

Learn How To Effectively Use Webhooks For Email Delivery

Mailgun Google Hangout Series

September 2nd 2015



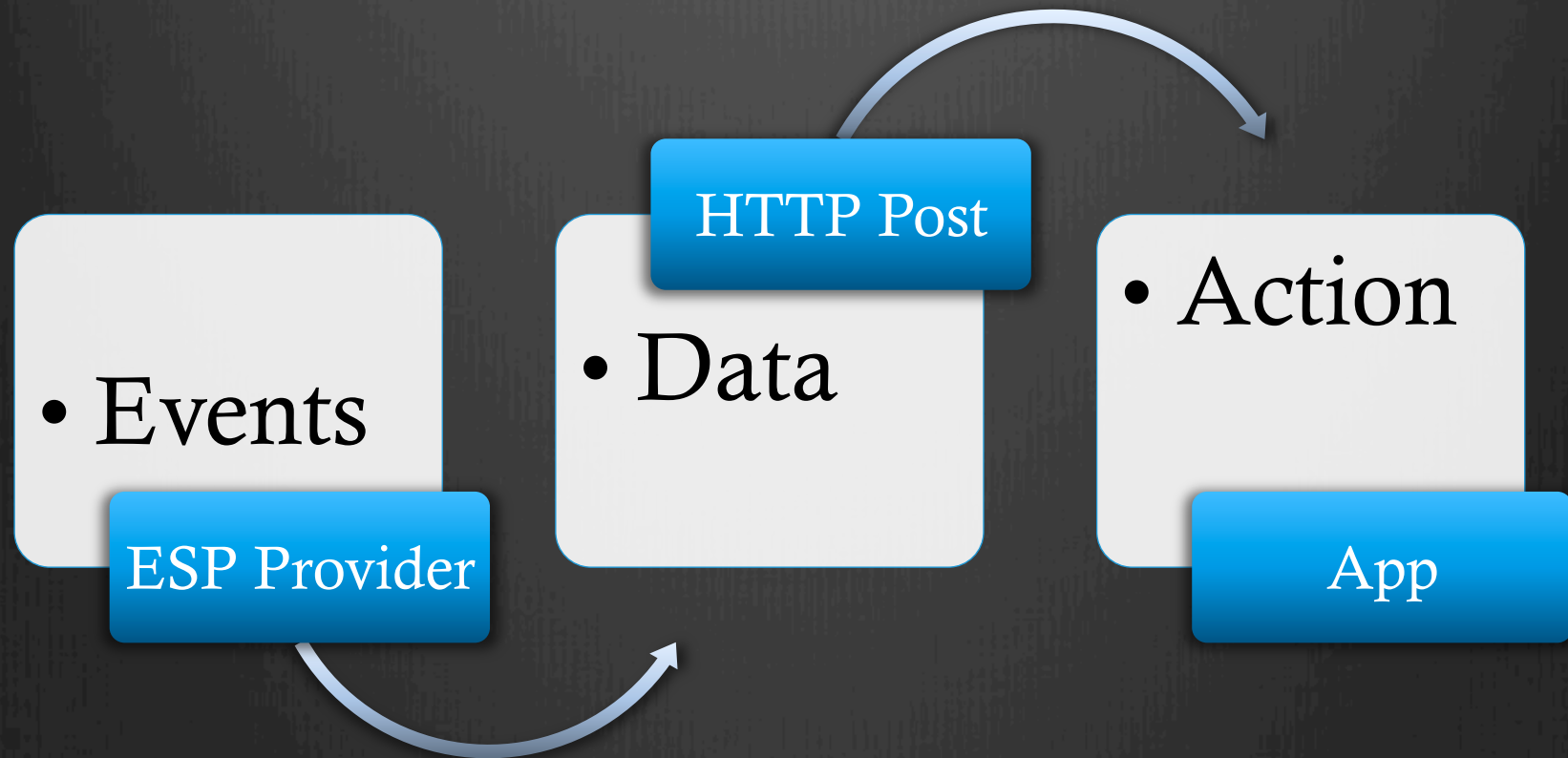
Agenda

- What are webhooks and why use them?
- Defining events and parameters
- Securing webhooks
- Setting up webhooks
- Use case by persona: Developer & User Experience

So... what's a webhook?



How it works



One email, many possible
events...

Upcoming Google Hangout: How to Use Webhooks

From: Angela from Mailgun [Add to Contacts](#)

Sent: Tue, Aug 18, 2015 at 3:32 pm

To:



Was the email successfully delivered? Or did it fail?

Did the email bounce? And was it a soft bounce or a hard bounce?

If delivered, did the recipient open the email?

Did the recipient click within the email?

Did the recipient unsubscribe?

Come Learn About Webhooks.

Hi Angela,

In this hangout, Chris Hammer, Email Expert at Mailgun, will introduce webhooks and demonstrate what they can do for email senders like you and how you can deploy them.

When: Wed, Sep 2nd @ 11am PST / 1pm CST / 2pm EST

What: 20 min presentation, 30 min Q&A

Topic: Learn How To Effectively Use Webhooks For Email Delivery

Submit your questions early!

[Register Now](#)

[Unsubscribe](#) from future Mailgun announcements.

Lots of emails, lots of parameters



Common Parameters

Event

Recipient

Message Headers

Sending Domain

Recipient identifying details
(country, device, email client, OS)



Setting up Webhooks

1. Choose the desired data.
2. Set up the URI.
3. Create scripts to capture data.
4. Finally, fill in the URI in the Mailgun panel for Mailgun to deliver this information to.

Securing Webhooks

A receiving URI must be public, so webhooks should be secured with a **signature, time stamp and token** to create a hash map using an API key to verify that the data is coming from the developer's ESP.

If you are a Mailgun customer,

When an event occurs, we will send data about the event to the webhook URL you specify in the Mailgun panel.















The data sent can be found at:

[https://documentation.mailgun.com/
user_manual.html#tracking-messages](https://documentation.mailgun.com/user_manual.html#tracking-messages)

Webhooks

voxcreator.com



Event	URL
 Delivered messages 	https://txf0yi4mc9o6.runscope.net/delivered
 Dropped messages 	https://txf0yi4mc9o6.runscope.net/dropped
 Hard bounces 	https://txf0yi4mc9o6.runscope.net/bounced
 Spam complaints 	https://txf0yi4mc9o6.runscope.net/complaint
 Unsubscribes 	https://txf0yi4mc9o6.runscope.net/unsubscribe
 Clicks 	https://txf0yi4mc9o6.runscope.net/click
 Opens 	Not Set

TRACK EVENTS WITH WEBHOOKS

Mailgun can notify you of events by sending an HTTP request to a webhook URL you define.

- [Webhooks documentation](#)
- [How do webhooks work?](#)
- [Debug webhooks with Postbin](#)

Use cases

- Tracking what happens to your emails (Developer)
- Tracking recipient engagement of your emails (User Experience)

Use case for the Developer



*I want to capture
bounce requests.*



<https://devcenter.heroku.com/articles/getting-started-with-python-o#specify-dependencies-with-pip>

Sample to catch bounce requests

```
bouncewebhook.py UNREGISTERED
bouncewebhook.py
1 from flask import Flask
2 from flask import request
3 import requests
4 from werkzeug import secure_filename
5 app = Flask(__name__)
6
7 @app.route('/bounced', methods=['GET', 'POST'])
8 def tracking():
9     if request.method == 'POST':
10         f = request.files['attachment-1']
11         d = request.form['message-headers']
12         filename = secure_filename(f.filename)
13         f.save('/tmp/' + filename)
14         requests.post(
15             "https://api.mailgun.net/v3/YOUR_DOMAIN_NAME/messages",
16             auth=("api", "YOUR_API_KEY"),
17             files=[("attachment", open('/tmp/' + filename))],
18             data={"from": "Uh oh <bounced@YOUR_DOMAIN_NAME>",
19                 "to": "YOU@YOUR_DOMAIN_NAME",
20                 "subject": "Your message bounced",
21                 "text": "The attached message bounced. Here are the headers:" + d})
22         return "OK"
23
24 if __name__ == '__main__':
25     app.run(host='0.0.0.0', port=3388, debug=True)
```

Line 24, Column 28 Tab Size: 4 Python

Configure bounce webhook in Mailgun

The screenshot shows the Mailgun web interface in a browser. The page title is "Webhooks - Mailgun". The URL bar shows "https://mailgun.com/app/webhooks/genericdomain.com". The navigation bar includes links for Support, Documentation, and a user profile for Chris Hammer. The main navigation menu has links for Domains, Mailing Lists, Logs, Routes, Tracking, Campaigns, Suppressions, and Webhooks (which is currently selected). On the left, a dropdown menu shows "genericdomain.com". The main content area is titled "Webhooks" and contains a table with two rows: "Delivered messages" (URL: Not Set) and "Dropped messages" (URL: https://infinite-dawn-7491.herokuapp.com/bounced). A modal dialog is open over the "Dropped messages" row, showing the "Event" as "Dropped messages" and the "URL" as "infinite-dawn-7491.herokuapp.com/bounced". The modal includes a "Test Webhook" button and a "Message: OK" confirmation. At the bottom of the modal are "Set Webhook URL" and "Cancel" buttons. To the right of the table, a section titled "TRACK EVENTS WITH WEBHOOKS" explains that Mailgun can notify via HTTP requests and provides links for documentation, how webhooks work, and debugging with Postbin. The footer contains links for Jobs, Help Center, Blog, Twitter, Team, Terms of Service, Privacy Policy, and a copyright notice for 2015 Rackspace, US Inc.

Event	URL
Delivered messages	Not Set
Dropped messages	https://infinite-dawn-7491.herokuapp.com/bounced

Webhooks

genericdomain.com

TRACK EVENTS WITH WEBHOOKS

Mailgun can notify you of events by sending an HTTP request to a webhook URL you define.

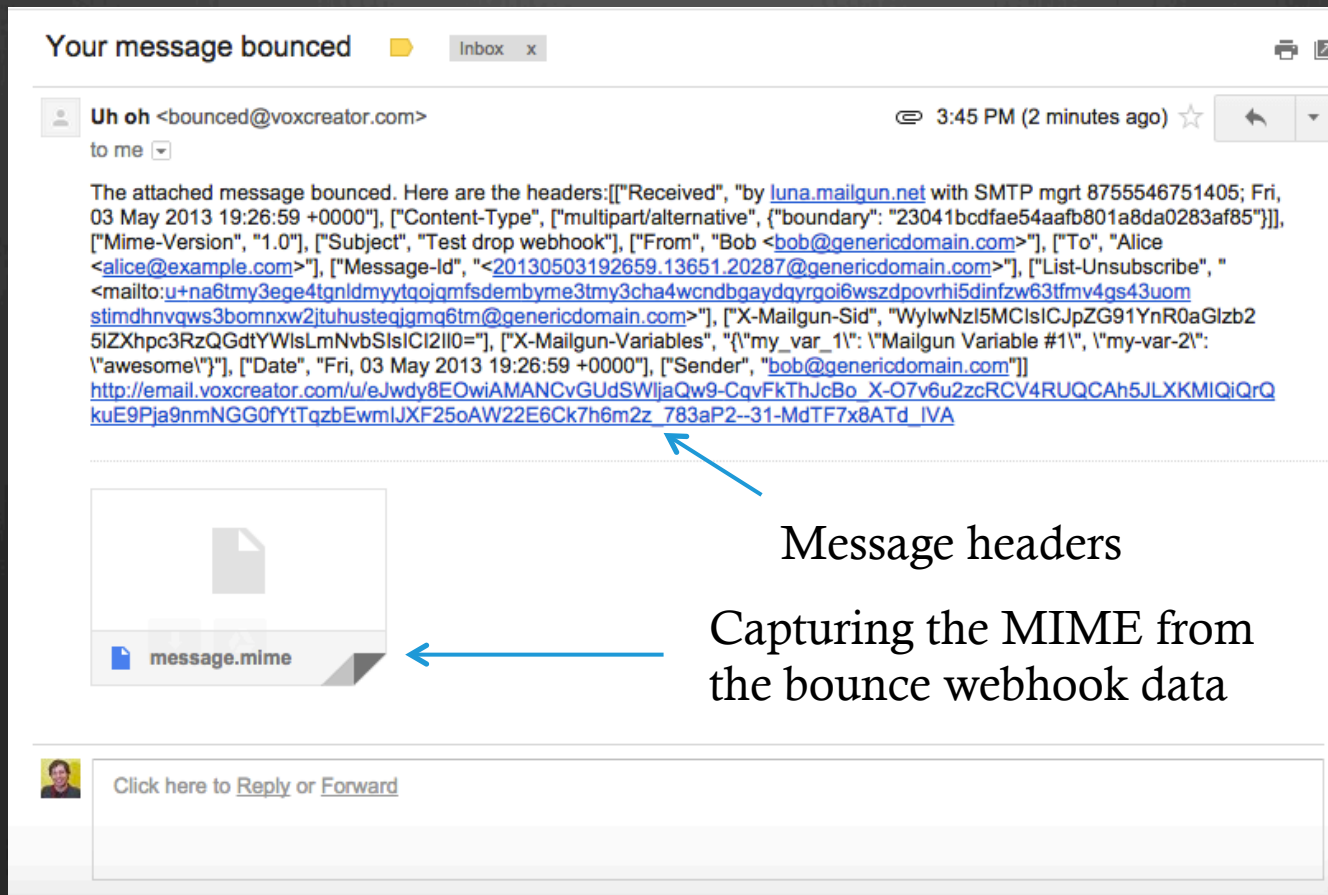
- Webhooks documentation
- How do webhooks work?
- Debug webhooks with Postbin

Jobs • Help Center • Blog • Twitter • Team • Terms of Service • Privacy Policy • © 2015 Rackspace, US Inc.

Check Heroku logs to see status of your request

```
(venv)MPL1MGDV30:helloflask chri4669$ heroku logs --tail -n 10
2015-09-01T20:37:42.902655+00:00 heroku[api]: Deploy 439aeb8 by chris@voxc
reator.com
2015-09-01T20:37:42.902783+00:00 heroku[api]: Release v17 created by chris@voxc
reator.com
2015-09-01T20:44:28.474849+00:00 heroku[web.1]: State changed from down to star
ting
2015-09-01T20:44:33.829030+00:00 heroku[web.1]: Starting process with command `
gunicorn webhook:app --log-file=-`
2015-09-01T20:44:36.228190+00:00 app[web.1]: [2015-09-01 20:44:36 +0000] [3] [I
NFO] Listening at: http://0.0.0.0:20333 (3)
2015-09-01T20:44:36.239672+00:00 app[web.1]: [2015-09-01 20:44:36 +0000] [9] [I
NFO] Booting worker with pid: 9
2015-09-01T20:44:36.227113+00:00 app[web.1]: [2015-09-01 20:44:36 +0000] [3] [I
NFO] Starting gunicorn 19.3.0
2015-09-01T20:44:36.228344+00:00 app[web.1]: [2015-09-01 20:44:36 +0000] [3] [I
NFO] Using worker: sync
2015-09-01T20:44:36.305760+00:00 app[web.1]: [2015-09-01 20:44:36 +0000] [10] [
INFO] Booting worker with pid: 10
2015-09-01T20:44:37.572027+00:00 heroku[web.1]: State changed from starting to
up
2015-09-01T20:45:46.892628+00:00 heroku[router]: at=info method=POST path="/bou
nced" host=infinite-dawn-7491.herokuapp.com request_id=84f42a5e-62f6-4a90-895e-
184d71bd26e8 fwd="173.203.37.68" dyno=web.1 connect=2ms service=167ms status=20
0 bytes=161
```

Check your email!



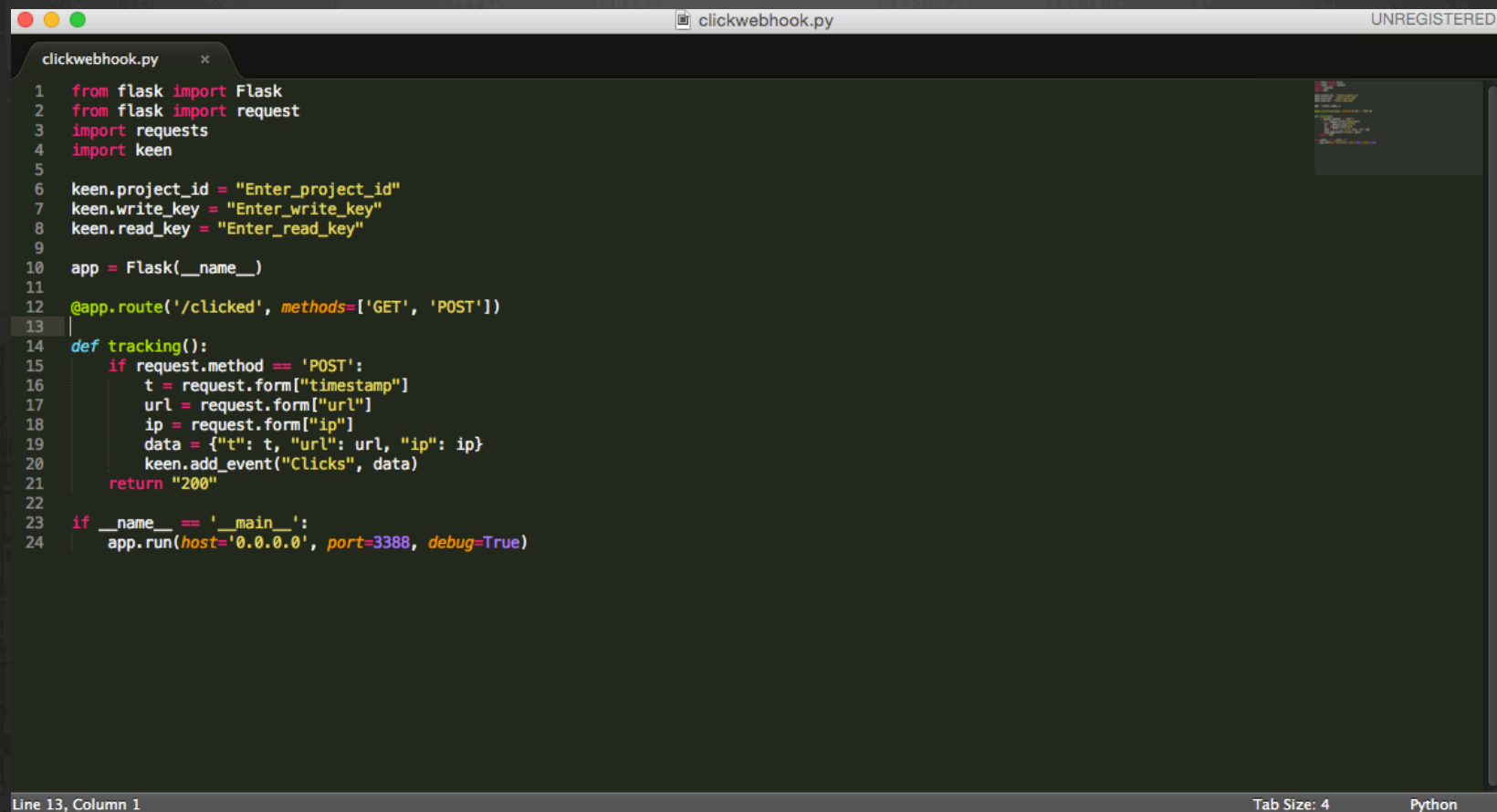
Use case (User Experience)



*I want to know how
engaging my emails
are by capturing clicks.*



Sample to catch clicks



The image shows a code editor window titled 'clickwebhook.py' with a tab icon and a close button. The editor contains Python code for a Flask application that uses the Keen.io analytics library to track clicks. The code is as follows:

```
1 from flask import Flask
2 from flask import request
3 import requests
4 import keen
5
6 keen.project_id = "Enter_project_id"
7 keen.write_key = "Enter_write_key"
8 keen.read_key = "Enter_read_key"
9
10 app = Flask(__name__)
11
12 @app.route('/clicked', methods=['GET', 'POST'])
13
14 def tracking():
15     if request.method == 'POST':
16         t = request.form["timestamp"]
17         url = request.form["url"]
18         ip = request.form["ip"]
19         data = {"t": t, "url": url, "ip": ip}
20         keen.add_event("Clicks", data)
21     return "200"
22
23 if __name__ == '__main__':
24     app.run(host='0.0.0.0', port=3388, debug=True)
```

The status bar at the bottom of the editor shows 'Line 13, Column 1', 'Tab Size: 4', and 'Python'. The top right corner of the editor window displays 'UNREGISTERED'.

Configure click webhook in Mailgun

genericdomain.com ▼

Delivered messages	Not Set
Dropped messages	https://infinite-dawn-7491.herokuapp.com/bounced
Hard bounces	Not Set
Spam complaints	Not Set
Unsubscribes	Not Set
Clicks	https://infinite-dawn-7491.herokuapp.com/clicked

URL you define.

- [Webhooks documentation](#)
- [How do webhooks work?](#)
- [Debug webhooks with Postbin](#)

For the Clicks webhook to work you need to add a CNAME to your DNS as outlined in your [domain settings](#) AND .

Event Clicks

URL

[Test Webhook](#)

Message: 200


[Set Webhook URL](#) [Cancel](#)

Check Heroku logs to see status of your request

```
helloflask — ruby — 101x21
(venv)MPL1MGDV30:helloflask chri4669$ heroku logs --tail -n 10
2015-09-02T01:47:23.117490+00:00 heroku[web.1]: Process exited with status 0
2015-09-02T02:07:05.489783+00:00 heroku[web.1]: Unidling
2015-09-02T02:07:05.489783+00:00 heroku[web.1]: State changed from down to starting
2015-09-02T02:07:10.445132+00:00 heroku[web.1]: Starting process with command `gunicorn webhook:app -
-log-file=-`
2015-09-02T02:07:13.852833+00:00 app[web.1]: [2015-09-02 02:07:13 +0000] [3] [INFO] Starting gunicorn
19.3.0
2015-09-02T02:07:13.861246+00:00 app[web.1]: [2015-09-02 02:07:13 +0000] [3] [INFO] Using worker: syn
c
2015-09-02T02:07:13.920197+00:00 app[web.1]: [2015-09-02 02:07:13 +0000] [9] [INFO] Booting worker wi
th pid: 9
2015-09-02T02:07:13.854060+00:00 app[web.1]: [2015-09-02 02:07:13 +0000] [3] [INFO] Listening at: htt
p://0.0.0.0:43501 (3)
2015-09-02T02:07:13.910475+00:00 app[web.1]: [2015-09-02 02:07:13 +0000] [10] [INFO] Booting worker w
ith pid: 10
2015-09-02T02:07:14.302887+00:00 heroku[web.1]: State changed from starting to up
2015-09-02T02:07:15.800238+00:00 heroku[router]: at=info method=POST path="/clicked" host=infinite-da
wn-7491.herokuapp.com request_id=3af1c90d-aa2d-4088-835e-cd0afecc34d3 fwd="173.203.37.70" dyno=web.1
connect=1ms service=249ms status=200 bytes=162
█
```

Keen.io

<https://elements.heroku.com/addons/keen>

 Docs Support ▾

Project Settings Explorer Saved Queries

Create a new query

Browse

Event Collection

Clicks

Preview collections

Analysis Type

count_unique

Target Property (required)

ip

Timeframe

Relative Absolute

this 14 days

The last 14 day including the current day.

Timezone: UTC

Give your query a name...

Metric ▾

1 Result

Save

</> Embed

?

Resources used

- Mailgun
- Heroku
- Keen.io
- Articles recommended to read:
 - https://documentation.mailgun.com/user_manual.html#tracking-messages
 - <https://devcenter.heroku.com/articles/getting-started-with-python-o#specify-dependencies-with-pip>
 - <http://resources.mailgun.com/webhooks.html>