DevOps with ADF & Databricks

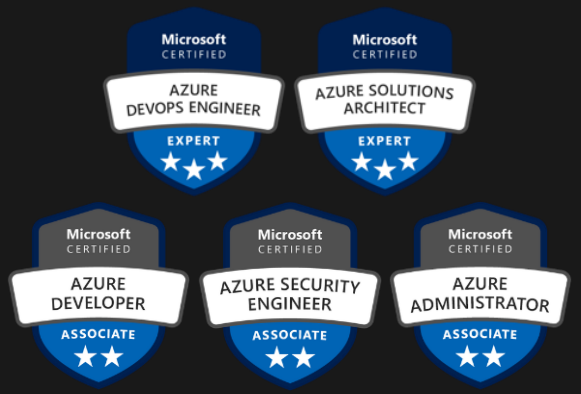
Pasi Huuhka

Pasi Huuhka

DevOps Architect

pasi.huuhka@zure.com

- DevOps expert & Developer from Finland
- Working on Azure since 2014
- Helped to develop & automate applications on Azure for 20+ customers from startups to enterprises



- Twitter: @DrBushyTop
- Blog: huuhka.net
- zure.ly/pasi-huuhka

- zure.ly/faug
- zure.ly/gdbc-2020

100%
Azure since 2011

52 / 55
experts

14,2
experience avg.

4,6 / 5
customer satisfaction

4
Azure MVPs

2
Offices

Microsoft Partner
Microsoft

Gold Application Development
Gold Cloud Platform
Gold Data Analytics
Gold Data Platform
Gold DevOps

MVP
Microsoft® Most Valuable Professional

Microsoft CERTIFIED Trainer

Partner Seller
Microsoft

Microsoft Partner

2019 Partner of the Year Finalist
Application Innovation Award

ZURE

@DrBushyTop

Overview of the session

- You will learn the following about Azure Data Factory v2
  - What is it?
  - How do I deploy it?
  - What are the development flow options?
  - Other magic tricks
- We will also look at Azure Databricks with similar themes
- Walkthrough of a use case & DevOps flow of the whole solution

Service Intros

Azure Data Factory v2

Azure Data Factory v2

A cloud-based data integration service that allows you to orchestrate and automate data movement and data transformation

Code-Free ETL as a Service

INGEST

- Multi-cloud and on-prem hybrid copy data
- 90+ native connectors
- Serverless and auto-scale
- Use wizard for quick copy jobs

CONTROL FLOW

- Design code-free data pipelines
- Generate pipelines via SDK
- Utilize workflow constructs: loops, branches, conditional execution, variables, parameters, …

DATA FLOW

- Code-free data transformations that execute in Spark
- Scale-out with Azure Integration Runtimes
- Generate data flows via SDK
- Designers for data engineers and data analysts

SCHEDULE

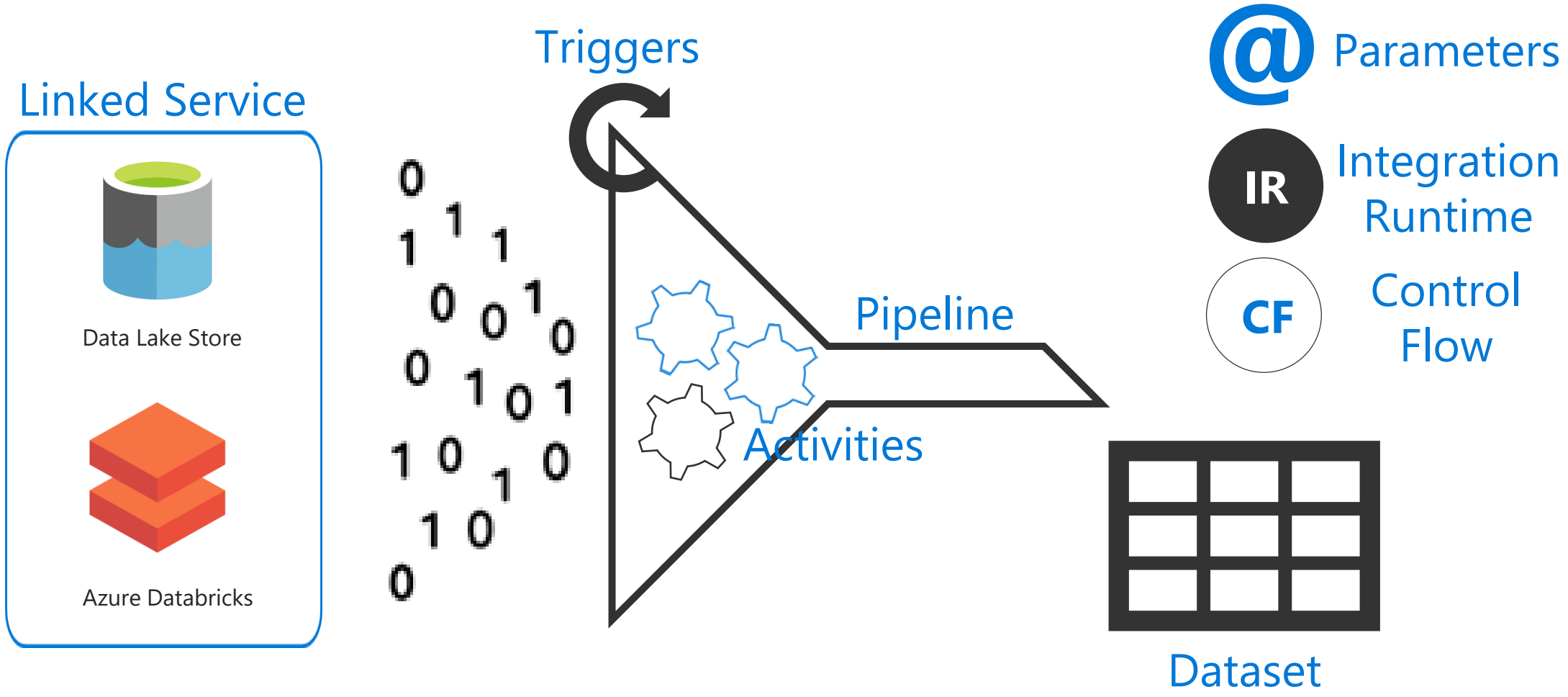
- Build and maintain operational schedules for your data pipelines
- Wall clock, event-based, tumbling windows, chained

MONITOR

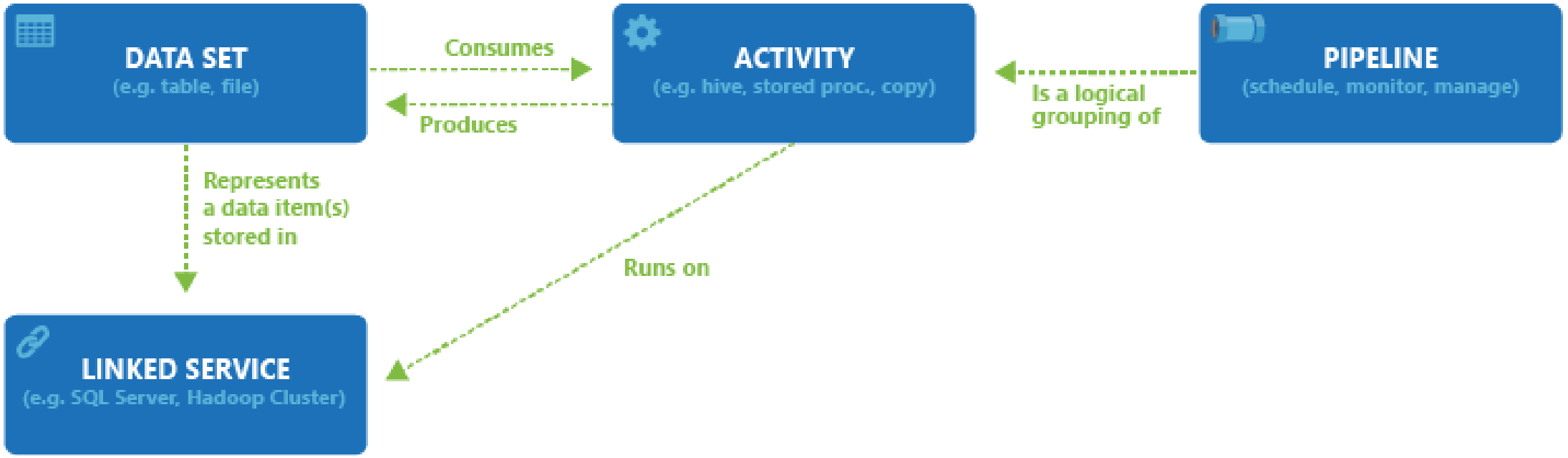
- View active executions and pipeline history
- Detail activity and data flow executions
- Establish alerts and notifications

Azure Data Factory process

Azure Data Factory Components

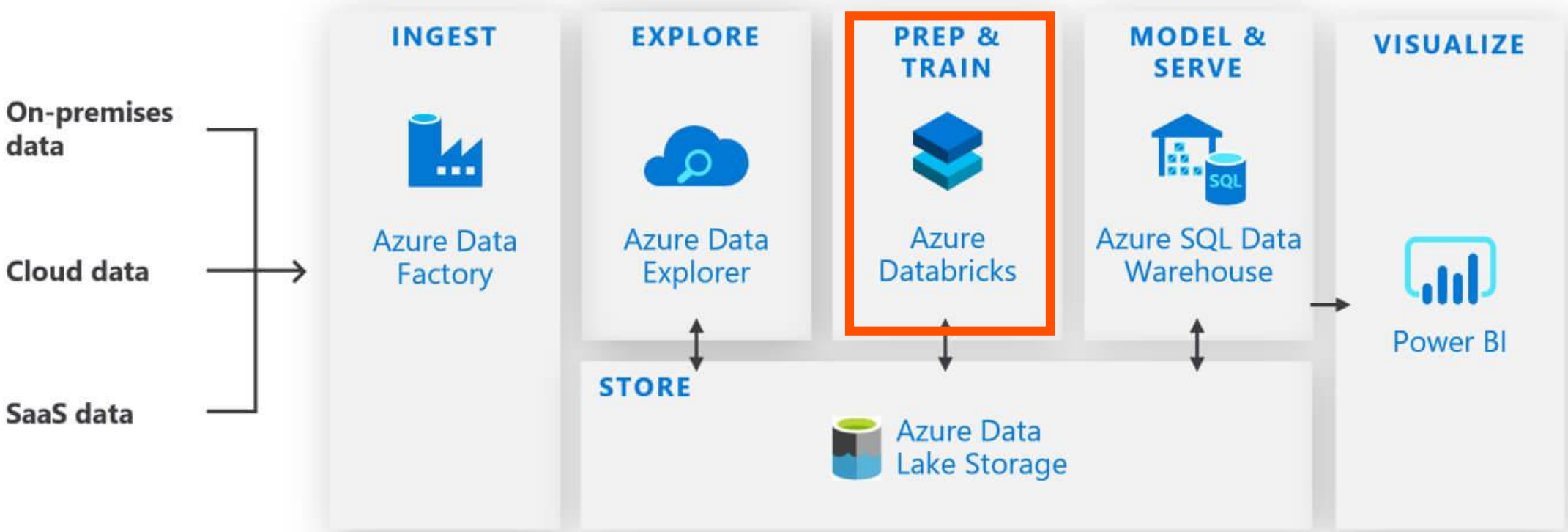


Component dependencies



Azure Databricks

Azure Databricks



Azure Databricks

Fast, easy, and collaborative Apache Spark™-based analytics platform

Increase productivity

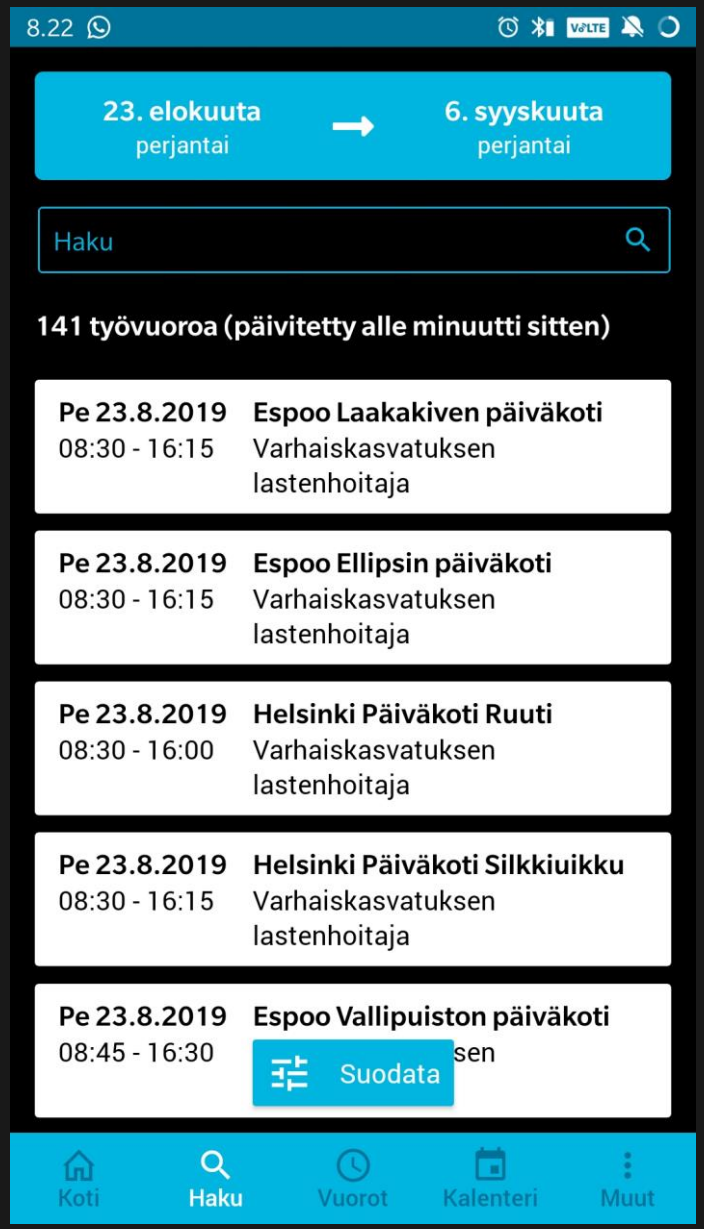
Build on a secure, trusted cloud

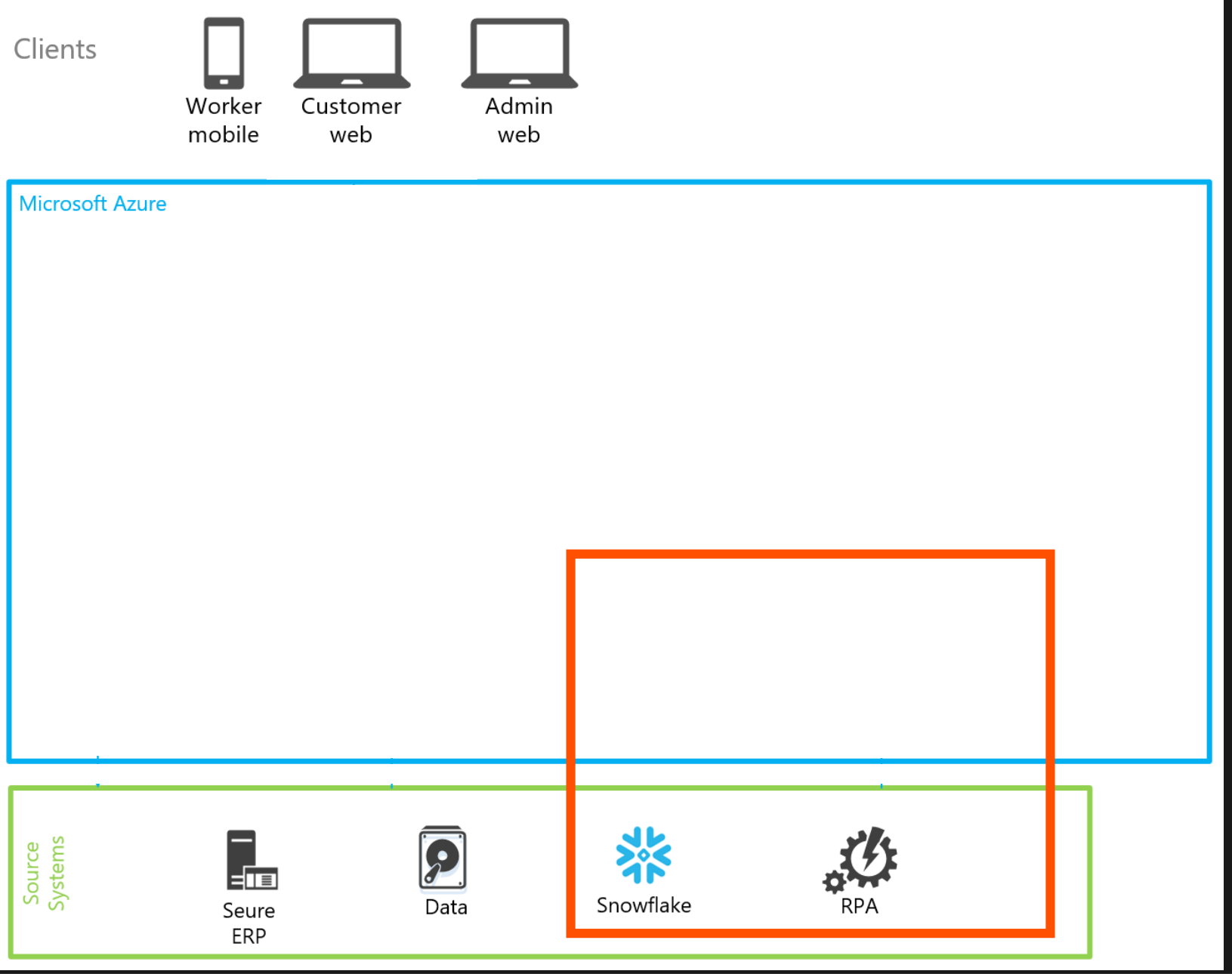
Scale without limits

Built with your needs in mind

- Role-based access controls
- Effortless autoscaling
- Live collaboration
- Enterprise-grade SLAs
- Best-in-class notebooks
- Simple job scheduling

Scenario


8.22
23. elokuuta
perjantai
6. syyskuuta
perjantai
Haku
141 työvuoroa (päivitetty alle minuutti sitten)
Pe 23.8.2019 Espoo Laakakiven päiväkoti
08:30 - 16:15 Varhaiskasvatuksen lastenhoitaja
Pe 23.8.2019 Espoo Ellipsin päiväkoti
08:30 - 16:15 Varhaiskasvatuksen lastenhoitaja
Pe 23.8.2019 Helsinki Päiväkoti Ruuti
08:30 - 16:00 Varhaiskasvatuksen lastenhoitaja
Pe 23.8.2019 Helsinki Päiväkoti Silkkiuikku
08:30 - 16:15 Varhaiskasvatuksen lastenhoitaja
Pe 23.8.2019 Espoo Vallipuiston päiväkoti
08:45 - 16:30
Suodata
Koti
Haku
Vuorot
Kalenteri
Muut


Clients
Worker mobile
Customer web
Admin web
Microsoft Azure
Source Systems
Seure ERP
Data
Snowflake
RPA

Requirements

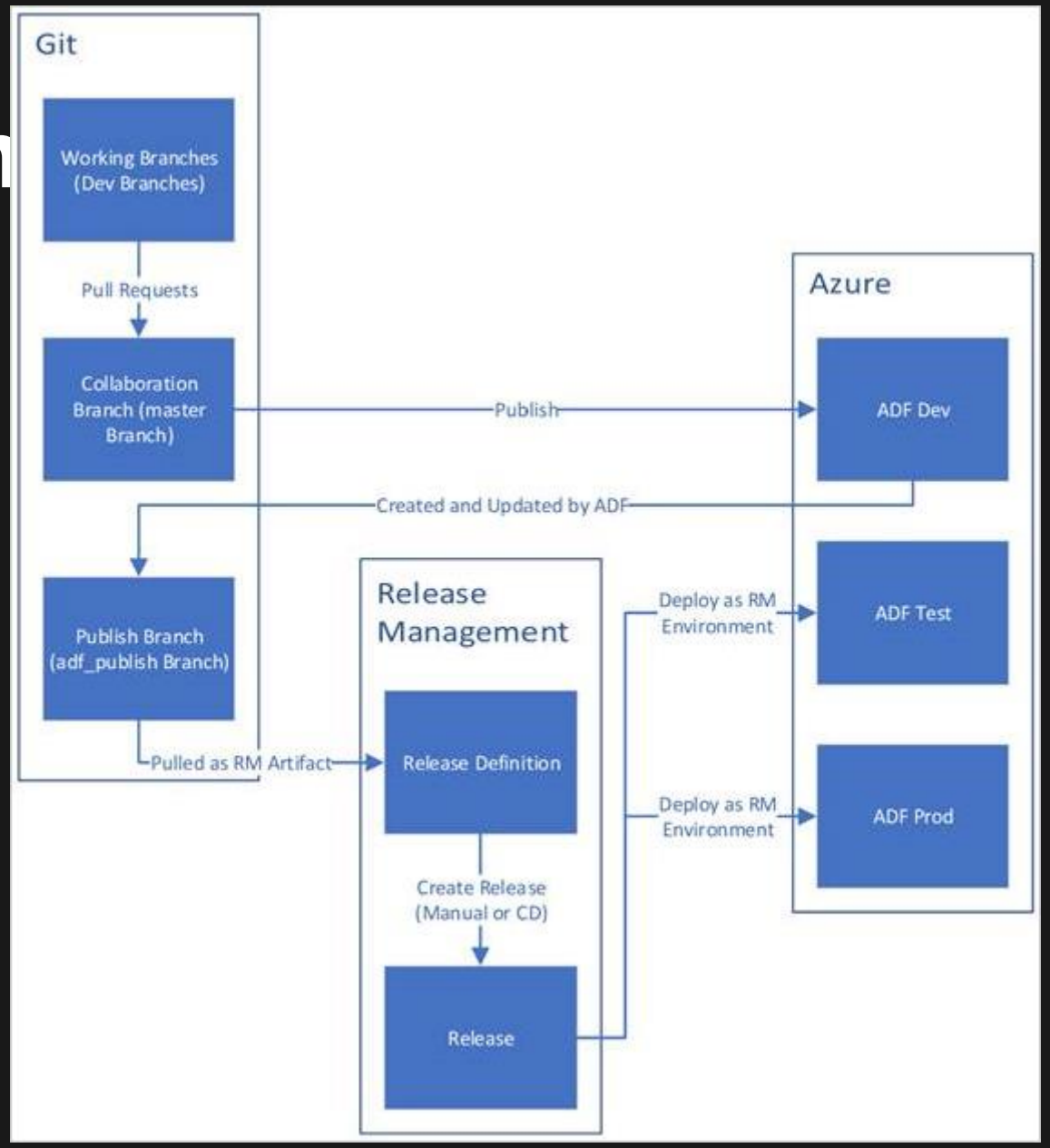
- General
  - ARM templates – complete mode
  - All code version controlled
- ADF
  - Possibility for Unit tests
- Databricks
  - Controlled deployment of custom libraries + notebooks from repository

ADF Development

ADF Development flow

- ADF instance per environment
- Development done in Dev ADF portal
- Either with direct deployment to ADF, or with a git integration
- ADF generates ARM templates behind the scenes
  - These will then get pushed to adf_publish branch
  - Then deployed to Test -> Prod with linked ARM

MS Recom


Git
Working Branches (Dev Branches)
Pull Requests
Collaboration Branch (master Branch)
Publish
Azure
ADF Dev
Created and Updated by ADF
Publish Branch (adf_publish Branch)
Release Management
Deploy as RM Environment
ADF Test
Pulled as RM Artifact
Release Definition
Deploy as RM Environment
ADF Prod
Create Release (Manual or CD)
Release

Simplified alternative

- One ADF instance for all environments
- Publish done from Git collaboration branch
- Pros:
  - Simple to understand, simple to do
- Cons:
  - No tracking of changes to production
  - Requires manual action every time

Portal Demo

Git integration

ARM templates used

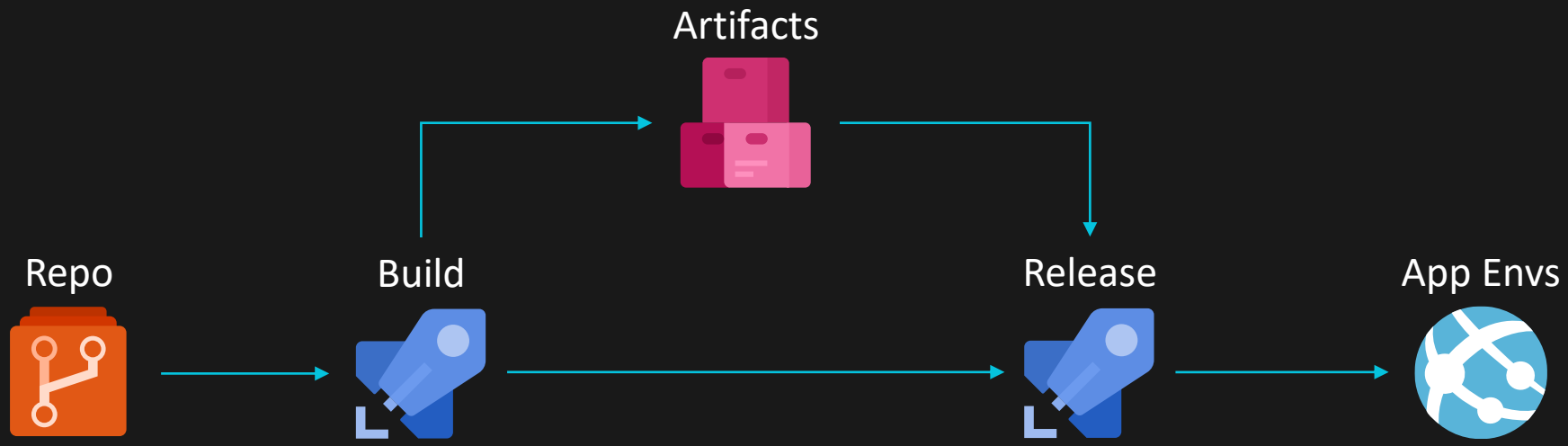
Adf_publish build

Databricks Development

Databricks Development flow

- Databricks workspace instance per environment
- Development done in Dev Databricks portal
- Git integration might only work for same tenant Azure DevOps?
- Notebook & custom library deployments through CI/CD pipelines

Pipeline structure



- Get Source
- Build SQL, databricks libs & notebooks, ARM
- Package, Version & Publish Artifacts

- Deploy to Dev automatically
- Deploy to Test and further with a manual trigger

Portal Demo

Git integration

ARM templates used

CI/CD

Improvements for the future

Things to figure out

- Dev flow for Databricks is a bit cumbersome (manual addition of notebooks etc.)
  - Real Git integration should be taken in use
- Unit testing should be done in adf_publish build
- Get rid of manual steps in deploying to test -> Continuous deployment & speed

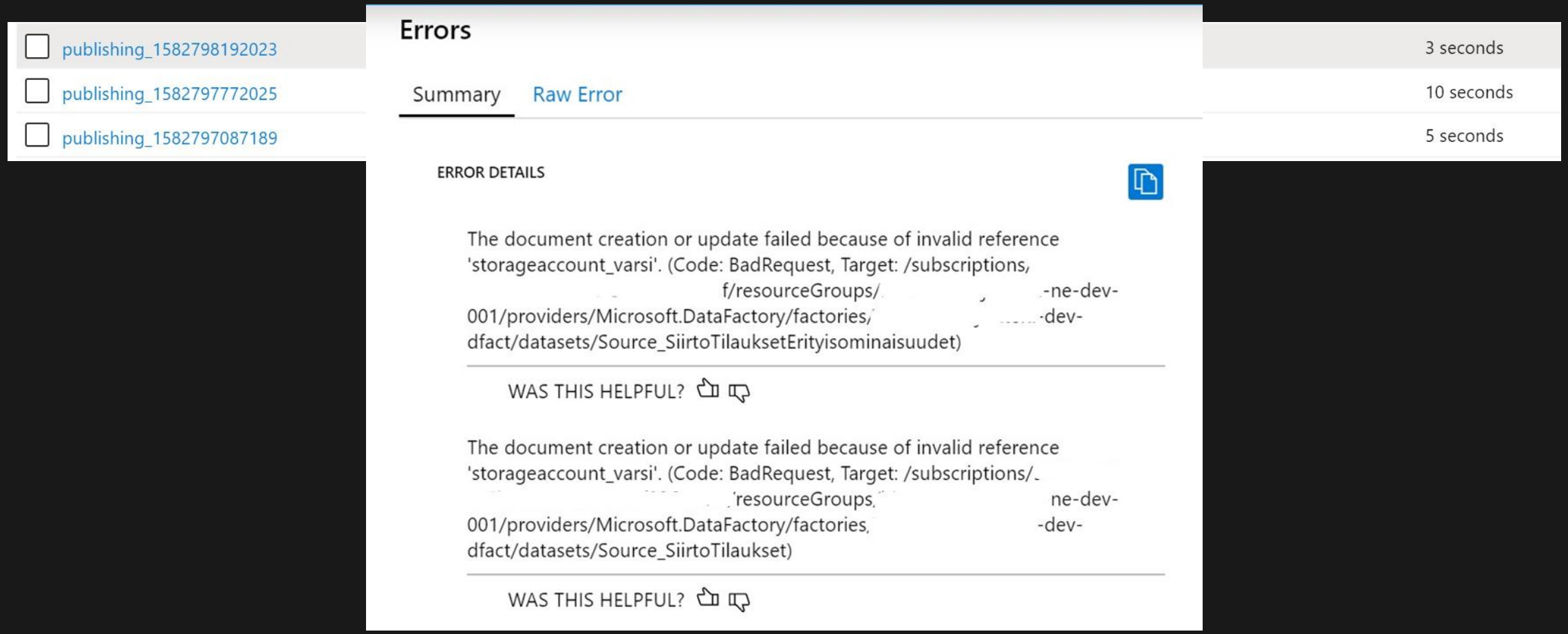
Problems

- Not all connectors for ADF are parameterized in the autogenerated ARM templates ☹
- ADF triggers cannot be updated when they are active
- Troubleshooting can be an issue, error messages from Git integration are unclear
- Databricks does not support automated key vault backed secret scope generation.
- Databricks does not support automated git integration

Troubleshooting example

- **Error**: "The publish branch is out of sync with the collaboration branch. This is likely due to publishing outside of Git mode."
- Suggested solution: "Remove git, add the current branch as new branch -> PR to collab branch"
- Tested alternative: "Recreate whole ADF from scratch -> publish collab branch again"

Troubleshooting example

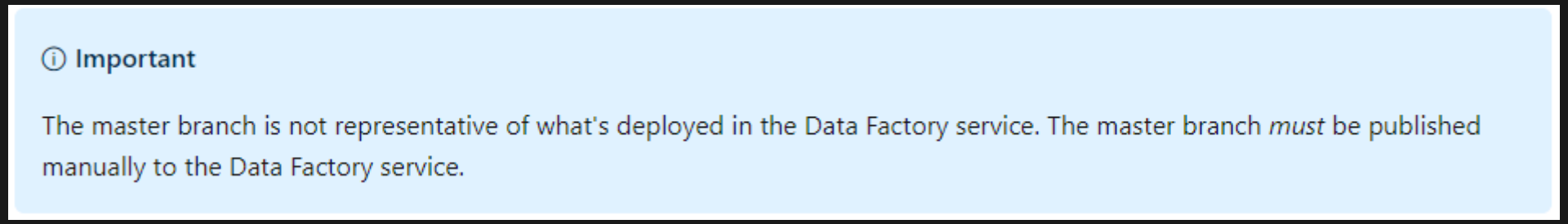


Takeaways

- Utilize Git integrations wherever possible
- Always use a key vault to store your secrets
- Look into linked ARM templates for deployment
- Remember how the services work under the hood when troubleshooting!

Resources & Links

- Unit tests for ADF v2
- ADF connector secrets with Git Integration
- Databricks REST API reference
- huuhka.net



> ⓘ **Important**
>
> The master branch is not representative of what's deployed in the Data Factory service. The master branch *must* be published manually to the Data Factory service.

Slides: zure.ly/pasi/adf

GDBC: zure.ly/gdbc-2020

Questions?

Thank you!


ZURE