FEF - local development

Updated Nov 4, 2026

Deploy for local development
- Installing nvm on macOS or Linux
- Clone the commons-frontend-app repository and set up nvm and Node
- Verify the default config is deploying successfully
  - Notes for local deployment

Local Development with Canine Data Commons as an example API: Update the FEF config to use a preferred data commons config and API
- Get API key for Canine Data Commons
- Create a local env file for remote access to Canine API
- Add or create config files
- Restart the dev server after an env file change
- Configure your config files for the new API

Local Login with Credentials

Config vs Code Changes

Deploy for local development

A development environment lets you run the commons locally while connecting to remote Gen3 services or Gen3 running locally on a Helm chart. This approach will let you quickly refine your config before you deploy it to production.

Local FEF development depends on the deployable app (commons-frontend-app) to customize FE configuration to your remote Gen3 API.

The highlighted sections emphasize the key actions to take.

Installing nvm on macOS or Linux

We use nvm (Node Version Manager) to manage Node and npm versions.

nvm lets you install and switch between Node.js versions. The Gen3 Frontend Framework requires Node 24.22+.

1. Install nvm

Run the install script from the nvm GitHub releases page, replacing `<version>` with the latest release tag (for example, `v0.40.x`):

- As of Nov 4, 2026 - this is:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.8/install.sh | bash
```

```shell
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/<version>/install.sh | bash
```

If you don't have `curl`, `wget -qO-` works in place of `curl -o-`.

2. Reload your shell

The script adds a few lines to your shell profile (`~/.zshrc` on modern macOS, `~/.bashrc` on most Linux distros).

Either close and reopen your terminal, or run:

```shell
source ~/.zshrc    # or ~/.bashrc
```

3. Verify

Verify the nvm function is loaded and accessible in your current terminal by running the command below.

- If it is, it will output `nvm`. Proceed to step 4.
- If it is not (outputs nothing, or an error), retry steps 1 and 2.

```shell
command -v nvm     # should print "nvm"
```

Verify the version of nvm is ≥11.19.

```shell
npm -v # confirm npm version ≥11.19
```

4. Install and use Node

```shell
nvm install 24     # installs Node 24.x
nvm use 24         # switches to it
```

```
node -v            # confirm version ≥24.22
```

To make it the default for new terminals:

Shell

```
nvm alias default 24
```

Clone the commons-frontend-app repository and set up nvm and Node

Locally clone the commons-frontend-app repo (https://github.com/uc-cdis/commons-frontend-app).

Shell

```
git clone https://github.com/uc-cdis/commons-frontend-app.git
```

Make sure you **change to be in the commons-frontend-app directory**, then install node and npm in the directory (this will use the repo's .nvmrc file)

Shell

```
nvm install
nvm use
```

Verify the default config is deploying successfully

Now that npm is installed, use that to download and set up project dependencies:

- From the root directory for the repo, run `npm i` (a shorthand alias for `npm install`). It will:
  - Download all required libraries listed in the package.json file into a local node_modules folder
  - Generate/update a package-lock.json file to lock dependency versions
- Then run `npm run dev:turbopack`. (*see the note below about using the turbopack dev server*) It will spin up the local Next.js development server with Turbopack. With Turbopack, as you edit and save code files, the page in your browser will automatically update in real time.

In your terminal, you will see:

```
> @gen3/commons-portal-app-base@0.12.88 dev:turbopack
> next dev
```

```
▲ Next.js 16.3.6 (Turbopack)
- Local:         http://localhost:3000
- Network:       http://192.168.0.235:3000
- Environments: .env.development.local, .env.development
✓ Ready in 202ms
version: 0.12.88
✓ Running next.config.js took 71ms
```

After this finishes initializing (when it shows the Ready line), your terminal will show the local URL (eg, `http://localhost:3000`), where you can view the app in your browser. Your terminal will continue to update the status of the site. In the browser, it should show the Gen3 Data Hub as the default config and backend API.

Notes for local deployment

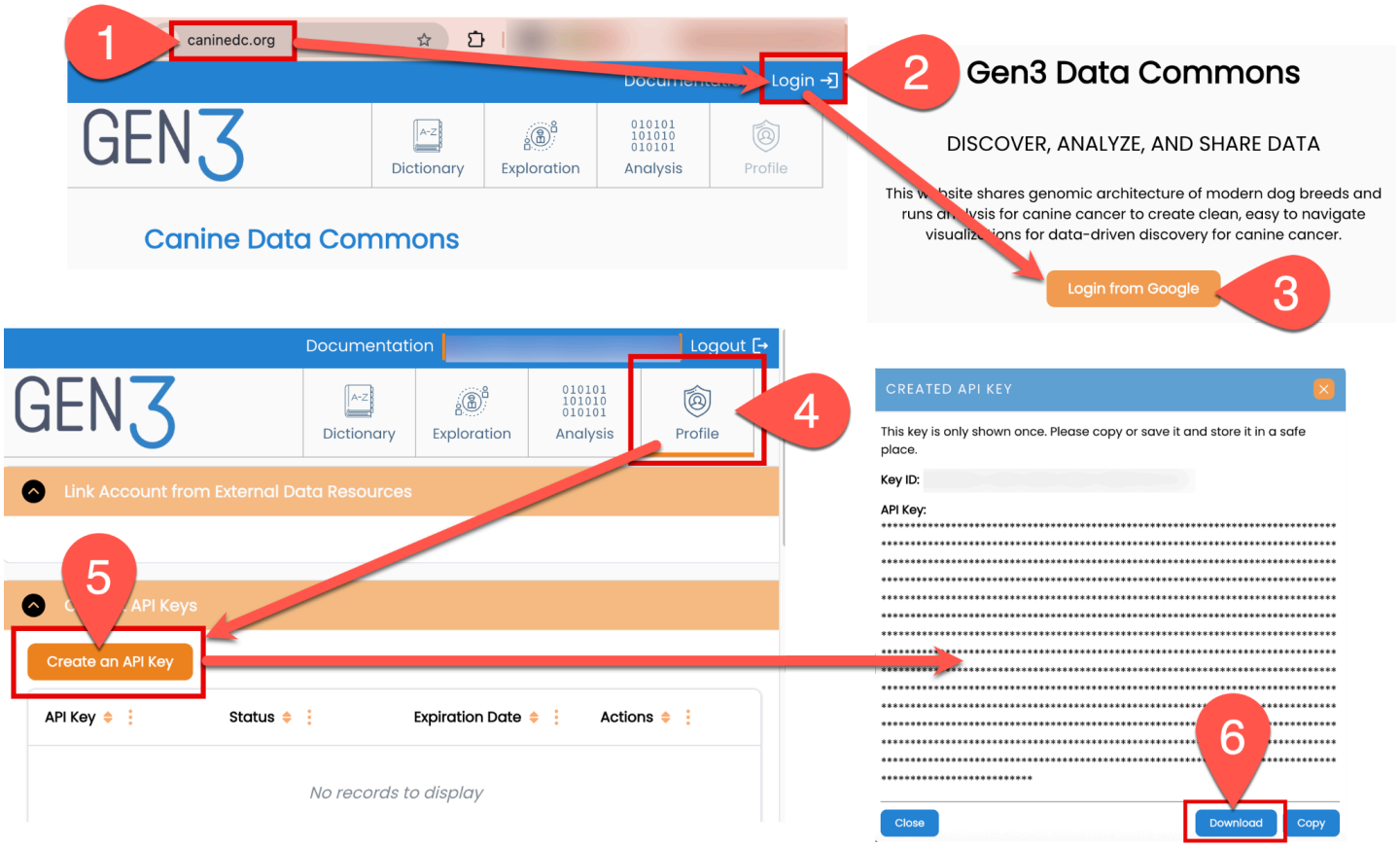
- If you use `npm run dev:turbopack` (as indicated in the instructions above), there are two features that may not work as expected:
    - The Query page will not be functional (although the page tile will still be present and the page will load). It uses unusual plugins that are not supported by webpack. If you want to USE the query page in local development, you must compile with regular webpack, but it is a slower compiler. (i.e., use `npm run dev` instead of `npm run dev:turbopack`)
    - Page protection (discussed in the Authentication and Authorization for Pages section below) will silently fail with `npm run dev:turbopack` until you also run `npm run build:middlewarePaths` after editing authz.json. Using `npm run dev` instead resolves the need to run the extra `build:middlewarePaths` script.
- If you need to terminate the deployment (e.g., so you can update the env file), use `ctrl+C`, then re-run the `npm run dev` command to restart.

Local Development with Canine Data Commons as an example API: Update the FEF config to use a preferred data commons config and API

Get API key for Canine Data Commons

The local deployment will not be able to generate an API key. Download this credential from the live site. You can use this credentials JSON to log in to your local deployment.

- Log in at https://caninedc.org/ and download an API key for Canine Data Commons



Create a local env file for remote access to Canine API

In the local root directory (i.e., the commons-frontend-app directory), there is an environment file called `.env.development`. This file tells the FEF which folder in the config directory to use for configuration, and what API to use for the source of databases. By default, it uses the Gen3 Data Hub commons:

Shell

```
# The commons_name should match the name of the folder in the config directory that holds the FEF config files.
NEXT_PUBLIC_GEN3_COMMONS_NAME=gen3
# Set the following to connect the development frontend to the API for your preferred Gen3 commons backend. Note that the configuration used in the config directory needs to match the backend, or the local deployment will be broken.
NEXT_PUBLIC_GEN3_API_TARGET=https://gen3.datacommons.io
NEXT_PUBLIC_GEN3_API=http://localhost:3000
```

To set up your development environment, make a copy of the `.env.development` in the root directory:

```
Shell

cp .env.development .env.development.local
```

Because the copy has `.local` at the end of the file name, in a Node.js environment, it will override the `.env.development` file when used in local development. But, it's included in the `.gitignore` file, so it won't be accidentally committed back to the repo.

In the `.env.development.local` file, make the following changes to point the local deployment to Canine Data Commons API (either open the file in a text editor and manually change it, or use the commands below to make the changes using the command line).

1. Remove the `NEXT_PUBLIC_GEN3_API` line *(While not absolutely necessary, removing this line avoids accidentally setting this to point at the remote commons instead of `NEXT_PUBLIC_GEN3_API_TARGET`, which would cause CORS and login errors.)*
2. Update these values:

```
Shell

NEXT_PUBLIC_GEN3_COMMONS_NAME=canine
NEXT_PUBLIC_GEN3_API_TARGET=https://caninedc.org
```

If you prefer to use the command line to make these updates:

```
Shell
cat << 'EOF' > .env.development.local
# The commons_name should match the name of the folder in the config directory that holds the
FEF config files.
NEXT_PUBLIC_GEN3_COMMONS_NAME=canine
# Set the following to connect the development frontend to the API for your
# preferred Gen3 commons backend. Note that the configuration used in the
# config directory needs to match the backend, or the local deployment will be broken.
NEXT_PUBLIC_GEN3_API_TARGET=https://caninedc.org
EOF
```

Add or create config files

It's best to start with config files that are configured to work somewhere, even if it's not the exact API you're developing for. If you already have config files, you can place them in the appropriate directories (see below).

Otherwise, copy the default Gen3 config directories in commons-frontend-app:

`/config/gen3`

`/public/images/gen3`

Note: `/public/icons` is also part of the config. However, it does not require a commons-specific directory. Make any icon changes directly in that folder.

Rename the copied directories to use the commons name, and place them next to the appropriate gen3 folder in `/config` and `/public`:

Because our `NEXT_PUBLIC_GEN3_COMMONS_NAME=canine`, we'll use

`/config/canine`

`/public/images/canine`

Restart the dev server after an env file change

Terminate the old deployment (`ctrl+c`).

Restart deployment to use the local env file using your preferred `npm run dev` command (`npm run dev:turbopack`)

Now you can view the local deployment from your local URL. The deployment is running FEF from the local `/canine` files using Canine Data Commons API as a backend.

Configure your config files for the new API

Now that you have your local development environment set up, iterate on configuring the files in `/config/canine` and `/public/images/canine` (and `/public/icons`, if you have any icon changes).

Because we used `npm run dev:turbopack`, we **only need to save the changed file for the changes to deploy** to the localhost. (You may need to reload the browser.) There are 4 exceptions to this statement:

Changes to `colors.json`	`GEN3_COMMONS_NAME=canine npm run build:colors` (regenerates `themeColors.json`)
Changes to `themeFonts.json`	Ctrl+C, `npm run dev:turbopack` (Tailwind reads this file only when the dev server starts)
Changes to `authz.json`	`npm run build:middlewarePaths` AND Ctrl+C, `npm run dev:turbopack`
Adding or changing icons in `/icons/`	`npm run build:icons` (regenerates the icon JSON)

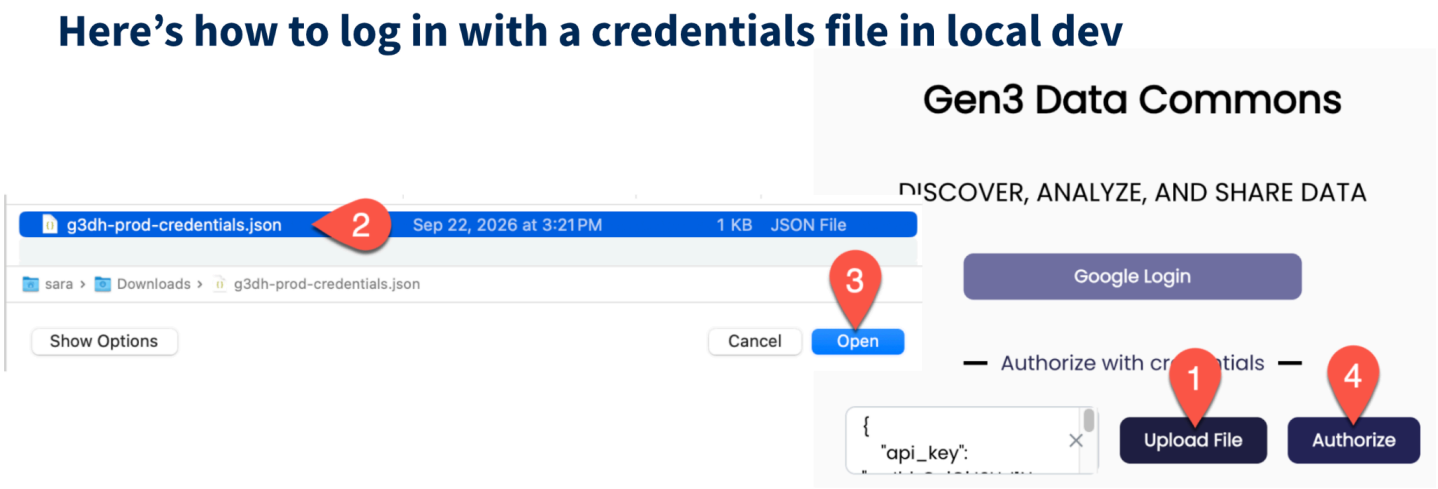
Local Login with Credentials

When you are pointing the FE to a remote backend, logging in the normal way from localhost to the remote API will not work because Fence will not allow redirect back to https://localhost. To allow testing features that require login, there is a property in the `login.json`, `showCredentialsLogin`, that enables remote login using an API key when set to `true`.

> **Note:** When serving the frontend locally with a remote backend, you will not be able to create an API key on the local Profile page.
> In the `/config/canine/login.json`, find this property and set it to `true`, then Save:

```
JSON
{
      "showCredentialsLogin" : true
}
```

You can read more about this local login feature in FEF docs.



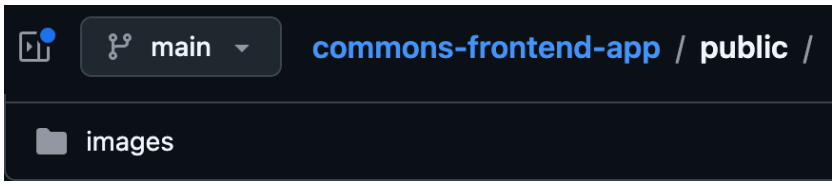
Config vs Code Changes

Below is a simplified display of directories in the commons-frontend-app. There are more directories, but these are the ones that have files you can easily update to customize your front end.



`/config` holds the configuration files

> **`/config/gen3`**: all the per-page JSON files for configuration of pages
>
> **`/config/icons`**: icon sets



`/public/images` holds images and logos for your configured frontend.

- Note that everything in the `/public` folder is public.

Any changes in these three directories are considered "config" changes and not "code" changes.

- **Config changes** use the standard FEF image while the config controls the look and behavior of the frontend just through **deployment with gitops**.
- **Code changes** (for example, anything in the `src` directory) require **forking the commons-frontend-app and building a new image** with the code changes to deploy the custom appearance or behavior

