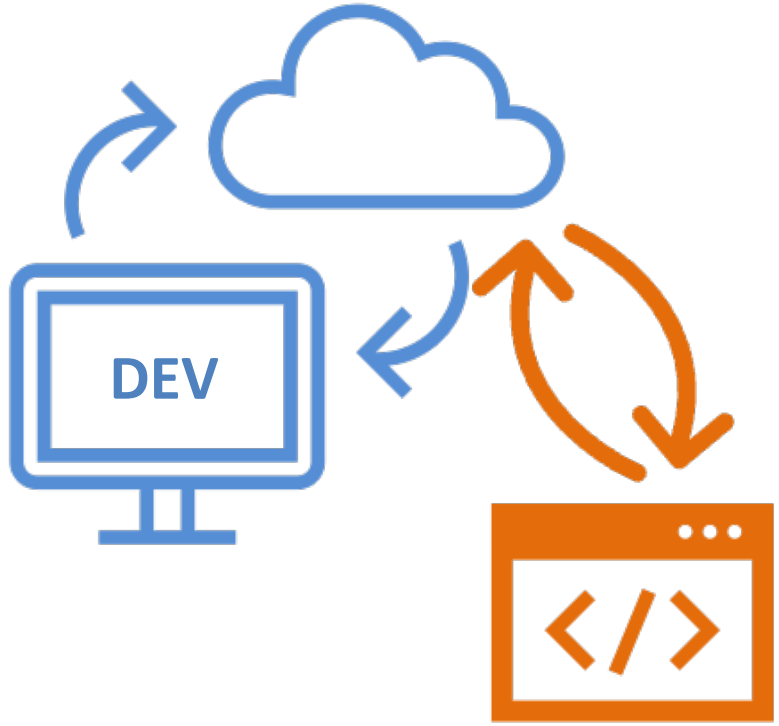
<WA/>
2024

Deployment

How/where to put the app in the “cloud”

Enrico Masala
Antonio Servetti



Deployment to Cloud Servers

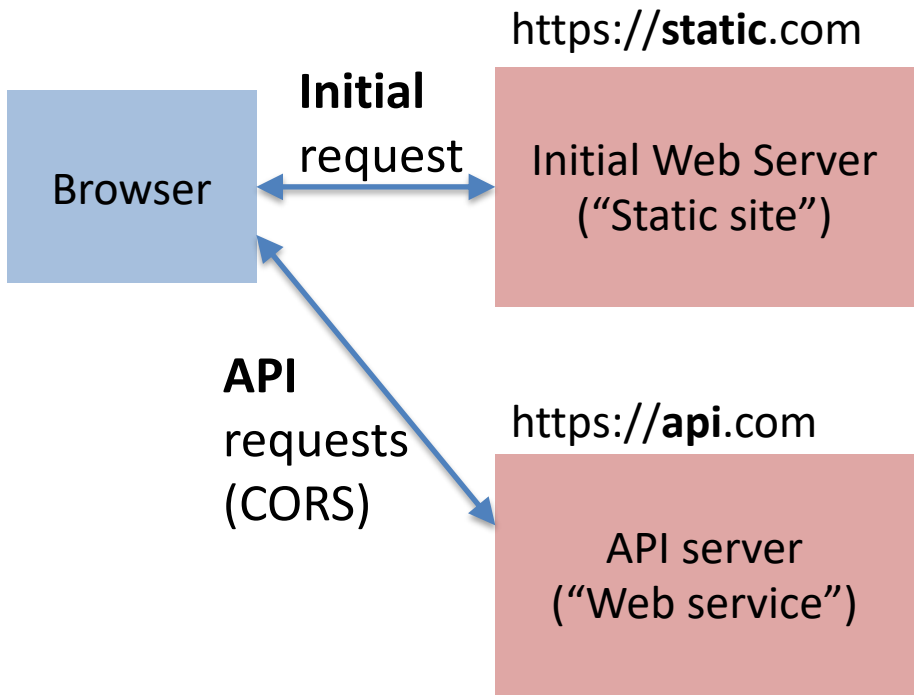
- Real applications are accessible by anybody connected to the Internet
- Need to set up a web server to
  - Serve the static files of the React client application (“landing page”)
  - Provide the API services used by the client
- Configuring servers from scratch, starting from bare virtual machines, is difficult, error prone, and requires expertise in system administration
  - Selecting the best web server software (Apache, nginx, …)
  - Configuring it
  - Configure the security part (HTTPS, certificates, private keys, algorithms, etc.)

Deployment to Cloud Servers

- Most of the activity is mechanical and the same for many cases
- Some online services offer the possibility to simplify deployment
  - **Pre-configured static or API servers**
  - Possibility to run node.js as backend server
  - Possibility to use a database instance
- Platform-as-a-Service approach (PaaS)
  - Many (paid) services exists
  - Very few free ones, especially for the API server (require computational resources)
    - Mostly oriented to become familiar with the system by deploying a toy project, then upgrade to a paid profile

Deployment Architecture

- Use two servers, configured with CORS
- The server runs as-is
- Client: create a “build” with vite
  - `npm run build`
- The build will be put in the `/dist` folder
  - These are all the files for the static web server
  - Note: keep the original files!
    From the build you cannot go back
    to the original sources



Vite Build

- `npm run build` produces all the files needed for a static web server
  - A static `index.html` file that loads:
  - JS file (including all libraries, Bootstrap, …), CSS file,
  - other resources (icons, fonts, etc.)

```
> react-qa@0.0.0 build
> vite build

vite v5.2.9 building for production...
✓ 350 modules transformed.
dist/index.html                                  0.46 kB │ gzip:  0.30 kB
dist/assets/bootstrap-icons-BtvjY1KL.woff2     130.40 kB
dist/assets/bootstrap-icons-BOrJxbIo.woff      176.03 kB
dist/assets/index-CFLmWFf5.css                 307.59 kB │ gzip: 44.37 kB
dist/assets/index-BHo4gWFZ.js                  220.24 kB │ gzip: 72.33 kB
✓ built in 692ms
```

Static Build

- Any needed resource is put into static files, served by the static server
  - Libraries (e.g., Bootstrap etc.) are minified to save bandwidth
- No leak of information towards third-party servers (CDNs) due to css, fonts, etc.)
  - Might be important for legal reasons (privacy law disclosures, etc.)

index.html

```
<!doctype html>
<html lang="en">
  <head> <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
    <script type="module" crossorigin src="/assets/index-BHo4gWFZ.js"></script>
    <link rel="stylesheet" crossorigin href="/assets/index-CFLmWFf5.css">
  </head>
  <body><div id="root"></div></body>
</html>
```

index-BHo4gWFZ.js

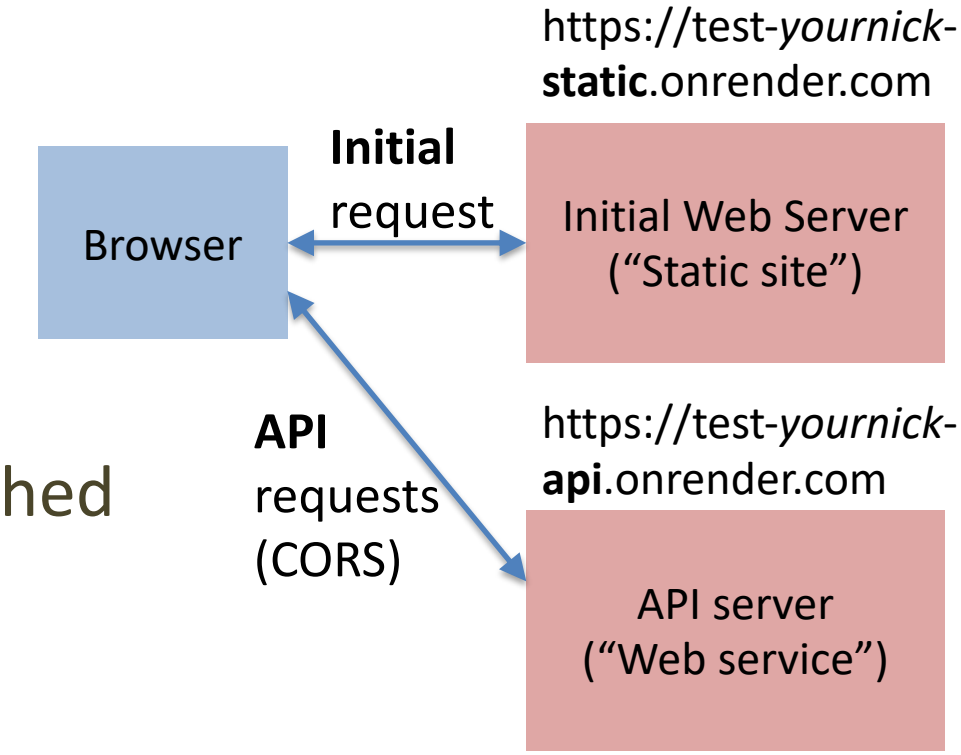
```
function gp(e,t){for(var
n=0;n<t.length;n++){const
r=t[n];if(typeof
r!="string"&&!Array.isArr
ay(r)){for(const l in
r)if(l!=="default"&&!(l
in e)){…
```

Suggested Deployment Approach

- Identify a free service that can support an API server running node.js
  - Example: https://render.com
- Create an account (free for basic, simple services)
- Make your projects (client & server) accessible to the service
  - Typically, via a git repository from which the service load the files
  - Git approach is easy to maintain and update: automatic deploy after commit/push
  - With render.com: create 1 GitHub repository (either private or public) with your code, in two separate folders (client and server)
    - If private, later you will have to grant permission to render.com to access it

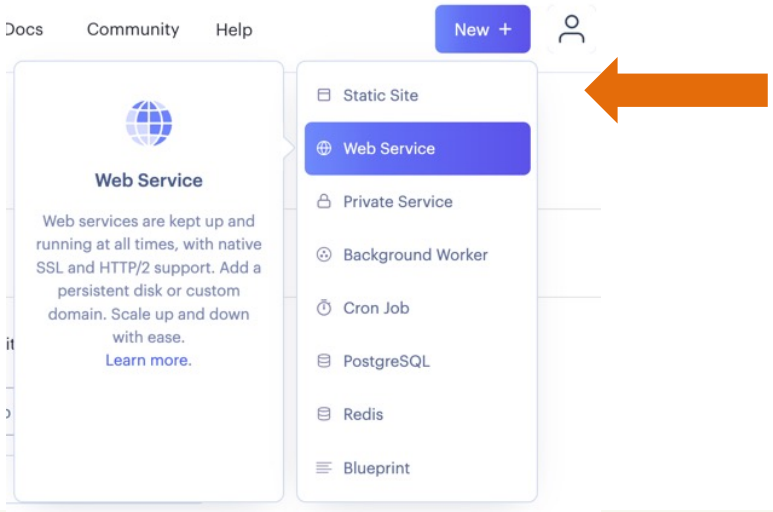
Implement the Architecture

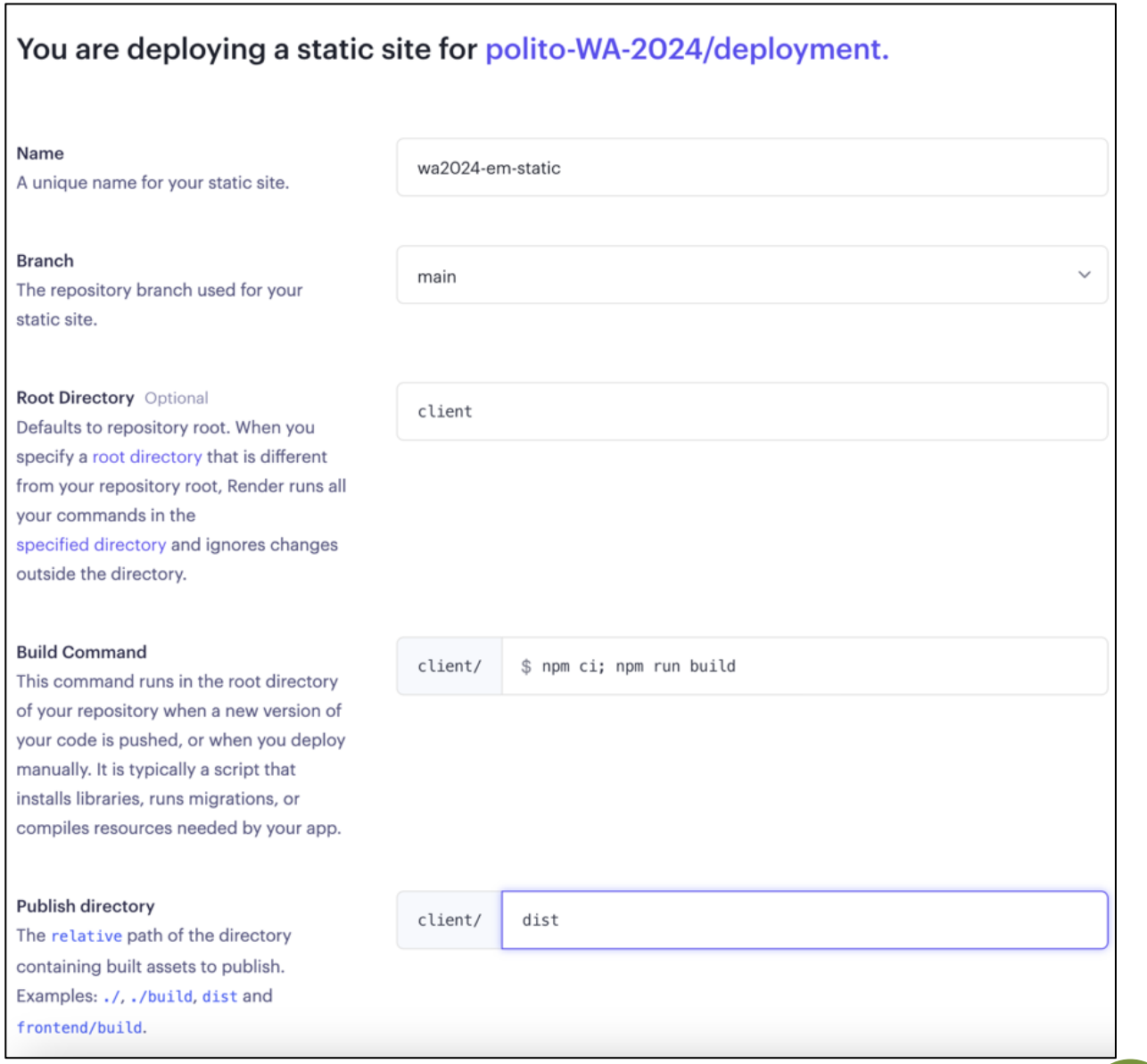
- Go to the https://render.com dashboard, create the following two services with names of your choice
  - **Static site**, e.g. test-*yournick*-static
  - **Web service**, e.g., test-*yournick*-api
- Note that the names will determine part of the URL at which the server can be reached
  - https://test-*yournick*-static.onrender.com
  - https://test-*yournick*-api.onrender.com



Static Site

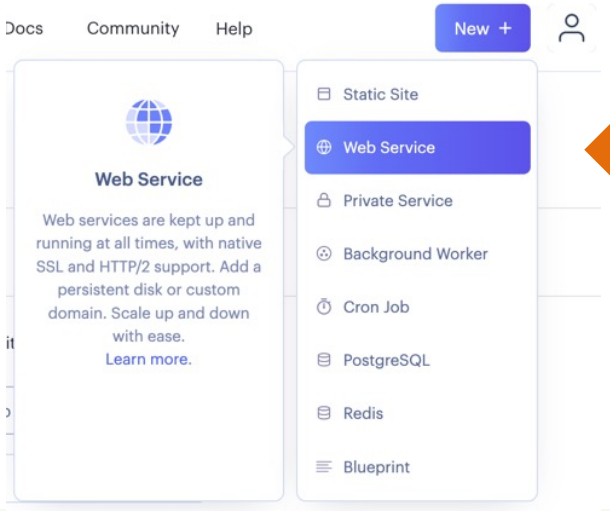
- Root Directory: `client`
- Build command:
  `npm ci; npm run build`
- Publish directory: `dist`

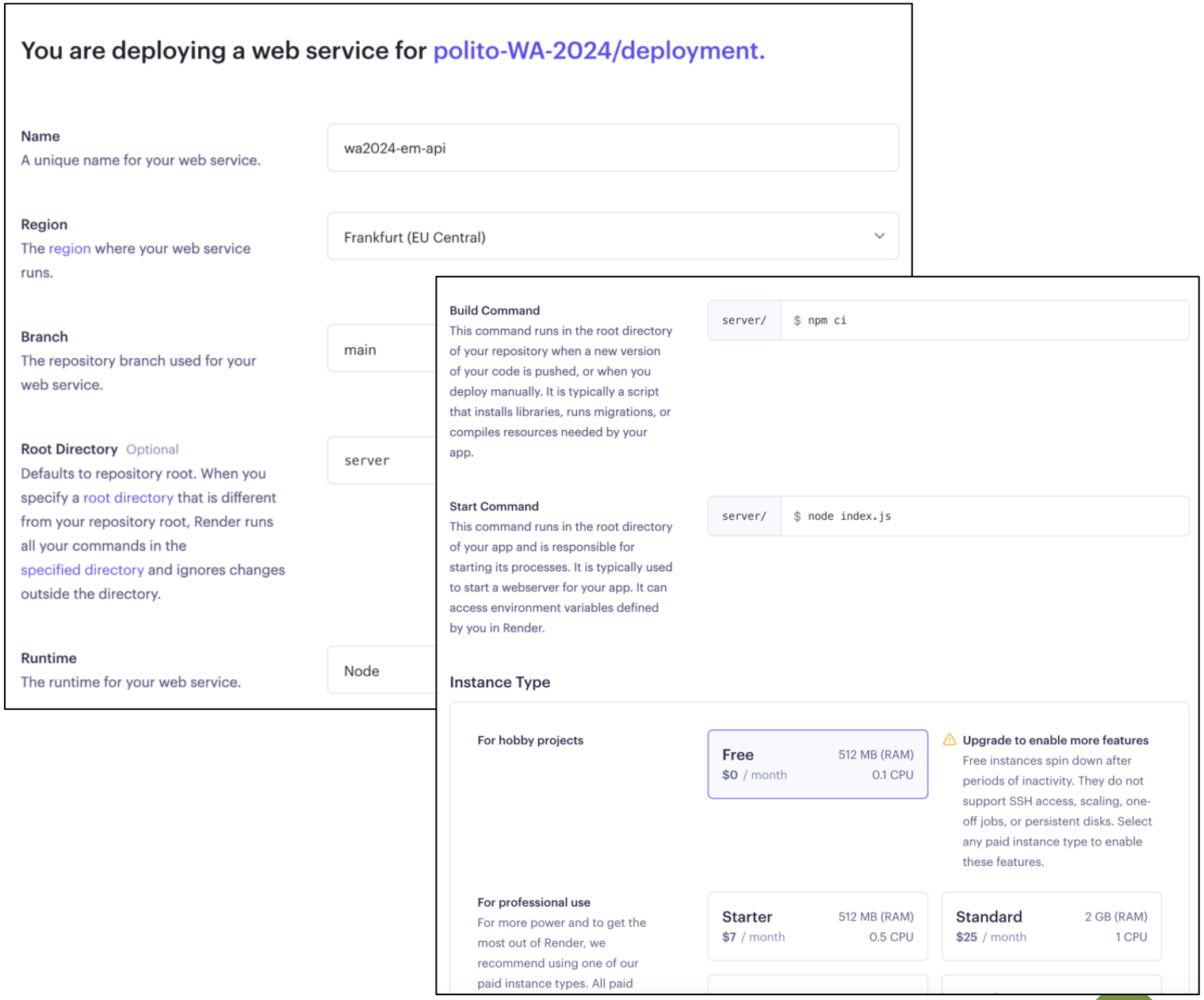




Web Service

- Root Directory: `server`
- Build command: `npm ci`
- Start command: `node index.js`





Preparing the Project for Deployment

- Configure CORS and HTTP requests correctly
  - The server must allow requests from https://test-*yournick*-static.onrender.com
  - The client must send API requests to https://test-*yournick*-api.onrender.com
- Make sure that the serve runs on the correct port
  - The render.com system requires the server to listen on localhost, port 10000
  - The port will be exposed to the public internet via a reverse-proxy server mechanism, that will also provide the TLS certificate for HTTPS
- Configure the use of the secure option in cookies
  - Modify the express-session configuration

Environment Variables

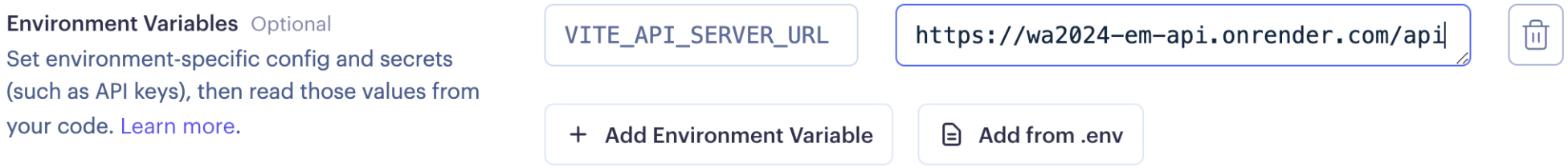
- In production environments, many configuration parameters are made available as environment variables
  - Accessible by the "`process.env`" Javascript special variable
  - **Very convenient to change configuration without changing files!**
    - Also, without having to commit/push such files to the git repository!
  - In render.com such values can be set/changed externally, using the dashboard
- Note: a manual re-deployment may be needed for changes to environment variables to take effect

Client Config

- For the client build with Vite, use
  - "`import.meta.env`" JS variable
  - Every variable name **must start** with **`VITE_`**...
  - This is to avoid accidentally exposing other environment variables

```
// API.js file
//const URL = 'http://localhost:3001/api';
const URL = import.meta.env.VITE_API_SERVER_URL;
```

Value to be set after starting the "Web Service" from the Dashboard



Server Config

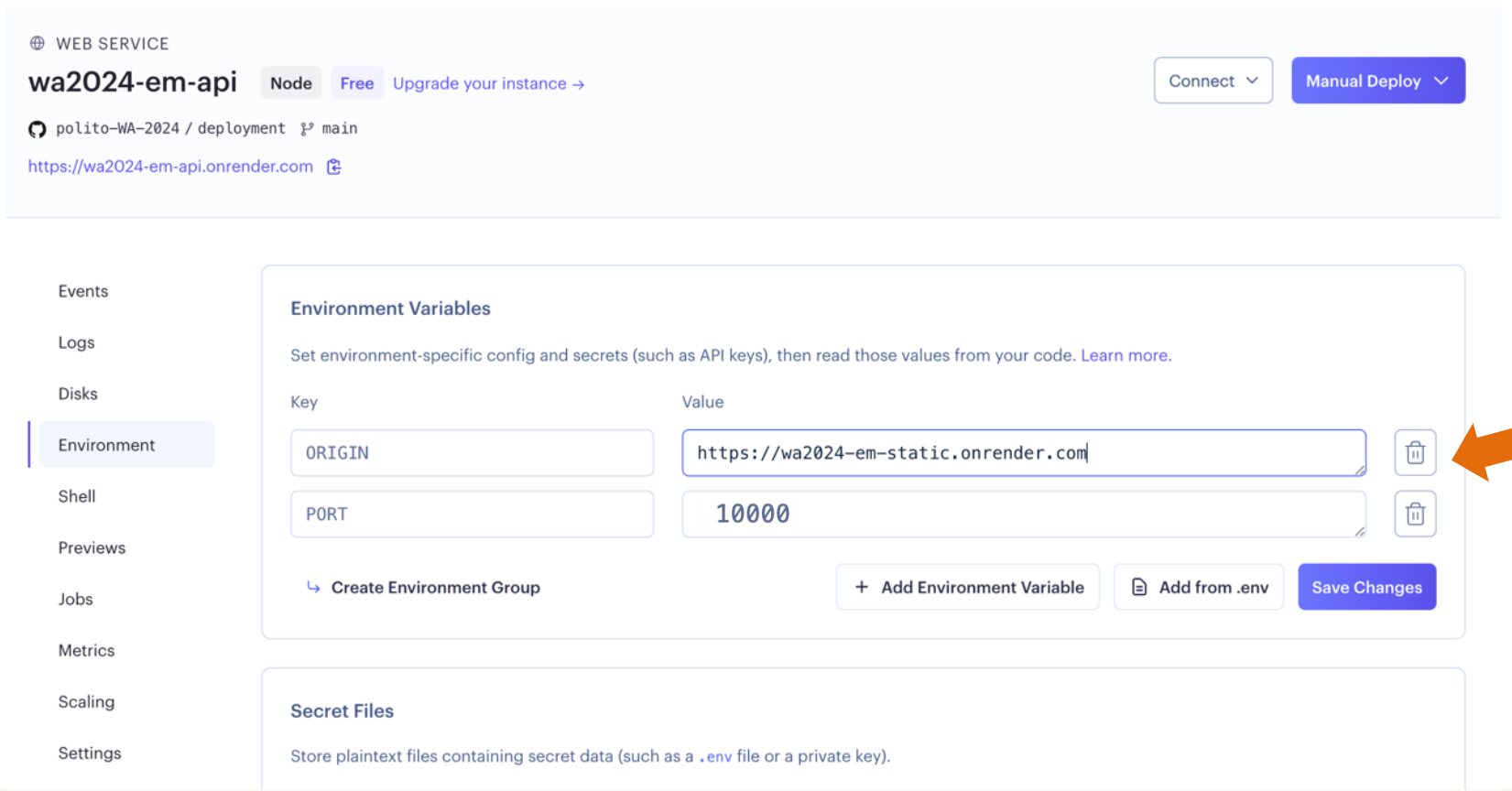
- Use two environment runtime variables, PORT and ORIGIN

```
const app = express();
//const port = 3001;
const port = process.env.PORT || 3001;

const corsOptions = {
  //origin: 'http://localhost:5173',
  origin: process.env.ORIGIN || 'http://localhost:5173',
  credentials: true,
};
```

Server Config

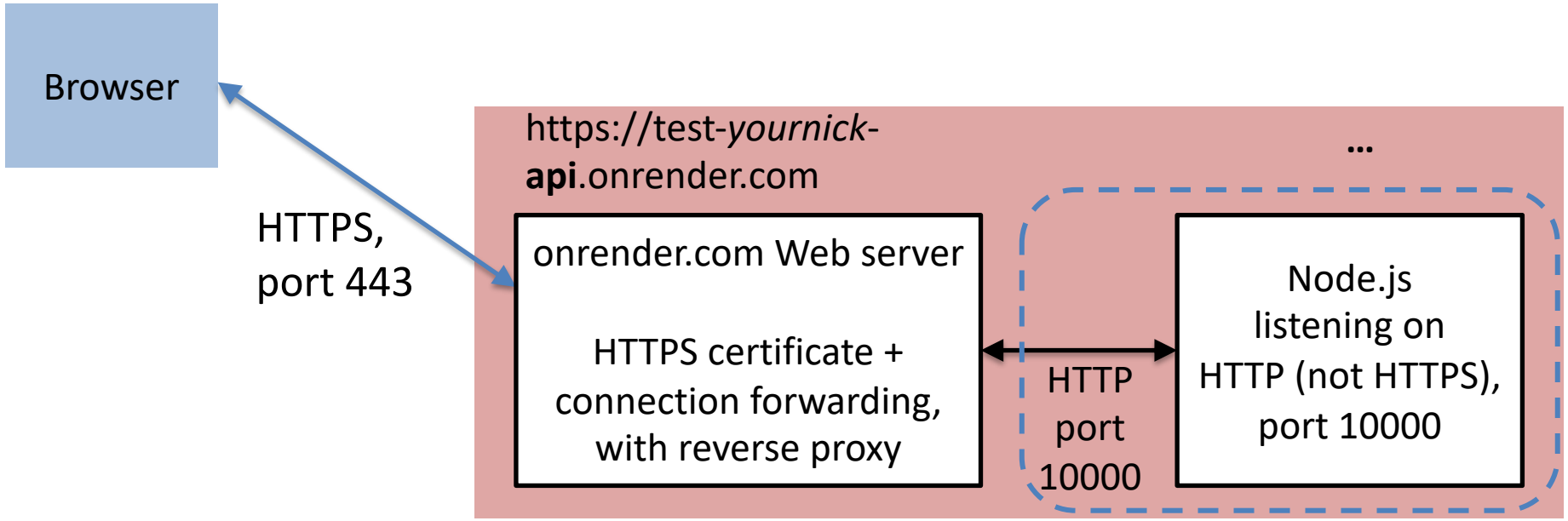
- Set the environment variables depending on the public URL of “Static Site”



Value to be set by looking at the “Static Site” URL from the Dashboard

Server Config: HTTPS management

- Node.js must be informed that it is sending a secure cookie on an insecure connection because it is behind a reverse proxy that will handle the HTTPS
  - Otherwise, it will silently refuse to do it



Fully isolated from other deployments in render.com

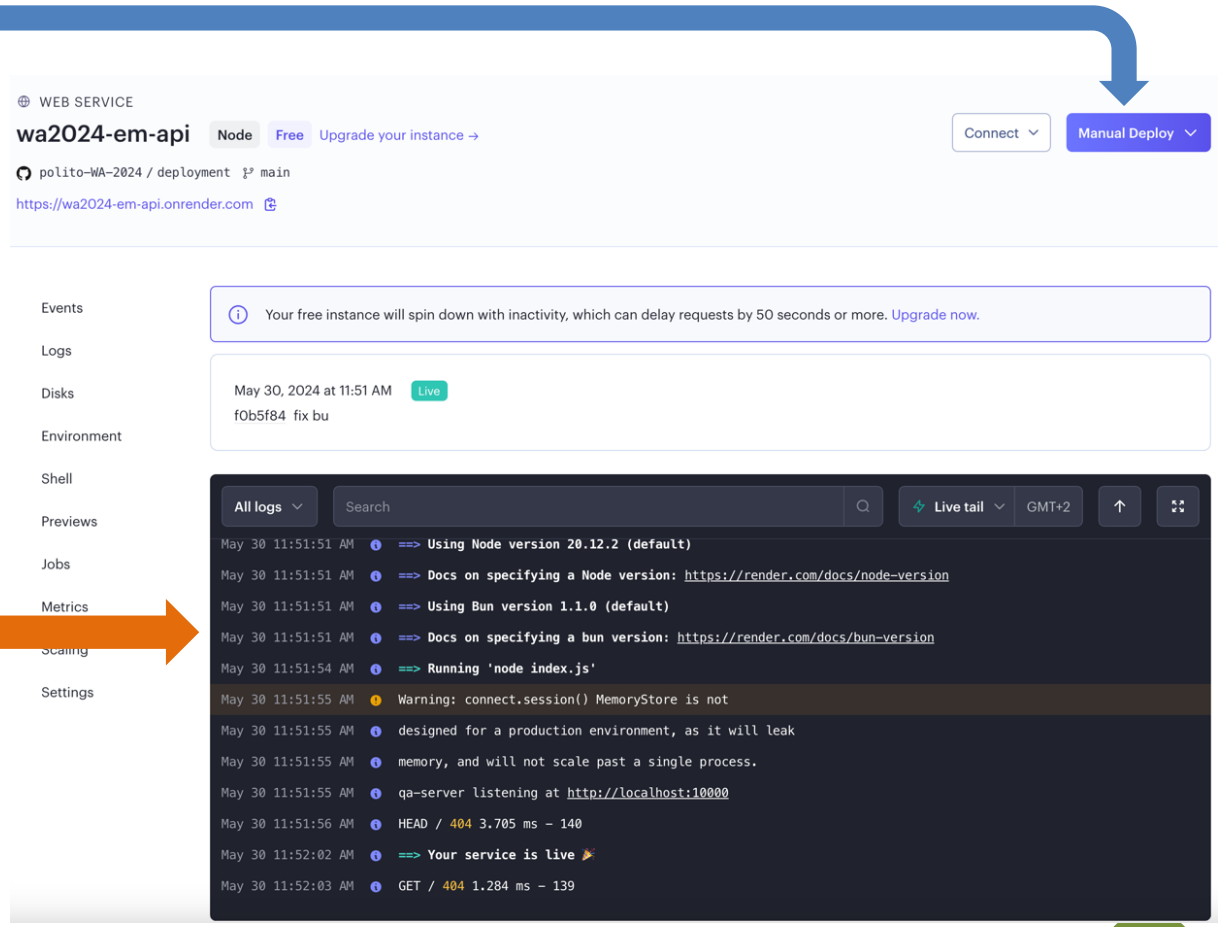
Server Config: HTTPS & Proxy Configuration

- Check if node is running in production mode
- Use the built-in NODE_ENV variable (already set by render.com)

```
if (process.env.NODE_ENV === 'production')
  app.set('trust proxy', 1);  // To work behind reverse-proxy
  // NB: Without this setting, “secure” cookies will not work
…
app.use(session({
  secret: '……', // this could also be an env variable if desired
  resave: false, saveUninitialized: false,
  cookie: { httpOnly: true,
    secure: (process.env.NODE_ENV==='production’? true : false) },
}));
```

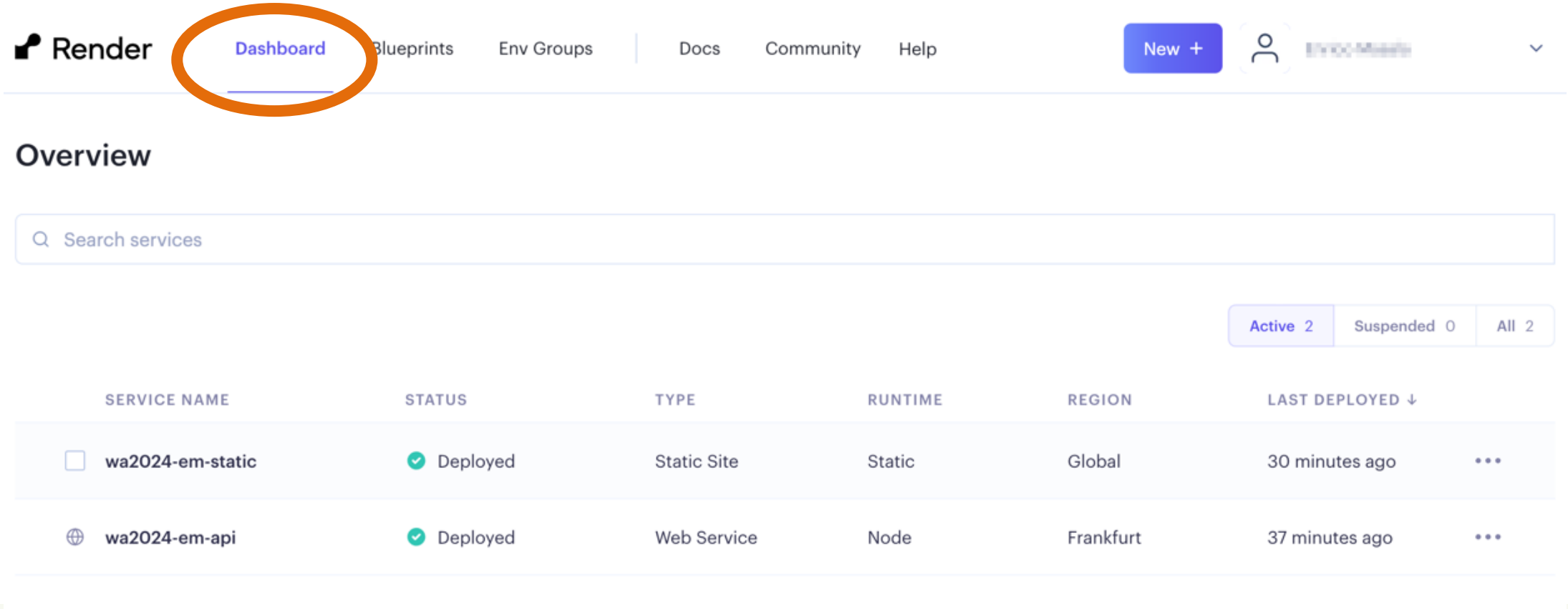
Actual Deployment

- From the dashboard, deploy both the “Static Site” and the “Web Service”
  - The first time you configure the service, deployment is automatically started
- You can check the result and errors by looking at the console which is integrated into the page



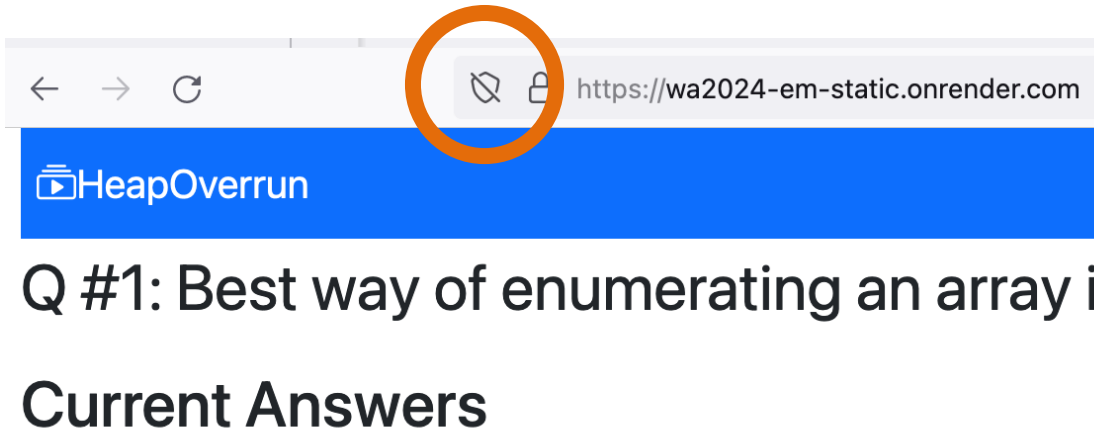
Active Services

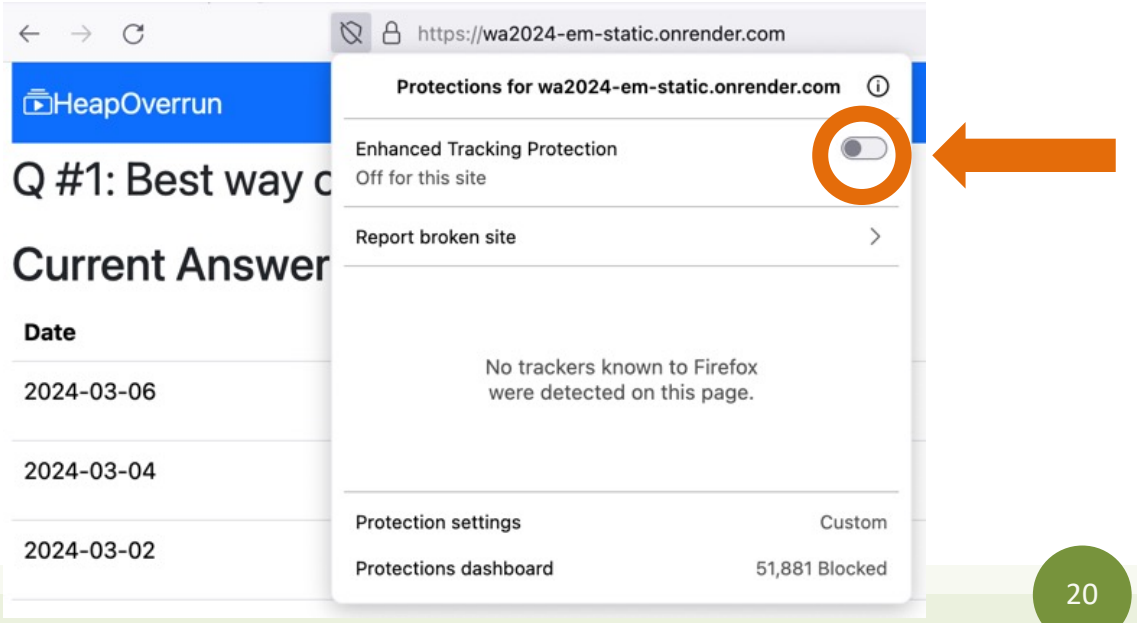
- Also look at the dashboard to check the status of your services



Test the App!

- Open a browser, go to https://test-*yournick*-static.onrender.com and test!
- **IMPORTANT: DISABLE** browser cross-site "Tracking Protection", e.g. in Firefox, click on the Shield Icon on the left side of the URL box
  - This is needed because you are using a public, shared domain name (onrender.com)





Beware: Notes

- In the free “Web Service”, files will be often deleted, after few minutes of inactivity (in particular, the sqlite3 file)
  - This limitation can only be removed by subscribing to non-free tiers
  - In any case, useful to test “toy” examples and become familiar with the system before paying for one
- Browsers are extremely careful when sharing information on domains where anybody can register subdomains for free (e.g., onrender.com)
  - Registering your own domain name is possible but not free
  - Remember: in this case you need to handle/provide your own HTTPS certificate

License

- These slides are distributed under a Creative Commons license “**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**”
- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- https://creativecommons.org/licenses/by-nc-sa/4.0/