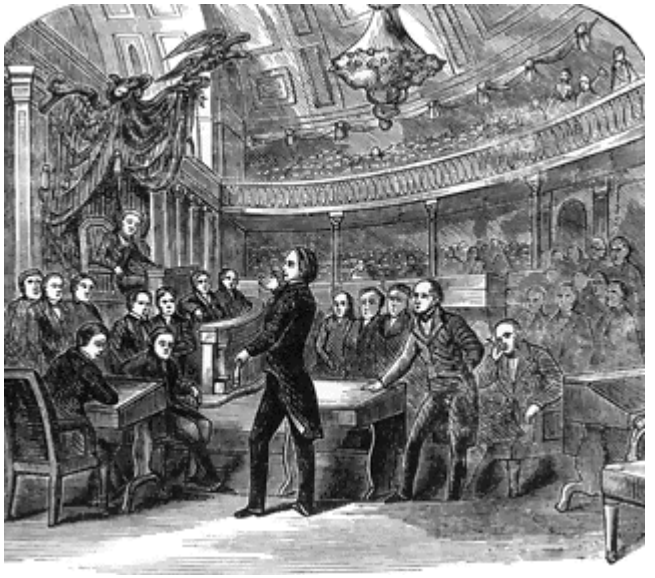
Writing Flask Apps for Fun and Politics



Hi

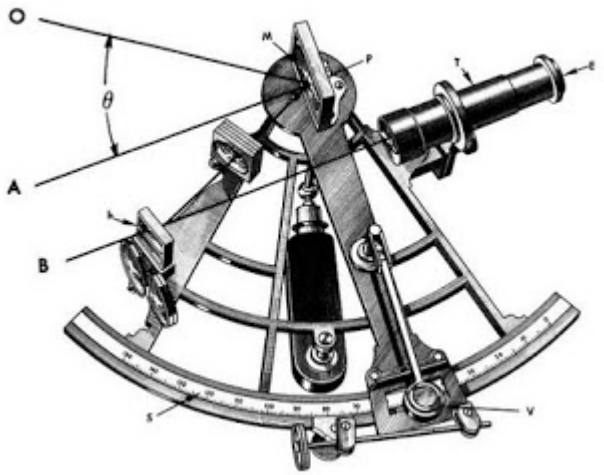
Isaac Slavitt

Random guy off the street

https://github.com/isms/boston-python-talk

Roadmap

- Campaign overview
- Sea stories
- Crash course in deploying apps to Heroku



The Campaign

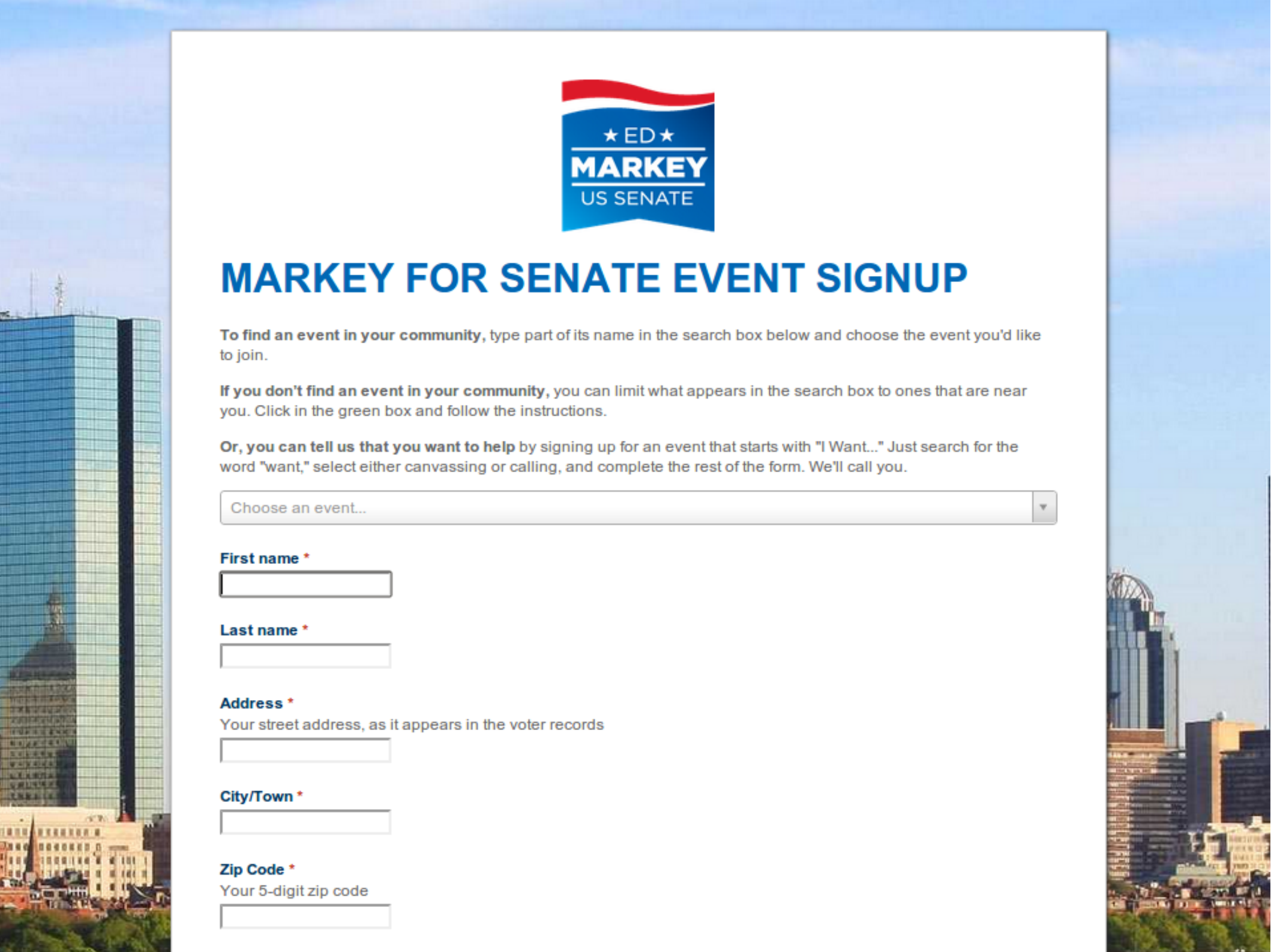
- Special election: short on time, not much $ for overhead
- Plenty of people who want to help, thousands of events
- Election SaaS called VoteBuilder, powerful but proprietary, no API access

Case Study I: Sign-up Page

- Campaign had WP-based website
- Event sign-ups on the official site “coming soon”
- Stopgap measure is plain old Google Form
- Manual multiple choice entry, no branding
- **Want easy way for people to discover any event near them, and sign up**

Case Study I: Sign-up Page

- Luckily can POST from arbitrary site to Google Form
- Can export event data from VoteBuilder
- Initial solution:
  - static sign-up page hosted on S3 – on load, use jQuery to populate multiple choice select with event data from JSON file
  - py script to parse exported events to a JSON file
  - s3cmd command line tool to upload JSON to S3



MARKEY FOR SENATE EVENT SIGNUP

To find an event in your community, type part of its name in the search box below and choose the event you'd like to join.

If you don't find an event in your community, you can limit what appears in the search box to ones that are near you. Click in the green box and follow the instructions.

Or, you can tell us that you want to help by signing up for an event that starts with "I Want..." Just search for the word "want," select either canvassing or calling, and complete the rest of the form. We'll call you.

Choose an event...

First name *

Last name *

Address *
Your street address, as it appears in the voter records

City/Town *

Zip Code *
Your 5-digit zip code

MARKEY FOR SENATE EVENT SIGNUP

To find an event in your community, type part of its name in the search box below and choose the event you'd like to join.

If you don't find an event in your community, you can limit what appears in the search box to ones that are near you. Click in the green box and follow the instructions.

Or, you can tell us that you want to help by signing up for an event that starts with "I Want..." Just search for the word "want," select either canvassing or calling, and complete the rest of the form. We'll call you.

Choose an event...

Malden

- 03/09/13 - Malden Canvass Kick off - 10:00 AM - 1:00 PM - Canvassing
- 03/10/13 - Malden Canvass Stop & Shop 99 Charles Street - 12:00 PM - 3:00 PM - Canvassing
- 03/13/13 - Malden Phonebank: Home of Iodiah Henry 38 Woodland RD - 6:30 PM - 8:30 PM - Phone Bank
- 03/16/13 - Malden Democratic City Committee Annual Breakfast -- Visibility - 8:30 AM - 9:15 AM - Visibility
- 03/16/13 - Malden Canvass Stop & Shop 99 Charles Street - 1:00 PM - 4:00 PM - Canvassing
- 03/17/13 - Malden Canvass Meeting at the Stop & Shop - 12:00 PM - 3:00 PM - Canvassing
- 03/20/13 - Malden Phonebank: Home of Joy Pearson 193 Sylvan St. - 6:30 PM - 8:30 PM - Phone Bank
- 03/23/13 - Malden Canvass Stop & Shop 99 Charles Street - 10:00 AM - 1:00 PM - Canvassing

Zip Code *

Your 5-digit zip code

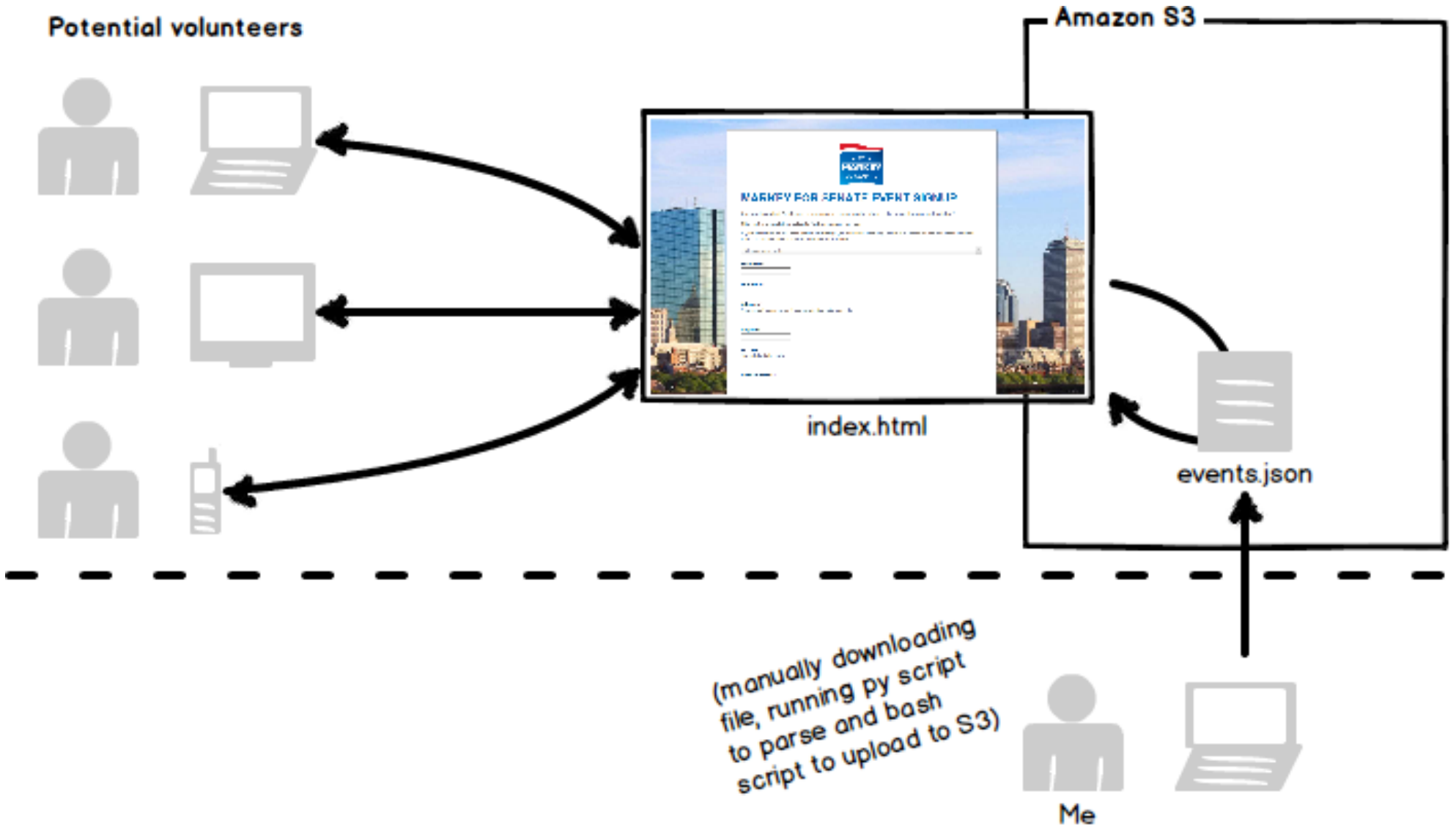
Sign-up Page

- Luckily can POST from
- Can export event data from VoteBuil
- Initial solution:
  - sults to a JSON file
  - and line t S3
  - static sign-up page hos
  - on load, use jQuery to ect with event data from

The jQuery Boston Meetup Group

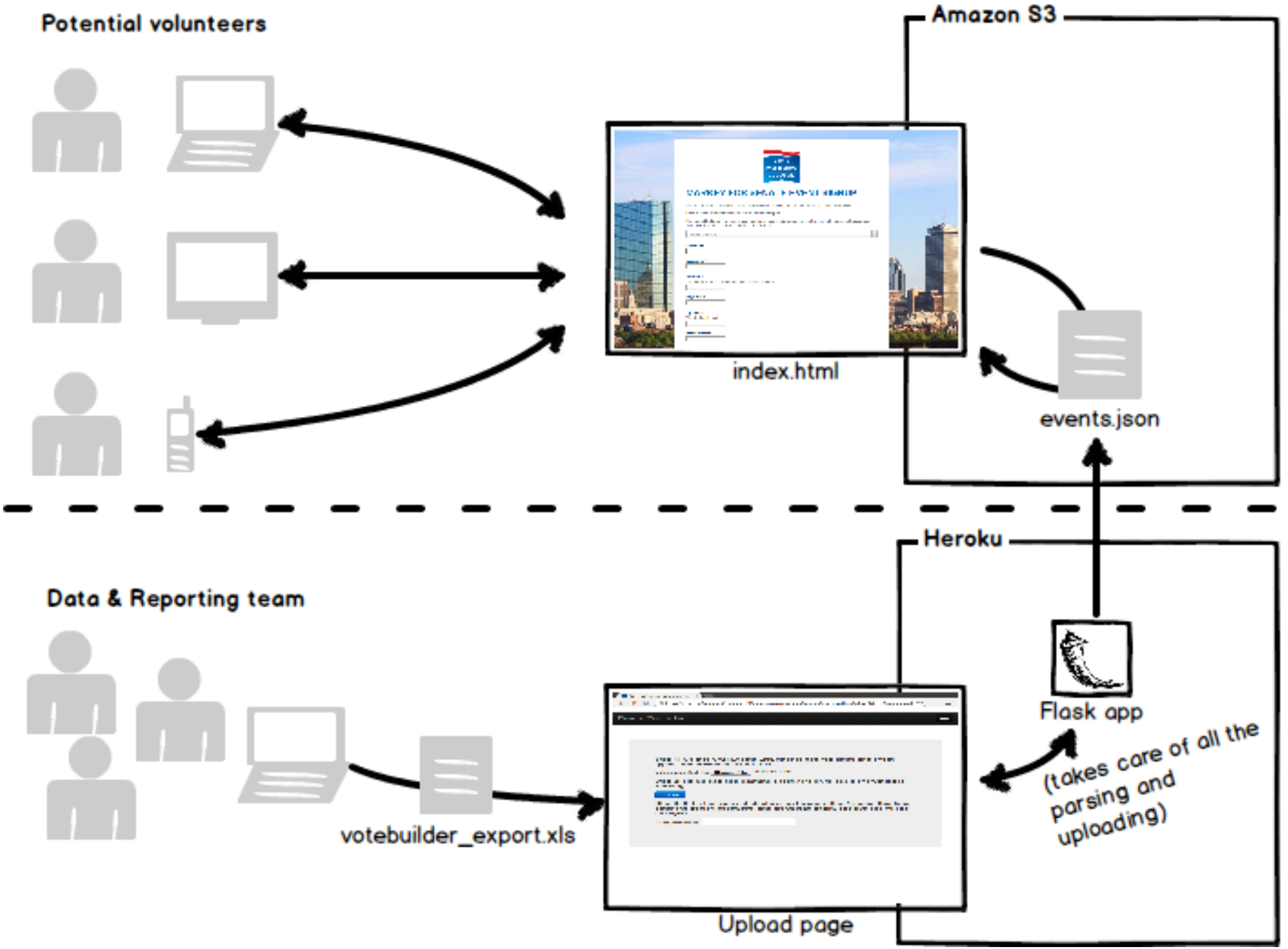
The Boston JavaScript Meetup Group

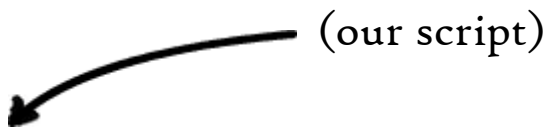
Boston Front End DEVELOPERS


Potential volunteers
Amazon S3
index.html
events.json
(manually downloading file, running py script to parse and bash script to upload to S3)
Me

Case Study I: Sign-up Page

- Updates way too cumbersome for one person
- Use volunteers?
- Would have to teach volunteers: “run python script ... oh, first install Python ... here's how to open a terminal ... now log into AWS with all our production sites ...”?
- There is a better way to do this


Potential volunteers
Amazon S3
index.html
events.json
Heroku
Data & Reporting team
Flask app
(takes care of all the parsing and uploading)
votebuilder_export.xls
Upload page




```
from event_parser import event_file_to_json
from werkzeug.utils import secure_filename

...

@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        f = request.files['file']
        filename = secure_filename(f.filename)
        folder = app.config['UPLOAD_FOLDER']
        path = os.path.join(folder, filename)
        f.save(path)

        json = event_file_to_json(f)
        ...
```

User can upload exported events file to Flask app ...

```
from boto.s3.key import Key
from boto.s3.connection import S3Connection

AWS_ACCESS_KEY = os.environ['AWS_ACCESS_KEY']
AWS_SECRET_KEY = os.environ['AWS_SECRET_KEY']

def upload(bucket_name, key, content):
    conn = S3Connection(AWS_ACCESS_KEY, AWS_SECRET_KEY)
    bucket = conn.get_bucket(bucket_name)
    k = Key(bucket)
    k.key = key  # in S3, key = path + filename
    k.set_metadata('Content-Type', 'application/json')
    k.set_contents_from_string(content)
```

... and send brand new, updated JSON file to S3

Step 1: Go into VAN Events List, filter to desired dates and event types, and download export file.

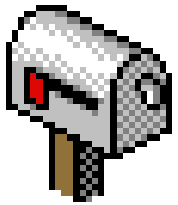
Select exported file: Choose File No file chosen

Step 2: Hit Upload and examine a test version to see if everything is working.

Upload

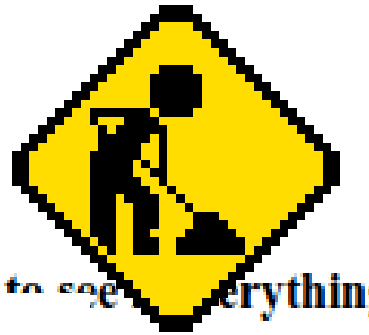
Step 3: If the test was satisfactory, go through Step 1 again, this time enter the deploy password, and hit Upload again. The live site will be changed.

Enter password:

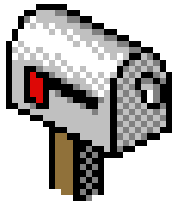


› VAN Event desired dates and event types, and
file.

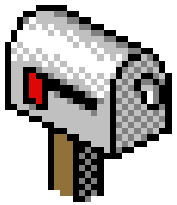
bose File No file chosen



Step 2: Hit Upload and examine a test version to see if everything is working.

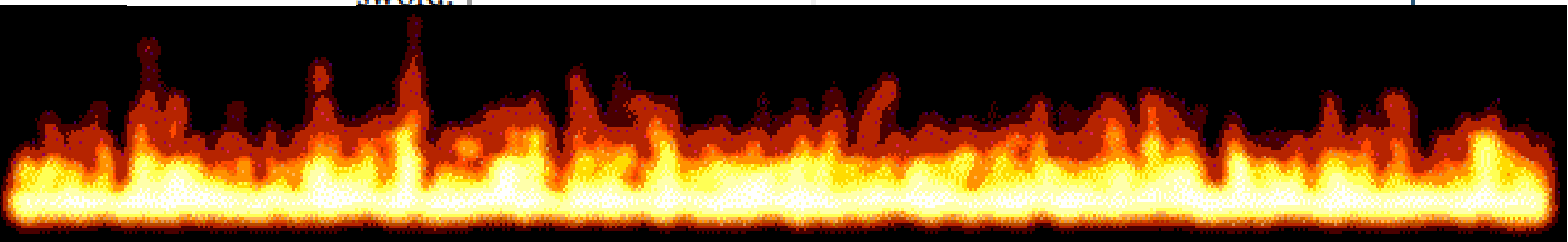


Upload



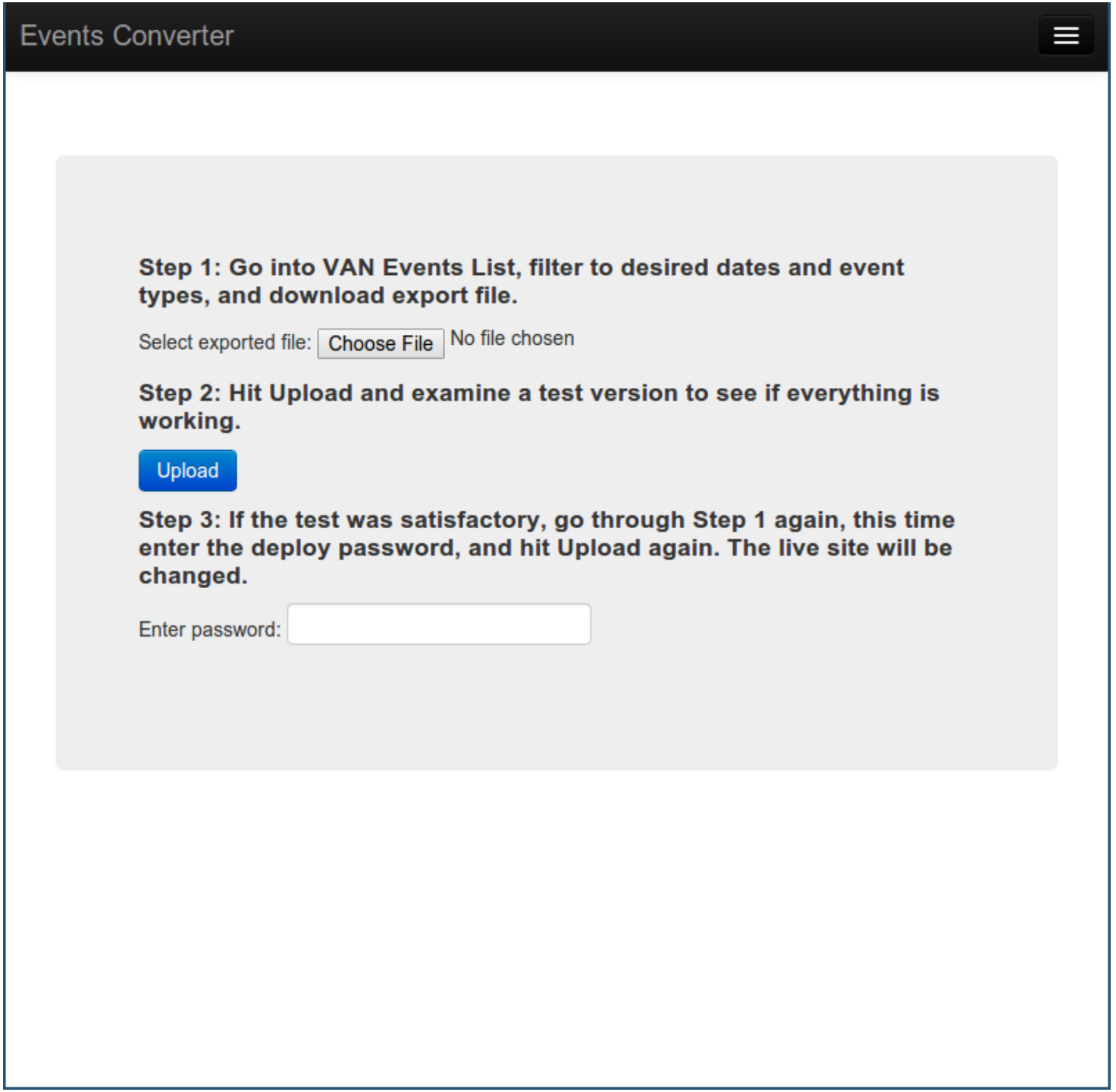
f the te[illegible]sfactory, go through Step 1 again, [illegible] enter
y passwo[illegible]nd hit Upload again. The live site will b[illegible]anged.

sword:



Twitter Bootstrap (getbootstrap.com) via BootstrapCDN (bootstrapcdn.com)

```
<link
href="//netdna.bootstrapcdn.com/twitter-bootstrap/2.3.1/
css/bootstrap-combined.min.css" rel="stylesheet">

<script
src="//netdna.bootstrapcdn.com/twitter-bootstrap/2.3.1/
js/bootstrap.min.js"></script>
```


Events Converter
Step 1: Go into VAN Events List, filter to desired dates and event types, and download export file.
Select exported file:
Choose File
No file chosen
Step 2: Hit Upload and examine a test version to see if everything is working.
Upload
Step 3: If the test was satisfactory, go through Step 1 again, this time enter the deploy password, and hit Upload again. The live site will be changed.
Enter password:

Join us this weekend!

Carl Nilsson info@edmarkey.com via bounce.bluestatedigital.com
to me

Apr 4

Dear Isaac

We need your help.

We have less than a month before election day, and we need you to help us make this our best weekend yet.

Last weekend, we blew away our record and knocked on 14,000 doors - *in just one day*. This weekend, we need to go even further - we need to hit 30,000 doors! That's a big number, we know, but we have to get there if we want Ed to win!

So, do you have a few hours to spare to help us knock on doors for Ed?

We're organizing more than 1,000 shifts across 240 canvasses throughout the state - **can you join our team?**

We've made amazing progress so far, but we can't stop now.

RSVP to knock on doors in your neighborhood, today.

Carl

Carl Nilsson
Field Director
The Markey Committee

Paid for by The Markey Committee

Case Study II: Event Listings

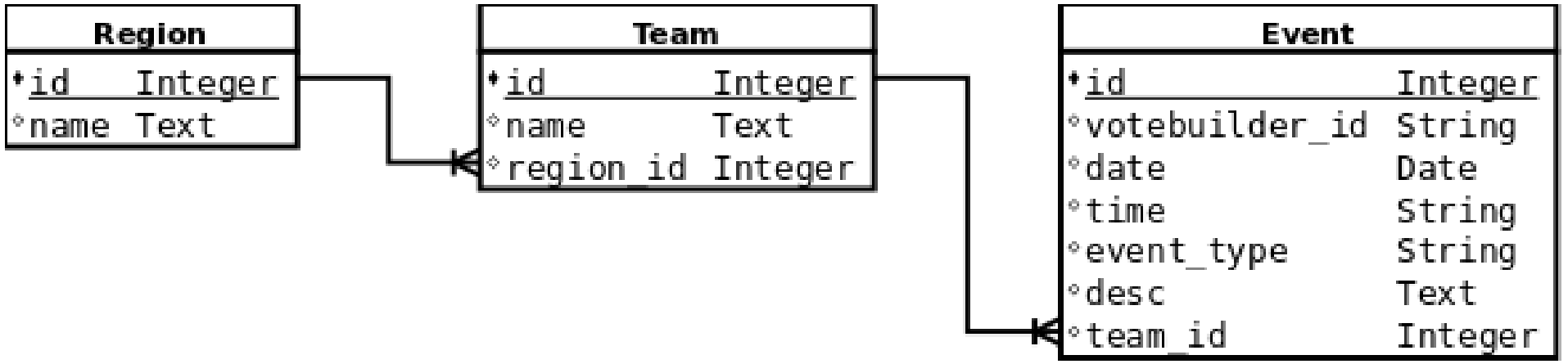
- Staff needed to look up all Event IDs for their region or team **all the time** for other systems & reporting
- Campaign sliced state up into regions, each region further broken down into teams,
- Busy team might have 30+ events per day, multiple steps to look up event in VoteBuilder
- Regions and teams are a fluid construct for organizing campaign staff – even if we had direct access to VoteBuilder data, these don't in exist for filtering

Case Study II: Event Listings

- Data & Reporting team maintains master spreadsheet with all event data exported from VoteBuilder
- Columns for region and team
- So we can just use this spreadsheet to populate a DB
  - Models with SQLAlchemy
  - Simple templates to displays everything

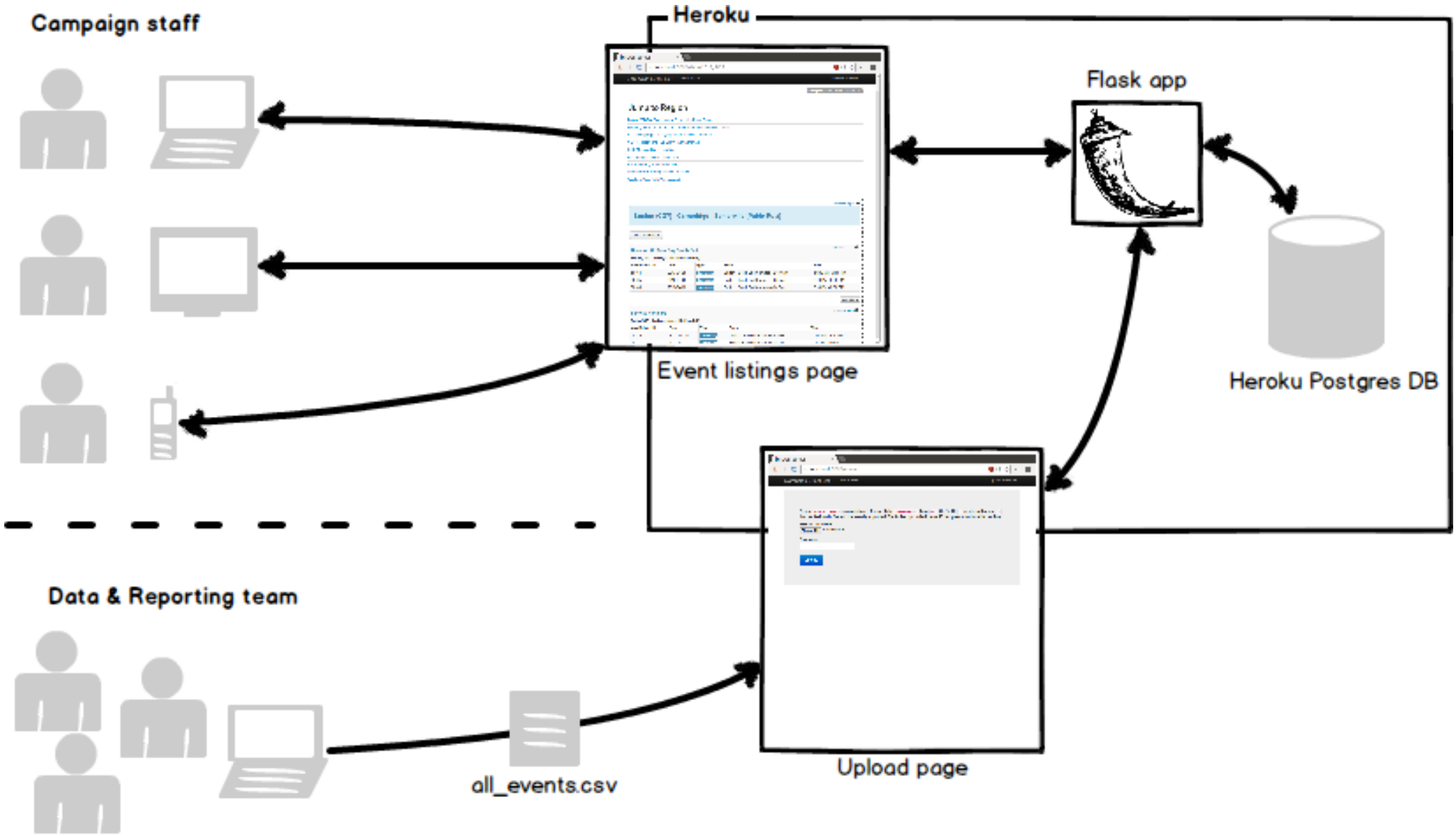
Flask SQLAlchemy

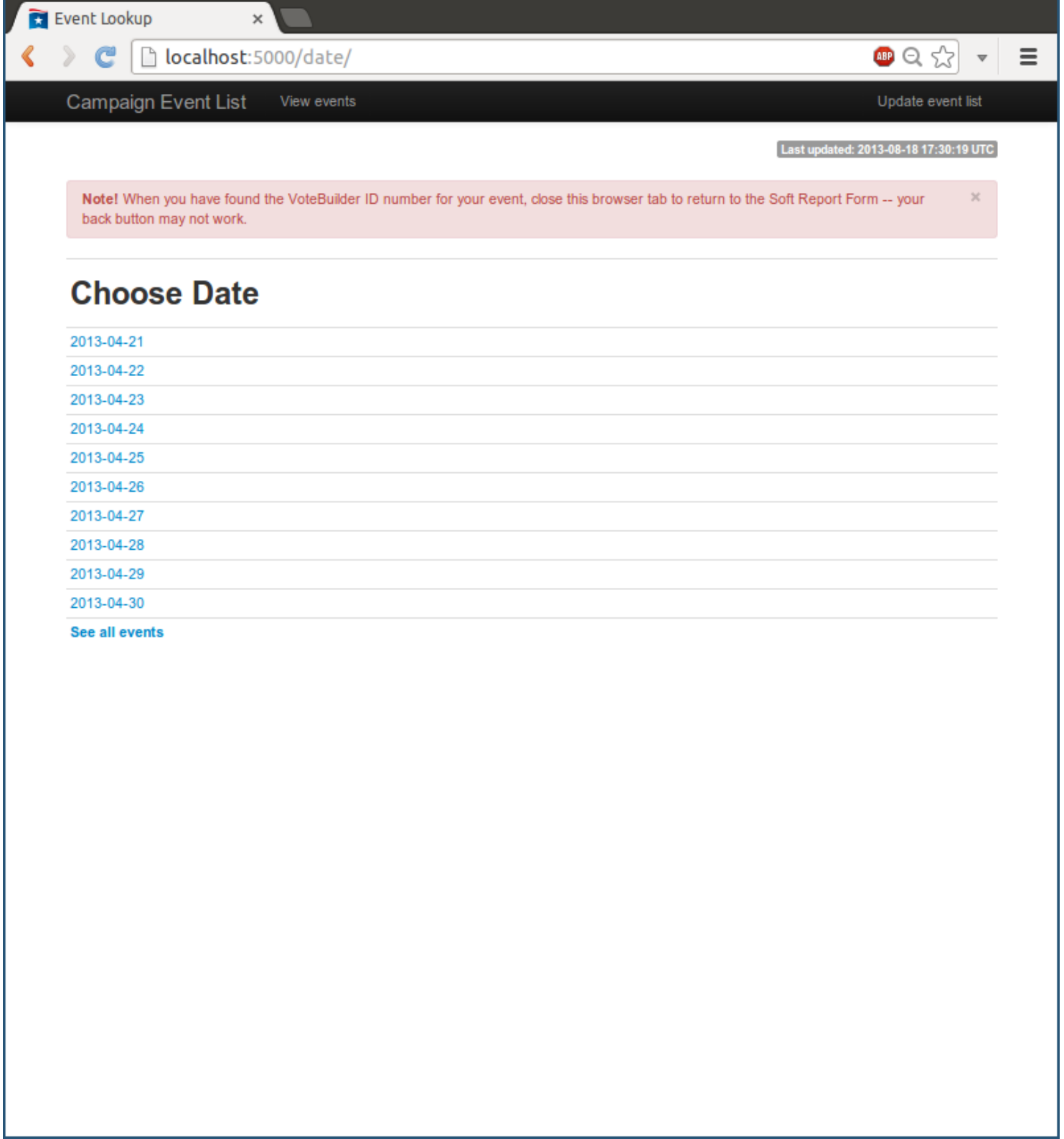
Case Study II: Event Listings

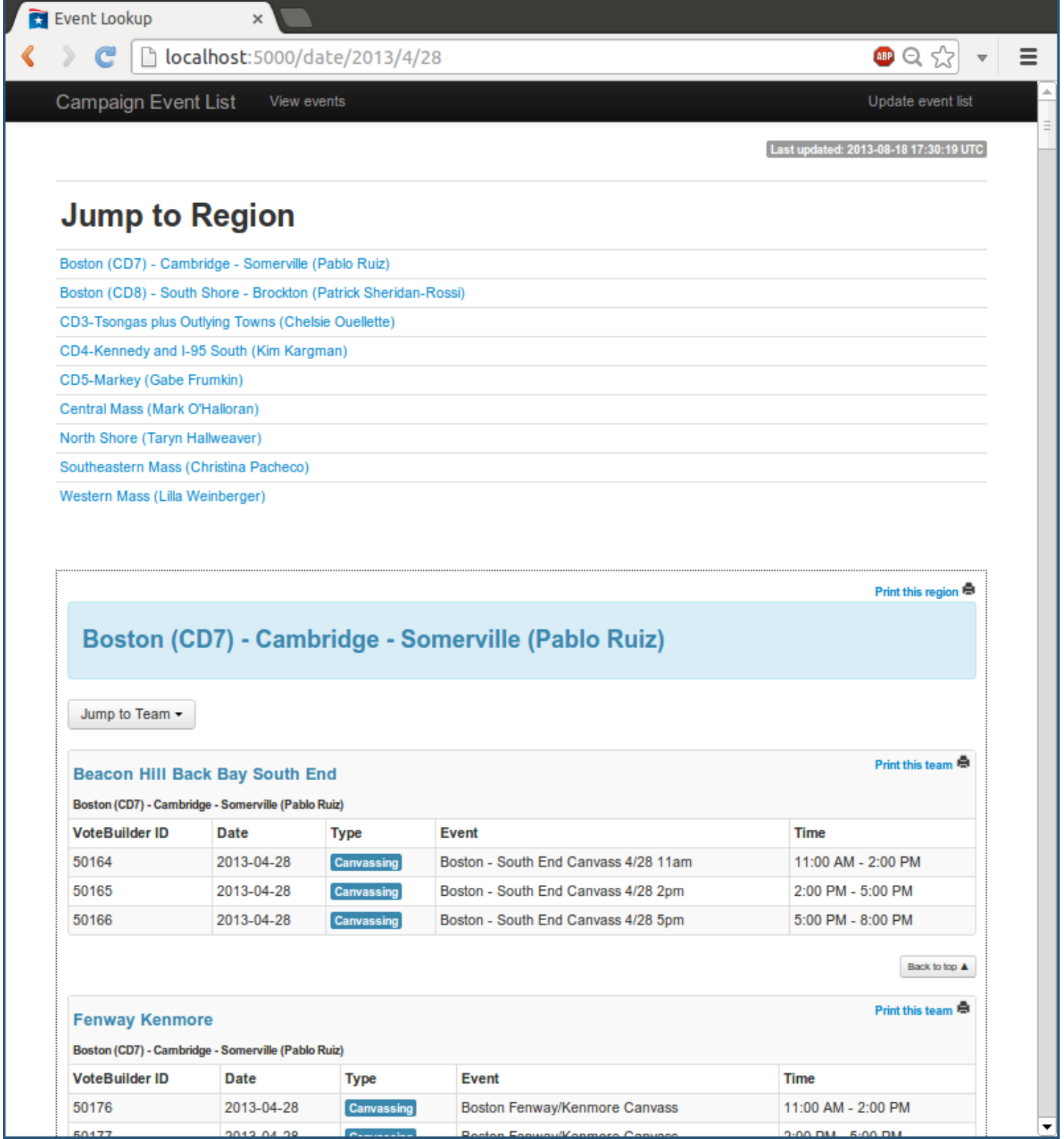


models.py

```
class Region(Model):
    __tablename__ = 'region'
    id = Column(Integer, primary_key=True)
    name = Column(Text())
    teams = relationship('Team', backref=backref('region'))
    ...

class Team(Model):
    __tablename__ = 'team'
    id = Column(Integer, primary_key=True)
    name = Column(Text())
    region_id = Column(Integer, ForeignKey('region.id'))
    events = relationship('Event', backref=backref('team'))
    ...

class Event(Model):
    __tablename__ = 'event'
    id = Column(Integer, primary_key=True)
    votebuilder_id = Column(String(20))
    date = Column(Date())
    time = Column(String(40))
    event_type = Column(String(60))
    desc = Column(Text())
    team_id = Column(Integer, ForeignKey('team.id'))
    ...
```


Campaign staff
Heroku
Flask app
Event listings page
Heroku Postgres DB
Data & Reporting team
all_events.csv
Upload page

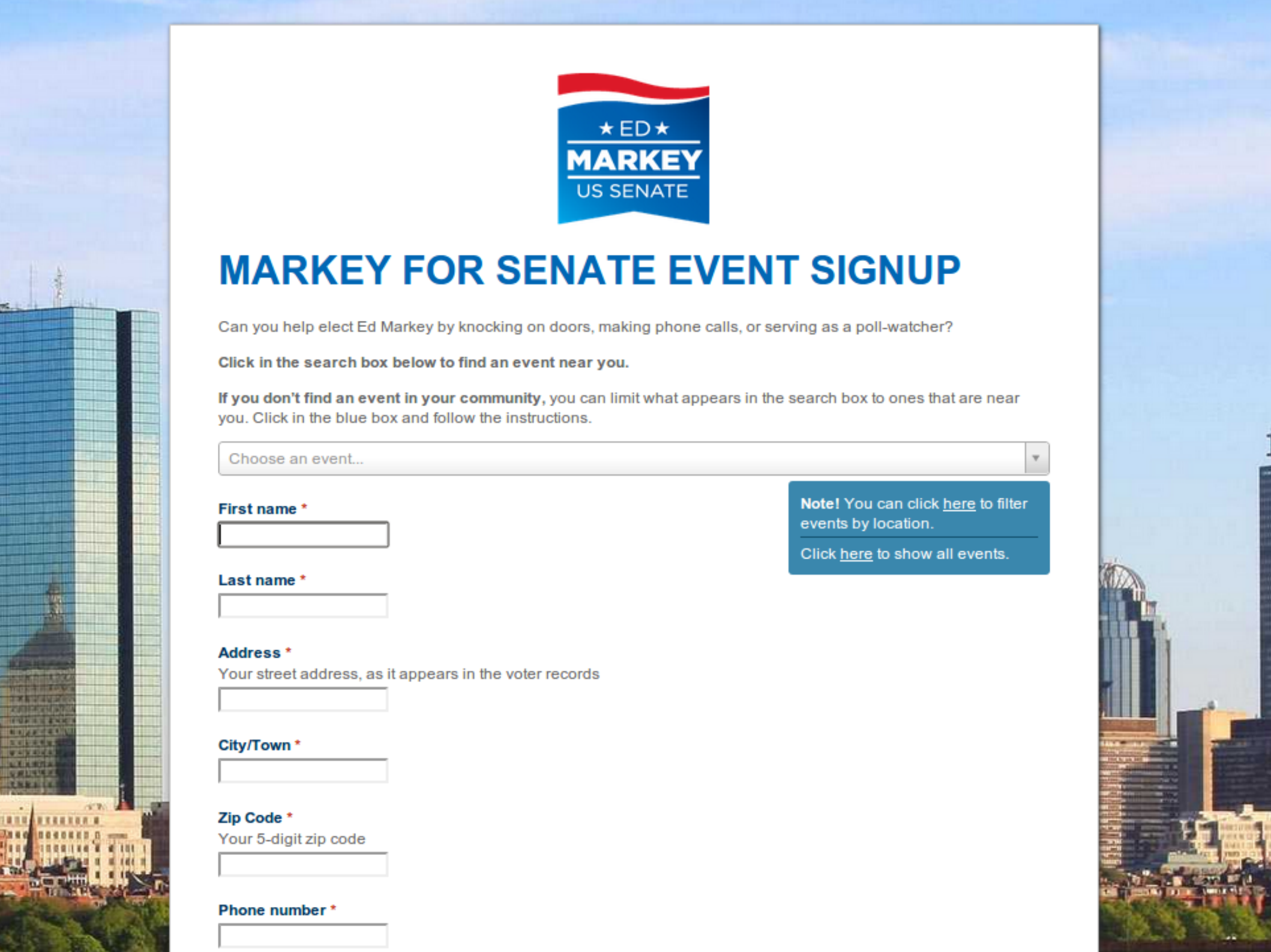

Event Lookup
localhost:5000/date/
Campaign Event List
View events
Update event list
Last updated: 2013-08-18 17:30:19 UTC
Note! When you have found the VoteBuilder ID number for your event, close this browser tab to return to the Soft Report Form -- your back button may not work.
Choose Date
2013-04-21
2013-04-22
2013-04-23
2013-04-24
2013-04-25
2013-04-26
2013-04-27
2013-04-28
2013-04-29
2013-04-30
See all events


Event Lookup
localhost:5000/date/2013/4/28
Campaign Event List
View events
Update event list
Last updated: 2013-08-18 17:30:19 UTC
Jump to Region
Boston (CD7) - Cambridge - Somerville (Pablo Ruiz)
Boston (CD8) - South Shore - Brockton (Patrick Sheridan-Rossi)
CD3-Tsongas plus Outlying Towns (Chelsie Ouellette)
CD4-Kennedy and I-95 South (Kim Kargman)
CD5-Markey (Gabe Frumkin)
Central Mass (Mark O'Halloran)
North Shore (Taryn Hallweaver)
Southeastern Mass (Christina Pacheco)
Western Mass (Lilla Weinberger)
Print this region
Boston (CD7) - Cambridge - Somerville (Pablo Ruiz)
Jump to Team
Beacon Hill Back Bay South End
Print this team
Boston (CD7) - Cambridge - Somerville (Pablo Ruiz)
VoteBuilder ID | Date | Type | Event | Time
50164 | 2013-04-28 | Canvassing | Boston - South End Canvass 4/28 11am | 11:00 AM - 2:00 PM
50165 | 2013-04-28 | Canvassing | Boston - South End Canvass 4/28 2pm | 2:00 PM - 5:00 PM
50166 | 2013-04-28 | Canvassing | Boston - South End Canvass 4/28 5pm | 5:00 PM - 8:00 PM
Back to top
Fenway Kenmore
Print this team
Boston (CD7) - Cambridge - Somerville (Pablo Ruiz)
VoteBuilder ID | Date | Type | Event | Time
50176 | 2013-04-28 | Canvassing | Boston Fenway/Kenmore Canvass | 11:00 AM - 2:00 PM

Case Study II: Event Listings

 **BONUS** 

- Now we have a team associated with every event in a Heroku Postgres DB
- Teams correspond to fine-grained geographical area
- Now when our **old** event updater app updates the JSON file, it can get each event's team from **new** DB
- A little more JS on the static sign-up page (modal UI, associate teams ↔ zip codes), and we can let the user filter by events within *x* miles of their zip



MARKEY FOR SENATE EVENT SIGNUP

Can you help elect Ed Markey by knocking on doors, making phone calls, or serving as a poll-watcher?

Click in the search box below to find an event near you.

If you don't find an event in your community, you can limit what appears in the search box to ones that are near you. Click in the blue box and follow the instructions.

Choose an event...

Note! You can click here to filter events by location.

Click here to show all events.

First name *

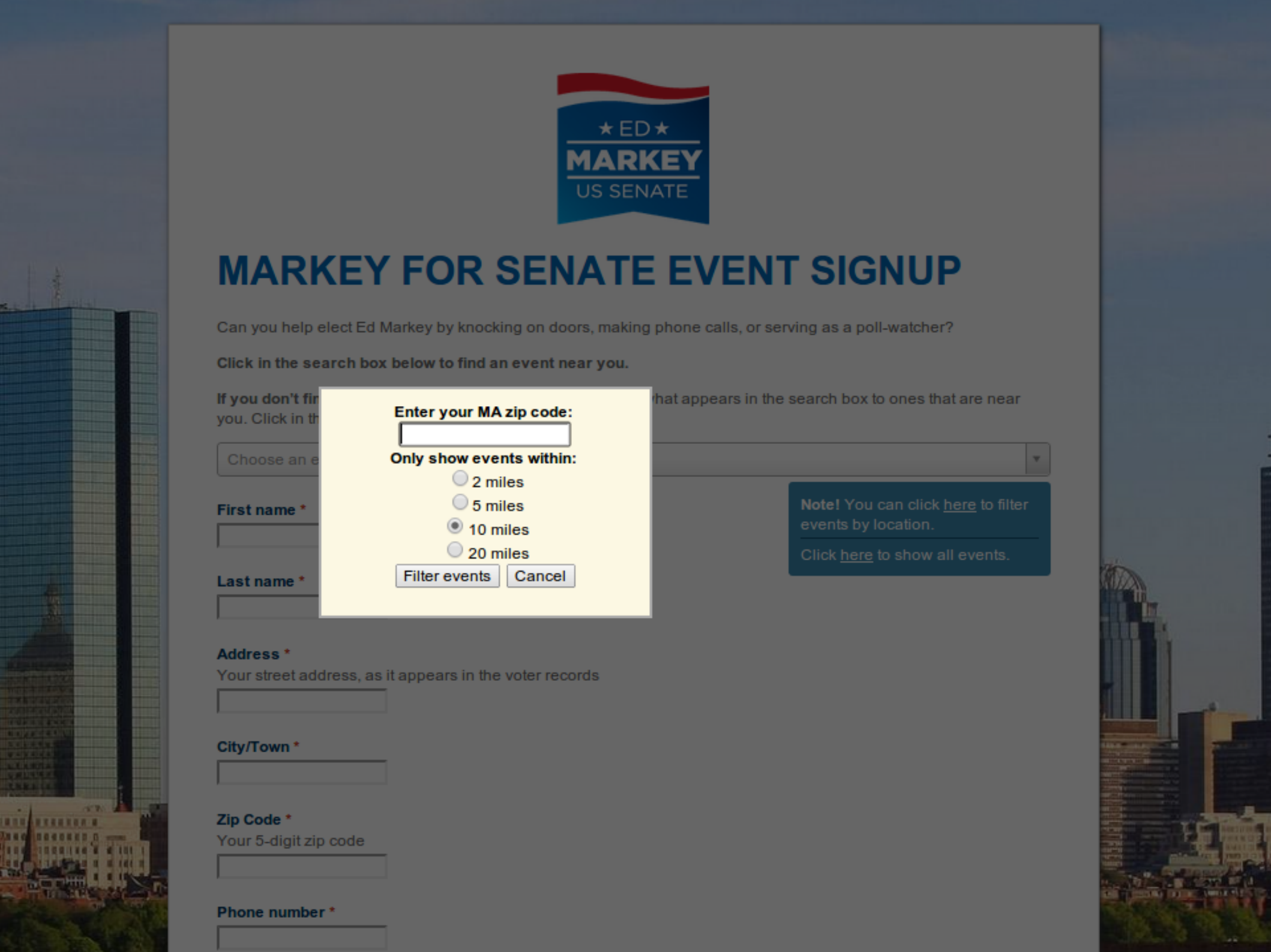
Last name *

Address *
Your street address, as it appears in the voter records

City/Town *

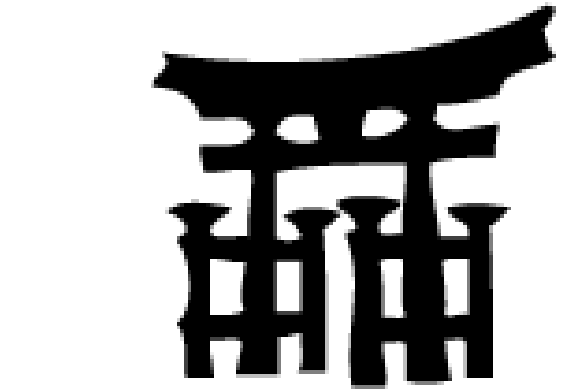
Zip Code *
Your 5-digit zip code

Phone number *


★ED★
MARKEY
US SENATE
MARKEY FOR SENATE EVENT SIGNUP
Can you help elect Ed Markey by knocking on doors, making phone calls, or serving as a poll-watcher?
Click in the search box below to find an event near you.
If you don't fir
hat appears in the search box to ones that are near
you. Click in th
Choose an e
Enter your MA zip code:
Only show events within:
2 miles
5 miles
10 miles
20 miles
Filter events
Cancel
Note! You can click here to filter events by location.
Click here to show all events.
First name *
Last name *
Address *
Your street address, as it appears in the voter records
City/Town *
Zip Code *
Your 5-digit zip code
Phone number *

Case Study III: Single Sign-up Pages

- Started getting requests from organizers for special one-off sign-up pages for **one specific event**
- Want same branding/style as main event list, but no event selection for potential volunteer
- Staff could shorten URL & blast out in e-mail to **targeted** groups
- First couple: “sure no problem” – but doesn't scale, time consuming and error prone


Jinja

```
...

<p>
  <h1>
    Lowell Rally with Ed Markey and Gov. Michael Dukakis
  </h1>
<p>
  <h2>
    Congressman Ed Markey<br>
    Democrat for United States Senate
  </h2>
<p>
  <strong>Sunday, June 23</strong><br>
  12:00 pm
</p>

...
```

We can take the single event format we've been using ...

```
...

<p>
  <h1>
    {{ event.title }}
  </h1>
<p>
  <h2>
    Congressman Ed Markey<br>
    Democrat for United States Senate
  </h2>
<p>
  <strong>{{ event.day }}</strong><br>
  {{ event.time }}
</p>

...
```

... abstract once, render ∞.

forms.py

```
from wtforms import Form, validators
from flask.ext.wtf import TextField, TextAreaField, PasswordField, BooleanField

class PageForm(Form):

    title = TextField('Event title (e.g. Lowell GOTV Rally)',
                      [validators.Required()])

    date = TextField('Date (e.g. June 24, 2013)')

    time = TextField('Time (e.g. 2:30 pm)')

    address = TextAreaField('Address')

    description = TextAreaField('Description')

    entry = TextAreaField('Event info to be submitted to the Google Doc')

    filename = TextField('Short name for [...]',
                         [validators.Length(min=6, max=35)])

    production = BooleanField('I [...] am now ready to upload as final')

    password = PasswordField('Password')
```

Enter event information

Event title (e.g. Lowell GOTV Rally)

Description

Date (e.g. June 24, 2013)

Time (e.g. 2:30 pm)

Address

Event info to be submitted to the Google Doc

Short name for this signup page, to be used for thefilename (e.g. lowell-rally)

Password

Create and upload page

I have previewed by submitting once without checking this box, and am now ready to upload it as final (if left unchecked, the page will be uploaded to the staging bucket for preview)

Enter event information

Event title (e.g. Lowell GOTV Rally)

- This field is required.

Description

Date (e.g. June 24, 2013)

Time (e.g. 2:30 pm)

Address

Event info to be submitted to the Google Doc

Short name for this signup page, to be used for thefilename (e.g. lowell-rally)

blah

- Field must be between 6 and 35 characters long.

Password

app.py

```
@app.route('/', methods=['GET', 'POST'])
def create_page():

    # get user input
    form = forms.PageForm(request.form)

    ...

    # render the signup page
    event = models.Event(form.title.data, form.description.data,
                         form.date.data, form.time.data,
                         form.address.data, form.entry.data)
    page_to_upload = render_template('signup_page.html', event=event)

    # upload to S3
    bucket = (settings.PRODUCTION_BUCKET if form.production.data
              else settings.STAGING_BUCKET)
    key = clean_key(form.filename.data)
    upload(bucket, key, page_to_upload)

    ...
```

You're invited!

TEST EVENT TITLE

Congressman Ed Markey
Democrat for United States Senate

August 30, 2013
7:30 pm

Event Conference Center
1234 Sample Address
Cambridge, MA 02138

This is a sample signup form created by the form creator app.
Line breaks work in this field because of the nl2br filter we created for Jinja template rendering.

I am interested in attending; please sign me up.

First name *

Case Study III: Single Sign-up Pages

Entire app is in the Github repo for this talk:

https://github.com/isms/boston-python-talk/signup-form-creator

(Certain things have been simplified or expanded to make for a better example – especially the security and staging/production stuff – but it's basically what we used.)

Working with Heroku

- We used it for **everything**
- Very easy to set up
- Great free tier (can have one dyno, one Postgres DB with up to 10,000 rows), no time limit, easy to scale
- Excellent tutorial on Heroku site: https://devcenter.heroku.com/articles/python
- We'll deploy a sample app

app.py

```
from flask import Flask, render_template_string

app = Flask(__name__)

TEMPLATE = '<html><body><h1>Hello {{ person }}</h1></body></html>'

@app.route('/')
def hello():
    return render_template_string(TEMPLATE, person="World")

if __name__=="__main__":
    app.run('0.0.0.0', port=5000, debug=True)
```

requirements.txt

```
Flask
gunicorn
```

Procfile

```
web: gunicorn app:app
```

First steps

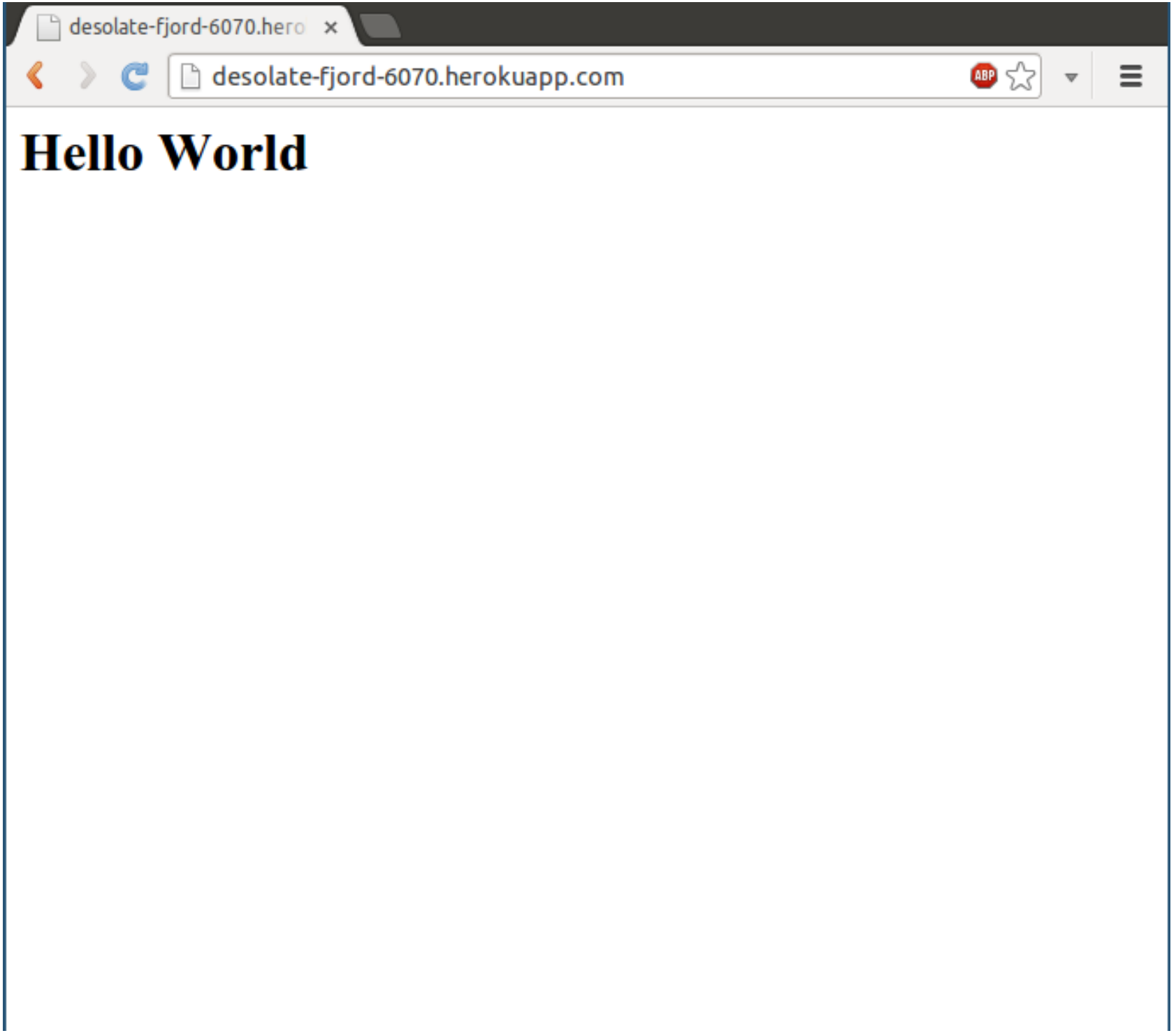
```
$ ls
app.py  Procfile  requirements.txt

$ git init
Initialized empty Git repository [...]

$ git add .
$ git commit -m 'initial commit'
[master (root-commit) 9185871] initial commit
 3 files changed, 14 insertions(+)
 create mode 100644 Procfile
 create mode 100644 app.py
 create mode 100644 requirements.txt
```

Deploying

```
$ heroku create
Creating desolate-fjord-6070... done, stack is cedar
http://desolate-fjord-6070.herokuapp.com/ |
git@heroku.com:desolate-fjord-6070.git
Git remote heroku added

$ git push heroku master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 517 bytes, done.
Total 5 (delta 0), reused 0 (delta 0)

-----> Python app detected
-----> No runtime.txt provided; assuming python-2.7.4.
-----> Preparing Python runtime (python-2.7.4)

[...]
```

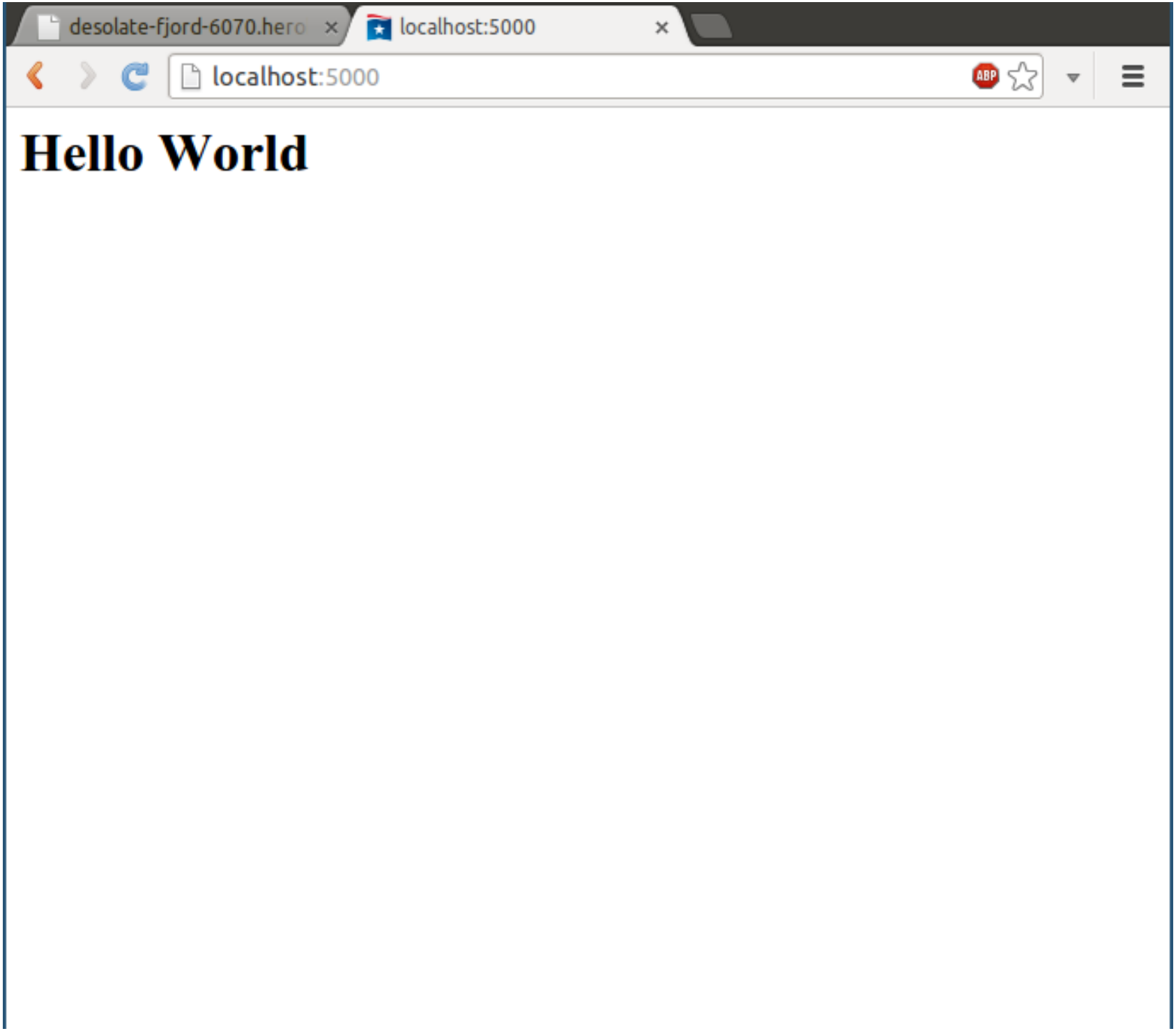
Deploying

```
-----> Installing Pip (1.3.1)
-----> Installing dependencies using Pip (1.3.1)
       Downloading/unpacking Flask (from -r requirements.txt
(line 1))
       Successfully installed Flask Werkzeug Jinja2 itsdangerous
markupsafe
       Cleaning up...
-----> Discovering process types
       Procfile declares types -> web

-----> Compiled slug size: 28.4MB
-----> Launching... done, v1
       http://desolate-fjord-6070.herokuapp.com deployed to
Heroku

To git@heroku.com:desolate-fjord-6070.git
 * [new branch]      master -> master
```


desolate-fjord-6070.hero
desolate-fjord-6070.herokuapp.com
Hello World

Running locally

```
$ foreman start
22:39:54 web.1  | started with pid 10528
22:39:54 web.1  | 2013-08-22 22:39:54 [10531] [INFO] Starting
gunicorn 0.17.4
22:39:54 web.1  | 2013-08-22 22:39:54 [10531] [INFO] Listening
at: http://0.0.0.0:5000 (10531)
22:39:54 web.1  | 2013-08-22 22:39:54 [10531] [INFO] Using
worker: sync
22:39:54 web.1  | 2013-08-22 22:39:54 [10536] [INFO] Booting
worker with pid: 10536
```


desolate-fjord-6070.hero
localhost:5000
localhost:5000
Hello World

Running locally

```
$ echo "web: python app.py" > Procfile.dev

$ foreman start -f Procfile.dev
22:45:36 web.1  | started with pid 10615
22:45:36 web.1  |  * Running on http://0.0.0.0:5000/
22:45:36 web.1  |  * Restarting with reloader
```

```
if __name__=="__main__":
    app.run('0.0.0.0', port=5000, debug=True)
```

Configuration variables

- Locally stored in `.env` – Foreman uses this file
- Can set remotely with `heroku config:set key=value`

```
# LOCAL

$ echo "API_KEY=f572d396fae92066287" > .env
$ cat .env
API_KEY=f572d396fae92066287


# REMOTE

$ heroku config:set API_KEY=f572d396fae92066287
Setting config vars and restarting desolate-fjord-6070... done,
v2
API_KEY: f572d396fae92066287
```

Configuration variables

- Why is this important?
  - Separation of concerns between configuration and business logic
  - Databases: can set a connection string locally for testing DB
  - Secrets: can add `.env` to `.gitignore` and keep it out of version control!

Adding a database (Heroku)

```
$ heroku addons:add heroku-postgresql:dev
Adding heroku-postgresql:dev on desolate-fjord-6070... done, v12
(free)
Attached as HEROKU_POSTGRESQL_RED_URL
Database has been created and is available

$ heroku pg:promote HEROKU_POSTGRESQL_RED_URL
Promoting HEROKU_POSTGRESQL_RED_URL to DATABASE_URL... done

$ heroku config
=== desolate-fjord-6070 Config Vars
DATABASE_URL:
postgres://jbyptmgjqxulf:6n71ODQ7T1GSEckqSMPK8sno@ec2-54-221-236-
4.compute-1.amazonaws.com:5432/d8kvjee12tob1p
HEROKU_POSTGRESQL_RED_URL:
postgres://jbyptmgjqxulf:6n71ODQ7T1GSEckqSMPK8sno@ec2-54-221-236-
4.compute-1.amazonaws.com:5432/d8kvjee12tob1p
```

Adding a database (local)

.env

```
DATBASE_URL=sqlite:///test.db
```

(Note: in real apps we should be using the same type of DB for testing and production.)

requirements.txt

```
Flask
Gunicorn
SQLAlchemy
Flask-SQLAlchemy
psycopg2
```

Adding a database

app.py

```
import os
from flask import Flask, render_template_string
from flask.ext.sqlalchemy import SQLAlchemy

TEMPLATE = '<html><body><h1>Hello {{ person }}</h1></body></html>'

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = os.environ['DATABASE_URL']
db = SQLAlchemy(app)

class Person(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(20))

@app.route('/')
def hello():
    people = Person.query.all()
    names = ', '.join([p.name for p in people])
    return render_template_string(TEMPLATE, person=names)
```

Run commands remotely

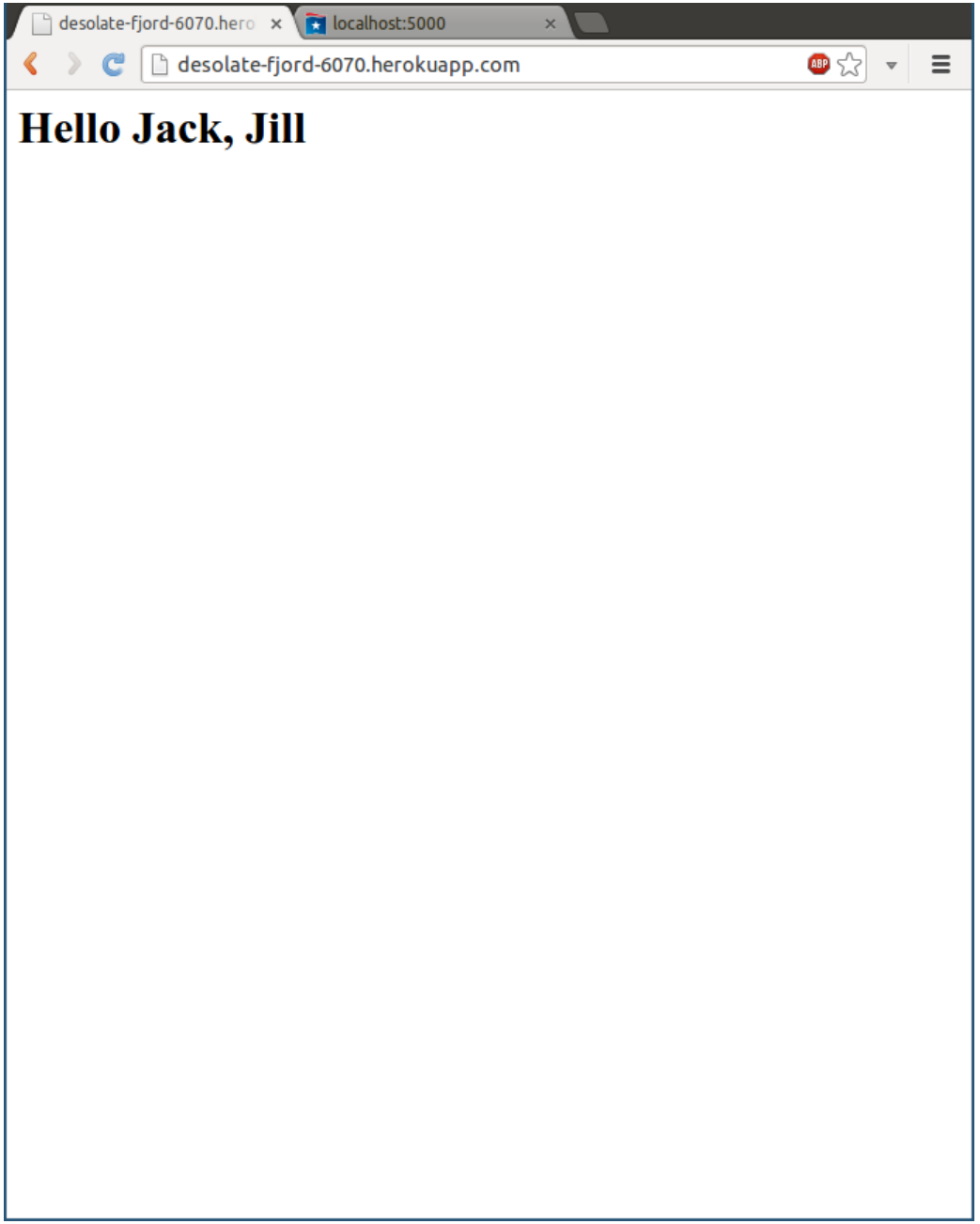
```
$ heroku run python
Running `python` attached to terminal... up, run.8017
Python 2.7.4 (default, Apr  6 2013, 22:14:13)
[GCC 4.4.3] on linux2

>>> from app import db, Person
>>> db.create_all()

>>> jack = Person(name='Jack')
>>> jill = Person(name='Jill')

>>> db.session.add(jack)
>>> db.session.add(jill)
>>> db.session.commit()
>>> Person.query.all()
[<app.Person object at 0x29b34d0>, <app.Person object at
0x29b3590>]
```

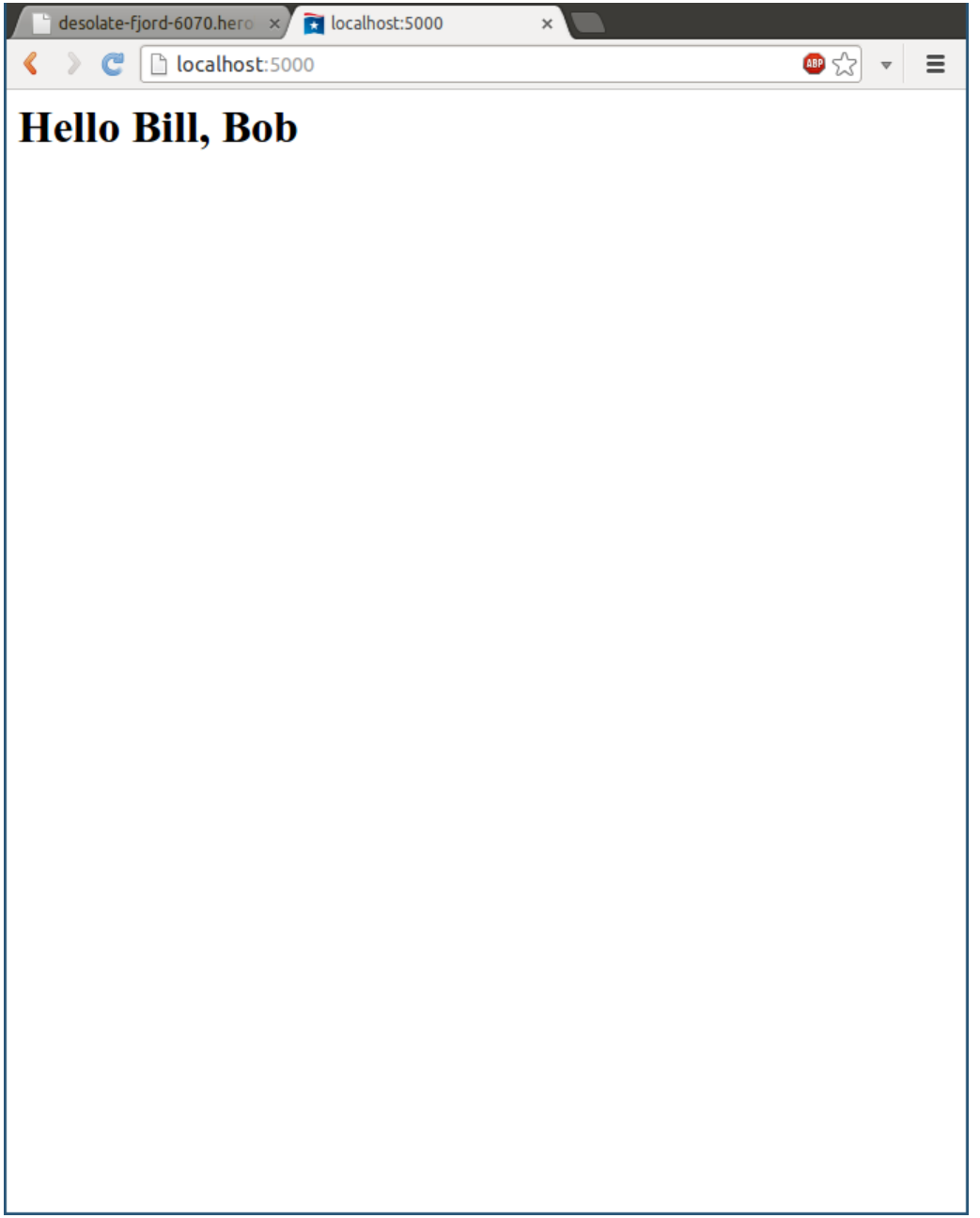
Run commands locally

```
$ foreman run python
Python 2.7.4 (default, Apr 19 2013, 18:28:01)
[GCC 4.7.3] on linux2

>>> from app import db, Person
>>> db.create_all()

>>> bob = Person(name='Bob')
>>> bill = Person(name='Bill')

>>> db.session.add(bob)
>>> db.session.add(bill)
>>> db.session.commit()
>>> Person.query.all()
[<app.Person object at 0x2979d50>, <app.Person object at
0x2979dd0>]
```

Interacting with Heroku DB

```
$ heroku pg:psql
psql (9.2.4)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

d8kvjee12tob1=> \dt
             List of relations
 Schema |  Name  | Type  |     Owner
--------+--------+-------+---------------
 public | person | table | jbyptmgjqxulf
(1 row)

d8kvjee12tob1=> SELECT * FROM person;
 id | name
----+------
  1 | Jack
  2 | Jill
(2 rows)
```


desolate-fjord-6070.hero
localhost:5000
desolate-fjord-6070.herokuapp.com
Hello Jack, Jill


desolate-fjord-6070.hero
localhost:5000
localhost:5000
Hello Bill, Bob

Wrap-up

- Flask is a Swiss army knife
- Free Heroku & free Amazon S3 will get you a surprisingly long way
- Using CSS framework makes internal apps look decent with little extra effort

Questions?