MAX PLANCK
COMPUTING & DATA FACILITY

Bits & Bytes

No.220, December 2025

Computer Bulletin of the Max Planck Computing and Data Facility (MPCDF)*
https://docs.mpcdf.mpg.de/bnb

High-performance Computing

Viper News

In early December also the second part of the Viper deployment, *Viper-GPU*, was formally accepted resulting in the removal of the formal warning of "user-risk" mode operation. The compute nodes are still awaiting a BIOS and firmware update to fix an issue with nodes occasionally crashing when Inifiniband traffic is routed across the two sockets of a node. As a workaround, the default MPI runtime configuration on *Viper-GPU* accounts for each of the two sockets communicating only via their local network interface into the (dual-rail) Infiniband network and disables the automatic splitting of large MPI messages across the two interfaces. Similar mitigations are necessary for applications built on top of other communication libraries such as rccl. Specifically for machine-learning frameworks like Pytorch, users are advised to follow the recommendations for running on Viper-GPU[1].

On *Viper-GPU* an additional storage system based on "NVMe over Fabrics" was deployed which allows users to optionally attach fast node-local scratch storage to all nodes of a job using Slurm commands. Details can be found in the technical documentation[2]. Early next year the machine will be further expanded by 27 compute nodes (*Viper-GPU*, with 2 MI300A APUs each) and 5 additional login nodes (*Viper-CPU* and *Viper-GPU*) as a compensation for various delays in the delivery and deployment process.

MPCDF continuously updates the software stack on both *Viper* machines and provides training and application support for users, in collaboration with software engineers and application experts from AMD and Eviden.

Markus Rampp

Software News

AMD software on *Viper-GPU*

The two most recent versions of the AMD ROCm software suite have been installed on *Viper-GPU*. The modules `rocm/7.0` and `rocm/7.1` provide the ROCm versions 7.0.1 and 7.1.0, respectively. A new version of the LLVM-based AMD compiler `amd-llvm` has also been made available. Its version number 22.2 follows a new versioning scheme aligned to the LLVM release numbers. Hence, `amd-llvm/22.2` is the successor of `amd-llvm/7.1`. Fortran users, in particular, are advised to use the `amd-flang` compiler provided by this module, as AMD continues to release updates (so called "drops") for `amd-flang` with a higher cadence than for ROCm.

Tobias Melson

New version of JAX on *Viper-GPU* and *Raven*

JAX[3] is a Python library offering high-performance numerical computing and large-scale machine learning functionality on various backends comprising CPUs and accelerator devices from various vendors. On *Viper-GPU* (AMD/ROCm), JAX 0.7.1 has been installed in the module hierarchy for `python-waterboa/2025.06` and `rocm/7.0`. As usual, it can be found via `find-module jax`. On *Raven* (Nvidia/CUDA), JAX 0.7.1 is also provided as a module compatible with `cuda/12.8` for convenience. But for Nvidia GPUs, users who prefer different releases can also generally install JAX following the offical documentation[4].

Sebastian Kehl

*Editors: Dr. Renate Dohmen & Dr. Markus Rampp, MPCDF

[1] https://docs.mpcdf.mpg.de/doc/computing/software/data_analytics-machine_learning.html#cautions-and-bestpractice-notes-for-ai-workloads-on-hpc-systems

[2] https://docs.mpcdf.mpg.de/doc/computing/viper-gpu-flash-accelerators.html

[3] https://docs.jax.dev/en/latest/

[4] https://docs.jax.dev/en/latest/installation.html#nvidia-gpu

Major NumPy version update introduced with `python-waterboa/2025.06`

Since summer 2025 a newer version of our own Python distribution (`python-waterboa/2025.06`) is available on the HPC systems and clusters. This comes with a few notable changes from the previous version, in particular with respect to the scientific package NumPy which is now available in version 2.1.3. Most importantly, with the major version 2.0.0 the developers of NumPy have decided to *remove* many things that have been marked as deprecated in previous versions. The most obvious breaking change is probably the removal of the type aliases `np.int` and `np.float`. Users will either have to adapt their code or explicitly load the old version of the module to fix these issues. For a detailed list of all changes, refer to the NumPy 2.0.0 release notes[5] and for guidance on how to port old code to the NumPy 2.0 migration guide[6].

Henri Menke

Intel software stack

The new Intel oneAPI 2025.3 has been made available on *Raven*, *Viper* and other clusters. It provides the compiler module `intel/2025.3`, the MPI module `impi/2021.17`, the MKL module `mkl/2025.3`, and modules for the Intel profiling tools. As usual, the scientific software stack has been compiled with this toolchain.

The new `ifx` compiler also contains an enhancement for Fortran codes using the `COMPLEX` datatype. The compiler is now in many cases able to apply vector load-store (VLS) operations that combine load and store instructions. Codes with strided access of complex arrays in loops can benefit from this improvent, which is enabled by default.

On *Raven*, the default MKL version being loaded by invoking `module load mkl` will change. Currently, it points to version 2021.1. Since then, substantial improvements have been made to MKL. Starting from January 1st, the `module load mkl` command will always refer to the most recent version, which is 2025.3 at the moment. Users can still explicitly load older releases by specifying a version (e.g., `module load mkl/2024.0`).

Tobias Melson

Provisioning of AI Software

Both AMD and Nvidia provide highly optimized software stacks for machine-learning (ML) and artificial intelligence (AI) applications via containers. It is thus recommended to run ML & AI workflows by using the latest containers provided by AMD (for *Viper*) or Nvidia (for *Raven*). Specific hints and further references can be found in our documentaion[7].

Andreas Marek

New ELPA module version scheme

The ELPA library[8] provides highly optimized solvers for dense symmetric (Hermitian) eigenproblems. It delivers great performance on CPUs and GPUs and is available on *Viper*, *Raven*, and other clusters. Users can invoke `find-module elpa` to see all available ELPA modules.

With the upcoming ELPA release next year, we will change the versioning scheme of the corresponding environment modules. Instead of using the full version number (e.g., `elpa/mpi/standard/2025.01.001`), the bugfix number will be dropped (e.g., `elpa/mpi/standard/2026.01`). Bugfix releases can thus be provided seamlessly without the need for users to adapt their build scripts.

Tobias Melson

Introducing the MPCDF LLM Inference Service

We are proud to welcome a new member to our service portfolio: the MPCDF LLM Inference Service (LLMIS), available at https://llm.mpcdf.mpg.de.

There are over 100,000 open language models hosted on Hugging Face[9]. They cover a wide range of parameter sizes, from tiny models with a few million parameters to really gigantic models with up to one trillion parameters. While smaller models can outperform larger ones on specific tasks, especially when fine-tuned, the general capabilities of the biggest open models rival those of closed models such as GPT or Gemini.

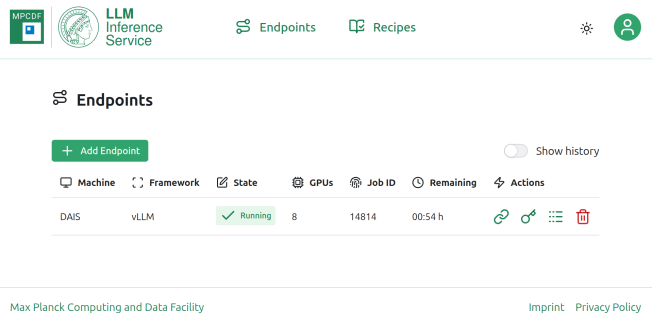

Figure 1: A screenshot of the LLM Inference Service app

[5] https://numpy.org/devdocs/release/2.0.0-notes.html
[6] https://numpy.org/devdocs/numpy_2_0_migration_guide.html
[7] https://docs.mpcdf.mpg.de/doc/computing/software/data_analytics-machine_learning.html
[8] https://elpa.mpcdf.mpg.de
[9] https://huggingface.co/models?pipeline_tag=text-generation&sort=trending

However, running even the "smaller" models efficiently already requires notable compute resources, and the largest models require substantial AI hardware that is often out of reach for individual research groups. MPCDF has both the resources and the expertise to operate these models. With our new *LLM Inference Service*, it becomes straightforward for you to interactively test even the largest open models available.

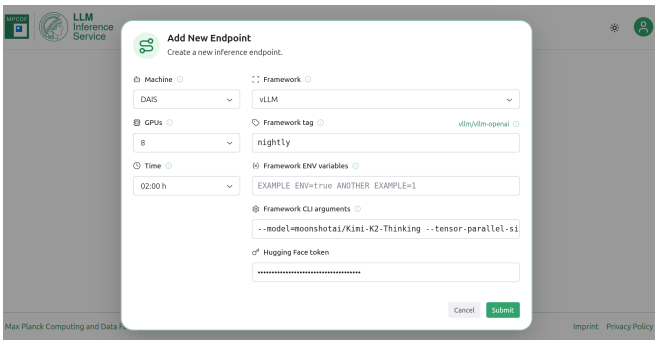

Figure 2: Screenshot of the "Add endpoint" dialog in the LLMIS

The LLM Inference Service is a flexible, yet easy-to-use web application that allows you to create endpoints exposing a model via a REST API. For this we rely on popular inference frameworks such as vLLM[10] and Ollama[11]. Via an intuitive UI, users can request the desired hardware and configure the framework. The service then takes care of submitting the Slurm job and routing the endpoint, so that you can conveniently access the REST API from your local machine or your existing tools.

Currently, two of the most powerful GPU systems at MPCDF, *dais*[12] and *Viper-GPU*[13], are connected to the service. We provide sensible default configurations for the frameworks to help you get started quickly. At the same time, you remain free to tune the framework parameters to your needs and to run any model and modality supported by the respective framework, including your own fine-tuned models, provided they are hosted on the Hugging Face Hub.

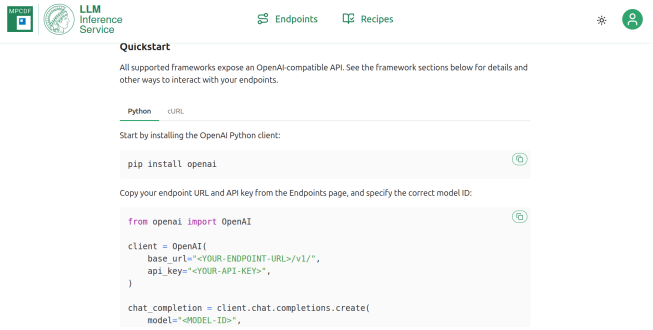

Figure 3: Screenshot of the Recipes page in the LLMIS

The LLM Inference Service is targeted at researchers who wish to interactively evaluate specific open models or conduct interactive user studies. For non-interactive workloads, such as extensive benchmarks or offline evaluations, we recommend using Slurm batch jobs to ensure efficient resource utilization. Example scripts for submitting such jobs are available in our LLMs-meet-MPCDF[14] GitLab repository. Additionally, for users interested in testing "standard" open models, the Chat AI[15] service by GWDG is an excellent alternative. It offers a user-friendly chat interface as well as access to an inference API[16].

We hope the LLM Inference Service will facilitate your scientific work and open up new research opportunities. You can expect the service to evolve over time as we add more hardware options and additional inference frameworks. Any kind of feedback is much appreciated; we are looking forward to hearing from you in the AI@MPCDF discourse channel[17], or via our helpdesk[18].

David Carreto Fidalgo, Nastassya Horlava, Andreas Marek

GitLab CI

Building Docker images with Podman-Runners

Some users of the *Continuous Integration* in GitLab need to create their own custom container images. As the Docker-based runners in our GitLab infrastructure are running in unprivileged mode, using *Kaniko* was one of the possible solutions. With Kaniko, a user can create container images from Dockerfiles inside Docker containers or Kubernetes pods. Due to some recurring issues, Kaniko's repository was archived earlier this summer, and

[10] https://github.com/vllm-project/vllm
[11] https://ollama.com/
[12] https://docs.mpcdf.mpg.de/doc/computing/dais-user-guide.html
[13] https://docs.mpcdf.mpg.de/doc/computing/viper-gpu-user-guide.html
[14] https://gitlab.mpcdf.mpg.de/dataanalytics-public/llms-meet-mpcdf
[15] https://docs.hpc.gwdg.de/services/chat-ai/index.html
[16] https://docs.hpc.gwdg.de/services/saia/index.html
[17] https://discourse.hpc.gwdg.de/c/ai-at-mpcdf/
[18] mailto:support@mpcdf.mpg.de

is no longer maintained.

After taking alternatives into consideration, we decided to introduce an alternative, more stable solution: two new GitLab runners specifically designed for building custom Docker images. The new runners are:

- podman-runner-01
- podman-runner-02

As the names suggest, these runners are Podman-based rather than Docker-based. Podman[19] offers a smoother, more secure way to create new container images inside a CI pipeline. Below is an example of how to build an image that you can integrate into your CI pipeline:

```
build_image:
  tags:
    - image-builder
  variables:
    IMAGE_TAG: $CI_REGISTRY_IMAGE:$CI_COMMIT_
      REF_SLUG
    BUILDAH_FORMAT: dockerNew GitLab Runner Tags
    BUILDAH_ISOLATION: chroot
  image: quay.io/buildah/stable

before_script:
  - buildah login -u "$CI_REGISTRY_USER"
      -p "$CI_REGISTRY_PASSWORD" $CI_REGISTRY
script:
  - buildah build -t $IMAGE_TAG .
  - buildah push $IMAGE_TAG
```

Required Values:
tags
Your job must be tagged with "image-builder" or it won't run on the correct GitLab runner.

BUILDAH_FORMAT
Use docker instead of oci, especially when building derivative images, for better compatibility.

BUILDAH_ISOLATION
Set this to chroot because the default runc runtime doesn't work in a rootless environment.

image
We recommend using either quay.io/buildah/stable or quay.io/podman/stable (Buildah is included in the Podman binaries).

before_script and script
These sections handle authentication to Gitlab's container registry and the image build/push process. Use buildah[20] instead of docker commands to avoid confusion and ensure compatibility. As the names suggest, these runners are Podman-based rather than Docker-based.

Building container images inside a rootless, unprivileged environment can be tricky, and Podman offers a smoother, more secure solution for this use case.

Keep in mind that different stages of your CI pipeline may run on separate GitLab runners. If your build stage produces files or binaries that need to be included in your final container image, you have to make those artifacts available across jobs and runners. To do this, add an artifact definition at the end of your build stage:

```
[...]

artifacts:
  paths:
    - path/to/artifact
  expire_in: 1h

[...]
```

This ensures that the output from one job can be reused in a later job that builds the container image.

If you are still using Kaniko or need to build custom Docker images within a containerized environment, this setup provides a secure, modern, and fully supported alternative!

Francesco Turcinovich

New tags for MPCDF GitLab runners

For executing Continuous Integration Pipelines, the MPCDF operates shared GitLab Runners, which can be used by any GitLab user. To better reflect the hardware landscape of the HPC clusters, the runner infrastructure has been continuously extended and equipped with new hardware, including AMD and Nvidia-based GPUs. In parallel, the basic capabilities of the shared runners have been streamlined so that all of them now support the same features (e.g. distributed cache).

To choose a runner with specific capabilities, users can specify tags in their CI pipelines. To make this tagging system more explicit and future-proof, we decided to re-implement it from scratch. The new tags will be added in addition to the existing ones. Until March 1st, 2026, both tagging systems can be used in parallel. After that date, the old tags will be removed, and only the new ones will remain. Please ensure that you have adapted your CI pipelines by then.

New Tags
mpcdf-shared:
All of our managed shared runners do have this tag. Use this tag if you want to make sure the job runs on one of the MPCDF shared GitLab runners instead of runners of other GitLab instances or user-started runners.

image-builder:
Use this tag when you need to build your own custom Docker image (read article "Building Docker images with Podman-Runners" above).

[19] https://www.redhat.com/en/blog/podman-inside-container
[20] https://docs.gitlab.com/ci/docker/buildah_rootless_tutorial/#configure-the-job

Hardware-specific tags
They are organized hierarchically based on their level of specificity

gpu:
Use when a runner with a GPU is needed, but the vendor or architecture remains unspecified.

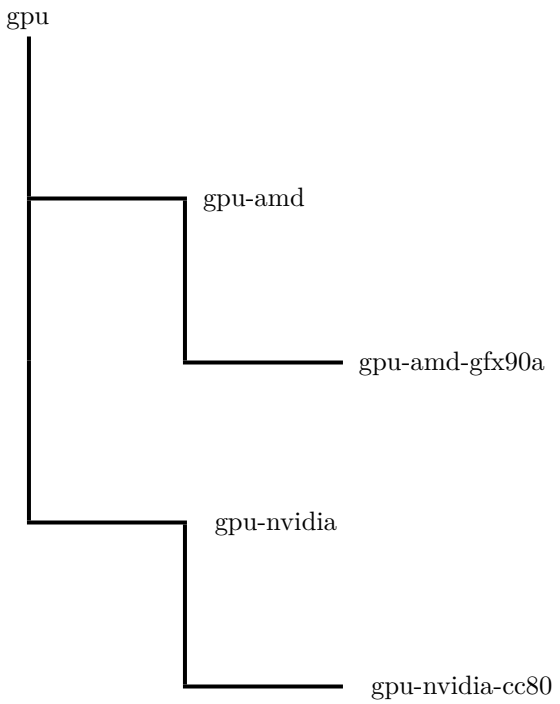

Figure 4: Hierarchy of tags for GPU runner usage

gpu-amd:
Use when an AMD GPU runner is needed, regardless of architecture or instruction set.

gpu-amd-gfx90a:
Use when an AMD runner with offload architecture gfx90a (e.g. for the MI200 GPU) is required.

gpu-nvidia:
Use when an Nvidia GPU runner is needed, regardless of architecture or instruction set.

gpu-nvidia-cc80:
Use when an Nvidia runner with Compute Capability 8.0 (e.g. for the A40 or A100 GPU) is required.

cpu:
Use when a runner with a CPU is needed, but the vendor or architecture remains unspecified.

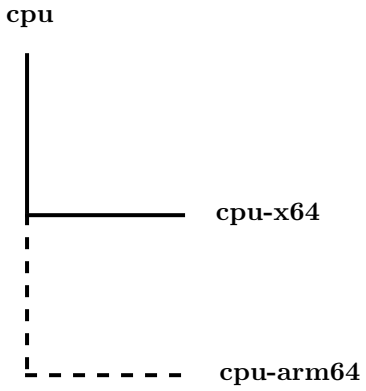

Figure 5: Hierarchy of tags for CPU runner usage. Currently no arm64 runner is available.

cpu-x64:
Use when a runner with x64 architecture is needed.

cpu-arm64:
Use when a runner with ARM64 architecture is needed in the future (dashed line in Fig. 5).

Old Tags
As our pool of runners is homogeneous in configuration, most of the old tags have become redundant or superfluous. Here is what we removed and why:

docker:
No longer needed, as all runners are Docker-ready.

distributedcache:
All runners are now configured to enable the distributed cache for job artifacts.

avx / avx2 / avx512:
All runners are compatible with these CPU flags.

shared:
Renamed to mpcdf-shared for clarity.

cloud:
Previously used to differentiate underlying hardware; no longer necessary.

modules:
Previously used before modules were made available as Docker images.

Francesco Turcinovich

New naming scheme for CI module images containing CUDA and ROCm

MPCDF offers a variety of Docker images for running jobs in GitLab CI pipelines. These images provide environment modules that are very similar to those on our

[21] https://docs.mpcdf.mpg.de/doc/data/gitlab/gitlabrunners.html#docker-images-for-ci-with-mpcdf-environment-modules

clusters, enabling users to run their codes on the GitLab CI runners with the same software stack as in production on the clusters. Detailed documentation can be found here[21].

Following our standard update policy, all images labelled 2025 will be frozen on January 1st, 2026. New images for 2026 will contain updated versions of the included packages.

Currently, we refrain from removing software from the images during the year. However, the rapid development and size of the ROCm package forces us to adapt the image naming scheme to further maintain this strategy. The following changes will apply with the turn of the year.

Based on the existing images for certain compiler and MPI combinations, we will create additional images containing `cuda` or `rocm` modules. For example, the image `gcc_15-openmpi_5_0-rocm_7_1` will be based on `gcc_15-openmpi_5_0` with an additional `rocm/7.1` module installed. Images that are not labelled accordingly will not contain installations of `cuda` or `rocm` modules.

All new images will be added to the existing list on this website[22]. Unversioned images will also be provided, for example `gcc-openmpi-cuda` or `gcc-openmpi-rocm`, pointing to the latest versioned images. This will be useful for users who are not concerned with the specific version of the compiler, MPI library, or GPU software, but who want to run their codes with the latest releases.

Be aware that only the latest LLVM-based new AMD compiler `amd-llvm` will be included in the `rocm` images. Due to a size limitation, older versions of this particular module will be deleted when the module gets updated. To use the latest version of the `amd-llvm` module in your pipeline automatically, load it with the command `module load amd-llvm` without specifying any version number.

Tobias Melson, Klaus Reuter

HPC-Cloud Software Updates

The MPCDF HPC-Cloud provides on-demand computing and storage resources to research projects of the Max Planck Institutes. As an infrastructure-as-a-service cloud, it offers self-provisioned servers, networking, and storage resources through a high-level API, CLI, and GUI.

For the past four years, the HPC-Cloud infrastructure has been deployed and managed by the TripleO tool. It has served us well during this time for tasks such as adding compute resources and managing software upgrades. However, in February 2023, it was announced[23] that the project would be discontinued, leaving us without an upgrade path for the OpenStack "Wallaby" release currently in production. Therefore, we started the process of switching to a different tool. Since the HPC-Cloud is a productive platform on which many projects and users depend, this posed the following technical challenge: How do we migrate away from TripleO with as little disruption as possible for our users?

Looking for alternatives

After an initial evaluation round covering many of the deployment tools[24] that can be used to provision a full-featured OpenStack cloud, we made the decision to migrate to Kolla-ansible. Some of the factors that tipped the scales in favor of this tool were its similarity in terms of architecture of the control, network and compute planes to TripleO, and an extensive community that has been keeping the project healthy, supporting the newest releases of the OpenStack components.

Kolla-ansible has several interesting characteristics that make it a good fit:

1. Control plane and compute architecture similar to TripleO, i.e. containerized services running on baremetal servers.
2. Companion project Kolla provided us with a workflow to build our own versions of the containerized OpenStack services, allowing us to patch them much more easily and freely than with TripleO.
3. Written in Ansible, which is already used extensively throughout the Virtualization and Storage Teams at the MPCDF, and is also highly customizable.

Preparing for the migration

Having decided on a new tool, the next challenge was to design a migration plan that allowed us to replace

[22] https://mpcdf.pages.mpcdf.de/ci-module-image/
[23] https://lists.openstack.org/pipermail/openstack-discuss/2023-February/032083.html
[24] https://www.openstack.org/software/project-navigator/deployment-tools

the TripleO-deployed services with Kolla-ansible-deployed equivalents, while still keeping the system up and running. In order to achieve this, we had to make sure that the Kolla-ansible services had the same configuration and features as before. This meant first changing the software versions of the deployed services as little as possible to maintain compatibility to the rest of the system, second, making sure the Kolla-ansible-generated configuration would be as similar as possible to the TripleO counterpart, and finally, testing that the system was still stable after deploying each replacement.

None of this could be fully automated, since it is not a feature that either of the two tools provide, and it is highly dependent on the initial configuration. Thus, it required significant manual work to adjust all details.

For each service, a rough outline of the migration procedure would be:

1. Analyze the TripleO configuration and try to match relevant settings to Kolla-ansible counterparts.
2. If not possible to map an option, customize the Ansible code that deploys the service or any relevant configuration file templates.
3. Move TripleO components out of the way, but leave containers and configuration files on any affected servers to permit changes to be reverted if needed.
4. Deploy with Kolla-ansible.
5. Compare the generated configuration to TripleO.
6. Test service integration.
7. Repeat until satisfied with the end result.

Several core services required even a few extra steps. For instance, the main database store had to be moved to a different location. Or highly-available services had to be unregistered first in order to stop all instances. All of which required adding additional logic before step four in the form of Ansible code.

Moreover, going through this iterative process on the productive system would have been impossible without disrupting the users, thus a test environment was used. Our test environment is a scaled-down version of the productive HPC-Cloud, with fewer, smaller servers, but with the same set of deployed services, configured with settings as similar as possible at all times.

Conclusions

Following a successful primary stage of the migration in early November, 17 out of 20 services have been migrated to Kolla-ansible. A secondary phase will be scheduled for early 2026 to finish up the remaining services, leaving us prepared for a major upgrade to the next OpenStack release later in the year. Apart from solving the issues mentioned above, there are already two user-visible changes: A more featureful version of the noVNC embedded graphical console and a new version of the web dashboard with 2FA support. Although not yet mandatory during this transition phase, the 2FA-enabled “MPCDF Login” authentication will become the only accepted method for the dashboard.

This new configuration puts us in a much better position to roll out major release upgrades as well as new services, which will result in a more robust HPC-Cloud with more features and better performance.

Brian Standley, Maximiliano Geier

Globus Migration to SelfService/MPCDF SSO

In January 2026 the Globus services at MPCDF will be re-configured to use the MPCDF Single sign on (SSO) and the opt-in service model via SelfService. These changes will impact all Globus Services at MPCDF (DataHub, GO-Nexus, GO-S3) and in some cases the association to Globus Groups within the Globus Web Portal. The migration to MPCDF SSO will provide both, 2FA and SSO functionality, for improved security and a better user experience. Moreover, these changes bring Globus in line with other MPCDF services such as GitLab and DataShare, providing a common user experience. A transition period is foreseen for January 12th until February 9th and we ask users to actively test their accounts during this period.

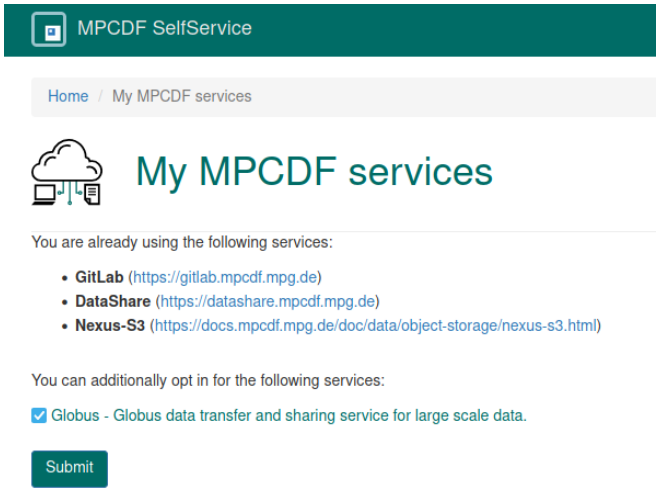

Figure 6: SelfService Opt-In for Globus

More details regarding each aspect of the re-configuration are provided here.

Opt-In: In the future users will be required to opt-in to the Globus Services via SelfService (see screenshot in Fig. 6). Where possible this option has already been enabled for existing Globus users. However, we kindly ask users to check their status in the SelfService[25]. Please note that opt-in is only available for standard MPCDF user accounts, invited guests (g-account) do not have this option.

SSO: From January 12th to February 9th access to the Globus endpoints at MPCDF will be possible using either login.datahub.mpcdf.mpg.de or mpcdf.mpg.de domains. The existing login domain (login.datahub.mpcdf.mpg.de) will be decommissioned on February 9th. After that date only the new domain (mpcdf.mpg.de) which is provided by the MPCDF SSO will be available for login.

Groups within the Globus Web Portal: Users who have registered for access to the *Max Planck Computing and Data Facility* and *MPCDF Flows Users* Groups within the Globus web portal will be required to re-register, ideally using their primary ID in Globus. On January 12th existing Group access to user with login.datahub.mpcdf.mpg.de as domain will be revoked, forcing users to re-register. On February 9th the login.datahub.mpcdf.mpg.de domain will be decommissioned and access to groups via that ID will no longer be possible.

We ask all Globus users to please check their Opt-In Status, test access to Globus services via MPCDF SSO and cross-check their group membership in the Globus web portal (re-applying for membership if needed).

Note: During the overlap period, January 12th to February 9th, both login domains may be used. However, only users who have opted in to the Globus service will be able to access the Globus endpoints at MPCDF.

John Alan Kennedy

News & Events

Multifactor authentication: deactivation of E-mail tokens

The deactivation of E-mail tokens that was announced in the last issue Bits & Bytes[26] had to be postponed. The plan is now to inform all users directly and then to deactivate in the course of the first quarter of 2026, after all users have switched over to one of the supported token types: app, external hardware tokens, SMS, and TAN list. Note, that SMS and TAN list are intended only as fallback.

Users are kindly asked to ensure that they have a valid app or hardware token, ideally in combination with an SMS token or a TAN list as backup.

Kathrin Beck, Andreas Schott

International HPC Summer School 2026

The International HPC Summer School (IHPCSS) 2026 will take place from July 12th to July 17th in Perth, Australia. The series of these annual events started 2010 in Sicily, Italy and provides advanced HPC knowledge to computational scientists, focusing on postdocs and PhD students. Through the participation of Canada, the USA, South Africa, Japan, Australia and Europe a truly international group of highly motivated students is meeting each year.

Interested students and postdoctoral fellows are invited to apply at the school's website[27] by January 31st, 2026. School fees, travel, meals and housing will be covered for all accepted applicants through funds from the European Union and EuroHPC. For further information, please visit the website of the summer school.

Erwin Laure

AMD workshop

Next spring, MPCDF will organize another AMD GPU workshop with a focus on the MI300A technology in *Viper-GPU*. This time, the workshop is split into two parts: A first part in collaboration with HLRS[28] (they also have AMD MI300A APUs in their Hunter system) with four half-day sessions of lectures and exercises in the afternoons of April 21st to 24th, 2026 , and a second part with a hackathon on *Viper-GPU* from April 27th to 29th. The online lectures of the first part will be given by AMD, and the accompanying exercises will be done on AMD cloud resources. The program will soon be published on our webpage, and attendees may select parts of the program according to their knowledge and topics of interest.

For the online hackathon on the MPCDF *Viper-GPU* system the week after users are invited to apply with their code to bring in and to work on profiling and optimization aspects or specific porting issues, supported by experts from AMD and MPCDF. Participants are expected to have a good understanding of their code and they should also be familiar with the relevant parts of the lectures of the previous week, as there will be no introductory

[25] https://selfservice.mpcdf.mpg.de
[26] https://docs.mpcdf.mpg.de/bnb/219.html#multifactor-authentication-deactivation-of-e-mail-tokens
[27] https://ss26.ihpcss.org
[28] https://www.hlrs.de

lectures for the week of the hackathon.

Registration for both events will open early next year and will be announced on our website.

Tilman Dannert

Meet MPCDF

In our "Meet MPCDF" series, one seminar is planned so far for the beginning of next year:

- February 5th, 15:30-16:30, Efficient usage of the tape archive

We encourage our users to propose further topics of their interest, e.g. in the domains of high-performance computing, data management, artificial intelligence or high-performance data analytics. Please send an E-mail to training@mpcdf.mpg.de[29].

Tilman Dannert

Introduction to MPCDF Services

The next session of our introductory online course, which is designed to familiarize new users with the MPCDF compute and data services, will be held on April 30th, 2026, 14:00-16:30, online. No registration is necessary, you can just join with the link published on our website. The link is only active at the time of the workshop.

Tilman Dannert

[29] mailto:training@mpcdf.mpg.de