Lab 09a - Implement Web Apps

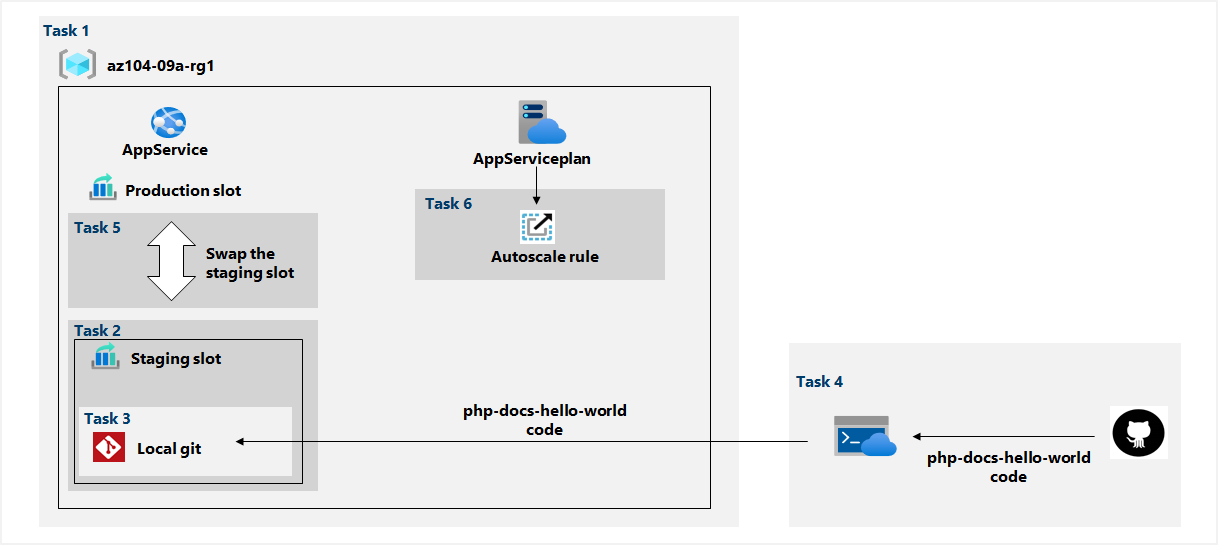
## **Lab scenario**

You need to evaluate the use of Azure Web apps for hosting Contoso’s web sites, hosted currently in the company’s on-premises data centers. The web sites are running on Windows servers using PHP runtime stack. You also need to determine how you can implement DevOps practices by leveraging Azure web apps deployment slots.

## **Objectives**

* Task 1: Create an Azure web app
* Task 2: Create a staging deployment slot
* Task 3: Configure web app deployment settings
* Task 4: Deploy code to the staging deployment slot
* Task 5: Swap the staging slots
* Task 6: Configure and test autoscaling of the Azure web app

## **Architecture diagram**



Exercise 1

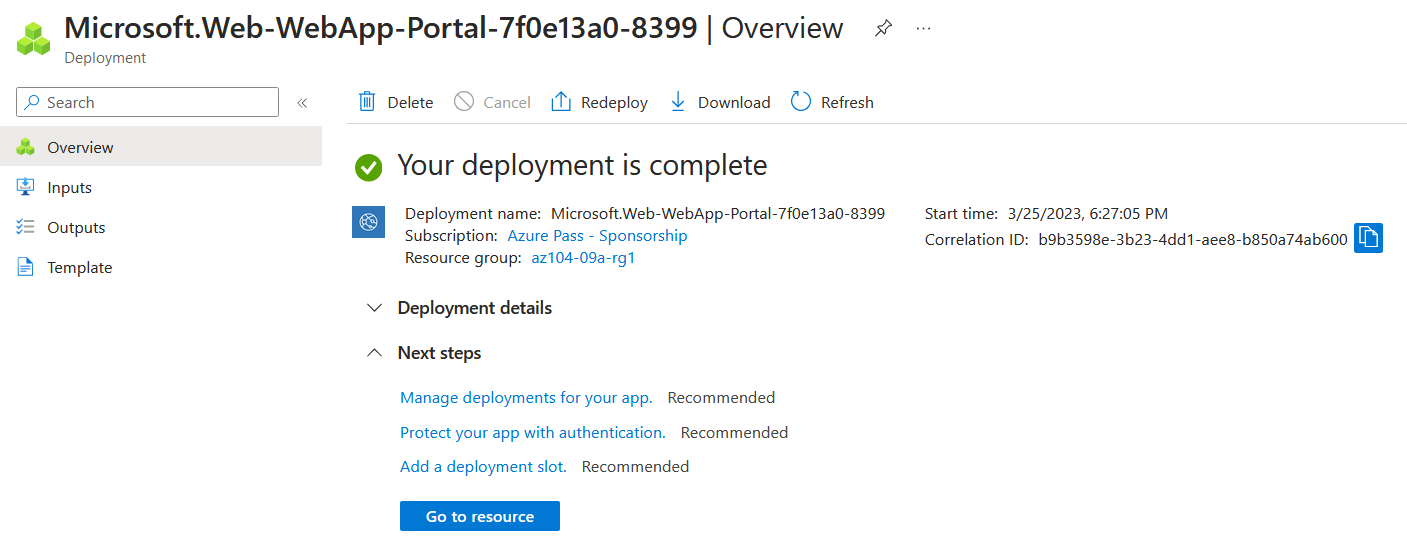
Task 1: Create an Azure web app

In this task, we will create an Azure web app.

From the Azure portal we search and select **App services** and **create** new with the following settings:

| Setting | Value |
| --- | --- |
| Subscription | the name of the Azure subscription you are using in this lab |
| Resource group | the name of a new resource group **az104-09a-rg1** |
| Web app name | any globally unique name |
| Publish | **Code** |
| Runtime stack | **PHP 8.0** |
| Operating system | **Linux** |
| Region | the name of an Azure region where you can provision Azure web apps |
| App service plan | accept the default configuration |

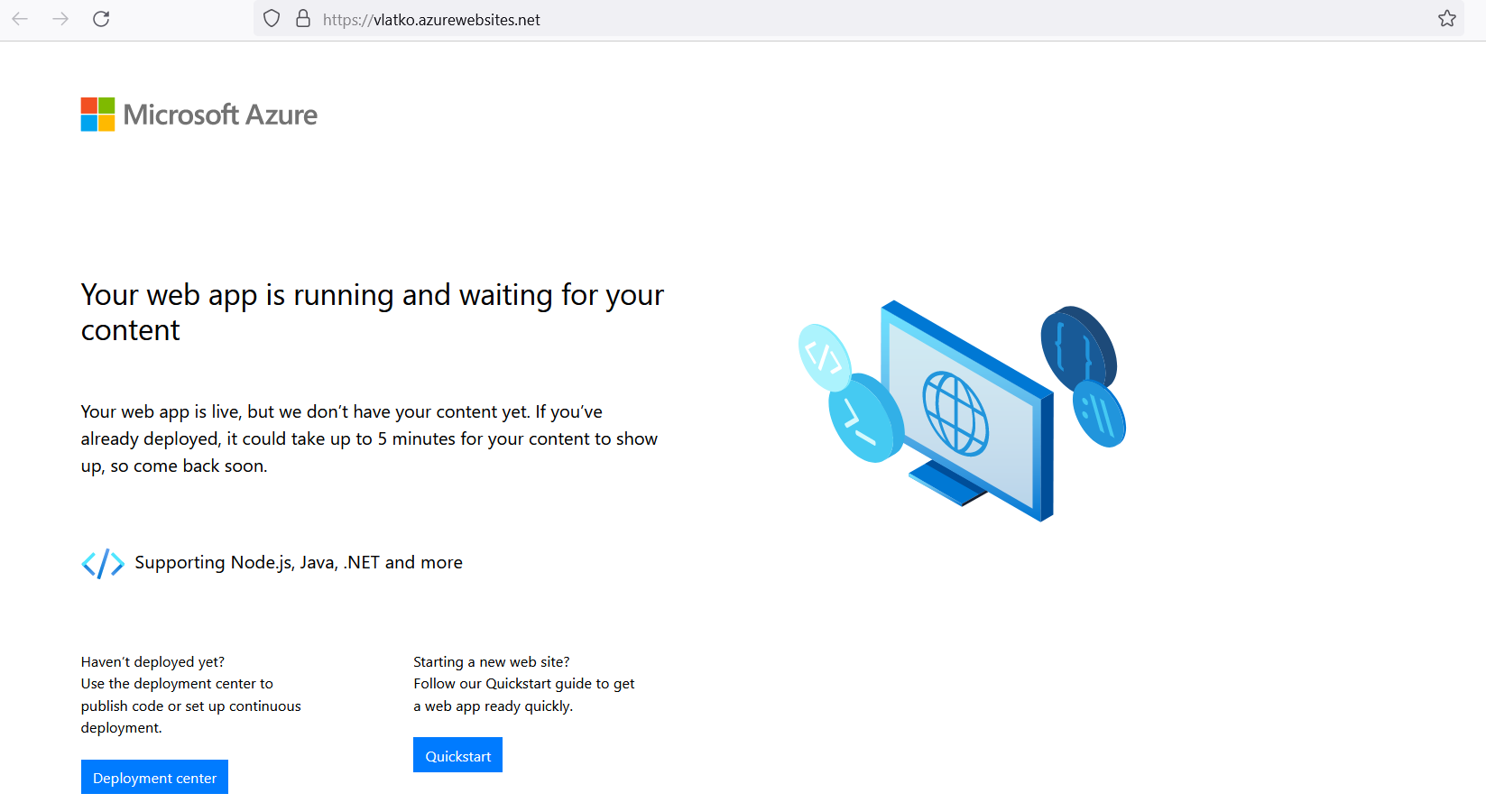
After the deployment is completed, we go to the resource.



Task 2: Create a staging deployment slot

In this task, we will create a staging deployment slot.

On the blade of the newly deployed web app, we click the URL to display the default web page

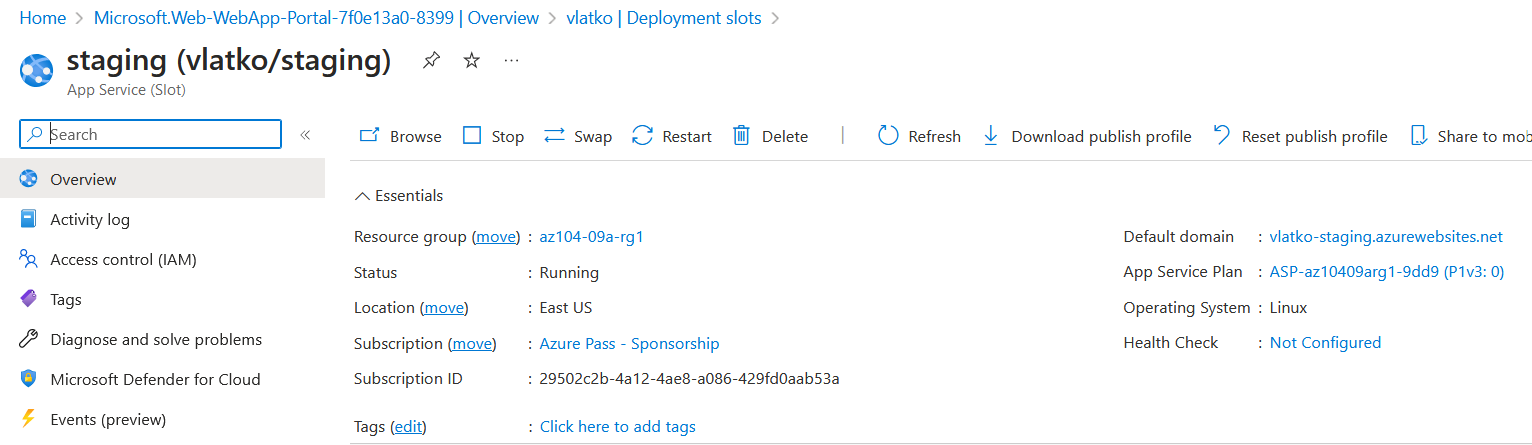


We close the new tab and back in the Azure portal we click on **Deployment slots.**

We click **Add slot**, and add a new slot with the following settings:

| Setting | Value |
| --- | --- |
| Name | **staging** |
| Clone settings from | **Do not clone settings** |

Back on the Deployment slot blade we click on the newly created staging slot and note that the URL differs from the one on the production slot.

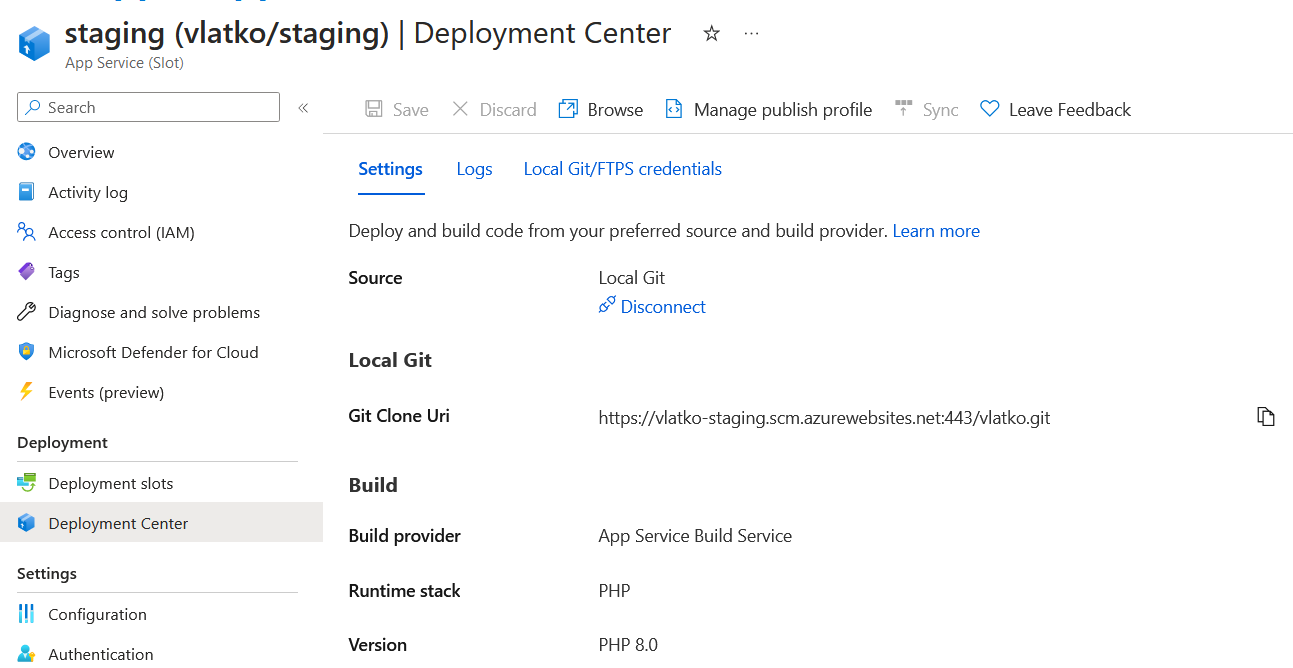


Task 3: Configure web app deployment settings

In this task, we will configure web app deployment settings.

On the deployment section of the staging slot we go to the **Deployment Center** and select the settings tab. Here in the Source drop-down list we select **Local Git** and click save.

We can copy the **Git Clone Url**, we will need it later in the next task.



We then select the **Local Git/FTPS credentials** tab and in the **User Scope** section we enter unique name and password, we will need these credentials in the next task.

Task 4: Deploy code to the staging deployment slot

In this task, we will deploy code to the staging deployment slot.

In the Azure portal we open **Cloud Shell** and enter the following command to clone the remote repository containing the code for the web app.

git clone <https://github.com/Azure-Samples/php-docs-hello-world>

Then we run the following to set the current location to the newly created clone

Set-Location -Path $HOME/php-docs-hello-world/

We run the following to add the remote git

git remote add vlatko283 <https://vlatko-staging.scm.azurewebsites.net:433/vlatko.git>

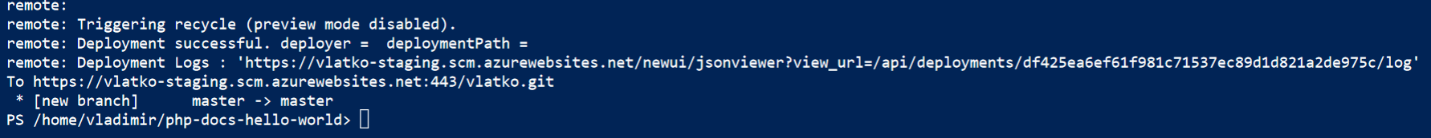
And last, we run the following to push the sample web app code from the repository to the staging deployment slot

git push vlatko283 master

We enter the name and the password that we set previously.

Text

Description automatically generated



We can now open the URL of the staging slot blade to verify that it displays Hello World! Message.

Graphical user interface, text, application

Description automatically generated

Task 5: Swap the staging slots

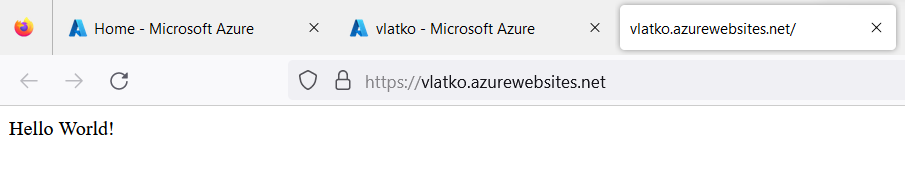
In this task, we will swap the staging slot with the production slot

We navigate back to the production slot and in the **Deployment slots** we click the **Swap** toolbar icon. We review the default settings and click **Swap**.

Graphical user interface, text, application

Description automatically generated

We can now view the URL and verify that the default web page is replaced with the **Hello World!** Page.



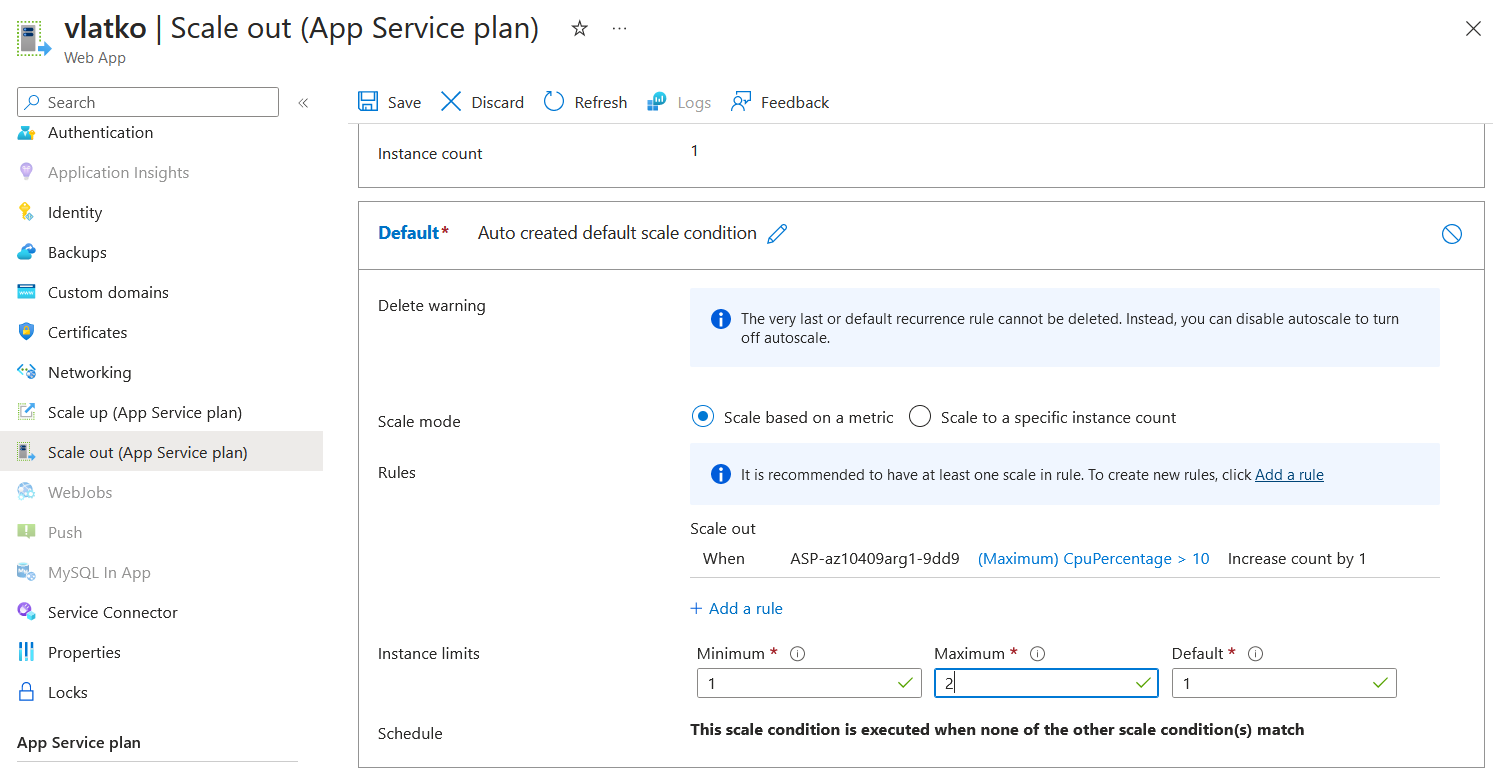
Task 6: Configure and test autoscaling of the Azure web app

In this task, we will configure and test autoscaling of Azure web app.

In the production blade we select **Scale out** and click **Custom autoscale**. We select **Scale based on a metric** and click **+ Add a rule with the following settings:**

| Setting | Value |
| --- | --- |
| Metric source | **Current resource** |
| Metric namespace | **standard metrics** |
| Metric name | **CPU Percentage** |
| Operator | **Greater than** |
| Metric threshold to trigger scale action | **10** |
| Duration (in minutes) | **1** |
| Time grain statistic | **Maximum** |
| Time aggregation | **Maximum** |
| Operation | **Increase count by** |
| Instance count | **1** |
| Cool down (minutes) | **5** |

Then we specify the following settings for the instance limits and click **Save**.



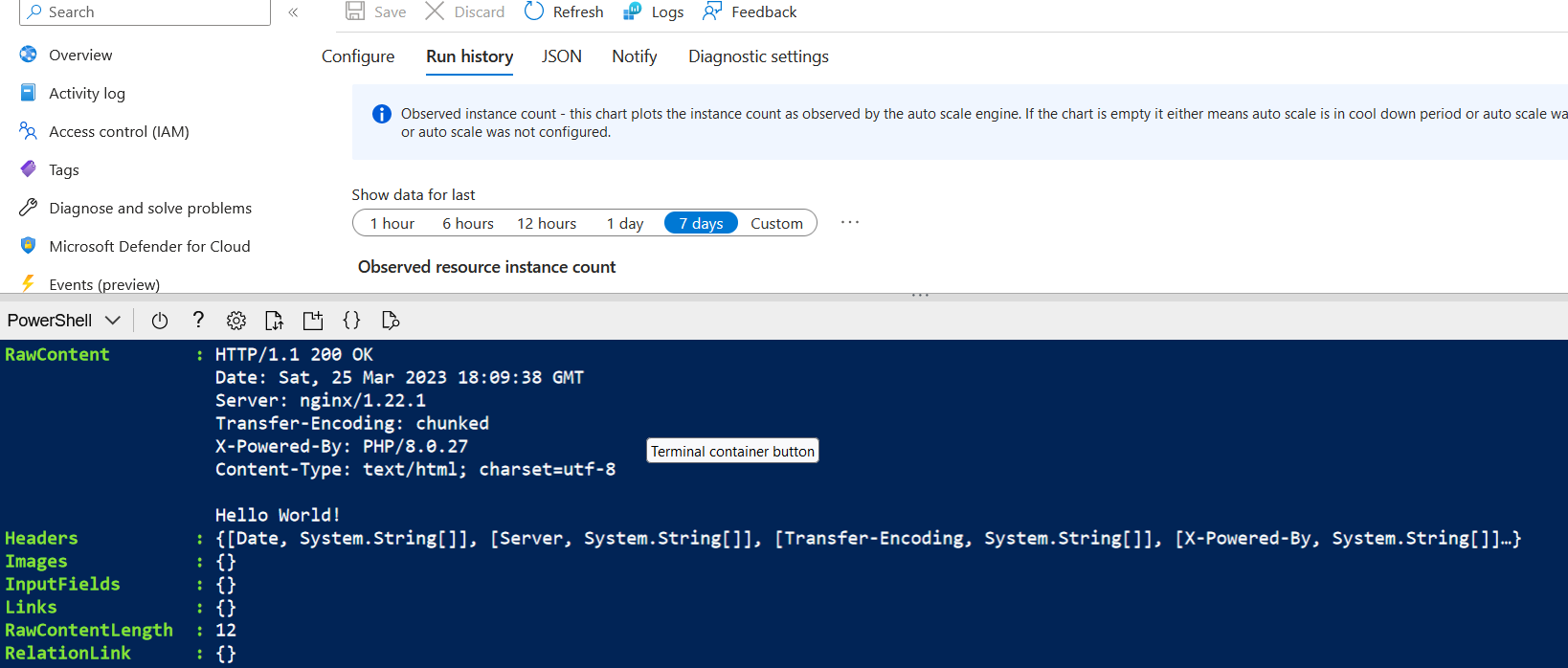
In the Azure portal we open Cloud Shell again and run the following to identify the URL of the web app:

$rgName = 'az104-09a-rg1'

$webapp = Get-AzWebApp -ResourceGroupName $rgName

We now run the following to start an infinite loop that sends the HTTP requests to the web app:

while ($true) { Invoke-WebRequest -Uri $webapp.DefaultHostName }



We minimize the Cloud Shell and in the Scale out section we select the **Run history** tab and check the **Observed resource instance count**. We monitor the utilization and the number of instances for a few minutes, once it has increased to 2 we reopen the Cloud Shell and terminate the loop by pressing **Ctrl+C**.

We can now clean up resources.