Investigating Third-Party Environments

HPCC Systems and Microsoft Azure Synapse Analytics

Table of Contents

1. Introduction………………………………………………………………….3
2. Analyzing the HPCC Data Tutorial………………………………………….4
3. Exploring the Azure Synapse Analytics Environment…………………….10

1. Deploying the Synapse Analytics Environment………………………….11
2. Uploading, Querying, and Visualizing files within Synapse Analytics……12
3. Establishing the SFTP connection………………………………………….14
   1. Demonstrating the Copy Data functionality with SFTP connection...17
4. Using REST API connection to Roxie Cluster for JSON file querying……22
5. Recreating the HPCC Data Tutorial Steps over Azure Synapse Analytics using Integrated Pipelines and Dataflow…………………………………...24

Introduction

This procedural and technical documentation discusses the integration of third-party environments with the HPCC Systems Platform. This document will focus on the Microsoft Azure Synapse Analytics environment.

Scope

TBD

Prerequisites

Particular prerequisites to interface this project involve understanding bare metal and web-hosted deployments of particular applications. A strong knowledge of scripting and querying is also recommended and eventually required. Specifically, the HPCC Systems is deployed locally as a single node cluster. Then, a company subscription to Microsoft Azure is required to access the target environment.

A screenshot of a computer

Description automatically generated

Bare Metal HPCC Deployment

For this part of the project, installing the HPCC Platform requires us to install the Windows Subsystem for Linux (WSL) first. Specifically, by version, WSL2. The official documentation: [Installing and Running the HPCC Systems Platform](https://cdn.hpccsystems.com/releases/CE-Candidate-7.0.0/docs/EN_US/Installing_and_RunningTheHPCCPlatform_EN_US-7.0.0-beta2.pdf). We can optionally use the ECL IDE or host the ECL framework over our local system. The ECL IDE can have preferences set to a local build of the Platform on our single-node cluster.

Relevant Research and Exploration

1. Our first area of research is related to dataset examination using the HPCC Platform. This is to be done given the preference that our deployment is local and within our ECL IDE. Using this official documentation: [HPCC Systems Data Tutorial](https://cdn.hpccsystems.com/releases/CE-Candidate-6.0.6/docs/HPCCDataTutorial-6.0.6-1.pdf), I was able to understand and dissect the functions of a Thor – raw, large data management – cluster and a Roxie – indexed data management – cluster. In my experience, this tutorial allows the developer to understand how workunits are published, hosted, and managed over the ECL framework.
   1. Initially, we have to upload our target data file into the Platform landing zone. Here we have an example file ‘OriginalPerson’ that is sprayed (imported) and uploaded to the landing zone: A screenshot of a computer

      Description automatically generated
   2. Now, we create several files that we are going to target to either the Thor or Roxie cluster for processing. Here, we use the following code to process and write raw data to the Thor by accessing the file we uploaded earlier and its corresponding output:A screenshot of a computer program

      Description automatically generatedA screenshot of a computer

      Description automatically generated
   3. Next step is to index:



* 1. Build a Query:

A computer code on a white background

Description automatically generated

* 1. Publish your Thor Query and as an example here is code to index our data by Zip:

A close-up of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

* 1. Follow the same steps, but use the indexed file to target Roxie:

A screenshot of a computer

Description automatically generated

Then Publish the query as before and run the Thor and Roxie queries on the local IP addresses mapped by the IDE:



Having a solid foundation on how data upload, movement, and interpretation works over the Platform, we can explore our Azure and Azure Synapse options now.

1. The second resource that was of importance was to initiate a Cloud Native Platform using our Ubuntu WSL2 command line interface setup and communication over the Azure platform.

**NOTE: This documentation is relevant for Azure Kubernetes Cluster deployments but some steps are relevant to deploying components required for Azure Synapse Analytics’ use case. So, these steps have been modified to accommodate Synapse environment integration.**

* 1. First, we use the command ‘az login --use-device-code’ to point to our directory and subscription over Azure.
  2. Next, we can create a resource group to manage all applications within Azure in our subscription over the CLI:

az group create --name hpccsynapse-rsg --location eastus --tags owner=“shounak” owner\_email=[shounak.joshi@lexisnexisrisk.com](mailto:shounak.joshi@lexisnexisrisk.com) deployment-method=cli

* 1. Next, we can focus on the storage account utility to copy subdirectories from our HPCC Platform repository to create and access HPCC Systems file shares under the same storage account within our resource group.

Subscription -> Resource Group -> Storage Account -> File Share.

In order to do this we need to clone our directory and copy all relevant file share content through CLI:

git clone [https://github.com/hpcc-systems/HPCC-Platform.git](https://github.com/hpcc-systems/HPCC-Platform.git/)

git checkout community\_8.4.22-1

mkdir ~/cloud

cp -r HPCC-Platform/helm/examples/azure ~/cloud

cp HPCC-Platform/helm/hpcc/values.yaml ~/cloud/azure

* 1. Now, we must change our environment configurations by accessing and editing the code within the env-sa (environment-storage account):

*# SUBSCRIPTION=*

STORAGE\_ACCOUNT\_NAME=dlsshounakj

SA\_RESOURCE\_GROUP=hpccsynapse-rsg

TAGS=’owner=”shounak” owner\_email=”shounak.joshi@lexisnexisrisk.com”'

*# Set the same location as Kubernetes cluster*

SA\_LOCATION=eastus

SA\_KEY\_DIR=’./keys’

SA\_SKU=Standard\_LRS

*# These settings are also for generating PersistentVolume.*

*# Settings in ../hpcc-azurefile/values.yaml will have higher priority.*

*# If you will set these with "--set" for helm install you need set them*

*# here also if you want to generate storage account, storage share and*

*# Kubernetes secret.*

SECRET\_NAME=hpccsynapsesecret

SECRET\_NAMESPACE="default"

SHARE\_NAMES="dalishare dllshare sashashare datashare lzshare"

* 1. Next, we configure our values.yaml using the command:

“Code ~/cloud/azure/values.yaml” to include the relevant ECL services to our environment

esp:

- name: eclwatch

*## Pre-configured esp applications include eclwatch, eclservices, and eclqueries*

application: eclwatch

auth: none

replicas: 1

service:az

*## port can be used to change the local port used by the pod. If omitted, the default port (8880) is used*

port: 8888

*## servicePort controls the port that this service will be exposed on, either internally to the cluster, or externally*

servicePort: 8010

*## Specify visibility: local (or global) if you want the service available from outside the cluster. Typically, eclwatch and wsecl are published externally, while eclservices is designed for internal use.*

visibility: global

*## Annotations can be specified on a service - for example to specify provider-specific information such as service.beta.kubernetes.io/azure-load-balancer-internal-subnet*

*#annotations:*

*# service.beta.kubernetes.io/azure-load-balancer-internal-subnet: "mysubnet"*

*## You can also specify labels on a service*

*#labels:*

*# mylabel: "3"*

*#resources:*

*# cpu: "1"*

*# memory: "2G"*

- name: eclservices

application: eclservices

auth: none

replicas: 1

service:

servicePort: 8010

visibility: cluster

*#resources:*

*# cpu: "250m"*

*# memory: "1G"*

- name: eclqueries

application: eclqueries

auth: none

replicas: 1

service:

visibility: global

servicePort: 8002

*#resources:*

*# cpu: "250m"*

*# memory: "1G"*

- name: esdl-sandbox

application: esdl-sandbox

auth: none

replicas: 1

service:

visibility: local

servicePort: 8899

*#resources:*

*# cpu: "250m"*

*# memory: "1G"*

- name: sql2ecl

application: sql2ecl

auth: none

replicas: 1

service:

visibility: local

servicePort: 8510

*#domain: hpccsql.com*

*#resources:*

*# cpu: "250m"*

*# memory: "1G"*

* 1. Now, we have to modify our hpcc-azurefile values.yaml file to configure our storage account and secret using the command “code ~/cloud/azure/hpcc-azurefile/values.yaml”:

*# Default values for hpcc-azurelfile.*

common:

mountPrefix: "/var/lib/HPCCSystems"

secretName: "hpccdemosecret"

secretNamespace: "default"

planes:

- name: dali

subPath: dalistorage

size: 1Gi

category: dali

sku: "Standard\_LRS"

shareName: dalishare

- name: dll

subPath: queries *# cannot currently be changed*

size: 1Gi

category: dll

rwmany: true

sku: "Standard\_LRS"

shareName: dllsshare

- name: sasha

subPath: sasha

size: 1Gi

rwmany: true

category: sasha

sku: "Standard\_LRS"

shareName: sashashare

- name: data

subPath: hpcc-data *# cannot currently be changed*

size: 3Gi

category: data *# NB: all "data" planes will be auto mounted by engine components and others that require access to data*

rwmany: true

sku: "Standard\_LRS"

shareName: datashare

- name: mydropzone

subPath: dropzone

size: 1Gi

rwmany: true

category: lz

sku: "Standard\_LRS"

shareName: lzshare

* 1. Finally, we can deploy our Storage Account and secret using the following commands one after the other:
     1. ~/cloud/azure/sa/create-sa.sh (Storage Account)
     2. ~/cloud/azure/sa/create-secret.sh (Storage Account Secret)

1. Exploring the Azure Synapse Analytics environment

Firstly, we can look at our resource group, “hpccsynapse-rsg”: NA screenshot of a computer

Description automatically generated

Here, we have our Azure Synapse Analytics workspace initialized and our storage account. The steps to initialize our Synapse Workspace are as follows.

**NOTE: The resource group must be deployed – with the appropriate tags – prior to creating the Synapse workspace. This was done previously through the Ubuntu WSL2 CLI. Once the resource group is deployed, we deploy the Synapse workspace – with the appropriate tags – and then assign the workspace to the created resource group. THESE MUST BE DONE SEPARATELY AND IN THIS ORDER or else we will encounter an error where resource provisioning is ‘DisallowedByPolicy’ because we have not initialized the tags (owner, owner\_email) properly, according to the subscription policy.**

**Here are the required categories to fill out to deploy our workspace:**

**A screenshot of a computer

Description automatically generated**

Now that we have initialized our Synapse workspace, we can look at how data is managed within Synapse to generate relevant analytics.

1. First, we can manually upload data to our generated file system within our storage account inside our resource group displayed in our workspace:

A screenshot of a computer

Description automatically generated

1. Next, we can write a SQL script that allows us to display and generate some analytics relevant to the dataset contained within the sample file:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

To generate more meaningful insights, we can query our data to visualize analytics. For example, in a file that contains sales data, we can find out which fiscal period and price point generates the most revenue by indexing and outputting the relevant data point within our accessed dataset through querying.

1. **Now, to connect our Azure/Synapse Analytics environment with HPCC, we have established a Secure File Transfer Protocol (SFTP) to encrypt and load files into our Azure Data Lake Storage Gen2. In order to this, we have a generated SSH key and credentials that map to our ECL Watch server that contains our HPCC Systems data files, landing zones, and logical files. The specific web-hosted IP for this, in my case, is in the format “XX.XXX.XXX.XXX:8010” in (IP Address:Port) format. Simply paste this into your web browser to access ECL Watch.**

A screenshot of a computer

Description automatically generated



Notice how we are in our landing zone, with a folder called “synapseshounak”. This folder contains our files that we want to be transferred to the Synapse Analytics environment through our SFTP connection. In the next step, we will look at how these are deployed.

1. **Establishing the SFTP connection as a Linked Service**
   1. In order to create our “source” as SFTP, we can select it as our desired linked service. Here we can connect via the default integration runtime using the Host: format “XX.XXX.XXX.XXX”and Port: 22 by default. We can enable SSH host key validation alongside our fingerprint, uploading our file that contains the private key content.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

1. **Establishing Azure Data Lake Storage Gen2 as the destination data storage (linked service)**
   1. In order to create our destination as a data storage, we can use Azure Apps to find a data storage service called Azure Data Lake Storage Gen 2. Here, we only need our storage account mapping and the secure key.

A screenshot of a computer

Description automatically generated

1. To link our storage account to our workspace, **we can set it up by default**, but we can also create private network endpoints to connect these two resources. **I have created these endpoints for demonstration and secure connection purposes.** Essentially, the workspace is one end and the storage account within the same resource group is another endpoint that will allow us to connect both resources. Here is the resource group visualizer:

A screenshot of a computer

Description automatically generated

1. Now that we have our connectors, let's look at our ECL Watch Landing Zone as our SFTP source data connection. This is our source folder that we will ingest from into Synapse: A screenshot of a computer

   Description automatically generated
2. \*\*\***On Synapse, we will use the Copy Data feature to establish our connectors, ingest, and access data within our file system\*\*\***
   1. First, we can setup a built-in copy task

A screenshot of a computer

Description automatically generated

* 1. Now, we setup our source data store connection

A screenshot of a computer

Description automatically generated

* 1. Now, follow the path to the folder

A screenshot of a computer

Description automatically generated

* 1. Maintain the CSV or desired file format settings

A screenshot of a computer

Description automatically generated

* 1. Now, we have our destination data storage “Azure Data Lake Storage Gen2” that maps to the file system in our storage account. This is still connected via the private network endpoint.

A screenshot of a computer

Description automatically generated

* 1. Now, we can finalize our settings to enter deployment state

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* 1. Notice how all files within the folder in ECL Watch have been successfully copied into the Synapse workspace via the data lake storage connection

A screenshot of a computer

Description automatically generated

To display to a user, this diagram shows how HPCC Systems connects with our Synapse environment:

A diagram of a diagram

Description automatically generated

A screenshot of a cell phone

Description automatically generated

4) Establishing the Roxie connection to Synapse Analytics Environment using REST API

Similar to the SFTP connection established to the HPCC landing zone, this Roxie connection creates GET and POST JSON requests. By using the proper Headers and JSON Test content, we can recreate Roxie functionality over Synapse using the linked service connection. Below is an image of this connection as a linked service:

A screenshot of a computer

Description automatically generated

In order to query a JSON file format, we need to manipulate SQL syntax;

This is a file loaded in as test.json. It can be loaded through a copy data function through linked services to the landing zone or be uploaded manually through local drive.

A screenshot of a computer program

Description automatically generated

The results successfully display as a relational table:

A screenshot of a computer

Description automatically generated

5) Recreating the HPCC Data Tutorial Steps over Azure Synapse Analytics using Integrated Pipelines and Dataflow

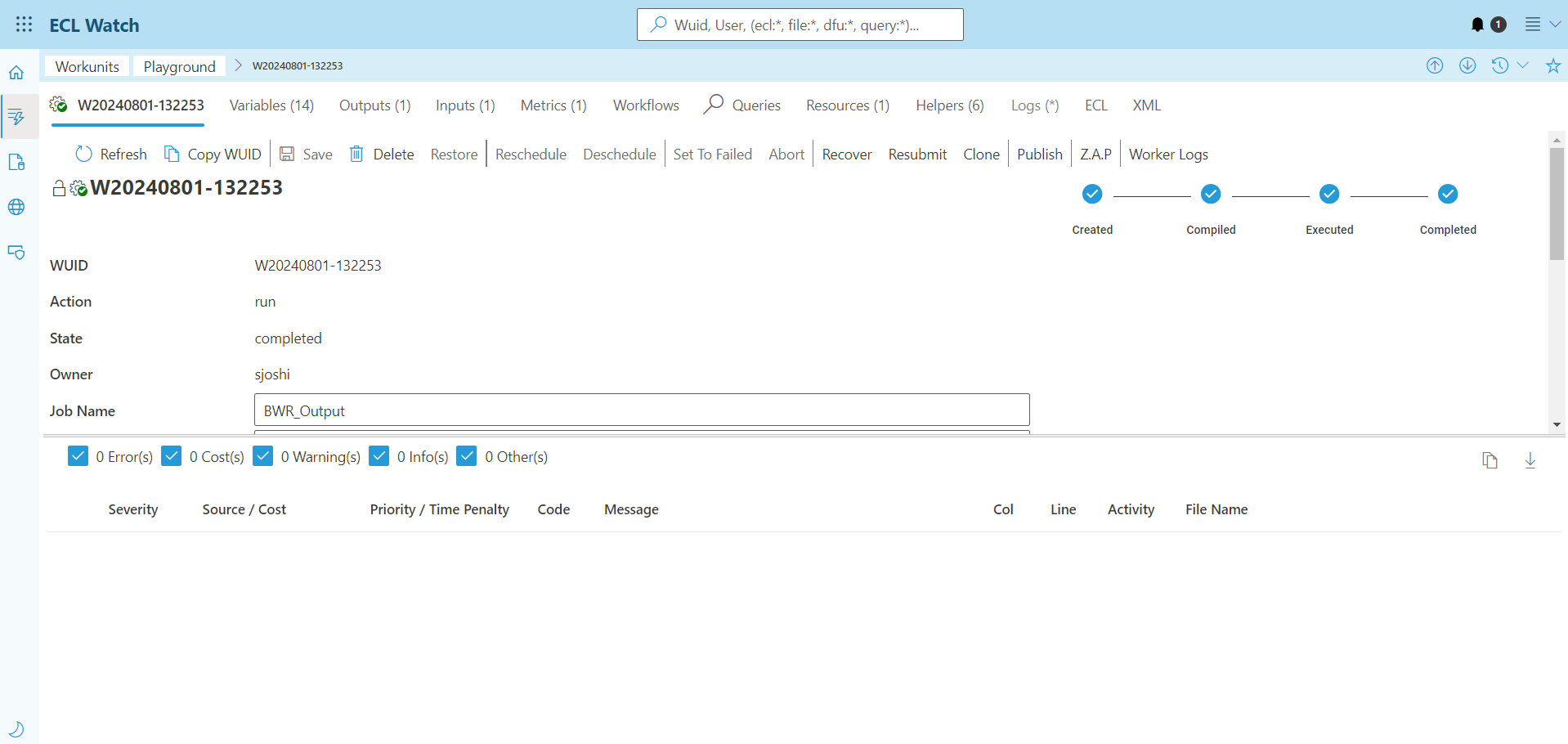
To recreate the HPCC Data Tutorial, I chose to use the same file “OriginalPerson” containing firstname, lastname, middlename, zip, street, city, and state column attributes.

1. As a first step, we need to convert the OriginalPerson file into a file format that can be displayed easily as a relational table. The following code changes in a new file help with this:

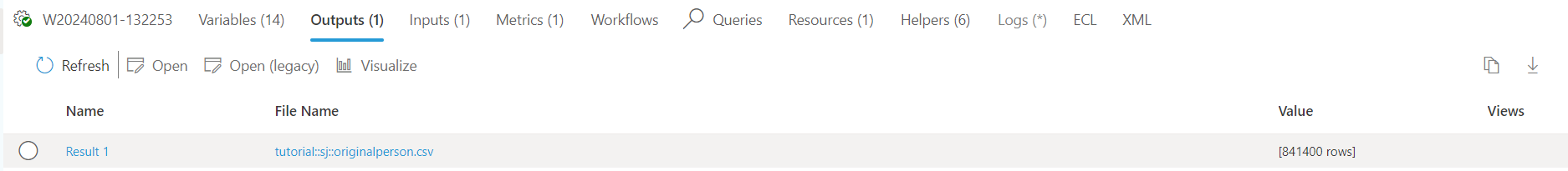
A screenshot of a computer

Description automatically generated

1. After this is done, we can publish this query to ECL Watch and get this:



Then, our output becomes a CSV file converted from the OriginalPerson binary file with no header:



Then, we can locally download the OriginalPerson file or test the copy data tool to our landing zone. In my case, for efficiency purposes, I decided to locally download and manually upload the file to recreate the data tutorial steps.

1. Then, we can leverage Synapse’s integrated pipelines and data flow features used for ETL and data delivery.
2. First, we have to initialize a pipeline that contains our data flow. Within the data flow, we set a “source1” to intake an integration dataset called “DelimitedText1” whose schema is configured by the OriginalPerson.csv file.

A white rectangular object with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Since our file does not have column headers, we can add these in the projection tab.

A screenshot of a computer

Description automatically generated

1. As a part of data manipulation, the data tutorial capitalizes all entries apart from the integer data type. All string entries are uppercase. We can do this using the schema modifier option called derivedColumn1.

A close-up of a document

Description automatically generated

A screenshot of a computer

Description automatically generated

Scrolling down...

A screenshot of a computer

Description automatically generated

The “upper” function allows us to pass in each column name as defined in our projection. **Also, at any point, we can enable the Data flow debug and Refresh the Data preview to see the output of our modifications.**

1. Next, we ingest the derivedColumn1 output into a filter1 row modifier transformation. Here, we can leverage parameters. To set data flow parameters, we can add this “zipCodeParam” parameter here:

A screenshot of a computer

Description automatically generated

A white rectangular object with black text

Description automatically generated

Setting the “Filter on” option for the zip column as “zip == $zipCodeParam” allows us to parameterize our data and filter rows using expressions on the zip column.

A screenshot of a computer

Description automatically generated

1. At this point, our transformation has been complete and is now ready for a destination called “Sink1”

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

This will output all data for our zip code as one file instead of using a default partition. Execution time is longer with this process but this will organize data with one trigger.

1. Now, we use our parameterized data flow and link it to trigger our pipeline. First, we need to set a “zipCodePipelineParam” parameter for our pipeline as an Int data type and dynamic input value upon triggering the pipeline.

A screenshot of a computer

Description automatically generated

Now, clicking on our data flow within the pipeline, we head to the Parameters tab:

A screenshot of a computer

Description automatically generated

Specifically, this is done using the Pipeline expression builder

A screenshot of a computer

Description automatically generated

1. Now, we can trigger our pipeline and monitor it in the Monitor tab on Synapse Analytics. Since this is a large file that contains ~800,000+ records, it took 1m 25s to execute.

A screenshot of a computer

Description automatically generated

1. For our purpose, I inputted 23832 as the example zip code. The output file was linked to my file system as part of the sink2 dataset linked service to the Azure Data Lake Storage Gen2.

When queried, this file outputs the same relational table as the Roxie output of our HPCC Data Tutorial.

A screenshot of a computer

Description automatically generated

Here is the code along with the relational table for querying and visualization:

A screenshot of a computer

Description automatically generated

This is the full diagram of our data flow to replicate the HPCC Data Tutorial:

A screenshot of a computer

Description automatically generated