GEN3 DATA COMMONS

Gen3 Frontend Framework Workshop

Gen3 Community Forum
November 4, 2026

THE UNIVERSITY OF CHICAGO CENTER FOR TRANSLATIONAL DATA SCIENCE

OHSU

OCC

Australian BioCommons

Waipapa Taumata Rau University of Auckland

The Agenda

- **Overview of Gen3 Frontend Framework (FEF)** and comparison to data-portal
- Prepare a **local development environment** for testing FEF configuration changes
- Locally **deploy default config** prepared for Gen3 Data Hub
- Walk through **how to configure FEF for a new data commons** (use Canine DC as the new commons)
  - **Navigation, theme, footer, landing page, login page**
  - Configure **page protection** to control access
- Open Discussion

Future

- Additional future workshop(s) on FEF topics
  - Explorer and Discovery page customization
  - App integration
  - Migration guide to simplify moving from Data Portal to FEF

Overview of FEF and Comparison to data-portal

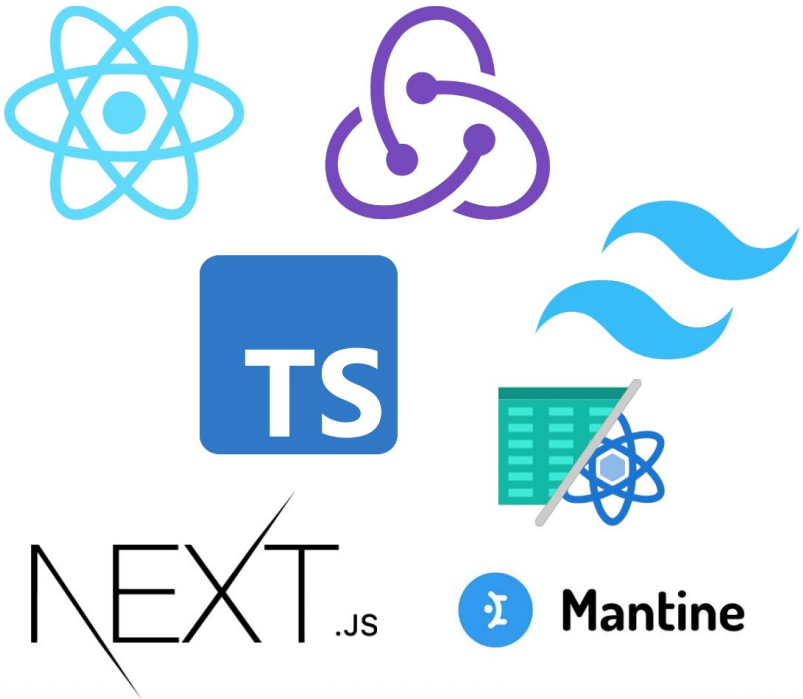
What is the frontend framework (FEF)?

- Replacement for the Data Portal service
  - Open-source React and Next.js toolkit for building Gen3 front ends
- Per-commons codebase which enables a more flexible source code management and deployment
- Improved user, development, and administrative experience
- Upgraded technology stack
- Extensible and customizable

FEF Tech Stack

Core Technology Stack

- Language: **TypeScript**
- UI library: **React 19**
- App framework: **Next.js 16** using the Pages Router
- UI components: **Mantine**
- Styling: **Tailwind CSS**
- **Redux-toolkit/RTK Query** for API calls to Gen3 services



FEF Tech Stack

FEF packages

- **@gen3/core:** contains the core components that are **used to interface with Gen3 services** (MDS, DD, Fence/Arborist, Indexd, etc)
- **@gen3/frontend:** Contains the **React components** (including Gen3 pages) that are used to render the user interface
- **@gen3/toolsff:** build-time CLI tools (icon processing, site path scanning)
- **@gen3/workspaces:** for a new JupyterLite-based workspace UI
- **@gen3/samplecommons:** Next.js development/reference app - **used to test the frontend** and to simplify development
- **@gen3/storybook:** component documentation

Read more about the FEF Architecture here

FEF vs Data Portal (Advantage: FEF)

GEN3 DATA COMMONS

Aspect to be compared	FEF	Data Portal
Technical Currency	Uses **modern web standards and technology**	Uses outdated development patterns
Maintenance / Dependencies	FEF is **more secure and much easier to maintain**. It uses a current version of Node with updated dependencies	
Accessibility	The FEF using a newer library for UI, **Mantine, which is more accessible**.	
Page access controls	FEF gives **more configurable control over access** to each page	Portal limits pages you can control access to (basically just workspace) without revoking full access to platform

FEF vs Data Portal (Advantage: FEF) (con't)

GEN3 DATA COMMONS

Aspect to be compared	FEF	Data Portal
Extensibility	**Yes** - you can **add your own apps** integrated in the FE.	Limited
Customizability	**Quite customizable**	More limited customization
Performance	Re-architected for **improved performance**	Less performant
User and Operator experience	Much better	Occasionally challenging

Re-architected Pages

- Resource browser (→ **Analysis** - available now)
  - Static HTML notebooks, dashboard apps, notebooks, Gen3 apps
- **Data Dictionary** (available now, graph feature coming soon)
  - Better search, better visualization between nodes
- **Discovery** page (available now, advanced config options soon)
  - Much more performant, tables can display links and viz
- **Submission** page (complete overhaul, coming soon)
  - Re-imagining how data ingestion works in gen3

Pages new to the FEF

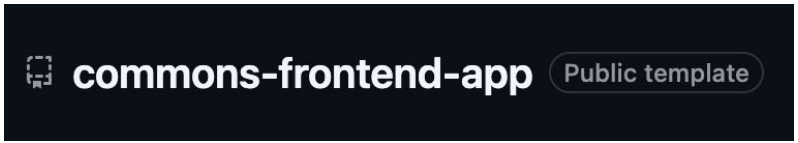
- **Landing page**
  - Highly configurable page to introduce the platform
- **Data Library**
  - Save data sets from Explorer and Discovery
- **Crosswalk**
  - Provide a UI to query and view crosswalk data
- **Workspaces** (note the plural)
  - JupyterLite workspace option
- **FHIRQuery**
  - Brand-new - pending backend support
- **Data Connections**
  - Example of a custom page

Running a “Hybrid” Frontend

Ask us for a guide and we’ll help you manage this

FEF architecture: framework and app





gen3-frontend-framework + commons-frontend-app

The framework (the reusable code) and the deployable app (what runs in your commons)

- **gen3-frontend-framework** is the framework - it's a monorepo that publishes the FEF libraries and build tools.
  - Designed for developing new features and bug fixes in FEF that disseminate more easily

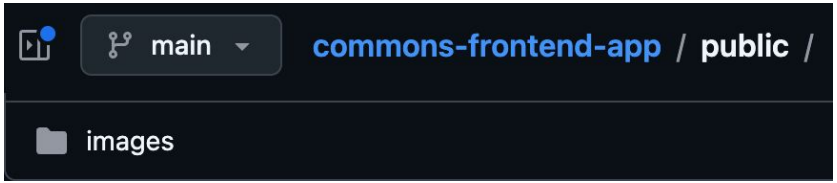
- **commons-frontend-app** is the deployable Next.js application that delivers the frontend built on those libraries. It's the container image Helm deploys.
  - designed as a **base for operators to configure or fork**.

gen3-frontend-framework: Where to look



- `develop` is the primary branch
- `/docs` has existing documentation about many FEF configuration features.
  - Still a work in progress, so some documentation is behind the code
- No need to clone or edit this repo! *(we welcome PRs to fix things, though)*

commons-frontend-app: Where to look



- `main` is the primary branch
- `/public/images` holds images and logos for your configured frontend.
    - Note that everything in the `/public` folder is public.

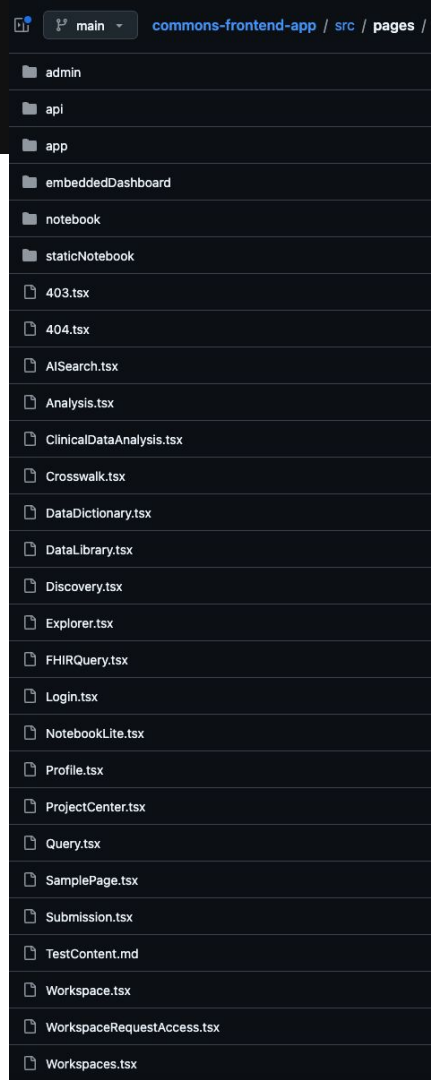


- `/config` holds the configuration files
    - `/config/gen3`: all the per-page JSON files
    - `/config/icons`: icon sets

commons-frontend-app: Where to look (con't)



- `/src/pages` - useful place to see all the possible page names in the config.
- *No configuration here - changes here are code changes*



Config vs. Code changes: The implications of each

Why does it matter?

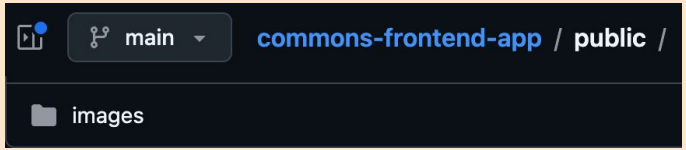
The FEF is designed to easily accommodate changes to the configuration files.

- If the **only changes are in configuration files**:
  - You **do not need to build your own image** to use in your Gen3 instance
  - Just provide the config files.
- If you make **changes elsewhere in commons-frontend-app**:
  - You must **fork the repo** and **build your own image**

Config change or Code change?

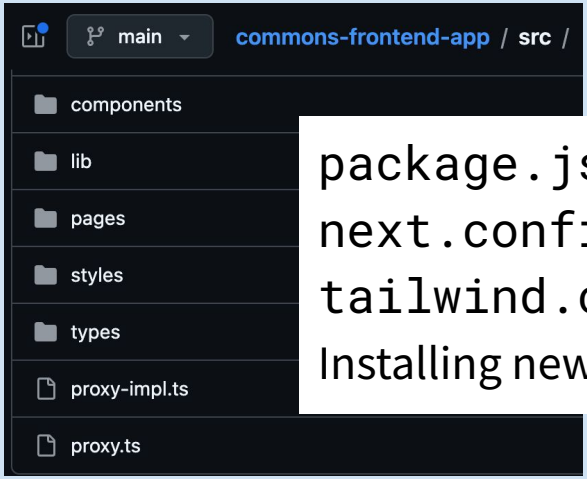
Config change





Gitops can use config files

Code change



`package.json`,
`next.config.js`,
`tailwind.config.js`,
Installing new fonts



Build your own image

Local deployment of default FEF

Local deployment guide also available as a doc for easier reference

Local deployment of default FEF - overview

1. Install `nvm` and `Node.js`
2. Locally-clone `commons-frontend-app` repo, install nvm and Node.js in the local directory, use npm to install dependencies
3. Use `npm run dev` (or `npm run dev:turbopack`) to launch default FEF locally
4. View the deployment from localhost

Install nvm and Node

Install nvm:

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

Then close and reopen the terminal window

Verify nvm is installed and loaded, and verify version:

`command -v nvm` → Look for output `nvm`

`npm -v` → Confirm npm version ≥11.19

Use nvm to install and use Node, and set it to default for new terminals:

`nvm install 24` → installs Node 24.x

`nvm use 24` → switches to it

`node -v` → confirm version ≥24.22

`nvm alias default 24` → make it the default for new terminals

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

Locally clone the commons-frontend-app repo:

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

Change to the commons-frontend-app directory

Install Node and npm in the directory:

```
nvm install
nvm use
```

Use npm to download and set up project dependencies:

```
npm i
```
→ installs Node 24.x

Deploy the default FEF to localhost

Spin up the local Next.js development server with Turbopack

`npm run dev:turbopack` → See note below about Turbopack

When it's spun up, your terminal will show the local URL (eg, `http://localhost:3000`) where you can view the app in your browser.

Notes about Turbopack:

- With Turbopack, as you edit and save code files, the page in your browser will automatically update in real time. It's much faster than `npm run dev`
- There are two features that may not work as expected:
  - **Query page** - will load, but will not work. If you want to USE it, run `npm run dev`
  - **Page protection** requires you also run `npm run build:middlewarePaths` after editing `authz.json`

Craig demos the default FEF deployment

Local development for FEF

Local development guide also available as a doc for easier reference

Local development for FEF - overview

GEN3 DATA COMMONS

1. Create a **local development environment file** pointed to your commons API
2. Create or import **config files** for your commons to be added to your **config/<commons> directory**
3. Create or import **image files** for your commons to be added to your **public/images/<commons> directory**
4. Terminate the old deployment (**ctrl+c**)
5. **Restart deployment** with updated env file using your preferred `npm run dev` command (**`npm run dev:turbopack`**)
6. **View the local deployment** from a local URL, running FEF from local files using remote commons databases
7. **Customize configuration files** in commons-frontend-app repo, viewing effects of changes at the local URL.

Things to remember for local FEF deployment/configuration

`NEXT_PUBLIC_GEN3_COMMONS_NAME` in local env file must match the `<commons>` directory name with the config and image files (i.e., `/config/<commons>` and `public/images/<commons>`)

`NEXT_PUBLIC_GEN3_COMMONS_NAME=`**`canine`**

`/config/`**`canine`**

`/public/images/`**`canine`**

(`/config/icons` does not need a commons folder)

Important Note for moving config to prod:

When you deploy in prod, the **config name needs to be changed back** to `/config/`**`gen3`** and `public/images` (no subdirectory)

Things to remember for local FEF deployment/configuration

Watch for config/backend mismatch

If the config you are starting from is quite different from the backend it is pointing to, the frontend may render very poorly. If you encounter this - update the navigation config first to ensure it includes the pages present in your backend.

Be sure to delete the `NEXT_PUBLIC_GEN3_API` line in local env file

Removing this line avoids accidentally setting this (instead of `NEXT_PUBLIC_GEN3_API_TARGET`) to point at the remote commons

Things to remember for local FEF deployment/configuration

GEN3 DATA COMMONS

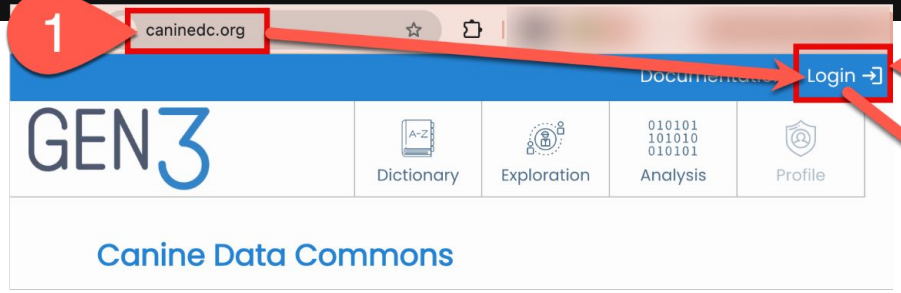
Get your API key from the live site, not the local deployment

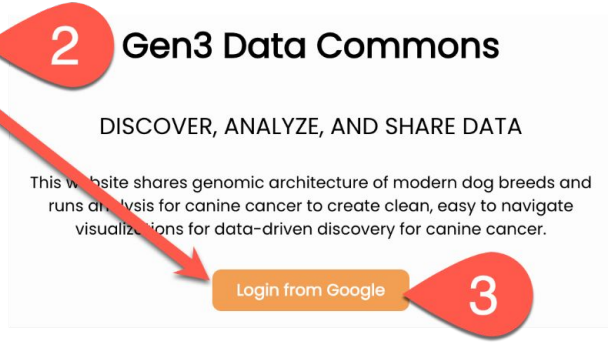
The local deployment will not be able to generate an API key. Download this credential from the live site. You can use this credentials JSON to log in to your local deployment.

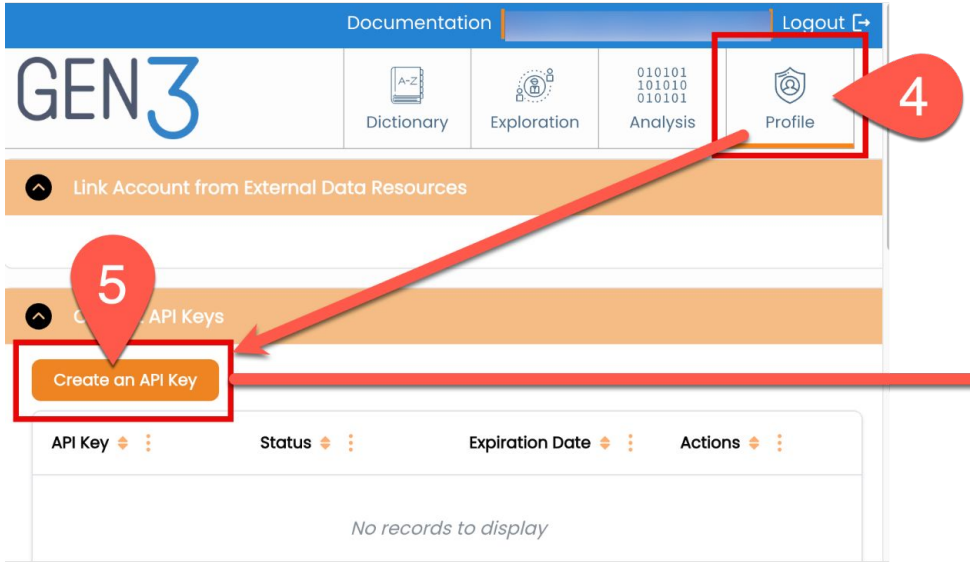
Regular login will not work in the local deployment

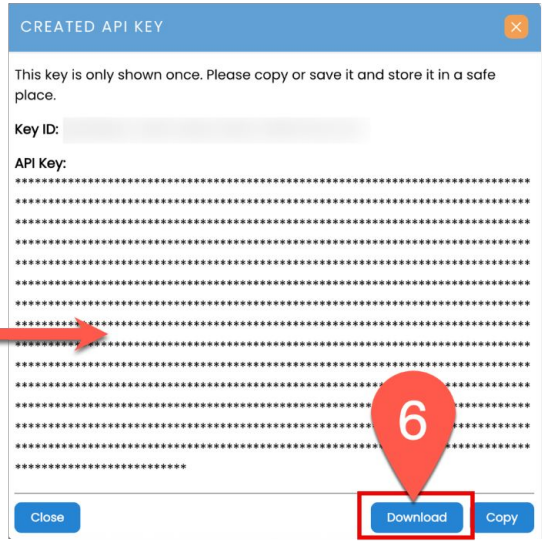
In the login.json, set `"showCredentialsLogin": true` to use the API key file to log in to your local deployment.

Let's get the Canine DC API key now









Create a local dev env file pointed to your commons API

Make a copy of the dev env file with `.local` as a suffix

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

In the local env file, make the following changes to point the local deployment to Canine Data Commons API

- Remove the `NEXT_PUBLIC_GEN3_API` line
- Update these values:
    - `NEXT_PUBLIC_GEN3_COMMONS_NAME=canine`
    - `NEXT_PUBLIC_GEN3_API_TARGET=https://caninedc.org`

Create config files for your commons

Copy the default config directories:

`/config/gen3`

`/public/images/gen3`

Rename the copied directories to have the commons name, and place them next to the appropriate gen3 folder:

`/config/canine`

`/public/images/canine`

View the local deployment with default config and new API

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`**)

View the local deployment from a local URL, running FEF from the local `/canine` files using Canine DC API.

Customize configuration files in /canine directories

Now, we will iterate on configuring the files in `/config/canine` and `/public/images`.

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).

Four exceptions

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

Configuring Navigation

config/canine/navigation.json



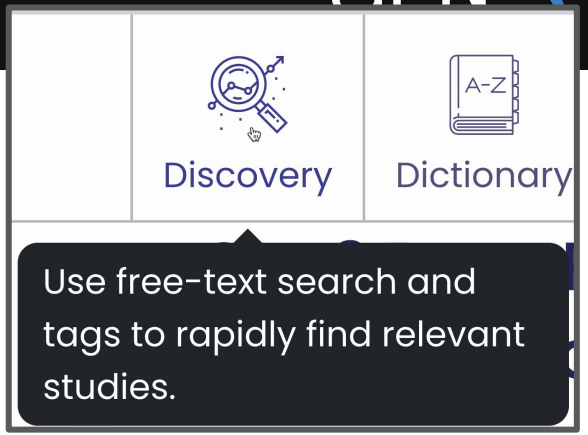
The navigation.json file has 2 main sections:

- `navigation`: Configures the **logo** and **navigation** items.
- `topBar`: Defines the buttons and links located in the **top bar** of the interface.

It also allows you to configure:

- Whether the **login button is visible** and whether **you can see navigation items that you don't have access** to use.
- **Styling** for the navigation panel - we'll cover this in the Themes section

navigation → items

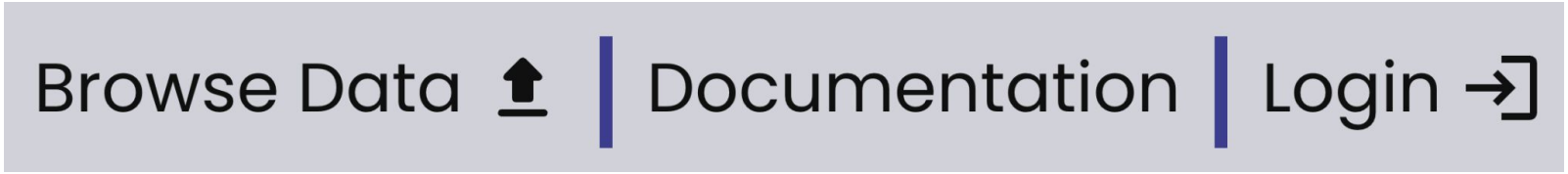


Each **navigation** item includes options for:

- **`icon`**: An icon
- **`href`**: A page route
- **`name`**: A page name
- **`tooltip`**: Tooltip options
- **`enabledWithNoAccess`**: When this is true, a user can still click the button even if they aren't authorized, instead of it being greyed out. This can be used to redirect users to an access request form.

```
"items": [
  {
    "icon": "gen3:query",
    "href": "/Discovery",
    "name": "Discovery",
    "tooltip": {
      "loginRequired": "Login required to access this page",
      "pending": "Checking your access…",
      "unauthorized": "You are logged in but not authorized to access this page",
      "authorized": "Use free-text search and tags to find relevant studies rapidly."
    },
    "enabledWithNoAccess": true
  },
```

topBar



Each **topBar** item includes options for:

- **`rightIcon`**: (optional) An icon displayed to the right of the item text
- **`href`**: The URL to which the item links
- **`name`**: The text displayed for the item
- **`newWindow`**: Whether the link should open in a new window/tab

```
"topBar": {
  "items": [
    {
      "rightIcon": "gen3:upload",
      "href": "/Submission",
      "name": "Browse Data"
    },
    {
      "href": "https://docs.gen3.org/",
      "name": "Documentation"
    }
  ],
  "loginButtonVisibility": "visible"
}
```

logo

```
"logo": {
  "src": "/icons/gen3-dark.svg",
  "width": 120,
  "height": 60,
  "title": "Gen3 Data Hub",
  "divider": true,
  "description": "logo for Gen3 Data Hub",
  "href": "/"
},
```

Logo includes options for:

- **`src`**: Specifies the path to the logo image file
- **`height`**: Defines the height of the logo in pixels
- **`width`**: Defines the width of the logo in pixels
- **`description`**: sets the alt-text
- **`title`**: a text label to the right of the logo image
- **`divider`**: when true, a thin vertical line appears to the right of the logo, separating it from the title
- **`classNames`**: sets styling overrides for the logo area
- **`href`**: Provides the URL where users should be redirected when they click on the logo



Craig demos navigation configuration

Configuring Theme

What does “theme” encompass?

GEN3 DATA COMMONS

The theme sets the site’s overall look:

- the **color** palette (defined in `colors.json` and generated into `themeColors.json` with `npm run build:colors`)
- the heading and body **fonts** (`themeFonts.json`)

Colors in FEF

Gen3 Color Theme is based on **USWDS theme color tokens**

- role-based color families
  - base, primary, secondary, accent, accentWarm, accentCool, chart, and utility

You **select a primary and secondary color**, and then **generate the palette** comprising these color roles. By using tools to generate the palette, you ensure **contrast sufficient for compliance with accessibility** regulations.

Colors in FEF (con't)



Edit the primary and secondary hex codes in `/config/canine/colors.json` (either leaving the other colors in place or changing them all)

Run `GEN3_COMMONS_NAME=canine npm run build:colors`.

This regenerates `themeColors.json`, which is what the FEF uses to control color selections in the portal and can be selected in overrides.

Get hex codes here https://www.color-hex.com/

Colors in FEF - /config/canine/colors.json

GEN3 DATA COMMONS

You can view the colors in themeColors.json by using the /admin/Colors endpoint (e.g., http://localhost:3000/admin/Colors)

```
{
  "primary": "#2C2C54",
  "secondary": "#AAABB8",
  "accent": "#474787",
  "base": "#ECECEC",
  "accentWarm": "#892115",
  "accentCool": "#255990",
  "chart": "#0d95A1"
}
```



Fonts in FEF - Config-change only

Fonts are set in `config/canine/themeFonts.json`.

For the font change to remain config-only, you must use either use:

- an **installed and imported font** (viewable in src/pages/_app.tsx) or
- a **system font** (e.g., Arial, Helvetica, sans-serif)

Fonts in FEF - Config-change only

- **`heading`**: the font for headings.
- **`content`**: the font for body text.
- **`fontFamily`**: the font for the Mantine components (buttons, inputs, tables, and so on). Set it to the same font for a consistent look.

`heading` and **`content`** take either a CSS-style string (`"Montserrat, sans-serif"`) or an array, and should end with a fallback like `sans-serif`.
To look at fonts that are installed and imported, you can look at `commons-frontend-app/src/pages/_app.tsx`

After updating fonts, restart the dev server for the new fonts to be implemented:

Ctrl+C, `npm run dev:turbopack`

```
{
  "heading": ["Poppins", "sans-serif"],
  "content": ["Poppins", "sans-serif"],
  "fontFamily": "Poppins"
}
```

Fonts in FEF - Code-Change

To install a new font, fork the commons-frontend-app repo (because this is a code change).

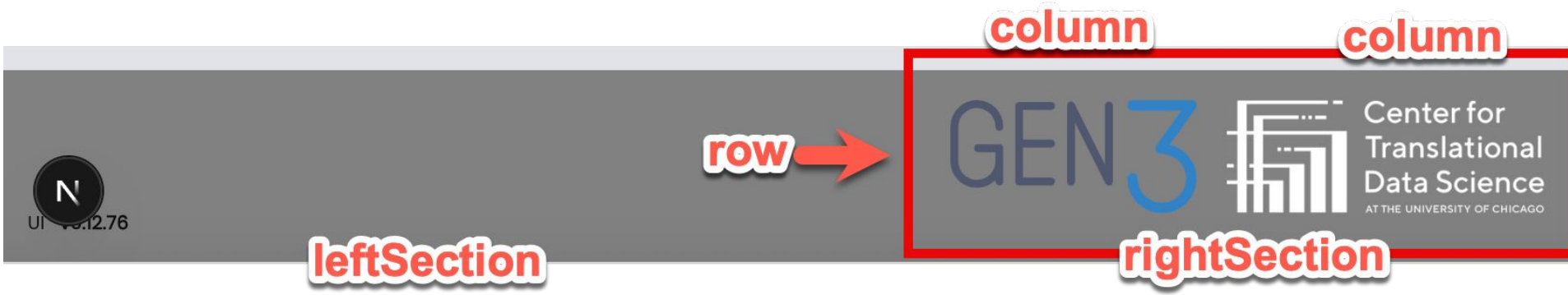
1. Install the font package: `npm i @fontsource/<font>`.
2. Import it in `src/pages/_app.tsx`, alongside the existing font imports.
3. Reference it in `themeFonts.json`.

To deploy this font change, you need your own image built from a fork of commons-frontend-app.

Craig demos theme configuration

Configure the Footer

config/canine/footer.json



The footer.json file has a left and right section

- Each section is a set of columns.
- Each column is a list of rows, and can have a heading

config/canine/footer.json

```
"rightSection": {
  "columns": [
    {
      "rows": [
        {
          "Icon": {
            "logo": "/icons/gen3-dark.svg",
            "width": 132,
            "height": 60,
            "description": "Gen3",
            "href": "https://gen3.org"
          }
        }
      ]
    },
    {
      "rows": [
        {
          "Icon": {
            "logo": "/icons/createdby.png",
            "width": 170,
            "height": 60,
            "description": "Center for Translational Dat
            "href": "https://ctds.uchicago.edu/"
          }
```



Each row is an object with one key that indicates what the row contains

- `Icon`: a logo image (can be linked)
- `Label`: Plain text
- `Link`: One text link
- `Links`: Several text links in a group
- `Section`: A nested set of columns

Put images in `/public`

Craig demos footer configuration

Configuring the Landing page

config/canine/landingPage.json

The landing page is the site's default page.

The homepage configuration is in the body section of the landingPage.json. Each element represents a homepage section or component.

`title`: Title area at the top
`splitarea`: Divides content into left and right sections
`break`: Adds spacing or a separation element.
`cardsArea`: Defines a section for multiple clickable cards.
`quoteArea`: Displays a quote or highlighted text.



```
"body": [
  {
    "title": {
      "text": "Gen3 Data Platform Dynamic Configuratio
      "level": 2
    }
  },
  {
    "splitarea": {
      "left": [
        {
          "text": "The <strong>Gen3 Data Platform</str
        },
        {
          "link": {
            "href": "/discovery",
            "linkType": "portal",
            "text": "Explore Gen3 Data"
          }
        },
        {
          "text": "The Gen3 platform consists of open-
        },
        {
          "link": {
            "href": "https://gen3.datacommons.io",
            "text": "Learn More"
          }
        }
      ],
      "right": [
        {
          "image": {
            "src": "https://gen3.org/figs/kv.svg",
            "alt": "Gen3 Data Platform"
          }
        }
      ]
    }
  },
```

Browse Data | Documentation | Login

GEN3 Gen3 Data Hub

Discovery | Dictionary | Exploration | Query | Analysis | Workspace | Workspaces | Profile | Data Library

Gen3 Data Platform Dynamic Configuration

title

The **Gen3 Data Platform** is a single web interface which allows visitors to **discover, access and analyze data**. Making data easily findable enables secondary, cross-study analyses, promotes dissemination of research and accelerates new discoveries.

EXPLORE GEN3 DATA

The Gen3 platform consists of open-source software services that support the emergence of healthy ecosystems by enabling the interoperation and creation of cloud-based data resources, including data commons and analysis workspaces. Gen3 aims to accelerate and democratize the process of scientific discovery by making it easy to manage, analyze, harmonize, and share large and complex datasets in the cloud.

LEARN MORE

splitarea



splitarea —> left

splitarea —> right

Search studies and related datasets for download or analysis in a workspace.

DISCOVER

Explore Tutorials and Example Analysis.

ANALYZE

cardsArea

quoteArea

Our vision is a world in which researchers have ready access to the data needed and the tools required to make data-driven discoveries that increase our scientific knowledge and improve the quality of life.

GEN3 Center for Translational Data Science

```
      "right": [
        {
          "image": {
            "src": "https://gen3.org/figs/kv.svg",
            "alt": "Gen3 Data Platform"
          }
        }
      ]
    }
  },
  {
    "break": "standard"
  },
  {
    "cardsArea": {
      "title": "",
      "cards": [
        {
          "icon": "MdOutlineSearch",
          "bodyText": "Search studies and related datasets for dow
          "btnText": "Discover",
          "href": "/Discovery",
          "linkType": "portal"
        },
        {
          "icon": "MdOutlineBarChart",
          "bodyText": "Explore Tutorials and Example Analysis.",
          "btnText": "Analyze",
          "href": "/resource-browser",
          "linkType": "portal"
        }
      ]
    }
  },
  {
    "quoteArea": {
      "quote": "Our vision is a world in which researchers have re
    }
  }
]
```


Gen3 Data Platform Dynamic Configuration

title

The **Gen3 Data Platform** is a single web interface which allows visitors to **discover, access and analyze data**. Making data easily findable enables secondary, cross-study analyses, promotes dissemination of research and accelerates new discoveries.

EXPLORE GEN3 DATA

The Gen3 platform consists of open-source software services that support the emergence of healthy ecosystems by enabling the interoperation and creation of cloud-based data resources, including data commons and analysis workspaces. Gen3 aims to accelerate and democratize the process of scientific discovery by making it easy to manage, analyze, harmonize, and share large and complex datasets in the cloud.

splitarea



LEARN MORE

splitarea —> left

splitarea —> right

Search studies and related datasets for download or analysis in a workspace.

DISCOVER

Explore Tutorials and Example Analysis.

ANALYZE

cardsArea

quoteArea

Our vision is a world in which researchers have ready access to the data needed and the tools required to make data-driven discoveries that increase our scientific knowledge and improve the quality of life.

No landing page (use another portal page as default)

If you want another page to be the default landing page, change the `redirectPage` property in the `landingPage.json` file in the config directory.
For example, set `redirectPage` to `"/Discovery"` to redirect to the Discovery page. If you do this, you do not need anything in the landingPage.json config file.

Example (this would be the entire `landingPage.json`):

```
{
  "redirectPage": "/Discovery"
}
```

Custom Landing Page (code change)

If the layout in the template does not meet the project design or requirements, **you can create a custom home page**.

However, this is a **code change**, not just a config change - so you would need to fork the commons-frontend-app and build your own image.

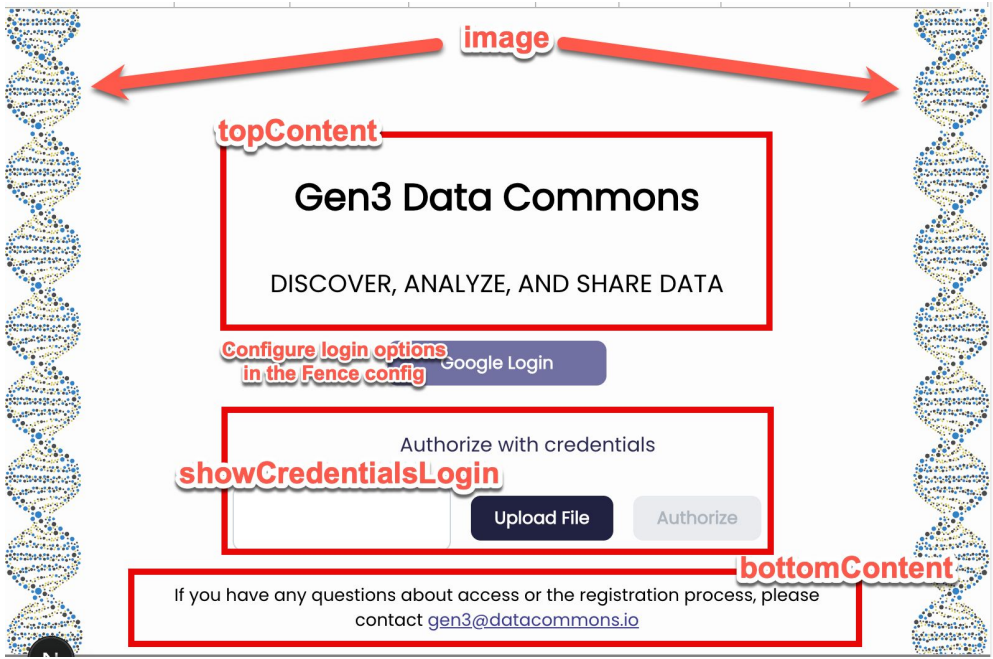
Craig demos landing page configuration

Configuring the Login page

config/canine/login.json

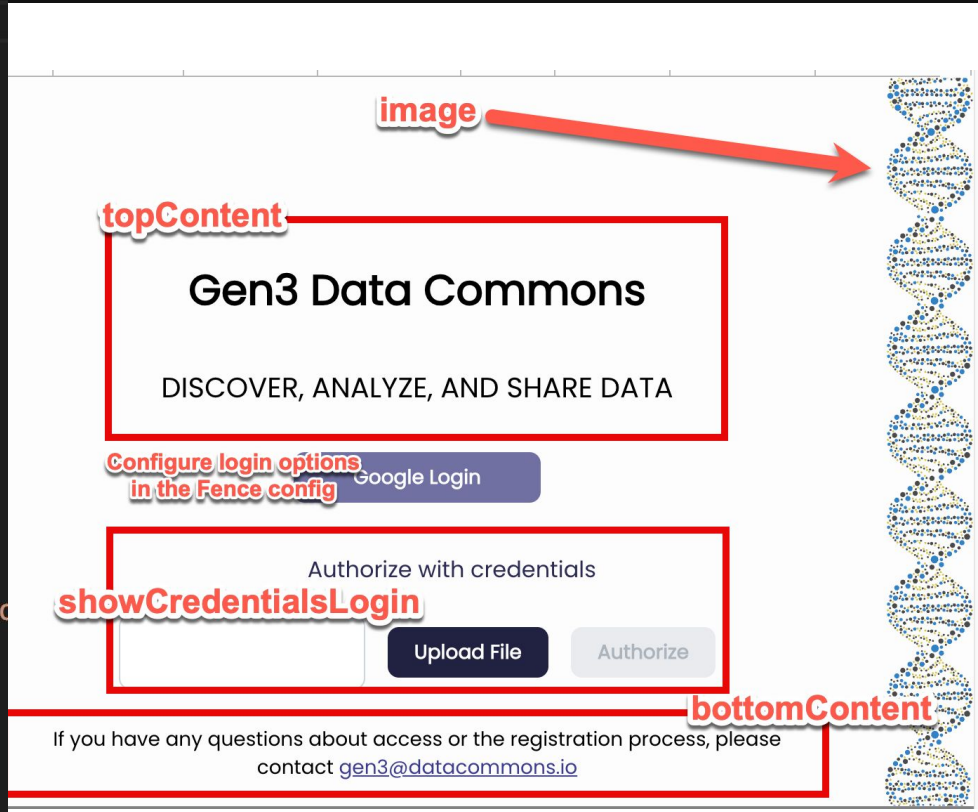
The login configuration has:

- **`topContent`**: this is the content above the login buttons
- **`bottomContent`**: this is at the bottom, below all logins
- **`image`**: this controls the image at the sides



config/canine/login.json

```
{
  "topContent": [
    {
      "text": "Gen3 Data Commons",
      "className": "text-center text-3xl font-bold"
    },
    {
      "text": "DISCOVER, ANALYZE, AND SHARE DATA",
      "className": "text-center text-xl font-medium"
    }
  ],
  "bottomContent": [
    {
      "type": "markdown",
      "text": "If you have any questions about access c
      "className": "text-center text-sm"
    }
  ],
  "image": "images/gene_side_texture.svg",
  "showCredentialsLogin" : true
}
```

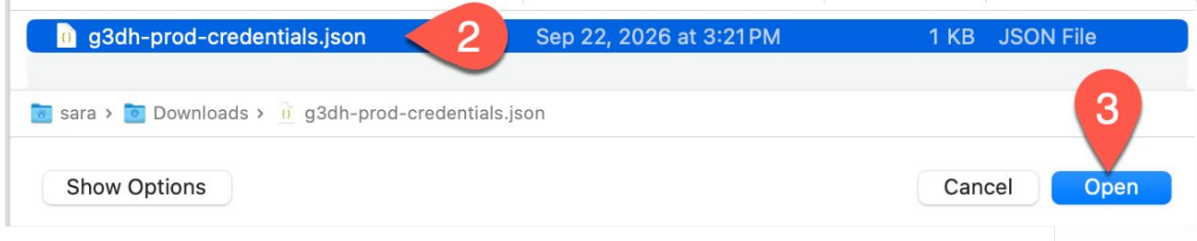


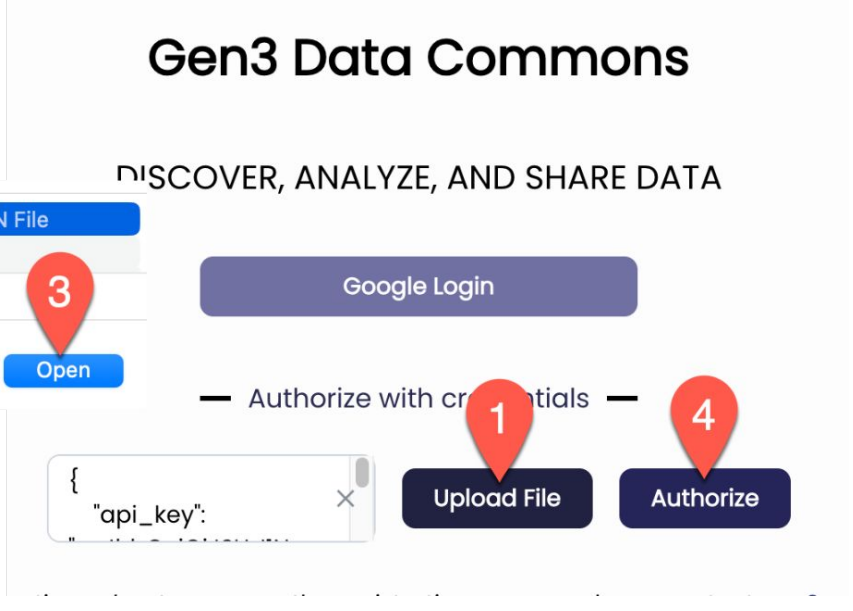
Log in with credentials in local dev

Another configuration on login.json includes:

- **`showCredentialsLogin`**: Adds the “log in with a credentials file” option, used for local development.

Here’s how to log in with a credentials file in local dev





config/canine/login.json

Other configurations on login.json include:

- **`loginProviderExtra`**: Extra text under a specific login option. Keys must exactly match the provider names Fence returns (e.g. "InCommon Login")
- **`loginBtnHorizontal`**: `true` lays out the login buttons side by side; the default is stacked

```
"loginProviderExtra": {
  "InCommon Login": [
    {
      "text": "If your organization is part of the InCommon Federation, ",
      "className": "text-center text-sm"
    },
    {
      "text": "users can access the portal by selecting their institution from the provider
      "className": "text-center text-sm"
    }
  ]
},
"bottomContent": [
```

Craig demos login page configuration

Page Access Control:
AuthN and AuthZ for FEF pages

Access control available for ANY page route!

FEF enables **login restrictions to be applied to any route/page** through the `/config/<commons>/authz.json` file.

- data-portal had no file-based route protection, and only allowed page protection for the Workspace page.

By default, **these pages are protected to require login**:

- Profile
- Data Library
- Workspace-related pages

All** the other pages default to not require login for access

***note: Login and error pages can never be protected through authz.json*

```
{
  "routes": {
    "/DataLibrary": {
      "loginRequired": true
    },
    "/Workspace": {
      "loginRequired": true
    },
    "/Workspaces": {
      "loginRequired": true
    },
    "/WorkspaceRequestAccess": {
      "loginRequired": true
    },
    "/Profile": {
      "loginRequired": true
    },
    "*": {
      "loginRequired": false
    }
  }
}
```

/config/canine/authz.json

GEN3 DATA COMMONS

Access control options

- No limit (open even if not logged in) - `loginRequired: false`
- Require login - `loginRequired: true`
- Require user to have a specific resource in their permissions - `authz: [“/<resource specified in the user yaml>”]`

Things to remember when using page protection

GEN3 DATA COMMONS

You must have the **resources controlling access to the page(s) in the user.yaml**, and you must have a way of **assigning those resources to the users** you want to have access. (Usually, this is through creating and granting policies.)

In local dev, `npm run dev:turbopack` will not update page protection. Instead, first run **`npm run build:middlewarePaths`**, **THEN** **`Ctrl+C`**, **`npm run dev:turbopack`**

Example Authz page protection config

```
{
  "routes": {
    "/Explorer": {
      "loginRequired": true,
      "authz": ["/pages/see-explorer"]
    },
    "/Workspace": {
      "loginRequired": true,
      "authz": ["/pages/see-workspace"]
    },
    "/Profile": {
      "loginRequired": true
    },
    "*": {
      "loginRequired": false
    }
  }
}
```

`config/canine/authz.json`

We want to **lock down the Explorer page and Workspace page** so that only people we **explicitly grant access** to can click those buttons.

We want the **Profile** page to be restricted to anyone **logged in**

We want all the other pages to be **open** to all.

Example useryaml updates to accompany page protxn

GEN3 DATA COMMONS

```yaml
roles:
# role to enable FEF page controls
  - id: page-user
    permissions:
    - id: user
      action:
        service: frontend
        method: access

resources:
# page access resources - for controlling who can see pages in configs
  - name: pages
    subresources:
    - name: see-workspace
      description: allows you to see the Workspace page (does not grant
    - name: see-explorer
      description: grants access to the Explorer page

policies:
# policies controlling access to pages
  - id: see-workspace-page
    description: grants access to the Workspace page
    resource_paths: [/pages/see-workspace]
    role_ids: [page-user]
  - id: see-explorer-page
    description: Grants access to the Explorer page
    resource_paths: [/pages/see-explorer]
    role_ids: [page-user]
```

Creating the page access policies

```yaml
authz:

  groups:
  # assigning policies to users by group
    # Can use workspaces
    - name: workspace_users
      policies:
        - workspace
        - see-workspace-page
      users:
        - workspace_user@univ.edu

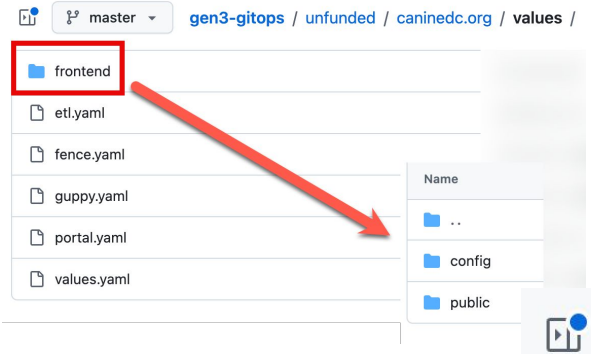
    - name: explorer_users
      policies:
        - see-workspace-page
      users:
        - explorer_user@univ.edu
```

Granting the page access policies

Craig demos page protection configuration

Deploying FEF configs in Helm

Config only? Add your config files to your values dir



master gen3-gitops / unfunded / caninedc.org / values / values.yaml

```
232   frontend-framework:
233     replicaCount: 4
234     enabled: true
240     customConfig:
241       enabled: true
242       repo: https://github.com/uc-cdis/gen3-gitops.git
243       branch: master
244       dir: unfunded/caninedc.org/values/frontend
245     image:
246       repository: quay.io/cdis/commons-frontend-app
247       tag: main
```

Code changes? Point to your commons fork image

master gen3-gitops / vadcprod / vpodc.data-commons.org / values / values.yaml

```
192   portal:
193     enabled: false
194   frontend-framework:
195     replicaCount: 2
196     enabled: true
197     image:
198       repository: quay.io/cdis/vpodc-data-commons
199       tag: "0.29"
200   global:
```



Questions?