

Go to github.
Settings.
Developer settings.
Oauth apps.
Create new app.
Give any name url etc.
Click register.

Register a new OAuth application

Application name *

CypressTesting

Something users will recognize and trust.

Homepage URL *

https://www.pavantestingtools.com/

The full URL to your application homepage.

Application description

Application description is optional

This is displayed to all users of your application.

Authorization callback URL *

https://www.pavantestingtools.com/

Your application's callback URL. Read our [OAuth documentation](#) for more information.

☐ Enable Device Flow

Allow this OAuth App to authorize users via the Device Flow.

Read the [Device Flow documentation](#) for more information.

Register application

Cancel

Client id generated. Click generate client secret.

The screenshot shows the GitHub OAuth application settings page for an application named 'CypressTesting'. The page is divided into a sidebar with 'General', 'Optional features', and 'Advanced' tabs, and a main content area. The 'General' tab is selected. The main content area shows the application owner 'pavanoltraining', a 'Transfer ownership' button, and a 'List this application in the Marketplace' button. Below this, it shows '0 users' and a 'Revoke all user tokens' button. The 'Client ID' is displayed as 'ded8c34b1cbcdcaf7149'. The 'Client secrets' section shows a 'Generate a new client secret' button. The page also includes a breadcrumb trail 'Settings / Developer settings / CypressTesting' and a top navigation bar with links like 'Marks', 'Profiles', 'Tab', 'Window', and 'Help'.

Settings / Developer settings / CypressTesting

CypressTesting

pavanoltraining owns this application. [Transfer ownership](#)

You can list your application in the [GitHub Marketplace](#) so that other users can discover it. [List this application in the Marketplace](#)

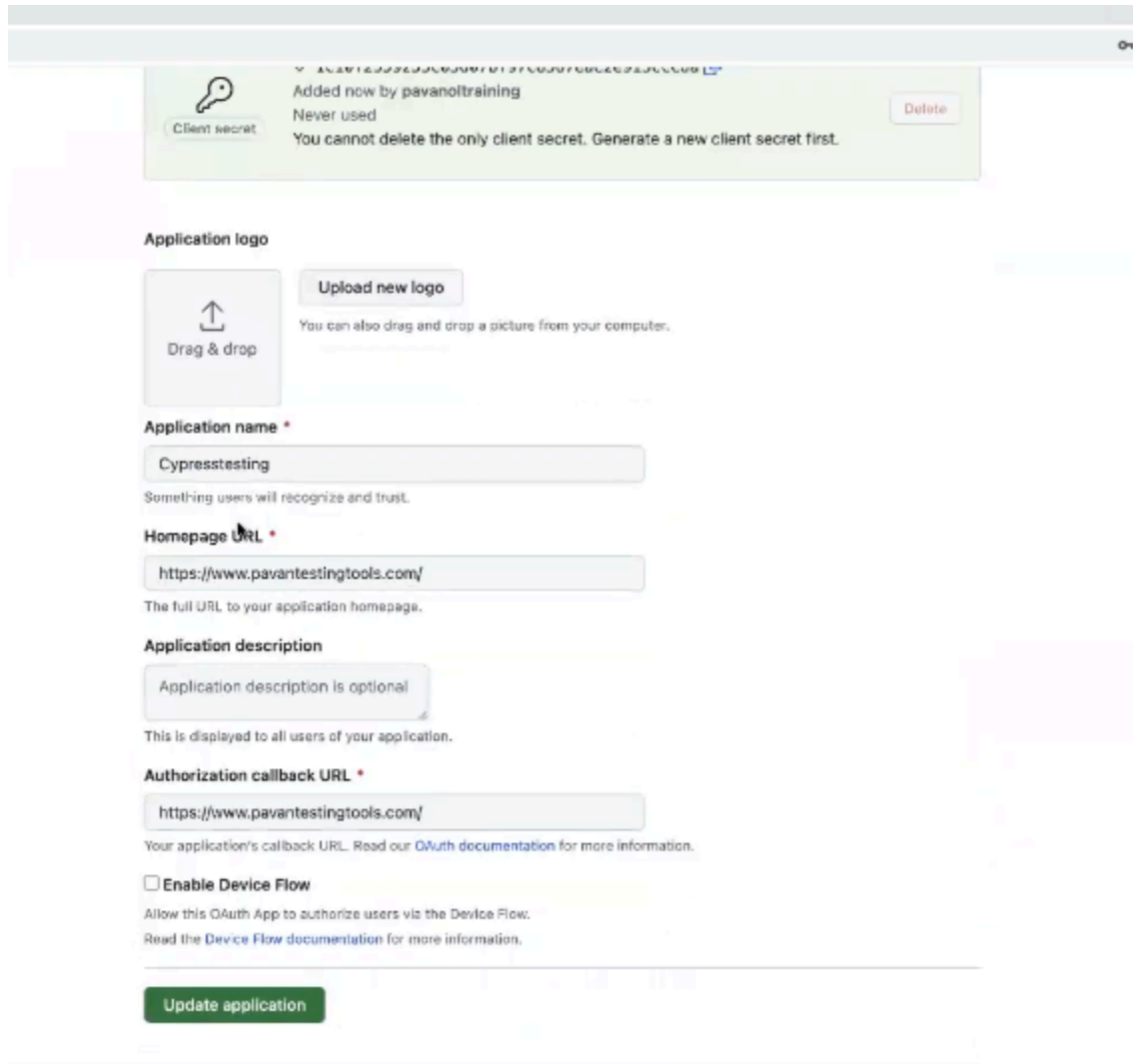
0 users [Revoke all user tokens](#)

Client ID
`ded8c34b1cbcdcaf7149`

Client secrets [Generate a new client secret](#)

You need a client secret to authenticate as the application to the API.

Enter github credentials.
Copy and keep it.
Click update application.



The screenshot shows the GitHub OAuth application configuration interface. At the top, there is a 'Client secret' section with a key icon, indicating it was added by 'pavanoltraining' and has never been used. A warning message states: 'You cannot delete the only client secret. Generate a new client secret first.' Below this is the 'Application logo' section, which includes an 'Upload new logo' button and a 'Drag & drop' area. The 'Application name' field is set to 'Cypresstesting'. The 'Homepage URL' field is set to 'https://www.pavantestingtools.com/'. The 'Application description' field is optional and currently empty. The 'Authorization callback URL' field is also set to 'https://www.pavantestingtools.com/'. There is an unchecked checkbox for 'Enable Device Flow'. At the bottom, there is a green 'Update application' button.

Client secret

Added now by pavanoltraining
Never used
You cannot delete the only client secret. Generate a new client secret first.

Delete

Application logo

Upload new logo

Drag & drop

You can also drag and drop a picture from your computer.

Application name *

Cypresstesting

Something users will recognize and trust.

Homepage URL *

https://www.pavantestingtools.com/

The full URL to your application homepage.

Application description

Application description is optional

This is displayed to all users of your application.

Authorization callback URL *

https://www.pavantestingtools.com/

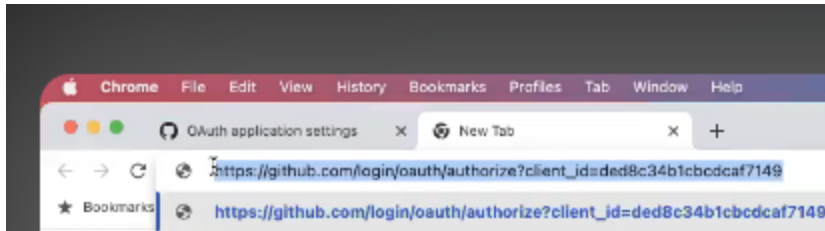
Your application's callback URL. Read our [OAuth documentation](#) for more information.

☐ Enable Device Flow

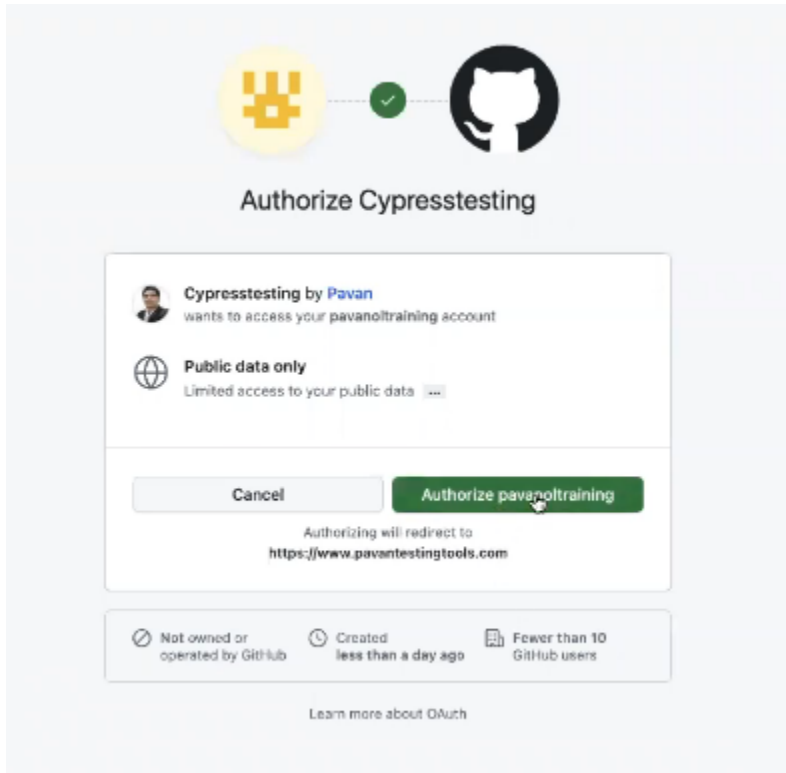
Allow this OAuth App to authorize users via the Device Flow.
Read the [Device Flow documentation](#) for more information.

Update application

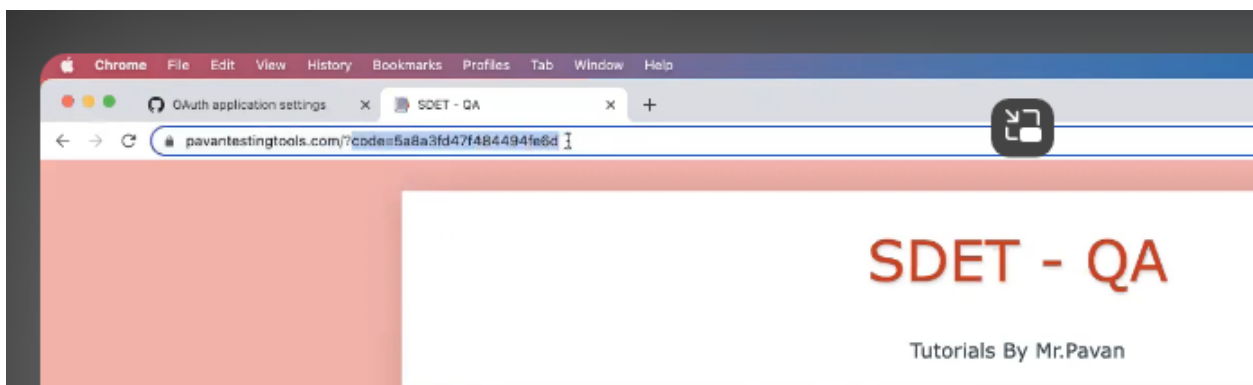
To get the auth code-
Go to the url provided by github.
It has client id.



Click authorise.



Now see the redirected url.
It contains the auth code.



Auth code newly created every time.

Now we want oauth 2 token.

```
1 //Pre-requisite: generate Auth code
2 //https://github.com/login/oauth/authorize/{client_id}
3 // Ex:  https://github.com/login/oauth/authorize?client_id=ded8c34b1cbcdcaf7149
4
5 /* 1) Get the OAuth2 access token
6 POST:  https://github.com/login/oauth/access_token
7 Query params
8      ----
9      client_id
10     client_secret
11     code
12
13 2) Send GET request by using access token.
14 https://api.github.com/user/repos
15 Auth: accessToken
16
17 */
```

```

1 describe("OAuth2",()=>{
2     let accessToken="";
3
4     it("Get OAuth2 Access Token",() => {
5         cy.request({
6             method: 'POST',
7             url: 'https://github.com/login/oauth/access_token',
8             qs: {
9                 client_id: 'ded8c34b1cbcdcaf7149',
10                client_secret: '1c10f2559255c03d07bf97c8567eae2e913ccda',
11                code: '8d3e8caac86fad208f2c'
12            }
13        })
14        .then((response) => {
15            //access_token=
16            //gho_DBRsZwOs5SCQJCVu342nNz1ITKQDj03kvnoU&scope=&token_type=bearer
17            expect(response.status).to.eq(200);
18            //get only the access token using split.
19            //we are breaking the above token nicely.
20            const params=response.body.split('&');
21            accessToken=params[0].split("=")[1];
22            cy.log("Generated token is:"+accessToken);
23
24        })
25    })
26
27
28    it("OAuth2.0 Demo",() => {
29        cy.request({
30            method: 'GET',
31            url: 'https://api.github.com/user/repos',
32            headers: {
33                Authorization:'Bearer '+accessToken
34            }
35        })
36        .then((response) => {
37
38            expect(response.status).to.eq(200);
39            expect(response.body[0].id).to.equal(201070920);
40
41        })
42    })
43
44 })

```

```

1
2 //Pre-requisite: generate Auth code
3 //https://github.com/login/oauth/authorize/{client_id}
4 // Ex:  https://github.com/login/oauth/authorize?client_id=ded8c34b1cbcdcaf7149
5
6 /* 1) Get the OAuth2 access token
7 POST:  https://github.com/login/oauth/access_token
8 Query params
9      ----
10      client_id
11      client_secret
12      code
13
14 2) Send GET request by using access token.
15 https://api.github.com/user/repos
16 Auth: accessToken
17
18 */
19
20 describe("OAuth2", () => {
21     let accessToken="";
22
23     it("Get OAuth2 Access Token", () => {
24         cy.request({
25             method: 'POST',
26             url: 'https://github.com/login/oauth/access_token',
27             qs: {
28                 client_id: 'ded8c34b1cbcdcaf7149',
29                 client_secret: '1c10f2559255c03d07bf97c8567eae2e913ccda',
30                 code: '8d3e8caac86fad208f2c'
31             }
32         })
33         .then((response) => {
34             //accessToken=
35             //gho_DBRsZw0s5SQJCvU342nNz1ITKQDj03kvnoU&scope=&token_type=bearer
36             expect(response.status).to.eq(200);
37             //get only the access token using split.
38             //we are breaking the above token nicely.
39             const params=response.body.split('&');
40             accessToken=params[0].split("=")[1];
41             cy.log("Generated token is:"+accessToken);
42
43         })
44     })
45
46
47     it("OAuth2.0 Demo", () => {
48         cy.request({
49             method: 'GET',
50             url: 'https://api.github.com/user/repos',
51             headers: {
52                 Authorization: 'Bearer '+accessToken
53             }
54         })
55         .then((response) => {
56
57             expect(response.status).to.eq(200);
58             expect(response.body[0].id).to.equal(201070920);
59
60         })
61     })
62

```

