SERVERLESS LAPS WITH INTUNE, FUNCTION APP AND KEY VAULT

SERVERLESS LAPS WITH INTUNE, FUNCTION APP AND KEY VAULT

Linkedin (https://www.linkedin.com/shareArticle?trk=Serverless+LAPS+with+Intune%2C+Function+App+and+Key+Vault&url=https%3A%2F%2Fwww.cloud-boy.be%2Fblog%2Fserverless-laps-with-intune-function-app-and-key-vault%2F)

Share (https://www.facebook.com/sharer.php?u=https%3A%2F%2Fwww.cloud-boy.be%2Fblog%2Fserverless-laps-with-intune-function-app-and-key-vault%2F)

Tweet (https://twitter.com/intent/tweet?text=Serverless%20LAPS%20with%20Intune%2C%20Function%20App%20and%20Key%20Vault&url=https
boy.be/blog/serverless-laps-with-intune-function-app-and-key-vault/&via=_Cloud_boy)

Whatsapp (https://api.whatsapp.com/send?text=Serverless%20LAPS%20with%20Intune%2C%20Function%20App%20and%20Key%20Vault%20https
boy.be%2Fblog%2Fserverless-laps-with-intune-function-app-and-key-vault%2F)

Mail (mailto:?subject=%20&body=%20https%3A%2F%2Fwww.cloud-boy.be%2Fblog%2Fserverless-laps-with-intune-function-app-and-key-vault%2F)

This article will describe how setup Serverless LAPS with Intune, Function App and Key Vault.

Situation:

- Full cloud device management (Azure AD Joined devices, Intune managed)
- No LAPS solution, because of no on-premise Active Directory

Microsoft Local Administrator Password Solution (LAPS) is a password manager that utilises Active Directory to manage and rotate passwords for local Administrator accounts across all of your Windows endpoints. LAPS is a great mitigation tool against lateral movement and privilege escalation, by forcing all local Administrator accounts to have unique, complex passwords, so an attacker compromising one local Administrator account can't move laterally to other endpoints and accounts that may share that same password. A benefit, compared to other password managers, is that LAPS does not require additional computers, or application servers, to manage these passwords. The management of these passwords is done entirely through Active Directory components.

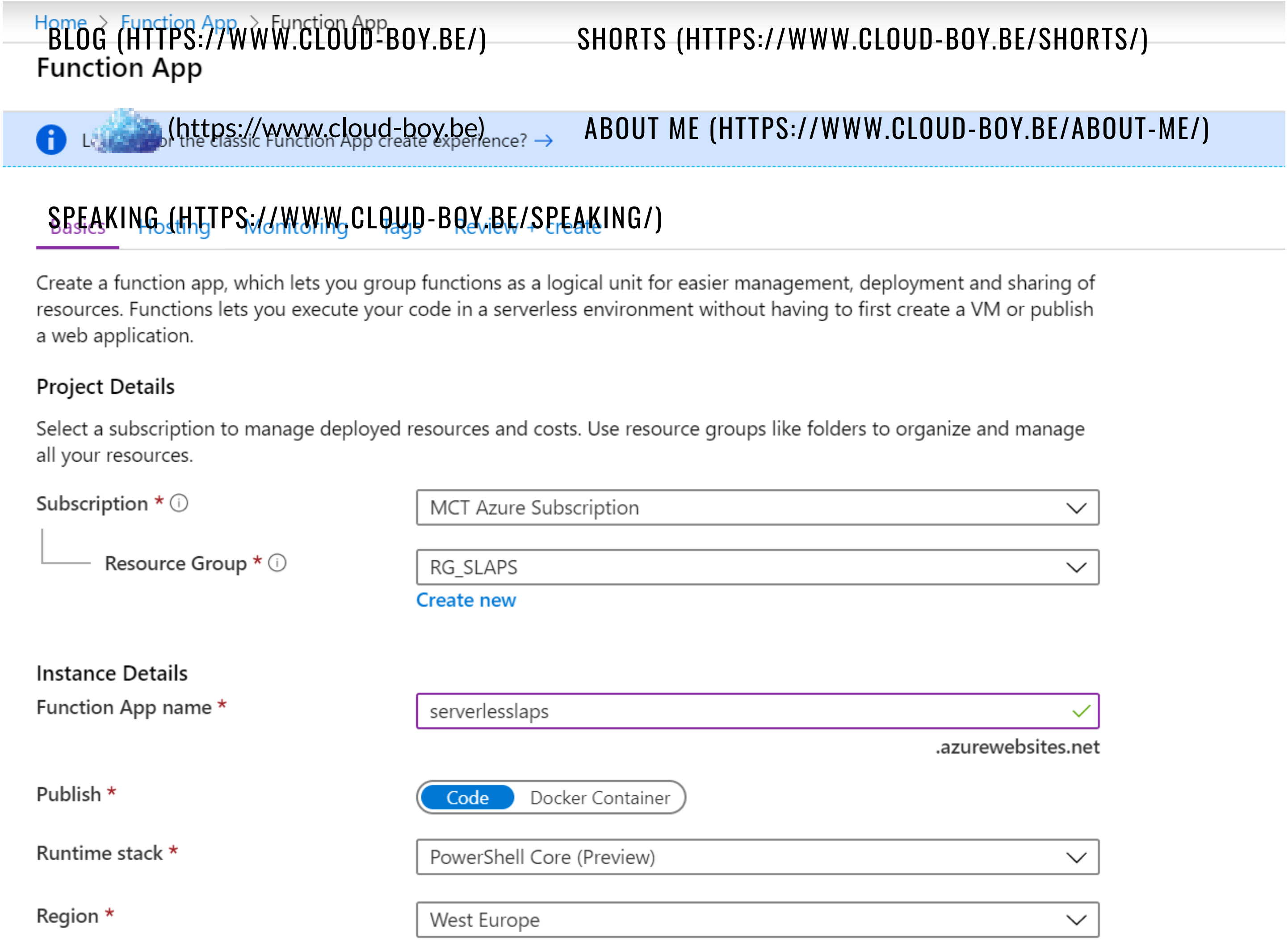
Target:

- Deploying LAPS, serverless, without an on-premise Active Directory.

I've stumbled across this blog post: https://www.srdn.io/2018/09/serverless-laps-powered-by-microsoft-intune-azure-functions-and-azure-key-vault/ (https://www.srdn.io/2018/09/serverless-laps-powered-by-microsoft-intune-azure-functions-and-azure-key-vault/) but it's a bit outdated with all the Azure changes in the last year, so I decided to update it. Credits to John Seerden for the PowerShell scripts though.

1. DEPLOY AN AZURE FUNCTION APP & CONFIGURE IT

In the Azure Portal, navigate to Function Apps and click on 'Add' to create a new Function App. Choose a Subscription and a Resource Group (or create a new one), give your Function App a name and as Runtime stack choose 'PowerShell Core (Preview)'. Click on 'Next'.


Home > Function App > Function App
Function App
L... or the classic Function App create experience? →
Basics
Hosting
Monitoring
Tags
Review + create

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 * : MCT Azure Subscription

Resource Group * : RG_SLAPS

Create new

Instance Details

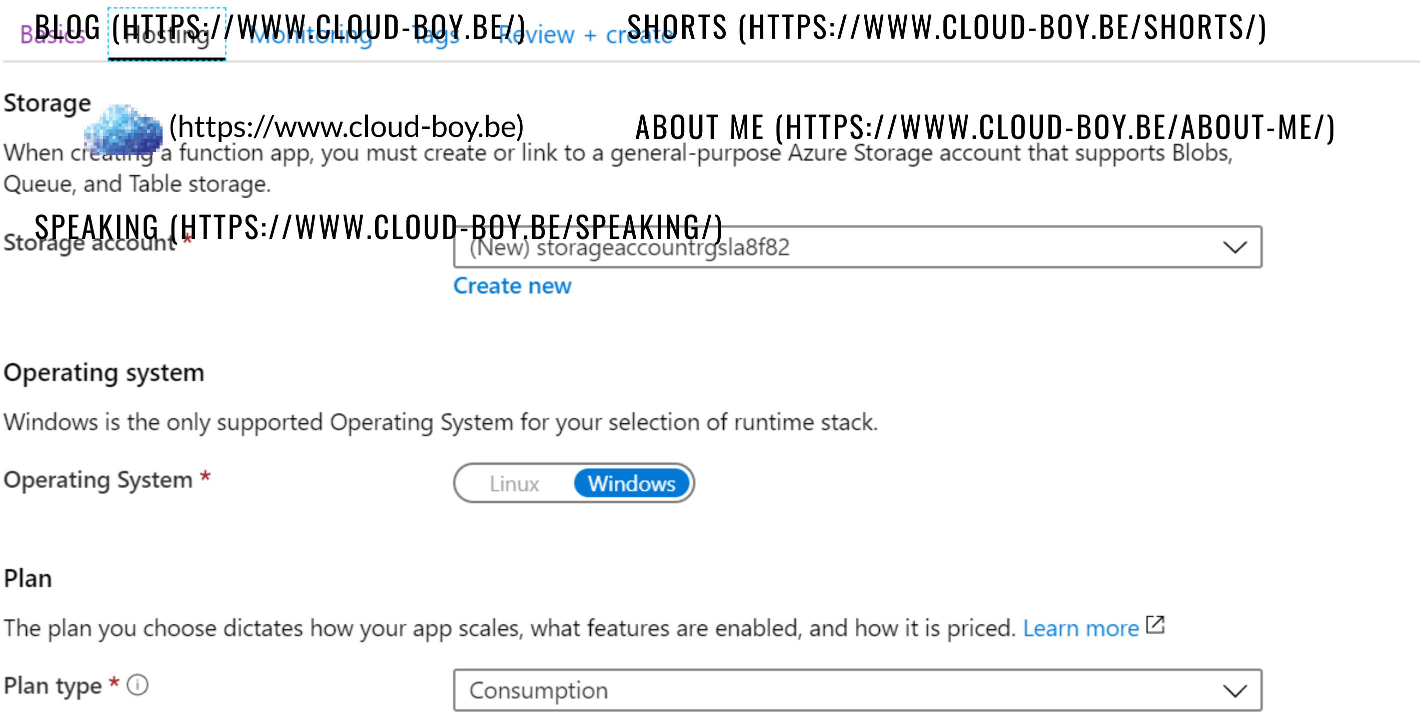
Function App name * : serverlesslaps .azurewebsites.net

Publish * : Code | Docker Container

Runtime stack * : PowerShell Core (Preview)

Region * : West Europe

Choose or create a new storage account, leave the Operating System setting on 'Windows'. And choose a plan type. In my scenario I choose for Consumption. Click on 'Next'.



Storage

When creating a function app, you must create or link to a general-purpose Azure Storage account that supports Blobs, Queue, and Table storage.

Storage account * (New) storageaccountrgsla8f82

Create new

Operating system

Windows is the only supported Operating System for your selection of runtime stack.

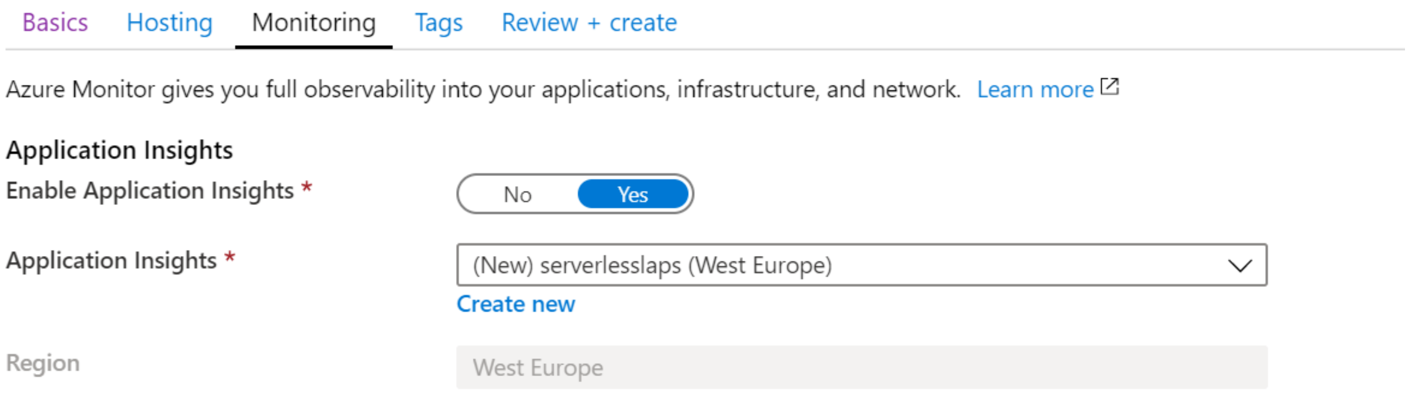
Operating System * Linux [✓ Windows]

Plan

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

Plan type * Consumption

Choose if you want to enable Application Insights (not necessary) and click on 'Review + Create' After the validation, click on 'Create' to deploy the Function App.



Azure Monitor gives you full observability into your applications, infrastructure, and network. Learn more

Application Insights

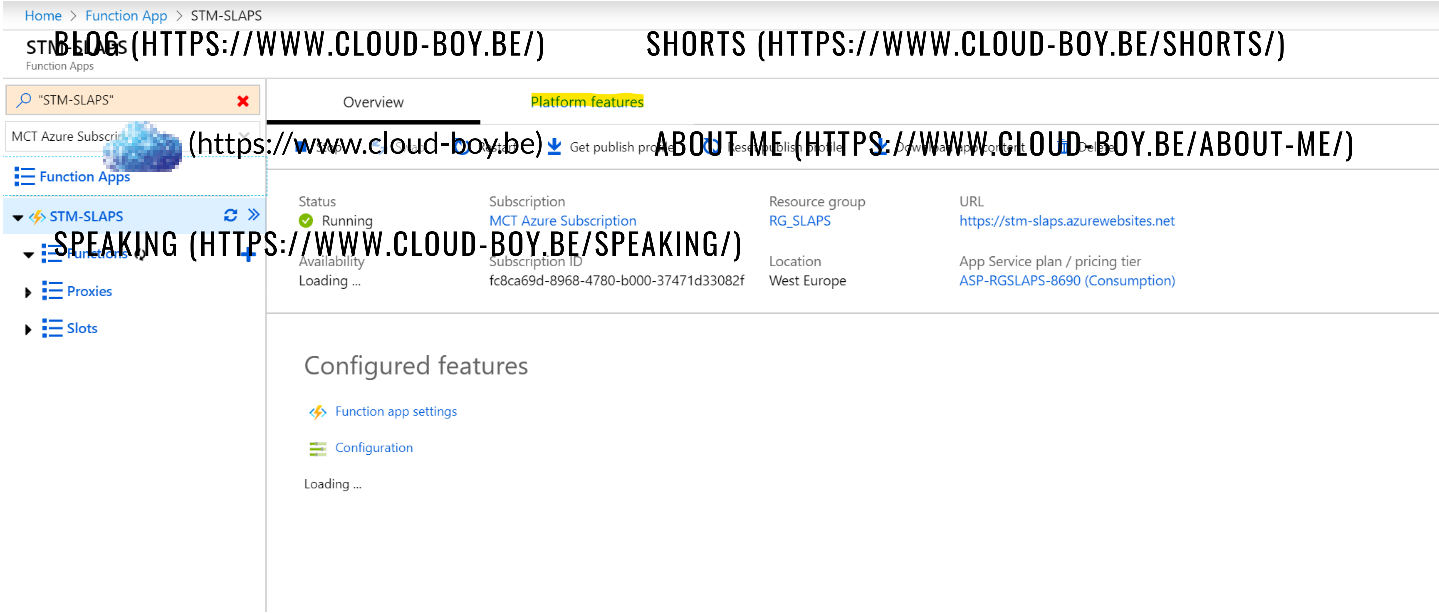
Enable Application Insights * No [✓ Yes]

Application Insights * (New) serverlesslaps (West Europe)

Create new

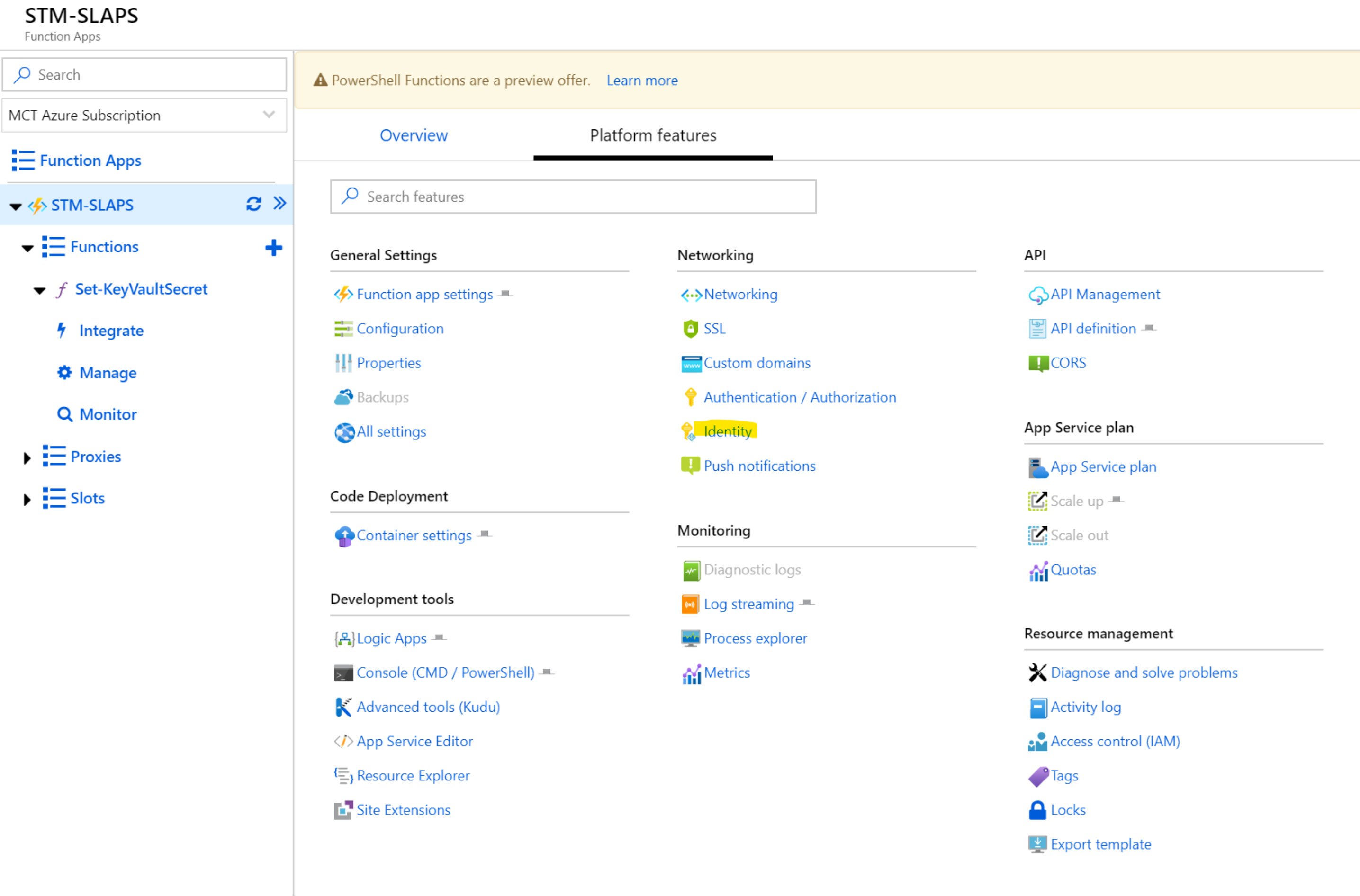
Region West Europe

Navigate to the function App, and click on 'Platform features'.



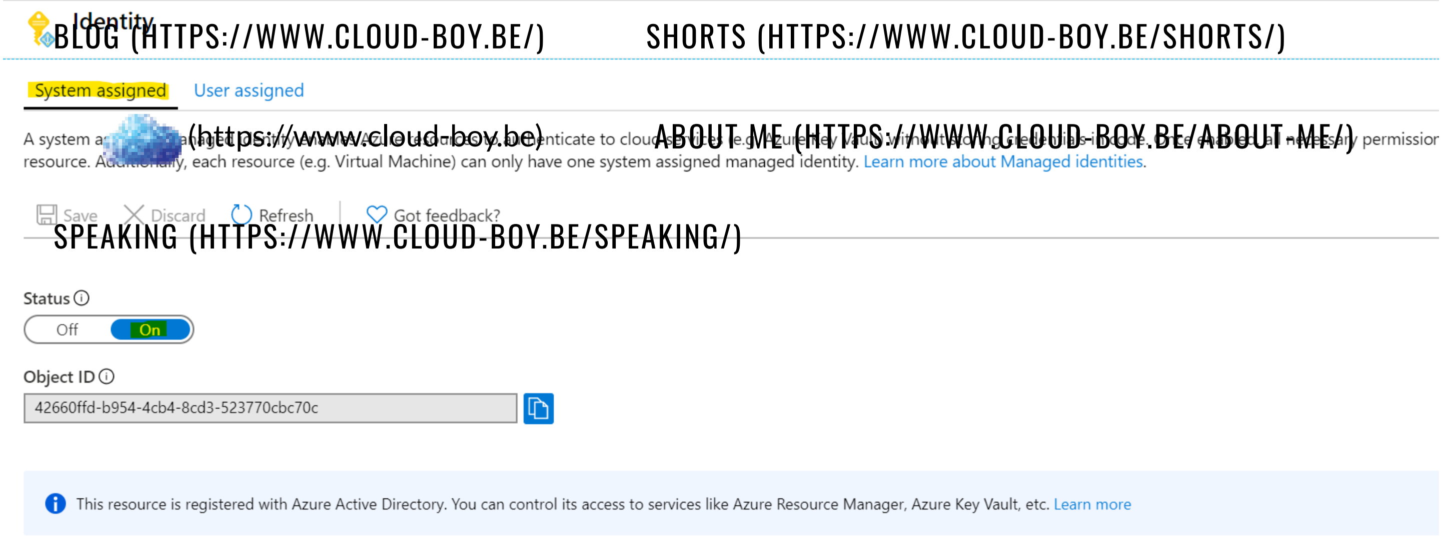
Status: Running
Availability: Loading ...
Subscription: MCT Azure Subscription
Subscription ID: fc8ca69d-8968-4780-b000-37471d33082f
Resource group: RG_SLAPS
Location: West Europe
URL: https://stm-slaps.azurewebsites.net
App Service plan / pricing tier: ASP-RGSLAPS-8690 (Consumption)

Click on 'Identity'.



In 'System Assigned' switch the status to 'On'. Click on 'Save'.

A system assigned managed identity enables Azure resources to authenticate to cloud services (e.g. Azure Key Vault) without storing credentials in code. We'll grant this managed identity access to our Key Vault later on.



Identity

System assigned User assigned

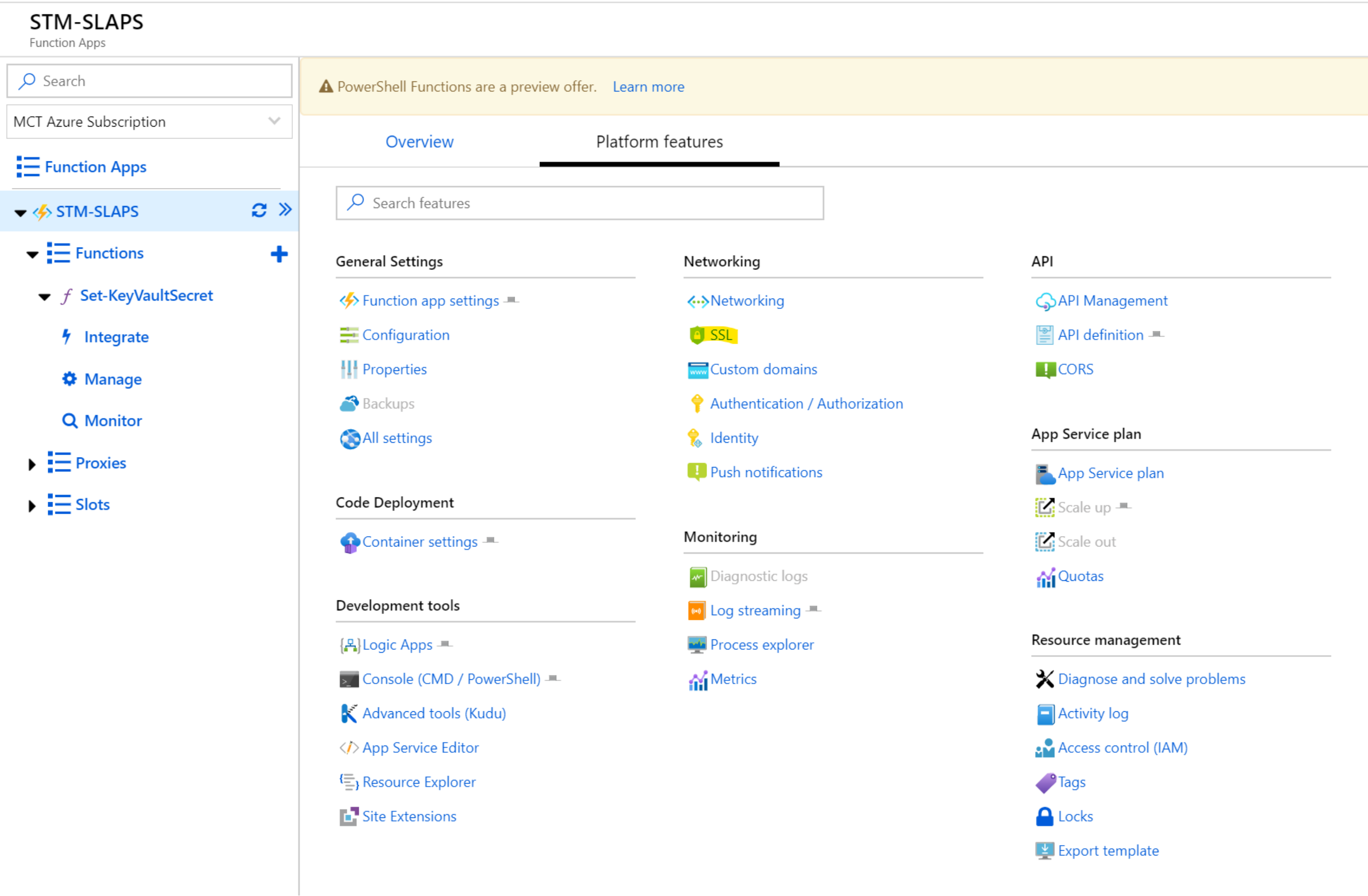
A system a anaged identity enables Azure resources to authenticate to cloud services (e.g. Azure Key Vault) without storing credentials in code. Once enabled, all necessary permission resource. Additionally, each resource (e.g. Virtual Machine) can only have one system assigned managed identity. Learn more about Managed identities.

Status: Off / On (On selected)

Object ID: 42660ffd-b954-4cb4-8cd3-523770cbc70c

This resource is registered with Azure Active Directory. You can control its access to services like Azure Resource Manager, Azure Key Vault, etc. Learn more

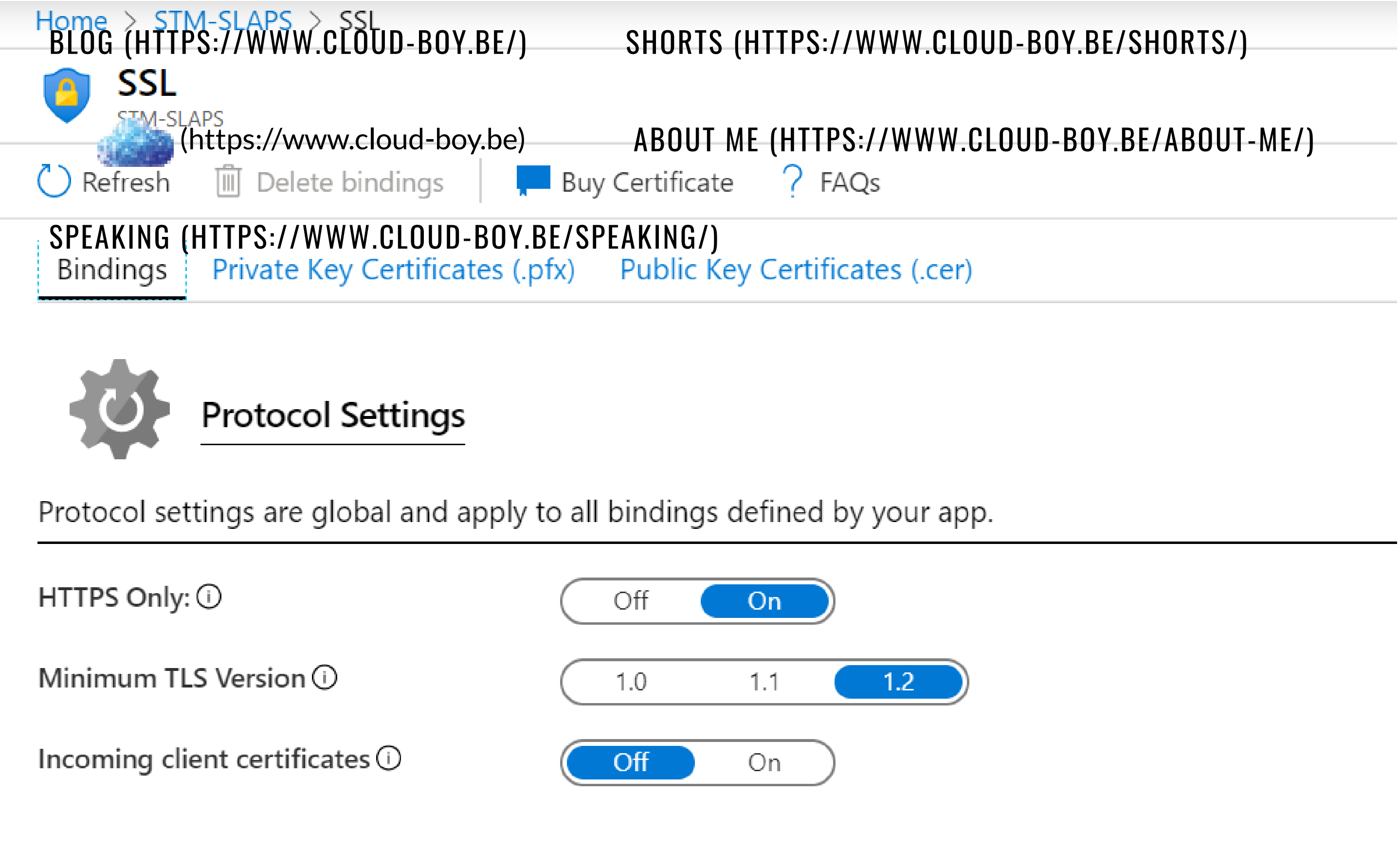
Our Function App also requires a minimum TLS version of 1.2, so go back to the 'Platform Features' and click on 'SSL'.



STM-SLAPS
Function Apps

PowerShell Functions are a preview offer. Learn more

Set HTTPS Only to 'On' and select Minimum TLS Version of '1.2'. Click on 'Refresh'.



Protocol Settings

Protocol settings are global and apply to all bindings defined by your app.

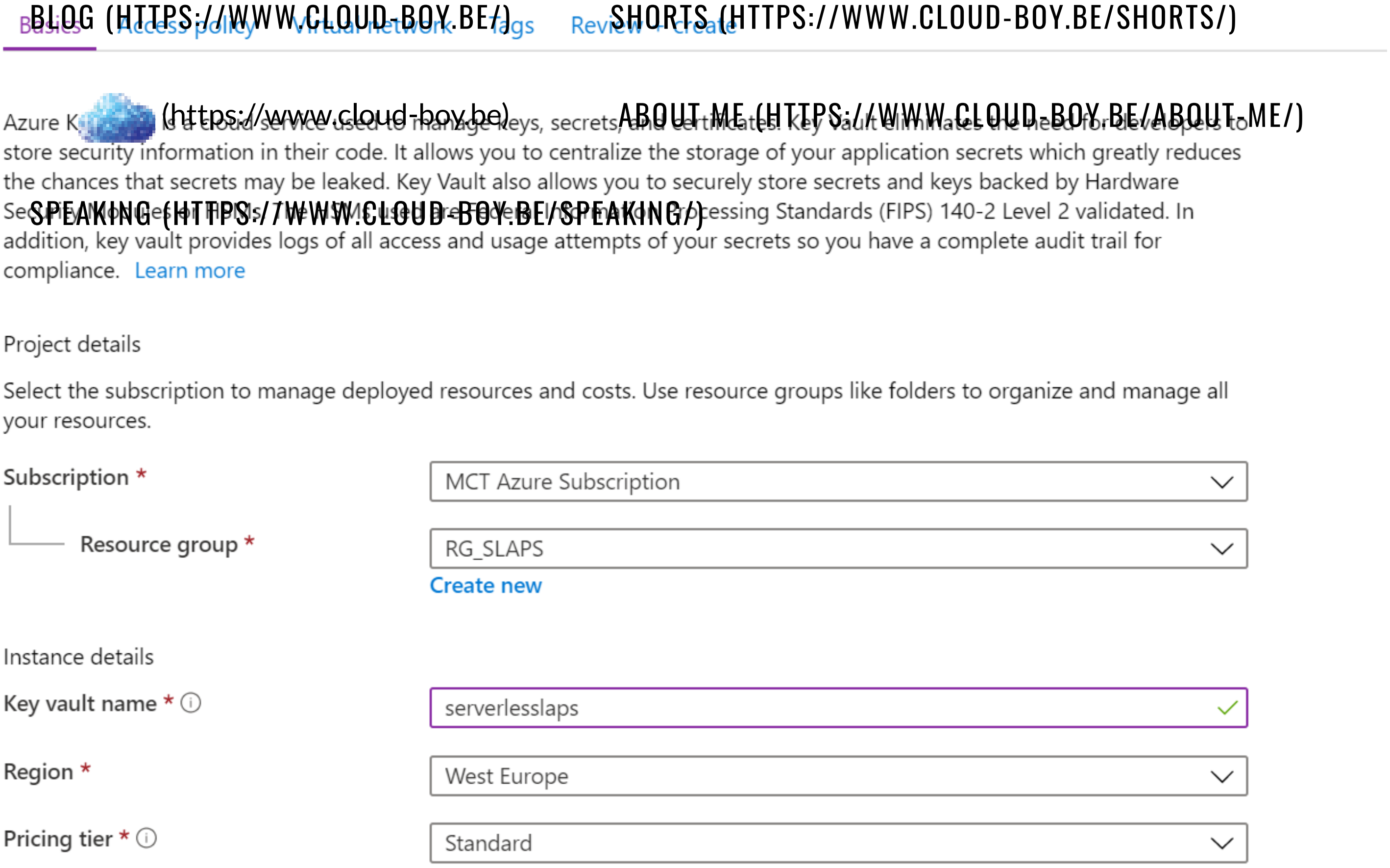
HTTPS Only: Off [✓ On]

Minimum TLS Version 1.0 1.1 [✓ 1.2]

Incoming client certificates [✓ Off] On

2. DEPLOY AN AZURE KEY VAULT AND GRANT OUR MANAGED SERVICE IDENTITY ACCESS

In the Azure Portal, navigate to Key Vaults and click on ‘Add’ to create a new Key Vault. Choose a Subscription and a Resource Group (or create a new one), give your Key Vault a name and leave the pricing tier on ‘Standard’. Click on ‘Next’.



Azure Key Vault is a cloud service used to manage keys, secrets, and certificates. Key Vault eliminates the need for developers to store security information in their code. It allows you to centralize the storage of your application secrets which greatly reduces the chances that secrets may be leaked. Key Vault also allows you to securely store secrets and keys backed by Hardware Security Modules or HSMs. The HSMs used are Federal Information Processing Standards (FIPS) 140-2 Level 2 validated. In addition, key vault provides logs of all access and usage attempts of your secrets so you have a complete audit trail for compliance. Learn more

Project details

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

Subscription * : MCT Azure Subscription

Resource group * : RG_SLAPS

Create new

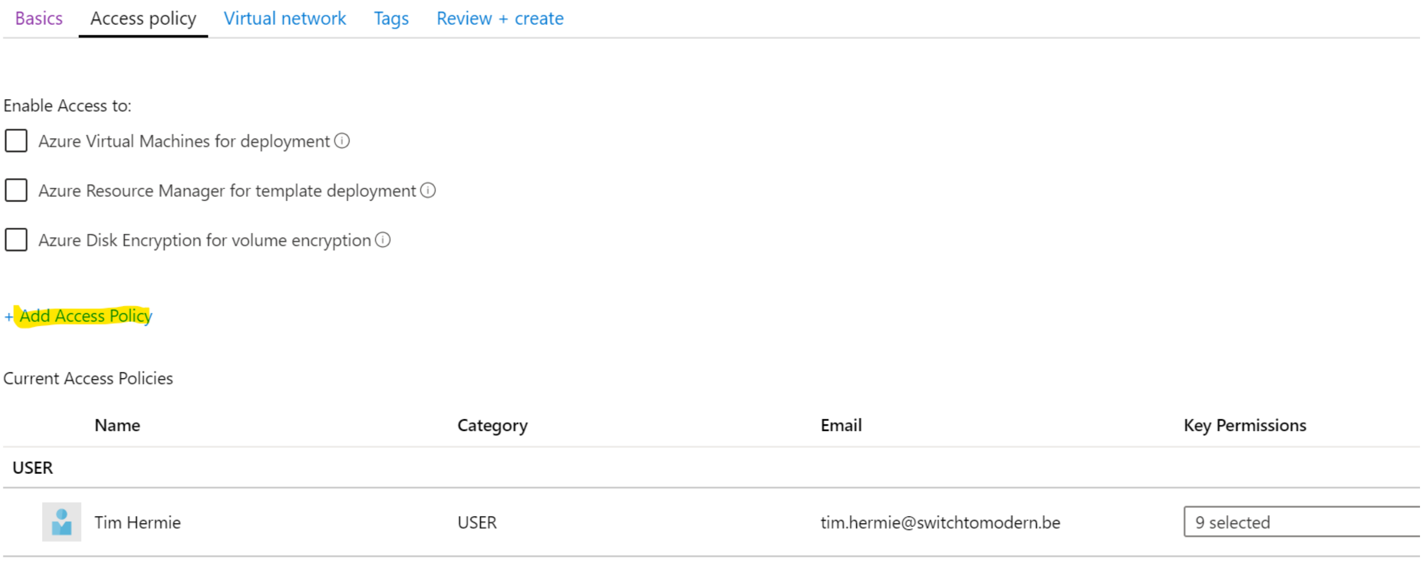
Instance details

Key vault name * : serverlessslaps

Region * : West Europe

Pricing tier * : Standard

Click on ‘Add Access Policy’.



Enable Access to:

☐ Azure Virtual Machines for deployment

☐ Azure Resource Manager for template deployment

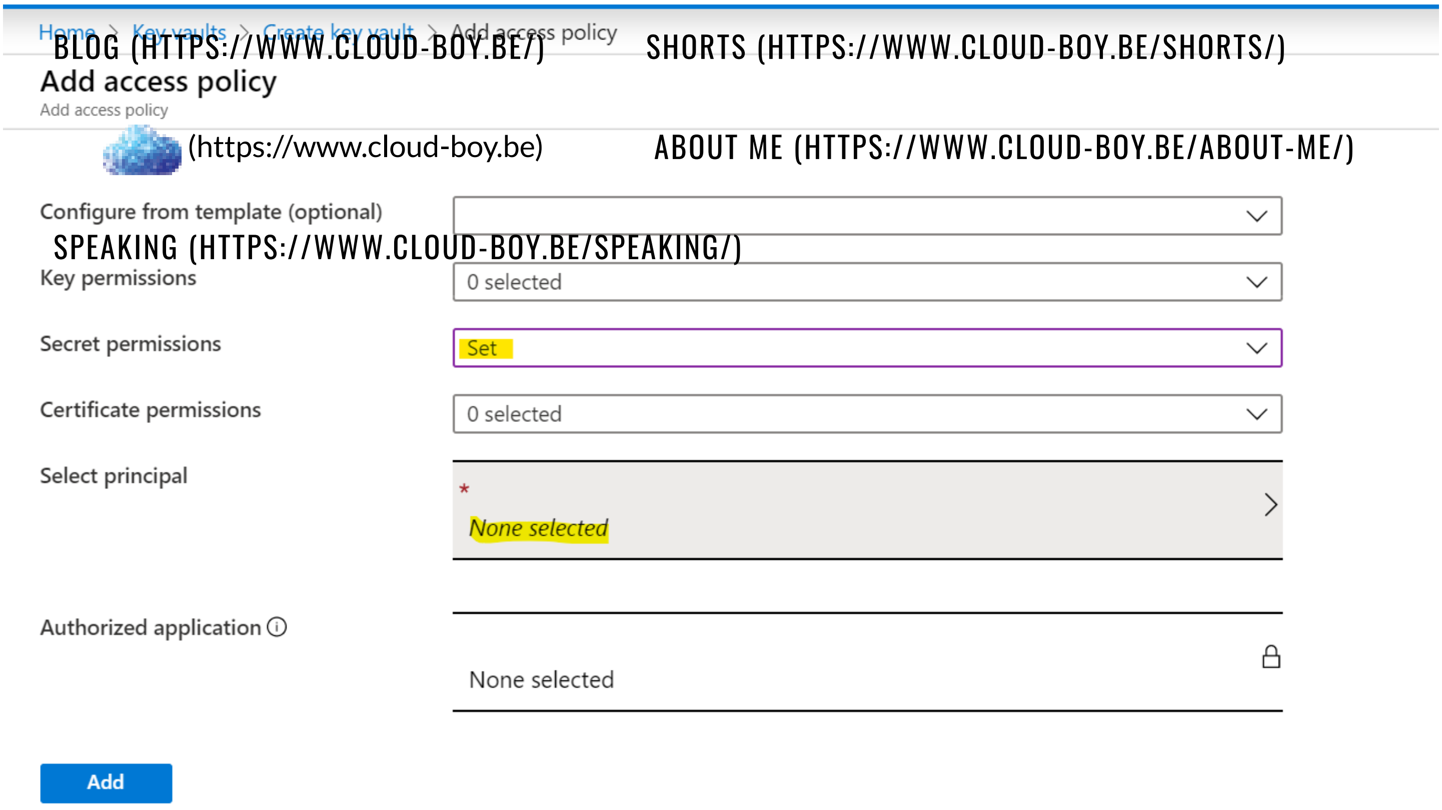
☐ Azure Disk Encryption for volume encryption

+ Add Access Policy

Current Access Policies

Name	Category	Email	Key Permissions
USER			
Tim Hermie	USER	tim.hermie@switchtomodern.be	9 selected

Select ‘Set’ in secret permissions. Afterwards click on ‘Select principal’.



Add access policy

Add access policy

Configure from template (optional):

Key permissions: 0 selected

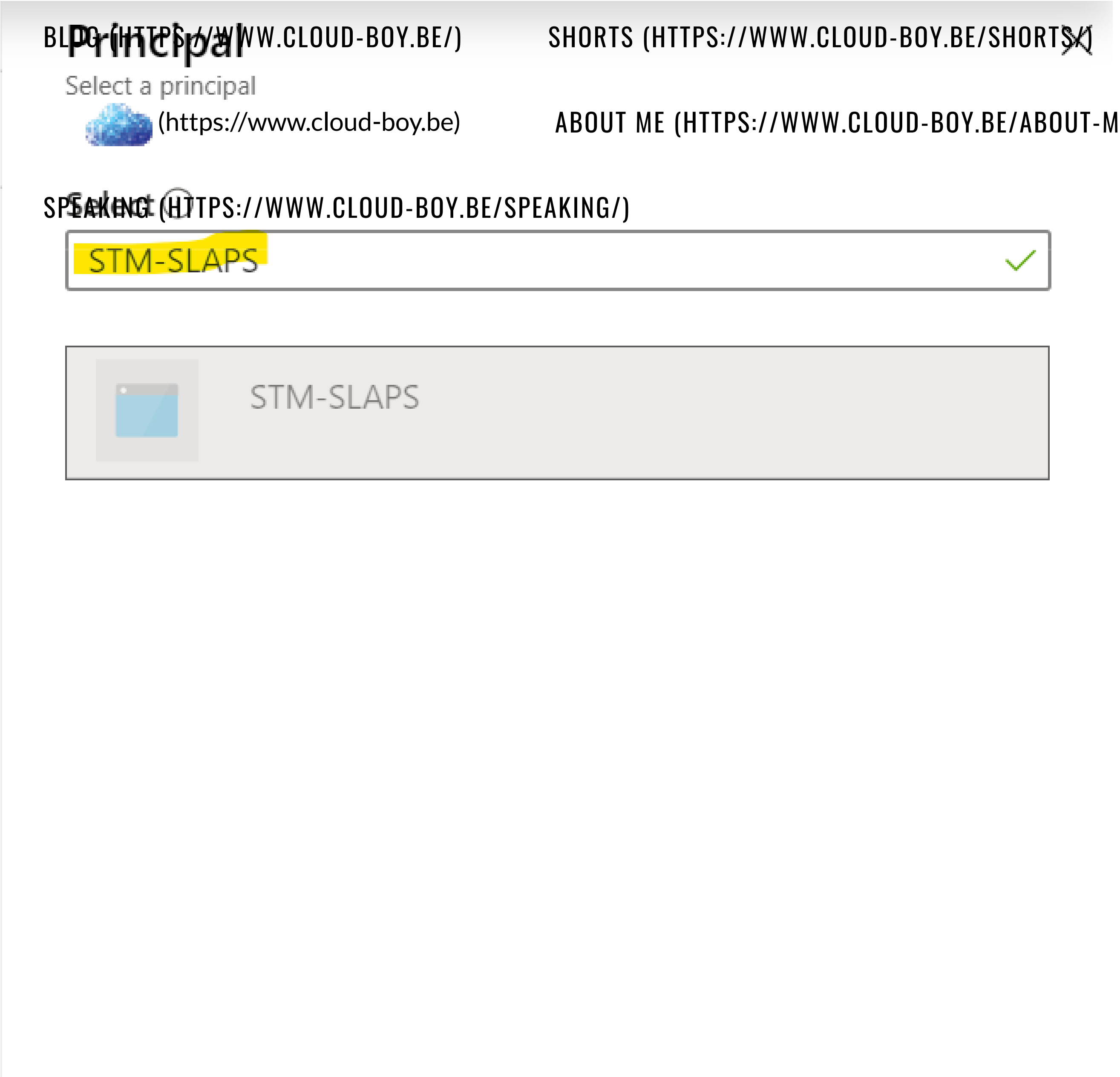
Secret permissions: Set

Certificate permissions: 0 selected

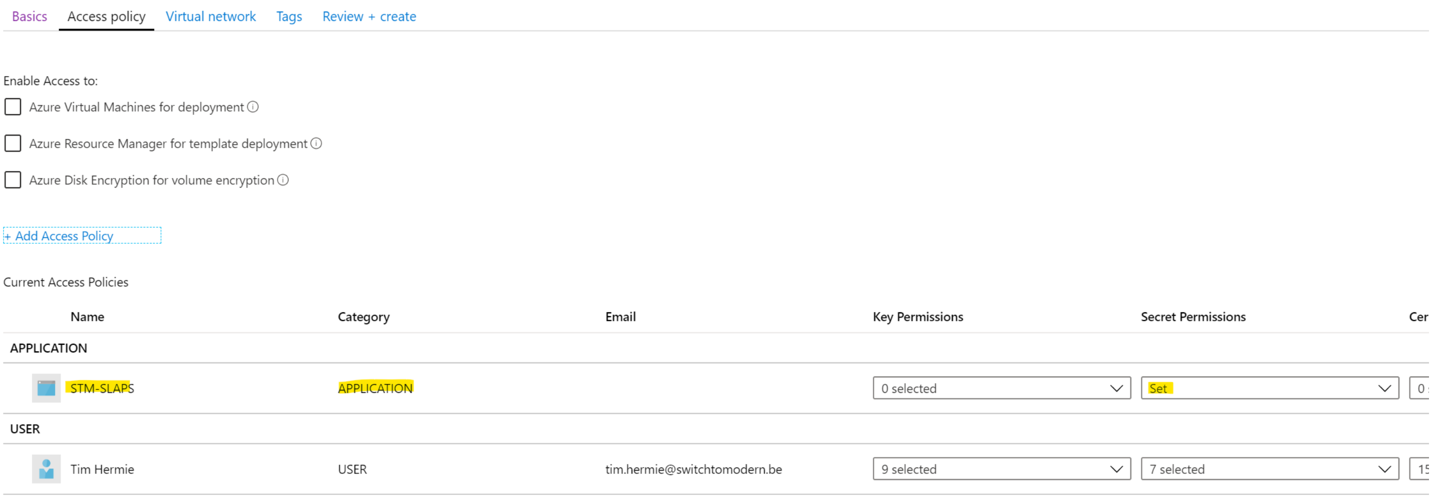
Select principal: * None selected

Authorized application: None selected

Here we will choose our newly made Funtion App (which we gave a system assigned managed identity). Select the principal & click on ‘Add’.



You’ll see this screen next when it’s done right:



Enable Access to:

☐ Azure Virtual Machines for deployment ⓘ

☐ Azure Resource Manager for template deployment ⓘ

☐ Azure Disk Encryption for volume encryption ⓘ

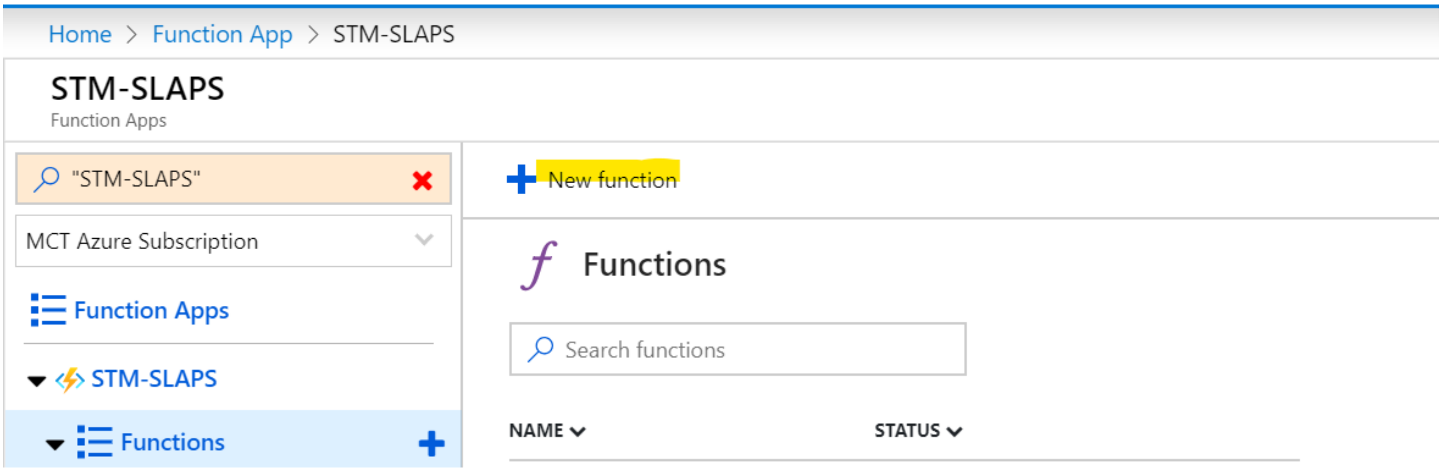
Current Access Policies

Name	Category	Email	Key Permissions	Secret Permissions	Cer
APPLICATION					
STM-SLAPS	APPLICATION		0 selected	Set	0
USER					
Tim Hermie	USER	tim.hermie@switchtomodern.be	9 selected	7 selected	15

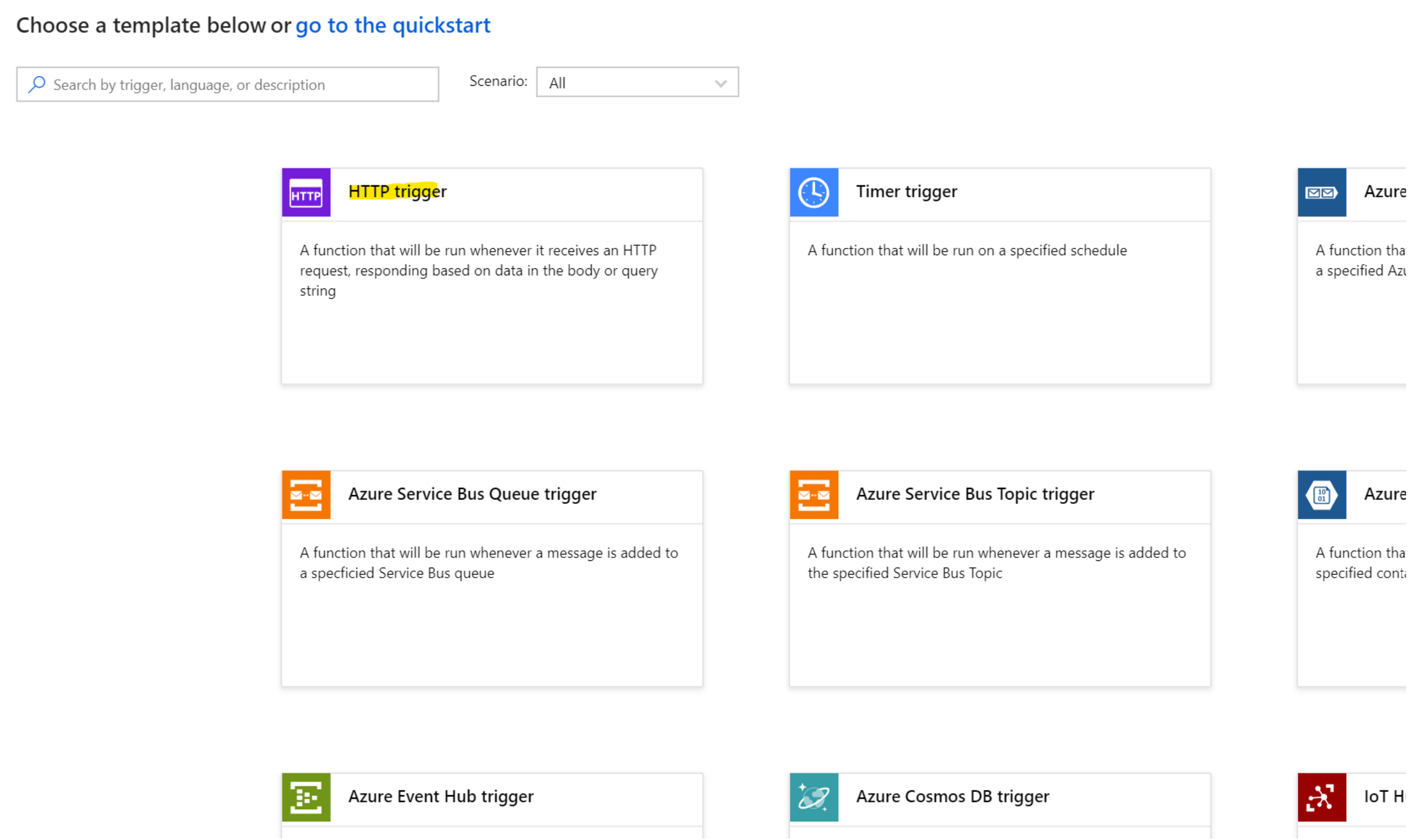
Click on ‘Review + Create’ and after validation click on ‘Create’.

3. CREATE AND TEST THE AZURE FUNCTION

Go back to your Function App. Select the Function and click on ‘New Function’.



Choose ‘HTTP Trigger’.


Choose a template below or go to the quickstart
Search by trigger, language, or description
Scenario: All

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

A function tha a specified Az

Azure Service Bus Queue trigger

A function that will be run whenever a message is added to a specficied 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

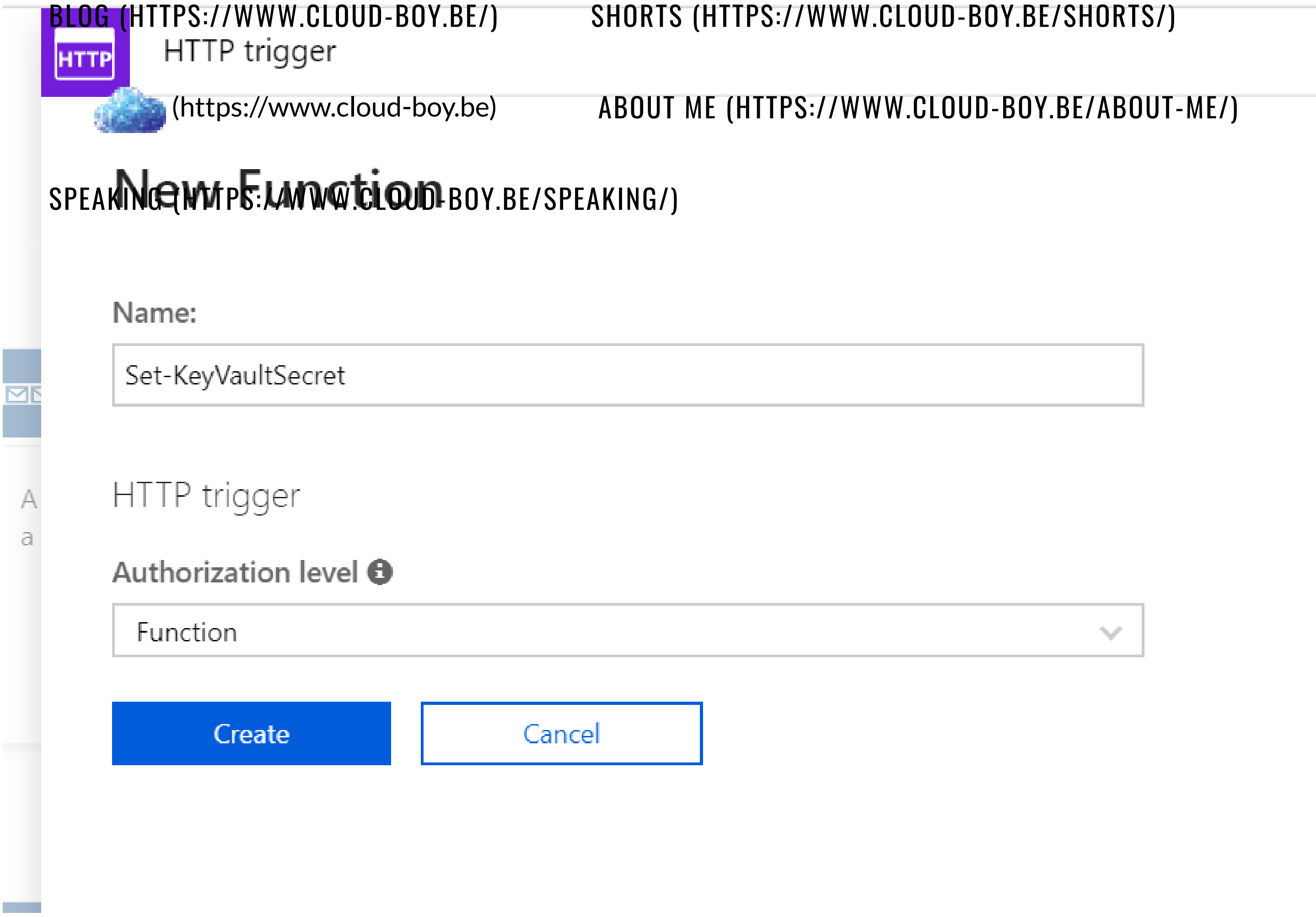
A function tha specified cont

Azure Event Hub trigger

Azure Cosmos DB trigger

IoT H

Give it the name ‘Set-KeyVaultSecret’. Authorisation level is ‘Function’. Click on ‘Create’.

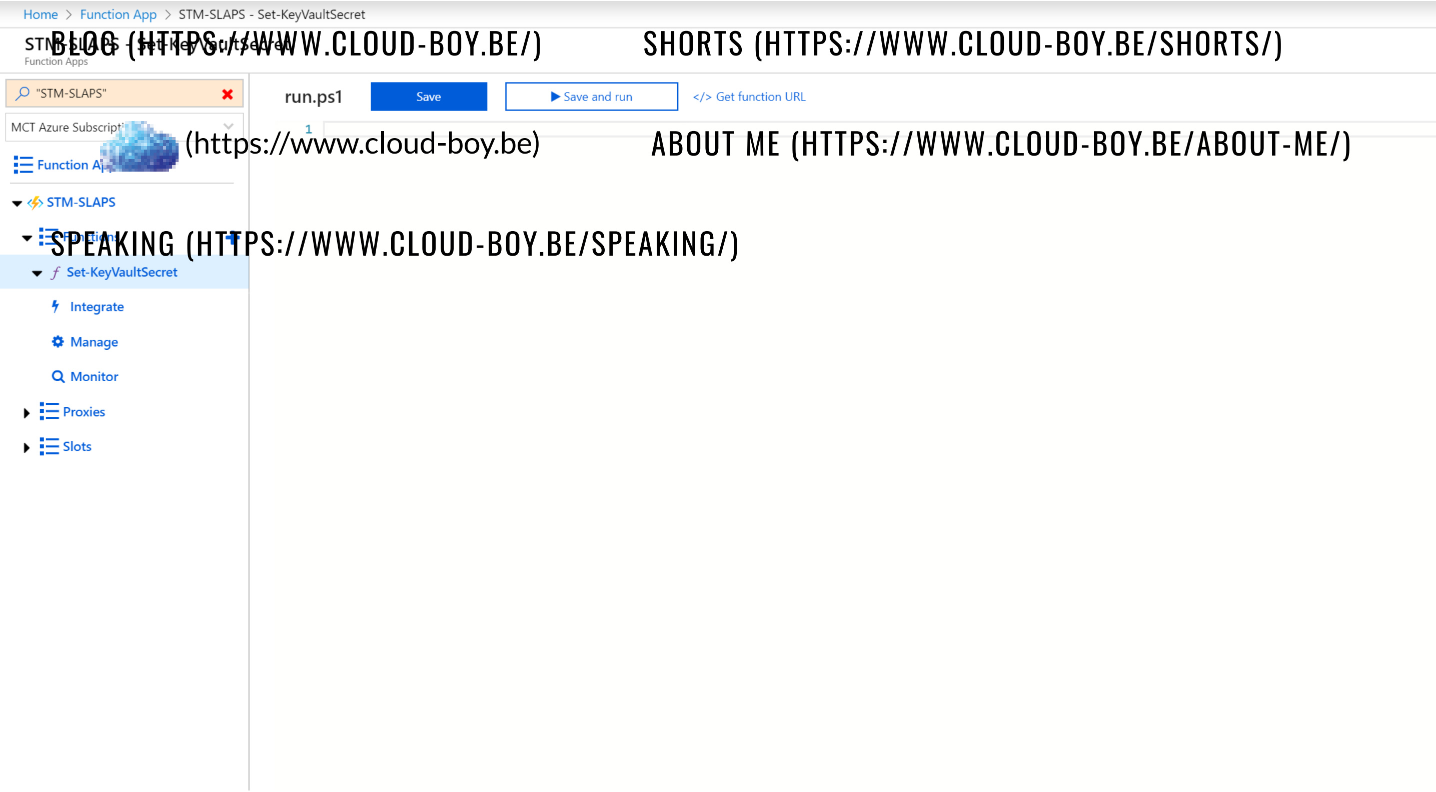


Name: Set-KeyVaultSecret

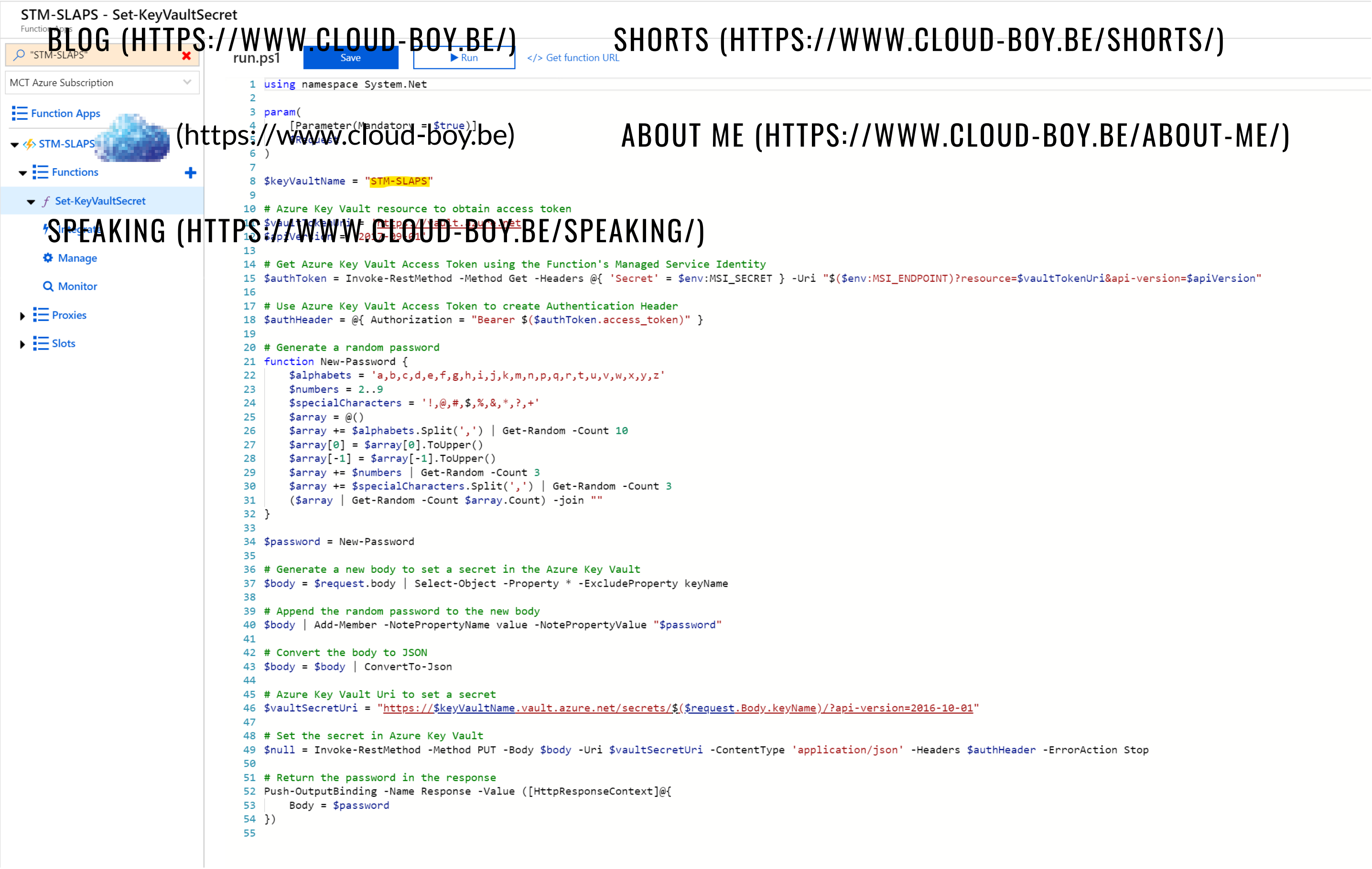
HTTP trigger

Authorization level: Function

Once deployed, click on your ‘Set-KeyVaultSecret’ function and delete all the code. You’ll function will be blank now:



Download Set-KeyVaultSecret.ps1 from https://github.com/jseerden/SLAPS (https://github.com/jseerden/SLAPS) and insert the code in your Function App. Edit the $keyVaultName variable with the name of your Key Vault. Click on ‘Save’.

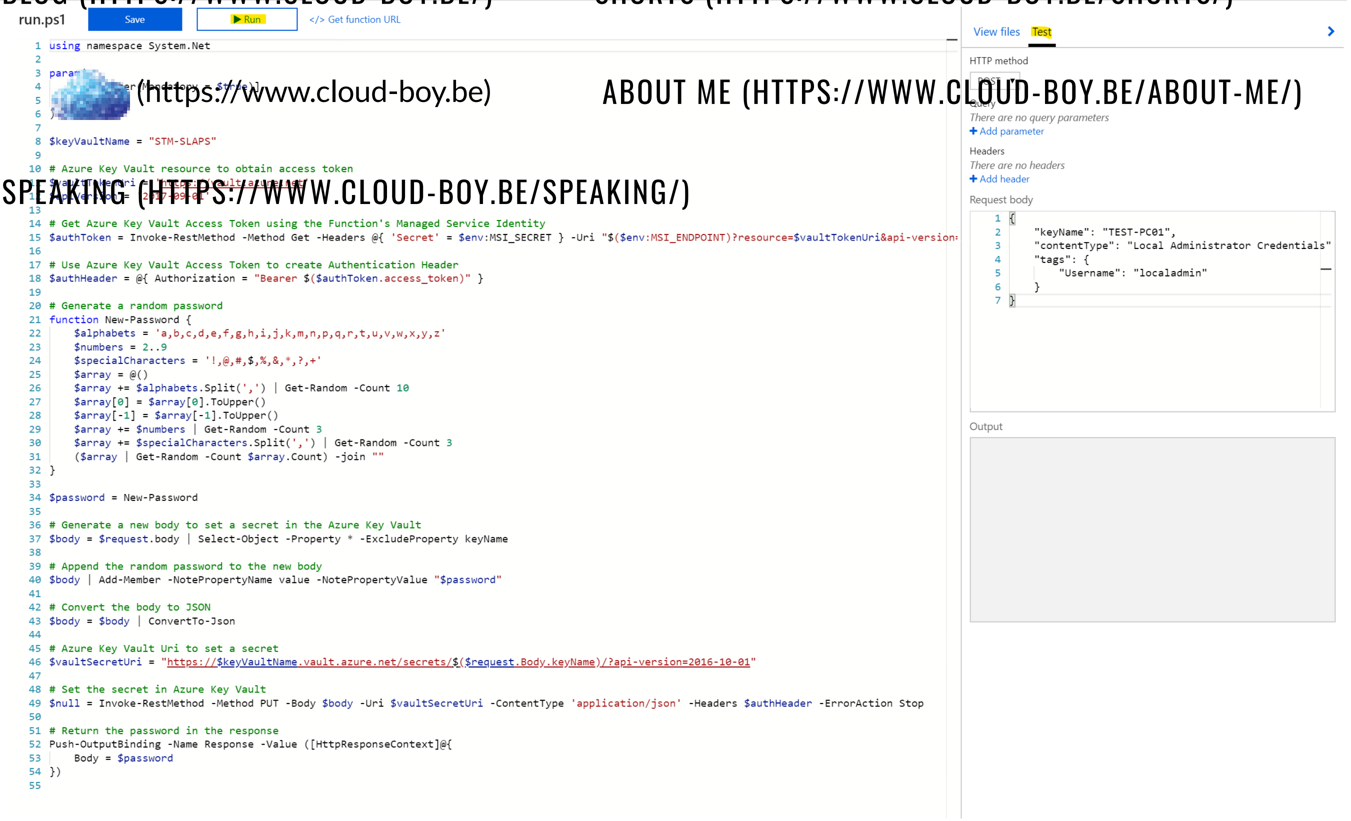


```
using namespace System.Net

param(
    [Parameter(Mandatory = $true)]
    $Request
)

$keyVaultName = "STM-SLAPS"

# Azure Key Vault resource to obtain access token
$vaultTokenUri = 'https://vault.azure.net'
$apiVersion = "2017-09-01"

# Get Azure Key Vault Access Token using the Function's Managed Service Identity
$authToken = Invoke-RestMethod -Method Get -Headers @{ 'Secret' = $env:MSI_SECRET } -Uri "$($env:MSI_ENDPOINT)?resource=$vaultTokenUri&api-version=$apiVersion"

# Use Azure Key Vault Access Token to create Authentication Header
$authHeader = @{ Authorization = "Bearer $($authToken.access_token)" }

# Generate a random password
function New-Password {
    $alphabets = 'a,b,c,d,e,f,g,h,i,j,k,m,n,p,q,r,t,u,v,w,x,y,z'
    $numbers = 2..9
    $specialCharacters = '!,@,#,$,%,&,*,?,+'
    $array = @()
    $array += $alphabets.Split(',') | Get-Random -Count 10
    $array[0] = $array[0].ToUpper()
    $array[-1] = $array[-1].ToUpper()
    $array += $numbers | Get-Random -Count 3
    $array += $specialCharacters.Split(',') | Get-Random -Count 3
    ($array | Get-Random -Count $array.Count) -join ""
}

$password = New-Password

# Generate a new body to set a secret in the Azure Key Vault
$body = $request.body | Select-Object -Property * -ExcludeProperty keyName

# Append the random password to the new body
$body | Add-Member -NotePropertyName value -NotePropertyValue "$password"

# Convert the body to JSON
$body = $body | ConvertTo-Json

# Azure Key Vault Uri to set a secret
$vaultSecretUri = "https://$keyVaultName.vault.azure.net/secrets/$($request.Body.keyName)/?api-version=2016-10-01"

# Set the secret in Azure Key Vault
$null = Invoke-RestMethod -Method PUT -Body $body -Uri $vaultSecretUri -ContentType 'application/json' -Headers $authHeader -ErrorAction Stop

# Return the password in the response
Push-OutputBinding -Name Response -Value ([HttpResponseContext]@{
    Body = $password
})

```

Now you can test the function by clicking on ‘Test’. Add the following code in the ‘Request body’ field. Click on ‘Run’.

```
{
  “keyName”: “TEST-PC01”,
  “contentType”: “Local Administrator Credentials”,
  “tags”: {
    “Username”: “localadmin”
  }
}
```

Be aware, if you copy paste from my site you have to replace the ” with new ones inside the Function App, otherwise you’ll get errors.



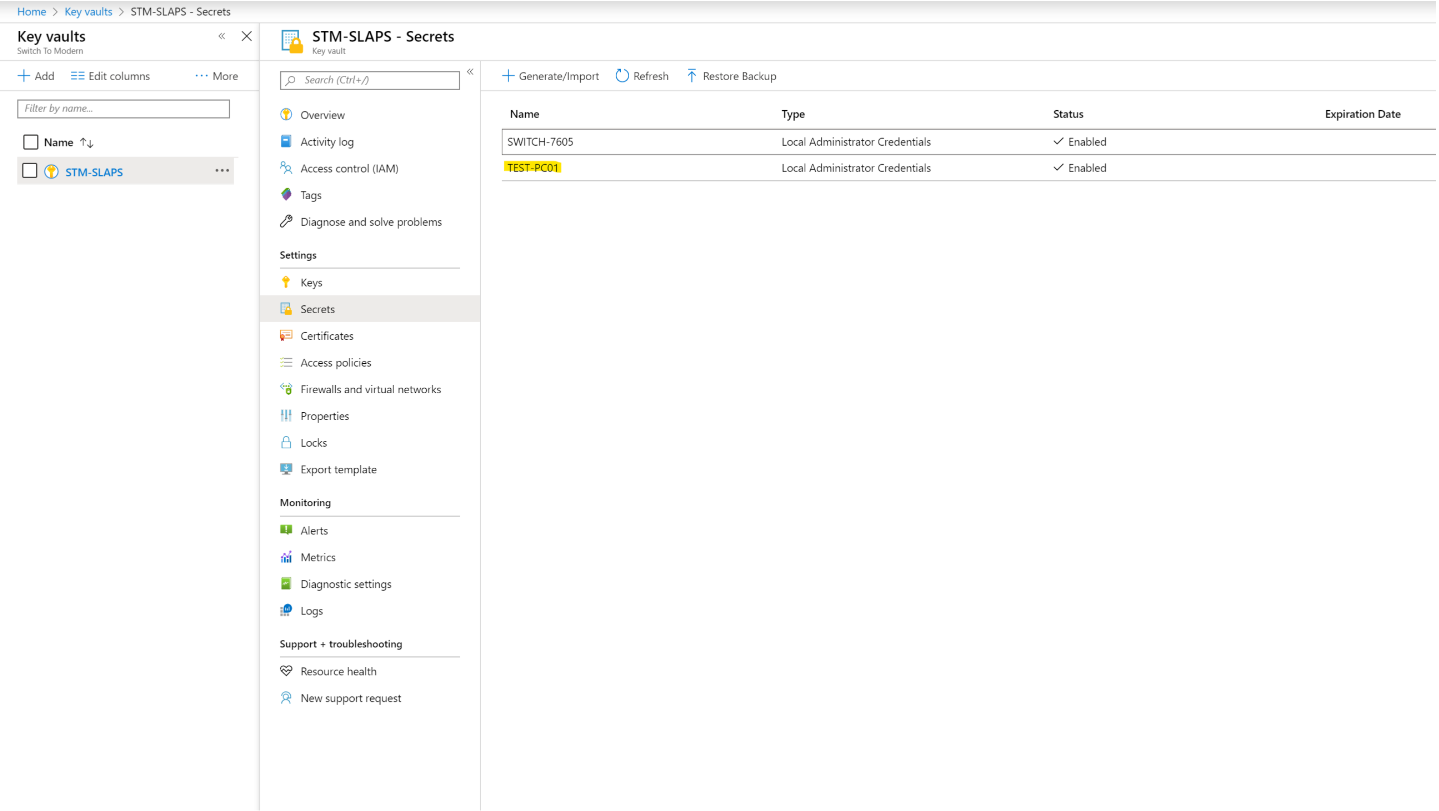
```
using namespace System.Net

param(
    [Parameter(Mandatory = $true)]
[illegible]
)

$keyVaultName = "STM-SLAPS"

# Azure Key Vault resource to obtain access token
$vaultTokenUri = 'https://vault.azure.net'
$apiVersion = '2017-09-01'

# Get Azure Key Vault Access Token using the Function's Managed Service Identity
$authToken = Invoke-RestMethod -Method Get -Headers @{ 'Secret' = $env:MSI_SECRET } -Uri "$($env:MSI_ENDPOINT)?resource=$vaultTokenUri&api-version=

# Use Azure Key Vault Access Token to create Authentication Header
$authHeader = @{ Authorization = "Bearer $($authToken.access_token)" }

# Generate a random password
function New-Password {
    $alphabets = 'a,b,c,d,e,f,g,h,i,j,k,m,n,p,q,r,t,u,v,w,x,y,z'
    $numbers = 2..9
    $specialCharacters = '!,@,#,$,%,&,*,?,+'
    $array = @()
    $array += $alphabets.Split(',') | Get-Random -Count 10
    $array[0] = $array[0].ToUpper()
    $array[-1] = $array[-1].ToUpper()
    $array += $numbers | Get-Random -Count 3
    $array += $specialCharacters.Split(',') | Get-Random -Count 3
    ($array | Get-Random -Count $array.Count) -join ""
}

$password = New-Password

# Generate a new body to set a secret in the Azure Key Vault
$body = $request.body | Select-Object -Property * -ExcludeProperty keyName

# Append the random password to the new body
$body | Add-Member -NotePropertyName value -NotePropertyValue "$password"

# Convert the body to JSON
$body = $body | ConvertTo-Json

# Azure Key Vault Uri to set a secret
$vaultSecretUri = "https://$keyVaultName.vault.azure.net/secrets/$($request.Body.keyName)/?api-version=2016-10-01"

# Set the secret in Azure Key Vault
$null = Invoke-RestMethod -Method PUT -Body $body -Uri $vaultSecretUri -ContentType 'application/json' -Headers $authHeader -ErrorAction Stop

# Return the password in the response
Push-OutputBinding -Name Response -Value ([HttpResponseContext]@{
    Body = $password
})
```

HTTP method
POST
Query
There are no query parameters
+ Add parameter
Headers
There are no headers
+ Add header
Request body

```
{
    "keyName": "TEST-PC01",
    "contentType": "Local Administrator Credentials",
    "tags": {
        "Username": "localadmin"
    }
}
```

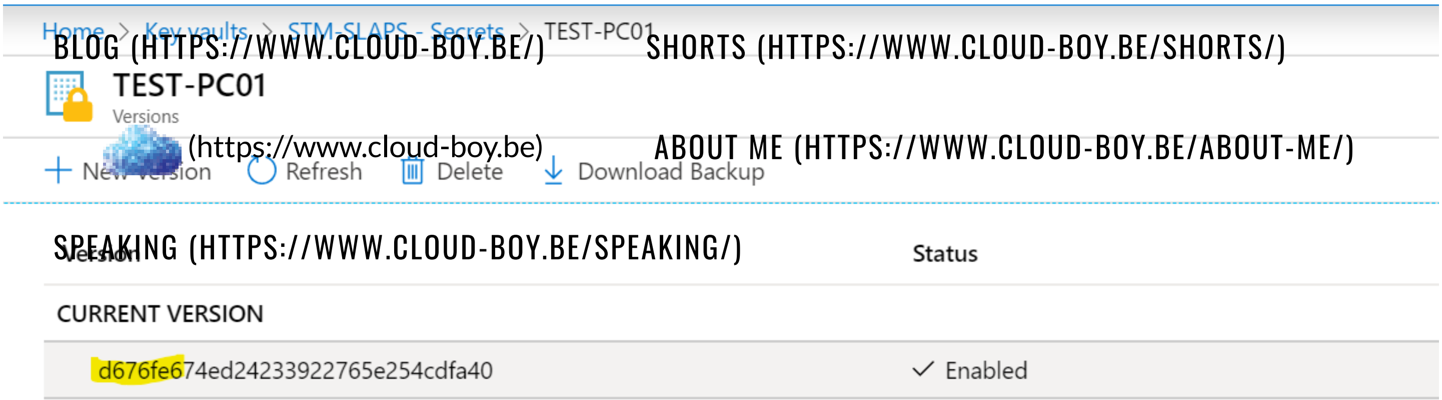
Output

Navigate to your Key vault and check if the local admin credentials are stored there. Click on TEST-PC01.



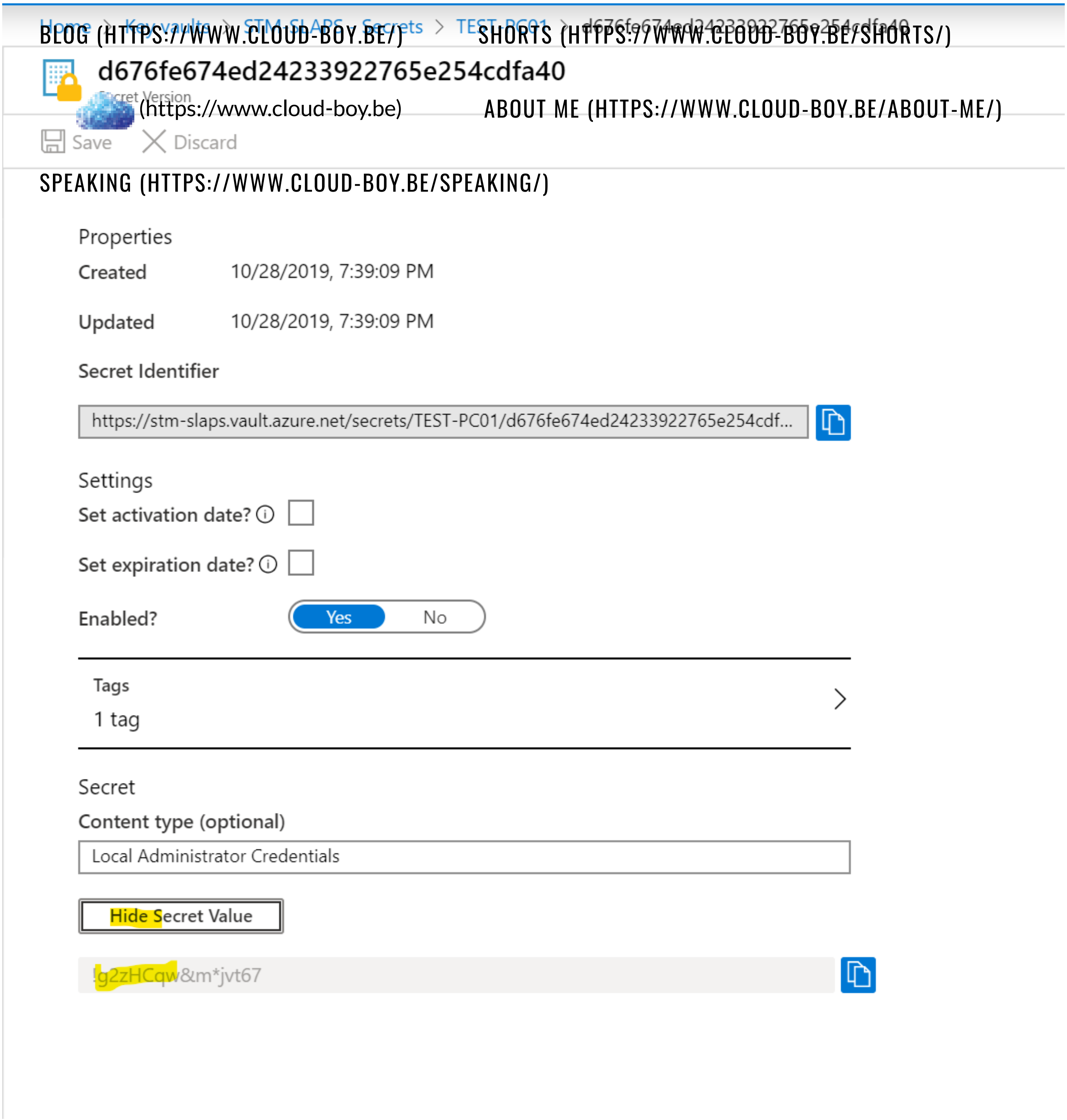
Name	Type	Status	Expiration Date
SWITCH-7605	Local Administrator Credentials	✓ Enabled	
TEST-PC01	Local Administrator Credentials	✓ Enabled	

Click on the current version of TEST-PC01



Version	Status
CURRENT VERSION	
d676fe674ed24233922765e254cdfa40	✓ Enabled

Click on ‘Show secret value’, your local admin password is stored there:


Home > Key vaults > STM-SLAPS - Secrets > TEST-PC01 > d676fe674ed24233922765e254cdfa40
Save
Discard

d676fe674ed24233922765e254cdfa40

Secret Version

Properties

Created 10/28/2019, 7:39:09 PM

Updated 10/28/2019, 7:39:09 PM

Secret Identifier

https://stm-slaps.vault.azure.net/secrets/TEST-PC01/d676fe674ed24233922765e254cdf...

Settings

Set activation date? ⓘ □

Set expiration date? ⓘ □

Enabled? [✓ Yes] No

Tags

1 tag

Secret

Content type (optional)

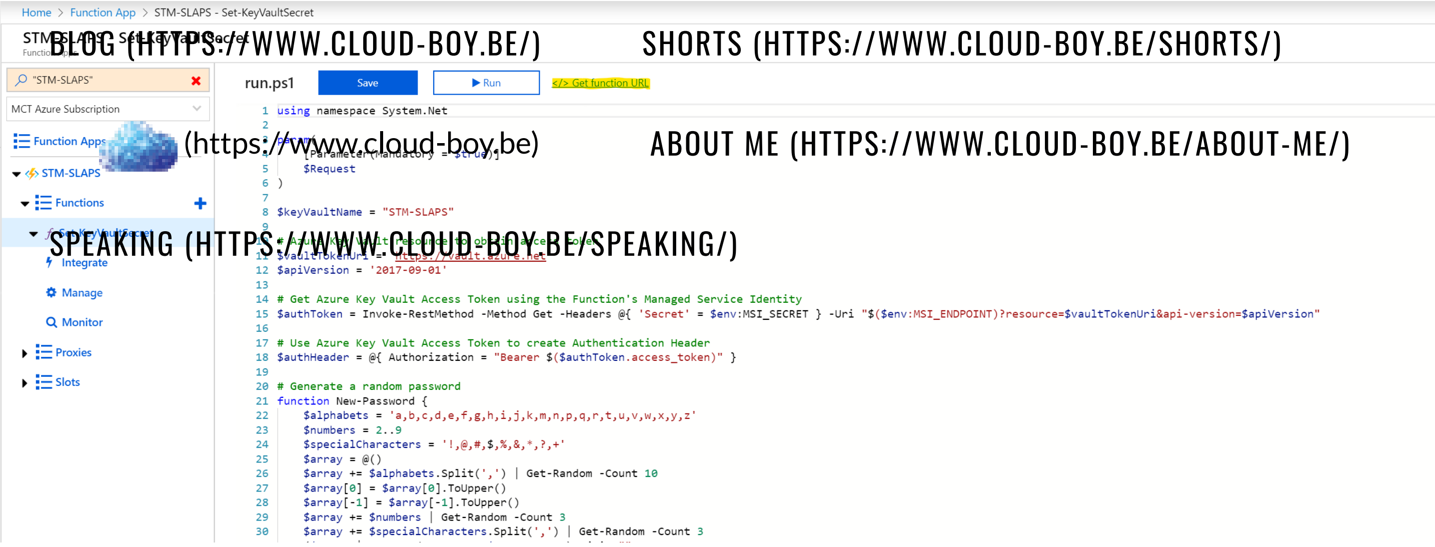
Local Administrator Credentials

Hide Secret Value

!g2zHCqw&m*jvt67

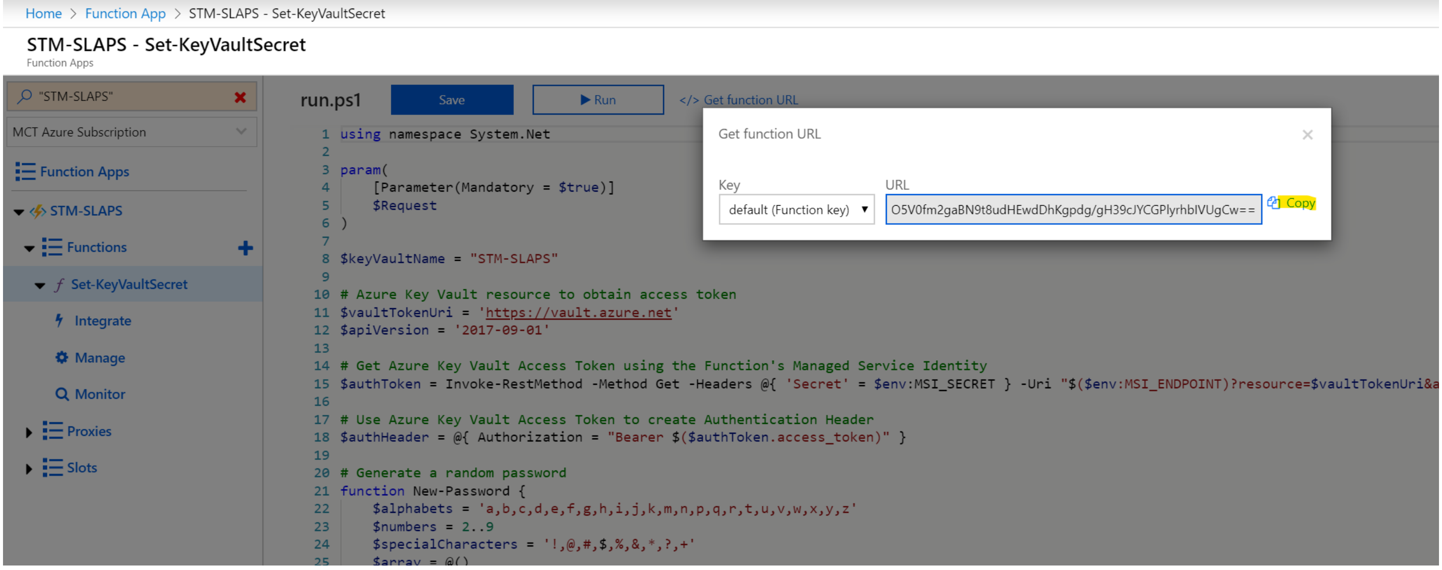
4. DEPLOY THE POWERSHELL SCRIPT WITH INTUNE

First we need the Function App URL. Navigate to your SetKeyVaultSecret Function App. Click on ‘Get Function URL’.



```
using namespace System.Net

param(
    [Parameter(Mandatory = $true)]
    $Request
)

$keyVaultName = "STM-SLAPS"

# Azure Key Vault resource to obtain access token
$vaultTokenUri = 'https://vault.azure.net'
$apiVersion = '2017-09-01'

# Get Azure Key Vault Access Token using the Function's Managed Service Identity
$authToken = Invoke-RestMethod -Method Get -Headers @{ 'Secret' = $env:MSI_SECRET } -Uri "$($env:MSI_ENDPOINT)?resource=$vaultTokenUri&api-version=$apiVersion"

# Use Azure Key Vault Access Token to create Authentication Header
$authHeader = @{ Authorization = "Bearer $($authToken.access_token)" }

# Generate a random password
function New-Password {
    $alphabets = 'a,b,c,d,e,f,g,h,i,j,k,m,n,p,q,r,t,u,v,w,x,y,z'
    $numbers = 2..9
    $specialCharacters = '!,@,#,$,%,&,*,?,+'
    $array = @()
    $array += $alphabets.Split(',') | Get-Random -Count 10
    $array[0] = $array[0].ToUpper()
    $array[-1] = $array[-1].ToUpper()
    $array += $numbers | Get-Random -Count 3
    $array += $specialCharacters.Split(',') | Get-Random -Count 3
```

Click on ‘Copy’.



Key: default (Function key)

URL: O5V0fm2gaBN9t8udHEwdDhKgpdg/gH39cJYCGPlyrhblVUgCw==

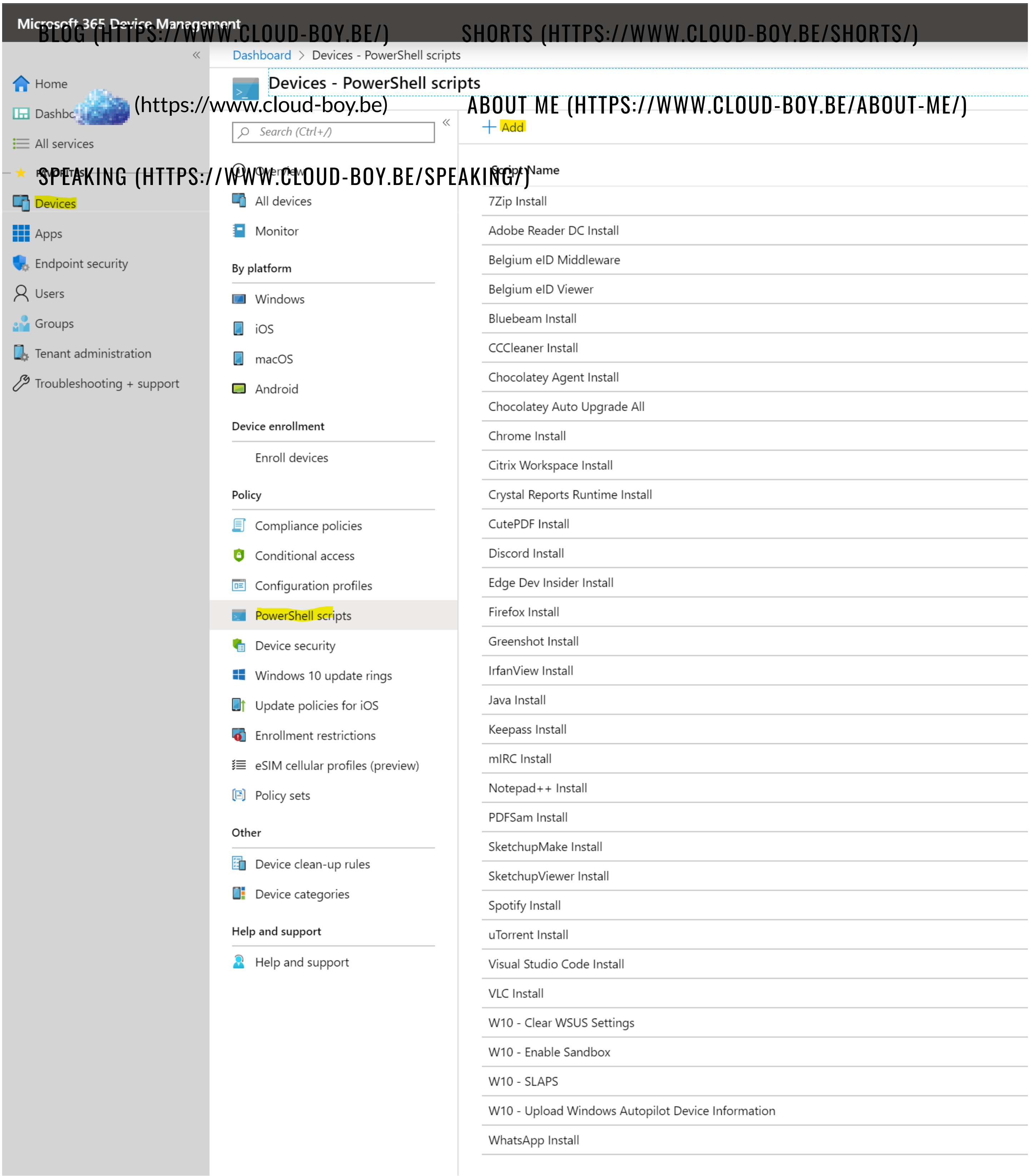
```
using namespace System.Net

param(
    [Parameter(Mandatory = $true)]
    $Request
)

$keyVaultName = "STM-SLAPS"

# Azure Key Vault resource to obtain access token
$vaultTokenUri = 'https://vault.azure.net'
$apiVersion = '2017-09-01'

# Get Azure Key Vault Access Token using the Function's Managed Service Identity
$authToken = Invoke-RestMethod -Method Get -Headers @{ 'Secret' = $env:MSI_SECRET } -Uri "$($env:MSI_ENDPOINT)?resource=$vaultTokenUri&a

# Use Azure Key Vault Access Token to create Authentication Header
$authHeader = @{ Authorization = "Bearer $($authToken.access_token)" }

# Generate a random password
function New-Password {
    $alphabets = 'a,b,c,d,e,f,g,h,i,j,k,m,n,p,q,r,t,u,v,w,x,y,z'
    $numbers = 2..9
    $specialCharacters = '!,@,#,$,%,&,*,?,+'
    $array = @()
```

Download New-LocalAdmin.ps1 from https://github.com/jseerden/SLAPS (https://github.com/jseerden/SLAPS) and edit the following variables:

$uri = ‘PASTE URL HERE’

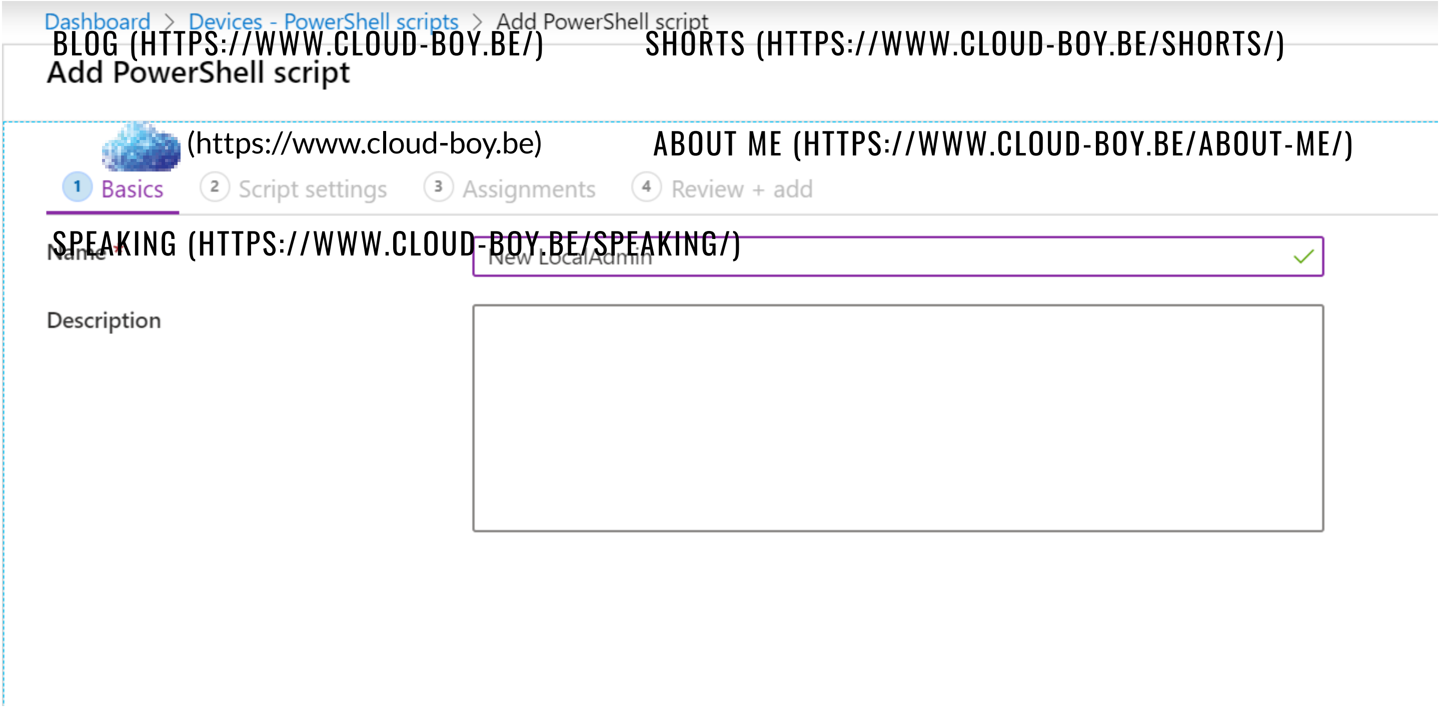
Save the .ps1 file.

Navigate to the Intune dashboard (https://devicemanagement.microsoft.com/ (https://devicemanagement.microsoft.com/)). Go to Devices – PowerShell scripts. Click on ‘Add’ to upload our New-LocalAdmin.ps1 script.



Script Name
7Zip Install
Adobe Reader DC Install
Belgium eID Middleware
Belgium eID Viewer
Bluebeam Install
CCCleaner Install
Chocolatey Agent Install
Chocolatey Auto Upgrade All
Chrome Install
Citrix Workspace Install
Crystal Reports Runtime Install
CutePDF Install
Discord Install
Edge Dev Insider Install
Firefox Install
Greenshot Install
IrfanView Install
Java Install
Keepass Install
mIRC Install
Notepad++ Install
PDFSam Install
SketchupMake Install
SketchupViewer Install
Spotify Install
uTorrent Install
Visual Studio Code Install
VLC Install
W10 - Clear WSUS Settings
W10 - Enable Sandbox
W10 - SLAPS
W10 - Upload Windows Autopilot Device Information
WhatsApp Install

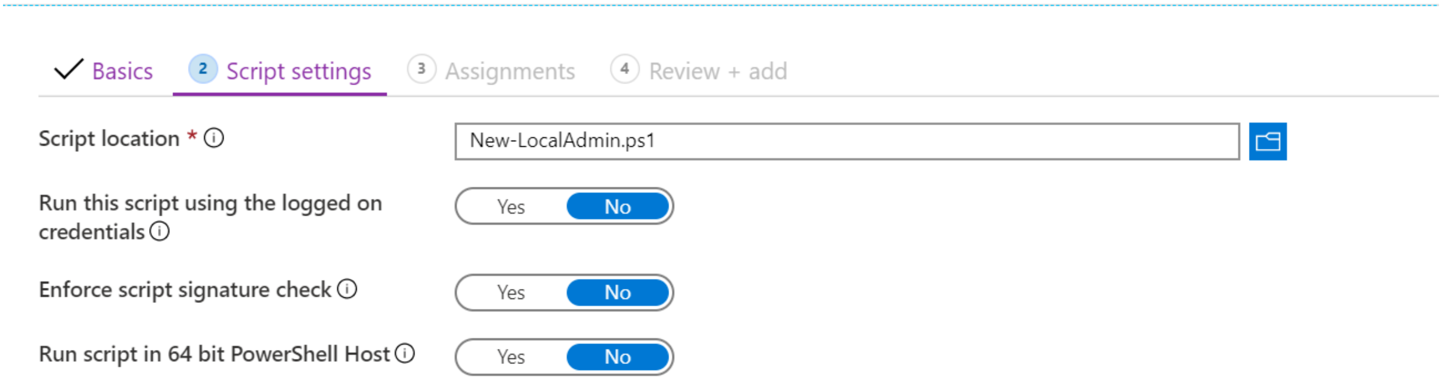
Name your script and click on ‘Next’.



Name * New LocalAdmin

Description

Choose your modified .ps1 script and leave the 3 settings on 'No'.



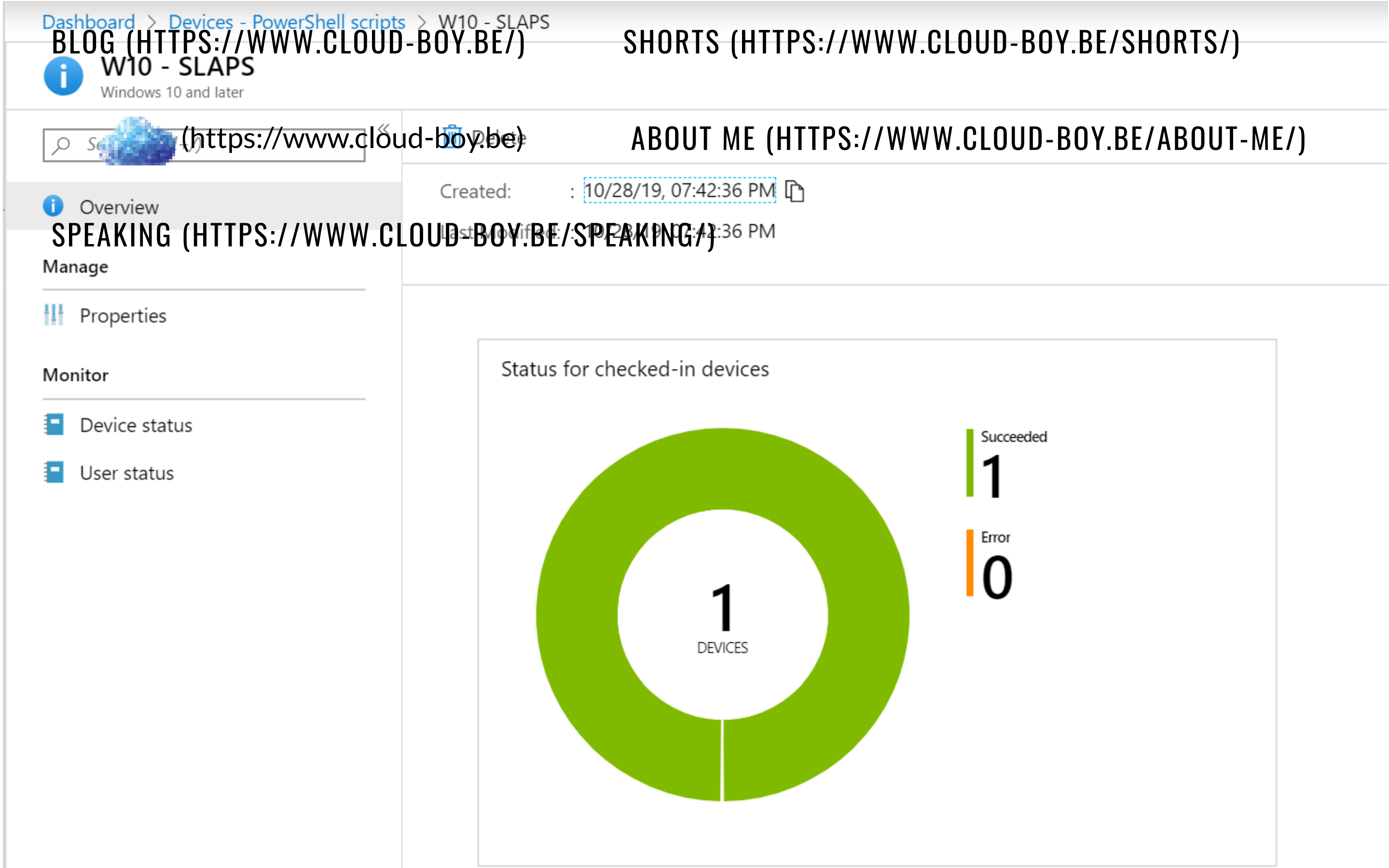
Script location * ⓘ New-LocalAdmin.ps1

Run this script using the logged on credentials ⓘ Yes [✓ No]

Enforce script signature check ⓘ Yes [✓ No]

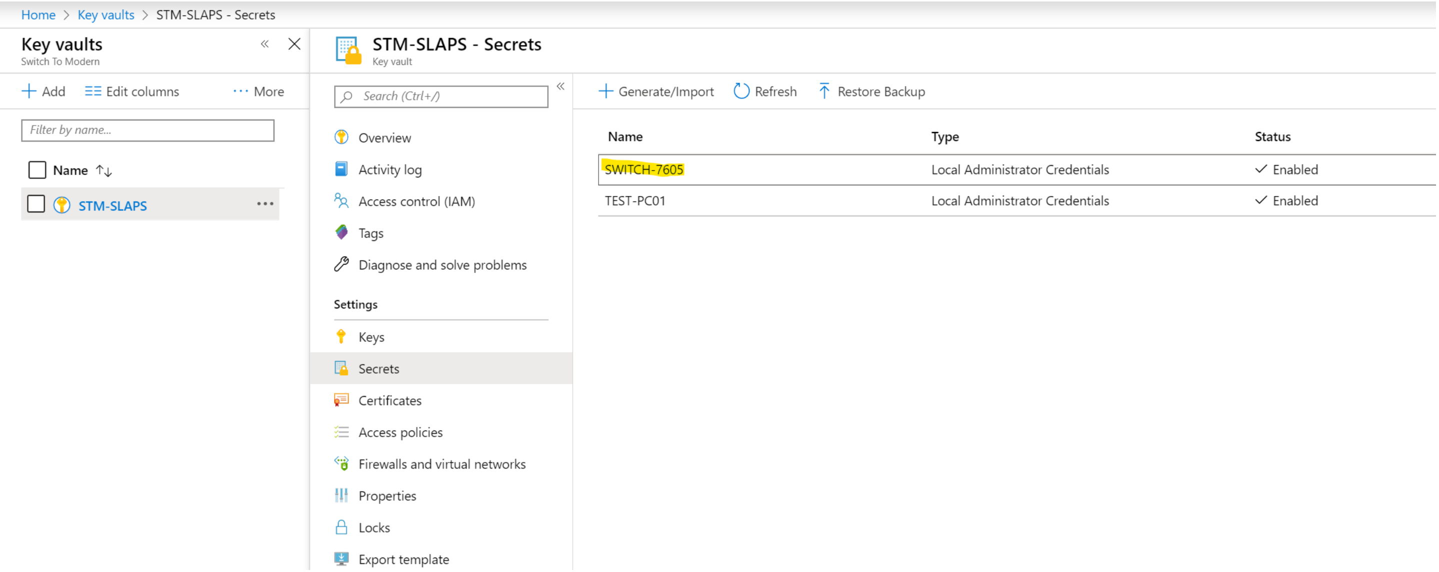
Run script in 64 bit PowerShell Host ⓘ Yes [✓ No]

Deploy it to your testgroup. And follow up. You should see that the script got deployed successfully to your target device.



Created: 10/28/19, 07:42:36 PM

And if you check again in your Azure Key Vault, the local admin password of your device should be there too:



Name	Type	Status
SWITCH-7605	Local Administrator Credentials	✓ Enabled
TEST-PC01	Local Administrator Credentials	✓ Enabled

Happy testing!

More articles on Intune:

- device hashes from HP for Autopilot pre-production testing (https://www.cloud-boy.be/blog/get-device-hashes-from-hp-for-autopilot-pre-production-testing/)
- Run as admin gives black screen in Quick Assist/TeamViewer – Intune fix (https://www.cloud-boy.be/blog/run-as-admin-gives-black-screen-in-quick-assist-teamviewer-intune-fix/)
- Intune – change Primary User of a device (https://www.cloud-boy.be/blog/intune-change-primary-user-of-a-device/)
- Ransomware protection (Controlled Folder Access) setup with Intune (https://www.cloud-boy.be/blog/ransomware-protection-controlled-folder-access-setup-with-microsoft-endpoint-manager/)
- Windows Hello for Business multi-factor unlock with Intune (https://www.cloud-boy.be/blog/windows-hello-for-business-multi-factor-unlock-with-mem-intune/)

Linkedin (https://www.linkedin.com/shareArticle?trk=Serverless+LAPS+with+Intune%2C+Function+App+and+Key+Vault&url=https%3A%2F%2Fwww.cloud-boy.be%2Fblog%2Fserverless-laps-with-intune-function-app-and-key-vault%2F)

Share (https://www.facebook.com/sharer.php?u=https%3A%2F%2Fwww.cloud-boy.be%2Fblog%2Fserverless-laps-with-intune-function-app-and-key-vault%2F)

Tweet (https://twitter.com/intent/tweet?text=Serverless%20LAPS%20with%20Intune%2C%20Function%20App%20and%20Key%20Vault&url=https
boy.be/blog/serverless-laps-with-intune-function-app-and-key-vault/&via=_Cloud_boy)

Whatsapp (https://api.whatsapp.com/send?text=Serverless%20LAPS%20with%20Intune%2C%20Function%20App%20and%20Key%20Vault%20https%
boy.be%2Fblog%2Fserverless-laps-with-intune-function-app-and-key-vault%2F)

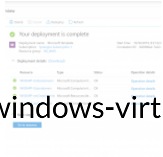
Mail (mailto:?subject=%20&body=%20https%3A%2F%2Fwww.cloud-boy.be%2Fblog%2Fserverless-laps-with-intune-function-app-and-key-vault%2F)

PREVIOUS POST

Silently remove SCCM client and enroll device in Intune (https://www.cloud-boy.be/blog/silently-remove-sccm-client-and-enroll-device-in-intune/)

NEXT POST

WVD – The template deployment not valid (https://www.cloud-boy.be/blog/windows-virtual-dekstop-the-template-deployment-is-not-valid/)



(https://www.cloud-boy.be/)