Step-by-step guide to deploy React app to Azure Static Web Apps using Azure Portal

1) Quick overview — what you'll do

1. Create a React app (or use your existing one) locally.
2. Push the code to a GitHub repo.
3. In the Azure Portal create a **Static Web App** and connect it to that GitHub repo/branch. Azure will add a GitHub Actions workflow that builds & deploys your app.
4. Fix client-side routing (single page app) with `staticwebapp.config.json` if needed.
5. Visit the generated URL (and optionally add a custom domain).

2) Prerequisites

- GitHub account and a GitHub repository.
- Azure subscription (free tier is fine) and access to the Azure Portal.
- Node.js and npm installed locally.
- Basic Git knowledge (init, commit, push).
- Your React app source (or create one following the steps below).

3) Create a React app (if you don’t have one yet)

```
npm create vite@latest  my-react-app
cd my-react-app
npm install
npm run dev
```

4) Put your app on GitHub

1. Create a new repository on GitHub.

2. In your project folder run:

```
git init
git add .
git commit -m "Initial commit"
git branch -M main
```

```
git remote add origin <your-repo-url>
git push -u origin main
```

5) Create the Static Web App in Azure Portal

1. Sign in to the Azure Portal.
2. Search for **Static Web Apps** → click **Create**.
3. Fill in the form:
    - **Subscription**: choose your subscription.
    - **Resource group**: create or select one.
    - **Name**: unique name (part of the default URL).
    - **Plan type**: Free (good for beginners).
    - **Region**: pick one near you.
4. Deployment details:
    - **Source**: choose **GitHub** and sign in.
    - **Repository**: pick the repo you pushed.
    - **Branch**: usually `main`.
5. Build details:
    - **Framework preset**: React.
    - **App location**: `/` (or folder name if your app is in a subfolder).
    - **Output location**: `build` (for create-react-app).
    - **API location**: leave empty unless you have an API.
6. Review + Create → Create.

Azure will now set up the app and automatically add a GitHub Actions workflow to your repo.

6) Watch the GitHub Actions workflow

- Go to your GitHub repo → **Actions** tab.
- A workflow will appear, running steps like installing dependencies, running `npm run build`, and deploying to Azure.
- Once it finishes, your app is deployed.

7) Access your app

- In Azure Portal, open your Static Web App resource.

- Under **Overview**, click the generated URL (e.g., `https://yourapp.azurestaticapps.net`).
- 🎉 Your app is live!

---

8) Fix React Router (SPA routing)

If you refresh a page other than `/` and get a 404, you need a config file.

Create a file called **`staticwebapp.config.json`** in your project root (so it ends up in the `build` folder after `npm run build`):

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

Commit and push this file — the workflow will redeploy your app, and routes will now work correctly.

---

9) Environment variables

- **Frontend variables** (like `REACT_APP_API_URL`) must be defined at build time.
  - Add them as **secrets** in your GitHub repo.
  - In the workflow file (`.github/workflows/azure-static-web-apps.yml`), pass them as environment variables:

    ```
    env:
      REACT_APP_API_URL: ${{ secrets.REACT_APP_API_URL }}
    ```

- **Backend variables** (if you use Azure Functions) can be set in the Azure Portal under **Configuration**.

---

10) Custom domain (optional)

1. In the Static Web App resource, go to **Custom domains** → **+ Add**.
2. Enter your domain (e.g., `www.myapp.com`).
3. Update DNS at your registrar to point to the provided Azure hostname.
4. Azure validates and automatically sets up HTTPS.

---

11) Common beginner issues

- **Build fails in GitHub Actions**: Run `npm run build` locally to find errors.
- **404 on refresh**: Add `staticwebapp.config.json` as shown above.
- **Secrets not working**: Ensure you add them as GitHub secrets and map them into the workflow environment.
- **Custom domain not resolving**: Check DNS propagation and that you added the right record type (usually CNAME).