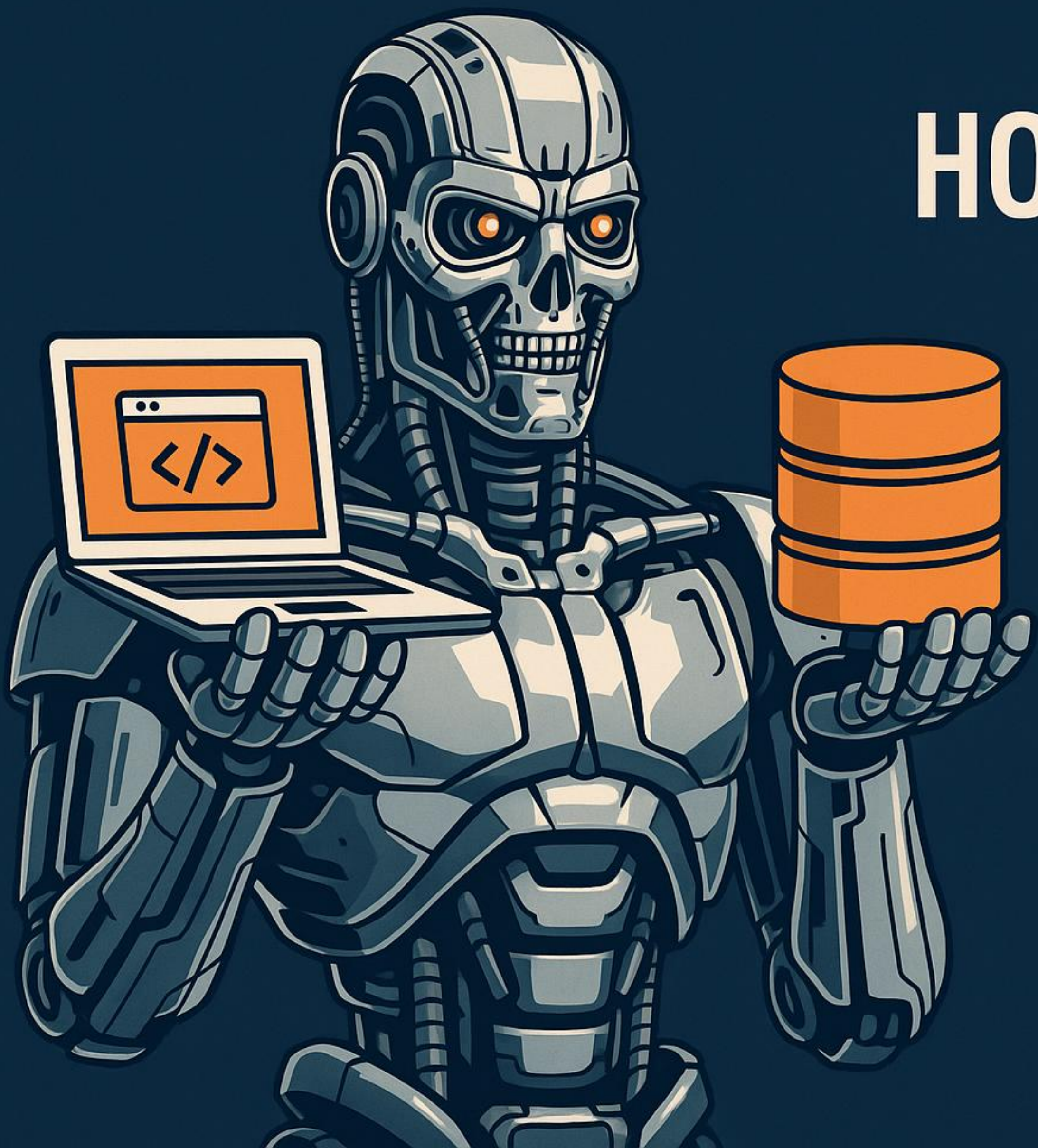


HOSTING APPS AND USING DATABASES

Overview

- Pushing app code to GitHub
- Hosting apps online
- Add a database to an app

GITHUB

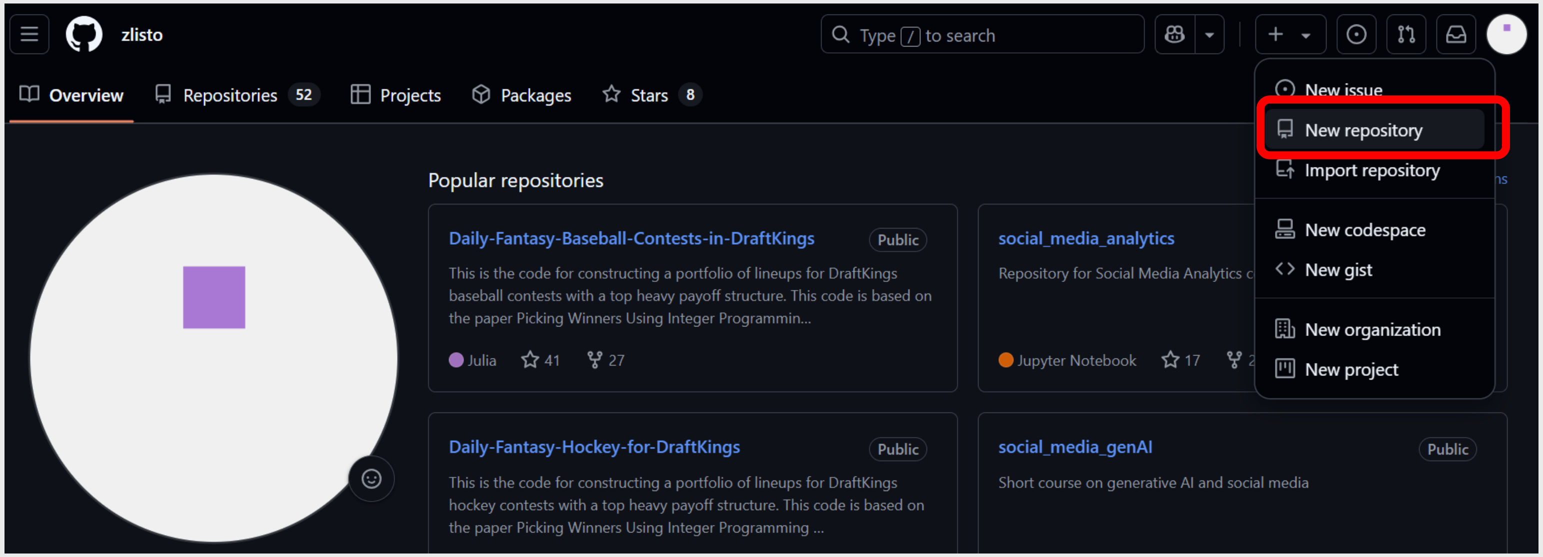
Test App Locally

- Once your code is done, run the app on your machine to make sure it works
  - Streamlit: streamlit run app.py
  - React: npm start

- If everything is working, you are ready to push it to GitHub

Create GitHub Repo

- Create a “New repository” (repo) on GitHub



Daily-Fantasy-Baseball-Contests-in-DraftKings Public
This is the code for constructing a portfolio of lineups for DraftKings baseball contests with a top heavy payoff structure. This code is based on the paper Picking Winners Using Integer Programmin...
Julia 41 27

social_media_analytics
Repository for Social Media Analytics c
Jupyter Notebook 17

Daily-Fantasy-Hockey-for-DraftKings Public
This is the code for constructing a portfolio of lineups for DraftKings hockey contests with a top heavy payoff structure. This code is based on the paper Picking Winners Using Integer Programming ...

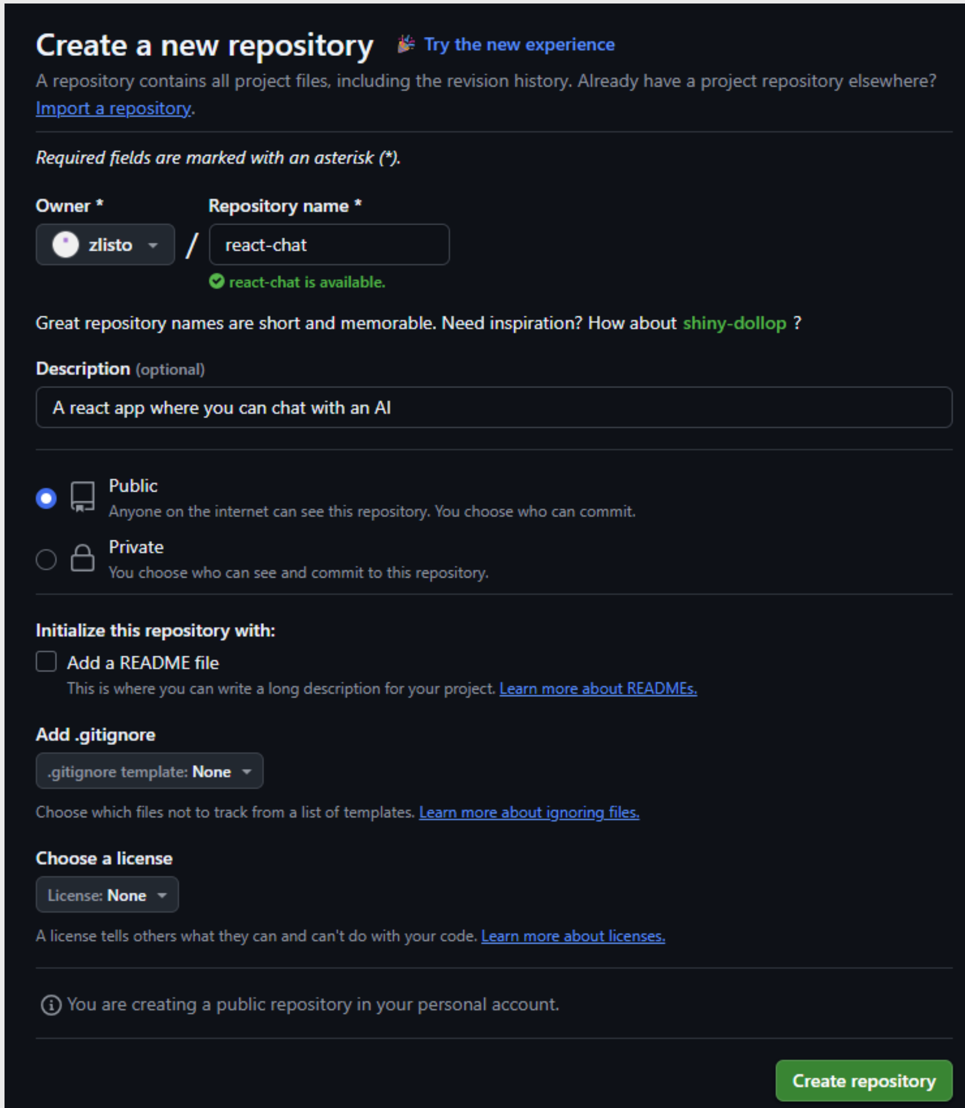
social_media_genAI Public
Short course on generative AI and social media

Create GitHub Repo

- Choose a name for the repo
- Choose public or private
  - Private if the code will make you rich one day
- Don’t add a README file
- Don’t add a .gitignore file
- Don’t choose a license

- We want an empty repo so we can push all our code to it

- AI will make these files for us



Create a new repository

Try the new experience

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner *: zlisto / Repository name *: react-chat

react-chat is available.

Great repository names are short and memorable. Need inspiration? How about shiny-dollop ?

Description (optional): A react app where you can chat with an AI

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private — You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file — This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore: .gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license: License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

You are creating a public repository in your personal account.

Initial Commit to Empty GitHub Repo

💻 TERMINAL WAY

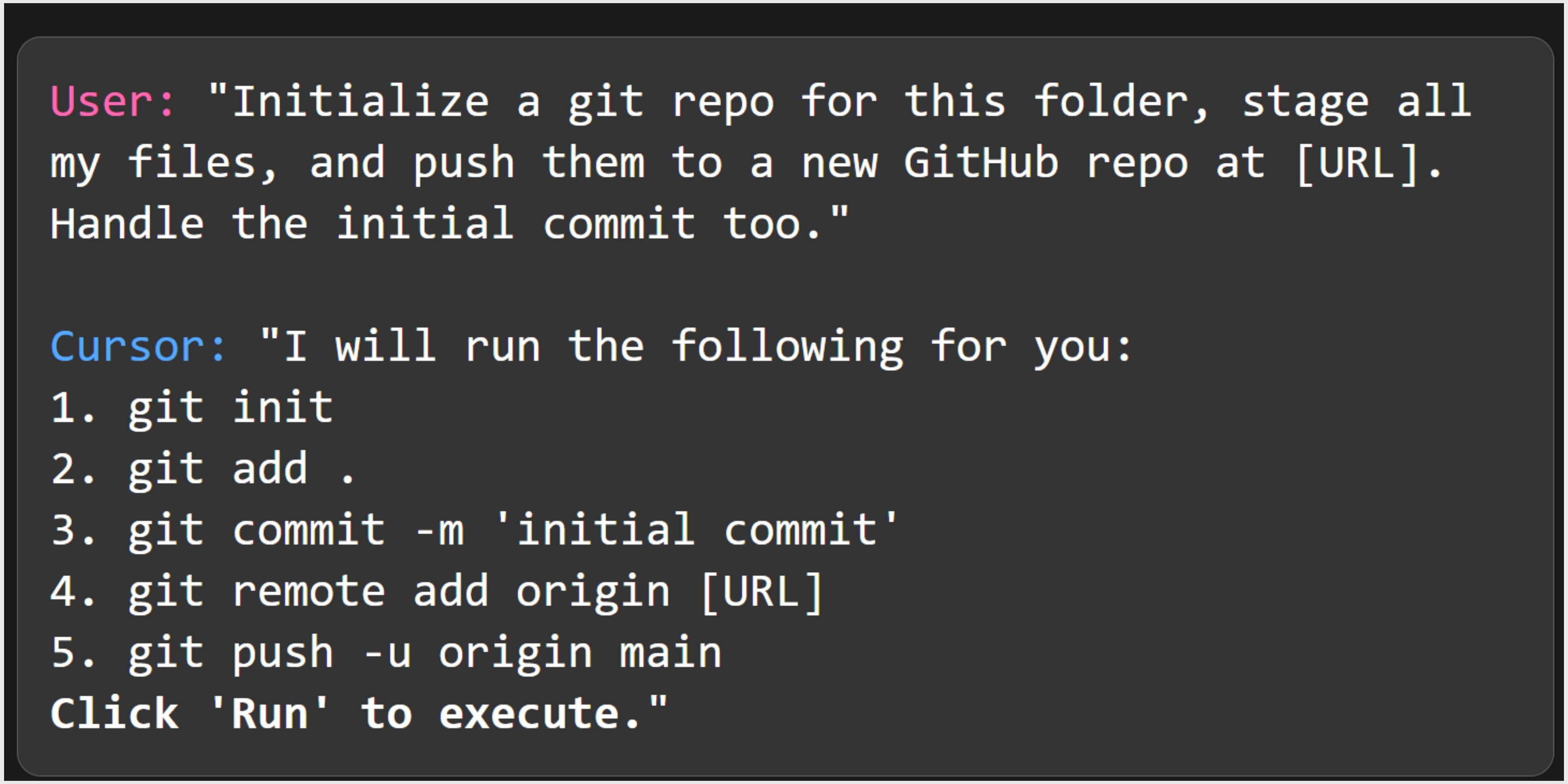
```
git init
git add .
git commit -m "initial vibe"
# Connect to the cloud:
git remote add origin [git URL]
git push -u origin main
```

✨ CURSOR WAY

1. **Initialize:** Source Control Tab → "Initialize Repository"
2. **Stage & Commit:** Click **+** then type your message and Commit.
3. **Add Remote:**
    - Click **... (Three Dots)** in Git sidebar.
    - Select **Remote** → **Add Remote**.
    - Name: `origin`.
    - URL: `https://github.com/<user>/<repo>.git`.
4. **Push:** Click **Publish Branch.**

💡 **Pro-Tip:** The URL always follows the format: `https://github.com/YOUR_USERNAME/YOUR_PROJECT_NAME.git`.

Initial Commit with AI



```
User: "Initialize a git repo for this folder, stage all
my files, and push them to a new GitHub repo at [URL].
Handle the initial commit too."

Cursor: "I will run the following for you:
1. git init
2. git add .
3. git commit -m 'initial commit'
4. git remote add origin [URL]
5. git push -u origin main
Click 'Run' to execute."
```

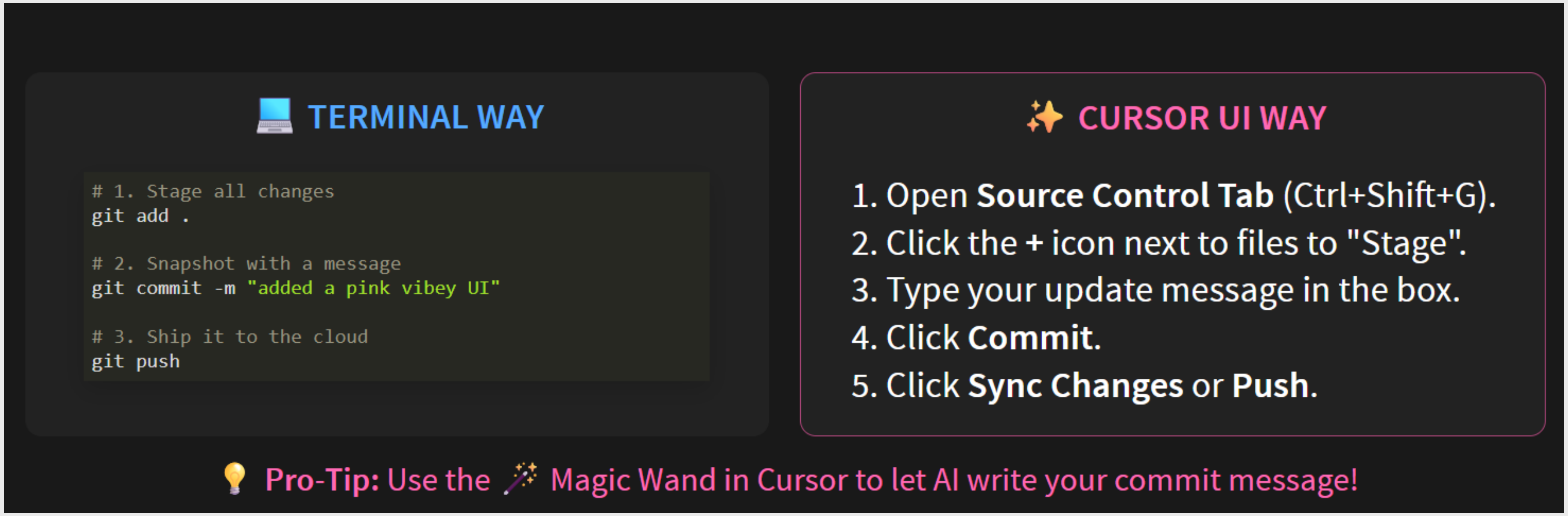
Pushing Changes to GitHub

- When you change your code, you need to update the repo as well

- You can do this without using the terminal in Cursor ☺
  - GitHub tab (click stuff)
  - Chat assistant (tell it stuff)

- The basic steps are
  1. Add files to be committed
  2. Commit the files with a comment describing the commit
  3. Push the commit to the repo

Pushing Changes to GitHub



💻 TERMINAL WAY

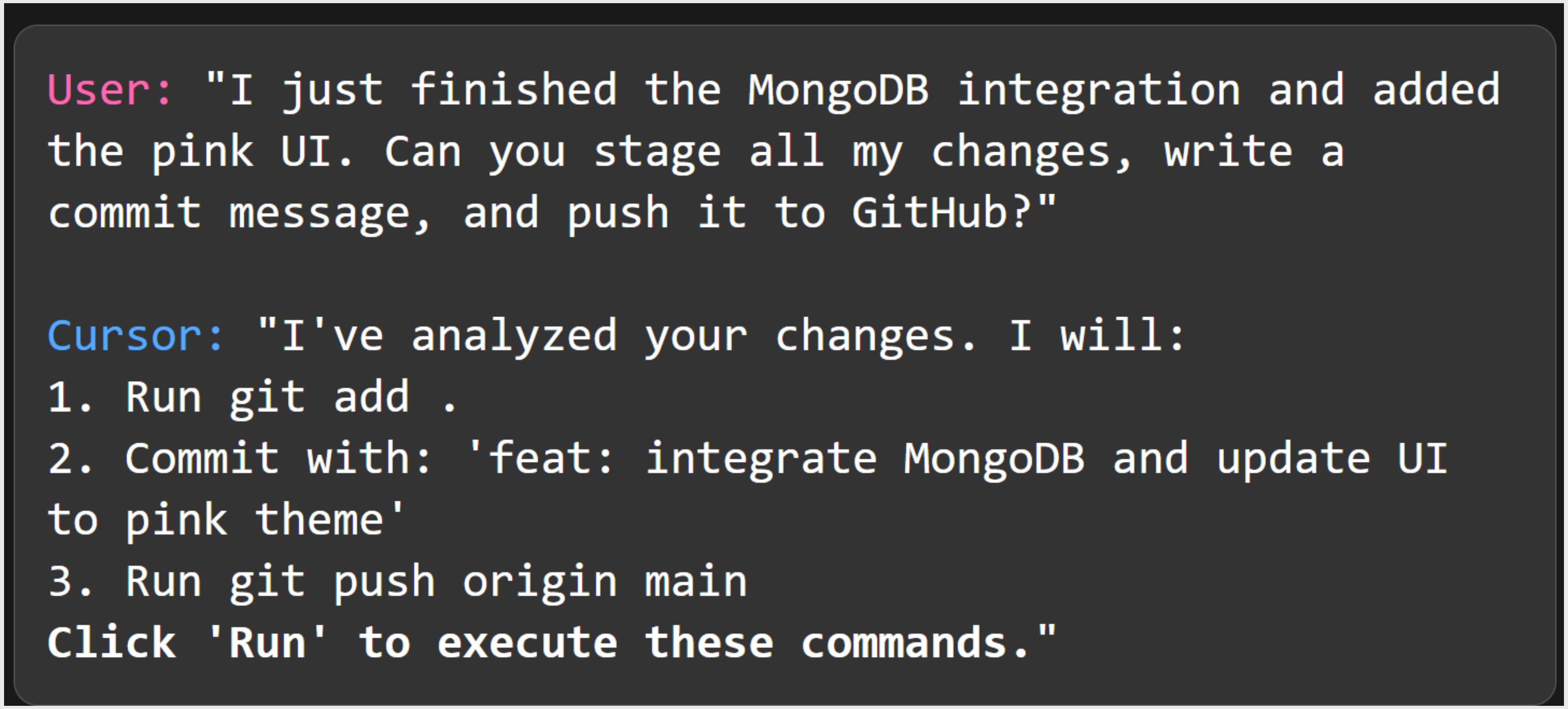
```
# 1. Stage all changes
git add .

# 2. Snapshot with a message
git commit -m "added a pink vibey UI"

# 3. Ship it to the cloud
git push
```

✨ CURSOR UI WAY

1. Open **Source Control Tab** (Ctrl+Shift+G).
2. Click the **+** icon next to files to "Stage".
3. Type your update message in the box.
4. Click **Commit**.
5. Click **Sync Changes** or **Push**.

💡 **Pro-Tip:** Use the 🪄 Magic Wand in Cursor to let AI write your commit message!

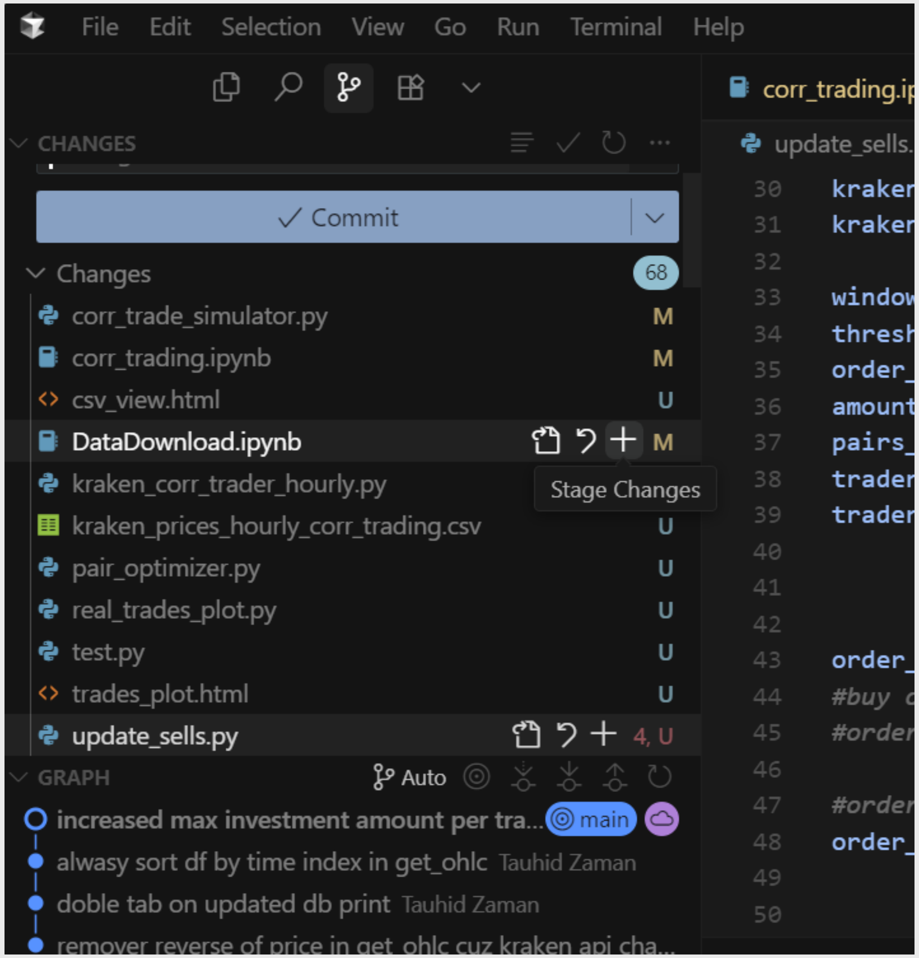
Pushing Changes with AI



```
User: "I just finished the MongoDB integration and added
the pink UI. Can you stage all my changes, write a
commit message, and push it to GitHub?"

Cursor: "I've analyzed your changes. I will:
1. Run git add .
2. Commit with: 'feat: integrate MongoDB and update UI
to pink theme'
3. Run git push origin main
Click 'Run' to execute these commands."
```

1. Stage Files to Be Committed

- Go into the GitHub tab
- Choose the files you want to stage for commit by clicking "+"
  - M = modified
  - U = untracked (not added to repo yet)



corr_trading.ip
update_sells.

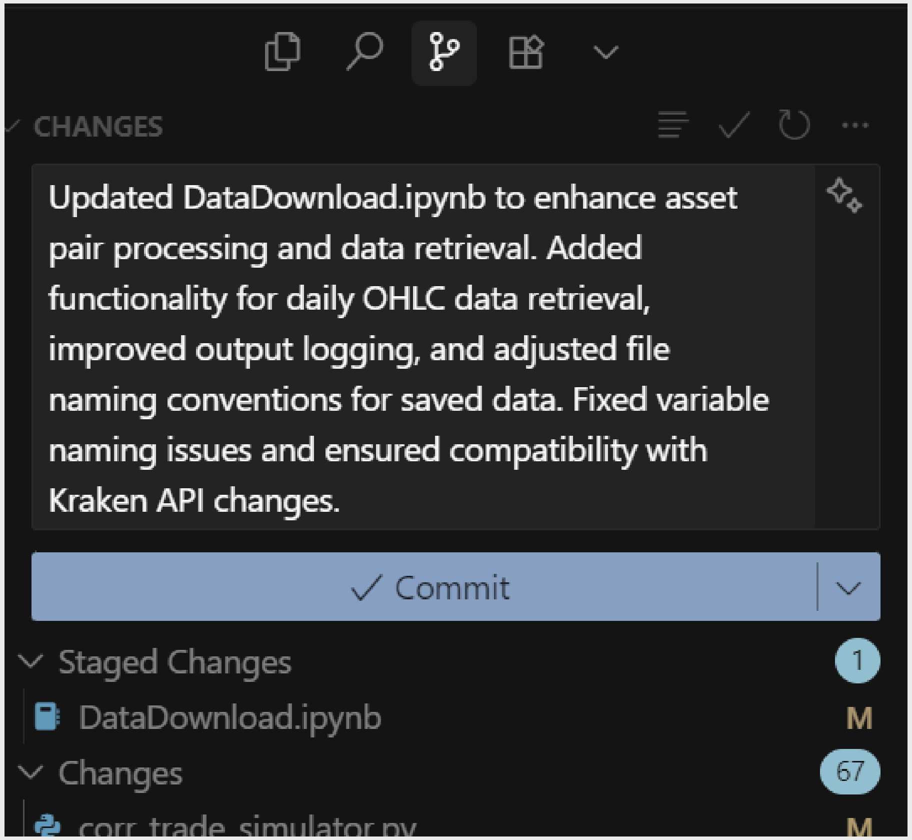
corr_trade_simulator.py M
corr_trading.ipynb M
csv_view.html U
DataDownload.ipynb M
kraken_corr_trader_hourly.py
kraken_prices_hourly_corr_trading.csv U
pair_optimizer.py U
real_trades_plot.py U
test.py U
trades_plot.html U
update_sells.py 4, U

increased max investment amount per tra...
alwasy sort df by time index in get_ohlc Tauhid Zaman
doble tab on updated db print Tauhid Zaman
remover reverse of price in get_ohlc cuz kraken api cha...

```
kraker
kraker

window
thresh
order_
amount
pairs_
trader
trader



order_
#buy c
#order

#order
order_
```

2. Write Commit Message

- **After you add the files, they will appear in the "Staged Changes" list**
- **Click the sparkles button to have the AI write a commit message (you need a commit message)**
- **Click "Commit"**



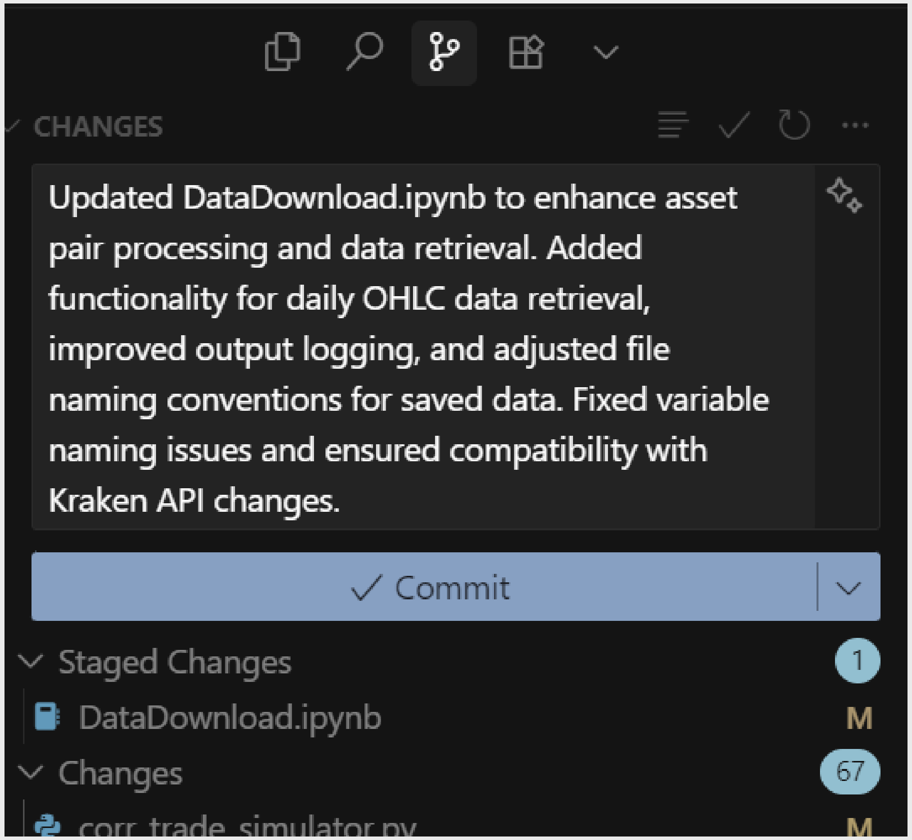
Updated DataDownload.ipynb to enhance asset pair processing and data retrieval. Added functionality for daily OHLC data retrieval, improved output logging, and adjusted file naming conventions for saved data. Fixed variable naming issues and ensured compatibility with Kraken API changes.

DataDownload.ipynb M

corr_trade_simulator.py M

3. Commit Changes

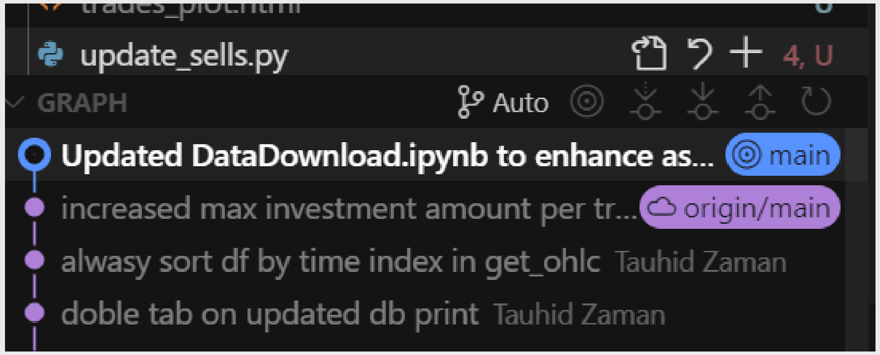
- Click “Commit”



Updated DataDownload.ipynb to enhance asset pair processing and data retrieval. Added functionality for daily OHLC data retrieval, improved output logging, and adjusted file naming conventions for saved data. Fixed variable naming issues and ensured compatibility with Kraken API changes.

4. Push Commit to Repo

- In the “GRAPH” tab you will see your commit in blue
- The current repo state is in purple



update_sells.py 4, U

GRAPH Auto

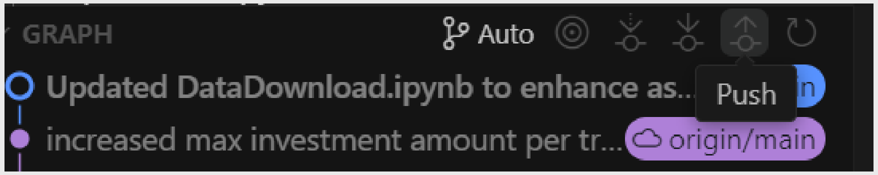
Updated DataDownload.ipynb to enhance as... main

increased max investment amount per tr... origin/main

alwasy sort df by time index in get_ohlc Tauhid Zaman

doble tab on updated db print Tauhid Zaman

- Click Push (the up arrow) to push your commit to the repo

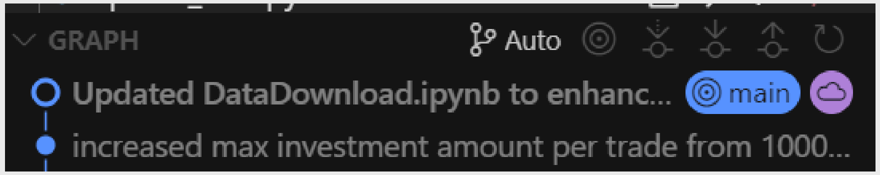


GRAPH Auto

Updated DataDownload.ipynb to enhance as... Push

increased max investment amount per tr... origin/main

- You will see your commit and the repo aligned after you push



GRAPH Auto

Updated DataDownload.ipynb to enhanc... main

increased max investment amount per trade from 1000...

Using a Cloned Repo

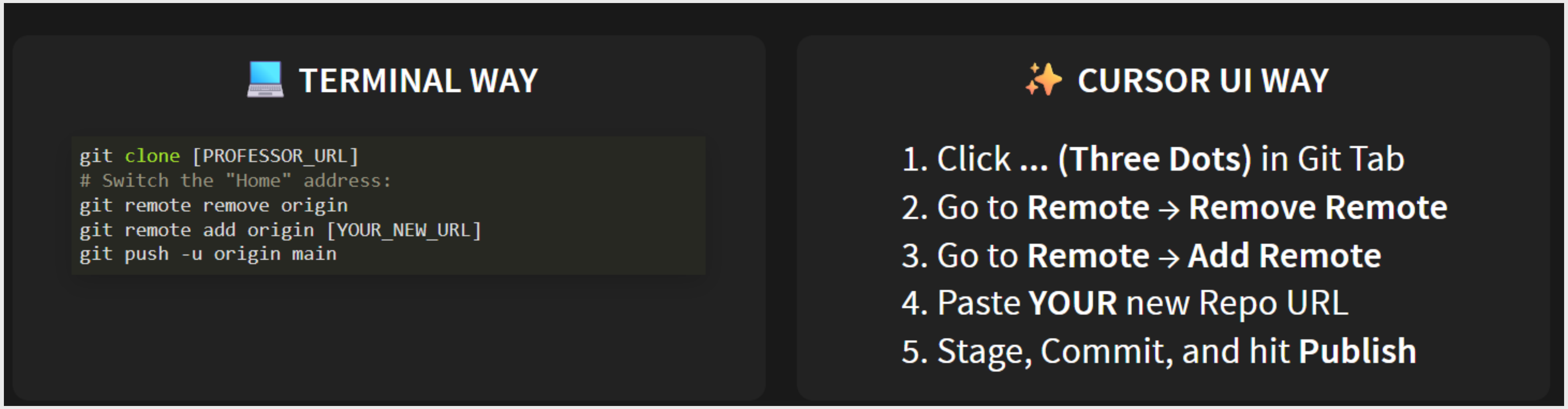
- You can clone a repo and then push it to your own repo so you can deploy it
  - You will do this for your homework

1. Clone the repo (git clone <url>)

2. Second, you delete the original remote

3. Third, you put your repo as the new remote

Using a Cloned Repo



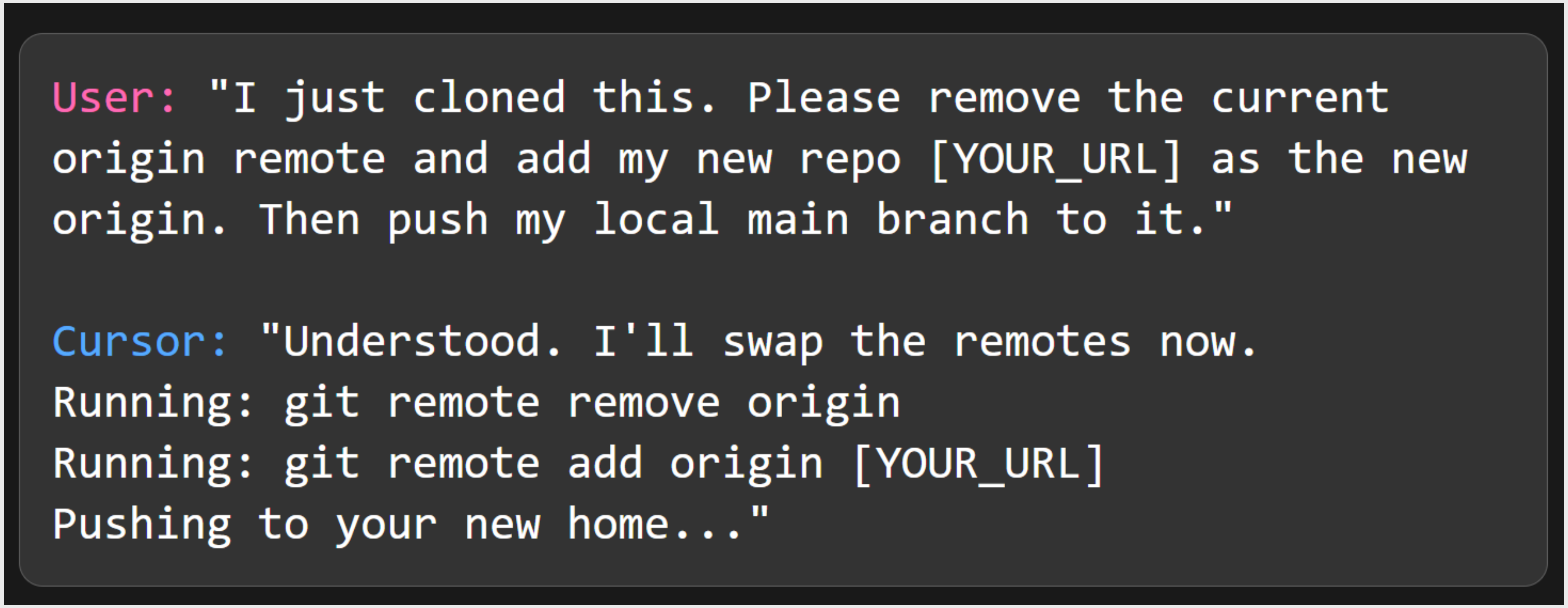
💻 TERMINAL WAY

```
git clone [PROFESSOR_URL]
# Switch the "Home" address:
git remote remove origin
git remote add origin [YOUR_NEW_URL]
git push -u origin main
```

✨ CURSOR UI WAY

1. Click **... (Three Dots)** in Git Tab
2. Go to **Remote** → **Remove Remote**
3. Go to **Remote** → **Add Remote**
4. Paste **YOUR** new Repo URL
5. Stage, Commit, and hit **Publish**

Using a Cloned Repo with AI



```
User: "I just cloned this. Please remove the current
origin remote and add my new repo [YOUR_URL] as the new
origin. Then push my local main branch to it."

Cursor: "Understood. I'll swap the remotes now.
Running: git remote remove origin
Running: git remote add origin [YOUR_URL]
Pushing to your new home..."
```

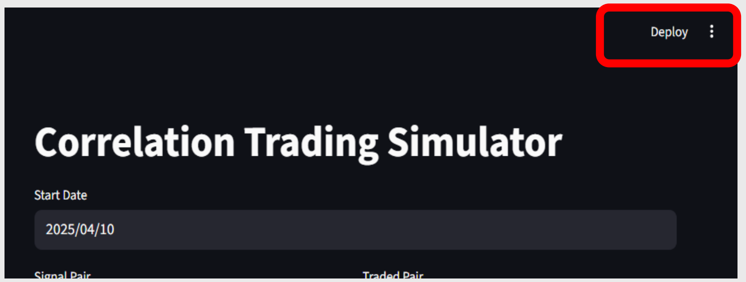
HOSTING APPS ONLINE

Hosting App Online

- Now that the app is on GitHub you can host it online
- Hosting is very easy and free for Streamlit
- Hosting is a bit more work for React and may cost some money

Hosting a Streamlit App

- Run the app locally on your machine
  - streamlit run app.py
- Click “deploy”
- Click “Deploy now” under Streamlit Community Cloud



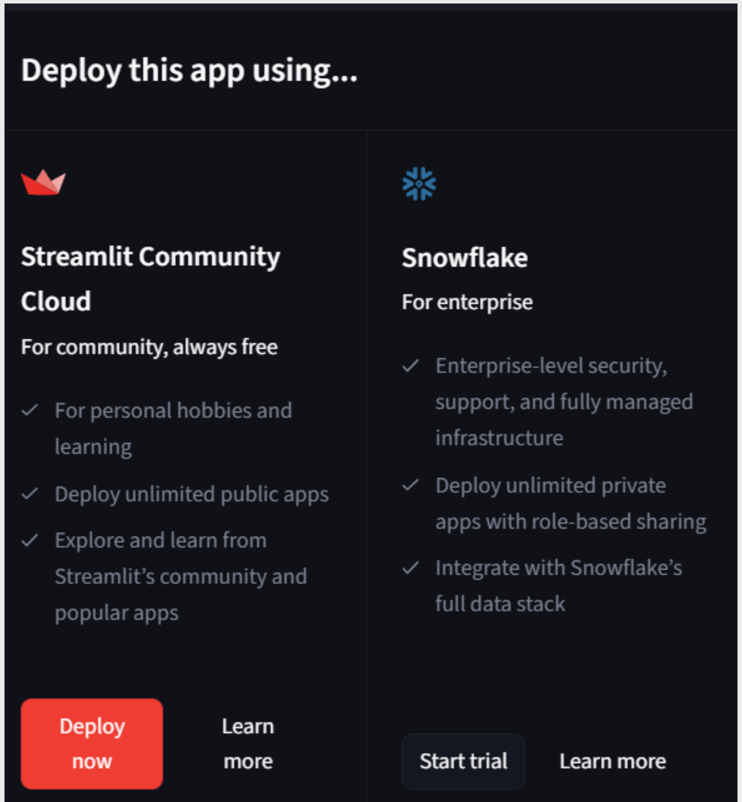
Correlation Trading Simulator

Start Date

2025/04/10

Signal Pair

Traded Pair


Deploy this app using...

Streamlit Community Cloud

For community, always free

- For personal hobbies and learning
- Deploy unlimited public apps
- Explore and learn from Streamlit’s community and popular apps

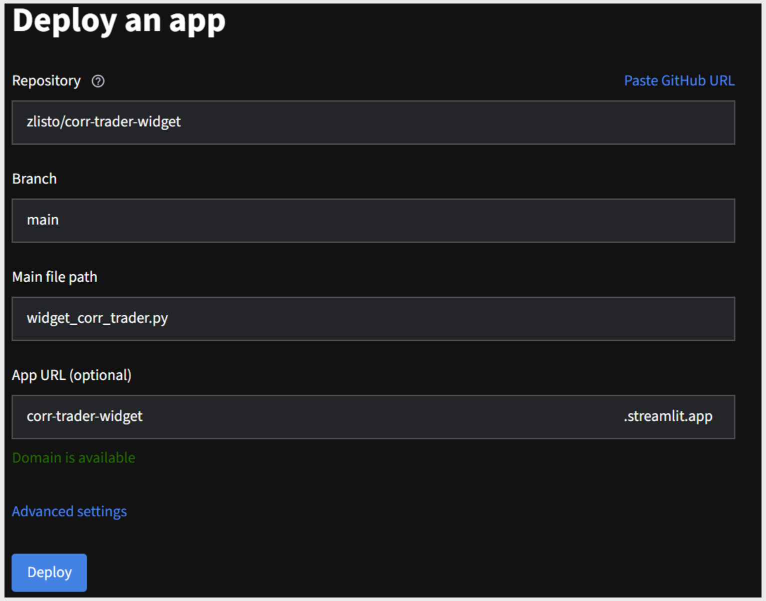
Snowflake

For enterprise

- Enterprise-level security, support, and fully managed infrastructure
- Deploy unlimited private apps with role-based sharing
- Integrate with Snowflake’s full data stack

Hosting a Streamlit App

- Streamlit fills in the basic info
  - You can give your app a custom URL if you like
- If you have a .env file, you need to manually put in those API keys in the "Advanced settings"
- Click "Deploy" and your app will be put online



Deploy an app

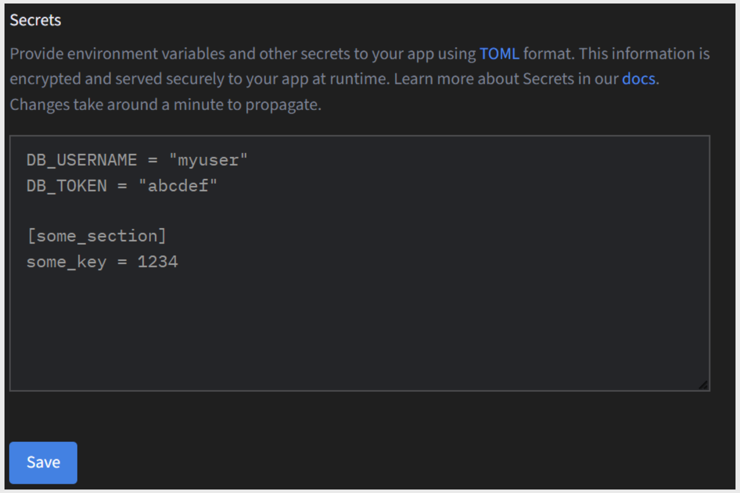
Repository: zlisto/corr-trader-widget

Branch: main

Main file path: widget_corr_trader.py

App URL (optional): corr-trader-widget.streamlit.app

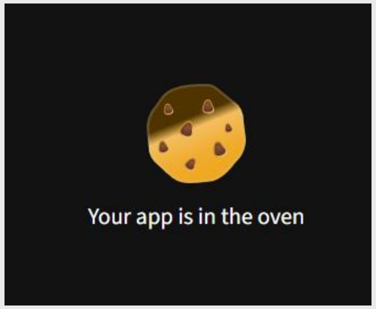
Domain is available



Secrets

Provide environment variables and other secrets to your app using TOML format. This information is encrypted and served securely to your app at runtime. Learn more about Secrets in our docs. Changes take around a minute to propagate.

```
DB_USERNAME = "myuser"
DB_TOKEN = "abcdef"

[some_section]
some_key = 1234
```



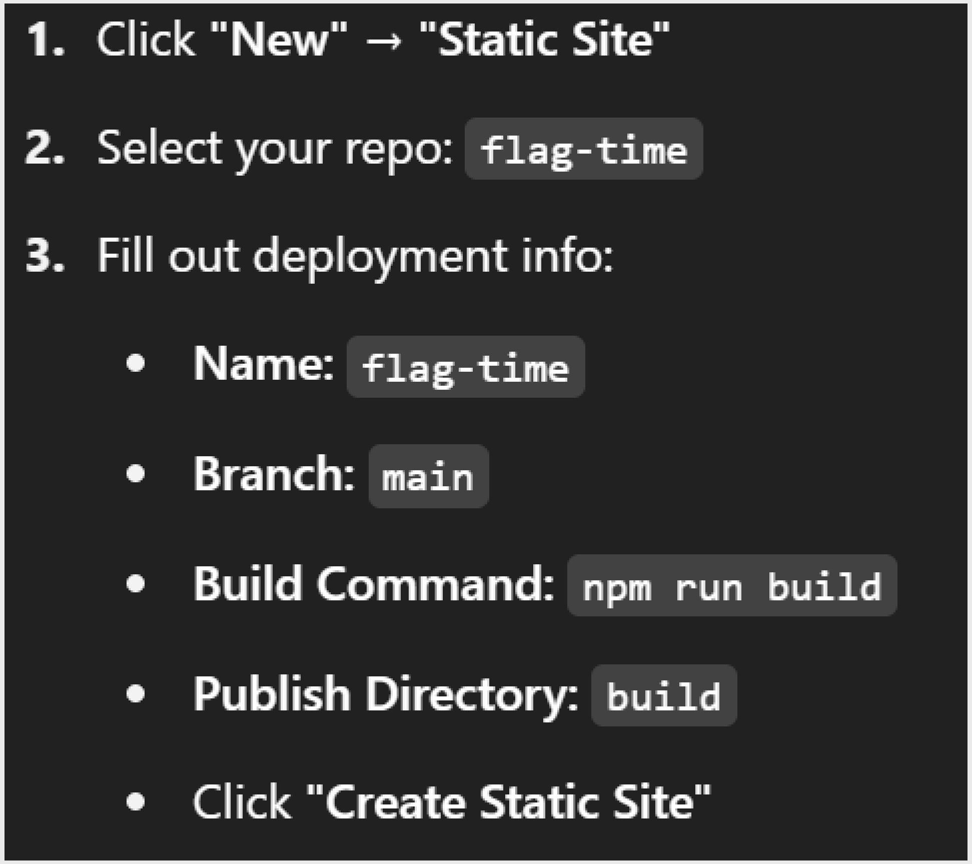
Hosting a React App

- We will use Render to host React Apps
  - Go to https://render.com
  - Click "Sign Up" and choose GitHub as login method
  - Authorize Render to access your GitHub repos

- Render will update the app automatically each time we push to the repo

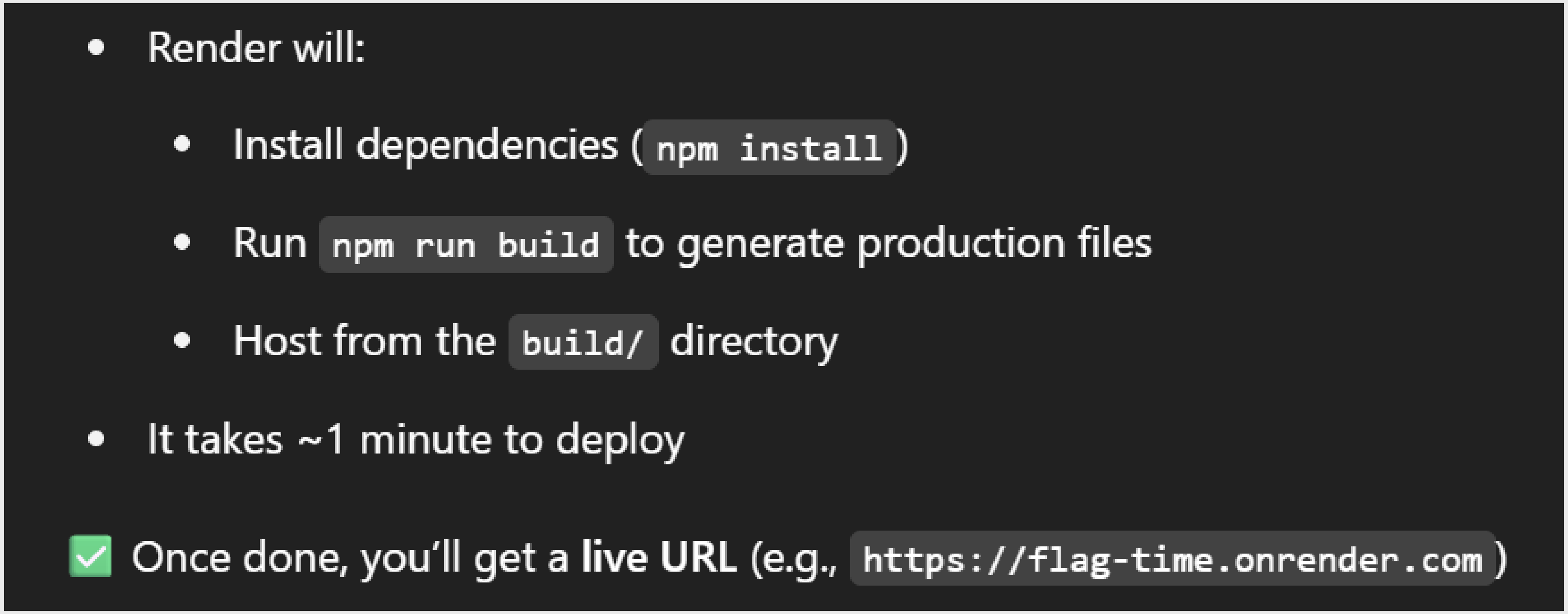
Create New Web Service

- Our app will be a new Static Site
- This information will be auto-filled by Render
  - The branch might be master instead of main



1. Click "New" → "Static Site"
2. Select your repo: `flag-time`
3. Fill out deployment info:
   - Name: `flag-time`
   - Branch: `main`
   - Build Command: `npm run build`
   - Publish Directory: `build`
   - Click "Create Static Site"

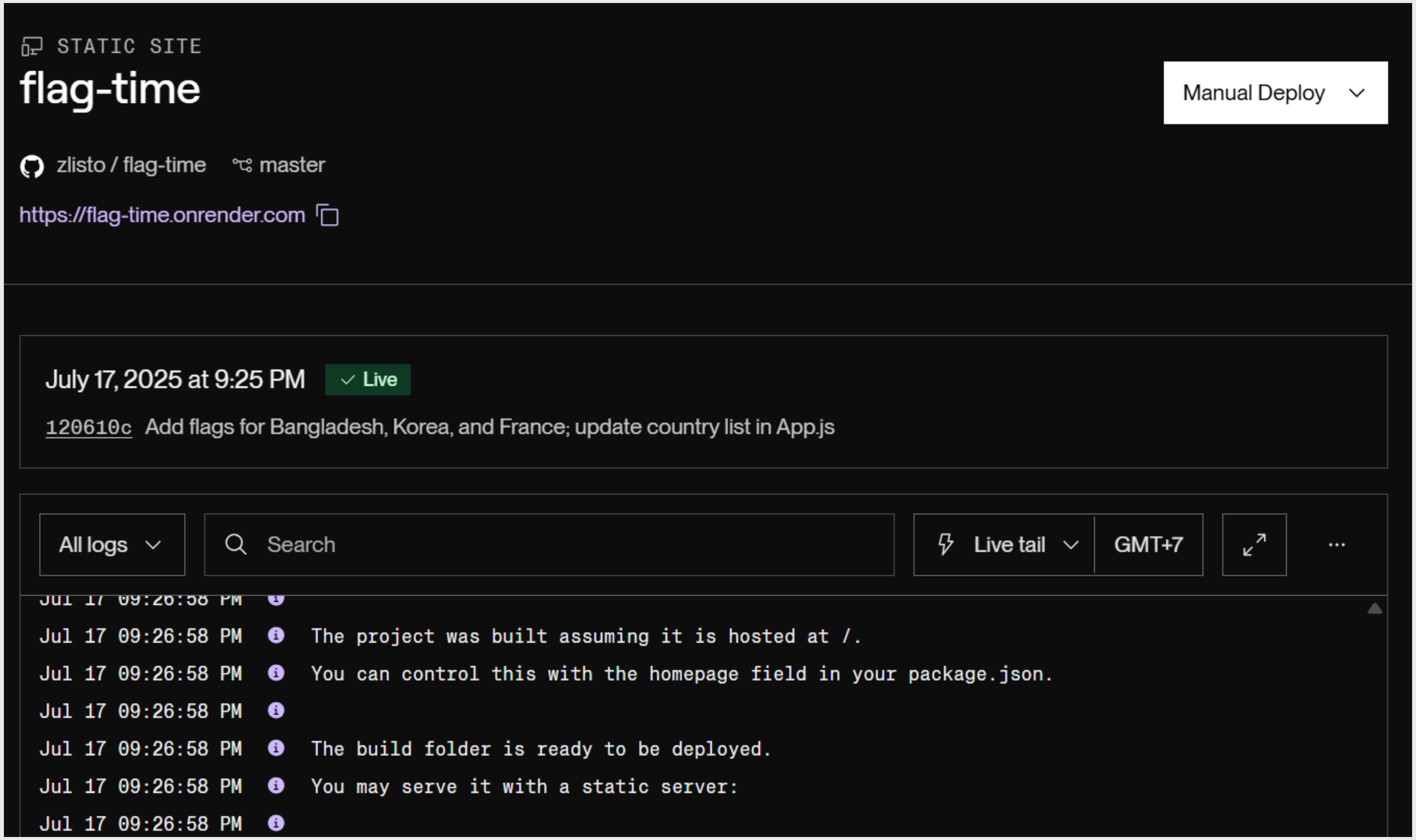
Let Render Cook



- Render will:
  - Install dependencies (`npm install`)
  - Run `npm run build` to generate production files
  - Host from the `build/` directory
- It takes ~1 minute to deploy

✅ Once done, you'll get a **live URL** (e.g., `https://flag-time.onrender.com`)

Deployed App



July 17, 2025 at 9:25 PM ✓ Live

120610c Add flags for Bangladesh, Korea, and France; update country list in App.js

```
Jul 17 09:26:58 PM  ℹ  The project was built assuming it is hosted at /.
Jul 17 09:26:58 PM  ℹ  You can control this with the homepage field in your package.json.
Jul 17 09:26:58 PM  ℹ
Jul 17 09:26:58 PM  ℹ  The build folder is ready to be deployed.
Jul 17 09:26:58 PM  ℹ  You may serve it with a static server:
Jul 17 09:26:58 PM  ℹ
```

DATABASES

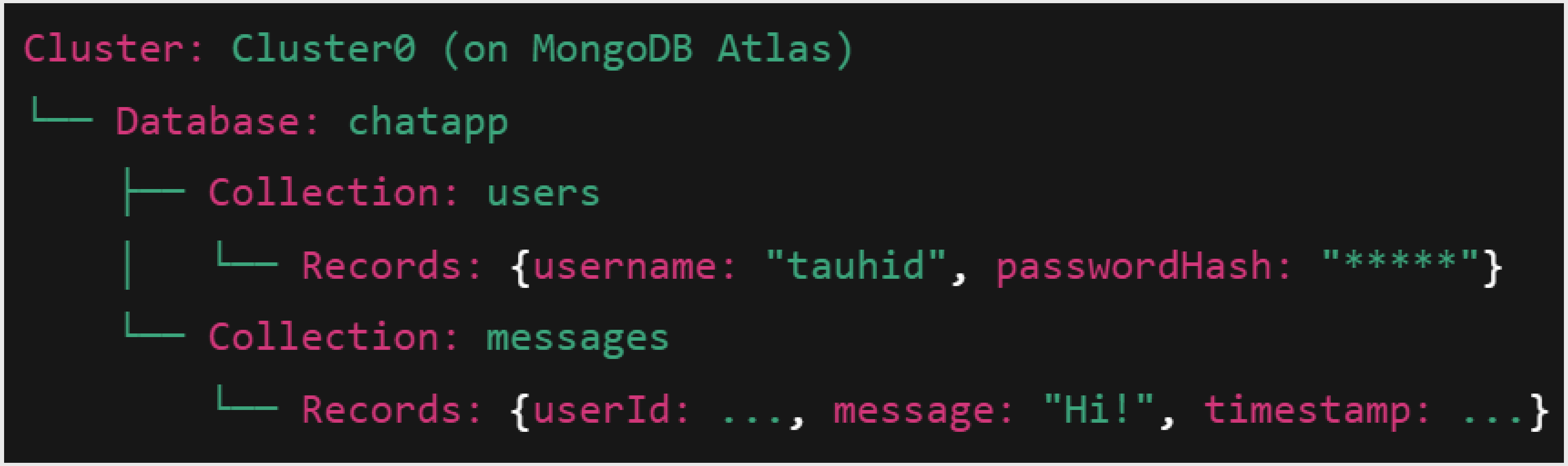
Databases

- A database is a structured place to store, access, and manage data
- Apps need to remember things
  - Users & passwords
  - AI chat history
  - Likes, posts, votes
- Without a database
  - You lose everything when the app restarts
  - No sharing of data across users or sessions

MongoDB

- MongoDB is a NoSQL database
  - Stores data as JSON-like records
- Great for flexible, modern apps
- Perfect for React, Node.js
- Cloud version = MongoDB Atlas

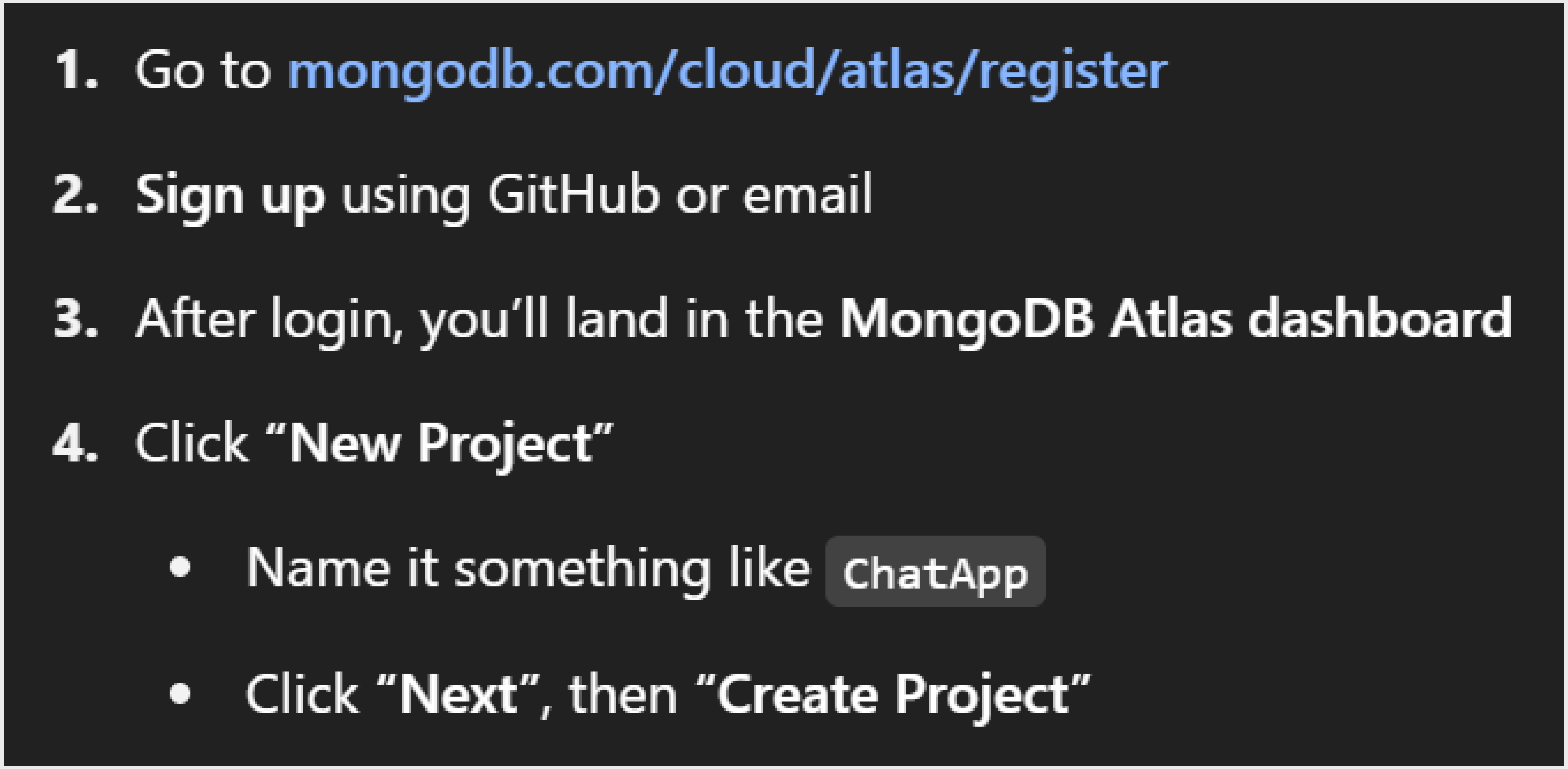
MongoDB Architecture



```
Cluster: Cluster0 (on MongoDB Atlas)
└── Database: chatapp
    ├── Collection: users
    │   └── Records: {username: "tauhid", passwordHash: "*****"}
    └── Collection: messages
        └── Records: {userId: ..., message: "Hi!", timestamp: ...}
```

- **Cluster - a server environment that can host multiple databases**
  - **You usually get one Free Tier shared cluster in MongoDB Atlas**
- **Database - a container for collections**
  - **Your app will have one database usually**
- **Collection - a flexible table that holds multiple records**
  - **Your app will have multiple collections**
- **Record - a JSON-like object stored in a collection**

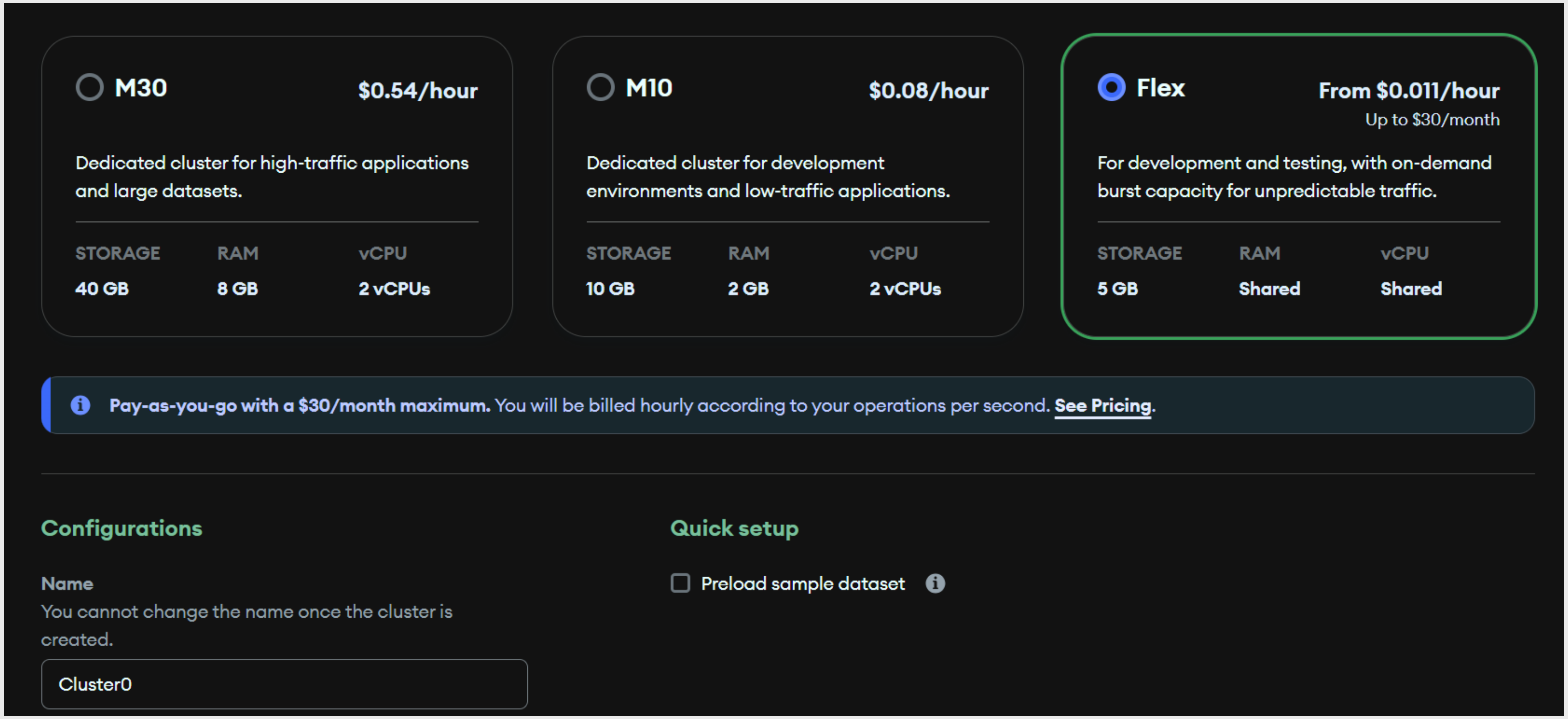
Create a MongoDB Account and Project



1. Go to mongodb.com/cloud/atlas/register
2. **Sign up** using GitHub or email
3. After login, you'll land in the **MongoDB Atlas dashboard**
4. Click "**New Project**"
   - Name it something like `ChatApp`
   - Click "**Next**", then "**Create Project**"

Create a MongoDB Cluster

- After you make your project, you will be asked to Create a Cluster for it
- Choose "Flex" and name it (your Flex should be free)



○ M30 — $0.54/hour

Dedicated cluster for high-traffic applications and large datasets.

STORAGE	RAM	vCPU
40 GB	8 GB	2 vCPUs

○ M10 — $0.08/hour

Dedicated cluster for development environments and low-traffic applications.

STORAGE	RAM	vCPU
10 GB	2 GB	2 vCPUs

● Flex — From $0.011/hour
Up to $30/month

For development and testing, with on-demand burst capacity for unpredictable traffic.

STORAGE	RAM	vCPU
5 GB	Shared	Shared

Pay-as-you-go with a $30/month maximum. You will be billed hourly according to your operations per second. **See Pricing**.

Configurations

Name
You cannot change the name once the cluster is created.

Cluster0

Quick setup

☐ Preload sample dataset

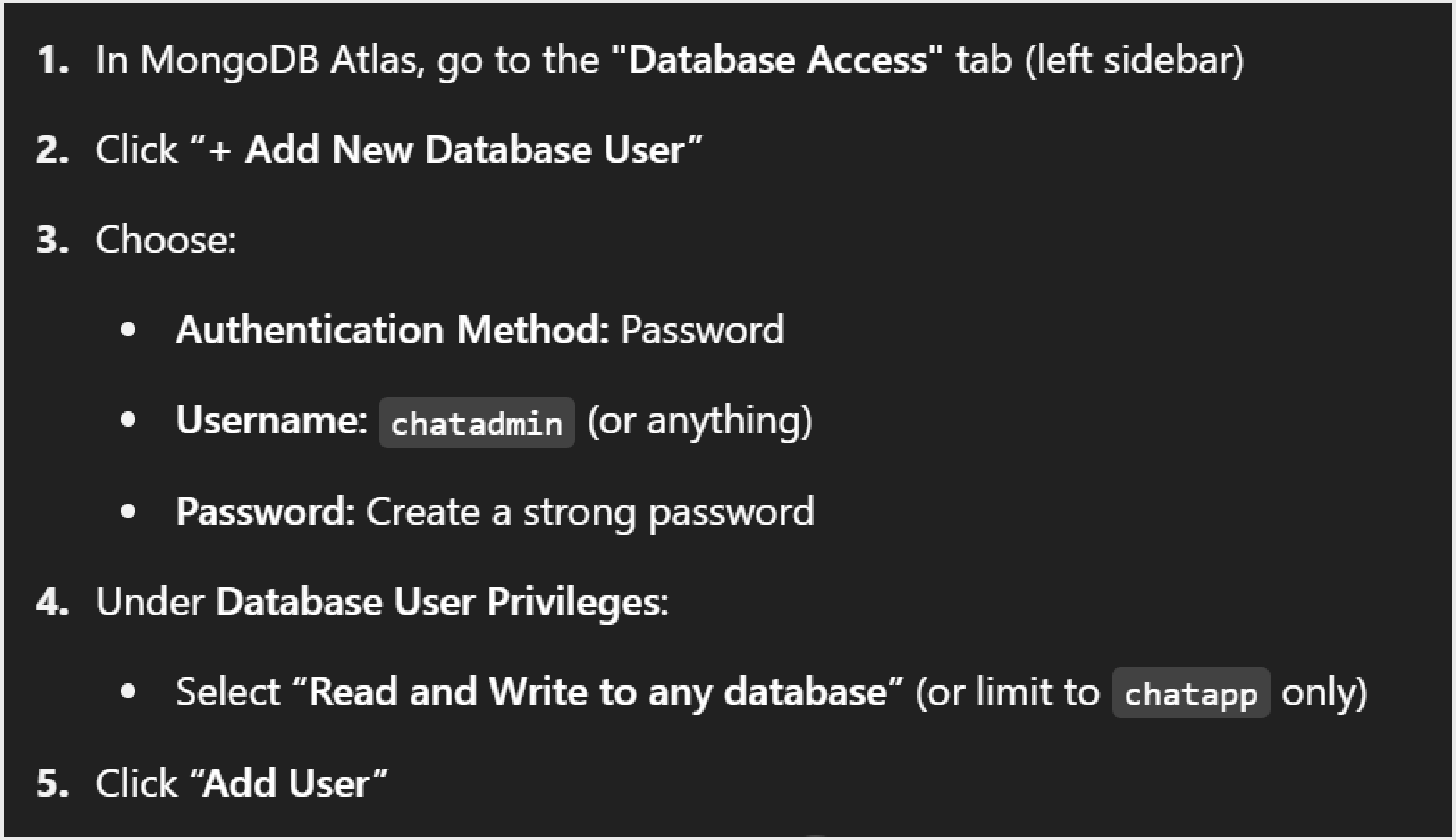
MongoDB Database User

- A database user is like a login for your app to access MongoDB

- It’s not the same as your MongoDB Atlas account

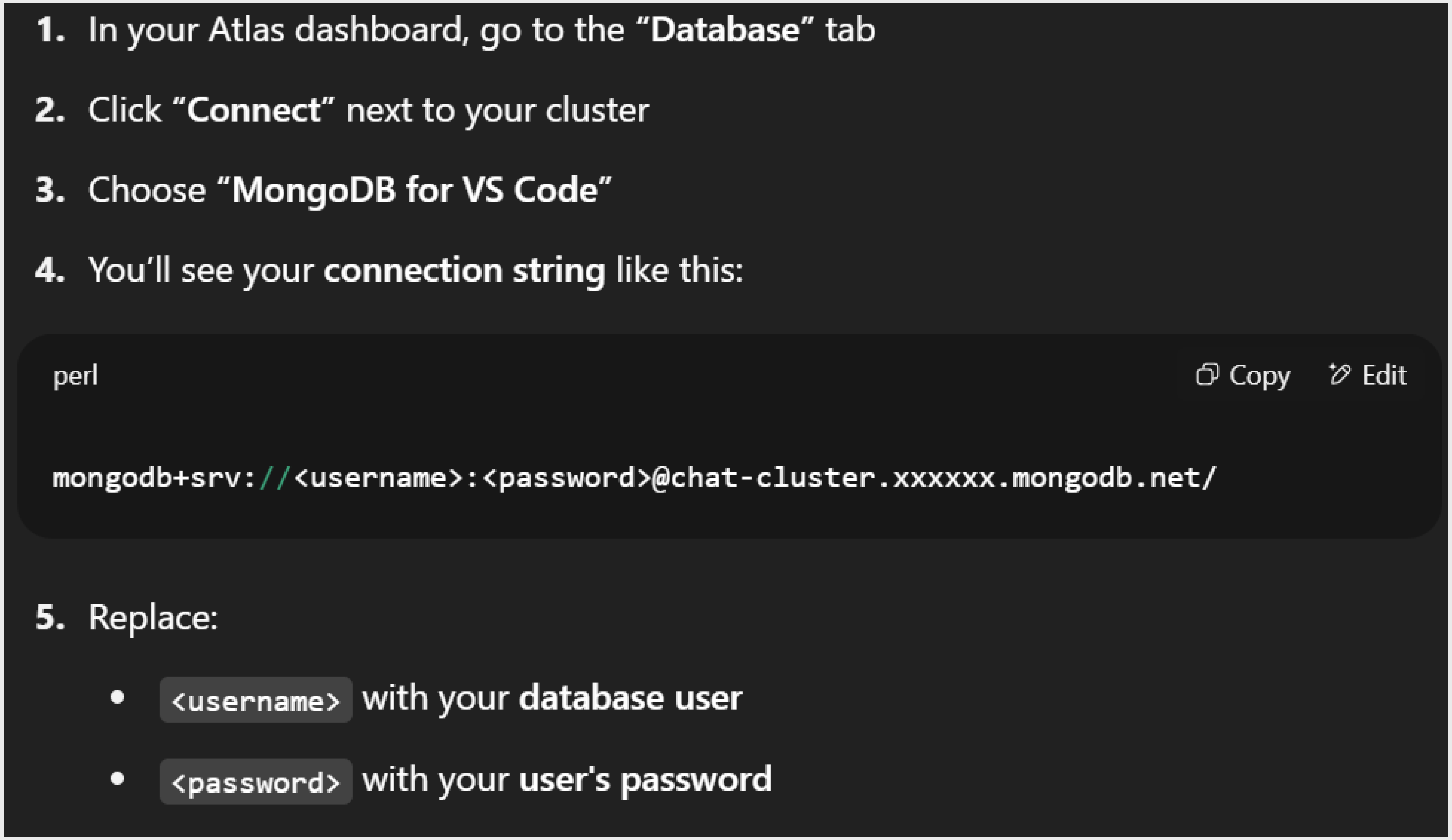
- Your app needs this user’s username + password to read/write data in your cluster
  - This will go in a .env file

Create a MongoDB Database User



1. In MongoDB Atlas, go to the "**Database Access**" tab (left sidebar)
2. Click "**+ Add New Database User**"
3. Choose:
   - **Authentication Method:** Password
   - **Username:** `chatadmin` (or anything)
   - **Password:** Create a strong password
4. Under **Database User Privileges:**
   - Select "**Read and Write to any database**" (or limit to `chatapp` only)
5. Click "**Add User**"

Connecting to MongoDB Database



1. In your Atlas dashboard, go to the "**Database**" tab
2. Click "**Connect**" next to your cluster
3. Choose "**MongoDB for VS Code**"
4. You'll see your **connection string** like this:

perl

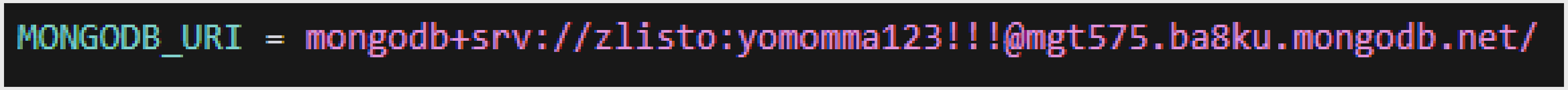
Copy Edit

```perl
mongodb+srv://<username>:<password>@chat-cluster.xxxxxx.mongodb.net/
```

5. Replace:
   - `<username>` with your **database user**
   - `<password>` with your **user's password**

Connecting to your MongoDB Collection

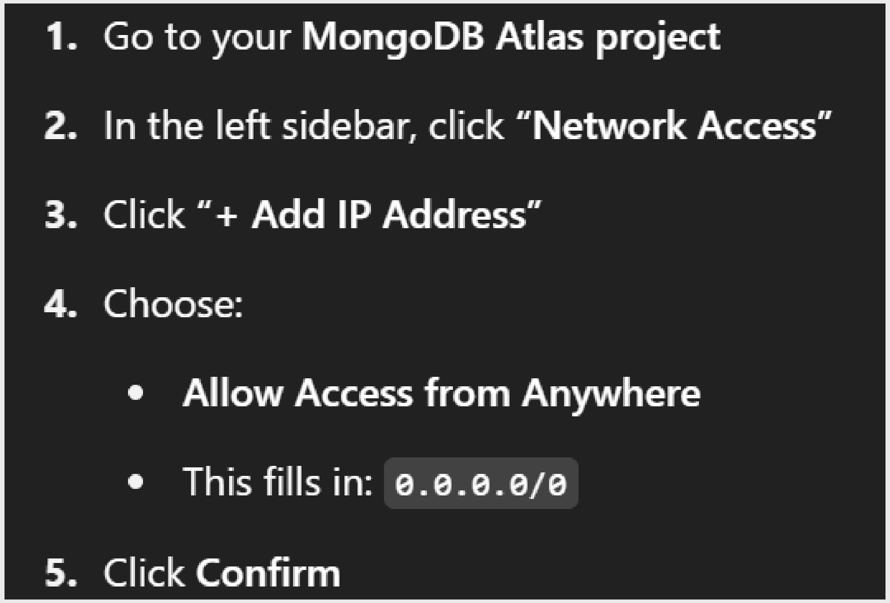
- You now have your MongoDB URI
- Put this into your .env file as MONGODB_URI



```
MONGODB_URI = mongodb+srv://zlisto:yomomma123!!!@mgt575.ba8ku.mongodb.net/
```

MongoDB Network Access

- **MongoDB Atlas blocks all incoming connections by default for security**
- **You must whitelist IP addresses allowed to connect to your cluster.**



1. Go to your **MongoDB Atlas project**
2. In the left sidebar, click "**Network Access**"
3. Click "**+ Add IP Address**"
4. Choose:
   - **Allow Access from Anywhere**
   - This fills in: `0.0.0.0/0`
5. Click **Confirm**

Using MongoDB with AI

- To do MongoDB things we just ask AI
  - “Create a database called “chat-app”
  - "Create a collection called 'users' and add 5 sample profiles."
  - "Find all messages from yesterday where the user was frustrated."
  - "Update the 'status' field to 'shipped' for this order ID.“
- Make sure the AI knows you have your MONGODB_URI in the .env file

Coding Session

- We will build a simple AI chatbot in React
- The chat storage will be done on MongoDB
- We will push the app to GitHub
- We will host the app Render