



API for Dummies: Easy File transferring Automation with Python & Power Automate

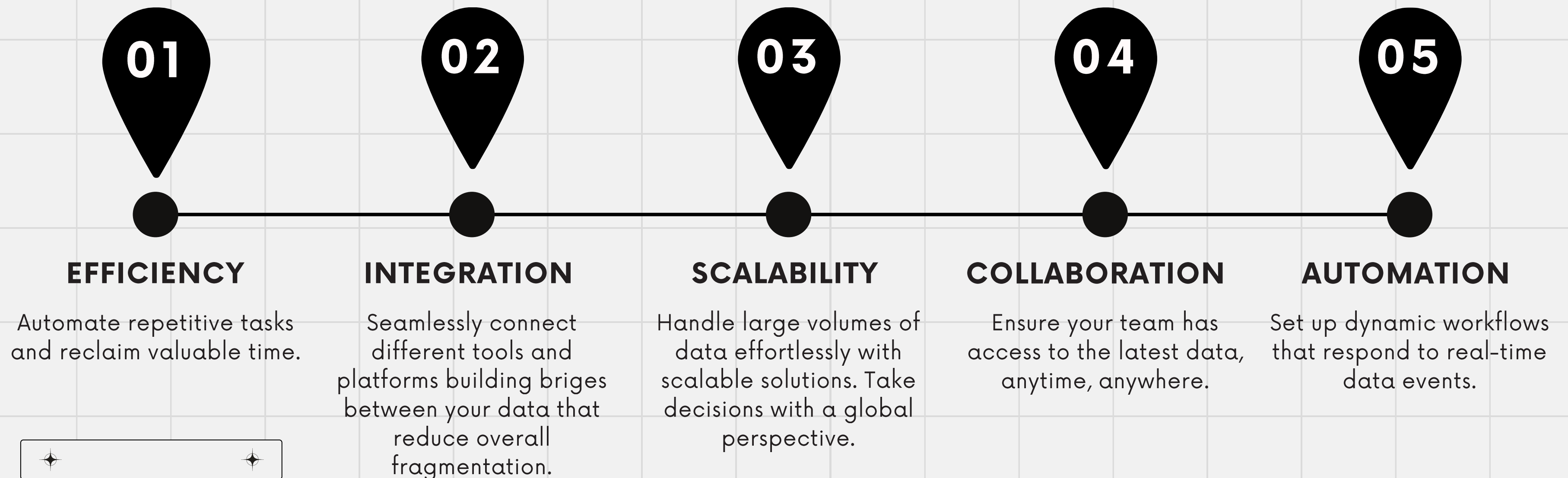
Git Hub repository: <https://github.com/tomasdevelopment/Automation-Flows-with-Python->



Why Automate Your Data Workflows?

Imagine a world where Excel evolves from manual tasks tool to a powerful automated system, enabling seamless data integration and freeing your team to focus on strategic decision-making, and data story telling.

Key Benefits of Automating Data Workflows with Python and Power Automate:



Write Your Python Workload Code

1

```
# Function to save DataFrame to an in-memory Excel file
def save_df_to_buffer(df):
    buffer = BytesIO()
    with pd.ExcelWriter(buffer, engine='xlsxwriter') as writer:
        df.to_excel(writer, index=False)
    buffer.seek(0)
    return buffer

# Function to encode file content to base64
def encode_file_to_base64(buffer):
    file_content = buffer.read()
    return base64.b64encode(file_content).decode('utf-8')
```

2

```
4 # Create the JSON payload with base64-encoded Excel and file name
5 json_payload = json.dumps({
6     "file_content": encoded_excel,
7     "file_name": file_name
8 })
9 print("JSON Payload:")
10 print(json_payload)
11
12 # Define the URL for the automate flow
13 url = 'https://prod-142.westus.logic.azure.com/workflows/c2fe2123/triggers/manual/paths/invoke/{JSON}?api-version=2016-06-01'
14
15 # Headers for the POST request
16 headers = {
17     'Content-Type': 'application/json'
18 }
19
20 # Make the POST request
21 response = requests.post(url, headers=headers, data=json_payload)
22
23 print(response.status_code)
```

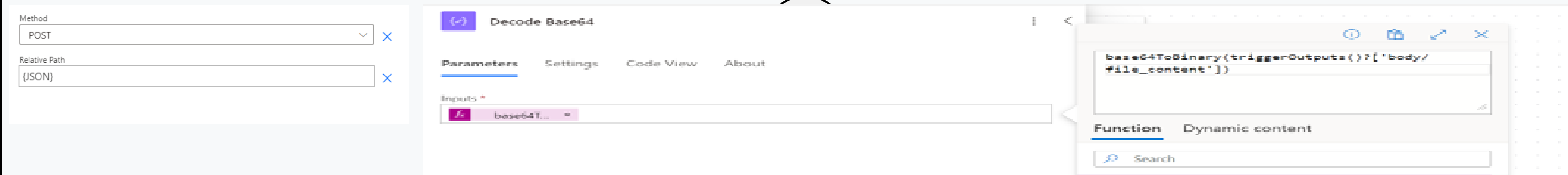
Start by saving the data frame you want to save on SharePoint, One Drive Business, other storage service to a buffer and encoding it to base 64. (More efficient than downloading the excel file on your hardisk)

Using requests and Json library write the script that creates your Payload and send a Post response to the flow URL.

Create your Power Automate Flow

3

Create your Power Automate Flow with an HTTP trigger that retrieves data with a POST method to its relative json path, then add a Compose Action



This compose action will decode the file you just encoded from Python @{base64ToBinary(triggerOutputs()?['body/file_content'])}

4

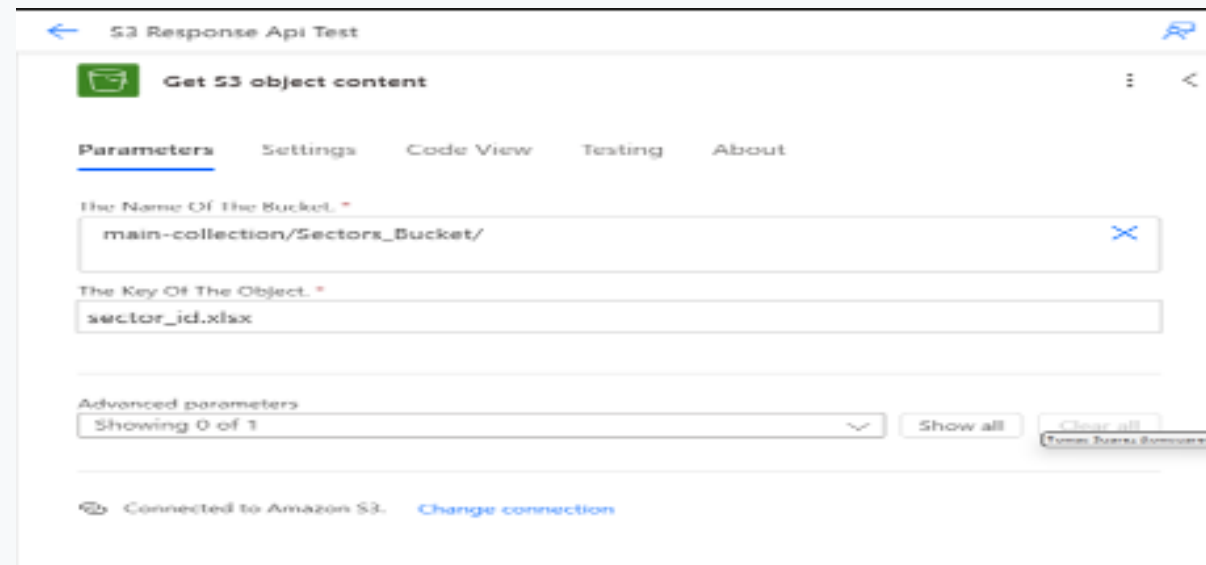
Add a Create File in Sharepoint Action



Add the file name from the HTTP Trigger (@{triggerBody()?['file_name']}) and use the outputs of the compose action as a file content.

Retrieve the data from Amazon S3 Bucket

5

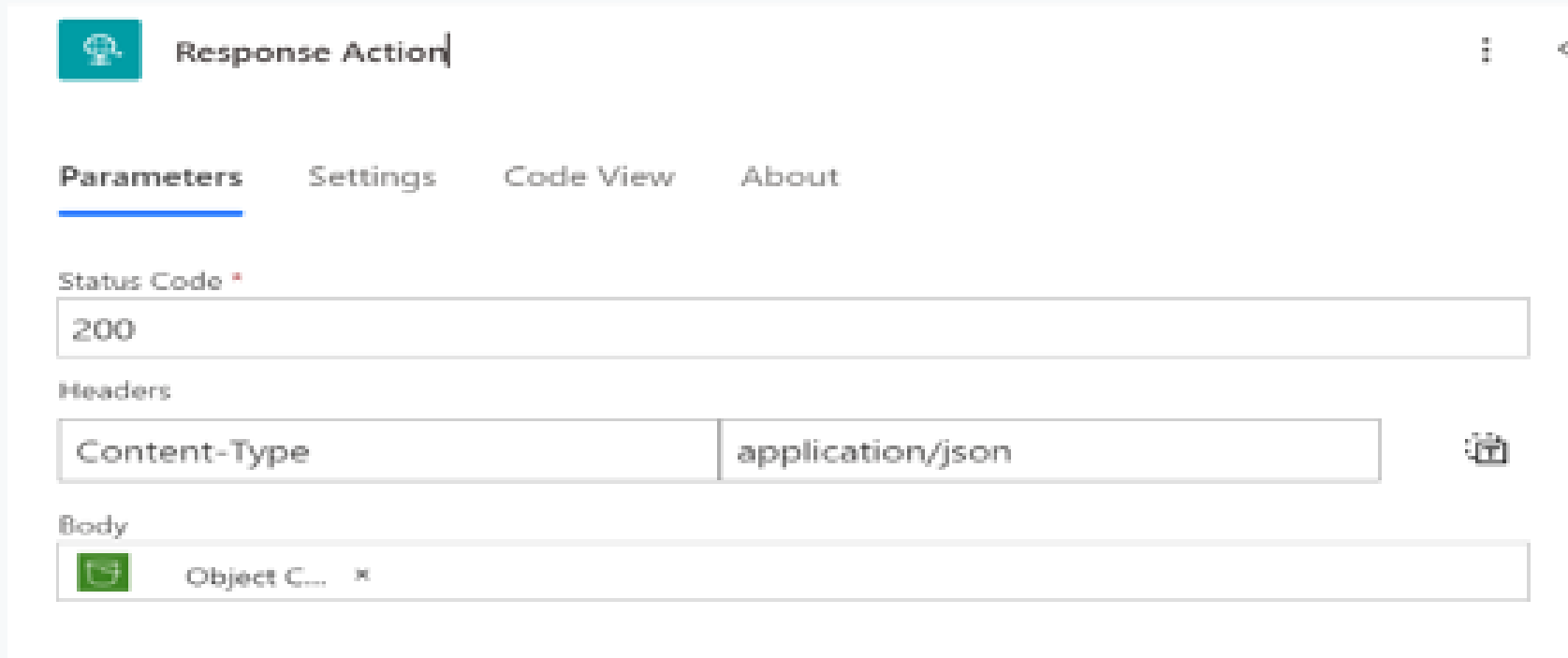


The screenshot shows the 'Get S3 object content' configuration screen. It has tabs for 'Parameters', 'Settings', 'Code View', 'Testing', and 'About'. Under 'Parameters', there are two input fields: 'The Name Of The Bucket' with the value 'main-collection/Sectors_Bucket/' and 'The Key Of The Object' with the value 'sector_id.xlsx'. Below these is an 'Advanced parameters' section with a dropdown set to 'Showing 0 of 1' and buttons for 'Show all' and 'Clear all'. At the bottom, it says 'Connected to Amazon S3' with a 'Change connection' link.

Create Connection to S3 and Retrieve S3 Object Content

Use the name of your bucket and the key (path) of the object in your bucket.

6



The screenshot shows the 'Response Action' configuration screen. It has tabs for 'Parameters', 'Settings', 'Code View', and 'About'. Under 'Parameters', there are three sections: 'Status Code' with the value '200', 'Headers' with a table containing 'Content-Type' and 'application/json', and 'Body' with a dropdown set to 'Object C...'. There is a trash icon next to the 'Content-Type' header.

Add the HTTP Response Action to Finish the Flow

Status 200 means the response was successful, and add the application/json content type with the body from the object content we just retrieved from Amazon S3 bucket.

**RESEARCH PRESENTATION
END**