Solution Automation

NetApp Solutions

NetApp
April 18, 2022

Table of Contents

Solution Automation

NetApp Solution Automation

Introduction

One of the objectives of validating and architecting solutions is to make the solution easily consumable. Therefore, it is paramount that the deployment and configuration of infrastructure and/or applications delivered through our solutions is simplified through automation. NetApp is committed to simplifying solution consumption through automation using RedHat Ansible.

Ansible is an open-source automation engine that helps IT teams automate application deployment, cloud provisioning, configuration management, and many other IT needs. Ansible is agentless and does not require a custom security infrastructure. You can manage the automation of multiple systems from your control system remotely via SSH making it a robust solution for IT teams looking to automate their tedious and repetitive IT needs.

If you are new to NetApp solution automation, you can use the following sections to set up your Ansible controller.

For more information about RedHat Ansible, see the documentation here.

Setup the Ansible control node (For CLI based deployments)

NetApp Solution Automation

Procedure

1. Requirements for the Ansible control node,:
   a. A RHEL/CentOS machine with the following packages installed:
      i. Python3
      ii. Pip3
      iii. Ansible (version greater than 2.10.0)
      iv. Git

If you have a fresh RHEL/CentOS machine without the above requirements installed, follow the below steps to setup that machine as the Ansible control node:

1. Enable the Ansible repository for RHEL-8/RHEL-7
   a. For RHEL-8 (run the below command as root)

```
subscription-manager repos --enable ansible-2.9-for-rhel-8-x86_64-
rpms
```

   b. For RHEL-7 (run the below command as root)

```
subscription-manager repos --enable rhel-7-server-ansible-2.9-rpms
```

2. Create a .sh file

```
vi setup.sh
```

3. Paste the below content in the file

```
#!/bin/bash
echo "Installing Python ------>"
sudo yum -y install python3 >/dev/null
echo "Installing Python Pip ------>"
sudo yum -y install python3-pip >/dev/null
echo "Installing Ansible ------>"
python3 -W ignore -m pip --disable-pip-version-check install ansible
>/dev/null
echo "Installing git ------>"
sudo yum -y install git >/dev/null
```

4. Make the file executable

```
chmod +x setup.sh
```

5. Run the script (as root)

```
./setup.sh
```

NetApp Solution Automation

Procedure

1. Requirements for the Ansible control node,:
   a. A Ubuntu/Debian machine with the following packages installed:
      i. Python3
      ii. Pip3
      iii. Ansible (version greater than 2.10.0)
      iv. Git

If you have a fresh Ubuntu/Debian machine without the above requirements installed, follow the below steps to setup that machine as the Ansible control node:

1. Create a .sh file

```
vi setup.sh
```

2. Paste the below content in the file

```
#!/bin/bash
echo "Installing Python ------>"
sudo apt-get -y install python3 >/dev/null
echo "Installing Python Pip ------>"
sudo apt-get -y install python3-pip >/dev/null
echo "Installing Ansible ------>"
python3 -W ignore -m pip --disable-pip-version-check install ansible
>/dev/null
echo "Installing git ------>"
sudo apt-get -y install git >/dev/null
```

3. Make the file executable

```
chmod +x setup.sh
```

4. Run the script (as root)

```
./setup.sh
```

NetApp solution automation

Procedure

This section describes the steps required to configure the parameters in AWX/Ansible Tower that prepare the environment for consuming NetApp automated solutions.

1. Configure the inventory.
    a. Navigate to Resources → Inventories → Add and click Add Inventory.
    b. Provide name and organization details and click Save.
    c. In the Inventories page, click the inventory resources you just created.
    d. If there are any inventory variables, paste them into the variables field.
    e. Go to the Groups sub-menu and click Add.
    f. Provide the name of the group, copy in the group variables (if necessary), and click Save.
    g. Click the group created, go to the Hosts sub-menu and click Add New Host.

h. Provide the hostname and IP address of the host, paste in the host variables (if necessary), and click Save.

2. Create credential types. For solutions involving ONTAP, Element, VMware, or any other HTTPS-based transport connection, you must configure the credential type to match the username and password entries.
   a. Navigate to Administration → Credential Types and click Add.
   b. Provide the name and description.
   c. Paste the following content into the Input Configuration:

```
fields:
- id: username
type: string
label: Username
- id: password
type: string
label: Password
secret: true
- id: vsadmin_password
type: string
label: vsadmin_password
secret: true
```

   a. Paste the following content into the Injector Configuration:

```
extra_vars:
password: '{{ password }}'
username: '{{ username }}'
vsadmin_password: '{{ vsadmin_password }}'
```

1. Configure credentials.
   a. Navigate to Resources → Credentials and click Add.
   b. Enter the name and organization details.
   c. Select the correct credential type; if you intend to use the standard SSH login, select the type Machine or alternatively select the custom credential type that you created.
   d. Enter the other corresponding details and click Save.
2. Configure the project.
   a. Navigate to Resources → Projects and click Add.
   b. Enter the name and organization details.
   c. Select Git for the Source Control Credential Type.
   d. Paste the source control URL (or git clone URL) corresponding to the specific solution.
   e. Optionally, if the Git URL is access controlled, create and attach the corresponding credential in Source Control Credential.

f. Click Save.

3. Configure the job template.
   a. Navigate to Resources → Templates → Add and click Add Job Template.
   b. Enter the name and description.
   c. Select the Job type; Run configures the system based on a playbook and Check performs a dry run of the playbook without actually configuring the system.
   d. Select the corresponding inventory, project, and credentials for the playbook.
   e. Select the playbook that you would like to run as a part of the job template.
   f. Usually the variables are pasted during runtime. Therefore, to get the prompt to populate the variables during runtime, make sure to tick the checkbox Prompt on Launch corresponding to the Variable field.
   g. Provide any other details as required and click Save.
4. Launch the job template.
   a. Navigate to Resources → Templates.
   b. Click the desired template and then click Launch.
   c. Fill in any variables if prompted on launch and then click Launch again.

NetApp Solution Automation

AWS Authentication Requirements for CVO and Connector Using NetApp Cloud Manager

To configure automated Deployments of CVO and Connectors using Ansible playbooks via AWX/Ansible Tower, the following information is needed:

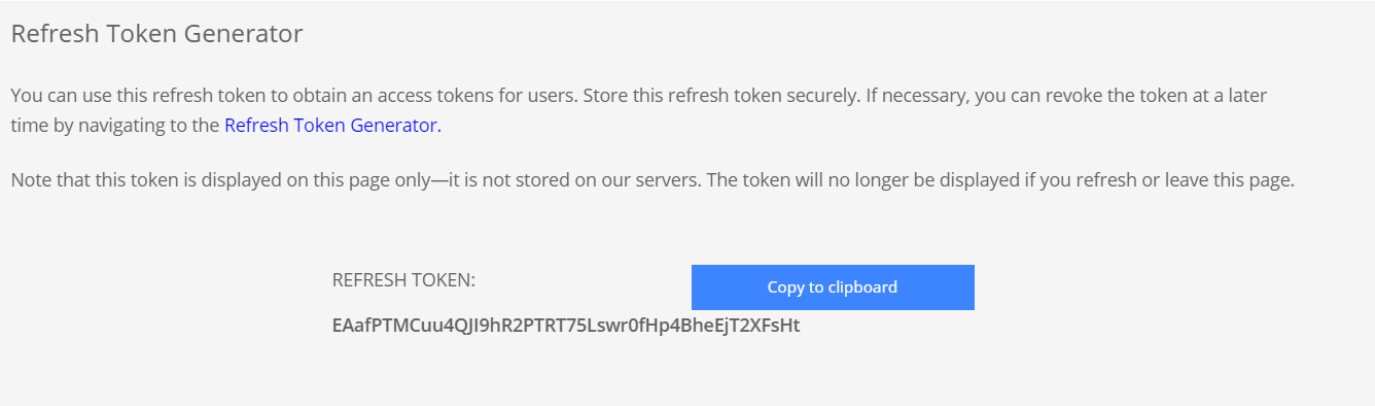
Acquiring Access/Secret Keys from AWS

1. To deploy CVO and Connector in Cloud Manager, we need AWS Access/Secret Key. Acquire the keys in AWS console by launching IAM-→Users-→your username-→security credentials-→Create Access key.
2. Copy access keys and keep them secured to use in Connector and CVO deployment.

If you lose your key, you can create another access key and delete the one you lost

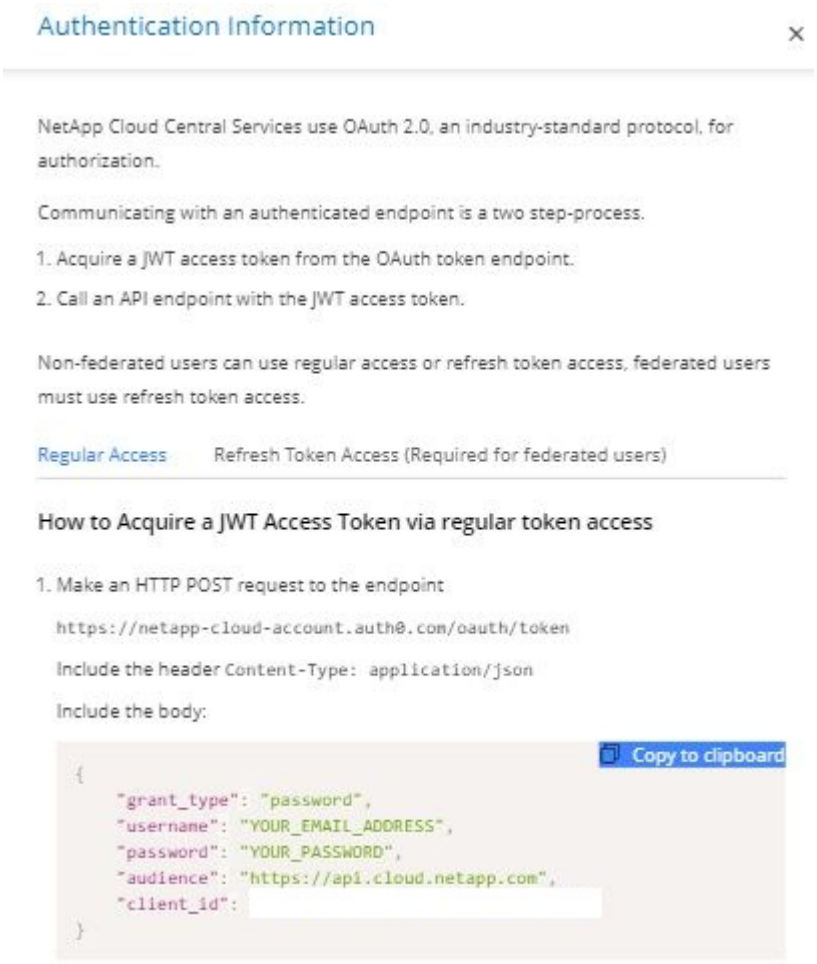
Acquiring Refresh Token from NetApp Cloud Central

1. Login into your cloud central account using your account credentials at https://services.cloud.netapp.com/refresh-token
2. Generate a refresh Token and save it for deployments.



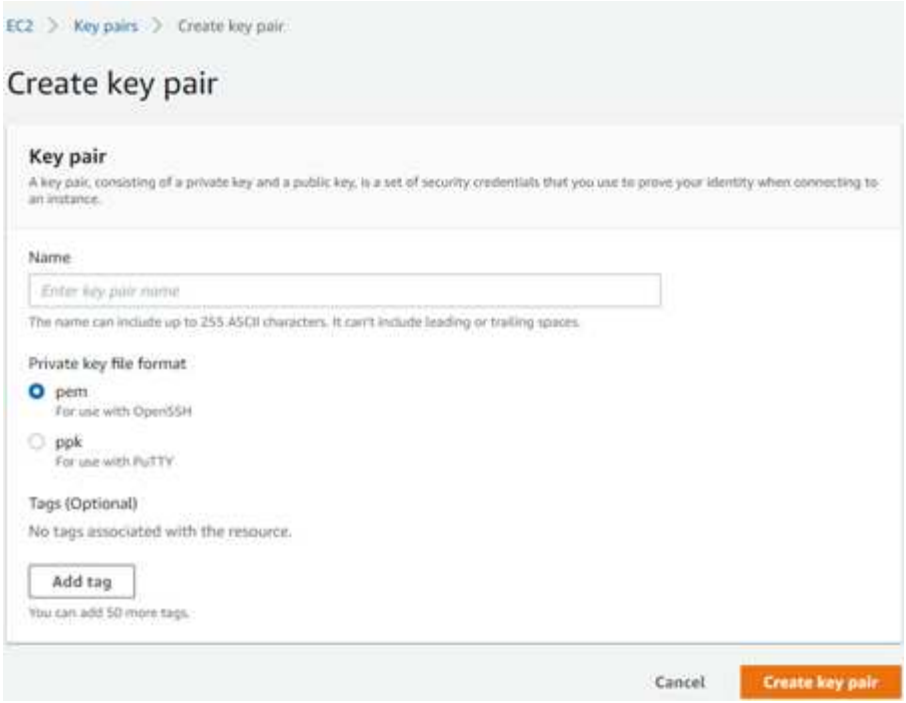
Acquiring Client ID

1. Access the API page to copy Client ID at https://services.cloud.netapp.com/developer-hub.
2. Click on "learn How to Authenticate", in the top right corner.
3. From the Authentication window that pops up, copy the Client ID from Regular Access if you require a username/password to login. Federated users with SSO should copy the client ID from the "Refresh Token Tab".



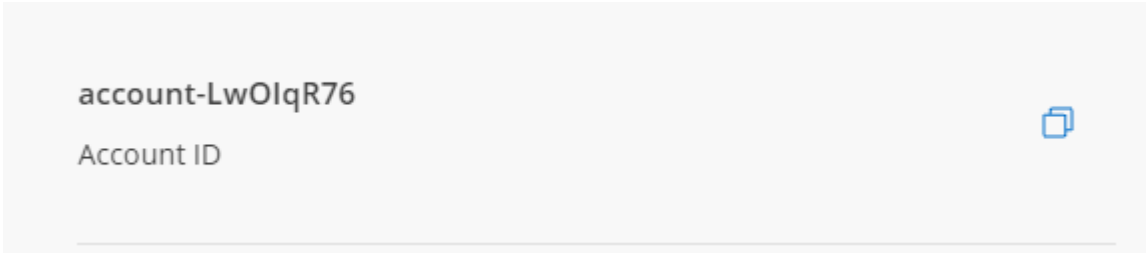
Acquiring Key Pair from AWS

1. In AWS console, search for “Key Pair” and create a key pair with “pem”. Remember the name of you key_pair, we will use it to deploy the connector.



Acquiring Account ID

1. In Cloud Manager, click on Account –> Manage Accounts and then copy the account id for use in variables for AWX.



Cloud Volumes Automation via Terraform

This solution documents the automated deployments of Cloud Volumes on AWS (CVO Single Node, CVO HA and FSX ONTAP) and Azure (CVO Single Node, CVO HA and ANF) using Terraform modules. The code can be found at https://github.com/NetApp-Automation/na_cloud_volumes_automation

Pre-requisites

1. Terraform >= 0.13
2. Cloud Manager Account
3. Cloud Provider Account – AWS, Azure
4. Host machine (any OS supported by Terraform)

Provider documentation

The documentation of Terraform provider for Cloud Manager is available at: https://registry.terraform.io/providers/NetApp/netapp-cloudmanager/latest/docs

Controlling the provider version

Note that you can also control the provider version. This is controlled by a required_providers block in your Terraform configuration.

The syntax is as follows:

```
terraform {
  required_providers {
    netapp-cloudmanager = {
      source = "NetApp/netapp-cloudmanager"
      version = "20.10.0"
    }
  }
}
```

Read more on provider version control.

Running Specific Modules

AWS

Unresolved directive in automation/cloud_volumes_terraform.adoc - include::automation/cloud_volumes_aws.adoc[]

Azure

Unresolved directive in automation/cloud_volumes_terraform.adoc - include::automation/cloud_volumes_azure.adoc[]

GCP

Unresolved directive in automation/cloud_volumes_terraform.adoc - include::automation/cloud_volumes_gcp.adoc[]