



Deploy Laravel and PHP applications on DigitalOcean, Linode, & AWS.

ADS VIA CARBON

# Laravel Socialite

- # Introduction
- # Upgrading Socialite
- # Installation
- # Configuration
- # Routing
- # Optional Parameters
- # Access Scopes
- # Stateless Authentication
- # Retrieving User Details

## # Introduction

In addition to typical, form based authentication, Laravel also provides a simple, convenient way to authenticate with OAuth providers using [Laravel Socialite](#). Socialite currently supports authentication with Facebook, Twitter, LinkedIn, Google, GitHub, GitLab and Bitbucket.



Adapters for other platforms are listed at the community driven [Socialite Providers](#) website.

## # Upgrading Socialite

When upgrading to a new major version of Socialite, it's important that you carefully review [the upgrade guide](#).

## # Installation

To get started with Socialite, use Composer to add the package to your project's dependencies:

```
composer require laravel/socialite
```

## # Configuration

Before using Socialite, you will also need to add credentials for the OAuth services your application utilizes. These credentials should be placed in your `config/services.php` configuration file, and should use the key `facebook`, `twitter`, `linkedin`, `google`, `github`, `gitlab` or `bitbucket`, depending on the providers your application requires. For example:

```
'github' => [
    'client_id' => env('GITHUB_CLIENT_ID'),
    'client_secret' => env('GITHUB_CLIENT_SECRET'),
    'redirect' => 'http://your-callback-url',
],
```



If the `redirect` option contains a relative path, it will automatically be resolved to a fully qualified URL.

## # Routing

Next, you are ready to authenticate users! You will need two routes: one for redirecting the user to the OAuth provider, and another for receiving the callback from the provider after authentication. We will access Socialite using the `Socialite` facade:

```
<?php

namespace App\Http\Controllers\Auth;

use Socialite;

class LoginController extends Controller
```

```

{
  /**
   * Redirect the user to the GitHub authentication page.
   *
   * @return \Illuminate\Http\Response
   */
  public function redirectToProvider()
  {
    return Socialite::driver('github')->redirect();
  }

  /**
   * Obtain the user information from GitHub.
   *
   * @return \Illuminate\Http\Response
   */
  public function handleProviderCallback()
  {
    $user = Socialite::driver('github')->user();

    // $user->token;
  }
}

```

The `redirect` method takes care of sending the user to the OAuth provider, while the `user` method will read the incoming request and retrieve the user's information from the provider.

You will need to define routes to your controller methods:

```

Route::get('login/github', 'Auth\LoginController@redirectToProvider');
Route::get('login/github/callback', 'Auth\LoginController@handleProviderCallback');

```

## # Optional Parameters

A number of OAuth providers support optional parameters in the redirect request. To include any optional parameters in the request, call the `with` method with an associative array:

```

return Socialite::driver('google')
->with(['hd' => 'example.com'])
->redirect();

```



When using the `with` method, be careful not to pass any reserved keywords such as `state` or `response_type`.

## # Access Scopes

Before redirecting the user, you may also add additional "scopes" on the request using the `scopes` method. This method will merge all existing scopes with the ones you supply:

```

return Socialite::driver('github')
->scopes(['read:user', 'public_repo'])
->redirect();

```

You can overwrite all existing scopes using the `setScopes` method:

```

return Socialite::driver('github')
->setScopes(['read:user', 'public_repo'])
->redirect();

```

## # Stateless Authentication

The `stateless` method may be used to disable session state verification. This is useful when adding social authentication to an API:

```

return Socialite::driver('google')->stateless()->user();

```

## # Retrieving User Details

Once you have a user instance, you can grab a few more details about the user:

```
$user = Socialite::driver('github')->user();

// OAuth Two Providers
$token = $user->token;
$refreshToken = $user->refreshToken; // not always provided
$expiresIn = $user->expiresIn;

// OAuth One Providers
$token = $user->token;
$tokenSecret = $user->tokenSecret;

// All Providers
$user->getId();
$user->getNickname();
$user->getName();
$user->getEmail();
$user->getAvatar();
```

#### Retrieving User Details From A Token (OAuth2)

If you already have a valid access token for a user, you can retrieve their details using the `userFromToken` method:

```
$user = Socialite::driver('github')->userFromToken($token);
```

#### Retrieving User Details From A Token And Secret (OAuth1)

If you already have a valid pair of token / secret for a user, you can retrieve their details using the `userFromTokenAndSecret` method:

```
$user = Socialite::driver('twitter')->userFromTokenAndSecret($token, $secret);
```

# Become a Laravel Partner

Laravel Partners are elite shops providing top-notch Laravel development and consulting. Each of our partners can help you craft a beautiful, well-architected project.

[Our Partners](#)

# Laravel

## Highlights

[Release Notes](#)  
[Getting Started](#)  
[Routing](#)  
[Blade Templates](#)  
[Authentication](#)  
[Authorization](#)  
[Artisan Console](#)  
[Database](#)  
[Eloquent ORM](#)  
[Testing](#)

## Resources

[Laracasts](#)  
[Laravel News](#)  
[Laracore](#)  
[Laracore EU](#)  
[Laracore AU](#)  
[Jobs](#)  
[Certification](#)  
[Forums](#)

## Partners

[Vehikl](#)  
[Tighten Co.](#)  
[Kirschbaum](#)  
[Byte 5](#)  
[64Robots](#)  
[Cubet](#)  
[DevSquad](#)  
[Ideil](#)  
[Cyber-Duck](#)  
[ABOUT YOU](#)  
[Become A Partner](#)

## Ecosystem

[Vapor](#)  
[Forge](#)  
[Envoyer](#)  
[Horizon](#)  
[Lumen](#)  
[Nova](#)  
[Echo](#)  
[Valet](#)  
[Mix](#)  
[Spark](#)  
[Cashier](#)  
[Homestead](#)  
[Dusk](#)  
[Passport](#)  
[Scout](#)  
[Socialite](#)

Laravel is a web application framework with expressive, elegant syntax. We believe development must be an enjoyable and creative experience to be truly fulfilling. Laravel attempts to take the pain out of development by easing common tasks used in most web projects.

Laravel is a Trademark of Taylor Otwell.  
Copyright © 2011-2019 Laravel LLC.

