# Azure VM Monitoring

This azure function will monitor some pre-defined and user defined tags for assets deployed in one or more Resource Groups (RGs) in your azure environment

These tags are then pushed to a firewall of your choice.

This feature is similar to the VM-Monitoring feature built into PAN-OS.

<https://www.paloaltonetworks.com/documentation/70/pan-os/pan-os/policy/enable-vm-monitoring-to-track-changes-on-the-virtual-network>

# Overview

The solution is distributed as a python script that runs as an Azure Function.

<https://azure.microsoft.com/en-us/services/functions/>

The script has been purposely written to have no external dependencies since python support in Azure Functions is still experimental!

# Use portal to create an Azure Active Directory application and service principal that can access resources

In order to make API calls into the Azure environment one needs to first and foremost create a service principal and then register your application with the Azure AD environment. This will provide you with various keys/IDs that will be used to generate an Azure Bearer Token that will be used in the header during the REST calls.

Please follow the instructions here to create an Azure Active Directory application and service principal:

<https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-create-service-principal-portal>

Ideally you can start from the “Create an Azure Active Directory application” portion (if your account has the ability to register applications). If not, you may need to talk to the account admin to allow the ability to register applications.

[Patrick: I may need some input from you on this piece since you did this in the CE subscription.

Any caveats to know about? I am doing this in my subscription where I am admin.]

Once you have registered your application (you can name it anything you like) you will need to note down a few things (these are also listed in the link above):

Application ID:

Secret key:

Tenant ID:

Lastly, Subscription ID: [<https://blogs.msdn.microsoft.com/mschray/2016/03/18/getting-your-azure-subscription-guid-new-portal/>]

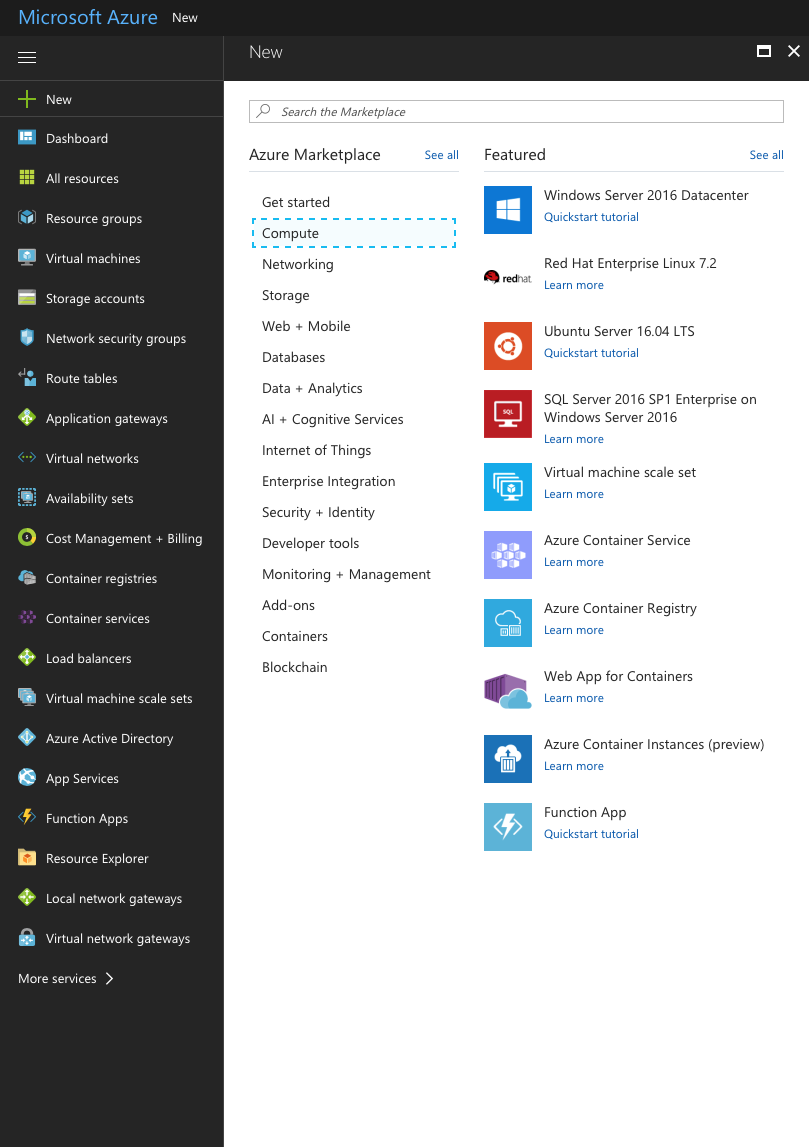
In the link above when you get to the “Assign application to role” part it is possible that you are the admin of your account and can assign an IAM role to your app. If so, add your application and assign it a reader role.

If you do not have access to do this, you may need to contact your admin to grant you access

[Patrick: I may need some input from you on this piece]

# Create an Azure Function using the Azure Portal

Log into the Azure portal (<https://portal.azure.com>) and create your Function App.



On the next screen enter your App Name, Select your subscription that you want to use for this App, Create a new Resource Group (or use an existing one).

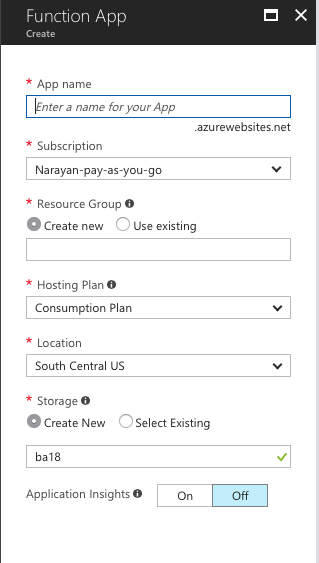
The next step is determined by the location of the firewall that receives the ip-to-tag mapping.

### Publically addressable firewalls

For firewalls that are publicly accessible you can choose the **Consumption plan**.

This plan allows you to pay-per-execution (and dynamically allocates resources based on your app’s load)

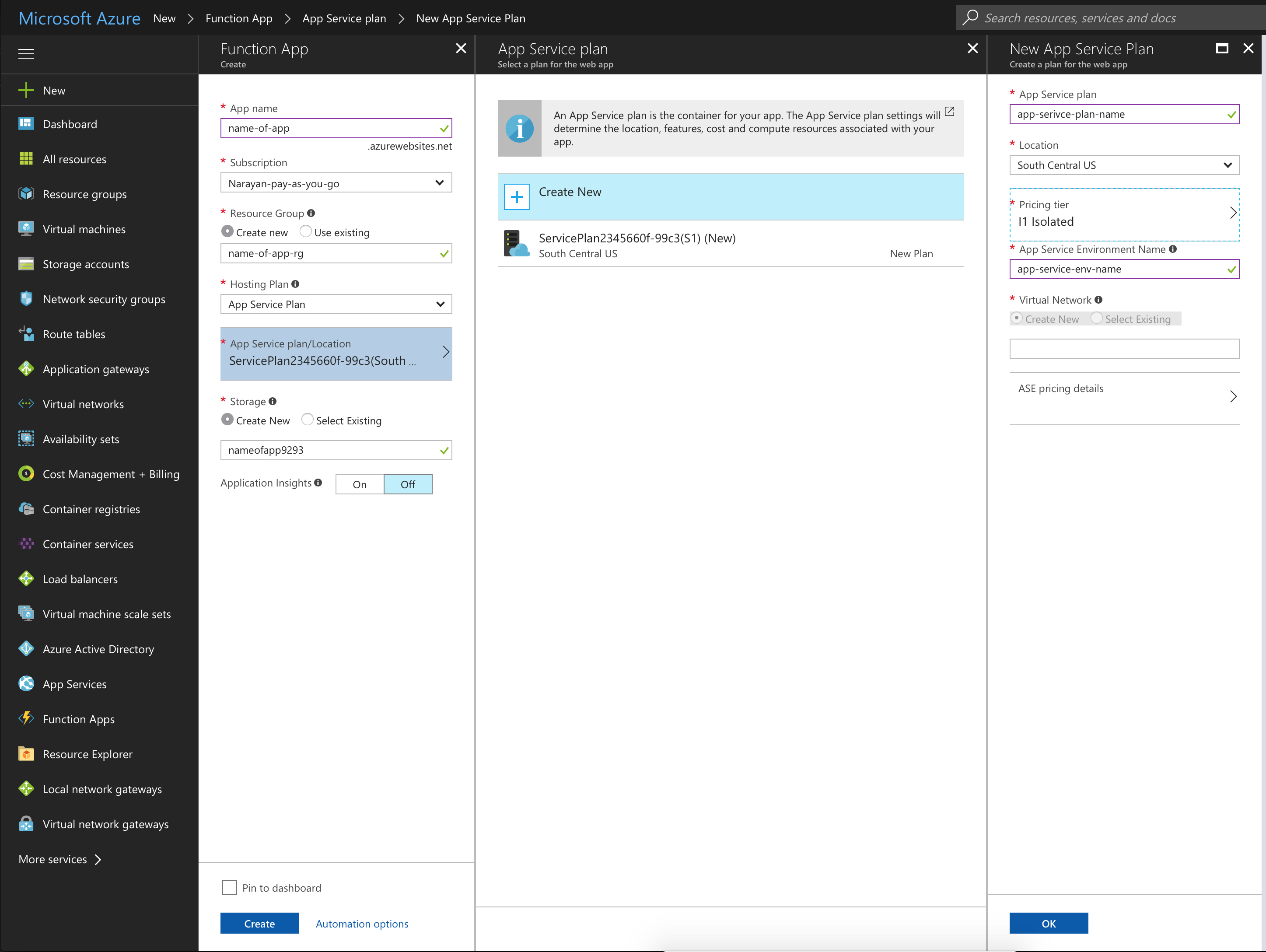
More information here: <https://azure.microsoft.com/en-us/pricing/details/functions/>



Enter a name for your function app. You can choose to launch it in an existing Resource Group or create a new one. Same goes for your storage account. Pick a location, and hit Create

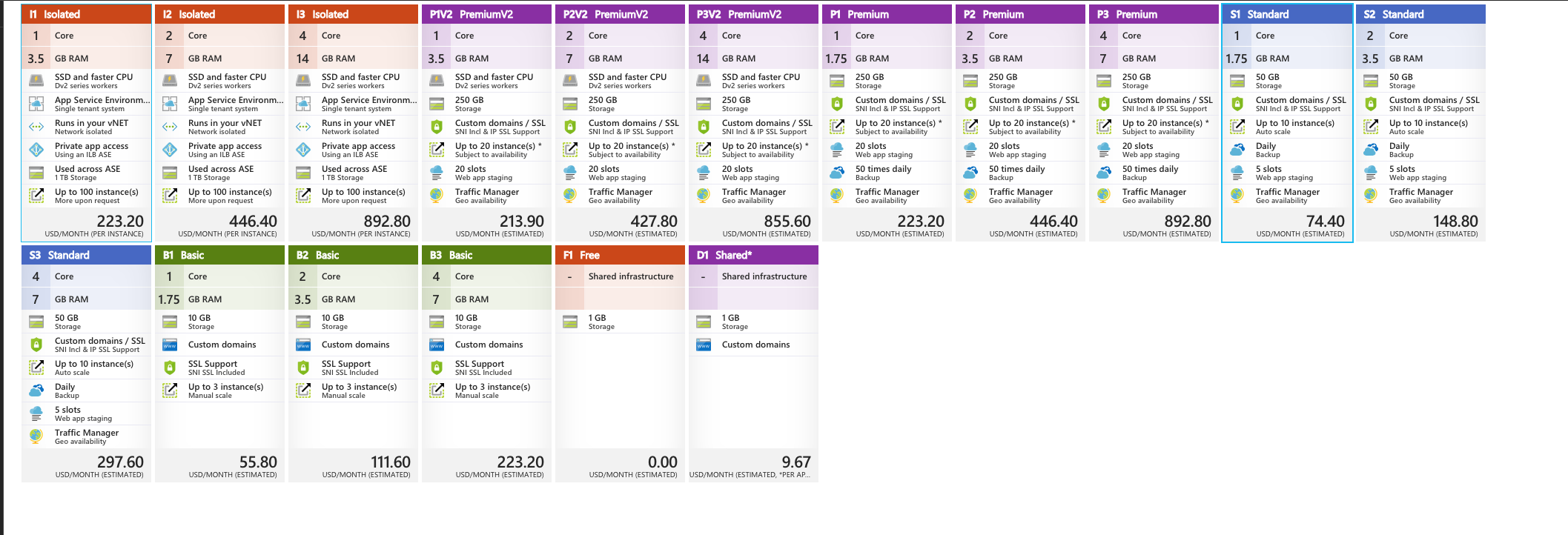
### Firewalls deployed in Azure (in the same account/subscription)

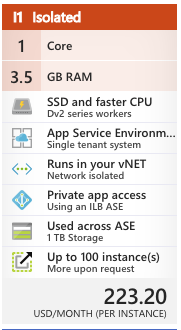
For firewalls that do not have publically addressable management interfaces you can choose the Hosting Plan to be an App Service Plan. This will allow you to “associate” the App Service Plan with a vNET. This vNET should be the vNET where all the firewall(s) reside.



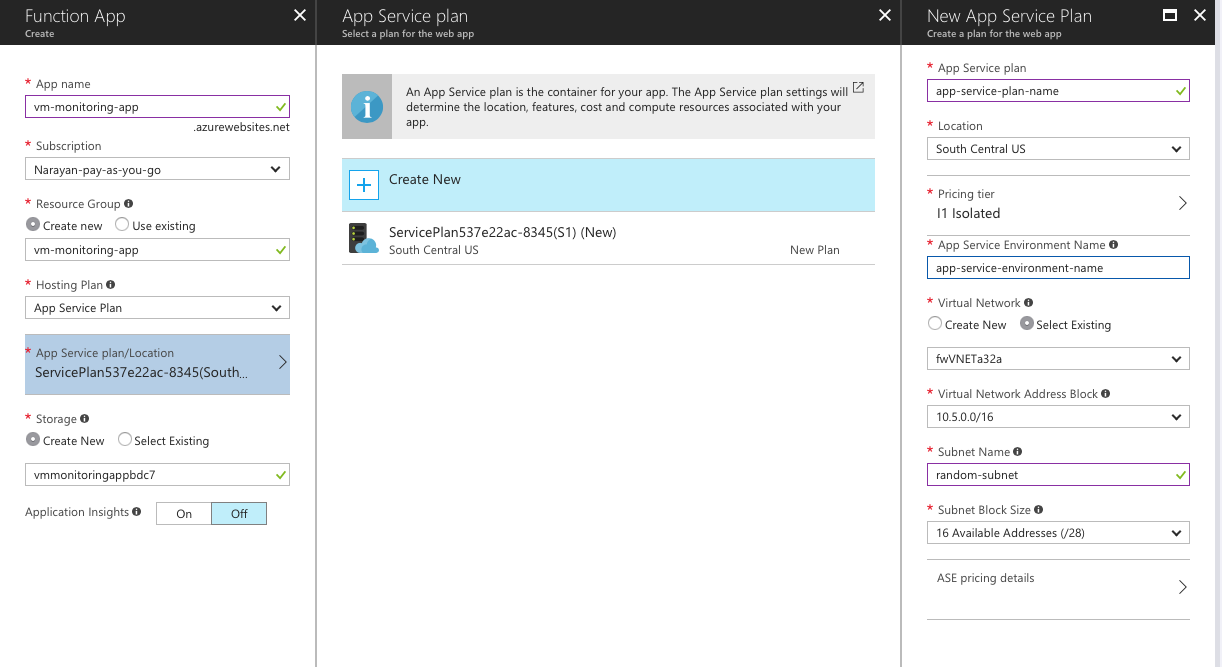
In order to deploy the Azure function in a vNET, you will need to create a new App Service Plan, and select the I1 Isolated Pricing Tier

Note: Before selecting the Isolated Tier, I would recommend checking the various pricing tiers.





You will need the tier that offers the option “Runs in your vNET (Network Isolated)”. As of writing this document, the cheapest tier that offers network isolation is the I1 tier.



Once you have selected your vNET, and filled in the required parameters, Create the function.

Note: Creating of the App service environment takes a very long time. Sometimes it can take about 2 hours.

# Clone the GitHub Repository

The next step is to clone the repo that hold the Azure Function script.

Please follow instructions listed here:

<https://help.github.com/articles/cloning-a-repository/>

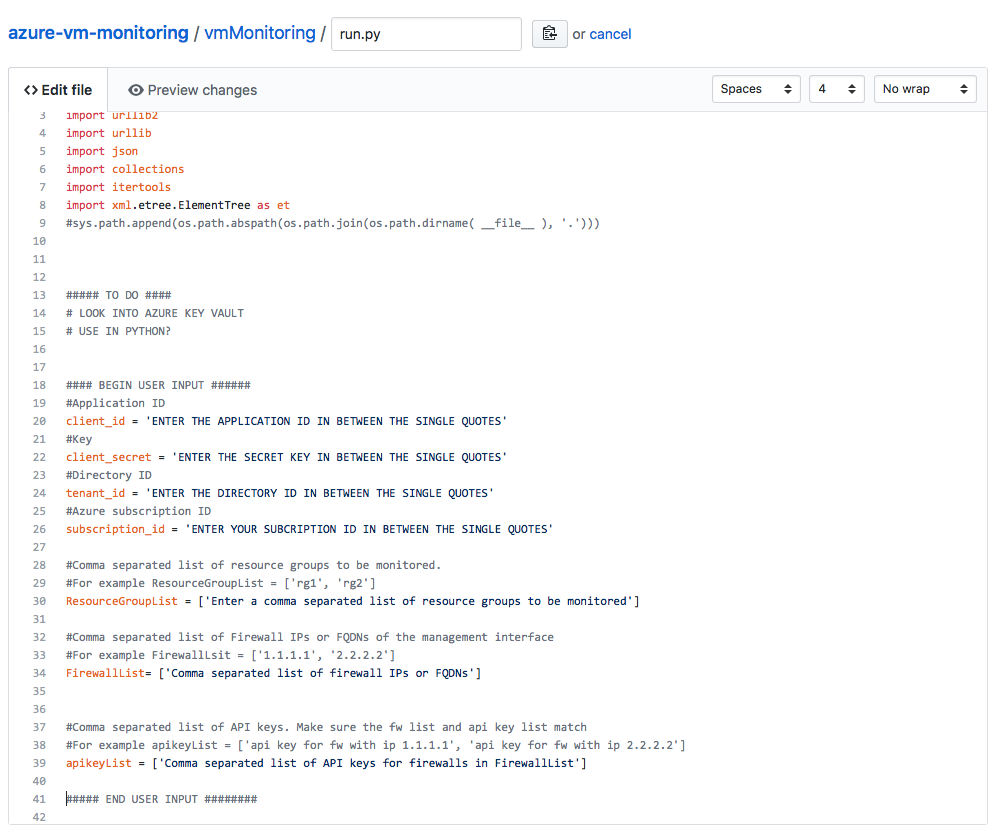
The URL for the repository to clone is:

<https://github.com/PaloAltoNetworks/azure-vm-monitoring.git>

**NOTE: Please make sure to change the repository from PUBLIC to PRIVATE as the script will hold some sensitive key information.**

# Edit the script

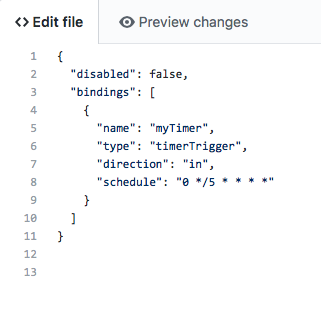
Open the run.py file, in the vmMonitoring directory in the GitHub portal itself by clicking the  icon and follow the comments in the file to make the appropriate changes:



Commit the changes to the run.py file.

# Modify the timer

By default the script runs every 5 minutes. If you want to change the timer, edit the function.json file and modify the schedule.



The value of schedule is a CRON expression. More here: <https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-timer>

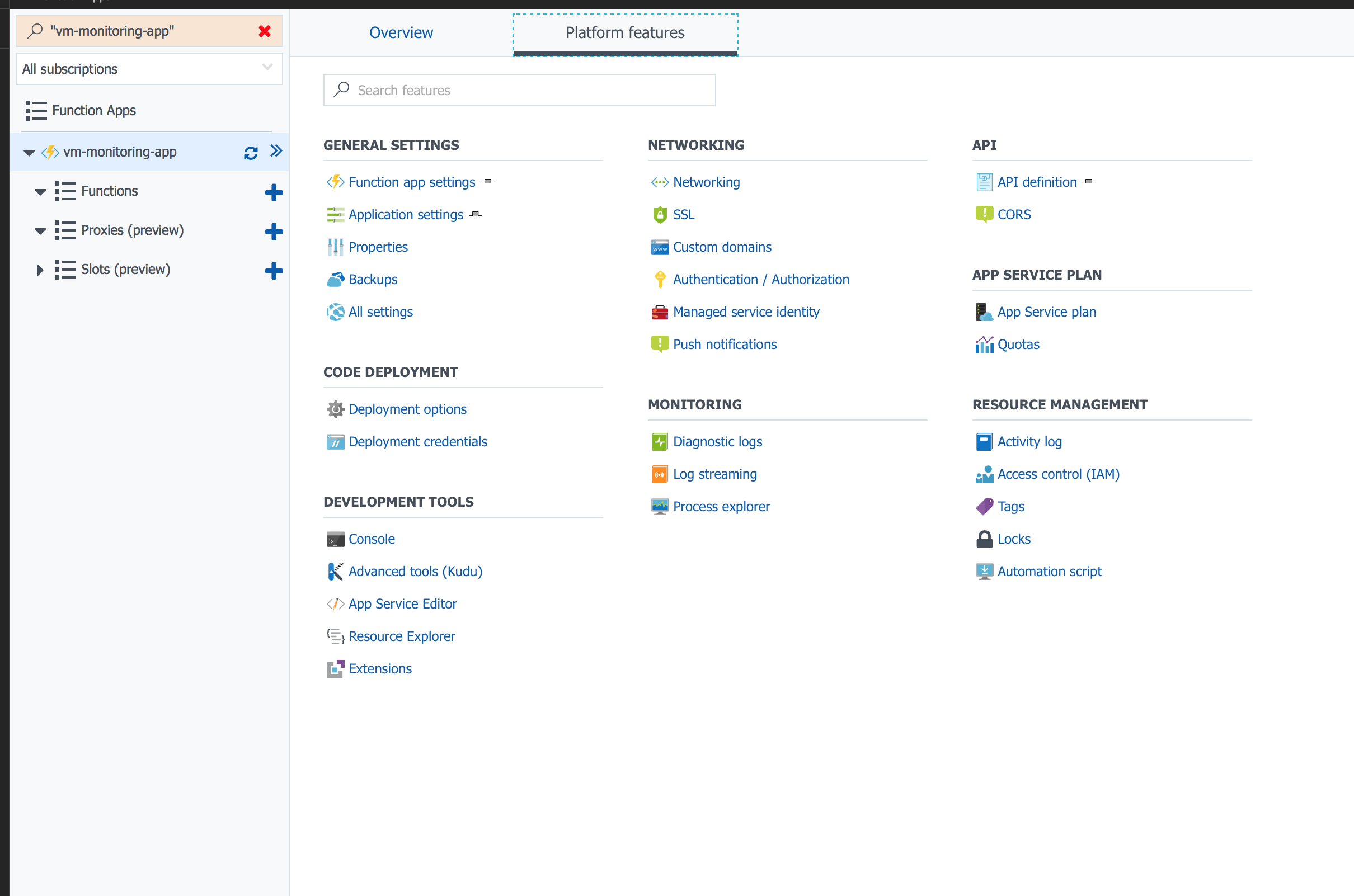
For example: Change the 5 to 1, to run the function every 1 minute.

Commit the changes to the function.json file

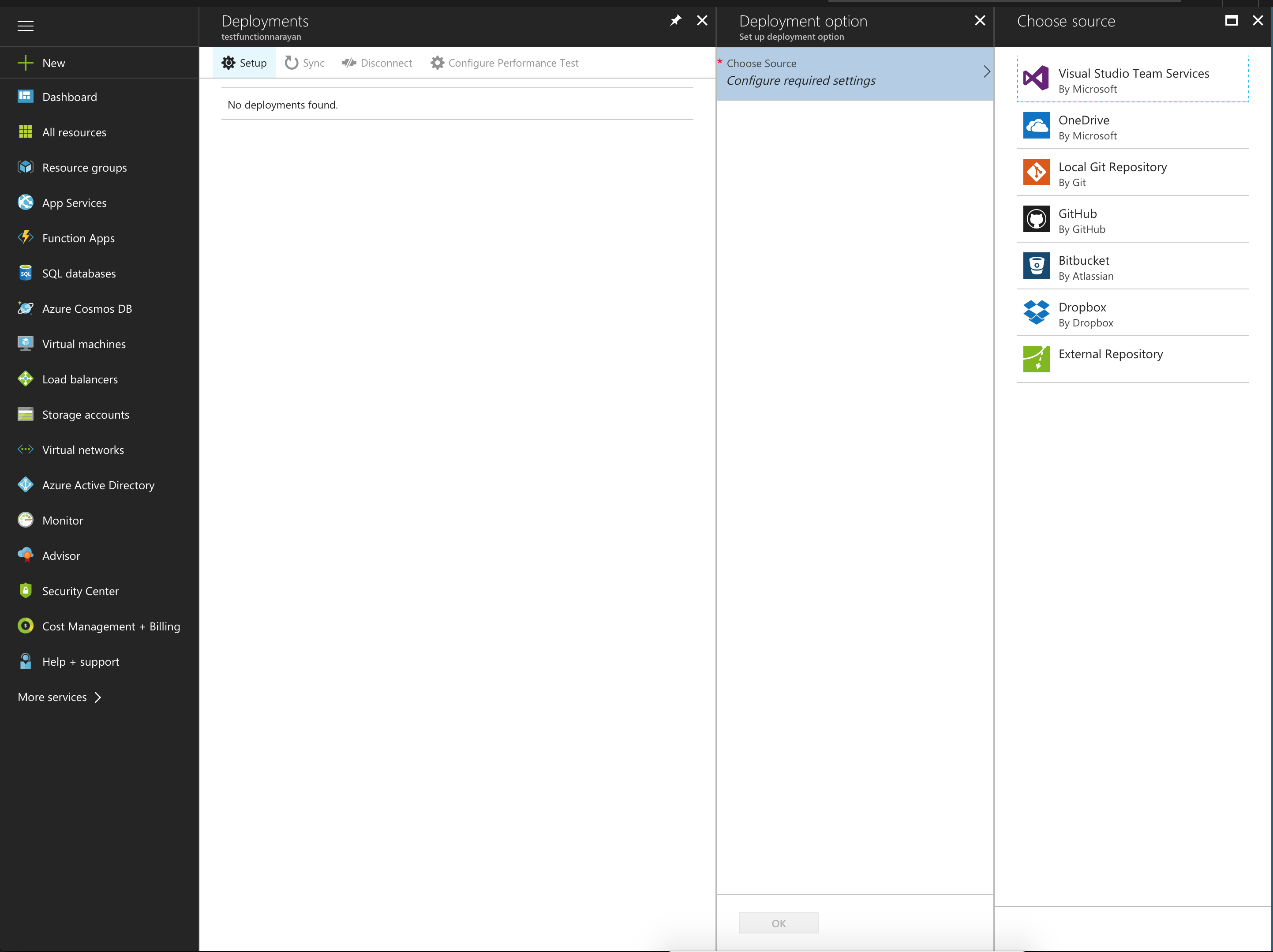
# Setup Continuous Deployment for your Azure Function

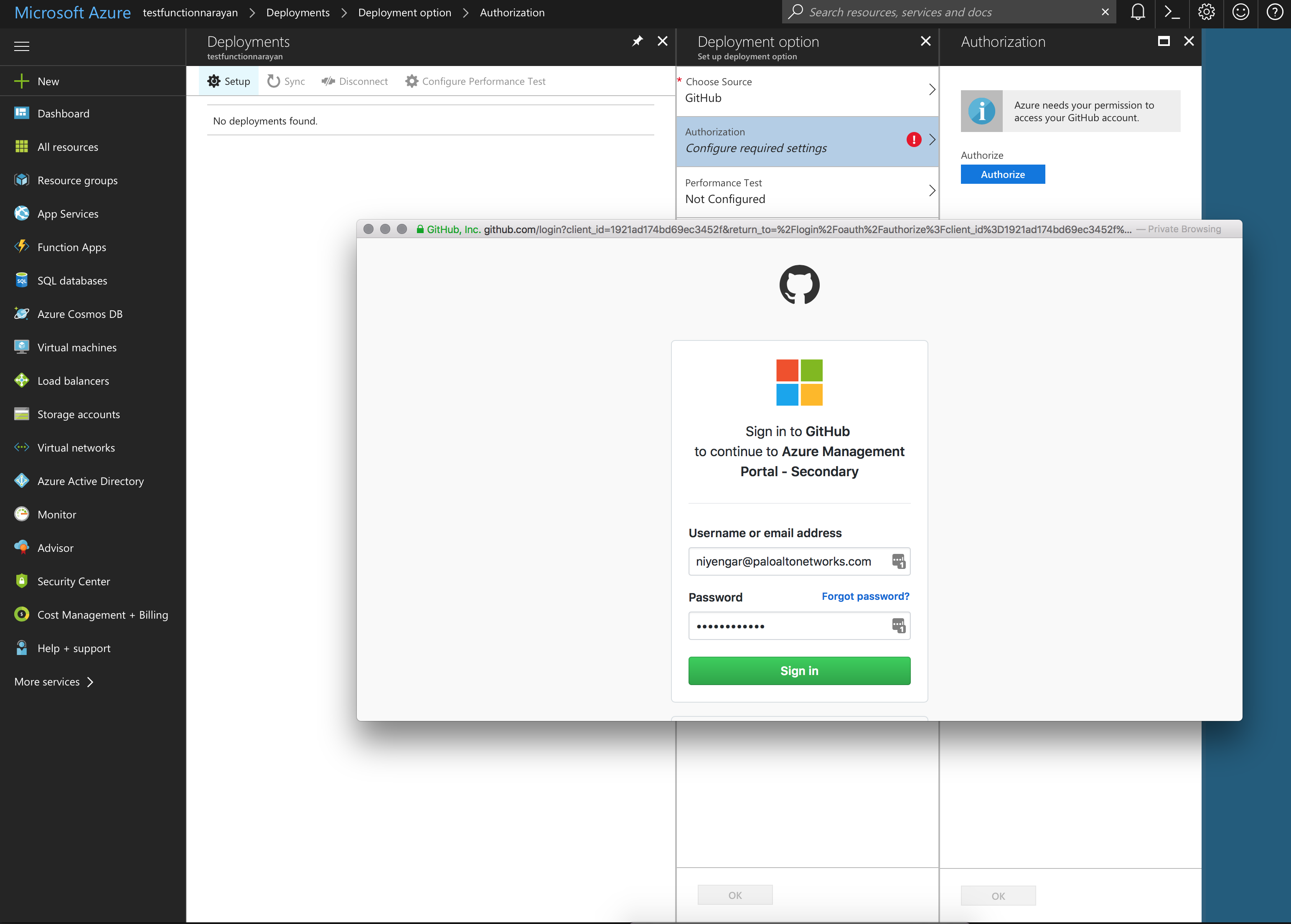
Once the Function App has been created, you can connect the GitHub repo to the function, to enable continuous deployment.

Navigate to the Function App blade on your Azure portal and click on the function that was just created. And click on the Platform Features tab, and click on Deployment Options



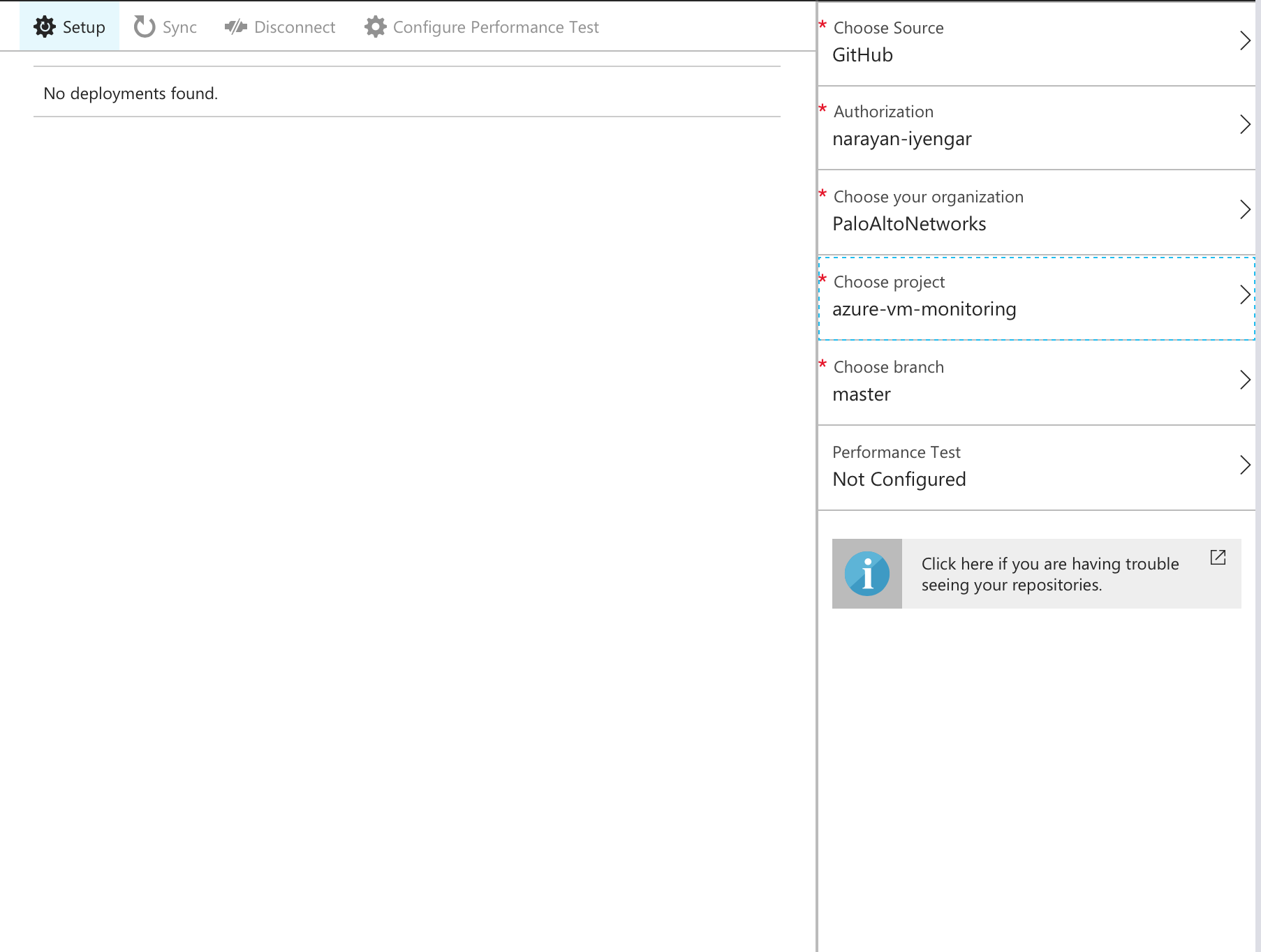
If you have never connected your GitHub account to Azure then you should be able to click on Setup and login and authorize Azure to access your GitHub profile



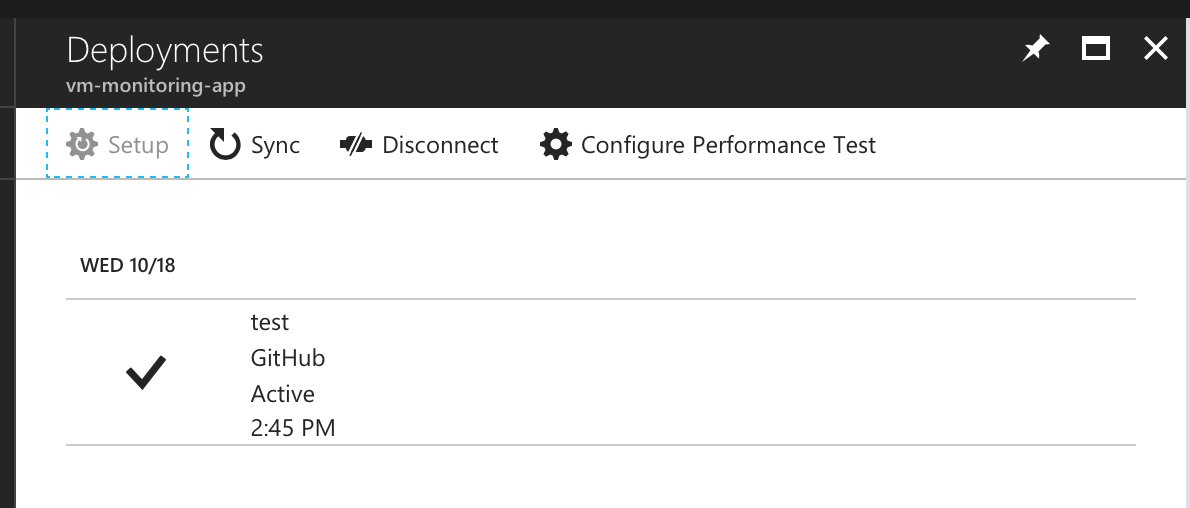


Once signed in and Authorized, you can select the GitHub repo where your code has been modified.

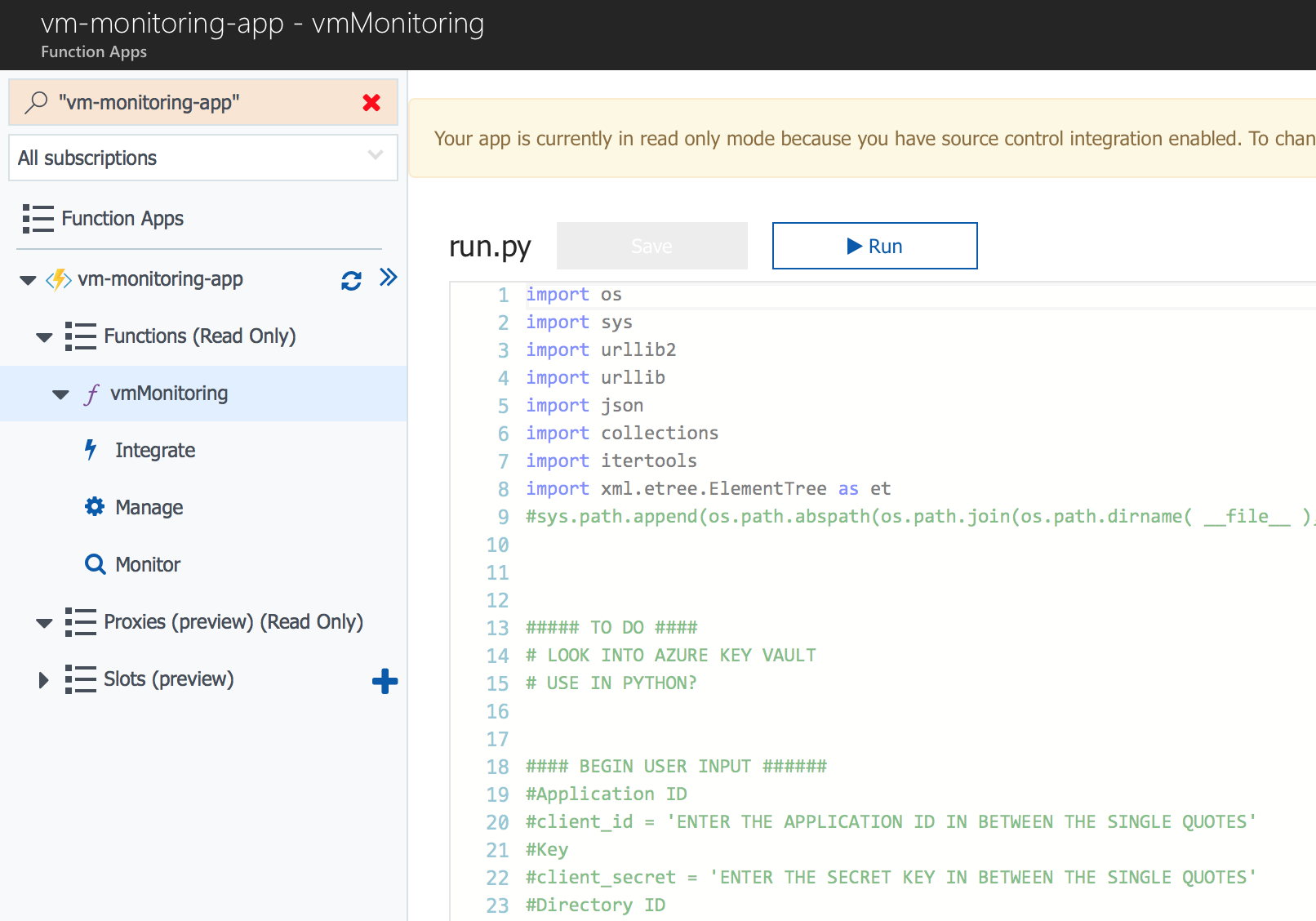
Select the appropriate repo and branch. You can leave the Performance Test option un-configured.



Once setup, the code is automatically pulled from GitHub and deployed.



You can now select the function and manually Run to verify correct operation.





Your Azure function is now setup to monitor you resource groups and push ip-to-tag mappings to the firewalls of your choosing.

Note: The function will only push updates to the firewalls if it finds changes in the tags. It will register new ip addresses to tags and unregister ip addresses that are no longer present from the firewalls.

Now you can log into the firewall and check the ip to tag mappings.

