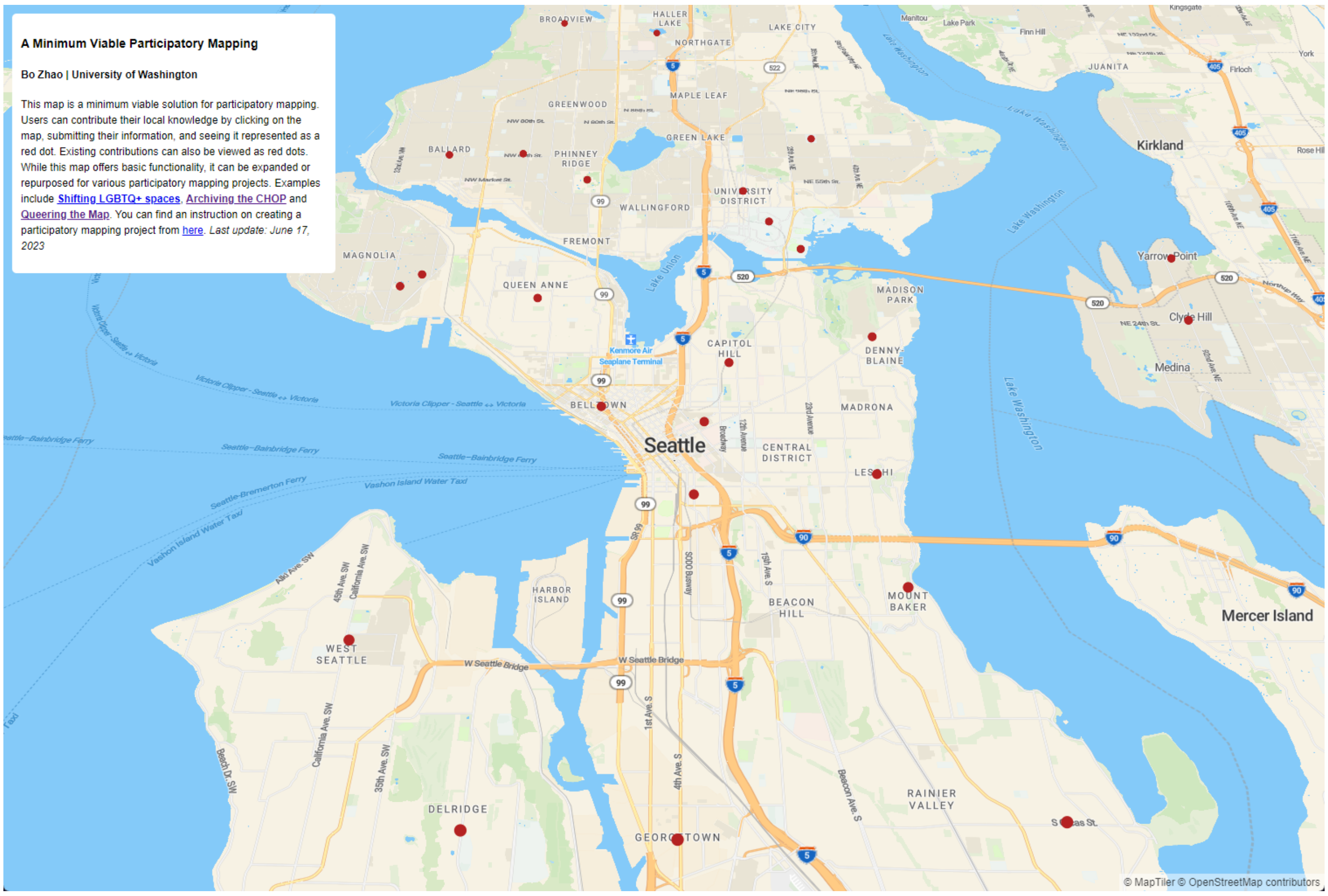
Crafting Your Own Participatory Mapping Project: A Guide

Bo Zhao | Department of Geography | University of Washington

First release: June 17th, 2023 | ***Last Update:*** June 23th, 2023

Participatory mapping, also known as community-based mapping, is a general term that refers to the process of creating maps by, for, or with local communities. It allows local communities to represent their own perceptions, knowledge, and experiences about their environment. These maps can be used for a variety of purposes, such as natural resource management, land use planning, advocacy for land rights, counter-mapping, etc. Examples include `Shifting LGBTQ+ Spaces`, `Archiving the CHOP`, and `Queering the Map`.

This tutorial helps geographers to create their own participatory mapping project. It offers a map demo (https://jakobzhao.github.io/participatory-mapping/, see figure 1) and a detailed instruction on how to create it. This map demo enables its users to contribute their local knowledge by clicking on a map, inputting their information, and viewing their input represented as a red dot. Existing contributions can also be viewed in a similar manner on the map. While this minimum viable map only offers essential functionality, it can be tailored or expanded to suit different participatory mapping initiatives. This approach can be particularly beneficial for digital geographers who want to kick-start their own participatory mapping projects.



A Minimum Viable Participatory Mapping

Bo Zhao | University of Washington

This map is a minimum viable solution for participatory mapping. Users can contribute their local knowledge by clicking on the map, submitting their information, and seeing it represented as a red dot. Existing contributions can also be viewed as red dots. While this map offers basic functionality, it can be expanded or repurposed for various participatory mapping projects. Examples include Shifting LGBTQ+ spaces, Archiving the CHOP and Queering the Map. You can find an instruction on creating a participatory mapping project from here. *Last update: June 17, 2023*

Figure 1. the screenshot of the minimum viable participatory mapping tool

Here are a few prerequisites to follow this tutorial. **Again, you do not need to be an expert, but you should be able to follow the instructions in this tutorial.** You should have:

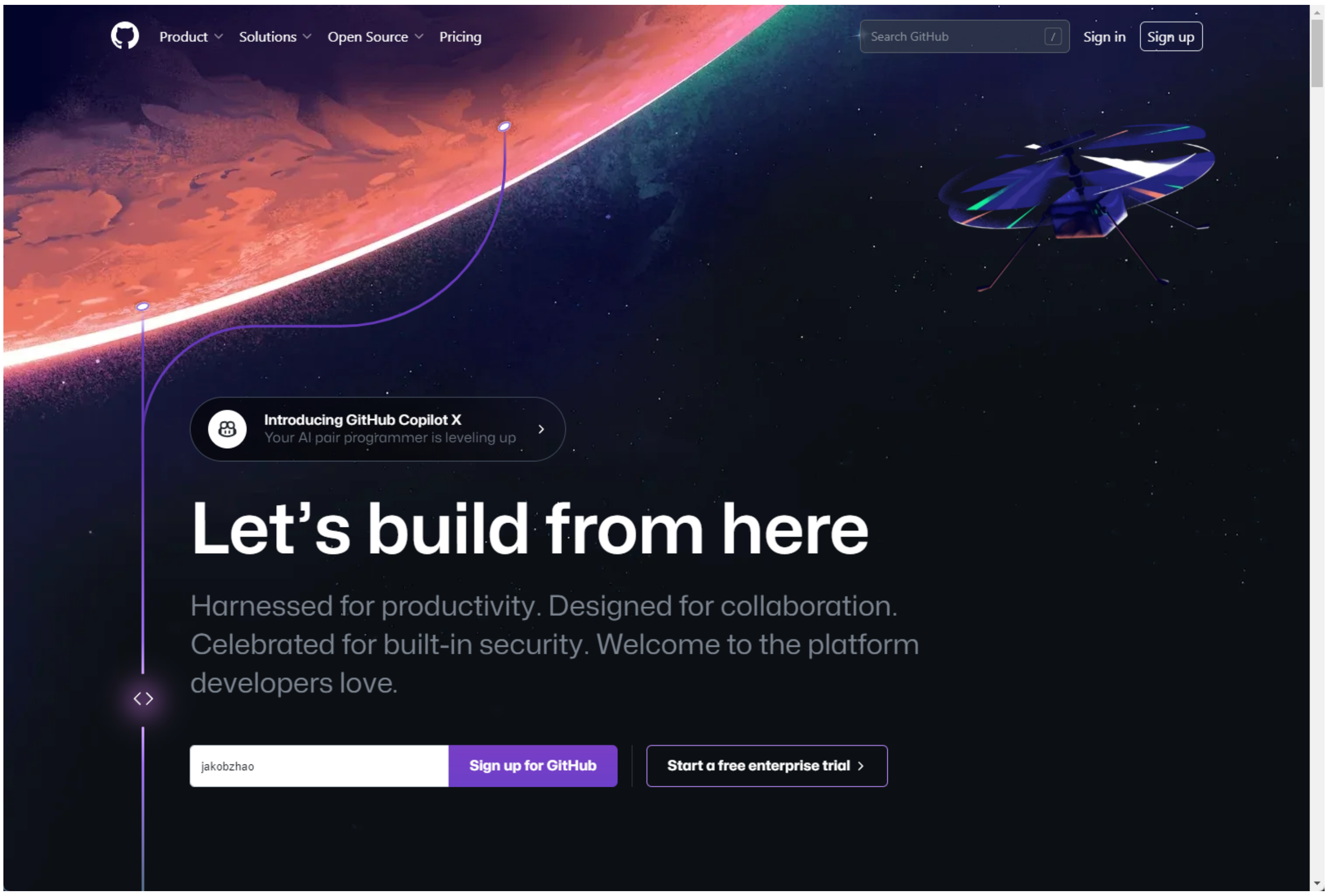
- A basic understanding of HTML, CSS, and JavaScript,
- Familiarity with the command line, Node.js, and the PostgreSQL database
- Some experience with open source web mapping, and
- Accounts on Github and Heroku, and be willing to pay $20 for the Heroku service.

> If you are not familiar with these topics, you can find many tutorials online. For example, W3Schools provides a comprehensive introduction to HTML, CSS, and JavaScript. The Command Line Crash Course is a good place to start learning the command line. The Node.js Tutorial is a good place to start learning Node.js. The PostgreSQL Tutorial is a good place to start learning PostgreSQL. The MapLibre GL JS Docs is a good place to start learning Leaflet. The Github Tutorial is a good place to start learning Github. The Heroku Tutorial is a good place to start learning Heroku. If you believe you are ready, let's get started!

2. Fork the repository

Here's a beginner-friendly overview on how to fork the repository at https://github.com/jakobzhao/participatory-mapping under your own account, and rename it to anything you want. This is the first step to create your own participatory mapping project.

1. Log into your GitHub account: Open your web browser and go to GitHub. In the top-right corner of the page, you'll see two options: `Sign in` and `Sign up`. If you have an account already, click `Sign in`. If you don't, click `Sign up` and follow the instructions to create a new account.

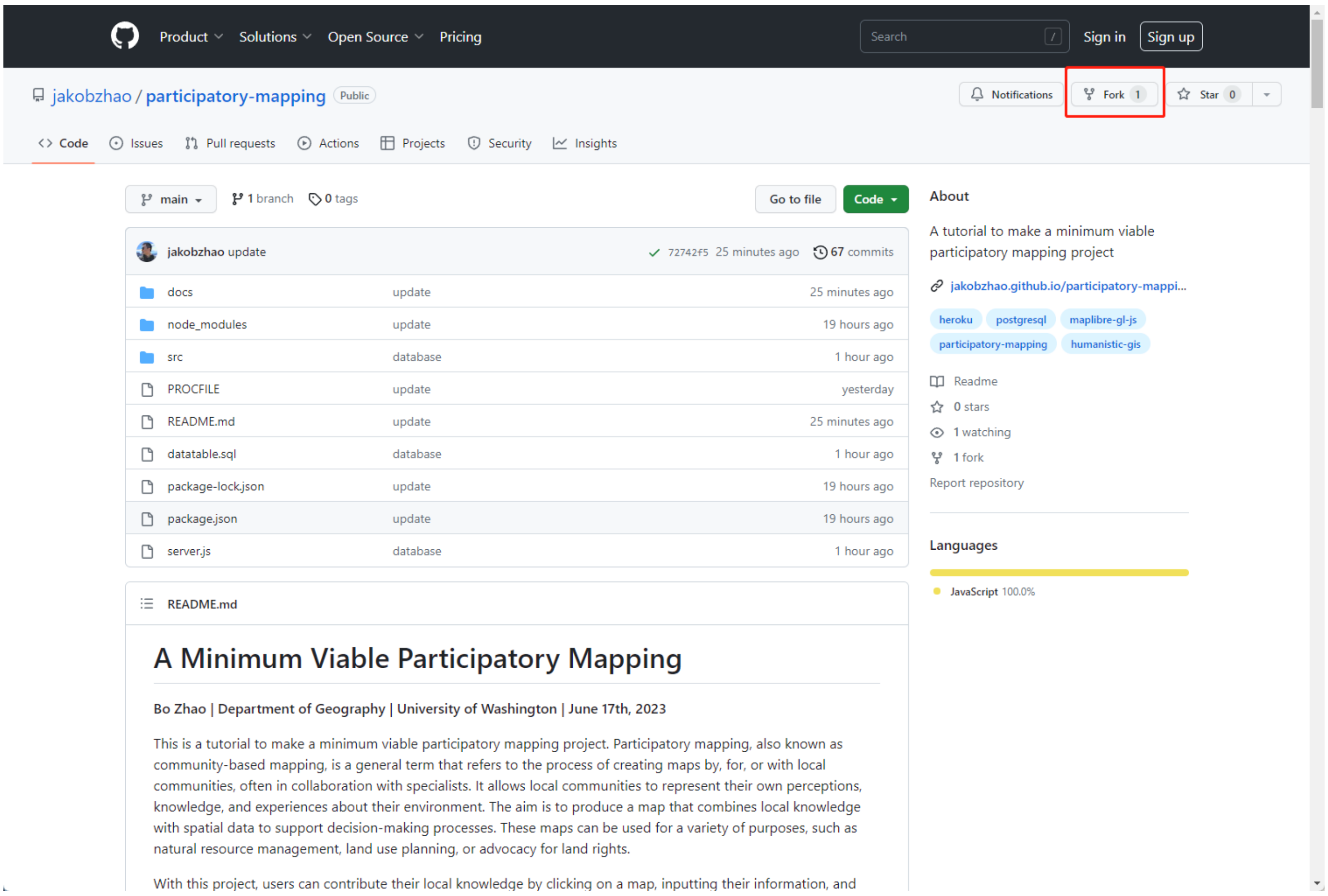

Product
Solutions
Open Source
Pricing
Search GitHub
Sign in
Sign up
Introducing GitHub Copilot X
Your AI pair programmer is leveling up
jakobzhao
Sign up for GitHub
Start a free enterprise trial

Let's build from here

Harnessed for productivity. Designed for collaboration. Celebrated for built-in security. Welcome to the platform developers love.

2. Go to the repository you want to fork: In your web browser, navigate to the specific URL of the repository you want to fork, which is https://github.com/jakobzhao/participatory-mapping. You can copy this URL and paste it into your browser's address bar.

3. Fork the repository: In the top-right corner of the page, you'll see a button that says 'Fork'. Click on it. This will direct you to a page that help you create a copy of the repository under your own GitHub account. A `fork` is essentially your own copy of someone else's project, which you can modify and use as you wish without affecting the original project.



jakobzhao / participatory-mapping Public

1 branch 0 tags

jakobzhao update ✓ 72742f5 25 minutes ago 67 commits

docs	update	25 minutes ago
node_modules	update	19 hours ago
src	database	1 hour ago
PROCFILE	update	yesterday
README.md	update	25 minutes ago
datatable.sql	database	1 hour ago
package-lock.json	update	19 hours ago
package.json	update	19 hours ago
server.js	database	1 hour ago

About

A tutorial to make a minimum viable participatory mapping project

jakobzhao.github.io/participatory-mappi...

heroku postgresql maplibre-gl-js participatory-mapping humanistic-gis

Readme
0 stars
1 watching
1 fork
Report repository

Languages

JavaScript 100.0%

README.md

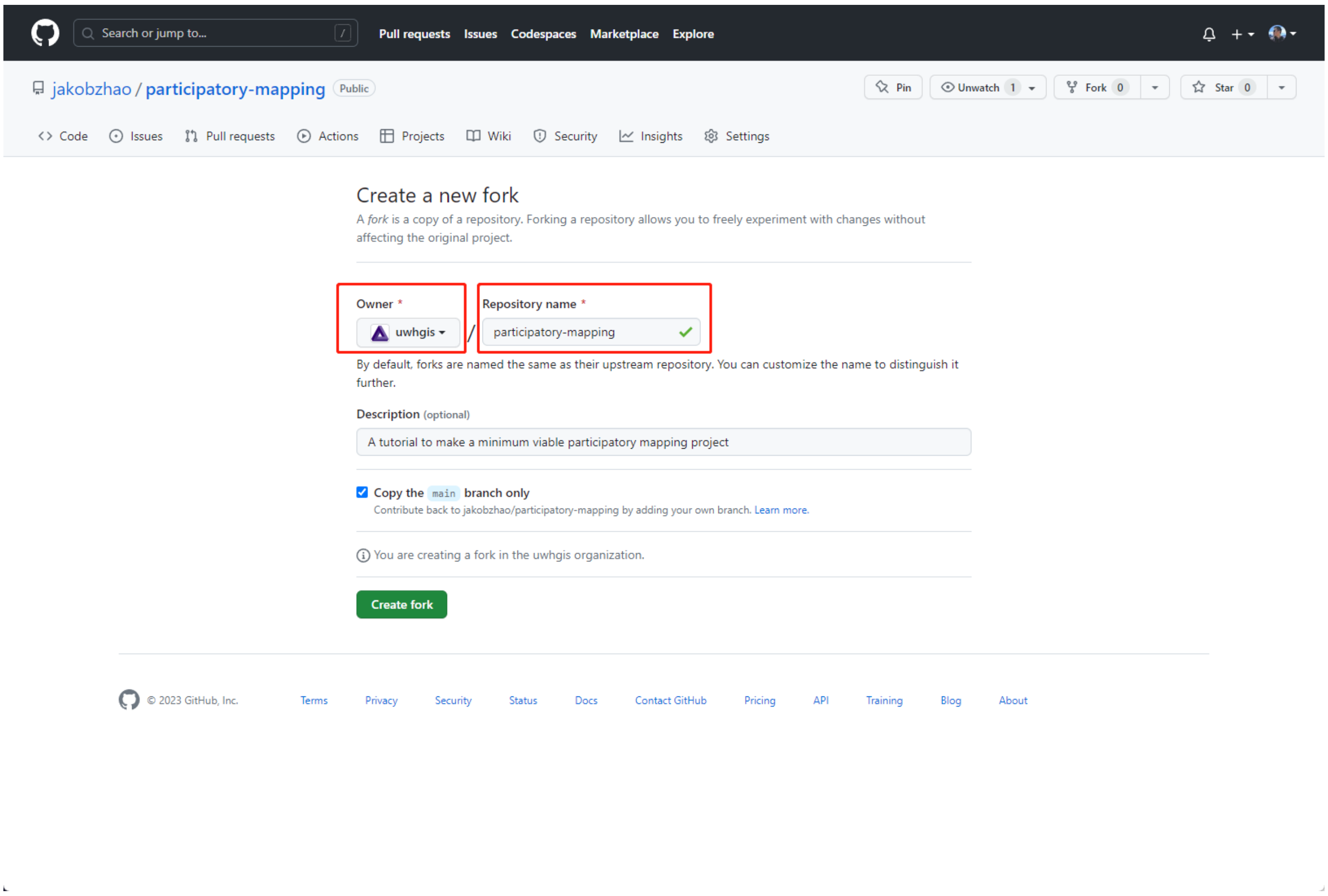
A Minimum Viable Participatory Mapping

Bo Zhao | Department of Geography | University of Washington | June 17th, 2023

This is a tutorial to make a minimum viable participatory mapping project. Participatory mapping, also known as community-based mapping, is a general term that refers to the process of creating maps by, for, or with local communities, often in collaboration with specialists. It allows local communities to represent their own perceptions, knowledge, and experiences about their environment. The aim is to produce a map that combines local knowledge with spatial data to support decision-making processes. These maps can be used for a variety of purposes, such as natural resource management, land use planning, or advocacy for land rights.

With this project, users can contribute their local knowledge by clicking on a map, inputting their information, and

4. Create the fork: You can rename your repository name, For example, you can type 'my-awesome-map' into the text box. From now on, you can access this repository at https://github.com/YourUsername/my-awesome-map, where YourUsername is replaced by your actual GitHub username.


Search or jump to...
Pull requests Issues Codespaces Marketplace Explore
jakobzhao / participatory-mapping Public
Pin Unwatch 1 Fork 0 Star 0
Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Owner * uwhgis / Repository name * participatory-mapping

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

A tutorial to make a minimum viable participatory mapping project

☑ Copy the `main` branch only

Contribute back to jakobzhao/participatory-mapping by adding your own branch. Learn more.

You are creating a fork in the uwhgis organization.

Create fork

© 2023 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

3. Understand the system mechanism

To know how the system works, we need to understand the structure of the repository, the functionality of the main files, and the system workflow.

3.1 repository structure

In order to comprehend the functioning of the system, it is essential to delve into the repository's structure, familiarize ourselves with the core files' functionalities, and grasp the overall workflow of the system. The structure of the repository and the functions of the contained files are show below:

```
├── README.md                    // project instruction
├── docs                         // the web client
│   ├── index.html
├── src                          // the web server
│   ├── config
│       ├── database.js
│   ├── controllers
│       ├── product.controller.js
│   ├── routes
│       ├── index.js
│       ├── product.routes.js
│   ├── app.js
├── node_modules                 // server-side dependencies
│   ├── ...
├── server.js                    // the main server file
├── package.json                 // the configuration file
├── database.sql                 // the database schema
├── package.json                 // the config file (automatically
generated)
├── package-lock.json            // the config file (automatically
generated)
├── PROCFILE                     // the config file for Heroku
```

- `docs`: This is a folder where the static files are stored. As a minimum viable solution, it only has one file `index.html`, but you can contain additional files to enrich your own project. In our project, this is the location where the web client is stored.
    - `docs/index.html`: serves as the main HTML document for the web application, which is a participatory mapping tool.
- `node_modules`: This is a folder where Node.js modules (or packages) are stored. When you use Node.js and install packages using the Node Package Manager (npm), those packages are placed in this folder. Node modules will build up the server and deal with a lot of the back-end functionality that you do not need to worry about.
- `src`: This is typically where the source code of the application is stored. For a Node.js application, this would usually contain JavaScript files. From a web gis system perspective, this is the location where the server is stored.
    - `src/config`: This is a folder where the configuration files are stored. In this case, it contains a file called `database.js`, which is used to configure the database connection. within this folder, `database.js` is used to configure the database connection. It contains the database connection parameters, such as the host, user, password, and database name.
    - `src/controllers`: This is a folder where the controllers are stored. In this case, it contains a file called `product.controller.js`, which is used to control the product. `product.controller.js` is used to control the product. It contains functions to create, retrieve, update, and delete products.
    - `src/routes`: This is a folder where the routes are stored. In this case, it contains two files: `index.js` and `product.routes.js`. The `index.js` file is used to define the routes for the application, and the `product.routes.js` file is used to define the routes for the product.
    - `src/app.js`: This is the main application file. It is used to configure the application and start the server.
- `PROCFILE`: A Procfile is a mechanism for declaring what commands are run by your application's dynos on the Heroku platform. It is used to explicitly declare what command should be executed to start your app.
- `README.md`: This is a markdown file that usually contains information about the software, such as what it does, how to install it, how to use it, and sometimes, how to contribute to it. It's the first thing people see when they visit the repository on GitHub.
- `datatable.sql`: This appears to be an SQL file, which typically contains SQL commands for interacting with a database. This could be creating tables, inserting data, querying data, etc. In this

case, it is likely used to set up the database structure for the participatory mapping application.

- `package-lock.json`: This is an automatically generated file by npm, which is used to keep track of the exact version of every package that is installed. This helps to ensure that the dependencies remain the same on all machines the project is installed on.

- `package.json`: This file is used in Node.js projects to keep track of all the packages (dependencies) that your project uses. This includes things like libraries and frameworks that your project depends on. It can also contain other metadata such as the project's name, description, and version.

- `server.js`: This is likely the main entry point for the application. In a Node.js application, this file usually sets up the server and starts it, often setting up routes for a web server and other server configuration.

3.2 system workflow

This workflow illustrates how the system uses a combination of static web hosting (via GitHub Pages), server-side operations (via Heroku and Node.js), and cloud-based database management (via Heroku and PostgreSQL) to provide a dynamic and interactive participatory mapping tool.

Let's delve into the system workflow step-by-step to understand how this particular system operates and how you can construct a similar one.

1. **Accessing the Website:** When a user types in https://jakobzhao.github.io/participatory-mapping/ in their browser, they are directed to the `index.html` file in the `docs` folder. This is because **GitHub Pages**, which is being used to host this site, serves static files and the docs folder is set as the root directory for the website.

2. **Client-Side Operations:** The `index.html` page serves as the client-side of the application, housing the primary functions of the participatory mapping tool. It can dispatch requests to, and receive responses from, the web services hosted on Heroku. For instance, when a user fills out a form and clicks the `submit` button, the client-side code sends a request to the server-side code, which then stores the user-contributed data in the database. The function `submitNewRecord` in the `index.html` file manages the data submission process. In this function, the code snippet `await fetch('https://participatory-mapping-70cdde6a8df5.herokuapp.com/api/add-record', settings)` forwards the data to the server-side code via the `/add-record` route. The `addRecord` function in the `controllers/productController.js` file processes the data and stores it in the database.

3. **Server-Side Operations:** The server-side of the application, hosted on Heroku, uses Node.js for coding. Heroku, a cloud-based platform, enables developers to build, run, and operate applications. The commands to kickstart the server on Heroku are detailed in the PROCFILE. The files `server.js`, `app.js`, and `routes/*.js` are primarily used to set up the server, create web applications, and establish routes to the services. For example, the code snippet `router.post('/add-record', productController.addRecord)` triggers the `addRecord` function in the `controllers/productController.js` file when the client-side code sends a request via the `/add-record` route.

4. **Web Services:** The server-side code handles client-side requests, processing them to return the appropriate response. These web services can retrieve existing data from the database or store new

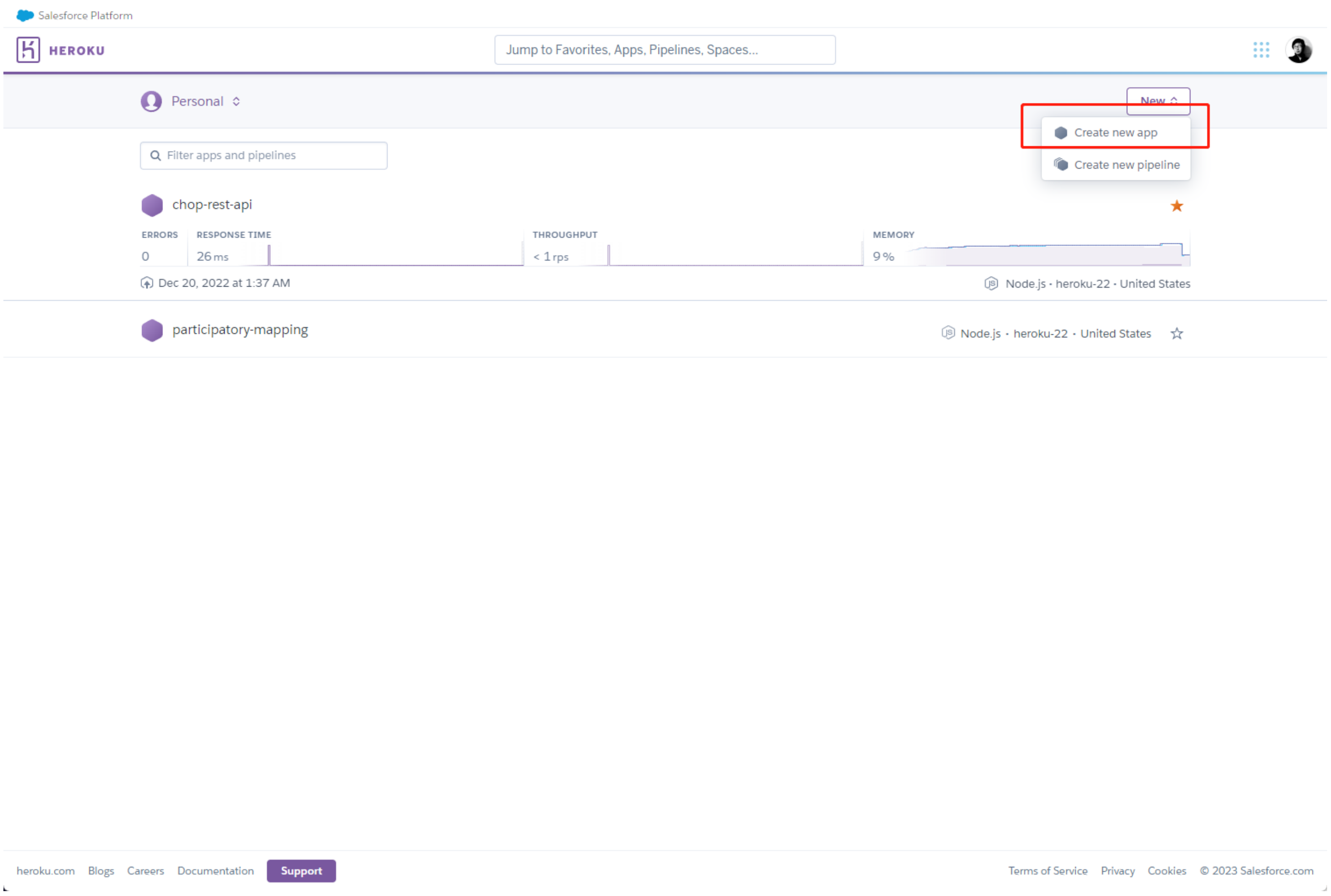
user-contributed data. For example, the `addRecord` function, defined in `controllers/productController.js`, extracts the contributed data (e.g., contributor, content, lat, lng) and stores them in the database. The code snippet `let {recordRows} = await db.query('INSERT INTO` tblRecord`(contributor, content, lat, lng) VALUES ($1, $2, $3, $4)', [contributor, content, lat, lng])` is used for this purpose.

5. **Database Operations:** The application uses a PostgreSQL database to store and manage data. This database is cloud-hosted through Heroku. Connection information for this database is provided in the `src/config/database.js` file, ensuring successful execution of the SQL statement in the data query method `db.query()`.

4. Deploy your project on Heroku

To deploy your forked GitHub repository to Heroku, you'll need to follow these step-by-step instructions. Before starting, ensure you have a Heroku account. If you don't have one, sign up for free at https://signup.heroku.com/.

- Create a new app on Heroku:
  - Log in to your Heroku account.
  - On the Heroku dashboard, click the `New` button, then select `Create new app`.
  - Give your app a unique name (e.g., `my-app-name`).
  - Choose the region closest to your location.
  - Click the `Create app` button.

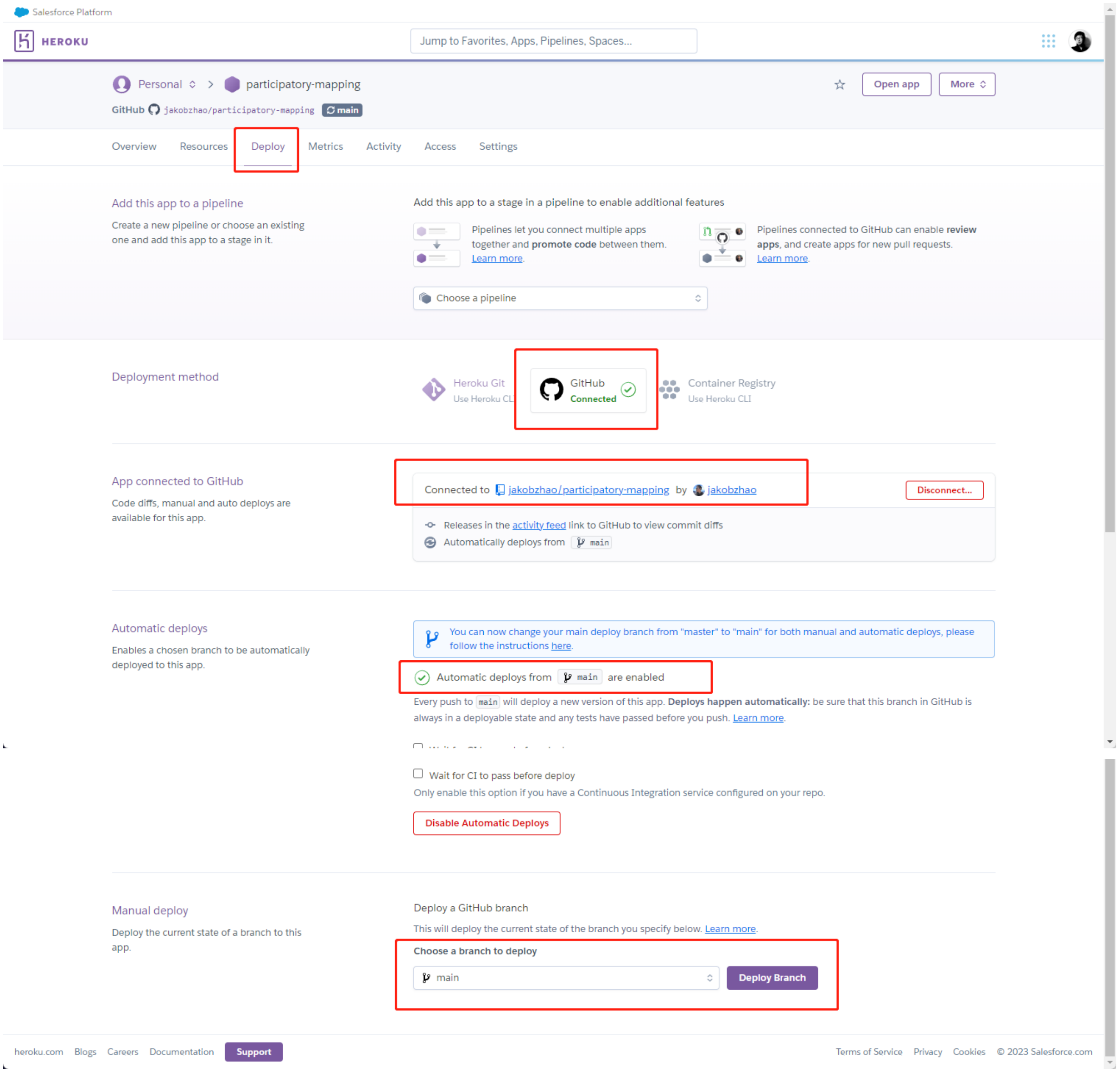


- Connect the Heroku app to your GitHub repository:

  - On the `Deploy` tab of your Heroku app dashboard, under `Deployment method`, choose `GitHub`.
  - In the `Connect to GitHub` section, search for the repository you forked and click the `Connect` button.

- Set up the build command:

  - In the `Deploy` tab of your Heroku app dashboard, scroll down to the `Manual deploy` section.
  - Click the `Deploy Branch` button to manually deploy the app for the first time.
  - In the `Automatic deploys` section, enable automatic deploys if desired.
  - In the `Manual deploy` section, make sure the correct branch is selected.



Salesforce Platform

HEROKU Jump to Favorites, Apps, Pipelines, Spaces...

Personal > participatory-mapping Open app More

GitHub jakobzhao/participatory-mapping main

Overview Resources Deploy Metrics Activity Access Settings

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Choose a pipeline

Deployment method

Heroku Git Use Heroku CLI | GitHub Connected | Container Registry Use Heroku CLI

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to jakobzhao/participatory-mapping by jakobzhao Disconnect...

Releases in the activity feed link to GitHub to view commit diffs

Automatically deploys from `main`

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

✓ Automatic deploys from `main` are enabled

Every push to `main` will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. Learn more.

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Disable Automatic Deploys

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to deploy

main Deploy Branch

heroku.com Blogs Careers Documentation Support Terms of Service Privacy Cookies © 2023 Salesforce.com

  - In the `Settings` tab of your Heroku app dashboard, Scroll down to the `Buildpacks` section and click on the `Add buildpack` button.
  - Choose the appropriate buildpack for your application. Since we are deploying a Node.js app, select the `Node.js` buildpack.
  - Click the `Save changes` button to update the buildpacks.

Congratulations! You have successfully deployed your forked GitHub repository to Heroku. The app is now live and accessible to the public. You can continue making changes to your repository and deploying updates to Heroku using the configured automatic deployment or manually triggering deployments. Remember to regularly check your Heroku app dashboard for logs and potential issues. You can also monitor your app's performance, scale it as needed, and manage other aspects of your deployed application through the Heroku dashboard.

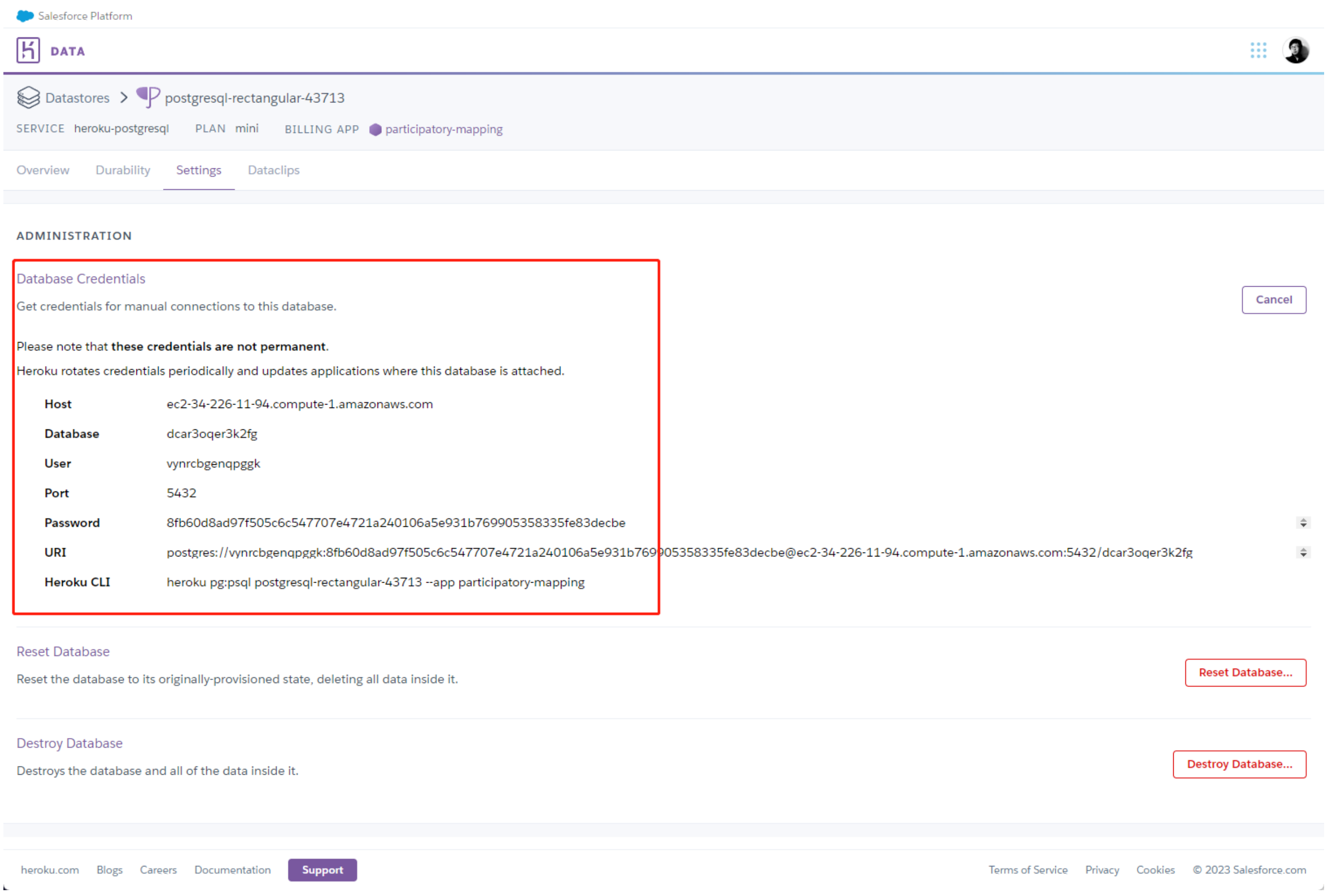
5. Initialize the database

This participatory mapping tool enables the data management through a cloud-based PostgreSQL database that is offered by Heroku. Initializing the database on Heroku involves a few steps. Here is a step-by-step guide on how to do it.

- Add a PostgreSQL add-on: In your Heorku app dashboard, click on the `Overview` tab. Under the `Installed add-ons` section, click on `Configure Add-ons`. In the `Add-ons` search bar, type `Heroku Postgres` and select it. FOr testing purpose, the mini plan costing $5 per moth is enough.

- Retrieve the Database URL: After adding the PostgreSQL add-on to your app, you need to retrieve the database URL. Here are the steps:

    - Click on the `Heroku Postgres` add-on under the `Resources` tab. This will take you to the Heroku Postgres dashboard.
    - In the Heroku Postgres dashboard, click on the `Settings` tab.
    - Click on the `View Credentials` button. This will display the credentials for your PostgreSQL database, including the database URL.



SERVICE heroku-postgresql PLAN mini BILLING APP participatory-mapping

ADMINISTRATION

Database Credentials

Get credentials for manual connections to this database.

Please note that **these credentials are not permanent**.

Heroku rotates credentials periodically and updates applications where this database is attached.

Host	ec2-34-226-11-94.compute-1.amazonaws.com
Database	dcar3oqer3k2fg
User	vynrcbgenqpggk
Port	5432
Password	8fb60d8ad97f505c6c547707e4721a240106a5e931b769905358335fe83decbe
URI	postgres://vynrcbgenqpggk:8fb60d8ad97f505c6c547707e4721a240106a5e931b769905358335fe83decbe@ec2-34-226-11-94.compute-1.amazonaws.com:5432/dcar3oqer3k2fg
Heroku CLI	heroku pg:psql postgresql-rectangular-43713 --app participatory-mapping

Reset Database

Reset the database to its originally-provisioned state, deleting all data inside it.

Destroy Database

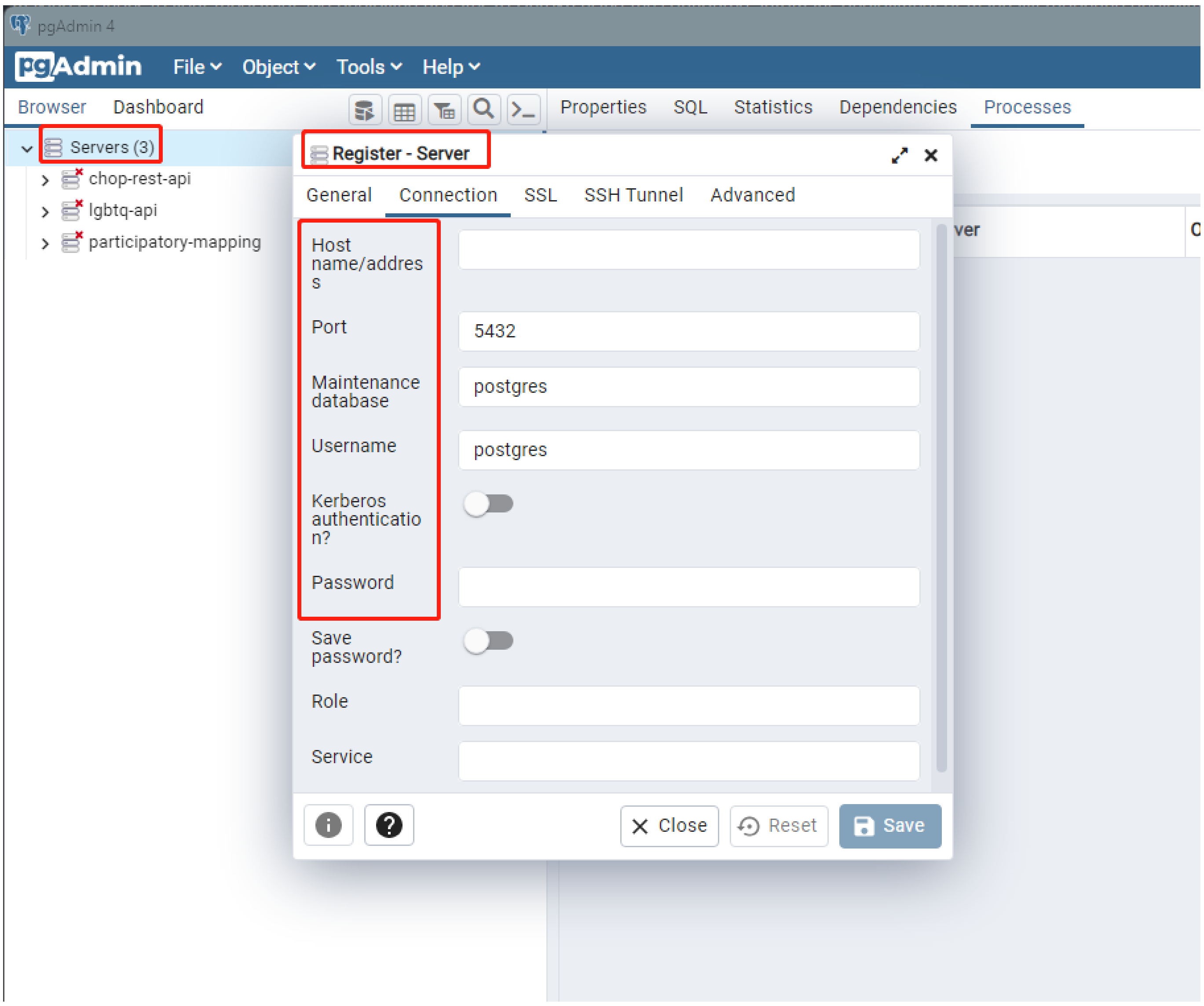
Destroys the database and all of the data inside it.

- Connect to your Database: To connect to the database, you can use any PostgreSQL client, such as `pgAdmin`. You need to have pgAdmin installed on your machine. If you don't have it yet, you can download it from the official pgAdmin website. The installation process may vary depending on your operating system.

- After installing pgAdmin, open it. You will see the pgAdmin dashboard. In pgAdmin, servers are essentially connections to databases. To create a new server:
  - Right-click on the `Servers` in the left panel and choose `Register > Server...`.
  - In the `Register - Server` dialog box, enter a name for the server under the `General` tab. This can be any name you like and doesn't affect the connection.
  - Click on the `Connection` tab. Here, you will enter the details of your Heroku Postgres database. These details can be found from the `View Credentials` section of your database in the Heroku dashboard.

    > - `Host`: This is the `Host` value in Heroku credentials.
    > - `Port`: This is the `Port` value in Heroku credentials.
    > - `Maintenance database`: This is the `Database` value in Heroku credentials.
    > - `Username`: This is the `User` value in Heroku credentials.
    > - `Password`: This is the `Password` value in Heroku credentials.

  - After entering these details, click `Save`.



Host name/address

Port: 5432

Maintenance database: postgres

Username: postgres

Kerberos authentication?

Password

Save password?

Role

Service

- Connect to the Database: After creating the server, you can connect to your database. In the left panel, navigate to `Servers > [Your Server Name] > Databases > [Your Database Name]`. Double-click on your database name to connect to it.

- Open SQL Query Tool: After connecting to your database, you can open the SQL Query Tool. Right-click on your database name and choose `Query Tool`. This will open a new SQL Query Tool tab.
- Initialize your Database: In the SQL Query Tool, you can now initialize your database with the provided SQL script. Paste the SQL script in the file `datatable.sql` into the query editor. After pasting the script, click on the `Execute` button (which looks like a `Play` button) to run the script. This will create your table in the database. You have now initialized a table in your PostgreSQL database on Heroku using pgAdmin.

6. Publish your map

Currently, the forked project is successfully deployed on Heroku, and the database is initialized. In this phase, we will guide you through the steps to customize the project and publish it online.

- Access the map:
  - **Method 1:** On your Heroku dashboard, go to the `Settings` tab and find the URL under the `Domains` section. This URL serves as the direct link to your map. Additionally, you can access the map by clicking the `View` button located at the top right corner of the dashboard. The URL follows a format like https://your-app-name.herokuapp-random-token.com/. For example, you can access the demo map using the URL https://participatory-mapping-70cdde6a8df5.herokuapp.com.
  - **Method 2:** If you find the Heroku URL lengthy and difficult to remember (due to the random token), you have another option. You can access your map through your-account-name.github.io. To do this, visit your GitHub repository page, navigate to `Settings`, and scroll down to the `GitHub Pages` section under the `General` tab. Here, you will find the URL for your map. Please ensure that your web map is built and deployed using the `docs` folder in the main branch. It is worth noting that, in the file `src/app.js`, please add the `https://your-account-name.github.io` (`your-account-name` in the url should be replaced by your actual account name) to the `origin` property of the `corsOptions` variable. By doing so, the data changes between heroku and `your-account-name.github.io` will be granted. This edit should be done in your forked repository, if you turn on the `automatic deploys` function, this corsOption will be immediately applied.Now, your map will be appropriately accessible via the URL https://your-account-name.github.io/your-repository-name/. For example, the demo map can be accessed through the URL https://jakobzhao.github.io/participatory-mapping/.
  - **Method 3:** If you wish to customize the domain name further, you have options available. Please refer to the GitHub documentation for more details. for detailed instructions on customizing the domain name. Alternatively, you can customize the domain through Heroku. For more information, consult the Heroku documentation.
- Updating the Info Panel:
  - To provide additional details about your map or introduce your own participatory map, you can update the information displayed in the info panel. This panel is located in the index.html file.
  - Open the index.html file and locate the section containing the title, description, and image of the info panel.
  - Modify the title, description, and image to reflect the desired information you want to present.

- Switching the Basemap:
  - Currently, the base map utilizes MapTiler's streets-v2 style. If you prefer a different style, you can explore the available map styles at www.maptiler.com.
  - Sign up for MapTiler to access the token for each free map style at https://cloud.maptiler.com/maps/.
  - When copying the map style, ensure you select the vector style option.
  - In the index.html file, locate the code snippet responsible for setting the basemap.
  - Replace the existing map style token with the newly copied token. This update will switch the basemap to the desired style.

  ```
  // Create a new map instance
  let map = new maplibregl.Map({
    container: 'map', // container id
    style: 'https://api.maptiler.com/maps/streets-v2/style.json?
  key=Cusoe5zmfmn26glFoeoe', // style URL
    center: [-122.3321, 47.6062], // starting position [lng, lat]
    pitch: 45, // pitch in degrees
    zoom: 12 // starting zoom
  });
  ```

  > For more advanced approaches to switching the basemap, you can refer to the maplibre examples.

- Collect your data: This minimum viable map only allows you collect the contributor's name, their message, and the location of the message. You can customize your forked repository to collect your own dataset. A few edits across the repository are required to enable its functionality. You have the option to edit the repository's code using an Integrated Development Environment (IDE) on your local machine or GitHub's web editing interface on the cloud. To learn more about how to edit a repository through the web interface, please check out this tutorial. In this tutorial, we will primarily focus on editing the repository through GitHub's web interface for simplicity. If you prefer using an IDE, most web developers recommend Visual Studio Code (VS Code). You can follow the steps below to edit the repository using VS Code.

  > Once forked and renamed, you can clone the repository to your local machine. To do this, you'll need to have Git installed on your computer. If you don't have Git installed, you can download it from `https://git-scm.com/downloads`. Once Git is installed, you can clone the repository to your local machine. To do this, you will need to use either Command Prompt on Windows or Terminal on Mac OS to complete this task. After opening the command prompt or terminal, please navigate to a desired directory for managing your projects using `cd`, like `cd C:\YourDirectory` in command prompt or `cd ~/YourDirectory` in Terminal. Then, clone the forked repository by typing `git clone https://github.com/YourUsername/my-awesome-map.git`
  >
  > Once cloned to your local machine, you can edit your repository in your local machine using VS code. Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux, and MacOS. It includes support for debugging, embedded Git control, GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring. You can download it

> from the Visual Studio Code download page. To open a repository in Visual Studio Code: please go to `File` -> `Open Folder...` on Windows or `File` -> `Open...` on MacOS, navigate to your cloned repository and open it. Now, you're ready to edit files in the repository.

- Customize Data Submission:
  - Based on your data collection plan, you need to edit the data submission process in the index.html file. For example, you can add a field to collect the contributor's email address. To do this, include an email text input field in the popupContent variable.
  - Define a CSS selector to stylize the email input field.
  - Append the collected email value to newRecord using newRecord.append('email', email);.
  - Update each geojson feature to include a property to store the email variable.
- Update the addRecord Method:
  - Navigate to the src/controllers/product.controller.js file.
  - Update the addRecord method to include the email field.
  - You can refer to the existing addRecord method in the file as a reference.
- Update the Data Table Schema:
  - Use pgAdmin (PostgreSQL administration and management tool) to access the data table schema.
  - Update the schema to include the email field.

Now, your map is ready to collect your data. You can share the map with your targeting audience and collect their data on the map.

Final Thoughts

This text provides a detailed, step-by-step tutorial on how to create a minimum viable participatory mapping project using several technical skills and platforms, including HTML, CSS, JavaScript, command line, Node.js, PostgreSQL, GitHub, and Heroku. The author effectively demystifies the process of participatory mapping by explaining concepts and providing actionable instructions.

Participatory mapping, as the author noted, is a crucial tool that empowers local communities to document their knowledge and experiences about their environment, thus supporting decision-making processes.

However, the complexity of the tutorial implies that this process is best suited for those with a foundational understanding of the technologies involved. The tutorial may be overwhelming for those without a background in these technologies or novice users. Furthermore, the cost of Heroku services could also be a limiting factor for some.

Despite some limitations, with careful execution and continuous enhancements, it can serve as an empowering platform that truly reflects the lived experiences of communities. As digital geographers continue to explore this field, we can expect to see more diverse and inclusive mapping projects that capture the myriad ways in which we relate to and experience our environment.