Home Assignment Azure Task - Runbook | Dani Kogel

Intro:
I would like to explain the reasoning behind some of my choices and approaches when solving the following assignment:

1. *Use the following link for an Azure subscription you can work with in (You should also have an invite link to that Azure subscription in your email Inbox, notice it might be landed in your junk mail folder): Azure*
2. *Create an ARM (Azure Resource Manager) template for storage account and create 2 storage accounts*
3. *Create an ARM template for a Windows or Linux server*
4. *Create a tool to deploy the Storage accounts and the server in a continuous manner. Any type of CD (Continues deployment) pipe can be used here, preferred is an Azure DevOps pipe (In order to bypass permissions, set your personal computer as Azure DevOps agent)*
5. *Write a script that will create, upload, and copy 100 blobs from Storage account A to B, execute it on the server you created earlier using CD pipeline.*
6. *Choose metrics to monitor that small system of 1 server and 2 storage accounts and create dashboard via "Monitor" to show that*

In General, I would want to make this process as much as “Start&Go” as possible, trying to create pipelines for most scripts in order to make it easier to automate, while also taking the possibility of scaling up into account.

Although I’m not very familiar with the best practices and approaches to many of the things the task requires, I’ve relied on various documentations, google and AI assistants.

Task 2 - Creating an arm template for storage accounts

I’ve used an example of a template and adapted it in order to create 2 storage accounts, while trying to avoid hard coding things like names and resource groups, I’ve used the Type of Standard LRS as it’s the simplest option of the all and since it’s a training task I couldn’t find a reason to use anything else. Redundancy isn't important and neither is cost at this case.

I’ve added a random string to the name so if we decide to deploy more than 2 storage accounts it wouldn't require much fiddling with values.

Also added those names as output variables as a best practice, since those names are semi random, it could be useful in certain scenarios.

Task 3 - Creating an arm template for vm(linux machine)

Now we’ll continue with another template for the vm, I’ve chosen to go with linux here, as it’s a lightweight option and in general i’m more familiar with it in CD scenarios. As cost not really a factor in this scale, I went with a slightly better machine of Standard_DS1_v2, as I figured it could allow me faster work with the creation of blobs. (many of the things were guesses at first

and were changed as I went along and familiarized myself more with azure). I've also decided to test out working with a parameter file.

Task 4 - Writing a deployment pipeline using Azure Devops

I wasn't too familiar with the deployment side of pipelines as previously I mostly used them for developing code and never really got the chance to experience scripting them and deploying things without automated processes that were already in place. At first I attempted to connect to the supplied subscription using a self hosted agent, but I've encountered an issue with Entra permissions that basically blocked me from using Devops on the supplied subscription so I went with my own.

The pipeline I've created is simple thanks to use of Devops, in which the Vm is implemented in a rg of a configured name, and then the storage accounts are created on it.

I couldn't think of a good reason to auto Trigger this pipeline unless there are some code changes in the Main repo in which a it would be a good approach to redeploy incase any setting were changed.

Task 5 - Creating and copying Blobs

From the start this task was vague and had many different approaches. I thought that I would need to use the VM I implemented in order to execute this script, since it was sharing a resource group I figured I can just write a script that SSH into the VM and deploys it. But I quickly found a better option, As my Local Agent already authenticated and connected,
I can just use him to to run scripts.

I created a pipeline, as I wanted a reliable way to do this task, In my head this task would be some sort of an archival system in a real scenario, where one storage account holds the data and the second one copies it periodically.

I've added tags to the storage accounts, as it would make it way easier to find them, and added a project specific tag aswell, just incase.

I've ended up using a ps1 script that was translated from java code i've done using various AI tools.

There are 2 scripts, which I divided into 2 jobs in the pipeline. One is to create the blobs and one is to copy them.

I've figured out 100 is a very random number and could easily be any other value so I wanted it to be customizable in the pipeline.

Another thing I wanted is to batch down the task, instead of waiting on a task for 100 creates why not do it 10 at a time in a async jobs? I've read a bit about the options of running a background jobs using power powershell and ended up using `Start-Job -ScriptBlock`
I will admit the code isnt of the best quality as I've needed to constantly try and translate a language im more proficient in to PS. I've also added checks incase the file already exists in order to try and avoid unnecessary operations.

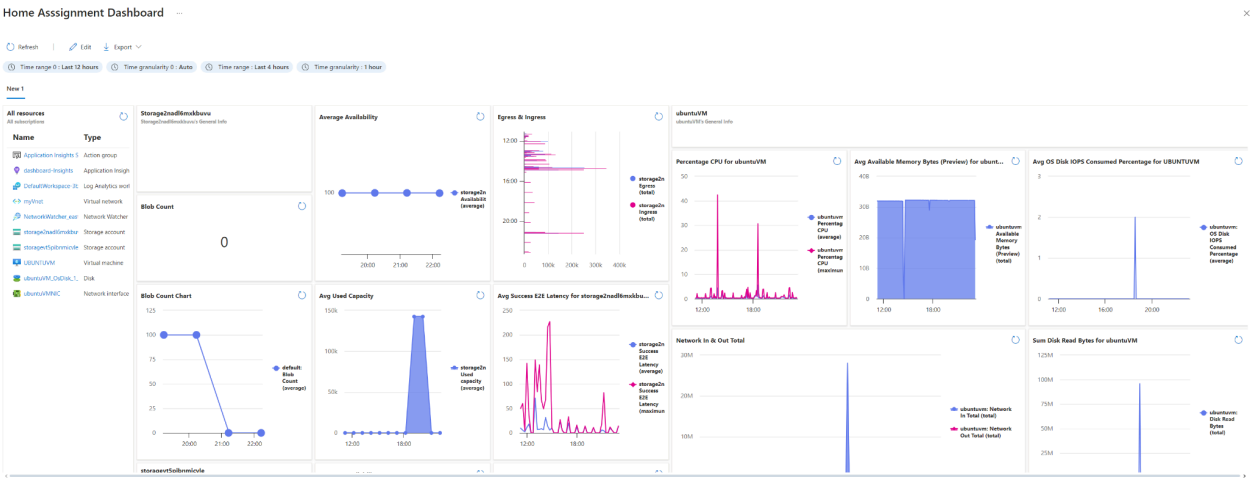
The create script workflow is -> Get the storage account names based on tags -> grab their keys -> create containers if they don't exist -> run the create job X times based on the batch size and the total amount, while checking if the blob already exists, if it does skip the creation(this is assuming the exists command is cheaper than the create blob one)

Copy one is working in a similar manner, there is some code duplication but I'm not quite sure what would be the best approach to store many of those variables so they can be used globally across many pipelines.

The pipeline itself is written as in 2 stages in order to allow them to run one after another while still giving the run them separately if the need ever arises

Task 6 - creating a dashboard

I've decided to track a few Health metrics, Blobs count and some metrics that were seem to be recommended as a good baseline.



I figured I'll group the metrics based on the resource they are part of.

Things like capacity, blob count and availability for storage accounts seem to make the most sense to track as they are changing and might indicate issues, while cpu and memory are things I would want to track on a vm to ensure it's not overloaded. Together with disk writes and and networks to ensure everything is working when it comes to sending requests to create blobs and actually creating them.