# counterfit

# az login

# mkdir azurite

# code . (na pasta da app)

# Tutorial Completo: Configuração e Comandos para Obter Blobs do Azure Storage

## 1. Criar o Grupo de Recursos

1. Criar um grupo de recursos:

**az group create --name gps\_group\_jrsn87 --location uksouth**

## 2. Configurar o IoT Hub

1. Criar o IoT Hub:

**az iot hub create --name gps-hub-jrsn87 --resource-group gps\_group\_jrsn87 --sku F1 --location uksouth –partition-count 2**

2. Registrar um dispositivo no IoT Hub:

**az iot hub device-identity create --device-id gps-device-jrsn87 --hub-name gps-hub-jrsn87**

## 3. Configuração Local do Azure Functions

1. Criar o projeto Azure Functions:

**mkdir gps-functions  
cd gps-functions  
func init --worker-runtime python**

2. Adicionar um Event Hub Trigger:

**func new --name iot-hub-trigger --template "Azure Event Hub trigger"**

3. Atualizar os arquivos de configuração:

No arquivo function.json:

{  
 "type": "eventHubTrigger",  
 "name": "events",  
 "direction": "in",  
 "eventHubName": "",  
 "connection": "IOT\_HUB\_CONNECTION\_STRING",  
 "cardinality": "many",  
 "consumerGroup": "$Default",  
 "dataType": "binary"  
}

No arquivo host.json:

"version": "[2.\*, 3.0.0)"

No arquivo local.settings.json:

{  
 "IsEncrypted": false,  
 "Values": {  
 "FUNCTIONS\_WORKER\_RUNTIME": "python",  
 "AzureWebJobsStorage": "UseDevelopmentStorage=true"  
 }  
}

4. Adicionar dependências necessárias ao requirements.txt:

**azure-functions  
azure-storage-blob**

5. Instalar as dependências:

**pip3 install -r requirements.txt**

## 4. Obter Connection Strings do IoT Hub

1. Obter a Connection String do dispositivo:

**az iot hub device-identity connection-string show --device-id gps-device-jrsn87 --hub-name gps-hub-jrsn87 --output table**

Copie o valor retornado e substitua <connection\_string> na app.py.

2. Obter a Connection String do IoT Hub para o Event Hub compatível:

**az iot hub connection-string show --default-eventhub --output table --hub-name gps-hub-jrsn87**

Copie o valor retornado e adicione ao arquivo local.settings.json no campo IOT\_HUB\_CONNECTION\_STRING.

## 5. Criar e Configurar a Conta de Armazenamento

1. Criar uma conta de armazenamento no Azure:

**az storage account create --resource-group gps\_group\_jrsn87 --sku Standard\_LRS --allow-blob-public-access true --name gpsstoragejrsn87**

2. Obter a Connection String da conta de armazenamento:

**az storage account show-connection-string --output table --name gpsstoragejrsn87**

Copie o valor retornado e adicione ao arquivo local.settings.json no campo STORAGE\_CONNECTION\_STRING.

## 6. Testar Localmente

1. Iniciar o emulador do Azure Storage (na pasta onde foi criado o diretorio azurite):

**azurite --location azurite**

deixar a janela a correr minimizada

2. Executar o projeto localmente:

**func start**

## 7. Implantar o Azure Functions na Nuvem

1. Criar o Function App:

**az functionapp create --resource-group gps\_group\_jrsn87 --runtime python --functions-version 3 --os-type Linux --consumption-plan-location uksouth --storage-account gpsstoragejrsn87 --name gps-functions-app**

2. Configurar as Application Settings:

**az functionapp config appsettings set --resource-group gps\_group\_jrsn87 --name gps-functions-app --settings "IOT\_HUB\_CONNECTION\_STRING=<connection string>" "STORAGE\_CONNECTION\_STRING=<connection string>"**

**EXEMPLO:**

**az functionapp config appsettings set --resource-group gps\_group\_jrsn87 --name gps-functions-app --settings "IOT\_HUB\_CONNECTION\_STRING=Endpoint=sb://ihsuprodlnres007dednamespace.servicebus.windows.net/;SharedAccessKeyName=iothubowner;SharedAccessKey=2zs59vb07ISlXrVrKP26WdZ/yVZ+aSpOAAIoTKnB17I=;EntityPath=iothub-ehub-gps-hub-jr-56280414-f6100da9d6" "STORAGE\_CONNECTION\_STRING=DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=gpsstoragejrsn87;AccountKey=BwDM3ptSe6g8qWCigLb2oz6sl6pPmzz+maS02C/1Q6N8/gmoOJROkqBPUj6SzrOByZOlpu0Gqt4j+AStlhtMSw==;BlobEndpoint=https://gpsstoragejrsn87.blob.core.windows.net/;FileEndpoint=https://gpsstoragejrsn87.file.core.windows.net/;QueueEndpoint=https://gpsstoragejrsn87.queue.core.windows.net/;TableEndpoint=https://gpsstoragejrsn87.table.core.windows.net/"**

3. Publicar o código na nuvem (correr comando no vs code dentro do diretorio da função):

**func azure functionapp publish gps-functions-app**

## 8. Verificar e Obter Blobs do Storage

1. Obter as chaves da conta de armazenamento:

**az storage account keys list --output table --account-name gpsstoragejrsn87**

2. Listar blobs no container gps-data:

**az storage blob list --container-name gps-data --output table --account-name gpsstoragejrsn87 --account-key <key1>**

**EXEMPLO:**

**az storage blob list --container-name gps-data --output table --account-name gpsstoragejrsn87 --account-key BwDM3ptSe6g8qWCigLb2oz6sl6pPmzz+maS02C/1Q6N8/gmoOJROkqBPUj6SzrOByZOlpu0Gqt4j+AStlhtMSw==**

3. Fazer o download de um blob:

**az storage blob download --container-name gps-data --account-name gpsstoragejrsn87 --account-key <key1> --name <blob\_name> --file <file\_name>**

**EXEMPLO:**

**az storage blob download --container-name gps-data \**

**--account-name gpsstoragejrsn87 \**

**--account-key BwDM3ptSe6g8qWCigLb2oz6sl6pPmzz+maS02C/1Q6N8/gmoOJROkqBPUj6SzrOByZOlpu0Gqt4j+AStlhtMSw== \**

**--name gps-device-jrsn87/0a27a718-ad07-11ef-8b63-df644612c108.json \**

**--file test-blob.json**

4. Vizualizar conteúdo do blob:

**cat test-blob.json**

**CASO FOSSE NECESSÁRIO USAR A REGISTRY MANAGER CONNECTION STRING ELA TERIA DE SER SUBSTITUIDA NO FICHEIRO local.settings.json**

**"REGISTRY\_MANAGER\_CONNECTION\_STRING": "<connection string>",**

**O COMANDO PARA A OBTER SERIA:**

**az iot hub connection-string show --policy-name service --output table --hub-name gps-hub-jrsn87**