Building a Regret-free Foundation for your Data Factory

Meagan Longoria

Consultant
Denny Cherry & Associates Consulting

DCAC

About Me

Intro

Meagan Longoria

Denver, CO

Work at Denny Cherry & Associates Consulting

Microsoft Data Platform MVP

Blogger, Speaker, Author, Technical Editor

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

Top Regrets for Azure Data Factory

Agenda

Poor resource organization in Azure

No/inconsistent key vault usage

Inappropriate use of version control

Tedious, manual deployments

Misunderstanding integration runtimes

Underutilizing parameterization

No established pipeline design patterns

Resource Organization

Resource Organization

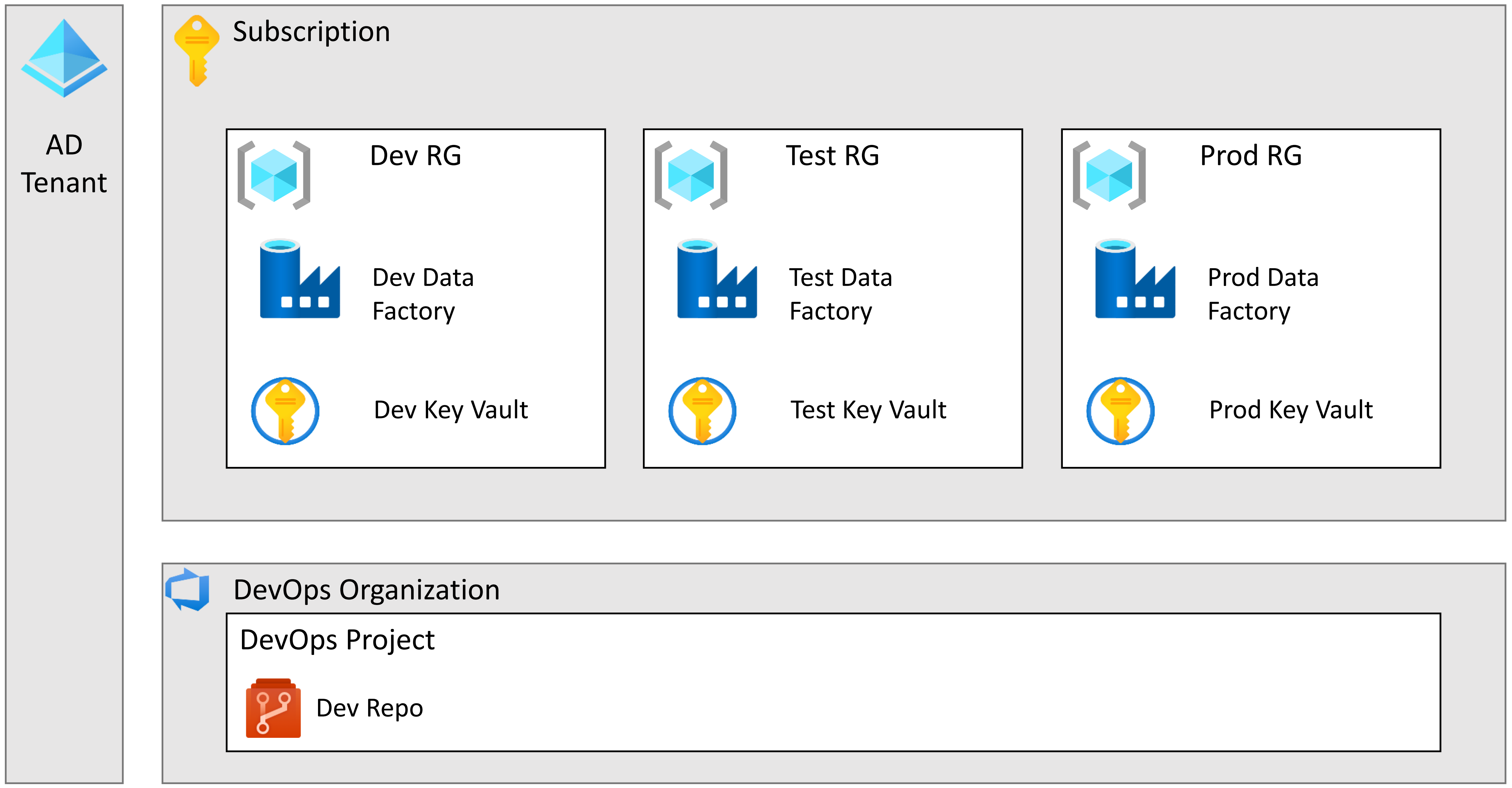
Separating environments

You need separate data factories and key vaults for each environment

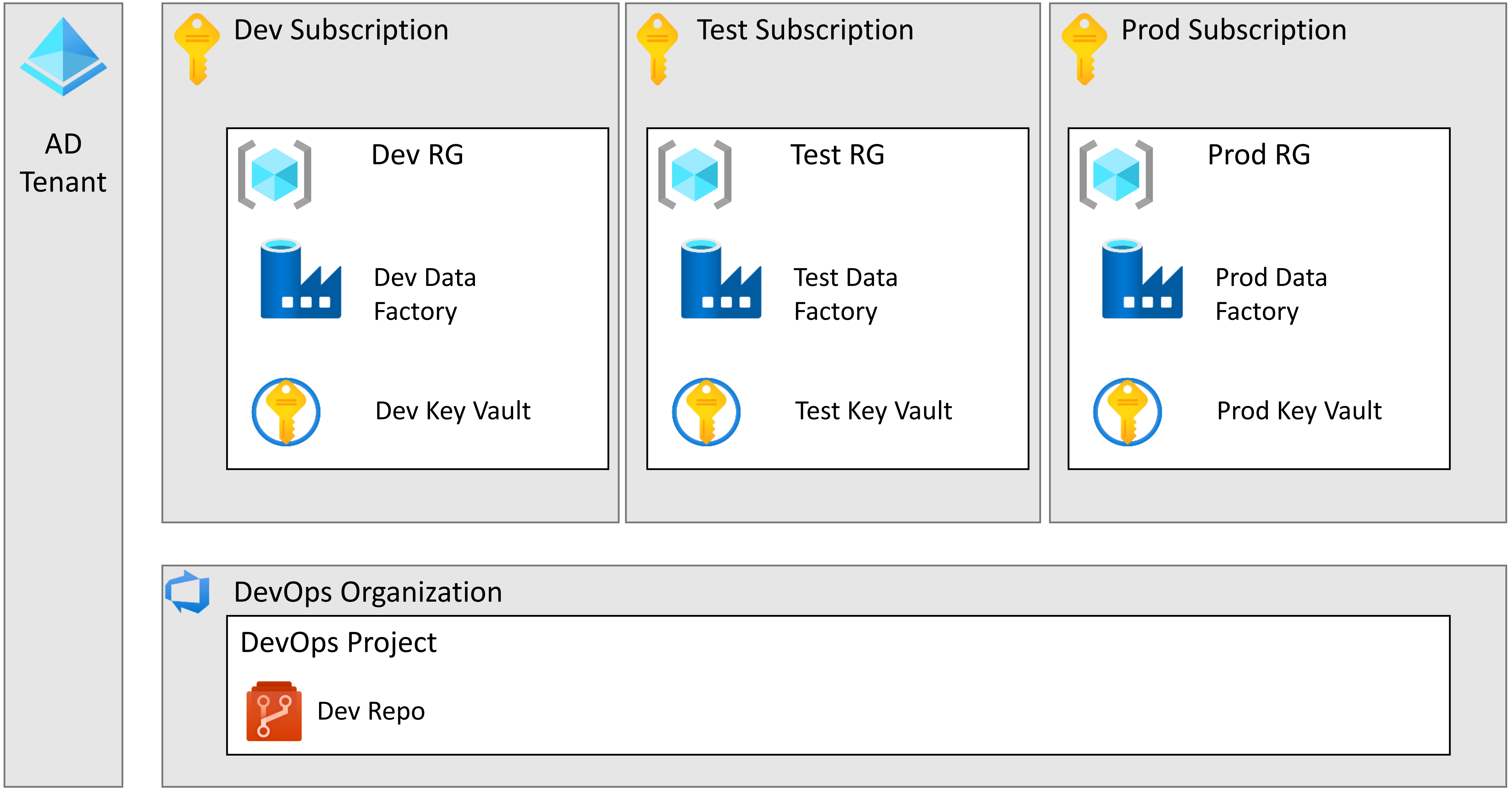
Common containers for separation:

- Resource Groups
- Subscriptions
- Tenants

Option 1: Separate Resource Groups



Option 2: Separate Subscriptions



Key Vault

Store credentials in Azure Key Vault

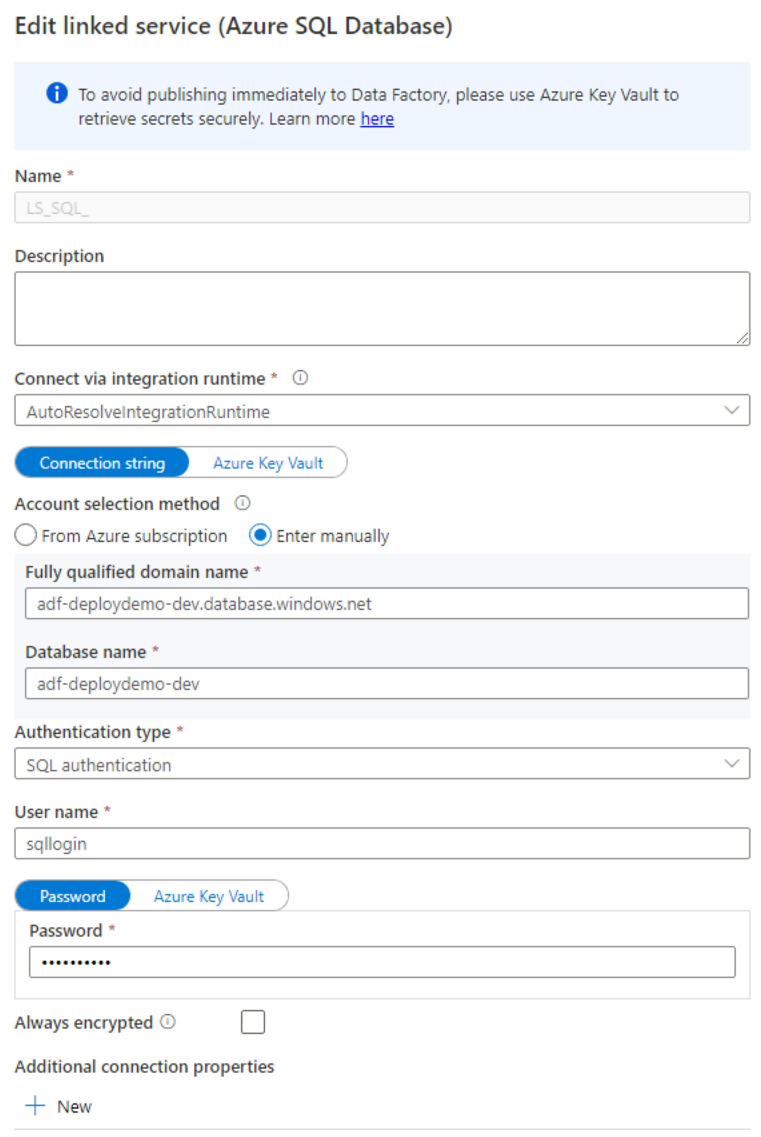
Key Vault

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

Version Control

DevOps Configuration

One project

One repo connected to development factory

Consequences for multiple repos

Connecting multiple factories to the same repo doesn’t work

Disable publish from ADF Studio
Use custom comment

Demo

Branching

Permanent branches: main, integration

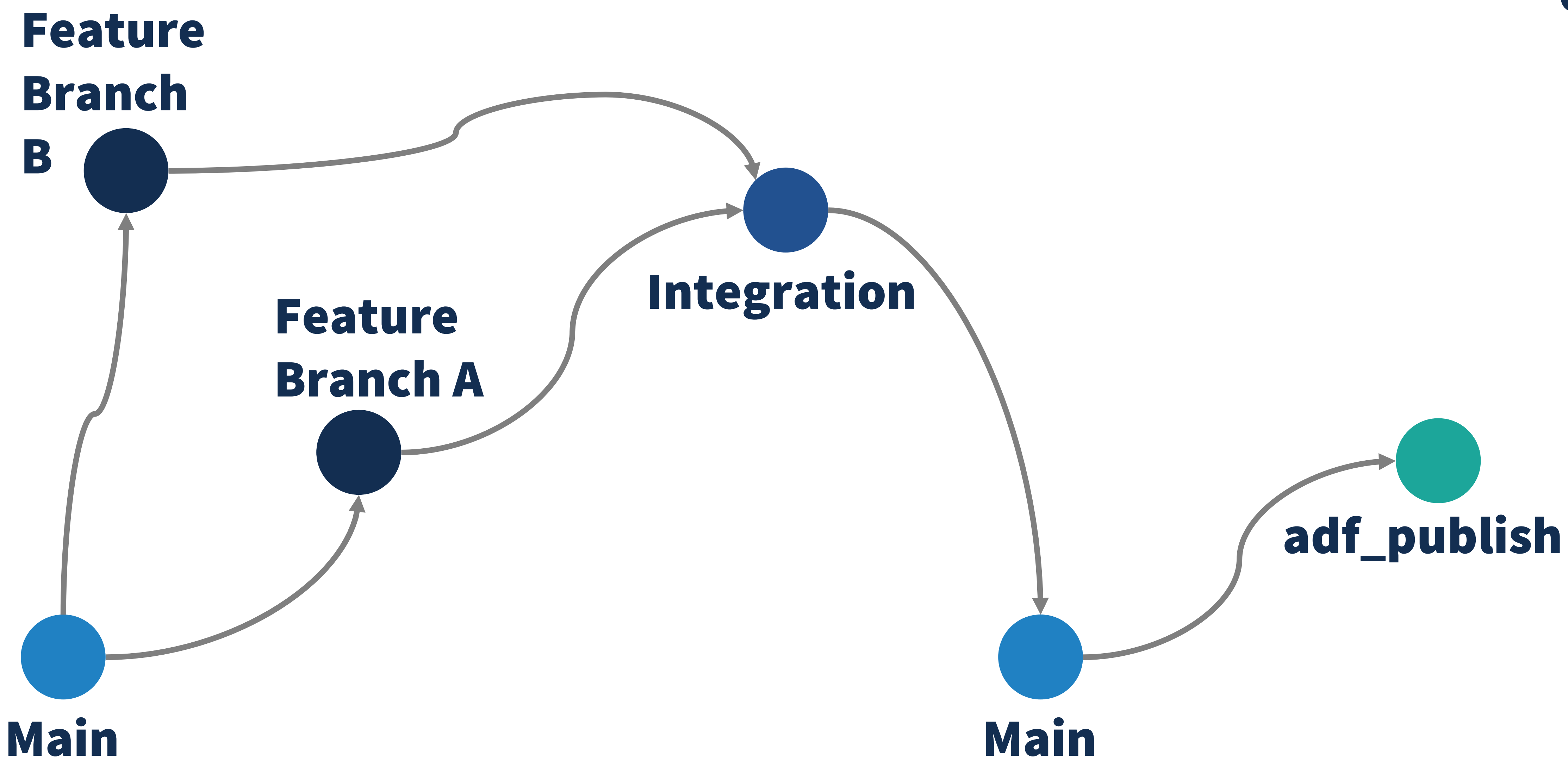
Developers should work in short-lived feature branches

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

Ways to deploy

Deployment

Main question:

Copy JSON files or ARM template?

Next question:

Manual, PowerShell/CLI, or DevOps pipeline?

ARM templates

Deployment can be manual or automated

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

Allows for selective deployment

Requires identifying correct order of deployment

DevOps pipeline with azure.datafactory.tools

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

Use JSON files in designated branch in source control

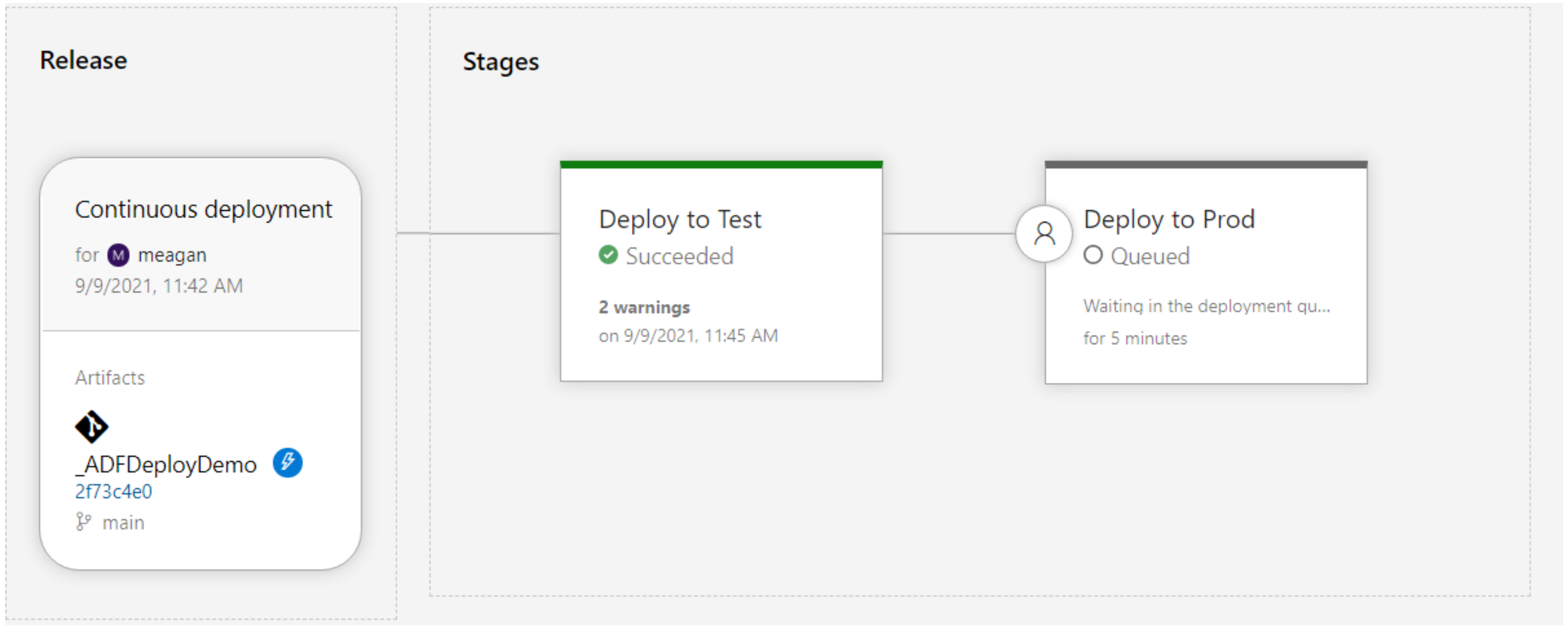
Selective and/or incremental deployment

Config files stored as CSV or JSON

Choose whether to delete objects in target not in source

Choose whether to stop/start triggers

DevOps release pipeline



Release

Continuous deployment
for 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

Types

Azure

Self-hosted

SSIS

Integration Runtimes

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

Integration Runtimes

Azure integration runtime

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 for interactive auth

Use with Managed vNet

Parameterization

Parameters

Parameterize your factory

Global parameters

Pipeline parameters

Dataset parameters

Linked service parameters

General guidance

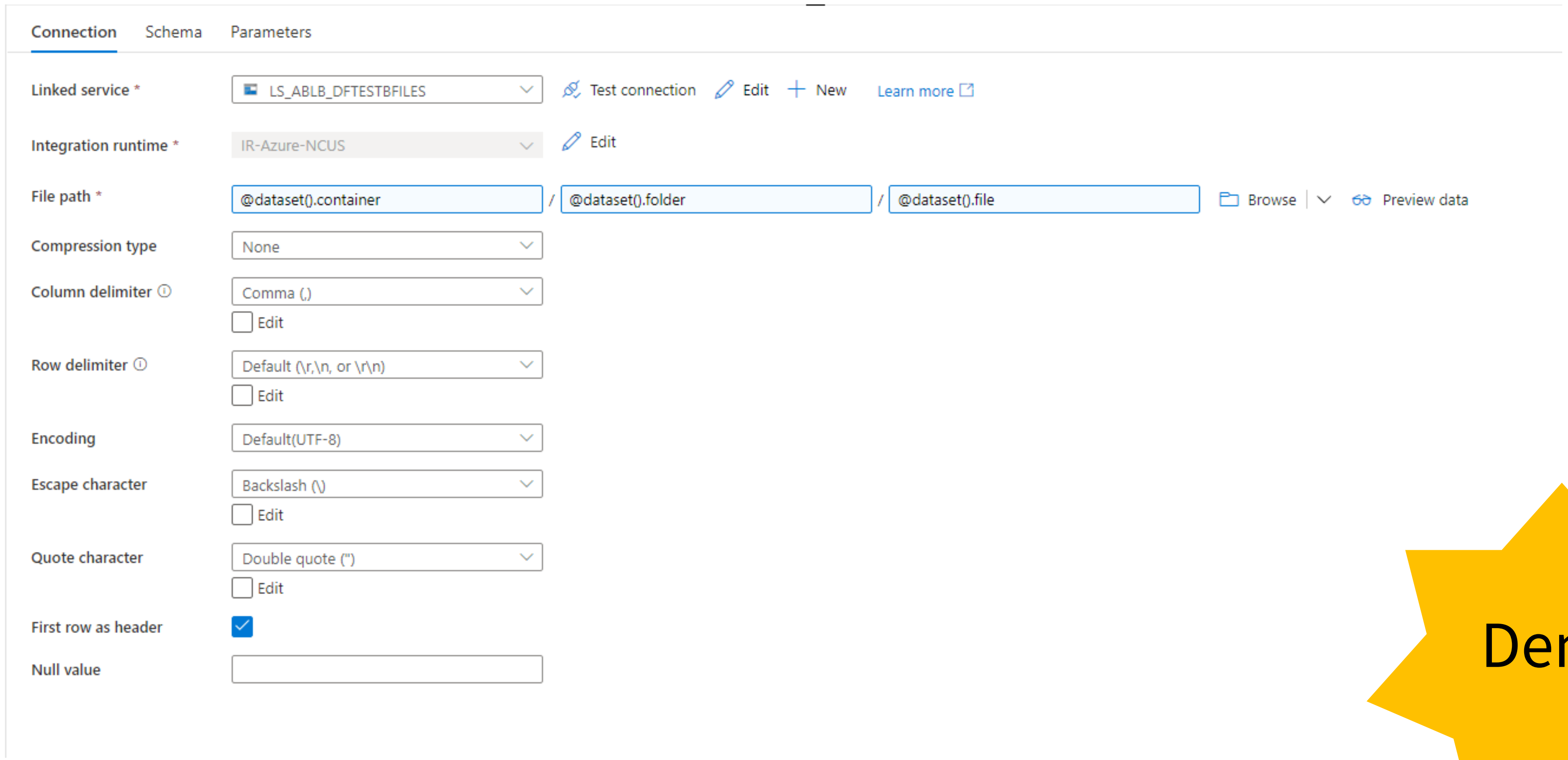
Parameters

Parameterize datasets. It’s easy to have dataset 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



Connection | Schema | Parameters

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

Demo

Design Patterns

Data Factory design patterns

Pipeline hierarchies

Dependencies and error handling

Design Patterns

Pipeline Hierarchies

Orchestrators
Executors
Workers
Utilities

Work around limitations of nested activities

Dependencies and Error Handling

Ensure you have retries set to handle transient errors

Set timeouts so you don’t have activities stuck for hours/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

ADF in Fabric

Biggest Differences

Differences

No datasets

Connections instead of linked services

Schedules instead of triggers

No integration runtimes, gateway instead of SHIR

Workspaces and deployment pipelines

No mapping data flows or SSIS

Monitoring on Fabric capacity

New activities!

Limitations

Limitations

Key Vault integration in connections

Pipelines scoped to items in their workspace

Managed identity requires F64 capacity or higher

Capacity management

Tour

Quick tour of Fabric Data Factory

Deployments in Fabric

Deployment pipelines

Connections use absolute reference instead of a relative name

Must parameterize the connection reference

Demo

New activities

Activities

Office 365 Outlook

Teams

Dataset Refresh

Dataflow Gen2

Demo

What's Coming in Q2 2024

Connect to data sources with service principal auth

Execute Spark jobs

Execute HDInsight jobs

Invoking cross-workspace data pipelines

Event-driven triggers

Apache Airflow

New connectors for Copy activity

Final Comments

Helpful Resources - ADF

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/

Helpful Resources - Fabric

Activity continuity between Azure Data Factory (ADF) and Data Factory in Fabric: https://learn.microsoft.com/en-us/fabric/data-factory/activity-parity

ADF to Fabric DF feature mapping: https://learn.microsoft.com/en-us/fabric/data-factory/compare-fabric-data-factory-and-azure-data-factory

Fabric Data Factory release plan: https://learn.microsoft.com/en-us/fabric/release-plan/data-factory

Dynamic Warehouse & Lakehouse Connections in Microsoft Fabric Data Pipelines: https://sqlkover.com/dynamic-warehouse-lakehouse-connections-in-microsoft-fabric-data-pipelines/

Meagan Longoria

Denny Cherry & Associates Consulting

DCAC*

Set up your data factory for success.

Datasavvy.me

@mmarie

/in/meaganlongoria

mmarie@techhub.social