VSCode and Github Copilot for NCAR Systems

CISL/CSG

Brett Neuman

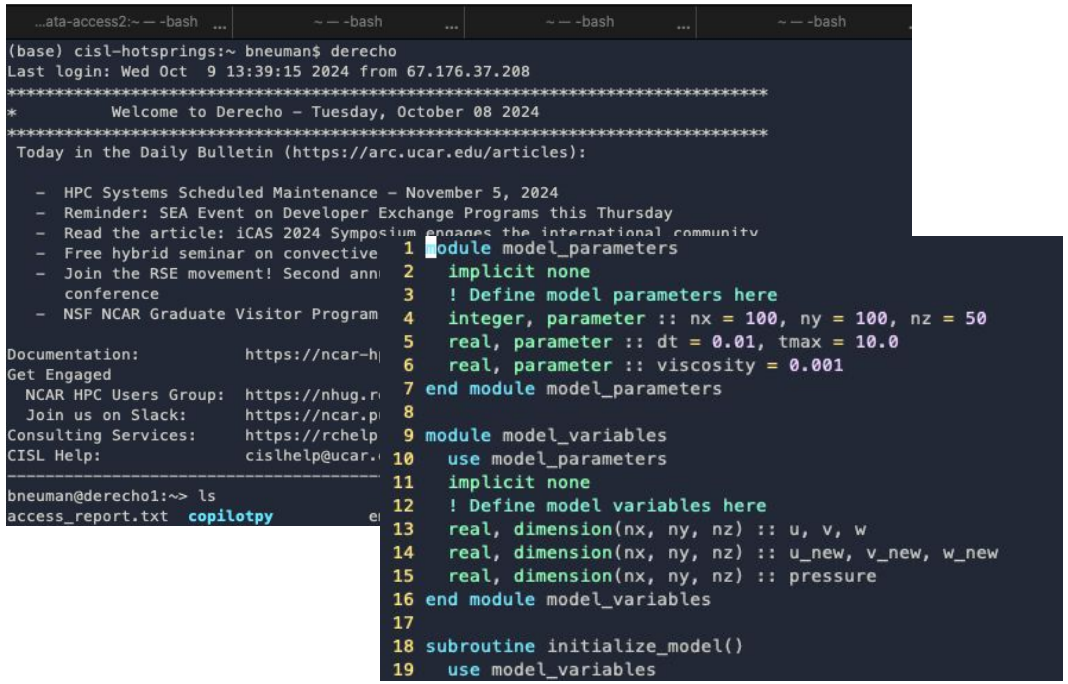
October 24th, 2024

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Common NCAR Remote Connections

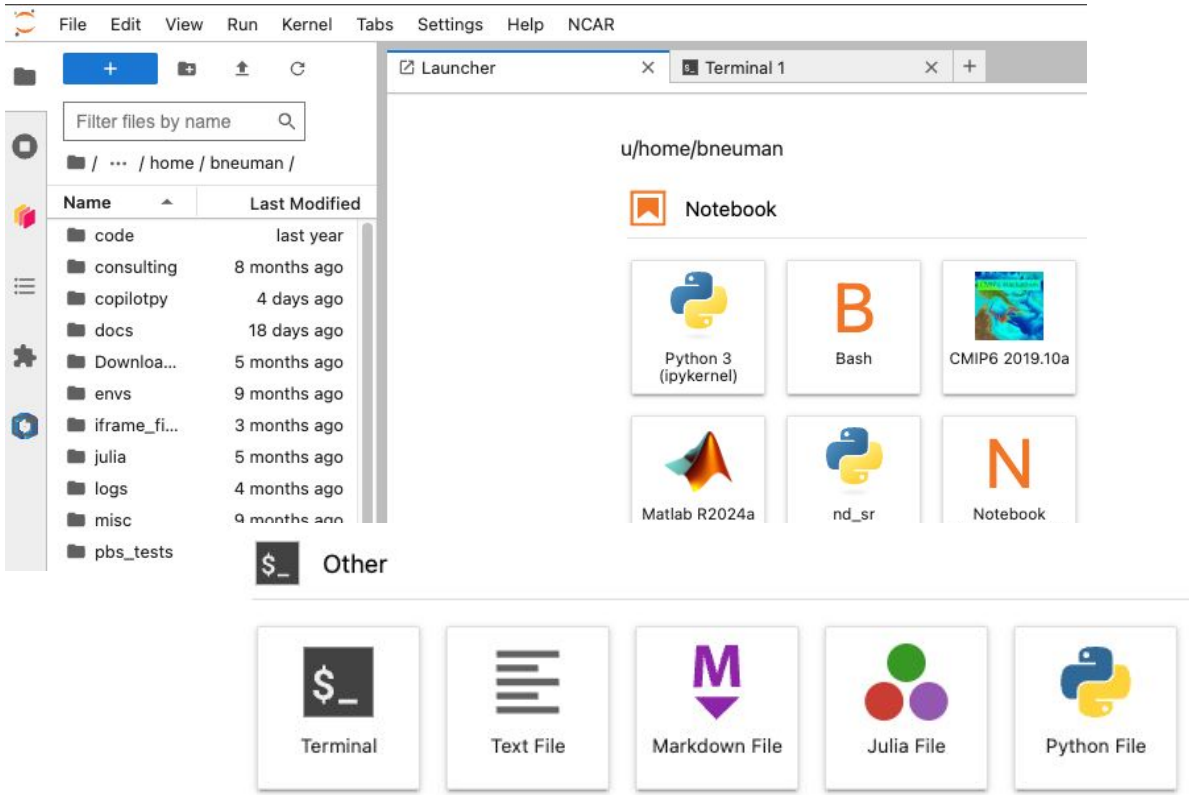
Terminal

- *vi / emacs*
- *tmux*
- *No GUI*



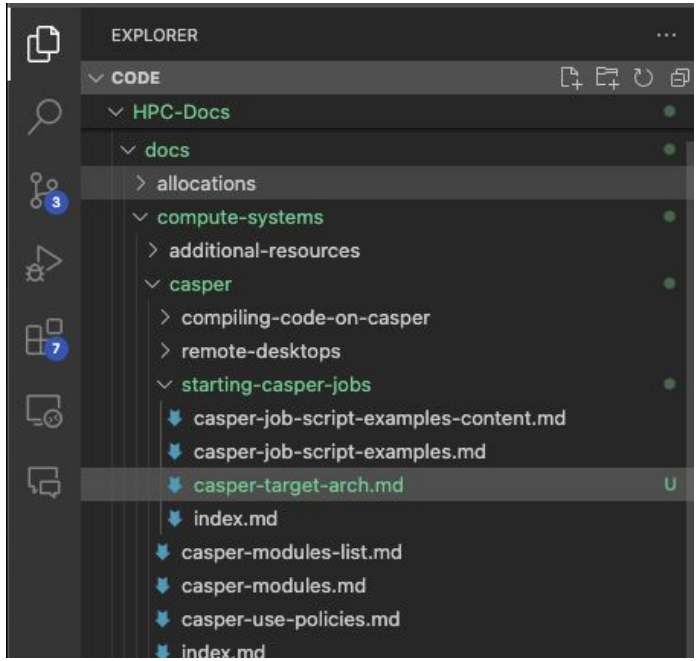
Jupyterhub

- *Jupyter notebooks*
- *Limited terminal integration*
- *GUI with file management*



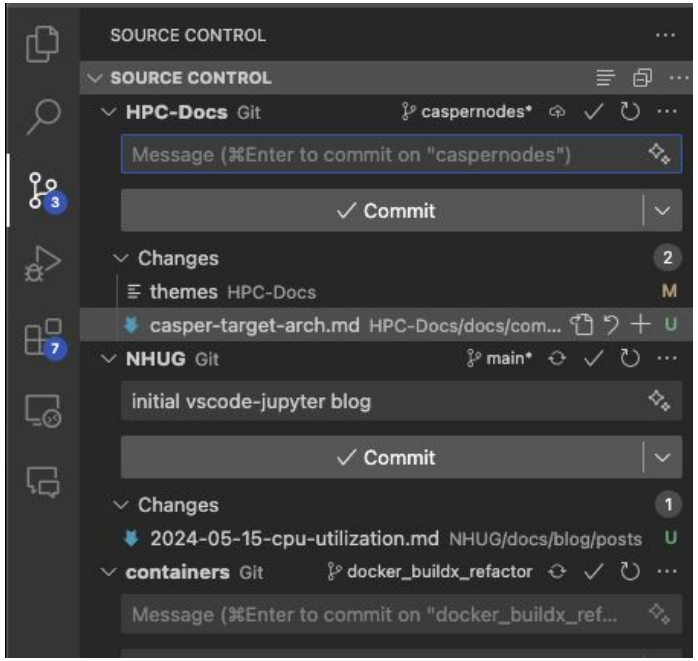
Why VSCode

File Explorer



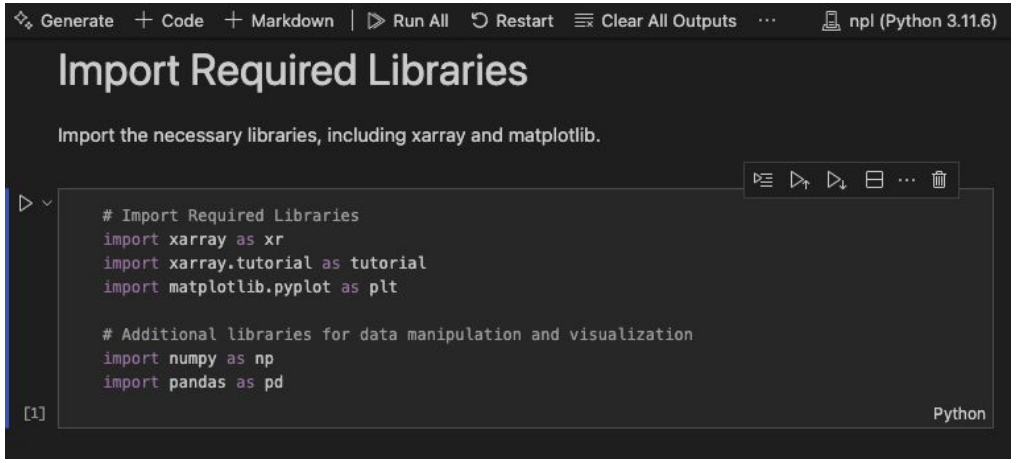
- GUI
- Connect to GLADE
- Transfer files between local and remote machines!

Source Control



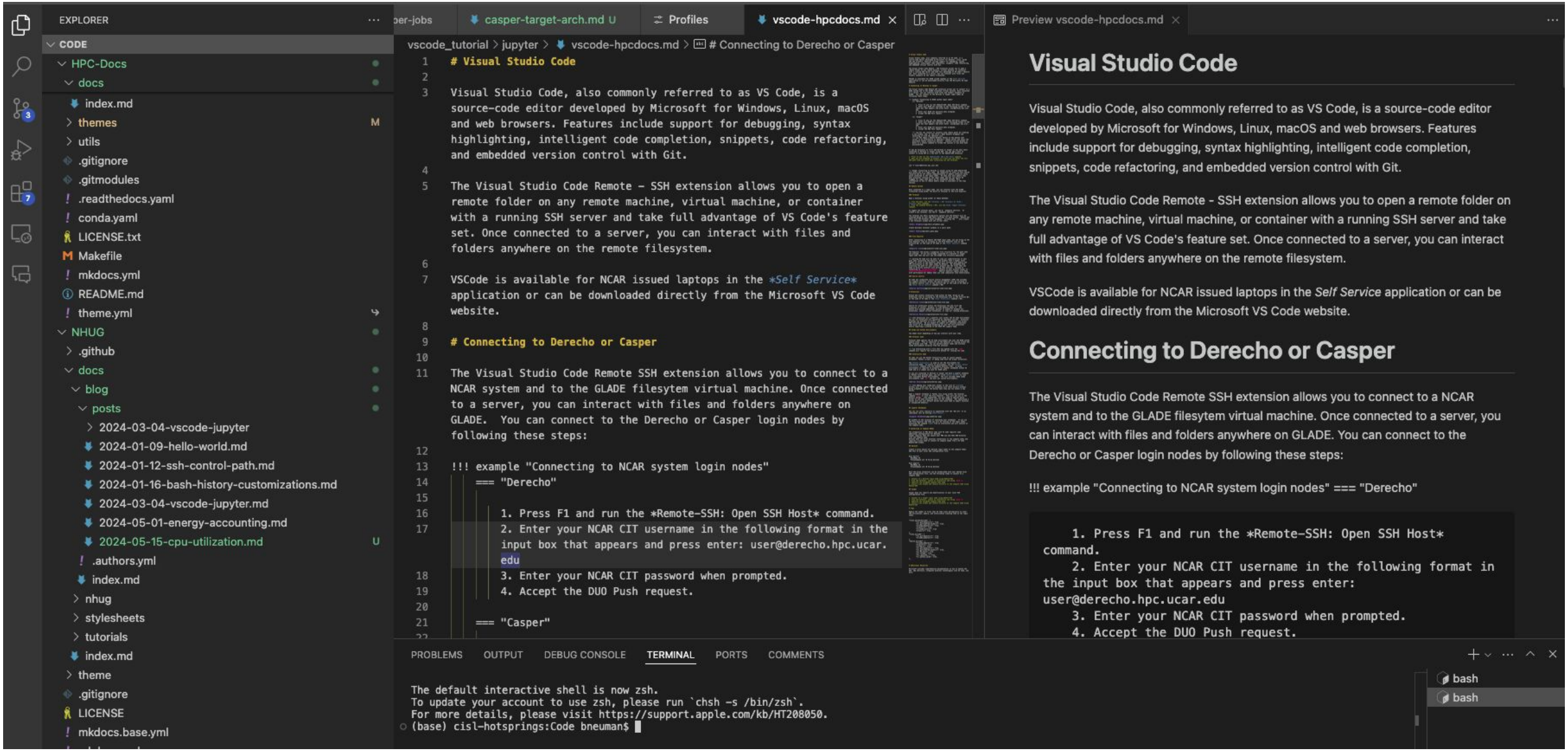
- Git visualizations
- Shortcut buttons for common operations

Extensions



- Jupyter notebooks
  - With your custom NCAR or local kernels
- RemoteSSH
  - Connect to Derecho and Casper
- Code Autocomplete (Intellisense)
  - Separate extensions from Copilot
- Copilot

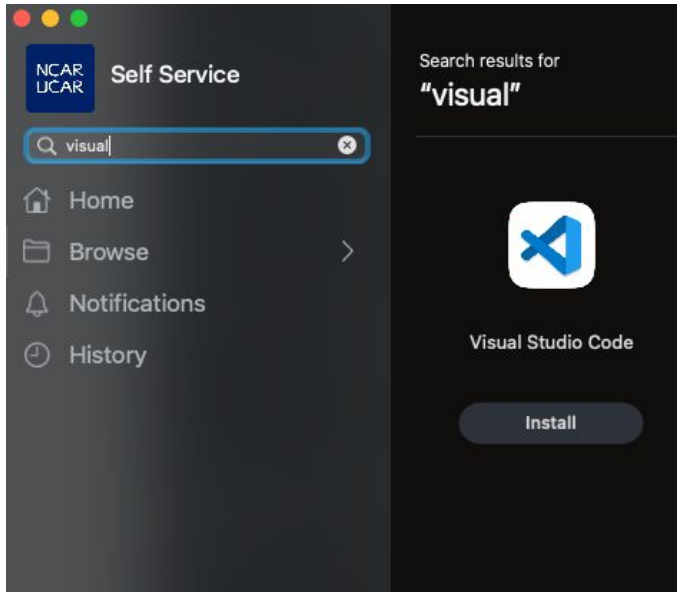
Why VSCode


EXPLORER
CODE
HPC-Docs
docs
index.md
themes
utils
.gitignore
.gitmodules
.readthedocs.yaml
conda.yaml
LICENSE.txt
Makefile
mkdocs.yml
README.md
theme.yml
NHUG
.github
docs
blog
posts
2024-03-04-vscode-jupyter
2024-01-09-hello-world.md
2024-01-12-ssh-control-path.md
2024-01-16-bash-history-customizations.md
2024-03-04-vscode-jupyter.md
2024-05-01-energy-accounting.md
2024-05-15-cpu-utilization.md
.authors.yml
index.md
nhug
stylesheets
tutorials
index.md
theme
.gitignore
LICENSE
mkdocs.base.yml
casper-target-arch.md
Profiles
vscode-hpcdocs.md
vscode_tutorial > jupyter > vscode-hpcdocs.md > # Connecting to Derecho or Casper
Visual Studio Code
Visual Studio Code, also commonly referred to as VS Code, is a source-code editor developed by Microsoft for Windows, Linux, macOS and web browsers. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded version control with Git.
The Visual Studio Code Remote - SSH extension allows you to open a remote folder on any remote machine, virtual machine, or container with a running SSH server and take full advantage of VS Code's feature set. Once connected to a server, you can interact with files and folders anywhere on the remote filesystem.
VSCode is available for NCAR issued laptops in the *Self Service* application or can be downloaded directly from the Microsoft VS Code website.
Connecting to Derecho or Casper
The Visual Studio Code Remote SSH extension allows you to connect to a NCAR system and to the GLADE filesytem virtual machine. Once connected to a server, you can interact with files and folders anywhere on GLADE. You can connect to the Derecho or Casper login nodes by following these steps:
!!! example "Connecting to NCAR system login nodes"
=== "Derecho"
1. Press F1 and run the *Remote-SSH: Open SSH Host* command.
2. Enter your NCAR CIT username in the following format in the input box that appears and press enter: user@derecho.hpc.ucar.edu
3. Enter your NCAR CIT password when prompted.
4. Accept the DUO Push request.
=== "Casper"
Preview vscode-hpcdocs.md
Visual Studio Code
Visual Studio Code, also commonly referred to as VS Code, is a source-code editor developed by Microsoft for Windows, Linux, macOS and web browsers. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded version control with Git.
The Visual Studio Code Remote - SSH extension allows you to open a remote folder on any remote machine, virtual machine, or container with a running SSH server and take full advantage of VS Code's feature set. Once connected to a server, you can interact with files and folders anywhere on the remote filesystem.
VSCode is available for NCAR issued laptops in the Self Service application or can be downloaded directly from the Microsoft VS Code website.
Connecting to Derecho or Casper
The Visual Studio Code Remote SSH extension allows you to connect to a NCAR system and to the GLADE filesytem virtual machine. Once connected to a server, you can interact with files and folders anywhere on GLADE. You can connect to the Derecho or Casper login nodes by following these steps:
!!! example "Connecting to NCAR system login nodes" === "Derecho"
1. Press F1 and run the *Remote-SSH: Open SSH Host* command.
2. Enter your NCAR CIT username in the following format in the input box that appears and press enter: user@derecho.hpc.ucar.edu
3. Enter your NCAR CIT password when prompted.
4. Accept the DUO Push request.
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) cisl-hotsprings:Code bneuman$
bash
bash

How to get VSCode

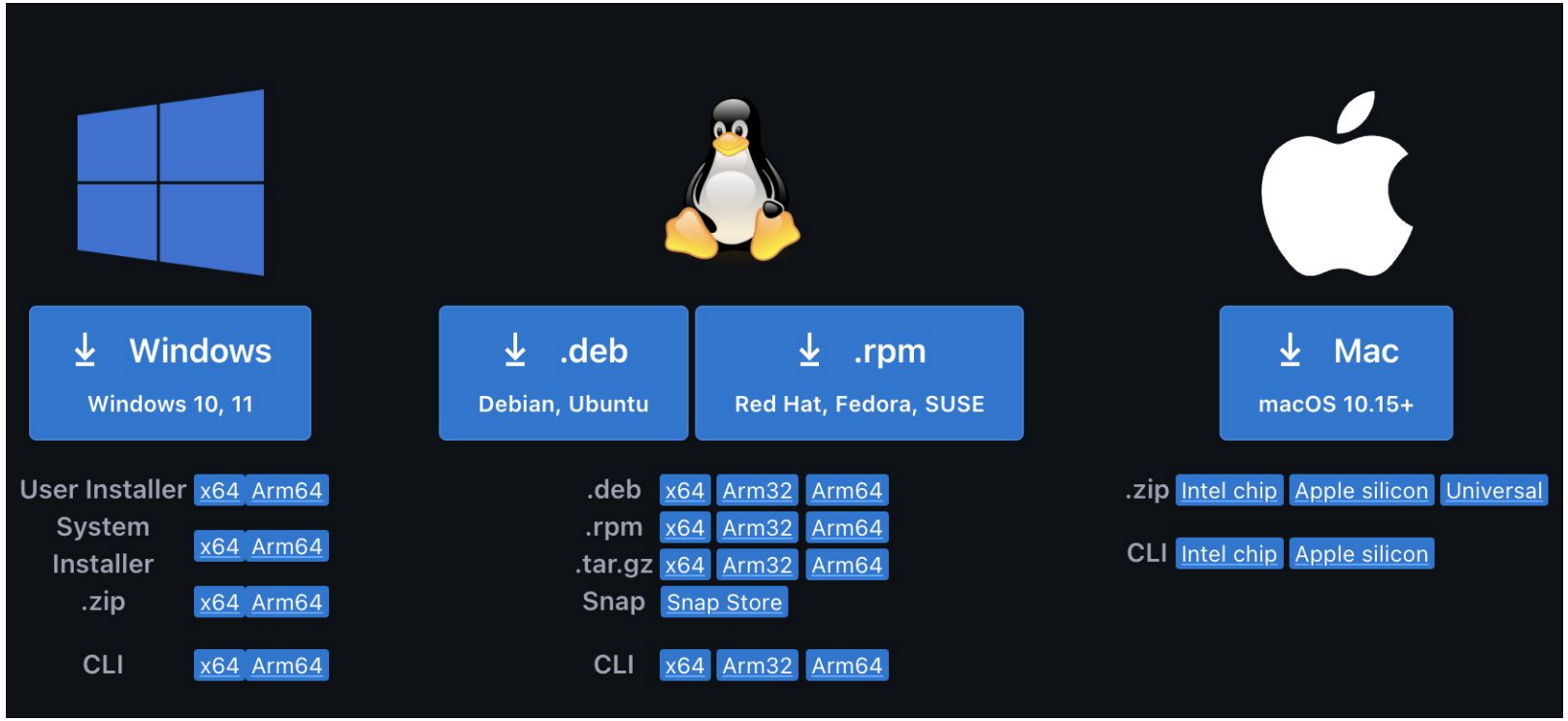
NCAR device

- Self Service



Otherwise

- https://code.visualstudio.com/download



Connecting to Derecho and Casper Login Nodes

The Visual Studio Code **RemoteSSH** extension allows you to connect to Derecho, Casper, and the GLADE filesystem. Once connected to a server, you can interact with files and folders anywhere on GLADE. You can connect to the Derecho or Casper login nodes by following these steps:

RemoteSSH

- *Press F1 and run the Remote-SSH: Open SSH Host command.*
- *Enter your NCAR CIT username in the following format in the input box that appears and press enter: user@casper.hpc.ucar.edu*
- *Enter your NCAR CIT password when prompted*
- *Accept the DUO Push request*

The >< Button on the bottom left can also be used to quickly connect or reconnect to remote systems

Once you have configured a remote server for the first time, you can use the local terminal to launch a remote window:

```
code --remote ssh-remote+derecho.hpc.ucar.edu
/glade/u/home/bneuman/myproject
```

Terminal vs. Interactive Modes

Terminal Mode

- Provides similar environment as the Mac Terminal over SSH
  - Ability to load modules, submit jobs, navigate filesystem, etc.
- Open files in the preview window using the command "`code <filename>`"
- Commands launched from the VSCode terminal will run on that node
  - e.g. On a compute node the terminal issued command will run on the compute node resources

Interactive Mode

- VSCode visualization functionality
  - Includes operations with the File Explorer
  - Opening files with the 'code' command to generate a preview window
- Utilizes the '.vs-server' folder and settings that are generated
  - Shows the running process as 'node'
- Will run on the node that RemoteSSH connected to

WARNING: You should always assume your code is running on the login node unless you connect a new window to a compute node with an active interactive PBS job

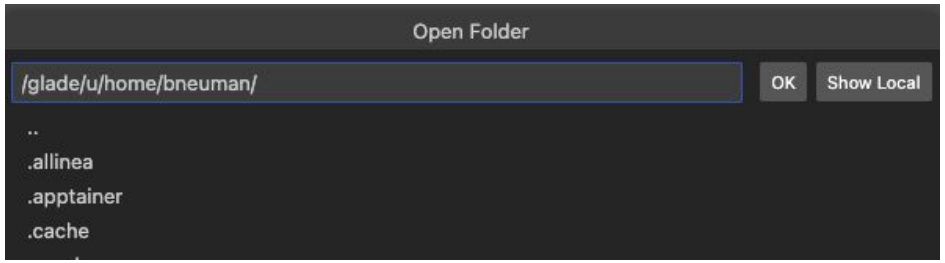
GLADE and File Explorer

Open Folder

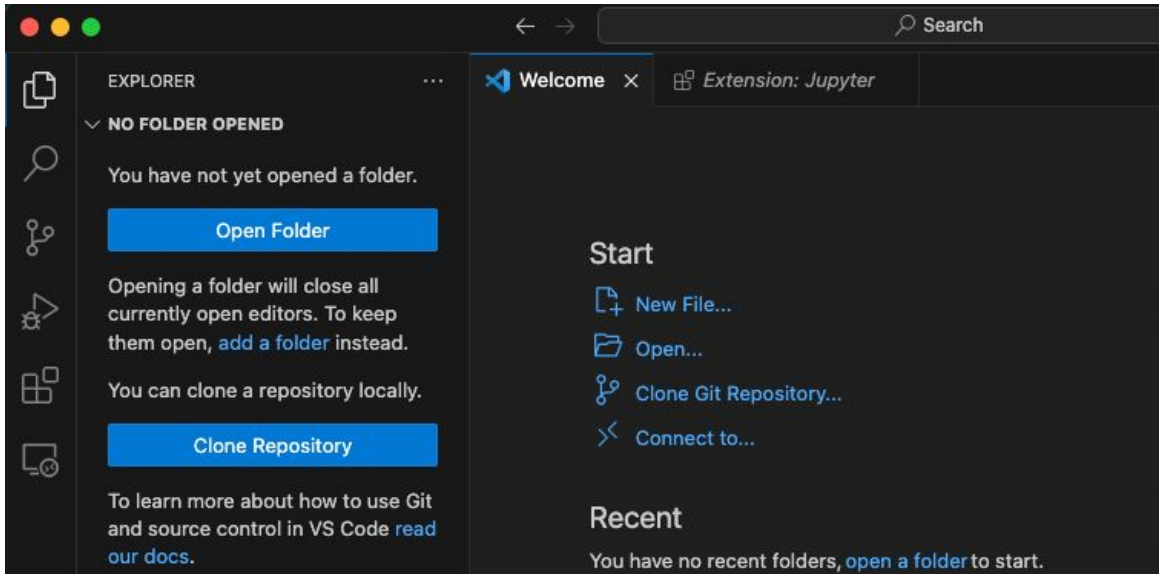
- Tries to open a new window

Opening a folder generates a “Workspace”

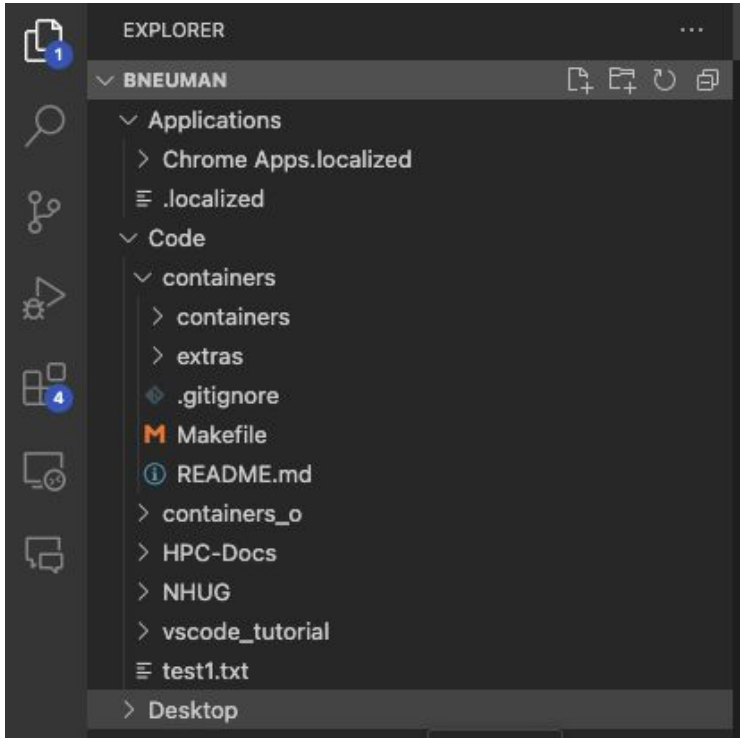
- VS Code scans this workspace for Intellisense and Copilot
- Be mindful of both CPU and memory utilization:
  - Intellisense will scan the entire workspace repeatedly
  - A larger workspace uses more memory on the login nodes



Prompt provides path to GLADE when connected to Derecho or Casper



Prompt to open a local or remote folder



Default File Explorer view

File Manager Resources

Reduce memory and code scan delay

- Select a small folder to open
  - Limitations due to hidden folders
    - Usually there are repos
  - Control over what is visible in a workspace

- Modify the Settings file and exclude files matching patterns related to git, conda environments, and hidden files
  - *Code* > *Preferences* > *Settings*
  - Still scans the workspace

```
"files.watcherExclude": {
    "**/.git/objects/**": true,
    "**/.git/subtree-cache/**": true,
    "**/node_modules/**": true,
    "**/dist/**": true,
    "**/conda-envs/**": true
    "**/env/**": true,
},
"files.exclude": {
    "**/.git/objects/**": true,
    "**/node_modules/**": true,
},
"search.exclude": {
    "**/node_modules/**": true,
    "**/dist/**": true,
    "**/tmp/**": true,
    "**/.git/objects/**": true,
    "**/.git/subtree-cache/**": true,
    "**/.DS_Store": true,
    "**/.vscode": true,
    "**/__pycache__": true,
    "**/.pytest_cache": true,
},
```

https://ncar-hpc-docs.readthedocs.io/en/latest/environment-and-software/vscode/#file-explorer

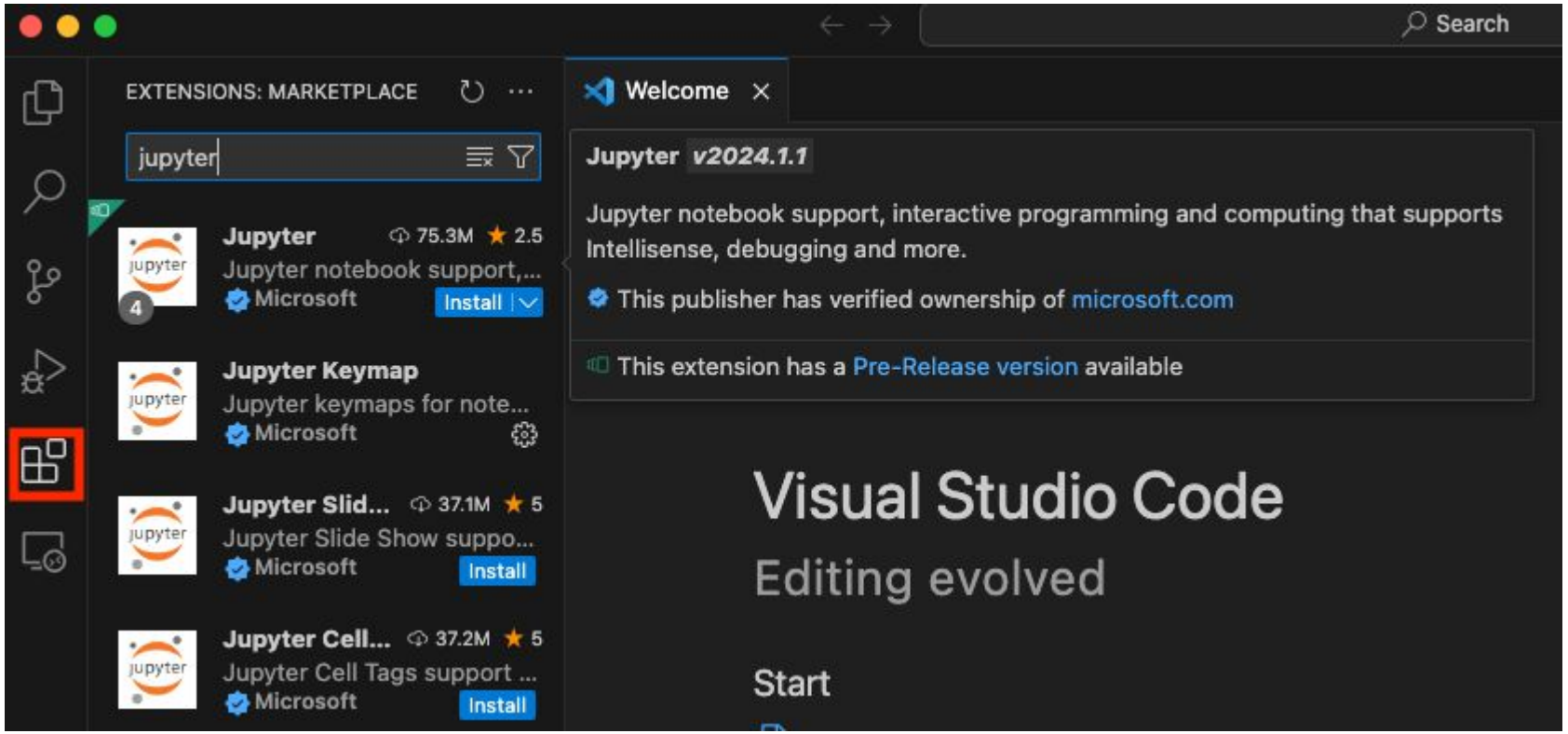
Jupyter Notebooks

Jupyter Extension

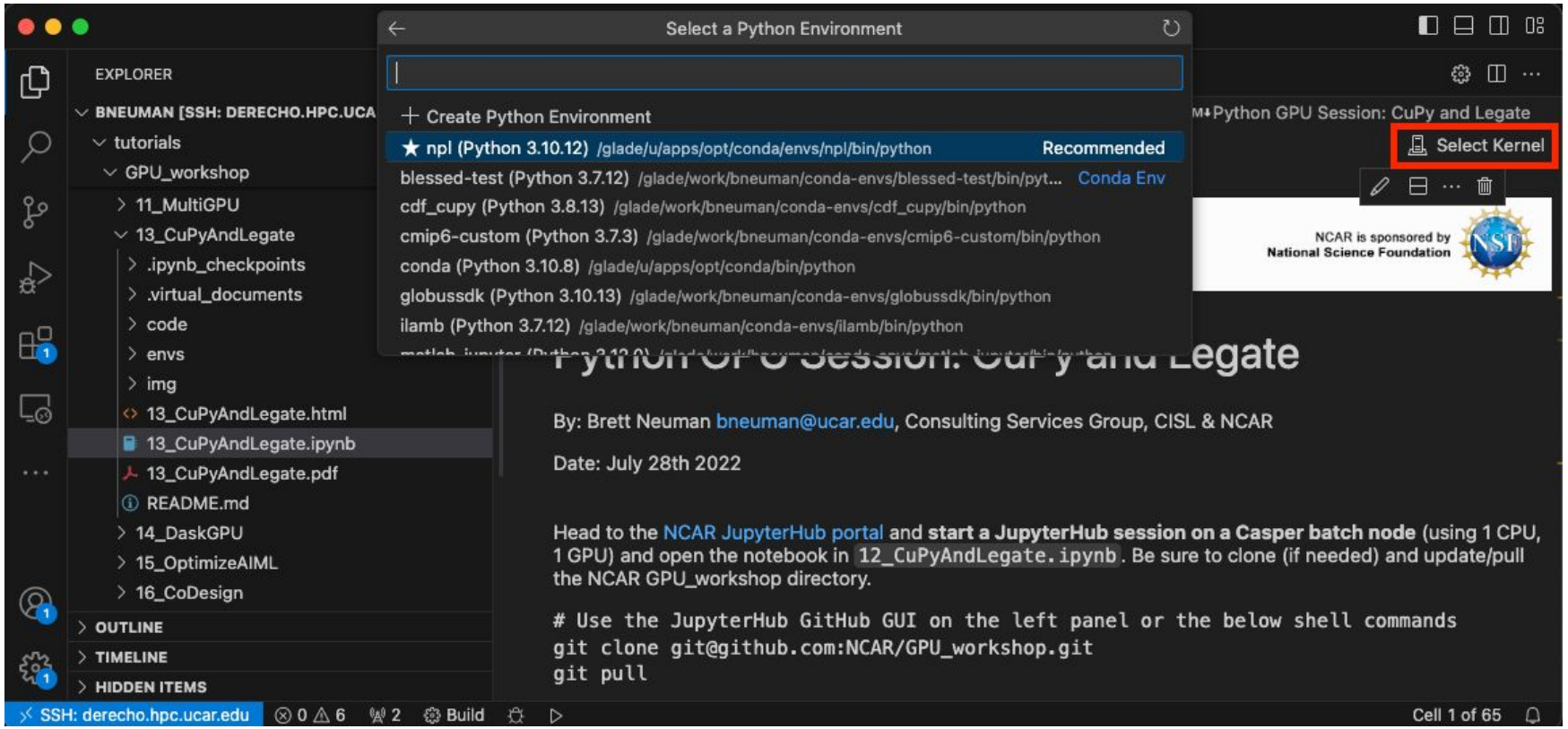
- Navigate to the Extensions tab
- Search for Jupyter
- To use custom conda environments to be available in VSCode, you must use the script:

  `create_conda_kernel`

  - This also makes the kernel available in Jupyterhub



Jupyter Notebook Kernels



Kernel Select

- To select an environment, use the Python: Select Interpreter command from the Command Palette (⇧⌘P) or F1
- Kernel can also be selected with the ‘Select Kernel’ on an open notebook

Connecting to Compute Nodes

For large, interactive workloads you should connect to Casper compute nodes

- Steps to connect to a compute node interactive session:
  1. Connect to a Casper login node using Remote-SSH
  2. From the Terminal, launch an interactive job using qsub -I

     ```
     qsub -I -A <project> -q casper -l select=1:ncpus=4:mpiprocs=4 -l
     walltime=01:00:00
     ```

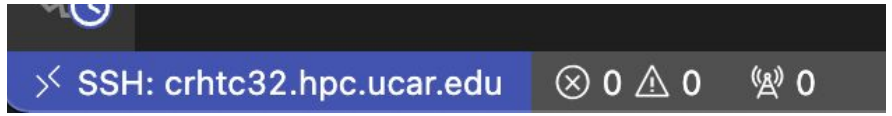
  3. Identify the assigned compute node name:

     ```
     $USER@crhtc32:~>

     echo $HOSTNAME
     ```

  4. Launch a new window and connect directly to the compute node using Remote-SSH:

     ```
     bneuman@crhtc32.hpc.ucar.edu
     ```



Derecho will only run interactive workloads on login nodes

- Current security on Derecho prevents direct login to compute nodes from VSCode's RemoteSSH

Copilot License

- Join the NCAR GitHub Organization: https://github.com/NCAR

- Coordinate with your lab's budgeting for a project key (billing key) for charging CoPilot Charges ($19/month).

- Submit a Help Request:
  - Create a request through the NCAR/UCAR Service Desk by emailing help@ucar.edu or using the Service Desk portal.
  - Provide your GitHub ID and Project Key for charging CoPilot charges.

- Install the Github Copilot Extension in VSCode and login using your Github credentials

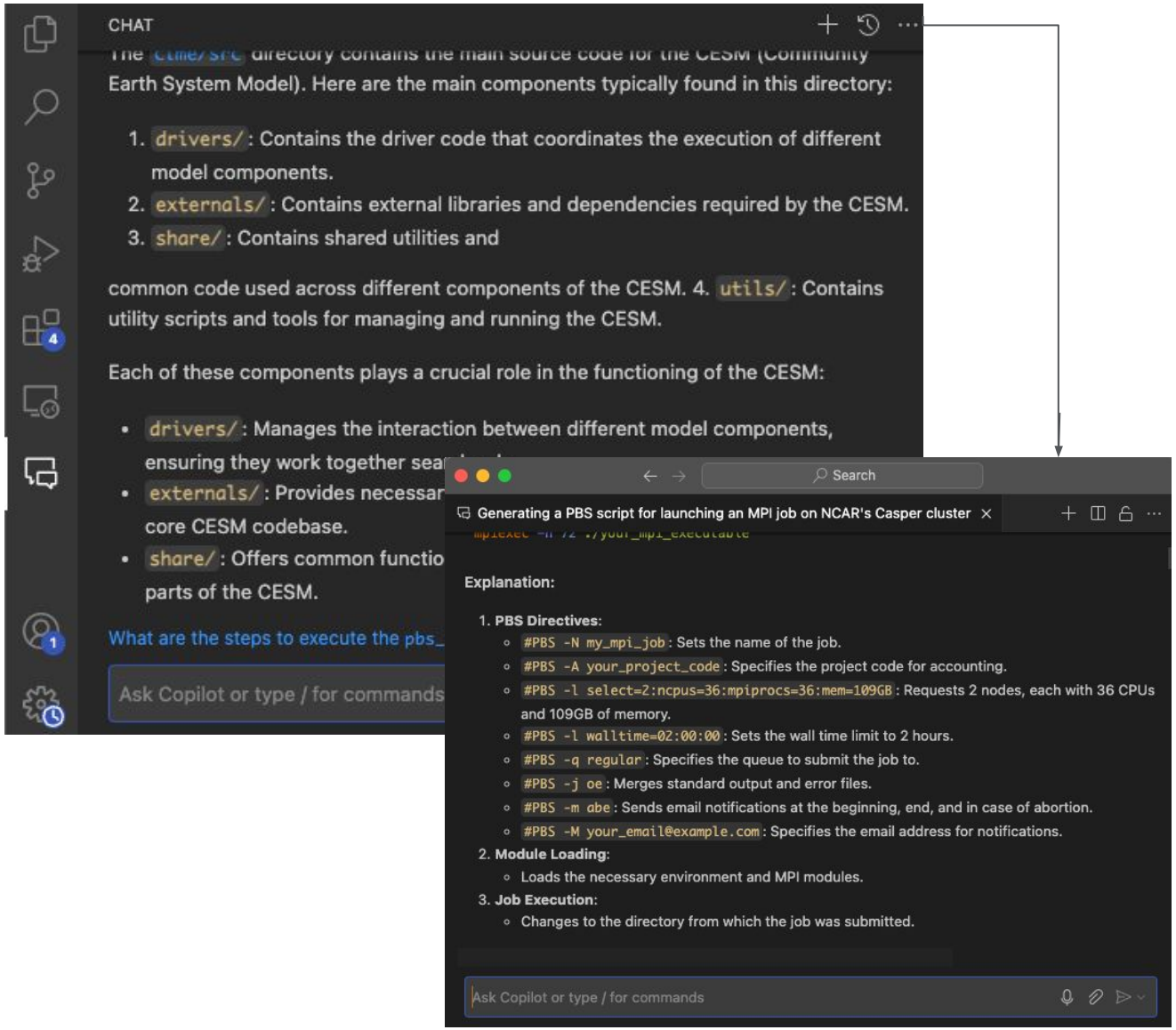
The Service Desk will process your request, confirm the billing information, and activate the license.

Note: NCAR only offers GitHub Copilot for Business, not the Enterprise tier.

Copilot in VSCode and the Chat Window

Copilot Chat

- Menu pane -> Copilot
  - Opens default chat window
- Text based generation via prompts
- Scope is contained to the running chat
  - New chat can be started with the +
  - Move existing chat to a new window or editor:
- … -> Open in New Window
  - New Window maintains scope and allows for new chat windows in parallel

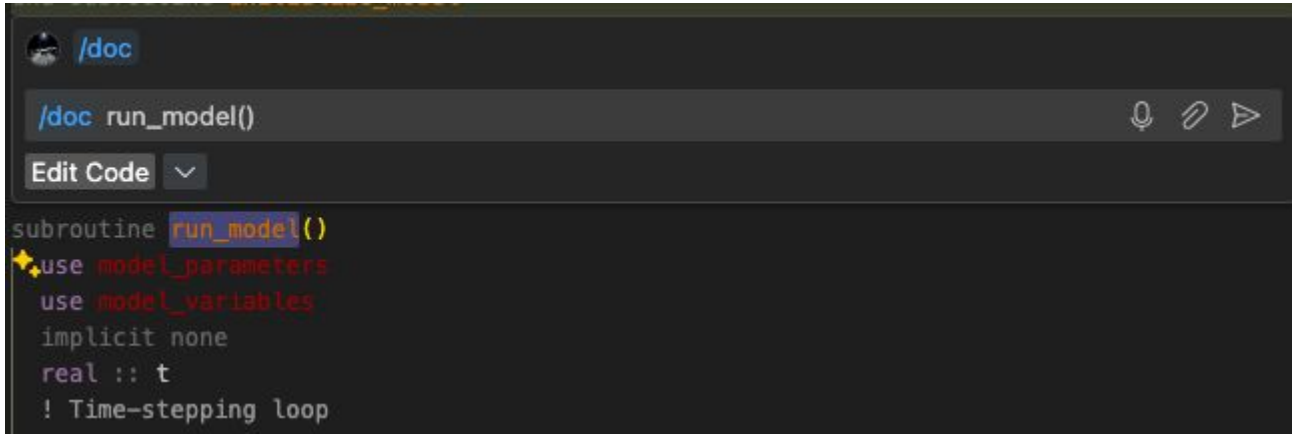


Copilot Chat vs. Copilot In-context

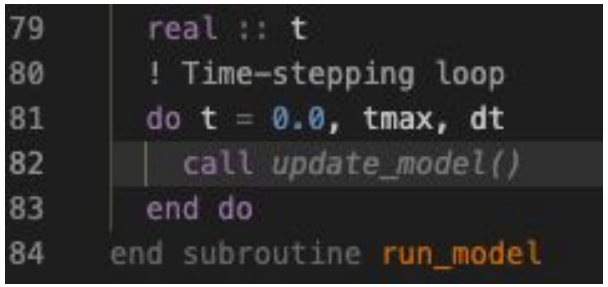
Copilot prompt in your active window

- Open the Copilot prompt with ⌘I on your active preview window
  - Highlight sections of code to target or use Function names

- Copilot Autocomplete attempts to generate new code based on scope



Copilot prompt opened in code editor



Copilot Autocomplete

Energy Use and Copilot

2022 Paper "Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model" (*Luccioni, Viguier, Ligozat, https://arxiv.org/pdf/2211.02001*)

- BLOOM:
  - 176 Billion parameters
  - Trained on 1.6 Petabytes of data

Total training time	118 days, 5 hours, 41 min
Total number of GPU hours	1,082,990 hours
Total energy used	433,196 kWh
GPU models used	Nvidia A100 80GB
Carbon intensity of the energy grid	57 gCO_2eq/kWh

Table 1: Key statistics about BLOOM model training – for more details about our methodology, see Section 4.2.

Table from https://arxiv.org/pdf/2211.02001

Inferring Energy Usage for Training

Projected power and CO2 emissions based on BLOOM

- Factors in three phases
  - Total power consumption for *Training*
  - CO2 Emissions of *Dynamic Energy* during Training
  - CO2 Emissions of *Dynamic Energy* and *Passive Datacenter Energy* during Training

Model name	Number of parameters	Datacenter PUE	Carbon intensity of grid used	Power consumption	CO_2eq emissions	CO_2eq emissions × PUE
GPT-3	175B	1.1	429 gCO_2eq/kWh	1,287 MWh	*502 tonnes*	552 tonnes
Gopher	280B	1.08	330 gCO_2eq/kWh	*1,066 MWh*	*352 tonnes*	380 tonnes
OPT	175B	*1.09* [2]	*231gCO_2eq/kWh*	*324 MWh*	70 tonnes	*76.3 tonnes* [3]
BLOOM	176B	1.2	57 gCO_2eq/kWh	433 MWh	25 tonnes	30 tonnes

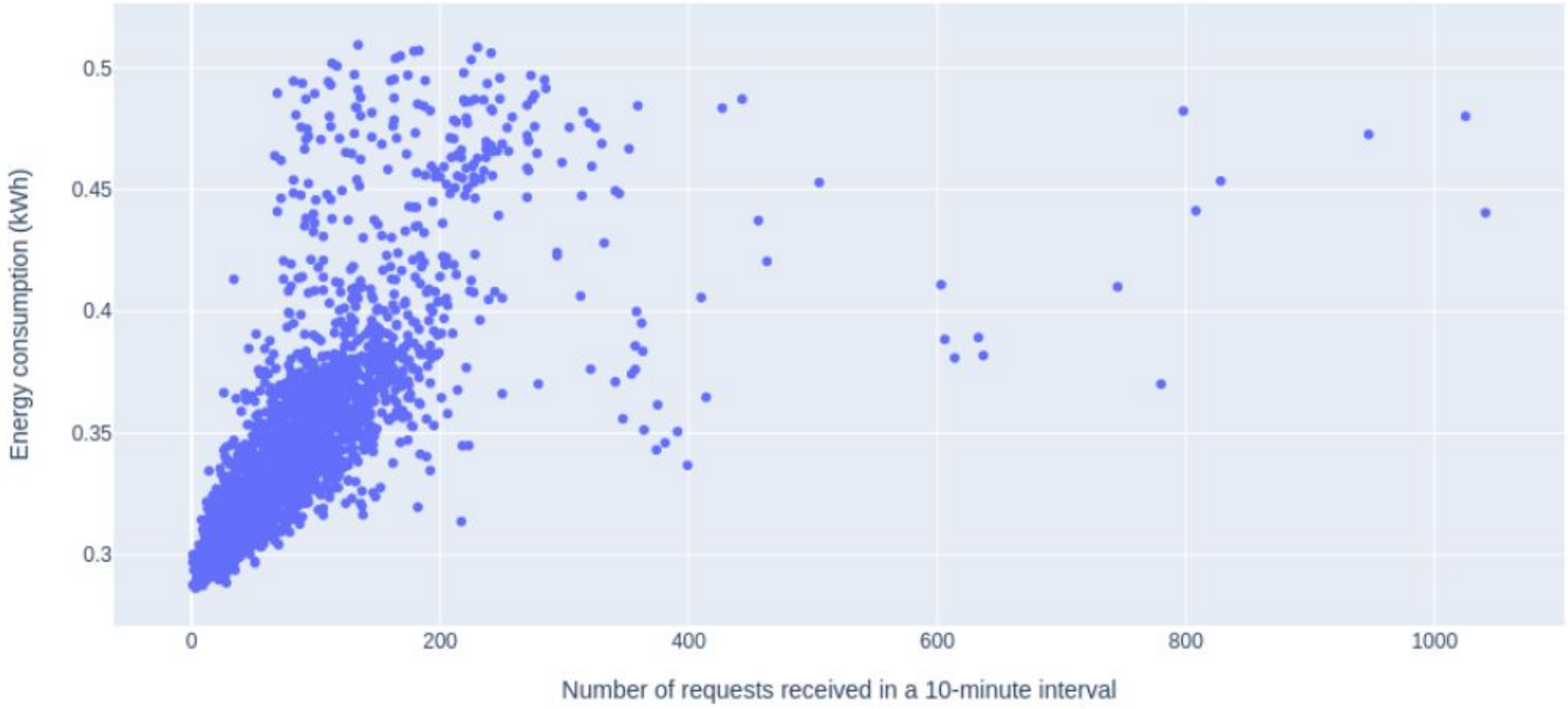
Table 4: Comparison of carbon emissions between BLOOM and similar LLMs. Numbers in *italics* have been inferred based on data provided in the papers describing the models.

Table from https://arxiv.org/pdf/2211.02001

Energy Usage - Post Training

Energy used per prompt estimations

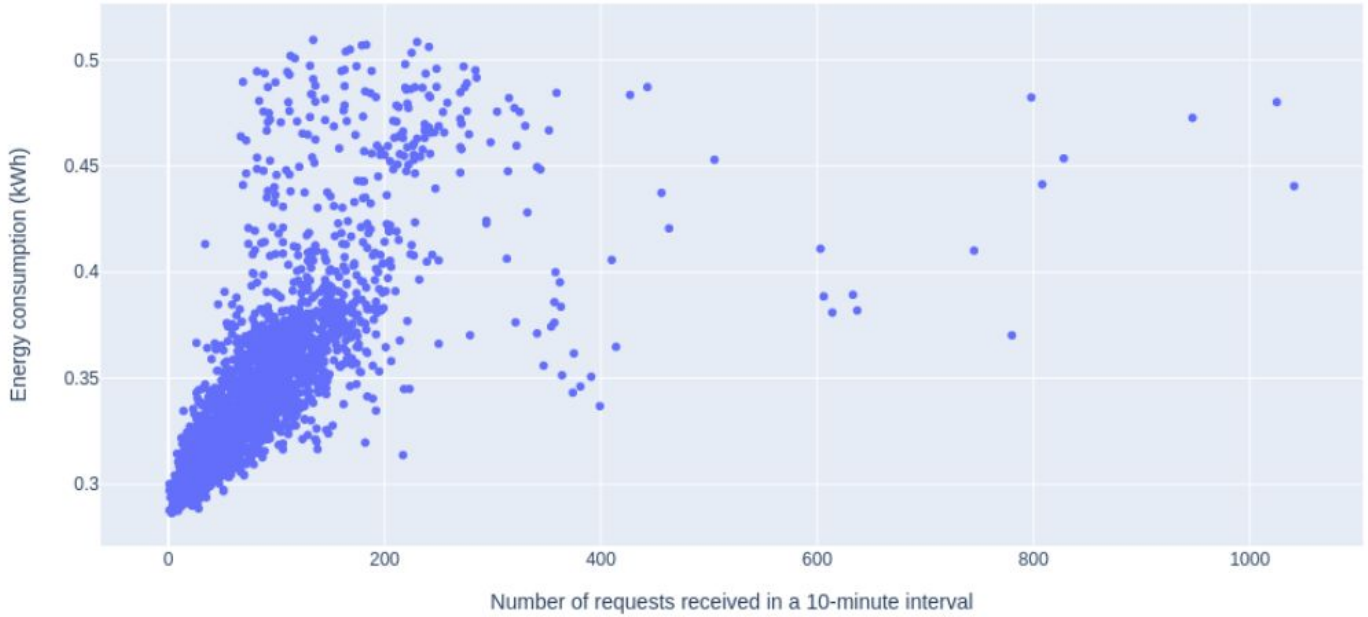
- ~1 request in a 10 minute window uses 0.30kWh of energy
  - Near idle amount of energy for running A100 GPU (idle measured at ~0.28kWh)
- Consumption energy mostly scales with requests
  - Occasional spikes in energy consumption



16x A100 GPUs running BLOOM, 176B Parameter LLM

Energy Usage - Post Training

Prompts	kWh	Light bulbs (LED 18Wh/bulb)
0	0.28	15.55
1	0.30	16.67
100	0.35	19.44
200	0.45	25.00
Prompts	**kWh**	**Liters of water (0.21MJ/liter)**
0	0.28	4.80
1	0.30	5.15
100	0.35	6.00
200	0.45	7.72



Gleick, P.H. and Cooley, H.S. "Energy implications of bottled water." *Environmental Research Letters* 4 (2009) 014009

Copilot Agents

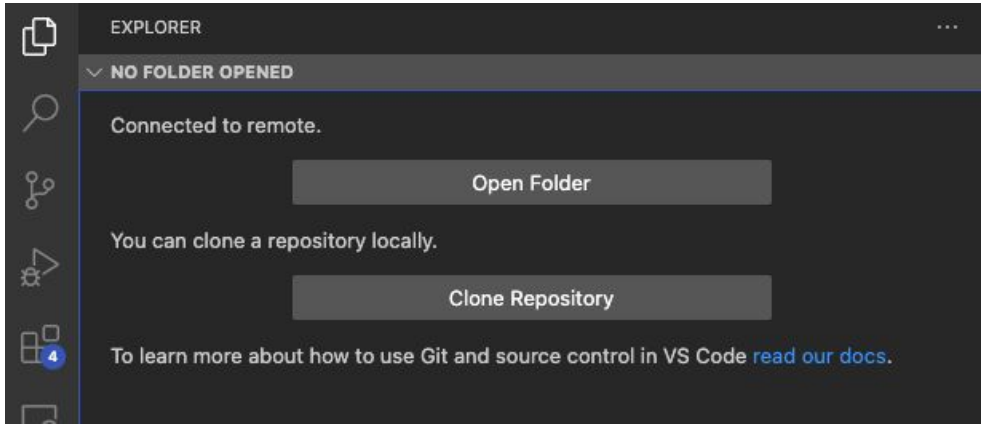
Prompts can be directed to produce better results:

- Using @workspace before you prompt gives more control on how the AI will process your prompt
- Available agents:
  - **@github**
    - Get answers grounded in web search and code search
    - Takes less time to produce results compared to *@workspace*
  - **@terminal**
    - Ask how to do something in the terminal
  - **@workspace**
    - Ask about your workspace
    - Can take significantly longer to produce results
      - More energy due to additional inference
      - Still sends your prompt to model with additional inputs from workspace
  - **@vscode**
    - Ask questions about VS Code

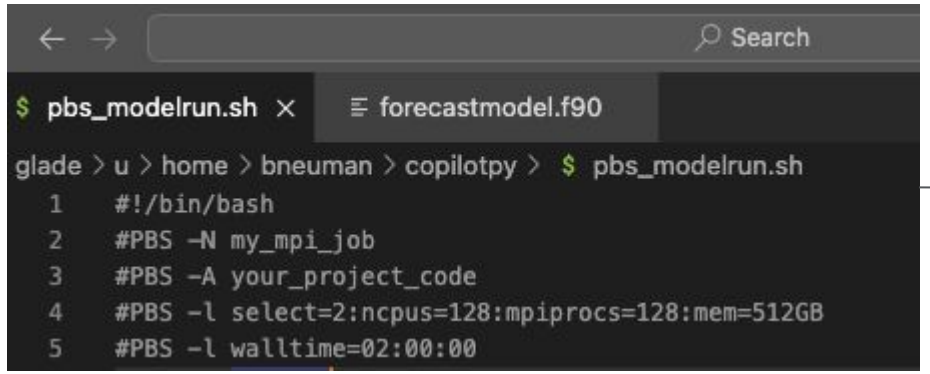
Workspace

Workspaces

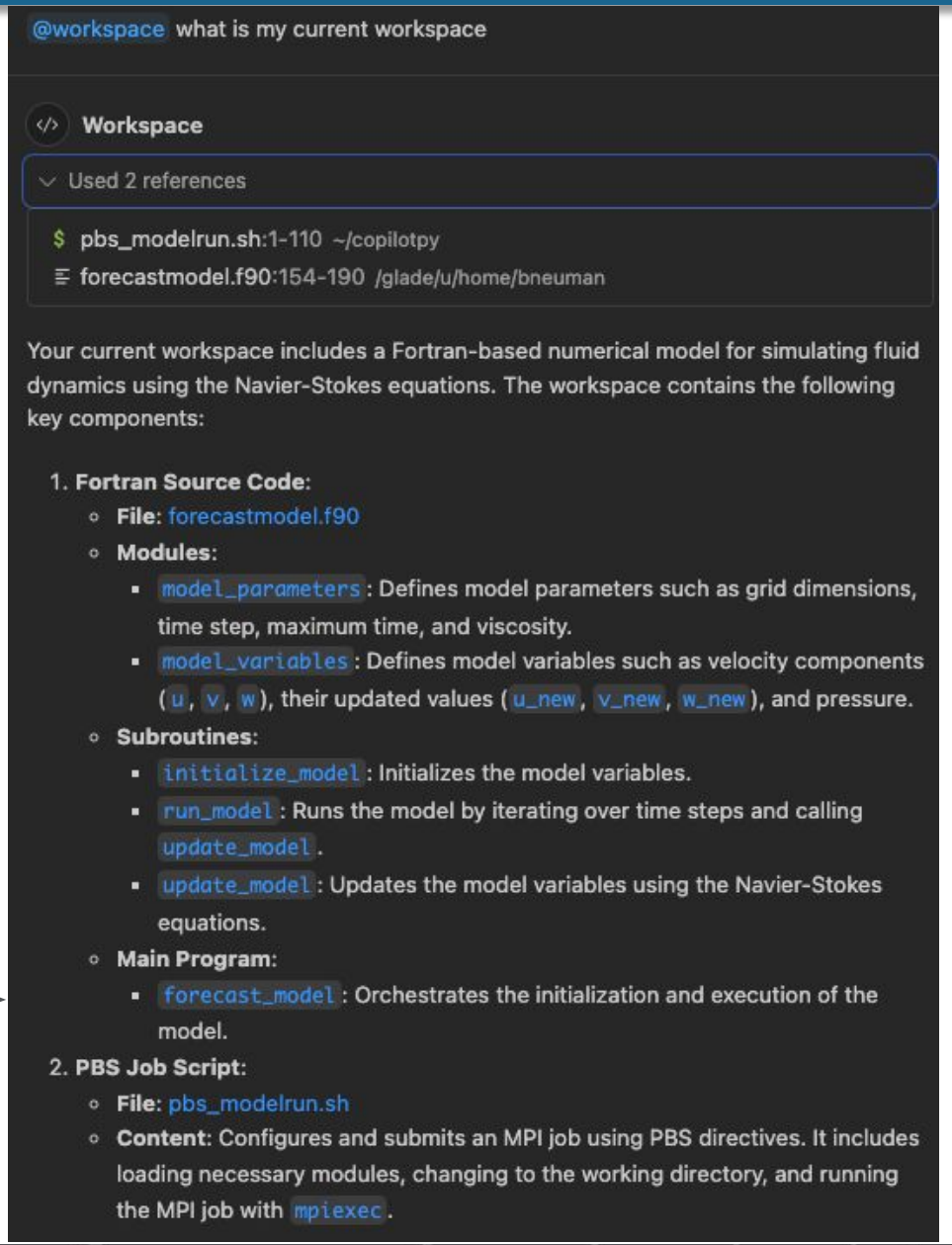
- Copilot defaults to *only* using the open files in the preview window



No path open in File Explorer



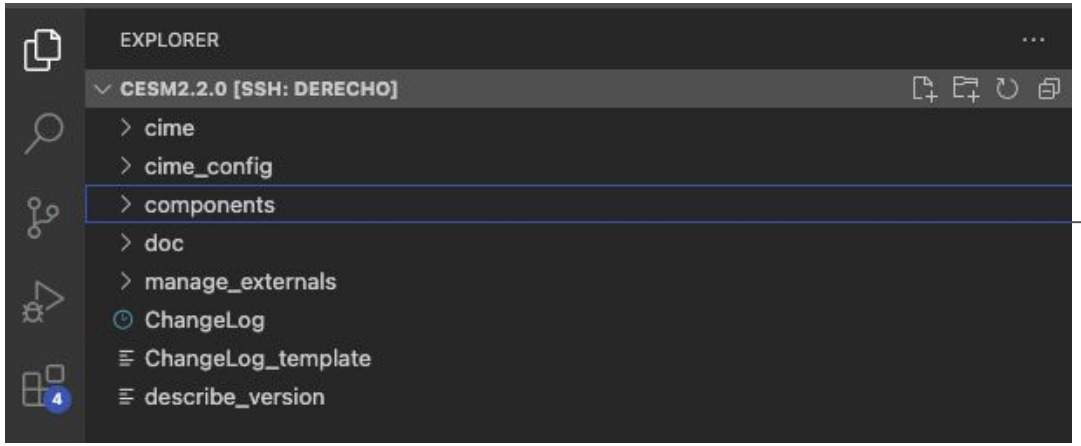
Files open in preview window



Workspaces with File Explorer

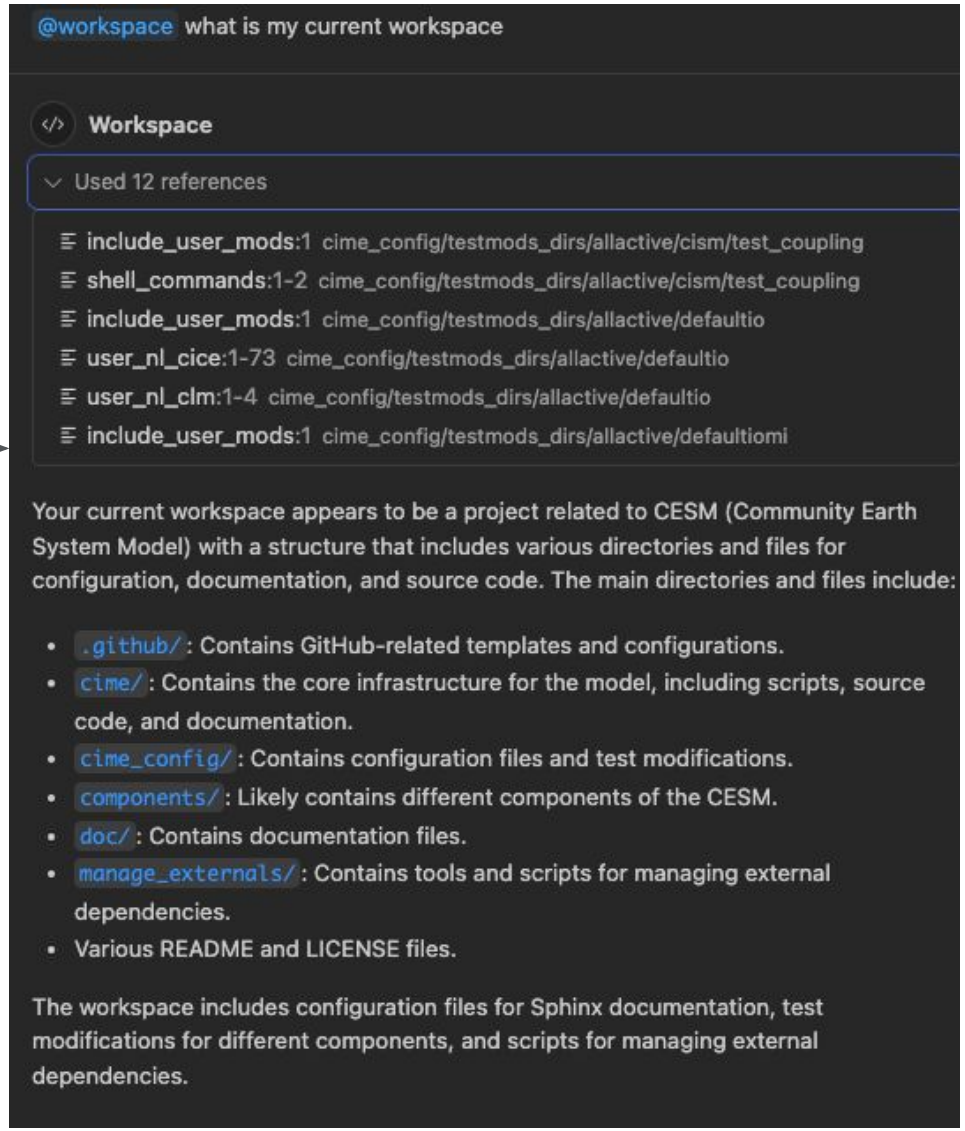
Workspaces with a path in File Explorer

- Searches entire workspace
- Prompt generation can take significantly longer



CESM path on GLADE open in File Explorer

No files open in preview window

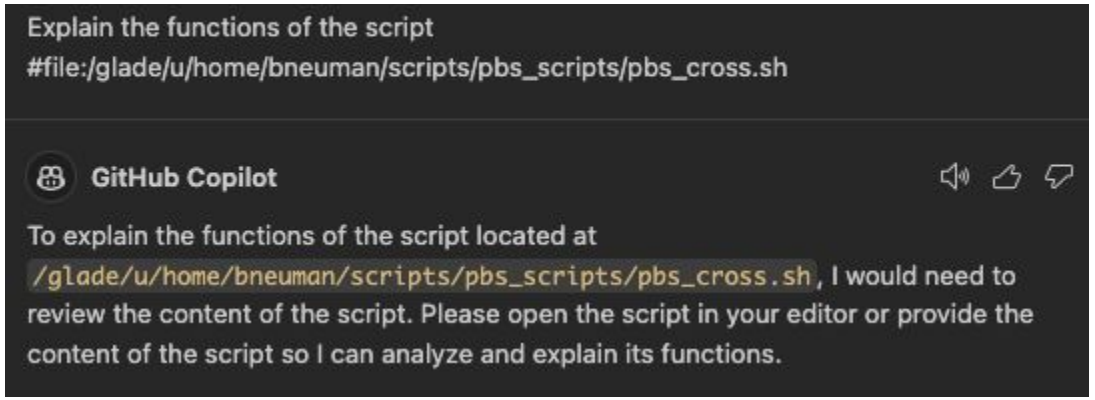


Methods to Specify Prompt Inputs

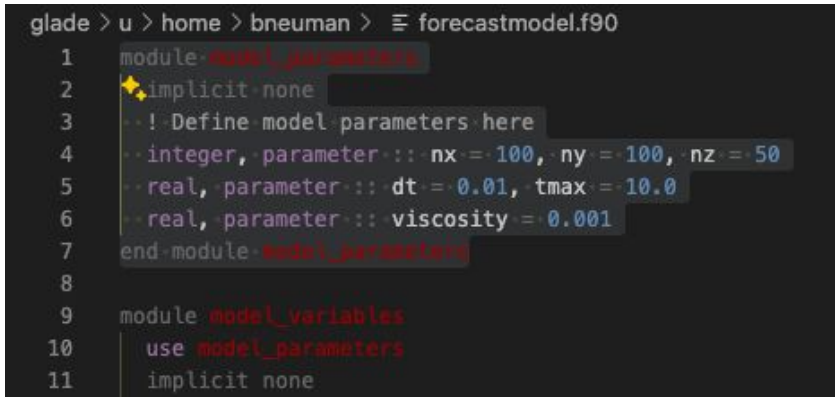
Additional methods to drill down on paths, files, or highlighted code

- **#editor**
  - The visible source code in the active editor
- **#file**
  - Specify a file
    - `Explain #file:src/main.cpp`
  - Still bound by the workspace
    - Won't open / analyze files outside of it
- **#selection**
  - Highlighted portion of an open file in the active window
    - `Explain #selection`
- **#terminalLastCommand**
- **#terminalSelection**

Not all options are available for in-context code

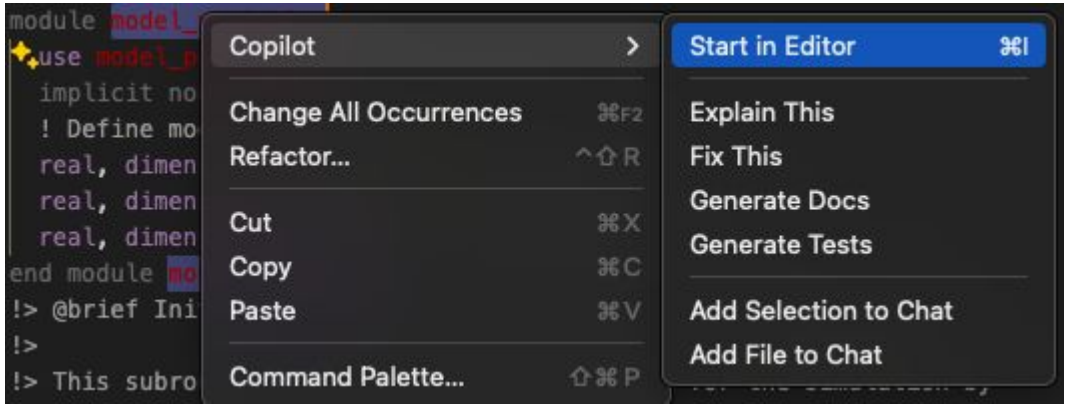


Limitation when selecting a file outside of your workspace



Highlight a portion of the code to use in *#selection*

Shortcut Behaviors



These Copilot shortcuts are available in-context by right clicking and selecting from the list

Common tasks have shortcuts available

- **/explain**
  - (@workspace) Explain how the code in your active editor works
    - defaults to @workspace scope
  - `/explain #file:src/main.cpp`
- **/fix**
  - (@workspace) Propose a fix for the problems in the selected code
    - Highlight sections to fix
- **/doc**
  - Add documentation for a symbol
    - Works on highlighted text
    - `/doc run_model()`
  - Specify documentation standards
    - `/doc using doxygen comment standard`
- **/tests**
  - Generates unit tests for highlighted code

Combining Agents and Copilot Functions + Custom Agents

Combining the three selections for a Prompt

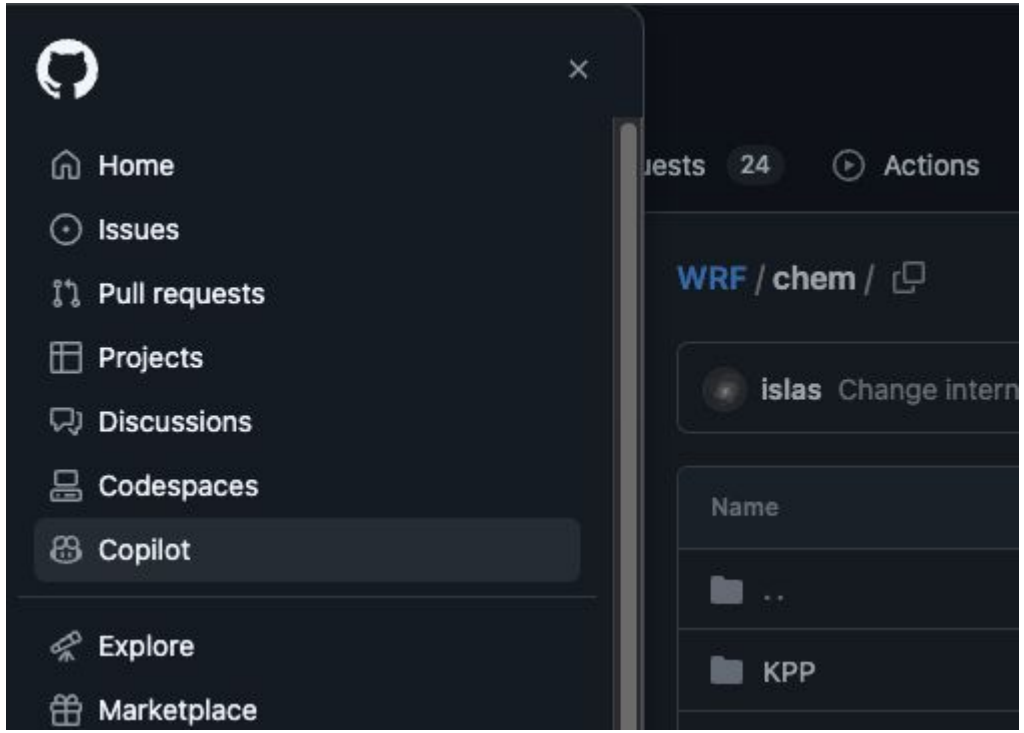
- Efficient way to prompt
- Provides more control over scope of data
  - `@workspace /explain #file:src/main.cpp`
  - `@workspace /fix #file:fibonacci.f90`
  - `@github /explain`

Copilot directly through Github repository

- Menu -> Copilot -> Search for a repository
  - Limited unless repository is indexed
    - Limit to number of indexed repositories

Custom Agents

- It is possible to create custom agents and



Menu from https://www.github.com
Select Copilot to search for a repository
Generate a chat window for the selected repository

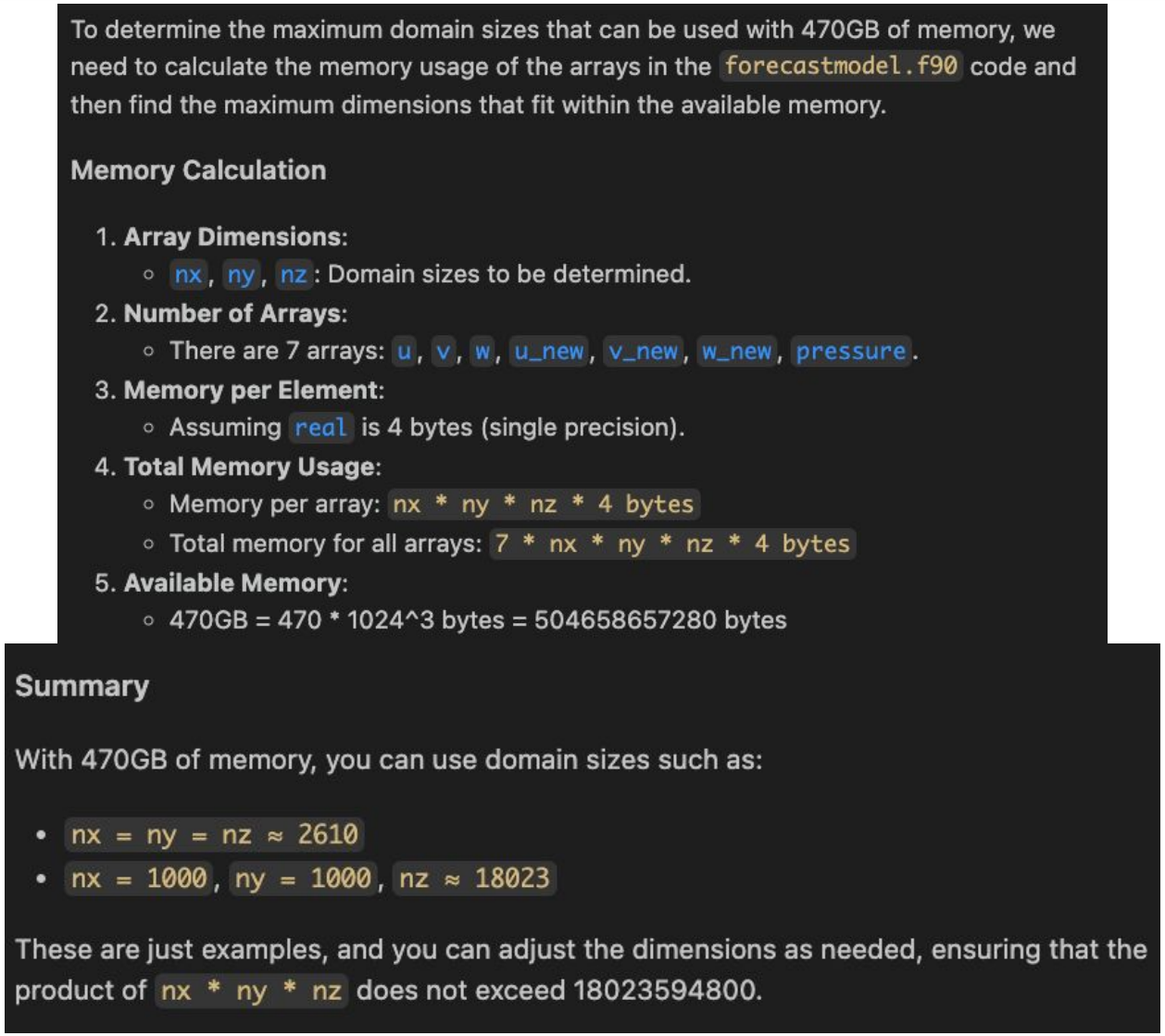
Discussion on Prompt Output

Running quite a few example prompts

- Based on *my experience with Copilot*
  - .. and a few similar reports of generating code

Things Copilot seems to be pretty good at:

- Scripting
  - bash, Python, etc.
- Code documentation
- Generating plots from data in scope
- Generalized code and functions
  - Webapps, Database operations, Fibonacci Sequence
- Estimating memory, MPI ranks, grid sizes
  - With user provided constraints



To determine the maximum domain sizes that can be used with 470GB of memory, we need to calculate the memory usage of the arrays in the `forecastmodel.f90` code and then find the maximum dimensions that fit within the available memory.

Memory Calculation

1. **Array Dimensions**:
   - `nx`, `ny`, `nz`: Domain sizes to be determined.
2. **Number of Arrays**:
   - There are 7 arrays: `u`, `v`, `w`, `u_new`, `v_new`, `w_new`, `pressure`.
3. **Memory per Element**:
   - Assuming `real` is 4 bytes (single precision).
4. **Total Memory Usage**:
   - Memory per array: `nx * ny * nz * 4 bytes`
   - Total memory for all arrays: `7 * nx * ny * nz * 4 bytes`
5. **Available Memory**:
   - 470GB = 470 * 1024^3 bytes = 504658657280 bytes

Summary

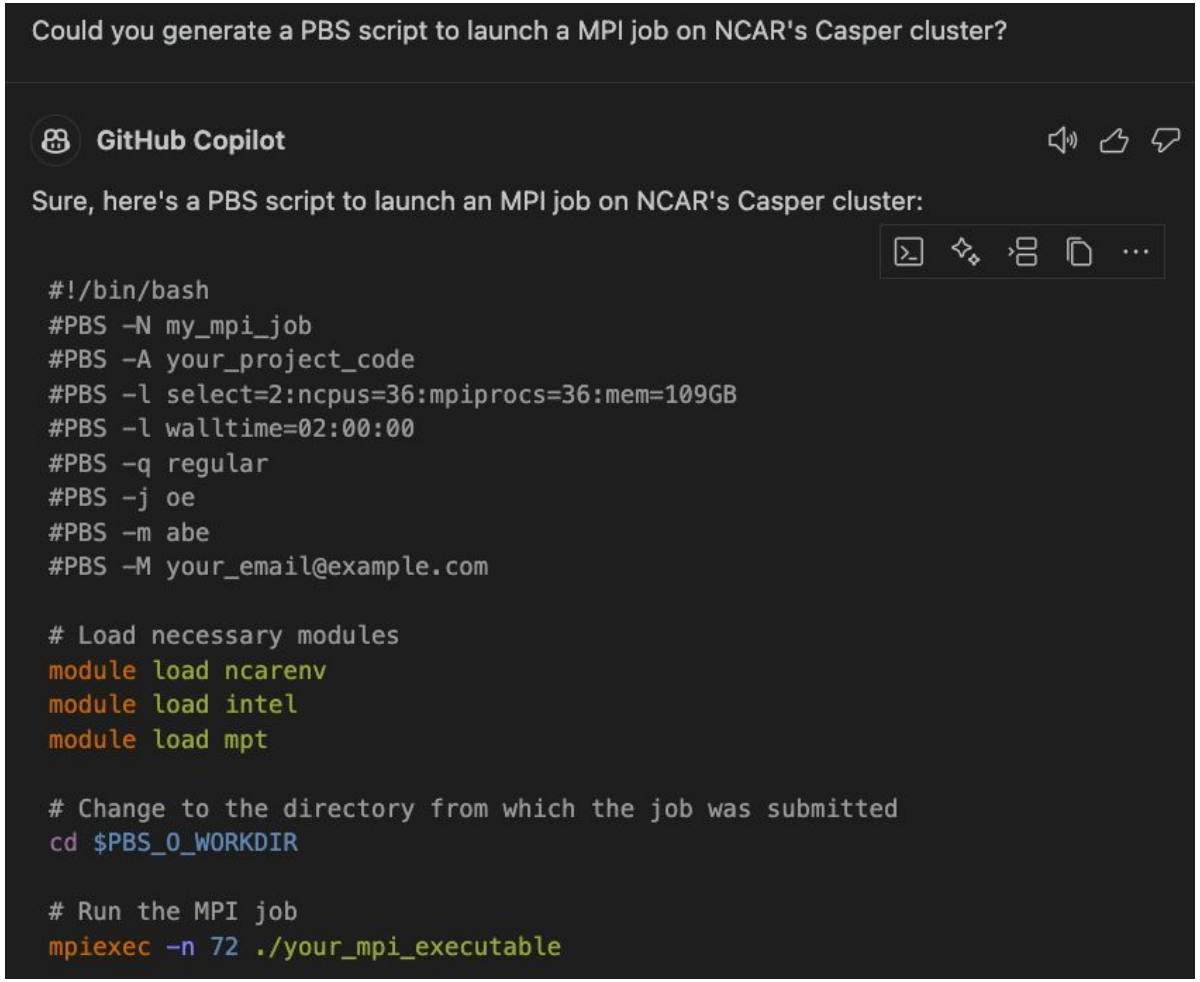
With 470GB of memory, you can use domain sizes such as:

- `nx = ny = nz ≈ 2610`
- `nx = 1000`, `ny = 1000`, `nz ≈ 18023`

These are just examples, and you can adjust the dimensions as needed, ensuring that the product of `nx * ny * nz` does not exceed 18023594800.

"What domain sizes can be used if I only have 470GB of memory?"

Discussion on Prompt Output - Pt 2



This pulls information from creating a PBS script for Cheyenne

Running quite a few example prompts

- Based on *my experience with Copilot*
  - .. and a few similar reports of generating code

Things Copilot seems it could work on improving:

- Dealing with large code bases
  - Will generate code in a function but never calls it, requires additional guidance
- Dealing with complex scientific HW/SW solutions
  - Requires many additional input constraints to provide non-trivial solutions
- Generating NCAR specific outputs
  - nco, PBS job scripts, etc.
  - @workspace of documentation does help with this