**CASE STUDY - CONTAINERIZATION USING DOCKER**

**Problem Statement:**

You work as a Devops Engineer in a leading Software Company. You have been asked to Dockerize the applications on the production server. The company uses custom software, therefore there is no pre-built container which can be used.

**Assume the following things:**

1. Assume the software to be installed is apache

2. Use an Ubuntu container

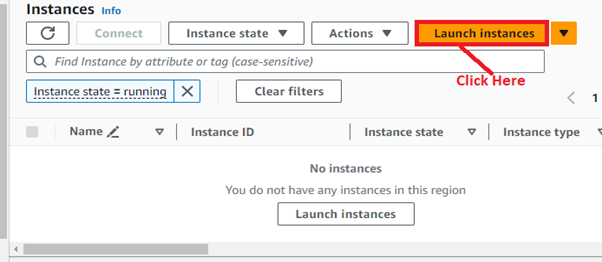
**The company wants the following things:**

1. The Developers will not be working with Docker, hence from their side you will just get the code. Write a Docker file which could put the code in the custom image that you have built.

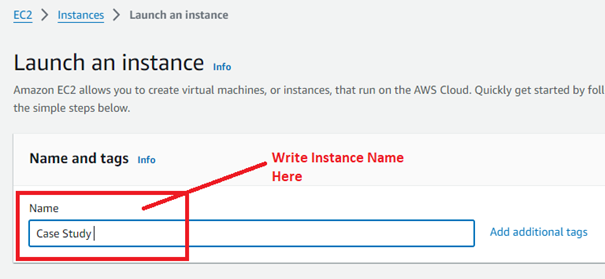
2. Push an Image to Docker-Hub with the above config.

**1. Create a New Instance using the Ubuntu Machine**

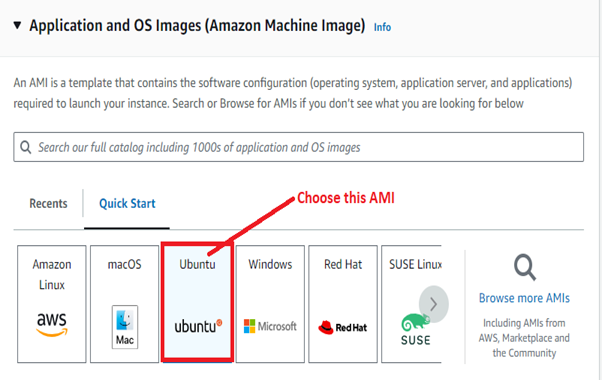
**Step 1: First, we**will**launch** an**instance. Click**onthe**“Launch Instance”.**



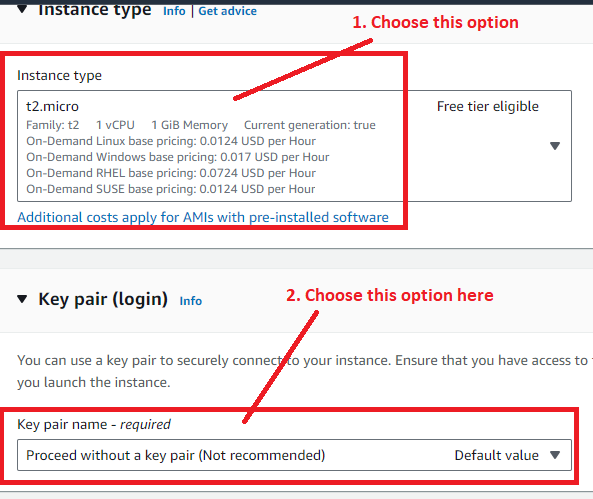
**Step 2: Choose**the**“Name”**asthe **“Case Study”.**



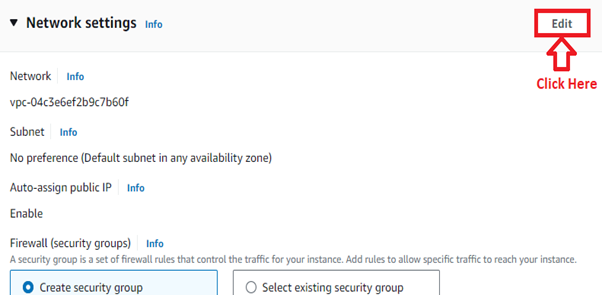
**Step 3: Select** the**“AMI”**as the **“Ubuntu”.**



**Step 4: Select** the **“Instance Type”** asthe **“t2.micro”**&**choose** the **“Key pair (login)”**as**“Proceed without a key pair”.**



**Step 5: Click**on the**“Edit”**inthe **“Network Settings”.**



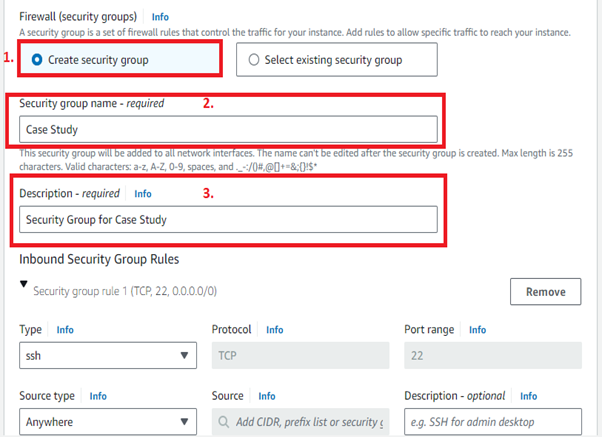
**Step 6: Select**these **options**here:

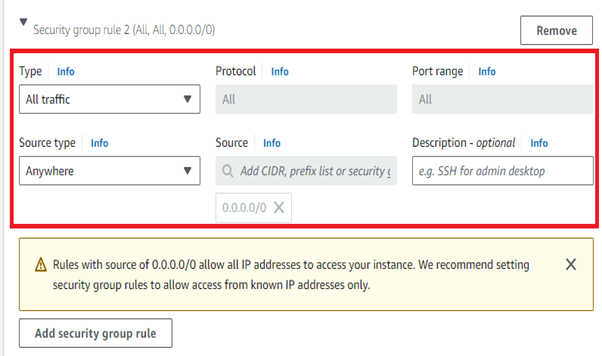
**a. First,**choose**“Create Security Group”**here.

**b. Security group name — required:**Case Study

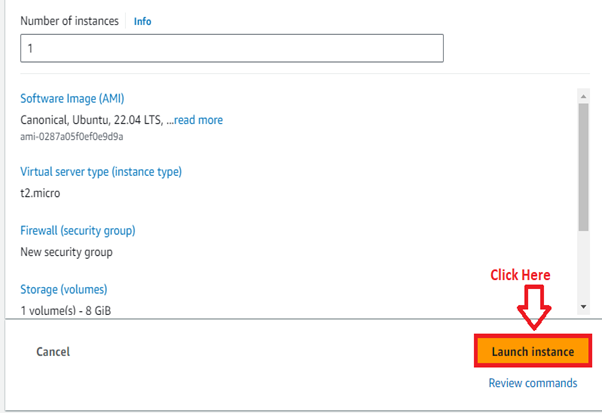
**c. Description:**Security Group for Case Study

**Select** the **“All Traffic”**inthe**“Security group rule 2”**withthe **“Source type”**as**“Anywhere”.**

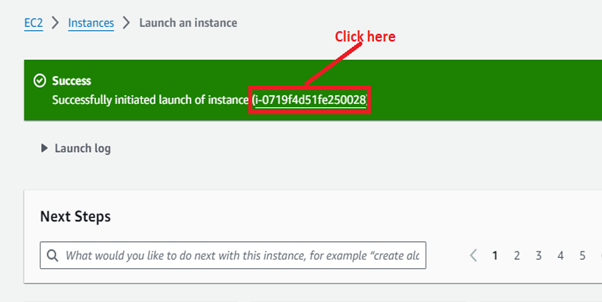




**Step 7: Click** on the**“Launch Instance”.**

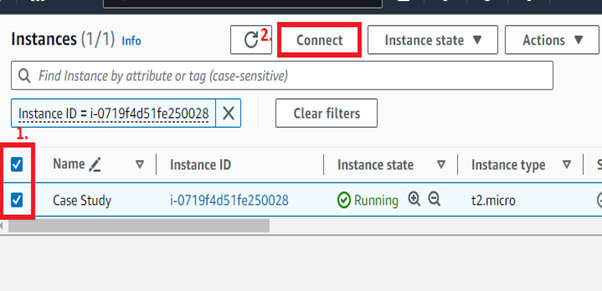


**Step 8: Click**on the**“Hyperlink”**to**view**the**“Instance”.**

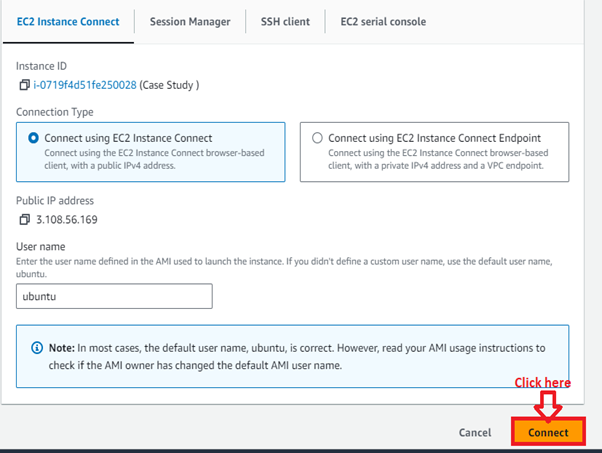


**Step 9: Go** to the **“Instance”**by **clicking** on the **hyperlink.**

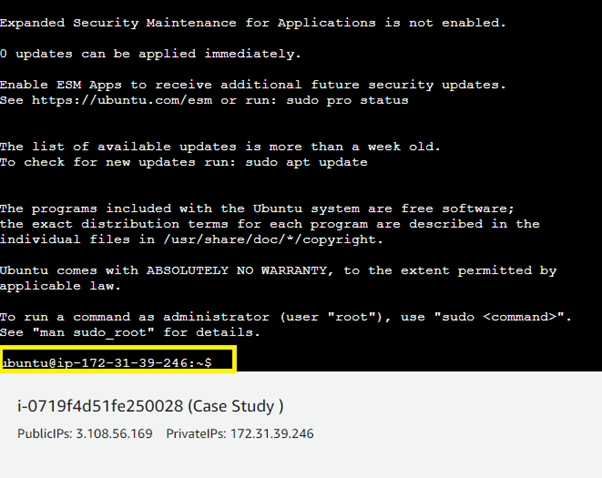
**Select**the**instance**&**click**on the**“Connect”.**



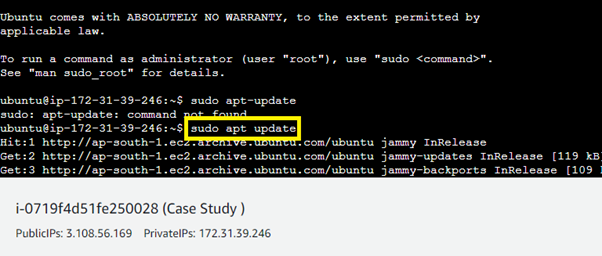
**Step 10:**Click**on “Connect”**in the **“EC2 Instance Connect”.**



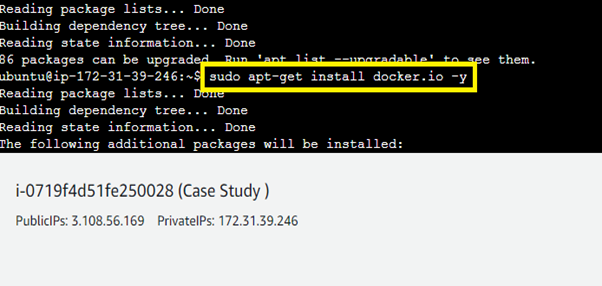
**Step 11: The “Ubuntu Machine”**is **ready**to**operate.**

**2. Update the Ubuntu Machine & Install Docker**

**Step 1: Run**this **command to update**the**machine: sudo apt update**



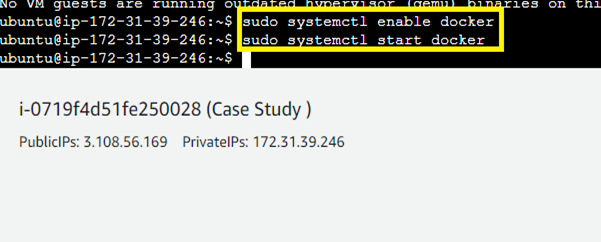
**Step 2: Install Docker using**this**command: sudo apt-get install docker.io –y**



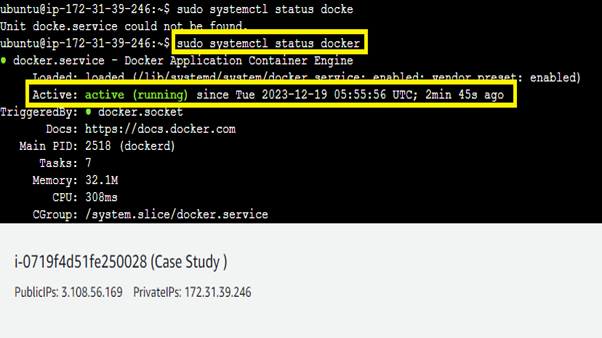
**Step 3: Use**these **commands**to**start & enable**the**docker:**

**sudo systemctl enable docker**

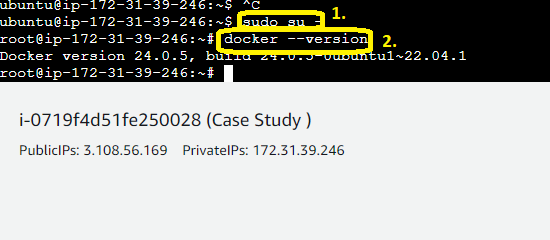
**sudo systemctl start docker**



**Step 4: To check** the**status**of the **Docker, type**this**command: sudo systemctl status docker**

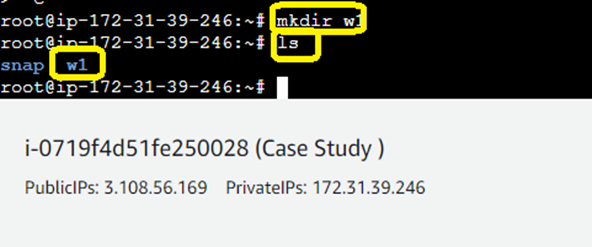


**Step 5: Login**as the**“root user”**throughthe**“sudo su –“**command.**Type** the **“docker --version”**to**check**the**Docker current version, which**we are**using.**

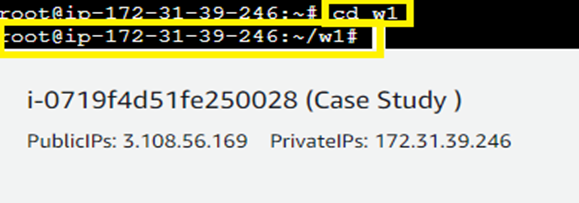


**3. Create a New Folder & Clone the Git-Hub Repository**

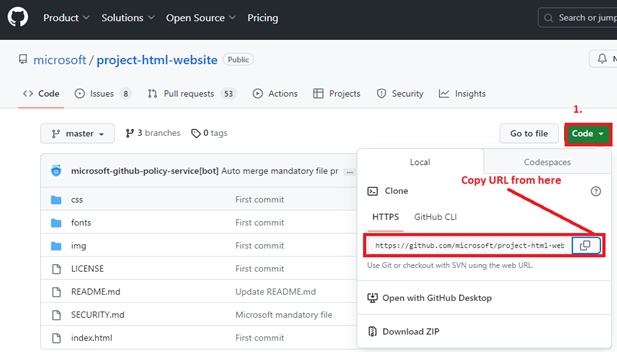
**Step 1: We** create the**“w1” directory**using the**“mkdir w1”**command &**type** the **“ls” command**to**check**the**directory**here.



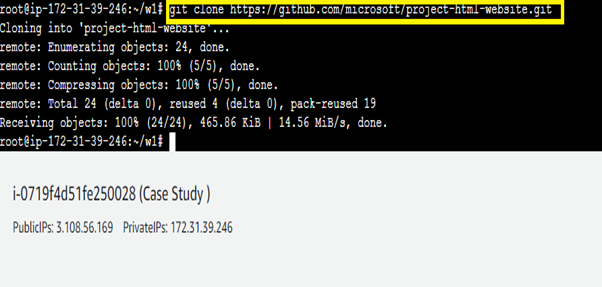
**Step 2: Go inside**the**directory “w1” using**the**command: cd w1**



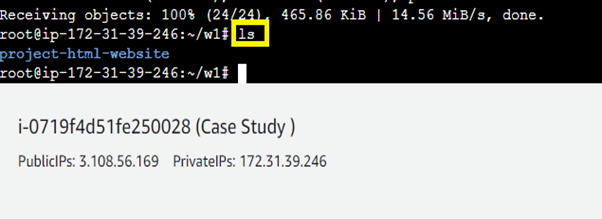
**Step 3: We**have**opened** the**“project-HTML-website”**directory on the **“Git Hub”. Click**onthe **“Code”**&**Copy**the**Given URL**from here: [**https://github.com/microsoft/project-html-website.git**](https://github.com/microsoft/project-html-website.git)



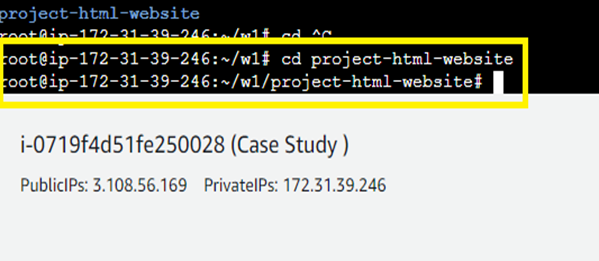
**Step 4: Type**this**command**to**download**the**“html website” repository**inthe**“w1”**directory: **git clone https://github.com/microsoft/project-html-website.git**



**Step 5: Run** the **“ls” command** &**the “project-html-website”**have beensuccessfully**downloaded.**

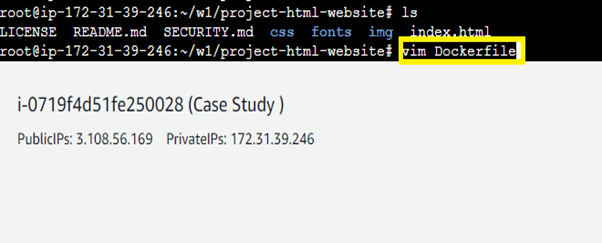


**Step 6: Run** the**“cd project-html-website” to go**inside this**repository. You**willland**inside**the**“project-html-website”: cd project-html-website**



**Problem 1 Solution: The Developers will not be working with Docker, hence from their side you will get the code. Write a Docker file that could put the code in the custom image that you have built.**

**Step 1: First, create**the**docker file using** this**command: vim Dockerfile**



**Step 2: Put**this**code inside**the Docker file**:**

**FROM ubuntu**

**RUN apt-get update**

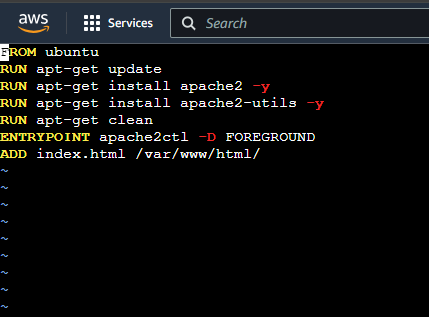
**RUN apt-get install apache2 -y**

**RUN apt-get install apache2-utils -y**

**RUN apt-get clean**

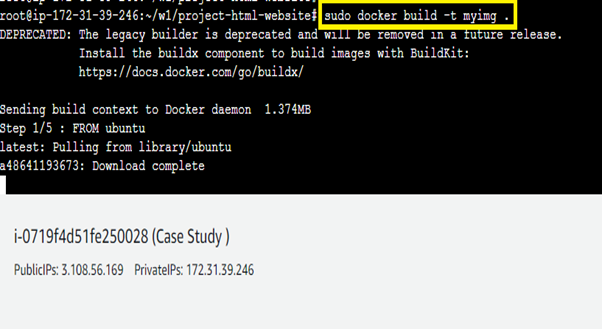
**ENTRYPOINT apache2ctl -D FOREGROUND**

**ADD index.html /var/www/html/**



**Press “ESC”**from the **keyboard**&**type “:wq!”**to**quit & save the file.**

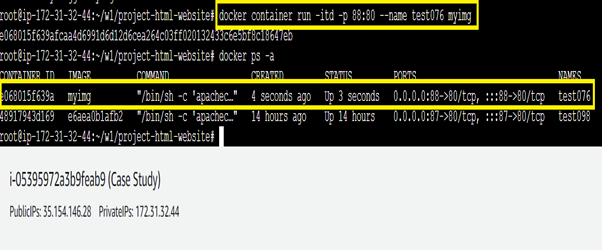
**Step 3:**Now,**create**an**image using**this **command: sudo docker build –t myimg .**



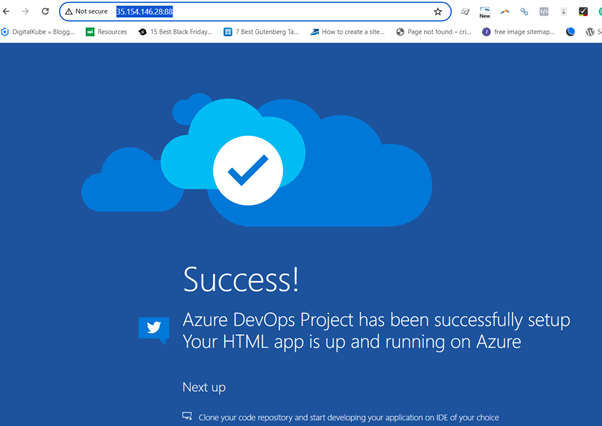
**Step 4: Now, create**a **container**using this**command: docker container run –itd -p 88:80 –name test076 myimg**

**Do “docker ps –a”**to**check whether**the **container**is**active**or**not.**

**We**have**created**the**container**on**the port 88.**



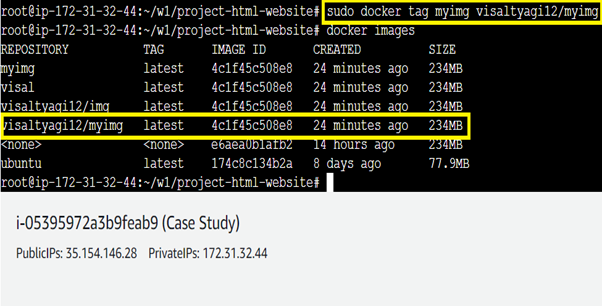
**Step 5: Copy**the**IP Address**&**Paste**it intothe**new browser address bar**like this:[**http://35.154.146.28:88/**](http://35.154.146.28:88/)



**Your website**has been**successfully deployed**over**port no. 88.**

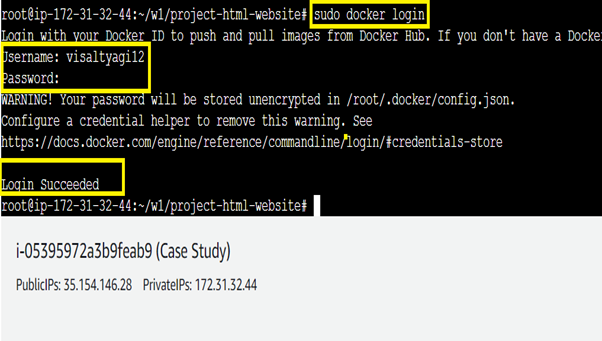
**Problem 2 Solution: Push an Image to Docker-Hub with the above config.**

**Step 1: Tag**an**image**to **push**into the**Docker Hub. Use**this**command: sudo docker tag myimg visaltyagi12/myimg**

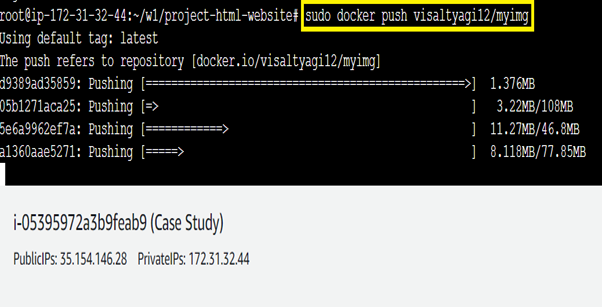


**An image**with the **name visaltyagi12/myimg**has been**successfully created.**

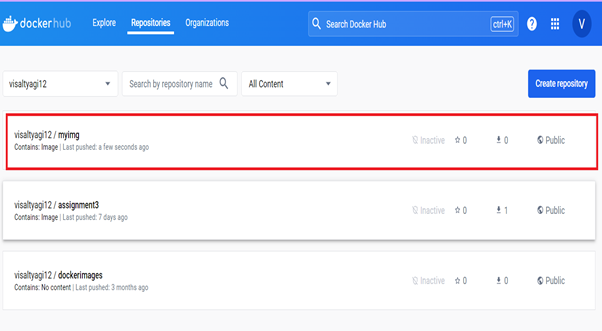
**Step 2:**First **login**into**docker using**this **command: sudo docker login**withthe **username & password. The login**will be**successful.**



**Step 3: Use**this**command**to**push**the**image: sudo docker push visaltyagi12/myimg**



**Step 4: Login**into your**Docker hub account & go**tothe **“Repository”**section. **All images**will be **shown**here.



**🡨------------------------------Docker Case Study Solution Completed---------->**