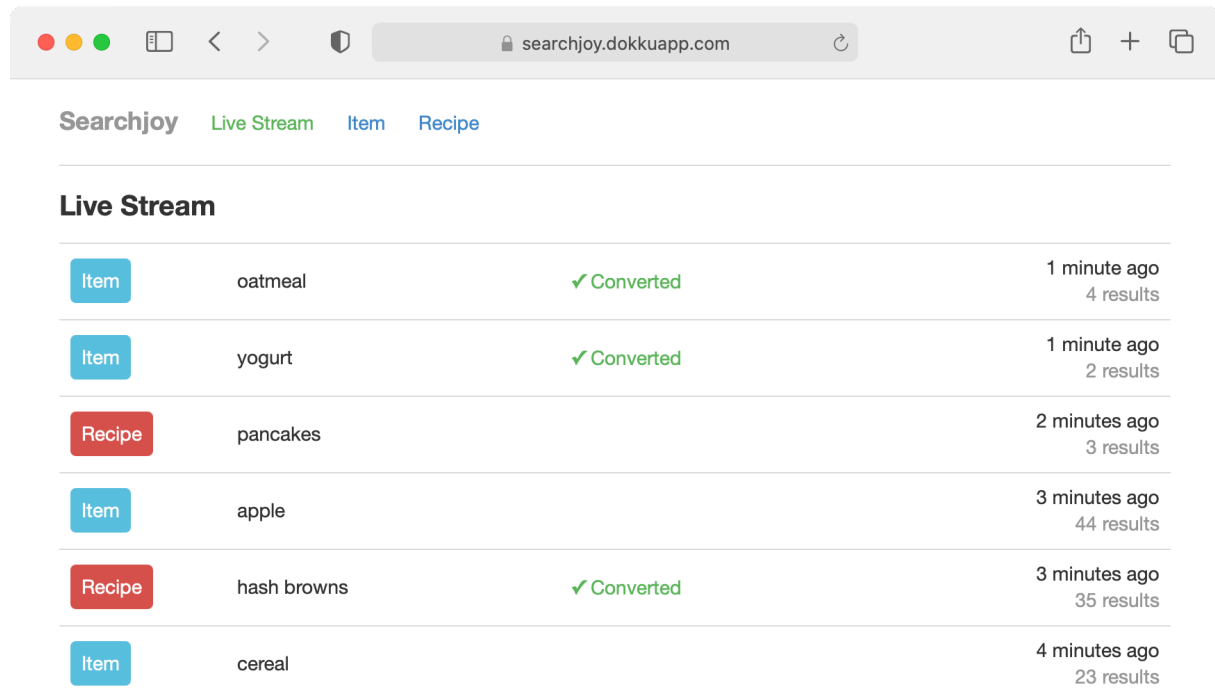

Searchjoy

Search analytics made easy

See it in action



The screenshot shows a web browser window with the URL `searchjoy.dokkuapp.com`. The application has a navigation bar with links for **Searchjoy**, **Live Stream**, **Item**, and **Recipe**. Below the navigation bar, the **Live Stream** section displays a table of search results. The table has four columns: a category button (Item or Recipe), the search term, a status (green checkmark and 'Converted'), and the time since the search along with the number of results.

Category	Search Term	Status	Time	Results
Item	oatmeal	✓ Converted	1 minute ago	4 results
Item	yogurt	✓ Converted	1 minute ago	2 results
Recipe	pancakes		2 minutes ago	3 results
Item	apple		3 minutes ago	44 results
Recipe	hash browns	✓ Converted	3 minutes ago	35 results
Item	cereal		4 minutes ago	23 results

- view searches in real-time
- track conversion rate week over week
- monitor the performance of top searches

Works with any search platform, including Elasticsearch, OpenSearch, Sphinx, and Solr

:cupid: An amazing companion to Searchkick



Installation

Add this line to your application's Gemfile:

```
1 gem "searchjoy"
```

And run the generator. This creates a migration to store searches.

```
1 rails generate searchjoy:install
2 rails db:migrate
```

Next, add the dashboard to your `config/routes.rb`.

```
1 mount Searchjoy::Engine, at: "searchjoy"
```

Be sure to protect the endpoint in production - see the Authentication section for ways to do this.

Track Searches

Track searches by creating a record in the database.

```
1 Searchjoy::Search.create(  
2   search_type: "Item", # typically the model name  
3   query: "apple",  
4   results_count: 12,  
5   user_id: 1  
6 )
```

With Searchkick, you can use the `track` option to do this automatically.

```
1 Item.search("apple", track: {user_id: 1})
```

If you want to track more attributes, add them to the `searchjoy_searches` table.

```
1 add_column :searchjoy_searches, :source, :string
```

Then, pass the values to the `track` option.

```
1 Item.search("apple", track: {user_id: 1, source: "web"})
```

Track Conversions

First, choose a conversion metric. At Instacart, an item added to the cart from the search results page counts as a conversion.

Next, when a user searches, keep track of the search id. With Searchkick, you can get the id with:

```
1 results = Item.search("apple", track: true)  
2 results.search.id
```

When a user converts, find the record and call `convert`.

```
1 search = Searchjoy::Search.find(params[:id])  
2 search.convert
```

Better yet, record the model that converted.

```
1 search.convert(item)
```

Authentication

Don't forget to protect the dashboard in production.

Devise

In your `config/routes.rb`:

```
1 authenticate :user, ->(user) { user.admin? } do
2   mount Searchjoy::Engine, at: "searchjoy"
3 end
```

Basic Authentication

Set the following variables in your environment or an initializer.

```
1 ENV["SEARCHJOY_USERNAME"] = "andrew"
2 ENV["SEARCHJOY_PASSWORD"] = "secret"
```

Data Retention

Data should only be retained for as long as it's needed. Delete older data with:

```
1 Searchjoy::Search.where("created_at < ?", 1.year.ago).find_in_batches
  do |searches|
2   search_ids = searches.map(&:id)
3   Searchjoy::Conversion.where(search_id: search_ids).delete_all
4   Searchjoy::Search.where(id: search_ids).delete_all
5 end
```

You can use Rollup to aggregate important data before you do.

```
1 Searchjoy::Search.rollup("Searches")
```

Delete data for a specific user with:

```
1 user_id = 123
2 search_ids = Searchjoy::Search.where(user_id: user_id).pluck(:id)
3 Searchjoy::Conversion.where(search_id: search_ids).delete_all
4 Searchjoy::Search.where(id: search_ids).delete_all
```

Customize

To customize, create an initializer `config/initializers/searchjoy.rb`.

Change the time zone

```
1 Searchjoy.time_zone = "Pacific Time (US & Canada)" # defaults to Time.  
  zone
```

Change the number of top searches shown

```
1 Searchjoy.top_searches = 500 # defaults to 100
```

Link to the search results

```
1 Searchjoy.query_url = ->(search) { Rails.application.routes.url_helpers  
  .items_path(q: search.query) }
```

Add additional info to the query in the live stream

```
1 Searchjoy.query_name = ->(search) { "#{search.query} #{search.city}" }
```

Show the conversion name in the live stream

```
1 Searchjoy.conversion_name = ->(model) { model.name }
```

Upgrading

1.0

Searchjoy now supports multiple conversions per search :tada:

Before updating the gem, create a migration with:

```
1 create_table :searchjoy_conversions do |t|  
2   t.references :search  
3   t.references :convertable, polymorphic: true, index: {name: "  
    index_searchjoy_conversions_on_convertable"}  
4   t.datetime :created_at  
5 end
```

Deploy and run the migration, then update the gem.

You can optionally backfill the conversions table

```
1 Searchjoy.backfill_conversions
```

And optionally remove `convertable` from searches

```
1 remove_reference :searchjoy_searches, :convertable, polymorphic: true
```

You can stay with single conversions (and skip all the previous steps) by creating an initializer with:

```
1 Searchjoy.multiple_conversions = false
```

History

[View the changelog](#)

Contributing

Everyone is encouraged to help improve this project. Here are a few ways you can help:

- Report bugs
- Fix bugs and submit pull requests
- Write, clarify, or fix documentation
- Suggest or add new feature

To get started with development and testing:

```
1 git clone https://github.com/ankane/searchjoy.git
2 cd searchjoy
3 bundle install
4 bundle exec rake test
```