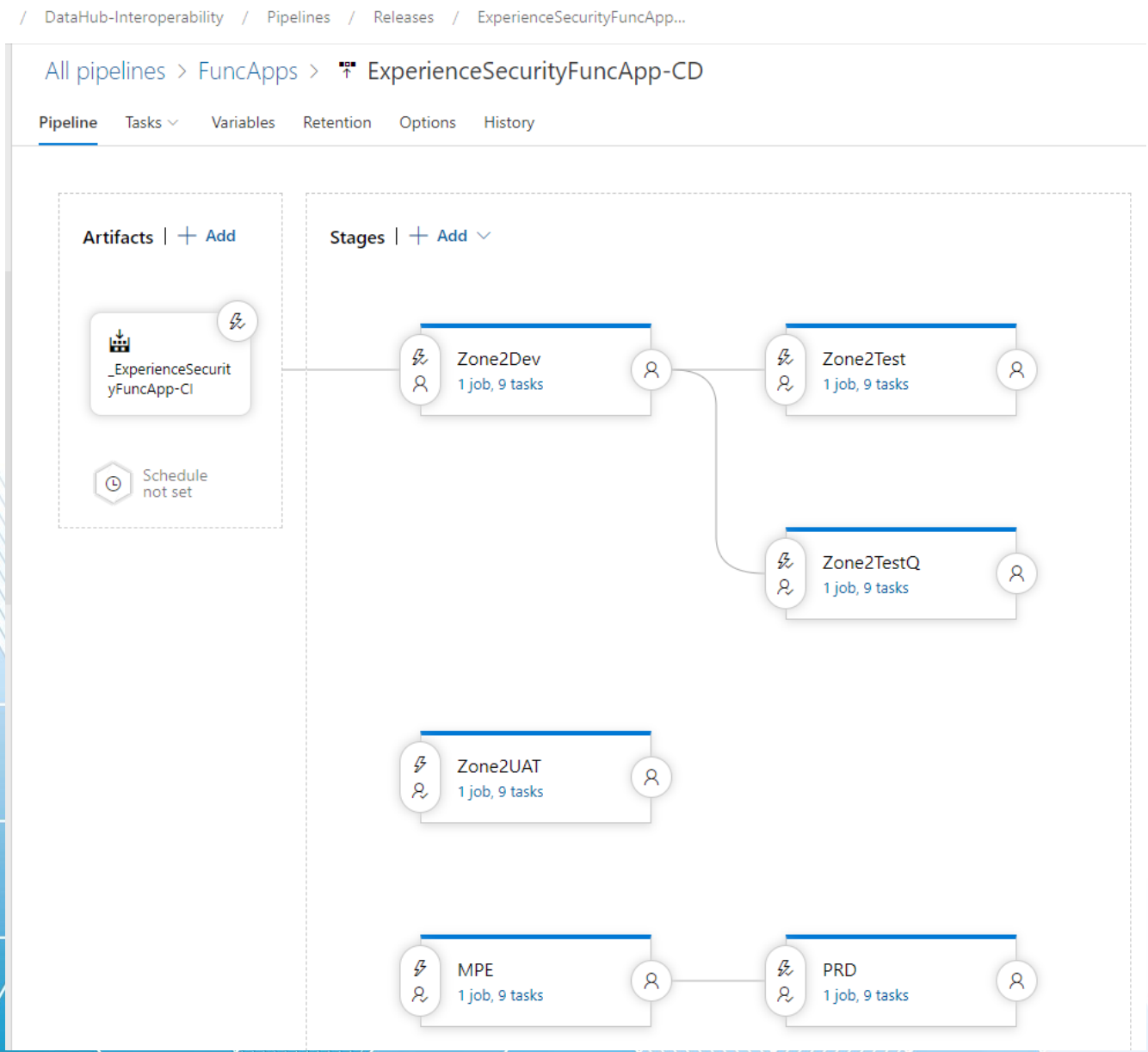
Creating a New Data Hub Environment - Zone2

Pre-requisites

A new environment will require new Resource Groups and User Assigned Managed Identity. A new Data Hub environment from Azure Devops perspective is essentially setup as a Release Stage across multiple Release Pipelines.



new-stage

Resource Groups

- All Resource Groups need to be created by ITB - ACE team prior to Release being run

- ACE will additionally grant permissions to the Resource Group to the Data Hub Service Principal (sp-) and the Integration teams Azure AD group
- This request is performed via a ServiceNow Universal Request. Will require approval from Manager - Integration
- See KBA0032185 - CORP - Azure - Request new Resource Group
- Example request - UR00104789 or CHT0230848 (old service now)

  Request Details. We need two resource groups created. One for the functions apps and one for the database resources.

Property	Value
Environment	Zone2
Subscription	EA-AUS-Z2
Resource Group Names	rg-dh-func-{env}-zone2 and rg-dh-db-{env}-zone2
Location	Australia East
Business Owner	Director, Application Development and Delivery
Cost Centre	2002568
Solution	Data Hub
RBAC	Contributors: sp-dh-zone2; CL-Kalpesh KHATRI; CL-Martin BANKS; CL-Brett BEATTIE

Managed Identity Account

- All Azure Functions are run using the same Managed User Identity.
- Managed identity Name: id-dh-zone2{env}-qed-qld

- SOC will initially create the Managed User Identity and assigned Azure Role **API Management Service Contributor** access to the API Management instance (to allow for the DSB->API sync)
- This request is performed via a ServiceNow UR See UR00104876 INC10626501 (old service now)
- Once created, the Managed User Identity is granted access to the DSB DB via a [manual script] (../HowToGuides/DSBApplicationCatalogueSQLDatabaseAlwaysEncrypted.md)
- The Managed User Identities Azure AD ClientId is also configured in the following **FuncAppSettings-** variable group settings
  - azureServicesAuthConnectionString (RunAs statement containing the AppId and TenantId)
  - providerFunctionManagedIdentityName (the managed identity resource name)

New Environment Creation

The following high level steps are involved in creating a new Data Hub environment

1. Execute the environment creation utility
2. Manually update Variable Groups described below
3. Manually update the Release Variables for Function Apps
4. Manually update the Release Variables for APIs
5. Execute the Appropriate Stage for the Release Pipelines following the Deployment Dependency Order
6. Configure the App Catalog Database to create Groups, API Operations, Permission. Described further in Deployment Dependency Order
7. Sync the App Catalog Database with Azure APIM. Described further in Deployment Dependency Order

Environment Creation Utility

The Environment Creation Utility will - replicate the Variable Groups and substitute values that are generic or following standard naming conventions. - replicate the Release Stage for Function Apps - replicate the Release Stage for APIs

This utility is a .Net Core 3.1 Console App and is driven via configuration in the appsetings.json file

The utility is located in the DataHub-Intereoperability repo. See the readme for setup and usage

The components that need to be created will be dependent on the requirements of the Project. See Components List to identify the components that may be required.

Library Variable Groups

Library Variable Groups are collection of settings that are shared between multiple release definitions. These variables group are partitioned by Functional Areas such as API settings, Function App Settings, Monitoring settings. For each Functional Area, variables group are created for specific environments such as Zone2Dev, Zone2Test, Zone2UAT. Example:

Functional Area: **FuncAppSettings-HostingPlans** - FuncAppSettings-HostingPlans-DSB-**{env}** - FuncAppSettings-HostingPlans-DSB-**MPE** - FuncAppSettings-HostingPlans-DSB-**Zone2UAT** - FuncAppSettings-HostingPlans-DSB-**PRD** - FuncAppSettings-HostingPlans-DSB-**Zone2Test** - FuncAppSettings-HostingPlans-DSB-**Zone2Dev**

Functional Area: **FuncAppSettings-** - FuncAppSettings-**{env}** - FuncAppSettings-**PRD** - FuncAppSettings-**MPE** - FuncAppSettings-**Zone2UAT** - FuncAppSettings-**Zone2Test** - FuncAppSettings-**Zone2Dev**

As part of new environment provisioning, typically the settings from an established environment such as Zone2Test will be cloned and variables updated as necessary. As new environments will only be created in Zone2 environment, a number of variables will remain unchanged.

There are however certain values that will need to be manually updated in the New Library Variable Groups in the Azure Devops UI as follows:

MonitoringSettingsDSB-**{env}**

Property	Value	Comments
eventHubConnectionString	{connectionstring}	Event Hub connection string for sending logs to. Zone2Test, Zone2Uat, Zone2{env} all share the same event hub.

FuncAppSettings-**{env}**

Property	Value	Comments
azureServicesAuthConnectionString	RunAs=App;AppId={application id};TenantId=db630ef6-1667-4b9b-b52e-341a723742d7	From the Azure Portal Azure Active Directory, search for the Managed Identity 'id-dh-zone2{env}-qed-qld' copy Application Id GUID
eipRelayTargetAPIAccessToken	{access token}	**Legacy/unused variable**. The SAS token can be copied from the Zone2 APIM Named Values. Zone2Test, Zone2Uat, Zone2{env} all share the token
parallelOrchestratorUrl	{connectionstring}	The connection string for the Parallel Orchestor Logic App is retrieved from the Azure Portal. Available after deploy the Logic App
serviceBusConnectionString	{connectionstring}	Available after deploying CommonAsyncEvents-CD. From Azure Portal, goto Service Bus > Shared access policies > SendAndListenOnly > Copy Primary Connection String

DSB-DB-CD

The following **{env}** scoped variables need to be updated afte the utility has run.

Property	Value	Comments
spDataHubSecret	Insert from Password Vault	From the password vault, copy password and set to Secret
sqlAdministratorLoginPassword	Insert from Password Vault	From the password vault, copy password and set to Secret

After the deployment has completed, you will need to run scripts to configure the Application Catalogue Database.

Firstly, Database users will need to be created as follows:

- Connect to the EIP workstation that has direct access to the Azure
- Start **SSMS 18** and connect to the DSB DB (*the physical name being /ApplicationCatalogue*):
  - Zone2Dev: db-dh-zone2dev-qed.database.windows.net,1433
  - Zone2: db-dh-zone2-qed.database.windows.net,1433
- Authenticate using your **CL** account
  - Choose authentication type: “Azure Active Directory - Universal with MFA”
  - Enter the **email address** of your CL account as the User Name
  - If you need to decrypt the secure columns, on the Options tab, enter “Column Encryption Setting=Enabled” in the additional connection parameters box

Add the Provider’s FunctionApp ManagedIdentity (configured as part of the Provider’s Release pipeline) and grant the required DB Role (permissions to call required stored procedures etc).

Manually run the below from SSMS using a CL account which belongs to the SQL Ad Admin Group.

```
CREATE USER [id-dh-zone2{env}-qed-qld] FROM EXTERNAL PROVIDER;
EXEC sp_addrolemember N'app_DataHubProviderReader', N'id-dh-zone2{env}-qed-qld'
GRANT VIEW ANY COLUMN ENCRYPTION KEY DEFINITION TO [id-dh-zone2{env}-qed-qld]
GRANT VIEW ANY COLUMN MASTER KEY DEFINITION TO [id-dh-zone2{env}-qed-qld]
```

Important Considerations

Prior to executing scripts to configure the App Catalog database, there are a few considerations to be made.

This is due to the fact that in Zone2, a single APIM instance exists to support multiple Data Hub environments. Naming conventions are used to differentiate the environments, like using an API suffix example, /student-api-testq/.

Some of these activities/learning have been captured in TestQ DSB Config

The DSB APIM Sync Process utilizes certain GUIDs for syncing between the App Catalog Databse and the corresponding APIM Artifacts. In particular

DSB Application Catalogue Entity	APIM Object
Group->GroupId	Product->Id
ApplicationGroup->ApplicationGroupId	Subscription->Id

GroupId

To ensure the **[Group].GroupId** is unique across all Zone2-{env} databases, the **03_PopulateGroup_*.sql** scripts need to be updated to use the environment name when generating the deterministic GUID for GropupId.

To resolve, the determination of the GroupId has been updated from

```
DECLARE @Group_Id UNIQUEIDENTIFIER = CAST(HASHBYTES('sha2_256', 'STUDENT') AS UNIQUEIDENTIFIER)
```

to

```
Declare @EnvironmentSuffix varchar(100) = '';

    -- Append '-UAT' suffix to Group name for UAT environment only (so that these match the UAT Product names in Zone2 APIM instance)
    IF @@SERVERNAME = 'db-dh-zone2uat-qed'
    BEGIN
      SET @EnvironmentSuffix = '-UAT'
    END
```

```
-- Append '-TestQ' suffix to Domains for TestQ environment only (so that these match the TestQ Produc
t names in Zone2 APIM instance)
IF @@SERVERNAME = 'db-dh-zone2testq-qed'
BEGIN
  SET @EnvironmentSuffix = '-TestQ'
END

--Groups Ids in Zone2 need to be unqiue as they share the same APIM instance. GroupId > ProductId i
n APIM.
DECLARE @StudentGroup_Id UNIQUEIDENTIFIER = CAST(HASHBYTES('sha2_256', 'STUDENT' +
@EnvironmentSuffix) AS UNIQUEIDENTIFIER)
```

ApplicationGroupId

The **[ApplicationGroup].ApplicationGroupId** becomes the APIM SubscriptionId. As multiple environments are hosted in the same APIM Instance in Zone2, the [ApplicationGroup].ApplicationGroupId needs to be environment aware. The **04_PopulateApplication_*.sql** need to be updated use the environment name when generating the deterministic GUID for ApplicationGroupId from

```
DECLARE @AppGroup_Id UNIQUEIDENTIFIER = CAST(HASHBYTES('sha2_256', 'STUDENT-' + @App
_iStudent) AS UNIQUEIDENTIFIER)
```

to

```
DECLARE @AppGroup_Id UNIQUEIDENTIFIER = CAST(HASHBYTES('sha2_256', 'STUDENT' + @Envir
onmentSuffix + '-' + @App_iStudent) AS UNIQUEIDENTIFIER)
```

Execute other scripts as described in the App Catalog Deployment Guide

See DSB DB SQL Always Encrypted and DSB Application Catalogue Maintenance