

Scenario

You are a Python developer who has just completed a fun Mad Libs application (a word game where one player prompts another for a list of words to substitute for blanks in a story). You want to deploy this application to the cloud to showcase your skills in your portfolio and allow your friends to easily access and enjoy the game. This will involve setting up Azure CLI, preparing an application for deployment, and deploying it to a cloud platform.

Objective

The objective of this activity is to guide you through the process of deploying a Flask application to the cloud using Microsoft Azure. This will involve setting up the Azure command-line interface (CLI), preparing your application for deployment, and successfully deploying it to the Azure platform. By the end of this activity, you will have a functional web application accessible to users via the cloud, showcasing your development skills and allowing others to interact with your creation.

Instructions


Download the mad libs application and save it on your machine's desktop.

Step 1: Set up Azure CLI

Step 1.1: Download and Install Azure CLI

The Azure CLI allows you to manage Azure resources from the command line. It is a powerful tool that can be used to automate tasks. The Azure CLI is often easier to use than the Azure portal for tasks that require automation or that need to be done repeatedly. In this example, you will use the Azure CLI to deploy an application to Azure.

Installation Steps:

1. Visit the Azure CLI page: <https://learn.microsoft.com/en-us/cli/azure/install-azure-cli> 
2. Click the "Install" link for your operating system.
3. Locate and download the latest version of the installer.
4. Once the download has completed, double-click the downloaded installer file in your web browser's Downloads folder.
5. Accept the license terms and choose the default installation location.
6. Open a new Terminal or a command prompt and type: **az --version**
7. If the Azure CLI was installed correctly, you will see the version number and other information. If not, ensure you have opened a new Terminal or command prompt.

Step 1.2: Configure Azure CLI

1. In your Terminal or command prompt window, type the following: **az login**
2. A new window will open where you can log in.. In a previous course, you should have set up an Azure account; log in using the same account as you did when you set up your account. The window will close automatically.
3. (If prompted) Choose the appropriate subscription and tenant. (Note: this will only appear if you have multiple subscriptions associated with your account)
4. Once logged in, enter the following command:
az webapp up --runtime PYTHON:3.9 --sku F1 --logs

This command will upload the contents of the current folder using the Python 3.9 runtime environment, using the F1 (free) tier. The free tier is a great way to demonstrate or test applications that don't have availability or performance requirements.

5. The deployment may take a few minutes to complete. (Troubleshooting tip: as you are using the free tier, you may get a warning about your creation operation being throttled. If this is the case, you can change to a paid tier or wait until the error clears. For free tiers, this can be as long as two days.)

6. If your deployment is successful, you will get a link that is in the format:
http://brave-cliff-b73da2a84c49ceb7fee8f649bf01e1.azurewebsites.net
(Note: this is a non-working example)

Step 2: View application on Azure portal

Step 2.1: Log in to the Azure portal

Open a web browser and navigate to <https://portal.azure.com> and sign in using your Azure credentials (the same account you used to deploy the application).

Step 2.2: Locate the resource group

Once logged in, on the left-hand navigation pane, click on "Resource groups".

In the Resource groups section, you will see a list of all resource groups associated with your Azure subscription. Find and click on the resource group where your application was deployed. The application in step 1 uses a default name, so you may need to investigate if you have multiple groups.

Step 2.3: Locate the resource

Within the selected Resource Group, you'll see a list of resources (like Web Apps, Virtual Machines, Databases, etc.) deployed in that group. Look for the name of your Web App resource.

Step 2.4: View the application

Click on the application resource (for example, if it's a Web App, click on the name of your Web App).

The Azure Portal will open a detailed view of that application, where you can:

- See deployment details.
- Access settings and configurations.
- View logs and performance metrics.
- View dashboards and visualizations for monitoring performance.
- Check the application's status and health.
- Shut down or start an application.
- The URL to access the deployed application.

The portal is generally more user-friendly than the CLI (command-line interface) that you used in the exercise, but it is also less powerful when it comes to testing, making changes, and debugging.

You can use the Azure portal to delete the application. If you were not using a free tier, and you left the application running, it could continue to incur charges.

Hints & Tips

For further information, you can view the Quickstart guide on the Microsoft Azure site [here](#) which will walk you through deploying Flask applications.

1. What is the primary purpose of deploying this application to Azure?

1 point

- To enable users to access the application over the internet.
- To automate the testing of the application.
- To store the application's code securely.
- To enhance the application's performance.

2. Why is the Azure CLI used in this process?

1 point

To visually design the user interface of the application.

To manage and automate interactions with Azure resources.

To write the Python code for the Flask application.

To test the application's functionality in a web browser.

3. You deployed your Flask application to Azure using the `az webapp up` command and specifying the `--sku F1` option. Why was the F1 tier used in this deployment?

1 point

It provides the highest level of performance and scalability.

It allows for testing and development without incurring costs.

It is the only tier that supports Python applications.

It guarantees the application will always be available.

4. Why might you choose to use the Azure CLI to manage your deployed Flask application instead of the Azure portal? Choose the best answer.

1 point

The CLI allows for point-and-click interactions, which can be easier for complex tasks.

The CLI provides a graphical user interface that is more intuitive for some users.

The CLI offers visualizations and dashboards for monitoring application performance.

The CLI allows for greater control and automation for tasks like testing and debugging.

Coursera Honor Code [Learn more](#)