Testing webhooks

In this article

Learn how to test your webhooks and your code that handles webhook deliveries.

About testing webhooks

You can test webhook delivery. This will let you verify that GitHub sends a webhook delivery in response to an event that you expect to trigger a webhook delivery.

You can also test your code that handles webhook deliveries by using your computer or codespace as a local server and forwarding webhook deliveries to your local server. This will let you develop and debug your code without deploying your code to your production server.

Testing webhook delivery

You can trigger a webhook event and verify that GitHub sent a webhook delivery.

1. Trigger your webhook. For example, if you are testing a repository webhook that is subscribed to the `issues` event, open an issue in the repository where the webhook is configured.

   You can also redeliver a previous webhook delivery. For more information, see "Redelivering webhooks."

   If you are using an organization or repository webhook, you can also use the REST API to trigger the `ping` event for your webhook. If you are using a repository webhook and your webhook is subscribed to the `push` event, you can use the REST API to trigger a test `push` event for your webhook. For more information, see "Repository webhooks" and "Organization webhooks."

2. Check GitHub to verify that a webhook delivery was sent. For information about how to do this for each webhook type, see "Viewing webhook deliveries."

If a webhook delivery was not sent, or if a webhook delivery was sent but GitHub indicates that the delivery failed, refer to the troubleshooting guide to help diagnose the problem. For more information, see "Troubleshooting webhooks."

Testing webhook code locally

In order to test your webhook code locally on your computer or codespace, you can use a webhook proxy URL to forward webhooks from GitHub to your computer or codespace.

You can use your computer or codespace as a local server to receive these forwarded webhooks.

The following sections demonstrate how to use smee.io to provide a webhook proxy URL and forward webhooks.

For specific examples of code and testing steps, see "Handling webhook deliveries."

Get a webhook proxy URL

1. In your browser, navigate to https://smee.io/.
2. Click **Start a new channel**.
3. Copy the full URL under "Webhook Proxy URL". You will use this URL in the following setup steps.

Configure a webhook to use the webhook proxy URL

Configure your webhook to use the webhook proxy URL from above. For more information, see "Creating webhooks" and "Editing webhooks."

Now, GitHub will send webhook deliveries to that URL.

Start a local server

On your computer or codespace, start a local server. The way that you do this depends on how your code to receive webhooks is written. For examples, see "Handling webhook deliveries."

You should make sure that your code can run locally. For example, if your code relies on environment variables on your server in production, you should make sure that the environment variables are also available on your local server.

You may also find it useful to add log statements so that you can verify that steps of your code executed as expected.

Keep your local server running while you test out your webhook.

Forward webhooks

1. If you don't already have smee-client installed, run the following command in your terminal:

   Shell

   ```
   npm install --global smee-client
   ```

2. To receive forwarded webhooks from smee.io, run the following command in your terminal. Replace `WEBHOOK_PROXY_URL` with your webhook proxy URL from earlier. Replace `PATH` with the path or route that your server will handle. Replace `PORT` with the port where your local server is listening.

   Shell

   ```
   smee --url WEBHOOK_PROXY_URL --path /PATH --port PORT
   ```

You should see output that looks like this, with the `WEBHOOK_PROXY_URL`, `PORT`, and `PATH` placeholders replaced with the values you specified:

```shell
Forwarding WEBHOOK_PROXY_URL to http://127.0.0.1:PORT/PATH
Connected WEBHOOK_PROXY_URL
```

Now, when your webhook proxy URL (smee.io URL) receives a webhook delivery from GitHub, smee will forward the webhook delivery to your local server.

3. Keep this running while you test out your webhook. When you want to stop forwarding webhooks, enter `Ctrl` + `C`.

At this point, you should have both your local server running and the smee forwarding running.

Trigger a webhook delivery

Trigger your webhook. For example, if you are testing a repository webhook that is subscribed to the `issues` event, open an issue in the repository where the webhook is configured.

You can also redeliver a previous webhook delivery. For more information, see "Redelivering webhooks."

Verify delivery

You can verify that GitHub sent a webhook delivery, that smee received and forwarded the delivery, and that your local server processed the webhook delivery.

Verify that GitHub sent a delivery

Check GitHub to verify that a webhook delivery was sent. For more information, see "Viewing webhook deliveries."

If a webhook delivery was not sent, or if a webhook delivery was sent but GitHub indicates that the delivery failed, refer to the troubleshooting guide to help diagnose the problem. For more information, see "Troubleshooting webhooks."

Verify that smee received your webhook delivery

Navigate to your webhook proxy URL on smee.io. You should see an event that corresponds to the event that you triggered or redelivered. This indicates that GitHub successfully sent a webhook delivery to the payload URL that you specified.

If you don't see your webhook delivery on smee.io, verify that your webhook is using your webhook proxy URL (smee.io URL).

Verify that smee forwarded your webhook delivery

In the terminal window where you ran `smee --url WEBHOOK_PROXY_URL --path /PATH --port PORT`, you should see something like `POST http://127.0.0.1:3000/webhook - 202`. This indicates that smee successfully forwarded your webhook to your local server.

If you don't see this, make sure that both the smee client and your local server are running. You should have these processes running in two separate terminal windows.

You should also check for errors in the terminal windows where you are running the smee client and your local server. The specific errors depend on how your code to receive webhooks is written. For examples, see "Handling webhook deliveries."

Verify that your local server processed the webhook delivery

At this point, you have verified that GitHub sent a webhook delivery and that smee forwarded the delivery to your local server. Now, you should verify that your code processed the webhook delivery as expected. The way that you do this depends on how your code to receive webhooks is written. For examples, see "Handling webhook deliveries."