The Einstein Toolkit Tutorial Server

Steven R. Brandt
Center for Computation and Technology
Louisiana State University
Baton Rouge, LA, USA
0000-0002-7979-2906

Roland Haas
National Center for Supercomputing Applications
University of Illinois
Urbana, IL, USA
0000-0003-1424-6178

***Abstract*—The Einstein Toolkit is a complex software system for numerical general relativity, a science domain that includes colliding black holes, neutron stars, supernovae, etc.**

As might be expected for a framework of this size and age (parts of it are over 20 years old), there is a significant learning curve to building it, running it, writing new modules for it, etc. Over the years, the Einstein Toolkit maintainers have given a number of tutorials for new users.

In recent years, we have created a tutorial server which allows us to streamline the teaching/learning process through the use of Jupyter notebooks and docker images. In this paper we describe the special considerations and adaptations required by the image and the notebook server that enable us to (1) easily make logins and manage accounts which streamlines both the classroom and the independent study experiences, (2) create a simplified but natural user experience for compiling and developing a complex C++ application, (3) scale to increasing class sizes.

I. Introduction

The Einstein Toolkit [1] is a community-driven software platform of core computational tools to advance and support research in relativistic astrophysics and gravitational physics. The Einstein Toolkit comprises several larger sub- projects which include a self-contained 1D self-force code, two large scale numerical astrophysics codes based on the Cactus Computational Toolkit (based on the traditional "Carpet" and the new and experimental "CarpetX" driver), and the Kuibit analysis and visualization framework. The suite of numerical astrophysics codes has historically been the major focus of the toolkit and remains one of the major areas of activity and the most commonly used component.

Typically, the Einstein Toolkit has two workshops each year (one in the US and one in Europe) in which training is offered for students or faculty that have not used the toolkit before. Basic lessons in compiling, running, and visualizing, as well as lessons in writing modules for the toolkit are provided. In addition to these official training events, the toolkit provides a "Tutorial Server" which students can access year-round.

Over the years, this system has evolved to become a flexible installation that can run in one of three ways: as a standalone Jupyter notebook for offline usage, as a JupyterHub configuration that uses CILogon for authentication for use by the Tutorial Server, and as a JupyterHub that uses our "Create-Your-Own-Login" server for workshops or classroom settings (i.e. the Workshop Server). This latter option has been recently upgraded to run inside a Docker Swarm to accommodate the larger class sizes we have experienced since the COVID-19 pandemic (the virtual workshops attracted larger numbers of students).

We will describe each configuration in detail.

II. The Standalone Server

Currently, the tutorial server is stored at https://github.com/einsteintoolkit/jupyter-et in the tutorial-server directory. For convenience, it is organized into a set of docker images that build upon each other.

The `base.docker` file is the lowest level of the docker files, installing all the required packages for compiling and running the toolkit.

In addition, it creates a precompiled build of the toolkit itself which users can copy into their home directories, thereby streamlining the compilation and installation phases. This is the most computationally expensive docker build. By factoring out this layer of the image, we separate the *science* layer from the *gateway* layers of the infrastructure.

The `notebook.docker` file adds the basic commands needed to launch a notebook.

The `start-notebook.sh` script sets the secret token for authenticating to the notebook and prints it to the console. Students can run this image on their laptop or desktop computers without need to create a login to the official tutorial server.

III. The CILogon Server (The Tutorial Server)

CILogon [2] is a federated identity and access management platform that allows researchers to access cyberinfrastructure using their home institutions' credentials. This provides advantages to both users and service providers. For users it avoids the need to create a dedicated username-password pair and provides them will a well designed, familiar login interface. If the home institutions uses dual factor logins, it also adds a level of security beyond plaintext passwords. For service providers, CILogon provides a standardized interface to all

identity providers which include universities and also ORCID. Institutional accounts provide a higher level of trust in the user's identity, avoiding creation of accounts on our tutorial server for anonymous users. ORCID, while only providing basic identity verification, nevertheless allows for manual vetting of accounts by inspecting publication records and possible institutional affiliations.

The CILogon plugin was configured to make it as easy as possible to administer user accounts on the Tutorial Server. While CILogon is helpful for managing login credentials, we did not wish to allow access to just anyone that CILogon can authenticate (which includes anonymous gmail addresses), so we built a system to verify and approve CILogon accounts.

In the first step, users attempt to login to our server and get a custom error page with a link to a PHP form on our main website (https://einsteintoolkit.org). This form has the user's name, email, and institution email prefilled with the correct values (as CILogon understands them). The user is then asked to fill out a short essay question describing why they want access to our server.

The PHP form then notifies the Einstein Toolkit maintainers of the pending account request via email to a dedicated mailing list that is monitored by the maintainers. This email contains all information provided in the PHP form. Maintainers then vet the request and, if needed, contact the user via email to obtain more information.

To enable the login, any of our maintainers can add an entry to a text file served by our main website, which our tutorial server checks during login. Each entry is a base64 encoding of an md5 encoding of a lower-case version of the user's email address. If this entry is present in the list on our main server, and the CILogon credentials validate, the user is authorized and an account is created for the user upon login.

This system enables any of our maintainers, including those who can't login to the physical machine hosting the tutorial server, to grant access to new users.

Typically, the Tutorial Server only needs to provide resources for two or three concurrent users (at most).

IV. The Create-Your-Own-Login Server (The Workshop Server)

For a classroom setting, it is not practical to use the CILogon system and validate a classroom full of new users (though we have tried). For a typical class of 30 to 40 users, this would use up about a half an hour of class time.

Another procedure we have attempted is the tried-and-true method of handing out slips of paper from a set of pregenerated usernames and passwords. Although we always stressed that the tutorials would not work if students shared accounts, they invariably did so. Even physics graduate students were unable to follow this particular instruction. Between this and problems copying the text from the paper to the screen and subsequent requests for help we, again, invariably lost about half an hour of each class time to the process of setting up accounts.

Finally, we developed the Create-Your-Own-Login authenticator (CYOL) [3] to provide access in a different way. Users create both a login and a password, and (in addition) must provide a special code that is only available in the classroom (written on a whiteboard or chalkboard at the front of the room). This code can (and should) be changed every day of class. The CYOL authenticator was previously described here [4].

This mechanism allows us to onboard an entire room of students in parallel. Typically, everyone gets logged in within 5 minutes or so.

In response to increased classroom sizes (in part, due to virtual workshops conducted during the COVID-19 pandemic), we found the need to expand our server for workshops. Our local hardware supports 48 cores and 256 GB of ram, but it has proved inadequate for classes that have sometimes been as large as 60 students.

Our new system runs on four nodes of our local hardware (but could be configured to use more) which we make accessible with Docker Swarm [5]. With this setup, one node serves as head node, and the other three serve as compute nodes. The source for this setup is available at `stevenrbrandt/swarm-cluster` on github (based on a fork of `rcanvil/swarm-cluster`).

V. Jupyter notebooks

The Tutorial Server and Workshop Server use Jupyter notebooks to present users with the steps to download, set up, compile the Einstein Toolkit, as well as run and analyze a simulation. Over the years we have experimented with multiple different formats before settling on Jupyter notebooks, including, but not limited to, PDF documents with copy & paste parts, plain HTML pages, and life-coding, Software-Carpentry style [6] presentations. PDF, while allowing for mixing text, images and math, regularly lead to issues when code text could not be copy & pasted due to the authoring software replacing input characters with typographical alternatives, for example replacing two dashes "--" with a long-dash "–". HTML on is severely limited in mixing math and typically cannot be delivered as a single file. Neither one allows for output of user input to be dynamically revealed as the users progress in their tutorial, resulting in output that does not quite match their input, e.g., showing a different user name or directory location. Finally any life-coding attempts invariably run into issues with students falling behind the displayed material or mistyping commands shown by the instructor. These issues typically require at least an extra person present to serve as a helper and even then lead to frustration on the part of the users who eventually "give up" on the tutorial.

Jupyter notebooks present an elegant solution to these issues since they allow mixing of command cells with textual and graphical instruction cells. Most of the commands entered by students are shell commands, and while Jupyter supports `bash` kernels, we require `Python` kernels to display results of plots and rendered images. Our notebooks thus make heavy use of `%%bash` cell magic to execute shell commands. In order to simulate a life terminal we customized `%%bash` to not buffer output and added custom Jupyter code [7] to follow output of long running commands. Instead of relying on external

editors we use Jupyter's `%%writefile` magic to create and modify input parameter files for simulations. A small amount of precomputed simulation result data is (lossy) encoded in the notebook and is used to plot student's results over the expected results thus providing better and more direct feedback than static plot would provide.

Together these changes let students create and explore a simulation inside of the Jupyter notebooks, while using the same commands used on high-performance clusters in a shell terminal. The notebooks also serve as a handy reference for common tasks and thus provide value past the end of the tutorial.

VI. Classes Held

The Tutorial Server and Workshop Server have been in use by the Einstein Toolkit community since at least 2017. Instances have been set up for approximately 2 workshops per year (one in the US, one in Europe), as well as for year-round training. Features and capabilities have been evolving over the years in our effort to streamline training of new users.

The first workshop this year was held at the Universidade de Aveiro in Portugal, June 19-23; and the second will be held at the Rochester Institute of Technology in New York, July 17-23.

The Portugal workshop was the first test of the Docker Swarm described in this paper. In previous years, the tutorials required helpers to go around the room (or chat online) and assist students in dealing with technical problems. This year, for the first time, the helpers remained idle. The system simply worked. From the standpoint of the instructors, this was the most effective configuration we have used.

Monitoring tools showed that despite having over fifty interactive users running jobs at the system, the computational resources were not overwhelmed, though some slowdown was observed. In Fig. 1 you see the CPU usage by time of day for both the head node and the most-used compute node. Each student ran jobs that used 2 cores. Since we had 144 cores in total, our CPU resources were more than adequate.

Of the 17 students to respond to the survey, *none* of the respondees had technical problems that caused them to abandon the tutorial server, and only a few (5) reported having problems of any kind.

A survey conducted after the first workshop made it clear that most students prefer lectures with embedded hands-on material (58.8%), while a substantial number prefer lectures followed by hands-on exercises (35.3%). Only a few (5.9%) did not appreciate the hands-on exercises.

We also received guidance on which tutorials were more effective, so we know which ones need tweaking for the second workshop.

Students particularly appreciated how easy it was to login (13), that no local software installation was needed (14), and the ability to scroll back and forth through the notebook cells (10).

Overall, student satisfaction was high (Fig. 2).

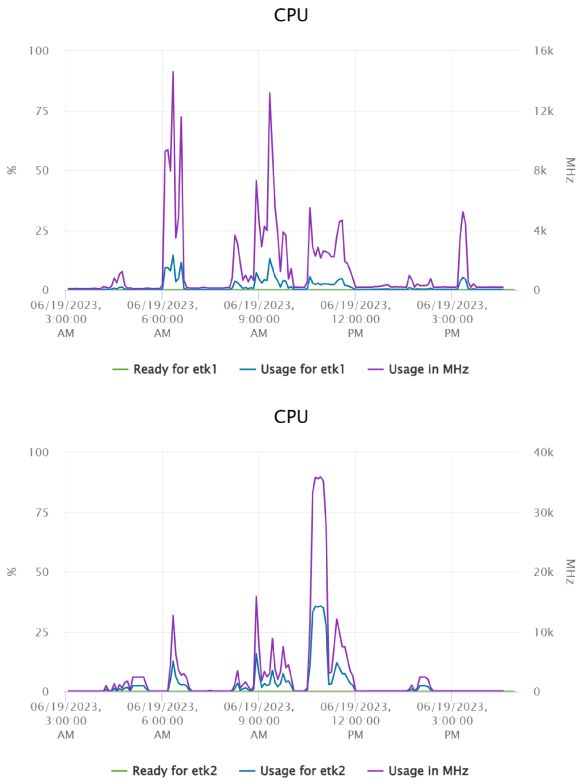

Fig. 1. The usage for the head node (etk1), and the most-used compute node (etk2).

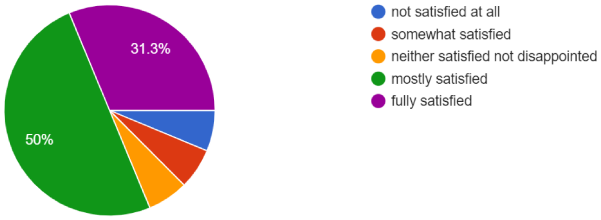

Fig. 2. The level of satisfaction of students who used the tutorial server.

VII. Load Testing

From time to time we have had technical difficulties during tutorials. Usually, these arise from heavy load. During one tutorial, a class of 40 students tried to run an OpenMPI job at the same time. It was only during this exercise that we discovered that the newly installed OpenMPI was binding each job to core zero (a new and unexpected behavior of the software compared to previous years). Even though we had an 80-core machine for this event, only one student could run a job at a time.

In another class, we had mysterious file issues that caused compiler failures for a large set of students. For some tutorials, we attempted a load test where a group of instructors all logged in to the system and tried to do things at the same time. However, we were never able to test at the scale of the classroom and therefore missed many issues.

However, the key realization was that we don't need to interact with the Jupyter GUI to test the load on the system.

We can simply run `nbconvert` in parallel for however many users we want. We can then measure correctness on the generated outputs and keep track of how long commands take to run.

Testing with `nbconvert` has identified potential problems with too many students creating and interacting with too many kernels at the same time. The ZMQ infrastructure underlying JupyterHub reuses ports and causes random failures. While we have some ideas of how to handle this problem, at the time of this writing we are unable to prevent it.

One obvious workaround for this problem would be to provide replicas of a the JupyterHub frontend. For technical reasons, we were not able to do this prior to the Portugal workshop, but the ZMQ port problem was not reported by any student.

VIII. Conclusion

The Einstein Toolkit Community has been running and maintaining a training portal or Science Gateway since approximately 2014. This Workshop Server has been refined and updated through the years to enable live, hands-on training at each of our two annual events (and some of the adhoc in-between events). The Tutorial Server has also been offered for free to any students wishing to make use of it throughout the year.

The Tutorial Server or Workshop Server typically needs to provide resources for each student to perform a significant computation (i.e. a low-resolution evolution of a Tolman-Oppenheimer-Volkoff star [8], [9]). Providing a streamlined way to enable students to come to understand the physics as well as the underlying computational infrastructure has provided an ongoing challenge.

Future work will focus on improving the scalability and reliability of this framework as well as increasing the quantity of lessons (i.e. notebooks).

IX. Acknowledgements

The authors would like to acknowledge NSF grants OAC 2004157, 2004879, 2103680, and 1238993 for financial support. They further express thanks to the Center for Computation and Technology (CCT) for the use of its virtualized hardware system and helpful conversations as well as support provided by Louisiana State University's HPC consulting services. In particular, we extend special thanks to Sai Pinnepalli at CCT and Phillip Marr at HPC.

References

[1] Leonardo Werneck, Samuel Cupp, Thiago Assumpção, Steven R. Brandt, Cheng-Hsin Cheng, Peter Diener, Jake Doherty, Zachariah Etienne, Roland Haas, Terrence Pierre Jacques, Beyhan Karakaş, Konrad Topolski, Bing-Jyun Tsao, Miguel Alcubierre, Daniela Alic, Gabrielle Allen, Marcus Ansorg, Maria Babiuc-Hamilton, Luca Baiotti, Werner Benger, Eloisa Bentivegna, Sebastiano Bernuzzi, Tanja Bode, Gabriele Bozzola, Brockton Brendal, Bernd Bruegmann, Manuela Campanelli, Federico Cipolletta, Giovanni Corvino, Roberto De Pietri, Alexandru Dima, Harry Dimmelmeier, Rion Dooley, Nils Dorband, Matthew Elley, Yaakoub El Khamra, Joshua Faber, Giuseppe Ficarra, Toni Font, Joachim Frieben, Bruno Giacomazzo, Tom Goodale, Carsten Gundlach, Ian Hawke, Scott Hawley, Ian Hinder, E. A. Huerta, Sascha Husa, Taishi Ikeda, Sai Iyer, Liwei Ji, Daniel Johnson, Abhishek V. Joshi, Hrishikesh Kalyanaraman, Anuj Kankani, Wolfgang Kastaun, Thorsten Kellermann, Andrew Knapp, Michael Koppitz, Nadine Kuo, Pablo Laguna, Gerd Lanferman, Paul Lasky, Lisa Leung, Frank Löffler, Hayley Macpherson, Joan Masso, Lars Menger, Andre Merzky, Jonah Maxwell Miller, Mark Miller, Philipp Moesta, Pedro Montero, Bruno Mundim, Patrick Nelson, Andrea Nerozzi, Scott C. Noble, Christian Ott, Ludwig Jens Papenfort, Ravi Paruchuri, Denis Pollney, Daniel Price, David Radice, Thomas Radke, Christian Reisswig, Luciano Rezzolla, Chloe B. Richards, David Rideout, Matei Ripeanu, Lorenzo Sala, Jascha A Schewtschenko, Erik Schnetter, Bernard Schutz, Ed Seidel, Eric Seidel, John Shalf, Ken Sible, Ulrich Sperhake, Nikolaos Stergioulas, Wai-Mo Suen, Bela Szilagyi, Ryoji Takahashi, Michael Thomas, Jonathan Thornburg, Chi Tian, Malcolm Tobias, Aaryn Tonita, Samuel Tootle, Paul Walker, Mew-Bing Wan, Barry Wardell, Allen Wen, Helvi Witek, Miguel Zilhão, Burkhard Zink, and Yosef Zlochower. The einstein toolkit, May 2023. To find out more, visit http://einsteintoolkit.org.

[2] Jim Basney, Heather Flanagan, Terry Fleury, Jeff Gaynor, Scott Koranda, and Benn Oshrin. Cilogon: Enabling federated identity and access management for scientific collaborations. *Proceedings of Science*, 351:031, 2019.

[3] Steven R. Brandt. Cyolauthenticator: Create your own login authenticator. https://github.com/stevenrbrandt/cyolauthenticator, 2019.

[4] Patrick Diehl and Steven R Brandt. Interactive c++ code development using c++ explorer and github classroom for educational purposes. *Concurrency and Computation: Practice and Experience*, page e6893, 2020.

[5] Docker swarm, 2014.

[6] Greg Wilson. Software carpentry: Getting scientists to write better code by making them more productive. *Computing in Science & Engineering*, November–December 2006. Summarizes the what and why of Version 3 of the course.

[7] Steven R. Brandt. scrolldown: Keep long running notebook in jupyter scrolled to the bottom. https://pypi.org/project/scrolldown/, 2019.

[8] Richard C Tolman. Effect of inhomogeneity on cosmological models. *Proceedings of the National Academy of Sciences*, 20(3):169–176, 1934.

[9] J. R. Oppenheimer and G. M. Volkoff. On massive neutron cores. *Phys. Rev.*, 55:374–381, Feb 1939.