Container Adoption in Campus High Performance Computing at Texas A&M University

Richard Lawrence
rlawrence@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Dhruva K. Chakravorty
chakravorty@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Lisa M. Perez
perez@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Wesley Brashear
wbrashear@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Zhenhua He
happidence1@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Joshua Winchell
jwinchell@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Honggao Liu
honggao@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

ABSTRACT

Containers promise benefits for both researchers and cyberinfrastructure (CI) professionals, by offering pre-assembled software stacks with a lower barrier to entry. However, the benefits of containers are best realized in a well-developed CI ecosystem with support from administrators. In this paper, we report on the emerging container ecosystem supported by Texas A&M University's High Performance Research Computing Group (HPRC)[1]; lessons learned from the perspective of a team that manages a diverse CI portfolio. Here we share our experience with the following: building a container ecosystem for HPRC computing resources, outreach to researcher communities, and documenting use cases that inspire uptake by others. HPRC container deployment has facilitated access to and support for a wide variety of research software at Texas A&M. Our progress is described from multiple perspectives that reflect the varied ways that our researchers and administrators utilize containers.

CCS CONCEPTS

• **Human-centered computing** → Visualization systems and tools; • **General and reference** → *Metrics*; • **Software and its engineering** → *Software configuration management and version control systems*; • **Security and privacy** → **Virtualization and security**; • **Social and professional topics** → **Informal education**; *Software selection and adaptation*; File systems management.

KEYWORDS

Containers, Rootless Docker, Singularity, Apptainer, Charliecloud, Podman, Slurm, Training, HPC, Cyberinfrastructure

ACM Reference Format:
Richard Lawrence, Dhruva K. Chakravorty, Lisa M. Perez, Wesley Brashear, Zhenhua He, Joshua Winchell, and Honggao Liu. 2024. Container Adoption in Campus High Performance Computing at Texas A&M University. In *Practice and Experience in Advanced Research Computing (PEARC '24), July 21–25, 2024, Providence, RI, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3626203.3670550

1 INTRODUCTION

Containers are a technology for facilitating scientific research workflows on high-performance computing (HPC) systems with several known benefits [13][16]. Container users are largely unaffected by changes to the host environment, such as relocation of the container to a different host or changes made to the host environment. This makes them attractive to researchers trying to reproduce results from others. The container user should be able to continue to use the same build of their software if the host architecture remains the same. This increases the longevity of their build and the reproducibility of any results obtained using that build [17]. Containers have the potential to increase the adoption of software applications at research computing centers [4]. Research computing centers, however, have environments that differ from "cloud" environments that were designed for container-native approaches [13]. Researchers may use containers in an *indirect* manner or *intentional* manner at these sites. Here, the life cycle of a code plays an important role. In these settings, a software maturity lifecycle

[1]High Performance Research Computing (HPRC)

PEARC '24, July 21–25, 2024, Providence, RI, USA

ACM ISBN 979-8-4007-0419-2/24/07
https://doi.org/10.1145/3626203.3670550

may be applied to classify application software as "large community codes," "mature research group code bases," "single user code," "development code," or "unused code." These applications could all be supported by containerized technologies in different ways [20]. Here, we visit various scenarios. Large community codes used by many researchers are candidates for containerization. This approach allows centers to support builds for specific architectures to ensure performance across all devices, and has been adopted by HPRC to support applications like CryoSPARC, "a state-of-the-art scientific software platform for cryo-electron microscopy used in research and drug discovery pipelines" [19]. In the second scenario, a research group may want to preserve and share its "in-house" code with select collaborators at other institutions. In this case, a center can offer researchers a means by which to share their code in a versioned manner via containers which is useful when building container native workflows. While the former approach has researchers using containers in an *indirect* manner, the latter approach is more *intentional*. These approaches are visualized in Figure 1.

Containerized applications can take on varied perspectives in a campus HPC environment. Metrics used to measure uptake include release cycles, downloads, and adoption sites [9]. HPC centers use containers in manners that extend beyond the metrics, such as the percentage of containerized applications, or the number of containers launched. It's tempting to ask "Are users independently choosing to adopt containers?" but that's a narrow view. We must also determine if it's useful for an application to be available in a containerized form. A center can certainly measure the containerization of community codes. A researcher developing ad-hoc code that will never be shared, may not use a containerized approach. Whenever their code becomes production-ready, and they decide to share their work, then it becomes relevant whether they choose containers or some other method of sharing. One must also account for how containers are used - intentionally or indirectly.

Mixed software module/container environments are becoming more common in research computing environments [24][22]. CI professionals may use containers to install software on central resources such as an HPC center. Such environments simplify a researcher's workflow by offering default applications within portals and toolchains [23]. Here a researcher may prefer to launch the default application, which could be containerized or not. In this setting, it is reasonable to ask if a researcher running an application should even be aware of the underlying technology, or have to worry about choosing a containerized or installed software environment. In such cases, what are the effective means to accurately measure container usage? In consideration of containers' role in making codes easier to share, one could count the number of research computing centers that support such a container. While this metric could inform the center about its standing among peers, it is, however, unlikely to be useful to researchers who may never need that application. A more informed approach could be to count the number of research groups adopting a technology since each is likely to have internally correlated container usage. In light of these observations, we must carefully define what it means for software to be available in a containerized form, and ask the question, "Are users benefitting?"

As we discuss these topics, it is important to understand containerization technologies typically used by researchers. Docker (Rootless), Podman (Rootless), Singularity / Apptainer, and Charliecloud are common varieties [13]. Container runtimes are designed for use on different computing resources. Singularity and Charliecloud, for example, use flat image files and single-user maps to obtain good performance on network file systems [14][18]. Other container runtimes, including Docker and Podman, lack these features because they were designed to operate in a cloud environment, which has different use cases and constraints. As such, it is anticipated that HPC-focused container runtimes will remain dominant on campus computing centers for the foreseeable future [13]. These container runtimes have been selected for use at shared campus computing centers because they can defer security issues to the Linux kernel by operating in an unprivileged (rootless) manner. From inside the container, one may self-identify as any user, including root, and this is mapped to their actual user identity outside the container [5]. The Linux user namespace feature enables this practice. It doesn't entirely eliminate the need to think about privilege, but it does give users the freedom to act as though they have whatever privilege is needed, including root access, even if they normally would not. For a large fraction of container use cases, this practice is sufficient to both build and run software within the container. As long as one is willing to trust the Linux kernel and the user namespace feature, supporting unprivileged containers adds only a negligible security risk [3]. Other Linux namespaces exist, such as the network namespace, but they are generally not needed for HPC applications.

2 MATERIALS AND METHODS

At Texas A&M HPRC, container-native workflows are supported on several computing clusters. These include the hybrid computing environment offered by the Grace cluster, and several systems supported by the National Science Foundation (NSF), including the CC* Launch cluster, the MRI FASTER hardware-composable cluster [8], and the ACES hardware-composable computing cluster [7]. All support different central processing unit (CPU) and graphics processing unit (GPU) technologies, with ACES extending support to field programmable gate array (FPGA), co-processor, intelligence processing unit (IPU), and vector engine technologies. While the Grace, Launch FASTER, and ACES computing environments use the Slurm scheduler, the ACES cluster will soon offer job orchestration via Kubernetes. Supporting applications for individual builds is a continuous challenge in such a varied environment. To ensure that researchers can use the latest center-managed software, or use software from community resources, HPRC supports Singularity and Charliecloud container runtimes by providing technical development, user interface integration, and researcher training. At the same time, rootless Docker and rootless Podman runtimes are also provided on the ACES cluster.

Singularity was initially chosen to support container workflows because at that time it was the only container runtime considered secure enough for use on shared systems while also offering good performance for HPC applications [1]. It was installed cluster-wide because at the time it required root permission to install. Newer versions of Singularity and Apptainer lack this constraint. Singularity

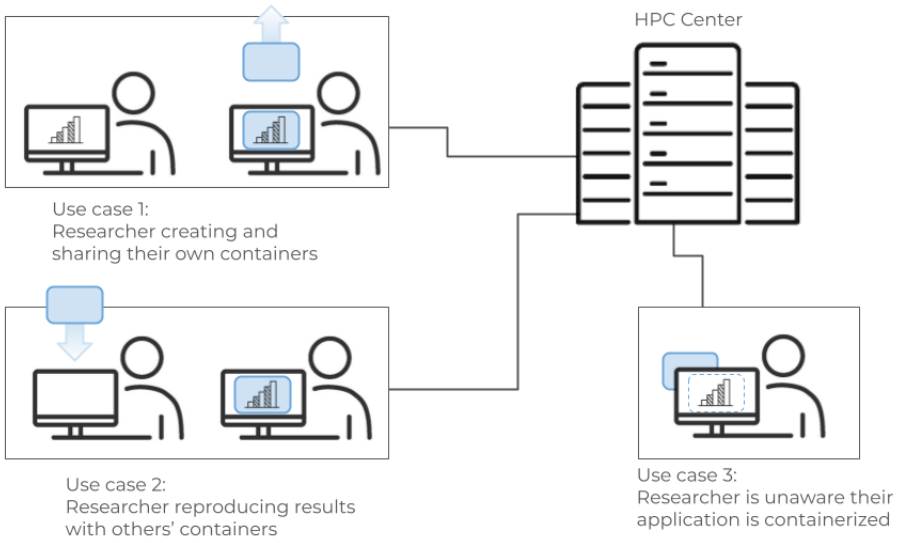

Figure 1: Diagram of three distinct container use cases emerging in HPC. Figure created using stock images from Adobe under Texas A&M's Education License, 2024.

was installed in one location, at the root level because its interface changes slowly enough that multiple version tenancy is not necessary. Usage of Singularity is tracked through Singularity's internal logs. Charliecloud, the unprivileged container runtime from Los Alamos National Lab, also meets these expectations for security and performance and was adopted more recently [18]. Charliecloud is installed in our module system because it is an actively developed project with new features coming out all the time. Thus, keeping the versions separate from each other in the module system provides a benefit to users. Lmod maintains a record of when modules are loaded to our site, which tracks Charliecloud usage.

Researchers use container runtimes to fetch up-to-date builds of scientific software from remote repositories curated by experts. For example, NVIDIA provides builds of GPU-ready software that are more up-to-date than system-installed builds and usually more performant than user-installed builds. Researchers copy and use these images whole, with little or no modification. This practice saves time for users and system administrators without sacrificing performance.

HPRC maintains several forms of support for researchers who wish to use containers, including training on the use of containers as in Table 1, container documentation in written form in our knowledge base [10] and in video form on our YouTube channel [11], and unique technical container support in our ACES web portal powered by Open OnDemand [12]. The Jupyter Notebook interactive app supports the use of user-supplied containers through both Singularity and Charliecloud engines. HPRC already supports containerized Jupyter Notebooks in the Open OnDemand framework using both Charliecloud and Singularity runtimes as shown in Figure 2.

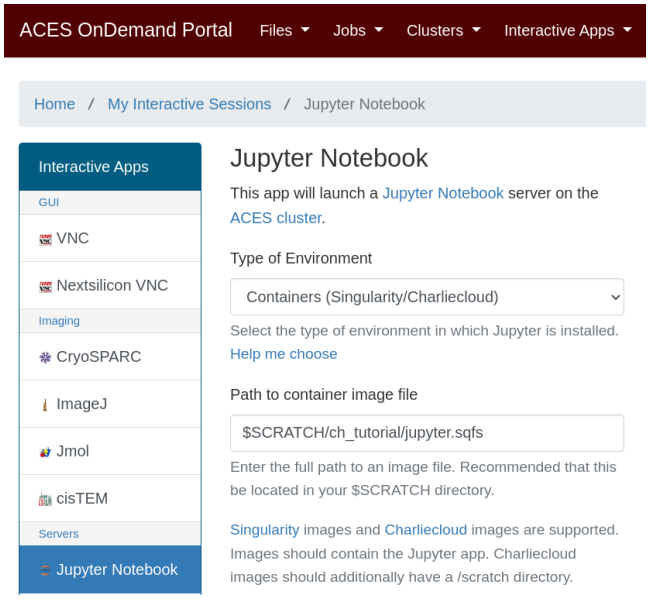

Figure 2: This screenshot was captured on ACES, and was used to train researchers who were new to containers during a September 2023 short course.

3 RESULTS

Container usage continues to grow on our systems, as shown in Figures 3 and 4. Researchers from a variety of scientific fields have adopted container use on our clusters, but most especially in the domain of biology and domains which employ machine learning

workflows. Please see Appendix A. The majority of container images seen in use on our clusters have been recognizable community codes with established container support. We see little evidence that researchers are containerizing their in-house codes.

We note that training and support for container usage are still broadly lacking across NSF ACCESS resource provider sites. Please see Appendix B. Beyond flagship sites, container adoption may be held back by insufficient support. Small sites still don't have an unprivileged container build option for users; they must either ask the system administrator to build a container image, or the user must build it themselves on another machine that they own. This could be due to the over-committed system administrators at these sites. The other half of small sites are supporting unprivileged builds using one or more variants of Singularity and/or Apptainer, and report that the ability for users to build autonomously has been game-changing for container adoption since it reduces the need for administrator support.

In an ACES workshop post-event survey, 77% of respondents who attended the Charliecloud session rated it "Very Informative."

Several HPRC courses geared towards biology make use of containerized applications, including "Introduction to Next Generation Sequencing Analysis," "Alphafold Protein Structure Prediction," CryoSPARC courses (CryoSPARC itself running in a container, for example), and courses covering NVIDIA's Parabricks software.

HPRC has recently installed rootless Docker and rootless Podman on an experimental basis and will offer courses starting in Summer 2024.

3.1 two anecdotes

In June 2023, an experienced researcher used Charliecloud to deploy their personal Dockerfile on the FASTER cluster, used the local GPUs, and interacted with a running Jupyter notebook by tunneling from their personal machine. The researcher was then able to perform machine learning tasks for research. The most difficult part of the implementation was working with the Conda environment within the container. Conda makes assumptions about the /home partition that are not necessarily true on HPC; it helps to anticipate this and manage it accordingly. Before reaching out, the researcher tried unsuccessfully to install this Conda environment on bare metal, which failed due to the limitations of the network filesystem. This anecdote showcases the importance of container runtimes which are designed to work around HPC hardware constraints.

In August 2023, a novice researcher used Charliecloud to fetch a container image for Progressive Cactus bioinformatics software [2] and modified it to suit their needs by editing one of the root-owned files in the image. This task was straightforward enough that the researcher was fully trained in the correct use of Charliecloud for this task in under an hour, and all steps were performed on a single login node. The researcher was then able to perform genomics tasks for research. Before reaching out for assistance, the researcher had unsuccessfully tried to perform these tasks using an older build of the Singularity runtime installed on an older HPC system running CentOS 7, which had disabled editing of root-owned files directly in the container image for security reasons. This anecdote showcases the importance of modern, unprivileged container runtimes.

4 DISCUSSION AND CONCLUSION

As the adoption of container technologies continues to grow, it becomes essential to assess their effectiveness and impact on researchers. The overwhelming majority of Charliecloud module loads have been one of two cases: HPRC staff testing or developing curriculum, and participants of a tutorial doing their exercises during class. A handful of users are independently loading a Charliecloud module outside of these contexts. This could mean that researchers are installing Charliecloud in their personal directories, or it could mean that researchers are using Charliecloud at other institutions after attending HPRC training, but more likely it means that Charliecloud adoption is still in its infancy.

Historically, HPC use cases have revolved around command-line interfaces and traditional applications deployed within containers. However, the Open OnDemand platform introduces interactive graphical computing, primarily managed by system administrators familiar with conventional software installations. Unfortunately, containers have been somewhat overlooked in this context. Nevertheless, cloud-based scenarios are now highlighting the advantages of interactive computing with containerized backends. To fully embrace this shift, effective graphical container management becomes essential for orchestrating the front end. As researchers adapt to these changes, we can expect a growing demand for comprehensive end-to-end graphical computing solutions [15].

The Kubernetes scheduler, which is built upon the foundation of containers, is expected to continue to grow in popularity. As it does, it will put pressure on application developers and users to containerize, which will in turn put pressure on HPC administrators to integrate container support into the HPC ecosystem.

The most important feature of a proposed container adoption metric should be that it can account for container adoption at multiple levels. Since researchers can benefit from containers that they are not even aware of, directly querying the community about their container experiences doesn't suffice. Instead, the focus should be on identifying which researchers are benefiting from container use, and how. The following metrics will help HPRC shape policies and inform documentation in the future:

- Release frequency and update frequency of versioned packages and libraries. Does container adoption deliver researchers more recent and/or more stable software?
- Deployment time; the time it takes from writing code to having it compiled and deployed in production is critical. Does container adoption accelerate design and testing processes?
- Rate of job crashes. Do containerized jobs crash less frequently than bare metal jobs?
- Satisfaction among researchers. Does container use lead to the continued use of an HPC platform?
- Ticket volume. Do container users require administrative assistance less frequently than bare metal users?
- Quantity and variety of products and services facilitated by containers. Does container adoption attract new research projects to HPC?

The direct benefits of containerization are well-known: software portability, reproducible science, and reduction of security risk. However, there are additional benefits of working to establish and support a container ecosystem. When researchers are properly

Event Title	Container Runtime	Date	Event Type
Introduction to Containers (Charliecloud) Tutorial	Charliecloud	February	Short Course /Tech Lab
Using Containers on HPRC Resources (Singularity/Apptainer)	Singularity	March	Short Course
ACES: Charliecloud Container Training	Charliecloud	July	Workshop
ACES: Introduction to Containers (Charliecloud) Tutorial	Charliecloud	September	Short Course
ACES: Fundamentals of Containers	Charliecloud/Singularity	October	Short Course
ACES: Containers for Scientific Workflows (Singularity / Apptainer)	Singularity	October	Short Course

Table 1: Training events for Containers in 2023 at Texas A&M

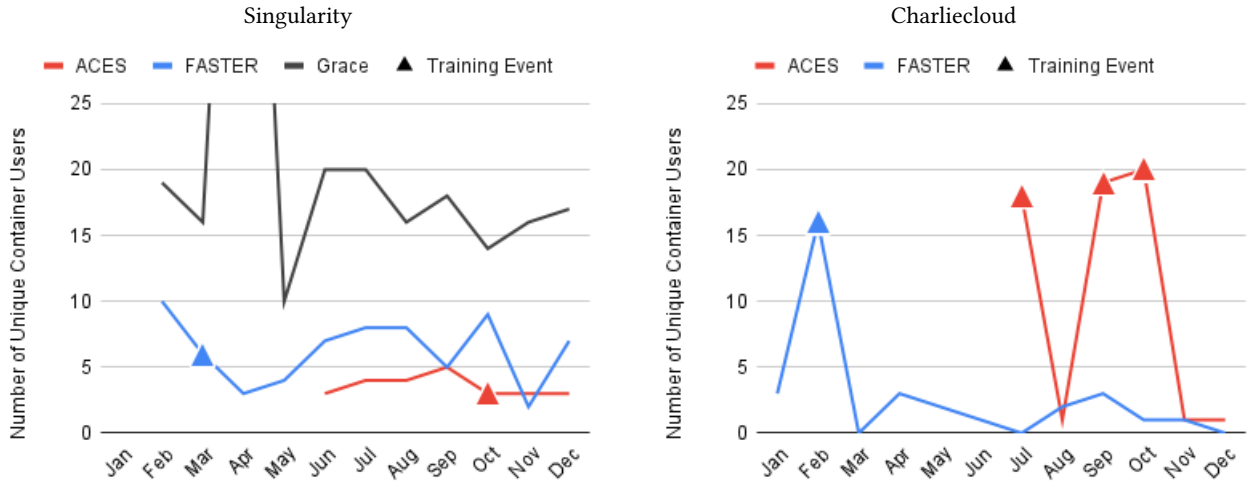

Figure 3: Number of unique container users on HPRC clusters in 2023 by month, separated by cluster (color) and container runtime (Singularity left, Charliecloud right). Training events are indicated by triangles. The large spike in April (72 users) is due to a semester coursework assignment from the biochemistry department; they were using an Alphafold image provided by HPRC.

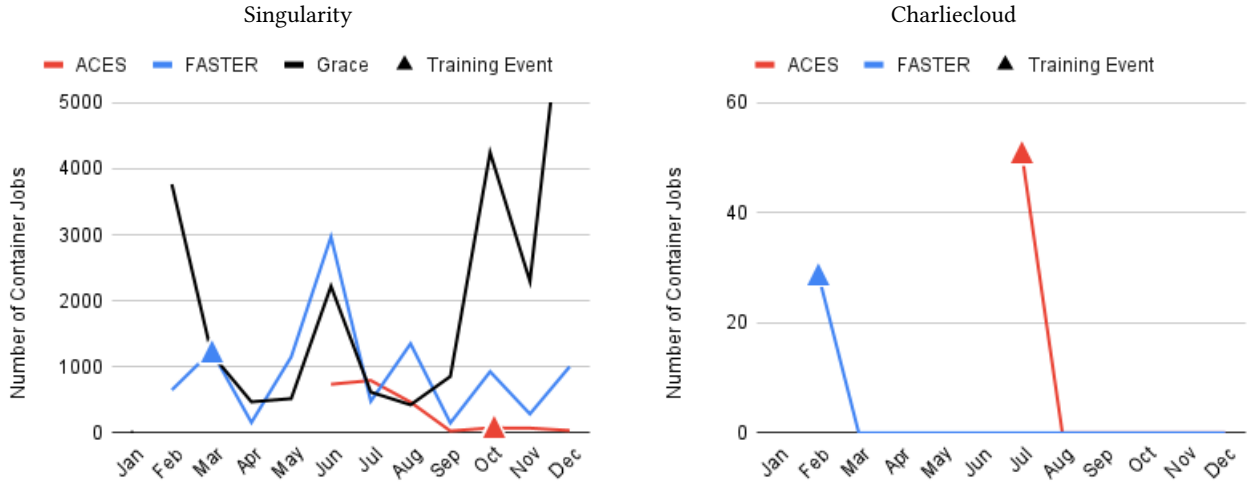

Figure 4: Number of container jobs on HPRC clusters in 2023 by month, separated by cluster (color) and container runtime (Singularity left, Charliecloud right). Training events are marked as triangles. The largest share of containerized jobs is owned by a handful of users on the Grace cluster, where Singularity is the only container runtime option.

trained in the use of modern container features, they are less reliant on administrator assistance to perform simple tasks. Administrators can provide a wider variety of research software, more rapidly. Container utilization has helped HPRC deploy scientific software, both with and without the researcher's knowledge. With this rapid growth of different container uses, the metrics need to evolve to track this moving target.

ACKNOWLEDGMENTS

This work was supported by NSF award number 2112356 ACES - Accelerating Computing for Emerging Sciences; NSF award number 2019129 FASTER - Fostering Accelerated Scientific Transformations, Education, and Research; and NSF CC* Cyberteam award number 1925764 SWEETER - SouthWest Expertise in Expanding, Training, Education and Research. We would like to thank Megan Phinney of Los Alamos National Lab, who led two of the training sessions reported on in this work. We would also like to acknowledge the following individuals who, although not directly involved in the scientific aspects of this work, played crucial roles in preparing and refining this manuscript: Elizabeth Leake, Reid Priedhorsky, and Joshua Winchell.

REFERENCES

[1] Carlos Arango Gutierrez, Remy Dernat, and John Sanabria. 2017. Performance Evaluation of Container-based Virtualization for High Performance Computing Environments. *Revista UIS Ingenierías* 18 (09 2017). https://doi.org/10.18273/revuin.v18n4-2019003

[2] J. Armstrong, G. Hickey, M Diekhans, and et al. 2020. Progressive Cactus is a multiple-genome aligner for the thousand-gen. *Nature* (2020). https://doi.org/10.1038/s41586-020-2871-y

[3] Luciano Baresi, Giovanni Quattrocchi, and Nicholas Rasi. 2024. A qualitative and quantitative analysis of container engines. *Journal of Systems and Software* 210 (2024), 111965. https://doi.org/10.1016/j.jss.2024.111965

[4] Erik Ferlanti, William J. Allen, Ernesto A. B. F. Lima, Yinzhi Wang, and John M. Fonner. 2023. Perspectives and Experiences Supporting Containers for Research Computing at the Texas Advanced Computing Center. In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis* (Denver, CO) *(SC-W '23)*. Association for Computing Machinery, New York, NY, USA, 155–164. https://doi.org/10.1145/3624062.3624587

[5] Olivier Flauzac, Fabien Mauhourat, and Florent Nolot. 2020. A review of native container security for running applications. *Procedia Computer Science* 175 (2020), 157–164. https://doi.org/10.1016/j.procs.2020.07.025 The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC),The 15th International Conference on Future Networks and Communications (FNC),The 10th International Conference on Sustainable Energy Information Technology.

[6] David Y. Hancock, Jeremy Fischer, John Michael Lowe, Winona Snapp-Childs, Marlon Pierce, Suresh Marru, J. Eric Coulter, Matthew Vaughn, Brian Beck, Nirav Merchant, Edwin Skidmore, and Gwen Jacobs. 2021. Jetstream2: Accelerating cloud computing via Jetstream. In *Practice and Experience in Advanced Research Computing* (Boston, MA, USA) *(PEARC '21)*. Association for Computing Machinery, New York, NY, USA, Article 11, 8 pages. https://doi.org/10.1145/3437359.3465565

[7] Zhenhua He, Sandra Nite, Joshua Winchell, Abhinand Nasari, Hieu Le, Jiao Tao, Dhruva Chakravorty, Lisa Perez, and Honggao Liu. 2023. Development of a Training Framework for Novel Accelerators. In *2023 IEEE Frontiers in Education Conference (FIE)*. 1–6. https://doi.org/10.1109/FIE58773.2023.10343498

[8] Zhenhua He, Aditi Saluja, Richard Lawrence, Dhruva K. Chakravorty, Francis Dang, Lisa M. Perez, and Honggao Liu. 2023. Performance of Distributed Deep Learning Workloads on a Composable Cyberinfrastructure. In *Practice and Experience in Advanced Research Computing* (Portland, OR, USA) *(PEARC '23)*. Association for Computing Machinery, New York, NY, USA, 12 pages. https://doi.org/10.1145/3569951.3603632

[9] Benjamin Holmes. 2019. Application lifecycle management for container-native development. https://developers.redhat.com/blog/2019/06/11/application-lifecycle-management-for-container-native-development. Accessed: 2024-05-16.

[10] HPRC. 2024. *Knowledge Base*. https://hprc.tamu.edu/kb/

[11] HPRC. 2024. *Youtube Channel*. https://www.youtube.com/channel/UCgeDEHE5GwkxYUGS0FDLmPw

[12] Dave Hudak, Doug Johnson, Alan Chalker, Jeremy Nicklas, Eric Franz, Trey Dockendorf, and Brian L. McMichael. 2018. Open OnDemand: A web-based client portal for HPC centers. *Journal of Open Source Software* 3, 25 (2018), 622. https://doi.org/10.21105/joss.00622

[13] R. Keller Tesser and E. Borin. 2022. Containers in HPC: a survey. *J Supercomput* (2022). https://doi.org/10.1007/s11227-022-04848-y

[14] Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. 2017. Singularity: Scientific containers for mobility of compute. *PLOS ONE* 12, 5 (05 2017), 1–20. https://doi.org/10.1371/journal.pone.0177459

[15] Richard Lawrence, Tri M. Pham, Phi T. Au, Xin Yang, Kyle Hsu, Stuti H. Trivedi, Lisa M. Perez, and Dhruva K. Chakravorty. 2022. Expanding Interactive Computing to Facilitate Informal Instruction in Research Computing. *The Journal of Computational Science Education* 13 (April 2022), 50–54. Issue 1. https://doi.org/10.22369/issn.2153-4136/13/1/9

[16] D. Moreau, K Wiebels, and C. Boettiger. 2023. Containers for computational reproducibility. *Nat Rev Methods Primers* (2023). https://doi.org/10.1038/s43586-023-00236-9

[17] "National Academies of Sciences, Engineering, and Medicine". 2019. *Reproducibility and Replicability in Science*. The National Academies Press, Washington, DC. https://doi.org/10.17226/25303

[18] Reid Priedhorsky, R. Shane Canon, Timothy Randles, and Andrew J. Younge. 2021. Minimizing privilege for building HPC containers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (St. Louis, Missouri) *(SC '21)*. Association for Computing Machinery, New York, NY, USA, Article 32, 14 pages. https://doi.org/10.1145/3458817.3476187

[19] A. Punjani, J. Rubinstein, D. Fleet, and et al. 2017. cryoSPARC: algorithms for rapid unsupervised cryo-EM structure determination. *Nat Methods* (2017). https://doi.org/10.1038/nmeth.4169

[20] Andrew Solis, William J. Allen, and Erik Ferlanti. 2022. Containerizing Visualization Software: Experiences and Best Practices. In *Practice and Experience in Advanced Research Computing* (Boston, MA, USA) *(PEARC '22)*. Association for Computing Machinery, New York, NY, USA, Article 22, 8 pages. https://doi.org/10.1145/3491418.3530769

[21] X. Carol Song, Preston Smith, Rajesh Kalyanam, Xiao Zhu, Eric Adams, Kevin Colby, Patrick Finnegan, Erik Gough, Elizabett Hillery, Rick Irvine, Amiya Maji, and Jason St. John. 2022. Anvil - System Architecture and Experiences from Deployment and Early User Operations. In *Practice and Experience in Advanced Research Computing* (Boston, MA, USA) *(PEARC '22)*. Association for Computing Machinery, New York, NY, USA, Article 23, 9 pages. https://doi.org/10.1145/3491418.3530766

[22] Yucheng Zhang and Lev Gorenstein. 2022. BioContainers on Purdue Clusters. In *Practice and Experience in Advanced Research Computing* (Boston, MA, USA) *(PEARC '22)*. Association for Computing Machinery, New York, NY, USA, Article 92, 2 pages. https://doi.org/10.1145/3491418.3535152

[23] Yucheng Zhang, Lev Gorenstein, Payas Bhutra, and Ryan DeRue. 2022. Containerized Bioinformatics Ecosystem for HPC. 1–10. https://doi.org/10.1109/HUST56722.2022.00006

[24] Gregory J. Zynda, Shweta Gopaulakrishnan, and John Fonner. 2021. RollingGantryCrane: Automation for unpacking containers into HPC environments. In *2021 3rd International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC)*. 29–34. https://doi.org/10.1109/CANOPIEHPC54579.2021.00008

A CONTAINERS IMAGES IN USE ON OUR CLUSTERS

alphafold, broad-gotc-prod, cactus, centos desktop, clara parabricks, dafoam, deep learning (intel), deep-learning (nvidia), fmriprep, freebayes, gem5, gptneox, hello world, hello world mpi, horovod, hpc-benchmarks, lammps, lifebitai, llm_eval, megatron_deepspeed, nemo, pav, pggb, python, pytorch (nvidia), pytorch-nemo, R, reads2maps, resnet, scipy, tensorflow, tensorflow 2, tensorflow gpu, tensorflow nn, tensorflow tnn, trimg, ubuntu desktop, vConTACT2, vsearch, wags.

B OTHER CAMPUS PUBLISHED CONTAINER TRAINING

All NSF ACCESS resource providers support the Singularity container runtime as a requirement for participation. However, among ACCESS resource providers, only Texas A&M, Purdue, SDSC, PSC, and TACC provide dedicated Singularity training while five other resource providers do not. Among ACCESS resource providers, only Texas A&M, Purdue [21], and Indiana University [6] have published support or training for Docker and Kubernetes. Besides Texas A&M, no other ACCESS sites have published support for Charliecloud or Podman at the time of writing.