Azure Databricks Provisioning:

Azure Databricks resource can be provisioned using the ARM templates that are provided in the given GitHub link.

There are lot of configuration activities post provisioning of Databricks. Following are the activities that needs to be completed after the Databricks infrastructure provisioning is completed:

1. Users creation in Databricks
2. User Groups creation in Databricks
3. Cluster creation in Databricks
4. Save ClusterID to Keyvault for further use
5. Workspace folder creation in Databricks
6. Folder permission to User groups
7. Cluster permission to user groups in Databricks

With the help of Databricks Rest APIs it is possible to automate the above tasks. However, following tasks are still in preview:

1. User creation in Databricks
2. User Groups creation in Databricks

Hence I have not implemented the above methods in the script that I have developed as the above Rest APIs cannot be used in production system.

Since Databricks Rest API follow a very structured request JSON, in order to form such a nested JSON easily from Powershell script, I have used Powershell nested classes which later I have converted to JSON and passed as body to Databricks Rest API.

Following is the flow of events for authentication mechanism to Databricks Rest API:

1. Get token from <https://login.microsoftonline.com/$TenantID/oauth2/token> using Azure service principal ClientID and ClientSecret.

if($isPipelineExec)

{

#################### Execute the below piece of code when executed from pipeline #########################

$context= Get-AzContext

$resource="2ff814a6-3304-4ab8-85cb-cd0e6f879c1d" #This is constant for Databricks

#Generate Azure AD Token

$adToken = [Microsoft.Azure.Commands.Common.Authentication.AzureSession]::Instance.AuthenticationFactory.Authenticate($context.Account,

$context.Environment,

$context.Tenant.Id.ToString(),

$null, [Microsoft.Azure.Commands.Common.Authentication.ShowDialog]::Never, $null, $resource).AccessToken

$azure\_token= $adToken

#Generate Management API Token

$azProfile = [Microsoft.Azure.Commands.Common.Authentication.Abstractions.AzureRmProfileProvider]::Instance.Profile

$profileClient = New-Object -TypeName Microsoft.Azure.Commands.ResourceManager.Common.RMProfileClient -ArgumentList ($azProfile)

$managementToken = $profileClient.AcquireAccessToken($context.Tenant.Id.ToString())

$azure\_mgmnt\_token= $managementToken.AccessToken

}

else

{

#################### Execute the below piece of code to run the script from local #########################

$ClientID = '<ApplicationID>'

$TenantID = '<TenantID>'

$ClientSecret = "<ClientSecret>"

$Resource = '2ff814a6-3304-4ab8-85cb-cd0e6f879c1d'

$TokenRequestParams = @{

Method = 'POST'

Uri = "https://login.microsoftonline.com/$TenantID/oauth2/token"

Body = @{

client\_id = $ClientId

resource = $Resource

grant\_type= 'client\_credentials'

client\_secret=$ClientSecret

}

}

$TokenCodeRequest = Invoke-RestMethod @TokenRequestParams

#Write-Host $TokenCodeRequest.access\_token

$azure\_token= $TokenCodeRequest.access\_token

1. Use the above token to acquire token from <https://management.core.windows.net>.

$ManagementTokenRequest = @{

Method = 'POST'

Uri = "https://login.microsoftonline.com/" + $TenantID + "/oauth2/token"

Body = @{

client\_id = $ClientId

resource = "https://management.core.windows.net/"

grant\_type= 'client\_credentials'

client\_secret=$ClientSecret

}

}

$ManagementTokenCodeReq = Invoke-RestMethod @ManagementTokenRequest

#Write-Host $ManagementTokenCodeReq.access\_token

$azure\_mgmnt\_token=$ManagementTokenCodeReq.access\_token

1. Get the provisioned Databricks Instance using the API “<https://management.azure.com/subscriptions/>” + $subscriptionID + “/resourcegroups/” + $resourceGroup + “/providers/Microsoft.Databricks/workspaces/” + $workspaceName + “?api-version=2018-04-01”

#At the time of development of this script the following Rest API returns a malformed #response.

Function GetDatabricksInstance ($azureMngmntToken, $subscriptionID, $resourceGroup, $workspaceName)

{

Write-Host "-----Getting Databricks workspace URL--------------"

$headers=@{

"Authorization"= "Bearer " + $azureMngmntToken;

}

$urlDBInstance="https://management.azure.com/subscriptions/"+ $subscriptionID +"/resourcegroups/" + $resourceGroup +"/providers/Microsoft.Databricks/workspaces/"+ $workspaceName +"?api-version=2018-04-01"

$responseDB=Invoke-WebRequest $urlDBInstance -Method Get -Headers $headers

###The below string manipulation is done because

###the above rest API is returning a malformed json

###which cannot be converted to object.

$str1= $responseDB.Content | Out-String

$sampleString="adb-XXXXXXXXXXXXXXXX.XX.azuredatabricks.net"

$str2= $str1.Substring($str1.IndexOf("adb-"),$sampleString.Length)

return $str2

}

1. Using the Databricks instance from above step and management token from step 2, create a Databricks PAT token.

Function GenerateDBPATToken ($azure\_token, $azure\_mgmnt\_token, $subscriptionID, $resourceGroup, $workspaceName, $DatabricksInstance, $keyVaultName, $env)

{

[hashtable]$access\_token = @{}

$access\_token.azToken=$azure\_token

$access\_token.mngmntToken=$azure\_mgmnt\_token

$access\_token.DatabricksInstance=$DatabricksInstance

Write-Host "------Check if PAT exists-----------"

$secretName=$kvsecretName='kvs-databricks-' + $env + 'auegteng'

$secret = Get-AzKeyVaultSecret -VaultName $keyVaultName -Name $secretName -AsPlainText

if($secret -eq $null)

{

Write-Host "-----Getting Databricks PAT Token--------------"

$headers=@{

"Authorization"= "Bearer " + $azure\_token;

"X-Databricks-Azure-SP-Management-Token" = $azure\_mgmnt\_token;

"X-Databricks-Azure-Workspace-Resource-Id" = "/subscriptions/"+$subscriptionID+"/resourceGroups/" + $resourceGroup +"/providers/Microsoft.Databricks/workspaces/" + $workspaceName;

}

$PATTokenRequest = @{

Method = 'POST'

Uri = "https://" + $DatabricksInstance +"/api/2.0/token/create"

Headers = $headers

}

$dbPATTokenRequest=Invoke-RestMethod @PATTokenRequest

$access\_token.patToken=$dbPATTokenRequest.token\_value

}

else

{

$access\_token.patToken=$secret

}

SavePatTokenKeyVault $keyVaultName $access\_token.patToken $env

return $access\_token

}

The PAT token acquired in the above step would be used to perform the other activities listed above.