Biff: self-hosted Firebase for Clojure

Jacob O'Bryant | findka.com/biff

What does that even mean?

- Web framework: curates a bunch of Clojure libraries for you
- Implements some of Firebase's core features (like query subscriptions)
- Deployment is easy (comes with Terraform config)

Goal: make web dev as easy as possible for solo developers without compromising on simplicity.

I made Biff because

- Full-time startup founder/bum since Jan 2019
- Many pivots, so got to try a bunch of things
- Favorite parts coalesced into Biff (May 2020)

Getting started

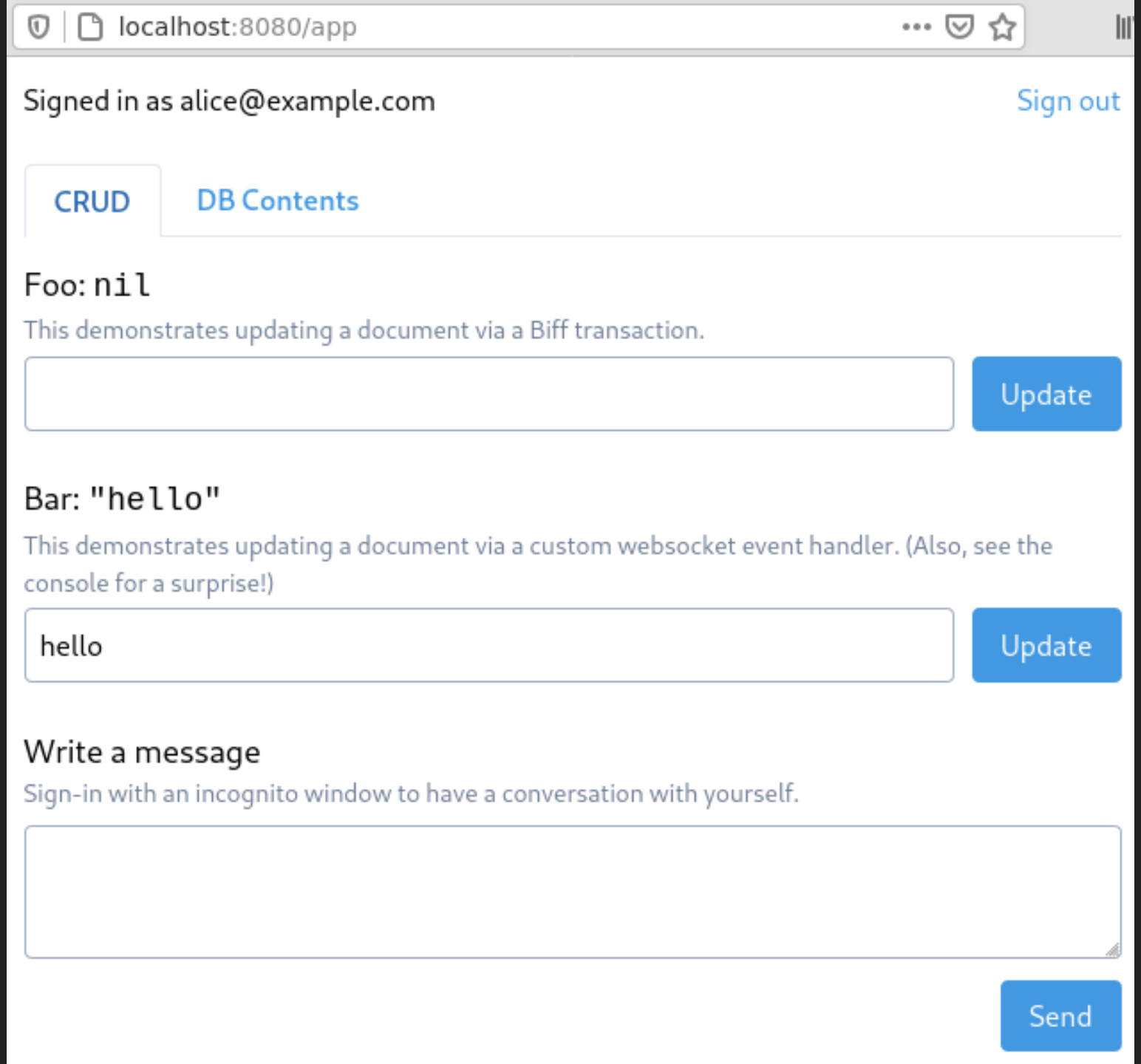
```
bash <(curl -s https://raw.githubusercontent.com/jacobobryant/biff/master/new-project.sh)
```

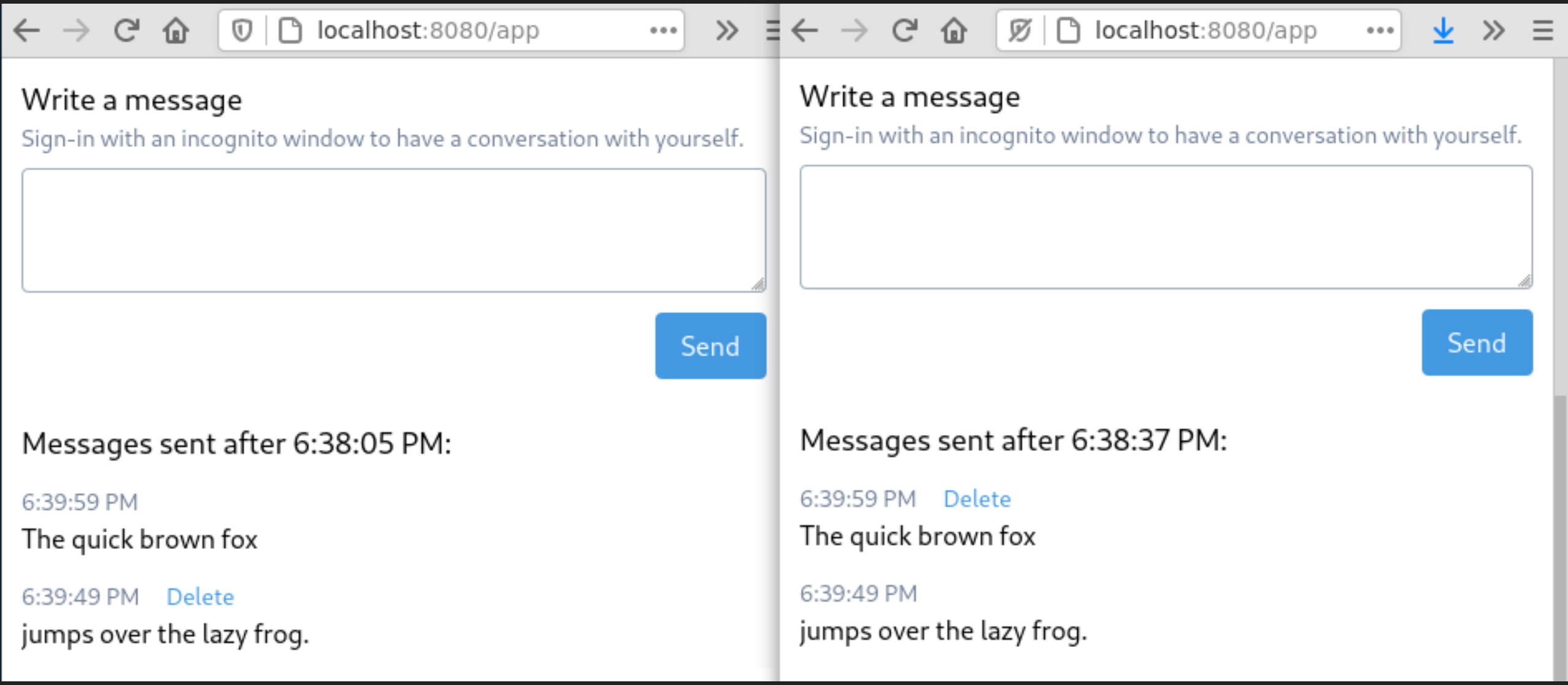
```
Creating a new Biff project. Available project types:

  1. SPA (single-page application). Includes ClojureScript, React, and
     Biff's subscribable queries. Good for highly interactive applications.

  2. MPA (multi-page application). Uses server-side rendering instead of
     React etc. Good for simpler applications.

Choose a project type ([spa]/mpa): spa
Creating a SPA project.
Fetching latest Biff version...
Enter name for project directory: example
Enter main namespace (e.g. example.core): example.core
Enter the domain you plan to use in production (e.g. example.com),
or leave blank to choose later: example.com
```


localhost:8080/app
Signed in as alice@example.com
Sign out
CRUD
DB Contents
Foo: nil
This demonstrates updating a document via a Biff transaction.
Update
Bar: "hello"
This demonstrates updating a document via a custom websocket event handler. (Also, see the console for a surprise!)
hello
Update
Write a message
Sign-in with an incognito window to have a conversation with yourself.
Send


localhost:8080/app
Write a message
Sign-in with an incognito window to have a conversation with yourself.
Send
Messages sent after 6:38:05 PM:
6:39:59 PM
The quick brown fox
6:39:49 PM Delete
jumps over the lazy frog.
localhost:8080/app
Write a message
Sign-in with an incognito window to have a conversation with yourself.
Send
Messages sent after 6:38:37 PM:
6:39:59 PM Delete
The quick brown fox
6:39:49 PM
jumps over the lazy frog.

Biff

A web framework + self-hosted deployment solution for Clojure

Jacob O'Bryant

Introduction

Biff is designed to make web development with Clojure as easy as possible without compromising on simplicity. The main target audience is early stage startups and hobbyists, where speed in the beginning really matters. Biff is also made to be easy to take apart: as your project grows and your requirements become more complex, you can peel back Biff's layers until you have the level of flexibility you need.

Biff is still fairly young, and there may be breaking changes. I've been using it in production for Findka since May 2020. To help Biff grow and to help me discover what needs improvement, I'm giving free one-on-one mentoring (pair programming, code review, design help, etc) to anyone who wants to learn Clojure web dev with Biff (as my schedule allows). If you're interested, fill out this quick survey.

Core features (a few of these were inspired by Firebase):

- **Query subscriptions**. Specify what data the frontend needs declaratively, and Biff will keep it up-to-date. Same level of query power as with Firebase.
- **Authentication**. Email link for now; password and SSO coming later.
- **Read/write authorization rules**. No need to set up a bunch of endpoints for CRUD operations. Queries and transactions can be submitted from the frontend as long as they pass the rules you define.
- **Database triggers**. Run code when documents of certain types are created, updated or deleted.
- **Crux**, an immutable document database with Datalog queries (see opencrux.com).
- **No-hassle deployment** using Terraform and DigitalOcean (you can add config for other cloud providers if needed). Biff can run on a single $5/month server. Later I'll add config for high availability, CI/CD, etc.
- Project templates for **SPAs and MPAs**. If you don't need high interactivity, you can use server-side rendering instead of React and ClojureScript.
- **Great documentation!**

Deployment

See Overview > Infrastructure. When you're ready to deploy, follow these steps:

1. Set up DigitalOcean

Biff comes with Terraform config for DigitalOcean. You can write your own config if you want to use a different provider (see Overview > Decomposing), but for now I'll assume you're using DigitalOcean. If you don't already have an account, you can sign up with my referral link which will give you $100 of credit for 60 days (and $25 for me if you stick with them).

You'll also need a domain that points to DigitalOcean's nameservers.

2. Update config

In `config/main.edn`, make sure `:biff/host` is set to the domain you want to use for your production app (e.g. `myapp.example.com`). If you've changed this since creating your Biff project, run `./task dev` (or `(biff.core/refresh)`) to make sure the Terraform config file (`infra/system.tf.json`) is up-to-date.

In `config/task.env`, update the following environment variables:

```
git push
./task deploy
```

> Libraries are simple, frameworks are easy. Frameworks can be good if they're both simple and easy.

What makes a framework simple?

Decomposability

```clojure
(ns example.core)

(defn -main []
  (biff.core/start-system
    {:biff/routes example.routes/routes
     :biff/static-pages example.static/pages
     :biff/rules #'example.rules/rules
     ...}
    biff.core/default-spa-components))
```

```clojure
(ns biff.core
  (:require [biff.components :as c]))

(def default-spa-components
  [c/init c/set-defaults c/start-crux c/start-sente ...])
```

```clojure
(ns biff.core
  (:require [biff.components :as c]))

(defonce system (atom nil))

(defn refresh []
  (let [{:keys [biff/after-refresh biff/stop]} @system]
    (doseq [f stop]
      (f))
    (clojure.tools.namespace.repl/refresh :after after-refresh)))

(defn start-system [config components]
  (reset! system
    (reduce (fn [sys component]
              (component sys))
      (merge {:biff/stop '()} config)
      components)))

(def default-spa-components
  [c/init c/set-defaults c/start-crux c/start-sente ...])
```

```clojure
(ns biff.components)

(defn start-web-server [{:biff.web/keys [handler host port] :as sys}]
  (let [server (jetty/run-jetty handler
                 {:host host
                  :port port
                  :join? false
                  :websockets {"/api/chsk" handler}
                  :allow-null-path-info true})]
    (update sys :biff/stop conj #(jetty/stop-server server))))
```

Decomposability... and projects that use it.

findka.com/biff

jacobobryant.com