

# Exercises: Deployment to Cloud

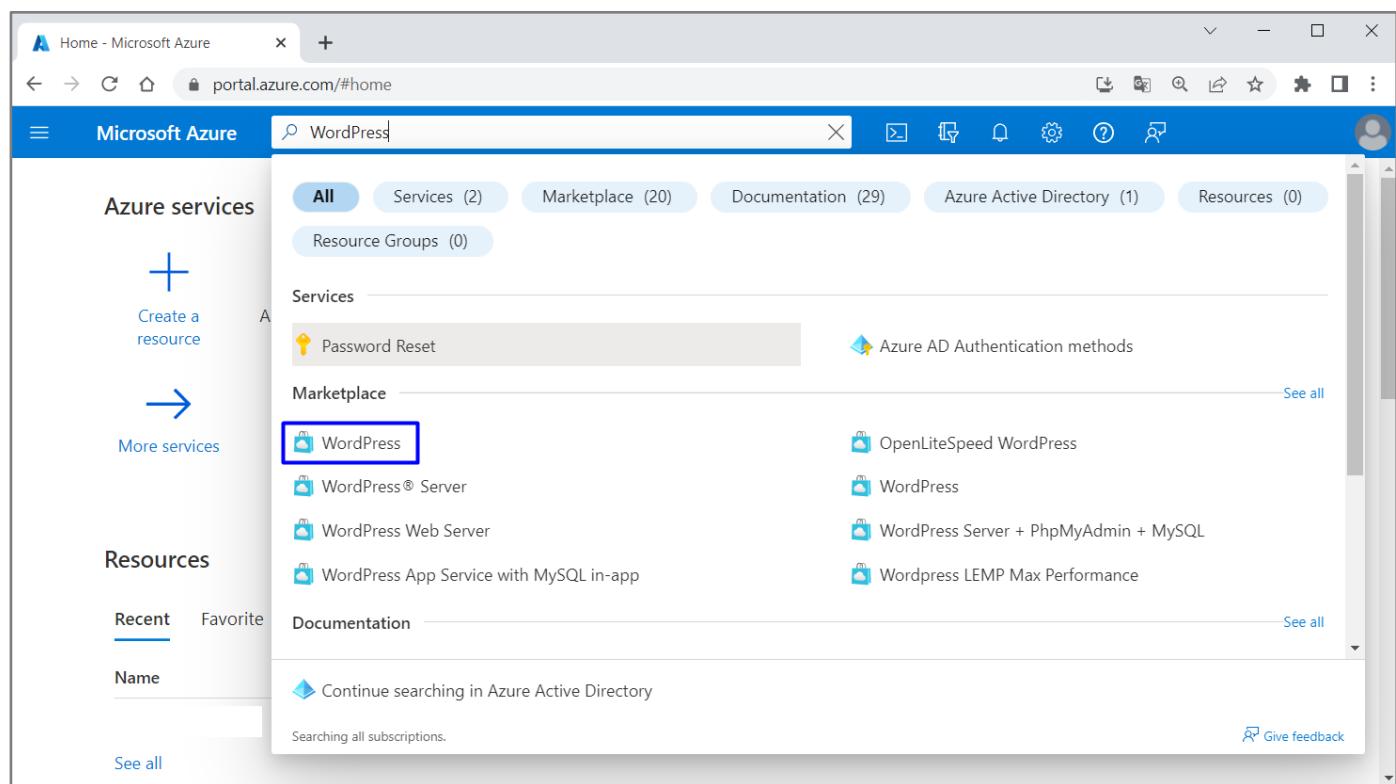
Exercises for the "[Containers and Clouds](#)" course @ SoftUni

## 1. Create WordPress Site Using Azure Portal

Our task now is to create a **WordPress site in Azure**. WordPress can be run on a **few different Azure services**: AKS, Virtual Machines, and App Service. Let's **deploy our first WordPress site to Azure App Service with Azure Database for MySQL – Flexible Server**.

### Step 1: Create the Site

Go to **Azure Portal** on <https://portal.azure.com> and **log in**. Then, in the **search box**, type "**WordPress**" and select **[WordPress]** from the **Marketplace**:



The screenshot shows the Microsoft Azure portal interface. The search bar at the top contains the text "WordPress". Below the search bar, there are several tabs: "All" (selected), "Services (2)", "Marketplace (20)", "Documentation (29)", "Azure Active Directory (1)", and "Resources (0)". In the "Marketplace" section, there is a list of items. The first item, "WordPress", is highlighted with a blue border. Other items listed include "OpenLiteSpeed WordPress", "WordPress", "WordPress Server + PhpMyAdmin + MySQL", and "Wordpress LEMP Max Performance". On the left sidebar, under "Azure services", there are buttons for "Create a resource" and "More services". Under "Resources", there are sections for "Recent" and "Favorite", and a "Name" search input field.

You should be on the "**Create WordPress on App Service**" page. For your convenience, you can follow this link:  
<https://portal.azure.com/#create/WordPress.WordPress>

Create WordPress on App Service x +

portal.azure.com/#create/WordPress.WordPress

Microsoft Azure Search resources, services, and docs (G+/)

Home > Create WordPress on App Service

Basics Advanced Tags Review + create

The capabilities of Azure App Service have been leveraged to provide you with a fully managed, performant, secure and scalable hosting solution for WordPress. [Learn More ↗](#)

**Project details**

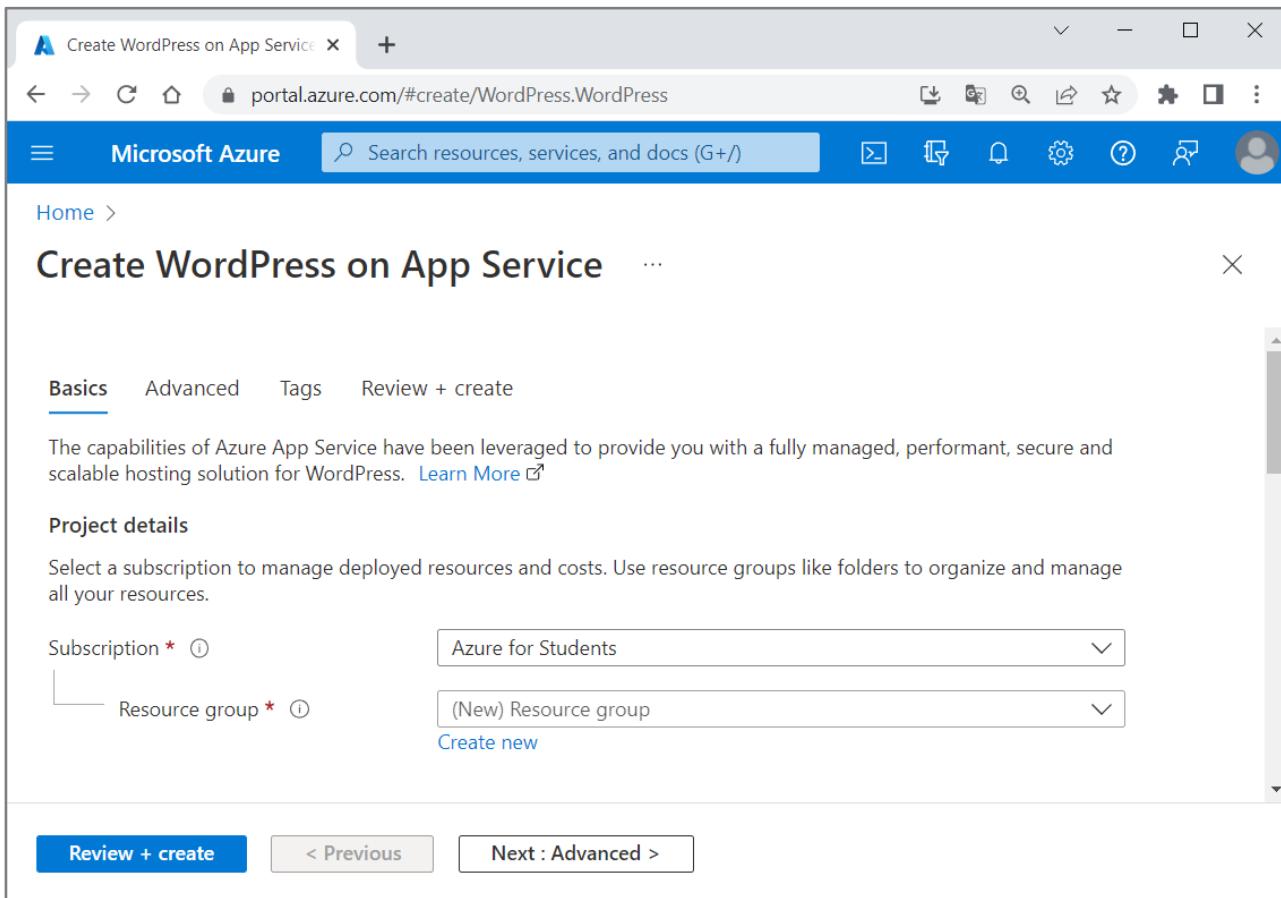
Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ Azure for Students

Resource group \* ⓘ (New) Resource group

Create new

Review + create < Previous Next : Advanced >



In the "**Basics**" tab, under "**Project details**", make sure the **correct subscription is selected – "Azure for Students"**:

Create WordPress on App Service

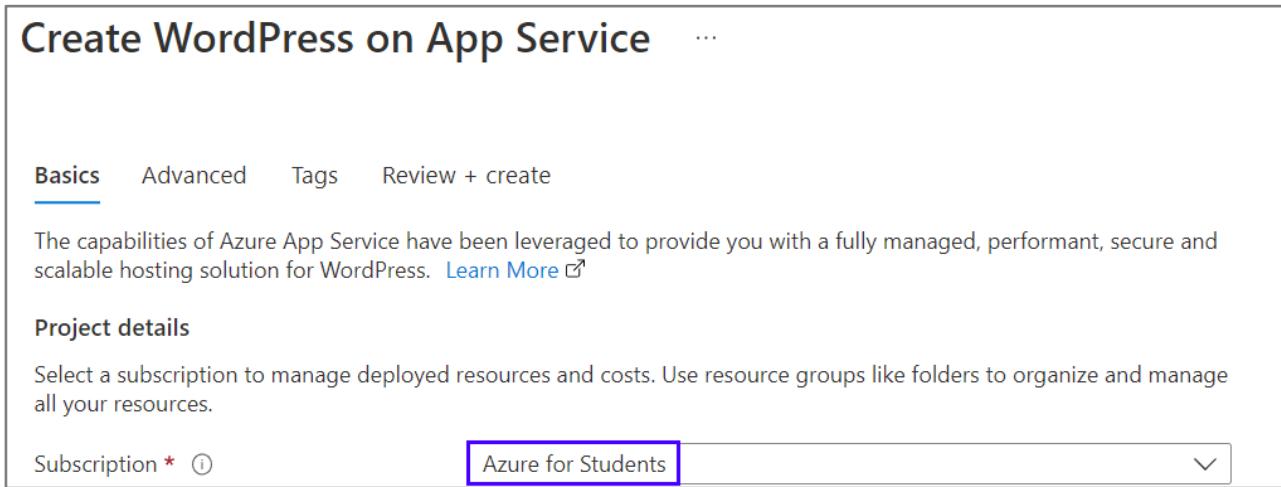
Basics Advanced Tags Review + create

The capabilities of Azure App Service have been leveraged to provide you with a fully managed, performant, secure and scalable hosting solution for WordPress. [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.

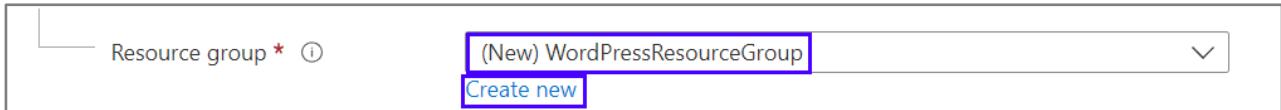
Subscription \* ⓘ Azure for Students



Then choose to [**Create new resource group**]. Type "**WordPressResourceGroup**" for the **name**:

Resource group \* ⓘ (New) WordPressResourceGroup

Create new

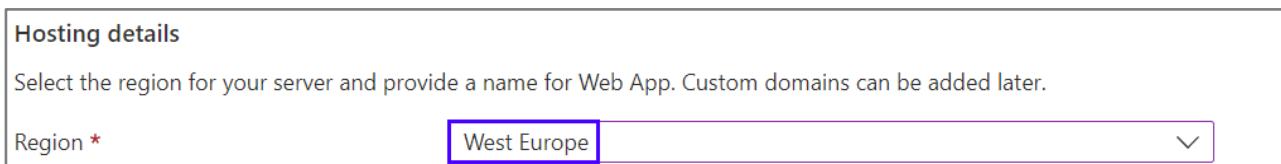


Select a **region** near you that you want to **serve your app from**:

**Hosting details**

Select the region for your server and provide a name for Web App. Custom domains can be added later.

Region \* West Europe



Provide a **name** for the Web App, which should be **unique**. You can name it "`wordpressapp{your name}`":

|                    |                   |   |
|--------------------|-------------------|---|
| Name *             | wordpressapppeter | ✓ |
| .azurewebsites.net |                   |   |

Select [**Basic**] for **hosting plan**:

|  |  |
|--|--|
| Hosting plans  |  |
| This hosting plan dictates what resources are available, what features are enabled and how it is priced. |  |
| Hosting plan   | <b>Basic</b><br>Basic App Service, Burstable MySQL database<br><a href="#">Change plan</a> |

Under "**WordPress Settings**", type an **admin email**, **admin username**, **admin password** and **admin confirm password**. The **admin email** here is used for **WordPress administrative sign-in** only:

|  |                           |
|--|---------------------------|
| WordPress setup  |                           |
| Select the language you want your website to be in. Provide admin email, username and password that you can use to access WordPress admin dashboard. |                           |
| Site language  | English (United States) ▾ |
| Admin email *  | ██████████ ✓              |
| Admin username *   | ██████████ ✓              |
| Admin password *   | ***** ✓                   |
| Confirm password *   | ***** ✓                   |

**Remember your credentials** for later. Then click on [**Review + create**]:

|                        |            |                   |
|------------------------|------------|-------------------|
| <b>Review + create</b> | < Previous | Next : Advanced > |
|------------------------|------------|-------------------|

Click on [**Create**] to **create the resources**:

Microsoft Azure Search resources, services, and docs (G+/)

Home > Create WordPress on App Service

Basics Advanced Tags Review + create

Summary

 **WordPress Hosting**  
by Microsoft

**Details**

|                |                                      |
|----------------|--------------------------------------|
| Subscription   | a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3 |
| Resource Group | WordPressResourceGroup               |
| Name           | wordpressappeter                     |

**App Service Plan (New)**

|                  |                                 |
|------------------|---------------------------------|
| Name             | ASP-WordPressResourceGroup-b89c |
| Operating System | Linux                           |

**Create** < Previous Next > Download a template for automation

Then **wait for the deployment to finish and click on the [Go to resource] button:**

Microsoft Azure Search resources, services, and docs (G+/)

Home > Microsoft.Web-Wordpress-Portal-466d3806-8363 | Overview

Deployment

Search Delete Cancel Redeploy Download Refresh

Overview Inputs Outputs Template

✓ Your deployment is complete

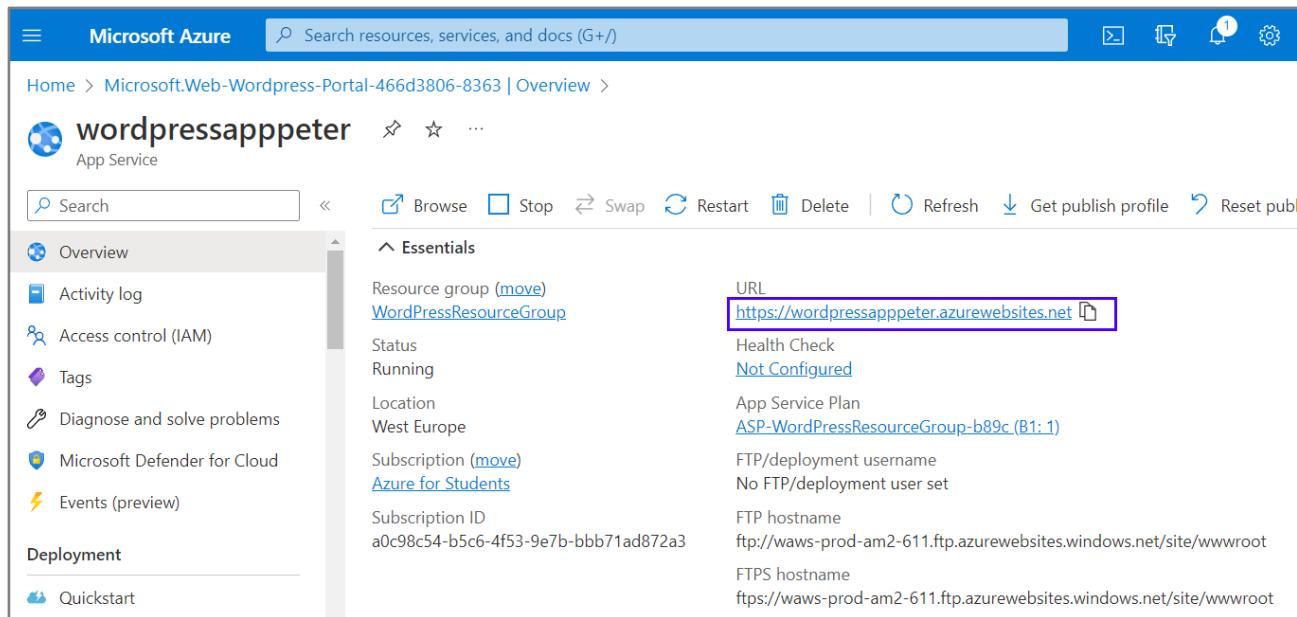
Deployment name: Microsoft.Web-Wordpress-Porta... Start time: 1/6/2023, 10:26:03 AM  
Subscription: Azure for Students Correlation ID: 45107c2c-70a3-4772-b0b8-63a93t  
Resource group: WordPressResourceGroup

Deployment details Next steps

**Go to resource**

## Step 2: Browse Site

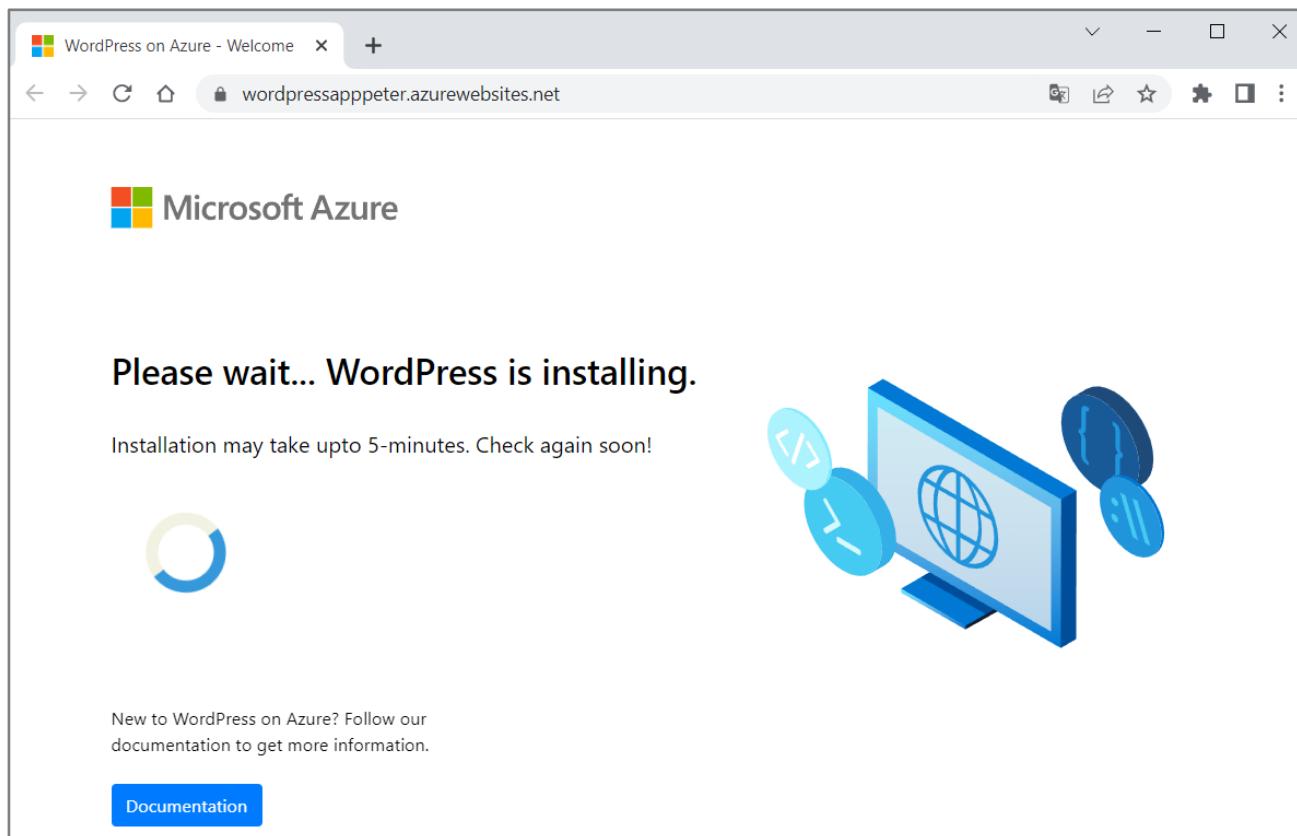
You can click on the URL of the site in the "App Service" page, in the [Overview] tab to access the site:



The screenshot shows the Microsoft Azure portal interface. In the top navigation bar, 'Microsoft Azure' is selected. Below it, a search bar says 'Search resources, services, and docs (G+ /)'. On the left, there's a sidebar with icons for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Deployment, and Quickstart. The main content area is titled 'wordpressapppeter' (App Service). It shows the 'Overview' tab selected. Under 'Essentials', the URL is listed as <https://wordpressapppeter.azurewebsites.net>.

Or you can also navigate directly to <https://<app-name>.azurewebsites.net>.

Your site needs time to install. Wait for it:



The screenshot shows a browser window with the title 'WordPress on Azure - Welcome'. The address bar contains 'wordpressapppeter.azurewebsites.net'. The page itself has a Microsoft Azure logo at the top. Below it, the text 'Please wait... WordPress is installing.' is displayed, followed by the note 'Installation may take upto 5-minutes. Check again soon!'. To the right, there's a 3D-style illustration of a computer monitor displaying a globe icon, surrounded by blue circles containing code snippets like '</>', '}', and '://'. At the bottom, there's a link 'Documentation'.

When ready, verify the **app is running properly**. If you **receive an error**, allow a few more minutes for the site to load and then **refresh the browser**.

WordPress on Azure

Sample Page

# Mindblown: a blog about philosophy.

## Hello world!

Welcome to WordPress. This is your first post.  
Edit or delete it, then start writing!

January 6, 2023

To access the WordPress Admin page, browse to **/wp-admin** and use the credentials you created in the "WordPress Settings" step:

The top screenshot shows a browser window with the URL <https://wordpressapppeter.azurewebsites.net/wp-admin> in the address bar, with the /wp-admin part highlighted by a blue box. The bottom screenshot shows the WordPress login screen at <https://wordpressapppeter.azurewebsites.net/wp-login....>. It features the classic blue 'W' logo, input fields for 'Username or Email Address' and 'Password', a 'Remember Me' checkbox, and a blue 'Log In' button. Below the form are links for 'Lost your password?' and '← Go to WordPress on Azure'.

The screenshot shows the WordPress dashboard with a sidebar containing links like Home, Posts, Media, Pages, Comments, Appearance, Plugins, Users, Tools, Settings, Performance, and Smush. A banner at the top says "WordPress 6.1.1 is available! Please update now." Below it, a message from Smush! encourages users to boost their Google PageSpeed score. The main area features a large blue "Welcome to WordPress!" banner.

## Step 3: Access the Database

You can also **access the MySQL Flexible Server database but not directly** because it is created behind a **private Virtual Network**.

To access or manage the database, we should first **get its credentials**, which are **generated automatically**. To retrieve these values after the deployment, go to "**Application Settings**" section of the "**Configuration**" page in **Azure App Service** and **copy the values** of **DATABASE\_USERNAME** and **DATABASE\_PASSWORD**:

The screenshot shows the Azure portal's configuration page for the "wordpressapppeter" app service. The left sidebar lists "Deployment" (Quickstart, Deployment credentials, Deployment slots, Deployment Center) and "Settings" (Configuration, Authentication, Application Insights, Identity). The "Configuration" option is selected. The main pane displays application settings with "DATABASE\_PASSWORD" and "DATABASE\_USERNAME" highlighted with a blue box. The table below shows other settings:

| Name                                | Value                             | Source      |
|-------------------------------------|-----------------------------------|-------------|
| DATABASE_HOST                       | Hidden value. Click to show value | App Service |
| DATABASE_NAME                       | Hidden value. Click to show value | App Service |
| <b>DATABASE_PASSWORD</b>            | Hidden value. Click to show value | App Service |
| <b>DATABASE_USERNAME</b>            | Hidden value. Click to show value | App Service |
| DOCKER_REGISTRY_SERVER_URL          | Hidden value. Click to show value | App Service |
| SETUP_PHPMYADMIN                    | Hidden value. Click to show value | App Service |
| WEBSITES_CONTAINER_START_TIME_LIMIT | Hidden value. Click to show value | App Service |
| WEBSITES_ENABLE_APP_SERVICE_STORAGE | Hidden value. Click to show value | App Service |
| WORDPRESS_ADMIN_EMAIL               | Hidden value. Click to show value | App Service |
| WORDPRESS_ADMIN_PASSWORD            | Hidden value. Click to show value | App Service |
| WORDPRESS_ADMIN_USER                | Hidden value. Click to show value | App Service |
| WORDPRESS_LOCALE_CODE               | Hidden value. Click to show value | App Service |

**Access the database by using phpMyAdmin** that's deployed with the WordPress site: navigate to <https://<sitename>.azurewebsites.net/phpmyadmin> and log in with the credentials:

The image shows two screenshots of the phpMyAdmin interface. The top screenshot is the login screen, featuring a logo of a sailboat and the text "Welcome to phpMyAdmin". It includes a "Language" dropdown set to "English", a "Log in" button, and fields for "Username" and "Password" which are highlighted with blue boxes. The bottom screenshot shows the "General settings" and "Appearance settings" sections. The "General settings" section contains options for "Change password", "Server connection collation" (set to "utf8mb4\_unicode\_ci"), and a "More settings" link. The "Appearance settings" section includes a "Language" dropdown set to "English", a "Theme" dropdown set to "pmahomme", and a "View all" link. A sidebar on the left lists databases: "New", "information\_schema", "mysql", "performance\_schema", "sys", and "wordpressa\_598239e2a6424ab4".

You can **explore the database and its tables**.

## Step 4: Explore App in Azure Portal

In **Azure Portal** you can **see and manage all app resources** that you created.

Look at the **resources** on the "**All resources**" page (you can search for it in the **search bar**):

The screenshot shows the Microsoft Azure 'All resources' page. At the top, there's a search bar and various navigation icons. Below the header, it says 'Home > All resources'. It displays a list of resources under the heading 'Software University (SoftUni) (softwareuniversity.onmicrosoft.com)'. The list includes:

- ASP-WordPressResourceGroup-b89c (App Service plan)
- wordpressa-598239e2a6424ab4adf35d06f0... (Azure Database for MySQL)
- wordpressa-7bd3e9d2bd-privatelink.mysql.... (Private DNS zone)
- wordpressa-7bd3e9d2bd-vnet (Virtual network)
- wordpressappeter (App Service)

Filters at the top include 'Subscription equals all', 'Resource group equals all', 'Type equals all', and 'No grouping'. At the bottom, it shows 'Showing 1 to 5 of 5 records.'

Look at each of resources and **explore Azure Portal**.

## Step 5: Delete App and Clean Up Resources

When you have successfully **uploaded your app to Azure** and **don't need it anymore**, it is time to **remove it**.

You can **delete all of the app resources** from your **Azure subscription** by **deleting the resource group**. Find **all resource groups** by going to **Azure Portal** and typing "Resource groups" in the search bar:

The screenshot shows the Microsoft Azure search interface. The search bar at the top contains the text 'Resource groups'. On the left, there's a sidebar with 'Azure services' and icons for 'Create a resource' and 'Subscriptions'. The main area shows a grid of service tiles, with one tile for 'Resource groups' highlighted with a blue border.

Microsoft Azure Search resources, services, and docs (G+) 1 ? X

Home > Resource groups ...

Software University (SoftUni) (softwareuniversity.onmicrosoft.com)

+ Create Manage view Refresh Export to CSV Open query Assign tags

Filter for any field... Subscription equals all Location equals all Add filter

0 Unsecure resources 0 Recommendations No grouping

List view Name ↑ Subscription ↑↓ Location ↑↓

WordPressResourceGroup Azure for Students West Europe

< Previous Page 1 of 1 Next > Showing 1 to 3 of 3 records. Give feedback

You have your "WordPressResourceGroup" group. Delete it by selecting [Delete resource group]:

Microsoft Azure Search resources, services, and docs (G+) 1 ? X

Home > Resource groups

WordPressResourceGroup ...

Resource group

+ Create Manage view Delete resource group Refresh Export to CSV Open query Assign tags

^ Essentials

Subscription (move) [Azure for Students](#) Deployments [1 Succeeded](#)

Subscription ID a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3 Location West Europe

Tags (edit) [Click here to add tags](#)

JSON View

Next, type the resource group name and click [Delete]:

Microsoft Azure Search resources, services, and docs (G+) 1 ? X

Home > Resource groups

TaskBoardResourceGroup ...

Resource group

Search Overview Activity log Access control (IAM) Tags Resource visualizer Events

+ Create Manage

^ Essentials

Subscription (move) [Azure for Students](#)

Subscription ID a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3

Tags (edit) [Click here to add tags](#)

Resources Recom

Filter for any field... Showing 1 to 5 of 5 records

Name Type Location

|                                |              |             |
|--------------------------------|--------------|-------------|
| taskboarddbserver              | SQL server   | West Europe |
| master (taskboarddbserver/...) | SQL database | West Europe |
| TaskBoard_db (taskboarddb...)  | SQL database | West Europe |
| TaskBoardApp_db (taskboard...) | SQL database | West Europe |
| TaskBoardPeter                 | App Service  | West Europe |

Delete Cancel

Are you sure you want to delete "TaskBo..." X

Warning! Deleting the "TaskBoardResourceGroup" resource group is irreversible. The action you're about to take can't be undone. Going further will delete this resource group and all the resources in it permanently.

TYPE THE RESOURCE GROUP NAME: TaskBoardResourceGroup

AFFECTED RESOURCES There are 6 resources in this resource group that will be deleted.

The resource group will be deleted after a while. All resources of your app will be deleted with it.

Now you have a **fully-functional WordPress site in Azure** and you know how to **manage it and its resources**.

## 2. Deploy a Node.js + MongoDB Web App to Azure

You are provided with a **Node.js app** with a **MongoDB database** called "TODO App" in the **resources**. It looks like this when **run**:

The screenshot shows a browser window with the URL `todoapp peter.azurewebsites.net`. The page title is "Todo List" and the subtitle is "ExpressJS/MongoDB Sample Application". The interface includes a "New task" input field with placeholder "Enter new task here" and a green "Add task" button. Below this is a "Current tasks" section with a table:

| Task        | Created    | Action  |
|-------------|------------|---|
| New Task    | 2023-01-06 | <button>Complete</button> <button>Delete</button> |
| Future Task | 2023-01-06 | <button>Complete</button> <button>Delete</button> |

Below the current tasks is a "Completed tasks" section with a table:

| Task          | Created    | Completed  |
|---------------|------------|------------|
| Finished Task | 2023-01-06 | 2023-01-06 |

You should **upload the app to GitHub**, as a start, and then **deploy it to Azure**, using **Azure App Service** and **Azure Cosmos DB for MongoDB**.

The process is pretty similar to **deploying an ASP.NET app with a SQL database** – the only difference is the **settings of the app**.

### Hints

You should fulfill the below steps to run the app in Azure:

1. Upload the "TODO App" project code from the **resources** to a **GitHub repository**.

The screenshot shows a GitHub repository page for "TODO-App: Sample". The repository has 1 branch and 0 tags. The "Code" tab is selected, showing a list of files:

- config
- models
- public/stylesheets
- routes
- views
- .env.sample
- LICENSE
- README.md
- app.js
- package-lock.json
- package.json

The README section contains the following text:

Sample JS + MongoDB app for deploy in Azure

Readme  
MIT license  
0 stars  
1 watching  
0 forks

No releases published  
Create a new release

No packages published  
Publish your first package

2. In **Azure Portal**, go to the "Create Web App + Database" page and **fill in the fields**. Create a **new resource group**. Give a **suitable name** to the Azure app. Choose **[Node 18 LTS]** as **runtime stack**. Don't change the **default selected database engine (Cosmos DB API for MongoDB)** and **copy the generated database name** because you will need it later.

## Create Web App + Database

Basics Tags Review + create

This template will create a secure by default configuration where the only publicly accessible endpoint will be your app following the recommended security best practices. [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) [REDACTED]<br>Create new |
| Region *         | West Europe                    |

### Web App Details

|                 |                              |
|-----------------|------------------------------|
| Name *          | [REDACTED].azurewebsites.net |
| Runtime stack * | Node 18 LTS                  |

### Database

**Info** Database access will be locked down and not exposed to the public internet. This is in compliance with recommended best practices for security.

|                 |   |
|-----------------|---|
| Engine *        | Cosmos DB API for MongoDB (recommended) |
| Account name *  | [REDACTED]                              |
| Database name * | [REDACTED]                              |

### Azure Cache for Redis

Add Azure Cache for Redis?

|                                     |
|-------------------------------------|
| <input type="radio"/> Yes           |
| <input checked="" type="radio"/> No |

### Hosting

Hosting plan \*

|   |
|---|
| <input checked="" type="radio"/> Basic - For hobby or research purposes |
| <input type="radio"/> Standard - General purpose production apps        |

**Review + create** < Previous Next : Tags >

3. Go to the "Configuration" page of the **App Service app** and create a [New application setting] with:
  - Name: "DATABASE\_NAME"
  - Value: the automatically generated database name you copied earlier (i.e. <app-name>-database)

**Add/Edit application setting**

Name  
DATABASE\_NAME 

Value  
-database 

Deployment slot setting

4. Next, select the "**MONGODB\_URI**" connection string on the same page and **copy its value**

**Add/Edit connection string**

Name  
MONGODB\_URI 

Value  
mongodb:// 

Type  
Custom

Deployment slot setting

5. Create one more **new application setting** with:

- Name: "DATABASE\_URL"
- Value: the "**MONGODB\_URI**" connection string you copied earlier (i.e., `mongodb://...`)

**Add/Edit application setting**

Name  
DATABASE\_URL 

Value  
mongodb:// 

Deployment slot setting

Don't forget to **save the settings**.

6. Your app is configured to work with the **Azure database**, so you should just go to the "**Deployment Center**" page and **deploy the app** from the **GitHub repository**.
7. A **GitHub workflow** should be created and it should **show a status of "Complete"**. It takes about 15-30 minutes.

8. You should be able to access the app on <https://<app-name>.azurewebsites.net> and work with it.

**NOTE:** Don't forget to delete the resources in your Azure account.

### 3. Deploy NGINX Server Container to Azure using Azure Portal

In this task, we will **deploy a container instance to Azure Container Instances** using the **Azure Portal**.

**Azure Container Instances** allows us to **run serverless and isolated Docker containers** in Azure and make its application available with a **fully qualified domain name (FQDN)**.

We will use the **NGINX server image** from **DockerHub**: <https://hub.docker.com/r/nginxdemos/hello>. You are already familiar with the **server**. When **deployed to Azure**, it will look like this:

Let's see how to do this. As a first step, open **Azure Portal** (<https://portal.azure.com>) and click on [**Create a resource**]:

The screenshot shows the Microsoft Azure home page. At the top left is the 'Home - Microsoft Azure' logo. The address bar contains 'portal.azure.com/#home'. The top navigation bar includes a search bar with 'Search resources, services, and docs (G+/-)', a 'Create' button, a 'Cloud Shell' icon, a bell icon, a gear icon, a question mark icon, and a person icon. Below the navigation bar, the heading 'Azure services' is displayed. A large blue button labeled 'Create a resource' with a plus sign is highlighted with a blue box. Other service icons include 'Resource groups', 'Reservations', 'All resources', 'App Services', 'Subscriptions', 'SQL databases', 'SQL servers', and 'Diagnostic settings'. An 'More services' link with a right-pointing arrow is also present.

Next, choose [Containers] → [Container Instances]:

The screenshot shows the 'Create a resource' page in the Azure portal. The left sidebar has a 'Categories' section with options like AI + Machine Learning, Analytics, Blockchain, Compute, Containers (which is highlighted with a blue box), Databases, Developer Tools, DevOps, Identity, Integration, Internet of Things, and IT & Management Tools. The main area features a search bar and sections for 'Popular Azure services' and 'Popular Marketplace products'. Under 'Popular Azure services', there are cards for Azure Kubernetes Service (AKS), Web App for Containers, Batch Service, Kubernetes - Azure Arc, Container App, and Container Instances (which is highlighted with a blue box). Under 'Popular Marketplace products', there are cards for NVIDIA GPU-Optimized Image for AI & HPC - v21.06.0, Windows Server 2022 Core Datacenter Minimal OS, Basic, Hyper-V Server on Windows Server 2016, Docker Engine Community on Ubuntu 20.04 LTS, and Docker Compose Server on Windows Server 2016.

On the "Basic" tab of the "Create container instance" page you should fill in the fields as follows:

- Use your "**Azure for Students**" subscription
- Create a **new resource group** with a suitable name
- Create a **name for your container**
- Choose a **region** near you
- Leave the "**Availability zones**" to [**None**]
- For the **image source**, choose [**Other registry**] (to use an **image from DockerHub**, which is the **default registry** for this **option**) and fill in the **image name**
- Leave the "**OS type**" to [**Linux**] and "**Size**" field with the **given options**

Microsoft Azure Search resources, services, and docs (G+)

Home > Create a resource >

## Create container instance

[Basics](#) [Networking](#) [Advanced](#) [Tags](#) [Review + create](#)

Azure Container Instances (ACI) allows you to quickly and easily run containers on Azure without managing servers or having to learn new tools. ACI offers per-second billing to minimize the cost of running containers on the cloud.

[Learn more about Azure Container Instances](#)

**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](#) [\(New\) NGINXServerResourceGroup](#) [Create new](#)

Resource group \* [\(New\) NGINXServerResourceGroup](#) [Create new](#)

**Container details**

Container name \* [nginx-server-container](#)

Region \* [\(Europe\) West Europe](#)

Availability zones [None](#)

Image source \* [Quickstart images](#) [Azure Container Registry](#) [Other registry](#) [Public](#) [Private](#)

Image \* [nginxdemos/hello](#)

If not specified, Docker Hub will be used for the container registry and the latest version of the image will be pulled.

OS type \* [Linux](#) [Windows](#)

This selection must match the OS of the image chosen above.

Size \* [1 vcpu, 1.5 GiB memory, 0 gpus](#) [Change size](#)

[Review + create](#) [< Previous](#) [Next : Networking >](#)

Click on **[Next]** to go to the "**Networking**" tab. There you should **fill in a DNS name for your container and choose [Tenant] for DNS scope reuse:**

Microsoft Azure Search resources, services, and docs (G+)

Home > Create a resource >

## Create container instance

Basics Networking Advanced Tags Review + create

Choose between three networking options for your container instance:

- **'Public'** will create a public IP address for your container instance.
- **'Private'** will allow you to choose a new or existing virtual network for your container instance. This is not yet available for Windows containers.
- **'None'** will not create either a public IP or virtual network. You will still be able to access your container logs using the command line.

Networking type  Public  Private  None

DNS name label  ✓

DNS name label scope reuse \*

Ports ✓

| Ports                | Ports protocol       |
|----------------------|----------------------|
| 80                   | TCP                  |
| <input type="text"/> | <input type="text"/> |

**Review + create** [< Previous](#) [Next : Advanced >](#)

The **name must be unique** within the Azure region where you create the container instance. Your container will be **publicly reachable** at `<dns-name-label>.⟨region⟩.azurecontainer.io`. An **auto-generated hash** is added as a **DNS name label** to your container instance's **fully qualified domain name (FQDN)**.

If you receive a "**DNS name label not available**" error message, try a **different DNS name label**.

**Leave all other settings** as their defaults and select **[Review + create]**, then **[Create]**. Wait for the **deployment to complete**:

Microsoft Azure Search resources, services, and docs (G+)

Home >

### Microsoft.ContainerInstances-20230110142813 | Overview

Deployment

Search [Delete](#) [Cancel](#) [Redeploy](#) [Download](#) [Refresh](#)

✓ Your deployment is complete

Deployment name: Microsoft.Container... Start time: 1/10/2023, 2:29:25 PM  
 Subscription: Azure for Students Correlation ID: 89de96e5-fd0a-4858-8f  
 Resource group: nginxServerResource...

✓ Deployment details ✓ Next steps

**Go to resource**

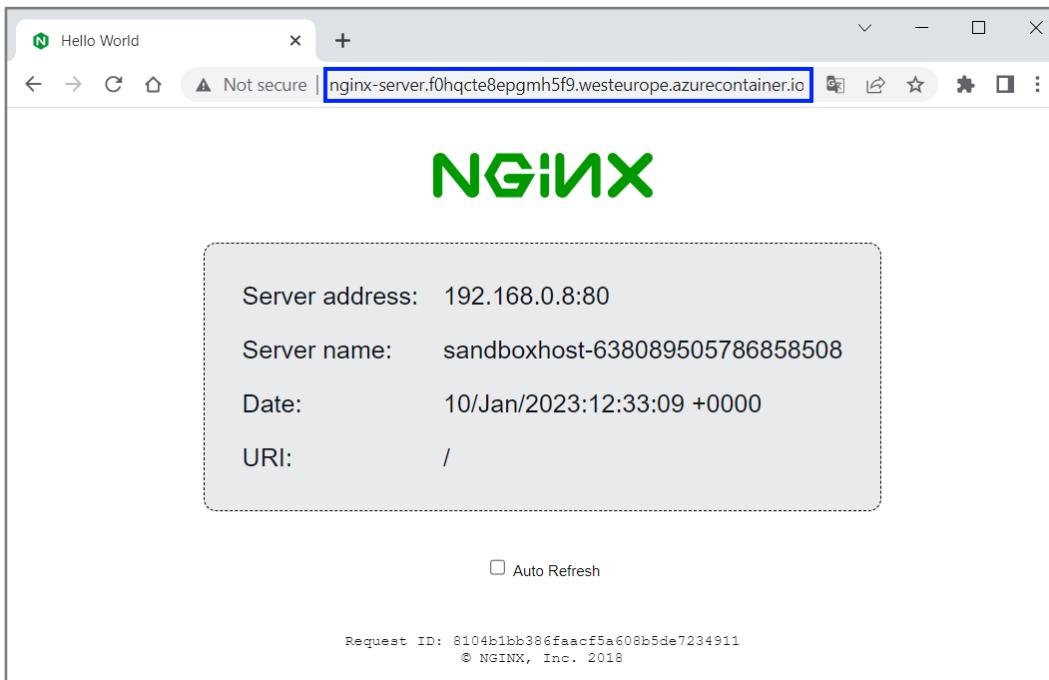
Then go to **nginx-server** container instance from the "**Container instances**" page:

| Name         | Resource group           | Location    | Status  | OS type | Total c... | Subscription       |
|--------------|--------------------------|-------------|---------|---------|------------|--------------------|
| nginx-server | nginxServerResourceGroup | West Europe | Running | Linux   | 1          | Azure for Students |

Look at the **FQDN** and the **status of the container** on the "Overview" page:

| Overview       | Activity log | Access control (IAM) | Tags | Containers | Identity | Properties | Locks |
|----------------|--------------|----------------------|------|------------|----------|------------|-------|
| <b>Running</b> |              |                      |      |            |          |            |       |

When the **status is "Running"**, navigate to the **container's FQDN** in your browser and you should see your **NGINX server**:



**NOTE:** if you receive an error, wait for the container a little bit more. After up to 5 minutes, it should be loading successfully.

If you have any other problems, you can look at the container logs in [Containers] → [Logs]:

| Name         | Image            | State   | Previous state | Start time              | Restart count |
|--------------|------------------|---------|----------------|-------------------------|---------------|
| nginx-server | nginxdemos/hello | Running | -              | 2023-01-10T12:30:06.... | 0             |

```
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf is not a file or does not exist
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/01/10 12:30:06 [notice] 19#19: using the "epoll" event method
2023/01/10 12:30:06 [notice] 19#19: nginx/1.23.3
2023/01/10 12:30:06 [notice] 19#19: built by gcc 12.2.1 20220924 (Alpine 12.2.1_git20220924-r4)
2023/01/10 12:30:06 [notice] 19#19: OS: Linux 5.10.102.2-microsoft-standard
2023/01/10 12:30:06 [notice] 19#19: getrlimit(RLIMIT_NOFILE): 1024:1048576
2023/01/10 12:30:06 [notice] 19#19: start worker processes
```

You can delete the container by deleting its resource group in the familiar to you way. You can also delete only the container by going to [Overview] and clicking on [Delete]:

The screenshot shows the Microsoft Azure Container Instances page. On the left, there's a sidebar with a search bar and links for Overview, Activity log, Access control (IAM), Tags, Containers, Identity, Properties, and Locks. The main area displays the 'nginx-server' container instance details under the 'Essentials' tab. The instance is running in the 'nginxServerResourceGroup' with an OS type of Linux, IP address 20.8.65.234, and FQDN nginx-server.f0hqcte8epgmh5f9.westeurope.azurecontainer.io. It's part of a subscription for 'Azure for Students' and has a single container. A 'Delete' button is highlighted with a blue box at the top right of the main content area.

## 4. Deploy NGINX Server Container to Azure using Azure CLI

Now we will see how to **deploy a NGINX server container to Azure** but using **Azure CLI commands**.

**NOTE:** Docker Desktop should be **running** in order for the commands to be executed successfully. Also, you should **open a CLI** to execute commands, for example **PowerShell**.

First, **log in to Azure** with the following command:

```
PS C:\Users\PC> docker login azure
login succeeded
```

A **browser window** for login to Azure will appear. Enter your **credentials** and you should be **successfully logged in**:

The image contains two screenshots. The left one shows a 'Sign in to Microsoft Azure' browser window with a 'Sign in' form. The email input field contains '@students.softuni.bg' and is highlighted with a red box. Below it are 'Next' and 'Sign-in options' buttons. The right screenshot shows a separate browser window titled 'Login successfully' with the URL 'localhost:10132/?code...'. It displays the message 'You have logged into Microsoft Azure!' and a note: 'You can close this window, or we will redirect you to the [Docker ACI integration documentation](#) in 10 seconds.'

You can **close the above browser window** and return to **PowerShell**.

Next, we should **create an ACI context**, which associates Docker with an Azure subscription and resource group so you can create and manage container instances. Our **context** will be called "nginxacicontext":

```
PS C:\Users\PC> docker context create aci nginxacicontext
Using only available subscription : Azure for Students (a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3)
? Select a resource group [Use arrows to move, type to filter]
> create a new resource group
```

Now you are prompted to **select your subscription** and a **resource group**. You should **choose the "Azure for Students" subscription** and **create a new resource group**, using the **arrow keys** and the **[Enter]** key on your keyboard.

In our case we have **only one subscription option** and **no resource groups**, so hitting **[Enter]** is enough:

```
PS C:\Users\PC> docker context create aci nginxacicontext
Using only available subscription : Azure for Students (a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3)
? Select a resource group create a new resource group
Resource group "6c71f2b8-547e-2cf3-ab83-4336e4556bc8" (eastus) created
Successfully created aci context "nginxacicontext"
```

A **resource group** and an **ACI context** were successfully created. Note that the **new resource group** is created with a **system-generated name**.

Confirm that you **added the ACI context** to your **Docker contexts** like this:

```
PS C:\Users\PC> docker context ls
NAME          TYPE      DESCRIPTION
default        *         moby
desktop-linux  moby
nginxacicontext  aci      6c71f2b8-547e-2cf3-ab83-4336e4556bc8@eastus
```

After **creating a Docker context**, you can **create a container in Azure**. First **switch to the ACI context** we created, so that Docker commands run in it:

```
PS C:\Users\PC> docker context use nginxacicontext
nginxacicontext
```

Run a **docker run** command to run a NGINX server container instance in Azure with exposed port 80:

```
PS C:\Users\PC> docker run -p 80:80 nginxdemos/hello
[+] Running 2/2
- Group pedantic-brahmagupta Created                                3.7s
- pedantic-brahmagupta       Created                                26.1s
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf is not a file or does not exist
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/01/11 08:48:34 [notice] 19#19: using the "epoll" event method
2023/01/11 08:48:34 [notice] 19#19: nginx/1.23.3
2023/01/11 08:48:34 [notice] 19#19: built by gcc 12.2.1 20220924 (Alpine 12.2.1_git20220924-r4)
2023/01/11 08:48:34 [notice] 19#19: OS: Linux 5.10.102.2-microsoft-standard
```

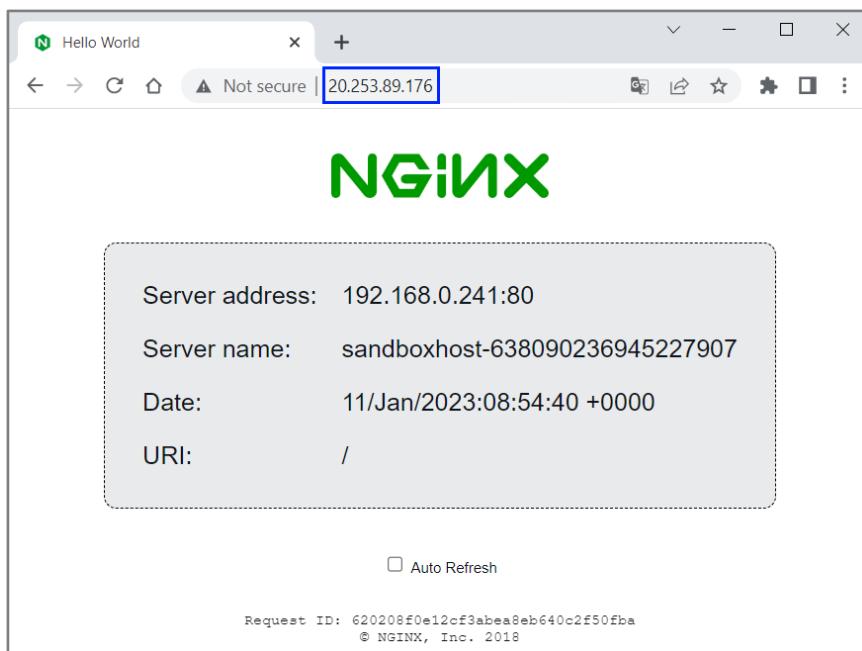
```
2023/01/11 08:48:34 [notice] 19#19: getrlimit(RLIMIT_NOFILE): 1024:1048576
2023/01/11 08:48:34 [notice] 19#19: start worker processes
2023/01/11 08:48:34 [notice] 19#19: start worker process 37
```

As you can see, the **name of the container** here is **pedantic-brahmagupta** (yours may be different). You can press **[Ctrl]** + **[C]** to exit the logs.

Then, use the following command to **see all containers** and get details about your **running container**, including its **public IP address**:

```
PS C:\Users\PC> docker ps
CONTAINER ID        IMAGE               COMMAND      STATUS        PORTS
pedantic-brahmagupta   nginxdemos/hello    "nginx -g..."  Running     20.253.89.176:80->80/tcp
```

Get this address and paste it in the browser to access your app (its **status** should be "Running"):



**NOTE:** it may be necessary that you **wait a few minutes** for the **container app to load**.

You can also **look at your container instance in Azure Portal**:

The screenshot shows the Microsoft Azure Container Instances page for a container named 'pedantic-brahmagupta'. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Containers, Identity, Properties, and Locks. The main content area displays the following details:

- Resource group (move):** 6c71f2b8-547e-2cf3-ab83-4336e4556bc8
- OS type:** Linux
- Status:** Running
- IP address (Public):** 20.253.89.176
- Location:** East US
- FQDN:** ---
- Subscription (move):** Azure for Students
- Subscription ID:** a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3
- Container count:** 1
- Tags (edit):** docker-single-container : docker-single-container

Note that in this case we **have an IP address**, not a **FQDN**.

Also, you can **pull the container logs** if you need them like this:

```
PS C:\Users\PC> docker logs pedantic-brahmagupta
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform
configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.
sh
10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf is not a file
or does not exist
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/01/11 08:48:34 [notice] 19#19: using the "epoll" event method
2023/01/11 08:48:34 [notice] 19#19: nginx/1.23.3
2023/01/11 08:48:34 [notice] 19#19: built by gcc 12.2.1 20220924 (Alpine 12.2.1_git
20220924-r4)
2023/01/11 08:48:34 [notice] 19#19: OS: Linux 5.10.102.2-microsoft-standard
2023/01/11 08:48:34 [notice] 19#19: getrlimit(RLIMIT_NOFILE): 1024:1048576
2023/01/11 08:48:34 [notice] 19#19: start worker processes
2023/01/11 08:48:34 [notice] 19#19: start worker process 37
10.92.0.11 - - [11/Jan/2023:08:54:40 +0000] "GET / HTTP/1.1" 200 7252 "-" "Mozilla/
5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108
.0.0.0 Safari/537.36" "-"
10.92.0.9 - - [11/Jan/2023:09:02:57 +0000] "\x16\x03\x01\x00\x85\x01\x00\x00\x81\x0
3\x03\x1D\xD2\xFD\xE1\xA5Q\x09\x19\x17iX\x01\x88\xAC\xB2T\x83\xA98}I\xAD\x9C/\x02\x
A5" 400 157 "-" "-" "-"
```

You can also **stop** and **delete** the container:

```
PS C:\Users\PC> docker stop pedantic-brahmagupta  
pedantic-brahmagupta
```

```
PS C:\Users\PC> docker rm pedantic-brahmagupta  
pedantic-brahmagupta
```

However, the **resource group** should be deleted from **Azure Portal**.

After this **first two tasks** you know **how to create Azure Container Instances** using **Azure Portal** and **Docker CLI**. You could also **do it through Azure CLI**, which we will **show you in the next task**.

## 5. Deploy the "Tracker" App to Azure

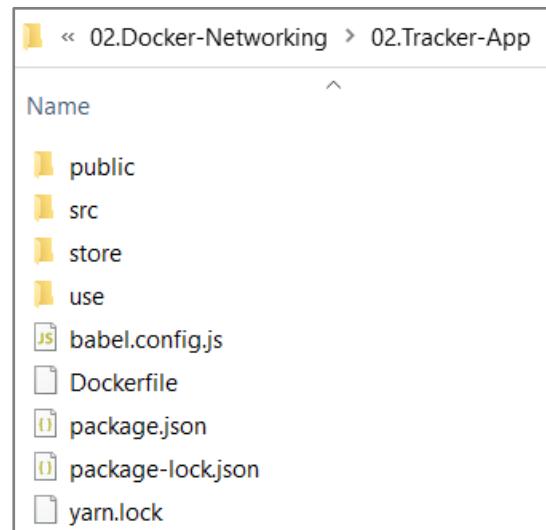
Your task now is to **run a simple JavaScript front-end app based on Vue.js** for keeping track of daily duties in a **Docker container**. It does not need **anything but an image** to run. It does not use a database or any other storage.

You're provided with **its files** it in the **resources**, together with a **Dockerfile** which runs the **app on NGINX server** (on the right):

You should **deploy the app to Azure Container Instances**.

### Install Azure CLI on Your Local Computer

The **Azure Command-Line Interface (CLI)** is a cross-platform **command-line tool**, which can be used to **connect to Azure** and **execute administrative commands** on **Azure resources**. It can be run from **inside a Docker container**.



Because the **Azure Cloud Shell** does not include the **Docker daemon**, you must **install both the Azure CLI and Docker Engine** on your **local computer** for this task. We will **use it**, as well, but later when we are done with Docker.

You already have **Docker**, so let's **install Azure CLI**. This can be done **through PowerShell** with the following **command**:

```
$ProgressPreference = 'SilentlyContinue'; Invoke-WebRequest -Uri  
https://aka.ms/installazurecliwindows -OutFile .\AzureCLI.msi; Start-Process  
msiexec.exe -Wait -ArgumentList '/I AzureCLI.msi /quiet'; rm .\AzureCLI.msi
```

```
Administrator: Windows PowerShell  
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Try the new cross-plat>  
PS C:\Windows\system32> $ProgressPreference = 'SilentlyContinue'; Invoke-WebRequest  
-Uri https://aka.ms/installazurecliwindows -OutFile .\AzureCLI.msi; Start-Process  
msiexec.exe -Wait -ArgumentList '/I AzureCLI.msi /quiet'; rm .\AzureCLI.msi
```

**NOTE:** **PowerShell** must be run as **administrator** for the **installation**. After that, run a **new PowerShell instance** in the standard way and **work with it**.

You can check your **Azure CLI version** with:

```
PS C:\Users\PC> az --version
azure-cli          2.44.1
core              2.44.1
telemetry         1.0.8

Dependencies:
msal                1.20.0
azure-mgmt-resource 21.1.0b1

Python location 'C:\Program Files (x86)\Microsoft SDKs\Azure\CLI2\python.exe'
Extensions directory 'C:\Users\EVELINA-PC\.azure\cliextensions'

Python (Windows) 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:37:59) [MSC v.1933 32
bit (Intel)]

Legal docs and information: aka.ms/AzureCliLegal

Your CLI is up-to-date.
```

Now you can go on with **creating a container image**.

## Step 1: Create a Container Image for Deployment

In this step, you will **package the "Tracker App"** into a container image that can be **run using Azure Container Instances**.

You are provided with a **Dockerfile**, so you should just **create the image**, for example "**tracker-app-image**". You should know how to do it:

```
PS D:\SoftUni\03.Tracker-App> docker build -t tracker-app-image .
[+] Building 0.7s (14/14) FINISHED
 ...
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn
how to fix them
```

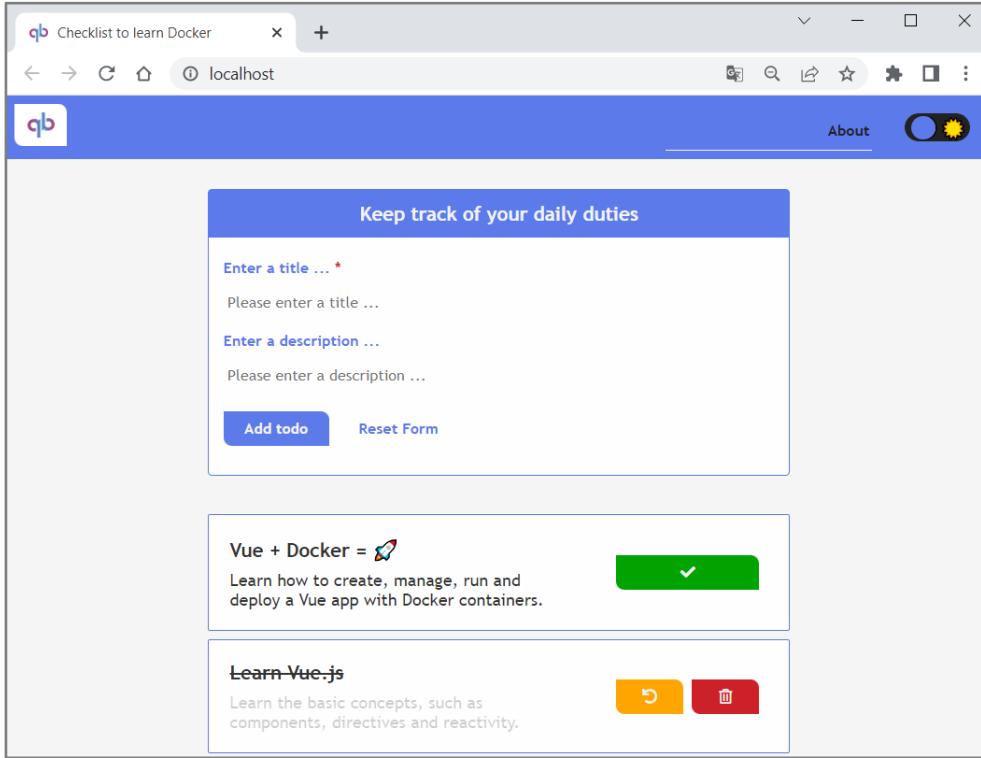
The **newly created image** should appear:

```
PS D:\SoftUni\03.Tracker-App> docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
tracker-app-image  latest  19749ad7ccaa  5 minutes ago  143MB
```

You can now use this **image to run a container locally** to see if it works:

```
PS D:\SoftUni\03.Tracker-App> docker run -d 10a7c7ae15f24f2de40e0f2923a4917597a8269e75db51390e664b1e9a50fc07
```

↓



We have verified that the **container image runs locally**, so let's see how to **push it to Azure Container Registry**.

## Step 2: Create an Azure Container Registry and Push a Container Image

An **Azure Container Registry** is a **managed registry service**. It stores and manages **private container images** and other artifacts, similar to the way **Docker Hub** stores **public Docker container images**.

Now we should **create an Azure container registry** and **push our image** to it.

First, **login to Azure** through **Azure CLI** with the **az login** command. A **web browser will be opened** and you should **enter your credentials**. Then **login** should be successful:

```
PS D:\SoftUni\03.Tracker-App> az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login in the web browser. If no web browser is available or if the web browser fails to open, use device code flow with `az login --use-device-code`.
[{"cloudName": "AzureCloud",
"homeTenantId": "REDACTED",
"id": "REDACTED",
"isDefault": true,
"managedByTenants": [],
"name": "Azure for Students",
"state": "Enabled",
"tenantId": "REDACTED",
"user": {
"name": "REDACTED",
"type": "user"
}
}]
```

## Create Azure Container Registry

Now you need a **resource group** to **deploy the container registry** that we will create to.

**Create a new resource group** in Azure with the `az group create` command. **Name the group** in a way you want and set it in the "**westeurope**" **region**:

```
PS D:\SoftUni\03.Tracker-App> az group create --name trackerapprg --location westeurope

{
  "id": "/subscriptions/a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3/resourceGroups/trackerapprg",
  "location": "westerneurope",
  "managedBy": null,
  "name": "trackerapprg",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

Once you've created the resource group, **create an Azure container registry** with the `az acr create` command. The **container registry name** must be **unique within Azure**, and contain **5-50 alphanumeric characters**:

```
PS D:\SoftUni\03.Tracker-App> az acr create --resource-group trackerapprg --name trackerapprcr --sku Basic
Resource provider 'Microsoft.ContainerRegistry' used by this operation is not registered. We are registering for you.
Registration succeeded.
{
  "adminUserEnabled": false,
  ...
  "zoneRedundancy": "Disabled"
}
```

You can also **see the container registry in Azure Portal** by going to the "**Container Registries**" page:

The screenshot shows the Microsoft Azure portal interface with the title "Container registries". The search bar at the top contains "portal.azure.com/#view/HubsExtension/BrowseResource/resourceType/Microsoft.ContainerRegistry". Below the search bar, there's a blue header bar with the Microsoft Azure logo and a search input field. The main content area displays a table of container registry entries. The first entry in the table is highlighted with a blue border around its "Name" column. The table has columns for Name, Type, Resource group, Location, and Subscription. The "Name" column shows "trackerapprcr", "Type" shows "Container registry", "Resource group" shows "trackerapprg", "Location" shows "West Europe", and "Subscription" shows "Azure for Students". At the bottom of the table, there are navigation links for "Previous", "Page 1 of 1", "Next >", and "Showing 1 to 1 of 1 records". On the right side of the table, there are buttons for "Give feedback" and "Add filter".

The screenshot shows the Microsoft Azure portal interface. At the top, there's a search bar and a navigation bar with icons for Home, Container registries, and other services. Below that, the main content area shows a 'trackerappcr' container registry. On the left, there's a sidebar with options like Overview, Activity log, Access control (IAM), Tags, Quick start, and Events. The main panel displays the 'Essentials' section with details such as Resource group (move) to 'trackerAppResourceGroup', Location as West Europe, Subscription (move) to 'Azure for Students', Subscription ID as a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3, Login server as 'trackerappcr.azurecr.io', Creation date as 1/11/2023, 2:29 PM GMT+2, SKU as Basic, Provisioning state as Succeeded, and Soft Delete (Preview) as Disabled.

Next, you need to **log in to the container registry** you created, so that you can **push images** to it. You should use the **az acr login** command and provide the **name of the registry** you created:

```
PS D:\SoftUni\03.Tracker-App> az acr login --name trackerappcr
Login Succeeded
```

Login should be **successful**.

## Tag Container Image

To **push a container image** to a **private registry** like **Azure Container Registry**, you must first **tag the image** with the **full name of the registry's login server**.

Get the **full login server name** for your **Azure container registry**:

```
PS D:\SoftUni\03.Tracker-App> az acr show --name trackerappcr --query loginServer --
output table
Result
-----
trackerappcr.azurecr.io ← -----
```

Now you should **tag the container image** we created in the previous step with the **login server of your container registry**. Also, add the **:v1 tag** to the **end of the image name** to indicate the **image version number**. Do it like this:

|  | local image name  | login server name       |
|--|-------------------|-------------------------|
| PS D:\SoftUni\03.Tracker-App> docker tag | tracker-app-image | trackerappcr.azurecr.io |
| tracker-app-image:v1                     |                   |                         |
| new image name                           | image version     |                         |

You can look and see that the **new image is created**:

```
PS D:\SoftUni\03.Tracker-App> docker images
REPOSITORY          TAG      IMAGE ID   CREATED
tracker-app-image   latest   19749ad7ccaa  2 hours ago
trackerappcr.azurecr.io/tracker-app-image   v1      19749ad7ccaa  2 hours ago
```

Now you **have your image**.

## Push image to Azure Container Registry

Now you can finally **push your new image to Azure Container Registry**. Do it as shown below:

```
PS D:\SoftUni\03.Tracker-App> docker push trackerappcr.azurecr.io/tracker-app-image
:v1
The push refers to repository [trackerappcr.azurecr.io/tracker-app-image]
309d25d04ced: Pushed
80115eeb30bc: Pushed
049fd3bdb25d: Pushed
ff1154af28db: Pushed
8477a329ab95: Pushed
7e7121bf193a: Pushed
67a4178b7d47: Pushed
v1: digest: sha256:7c4287352e7d3ff0462310c25ecb9d11cb284d4fc9dc85887aabc15428340abf
size: 1780
```

Your **image should be pushed successfully**. To verify this, **list the images** in your **container registry**:

```
PS D:\SoftUni\03.Tracker-App> az acr repository list --name trackerappcr
[
  "tracker-app-image"
]
```

You have your **image in Azure Container Registry**. You can also **check in Azure Portal** when you select the **container registry → [Repositories]**:

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes the Microsoft Azure logo, a search bar, and various navigation icons. Below the header, the URL is visible as portal.azure.com/#@softwareuniversity.onmicrosoft.com/resource/subscriptio... . The main content area displays the 'trackerappcr | Repositories' page for a container registry. On the left, there's a sidebar with 'Services' listed: 'Repositories' (which is selected and highlighted with a blue border), 'Webhooks', and 'Replications'. The main pane shows a search bar and a 'Search to filter repositories ...' input field. A list of repositories is shown, with 'tracker-app-image' highlighted by a blue border.

The screenshot shows the Microsoft Azure portal interface. At the top, there's a search bar and a navigation bar with icons for Home, Container registries, trackerappcr | Repositories, and a user profile. Below this, the main content area displays a repository named "tracker-app-image". The repository details include its name, last updated date (1/11/2023, 2:54 PM GMT+2), tag count (1), and manifest count (1). A search bar for filtering tags is also present. The repository listing shows one tag, v1, with a digest hash and last modified date of 1/11/2023, 2:54 PM GMT+2.

In this step you **prepared an Azure container registry** for use with **Azure Container Instances**, and pushed a **container image** to the registry.

## Step 3: Deploy a Container Application

Finally, we have everything needed to **deploy a container to Azure Container Instances**.

### Get Registry Credentials

When you **deploy an image** that's **hosted in a private Azure container registry** like the one created, you must **supply credentials** to access the registry. For this to happen, you shall **create and configure an Azure Active Directory (AD) service principal** with **pull permissions** to your registry.

To do this, you will need the **following commands**:

```
$ACR_NAME='$containerRegistry'
$SERVICE_PRINCIPAL_NAME='$servicePrincipal'

# Obtain the full registry ID
$ACR_REGISTRY_ID=$(az acr show --name $ACR_NAME --query "id" --output tsv)

# Create and configure the service principal with pull permissions to your registry
$PASSWORD=$(az ad sp create-for-rbac --name $SERVICE_PRINCIPAL_NAME --scopes
$ACR_REGISTRY_ID --role acrpull --query "password" --output tsv)
$USER_NAME=$(az ad sp list --display-name $SERVICE_PRINCIPAL_NAME --query
"[].appId" --output tsv)

# Output the service principal's credentials
echo "Service principal ID: $USER_NAME"
```

```
echo "Service principal password: $PASSWORD"
```

You should **modify them** for your environment:

- **ACR\_NAME** should keep the **name of your Azure container registry**, which you already created
- **SERVICE\_PRINCIPAL\_NAME** should keep a **new name for your service principal**, which should be unique within your Active Directory tenant

**Execute the commands** one by one with the changes:

```
PS D:\SoftUni\03.Tracker-App> $ACR_NAME='trackerappcr'
PS D:\SoftUni\03.Tracker-App> $SERVICE_PRINCIPAL_NAME='trackerappsp'
PS D:\SoftUni\03.Tracker-App> $ACR_REGISTRY_ID=$(az acr show --name $ACR_NAME --query "id" --output tsv)
PS D:\SoftUni\03.Tracker-App> $PASSWORD=$(az ad sp create-for-rbac --name $SERVICE_PRINCIPAL_NAME --scopes $ACR_REGISTRY_ID --role acrpull --query "password" --output tsv)
WARNING: Found an existing application instance: (id) a830c596-47a5-4dff-9b58-56b9b319ee59. We will patch it.
WARNING: Creating 'acrpull' role assignment under scope '/subscriptions/a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3/resourceGroups/trackerapprg/providers/Microsoft.ContainerRegistry/registries/trackerappcr'
WARNING: The output includes credentials that you must protect. Be sure that you do not include these credentials in your code or check the credentials into your source control. For more information, see https://aka.ms/azadsp-cli
PS D:\SoftUni\03.Tracker-App> echo "Service principal ID: $USER_NAME"
Service principal ID: [REDACTED]
PS D:\SoftUni\03.Tracker-App> echo "Service principal password: $PASSWORD"
Service principal password: [REDACTED]
```

You should **have no errors** and finally you should have your **service principal id and password**. Once you have these **credentials**, you can **configure your applications and services** to authenticate to your container registry as the service principal. **Copy them** because you will need them.

**NOTE:** If you get the "**Insufficient privileges to complete the operation.**" error message, go to "Container registries" in the **Azure Portal** and select the **trackerapp** container registry:

The screenshot shows the Azure Container registries blade. At the top, there's a header with 'All services >' and the title 'Container registries'. Below the title, it says 'Software University (SoftUni) (softwareuniversity.onmicrosoft.com)'. There are several buttons: '+ Create', 'Manage view', 'Refresh', 'Export to CSV', 'Open query', and 'Assign tags'. Below these are filter buttons for 'Subscription equals all', 'Resource group equals all', 'Location equals all', and an 'Add filter' button. A message 'Showing 1 to 1 of 1 records.' is displayed. The main table has columns: 'Name', 'Type', 'Resource group', 'Location', and 'Subscription'. One record is listed: 'trackerappcr' (Container registry), 'trackerapprg' (Resource group), 'West Europe' (Location), 'Azure for Students' (Subscription). The 'Name' column header is highlighted with a blue box.

After that, select **Access keys** and enable the **Admin user**:

The screenshot shows the 'Access keys' page for the 'trackerapprc' container registry. On the left, there's a sidebar with various navigation options like Overview, Activity log, Access control (IAM), Tags, Quick start, Events, Settings, and Access keys (which is selected and highlighted with a blue border). The main area displays the registry name ('trackerapprc'), login server ('trackerapprc.azurecr.io'), and an 'Enabled' toggle switch. Below that, two sets of access keys are listed: 'password' and 'password2', each with a 'Regenerate' button.

## Deploy Container

Now use the `az container create` command to **deploy a container in Azure**. Use this command and replace `<resource-group-name>`, `<image-name>`, `<acrLoginServer>`, `<service-principal-ID>` and `<service-principal-password>` with your corresponding values. Also, replace `<container-name>` with a **name for your container** and `<aciDnsLabel>` with a new DNS name that you like:

```
az container create --resource-group <resource-group-name> --name <container-name>
--image <acrLoginServer>/<image-name>:v1 --cpu 1 --memory 1 --registry-login-
server <acrLoginServer> --registry-username <service-principal-ID> --registry-
password <service-principal-password> --ip-address Public --dns-name-label
<aciDnsLabel> --ports 80
```

**NOTE:** If you have enabled the `Admin user` in the previous step, just replace `<service-principal-ID>` with `<username>` and `<service-principal-password>` with `<password>`

Container deployment should be **successful**:

```
PS D:\SoftUni\03.Tracker-App> az container create --resource-group trackerapprg --
name trackerapp --image trackerapprc.azurecr.io/tracker-app-image:v1 --cpu 1 --mem
ory 1 --registry-login-server trackerapprc.azurecr.io --registry-username
[REDACTED] --registry-password [REDACTED]
[REDACTED] --ip-address Public --dns-name-label trackerappdns --ports 80
{
  "containers": [
    {
      "command": null,
      ...
    },
    {
      "volumes": null,
      "zones": null
    }
  ]
}
```

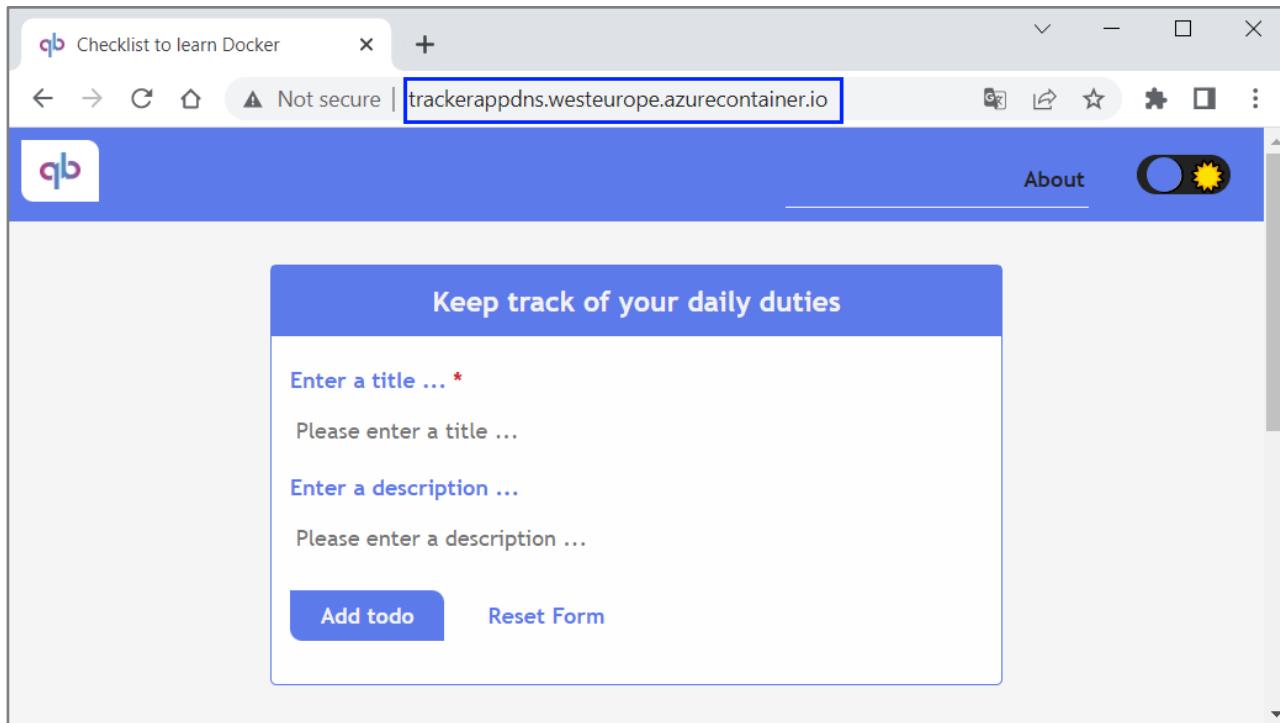
To view the container state of deployment, use:

```
PS D:\SoftUni\03.Tracker-App> az container show --resource-group trackerapprg --na
me trackerapp --query instanceView.state
"Running"
```

Once the **deployment succeeds**, display the **container's fully qualified domain name (FQDN)**:

```
PS D:\SoftUni\03.Tracker-App> az container show --resource-group trackerapprg --na
me trackerapp --query ipAddress.fqdn
"trackerappdns.westeurope.azurecontainer.io"
```

Use the **FQDN** to access your app in the browser (it contains the **DNS label** you chose):



It should be **fully working**.

## Explore and Remove Container

You can look at the **created container instance in Azure Portal** too:

All resources - Microsoft Azure

portal.azure.com/#view/HubsExtension/BrowseAll

## All resources

Software University (SoftUni) (softwareuniversity.onmicrosoft.com)

+ Create    Manage view    Refresh    Export to CSV    Open query    Assign tags    Delete

Filter for any field...    Subscription equals all    Resource group equals all    Type equals all    Location equals all    Add filter

Unsecure resources    Recommendations    No grouping    List view

| Name ↑                                | Type ↑              | Resource group ↑ | Location ↑  | Subscription ↑     |
|---------------------------------------|---------------------|------------------|-------------|--------------------|
| <input type="checkbox"/> trackerapp   | Container instances | trackerapprg     | West Europe | Azure for Students |
| <input type="checkbox"/> trackerapprc | Container registry  | trackerapprg     | West Europe | Azure for Students |

< Previous    Page 1 of 1    Next >    Showing 1 to 3 of 3 records.    Give feedback

All resources - Microsoft Azure

portal.azure.com/#view/HubsExtension/BrowseAll

## trackerapp

Container instances

Search    Start    Restart    Stop    Delete    Refresh

Overview    Essentials    JSON View

Activity log    Access control (IAM)    Tags

Settings

Containers    Identity    Properties

Resource group ([move](#))  
trackerapprg

Status  
Running

Location  
West Europe

Subscription ([move](#))  
Azure for Students

Subscription ID  
a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3

OS type  
Linux

IP address (Public)  
20.23.87.25

FQDN  
trackerapppdns.westeurope.azurecontainer.io

Container count  
1

You can also view the **log output of the container** from **Azure CLI**:

```
PS D:\SoftUni\03.Tracker-App> az container logs --resource-group trackerapprg --name trackerapp
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
```

And finally, **clear all resources by deleting the resource group** with **az group delete**:

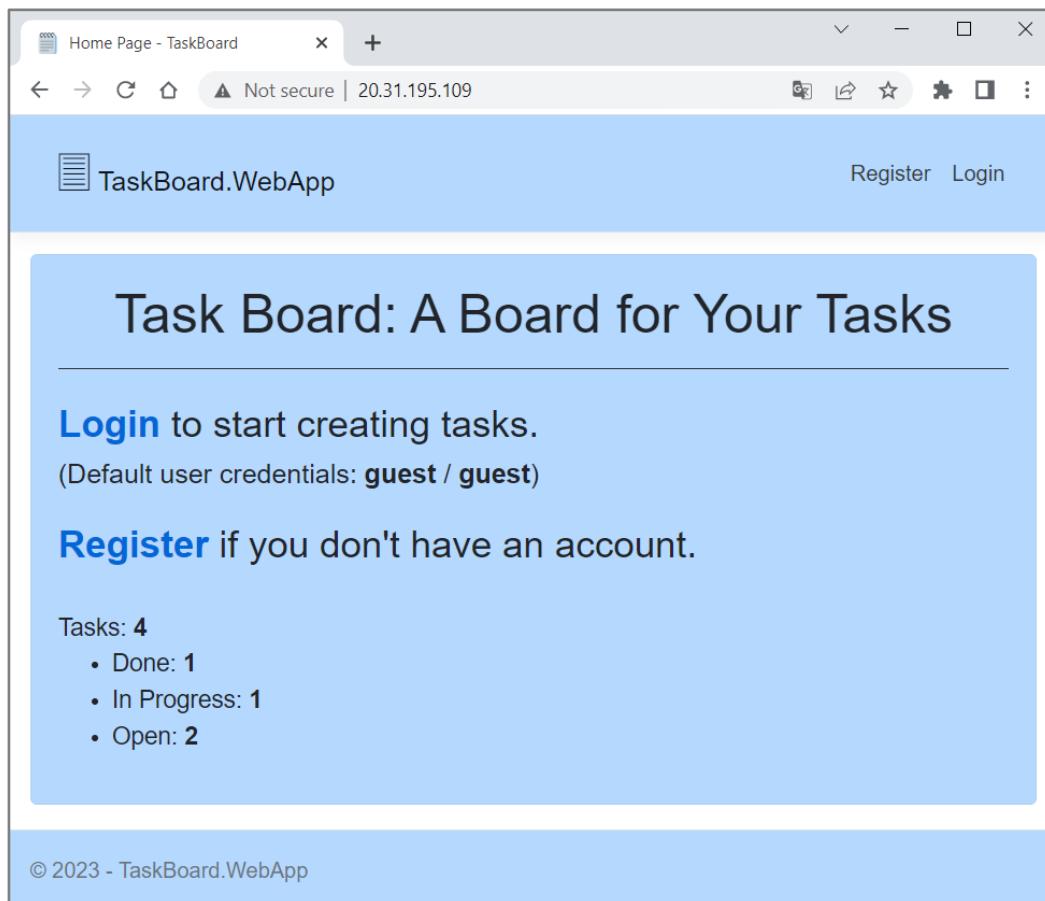
```
PS D:\SoftUni\03.Tracker-App> az group delete --name trackerapprg
Are you sure you want to perform this operation? (y/n): y
```

Now we know how to **create resource groups**, **container registries**, **container instances**, etc. and **deploy a container app in Azure**.

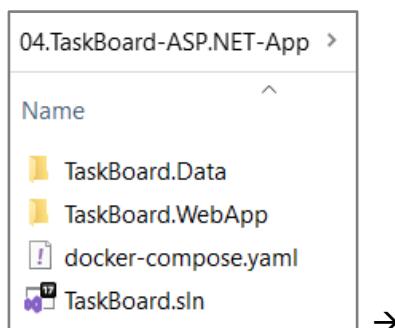
## 6. Deploy the "TaskBoard" App using Docker Compose

In this task, we will **deploy** the **TaskBoard multi-container app** as a **container group** in **Azure Container Instances**, using **Docker Compose**. This is a way to easily switch from local to cloud deployment.

When successfully **deployed to Azure**, the app will look like this on its **provided IP address**:



You have the **TaskBoard** app with a **docker-compose.yaml** file for it in the **resources**:



```

docker-compose.yaml

version: "3.8"

services:
  sqlserver:
    container_name: sqlserver
    image: mcr.microsoft.com/mssql/server
    ports:
      - "1433:1433"
    environment:
      - ACCEPT_EULA=Y
      - MSSQL_SA_PASSWORD=yourStrongPassword12#
    volumes:
      - sqldata:/var/opt/mssql
  web-app:
    container_name: web-app
    build:
      dockerfile: ./TaskBoard.WebApp/Dockerfile
    ports:
      - "5000:80"
    restart: on-failure

volumes:
  sqldata:

```

To create and use a **volume** in Azure, we will use the **Azure File Share service**. As you can see, **we haven't defined a custom network**, as **Azure will create a default one**. If you want, however, you can search and create a **custom network in Azure** for your containers.

Let's see how to **deploy the app in ACI**.

## Step 1: Create Azure Container Registry

The first thing we should do is to **create a container registry** in Azure, using the **CLI**. We will **create a resource group** named "**taskBoardResourceGroup**" and a **container registry** with name "**taskboardcrr**". You can do this by yourself:

```

PS D:\SoftUni\TaskBoard-ASP.NET-App> [REDACTED]
{
  "id": "/subscriptions/a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3/resourceGroups/taskboardResourceGroup",
  "location": "westeurope",
  "managedBy": null,
  "name": "taskboardResourceGroup",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}

```

```

PS D:\SoftUni\TaskBoard-ASP.NET-App> [REDACTED]
{
  "adminUserEnabled": false,
  "anonymousPullEnabled": false,
  ...
  "location": "westeurope",
  "loginServer": "taskboardcrr.azurecr.io",
  "name": "taskboardcrr",
  ...
  "resourceGroup": "taskBoardResourceGroup",
  "sku": {
    "name": "Basic",
  ...
}

```

You can also look at them in **Azure Portal**:

**taskboardResourceGroup** Resource group

**Overview**

- Subscription (move) [Azure for Students](#)
- Subscription ID: a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3
- Tags (edit) [Click here to add tags](#)
- Deployments: No deployments
- Location: West Europe

**Resources**

| Name        | Type               | Location    |
|-------------|--------------------|-------------|
| taskboardcr | Container registry | West Europe |

**taskboardcr** Container registry

**Overview**

- Resource group (move) [taskBoardResourceGroup](#)
- Location: West Europe
- Subscription (move) [Azure for Students](#)
- Subscription ID: a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3

**Essentials**

| Value                                   |
|---|
| Login server: taskboardcr.azurecr.io    |
| Creation date: 2/9/2023, 11:49 AM GMT+2 |
| SKU: Basic                              |
| Provisioning state: Succeeded           |
| Soft Delete (Preview): Disabled         |

Then you should **log in** to the **taskboardcr container registry** that you have just created:

```
PS D:\SoftUni\TaskBoard-ASP.NET-App> az acr login --name taskboardcr
Login Succeeded
```

## Step 2: Modify Docker Compose File

Now you should make some **modifications to the provided docker-compose.yaml file**, so that it works in **Azure** later.

First, you should make sure that in **each container** the **default container port** and the **exposed port** are the same:

```
version: "3.8"
services:
  sqlserver:
    container_name: sqlserver
    image: mcr.microsoft.com/mssql/server
    ports:
      - "1433:1433"
    environment: ...
    volumes: ...
  web-app:
    container_name: web-app
    build: ...
    ports:
      - "80:80"
    restart: on-failure
volumes: ...
```

This is because **port mapping** is **not supported** with ACI.

Next, you should **add an image property** to the **web-app service**. It should **contain the login service name** of your container registry (**taskboardcr.azurecr.io**) + / + **your image name**:

```
version: "3.8"
services:
  sqlserver: ...
  web-app:
    container_name: web-app
    build: ...
    image: taskboardcr.azurecr.io/taskboard-image
    ports: ...
    restart: on-failure
volumes: ...
```

These were the **changes** we needed to make for now. Because of these changes, the **taskboard-image** will be **tagged** for your **Azure container registry** on the next step. Then it can be **pulled to run in Azure Container Instances**.

Finally, you should **set deployment options** for **resources to be reserved** for the **sqlserver** container:

```

version: "3.8"

services:
  sqlserver:
    container_name: sqlserver
    image: mcr.microsoft.com/mssql/server
    ports: ...
    deploy:
      resources:
        reservations:
          cpus: '2'
          memory: 2GB
    environment: ...
    volumes: ...
  web-app: ...
volumes: ...

```

The **SQL container** needs **2 CPU** and **2GB memory** at least to work properly. If it is **missing resources**, it would not be able to **create all database tables** it needs and this would result in **errors**.

**NOTE:** it is important to know that **services names** should contain only lowercase letters, numbers and hyphens and the **password** should not contain symbols like "\$". If you have **any problems** with the **docker-compose up** command later, **return to this step** and think about a **possible problem** in the **docker-compose.yaml** file.

### Step 3: Run Multi-Container App Locally

Use the **docker-compose.yaml** file we modified to **build the container image** and **start the TaskBoard application locally**:

```

PS D:\SoftUni\TaskBoard-ASP.NET-App> docker-compose up --build -d
[+] Building 14.6s (19/19) FINISHED
=> [internal] load build definition from Dockerfile
...
[+] Running 3/3
- Network taskboard-aspnet-app_default  Created          0.8s
- Container web-app                  Started         2.1s
- Container sqlserver                Started         2.1s

```

When completed, you will have the **Docker containers** and the **tagged web-app service image**:

|  |              |                         |                   |                      |
|--|--------------|-------------------------|-------------------|----------------------|
| taskboard-aspnet-app                   | -            | Running (2/2)           |                   |                      |
| web-app                                | 1d0a9034f27a | taskboardcr.azurecr.io/ | Running 80:80     | 10 seconds ago       |
| sqlserver                              | d7bd9f8cd70e | mcr.microsoft.com/ms    | Running 1433:1433 | 10 seconds ago       |
| taskboardcr.azurecr.io/taskboard-image | f7e7a818c4c5 | latest                  | In use            | 2 days ago 286.27 MB |

Make sure that the **app is working correctly** with the database:

Home Page - TaskBoard

localhost

TaskBoard.WebApp

Register Login

# Task Board: A Board for Your Tasks

**Login** to start creating tasks.  
(Default user credentials: guest / guest)

**Register** if you don't have an account.

Tasks: 4

- Done: 1
- In Progress: 2
- Open: 1

© 2023 - TaskBoard.WebApp

Then you can **stop the app** and **remove the containers**.

You can see in Docker that we have the **container image** and it is **tagged**:

| Name   | Tag    | Status | Created                | Size      |
|--|--------|--------|------------------------|-----------|
| taskboardcr.azurecr.io/taskboard-image<br>258a7aa9c675 | latest | In use | less than a minute ago | 285.51 MB |

So now we should **push it to our Azure container registry**. Use the **docker-compose push** command and the **image should be available in the registry**:

```
PS D:\SoftUni\TaskBoard-ASP.NET-App> az acr repository show --name taskboardcr --repository taskboard-image
{
  "changeableAttributes": {
    "deleteEnabled": true,
    "listEnabled": true,
    "readEnabled": true,
    "teleportEnabled": false,
    "writeEnabled": true
  },
  "createdTime": "2023-02-09T12:18:36.9712859Z",
  "imageName": "taskboard-image",
  "lastUpdateTime": "2023-02-09T12:18:37.073611Z",
  "manifestCount": 1,
  "registry": "taskboardcr.azurecr.io",
  "tagCount": 1
}
```

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes the Azure logo, a search bar, and various icons. Below the header, the URL is [https://portal.azure.com/#view/Microsoft\\_Azure\\_ContainerRegistries/RepositoryBlade/id/%2Ftaskboardcr%2Ftaskboard-image](https://portal.azure.com/#view/Microsoft_Azure_ContainerRegistries/RepositoryBlade/id/%2Ftaskboardcr%2Ftaskboard-image). The main content area displays the details for the 'taskboard-image' repository. It shows the repository name, last updated date (2/9/2023, 2:18 PM GMT+2), tag count (1), manifest count (1), and a digest (sha256:61199f343c8bc734416959689040a357ba...). A search bar at the bottom allows filtering by tag.

Later we will use this image.

## Step 4: Set up Azure File Share Storage

As you know, a **SQL Server container** needs a **volume to persist data**. **Azure File Share** is used to support **volumes** for **ACI containers**. We should **create a file share** and use it in our **docker-compose.yaml** file.

**Azure file shares** are deployed into **storage accounts**. An **Azure storage account** contains all of your **Azure Storage data objects**: blobs, files, queues, and tables. Azure supports multiple types of storage accounts.

Azure can create a **file share** by itself when you add it in the **Docker Compose file**. **Mount the volume** to the SQL Server container's **directory** and **specify the volume driver (azure\_file)** and its **options (file share and storage account names)** like this:

The screenshot shows a code editor with a file named 'docker-compose.yaml'. The file content is as follows:

```
version: "3.8"
services:
  sqlserver:
    container_name: sqlserver
    image: mcr.microsoft.com/mssql/server
    ports: ...
    deploy: ...
    environment: ...
    volumes:
      - sqldata:/var/opt/mssql
  web-app: ...
volumes:
  sqldata:
    driver: azure_file
    driver_opts:
      share_name: sql-volume
      storage_account_name: taskboardstorageacc
```

The 'volumes' section is highlighted with a blue box.

We set the **name of the volume** to "sql-volume" and the **storage account name** to "taskboardstorageacc" but you can **use different names**.

Your app is **fully prepared for deployment to ACI**.

## Step 5: Create Azure Context

You should **create an ACI context** to associate Docker with an Azure subscription and resource group so you can **create and manage container instances**. You know how to **create a context**, so create the **taskboardcontext aci context** in the **taskBoardResourceGroup resource group** by yourself:

```
PS D:\SoftUni\TaskBoard-ASP.NET-App> az acri context create --name taskboardcontext --resource-group taskBoardResourceGroup --subscription "Azure for students (a0c98c54-b5c6-4f53-9e7b-bbb71ad872a)"
? Select a resource group taskboardResourceGroup (westeurope)
Successfully created aci context "taskboardcontext"
```

Then use the **created context**:

```
PS D:\SoftUni\TaskBoard-ASP.NET-App> az acri context use taskboardcontext
```

This will make the **subsequent commands executed in this context**, e.g., in **Azure**.

## Step 6: Deploy App to Azure Container Instances

Finally, you should **run the docker compose up command** in the **aci context** to **start the application in Azure Container Instances**:

```
PS D:\SoftUni\TaskBoard-ASP.NET-App> docker compose up
[+] Running 5/5
  - taskboardstorageacc              Created            21.5s
  - sql-volume                      Created            1.2s
  - Group taskboard-aspnet-app     Created            7.8s
  - web-app                         Created           93.1s
  - sqlserver                       Created           93.1s
```

The **taskboard-image image** will be **pulled from your container registry**. After a while, **storage account** and **file share** are **created** and will be used for **volume** in the container. Then, **containers are deployed as a group in ACI**.

**NOTE:** if any errors occur or the command does not run the containers, think about what you may have done wrong in the previous steps. Especially, you may have problems with your **docker-compose.yaml file**. Try to solve the problems and **run the app in Azure successfully**.

To see the **running containers** and the **IP address assigned to the container group**, execute the following command:

| CONTAINER ID                   | IMAGE   | STATUS  | PORTS                       |
|--------------------------------|---|---------|-----------------------------|
| taskboard-aspnet-app_sqlserver | mcr.microsoft.com/m.../taskboard-aspnet-app/_taskboard-aspnet-app_sqlserver | Running | 20.23.53.217:1433->1433/tcp |
| taskboard-aspnet-app_web-app   | mcr.microsoft.com/m.../taskboard-aspnet-app/_taskboard-aspnet-app_web-app   | Running | 20.23.53.217:80->80/tcp     |

You can also look at them in **Azure Portal**:

**taskboard-aspnet-app**

Container instances

Search

Start | Restart | Stop | Delete | Refresh

Overview

Activity log | Access control (IAM) | Tags

Settings

Containers | Identity | Properties | Locks

Essentials

| Resource group (move)                | OS type             |
|--------------------------------------|---------------------|
| taskboardResourceGroup               | Linux               |
| Status                               | IP address (Public) |
| Running                              | 20.31.195.109       |
| Location                             | FQDN                |
| West Europe                          | ---                 |
| Subscription (move)                  | Container count     |
| Azure for Students                   | 3                   |
| Subscription ID                      |                     |
| a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3 |                     |
| Tags (edit)                          |                     |

JSON View

You can also look at each container's logs in [Settings] → [Containers].

And here are the **storage account** and the **file share** that were created:

**taskboardstorageacc**

Storage account

Search

Upload | Open in Explorer | Delete | Move | Refresh | Open in mobile | ...

Overview

Activity log | Tags

Diagnose and solve problems

Access Control (IAM)

Data migration

Events

Storage browser

Data storage

Containers | File shares

Essentials

| Resource group (move)                | Performance                     |
|--------------------------------------|---------------------------------|
| taskBoardResourceGroup               | Standard                        |
| Location                             | Replication                     |
| West Europe                          | Locally-redundant storage (LRS) |
| Subscription (move)                  | Account kind                    |
| Azure for Students                   | Storage (general purpose v1)    |
| Subscription ID                      | Provisioning state              |
| a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3 | Succeeded                       |
| Disk state                           | Created                         |
| Available                            | 2/9/2023, 12:16:44 PM           |

JSON View

taskboardstorageacc | File shares

Storage account

Search

+ File share Refresh

File share settings

Active Directory: Not configured Default share-level permissions: Disabled Soft delete: 7 days

Maximum capacity: 5 TiB Security: Maximum compatibility

Search file shares by prefix (case-sensitive)

Show deleted shares

| Name       | Modified              | Quota |
|------------|-----------------------|-------|
| sql-volume | 2/9/2023, 12:31:18 PM | 5 TiB |

## Step 7: Run the App in Azure

At the end, use the IP address of the app to navigate to it. Wait for several minutes, refresh the browser several times and your TaskBoard app should be up and working:

Home Page - TaskBoard

Not secure | 20.31.195.109

TaskBoard.WebApp

Register Login

## Task Board: A Board for Your Tasks

**Login** to start creating tasks.  
(Default user credentials: **guest / guest**)

**Register** if you don't have an account.

Tasks: 4

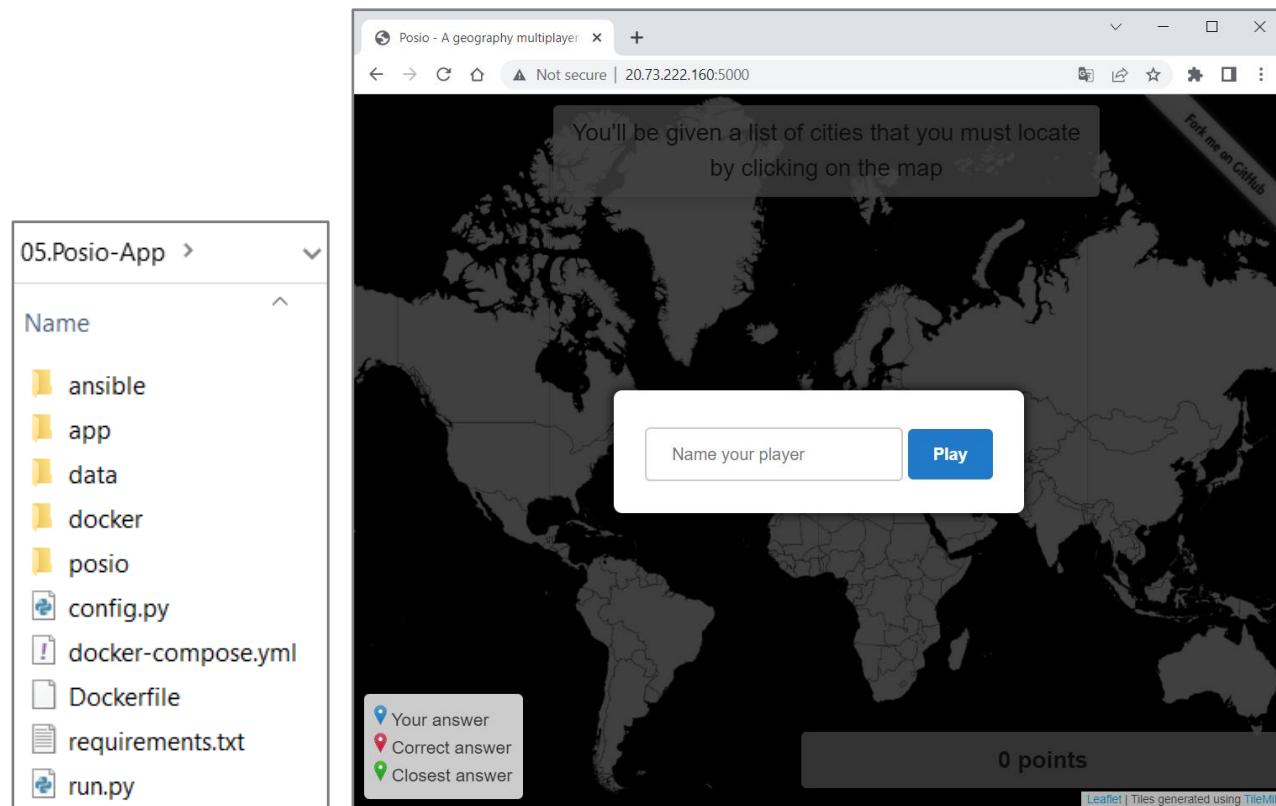
- Done: 1
- In Progress: 1
- Open: 2

© 2023 - TaskBoard.WebApp

**NOTE:** you may have multiple errors in the container logs because the SQL database takes time to be created and the web app fails to connect to it several times. However, if this behavior doesn't stop or any of your containers is terminated, then you should search for a problem with the app or its deployment.

## 7. Deploy the "Posio" App

Here is one more task for you to try and **deploy a simple app to Azure Container Instances** using a **Docker Compose file**. The "Posio" app is a **simple multiplayer geography game** using Websockets. You have it in the resources and it should look like this when **deployed**:

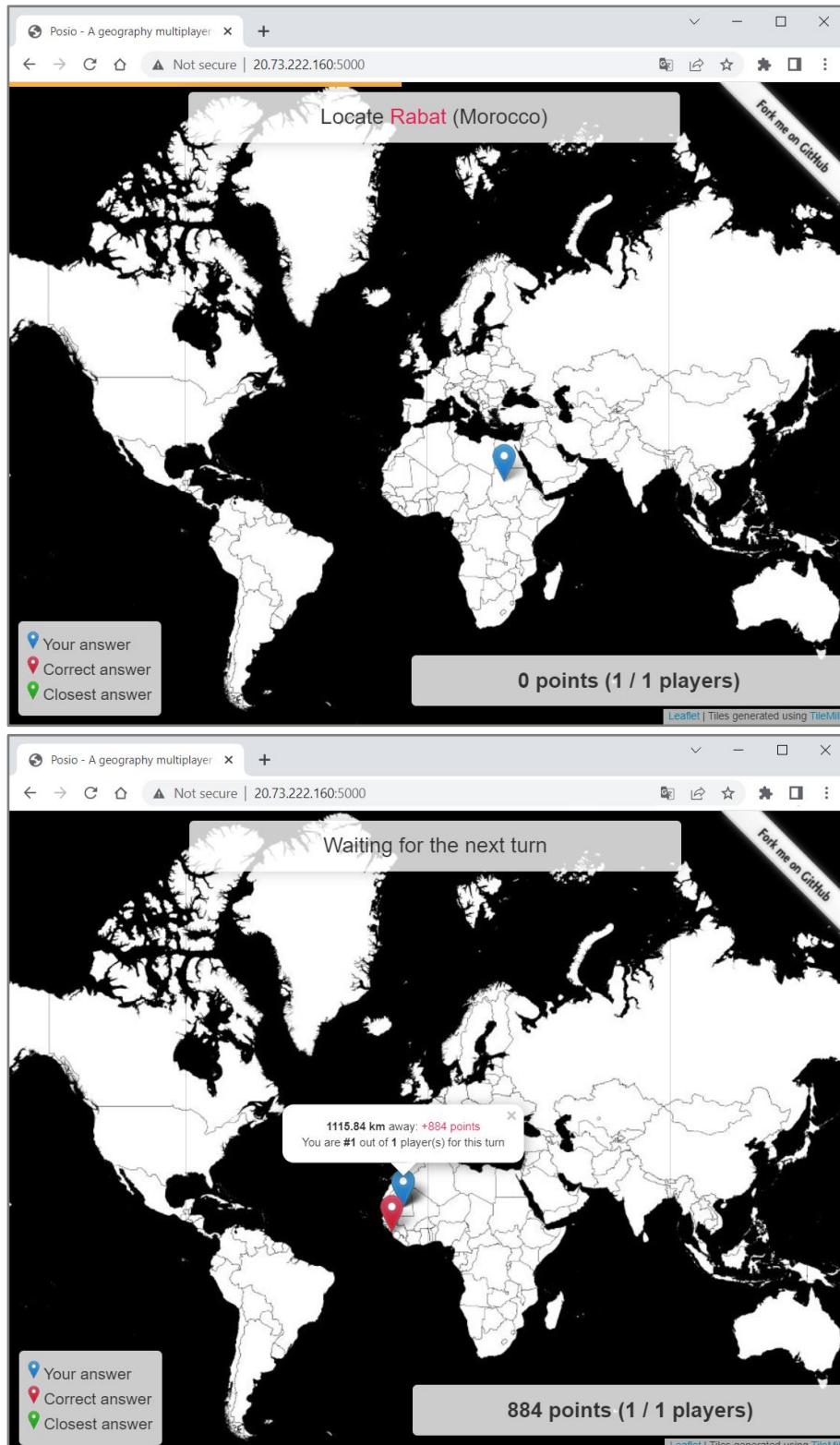


The screenshot shows the Microsoft Azure portal with the search bar "Search resources, services, and docs (G+ /)" and the user profile icon. The main area shows a container instance named "posio-app". The "Overview" section displays the following details:

| Resource group      | : posioResourceGroup             |
|---------------------|----------------------------------|
| Status              | : Running                        |
| Location            | : West Europe                    |
| Subscription        | : Azure for Students             |
| Subscription ID     | : 8238f760-177c-42c1-ba0e-749... |
| OS type             | : Linux                          |
| IP address (Public) | : 20.73.222.160                  |
| FQDN                | : ---                            |
| Container count     | : 1                              |
| Tags                | : More (1)                       |

To **deploy** the app, follow the steps from the **previous task**: create a container registry and push the app image, set up a file share to be created, create context and deploy the app to ACI.

When done, the **app should be accessed** on {IP Address}:5000 and you should be able to **play the game**:



You can play with friends – there is a **rank of players**.