

Key2 Consulting

Azure Resource Provisioning: Data Integration

**October 7, 2023**

**Table of Contents**

[Document Intent 4](#_Toc149895151)

[1 Resource Group Documentation 4](#_Toc149895152)

[1.1 Resource Group Access Control 4](#_Toc149895153)

[1.2 Resource Naming Conventions 4](#_Toc149895154)

[2 Download EIA Demo GitHub Repository 4](#_Toc149895155)

[3 Create a Resource Group 5](#_Toc149895156)

[4 Create Resources 7](#_Toc149895157)

[4.1 Resource: Key Vault 7](#_Toc149895158)

[4.2 Resource: Azure Data Lake Storage Gen2 Storage Account 9](#_Toc149895159)

[4.3 Resource: Azure Data Factory (ADF) Workspace 12](#_Toc149895160)

[4.4 Resource: Azure SQL Database Server 13](#_Toc149895161)

[4.5 Resource: Azure Databricks Workspace 16](#_Toc149895162)

[4.6 Resource: Logic App to Send Email Alerts in ADF 18](#_Toc149895163)

[5 Databricks Resource Configuration 24](#_Toc149895164)

[5.1 Configure an Azure Key Vault-backed Secret Scope in Databricks 24](#_Toc149895165)

[5.2 Create an Azure AD Application and Service Principal 26](#_Toc149895166)

[5.3 Configure Key Vault for Databricks Mount 26](#_Toc149895167)

[5.4 Configure AD Application as Storage Blob Data Contributor 27](#_Toc149895168)

[5.5 Mount ADLS on Azure Databricks 29](#_Toc149895169)

[6 ADF Resource Configuration 31](#_Toc149895170)

[6.1 Configure ADF Workspace as Storage Blob Data Contributor 31](#_Toc149895171)

[6.2 Configure ADF Workspace as a Contributor to ADF Workspace 31](#_Toc149895172)

[6.3 Add ADF Workspace as a Contributor to Databricks 32](#_Toc149895173)

[7 Key Vault Secrets 33](#_Toc149895174)

[7.1 Create Secret for ADLS Access Key 33](#_Toc149895175)

[7.2 Create Secret for SQL Service Accounts 34](#_Toc149895176)

[7.3 Create Secret for EIA API Key 34](#_Toc149895177)

[8 Configure Access Policies for ADF in Key Vault 35](#_Toc149895178)

[9 Import DIF Azure SQL Database 36](#_Toc149895179)

[10 Connect GitHub Repo to ADF and Databricks 41](#_Toc149895180)

[11 ADF Configuration 47](#_Toc149895181)

[11.1 Configure Environment Specific Settings in ADF 47](#_Toc149895182)

[11.2 Overwrite live mode in ADF 50](#_Toc149895183)

[12 DIF Database Configuration Explained 51](#_Toc149895184)

[12.1 Configure DIF Database 51](#_Toc149895185)

[13 Data Source Configuration 53](#_Toc149895186)

[13.1 Configure Parameter Files for EIA Pipelines (ADF) 53](#_Toc149895187)

[13.2 Download Source Files for Landing Zone 54](#_Toc149895188)

[14 Delta Live Tables 55](#_Toc149895189)

[14.1 Pipeline Settings 56](#_Toc149895190)

[15 Trigger ADF Pipelines 58](#_Toc149895191)

[15.1 Trigger Order 59](#_Toc149895192)

## 

# Document Intent

This document describes the steps required for data integration using Azure. The Resource Group, Resources, and source code described in this document are intended to be used for sandbox purposes only. Do not follow these instructions for development purposes.

# Resource Group Documentation

A Resource Group is used for data integration. The resource group is a logical container used to manage and organize related Azure resources. These resources will be used in the overall framework to accomplish the stated goal. Please verify and note the following points before proceeding to step 2.

## Resource Group Access Control

To create the Resource Group, the end user must have the Contributor role at the Subscription level. After creating the Resource Group, the end user must then be granted Owner access to the Resource Group.

## Resource Naming Conventions

Microsoft has published naming convention [guidelines](https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming) for resources provisioned in Azure. A naming convention helps the end user quickly identify the resource's type, associated workload, environment, and the Azure region hosting it. The naming convention will observe the following format:

* [Resource Type]-[Project Name/Workload]-[Environment]-[Azure Region]-[Instance]

There may be slight exceptions, depending on the resource type and situation, but it is important to adhere to this convention as much as possible. The string below illustrates the name given to the Resource Group that will be recreated in the next section.

**adf-key2sandbox-dev-eastus-001**

Bear in mind that your resource group and resources should follow the same structure. For example, if your name is John Smith, the following names should be assigned to your resources:

|  |  |
| --- | --- |
| **Resource Type** | **Example Name** |
| Resource Group | rg-jsmithsandbox-dev-eastus-001 |
| Storage Account | stjsmithsboxdeveastus001 |
| Key Vault | kv-jsmith-dev-eastus-001 |
| Azure Data Factory | adf-jsmithsandbox-dev-eastus-001 |
| Azure Databricks | dbw-jsmithsandbox-dev-eastus-001 |
| Azure SQL server | sql-jsmithsandbox-dev-eastus-001 |
| Azure SQL Database | sqldb\_dev\_jsmithsandbox\_difconfig |

**Do not use the exact names of the resources provisioned below in this document!** These specific resource names may already exist in Azure. Use the names based on your first and last name as illustrated in the table above.

# Download EIA Demo GitHub Repository

The **key2-demo-eia repo** in GitHub was created to showcase an Azure data integration solution using data sourced from the US Energy Information Administration (EIA). This repo will be cloned and used for source control in subsequent steps, but you’ll first need to access certain files in this repo to create and configure the resource group.

1. Using a web browser, Navigate to the repository for the EIA demo in GitHub:

**https://github.com/Key2Consulting/key2-demo-eia**

1. On the main page of the **key2-demo-eia** repo in GitHub, click on the **Code** dropdown and select **Download ZIP**. This will download the repo as a zip file locally to your computer.

A screenshot of a computer

Description automatically generated

1. Unzip the zip file.

# Create a Resource Group

The first step is to create the Resource Group itself.

1. In the leftmost Resource Menu (sometimes referred to as the Left Pane), click on **Resource groups**.

Graphical user interface, application

Description automatically generated

1. This expands a blade that displays all Resource Groups to which the user has access. In the Command Bar across the top of this screen, click on **+** **Create**.Graphical user interface

   Description automatically generated with medium confidence
2. On the **Create a resource group** screen, select/enter the fields as shown below.

A screenshot of a computer

Description automatically generated

1. Click on the **Next: Tags >** button and enter the tags shown below:

A screenshot of a computer

Description automatically generated

1. Click on the **Next: Review + create >** button and ensure there are no validation errors. Then click on the **Create** button to create the Resource Group.

A screenshot of a computer

Description automatically generated

1. After the Resource Group has been successfully created, click on the **Resource groups** option in the Left Pane. Observe the newly created Resource Group is now listed.

# Create Resources

The next step is to create the resources you will need for the project. Remember that resource names must adhere to the below convention:

**[Resource Type]-[Project Name/Workload]-[Environment]-[Azure Region]-[Instance]**

**adf-key2sandbx-dev-eastus-001**

## Resource: Key Vault

Azure Key Vault is a repository used to encrypt keys, certificates, and secrets such as passwords. The resources created in this document will need to authenticate and communicate with each other. Key Vault avoids the practice of hard coding passwords and other keys as well as providing other security tools needed in a data integration solution.

1. From the Left Pane, click on **Create a Resource**.
2. In the search bar at the top of the **Create a Resource** page, type in **Key Vault** and hit Enter.

Graphical user interface, application

Description automatically generated

1. Click on **Create** under the **Key Vault** search result.

A screenshot of a phone

Description automatically generated

1. On the **Basics** tab, populate the fields as shown below.

A screenshot of a computer

Description automatically generated

1. On the **Access configuration** tab, select the **Vault access policy** option under **Permission model**.
2. On the **Networking** tab, leave the default settings.
3. On the **Tags** tab, enter the Environment and Project values configured for the Resource Group.
4. Click on the **Review + create** tab, ensure there are no validation errors, and then click on the **Create** button.
5. After the Key Vault has been successfully created, click on **Resource groups** in the **Left Pane**. Drill down on the Resource Group created above. Observe that the **Key Vault** is now listed under Resources.
6. You will return to Key Vault in later steps to make the needed entries as you provision the other Azure resources.

## Resource: Azure Data Lake Storage Gen2 Storage Account

Azure Data Lake Storage Gen2 (ADLS) makes Azure Storage the foundation for building data lakes on Azure. This section details the steps needed to create a Storage Account with hierarchical namespace enabled.

1. From the Left Pane, click on **Create a Resource**.
2. In the search bar at the top of the **Create a Resource** page, type in **Storage Account** and hit Enter.

Graphical user interface, text, application

Description automatically generated

1. Click on **Create** under the **Storage account** search result.
2. On the **Basics** tab, populate the fields as shown below.

A screenshot of a computer

Description automatically generated

1. On the **Advanced** tab, click on the checkbox next to **Enable hierarchal namespace**. This is the key configuration that enables ADLS capabilities in the storage container.

A screenshot of a computer

Description automatically generated

1. On the **Networking** and **Data protection** tabs, leave the default settings.
2. On the Tags tab, enter the Environment and Project values configured for the Resource Group.
3. Click on the **Review** tab, double-check your entries, and then click on the **Create** button.
4. After the **Storage Account** has been successfully created, click on **Resource groups** in the **Left Pane**. Drill down on the Resource Group created above. Observe that the **Storage Account** is now listed under Resources.
5. Drill down on the **Storage Account**. This will launch the **Overview** page for the resource.
6. Click on the **Containers** option in the menu pane.
7. On the **Containers** screen, click the **+ Container** button.
8. Add a container named **bronzezone**. Set the **Anonymous access level** setting to **Private (no anonymous access)**.
9. Add three more containers named **silverzone**, **goldzone**, and **landingzone** with the same access levels configured for the bronzezone.
10. You have now completed the initial provisioning and configurations needed for the data lake in this Resource Group. Browsing and modifying the data lake is cumbersome in the Azure portal. A more robust and user-friendly experience can be achieved using **Azure Storage Explorer**. This is a free desktop tool that can be [downloaded](https://azure.microsoft.com/en-us/products/storage/storage-explorer) from Microsoft.
11. The image below shows the newly created blob containers in the Explorer pane of **Azure Storage Explorer.**

A screenshot of a computer

Description automatically generated

## Resource: Azure Data Factory (ADF) Workspace

ADF is a data integration service used to develop ETL/ELT pipelines. In this section, only the ADF workspace itself will be provisioned. Pipelines will be cloned from a repo in a subsequent step.

1. From the Left Pane, click on **Create a Resource**.
2. In the search bar at the top of the **Create a Resource** page, type in **Data Factory** and hit Enter.
3. Click on **Create** under the **Data Factory** search result.
4. On the **Basics** tab, populate the fields as shown below.

A screenshot of a computer

Description automatically generated

1. Click on the **Git configuration** tab and ensure the checkbox for **Configure Git later** is checked.

Graphical user interface, text, application

Description automatically generated

1. On the **Networking** and **Advanced** tabs, leave the default settings.
2. On the **Tags** tab, enter the same name-value pairs that have been entered for the resources above.
3. Click on the **Review + create** tab, ensure there are no validation errors, and then click on the **Create** button.
4. After the Data Factory has been successfully created, click on **Resource groups** in the **Left Pane**. Drill down on the Resource Group created above. Observe that the **Data Factory** is now listed under Resources.

## Resource: Azure SQL Database Server

Azure SQL Database is a cloud service that runs on the latest version of the on-prem SQL Server database engine. The data integration Resource Group being configured in this document contains one Azure SQL database for the **Data Ingestion Framework** (DIF) database.

This database will be scoped to a single **Azure SQL Database server**, which is a logical construct used to manage groups of Azure SQL databases. The database itself will not be provisioned in this section as this custom database will be deployed via SQL Server Management Studio after the **Azure SQL Database server** resource is provisioned below.

1. From the Left Pane, click on **Create a Resource**.
2. In the search bar at the top of the **Create a Resource** page, type in **Azure SQL** and hit Enter.
3. Click on the **Create** button under the **Azure SQL** search result.
4. On the **Select SQL deployment option** screen, change the **Resource type** under **SQL databases** to **Database *server*** and click Create.

Graphical user interface, text, application

Description automatically generated

1. On the **Basics** tab, populate the key fields as shown below:
   1. Enter the **Server name** using the proper naming convention.
   2. Select the **Use both SQL and Microsoft Entra authentication** option
   3. Click on **Set admin** and select your Azure AAD account (now branded as Microsoft Entra) in the **Set Microsoft Entra admin** configuration.
   4. Save the server admin login and password somewhere. You will need it later.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Click on the **Networking** tab and set **Allow Azure services and resources to access this server** to **Yes**.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. On the **Additional settings** tab, leave the default options configured.
2. On the **Tags** tab, enter the same name-value pairs that have been entered for the resources above.
3. Click on the **Review + create** tab, double-check your entries, and then click on the **Create** button.
4. After the **Azure SQL Database server** has been successfully created, click on **Resource groups** in the **Left Pane**. Drill down on the Resource Group created above. Observe that the **SQL server** is now listed under Resources.

## Resource: Azure Databricks Workspace

Azure Databricks is an Apache Spark-based data engineering and analytics platform. This section details how to create a Databricks workspace.

1. From the Left Pane, click on **Create a Resource**.
2. In the search bar at the top of the **Create a Resource** page, type in **Databricks** and hit Enter.
3. Click on the **Create** button under the **Azure Databricks** search result.
4. On the **Basics** tab, populate the fields as shown below.

A screenshot of a computer

Description automatically generated

1. On the **Networking** and **Encryption** tabs, leave the default settings.
2. On the **Tags** tab, enter the same name-value pairs that have been entered for the resources above.
3. Click on the **Review + create** tab, double-check your entries, and then click on the **Create** button.
4. After the **Azure Databricks workspace** has been successfully created, click on **Resource groups** in the **Left Pane**. Drill down on the Resource Group created above. Observe that the **Azure Databricks Service** is now listed under Resources.
5. Drill down on the Azure Databricks Service link and click on **Launch Workspace**.
6. A new browser tab is opened and the Databricks UI is launched.
7. Click on the Compute item in the leftmost tab.

A screenshot of a computer

Description automatically generated

1. On the **All-purpose compute** tab, click on the **Create compute** button.
2. On the **Compute/New Compute** page, populate the fields exactly as shown below. Be sure to select these key options:
   1. Change the name of the cluster at the top of the page to **dbc-key2sandbox-dev-001** as it will display your name by default.
   2. Set Policy to unrestricted.
   3. Under **Databricks runtime version**, select the latest non-beta version.
   4. Uncheck the **Use Photon Acceleration** option and select **Standard\_F4** under Worker type.
   5. Change the **Terminate** **after** setting as it defaults to 120 minutes.

A screenshot of a computer

Description automatically generated

1. Verify your selections and click **Create Cluster**.

## Resource: Logic App to Send Email Alerts in ADF

Azure Logic Apps is a cloud service used to write and run automated workflows. Similar to Power Apps they can be created with very little or no code.

1. From the Left Pane, click on **Create a Resource**.
2. In the search bar at the top of the **Create a Resource** page, type in **Logic App** and hit Enter.

A screenshot of a phone

Description automatically generated

1. Click on **Create -> Logic App**.
2. On the **Basics** tab, populate the fields as shown below.

A screenshot of a computer screen

Description automatically generated

1. On the **Tags** tab, enter the same name-value pairs that have been entered for the resources above.
2. Click on the **Review + create** tab, ensure there are no validation errors, and then click on the **Create** button.
3. After the Logic App has been successfully created, click on **Resource groups** in the **Left Pane**. Drill down on the Resource Group created above. Observe that the **Logic App** is now listed under Resources.
4. Drill down on the Logic App.
5. Click on **Edit** to launch the Logic Apps Designer
6. Click on **When a HTTP request is received** to start designing the workflow.

A screenshot of a computer

Description automatically generated

1. Inside the object you will see the following:

A screenshot of a computer screen

Description automatically generated

Here is the JSON seen above – copy/paste it into the main box “Request Body JSON Schema” –

{

    "properties": {

        "DataFactoryName": {

            "type": "string"        },

        "EmailTo": {

            "type": "string"        },

        "ErrorMessage": {

            "type": "string"        },

        "PipelineName": {

            "type": "string"        },

        "Subject": {

            "type": "string"        }

    },

    "type": "object"

}

1. Below the 1st task click on **+New Step** and then search for **Office 365 Outlook**.

A screenshot of a computer

Description automatically generated

1. Then find **Send an email (v2)** from the subtasks.

A screenshot of a computer

Description automatically generated

1. The next step requires you to sign in using your Azure AD account.
2. You will now configure the send email task to use the JSON parameters configured in the When a HTTP request is received task

A screenshot of a computer

Description automatically generated

Body:

Data Factory Name: add dynamic content for DataFactoryName

Pipeline Name: add dynamic content for PipelineName

ErrorMessage: add dynamic content for ErrorMessage

Subject:

Add dynamic content for Subject

To:

add dynamic content for EmailTo

Importance is Normal

1. Finally, click on Save to capture the Logic App. The dynamic content will be passed from the ADF pipelines.

# Databricks Resource Configuration

We want our resources to be able to interact with one another. These steps allow Databricks to be able to interact with the Azure Data Lake Storage Account.

## Configure an Azure Key Vault-backed Secret Scope in Databricks

To reference secrets stored in an Azure Key Vault from within the Azure Databricks workspace, a secret scope backed by Azure Key Vault needs to be configured. The secret scope provides access to the secrets in the corresponding Key Vault instance from that secret scope.

1. If not already in the Databricks UI, click on the **Launch Workspace** button from the Databricks resource Overview page.
2. A new browser tab is opened and the Databricks UI is launched.
3. In the address bar of the browser copy the URL that is displayed to the clipboard.

Graphical user interface, text, application

Description automatically generated

1. Paste the URL into a text editor and select the Databricks **instance** as shown below.

Graphical user interface, text, application

Description automatically generated

1. Copy only the Databricks **instance** string to your clipboard and then paste the string into your text editor.

Graphical user interface, text, application, email

Description automatically generated

1. Replace the **<databricks-instance>** tag in the URL below with your instance string. This URL is case-sensitive; the scope in createScope must be uppercase.

https://<databricks-instance>#secrets/createScope

1. When replaced, the URL should look like the following example URL. Copy and paste this URL into a new tab of your web browser.

Graphical user interface, text, application, email

Description automatically generated

1. The **Create Secret Scope** page is now displayed in Databricks. Populate the fields as shown below. Note that the **DNS Name** and **Resource ID** values can be copied from the **Properties** tab of the **Key Vault** in the Azure Portal. The DNS Name and Resource ID correspond to the Vault URI and Resource ID fields of the Key Vault properties tab.

A screenshot of a computer

Description automatically generated

1. Click on the **Create** button.

## Create an Azure AD Application and Service Principal

For Databricks to access or modify the Storage Account resource, you will first need to create an Azure AD application and service principal.

The steps needed to create this application and service principal are detailed at this [location](https://learn.microsoft.com/en-us/azure/active-directory/develop/howto-create-service-principal-portal#register-an-application-with-azure-ad-and-create-a-service-principal) on Microsoft’s website. The name of the application should be configured as **srv-key2sandbox-databricks-dev**.

Be sure to use **Option 3**: **Create a new application secret**. Be sure to also copy the string under **Value** *immediately* after creating the secret. This string will only be available for copy just after creation. If you fail to copy the value, you will need to delete and create a new secret.

A screenshot of a computer

Description automatically generated

Be sure to copy the following values to a text editor as you perform these steps. You will need each one to store as a secret in the Key Vault.

1. Application (client) ID – displayed on the **Overview** page of the App registration
2. Directory (tenant) ID – displayed on the **Overview** page of the App registration
3. Application Secret - Secret Value shown on the Certificates & secrets page of the App registration

## Configure Key Vault for Databricks Mount

Three secrets will be configured in the Key Vault resource. A list of each secret name, along with its description is provided below:

1. **key2sandbox-adls-clientid** - Application (client) ID for the Azure AD application.
2. **key2sandbox-adls-tenantid** - Directory (tenant) ID for the Azure AD application.
3. **key2sandbox-adls-clientsecret** – Secret Value for the Azure AD application.

To add these Secrets, the following steps should be performed:

1. Navigate to the Overview screen of the Resource Group and drill down on the **Key Vault**.
2. Click on the **Secrets** option in the left-most menu pane for the **Key Vault**.
3. In the Command Bar across the top of this screen, click on +**Generate/Import**.
4. In the **Create a secret** screen, populate the fields as shown below. For the **Secret value** field, paste in Azure AD application keys that were copied to a text file in the previous section.

A screenshot of a computer

Description automatically generated

1. Click on the **Create** button and verify that your secret was created successfully.
2. Repeat this process for each of the other two secrets related to the Azure AD application.

## Configure AD Application as Storage Blob Data Contributor

Another step needed to enable Databricks access to ADLS is to configure the Azure AD application created above as a **Storage Blob Data Contributor** in the **Access Control (IAM)** page of the Storage Account Resource. You will need to be granted Owner access to the Resource Group to complete this step.

1. From the **Access Control (IAM)** screen of the stkey2sandbxdeveastus001 Storage Account, click on the **+Add** button, then select **Add role assignment**.

A screenshot of a computer

Description automatically generated

1. On the **Role** tab of the **Add role assignment** screen, type in *Storage Blob Data Contributor* in the search box on the **Job function roles** subtab, then select *Storage Blob Data Contributor* from the list below. Then click **Next**.

A screenshot of a computer

Description automatically generated

1. On the **Members** tab of the **Add role assignment** screen, click on **+Select members**, then type *srv-key2sandbox-databricks-dev* in the blade at the right side of the screen.
2. Select *srv-key2sandbox-databricks-dev* from the list below and click **Select**.

A white background with black and white clouds

Description automatically generated

1. Click on **Review + assign**

When complete the **srv-key2sandbox-databricks-dev** application should appear as follows on the **Role assignments** tab of the **Storage Account**.

A screenshot of a computer

Description automatically generated

## Mount ADLS on Azure Databricks

A mount point enables interaction with the files in ADLS from within a Databricks notebook. In a section above, four blob containers in the Azure Storage Account named **landingzone**, **bronzezone, silverzone,** and **goldzone** were created. Before these containers can be accessed from Databricks, a one-time Python script needs to be run. Mount points work by creating a local alias under the **/mnt** directory that stores the following information:

* Location of the cloud object storage.
* Driver specifications to connect to the storage account or container.
* Security credentials are required to access the data.

The following parameters are needed to create each mount point. These are examples:

1. The Storage Account name: **stkey2sandbxdeveastus001**
2. The Secret Scope name: **ss-kv-key2sandbox-dev-eus-001**
3. The Key Vault secrets related to the Azure AD application:
   1. **key2sandbox-adls-clientid** - Application (client) ID for the Azure AD application.
   2. **key2 sandbox-adls-tenantid** - Directory (tenant) ID for the Azure AD application
   3. **key2sandbox-adls-clientsecrect** - Secret Value for the Azure AD application.

The next steps detail how to mount the containers from a Databricks notebook.

1. Click on the **Launch Workspace** button from the Databricks resource Overview page.
2. A new browser tab is opened and the Databricks UI is launched.
3. In the leftmost pane, click on **Workspace**. Then click on your account under **Users**. Then select the **Import** option by clicking on the **three ellipses** on the right side of the screen.

A screenshot of a computer

Description automatically generated

1. Browse to the **documentation** folder in the unzipped **key2-demo-eia-main** file you sourced earlier from GitHub and select the **mount-storage.dbc** file. Click on **Import**.

A screenshot of a computer

Description automatically generated

1. The **mount-storage** notebook contains Python variables and string literals that need to be configured to mount a container to Databricks. The image below shows how these variables should be configured for the **bronzezone**.

A screenshot of a computer

Description automatically generated

1. Execute the Notebook by clicking on the **Run -> Run** **all** command. After running the notebook for the bronzezone, reconfigure the variables and rerun the notebooks for the **silverzone**, **goldzone**, and **landingzone**.

# ADF Resource Configuration

We want our resources to be able to interact with one another. These steps allow ADF to be able to interact with the Azure Data Lake Storage Account, call rest APIs, and access Databricks

## Configure ADF Workspace as Storage Blob Data Contributor

To enable the ADF Workspace to access ADLS, the workspace must be configured as a **Storage Blob Data Contributor** in the **Access Control (IAM)** page of the Storage Account resource. When complete, the ADF resource (workspace name) should appear as follows on the **Role assignments** tab.

A screenshot of a computer

Description automatically generated

## Configure ADF Workspace as a Contributor to ADF Workspace

To enable the ADF Workspace to call the Azure REST APIs, the workspace must be configured as a **Contributor** in the **Access Control (IAM)** page of the ADF Workspace. When complete, the ADF resource (workspace name) should appear as follows on the **Role assignments** tab of the **ADF Workspace**.

A screenshot of a computer

Description automatically generated

## Add ADF Workspace as a Contributor to Databricks

To enable the ADF Workspace to access Databricks, the workspace must be configured as a **Contributor** in the **Access Control (IAM)** page of the Databricks resource. **Contributor** is under Privileged Admin Role. When complete, the ADF resource (workspace name) should appear as follows on the **Role assignments** tab of the **Databricks Workspace**.

A screenshot of a computer

Description automatically generated

# Key Vault Secrets

You want to lock down access to the database, the storage account, and access to the API. To do so, you will need to create secrets in the Key Vault.

## Create Secret for ADLS Access Key

A secret will be configured in the Key Vault that stores the **Access Key** for the ADLS **Storage Account**. The Access Key is used by a Linked Service in ADF to read/write files during pipeline execution.

1. From your resource group page, drill down on the **stkey2sandbxdeveastus001** Storage Account. This will launch the **Overview** page for the resource.
2. Click on the **Access keys** option in the menu pane for the **Storage Account**.

On the **Access keys** page, click on the **Show** button next to the **Key** field of **key1**. This will remove the input mask and display a **Copy to clipboard** button.

A screenshot of a computer

Description automatically generated

1. Click on the button and copy the key to a text file as it will be needed in the next steps.
2. Navigate back to the Overview screen of the Resource Group and drill down on the **Key Vault**.
3. Click on the **Secrets** option in the menu pane for the **Key Vault**.

In the Command Bar across the top of this screen, click on **Generate/Import**.

1. In the **Create a secret** screen, populate the fields as shown below. For the **Secret value** field, paste in the Storage Account access key that was copied earlier to a text file.

A screenshot of a computer

Description automatically generated

1. Click on the **Create** button and verify that your secret was created successfully.

## Create Secret for SQL Service Accounts

A secret will be configured in Key Vault to store the password for the Azure SQL database account created above:

1. Navigate to the Overview screen of the Resource Group and drill down on the **Key Vault**.
2. Click on the **Secrets** option in the menu pane for the **Key Vault**.

In the Command Bar across the top of this screen, click on +**Generate/Import**.

1. In the **Create a secret** screen, populate the fields as shown below. For the **Secret value** field, paste in the password you configured earlier for the Azure SQL server.

A screenshot of a computer

Description automatically generated

1. Click on the **Create** button and verify that your secret was created successfully.

## Create Secret for EIA API Key

A secret will be configured in the Key Vault to store the key for the EIA API. This is a free service, and you can register for a personal key by following this [link](https://www.eia.gov/opendata/documentation.php) and clicking on Register.

1. Navigate to the Overview screen of the Resource Group and drill down on the **Key Vault**.
2. Click on the **Secrets** option in the menu pane for the **Key Vault**.

In the Command Bar across the top of this screen, click on +**Generate/Import**.

1. In the **Create a secret** screen, populate the fields as shown below. For the **Secret value** field, paste in the API key.

A screenshot of a computer

Description automatically generated

1. Click on the **Create** button and verify that your secret was created successfully.

# Configure Access Policies for ADF in Key Vault

For ADF to securely read the secret values stored in the Key Vault, it is necessary to configure the permission model in the Key Vault. The Key Vault created earlier in this document was provisioned with the **Vault access policy** option.

1. From your Resource groups page, drill down on the **Key Vault**. This will launch the **Overview** page for the resource.
2. Click on the **Access policies** option in the menu pane of the Key Vault.

In the Command Bar across the top of this screen, click on **+ Create**.

1. On the **Permissions** tab, check the checkboxes for **Get** and **List** under the **Secret permissions** column and click **Next**.

A screenshot of a computer

Description automatically generated

1. On the **Principal** tab, type in the name of the **adf-key2sandbox-dev-eastus-001** resource in the text box. Click on **adf-key2sandbox-dev-eastus-001** below the text box and ensure it is displayed as the **Selected item** at the bottom of the page.

A screenshot of a computer

Description automatically generated

1. On the **Review + create** tab, review the selected configurations and then click on the **Create** button.
2. The **adf-key2sandbox-dev-eastus-001** resource should now appear under Application on the Access policies page.

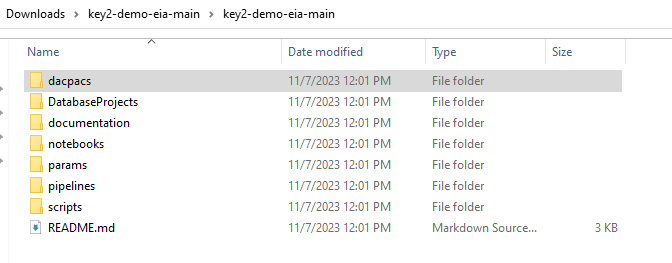
A screenshot of a computer

Description automatically generated

# Import DIF Azure SQL Database

In an earlier step, an Azure SQL server was provisioned. The next step is to import the DIF Azure SQL database to this server.

1. Using windows explorer, navigate to the **dacpacs** subfolder in the unzipped key2-demo-eia-main file you sourced earlier from GitHub:



1. In the **dacpacs** folder, locate the file named **sqldb\_dev\_key2demo\_difconfig.dacpac** and rename it to **sqldb\_dev\_key2sandbox\_difconfig.dacpac .**
2. Connect to the **sql-key2sandbox-dev-eastus-001.database.windows.net server** that was created earlier in this document. Be sure to use **SQL Server Authentication** and the creds for the **Server admin login** you created in the “Create an Azure SQL Database Server” section above

A screenshot of a computer

Description automatically generated

1. If you are prompted about creating a New Firewall Rule, navigate to the **Networking** screen of the **sql-key2sandbox-dev-eastus-001** resource in the Azure Portal and add your client to the **Firewall rules** section, then click **Save**.

A screenshot of a computer

Description automatically generated

1. You should now be able to successfully log into the **sql-key2sandbox-dev-eastus-001** Azure SQL server in SSMS.
2. Right-click on Databases in Object Explorer and select **Deploy Data-tier Application**

A screenshot of a computer

Description automatically generated

1. On the **Select Package** screen of the **Deploy Data-tier Application** screen, click on Browse and select the .dacpac file that was renamed earlier, and click on Open.

**sqldb\_dev\_key2sandbox\_difconfig.dacpac**

1. Click on **Next**. Be patient for a minute or two as SSMS communicates with Azure.
2. Click on Next two more times until you are brought to the Deploy DAC screen and wait another couple of minutes while the .dacpac is deployed.
3. After the .dacpac is deployed successfully, click Finish.
4. **Important: Immediately after the .dacpac is deployed, navigate to the sql-key2sandbox-dev-eastus-001 resource in the Azure portal.** Failure to do so will result in an oversized Azure SQL database that is costly and not needed for sandbox purposes. The remaining steps will resize the Azure SQL database that was just deployed.
5. On the SQL databases screen, you should now be able to see the sqldb\_dev\_key2sandbox\_difconfig Azure SQL database. Drill down on this database.

A screenshot of a computer

Description automatically generated

1. In the left pane, click on the **Compute + storage** option and adjust the **Service tier, DTUs, Data max size,** and **Backup storage redundancy** options as shown below.

A screenshot of a computer

Description automatically generated

1. Click on Apply. It will take a couple of minutes for the databases to be resized before your changes are reflected in the Azure portal.

# Connect GitHub Repo to ADF and Databricks

Now that Azure resources have been provisioned, the next step is to connect to a GitHub repository so that ADF pipelines and Databricks Notebooks can be pulled in. However, a feature branch must first be created in GitHub that is based on the latest version of the Key2 demo EIA main branch.

As was mentioned earlier, the resource group and source code described in this document are intended to be used for sandbox purposes only. Do not follow these instructions for development purposes.

1. Using a web browser, Navigate to the Key2 repository for the EIA demo in GitHub:

https://github.com/Key2Consulting/key2-demo-eia

1. Drill down on the **branches** option

A screenshot of a computer

Description automatically generated

1. On the branches screen, click on New branch.

A white rectangular object with black text

Description automatically generated

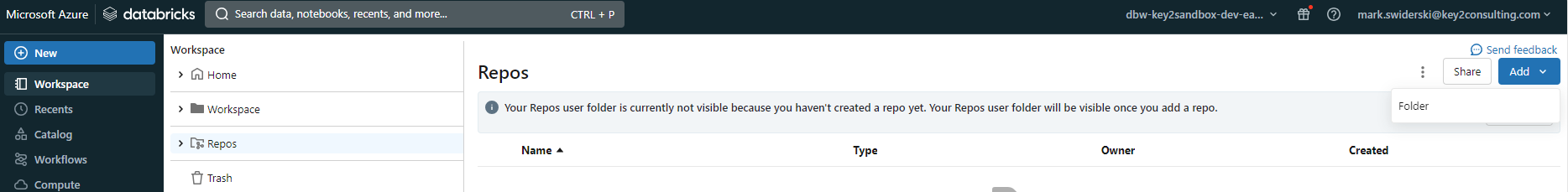
1. In **New branch name**, enter a name with the following pattern and ensure the **Source** is set to **main**:

YYYYMMDD\_firstinitiallastname\_eia\_sandbox\_donotpull

A screenshot of a computer

Description automatically generated

1. Click on **Create new Branch**
2. In the Azure Portal, launch the dbw-key2sandbox-dev-eastus-001 **Databricks** workspace.
3. In the left pane, click on Workspace -> Repos -> Add -> Folder



1. In the New folder screen, type in **Shared** (Databricks is case-sensitive), then click on **Create**.

A white background with blue lines

Description automatically generated

1. Ensure the **Shared** folder is selected under Repos, then click on Add -> Repo

A screen shot of a computer

Description automatically generated

1. In the Add Repo screen, make the following configurations and click on Create Repo. You can copy/paste the Git repository URL from GitHub:

A screenshot of a computer

Description automatically generated

1. Select the **Link Git account** option on the next screen and click on **Link**.
2. On the following screen, click on **Configure in GitHub**. This will launch a new browser window that guides you through installing the Databricks application to your personal GitHub account. When complete, return to Databricks and click on **Clone**.

A screenshot of a computer

Description automatically generated

1. After the repo is cloned to the Databricks workspace, you will observe it is listed under the Shared folder in Repos. You will also observe that a **notebooks** folder is now present in the browser. You’ll now need to change the branch from main to the one you created earlier in GitHub Click on the **key2-demo-eia repo** in the Shared repo folder. Then click on **main**.

A screenshot of a computer

Description automatically generated

1. In the drop-down in the upper left corner of the ke2-demo-eia screen, select the branch you created earlier in GitHub. After selecting it, close the screen by clicking on the X in the upper right corner.

A screenshot of a chat

Description automatically generated

1. Observe that this branch is now active in the key2-demo-eia share repository in Databricks. Do not change this setting in your sandbox environment.

A screenshot of a computer

Description automatically generated

1. In the Azure Portal, launch the adf-keysandbox-dev-eastus-001 ADF workspace.
2. In the left pane, click on Manage -> Git configuration – Configure

A screenshot of a computer

Description automatically generated

1. In the **Configure a repository** blade that opens at the right, select **GitHub** as the **Repository type** and enter **Key2Consulting** as the **GitHub repository owner**. Click on **Continue**.

A screenshot of a computer

Description automatically generated

1. On the next screen,
   1. Select **key2-demo-eia** as the **Repository name**.
   2. Ensure that you select the branch you just created in GitHub as the **Collaboration branch**.
   3. Append the name of your branch to the **Publish branch** which is defaulted to adf\_publish. **Do not leave the Publish branch as adf\_publish**!
   4. The Root folder should also be set to **/pipelines**.
   5. Check the boxes for **Custom comment** and **Import existing resources**.
   6. When complete, your configurations should be similar to what is entered below.

A screenshot of a computer

Description automatically generated

1. Click on Apply
2. After the repository has been connected successfully, click on Author in the left pane and observe that Pipelines and other resources are now present under Factory Resources. Observe that the branch is set to the branch created earlier in GitHub.

A screenshot of a computer

Description automatically generated

1. Click on **Validate all**. Observe that no errors were found.
2. Click on Publish. This will publish the collaboration branch resources to the Data Factory service.

# ADF Configuration

Setting up Linked Services

## Configure Environment Specific Settings in ADF

Now that ADF has been connected to the feature branch, you will need to change environment-specific connection settings.

1. In the Left Pane in ADF, click on Manage -> Linked Services.
2. Click on the **LS\_Core\_Databricks\_pool** linked service

A screenshot of a computer

Description automatically generated

1. In the **Edit linked service** blade that opens at the right, update the following fields:
   1. **Databricks Workspace URL** – copy/paste from the **URL** field on the **Overview** screen of Databricks resource in the Azure portal.
   2. **Workspace resource ID** – copy/paste from the **Id** field on the **Properties** screen of Databricks resource in the Azure portal.

A close-up of a computer screen

Description automatically generated

* 1. **Choose from existing clusters** – select the **dbc-key2sandbox-dev-001** option.

A screenshot of a computer

Description automatically generated

1. Click on **Test connection** and if successful, click on **Save**.
2. Click on the **LS\_DIF\_KeyVault** linked service.
3. In the **Edit linked service** blade that opens at the right, update the following fields:
   1. Base URL field – copy/paste from the Vault URI field on the **Overview** screen of the Key Vault resource in the Azure portal.

A screenshot of a computer

Description automatically generated

1. Click on **Test connection** and if successful, click on **Save**.
2. Click on the **LS\_DIF\_Metadata** linked service.
3. In the **Edit linked service** blade that opens at the right, update the following fields:
   1. **Full qualified domain name** – copy/paste from the Server name field in the **Overview** screen of the **SQL database** resource in the Azure portal
   2. **Database name** – copy/paste from the title in the Overview screen of the **Azure SQL database resource** (Same location as above) in the Azure portal (sqldb\_dev\_key2sandbox\_difconfig)
   3. **User name** – copy/paste from the **Server admin login** field on the **Properties** screen of the Azure SQL database resource in the Azure portal.
   4. **Secret name** – the name of the secret in Key Vault used to store the password for the User name configured above (key2sandbox-sql-secret-dev-az)

A screenshot of a computer

Description automatically generated

1. Click on **Test connection** and if successful, click on **Save**.
2. Click on **Validate all**. Observe that no errors were found.
3. Click on **Publish**. This will publish the collaboration branch resources to the Data Factory service.

## Overwrite live mode in ADF

Because the factory resources were cloned from a repo, as opposed to creating them directly or importing the resources via PowerShell, you’ll need to manually overwrite the live mode of the Git configuration in ADF.

1. In the Left Pane in ADF, click on **Manage -> Git configuration**.
2. Click on **Overwrite live mode**.

A screenshot of a computer

Description automatically generated

1. Click OK on the **Preview overwrite live mode?** prompt.

A screenshot of a computer

Description automatically generated

1. Click on Overwrite in the Pending changes blade that has opened on the right.

# DIF Database Configuration Explained

It is recommended that the user open and read “Key2 Data Ingestion Framework Overview” in the documentation first, as this goes into the best detail of the role the DIF Database plays. From that point on, follow the resources listed at the bottom of that document.

## Configure DIF Database

The next configuration-related step involves populating the tables in the DIF Azure SQL database with configurations related to the EIA demo project.

1. Navigate to the scripts\DIF folder in the unzipped **key2-demo-eia-main** file you sourced earlier from GitHub.

A screenshot of a computer

Description automatically generated

1. Using SSMS, run the SQL files in ascending order by name against the **sqldb\_dev\_key2sandbox\_difconfig** database.
2. The first file, **00\_Key2Demo\_DIF\_postdeploy.sql**, does not require any changes to be made before executing the script.
3. All other files require that you first review the script and make environment-specific changes. For example, in the below snippet, you will need to make changes so that the **@SystemName**, **@SystemFQDM,** and **@SystemSecretName** params are updated to reflect those values that apply to the resource group you’ve created using this document.

A screenshot of a computer program

Description automatically generated

1. Before running these scripts (and eventually the ADF pipelines), you must first read through the documentation related to the DIF and associated ADF pipelines. These documents can be found in the documentation folder of the repository.

A screenshot of a computer

Description automatically generated

# Data Source Configuration

The project has two data sources. Information gathered from the EIA API and flat file information in the form of a CSV. See below for setup.

## Configure Parameter Files for EIA Pipelines (ADF)

The ADF pipelines that consume the EIA API do not use configurations in the DIF to store values for pipeline execution. Instead, JSON files are used to store these configurations. These files can be found in the **params** folder of the repository you downloaded in a previous step.

A screenshot of a computer

Description automatically generated

1. Detailed descriptions for each parameter and the EIA API pipelines themselves can be found in the document named **Key2 Data Ingestion Framework - ADF Pipeline Overview.docx**. Use this document to guide you through updating the parameters with values that are specific to your resource group, extract timeframe, etc.
2. After updating the configuration files, be sure to upload them to the appropriate folder in the storage account. Be sure to also create a storage container in ADLS and mount point in Databricks to the storage account location where the files are located. For example, in the below image, the param files have been uploaded to a folder named **eiaapi\_params** in a container named **params**.

A screenshot of a computer

Description automatically generated

## Download Source Files for Landing Zone

The EIA API is used to source the majority of transactional energy data from the EIA. However, there are a few master file data sources that use CSV files as the data source because this data is not available via the API.

These files are available via download from the EIA website. However, we’ve placed these files in a Teams files share so they are more readily available for sandbox purposes.

1. Navigate to the following Teams location and download each folder to your workstation.

A screenshot of a computer

Description automatically generated

1. Unzip each folder and then upload these files to the **landingzone** in the storage account configured in this document. For example, when finished, the files in the **eia860** folder should appear as follows in the **landingzone**.

A screenshot of a computer

Description automatically generated

# Delta Live Tables

Our EIA pipeline in ADF gathers the data via the API and puts it into a landing zone. From there, the data needs to be in the data lake. The parameters must reflect the destination in the landing zone that the pipelines will call. For example:

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Pipeline Settings

See below for how the pipelines should look

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Trigger ADF Pipelines

All needed configurations have now been performed and you are ready to start triggering the ADF pipelines to further your discovery about how data integration is performed in Azure. The pipelines sourced in the repo have been configured with default parameters which will allow you to trigger them easily.

1. In the ADF Workspace, click on **Author**. Expand **Pipelines**, then expand the **DIF DelimitedFile** folder and click on the **PL\_DIF\_DelimitedFile\_GroupOrder** pipeline.
2. Click on **Add trigger -> Trigger now**.

A screenshot of a computer

Description automatically generated

1. In the Pipeline run blade that has opened on the right side of the screen, click OK.

A screenshot of a computer

Description automatically generated

1. In the leftmost pane, click on Monitor and observe that the pipeline has successfully started executing.

A screenshot of a computer

Description automatically generated

1. After all pipelines have been successfully executed, it is highly recommended that you return to the Author canvass and study all activities in the pipelines. Do this for all pipelines in the workspace.

## Trigger Order

The trigger order should be:

1. PL\_EIA\_API\_Main
2. The following pipelines: A screenshot of a computer

   Description automatically generated
3. Execute the following in the SQL Server Editor. “CREATE SCHEMA eia\_gold”
4. PL\_DIF\_DelimitedFile\_GroupOrder
5. PL\_DIF\_Lakehouse\_GroupOrder