Full Deployment Guide – Vedic Astro App

This document describes a reusable deployment process for the Vedic Astro application using:

1. **GitHub** – source control and deployment trigger.
2. **MongoDB Atlas** – managed MongoDB database.
3. **Render.com** – backend hosting (Node/Express).
4. **Vercel** – frontend hosting (React/Vite).

You can follow these steps for the first deployment and for any future re-deployments.

---

1. Prerequisites

1. **Accounts**
    1. GitHub account.
    2. MongoDB Atlas account.
    3. Render account.
    4. Vercel account.
2. **Local Tools**
    1. Git installed.
    2. Node.js + npm installed (for local testing).
3. **Project Structure (expected)**
    1. `backend/` – Node/Express backend, `server.js`, `package.json`.
    2. `frontend/` – React/Vite frontend, `vite.config.js`, `package.json`.

---

2. One-Time Setup

2.1 Push Project to GitHub

1. Initialize and commit (if not already):

    ```
    git init
    git add .
    git commit -m "Initial commit for deployment"
    ```

2. Create a GitHub repository (for example `vedic-astro-app`).

3. Connect and push:

    ```
    git remote set-url origin https://github.com/<GITHUB_USER>/vedic-astro-
    app.git
    ```

```
git branch -M main
git push -u origin main
```

4. Verify that the repository is visible on GitHub.

2.2 Configure MongoDB Atlas

1. Log in to MongoDB Atlas and create a **free Shared Cluster**.

2. Create a database user with username and password.

3. In **Network Access**, add IP `0.0.0.0/0` (allows cloud services to connect).

4. Get the connection string:

   1. Go to **Databases** → your cluster → **Connect** → **Drivers**.
   2. Select **Node.js**.
   3. Copy the URI and replace `<password>` with your real password.
   4. Ensure special characters in the password are URL-encoded (Atlas usually does this for you).

5. Keep this connection string safe; it will be used as `MONGODB_URI`.

---

3. Backend Deployment – Render.com

3.1 Create / Configure the Web Service

1. Go to the Render dashboard.
2. Click **New** → **Web Service**.
3. Select your GitHub repository or provide its public Git URL.
4. Configure the service:
   1. **Name**: `vedic-astro-api` (or similar).
   2. **Branch**: `main`.
   3. **Root Directory**: `backend`.
   4. **Runtime**: Node.
   5. **Build Command**: `npm install`.
   6. **Start Command**: `node server.js`.

3.2 Environment Variables (Render)

1. Open the **Environment** or **Environment Variables** section.

2. Add the following variables:

   1. `PORT` = `10000` (or the port recommended by Render).
   2. `MONGODB_URI` = your MongoDB Atlas connection string.
   3. `JWT_SECRET` = a long, random string used to sign JWT tokens.

3. Optional variables (if email features are used):

   1. `EMAIL_USER` = SMTP username (for example Gmail address).

   2. `EMAIL_PASSWORD` = SMTP or app-specific password.

3.3 Deploy and Verify Backend

1. Click **Create Web Service** (for first time) or **Manual Deploy** → **Clear build cache & deploy** (for updates).

2. Wait until the service status is **Live**.

3. In the **Logs** tab, ensure that:

   1. Dependencies install successfully.
   2. The server starts on the configured `PORT`.
   3. The MongoDB connection succeeds (no `ENOTFOUND` or authentication errors).

4. Note the public backend URL, for example:

   ```
   https://vedic-astro-api.onrender.com
   ```

5. Use this URL later as `VITE_API_URL` for the frontend.

---

4. Frontend Deployment – Vercel (Vite + React)

4.1 Import Project

1. Go to the Vercel dashboard.
2. Click **Add New...** → **Project**.
3. Select the same GitHub repository.
4. In "Root Directory" settings, choose `frontend`.
5. Set **Framework Preset** to `Vite` if not auto-detected.

4.2 Environment Variables (Vercel)

1. In the Vercel project setup screen, locate **Environment Variables**.

2. Add:

   1. `VITE_API_URL` = the public backend URL from Render, for example:

      ```
      https://vedic-astro-api.onrender.com
      ```

   2. Do not include a trailing slash `/` at the end.

3. Optional additional variables (if used by the frontend):

   1. `VITE_APP_NAME`
   2. `VITE_GOOGLE_CLIENT_ID`

4.3 Deploy and Verify Frontend

1. Click **Deploy**.
2. Wait for the deployment to become **Ready**.
3. Note the Vercel domain, for example:

   ```
   https://vedic-astro-app.vercel.app
   ```

4. Open this URL in a browser and verify:
   1. The app loads correctly.
   2. Main user flows (e.g. guest login, chart generation) work.
5. In browser DevTools → Network tab, confirm API requests are going to `VITE_API_URL` (Render backend) and returning successful responses.

---

5. End-to-End Checklist

1. **Backend**
   1. Render service is Live.
   2. Logs show no critical errors.
   3. MongoDB Atlas connection works.
2. **Frontend**
   1. Vercel deployment is Ready.
   2. `VITE_API_URL` correctly points to the Render backend.
3. **User Flows**
   1. Initial page loads without errors.
   2. User can proceed as guest or via login (if enabled).
   3. Chart generation and retrieval work end-to-end.

---

6. Updating the Application (Future Deployments)

1. Make code changes locally in `backend` or `frontend`.
2. Commit and push to `main`:

   ```
   git add .
   git commit -m "Describe your change"
   git push
   ```

3. **Render**
    1. Automatically detects new commits and redeploys the backend.
    2. Alternatively, trigger **Manual Deploy** from the Render dashboard.
4. **Vercel**
    1. Automatically rebuilds the frontend when new commits are pushed to the configured branch.
    2. You can also trigger manual redeploys from the Vercel dashboard.
5. Only change secrets and configuration values in:
    1. Render environment variables.
    2. Vercel environment variables.
    3. Local `.env` files (which must not be committed to Git).

---

7. Cloning or Recreating the Setup

To recreate the full setup for another environment (for example staging or a second production instance):

1. **Database**
    1. Create a new MongoDB Atlas database or cluster.
    2. Generate a new connection string.
2. **Backend (Render)**
    1. Create a new Web Service pointing to the same GitHub repo.
    2. Use a different service name (for example `vedic-astro-api-staging`).
    3. Set environment variables with the new `MONGODB_URI` and `JWT_SECRET`.
3. **Frontend (Vercel)**
    1. Create a new Vercel project using the same GitHub repo.
    2. Set the root directory to `frontend`.
    3. Configure `VITE_API_URL` to point to the new Render service URL.
4. Verify the new environment using the same end-to-end checklist.

---

8. Notes and Best Practices

1. **Security**
    1. Never commit `.env` files or secrets to Git.
    2. Restrict MongoDB Atlas network access if you later move away from fully public (`0.0.0.0/0`).
2. **Observability**
    1. Use Render logs to debug backend issues.
    2. Use browser DevTools and Vercel logs to debug frontend/runtime issues.

3. **Scalability**

    1. Upgrade Atlas cluster tier if database load increases.
    2. Upgrade Render and Vercel plans when you need more resources or uptime guarantees.

---

9. Quick Summary

1. Code lives in a single GitHub repository.
2. Backend is deployed from `/backend` to **Render.com** with environment variables configured for MongoDB Atlas and JWT.
3. Frontend is deployed from `/frontend` to **Vercel**, with `VITE_API_URL` set to the Render backend URL.
4. Pushing code to GitHub automatically triggers new deployments on both Render and Vercel.