📄 PoC Documentation: Multi-Client React App Deployment using AWS S3 + CloudFront + Subdomain Routing

Objective

To create a single dynamic ReactJS build that can serve different content for multiple clients (Example: HHM , Stepstone) using subdomain-based routing via CloudFront CDN and AWS S3.

PoC Architecture

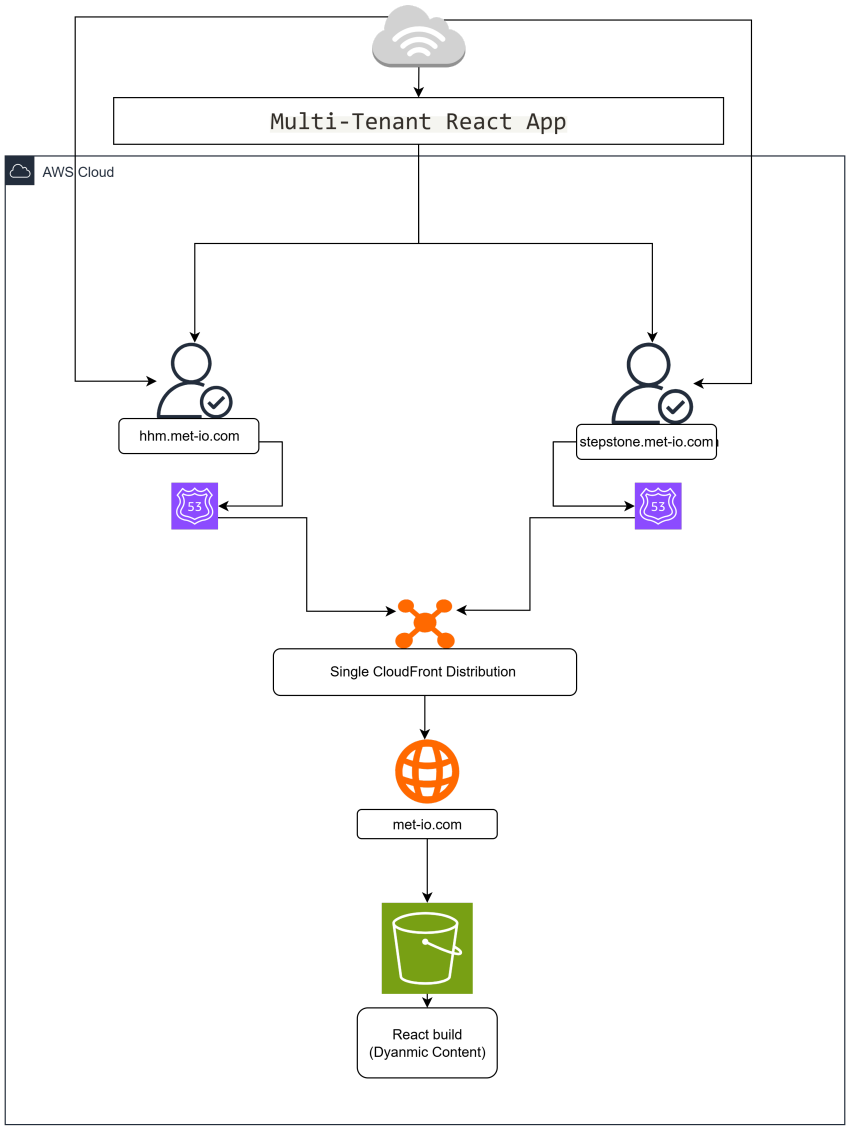


fig show : multi-tenant react-app architecture

Tech stack used

- **ReactJS**
- **Ec2 instance (for create a sample react-app)**
- **Git hub (for store the code)**
- **AWS S3 (for hosting build)**
- **AWS CloudFront (for CDN)**
- **AWS Route 53 (for domain management)**
- **AWS Certificate Manager (SSL for subdomains)**

- **Custom Domain:**

met-io.com

- **Subdomains: hhm.met-io.com , stepstone.met-io.com**

Step-by-Setup Implementation

1. Create the Sample React-app

On the
EC2 instance, Node.js and its dependencies were installed. After that, the following command was executed to create a new React application:

```
npx create-react-app met-react-host
```

After creating the React app, move into the project directory using the following command:

```
cd met-react-host
```

Install Material UI in the React App

To use **Material UI (MUI)** components in your React application, install the required packages by running the following command:

```
npm install @mui/material @emotion/react @emotion/styled
```

This command installs:

- `@mui/material`: Core Material UI component library.
- `@emotion/react` and `@emotion/styled`: Styling libraries required by MUI for theming and CSS-in-JS.

After installation, you'll be able to import and use MUI components in your React project.

Modify React App Source Files

After creating the React app and installing Material UI, you proceeded to edit source files inside the `src/` folder.

Navigate to the `src/` folder:

`cd src/`

List the files in the `src/` directory:

`ls`

You likely saw files like `App.js`, `index.js`, `App.css`, etc.

Create or edit a custom file (e.g., `client.js`):

`sudo nano client.js`

edit the file for you requirements:

```
// src/client.js
export function getClient() {
const hostname = window.location.hostname;
const subdomain = hostname.split('.')[0];
if (subdomain === 'hhm') return 'hhm';
if (subdomain === 'stepstone') return 'stepstone';
return 'default';
}
```

save the file and exit.

This opens `client.js` in the Nano text editor with superuser permissions. You may be defining an API client or configuration here.

Edit the main app component file:

sudo nano App.js

edit the file for you requirements and call the client functions to redirect properly to the users:

```
import React from "react";
```

```
function getClient() {
const hostname = window.location.hostname;
const subdomain = hostname.split('.')[0];
if (subdomain === "hhm") return "hhm";
if (subdomain === "stepstone") return "stepstone";
return "default";
}
```

```
function App() {
const client = getClient();
```

```
return (
<div className="App" style={{ textAlign: "center", marginTop: 50 }}>
<header className="App-header">
<h1>Welcome to {client.toUpperCase()} Portal</h1>
<p>
This content is dynamically rendered based on the subdomain.
</p>
{client === "hhm" && <p>HHM specific content goes here!</p>}
{client === "stepstone" && <p>Stepstone specific content goes here!</p>}
{client === "default" && <p>This is the default content for other domains.</p>}
</header>
</div>
);
}
```

```
export default App;
```

After integrating the functionality from `client.js` into the React application, the corresponding frontend components were updated to execute the logic properly and map it to the correct users.

Once the integration and mapping were completed, the changes were **saved and exited** from the editor.

`cd ..`
Changes the current directory to the parent directory. You might be stepping back from the
`src/` or similar folder to the root of your project.

`npm run build`
Runs the build script defined in your
`package.json` file. This usually bundles your application (e.g., React or Vue app) into static files for production.

2. Uploading Build Files to S3 for Hosting

After generating the build files, use the following CLI command to upload them to your S3 bucket for hosting

Install AWS CLI:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.z
unzip awscliv2.zip
sudo ./aws/install
aws --version
```

You're downloading, unzipping, installing, and verifying the AWS CLI. All standard.

Configure AWS Credentials:

```
cd met-react-host/
```

Deploy React build to S3:

```
aws s3 sync build/ s3://met-react-host/ --delete
```

Syncs your local
`build/` directory with the S3 bucket `met-react-host`, deleting any extra files in the bucket.

Deployment Checklist

To make sure your React app works properly after being deployed to S3:

1. **S3 Bucket Static Website Hosting:**
   - In the AWS Console, go to your `met-react-host` bucket.
   - Enable **Static Website Hosting**.
   - Set `index.html` as the index document (and optionally `error.html`).
2. **Permissions:**

Your S3 bucket needs to allow public read access (if your site is public).

A common bucket policy:

```
json
CopyEdit
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::met-react-host/*"
    }
```

```
  ]
}
```

Given the permission to the s3 bucket , If Everything Works

You'll see your React app running at:

🌐 http://met-react-host.s3-website.ap-south-1.amazonaws.com

3.Purchasing and Configuring Domain `met-io.com` via AWS Route 53

Your build is working well in the S3 bucket. After uploading the build files to S3, proceed to the next step: Register a domain in Route 53 for your project.

Steps

1. Sign in to AWS Management Console

- Open the AWS Management Console.
- Navigate to **Route 53** service

2. Search for Domain `met-io.com`

- Click on **"Registered domains"** in the left sidebar.
- Click **"Register Domain"**.
- Enter the domain name `met-io.com` in the search box.
- Check if the domain is available for registration.

3. Register the Domain

- If available, click **"Add to cart"** and then **"Continue"**.
- Fill in your contact details (Registrant, Administrative, Technical, Billing).
- Review the domain registration fees and terms.
- Complete the purchase by following on-screen instructions.

4. Request SSL Certificate in AWS Certificate Manager (ACM)

- Requested a **public SSL certificate** in the **us-east-1** region for the subdomains:
    1. `hhm.met-io.com`
    2. `stepstone.met-io.com`
- Validated domain ownership via DNS.

5. Associate SSL Certificate with Route 53 Domain

- After certificate issuance, associated the certificate with the **Route 53 hosted zone** for
- `met-io.com`

6. Create CloudFront Distribution

Created a **CloudFront distribution** with the following configurations:

Origin:

S3 bucket origin pointing to `met-react-host` bucket with specified **origin path** if applicable.

```
{
"Version": "2008-10-17",
"Id": "PolicyForCloudFrontPrivateContent",
"Statement": [
{
"Sid": "AllowCloudFrontServicePrincipal",
"Effect": "Allow",
"Principal": {
"Service": "cloudfront.amazonaws.com"
},
"Action": "s3:GetObject",
"Resource": "arn:aws:s3:::met-react-host/*",
"Condition": {
"StringEquals": {
"AWS:SourceArn": "arn:aws:cloudfront::346367215290:distribution/E3W0RGI
}
```

```
}
}
]
}
```

Configured Origin Access Control (OAC) policy to restrict S3 bucket access to CloudFront only.

Alternate Domain Names (CNAMEs):

1. Added `hhm.met-io.com`
2. Added `stepstone.met-io.com`

SSL Certificate:

- Associated the ACM certificate requested earlier.

Behavior Settings:

- **Viewer Protocol Policy:** Set to **Redirect HTTP to HTTPS** to enforce secure connections.
- **Purpose:** Automatically redirects all HTTP requests to HTTPS, enforcing secure connections.

7. Configure Route 53 DNS Records

- Created **A (Alias)** records in Route 53 for:
  - `hhm.met-io.com` pointing to the CloudFront distribution.
  - `stepstone.met-io.com` pointing to the CloudFront distribution.

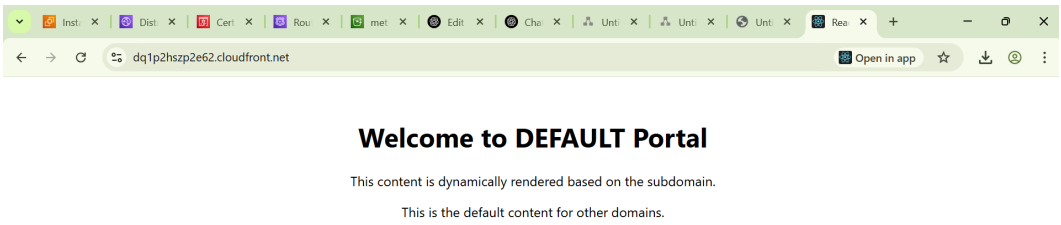
8. Verify Setup

- Accessed:
  - `https://hhm.met-io.com`
  - `https://stepstone.met-io.com`
- Verified the React app is served securely over HTTPS from CloudFront, using the S3 origin bucket.

Final Output

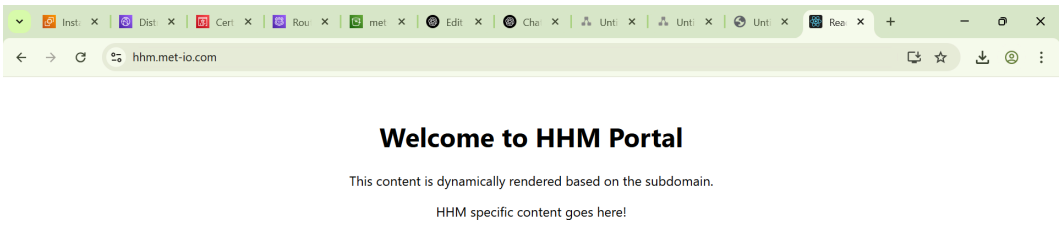
The React application is successfully hosted and accessible through the CloudFront CDN at DNS URL:

https://dq1p2hszp2e62.cloudfront.net



The React application is successfully hosted and accessible through the CloudFront CDN at sub-domain of hhm URL this dynamic content for HHM portal frontend page will appear and it will only access for hhm.met-io.com:

https://hhm.met-io.com/



The React application is successfully hosted and accessible through the CloudFront CDN at Sub-domain of Stepstone URL:

https://stepstone.met-io.com/