Azure-ML - ML-Pipelines

This document is a guide that explains how to build a Machine Learning Pipeline in Azure with Azure-ML. The basic idea based is to clone a Github-repo that contains a generic code structure for Azure-ML pipelines and then follow the instructions for creating the necessary Azure recources for the new pipeline.

Azure-ML

Storage account

The first thing we need to do is create a Azure Storage account for the Azure-ML workspace. Go into Storage accounts and create a new. Give it the name of azureml**<productname>**storage. Use West Europe as region and choose Locally redundant storage (LRS) for redundancy. Since Azure-ML do not support Data Lake Storage Gen 2 we can't enable hierarchical namespace.

Azure-ML workspace

Then we create a new Azure-ML workspace. This can be done Programmatically, with ARM-templates or manually. To do this manually simply click create under the Azure-ML service. Give it the name of azureml**<productname>**. Choose West Europe for region, the Storage account you just created, choose the keyvault namee advancedanalyticsdwhkv and then choose create new Application insights. Container registry can be left with None.

Register Datastores for the workspace (optional)

It is possible to register Datastores in the Azure-ML studio for easy access of data in the Pipeline. Register the necessary Storage Accounts that the Pipeline need access to. This might be considered best practice as we can register Datasets from a Datastore so we keep track of the versions of the Data.

Service Principal

We need a Service Principal (SP) for using Github-actions. To create a SP we use the bash-prompt (or login az in the command prompt) and write the following (add name, subscription-id and resource group):

```
az ad sp create-for-rbac --name {nameofsp} --role contributor --scopes
/subscriptions/{subscription-id}/resourceGroups/{MyResourceGroup} --sdk-
auth
```

The information for this can be found in Overview of the Azure-ML workspace. When this step is done copy the json-output to a text editor. We will use it later and it contains information about the clientIdGUID.

Then we need to add some roles to the SP:

```
az keyvault set-policy -n {keyVaultName} --secret-permissions get list
--spn {clientIdGUID}
```

As of this moment Advance Analytics do not have the athorization level to create SP:s. Ask Christer Bergström to run the code.

Add SP details to Keyvalut

The SP accesses the secrets in the Keyvault. Add the following three secrets to the advancedanalyticsdwhkv keyvault:

- AZUREML<PRODUCTNAME>-SP-CLIENTID - value
- AZUREML<PRODUCTNAME>-SP-CLIENTSECRET - value
- AZUREML<PRODUCTNAME>-SP-TENANTID - value

You find the ClientID, ClientSecret and TenantID in the JSON-output from earlier step.

Compute cluster

In the Azure-ML workspace studio; create a new cluster. For example Standard_D14_v2 is used for Recsys. Fill in 0 minimum node and for example 4 maximum nodes.

Github

Create a new Github-repo for the ML-pipeline. Then we need to clone the ML-Template repository: https://github.com/ahlsell-group/advancedanalytics-mlpipeline-template to the new Github-repo. One easy way is to first clone the ML-Template. Then fill in the following in the terminal:

```
git remote set-url origin <url-of_new_repo>
git push -u origin master
```

Github Secrets

Now we need to add the information of the SP to Github repo. Go into settings Secrets and then add New Repository Secret. Take the JSON-output you got from creating the SP and paste it as is to the window and name it, for example AZURE_CREDENTIALS.

Azure config-files

Now the Azure-ML workspace and Github-repo is all set for use. We need to change some of the code-files in order to set up communication with the right sources. The following folders that we need to look over is:

./.cloud/.azure/

The files in this folder is used when we execute Azure-specific Github actions. We only need this for Github actions. We have three files here:

- compute.json - Change the name of the compute-name to the same name as the Compute-cluster you created in earlier step
- run.json - Doesn't need to be changed unless the run_config file is named differently
- workspace.json - Change the name to the Azure-ML workspace that was created in earlier step and change it to correct resource group used for that step

./azure_ml/src/.azureml/

The files in this folder is used for the main script (model1_pipeline.py in the template-version) is run in order for the program to communicate with the correct Azure-ML workspace. We have 3 files here:

- config.json - Change this file so that the correct subscription-id, resource group and workspace name is used. This information can be found in the Overview as explained above
- environment_scriptstep.yml - This file contains the conda-environment for the PythonScriptSteps of a pipeline. Change this if you need other libraries for this step
- reuirements_sparkstep.txt - This file contains the pip-environment for the SynapseSparkStep of a pipeline. Change this if you need other libraries for this step. As of this moment it seems as if conda environments are not supported in SynapseSparkSteps