COMRADES

Configuration & Set-up Manual for the COMRADES Platform

In partnership with:

Ushahidi

KMi Knowledge Media Institute

The University Of Sheffield.

TU Delft

Gov2u

COMRADES Configuration Manual and User Guide

This manual was published in January 2019. Cover photo shows members of Uchaguzi testing aspects of the COMRADES platform while monitoring the 2017 Kenyan General Election.

What is COMRADES?

COMRADES (Collective Platform for Community Resilience and Social Innovation during Crises) is a European Union-funded project that aims to empower communities with intelligent socio-technical solutions to help them reconnect, respond to, and recover from crisis situations. The project received funding from the Horizon 2020 Research and Innovation programme under grant agreement #687847.

This manual outlines how and where to access the open-source code that runs the COMRADES platform and how to configure it. For more information on the project itself, you can visit the COMRADES project website at https://www.comrades-project.eu/ or visit the factsheet for the project on the European Commission website here: https://cordis.europa.eu/project/rcn/198819/factsheet/en.

Location & Access of Computer Code

As technical partner for the COMRADES project, Ushahidi has created a COMRADES deployment for use by the consortium partners that is accessible at: https://comrades.ushahidi.com. A "deployment" in Ushahidi's context is a hosted instance of the Platform. While this deployment is viewable by the public, only those with specific permissions can access, manage, and publish data within the deployment.

The code base for the COMRADES platform and the deployment is contained in the open source, online, version code repository known as Github[1]. Various aspects of the code are contained in "repositories." Very briefly, the COMRADES platform is composed of two core modules: a PHP based API system and a Nodejs Web Client. For the purposes of this project, two additional Github repositories have been created which contain code for connecting the platform to a variety of external service for use by consortium members:

- https://github.com/ushahidi/platform-client-comrades contains code for the graphical user interface (GUI) of the platform.
- https://github.com/ushahidi/platform-comrades contains the primary source code for the functionality of the platform.
- https://github.com/ushahidi/comrades-service-proxy contains code that fetches data from the services developed for COMRADES, such as YODIE, CREES, and EMINA, which are explained further in this deliverable.
- https://github.com/ushahidi/platform-facebook-bot contains code for a chatbot that can be deployed in Facebook messenger and connected to the COMRADES platform.

These repositories store all source code related to the COMRADES platform. Each repository contains instructions for downloading and configuring the code within that repository. Additionally, a copy of this configuration manual and user guide is contained within each Github repository. This manual explains how to all of the code should be configured to work together as a complete platform.

[1] https://github.com

Platform Setup

Overview

This section describes basic setup steps required to fully setup a COMRADES Platform deployment integrated with YODIE and CREES, which are web services developed to automatically annotate and categorize incoming data.

Installing

The installation procedure will vary depending on your setup, but the requirements for installing on a server are:

- A web server supporting PHP. This can be apache2, nginx or a hosting provider
- PHP invokable from command line
- The following PHP modules installed: curl, json, mcrypt, mysqli, pdo, pdo_mysql, imap and gd
- A MySQL database server

These instructions assume that you know how to create a database in your MySQL server and obtain user credentials with access to such database.

The instructions and example commands are written specifically for Debian Linux or a derivative of it (Ubuntu, Mint, etc). You may have to adjust some things if you are installing on a different flavour of Linux, or a different OS.

Apache 2 with mod_php

1. Ensure `mod_rewrite` is installed and enabled in your apache server.
2. Copy into your document root the contents of the `html/` folder after unzipping the ushahidi-platform-release-vX.Y.Z.tar.gz bundle file.
3. The `dist/` folder contains the suggested configurations for the virtual host (`apache-vhost.conf`). The configs are quite default, you just need to ensure that there is an `AllowOverride` directive set to `All` for your document root (where the app has been unzipped).
4. Create a `platform/.env` file with your database credentials, such as:
   ```
   DB_HOST=<address of your MySQL server>
   DB_NAME=<name of the database in your server>
   DB_USER=<user to connect to the database>
   DB_PASS=<password to connect to the database>
   DB_TYPE=MySQLi
   ```
5. Run the database migrations, execute this command from the platform folder:
   `./bin/phinx migrate -c application/phinx.php`
6. Ensure that the folders `logs`, `cache` and `media/uploads` under `platform/application` are all owned by the user that the web server is running as. i.e. in

Debian derived Linux distributions, this user is `www-data`, belonging to group `www-data`, so you would run:

```
chown -R www-data:www-data
platform/application/{logs,cache,media/uploads}
```

7. Set up the cron jobs for tasks like receiving reports and sending e-mail messages. You'll need to know again which user your web server is running as. We'll assume the Debian standard `www-data` here.
   Run the command `crontab -u www-data -e` and ensure the following lines are present in the crontab:

```
MAILTO=<your email address for system alerts>
*/5 * * * * cd <your document root>/platform && ./bin/ushahidi
dataprovider outgoing >> /dev/null
*/5 * * * * cd <your document root>/platform && ./bin/ushahidi
dataprovider incoming >> /dev/null
*/5 * * * * cd <your document root>/platform && ./bin/ushahidi
savedsearch >> /dev/null
*/5 * * * * cd <your document root>/platform && ./bin/ushahidi
notification queue >> /dev/null
*/5 * * * * cd <your document root>/platform && ./bin/ushahidi
webhook send >> /dev/null
```

8. Restart your apache web server and access your virtual host. You should see your website and be able to login with the credentials user name admin and password admin

Make sure to change the credentials. Specially if the website is exposed to be accessed by anyone other than you.

nginx with php-fpm

The procedure is pretty similar to the one detailed for apache above, with the following exceptions.

1. `mod_rewrite` is specific for Apache, in nginx the module is named `ngx_http_rewrite_module`. It's usually included and enabled.
2. instead of configuring Apache, you would need to configure nginx. For configuring nginx see the example `nginx-site.conf` in the `dist` folder. You would usually drop this file in a place where it's included from the main configuration file. It assumes php-fpm is listening in port 9000 of localhost.
3. The default php-fpm configuration should work. Most importantly, you need to ensure the `listen` directive matches the `fastcgi_pass` directive in the nginx host configuration file.
4. Once you are done, restart both your nginx and php-fpm services.

Shared hosting (Cpanel, Dreamhost, Bluehost, etc.)

In general, the instructions for apache can be taken as a guideline. Each shared hosting provider comes with their own set of particularities, so we can only provide general directions here. In all cases, you'll need to ensure that:

1. Decompress the release file and place the contents of the `html` folder in the webroot of your shared hosting domain or subdomain.
2. Create a database for your website and write the access details in the `.env` file (as per step 4 of Apache 2 instructions)
3. You have command line access (SSH) in order to run the `phinx` database migration utility in step 5 of Apache 2 instructions.
4. A URL rewriting mechanism has to be in place so that:
    a. Requests to `/platform/api/v3/*` are to be forwarded to the `index.php` script `inside/platform/httpdocs`.
    b. When invoking that script, the `api/v3/*` part of the url should be passed to the script into the a `$_SERVER` or environment variable.
    c. If your host uses Apache and supports `.htaccess` files, most of this should be taken care of for you.

Installing for Development

Installing the API

Getting the API code

First, you will need a copy of the source code, which lives in our Github repository:
git clone git@github.com:ushahidi/platform-comrades.git If you're getting set up for development, you might want to fork the repository first. Once you have the code, the next step is to prepare a web server.

Prerequisites

- Vagrant (Windows users may be required to Enable VT-X (Intel Virtualization Technology) in the computer's bios settings, disable Hyper-V on program and features page in the control panel, and install the VirtualBox Extension Pack)
- VirtualBox (Windows users may be required to Enable VT-X (Intel Virtualization Technology) in the computer's bios settings, disable Hyper-V on program and features page in the control panel, and install the VirtualBox Extension Pack)
- Composer
- PHP >= 5.6

Installing

First up we need to install the PHP dependencies

- `cd platform`
- `composer install`

If you get an error about "The requested PHP extension ... is missing from your system" you might need to run composer install --ignore-platform-reqs instead. You generally won't need all the PHP extensions on your host machine as they're installed in the vagrant box instead.

Then you can bring up the vagrant server and provision it:

```
vagrant up && vagrant provision
```

Our vagrant box is built on Laravel's Homestead[2], a pre-packaged Vagrant box that provides you with a pre-built development environment. Homestead includes the Nginx web server, PHP 7.1, MySQL, Postgres, Redis, Memcached, Node, and all of the other goodies you might need.

If you see errors such as "`Vagrant was unable to mount VirtualBox shared folders...`", try upgrading VirtualBox or edit Homestead.yaml and change the folders to NFS as shown below, then re-run "vagrant" up.

```
    -
        map: "./"
        to: /vagrant
        type: "nfs"
    -
        map: "./"
        to: /home/vagrant/Code/platform-api
        type: "nfs"
```

At this point you should have a running web server but your deployment isn't set up yet. We still need to configure the database and run migrations.

```
cp .env.example .env
composer migrate
```

Go to 192.168.33.110 to check the API is up and running. You should see some JSON with an API version, endpoints and user info.

Installing the client

Getting the client code

First, you will need a copy of the source code, which lives in our Github repository: git clone git@github.com:ushahidi/platform-client-comrades.git. The latest install instructions for the client are always in the README. If you have any trouble check those instructions first.

Client dependencies

First you'll need nodejs or io.js installed, npm takes care of the rest of our dependencies.

- nodejs >= v4.0

Install, build and run a local dev server

[2] See https://laravel.com/docs/5.4/homestead#per-project-installation

1. Clone the repo
   `git clone https://github.com/ushahidi/platform-client.git`
   Note: if you're getting set up for development, you might want to fork the repository first.
2. Navigate to project root
   `cd platform-client`
3. Install Build Requirements
   `npm install -g gulp`
4. Install Packages
   `npm install`
5. Set up build options. Create a `.env` file, you'll need to point `BACKEND_URL` at an instance of the platform api (If you followed the vagrant instructions above that'll be: http://192.168.33.110)
   `BACKEND_URL=http://192.168.33.110`
6. Run gulp
   `gulp`
7. You should now have a local development server running on `http://localhost:3000/`

Logging In

1. The default install creates a user admin with password “admin”. Once logged in this user can create further user accounts or give others admin permissions too.
2. In the bottom left hand side of the screen you’ll see “Log In” and “Sign Up”. “Click on Log In.” See figure 1.

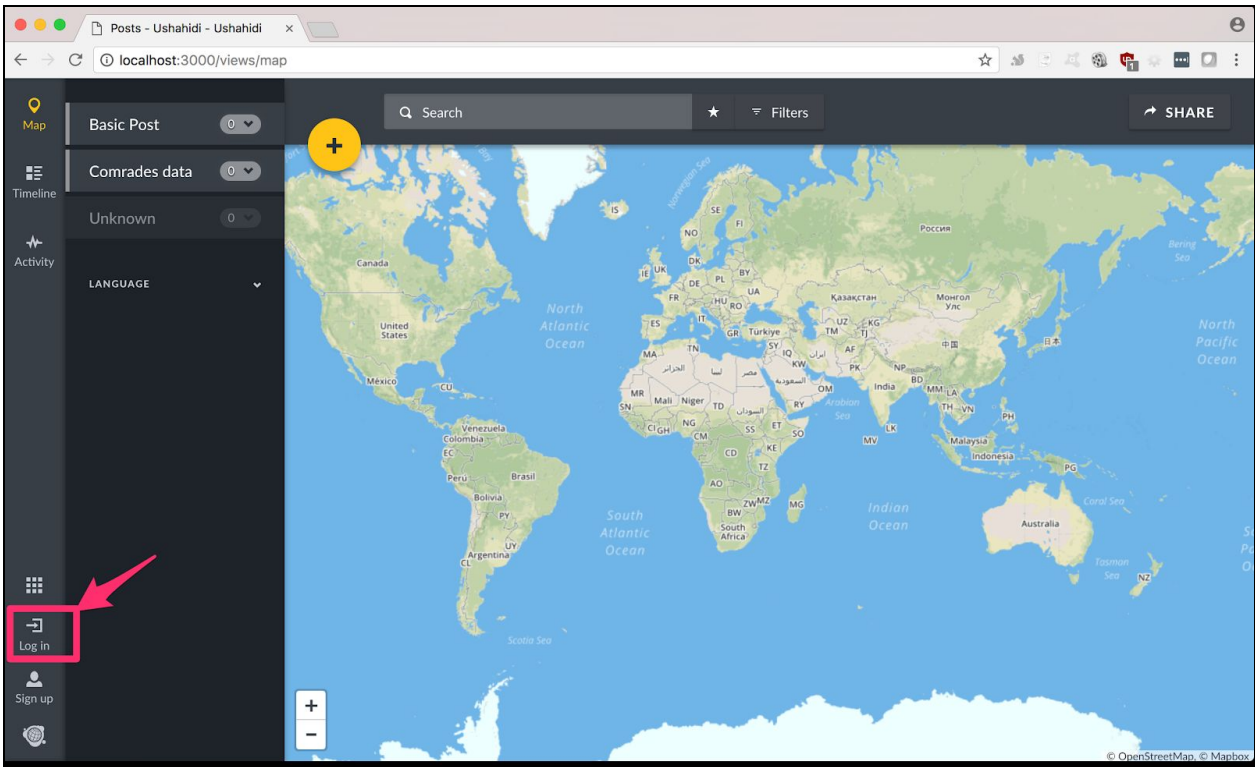


Figure 1. Default view with Log In icon highlighted.

3. A new dialog window should appear in the centre of the screen. For “Email Address”, enter the email address associated with your account. In case you have lost or forgotten your password, we have a password reset facility. Simply click “Forgot your password”, beneath the password field, and a Password Reset Link will be sent to your email address.
4. Once you have entered your Email Address and Password, click on “Log In”, the dialog will disappear and your account will load.

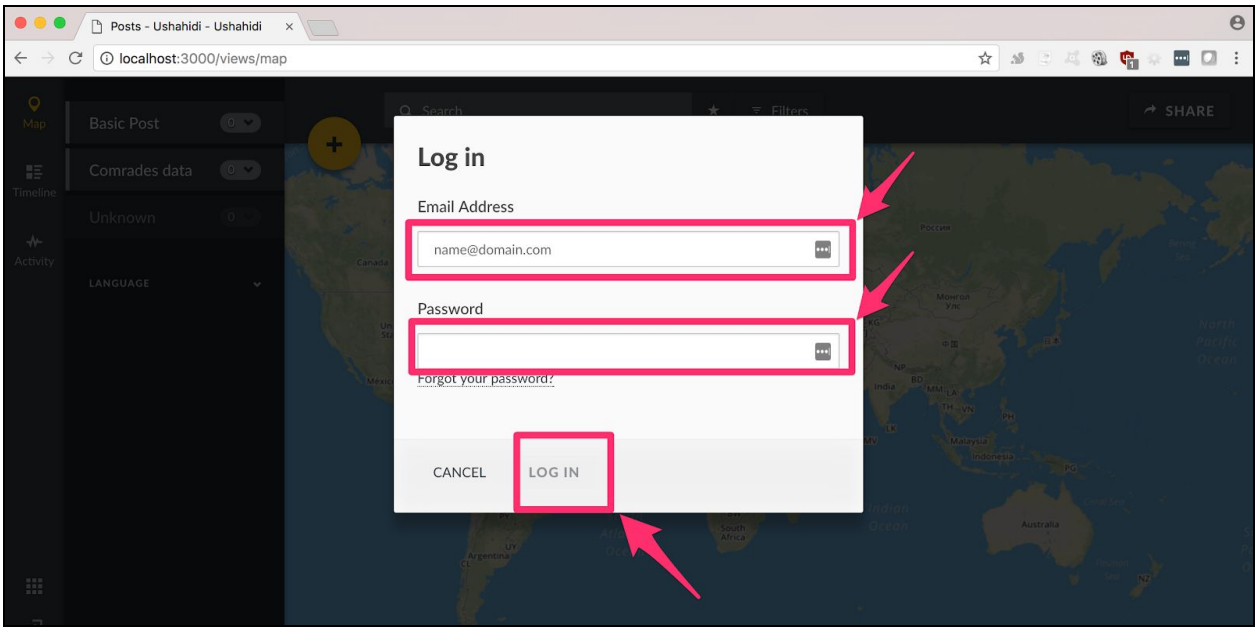


Figure 2. Login dialogue window.

Settings Menu

1. The settings section is accessed via the menu item on the left hand side of the screen. The settings menu is symbolised by a gear icon, .

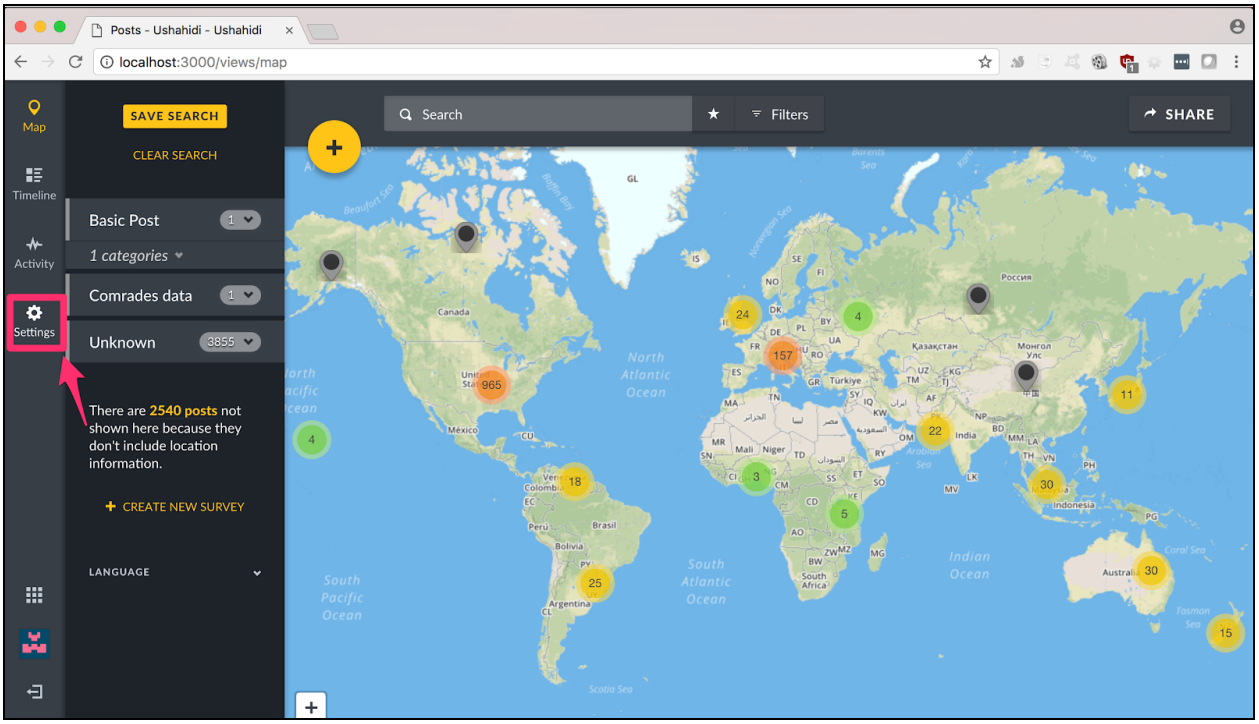


Figure 3. Accessing the "Settings" menu, highlighted.

Categories

1. First we will configure categories that we need for our COMRADES Platform.
2. Within in the Settings Section you will see a list of configuration options.
3. Click on “Categories” marked in figure 4.

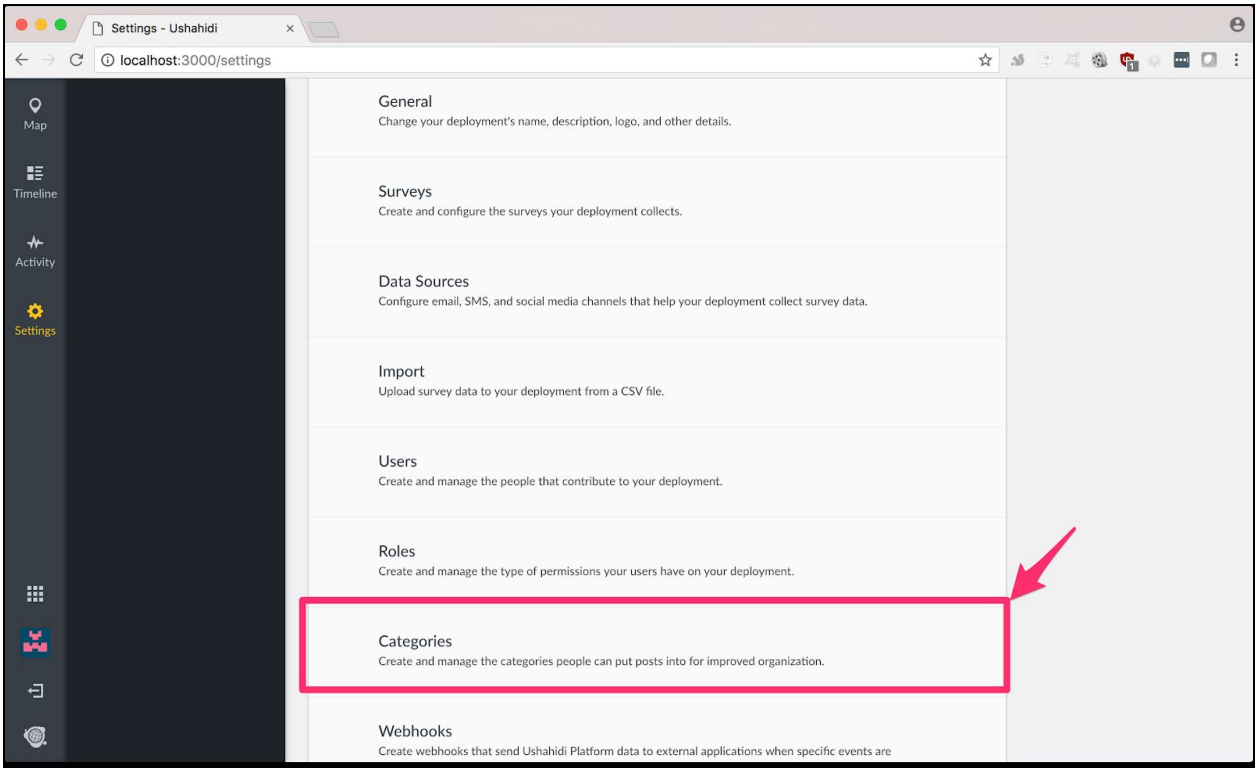


Figure 4. Accessing the “Categories” menu, highlighted.

4. Within the Categories section, initially you will see that there were “No categories found”. We will now add some.
5. Towards the top left of the Categories List page you will see a round yellow button with a plus inside it.

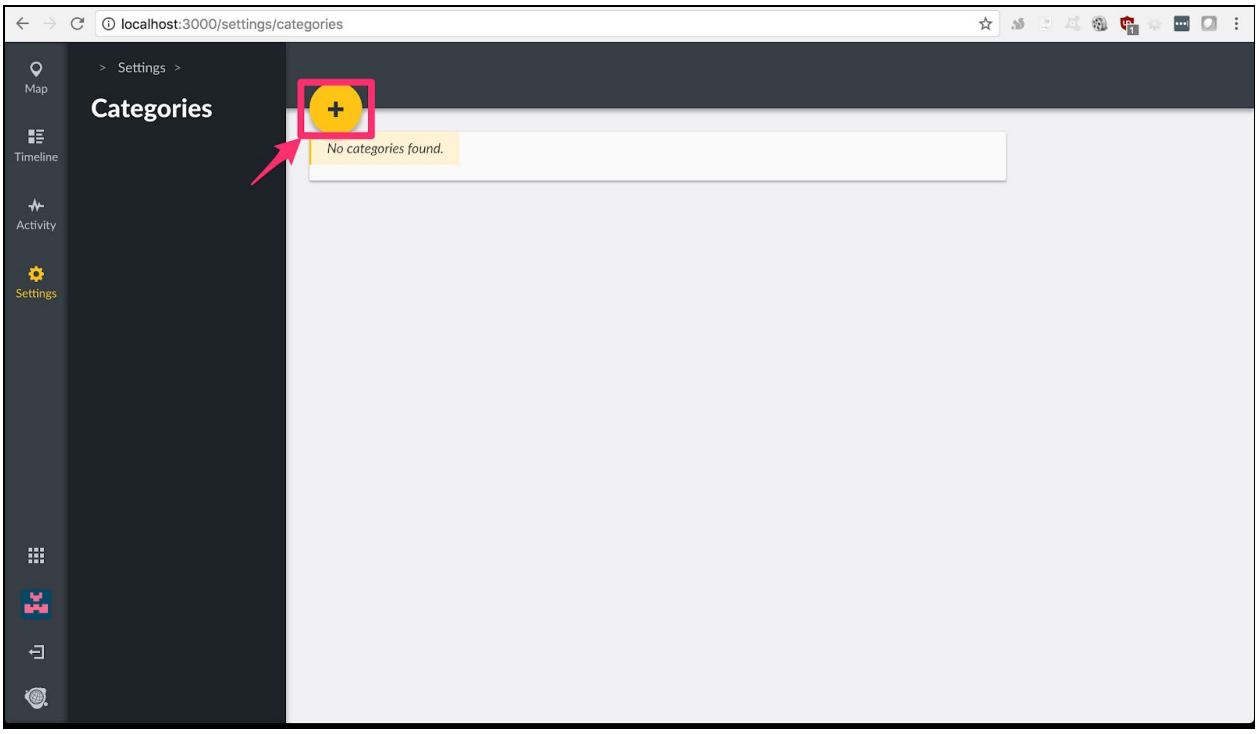
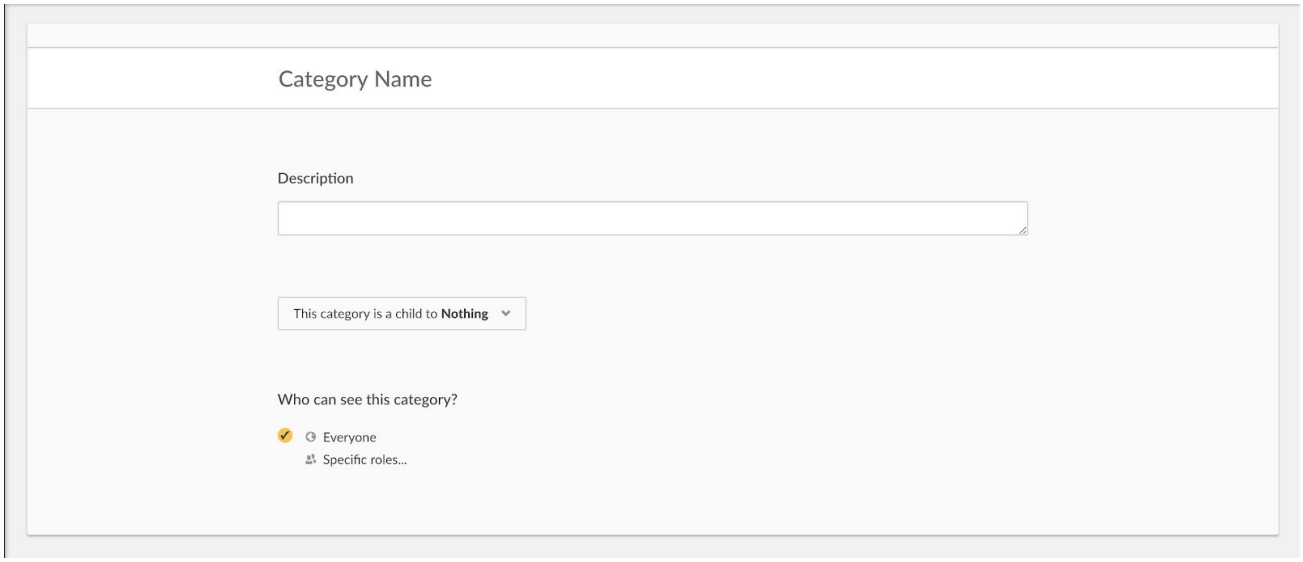


Figure 5. Adding Categories

6. On the Category Edit page (see screenshot) fill in the Category Name, Description, Parent, and Permissions



a. Name: add a name for your category, this will be the first thing a user sees of that category (and often the only part as description is only shown in some scenarios)
b. Description: a description of what your category is about.
c. Parent: the "this category is a child to " dropdown can be clicked to see a list of other categories that can be used as a parent to this one. Adding a parent is optional.

    d. Permissions: "who can see this category" , select a specific role or keep the "everyone" option for it to be visible for everyone
7. Click "Save" and your category will be created

The following list contains all the categories that need to be added to a deployment using the comrades service proxy with CREES, EMINA and YODIE. For more information about the Crisis Event Extraction Service (CREES) from the Open University, see documentation on the CREES website here: https://evhart.github.io/crees/ or reference Burel et al. (2017). For information on Yet Another Open Data Information Extraction System (YODIE) from the University of Sheffield see website here: https://gate.ac.uk/applications/yodie.html or reference Gorrell et al (2015). For information on Emergent Informativeness and Actionability (EMINA) service from the University of Sheffield see the code base here: https://github.com/GateNLP/emina or reference Derczynski et al. (2018).

Categories required by CREES

Category: Event Related
Subcategories:

- `non-related`
- `related`

Category: Info Type
Subcategories:

- `affected_individuals`
- `caution_and_advice`
- `donations_and_volunteering`
- `infrastructure_and_utilities`
- `not_applicable`
- `not_labeled`
- `other_useful_information`
- `sympathy_and_support`

Category: Event Type

Subcategories:

- `bombings`
- `collapse`
- `crash`
- `derailment`
- `earthquake`
- `explosion`
- `fire`
- `floods`
- `haze`

- meteorite
- none
- shootings
- typhoon
- wildfire

Category: Event Related
Subcategories:

- non-related
- related

Categories required by Yodie

Yodie does not require any predefined categories.

Categories required by Actionability service (EMINA)

Category: You don't need to group the results, but we recommend creating a parent category (named "Actionability" or any other unique name) to see the different results in a more organized format, as the actionability service can return more than 1 category.
Subcategories:

- Opinion or individual message (e.g. thoughts and prayers)
- General reporting about the rescue effort (from the media or the public)
- Damage to infrastructure, livelihoods etc.
- Changes in accessibility - limits to access, or other changes in how much transport can get through.
- Mention of a group that is responding (e.g. volunteers, military, etc.)
- A specific resource, where the kind of need is given explicitly.
- Mention by name of the geographic areas affected
- Threats to the general crisis response. Weather warnings, fires, military action etc.
- Changes in the local environment (weather, hazards, etc.); e.g., a storm is intensifying

Category: Informative
Category: Not informative

Categories required by Veracity service (EMINA)

Category: You don't need to group the results, but we recommend creating a parent category (named "Veracity" or any other unique name) to see the different results in a more organized format and understand where the category is coming from.
Subcategories:

- Rumor
- Not a rumor
- Unverified

Create Survey

1. Next we must create a survey(s) that will be responsible for giving structure to the data we will be dealing with.
2. Return to the Settings Menu, click on “Surveys”

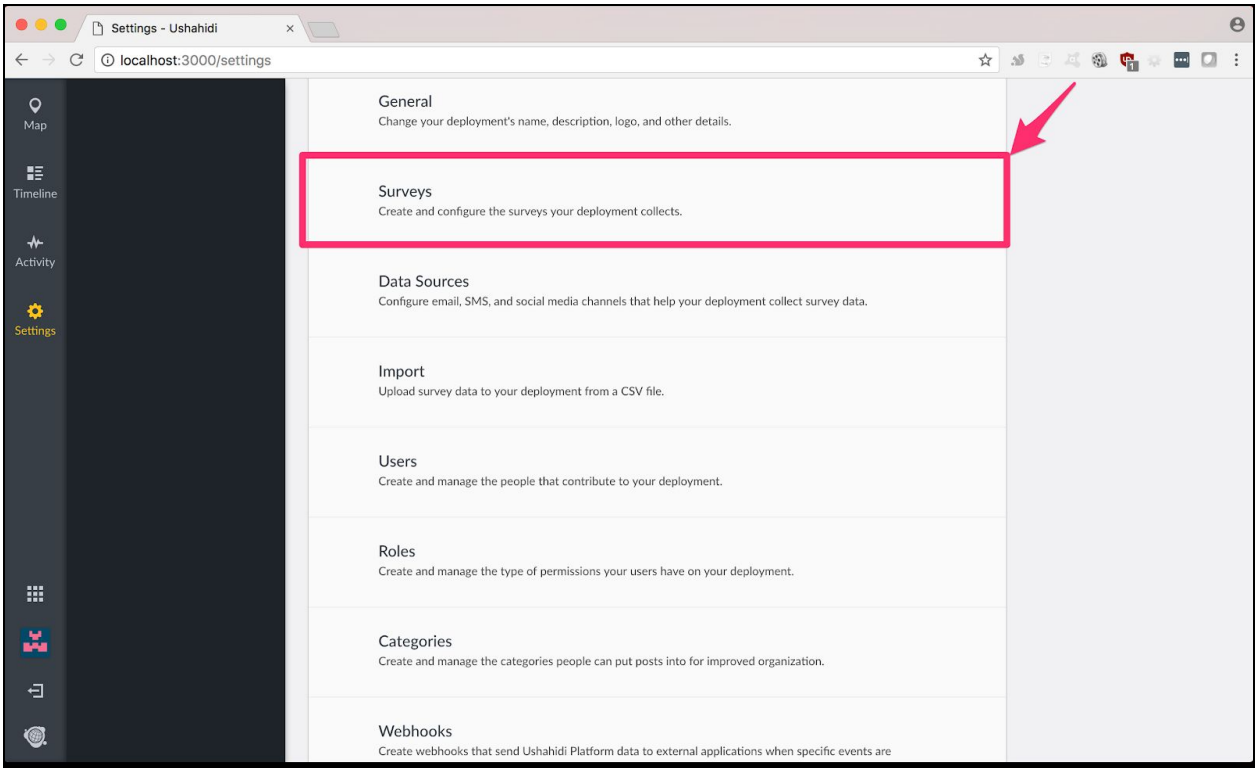


Figure 6. Locating the survey configuration menu

3. On the Survey List page, you will see some existing surveys from this page it is possible to duplicate or delete existing surveys, via the ellipsis(...) menu at the right hand end of each Survey item.
4. To create a new Survey, click the add new survey button, the round yellow button with a plus in the centre

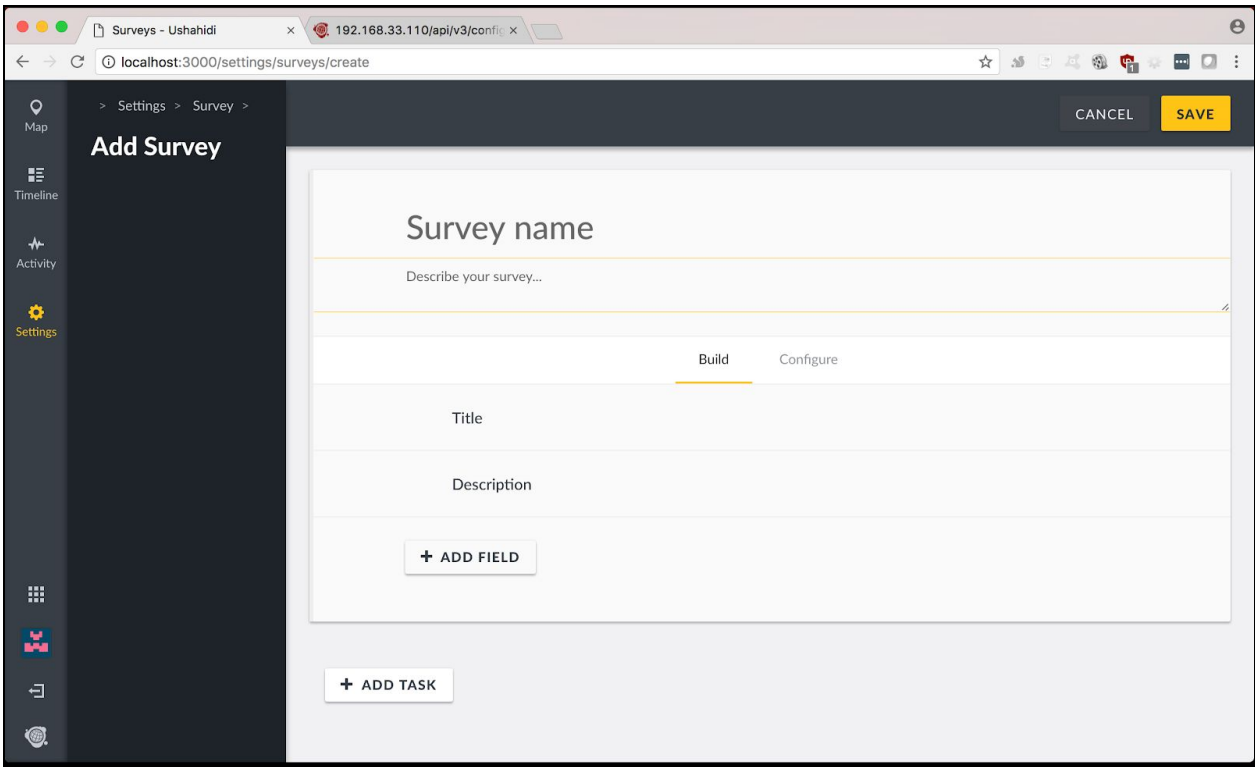


Figure 6. A default survey

5. The above shows a blank new Survey, most fields such as Name and Description are self explanatory

6. Enter a Name for your Survey in the field marked “Survey name” and enter a description for the survey in the field marked “Describe your survey.”
7. Once you have added the basic details, you can customise your Survey to better fit the data you wish to handle.

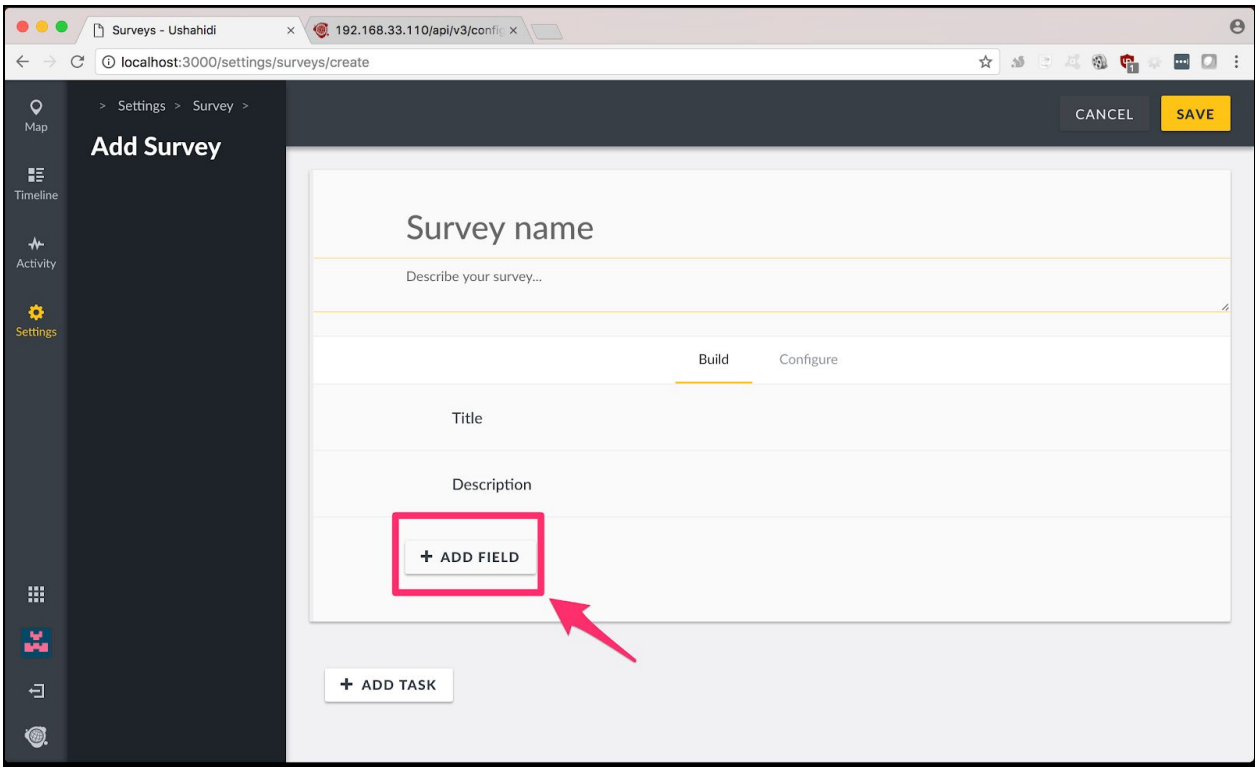


Figure 7. Adding additional fields for information in your survey.

8. Each Survey comes with two basic fields, Title and Description these fields are immutable. To add additional fields click the "+Add Field" button, as shown in the image above
9. After clicking "+Add Field", a dialog as shown below should appear in the centre of the screen

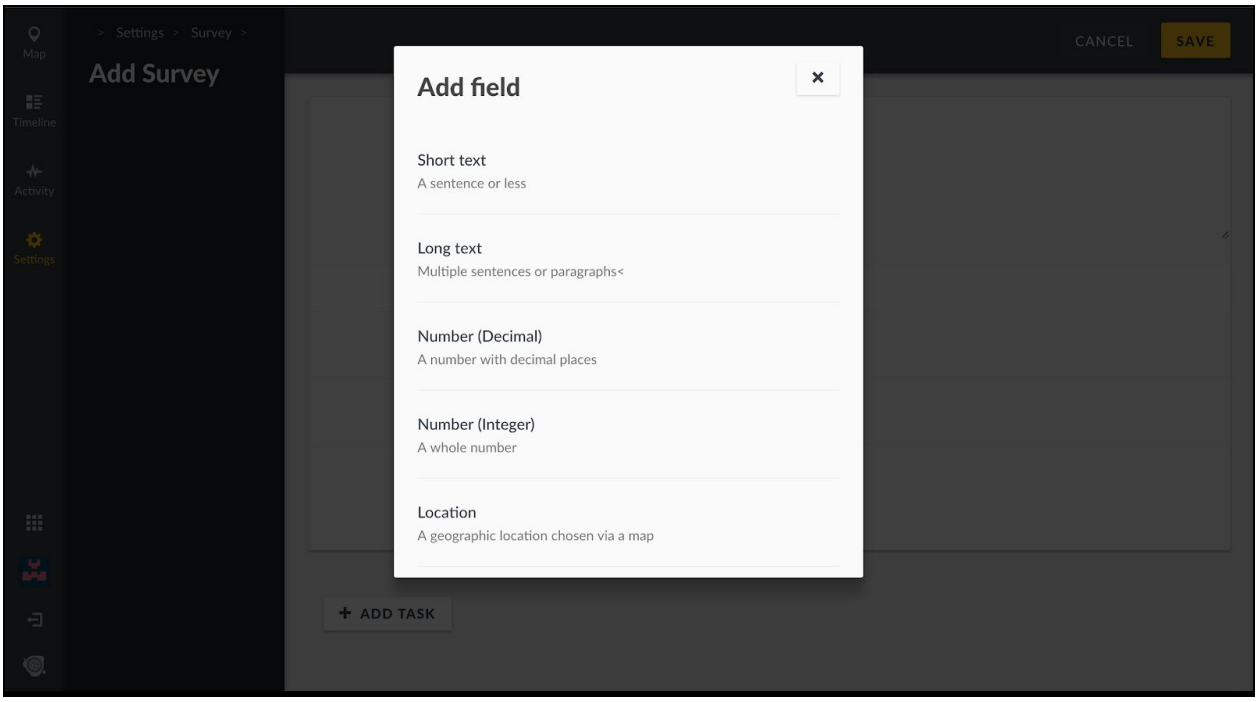


Figure 8. Choose different field types from the menu.

10. The list of possible fields are as follows:
    - **Short Text** - a sentence or less of text.
    - **Long Text** - multiple sentences of text.
    - **Number (Decimal)** - a number with decimal places.
    - **Number (Integer)** - a whole number.
    - **Location** - A geographic location chosen via a map.
    - **Date** - A date with year, month, day.
    - **Date & time** - A date with time in hours and minutes.
    - **Select** - a dropdown list of items
    - **Radio Button(s)** - Allows the user to choose only one of a predefined set of options.
    - **Checkbox(es)** - Allows the user to choose one or more of a predefined set of options
    - **Related Post** - Create a relation between different surveys within a deployment.
    - **Image** - Allows image(s) to be uploaded to the post.
    - **Embed video** - Allows YouTube or Vimeo videos to be embedded in a post.
    - **Markdown** - Allows the user of markdown syntax to style the field content.
    - **Categories** - Create and manage the categories people can put posts into for improved organization.
11. There is only one requisite field that we must add to Surveys to support YODIE integration, it is a Markdown field. This field can contain markdown syntax and will display the content styled via the markdown. The reason we need a markdown field is that the data returned from the COMRADES service proxy for Yodie requests is returned in markdown format. As we will cover in more detail later in the Webhooks section it is necessary to have a results field to store the result of the Yodie annotation.

12. In order to add a Markdown field, select “Markdown” from the list of field types, a new dialog will open showing options for the field.

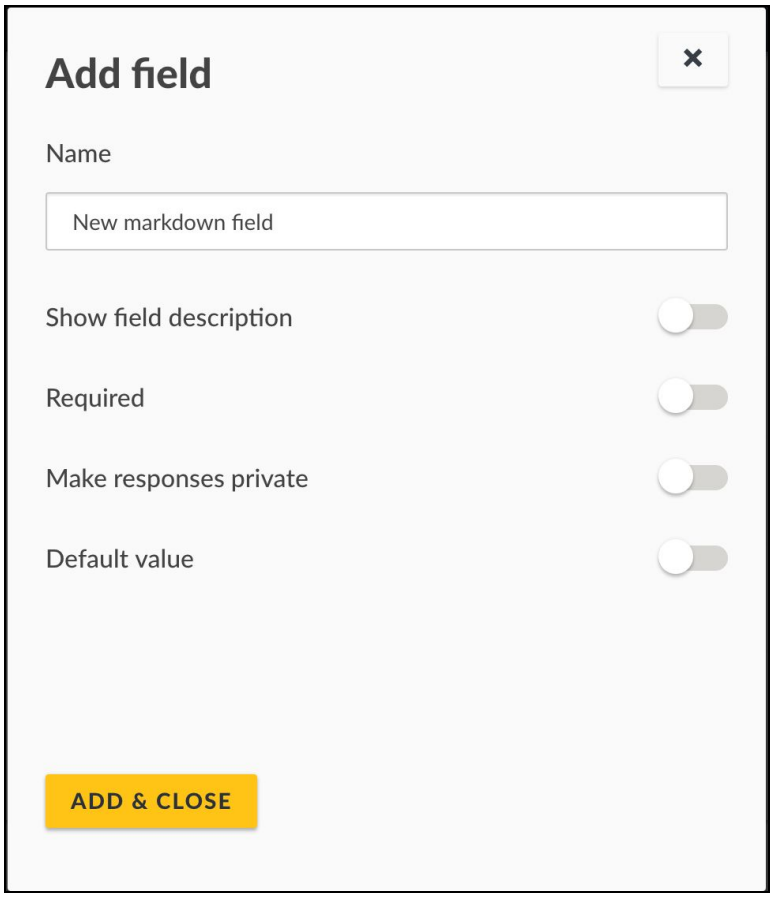
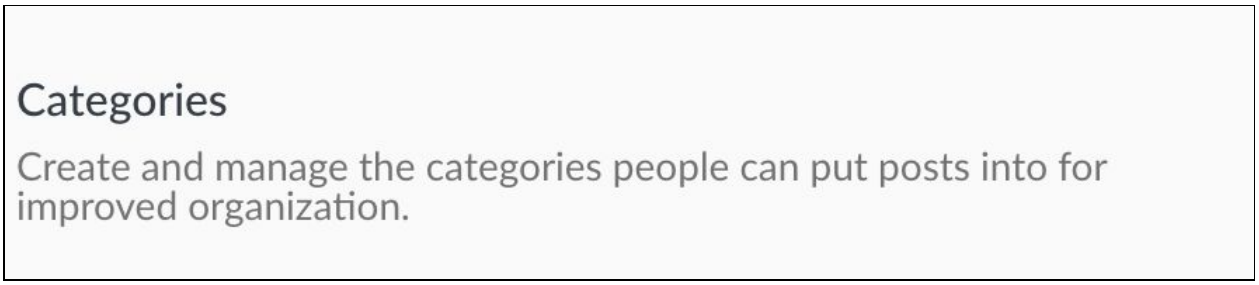
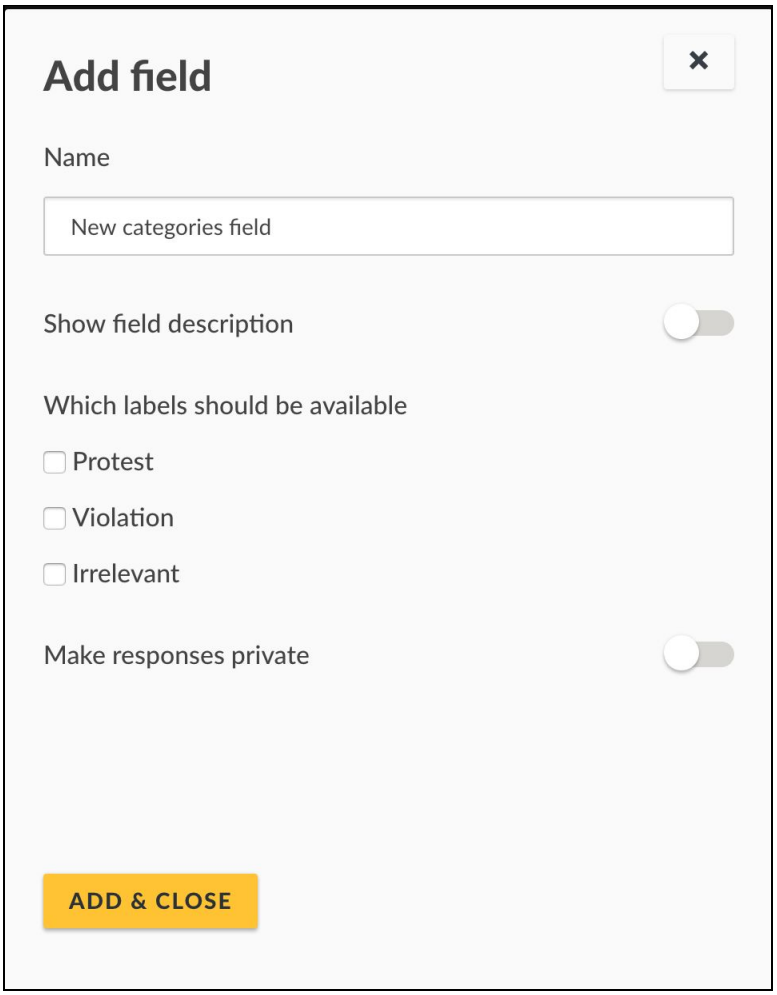


Figure 9. A “markdown” field must be created to use the YODIE service, which annotates the text of a post using markdown.

13. The fields above are relatively straightforward, but it is worth noting the “Required” and “Make responses private” fields.
14. If “Required” is enabled then it will not be possible to save or edit any post of this Survey type if it does not have a value for this field.
15. If the “Make responses private”, then the response to this field will only be visible to users with permission to access a given post. The field response will not appear when the Post is published.
16. When you are happy with the configuration of your new Markdown field, you can click Add & Close
17. Next we need to create a Categories field, this field is required for integration of a Survey with the CREES tool. The CREES tool will label a given post, in order to store and display those labels the Survey must have a Category field.
18. Begin again by pressing “+Add Field”, then select Category from the field list



19. The Category field type has slightly different options to the Markdown field.



20. You can select specifically which Categories from the set of all Categories configured in your deployment that this Category field can contain. For example, for a particular Survey you may only want it to handle a certain type of CREES labeling, violence for example, in that case you can reduce your Category list to just ones related to the Violence type. Or you can add all categories, the available Categories can be edited further at a later time.
21. Once you have added all the fields you need, it's a good idea to save your Survey in progress, at the moment the COMRADES **platform does not automatically save your data** so it's good practice to save periodically.
22. To do so you can click the yellow "save" button in the top right hand corner, this button will be gray if required fields such as "Name" have not been filled out.

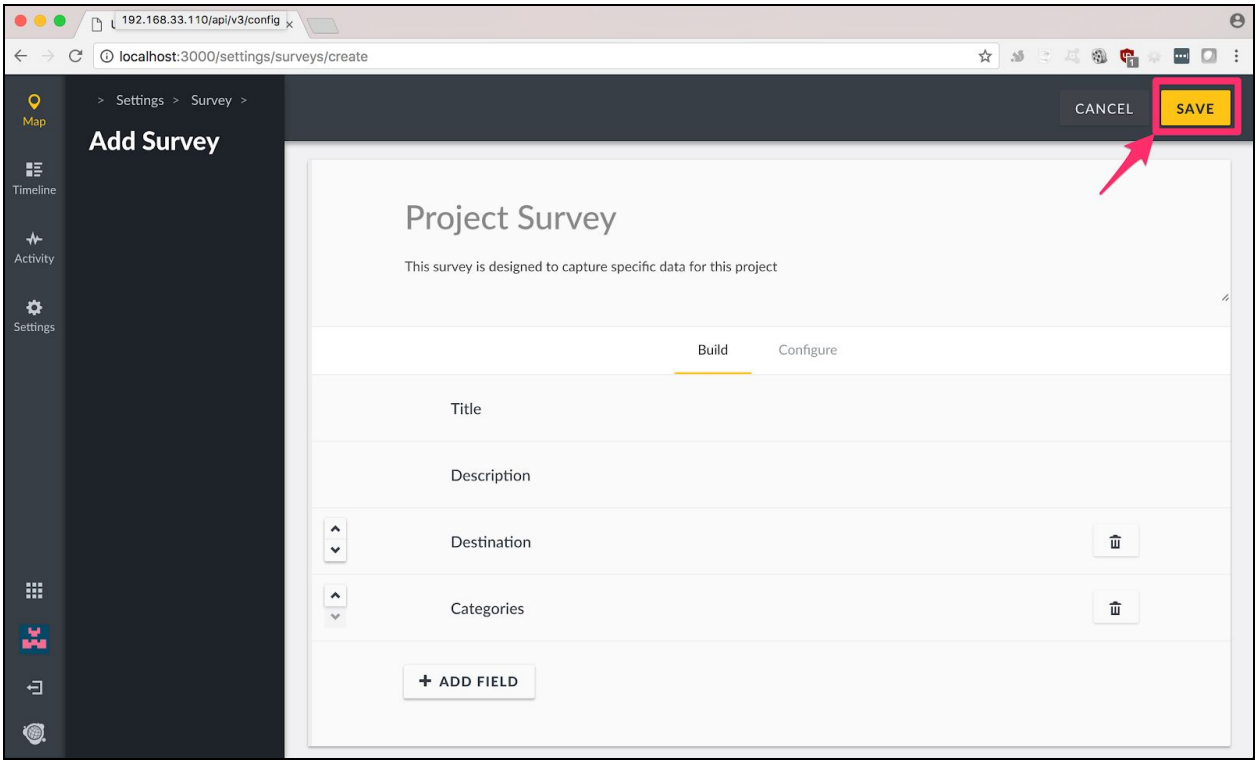


Figure 10. Be sure to save your work as you create surveys.

23. It is possible to further Configure Advanced options for your Survey, to view these options click on “Configure” as shown in the image below

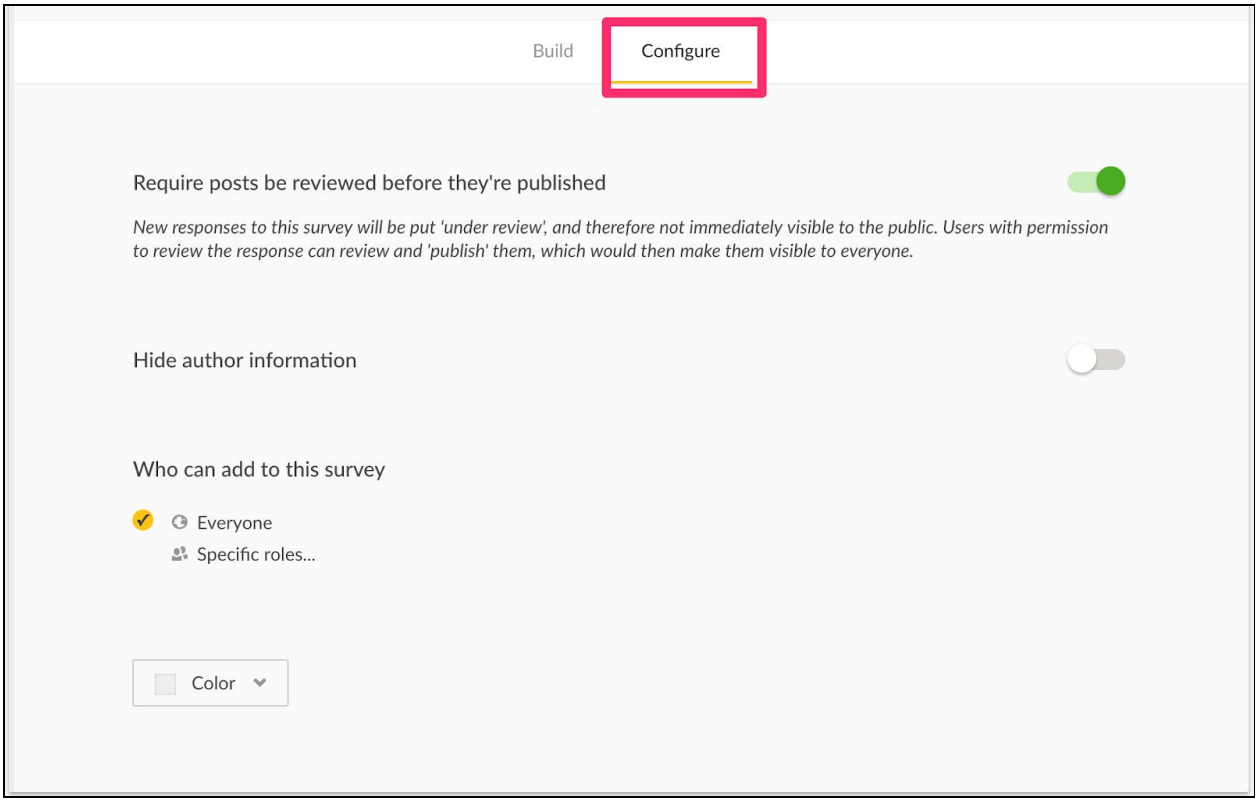


Figure 11. Configuring permissions within surveys.

24. Here you can configure the following:
    - Require post to be reviewed before they are published
    - Hide Author information from all users that do not have the Edit Posts permission
    - Control what User Roles have permission to Add/Edit Posts of this Survey type
25. To allow further organisation, flow control and more abstract structure. Surveys can also have Tasks. These are essentially separate sections within a Survey comprised of Optional Fields that can be ordered as desired. For example, if you wish to create a verification Task associated with a given Survey that is only visible to internal users it is possible to do that.

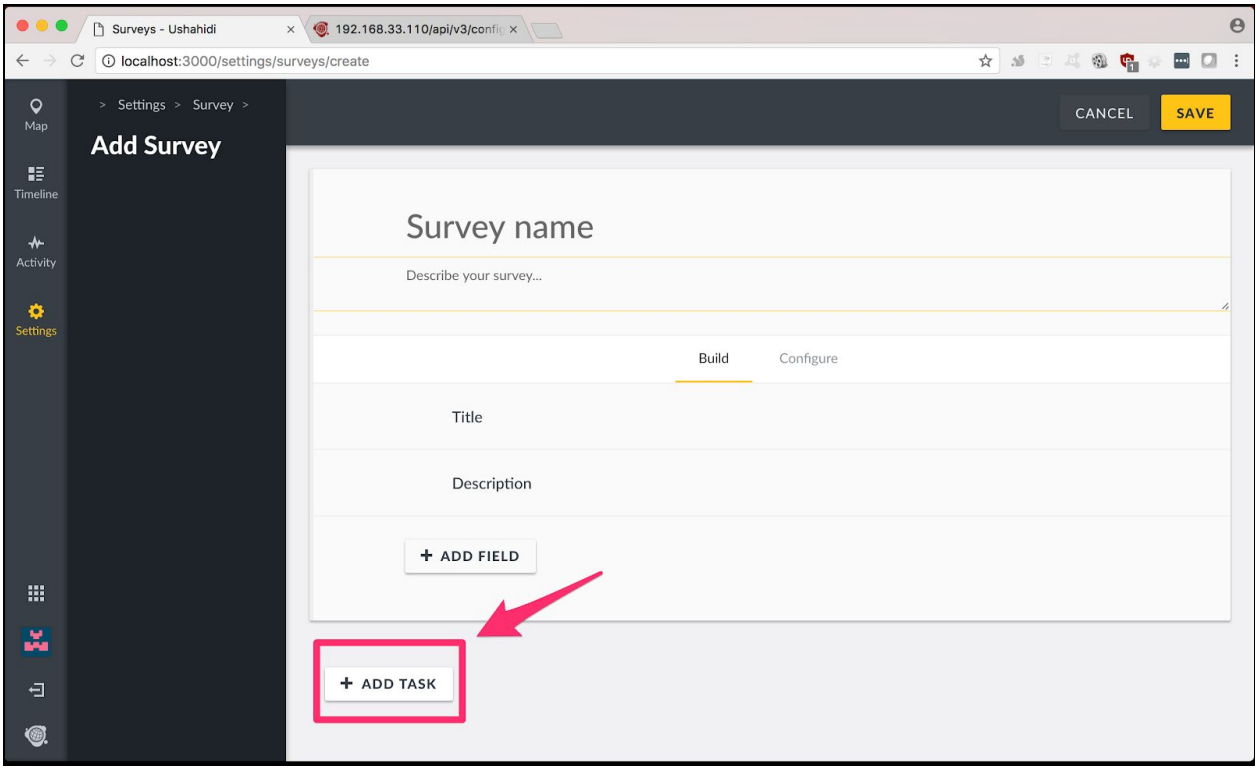


Figure 12. Configuring tasks for a survey.

26. To create a New Task simply click the "+Add Task" button shown in the image above
27. A dialog similar to that for Adding Fields will open

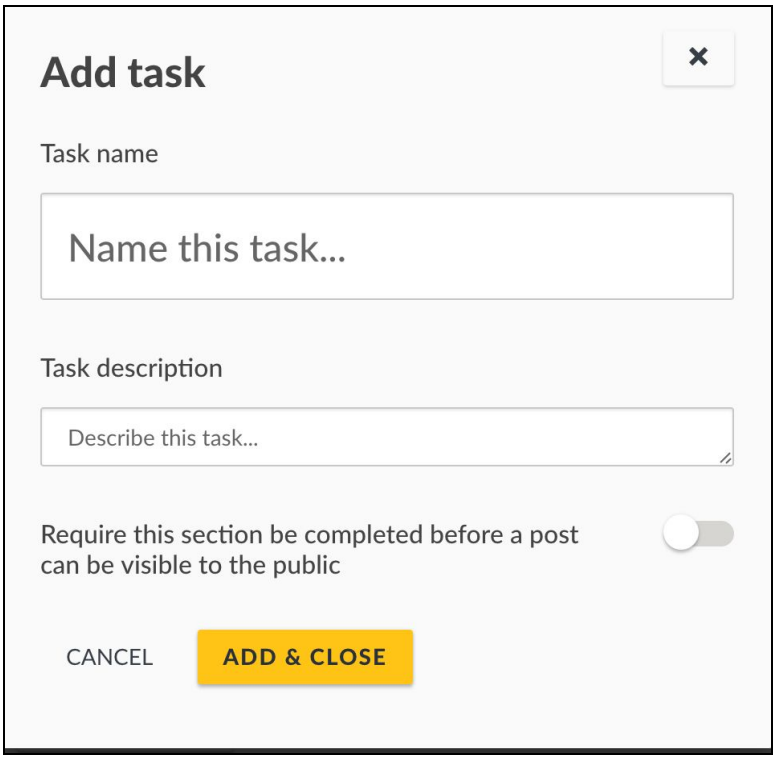
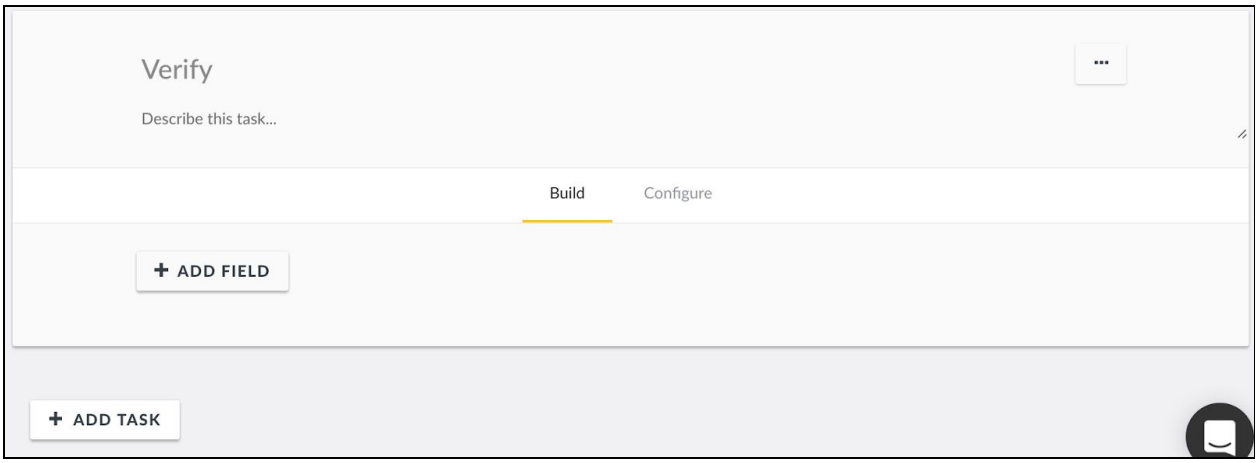
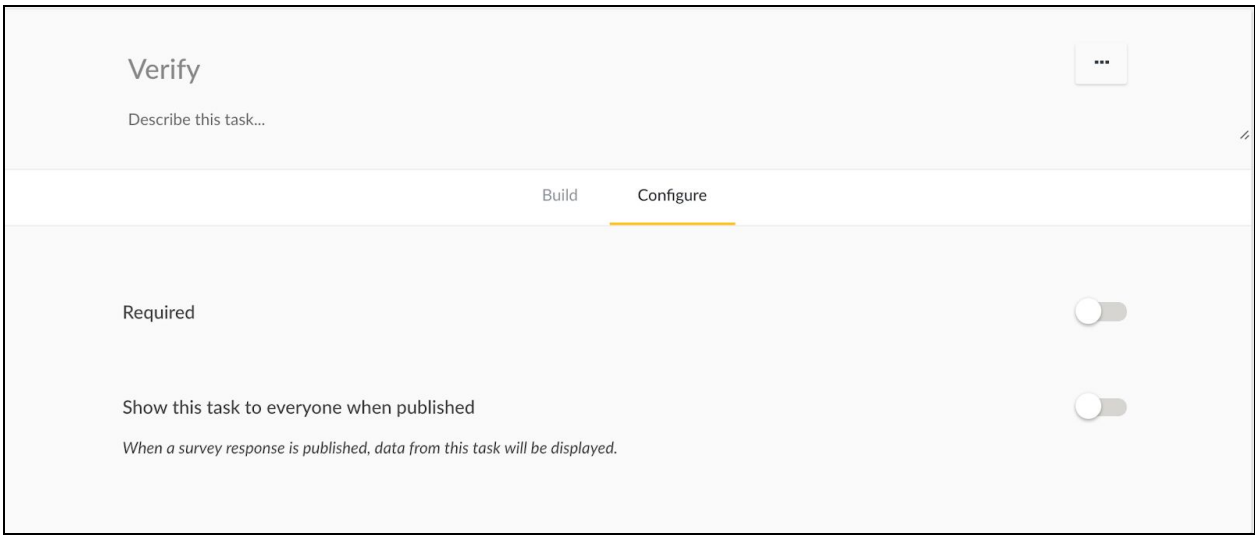


Figure 13. Default new task view

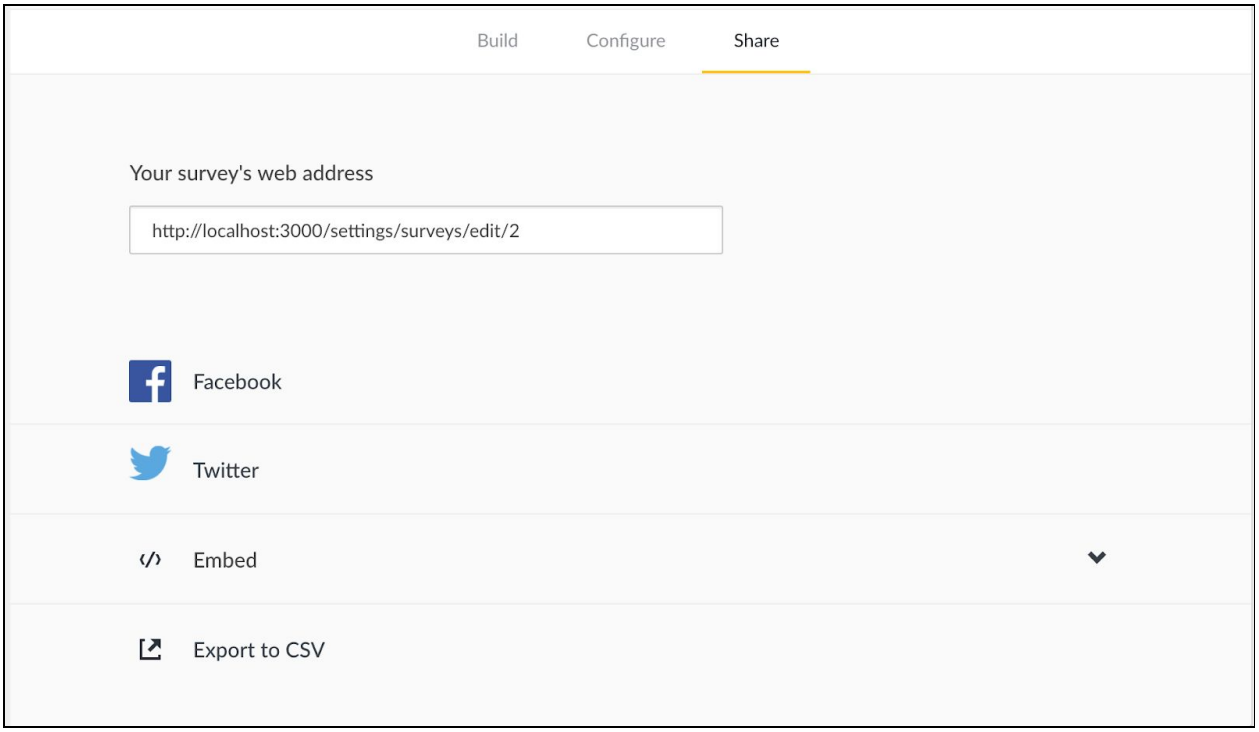
28. It is possible to require that users must complete a specific Task before a Post can be published.
29. Adding a new task "Verify" will add a new section to the Survey as shown below



30. You can then add as many custom fields as you wish using the "+Add Field" button of the task.
31. Task properties can also be edited via the Task "Configure" tab



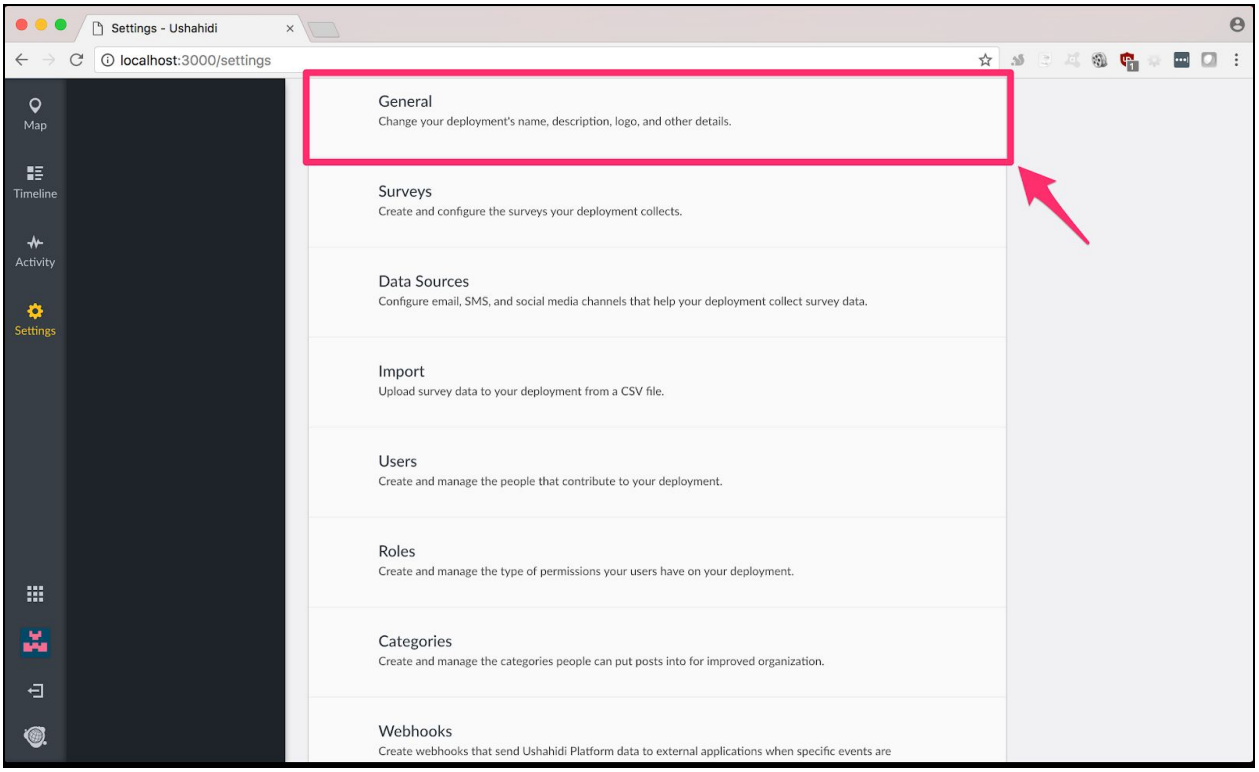
32. You can force a Task to be required to have been complete(that all its required fields have a value) before a Post of this Survey type can be published.
33. You can also control whether the Task is visible to users without Post Edit permissions when the post has been published
34. Lastly, there is one final Option for a Survey, the Share Menu



35. This menu provides the ways in which a given Survey can be shared. What will be shared is a direct/embedded link to a blank Post where others can start entering data.
36. You can also export all Posts of this Survey type to CSV via this interface.

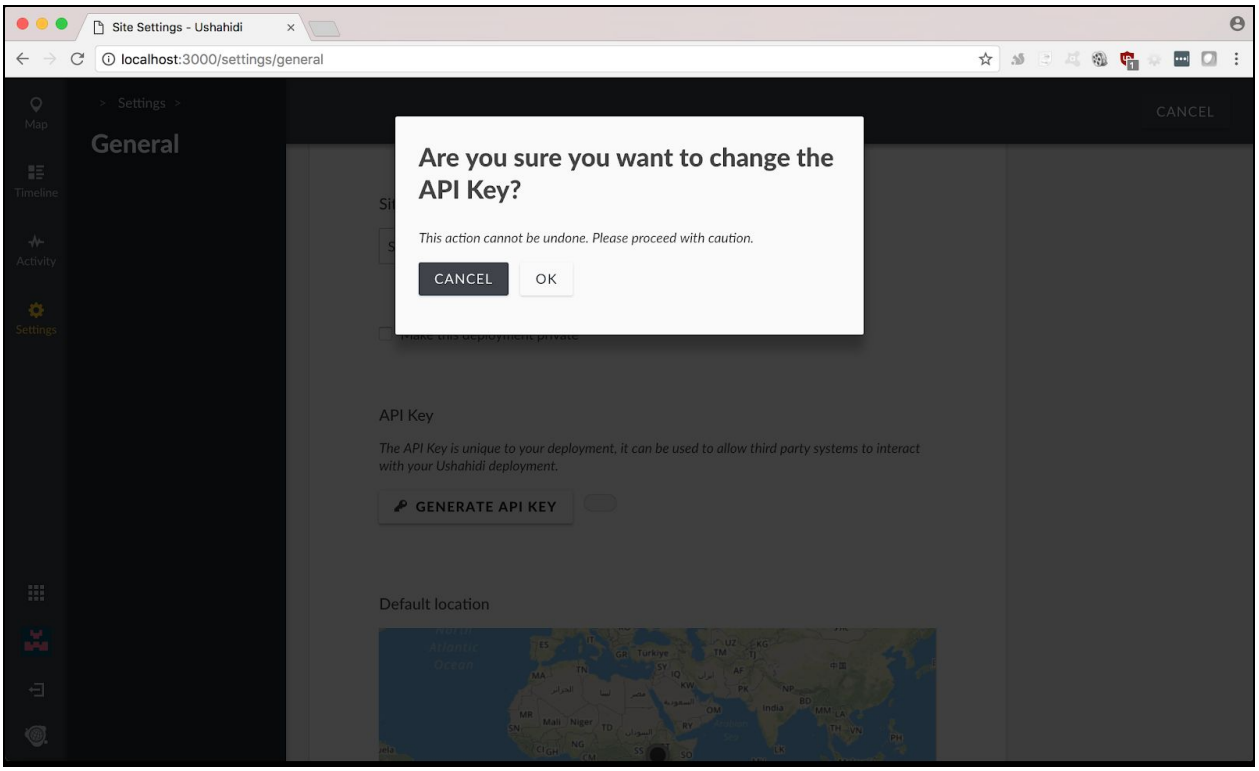
API Key

1. In order to allow integration with external services we have added an API key to the Platform, this key acts as a form of shared secret and should only be made accessible to Admin users.
2. In new deployments we must generate an API key to get started if it is the first time we have setup the deployment. First, go to the Settings Menu and click the “General” menu option.



3. Scroll down the General Settings page to the section “API Key”
4. If a key already exists, please be cautious and only regenerate the key if you are certain that it will not disrupt any external tool integrations
5. To generate a new Key or to generate a Key for the first time, click the “Generate API Key” button

6. You will receive a warning as shown below, **please only proceed if you certain you wish to (re)generate the API key.**

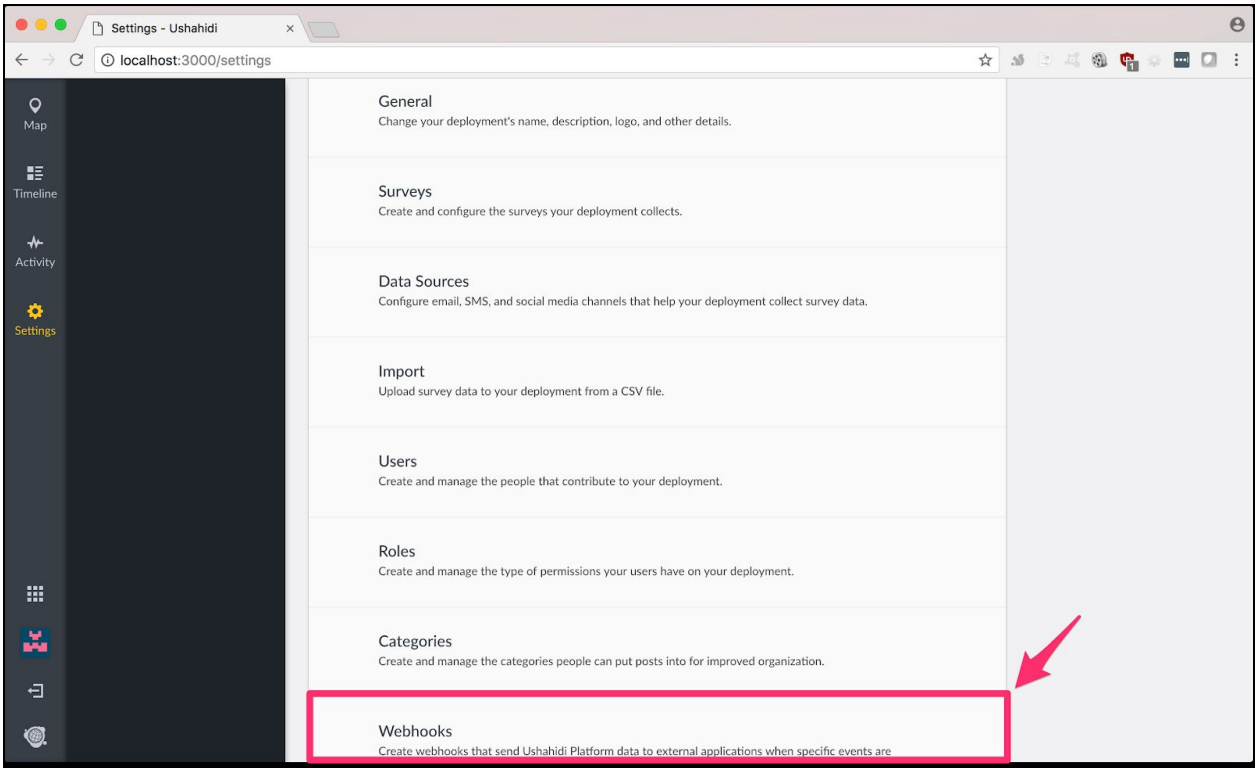


7. Once the Key generation is completed you will see something similar to the Key shown below

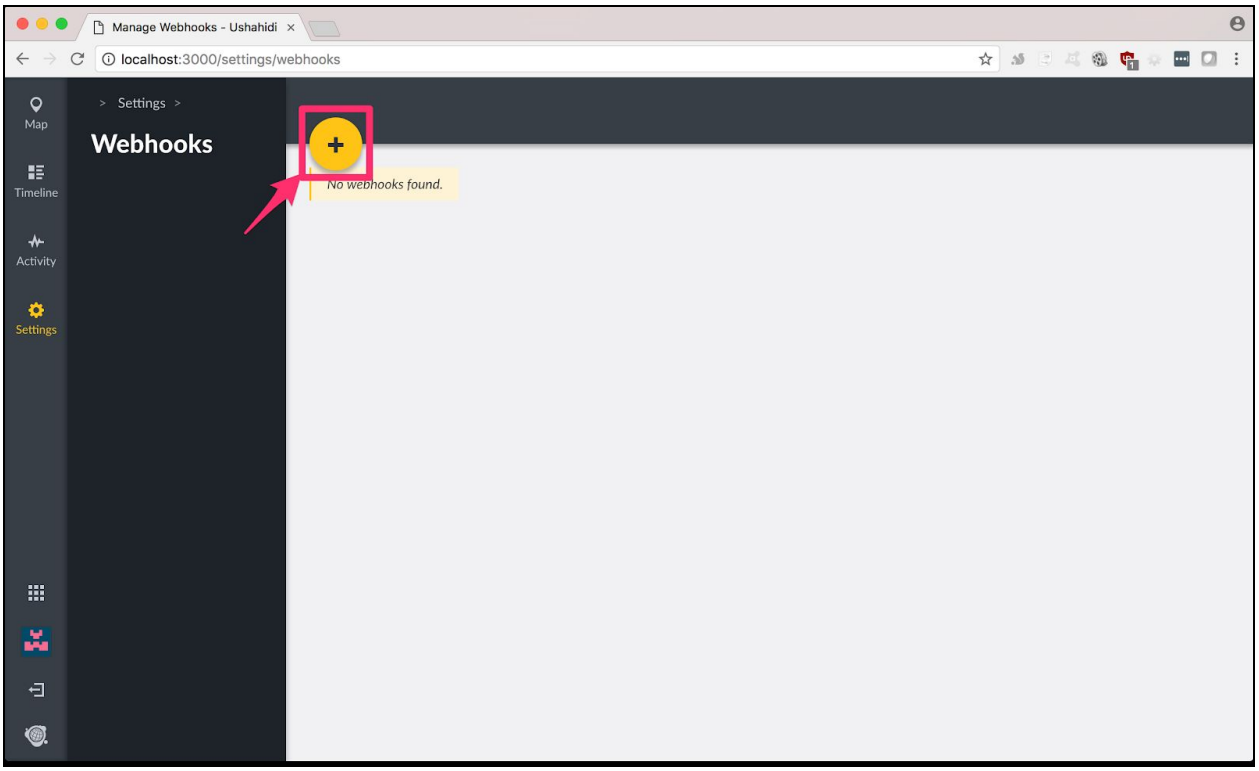
8. We will need to copy this Key again later in order to configure our external services but for now we are finished creating an API Key.

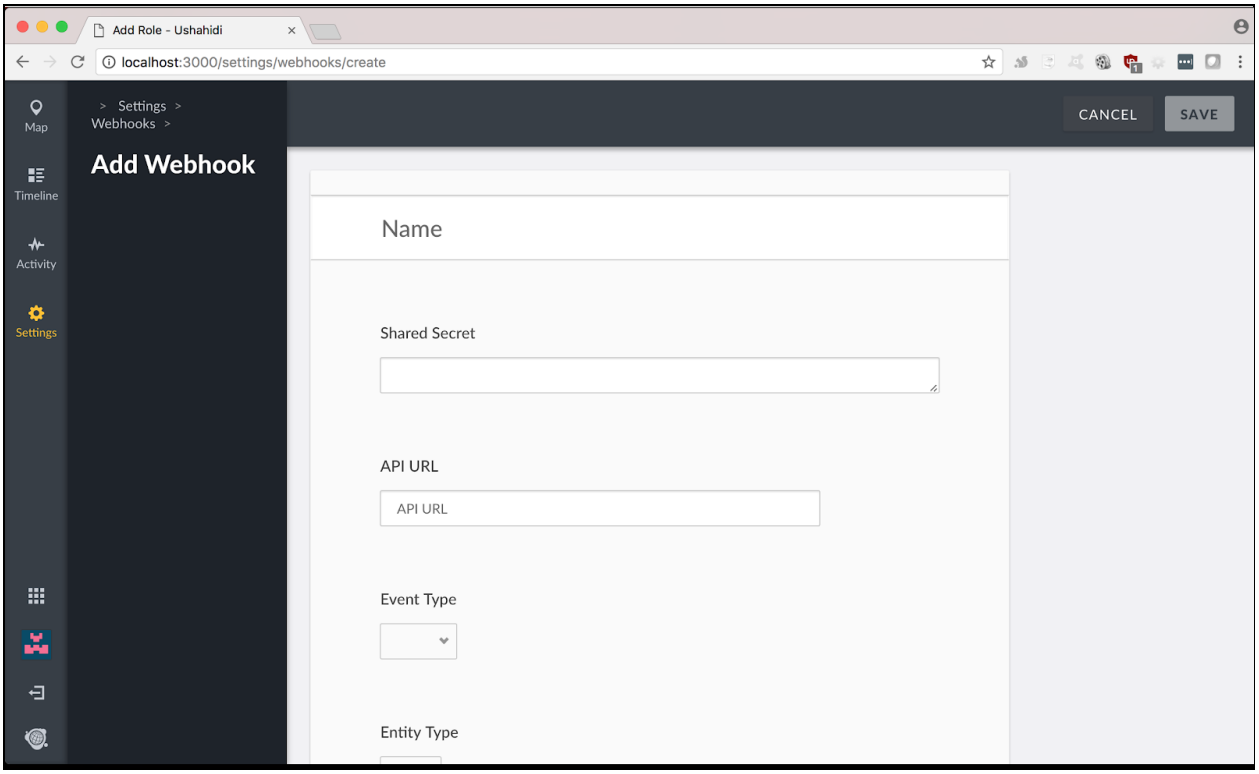
Webhooks

1. In order to directly share data with external service we have added Webhooks to the platform.
2. We need to create a webhook for each of the external service we wish to end data to.
3. Data will be sent by the webhook when a certain event occurs, currently allowed event types are Create, Update
4. To get started we go the Settings Menu and click “Webhooks” menu item

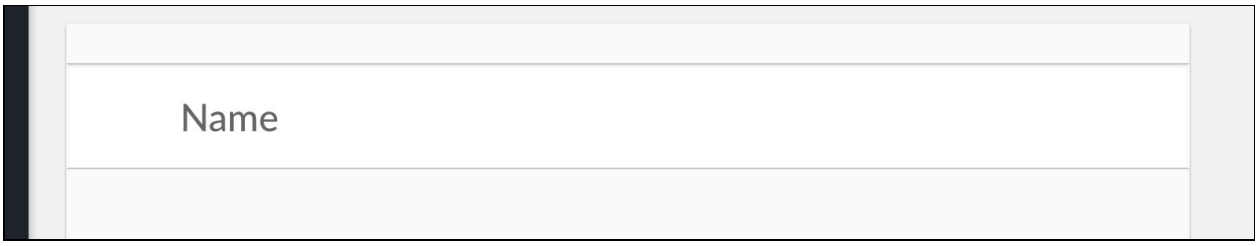


5. On the Webhooks List page we click to Add a new webhook as shown below

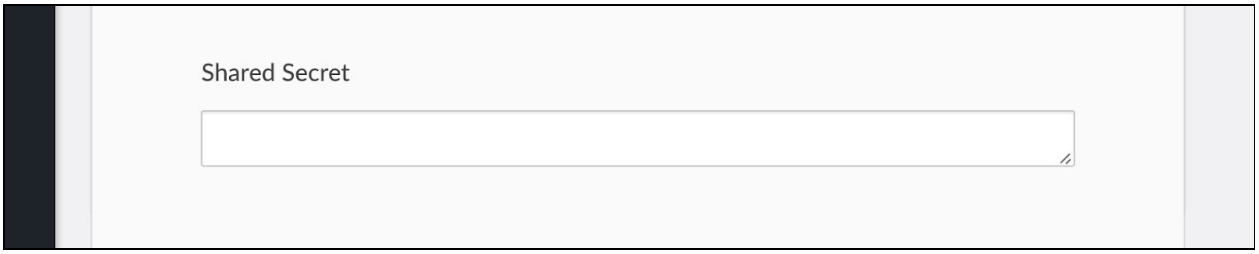




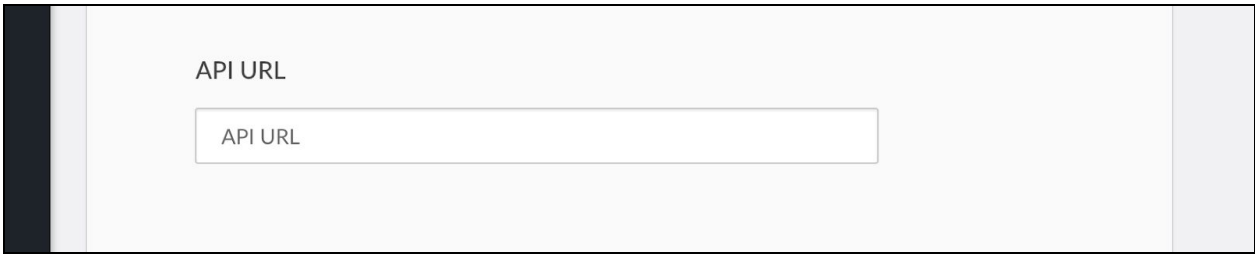
6. There are several important fields on a Webhook, the first most important is Name. We should give our Webhooks descriptive names so that there purpose and what service they send data to is clear at a glance.



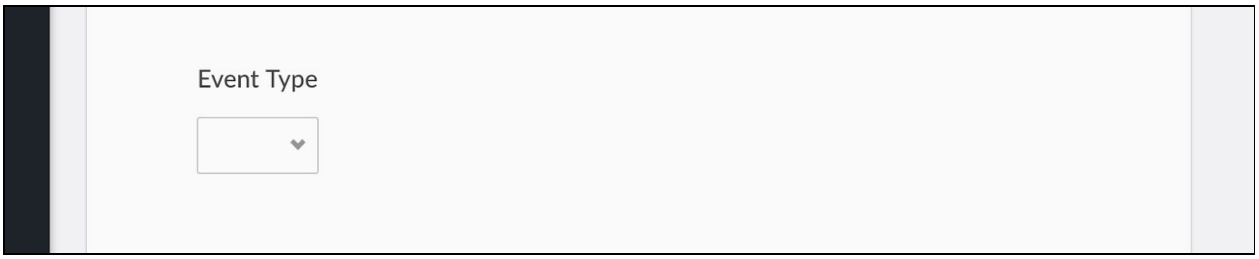
7. A shared secret, is a unique key that will be used to cryptographically sign the data sent between the Platform and the external service. This shared secret will also be configured on the COMRADES Proxy service, this will be discussed in later sections. The shared secret must be at least 20 characters long.

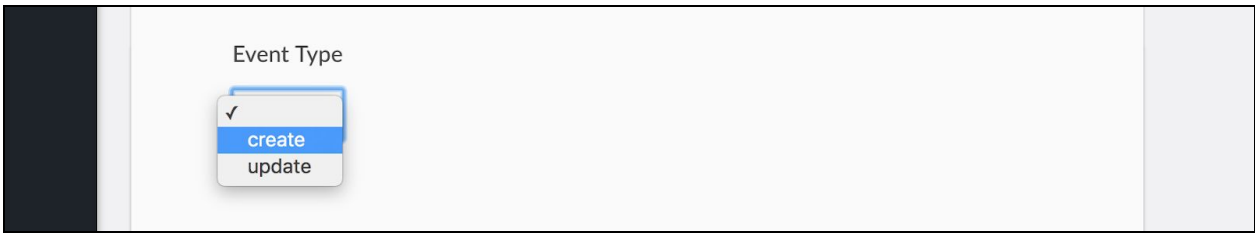


8. The API Url, is the url that identifies precisely where the Platform should send the event data

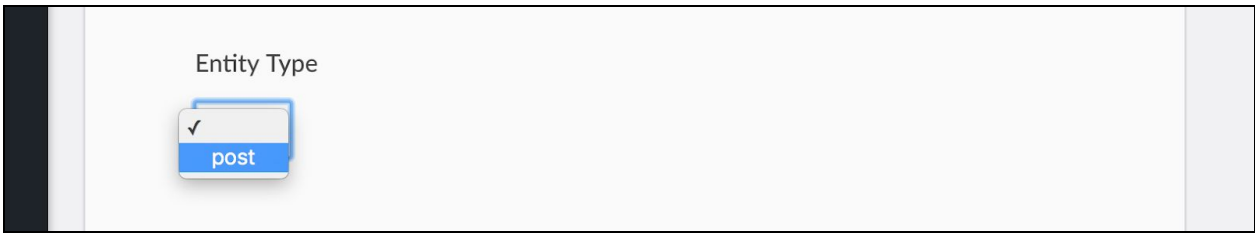


9. Event Type describes when a webhook should send data, the two options are on Create and on Update

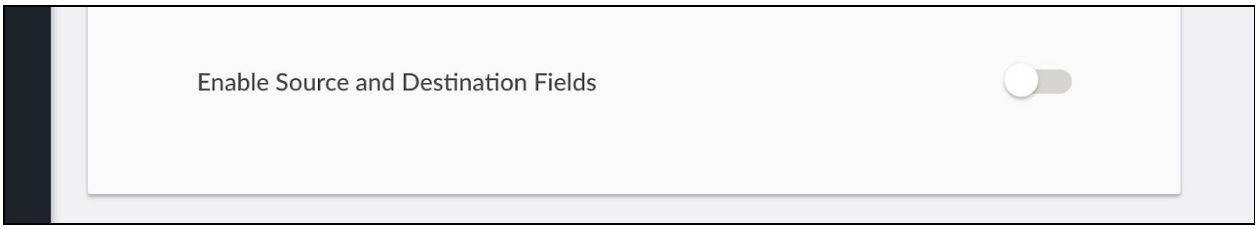




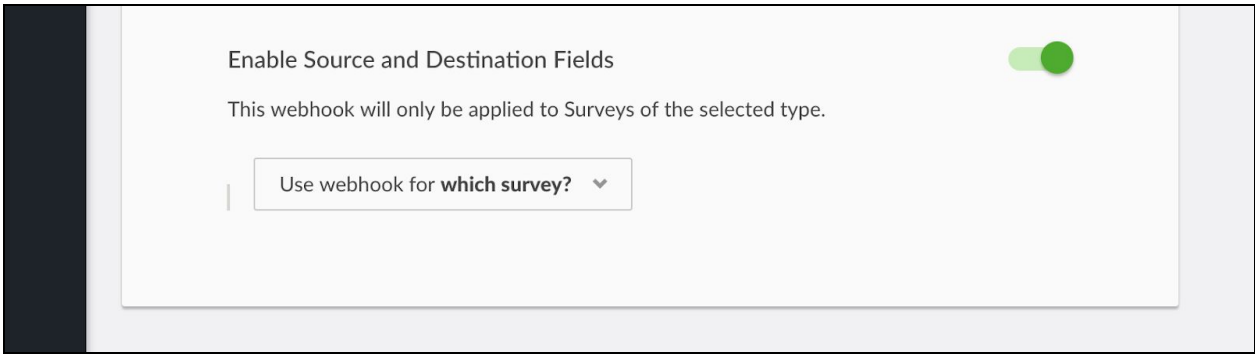
10. The Entity Type, the Platform entity on which the webhook should execute, at the moment only Post can be selected.

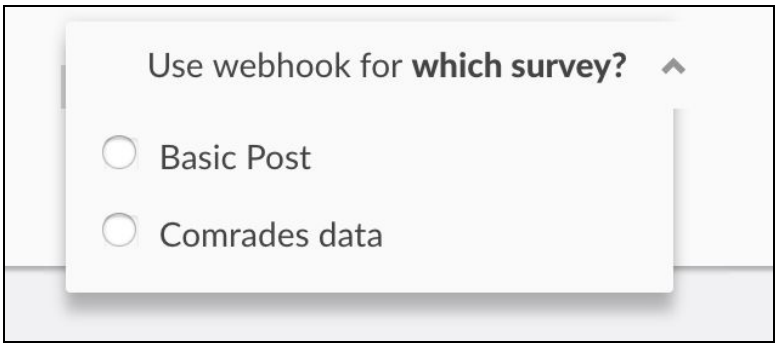


11. So if we select the Event Type “Create” and the Entity Type “Post”, then when a Post is created this Webhook will execute and send all the Post data to the API Url defined above, signed using the defined Shared Secret
12. We can configure out Webhook to pass additional context specific data. For the EMINA, CREES and YODIE integrations we need to tell the COMRADES Proxy Service which fields it should pay attention to. So we can specify explicitly which fields the Proxy should treat as Source and Destination.
13. First we must enable the Source and Destination Field feature

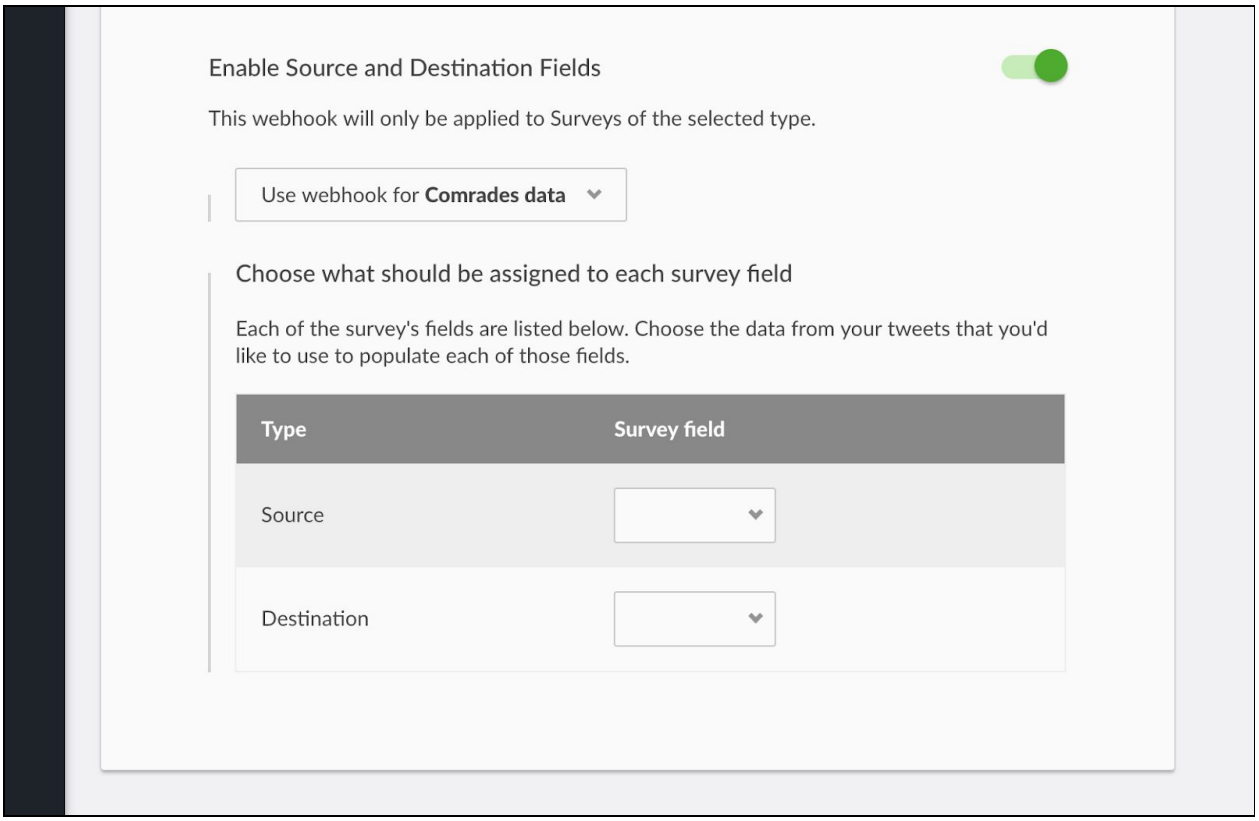


14. Then we select which Survey this Webhook will be associated with, it will only be executed for Posts of the defined Survey type

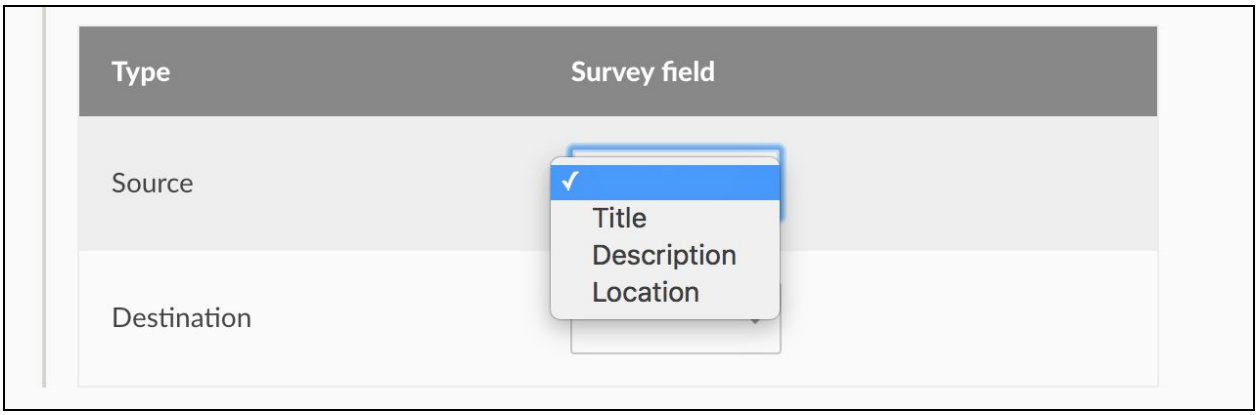




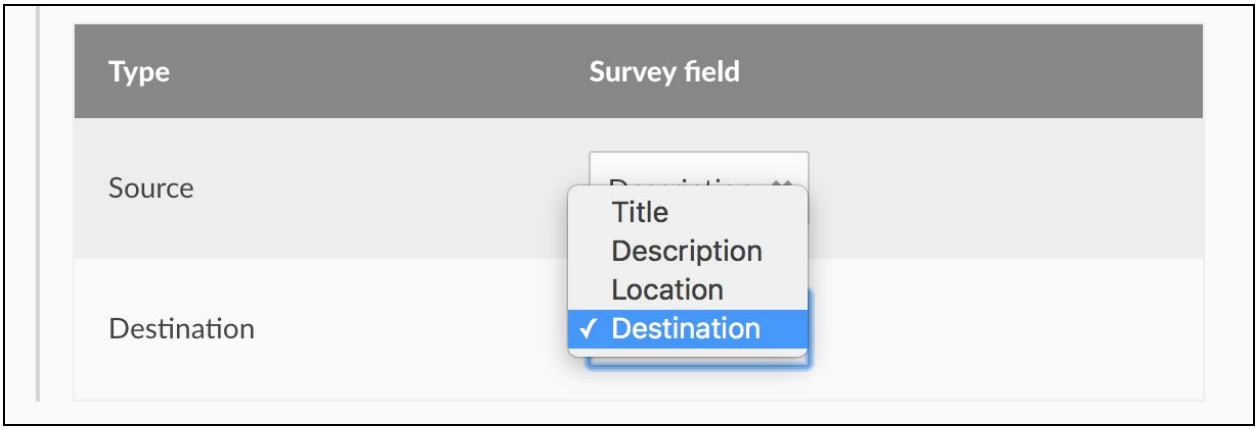
15. Having chosen the Survey, we next designate the Source and Destination fields for the given Survey type

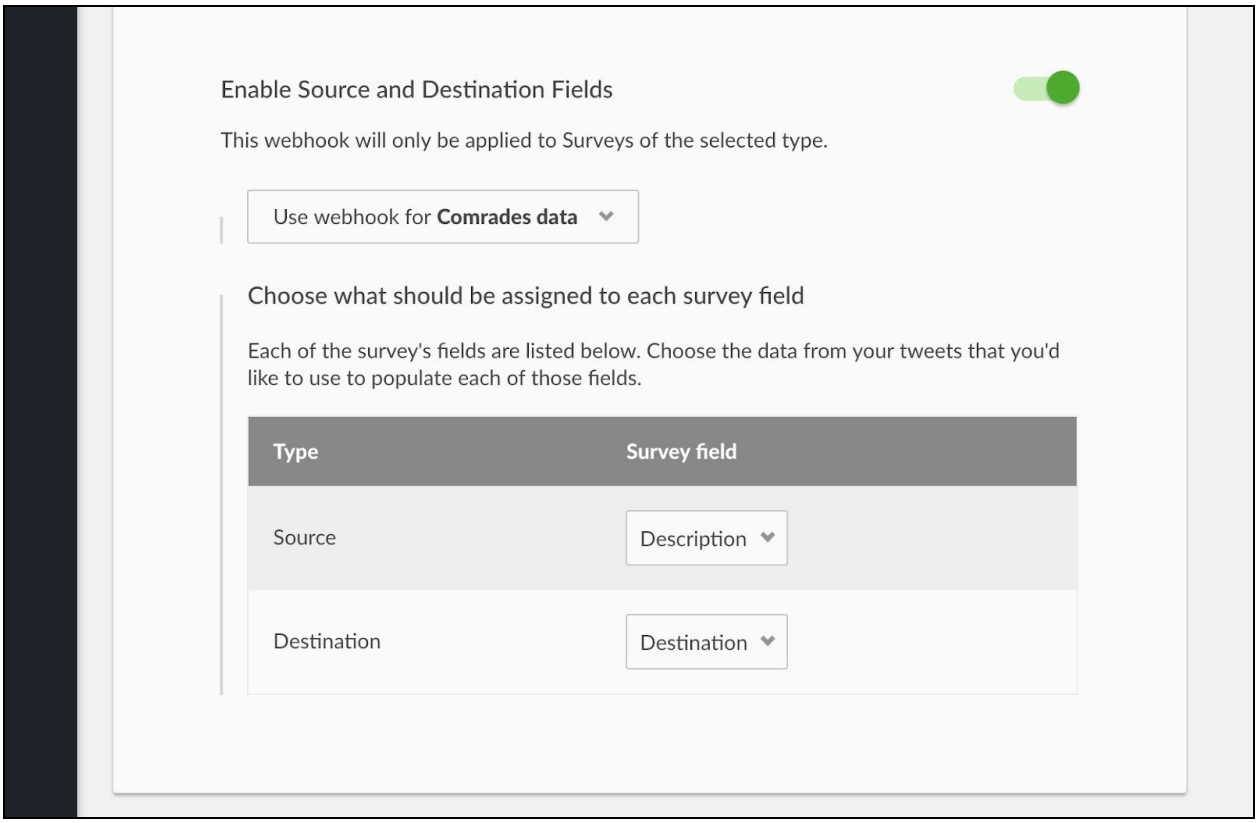


16. Select a Source field

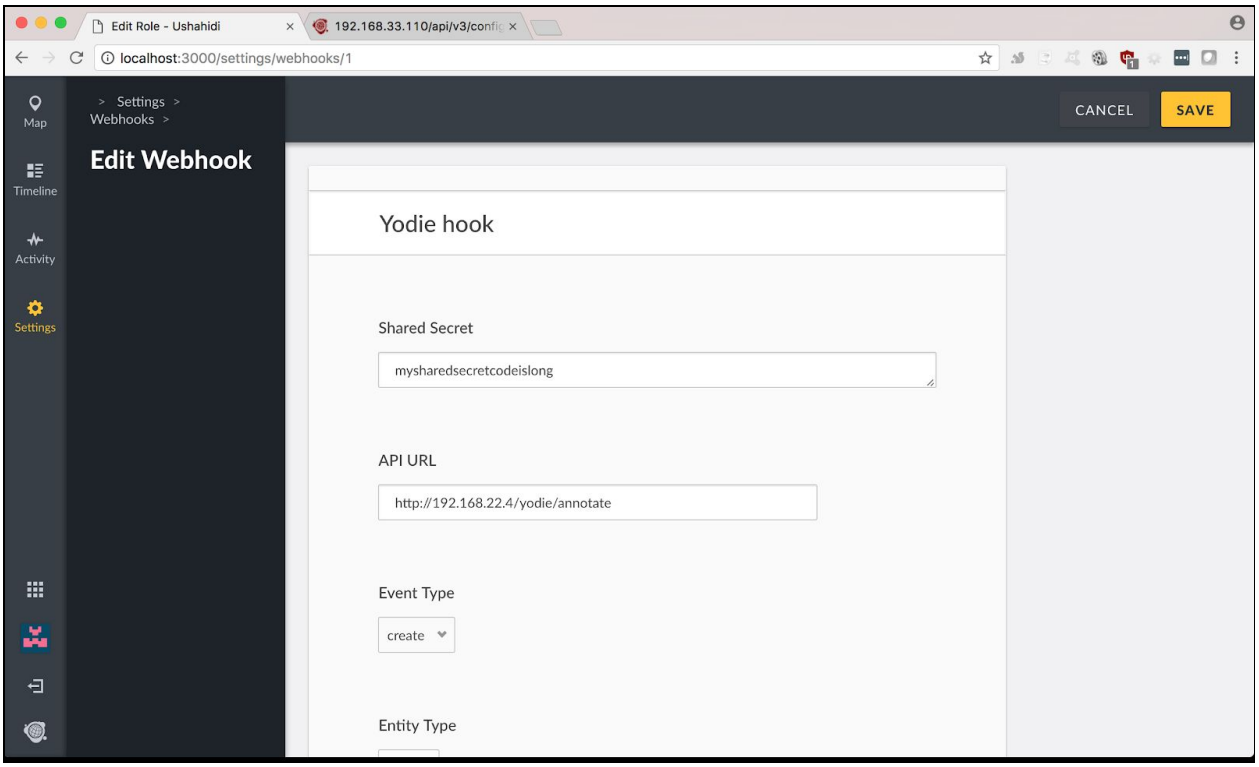


17. Select a Destination field



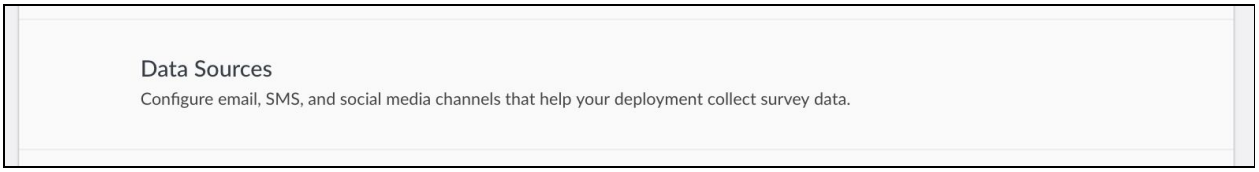


18. Save your new Webhook

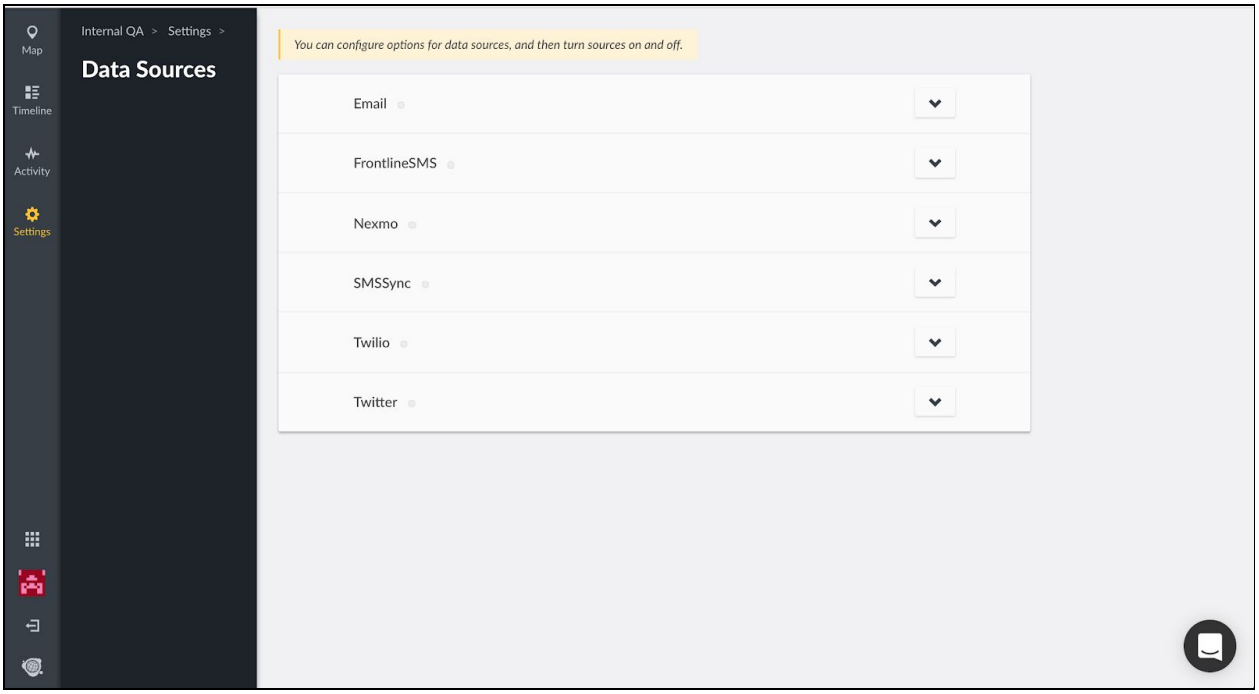


Datasources

1. First from the Settings Menu select the Datasources menu item
2. We will now configure some inbound data sources so that we can automatically import data into our Platform deployment



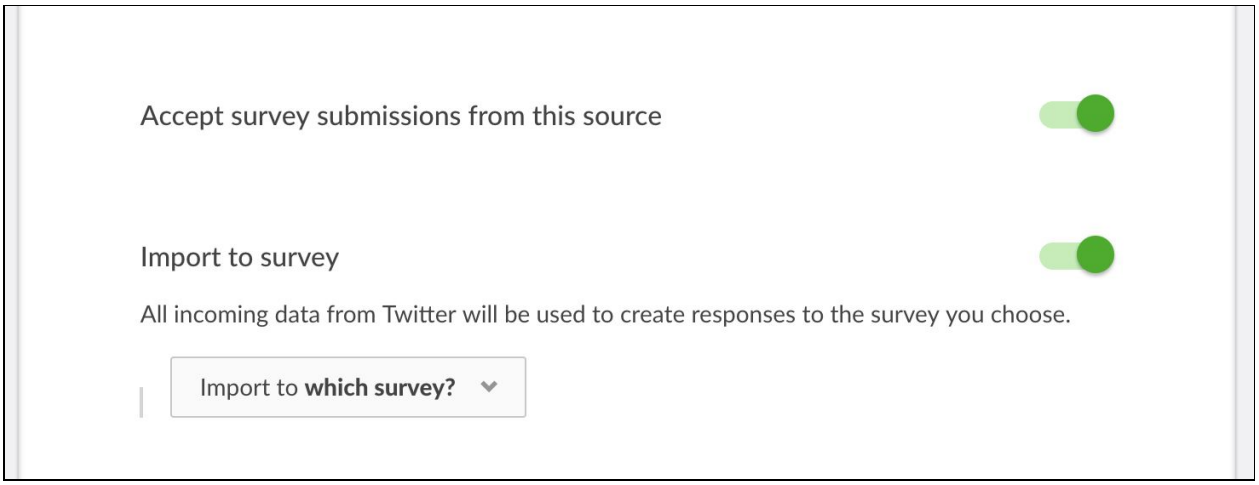
3. Platform supports 6 datasources of 3 types:
    a. Email, this service will connect directly to a given email account and import all Emails, once it has drawn in all emails from the inbox it will periodically recheck and pull in any new Emails. The imported Emails will be available as Posts in the Platform.
    b. SMS
        i. FrontlineSMS
        ii. Nexmo
        iii. SMSSync
        iv. Twilio
    c. Twitter, this service will connect to Twitter via provided Twitter credentials and important all Tweets that match a certain defined search criteria. It will continue to import Tweets periodically, these Tweets will be available as Posts in the Platform



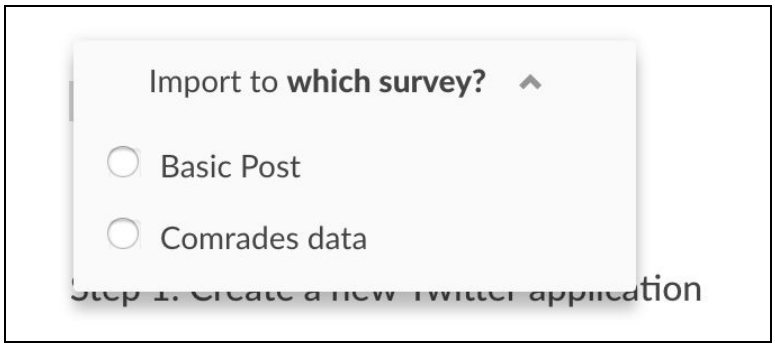
4. We will now walk through enabling and configuring the Twitter datasource, other than config parameters required all datasources are very similar. More detail for the other data sources' configurations is available on the Ushahidi Docs.
5. First, click on the Twitter menu item and it should expand

6. You can follow the basic config steps, once you have completed them move to step 7 below.
7. Once you have filled in the basic information we will next enable the data source

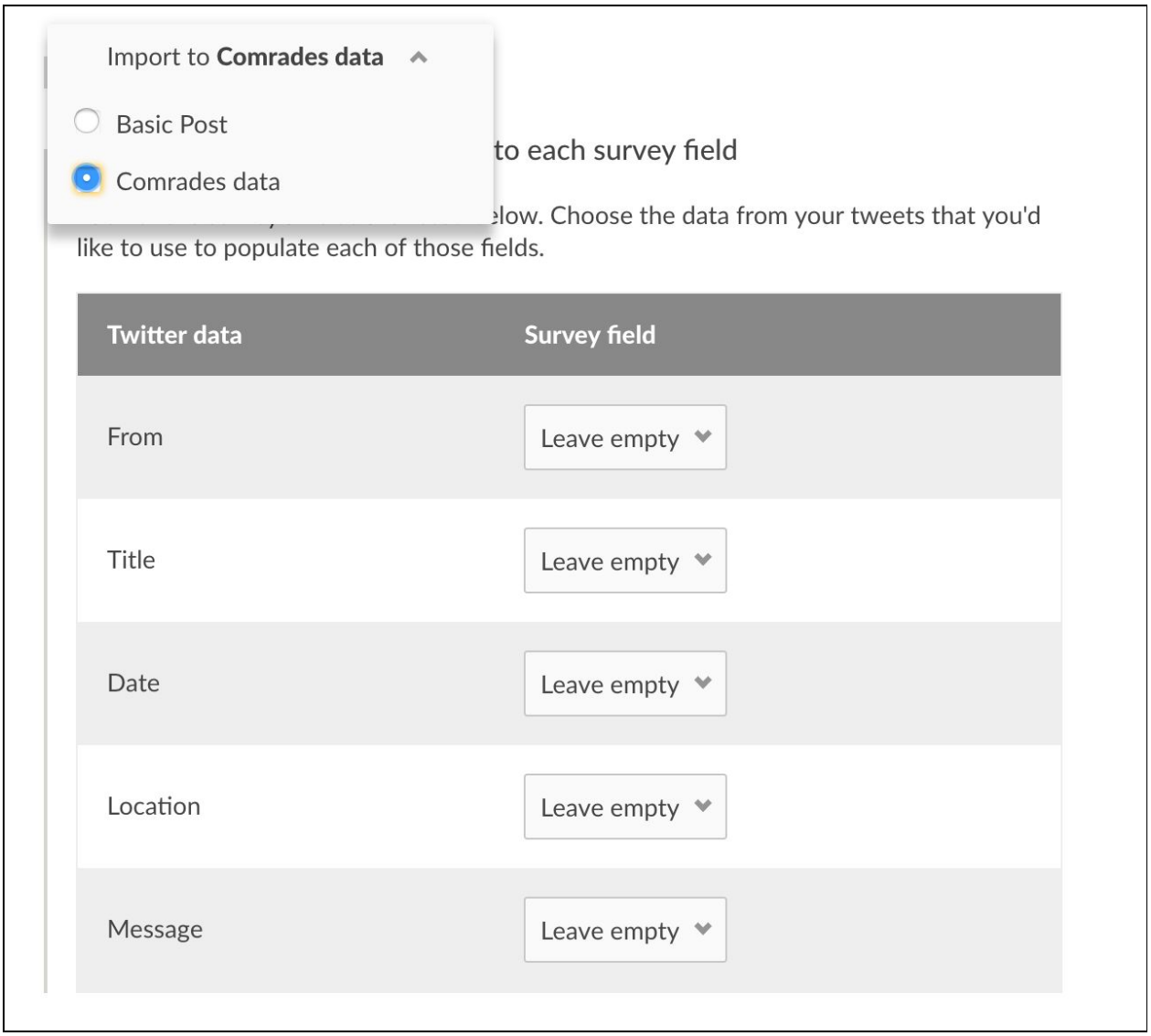
8. Next we need to associate the inbound data with a specific Survey, if we do not then all data from this source will be imported to the Unstructured or Unknown data category. That data is not initially associated with a Survey but can be associated subsequently



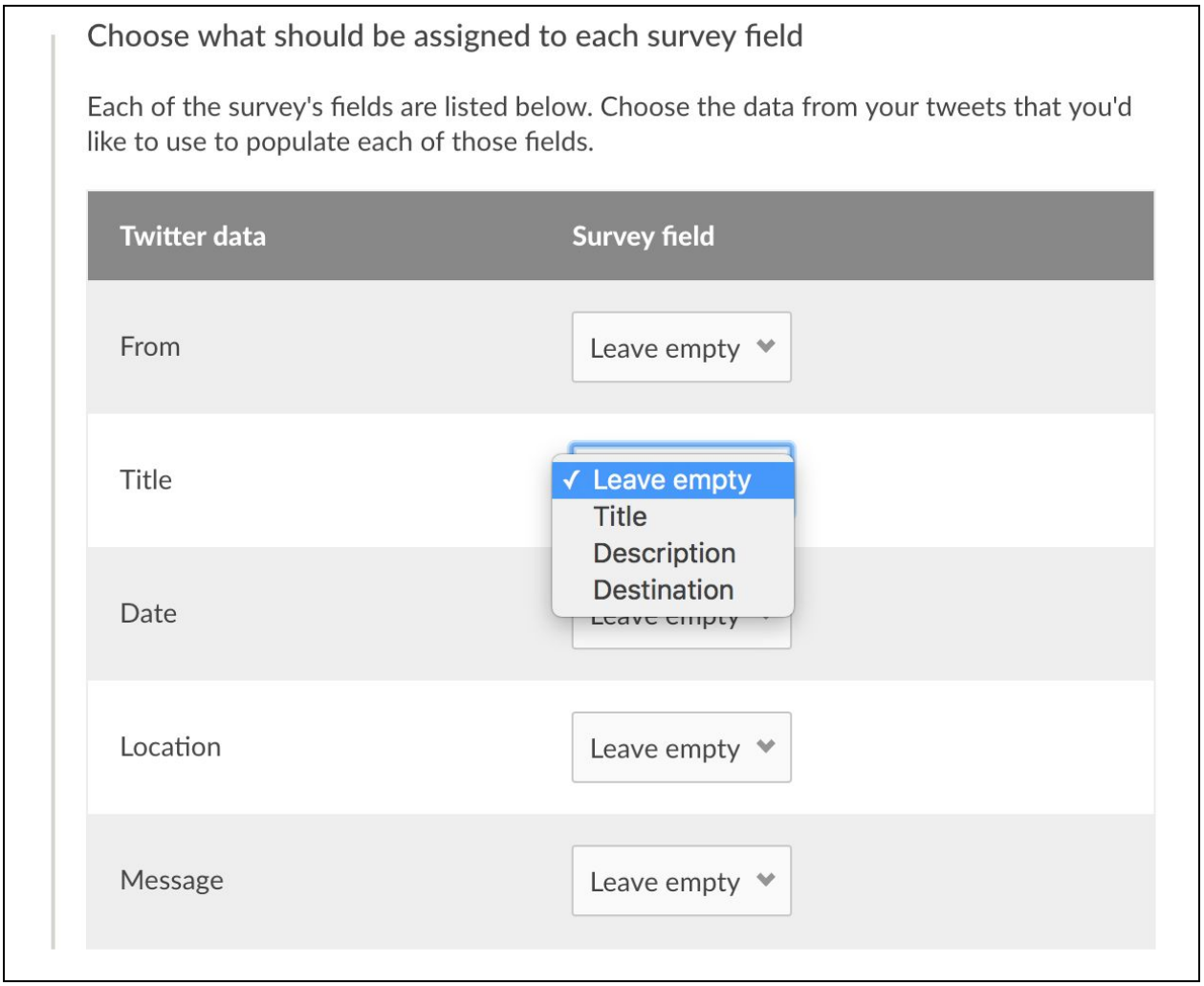
9. We next, select which Survey we want to make the association with



10. Having select the Survey we want, a new section will appear allowing us to map the inbound data fields to the appropriate Survey fields



11. You can choose whichever mapping you like, however, only Data fields can be assigned to Date fields and only Location fields can be assigned to Location Fields.



12. Once the mapping is complete, hit save and your data source will start importing on a roughly 15 minute cycle
13. Click the yellow “save” button.

COMRADES Proxy Service Setup

Overview

The COMRADES Proxy Service is relatively simple to configure, there is only one config file stored in the tools root directory, the file is named .env. By default a new COMRADES Proxy Service deployment comes with an example variable file called .env.example. This can be used as the basis for the systems configuration.

Config Options

```
APP_ENV=local
APP_DEBUG=true
APP_KEY=
APP_TIMEZONE=UTC

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret

CACHE_DRIVER=memcached
SESSION_DRIVER=memcached
QUEUE_DRIVER=database

USHAHIDI_PLATFORM_SHARED_SECRET=<your_shared_secret_at
_least_20_chars>
USHAHIDI_PLATFORM_API_URL=<full_url_of_ushahidi_platfo
rm_instance>
USHAHIDI_PLATFORM_API_KEY="<platform api key>"
```

```
YODIE_API_URL=<full_url_of_yodie_api>
YODIE_API_KEY=<yodie_api_key>
YODIE_API_SECRET=<yodie_api_secret>

CREES_API_URL="<full_url_of_crees_api>"

CREES_EVENT_RELATED="eventRelated"
CREES_EVENT_TYPE="eventType"
CREES_INFO_TYPE="infoType"
```

Database Options:

The following options should be changed as appropriate for your setup

- `DB_CONNECTION`
- `DB_HOST`
- `DB_PORT`
- `DB_DATABASE`
- `DB_USERNAME`
- `DB_PASSWORD`

Cache Options:

In general, it is not necessary to change these options, but if you do wish to then you can find more information about the caching options:

- `CACHE_DRIVER`
- `SESSION_DRIVER`
- `QUEUE_DRIVER`

Ushahidi Platform Options:

- `USHAHIDI_PLATFORM_SHARED_SECRET`, this field should be set to the value defined for the Webhooks you have created on the COMRADES PLatform
- `USHAHIDI_PLATFORM_API_KEY`, this should be set to the value of the COMRADES Platform API Key which you (re)generate in the API Key section

- USHAHIDI_PLATFORM_API_URL, this should be set to the URL for the COMRADES platform endpoint. This is where the annotated/label results will be sent to update the data for a given Post. The default location is

YODIE Options:

- YODIE_API_URL, this is the YODIE api url endpoint where you wish to send the data to be annotated
- YODIE_API_KEY, this is the unique key of your account with Yodie
- YODIE_API_SECRET, this is the secret of your account with Yodie

CREES Options:

- CREES_API_URL, this is the URL of the CREES API endpoint that you wish to send the data to be labelled to
- The following fields should not need to be changed from the provided values:
    - CREES_EVENT_RELATED,
    - CREES_EVENT_TYPE,
    - CREES_INFO_TYPE,

Actionability service Options:

- ACTIONABILITY_API_URL= this is the actionability api url endpoint where you wish to send the data to be categorized
- ACTIONABILITY_QUOTA= initial service throttling quota
- ACTIONABILITY_REQUEST_COST= quota costs per request sent (usually 1)
- ACTIONABILITY_QUOTA_RESET= this is the delay the actionability service has to reset the quota limits
- ACTIONABILITY_API_USERNAME= this is your username for the actionability service
- ACTIONABILITY_API_PASSWORD= this is your password for the actionability service

Veracity service Options:

- VERACITY_API_URL= this is the actionability api url endpoint where you wish to send the data to be categorized
- VERACITY_QUOTA= initial service throttling quota

- `VERACITY_REQUEST_COST=` quota costs per request sent (usually 1)
- `VERACITY_QUOTA_RESET=` this is the delay the veracity service has to reset the quota limits
- `VERACITY_API_USERNAME=` this is your username for the actionability service
- `VERACITY_API_PASSWORD=` this is your password for the actionability service

Facebook Bot Setup

The Ushahidi facebook-bot is built with Lumen and a mysql database.

Prerequisites

1. Make sure you have the following installed:
    a. Composer
    b. Vagrant
    c. Virtual box
2. You must have an instance of the COMRADES Platform deployed and accessible via the internet, as you will need to point the facebook bot to your deployment.

Set-up

- Clone the repo: https://github.com/ushahidi/platform-facebook-bot
- Create a `.env` file. An example-file is found in `.env.example`

The script

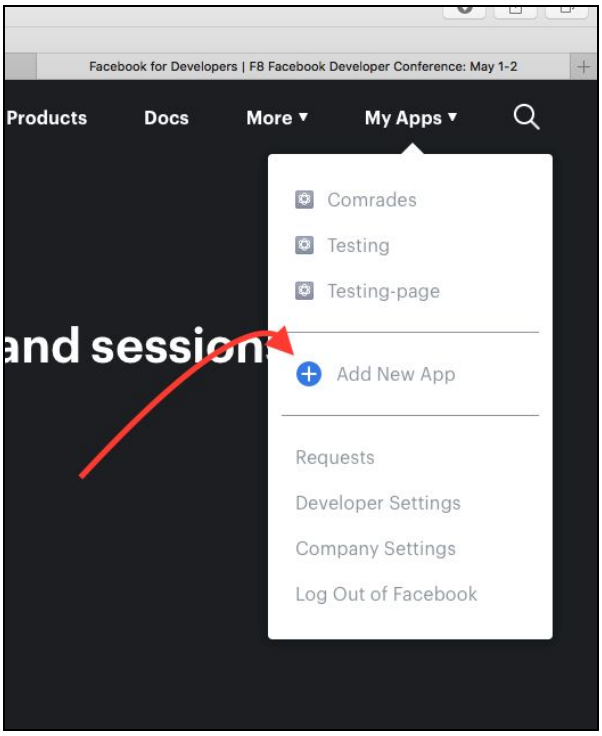
Bot uses a predefined script when talking to the users. It is not possible to adjust the questions themselves but during setup, some information about the ushahidi-deployment and the campaign is needed. That information is added in the `.env` file and is used to create and adjust this script to the organisation using the bot. These are:

TITLE: The title of the Ushahidi-deployment connected to the the bot is used
AIM: The aim/a short description of the campign where the bot is used.
NAME: The name of the bot
TWITTER_NAME: The twitter-name for the organisation/campaign using the bot
TWITTER_HASH: The hashtag used for the organisation/campaign

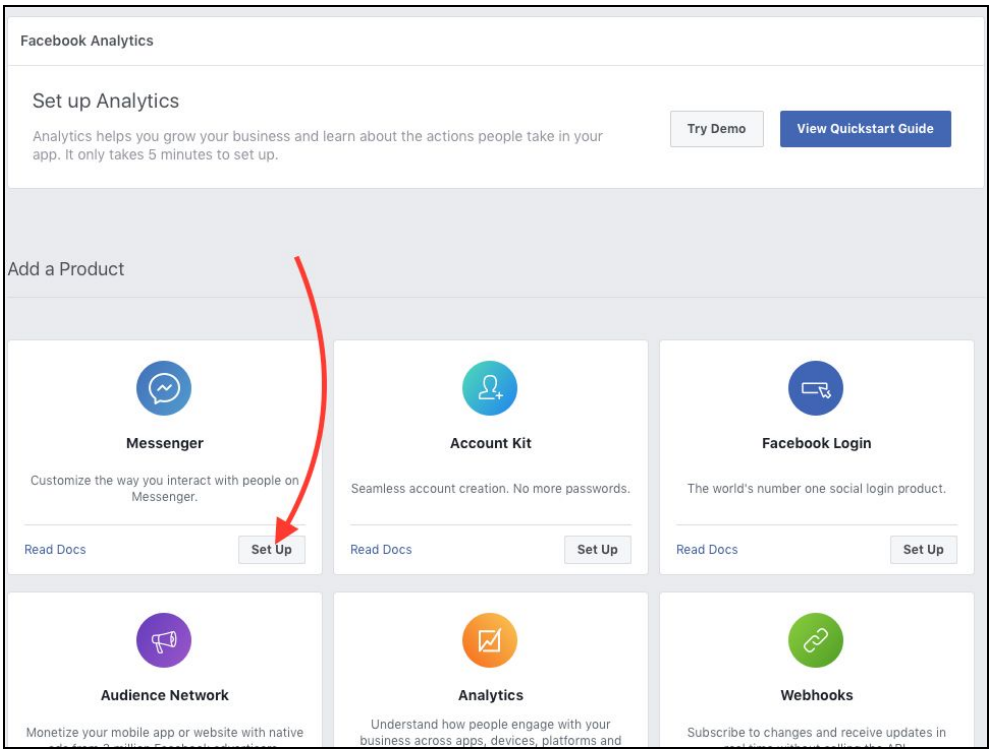
The full script of the bot can be found in the next section.

Credentials

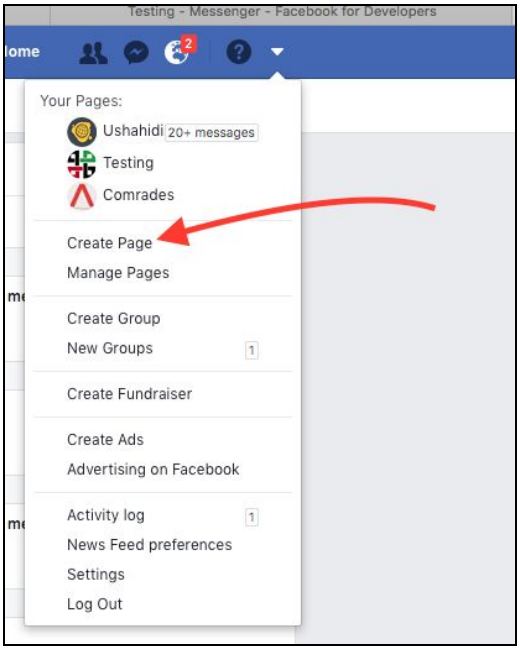
For some of the credentials in the `.env-file` you need to create a "Facebook-app". You do that here: https://developers.facebook.com.



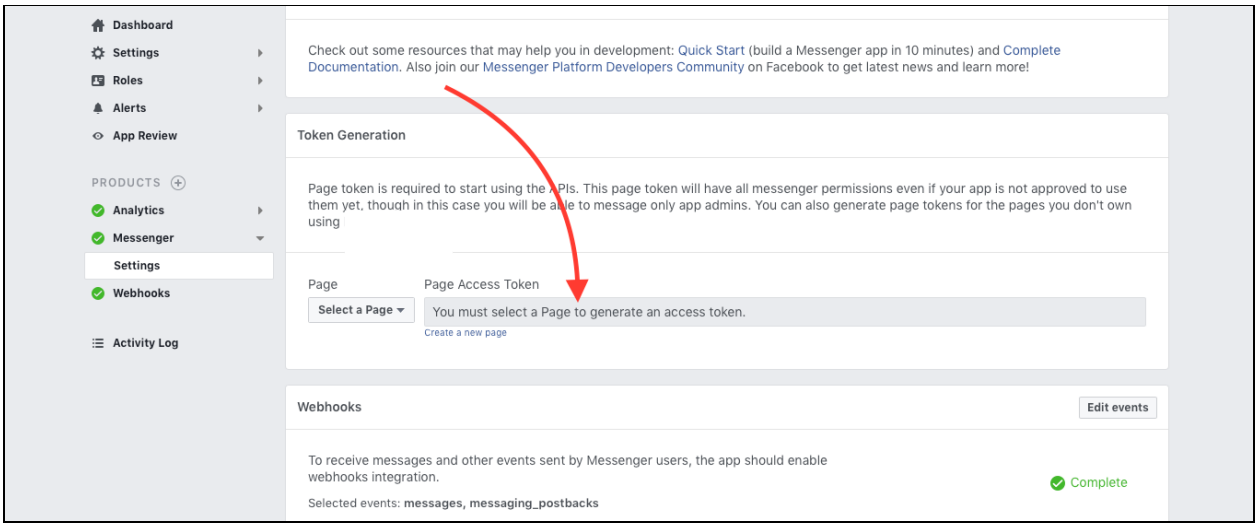
After creating the app, select “Messenger” under “Add a product” in your app-settings:



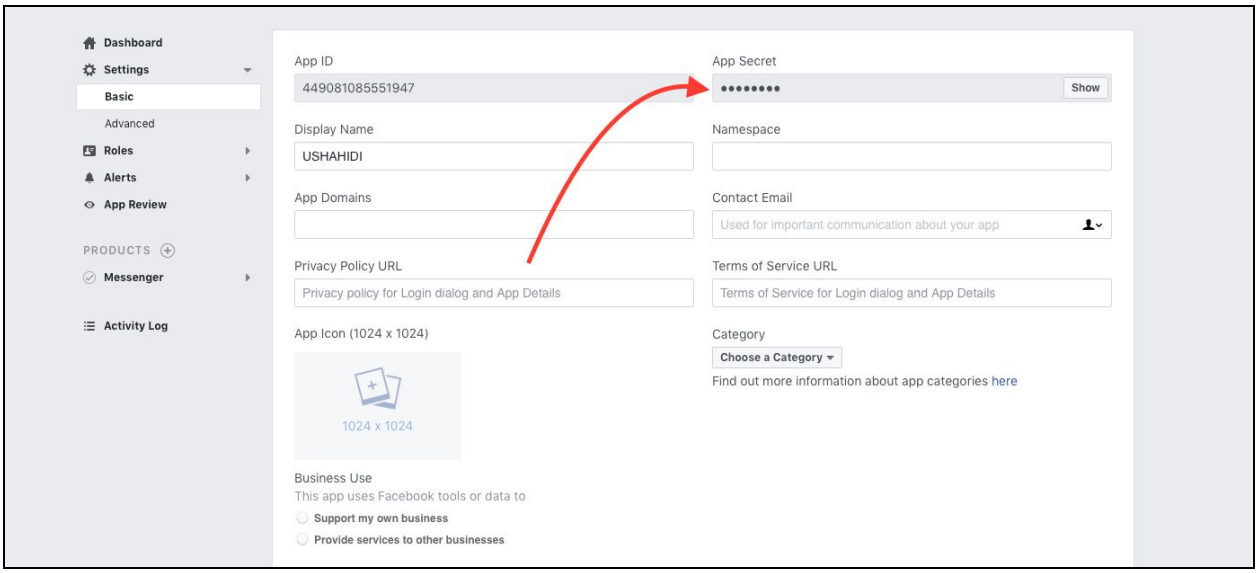
In order to get an access-token, you need to connect it to a facebook-page. If you don't have one already, go to the "normal" facebook (not the developer-one) and create a page:



After that, go back to developer-facebook and your app, in the messenger-settings, you can select your page and generate an access-token:



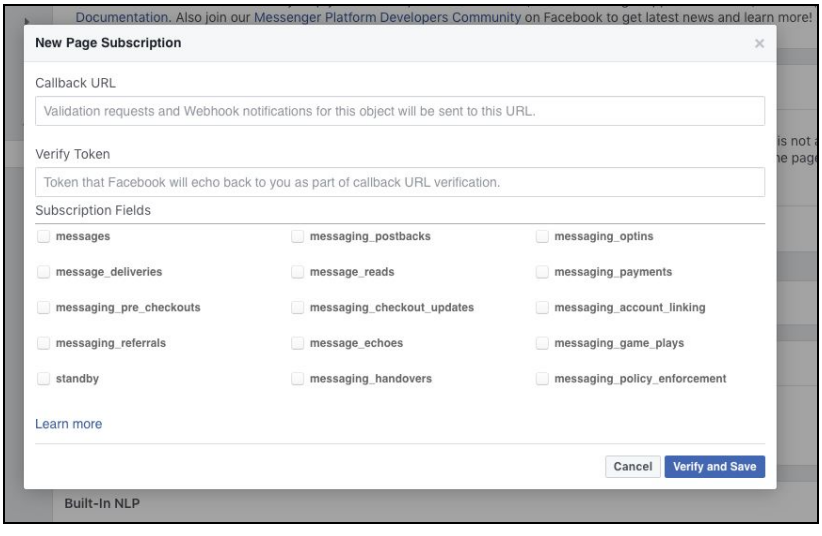
Add the access-token to your .env file together with a verify-token of your choice and the facebook secret token found here:



When all those credentials are in your .env-file, together with the credentials for the Ushahidi-deployment you want to use to send reports with, you start the application with

- vagrant up and then
- artisan php migrate --seed.

Now go back to developers.facebook.com and click on Messenger->settings to set up your webhooks.



Use your-url/webhook as callback-url and verify-token is the same as you choose in the .env file. Select messages and messaging_postbacks. Click on verify and save.

Go to your page and send a message to the page to start talking to the bot. It is good to add a start-button and a persistent-menu, instructions for those could be found in the facebook-documentation available at the following URLs (accessed 30 January 2019) :

- https://developers.facebook.com/docs/messenger-platform/reference/messenger-profile-api/get-started-button
- https://developers.facebook.com/docs/messenger-platform/send-messages/persistent-menu

Facebook Bot-script

When the user start talking to the bot:

- Hello! Welcome to {TITLE}
- {NAME} is creating an open-source resilience platform help communities reconnect, respond to, and recover from crisis situations.
- Help us test the platform by sending us a report!
- [Send a report] ← this is a button the user can click on

When the user clicks "Send a report":

- In a few words, describe what you would like to report.

After the user writes a description he/she gets prompted with:

- Anything else? If not, do you want to add a location or image to your report?

[Send my report] ← this is a button the user can click on

[Add an image] ← this is a button the user can click on

[Add a location] ← this is a button the user can click on

User selects option "add an image":

- Ok great! Just add a photo here like a normal chat, and I'll attach it to your report

The user uploads it and the bot replies with:

- Ok, got it. Do you want to add a location to your report?
- [Send my report] ← this is a button the user can click on
- [Add a location] ← this is a button the user can click on

Or (if location is previously added):

- Perfect! Do you want to send your report to us?
- [Send my report] ← this is a button the user can click on
- [Add an image] ← this is a button the user can click on

User selects option "add a location"

- Please add your location below

User selects location on a map

- Ok, got it. Do you want to add an image to your report?

Or (if image is previously added) :

- Perfect! Do you want to send your report to us?

User selects option “Send my report

- Your report has been saved. Nice work!
- A moderator will check your report before it is published to {NAME}
- What do you want to do next?

[Send another report] ← this is a button the user can click on (report-flow starts again)

[Go to {NAME}]← this is a button the user can click on (link to the deployment)

Configuring Data Export to the Humanitarian Data Exchange (HDX)

What is HDX?

HDX is an open platform for sharing data across crises and organisations. Its goal is to make humanitarian data easy to find and use for analysis. HDX is managed by the United Nations Office for the Coordination of Humanitarian Affairs (OCHA) and is located in The Hague. OCHA is part of the United Nations Secretariat and is responsible for bringing together humanitarian actors to ensure a coherent response to emergencies. HXL is a lightweight, data tagging standard that allows for data to be quickly harmonized, concatenated, and compared. The standard was developed by representatives from the Humanitarian Innovation Fund, IOM, OCHA, Save the Children, IFRC, UNHCR, UNICEF, USAID, the World Bank, and the World Food Program.

The COMRADES platform has integrated with the United Nations' Humanitarian Data Exchange (HDX) and supports the associated Humanitarian Exchange Language (HXL). This integration ensures that COMRADES users can export data sets in an acknowledged humanitarian data standard, to a secure and recognized humanitarian data sharing exchange that is in use by over 200 of the world's most important humanitarian organizations.

Users with both a COMRADES deployment and an HDX account can export data from their deployment into their HDX account. The export can be appropriately tagged with HXL to allow the data to work with HDX's suite of online tools including "quick charts" functionality. Building this functionality ensures that users' data are stored securely even if the deployment is discontinued and is easily discoverable by a significant portion of the humanitarian community.

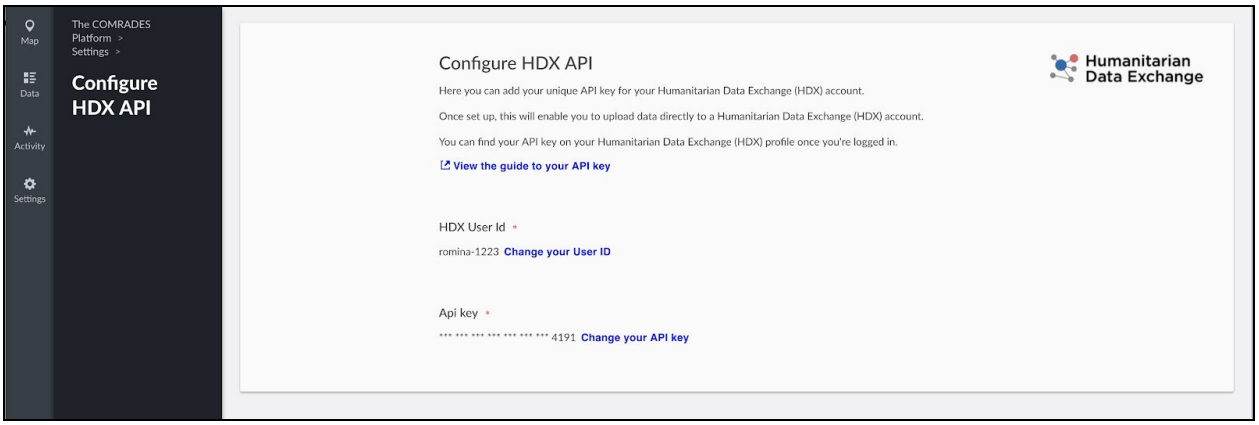
How to use exports and HDX

The platform has 3 ways to export data. You can choose to export data as a plain CSV file, download a CSV export file that includes a row for HXL tags and attributes, or upload the export to the HDX platform. The following section explains the different options for exporting data and how to use them.

HDX Upload and Exports

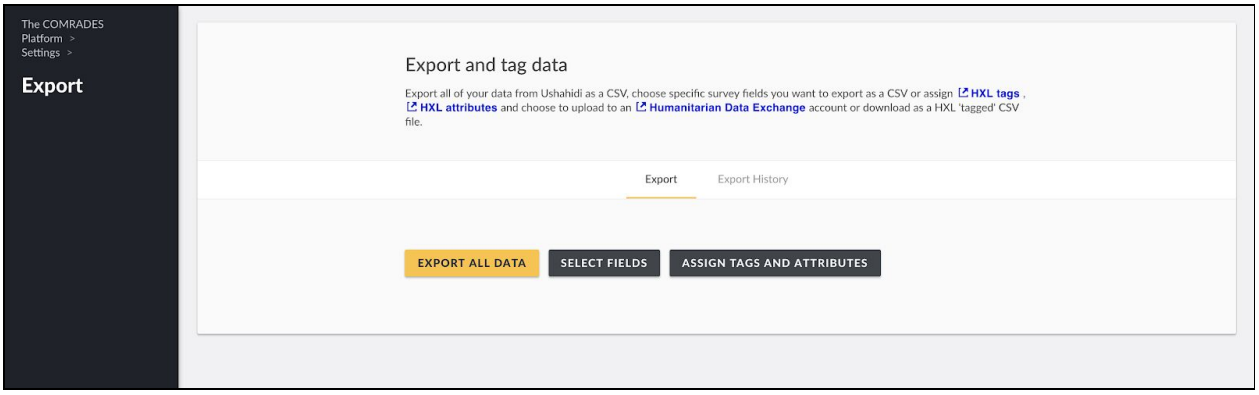
Configure HDX API Screen

Before starting a HDX upload, you need to configure the HDX User Id and Api Key in the "Configure HDX API" screen, accessible by the Settings list or in the link that appears while trying to start a HDX export for the first time. This information is saved and used for each specific user when uploading to HDX, instead of being deployment-wide settings.



Starting an Export

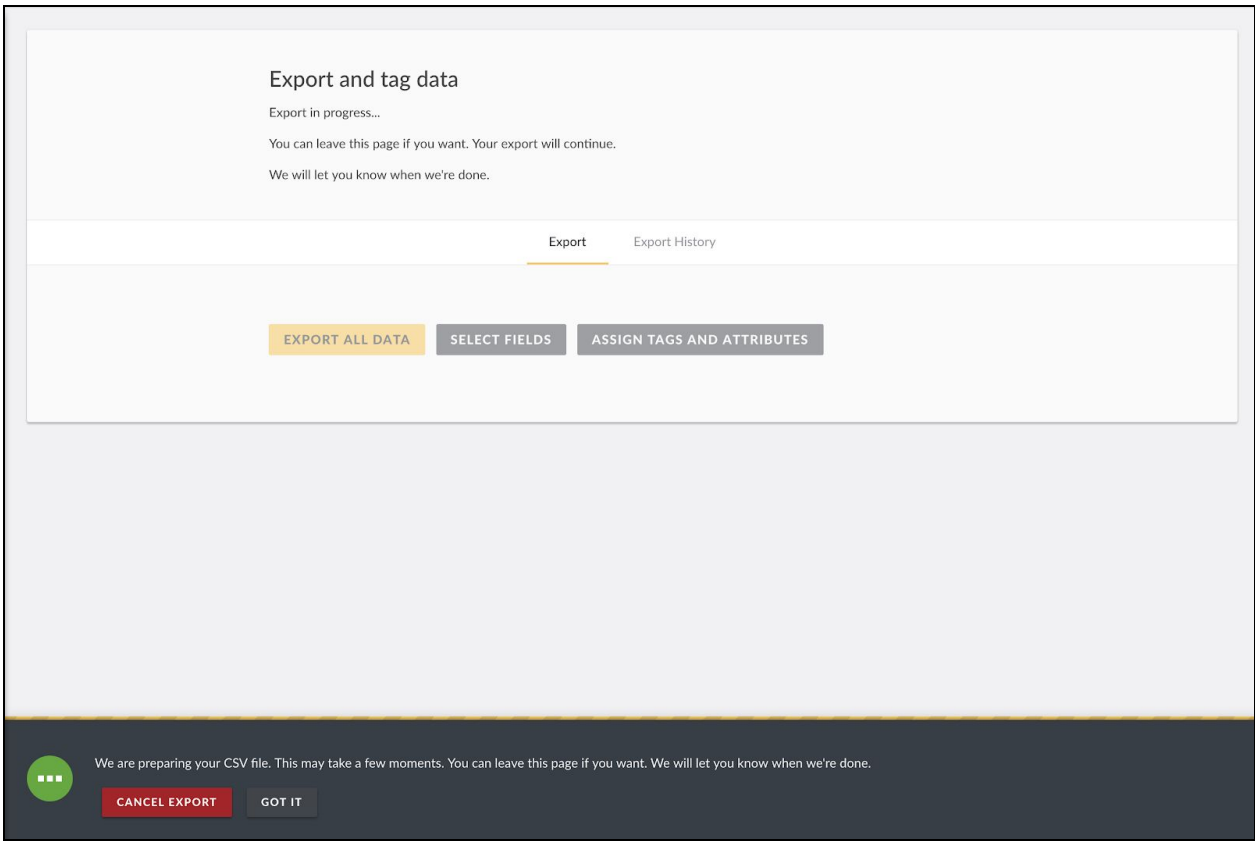
To export data, go to *Settings* and click the *Export and Tag Data* link.



In the *Export and Tag Data* screen you will see three buttons:

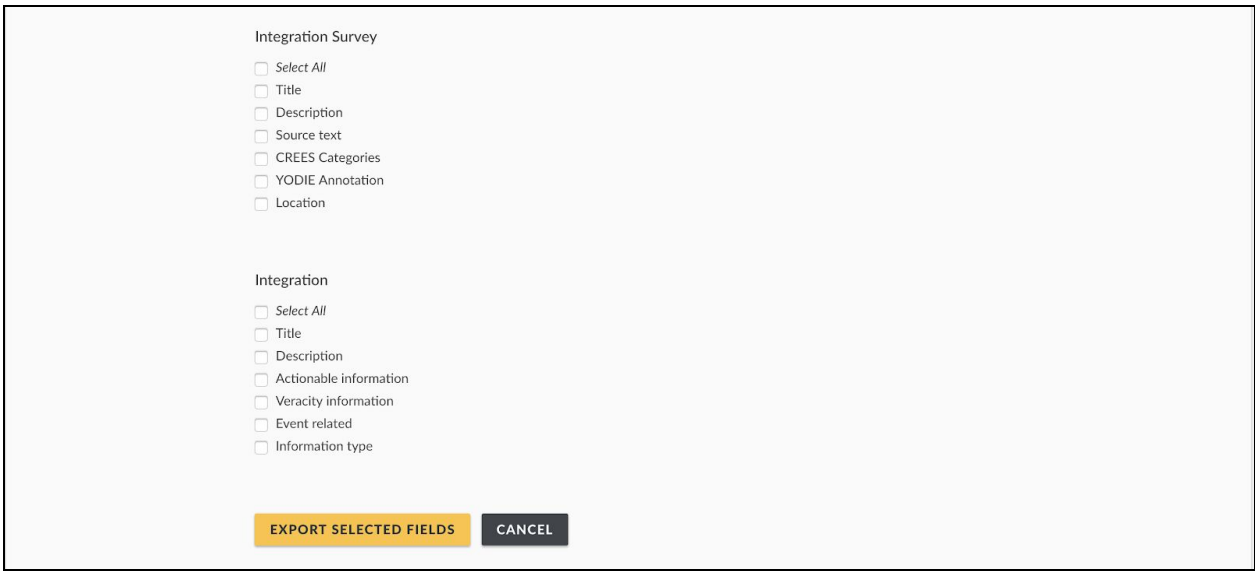
Button: "Export All Data"

This will generate a CSV file with all the posts in your deployment that you have access to. It will take a few minutes, and you can leave the screen while it's being processed by the system. When the export is completed, you will see a notification and the option to download the file in your browser. The screenshots below displays the initial you will see when you request an export, and the notification once it's finished.



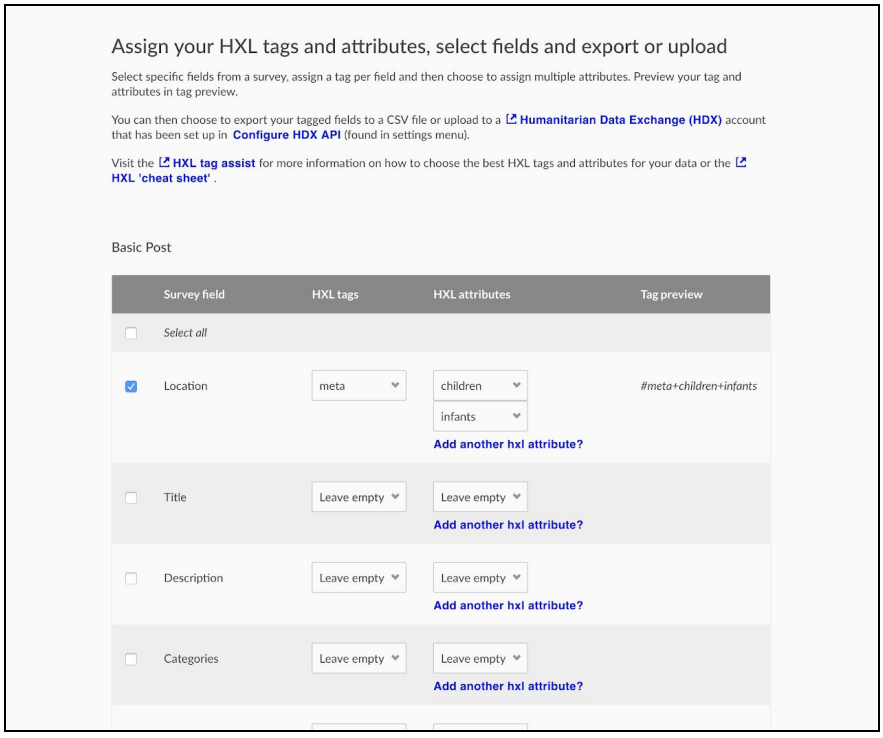
Button: "Select fields"

This will display all the fields and surveys, and after selecting the ones you want to export and clicking the "Export Selected Fields" button, it will generate a CSV file that only contains the fields and surveys selected, plus a few default fields that are included in all exports such as Survey and Post Id. The screenshot below shows the list of fields with checkbox to select each of them.

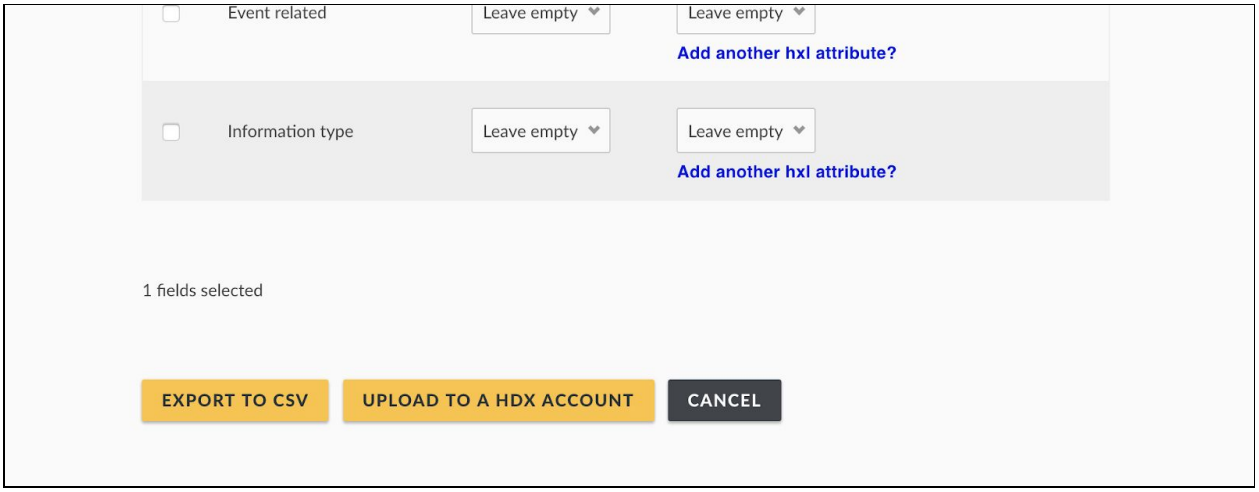


Button: "Assign tags and attributes" (HDX Export)

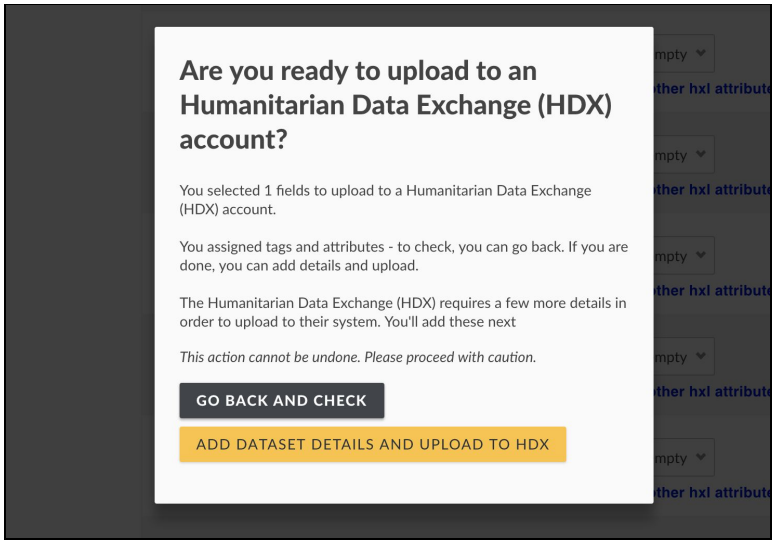
Click this button to configure and start an HDX export. It will take you to a screen where you can select the fields to be included and what HXL attributes and tags they map to. For a field to be included in the export, it has to be selected in the checkboxes to the left of each field. You can also click "Select All" for each survey if you want all its fields to be added.



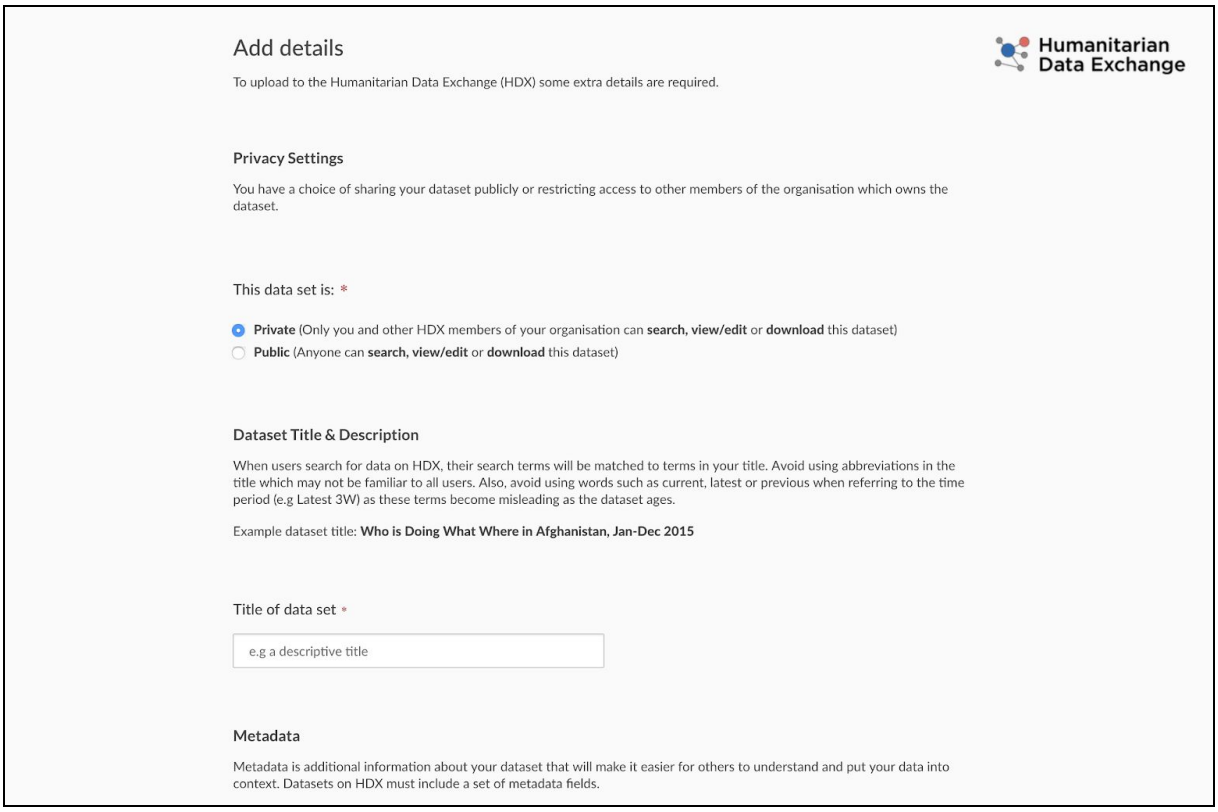
After you are done selecting the fields you want to add to your HDX export, you have the option to export it as a regular file (this will generate a CSV with HXL tags, and behaves like the regular export) or to upload it to an HDX account.
Click on the “UPLOAD TO HDX ACCOUNT” button to continue.

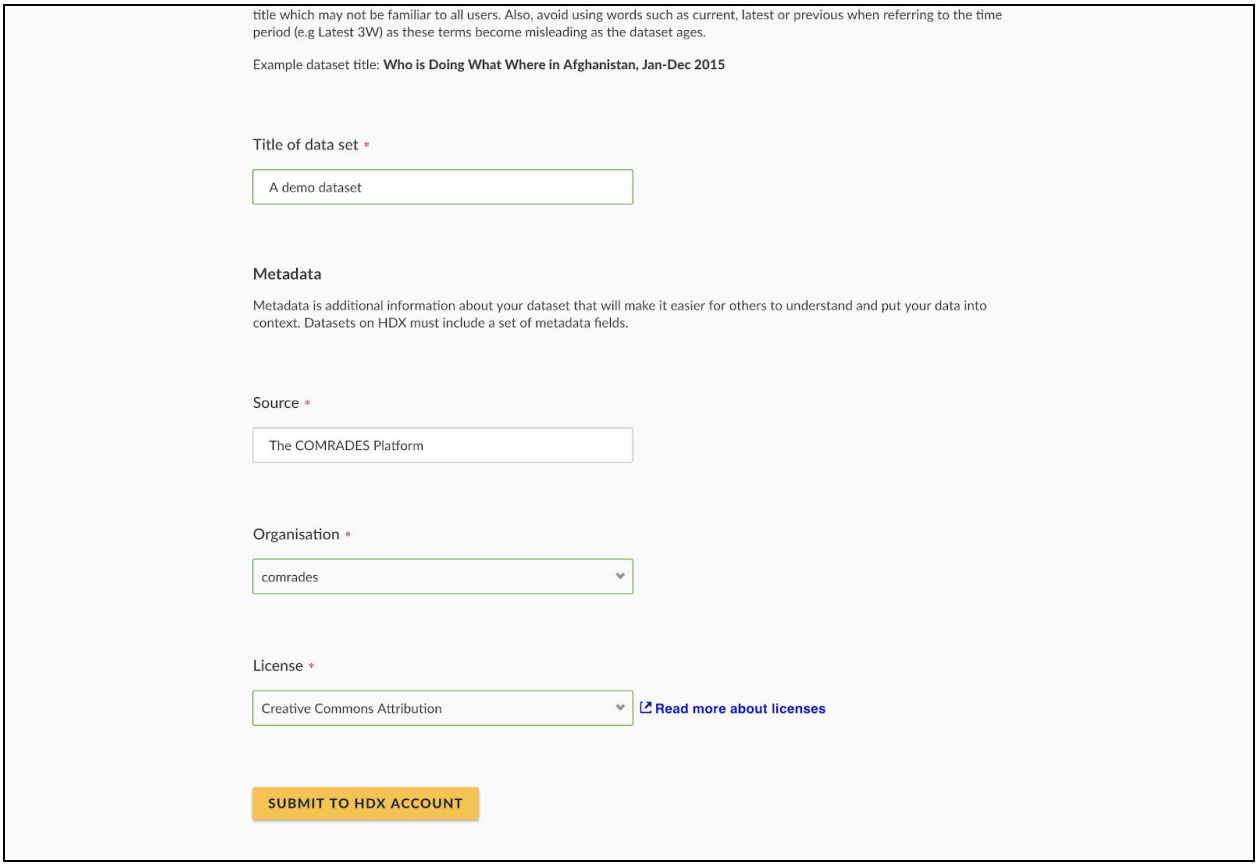


You will see a warning to verify your content before you continue, and you can either check again or continue the process.

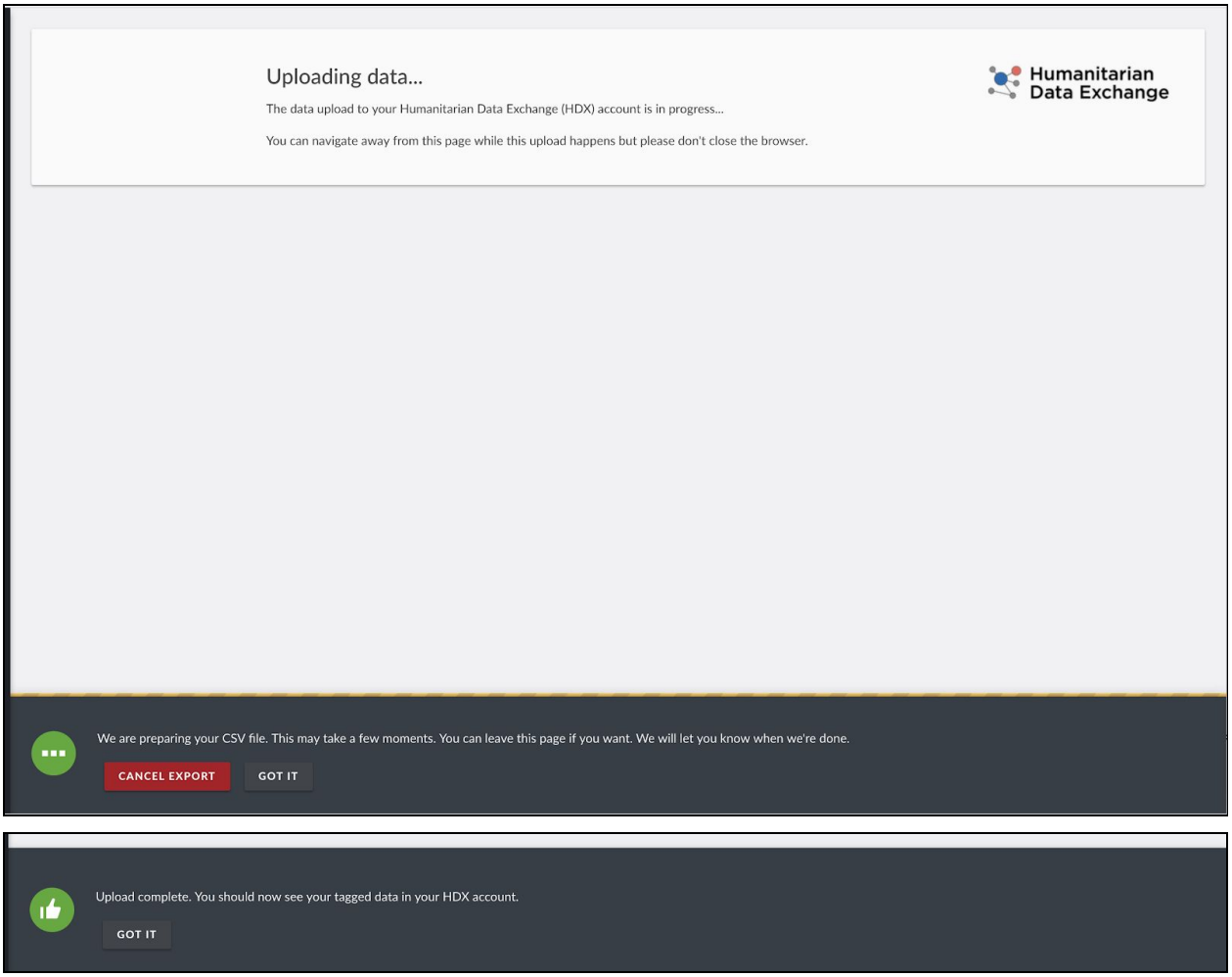


Click on the “ADD DATASET DETAILS AND UPLOAD TO HDX” button to continue. You will see the HDX configuration screen where you can select the options that will be applied when your dataset is uploaded to your HDX account.

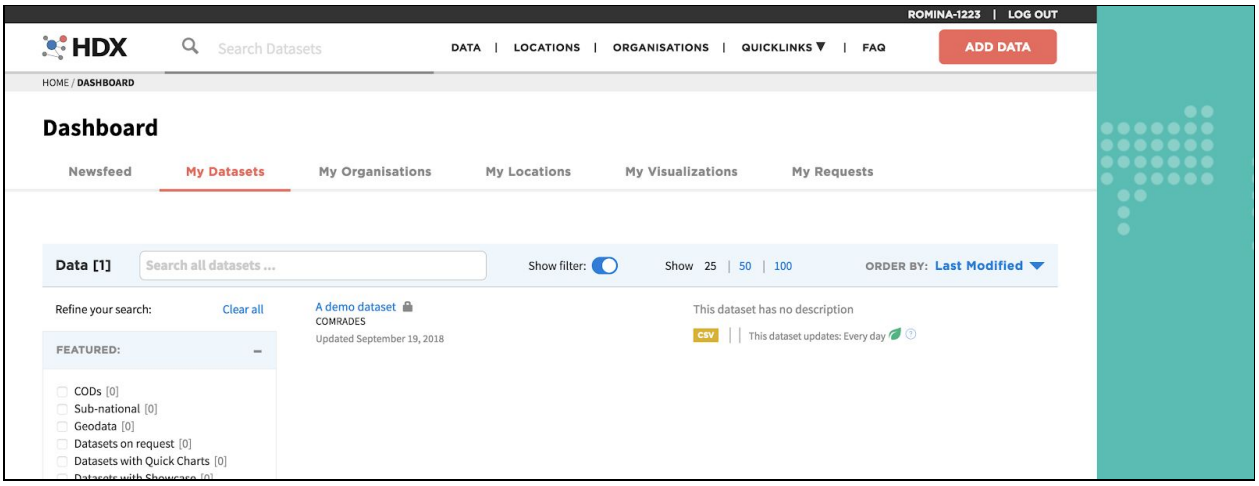

Add details
Humanitarian Data Exchange
To upload to the Humanitarian Data Exchange (HDX) some extra details are required.
Privacy Settings
You have a choice of sharing your dataset publicly or restricting access to other members of the organisation which owns the dataset.
This data set is: *
Private (Only you and other HDX members of your organisation can search, view/edit or download this dataset)
Public (Anyone can search, view/edit or download this dataset)
Dataset Title & Description
When users search for data on HDX, their search terms will be matched to terms in your title. Avoid using abbreviations in the title which may not be familiar to all users. Also, avoid using words such as current, latest or previous when referring to the time period (e.g Latest 3W) as these terms become misleading as the dataset ages.
Example dataset title: Who is Doing What Where in Afghanistan, Jan-Dec 2015
Title of data set *
e.g a descriptive title
Metadata
Metadata is additional information about your dataset that will make it easier for others to understand and put your data into context. Datasets on HDX must include a set of metadata fields.

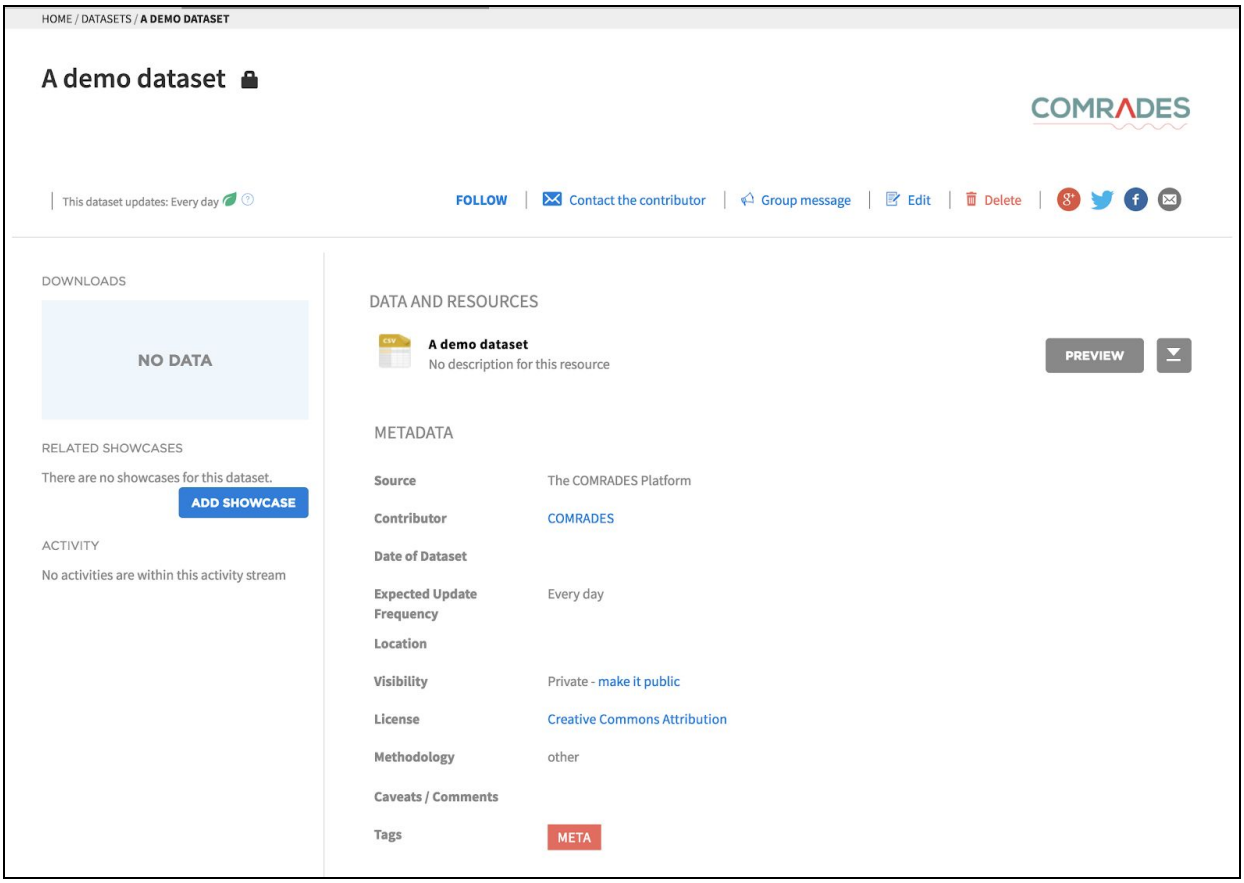

title which may not be familiar to all users. Also, avoid using words such as current, latest or previous when referring to the time period (e.g Latest 3W) as these terms become misleading as the dataset ages.
Example dataset title: Who is Doing What Where in Afghanistan, Jan-Dec 2015
Title of data set *
A demo dataset
Metadata
Metadata is additional information about your dataset that will make it easier for others to understand and put your data into context. Datasets on HDX must include a set of metadata fields.
Source *
The COMRADES Platform
Organisation *
comrades
License *
Creative Commons Attribution
Read more about licenses
SUBMIT TO HDX ACCOUNT

The organizations available in the Organisation list are all the organisations your HDX account has access to. In the screenshot above, we selected the Comrades organisation, a CCA license and left the Source as the default value (deployment name). When you are happy with the configuration, click on SUBMIT TO HDX ACCOUNT. You will see a notification that the export is being prepared, and can leave the page and continue working while it is being generated. You will be notified once the upload is ready.

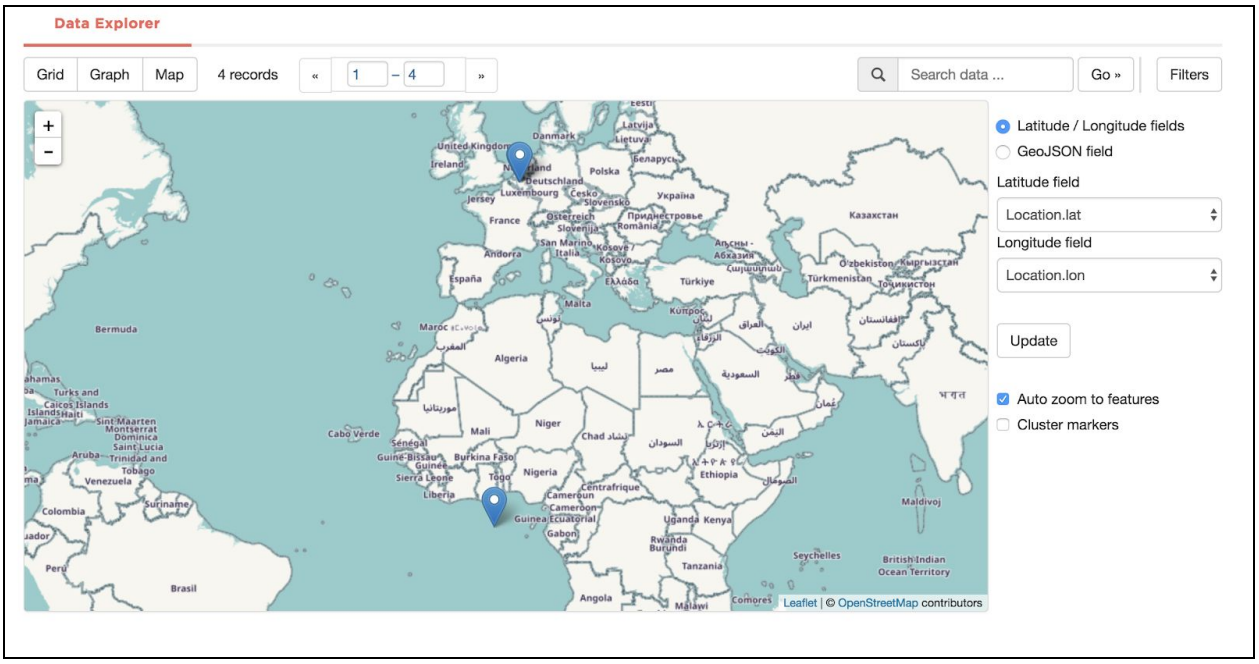


Once the HDX upload finishes, you can see the exported data in your HDX account by logging in and going to the “My datasets” page in HDX https://data.humdata.org/dashboard/datasets
If you want to see the details of your dataset, you can click on the blue link that has the name of your dataset.





Now that your dataset is available in the HDX page, you can click on PREVIEW to see other visualization options. In the following screenshot you can see the exported data in the map (2 pins) since we had 2 posts with a location.



How to setup HDX for the platform

In the platform api repo, verify the following value is in the `.env` file, or add it if it's not present.
`HDX_URL=https://data.humdata.org`

To manage Exports the COMRADES Platform API uses Laravel Artisan Queues, it is best to run this as a background task on the server where the API is deployed. To do this:

1. `sudo apt-get install supervisor`
2. Follow the steps to configure supervisor
3. The appropriate configuration for the COMRADES Platform API is as shown below

```
[program:service-worker-{{ api_worker_id }}]
 process_name=%(program_name)s_%(process_num)02d
 command=php {{ api_base_path }}/artisan queue:work --sleep=3 --tries=3
--daemon
 autostart=true
 autorestart=true
 user={{ www_user }}
 numprocs=4
 redirect_stderr=true
 stdout_logfile={{ api_base_path }}/storage/logs/worker.log
```

You will need to provide values for:

- `Api_base_path`, should be the root directory for your Platform installation
- `www_user`, the appropriate user account you want to run the queues under, **it is best not to run any tools as the *root* user**

Bibliography

Burel, G; Saif, H.; Fernandez, M. and Alani, H. (2017). *On Semantics and Deep Learning for Event Detection in Crisis Situations*. In: Workshop on Semantic Deep Learning (SemDeep), at ESWC 2017, 29 May 2017, Portoroz, Slovenia.

Derczynski, L.; Meesters, K.; Bontcheva, K. and Maynard, D (2018). *Helping crisis responders find the informative needle in the tweet haystack*. In Iscram 2018 conference proceedings 15th international conference on information systems for crisis response and management.

Gorrell, G.; Petrak, J.; and Bontcheva K. (2015). *Using @Twitter Conventions to Improve #LOD-based Named Entity Disambiguation*. In The Semantic Web. Latest Advances and New Domains, pp. 171-186. Springer