Vibe Coding

The gap between your idea and working code just collapsed

Raiqa Rasool - EPSCI Computer Scientist - CST Division

May 15, 2026 | rasool@jlab.org

“VIBE CODING” — WHERE IT ALL BEGAN

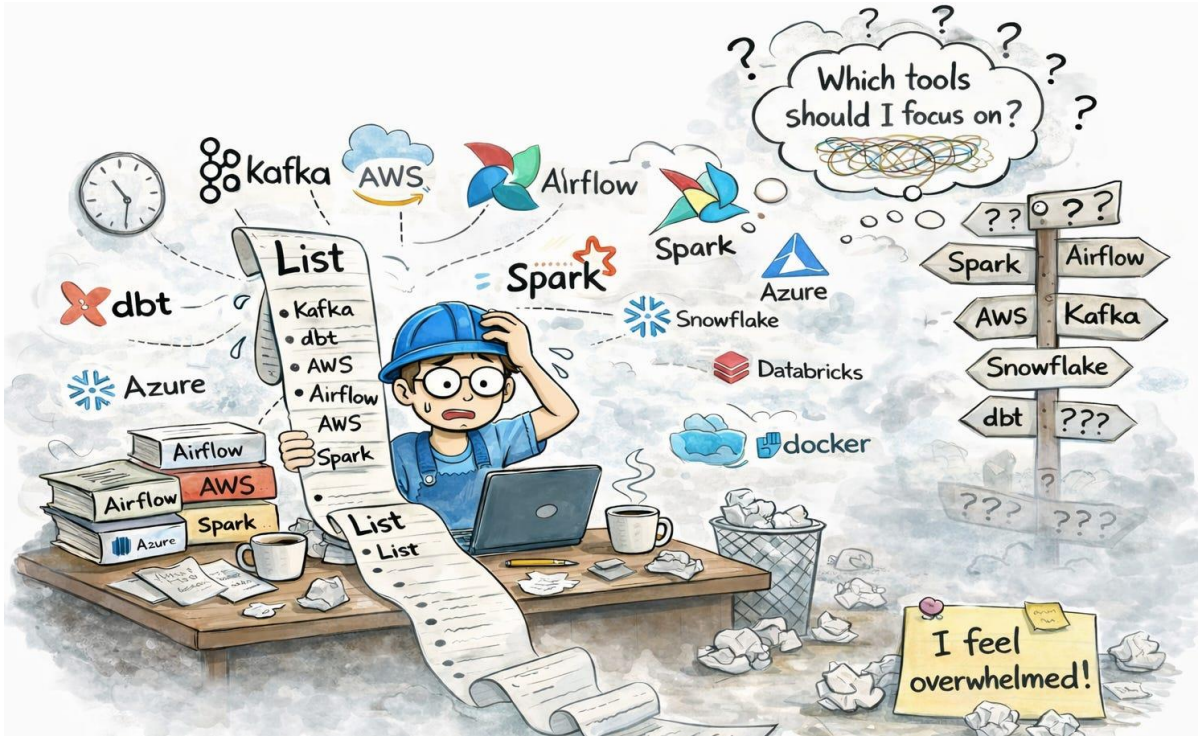
- The term was first coined by Andrej Karpathy in a February 2025 social media post:
  - Andrej Karpathy - AI researcher, OpenAI co-founder, former Tesla AI Director
  - https://karpathy.ai/
- It went viral almost overnight, signaling a new shift in how we code
  - A shift comparable to:
    - Assembly → Python / Java
- *From writing code → expressing intent*

https://x.com/karpathy/status/1886192184808149383

Jefferson Lab

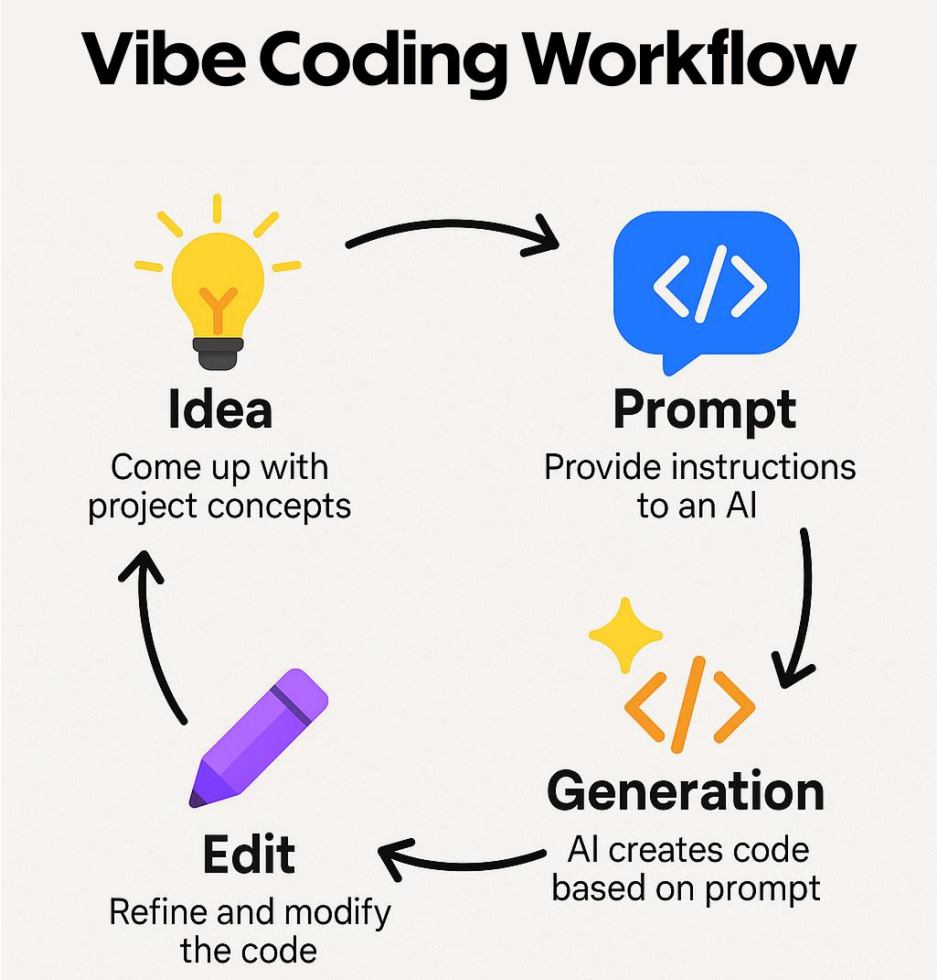
Traditional Coding (Old Way)

- You have a great idea → but first:
  - decide on framework
  - learn:
    - new languages
    - new tools etc.
  - fight syntax and debugging
- The idea waits…
  - while tooling slows you down.



VIBE CODING (NEW WAY)

- Just need one skill:
  - Clearly explaining what you want to AI
- Ideas go straight from mind → execution
- Languages & tools fade into the background
- Focus shifts from *how to code* → *what to build*

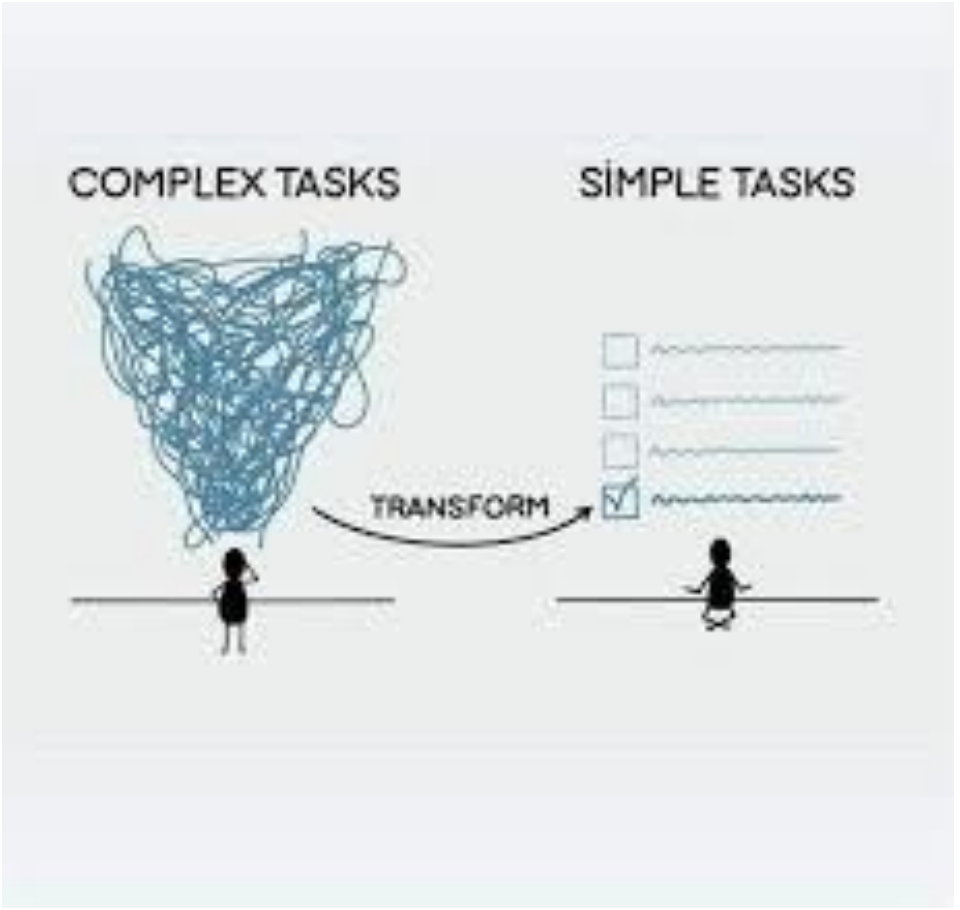


https://tinyurl.com/4c73rvr5

WHY PHYSICISTS WIN HERE

- Physicists already think in the language AI understands:
  - Break complex systems into simple parts
  - Define relationships precisely
  - Work from models, not syntax
- That's essentially prompt engineering.

- Your mental models map directly to AI systems.



Good Physicist = Great AI Collaborator

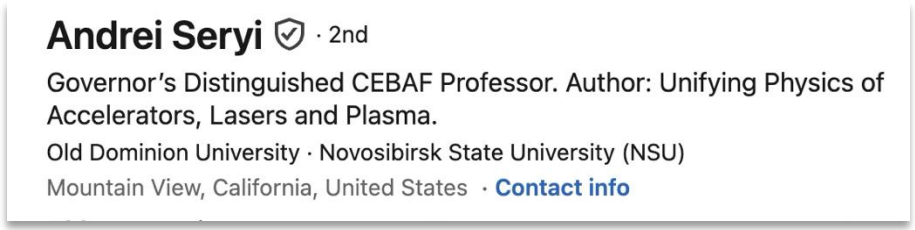
SO THE CONCLUSION IS..

- Vibe coding removes engineering friction.
- You focus on intent, not implementation.
- ***Less coding. More thinking.***
- Which is exactly why this is so powerful for physicists.
  - *Let's look at what they've already built with it.*

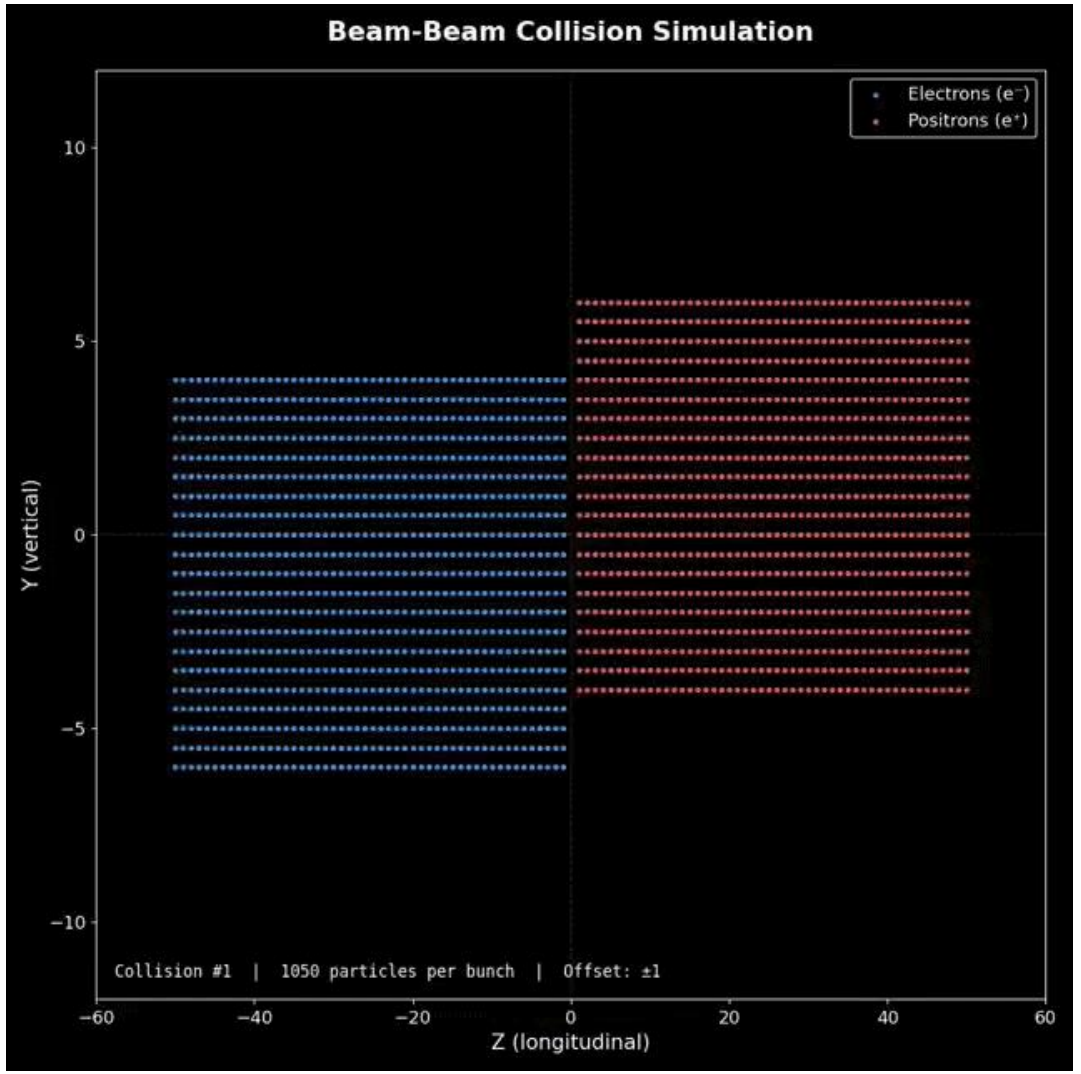
When Physicists Vibe

Accelerator physics collision simulator

- Simulates electron–positron beam collisions in a particle accelerator.
  - Beam offset interactions
  - Focusing forces
  - Instability behavior
- *Why tiny misalignments can destabilize entire collisions*



https://www.linkedin.com/in/andrei-seryi/



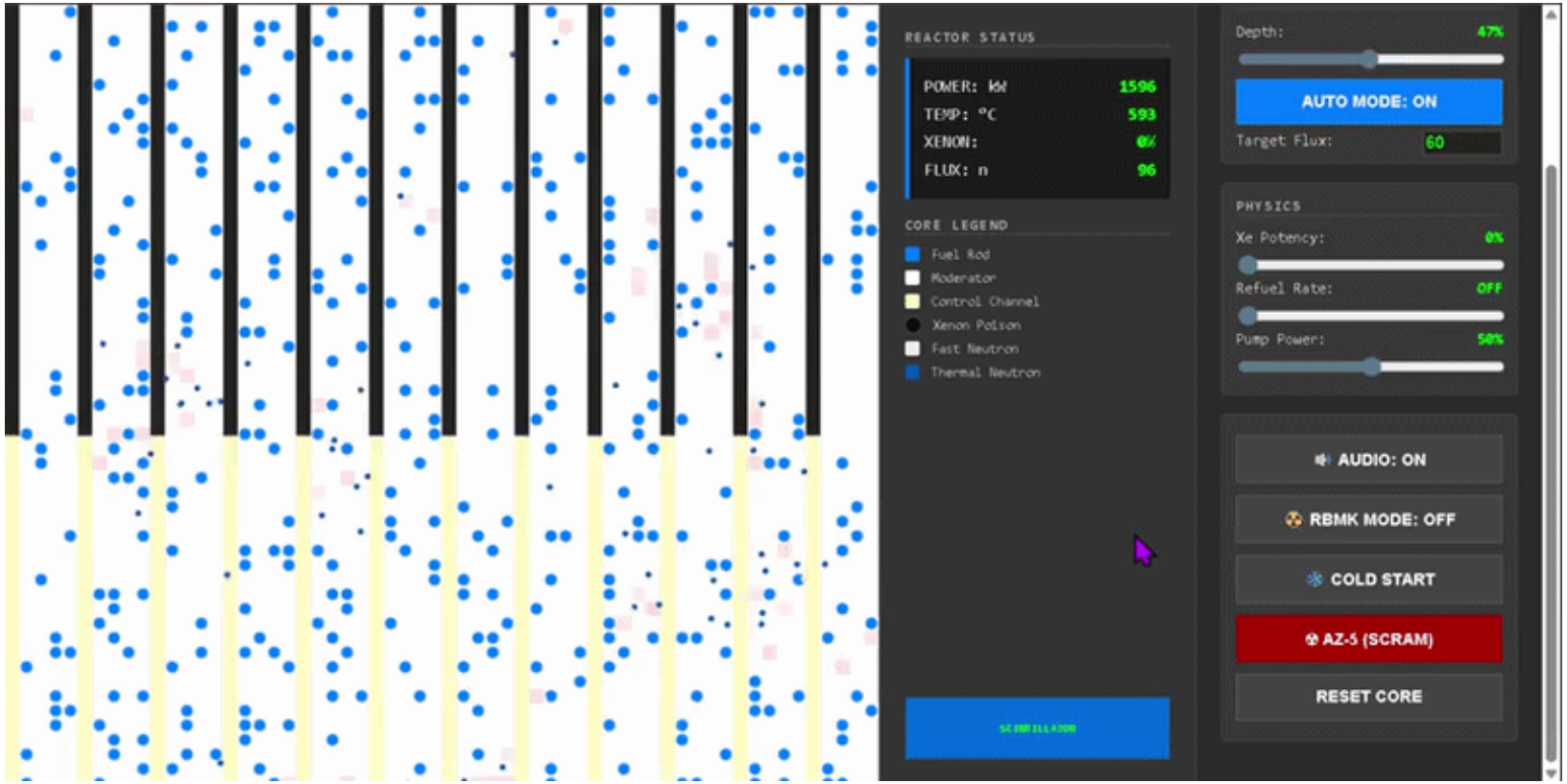
https://tinyurl.com/yeym9u29

Jefferson Lab

When Physicists Vibe

Nuclear reactor simulation tool

- A vibe-coded browser-based sandbox simulating nuclear fission in real time.
  - Models neutron movement, fuel rods, moderation, and chain reactions
- Lets users control reactor behavior via control rods, coolant flow, and fission rate
- Visual, interactive system for exploring reactor dynamics



https://tinyurl.com/y5u2666u

When Physicists Vibe

Three-body orbital simulator

- A vibe-coded simulation inspired by *The Three-Body Problem*.
  - Models three stars interacting under gravity
- Lets you vary mass, size, and initial conditions
- Shows how small changes can lead to stability or chaos
- *Explores why some celestial systems remain stable while others collapse*

https://tinyurl.com/53h743b4

Jefferson Lab

When Physicists Vibe

Quantum Mechanics Visualizer

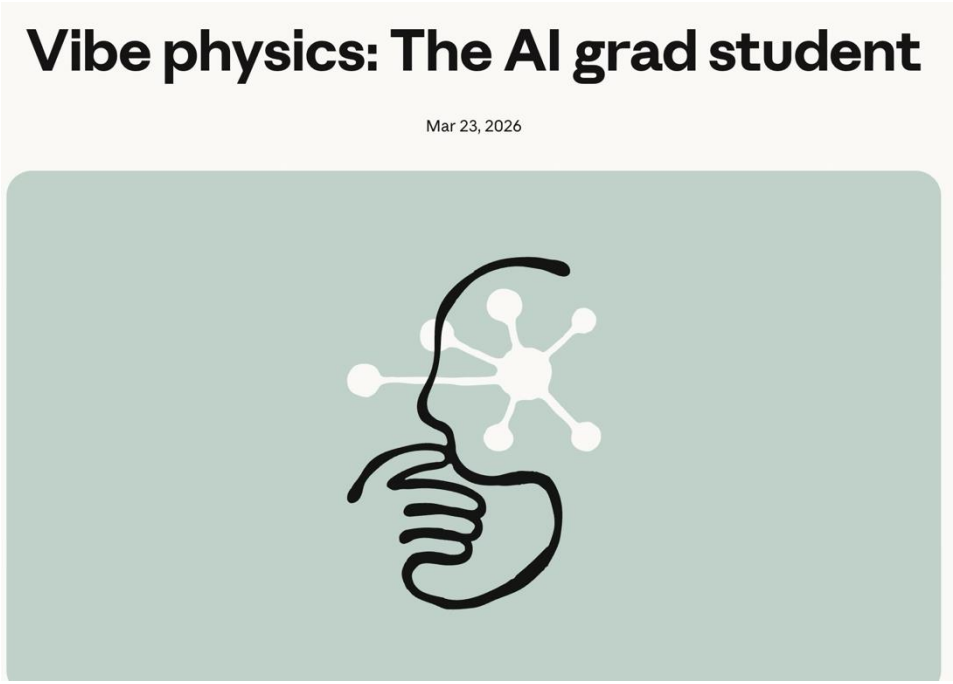
- A vibe-coded platform that makes quantum mechanics visually explorable.
  - Visualizes superposition, entanglement, tunneling, and wave–particle duality
- Real-time animations for intuitive learning
- *Turns abstract quantum theory into interactive experience*

https://tinyurl.com/2j42ybe2

When Physicists Vibe

Vibe physics: The AI grad student

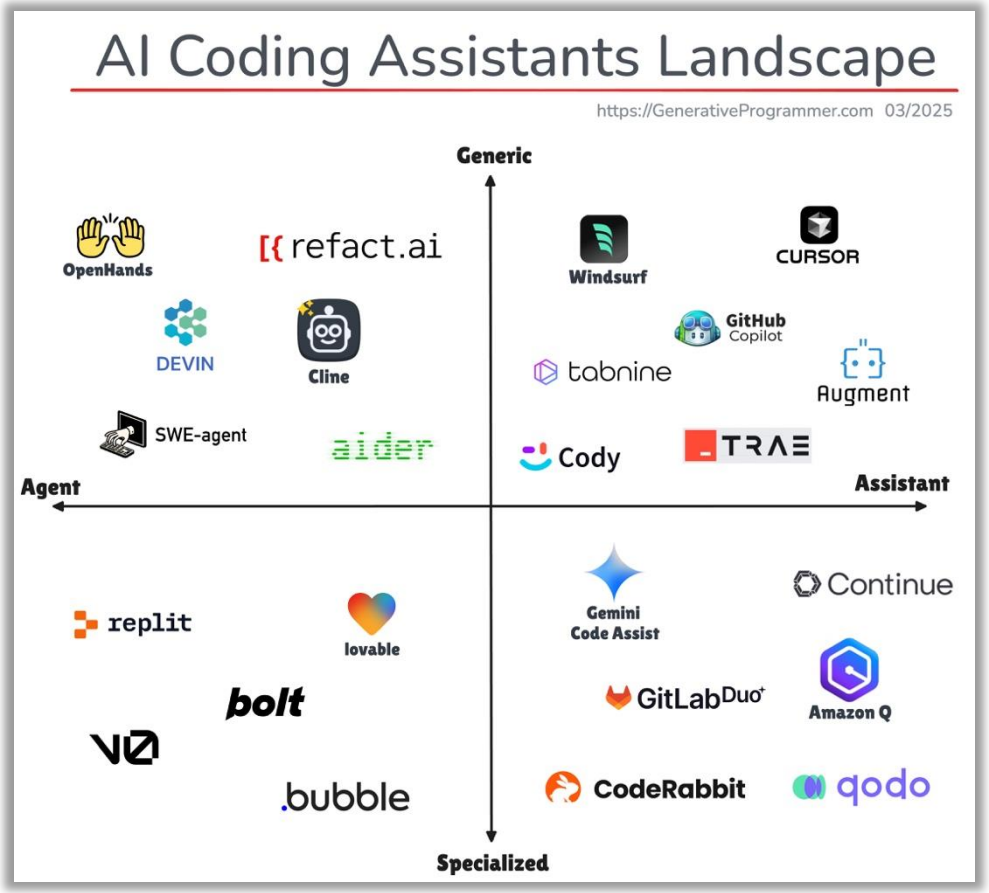
- An AI-assisted research workflow followed by a high-energy physicist Matthew Schwartz
  - AI helped with derivations, simulations, and paper drafting
  - Rapid research iteration with AI assistance
  - Continuous human verification and scientific oversight was required
- *Fast acceleration of research, but not independent scientific reasoning*



https://www.anthropic.com/research/vibe-physics

AI CODING TOOLS - CLASSIFICATION

- AI coding tools are evolving rapidly
  - New ones appear almost daily
- There's no single "correct" classification
- However, these two broad dimensions can help you pick tools based on your workflow needs:
  1. Specialized vs. Generic
  2. Autonomous vs. Assistant

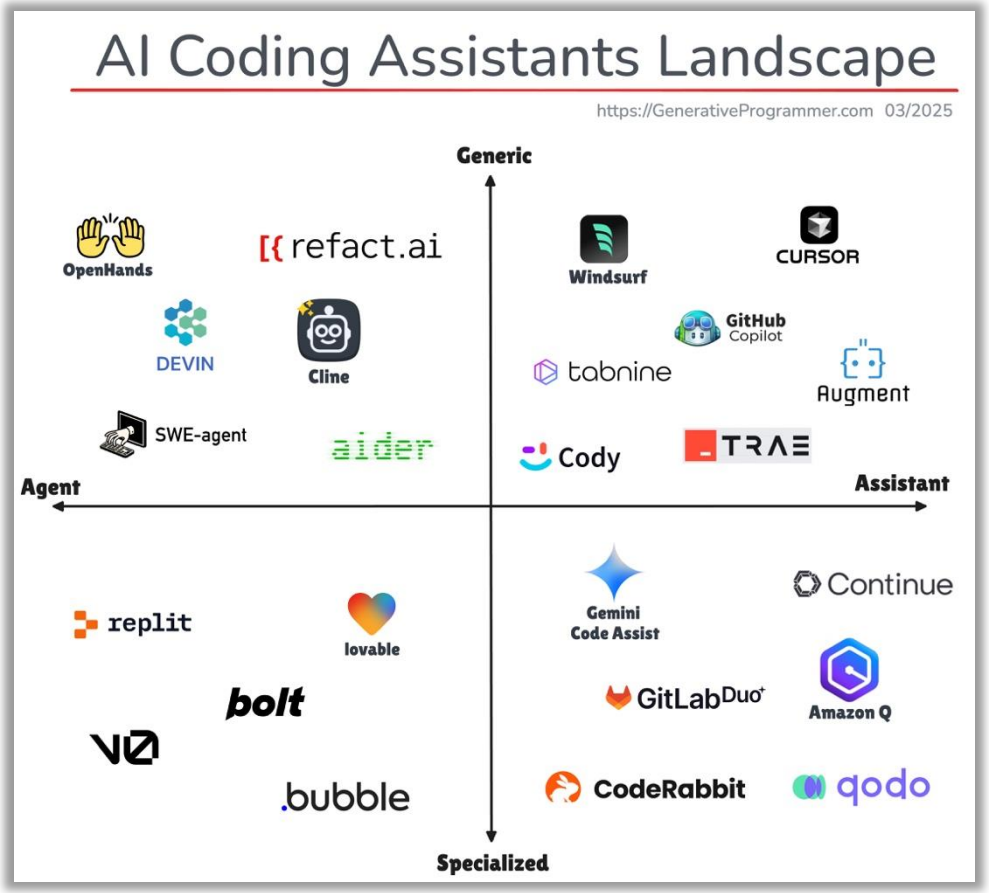


https://generativeprogrammer.com/p/ai-coding-assistants-landscape
March 3, 2025

AI CODING TOOLS- CLASSIFICATION

1. Specialized vs. Generic

- *Specialized Tools*
  - Built for specific tasks
  - Examples: test generation, PR reviews, UI scaffolding, cloud workflows
  - Best when you need to solve a narrow problem quickly
- *Generic Tools*
  - Broad, all-purpose assistants (IDE, CLI, chat)
  - Support: completion, refactoring, search, explanations
  - Best for day-to-day development across projects



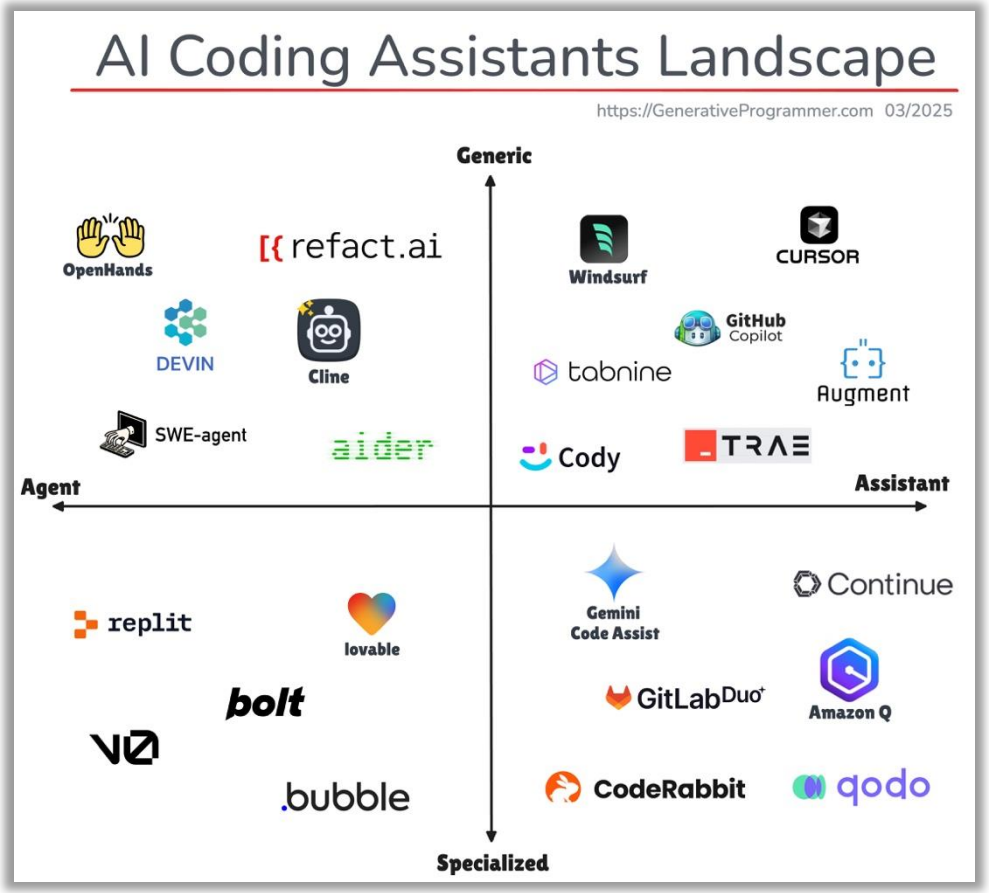
https://generativeprogrammer.com/p/ai-coding-assistants-landscape

March 3, 2025

AI CODING TOOLS - CLASSIFICATION

2. Autonomous vs. Assistant

- *Autonomous (Agentic) Tools*
  - Execute tasks end-to-end with minimal input
  - Can fix issues, run commands, validate code
  - Behave like a self-operating developer
- *Assistant Tools*
  - Reactive and collaborative
  - Suggest, explain, and generate code
  - Require human guidance and decision-making



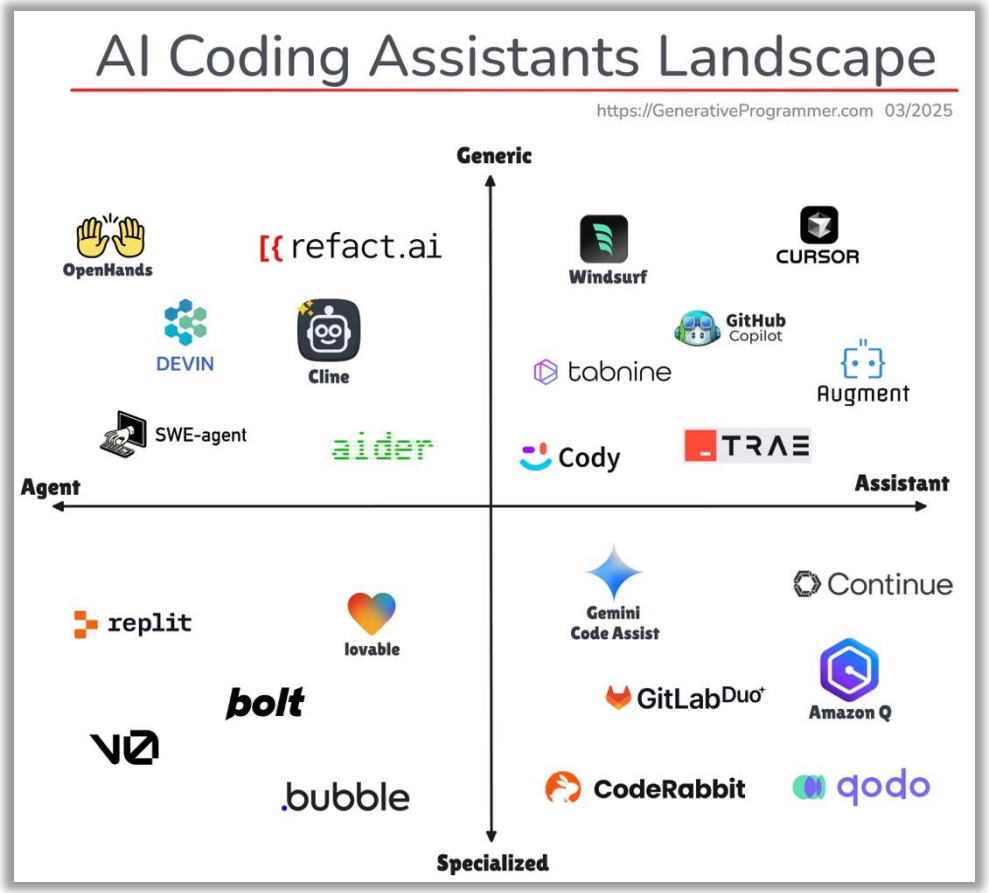
https://generativeprogrammer.com/p/ai-coding-assistants-landscape
March 3, 2025

AI CODING TOOLS - CLASSIFICATION

2. Autonomous vs. Assistant

Reality: It's a Spectrum

- Many tools blend both models
- Example: IDE assistants that can also run autonomous workflows
  - (e.g., Cursor, Windsurf)
- *The trend is increasingly moving toward agentic systems:*
  - Tools can independently handle larger parts of the workflow while the user provides high-level guidance.



https://generativeprogrammer.com/p/ai-coding-assistants-landscape

March 3, 2025

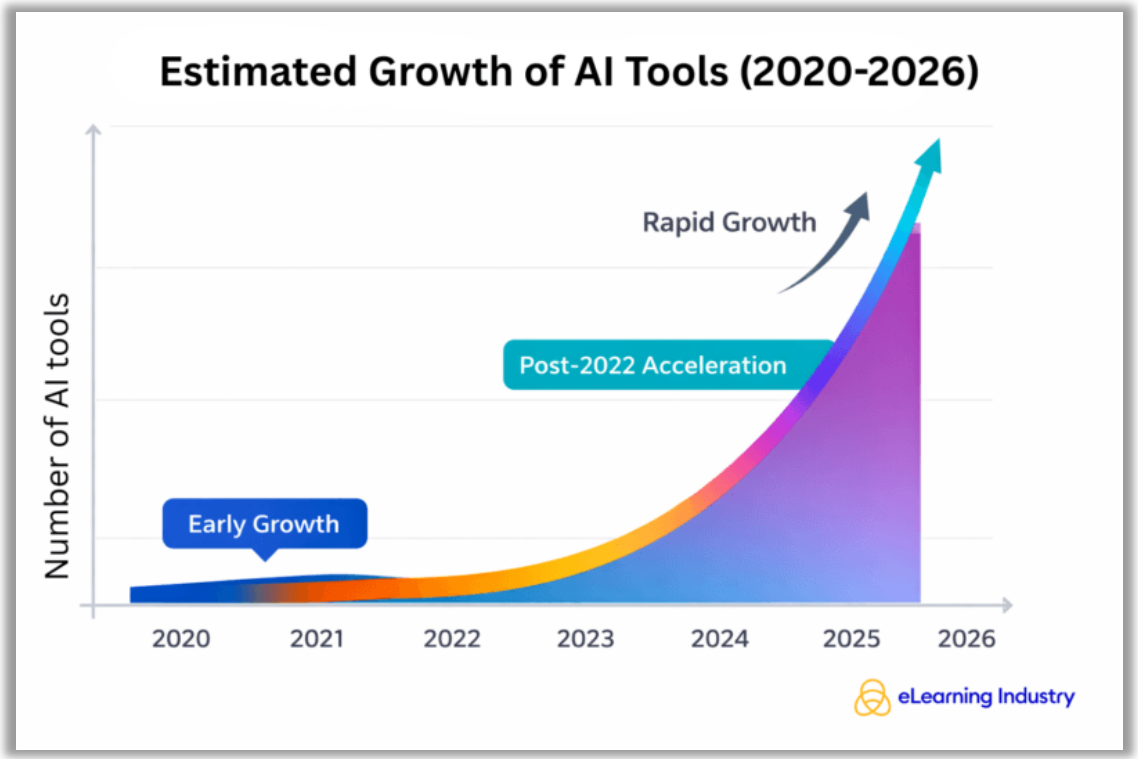
Jefferson Lab

AI Coding Tools - Choosing the Right One

- The AI coding tools landscape is evolving extremely fast
- New tools emerge constantly — *any list is outdated almost immediately*
- Even tools shown earlier (from March 2025) no longer reflect the current state

Tool Rankings:

- Different blogs and benchmarks rank tools differently
- Rankings depend heavily on the evaluation criteria used
- There is no universal definition of the “best” tool



https://tinyurl.com/2k9z42a4

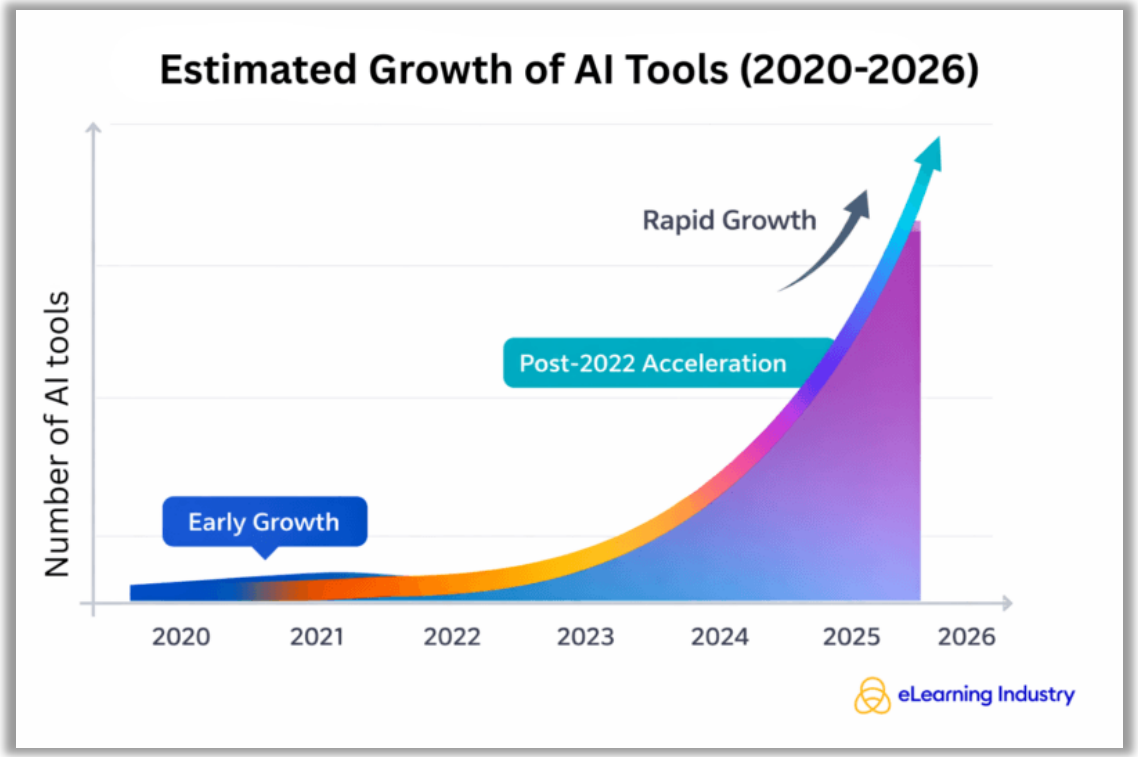
AI Coding Tools - Choosing the Right One

What This Means for You

- Rankings and lists provided here are reference points, not answers
- There is no one-size-fits-all tool

Key Takeaway

- The “best” tool depends on:
  - Your workflow
  - Your use case
  - Your comfort with control vs. automation
- *Your job isn’t to find the best tool*
- *It’s to find what works best for you*



https://tinyurl.com/2k9z42a4

Jefferson Lab

AI CODING TOOLS – 2026 (VERDENT)

Tool	Best For	SWE-bench Score	Price	IDE Support
GitHub Copilot	General development	12.3%	$10-19/mo	VS Code, JetBrains, Neovim
Claude Code	Terminal workflows	N/A (CLI tool)	Free (API costs)	Terminal/CLI
Cursor	AI-native experience	~40% (estimated)	$20/mo	Built-in IDE
Verdent	Enterprise projects	76.1%	$19-179/mo	VS Code, JetBrains
Codeium	Free option	~35% (estimated)	Free-$12/mo	40+ IDEs
Tabnine	Privacy-first teams	N/A (local models)	$12-39/mo	Most IDEs
Amazon Q	AWS workflows	N/A	$19/mo	VS Code, JetBrains
Cody	Large codebases	N/A	Free-$9/mo	VS Code, JetBrains

https://www.verdent.ai/guides/best-ai-coding-assistant-2026 - Feb 5, 2026

AI CODING TOOLS – 2026 (TECH INSIDER)

AI CODING TOOL	MARKET SHARE (2026)	MONTHLY ACTIVE DEVELOPERS	PRICING (PRO TIER)
GitHub Copilot X	37%	28 million	$19/month
Cursor	18%	14 million	$20/month
Codeium / Windsurf	12%	9.5 million	$15/month
Amazon Q Developer	10%	7.8 million	$19/month
Google Gemini Code Assist	9%	7.2 million	$22/month
Tabnine Enterprise	5%	3.9 million	$39/month
Open Source Tools (Combined)	9%	7.1 million	Free

https://tech-insider.org/ai-coding-tools-2026-transforming-software-development/ - March 15, 2026

AI CODING TOOLS – 2026 (LAUNCHPAD)

Tool	Type	Key selling point	Exists as	Free plan	Price
Launchpad	Platform	Pre-built enterprise infrastructure for B2B SaaS	Cloud platform	Yes	$900/mo
GitHub Copilot	Coding Tool	Microsoft-backed with tight VSCode integration	IDE extension	Yes	$10/mo
Cursor	Coding Tool	Agent-first IDE supporting 8 simultaneous agents	Standalone IDE	Yes	$20/mo
Windsurf	Coding Tool	Clean UI with Cascade interface for agentic workflows	Standalone IDE + Extensions	Yes	$15/mo
Claude Code	Coding Tool	Terminal-based agentic coding with full autonomy	Command-line tool	No	$20/mo
Tabnine	Coding Tool	Privacy-first with local execution	IDE extension	Yes	$59/user per month

https://launchpad.io/blog/22-best-ai-coding-tools-speed-development-2026 - Jan 21, 2026

AI CODING TOOLS – 2026 (LAUNCHPAD)

Tool	Type	Key selling point	Exists as	Free plan	Price
OpenAI Codex	Coding Tool	Strong at understanding complex prompts	ChatGPT, API	No	$20/mo
Google Gemini	Coding Tool	Embedded across Google tools	Cloud API + IDE integrations	Yes	$22.80/user per mo
Replit	Platform	Cloud IDE with autonomous agent	Cloud IDE + Platform	Yes	$25/mo
Cline	Coding Tool	Open-source VSCode extension	IDE extension	Yes	$20/user per month
Amazon Q Developer	Coding Tool	Deep AWS integration	IDE extension + AWS Console	Yes	$19/user per month
Zed	Coding Tool	Ultra-fast (120 fps) Rust-based editor	Standalone IDE	Yes	$10/mo

https://launchpad.io/blog/22-best-ai-coding-tools-speed-development-2026 - Jan 21, 2026

AI CODING TOOLS – 2026 (LAUNCHPAD)

Tool	Description
Aider	Command-line AI coding assistant that edits code in your local git repository
Bolt.new	Browser-based full-stack app builder
Qodo	AI-powered code review platform with 15+ agentic workflows
JetBrains AI Assistant	Native AI integration across JetBrains IDEs
Codiga	Static code analyzer and snippet manager powered by AI
AskCodi	AI coding assistant with task-specific agents
CodeGPT	Bring-your-own-API-key AI coding assistant for IDEs
CodeRabbit	AI code review tool
Kiro	Agentic AI-powered IDE
Devin AI	First fully autonomous AI software engineer

https://launchpad.io/blog/22-best-ai-coding-tools-speed-development-2026 - Jan 21, 2026

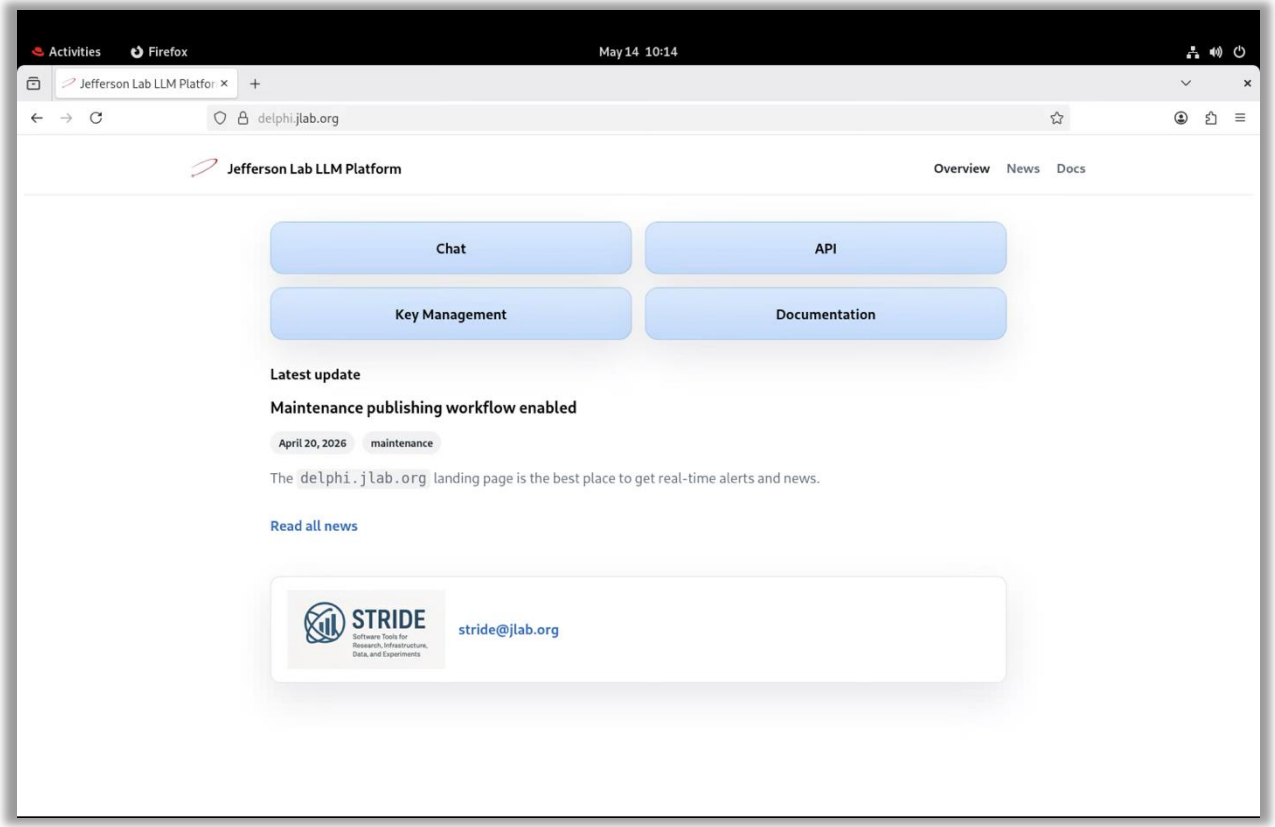
JLAB LLM UPDATES

AI Models Hosted on JLab Resources

- delphi.jlab.org provides details on current LLM models deployed on JLab infrastructure.
  - *Current Status*
    - Platform is currently in beta / active development
    - Heavy multi-user load handling is still being evaluated
    - Because of this, we are not relying on it for today's live demo
    - Access is currently limited to JLab systems only
      - How to Access Delphi
        - Login to VDI (https://vdi.jlab.org/)
        - Open terminal
        - Run command: `firefox`
        - Once firefox is open, enter this url:
          - https://delphi.jlab.org
  - *Future Goal:*
    - Once performance and scalability are fully validated, broader external access might be enabled outside JLab systems.
- *Acknowledgment*
  - Special thanks to Casey Morean, Physics AI / ML Data Scientist, for delphi.

DELPHI.JLAB.ORG — QUICK OVERVIEW

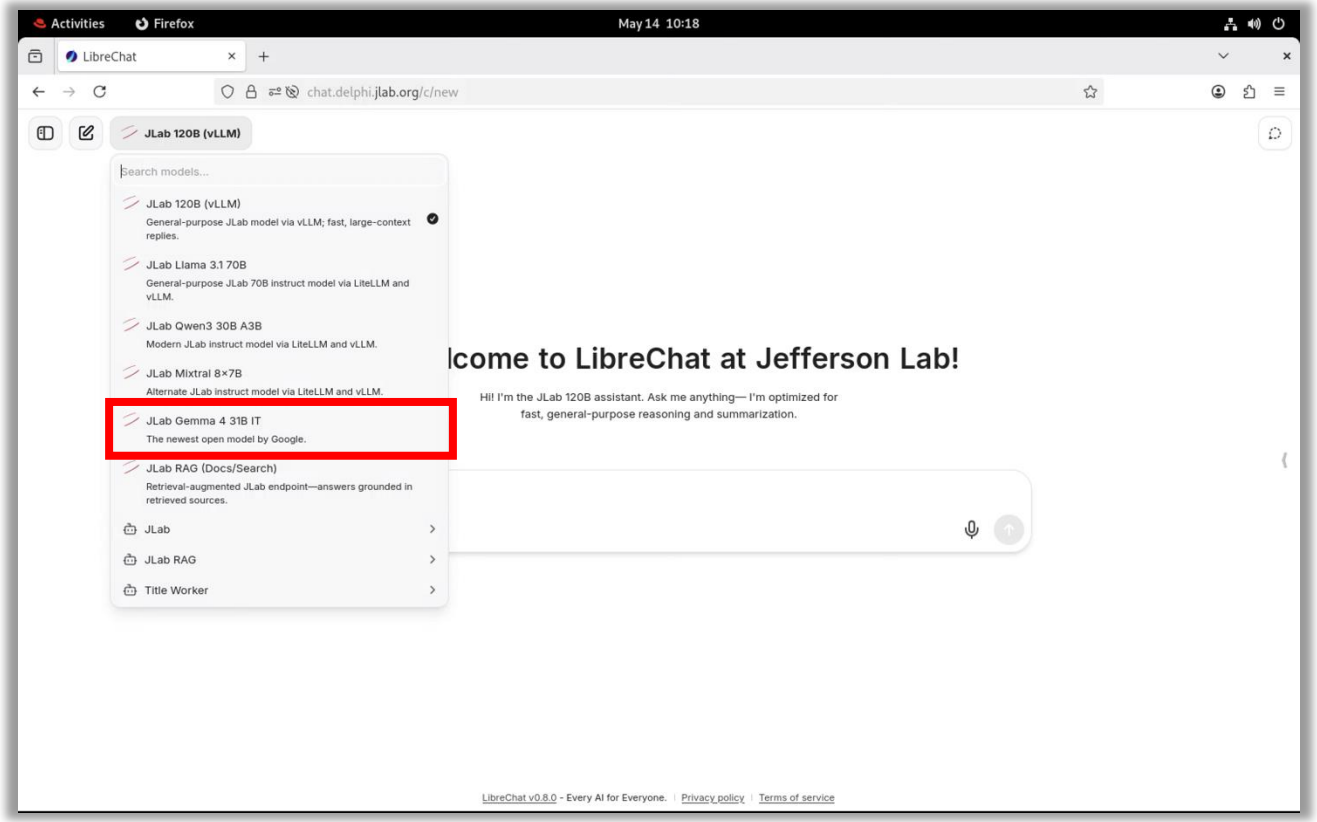
- Main Dashboard
  - After logging into delphi.jlab.org, you will land on the main dashboard.
  - What's Available
    - Central access point for deployed LLM models
    - Chat interface for interacting with models
    - API access for development workflows
    - Documentation and key management
    - Deployed models' status and outage updates



Jefferson Lab

DELPHI.JLAB.ORG — QUICK OVERVIEW

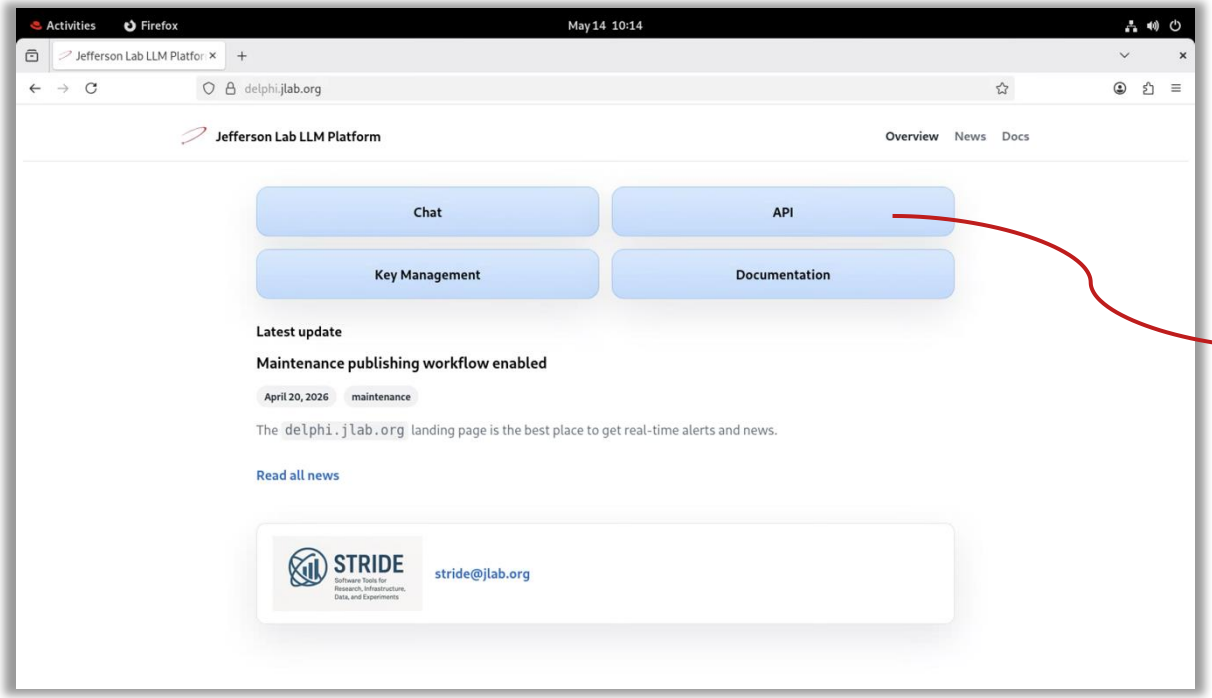
- Chat Interface & Models
  - Clicking “Chat” opens the AI chat interface with currently deployed models.
  - *Latest Available Model*
    - Gemma 4 31B IT
    - Released by Google DeepMind
    - Part of the Gemma family of open-weight language models
    - “31B IT” = 31 billion parameter Instruction-Tuned model
    - First deployed model at JLab with image understanding capabilities

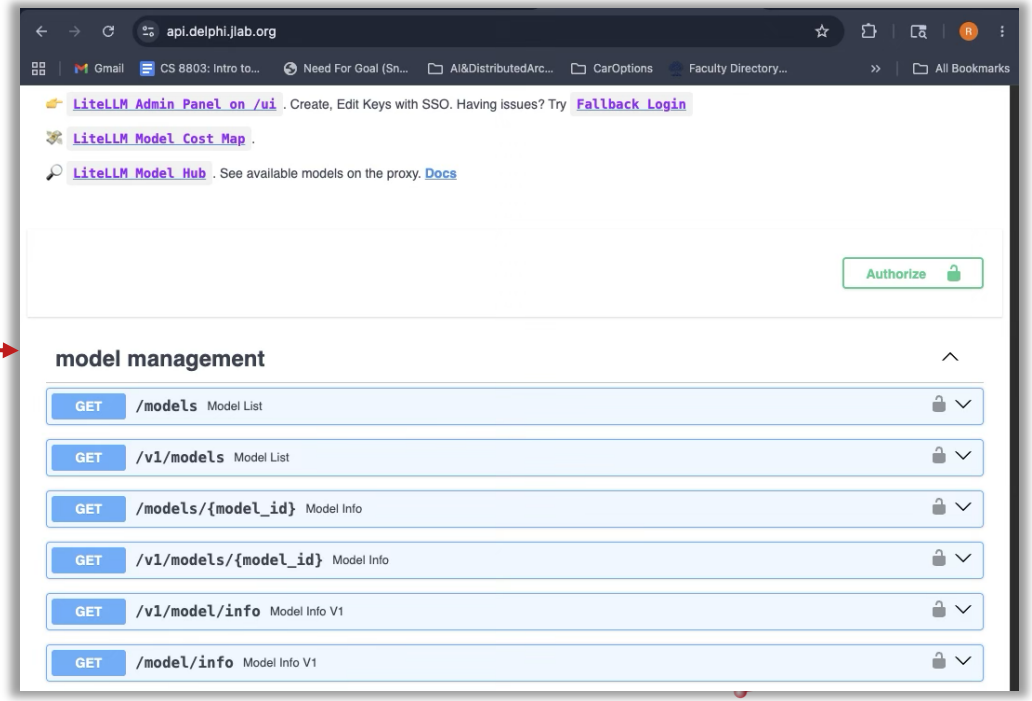


Jefferson Lab

DELPHI.JLAB.ORG — QUICK OVERVIEW

- API Access
  - Delphi can also be accessed programmatically through APIs, enabling integration into development workflows and automation tools.
  - Key Resources
    - API reference
    - Key Management
    - Documentation
- These sections provide everything needed for setup, authentication, and integration.





DELPHI.JLAB.ORG — QUICK OVERVIEW

- API Access & VS Code Integration (Agent Workflow)
  - You can connect the API directly into VS Code to use models as coding assistants or agents during development.
    - Resources
      - Roo Code Documentation
      - Roo Code GitHub
      - VS Code Insiders Download
  - What This Enables
    - AI-assisted coding directly inside VS Code
    - Agent-style workflows (multi-step coding tasks)
    - Faster iteration during development
    - Integration with JLab-hosted models via API

To Sum up…

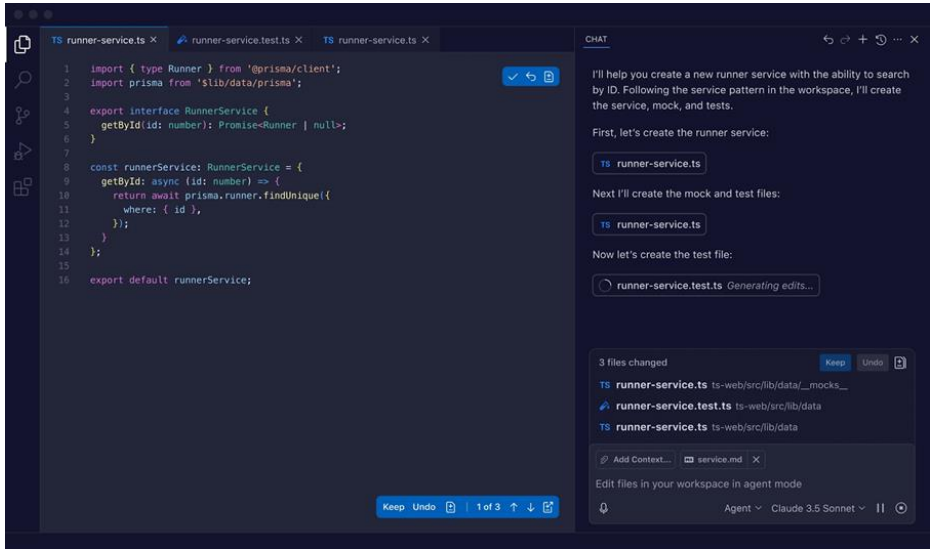
- delphi.jlab.org is currently a beta platform
  - Performance may differ from industry tools like GitHub Copilot
    - Expect occasional limitations in:
      - speed
      - stability
      - user experience
    - This is normal, the system is actively evolving
  - Reporting Issues
    - If something looks wrong or broken:
      - Email: stride@jlab.org
- Efforts are also ongoing toward access to top-tier industry models.
  - Example: Google Gemini models
- Stay Updated
  - To keep up with future updates, and new deployments, subscribe to: JLab SciComp Briefs

WHERE TO START

- As you can already see, there are many AI coding tools out there, and it can feel overwhelming.
- To make it easier, here are three commonly used tools today:
  - GitHub Copilot
  - Cursor
  - Claude Code

- ***Where We Will Start Today***
  - We will start with **GitHub Copilot** in VS Code because:
    - It's great for daily development
    - Easy to install, easy to use, zero friction.
    - Next slides will focus on setting it up in VS Code.
  - For Terminal Users
    - GitHub Copilot CLI is also available https://github.com/github/copilot-cli
    - However, for this session, we will use the VS Code chat interface for hands-on practice.

GitHub Copilot CLI



GitHub Copilot VsCode

Jefferson Lab

HANDS-ON SETUP SESSION

GET SET UP TO VIBE

- Before we are ready for vibe coding, we need to complete the required setup.
- We will be doing all coding after SSH'ing into ifarm, since it already has Git installed (which will be our best friend during iterative coding and changes).
- We will mainly go through these steps:
  1. Install VS Code
  2. Sign in to GitHub Copilot
  3. Understand GitHub Copilot Chat Interface
  4. Install VS Code Remote SSH Extension
  5. Configure SSH
  6. Connect to ifarm
  7. Open folder on ifarm
  8. Open chat in the folder
  9. X11 Forwarding

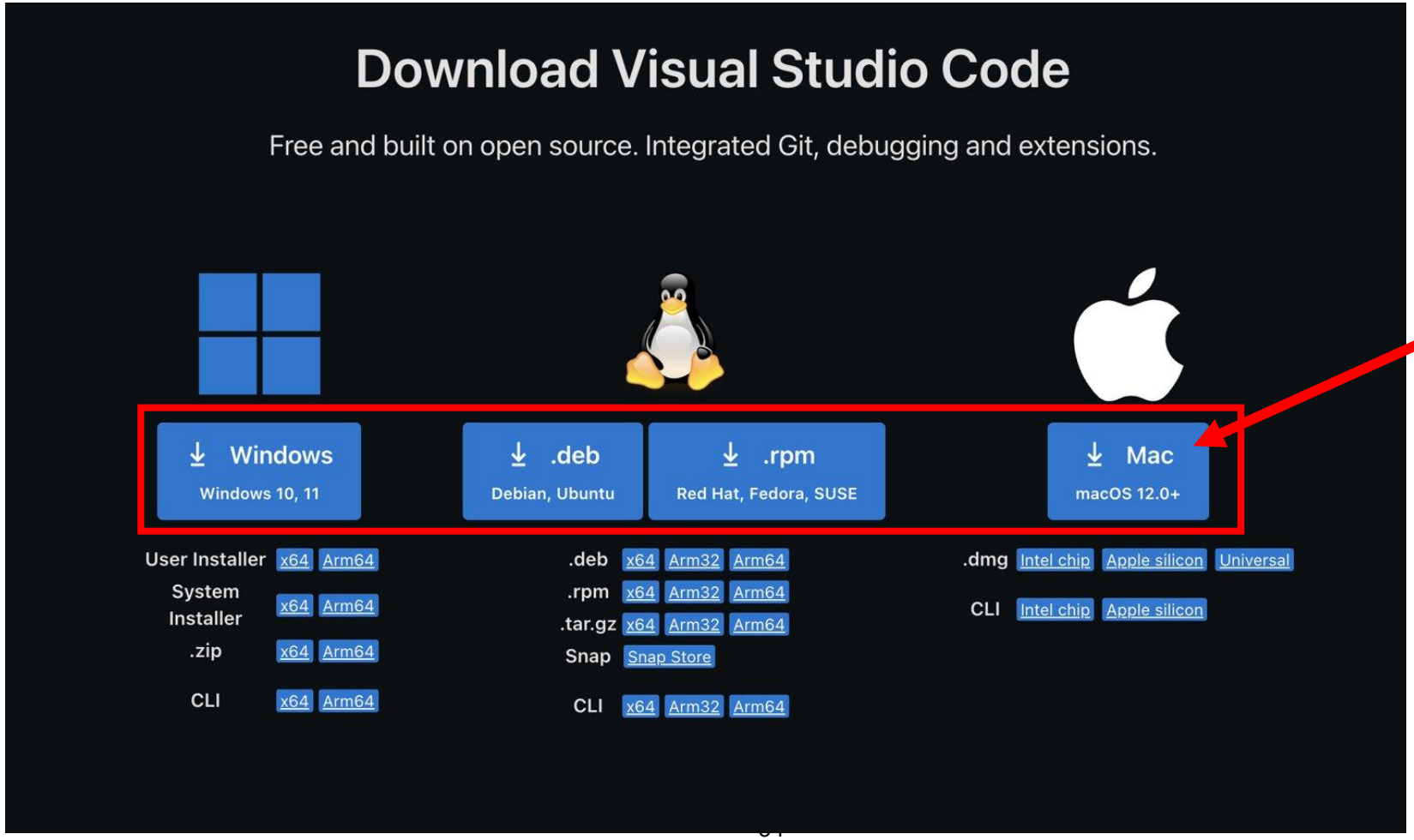
Required for working on ifarm

We will briefly cover SSH access to ifarm during this session but for detailed instructions, refer to the previous session: https://indico.jlab.org/event/1069/attachments/13992/22682/AccessingAndNavigatingIFarm.pdf

STEP 1: INSTALL VSCODE

GET SET UP TO VIBE - STEP 1: INSTALL VS CODE

- Download VS Code based on your OS: https://code.visualstudio.com/download
- Install it by clicking the downloaded file and following the setup instructions

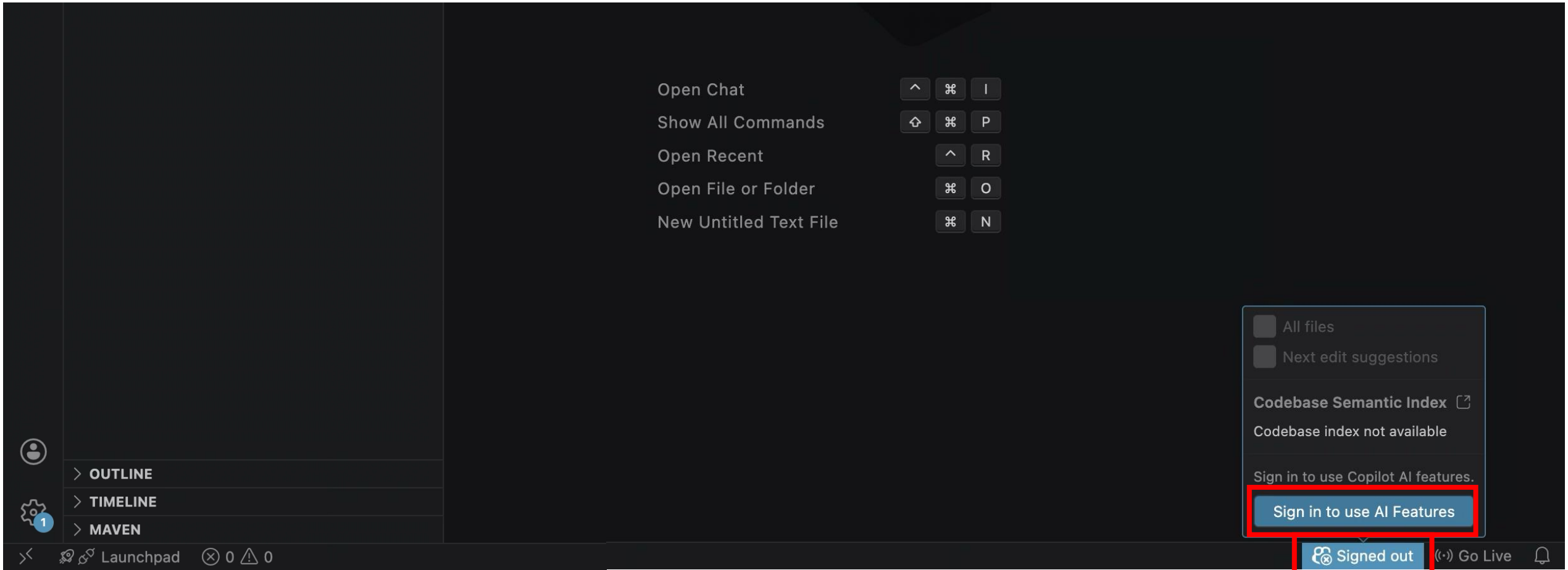


Click one of these with download icon based on your OS

STEP 2: SIGN IN TO GITHUB COPILOT

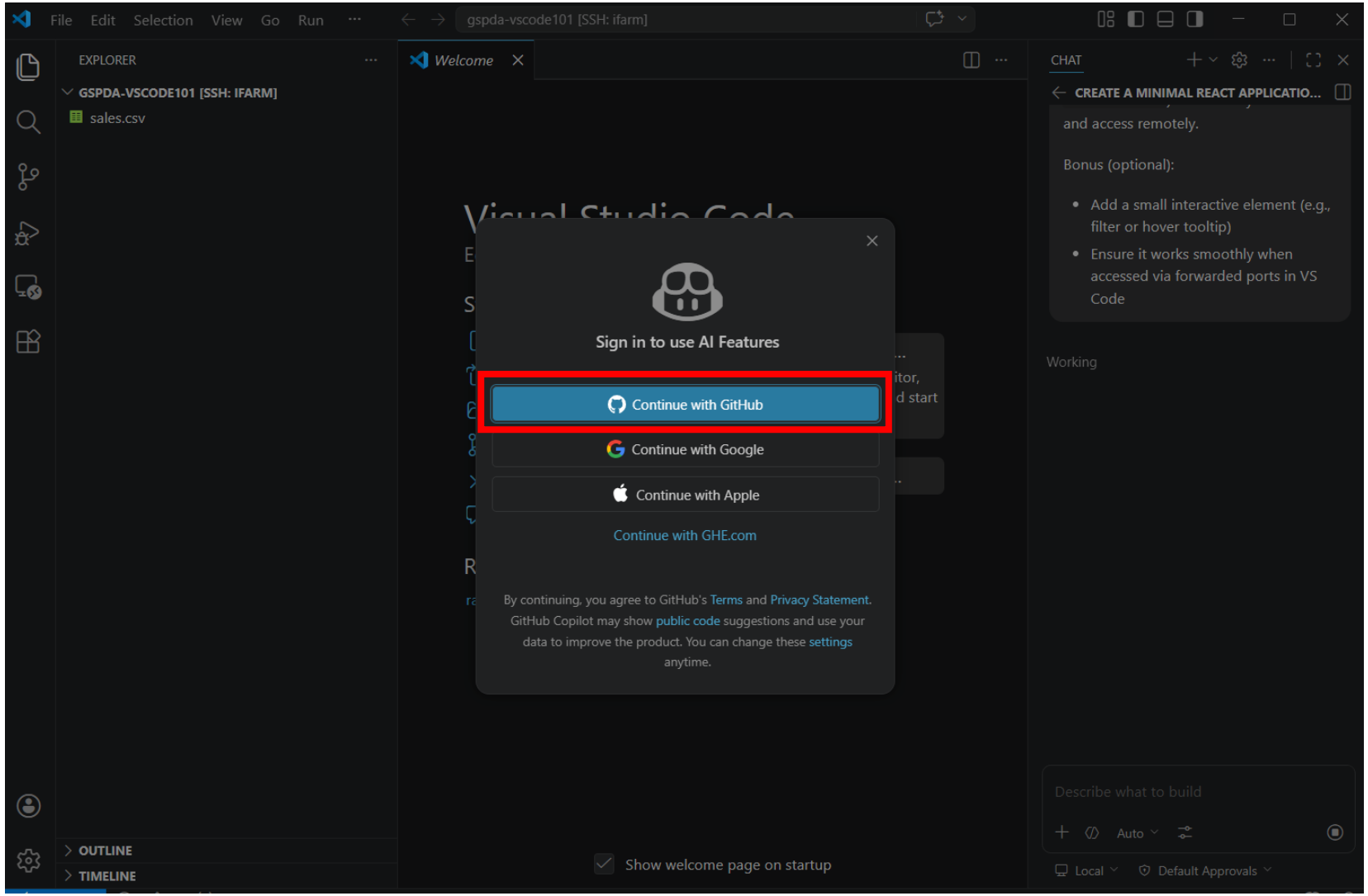
GET SET UP TO VIBE - STEP 2: SIGN IN TO GITHUB COPILOT

- VS Code already has GitHub Copilot available, so no installation is needed.
- However, before you can start using GitHub Copilot, you need to sign in.
  - Go to the bottom-right corner of VS Code
  - Hover over the GitHub Copilot icon
  - A prompt will appear: “Sign in to use AI features”. Click it and complete the sign-in process



GET SET UP TO VIBE - STEP 2: SIGN IN TO GITHUB COPILOT

- Sign in using any method you prefer
  - Recommended: Sign in with GitHub (we will use this in the next slides)



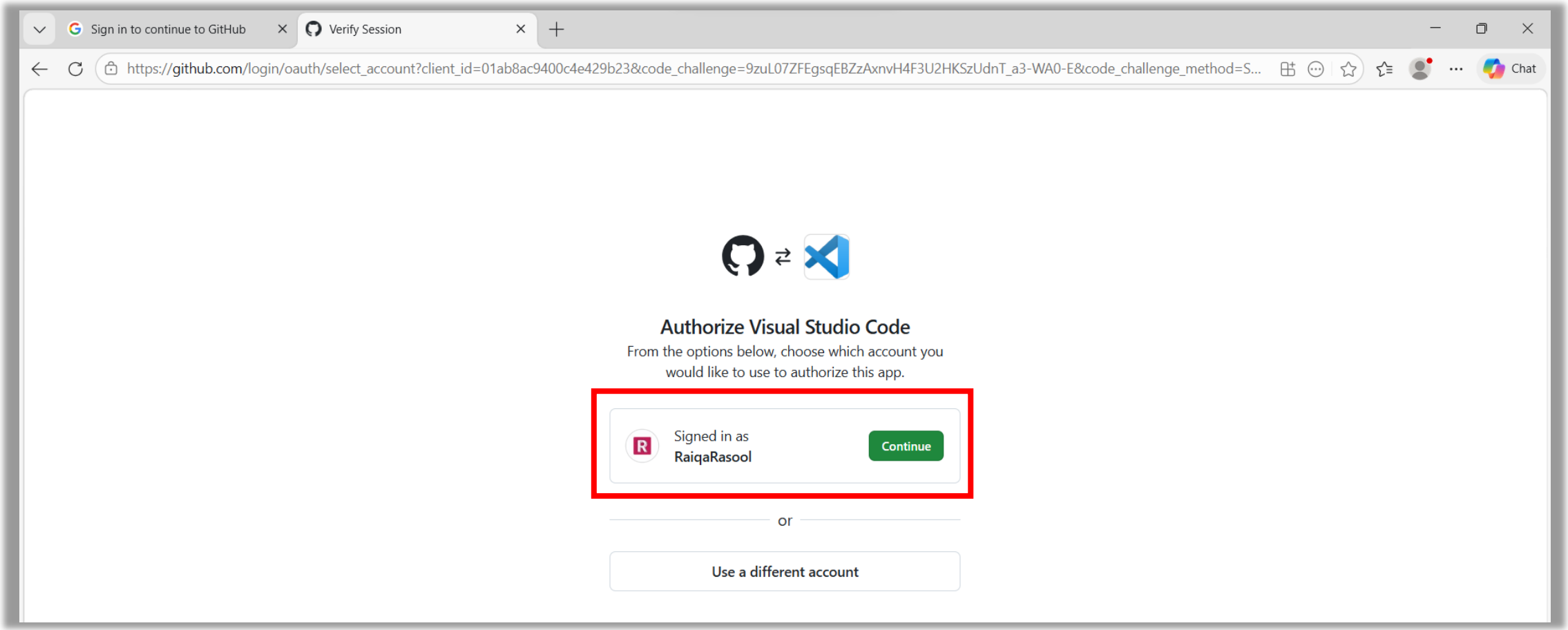
Note for Students:
If you are a student, make sure to use a GitHub account associated with your student email. This can give you access to the free GitHub Copilot Pro. Setup instructions and eligibility details: https://docs.github.com/en/copilot/how-tos/copilot-on-github/set-up-copilot/enable-copilot/set-up-for-students

Important: Starting April 20, 2026, new sign-ups for Copilot Pro, Copilot Pro+, and student plans are temporarily paused.

Jefferson Lab

GET SET UP TO VIBE - STEP 2: SIGN IN TO GITHUB COPILOT

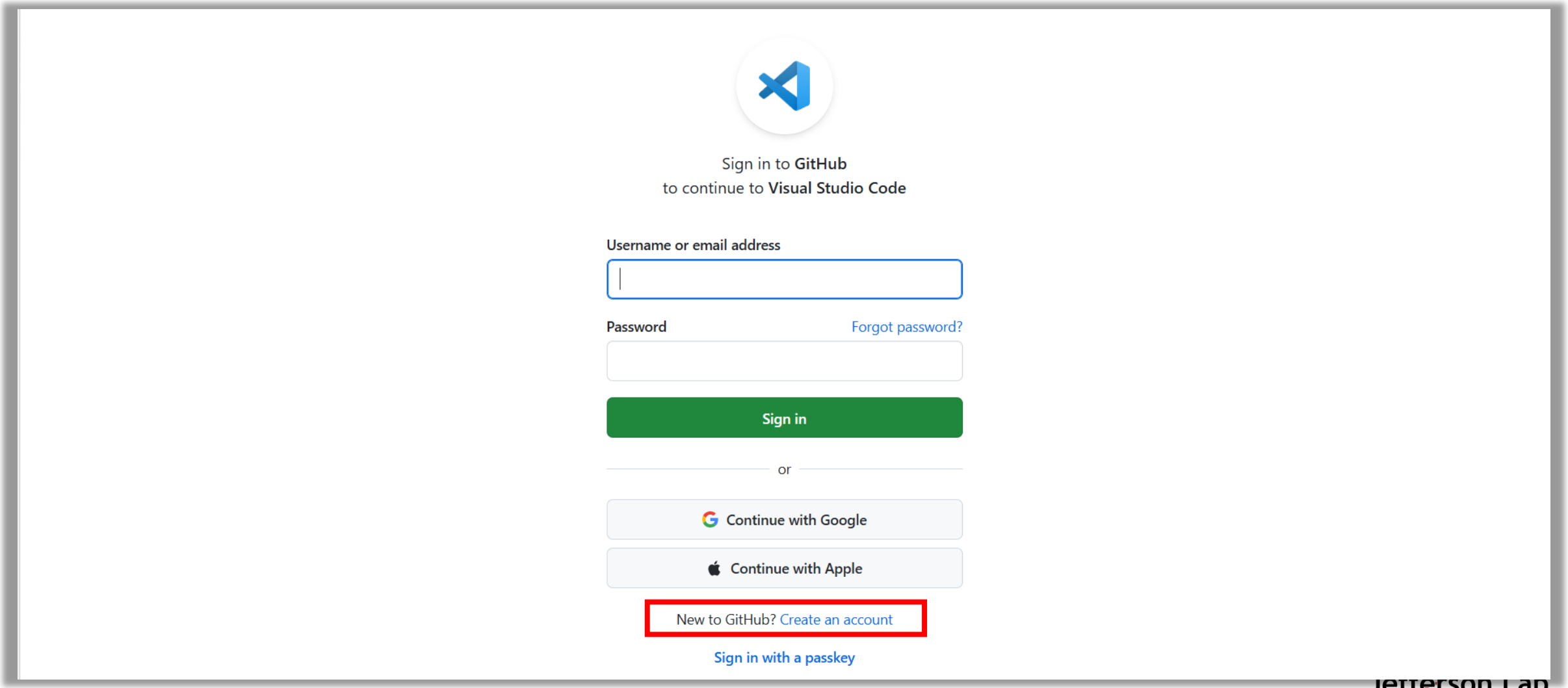
- Sign in using your GitHub account by clicking “Continue”
- If you don’t have a GitHub account yet, you will be redirected to quickly create one
- Once created, you can link it and continue the sign-in process



Jefferson Lab

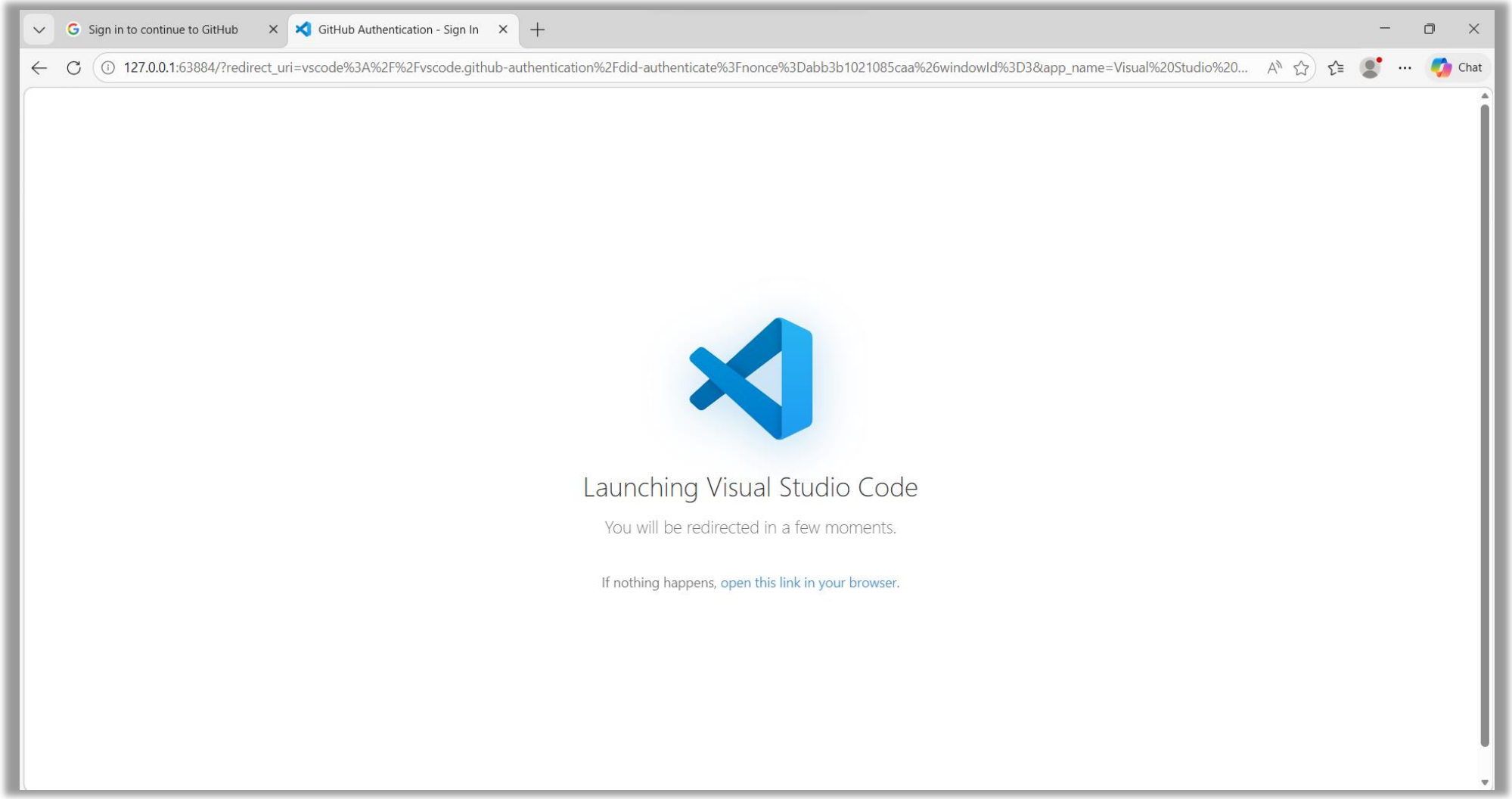
GET SET UP TO VIBE - STEP 2: SIGN IN TO GITHUB COPILOT

- Enter your username and password if prompted
- If you don't already have an account, create a new one and link it



GET SET UP TO VIBE - STEP 2: SIGN IN TO GITHUB COPILOT

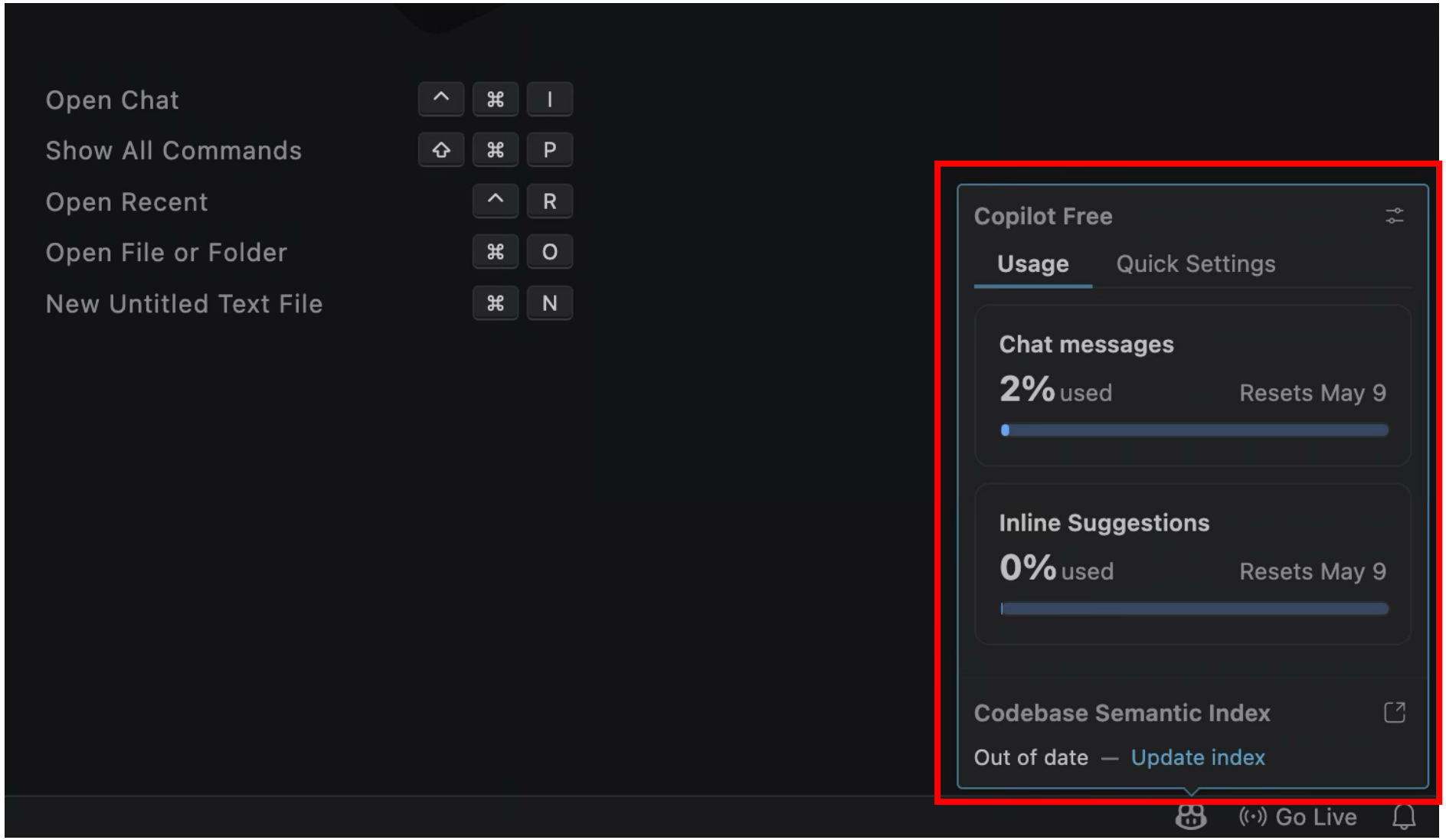
- Once linked you will be redirected to VS Code



Jefferson Lab

GET SET UP TO VIBE - STEP 3: SIGN IN TO GITHUB COPILOT

- After getting redirected, hover over the GitHub Copilot icon again. You should now see your usage status / activity stats. This confirms Copilot is successfully enabled in VS Code



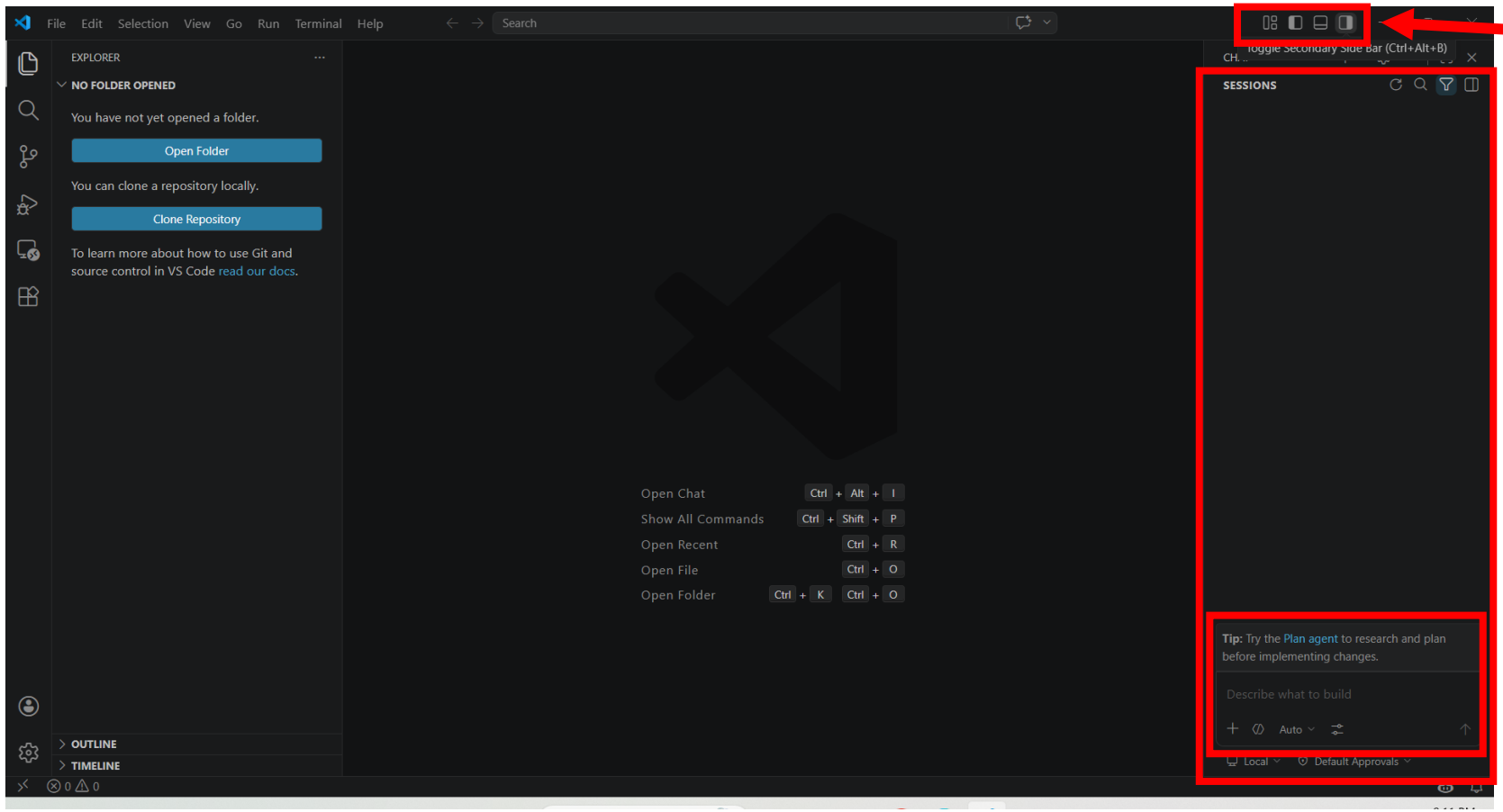
Jefferson Lab

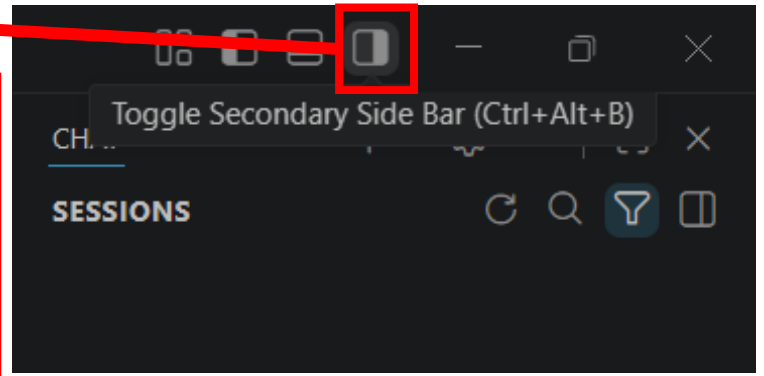
STEP 3: UNDERSTAND CHAT INTERFACE

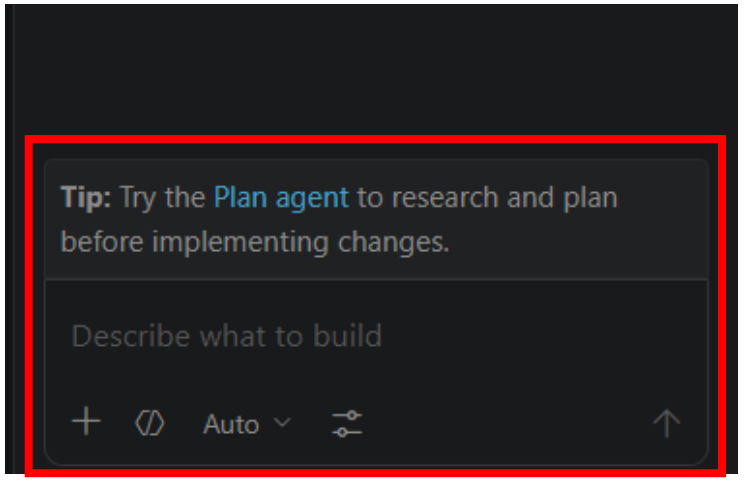
GET SET UP TO VIBE - STEP 3: UNDERSTAND CHAT INTERFACE

Now Click the Secondary Sidebar toggle icon in the top-right corner or use the shortcut: Ctrl/Cmd + Alt + B

- This will open the GitHub Copilot chat interface in VS Code.





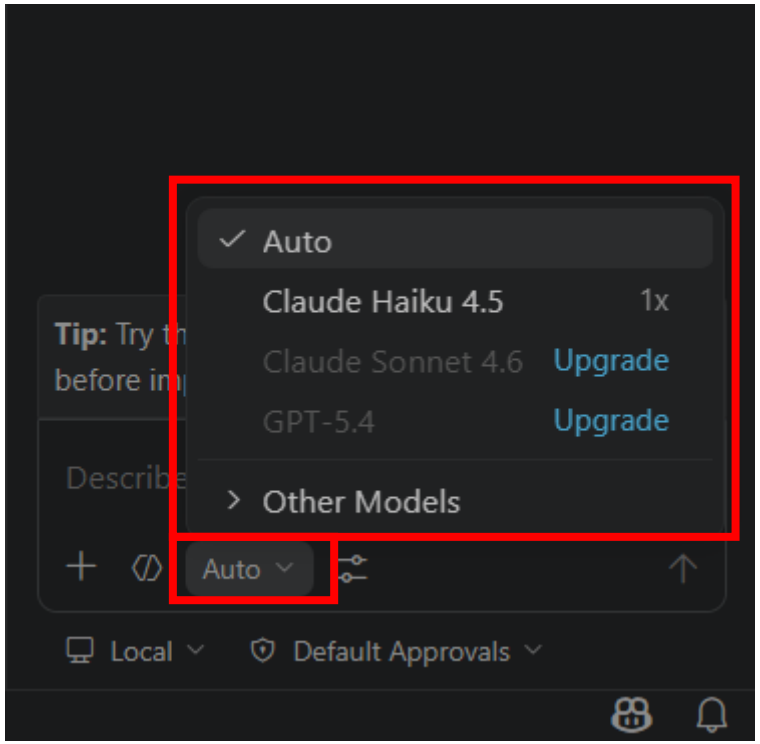


Jefferson Lab

GET SET UP TO VIBE - STEP 3: UNDERSTAND CHAT INTERFACE

1. Changing the Model

- Click the “Auto” dropdown in the chat interface
  - Select the AI model you want to use for coding
- Refer to: https://arena.ai/leaderboard/text to choose models based on your use case

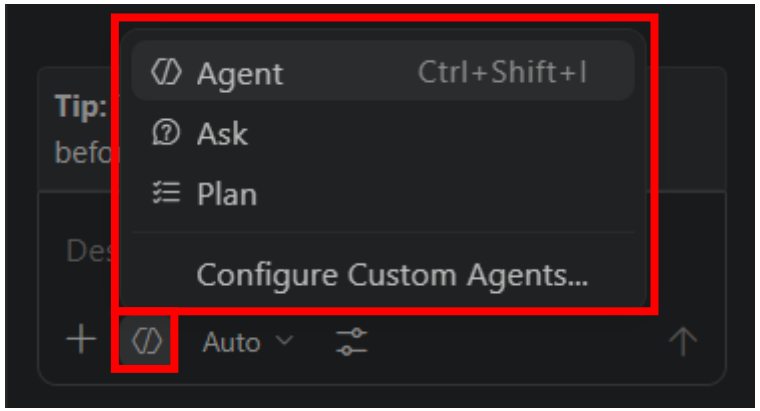


Jefferson Lab

GET SET UP TO VIBE - STEP 3: UNDERSTAND CHAT INTERFACE

2. Changing the Mode

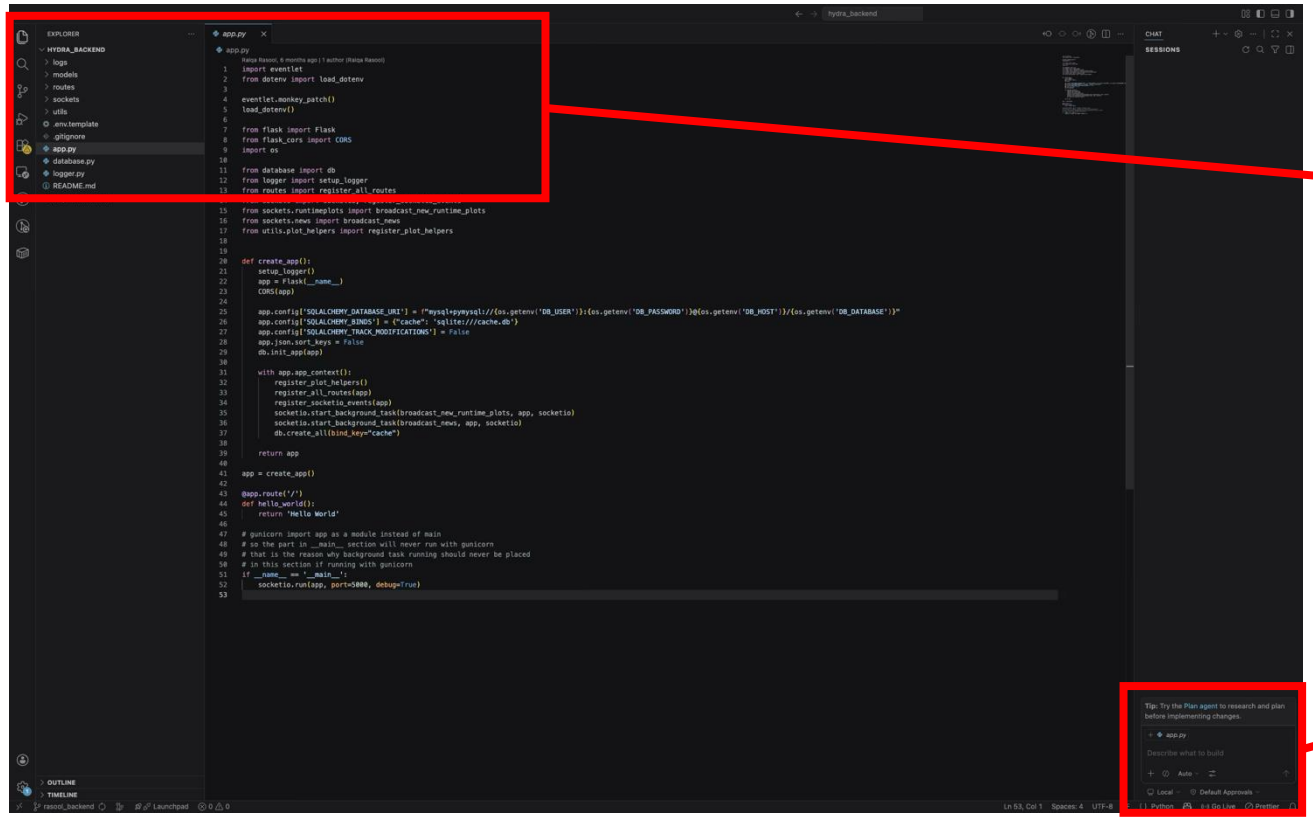
- You can switch between different working modes:
- **Agent Mode**
  - Makes changes directly in your files based on the prompt
- **Ask Mode**
  - Used for questions and discussion
  - Does *not* modify any files
  - Useful for exploring ideas before making changes
- **Plan Mode**
  - Used in early stages of development
  - Helps brainstorm and structure your approach before implementation

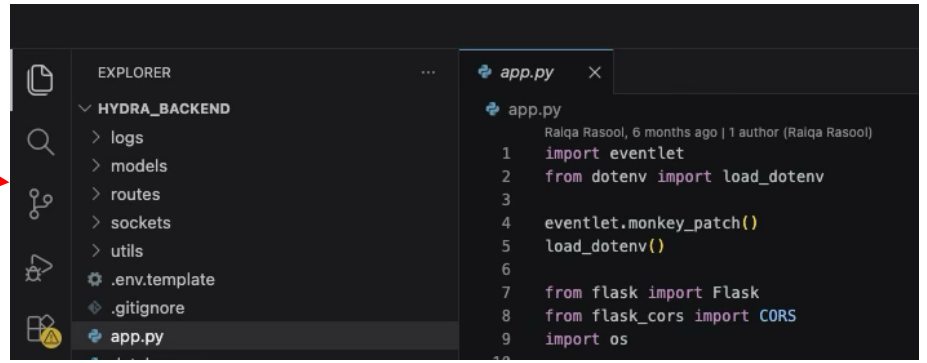


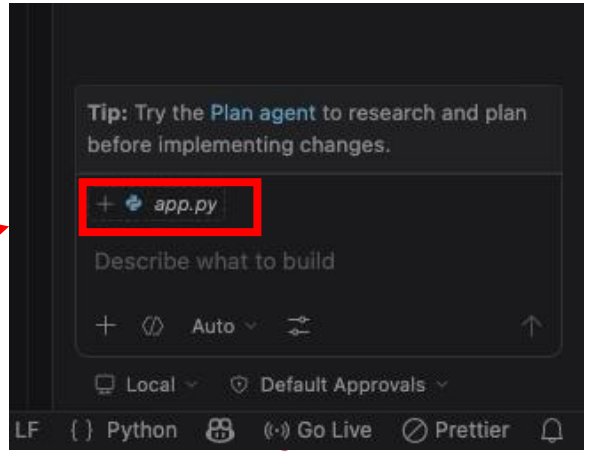
GET SET UP TO VIBE - STEP 3: UNDERSTAND CHAT INTERFACE

3. Adding Context

- Once you open a folder in Visual Studio Code, your workspace becomes available to Copilot as context.
- However, explicitly referencing files or code helps the AI focus on the exact parts relevant to your task.
  1. Opened files are automatically added as active context



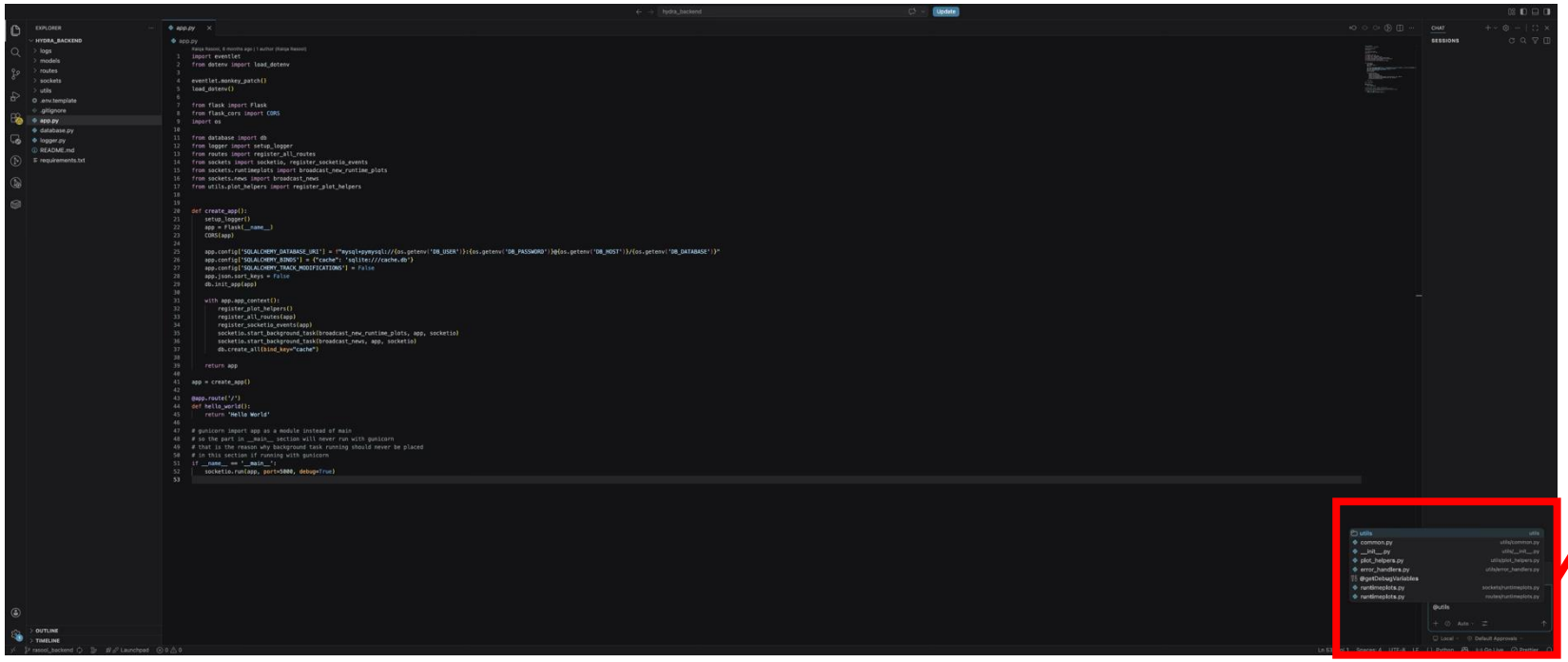


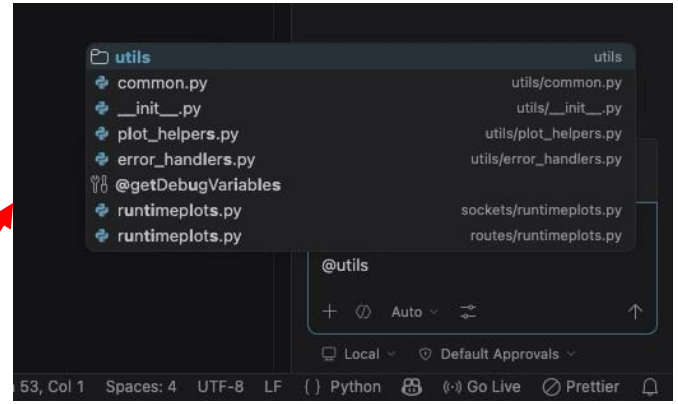


GET SET UP TO VIBE - STEP 3: UNDERSTAND CHAT INTERFACE

3. Adding Context

- Once you open a folder in Visual Studio Code, your workspace becomes available to Copilot as context.
- However, explicitly referencing files or code helps the AI focus on the exact parts relevant to your task.
  - **2.** Reference specific files/folders directly in chat using #/@ followed by the name
    - Example: #app.py, @app.py or @utils, #utils
    - Copilot will understand which file/folder you are referring to.





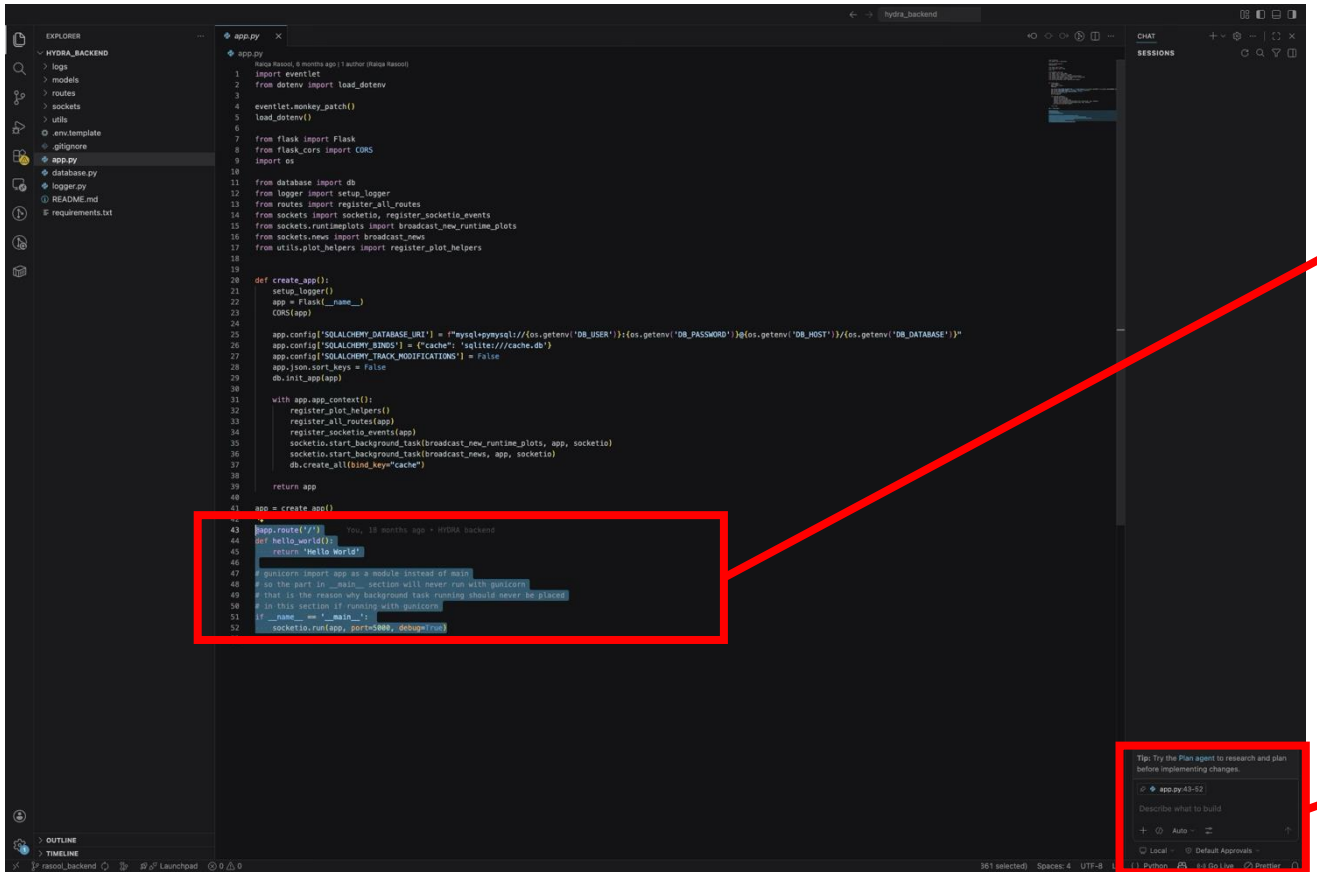
Jefferson Lab

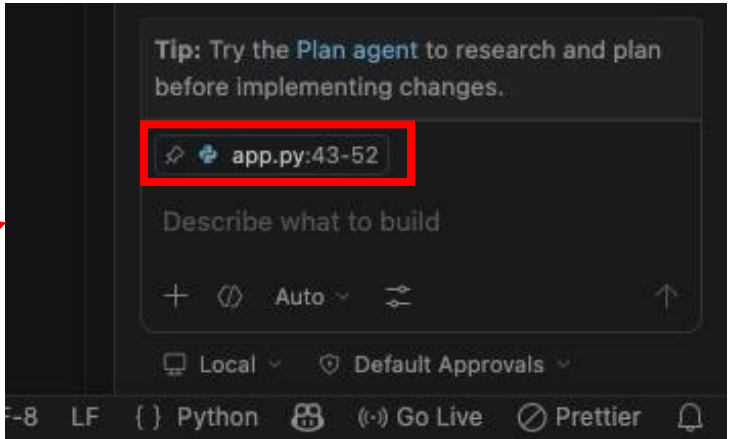
GET SET UP TO VIBE - STEP 3: UNDERSTAND CHAT INTERFACE

3. Adding Context

- Once you open a folder in Visual Studio Code, your workspace becomes available to Copilot as context.
- However, explicitly referencing files or code helps the AI focus on the exact parts relevant to your task.

  3. Highlight specific lines of code to focus the AI on a particular section

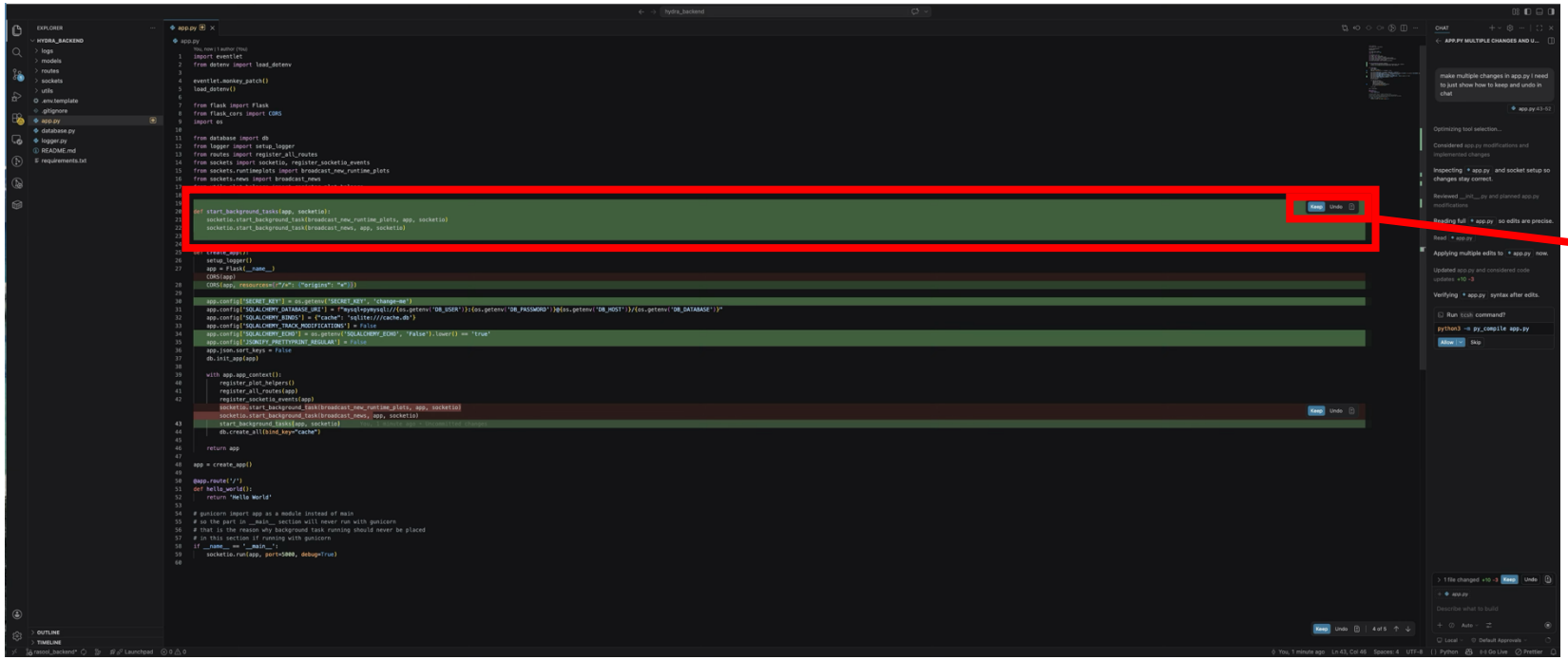


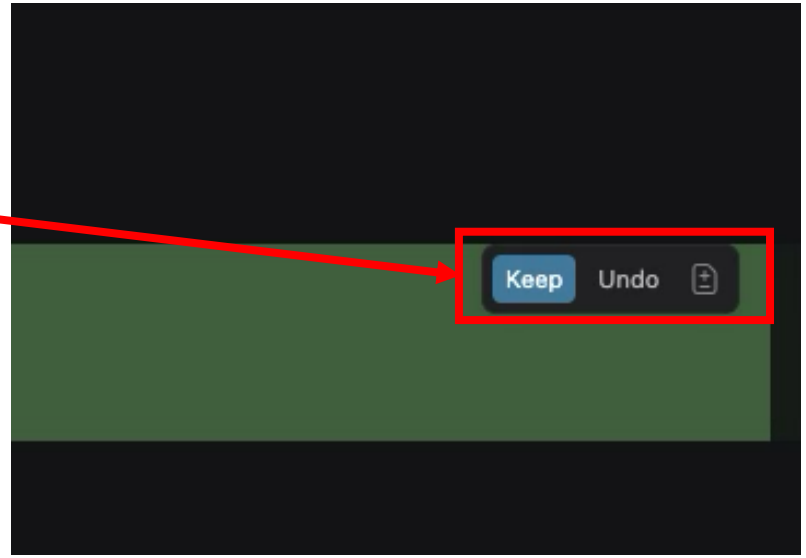


GET SET UP TO VIBE - STEP 3: UNDERSTAND CHAT INTERFACE

3. Undo / Keep Changes

- When you use Copilot Chat (Agent mode) for changes, it edits your files and highlights all modifications:
  - Green lines = added code
  - Red lines = deleted code
- You can review and control changes in multiple ways:
  1. Per line: Hover over changes to keep or undo individual edits





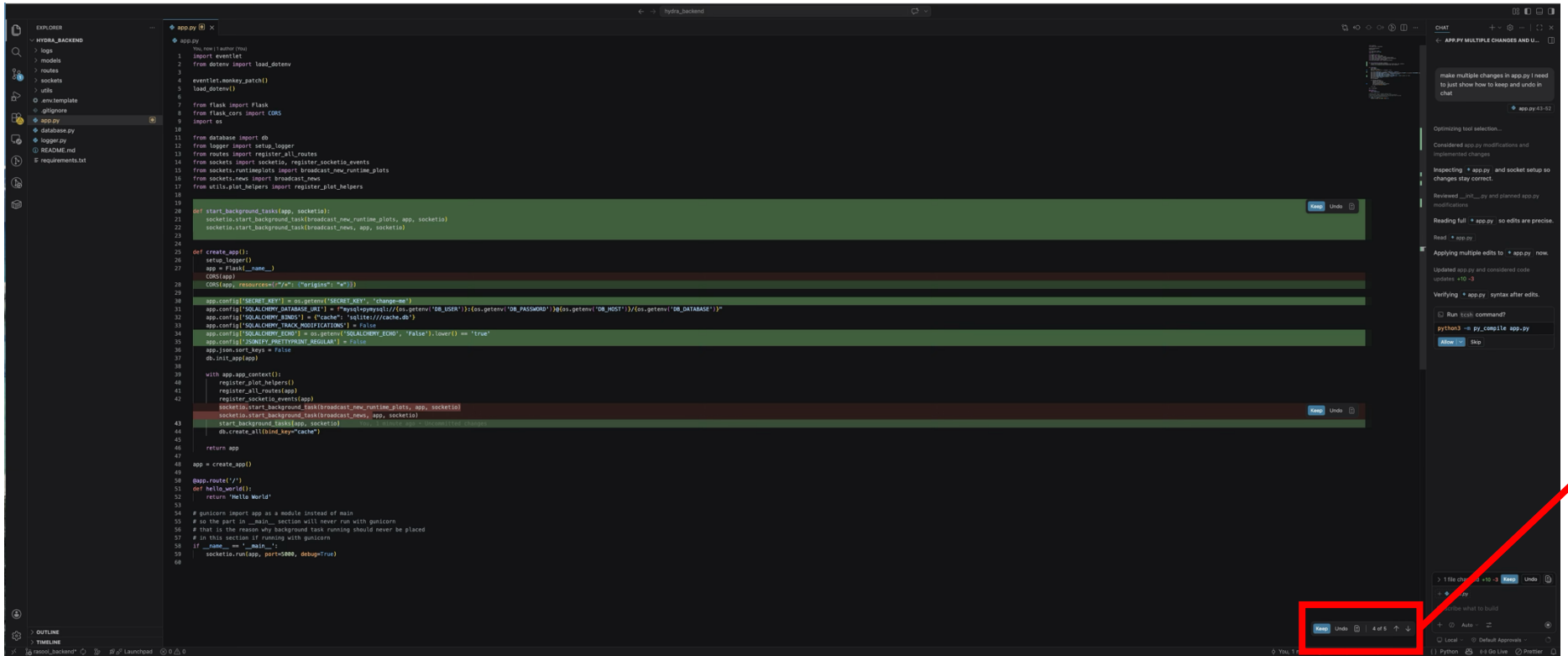
Jefferson Lab

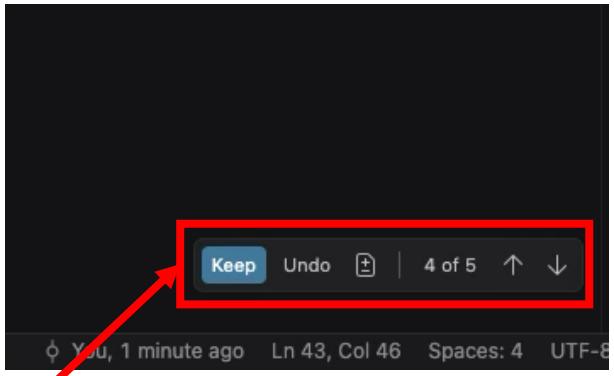
GET SET UP TO VIBE - STEP 3: UNDERSTAND CHAT INTERFACE

3. Undo / Keep Changes

- When you use Copilot Chat (Agent mode) for changes, it edits your files and highlights all modifications:
  - Green lines = added code
  - Red lines = deleted code
- You can review and control changes in multiple ways:

  2. Per file: Use the Undo/Keep in the bottom-right corner to accept or reject all changes within single file





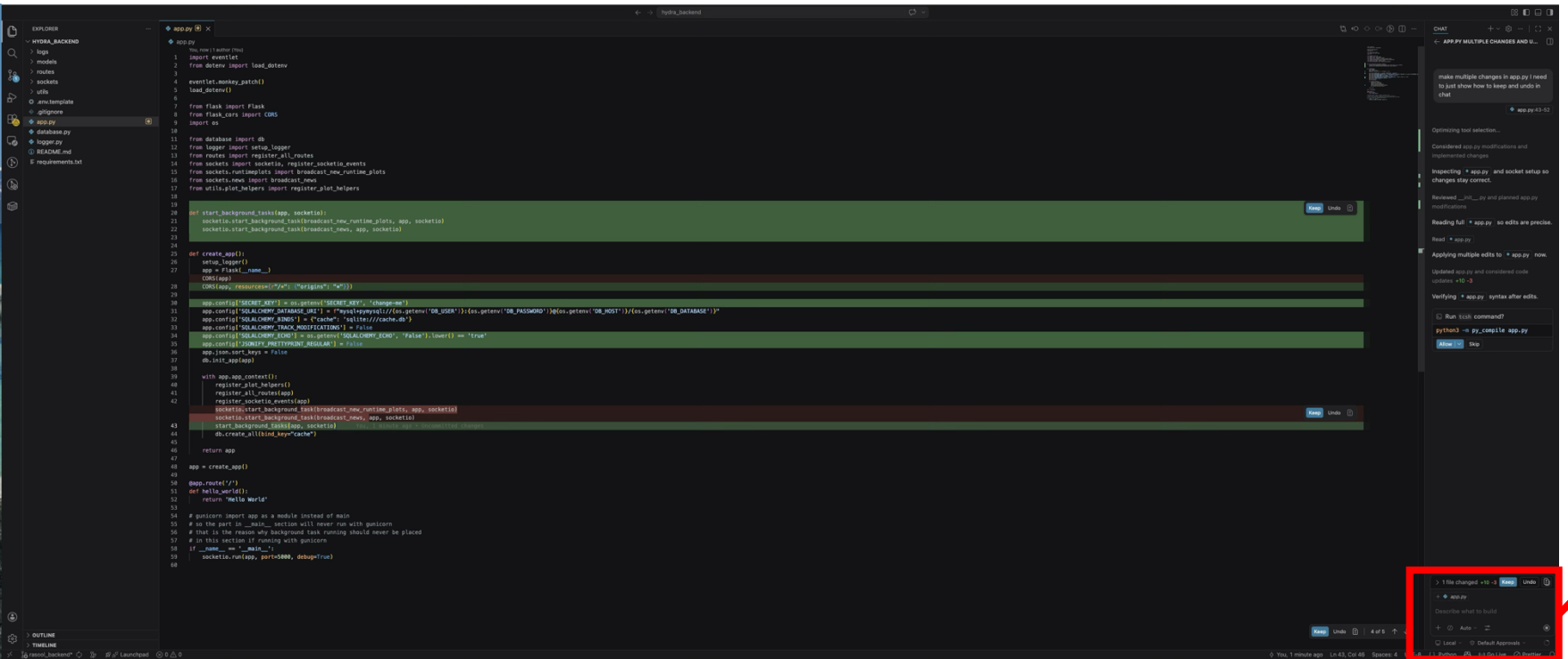
Jefferson Lab

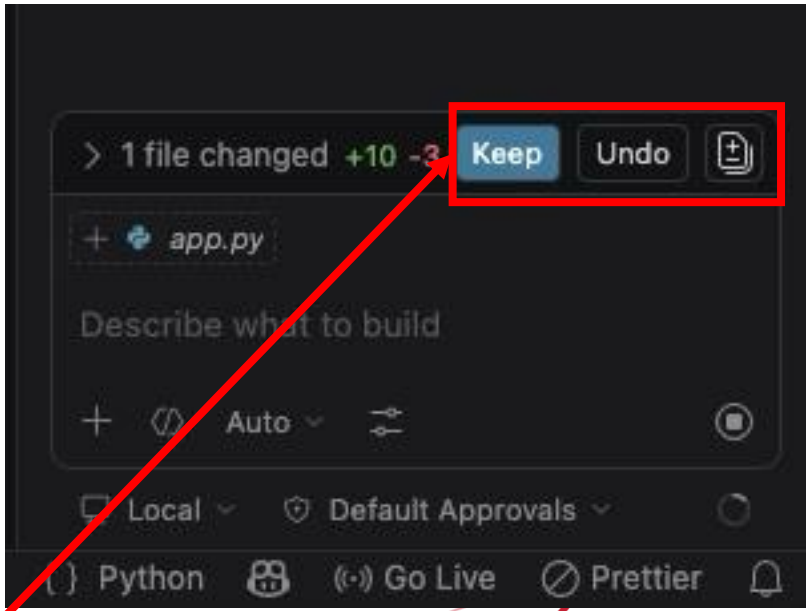
GET SET UP TO VIBE - STEP 3: UNDERSTAND CHAT INTERFACE

3. Undo / Keep Changes

- When you use Copilot Chat (Agent mode) for changes, it edits your files and highlights all modifications:
  - Green lines = added code
  - Red lines = deleted code
- You can review and control changes in multiple ways:

  3. All changes: Use the Keep / Undo option near the chat box to accept or reject all the changes across multiple files

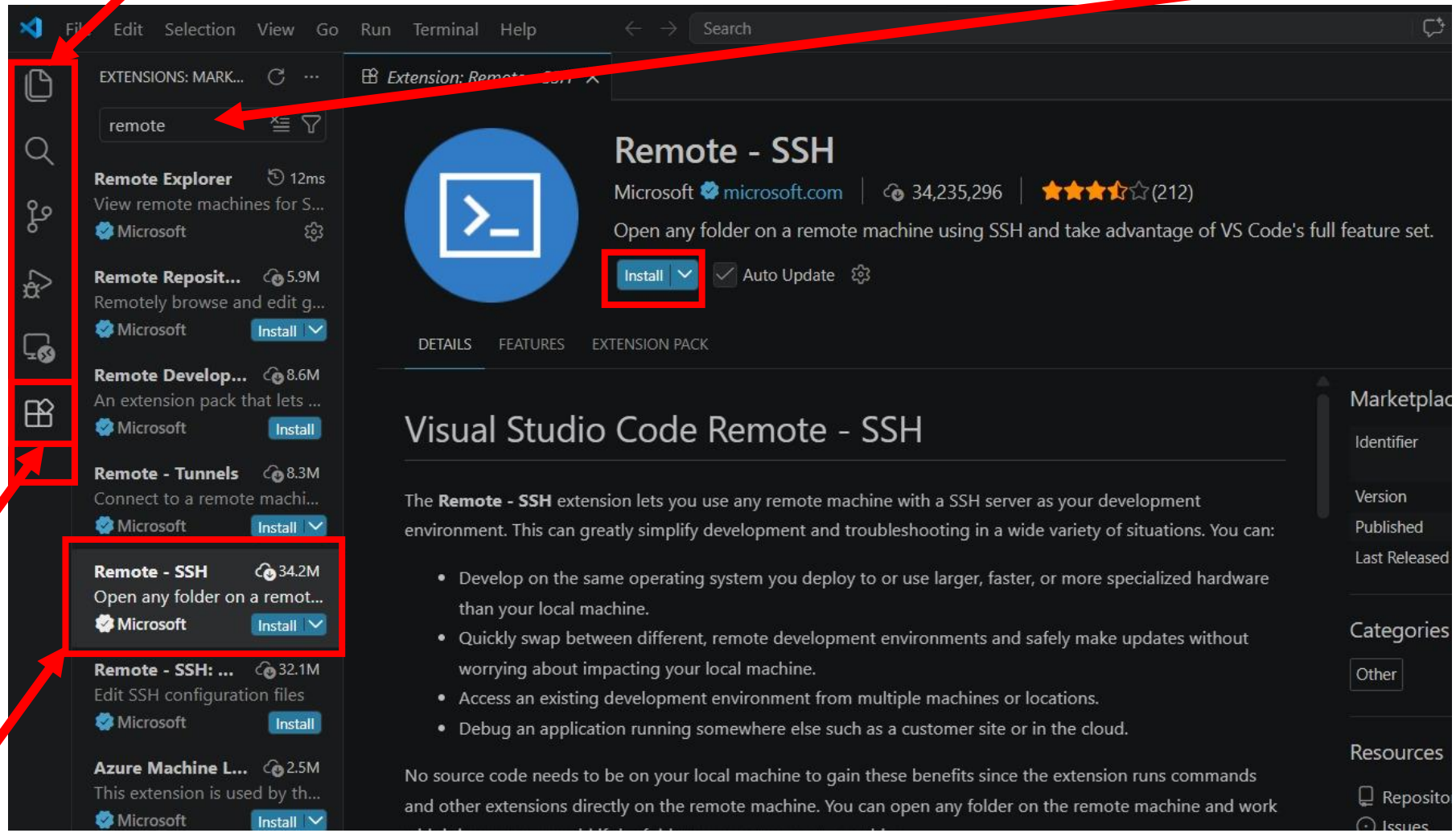




STEP 4: INSTALL REMOTE SSH EXTENSION

GET SET UP TO VIBE - STEP 4: INSTALL REMOTE SSH EXTENSION

- Open VS Code
- Go to Activity Bar and switch to extensions panel and install Remote-SSH extension (search remote)



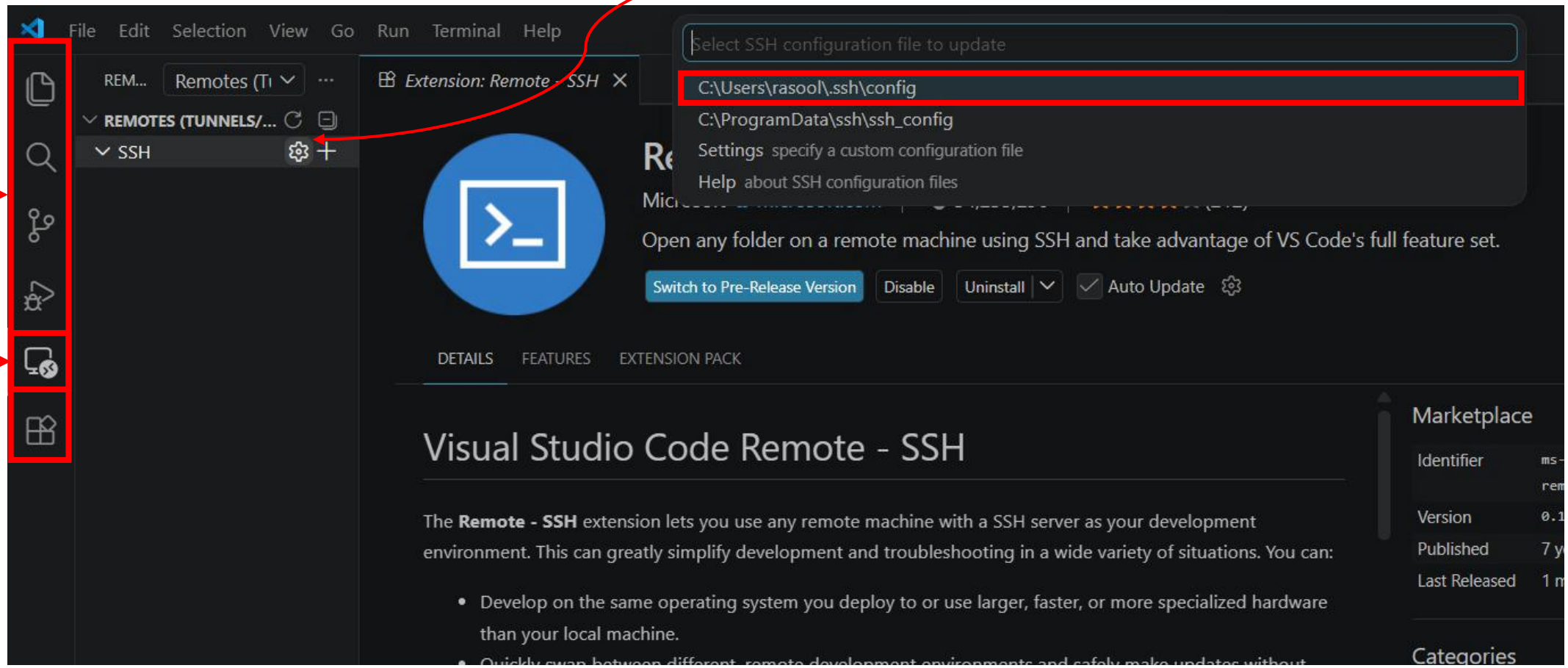
Extensions

Click here

STEP 5: CONFIGURE SSH

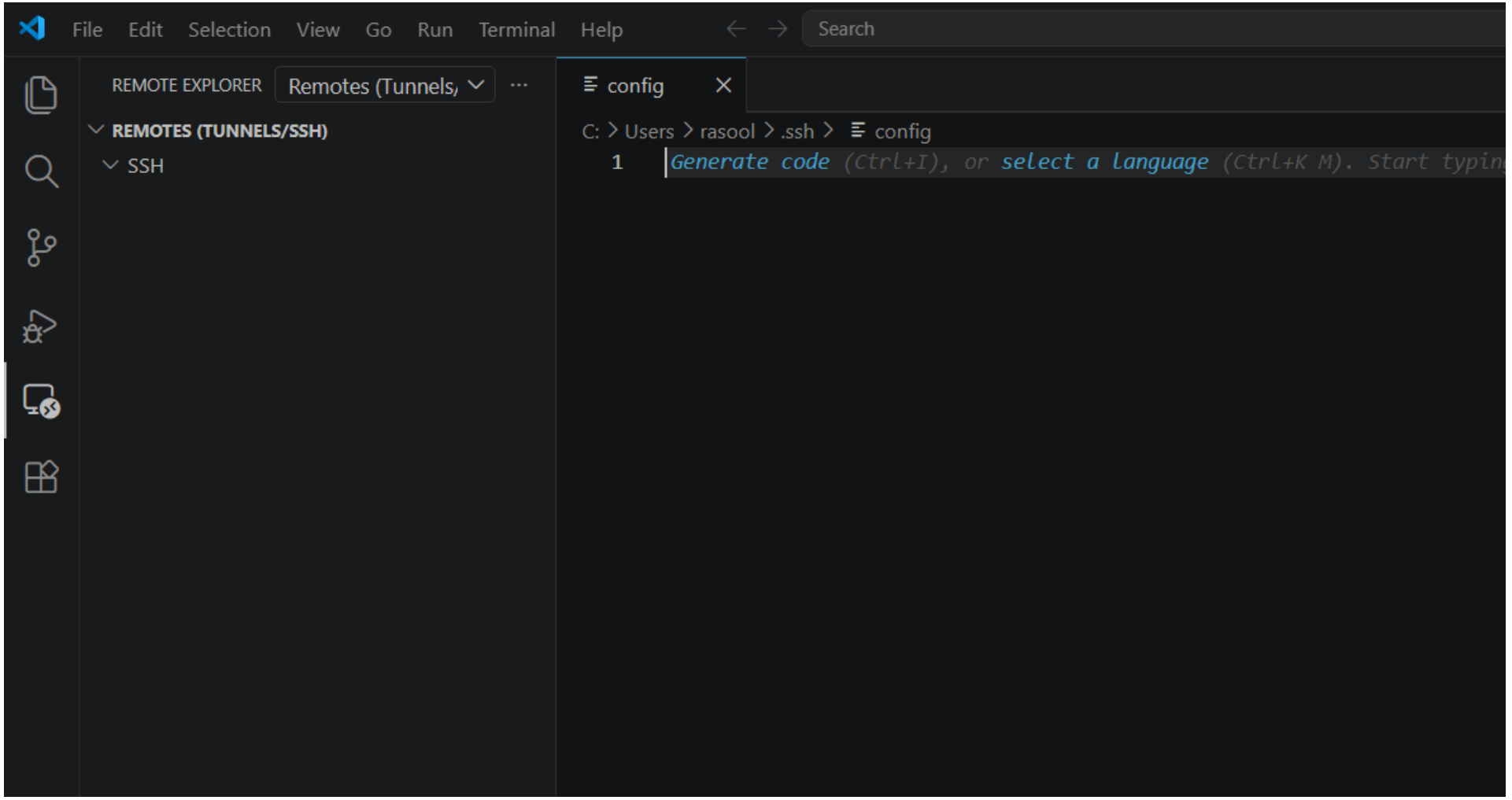
GET SET UP TO VIBE - STEP 5: CONFIGURE SSH

- Go to Remote Explorer panel using Activity Bar and click on settings icon. Now hit enter selecting ~/.ssh/config file



GET SET UP TO VIBE - STEP 5: CONFIGURE SSH

- This will open an empty ~/.ssh/config file to edit:

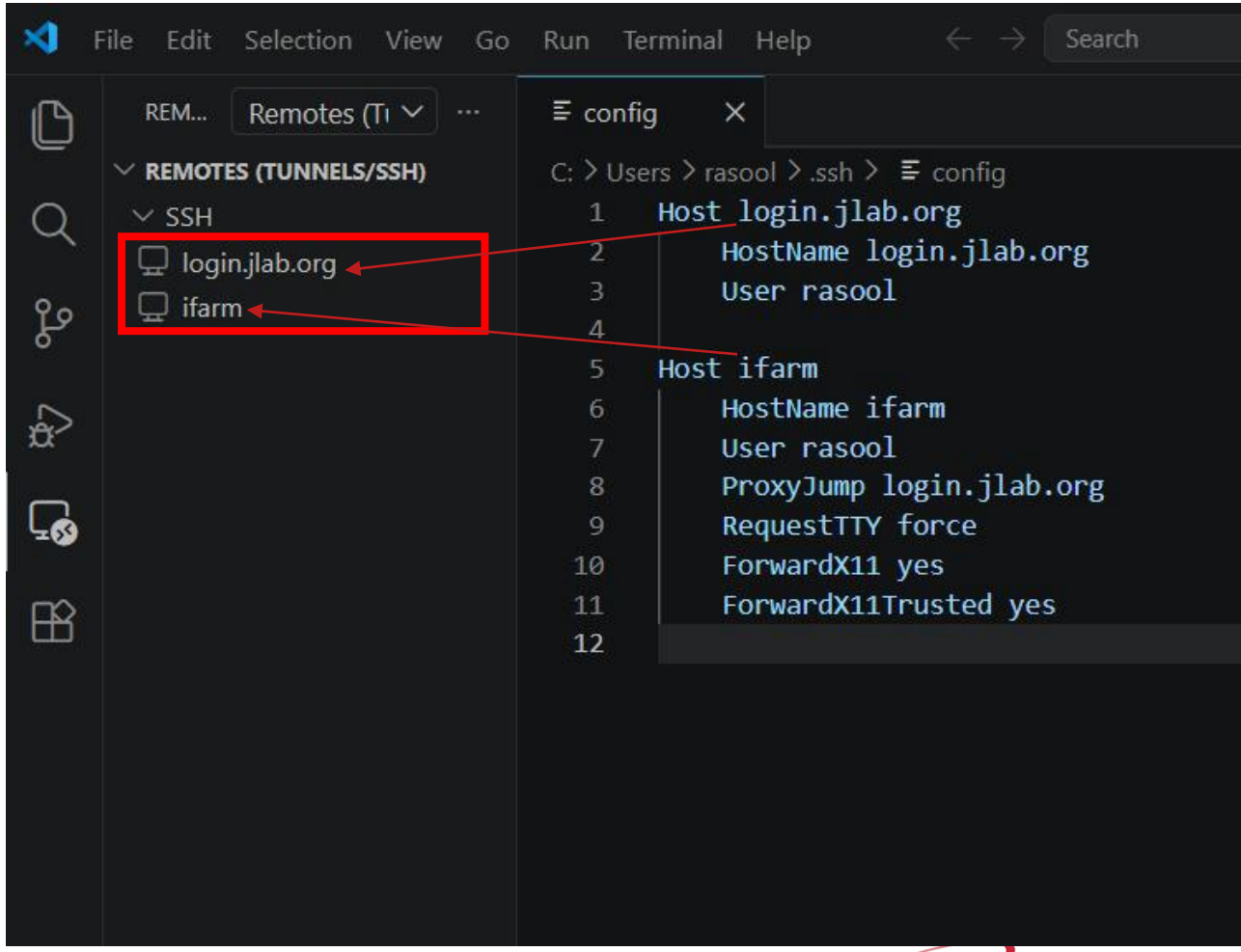


GET SET UP TO VIBE - STEP 5: CONFIGURE SSH

- Paste the exact content into the config file, then press **Ctrl/Cmd + S** to save. Make sure the indentation (tabs and spacing) is correct.

```
Host login.jlab.org
    HostName login.jlab.org
    User <your_jlab_username>

Host ifarm
    HostName ifarm
    User <your_jlab_username>
    ProxyJump login.jlab.org
    RequestTTY force
    ForwardX11 yes
    ForwardX11Trusted yes
```



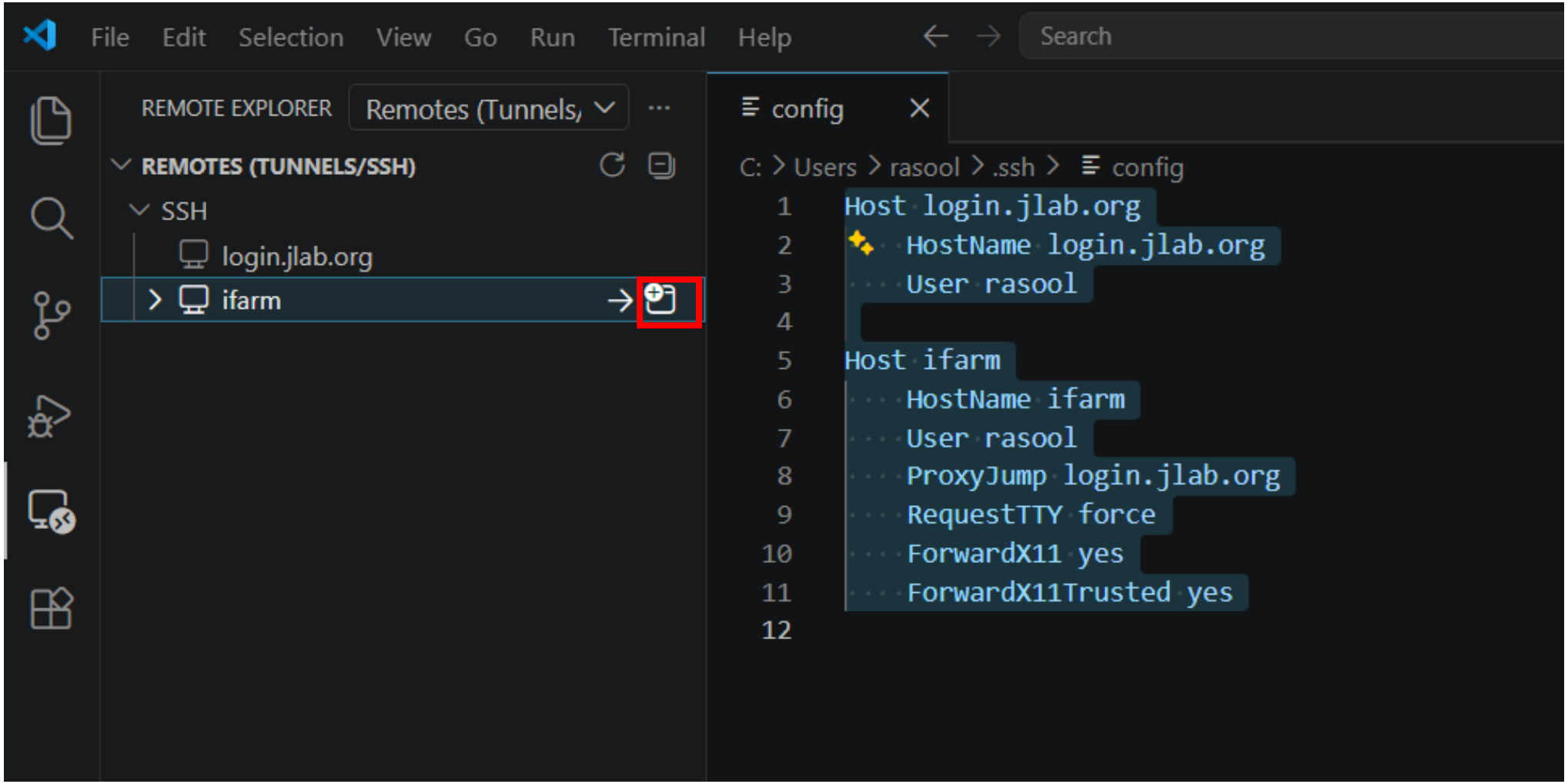
- You will start seeing host names in the Remote Explorer Panel

Note: Replace <your_jlab_username> with your actual JLab username in the file.

STEP 6: CONNECT TO IFARM

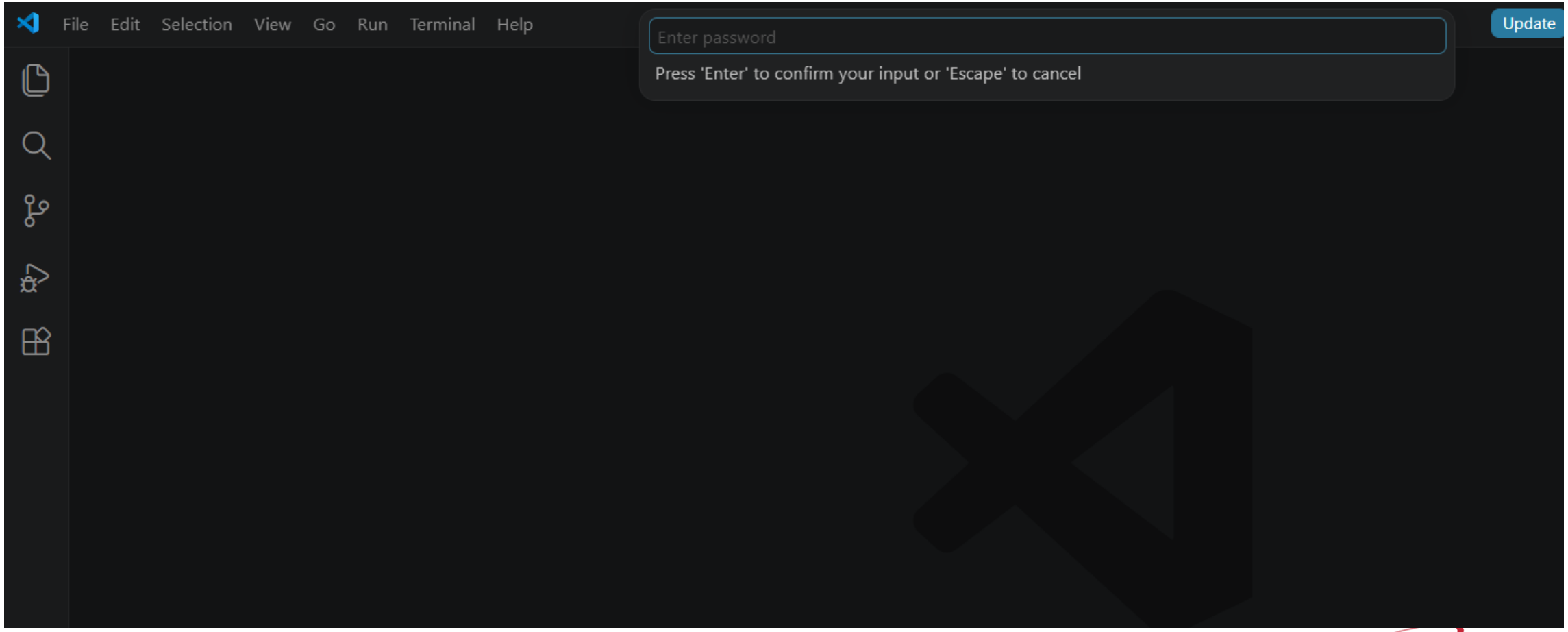
GET SET UP TO VIBE - STEP 6: CONNECT TO IFARM

- Go to Remote Explorer and click on open in new window icon:



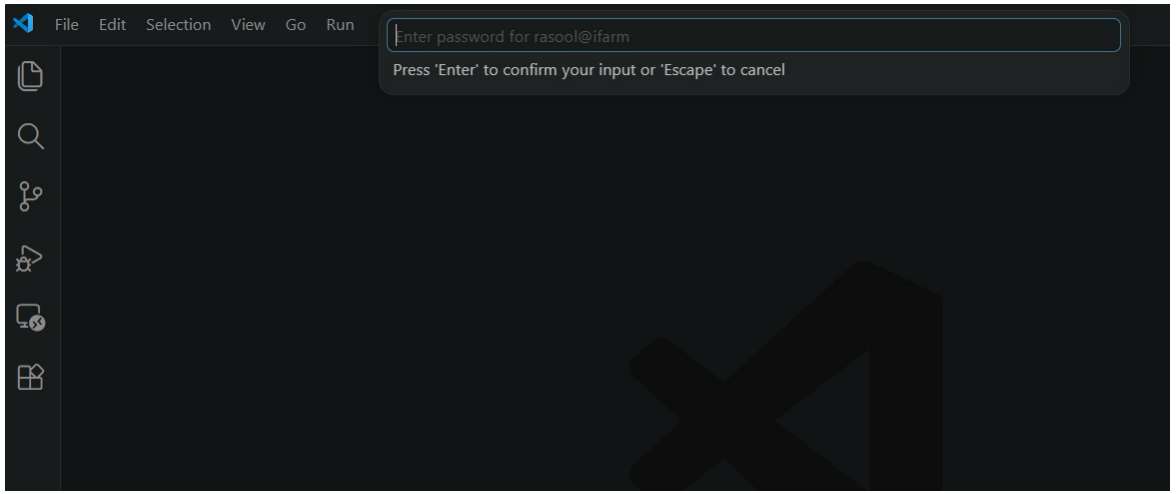
GET SET UP TO VIBE - STEP 6: CONNECT TO IFARM

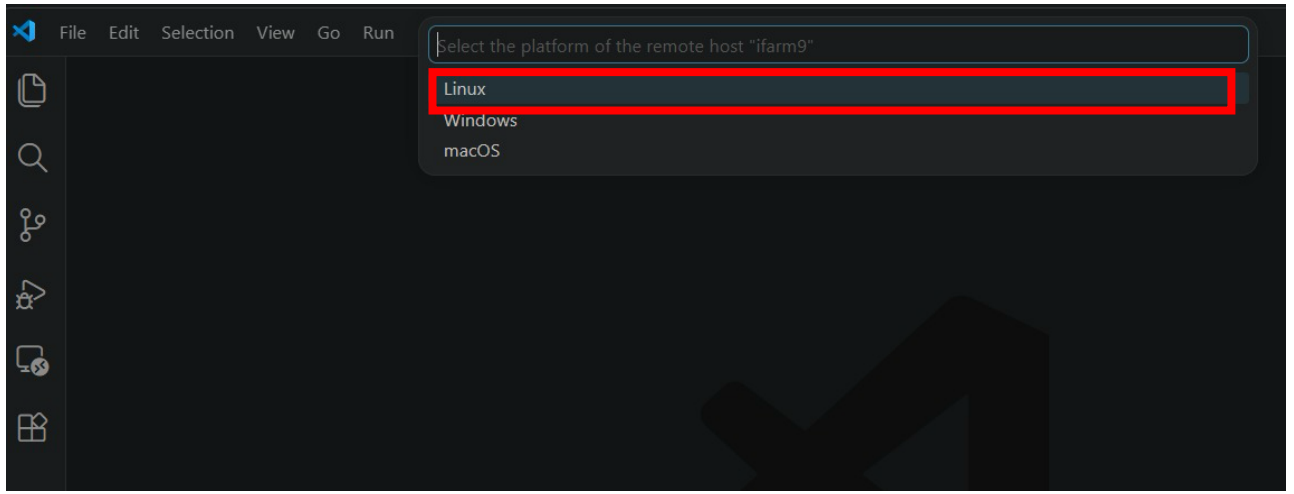
- A new window will open with a password prompt.
  - Enter your **6-digit PIN + MobilePass/USB token**.
  - You are logging into login.jlab.org, which acts as the gateway to access ifarm.
    - The ProxyJump login.jlab.org setting in the config automatically handles this intermediate step.

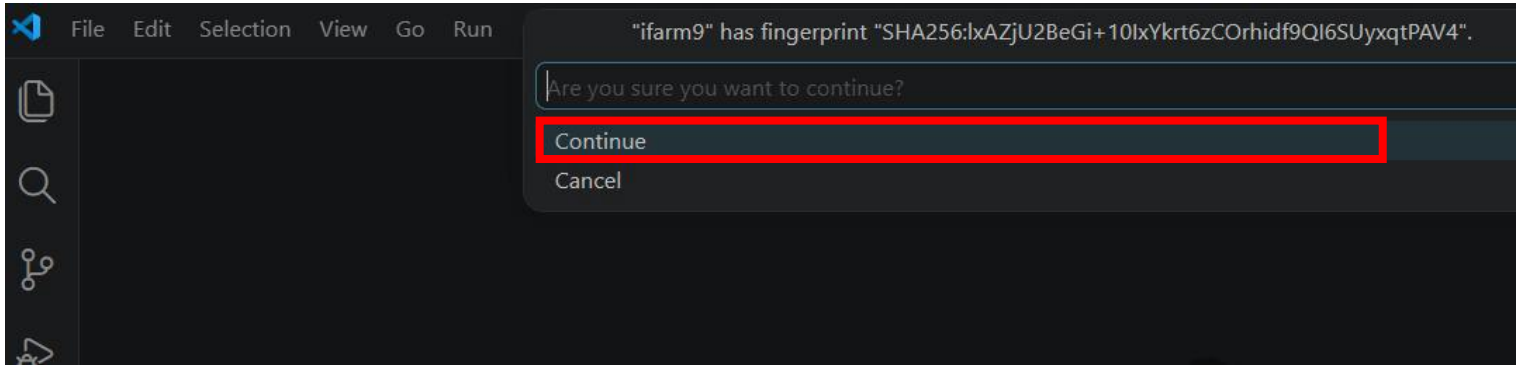


GET SET UP TO VIBE - STEP 6: CONNECT TO IFARM

- After entering your password for login.jlab.org, a second prompt will appear for @ifarm. Enter your CUE (JLab) account password to complete the login.
- For any additional pop-ups, simply select the default trusted option such as “Linux” or “Continue”, depending on what is shown.



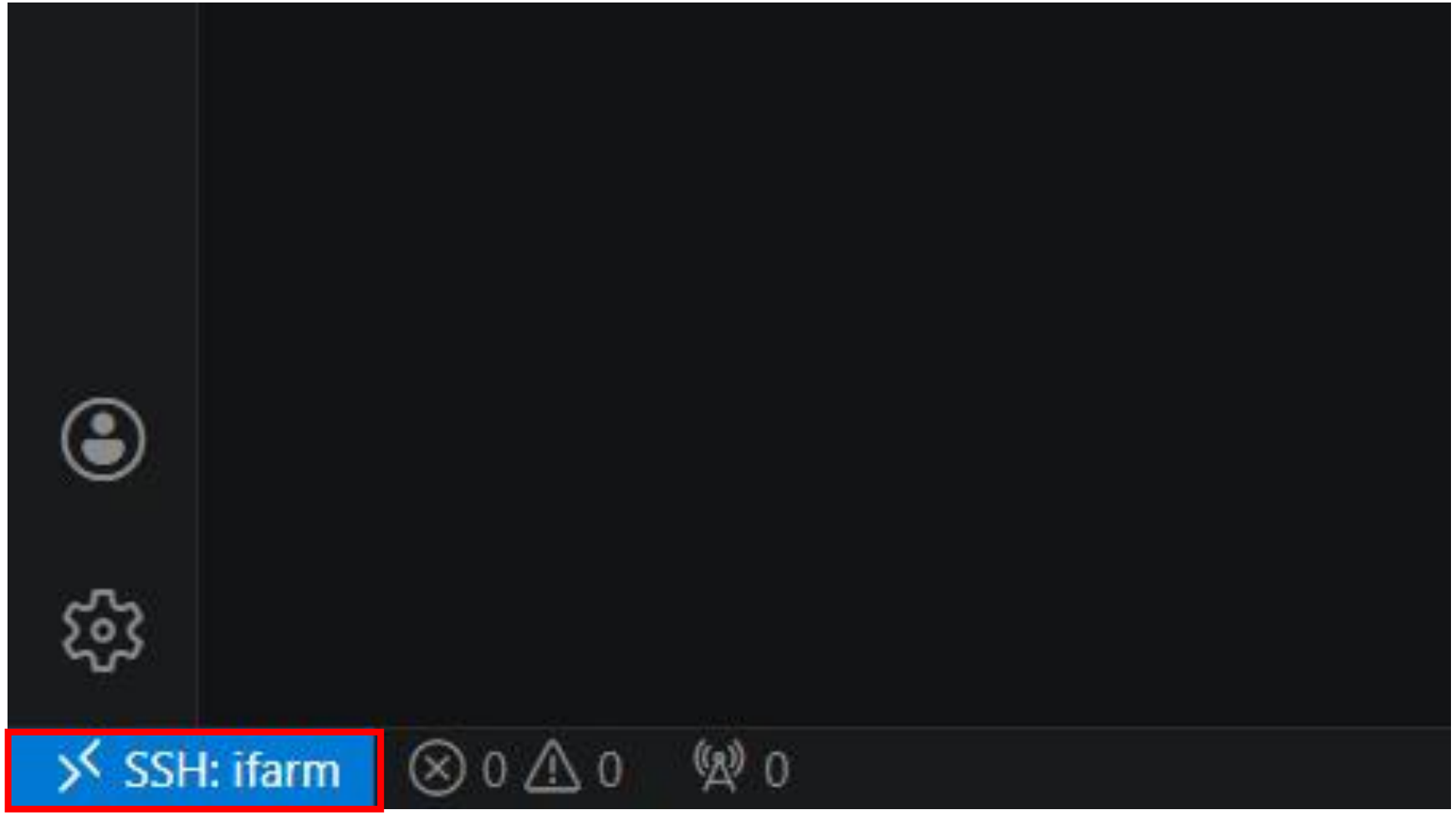




Jefferson Lab

GET SET UP TO VIBE - STEP 6: CONNECT TO IFARM

- After completing all the prompts, you will be successfully connected to **ifarm**. You should now see **“SSH: ifarm”** displayed in the bottom-left corner of VS Code, confirming the connection is active.

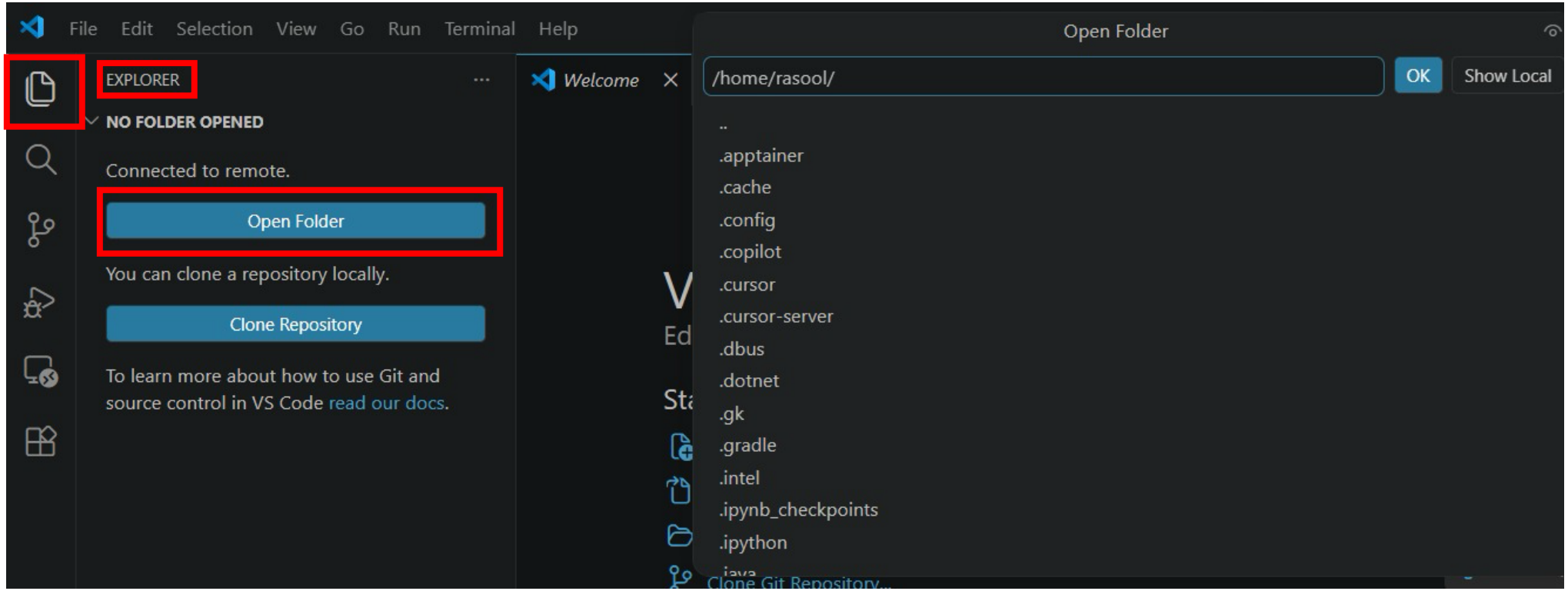


62

STEP 7: OPEN A FOLDER ON IFARM

GET SET UP TO VIBE - STEP 7: OPEN A FOLDER

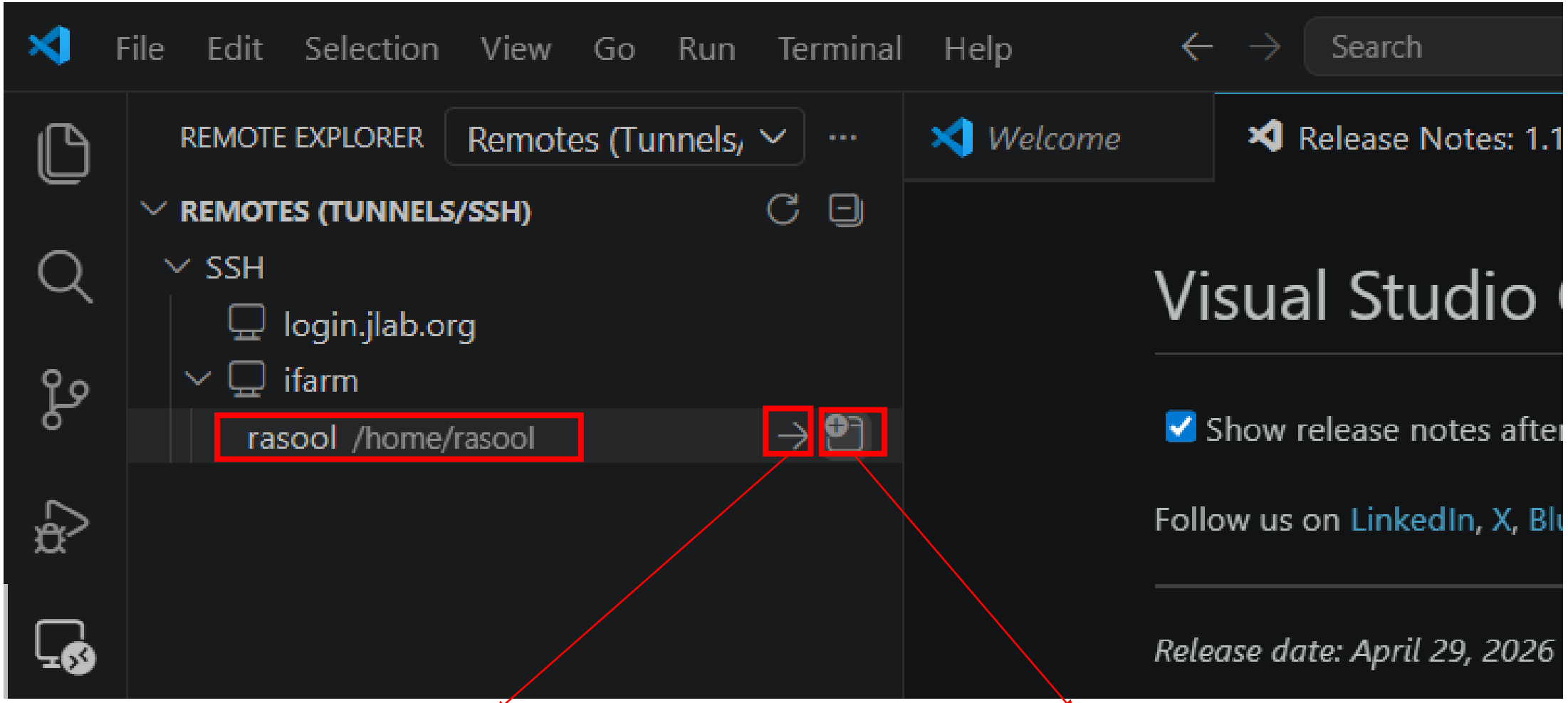
- To open a folder on ifarm, go to the Explorer panel in the Activity Bar and click “Open Folder”.
  - A prompt will appear at the top where you can enter a directory path.
    - Navigate to the folder where you will be working
      - Prefer working under /w (assigned by your group/hall)



Note: Avoid using your home directory. It fills up quickly and can cause issues like: VS Code connection failure due to disk quota exceeded errors (Refer to “Navigating ifarm” slides 51–58 for details)

GET SET UP TO VIBE - STEP 7: OPEN A FOLDER

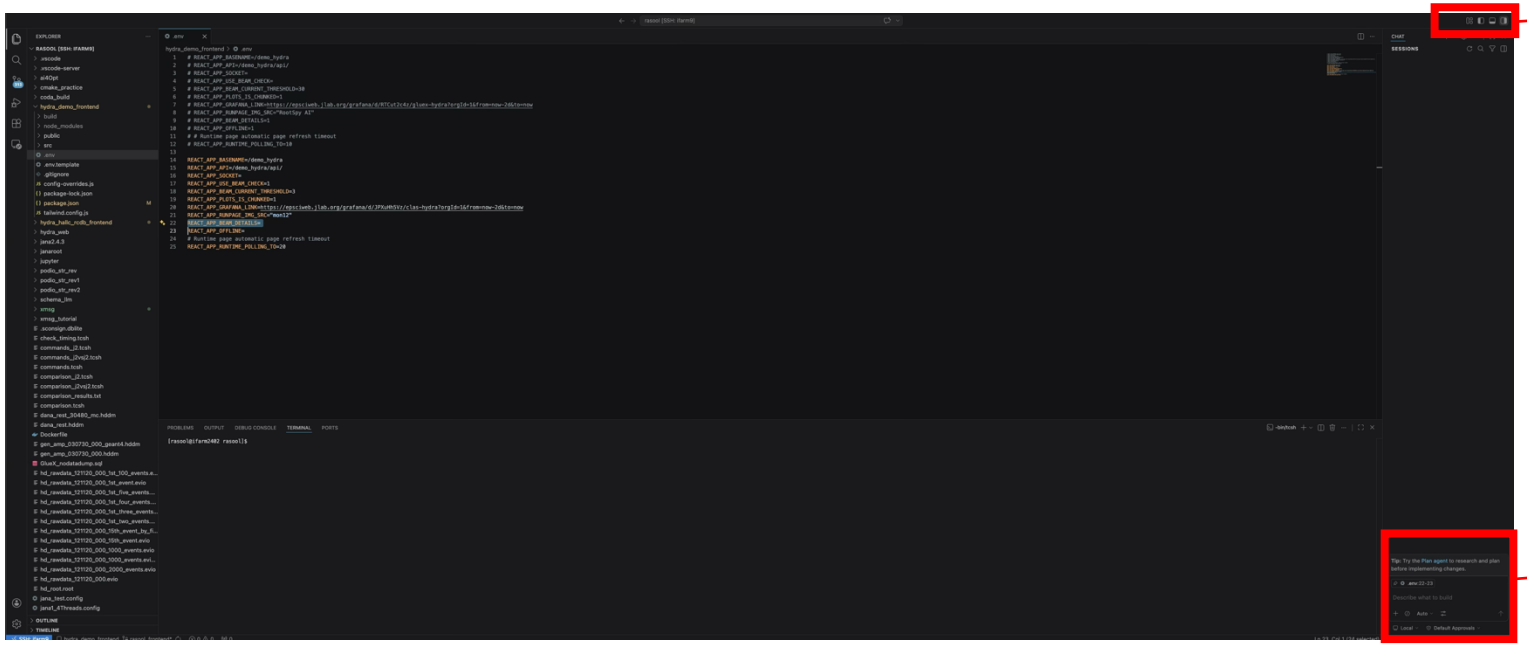
- After connecting to a folder once on a host that path get saved and next time you can directly connect to host on that path by just clicking on that path

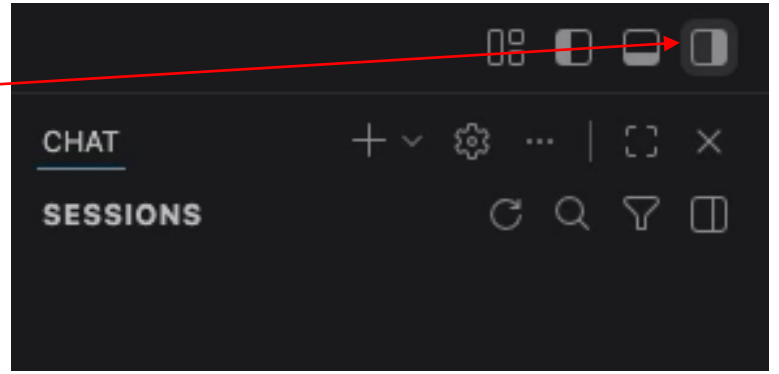


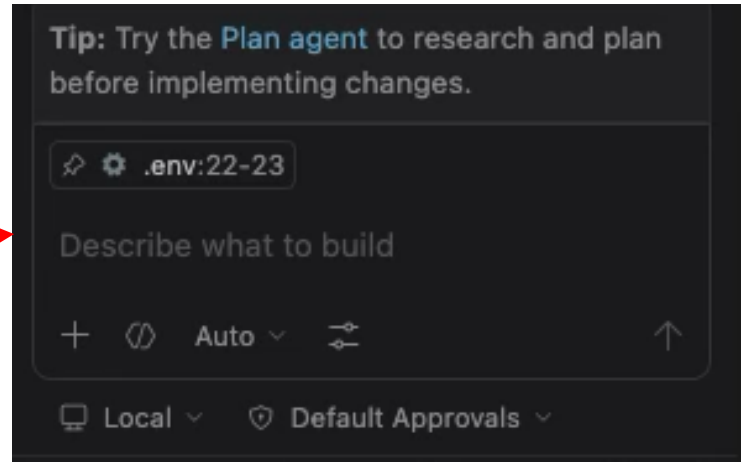
STEP 8: OPEN CHAT IN THE FOLDER ON IFARM

GET SET UP TO VIBE - STEP 8: OPEN CHAT IN THE FOLDER

- Once you are on ifarm and have navigated to your folder, you can open the side chat panel and start using Copilot to make changes, generate code, or get help with anything in the project.
  - As mentioned earlier, the entire folder is now available as AI context, so you can reference files, code lines, and other parts of the project directly in your prompts.







Tip: Using Git while vibe coding is a cheat code—you can commit working versions and easily roll back to them if new changes break anything.

Jefferson Lab

67

STEP 9: X11 FORWARDING

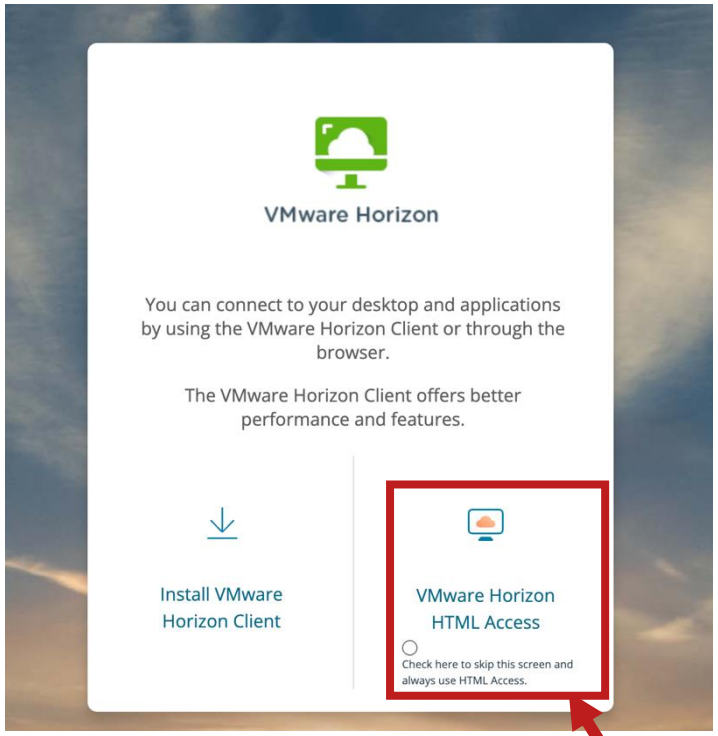
GET SET UP TO VIBE - STEP 9: X11 FORWARDING

- **Running GUI Applications on ifarm**
  - During the hands-on session, we will build and run a GUI application on ifarm while displaying it on your local machine. This requires X11 forwarding.
  - A detailed explanation of X11 forwarding and multiple setup options are covered in the “Accessing & Navigating farm” session.
    - https://indico.jlab.org/event/1069/attachments/13992/22682/AccessingAndNavigatingIFarm.pdf
  - Here, we will use the method with the simplest setup that works across all operating systems.
    - **Steps**
      1. Log in to the VDI
      2. Open a Terminal
      3. SSH into ifarm using the -Y flag for X11 forwarding
      4. Navigate to your project directory
      5. Run your application

GET SET UP TO VIBE - STEP 9: X11 FORWARDING

1. Log in to the VDI

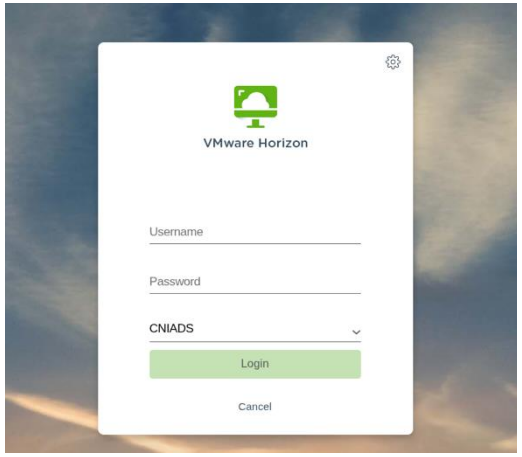
1.



https://vdi.jlab.org

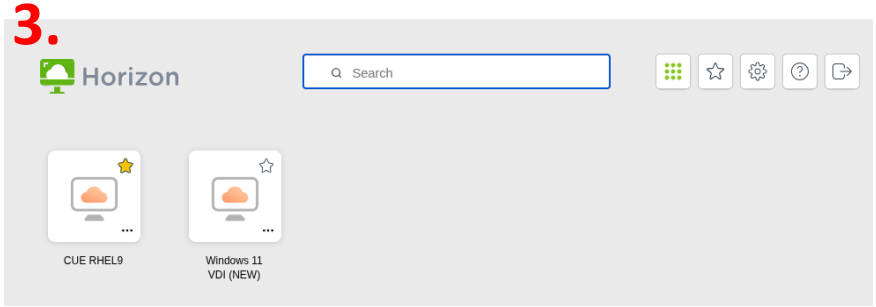
Web access

2.



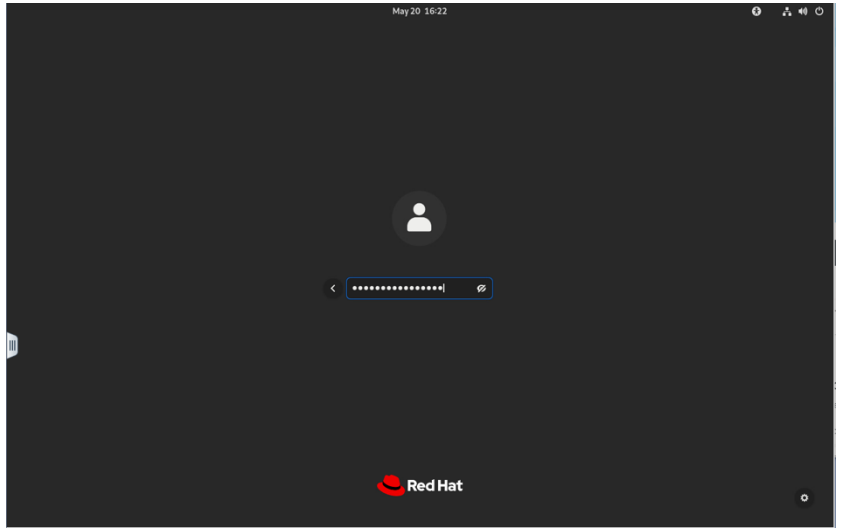
Log in with CUE Username and Password

3.



Select RHEL9 machine (Windows requires SmartCard!!)

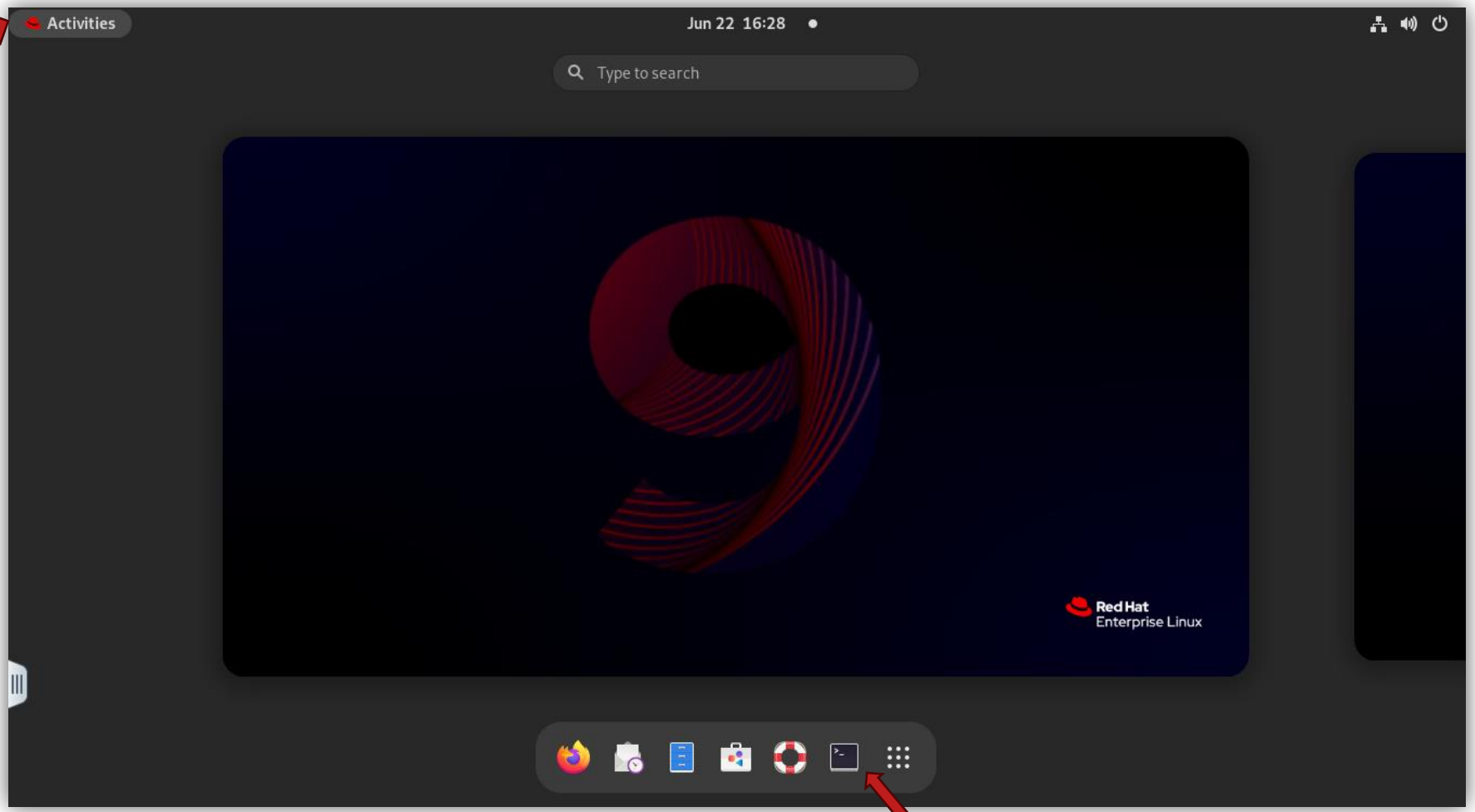
4. Enter CUE Username
(Password = **MobilePASS** SAS MFA PIN+Code)



GET SET UP TO VIBE - STEP 9: X11 FORWARDING

2. Open the terminal

1. Click activities button



2. Click Terminal Icon

GET SET UP TO VIBE - STEP 9: X11 FORWARDING

3. SSH into ifarm :
    - SSH command:
        - ssh -Y -J <username>@login.jlab.org <username>@ifarm.jlab.org
            - -Y = X11 forwarding with encryption
            - -J = ProxyJump

```
[rasool@cue-vdi9122 ~]$ ssh -Y -J rasool@login.jlab.org rasool@ifarm.jlab.org
(rasool@login.jlab.org) Password:
rasool@ifarm.jlab.org's password:
```

Enter PIN + USB/MobilePass Token

Enter JLab Account Password

4. Navigate to your project directory

```
[rasool@ifarm2402 ~]$ cd /w/epsci-scshelf2103/rasool/handsOnWorkshop/
[rasool@ifarm2402 handsOnWorkshop]$
```

5. Run your app

```
rasool@ifarm2402:handsOnWorkshop
[rasool@ifarm2402 handsOnWorkshop]$ python main.py
Loaded 3312812 events
```

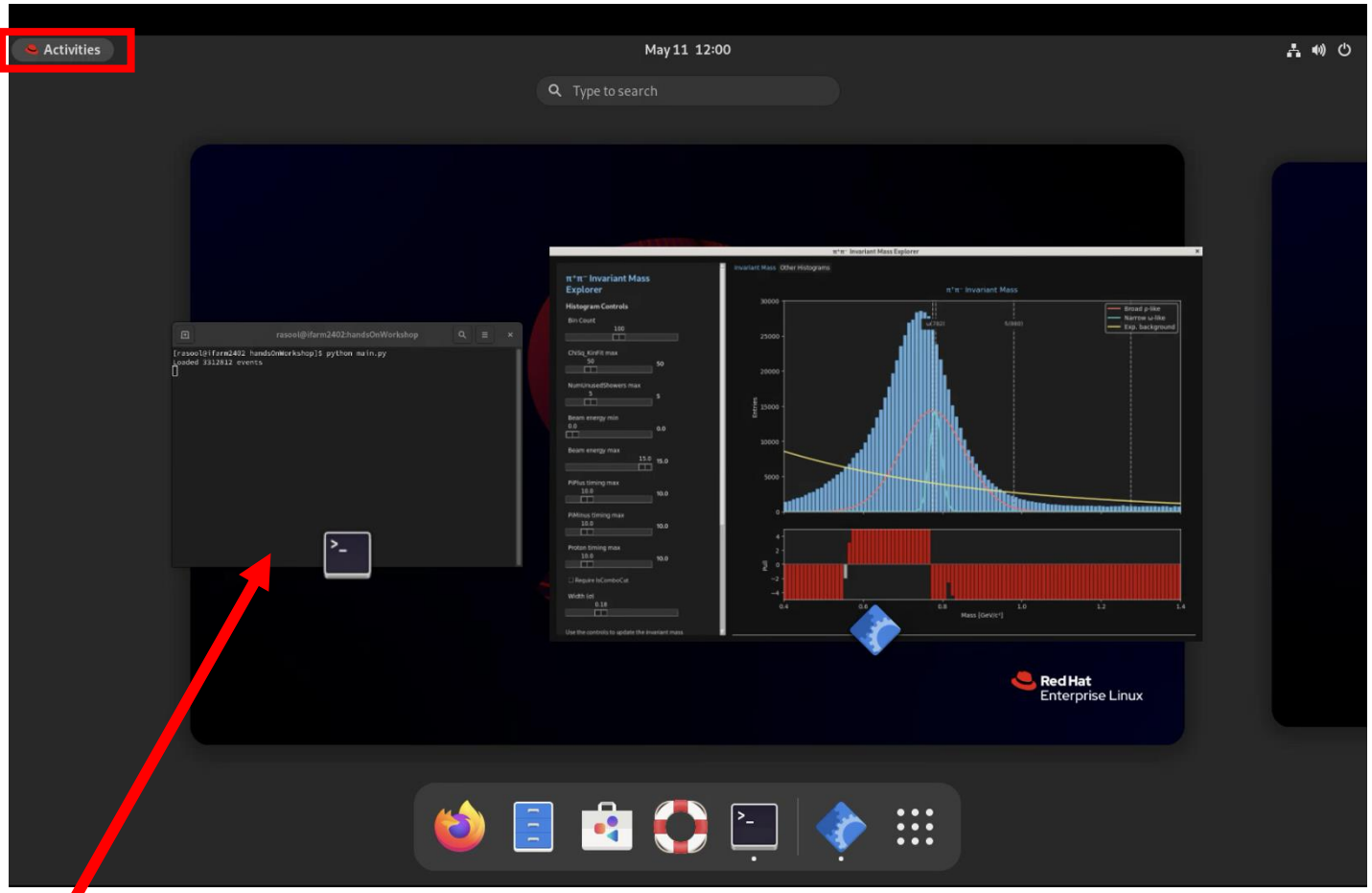
GET SET UP TO VIBE - STEP 9: X11 FORWARDING

- **Exiting the GUI Application**
  - The GUI application may open in full-screen mode. To close it, you can use either of the following methods:
    - Click the close (✕) button on the application window
    - Or:
      - Open **Activities** from the top-left corner
      - Return to the terminal window
      - Press Ctrl + C
      - Wait a few seconds for the application to terminate properly

GET SET UP TO VIBE - STEP 9: X11 FORWARDING

- **Exiting the GUI Application**
  - The GUI application may open in full-screen mode. To close it, you can use either of the following methods:
    - Click the close (X) button on the application window
    - Or:
      - Open **Activities** from the top-left corner
      - Return to the terminal window
      - Press Ctrl + C
      - Wait a few seconds for the application to terminate properly

1. Click the Activities on top



2. Select the terminal to select it
3. Enter Ctrl + C in the terminal

Continuing Your AI Learning Journey

AI Community & Meetups

- **@ JLab**
  - Weekly informal LLM/AI agent discussions
  - *Wednesdays | 12:00–1:00 PM (lunch)*
  - Scientists share projects, ideas, and new learnings
  - Contact Thomas Britton for invite:  tbritton@jlab.org
  - May occasionally be skipped depending on schedule
  - Not necessarily beginner-focused
- **@ Peninsula**
  - *Peninsula Builders / Developers Study Group*
    - *Every other Tuesday | 6:00–7:30 PM*
    - *Location: CNU (Luter 170)*
    - Format: 1 hr AI Study + 30 min sharing interesting findings
  - Hosted by: Raiqa Rasool (rasool@jlab.org)
  - Details & RSVP: Meetup Link
- **@ Hampton**
  - *The AI Collective Hampton Roads*
    - Free online + in-person AI events (Norfolk & Virginia Beach)
    - Covers AI tools, workflows, and practical usage
    - *Peninsula Builders Study Group is also part of this network*
    - More info: AI Collective Hampton

Thanks

Any Questions?