**Steps Involved In creating Pipeline:**

**Data Ingestion using Azure Blob Storage:**

The population dataset is loaded from the local into storage account as a gzipped file and then it is copied into data Lake Gen 2 as .tsv file. [Resources used Created Storage account and loaded the zip file by uploading into a container. Created data lake and created raw container folder for uploading the file.]

Create Azure Data Factory and based on the pipeline structure first create Linked Services under manage tab(ls\_ablob\_name). These linked services help establish connection between the storage account, data lake with adf for performing the copy activity.

Created a pipeline named(pl\_injest\_population\_data). During Creation I have set the concurrency to 1. Used copy data module for copying data. In the General section of the copy data name retrial count and Timeout option I have set them. I have connected the source and sink datasets to the copy activity and validated and debugged the data copying activity.

A screenshot of a computer

AI-generated content may be incorrect.

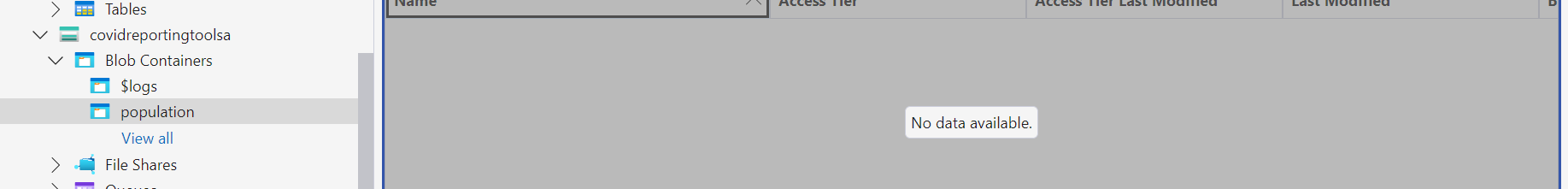
Used Validation activity before copy data activity. Purpose of validation activity came into picture because in Realtime scenarios the file can be updated daily based on the data available. To get the data at certain times and validate the time of execution we can use the validation activity. Whether the file is present or not, the present can be known using the Validation activity block.

Can check by performing the source file deletion the activities will fail due to time out.

A screenshot of a computer

AI-generated content may be incorrect.

Failed Case



I have Set the Timed Out Feature For 30 secs and the file is not that large so it got failed.

A screenshot of a computer

AI-generated content may be incorrect.

Used Get meta data activity in order to check the meta data for the file.[column count, size, exists for validating purpose].

A screenshot of a cell phone

AI-generated content may be incorrect.

If condition activity for checking if anything is breaking or not.

Using add dynamic content check if the column count getting from the get metadata activity is correct or not then only perform copy activity.

I have implemented the If activity for validating the metadata before performing the copy behavior. True validates to Copy behavior and False validates to failure case.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

In if condition checked if the file has column count of 13 or not.

A screenshot of a computer

AI-generated content may be incorrect.

Did the Fail activity to check its purpose I have adjusted the condition in the If activity.

A screenshot of a computer

AI-generated content may be incorrect.

For False case fail activity is used.

Used Delete activity as soon as the copy activity is executed. Delete activity deletes the source file.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

I have used the event-based trigger for the pipeline Upon the source file insertion the pipeline execution starts.

A screenshot of a computer

AI-generated content may be incorrect.

After deletion I am able to see the logging as it is enabled in the setting in the Delete Activity.

Implemented Storage Events Trigger which is better suitable case. When the file falls in the Storage account the trigger will be clicked automatically and the pipeline is activated and will transform the file into next steps.

**Data Ingestion from HTTP:**

<https://raw.githubusercontent.com/cloudboxacademy/covid19/refs/heads/main/ecdc_data/cases_deaths.csv>

<https://raw.githubusercontent.com/cloudboxacademy/covid19/refs/heads/main/ecdc_data/country_response.csv>

<https://raw.githubusercontent.com/cloudboxacademy/covid19/refs/heads/main/ecdc_data/hospital_admissions.csv>

<https://raw.githubusercontent.com/cloudboxacademy/covid19/refs/heads/main/ecdc_data/testing.csv>

The process goes this way to create a linked service, source and sink datasets. Then create a pipeline where all the data is being copied.

Implementing Parameters and Variables In Pipeline to import multiple datasets at the same time by passing the URL as a parameter into linked service.

Make dataset parameters for relative URL and filename as parameterized and in the pipeline initialize variables and declare values to them.

Debug the pipeline for the hospital \_admissions.csv

Now Generalize the cases deaths datasets and pipeline into ECDC. As the parameters are used in the pipeline it can be used for any dataset.

Created a schedule trigger and passed the parameters from the trigger into the pipeline in order to get the hospital admissions data.

Used Control flow activity as there can be many files for importing. We can use lookup activity and foreach activity for the retrieval of the URLs from the Json Document.

Made the baseURL as a parameter all the way from linked services, dataset, pipeline, copy activity for all of them created a parameter and dynamically assigned the parameter.

Trigger is set and used Look up and foreach deleted all the pipeline parameters as they are no longer needed and changed the copy activity parameters of sink and source to for each activity output and input of the foreach will be lookup activity.

A screenshot of a computer

AI-generated content may be incorrect.

I have used the URLs as a set of json objects and passed to the pipeline using the Lookup activity and iterated every value using foreach activity and passed values to the copy ECDC Data activity.

A screenshot of a computer

AI-generated content may be incorrect.

**Data Transformation:**

Created a Dataflow for the cases deaths dataset. Transformations used in the data flow are source, filter, select, pivot, look up, sink and at the end create pipeline for its execution.

We need to create a data flow where we can perform many transformations but every dataflow must contain the source and sink transformations.

You can have as many transformations as possible in between the source and sink transformation. Microsoft instructed us to maintain a minimum number of transformations.

These can be tested on debug clusters. As the transformation is converted into spark and it is executed on a Microsoft data bricks cluster which is a distributed execution cluster.

Dataflow for Cases and deaths.

Examine the data in the cases deaths dataset we have all countries data but for this lecture we only used Europe data so filter the data. Country code is 3 digits but, in some files, it is 2 digits, so both exist in the file. We have indicators and daily count. Instead, we can maintain confirmed cases count and deaths count to maintain standard rows. Replace date column to reported date and remove rate 14 days.

For performing the country code transformation, we are going to use a lookup file. We big cases deaths dataset for validation purpose in the debug we used country lookup for the debug [It contains data of GBR(UK) and India based on which validation is done].

Implementing Source transformation and sink transformation at least once is mandatory. In source it needs to be linked, it can be inline or dataset. Can be chosen if you need to use source in multiple dataflows better go for dataset and inline has some limitations it is supported only in few.

In source settings we have options for allowing or limiting schema drift which means changes in the schema.

Sampling generally dataflow picks few samples from source and validates them. In the case we are taking the validation and using in the debug settings as we took country-based cases and deaths for UK and India.

In source options we have options like after transformation deleting source or moving the transformation to new file.

Projection brings all the column names and its types and makes them detect using internal inhibition.

In the inspect and preview we can see the exact output after transformation.

A screenshot of a computer

AI-generated content may be incorrect.

Start Debug first Data Transformation of Cases and Deaths file. Here in step, we used source and named it in which we have dataset of choice dataset or inline. Choosing dataset gives an advantage of using it for multiple use cases. But Inline is having very limited support. Schema options and sampling data options. Schema if validating and any new columns arrived it won’t accept. Sampling is set to disable as in this we are doing sampling manually. Projection will detect the Datatype of the column automatically. Inspect Tab will show the schema of the output after transformation and preview will show the sample data.

In Filter Activity use the Visual editor for doing necessary manipulations where all the necessary transformations can be done.

Select Transformation Where we have the necessary columns, we work and you have rule-based mapping and mapping. In rule-based mapping one can take advantage of visual studio and perform necessary transformations. We have an option for removing duplicate columns. In our activity deleted continent and rated-14days and renamed reported date using rule based mapping.

Pivot Transformation for pivoting the columns of indicator and daily counts into confirmed cases count and deaths count. All the non-participating columns will fall under group by section and pivot key is the converting columns. The way they can be merged can be looked using the pivot columns where you can navigate to the visual code editor and write the code.

Lookup Transformation where we used to include the country codes of 2 digit and 3 digit. Create the source dataset for country lookup and use lookup transformation and make necessary changes and provide the condition on which they needs to be combined.

Created a sink and added a container to keep all the processed or transformed data in it.

A close-up of a diagram

AI-generated content may be incorrect.

Select Transformation is used to add or remove the fields of the lookup transformation after its flow of the data.

Conditional split is required in our data because it has the weekly and daily data. We can divide the data based on daily and weekly also for the weekly data we have date value as null. We can redefine the start and end date of the week using the custom dimdate lookup file.

For dimdate source transformation we used derived transformation which can be used to modify or create a new column where it is exactly match to create our necessary columns in original existing columns with the mapping of the dimdate column.

A computer screen shot of a computer

AI-generated content may be incorrect.

After this Step we need to prepare the input data to run on the HDInsight Activity. It is very much similar to Spark. Spark runs on distributed clusters same way HDInsight works. These expect the input data as folder and they divide the data into chuncks and works with it. In Data Factory You can work directly with file or the dataset but in HDInsight it expects the input as a folder.

**Cluster Type**- Single Node

**Node Type** - Standard\_D4a\_v4

Create a Azure Databricks Service using the subscription available. On creation it will take you to data bricks workspace. Upon Launching the Workspace. You will be navigated to a different workspace environment. It uses Azure ID for logging in.

Azure Databricks clusters are created. A cluster is nothing but a computation resources generally databricks consists of one driver node and one or more worker nodes. We can create All purpose/Interactive clusters and Job Clusters(These are automatically terminated).Once terminated cannot be restarted.

New🡪Create Cluster🡪Give Name🡪Choose Single node as you are in free tier🡪performance choose which has LTS[Long term support]🡪Set the timer to defaultly delete the cluster🡪Choose node type where you have costing for this choice.

For using data lake in azure databricks🡪create azure service principle🡪grant access for datalake to azure service principle🡪create mount in databricks using service principle.

I have Created the Azure Service Principle, attached it to datalake and created tokens for datalake to attach to azure databricks for mounting the data.

I have executed the Mounting python Script and kept the containers that needs to be processed on Databricks.

In ADF Created pipeline and run the databricks cluster by adding the custom cluster created.

Transformation on Population. We only need 2019 population so flush out all the other years. Dataset consists of Age group and country code which needs to be split and worked on.

After Transformation Copied all the data sql databases. Created a Sql database and used azure data studio to access the database. Connected sql server to azure data studio.

Inorder to copy the data from datalake to sql we can use the facility of the pipeline creation where use the copy data activity and take the source as processed dataset and sink as the sql table[Create the linked service and dataset for the sink as well.]. Integrate the dataset with sql and point the test connection using the linked service. Thus, linked service are used to point to the different storage accounts. It is like the variable declared in a class for all the dataset.

For setting the source give the name and give created dataset and use the wildcard path it automatically picks the cases deaths or hospital admissions or testing. In the sink Write the Precopy script as Truncate[This helps if every time we run pipeline again and again the data won’t get duplicated in the sql]. For testing I use mapping and it helped to map the source column. In between in the dataset we have unmapped columns so map the data accordingly.

**Data Orchestration:**

Requirements: Pipeline Executions are full automated. Pipelines run at regular intervals or on an event occurring. Activities only run once the upstream dependency has been satisfied. Easier to monitor for execution progress and issues.

Capability: Dependency between activities inside a pipeline, Dependency between pipelines within a parent pipeline. Dependency between triggers [only Tumbling window triggers]. Custom-made solution.

Create a Pipeline to run Population data.

For Population Ingestion I have Created Ingest Pipeline where when a file is kept in the pointed storage account it will automatically trigger the pipeline using the storage event trigger when the file arrives with the mentioned starting name. It can be Connected with the Processing pipeline where a pipeline is created with the custom Databricks Cluster. Upon the file arrive first pipeline is executed and then I am attaching the databricks cluster to it so that it will be triggered after the first trigger happened. This make the Complete Pipeline Orchestration for Population Data.

Processed Population Data is only invoked upon completion of the ingest population Data. Inorder to invoke the Ingest Population Data use event trigger.

In the actual plan we have the processing pipeline to be executed on databricks cluster.But as the free tier is not able to execute or accommodate the cluster so I used the dataflow which is built in Azure data Factory.

Data Orchestration: [Option 2 Trigger Dependency]

I have Created the Trigger based Orchestration where all the triggers are created for each of the pipeline which is a tumbling window trigger. And attached and added the dependency to one after the other. Tumbling window is created for five of them name for cases and deaths, Hospital admissions pipelines[First pipeline triggered is ecdc filelist🡪Second pipeline triggered is processed cases deaths and hospital admissions🡪Third is Sqlize cases deaths and hospital admissions].

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

In the Advanced Settings There is Offset and window size these can be use when you have yesterday’s data processed today and todays data processed tomorrow.

**Monitoring**

Things that we can monitor Azure Fata Factory Resource [size, total objects]. Integration runtime [CPU, Available memory], Trigger runs [Success or fail, Issue fix], Pipeline runs and activity runs.

Data Factory Monitor: Ability to monitor status of pipeline/triggers, can be used to rerun failed pipelines/triggers, Ability to send alerts from base level metrics, Provides base level metrics and logs, Pipeline runs are stored only for 45 days.

Azure Monitor: Ability to route the diagnostic data to other storage solutions, Provides richer diagnostic data, Ability to write complex queries and custom reporting, Ability to report across multiple data factories.

In the monitor tab we can create a Email or SMS using Alert and metrics tab in ADF Monitor Tab. You can also report using the Metrics tab under the alert and metrics section.

Create New Alert metrics🡪Give Name🡪Add the Severity🡪 Add the target criteria and set the time how many times or what time gap for failure that you wanted to know🡪Configure whether you want a email or mobile notification.

You Can rerun Complete pipeline or you can rerun from where the pipeline got failed.

Better running the trigger completely fixes when there is dependent triggers as we did the tumbling window triggers. These depend on one another. So better run complete trigger to potrey it as a successful trigger.

You Check all the metrics under the alert and metrics and metrics tab show the all metrics that you can filter. If there is any failure you pick the option and check them. In the monitoring tab you can pin the metrics and save them to your dashboard.

In the Azure monitor tab you can look for the Diagnostic setting where you can look for the resource that you wanted to capture all the metrics and store them in resource you wanted to store. If you need only specific days outcome lets mentioning about retention period.

For creating log analytics workspace. Log analytics workspace🡪Give name🡪Choose RG🡪Select the pricing tire🡪Review create🡪Create After creation search for diagnostic settings in adf and add log analytics workspace where the logs will be analyzed. Set Destination tables to resource specific.

Goto Log analytics🡪Logs🡪It will give you queries and lots of information about the log analytics. For Querying in Log Analytics workspace, it uses Kusto Query Language.

KQL is very similar language to SQL but it has few differences.

A screenshot of a computer

AI-generated content may be incorrect.

Kusto Query Language comes into handy when there is Large number of Pipelines you want to query it.

After the Query Execution You will be able to get the Graphs and Add them to the dashboard that you where working on.

Kusto Query language has an option of generating the charts based on the query execution. One can implement bar charts code in the queries itself.

Introduction to Power BI Desktop

**Continuous Integration/Continuous Delivery:**

Characteristics Of Devops: Collaboration, trust and transparency, Agile Development Approach, Continuous Integration/ Delivery, Automation and Continuous Improvement.

For any project

Plan🡪 Code🡪Build🡪Test 🡪 Release🡪Deploy 🡪Improve🡪Monitor-->Plan

Continuous Integration Continuous Delivery Continuous Improvement

/Deployment

For a Java Project we get .class file Here in ADF we get JSON and ARM Template. Testing of the ARM Template can be done using .NET Code.

**Option 1 Manual Integration/ Automated Delivery**

For manual build used in ADF previously they used ADF publish and use release scripts in Deployments but now Microsoft introduced a package for deployment which will automate the build in deployment pipelines.

Azure DevOps Consists of Boards, Repos, Pipelines, Test Plans, Artifacts.

Azure DevOps Consists of Organization. This can be One or more for a company based in the Business Units and Each organization may contain one or more projects in it as well.

Creating new Data Factories with Name Format [[dev-ci-cd-demo-adf-vchalla3](https://portal.azure.com/#resource/subscriptions/3fe03faf-37af-4ca6-b4b1-9ca27b7eb8c4/resourceGroups/dev-ci-cd-demo-rg/providers/Microsoft.DataFactory/factories/dev-ci-cd-demo-adf-vchalla3)] Dev🡪Test🡪Prod environments for the DevOps Purpose.

Now Goto Repos section in the Azure DevOps and create a git branch with the same synonym names that you can recognize them directly.

Set the policies for the main branches that it cannot be accessible directly. Once a feature branch is created we can develop pipeline and debug pipeline and create Pull Request and review and approve Pull Request and move the code to main branch.

Simple Demo For the Pipeline creation with Wait activity for demo now you can validate and debug but you won’t be able to publish as now you were in feature branch whereyou can only commit changes to it and once done check in Azure Repo you will be able to see the json code of the pipeline.

Once the changes are committed you can go give for the pull request from the adf which directory you are working that directory. And once done come to Azure Devops you will be able to see the Pull Requests and review it and approve. Usually in an organization a teamlead or the other person will review and approve the code.

For you after the approval it can be merged You have to complete the merge then only you will be able to see the changes in the main branch. Every time we create a branch and saves it then the changes will be committed to the branch. If you wanted to update the branch do the same procedure again [Create new branch🡪Save the changes🡪create pull request🡪see the changes and approve🡪Complete the merge.]

You can publish the repo by clicking the publish in main branch and it will create the adf publish where all the changes in the repo will be stored under the ARM Templates. Before publishing the changes check the live mode you won’t be able to see any of the changes or pipelines in live and now after publishing you can see the changes in the Live mode with the created pipeline as the Publish will generate the ARM Templates.

After Publish in the Repo you can be able to see the adf publish branch newly created. Check the branch it will consists the ARM templates and necessary dependency folders in it.

Add task and select the deployment template type using for the pipeline which is ARM template. Edit the settings where you will need to create Azure Service principle and select the resource group which is test-ci-cd-demo-vchalla3-rg.

Select the linked artifacts and the branch. Create a variable once the variable is created which means that it can be used for the other pipeline also.

One creation of the Artifact template you can do the manual trigger [Create release] and run the pipeline. After first manual deployment we have chosen the option of continuous deployment which means if there is any changes in the branch from a feature branch and then it is merged then from the main branch you publish it will then do auto deployment of the changes will happen which means the pipeline will run once there is an execution of the change in ARM Template.

When the Objects are deleted or the active triggers are updated it will give an error.

Example:2 In release\_1 delete the second pipeline and for now create a trigger and we will update in the second release. Create a tumbling window trigger and save it. Create a pull request and go azure devOps to approve it. Approve the pipeline which includes the trigger and pipeline deleted in it.

When there is object deletion that won’t be happening in the test because ARM template will only include any objects, but it won’t be deleting them. Coming to the trigger created it will be created but we need to start it and deploy it.

Now make changes in the trigger it will automatically be saved but when published it will be failed in the deployment process build will be failed as an active trigger cannot be updated it will not be changed as it was active.

For fixing the issue in the deployment process you will be needing pre-deployment scripts and post deployment scripts.

Microsoft already has the scripts repo in github it has to types of scripts one is where the pre and post deployment PowerShell scripts will stop the active triggers whether it has changed in it or not. The second one will stop the one trigger that needs to be changed. Microsoft has documentation for everything it provided the commands to run the scripts as well.

Create a feature branch and create new folder [release folder]. Commit/upload the file to folder. Then create a pull request and merge to main. Add the file as the artifact to the pipeline. Add the PowerShell template and make necessary changes to make it active. Provide script path and arguments[Command] and give pre and post deployment arguments. We have pre and post deployments to work on based on that create two templates. Later on merge the changes to main branch.

Next stage is adding the production pipeline to the actual pipeline and remove the static values and create the pipeline variables.

Hardcoded values are kept in ARM template. Like name of resource group, location where it got deployed and name of data factory. Now we create the variables for all these values and change them when we extend this pipeline to production stage.

When you select pipeline environment you have variables section where you will be able to choose the pipeline variables or variable groups. For this project we use only the Pipeline variables for three things that we discussed. By adding we can just fix the scope of the variable based on the test or production. Make changes in using the $() where the actual values are present to make the pipeline independent of variables. These changes are made in Pre deployment and Post deployment. Save and create manual release instead of auto deployment, Click on current pipeline and add a new stage and you just need to change only the pipeline variables for the production environment. Clone the stage not copy it. Just change the variables for the production environment. We need to give access to the service principle to grant access for the production deployment. Add the service principle for the prod resource group. Also when you clone the test make sure to include the pre deployment condition for the prod because we wanted to have approval before the deployment is done.

Now for testing purpose you can create a test pipeline where we will be able to commit the changes to it. But the prod will not be participating in the continuous deployment as it needs and approval for a person who we gave approval access to. After releases you can check whether the pipeline got deployed or not. In this project we used pipeline variables but there is option for variable groups which will give more insights about the data.

In option 1 of implementation we cannot automate everything but we will try automating them in option 2. In option 1 after we approve the pull request we often want to go to the env and click on publish button to generate the ARM Template to get the changes to be published from the main branch to successive branches. We can automate the publish and generation of the arm template after a successful build of the pipeline which is done by npm package. Microsoft provided necessary documentation for the implementation of the NPM package.

Read Automated solution for the build pipeline to invoke npm package based on the successful build.

It is Good practice to create a feature branch and then push changes to the main branch. For pushing the build related support files create a feature branch and create folder named build and add a readme file where every file will be included later on. For the purpose of automating the build there is a script provided by Microsoft to automate the pipeline deployment using successful build that code is provided in a YAML file.

Create a new pipeline. Clone it and rename from the first one and later delete the artifact which is coming from repository and add the new build pipeline that is created using YAML file. Change the ARM Template location that is previously in the repository now point it to the ARM template generated by

In the new pipeline ARM Template is coming from build pipeline rather than from adf publish branch so we need to delete the artifact and create a new one which is pointing to build pipeline. In the stage also we need to change the pointing ARM template goto tasks and change ARM Template. Copy path of the ARM Template and change in Pre and Post deployment templates as well. Do the Changes to test and prod environments. Create a dev stage from test clone it. Change the order of the stages to dev test prod to avoid errors.

Before deployment give access to service principle for the resource group to avoid any errors. Deploy and check all the stages. Before we have done the manual release to make the continuous deployment change option in artifact to deploy after build is completed. Run and check whether it is working or not.

When we are working in the release pipeline scenario. When we use dev environment then dev env will be used like the data lake or SQL server will be used from the dev env so for test and prod those needs to be changed where those needs to be pointed to their respective environments. For those we need to provide access permissions and also we need to create them and attach them in the artifacts.

For the data lakes or any other resources that we are creating make naming conventions same as data factory.