SAS® GLOBAL FORUM 2020 #SASGF

Paper 4981-2020

Fast Deployments and Cost Reductions: SAS® in the Azure Cloud with HDInsight and the Azure Data Lake

Andrew Williams, Vanquis Bank Ltd

ABSTRACT

Cost reduction is often a key objective of businesses but reducing costs without compromising on performance can be a challenge. The objective of this project was to reduce costs while improving performance. We found that combining SAS® with Microsoft Azure Data Lake Gen2 could be part of the solution. It encompasses the power of a file system that is compatible with Apache Hadoop with integrated hierarchical namespace, along with the massive scale and economy of Microsoft Azure Blob storage. With the use of SAS® Grid Manager for Platform, our Kerberized platform talks to multiple Microsoft Azure HDInsight clusters, enabling us to distribute the different demands of users while reading from the same data lake. There are some key requirements that you need to be aware of prior to deployment but once they are in place, the deployment time for a new SAS and Hadoop cluster is hours, rather than weeks or months. The advantage of having the data physically stored separate from the Hadoop cluster is that once you have destroyed the cluster, the data is still there. In this paper, we discuss this further and explain the steps required to make the most out of SAS® integrating with Azure HDInsight on the Azure Data Lake Gen2.

INTRODUCTION

This paper demonstrates how to leverage the power of the Azure Data Lake Gen2, using HDInsight whilst connecting from SAS 9.4M6 or SAS Viya. The paper will look at the design options that we implemented, what pre-requirements are needed for the solution, and will include important information, such as what security is required, and performance settings for optimizing the solution.

The main objective of this project was to reduce our cost footprint for SAS® and Hadoop. We had a SAS and Hadoop cluster in the Azure Cloud in place for a few years prior to this project, but as data size increased, and business requirements changed, we needed to have a flexible approach to the business needs, whilst also keeping costs under control.

One objective of this project was to be able to run multiple different SAS and Hadoop environments, while also reducing the high cost of the platform.

After much debate and scrutinizing costs, it was clear that the majority of the cost could be attributed to the storage layer. We considered various solutions for this and the Azure Data Lake Gen2 seemed by far the most flexible and cost effective. As HDInsight (which is effectively a Hortonworks Cluster) goes hand in hand with the Azure Data Lake, we decided to migrate to this as well.

This decision meant that we could detach the storage of HDFS from the Hadoop Cluster. This gave the flexibility to create multiple HDInsight clusters pointing to the same source data. We then also have the option to scale down or destroy HDInsight clusters, without the requirement for exporting or redistributing any of the source data.

You will need to have Azure Active Directory Domain Services set up before you configure HDInsight. This means that you need to sync your on-premise users into Azure's version of

Active Directory. This may sound simple, but there are certain caveats that you will need to address first, and some important rules to remember going forward.

There is not much difference between connecting to HDInsight and connecting to Hadoop, so while it is a straightforward change for users, there are a couple of important features that this paper will address.

The paper will also note which Azure VM's we recommend to use for the SAS cluster. We found that for the SAS Grid environment in particular, performance was high when using the NVMe disks as the SAS Work. The paper will also note any optimizations you can use to speed up your connection to HDInsight.

The end to end deployment of building an HDInsight Cluster, and connecting from a SAS environment, can be completed under an hour with the aid of automatic deployment, using scripts via Ansible or similar. This gives users the ability to purchase environments on demand for independent projects, while keeping costs low due to how the data is stored.

BACKGROUND

Having migrated to Azure a few years ago, after time it became apparent that the ongoing costs of the platform were not feasible. As with most Cloud providers you do not own the infrastructure, you pay a monthly fee (you can pay a lower rate if you purchase for one year or three years). One obvious advantage of using Cloud technology is that if you need to upgrade a server you can do so with a click of a button. However, we were discovering that the main cost of the platform was the storage.

There were various options with storage in the Cloud: Managed Disks, Storage Blobs, and the Data Lake. Our existing setup comprised one Hadoop Environment which consisted of 5 head nodes and 15 worker nodes, with around 90TB of storage attached. There are different levels of Azure Managed Disks (Standard HDD, Standard SSD, Premium SSD), along with different types of redundancy built into the storage. The larger the disk, the more IOPS, and the higher throughput you will get, although you will pay a premium for this. Therefore, it is important to make the right decision if you are going to use Azure Managed Disks because you want optimal performance without paying an excessive amount. With Hadoop however, you want to ensure that you have enough storage to handle the data, and taking into account the replication factor, you need to ensure that you have three times the amount of storage you require. Over time costs started to mount, and we only had the one Hadoop Environment which was not sustainable for development and production. However, by adding another cluster we were potentially looking at doubling our costs.

On our existing platform we had only one Hadoop Cluster, and one SAS Environment. Originally this was using SAS Grid on Hadoop, but this was later switched to SAS Grid on Platform to free up extra resources on the Hadoop Cluster. It would be preferable to have multiple SAS environments for promotion, and separate development, and production processes.

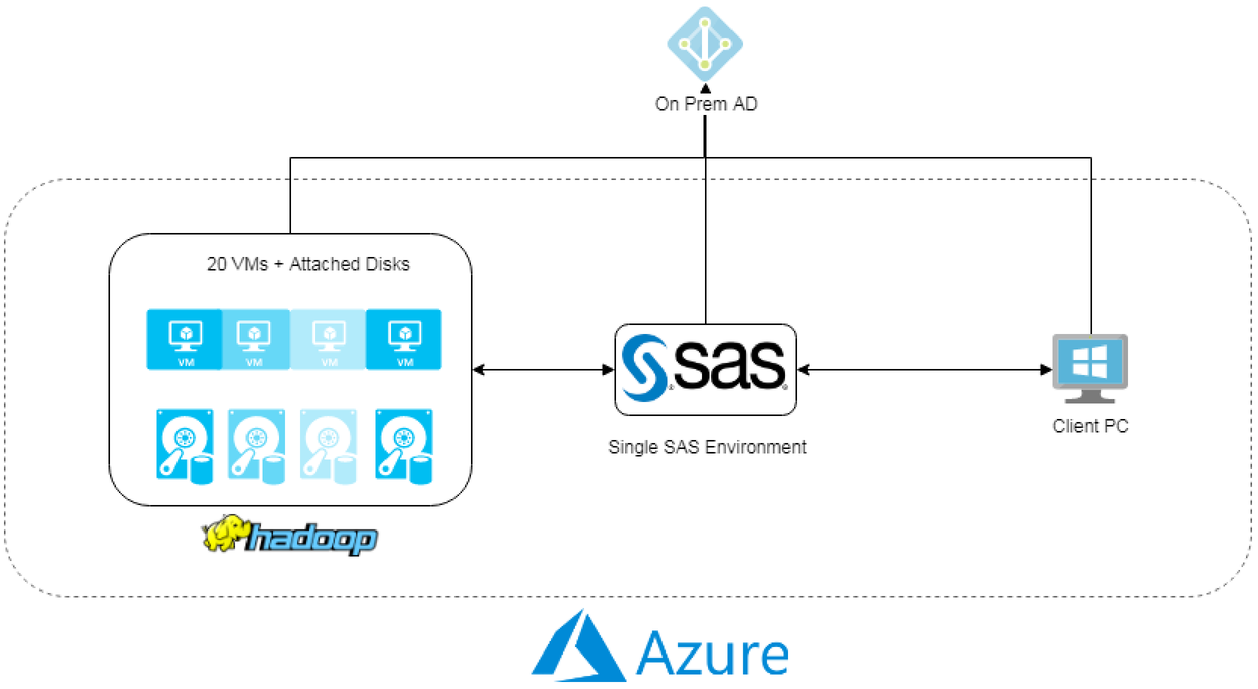


Figure 1. Existing Platform

As shown in Figure 1, on the existing platform, users would log onto a client PC hosted in Azure. They would primarily connect to a single SAS environment, which would talk to a single Hadoop cluster hosting all the data. All servers are configured to talk back to an on-premise Active Directory for authentication, this is an important note for later in the paper.

Our challenge was to create multiple environments for the users whilst reducing costs.

DESIGN

It was clear that the Azure Managed Disk costs were not sustainable. Considering the different options offered by Azure, it was a choice between Blob Storage or the Azure Data Lake Gen2. The Azure Data Lake Gen2 is built on Blob Storage but you can define POSIX permissions, so we decided to go with this storage type.

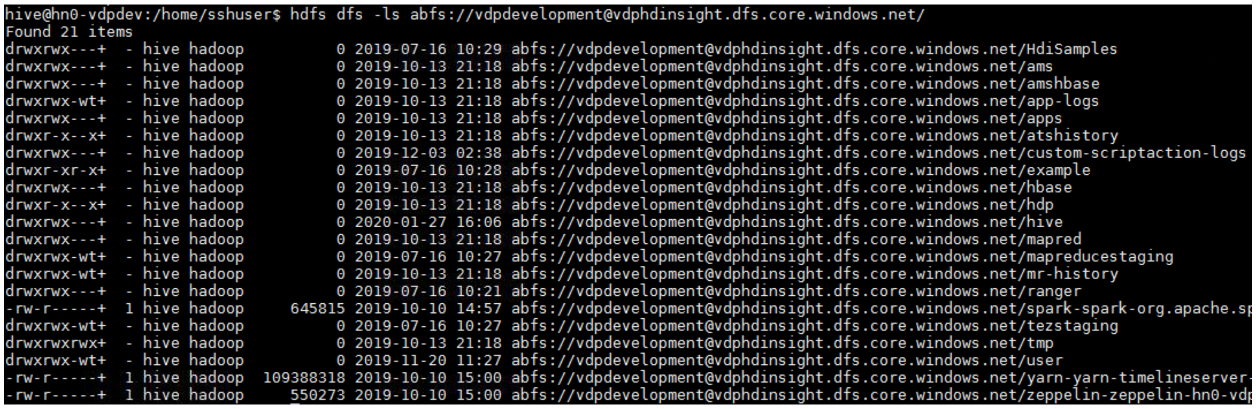
AZURE DATA LAKE GEN2

The Azure Data Lake Storage Gen2 storage is designed to service multiple petabytes of information while sustaining hundreds of gigabits of throughput. The petabyte service levels exceeded our needs, but we wouldn't be far from that requirement in the future. Unlike most storage offerings from Azure, an important factor of the Data Lake Storage Gen2 is the addition of a hierarchical namespace. The hierarchical namespace organizes objects/files into a hierarchy of directories for efficient data access.

You can manage the Data Lake Gen2 via the Azure Portal or Azure Storage Explorer, or like traditional HDFS, you can run the same commands on the Azure DataLake Gen2, apply ACL permissions, create directories etc.

To access files stored on the Azure Data Lake Gen2 using HDFS commands, you require the ABFS driver to be installed (this is generally available within all Apache Hadoop environments, including Azure HDInsight, Azure Databricks, and SQL Data Warehouse).

The ABFS driver is designed specifically for large data analytics. The corresponding REST APIs are connected through the endpoint dfs.core.windows.net.



```
hive@hn0-vdpdev:/home/sshuser$ hdfs dfs -ls abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/
Found 21 items
drwxrwx---+  - hive hadoop          0 2019-07-16 10:29 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/HdiSamples
drwxrwx---+  - hive hadoop          0 2019-10-13 21:18 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/ams
drwxrwx---+  - hive hadoop          0 2019-10-13 21:18 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/amshbase
drwxrwx-wt+  - hive hadoop          0 2019-10-13 21:18 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/app-logs
drwxrwx---+  - hive hadoop          0 2019-10-13 21:18 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/apps
drwxr-x--x+  - hive hadoop          0 2019-10-13 21:18 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/atshistory
drwxrwx---+  - hive hadoop          0 2019-12-03 02:38 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/custom-scriptaction-logs
drwxr-xr-x+  - hive hadoop          0 2019-07-16 10:28 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/example
drwxrwx---+  - hive hadoop          0 2019-10-13 21:18 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/hbase
drwxr-x--x+  - hive hadoop          0 2019-10-13 21:18 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/hdp
drwxrwx---+  - hive hadoop          0 2020-01-27 16:06 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/hive
drwxrwx---+  - hive hadoop          0 2019-10-13 21:18 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/mapred
drwxrwx-wt+  - hive hadoop          0 2019-07-16 10:27 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/mapreducestaging
drwxrwx-wt+  - hive hadoop          0 2019-10-13 21:18 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/mr-history
drwxrwx---+  - hive hadoop          0 2019-07-16 10:21 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/ranger
-rw-r-----+  1 hive hadoop     645815 2019-10-10 14:57 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/spark-spark-org.apache.sp
drwxrwx-wt+  - hive hadoop          0 2019-07-16 10:27 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/tezstaging
drwxrwxrwx+  - hive hadoop          0 2019-10-13 21:18 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/tmp
drwxrwx-wt+  - hive hadoop          0 2019-11-20 11:27 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/user
-rw-r-----+  1 hive hadoop  109388318 2019-10-10 15:00 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/yarn-yarn-timelineserver-
-rw-r-----+  1 hive hadoop     550273 2019-10-10 15:00 abfs://vdpdevelopment@vdphdinsight.dfs.core.windows.net/zeppelin-zeppelin-hn0-vdp
```

Figure 2. Azure Data Lake Gen2 via HDFS commands

Unlike previous Cloud storage options, you can enforce security by defining POSIX permissions on directories or individual files.

Cost

The Data Lake Storage Gen2 offers low-cost storage capacity and transactions. With most Cloud providers there is a cost for storing the data, usually separated into different types, HOT, COLD and ARCHIVE, with prices dependent on access requirements.

In addition to storage costs, with most cloud providers you will also pay a transaction fee, generally for read and write operations.

Built-in features such as Azure Blob storage lifecycle, helps ensure costs are kept to a minimum. When data transitions through a complete lifecycle the billing rates for ADLS Gen2 change

We have reduced our storage costs by approximately 90%, by moving from Premium Disks attached to the Hadoop Cluster, to ADLS Gen2.

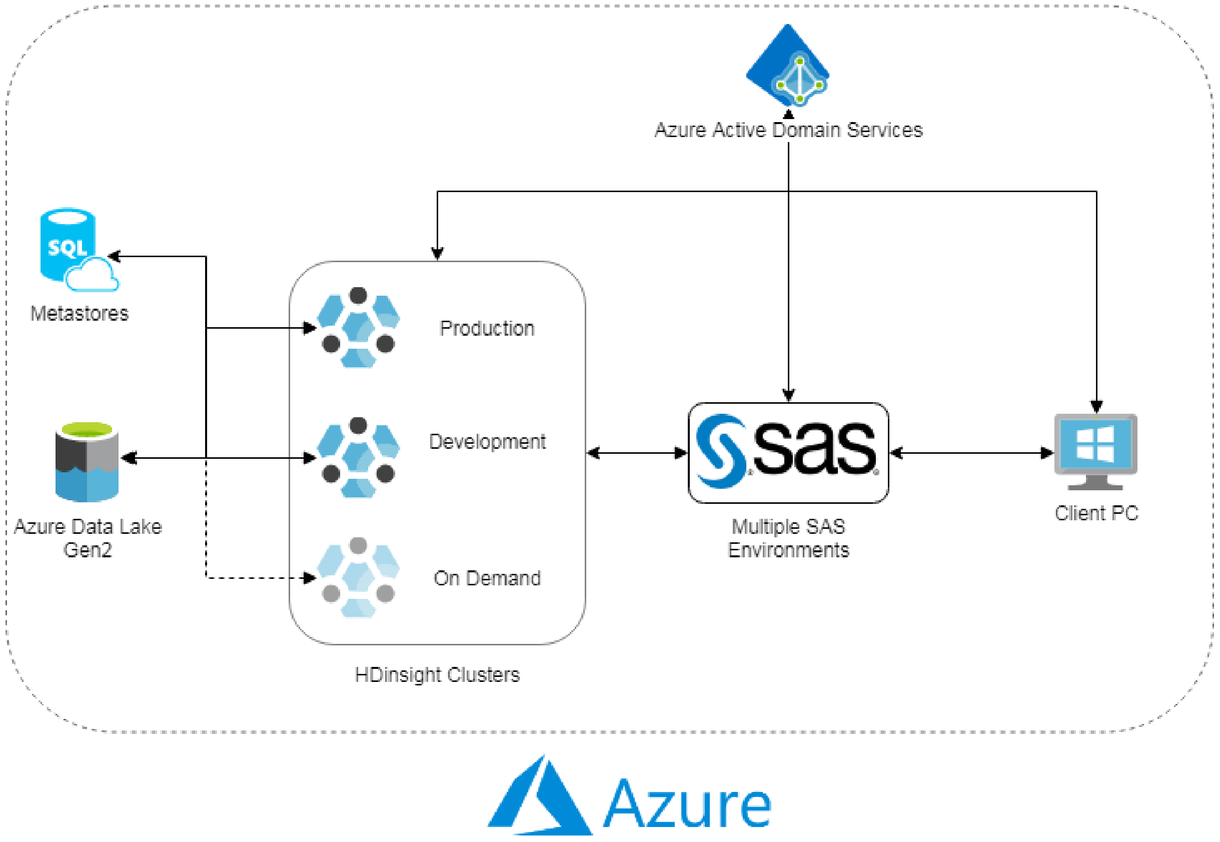


Figure 3. New Platform Design

SAS AND HDINSIGHT

There are various conditions that we need to establish for SAS to work smoothly with HDInsight and the Azure Data Lake.

Azure Active Directory Domain Services

A pre-requirement of HDInsight is to have all users synced into the Azure Active Directory.

We already had users syncing to our Azure Active Directory which links applications such as Office 365. However, a requirement of HDInsight is to secure resources to the Azure Active Directory. Our existing Hadoop environment is currently secured to our on-premise Active Directory, but for HDInsight this isn't possible.

You will need an Azure Administrator for your site to switch this on, which may take a few hours to sync all of the on-premise directory. Once completed you can then register any VM's to Azure AD, and use that as the domain controller.

NOTE: Be careful as user passwords will not sync up, so you need to change your password, to then force a sync, and be able to log into an Azure AD secured VM. Also, for on-premise accounts where you might have the setting 'password never expires', this setting is no longer valid. You will need to reset the password every 3 months to sync with Azure.

Managed Identity

You will also need to create a managed identity user in Azure. This will be used to register service principles and DNS entries for HDInsight, so this will require permission to edit the Azure AD that we have created. You will need to give this account permissions as owner of the storage that you are going to use, in this case ADLS Gen2, because all Hive queries by default will connect to the storage using this managed identity.

Kerberos

HDInsight clusters are secured with Kerberos authentication as default, and this is linked to Azure Active Directory. To avoid cross domain issues or complications you should register the SAS Virtual Machines in Azure Active Directory, as opposed to the on-premise Active Directory.

We have used SAS Grid with Platform with the normal configuration steps, to allow Kerberos delegation. There are no special steps required here to access HDInsight.

SAS in the Azure Cloud

We carried out testing with various types of VMs that Azure offer, there are memory optimized VMs or storage optimized. As previously mentioned with regards to disks, the larger VM you select the better IOPS you will have, but obviously this will cost you more. We settled on the below configuration following copious testing:

Metadata 3x E8s v3 (8 vcpus, 64 GB memory)

Grid 3x L16s V2 (16 vcpus, 128 GiB memory, NVMe 2x1.TB (used for SAS Work ~400-500 read/write mb/s)

Midtier 2x E16s v3 (16 vcpus, 128 GiB memory)

For the SAS Grid (where most of your execution will happen), the L Series are by far the best performers if you use the NVMe (/mnt/resource) that comes with the hosts. We found that adding disks to the host and using SAS Work will not yield the same performance.

DEPLOYMENT AND CONFIGURATION

HDINSIGHT

To create HDInsight clusters, you can simply go onto the Azure Portal, fill in a few pages of information, and then deploy. If you are happy with the settings then you can save these to a file and redeploy via an Azure Script in Powershell etc. or make a call and deploy in Ansible. Then you have an automated method of creating an Hadoop Cluster, with the ability to clone the clusters, or create in the morning before users log on, and destroy in the evening, if you don’t want to pay to have it running overnight.

Metastore

To reduce manual tasks, if you are going to destroy and recreate your HDInsight clusters multiple times, then it is a good idea to create a SQL database. This must be a SQL database in Azure. The advantage of this is that tables and database Metadata and Ranger policies can be stored away and kept for future use. We have used the same SQL database for each of our clusters. The reason for this is that if we spin up a new cluster, then it is extremely easy to grant access to a database as the Metastore is already active. We just need to give the Ranger permissions to see the table or database. If you decide to use this method, then ensure that your Hive databases are unique if you have multiple HDInsight clusters.

FileSystems

Once you have created an Azure Data Lake Gen2 Storage Account, your HDFS filesystem will be created inside a storage container. You should have separate containers for each HDInsight cluster you create, these can be under the same Azure Data Lake Gen2 Storage Account. This is the method that we have deployed. It gives the ability to create a cluster on demand to a set of users if required and have access to data from a different cluster, or in a shared storage container.

It is important to note that while HDInsight can talk to multiple storage containers in the same Azure Data Lake Gen2 account, it can only talk to one Azure Data Lake Gen2 account at a time. This is worth noting for any design considerations.

When deploying HDInsight, HDInsight can create the container if it doesn't previously exist. Note of caution on this, when running the SAS Hadoop tracer script, this may fail as it is required to run commands against ADLS and connect to Ambari resources. You can however run the script as the Hive user directly via command line.

Security

As previously mentioned, when deploying HDInsight, it is best to point the Ranger database to an existing SQL database to keep permissions when clusters are destroyed and recreated. When deploying you will need to select which AD Groups you wish to use to sync users onto the platform. Traditionally, you would point your cluster to an OU to gather the users. However, in this instance you simply select which group(s) to use. It is important here to select all of the users that you need. By default, all users will have the same permissions when you create the cluster on the first occasion, with the requirement to select any Ambari Admin permissions after creation.

CONNECTING SAS TO HDINSIGHT

You will find the process of connecting SAS to HDInsight similar to that of connecting SAS to most Hadoop providers. However, there are a few points to be aware of to ensure that this runs smoothly, and to avoid any unexpected errors.

Hadoop Tracer

As with the Hadoop Tracer supplied by SAS, you can either run via the SAS Deployment Wizard, or copy the Hadoop Tracer onto the Hadoop cluster, and run from there. With HDInsight we have found that it is much easier to run the command on the HDInsight cluster, rather than from the SAS Deployment Wizard.

The Hadoop Tracer module will run a set of scripts which are used to collect JAR and XML files from HDInsight, these will then enable SAS to connect to HDInsight through a Libname statement or the Pass-Through Facility.

If you plan on destroying your HDInsight cluster at the end of the day then you are going to find that your JAR and CONF files may be out of date. Microsoft have added slight updates to JAR files so you should ensure that you have the latest if you rebuild the cluster.
When running the script each time, if you use Zookeeper to connect from SAS you will find that some of the host names have changed. For example, when you first build the cluster it may be zk0,zk1,zk2, but upon rebuilding the hosts may be zk0,zk3,zk5.

When running the Hadoop Tracer script, you will need to be an admin in Ambari, as it runs certain commands. Also, you will have to be able to run HDFS commands on the ADLS filesystem which will read and write to certain locations. You have two choices; you can run the script as the Hive user, or, if you create the ADLS filesystem prior to deploying then you can add a service account, which will inherit read/write/execute permissions on anything created.
Remember that your data doesn't have to be stored in the same ADLS Gen2 container as your default filesystem.

Run as End User v Run as Hive

If you are familiar with Hadoop, most companies will ensure that all Hive queries run as the end user. This allows you to set user limits and restrict how much processing power one user can take up, to allow a fair use policy.

HDInsight works a bit differently if you are going to use the Azure Data Lake Gen2 as well.

Firstly, by default everything is configured to run as the Hive user (remember this uses the managed identity to connect to ADLS Gen2), for ease of use. As the filesystem is created from scratch, it won't know which user groups to secure it with. By default, it is secured to only allow the managed identity to read the data on ADLS Gen2. If you destroy and then recreate the cluster, the filesystem will remain so that any changes to permissions will be kept for future clusters created.

If you want everything to run as the end user, then you will need to apply user/group permissions to the Data Lake, either in Storage Explorer, HDFS 'setfacl' commands, or via Powershell commands.

You will need to ensure that within directories such as athistory, tmp, and other directories in the ADLS Gen2 filesystem, users have the appropriate read and write permission.

On the container level, users or groups will need to be granted read and execute permission.

Finally, in the location where Hive databases are stored, the required permissions for read or write access will be needed to be granted to the appropriate group(s) or user(s).

DNS Entries

As we know, the managed identity is responsible for adding the cluster nodes and service accounts into Active Directory. Then your SAS Virtual Machines can use the Active Directory DNS name of the nodes to communicate.

However, on 'cluster destroy' these entries are not removed. If you build a new cluster with the same name, then you will find that the IP addresses stored in your AD are different.

The options are either to have an Azure Admin at your site remove the entries before you recreate the cluster, or you can copy the hosts file from one of the HDInsight nodes to your SAS Virtual Machines.

Log into Ambari first

When using SAS or Beeline to connect to Hive via JDBC and read data, this will work fine. However, when it comes to writing data to Hive, this is a different story. After submitting your query, after around 30 minutes, it will reply with the below error regarding token not found:

```
ERROR: Status code: -1 error code: null error message:
InvalidAbfsRestOperationExceptionjava.io.IOException:
com.microsoft.azure.storage.exceptions.TokenNotFoundException: Token
does not exist in TokenManager (Response Code: 404)
```

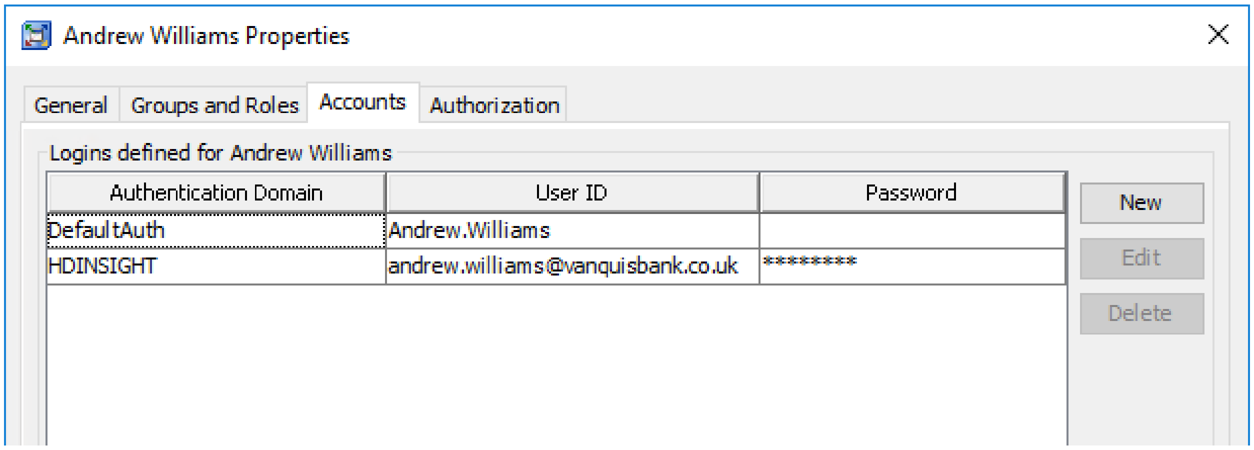
This is because you need to generate an OAuth token to be able to communicate with the ADSL Gen2 storage. One way to do this is to log into Ambari URL. You will need to do this each time that the cluster is created. The user logon for this is your full email address, the same that you would use to log into the Azure Portal, or your email online.

The workaround that we have for this is to add the following code into the SAS Autoexec. This will then make a HTTP call to the Amabri URL and using the password and email address stored in SAS Metadata, complete the authentication behind the scenes for our users.

```
filename resp TEMP;
filename headers TEMP;

proc http
      method="POST"
      url="https://prodcluster-int.azurehdinsight.net"
      WEBAUTHDOMAIN="HDINSIGHT"
      headerout=headers
      out=resp
      HEADEROUT_OVERWRITE;
run;
```

In SAS Metadata we have the authentication domain setup as HDINSIGHT to match the above code.


Andrew Williams Properties
General
Groups and Roles
Accounts
Authorization
Logins defined for Andrew Williams
New
Edit
Delete

Authentication Domain	User ID	Password
DefaultAuth	Andrew.Williams	
HDINSIGHT	andrew.williams@vanquisbank.co.uk	*********

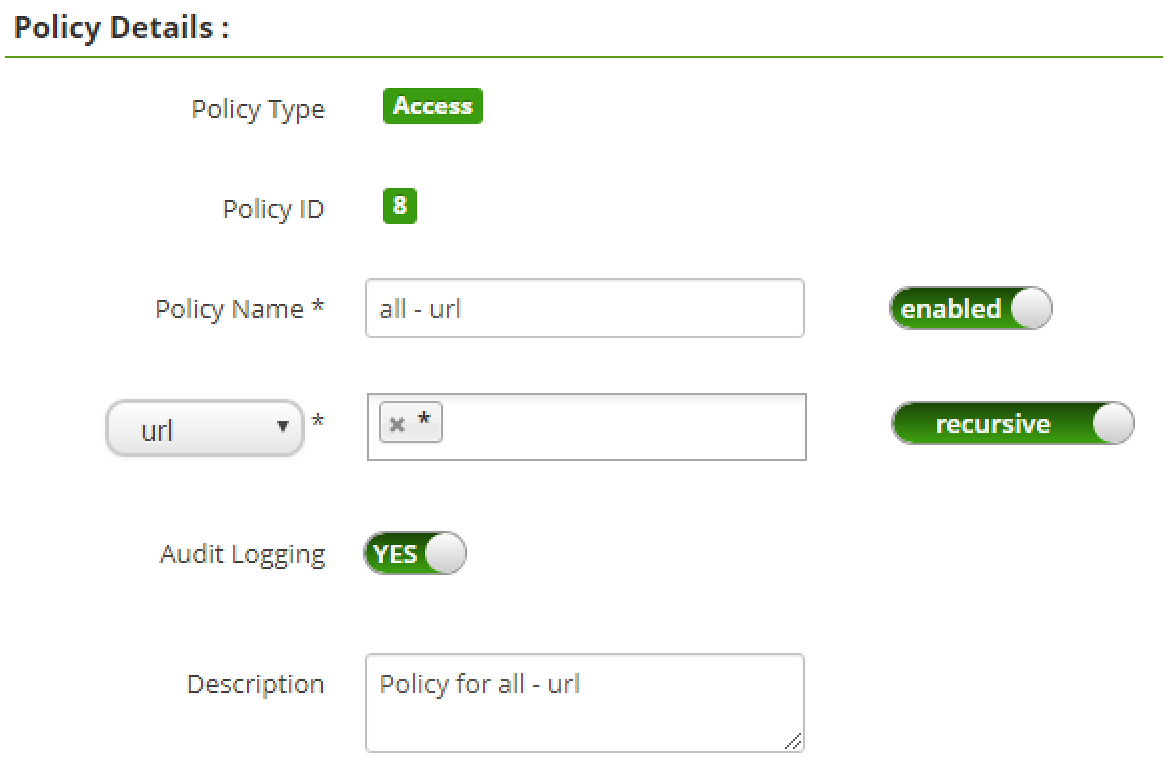
Display 1. User Properties in SAS Management Console

Notice that the User ID is the email address. This is because when logging into Ambari in HDInsight, the email address is used as the login.

Ranger Policies

To read and write data to the Azure Data Lake, even if you have logged into Ambari first, there is one Ranger policy which you need to ensure that all users are added to. This allows the calls out of HDInsight to the Azure Data Lake.

The policy is set up by default when you start your cluster, but it will only include local users and you need to allow your AD users to have 'read' and 'execute' permissions on this. See below for an example of this policy.



Policy Details :

Policy Type	Access	
Policy ID	8	
Policy Name *	all - url	enabled
url *	*	recursive
Audit Logging	YES	
Description	Policy for all - url	

Display 2. Ranger Policy for URL

Hive Transport Mode

This is more of a generic issue with Hadoop but it is worth knowing that when you come to connect to HDInsight, it may not work out of the box for some users. This is due to the number of Active Directory groups that the user is a member of.

There are two solutions for this. You can either change the transport mode from HTTP to BINARY as per the following setting:

hive.server2.transport.mode

Or, if you wish to continue using HTTP, you will need to increase the header size of the requests by adding the following two parameters into the Hive configuration in Ambari:

hive.server2.thrift.http.request.header.size = 32768

hive.server2.thrift.http.response.header.size = 32768

Kerberos TGT ERROR with Tez

One issue discovered was that queries may randomly fail when executed from any of Hive UI, Beeline, or SAS, against the Tez engine. The error reported is that a Kerberos tgt could not be found. This issue only occurs with HDInsight clusters with the Azure Data Lake Gen2 filesystem.

The error returned would look similar to the below:

```
killed/failed due to:OTHER_VERTEX_FAILURE]DAG did not succeed due to
VERTEX_FAILURE. failedVertices:1 killedVertices:1: Caused by:
       org.apache.hadoop.security.authentication.client.Authentication
Exception: GSSException: No valid credentials provided (Mechanism
level: Failed to find any Kerberos tgt)
```

However, at the time of writing, this has now been fixed with the following workaround that needs to be added in Ambari, into HDFS Custom core-site.xml:

hadoop.security.token.service.use_ip=false

Once applied, restart HDFS and Hive services and the error will disappear.

PERFORMANCE TUNING

Most performance tuning techniques are generic across Hadoop clusters, however there are some key areas to focus on.

OUT OF THE BOX HDINSIGHT CONFIG

Depending on your data structures, and the number of users who will be using the platform, as with any other Hadoop environment you will need to tune your environment, such as heapsizes, and yarn queue configuration, as per other Hadoop clusters. You can call these changes via an Ansible script so that you don't have to do this each time. Recent updates to HDInsight now allow the use of an Ambari Metastore to keep these changes across cluster deletion and creation.

READ_METHOD=JDBC VS HDFS

The method that you will want to choose will depend on how much data that you are reading from HDInsight. A lot of our users like to click open tables to view the data in tools such as Enterprise Guide and SAS Studio. Clicking on the table to view the data calls a Hive query, which generates the results, as there is only a small amount of data. Read_Method=jdbc is the best method to use here.

However if you have batch jobs, which pull a significant amount of data down to SAS such as 2GB+ then I recommend using Read_Method=HDFS as the data is pulled significantly faster after the Hive query is executed.

SIZE OF FILES ON THE DATA LAKE (256MB - 100GB)

If you are using Hive tables then the way in which these tables are created on the Data Lake is very important. For example, we had a process which writes applications in ORC formats to the Data Lake, these then end up consisting of thousands of files to build the final table. You will soon discover that queries take hours as opposed to minutes. Even partitioning the table will have little effect.

A best practice when querying via HDInsight is to ensure that files on the Datalake are between 256mb and 100GB in size, otherwise significant performance issues can be observed.

SAS VIYA TO ADLS GEN2

A new feature of SAS VIYA 3.5 is that it now has the ability to talk directly to the ADLS Gen2, instead of having to run queries via Hive to access ORC files listed here.

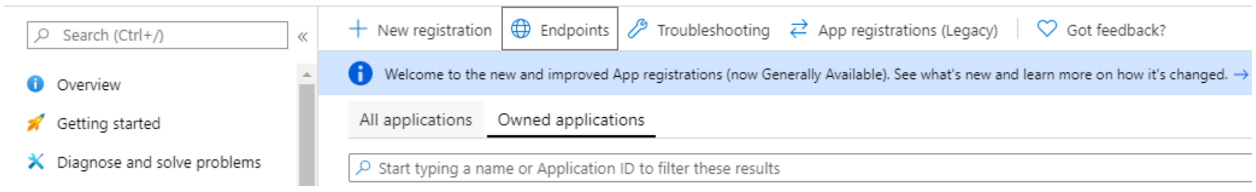
To do this you need to create an Azure Application. This is to access the Data Lake, similar to when we go via HDInsight, you need to generate an OAuth token to have permission to read and write data to it.

When creating the Azure Application, you will need an Azure Administrator to grant delegated access, in order to have full access to the Azure Data Lake Service, Access Azure Storage and Sign In and Read User Profile permissions.

Most of this is documented, the issue we found was that the Azure Application requires a Redirect URI, however it is not obvious what the redirect URI should be.

You should redirect to use the Azure Redirect URI's for the oauth2/v2.0/authorize

These can be found under App Registrations -> Endpoints:



Welcome to the new and improved App registrations (now Generally Available). See what's new and learn more on how it's changed. →

Figure 4. Azure Endpoints

Once the application is setup, to access the Data Lake you will need to setup a caslib. An example code is below. Application ID and Tenant ID can be found from the Azure Application created.

```
cas casauto sessopts=(azureTenantId='<Tenant ID>');

caslib azds datasource=(
      srctype="adls"
      accountname="<ADLSGEN2ACCOUNTNAME>"
      filesystem="<ADLSGEN2CONTAINER>"
      applicationId="<Application ID>"
   )
     subdirs;

proc casutil;
   list files;
quit;
```

As the authorization will grant access to the end user then you will need to apply permissions to the storage container and the filesystem to allow users to read and write to the ADLS Gen2 filesystem.

CONCLUSION

As our solution has enabled much more flexibility with the platform, aside from the obvious cost savings from using the Data Lake, we can also create clusters on demand to users. The flexibility of SAS makes it very easy to connect to multiple clusters, so from the end user's point of view, connections can be made to different environments with ease.

Obviously there have been issues encountered along the way, but all issues encountered have been addressed in this paper and all solutions to these published.

Our solution has enabled us to move from a single SAS and Hadoop environment, to multiple SAS and Hadoop environments, to allow proper change control, and a dedicated cluster for production jobs.

With the new features of SAS Viya it is now possible to talk directly to the Azure Data Lake Gen2, as opposed to going via a HDInsight cluster. This will give future flexibility of using a range of different techniques to access the data stored on the Azure Data Lake Gen2.

REFERENCES

- Microsoft "Introduction to Azure Data Lake Storage Gen2". Accessed February 25, 2020. https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-introduction

ACKNOWLEDGMENTS

Thanks to the Engineering Team at Vanquis Bank for working hard to deploy the solution mentioned in this paper. Special thanks to Louis Allen (Infrastructure Engineer) who aided on the infrastructure side and built infrastructure as code for the above solution. Thanks to Rajan Mistry (SAS Administrator) who carried out excellent work on the build and testing of the HDInsight clusters and data migration.

RECOMMENDED READING

- *Optimize Azure Data Lake Storage Gen2 for performance*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Andrew Williams
Vanquis Bank Ltd
Andrew.williams@vanquisbank.co.uk