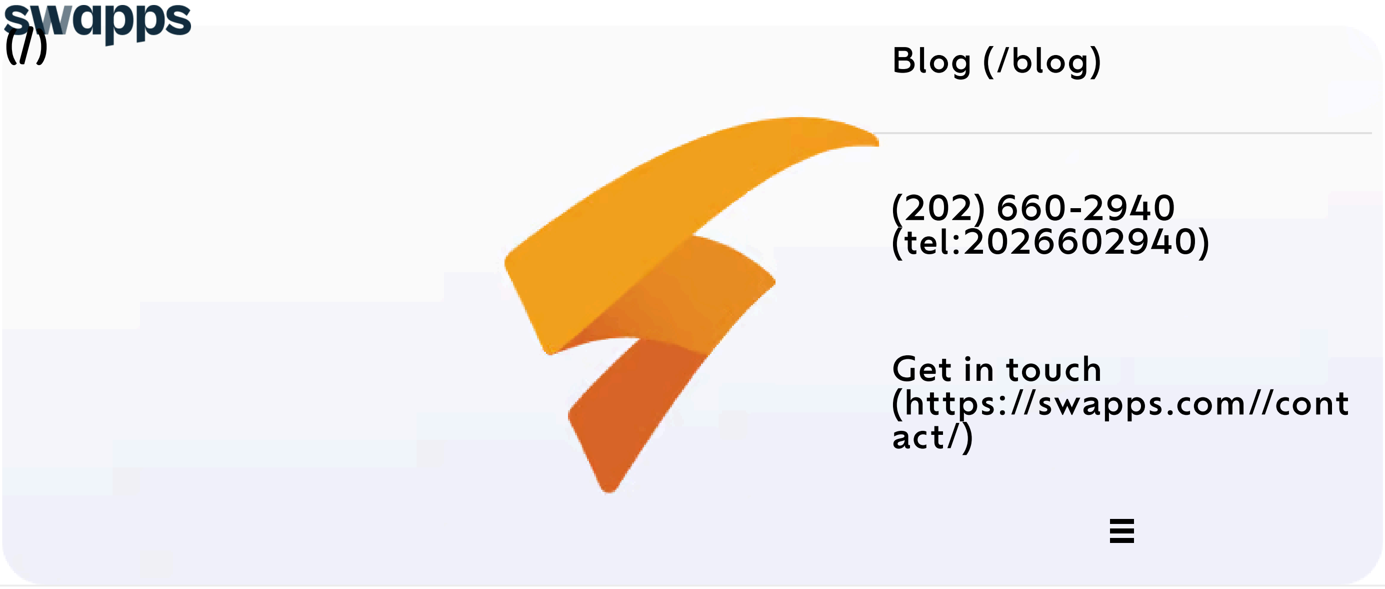
How to add webhooks to a Django project and test them locally

Maria Isabel Jaramillo (https://swapps.com/blog/author/mijaramillo/) – April 1, 2022



In our daily work, we need to receive information about events in external environments, which is why APIs have become popular today, they allow us to connect our applications with other platforms and exchange information using requests. However, every day greater interconnection is sought, seeking to make fewer requests to the APIs, for this, one of the most used approaches is webhooks.

Webhooks are notifications that are triggered automatically when a specific event occurs and are sent through the web. When the event occurs, a notification that contains the response data (or event data) is immediately sent to your chosen destination: the

URL or web application you set. Your webhook should send a **2XX** HTTP response status code back to let the platform know that you received the webhook data.

In this way, we no longer have to ask the platform if the event has occurred, but the platform notifies us as soon as it occurs.

Prepare our environment to test webhooks locally

1. Install ngrok to test the webhook locally. It will expose our local server ports to the Internet creating a proxy (https://ngrok.com/download (https://ngrok.com/download))

On terminal:

```
curl -s https://ngrok-agent.s3.amazonaws.com/ngrok.asc | sudo tee
/etc/apt/trusted.gpg.d/ngrok.asc >/dev/null &&

        echo “deb https://ngrok-agent.s3.amazonaws.com buster main” | sudo tee
/etc/apt/sources.list.d/ngrok.list &&

        sudo apt update && sudo apt install ngrok
```

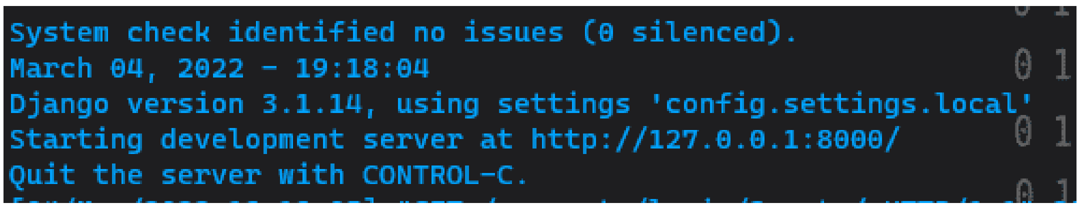
2. Sign up on https://dashboard.ngrok.com/ (https://dashboard.ngrok.com/) to get a token, you can see it here https://dashboard.ngrok.com/

On terminal:

```
ngrok authtoken <token>
```

3. Run ngrok to create the proxy using the same port Django uses when you run *python manage.py runserver*

```
ngrok http <port>
```

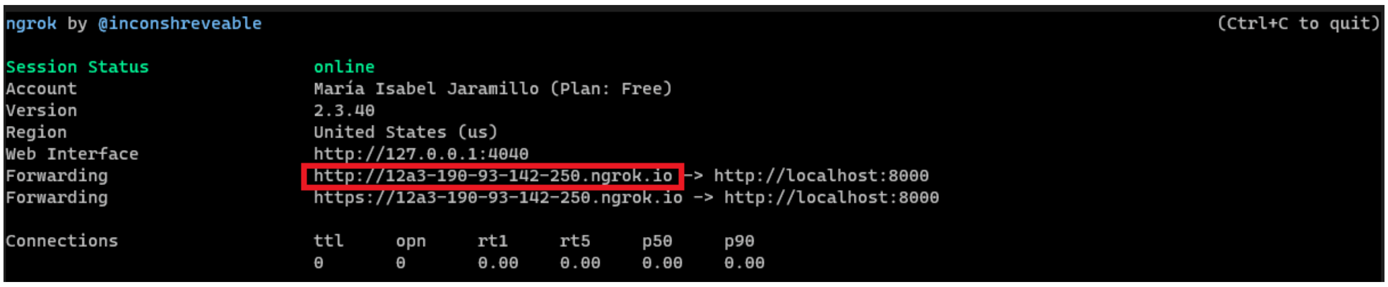


```
System check identified no issues (0 silenced).
March 04, 2022 - 19:18:04
Django version 3.1.14, using settings 'config.settings.local'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

In this example port, 8000 is used, so in the terminal, you run:

ngrok http 8000

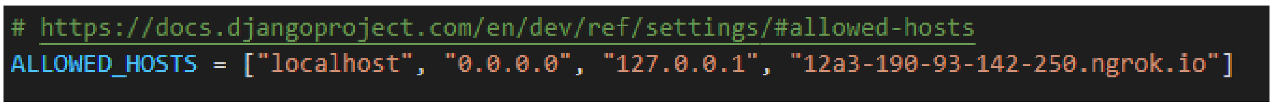
4. Ngrok is running! We will use the http URL



```
ngrok by @inconshreveable                                                                     (Ctrl+C to quit)

Session Status                online
Account                       María Isabel Jaramillo (Plan: Free)
Version                       2.3.40
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://12a3-190-93-142-250.ngrok.io -> http://localhost:8000
Forwarding                    https://12a3-190-93-142-250.ngrok.io -> http://localhost:8000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

5. Allow our django project to use that URL to host the app

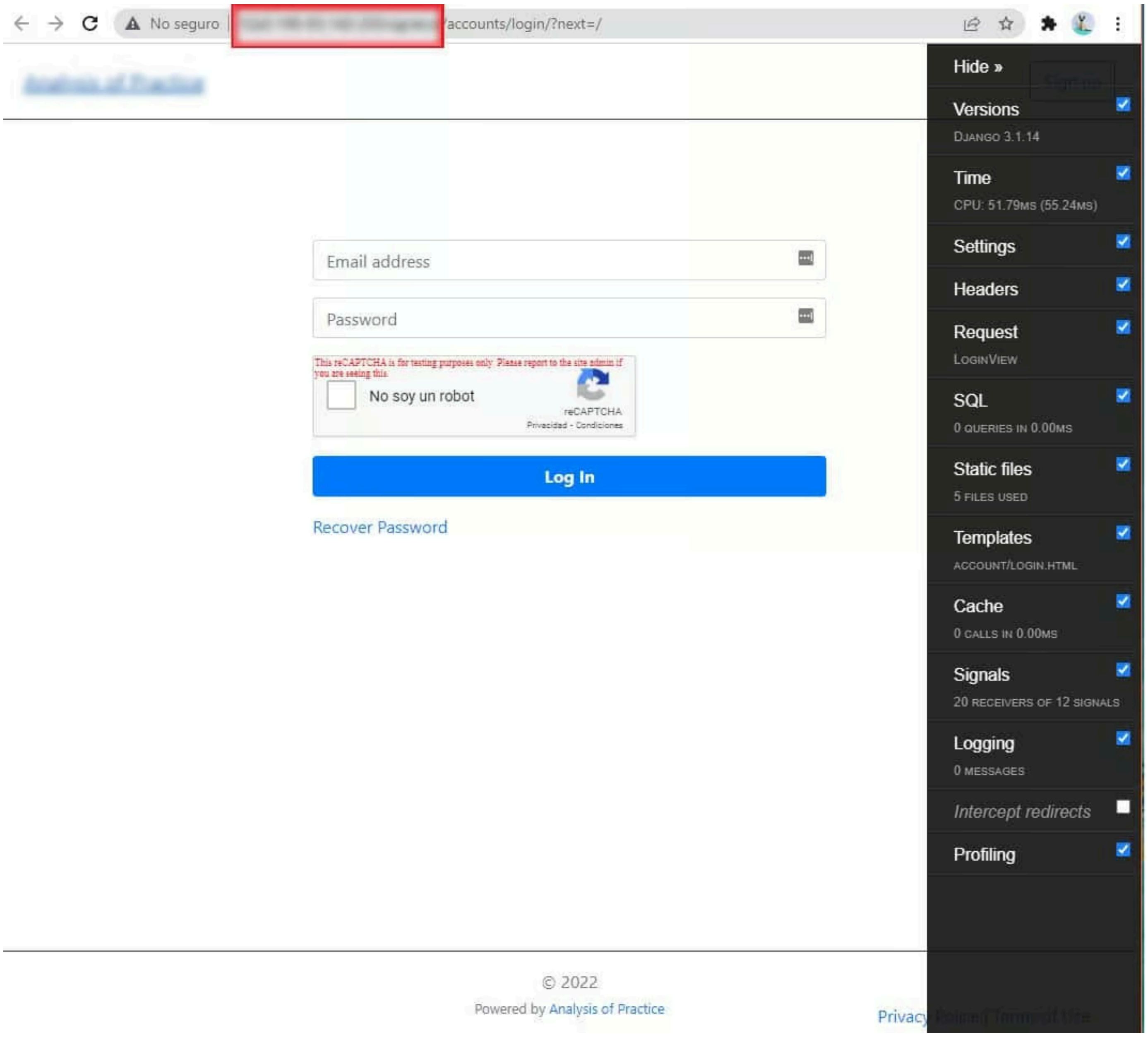
In your settings.py file (config/settings/local if your project uses Django cookiecutter) add the url provided by ngrok to 'ALLOWED HOSTS' without the http…



```
# https://docs.djangoproject.com/en/dev/ref/settings/#allowed-hosts
ALLOWED_HOSTS = ["localhost", "0.0.0.0", "127.0.0.1", "12a3-190-93-142-250.ngrok.io"]
```

6. Test that the project is running at the ngrok URL

Open a new terminal and execute python manage.py runserver, and then open the URL provided by ngrok in the browser, you should see the django app running.

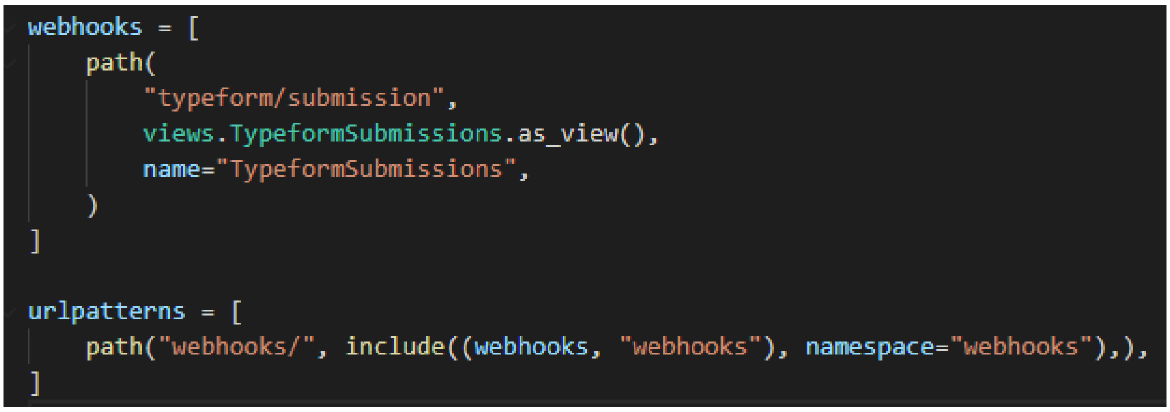


With this, we will be able to test the created webhooks locally. Now let's create one!

Configure Django app-side webhook

1. Create an endpoint for the webhook

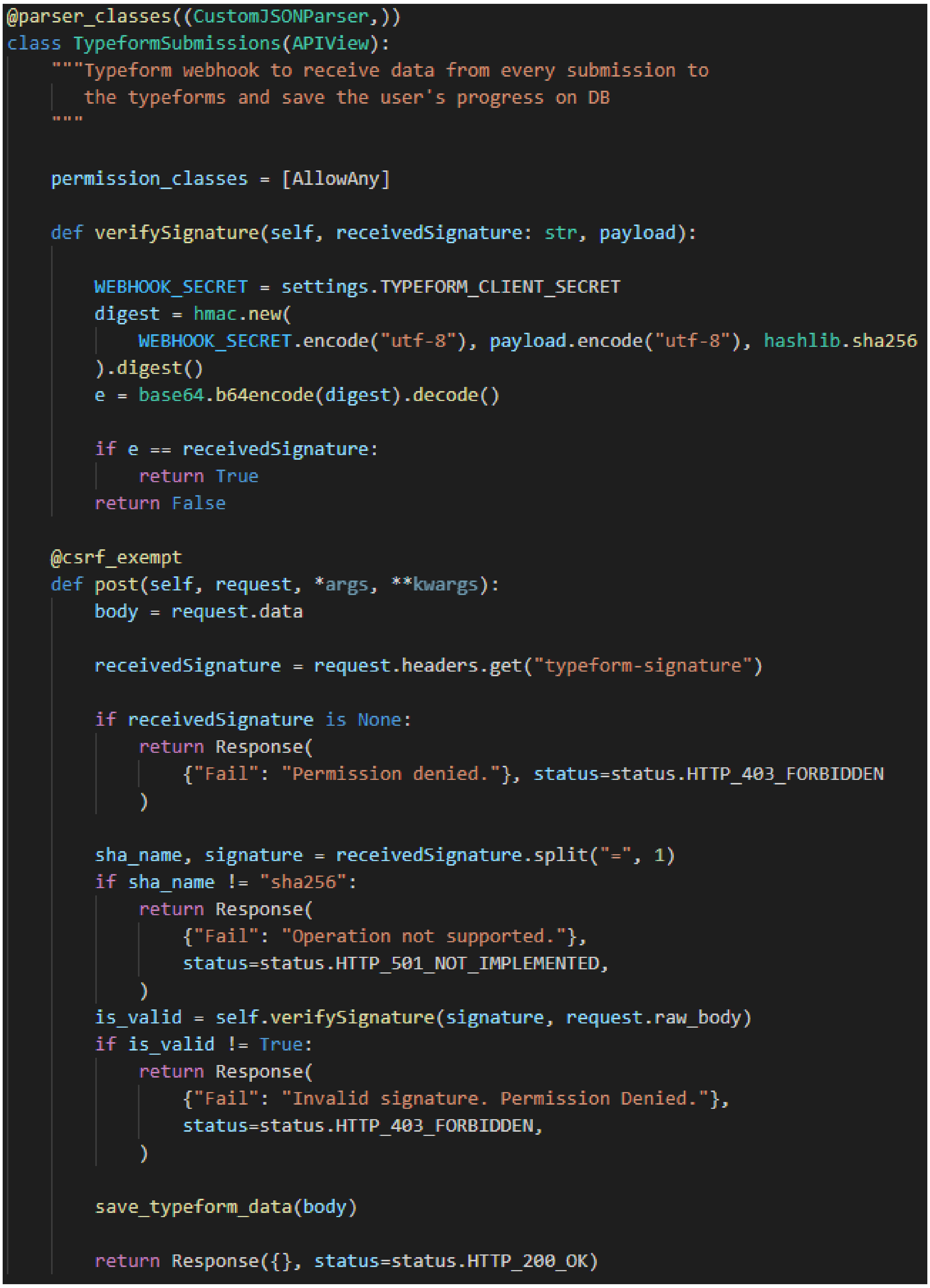
We must create an URL to receive the platform requests in the Django project



```python
webhooks = [
    path(
        "typeform/submission",
        views.TypeformSubmissions.as_view(),
        name="TypeformSubmissions",
    )
]

urlpatterns = [
    path("webhooks/", include((webhooks, "webhooks"), namespace="webhooks"),),
]
```

2. Create a view for the URL

This is an example of a view for the case that we have been developing:



```
@parser_classes((CustomJSONParser,))
class TypeformSubmissions(APIView):
    """Typeform webhook to receive data from every submission to
       the typeforms and save the user's progress on DB
    """

    permission_classes = [AllowAny]

    def verifySignature(self, receivedSignature: str, payload):

        WEBHOOK_SECRET = settings.TYPEFORM_CLIENT_SECRET
        digest = hmac.new(
            WEBHOOK_SECRET.encode("utf-8"), payload.encode("utf-8"), hashlib.sha256
        ).digest()
        e = base64.b64encode(digest).decode()

        if e == receivedSignature:
            return True
        return False

    @csrf_exempt
    def post(self, request, *args, **kwargs):
        body = request.data

        receivedSignature = request.headers.get("typeform-signature")

        if receivedSignature is None:
            return Response(
                {"Fail": "Permission denied."}, status=status.HTTP_403_FORBIDDEN
            )

        sha_name, signature = receivedSignature.split("=", 1)
        if sha_name != "sha256":
            return Response(
                {"Fail": "Operation not supported."},
                status=status.HTTP_501_NOT_IMPLEMENTED,
            )
        is_valid = self.verifySignature(signature, request.raw_body)
        if is_valid != True:
            return Response(
                {"Fail": "Invalid signature. Permission Denied."},
                status=status.HTTP_403_FORBIDDEN,
            )

        save_typeform_data(body)

        return Response({}, status=status.HTTP_200_OK)
```

The view you develop should have:

- Definition of post
- Eximit the post from csrf validation

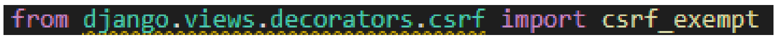
- In this case, the system uses a secret that needs to be validated, a *Signature Validation*. Here, we must give the platform a '*secret*' to validate that the received posts actually come from the source we expect. However, other webhooks may use a different authentication approach, so this is not the only way to do it.

3 Definition of post

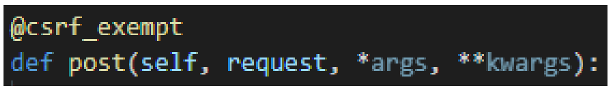
Here we will receive the request, we will validate the signature and we will put our logic.

4. Eximit the post from csrf validation

The platform will not send us the csrf token so we need to tell Django that these requests are still valid. For this, we import the decorator and add it to the post.



```
from django.views.decorators.csrf import csrf_exempt
```



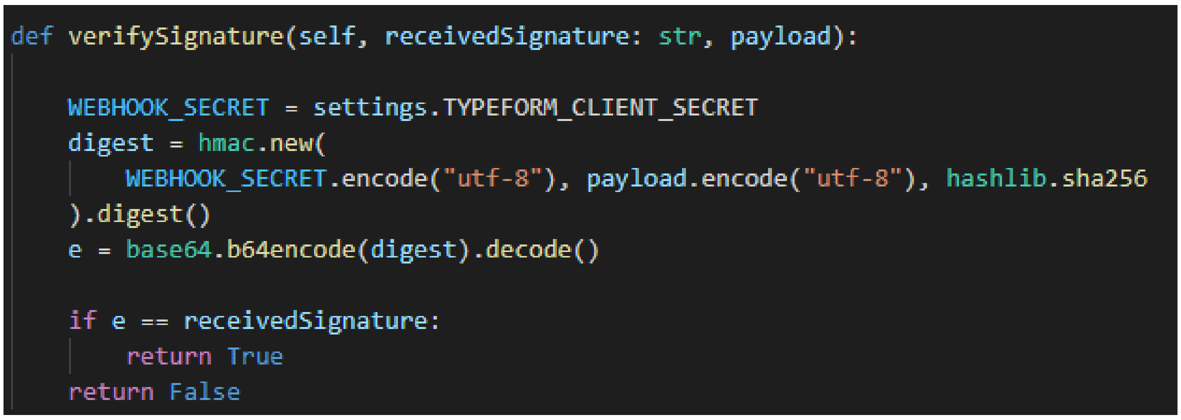
```
@csrf_exempt
def post(self, request, *args, **kwargs):
```

5. Signature validation:

The implementation of this point may vary on each platform. Look in the webhooks documentation of your platform how it should be implemented.

The general idea behind this is that we must give a *secret* to the platform, we should have this same secret in our .env. The platform is going to send us the requests with a signature using the secret, a hash. If the data does indeed come from the platform where we have the webhook (we'll create it later), we should be able to recreate this hash with the secret we have in the .env.

This is what this function does (it's just an example):



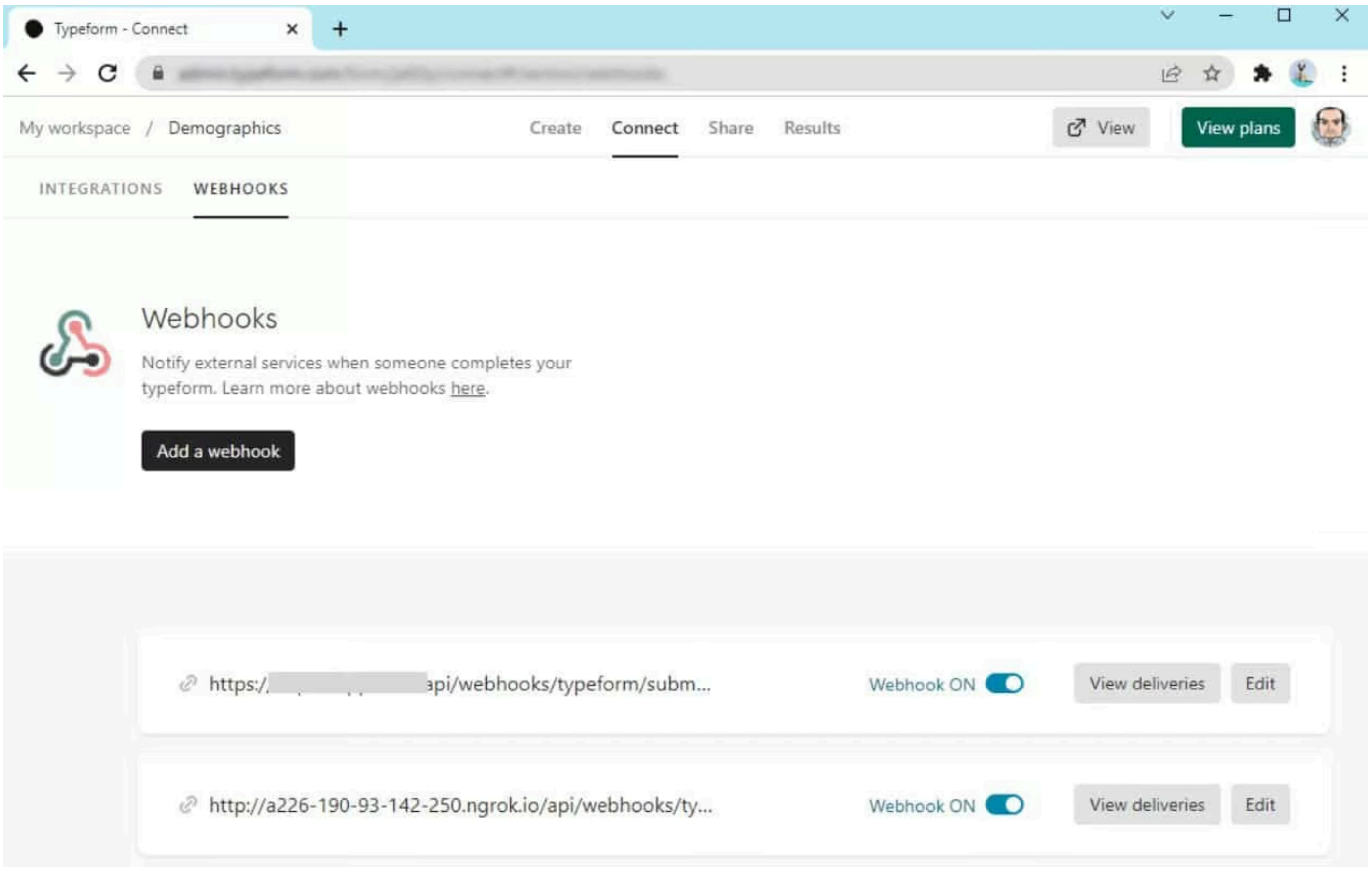
```python
def verifySignature(self, receivedSignature: str, payload):

    WEBHOOK_SECRET = settings.TYPEFORM_CLIENT_SECRET
    digest = hmac.new(
        WEBHOOK_SECRET.encode("utf-8"), payload.encode("utf-8"), hashlib.sha256
    ).digest()
    e = base64.b64encode(digest).decode()

    if e == receivedSignature:
        return True
    return False
```

Make sure that the body of the request or the info with which you will recreate the hash is exactly the same as what the platform would use, this includes spaces, indentation, type of quotes (double, single), etc.

Create a webhook on the platform of interest

1. Find the webhook option on the platform

Look for the 'Webhooks' option on the platform of your interest, you can validate in the documentation if this feature is available and where to find it. You should look for an icon like. In this case, I will do the examples with Typeforms but you will find that it is similar between platforms.

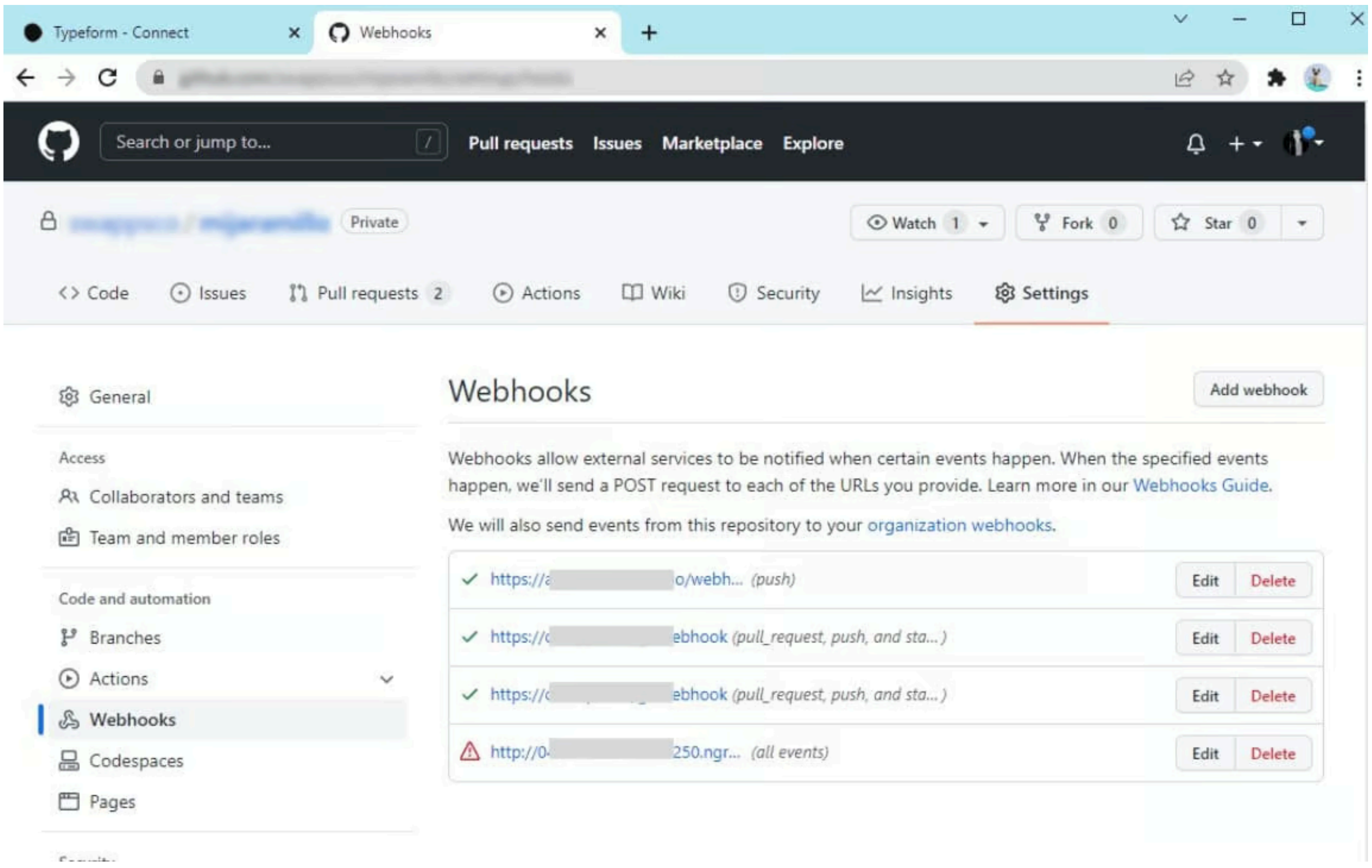


Webhooks

Notify external services when someone completes your typeform. Learn more about webhooks here.

https:/[redacted]api/webhooks/typeform/subm...	Webhook ON	View deliveries	Edit
http://a226-190-93-142-250.ngrok.io/api/webhooks/ty...	Webhook ON	View deliveries	Edit

As you can see, Github is also a platform where webhooks can be created and it looks very similar to the previous example.



Webhooks

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.

We will also send events from this repository to your organization webhooks.

✓	https://[redacted]o/webh... (push)	Edit	Delete
✓	https://[redacted]ebhook (pull_request, push, and sta...)	Edit	Delete
✓	https://[redacted]ebhook (pull_request, push, and sta...)	Edit	Delete
⚠	http://0-[redacted]250.ngr... (all events)	Edit	Delete

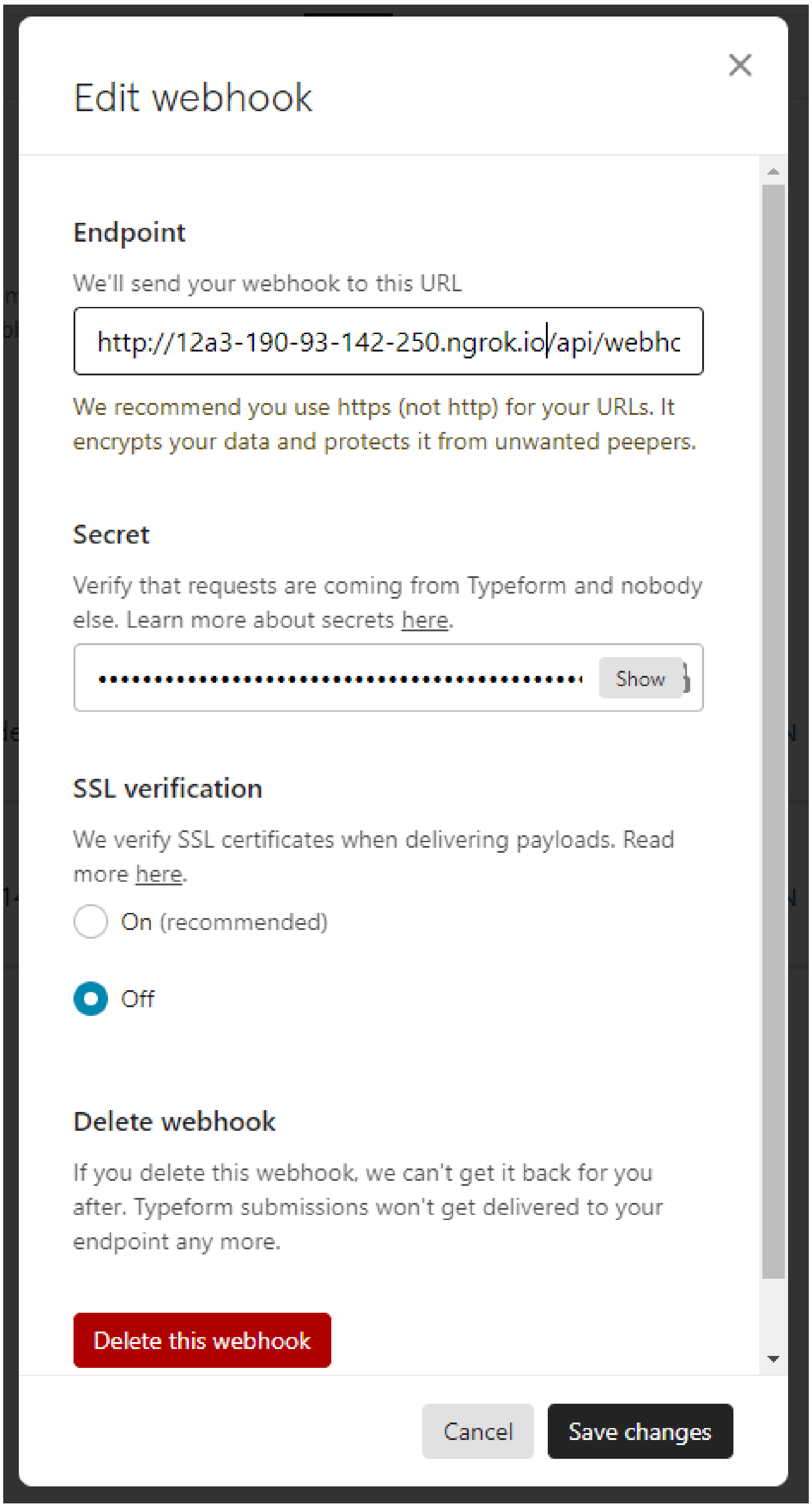
2. Create a webhook:

For this you will need:

  - **Endpoint (Required):** URL that will receive the notifications (POST type request) from the platform each time the event occurs. We will use the URL provided by ngrok along with the pattern we defined previously in the urls.py file of the django project to receive the notifications.

- **Secret (Optional):** To validate that the requests that arrive at our endpoint have been sent by the platform, we must add a secret, otherwise it would not be safe to process that request, its origin would be doubtful.

The endpoint for the URL defined in this Django project would be ***http://12a3-190-93-142-250.ngrok.io/api/webhooks/typeform/submission***


Edit webhook

Endpoint

We'll send your webhook to this URL

http://12a3-190-93-142-250.ngrok.io/api/webhc

We recommend you use https (not http) for your URLs. It encrypts your data and protects it from unwanted peepers.

Secret

Verify that requests are coming from Typeform and nobody else. Learn more about secrets here.

•••

SSL verification

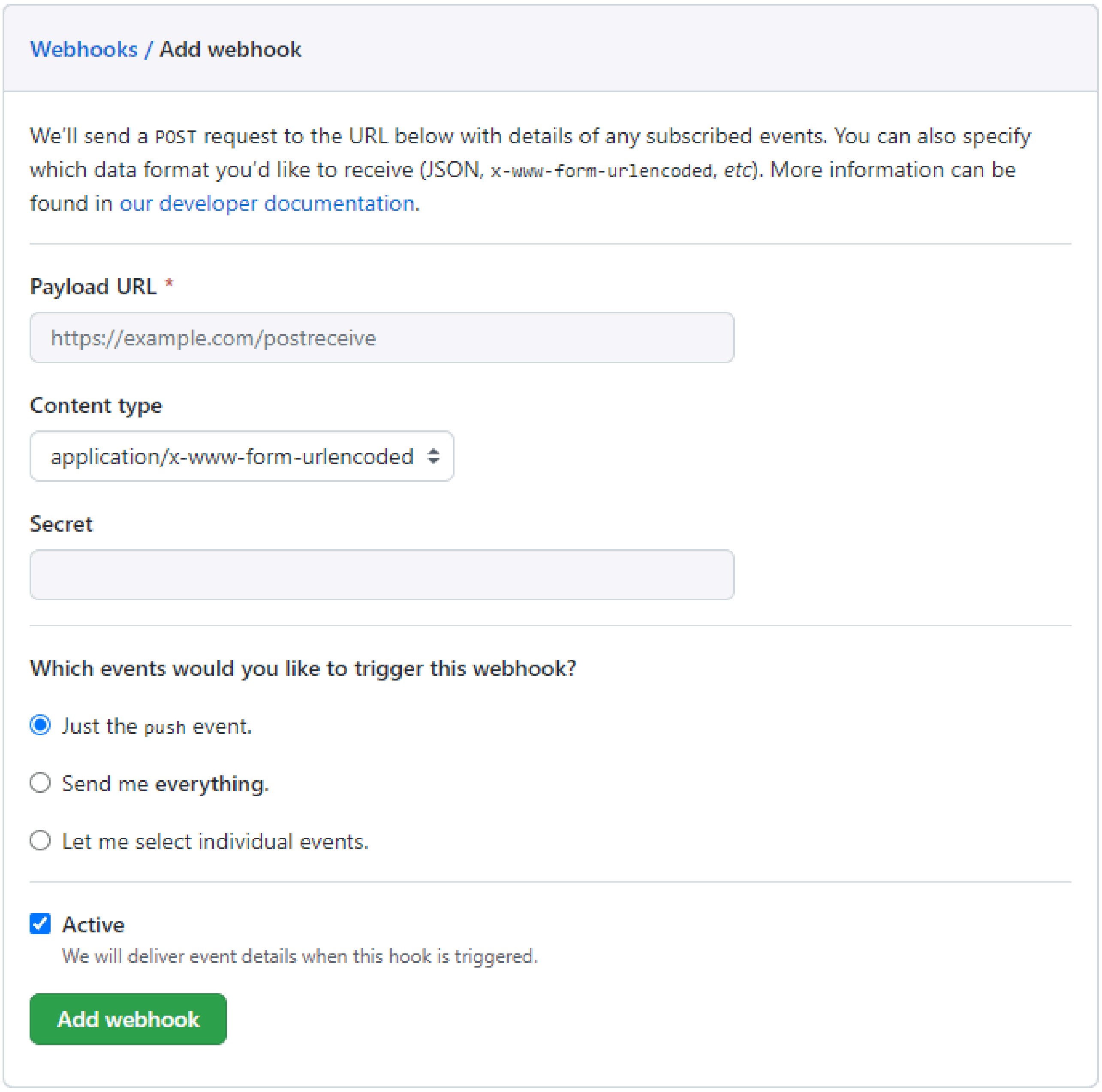
We verify SSL certificates when delivering payloads. Read more here.

On (recommended) [✓ Off]

Delete webhook

If you delete this webhook, we can't get it back for you after. Typeform submissions won't get delivered to your endpoint any more.

Similarly, it would be done on other platforms, here the Payload URL is our endpoint:


Webhooks / Add webhook

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *

https://example.com/postreceive

Content type

application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?

● Just the `push` event.

○ Send me **everything**.

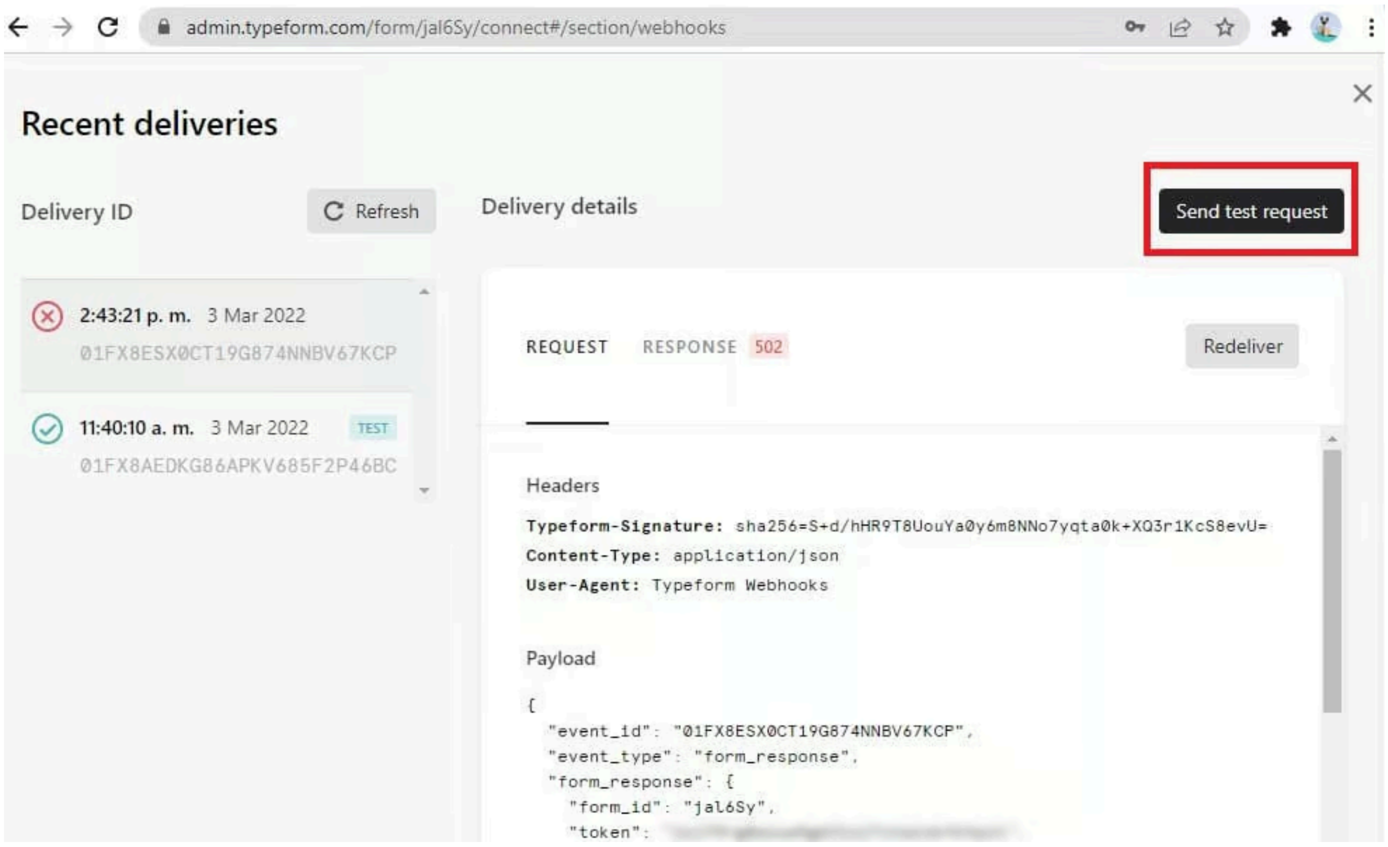
○ Let me select individual events.

✓ Active

We will deliver event details when this hook is triggered.

3. Test the webhook

You can fire the event manually from the platform, or if it offers a testing option when selecting the webhook, generate a test request:


admin.typeform.com/form/jal6Sy/connect#/section/webhooks
Recent deliveries
Delivery ID
Refresh
2:43:21 p. m. 3 Mar 2022
01FX8ESX0CT19G874NNBV67KCP
11:40:10 a. m. 3 Mar 2022 TEST
01FX8AEDKG86APKV685F2P46BC
Delivery details
Send test request
REQUEST
RESPONSE 502
Redeliver

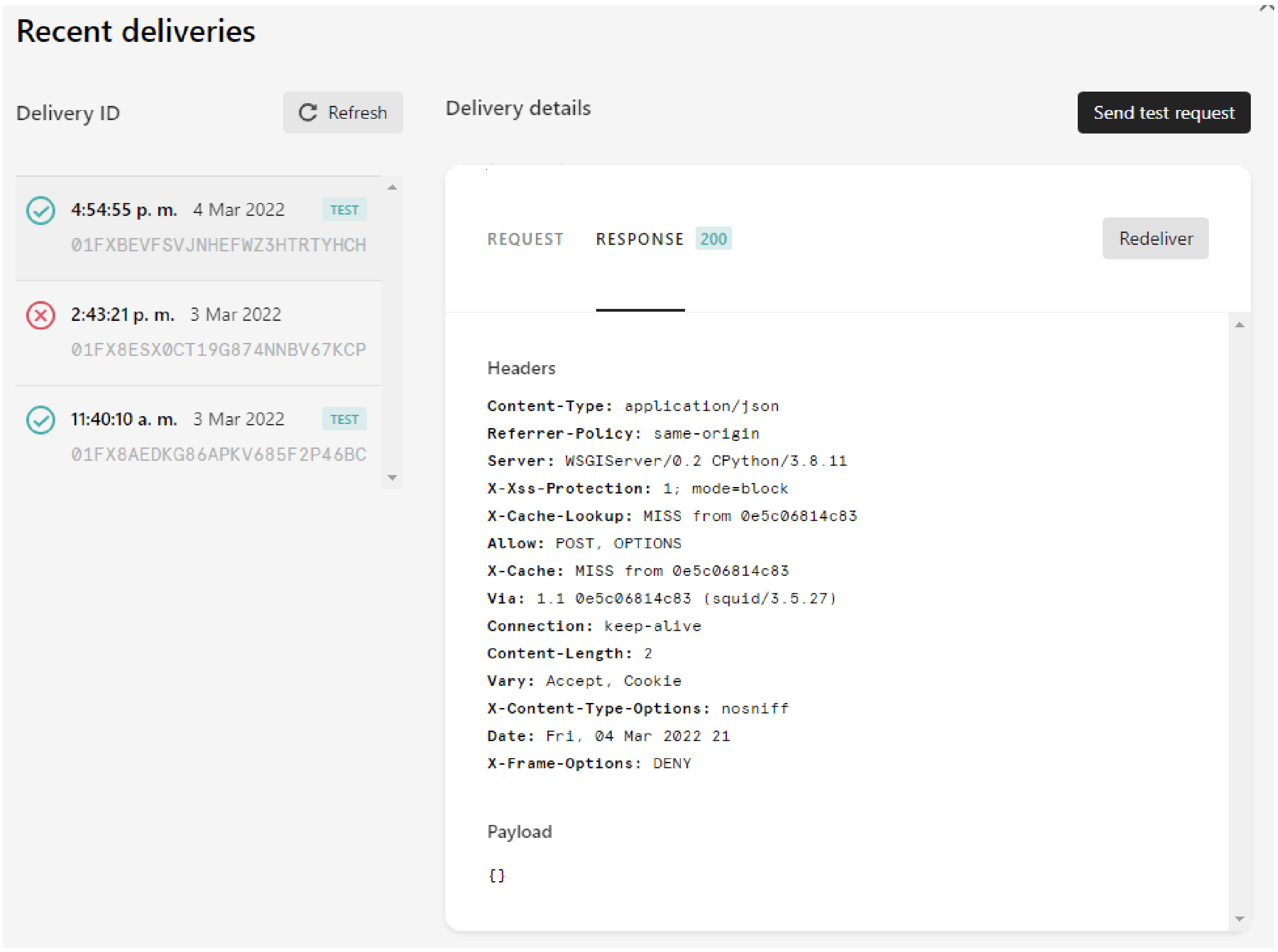
Headers

```
Typeform-Signature: sha256=S+d/hHR9T8UouYa0y6m8NNo7yqta0k+XQ3r1KcS8evU=
Content-Type: application/json
User-Agent: Typeform Webhooks
```

Payload

```
{
  "event_id": "01FX8ESX0CT19G874NNBV67KCP",
  "event_type": "form_response",
  "form_response": {
    "form_id": "jal6Sy",
    "token": [redacted]
```

There you can validate the body and headers of the request made to the Django app, as well as the response received by us:


Recent deliveries
Delivery ID
Refresh
Delivery details
Send test request
4:54:55 p. m. 4 Mar 2022 TEST
01FXBEVFSVJNHEFWZ3HTRTYHCH
2:43:21 p. m. 3 Mar 2022
01FX8ESX0CT19G874NNBV67KCP
11:40:10 a. m. 3 Mar 2022 TEST
01FX8AEDKG86APKV685F2P46BC
REQUEST
RESPONSE 200
Redeliver

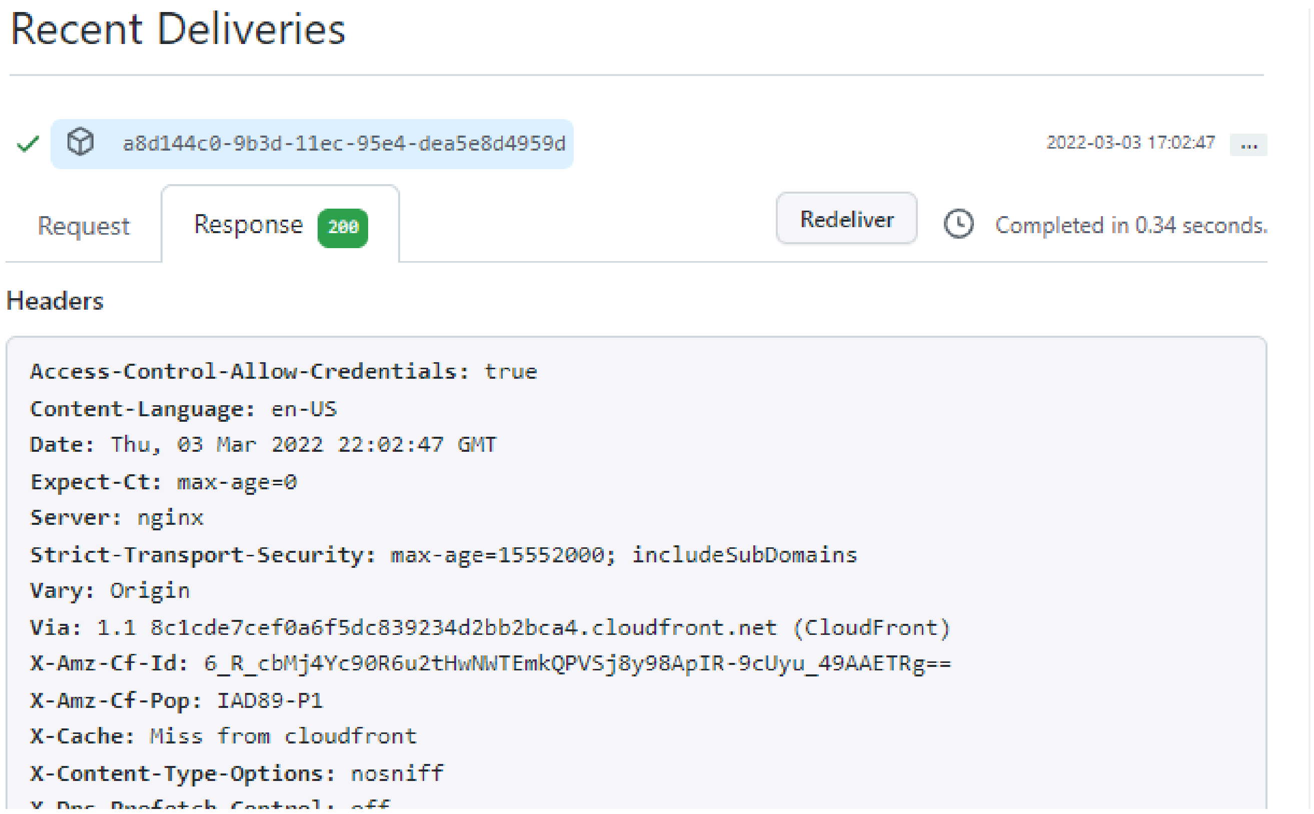
Headers

```
Content-Type: application/json
Referrer-Policy: same-origin
Server: WSGIServer/0.2 CPython/3.8.11
X-Xss-Protection: 1; mode=block
X-Cache-Lookup: MISS from 0e5c06814c83
Allow: POST, OPTIONS
X-Cache: MISS from 0e5c06814c83
Via: 1.1 0e5c06814c83 (squid/3.5.27)
Connection: keep-alive
Content-Length: 2
Vary: Accept, Cookie
X-Content-Type-Options: nosniff
Date: Fri, 04 Mar 2022 21
X-Frame-Options: DENY
```

Payload

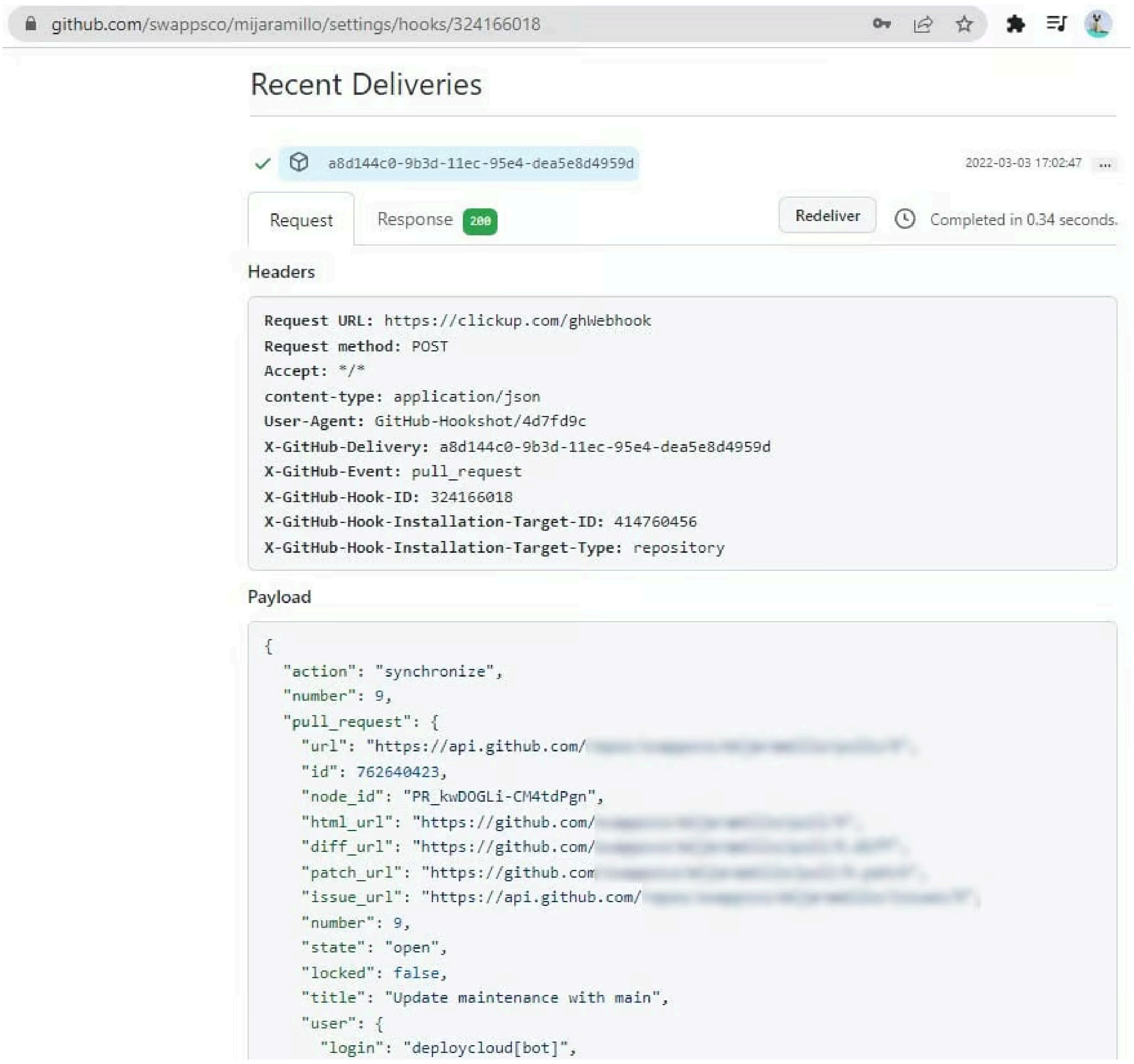
```
{}
```

It works similarly on other platforms:



Headers

```
Access-Control-Allow-Credentials: true
Content-Language: en-US
Date: Thu, 03 Mar 2022 22:02:47 GMT
Expect-Ct: max-age=0
Server: nginx
Strict-Transport-Security: max-age=15552000; includeSubDomains
Vary: Origin
Via: 1.1 8c1cde7cef0a6f5dc839234d2bb2bca4.cloudfront.net (CloudFront)
X-Amz-Cf-Id: 6_R_cbMj4Yc90R6u2tHwNWTEmkQPVSj8y98ApIR-9cUyu_49AAETRg==
X-Amz-Cf-Pop: IAD89-P1
X-Cache: Miss from cloudfront
X-Content-Type-Options: nosniff
X-Dns-Prefetch-Control: off
```


github.com/swappsco/mijaramillo/settings/hooks/324166018
Recent Deliveries
a8d144c0-9b3d-11ec-95e4-dea5e8d4959d
2022-03-03 17:02:47
Request
Response 200
Redeliver
Completed in 0.34 seconds.

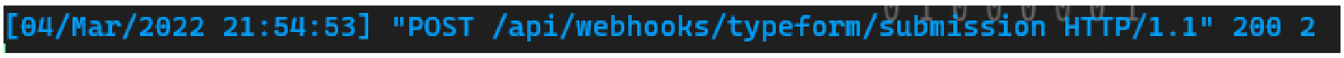
Headers

```
Request URL: https://clickup.com/ghWebhook
Request method: POST
Accept: */*
content-type: application/json
User-Agent: GitHub-Hookshot/4d7fd9c
X-GitHub-Delivery: a8d144c0-9b3d-11ec-95e4-dea5e8d4959d
X-GitHub-Event: pull_request
X-GitHub-Hook-ID: 324166018
X-GitHub-Hook-Installation-Target-ID: 414760456
X-GitHub-Hook-Installation-Target-Type: repository
```

Payload

```
{
  "action": "synchronize",
  "number": 9,
  "pull_request": {
    "url": "https://api.github.com/[redacted]
    "id": 762640423,
    "node_id": "PR_kwDOGLi-CM4tdPgn",
    "html_url": "https://github.com/[redacted]
    "diff_url": "https://github.com/[redacted]
    "patch_url": "https://github.com[redacted]
    "issue_url": "https://api.github.com/[redacted]
    "number": 9,
    "state": "open",
    "locked": false,
    "title": "Update maintenance with main",
    "user": {
      "login": "deploycloud[bot]",
```

You will see that the request will appear in both terminals, where our project is running and the ngrok:



```
[04/Mar/2022 21:54:53] "POST /api/webhooks/typeform/submission HTTP/1.1" 200 2
```