

DemocracyLab Contributor Guide

[Links](#)

[Collaboration](#)

[Code](#)

[Testing](#)

[Branches](#)

[Environments](#)

[Production \(Please do not create any test data here\)](#)

[Pre-Production \(Feel free to create test data here\)](#)

[Design](#)

[Environment Setup](#)

[Option 1: Using Linux VM](#)

[Get VM Image file](#)

[Set up VirtualBox](#)

[Set up VM](#)

[Verify VM is working](#)

[Option 2: Using Docker](#)

[Install Docker](#)

[Windows \(RECOMMENDED: Professional or other version supporting Hyper-V\)](#)

[Windows \(Home or other version not supporting Hyper-V\)](#)

[Mac](#)

[Ubuntu](#)

[Run Docker](#)

[View Page](#)

[Required: Environment Variables file](#)

[Code Changes](#)

[After Python changes](#)

[After JS Changes](#)

[Running Commands within Docker container](#)

[Option 3: Setting up Environment On Local Box](#)

[Disclaimer](#)

[Clone Git Repository](#)

[Install Python](#)

[Install Node.js](#)

[Ubuntu](#)

[Install Yarn package manager](#)

[Windows](#)

[Ubuntu](#)

[Install PostgreSQL](#)

[Mac](#)

[Ubuntu](#)

[Windows](#)

[Configure PostgreSQL database](#)

[Set up other Environment Variables](#)

[Run Development Server](#)

[S3 Setup](#)

[Option 1: Use Test S3 Bucket](#)

[Option 2: Set up your own S3 Bucket](#)

[Create Bucket](#)

[Create IAM User](#)

[Set Environment Variables](#)

[Running the Site](#)

[Setting up Database](#)

[Building Static Assets](#)

[Running Development Server](#)

[Load Test Data](#)

[Admin Console](#)

[Admin Credentials](#)

[VM](#)

[Test Data](#)

[Git Steps](#)

[Set User Information](#)

[Sync Latest Changes on Branch \(green text optional steps if you have changes in progress\)](#)

[Update Dependencies](#)

[Python](#)

[Node.js](#)

[Merging Changes to Master](#)

[Test Frameworks](#)

[JavaScript Testing Framework](#)

[Django Testing Tools \(docs\)](#)

[Troubleshooting](#)

[Cache table not found](#)

[Front-end changes not showing up](#)

[Development Process](#)

[Working on a Feature](#)

[1. Sync master branch to bring in latest changes](#)

[2. Create a new feature branch](#)

[3. Implement feature and ensure it is working locally](#)

- [4. Submit pull request](#)
- [5. Merge any changes from master \(if needed\)](#)
- [6. Merge feature into master](#)

[Style Guide](#)

[Structure](#)

[Third-party CSS](#)

[Best Practices](#)

[Technology Reference](#)

[Django](#)

[Overview](#)

[Models](#)

[Applying Database Changes](#)

[HTML Templates](#)

[Learning Django](#)

[Learning Python](#)

[React.js](#)

[Learning React](#)

[Flux](#)

[Flow](#)

[SASS](#)

[Prettier](#)

Links

Collaboration

Join Slack Channel:

https://join.slack.com/t/democracylab-org/shared_invite/enQtMjY3OTQ1NDI2NzU1LWYzYzNjZDQ5OTQzYzY1NjA1OTFmNjlxNzVhZjhhYjc0MDEzMzk2M2U4MTg0YjE1MmFhN2FkOTgxOTY1NzlwY2U

Slack Developers Channel: <https://democracylab-org.slack.com/messages/C7BC3AM0C>

Trello Board: [Development](#) and [Feature Development](#)

Google Calendar:

<https://calendar.google.com/calendar?cid=cTl2dnQ1bm9sY3VwcGJkZmltc2puNHJ1NWtAZ3JvdXAuY2FsZW5kYXluZ29vZ2xlLmNvbQ>

Code

GitHub: <https://github.com/DemocracyLab/CivicTechExchange>

Testing

Test Cases: https://1drv.ms/x/s!Aqv8yA0KSEeLmxGKNAhgl9zxr5_F

Branches

Working Branch: [master](#)

Environments

Production (Please do not create any test data here)

<https://www.democracylab.org/> (Please do not create any test data here)

Pre-Production (Feel free to create test data here)

Staging (Automatically pulls from master branch): <https://democracy-lab-staging.herokuapp.com>

Production Mirror: <https://democracy-lab-prod-mirror.herokuapp.com>

Production Mirror #2: <https://democracy-lab-dev.herokuapp.com>

Design

UI Mockups:

<https://www.figma.com/file/WADcmVjJh5ARVoZ09xlpfdFN/DemocracyLab?node-id=1876%3A1233>

Feature Acceptance Criteria:

<https://docs.google.com/document/d/1Sf6yUnvDZ7qN3RguX5CP1mIMPi7h9s67icrS-144oZE/edit?usp=sharing>

Environment Setup

Option 1: Using Linux VM

Get VM Image file

Download .ova image file from [here](#)

Set up VirtualBox

- Download Oracle VirtualBox for your platform [here](#)
- Install VirtualBox

Set up VM

- Open VirtualBox, click File -> Import Appliance
- Select .ova file and click 'Next'
- In the setup page, adjust the amount of CPU(s)/RAM based on your machine specs
- Check 'Reinitialize the MAC address of all network cards'

[I see a sector called MAC Address Policy with 3 choices,

1. Include only NAT network adapters MAC addresses,
2. Include all network adaptor MAC addresses,
3. Generate new MAC Addresses for all network adaptors

The default is 2 and I am trying that.]

- Click 'Import' Wait a bit for the import process to finish

Verify VM is working

- After the import finishes, you should see an entry in VirtualBox called 'DemocracyLab-Image'
- Right-click on entry and click 'Start'
- The VM should boot up to a login screen asking for the 'dev-admin' password.
- Enter Password: code4Gud!
- Open 'CONTRIBUTOR README' file and run the commands after 'Open Shell:' in the console

Option 2: Using Docker

Install Docker

Windows (RECOMMENDED: Professional or other version supporting Hyper-V)

<https://hub.docker.com/editions/community/docker-ce-desktop-windows>

Windows (Home or other version not supporting Hyper-V)

https://docs.docker.com/toolbox/toolbox_install_windows/

Mac

<https://hub.docker.com/editions/community/docker-ce-desktop-mac>

Ubuntu

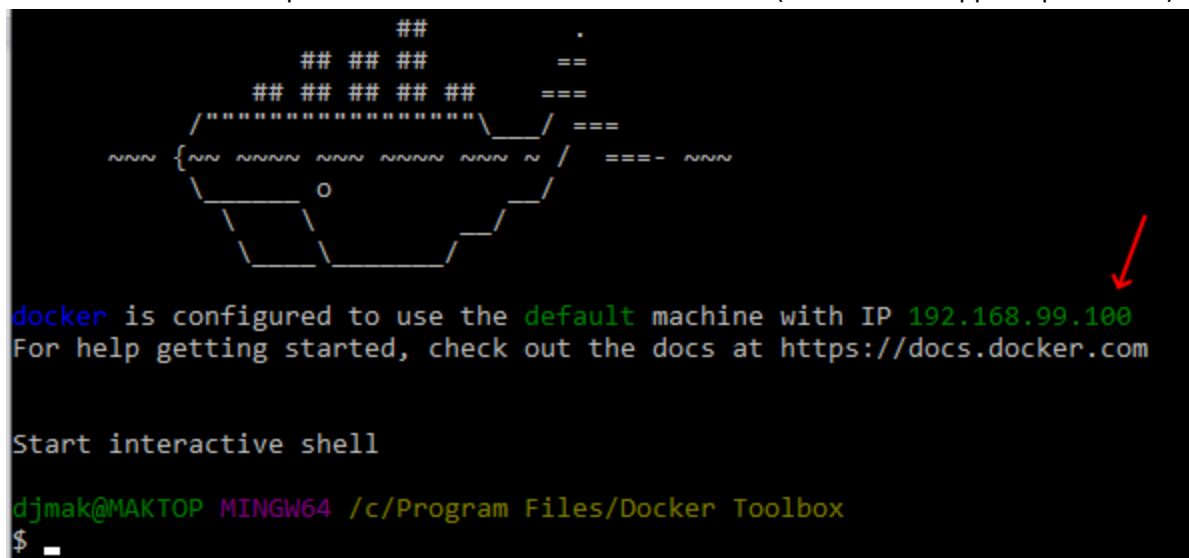
<https://docs.docker.com/install/linux/docker-ce/ubuntu/>

Run Docker

- If using VirtualBox, make sure VirtualBox is running
- Open docker terminal
- Navigate to DemocracyLab Folder (installed from git)
- ensure docker-compose is installed: <https://docs.docker.com/compose/install/>
- Add a file called .env if it does not exist by copying example.env to .env.
- run `docker-compose build`
- run `docker-compose up`

View Page

Check the Docker startup console to see where the site is hosted (make sure to append port :8000)



```
##
## ##
## ## ##
#####
NNN { NN NNNN NNN NNNN NNN N / ===- NNN
      O
      /
      /
      /

docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com

Start interactive shell

djmak@MAKTOP MINGW64 /c/Program Files/Docker Toolbox
$
```

Required: Environment Variables file

Copy file 'example.env' to file '.env' and set environment variables there

Code Changes

To push code changes up to docker image, run the following

After Python changes

Python changes are reflected automatically after .py files are saved.

After JS Changes

- `docker-compose up`

Running Commands within Docker container

Prefix command with “docker exec ID”, where ID can be gotten from:

```
docker ps --filter "name=civictechexchange-web" --format "{{.ID}}"
```

Option 3: Setting up Environment On Local Box

Disclaimer

We highly recommend going with the [VM option](#) or the Docker option if at all possible.

Clone Git Repository

In the directory you want to work in:

git clone <https://github.com/DemocracyLab/CivicTechExchange.git>

Install Python

<https://www.python.org/downloads/>

Required Python version: 3.6.9

pip install -r requirements.txt (only need to ‘sudo’ if not using virtual environment)

Install Node.js

Ubuntu

```
curl -sL https://deb.nodesource.com/setup\_8.x | sudo -E bash -  
sudo apt-get install nodejs
```

Install Yarn package manager

Windows

<https://yarnpkg.com/lang/en/docs/install/#windows-stable>

Ubuntu

```
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -  
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.d/yarn.list  
sudo apt-get update && sudo apt-get install yarn
```

Install PostgreSQL

Mac

Ubuntu

<TBD>

Windows

<https://www.postgresql.org/download/windows/>

Configure PostgreSQL database

1. Create Database for DemocracyLab
2. Create Admin user
3. Set the following environment variable:
DL_DATABASE= “{'default': {
 'ENGINE': 'django.db.backends.postgresql',
 'NAME': '<database name>',
 'USER': '<admin user name>',
 'PASSWORD': '<admin user password>',
 'HOST': '127.0.0.1',
 'PORT': '<5432 unless configured differently>',
}}”

Set up other Environment Variables

For list, see [democracylab environment variables.sh](https://democracylab.github.io/environment-variables.sh) (Note: It's not advised to use this file directly for your environment setup, but instead make a copy you can customize for your environment outside of your git repository folder so any sensitive info doesn't get checked in accidentally)

Run Development Server

Run

python manage.py runserver

Navigate to <http://127.0.0.1:8000/>, and if you see the site, you're ready to go!

S3 Setup

Option 1: Use Test S3 Bucket

Ask Marlon for credentials.

NOTE: Please do not share or check in these credentials. Besides the security concerns, AWS will disable the key if it detects the key in GitHub, forcing everyone to get new credentials.

Option 2: Set up your own S3 Bucket

Create Bucket

1. Create a bucket in S3
2. Add bucket policy
 - a. Navigate to Bucket, click Permissions -> Bucket Policy
 - b. Enter the following:

```
{
  "Version": "2012-10-17",
  "Id": "Policy1508265135508",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::<your bucket name>"
    },
    {
      "Sid": "Stmt1508265128414",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:PutObject",
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::<your bucket name>/*"
    }
  ]
}
```
3. Add CORS configuration

- a. Navigate to Bucket, click Permissions -> CORS configuration
- b. Enter the following


```
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration xmlns="https://s3.amazonaws.com/doc/2006-03-01/">
<CORSRule>
  <AllowedOrigin>*</AllowedOrigin>
  <AllowedMethod>GET</AllowedMethod>
  <AllowedMethod>POST</AllowedMethod>
  <MaxAgeSeconds>3000</MaxAgeSeconds>
  <AllowedHeader>*</AllowedHeader>
</CORSRule>
</CORSConfiguration>
```

Create IAM User

1. Navigate to 'My Security Credentials' -> 'Get Started with IAM Users' -> Click 'Add user'
2. Name user(something like 'democracyLab_app' and check 'Programmatic access'. click 'Next'
3. Click 'Attach existing policies directly' -> Create Policy -> Create Your Own Policy
4. Policy Name: user_s3_access
Description: This lets the democracylab account access S3
Policy Document:


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "*"
    }
  ]
}
```
5. Click 'Validate Policy' to verify you pasted it right, then 'Create Policy'
6. On the IAM user page, click the 'Security credentials' tab and 'Create access key'
7. A key will be generated with a secret access key THAT YOU CAN ONLY SEE THIS ONCE.
Either copy and paste it somewhere safe, or download .csv file.

Set Environment Variables

In the environment where you run the django server, set the following environment variables:

AWS_ACCESS_KEY_ID=<IAM user's Access key ID>

AWS_SECRET_ACCESS_KEY=<IAM user's secret access key>

S3_BUCKET=<The name of your S3 bucket>

Running the Site

Setting up Database

Run:

...

```
$ python manage.py createcachetable
```

```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
```

...

Building Static Assets

Run:

```
npm run build
```

For a faster build:

```
npm run dev
```

Running Development Server

Run:

```
python manage.py runserver
```

Load Test Data

Delete Data first:

```
python manage.py flush --noinput
```

Load Test Data:

```
python manage.py loaddata testdata.json
```

Admin Console

<http://127.0.0.1:8000/admin/>

Admin Credentials

VM

username: dev-admin

Password: code4Gud!

Test Data

See

<https://github.com/DemocracyLab/CivicTechExchange/blob/511540374580491965e2541f23edb55f7e639a37/common/fixtures/testdata%20readme.txt#L14>

Git Steps

Set User Information

```
git config --global user.name "<Your Name>"
```

```
git config --global user.email "<Your Email>"
```

Sync Latest Changes on Branch (green text optional steps if you have changes in progress)

```
git stash save
```

```
git checkout <branchname>
```

```
git fetch
```

```
git reset --hard origin/<branchname>
```

```
git stash pop
```

```
--Fix Conflicts--
```

Update Dependencies

Python

pip install -r requirements.txt

Node.js

yarn install

Merging Changes to Master

git checkout master

git pull

git checkout <feature branch>

git merge master

-- Test to make sure your changes play nicely with the changes from master, fix any conflicts--

git push origin <feature branch>

Test Frameworks

JavaScript Testing Framework

[Jest](#) is Facebook's testing framework, based on Jasmine.

To run the suite of tests, use the command `npm test` or simply `npm t`

The test runner will run all files that end in `.test.js`

API calls can be mocked using [jest-fetch-mock](#).

Snapshot testing is possible using [react-test-renderer](#).

It can render React components to pure JavaScript objects, without depending on the DOM.

Django Testing Tools ([docs](#))

[Running tests](#)

To run the suite of tests, use the command `python manage.py test`

The test runner runs all files named like `test*.py`

[Placing tags on tests](#) facilitates running only a particular subset.

Database fixtures can be automatically generated using [django-autofixture](#).

Troubleshooting

Cache table not found

Run `python manage.py createcachetable`

Front-end changes not showing up

Make sure you have

- `npm run build` or `npm run dev`
- Not edited SCSS files inside the `staticfiles` directory, which gets overwritten after every build
- Hard refreshed in the browser (or cleared browser cache as a last resort)

Development Process

Working on a Feature

1. Sync master branch to bring in latest changes

Commands:

```
git checkout master
git pull
```

2. Create a new feature branch

Run Commands:

```
git checkout -b <insert new branch name> master
```

Note: When choosing a name for the branch, it's helpful to pick one that describes the feature or changes to be made

Note#2: Generally we recommend working from DemocracyLab/CivicTechExchange rather than from a fork, as it simplifies test deployments. We generally try to give write access during the onboarding process, but please reach out if we haven't added you.

3. Implement feature and ensure it is working locally

1. Make code changes
2. Deploy them locally and test to make sure they work
3. Commit code changes. Use commit names descriptive of the change. ('git commit -m "Changed feature X"')

4. Submit pull request

Run command:

```
git push origin <your branch name>
```

Navigate to <https://github.com/DemocracyLab/CivicTechExchange/pulls>

Click 'New Pull Request'

Select your branch in the field 'Compare: ', then click 'Create pull request'

On the next page, fill out summary of your changes, and click 'Create pull request'

- In the body of the changes, add in 'closes #XXX' where XXX is the id of any issue(s) you're working on.

Notify team in slack #developer channel

Wait for someone to review

If any changes are needed, committing and pushing those changes should update the PR automatically.

5. Merge any changes from master (if needed)

Run commands:

```
git checkout master
```

```
git pull
```

6. Merge feature into master

This is something the engineering lead currently takes care of.

Style Guide

This guide will explain how DemocracyLab CSS is structured and how to write a new stylesheet, how to manage third-party CSS, and offer some best practice suggestions. DemocracyLab uses SASS to compile many partial files (.scss) into one minified stylesheet which is served to the end user. We have one partial per React component so our component structure is mirrored by our style structure.

Structure

DemocracyLab's style structure consists of three main parts:

- a **partials** directory, where all the DemocracyLab written CSS resides
- a **vendors** directory, where all third-party CSS resides, such as Bootstrap.
- **styles.scss** which is the root file for the SASS compiler - any CSS you write must be referred to in styles.scss or else SASS won't make use of it.

To add a new SASS partial, and using the example of a React component named SiteSearch.jsx. First, create a new file named `_SiteSearch.scss` in the `/partials/` directory and put your CSS rules here. Note the filename should match the React component name with the addition of a leading underscore. Next, import your partial in `styles.scss` - in this example, add `@import 'partials/SiteSearch';` Note the leading underscore and file extension are not required in the import statement. Finally, `npm run build` and the new stylesheet will be compiled as part of the build process.

Third-party CSS

All third-party CSS should reside in the `/vendor/` directory, to keep a clear distinction between what DemocracyLab uses that someone else wrote and what we've written.

DemocracyLab uses Bootstrap 4 as the foundation for the DemocracyLab site style, especially for layout and responsive breakpoints. We have the entire Bootstrap 4 style available but only load the parts we currently need. If you find a Bootstrap class isn't working properly, check `styles.scss` to make sure the correct Bootstrap partial is being loaded. For further information, see Bootstrap's documentation: <https://getbootstrap.com/docs/4.0/getting-started/introduction/>

Best Practices

Generally, whatever approach works for you and the site is fine, but for style consistency with your fellow developers, these are some guidelines that might help if you're unsure:

- Unit declarations: Consistent units like px are preferred; but relative units like percent should be used if needed.
- Class naming: Semantic class names following the format of `ComponentName-className` are preferred. For example `.SiteSearch-searchButton` would style the search button inside the SiteSearch component.
- Variables: Please note the semantic distinctions; even if `$color-text-light` and `$color-background-light` are both `#ffffff`, only use the text variable for text and the background for background. If you find yourself reusing the same declaration repeatedly, and it's likely to be defined across multiple components, consider defining it as a variable.
- Responsive Breakpoints: See <https://getbootstrap.com/docs/4.0/layout/overview/#responsive-breakpoints> for breakpoint mixins if your component needs them.

Technology Reference

Django

Overview

Django is a python framework for rendering web pages on the server. We are currently transitioning away from rendering the web pages on the server, and in the future will only use it to power the API that serves the raw data used by the frontend.

Models

<https://docs.djangoproject.com/en/1.11/topics/db/models/>

Applying Database Changes

First try:

```
python manage.py makemigrations
```

```
python manage.py migrate
```

If that fails, your best bet is probably to just erase the database (which also necessitates re-creating the superuser):

```
Python manage.py flush
```

```
Python manage.py createsuperuser --username dev-admin --email <your email>
```

HTML Templates

In-Depth Overview:

<https://docs.djangoproject.com/en/1.11/topics/templates/#the-django-template-language>

Tag Reference:

<https://docs.djangoproject.com/en/1.11/ref/templates/builtins/#ref-templates-builtins-tags>

Learning Django

<https://www.djangoproject.com/>

Learning Python

React.js

React is a declarative client-side framework for single-page applications. With React, the developer constructs a component tree, wherein every component has access to `props` (information which it receives from its parent) and `state` (information which it manages itself.) Each React component is represented as a JavaScript class, and has at minimum a `render` method, which declaratively describes how the component renders itself using its `props` and `state`.

Feel free to reach out to Lowell for any questions.

Reference: <https://reactjs.org/>

Learning React

Flux

Flux is a framework for managing application state. It implements the observer pattern; objects can send updates to a Flux store, as well as subscribe to updates from a Flux store. Flux can be useful to a React application because it simplifies passing state from components that are not nearby each other in the component tree. For example, if our component tree has a root R with overall structure: C <- B <- A <- R -> D -> E -> F, it becomes tedious to have component F pass information to component C in the standard React data flow. However, with Flux, component C can subscribe to a Flux store, and component F can send updates to the same store, significantly simplifying an otherwise tedious component-by-component passing of state.

Feel free to reach out to Lowell for any questions.

Reference: <https://facebook.github.io/flux/>

Flow

Flow is a static type checker for JavaScript. On its own, there is absolutely no type checking in JavaScript -- errors are thus often encountered at runtime. With Flow, it is most common to only type the parameters and return types for functions, although individual variables can also be explicitly typed. It is also straightforward in Flow to define your own types, such as Enums and Objects.

Feel free to reach out to Lowell for any questions.

Reference: <https://flow.org/>

SASS

SASS is a CSS extension framework, where you can write regular CSS but additionally take advantage of features such as variables, mixins, nested syntax, functions to compute style rules, and more. Additionally, the SASS partial structure allows the per-component stylesheet files to be separated and easy to work with for developers without impacting end user performance.

Feel free to reach out to Peter Breen on Slack for any questions.

- Reference: <https://sass-lang.com/>
- Documentation: https://sass-lang.com/documentation/file.SASS_REFERENCE.html

Prettier

[Prettier](#) is an opinionated javascript code formatter that we've adopted to help standardize our js code formatting with a minimum of effort. The easiest way to use it is to install a plugin for [your preferred IDE](#).