nuvolaris

AGENT 41

Agent41 Course

Apache OpenServerless

Lesson 1

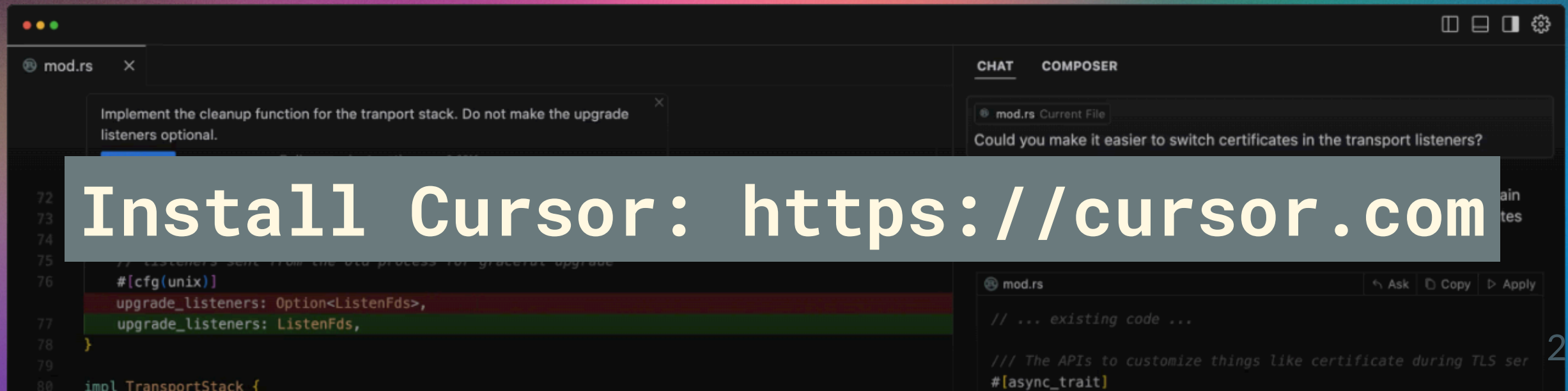
Vibe Coding with Cursor

The AI Code Editor

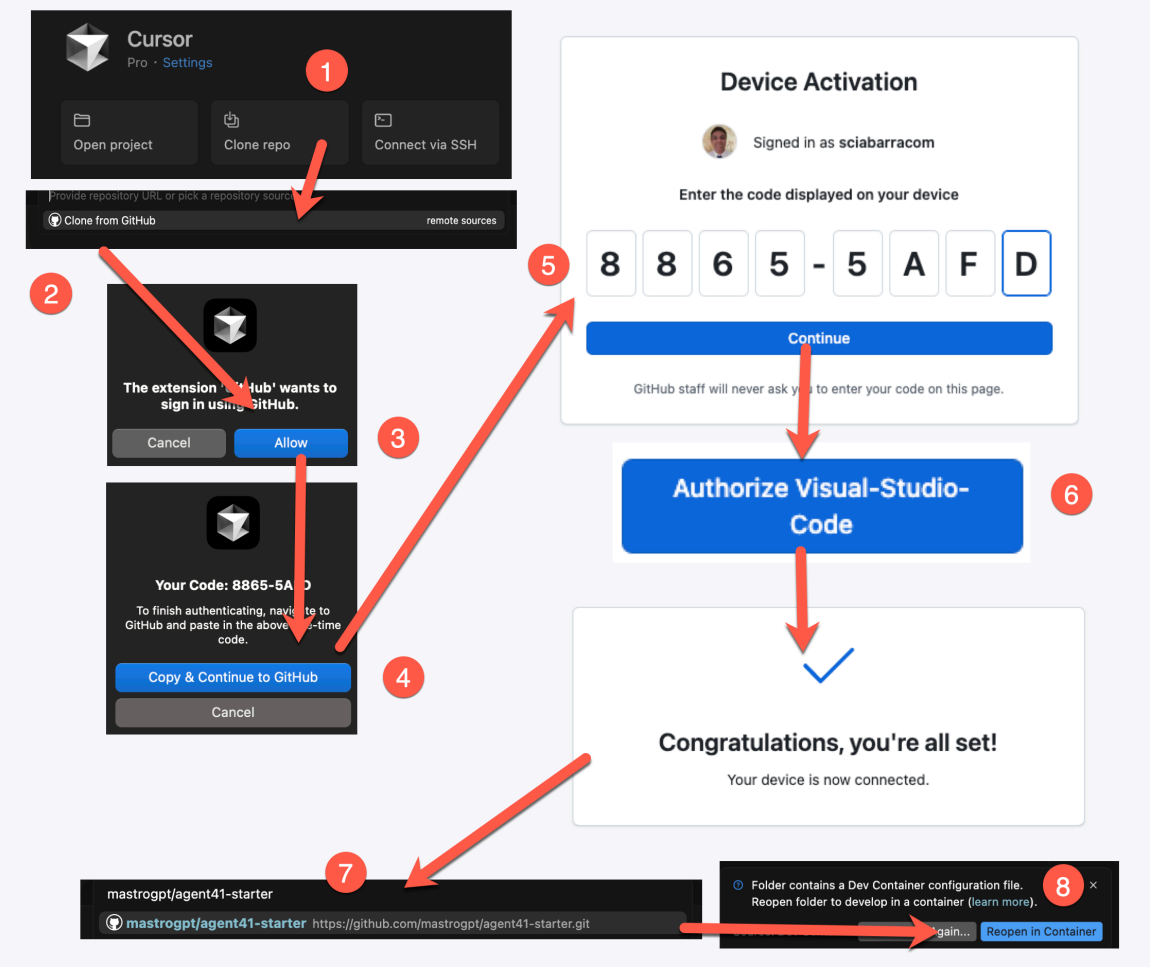
Built to make you extraordinarily productive, Cursor is the best way to code with AI.

DOWNLOAD FOR MACOS

ALL DOWNLOADS



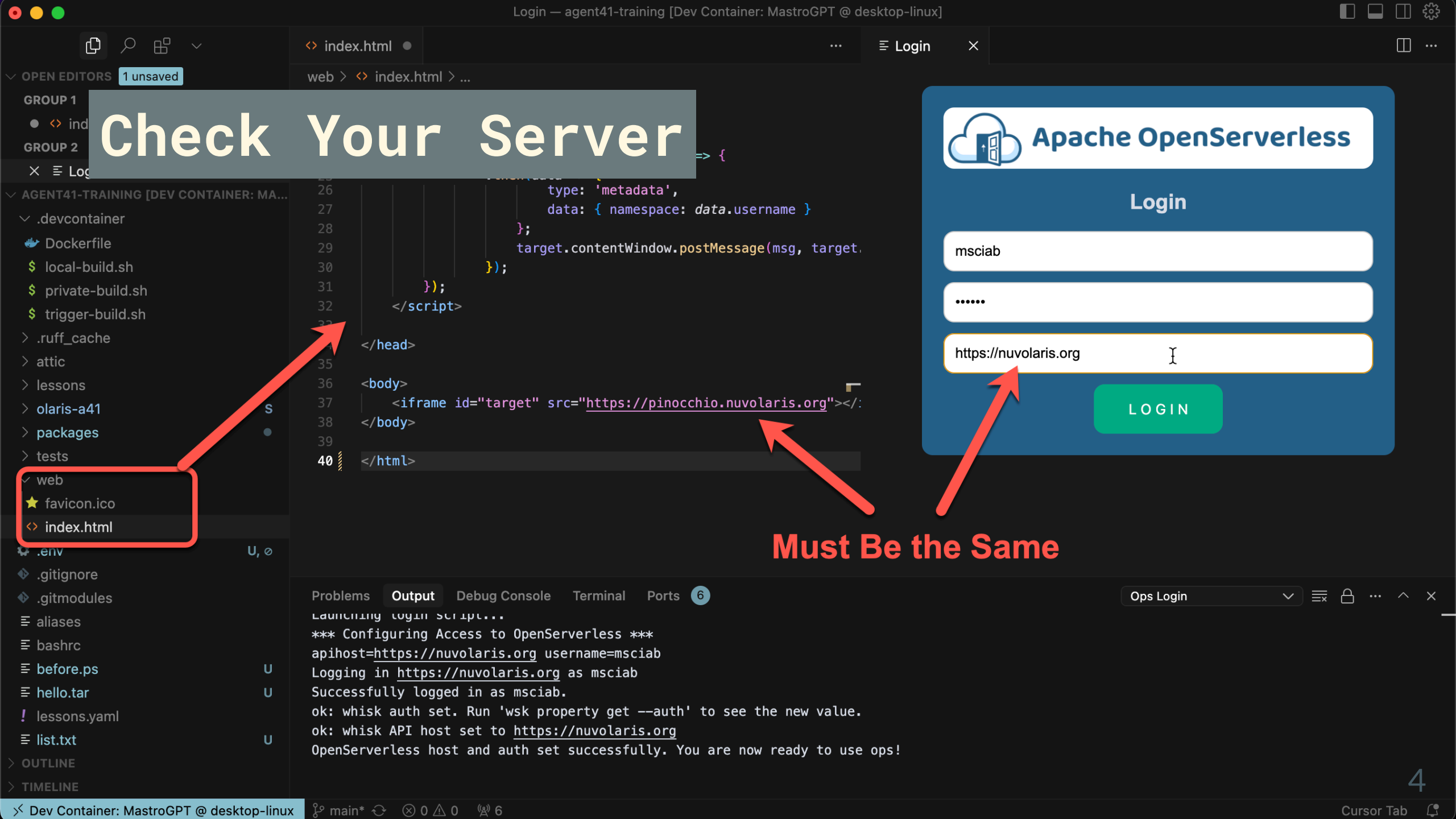
Install Cursor: https://cursor.com



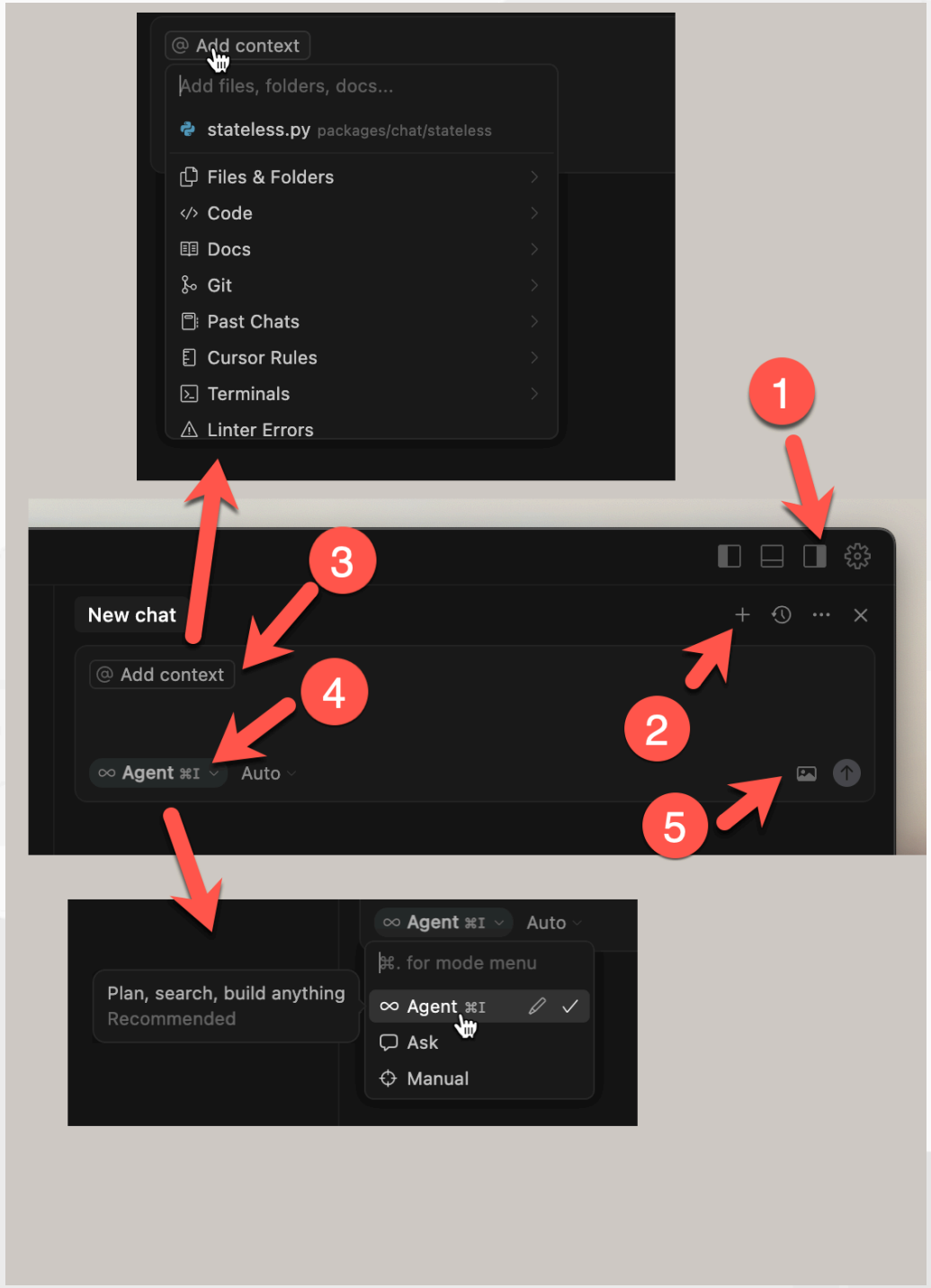
Setup

1. Clone repo
2. Clone from GitHub
3. Allow GitHub access
4. Copy Device Code
5. Paste Device Code
6. Authorize VSCode
7. Clone
   `mastrogpt/agent41-starter`
8. Open Devcontainer

Check Your Server

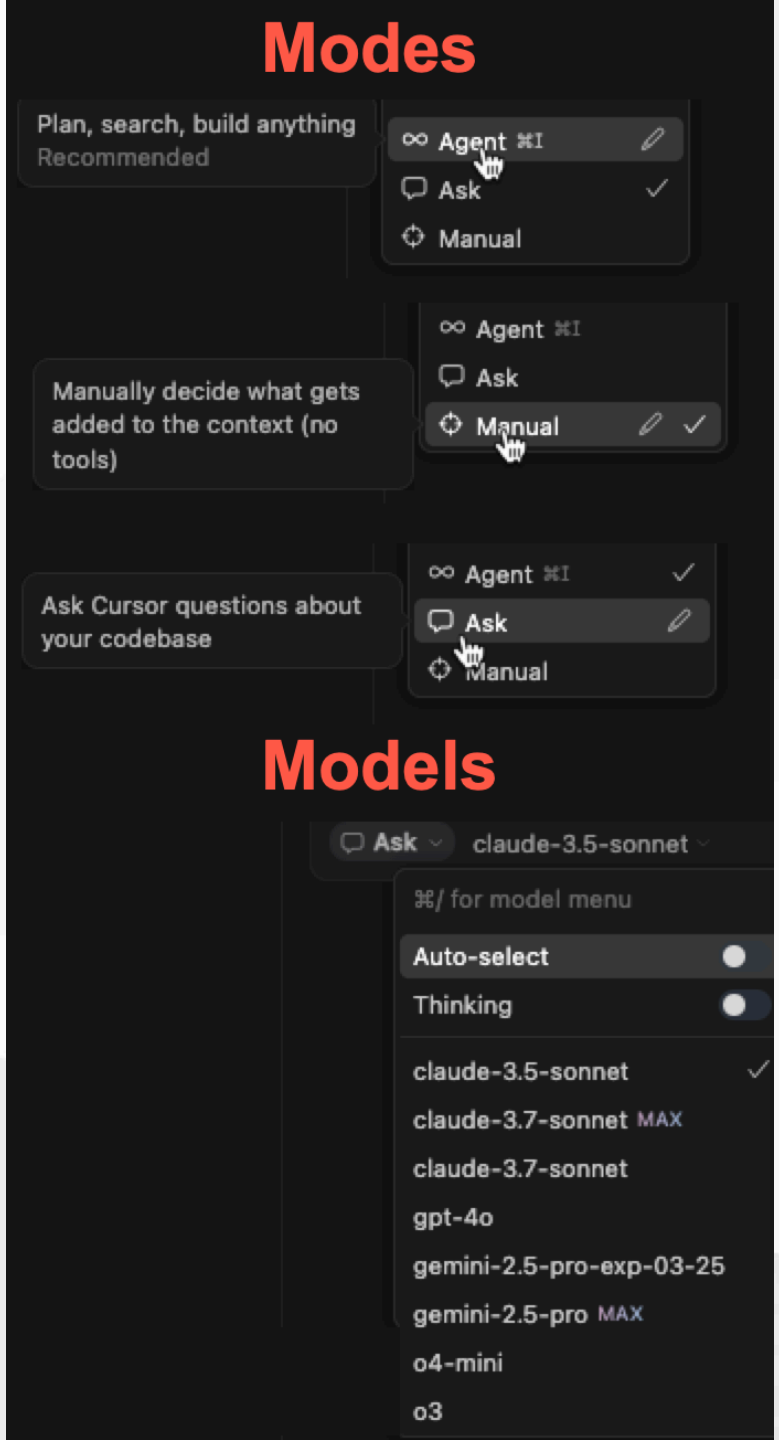


1. Open AI Chat
2. Create a New Chat
3. Select the context
    - File & Folders
    - Code
    - More...
4. Select the Mode
    - Agent
    - Ask
    - Manual



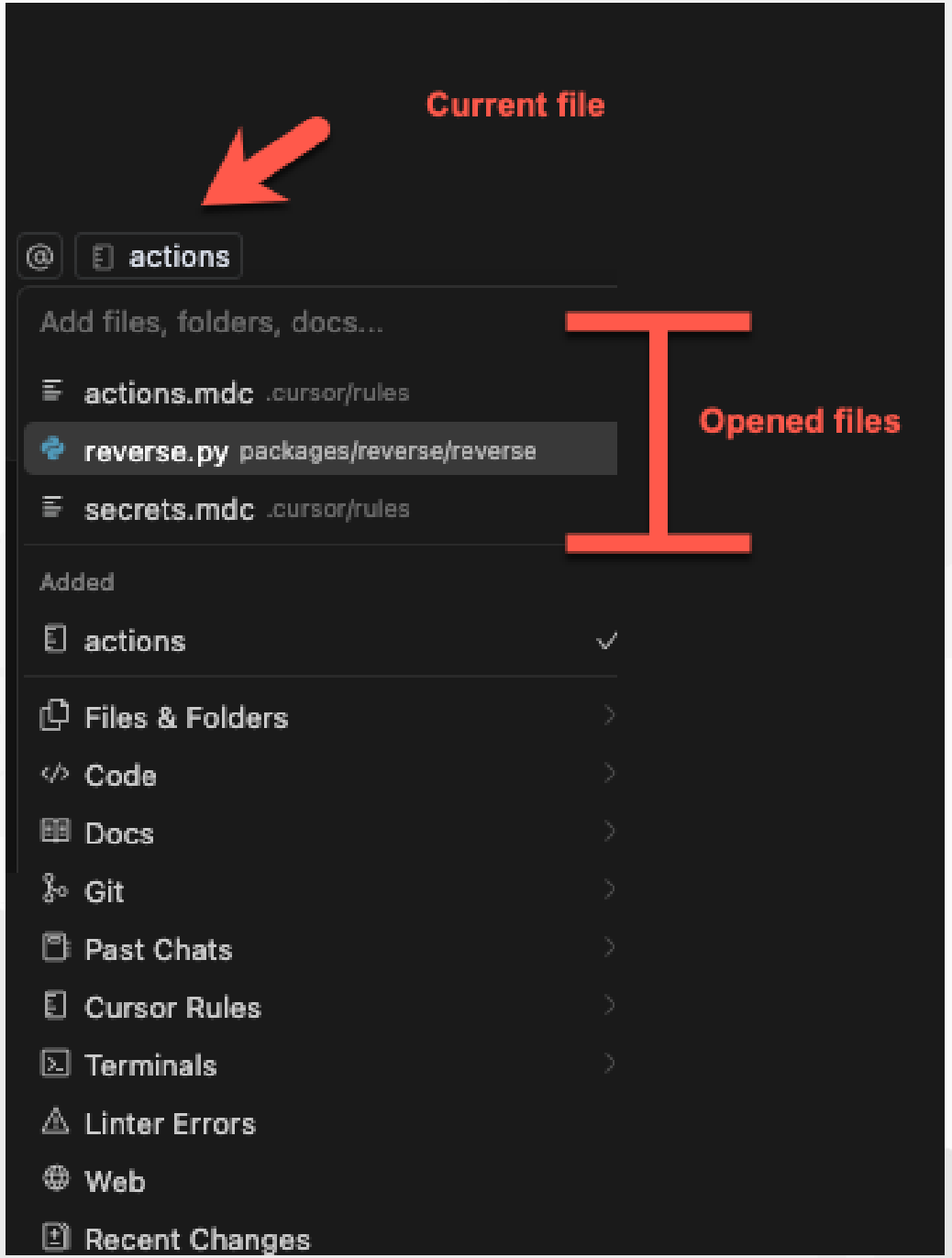
Modes

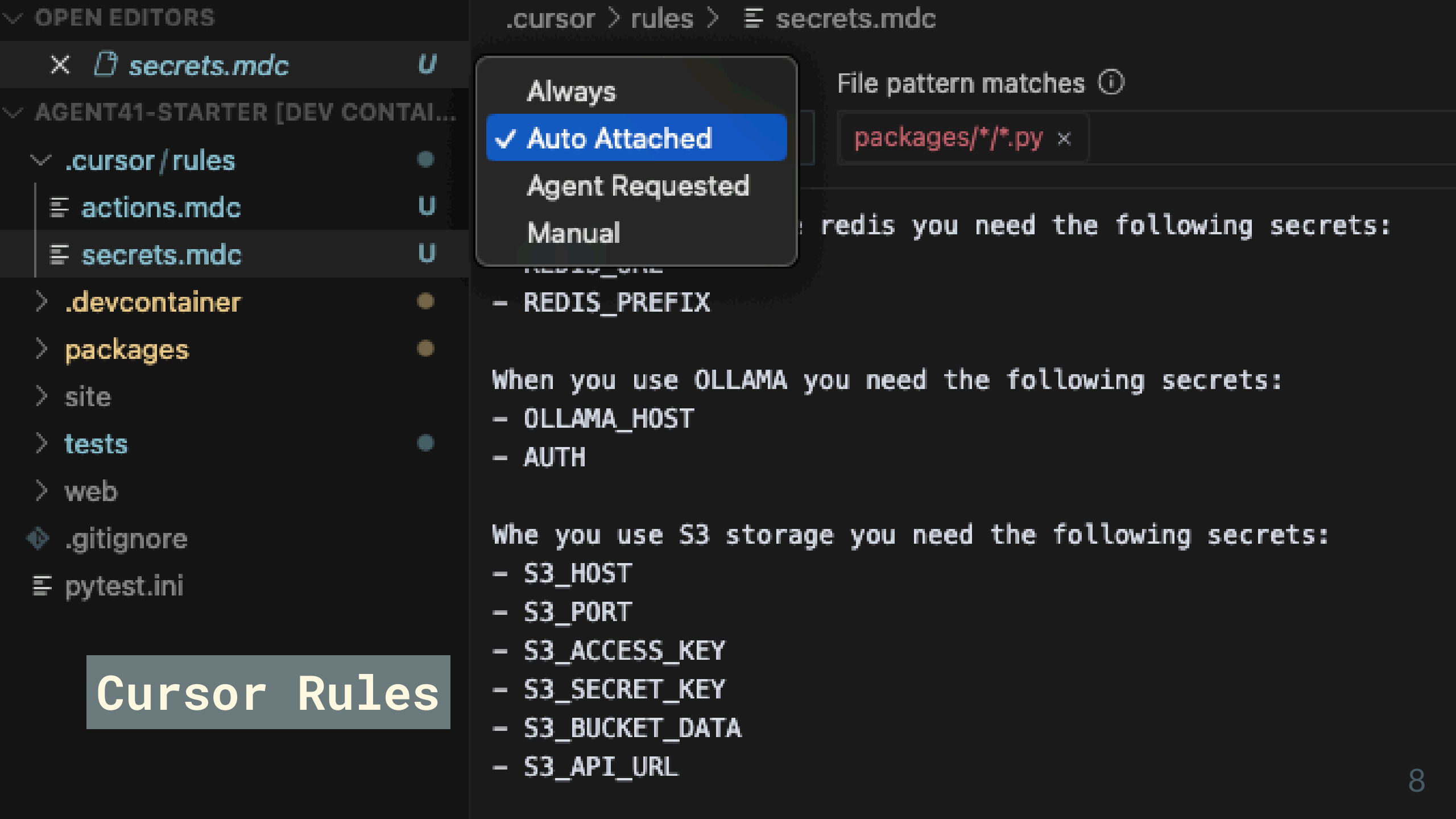
- **Agent**: fully automated
  - Find the context
  - Propose solution
  - Implements them
- **Ask**
  - As agent but do not act
- **Manual**
  - You have to provide the context



Contexts

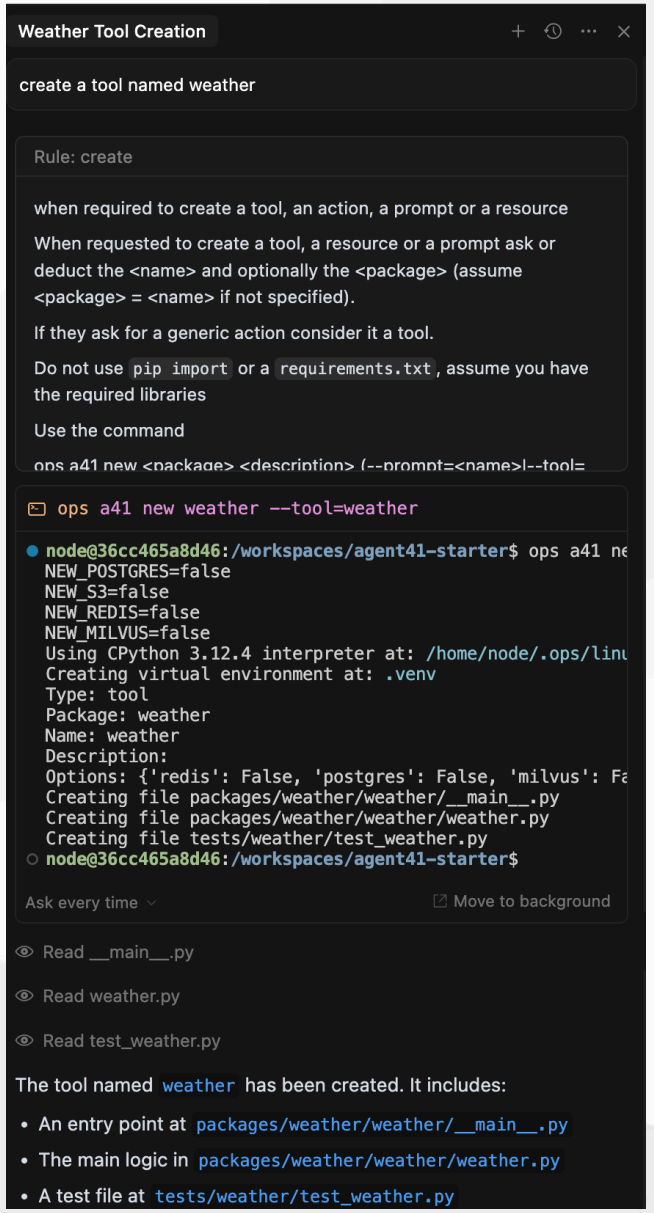
- Files and Folders
- Code (snipptets)
- Docs
- Past git revisions
- Past chats
- Terminal commands
- Errors
- Web
- **Cursor Rules**




OPEN EDITORS
secrets.mdc U
AGENT41-STARTER [DEV CONTAI...
.cursor / rules
actions.mdc U
secrets.mdc U
.devcontainer
packages
site
tests
web
.gitignore
pytest.ini
.cursor > rules > secrets.mdc
Always
Auto Attached
Agent Requested
Manual
File pattern matches
packages/*/*.py
redis you need the following secrets:
- REDIS_PREFIX
When you use OLLAMA you need the following secrets:
- OLLAMA_HOST
- AUTH
Whe you use S3 storage you need the following secrets:
- S3_HOST
- S3_PORT
- S3_ACCESS_KEY
- S3_SECRET_KEY
- S3_BUCKET_DATA
- S3_API_URL
Cursor Rules

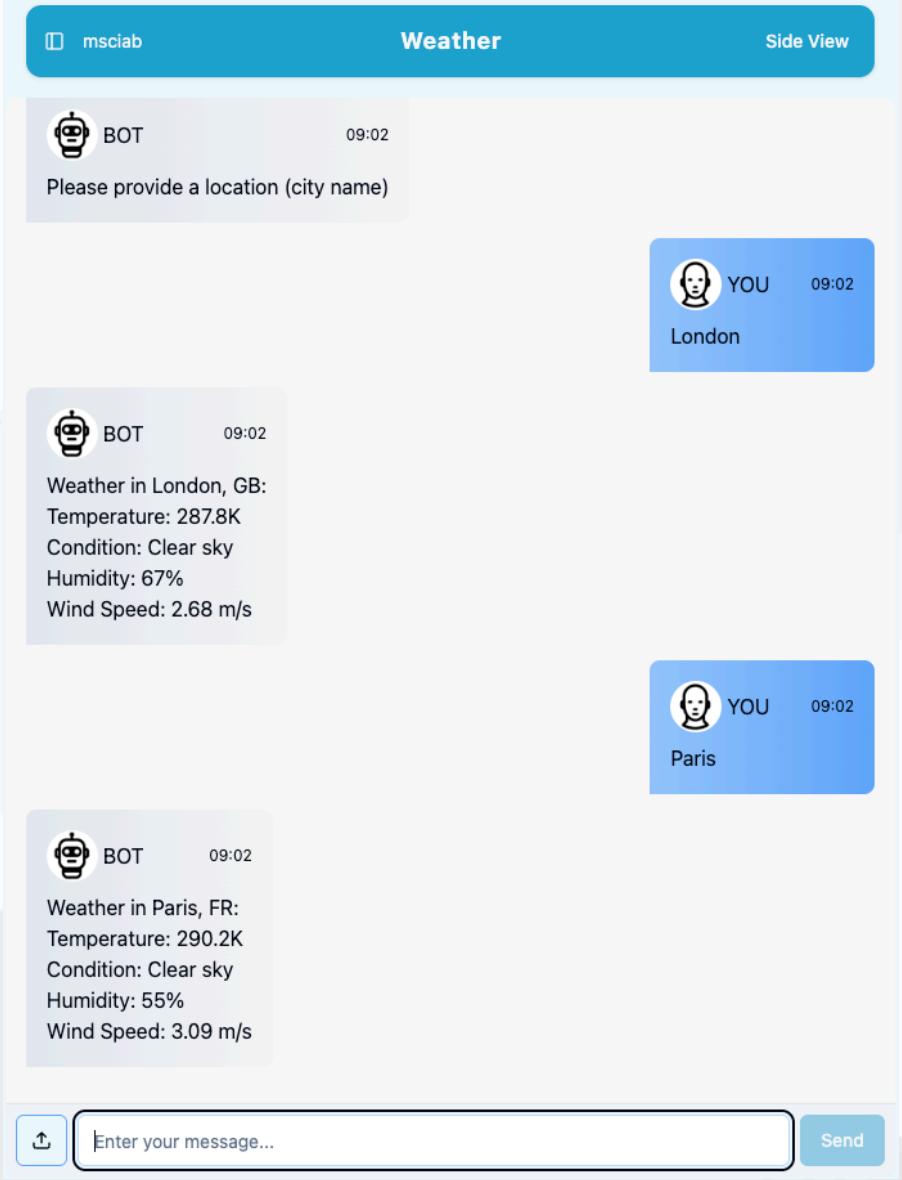
Create Weather Tool

- `create a tool named weather`
  - **use a rule**
- `access to a service to get the wether in a location`
  - **asks a a question**
- `use openweather`
  - **my answer**



Implement Weather Tool

- `read the api key`
- `make a request to openwhethermap and retrieve the wheather for a given location`
- `add to the index`
- test and deploy



Use Ollama to Parse Time

- `write a function ask_ollama to ask using model llama3.1:8b and return its anwser`
  - it should select or use the `ollama` rule
- `write parse_time_and_location invoking ask_ollama assuming it is a request for a wheather prediction at a given location and optionally a time. Return the answer as a tuple with 3 values: the location of the request, the time of the request as a number of in hours in advance from now, and the number of days in advance. Default to 0 for both if not specified`

Parse Time fixes and tests

- LLM does not know the current day and time we have to specify it
  - `provide the current time and date to the prompt`
- Sometimes the output is not a json object but is is embedded:
  - `check the respnse and extract a json object if it is embedded in the answer`
- We need to add tests:
  - `write tests to check form London, London tomorrow moning, Paris next Week and Moscow in the night`

Close the loop fixing the `weather` function

```
Change the weather function to use the parse_and_time location and
use always a forecast. In the result answer specify the date and
time for the forecast in the local time.
```

```
write tests for London, London tomorrow, Paris next week
```