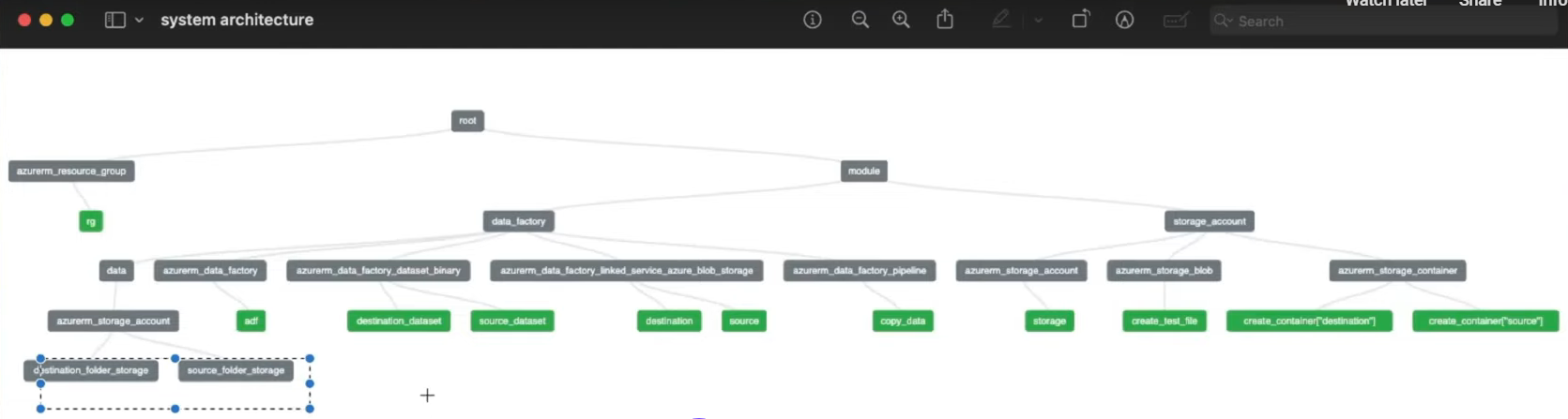
Data Engineering with CICD – Using Terraform on Azure

This project focuses on Continuous Integration (CI) and Continuous Deployment (CD) within the realm of modern data engineering.

Terraform is used to create automate infrastructure as Code to create - Azure Resource Group, Azure Data Factory with Copy Activity & Storage Account with container to store source and destination files.

# System Architecture:



# Prerequsites:

## Install Azure Cli

Azure CLI installation: <https://learn.microsoft.com/en-us/cli/azure/>

## Install terraform Cli

# Terraform installation: https://developer.hashicorp.com/terraform/install

## Install pycharm

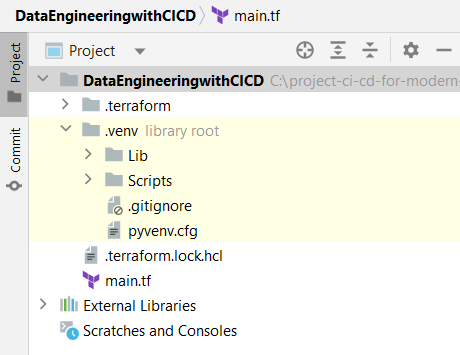
https://www.jetbrains.com/pycharm/

# PyCharm-Create new project under C:\project-ci-cd-for-modern-data-engineering

## Login to Azure account:

(.venv) PS C:\project-ci-cd-for-modern-data-engineering\DataEngineeringwithCICD> az login

Log into your Azure Account.

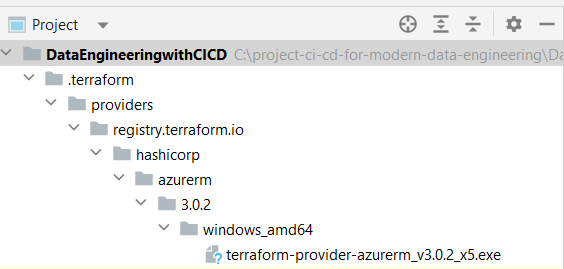


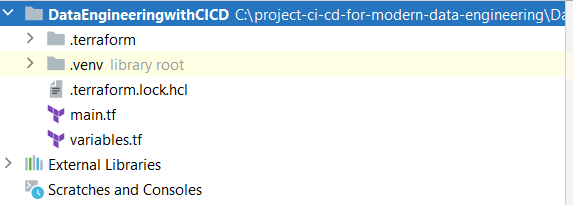
## Create main.tf:

|  |
| --- |
| terraform{  required\_providers {  azurerm = {  source = "hashicorp/azurerm"  version = "~> 3.0.2 "  }  }  required\_version = ">= 1.1.0" }  provider "azurerm" {  features {}  } |

## Run: terraform init

|  |
| --- |
| (.venv) PS C:\project-ci-cd-for-modern-data-engineering\DataEngineeringwithCICD> terraform init  Initializing the backend...  Initializing provider plugins...  - Finding hashicorp/azurerm versions matching "~> 3.0.2"...  - Installing hashicorp/azurerm v3.0.2...  - Installed hashicorp/azurerm v3.0.2 (signed by HashiCorp)  Terraform has created a lock file .terraform.lock.hcl to record the provider  selections it made above. Include this file in your version control repository  so that Terraform can guarantee to make the same selections by default when  you run "terraform init" in the future.  Terraform has been successfully initialized!  You may now begin working with Terraform. Try running "terraform plan" to see  any changes that are required for your infrastructure. All Terraform commands  should now work.  If you ever set or change modules or backend configuration for Terraform,  rerun this command to reinitialize your working directory. If you forget, other  commands will detect it and remind you to do so if necessary. |





## Create variable.tf

|  |
| --- |
| variables "resource\_group\_name" {  description = "Name of the resource group"  type = string }   variables "location" {  description = "Name of the resource location/region"  type = string }   variables "tags" {  description = "Tags associated with your resource"  type = string } |

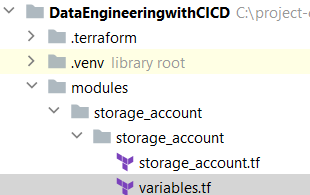
## Update main.tf

Add following code

|  |
| --- |
| resource "azurerm\_resource\_group" "rg" {  name = *var*.resource\_group\_name  location = *var*.location  tags = *var*.tags } |

# Create Modules

## Create Modules🡪Storage\_account🡪 Storage\_account -> storage\_account.tf

****

|  |
| --- |
| resource "azurerm\_storage\_account" "storage" {  name = *var*.storage\_account\_name  resource\_group\_name = *var*.resource\_group\_name  location = *var*.location  account\_tier = "Standard"  account\_replication\_type = "LRS"  tags = {  environment = "development"  } }  resource "azurerm\_storage\_container" "create\_container" {  for\_each = {  source = *var*.source\_folder\_name,  destination = *var*.destination\_folder\_name  }   name = each.key  storage\_account\_name = azurerm\_storage\_account.storage.name  container\_access\_type = *var*.container\_access\_type }  resource "azurerm\_storage\_blob" "create\_test\_file" {  name = "test.txt"  storage\_account\_name = azurerm\_storage\_account.storage.name  storage\_container\_name = azurerm\_storage\_container.create\_container["source"].name  type = "Block"  source\_content = "Hello Asim!" }  output "storage\_account\_key" {  value = azurerm\_storage\_account.storage.primary\_access\_key } |

## Create Modules🡪Storage\_account🡪 Storage\_account ->variables.tf

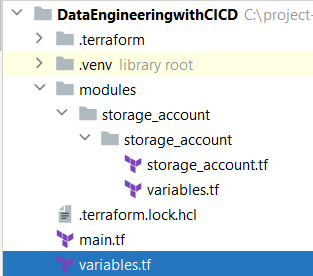
|  |
| --- |
| variable "storage\_account\_name" {  description = "The storage account name"  type = string }  variable "resource\_group\_name" {  description = "The name of the resource group"  type = string }  variable "location" {  description = "The location/region of the resource"  type = string }  variable "source\_folder\_name" {  description = "The source folder name"  type = string }  variable "destination\_folder\_name" {  description = "The destination folder name"  type = string }  variable "container\_access\_type" {  description = "The access type for the storage account container"  type = string  default = "private" } |

## Link the Module to root (main.tf)

## Update main.tf

|  |
| --- |
| module "storage\_account" {  source = "./modules/storage\_account/storage\_account"   resource\_group\_name = *var*.resource\_group\_name  storage\_account\_name = *var*.storage\_account\_name  location = *var*.location  source\_folder\_name = *var*.source\_folder\_name  destination\_folder\_name = *var*.destination\_folder\_name   depends\_on = [  azurerm\_resource\_group.rg  ] } |

## Update variables.tf

****

|  |
| --- |
| variable "storage\_account\_name" {  description = "The storage account name"  type = string }  variable "source\_folder\_name" {  description = "The source folder name"  type = string }  variable "destination\_folder\_name" {  description = "The destination folder name"  type = string } |

## Run terraform init

## Run terraform validate

## Fix error on var.tags

**You will see following error:**

|  |
| --- |
| **│ Error: Incorrect attribute value type**  **│**  **│ on main.tf line 18, in resource "azurerm\_resource\_group" "rg":**  **│ 18: tags = var.tags**  **│ ├────────────────**  **│ │ var.tags is a string, known only after apply**  **│**  **│ Inappropriate value for attribute "tags": map of string required.** |

**Update variable.tf: “tags” entry**

|  |
| --- |
| variable "tags" {  description = "The tags associated with your resource"  type = **map(string)** } |

## Create variables.tfvars file:

**This is required to provide all the variable values**

|  |
| --- |
| resource\_group\_name = "codewithasim-rg" storage\_account\_name = "codewithasimsa" location = "centralindia"  tags = {  Environment = "development" }  source\_folder\_name = "source" destination\_folder\_name = "destination" |

## Run terraform plan -var-file="variables.tfvars"

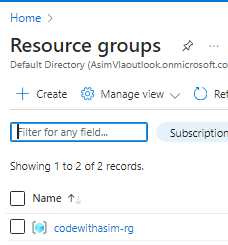
|  |
| --- |
| Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:  + create  Terraform will perform the following actions:  # azurerm\_resource\_group.rg will be created  + resource "azurerm\_resource\_group" "rg" {  + id = (known after apply)  + location = "centralindia"  + name = "codewithasim-rg"  + tags = {  + "Environment" = "development"  }  }  # module.storage\_account.azurerm\_storage\_account.storage will be created  + resource "azurerm\_storage\_account" "storage" {  + access\_tier = (known after apply)  + account\_kind = "StorageV2"  + account\_replication\_type = "LRS"  + account\_tier = "Standard"  + allow\_nested\_items\_to\_be\_public = true  + enable\_https\_traffic\_only = true  + id = (known after apply)  + infrastructure\_encryption\_enabled = false  + is\_hns\_enabled = false  + large\_file\_share\_enabled = (known after apply)  + location = "centralindia"  + min\_tls\_version = "TLS1\_2"  + name = "codewithasimsa"  + nfsv3\_enabled = false  + primary\_access\_key = (sensitive value)  + primary\_blob\_connection\_string = (sensitive value)  + primary\_blob\_endpoint = (known after apply)  + primary\_blob\_host = (known after apply)  + primary\_connection\_string = (sensitive value)  + primary\_dfs\_endpoint = (known after apply)  + primary\_dfs\_host = (known after apply)  + primary\_file\_endpoint = (known after apply)  + primary\_file\_host = (known after apply)  + primary\_location = (known after apply)  + primary\_queue\_endpoint = (known after apply)  + primary\_queue\_host = (known after apply)  + primary\_table\_endpoint = (known after apply)  + primary\_table\_host = (known after apply)  + primary\_web\_endpoint = (known after apply)  + primary\_web\_host = (known after apply)  + queue\_encryption\_key\_type = "Service"  + resource\_group\_name = "codewithasim-rg"  + secondary\_access\_key = (sensitive value)  + secondary\_blob\_connection\_string = (sensitive value)  + secondary\_blob\_endpoint = (known after apply)  + secondary\_blob\_host = (known after apply)  + secondary\_connection\_string = (sensitive value)  + secondary\_dfs\_endpoint = (known after apply)  + secondary\_dfs\_host = (known after apply)  + secondary\_file\_endpoint = (known after apply)  + secondary\_file\_host = (known after apply)  + secondary\_location = (known after apply)  + secondary\_queue\_endpoint = (known after apply)  + secondary\_queue\_host = (known after apply)  + secondary\_table\_endpoint = (known after apply)  + secondary\_table\_host = (known after apply)  + secondary\_web\_endpoint = (known after apply)  + secondary\_web\_host = (known after apply)  + shared\_access\_key\_enabled = true  + table\_encryption\_key\_type = "Service"  + tags = {  + "environment" = "development"  }  + blob\_properties {  + change\_feed\_enabled = (known after apply)  + default\_service\_version = (known after apply)  + last\_access\_time\_enabled = (known after apply)  + versioning\_enabled = (known after apply)  + container\_delete\_retention\_policy {  + days = (known after apply)  }  + cors\_rule {  + allowed\_headers = (known after apply)  + allowed\_methods = (known after apply)  + allowed\_origins = (known after apply)  + exposed\_headers = (known after apply)  + max\_age\_in\_seconds = (known after apply)  }  + delete\_retention\_policy {  + days = (known after apply)  }  }  + network\_rules {  + bypass = (known after apply)  + default\_action = (known after apply)  + ip\_rules = (known after apply)  + virtual\_network\_subnet\_ids = (known after apply)  + private\_link\_access {  + endpoint\_resource\_id = (known after apply)  + endpoint\_tenant\_id = (known after apply)  }  }  + queue\_properties {  + cors\_rule {  + allowed\_headers = (known after apply)  + allowed\_methods = (known after apply)  + allowed\_origins = (known after apply)  + exposed\_headers = (known after apply)  + max\_age\_in\_seconds = (known after apply)  }  + hour\_metrics {  + enabled = (known after apply)  + include\_apis = (known after apply)  + retention\_policy\_days = (known after apply)  + version = (known after apply)  }  + logging {  + delete = (known after apply)  + read = (known after apply)  + retention\_policy\_days = (known after apply)  + version = (known after apply)  + write = (known after apply)  }  + minute\_metrics {  + enabled = (known after apply)  + include\_apis = (known after apply)  + retention\_policy\_days = (known after apply)  + version = (known after apply)  }  }  + routing {  + choice = (known after apply)  + publish\_internet\_endpoints = (known after apply)  + publish\_microsoft\_endpoints = (known after apply)  }  + share\_properties {  + cors\_rule {  + allowed\_headers = (known after apply)  + allowed\_methods = (known after apply)  + allowed\_origins = (known after apply)  + exposed\_headers = (known after apply)  + max\_age\_in\_seconds = (known after apply)  }  + retention\_policy {  + days = (known after apply)  }  + smb {  + authentication\_types = (known after apply)  + channel\_encryption\_type = (known after apply)  + kerberos\_ticket\_encryption\_type = (known after apply)  + versions = (known after apply)  }  }  }  # module.storage\_account.azurerm\_storage\_blob.create\_test\_file will be created  + resource "azurerm\_storage\_blob" "create\_test\_file" {  + access\_tier = (known after apply)  + content\_type = "application/octet-stream"  + id = (known after apply)  + metadata = (known after apply)  + name = "test.txt"  + parallelism = 8  + size = 0  + source\_content = "Hello Asim!"  + storage\_account\_name = "codewithasimsa"  + storage\_container\_name = "source"  + type = "Block"  + url = (known after apply)  }  # module.storage\_account.azurerm\_storage\_container.create\_container["destination"] will be created  + resource "azurerm\_storage\_container" "create\_container" {  + container\_access\_type = "private"  + has\_immutability\_policy = (known after apply)  + has\_legal\_hold = (known after apply)  + id = (known after apply)  + metadata = (known after apply)  + name = "destination"  + resource\_manager\_id = (known after apply)  + storage\_account\_name = "codewithasimsa"  }  # module.storage\_account.azurerm\_storage\_container.create\_container["source"] will be created  + resource "azurerm\_storage\_container" "create\_container" {  + container\_access\_type = "private"  + has\_immutability\_policy = (known after apply)  + has\_legal\_hold = (known after apply)  + id = (known after apply)  + metadata = (known after apply)  + name = "source"  + resource\_manager\_id = (known after apply)  + storage\_account\_name = "codewithasimsa"  }  Plan: 5 to add, 0 to change, 0 to destroy. |

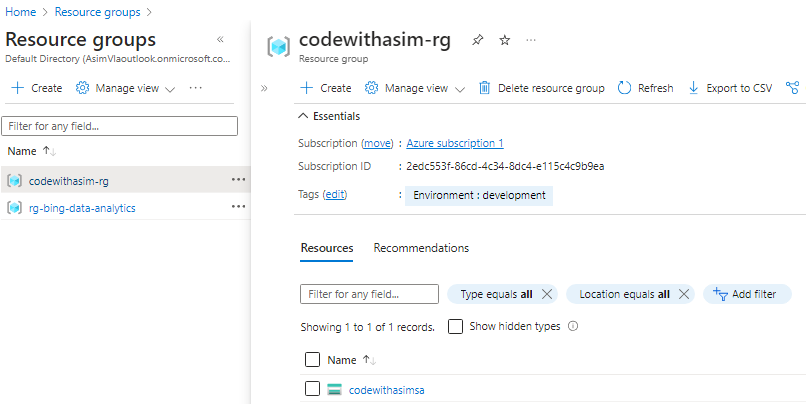
## Verify Terraform Resource creation

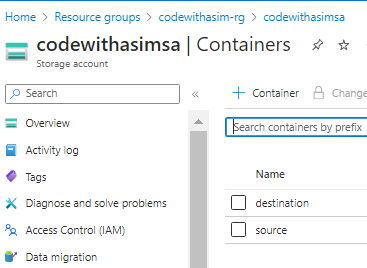
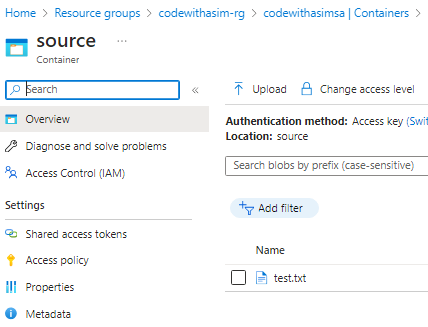
## terraform apply -var-file="variables.tfvars"

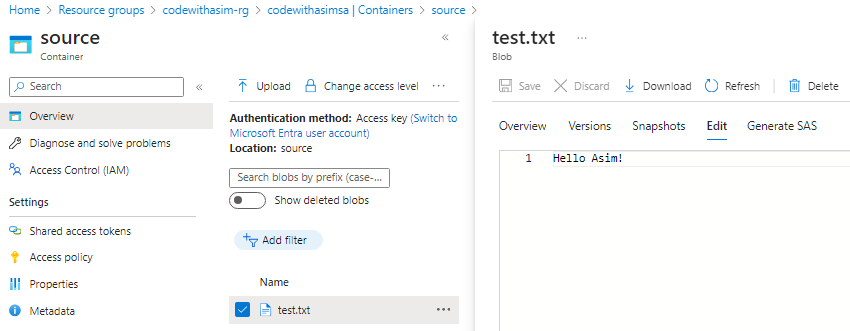
|  |
| --- |
| Plan: 5 to add, 0 to change, 0 to destroy.  Do you want to perform these actions?  Terraform will perform the actions described above.  Only 'yes' will be accepted to approve.  Enter a value: yes  azurerm\_resource\_group.rg: Creating...  azurerm\_resource\_group.rg: Creation complete after 1s [id=/subscriptions/<Subscription Id>/resourceGroups/codewithasim-rg]  module.storage\_account.azurerm\_storage\_account.storage: Creating...  module.storage\_account.azurerm\_storage\_account.storage: Still creating... [10s elapsed]  module.storage\_account.azurerm\_storage\_account.storage: Still creating... [20s elapsed]  module.storage\_account.azurerm\_storage\_account.storage: Creation complete after 27s [id=/subscriptions/<subscriptionId>/resourceGroups/codewithasim-rg/providers/Microsoft.Storage/storageAccounts/codewithasimsa]  module.storage\_account.azurerm\_storage\_container.create\_container["source"]: Creating...  module.storage\_account.azurerm\_storage\_container.create\_container["destination"]: Creating...  module.storage\_account.azurerm\_storage\_container.create\_container["source"]: Creation complete after 0s [id=https://codewithasimsa.blob.core.windows.net/source]  module.storage\_account.azurerm\_storage\_container.create\_container["destination"]: Creation complete after 0s [id=https://codewithasimsa.blob.core.windows.net/destination]  module.storage\_account.azurerm\_storage\_blob.create\_test\_file: Creating...  module.storage\_account.azurerm\_storage\_blob.create\_test\_file: Creation complete after 0s [id=https://codewithasimsa.blob.core.windows.net/source/test.txt]  Apply complete! Resources: 5 added, 0 changed, 0 destroyed. |

## Verify resources created on Azure Portal:







## Tear down resources created on Azure Portal:

(.venv) PS C:\project-ci-cd-for-modern-data-engineering\DataEngineeringwithCICD> terraform destroy -var-file="variables.tfvars"

|  |
| --- |
| Plan: 0 to add, 0 to change, 5 to destroy.  Do you really want to destroy all resources?  Terraform will destroy all your managed infrastructure, as shown above.  There is no undo. Only 'yes' will be accepted to confirm.  Enter a value: yes  module.storage\_account.azurerm\_storage\_blob.create\_test\_file: Destroying... [id=https://codewithasimsa.blob.core.windows.net/source/test.txt]  module.storage\_account.azurerm\_storage\_blob.create\_test\_file: Destruction complete after 1s  module.storage\_account.azurerm\_storage\_container.create\_container["source"]: Destroying... [id=https://codewithasimsa.blob.core.windows.net/source]  module.storage\_account.azurerm\_storage\_container.create\_container["destination"]: Destroying... [id=https://codewithasimsa.blob.core.windows.net/destination]  module.storage\_account.azurerm\_storage\_container.create\_container["source"]: Destruction complete after 0s  module.storage\_account.azurerm\_storage\_container.create\_container["destination"]: Destruction complete after 0s  module.storage\_account.azurerm\_storage\_account.storage: Destroying... [id=/subscriptions/<Subscription Id>/resourceGroups/codewithasim-rg/providers/Microsoft.Storage/storageAccounts/codewithasimsa]  module.storage\_account.azurerm\_storage\_account.storage: Destruction complete after 4s  azurerm\_resource\_group.rg: Destroying... [id=/subscriptions/<subscription Id>/resourceGroups/codewithasim-rg]  azurerm\_resource\_group.rg: Still destroying... [id=/subscriptions//<subscription Id>/resourceGroups/codewithasim-rg, 10s elapsed]  azurerm\_resource\_group.rg: Destruction complete after 16s  Destroy complete! Resources: 5 destroyed. |

## Create Modules🡪data\_factory🡪data\_factory🡪 data\_factory.tf

|  |
| --- |
| resource "azurerm\_data\_factory" "adf" {  name = *var*.adf\_name  resource\_group\_name = *var*.resource\_group\_name  location = *var*.location   identity {  type = "SystemAssigned"  } }  data "azurerm\_storage\_account" "source\_folder\_storage" {  name = *var*.storage\_account\_name  resource\_group\_name = *var*.resource\_group\_name }  data "azurerm\_storage\_account" "destination\_folder\_storage" {  name = *var*.storage\_account\_name  resource\_group\_name = *var*.resource\_group\_name }  resource "azurerm\_data\_factory\_linked\_service\_azure\_blob\_storage" "source" {  name = "source-storage"  data\_factory\_id = azurerm\_data\_factory.adf.id  connection\_string = *data*.azurerm\_storage\_account.source\_folder\_storage.primary\_connection\_string }  resource "azurerm\_data\_factory\_linked\_service\_azure\_blob\_storage" "destination" {  name = "destination-storage"  data\_factory\_id = azurerm\_data\_factory.adf.id  connection\_string = *data*.azurerm\_storage\_account.destination\_folder\_storage.primary\_connection\_string }  *#source and sink dataset to blob storage* resource "azurerm\_data\_factory\_dataset\_binary" "source\_dataset" {   name = "source\_dataset"  data\_factory\_id = azurerm\_data\_factory.adf.id  linked\_service\_name = azurerm\_data\_factory\_linked\_service\_azure\_blob\_storage.source.name   sftp\_server\_location {  filename = "test.txt"  path = "source"  } }  resource "azurerm\_data\_factory\_dataset\_binary" "destination\_dataset" {   name = "destination\_dataset"  data\_factory\_id = azurerm\_data\_factory.adf.id  linked\_service\_name = azurerm\_data\_factory\_linked\_service\_azure\_blob\_storage.destination.name   sftp\_server\_location {  filename = "test-${formatdate("YYYY-MM-DD-hh-mm-ss", timestamp() )}.txt"  path = "destination"  } }  resource "azurerm\_data\_factory\_pipeline" "copy\_data" {   name = "copy\_data\_pipeline"  data\_factory\_id = azurerm\_data\_factory.adf.id   activities\_json = <<JSON [  {  "name": "CopyFromSourceToDestination",  "type": "Copy",  "typeProperties": {  "source": {  "type": "BinarySource",  "recursive": true  },  "sink": {  "type": "BinarySink"  },  "enableStaging": false  },  "policy": {  "timeout": "7.00:00:00",  "retry": 0,  "retryIntervalInSeconds": 30,  "secureInput": false,  "secureOutput": false  },  "scheduler": {  "frequency": "Day",  "interval": 1  },  "external": true,  "inputs": [  {  "referenceName": "source\_dataset",  "type": "DatasetReference"  }  ],  "outputs": [  {  "referenceName": "destination\_dataset",  "type": "DatasetReference"  }  ]  } ] JSON   depends\_on = [  azurerm\_data\_factory\_dataset\_binary.source\_dataset,  azurerm\_data\_factory\_dataset\_binary.destination\_dataset,  ] } |

## Create Modules🡪data\_factory🡪data\_factory🡪 variables.tf

|  |
| --- |
| variable "storage\_account\_name" {  description = "The storage account name"  type = string }  variable "resource\_group\_name" {  description = "The name of the resource group"  type = string }  variable "location" {  description = "The location/region of the resource"  type = string }  variable "adf\_name" {  description = "The data factory name"  type = string } |

## Update main.tf to add data\_factory module & dependency:

|  |
| --- |
| terraform{  required\_providers {  azurerm = {  source = "hashicorp/azurerm"  version = "~> 3.0.2 "  }  }  required\_version = ">= 1.1.0" }  provider "azurerm" {  features {}  }  resource "azurerm\_resource\_group" "rg" {  name = *var*.resource\_group\_name  location = *var*.location  tags = *var*.tags }  module "storage\_account" {  source = "./modules/storage\_account/storage\_account"   resource\_group\_name = *var*.resource\_group\_name  storage\_account\_name = *var*.storage\_account\_name  location = *var*.location  source\_folder\_name = *var*.source\_folder\_name  destination\_folder\_name = *var*.destination\_folder\_name   depends\_on = [  azurerm\_resource\_group.rg  ] }  module "data\_factory" {  source = "./modules/data\_factory/data\_factory"  adf\_name = *var*.adf\_name  location = *var*.location  resource\_group\_name = *var*.resource\_group\_name  storage\_account\_name = *var*.storage\_account\_name   depends\_on = [  *module*.storage\_account  ] } |

## Update variables.tf to add data\_factory variables

|  |
| --- |
| variable "resource\_group\_name" {  description = "The name of the resource group"  type = string }  variable "location" {  description = "The location/region of the resource"  type = string }  variable "tags" {  description = "The tags associated with your resource"  type = map(string) }  variable "storage\_account\_name" {  description = "The storage account name"  type = string }  variable "source\_folder\_name" {  description = "The source folder name"  type = string }  variable "destination\_folder\_name" {  description = "The destination folder name"  type = string }  variable "adf\_name" {  description = "The data factory name"  type = string } |

## Update variables.tfvars to add data\_factory variables

|  |
| --- |
| resource\_group\_name = "codewithasim-rg" storage\_account\_name = "codewithasimsa" location = "centralindia"  tags = {  Environment = "development" } source\_folder\_name = "source" destination\_folder\_name = "destination"  adf\_name = "codewithasim-asim" |

## Error: on terraform plan -var-file="variables.tfvars"

|  |
| --- |
| (.venv) PS C:\project-ci-cd-for-modern-data-engineering\DataEngineeringwithCICD> terraform plan -var-file="variables.tfvars"  module.data\_factory.data.azurerm\_storage\_account.destination\_folder\_storage: Reading...  module.data\_factory.data.azurerm\_storage\_account.source\_folder\_storage: Reading...  ╷  │ Error: Storage Account: (Name "codewithasimsa" / Resource Group "codewithasim-rg") was not found  │  │ with module.data\_factory.data.azurerm\_storage\_account.source\_folder\_storage,  │ on modules\data\_factory\data\_factory\data\_factory.tf line 11, in data "azurerm\_storage\_account" "source\_folder\_storage":  │ 11: data "azurerm\_storage\_account" "source\_folder\_storage" {  │  ╵  ╷  │ Error: Storage Account: (Name "codewithasimsa" / Resource Group "codewithasim-rg") was not found  │  │ with module.data\_factory.data.azurerm\_storage\_account.destination\_folder\_storage,  │ on modules\data\_factory\data\_factory\data\_factory.tf line 16, in data "azurerm\_storage\_account" "destination\_folder\_storage":  │ 16: data "azurerm\_storage\_account" "destination\_folder\_storage" {  │ |

## Resolve Error; Add following code in module "data\_factory"

|  |
| --- |
| module "data\_factory" {  source = "./modules/data\_factory/data\_factory"  adf\_name = *var*.adf\_name  location = *var*.location  resource\_group\_name = *var*.resource\_group\_name  storage\_account\_name = *var*.storage\_account\_name   depends\_on = [  *module*.storage\_account  ] } |

# Final Run:

## Run terraform init

Run terraform validate

## Run terraform plan -var-file="variables.tfvars"

|  |
| --- |
| (.venv) PS C:\project-ci-cd-for-modern-data-engineering\DataEngineeringwithCICD> terraform plan -var-file="variables.tfvars"  Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:  + create  <= read (data resources)  Terraform will perform the following actions:  .  .  .  Plan: 11 to add, 0 to change, 0 to destroy. |

## terraform plan -var-file="variables.tfvars" -out=plan.out

Saved the plan to: plan.out

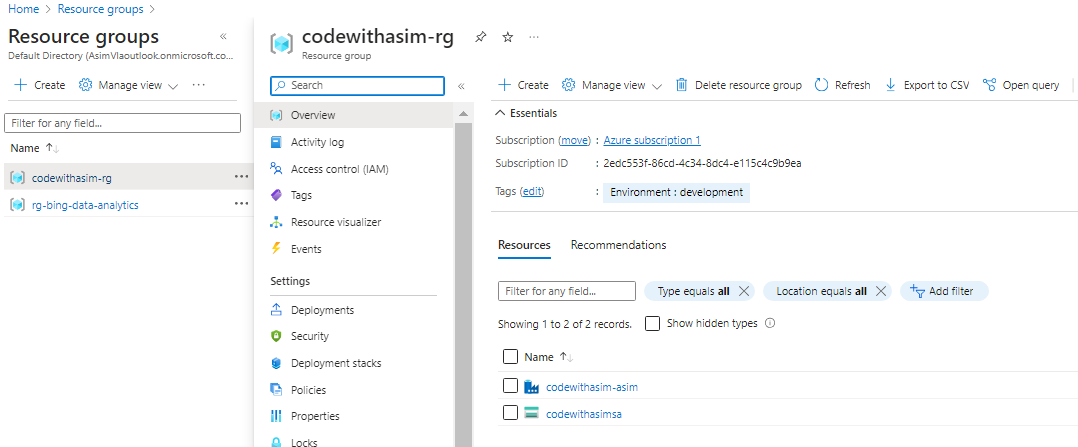
To perform exactly these actions, run the following command to apply:

## terraform apply "plan.out"

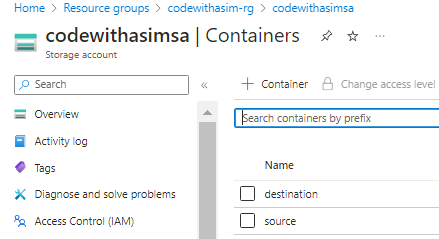
|  |
| --- |
| (.venv) PS C:\project-ci-cd-for-modern-data-engineering\DataEngineeringwithCICD> terraform apply plan.out  azurerm\_resource\_group.rg: Creating...  azurerm\_resource\_group.rg: Creation complete after 1s [id=/subscriptions/<Subscription Id>/resourceGroups/codewithasim-rg]  module.storage\_account.azurerm\_storage\_account.storage: Creating...  module.storage\_account.azurerm\_storage\_account.storage: Still creating... [10s elapsed]  module.storage\_account.azurerm\_storage\_account.storage: Still creating... [20s elapsed]  module.storage\_account.azurerm\_storage\_account.storage: Creation complete after 26s [id=/subscriptions/<Subscription Id>/resourceGroups/codewithasim-rg/providers/Microsoft.Storage/storageAccounts/codewithasimsa]  module.storage\_account.azurerm\_storage\_container.create\_container["source"]: Creating...  module.storage\_account.azurerm\_storage\_container.create\_container["destination"]: Creating...  module.storage\_account.azurerm\_storage\_container.create\_container["destination"]: Creation complete after 0s [id=https://codewithasimsa.blob.core.windows.net/destination]  module.storage\_account.azurerm\_storage\_container.create\_container["source"]: Creation complete after 0s [id=https://codewithasimsa.blob.core.windows.net/source]  module.storage\_account.azurerm\_storage\_blob.create\_test\_file: Creating...  module.storage\_account.azurerm\_storage\_blob.create\_test\_file: Creation complete after 0s [id=https://codewithasimsa.blob.core.windows.net/source/test.txt]  module.data\_factory.data.azurerm\_storage\_account.destination\_folder\_storage: Reading...  module.data\_factory.data.azurerm\_storage\_account.source\_folder\_storage: Reading...  module.data\_factory.azurerm\_data\_factory.adf: Creating...  module.data\_factory.data.azurerm\_storage\_account.destination\_folder\_storage: Read complete after 1s [id=/subscriptions//<Subscription Id>/resourceGroups/codewithasim-rg/providers/Microsoft.Storage/storageAccounts/codewithasimsa]  module.data\_factory.data.azurerm\_storage\_account.source\_folder\_storage: Read complete after 1s [id=/subscriptions/<Subscription Id>/resourceGroups/codewithasim-rg/providers/Microsoft.Storage/storageAccounts/codewithasimsa]  module.data\_factory.azurerm\_data\_factory.adf: Creation complete after 9s [id=/subscriptions/<Subscription Id>/resourceGroups/codewithasim-rg/providers/Microsoft.DataFactory/factories/codewithasim-asim]  module.data\_factory.azurerm\_data\_factory\_linked\_service\_azure\_blob\_storage.destination: Creating...  module.data\_factory.azurerm\_data\_factory\_linked\_service\_azure\_blob\_storage.source: Creating...  module.data\_factory.azurerm\_data\_factory\_linked\_service\_azure\_blob\_storage.source: Creation complete after 1s [id=/subscriptions/<Subscription Id>/resourceGroups/codewithasim-rg/providers/Microsoft.DataFactory/factories/codewithasim-asim/linkedservices/source-storage]  module.data\_factory.azurerm\_data\_factory\_linked\_service\_azure\_blob\_storage.destination: Creation complete after 1s [id=/subscriptions/<Subscription Id>/resourceGroups/codewithasim-rg/providers/Microsoft.DataFactory/factories/codewithasim-asim/linkedservices/destination-storage]  module.data\_factory.azurerm\_data\_factory\_dataset\_binary.source\_dataset: Creating...  module.data\_factory.azurerm\_data\_factory\_dataset\_binary.destination\_dataset: Creating...  module.data\_factory.azurerm\_data\_factory\_dataset\_binary.destination\_dataset: Creation complete after 0s [id=/subscriptions/<Subscription Id>/resourceGroups/codewithasim-rg/providers/Microsoft.DataFactory/factories/codewithasim-asim/datasets/destination\_dataset]  module.data\_factory.azurerm\_data\_factory\_dataset\_binary.source\_dataset: Creation complete after 0s [id=/subscriptions/<Subscription Id>/resourceGroups/codewithasim-rg/providers/Microsoft.DataFactory/factories/codewithasim-asim/datasets/source\_dataset]  module.data\_factory.azurerm\_data\_factory\_pipeline.copy\_data: Creating...  module.data\_factory.azurerm\_data\_factory\_pipeline.copy\_data: Creation complete after 1s [id=/subscriptions/<Subscription Id>/resourceGroups/codewithasim-rg/providers/Microsoft.DataFactory/factories/codewithasim-asim/pipelines/copy\_data\_pipeline]  Apply complete! Resources: 11 added, 0 changed, 0 destroyed. |

# Verify Resources on Azure Portal:

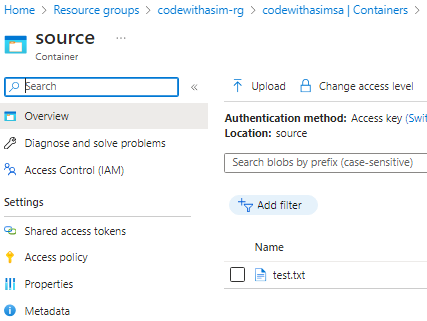
## Resource Group

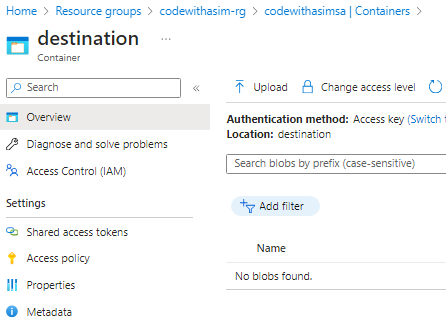


## Storage Account

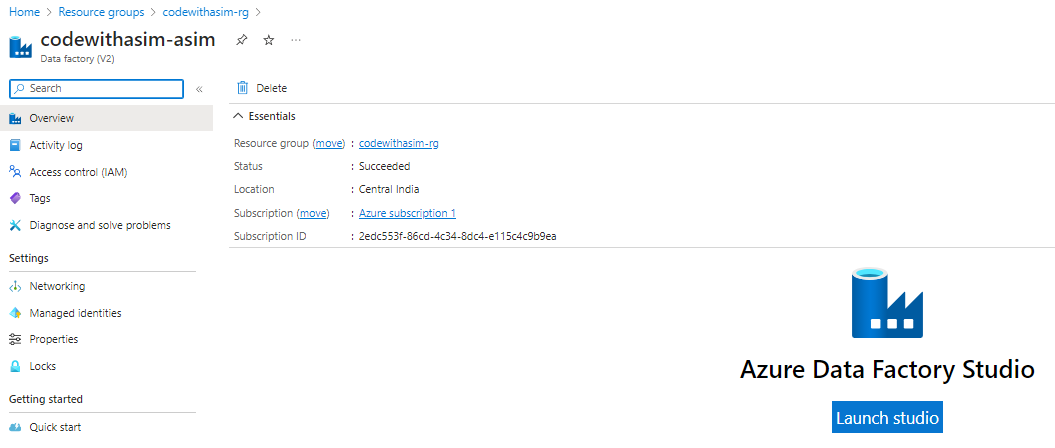


### Containers:

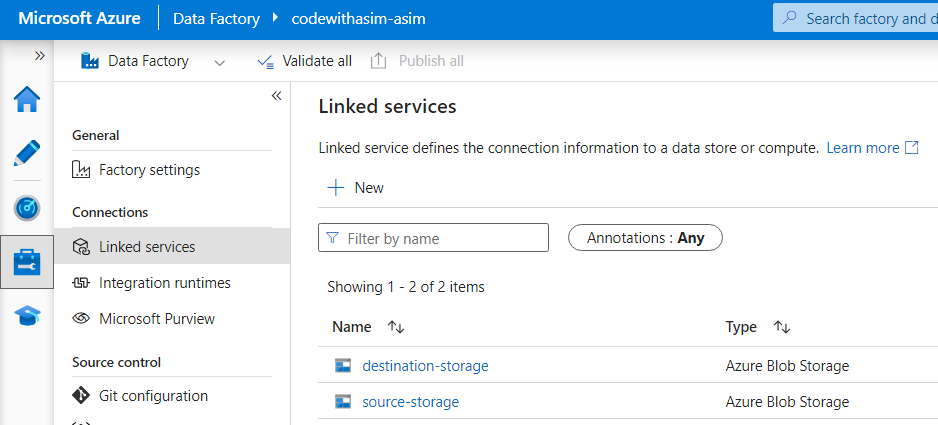


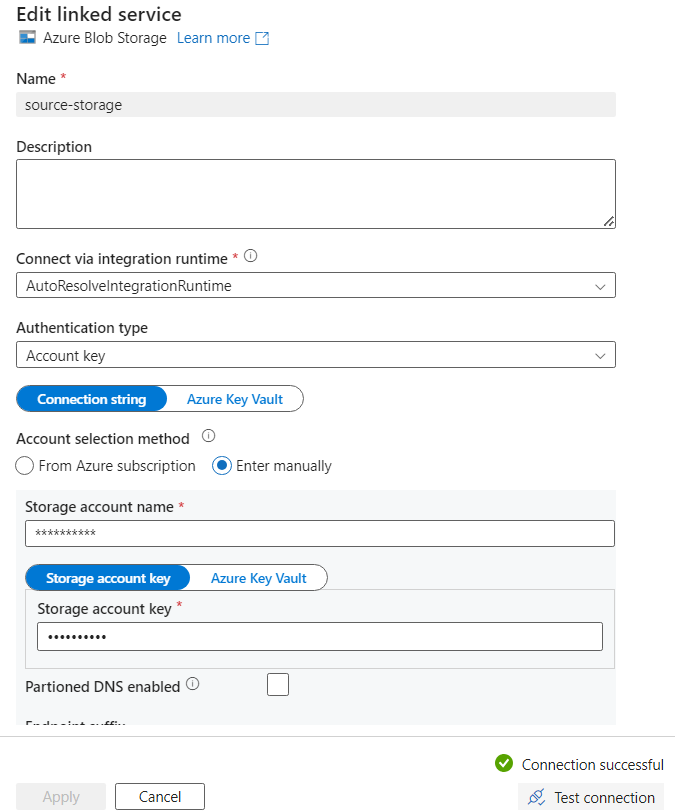


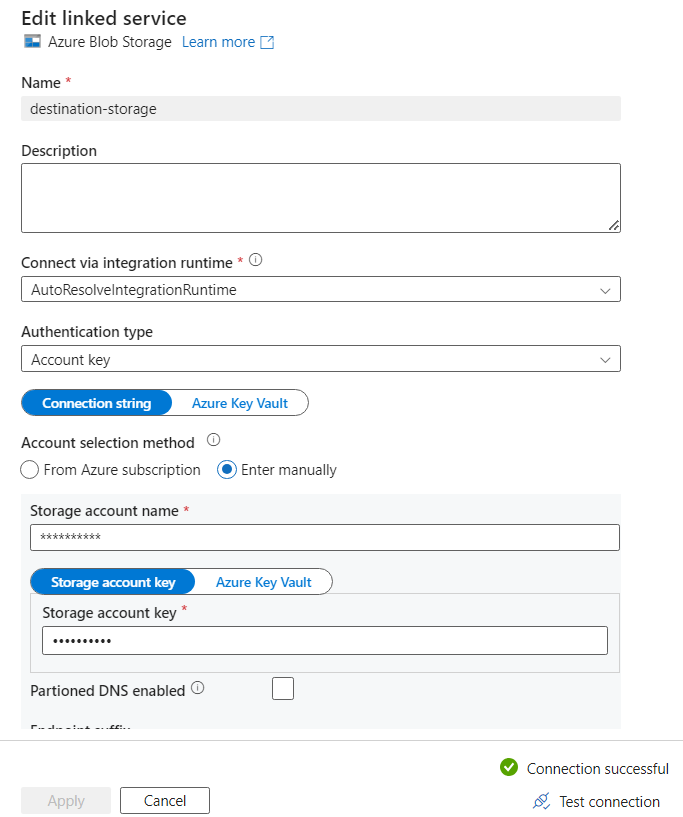
## Azure Data Factory



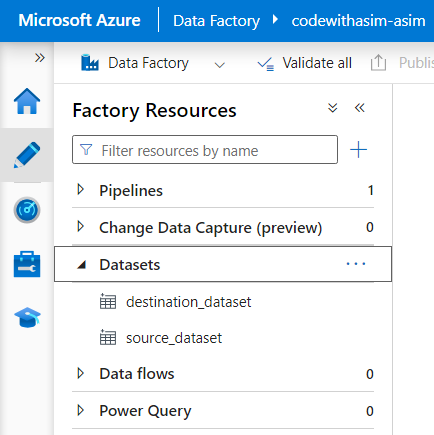
### Linked Service

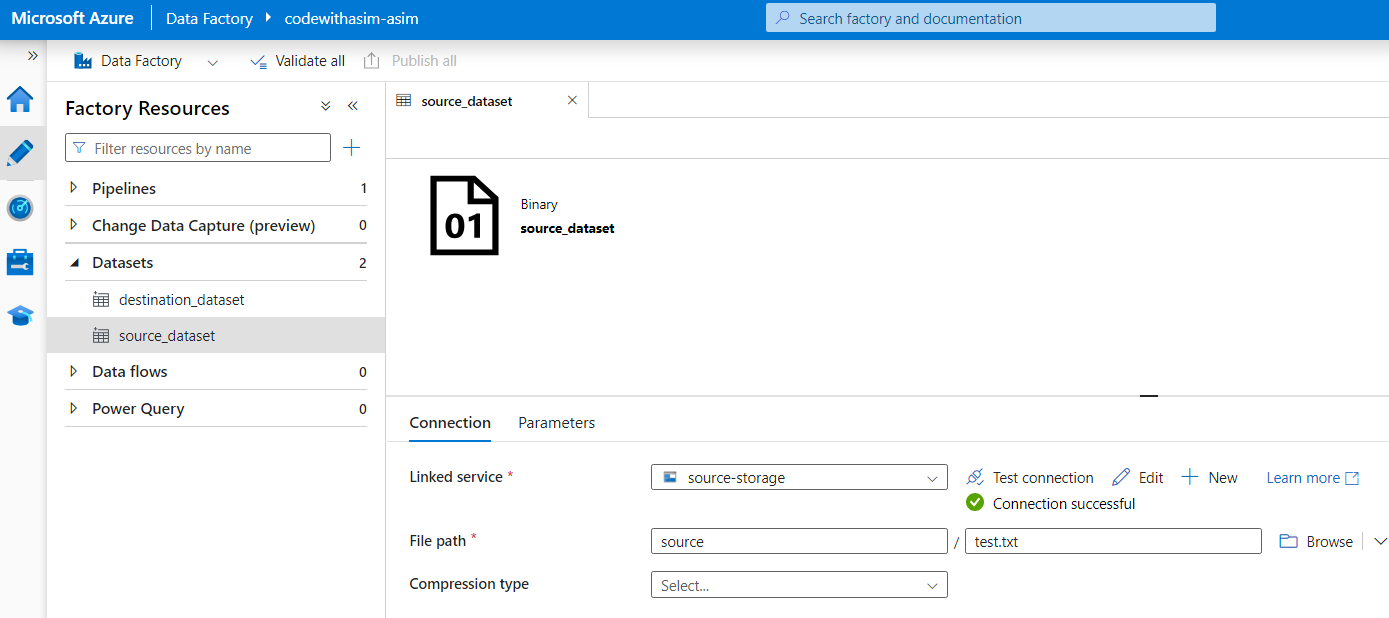


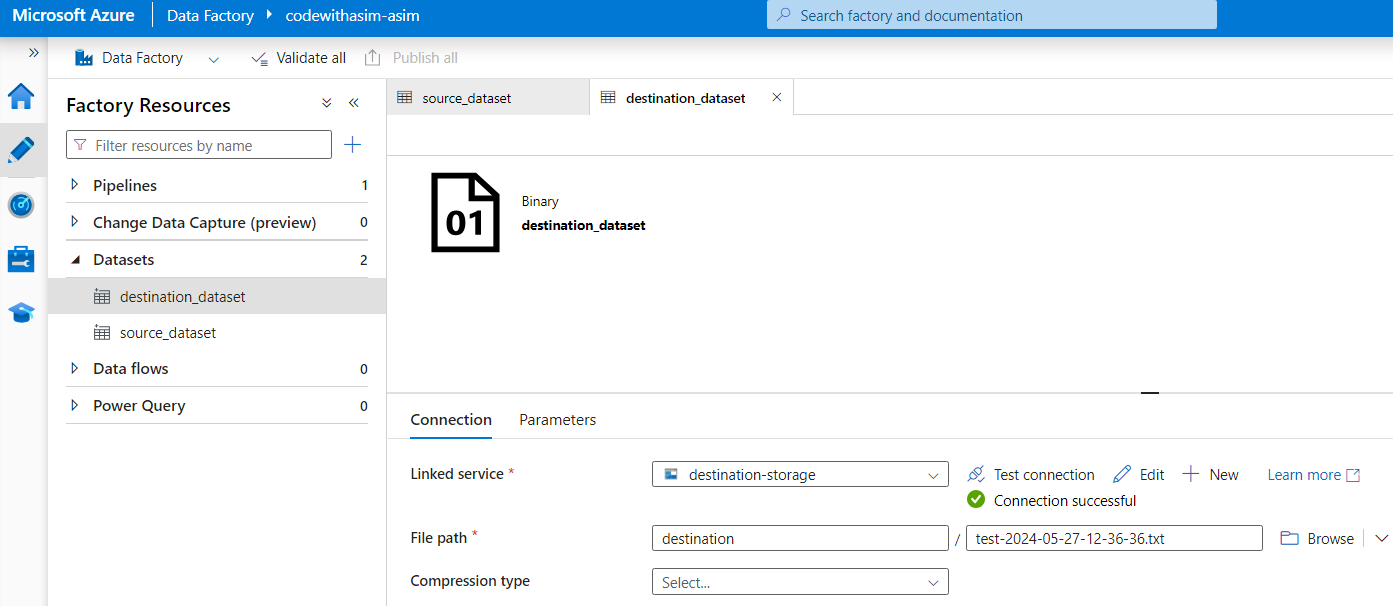




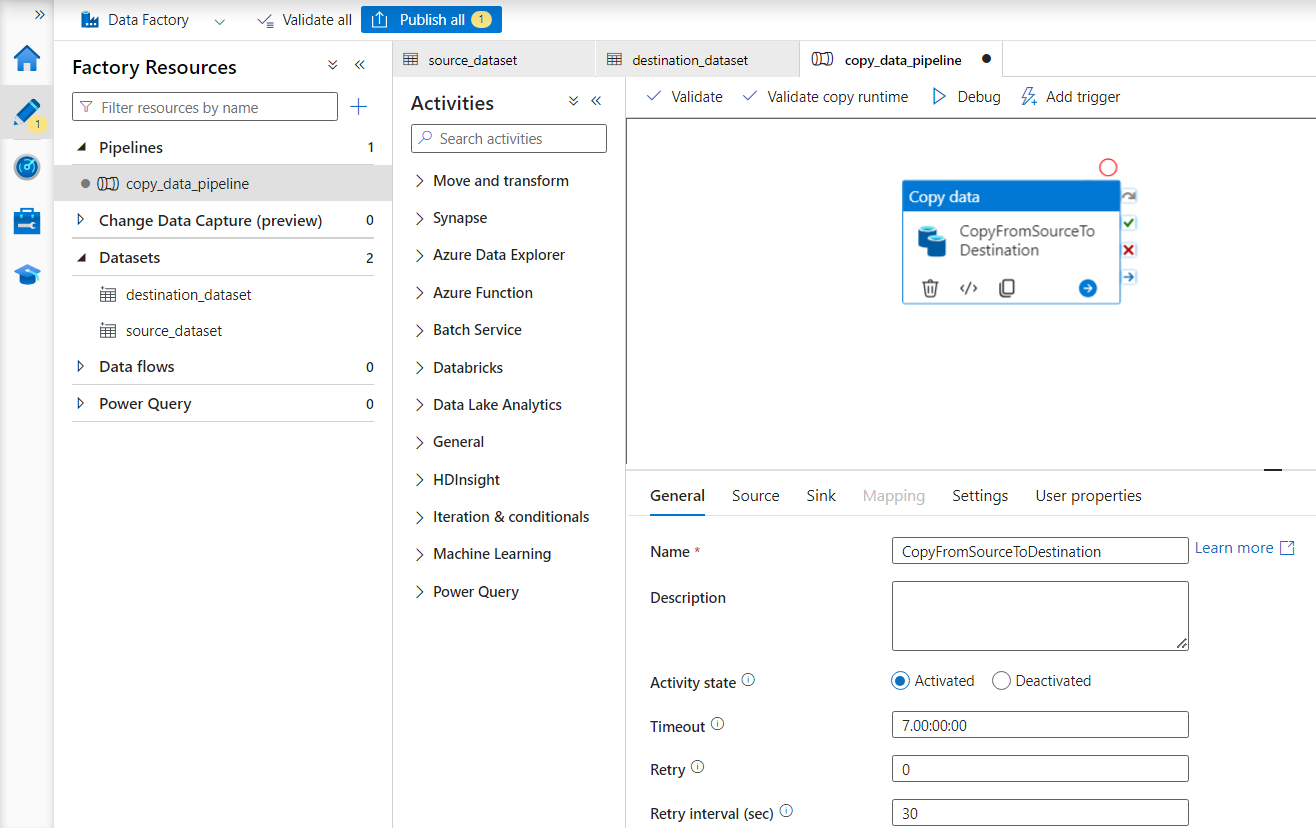
### Datasets



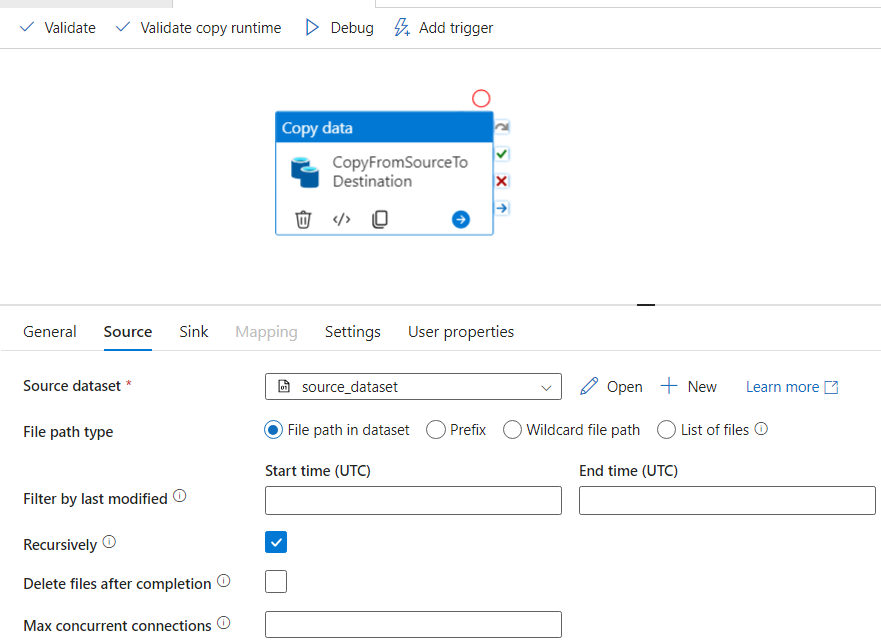


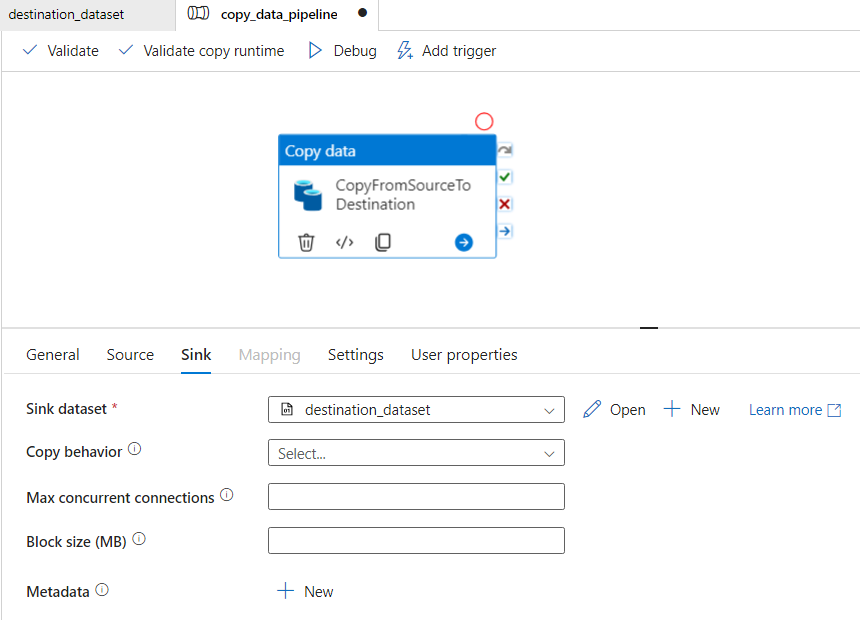


### Data Pipeline

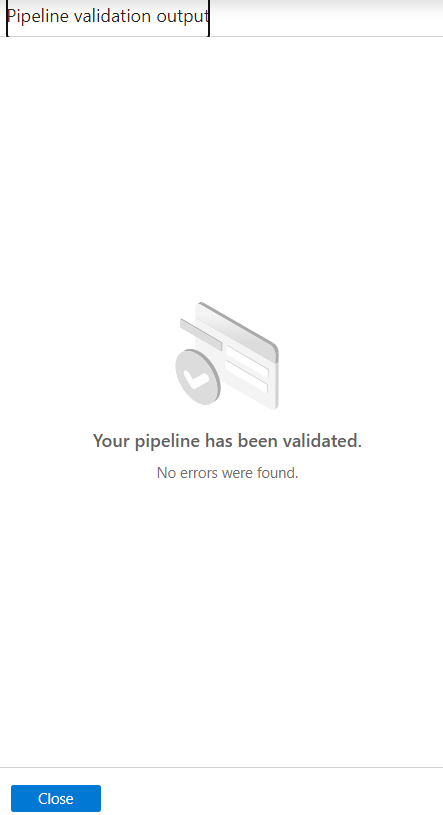


### Copy Activity

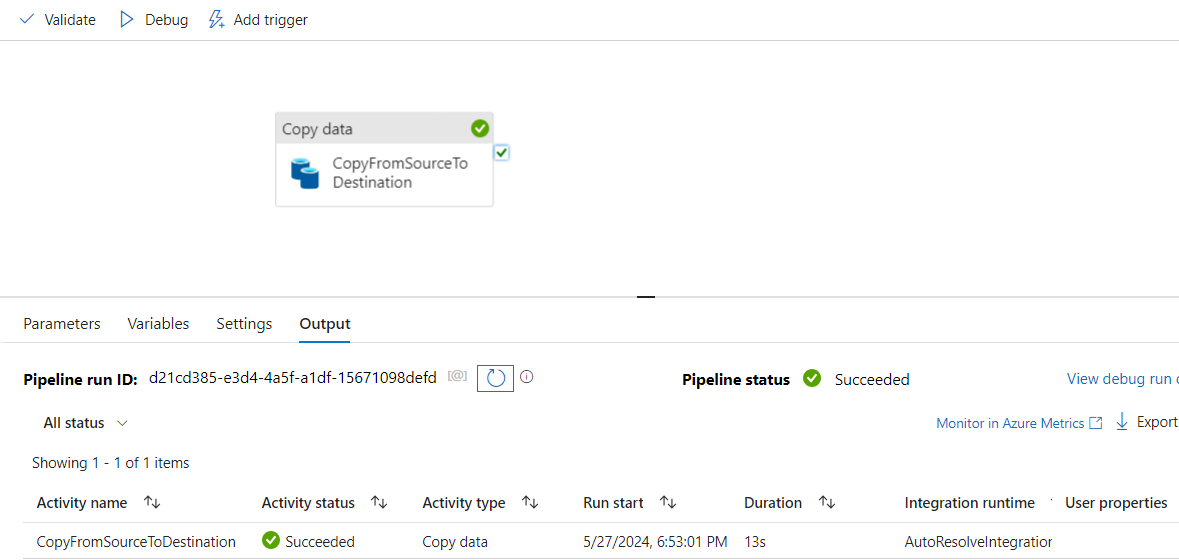




### Validate Pipeline

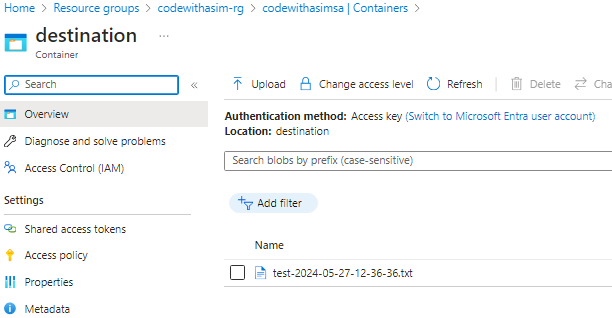


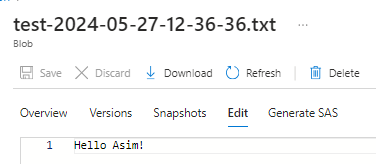
### Debug



### Result:

Source file is successfully copied to destination folder





# Destroy Entire Infrastructure

Run (.venv) PS C:\project-ci-cd-for-modern-data-engineering\DataEngineeringwithCICD> terraform destroy plan.out

╷

│ Error: Destroy can't be called with a plan file

│

│ If this plan was created using plan -destroy, apply it using:

│ terraform apply "plan.out"

## Run: terraform destroy -var-file="variables.tfvars"

(.venv) PS C:\project-ci-cd-for-modern-data-engineering\DataEngineeringwithCICD> terraform destroy -var-file="variables.tfvars"

|  |
| --- |
| .  .  .  Do you really want to destroy all resources?  Terraform will destroy all your managed infrastructure, as shown above.  There is no undo. Only 'yes' will be accepted to confirm.  Enter a value: yes  Destroy complete! Resources: 11 destroyed. |