UNIVERSITY OF INFORMATION TECHNOLOGY

FACULTY OF COMPUTER NETWORKS AND COMMUNICATIONS

NETWORKS AND SYSTEMS ADMINISTRATION

PROJECT REPORT

GROUP 07:

DEPLOY WEB APPLICATION WITH DOCKER/ AZURE CONTAINER

Lecturer: MSc. Tran Thi Dung

Students: Nguyen Dai Nghia - 21521182

Pham Hoang Phuc - 21521295

Hoang Gia Bao - 21521848

Table of contents

CHAPTER 1: INTRODUCTION

1.1 General information

1.1.1 Docker overview

Docker is an open platform for developing, shipping and running applications. It allows applications to be separated from the infrastructure so that software can be delivered quickly. Docker provides the ability to package and run an application in an isolated environment called a container. The isolation and security lets us to run many containers simultaneously on a given host. Containers contain everything needed to run the application, so we don't need to rely on what's installed on the host. They are highly portable and can run on any system that has Docker installed, regardless of the underlying operating system or infrastructure. This portability ensures consistency across different environments and reduces the risk of compatibility issues.

By taking advantage of Docker for deploying applications, we can significantly reduce the delay between writing code and running it in production.

1.1.2 Azure overview

Microsoft Azure is a cloud computing platform and an online portal to access and manage resources and services provided by Microsoft. It provides a wide range of cloud services, including compute, storage, analytics and networking, to help individuals and organizations build, deploy, and manage applications and services through Microsoft-managed data centers.

Azure App Service is a fully managed platform-as-a-service (PaaS) offering provided by Microsoft Azure. It allows to build, deploy and scale web, mobile, API applications easily without having to work with the underlying servers, storage or network assets. It offers different deployment options, including code-based deployments, container-based deployments, and integration with source control systems like GitHub and Azure DevOps.

1.2 Component

1.2.1 Docker

The Docker daemon

The Docker daemon operates as a background service on the host machine. It listens for Docker API requests and manages Docker objects such as images, containers, networks and volumes. A daemon can also communicate with other daemons to manage Docker services.

The Docker client

The Docker client is the way that users communicate with Docker. Whenever a user gives a command to Docker, the Docker client sends the desired command to the Docker daemon, which carries them out with Docker API. The Docker client can communicate with more than one daemon.

Dockerfile

A Dockerfile is a text document that contains all the commands users can call on the command line to build a Docker image. It is an instruction file that specifies the environment and steps required to set up and run applications or services in a container.

Docker Image

Docker images are read-only templates that include everything needed for running applications. They contain the code, libraries, tools, dependencies and other files needed to create a container to run on the Docker platform. When a user runs an image, it can become one or multiple container instances.

Docker container

Docker container is a runnable instance of an image that can be managed through the Docker API. We can use container for packaging an application with all the components it need, then ship it all out as one unit. Containers run in an isolated runtime environment, separate from the host machine and other containers and have defined resources.

Because of their isolation, containers are well-suited for securely running software like databases or web applications that need access to sensitive resources.

Docker Desktop

Docker Desktop is an application that provides a user-friendly interface and toolset for developers to build and deploy applications using Docker containers on a local machine. It is available for both macOS and Windows operating systems.

Docker Desktop may include components like Kubernetes, Credential Helper and Docker features such as the Docker Engine, Docker Compose, Docker CLI, and Docker Content Trust.

Docker registries

Docker registry is a place that stores Docker Images. It is where Docker images can be pushed, pulled and shared. Docker Hub is the default public registry provided by Docker that everyone can use to access images.

1.2.2 Azure App Service

Azure Web App

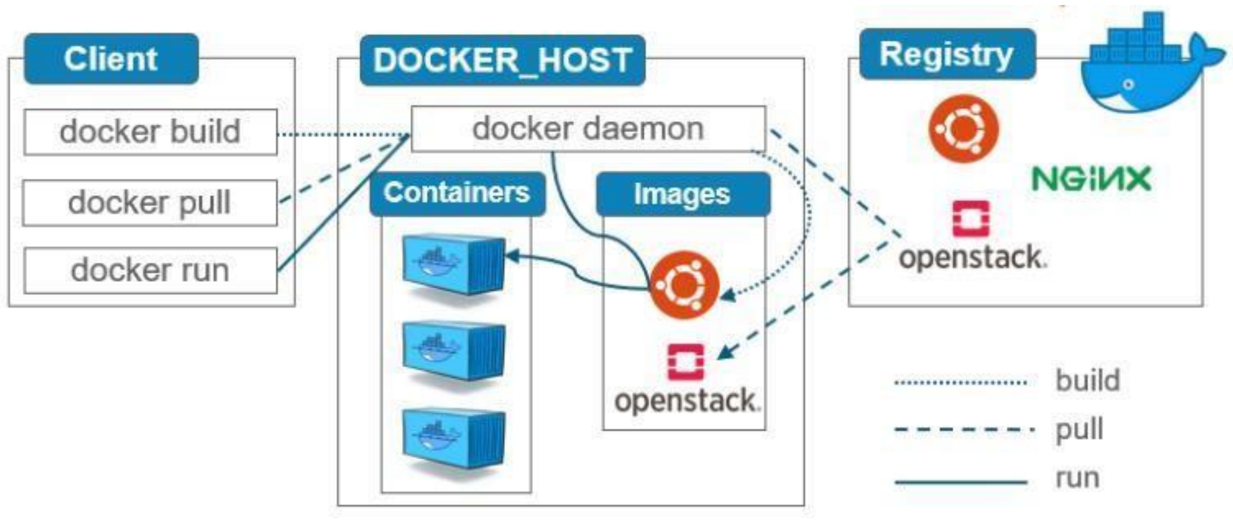
Azure Web App is a specific type of Azure App Service that is optimized for hosting web applications. With Azure Web App, developers can deploy various type of web applications, including static websites, dynamic web apps, and web APIs, with multiple programming languages and frameworks.

App Service Plan

An App Service Plan defines a set of compute resources for web application to run. It represents the underlying infrastructure and configuration settings for web applications. When creating an App Service plan in a certain region, a set of compute resources is created for that plan in that region. Each App Service plan defines OS, region, number of VM instances, size of VM instances, and pricing tier.

1.3 Operation

1.3.1 Docker workflow



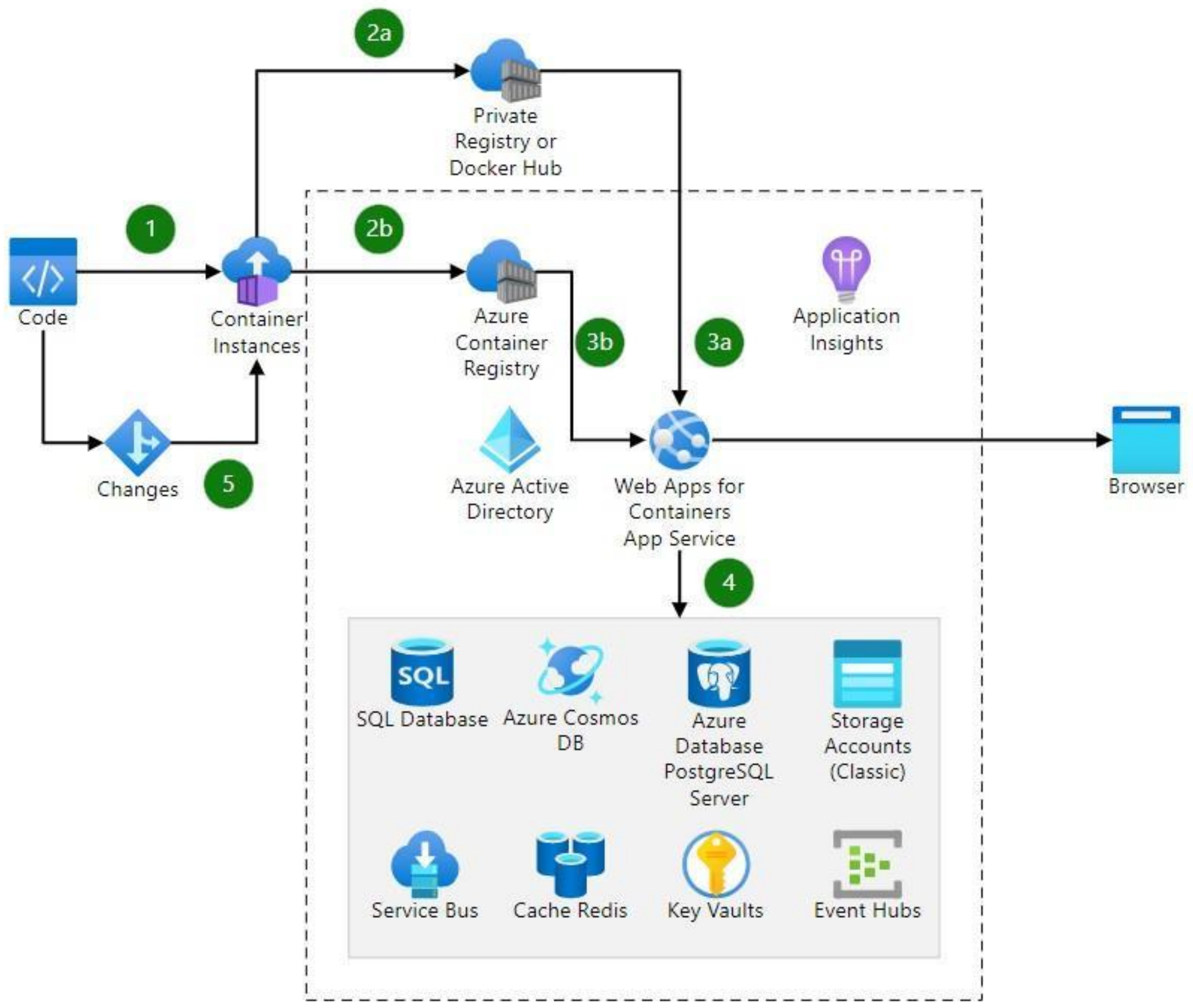
Docker's architecture has the client, daemon, host, and registry communicate with each other. Docker's operation process includes the following main activities:

Build a Docker image: Docker looks for a Dockerfile and creates a build context based on it. It will be sent to the Docker daemon to process the instructions in the Dockerfile and performs the necessary actions to build the image. Docker reads these instructions one by one and executes them. After executing all the instructions, Docker builds the final image. Then Docker assigns a unique identifier to the built image.

Run a container: Docker checks if image is available locally. If the image is not found, Docker attempts to pull it from Docker registries. Docker creates a new container based on the specified image and applies the configuration in that, including the environment variables, network settings, volume mounts, and other container-specific parameters. Then Docker starts the container and initiates the process defined in the image. The container can interact with the host machine and other containers through exposed ports or shared volumes. It continues running until the process it started completes or is manually terminated.

Pull images from Docker registries: Docker checks if the specified image exists in the registry through the provided image name and tag. If the image is found, Docker initiates the image download process. After the image is downloaded, Docker assigns a unique identifier to the image. If the image has a specified tag, Docker associates that tag with the image.

1.3.2 Azure App Service workflow

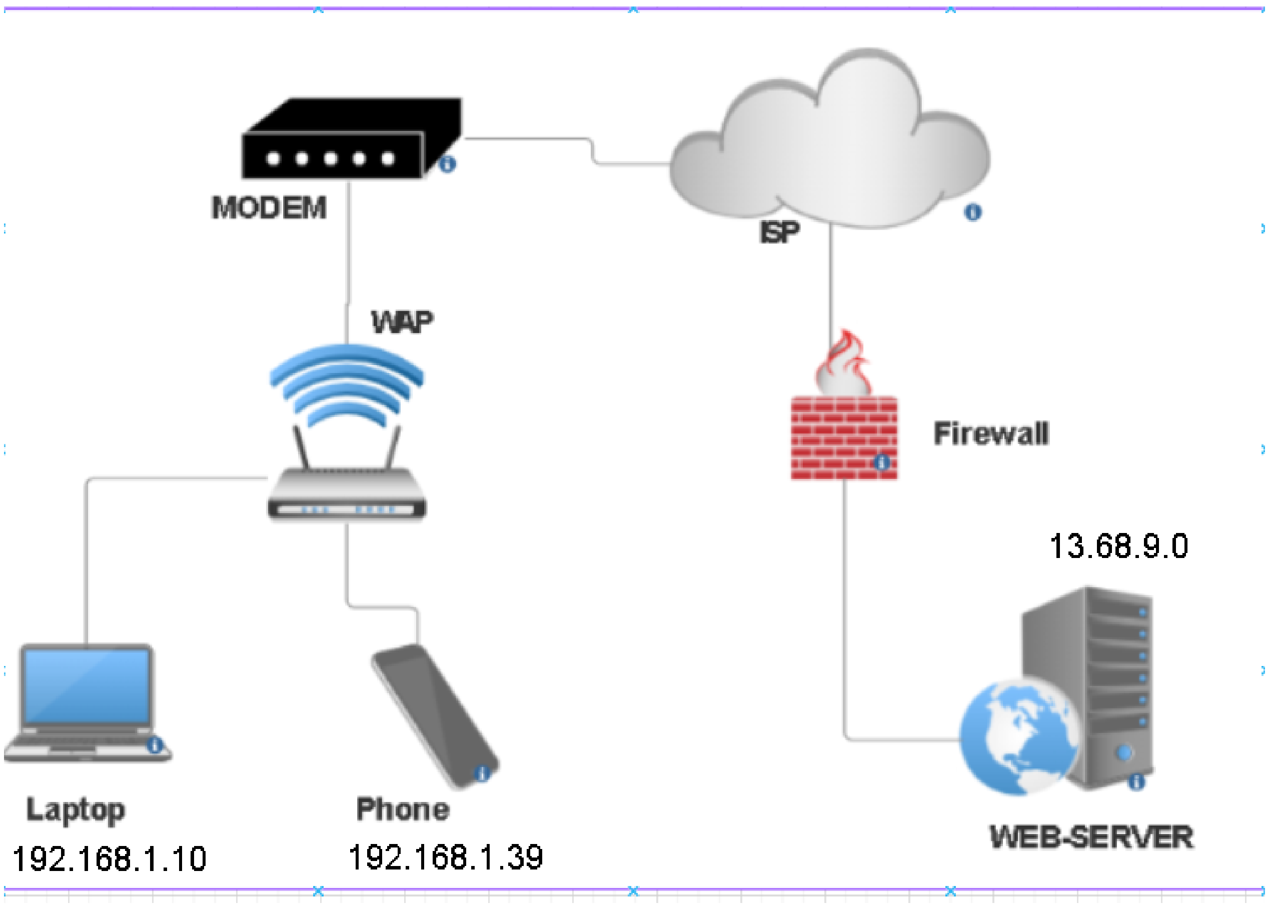


The process of deploying a web application with Azure App Service:

1. Converting existing web application to container.
2. Pushing container to Docker Hub or an Azure Container Registry.
3. App Service pulls image with credentials for Docker Hub or Azure Container Registry.
4. Accessing other Azure resources with service connectors.

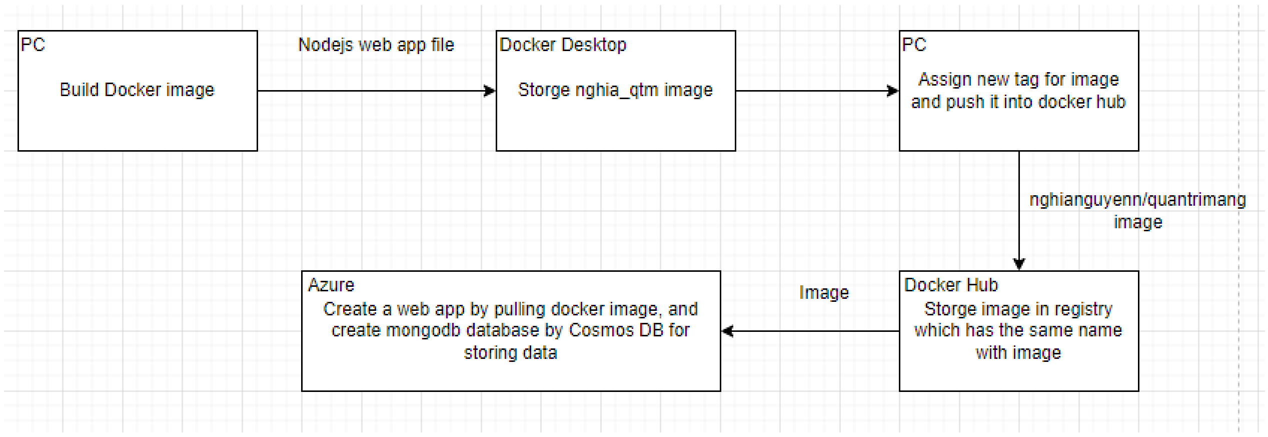
5. When developer pushes new image to the container registry, App Service updates are triggered.

CHAPTER 2: IMPLEMENTATION



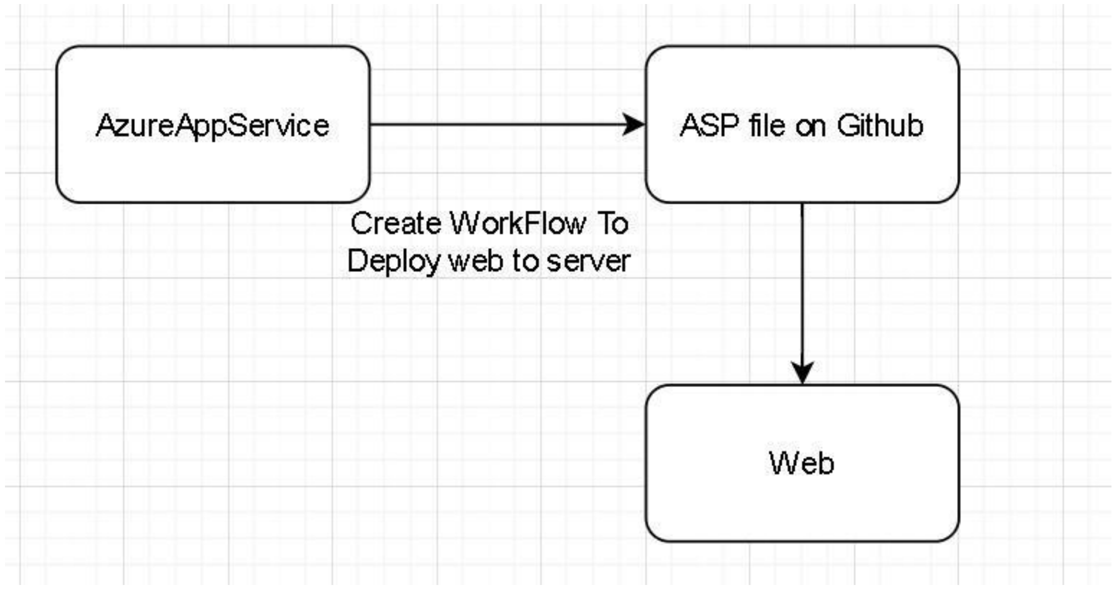
2.1 Topology

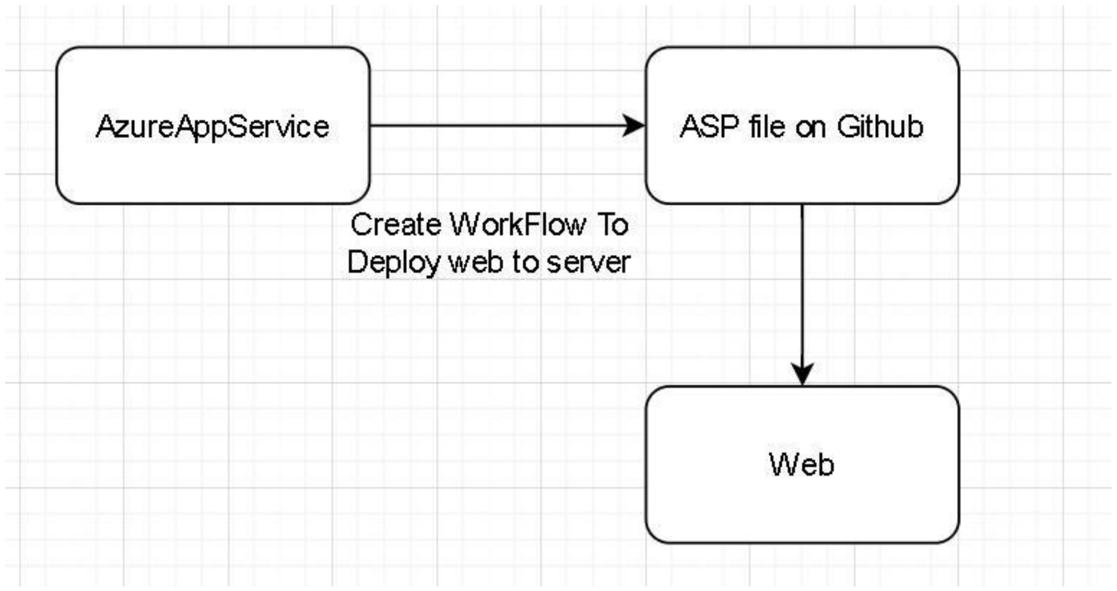
2.1.1 Docker


PC
Build Docker image
Nodejs web app file
Docker Desktop
Storge nghia_qtm image
PC
Assign new tag for image and push it into docker hub
nghianguyenn/quantrimang image
Docker Hub
Storge image in registry which has the same name with image
Image
Azure
Create a web app by pulling docker image, and create mongodb database by Cosmos DB for storing data

We build web with Nodejs and use that file to build image, image after finish build will push to our Docker hub. After that, in azure we pull that image to host a web app. So, that is the way we deploy our web with Docker for building image and azure for hosting.

2.1.2 Azure





Azure's Workflow Automation for Effortless Web Deployment

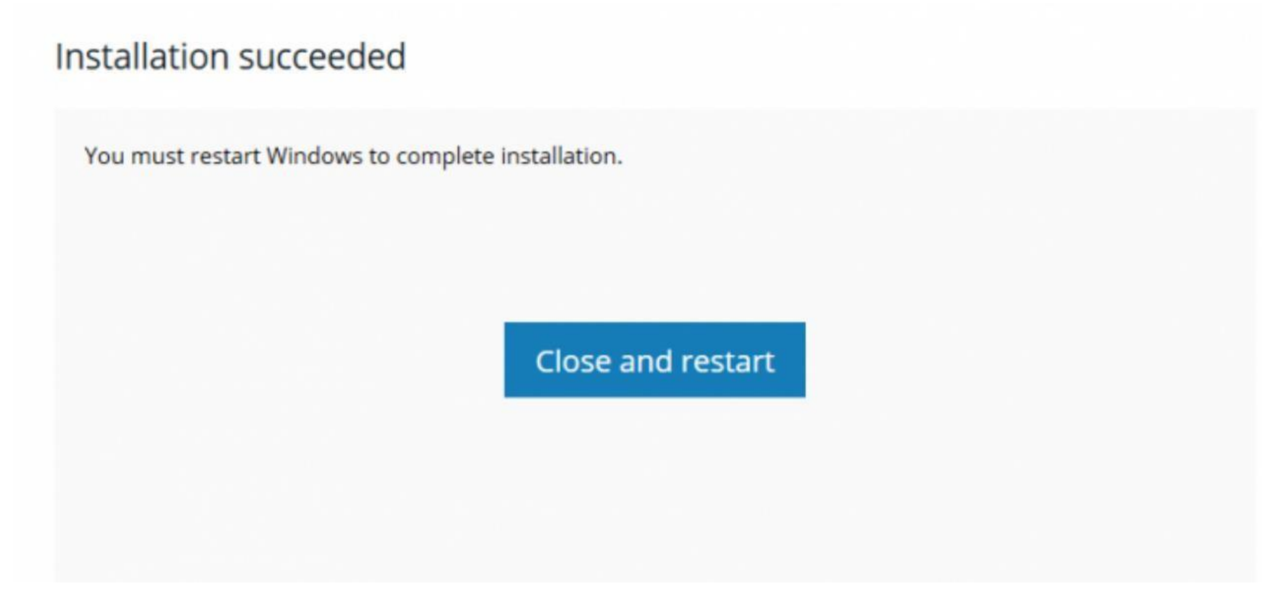
Azure simplifies web deployment through Azure DevOps and GitHub Actions. Create YAML-based workflows to automate website deployment from your GitHub repository to Azure. These workflows trigger automatically on code changes, ensuring efficient, error-free deployments. Azure's integration streamlines the process, saving time and resources, and guarantees consistent web application deployments.

2.2 Installation

2.2.1 Docker

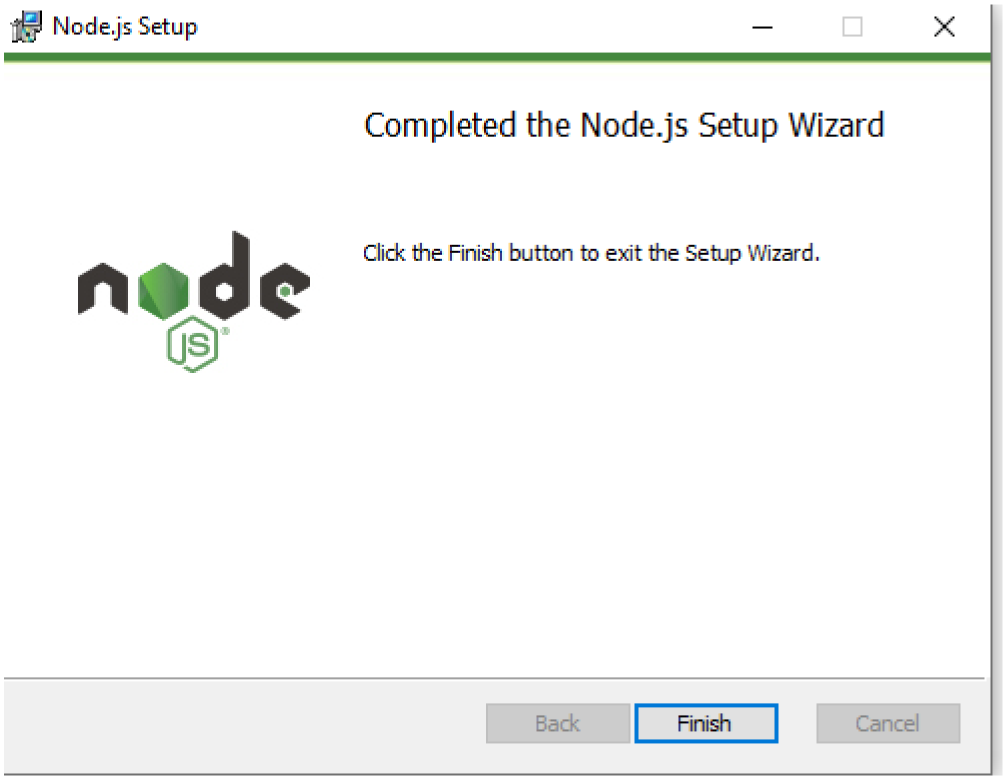
Of course, first I have to install Docker for my computer.

After installing successfully, the following display appears:


Installation succeeded

You must restart Windows to complete installation.

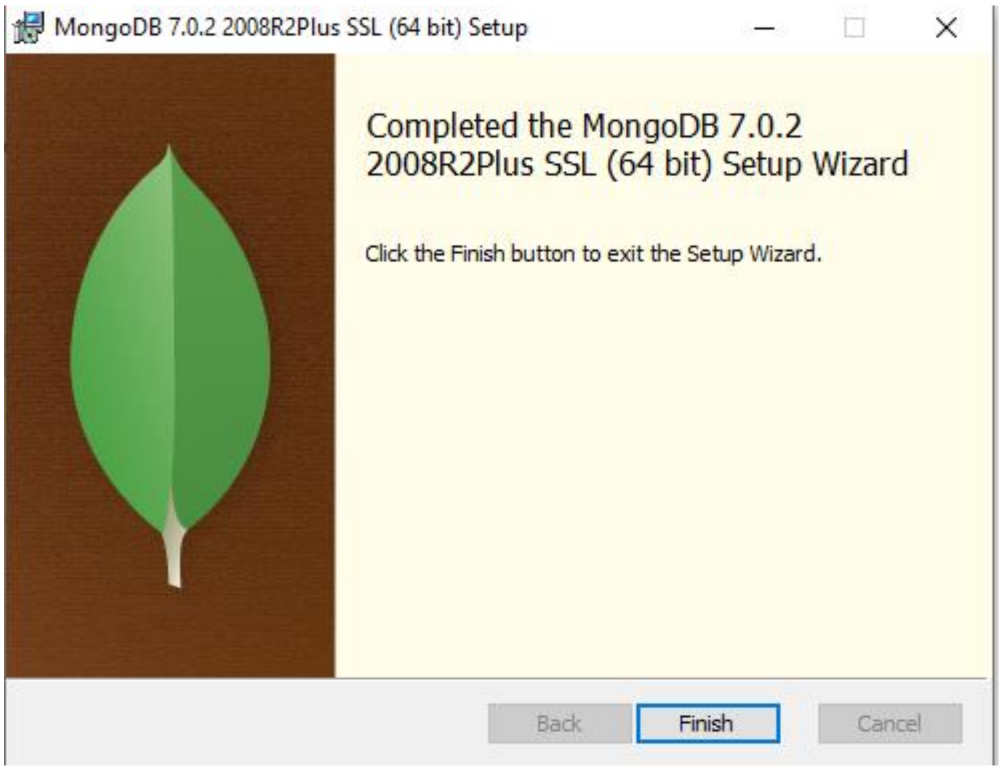
And then, because I use Nodejs, I have to install Nodejs too:



Completed the Node.js Setup Wizard

Click the Finish button to exit the Setup Wizard.

Next is MongoDB:


MongoDB 7.0.2 2008R2Plus SSL (64 bit) Setup

Completed the MongoDB 7.0.2 2008R2Plus SSL (64 bit) Setup Wizard

Click the Finish button to exit the Setup Wizard.

2.2.2 Azure

First, I sign up for Azure Portal.

As a student, I use Azure for Students, and with this platform, I can easily deploy my website.

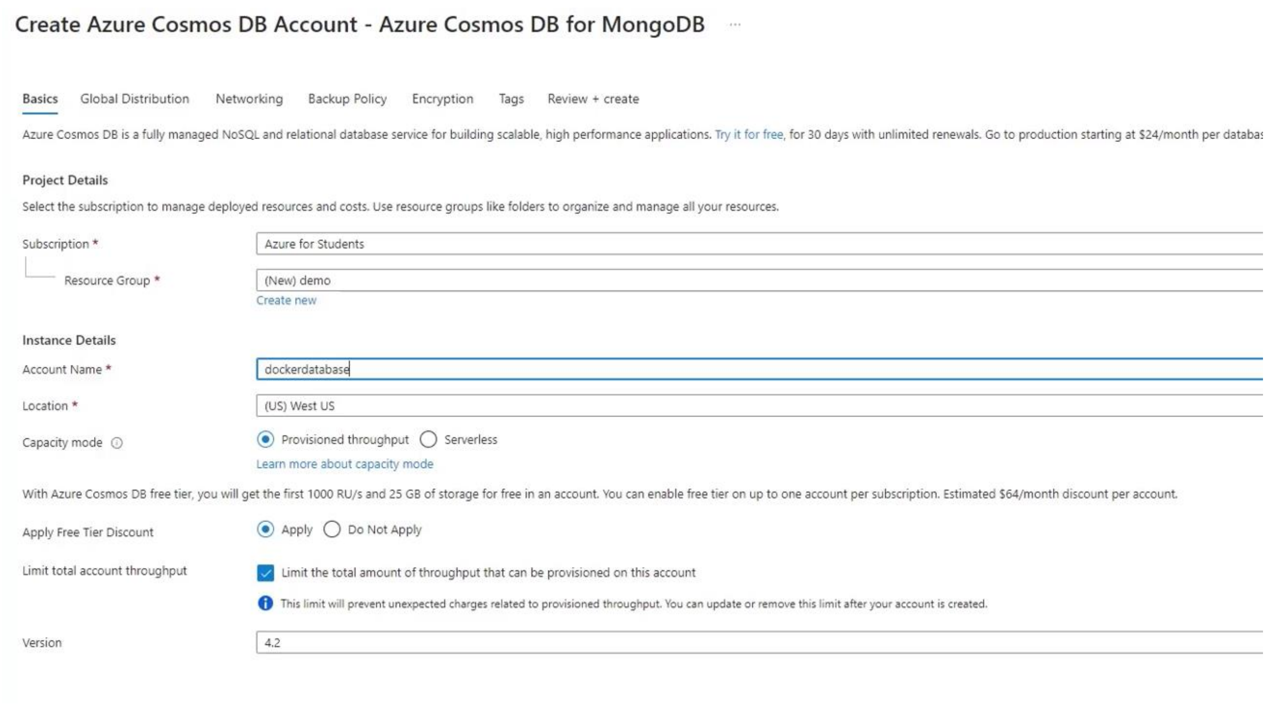
portal.azure.com/#@hcmuit.edu.vn/resource/subscriptions/67155ecc-4d9a-4868-b0b7-f153dc8e88a2/resourceGroups/DemoASP/providers/Microsoft.Web/sites/ASPtestt/troubleshoot

Microsoft Azure
Search resources, services, and docs (G+/)
21521848@ms.uit.edu...
TRƯỜNG ĐH CNTT - UNIVERSI...

2.3 Configuration

2.3.1 Mongo DB

We use Azure Cosmos DB for creating Mongo DB database:



Create Azure Cosmos DB Account - Azure Cosmos DB for MongoDB

Azure Cosmos DB is a fully managed NoSQL and relational database service for building scalable, high performance applications. Try it for free, for 30 days with unlimited renewals. Go to production starting at $24/month per databas

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Azure for Students
Resource Group *	(New) demo

Create new

Instance Details

Account Name *	dockerdatabase
Location *	(US) West US
Capacity mode	[✓ Provisioned throughput] Serverless

Learn more about capacity mode

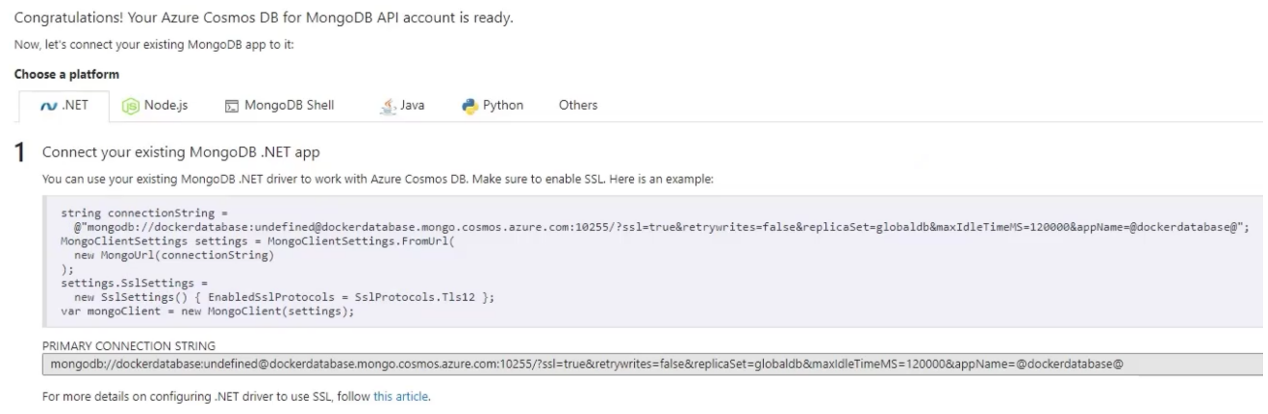
With Azure Cosmos DB free tier, you will get the first 1000 RU/s and 25 GB of storage for free in an account. You can enable free tier on up to one account per subscription. Estimated $64/month discount per account.

Apply Free Tier Discount	[✓ Apply] Do Not Apply
Limit total account throughput	✓ Limit the total amount of throughput that can be provisioned on this account

This limit will prevent unexpected charges related to provisioned throughput. You can update or remove this limit after your account is created.

Version	4.2

After creating successfully:



Congratulations! Your Azure Cosmos DB for MongoDB API account is ready.

Now, let's connect your existing MongoDB app to it:

Choose a platform

1 Connect your existing MongoDB .NET app

You can use your existing MongoDB .NET driver to work with Azure Cosmos DB. Make sure to enable SSL. Here is an example:

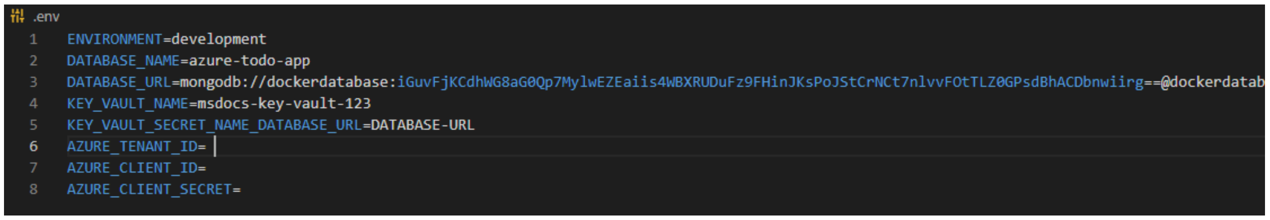
```
string connectionString =
  @"mongodb://dockerdatabase:undefined@dockerdatabase.mongo.cosmos.azure.com:10255/?ssl=true&retrywrites=false&replicaSet=globaldb&maxIdleTimeMS=120000&appName=@dockerdatabase@";
MongoClientSettings settings = MongoClientSettings.FromUrl(
  new MongoUrl(connectionString)
);
settings.SslSettings =
  new SslSettings() { EnabledSslProtocols = SslProtocols.Tls12 };
var mongoClient = new MongoClient(settings);
```

PRIMARY CONNECTION STRING

```
mongodb://dockerdatabase:undefined@dockerdatabase.mongo.cosmos.azure.com:10255/?ssl=true&retrywrites=false&replicaSet=globaldb&maxIdleTimeMS=120000&appName=@dockerdatabase@
```

For more details on configuring .NET driver to use SSL, follow this article.

We put the connection string in our code (line database_url):

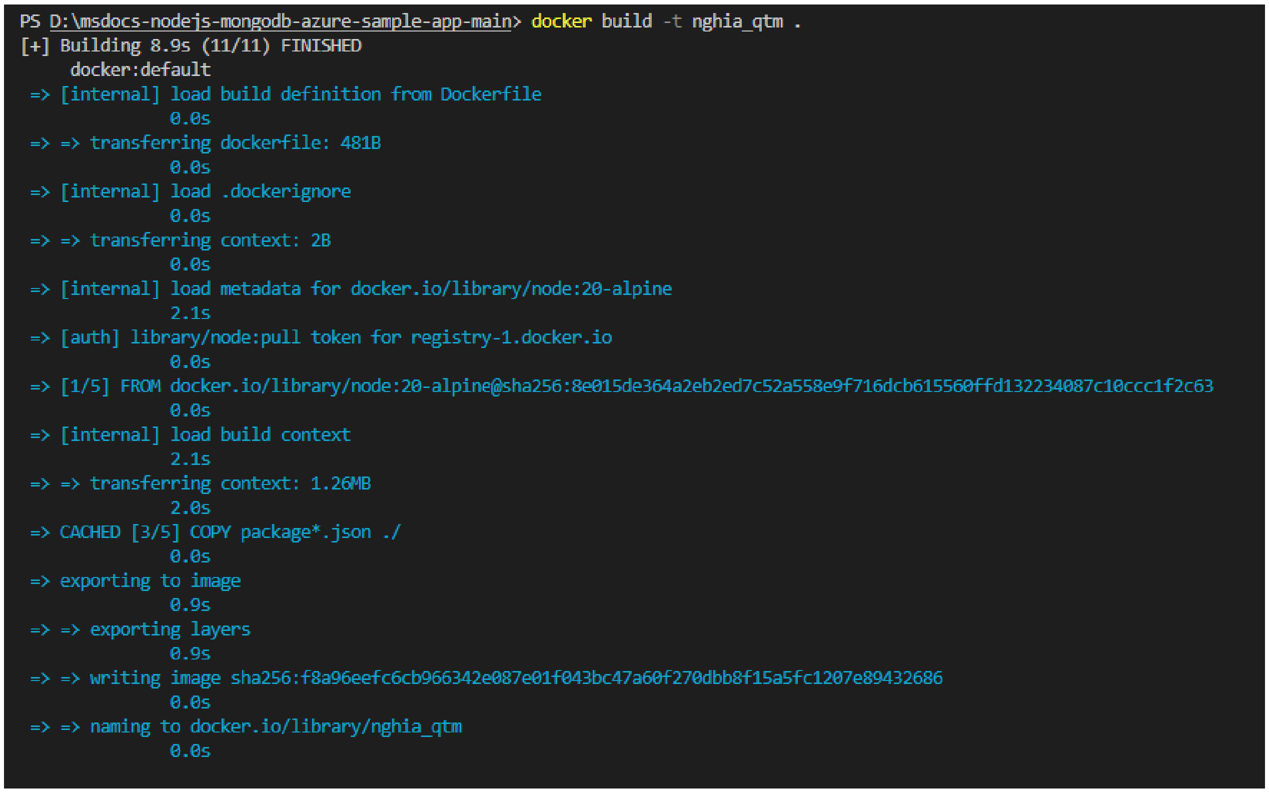


.env

```
ENVIRONMENT=development
DATABASE_NAME=azure-todo-app
DATABASE_URL=mongodb://dockerdatabase:iGuvFjKCdhWG8aG0Qp7MylwEZEaiis4WBXRUDuFz9FHinJKsPoJStCrNCt7nlvvFOtTLZ0GPsdBhACDbnwiirg==@dockerdatab
KEY_VAULT_NAME=msdocs-key-vault-123
KEY_VAULT_SECRET_NAME_DATABASE_URL=DATABASE-URL
AZURE_TENANT_ID=
AZURE_CLIENT_ID=
AZURE_CLIENT_SECRET=
```

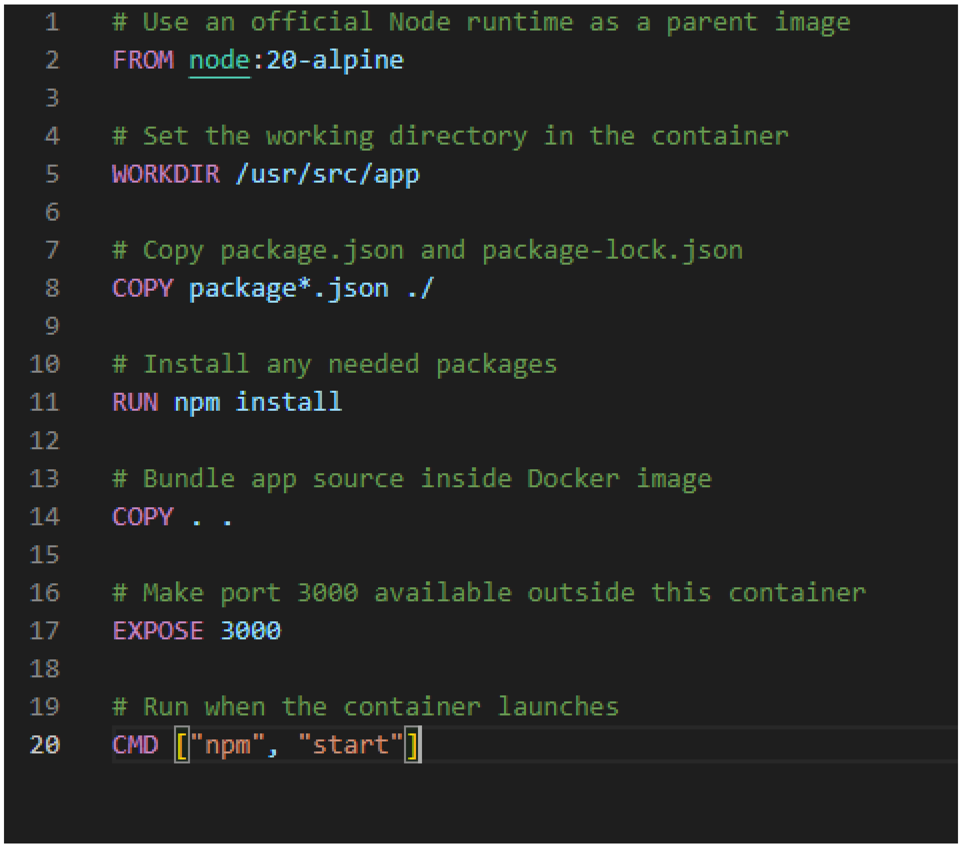
2.3.2 Docker

At the first, I build my Nodejs web into image that has name is nghia_qtm:



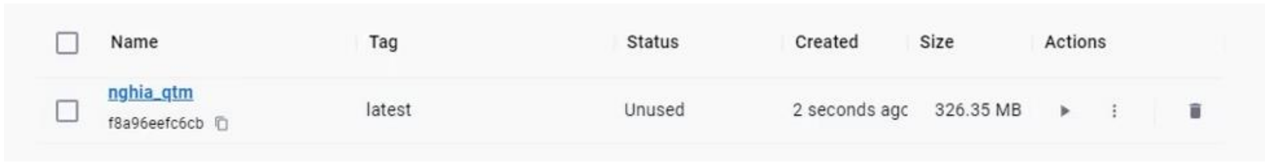
```
PS D:\msdocs-nodejs-mongodb-azure-sample-app-main> docker build -t nghia_qtm .
[+] Building 8.9s (11/11) FINISHED
     docker:default
 => [internal] load build definition from Dockerfile
               0.0s
 => => transferring dockerfile: 481B
               0.0s
 => [internal] load .dockerignore
               0.0s
 => => transferring context: 2B
               0.0s
 => [internal] load metadata for docker.io/library/node:20-alpine
               2.1s
 => [auth] library/node:pull token for registry-1.docker.io
               0.0s
 => [1/5] FROM docker.io/library/node:20-alpine@sha256:8e015de364a2eb2ed7c52a558e9f716dcb615560ffd132234087c10ccc1f2c63
               0.0s
 => [internal] load build context
               2.1s
 => => transferring context: 1.26MB
               2.0s
 => CACHED [3/5] COPY package*.json ./
               0.0s
 => exporting to image
               0.9s
 => => exporting layers
               0.9s
 => => writing image sha256:f8a96eefc6cb966342e087e01f043bc47a60f270dbb8f15a5fc1207e89432686
               0.0s
 => => naming to docker.io/library/nghia_qtm
               0.0s
```

In order to build that, must have following code (with explanation) in dockerfile:



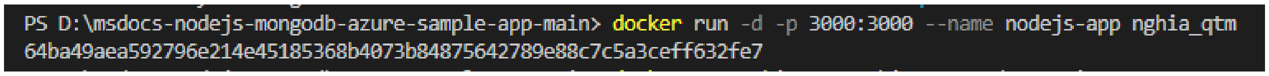
```dockerfile
# Use an official Node runtime as a parent image
FROM node:20-alpine

# Set the working directory in the container
WORKDIR /usr/src/app

# Copy package.json and package-lock.json
COPY package*.json ./

# Install any needed packages
RUN npm install

# Bundle app source inside Docker image
COPY . .

# Make port 3000 available outside this container
EXPOSE 3000

# Run when the container launches
CMD ["npm", "start"]
```

Here is the result:



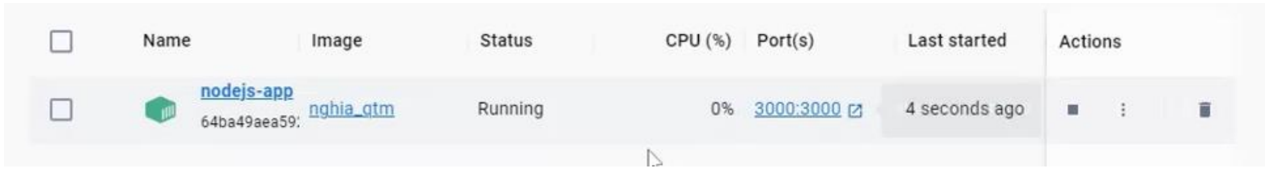
Name	Tag	Status	Created	Size	Actions
nghia_qtm f8a96eefc6cb	latest	Unused	2 seconds agc	326.35 MB	

Next, I test this image on local:

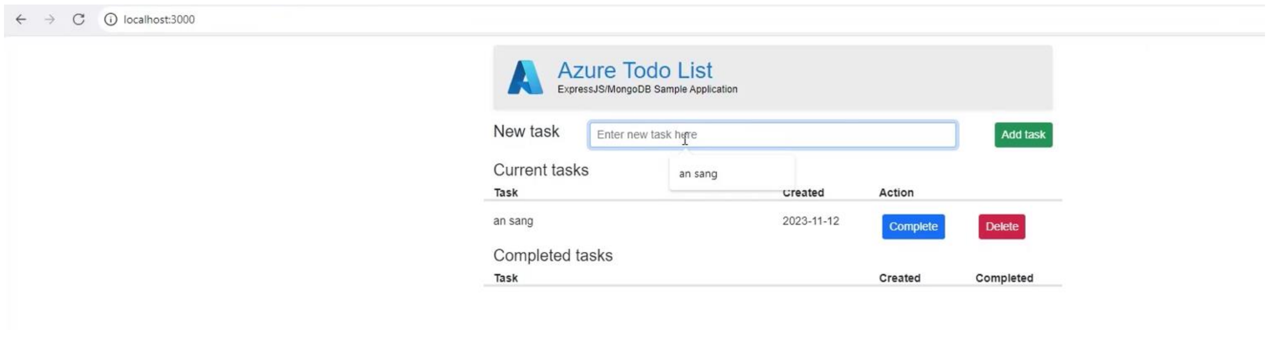


```
PS D:\msdocs-nodejs-mongodb-azure-sample-app-main> docker run -d -p 3000:3000 --name nodejs-app nghia_qtm
64ba49aea592796e214e45185368b4073b84875642789e88c7c5a3ceff632fe7
```

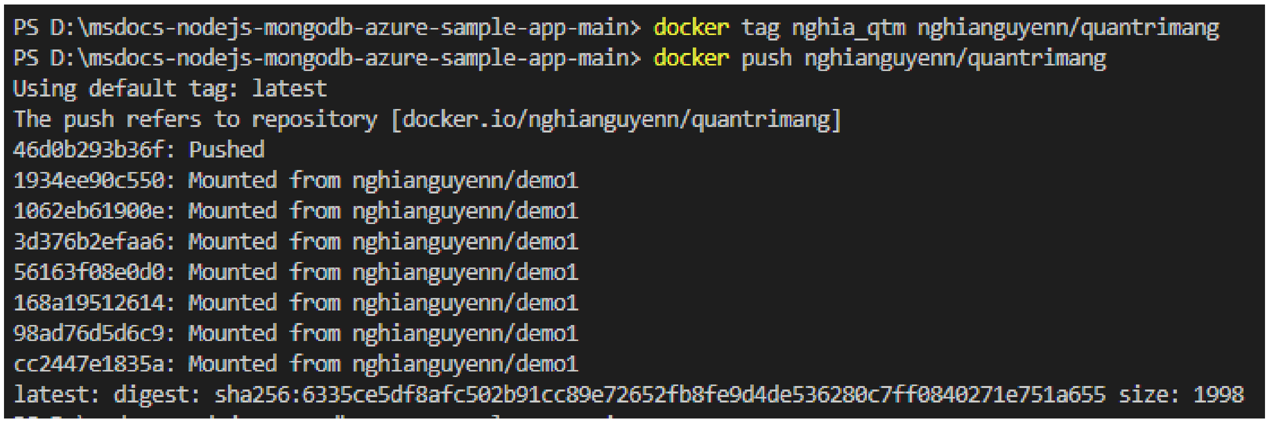
Here is the result:



Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
nodejs-app 64ba49aea59:	nghia_qtm	Running	0%	3000:3000	4 seconds ago	

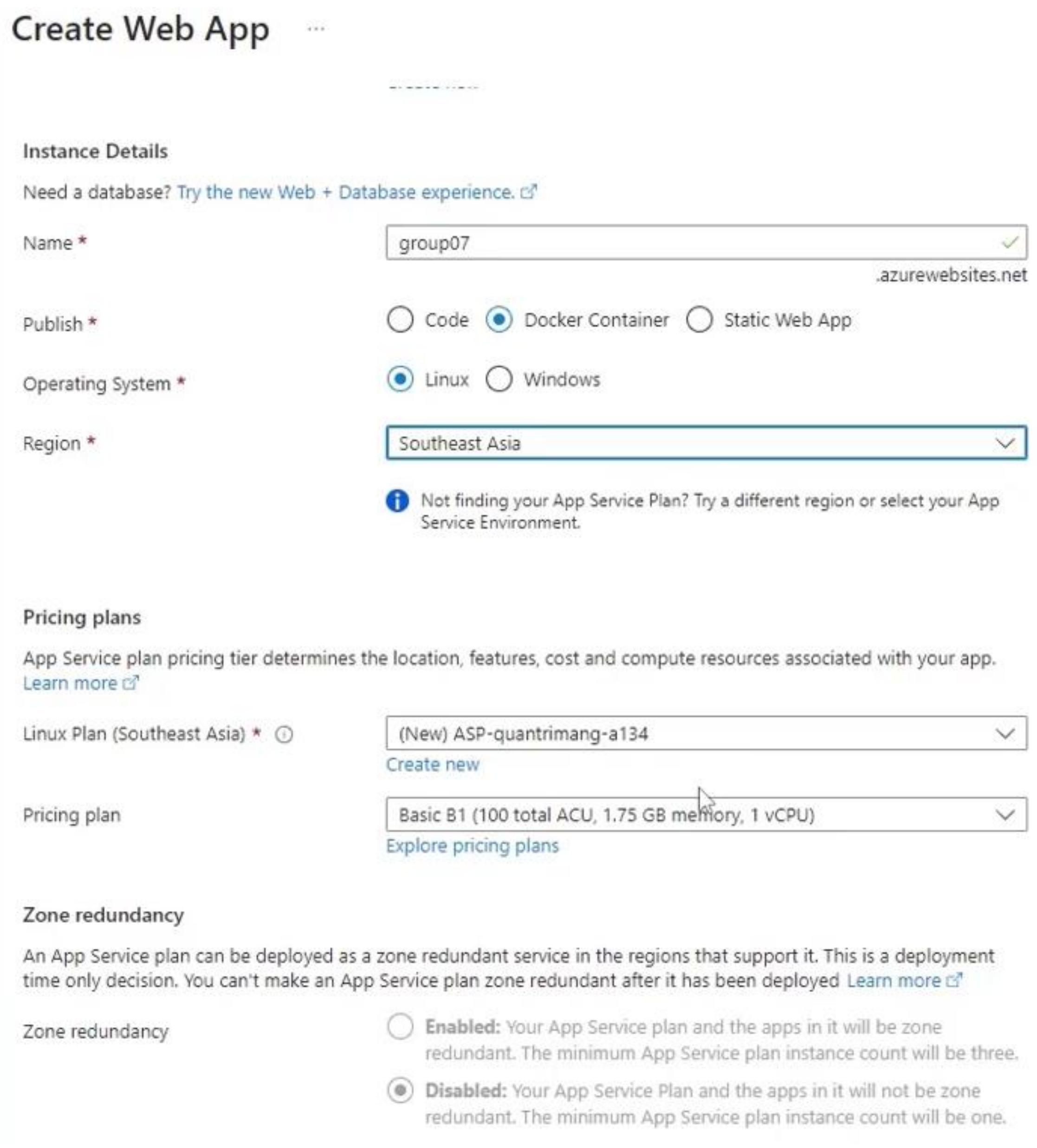


Then, pushing the image to my repository named nghianguyenn/quantrimang on Docker Hub:



```
PS D:\msdocs-nodejs-mongodb-azure-sample-app-main> docker tag nghia_qtm nghianguyenn/quantrimang
PS D:\msdocs-nodejs-mongodb-azure-sample-app-main> docker push nghianguyenn/quantrimang
Using default tag: latest
The push refers to repository [docker.io/nghianguyenn/quantrimang]
46d0b293b36f: Pushed
1934ee90c550: Mounted from nghianguyenn/demo1
1062eb61900e: Mounted from nghianguyenn/demo1
3d376b2efaa6: Mounted from nghianguyenn/demo1
56163f08e0d0: Mounted from nghianguyenn/demo1
168a19512614: Mounted from nghianguyenn/demo1
98ad76d5d6c9: Mounted from nghianguyenn/demo1
cc2447e1835a: Mounted from nghianguyenn/demo1
latest: digest: sha256:6335ce5df8afc502b91cc89e72652fb8fe9d4de536280c7ff0840271e751a655 size: 1998
```

After that, we use Azure App Service to deploy a web app:



Create Web App

Instance Details

Need a database? Try the new Web + Database experience.

Name *	group07 .azurewebsites.net
Publish *	Code [✓ Docker Container] Static Web App
Operating System *	[✓ Linux] Windows
Region *	Southeast Asia

Not finding your App Service Plan? Try a different region or select your App Service Environment.

Pricing plans

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. Learn more

Linux Plan (Southeast Asia) *	(New) ASP-quantrimang-a134
Pricing plan	Basic B1 (100 total ACU, 1.75 GB memory, 1 vCPU)

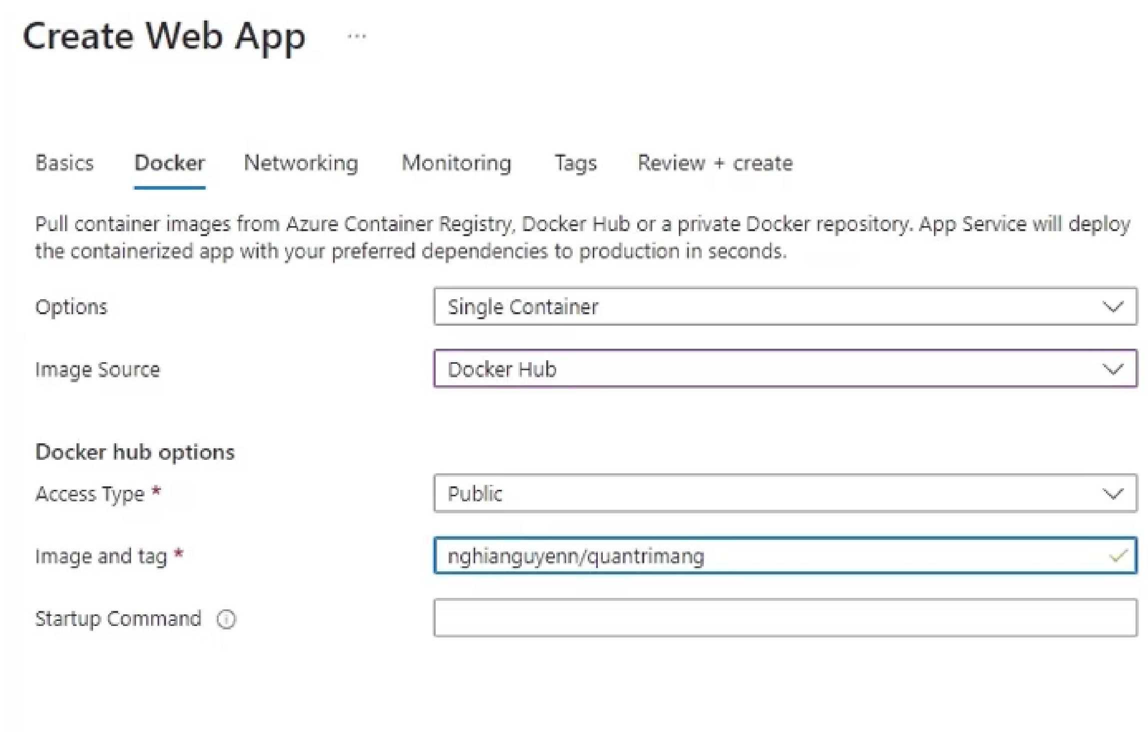
Zone redundancy

An App Service plan can be deployed as a zone redundant service in the regions that support it. This is a deployment time only decision. You can't make an App Service plan zone redundant after it has been deployed Learn more

Zone redundancy

Enabled: Your App Service plan and the apps in it will be zone redundant. The minimum App Service plan instance count will be three.

[✓ Disabled: Your App Service Plan and the apps in it will not be zone redundant. The minimum App Service plan instance count will be one.]


Create Web App

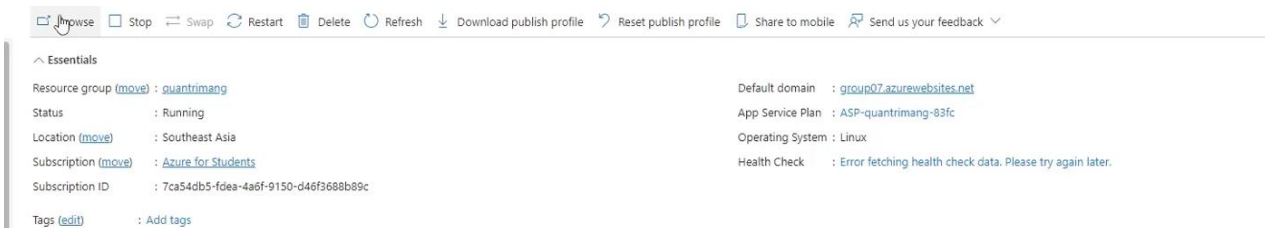
Pull container images from Azure Container Registry, Docker Hub or a private Docker repository. App Service will deploy the containerized app with your preferred dependencies to production in seconds.

Options	Single Container
Image Source	Docker Hub

Docker hub options

Access Type *	Public
Image and tag *	nghianguyenn/quantrimang
Startup Command	

Setup successfully:

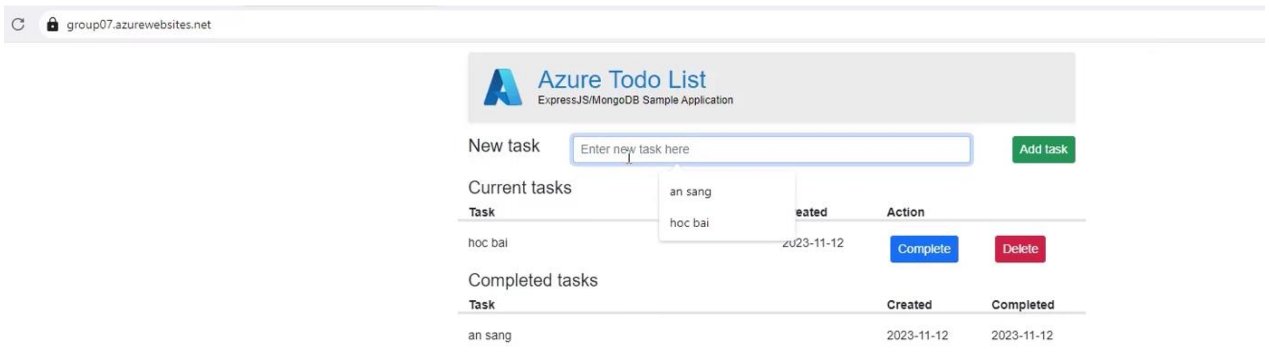


Essentials

Resource group (move)	: quantrimang
Status	: Running
Location (move)	: Southeast Asia
Subscription (move)	: Azure for Students
Subscription ID	: 7ca54db5-fdea-4a6f-9150-d46f3688b89c
Tags (edit)	: Add tags
Default domain	: group07.azurewebsites.net
App Service Plan	: ASP-quantrimang-83fc
Operating System	: Linux
Health Check	: Error fetching health check data. Please try again later.

Finally, I access to my web through url: group07.azurewebsites.net

Here is the result:



Current tasks

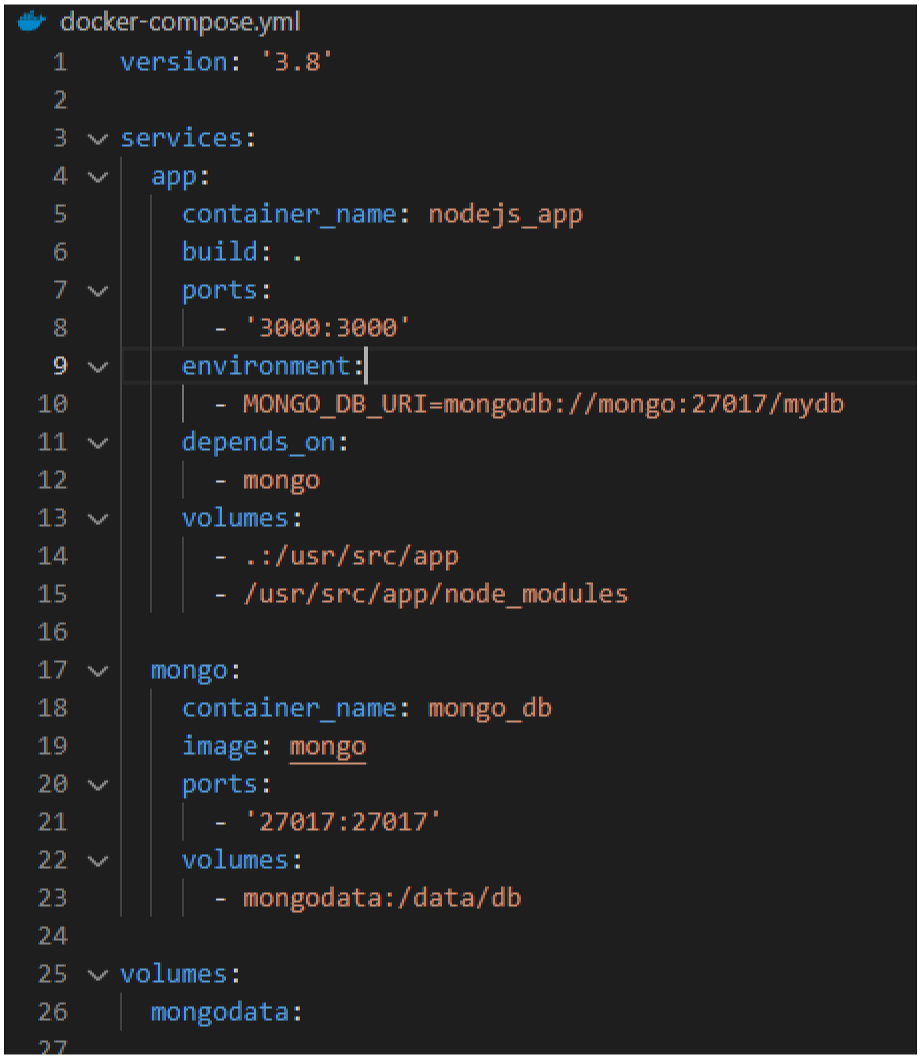
Task	Created	Action
hoc bai	2023-11-12	

Completed tasks

Task	Created	Completed
an sang	2023-11-12	2023-11-12

2.3.3 Docker Compose

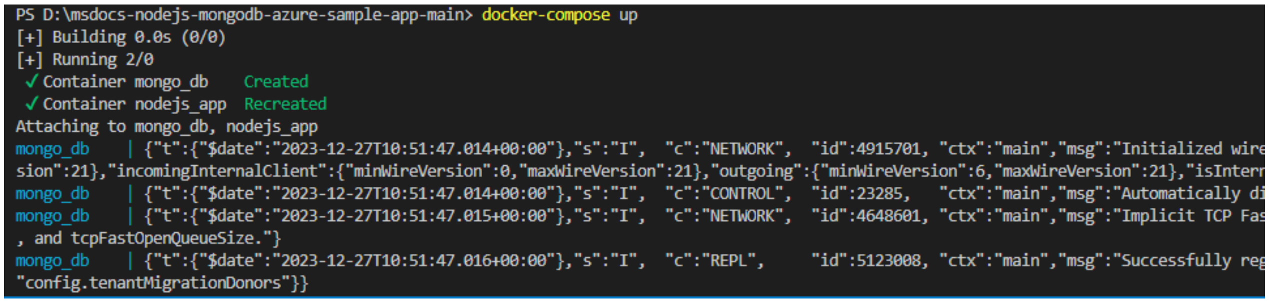
Here is the code for Docker compose file:



docker-compose.yml

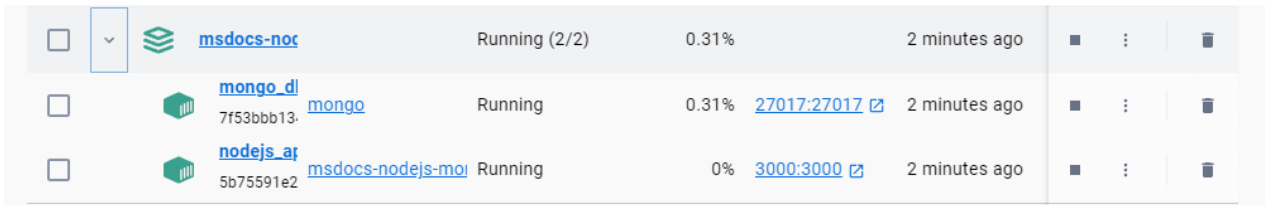
```yaml
version: '3.8'

services:
  app:
    container_name: nodejs_app
    build: .
    ports:
      - '3000:3000'
    environment:
      - MONGO_DB_URI=mongodb://mongo:27017/mydb
    depends_on:
      - mongo
    volumes:
      - .:/usr/src/app
      - /usr/src/app/node_modules

  mongo:
    container_name: mongo_db
    image: mongo
    ports:
      - '27017:27017'
    volumes:
      - mongodata:/data/db

volumes:
  mongodata:
```

Using command “docker compose up” to run Docker compose:

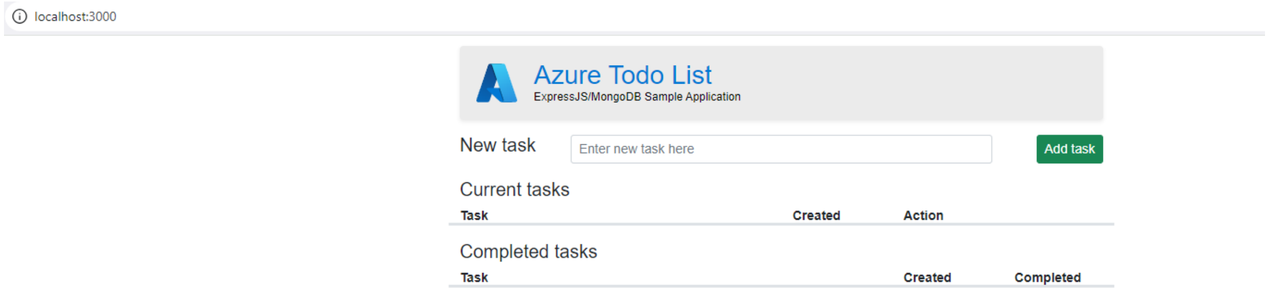


```
PS D:\msdocs-nodejs-mongodb-azure-sample-app-main> docker-compose up
[+] Building 0.0s (0/0)
[+] Running 2/0
 ✓ Container mongo_db     Created
 ✓ Container nodejs_app   Recreated
Attaching to mongo_db, nodejs_app
mongo_db    | {"t":{"$date":"2023-12-27T10:51:47.014+00:00"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"main","msg":"Initialized wire
sion":21},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":21},"outgoing":{"minWireVersion":6,"maxWireVersion":21},"isIntern
mongo_db    | {"t":{"$date":"2023-12-27T10:51:47.014+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automatically di
mongo_db    | {"t":{"$date":"2023-12-27T10:51:47.015+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main","msg":"Implicit TCP Fas
, and tcpFastOpenQueueSize."}
mongo_db    | {"t":{"$date":"2023-12-27T10:51:47.016+00:00"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"main","msg":"Successfully reg
"config.tenantMigrationDonors"}}
```

Here’s our result:



msdocs-noc		Running (2/2)	0.31%		2 minutes ago	
mongo_dl 7f53bbb13	mongo	Running	0.31%	27017:27017	2 minutes ago	
nodejs_ap 5b75591e2	msdocs-nodejs-mo	Running	0%	3000:3000	2 minutes ago	



localhost:3000

Azure Todo List

ExpressJS/MongoDB Sample Application

New task: Enter new task here — Add task

Current tasks

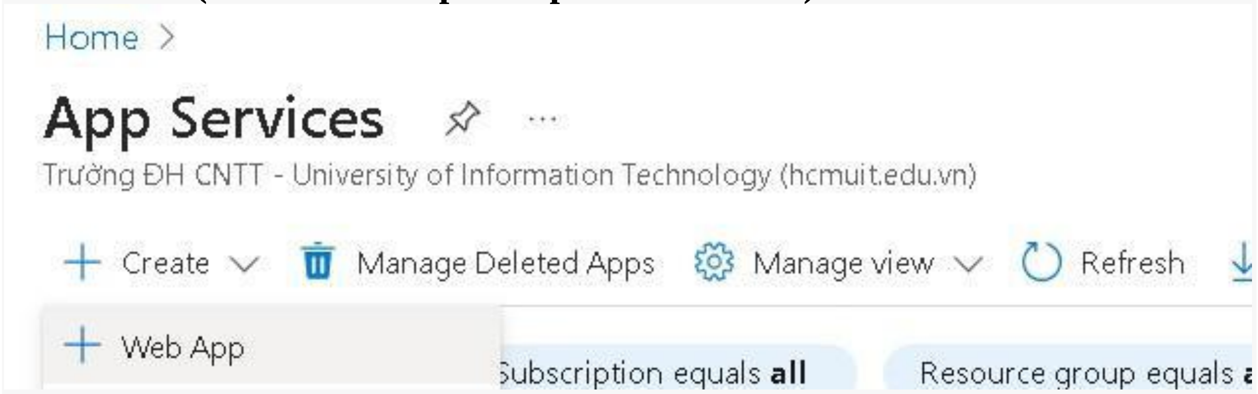
Task	Created	Action

Completed tasks

Task	Created	Completed

But it can only run local, we can't deploy it into azure

2.3.4 Azure (This is the old part reported midterm)

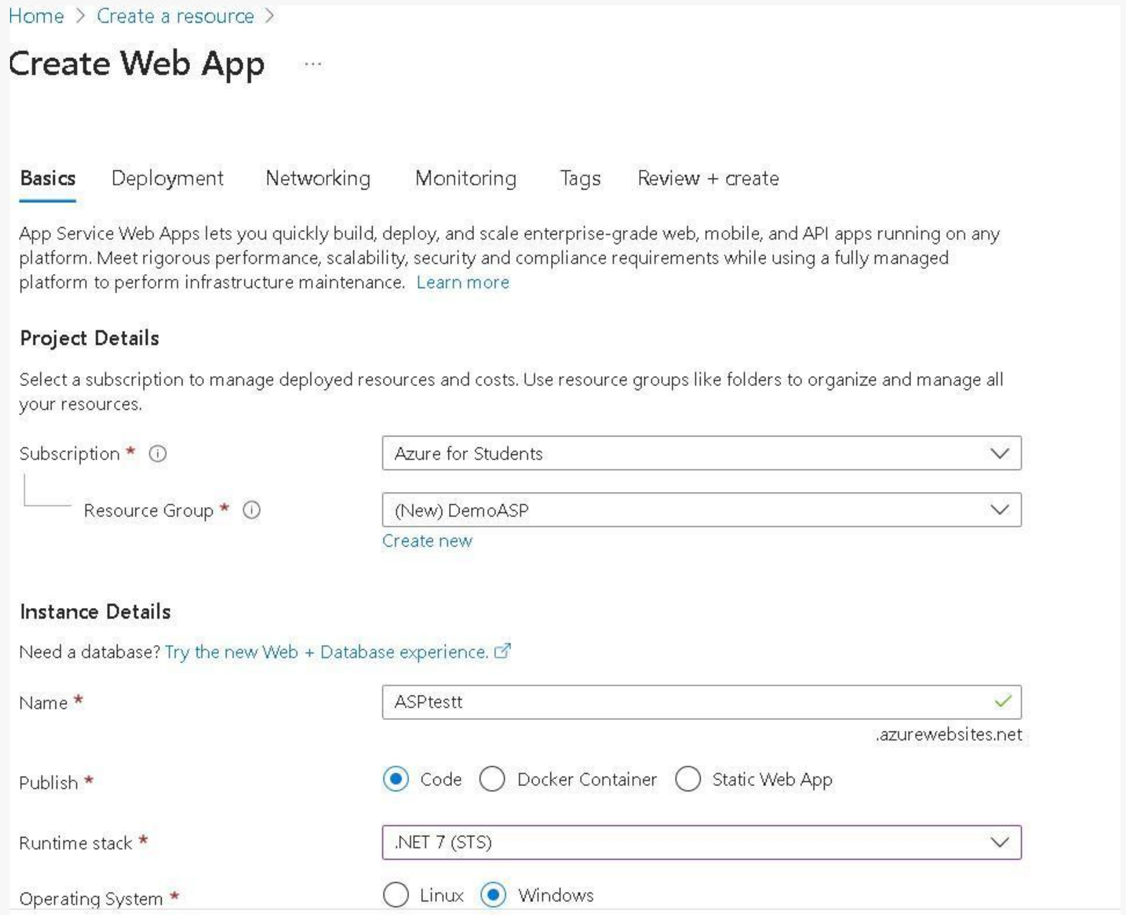


Step 1: Choose App Service and Create Web App

Step 2: Configuration

I am using a web application developed with ASP.NET Core so I need to use .NET runtime stack 7.0 and publish code.

I am using a web application developed with ASP.NET Core



Home > Create a resource >

Create Web App

Basics | Deployment | Networking | Monitoring | Tags | Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. Learn more

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *: Azure for Students

Resource Group *: (New) DemoASP

Create new

Instance Details

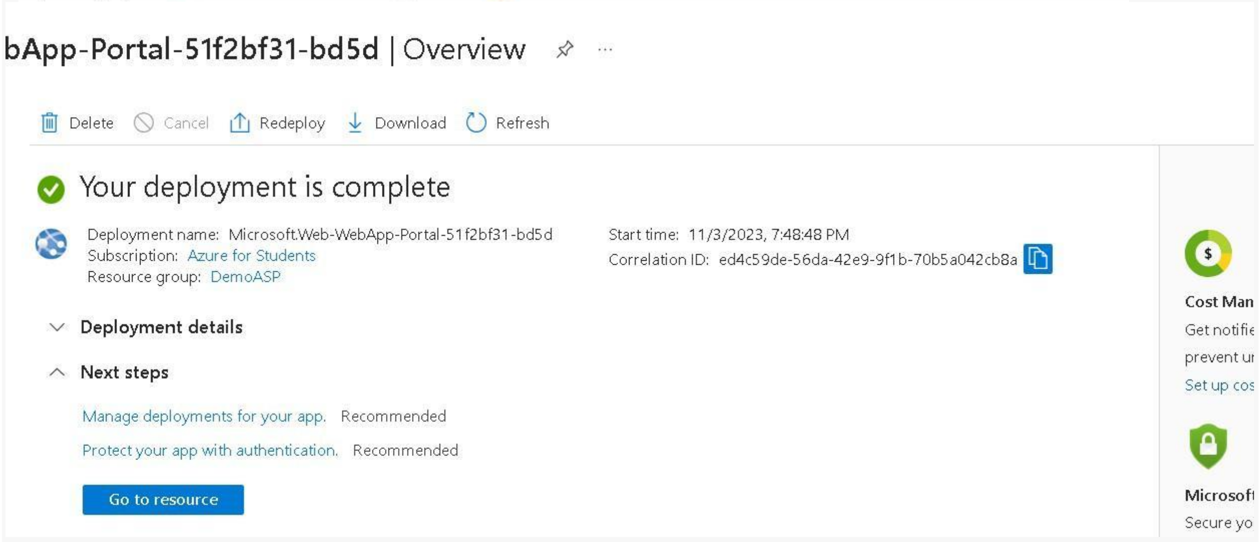
Need a database? Try the new Web + Database experience.

Name *: ASPtestt .azurewebsites.net

Publish *: [✓ Code] Docker Container Static Web App

Runtime stack *: .NET 7 (STS)

Operating System *: Linux [✓ Windows]



bApp-Portal-51f2bf31-bd5d | Overview

Delete Cancel Redeploy Download Refresh

Your deployment is complete

Deployment name: Microsoft.Web-WebApp-Portal-51f2bf31-bd5d
Subscription: Azure for Students
Resource group: DemoASP

Start time: 11/3/2023, 7:48:48 PM
Correlation ID: ed4c59de-56da-42e9-9f1b-70b5a042cb8a

Deployment details

Next steps

Manage deployments for your app. Recommended

Protect your app with authentication. Recommended

Go to resource

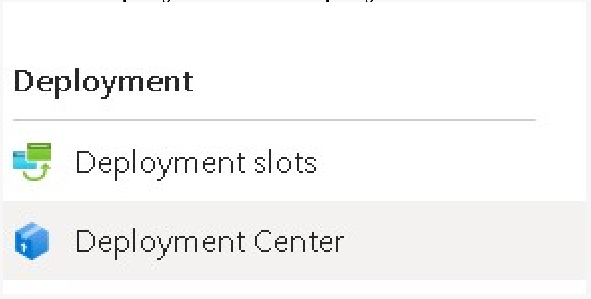
Cost Man
Get notifie
prevent ur
Set up cos

Microsoft
Secure yo

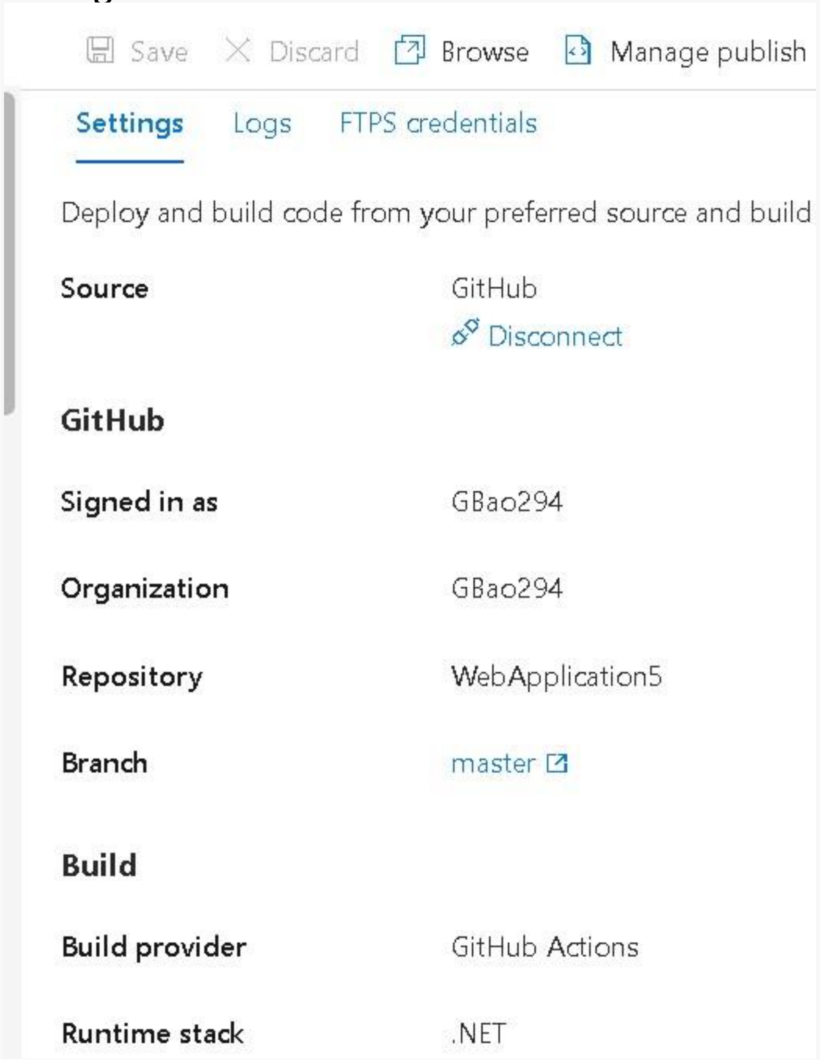
After creating, we get

⇨ **Now we have a task to deploy our web**

Step 3: Push web to server
Go to Deployment -> Deployment Center



Configuration:



Deploy and build code from your preferred source and build

Source: GitHub

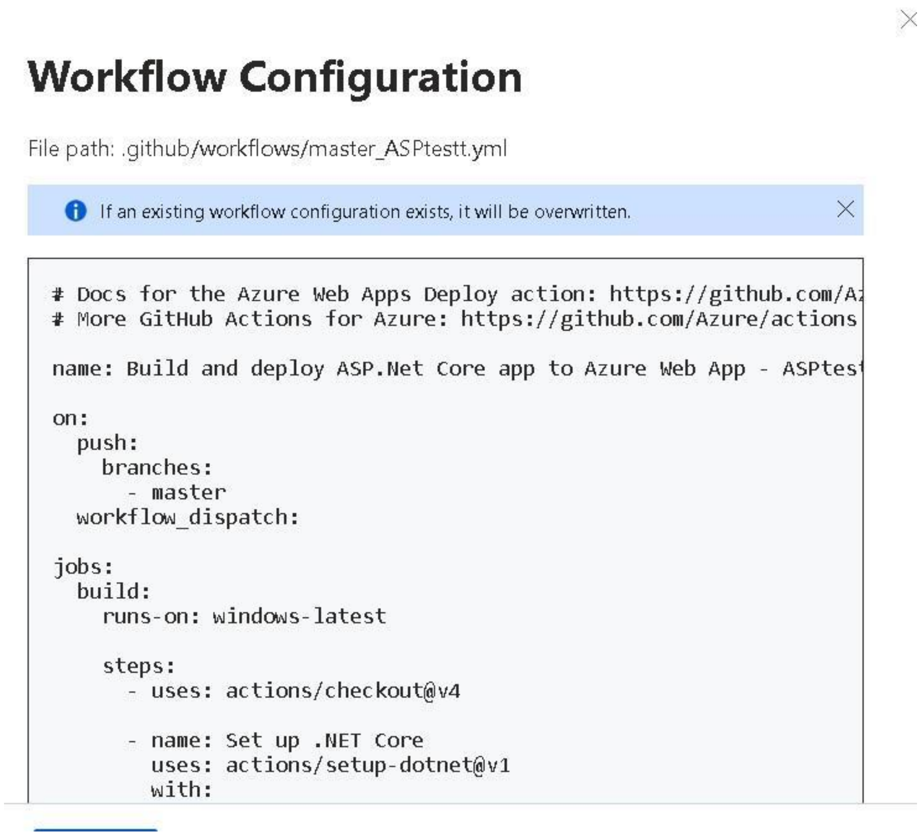
GitHub

Signed in as	GBao294
Organization	GBao294
Repository	WebApplication5
Branch	master

Build

Build provider	GitHub Actions
Runtime stack	.NET

Save And Waiting, at this step Azure will create a YAML file workflow



File path: .github/workflows/master_ASPtestt.yml

If an existing workflow configuration exists, it will be overwritten.

```
# Docs for the Azure Web Apps Deploy action: https://github.com/Az
# More GitHub Actions for Azure: https://github.com/Azure/actions

name: Build and deploy ASP.Net Core app to Azure Web App - ASPtes

on:
  push:
    branches:
      - master
  workflow_dispatch:

jobs:
  build:
    runs-on: windows-latest

    steps:
      - uses: actions/checkout@v4

      - name: Set up .NET Core
        uses: actions/setup-dotnet@v1
        with:
```

Explain:

Certainly, here's a simplified summary of what the workflow does:

1. **Build Job: **

- It runs on a Windows environment.

- Checks out the code from the repository.

- Sets up the .NET Core environment.

- Builds and publishes the ASP.NET Core application.

- Uploads the published artifacts.

2. **Deploy Job: **

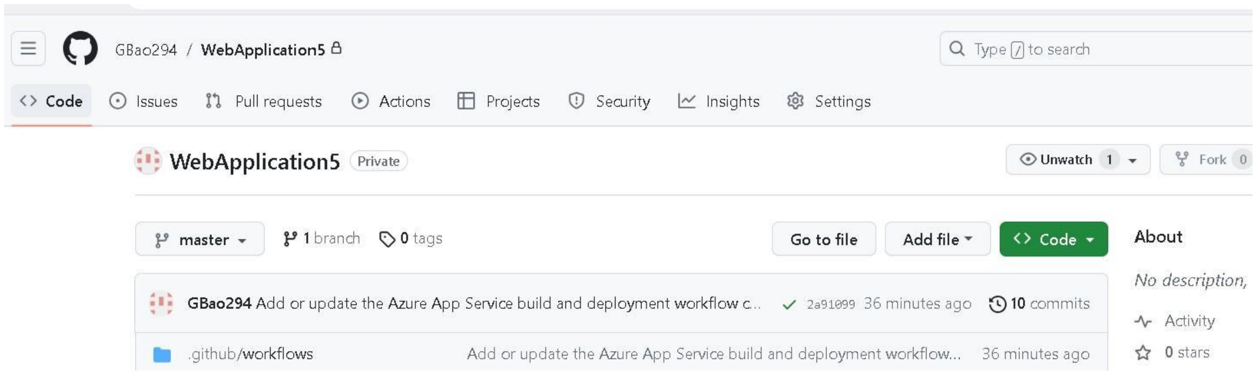
- Also runs on a Windows environment.

- Depends on the successful completion of the Build Job.

- Downloads the previously uploaded artifacts.

- Deploys the ASP.NET Core application to an Azure Web App, using a specified Azure deployment slot and publish profile.

In essence, this workflow automates the build and deployment process of an ASP.NET Core application to an Azure Web App whenever changes are pushed to the 'master' branch or manually triggered.

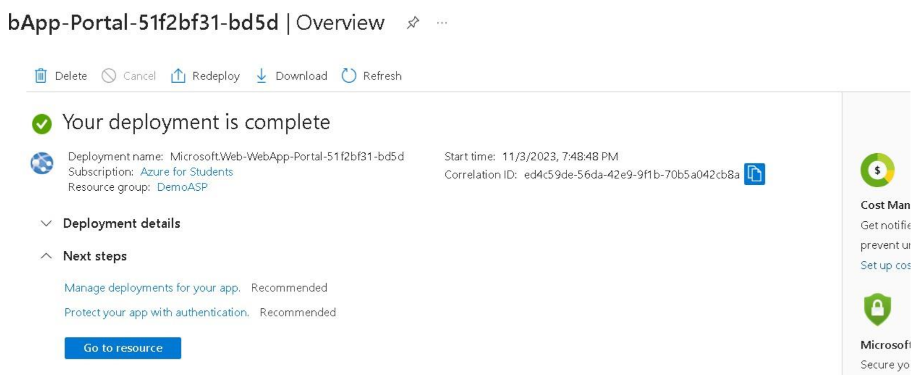


GBao294 Add or update the Azure App Service build and deployment workflow c... ✓ 2a91099 36 minutes ago 10 commits

.github/workflows Add or update the Azure App Service build and deployment workflow... 36 minutes ago

LINK GITHUB : https://github.com/GBao294/WebApplication5

Result:

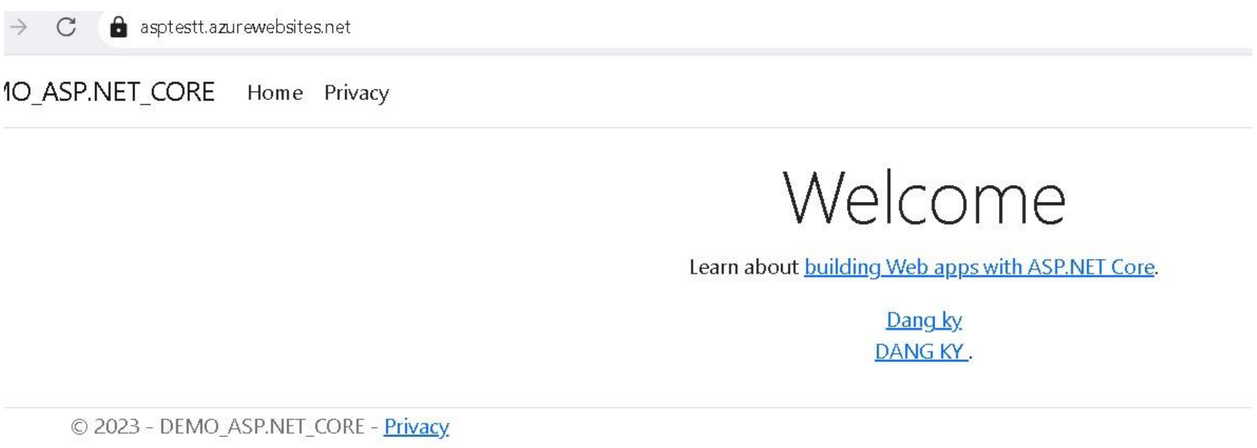


Your deployment is complete

Deployment name: Microsoft.Web-WebApp-Portal-51f2bf31-bd5d
Subscription: Azure for Students
Resource group: DemoASP

Start time: 11/3/2023, 7:48:48 PM
Correlation ID: ed4c59de-56da-42e9-9f1b-70b5a042cb8a

Manage deployments for your app. Recommended
Protect your app with authentication. Recommended



Welcome

Learn about building Web apps with ASP.NET Core.

Dang ky
DANG KY .

© 2023 - DEMO_ASP.NET_CORE - Privacy

Link web: https://asptestt.azurewebsites.net/

This is a very basic asp.net web app for demo deploy, in the next report we will change to a full functional web with frontend, backend and database.

CHAPTER 3: RESULT

(Watch in video)

APPENDIX

Self-evaluation

	Point
Report format	0.75
Presentation	0.75
Theory	1.5
Demonstration	4.25

Task assignment

Member	**Task**	**Percent Complete**
Pham Hoang Phuc	Research information about components and operation of Docker and Azure (1.1 -> 1.3)	**100%**
Nguyen Dai Nghia	Implement Docker and using Azure service to deploy (2.2, 2.3)	**90%**
Hoang Gia Bao	Research features of Azure, Docker	**100%**

Q&A

Question	**Answer**
Is there a limit to how many image the Docker hub can hold	Không có giới hạn cụ thể về số lượng hình ảnh mà Docker Hub có thể lưu trữ. Tuy nhiên, có các hạn chế

	về dung lượng tổng cộng cho mỗi tài khoản người dùng
Mình không hiểu về components	Components trong ngữ cảnh Docker : bao gồm các container, images, networks, volumes, và các thành phần khác.
So is Docker Hub similar to GitHub and what's the big difference between the two?	Docker Hub và GitHub phục vụ mục đích khác nhau. Docker Hub là nơi lưu trữ và chia sẻ container images, trong khi GitHub chủ yếu dành cho quản lý mã nguồn và phiên bản của dự án
Nếu host nhiều website trong 1 hệ thống thì làm thế nào để các Docker có thể giao tiếp với nhau?	Tham khảo Docker Network
Tại sao Docker lại deploy nhanh hơn?	Docker triển khai nhanh hơn nhờ vào việc sử dụng containerization, giúp giảm thời gian cài đặt và cấu hình môi trường, cũng như khả năng chia sẻ các layer giữa các images.
How many services can Docker run at one time?	Docker có thể chạy nhiều dịch vụ (services) cùng một lúc, tùy thuộc vào tài nguyên hệ thống và cấu hình của container.
Which installation scenario of Docker Compose did you use and why? List the difficulties when you install it.	Scenario: Installing Docker Compose via Package Managers. No difficult
Làm sao để truy cập vào database đã tạo?	Tạo 1 database riêng trên azure và docker sử dụng database đó thông qua code được configure sẵn trong backend
Cách thức tích hợp bảo mật và quản lý quyền truy cập cho ứng dụng web	Đối với Docker, có thể sử dụng các giải pháp như Docker Security Scanning và quản lý quyền truy cập thông qua cấu hình container và môi trường. Tuy nhiên khi deploy trên azure bạn có thể dùng thêm các công cụ giám sát có sẵn của azure