Resources to get you started on the involved technologies:

- [Azure Data Factory](https://docs.microsoft.com/en-us/azure/data-factory/)

- [Azure Data Lake Analytics](https://docs.microsoft.com/en-us/azure/data-lake-analytics/)

- [Azure ARM Templates](https://azure.microsoft.com/en-us/resources/templates/)

- [Azure ARM Template Samples](<https://github.com/Azure/azure-quickstart-templates>)

- [Azure Managed App]( <https://docs.microsoft.com/en-us/azure/managed-applications/>)

- [Azure Managed App Samples]( <https://github.com/Azure/azure-managedapp-samples/tree/master/samples>)

Azure Managed App on Office 365 can be broken down into 3 components:

1. Data Ingestion from Euclid/Office365
2. Data processing/analytics to produce intelligent data
3. UX to surface the intelligent data

This sample covers all three components:

1. We use Azure Data Factory (ADF) with copy activity to move data from O365 to your "target" adls.

2. We then have an azure web app that reads the data at "target" adls and outputs intelligent data.

# Create and publish an O365 powered azure managed application:

The instructions below will help you create and publish an azure managed application internally.

You can take a look at this doc for reference <https://docs.microsoft.com/en-us/azure/managed-applications/publish-service-catalog-app>

# Step 0:

### Bash

Before you start on the instructions below, please ensure that you have [node and npm](<https://docs.npmjs.com/getting-started/installing-node> ), [jq](<https://stedolan.github.io/jq/download/> ) and [azure-cli](<https://docs.microsoft.com/en-us/azure/cli-install-nodejs> ) installed.

### Powershell

Before you start on the instructions below, please ensure that you have [Azure Powershell](<https://docs.microsoft.com/en-us/powershell/azure/install-azurerm-ps?view=azurermps-4.4.0> ) installed.

After installing the above pre-requisites fire below command in shell of your choice and follow the instructions to login to your azure subscription.

Bash

```bash

azure login

```

Powershell

```bash

Login-AzureRmAccount

## Step 1:

Create a template that defines the resources to deploy with the managed application. ( refer mainTemplate.json)

If you look at the mainTemplate.json, it consists of three main sections

### Parameters :

Contains the list of parameters whose values will be provided by the user.

|  |  |
| --- | --- |
| **Parameter Name** | **Description** |
| WebSiteName | The website name, used as the prefix in the url of the published web app. E.g. <websitename>.azurewebsites.net |
| DestinationServicePrincipalAadId | The AAD Id of the service principal to be granted access to the destination Data Lake store |
| DestinationServicePrincipalId | The app Id of the service principal that has access to the destination Data Lake store |
| DestinationServicePrincipalKey | The app secret of the service principal that has access to the destination Data Lake store |
| Office365TenantId | The O365 tenant for which data needs to be extracted |
| ServicePrincipalId | The app Id of the service principal that has access to the source O365 |
| ServicePrincipalKey | The app secret of the service principal that has access to the source O365 |

Create Azure AD service principal to grant access to the "Destination" ADLS following these [instructions] (<https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-create-service-principal-portal> ).

You need to follow sections titled "**Create an Azure Active Directory application**" and "**Get application ID and authentication key**".

> \*\*Note:\*\* the sign-on url should be https://\*\*webSiteName\*\*.azurewebsites.net.

> \*\*webSiteName\*\* can be anything you want, just ensure that you think of something that will be unique around the globe and it will be the same value that will be specified to the websitename parameter

#### Filling in the parameters:

- `WebSiteName` is the same value you set for the sign-on url.

- ` DestinationServicePrincipalId ` is the Application Id of your service principal.

- `DestinationServicePrincipalKey` is the Authentication Key you generated for your service principal.

- `DestinationServicePrincipalAadId` is the `Object Id` (bash)/`Id` (powershell) you receieve when you run the following command:

> \*\*Note:\*\* Do not use the object id that you see on azure portal

Bash

```bash

azure ad sp show -c [service principal friendly name]

```

Powershell

```bash

Get-AzureRmADServicePrincipal -SearchString [service principal friendly name]

```

### Variables :

Contains the list of variables. You can update the variable values if needed.

### Resources :

Contains the list of resources that will be deployed as a part of the managed app creation.

Below are few of the resources that will be deployed as a part of the mainTemplate.json explained briefly.

DestinationAdlsAccount: Creates the destination Data Lake store in the customer's subscription used in the ADF pipeline for the data output.

DataFactory: Creates the ADF pipeline that copies data from O365 to the newly created destination ADLS (DestinationAdlsAccount that was created above)

* **SourceLinkedService** : Creates the link to O365 which is used as the source of the data extraction. Using service principal supplied by the source ADLS owner.
* **DestinationLinkedService** : Creates the link to the newly created destination ADLS, using service principal supplied by the customer deploying this template.
* **InputDataset** : You should change the structure in this resource to match the table and columns that you would like to extract. In this template we are trying to extract messages and the structure corresponds to that Office365 table.
* **OutputDataset** : Corresponds to the DestinationAdlsAccount where we wanted the data to be copied to.
* **Pipeline** : The Copy activity pipeline that copies the data from source O365 to the destination ADLS
* **PipelineTriggers**: Contains settings to ensure the copy pipeline can be scheduled to run periodically.

WebApp: Creates the web app that uses data stored in the newly created destination ADLS.

## Step 2:

Define the user interface elements for the portal when deploying the managed application. ( refer createUiDefinition.json). The Azure portal uses the createUiDefinition.json file to generate the user interface for users who create the managed application. You define how users provide input for each parameter. You can use options like a drop-down list, text box, password box, and other input tools. To learn how to create a UI definition file for a managed application, see [Get started with CreateUiDefinition](https://docs.microsoft.com/en-us/azure/managed-applications/create-uidefinition-overview).

The values of the parameters defined in mainTemplate.json are supplied through the UI generated by createUiDefinition.json when the managed application is being created.

## Step 3:

Follow the steps under the section ‘**Packages the Files’** in <https://docs.microsoft.com/en-us/azure/managed-applications/publish-service-catalog-app> for packaging the template files and uploading them to a blob storage.

## Step 4:

Create a user group or application for managing the resources on behalf of a customer by following the steps under the section ‘**Create the managed application definition’** in <https://docs.microsoft.com/en-us/azure/managed-applications/publish-service-catalog-app>

Get the role definition ID by following the steps in [https://docs.microsoft.com/en-us/azure/managed-applications/publish-service-catalog-app#get-the-role-definition-id](https://docs.microsoft.com/en-us/azure/managed-applications/publish-service-catalog-app)

## Step 5:

Create the managed application definition using the following command

New-AzureRmManagedApplicationDefinition `

-Name "EuclidManagedApp" `

-Location "westcentralus" `

-ResourceGroupName appDefinitionGroup `

-LockLevel ReadOnly `

-DisplayName "O365 powered Managed Application" `

-Description " O365 powered Managed Application " `

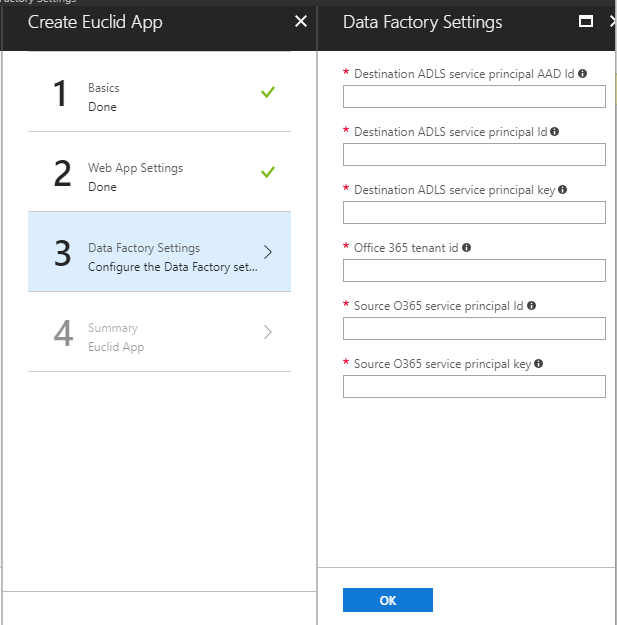
-Authorization "<group-id>:<owner-id> that was got from step 4" `

-PackageFileUri <Path to the blob storage where the zip file was uploaded in Step 3>

## Step 6:

### Creating the Managed Application

You can create the Managed Application by following the steps listed below.

* Go to Azure Portal and choose ‘**Managed Applications’** from All Services
* Click on Add and you will see the Managed Application definition that we created above.
* Select the Managed App definition that you want to create and click on create.
* The Create app window appears as below where you can specify the values for the parameters (discussed in Step 1)

The deployment of the app starts and once it completes you will be able to see it in the dashboard.

## Step 7:

Click on the app and in the overview section you will see two resource groups. Click on the managed resource group.

You will notice that all the resources mentioned in the ARM template have been created successfully.

### Running the ADF Pipeline

Execute the following command to kick off the ADF pipeline in powershell/ bash-

* Create variable for resource group name, data factory name, and trigger name
  + - $resourceGroupName = < name of the managed resource group which contains the data factory >
    - $dataFactoryName = <data factory name>
    - $pipelineName = < pipeline name that was specified in the ARM template under variables section>
* Start the pipeline:

Invoke-AzureRmDataFactoryV2Pipeline

-ResourceGroupName <Name of the Managed Resource Group which contains the Data Factory>

-DataFactory <Data Factory Name>

-PipelineName <Name of the pipeline that was specified in the ARM template under variables section>

To run the pipeline on a scheduled basis-

* Create variable for resource group name, data factory name, and trigger name
  + - $resourceGroupName = <managed resource group name>
    - $dataFactoryName = <data factory name>
    - $triggerName = <trigger name>
* Start trigger:

Start-AzureRmDataFactoryV2Trigger

-ResourceGroupName $resourceGroupName

-DataFactoryName $dataFactoryName

-TriggerName $triggerName

* Get status of the trigger:

Get-AzureRmDataFactoryV2Trigger

-ResourceGroupName $resourceGroupName

-DataFactoryName $dataFactoryName

-Name $triggerName

* Example response:
  + - TriggerName : MarsEuclidTestTrigger
    - ResourceGroupName : MarsEuclid1\_ResourceGroup
    - DataFactoryName : MarsEuclid2DataFactory
    - Properties : Microsoft.Azure.Management.DataFactory.Models.ScheduleTrigger
    - RuntimeState : Started
* If you need to stop the trigger, do the following:

Stop-AzureRmDataFactoryV2Trigger

-ResourceGroupName $resourceGroupName

-DataFactoryName $dataFactoryName

-TriggerName $triggerName

### Monitor the Pipeline

* In the Azure Portal, go to your data factory.
* Click on the Author & Monitor quick link tile.
* In the Azure Data Factory page that opens, click on the monitor icon. You will be able to see the ADF pipelines and their status

# Web App Sample UX

