API

Webhooks

The Repository Webhooks API allows repository admins to manage the post-receive hooks for a repository. Webhooks can be managed using the JSON HTTP API, or the PubSubHubbub API.

If you would like to set up a single webhook to receive events from all of your organization's repositories, check out our API documentation for Organization Webhooks.

List hooks

```
GET /repos/:owner/:repo/hooks
```

Response

```
Status: 200 OK
Link: <https://api.github.com/resource?page=2>; rel="next",
      <https://api.github.com/resource?page=5>; rel="last"
```

```
[
  {
    "id": 1,
    "url": "https://api.github.com/repos/octocat/Hello-World/hooks/1",
    "test_url": "https://api.github.com/repos/octocat/Hello-World/hooks/1/test",
    "ping_url": "https://api.github.com/repos/octocat/Hello-World/hooks/1/pings",
    "name": "web",
    "events": [
      "push",
      "pull_request"
    ],
    "active": true,
    "config": {
      "url": "http://example.com/webhook",
      "content_type": "json"
    },
    "updated_at": "2011-09-06T20:39:23Z",
    "created_at": "2011-09-06T17:26:27Z"
  }
]
```

Get single hook

```
GET /repos/:owner/:repo/hooks/:id
```

Response

```
Status: 200 OK
```

```
{
  "id": 1,
  "url": "https://api.github.com/repos/octocat/Hello-World/hooks/1",
  "test_url": "https://api.github.com/repos/octocat/Hello-World/hooks/1/test",
  "ping_url": "https://api.github.com/repos/octocat/Hello-World/hooks/1/pings",
  "name": "web",
  "events": [
    "push",
    "pull_request"
  ],
  "active": true,
  "config": {
    "url": "http://example.com/webhook",
    "content_type": "json"
  },
  "updated_at": "2011-09-06T20:39:23Z",
  "created_at": "2011-09-06T17:26:27Z"
}
```

Create a hook

```
POST /repos/:owner/:repo/hooks
```

Note: Repository service hooks (like email or Campfire) can have at most one configured at a time. Creating hooks for a service that already has one configured will update the existing hook.

Repositories can have multiple webhooks installed. Each webhook should have a unique `config`. Multiple webhooks can share the same `config` as long as those webhooks do not have any `events` that overlap.

Parameters

Name	Type	Description
`name`	`string`	**Required**. Use `web` for a webhook or use the name of a valid service. (See /hooks for the list of valid service names.)
`config`	`object`	**Required**. Key/value pairs to provide settings for this hook. These settings vary between hooks and some are defined in the github-services repository. Booleans are stored internally as "1" for true, and "0" for false. Any JSON `true` / `false` values will be converted automatically.
`events`	`array`	Determines what events the hook is triggered for. Default: `["push"]`
`active`	`boolean`	Determines whether the hook is actually triggered on pushes.

Example

To create a webhook, the following fields are required by the `config`:

- `url`: A required string defining the URL to which the payloads will be delivered.
- `content_type`: An optional string defining the media type used to serialize the payloads. Supported values include `json` and `form`. The default is `form`.
- `secret`: An optional string that's passed with the HTTP requests as an `X-Hub-Signature` header. The value of this header is computed as the HMAC hex digest of the body, using the `secret` as the key.
- `insecure_ssl`: An optional string that determines whether the SSL certificate of the host for `url` will be verified when delivering payloads. Supported values include `"0"` (verification is

performed) and `"1"` (verification is not performed). The default is `"0"`.

Here's how you can create a hook that posts payloads in JSON format:

```
{
  "name": "web",
  "active": true,
  "events": [
    "push",
    "pull_request"
  ],
  "config": {
    "url": "http://example.com/webhook",
    "content_type": "json"
  }
}
```

Response

```
Status: 201 Created
Location: https://api.github.com/repos/octocat/Hello-World/hooks/1
```

```
{
  "id": 1,
  "url": "https://api.github.com/repos/octocat/Hello-World/hooks/1",
  "test_url": "https://api.github.com/repos/octocat/Hello-World/hooks/1/test",
  "ping_url": "https://api.github.com/repos/octocat/Hello-World/hooks/1/pings",
  "name": "web",
  "events": [
    "push",
    "pull_request"
  ],
  "active": true,
  "config": {
    "url": "http://example.com/webhook",
    "content_type": "json"
  },
  "updated_at": "2011-09-06T20:39:23Z",
  "created_at": "2011-09-06T17:26:27Z"
}
```

Edit a hook

```
PATCH /repos/:owner/:repo/hooks/:id
```

Parameters

Name	Type	Description

`config`	`object`	Key/value pairs to provide settings for this hook. Modifying this will replace the entire config object. These settings vary between hooks and some are defined in the github-services repository. Booleans are stored internally as "1" for true, and "0" for false. Any JSON `true` / `false` values will be converted automatically.
`events`	`array`	Determines what events the hook is triggered for. This replaces the entire array of events. Default: `["push"]`
`add_events`	`array`	Determines a list of events to be added to the list of events that the Hook triggers for.
`remove_events`	`array`	Determines a list of events to be removed from the list of events that the Hook triggers for.
`active`	`boolean`	Determines whether the hook is actually triggered on pushes.

Example

```
{
  "active": true,
  "add_events": [
    "pull_request"
  ]
}
```

Response

```
Status: 200 OK
```

```
{
  "id": 1,
  "url": "https://api.github.com/repos/octocat/Hello-World/hooks/1",
  "test_url": "https://api.github.com/repos/octocat/Hello-World/hooks/1/test",
  "ping_url": "https://api.github.com/repos/octocat/Hello-World/hooks/1/pings",
  "name": "web",
  "events": [
    "push",
    "pull_request"
  ],
  "active": true,
  "config": {
    "url": "http://example.com/webhook",
    "content_type": "json"
  },
  "updated_at": "2011-09-06T20:39:23Z",
  "created_at": "2011-09-06T17:26:27Z"
}
```

Test a `push` hook

This will trigger the hook with the latest push to the current repository if the hook is subscribed to `push` events. If the hook is not subscribed to `push` events, the server will respond with 204 but no test POST will be generated.

```
POST /repos/:owner/:repo/hooks/:id/tests
```

Note: Previously `/repos/:owner/:repo/hooks/:id/test`

Response

```
Status: 204 No Content
```

Ping a hook

This will trigger a ping event to be sent to the hook.

```
POST /repos/:owner/:repo/hooks/:id/pings
```

Response

```
Status: 204 No Content
```

Delete a hook

```
DELETE /repos/:owner/:repo/hooks/:id
```

Response

```
Status: 204 No Content
```

Receiving Webhooks

In order for GitHub to send webhook payloads, your server needs to be accessible from the Internet. We also highly suggest using SSL so that we can send encrypted payloads over HTTPS.

Webhook Headers

GitHub will send along several HTTP headers to differentiate between event types and payload identifiers.

Name	Description

`X-GitHub-Event`	The event type that was triggered.
`X-GitHub-Delivery`	A guid to identify the payload and event being sent.
`X-Hub-Signature`	The value of this header is computed as the HMAC hex digest of the body, using the `secret` config option as the key.

PubSubHubbub

GitHub can also serve as a PubSubHubbub hub for all repositories. PSHB is a simple publish/subscribe protocol that lets servers register to receive updates when a topic is updated. The updates are sent with an HTTP POST request to a callback URL. Topic URLs for a GitHub repository's pushes are in this format:

```
https://github.com/:owner/:repo/events/:event
```

The event can be any event string that is listed at the top of this document.

Response format

The default format is what existing post-receive hooks should expect: A JSON body sent as the `payload` parameter in a POST. You can also specify to receive the raw JSON body with either an `Accept` header, or a `.json` extension.

```
Accept: application/json
https://github.com/:owner/:repo/events/push.json
```

Callback URLs

Callback URLs can use either the `http://` protocol, or `github://`. `github://` callbacks specify a GitHub service.

```
# Send updates to postbin.org
http://postbin.org/123

# Send updates to Campfire
github://campfire?subdomain=github&room=Commits&token=abc123
```

Subscribing

The GitHub PubSubHubbub endpoint is: `https://api.github.com/hub`. (GitHub Enterprise users should use `http://yourhost/api/v3/hub` as the PubSubHubbub endpoint, but not change the `hub.topic` URI format.) A successful request with curl looks like:

```
curl -u "user" -i \
  https://api.github.com/hub \
  -F "hub.mode=subscribe" \
  -F "hub.topic=https://github.com/:owner/:repo/events/push" \
  -F "hub.callback=http://postbin.org/123"
```

PubSubHubbub requests can be sent multiple times. If the hook already exists, it will be modified according to the request.

Parameters

Name	Type	Description
`hub.mode`	`string`	**Required**. Either `subscribe` or `unsubscribe`.
`hub.topic`	`string`	**Required**. The URI of the GitHub repository to subscribe to. The path must be in the format of `/:owner/:repo/events/:event`.
`hub.callback`	`string`	The URI to receive the updates to the topic.
`hub.secret`	`string`	A shared secret key that generates a SHA1 HMAC of the outgoing body content. You can verify a push came from GitHub by comparing the raw request body with the contents of the `X-Hub-Signature` header. You can see the PubSubHubbub documentation for more details.

Navigate the docs…