

Creating the *GitHub Commit and 2FA Data Fetch* Automation Using Blink

Overview

To automate is to take work that can be done by humans in a slow and time consuming fashion, and give it to a machine to pull, sort and translate the data. The machine can even answer questions regarding the data by the use of “print” statements. In this automation we will take a repository, understand who started the repository, recall the repository, and answer questions regarding the repository.

This Automation will do the following:

- Tell you the name of the last commit into a specific repository
- Indicate if the current GitHub user is using 2 factor authentication
- Fetch data from a JSON file, parse the data and tell you who authored the longest post and how many words were written.

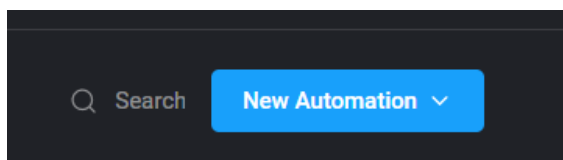
Accounts Needed

The accounts needed for this specific automation include a verified Blink account and a GitHub account. After both accounts are obtained and completely set up, we will start by opening the Blink account.

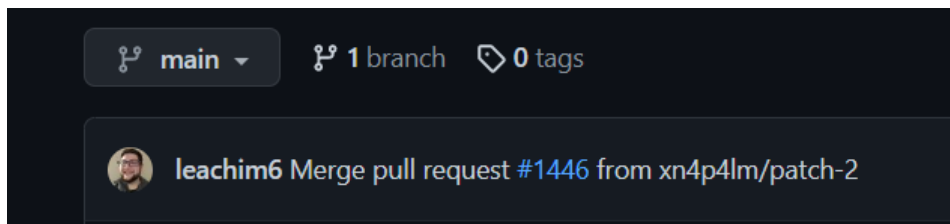
Getting Started

The following steps should be taken to create this automation:

1. After logging into your Blink account on the right side of the screen click **New Automation** and select **From scratch**.



2. On the Create New Automation screen, enter **Automation Name** and select a **type of trigger**. Click **Create Automation**. You are redirected to the **Edit page** of the Automation.
3. From the given repository, find the owner's name, make note of this for step 5.



Create Step 1: List Commits

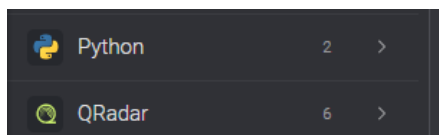
4. On the left hand side of the screen is the **Action Panel**, find the **GitHub pack** and drag the **List Commit** step onto the canvas.



5. Enter the **owner's name** and preference of pages (at the bottom). For this example we chose 1.

Create Step 2: Print Latest Commit User

6. In the **Action Panel**, find the **Python pack** and drag the **Run Python** step onto the canvas.

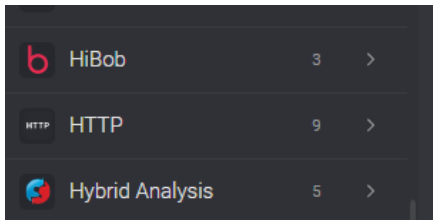


7. Enter the following **code**:

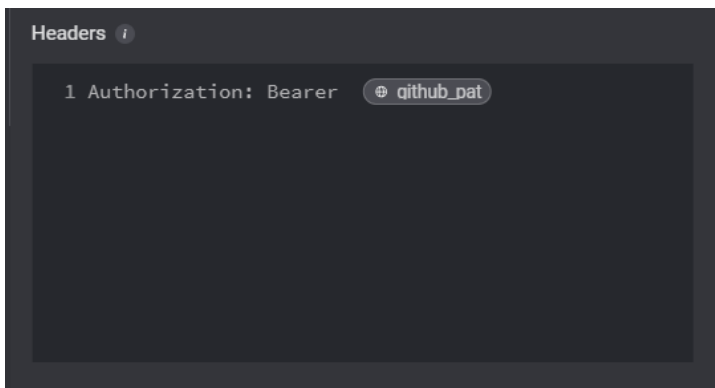
```
PARAMETERS Action info  
  
Code * i  
  
1 commits = context.steps.1.output  
2 if len(commits) == 0:  
3     raise ValueError("No commits found")  
4  
5 first_commit = commits[0]  
6 print(first_commit.get("commit", {}).get("author",  
    {}).get("name"))
```

Create Step 3: Fetch GitHub Authenticated User

8. For this step we must obtain a token from GitHub, log into your GitHub account and a brief Google search will explain how to obtain the token, in short:
 - Click on your profile photo
 - Left sidebar click **Developer Tools**
 - Left sidebar click **Personal Access Tokens**
 - Left sidebar click **new token**
9. In the **Action Panel**, find the **HTTP pack** and drag the **Send HTTP POST request** step onto the canvas.

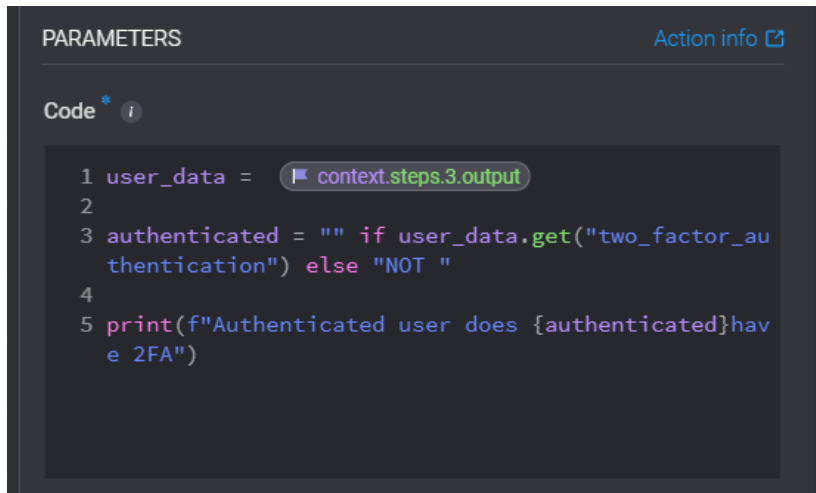


10. Put the following **URL** for the GitHub user API : <https://api.github.com/user>
11. In the **Headers** window put the following text:



Create Step 4: Check User 2FA

12. In the **Action Panel**, find the **Python pack** and drag the **Run Python** step onto the canvas.
13. Enter the following **code** to determine if the current user is using 2 factor authentication:



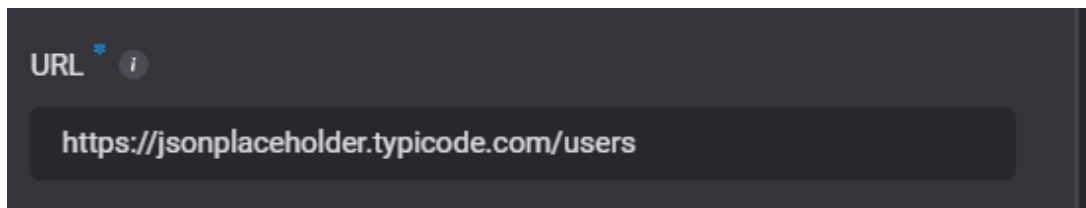
```
PARAMETERS Action info

Code ⓘ

1 user_data = context.steps.3.output
2
3 authenticated = "" if user_data.get("two_factor_authentication") else "NOT "
4
5 print(f"Authenticated user does {authenticated} have 2FA")
```

Create Step 5: Fetch Users

14. In the **Action Panel**, find the **HTTP pack** and drag the **Send HTTP POST request** step onto the canvas and enter the following URL:

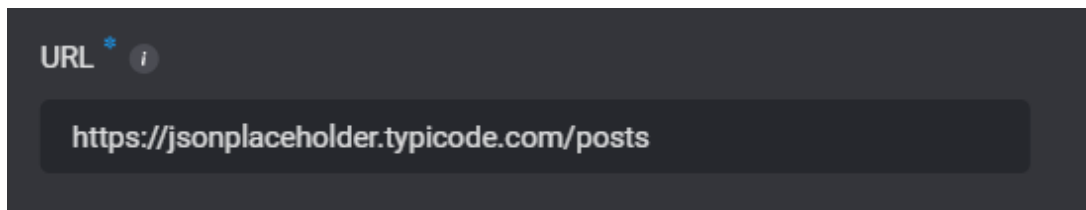


```
URL ⓘ

https://jsonplaceholder.typicode.com/users
```

Create Step 6: Fetch Posts

15. In the **Action Panel**, find the **HTTP pack** and drag the **Send HTTP GET request** step onto the canvas and enter the following URL:



```
URL ⓘ

https://jsonplaceholder.typicode.com/posts
```

Create Step 7: Output Author with Highest Word Count

16. In the **Action Panel**, find the **Python pack** and drag the **Run Python** step onto the canvas.

17. Enter the following **code** to sort the usernames and posts in numeric order, highest word count to least and to determine who authored the longest post and how many words:

Code * i

```

1 users = {
2     user["id"]: user["name"]
3     for user in context.steps.5.output
4 }
5 posts = context.steps.6.output
6
7 counts = {}
8
9 def get_word_count(body):
10     text = body.replace(r"\n", " ")
11     return len(text.split(" "))
12

```

PARAMETERS [Action info](#)

Code * i

```

13 for post in posts:
14     user_id = post["userId"]
15     if user_id not in counts:
16         counts[user_id] = 0
17     counts[user_id] += get_word_count(post["body"])
18
19 sorted_counts = sorted(counts.items(), key=lambda
20     x: x[1])
21 top_user_id, words = sorted_counts[-1]
22 top_user_name = users[top_user_id]
23
24 print(

```

```

23
24 print(
25     f"The top user is {top_user_name} "
26     f"with {words} words written."
27 )

```

18. Click **Test Run** to run the Automation. If you receive an error, review the steps to ensure you have entered all the code and inputs in accordance with this guide.