A diagram of a software development process

Description automatically generated

**CI/CD Flow:**

1. Each user makes changes in their private branches.
2. Push to master isn't allowed. Users must create a pull request to make changes.
3. The Azure DevOps pipeline build is triggered every time a new commit is made to master. It validates the resources and generates an ARM template as an artifact if validation succeeds.
4. The DevOps Release pipeline is configured to create a new release and deploy the ARM template each time a new build is available.

**CI/CD step by step:**

**Step 0:**

* **As we need hosted parallelism to run pipeline in step 2, this can be requested for free by filling out the form at below link, then submit, and await for approval. Usually it takes up to 24 hours to obtain the approval.**
* [**https://aka.ms/azpipelines-parallelism-request**](https://aka.ms/azpipelines-parallelism-request)**.**

**A screenshot of a computer

Description automatically generated**

**Step1:**

Create 3 Data Factory located within 3 different resource group:

1/ For Development

2/ For Test

3/ For Production

**Step 2:**

Data Factory Git Configuration using Dev Ops Portal

A screenshot of a computer

Description automatically generated

Select branch under Repo, then click on the 3 dots sign to create branch policy

A screenshot of a computer

Description automatically generated

Go to Development Data Factory to set up the Git depository

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

We cant change the name of the publish branch, it’s always be **adf\_publish**.

**A screenshot of a computer

Description automatically generated**

*No worry to see this error, it’s meant we’re not allowed to work directly on the main branch, we’re required to work on the feature branch.*

**Step 3: Continuous Integration:** this step can be done either in the Dev Ops portal or the Data Factory UI

* **Create feature branch in Development Data Factory**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

The branch was created with feature/feature01 to group all of the feature branch under the folder namely feature for ease of reference.

A screenshot of a computer

Description automatically generated

It’s confirmed that we’re at the feature branch.

* **Develop pipeline**

**A screenshot of a computer

Description automatically generated**

* **Debug pipeline**

**A screenshot of a computer

Description automatically generated**

Debug the pipeline, then click save (the publish button was greyed out as it’s required approval for the change to take effect).

A screenshot of a computer

Description automatically generated

* **Create Pull Request**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

either ways is fine.

A screenshot of a computer

Description automatically generated

* **Review, approve Pull Request**

**A screenshot of a computer

Description automatically generated**

* **Complete, and Merge to the main branch**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

Only the main branch remain, the feature branch was merged to main branch, and deleted thereafter.

**Step4:** Create a Build Pipeline which will be triggered by the change at the main branch. Basically this Build Pipeline will validate the resource, and generate the ARM template.

create packge.json, and CI\_CD\_pipeline.yml files. Detail can be found on below link

* [Automated publishing for continuous integration and delivery - Azure Data Factory | Microsoft Learn](https://learn.microsoft.com/en-us/azure/data-factory/continuous-integration-delivery-improvements)

For package.json

{

    "scripts":{

        "build":"node node\_modules/@microsoft/azure-data-factory-utilities/lib/index"

    },

    "dependencies":{

        "@microsoft/azure-data-factory-utilities":"^1.0.0"

    }

}

For and CI\_CD\_pipeline.yml:

*# Sample YAML file to validate and export an ARM template into a build artifact*

*# Requires a package.json file located in the target repository*

trigger:

- main *#as soon as there's merge to the main branch, the trigger will work.*

pool:

  vmImage: 'ubuntu-latest'

variables:

  subscriptionId: '665fb058-e95e-4ae8-a1ef-cbc4c3f97955'           *#my subscsription ID*

  resourceGroup: …………………..            *#resource group for the development data factory*

  dataFactory: '……………………………'              *#development data factory name*

  PackageFolder: 'build'    *#the git folder under which we have the package.json file*

  adfRootFolder: ''         *#as it's root folder, we leave it blank. This is the folder where we have the ADF resource*

steps:

*# Installs Node and the npm packages saved in the package.json file in the build*

- task: UseNode@1

  inputs:

    version: '18.x'

  displayName: 'Install Node.js'

- task: Npm@1

  inputs:

    command: 'install'

    workingDir: '$(Build.Repository.LocalPath)/$(PackageFolder)' *#replace with the package.json folder*

    verbose: true

  displayName: 'Install npm package'

*# Validates all of the Data Factory resources in the repository. You'll get the same validation errors as when "Validate All" is selected.*

*# Enter the appropriate subscription and name for the source factory. Either of the "Validate" or "Validate and Generate ARM temmplate" options are required to perform validation. Running both is unnecessary.*

- task: Npm@1

  inputs:

    command: 'custom'

    workingDir: '$(Build.Repository.LocalPath)/build' *#replace with the package.json folder*

    customCommand: 'run build validate $(Build.Repository.LocalPath)/$(adfRootFolder) /subscriptions/$(subscriptionId)/resourceGroups/$(resourceGroup)/providers/Microsoft.DataFactory/factories/$(dataFactory)'

  displayName: 'Validate Data Factory Resource'

*# Validate and then generate the ARM template into the destination folder, which is the same as selecting "Publish" from the UX.*

*# The ARM template generated isn't published to the live version of the factory. Deployment should be done by using a CI/CD pipeline.*

- task: Npm@1

  inputs:

    command: 'custom'

    workingDir: '$(Build.Repository.LocalPath)/<folder-of-the-package.json-file>' *#replace with the package.json folder*

    customCommand: 'run build export $(Build.Repository.LocalPath)/$(adfRootFolder) /subscriptions/$(subscriptionId)/resourceGroups/$(resourceGroup)/providers/Microsoft.DataFactory/factories/$(dataFactory) "ArmTemplate"'

*#For using preview that allows you to only stop/ start triggers that are modified, please comment out the above line and uncomment the below line. Make sure the package.json contains the build-preview command.*

*#customCommand: 'run build-preview export $(Build.Repository.LocalPath) /subscriptions/222f1459-6ebd-4896-82ab-652d5f6883cf/resourceGroups/GartnerMQ2021/providers/Microsoft.DataFactory/factories/Dev-GartnerMQ2021-DataFactory "ArmTemplate"'*

  displayName: 'Validate and Generate ARM template'

*# Publish the artifact to be used as a source for a release pipeline.*

- task: PublishPipelineArtifact@1

  inputs:

    targetPath: '$(Build.Repository.LocalPath)/$(PackageFolder)/ArmTemplate' *#replace with the package.json folder*

    artifact: 'ArmTemplates'

    publishLocation: 'pipeline'

  displayName: 'Publish ARM template'

Once done, commit, approve, complete, and merge to the main branch.

A screenshot of a computer

Description automatically generated

Create a feature branch in the repository to add these 2 files to the repository, then switch to the feature branch.

A screenshot of a computer

Description automatically generated

Then go to file, and open a Build folder

A screenshot of a computer

Description automatically generated

Commit the change, then upload the 2 files to Build folder, then again create a Pull request to approve, and complete the upload.

Create a Build pipeline as below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Click on the drop down, and select save

A screenshot of a computer

Description automatically generated

the Build pipeline was given a name of the repository, therefore we should change the name of the pipeline to something more easier of reference. Click on the below 3 dots to change the name of the Build pipeline.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Run the Build pipeline:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1 artifact was created, and once we click on it, we find the ARM template as below:

A screenshot of a computer

Description automatically generated

**Step 5:** Create Release Pipeline which will pull the artifact from the Build Pipeline to deploy in Dev ADF, Test ADF and Production ADF.

* Add an artifact in which the source which come from build pipeline
* In the Test task, change the ARM template from the repository to the ARM template created by the build pipeline -> select the ARM templateFor Factory.json, and the same for the parameter as well.
* At test stage: Copy the value at the ARM template, then copy to replace for Pre Deployment and Post Deployment at Script Arguments. Then do the same for the production stage as well.
* Enable the trigger for the Release Pipeline (upon completion of the build pipeline) -> click on trigger for the artifact under the release pipeline.

Please see the pdf file namely Release Pipeline for how to turn on the Release Pipeline option

A screenshot of a computer

Description automatically generated

The source for the artifact is coming from the ARM template created by the Build Pipeline, and under Stage, there should be 2 stages: one for Test, and one for Production.

Add task to the Test

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Select the advance option from the drop down menu to assign the access (contributor role) to a specific group, not the entire subscription if we click on the Authorize. This mean the pipeline will have contributor role to the test-CI-CD-rg resource group which allow the pipeline to deploy the ARM template to this resource group.

A screenshot of a computer

Description automatically generated

Change to test-ci-cd-adf79 from dev

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated