

# Hands-on Lab: Use Azure Key Vault to Authenticate Access to a Storage Account from Azure Data Factory

**Trainer / Creator:** Dr. Sandeep Kumar Sharma

**Level:** Beginner — step-by-step

**Duration:** 40–60 minutes

---

## Learning objectives

By the end of this lab the participant will be able to:

1. Create an Azure Key Vault and securely store a secret (the storage account key).
2. Enable a system-assigned managed identity for Azure Data Factory and grant it permission to read secrets from Key Vault.
3. Create and configure an Azure Key Vault Linked Service in Azure Data Factory.
4. Create a Storage Linked Service in ADF that retrieves its account key from Key Vault (no plaintext key in ADF UI).
5. Run a simple pipeline that uses the secure linked service and verify connectivity.

## Learning outcomes

Participants will understand how to replace embedded account keys with Key Vault secrets and how to allow ADF to securely retrieve those secrets at runtime. They will also learn best-practices for secret handling and an alternative, more secure approach using Managed Identity + Role-Based Access to Storage.

---

## Why not use the storage account key directly? (Problem with traditional approach)

Traditionally teams paste the storage account *access key* into an application or into Data Factory linked service configuration. This approach has several drawbacks:

- **Secret sprawl:** Keys get copied into many places (e.g., peoples' notes, configuration files), increasing exposure.
- **Difficult rotation:** If you rotate the key you must find and update every location that uses it.
- **Higher blast radius:** If a key is leaked, an attacker can access the storage account until the key is rotated.

Using **Azure Key Vault** centralizes secret storage, provides audit trails, and makes rotation and access control easier. When combined with ADF managed identity, you can keep secrets out of configuration and reduce risk further.

---

## Prerequisites

- An Azure subscription with permission to create Resource Groups, Storage Accounts, Key Vaults, and Azure Data Factory.
- Owner or Contributor permissions on a resource group for this lab (or appropriate delegated permissions).
- Basic familiarity with the Azure Portal.

Note: This lab demonstrates storing an *existing* storage account key in Key Vault and referencing it. For higher security, prefer using ADF Managed Identity + Azure RBAC on the storage account (shown in the Recommendations section) which avoids storing keys at all.

---

## Architecture (simple)

```
[ADF (system-assigned managed identity)]
  | (Key Vault access: Get secret)
  v
[Azure Key Vault] --stores--> secret: storage-account-key
  |
  v
[Azure Storage Account] <-- secret value = account key used by Linked Service
```

## Lab steps (step-by-step)

### Step 0 — Create a Resource Group (optional)

1. Sign in to the Azure portal.
2. Search **Resource groups** → **+ Create**.
3. Select Subscription, enter **Resource group name** e.g., `rg-adf-keyvault-lab`, choose Region, then **Review + Create** → **Create**.

### Step 1 — Create (or identify) the Storage Account

1. In the portal, search **Storage accounts** → **+ Create**.
2. Subscription: choose your subscription. Resource group: select `rg-adf-keyvault-lab`.
3. Storage account name: `adflabstorage<random>` (must be globally unique). Keep defaults (or enable hierarchical namespace if desired for ADLS Gen2).
4. Click **Review + Create** → **Create**.

5. After creation, go to the storage account → **Access keys** under **Security + networking**. Copy one of the **key** values (you will store this in Key Vault in Step 3).

**Important:** For production do not rely on account keys. Use Azure AD-based authorization (role assignments) where possible.

## Step 2 — Create an Azure Key Vault

1. Search **Key vaults** → **+ Create**.
2. Select Subscription and the `rg-adf-keyvault-lab` resource group.
3. Name: `kv-adf-lab-<suffix>`
4. Pricing tier: Standard is fine. Leave defaults for soft-delete enabled (recommended).
5. Click **Review + Create** → **Create**.

## Step 3 — Add the storage account key as a secret in Key Vault

1. Open the Key Vault you created → **Secrets** → **+ Generate/Import**.
2. **Upload options:** Manual. **Name:** `storage-account-key` (or similar, use a clear naming convention).
3. **Value:** paste the Storage account key copied in Step 1.
4. Optionally set activation and expiry dates; click **Create**.

## Step 4 — Enable system-assigned managed identity on Azure Data Factory

1. Search **Data factories** and open your Data Factory. If you don't have one, create a new Azure Data Factory instance (`adf-lab-<suffix>`).
2. In the Data Factory blade (not the ADF Studio), select **Identity** under **Settings**.
3. Turn **System assigned** to **On** and click **Save**.
4. After saving, note the **Object (principal) ID** shown — you will use this to grant Key Vault permissions.

## Step 5 — Grant Key Vault permission to the ADF managed identity

Option A — Key Vault Access Policies (classic, simple)

1. In the Key Vault → **Access policies** → **+ Add Access Policy**.
2. Configure: **Secret permissions:** check **Get** and **List** (minimum required).
3. In **Select principal**, search for your Data Factory name (it appears as the managed identity) and add it.
4. Click **Add** then **Save**.

Option B — (Alternative) Use Azure RBAC for Key Vault (recommended long-term)

- Assign the `Key Vault Secrets User` (or `Key Vault Reader`) role to the Data Factory managed identity scoped to the Key Vault resource. This uses Azure RBAC instead of Access Policies.

## Step 6 — Create a Key Vault Linked Service in Azure Data Factory

1. Open Azure Data Factory Studio (select the Data Factory → **Author & Monitor**).
2. In ADF Studio, go to **Manage (gear icon) → Linked services → + New**.
3. Search for **Azure Key Vault** and select **Azure Key Vault (Azure)** linked service type.
4. Fill details:
5. **Name:** `LS_KeyVault_AdfLab`
6. **Authentication method:** **Managed Identity** (System-assigned)
7. **Azure Key Vault URL:** copy from the Key Vault Overview (e.g., `https://kv-adf-lab-xyz.vault.azure.net/`).
8. Click **Test connection** — it should succeed (because ADF has identity and Key Vault allows it).
9. Click **Create**.

## Step 7 — Create the Storage Linked Service that reads key from Key Vault

1. In **Manage → Linked services → + New** search **Azure Blob Storage** (or **Azure Data Lake Storage Gen2** depending on your storage type).
2. Select the storage type you created and click **Continue**.
3. **Name:** `LS_Storage_AdfLab`
4. For **Authentication method** choose **Account key** (we will supply the key by reference — not paste).
5. For **Account selection method** pick **Enter manually** and provide the storage account name or URL.
6. For the **Account key** property click the small **Add dynamic content** link (or the Key icon) in the field.
7. In the dynamic content builder, choose **Add Key Vault reference** (or select the Key Vault linked service you created and then select the secret `storage-account-key`).

If you prefer JSON editing (advanced): the linked service JSON for the account key looks like:

```
"parameters": {},
"type": "AzureBlobStorage",
"typeProperties": {
  "connectionString": "",
  "accountKey": {
    "type": "AzureKeyVaultSecret",
    "store": {
      "referenceName": "LS_KeyVault_AdfLab",
      "type": "LinkedServiceReference"
    },
  },
  "secretName": "storage-account-key"
}
```

1. Click **Test connection** → It should succeed. Save the linked service.

## Step 8 — Create datasets and a simple pipeline to validate

1. **Create source dataset:** e.g., an Azure Blob dataset pointing to a container (use the `LS_Storage_AdfLab` linked service).

2. **Create sink dataset:** e.g., another container in the same storage account or another storage account (for demo you can copy from a folder to another folder in the same account).
  3. Build a simple **Copy Data** pipeline:
  4. Source: pick the source dataset.
  5. Sink: pick the sink dataset.
  6. Publish all changes and **Trigger Now** the pipeline.
  7. Monitor the pipeline run. If the Key Vault reference works and permissions are correct, the copy activity will succeed and you will see activity log entries.
- 

## Troubleshooting tips

- **Key Vault test connection fails:** Verify ADF managed identity is enabled and Key Vault access policy includes `Get` on secrets for that identity.
  - **Linked service test fails for storage:** Confirm you referenced the correct secret name and that the secret value contains the exact account key (no extra whitespace).
  - **Permission errors while accessing storage at runtime:** The account key in Key Vault may be stale if the storage account key was rotated — update the secret with the new key or adopt managed identity + RBAC approach.
- 

## Cleanup (optional)

To avoid charges, delete the resources you created (Resource Group → Delete resource group) or individually delete the Data Factory, Key Vault, and Storage Account.

---

## Security recommendations & alternatives

1. **Prefer Azure AD + RBAC for storage:** Instead of storing keys at all, enable Managed Identity for ADF and assign it the `Storage Blob Data Contributor` role on the storage account. Then create the storage linked service in ADF using **Managed Identity / Azure AD** authentication. This avoids secrets altogether.
  2. **Use Key Vault RBAC:** Prefer Azure RBAC-based access for Key Vault rather than legacy access policies.
  3. **Rotate keys:** If you must use account keys, rotate them regularly and update Key Vault secrets accordingly.
  4. **Audit:** Enable Key Vault logging and monitor access to secrets.
-

## **What we demonstrated (summary)**

- Centralized secret storage using Azure Key Vault.
- How to grant ADF managed identity permission to read Key Vault secrets.
- How to configure an ADF linked service to read the storage account key from Key Vault at runtime.