

Alphasense Home Task

Liu Jle

Summarization

Live Demo: <https://alphasense.vercel.app/>

Web Component:

NPM Package: <https://www.npmjs.com/package/@liujie2017/weather-widget>

Jsdelivr CDN: <https://www.jsdelivr.com/package/npm/@liujie2017/weather-widget>

Web Server:

GraphQL Sandbox: <https://alphasense-weather-app-stxq8.ondigitalocean.app/>

GraphQL Weather Backend

This project features a sophisticated GraphQL API designed to retrieve precise weather information based on geographical coordinates (latitude and longitude). It seamlessly integrates with an external weather data service, enhancing reliability through sophisticated caching, retry mechanisms, and comprehensive error handling strategies.

Key Features

- **Dynamic Weather Data Retrieval:** Access real-time weather data by specifying latitude and longitude coordinates.
- **Robust External API Integration:** Incorporates an external weather service with built-in caching to optimize response times and minimize unnecessary external API calls.

- **Advanced Reliability Measures:** Employs retry logic and error handling to ensure consistent and dependable data access, even in the face of intermittent service interruptions.
- **Dockerized Deployment:** Simplifies the setup, development, and deployment process via Docker, enabling consistent environments across development and production stages.
- **Comprehensive Testing Suite:** Ensures the reliability and integrity of main functionalities through a combination of unit and integration tests, rigorously verifying each aspect of the server's operations.
- **Production-Ready:** The server is not just a concept; it has been deployed to a Production environment, providing enhanced visibility and real-world testing to ensure it meets the demands of live applications.

Getting Started

Prerequisites

- Docker
- Node.js (for local development)

Installation

Clone the Repository

```
JavaScript
git clone https://github.com/jieliu218/alphasense.git
cd weather_backend
```

```
Unset
// .env: create a .env file under root.
```

```
OPEN_METEO_BASE_URL = "https://api.open-meteo.com/v1/"
```

Start the Server

Local

```
JavaScript  
npm i  
npm run dev
```

...

🚀 Server listening at: <http://localhost:4000/>











Docker

Run it locally with Docker

```
Unset  
docker-compose build  
docker-compose up
```

...

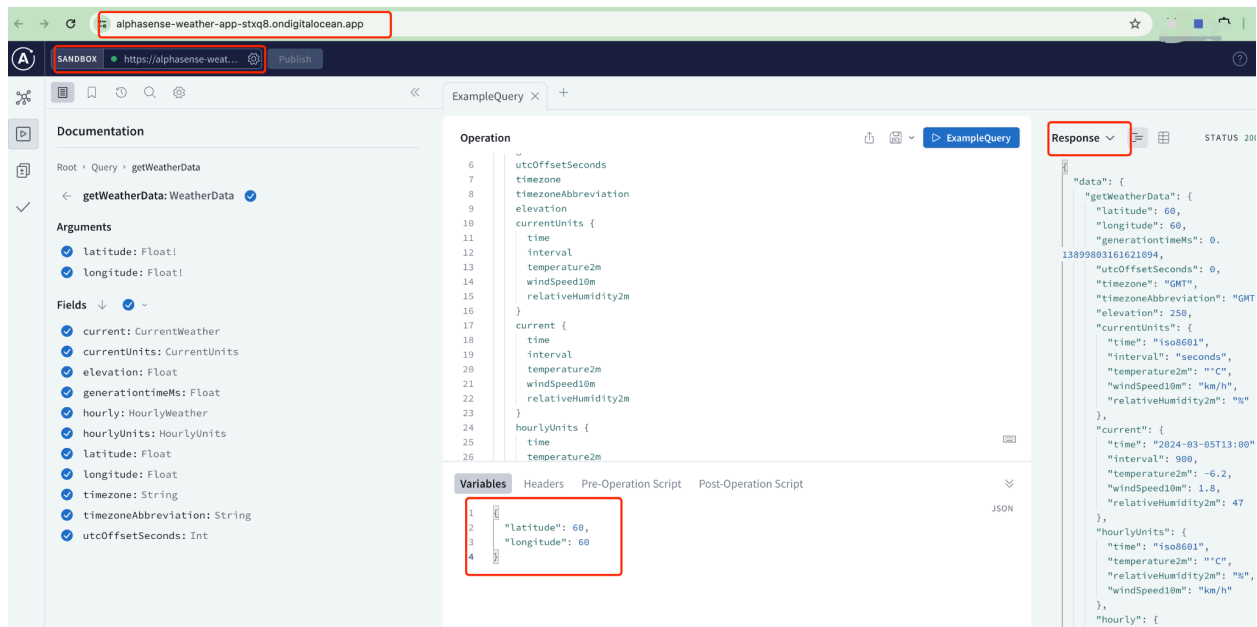
🚀 Server listening at: <http://localhost:4000/>

<input type="checkbox"/>		weather_backend	Running (1/1)	20 hours ago	0%				
<input type="checkbox"/>		weather_backend_backend_1 738c6ea5e48a 	weather_backend_backe Running	4000:4000 	20 hours ago	0%			

Production

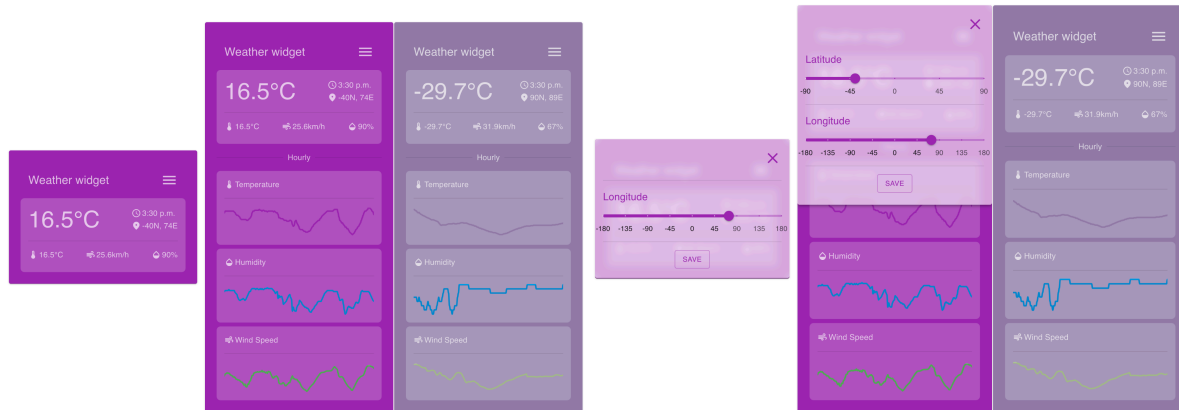
It has been deployed to a Production environment. Visit Apollo Sandbox with URL:

<https://alphasense-weather-app-stxq8.ondigitalocean.app/>



Weather Widget (Front-end)

The Weather Widget is a dynamic, React-based web component that leverages GraphQL to fetch weather data for different locations. It's designed to be flexible, allowing integration into any front-end framework.



Key features

- **GraphQL Data Fetching:** Utilizes @tanstack/react-query and graphql-request for efficient data fetching with GraphQL. The data fetch interval is set to every 900 seconds, ensuring up-to-date weather information.
- **Caching:** Implements caching mechanisms to reduce network requests and load data quickly, enhancing the user experience.
- **Framework Agnostic:** As a React-based web component, it can be seamlessly integrated into any framework-based front-end, providing versatility in development.
- **Theme Support:** Adopts the same theme definition as MUI (Material-UI), allowing for easy customization and consistency across your application.
- **Location Customization:** Users can specify latitude and longitude to fetch weather data for different locations, offering global weather information at your fingertips.
- **Visibility Control:** Features attributes to hide and show hourly weather data, giving users control over the information displayed.
- **Unit Testing:** Comes with main unit tests added, ensuring reliability and stability of the widget.
- **CDN and NPM Support:** The Weather Widget has been deployed to npm for easy installation and CDN via jsFiddle for quick integration.

Installation

The Weather Widget has been deployed to npm for easy installation and CDN via jsFiddle for quick integration. Here are two ways to use it:

Via npm

To install the Weather Widget via npm, run the following command in your project directory:

```
JavaScript
npm i @liujie2017/weather-widget@latest
```

Then, you can import and use the widget in your React application:

```
JavaScript
import { green, purple } from "@mui/material/colors";
import "@liujie2017/weather-widget"

import "./App.css";

declare global {
  namespace JSX {
    interface IntrinsicElements {
      ["weather-widget"]: any;
    }
  }
}

function App() {
  const themePurple = {
    palette: {
      primary: {
        main: purple[500],
      },
    },
  },
```

```

        secondary: {
          main: green[500],
        },
        warning: {
          main: purple[500],
          light: purple[100],
        },
      },
    };
    return (
      <>
      <weather-widget latitude={-40} longitude={74} theme={
        theme={JSON.stringify(themePurple)}
      }></weather-widget>
      </>
    );
  }

  export default App;

```

Via CDN (jsFiddle)

For quick prototyping or integration without npm, you can use the Weather Widget directly in your HTML through a CDN provided by jsFiddle. Include the following script tag in your HTML file:

```

JavaScript
<script
src="https://cdn.jsdelivr.net/npm/@liujie2017/weather-widget@0.0.1/dist/index.e
s.min.js"></script>

```

After including the script, you can use the widget in your HTML as follows:

JavaScript

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
    <weather-widget
      latitude="90"
      longitude="89"
      theme='{
        "palette":{
          "primary":{
            "main":"#917ca9"
          },
          "secondary":{
            "main":"#a4bc8a"
          },
          "warning":{
            "main":"#917ca9",
            "light":"#bcaacf"
          }
        }
      }'
    ></weather-widget>
    <script type="module"
src="https://cdn.jsdelivr.net/npm/@liujie2017/weather-widget@0.0.1/dist/index.e
s.min.js"></script>
  </body>
</html>
```

Customization

The Weather Widget supports several props for customization:

- `latitude` and `longitude`: Specify the location to fetch weather data for.
- `showHourly`: Boolean attribute to control the visibility of hourly weather data.

- `theme`: Customize the widget's appearance using MUI theme options.

Local dev

```
JavaScript  
npm i  
npm run dev
```

Weather App

The Weather App is a demonstration application showcasing the versatile usage of the weather-widget in a Next.js environment. It aims to illustrate how easily the widget can be integrated into modern web applications to provide real-time weather information.

Live Demo

The application is live at <https://alphasense.vercel.app/>. Feel free to visit the site and explore the different ways weather-widget can be utilized within a web application.

Features

The Weather App demonstrates four key use cases of the weather-widget:

1. Default Usage

Shows how to use the weather-widget in its simplest form without any customization. Ideal for developers who want to quickly integrate weather functionality with minimal setup.

2. Custom Theme

Illustrates how to apply a custom theme to the weather-widget to align with your application's branding or color scheme. This use case is perfect for maintaining consistent visual aesthetics throughout your app.

3. Custom Attributes

Demonstrates the flexibility of the weather-widget by showcasing how to use custom attributes to alter its behavior and presentation. This use case provides insights into the widget's configurability to meet specific requirements.

4. HTML and CDN Script

Explores how the weather-widget can be integrated into a static HTML page using a CDN script. This use case is particularly useful for developers working on projects without complex build processes or those looking to add weather functionality to existing static sites.