Image courtesy: <https://resources.snowflake.com/solution-briefs/snowflake-cloud-data-platform-solution-brief>

Building Snowpipe on Azure Blob Storage Using Azure Portal Web UI for Snowflake Data Warehouse

[Christopher Tao](https://medium.com/@qiuyujx?source=post_page-----f0cdd7997250----------------------)

Follow

[Dec 14](https://medium.com/@qiuyujx/building-snowpipe-on-azure-blob-storage-using-azure-portal-web-ui-for-snowflake-data-warehouse-f0cdd7997250?source=post_page-----f0cdd7997250----------------------) · 10 min read

Snowpipe is a built-in data ingestion mechanism of Snowflake Data Warehouse. It is able to monitor and automatically pick-up flat files from cloud storage (e.g. Azure Blob Storage, Amazon S3) and use “COPY INTO” SQL command to load the data into a Snowflake table.

In the official documentation, you’ll find a nice tutorial:

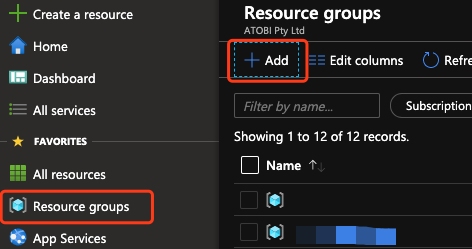
*Automating Snowpipe for Azure Blob Storage*[*https://docs.snowflake.net/manuals/user-guide/data-load-snowpipe-auto-azure.html*](https://docs.snowflake.net/manuals/user-guide/data-load-snowpipe-auto-azure.html)

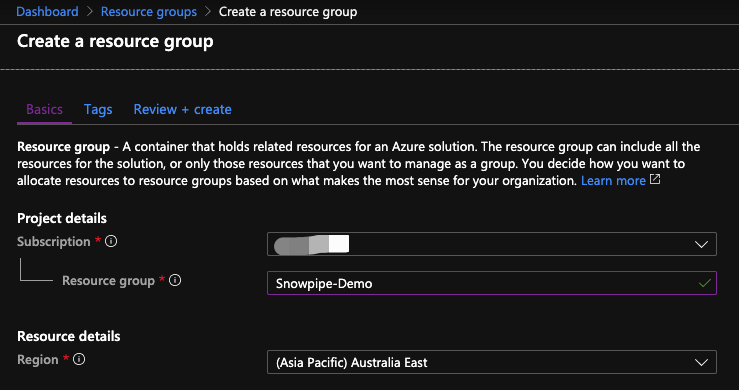
However, this article uses Azure CLI (command line interface) to build the cloud storage component and the Azure Event Subscription. In my opinion, this is not intuitive enough especially for new users and the people who do not have enough knowledge of Azure. Even though they can build the Snowpipe following these commands, it is still might be agnostics for the overall architecture and data flow. In other words, this tutorial tells you how to build a Snowpipe, but difficult to let you understand it.

In this tutorial, I’ll introduce how to build a Snowpipe using the Azure Portal, which is the Web-based UI of Azure, which I believe will give you better intuition about how the Snowpipe works.

**Create Storage Accounts**

To begin with, let’s create a resource group to organise the storage accounts that are going to be built for Snowpipe. On your Azure Portal, navigate to **Resource groups**and click **Add**. Then, input name for this resource group. You may choose a region that is identical/close to your Snowflake region for the best performance.



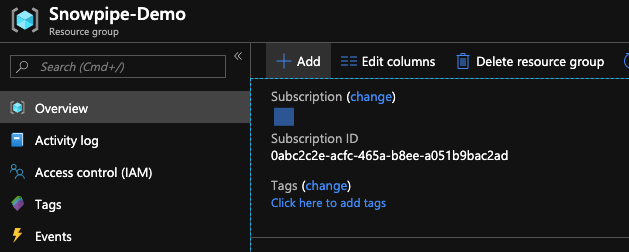


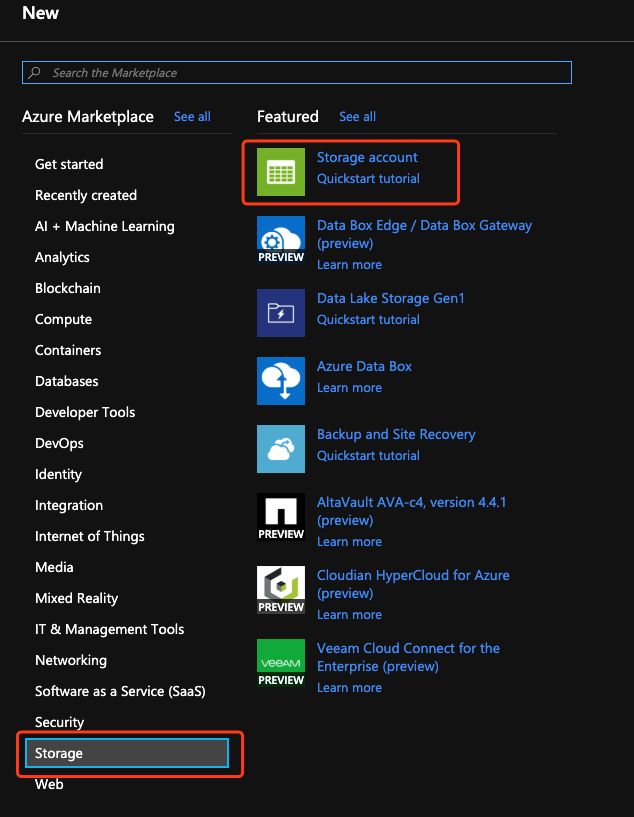
Then, let’s create the storage account for staging and event queuing messages.

In the official tutorial, there are two storage accounts created, one blob storage for staging the flat files, and another storage account with storage queue to facilitate the Event subscription.

In fact, I found that actually one Azure Storage Account with both Blob Container and Storage Queue is enough. Let’s create such a storage account.

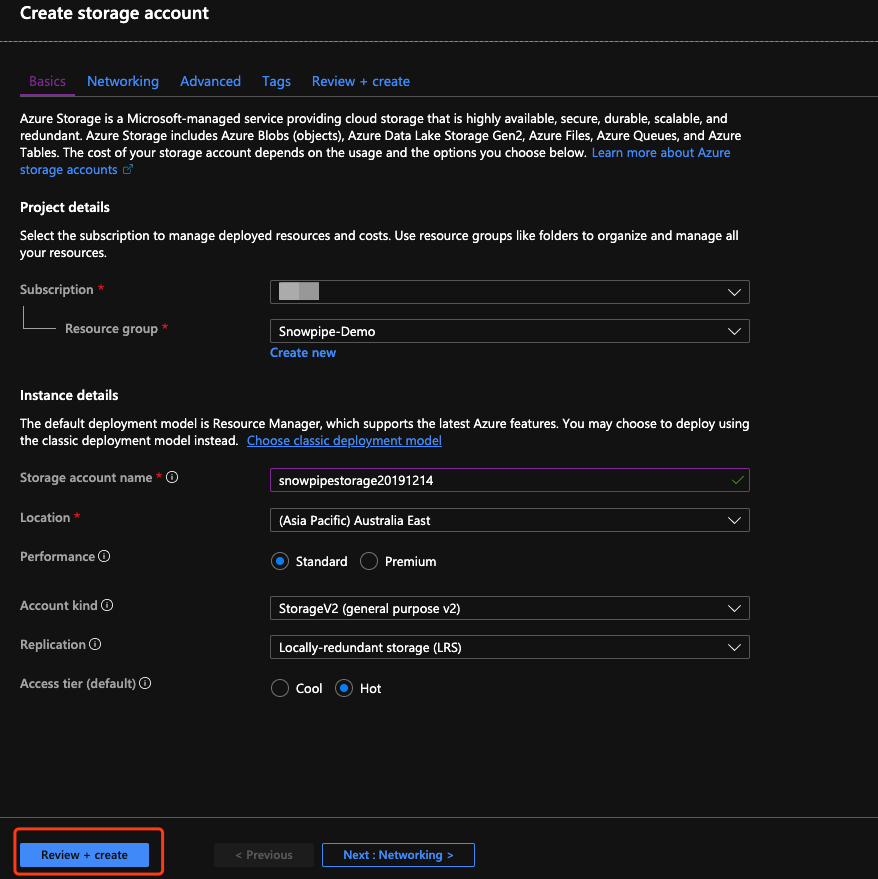
In the resource group we’ve just created, click the **Add** button to create a resource. In the Azure Marketplace, choose **Storage** category and in the “Featured” list choose **Storage Account**.





Note that the name of the storage account should be unique globally, so I added the date as the suffix. This is because every storage account has a DNS CName can be accessed ([https://<name>.queue.core.windows.net](https://storagesfqueuedemo.queue.core.windows.net/)).

Make sure you choose “**StorageV2**” for the **Account kind** to have both **Blob Container** and **Storage Queue**. Then, click **Review + create** to validate and create the storage account.

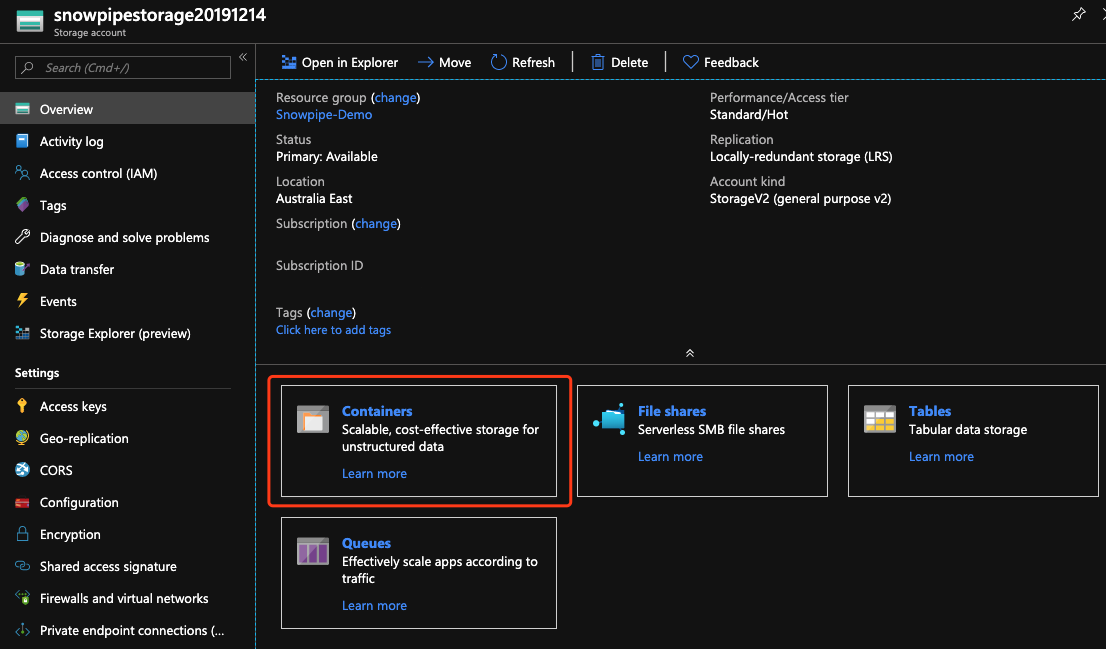


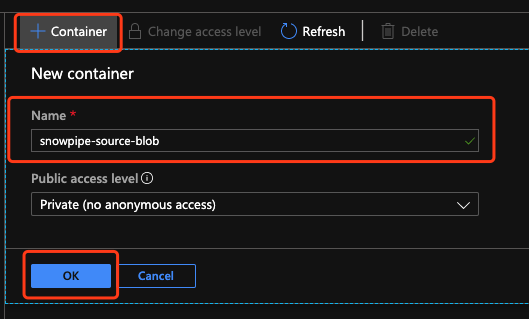
**Prepare the Storage Account**

Now we have a storage account. But before we can utilise it, we need to create:

* A Blob Container for the staged files that will be loaded by Snowpipe
* A Storage Queue as the endpoint of the Event subscription

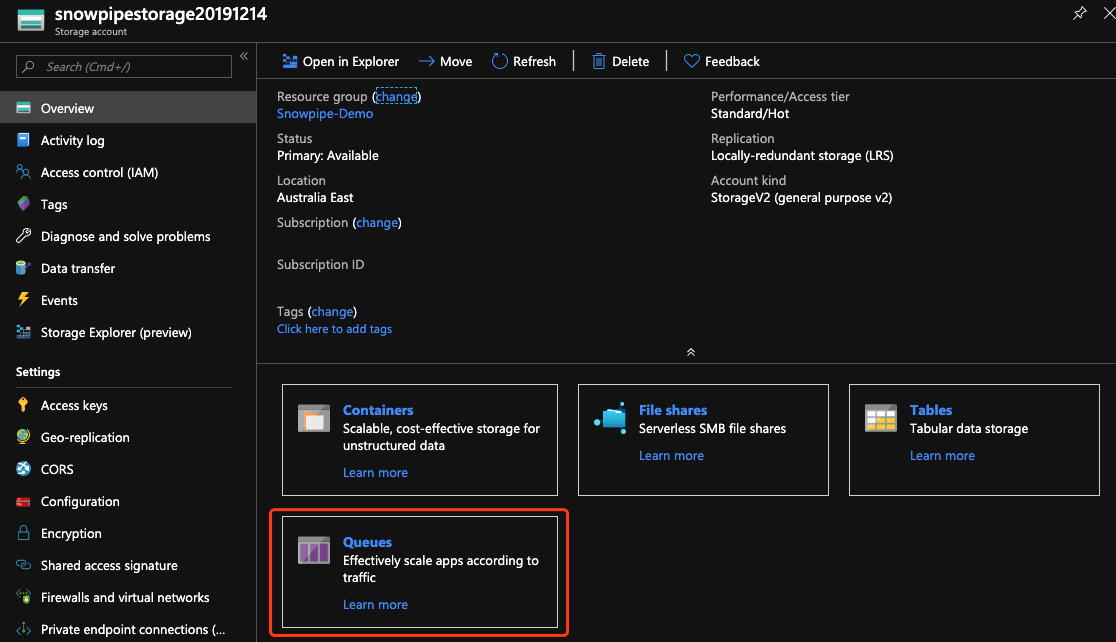
To create the Blob Container, click the **Containers** button, then click **+Container**. Give a name to this container whichever makes sense to you, and then click **OK** to create it.

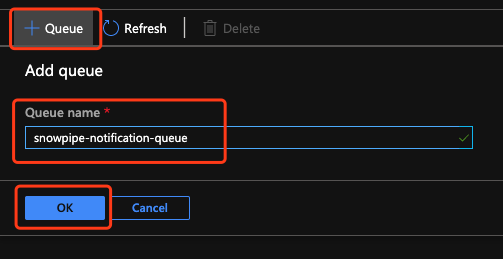




We’ve got a Blob Container now. Let’s create the Storage Queue.

Go back to the Storage Account Overview page. Click Queues to create a Storage Queue.

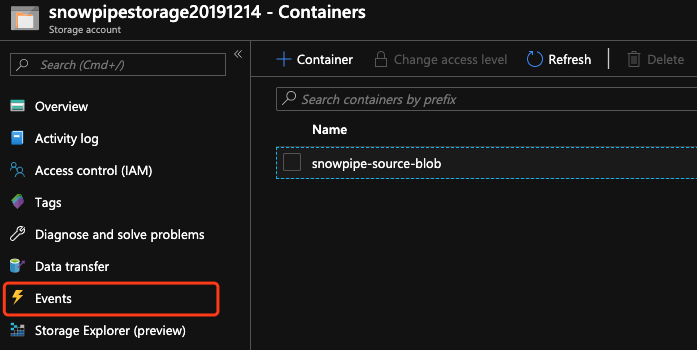




**Create Event Grid Subscription**

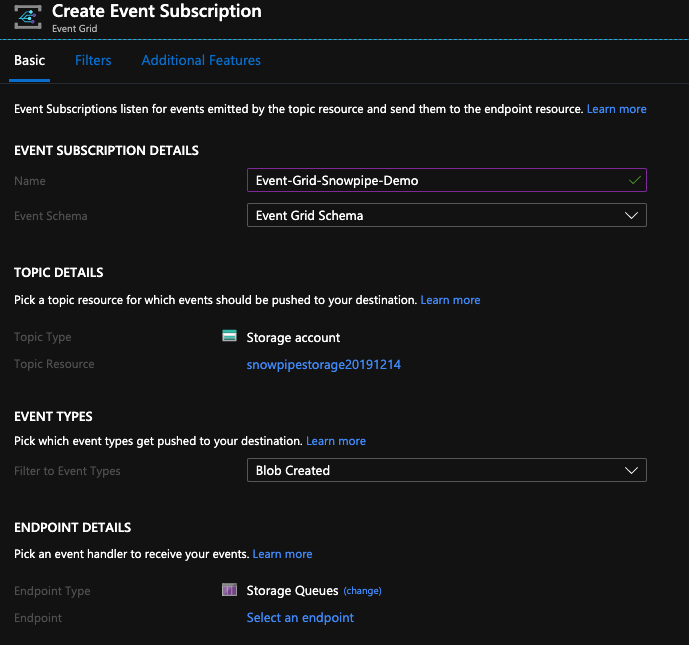
Next, let’s create the Event Grid Subscription for the Blob Container, and set the endpoint to the Storage Queue.

Navigate to the **Storage Account** -> **Containers**, click **Events** tab.



There should not be any Event Subscriptions there. Click **+ Event Subscription** to create one.

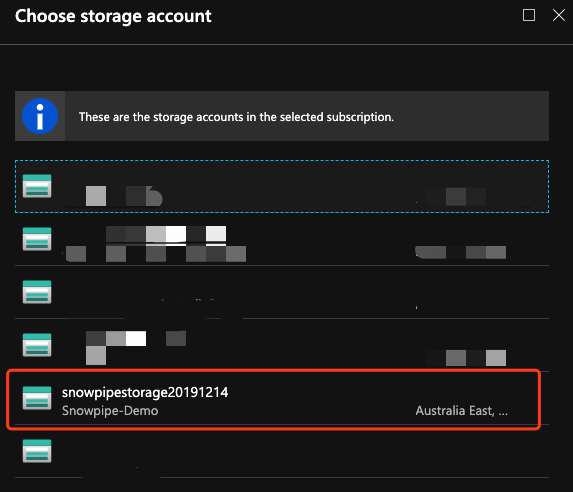
Give a name to the Event Subscription. Choose the default **Event Grid Schema**. For the **Event Types**, make sure you selected **Blob Created**, which is what we want for Snowpipe. Then, choose **Storage Queues** as the **Endpoint Type**.



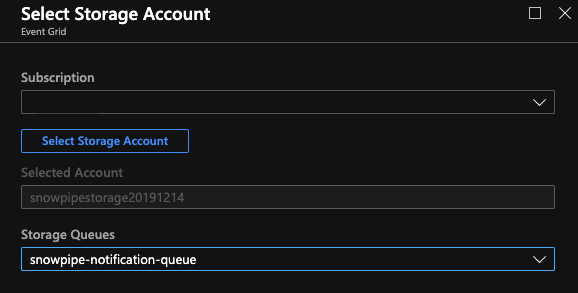
Don’t forget to choose a Storage Queue. Click **Select an endpoint** link that is under the **Storage Queues** you’ve just selected, a new blade navigation window should popup on the right. Firstly, select your subscription, your current active subscription should already be there by default. Click **Select Storage Account**.



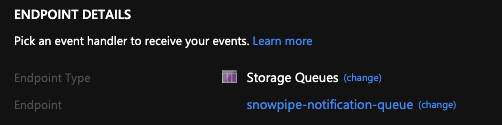
Another blade window popup letting you choose the storage account, so choose the one we’ve created.



In the final blade window, choose the storage queue we created for this, and then click **Confirm Selection**.



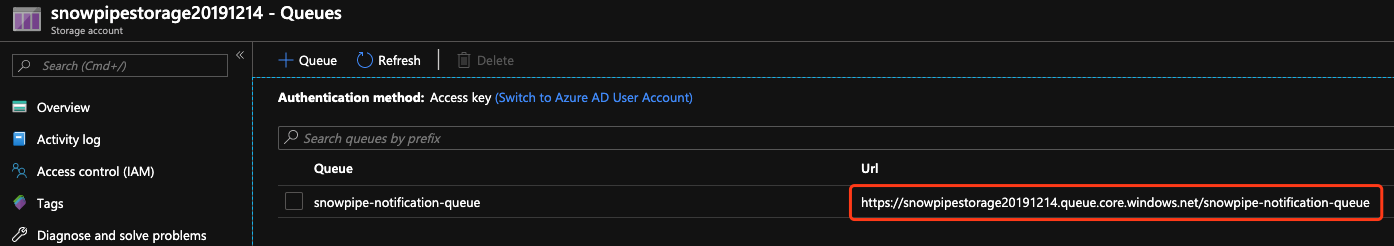
Now, you should see that the storage queue is displaying as follows.



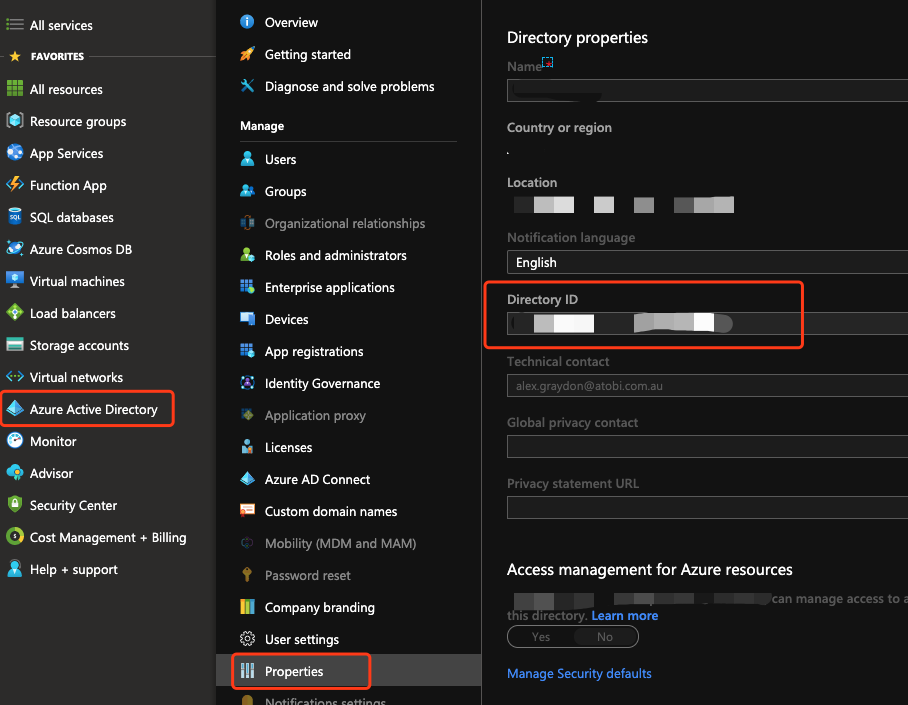
**Create an Integration in Snowflake**

After that, we need to create an **Integration** in Snowflake. Let’s firstly record some information from Azure that are needed for the integration.

Firstly, note down the URL of the storage queue. Navigate to the Storage Queue, you’ll be able to get the URL along with the Queue name as follows:



Then, go to **Azure Active Directory** -> **Properties** to get the **Directory ID** which will be used for Snowflake to access your Azure Subscription later.



To create an Integration in Snowflake, you’ll need to be Account Admin. Run the below SQL command.

create notification integration SNOWPIPE\_DEMO\_EVENT  
 enabled = true  
 type = queue  
 notification\_provider = azure\_storage\_queue  
 azure\_storage\_queue\_primary\_uri = '[<your\_storage\_queue\_url>'](https://storagesfqueuedemo.queue.core.windows.net/sfeventqueue')  
 azure\_tenant\_id = '<your\_directory\_id>';

Note: It is highly recommended to use ALL UPPER CASES when defining the Integration names, to avoid case sensitivity in some Snowflake scenarios.

*Tips: Once done, you can run show integrations; command to retrieve all the Integrations in your Snowflake account;*

**Authentication of Snowflake Application to Azure**

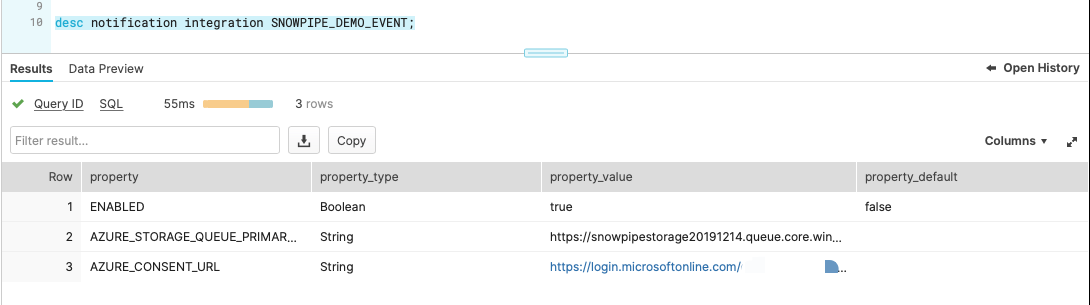
Now, Snowflake knows where to go to Azure to get the notification (Azure Events), but we still need to let Azure authenticate our Snowflake account. By default, for security purpose, we should never let our Azure Storage Account for staging our data to be accessible publicly, of course.

Let’s firstly run the following SQL command in Snowflake:

DESC notification integration SNOWPIPE\_DEMO\_EVENT;

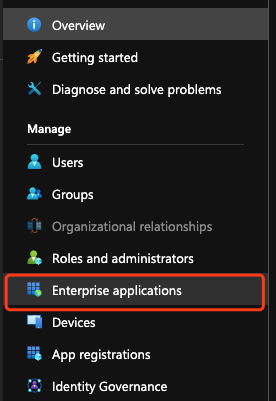
In the result pane, you’ll see the AZURE\_CONSENT\_URL property, and the login URL is accessible on the property\_value column. Copy and paste it in your browser to login to your Azure account as you usually do.

Permission granting notice should be given by Azure, click **Accept** button.



Once done, you should be redirected to Snowflake official website. Now you can close this web page.

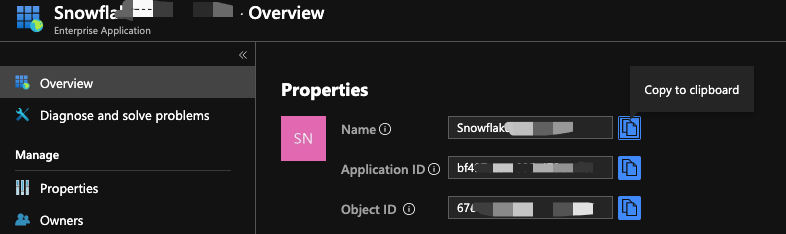
Now, go to your **Azure Active Directory** again, click **Enterprise applications** tab.



Scroll to the bottom, you will see the Snowflake application Name.

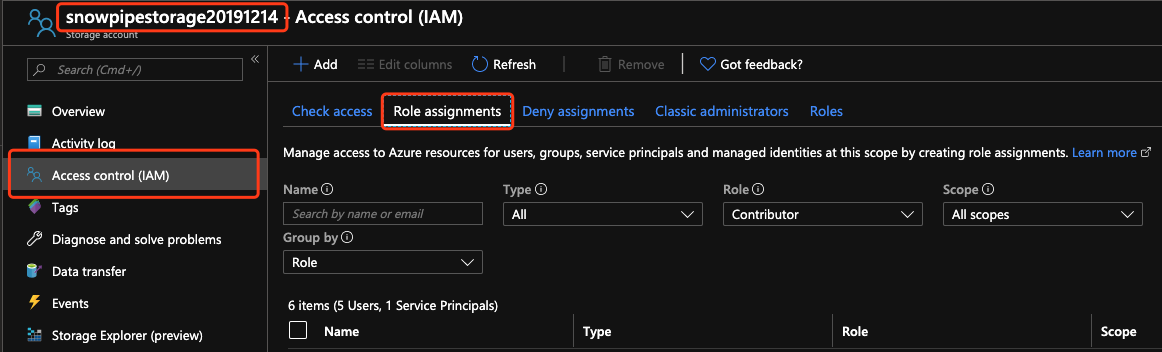
https://miro.medium.com/max/624/1*fykduWiEHTOtUCrbirzzOw.png

Click the name will let you go to the details of the Snowflake application integration in details. Record the **Name**appears in the Properties for later usage.

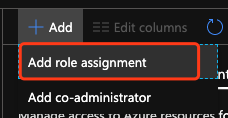


Now, we need to grant Snowflake with access to the Storage Queue. otherwise, it won’t get the event message.

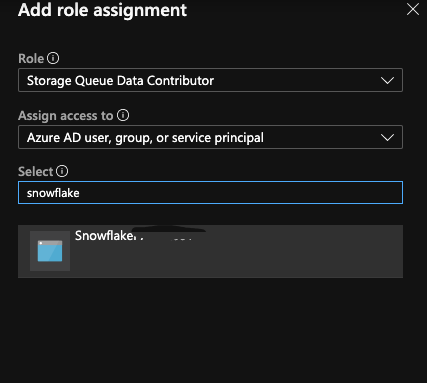
Navigate to the **Storage Account** -> **Access control (IAM)** -> **Role assignments**. Snowflake is still not there, so we’ll need to add the role assignment.



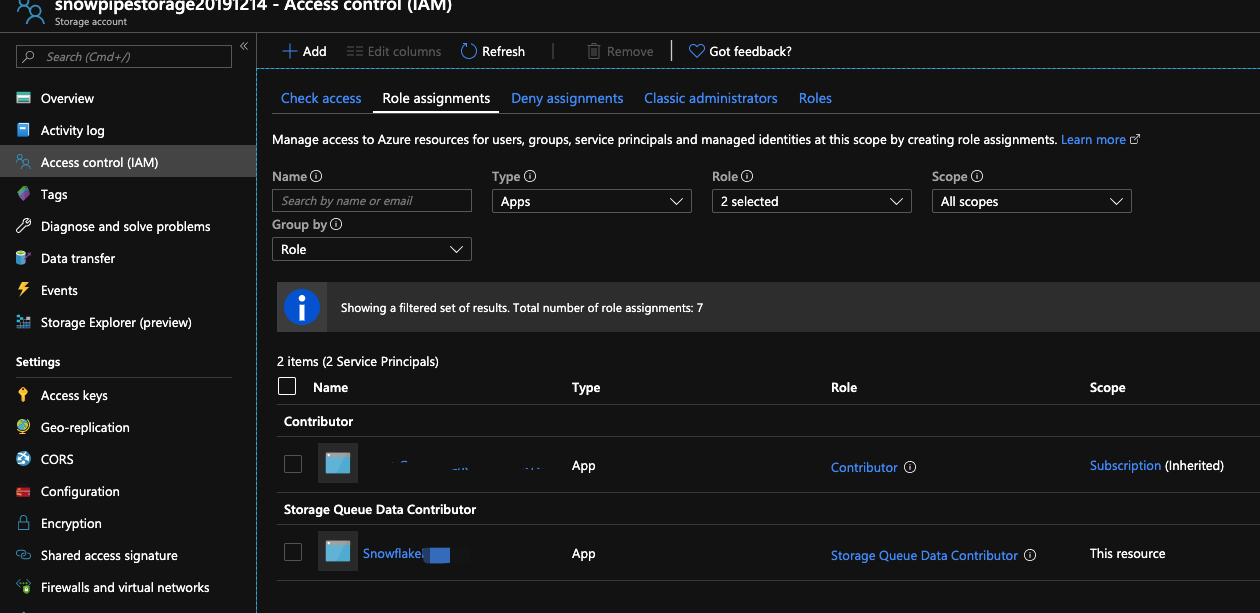
Click **Add** button on the top, and then select **Add role assignment**.



In the pop-up blade window, choose **Storage Queue Data Contributor** as the role, as we don’t want to grant unnecessary larger permission to it. Then, type in “snowflake” to search in the **Select** input field, the Snowflake application account name should be retrieved. If you have multiple Snowflake accounts, make sure you select the correct one that you recorded in the above section. Then, click **save** button to apply the change.



Now, you can see the Snowflake account int the **Role assignments** list.



**Create a Snowflake Stage**

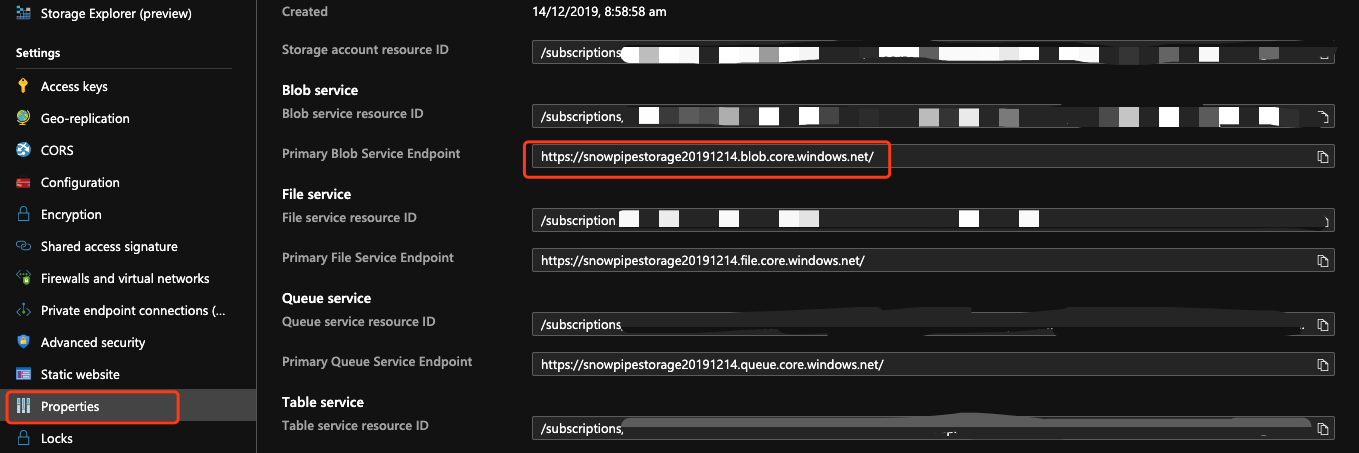
Let’s create a Snowflake stage first. This will be used by the Snowpipe as data source.

Create a Database:

CREATE DATABASE SNOWPIPE\_DEMO;

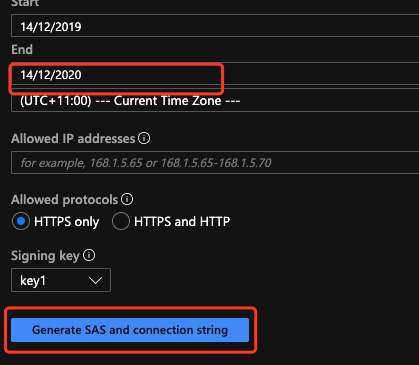
Let’s use the PUBLIC schema because this is just an experiment.

To create a Snowflake external stage, we need to get the Azure Blob Container URL and SAS (Shared Access Signature). Let’s go to the **Storage Account** -> **Properties** to get the URL.



Then, go to the Shared access signature tab to create a SAS token. Here, I would like to change the year the 1 year after, so the token will be valid for one year. You may want to use a different end date time to based on your requirements. After that, click **Generate SAS and connection string**.

https://miro.medium.com/max/30/1*Y_PSYDoRHGPu-MA1W6EK-A.png?q=20



The SAS URL should be shown as follows. Please note that this is an URL, the token should be the string starts from the question mark “?”

https://miro.medium.com/max/30/1*4FpmkgZc4pBdOURS-6QHXA.png?q=20



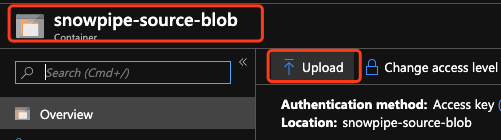
The completed SQL command is shown as follows (Please note that this is an example, you will need to replace the url and credentials with yours):

CREATE OR REPLACE STAGE "SNOWPIPE\_DEMO"."PUBLIC"."DEMO\_STAGE"  
 url = 'azure://snowpipestorage20191214.blob.core.windows.net/snowpipe-source-blob/'  
 credentials = (azure\_sas\_token=  
 '?sv=2019-02-02&ss=bfqt&srt=sco&sp=rwdlacup&se=2020-12-14T18:44:03Z&st=2019-12-14T10:44:03Z&spr=https&sig=ZPN2qcMw64k44td90gSMzvC7yZuQjnQZZCD2xAUS25Y%3D'  
 );

Let’s test the connection between Snowflake and the Blob Container. Open your text editor and write the following content, and then save the file as emp.csv

1,Chris  
2,John  
3,Jade

Upload to the Blob Container.



Then, run the following command in Snowflake. You should be able to see the file in the result pane.

ls @"SNOWPIPE\_DEMO"."PUBLIC"."DEMO\_STAGE";

**Create Snowpipe**

Now we reach the final stage, creating the Snowpipe. Let’s firstly create a table as the destination of the Snowpipe.

CREATE OR REPLACE TABLE "SNOWPIPE\_DEMO"."PUBLIC"."EMPLOYEE" (  
 id STRING,  
 name STRING  
);

Then, create the Snowpipe using the following SQL command

CREATE OR REPLACE pipe "SNOWPIPE\_DEMO"."PUBLIC"."DEMO\_PIPE"  
 auto\_ingest = true  
 integration = 'SNOWPIPE\_DEMO\_EVENT'  
 as  
 copy into "SNOWPIPE\_DEMO"."PUBLIC"."EMPLOYEE"  
 from @"SNOWPIPE\_DEMO"."PUBLIC"."DEMO\_STAGE"  
 file\_format = (type = 'CSV');

Now, you’ll be able to show the pipe that you’ve just created.

SHOW PIPES;

**Snowpipe Load Existing Files**

The emp.csv file had been there in the Blob Container before we created the Snowpipe. So, it won’t automatically load the file, since it relies on the “Blob Created” Event to trigger itself, but that event happened before it was created.

We do have an approach to load the existing files in the Blob Container. Using the command below will manually refresh the Snowpipe to let it loading any files existing in the Blob Container but haven’t been loaded.

ALTER PIPE "SNOWPIPE\_DEMO"."PUBLIC"."DEMO\_PIPE" REFRESH;

Once you run this command, the result pane shows as follows, which means that the loading request has been sent to the emp.csv file:

https://miro.medium.com/max/30/1*CdxD5EusnTenJOQJ11pGgA.png?q=20



Now, we can ran SELECT \* FROM EMPLOYEE and you’ll see the rows loaded from the file.



**Test the Snowpipe**

Finally, let’s test the Snowpipe by uploading a new file to the Blob Container.

Create the emp\_2.csv file as follows, and upload it to the snowpipe-source-blob Blob Container.

4, Alex  
5, Olivier  
6, Frank

Wait for a while, retrieve the EMPLOYEE table again, the new records are loaded!



**Future Works**

This tutorial has covered how to create a Snowpipe and focusing on “utilising Azure Portal” rather than using Azure CLI like the Snowflake official documentation does.

If you’re interested in what is the best practice of building an automated data pipeline in Snowflake, from the data ingestion to the production databases, please keep an eye on the next post in this topic. It will cover Snowflake Streams and Tasks.