Student Name: **Weight: 3%**

Student ID: **Marks: /10**

Lab: Azure App Service

Lab Objectives

In this lab, you will be learning how to create and manage the Azure App service deploy a virtual machine with an ARM template. You will:

1. Create a web app.
2. Create a PHP MySQL app.
3. Create an Azure Function.

Lab Requirements

- Up to date browser
- Azure account
- A free GitHub account

Instructions

1. Working individually, follow the procedure below.
2. Take screenshots, as described in the *Marking Criteria* section.
3. Create a document that includes all screenshots appropriately titled and described, and then upload it to Brightspace, as indicated by your instructor.
4. Be sure to include your name and student ID in the document.

Marking Criteria

Screenshots	Marks
Browser Window with FQDN, running PHP/MySQL with tasks	/5
Start VM Azure Function Test/Run window and insights log	/5
Total	**/10**

Note: This icon indicates when a screenshot is required.

Source: Flatiron.com, Freepik, Image: screenshot_983871

Procedure

Part 1: Create a Web App

Azure virtual machines are part of the IAAS (infrastructure as a service) offerings in Azure. To create a website using an Az VM, you would have to:

1. Create the VM.
2. Install a web server program like Apache on the VM.
3. Create your web pages.
4. Configure your web server program.
5. Set up your networking and IPs.
6. Manage and maintain the systems.

Azure App service is part of Azure PAAS (platform as a service) offerings. It allows you to create web applications without managing the underlying operating system, networking or web software. Azure app service is defined by Microsoft as a “fully managed web application hosting platform.”

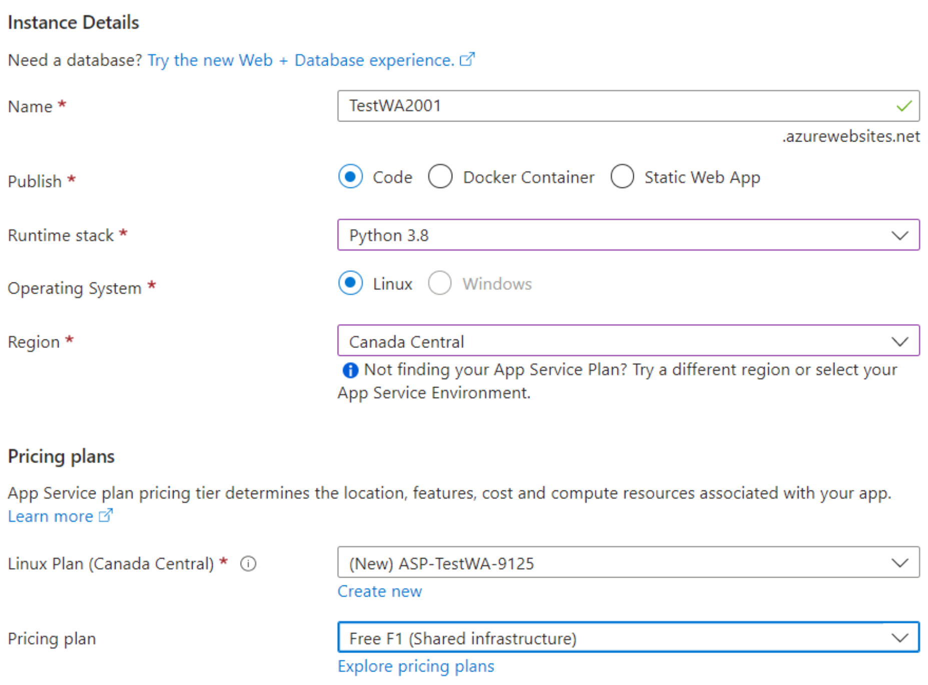
☐ Navigate to the **Azure Web Services** page and click **Create App Service**.

☐ Select or create a resource group and give the app a name.

Note: The web app name must be globally unique and the site will end with **.azurewebsites.net**.

☐ Your app service can be deployed as code, as a Docker image or simply as a static web site. You created Docker images in a previous lab, so select **Code** as the publish method.

☐ Choose **Python 3.8**, your region and the **F1 Free Pricing** plan.



Instance Details

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

Name *	TestWA2001 .azurewebsites.net
Publish *	[✓ Code] Docker Container Static Web App
Runtime stack *	Python 3.8
Operating System *	[✓ Linux] Windows
Region *	Canada Central

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

Pricing plans

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

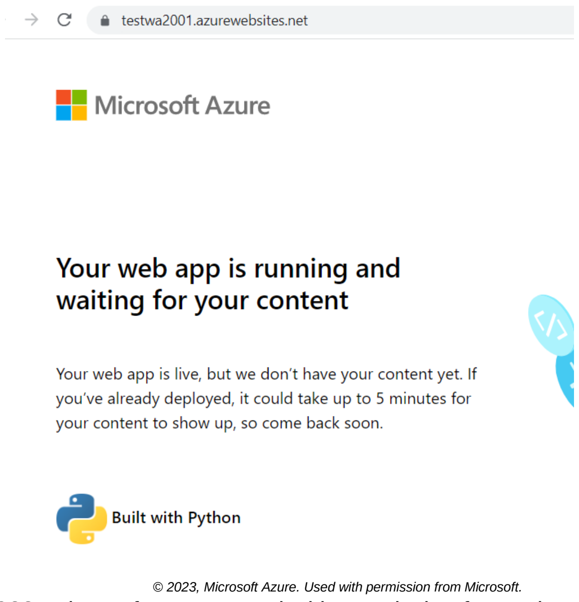
Linux Plan (Canada Central) *	(New) ASP-TestWA-9125
Pricing plan	Free F1 (Shared infrastructure)

© 2023, Microsoft Azure. Used with permission from Microsoft.

☐ Review and create the app.

☐ When the deployment has finished, go to the overview page for the app and click the default domain name.

Your web app opens in a new browser tab with a basic app page.



testwa2001.azurewebsites.net

Microsoft Azure

Your web app is running and waiting for your content

Your web app is live, but we don't have your content yet. If you've already deployed, it could take up to 5 minutes for your content to show up, so come back soon.

Built with Python

© 2023, Microsoft Azure. Used with permission from Microsoft.

© 2023, Microsoft Azure. Used with permission from Microsoft.

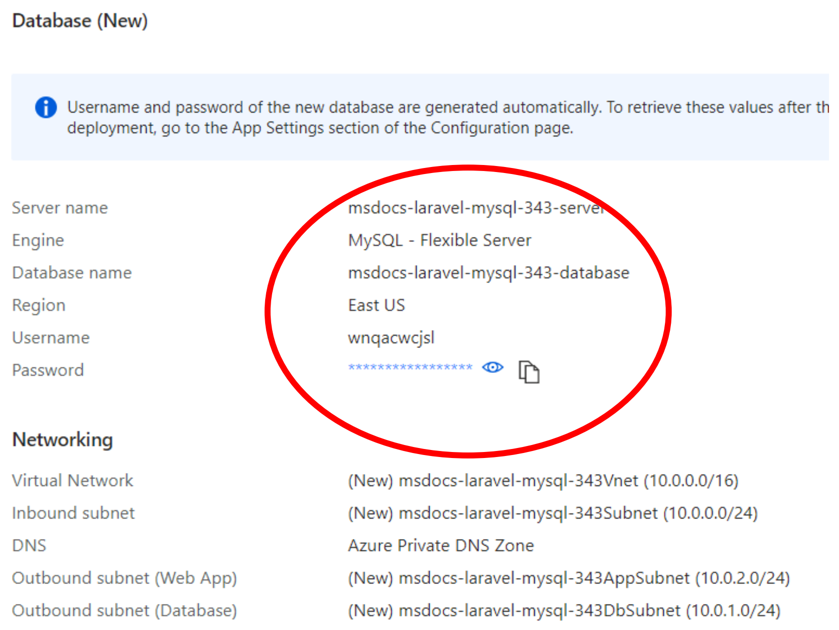
- ☐ In the Azure portal, scroll down the overview page for the app to see all of the information for the app, including its IP address.
- ☐ Delete the resource group if you want to stop accruing charges.

Part 2: Create a PHP MySQL App

In the last section, you created an app but there is no code in the app and it doesn't really do anything. In this section, you'll create a PHP app that is connected to a MySQL database.

Important notes for this section:

- When working through the tutorial, take your time, read the instructions carefully and look at the screenshots.
- Start at *1 - Create App Service and MySQL resources*.
- You only need to pull the application locally if you want to run it locally, so the *Sample Application* section is not necessary.
- In Step 6 of *1 - Create App Service and MySQL resources,* click the **Review** button and copy the database name, username and password **BEFORE** you click the **Create** button because you may not be able to see this information in the connection settings later in the tutorial.



Database (New)

Username and password of the new database are generated automatically. To retrieve these values after th deployment, go to the App Settings section of the Configuration page.

Server name	msdocs-laravel-mysql-343-server
Engine	MySQL - Flexible Server
Database name	msdocs-laravel-mysql-343-database
Region	East US
Username	wnqacwcjsl
Password	****************

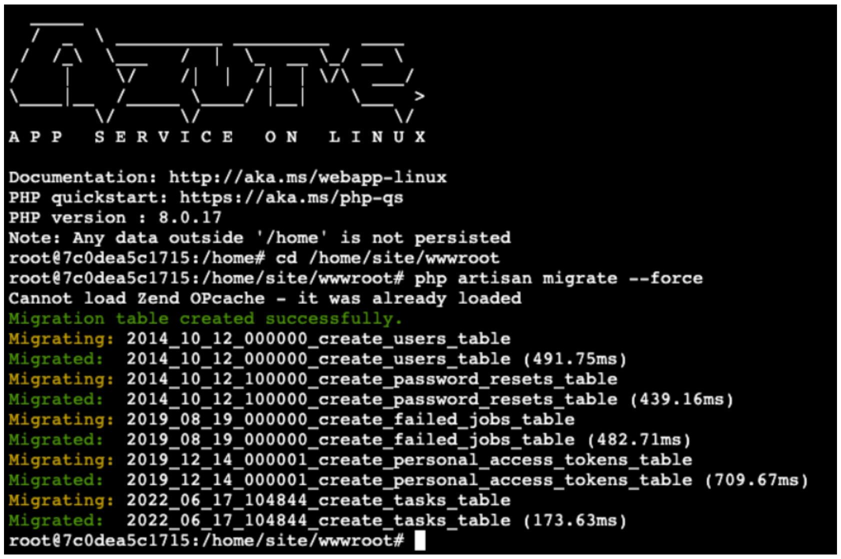
Networking

Virtual Network	(New) msdocs-laravel-mysql-343Vnet (10.0.0.0/16)
Inbound subnet	(New) msdocs-laravel-mysql-343Subnet (10.0.0.0/24)
DNS	Azure Private DNS Zone
Outbound subnet (Web App)	(New) msdocs-laravel-mysql-343AppSubnet (10.0.2.0/24)
Outbound subnet (Database)	(New) msdocs-laravel-mysql-343DbSubnet (10.0.1.0/24)

© 2023, Microsoft Azure. Used with permission from Microsoft.

- Things may take several minutes to work, so make sure a task is complete before you start the next one.

- When generating the database schema, you should see a similar window to the one shown below. If you don't, wait a minute or two and try again. However, be aware that these commands time out, so you may have to redo them during the deployment.



```
  _____
  /  _  \ __________ _________   ____
 /  /_\  \\___   /  |  \_  __ \_/ __ \
/    |    \/    /|  |  /|  | \/\  ___/
\____|__  /_____ \____/ |__|    \___  >
        \/      \/                  \/
A P P   S E R V I C E   O N   L I N U X

Documentation: http://aka.ms/webapp-linux
PHP quickstart: https://aka.ms/php-qs
PHP version : 8.0.17
Note: Any data outside '/home' is not persisted
root@7c0dea5c1715:/home# cd /home/site/wwwroot
root@7c0dea5c1715:/home/site/wwwroot# php artisan migrate --force
Cannot load Zend OPcache - it was already loaded
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (491.75ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (439.16ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (482.71ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated:  2019_12_14_000001_create_personal_access_tokens_table (709.67ms)
Migrating: 2022_06_17_104844_create_tasks_table
Migrated:  2022_06_17_104844_create_tasks_table (173.63ms)
root@7c0dea5c1715:/home/site/wwwroot# 
```

© 2023, Microsoft Azure. Used with permission from Microsoft.

☐ Complete the Microsoft Tutorial: Build a PHP and MySQL app in Azure App Service (https://learn.microsoft.com/en-us/azure/app-service/tutorial-php-mysql-app).

Section 3: Create an Azure Function App

Azure Functions allow you to trigger and run a small piece of code (function) that only accrues costs when it runs. Azure Functions are event driven and serverless (you do not have to manage a server operating system). Examples include:

- Sending someone an automated email if they upload a file
- Running a backup on a schedule
- Performing a calculation when certain database entries are made

Azure functions can be created in many languages but the type of accounts you have may affect which languages can be used. In this section, you'll create an Azure Function that allows you to start a pair of virtual machines from an HTTP trigger event.

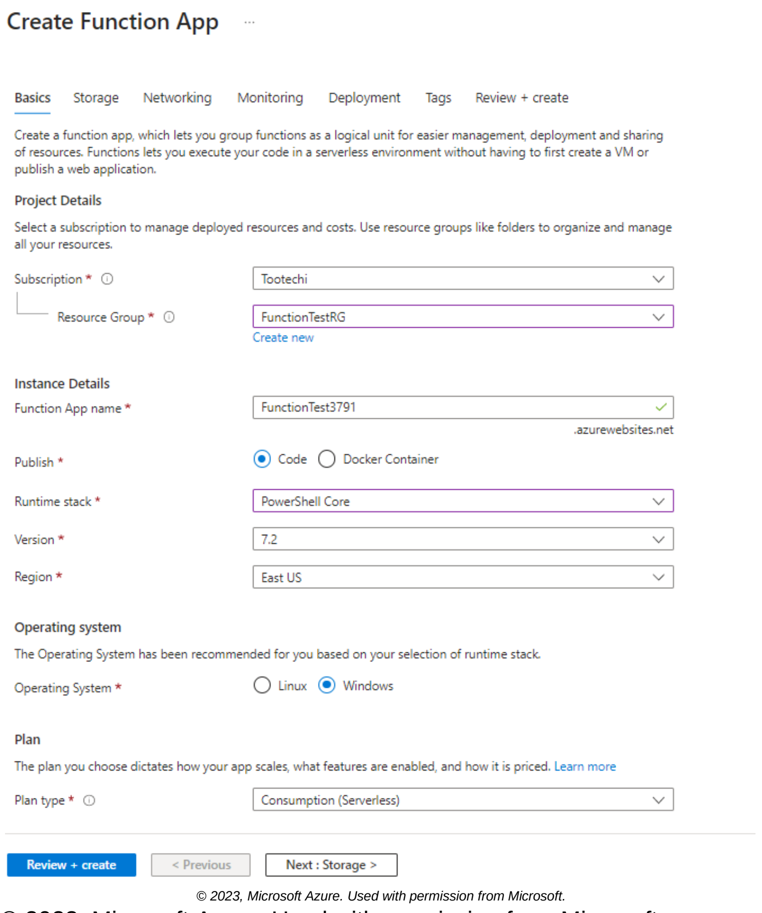
Note: Throughout this section, be prepared to be patient and refresh the window frequently.

☐ Create two inexpensive virtual machines in a new resource group, and then stop (deallocate) them once they are deployed.

Note: Remember that, even stopped, virtual machines accrue charges. It is best to create them with no Public Inbound Ports.

- ☐ Create the Function App, which is the container or environment for your function, code and bindings, by navigating to the **Azure Functions App** page and clicking the **Create Function App** button.
- ☐ Select the resource group you created for this section and give your Function App a unique name.
- ☐ Select **PowerShell Core** for the code and the latest version.
- ☐ Select your region and the plan type as **Consumption (Serverless)**. This plan has the fewest features and therefore costs the least.

  Note: For more information, see: [Azure Functions hosting options](https://learn.microsoft.com/en-us/azure/azure-functions/functions-scale) (https://learn.microsoft.com/en-us/azure/azure-functions/functions-scale).



Create a function app, which lets you group functions as a logical unit for easier management, deployment and sharing of resources. Functions lets you execute your code in a serverless environment without having to first create a VM or publish a web application.

Project Details

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

Subscription *: Tootechi
Resource Group *: FunctionTestRG
Create new

Instance Details

Function App name *: FunctionTest3791 .azurewebsites.net
Publish *: [✓ Code] Docker Container
Runtime stack *: PowerShell Core
Version *: 7.2
Region *: East US

Operating system

The Operating System has been recommended for you based on your selection of runtime stack.

Operating System *: Linux [✓ Windows]

Plan

The plan you choose dictates how your app scales, what features are enabled, and how it is priced. Learn more

Plan type *: Consumption (Serverless)

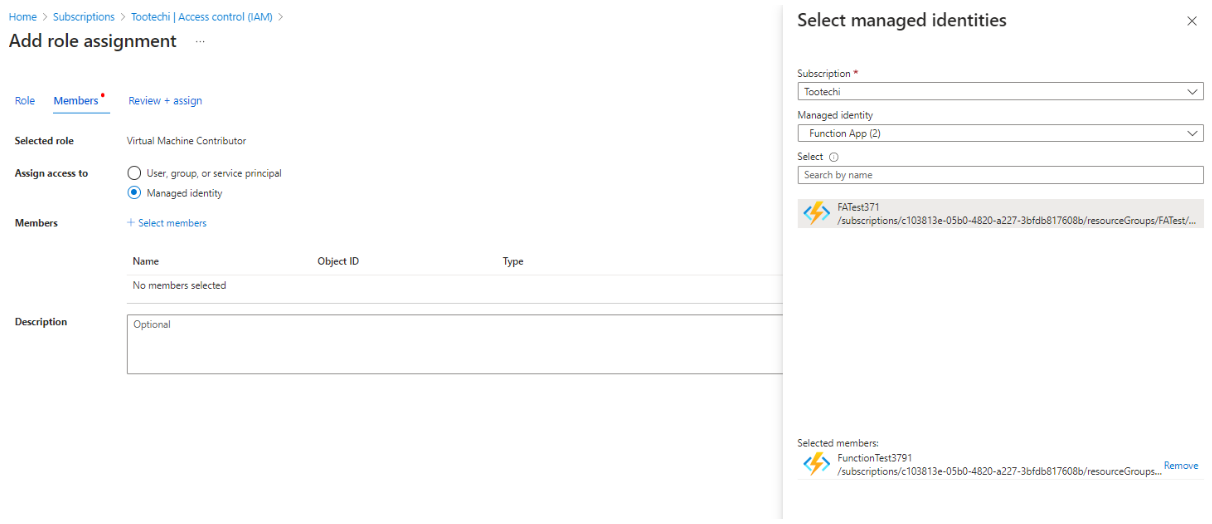
© 2023, Microsoft Azure. Used with permission from Microsoft.

© 2023, Microsoft Azure. Used with permission from Microsoft.

- ☐ Click **Next** to advance to the Storage page. Since functions are serverless, they must be stored somewhere, so Azure creates a new storage account by default. Leave the default name or rename it to something more recognizable.
- ☐ Go to the **Monitoring** page and make sure *Application Insights* is set to **Yes**.
- ☐ Review and create the function.

As discussed previously, Azure uses RBAC to allow identities to access resources and perform tasks. While we tend to think of identities as users, a service can also be an identity. Your Function app is going to start a virtual machine, so it needs to have a role that allows it to start virtual machines.

- ☐ From the Overview page for the Function app, select **Identity** from the blade menu.
- ☐ Set the status to **On** and save.
- ☐ Navigate to your Subscriptions page, select **IAM (Access Control)** from the blade menu and then select **Add > New Role Assignment**.
- ☐ Type **Virtual Machine Contributor** into the search box and select it. Note the permissions given to the role.
- ☐ Click **Next** and select **Managed Identity** instead of users and groups.
- ☐ Click **Select Members** and then select your Function App.


Home > Subscriptions > Tootechi | Access control (IAM) >

Add role assignment

Role | Members | Review + assign

Selected role: Virtual Machine Contributor

Assign access to: ( ) User, group, or service principal [✓ Managed identity]

Members: + Select members

Name	Object ID	Type
No members selected		

Description: Optional

Select managed identities

Subscription *: Tootechi

Managed identity: Function App (2)

Select: Search by name

FATest371
/subscriptions/c103813e-05b0-4820-a227-3bfdb817608b/resourceGroups/FATest/...

Selected members:

FunctionTest3791
/subscriptions/c103813e-05b0-4820-a227-3bfdb817608b/resourceGroups... Remove

© 2023, Microsoft Azure. Used with permission from Microsoft.

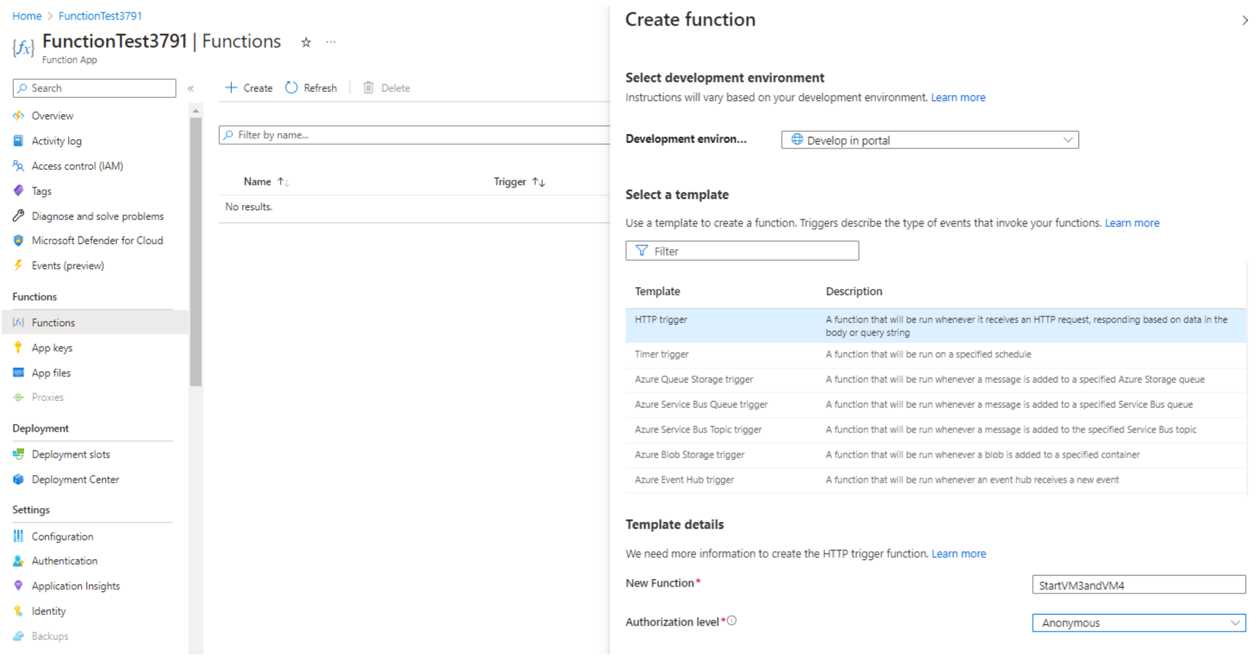
© 2023, Microsoft Azure. Used with permission from Microsoft.

- ☐ Review and assign the role.
- ☐ Return to the main page for your function app and select **Identity**.
- ☐ Under **Permissions** select **Azure Role Assignments** and verify that it has the *Virtual Machine Contributor* role.

  Note: You can assign roles from this screen as well.

You have now created a container/environment with the appropriate permissions. Next, you need to create the function you want to run.

- ☐ Go back to the main page for your function app, select **Functions** from the blade menu and click **Create**.
- ☐ Select the **HTTP trigger** function because you want the action to be set in motion by HTTP.
- ☐ Give the function an appropriate name and set the **Authorization level** to **Anonymous**. This is the authorization to trigger the function.



No results.

Create function

Select development environment

Instructions will vary based on your development environment. Learn more

Development environ... Develop in portal

Select a template

Use a template to create a function. Triggers describe the type of events that invoke your functions. Learn more

Template	Description
HTTP trigger	A function that will be run whenever it receives an HTTP request, responding based on data in the body or query string
Timer trigger	A function that will be run on a specified schedule
Azure Queue Storage trigger	A function that will be run whenever a message is added to a specified Azure Storage queue
Azure Service Bus Queue trigger	A function that will be run whenever a message is added to a specified Service Bus queue
Azure Service Bus Topic trigger	A function that will be run whenever a message is added to the specified Service Bus topic
Azure Blob Storage trigger	A function that will be run whenever a blob is added to a specified container
Azure Event Hub trigger	A function that will be run whenever an event hub receives a new event

Template details

We need more information to create the HTTP trigger function. Learn more

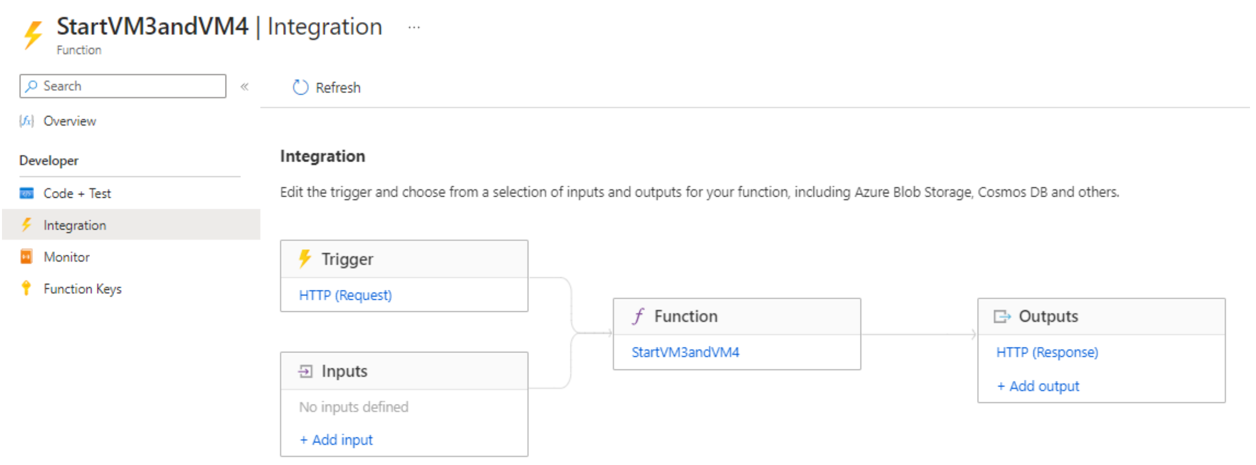
New Function* StartVM3andVM4

Authorization level* Anonymous

© 2023, Microsoft Azure. Used with permission from Microsoft.

© 2023, Microsoft Azure. Used with permission from Microsoft.

- ☐ Create the function.
- ☐ From the **Functions** page, select **Integration** from the blade menu. Notice that the Trigger is an HTTP (request), with the function you created in the middle, and then the output is an HTTP (response).

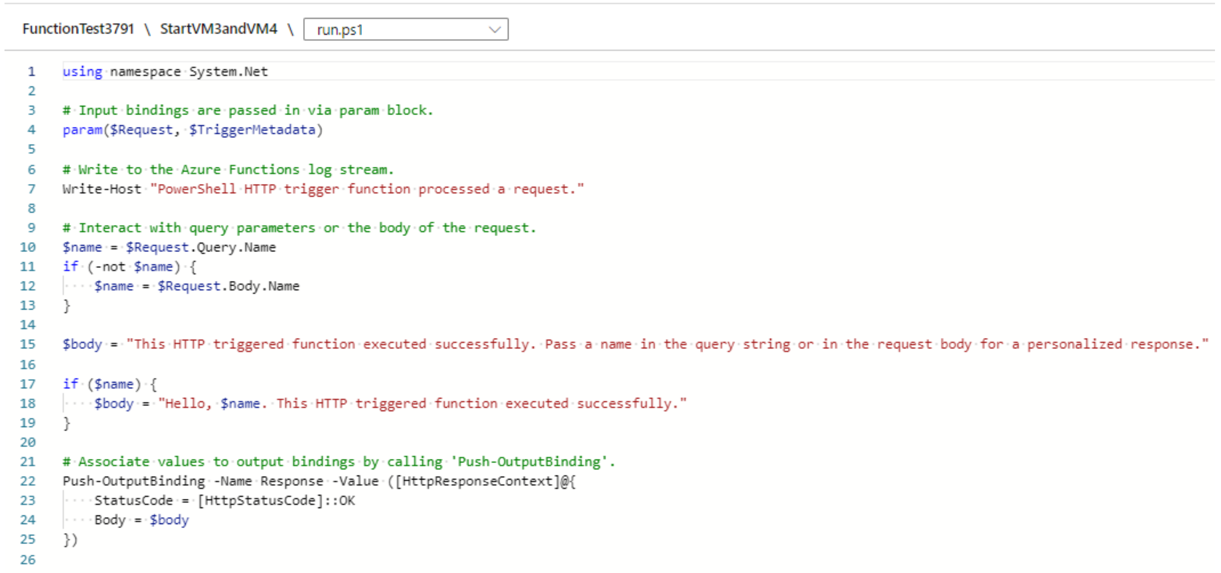


© 2023, Microsoft Azure. Used with permission from Microsoft.

© 2023, Microsoft Azure. Used with permission from Microsoft.

☐ Select **Code + Test** from the blade menu and ensure that **Run.PS1** is selected at the top of the page, and that you can see the default code for this function.

- The Write-Host line tells the system what information to write to the log file.
- The $Name variable allows you to pass a name to the HTTP trigger.
- The $body is pushed to the HTTP response.



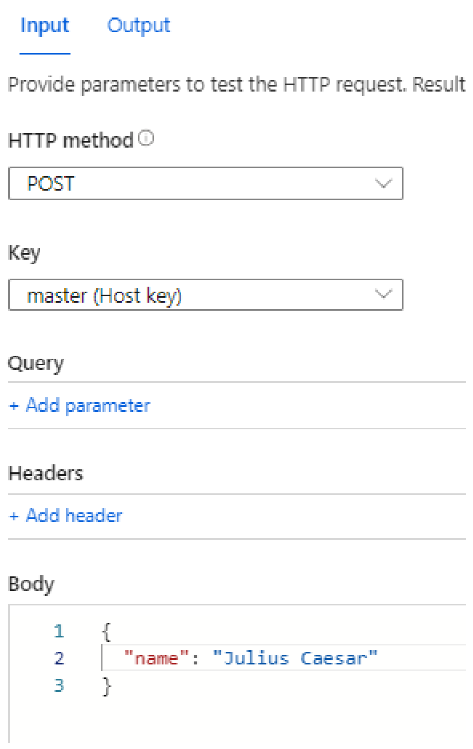
```
using namespace System.Net

# Input bindings are passed in via param block.
param($Request, $TriggerMetadata)

# Write to the Azure Functions log stream.
Write-Host "PowerShell HTTP trigger function processed a request."

# Interact with query parameters or the body of the request.
$name = $Request.Query.Name
if (-not $name) {
    $name = $Request.Body.Name
}

$body = "This HTTP triggered function executed successfully. Pass a name in the query string or in the request body for a personalized response."

if ($name) {
    $body = "Hello, $name. This HTTP triggered function executed successfully."
}

# Associate values to output bindings by calling 'Push-OutputBinding'.
Push-OutputBinding -Name Response -Value ([HttpResponseContext]@{
    StatusCode = [HttpStatusCode]::OK
    Body = $body
})

```

© 2023, Microsoft Azure. Used with permission from Microsoft.

© 2023, Microsoft Azure. Used with permission from Microsoft.

☐ Click **Test/Run** from the top menu. The Input/Output screen opens on the right.

☐ In the **Input** screen, change the name in the Body to your name instead of Azure.



Provide parameters to test the HTTP request. Result

HTTP method: POST

Key: master (Host key)

Query

Headers

Body

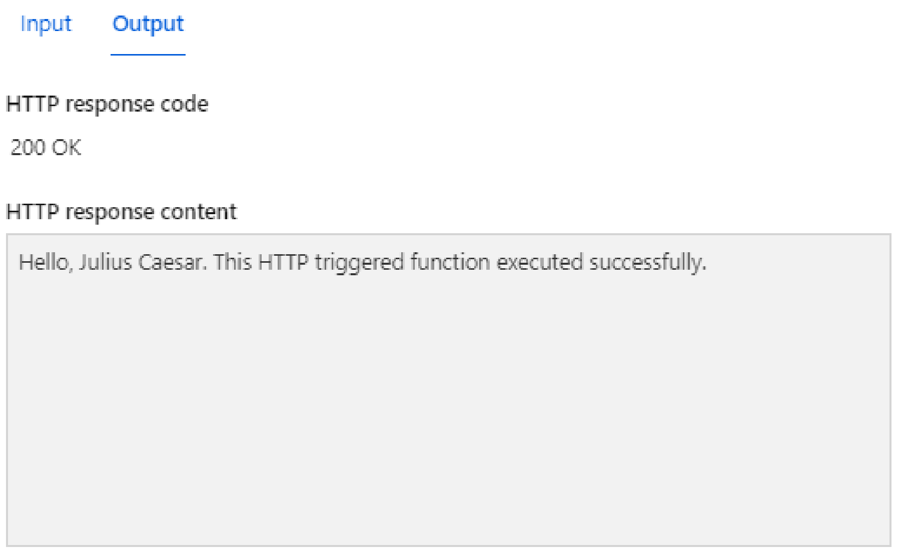
```
{
  "name": "Julius Caesar"
}
```

© 2023, Microsoft Azure. Used with permission from Microsoft.

© 2023, Microsoft Azure. Used with permission from Microsoft.

☐ Click the **Run** button.

The Output screen shows that the function ran successfully and that the output is text from the $body variable.



HTTP response code

200 OK

HTTP response content

```
Hello, Julius Caesar. This HTTP triggered function executed successfully.
```

© 2023, Microsoft Azure. Used with permission from Microsoft.

© 2023, Microsoft Azure. Used with permission from Microsoft.

- ☐ Note the *Application Insights* on the bottom of the screen. If you need to troubleshoot, this information will be very helpful.

You ran the function and passed it your name, but this was in the test environment. The next step is to test that it works from HTTP.

- ☐ Close the *Input/Output* page, select **Get Function URL** from the top page and copy the URL.
- ☐ Paste the URL into a browser. A page should appear that says:

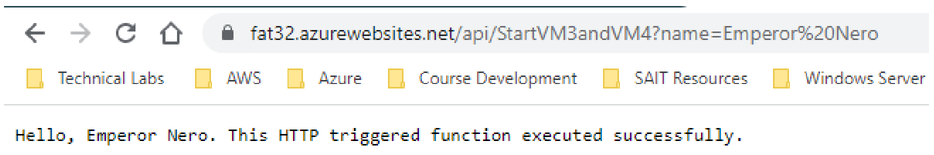
```
This HTTP triggered function executed successfully. Pass a name in the
query string or in the request body for a personalized response
```

  It doesn't have a name because you did not pass the function that information.

- ☐ Add the following to the end of the URL to pass your name to the function:

```
?name=<your_name>
```

Now you should be greeted like you were in the test run.



```
fat32.azurewebsites.net/api/StartVM3andVM4?name=Emperor%20Nero
```

```
Hello, Emperor Nero. This HTTP triggered function executed successfully.
```

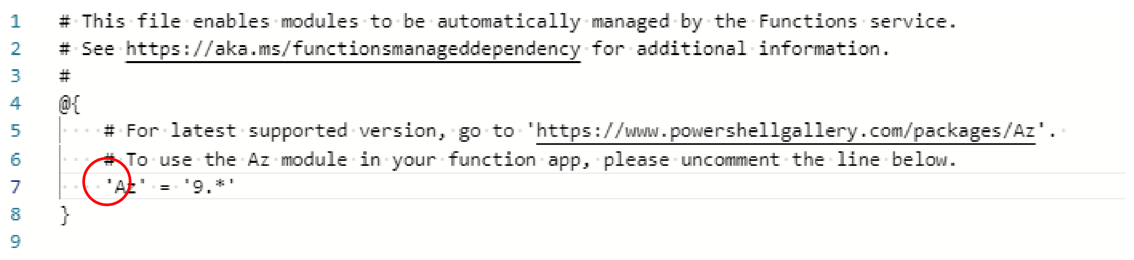
You have a working function, trigger and bindings but now you need to add the code to start your virtual machines. Just like in a Windows server, before you can use a PowerShell command you must have the appropriate PowerShell module loaded.

- ☐ Go to the main page for your **Function App** (not the function) and select **App Files** from the blade menu.

  At the top of the page are three options:

  - Host.json – configuration information that affects all the functions in a function app
  - Profile.ps1 – a profile for the individual function app
  - Requirements.psd1 – add additional requirements for the code in the app

- ☐ Select the **Requirements** option.
- ☐ The command to use the Azure PowerShell module is already in the requirements but it is commented out. Delete the comment (#) character beside the **'Az' =** line.



```
# This file enables modules to be automatically managed by the Functions service.
# See https://aka.ms/functionsmanageddependency for additional information.
#
@{
    # For latest supported version, go to 'https://www.powershellgallery.com/packages/Az'.
    # To use the Az module in your function app, please uncomment the line below.
    'Az' = '9.*'
}
```

☐ Save the change and return to the functions **Code + Test** page. **Run.PS1** should be selected on the top menu.

☐ Click above the line: # *Write to the Azure Functions log stream* and create a few blank lines.

☐ Insert the code shown below but change VM1 and VM2 to the names of your virtual machines.

Notes:

- You're writing a line to the log file (Write-Host) and then using the PowerShell **Get-AzVM** and **Start-AzVM** commands.
- When copying and pasting, sometimes single quote characters are changed to grave accent characters. Recreate the single quote characters, just to be sure.

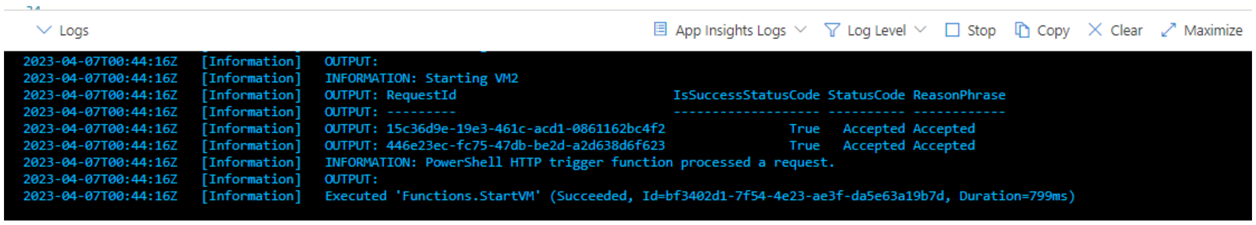
```
$VMs = @('VM1‘,‘VM2‘)
foreach($VM in $VMs)
{
    Write-Host “Starting $VM”
    Get-AzVM –Name $VM | Start-AzVM –NoWait
}
```

☐ Change the body text lines to something appropriate and save your changes.

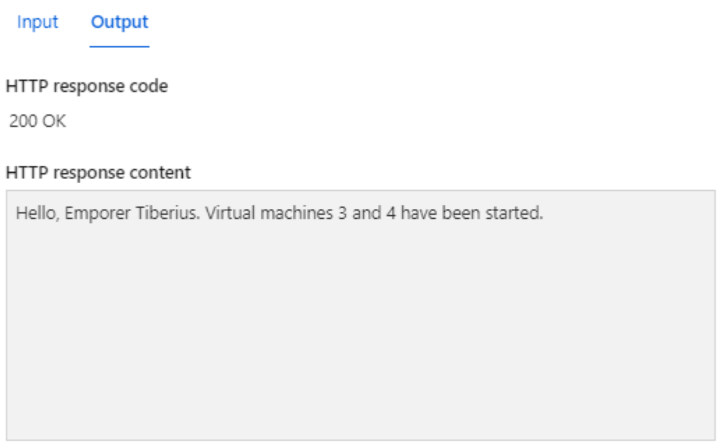
☐ Ensure that your virtual machines are stopped and then click the **Test/Run** button and see if it works.

This will take a few minutes because the PowerShell dependency has to be downloaded and the two virtual machines started.

☐ If you get errors, read the log information in the window at the bottom of the screen and check your syntax. If the download and set up take too long or has problems, you may get a HTTP response failure (this is common). Wait a couple of minutes and run it again. You should see something similar to the log below.



```
2023-04-07T00:44:16Z   [Information]   OUTPUT:
2023-04-07T00:44:16Z   [Information]   INFORMATION: Starting VM2
2023-04-07T00:44:16Z   [Information]   OUTPUT: RequestId                            IsSuccessStatusCode StatusCode ReasonPhrase
2023-04-07T00:44:16Z   [Information]   OUTPUT: ---------                            ------------------- ---------- ------------
2023-04-07T00:44:16Z   [Information]   OUTPUT: 15c36d9e-19e3-461c-acd1-0861162bc4f2                True   Accepted Accepted
2023-04-07T00:44:16Z   [Information]   OUTPUT: 446e23ec-fc75-47db-be2d-a2d638d6f623                True   Accepted Accepted
2023-04-07T00:44:16Z   [Information]   INFORMATION: PowerShell HTTP trigger function processed a request.
2023-04-07T00:44:16Z   [Information]   OUTPUT:
2023-04-07T00:44:16Z   [Information]   Executed 'Functions.StartVM' (Succeeded, Id=bf3402d1-7f54-4e23-ae3f-da5e63a19b7d, Duration=799ms)
```



HTTP response code

200 OK

HTTP response content

Hello, Emporer Tiberius. Virtual machines 3 and 4 have been started.

© 2023, Microsoft Azure. Used with permission from Microsoft.

© 2023, Microsoft Azure. Used with permission from Microsoft.

☐ Check to see whether your virtual machines have started.

Note: This may take a few minutes and require a few screen refreshes. Be patient.

Now, you need to test that your function can be triggered by HTTP.

☐ Stop (deallocate) the two virtual machines.

☐ Copy the function URL if you need to and paste it into a new browser window.

☐ Check that your virtual machines have started.

☐ Delete resources that you’re not using

Resource

Azure Functions PowerShell developer guide (https://learn.microsoft.com/en-us/azure/azure-functions/functions-reference-powershell?tabs=portal)