Programming for Network Engineer

**Lab4: Package & Deploy the Python on Cloud**

CONTENTS

[A. OVERVIEW 2](#_Toc204502787)

[1. Objectives 2](#_Toc204502788)

[a. Understand how to work with containers in local and cloud environment: 2](#_Toc204502789)

[b. Environment 2](#_Toc204502790)

[2. Guide to build the server on Docker desktop 2](#_Toc204502791)

[a. Use this Dockerfile and the bin-chatserver.py 2](#_Toc204502792)

[3. Create Container Instance on Azure Cloud 4](#_Toc204502793)

[Create Azure Account with $200 free credit: Create Your Azure Free Account Or Pay As You Go | Microsoft Azure 4](#_Toc204502794)

[B. Reference 8](#_Toc204502797)

# OVERVIEW

## Objectives

### Understand how to work with containers in local and cloud environment:

* Understand how to work with Docker desktop.
* Create Dockerfile, push to Docker hub registry

### Environment

* Python Virtual Environment: <https://www.anaconda.com/docs/getting-started/miniconda/main>
* IDE: PyCharm Community, VS Code
* Download and install Docker desktop: <https://desktop.docker.com/win/main/amd64/Docker%20Desktop%20Installer.exe?utm_source=docker&utm_medium=webreferral&utm_campaign=dd-smartbutton&utm_location=module>

## Guide to build the server on Docker desktop

### Use this Dockerfile and the bin-chatserver.py

|  |
| --- |
| FROM python:3.12-slim  # Create a non-root user to run the application  RUN useradd -m appuser  # Set working directory in the container  WORKDIR /app  # Copy requirements first (for better Docker layer caching)  COPY requirements.txt .  RUN pip install --no-cache-dir -r requirements.txt  # Copy only necessary files to the container  COPY server.py .  # Create the rooms directory where files will be stored  RUN mkdir -p rooms  # Change ownership of the app directory to appuser  RUN chown -R appuser:appuser /app  # Switch to the non-root user  USER appuser  # Expose the port the app runs on  EXPOSE 8888  # Command to run the application  CMD ["python", "server.py"] |

Command to build the Docker image:

*$ docker build -t 0phaimchau/project:v1 .*

* 0phaimchau: is the username of hub.docker.io
* project: is the name of the image
* v1: is the version of the image

Run the container from the image:

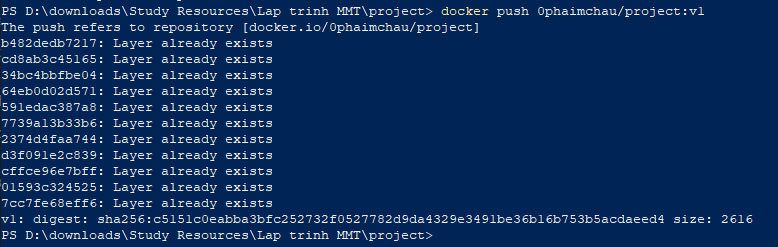
*$ docker run -p 8080:8080 0phaimchau/project:v1*

* -p: to map the inside container port as 8080 to outside port as 8080

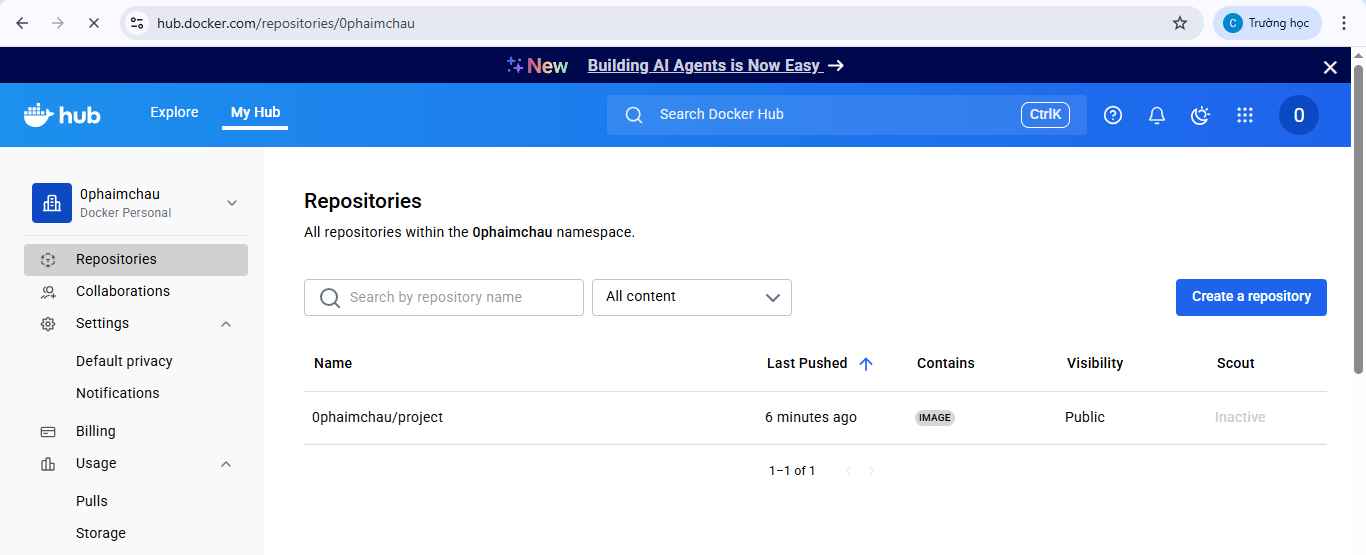
After running you can run the client.py

Before pushing the container to Docker Hub you need to register and login to your account here: <https://hub.docker.com/> . Command to push

*$ docker push 0phaimchau/project:v1*

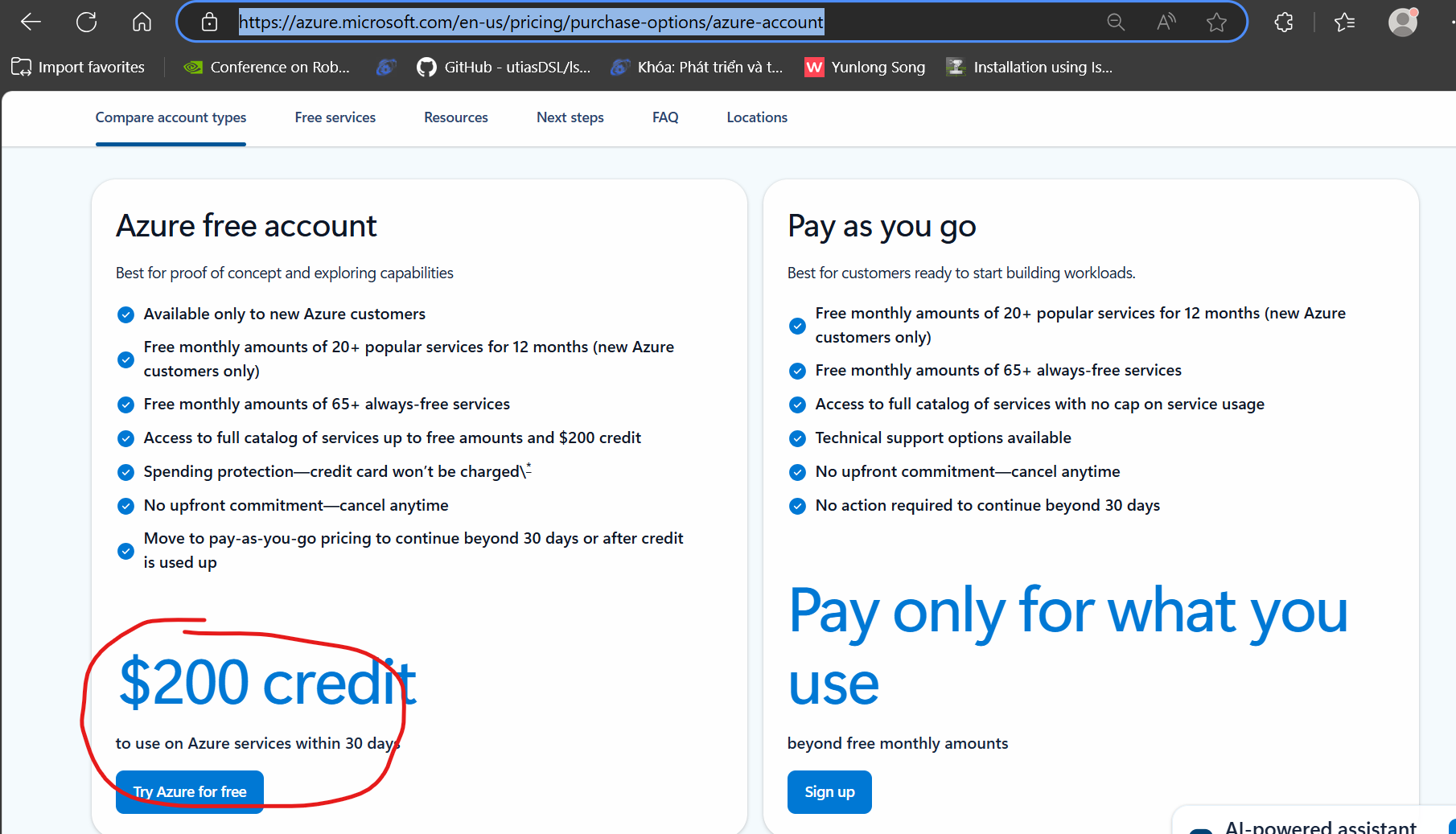
**

After the push, you should check your Docker Hub Account to see if the image was there:



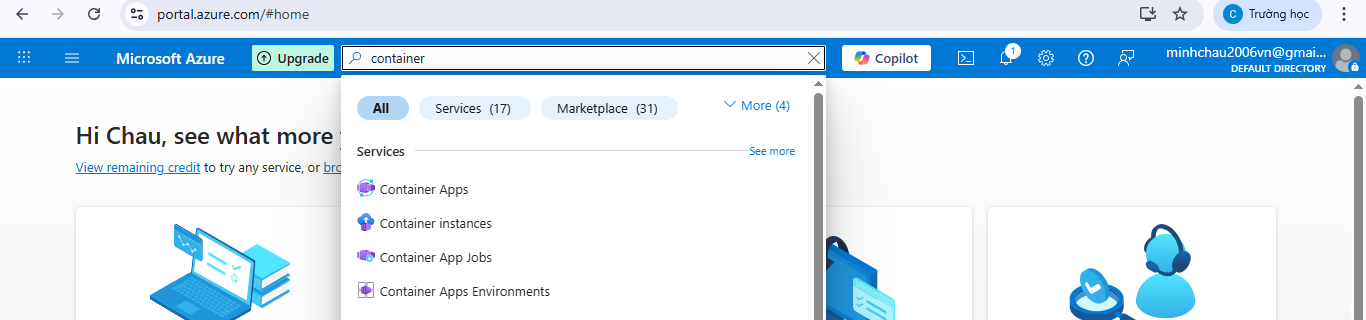
## Create Container Instance on Azure Cloud

### Create Azure Account with $200 free credit: [Create Your Azure Free Account Or Pay As You Go | Microsoft Azure](https://azure.microsoft.com/en-us/pricing/purchase-options/azure-account)

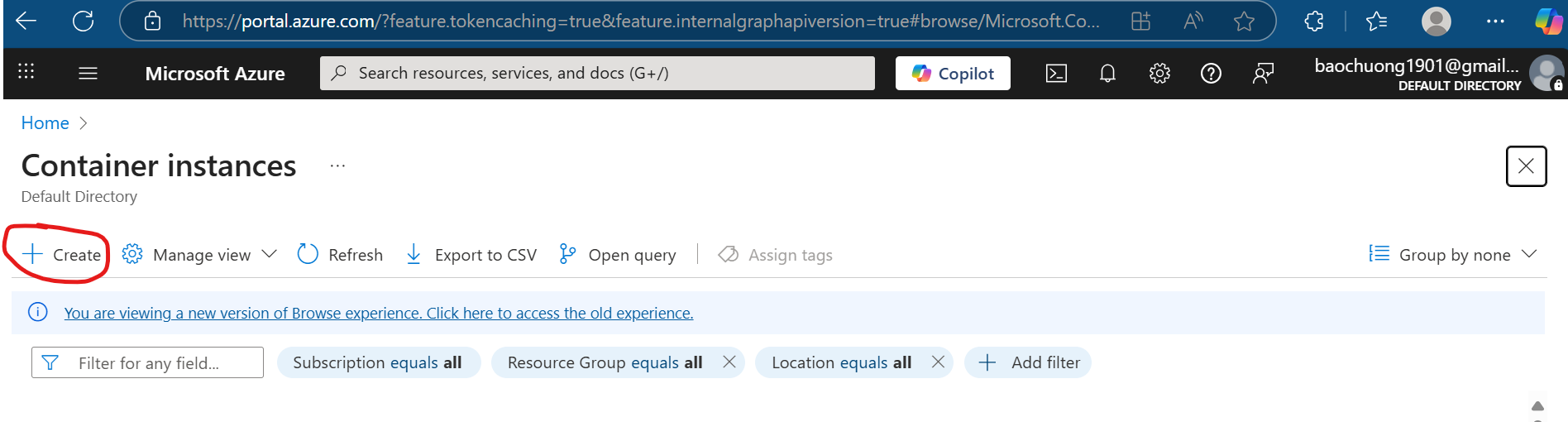


After you create your Azure account, go to: <https://portal.azure.com/#home>

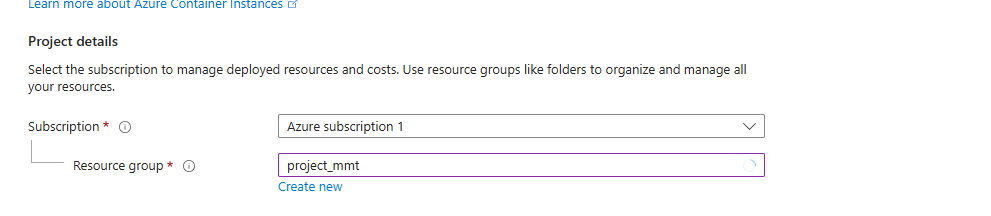
1. Search for “container” and choose container instance:



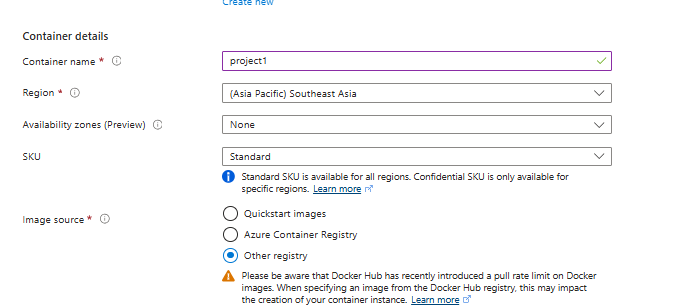
1. Choose “Create”



1. In the wizard, enter the name of new resource group

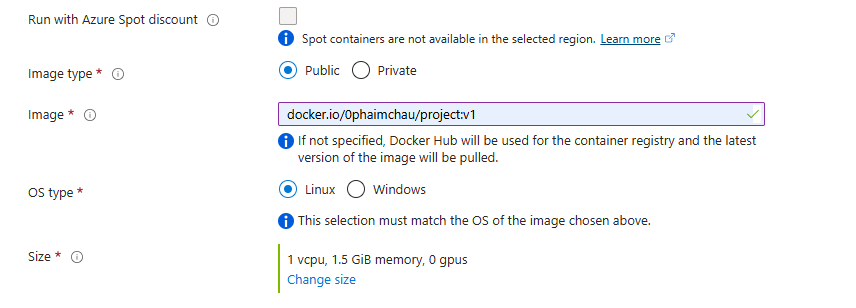


1. Name the container and choose “Other Registry”

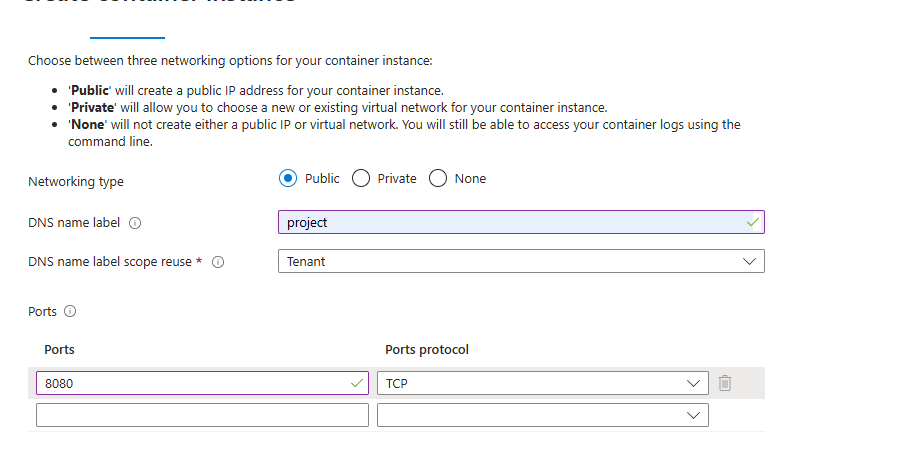


1. You need to make sure image already pushed to docker hub

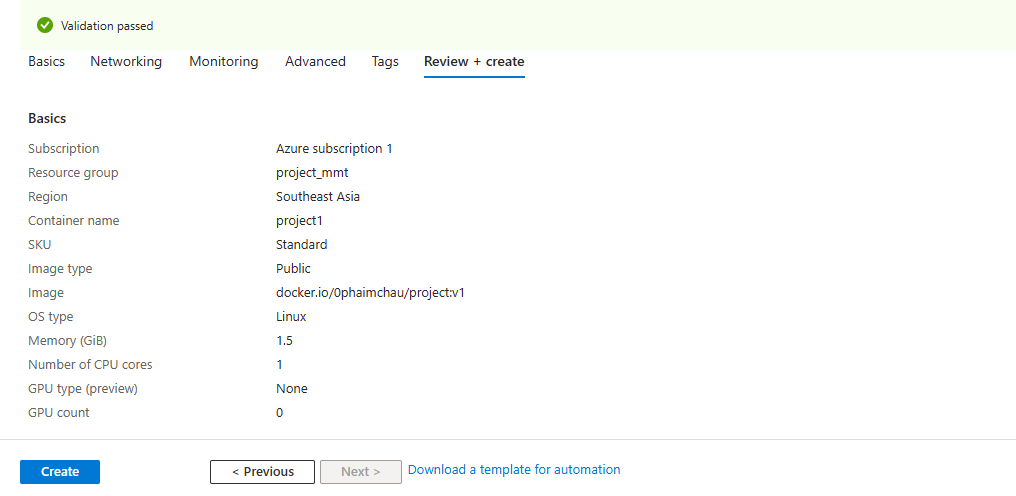
Image: docker.io/<your\_account\_name>/<name\_of\_image>:<tag\_version>



1. Click “Next: Networking” , you can (optionally) enter the DNS label and need you make sure to enter the right PORT number



1. Now, you can click “Next: Monitoring” , then “Next: Advanced”, then “Next: Tags” with everything keep as default and finally click “Next: Review + Create”. Click “Create”



# Reference