**Step-by-Step Azure POC Implementation Guide (With Code)**

**1) Azure Networking Basics**

We’ll create a **Resource Group**, **VNet/Subnet**, and **NSG** first.

**Portal Steps:**

1. Go to **Azure Portal → Create a resource → Resource group**.
   * Name: POC-Analytics.
   * Region: Choose one region for all services (e.g., Central India).
2. **Create Virtual Network**:
   * Search: **Virtual Networks → Create**
   * Name: poc-vnet.
   * Address space: 10.0.0.0/16.
   * Add subnet: default, Address prefix: 10.0.1.0/24.
3. **Create NSG (Network Security Group)**:
   * Search: **Network Security Groups → Create**.
   * Name: poc-vm-nsg.
   * Inbound rule: Allow SSH (port 22) and HTTP (80) for your VM/Web app.

**2) Azure Linux VM (via PuTTY)**

**Portal Steps:**

1. **Create VM**:
   * Search: **Virtual Machines → Create**.
   * Name: poc-vm.
   * Image: **Ubuntu LTS**.
   * Size: B1s (free tier).
   * Authentication: **SSH key**.
2. **Networking**:
   * Select poc-vnet and default subnet.
   * Attach poc-vm-nsg.
3. After deployment:
   * Download .pem or .ppk file.
   * Use PuTTYgen to convert to .ppk (if needed).
   * Connect: ssh azureuser@<Public-IP> via PuTTY.

**Code Template (Sample Data Generation):**

bash

CopyEdit

#!/bin/bash

# generate sample log data

mkdir ~/data

for i in {1..10}; do

echo "$(date), user$i, $((RANDOM%100))" >> ~/data/rawdata.csv

done

Upload this to storage later.

**3) Azure Storage Account**

**Portal Steps:**

1. Search: **Storage Accounts → Create**.
   * Name: mypocanalytics
   * Region: same as VM.
2. Inside Storage Account:
   * Create 2 containers: raw and processed.
   * Enable blob settings for storage account
   * A screenshot of a computer

     AI-generated content may be incorrect.

**Upload Data:**

* From VM: Install Azure CLI.

bash

CopyEdit

az login

az storage blob upload --account-name pocstorage<unique> \

--container-name raw --file ~/data/rawdata.csv --name rawdata.csv

**4) Azure SQL Database**

**Portal Steps:**

1. Search: **SQL Database → Create**.
   * Name: pocdb.
   * New server: server-mysql
   * Admin : sheetal
   * Password : Admin123
   * Basic tier.
   * JDBC Connection String :
   * jdbc:sqlserver://server-mysql.database.windows.net:1433;database=pocdb;user=sheetal@server-mysql;password={your\_password\_here};encrypt=true;trustServerCertificate=false;hostNameInCertificate=\*.database.windows.net;loginTimeout=30;
2. Firewall: Allow Azure services and your IP.

**Create Table:** ProcessedData

**Accessing SQL server from linux VM**

sudo apt-get update

sudo apt-get install mssql-tools unixodbc-dev

Add it to your PATH if needed:

echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc

source ~/.bashrc

**Connect Using below command :**

sqlcmd -S server-mysql.database.windows.net -U sheetal -P Admin123 -d pocdb

SELECT name FROM sys.databases;

GO

SELECT \* FROM Products;

GO

**Allow the services and resources to access this server – This is main to connect the sql server**

A screenshot of a computer

AI-generated content may be incorrect.

**6) Azure Data Factory (ADF)**

**Portal Steps:**

1. Search: **Data Factory → Create**.
   * Name: poc-analytics-adf1.
2. **Launch Studio**:
   * **Linked Services**: Connect Blob Storage & Azure SQL DB.
   * **Pipeline**:
     + Activity: **Copy Data** (source: processed container CSV → destination: SQL table).
     + Mapping columns.

**Publish and Trigger manually or schedule.**

**1. Create Azure Data Factory (ADF)**

**🔹 In the Azure Portal:**

1. **Go to Search (top bar) → Type: Data Factory → Click Create**
2. **Fill out the basic details:**
   * **Name: poc-analytics-adf1**
   * **Region: Choose same region as your Blob/SQL if possible**
   * **Version: V2**
3. **Click Review + Create → then Create**

**✅ 2. Launch ADF Studio**

**Once deployed:**

1. **Go to the Data Factory resource**
2. **Click "Launch Studio" (opens a new tab with the UI)**

**✅ 3. Create Linked Services**

**These allow ADF to connect to your data sources.**

**🔹 In ADF Studio:**

* **Go to Manage (⚙️ gear icon on left pane)**
* **Click Linked Services**
* **Click + New**

**➤ For Azure Blob Storage:**

* **Choose Azure Blob Storage**
* **Authentication method:**
  + **Use Account Key, or Managed Identity (if supported)**
* **Test Connection → Create**

**➤ For Azure SQL Database:**

* **Choose Azure SQL Database**
* **Fill in:**
  + **Server name**
  + **Database name (pocdb, for example)**
  + **Authentication (SQL auth or Azure AD)**
* **Test Connection → Create**

**✅ 4. Create Pipeline**

**Back in Author tab (left menu):**

**🔹 Steps:**

1. **Click + → Pipeline → name it (CopyCsvToSql)**
2. **From the Activities pane, drag Copy Data onto the canvas**

**✅ 5. Configure Copy Activity**

**🔹 Source:**

* **Click on Source tab**
* **Choose your Blob Storage linked service**
* **Browse to the processed container**
* **Select the CSV file(s)**
* **Set:**
  + **File format: DelimitedText**
  + **First row as header: True**
  + **Column delimiter: ,**

**🔹 Sink (Destination):**

* **Go to Sink tab**
* **Select your Azure SQL Database linked service**
* **Choose the destination table (create it ahead of time if needed)**

**✅ 6. Mapping Columns**

* **Go to the Mapping tab**
* **If schema is simple, click Import Schemas**
* **Manually map CSV columns to SQL table columns if needed**

**✅ 7. Publish and Trigger**

* **Click Publish All (top bar) → Wait for publish to complete**
* **Then click Add Trigger → Trigger Now (or schedule it)**

**✅ Optional: Create Schedule**

1. **Go to Triggers → + New**
2. **Set the schedule (daily, hourly, etc.)**
3. **Link it to the pipeline**

**✅ Done!**

**Your ADF pipeline now:**

* **Pulls CSV data from Blob Storage's processed container**
* **Maps it to a SQL table in Azure SQL DB**
* **Runs manually or on a schedule**

**7) Azure Function (Automation)**

**Portal Steps:**

1. Search: **Function App → Create**.
   * Name: poc-function.
   * Runtime: Python.
2. Create a **Blob trigger function**:
   * Trigger when new file arrives in raw.
   * Call Databricks Job REST API (or trigger ADF).

**Code Template (Python Blob Trigger):**

python

CopyEdit

import logging

import azure.functions as func

import requests

def main(myblob: func.InputStream):

logging.info(f"New blob: {myblob.name}")

# Trigger Databricks job

response = requests.post("https://<databricks-instance>/api/2.0/jobs/run-now",

headers={"Authorization": "Bearer <token>"},

json={"job\_id": "<your-job-id>"})

**8) Azure Web App (Dashboard)**

**Portal Steps:**

1. Search: **App Service → Create**.
   * Runtime: Python/Flask.
2. Deploy via GitHub or Zip.



