Jeff Keir (jeff.keir@mckesson.com)

Azure Storage Service Encryption

Contents

[Final Project Summary 2](#_Toc506056385)

[Problem Practicum. 3](#_Toc506056386)

[Conclusion: 13](#_Toc506056387)

# Final Project Summary

Azure Storage Service Encryption

**Problem Statement:**

According to breachlevelindex.com, as of 10-Feb 2018 over 9.2 billion data records have been lost or stolen since 2013 - a frequency today of 57 records per second[[1]](#footnote-1). Only 4% of those records were encrypted. Microsoft Azure demonstrates commitment to safeguarding all data by offering a multitude of data protection services. Azure Storage Service Encryption is one such solution. This problem set seeks to unpack how storage service encryption is enabled, how to verify that it is enabled, and demonstrate that data passing through storage service encryption interacts seamlessly with practical applications. In a publication from August 2017 Microsoft implied that blob, file, queue and table services would all be encrypted by SSE[[2]](#footnote-2).

**Overview of the Technology:**

Azure Storage Service Encryption is a server-side “toggle” impacting the suite of existing services that leverage Azure storage. Once activated, all subsequent storage devices instantiated within the service domain are encrypted. Data written prior to activating the toggle are encrypted after a new read/write operation.

**High Level Steps:**

1. Instantiate a new storage service and visualize the toggle
2. Query the storage devices to visualize the encryption state
3. Toggle the encryption state
4. Inspect data within a storage device
5. Re-create the messaging service used in Azure Deep Dive homework 8 within a secure storage service and visualize the seamless interoperation

**Code Source:**

<https://github.com/blumu/azure-content/blob/master/articles/event-hubs/event-hubs-archive-python.md>

**Hardware Used:**

Windows 7 64b 16Gb RAM HP Zbook laptop

**Software Used:**

Azure Cloud Shell (Azure CLI 2.0 (bash))

Python 2.7.5 (<https://www.python.org/downloads/>)

VS Code 1.18.1 (<https://code.visualstudio.com/download>)

**YouTube Links:**

2 Min: <https://youtu.be/f_FUk-OmsGE>

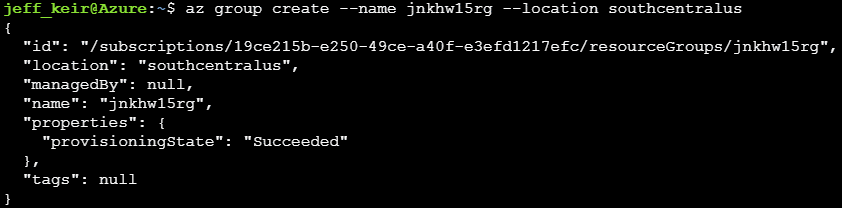
15 Min: <https://youtu.be/l2DjdfS3t9g>

# Problem Practicum.

Create new storage and visualize the encryption settings of the Azure Storage Service. Examine the default properties of blob, file, table, and queue storage types. Test the assertions made by Microsoft regarding the property setting, and the services protected by the property. Utilize the storage service in a programmatic application and visualize the encryption states throughout the operation(s).

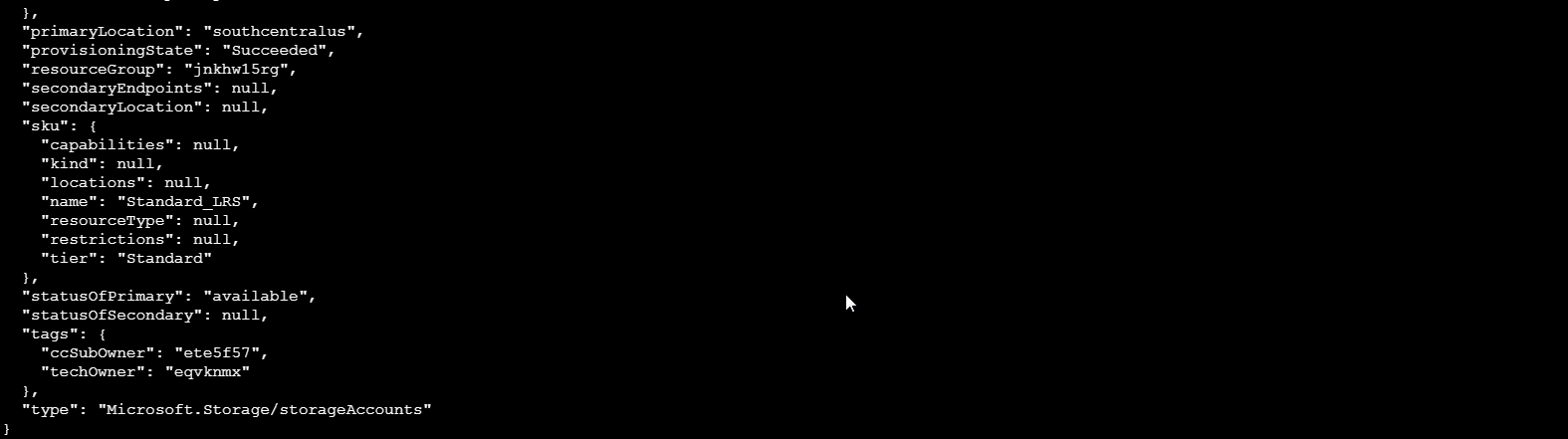
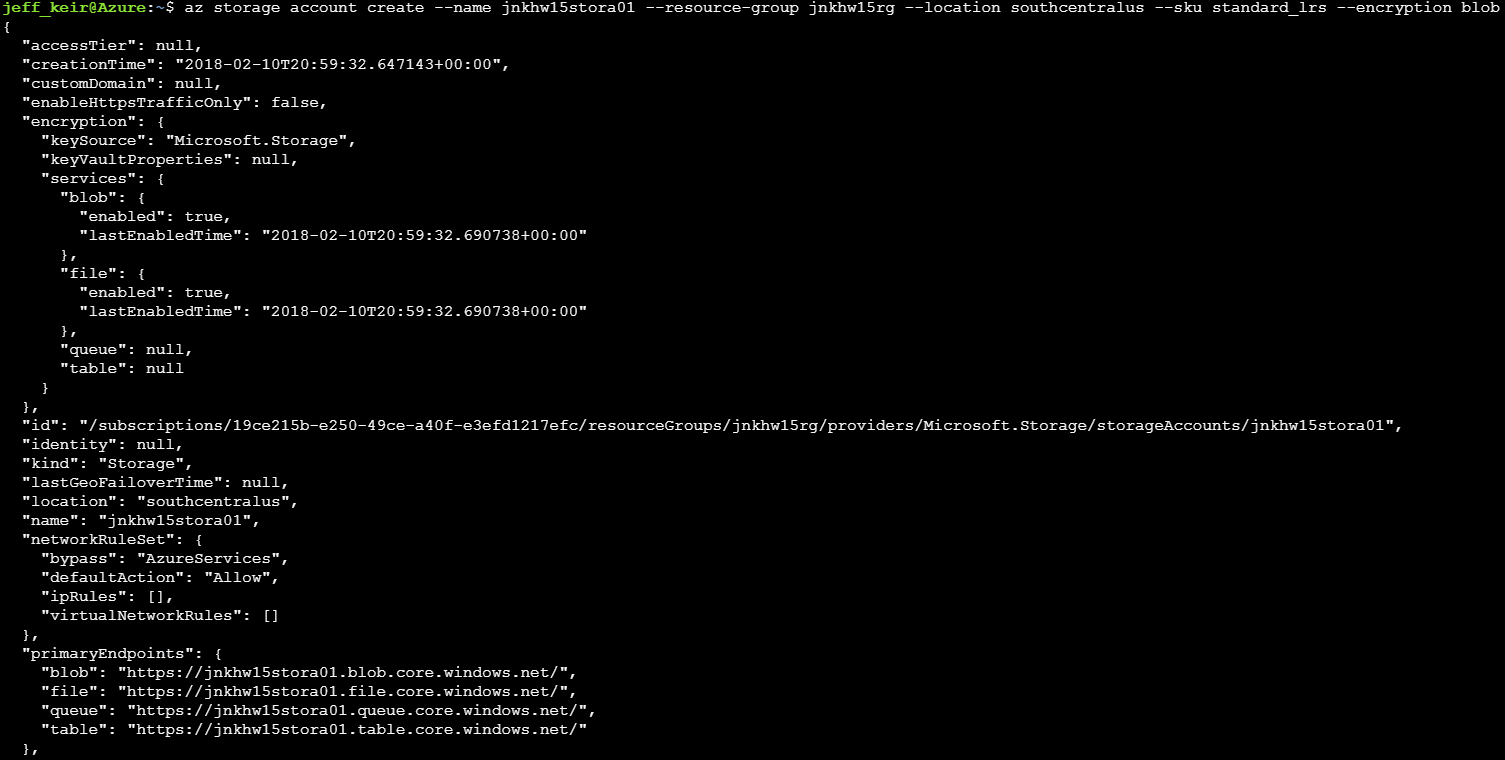
Create a resource group:

az group create --name jnkhw15rg --location southcentralus



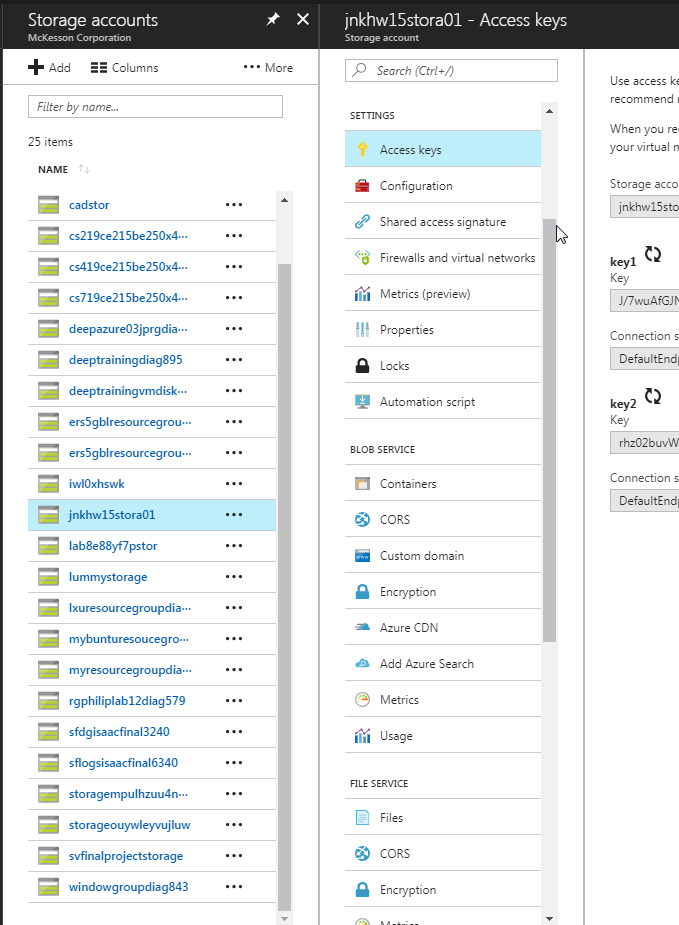
Create a storage account:

az storage account create --name jnkhw15stora01 --resource-group jnkhw15rg --location southcentralus --sku standard\_lrs --encryption blob



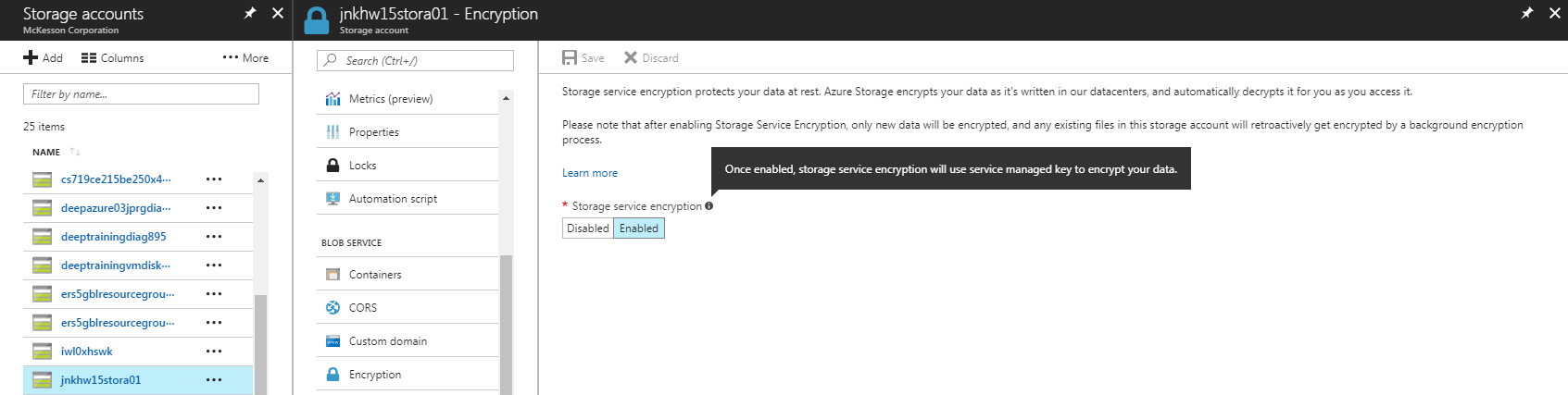
Notice for the blob creation that blob & file are enabled, table and queue are not by default (but can be added)

Examine the properties of the storage encryption from the Azure Portal



Encryption keys are automatically generated upon storage creation.

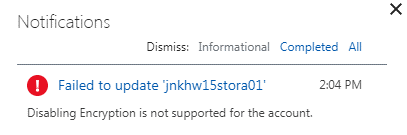
Within the storage account properties blade there are also blob and file service encryption sub-blades visible.



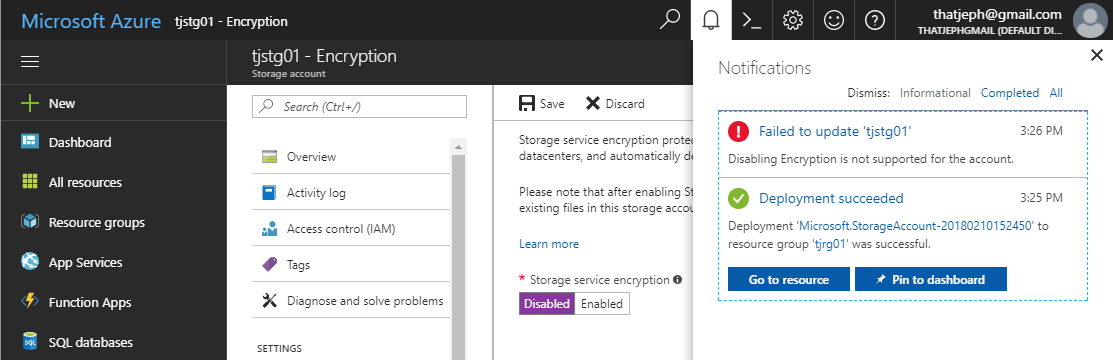
Storage account created with encryption by default

Disabling encryption is not supported. At all. Like, why is this here?

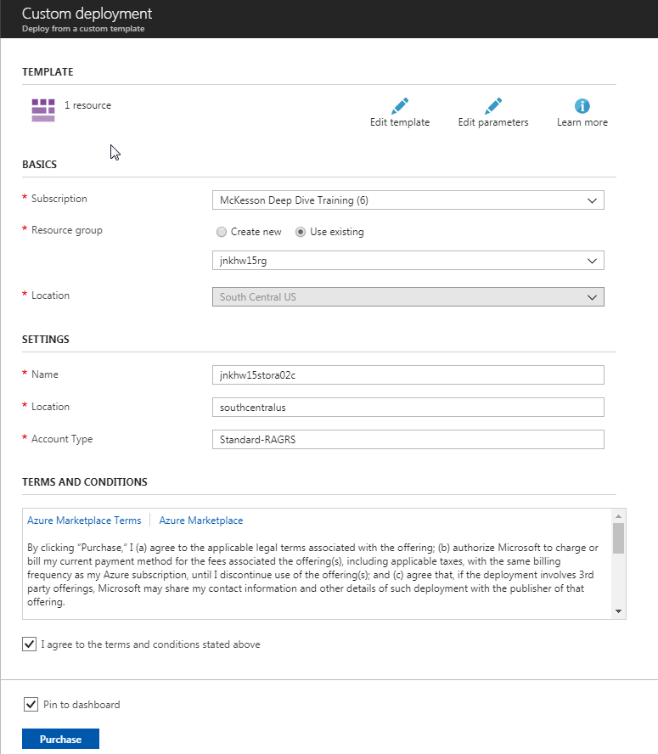
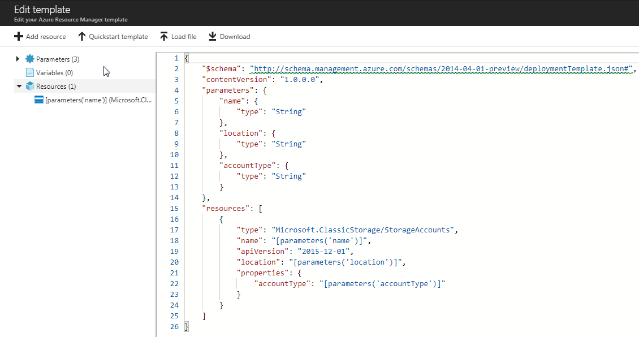
It is here solely to enable encryption on legacy storage account sub-services created prior to June of 2017.

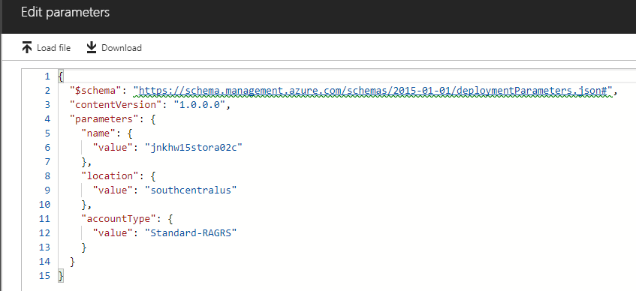


To test whether disabling is a limitation of this trial account, I created a new trial account using a personal credit card. The results were identical to above – disabling RMStorage encryption in the portal simply does not work.



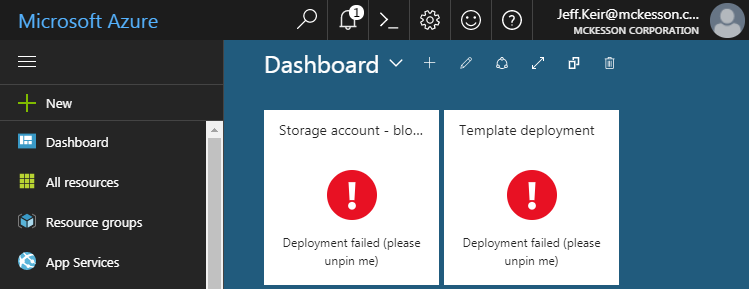
Consider a custom deployment to enable Classic Storage. It is possible to engage a custom deployment and modify the JSON template or just the parameters within the JSON template.



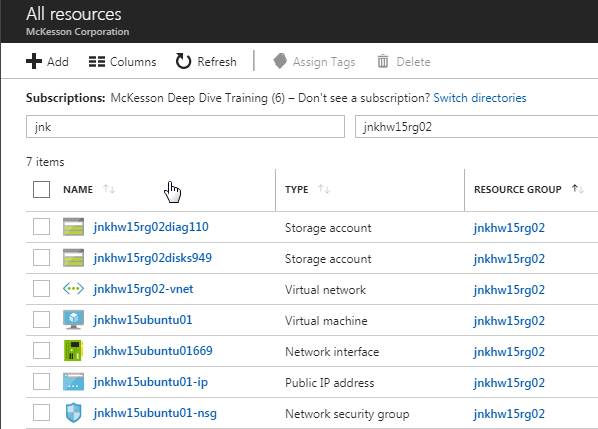


**End Problem 01, Solution 01**

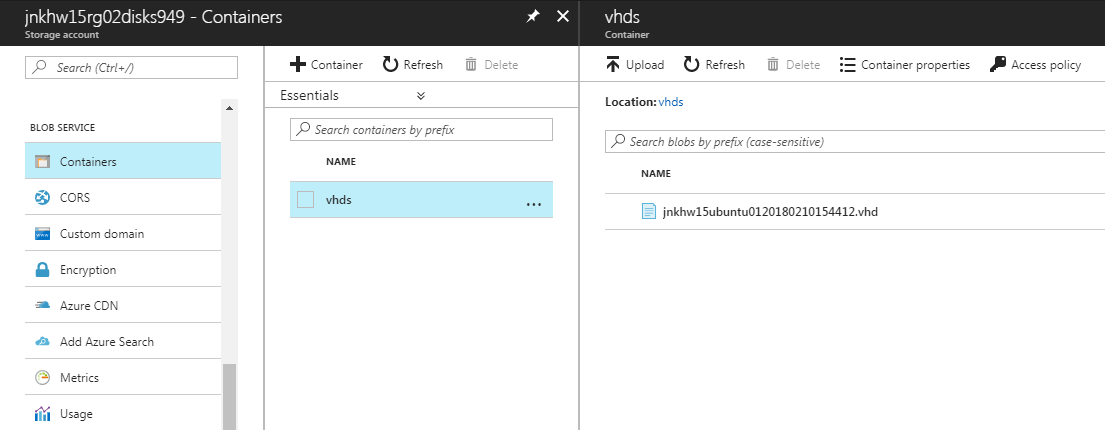
The results are lackluster.



Dig deeper:  
Instantiate an Ubuntu 16.04 server VM using *unmanaged* disks (according to Microsoft all managed disks are encrypted by default, unmanaged disks are not mentioned) and review the results in the resources pane. Below is a capture of the parts making up an Ubuntu (or any) server within the cloud:

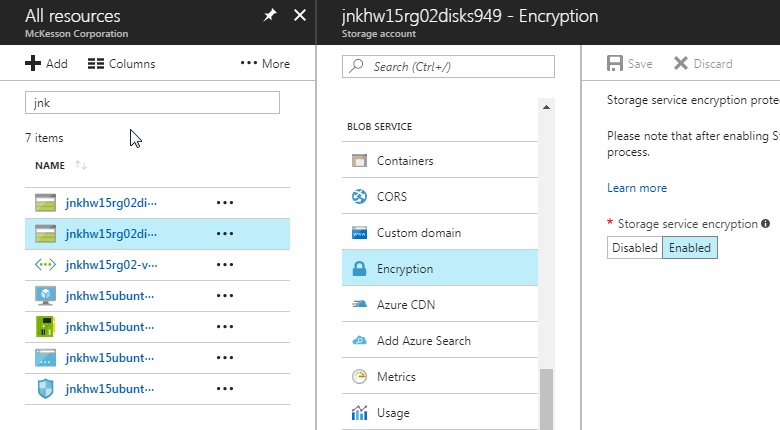


…disks949 are the germane object in this case, a device type of “storage account”. Virtual hard disks (VHDs) are stored as a .vhd file within a blob container. We can visualize this by drilling into the blob settings on the storage account:

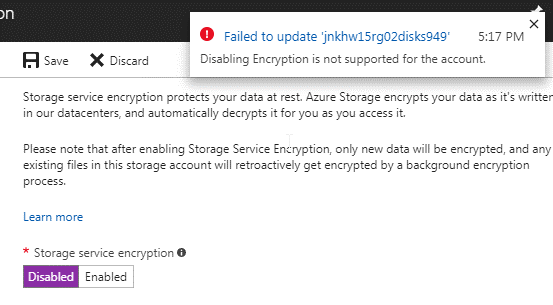


This vanilla Ubuntu server has only a single disk for OS and storage.

When we explore the encryption blade under the service, we find that the blob is in fact encrypted by SSE.



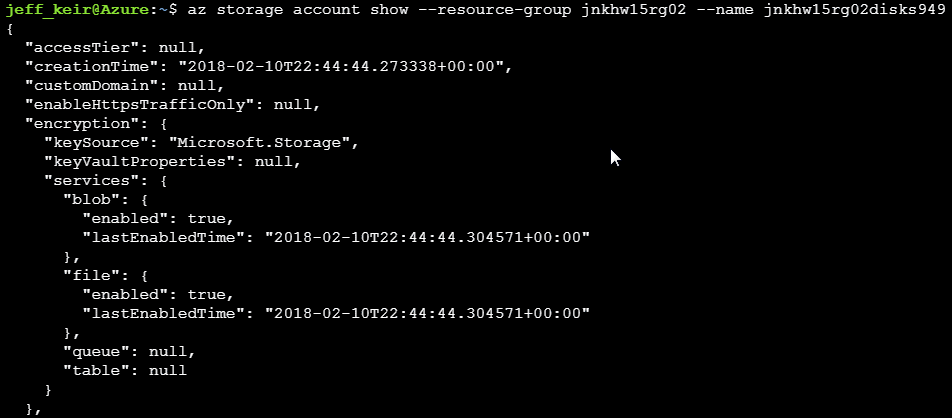
…and we are unable to disable it:



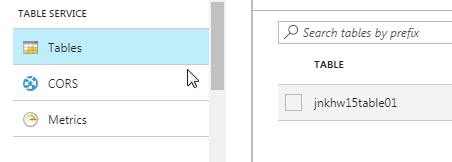
We now have two storage accounts created first deliberately using the CLI, and second created coincident to instantiating a server VM. In both cases the SSE encryption was enabled without question and is ‘locked’ against change via the portal.

What about the CLI?

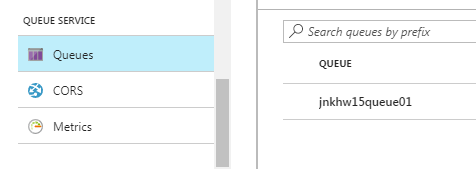
Here we can see encryption as a property. Note that Queue and Table are “null” which was one of our tests for this problem – there is no encryption from SSE on these elements. At this stage these elements do not exist.



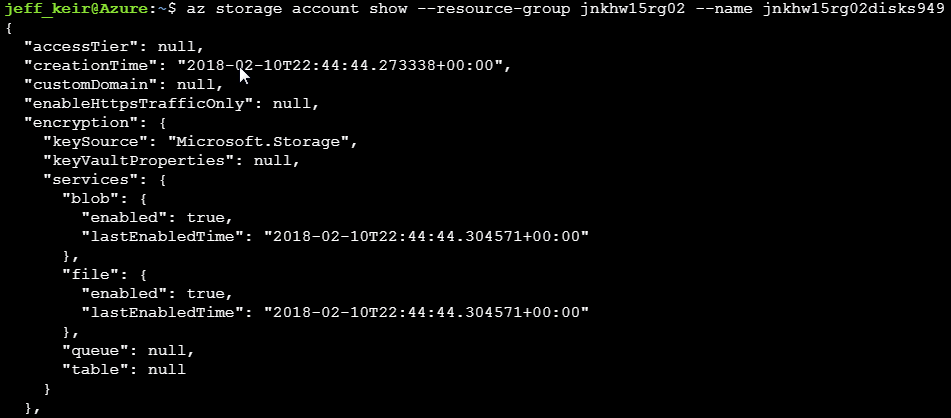
To understand if Queues and Tables instantiated after a storage account has been created, a Queue and a Table are added via the portal:

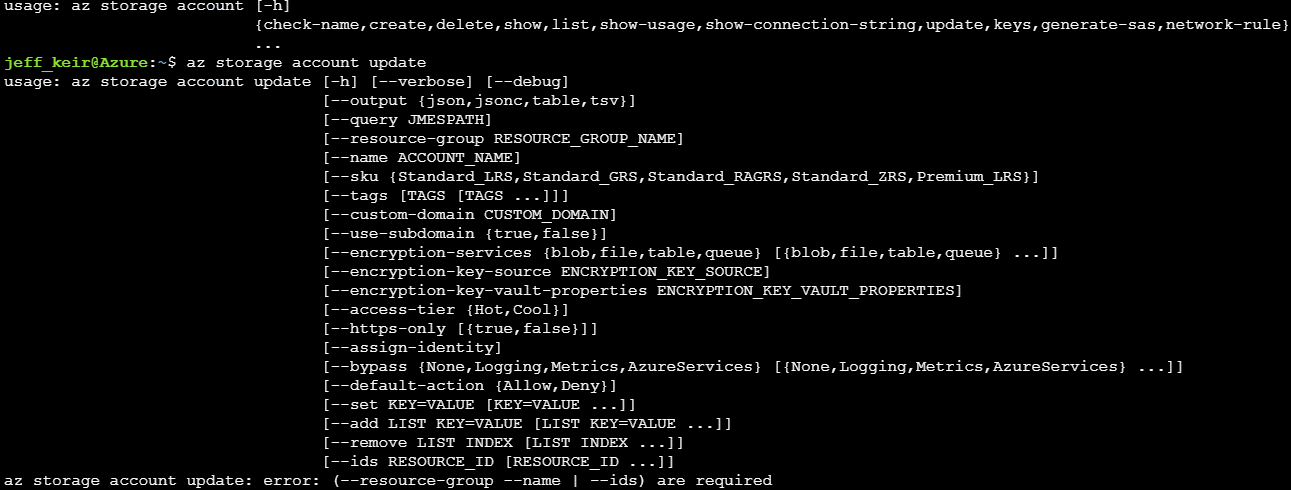


No encryption property is apparent even after creating a table and a queue



Re-visualizing in the CLI shows the encryption state for these storage types remains null even after creation within an encrypted storage service:

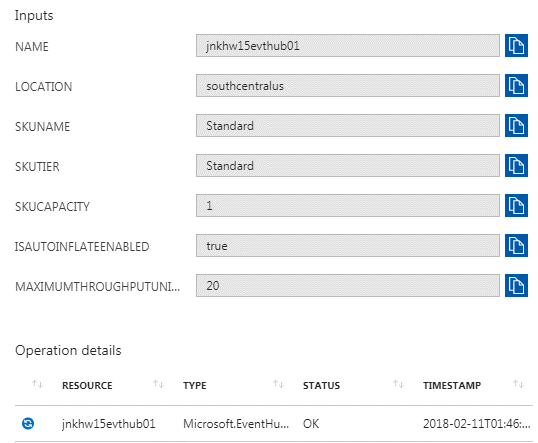


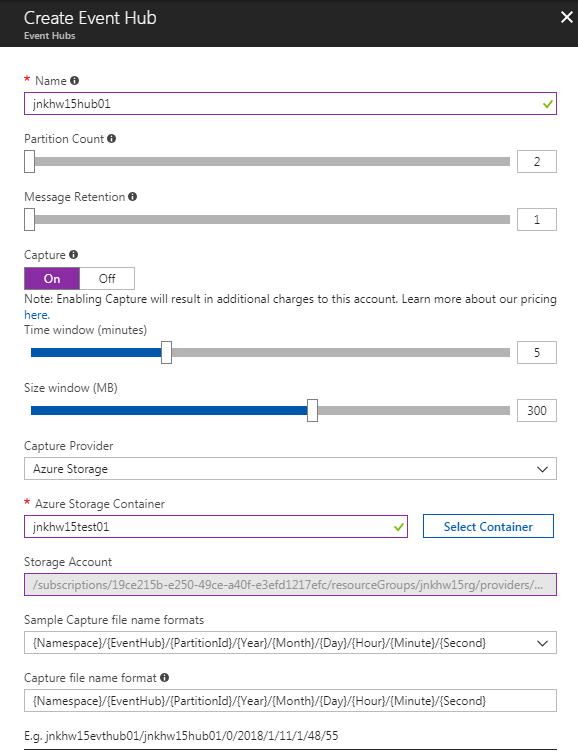
Within the CLI usage hints for storage we can see that there are some encryption elements we can change:  


Namely: service type, key source, and key vault properties. But there is no actual toggle.

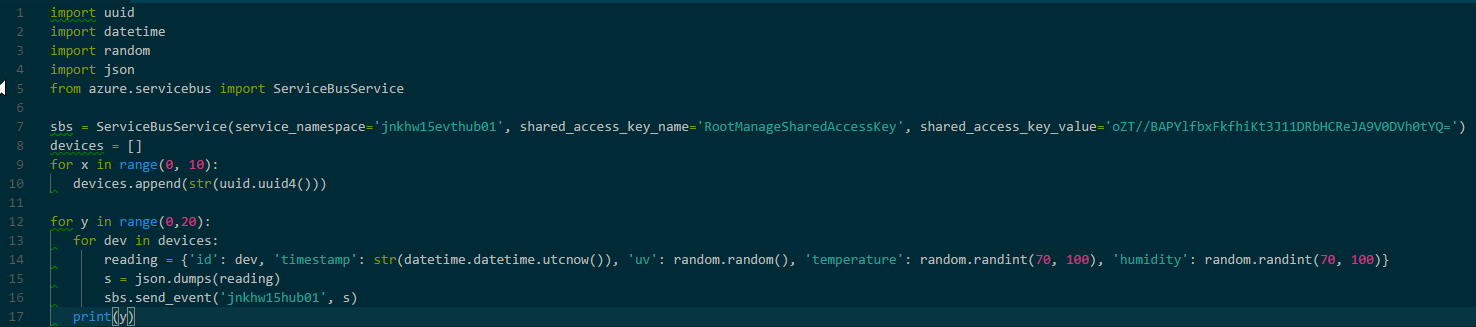
Recreate the messaging services from Deep Dive Homework 09, demonstrate the encryption state, and that the base functionality is successful:

Namespace:

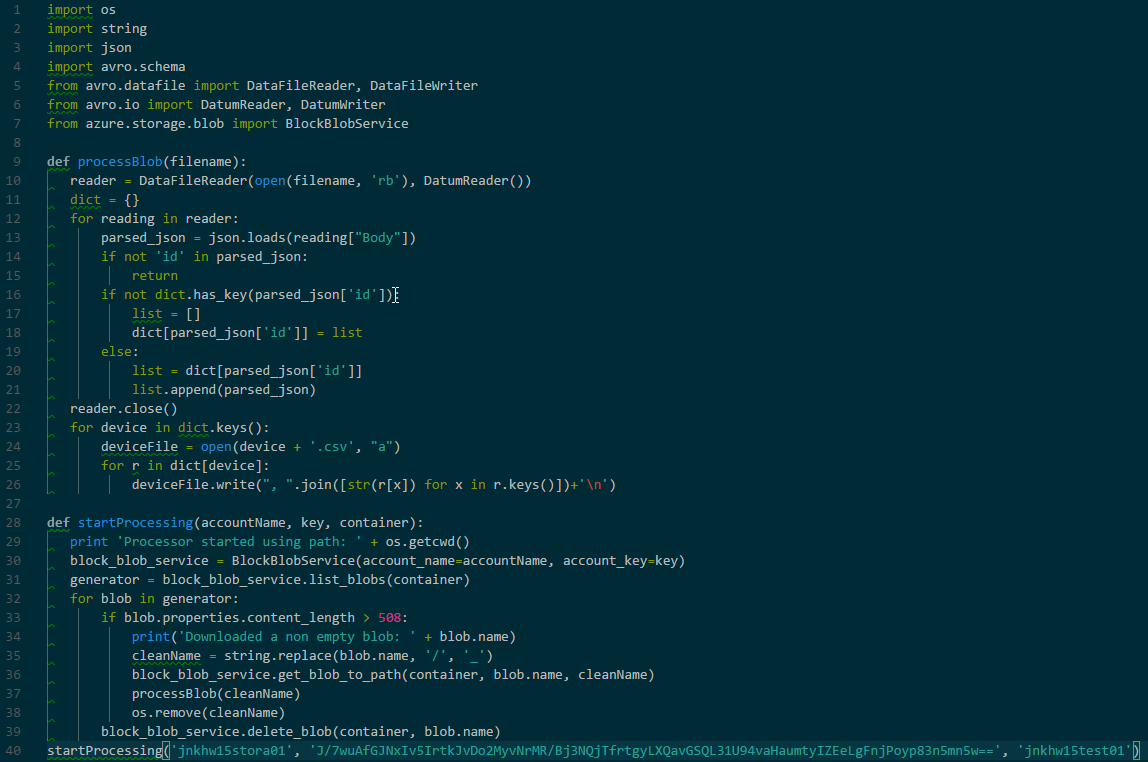




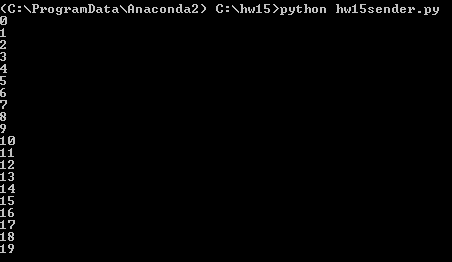
Code – the following code snippets are taken from <https://github.com/blumu/azure-content/blob/master/articles/event-hubs/event-hubs-archive-python.md>:  
This script sends 20 messages to the storage service

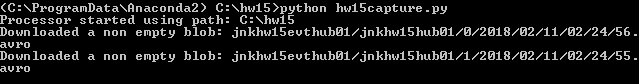


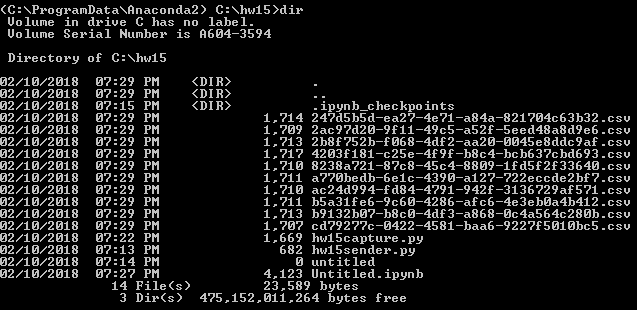
This script calls upon the storage service to process messages

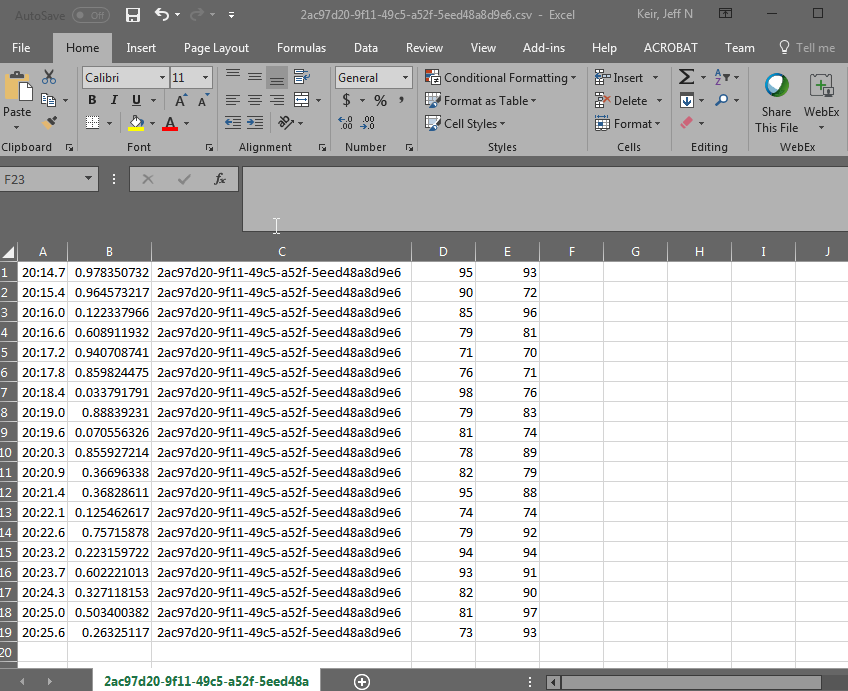


The practical application of these code samples is simply to see that data is indeed written into an encrypted storage blob “jnkhw15hub01”, subsequently extracted and manipulated, the results written to blob “jnkhw15test01” and then a copy to the local machine and readable.









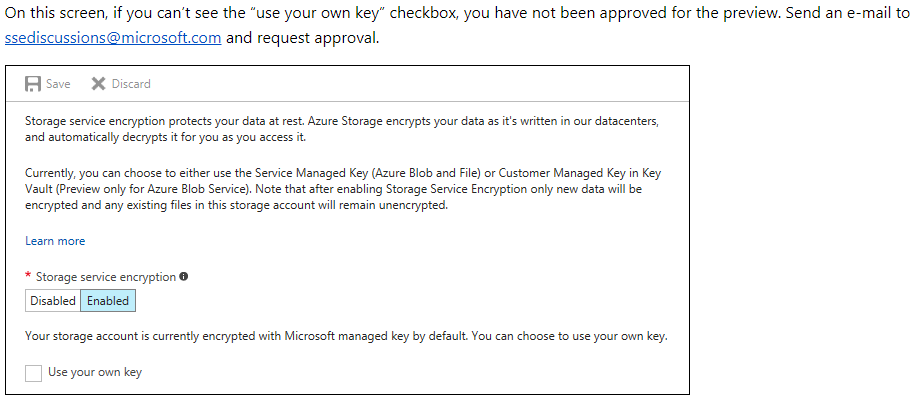
# Conclusion:

It is evident that encryption is enabled, as Microsoft claims, on Resource Manager Storage by default. Furthermore, it is easy to see the settings either in the portal or via the CLI.

What is not evident is any mechanism by which the administrator could deactivate the service. Though the CLI and APIs are referenced frequently in the documentation, no actual syntax is provided to toggle the encryption state. Relatedly, there is little given information on the amount of disk-access-time added to operations by the existence of Storage Service Encryption. Practical experience indicates that the overhead is small, though one imagines a proportionate increase in wait time as the size of a given data-set increases.

As recently as 10 months ago it may have been possible to instantiate unencrypted storage and toggle the setting, but today it appears impossible.

All indications are that Microsoft is still working on the tools that allow developers to manage encryption across the storage service *manually*. The foundation for this conclusion is inferred from a reference to the preview of customer-managed storage service encryption on the document page:



<https://docs.microsoft.com/en-us/azure/storage/common/storage-service-encryption-customer-managed-keys>

1. <http://breachlevelindex.com/> [↑](#footnote-ref-1)
2. https://azure.microsoft.com/en-us/blog/announcing-default-encryption-for-azure-blobs-files-table-and-queue-storage/ [↑](#footnote-ref-2)