Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2024, 11(9):47-50

Research Article ISSN: 2394 - 658X

Different ways to connect DFS windows folder & Apache Airflow

Rameshbabu Lakshmanasamy

Senior Data Engineer, Jewelers Mutual Group

ABSTRACT

There are always circumstances to establish a connection to a DFS windows network folder from airflow for file transfer/movement across various systems/applications. This one specific use case has always been challenging when we would develop a pipeline/process to source files or to write files into the DFS folder. There are operator challenges, network challenges, firewall config challenges depending on the policies by the internal network and sysadmin teams. In this article, we are going to discuss ways to create that pipeline between the airflow and dfs folder, either to read from or write into.

Keywords: Apache Airflow, Orchestration tool, DFS, Samba, SMB, CustomDFSOperator

INTRODUCTION

Though extensive search for an appropriate built in ready airflow operator that is exclusive to the use case of establishing a handshake to DFS folder to airflow yields not much results, there are few options we have that we will be discussing in this paper. This can be accomplished through the following ways –

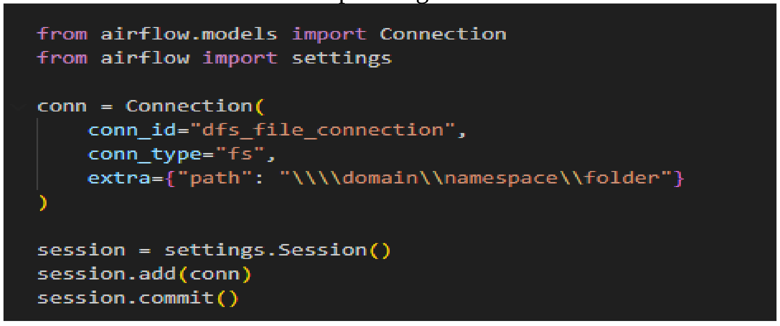
(1) Direct File(Path) Configuration
(2) SSH Connection
(3) Custom Operator for DFS specific operations
(4) Samba provided public operator GCSToSambaOperator (uses SMB protocol)
(5) HTTP/API Connection (in case of DFS exposed with an API)

Background:

In any IT organization where numerous systems/applications exist, and there is a constant need for file movements from one system to another system. Sourcing file from S3, Reading file from GCS, Writing files to S3 or GCS, Reading or Writing files into DFS folders are different use cases to name a few. For other than DFS folder related file access, rest of the options have numerous ways with public operators in the market. However, coming to the DFS access, we always faced issue on how to implement this pipeline to read or write files into one. Here in this article let us discuss few of the options we got to accomplish this.

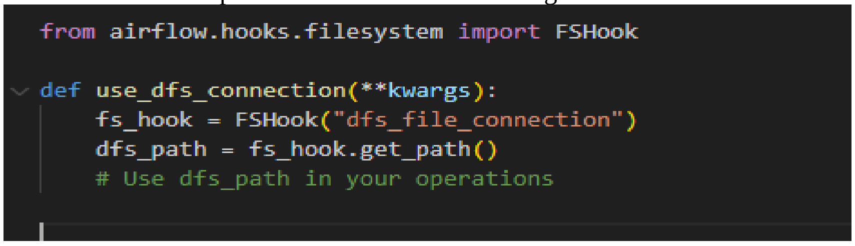
(1) Direct File(Path) Configuration

This simplest method treats the DFS share as a local file system. Create a airflow connection – either directly in UI or via code as below. Here full path is given for the network share folder.



```
from airflow.models import Connection
from airflow import settings

conn = Connection(
    conn_id="dfs_file_connection",
    conn_type="fs",
    extra={"path": "\\\\domain\\namespace\\folder"}
)

session = settings.Session()
session.add(conn)
session.commit()
```

Access the network path with the connection configured like above.

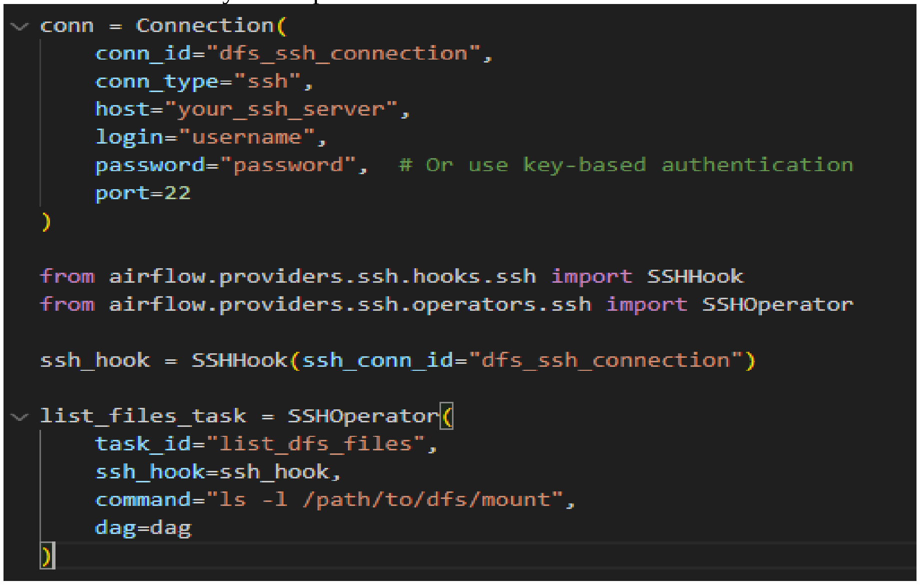


```
from airflow.hooks.filesystem import FSHook

def use_dfs_connection(**kwargs):
    fs_hook = FSHook("dfs_file_connection")
    dfs_path = fs_hook.get_path()
    # Use dfs_path in your operations
```

Again, in practical scenarios, there may be access issues to the folder, and network team would need to be involved for firewall clearance and access credentials if any are needed.

(2) SSH Connection (when DFS accessible via SSH)

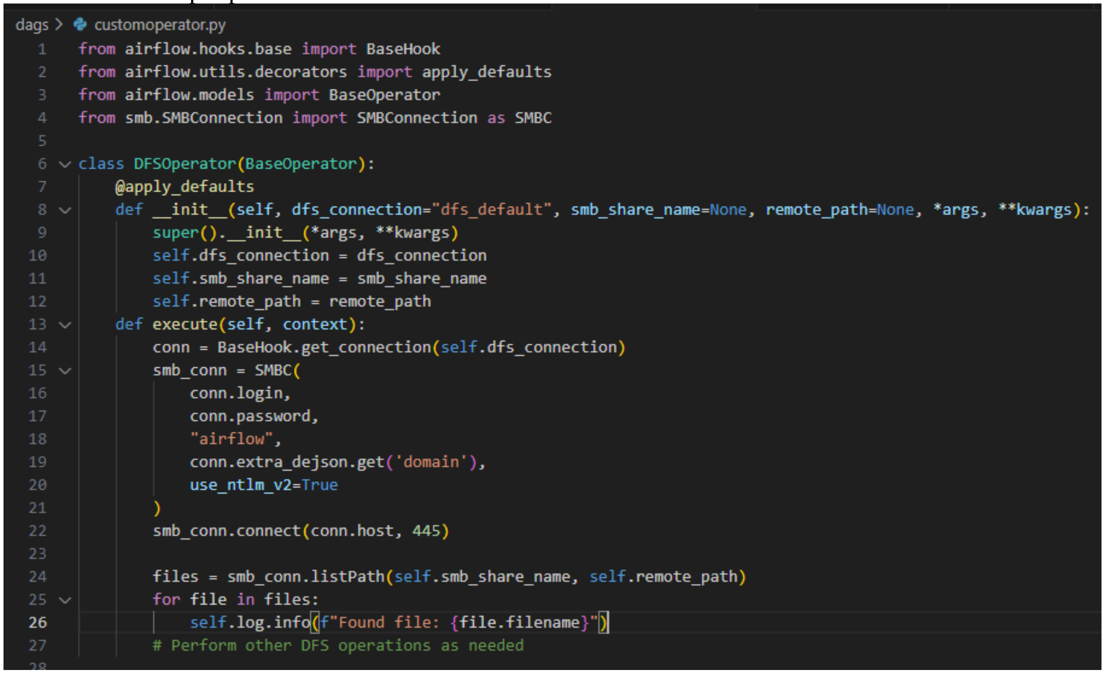
Below is another way to setup the DAG to establish a DFS connection .



```
conn = Connection(
    conn_id="dfs_ssh_connection",
    conn_type="ssh",
    host="your_ssh_server",
    login="username",
    password="password",  # Or use key-based authentication
    port=22
)

from airflow.providers.ssh.hooks.ssh import SSHHook
from airflow.providers.ssh.operators.ssh import SSHOperator

ssh_hook = SSHHook(ssh_conn_id="dfs_ssh_connection")

list_files_task = SSHOperator(
    task_id="list_dfs_files",
    ssh_hook=ssh_hook,
    command="ls -l /path/to/dfs/mount",
    dag=dag
)
```

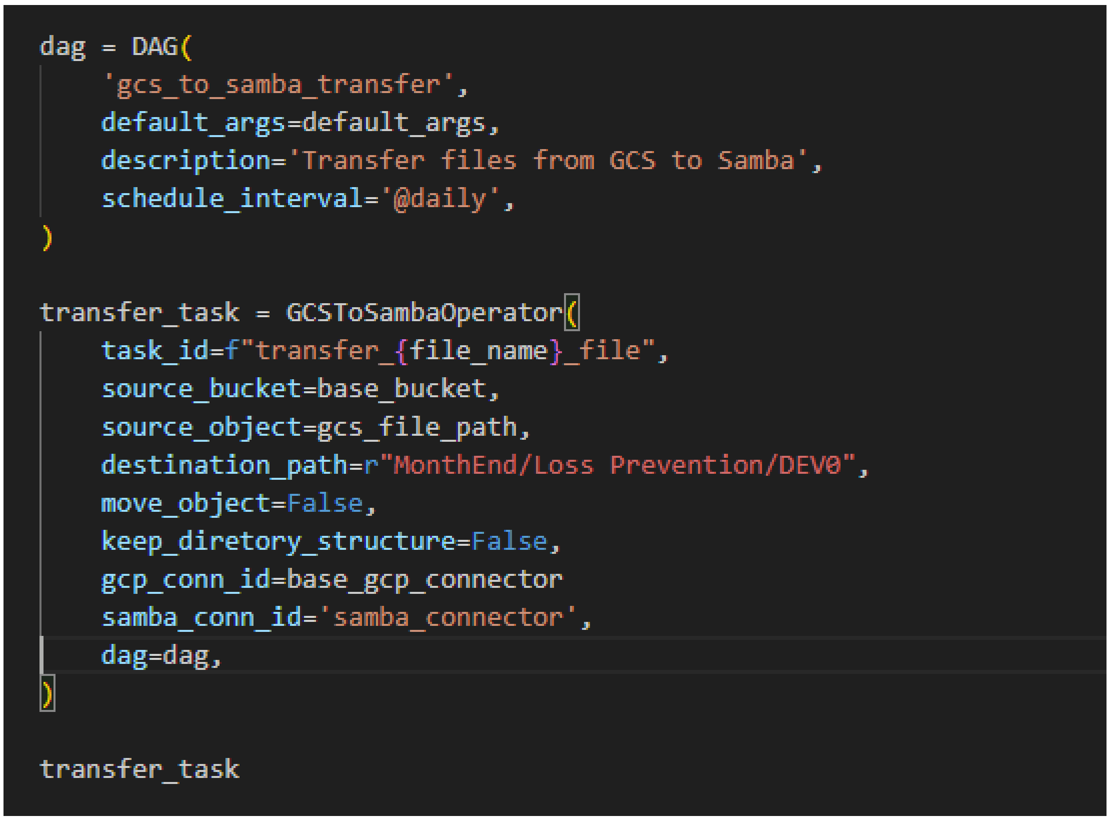
(3) Custom Operator for complex ops:

If more flexibility is needed but is okay to take that extra implementation effort, custom operators are the way to go. This helps with more complex operations customized to our own system/applications or more specific DFS interactions. Sample pseudo code below –



```
from airflow.hooks.base import BaseHook
from airflow.utils.decorators import apply_defaults
from airflow.models import BaseOperator
from smb.SMBConnection import SMBConnection as SMBC

class DFSOperator(BaseOperator):
    @apply_defaults
    def __init__(self, dfs_connection="dfs_default", smb_share_name=None, remote_path=None, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.dfs_connection = dfs_connection
        self.smb_share_name = smb_share_name
        self.remote_path = remote_path
    def execute(self, context):
        conn = BaseHook.get_connection(self.dfs_connection)
        smb_conn = SMBC(
            conn.login,
            conn.password,
            "airflow",
            conn.extra_dejson.get('domain'),
            use_ntlm_v2=True
        )
        smb_conn.connect(conn.host, 445)

        files = smb_conn.listPath(self.smb_share_name, self.remote_path)
        for file in files:
            self.log.info(f"Found file: {file.filename}")
        # Perform other DFS operations as needed
```

(4) Operator from Samba (utilizing SMB protocol) if GCS is involved:
Here is the definition of SAMBA from samba.org page itself – A Standard Windows interoperability suite of programs for Linux and unix. Samba has provided secure, stable, and fast file transfer services using the SMB/CIFS protocol, such as all versions of windows, linux and many others.
In simple terms, Samba is like a universal adapter for file sharing. It allows applications/systems with different OS's seamlessly interact each other as if they are of same family. It simply helps breakdown the language barriers, and help seamlessly handshake thereby making file-sharing a breeze in mixed-OS situations. In essence, it is just a friendly translator.
In this model, we will leverage GCSToSambaOperator – which is a part of official Apache Airflow providers package, designed to transfer files from GCS to a samba share. Below example details on moving file from GCS bucket into a Windows network DFS folder.
Key parameters needed for this Operator are –
source_bucket – GCS bucket where files needs to be read/located
source_object – Object name with file path in GCS bucket
destination_path – The destination path of the samba share
samba_conn_id – The connection ID for samba connection. The windows host, user access credentials goes in here.
gcp_conn_id – The connection ID for the Google Cloud connection goes in here.
Basically, the functionality goes like this – connects to the gcs bucket specified, download the file in GCS bucket, connects to Samba share using connection details provided, and uploads the file to the specified location on the Samba share (samba_conn_id).
<u>Some practical issues to be considered:</u>
SMB servers usually returns shortnames like "jmidfs01" instead of FQDN (Fully qualified domain names) like "jmidfs01.subdomainnw.local". This DNS resolution failures, would need to be configured with the help of internal network teams. This might occur even if we try to use the direct IP addresses instead of hostnames.
For long term scalability, its better the network team implements a DNS peering between hosted zones involved.

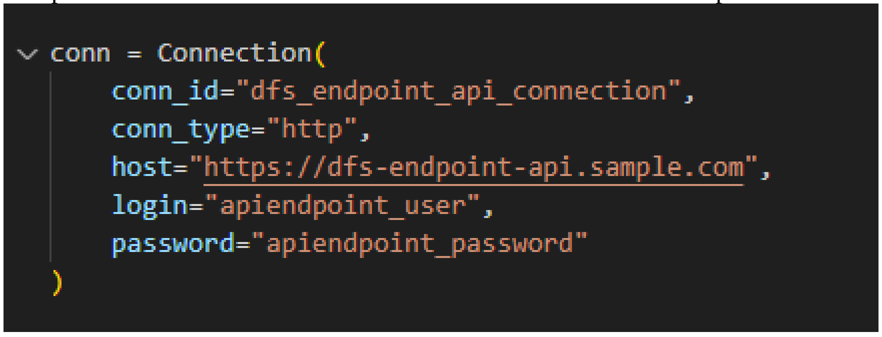


```
dag = DAG(
    'gcs_to_samba_transfer',
    default_args=default_args,
    description='Transfer files from GCS to Samba',
    schedule_interval='@daily',
)

transfer_task = GCSToSambaOperator(
    task_id=f"transfer_{file_name}_file",
    source_bucket=base_bucket,
    source_object=gcs_file_path,
    destination_path=r"MonthEnd/Loss Prevention/DEV0",
    move_object=False,
    keep_diretory_structure=False,
    gcp_conn_id=base_gcp_connector
    samba_conn_id='samba_connector',
    dag=dag,
)

transfer_task
```

(5) HTTP/API Connection

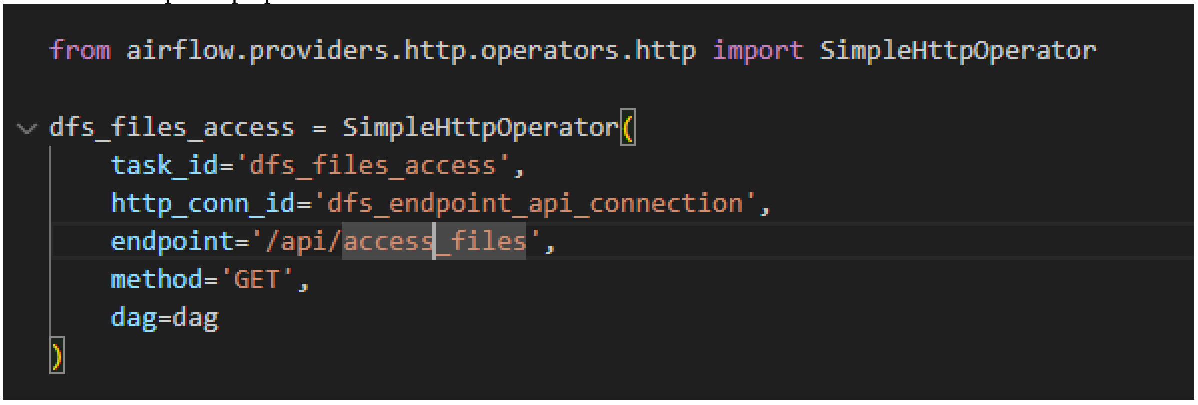
If the DFS System exploses an API endpoint, we can leverage the Airflow HTTP operators and connections. Example pseudo code snippet below –

Setup the connection either in Airflow UI or via code with below specs



```
conn = Connection(
        conn_id="dfs_endpoint_api_connection",
        conn_type="http",
        host="https://dfs-endpoint-api.sample.com",
        login="apiendpoint_user",
        password="apiendpoint_password"
)
```

Utilize the SimpleHttpOperator like below –



```
from airflow.providers.http.operators.http import SimpleHttpOperator

dfs_files_access = SimpleHttpOperator(
        task_id='dfs_files_access',
        http_conn_id='dfs_endpoint_api_connection',
        endpoint='/api/access_files',
        method='GET',
        dag=dag
)
```

CONCLUSION

Above are some of the options to create a successful connection to a DFS network folder. In real life practical scenarios, it mostly involves the internal IT network or sysadmin teams as well for firewall clearance, access privileges or DNS configuration related activities. Some org might have the DFS exposed with a REST API endpoint, which can make use of the HttpOperators. In use cases involving GCS buckets, the GCSToSambaOperator can be utilized. Also the SSHConnection or direct IP address methods can be used depending on the scenarios we fall under.

REFERENCES

[1]. Official Apache Airflow Providers for Samba - https://airflow.apache.org/docs/apache-airflow-providers-samba/stable/index.html
[2]. Documentation of GCSToSambaOperator – usage, parameters - https://airflow.apache.org/docs/apache-airflow-providers-samba/stable/transfer/gcs_to_samba.html
[3]. SimpleHttpOperator references for usage and source code - https://airflow.apache.org/docs/apache-airflow-providers-http/stable/_api/airflow/providers/http/operators/http/index.html
[4]. SSHOperator usage reference - https://docs.aws.amazon.com/mwaa/latest/userguide/samples-ssh.html#samples-ssh-code
[5]. SMB Protocol and CIFS Protocol Overview - https://learn.microsoft.com/en-us/windows/win32/fileio/microsoft-smb-protocol-and-cifs-protocol-overview