Trigger a SnapMirror Volume Replication Update

NetApp Solutions

Dorian Henderson, Kevin Hoke
January 21, 2021

Table of Contents

Trigger a SnapMirror Volume Replication Update

The example DAG outlined in this section implements a workflow that takes advantage of NetApp SnapMirror data replication technology to replicate the contents of a volume between different ONTAP clusters.

This pipeline can be used to replicate data of any type between ONTAP clusters that might or might not be located at different sites or in different regions. Potential use cases include the following:

- Replicating newly acquired sensor data gathered at the edge back to the core data center or to the cloud to be used for AI/ML model training or retraining.
- Replicating a newly trained or newly updated model from the core data center to the edge or to the cloud to be deployed as part of an inferencing application.

Prerequisites

For this DAG to function correctly, you must complete the following prerequisites.

- You must have created a connection in Airflow for your ONTAP system as outlined in Prerequisite #1 in the section “Implement an End-to-End AI Training Workflow with Built-in Traceability and Versioning. ”
- You must have already initiated an asynchronous SnapMirror relationship between the source and the destination volume according to standard configuration instructions. For details, refer to official NetApp documentation.

DAG Definition

The Python code excerpt that follows contains the definition for the example DAG. Before executing this example DAG in your environment, you must modify the parameter values in the `DEFINE PARAMETERS` section to match your environment.

```
# Airflow DAG Definition: Replicate Data - SnapMirror
#
# Steps:
#   1. Trigger NetApp SnapMirror update
from airflow.utils.dates import days_ago
from airflow.secrets import get_connections
from airflow.models import DAG
from airflow.operators.python_operator import PythonOperator
##### DEFINE PARAMETERS: Modify parameter values in this section to match
your environment #####
## Define default args for DAG
replicate_data_snapmirror_dag_default_args = {
    'owner': 'NetApp'
}
## Define DAG details
replicate_data_snapmirror_dag = DAG(
    dag_id='replicate_data_snapmirror',
    default_args=replicate_data_snapmirror_dag_default_args,
```

```
    schedule_interval=None,
    start_date=days_ago(2),
    tags=['data-movement']
)
## Define SnapMirror details (change values as necessary to match your
environment)
# Destination ONTAP system details
airflowConnectionName = 'ontap_ai'  # Name of the Airflow connection that
contains connection details for the destination ONTAP system's cluster
admin account
verifySSLCert = False   # Denotes whether or not to verify the SSL cert
when calling the ONTAP API
# SnapMirror relationship details (existing SnapMirroer relationship for
which you want to trigger an update)
sourceSvm = "ailab"
sourceVolume = "sm"
destinationSvm = "ai221_data"
destinationVolume = "sm_dest"
########################################################################
######################
# Define function that triggers a NetApp SnapMirror update
def netappSnapMirrorUpdate(**kwargs) -> int :
    # Parse args
    for key, value in kwargs.items() :
        if key == 'sourceSvm' :
            sourceSvm = value
        elif key == 'sourceVolume' :
            sourceVolume = value
        elif key == 'destinationSvm' :
            destinationSvm = value
        elif key == 'destinationVolume' :
            destinationVolume = value
        elif key == 'verifySSLCert' :
            verifySSLCert = value
        elif key == 'airflowConnectionName' :
            airflowConnectionName = value
    # Install ansible package
    import sys, subprocess, os
    print("Installing required Python modules:\n")
    result = subprocess.check_output([sys.executable, '-m', 'pip',
'install', '--user', 'ansible', 'netapp-lib'])
    print(str(result).replace('\\n', '\n'))

    # Retrieve ONTAP cluster admin account details from Airflow connection
    connections = get_connections(conn_id = airflowConnectionName)
    ontapConnection = connections[0]    # Assumes that you only have one
```

```
connection with the specified conn_id configured in Airflow
    ontapClusterAdminUsername = ontapConnection.login
    ontapClusterAdminPassword = ontapConnection.password
    ontapClusterMgmtHostname = ontapConnection.host

    # Define temporary Ansible playbook for triggering SnapMirror update
    snapMirrorPlaybookContent = """
---
- name: "Trigger SnapMirror Update"
  hosts: localhost
  tasks:
  - name: update snapmirror
    na_ontap_snapmirror:
      state: present
      source_path: '%s:%s'
      destination_path: '%s:%s'
      hostname: '%s'
      username: '%s'
      password: '%s'
      https: 'yes'
      validate_certs: '%s'""" % (sourceSvm, sourceVolume, destinationSvm,
destinationVolume, ontapClusterMgmtHostname,
        ontapClusterAdminUsername, ontapClusterAdminPassword,
str(verifySSLCert))
    print("Creating temporary Ansible playbook.\n")
    snapMirrorPlaybookFilepath = "/home/airflow/snapmirror-update.yaml"
    snapMirrorPlaybookFile = open(snapMirrorPlaybookFilepath, "w")
    snapMirrorPlaybookFile.write(snapMirrorPlaybookContent)
    snapMirrorPlaybookFile.close()
    # Trigger SnapMirror update
    print("Executing Ansible playbook to trigger SnapMirror update:\n")
    try :
        result = subprocess.check_output(['ansible-playbook',
snapMirrorPlaybookFilepath])
        print(str(result).replace('\\n', '\n'))
    except Exception as e :
        print("Exception:", str(e).strip())
        print("Removing temporary Ansible playbook.")
        os.remove(snapMirrorPlaybookFilepath) # Remove temporary Ansible
playbook before exiting
        raise
    # Remove temporary Ansible playbook before exiting
    print("Removing temporary Ansible playbook.\n")
    os.remove(snapMirrorPlaybookFilepath)
    # Return success code
    return 0
```

```
# Define DAG steps/workflow
with replicate_data_snapmirror_dag as dag :
    # Define step to trigger a NetApp SnapMirror update
    trigger_snapmirror = PythonOperator(
        task_id='trigger-snapmirror',
        python_callable=netappSnapMirrorUpdate,
        op_kwargs={
            'airflowConnectionName': airflowConnectionName,
            'verifySSLCert': verifySSLCert,
            'sourceSvm': sourceSvm,
            'sourceVolume': sourceVolume,
            'destinationSvm': destinationSvm,
            'destinationVolume': destinationVolume
        },
        dag=dag
    )
```

Next: Trigger a Cloud Sync Replication Update