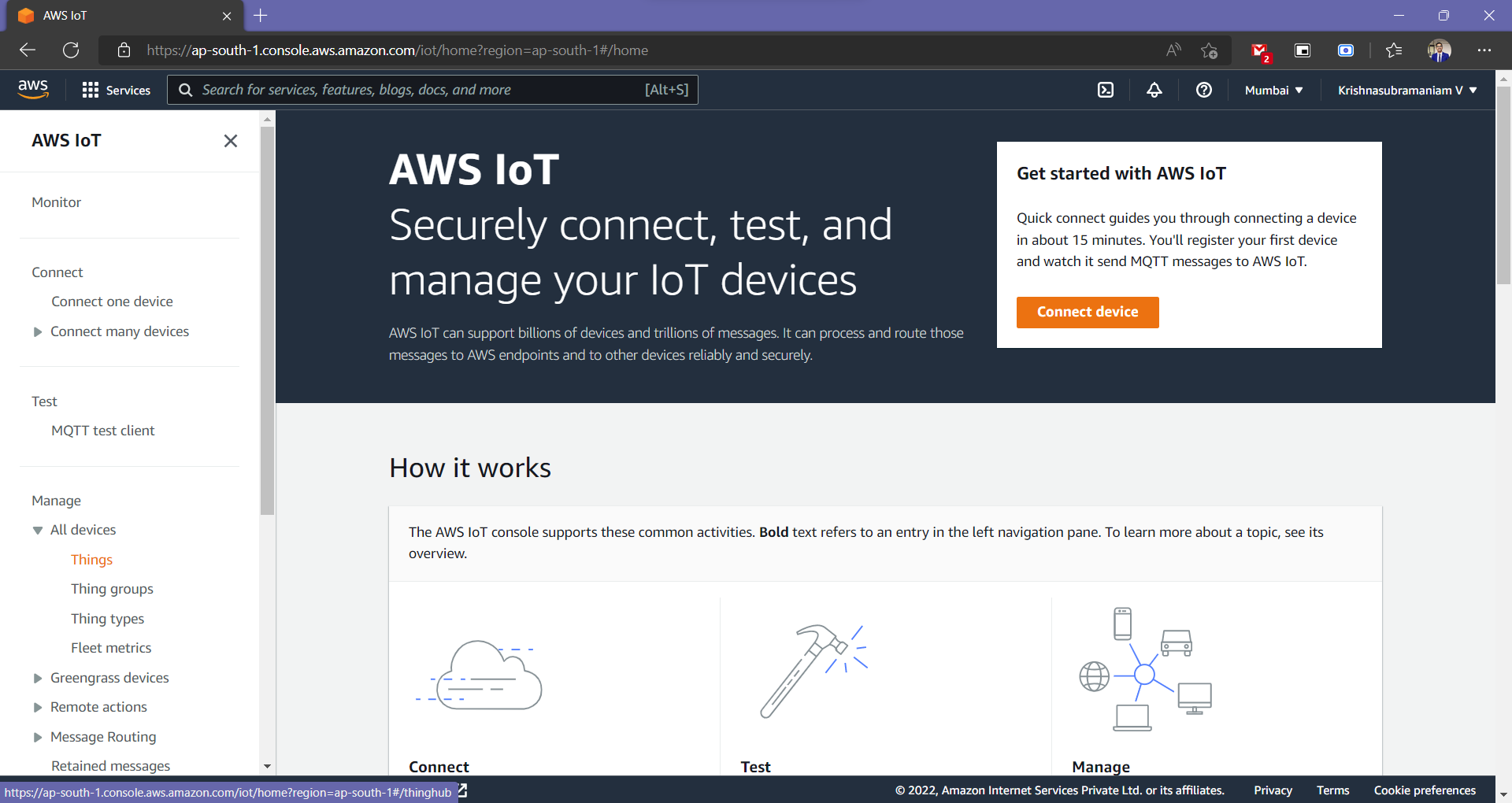
**IOE Lab**

**ASSIGNMENT NO. 2**

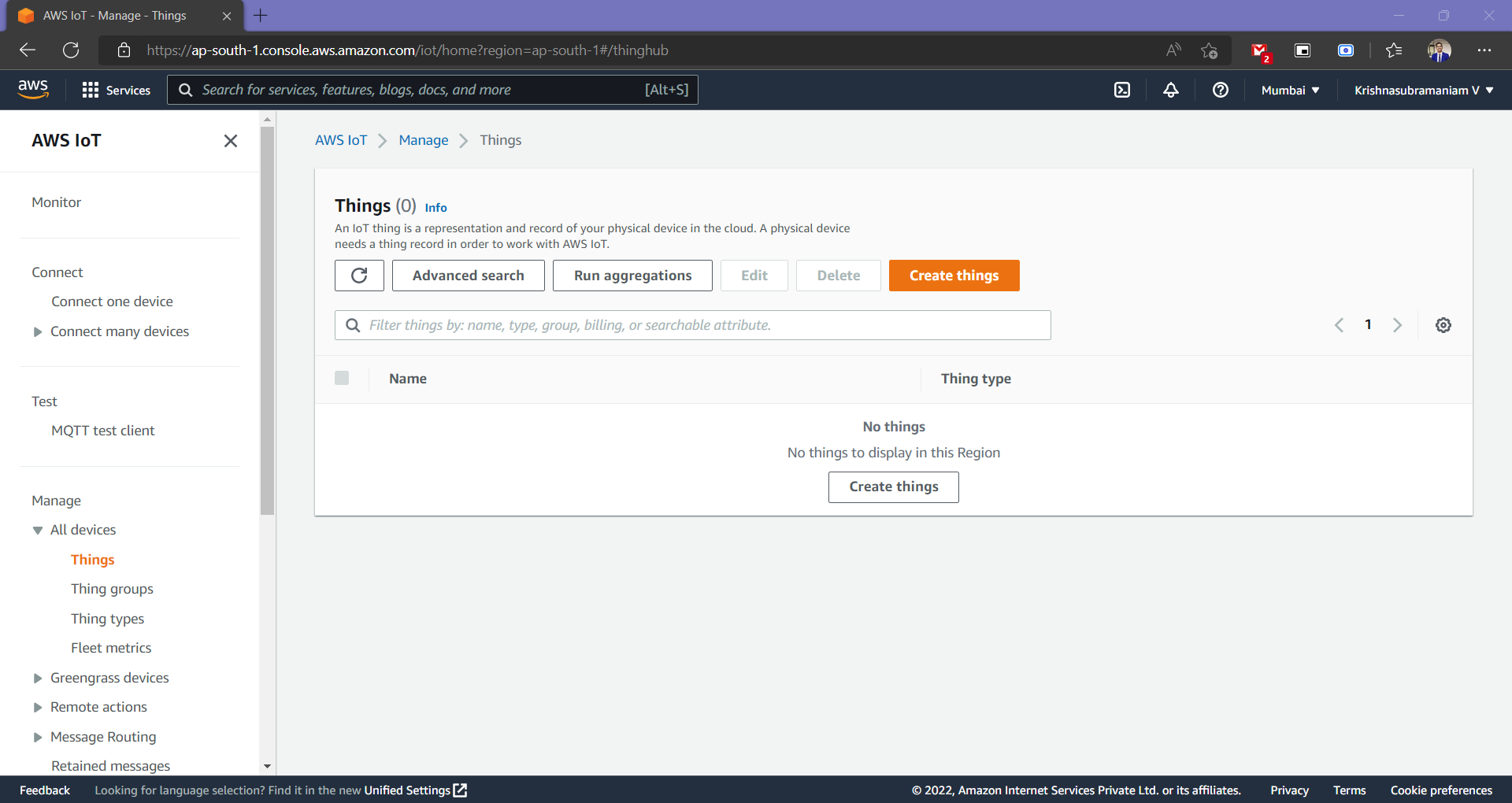
**Aim**: Transfer of data using MQTT/REST/CoAP protocol

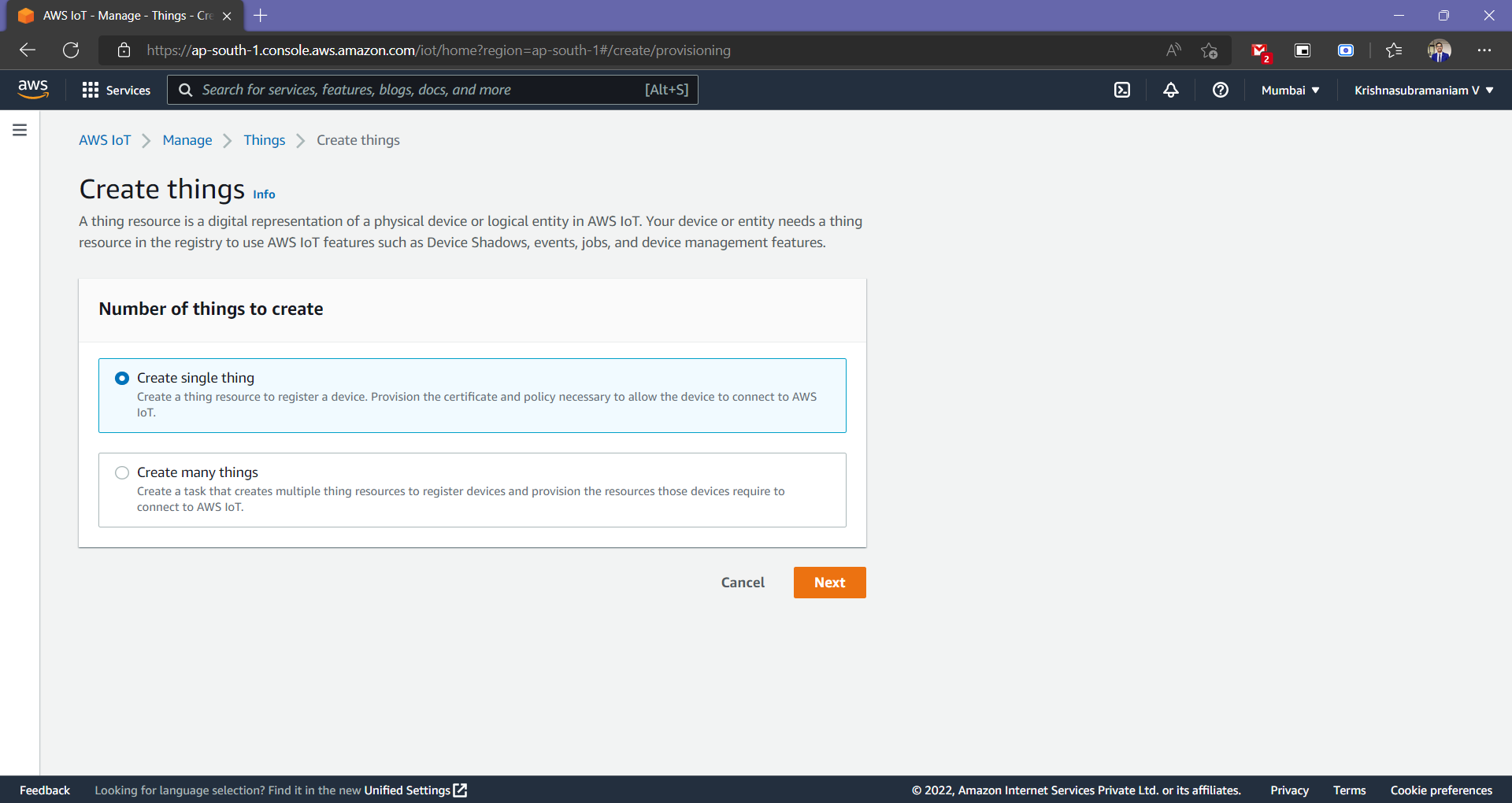
**Transfer the data from the file (use files) on to the channel/things created on AWS using MQTT/REST/CoAP protocol.**

1. Sign in into the AWS account. After successfully signing in, the AWS Management Console window will open. In the services search tab at the top write ‘IoT core’ & hit enter.
2. You can click on IoT Core, so an AWS IoT Dashboard will appear now.

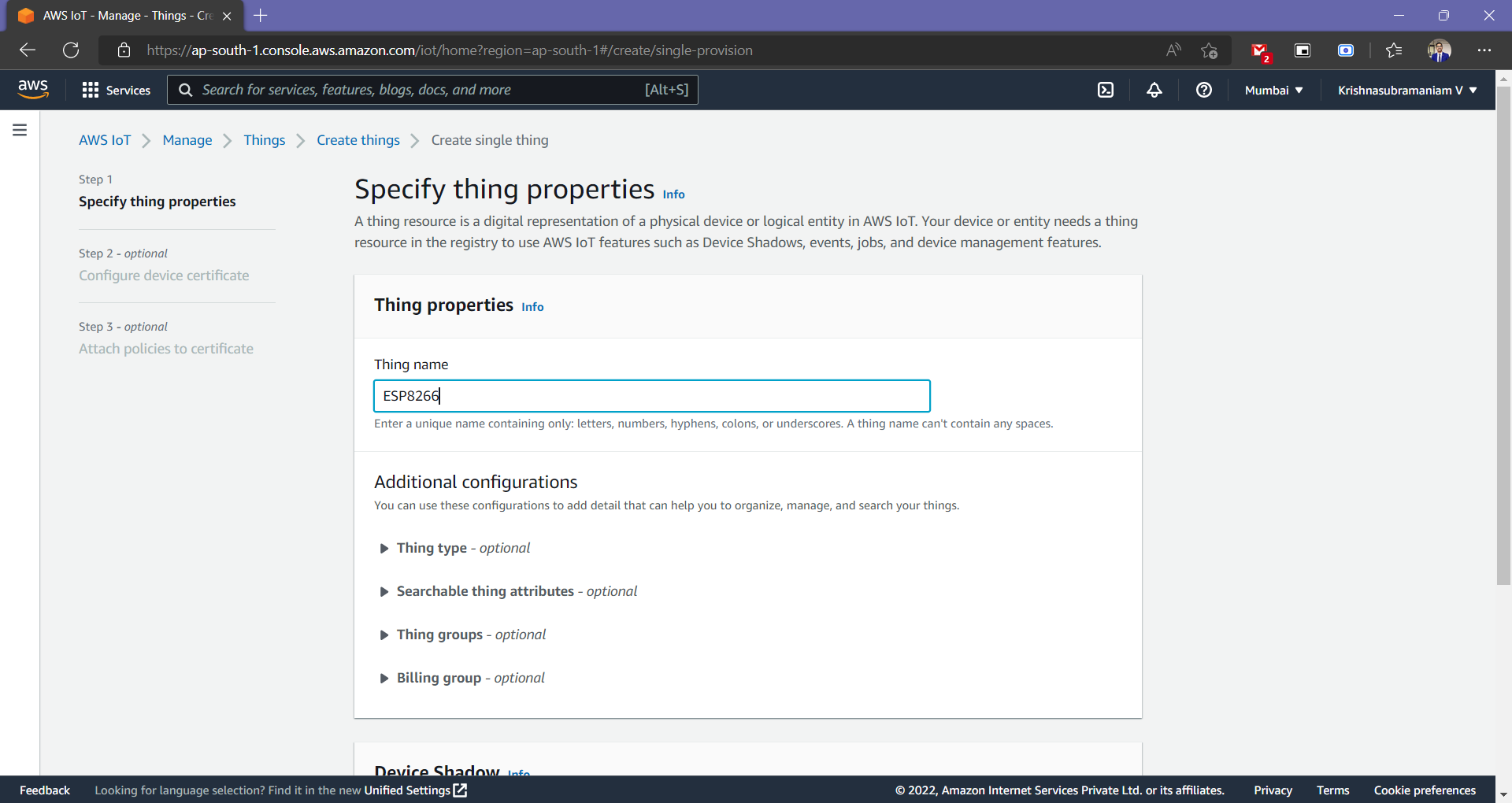


1. Go to the Things section. Create a single thing. Then click on Next.

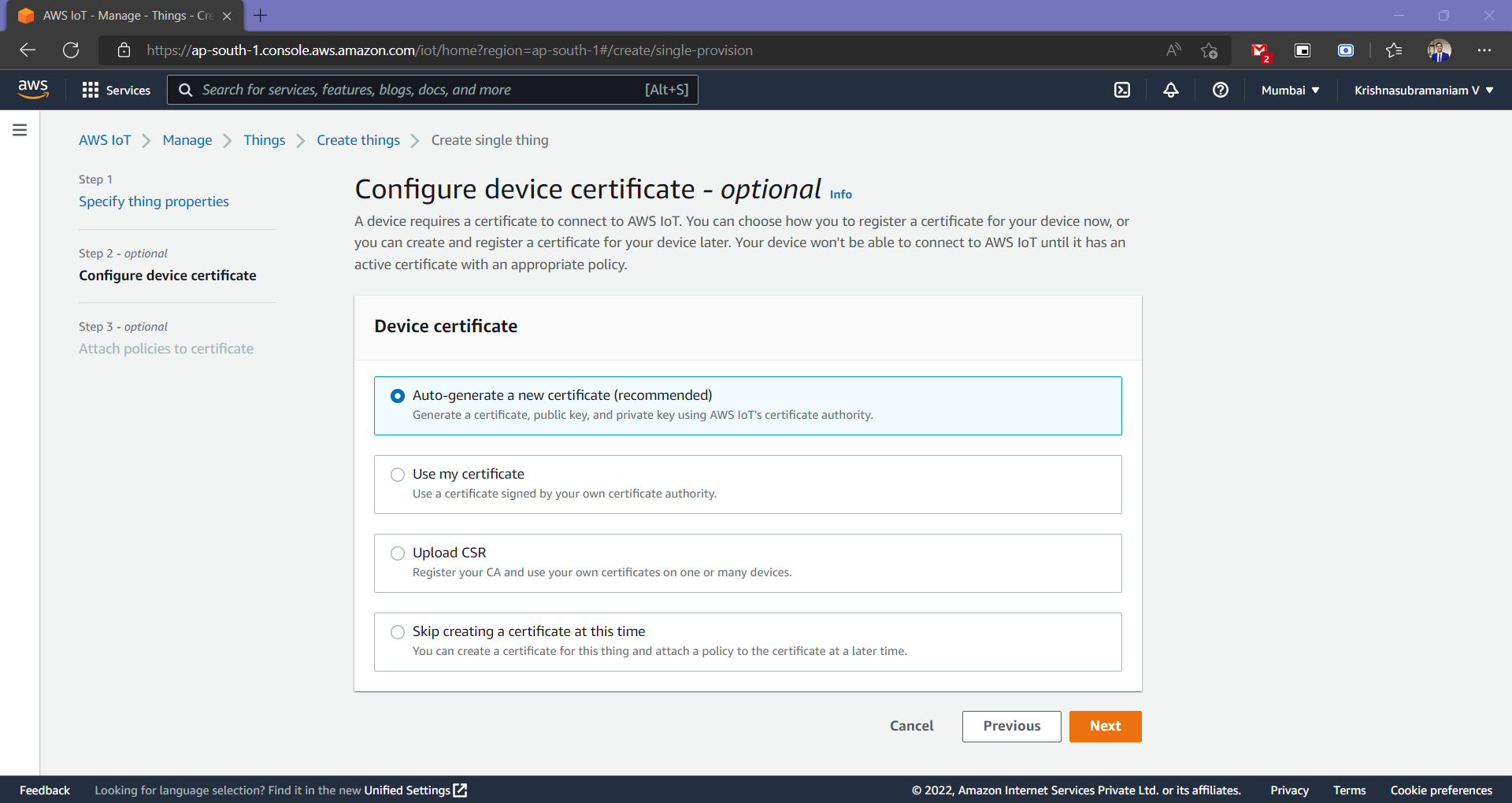




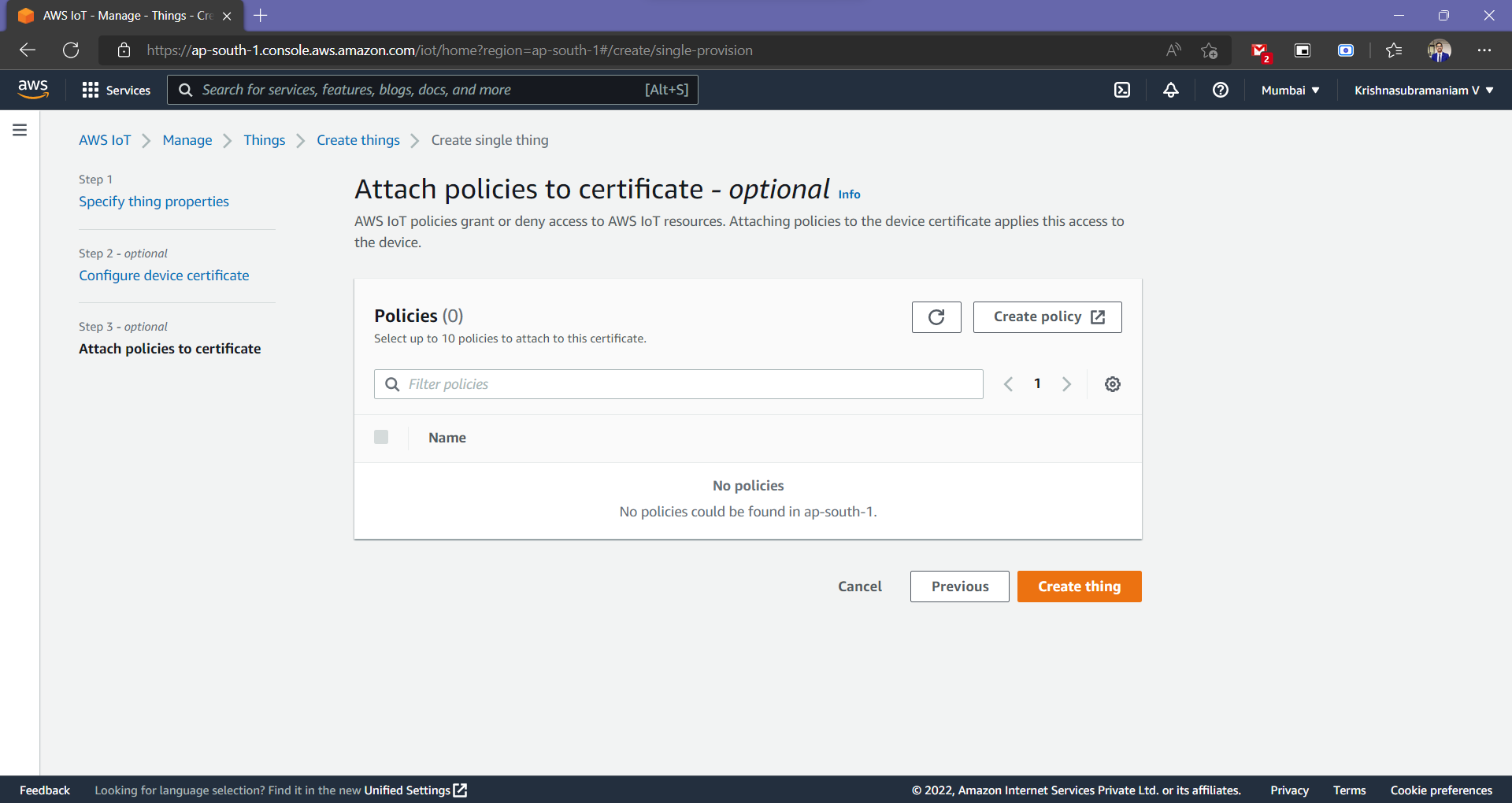
1. Now we need to specify the Thing properties. First, give a thing a name. You can name it anything. For example, I will name it ESP8266. Under additional configurations, there is no need to make any changes. Under the device shadow option, select the first option as No shadow. Then click on Next.



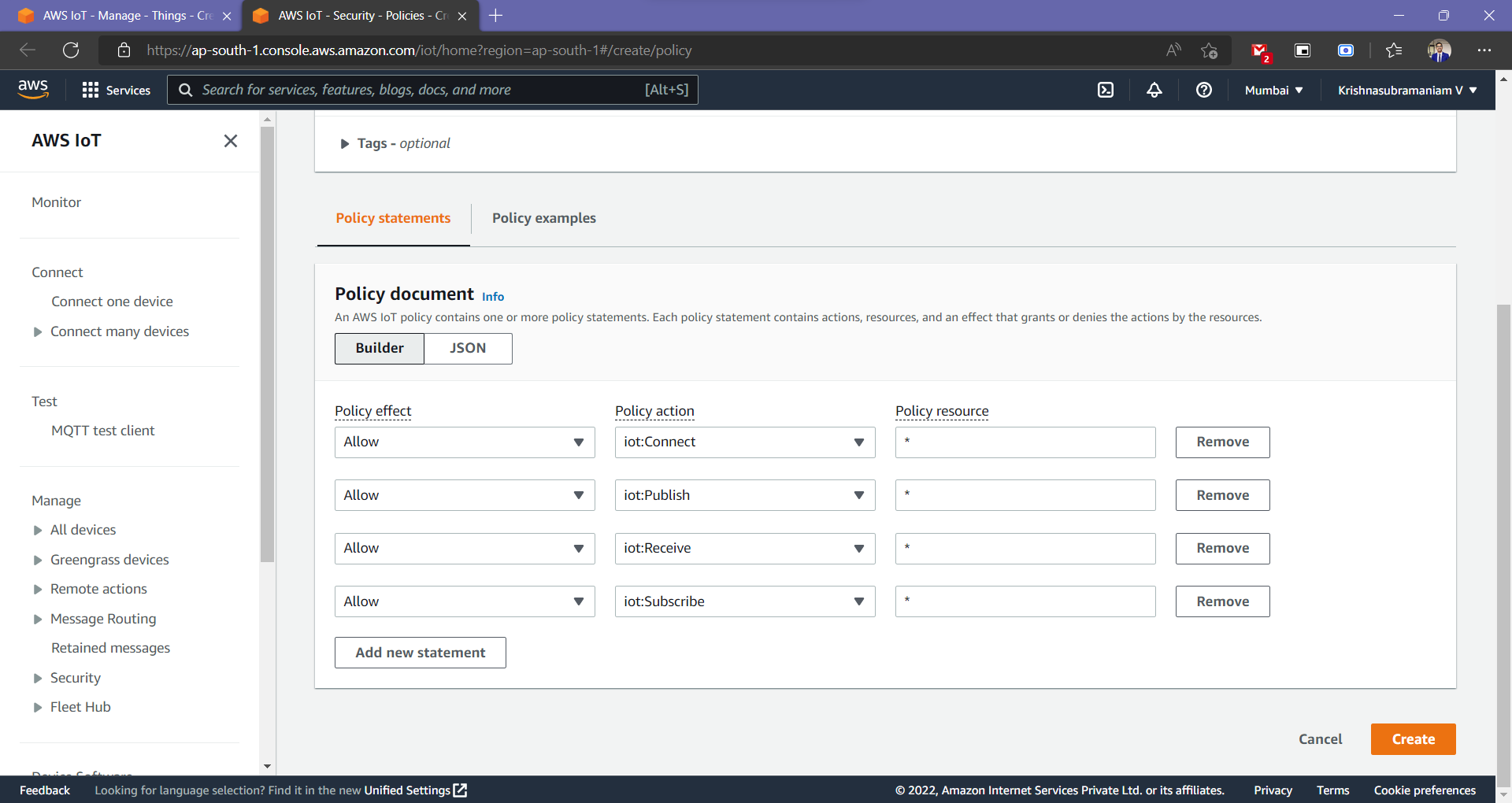
1. Now you need to configure the device certificate. So here you can auto-generate a new certificate or use your own certificate or upload CSR or skip this. But the AWS recommendation is to select the Auto Generate New Certificate. Then click on Next.



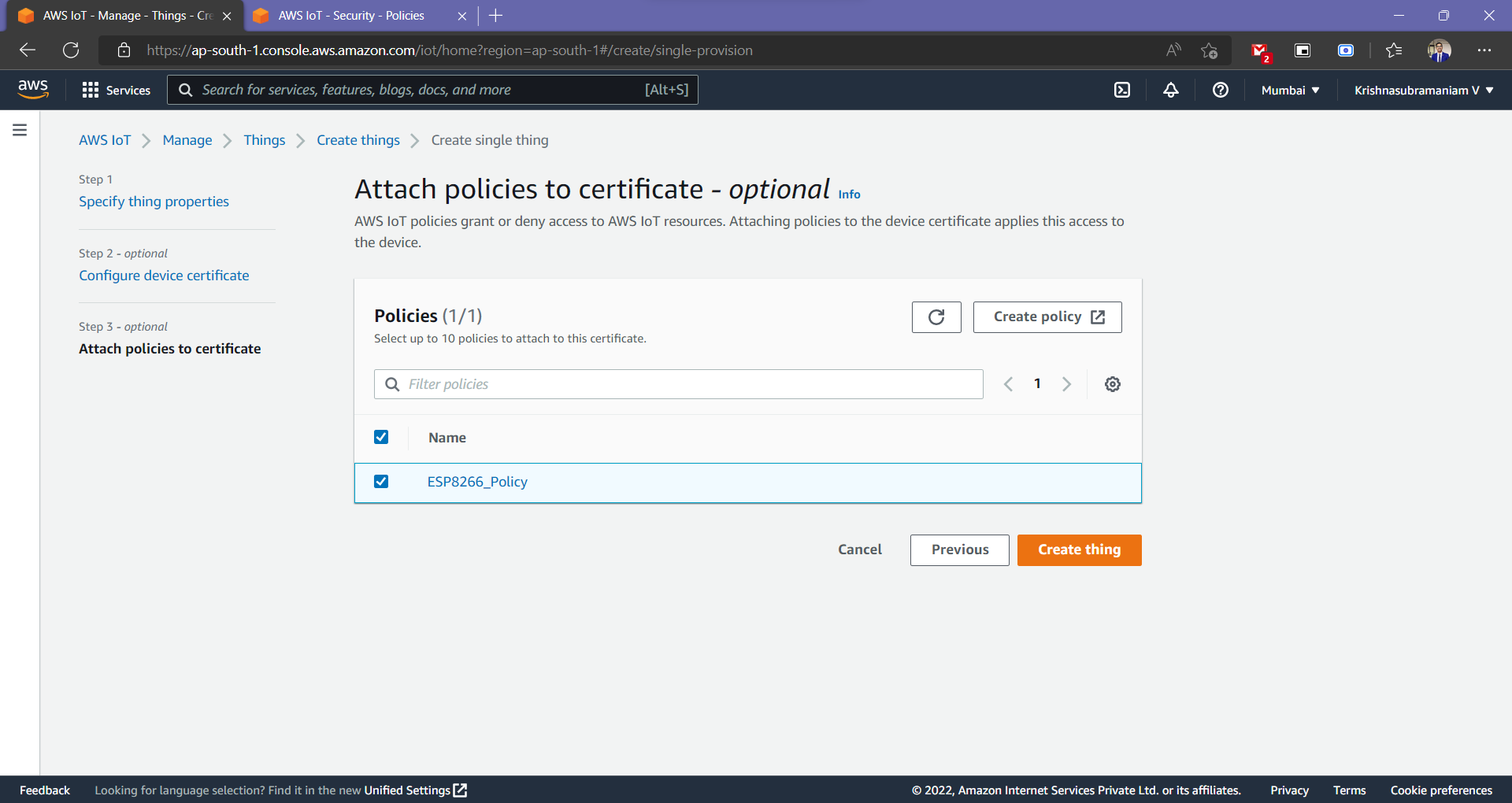
1. Now we need to attach a policy to the Things we created. But no policies are here right now. So we need to create a policy first.



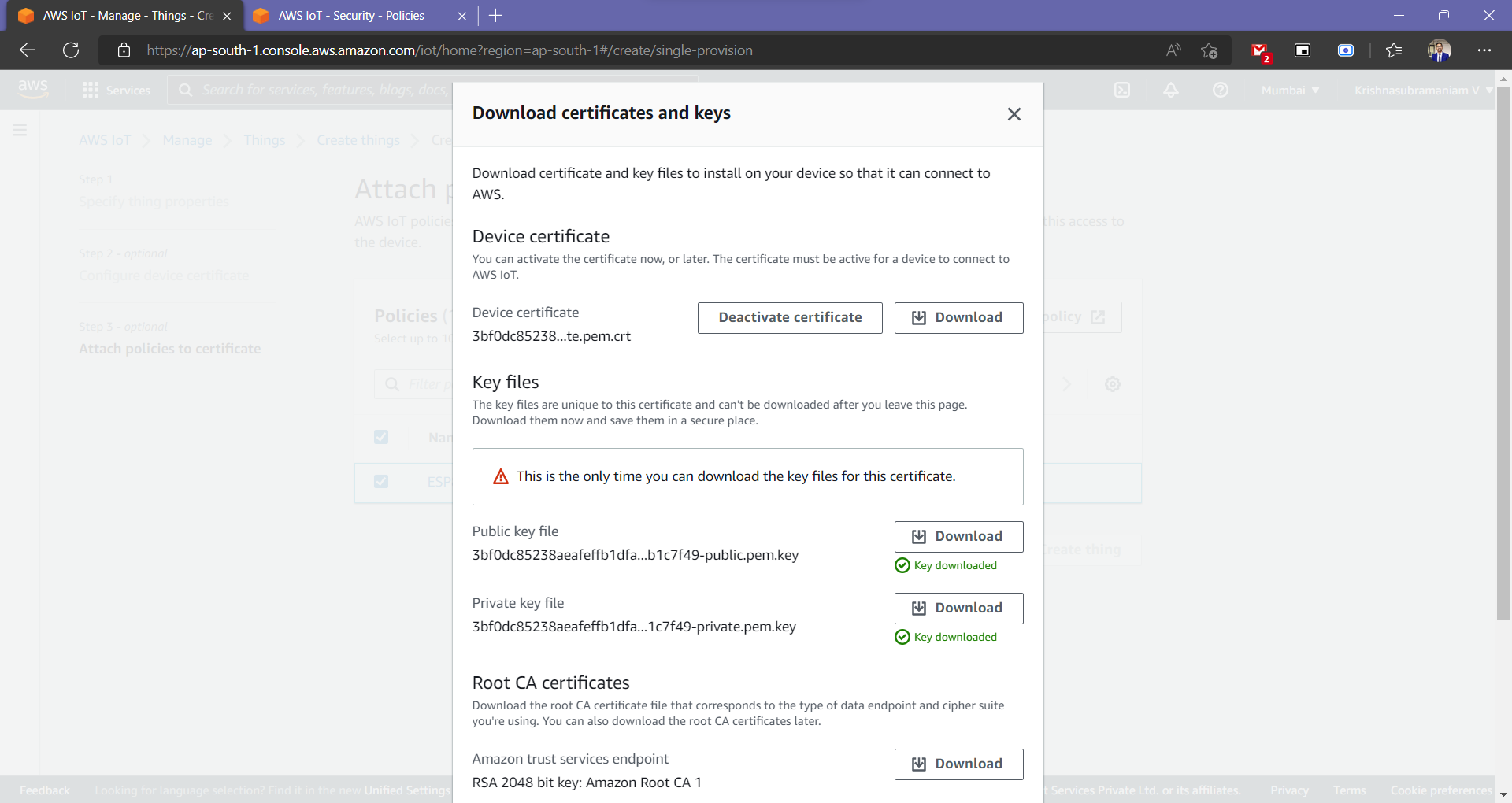
1. So click on create policy. Here give any name to the policy. Under the action, type IoT. So multiple options will pop up. From here we will only need to Publish, Subscribe, Connect and Receive. Now click on create to create the policy. So the policy has been created successfully.



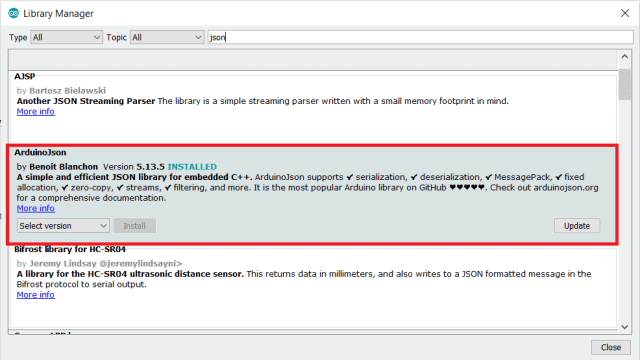
1. Now go back to the Create Thing option. So a policy option will appear. We need to attach the policies to the certificate. So select the appeared policy and click on create a thing.



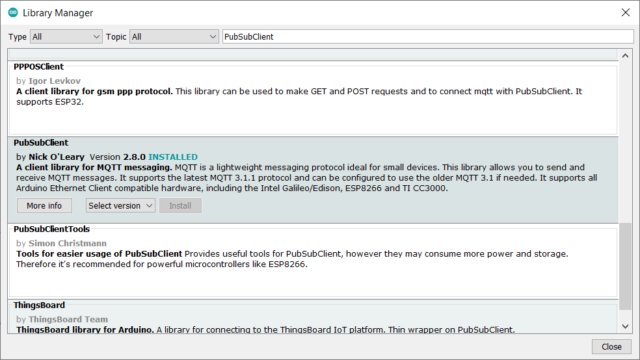
1. Now we need to download the required certificates from this list. First, download the device certificate and then rename it as a device certificate for identification. Also, download the public key and rename it as a public key. Then download the private key and rename it as a private key.
2. In the Root CA Certificates, there are two certificates here. But we just need a Root CA1 certificate, so download it as well. So we have downloaded all the certificates that we need for our project.



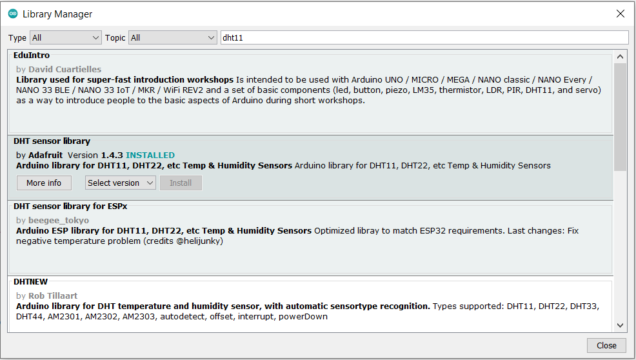
1. Install necessary Arduino Libraries. So first go to the library manager and search for “JSON” & install the library as shown in the figure below.



1. Again go to the library manager and search for “PubSubClient” & install the library from Nick O’Leary.



1. Search for “DHT11” & install the library as shown below.



1. Now to write Source Code/Program for connecting AWS IoT Core with ESP8266. Open a new sketch in Arduino IDE & paste the following code and save it as Main.ino.

| *#include <ESP8266WiFi.h>* *#include <WiFiClientSecure.h>* *#include <PubSubClient.h>* *#include <ArduinoJson.h>* *#include <time.h>* *#include "secrets.h"* *#include "DHT.h"*   *#define DHTPIN 4 // Digital pin connected to the DHT sensor* *#define DHTTYPE DHT11 // DHT 11*   DHT dht(DHTPIN, DHTTYPE);   float h ; float t; unsigned long lastMillis = 0; unsigned long previousMillis = 0; const long interval = 5000;   *#define AWS\_IOT\_PUBLISH\_TOPIC "esp8266/pub"* *#define AWS\_IOT\_SUBSCRIBE\_TOPIC "esp8266/sub"*   WiFiClientSecure net;   BearSSL::X509List cert(cacert); BearSSL::X509List client\_crt(client\_cert); BearSSL::PrivateKey key(privkey);   PubSubClient client(net);   time\_t now; time\_t nowish = 1510592825;     void NTPConnect(void) {  Serial.print("Setting time using SNTP");  configTime(TIME\_ZONE \* 3600, 0 \* 3600, "pool.ntp.org", "time.nist.gov");  now = time(nullptr);  while (now < nowish)  {  delay(500);  Serial.print(".");  now = time(nullptr);  }  Serial.println("done!");  struct tm timeinfo;  gmtime\_r(&now, &timeinfo);  Serial.print("Current time: ");  Serial.print(asctime(&timeinfo)); }     void messageReceived(char \*topic, byte \*payload, unsigned int length) {  Serial.print("Received [");  Serial.print(topic);  Serial.print("]: ");  for (int i = 0; i < length; i++)  {  Serial.print((char)payload[i]);  }  Serial.println(); }     void connectAWS() {  delay(3000);  WiFi.mode(WIFI\_STA);  WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);    Serial.println(String("Attempting to connect to SSID: ") + String(WIFI\_SSID));    while (WiFi.status() != WL\_CONNECTED)  {  Serial.print(".");  delay(1000);  }    NTPConnect();    net.setTrustAnchors(&cert);  net.setClientRSACert(&client\_crt, &key);    client.setServer(MQTT\_HOST, 8883);  client.setCallback(messageReceived);      Serial.println("Connecting to AWS IOT");    while (!client.connect(THINGNAME))  {  Serial.print(".");  delay(1000);  }    if (!client.connected()) {  Serial.println("AWS IoT Timeout!");  return;  }  // Subscribe to a topic  client.subscribe(AWS\_IOT\_SUBSCRIBE\_TOPIC);    Serial.println("AWS IoT Connected!"); }     void publishMessage() {  StaticJsonDocument<200> doc;  doc["time"] = millis();  doc["humidity"] = h;  doc["temperature"] = t;  char jsonBuffer[512];  serializeJson(doc, jsonBuffer); // print to client    client.publish(AWS\_IOT\_PUBLISH\_TOPIC, jsonBuffer); }     void setup() {  Serial.begin(115200);  connectAWS();  dht.begin(); }     void loop() {  h = dht.readHumidity();  t = dht.readTemperature();    if (isnan(h) || isnan(t) ) // Check if any reads failed and exit early (to try again).  {  Serial.println(F("Failed to read from DHT sensor!"));  return;  }    Serial.print(F("Humidity: "));  Serial.print(h);  Serial.print(F("% Temperature: "));  Serial.print(t);  Serial.println(F("°C "));  delay(2000);    now = time(nullptr);    if (!client.connected())  {  connectAWS();  }  else  {  client.loop();  if (millis() - lastMillis > 5000)  {  lastMillis = millis();  publishMessage();  }  } } |
| --- |

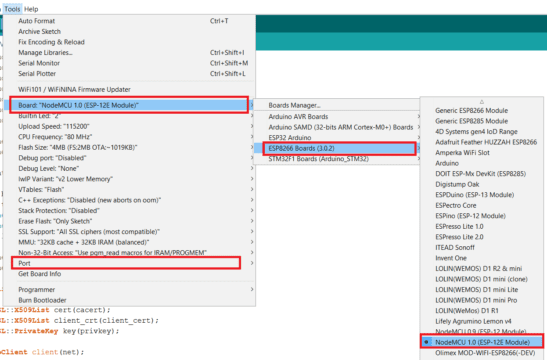
1. Open a New Tab in Arduino IDE and name it as Secrets.h. And paste the following code to this file.

| *#include <pgmspace.h>*   *#define SECRET*   const char WIFI\_SSID[] = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"; //TAMIM2.4G const char WIFI\_PASSWORD[] = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"; //0544287380   *#define THINGNAME "\*\*\*\*\*\*\*\*\*\*\*\*\*\*"*   int8\_t TIME\_ZONE = -5; //NYC(USA): -5 UTC   const char MQTT\_HOST[] = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";     static const char cacert[] PROGMEM = R"EOF( -----BEGIN CERTIFICATE-----   -----END CERTIFICATE----- )EOF";     // Copy contents from XXXXXXXX-certificate.pem.crt here ▼ static const char client\_cert[] PROGMEM = R"KEY( -----BEGIN CERTIFICATE-----   -----END CERTIFICATE-----   )KEY";     // Copy contents from XXXXXXXX-private.pem.key here ▼ static const char privkey[] PROGMEM = R"KEY( -----BEGIN RSA PRIVATE KEY-----   -----END RSA PRIVATE KEY-----   )KEY"; |
| --- |

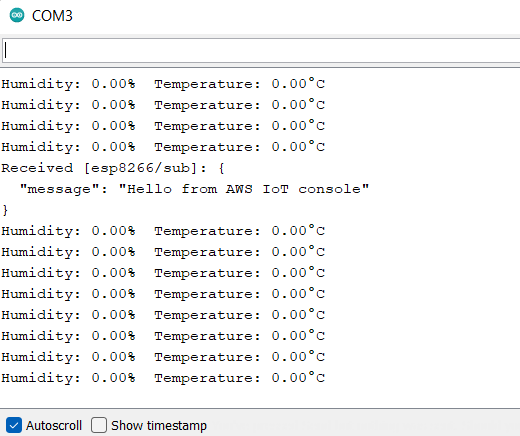
1. Now it’s time to modify the Arduino Sketch File. Go to secrets.h tab and begin the modification.

* Paste the thing name to the following line of code.
* Under the WiFi SSID and password, enter the WiFi SSID and Password of your local network.
* Click on the copy icon to copy the endpoint. Go back to Arduino IDE and paste it on the following line.
* You need to insert the Amazon Root CA1 in between the following line.
* Under the “Device Certificate” lines, we need to paste the device certificate text.
* Under the “Device Private Key“, we need to insert the device’s private key.

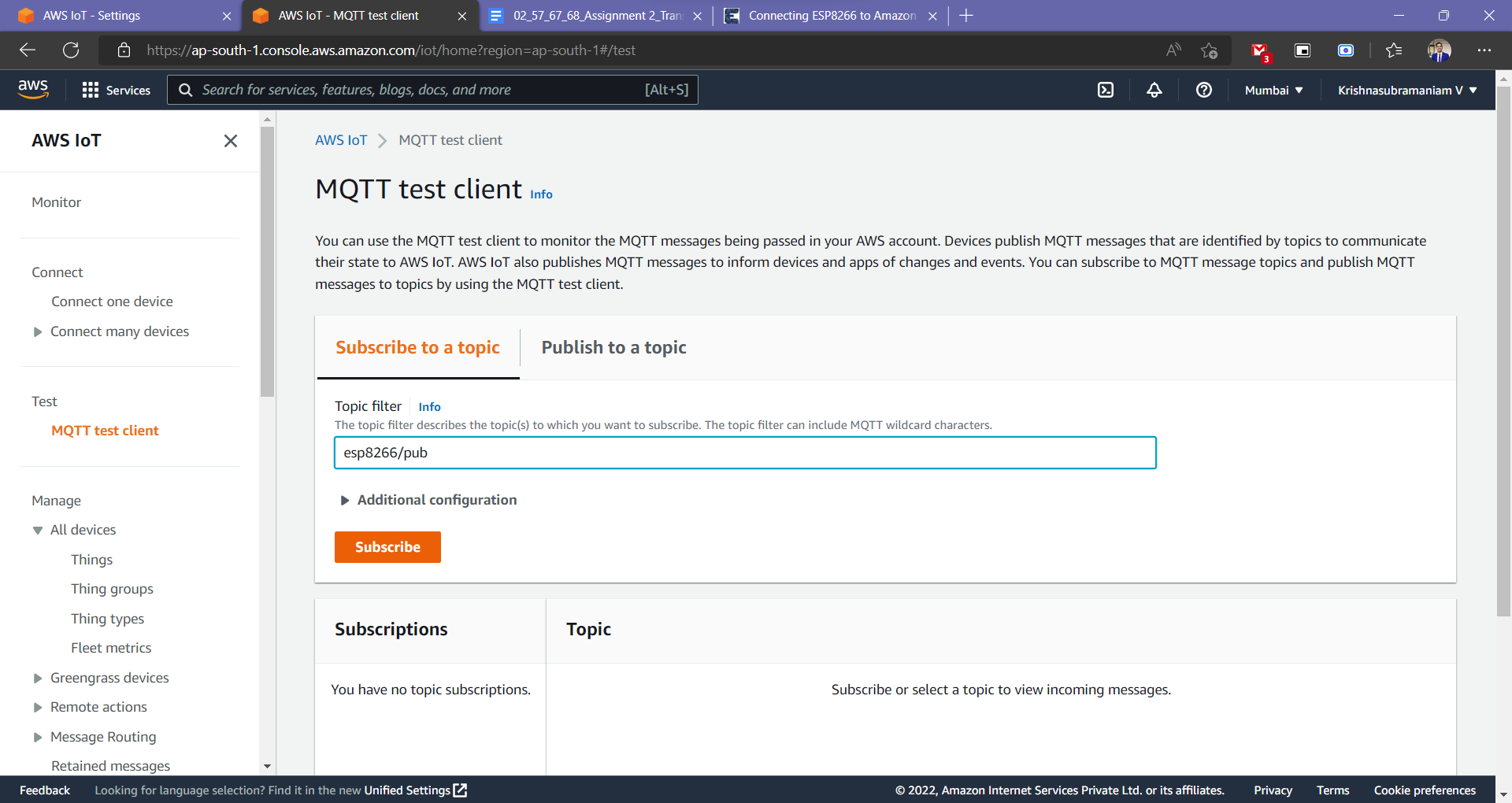
1. Once all the modification is done, connect the NodeMCU ESP8266 to your computer. Then go to the tools & select NodeMCU 1.0 Board that you are using for this project. Also, select the COM port. Then click on the upload option to upload the code to the ESP8266 board.



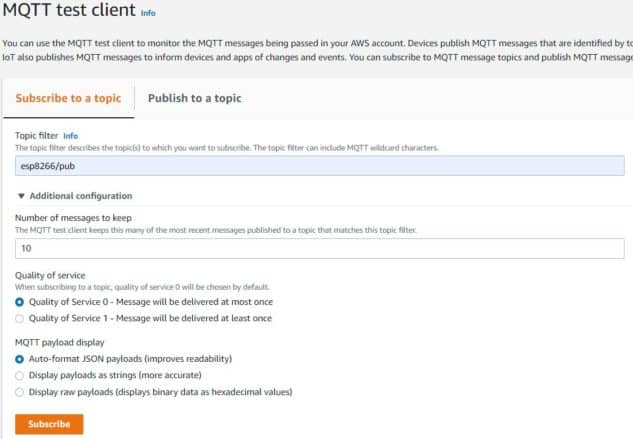
1. Once the code uploading is done, open the Serial Monitor. The ESP8266 will try connecting to the WiFi Network. Once it gets connected to the WiFi Network, it will try connecting to the AWS IoT Server.



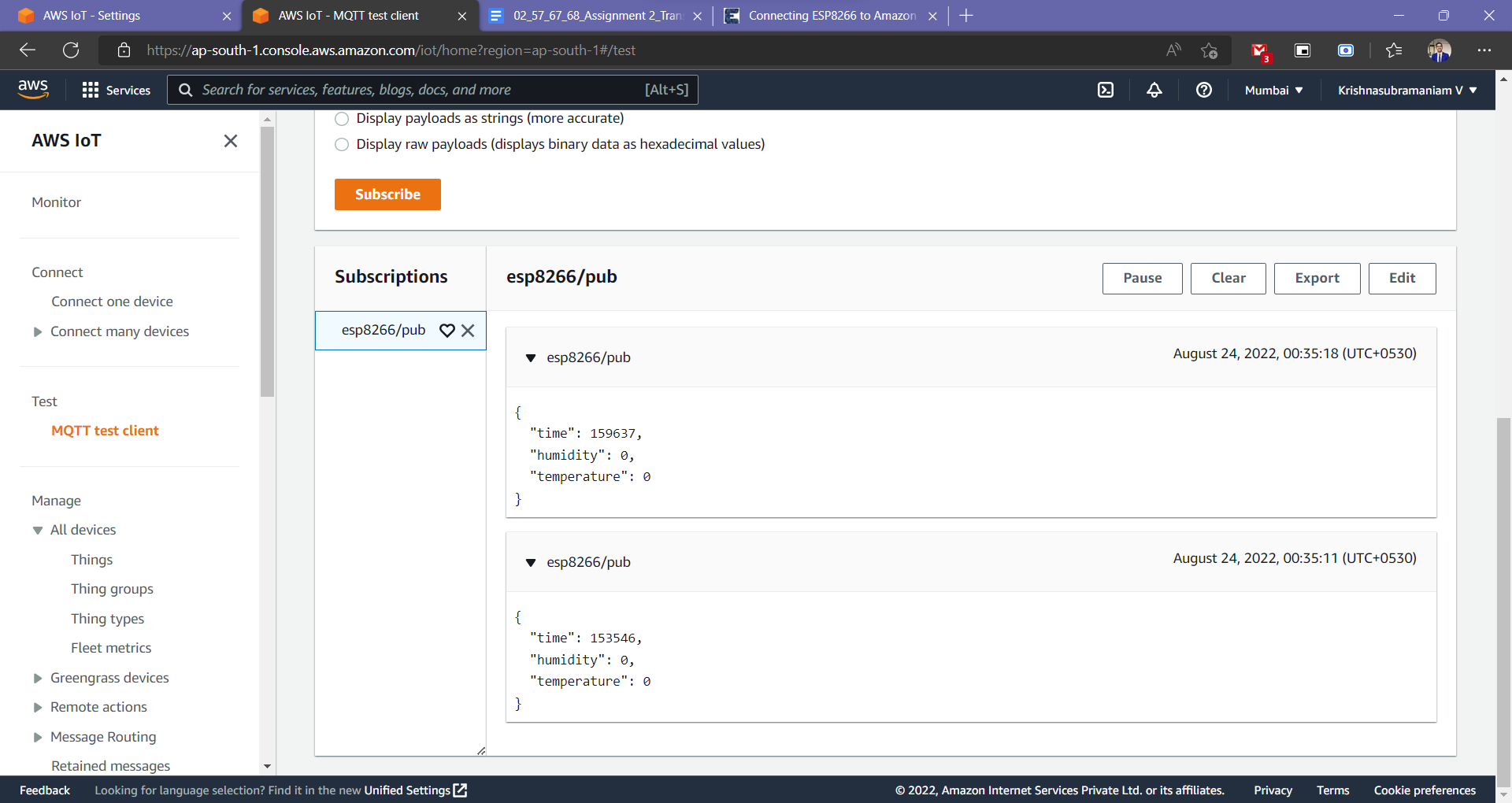
1. Now we need to subscribe to Sensor Data to AWS Dashboard. To check that, go to the test section of AWS Dashboard. Under the test section, we have an option for subscribe and publish.



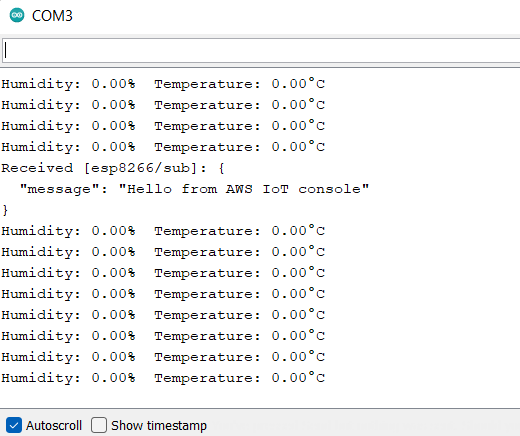
1. Now to see the data, you need to subscribe to a topic. For that type “esp8266/pub“ under the topic filter section. In the additional configuration, you can make changes if you want.



1. Then click on subscribe. When you hit the subscribe button, immediately the data from ESP8266 will be uploaded to AWS Dashboard. Thus, you have successfully sent the DHT11 Sensor data & time data to Amazon AWS IoT Core using ESP8266.



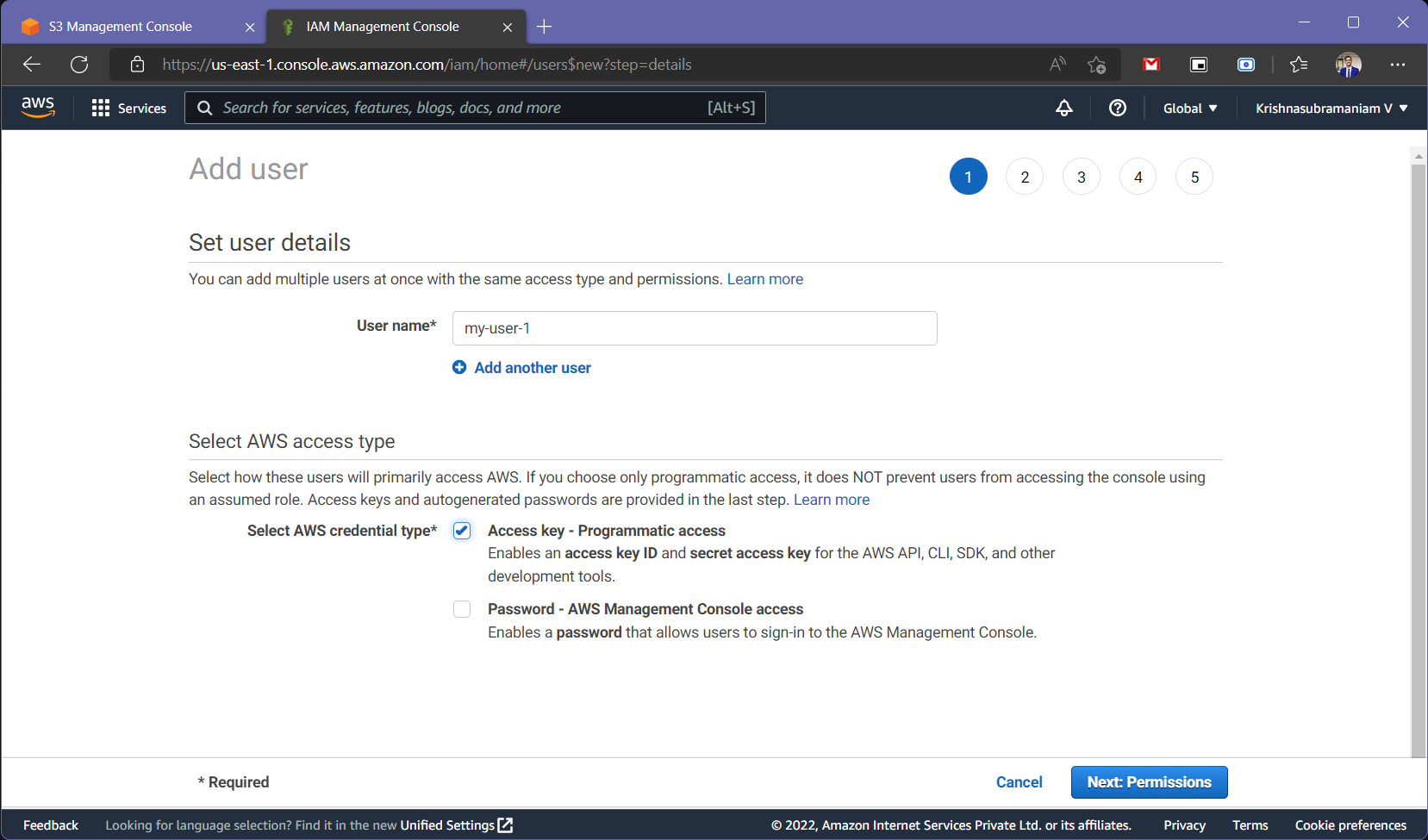
1. Now we need to publish the Data to the Serial Monitor. For that type “esp8266/sub“ under the topic filter section. Under additional configuration do nothing. Then click on publish. Immediately you can see the message sent to the Serial Monitor. You can use this method to Control an LED using the publish method.



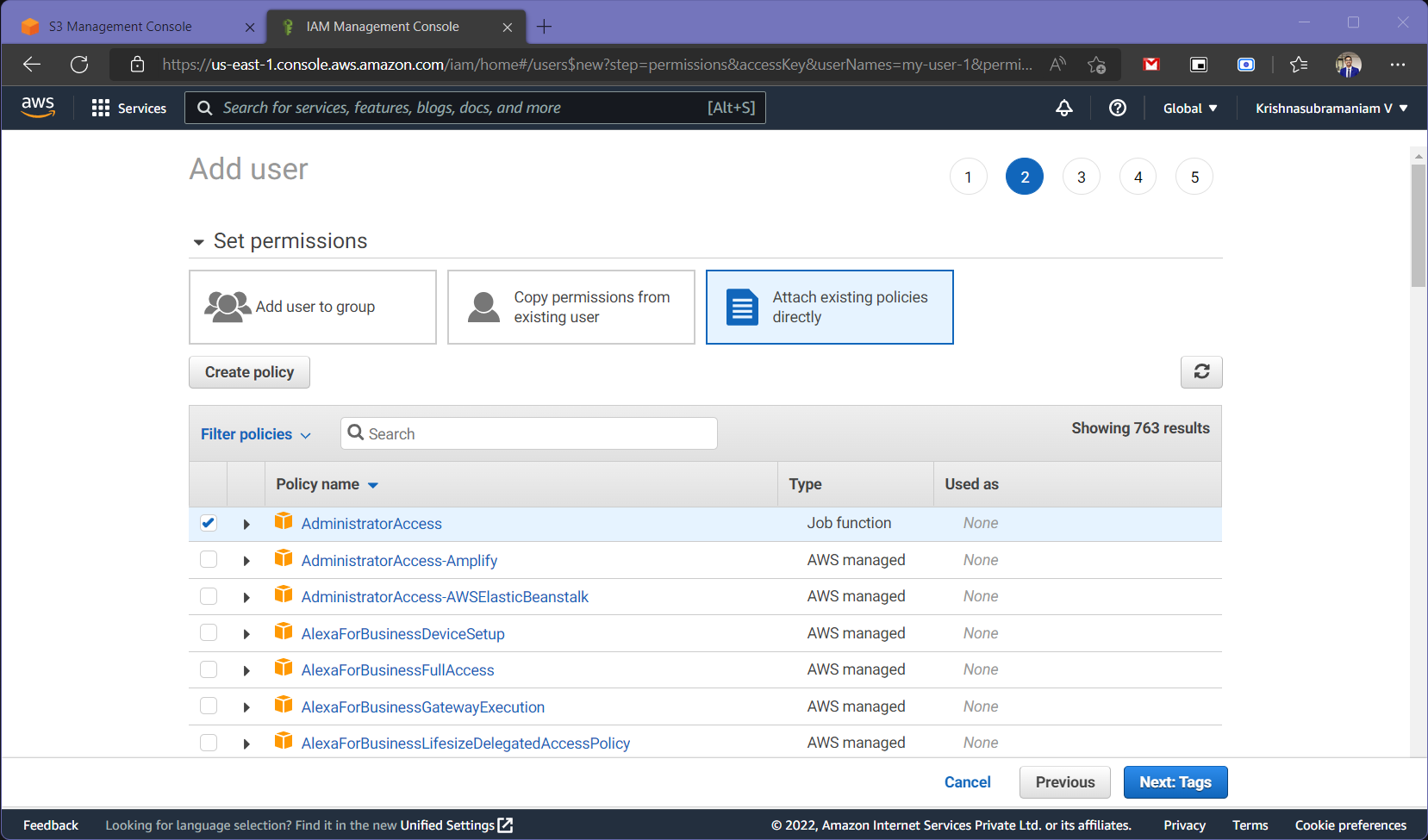
This is how you can send or receive data from Amazon AWS IoT Core using ESP8266 using AWS MQTT.

**Transfer the data through a program (like dummy data generated using a random function) on the channel/things created on AWS.**

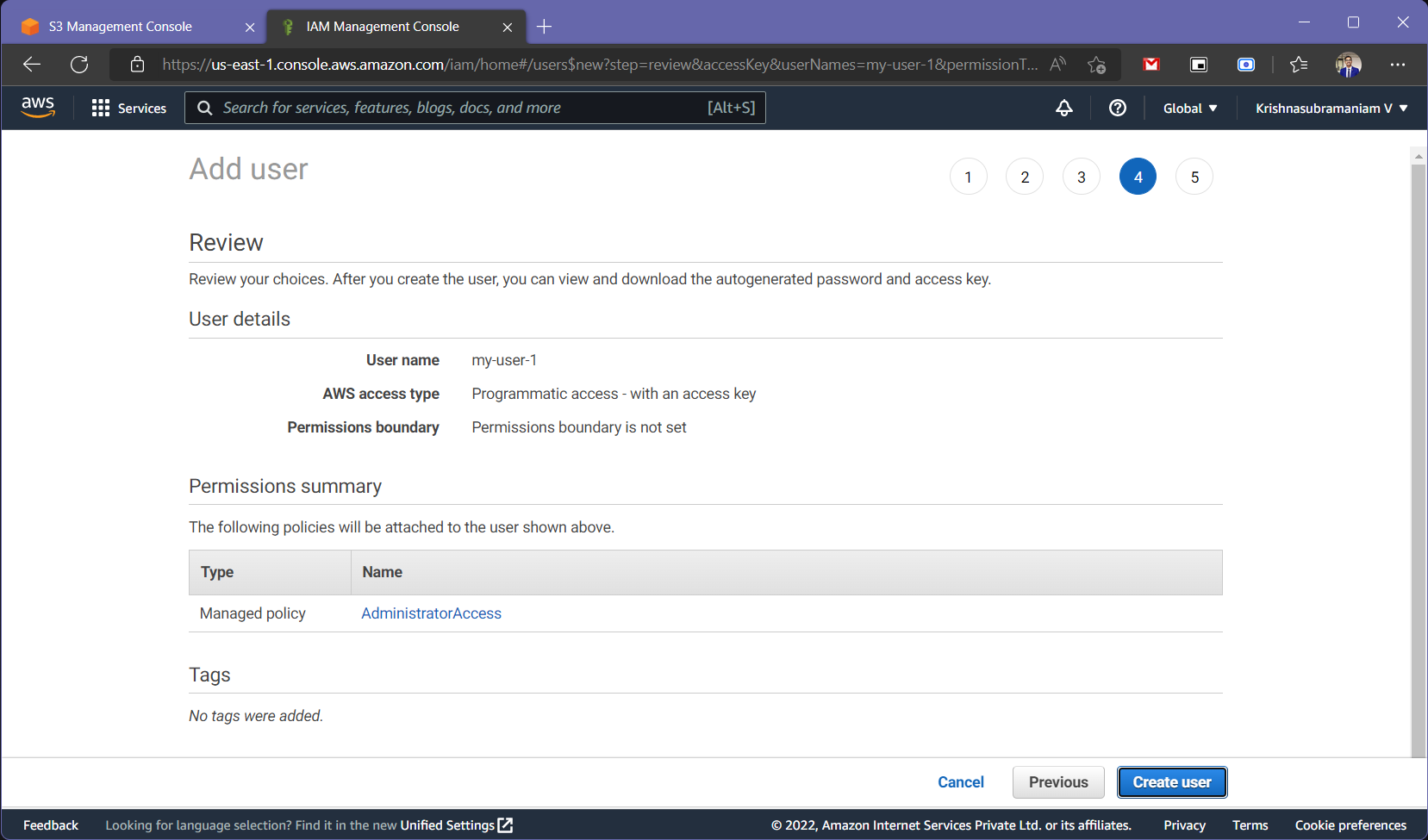
1. Create a new IAM user and go to the Users tab. Click on Add users and enter a username in the field.



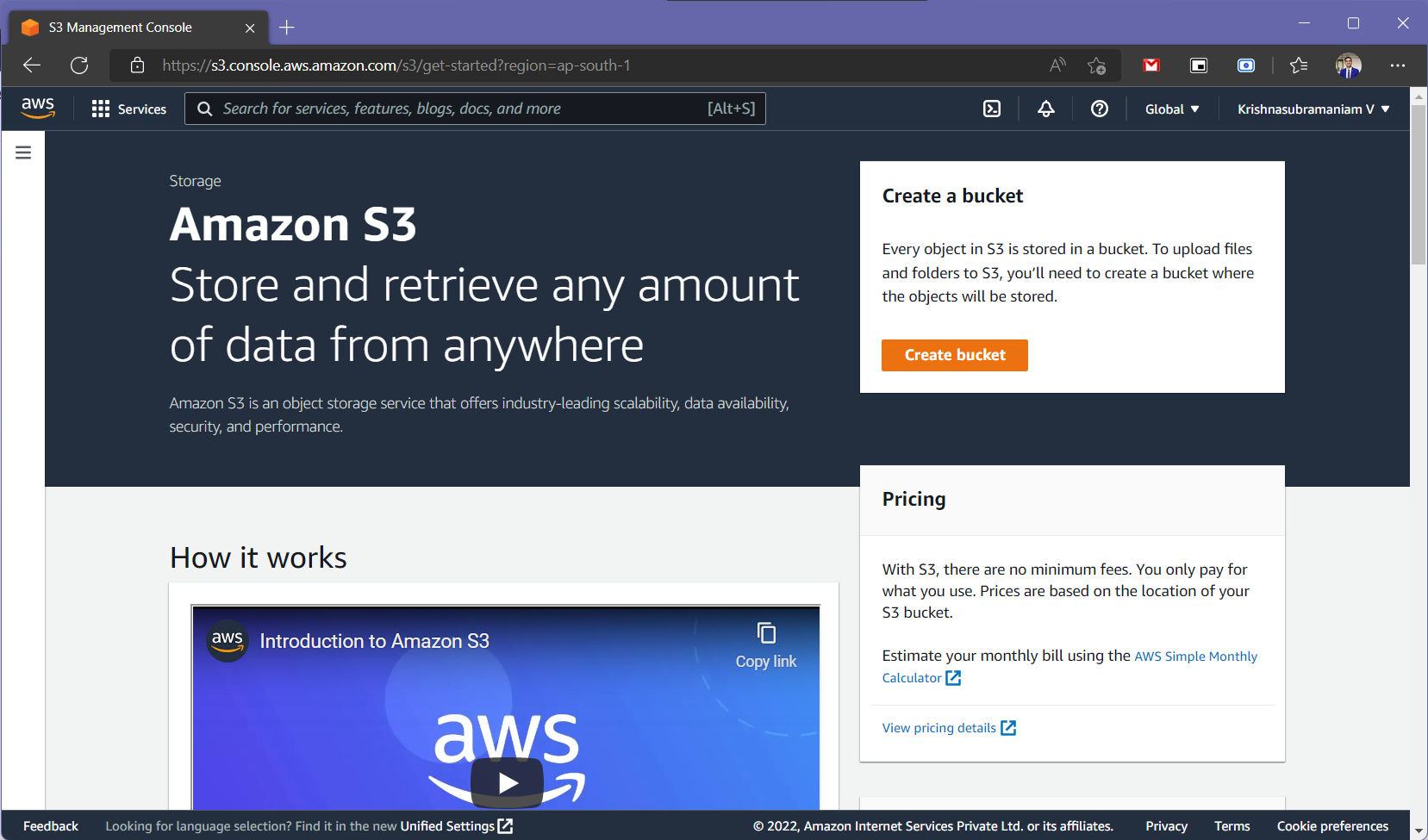
1. Tick the "Access key - Programmatic access field" (essential). Click "Next" and "Attach existing policies directly."



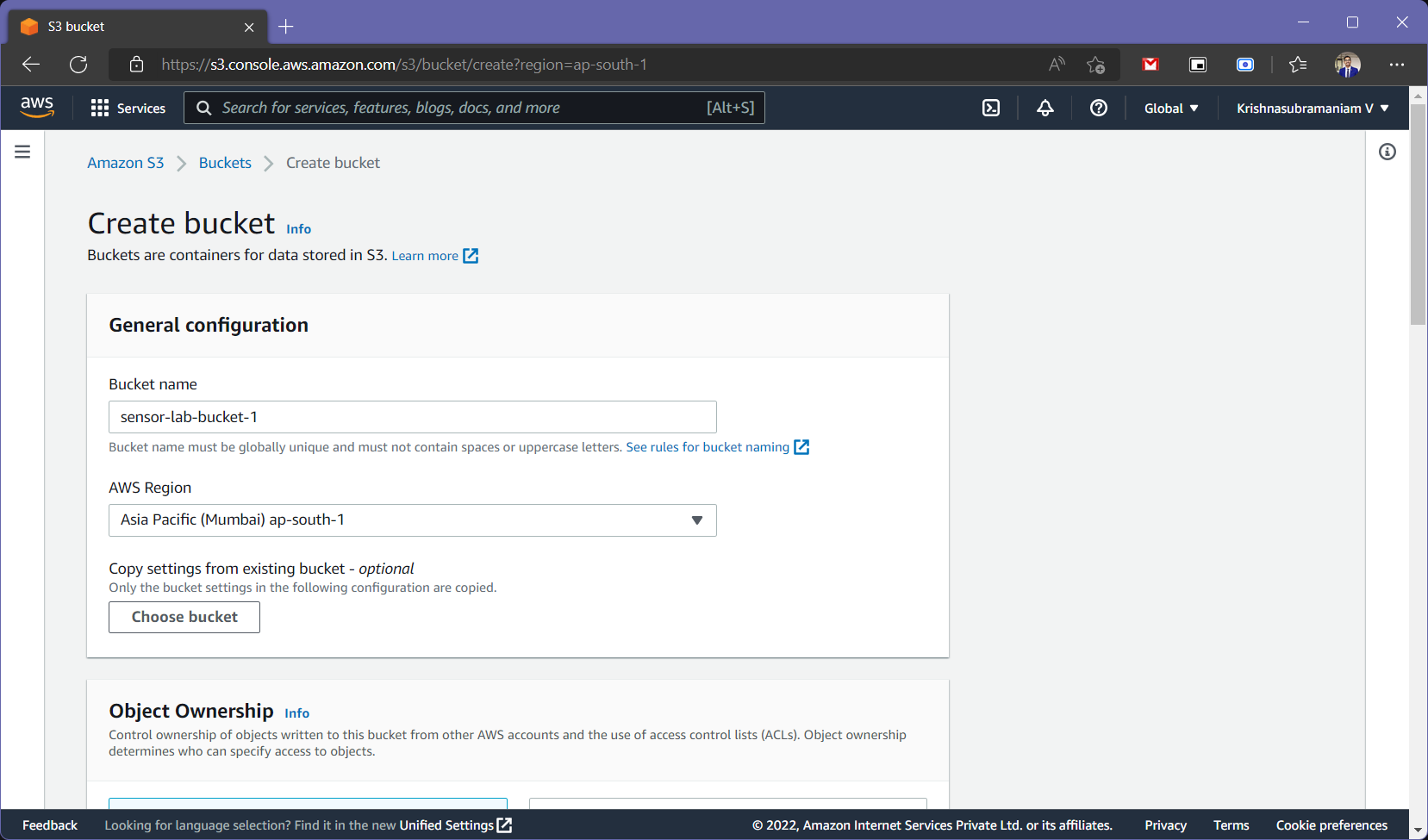
1. Tick the "AdministratorAccess" policy and click "Next" until you see the "Create user" button. Finally, download the given CSV file of your user's credentials.



1. Click services in the top left corner and scroll down to storage and select S3 from the right-hand list. Click "Create bucket" and give it a name.

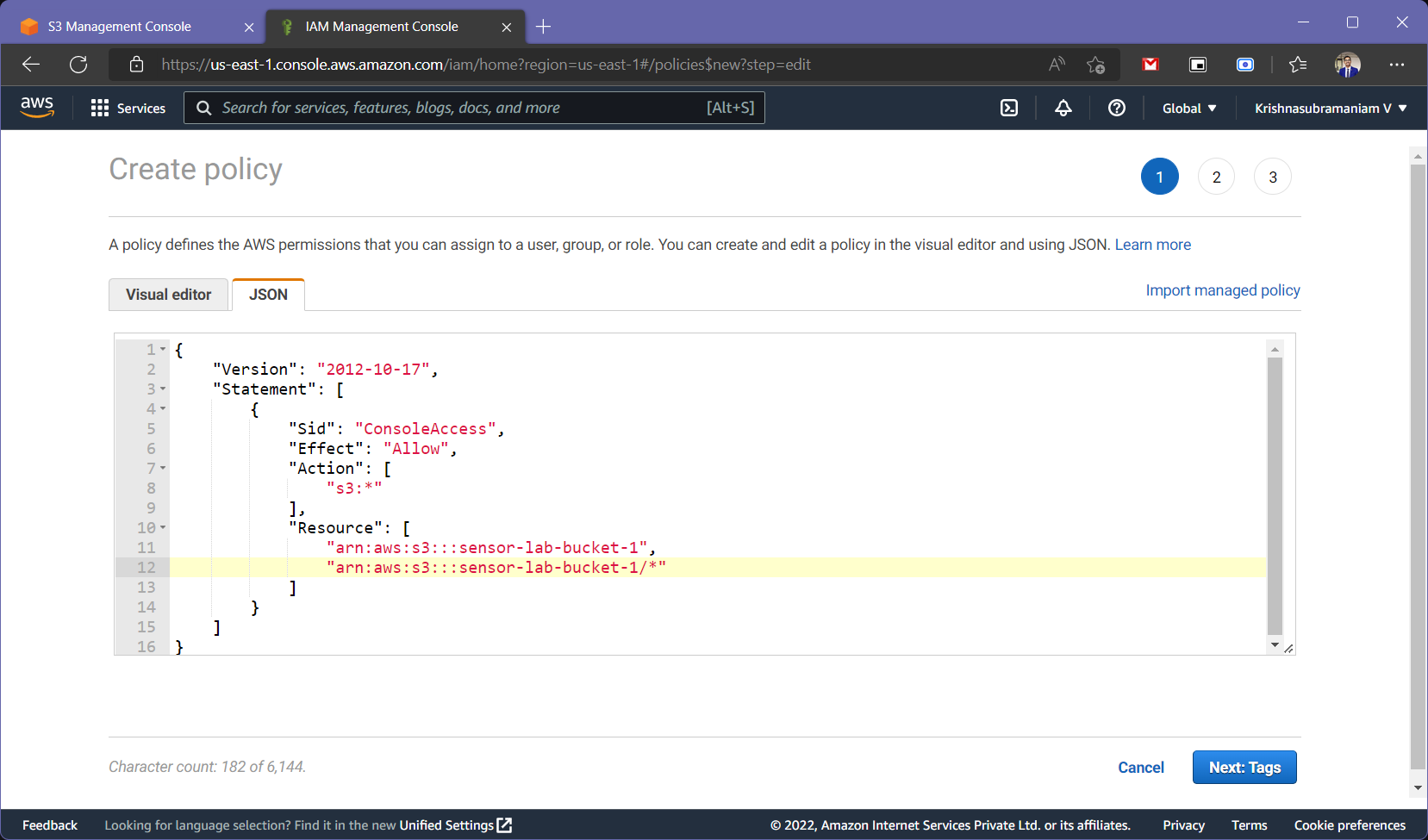


1. You can choose any region you want. Leave the rest of the settings and click "Create bucket" once more.

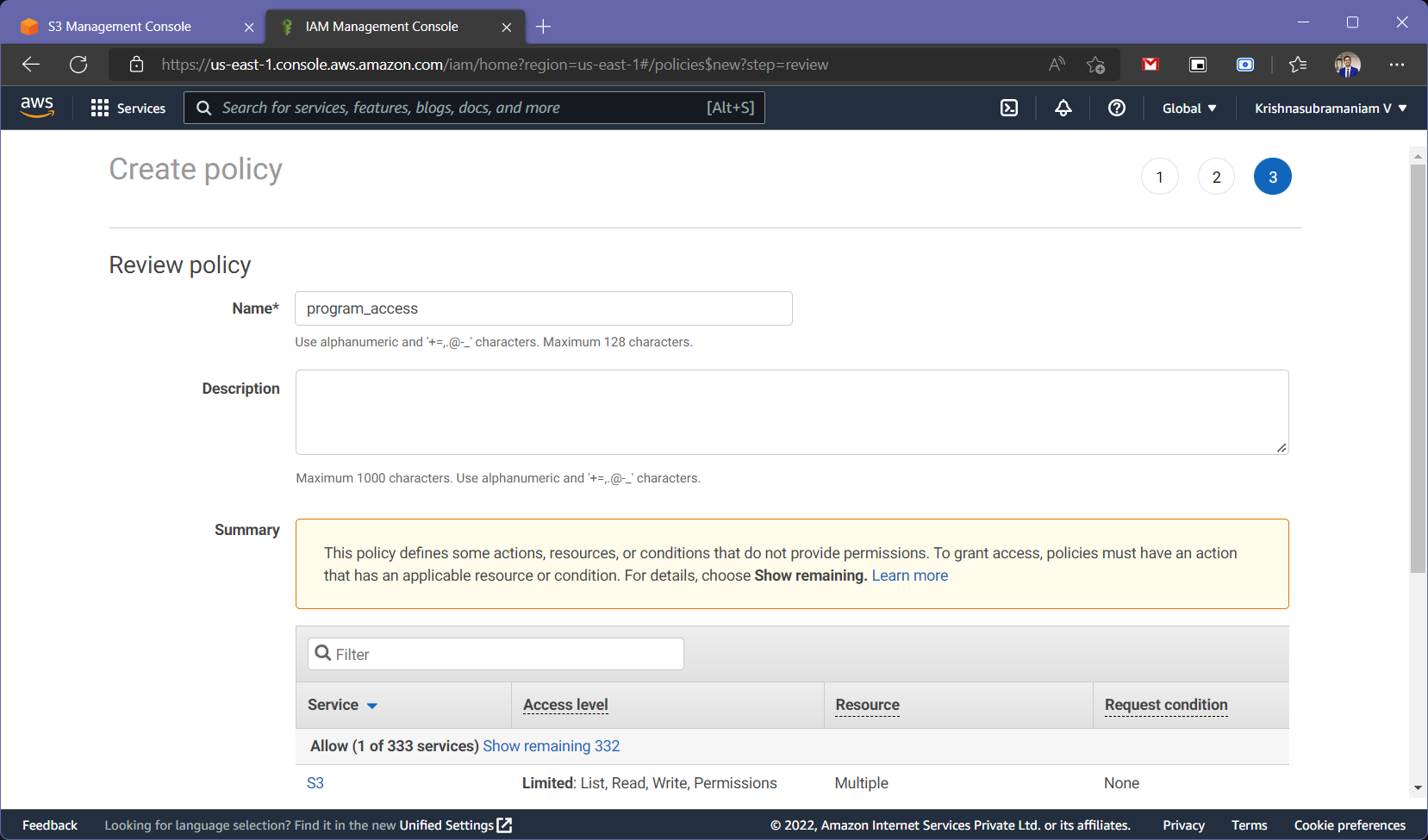


1. Go to the Policies tab and click "Create a policy." Click the "JSON" tab and insert the code below:

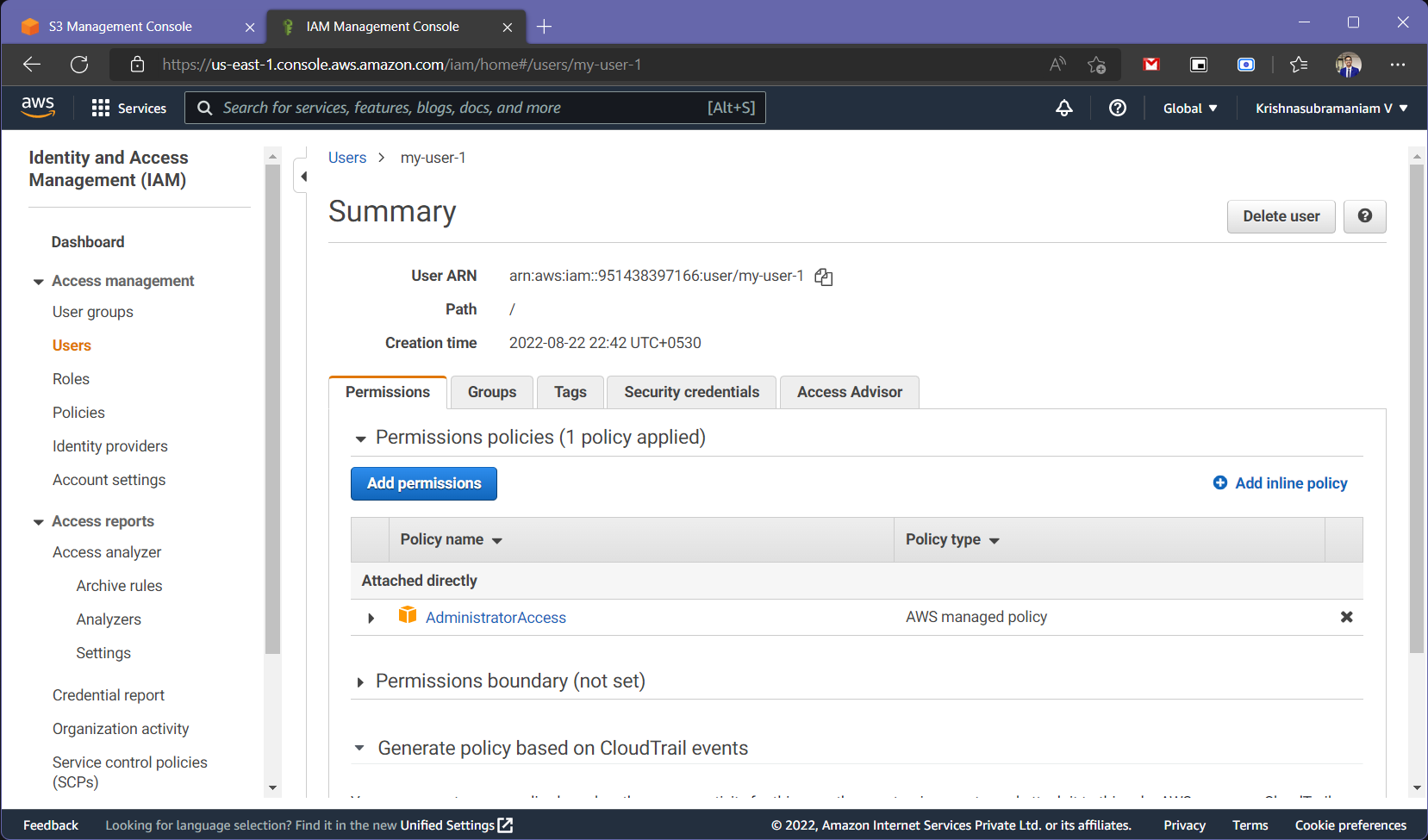
| {  "Version": "2012-10-17",  "Statement": [  {  "Sid": "ConsoleAccess",  "Effect": "Allow",  "Action": [  "s3:\*"  ],  "Resource": [  "arn:aws:s3:::sensor-lab-bucket-1",  "arn:aws:s3:::sensor-lab-bucket-1/\*"  ]  }  ] } |
| --- |



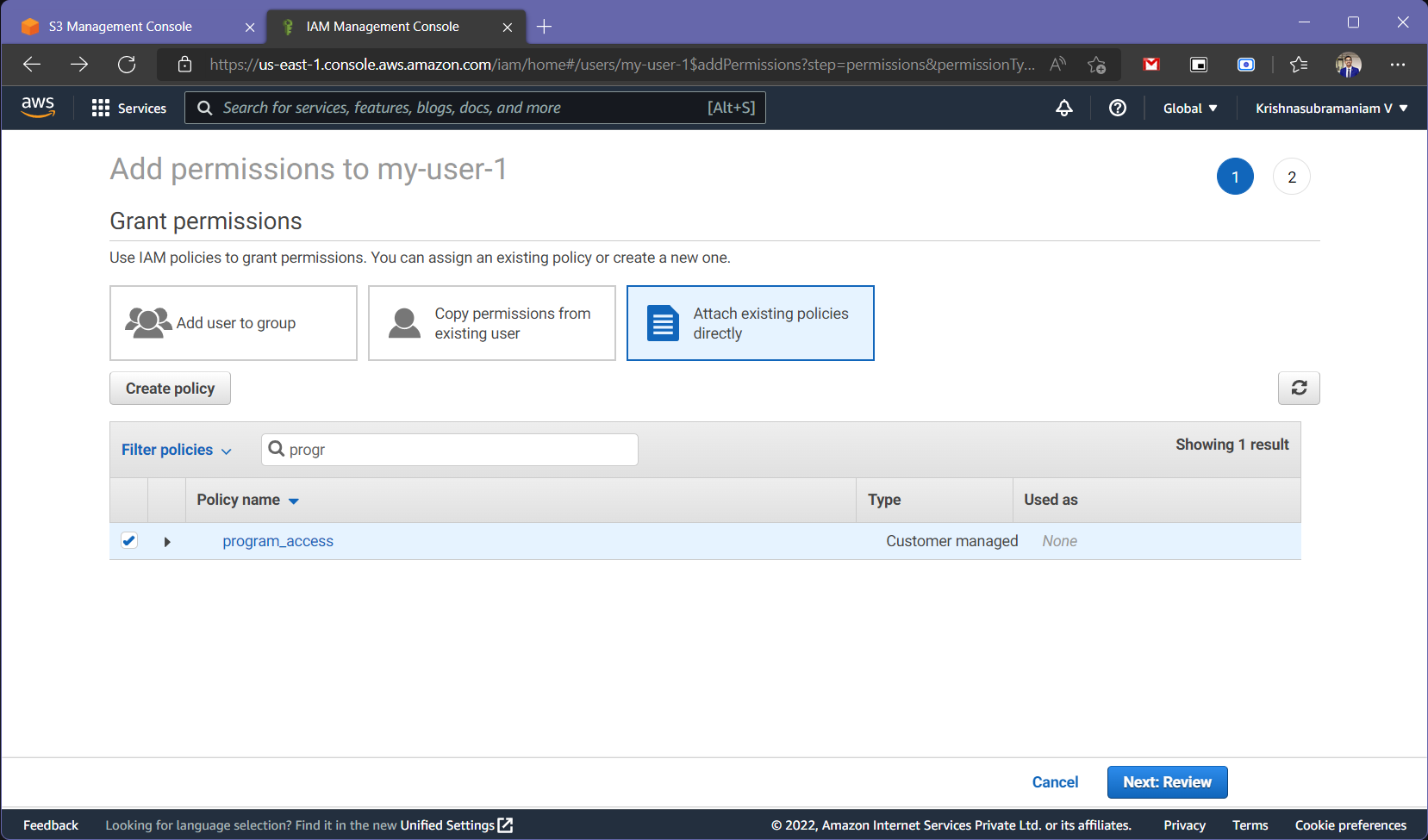
1. Now we are putting s3:\* to allow any interaction to our bucket. Click on Next->Next. Give the name to your policy and click Create.



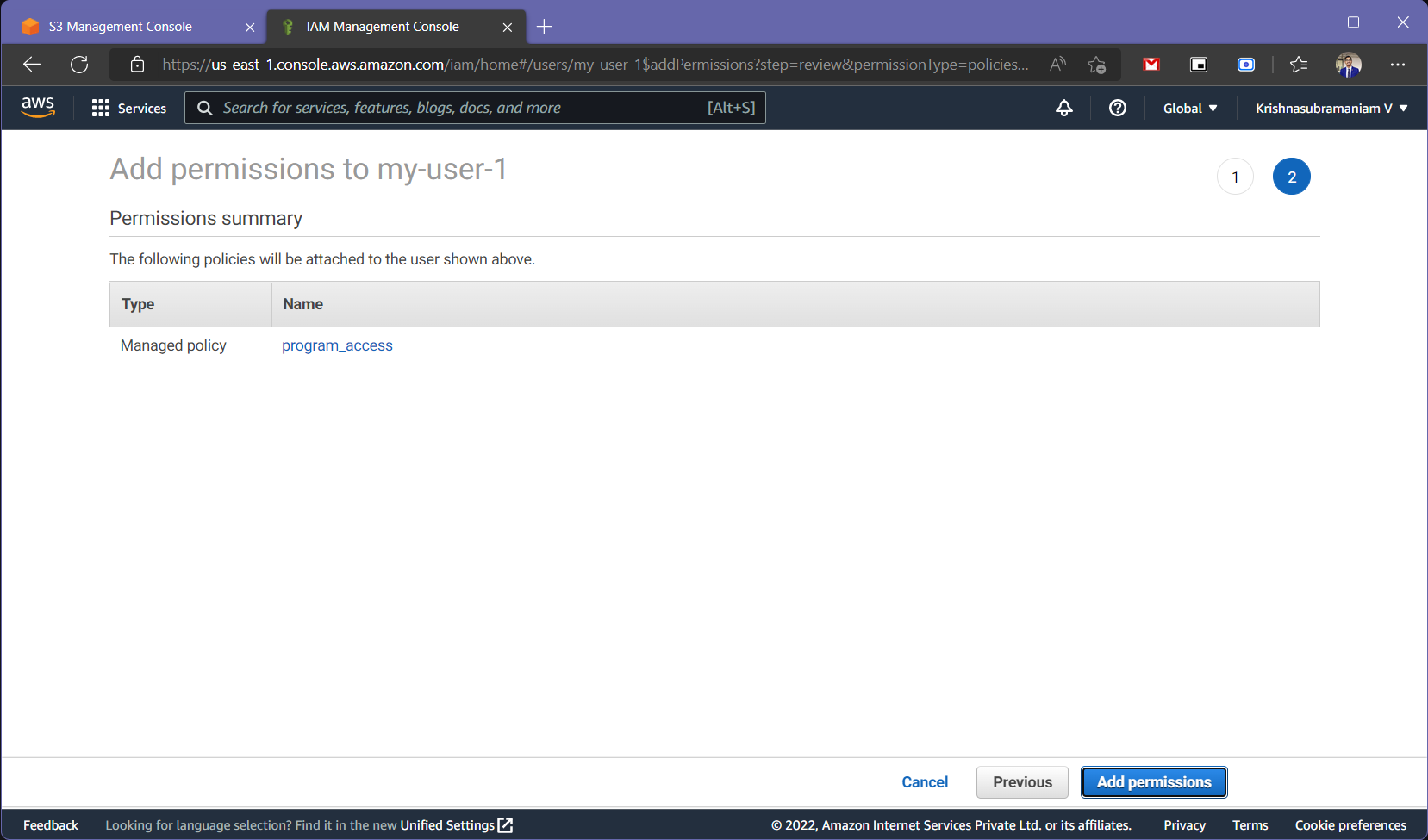
1. This policy is only attached to the bucket, and we should connect it to the user as well so that your API credentials work correctly. Go to the Users tab and click on the user we created in the last section.



1. Click the "Add permissions" button. Click the "Attach existing policies" tab and filter them by the policy we just created.



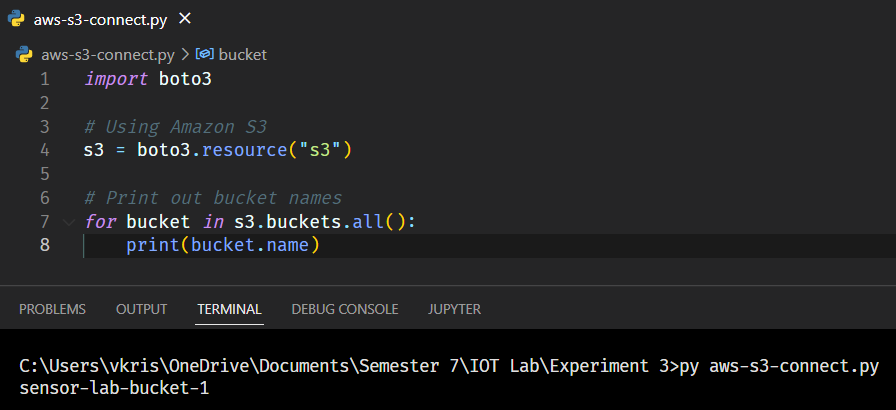
1. Tick the policy, review it and click "Add" the final time.



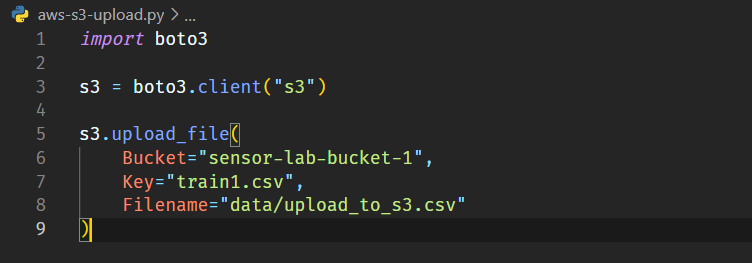
**Read from the things (channel) and display it on the console.**

1. Download AWS Console

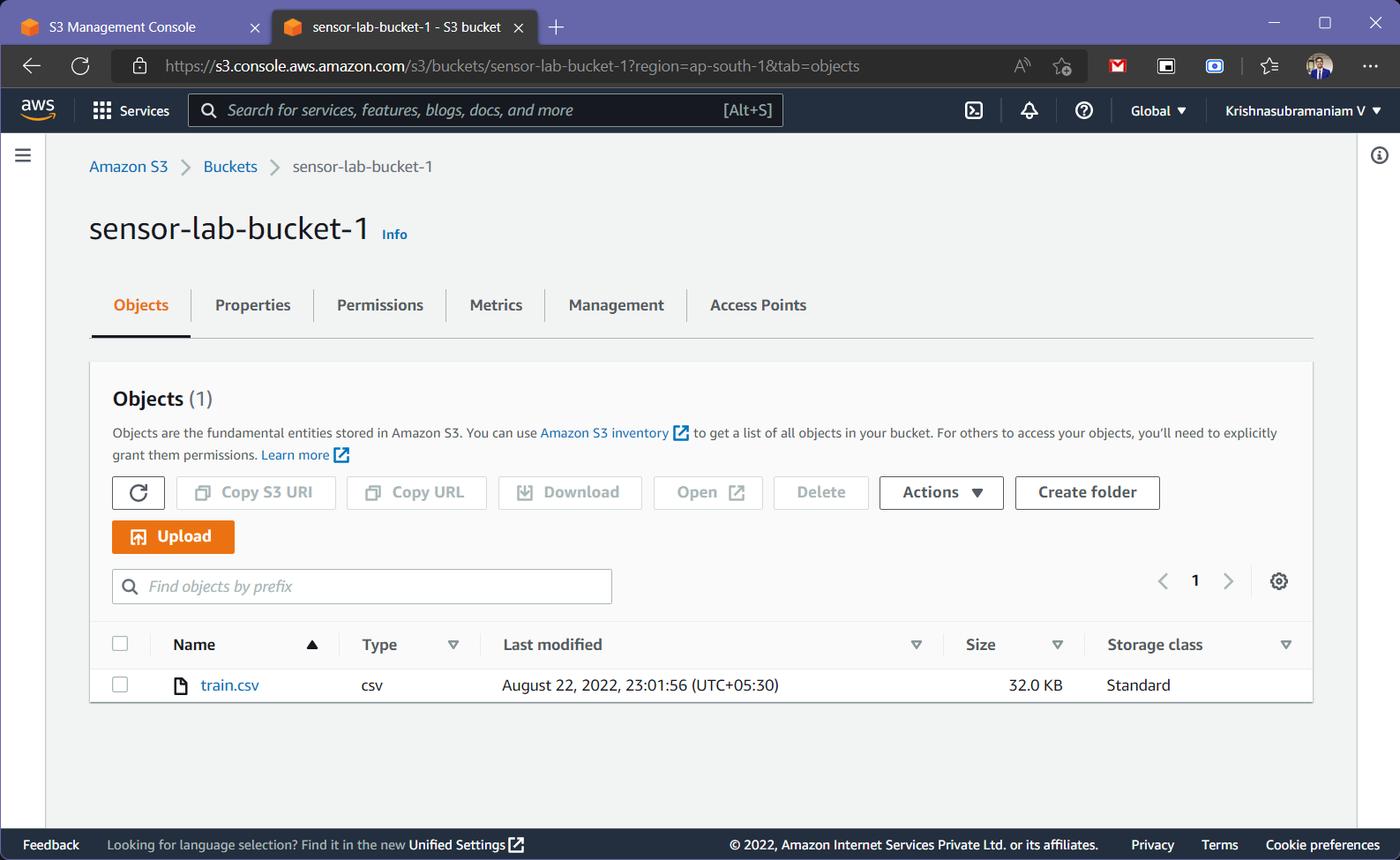
Testing connection:



1. Uploading file:

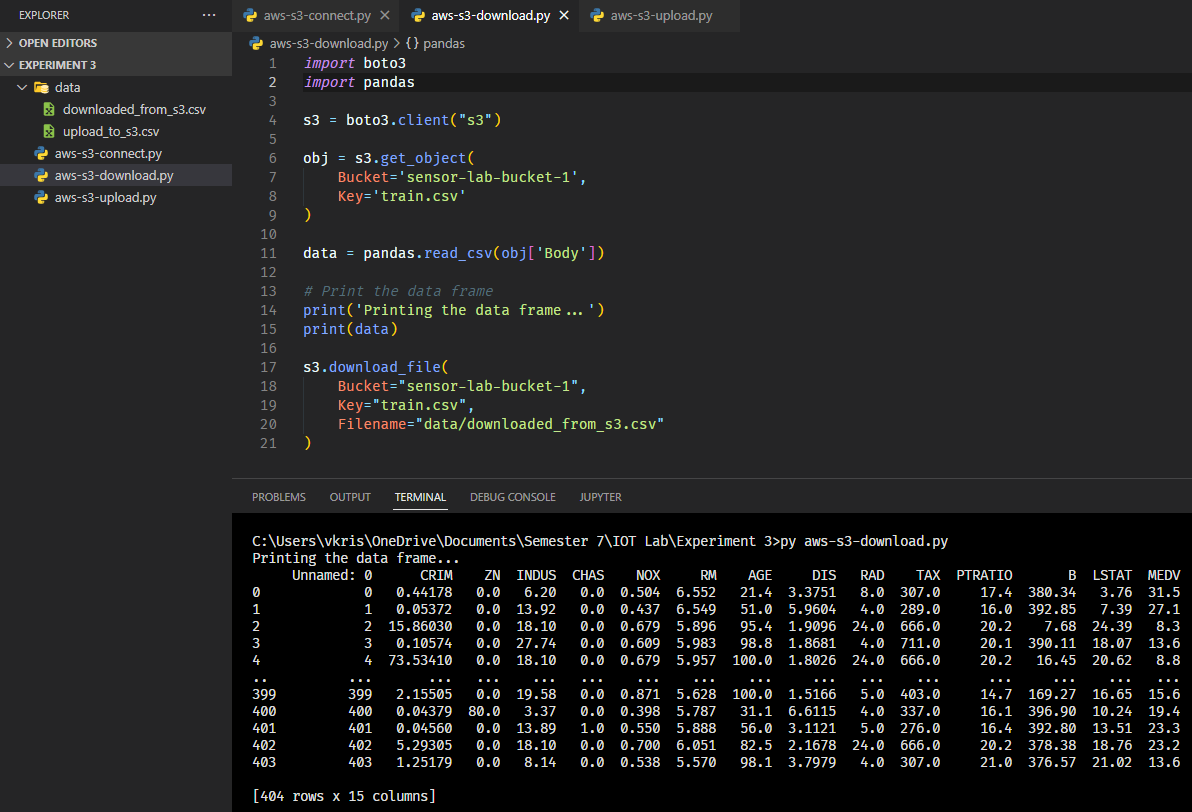


1. Check the uploaded file and display it on the console.



**Read from the things (channel) and save the data into the file in a proper format.**

1. Downloading the file and saving it into a proper format.



This is how you can send or receive data through a program, display it on the console and save the data into the file in a proper format.

**Conclusion**: Hence, we understood how to transfer data using MQTT/REST/CoAP protocol, send or receive data through a program, display it on the console and save the data into the file in a proper format.