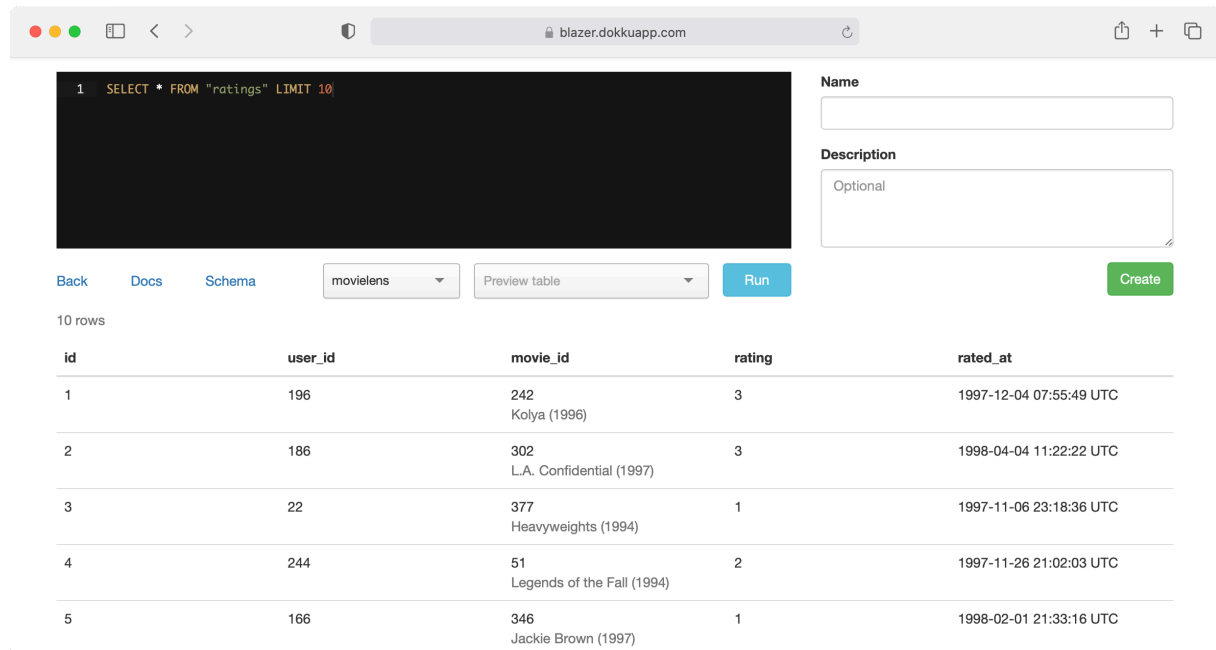

Blazer

Explore your data with SQL. Easily create charts and dashboards, and share them with your team.

Try it out



The screenshot shows the Blazer web interface in a browser window. The address bar shows `blazer.dokkuapp.com`. The main area has a dark SQL editor with the query `1 SELECT * FROM "ratings" LIMIT 10;`. To the right of the editor are input fields for 'Name' and 'Description' (with 'Optional' as a placeholder). Below the editor are tabs for 'Back', 'Docs', and 'Schema', a dropdown menu currently showing 'movielens', a 'Preview table' button, a 'Run' button, and a green 'Create' button. Below these elements, it says '10 rows' and displays a table with 5 rows of movie ratings.

id	user_id	movie_id	rating	rated_at
1	196	242 Kolya (1996)	3	1997-12-04 07:55:49 UTC
2	186	302 L.A. Confidential (1997)	3	1998-04-04 11:22:22 UTC
3	22	377 Heavyweights (1994)	1	1997-11-06 23:18:36 UTC
4	244	51 Legends of the Fall (1994)	2	1997-11-26 21:02:03 UTC
5	166	346 Jackie Brown (1997)	1	1998-02-01 21:33:16 UTC

Blazer is also available as a Docker image.

:tangerine: Battle-tested at Instacart



Features

- **Multiple data sources** - PostgreSQL, MySQL, Redshift, and many more
- **Variables** - run the same queries with different values
- **Checks & alerts** - get emailed when bad data appears
- **Audits** - all queries are tracked
- **Security** - works with your authentication system

Docs

- Installation
- Queries

-
- Charts
 - Dashboards
 - Checks
 - Cohorts
 - Anomaly Detection
 - Forecasting
 - Uploads
 - Data Sources
 - Query Permissions

Installation

Add this line to your application's Gemfile:

```
1 gem "blazer"
```

Run:

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

And mount the dashboard in your `config/routes.rb`:

```
1 mount Blazer::Engine, at: "blazer"
```

For production, specify your database:

```
1 ENV["BLAZER_DATABASE_URL"] = "postgres://user:password@hostname:5432/
  database"
```

When possible, Blazer tries to protect against queries which modify data by running each query in a transaction and rolling it back, but a safer approach is to use a read-only user. See how to create one.

Checks (optional) Be sure to set a host in `config/environments/production.rb` for emails to work.

```
1 config.action_mailer.default_url_options = {host: "blazer.dokkuapp.com"
  }
```

Schedule checks to run (with cron, Heroku Scheduler, etc). The default options are every 5 minutes, 1 hour, or 1 day, which you can customize. For each of these options, set up a task to run.

```
1 rake blazer:run_checks SCHEDULE="5 minutes"
2 rake blazer:run_checks SCHEDULE="1 hour"
3 rake blazer:run_checks SCHEDULE="1 day"
```

You can also set up failing checks to be sent once a day (or whatever you prefer).

```
1 rake blazer:send_failing_checks
```

Here's what it looks like with cron.

```
1 */5 * * * * rake blazer:run_checks SCHEDULE="5 minutes"
2 0 * * * * rake blazer:run_checks SCHEDULE="1 hour"
3 30 7 * * * rake blazer:run_checks SCHEDULE="1 day"
4 0 8 * * * rake blazer:send_failing_checks
```

For Slack notifications, create an incoming webhook and set:

```
1 BLAZER_SLACK_WEBHOOK_URL=https://hooks.slack.com/...
```

Name the webhook “Blazer” and add a cool icon.

Authentication

Don't forget to protect the dashboard in production.

Basic Authentication

Set the following variables in your environment or an initializer.

```
1 ENV["BLAZER_USERNAME"] = "andrew"
2 ENV["BLAZER_PASSWORD"] = "secret"
```

Devise

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

Other

Specify a `before_action` method to run in `blazer.yml`.

```
1 before_action_method: require_admin
```

You can define this method in your ApplicationController.

```
1 def require_admin
2   # depending on your auth, something like...
3   redirect_to root_path unless current_user && current_user.admin?
4 end
```

Be sure to render or redirect for unauthorized users.

Permissions

PostgreSQL

Create a user with read-only permissions:

```
1 BEGIN;
2 CREATE ROLE blazer LOGIN PASSWORD 'secret';
3 GRANT CONNECT ON DATABASE dbname TO blazer;
4 GRANT USAGE ON SCHEMA public TO blazer;
5 GRANT SELECT ON ALL TABLES IN SCHEMA public TO blazer;
6 ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO
   blazer;
7 COMMIT;
```

MySQL

Create a user with read-only permissions:

```
1 CREATE USER 'blazer'@'127.0.0.1' IDENTIFIED BY 'secret';
2 GRANT SELECT, SHOW VIEW ON dbname.* TO 'blazer'@'127.0.0.1';
3 FLUSH PRIVILEGES;
```

Sensitive Data

If your database contains sensitive or personal data, check out Hypershield to shield it.

Encrypted Data

If you need to search encrypted data, use blind indexing.

You can have Blazer transform specific variables with:

```
1 Blazer.transform_variable = lambda do |name, value|
2   value = User.generate_email_bidx(value) if name == "email_bidx"
3   value
4 end
```

Queries

Variables

Create queries with variables.

```
1 SELECT * FROM users WHERE gender = {gender}
```

Use {start_time} and {end_time} for time ranges. Example

```
1 SELECT * FROM ratings WHERE rated_at >= {start_time} AND rated_at <= {
  end_time}
```

Smart Variables

Example

Suppose you have the query:

```
1 SELECT * FROM users WHERE occupation_id = {occupation_id}
```

Instead of remembering each occupation's id, users can select occupations by name.

Add a smart variable with:

```
1 smart_variables:
2   occupation_id: "SELECT id, name FROM occupations ORDER BY name ASC"
```

The first column is the value of the variable, and the second column is the label.

You can also use an array or hash for static data and enums.

```
1 smart_variables:
2   period: ["day", "week", "month"]
3   status: {0: "Active", 1: "Archived"}
```

Linked Columns

Example - title column

Link results to other pages in your apps or around the web. Specify a column name and where it should link to. You can use the value of the result with {value}.

```
1 linked_columns:
2   user_id: "/admin/users/{value}"
3   ip_address: "https://www.infosniper.net/index.php?ip_address={value}"
```

Smart Columns

Example - occupation_id column

Suppose you have the query:

```
1 SELECT name, city_id FROM users
```

See which city the user belongs to without a join.

```
1 smart_columns:
2   city_id: "SELECT id, name FROM cities WHERE id IN {value}"
```

You can also use a hash for static data and enums.

```
1 smart_columns:
2   status: {0: "Active", 1: "Archived"}
```

Caching

Blazer can automatically cache results to improve speed. It can cache slow queries:

```
1 cache:
2   mode: slow
3   expires_in: 60 # min
4   slow_threshold: 15 # sec
```

Or it can cache all queries:

```
1 cache:
2   mode: all
3   expires_in: 60 # min
```

Of course, you can force a refresh at any time.

Charts

Blazer will automatically generate charts based on the types of the columns returned in your query.

Note: The order of columns matters.

Line Chart

There are two ways to generate line charts.

2+ columns - timestamp, numeric(s) - Example

```
1 SELECT date_trunc('week', created_at), COUNT(*) FROM users GROUP BY 1
```

3 columns - timestamp, string, numeric - Example

```
1 SELECT date_trunc('week', created_at), gender, COUNT(*) FROM users
   GROUP BY 1, 2
```

Column Chart

There are also two ways to generate column charts.

2+ columns - string, numeric(s) - Example

```
1 SELECT gender, COUNT(*) FROM users GROUP BY 1
```

3 columns - string, string, numeric - Example

```
1 SELECT gender, zip_code, COUNT(*) FROM users GROUP BY 1, 2
```

Scatter Chart

2 columns - both numeric - Example

```
1 SELECT x, y FROM table
```

Pie Chart

2 columns - string, numeric - and last column named `pie` - Example

```
1 SELECT gender, COUNT(*) AS pie FROM users GROUP BY 1
```

Maps

Columns named `latitude` and `longitude` or `lat` and `lon` or `lat` and `lng` - Example

```
1 SELECT name, latitude, longitude FROM cities
```

or a column named `geojson`

```
1 SELECT name, geojson FROM counties
```

To enable, get an access token from Mapbox and set `ENV["MAPBOX_ACCESS_TOKEN"]`.

Targets

Use the column name `target` to draw a line for goals. Example

```
1 SELECT date_trunc('week', created_at), COUNT(*) AS new_users, 100000 AS
   target FROM users GROUP BY 1
```

Dashboards

Create a dashboard with multiple queries. Example

If the query has a chart, the chart is shown. Otherwise, you'll see a table.

If any queries have variables, they will show up on the dashboard.

Checks

Checks give you a centralized place to see the health of your data. Example

Create a query to identify bad rows.

```
1 SELECT * FROM ratings WHERE user_id IS NULL /* all ratings should have
   a user */
```

Then create check with optional emails if you want to be notified. Emails are sent when a check starts failing, and when it starts passing again.

Cohorts

Create a cohort analysis from a simple SQL query. Example

Create a query with the comment `/* cohort analysis */`. The result should have columns named `user_id` and `conversion_time` and optionally `cohort_time`.

You can generate cohorts from the first conversion time:

```
1 /* cohort analysis */
2 SELECT user_id, created_at AS conversion_time FROM orders
```

(the first conversion isn't counted in the first time period with this format)

Or from another time, like sign up:

```
1 /* cohort analysis */
2 SELECT users.id AS user_id, orders.created_at AS conversion_time, users
   .created_at AS cohort_time
3 FROM users LEFT JOIN orders ON orders.user_id = users.id
```

This feature requires PostgreSQL or MySQL 8.

Anomaly Detection

Blazer supports three different approaches to anomaly detection.

Prophet

Add prophet-rb to your Gemfile:

```
1 gem "prophet-rb"
```

And add to `config/blazer.yml`:

```
1 anomaly_checks: prophet
```

Trend

Trend uses an external service by default, but you can run it on your own infrastructure as well.

Add trend to your Gemfile:

```
1 gem "trend"
```

And add to `config/blazer.yml`:

```
1 anomaly_checks: trend
```

For the self-hosted API, create an initializer with:

```
1 Trend.url = "http://localhost:8000"
```

AnomalyDetection.rb

Add anomaly_detection to your Gemfile:

```
1 gem "anomaly_detection"
```

And add to `config/blazer.yml`:

```
1 anomaly_checks: anomaly_detection
```

Forecasting

Blazer supports for two different forecasting methods. Example

A forecast link will appear for queries that return 2 columns with types timestamp and numeric.

Prophet

Add prophet-rb to your Gemfile:

```
1 gem "prophet-rb", ">= 0.2.1"
```

And add to `config/blazer.yml`:

```
1 forecasting: prophet
```

Trend

Trend uses an external service by default, but you can run it on your own infrastructure as well.

Add trend to your Gemfile:

```
1 gem "trend"
```

And add to `config/blazer.yml`:

```
1 forecasting: trend
```

For the self-hosted API, create an initializer with:

```
1 Trend.url = "http://localhost:8000"
```

Uploads

Create database tables from CSV files. Example

Run:

```
1 rails generate blazer:uploads
2 rails db:migrate
```

And add to `config/blazer.yml`:

```
1 uploads:
2   url: postgres://...
3   schema: uploads
4   data_source: main
```

This feature requires PostgreSQL. Create a new schema just for uploads.

```
1 CREATE SCHEMA uploads;
```

Data Sources

Blazer supports multiple data sources :tada:

Add additional data sources in `config/blazer.yml`:

```
1 data_sources:
2   main:
3     url: <%= ENV["BLAZER_DATABASE_URL"] %>
4     # timeout, smart_variables, linked_columns, smart_columns
5   catalog:
6     url: <%= ENV["CATALOG_DATABASE_URL"] %>
7     # ...
8   redshift:
9     url: <%= ENV["REDSHIFT_DATABASE_URL"] %>
10    # ...
```

Full List

- Amazon Athena
- Amazon Redshift
- Apache Drill

-
- Apache Hive
 - Apache Ignite
 - Apache Spark
 - Cassandra
 - Druid
 - Elasticsearch
 - Google BigQuery
 - IBM DB2 and Informix
 - InfluxDB
 - MySQL
 - Neo4j
 - OpenSearch
 - Oracle
 - PostgreSQL
 - Presto
 - Salesforce
 - Socrata Open Data API (SODA)
 - Snowflake
 - SQLite
 - SQL Server

You can also create an adapter for any other data store.

Note: In the examples below, we recommend using environment variables for urls.

```
1 data_sources:
2   my_source:
3     url: <%= ENV["BLAZER_MY_SOURCE_URL"] %>
```

Amazon Athena

Add aws-sdk-athena and aws-sdk-glue to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     adapter: athena
4     database: database
5
6     # optional settings
7     output_location: s3://some-bucket/
8     workgroup: primary
9     access_key_id: ...
10    secret_access_key: ...
```

```
11     region: ...
```

Here's an example IAM policy:

```
1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Effect": "Allow",
6              "Action": [
7                  "athena:GetQueryExecution",
8                  "athena:GetQueryResults",
9                  "athena:StartQueryExecution"
10             ],
11             "Resource": [
12                 "arn:aws:athena:region:account-id:workgroup/primary"
13             ]
14         },
15         {
16             "Effect": "Allow",
17             "Action": [
18                 "glue:GetTable",
19                 "glue:GetTables"
20             ],
21             "Resource": [
22                 "arn:aws:glue:region:account-id:catalog",
23                 "arn:aws:glue:region:account-id:database/default",
24                 "arn:aws:glue:region:account-id:table/default/*"
25             ]
26         }
27     ]
28 }
```

You also need to configure S3 permissions.

Amazon Redshift

Add `activerecord6-redshift-adapter` or `activerecord5-redshift-adapter` to your Gemfile and set:

```
1  data_sources:
2      my_source:
3          url: redshift://user:password@hostname:5439/database
```

Use a read-only user.

Apache Drill

Add `drill-sergeant` to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     adapter: drill
4     url: http://hostname:8047
```

Use a read-only user.

Apache Hive

Add hexspace to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     adapter: hive
4     url: sasl://user:password@hostname:10000/database
```

Use a read-only user. Requires HiveServer2.

Apache Ignite

Add ignite-client to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     url: ignite://user:password@hostname:10800
```

Use a read-only user (requires a third-party plugin).

Apache Spark

Add hexspace to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     adapter: spark
4     url: sasl://user:password@hostname:10000/database
```

Use a read-only user. Requires the Thrift server.

Cassandra

Add cassandra-driver (and sorted_set for Ruby 3+) to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     url: cassandra://user:password@hostname:9042/keyspace
```

Use a read-only role.

Druid

Enable SQL support on the broker and set:

```
1 data_sources:
2   my_source:
3     adapter: druid
4     url: http://hostname:8082
```

Use a read-only role.

Elasticsearch

Add elasticsearch to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     adapter: elasticsearch
4     url: http://user:password@hostname:9200
```

Use a read-only role.

Google BigQuery

Add google-cloud-bigquery to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     adapter: bigquery
4     project: your-project
5     keyfile: path/to/keyfile.json
```

IBM DB2 and Informix

Add ibm_db to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     url: ibm-db://user:password@hostname:50000/database
```

Use a read-only user.

InfluxDB

Add influxdb to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     adapter: influxdb
4     url: http://user:password@hostname:8086/database
```

Use a read-only user. Supports InfluxQL.

MySQL

Add mysql2 to your Gemfile (if it's not there) and set:

```
1 data_sources:
2   my_source:
3     url: mysql2://user:password@hostname:3306/database
```

Use a read-only user.

Neo4j

Add neo4j-core to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     adapter: neo4j
4     url: http://user:password@hostname:7474
```

Use a read-only user.

OpenSearch

Add opensearch-ruby to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     adapter: opensearch
4     url: http://user:password@hostname:9200
```

Use a read-only user.

Oracle

Add activerecord-oracle_enhanced-adapter and ruby-oci8 to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     url: oracle-enhanced://user:password@hostname:1521/database
```

Use a read-only user.

PostgreSQL

Add pg to your Gemfile (if it's not there) and set:

```
1 data_sources:
2   my_source:
3     url: postgres://user:password@hostname:5432/database
```

Use a read-only user.

Presto

Add presto-client to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     url: presto://user@hostname:8080/catalog
```

Use a read-only user.

Salesforce

Add restforce to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     adapter: salesforce
```

And set the appropriate environment variables:

```
1 SALESFORCE_USERNAME="username"
2 SALESFORCE_PASSWORD="password"
3 SALESFORCE_SECURITY_TOKEN="security token"
4 SALESFORCE_CLIENT_ID="client id"
5 SALESFORCE_CLIENT_SECRET="client secret"
6 SALESFORCE_API_VERSION="41.0"
```

Use a read-only user. Supports SOQL.

Socrata Open Data API (SODA)

Set:

```
1 data_sources:
2   my_source:
3     adapter: soda
4     url: https://soda.demo.socrata.com/resource/4tka-6guv.json
5     app_token: ...
```

Supports SoQL.

Snowflake

First, install ODBC. For Homebrew, use:

```
1 brew install unixodbc
```

For Ubuntu, use:

```
1 sudo apt-get install unixodbc-dev
```

For Heroku, use the Apt buildpack and create an `Aptfile` with:

```
1 unixodbc-dev
2 https://sfc-repo.snowflakecomputing.com/odbc/linux/2.21.5/snowflake-
   odbc-2.21.5.x86_64.deb
```

```
This installs the driver at /app/.apt/usr/lib/snowflake/odbc/lib/libSnowflake
.so
```

Then, download the Snowflake ODBC driver. Add `odbc_adapter` to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     adapter: snowflake
4     conn_str: Driver=/path/to/libSnowflake.so;uid=user;pwd=password;
               server=host.snowflakecomputing.com
```

Use a read-only role.

SQLite

Add `sqlite3` to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     url: sqlite3:path/to/database.sqlite3
```

SQL Server

Add `tiny_tds` and `activerecord-sqlserver-adapter` to your Gemfile and set:

```
1 data_sources:
2   my_source:
3     url: sqlserver://user:password@hostname:1433/database
```

Use a read-only user.

Creating an Adapter

Create an adapter for any data store with:

```
1 class FooAdapter < Blazer::Adapters::BaseAdapter
2   # code goes here
3 end
4
5 Blazer.register_adapter "foo", FooAdapter
```

See the Presto adapter for a good example. Then use:

```
1 data_sources:
2   my_source:
3     adapter: foo
4     url: http://user:password@hostname:9200/
```

Query Permissions

Blazer supports a basic permissions model.

1. Queries without a name are unlisted
2. Queries whose name starts with # are only listed to the creator
3. Queries whose name starts with * can only be edited by the creator

Learn SQL

Have team members who want to learn SQL? Here are a few great, free resources.

- The Data School
- SQLBolt

Useful Tools

For an easy way to group by day, week, month, and more with correct time zones, check out Group-date.sql.

Standalone Version

Looking for a standalone version? Check out Ghost Blazer.

Performance

By default, queries take up a request while they are running. To run queries asynchronously, add to your config:

```
1  async: true
```

Note: Requires caching to be enabled. If you have multiple web processes, your app must use a centralized cache store like Memcached or Redis.

```
1  config.cache_store = :mem_cache_store
```

Archiving

Archive queries that haven't been viewed in over 90 days.

```
1 rake blazer:archive_queries
```

Content Security Policy

If views are stuck with a `Loading...` message, there might be a problem with strict CSP settings in your app. This can be checked with Firefox or Chrome dev tools. You can allow Blazer to override these settings for its controllers with:

```
1 override_csp: true
```

Upgrading

3.0

Maps now use Mapbox GL JS v1 instead of Mapbox.js, which affects Mapbox billing.

2.6

Custom adapters now need to specify how to quote variables in queries (there is no longer a default)

```
1 class FooAdapter < Blazer::Adapters::BaseAdapter
2   def quoting
3     :backslash_escape # single quote strings and convert ' to \' and \
                        to \\
4     # or
5     :single_quote_escape # single quote strings and convert ' to ''
6     # or
7     ->(value) { ... } # custom method
8   end
9 end
```

2.3

To archive queries, create a migration

```
1 rails g migration add_status_to_blazer_queries
```

with:

```
1 add_column :blazer_queries, :status, :string
2 Blazer::Query.update_all(status: "active")
```

2.0

To use Slack notifications, create a migration

```
1 rails g migration add_slack_channels_to_blazer_checks
```

with:

```
1 add_column :blazer_checks, :slack_channels, :text
```

History

[View the changelog](#)

Thanks

Blazer uses a number of awesome open source projects, including Rails, Vue.js, jQuery, Bootstrap, Selectize, StickyTableHeaders, Stupid jQuery Table Sort, and Date Range Picker.

Demo data from MovieLens.

Want to Make Blazer Better?

That's awesome! 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 features

Check out the dev app to get started.