Agorakit Documentation

Release 1.0

Philippe Jadin

Jun 26, 2019

Installation

Agorakit is a web based open source groupware for collectives. By creating collaborative groups, people can discuss, organize events, store files and keep everyone updated when needed. Agorakit is a forum, agenda, file manager and email notifier.

Manage communication, decision making, membership, files and events. Flexible email notifications per group, per user preferences. Most of the time no admin is involved in the process as we try to keep it as horizontal as possible.

CHAPTER 1

Installation

Before installing Agorakit, if you just want to see if it fits your requirements, you might just create a free account on https://app.agorakit.org. This instance is free for citizen-activists and for evaluation purposes.

If you want managed hosting of a private Agorakit instance, contact us at info [at] agorakit.org

On the other hand if you want to install it on your own server, here are the required steps :

1.1 Requirements

You need a good web hosting provider that provides the following :

- **php >= 7.1.3 with the following extensions :**
  - OpenSSL PHP Extension
  - PDO PHP Extension
  - Mbstring PHP Extension
  - Tokenizer PHP Extension
  - XML PHP Extension
  - Ctype PHP Extension
  - JSON PHP Extension
  - BCMath PHP Extension
- Mysql or MariaDb
- Composer
- Git
- the ability to run cron jobs

All those features together are hard to find, so people are obliged to use a VPS and setup everything themselves. This is a riskier proposal if you don't know how it works.

We have been very successful with https://www.alwaysdata.com shared hosting (By the way they host free of charge the free instance of Agorakit at https://app.agorakit.org).

1.2 Installation

Currently, you need to know how to install a Laravel application using the command line. This is perfectly standard and documented here : https://laravel.com/docs/master/installation.

Clone the repository:

```
$ git clone https://github.com/philippejadin/agorakit.git
```

Create and edit the configuration file from the example provided:

```
$ cp .env.example .env
$ nano .env
```

You need to set at least your database credentials & site name. Check that your database exists and is reachable with those credentials.

Here is a description of every setting in the .env file:

```
APP_ENV=local  // local or production
APP_DEBUG=true // show the debugbar and extended errors or not
APP_KEY=SomeRandomString // will be auto generated
APP_NAME='Agorakit' // name of your application
APP_URL=http://locahost // base url
APP_LOG=daily // log rotation
APP_DEFAULT_LOCALE=en // default locale when not detected from user browser

DB_HOST=localhost // host of your mysql server
DB_DATABASE=agorakit // db name of your sql DB
DB_USERNAME=root // login of mysql
DB_PASSWORD= // password of mysql

CACHE_DRIVER=file // driver to use for caching
SESSION_DRIVER=file // driver to use for storing sessions
QUEUE_DRIVER=sync // driver to use for queues

MAIL_DRIVER=smtp // driver to use for sending emails. Use mail to use php built-in␣
↪mail function
MAIL_HOST=mailtrap.io // hostname if you use smtp for sending mails
MAIL_PORT=2525 // port if you use smtp for sending mails
MAIL_USERNAME=null // login if you use smtp for sending mails
MAIL_PASSWORD=null // password if you use smtp for sending mails
MAIL_ENCRYPTION=null // encryption if you use smtp for sending mails

MAIL_FROM=admin@localhost // from email adress used when sending admin emails
MAIL_FROM_NAME=Agorakit // name of sender of admin emails
MAIL_NOREPLY=noreply@localhost // no reply adress for service messages

MAPBOX_TOKEN=null // Create a Mapbox account and generate a token to enable␣
↪geolocalisation and display maps
```

Download all the packages needed:

```
$ composer install
```

Generate a key:

```
$ php artisan key:generate
```

Migrate (create all tables in) the database:

```
$ php artisan migrate
```

(Optional) Create sample content the database:

```
$ php artisan db:seed
```

Don't do this last step for a production install since it will create an admin user and dummy groups and content.

1.3 Setup your web server

Then setup your web server to serve the /public directory. This is very important, since you don't want to expose the rest of the directories (for example you DON'T want to expose your .env file!)

1.4 Setup a cron job

Follow Laravel cron documentation here : https://laravel.com/docs/master/scheduling The cron jobs are used to send group summaries at a fixed interval, for the inbound email handler and for various database interactions.

1.5 Setup geolocalisation and mapping

Create an account at Mapbox.com and create an api token. Then fill this api token in your .env file. With this, you will get geocoding and maps. We switched from Google maps to Mapbox because Google Maps now requires a credit card even for the free tier. Mapbox free tier is probably enough for your use (50k displays / month at the time of writing)

1.6 Setup inbound emails

This additional step allows you to have one mailbox for each group so members can post by email.

You need an email address on server with imap. It must either be a catch all on a subdomain (or even on a domain) or a server supporting "+" addressing (gmail for example allows this).

Let's say you installed Agorakit on agora.example.org Create a catchall mailbox on *.@agora.example.org

Then go to admin settings and fill the form there (end of the page).

You need to fill server & login & password

Then you need to fill prefix and suffix. Two cases there :

For a catch all there is no prefix. The suffix in the above example would be @agora.example.org . This will create emails like group-slug@agora.example.storing

On the other hand if you use "+" addressing (a gmail box for instance, let's call it agorakit@gmail.com),

- prefix will be agorakit+
- suffix will be @gmail.com

Wich create emails like agorakit+group-slug@gmail.com

If you enable inbound email, the mailbox will be automatically checked and processed email will be put in a "processed" folder under INBOX. Failed emails will be similarly put a "Failed" folder under INBOX for inspection.

CHAPTER 2

Update your installation

It is important to keep an up to date installation of Agorakit We try to keep the master branch always in a good, safe, and working condition (this is called a "rolling release" model).

That means that tests passes and that you get the latest features directly from the master branch.

2.1 Make a backup

Make a backup of your SQL database in case something goes wrong.

2.2 Update script for automated updates

There is a helper script that does the update for you :: $ ./update

Beware that the script will migrate your database without asking for confirmation. Always make a backup of the database just in case something goes wrong.

2.3 Proceed with the update manually

You can at anytime do this to update your install

```
$ php artisan down
$ git pull
$ composer install
$ php artisan migrate
$ php artisan up
```

2.4 If something goes wrong

Restore your database backup and git checkout a previous (working) version. Then re-run composer install.

Contact me if an update fails (it never happened so this kind of failure is highly interesting information for the project).

CHAPTER 3

Quick start after installation

3.1 Discover your installation

Open your browser and go to the url of your web server. If using php's built-in web server it will be : http://localhost:8000

3.2 Register account

Register a first user account : it will be you, and you will be super admin, yeah! Note that you need to have email sending working in order to validate your account.

3.3 Welcome group

Create one or more groups. A welcome and a test group are often useful.

3.4 Introduction text on the homepage

Set an intro text on the homepage for newcomers : go to the admin page from your user profile dropdown. Using the editor you can add external images and whatever you want.

3.5 Invite people

Invite people to one or more groups using the invite feature of each group. You can also add people directly if the already registered on your install. This is particularly useful if people have a hard time to confirm their email address and membership (trust me it happens).

3.6 Creating discussions

Each group allows one to create discussions. Within discussions and comments, you can use a special syntax to reference existing files and discussions, and mention users. Just start typing @ to mention someone (autocomplete appears) use f: to link to a file, and d: to link to a discussion.

You may also directly attach a file to a discussion. Use the "Browse..." button below the discussion form to upload a file from your computer, attach it to the group, and mention it in the discussion in one go.

3.7 Creating actions

Each group has a calendar where you can add actions/events. You just need to set a start date and time, and an optional duration. If no duration is set, it is assumed to be 1 hour.

3.8 Creating files

You can add file to groups directly in the files tab. Just upload one or more files, add some tags for organization. As explained above, you can also directly attach files to comments and discussions.

3.9 Using tags

Tags (aka Labels) are very important to organize your content. You can tag anything in Agorakit (files, users, discussions, groups, actions). You can either choose to use freeform tagging (anyone can choose any tag) or restrict the tags you cant to use in each group (more like github labels for example).

In the later case, a group admin may decide which tags are allowed in the group. It is often a good idea to restrict available tags : "less is more" and "everyone has a different way to organize content, so let's stick to one way of doing it".

Group admins may also define tag colors. The colors are shared among groups.

The allowed tags can be defined in the "Admin" tab of the group, under "Tags".

Once content is tagged, it can be found as a simple overview page, showing all tags and showing all content attached to a particular tag. Great to have a big overview of your groups.

3.10 Last step : profit !

(or in some cases, revolution!)

CHAPTER 4

Developing with Agorakit

You want to contribute to the project? Great!

First follow the installation instructions including the creation of sample content using

```
$ php artisan db:seed
```

4.1 Setup your environment, option 1 : Using php's built in server

If you want to start a local server for development:

```
$ php artisan serve
```

The install will be available to 127.0.0.1:8000

4.2 Setup your environment, option 2 : Using Vagrant

To use vagrant, you need to install : [Virtual Box](https://www.virtualbox.org/wiki/Downloads) and [Vagrant](https://www.virtualbox.org/wiki/Downloads).

After in the root of your project:

```
$ vagrant up
$ vagrant ssh
$ cd agorakit
$ composer install
$ php artisan key:generate
$ php artisan migrate
$ php artisan db:seed (if you want the db filled with fake infos)
```

Then you will have access to the project using :

http://192.168.10.12/

Modifying your hosts file

You must add the "domains" for your Nginx sites to the hosts file on your machine. The hosts file will redirect requests for your Homestead sites into your Homestead machine. On Mac and Linux, this file is located at /etc/hosts. On Windows, it is located at C:WindowsSystem32driversetchosts.

CHAPTER 5

Working on design and css

Install nodejs and npm (current version of nodejs is 10)

Then in the root of the project run:: $ npm install

You will be able to have auto updated browser when you change a file by running:: $ npm run watch

When you are done, run:: $ npm run prod

To generate production ready css and js files.

CHAPTER 6

Testing your code

Agorakit is tested using the Laravel testing framework.

In order to test, you need to have an existing testing database. Just create an additional empty DB, for instance agorakit_testing and check in the phpunit.xml file that everything matches.

Before comiting code, you should either write more tests (in this case you deserve a cookie). Or at least check that you didn't break anything by simply typing:

```
./vendor/phpunit/phpunit/phpunit
```

. . . in the root of your project.

No error should appear (provided that you have everything correctly set up.

We use travis ci to run all those tests on commit so it will be done automatically for you at some point :-)

CHAPTER 7

Writing tests

Don't hesitate to write tests. We favor well defined tasks an end user would really accomplish, like registering, creating an account, posting, uploading, etc… It has served us very well in the past to spot errors and it really mirrors real use cases. Although we are open to other kind of tests as well…

CHAPTER 8

Provide external authentification to Agorakit

You can provide facebook, google, twitter and/or github authentication to your users.

This feature is a work in progress, help is highly welcome.

To do so, get a client id, a client secret from each provider you want to use. Those are a string of characters you need to put in your .env file

Agorakit will automatically put the links on the login and registration page for each provider when [PROVIDER]_ID is set

- For Twitter go to : https://apps.twitter.com/app/new
- For Facebook, go to: https://developers.facebook.com/apps
- For Github go to https://github.com/settings/applications/new
- For Google go to : http://console.developers.google.com/

It's a bit of a pain to find where to get the client_id and client_key, so good luck :-)

You also need to set [PROVIDER]_URL to http[s]://[yourdomain]/auth/[provider]/callback for each provider, in your .env file