

HIGH PERFORMANCE RESEARCH COMPUTING

Programming Using the Jupyter AI Assistant

March 27th, 2026

Keegan Smith



High Performance
Research Computing

DIVISION OF RESEARCH



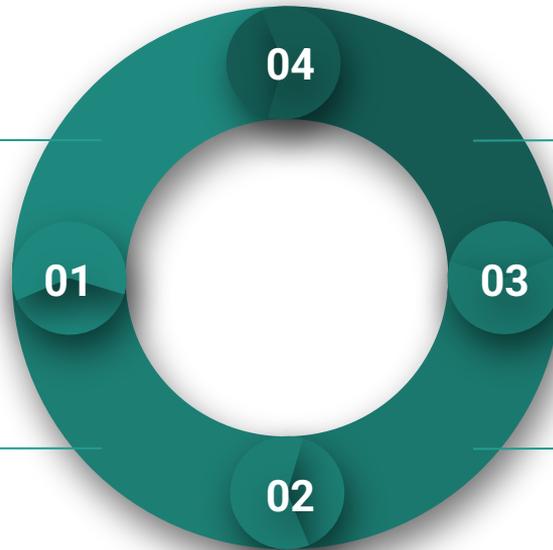
Schedule

Lab 1: Creating a session (15 minutes)

We will create a jupyter AI Assistant session from the ACES portal.

Lab 2: Debugging with Jupyter AI (15 mins)

We will go through some examples with two popular Python libraries: Pandas and Matplotlib for data exploration.



Lab 3: Creating a PyTorch Application (15 minutes)

We will create and debug a Pytorch application from scratch with the assistance of Jupyter AI.

Lab 4: Creating a Tensorflow Application (15 minutes)

We will create and debug a Tensorflow application from scratch with the assistance of Jupyter AI.

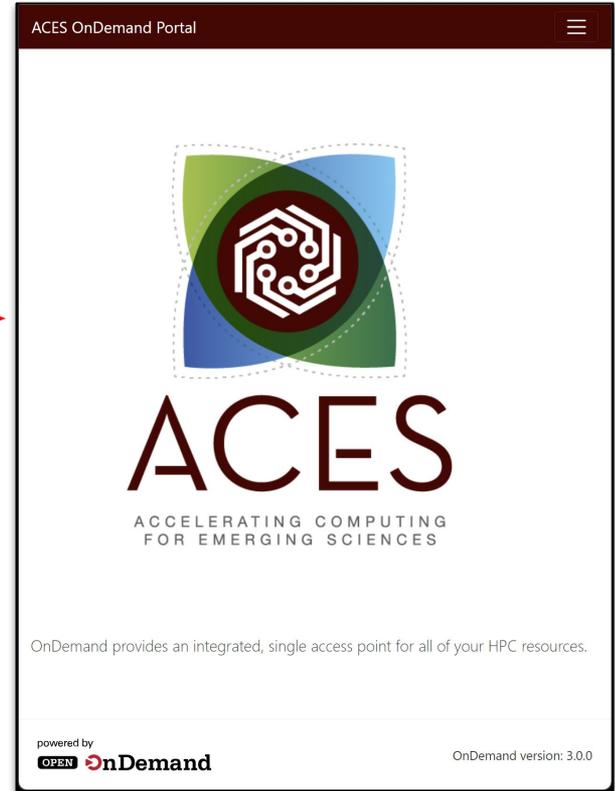


ACES Portal

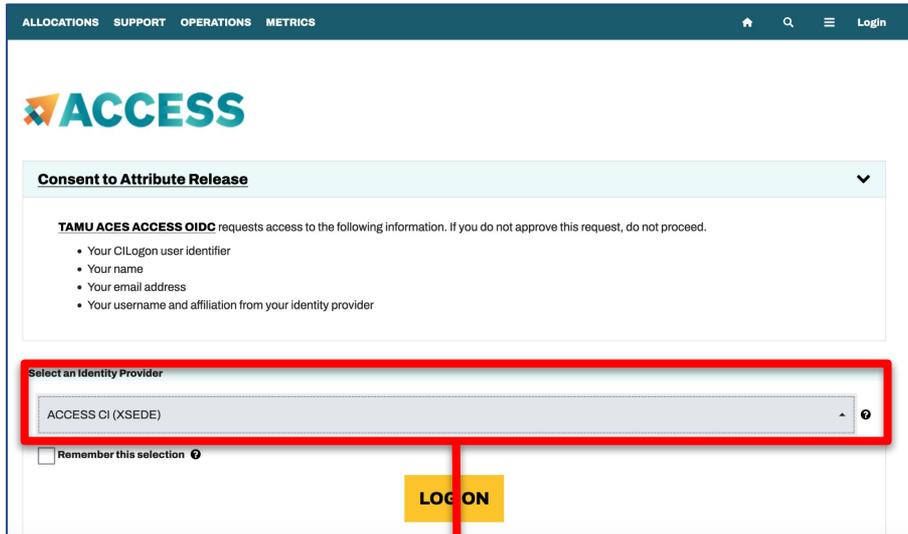


ACES Portal portal-aces.hprc.tamu.edu is the web-based user interface for the ACES cluster

Open OnDemand (OOD) is an advanced web-based graphical interface framework for HPC users

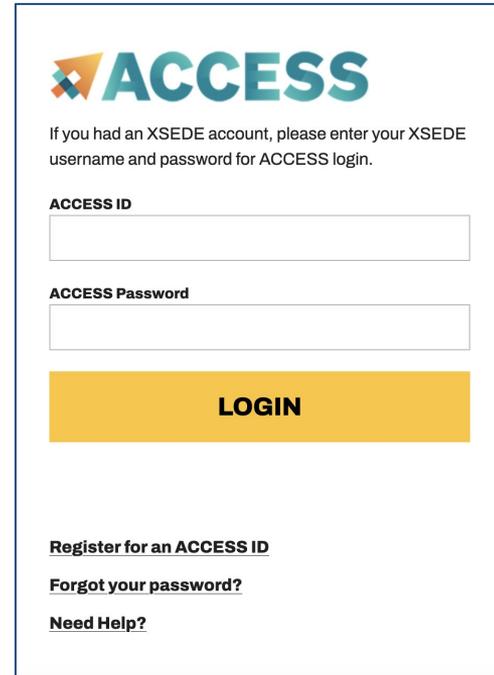


Accessing ACES via the Portal (ACCESS)



The screenshot shows the ACCESS portal interface. At the top, there is a navigation bar with links for ALLOCATIONS, SUPPORT, OPERATIONS, and METRICS, along with a search icon and a 'Login' link. Below the navigation bar is the ACCESS logo. A 'Consent to Attribute Release' section is visible, with a dropdown arrow. The consent text states: 'TAMU ACES ACCESS OIDC requests access to the following information. If you do not approve this request, do not proceed.' The list of requested information includes: 'Your CILogon user identifier', 'Your name', 'Your email address', and 'Your username and affiliation from your identity provider'. Below the consent section is a 'Select an Identity Provider' section. A dropdown menu is open, showing 'ACCESS CI (XSEDE)' as the selected option. A red box highlights this dropdown menu. Below the dropdown is a checkbox labeled 'Remember this selection' with an information icon. A yellow 'LOG ON' button is positioned below the dropdown menu, with a red line pointing from the dropdown to the button.

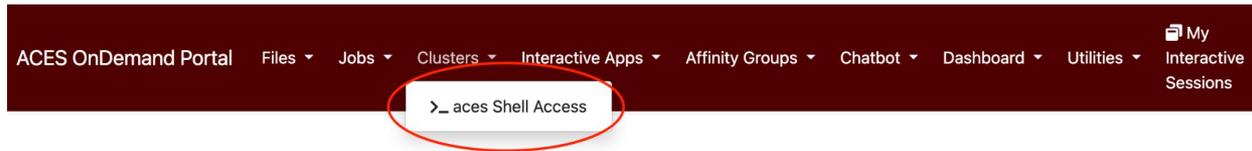
Select the Identity Provider appropriate for your account.



The screenshot shows the ACCESS portal login form. At the top, there is the ACCESS logo. Below the logo is the text: 'If you had an XSEDE account, please enter your XSEDE username and password for ACCESS login.' There are two input fields: 'ACCESS ID' and 'ACCESS Password'. Below the input fields is a yellow 'LOGIN' button. At the bottom of the form, there are three links: 'Register for an ACCESS ID', 'Forgot your password?', and 'Need Help?'.

Log-in using your ACCESS or institutional credentials.

Setting up



OnDemand provides an integrated, single access point for all of your HPC resources.

Message of the Day

IMPORTANT POLICY INFORMATION

- **Unauthorized use of HPRC resources is prohibited and subject to criminal prosecution.**
- **Use of HPRC resources in violation of United States export control laws and regulations is prohibited.**
- **Sharing HPRC account and password information is in violation of State Law. Any shared accounts will be DISABLED.**
- **Authorized users must also adhere to ALL policies at: <https://hprc.tamu.edu/policies>**

!! WARNING: THERE ARE ONLY NIGHTLY BACKUPS OF USER HOME DIRECTORIES. !!



Setting up

In your terminal, type:

```
bash /scratch/training/jupyter_ai_intro/setup.sh
```

This will put the notebook `jupyter_ai_intro.ipynb` in your `$SCRATCH` directory.



Starting Jupyter AI Assistant

The screenshot shows the ACES OnDemand Portal interface. At the top, a dark red navigation bar contains the text "ACES OnDemand Portal" and several menu items: "Files", "Jobs", "Clusters", "Interactive Apps", "Affinity Groups", "Chatbot", "Dashboard", "Utilities", "My Interactive Sessions", "Develop", "Help", and a user profile icon. The "Interactive Apps" dropdown menu is open, displaying a list of applications categorized into "GUI", "Imaging", and "Servers". The "Jupyter AI Assistant" option is circled in red. Other options in the "Servers" category include "Jupyter Notebook", "JupyterLab", "RStudio", "TensorBoard", and "Tutorials OnDemand". Below the navigation bar, the main content area features the ACES logo (Accelerating Computing for Emerging Sciences), a "Message of the Day" section, and an "IMPORTANT POLICY INFORMATION" section with a list of rules. At the bottom left, it says "powered by OPEN OnDemand" and at the bottom right, "OnDemand version: 3.1.10".

ACES OnDemand Portal Files Jobs Clusters Interactive Apps Affinity Groups Chatbot Dashboard Utilities My Interactive Sessions Develop Help

ACES
ACCELERATING COMPUTING
FOR EMERGING SCIENCES

OnDemand provides an integrated, secure environment for your HPC resources.

Message of the Day

IMPORTANT POLICY INFORMATION

- Unauthorized use of HPRC resources for non-research purposes is prohibited.
- Use of HPRC resources in violation of applicable laws and regulations is prohibited.
- Sharing HPRC account and passwords is prohibited.
- Authorized users must also adhere to applicable laws and regulations.

!! WARNING: THERE ARE ONLY NIGHTLY BUILD IMAGES AVAILABLE FOR THESE APPLICATIONS. !!

powered by OPEN OnDemand

OnDemand version: 3.1.10



Starting Jupyter AI Assistant

Home / My Interactive Sessions / Jupyter AI Assistant

Interactive Apps

- GUI
- MATLAB
- NextSilicon VNC
- VNC
- Imaging
- CryoSPARC
- CryoSPARC 4.2.1
- ImageJ
- Jmol
- Paraview
- cisTEM
- Servers
- Jupyter AI Assistant**
- Jupyter Notebook
- JupyterLab
- RStudio

Jupyter AI Assistant version: 5c10a8f

This app will launch a [JupyterLab](#) server on the [ACES cluster](#) with the HPRC custom jupyter ai extension.

Module

Python/3.11.3 (foss/2023a)

Optional python environment to be activated

Enter the name of the python virtual environment to be activated. (Optional)

The default virtualenv for Python versions have jupyterlmod which enables loading lmod software modules.

Leave blank to use the default environment for the selected Module.

Your optional python virtual environment must have been previously built with one of the Python modules listed in the Module option above. See instructions.

Select Path

Node type

CPU only

Python/3.11.3 (foss/2023a)

Resources

- TensorBoard
- Tutorials OnDemand
- TEST_HPRC_ONLY
- Hugging (Inter)Face

Interactive Apps [Sandbox]

- Servers
- Jupyter AI Assistant
- inference

Number of hours (max 168)

1

Number of cores (max 96)

1

Total GB memory (max 488)

5

Account

This field is optional.

Email

This field is optional.

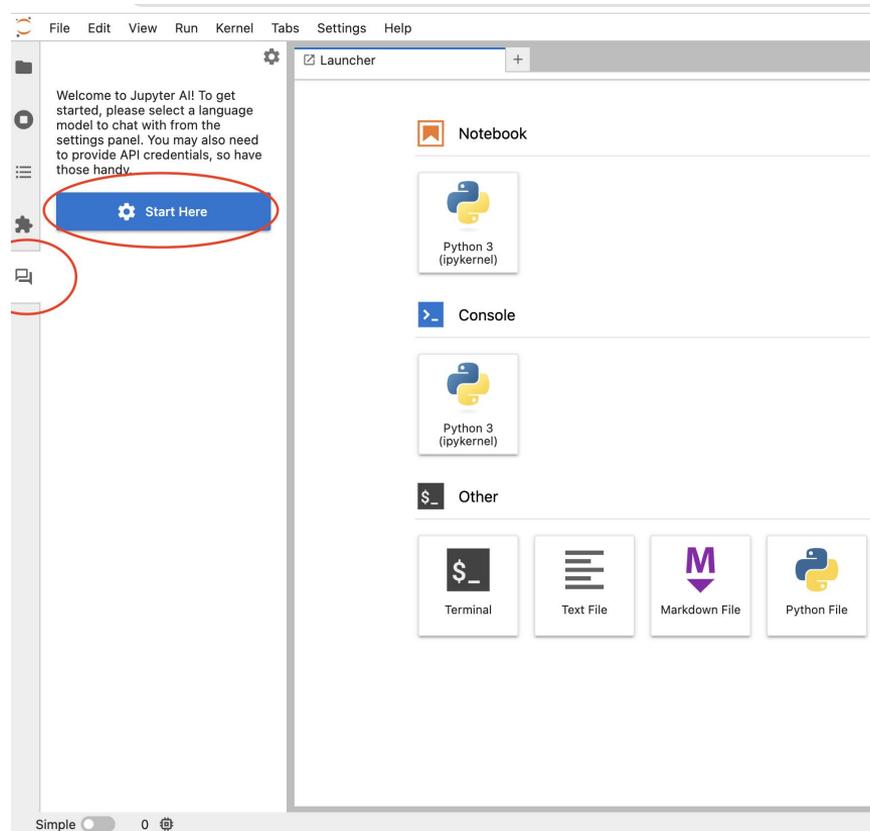
I would like to receive an email when the session starts

Launch

* The Jupyter AI Assistant session data for this session can be accessed under the [data root directory](#).

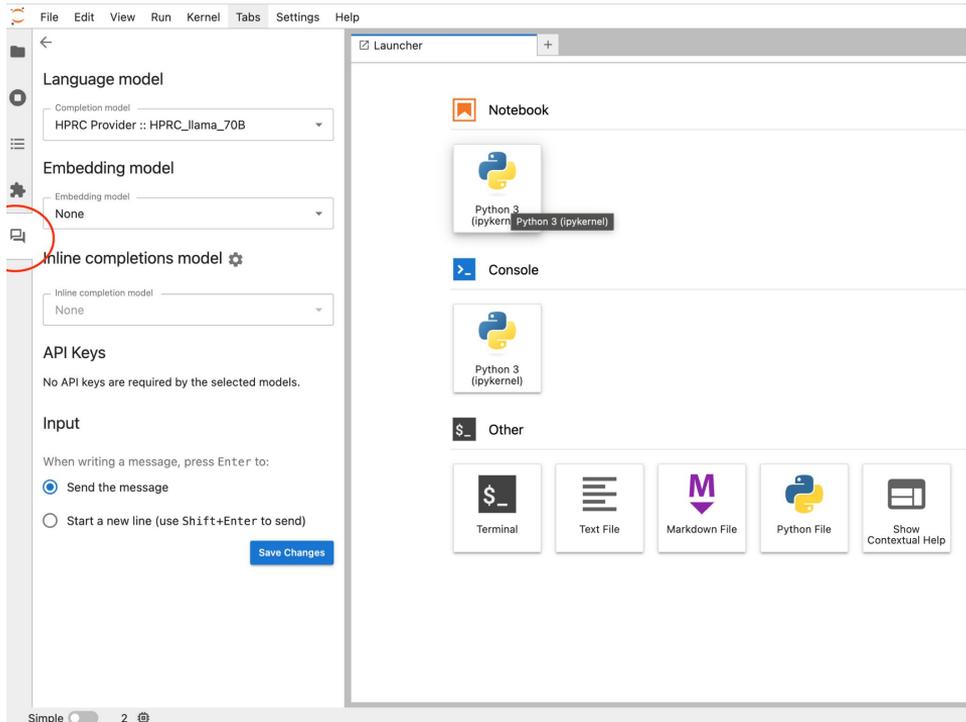


Using Jupyter AI Assistant

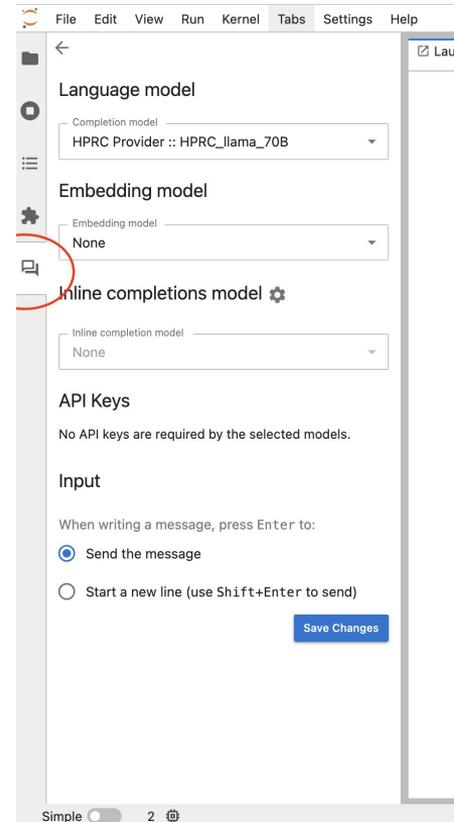


Using Jupyter AI Assistant

Click on the message icon on the left side of the screen



The screenshot shows the Jupyter AI Assistant interface. On the left sidebar, there are several settings sections: 'Language model' (Completion model: HPRC Provider :: HPRC_llama_70B), 'Embedding model' (Embedding model: None), 'Inline completions model' (Inline completion model: None), and 'API Keys' (No API keys are required by the selected models). Below these is the 'Input' section with two radio button options: 'Send the message' (selected) and 'Start a new line (use Shift+Enter to send)'. A 'Save Changes' button is at the bottom of the sidebar. The main area shows a 'Launcher' with options for 'Notebook', 'Console', and 'Other' (Terminal, Text File, Markdown File, Python File, Show Contextual Help). A red circle highlights the message icon in the sidebar.



The screenshot shows the Jupyter AI Assistant interface with the right sidebar open. The sidebar contains the following sections: 'Language model' (Completion model: HPRC Provider :: HPRC_llama_70B), 'Embedding model' (Embedding model: None), 'Inline completions model' (Inline completion model: None), 'API Keys' (No API keys are required by the selected models), and 'Input' (When writing a message, press Enter to: 'Send the message' selected, 'Start a new line (use Shift+Enter to send)'). A 'Save Changes' button is at the bottom of the sidebar. A red circle highlights the message icon in the sidebar.



Using Jupyter AI Assistant

Select the “HPRC Provider :: HPRC_llama_70B”
Completion model from
the dropdown

Click, “Save Changes”

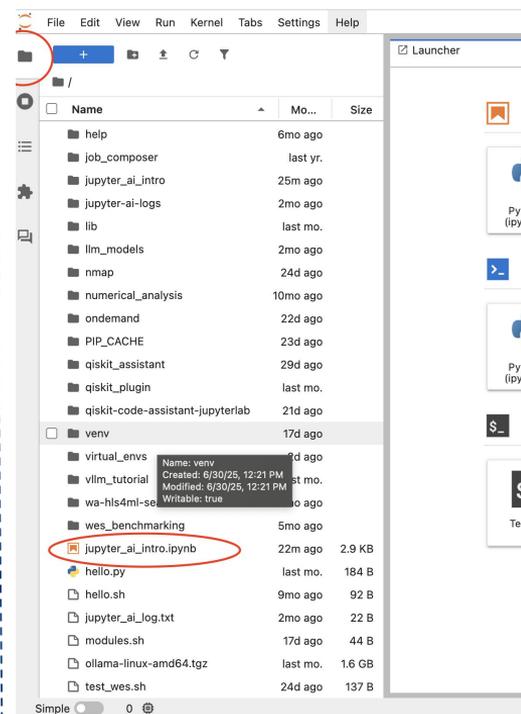
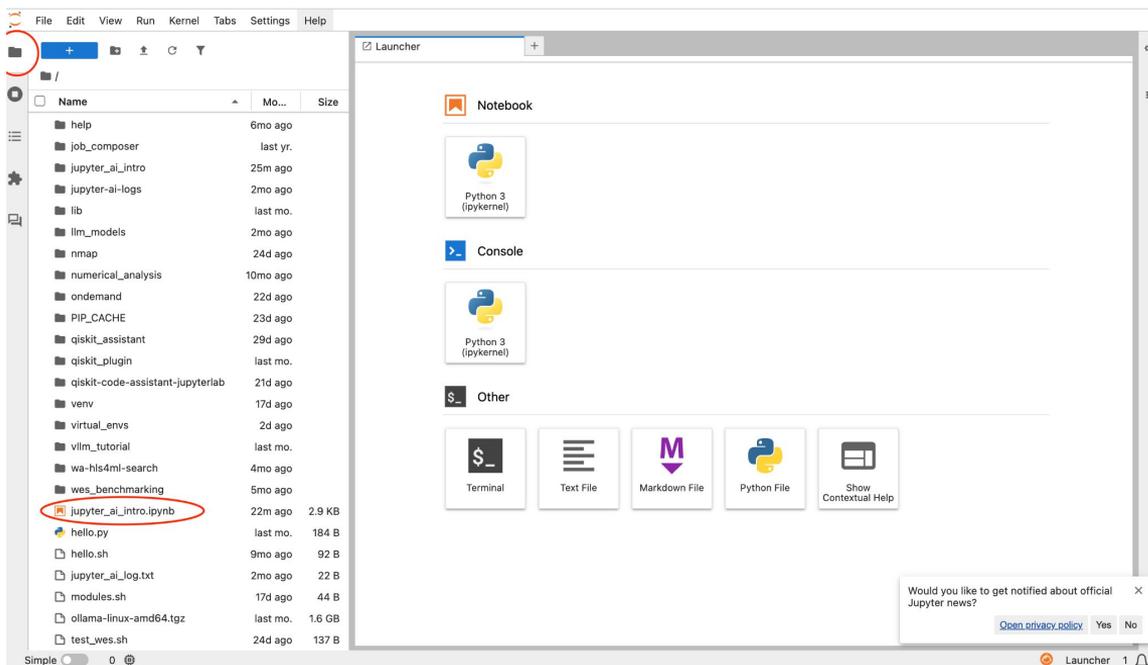
Click the back arrow on
the top left to open the
chat window

The screenshot shows the Jupyter AI Assistant settings panel. The 'Completion model' dropdown is set to 'HPRC Provider :: HPRC_llama_70B' and is circled in red. The 'Embedding model' is set to 'None'. The 'Inline completions model' is also set to 'None'. Under 'API Keys', it states 'No API keys are required by the selected models.' Under 'Input', there are two radio buttons: 'Send the message' (selected) and 'Start a new line (use Shift+Enter to send)'. A blue 'Save Changes' button is circled in red at the bottom right of the settings panel. The background shows the Jupyter interface with a 'Launcher' tab and options for Notebook, Console, and Terminal.



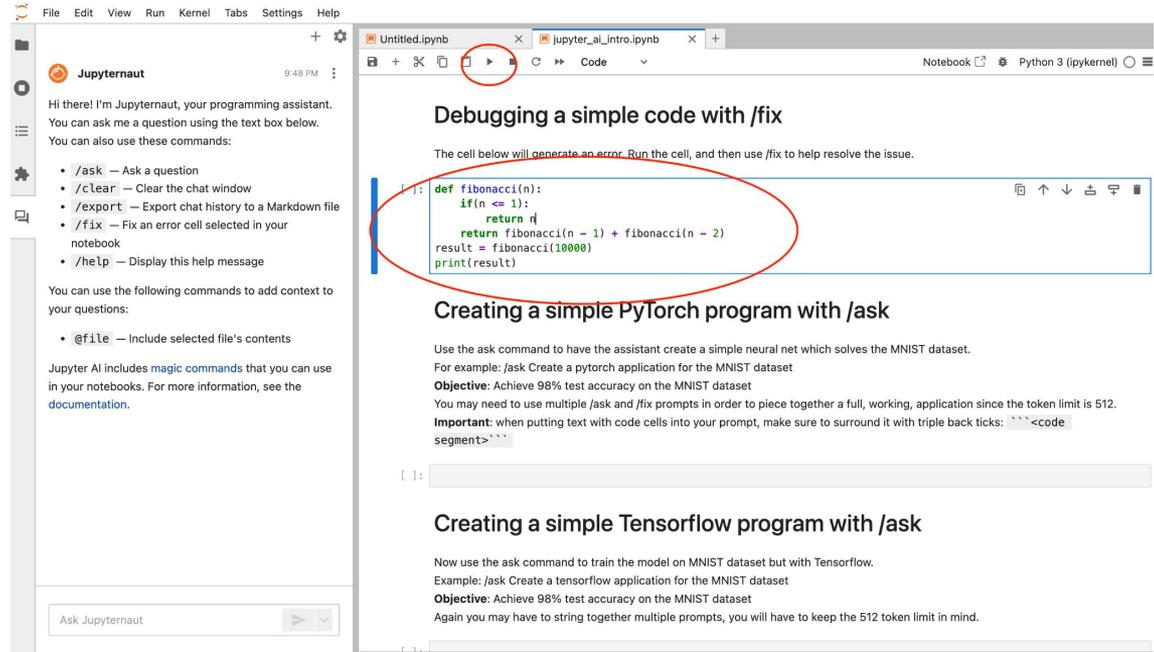
Main Content

Open the “jupyter_ai_intro.ipynb” notebook in your jupyter ai assistant session:



Debugging a simple program

- Click on the highlighted code cell
- Run the code cell with the highlighted button at the top of the screen
- Use the `/fix` command in the chat window to help debug the issue



The screenshot displays the Jupyter AI interface. On the left is a chat window with the Jupyter AI assistant's name and a list of commands: `/ask`, `/clear`, `/export`, `/fix`, and `/help`. The main area shows a Jupyter notebook with a code cell containing a Python function for calculating Fibonacci numbers. The code is:

```
def fibonacci(n):  
    if(n <= 1):  
        return 1  
    return fibonacci(n - 1) + fibonacci(n - 2)  
result = fibonacci(10000)  
print(result)
```

 The code cell is highlighted with a blue border, and a red circle highlights the run button (a play icon) at the top of the cell. Below the code cell, there are sections titled "Creating a simple PyTorch program with /ask" and "Creating a simple Tensorflow program with /ask", each with an example prompt and objective.



Creating a simple PyTorch program

- Use the `/ask` command to help develop code for solving the MNIST dataset with pytorch

The screenshot shows the Jupyter AI interface. On the left is a chat window titled 'Jupyter AI' with a list of commands: `/ask` (Ask a question), `/clear` (Clear the chat window), `/export` (Export chat history to a Markdown file), `/fix` (Fix an error cell selected in your notebook), `/help` (Display this help message), and `@file` (Include selected file's contents). The main notebook area on the right contains the following text:

Creating a simple PyTorch program with `/ask`

Use the `ask` command to have the assistant create a simple neural net which solves the MNIST dataset.
For example: `/ask Create a pytorch application for the MNIST dataset`
Objective: Achieve 98% test accuracy on the MNIST dataset
You may need to use multiple `/ask` and `/fix` prompts in order to piece together a full, working, application since the token limit is 512.
Important: when putting text with code cells into your prompt, make sure to surround it with triple back ticks: ````<code segment>````

[]:

Creating a simple Tensorflow program with `/ask`

Now use the `ask` command to train the model on MNIST dataset but with Tensorflow.
Example: `/ask Create a tensorflow application for the MNIST dataset`
Objective: Achieve 98% test accuracy on the MNIST dataset
Again you may have to string together multiple prompts, you will have to keep the 512 token limit in mind.

[]:

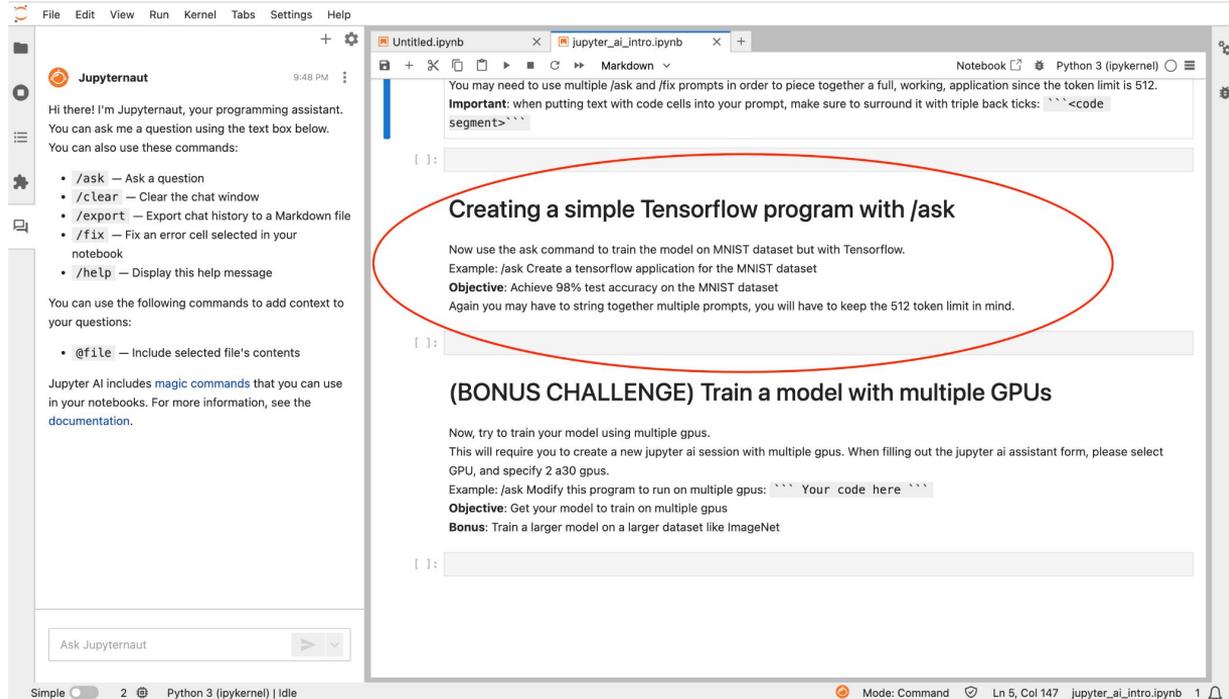
(BONUS CHALLENGE) Train a model with multiple GPUs

Now, try to train your model using multiple gpus.
This will require you to create a new jupyter ai session with multiple gpus. When filling out the jupyter ai assistant form, please select GPU, and specify 2 a30 gpus.
Example: `/ask Modify this program to run on multiple gpus: ``` Your code here ````
Objective: Get your model to train on multiple gpus
Bonus: Train a larger model on a larger dataset like ImageNet



Creating a simple Tensorflow program

- Use the `/ask` command to help develop code for solving the MNIST dataset with Tensorflow

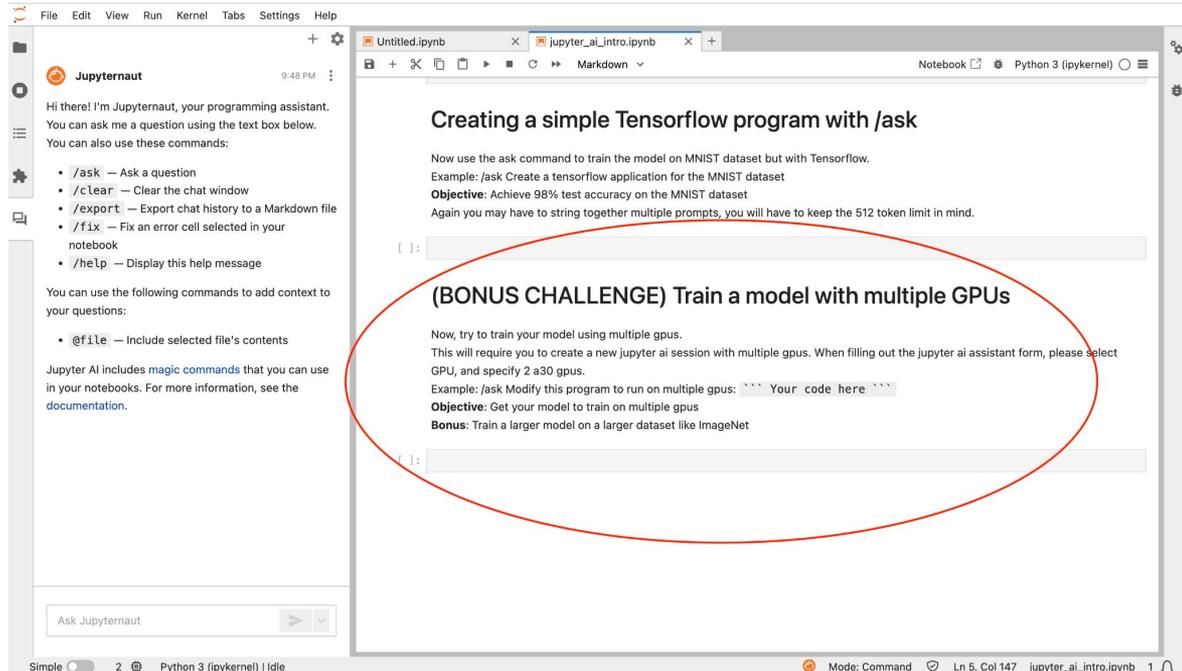


The screenshot shows a Jupyter Notebook interface with two main panes. The left pane is a chat window titled "JupyterNaut" with a timestamp of 9:48 PM. It contains a greeting and a list of commands: `/ask` (Ask a question), `/clear` (Clear the chat window), `/export` (Export chat history to a Markdown file), `/fix` (Fix an error cell selected in your notebook), `/help` (Display this help message), and `@file` (Include selected file's contents). The right pane is a code editor for a notebook titled "jupyter_ai_intro.ipynb". It contains a Markdown cell with the following text: "You may need to use multiple `/ask` and `/fix` prompts in order to piece together a full, working, application since the token limit is 512. **Important:** when putting text with code cells into your prompt, make sure to surround it with triple back ticks: ````<code>segment>````". Below this is a code cell containing the text: "Creating a simple Tensorflow program with `/ask`". This code cell is circled in red. The code cell contains the following text: "Now use the ask command to train the model on MNIST dataset but with Tensorflow. Example: `/ask` Create a tensorflow application for the MNIST dataset **Objective:** Achieve 98% test accuracy on the MNIST dataset Again you may have to string together multiple prompts, you will have to keep the 512 token limit in mind." Below this is another code cell containing the text: "(BONUS CHALLENGE) Train a model with multiple GPUs". This code cell contains the following text: "Now, try to train your model using multiple gpus. This will require you to create a new jupyter ai session with multiple gpus. When filling out the jupyter ai assistant form, please select GPU, and specify 2 a30 gpus. Example: `/ask` Modify this program to run on multiple gpus: ```` Your code here ```` **Objective:** Get your model to train on multiple gpus **Bonus:** Train a larger model on a larger dataset like ImageNet". The bottom status bar shows "Simple", "2", "Python 3 (ipykernel) | Idle", "Mode: Command", "Ln 5, Col 147", "jupyter_ai_intro.ipynb", and a notification icon.



Bonus Challenge

- Create a new Jupyter AI Assistant session with 2 A30 GPUs
- Use `/ask` and `/fix` to help develop code which uses multiple gpus to train a large model on the ImageNet dataset



The screenshot shows the Jupyter AI Assistant interface. On the left is a sidebar with a chat window containing instructions and a list of commands: `/ask`, `/clear`, `/export`, `/fix`, and `/help`. The main area displays a notebook titled "Creating a simple Tensorflow program with /ask". The notebook content includes an example prompt: `/ask Create a tensorflow application for the MNIST dataset` and an objective: "Achieve 98% test accuracy on the MNIST dataset". Below this, a red oval highlights a section titled "(BONUS CHALLENGE) Train a model with multiple GPUs". This section contains instructions to train a model using multiple GPUs, an example prompt: `/ask Modify this program to run on multiple gpus: ``` Your code here ````, and an objective: "Get your model to train on multiple gpus". A bonus note suggests training a larger model on a larger dataset like ImageNet. The interface also shows a command input field at the bottom left and a status bar at the bottom right indicating "Mode: Command" and "Ln 5, Col 147".



Technical Challenges

- Running LLM inference server on Intel PVC requires a full exclusive node
 - OneCCL does not play well with slurm, see <https://community.intel.com/t5/oneAPI-Registration-Download/ccl-issue-with-Multi-GPU-AI-Training-Data-Parallel-with-Intel/td-p/1609982>
- vLLM on xpu device would sometimes hang indefinitely when queuing prompts
 - Workaround: design an in-take web server to distribute prompts onto the backend LLM inference servers



Acknowledgements

This work was supported by

- the National Science Foundation (NSF), award numbers:
 - 2112356 - ACES - Accelerating Computing for Emerging Sciences
 - 1925764 - SWEETER - SouthWest Expertise in Expanding, Training, Education and Research
 - 2019129 - FASTER - Fostering Accelerated Scientific Transformations, Education, and Research
 - 2411377 - GOODLUCK - Growing Open OnDemand: Leveraging Unified Community Knowledge
- Staff and students at Texas A&M High-Performance Research Computing.
- ACCESS CCEP pilot program, Tier-II



Need Help?

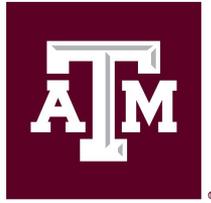
First check the [FAQ](#)

- [Knowledge Base](#)
- Send us a ticket using the dashboard tab on our [web portal](#)
- Email further questions to help@hprc.tamu.edu

Help us help you -- when you contact us, tell us:

- Which cluster you're using
- Your username
- Job id(s) if any
- Location of your jobfile, input/output files
- Application used, if any
- Module(s) loaded, if any
- Error messages
- Steps you have taken, so we can reproduce the problem





High Performance
Research Computing
DIVISION OF RESEARCH

<https://hprc.tamu.edu>

HPRC Helpdesk:

help@hprc.tamu.edu

Phone: 979-845-0219

Take our short course survey!



https://u.tamu.edu/hprc_shortcourse_survey

HPRC Survey

https://u.tamu.edu/hprc_shortcourse_survey

Thank you! Any Questions?





High Performance Research Computing

DIVISION OF RESEARCH

<https://hprc.tamu.edu>

HPRC Helpdesk:

help@hprc.tamu.edu

Phone: 979-845-0219

Help us help you. Please include details in your request for support, such as, Cluster (ACES, Faster, Grace, Launch), NetID ACCESS ID or Username, Job information (Job ID(s), Location of your jobfile, input/output files, Application, Module(s) loaded, Error messages, etc), and Steps you have taken, so we can reproduce the problem.

