

Exercises: Deployment to Cloud

Problems for exercises for the "[Containers and Clouds](#)" course @ SoftUni.

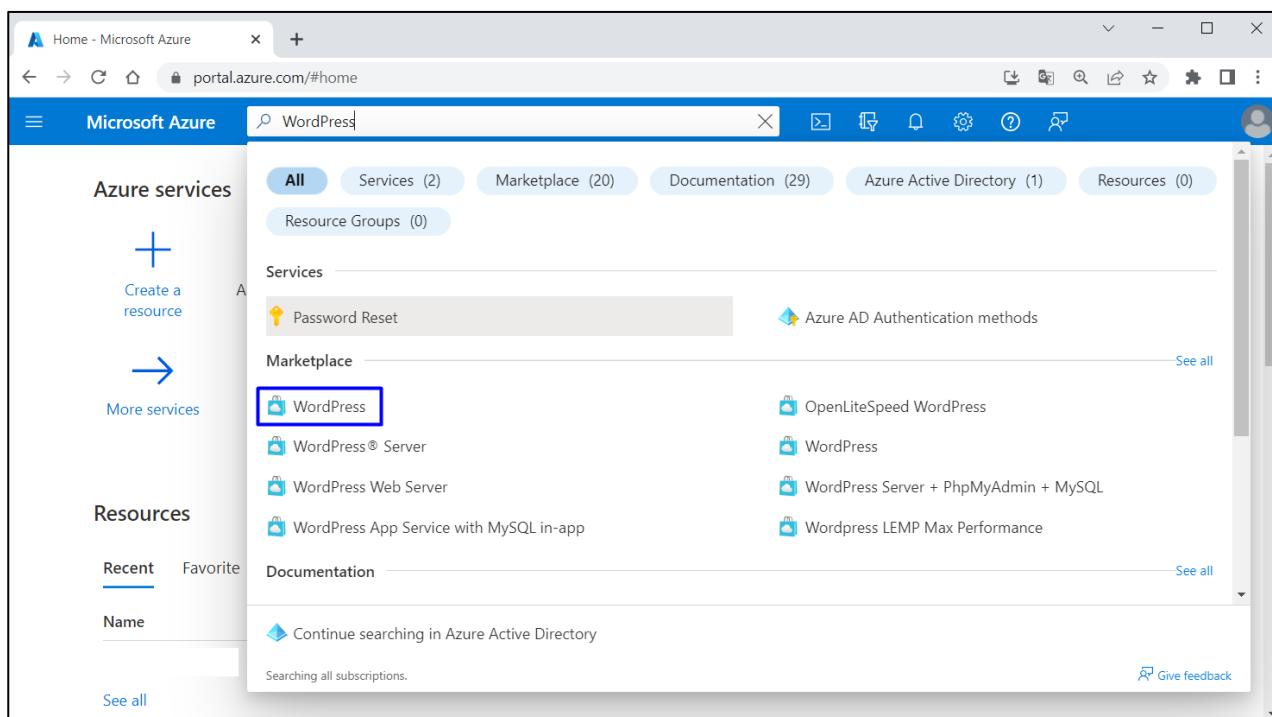
In this exercise we will **explore Azure services** and **Azure Portal** and will learn how to **deploy apps to Azure App Service** in different ways.

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 filter buttons: "All", "Services (2)", "Marketplace (20)", "Documentation (29)", "Azure Active Directory (1)", and "Resources (0)". The "Marketplace" section is expanded, displaying a list of items. The first item in the list, "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 sections for "Create a resource" (with a plus icon), "More services" (with a right arrow icon), and "Resources" (with a recent and favorite tab). The "Recent" tab is selected. There is also a "Name" search input field and a "See all" link. At the bottom of the page, there is a footer with a "Give feedback" link.

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>

The screenshot shows the 'Create WordPress on App Service' wizard in the Microsoft Azure portal. The 'Basics' tab is selected. In the 'Project details' section, the 'Subscription' dropdown is set to 'Azure for Students'. Below it, the 'Resource group' dropdown shows '(New) Resource group' with a 'Create new' link. At the bottom, there are navigation buttons: 'Review + create' (highlighted in blue), '< Previous', and 'Next : Advanced >'.

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

The screenshot shows the 'Create WordPress on App Service' wizard in the Microsoft Azure portal. The 'Basics' tab is selected. In the 'Project details' section, the 'Subscription' dropdown is highlighted with a purple box and contains 'Azure for Students'. Below it, the 'Resource group' dropdown shows '(New) WordPressResourceGroup' with a 'Create new' link. At the bottom, there are navigation buttons: 'Review + create' (highlighted in blue), '< Previous', and 'Next : Advanced >'.

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

The screenshot shows a 'Create new resource group' dialog. The 'Resource group' dropdown is highlighted with a purple box and contains '(New) WordPressResourceGroup'. Below it, there is a 'Create new' link. The entire dialog is enclosed in a light gray border.

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

The screenshot shows the 'Hosting details' section. It includes a note: 'Select the region for your server and provide a name for Web App. Custom domains can be added later.' Below is a 'Region' dropdown highlighted with a purple box, containing 'West Europe'. At the bottom, there is a horizontal line with a copyright notice and social media links.

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	Basic Basic App Service, Burstable MySQL database Change plan

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**]:

Review + create	< 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

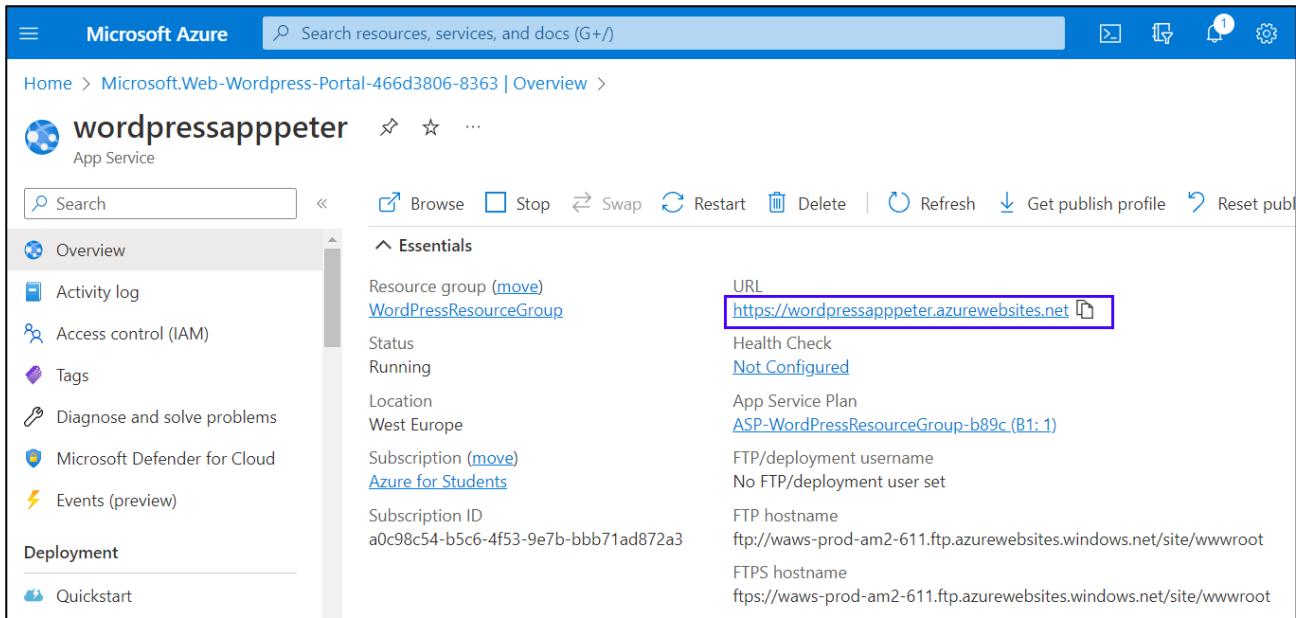
Deployment name: Microsoft.Web-Wordpress-Porta... Start time: 1/6/2023, 10:26:03 AM
Subscription: Azure for Students Correlation ID: 45107c2c-70a3-4772-b0b8-63a93t
Resource group: WordPressResourceGroup

Deployment details Next steps

Go to resource

Step 2: Browse Site

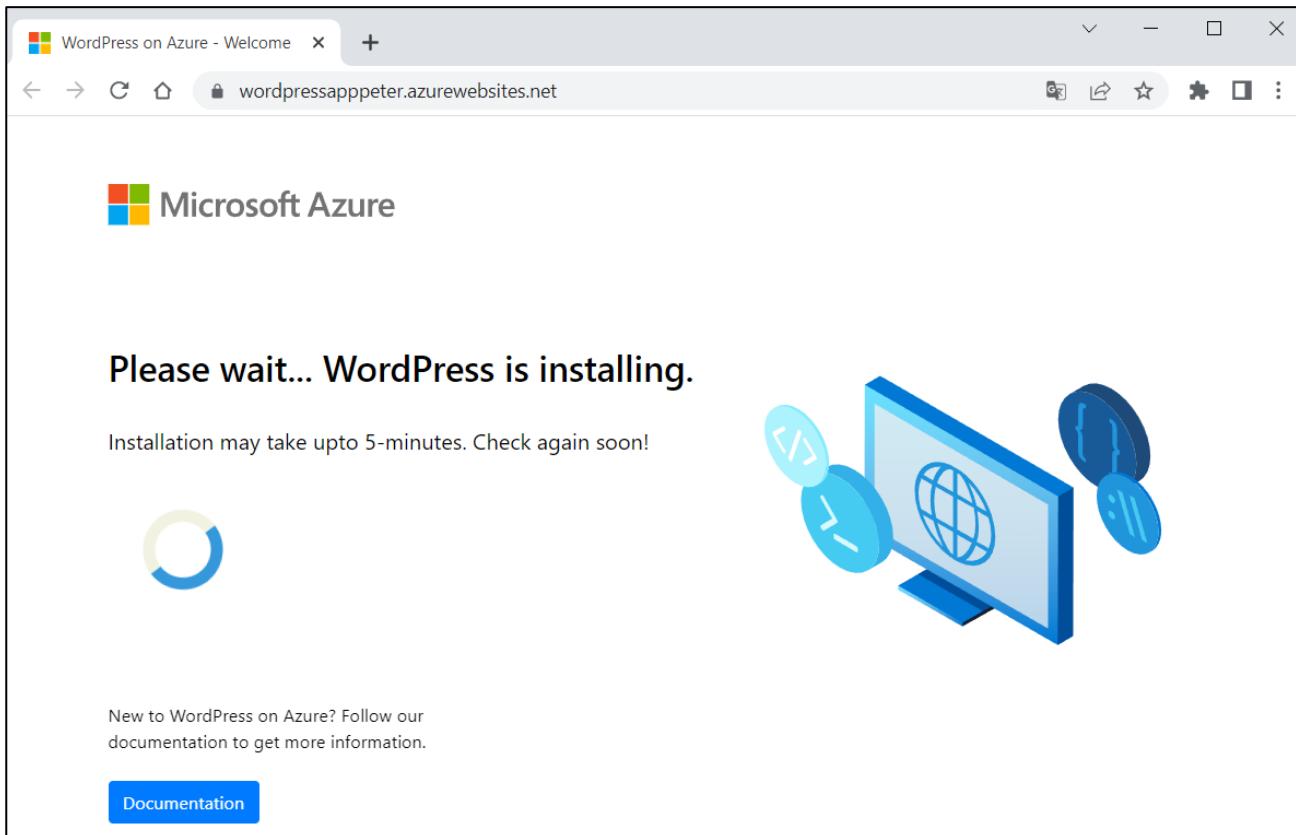
You can click on the URL of the site in the "App Service" page, in the [Overview] tab to access the site:



The screenshot shows the Microsoft Azure portal with the search bar at the top. Below it, the navigation bar includes Home, Microsoft.Web-Wordpress-Portal-466d3806-8363 | Overview, and a gear icon for settings. The main content area displays the 'wordpressapppeter' App Service details. On the left, a sidebar lists Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Deployment, and Quickstart. The 'Overview' section is expanded, showing the Resource group (WordPressResourceGroup), Status (Running), Location (West Europe), Subscription (Azure for Students), and Subscription ID (a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3). On the right, the 'Essentials' section shows the URL <https://wordpressapppeter.azurewebsites.net>, which is highlighted with a blue box.

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 web browser window with the title 'WordPress on Azure - Welcome'. The address bar contains the URL <https://wordpressapppeter.azurewebsites.net>. The page content includes the Microsoft Azure logo, a large progress bar, and the text 'Please wait... WordPress is installing.' Below this, it says 'Installation may take upto 5-minutes. Check again soon!'. To the right, there's an illustration of a computer monitor displaying a globe and code snippets (</>, < />, {}, {}, //). At the bottom, it says 'New to WordPress on Azure? Follow our documentation to get more information.' and features a blue 'Documentation' button.

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:

WordPress on Azure

https://wordpressapppeter.azurewebsites.net/wp-admin

Log In < WordPress on Azure

wordpressapppeter.azurewebsites.net/wp-login....

Username or Email Address

Password

Remember Me

Lost your password?
← Go to WordPress on Azure

The screenshot shows the WordPress dashboard. At the top, there's a banner for Smush! version 6.1.1. Below it, a large blue banner says "Welcome to WordPress!" with a link to learn more about the 6.1 version. The left sidebar contains links for Home, Posts, Media, Pages, Comments, Appearance, Plugins (2), Users, Tools, Settings, Performance (2), and Smush.

Step 3: Access the Database

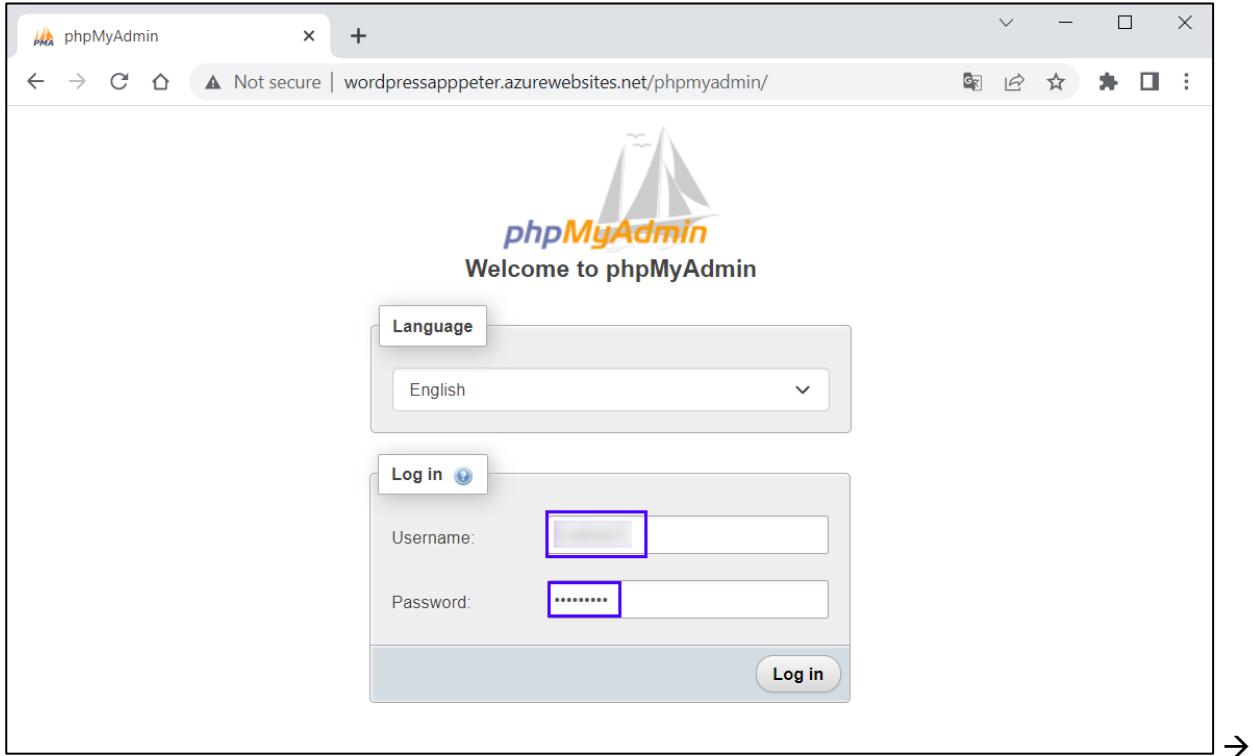
You can also **access the MySQL Flexible Server database but not directly** because it is created behind a **private Virtual Network**.

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 "Configuration" tab is selected. A table lists application settings:

Name	Value	Source
DATABASE_HOST	(Hidden value. Click to show value)	App Service
DATABASE_NAME	(Hidden value. Click to show value)	App Service
DATABASE_PASSWORD	(Hidden value. Click to show value)	App Service
DATABASE_USERNAME	(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:



A screenshot of the phpMyAdmin dashboard. The left sidebar shows a tree view of databases: "New", "information_schema", "mysql", "performance_schema", "sys", and "wordpressa_598239e2a6424ab4". The main panel has two tabs: "General settings" and "Appearance settings". The "General settings" tab shows "Change password", "Server connection collation" set to "utf8mb4_unicode_ci", and a "More settings" link. The "Appearance settings" tab shows "Language" set to "English" and "Theme" set to "pmahomme". A "Console" tab is visible at the bottom.

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 summary: 'Software University (SoftUni) (softwareuniversity.onmicrosoft.com)' with 0 unsecure resources and 0 recommendations. There are buttons for 'Create', 'Manage view', 'Refresh', 'Export to CSV', 'Open query', 'Assign tags', and 'Delete'. A filter bar at the top allows filtering by 'Subscription equals all', 'Resource group equals all', 'Type equals all', and 'Add filter'. A dropdown menu shows 'No grouping'. The main area is a 'List view' showing the following resources:

	Name	Type	Resource group	Location	Subscription
<input type="checkbox"/>	ASP-WordPressResourceGroup-b89c	App Service plan	WordPressResourceGr...	West Europe	Azure for Students
<input type="checkbox"/>	wordpressa-598239e2a6424ab4adf35d06f0...	Azure Database for My...	WordPressResourceGr...	West Europe	Azure for Students
<input type="checkbox"/>	wordpressa-7bd3e9d2bd-privatelink.mysql....	Private DNS zone	WordPressResourceGr...	Global	Azure for Students
<input type="checkbox"/>	wordpressa-7bd3e9d2bd-vnet	Virtual network	WordPressResourceGr...	West Europe	Azure for Students
<input type="checkbox"/>	wordpressappeter	App Service	WordPressResourceGr...	West Europe	Azure for Students

At the bottom, there are navigation buttons for '< Previous', 'Page 1 of 1', 'Next >', and 'Showing 1 to 5 records.' A 'Give feedback' link is also present.

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 search results for 'Resource groups':

- All (29)
- Services (29)
- Marketplace (1)
- Documentation (29)
- Resources (0)
- Resource Groups (0)
- Azure Active Directory (0)

A blue arrow points to the 'Resource groups' link in the search results.

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

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 ↑↓ (WordPressResourceGroup) Subscription ↑↓ Azure for Students Location ↑↓ 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+/)

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+/)

Home > Resource groups TaskBoardResourceGroup Resource group

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

Settings Deployments Security Policies Properties Locks

+ Create Manage

^ Essentials

Subscription (move) Azure for Students Deployment ID a0c98c54-b5c6-4f53-9e7b Tags (edit) Click here to add tags

Resources Recom Filter for any field... Showing 1 to 5 of 5 records List view Name ↑↓

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

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.

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

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

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with 'WordpressResourceGroup' selected. The main area displays a confirmation dialog box titled 'Are you sure you want to delete "WordP..."' with a warning message: 'Warning! Deleting the "WordpressResourceGroup" 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.' Below this, a text input field says 'TYPE THE RESOURCE GROUP NAME:' with 'WordpressResourceGroup' typed in. A table titled 'AFFECTED RESOURCES' lists six resources: 'ASP-WordpressResourceGro...', 'wordpressa-598239e2a6424a...', 'wordpressa-7bd3e9d2bd-pr...', and 'wordpressa-7bd3e9d2bd-pr...'. At the bottom of the dialog are 'Delete' and 'Cancel' buttons, with 'Delete' highlighted.

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 web browser window with the URL 'todoappeter.azurewebsites.net'. The page title is 'Todo List' and it's described as an 'ExpressJS/MongoDB Sample Application'. The interface includes a 'New task' input field and a 'Add task' button. Below this, there are sections for 'Current tasks' and 'Completed tasks'. The 'Current tasks' section lists two items: 'New Task' and 'Future Task', each with 'Created' dates of '2023-01-06' and 'Action' buttons for 'Complete' and 'Delete'. The 'Completed tasks' section lists one item: 'Finished Task' with a 'Created' date of '2023-01-06' and a 'Completed' status.

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'. The repository is public and has 1 branch and 0 tags. The main commit is 'Initial commit' made 2 minutes ago. The commit history includes several files: config, models, public/stylesheets, routes, views, .env.sample, LICENSE, README.md, app.js, package-lock.json, and package.json. All commits are initial commits made 2 or 3 minutes ago. The repository has 2 commits, 0 stars, 1 watching, and 0 forks. It also has no releases published and no packages published.

File	Commit	Time Ago
config	Initial commit	2 minutes ago
models	Initial commit	2 minutes ago
public/stylesheets	Initial commit	2 minutes ago
routes	Initial commit	2 minutes ago
views	Initial commit	2 minutes ago
.env.sample	Initial commit	2 minutes ago
LICENSE	Initial commit	3 minutes ago
README.md	Initial commit	3 minutes ago
app.js	Initial commit	2 minutes ago
package-lock.json	Initial commit	2 minutes ago
package.json	Initial commit	2 minutes ago

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 16 LTS]** as **runtime stack**. Don't change the **default selected database engine (Cosmos DB for MongoDB)** and **copy the generated database name** because you will need it later.

Create Web App + Database

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]

Create new

Region *

[REDACTED]

Web App Details

Name *

[REDACTED] ✓

.azurewebsites.net

Runtime stack *

Node 16 LTS

Database

ⓘ 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] ✓

Hosting

Hosting plan *

Basic - For hobby or research purposes

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

Value

Deployment slot setting

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

Add/Edit connection string

Name	<input type="text" value="MONGODB_URI"/>	<input type="button" value="X"/>
Value	<input type="text" value="mongodb://"/> <input type="button" value="Copy"/>	<input type="button" value="X"/>
Type	<input type="text" value="Custom"/> <input type="button" value="▼"/>	
<input type="checkbox"/> Deployment slot setting		

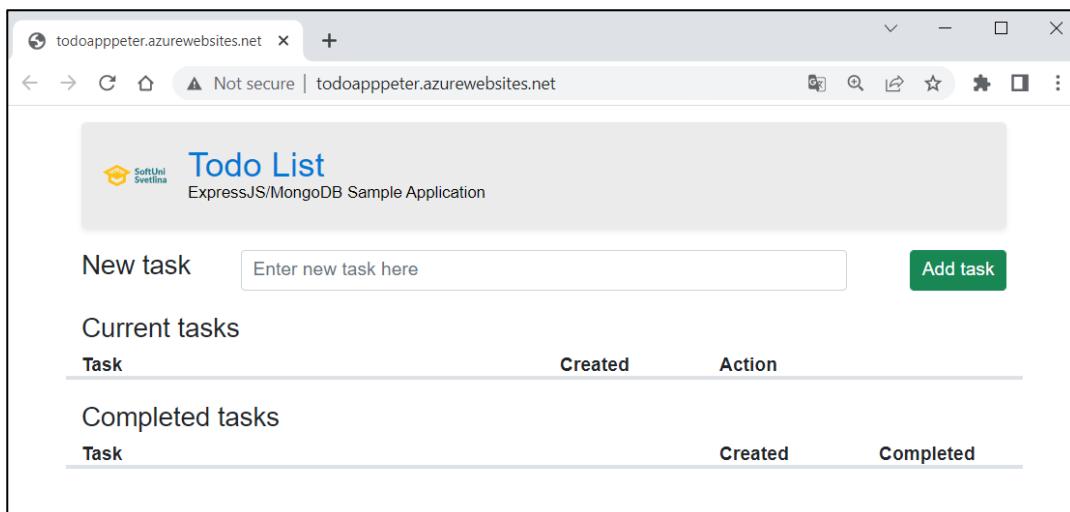
5. Create one more **new application setting** with:
 - **Name:** "DATABASE_URL"
 - **Value:** the "MONGODB_URI" **connection string** you copied earlier (i.e. `mongodb://...`)

Add/Edit application setting

Name	<input type="text" value="DATABASE_URL"/>	<input type="button" value="X"/>
Value	<input type="text" value="mongodb://"/> <input type="button" value="Copy"/>	<input type="button" value="X"/>
<input type="checkbox"/> 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**.



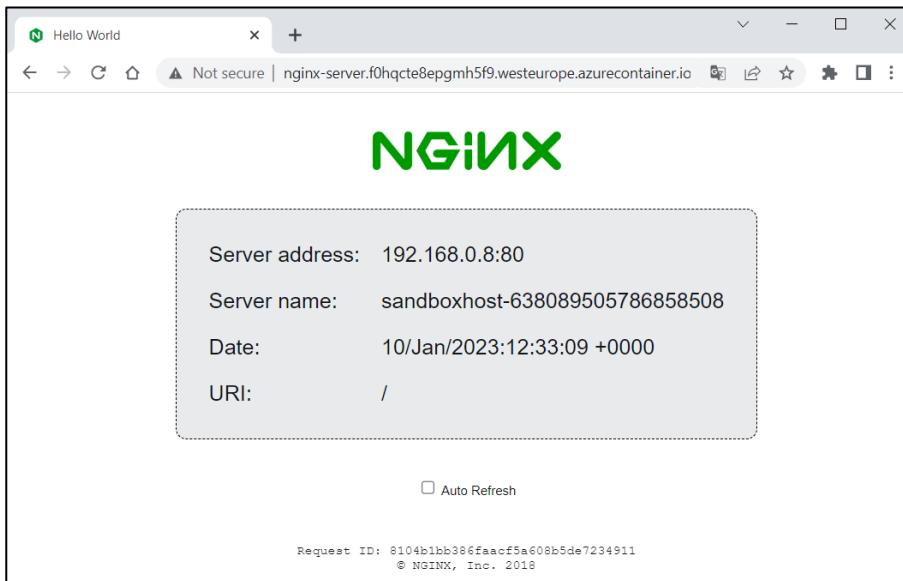
NOTE: Don't forget to delete the resources in your Azure account.

3. Deploy NGINX Server Container to Azure using Azure Portal

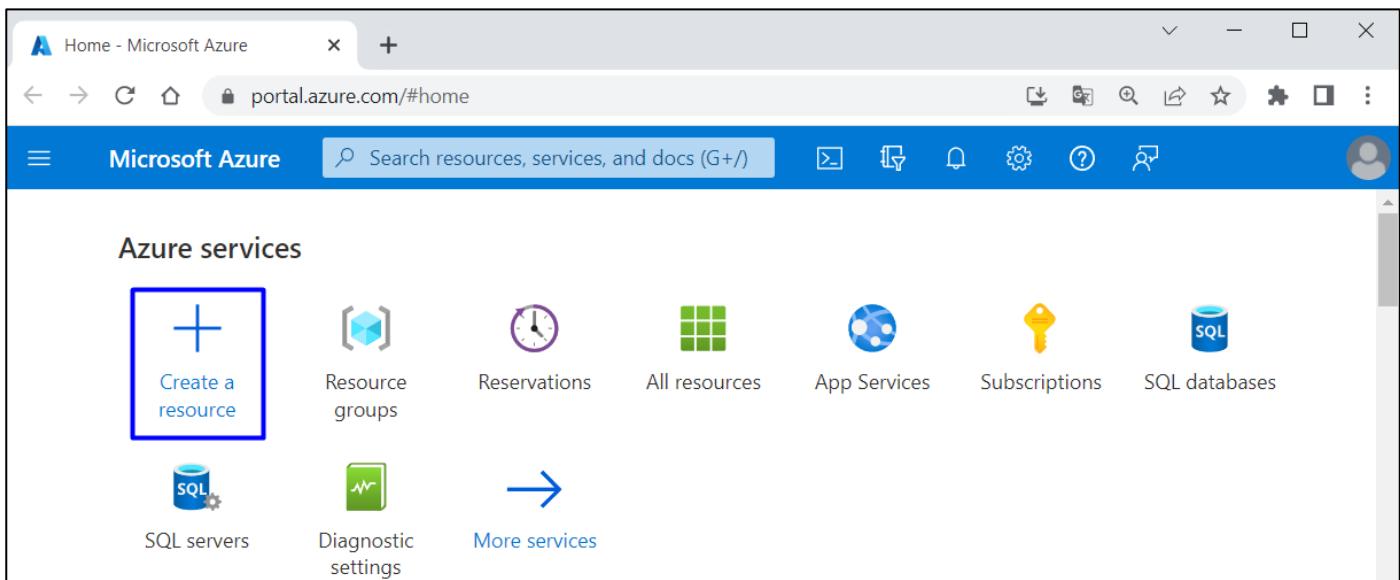
In this task, we will **deploy a container instance to Azure Container Instances** using the **Azure Portal**.

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]**:



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

Get Started

Recently created

Categories

- AI + Machine Learning
- Analytics
- Blockchain
- Compute
- Containers**
- Databases
- Developer Tools
- DevOps
- Identity
- Integration
- Internet of Things
- IT & Management Tools

Popular Azure services [See more in All services](#)

Popular Marketplace products [See more in Marketplace](#)

- Azure Kubernetes Service (AKS) [Create](#) | [Docs](#) | [MS Learn](#)
- Web App for Containers [Create](#) | [Docs](#) | [MS Learn](#)
- Batch Service [Create](#) | [Docs](#) | [MS Learn](#)
- Kubernetes - Azure Arc [Create](#) | [Docs](#) | [MS Learn](#)
- Container App [Create](#) | [Docs](#)
- Container Instances** [Create](#) | [Learn more](#) | [MS Learn](#)
- NVIDIA GPU-Optimized Image for AI & HPC - v21.06.0 [Create](#) | [Learn more](#)
- Windows Server 2022 Core Datacenter Minimal OS [Create](#) | [Learn more](#)
- intel. Basic [Create](#) | [Learn more](#)
- Hyper-V Server on Windows Server 2016 [Create](#) | [Learn more](#)
- Docker Engine Community on Ubuntu 20.04 LTS [Create](#) | [Learn more](#)
- Docker Compose Server on Windows Server 2016 [Create](#) | [Learn more](#)

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](#)

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

Container details

Container name *	nginx-server-container
Region *	(Europe) West Europe
Availability zones	None
Image source *	<input type="radio"/> Quickstart images <input type="radio"/> Azure Container Registry <input checked="" type="radio"/> Other registry
Image type *	<input checked="" type="radio"/> Public <input type="radio"/> Private
Image *	nginxdemos/hello
<small>If not specified, Docker Hub will be used for the container registry and the latest version of the image will be pulled.</small>	
OS type *	<input checked="" type="radio"/> Linux <input type="radio"/> Windows
<small>This selection must match the OS of the image chosen above.</small>	
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:

Create container instance

Basics	Networking	Advanced	Tags	Review + create			
Choose between three networking options for your container instance:							
<ul style="list-style-type: none"> '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	<input checked="" type="radio"/> Public <input type="radio"/> Private <input type="radio"/> None						
DNS name label	nginx-server						
DNS name label scope reuse *	Tenant						
Ports	<table border="1"> <thead> <tr> <th>Ports</th> <th>Ports protocol</th> </tr> </thead> <tbody> <tr> <td>80</td> <td>TCP</td> </tr> <tr> <td><input type="text"/></td> <td><input type="text"/></td> </tr> </tbody> </table>	Ports	Ports protocol	80	TCP	<input type="text"/>	<input type="text"/>
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**:

The screenshot shows the Microsoft Azure Container Instances Overview page for a deployment named "Microsoft.ContainerInstances-20230110142813". The status is "Your deployment is complete". Deployment details include: Deployment name: Microsoft.Container..., Start time: 1/10/2023, 2:29:25 PM; Subscription: Azure for Students, Correlation ID: 89de96e5-fd0a-4858-8f. Below the main message are sections for "Deployment details" and "Next steps", with a "Go to resource" button.

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

The screenshot shows the Microsoft Azure Container instances page. In the search bar, "Container" is typed. The results show "Container instances" highlighted with a blue border. The main table lists one container instance: "nginx-server" (Resource group: "nginxServerResourceGroup", Location: "West Europe", Status: "Running", OS type: "Linux", Total c...: "1", Subscription: "Azure for Students").

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

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

Home > Container instances >

nginx-server

Container instances

Search Start Restart Stop Delete Refresh

Overview Activity log Access control (IAM) Tags

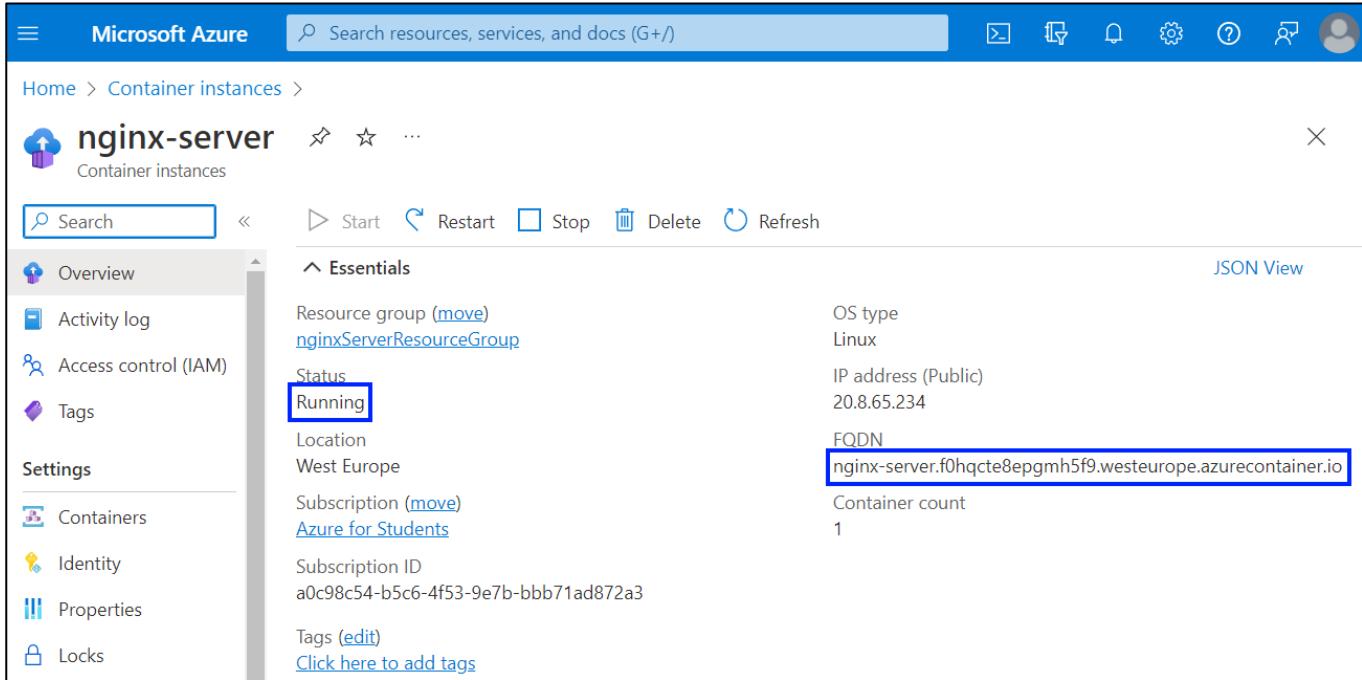
Settings

Containers Identity Properties Locks

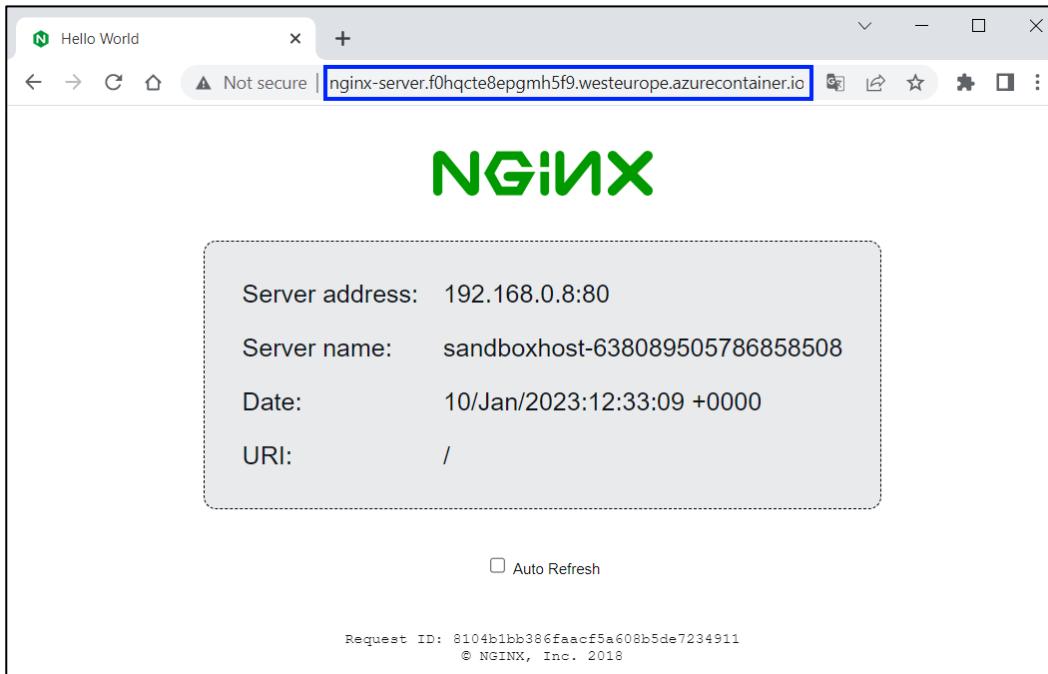
Essentials

Resource group ([move](#)) [nginxServerResourceGroup](#) OS type Linux
Status **Running** IP address (Public) 20.8.65.234
Location West Europe FQDN [nginx-server.f0hqcte8epgmh5f9.westeurope.azurecontainer.io](#)
Subscription ([move](#)) [Azure for Students](#) Container count 1
Subscription ID a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3
Tags ([edit](#)) [Click here to add tags](#)

JSON View



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



The browser window displays the NGINX logo and some server information:

Server address: 192.168.0.8:80
Server name: sandboxhost-638089505786858508
Date: 10/Jan/2023:12:33:09 +0000
URI: /

At the bottom, it shows Request ID: 8104b1bb386faacf5a608b5de7234911 and © NGINX, Inc. 2018.

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]:

The screenshot shows the Microsoft Azure Container Instances interface. On the left, a sidebar lists navigation options: Home, Container instances, Overview, Activity log, Access control (IAM), Tags, Settings, Containers (which is selected and highlighted with a blue border), Identity, Properties, Locks, Monitoring, Metrics, and Alerts. The main content area displays a table of containers. There is one container listed: "nginx-server" (Image: nginxdemos/hello, State: Running, Previous state: -, Start time: 2023-01-10T12:30:06..., Restart count: 0). Below the table, there are tabs for Events, Properties, Logs (which is selected and highlighted with a blue border), and Connect. The Logs tab displays the container's log output, which includes messages about Docker entrypoint scripts, configuration, and startup.

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 interface. The sidebar on the left is identical to the previous screenshot. The main content area shows the "nginx-server" container details. The "Delete" button in the top toolbar is highlighted with a blue border. Below the toolbar, the "Essentials" section provides information such as Resource group (nginxServerResourceGroup), OS type (Linux), IP address (Public) 20.8.65.234, FQDN (nginx-server.f0hqcte8epgmh5f9.westeurope.azurecontainer.io), Container count (1), and other details like Status (Running), Location (West Europe), Subscription (Azure for Students), and Subscription ID (a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3). A "JSON View" link is also present.

4. Deploy NGINX Server Container to Azure using Azure CLI

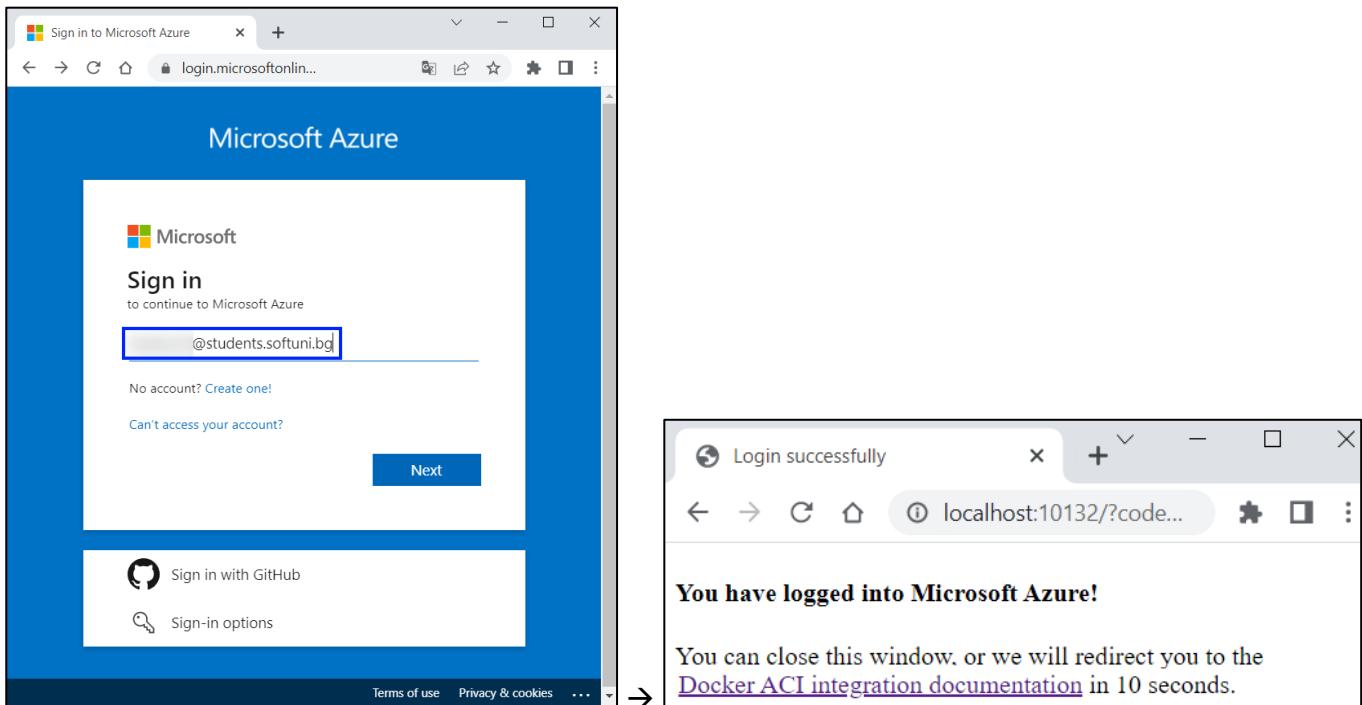
Now we will see how to **deploy a NGINX server container to Azure** but using **Azure CLI commands**.

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**:



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_git20
220924-r4)
2023/01/11 08:48:34 [notice] 19#19: OS: Linux 5.10.102.2-microsoft-standard
```

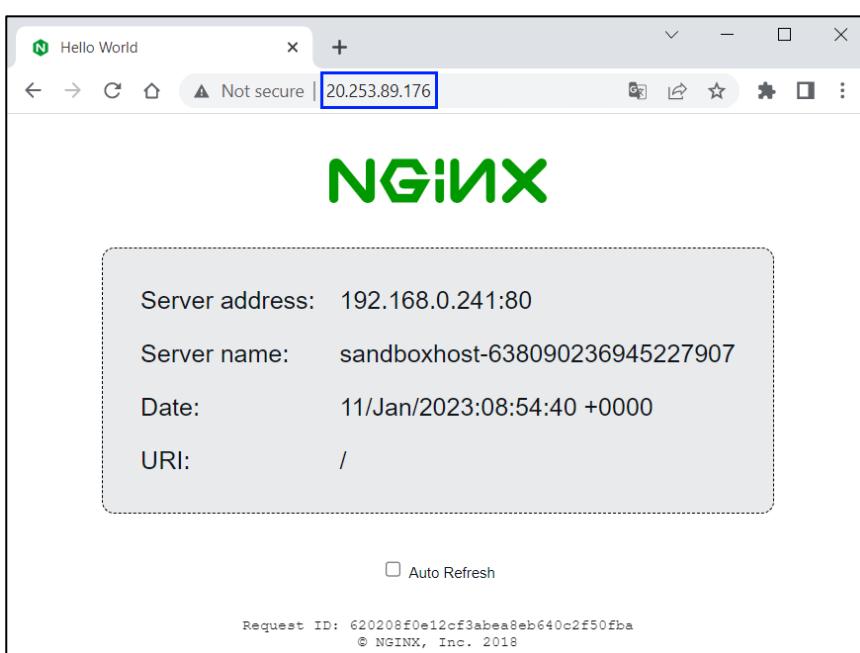
```
2023/01/11 08:48:34 [notice] 19#19: getrlimit(RLIMIT_NOFILE): 1024:1048576
2023/01/11 08:48:34 [notice] 19#19: start worker processes
2023/01/11 08:48:34 [notice] 19#19: start worker process 37
```

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"     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 portal interface for a container instance named 'pedantic-brahmagupta'. The left sidebar has 'Container instances' selected. The main area displays the instance details under the 'Essentials' tab. Key information includes:

Resource group (move)	OS type
6c71f2b8-547e-2cf3-ab83-4336e4556bc8	Linux
Status	IP address (Public)
Running	20.253.89.176
Location	FQDN
East US	---
Subscription (move)	Container count
Azure for Students	1
Subscription ID	
a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3	
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\xA5\x09\x19\x17\x01\x88\xAC\xB2\x83\xA9\xAD\x9C\x02\xA5" 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

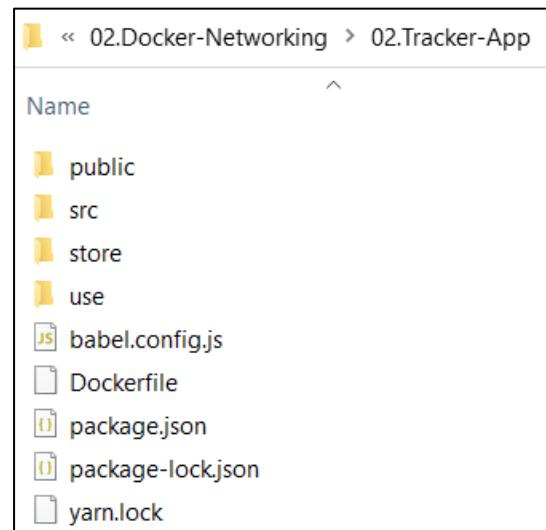
Your task now is to **run a simple JavaScript front-end app based on Vue.js** for keeping track of daily duties in a **Docker container**. It does not need **anything but an image** to run. It does not use a database or any other storage.

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:

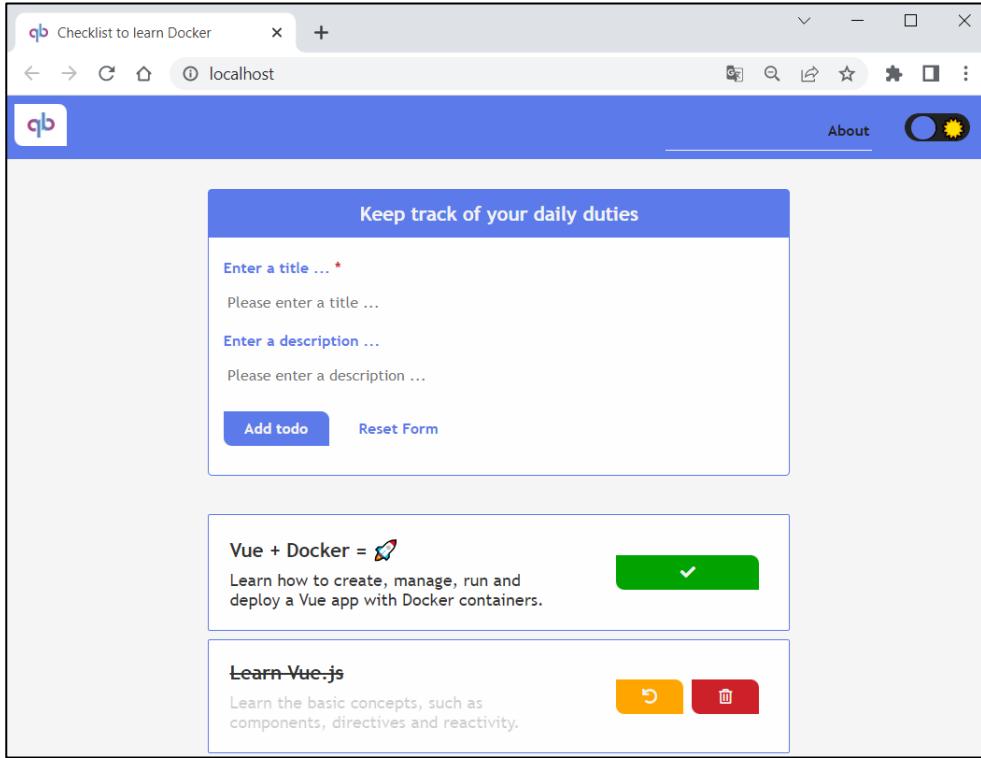
```
PS D:\SoftUni\03.Tracker-App> docker build -t tracker-app-image .
[+] Building 0.7s (14/14) FINISHED
...
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn
how to fix them
```

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 -it 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 header, there's a "Search resources, services, and docs (G+)" bar and a user profile icon. The main content area displays a table of container registries. The first row in the table is highlighted with a blue border, showing the following details:

Name	Type	Resource group	Location	Subscription	...
trackerapprcr	Container registry	trackerapprg	West Europe	Azure for Students	...

At the bottom of the table, there are navigation buttons: < Previous, Page 1 of 1, Next >, and a "Give feedback" link.

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 is titled 'trackerappcr' and shows it's a 'Container registry'. On the left, there's a sidebar with links like Overview, Activity log, Access control (IAM), Tags, Quick start, and Events. The main pane is titled 'Essentials' and provides detailed information about the resource group, location, subscription, and provisioning status. A 'JSON View' link is also present.

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' being the only item currently visible, also highlighted with a blue border.

The screenshot shows the Microsoft Azure portal interface. The URL in the address bar is portal.azure.com/#view/Microsoft_Azure_ContainerRegistries/RepositoryBlade.... The page title is "tracker-app-image - Microsoft Azure". The main content area displays the "tracker-app-image" repository details. It includes a "Repository" section with a refresh button, a "Manage deleted artifacts" link, and a "Delete repository" button. Below this is an "Essentials" section with a "JSON View" link. The essentials table shows the following data:

Repository	Tag count
tracker-app-image	1
Last updated date	Manifest count
1/11/2023, 2:54 PM GMT+2	1

Below the table is a search bar with the placeholder "Search to filter tags ...". Underneath the search bar are three columns: "Tags ↑↓", "Digest ↑↓", and "Last modified". A single tag entry is shown: "v1" with digest "sha256:7c4287352e7d3ff0462310c25ecb9d11cb2..." and last modified "1/11/2023, 2:54 PM GMT+2". There is also a "..." button.

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 'Container registries' blade in the Azure portal. It displays a single record for a container registry named 'trackerappcr'. The details shown are: Type: Container registry, Resource group: trackerapprg, Location: West Europe, Subscription: Azure for Students. The 'trackerappcr' entry is highlighted with a blue border.

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 (which is turned on). Below that, there are two sets of access keys: 'password' and 'password2'. Each key has a 'Regenerate' button next to it.

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
    }
  ]
}
```

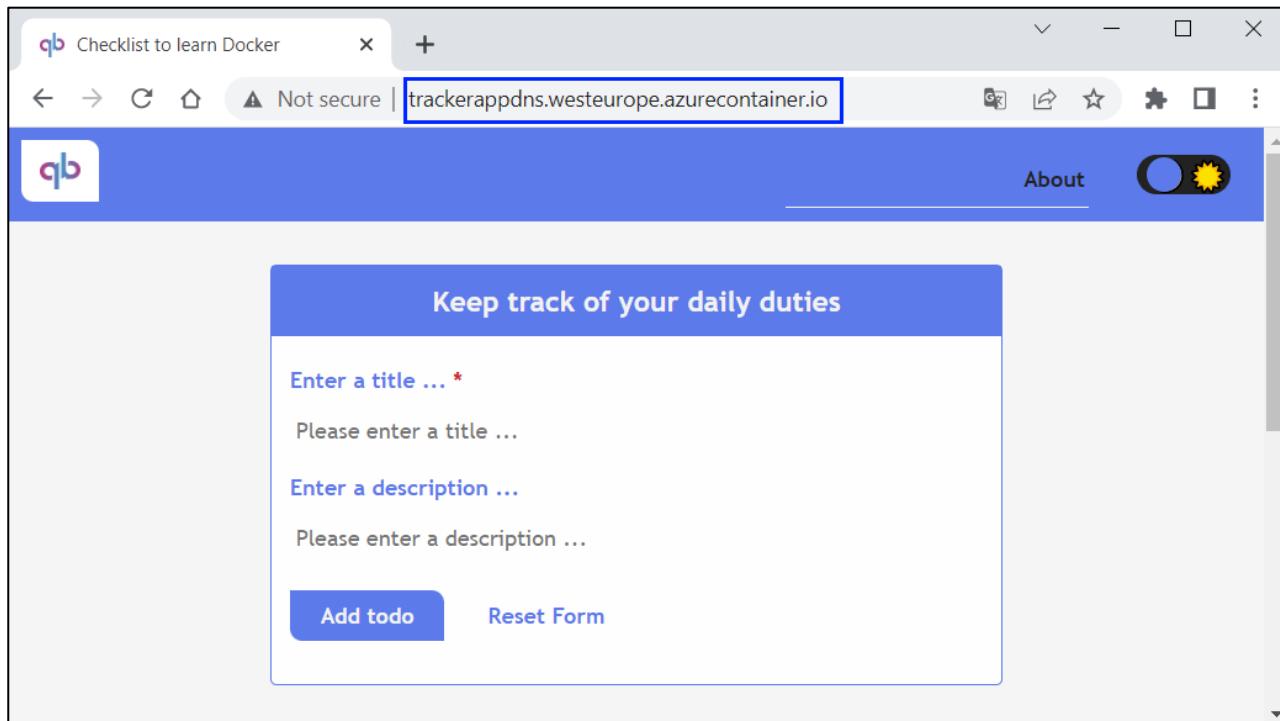
To view the container state of deployment, use:

```
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 ↑
trackerapp	Container instances	trackerapprg	West Europe	Azure for Students
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 Activity log Access control (IAM) Tags

Essentials

Resource group ([move](#)) trackerapprg OS type Linux
Status Running IP address (Public) 20.23.87.25
Location West Europe FQDN trackerapppdns.westeurope.azurecontainer.io
Subscription ([move](#)) Azure for Students Container count 1
Subscription ID a0c98c54-b5c6-4f53-9e7b-bbb71ad872a3

JSON View

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**:

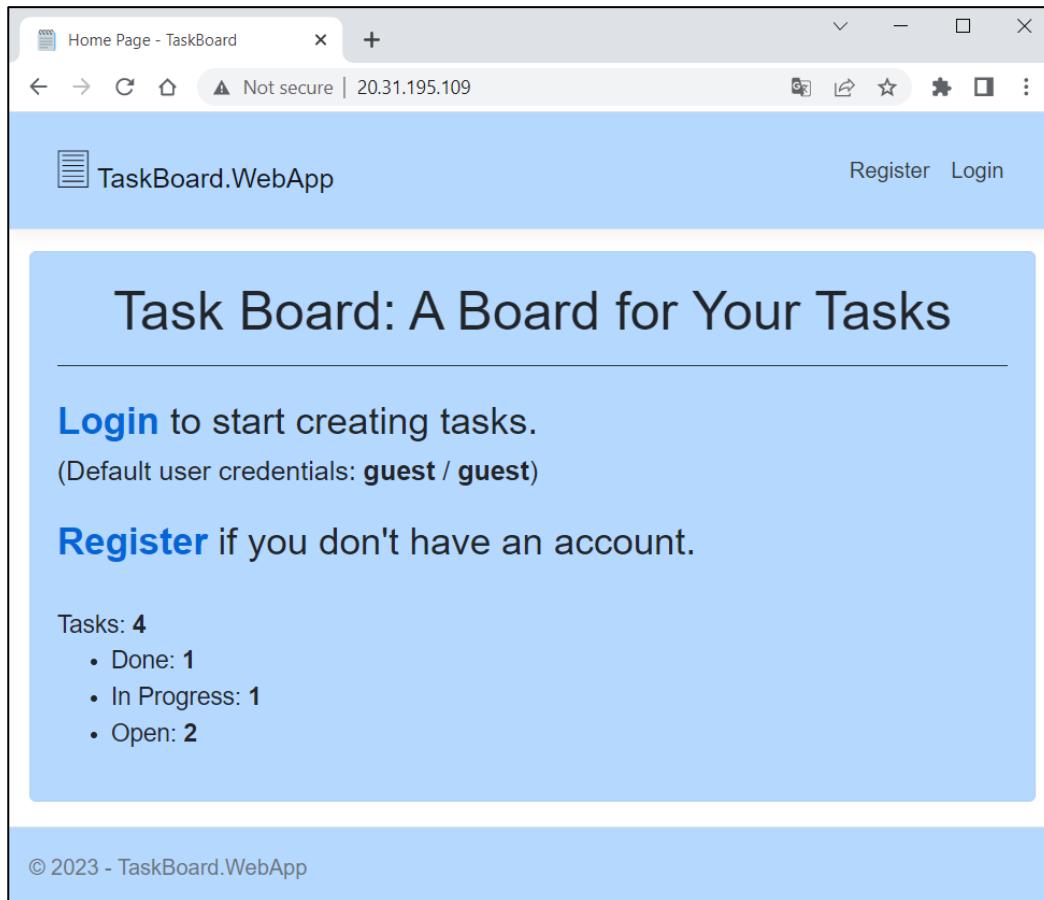
```
PS D:\SoftUni\03.Tracker-App> az group delete --name trackerapprg
Are you sure you want to perform this operation? (y/n): y
```

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

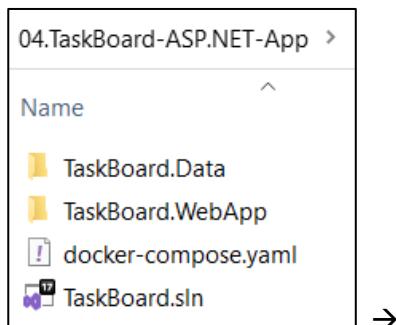
6. Deploy the "TaskBoard" App using Docker Compose

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 "**taskboardcr**". 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": "taskboardcr.azurecr.io",
  "name": "taskboardcr",
  ...
  "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 Recommendations

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

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:

The screenshot shows a web browser window with the title "Home Page - TaskBoard". The address bar says "localhost". The page header includes the logo "TaskBoard.WebApp", "Register", and "Login". The main content area has a large heading "Task Board: A Board for Your Tasks". Below it, there are two sections: "Login to start creating tasks. (Default user credentials: guest / guest)" and "Register if you don't have an account.". A summary section below lists "Tasks: 4" with a breakdown: "Done: 1", "In Progress: 2", and "Open: 1". At the bottom of the page is a footer with the text "© 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 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 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 search bar for tags. A JSON View link is also present. The table below lists the tag 'latest' with its digest and last modified date.

Tags ↑	Digest ↑↓	Last modified
latest	sha256:61199f343c8bc734416959689040a357ba...	2/9/2023, 2:18 PM GMT+2

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:

```
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
```

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> Using only available 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>  
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:

```
PS D:\SoftUni\TaskBoard-ASP.NET-App> docker ps
CONTAINER ID        IMAGE               STATUS    PORTS
taskboard-aspnet-app_sqlserver mcr.microsoft.com/m... Running  20.23.53.217:1433->1433/tcp
taskboard-aspnet-app_web-app   taskboardcr.azurecr... 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

A screenshot of the Microsoft Azure Storage account settings for 'taskboardstorageacc'. The left sidebar shows 'File shares' selected. The main pane displays 'File share settings' with 'sql-volume' listed under 'Name'. Other details include Active Directory: Not configured, Default share-level permissions: Disabled, Soft delete: 7 days, Maximum capacity: 5 TiB, and Security: Maximum compatibility.

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:

The screenshot shows a web browser window titled 'Home Page - TaskBoard'. The URL bar contains 'Not secure | 20.31.195.109'. The page itself is titled 'TaskBoard.WebApp' and features a large blue header with the text 'Task Board: A Board for Your Tasks'. Below this, there's a 'Login' section with the text 'Login to start creating tasks.' and '(Default user credentials: guest / guest)'. There's also a 'Register' section with the text 'Register if you don't have an account.'. At the bottom, a summary of tasks is provided: 'Tasks: 4' with items: '• Done: 1', '• In Progress: 1', and '• Open: 2'. The footer of the page includes the copyright notice '© 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 two windows. On the left is a file explorer window titled "05.Posio-App" showing the project structure:

- Name
- ansible
- app
- data
- docker
- posio
- config.py
- docker-compose.yml
- Dockerfile
- requirements.txt
- run.py

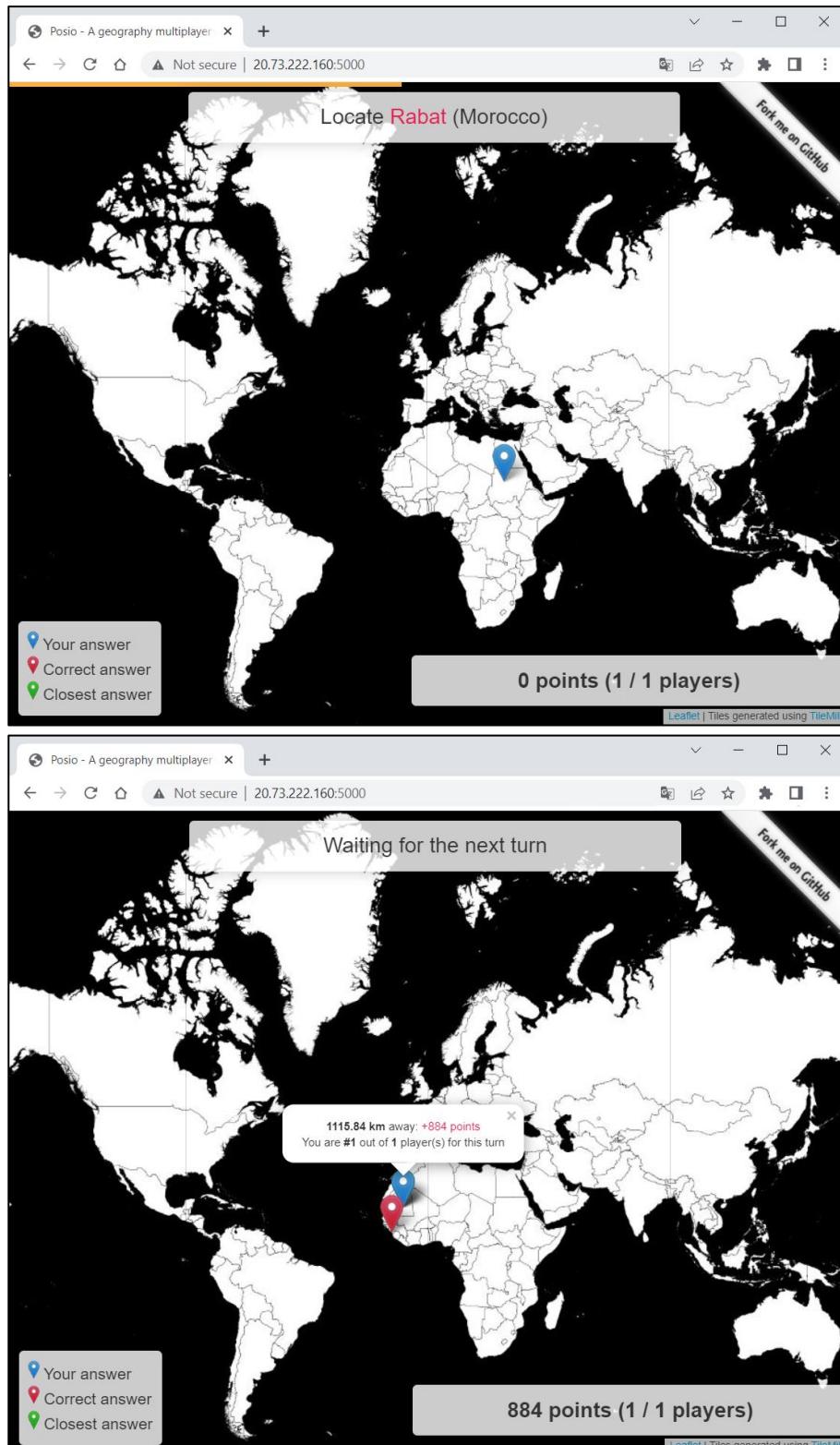
On the right is a browser window showing the "Posio - A geography multiplayer" game. It features a world map with a message: "You'll be given a list of cities that you must locate by clicking on the map". Below the map is a player input field with "Name your player" and a "Play" button. At the bottom, there's a legend: "Your answer", "Correct answer", and "Closest answer", and a score indicator showing "0 points". The browser address bar shows the URL: 20.73.222.160:5000.

The screenshot shows the Microsoft Azure portal interface for the "posio-app" container instance. The left sidebar has sections for Overview, Activity log, Access control (IAM), Tags, Settings (Containers, Identity, Properties, Locks), and Monitoring. The main pane displays the "Essentials" section with the following details:

Essentials	
Resource group (move)	: posioResourceGroup
Status	: Running
Location	: West Europe
Subscription (move)	: 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 (edit)	: 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**.