Timeline

Week 1 :

As I had little exposure to PowerShell and none with Azure prior to this assessment I had to spend some time to get more familiar with these terms. During that week I have spent approximately 10 hours, about 1 hour per day until Friday and the rest in the weekend. I saw some tutorials from YouTube, from linda.com and also read the related Microsoft pages .I also got a bit familiar with the online azure management portal.

Week2: I started experimenting with simple loops and user input, I downloaded azure module and tried the basic commands for the management of resources and accounts.

I studied templates from the github and I found from Microsoft how to create the policy template.

I started the implementation of the script and of the templates and of course I did about 100 tests with deployments at every stage. Unfortunately although in theory I had more time due to the public holiday on Monday my current situation is such that I couldn’t find and put in more time than the 1st week so I have used about 10-12 hours broken throughout the week.

So in total for this assessment I spent 20-22 hours with half of them going towards gaining awareness on the topic and search my interest within it.

Now I feel much more comfortable already and I no longer find this assessment as challenging.

If I was to work with you and thus to be able to work during my most productive hours it would take me much lesser time and I would find ways also to implement this better .

The deployment Script

My intention with this script was to make it very simple and to follow the best practices. As I am new to PowerShell and PowerShell for azure I followed Microsoft’s suggestions so as to make sure that it follows the best practices. I had lots of side ideas but I decided not to reinvent the wheel at this point and stick closer to the assessment tasks.

This script asks from the user some necessary input and stores it into the parameters’ section to be then used during execution. It first calls for subscription id, then for the resource group name to be used for the resource group creation, resource group location and deployment name. I also set the path to the template.json (main template), the parameters.json and for the policy template.

For the scope of this assessment the resource group location is predefined to be “west Europe” so there was no need to request it as input but I thought it would be better to allow for more flexibility.

I used 3 functions, 1 to register the resource providers (network, storage), one function to create the tags (I assigned them statically but having a function makes it easier to copy it and modify it in the future) and lastly a function to register the policy template as a definition and assign it to our resource group.

During the execution we sign in to azure and then I select the specified subscription id. Then a small loop registers the resource group providers that I specify in the resourcegroupproviders array one after the other. For this assessment the providers are Microsoft.network, Microsoft.storage but of course we can add more to be registered if needed.

Following that I check if the resource group exists, in which case I proceed to add the resources to that, or if it does not exist, the loop continues to create it.

At this point we have a resource group so I call the function to add the tags and then the deployment of the template and parameters starts.

Lastly I call the function to register the policy template and assign it to the resource group. I do that at the end because it would block the deployment if I had done that before.

Resources and json files

Storage account

The storage account uses service encryption with Microsoft-managed encryption keys as this would be the simplest for the customers. Nevertheless it is very easy to update with their own keys if they choose so at a later stage. If the customer wants to manage his own keys (need changing every 90 days) then Microsoft provides the PowerShell commands to implement that. In brief a storage account identity must be created and assigned to the storage account. Then you need to enable the two key protection features, Soft Delete and Do Not Purge. Then one can specify his keys. Another option is to create a key vault and put the keys there and then associate that key to the storage identity.

I made the assumption that the service encryption with Microsoft managed keys is the simplest way for a non IT oriented customer to be secure.

The storage account name needs to be unique and in low case letters. To accomplish that I introduced the NamePrefix parameter and then the used “unique string” which takes as parameters the subscription id and the resource group id. This way the unique string hash has more chances to be globally unique. I concat the name prefix with the unique hash and all of them into low case via the “tolower”.

The prefix can also be used in future implementations so as to provide for more resource names.

Virtual network and subnets:

Since the subnets are requested to be 3 they cannot be equally divided (or else I would lose lots of address space by an unused 4th subnet). To implement that I have first used one bit to make two subnets and then divided one of the two subnets in two again resulting in total in 3 subnets. 2 have a subnet of /14 and a larger one of /13.

Policy

The policy template is designed in order for this specific resource account to be able to access Compute, Network and Storage. To do that I deny access to anything that is not “in” the array I provide. I think it is easier to alter the policy by using the “Allowed resource types “approach and specify the array of resource types that are permitted, denying all other activities.