Building a Regret-free Foundation for your Data Factory

Meagan Longoria

Denny Cherry & Associates Consulting

DCAC

Intro

About Me

Meagan Longoria

Denver, CO

Work at Denny Cherry & Associates Consulting

Microsoft Data Platform MVP

Blogger, Speaker, Author, Technical Editor

Building a new Azure Data Factory and not sure what you don’t know?

Top Regrets

Agenda

Poor resource organization in Azure

Lack of naming conventions

No/inconsistent key vault usage

Inappropriate use of version control

Tedious, manual deployments

Misunderstanding integration runtimes

Underutilizing parameterization

No established pipeline design patterns

Lack of comments and documentation

Resource Organization

Resource Organization

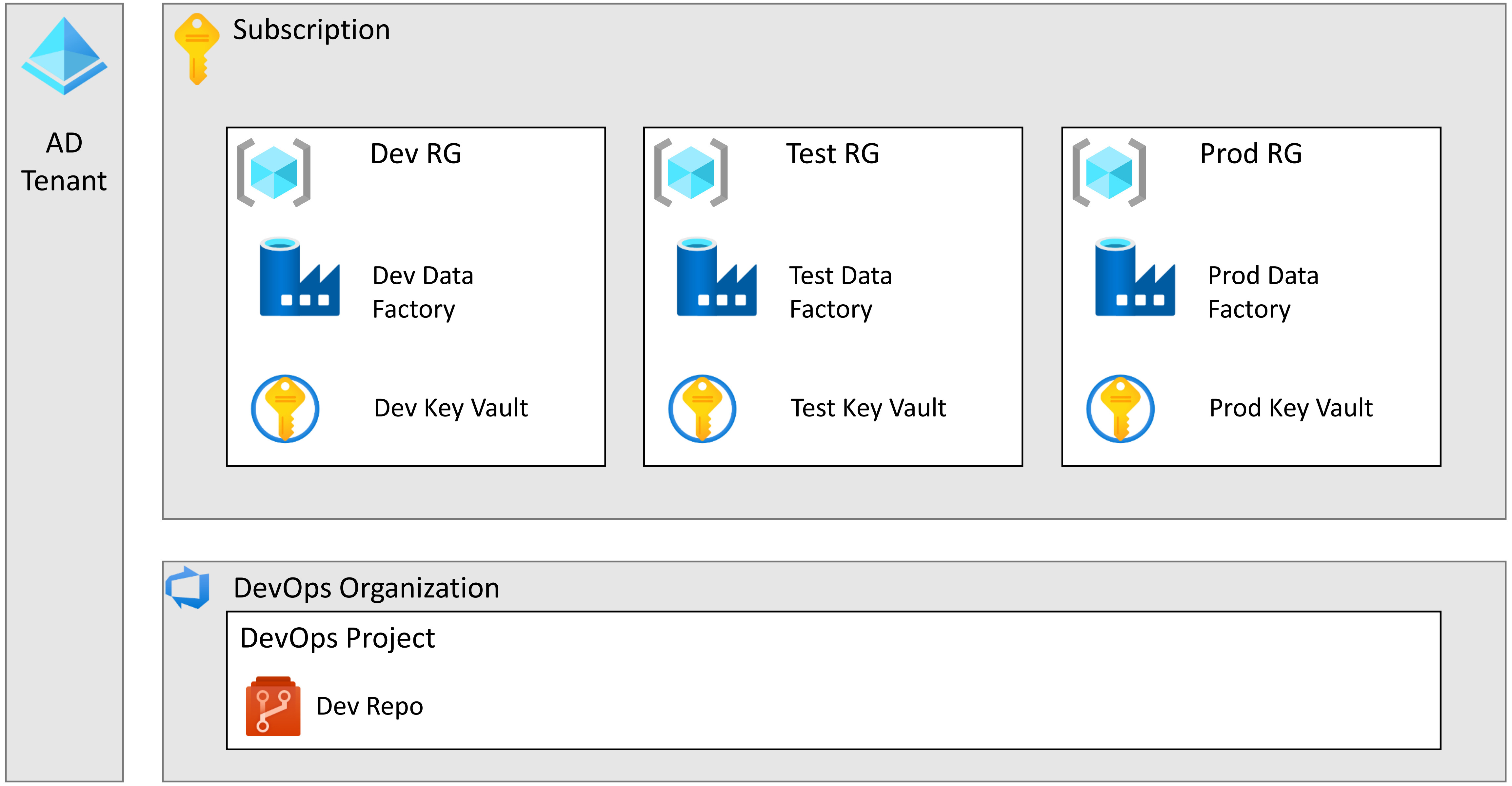
Separating environments

You need separate data factories and key vaults for each environment

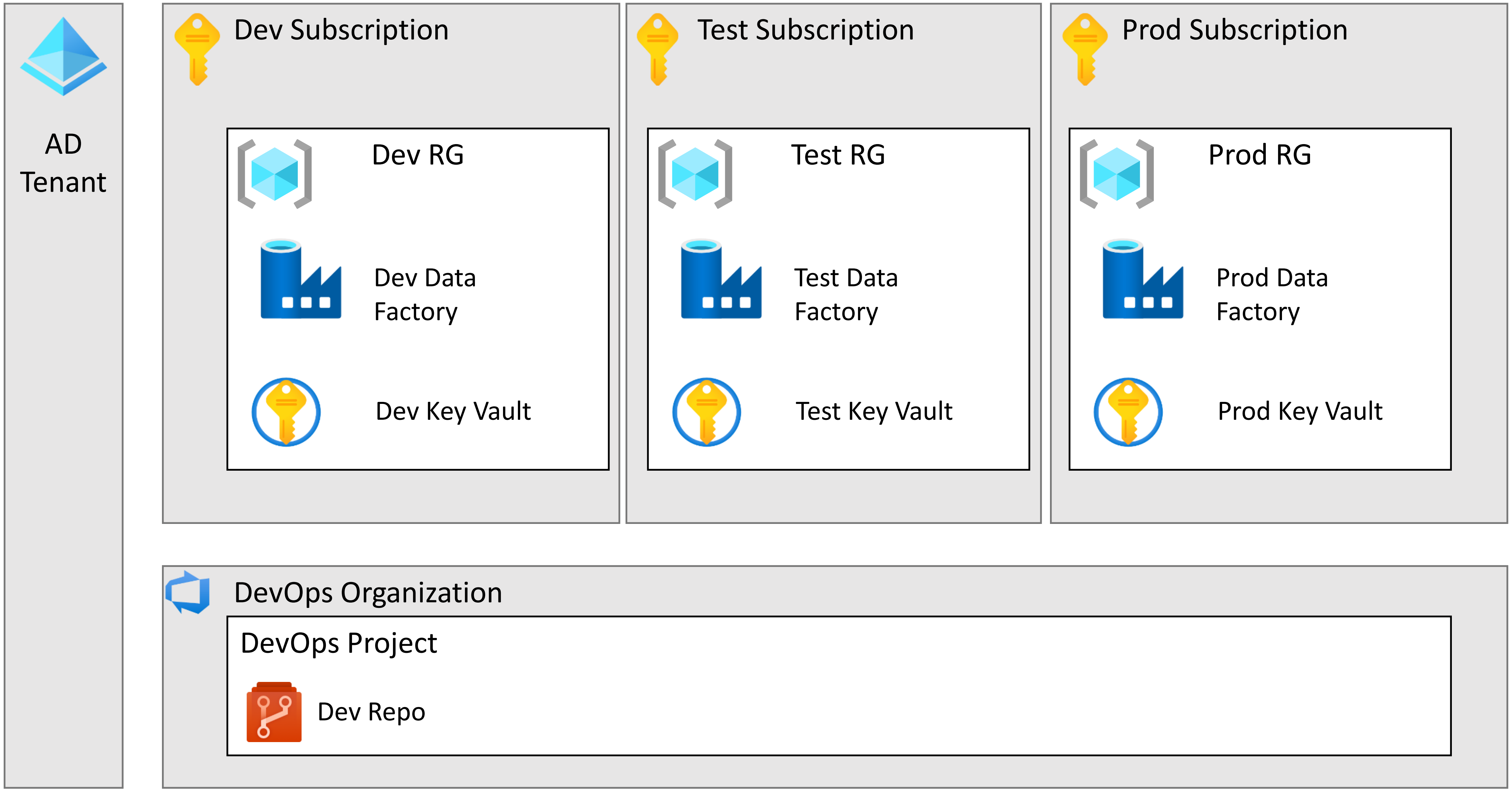
Common containers for separation:

- Resource Groups
- Subscriptions
- Tenants

Option 1: Separate Resource Groups



Option 2: Separate Subscriptions



Naming Conventions

Naming Conventions

Two levels of naming conventions

Azure resources

Data Factory artifacts

Naming Azure resources

Naming scopes and requirements

Naming components

Example naming convention:

<resource type><workload/application><environment>

<resource type><workload/application><environment><Azure region><instance>

Names vs Tags

Yes, you should use tags!

You can use tags to distinguish types and environments, but will others?

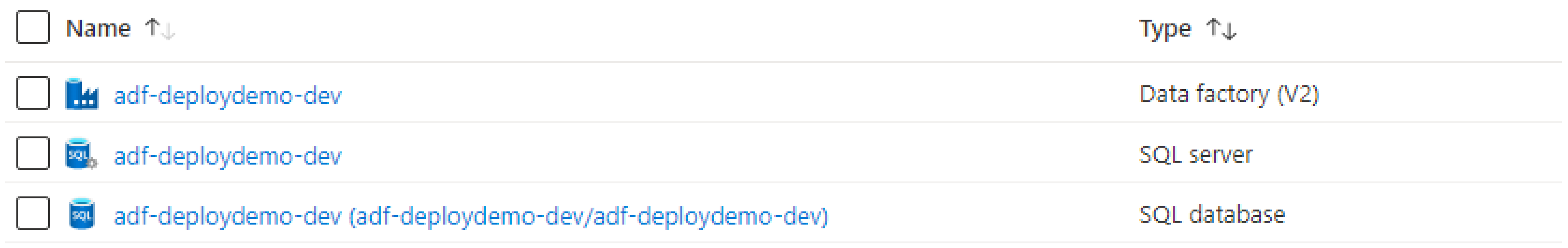
I name defensively because I don’t know who all will interact in the Azure Portal or via code

The most important thing is to be consistent!

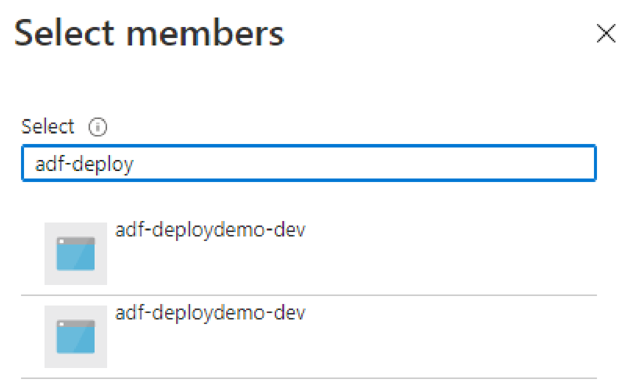
Make resource names unique

Managed identities assume the name of the resource

Non-unique resource names cause confusion with access management and PowerShell/CLI



Name ↑↓	Type ↑↓
adf-deploydemo-dev	Data factory (V2)
adf-deploydemo-dev	SQL server
adf-deploydemo-dev (adf-deploydemo-dev/adf-deploydemo-dev)	SQL database



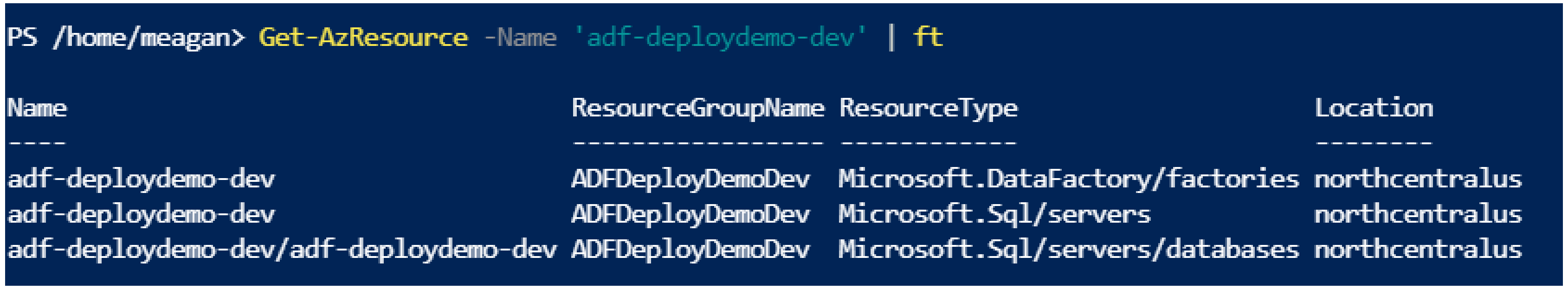
Select members

Select

adf-deploy

adf-deploydemo-dev

adf-deploydemo-dev



```
PS /home/meagan> Get-AzResource -Name 'adf-deploydemo-dev' | ft

Name                                  ResourceGroupName ResourceType                      Location
----                                  ----------------- ------------                      --------
adf-deploydemo-dev                    ADFDeployDemoDev  Microsoft.DataFactory/factories   northcentralus
adf-deploydemo-dev                    ADFDeployDemoDev  Microsoft.Sql/servers             northcentralus
adf-deploydemo-dev/adf-deploydemo-dev ADFDeployDemoDev  Microsoft.Sql/servers/databases   northcentralus
```

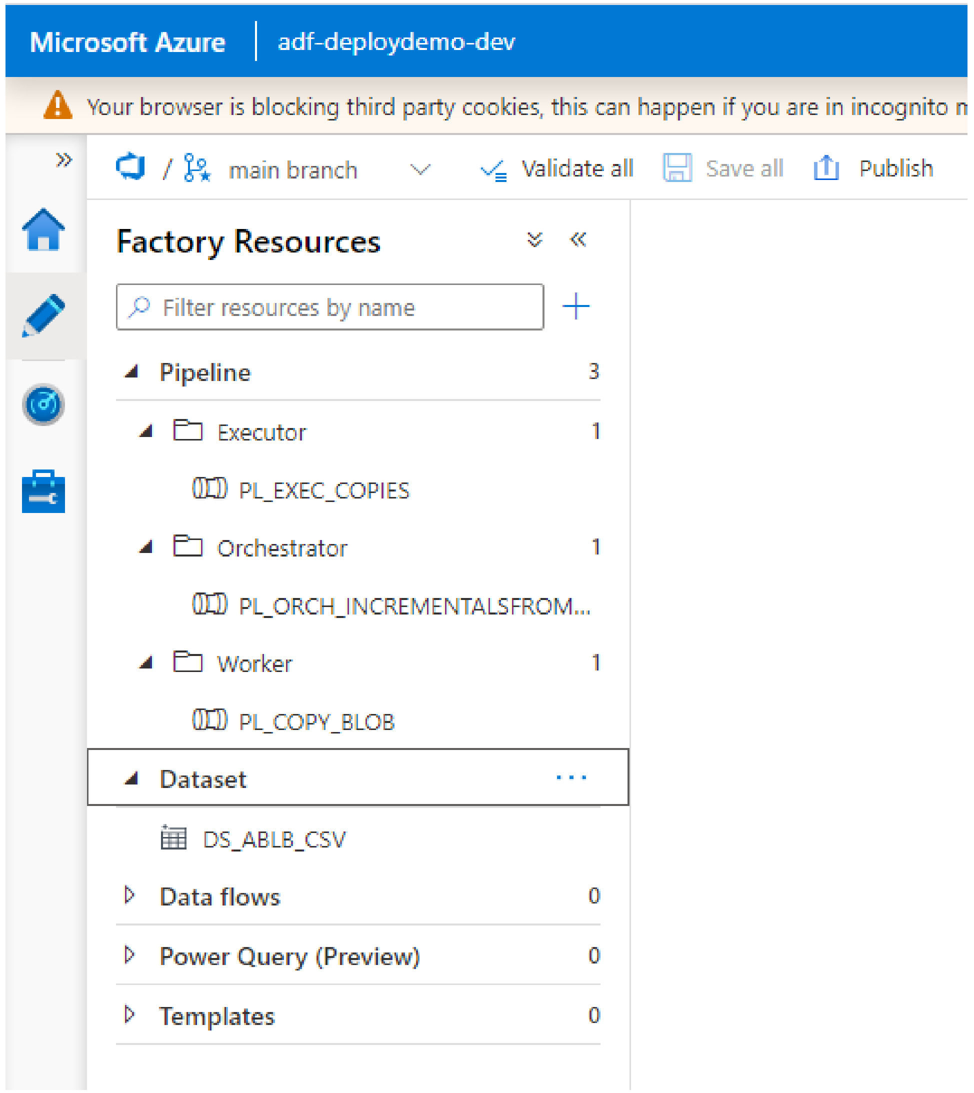
Naming Data Factory artifacts

Use abbreviations for artifact type:

- PL – pipeline
- DS – dataset
- LS – linked service

- Pipelines should indicate what they do (copy, transform, execute SSIS)
- Datasets and linked service names should indicate type and subject of data

Artifact naming example


Microsoft Azure | adf-deploydemo-dev

Your browser is blocking third party cookies, this can happen if you are in incognito m

main branch · Validate all · Save all · Publish

Factory Resources

Filter resources by name

- Pipeline 3
  - Executor 1
    - PL_EXEC_COPIES
  - Orchestrator 1
    - PL_ORCH_INCREMENTALSFROM...
  - Worker 1
    - PL_COPY_BLOB
- Dataset
  - DS_ABLB_CSV
- Data flows 0
- Power Query (Preview) 0
- Templates 0

Key Vault

Store credentials in Azure Key Vault

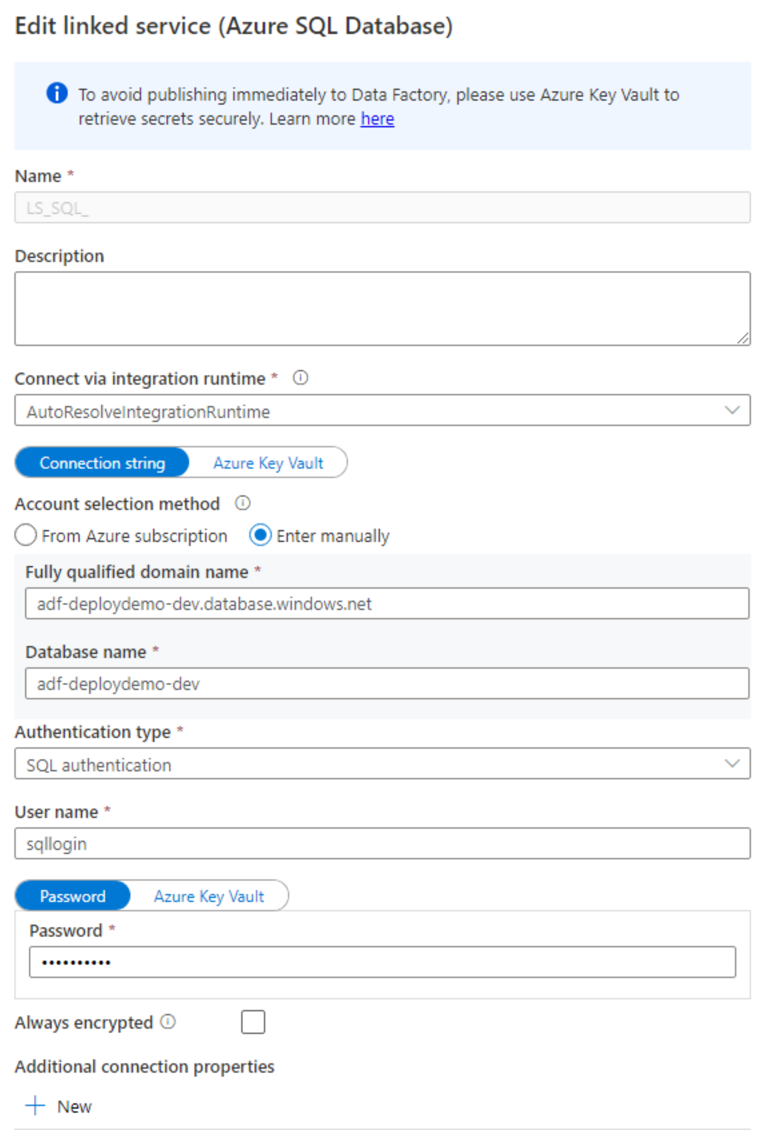
Key Vault

Centralized, more secure

Use the AKV linked service or a web activity to retrieve credentials

Keeps linked service from being immediately published, stays with branch

Data Factory with Key Vault Demo



Edit linked service (Azure SQL Database)

To avoid publishing immediately to Data Factory, please use Azure Key Vault to retrieve secrets securely. Learn more here

Name *

LS_SQL_

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

[✓ Connection string] Azure Key Vault

Account selection method

From Azure subscription [✓ Enter manually]

Fully qualified domain name *

adf-deploydemo-dev.database.windows.net

Database name *

adf-deploydemo-dev

Authentication type *

SQL authentication

User name *

sqllogin

[✓ Password] Azure Key Vault

Password *

••••••••••

Always encrypted ☐

Additional connection properties

+ New

Version Control

DevOps Configuration

Version Control

One project

One repo connected to development factory

Consequences for multiple repos

Connecting multiple factories to the same repo doesn't work

Released in 2022:
- Disable publish from ADF Studio
- Use custom comment

Demo

Branching

Permanent branches: main, integration

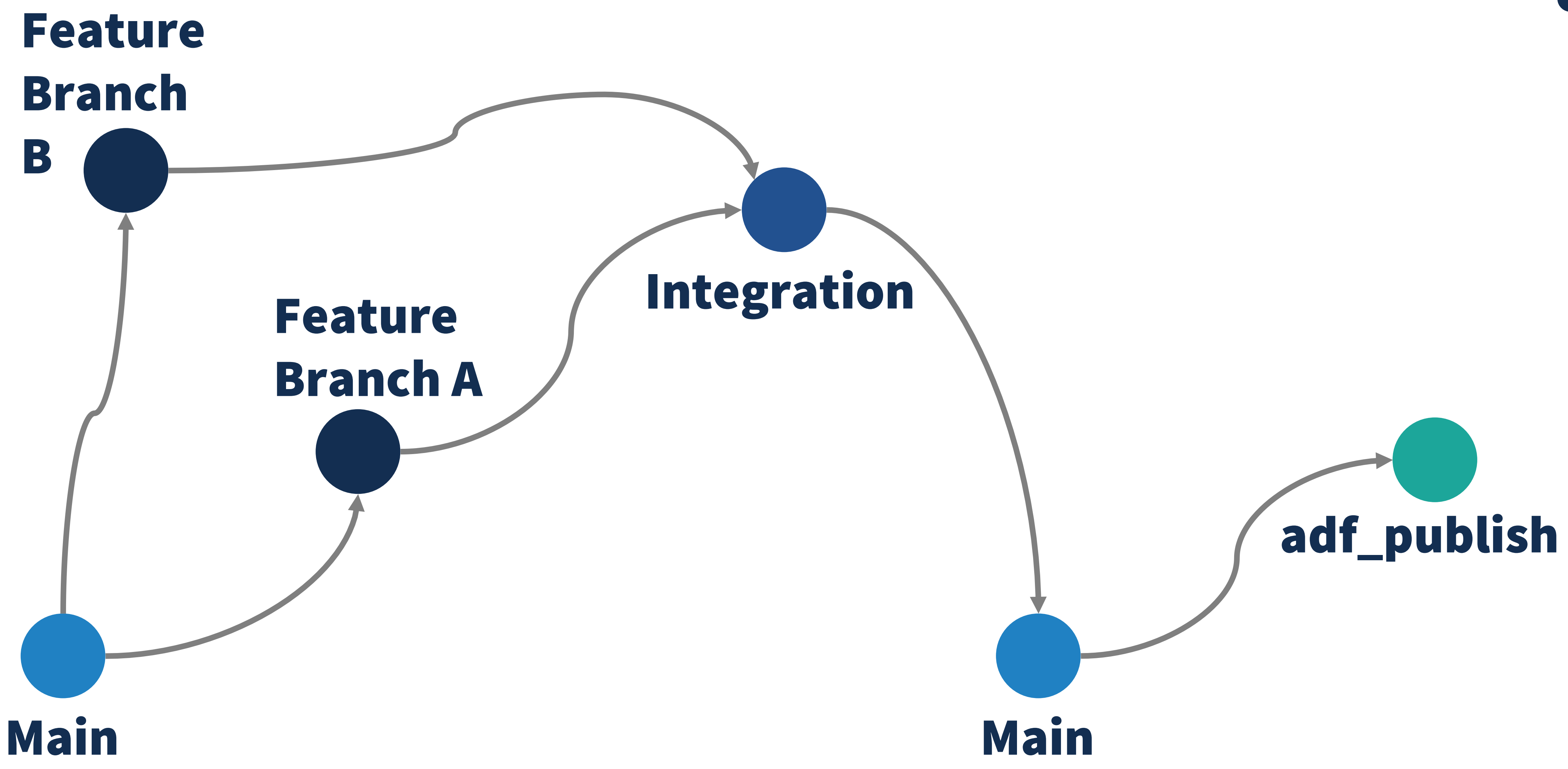
Developers should work in short-lived feature branches

After unit testing, developers merge to integration

After integration testing, pull request to main

Main should always contain code that is ready to be deployed to the next environment

Branching and publish example



Deployment

Deployment

Ways to deploy

Main question:
Copy JSON files or ARM template?

Next question:
Manual, PowerShell/CLI, or DevOps pipeline?

Demo

ARM templates

Deployment can be manual or automated

Use ADF global parameters to change pipeline values for different environments

Use ARM template parameters for linked services values in different environments.

Requires that all ADF artifacts be deployed each time

Requires that parameterized elements are exposed in template parameters

ARM templates plus additional steps

You may want to:

Be sure you have generated current ARM template

Stop triggers before deploying and restart after

Add/update triggers after deployment

Store ARM template parameters file for each environment

Update any additional values/delete extra objects

Deploy JSON files

Deployment can be manual or automated

Files are deployed from a chosen source control branch (usually main)

Use ADF global parameters to change pipeline values for different environments

Use a reference file and code (PowerShell) to update values or substitute an individual JSON file

Allows for selective deployment

DevOps pipeline with Deploy Data Factory

Azure DevOps and the Deploy Azure Data Factory by SQLPlayer extension (free)

Use JSON files in designated branch in source control

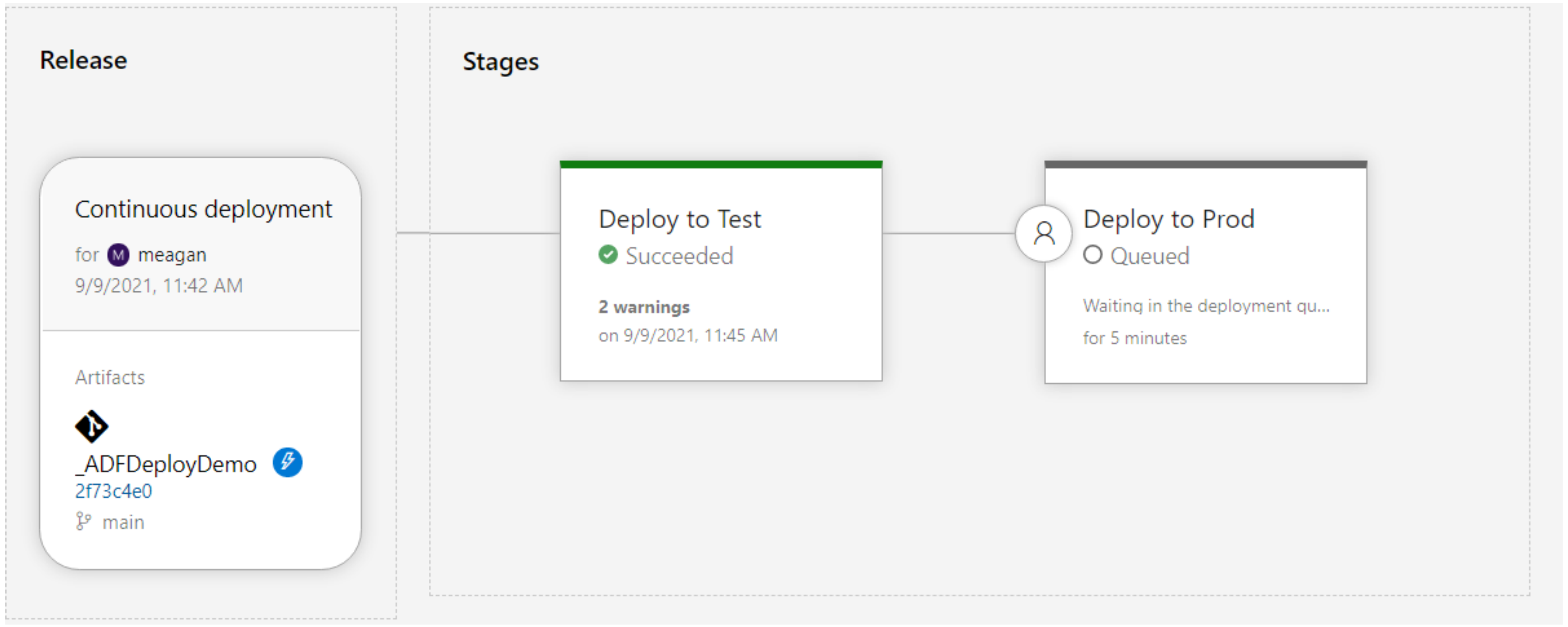
Selective deployment

Config files stored as CSV

Choose whether to delete objects in target not in source

Choose whether to stop/start triggers

DevOps release pipeline



Release

Continuous deployment
for M meagan
9/9/2021, 11:42 AM

Artifacts
_ADFDeployDemo
2f73c4e0
main

Stages

Deploy to Test
Succeeded
2 warnings
on 9/9/2021, 11:45 AM

Deploy to Prod
Queued
Waiting in the deployment qu...
for 5 minutes

Demo

Integration Runtimes

Integration Runtimes

Types

Azure

Self-hosted

SSIS

Integration Runtimes

Self-hosted integration runtimes

Needed with any private network (even in Azure)

Give it the cores, RAM, hard drive space it needs

Share IRs for lower environments to save costs

Size appropriately for concurrent workloads when sharing

Make sure appropriate libraries are installed and updated

Azure integration runtime

Integration Runtimes

Used for copy between cloud data stores and for data flows

Auto-scales based upon prescribed DIUs

Provision your Azure IR so you are sure of the region and avoid data egress charges

Be sure to set TTL when using data flows

Carefully monitor performance with Managed vNet

Parameterization

Parameterize your factory

Parameters

Global parameters

Pipeline parameters

Dataset parameters

Linked service parameters

Parameters

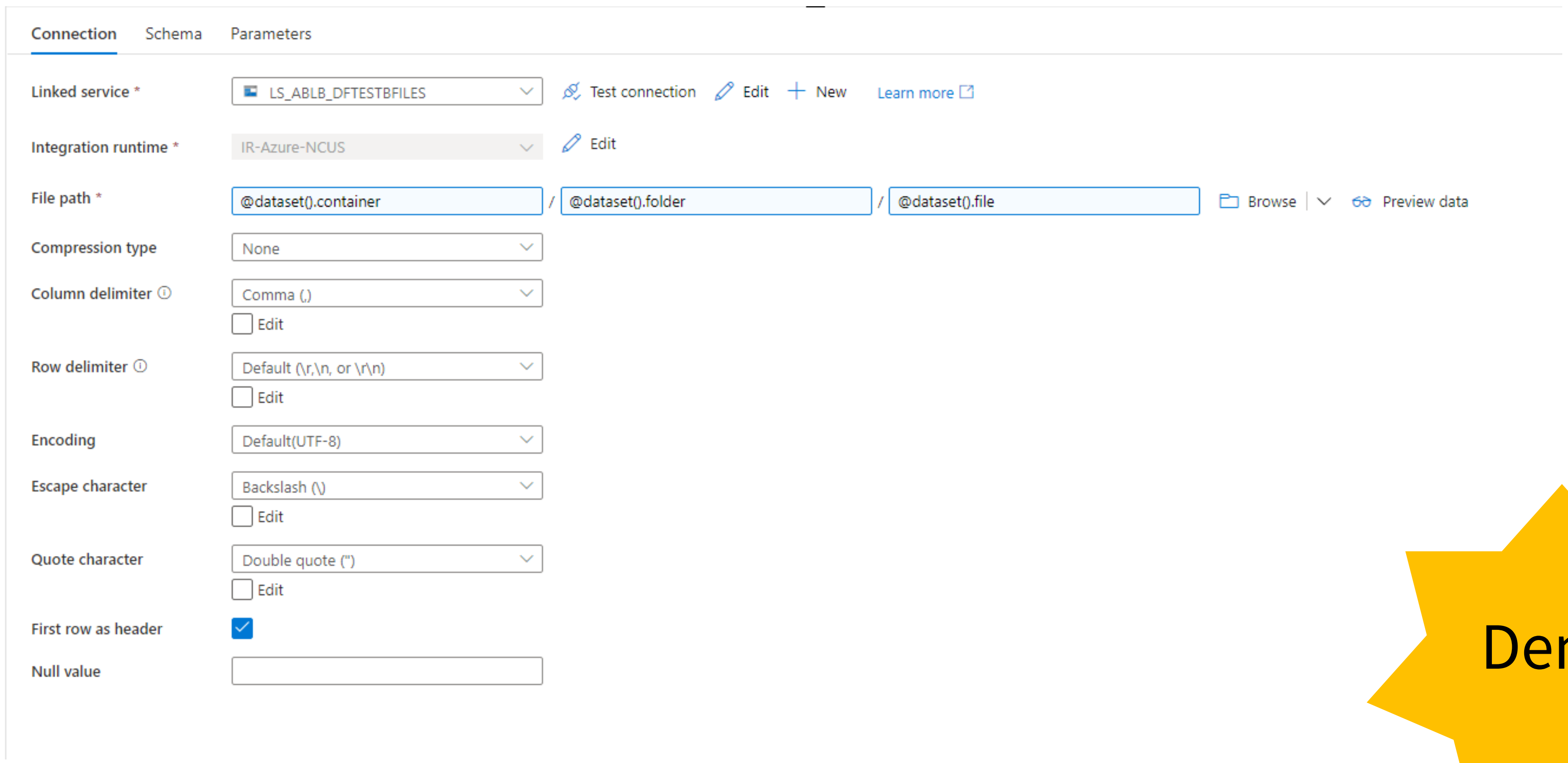
General guidance

Parameterize datasets. It’s easy to have parameter explosion if you don’t.

Linked Services can be 1:1 or parameterized. What makes the most sense in your context?

Parameterize pipelines whenever practical, to make them reusable.

Parameterizing datasets



Linked service *	LS_ABLB_DFTESTBFILES
Integration runtime *	IR-Azure-NCUS
File path *	@dataset().container / @dataset().folder / @dataset().file
Compression type	None
Column delimiter	Comma (,) ☐ Edit
Row delimiter	Default (\r,\n, or \r\n) ☐ Edit
Encoding	Default(UTF-8)
Escape character	Backslash (\) ☐ Edit
Quote character	Double quote (") ☐ Edit
First row as header	✓
Null value	

Demo

Design Patterns

Data Factory design patterns

Pipeline hierarchies

Dependencies and error handling

Design Patterns

Dependencies and Error Handling

Ensure you have retries set to handle transient errors

Set timeouts so you don’t have activities stuck for days

Log errors in a way that makes the info easily usable – send data to Log Analytics and/or another database

Understand when a pipeline fails and plan notifications accordingly

ADFStatus.pdf

Comments & Documentation

Documentation

Document in your code

Not possible to comment the json code behind pipelines

Built-in features to provide notes:

- Pipeline description
- Activity description
- Linked service description
- Integration runtime description
- Annotations
- User properties

Additional Documentation

Documentation

Use the wiki in your DevOps project

Document large commits/releases

Final Comments

Helpful Resources

Azure Cloud Adoption Framework: https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming

Data Factory naming convention: https://erwindekreuk.com/2019/04/azure-data-factory-naming-conventions/

Pipeline hierarchies: https://mrpaulandrew.com/2019/09/25/azure-data-factory-pipeline-hierarchies-generation-control/

ADF tools from SQL Player: https://sqlplayer.net/adftools/

Activity failures and pipeline outcomes: https://datasavvy.me/2021/02/18/azure-data-factory-activity-failures-and-pipeline-outcomes/

Set up your data factory for success.

Meagan Longoria

Denny Cherry & Associates Consulting

DCAC*

Datasavvy.me

@mmarie

/in/meaganlongoria

mmarie@techhub.social