DATA() { EXPOSED;

DATA EXPOSED SPECIAL

Around the Clock with Azure SQL and Azure Data Factory

Americas
February 3, 2021
09:00 - 17:00 PT

Asia
February 4, 2021
09:00 - 17:00 SGT

16 Sessions | 2 Ask the Expert Panels | 1 Hackathon

HOSTED BY
Wee Hyong Tok & Anna Hoffman

DATA() { EXPOSED;

Data security best practices for data engineers using Data Factory

Abhishek Narain
Senior Program Manager
@narainabhishek

Security considerations

1. **Credential management**
   - Store in Azure Key Vault and reference from ADF during runtime
   - Store in ADF service (always encrypted)
2. **Authentication**
   - Windows, Basic, Service Principal, Managed identity
3. **Data in transit**
   - Data encrypted in transit through HTTPS/TLS
4. **Data at rest**
   - Leverage data store's encryption-at-rest capability: SQL TDE, Azure Blob/Table Storage SSE, etc
5. **Network**
   - Managed Virtual Network – compute network isolation (data plane)
   - ADF Private link – connect securely to ADF service using private endpoint (control plane)
6. **Meta-data encryption (Meta-data at rest)**
   - ADF managed key encryption
   - Customer managed key (CMK) encryption (double encryption)

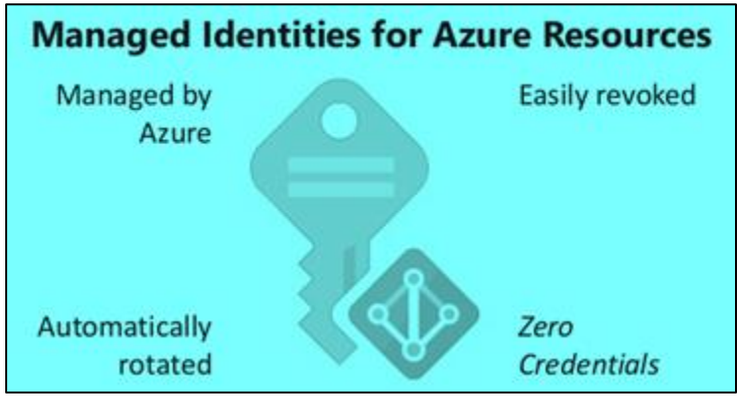
Build password-free data pipelines

Preferred authentication mechanism –

If **Data store == 'MSI' auth**
Then **Use MSI auth (AAD-based)**
Else **Use Key-Vault to store credentials & reference it in the Linked service**
*ADF connects with Key-vault to retrieve the credentials during runtime using 'MSI' authentication

Benefits of using MSI (Managed identity) authentication:

•Managed identities eliminate the need for data engineers having to manage credentials by providing an identity for the Azure resource in Azure AD and using it to obtain Azure Active Directory (Azure AD) tokens. In our case, Data Factory obtains the tokens using its Managed Identity and accesses the respective Data Store or Key Vault.
•It lets you provide fine-grained access control to a particular Data Factory instances using Azure AD.
•Since MSI is unique per Data Factory and is system-assigned there is no way someone can impersonate or use the resolved tokens causing security breaches.



Data access strategies from Azure Data Factory

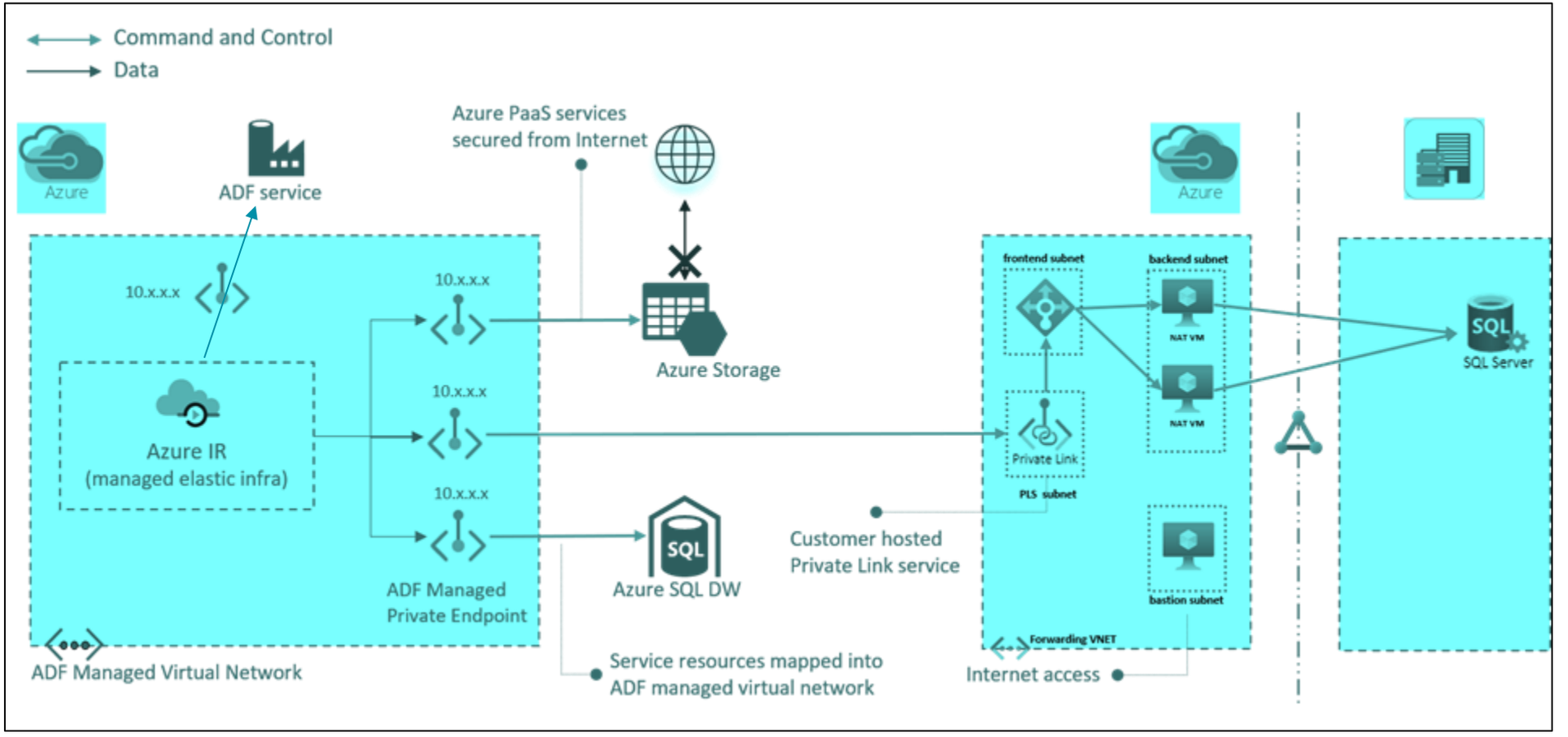
1. Private endpoint (Azure IR in Managed VNet/ SHIR)
2. Static IP (SHIR)
3. Static IP range/ Service Tags (Azure IR)
4. Trusted Service firewall exception with MSI auth (Azure IR)
5. Public endpoint (Allow all service)

		Integration Runtime							
Data Stores		*Azure IR*				*SHIR (Vnet/ on-premise)*		*Azure IR (Managed Vnet)*	
Supported Network Security Mechnaism on Data Stores		**Trusted Service**	**Static IP range**	**Service Tags**	**Allow Azure Services**	**Static IP**	**Trusted Services**	**Private endpoint**	**Public endpoint**
Azure PaaS Data stores	Azure Cosmos DB	-	Yes	-	Yes	Yes	-	Yes	Yes
	Azure Data Explorer	-	Yes*	Yes*	-	-	-	Yes	Yes
	Azure Data Lake Gen1	-	Yes	-	Yes	Yes	-	-	Yes
	Azure Database for MariaDB, MySQL, PostgreSQL	-	Yes	-	Yes	Yes	-	Yes	Yes
	Azure File Storage	-	Yes	-	-	Yes	-	Yes	Yes
	Azure Storage (Blog, ADLS Gen2)	Yes (MSI auth only)	Yes	-	-	Yes	Yes (MSI auth only)	Yes	Yes
	Azure SQL DB, SQL DW (Synapse Analytics), SQL MI	-	Yes	-	Yes	Yes	-	Yes	Yes
	Azure Key Vault (for fetching secrets/ connection string)	Yes (MSI auth only)	Yes	-	-	Yes	Yes (MSI auth only)	Yes	Yes
Other PaaS/ SaaS Data stores	AWS S3, SalesForce, Google Cloud Storage, etc.	-	Yes	-	-	Yes	-	-	Yes
Azure IaaS	SQL Server, Oracle, etc.	-	Yes	Yes	-	Yes	-	Yes (PLS)	Yes
On-premise IaaS	SQL Server, Oracle, etc.	-	Yes	-	-	Yes	-	Yes (PLS)	Yes

Secure pipelines by running in managed virtual network

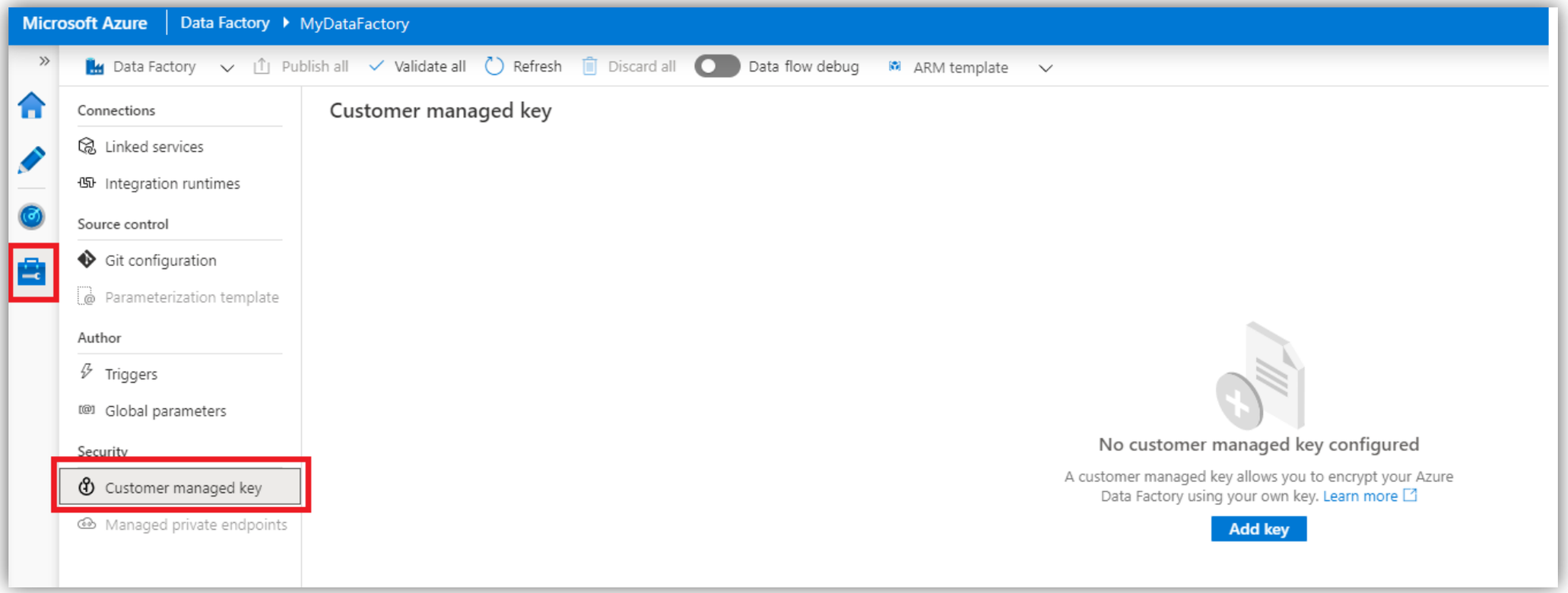
Azure IR in Managed Virtual Network

- Private connection to data stores using Managed Private Network



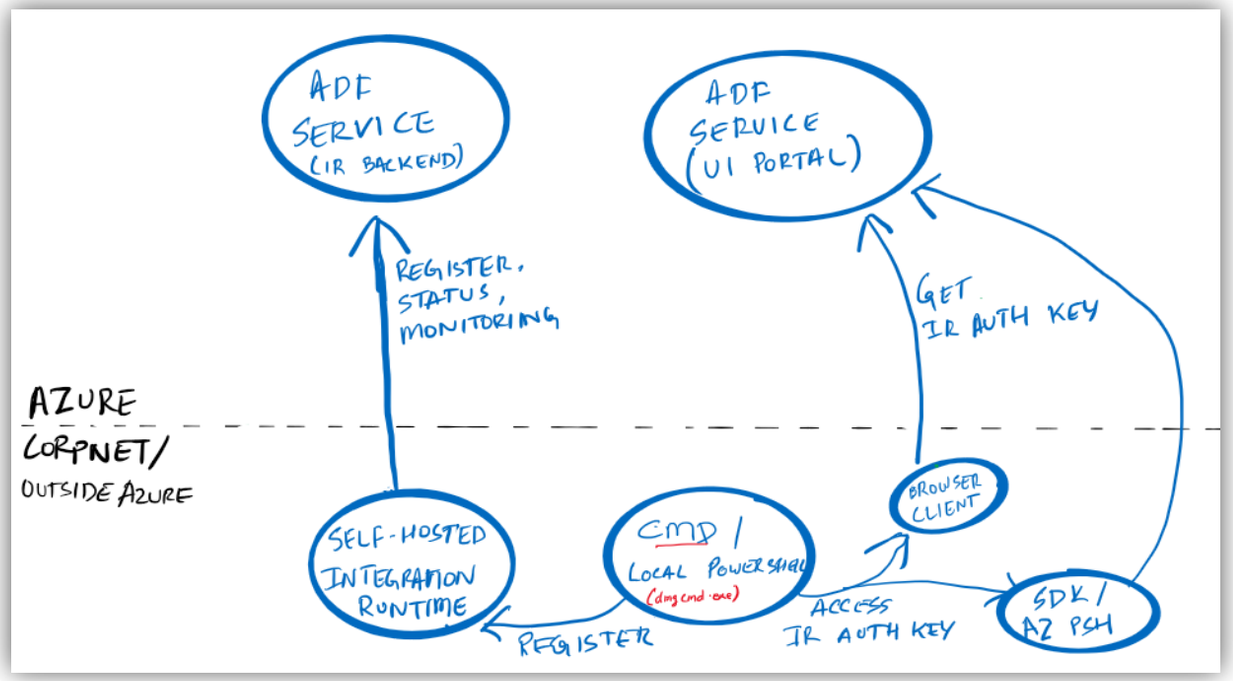
Encrypt ADF metadata using your key

Microsoft encrypts the meta-data using Microsoft managed keys
Customer can additionally encrypt the metadata with their own-keys (CMK)
[Encrypt Azure Data Factory with customer-managed key - Azure Data Factory | Microsoft Docs]



Self-hosted Integration runtime

Registration flow



Create Linked Service on SHIR flow

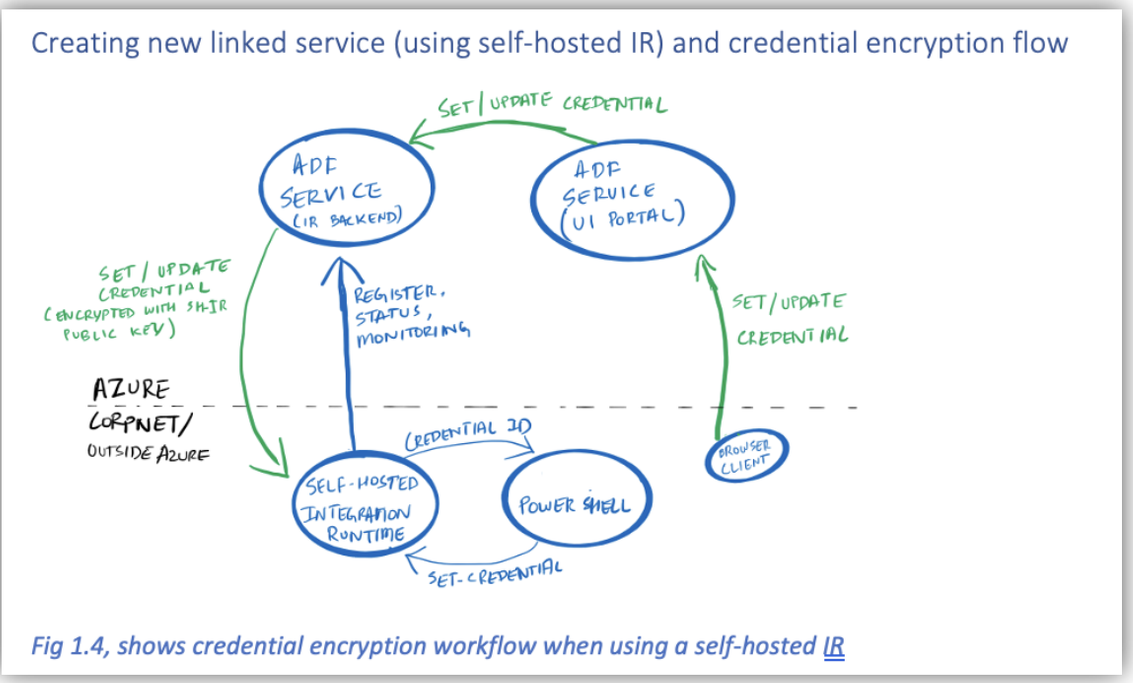


Fig 1.4, shows credential encryption workflow when using a self-hosted IR

Linked service payload encryption

	Payload before encryption (explicit encryption)	Payload after encryption (explicit encryption)
Dataset example	{ "properties": { "type": "AzureSqlTable", "typeProperties": { "tableName": { "type": "SecureString", "value": "dbo.SourceTable" } }, "linkedServiceName": { "referenceName": "SqlAzureLinkedService", "type": "LinkedServiceReference" } }, "name": "SqlAzureDataset" }	{ "properties": { "type": "AzureSqlTable", "typeProperties": { "tableName": { "type": "SecureString", "value": "xxxxxxxxxxxxxxxx", //Base64String } }, "linkedServiceName": { "referenceName": "SqlAzureLinkedService", "type": "LinkedServiceReference" } }, "name": "SqlAzureDataset" }

- You can additionally declare properties as secure string to encrypt the values

DATA() {
EXPOSED;

Learn with us!

View our on-demand playlist:
aka.ms/azuresqlandadf

@AzureSQL
@AzDataFactory

Microsoft Azure