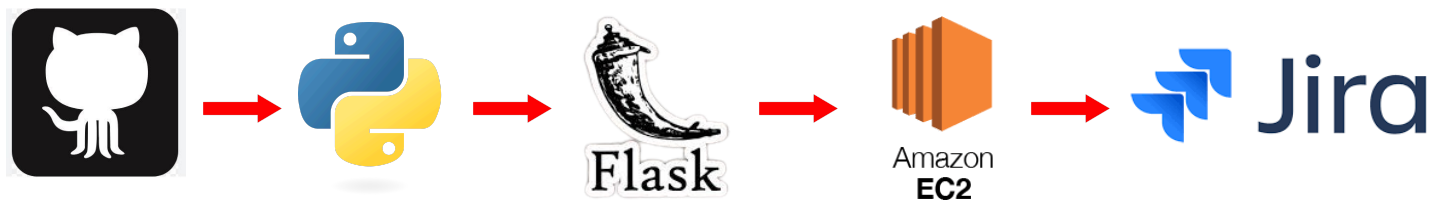


Github automation with creating JIRA tickets (Hosting Flask APP in EC2 Instance and Python Scripting).



STAGE 01 : Lists Projects in JIRA

- Creating test JIRA Account and team managed SCRUM project

Name	Key	Type	Lead	Project URL	More actions
Python-ec2-automation	UH	Team-managed software	Udantha Herath		...

- In the profile settings > Go to Security tab > Create API token

API Tokens

[Create API token](#)
[Revoke all API tokens](#)

Your API tokens need to be treated as securely as any other password. You can only create a maximum of 25 tokens at a time.

New tokens may take up to a minute to work after they've been created.

Token name	Created	Last accessed	Action
python	Dec 17, 2024	30 minutes ago	Revoke

- Go to JIRA API official documentation and get the code for List projects. Then customize the code as your own.

REST API v3

- GET** Get all projects
- POST** Create project
- GET** Get recent projects
- GET** Get projects paginated
- GET** Get project
- PUT** Update project
- DEL** Delete project
- POST** Archive project
- POST** Delete project asynchronously
- POST** Restore deleted or archived project
- GET** Get all statuses for

GET **Get all projects** **DEPRECATED**

Returns all projects visible to the user. Depreciated, use [Get projects paginated](#) that supports search and pagination.

This operation can be accessed anonymously.

Permissions required: Projects are returned only where the user has [Browse Projects](#) or [Administer projects](#) [project permission](#) for the project.

Data Security Policy: Exempt from app access rules

Scopes

OAuth 2.0 scopes required:

Classic **RECOMMENDED:** read:jira-work

Granular: read:issue-type:jira, read:project:jira, read:project.property:jira, read:user:jira, read:application-role:jira ... [\(Show more\)](#)

Connect app scope required: READ

GET /rest/api/3/project

- Forge
- curl
- Node.js
- Java
- Python**
- PHP

```

1 # This code sample uses the 'requests' library:
2 # http://docs.python-requests.org
3 import requests
4 from requests.auth import HTTPBasicAuth
5 import json
6
7 url = "https://your-domain.atlassian.net/rest/api/3/p
8
9 auth = HTTPBasicAuth("email@example.com", "<api_token
10
11 headers = {
12     "Accept": "application/json"
13 }
14
15 response = requests.request(
16     "GET",
17     url,
18     headers=headers,
19     auth=auth

```

This will list all the projects that you have created.

- 1st replace the following on the example code.
 - 1“Email” that you have give to open up the JIRA account
 - Modify the “URL” which relates to JIRA project account
Eg : <https://udanthaherath.atlassian.net>
 - Modify the “API Token”

Full Customized code

```

# This code sample uses the 'requests' library:
# http://docs.python-requests.org
import requests
from requests.auth import HTTPBasicAuth
import json

url = "https://udanthaherath.atlassian.net/rest/api/3/project"

API_TOKEN = "####"

auth = HTTPBasicAuth("udanthaherath@gmail.com", API_TOKEN)

headers = {
    "Accept": "application/json"
}

response = requests.request(
    "GET",
    url,
    headers=headers,
    auth=auth
)

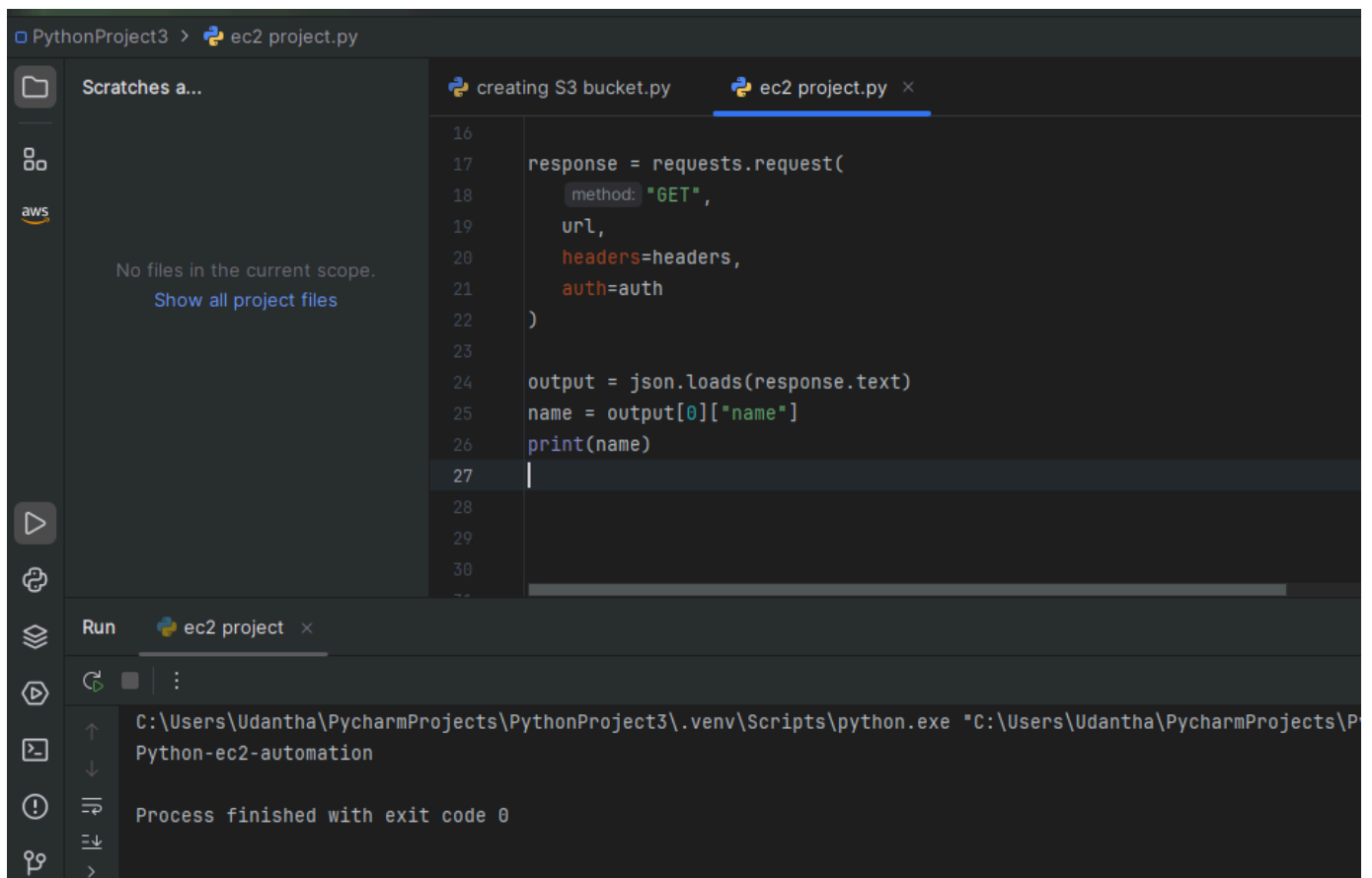
```

```
# print(json.dumps(json.loads(response.text), sort_keys=True, indent=4,
separators=(",", ": ")))
output = json.loads(response.text)
name = output[0]["name"]
print(name)
```

Always try to keep passwords/API Tokens as environment variables for security purposes and to follow the best practices.

- As a result, A .json output will be received.
- Then we have to convert this whole result to a list with dictionaries to sort out a specific result that is needed.
- For that, Use this specific code block to convert the json result at the end of the code

```
Output = json.loads(response.text)
Name = response[0]["name"]
print(Name)
```



STAGE 02 : Raise an Issue completely by a python script

- Go to JIRA API official documentation again. Search for “Issues” and click on “Create Issue”. Then copy down the python code for customization.

The screenshot shows the JIRA REST API documentation for the 'Create issue' endpoint. On the left, a sidebar lists various API endpoints under 'Issues', with 'Create issue' (POST) selected. The main content area explains the endpoint's purpose and provides details on required fields like `issueType` and `parent`. It also mentions permissions and data security policies. On the right, a code editor shows a Python script using the `requests` library to make a POST request to the JIRA API, including headers for `Accept` and `Content-Type`, and a JSON payload for the issue fields.

- Again replace the following on the example code.
 - “Email” that you have give to open up the JIRA account
 - Modify the “URL” which relates to JIRA project account
Eg : `https://udanthaherath.atlassian.net`
 - Modify the “API Token”

Full Customized code

```
# This code sample uses the 'requests' library:
# http://docs.python-requests.org
import requests
from requests.auth import HTTPBasicAuth
import json

url = "https://udanthaherath.atlassian.net/rest/api/3/issue"

API_TOKEN =
"ATATT3xFfGF0JOCsIKF0TPduFid1msw_jBLW6P3JeWF-OO5uG1SBffFeQXVnGIjHqtyvH711YXtaasLV89bT
AM3vqyN74qkQomDl5MYNks2uoUFyg5QRUcwXsoG7TsXpnlesIa-pvXw_wYjqneS1UgTfoCLQdvLuAwX7kbDBr
ggS0qDUOlXiluY=3F66DDAF"

auth = HTTPBasicAuth("udanthaherath@gmail.com", API_TOKEN)

headers = {
    "Accept": "application/json",
    "Content-Type": "application/json"
}

payload = json.dumps( {
    "fields": {
        "description": {
            "content": [
                {
                    "content": [
```

```

        {
            "text": "Hello! This is My 1st ticket.",
            "type": "text"
        }
    ],
    "type": "paragraph"
}
],
"type": "doc",
"version": 1
},
"issuetype": {
    "id": "10008"
},
"project": {
    "key": "UH"
},
"summary": "My 1st Ticket",
},
"update": {}
} )
response = requests.request(
    "POST",
    url,
    data=payload,
    headers=headers,
    auth=auth
)

print(json.dumps(json.loads(response.text), sort_keys=True, indent=4,
separators=(",", ": ")))

```

- Then remove the unnecessary fields from the code as follows. **Sp : You have to be familiar with UI before API Configuration.**
- When the programme runs, the output is as follows.

The screenshot shows a PyCharm IDE with a project named 'PythonProject3'. The file explorer on the left shows a folder 'Project' and a file 'Create JIRA Issue.py'. The main editor displays the code for 'Create JIRA Issue.py', which uses the 'requests' library to make a POST request to a JIRA API endpoint. The code is as follows:

```

55
56 response = requests.request(
57     method="POST",
58     url,
59     data=payload,
60     headers=headers,
61     auth=auth
62 )
63
64 print(json.dumps(json.loads(response.text), sort_keys=True, indent=4, separators=(",", "

```

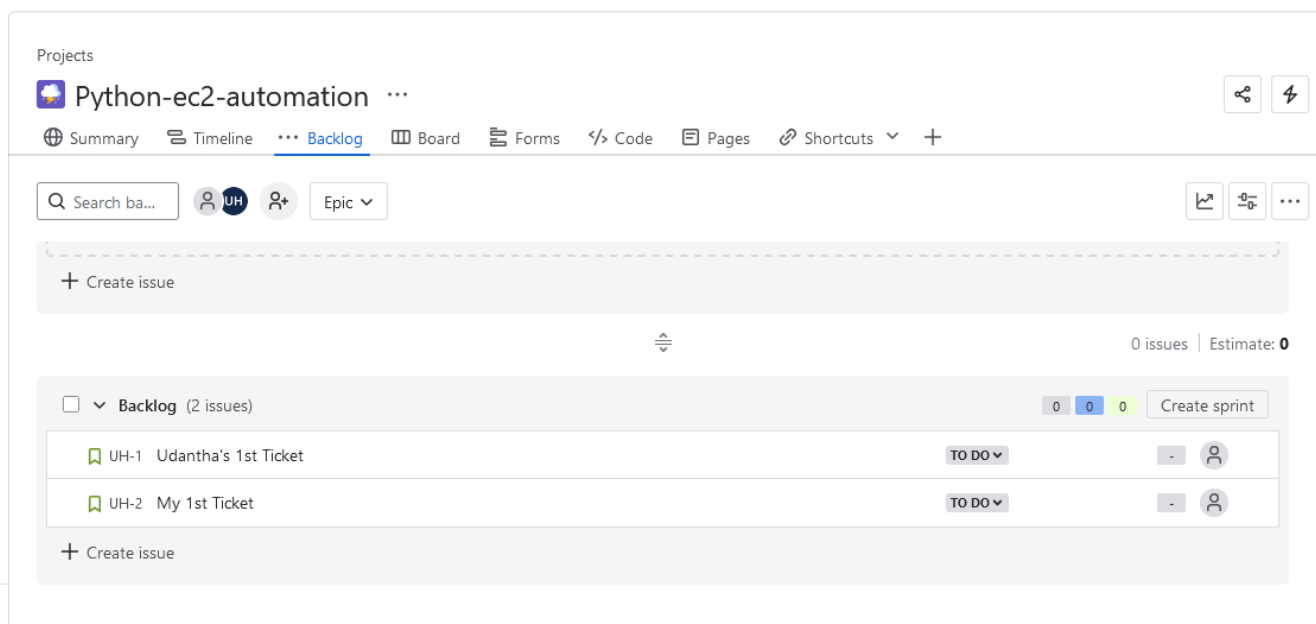
The Run console at the bottom shows the output of the script, which is a JSON object representing a JIRA issue:

```

C:\Users\Udantha\PycharmProjects\PythonProject3\.venv\Scripts\python.exe "C:\Users\Udantha\PycharmProjects\PythonProject3\0
{
  "id": "10000",
  "key": "UH-1",
  "self": "https://udanthaerath.atlassian.net/rest/api/3/issue/10000"
}

```

Final results on the JIRA UI end.



STAGE 03 : API configuration with EC2 Flask app and Github

This Stage is the final stage and consists with several steps

1. Above written python code should be converted to a Flask API
2. Establish a webhook on GITHUB

Udantha Herath (BEng (Hons), MBCS, AWS-ASA, RHCSA, MCITP, MCTS, MCP)

3. Run a Flask server on AWS EC2 instance
4. Raise comments on Issues that made by DEV teams

1. Complete code of the Flask API

```
from flask import Flask
import requests
from requests.auth import HTTPBasicAuth
import json

#creating a Flask app instance
app = Flask(__name__)

@app.route("/createJIRAticket", methods = ['POST'])
def createJIRAticket():

    url = "https://udanthaherath.atlassian.net/rest/api/3/issue"

    API_TOKEN = "###"

    auth = HTTPBasicAuth("udanthaherath@gmail.com", API_TOKEN)

    headers = {
        "Accept": "application/json",
        "Content-Type": "application/json"
    }
    payload = json.dumps({
        "fields": {
            "description": {
                "content": [
                    {
                        "content": [
                            {
                                "text": "Hello! This is My 1st ticket.",
                                "type": "text"
                            }
                        ],
                        "type": "paragraph"
                    }
                ],
                "type": "doc",
                "version": 1
            },
            "issuetype": {
                "id": "10008"
            },
            "project": {
                "key": "UH"
            },
            "summary": "My 1st Ticket",
        },
        "update": {}
    })
```

```

}))
response = requests.request(
    "POST",
    url,
    data=payload,
    headers=headers,
    auth=auth
)

return json.dumps(json.loads(response.text), sort_keys=True, indent=4,
separators=(",", ": "))

app.run('0.0.0.0')

```

2. Establish a webhook on GITHUB

- Go to your GITHUB relevant repository
- Settings > Webhooks
- Create URL with your public IP of the EC2 instance:port/YourAppName
 - Eg : http://124.43.9.568:5000/YourAppName
- Then save and run the script in EC2 instance manually to confirm the webhook will successful authenticate with EC2 API

Webhooks

[Add webhook](#)

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).


✓ <http://47.128.220.150:5000/createJIR...> (issue_comment)

Last delivery was successful.

[Edit](#) [Delete](#)

3. Run a Flask server on AWS EC2 instance

- Install Python on Ec2 instance
- Create .py file and run the scripts while pasting above script
- Verify the webhook works fine.

✓  255e5fc0-bd2b-11ef-84f7-4cbe2afe8c2a ping

2024-12-18 16:00:50 ...

4. Raise comments on Issues that made by DEV teams

Label issues and pull requests for new contributors [Dismiss](#)

Now, GitHub will help potential first-time contributors [discover issues](#) labeled with [good first issue](#)

Filters Labels 9 Milestones 0 [New issue](#)

<input type="checkbox"/>	3 Open	0 Closed	Author	Label	Projects	Milestones	Assignee	Sort
<input type="checkbox"/>	test3		#3 opened 17 hours ago by Cooluda					4
<input type="checkbox"/>	test2		#2 opened 17 hours ago by Cooluda					2
<input type="checkbox"/>	I need to debug whole code		#1 opened 17 hours ago by Cooluda					1

I need to debug whole code #1

[Open](#) Cooluda opened this issue 17 hours ago · 1 comment [Edit](#) [New issue](#)

Cooluda commented 17 hours ago Owner ...

I need to debug whole code

Cooluda commented 17 hours ago Owner Author ...

Start the compiler

[Add a comment](#)

Write H B I

Add your request here

Assignees ⚙️

No one—[assign yourself](#)

Labels ⚙️

None yet

Projects ⚙️

None yet

Milestone ⚙️

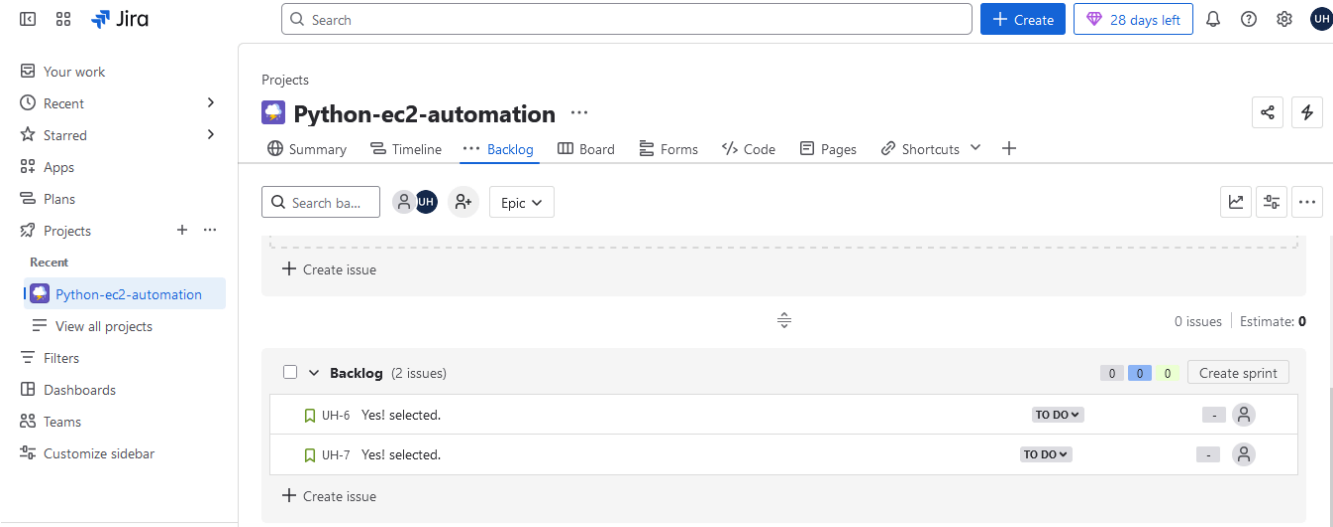
No milestone

Development ⚙️

[Create a branch](#) for this issue or link a pull request.

Notifications Customize

- Now you can notice your comments on issues will automatically raise on JIRA tickets.



THANK YOU.