
🚀 Step-by-Step Guide: Deploy Python Function in Azure Portal

🔹 Step 1: Create a Function App

1. Go to [Azure Portal](https://portal.azure.com)
2. Click **"Create a resource"** → Search for **"Function App"**
3. Click **Create** and fill in:
    - **Subscription**: Select yours
    - **Resource Group**: Create or reuse one
    - **Function App name**: Must be globally unique
    - **Region**: Choose nearby
    - **Runtime stack**: Select **Python**
    - **Version**: Choose Python 3.9 or 3.10
    - **Hosting plan**: Select **Consumption (Serverless)**
4. On the **Storage tab**, select or create your storage account (e.g., mikestorage)
5. Click **Review + Create** → then **Create**

🔹 Step 2: Add a Timer Trigger Function

1. Go to your new Function App
2. In the left menu, click **Functions** → **+ Add**
3. Choose:
    - **Development environment**: **Develop in Portal**
    - **Template**: **Timer Trigger**
    - **Name**: GenerateOutpatientData
    - **Schedule**: 0 */5 * * * * (runs every 5 minutes)
4. Click **Add**

🔹 Step 3: Paste the Refactored Python Script

Replace the default code with this:

```
import logging

import os

import random
import datetime

import csv

import io

import azure.functions as func

from azure.storage.blob import BlobServiceClient

import names


# Environment variables

STORAGE_CONNECTION_STRING = os.getenv("AzureWebJobsStorage")

CONTAINER_NAME = os.getenv("BLOB_CONTAINER", "outpatient-data")

BLOB_PREFIX = os.getenv("BLOB_PREFIX", "outpatient_")


# Data pools

CLINICS = [('University Hospital', 'Cardiology'), ('Royal Glamorgan', 'Dermatology')]

PROVIDERS = [('D21', 'Dr. Evans'), ('D45', 'Dr. Jones')]

DIAGNOSES = [('I10', 'Essential hypertension'), ('L20.9', 'Atopic dermatitis')]

STATUSES = ['Attended', 'DNA']


def generate_nhs_number():

    digits = [random.randint(0, 9) for _ in range(9)]

    checksum = 11 - (sum((10 - i) * d for i, d in enumerate(digits)) % 11)

    if checksum == 11: checksum = 0

    if checksum == 10: return generate_nhs_number()

    return ''.join(map(str, digits + [checksum]))


def generate_patient():

    return {

        "PatientID": f"P{random.randint(500, 999)}",

        "NHS_ID": generate_nhs_number(),

        "FirstName": names.get_first_name(),

        "LastName": names.get_last_name(),
        "DOB": f"{random.randint(1940, 2010)}-{random.randint(1,12):02d}-
{random.randint(1,28):02d}",
        "Postcode": f"CF{random.randint(10,40)}
{random.randint(1,9)}{random.choice('ABCDEFGHIJKLMNOPQRSTUVWXYZ')}{random.choice('ABCDE
FGHIJKLMNOPQRSTUVWXYZ')}"
    }


def generate_appointment(patient, appointment_id):
    clinic, specialty = random.choice(CLINICS)
    provider_id, provider_name = random.choice(PROVIDERS)
    code, description = random.choice(DIAGNOSES)
    status = random.choice(STATUSES)
    return [
        f"A{appointment_id}", datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        patient["PatientID"], patient["NHS_ID"], patient["FirstName"], patient["LastName"],
        patient["DOB"], patient["Postcode"], clinic, specialty,
        provider_id, provider_name, code, description, status
    ]


def main(mytimer: func.TimerRequest) -> None:
    logging.info("⏱ Synthetic outpatient data generation started.")
    blob_service = BlobServiceClient.from_connection_string(STORAGE_CONNECTION_STRING)
    container_client = blob_service.get_container_client(CONTAINER_NAME)
    try:
        container_client.create_container()
    except Exception:
        pass  # Container may already exist


    output = io.StringIO()
    writer = csv.writer(output)
    writer.writerow([
        'AppointmentID', 'AppointmentDateTime', 'PatientID', 'NHS_ID', 'PatientFirstName',
        'PatientLastName', 'PatientDOB', 'PatientPostcode', 'ClinicName', 'Specialty',
        'ProviderID', 'ProviderName', 'DiagnosisCode', 'DiagnosisDescription', 'AppointmentStatus'
    ])

    for i in range(10):  # Generate 10 records per run
        patient = generate_patient()
        writer.writerow(generate_appointment(patient, 1000 + i))

    blob_name = f"{BLOB_PREFIX}{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')}.csv"
    container_client.upload_blob(name=blob_name, data=output.getvalue(), overwrite=True)
    logging.info(f"✅ Uploaded synthetic data to blob: {blob_name}")
```

Click **Save**.

🔹 Step 4: Configure Environment Variables

1. Go to **Configuration** → **Application Settings**
2. Add:
   - BLOB_CONTAINER: outpatient-data
   - BLOB_PREFIX: outpatient_
3. Save and restart the Function App

🔹 Step 5: Monitor Output

- Go to your Blob Storage (mikestorage) → Container outpatient-data
- You'll see CSV files uploaded every 5 minutes
- Use **Application Insights** to monitor logs and performance

---

✅ What You've Achieved

Feature	**Result**
Serverless Python Function	Runs on schedule without manual triggers
Blob Storage Integration	Data saved directly to cloud container
Cost Optimization	Uses Consumption Plan (pay-per-execution)

Feature	Result
Easy Configuration	Change container or prefix anytime