Cloud Computing Simulations and IoT Integration

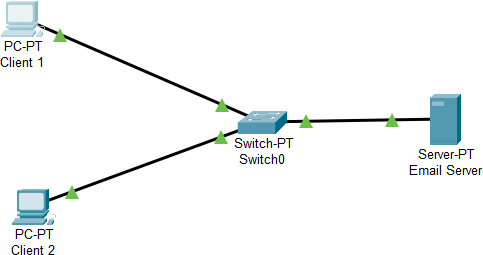
Designed and implemented network simulations for email, web, and FTP communications, ensuring seamless data transmission between clients and servers.

* Configured and demonstrated Email, Web, and FTP server-client models using networking principles to facilitate real-time data exchange.
* Developed and deployed an IoT Hub on Microsoft Azure, enabling bi-directional communication between cloud and connected devices.
* Implemented Device-to-Cloud (D2C) and Cloud-to-Device (C2D) messaging using Azure IoT Hub and VS Code IoT Toolkit for real-time monitoring and automation.
* Utilized DNS configuration, HTTP setup, and FTP transfer protocols to simulate real-world cloud computing environments.

**Experiment -1** Simulating the communication using e-mail applications

1. **Create a topology with two e-mail clients and email server**

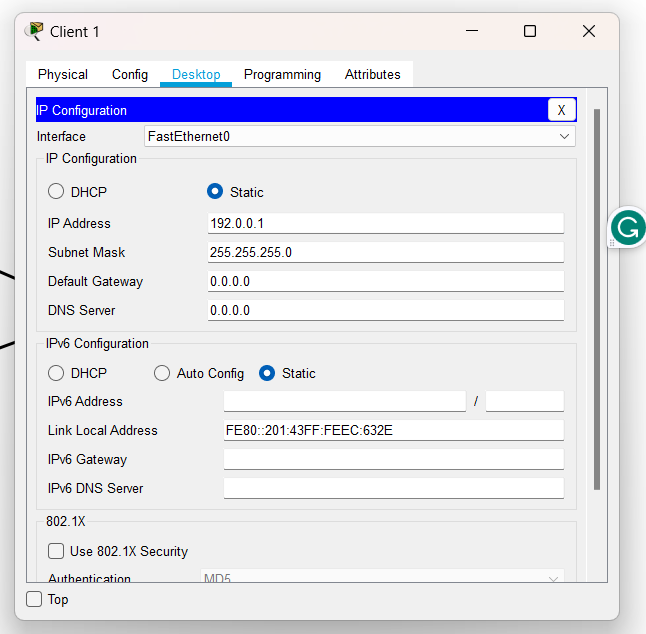
2 Email Clients and 1 Email Server are connected to the switch.

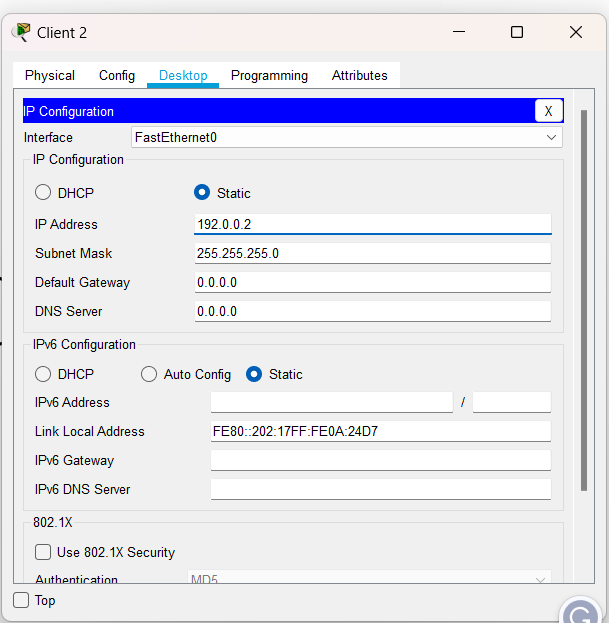


1. **Configure email clients and email server (use your reg. number)**

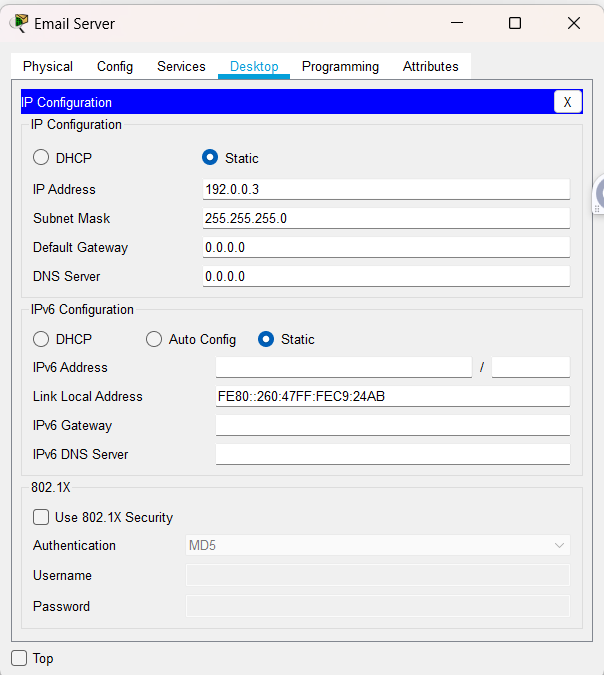
Configuring Client 1 and Client 2 System and the Email Server

Client System Configuration:

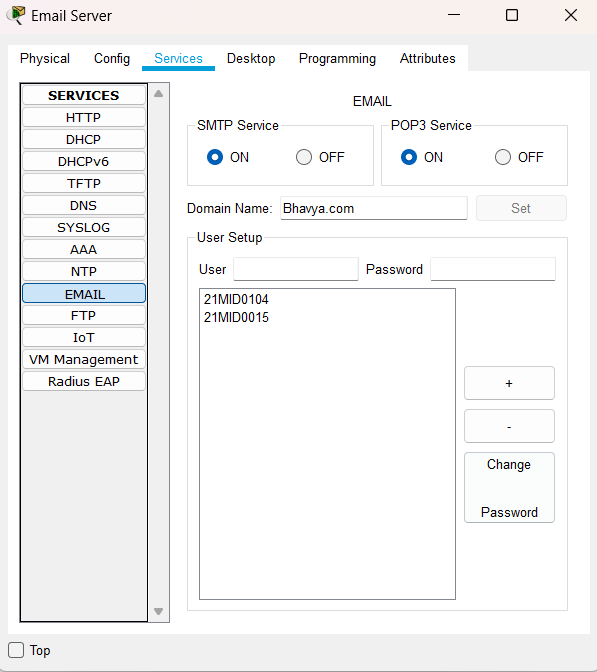


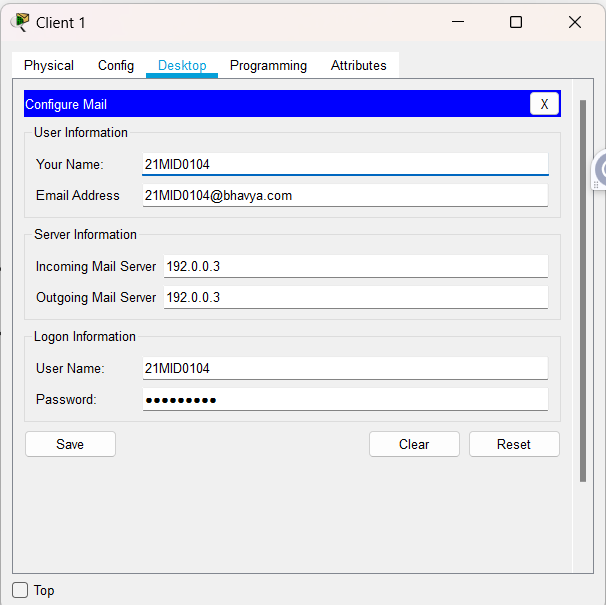


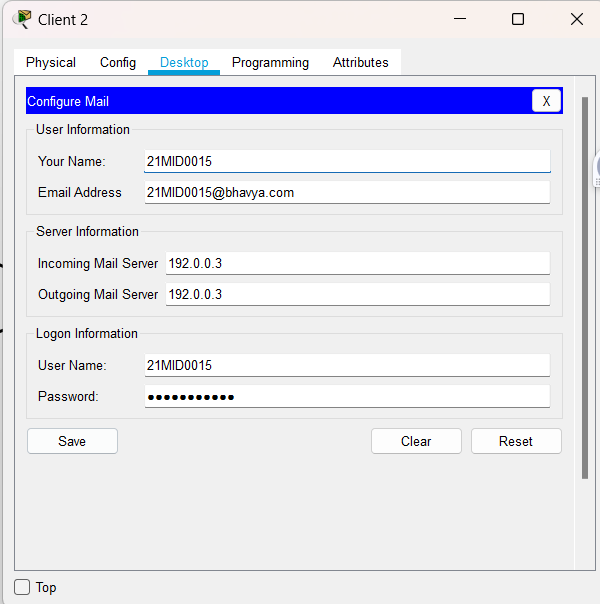
Server Configuration:



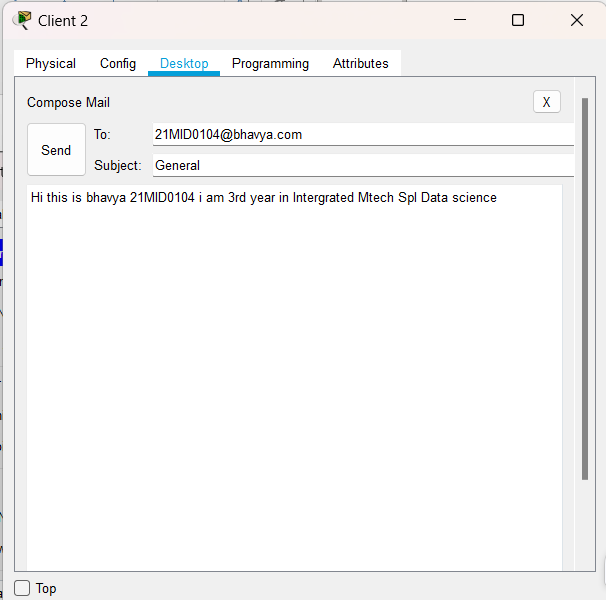
Email Configuration SetUp:

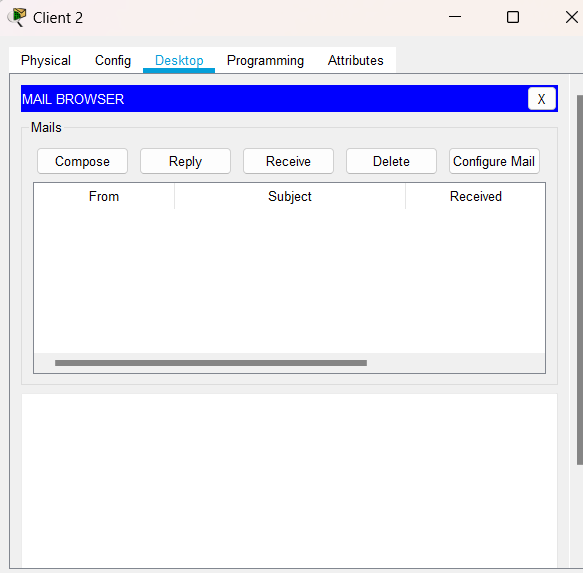




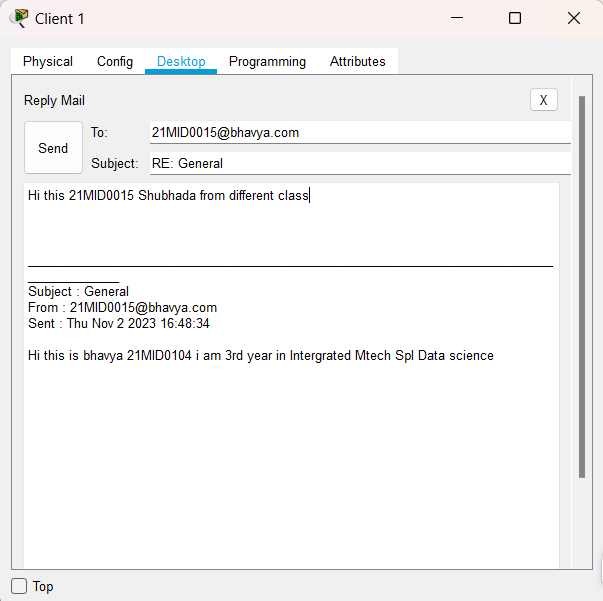
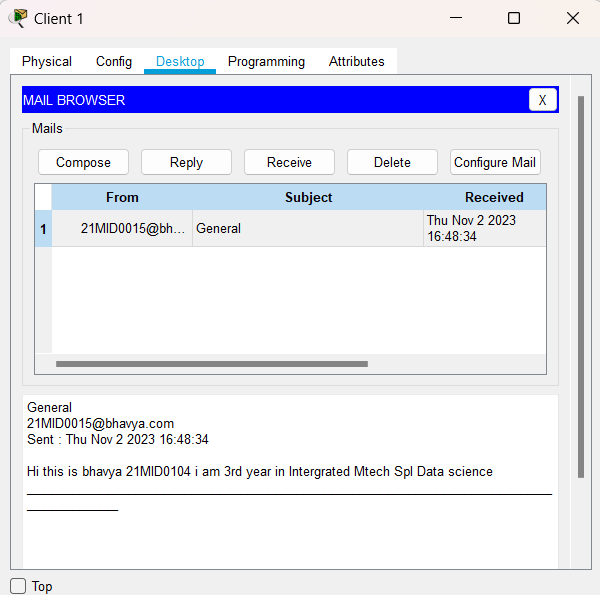


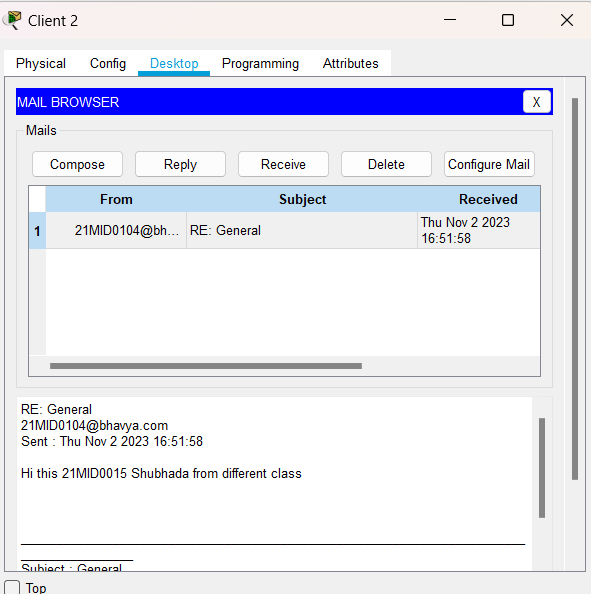
1. **Demonstrate for sending and receiving the mails**. Sending Mail from client 2 to client 1:





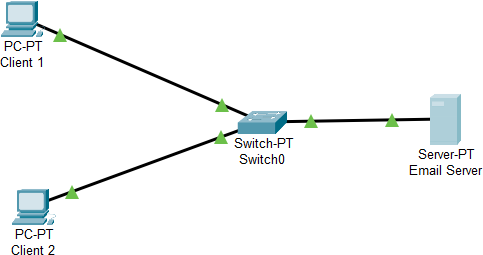
Replying to the mail:





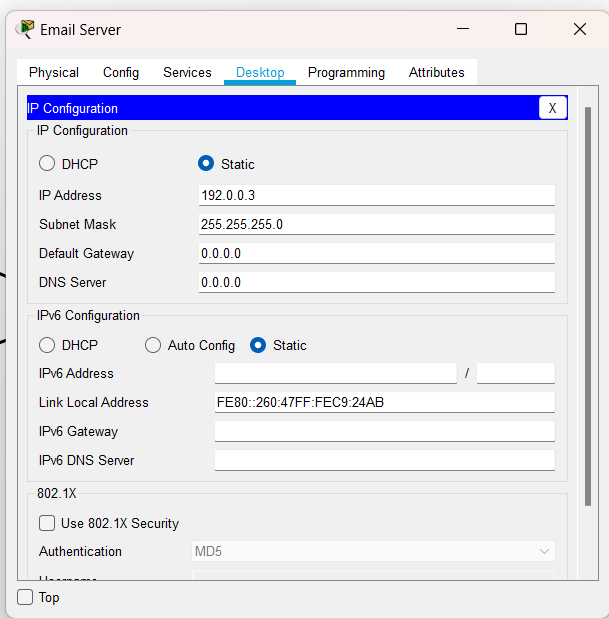
**Experiment -2 Simulating the communication using web applications**

1. **Create a topology with two web clients and web server** Connecting 2 Web Clients and 1 Web Server with the switch which is to be in a single LAN

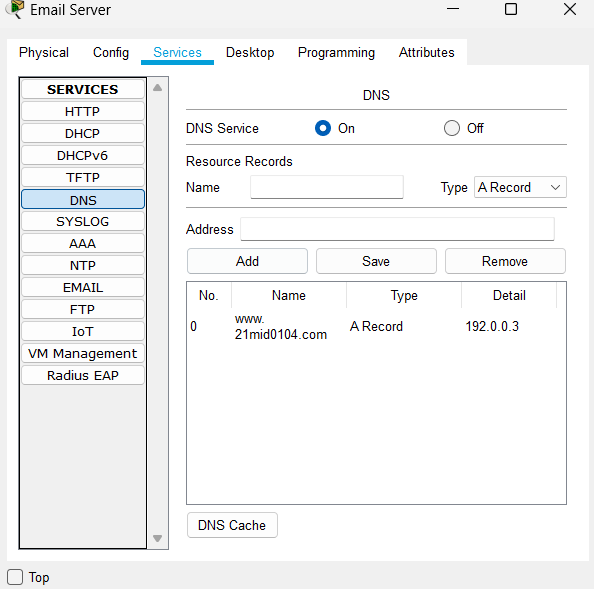


1. **Configure web clients and web server and DNS server**

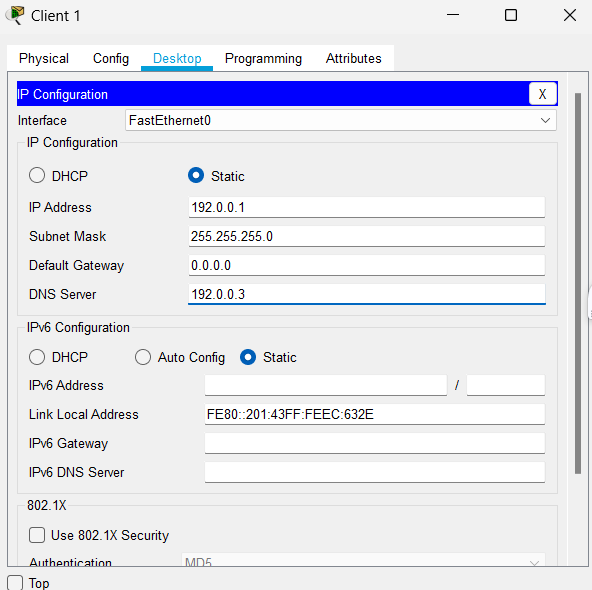
Server Configuration:

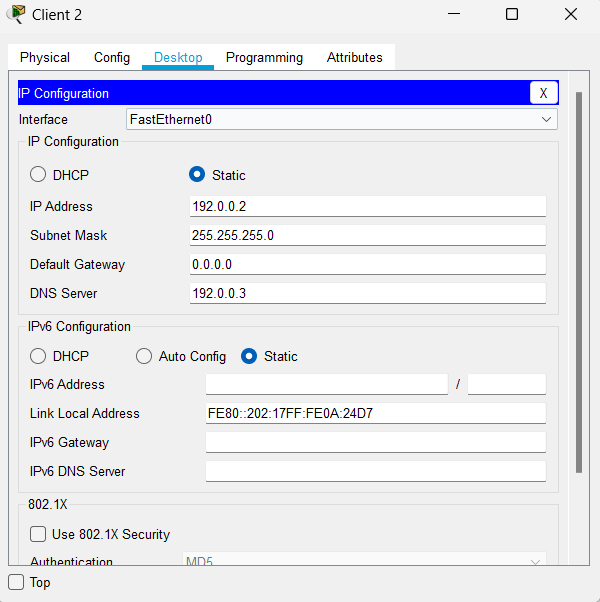


DNS Configuration of Server:

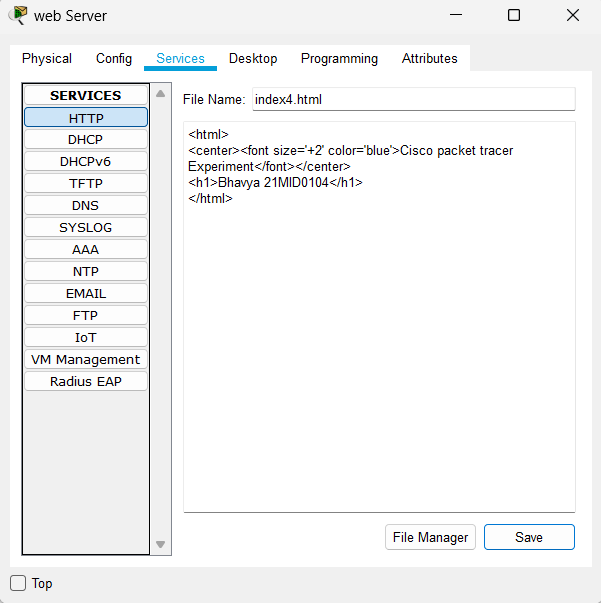


Client System Configuration:

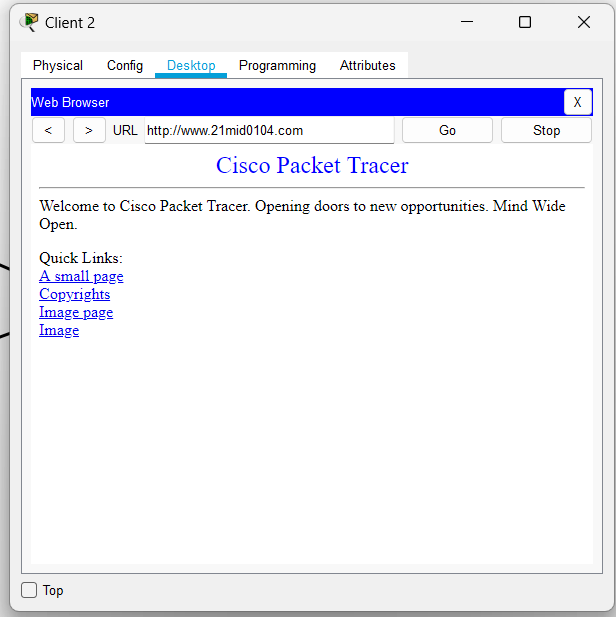
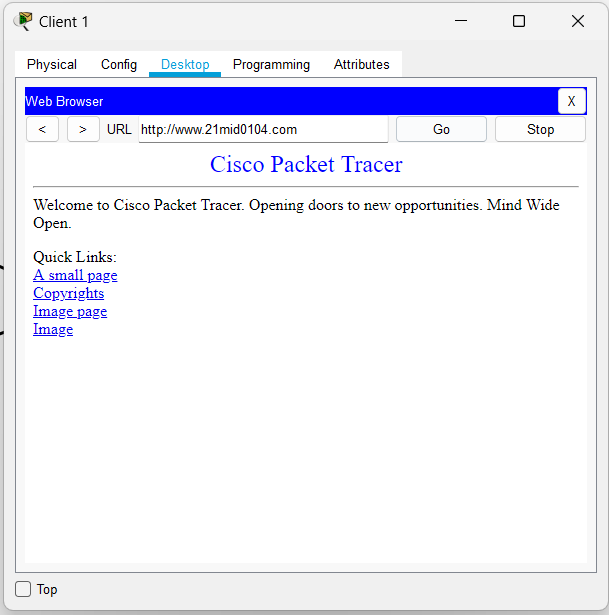




Server HTTP Configuration:

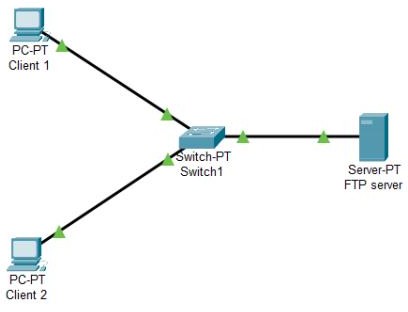


1. **Demonstrate for giving the web request. Ensure that the page should display your register number.**

****

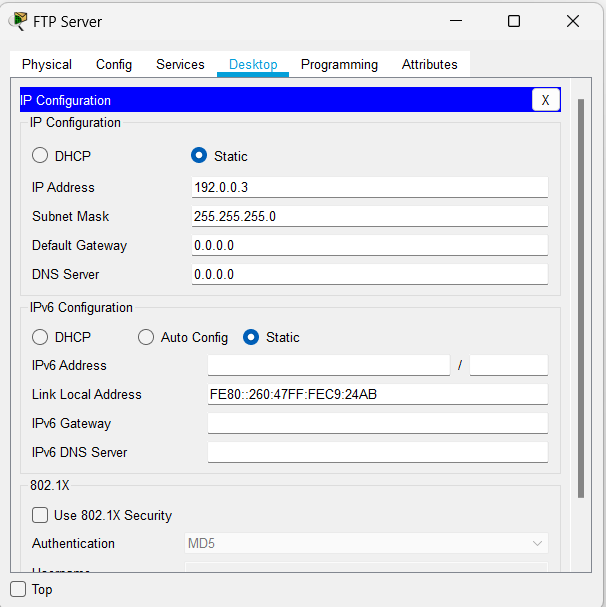
**Experiment -3 Simulating the communication using FTP applications**

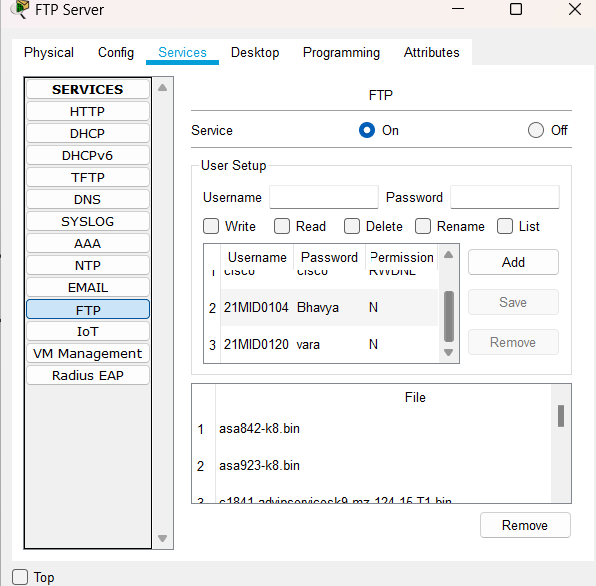
1. Create a topology with two FTP clients and FTP server Connecting 2 FTP Client and 1 FTP Server with the Switch.



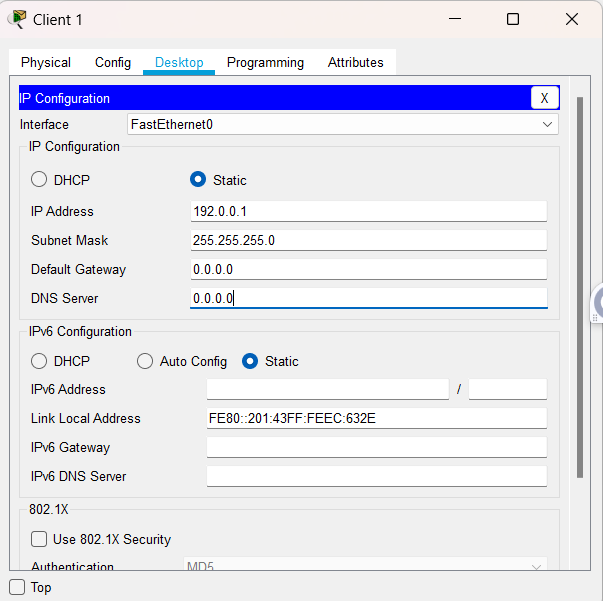
1. **Configure FTP clients and FTP server**

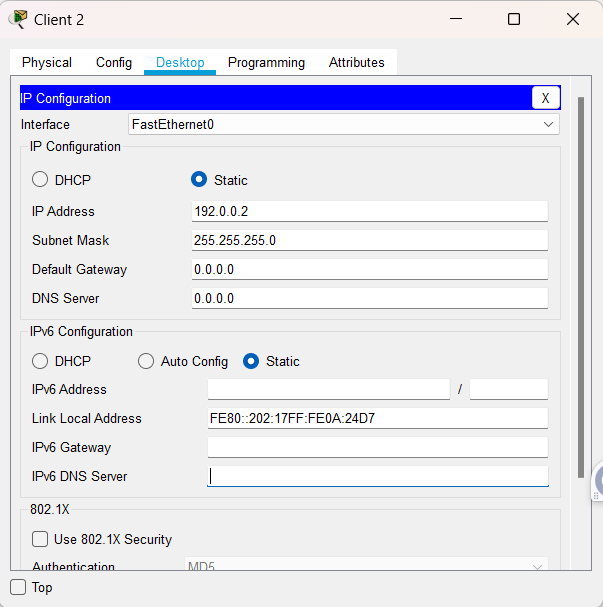
FTP Server Configuration:





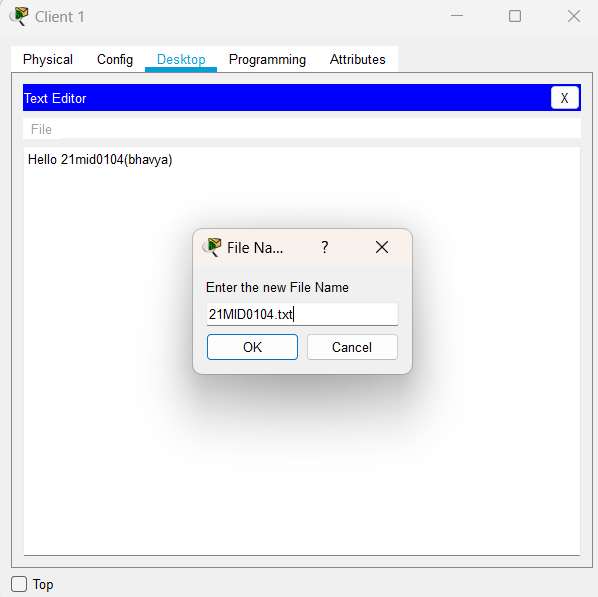
Client System Configuration:



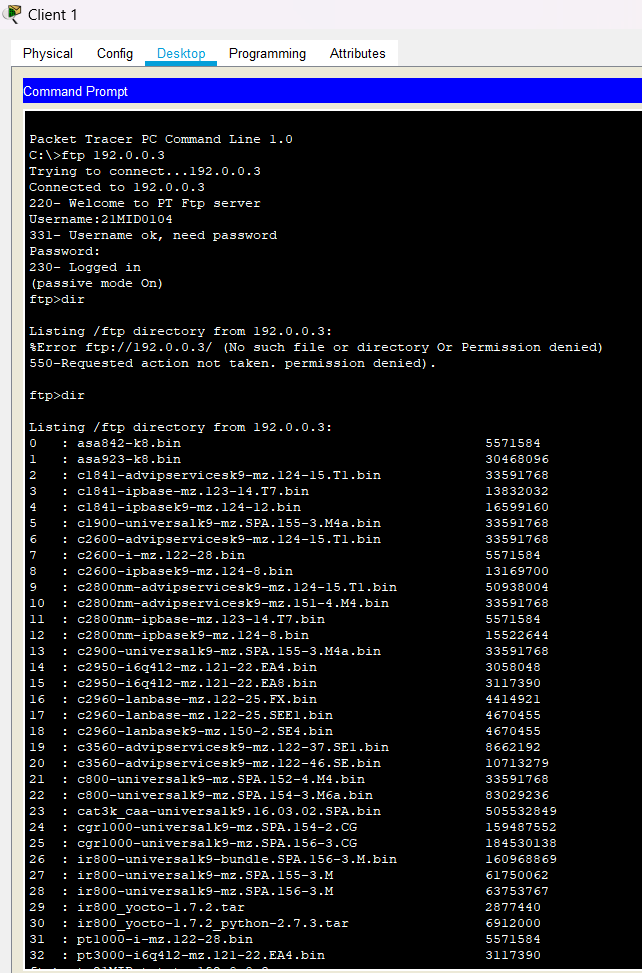


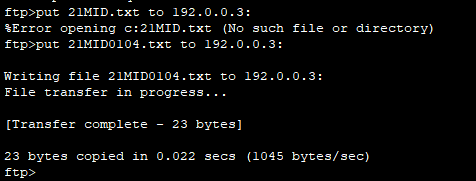
1. **Demonstrate for sending and receiving the file. Ensure that the filename with your register number.**

**Creating File in Client1:**

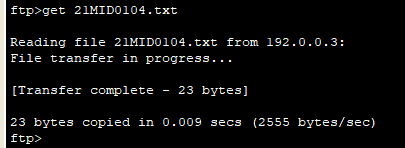
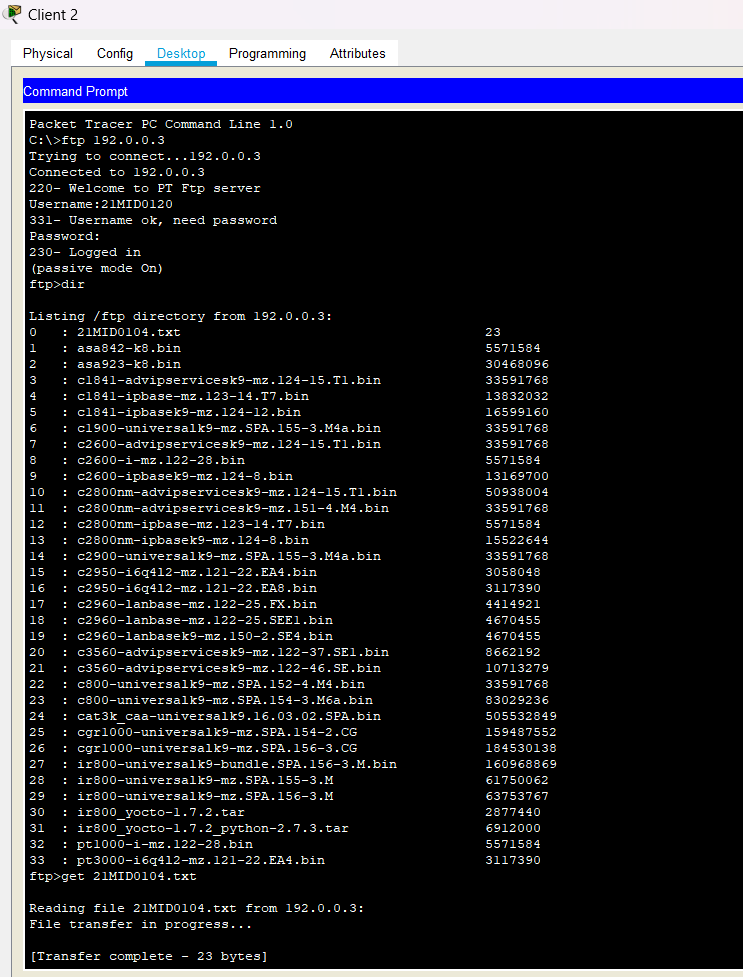
****

**Sending File to the Server:**

****



**Receiving File from the Server:**

****

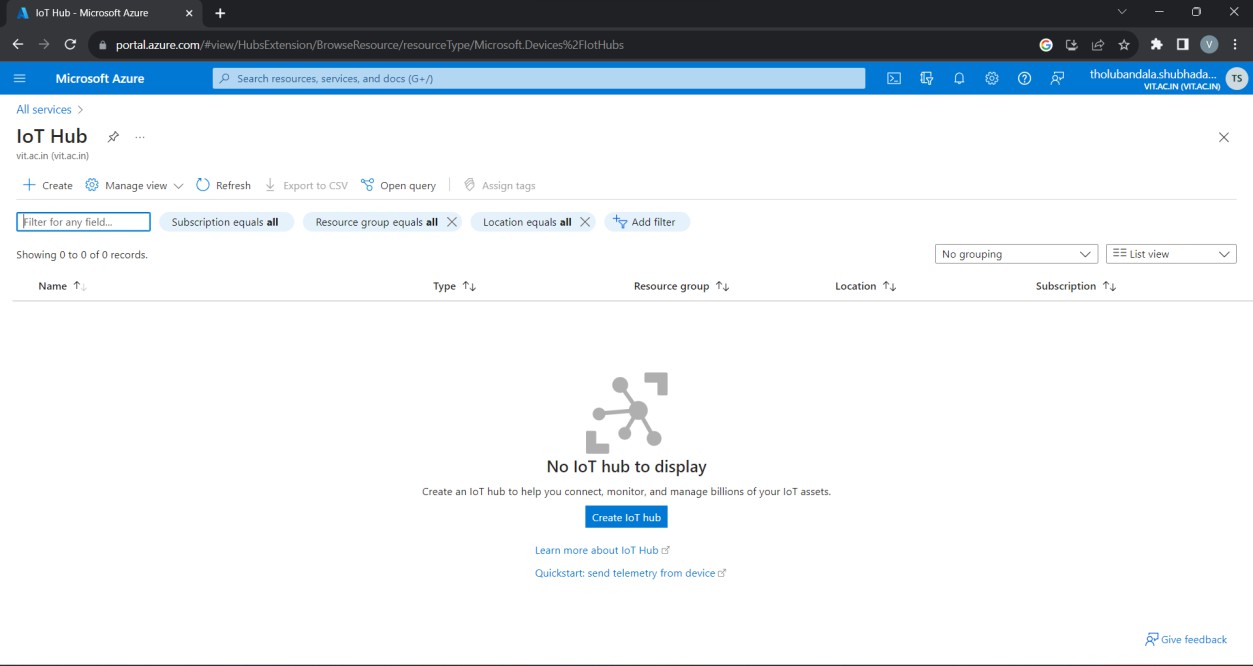
**IoT hub through Azure and establish a Bi-directional communication**

**Aim:** To build the IoT hub environment in the cloud and deploy Bidirectional Communication with the IoT Hub in the Cloud.

**Procedure:**

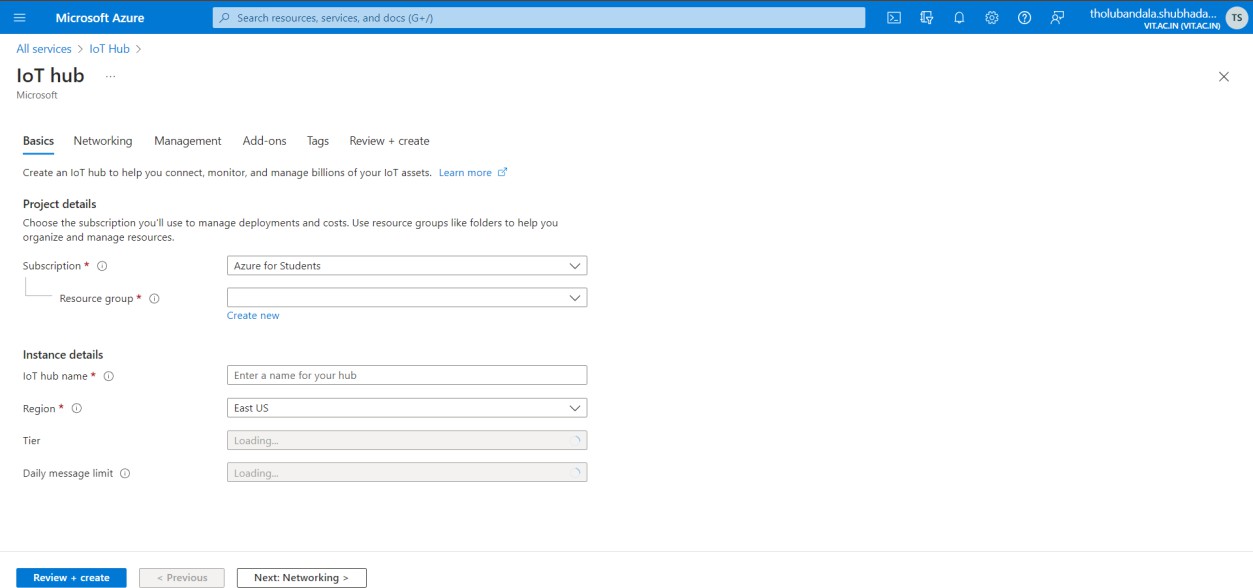
**Section I:** Creating an IoT hub

**Step 1:** Search for IoT hub and there it is IoT hub for Microsoft. Let's go ahead and create this.

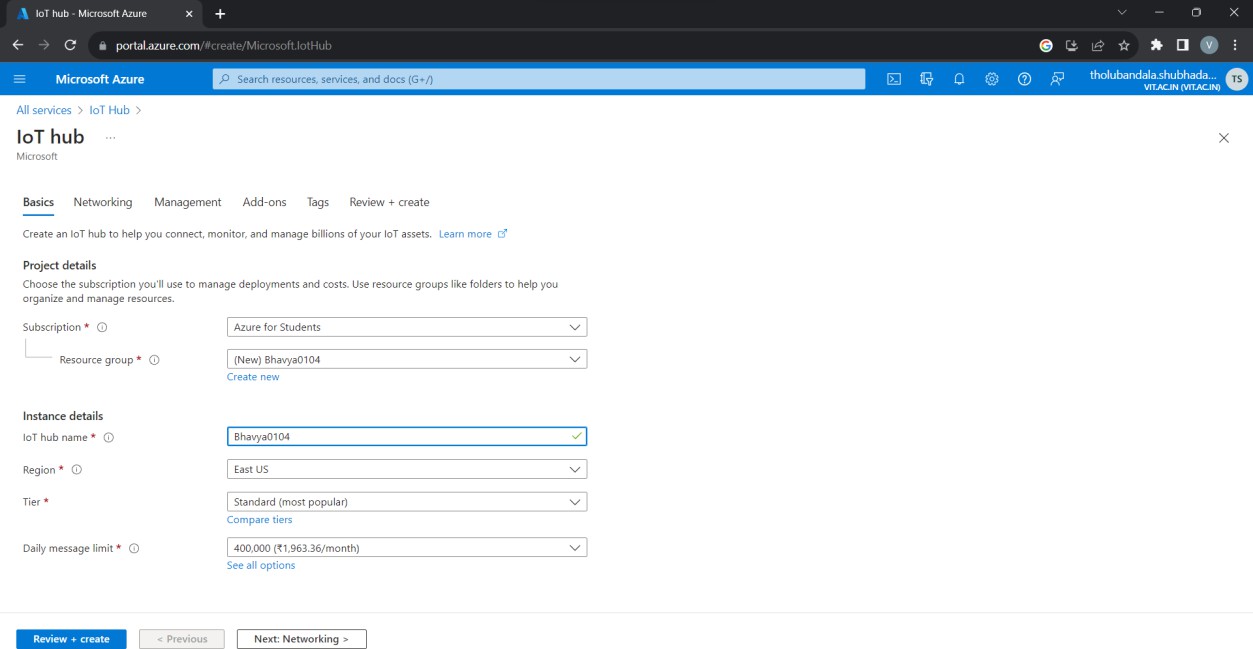


**Step 2:** There are several options we can configure for our new hub. Once you’ve done this, move on to the size and scale options. Here we can configure the pricing and scale tier.

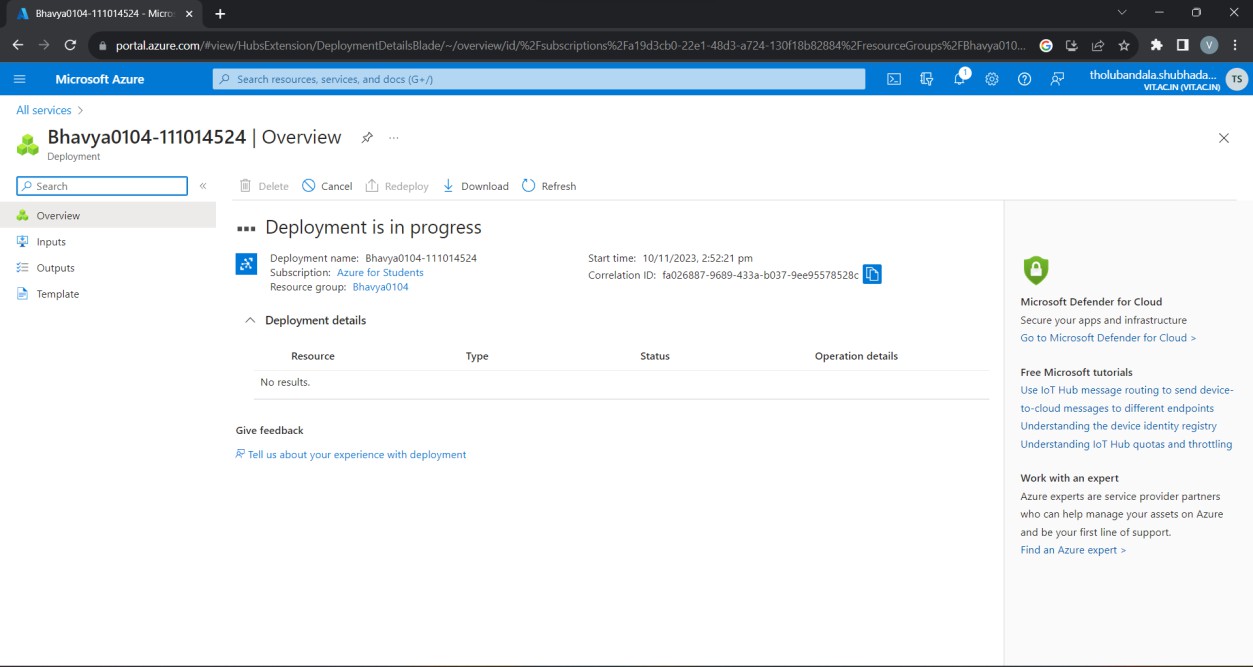
**Step: 2.1:** Create a Resource Group by giving a name in the combination of your number and name with the prefix of rg.



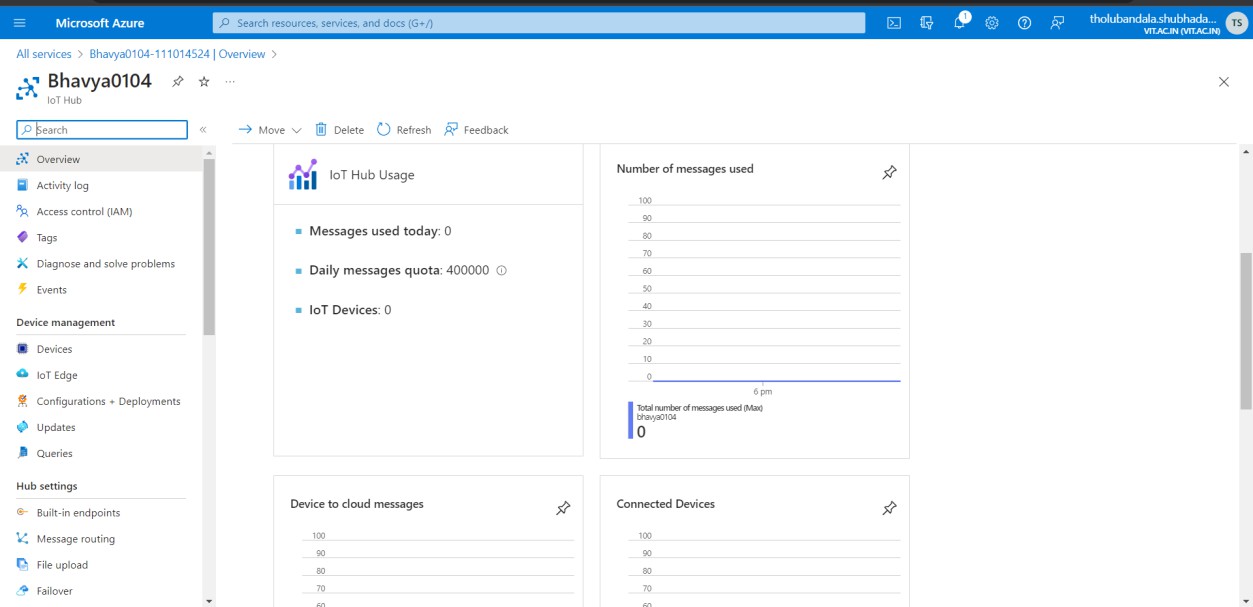
**Step 2.2** Give a name for the IoT hub in the combination of your number and name.



**Step 3** Click on Review + Create. Then you will get a following window. It takes some time to complete the deployment.



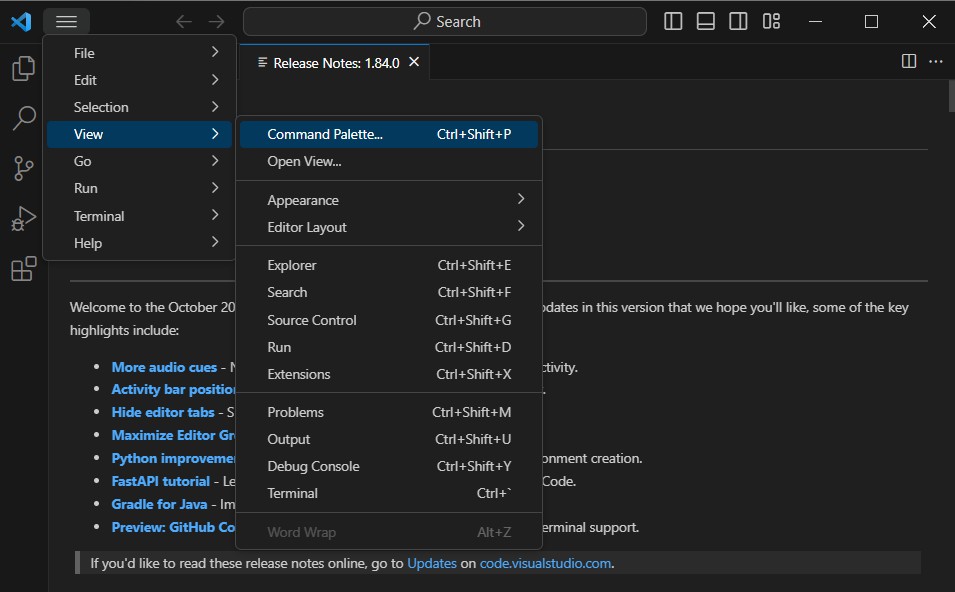
**Step 4:** Click on Go to the resource. The summary view shows us some basic stats about our hub, and if we scroll down, we can see how many messages we've sent, as well as how many devices we have while we're here

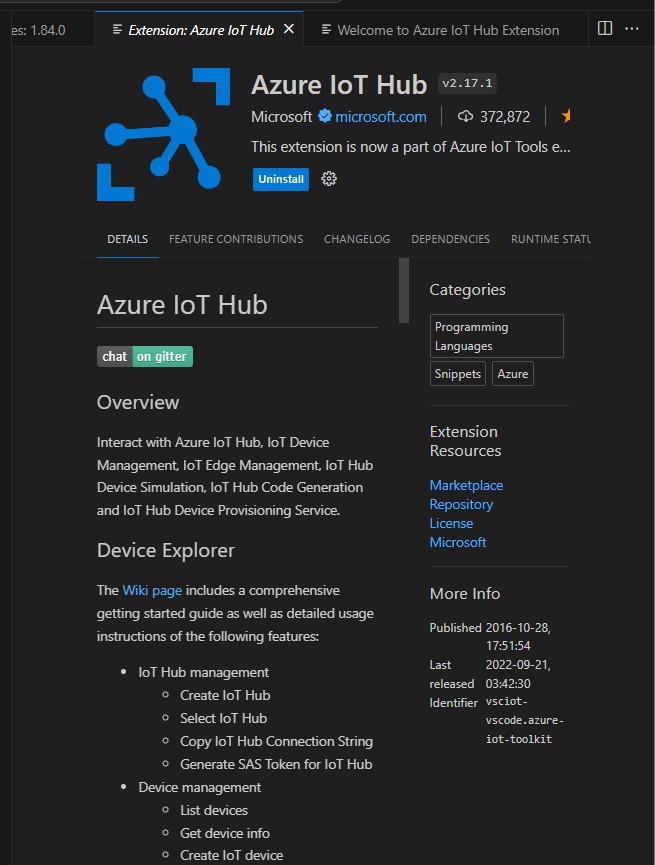


**Section II:** Creating a device (VS Code)

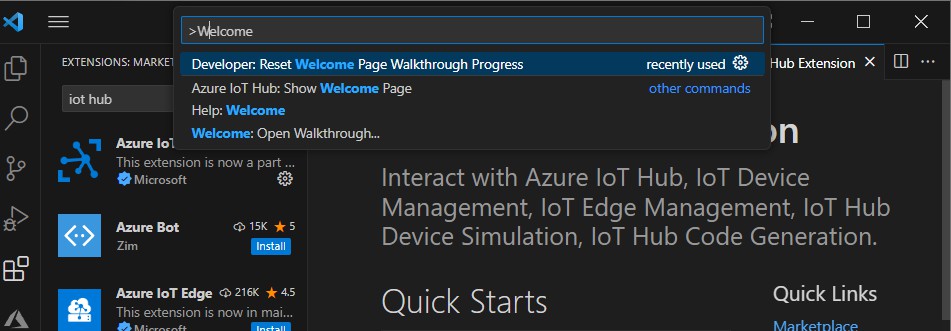
**Step 5:** Here we are in VS Code that is already installed with the Azure IoT Hub Toolkit extensions. If not install VS Code and add IoT Hub Toolkit extensions. Access Azure IoT Hub Toolkit through the Command Palette. One way to get to the Command Palette is by

going to the View menu and choosing Command Palette, like this.

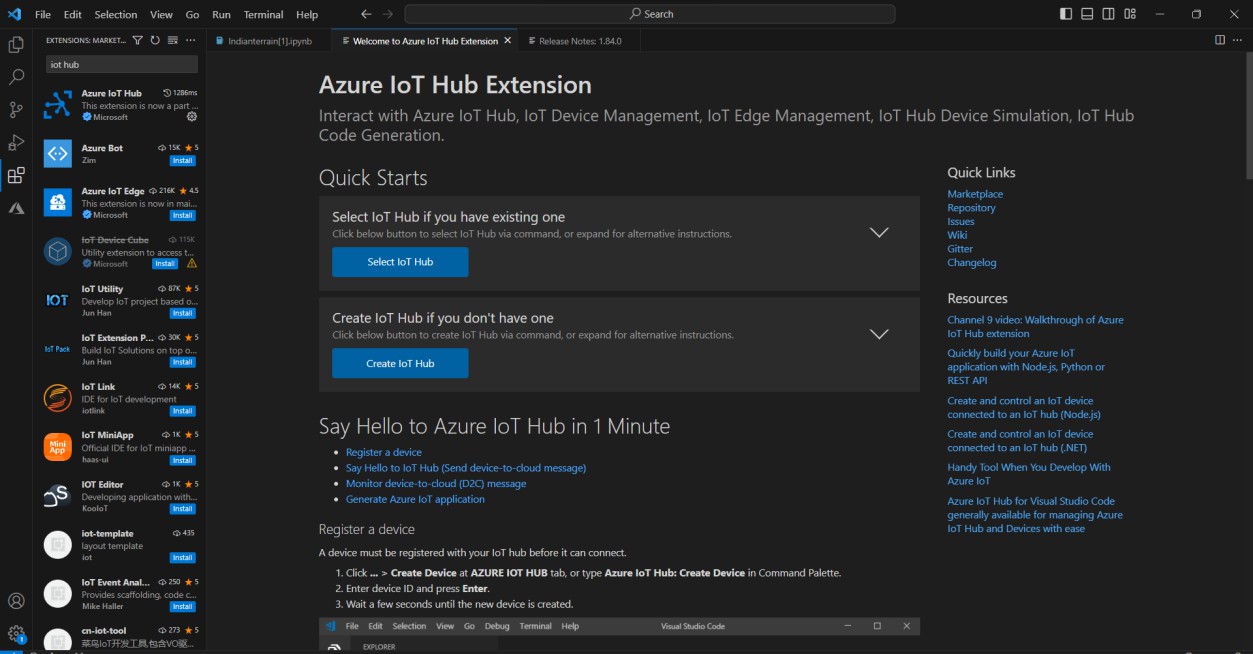




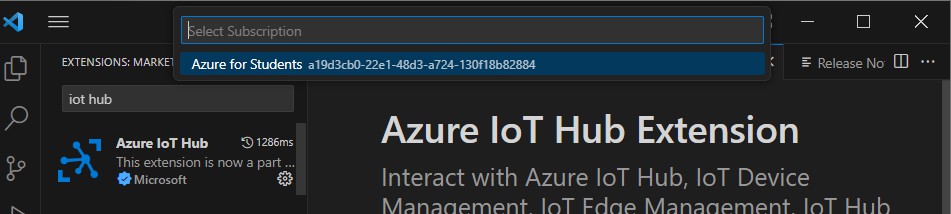
**Step 6:** Type welcome in our Command Palette, then choose to show the Azure IoT Hub welcome page.



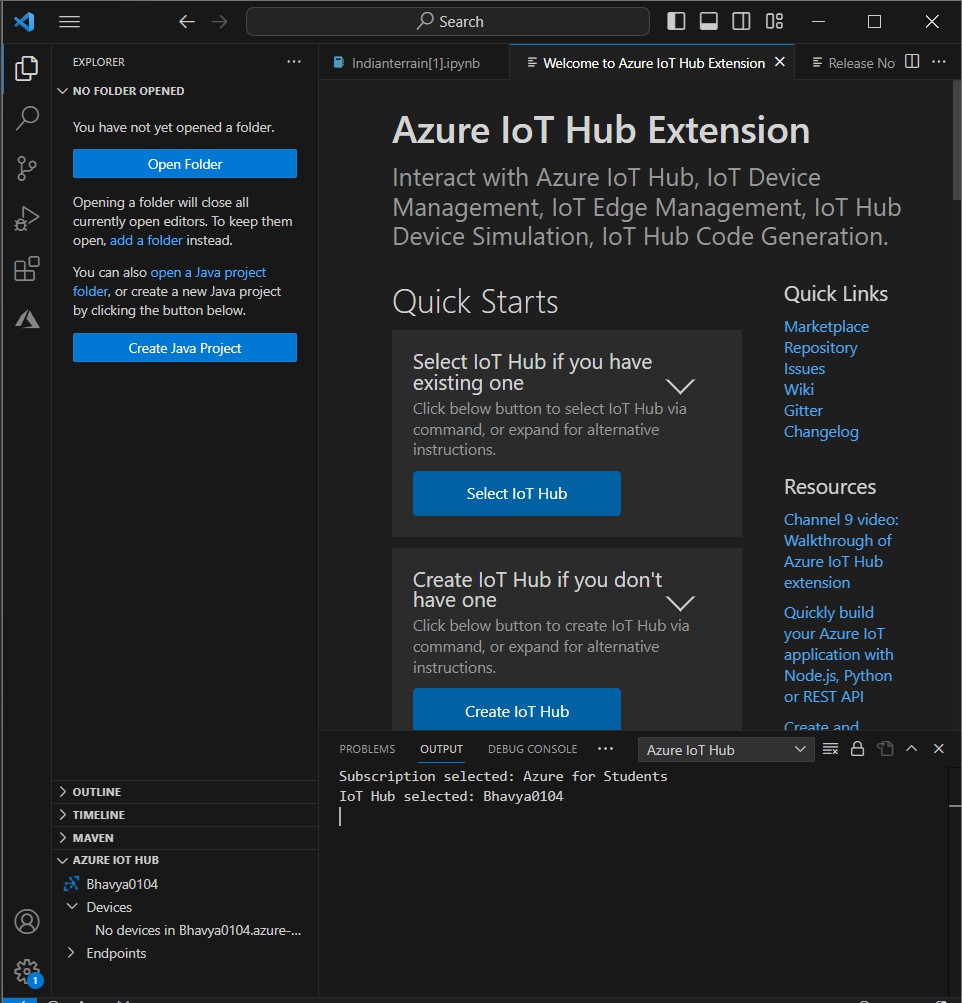
**Step 7:** This welcome page has some links and instructions to help us get started. We already have a hub, so let's choose to select an existing hub.



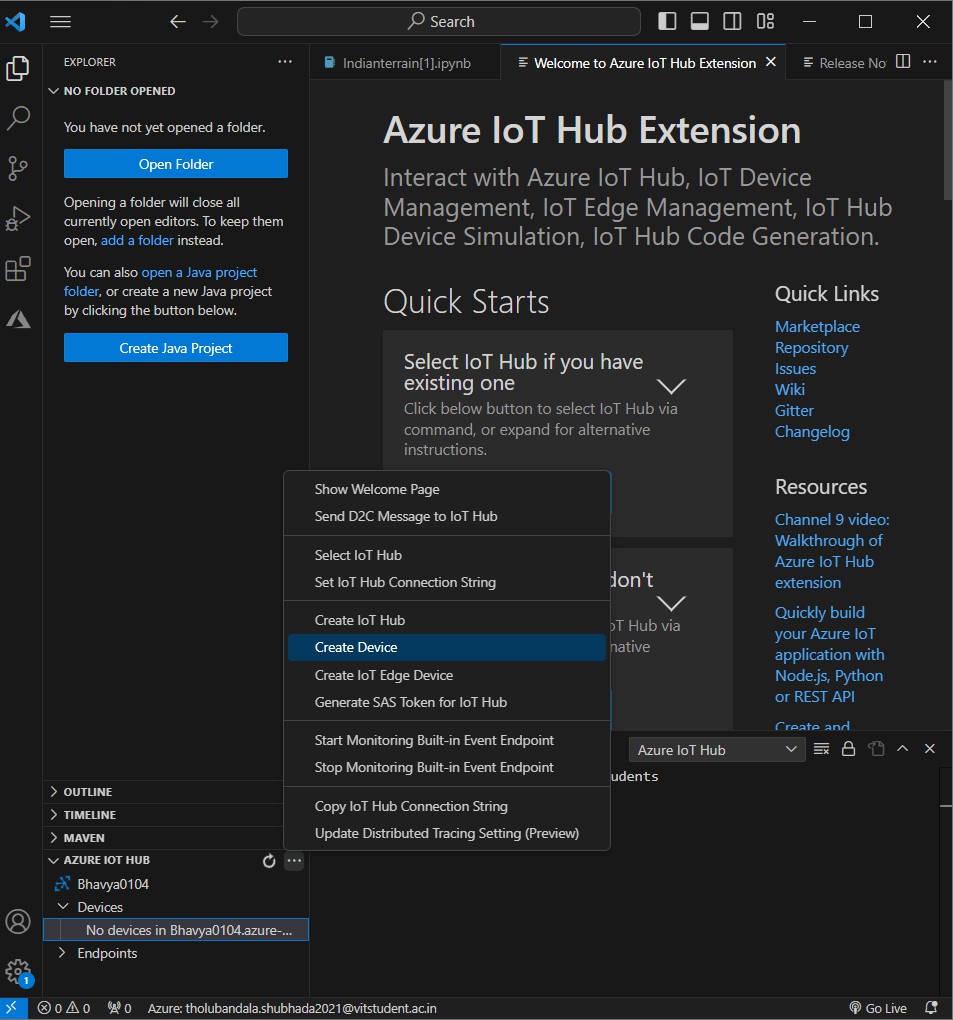
That opens up the command Palette where we can select the subscription to use, and then we can select the hub we want to use.



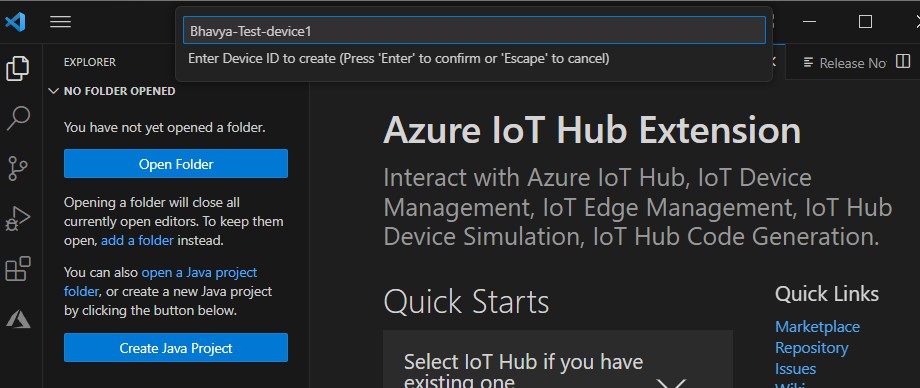
Now that we're connected, we can expand the Azure IoT Hub Devices panel down here, and we can see that we have no devices yet



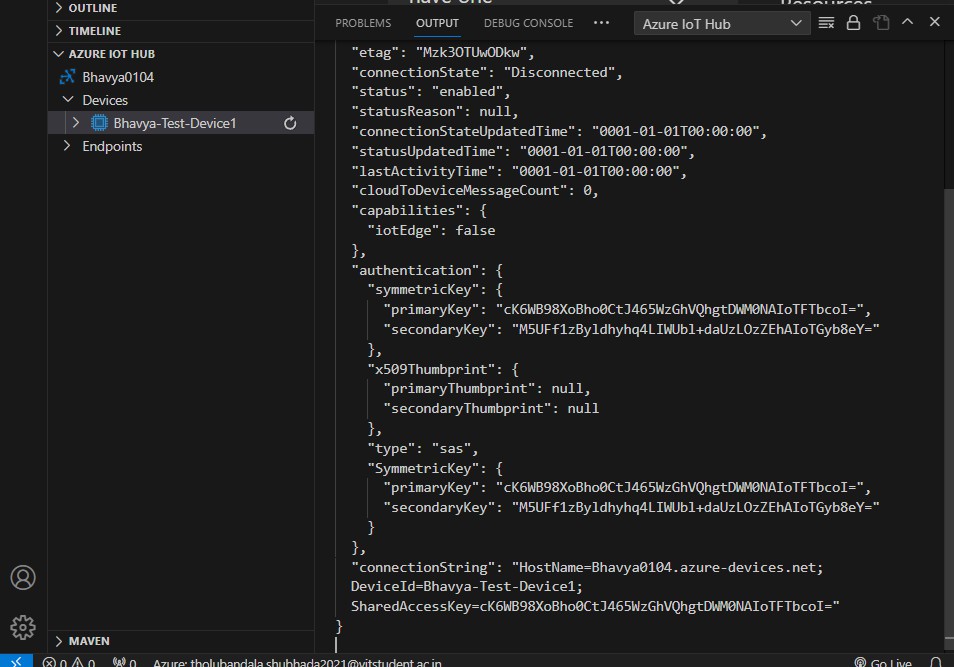
**Step 8:** Click the Action menu, and then choose Create Device.



**Step 9:** That will again pop open the Command Palette, this time for us to type a name for our device. Here, the name Bhavya-Test-device1 is used



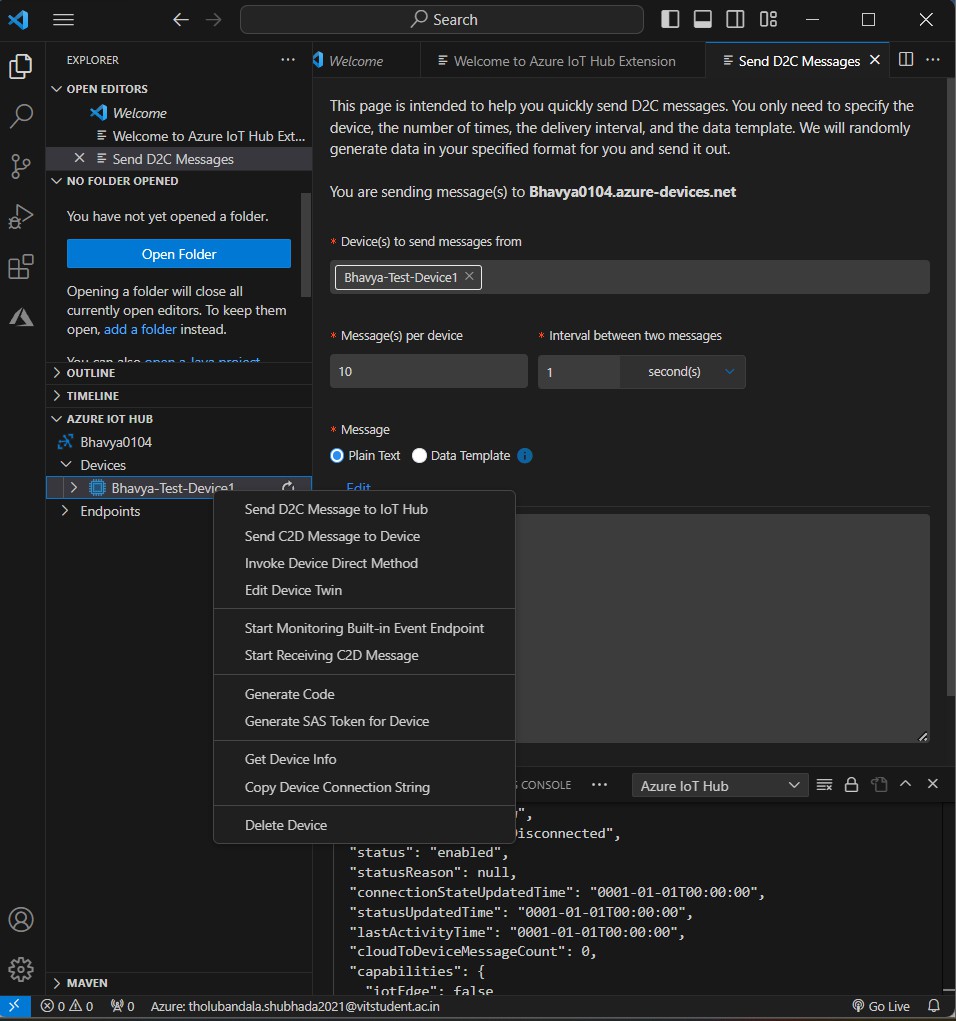
**Step 10:** After successful creation the following message will be displayed in Terminal section



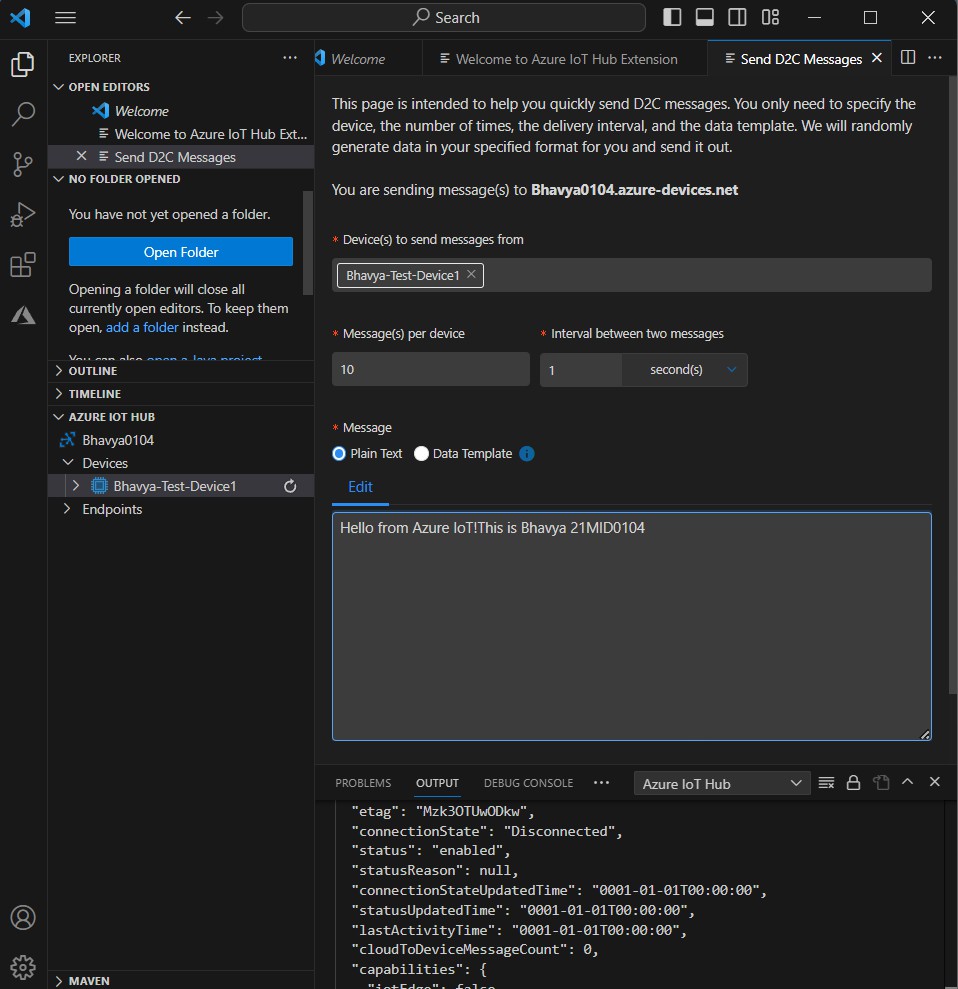
**Section III:** Sending a D2C Message (VS Code)

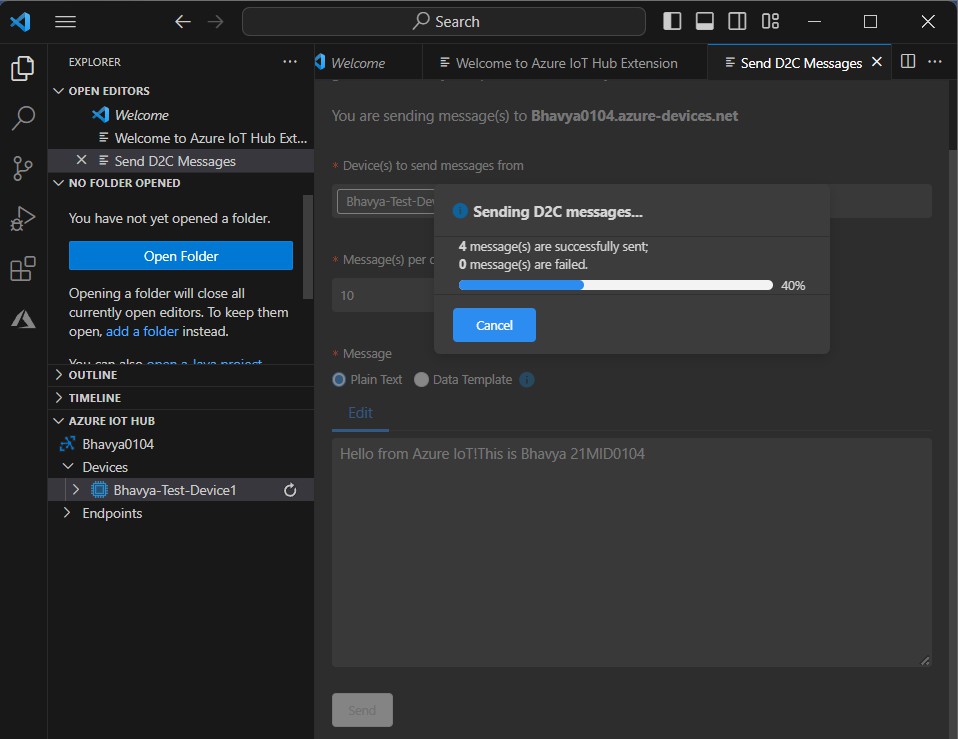
**Note:** Now that we've created device identities on our hub, we're ready to explore sending and receiving messages, but how are we going to do that without a real device? Well, we actually don't need a

real device, because Microsoft has given us tools that allow us to send and receive messages as if we were using a real device.

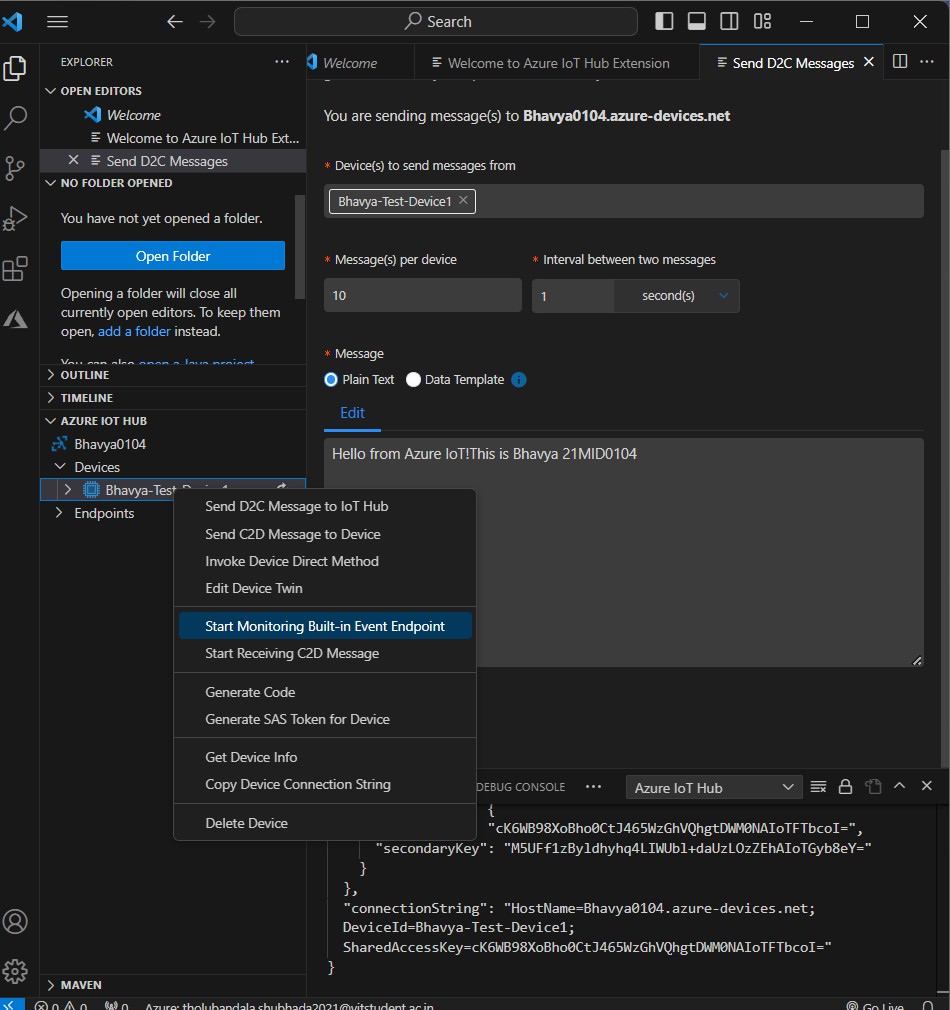


That will open the Command Palette for us to type the message that we want to send. Let's send Hello from VS Code

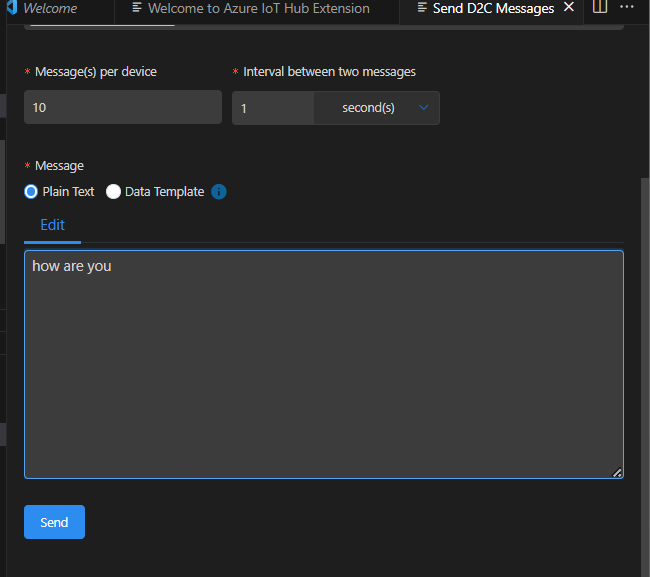


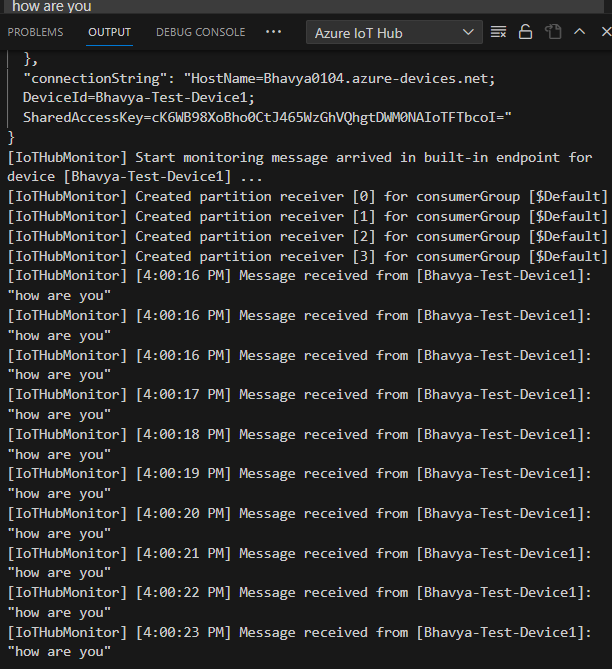


**Step 11:** Well, it looks like it worked, but how can we be sure? We need some way to monitor and see what messages our IoT Hub is actually receiving. Fortunately for us, we can do that right here in VS Code. Let's right-click on our device again, and this time let's choose **Start Monitoring Built-in Event Endpoint**



Now let's send another message. Let's give this message a different payload so we can tell the difference between this message and the one we sent before. And look at that, there's our message. It arrived at the hub exactly as we expected.

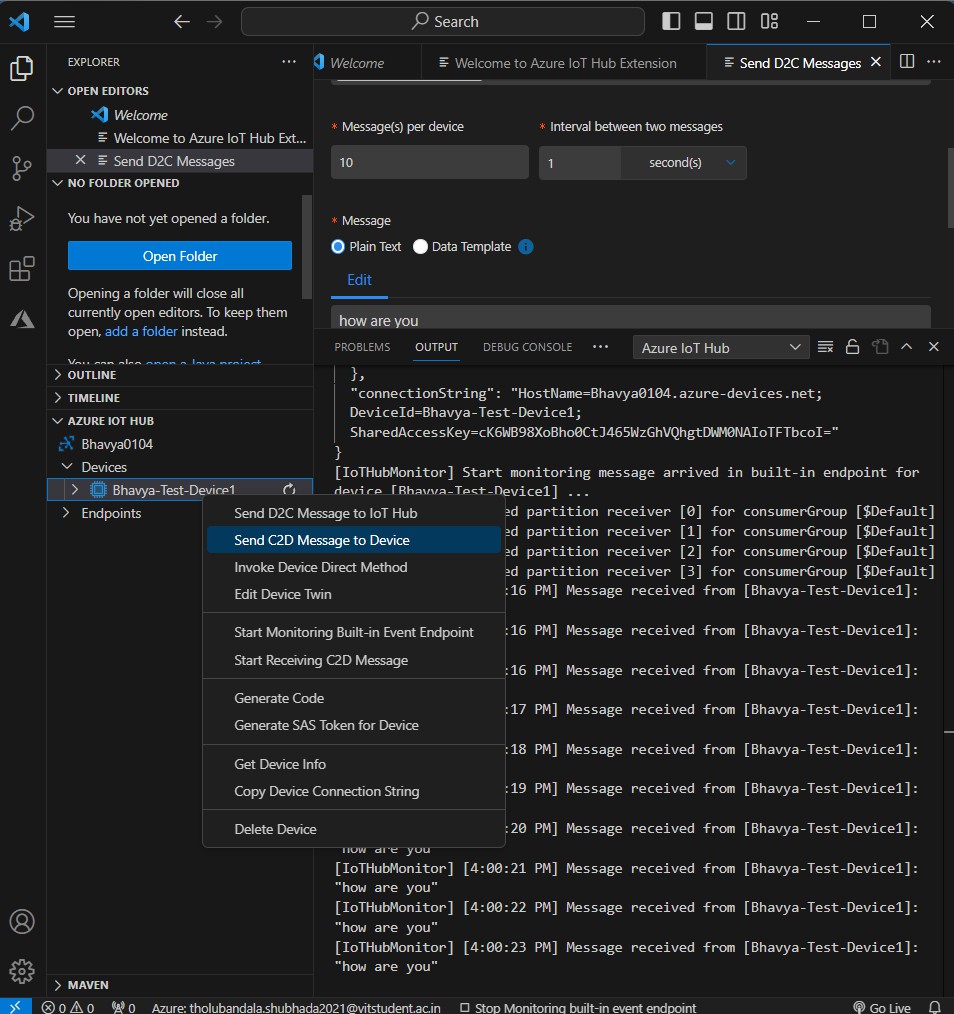




**Section IV:** Sending a C2D Message (VS Code)

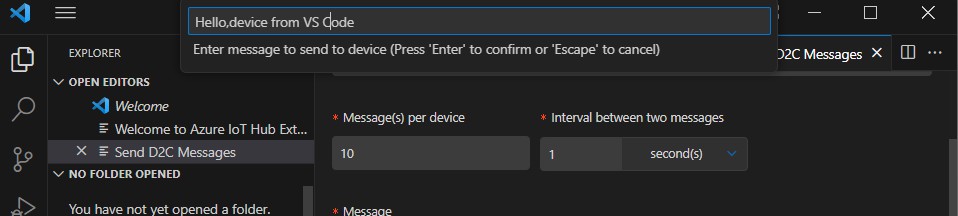
**Note:** We don't need a real device to explore how IoT Hub handles cloud-to-device messaging. All we need is our trusty IoT extensions for VS Code. Let's check it out. Sending a cloud-to-device message is basically the same as sending a device-to-cloud message.

**Step 12:** Right-click the device and choose "Send C2D Message to Device"

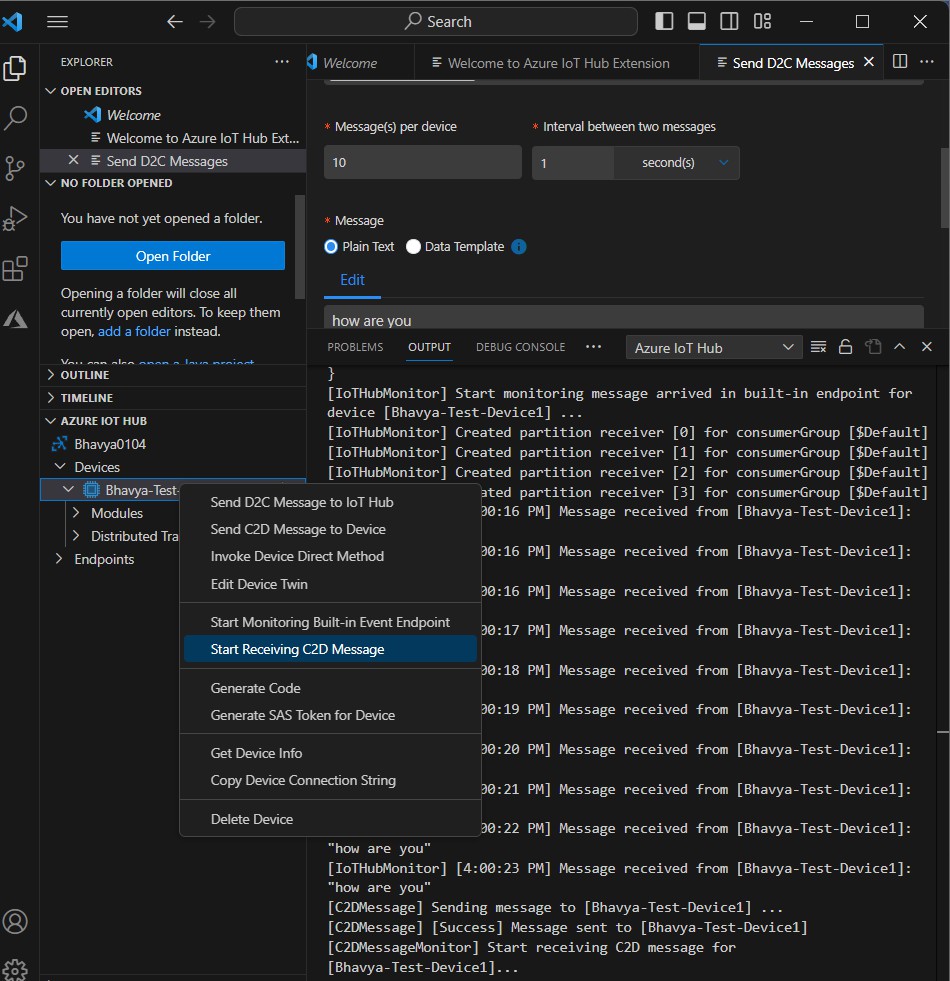


**Step 13:** That opens the Command Palette where we can drop in the value we want to send, and off the message goes. VS Code also gives

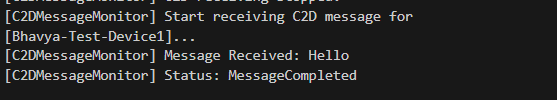
us the ability to receive that message without the need for a real device.



**Step 14:** Right-click on our device again, and this time choose Start Receiving C2D Message.



**Step 15:** Send another message. VS Code received the message on behalf of our device from the hub just like a real device would



**Step 16:** Go to Azure and open the created IoT Hub. It will show the number of messages used

