Deploying a Vite + React SPA on Azure (Static Web Apps) with a REST API on Azure App Service

This document serves to guide on building a clean, repeatable setup to deploy a **Vite + React** single-page app (SPA) to **Azure Static Web Apps** and a **Node/Express REST API** to **Azure App Service (Web App)**, wired together with **Auth0** for authentication.

It also documents two issues faced during development with the solutions:

1. **GitHub Action workflow failing due to a relative import** (e.g., import "../component/HomePage") and how switching to an **@ alias** fixed it.
2. **Handling keys/secrets for Auth0** in both the SPA and API safely.

1) Architecture Overview

- **Frontend:** Vite + React SPA → **Azure Static Web Apps (SWA)**
    - Built on GitHub Actions and deployed to SWA.
    - SPA routing handled by SWA's routes.json fallback to index.html.
- **Backend:** Node/Express REST API → **Azure App Service (Web App)**
    - Deployed via GitHub Actions to an App Service plan (Linux recommended).
    - CORS enabled for the SPA domain.
- **Auth: Auth0**
    - SPA uses @auth0/auth0-react with domain & client ID.

Browser —> Azure Static Web Apps (React SPA) —(fetch + Auth0)—> Azure App Service (REST API)

2) Prerequisites

- Azure subscription with permission to create:
    - 1 × **Static Web App** (Free tier or any as you prefer)
    - 1 × **App Service** (Linux, Node runtime)
- GitHub repository for the Vite + React app and API (deployed from separate branches)
- We used Node.js 20+ locally and CI (Continuous Integration)
- Auth0 tenant with an Application (SPA) and an API configured

3) Frontend (Vite + React) Setup

3.1 Project structure (example)

```
root/
src/
components/
pages/
main.tsx
App.tsx
index.html
vite.config.ts
tsconfig.json (or jsconfig.json)
package.json
```

3.2 Fixing the Relative Import Build Failure (use an @ alias)

Symptom: GitHub Action build failed on Azure due to a relative import like:

import HomePage from "../component/HomePage" with an error saying the build process can not find the files being imported.

This can fail on case sensitivity, path resolution differences, or brittle deep ../.. paths.

Solution used: Define a stable @ alias to src/ and update imports.

Inside vite.config.ts:

```
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';
import path from 'node:path';

export default defineConfig({
  plugins: [react()],
  resolve: {
    alias: {
      '@': path.resolve(__dirname, 'src'),
    },
  },
});
```

tsconfig.json (or jsconfig.json)

```
{
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@/*": ["src/*"]
    }
  }
}
```

Then update imports

```
// Before
// import HomePage from "../component/HomePage";

// After
import HomePage from "@/components/HomePage";
```

→ After introducing the alias, GitHub Actions could complete the build successfully.

3.3 Environment Variables (Vite)

Vite exposes variables at build time via import.meta.env. Only variables prefixed with VITE_ are injected.

Example .env (local only; do not commit):

```
VITE_AUTH0_DOMAIN=your-tenant.eu.auth0.com
VITE_AUTH0_CLIENT_ID=abc123...
VITE_AUTH0_AUDIENCE=https://your-api-identifier
VITE_API_BASE_URL=https://your-api.azurewebsites.net
```

Use them in code:

```
const domain = import.meta.env.VITE_AUTH0_DOMAIN;
const clientId = import.meta.env.VITE_AUTH0_CLIENT_ID;
const audience = import.meta.env.VITE_AUTH0_AUDIENCE;
const apiBase = import.meta.env.VITE_API_BASE_URL;
```

3.4 Auth0 in the SPA

Install the Auth0 dependency:

npm i @auth0/auth0-react

Wrap your app:

```
import { Auth0Provider } from '@auth0/auth0-react';


<Auth0Provider
  domain={import.meta.env.VITE_AUTH0_DOMAIN}
  clientId={import.meta.env.VITE_AUTH0_CLIENT_ID}
  authorizationParams={{
    redirect_uri: window.location.origin,
    audience: import.meta.env.VITE_AUTH0_AUDIENCE, // to get API
access tokens
  }}
>
  <App />
</Auth0Provider>
```

3.5 SPA Routing (Azure SWA)

Create staticwebapp.config.json at the project root to enable SPA fallback and CORS headers if needed:

```
{
  "navigationFallback": {
    "rewrite": "/index.html",
    "exclude": ["/assets/*", "/*.{png,jpg,gif,svg,css,js}"]
  }
}
```

4) Backend (Azure App Service) Setup

4.1 Example Express API

```
api/
models/
routes/
  package.json
  server.js
```

4.2 App Service Configuration (Environment Settings)

In your App Service → **Configuration** → **Application settings**, add:

- AUTH0_DOMAIN = your-tenant.eu.auth0.com
- AUTH0_AUDIENCE = https://your-api-identifier
- CORS_ORIGIN = https://<your-swa>.azurestaticapps.net

App Service restarts when you save settings.

4.3 CORS

If you use the App Service CORS blade, add your SWA domain. Alternatively, keep it in app code via cors() as above.

5) GitHub Actions – CI/CD

5.1 SPA → Azure Static Web Apps

In Azure Portal, create a **Static Web App** and point it to your GitHub repo. Azure will auto-create a workflow similar to:

.github/workflows/azure-static-web-apps.yml (example)

```
name: Deploy SPA to Azure Static Web Apps
on:
 push:
  branches: [ main ]
jobs:
 build_and_deploy:
  runs-on: ubuntu-latest
  steps:
   - uses: actions/checkout@v4
   - uses: actions/setup-node@v4
    with:
     node-version: 20
   - run: npm ci
   - run: npm run build
    env:
     VITE_AUTH0_DOMAIN: ${{ secrets.VITE_AUTH0_DOMAIN }}
     VITE_AUTH0_CLIENT_ID: ${{ secrets.VITE_AUTH0_CLIENT_ID }}
     VITE_AUTH0_AUDIENCE: ${{ secrets.VITE_AUTH0_AUDIENCE }}
     VITE_API_BASE_URL: ${{ secrets.VITE_API_BASE_URL }}
   - uses: Azure/static-web-apps-deploy@v1
    with:
     azure_static_web_apps_api_token: ${{ secrets.AZURE_STATIC_WEB_APPS_API_TOKEN }}
     app_location: "/"
     output_location: "dist"
```

Important: Because Vite injects env vars *at build time*, define the VITE_* values as **GitHub Secrets** and pass them as environment variables during the npm run build step.

5.2 API → Azure App Service (Web App)

Create a publish profile in App Service → **Overview** → **Get publish profile**, then add it to GitHub Secrets as AZURE_WEBAPP_PUBLISH_PROFILE.

.github/workflows/deploy-api.yml (simplified)

```
name: Deploy API to Azure App Service
on:
 push:
  branches: [ main ]
  paths:
   - 'api/**'
jobs:
 build_and_deploy:
  runs-on: ubuntu-latest
  defaults:
   run:
    working-directory: api
  steps:
   - uses: actions/checkout@v4
   - uses: actions/setup-node@v4
    with:
     node-version: 20
   - run: npm ci --omit=dev
   - uses: azure/webapps-deploy@v3
    with:
     app-name: "your-api-appservice-name"
     publish-profile: ${{ secrets.AZURE_WEBAPP_PUBLISH_PROFILE }}
     package: api
```

App settings (AUTH0_*, CORS_ORIGIN) live in the App Service configuration, not in the workflow.

6) Auth0 Configuration

6.1 Auth0 Application (SPA)

- **Application Type:** Single Page Web Applications
- **Allowed Callback URLs:** When deploying, https://<your-swa>.azurestaticapps.net or refer to development guide for the localhost URL
- **Allowed Logout URLs:** When deploying, https://<your-swa>.azurestaticapps.net or refer to development guide for the localhost URL
- **Allowed Web Origins:** When deploying, https://<your-swa>.azurestaticapps.net or refer to development guide for the localhost URL
- **Allowed Origins (CORS):** When deploying, https://<your-swa>.azurestaticapps.net or refer to development guide for the localhost URL

Populate these values as GitHub Secrets for the SPA build:

- VITE_AUTH0_DOMAIN
- VITE_AUTH0_CLIENT_ID
- VITE_AUTH0_AUDIENCE

6.2 Auth0 API (Machine-to-Machine / Identifier)

- Create an **API** in Auth0 with an **Identifier** (this becomes your audience).
- Enable RBAC and **Add Permissions in the Access Token** if you plan to use user scopes.

7) Handling Keys to Make Auth0 Work (Securely)

Frontend (Vite):

- Put values in **GitHub Secrets** and inject at build time as VITE_* envs.
- Remember: Anything in the SPA **ends up in the client bundle** (domain, clientId, audience are not secrets; they can be public). Do **not** put truly secret keys in the SPA.

Backend (API on App Service):

- Store secrets and configuration in **App Service → Configuration → Application settings** (encrypted at rest). Examples:
    - AUTH0_DOMAIN, AUTH0_AUDIENCE
    - Any additional secret keys (e.g., DB connection strings)
- **Never** check secrets into git.
- If you must reference them in GitHub Actions, use **GitHub Secrets**.

Auth0 Keys Checklist:

- SPA VITE_AUTH0_DOMAIN, VITE_AUTH0_CLIENT_ID, VITE_AUTH0_AUDIENCE in GitHub Secrets
- API AUTH0_DOMAIN, AUTH0_AUDIENCE in App Service settings
- Auth0 **Allowed URLs** updated to the SWA production domain
- CORS: API allows the SWA origin

8) Local Development

Frontend:

npm run dev

API:

cd api

npm run start

Create a .env.local for Vite and a separate .env for the API (never commit). Use a proxy during local development if you want to avoid CORS:

9) Common Pitfalls & Troubleshooting

9.1 GitHub Action Build Failure from Relative Imports

- **Cause:** Deep relative paths or case sensitivity differences break imports.
- **Fix:** Use a Vite alias (@ → src) and update imports; ensure tsconfig.json paths match.

9.2 VITE_* Variables Not Available in the SPA

- **Cause:** Not prefixed with VITE_ or not provided at build time in CI.
- **Fix:** Prefix envs with VITE_ and pass them to the npm run build step.

9.3 401/403 Calling the API

- **Cause:** Token missing/invalid, wrong audience, or CORS blocked.
- **Fix:** Request token with the **correct audience** in the SPA; set CORS origin in API; validate with issuer and audience.

9.4 SPA 404 on Refresh / Deep Links

- **Cause:** Static host tries to resolve route as a file.
- **Fix:** staticwebapp.config.json with navigationFallback to /index.html.

9.5 Mixed Content / HTTPS

- **Cause:** API served over http while SPA is https.
- **Fix:** Use https for API (App Service with default TLS) and reference that URL in VITE_API_BASE_URL.

9.6 GitHub Actions Secrets Not Found

- **Cause:** Secrets missing or workflow from a fork PR.
- **Fix:** Define secrets in the target repo; note that secrets don't pass to workflows triggered from forks unless explicitly allowed.

10) Verification Checklist (Production)

- **SWA URL** responds with built SPA
- **API URL** responds on /public
- **Protected route** /private returns 401 without token, 200 with valid token
- SPA login redirects to Auth0, returns to SWA, and fetches protected data with bearer token

- CORS preflight and requests succeed from SPA to API

11) Appendices

11.1 Minimal staticwebapp.config.json

```
{
  "navigationFallback": { "rewrite": "/index.html" }
}
```

11.2 Example .gitignore

```
# Node
node_modules

# Vite build
/dist

# Env files
.env*
```

11.3 Useful Azure CLI (optional)

```
# Login
az login

# Show SWA
az staticwebapp list -o table

# Show App Service settings
az webapp config appsettings list \
  --name your-api-appservice-name \
  --resource-group your-rg -o table
```

12) Summary of the Two Documented Issues that one of the dev team members faced

1. **GitHub Action build failed with relative path imports (e.g., import "../component/HomePage").**
   Resolution: Added @ alias to src/ via vite.config.ts and tsconfig.json, updated imports to @/components/…. Build then completed successfully.

2. **Handling keys for Auth0.**
   Resolution: For the SPA, placed non-secret config in GitHub Secrets as VITE_* and injected at build time. For the API, stored Auth0 config in **App Service → Configuration** (server-side)… CORS allows only the SWA origin.