GitHub Project

Webhook handling and other sysadmin tasks

Anonymous · 21/08/2026 · #php

Webhook handling and other sysadmin tasks

A lightweight, no-dependency PHP library for handling GitHub webhooks, with a focus on security and extensibility.

Overview

This library provides a simple, secure way to handle GitHub webhooks, allowing you to easily react to GitHub events such as push notifications, pull requests, workflow runs, and more.

Features

- **Zero Dependencies**: Doesn't require any external packages.
- **Secure**: Built with security best practices, including HMAC signature validation.
- **Extensible**: Easily add custom handlers for any GitHub event.
- **Typed**: Fully typed for modern PHP 8.3 environments.
- **Event-driven**: Handle different GitHub webhook events with separate handlers.

Installation

Via Composer

```
composer require derafu/github
```

Manual Installation

Clone the repository:

```
git clone https://github.com/derafu/github.git
```

Usage

Basic Setup

1. Create a webhook endpoint:

```
<?php
// webhook.php

declare(strict_types=1);

namespace Derafu\GitHub\Webhook;

use Derafu\GitHub\Logger;
use Derafu\GitHub\Webhook\Handler;
use Derafu\GitHub\Webhook\Response;
use Exception;

// Load the autoloader.
require 'vendor/autoload.php';

// Load configuration.
$config = require 'config.php';

// Create the handler.
$logger = new Logger();
$handler = new Handler($config, $logger);

// Handle the webhook.
try {
    $notification = $handler->handle();
    $response = $notification->getResponse();
} catch (Exception $e) {
    $response = new Response([
        'code' => $e->getCode() ?: 1,
        'data' => [
            'message' => $e->getMessage(),
            'logs' => $logger->getLogs(),
        ],
    ]);
}

// Send the response.
http_response_code($response->getHttpCode());
header('Content-Type: application/json');
echo json_encode($response->toArray(), JSON_PRETTY_PRINT);
```

2. Create a configuration file:

```
<?php
// config.php

declare(strict_types=1);

use Derafu\GitHub\Webhook\EventHandler\WorkflowRunHandler;
use Derafu\GitHub\Webhook\Notification;

return [
    // Add handlers for different events.
    'handlers' => [
        'workflow_run' => fn (Notification $notification) =>
WorkflowRunHandler::deploy(
            $notification
        ),
    ],
];
```

GitHub Webhook Configuration

1. Go to your GitHub repository.
2. Navigate to Settings > Webhooks > Add webhook.
3. Set the Payload URL to your webhook endpoint (e.g., `https://example.com/webhook.php`).
4. Set Content type to `application/json`.
5. Set a Secret that matches your configuration.
6. Choose which events to receive.
7. Ensure the webhook is active.

Handler Examples

Workflow Run Handler

This handler deploys your application when a GitHub Action workflow completes successfully:

```
<?php

namespace Derafu\GitHub\Webhook\EventHandler;

use Derafu\GitHub\Webhook\Notification;

final class WorkflowRunHandler
{
    public static function deploy(
        Notification $notification,
        string $deployer,
        array $sites
```

```
    ): ?string {
        $payload = $notification->getPayload();

        $branch = $payload->workflow_run->head_branch;
        $workflow = $payload->workflow->name;
        $status = $payload->workflow_run->status;
        $conclusion = $payload->workflow_run->conclusion;

        // Check for matching sites and deploy if conditions are met.
        foreach ($sites as $site => $config) {
            // Deployment logic here.
        }

        return null;
    }
}
```

The included handler uses Docker and Deployer.

Custom Handler Example

You can create handlers for any GitHub event:

```
<?php
// In your config.php

'push' => function (Notification $notification) {
    $payload = $notification->getPayload();
    $repo = $payload->repository->name;
    $branch = str_replace('refs/heads/', '', $payload->ref);

    // Your custom logic here.

    $notification->setResponse("Processed push to $repo on branch $branch.");
},
```

Security Best Practices

- Always validate the webhook signature with your secret.
- Use environment variables for storing secrets.
- Escape any arguments used in shell commands with `escapeshellarg()`.
- Validate repository names against a whitelist.
- Implement proper error handling and logging.

Supported Events

The library supports all GitHub webhook events, including:

- `fork`
- `ping`
- `push`
- `pull_request`
- `release`
- `star`
- `status`
- `workflow_run`

For the events not explicitly declared, you can configure a closure for the `default` handler and take any action from there.

Last updated on 21/08/2026