# Deploying Simple AI Agents on Various Platforms

Deploying an AI agent involves hosting your application’s code so that users can interact with it online. In this guide, we compare several deployment options – from easy, **Git-based platforms** (like Hugging Face Spaces or Streamlit Cloud) to more **general cloud providers** (like AWS). The goal is a **simple, modular solution** where heavy AI processing is outsourced (e.g. using OpenAI’s API for language tasks and a cloud vector database), with the flexibility to plug in custom models or databases later. We assume you have a Python-based agent (using frameworks like Streamlit or FastAPI) and possibly a Node.js component; we’ll cover how to deploy these components step by step.

## Architecture and Preparation 📐

Before deploying, it’s important to structure your AI agent for **modularity** and external services:

* **Outsource Heavy Processing:** Rather than hosting a large language model yourself, use a hosted API (e.g. OpenAI) for inference and a managed vector database for similarity search. This *retrieval-augmented generation* architecture is common in modern LLM apps[[1]](https://a16z.com/emerging-architectures-for-llm-applications/#:~:text=There%20are%20many%20different%20ways,possible%20now%20with%20foundation%20models). It keeps your deployable app lightweight, as the cloud handles the heavy LLM computation.
* **Modularity for Custom Solutions:** Design your code so that the interfaces to the LLM and database are abstracted. For example, if you use a library like LangChain or LangGraph, keep the vector store and LLM as interchangeable components. This way, you can later **swap in custom solutions** (like an open-source model on your own server, or a different database) without changing the overall app structure[[2]](https://a16z.com/emerging-architectures-for-llm-applications/#:~:text=,offerings%20like%20those%20listed%20above). The deployment process remains the same – you’d just point the app to a different API endpoint or database URL.
* **Environment Variables for Secrets:** Prepare to use environment variables (or secrets management) for any API keys or credentials (OpenAI keys, database URLs, etc.). Never hard-code secrets in your code. All the platforms discussed allow setting secret keys that your app can read at runtime. For example, Streamlit Cloud and Hugging Face Spaces let you define secrets in the app settings, which your code can access via st.secrets or as env variables[[3]](https://docs.streamlit.io/deploy/streamlit-community-cloud/deploy-your-app/secrets-management#:~:text=Docs%20docs,your%20code%20to%20read%20secrets)[[4]](https://huggingface.co/docs/huggingface_hub/en/guides/manage-spaces#:~:text=%3E%3E%3E%20api.delete_space_variable%28repo_id%3Drepo_id%2C%20key%3D). Similarly, AWS Elastic Beanstalk (EB) allows configuring environment variables for your app (including any keys or settings)[[5]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=After%20the%20environment%20update%20is,credentials%20to%20our%20FastAPI%20app).
* **GitHub Repository:** Ensure your project code is in a Git repository (e.g. on GitHub). Many deployment platforms integrate directly with GitHub for easy CI/CD. *Note:* GitHub itself is not a hosting service for running back-end apps (aside from GitHub Pages for static sites or Codespaces for development). However, it’s used to version your code and connect to the deployment services below.

With these in place, you’re ready to deploy. Next, we’ll explore **two beginner-friendly deployment options** (Hugging Face Spaces and Streamlit Community Cloud), then an **AWS deployment**, and finally mention other alternatives. Each option includes detailed steps and notes on pros/cons.

## Option 1: Deploy on Hugging Face Spaces (Streamlit or Gradio App)

Hugging Face Spaces is a free-hosting platform for ML apps. It supports Streamlit and Gradio apps out-of-the-box, and even custom Docker setups[[6]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Fhuggingface,has%20a%20free%20tier%20for). The free tier gives you a machine with **2 CPU cores and 16 GB RAM** to run your app[[7]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=code%2C%20and%20Spaces%20lets%20you,Overall%2C%20Hugging%20Face%20is%20beginner) – plenty for most AI demos. This platform is great for showcasing a Streamlit-based chatbot or agent quickly, with all heavy language processing offloaded to APIs.

**Steps to Deploy on Hugging Face Spaces:**

1. **Prepare your App**: Make sure your application script (e.g. app.py if using Streamlit) is working locally. Create a requirements.txt listing all Python dependencies. For example, your requirements might include streamlit, fastapi, langchain, psycopg2 (for Postgres), etc. This requirements file ensures the cloud environment installs all needed packages[[8]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=you%20to%20run%20git%20commands,app%20files%20to%20the%20web).
2. **Sign Up on Hugging Face**: Create a Hugging Face account (free) if you don’t have one[[9]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=3,Face%20Account).
3. **Create a New Space**: On Hugging Face, go to the Spaces section and create a new Space[[10]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=3). Choose an appropriate name and select the SDK **“Streamlit”** (or **“Gradio”** if you built a Gradio interface). Set visibility to “public” (unless you have a paid plan for private Spaces). Then click **Create Space**[[11]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,Hit%20the%20%E2%80%98Create%20space%E2%80%99%20button). Spaces will provision a repo for your app.
4. **Upload Code to the Space**: You have two ways:
5. **Via Git**: The Space you created is essentially a Git repository. You can find its Git URL on the Space page. Clone the repo to your local machine using git clone <your-space-url>[[12]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=3,your%20New%20Hugging%20Face%20Space). This gives you an empty folder. Copy your app.py (or main script) and requirements.txt into this folder[[13]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,to%20your%20Hugging%20Face%20Space). Then git add ., git commit -m "first commit", and git push back to the Hugging Face remote[[14]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,to%20your%20Hugging%20Face%20Space).
6. *Or*, **Upload via Web UI**: On the Space page, you can upload files directly. However, using Git is recommended for version control. Since you mentioned familiarity with GitHub, the git method should be comfortable.
7. **Automatic Build and Launch**: Once you push the code, Hugging Face will automatically build the environment (install requirements) and launch your app. Within a minute or two, your app becomes live at a URL like https://huggingface.co/spaces/<YourUsername>/<YourSpaceName>[[15]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=cd%20name_of_your_new_folder%20git%20add%20,git%20push). You’ll see logs on the Space page if it’s building or if any errors occur.
8. **Configure Secrets (API keys)**: If your app needs secret keys (for OpenAI, etc.), go to the Space’s Settings > **Repository secrets**. Add your secrets there (e.g. OPENAI\_API\_KEY) – these will be injected as environment variables in the running app[[16]](https://huggingface.co/docs/huggingface_hub/en/guides/manage-spaces#:~:text=3)[[4]](https://huggingface.co/docs/huggingface_hub/en/guides/manage-spaces#:~:text=%3E%3E%3E%20api.delete_space_variable%28repo_id%3Drepo_id%2C%20key%3D). In a Streamlit app, you can also use a .streamlit/secrets.toml file or directly access os.environ. The Hugging Face documentation confirms that any secrets you add are available inside your app environment[[4]](https://huggingface.co/docs/huggingface_hub/en/guides/manage-spaces#:~:text=%3E%3E%3E%20api.delete_space_variable%28repo_id%3Drepo_id%2C%20key%3D). This keeps your keys safe (they won’t be visible in the public code).
9. **Test the Live App**: Navigate to your Space URL. You should see your Streamlit interface and can interact with the AI agent. The app will call the external APIs (OpenAI, etc.) just as it did locally. Hugging Face keeps the app running as long as it’s being used. If the app is idle for too long, it may sleep, but it will wake on the next user visit.

**Pros & Cons (Hugging Face Spaces):**

* ✅ **Free and Easy:** Great for quick demos or capstone projects – no server setup, just push your code. You get a nice subdomain and can showcase your project to anyone[[7]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=code%2C%20and%20Spaces%20lets%20you,Overall%2C%20Hugging%20Face%20is%20beginner)[[17]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Fstreamlit,1%20hr%20idle).
* ✅ **Generous Resources:** 16 GB RAM on the free tier is quite high[[7]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=code%2C%20and%20Spaces%20lets%20you,Overall%2C%20Hugging%20Face%20is%20beginner), enabling use of moderately large models or data in memory (though the heavy lifting should be on external services as planned).
* ✅ **Streamlit/Gradio Support:** Streamlit apps run seamlessly (the Space automatically handles running streamlit run app.py). This suits your use of Streamlit for a UI. If you had a FastAPI app instead, you could still use Spaces by selecting the “Docker” SDK and providing a Dockerfile, but that’s more work – Streamlit is plug-and-play here.
* ⚠️ **Public by Default:** On the free plan, your Space (code and app) must be public. Anyone can see your repository files by default[[18]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Streamlit%20Cloud). This is usually fine for a student project, but be sure not to expose secrets (use the secrets feature as described). If you need a private app, Hugging Face requires a paid upgrade.
* ⚠️ **UI Frame:** The app appears within the Hugging Face interface. There’s a top bar with the Hugging Face logo and menus. It’s mostly cosmetic, but the app isn’t full-screen by itself[[19]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Hugging%20Face%20Spaces). For most purposes this is not an issue (and the HF branding can lend credibility that it’s an ML app platform).
* ⚠️ **Limited Customization:** You cannot easily add a custom domain on the free Spaces. Also, while you *can* run any code, you don’t get advanced controls over the server (it’s managed for you). If your app needs long-running background processes or a database on the same host, that might be tricky on Spaces. But for an interactive agent relying on external DB/LLM, Spaces is perfect.

## Option 2: Deploy on Streamlit Community Cloud

Another beginner-friendly option is **Streamlit Community Cloud** (formerly Streamlit Sharing). This is the official hosting by Streamlit, where you can deploy a Streamlit app directly from a GitHub repo[[17]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Fstreamlit,1%20hr%20idle). It’s also free for public apps. The workflow is: push your code to GitHub, then Streamlit Cloud pulls that repo and runs the app. Given you have some GitHub experience, this could be straightforward.

**Steps to Deploy on Streamlit Cloud:**

1. **GitHub Repo:** Ensure your project is on GitHub. If not, create a new GitHub repository and upload your files (app.py, requirements.txt, etc.)[[20]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Create%20a%20GitHub%20account%20here%3A,com)[[21]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=4,Repository%20by%20Drag%20and%20Drop). You can do this via Git (commit and push) or even drag-and-drop on GitHub’s interface[[22]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=4,Repository%20by%20Drag%20and%20Drop). The repository can be public (required for free tier) so others can see your code.
2. **Streamlit Cloud Account:** Sign up for Streamlit Community Cloud using your email or GitHub login[[23]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=4,Cloud%20Account). It’s free. Once logged in, find the option to **“New app”**[[24]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,the%20account%20you%20already%20made).
3. **Link GitHub and Deploy:** Click “New app” and follow the prompts to connect your GitHub account (you may need to authorize Streamlit to access your repos)[[24]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,the%20account%20you%20already%20made). Then select the repository and branch where your Streamlit app lives[[25]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,Click%20on%20Deploy). Specify the main file path (e.g. app.py). Finally, click **Deploy**. Streamlit will launch a server, install the requirements, and run your app.
4. **Set Secrets:** In Streamlit Cloud, you can define secrets by going to the app’s settings on the cloud dashboard. Add your API keys or DB connection strings there (as key-value pairs). Streamlit will make them available via st.secrets["YOUR\_KEY"] in your code, or as environment variables[[3]](https://docs.streamlit.io/deploy/streamlit-community-cloud/deploy-your-app/secrets-management#:~:text=Docs%20docs,your%20code%20to%20read%20secrets). For example, you can set OPENAI\_API\_KEY in the secrets, and in your app do openai.api\_key = st.secrets["OPENAI\_API\_KEY"]. This keeps sensitive info out of the public repo.
5. **Access Your App:** Once deployed, Streamlit gives you a URL like https://<username>-<project-name>.streamlit.app. Visit this URL to use your AI agent. Streamlit Cloud tends to have fast startup and low latency for apps[[18]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Streamlit%20Cloud), so your app should feel responsive. If no one uses it for some time (an hour of inactivity), the app might sleep, but it will wake on access (with a few seconds delay to cold-start).

**Pros & Cons (Streamlit Cloud):**

* ✅ **Easy GitHub integration:** You just push code to GitHub and let Streamlit Cloud handle the rest[[17]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Fstreamlit,1%20hr%20idle). Any updates to the main branch can auto-trigger redeploy, making iterative development easy.
* ✅ **Optimized for Streamlit:** Since you’re using Streamlit, this platform is literally built for that. You don’t need to worry about running a separate web server or container. Also, Streamlit Cloud often loads apps faster (after initial deploy) compared to some other free hosts[[18]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Streamlit%20Cloud).
* ✅ **Free Tier for Public Apps:** You can deploy unlimited public apps for free[[26]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=plain%20Python,paid). This is ideal for a capstone project showcase. (Private apps or higher resources would require a paid plan or the Streamlit Team edition, but likely unnecessary for now.)
* ⚠️ **Public Code:** Like HF Spaces, the free tier requires the repository to be public. This means your source code is visible (though casual users of the app only see the interface, not the code). If your capstone code is okay to open-source, this is not a problem. Otherwise, to keep code private you’d need to upgrade or use a different platform (e.g. Heroku or AWS)[[27]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Streamlit%20Cloud%20offers%20the%20quickest,App%20is%20a%20viable%20alternative).
* ⚠️ **Resource Limits:** Streamlit Cloud’s free resources are sufficient for many apps, but not as high as HF Spaces. There’s a limit on RAM and runtime per app (and apps sleeping when idle)[[28]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=The%20free%20tier%20allows%20unlimited,paid)[[29]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=can%20make%20a%20simple%20form,paid). Very large models or very long-running processes might not be suitable. Given your agent uses external APIs for heavy work, you should be well within limits.
* ⚠️ **Python-only:** Streamlit Cloud is primarily for Python apps. If you have a Node.js component or any non-Python service, Streamlit Cloud wouldn’t directly host that. You’d host the Python part (agent logic/UI) here, and perhaps host a Node frontend elsewhere if needed. Most likely, you won’t need Node for UI if you use Streamlit. (If you do need Node, consider other platforms below.)

## Option 3: Deploy on AWS (Elastic Beanstalk)

For a more **robust, production-grade deployment**, you can use Amazon Web Services. AWS offers fine-grained control and the ability to keep your code and data completely private. A beginner-friendly AWS service for web apps is **Elastic Beanstalk (EB)**, which abstracts a lot of the DevOps. With EB, you can deploy a Python FastAPI app or even a Node.js app by just uploading your code or linking a Git repo, and AWS will handle provisioning servers, load balancers, and connecting to a database[[30]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=AWS%20Elastic%20Beanstalk%20,It%20supports%20applications%20developed%20in)[[31]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=There%27s%20no%20additional%20charge%20for,resources%20that%20your%20application%20consumes). Essentially, EB orchestrates EC2 (virtual machines) and other services under the hood but presents a simple interface to you.

*Figure: Example architecture of an AWS Elastic Beanstalk environment. EB will provision EC2 instances (virtual servers) to run your app, an optional load balancer for scaling traffic, and can integrate with an RDS database (e.g. PostgreSQL) for persistence*[*[30]*](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=AWS%20Elastic%20Beanstalk%20,It%20supports%20applications%20developed%20in)[*[32]*](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=Create%20a%20DB%20with%20the,Apply)*. In our scenario, the app server would handle API calls or a web UI, and interact with external services (OpenAI API, vector DB). The heavy LLM workload is offloaded to OpenAI, so a small EC2 instance can handle the rest.*

**Steps to Deploy on AWS Elastic Beanstalk:**

1. **AWS Account Setup:** If you don’t have one, sign up for AWS and log in to the AWS Management Console. AWS offers a **Free Tier** for new accounts (12 months of certain free resources)[[33]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=Elastic%20Beanstalk%20CLI). For example, you can get a t2.micro EC2 instance and a small database instance free for a year, which is sufficient for testing deployment[[32]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=Create%20a%20DB%20with%20the,Apply)[[34]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=,to%20review%20AWS%20PostgreSQL%20pricing). Be mindful to stay within free tier specs (e.g. select only t2.micro sizes) to avoid charges[[35]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=,Password%3A%20pick%20a%20strong%20password).
2. **Package Your Application:** Decide how you will run your app on AWS:
3. If it’s a **FastAPI** or Flask app, you should have an entry point (like main.py) that launches the app (probably via Uvicorn or Gunicorn for FastAPI). Elastic Beanstalk can deploy this directly using the Python runtime.
4. If it’s a **Streamlit** app, deploying on EB is less common (Streamlit is usually deployed on Streamlit Cloud or similar). It is possible, but typically you’d run streamlit run on an EC2 instance. A simpler approach on AWS for a Streamlit UI would be using a container service or just using HF/Streamlit Cloud. So here, let’s assume you either have an **API (FastAPI)** for the agent, or you containerize the Streamlit app. (FastAPI is great if you plan a separate front-end or integration; Streamlit is great for a standalone UI. Either can be deployed on EB.)
5. If you have a **Node.js** component (say, a front-end or a bot server), you can deploy that with EB as well by selecting a Node.js platform. You might even deploy it separately from the Python API and have them communicate (though for simplicity, many capstone projects stick to one backend).
6. **Create an Elastic Beanstalk Environment:** In AWS Console, go to Elastic Beanstalk service. Click “Create New Application,” give it a name (e.g. “my-ai-agent-app”). Then create an **Environment** (Web Server environment). Choose the appropriate platform: for a FastAPI app use *Python* platform (choose the latest Python version EB supports), or *Node.js* if deploying a Node app[[36]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=deploying%20and%20scaling%20web%20applications,that%20EB%20doesn%27t%20currently%20support). EB will ask for your code:
7. You can upload a **ZIP file** of your application source code. For a Python FastAPI, this zip would include your .py files, requirements.txt, perhaps a Procfile or application.py telling EB how to start the app (EB can often detect it if you name the main file application.py or have a Procfile with the start command).
8. Alternatively, you can use the **EB CLI** to deploy (after initializing with eb init). For beginners, the web console or AWS CLI with simple commands may be easier. For example, using EB CLI: eb init -p python-3.9 my-ai-agent-app (initializes the project) then eb create my-ai-agent-env (deploys environment)[[37]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=,Configure%20RDS)[[38]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=%24%20git%20clone%20%5Bemail%20protected%5D%3Aduplxey%2Ffastapi,songs). The TestDriven tutorial provides step-by-step for deploying a FastAPI app with EB CLI[[39]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=In%20this%20tutorial%2C%20we%27ll%20walk,application%20to%20AWS%20Elastic%20Beanstalk)[[40]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=We%27ll%20be%20deploying%20a%20simple,songs%20in%20this%20tutorial).
9. **Database Setup (if needed):** Since you have a local PostgreSQL with pgvector, you likely want a cloud database. AWS’s solution is **RDS (Relational Database Service)**. You can have EB create an RDS PostgreSQL instance and attach it to your environment. In the EB console, under your environment’s **Configuration**, add a Database and choose PostgreSQL (set username/password)[[41]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=Click%20,Edit)[[42]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=,Password%3A%20pick%20a%20strong%20password). EB will launch the DB and provide the connection info to your app automatically via environment variables[[5]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=After%20the%20environment%20update%20is,credentials%20to%20our%20FastAPI%20app). Specifically, variables like RDS\_HOSTNAME, RDS\_USERNAME, RDS\_PASSWORD, etc. will be injected. Your app should read these to configure the database connection[[43]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=We%20can%20now%20use%20these,with%20the%20following)[[44]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=username%3Dos.environ,else%3A%20DATABASE_URL%20%3D). For example, instead of connecting to localhost, use those env vars to connect to the AWS host[[45]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=if%20%27RDS_DB_NAME%27%20in%20os,). This way, you swap out your local Postgres for the cloud one seamlessly (you may need to run migrations or setup the vector extension on the new DB – PGVector extension can be enabled on Amazon RDS by using a PostgreSQL version that supports it and executing CREATE EXTENSION pgvector, since AWS RDS Postgres allows certain extensions).
10. *Alternative:* If managing RDS is too much, you could use a hosted vector DB service (like Pinecone, Weaviate Cloud, or Supabase) and just have your app call that. It would be similar to calling OpenAI – just another API. This avoids running a DB inside AWS for now. But using RDS via EB is a good learning experience and keeps all pieces in AWS.
11. **Environment Variables and Secrets:** In addition to the DB creds which EB sets, you should add your own config vars (like the OpenAI API key). In EB, you can set environment variables either through the EB CLI or in the console under Configuration > Software > Environment Properties. Add a property OPENAI\_API\_KEY (for example) and its value. When the app runs on the EC2, it will have that in its env, so your code os.getenv("OPENAI\_API\_KEY") will retrieve it[[5]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=After%20the%20environment%20update%20is,credentials%20to%20our%20FastAPI%20app). This keeps secrets out of your codebase. Do similar for any other API keys or settings.
12. **Deploy and Test:** If you used the EB web console, you would upload the source bundle and click Deploy. If using CLI, eb deploy will push the latest code to AWS[[46]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=Commit%20the%20changes%20to%20git,and%20deploy). EB will take a few minutes to set up resources (EC2, security groups, load balancer if any, RDS if added). Once it’s done, you’ll get a **URL** for your app, something like http://my-ai-agent-app-env.eba-xyz123.us-west-2.elasticbeanstalk.com. Visit this URL – it should be running your FastAPI (you’ll see the interactive docs if it’s a REST API) or your Streamlit app (if you managed to deploy that). Test your agent’s functionality.
13. **(Optional) Domain and HTTPS:** If this is a long-running project, you might want a custom domain and SSL. AWS EB can help here: you’d buy a domain (AWS Route53 or any registrar), point it to the EB environment’s URL, and provision an SSL certificate via AWS Certificate Manager. EB can attach the certificate to the load balancer so your app runs on https://[[47]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=To%20serve%20your%20application%20via,HTTPS%2C%20we%27ll%20need%20to)[[48]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=1.%20Port%20,certificate%20that%20you%20just%20created). This is more of a “nice to have” for production. For a capstone demo, using the default EB URL with HTTP is usually fine (or you can use a free SSL service or simply trust the AWS domain).
14. **Monitoring and Scaling:** AWS will keep your app running 24/7. EB dashboards show health metrics. If needed, you can scale up the instance size or count. For example, if your agent became popular, you could increase the EC2 instance type or allow auto-scaling to multiple instances. Elastic Beanstalk makes this fairly easy to configure. By default, a single t2.micro (1 vCPU, ~1GB RAM) can handle a modest load, especially since the real work (OpenAI calls) happens outside. *Cost note:* one t2.micro and a small RDS can be free tier or just a few dollars a month after free tier. Always shut down resources when not needed to avoid surprise costs.

**Pros & Cons (AWS Elastic Beanstalk):**

* ✅ **Full Control & Privacy:** Your app runs on your own cloud instance. The code isn’t exposed to the public (unlike on Spaces/Streamlit Cloud). You can keep the repository private and still deploy. You also have flexibility to install system packages, use specific Python versions, etc. EB supports many languages and even Docker if needed[[36]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=deploying%20and%20scaling%20web%20applications,that%20EB%20doesn%27t%20currently%20support).
* ✅ **Integrated Services:** Need a database? EB can provision RDS and pass connection vars automatically[[5]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=After%20the%20environment%20update%20is,credentials%20to%20our%20FastAPI%20app). Need object storage? You can use S3. Essentially, it’s easy to plug into the whole AWS ecosystem, which is useful if your agent grows in complexity.
* ✅ **Scalability and Custom Domain:** As a more “production” solution, you can add a custom domain, HTTPS, scale out to multiple servers, add monitoring, etc., all within AWS tools[[47]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=To%20serve%20your%20application%20via,HTTPS%2C%20we%27ll%20need%20to)[[48]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=1.%20Port%20,certificate%20that%20you%20just%20created). This would impress as an engineering solution beyond a basic demo.
* ⚠️ **Higher Complexity:** There is a learning curve. You have to understand AWS basics (EC2, RDS, environment variables) and possibly troubleshoot deployment logs if something goes wrong. EB simplifies a lot, but it’s still more involved than pushing to HF or Streamlit Cloud. For instance, you might need to adjust your code for a different environment (e.g., ensure the app listens on the correct port as AWS expects, which is usually handled if you use standard frameworks).
* ⚠️ **Cost Management:** While there is a free tier, there’s a risk of incurring charges if you pick a larger instance or forget to terminate resources. Always double-check the settings (e.g., use t2.micro as recommended[[35]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=,Password%3A%20pick%20a%20strong%20password)). AWS is pay-as-you-go, so a small app that doesn’t consume much beyond free tier will cost very little, but it’s not “100% free” forever like the limited community platforms. For a short-term capstone project, the free tier is likely enough[[34]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=,to%20review%20AWS%20PostgreSQL%20pricing).
* ⚠️ **No Native UI for Streamlit:** If your app is a Streamlit UI, AWS EB isn’t the typical deployment path (though you *could* deploy it by running streamlit on an EC2, it’s a bit hacky). EB is best for API services or web backends. If you want a quick web UI, HF Spaces or Streamlit Cloud are far simpler. On AWS, an alternative for a UI would be to use a service like AWS Amplify or Lightsail with a container, but those also add complexity. Thus, consider AWS primarily if you have a FastAPI/Node service or if you require the control for your project’s criteria.

## Other Deployment Alternatives 🚀

Beyond the three options above, there are other platforms you might consider, each with its own trade-offs. Here’s a brief comparison:

* **Heroku (Cloud Application Platform):** Heroku was a popular PaaS for deploying apps with a Git push. It’s very beginner-friendly (similar to Render/Railway below) and supports Python, Node, etc. However, as of late 2022, Heroku’s free tier was discontinued – it now requires payment for continuous hosting. If you have credits or don’t mind a small cost, Heroku could be used (the workflow: commit code, connect the GitHub repo to Heroku or use Heroku CLI to push, and the app will run). Heroku offers easy environment variable management and can provision add-ons like a Postgres database. In the Medium article we saw, the author suggests Heroku as a viable alternative if you need a **private app** (since HF and Streamlit free tiers require public code)[[18]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Streamlit%20Cloud). So, if keeping your code hidden is a priority and you want ease, a small Heroku Dyno might be worth it. Otherwise, free options like Render are available.
* **Render and Railway (Free Hobby Tiers):** Render.com and Railway.app are modern cloud platforms that deploy apps directly from your GitHub. They are often cited as **Heroku alternatives** with generous free plans[[49]](https://medium.com/@connect.hashblock/i-used-streamlit-to-build-a-real-time-ai-tool-heres-what-i-learned-5bb829c69cad#:~:text=Image%3A%20Hash%20Block)[[50]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=3). For example, **Render** can deploy a web service from your repo – you specify a start command (like uvicorn main:app) and it takes care of building and running. Render’s free tier gives 750 hours/month of a micro instance (enough to run continuously) and 1 GB of RAM[[51]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Frender,support%20is%20only%20on%20paid). **Railway** gives some free credits (e.g. $5/month equivalent) and is also very easy to use[[52]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Frailway,You%20can%20add%20authentication). Both support adding environment variables through a web UI, and you can add a database (Railway can spin up a Postgres instance for you as an add-on, for example). The process with these is: login, link your GitHub repo, choose the branch, and it auto-deploys. These platforms are great if you have a **FastAPI or Node backend** to host. Many developers use them to host Discord bots, small APIs, or ML demos. The downside is the free tier apps may sleep on inactivity (Render will sleep after 15 minutes idle by default)[[53]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=can%20deploy%20web%20services%20%28e,support%20is%20only%20on%20paid), and you might need to re-deploy if you exceed free usage. But for a capstone demo, they are cost-free and quite convenient.
* **Hugging Face Inference Endpoints:** Since you mentioned using OpenAI and also having Hugging Face tools, note that Hugging Face offers a service to deploy your own models as managed endpoints. This is more if you had a custom model to host (which you likely don’t, since you use OpenAI and transformers locally). In our context, Inference Endpoints aren’t needed – we are deploying the whole app interface. Just for completeness: HF Inference Endpoints could host a Transformer model behind an API, which you could then call from your app (outsourcing the model inference to HF’s infrastructure). But this is a paid service for production use; we assume you’ll stick to OpenAI API or similar, as stated.
* **Docker Containers on Cloud Services:** If you’re familiar with Docker, you could containerize your application and run it on various services: **Google Cloud Run** (which has a free tier and can run containers serverlessly), **Azure Container Instances**, **AWS ECS/Fargate**, or even **DigitalOcean App Platform**. For instance, Google Cloud Run is developer-friendly and free for a certain amount of CPU-time[[54]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Each%20major%20cloud%20,up%20cloud%20accounts%20and%20environments). You’d build a Docker image for your app (including possibly both your FastAPI and a Node frontend in one image, or separate). However, introducing Docker and CI/CD might be overkill for your first deployment. If you do have containerization experience, Cloud Run or ECS with Fargate can be efficient and scalable solutions. Otherwise, platforms like Render essentially do the containerization for you in the background.
* **Other ML Demo Platforms:** There are niche options like **Gradio Spaces** (similar to Streamlit on HF), or **PythonAnywhere** (lets you run small Python web apps easily, though with limitations)[[55]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=4). Even Jupyter-based solutions like **Binder** or **Kaggle**/**Colab** notebooks can *share* an interactive environment, but they aren’t true deployments – the user would see code or have to run cells. For an interactive chatbot, those aren’t ideal. Given the tools you have, the earlier options cover the best mix of simplicity and functionality.

## Comparison Summary 📝

To help you choose, here’s a quick comparison of the platforms:

* **Hugging Face Spaces** – *Best for quick ML demos with Streamlit/Gradio.* **Ease:** Very high (just git push to deploy)[[12]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=3,your%20New%20Hugging%20Face%20Space)[[15]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=cd%20name_of_your_new_folder%20git%20add%20,git%20push). **Cost:** Free (public). **Scalability:** Meant for demos (one instance, no scaling, but generous RAM). **Code visibility:** Public by default. **When to use:** Perfect for showcasing your capstone project with minimal DevOps work, especially if you’ve built a nice Streamlit UI.
* **Streamlit Community Cloud** – *Best for pure Streamlit apps, tightly integrated with GitHub.* **Ease:** Very high[[24]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,the%20account%20you%20already%20made)[[25]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,Click%20on%20Deploy). **Cost:** Free (public). **Performance:** Fast load times for UI[[18]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Streamlit%20Cloud); sleeps on idle. **Limitations:** Public code, primarily Python/Streamlit only. **When to use:** If your project is a Streamlit app and you want a quick live demo without worrying about servers.
* **Render / Railway (PaaS)** – *Best for custom web services (FastAPI, Node) on a budget.* **Ease:** High (deploy from GitHub, configure env vars in dashboard). **Cost:** Free hobby tiers (with usage limits)[[51]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Frender,support%20is%20only%20on%20paid)[[52]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Frailway,You%20can%20add%20authentication). **Flexibility:** Can run any backend, add databases. **When to use:** If you need your app running as an API or need to deploy a Node.js service in addition to the Python logic. Also good if you want the app not to be easily visible to others (you can keep repo private and just supply code to Render).
* **AWS (Elastic Beanstalk or similar)** – *Best for full control and learning industry-level deployment.* **Ease:** Medium/Low (must handle AWS setup, but EB makes it easier than raw EC2)[[30]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=AWS%20Elastic%20Beanstalk%20,It%20supports%20applications%20developed%20in). **Cost:** Potentially free for small usage (within free tier)[[35]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=,Password%3A%20pick%20a%20strong%20password), but verify. **Scalability:** High (designed for production scaling, multiple instances, etc.). **Code visibility:** Private. **When to use:** If your capstone requires deploying on a cloud provider or you want experience with AWS. Also if you need to integrate with other AWS services or keep everything under your control. Great for an engineering report to mention that you deployed on AWS, though more effort is needed to set up.
* **Node-Specific Note:** If your project *requires* a Node.js front-end or bot, you might deploy the Node app on a service like Vercel (good for front-ends/Next.js, but also supports Node functions) or on Render as mentioned. Vercel is very easy for front-end projects linked to GitHub. However, for an AI agent, usually a simple UI in Streamlit suffices, so you may not need a separate Node deployment unless you’re integrating with a web frontend or messenger platform.

In conclusion, for passing your AI Engineering capstone, **a two-pronged approach could work well**:

1. **Deploy the demo quickly on a free platform (HF Spaces or Streamlit Cloud)** so that your professors/colleagues can easily try the agent via a web interface. This addresses the requirement of a working, accessible prototype.
2. **Explain a plan for a production-grade deployment (AWS)** in your report, highlighting that major processing is outsourced (OpenAI API, etc.) and demonstrating you understand how to scale or customize the solution. You might even perform a basic EB deployment to show it’s running on AWS, if time permits.

This way, you show both practical delivery and engineering depth. Each platform has its strengths, so choose based on what aspect you want to emphasize. Good luck with your deployment! 🚀

**Sources:**

* A16Z, *“Emerging Architectures for LLM Applications”* – reference for typical LLM app stack (hosted APIs, vector DBs, etc.)[[1]](https://a16z.com/emerging-architectures-for-llm-applications/#:~:text=There%20are%20many%20different%20ways,possible%20now%20with%20foundation%20models)[[2]](https://a16z.com/emerging-architectures-for-llm-applications/#:~:text=,offerings%20like%20those%20listed%20above).
* *KDnuggets:* “7 Best Free Platforms to Host ML Models” – covers Hugging Face Spaces, Streamlit Cloud, Render, Railway, and cloud free tiers[[6]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Fhuggingface,has%20a%20free%20tier%20for)[[17]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Fstreamlit,1%20hr%20idle)[[51]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Frender,support%20is%20only%20on%20paid)[[52]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Frailway,You%20can%20add%20authentication)[[56]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=are%20excellent%20for%20ML%20models,Ultimately%2C%20the%20best%20platform).
* Anuj Karn, *“Deploy Streamlit Web Apps to HuggingFace and Streamlit Cloud”* – step-by-step guide for deploying a Streamlit app on HF Spaces[[12]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=3,your%20New%20Hugging%20Face%20Space)[[15]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=cd%20name_of_your_new_folder%20git%20add%20,git%20push) and Streamlit Cloud[[24]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,the%20account%20you%20already%20made)[[25]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,Click%20on%20Deploy)[[18]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Streamlit%20Cloud).
* Hugging Face Docs – *Managing Spaces* (how to add secrets and hardware settings on HF Spaces)[[4]](https://huggingface.co/docs/huggingface_hub/en/guides/manage-spaces#:~:text=%3E%3E%3E%20api.delete_space_variable%28repo_id%3Drepo_id%2C%20key%3D).
* Nik Tomazic (TestDriven.io), *“Deploying a FastAPI Application to Elastic Beanstalk”* – explains Elastic Beanstalk concepts and integration with AWS services[[30]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=AWS%20Elastic%20Beanstalk%20,It%20supports%20applications%20developed%20in)[[31]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=There%27s%20no%20additional%20charge%20for,resources%20that%20your%20application%20consumes) and how EB provides database credentials to the app (RDS integration)[[5]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=After%20the%20environment%20update%20is,credentials%20to%20our%20FastAPI%20app)[[45]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=if%20%27RDS_DB_NAME%27%20in%20os,).
* AWS EB Tutorial – details on setting up a Postgres RDS in EB and staying within free tier limits[[32]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=Create%20a%20DB%20with%20the,Apply)[[34]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=,to%20review%20AWS%20PostgreSQL%20pricing).

[[1]](https://a16z.com/emerging-architectures-for-llm-applications/#:~:text=There%20are%20many%20different%20ways,possible%20now%20with%20foundation%20models) [[2]](https://a16z.com/emerging-architectures-for-llm-applications/#:~:text=,offerings%20like%20those%20listed%20above) Emerging Architectures for LLM Applications | Andreessen Horowitz

<https://a16z.com/emerging-architectures-for-llm-applications/>

[[3]](https://docs.streamlit.io/deploy/streamlit-community-cloud/deploy-your-app/secrets-management#:~:text=Docs%20docs,your%20code%20to%20read%20secrets) Secrets management for your Community Cloud app - Streamlit Docs

<https://docs.streamlit.io/deploy/streamlit-community-cloud/deploy-your-app/secrets-management>

[[4]](https://huggingface.co/docs/huggingface_hub/en/guides/manage-spaces#:~:text=%3E%3E%3E%20api.delete_space_variable%28repo_id%3Drepo_id%2C%20key%3D) [[16]](https://huggingface.co/docs/huggingface_hub/en/guides/manage-spaces#:~:text=3) Manage your Space

<https://huggingface.co/docs/huggingface_hub/en/guides/manage-spaces>

[[5]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=After%20the%20environment%20update%20is,credentials%20to%20our%20FastAPI%20app) [[30]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=AWS%20Elastic%20Beanstalk%20,It%20supports%20applications%20developed%20in) [[31]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=There%27s%20no%20additional%20charge%20for,resources%20that%20your%20application%20consumes) [[32]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=Create%20a%20DB%20with%20the,Apply) [[33]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=Elastic%20Beanstalk%20CLI) [[34]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=,to%20review%20AWS%20PostgreSQL%20pricing) [[35]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=,Password%3A%20pick%20a%20strong%20password) [[36]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=deploying%20and%20scaling%20web%20applications,that%20EB%20doesn%27t%20currently%20support) [[37]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=,Configure%20RDS) [[38]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=%24%20git%20clone%20%5Bemail%20protected%5D%3Aduplxey%2Ffastapi,songs) [[39]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=In%20this%20tutorial%2C%20we%27ll%20walk,application%20to%20AWS%20Elastic%20Beanstalk) [[40]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=We%27ll%20be%20deploying%20a%20simple,songs%20in%20this%20tutorial) [[41]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=Click%20,Edit) [[42]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=,Password%3A%20pick%20a%20strong%20password) [[43]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=We%20can%20now%20use%20these,with%20the%20following) [[44]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=username%3Dos.environ,else%3A%20DATABASE_URL%20%3D) [[45]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=if%20%27RDS_DB_NAME%27%20in%20os,) [[46]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=Commit%20the%20changes%20to%20git,and%20deploy) [[47]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=To%20serve%20your%20application%20via,HTTPS%2C%20we%27ll%20need%20to) [[48]](https://testdriven.io/blog/fastapi-elastic-beanstalk/#:~:text=1.%20Port%20,certificate%20that%20you%20just%20created) Deploying a FastAPI Application to Elastic Beanstalk | TestDriven.io

<https://testdriven.io/blog/fastapi-elastic-beanstalk/>

[[6]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Fhuggingface,has%20a%20free%20tier%20for) [[7]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=code%2C%20and%20Spaces%20lets%20you,Overall%2C%20Hugging%20Face%20is%20beginner) [[17]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Fstreamlit,1%20hr%20idle) [[26]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=plain%20Python,paid) [[28]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=The%20free%20tier%20allows%20unlimited,paid) [[29]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=can%20make%20a%20simple%20form,paid) [[50]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=3) [[51]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Frender,support%20is%20only%20on%20paid) [[52]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Link%3A%20https%3A%2F%2Frailway,You%20can%20add%20authentication) [[53]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=can%20deploy%20web%20services%20%28e,support%20is%20only%20on%20paid) [[54]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=Each%20major%20cloud%20,up%20cloud%20accounts%20and%20environments) [[55]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=4) [[56]](https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models#:~:text=are%20excellent%20for%20ML%20models,Ultimately%2C%20the%20best%20platform) 7 Best FREE Platforms to Host Machine Learning Models - KDnuggets

<https://www.kdnuggets.com/7-best-free-platforms-to-host-machine-learning-models>

[[8]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=you%20to%20run%20git%20commands,app%20files%20to%20the%20web) [[9]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=3,Face%20Account) [[10]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=3) [[11]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,Hit%20the%20%E2%80%98Create%20space%E2%80%99%20button) [[12]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=3,your%20New%20Hugging%20Face%20Space) [[13]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,to%20your%20Hugging%20Face%20Space) [[14]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,to%20your%20Hugging%20Face%20Space) [[15]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=cd%20name_of_your_new_folder%20git%20add%20,git%20push) [[18]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Streamlit%20Cloud) [[19]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Hugging%20Face%20Spaces) [[20]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Create%20a%20GitHub%20account%20here%3A,com) [[21]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=4,Repository%20by%20Drag%20and%20Drop) [[22]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=4,Repository%20by%20Drag%20and%20Drop) [[23]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=4,Cloud%20Account) [[24]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,the%20account%20you%20already%20made) [[25]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=,Click%20on%20Deploy) [[27]](https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2#:~:text=Streamlit%20Cloud%20offers%20the%20quickest,App%20is%20a%20viable%20alternative) Deploy Streamlit Web Apps to the Huggingface and Streamlit Cloud | by Anuj Karn | Towards Data Engineering | Medium

<https://medium.com/towards-data-engineering/deploy-streamlit-web-apps-to-the-huggingface-and-streamlit-cloud-b5c7506e70e2>

[[49]](https://medium.com/@connect.hashblock/i-used-streamlit-to-build-a-real-time-ai-tool-heres-what-i-learned-5bb829c69cad#:~:text=Image%3A%20Hash%20Block) I Used Streamlit to Build a Real-Time AI Tool — Here’s What I Learned | by Hash Block | Jul, 2025 | Medium

<https://medium.com/@connect.hashblock/i-used-streamlit-to-build-a-real-time-ai-tool-heres-what-i-learned-5bb829c69cad>