CARME

An Open-Source Framework for Multi-User,
Interactive Jobs on Distributed GPU-Systems

The Carme Team
Competence Center for High Performance Computing,
Fraunhofer ITWM, Kaiserslautern, GERMANY

Fraunhofer
ITWM

© freepik

Machine Learning in the Wild

Why We Need a Tool for ML/DL on HPC Clusters

Fraunhofer ITWM

Machine Learning in the Wild

Why We Need a Tool for ML/DL on HPC Clusters

Machine Learning and Data Analytics

- showed **remarkable success** in many distinct fields
  (e.g. image processing, text translation, medical imaging, …)
- rise of **hybrid** and/or **ML/DL** supported **algorithms**
  (e.g. climate forecast, physics informed networks, …)
- **model size increases** drastically
  (needs more memory & compute power)
- generate **large investments** in new **multi-GPU hardware**

Utilize HPC Systems

The Way It Was for Decades

- other fields use **HPC systems for decades**
  (e.g., physics, chemistry, mathematics, …)
- HPC clusters are set up to **make use of** the **hardware** that is installed
  (e.g., multi-node jobs)
- **calculate huge problems**
  (e.g., extreme memory and computationally intensive simulations)
- **command line driven** usage
  (e.g., submit non-interactive batch jobs, jobs do not start directly)

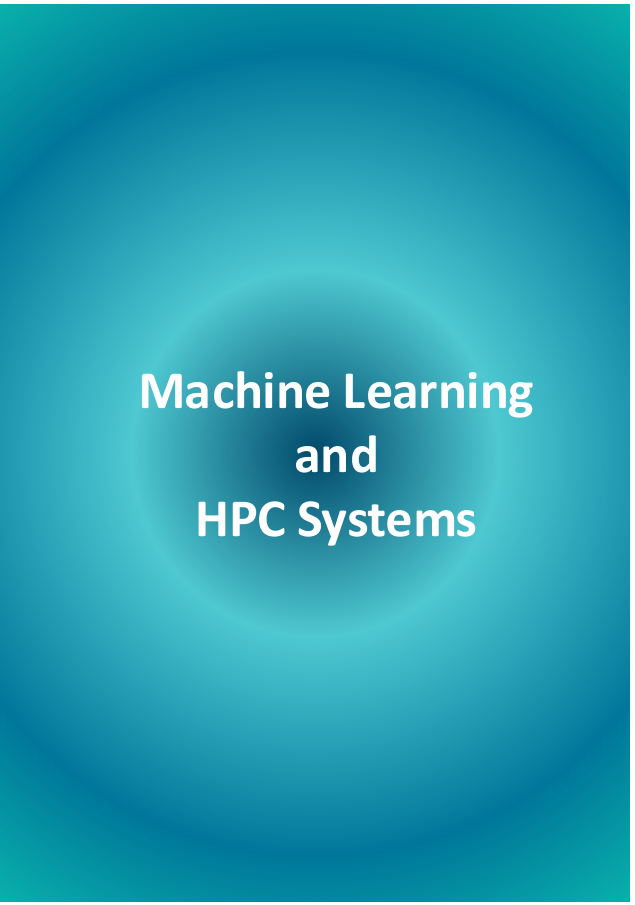
© iStock

HPC Systems

- huge compute resources
- resources are spread over various nodes
- often focus on parallel usage
- limited runtimes

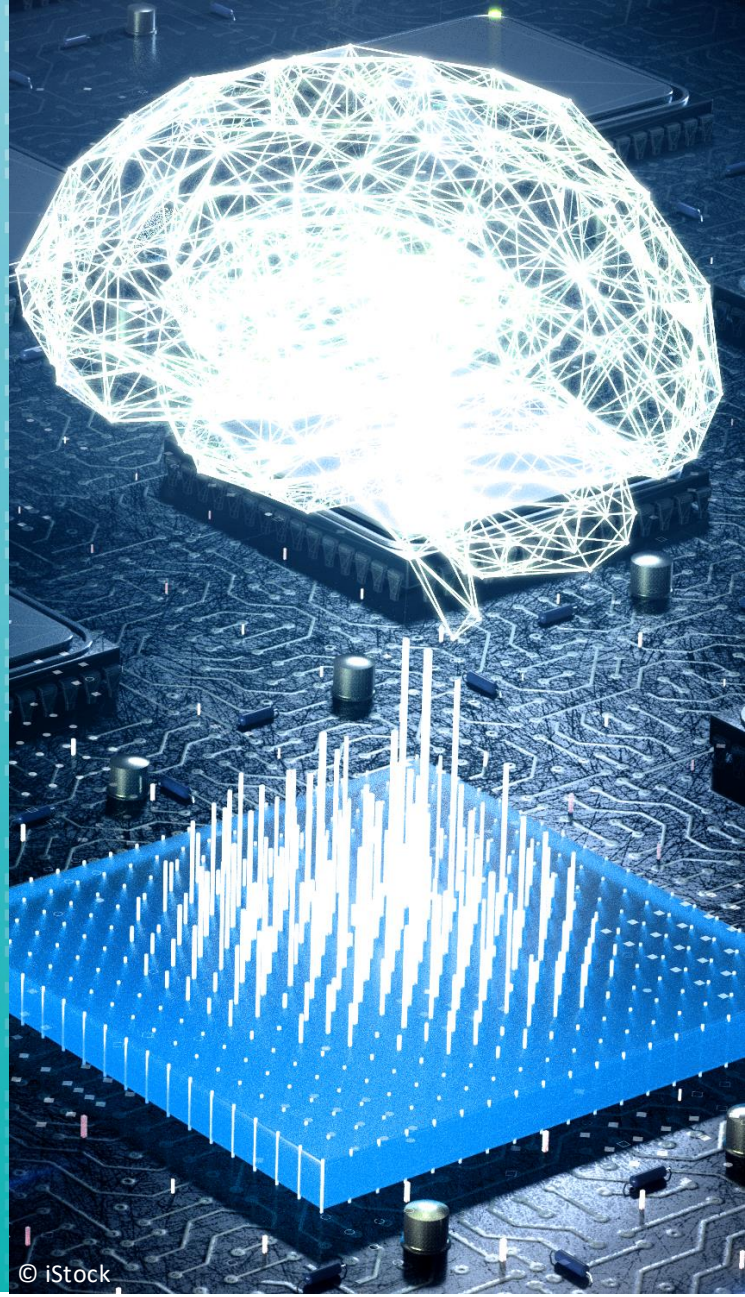
Machine Learning and HPC Systems

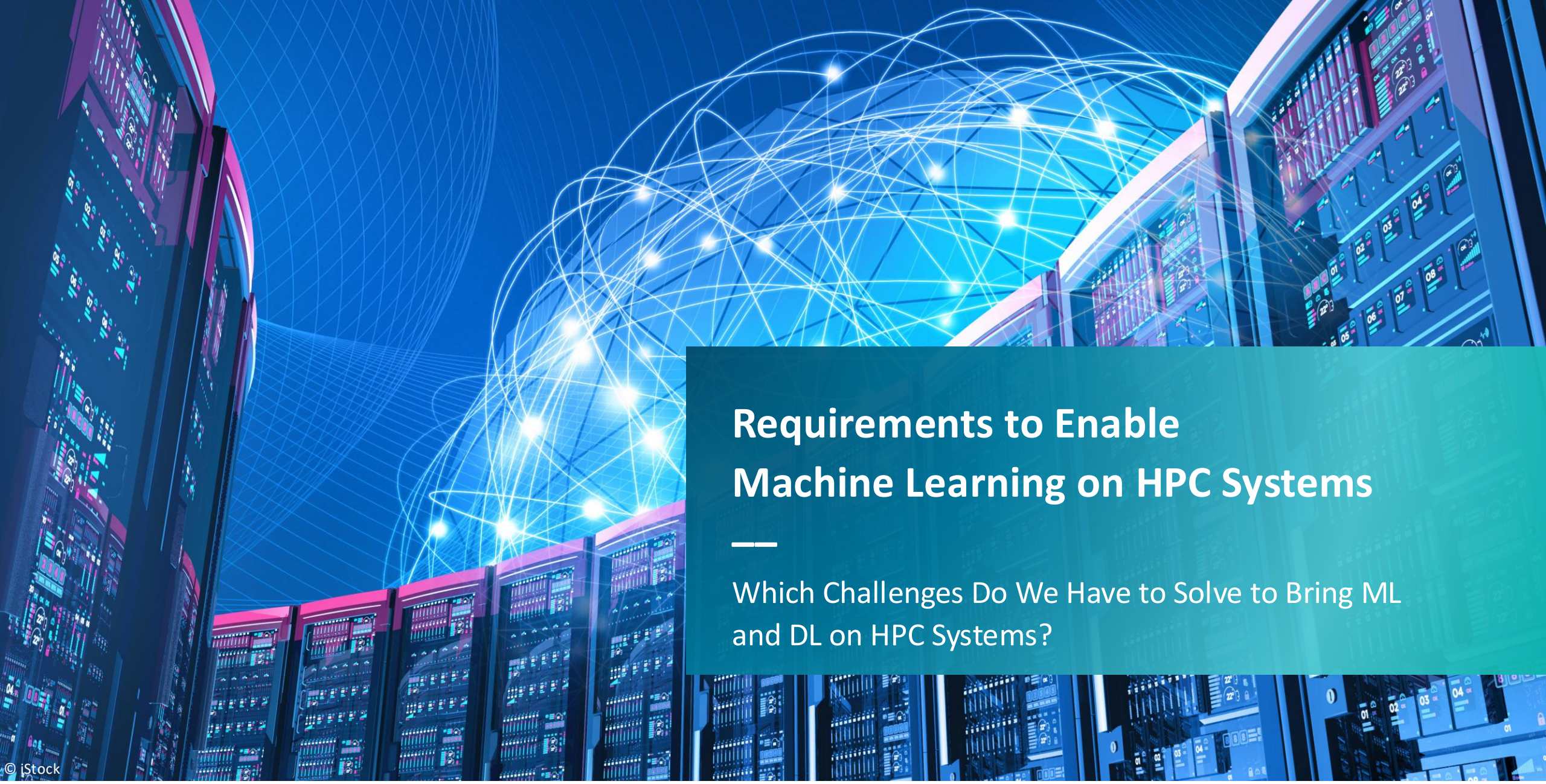
What We Would Expect



- efficiently **manage** and **use** your **resources**
  (how to utilize the full power of multi-GPU systems?)
- **match your workflow** with these resources
  (how can I develop interactively?)
- scale your applications to **use multiple GPUs** and/or **nodes**
  (how do I easily utilize multiple GPUs in various nodes?)
- manage **data I/O** and data **storage**
  (how do I access my data?)

Machine Learning
and
HPC Systems
Two Distinct Worlds?





Requirements to Enable Machine Learning on HPC Systems

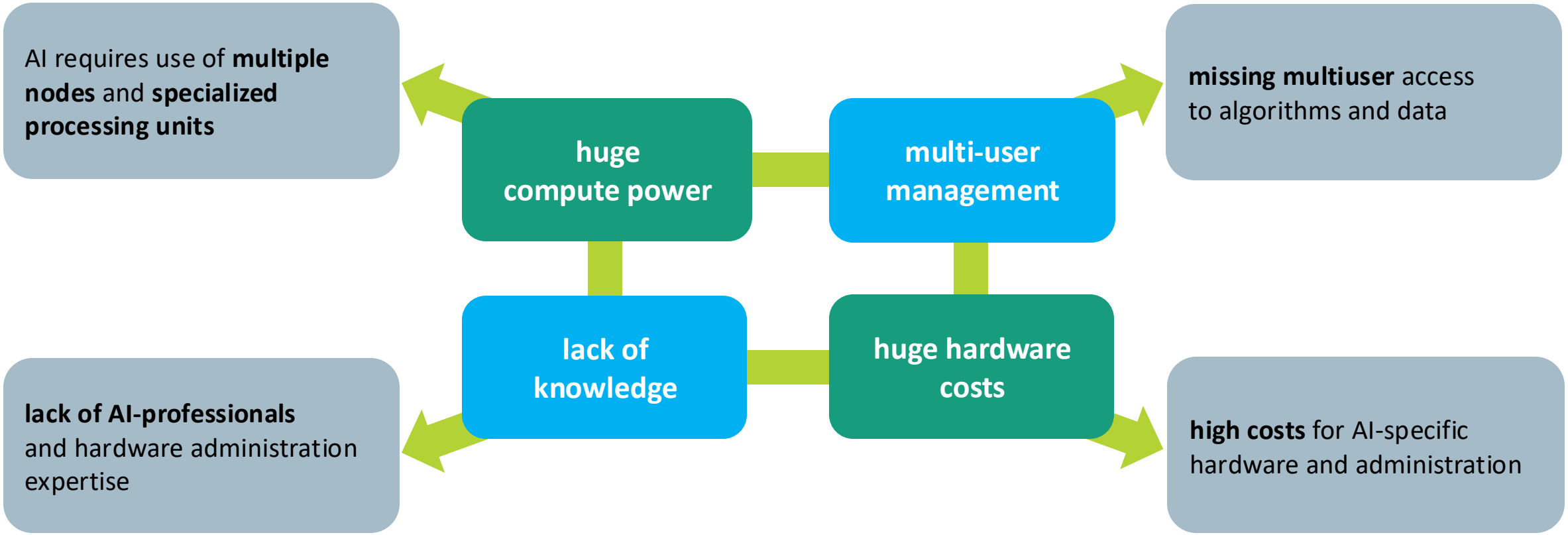
—

Which Challenges Do We Have to Solve to Bring ML and DL on HPC Systems?

© iStock

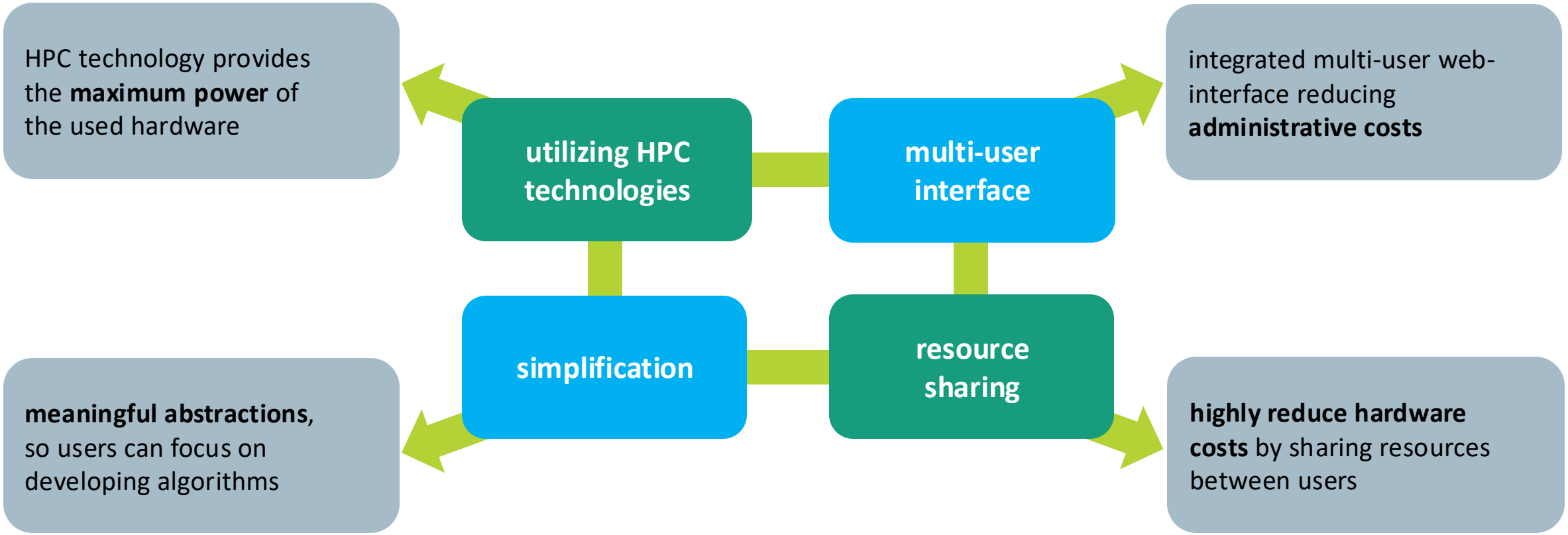
Machine Learning and HPC Systems

Things We Must Take Care of on Hpc Systems



Machine Learning and HPC Systems

Our Proposed Solution: Proven HPC Tools



Combining Open-Source ML Tools with HPC Backends

The CARME Way

Carme – an open-source framework to manage resources for multiple users running interactive jobs on an HPC Clusters

combines

- **secure web frontend main GUIs**
  - Code-Server/JupyterLab
  - TensorBoard
- **anaconda environments**
  - miniconda / mambaforge

Data Science

- **user and job management**
  - **LDAP** and **SLURM**
- **data I/O technologies** (e.g. **BeeGFS**)
- **using containers** (**Singularity**)
- **monitoring tools** (e.g. **Zabbix**)

HPC

Open–Source Tools

https://carmeteam.github.io/Carme/

The Details of Our Solution

What It Is Really About

© freepik

Resource Management

How to Assign Resources to Competing Users?

© freepik

Resource Management

Proven HPC Tools

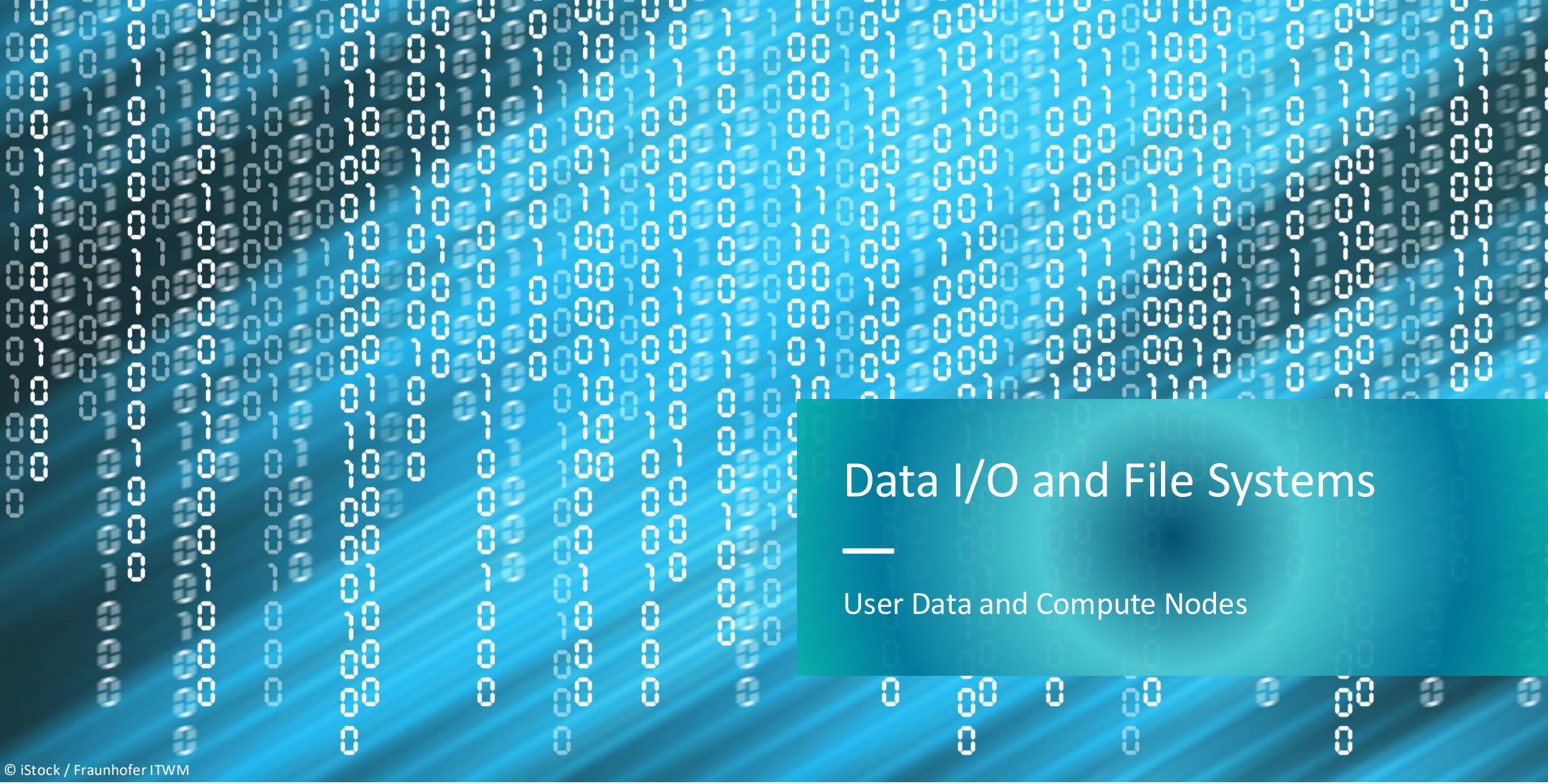
LDAP[1] (user management)

- LDAP = Lightweight Directory Access Protocol
- use your LDAP
- different user roles

SLURM[2] (job scheduler)

- easy integration in existing installations
- make use of cgroups plugins
- using quotas, queues, resource reservation
- many useful extensions available
  (e.g., preemtion, hiberate if idle, …)

[1] https://www.openldap.org
[2] https://slurm.schedmd.com



Data I/O and File Systems

User Data and Compute Nodes

Data I/O and File Systems

BeeGFS[3]

- redundant, parallel file system
- **huge** data **throughput** (via infiniband)
- temporary job FS (BeeOND) on local SSDs between node
- **open-source** (commercial support available)


© iStock

[3] https://www.beegfs.io

Maintenance and Usability

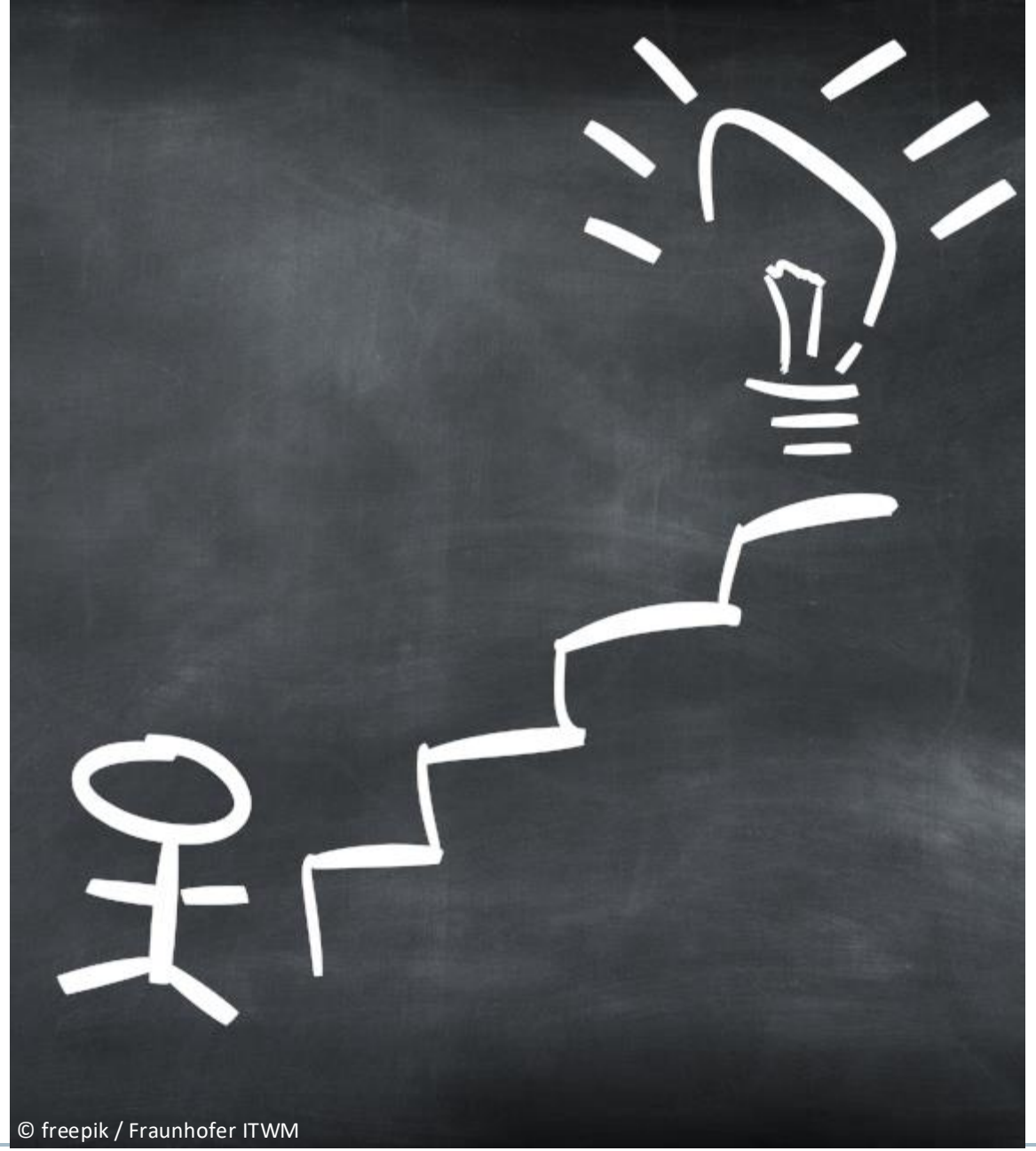
Meet User Software Demands

© freepik

Maintenance and Useability

Demandings, We Have to Care

- **using graphic cards for calculations**
  - install drivers, the corresponding libraries and TensorFlow, PyTorch, …
  - easy GPU access
- **ML/DL algorithms**
  - libraries, programs and dependencies change fast
  - different algorithms need different programs/libraries
- **programs/tools often available as deb-packages**
  - but many HPC clusters are rpm-based
- **HPC clusters have very heterogeneous users**
  - need a lot of different tools and some dependencies may collide
  - users cannot be root



Maintenance and Useability

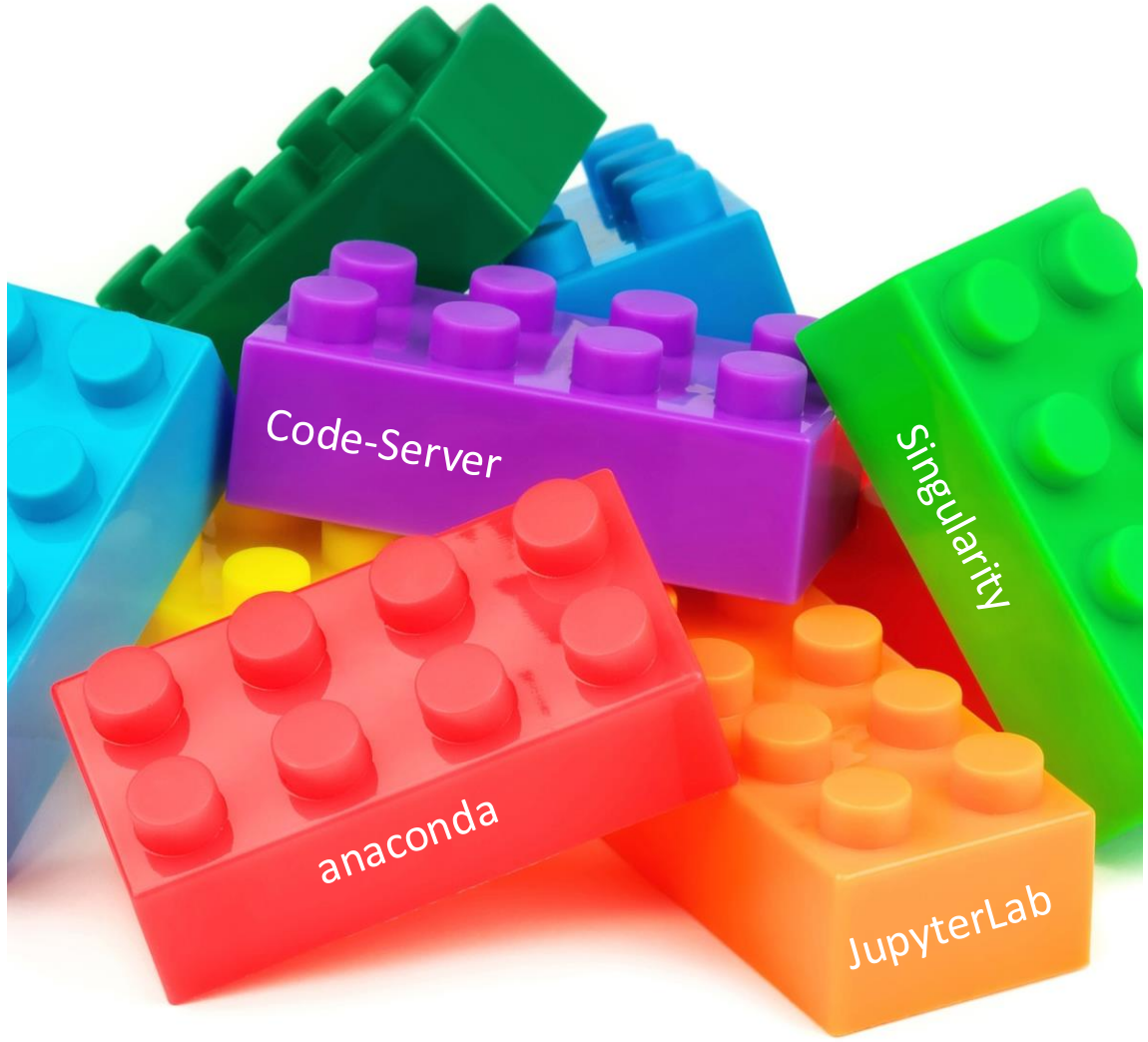
Easy for Users and Admins

easy to use (for users)

- access via web interface
- interactive development tools
  (e.g., Code-Server, TheiaIDE, JupyterLab)
- direct GPU access
- uncomplicated multi-node/-GPU usage

easy maintainable (for administrators)

- provide different libs for specific user groups
- use anaconda environments
- stick to singularity containers



© freepik / Fraunhofer ITWM

Maintenance and Usability

Portable Software Environments

singularity[4] containers

- provide the OS that ML/DL users need
- on the host: base OS and graphics drivers
  in the image: all other dependencies
- no root-privileges to start the container
- only needed folders mounted
  (e.g., `/home/USERNAME`, `/scratch/USERNAME`)
- create images from scratch (takes a few minutes)
  or transform an existing (clean) docker image

[4] https://www.sylabs.io/singularity

Maintenance and Useability

The Base Environment CARME Provides

- set of **basig software**
  (e.g. vim, nano, gcc, gdb, …)
- everything **ready to use IB**
  (using free drivers)
- **MPI** and **GPI** are installed per default
  (GPISpace installation ready)
- **miniconda** and/or **mambaforge**
- **Python 3.X**
- **Code-Server** and **JupyterLab**
- **headless** or **web-based applications** can run inside the container

Maintenance and Useability

Userspace Package Management

anaconda[5] based environments

- **open-source** (normal and enterprice versions)
- central **miniconda/mambaforge** installation inside the image (people can use it directly)
- users have the freedom to **install** (most) of the **libs and tools** they need (precise version control)
- all **PyTorch** and **TensorFlow** versions possible
- easily **share** and **save** your **environments**
- supporting **cooperation**

[5] https://www.anaconda.com

© freepik

Resource Utilization

Who Has Access to Which Resources?

Resource Utilization

Control the Resources You Have

- CARME utilizes the **scheduler resource limits** (fully respects non-carme jobs, cgroups, queues, etc.)
- **additional limits** can be defined **on top** of the scheduler limits (modify #(nodes), #(GPUs), running jobs, etc.)
- **integrates** smoothly **in** your **existing scheduler** (only a CARME feature flag is needed)
- everything **inside a job** runs inside an **isolated namespace** (programs cannot escape a job)
- inside a job you can **only** see the **GPUs associated with** the **job** (no access to information of other GPUs on the node)

© iStock

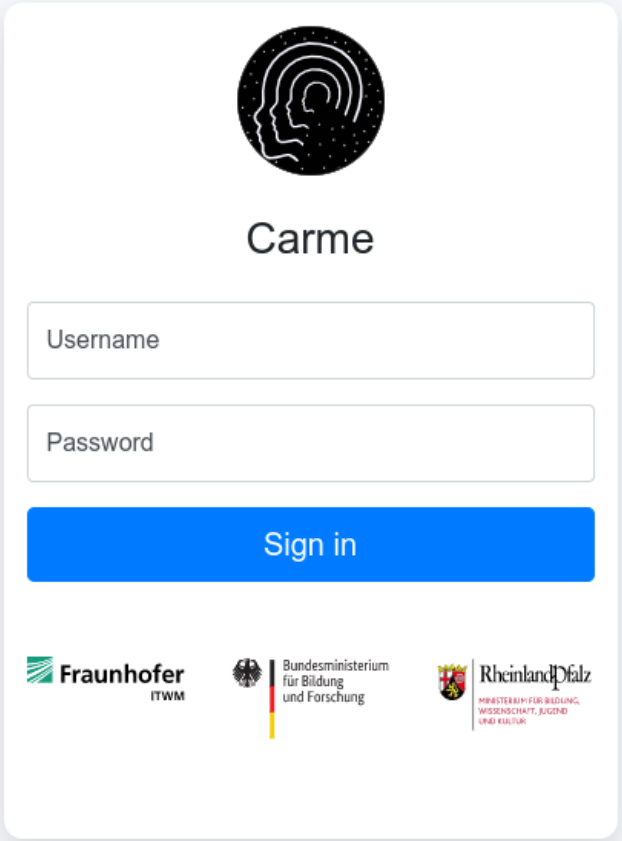
Interactive and Secure Multi-User Environments

ML/DL and Data Science Users Want Interactive Access

Interactive Usage

Secure and Simple Access

- access via **web interface**
- **no additional software** (like ssh or special tool)
- runs **behind your firewall**
- **secure autehntification** (e.g. via our LDAP) and additional **2FA**



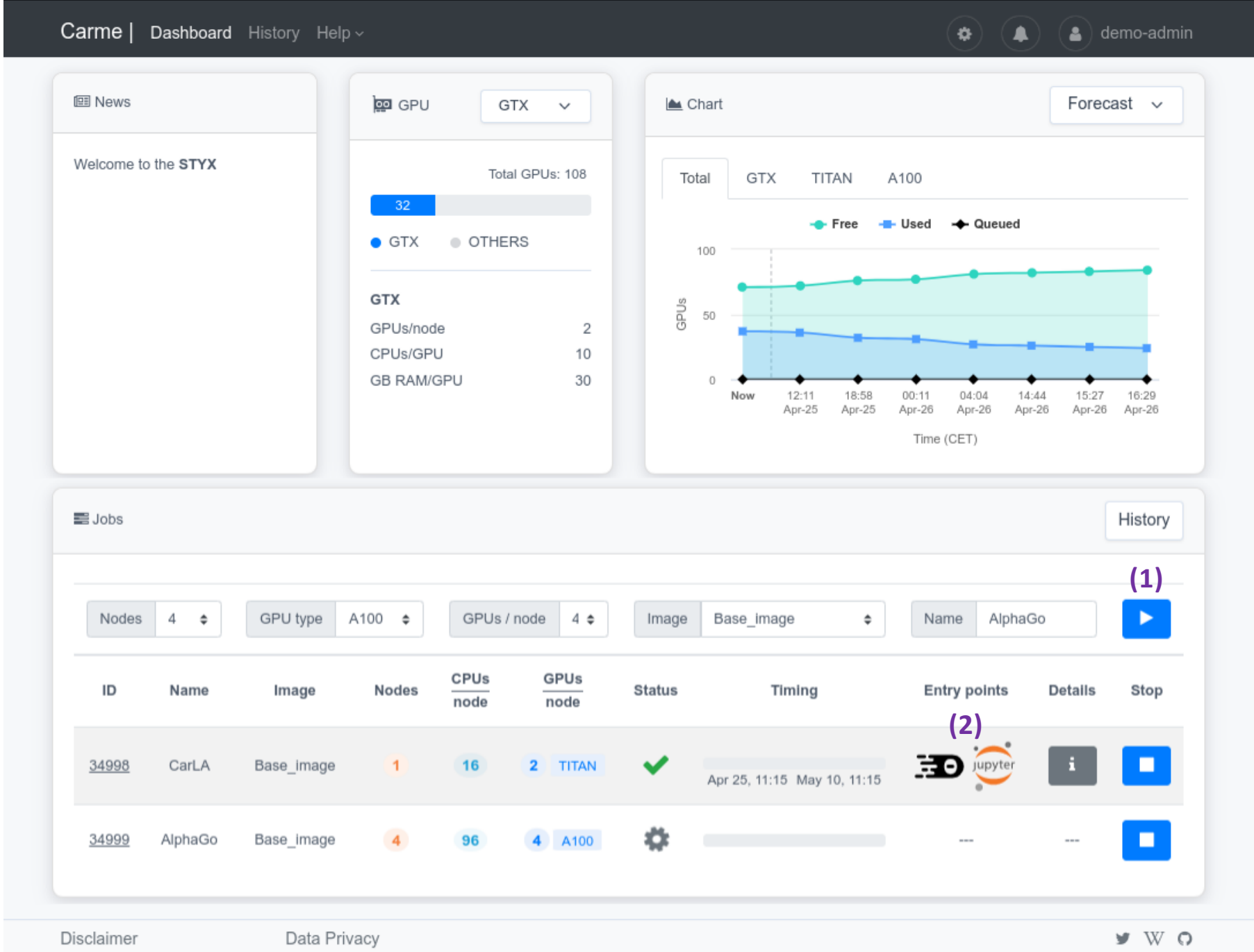
Interactive Usage

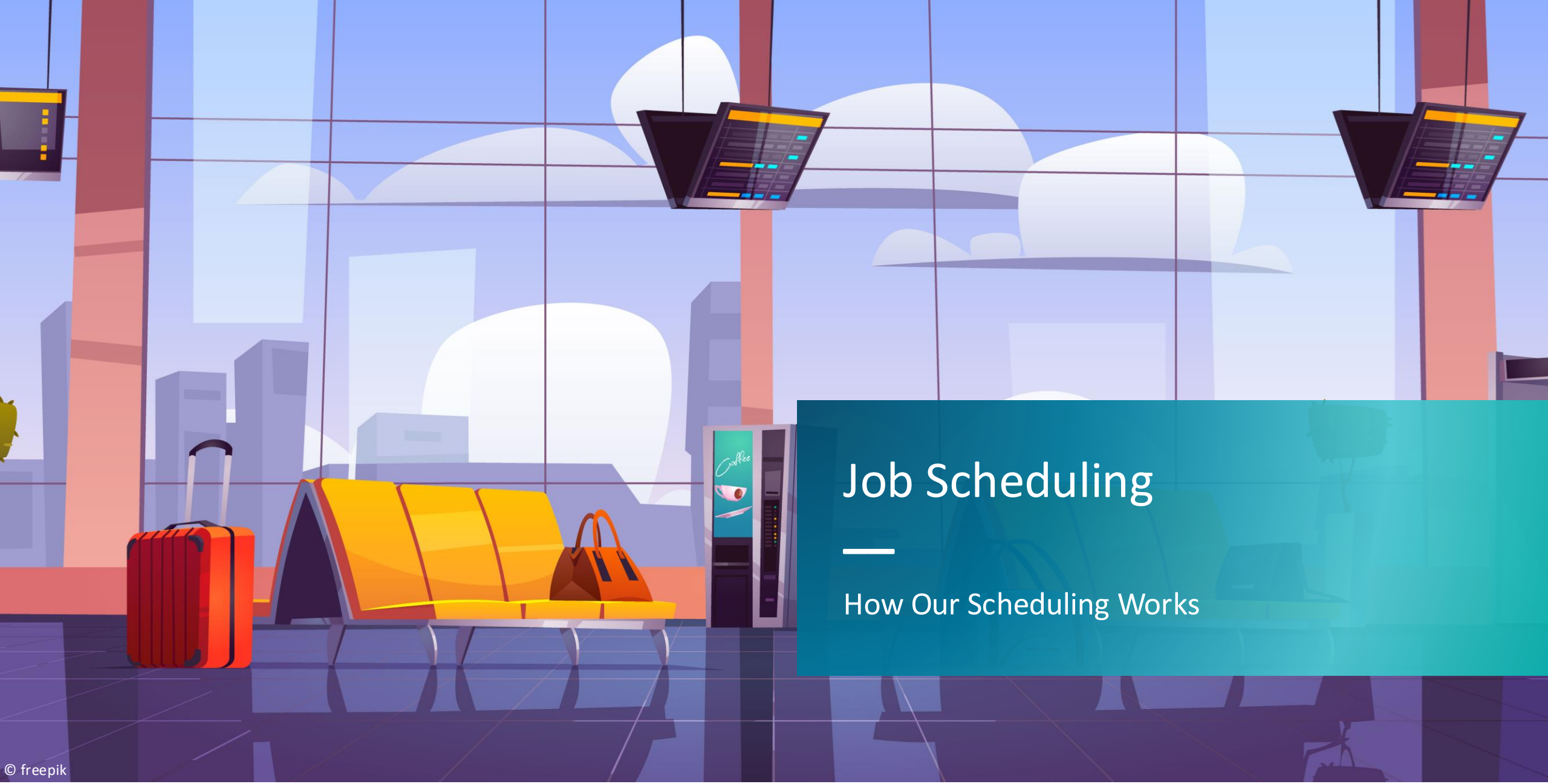
The User Control Center

two clicks to development

(1) start a job

(2) select your preferred development environment

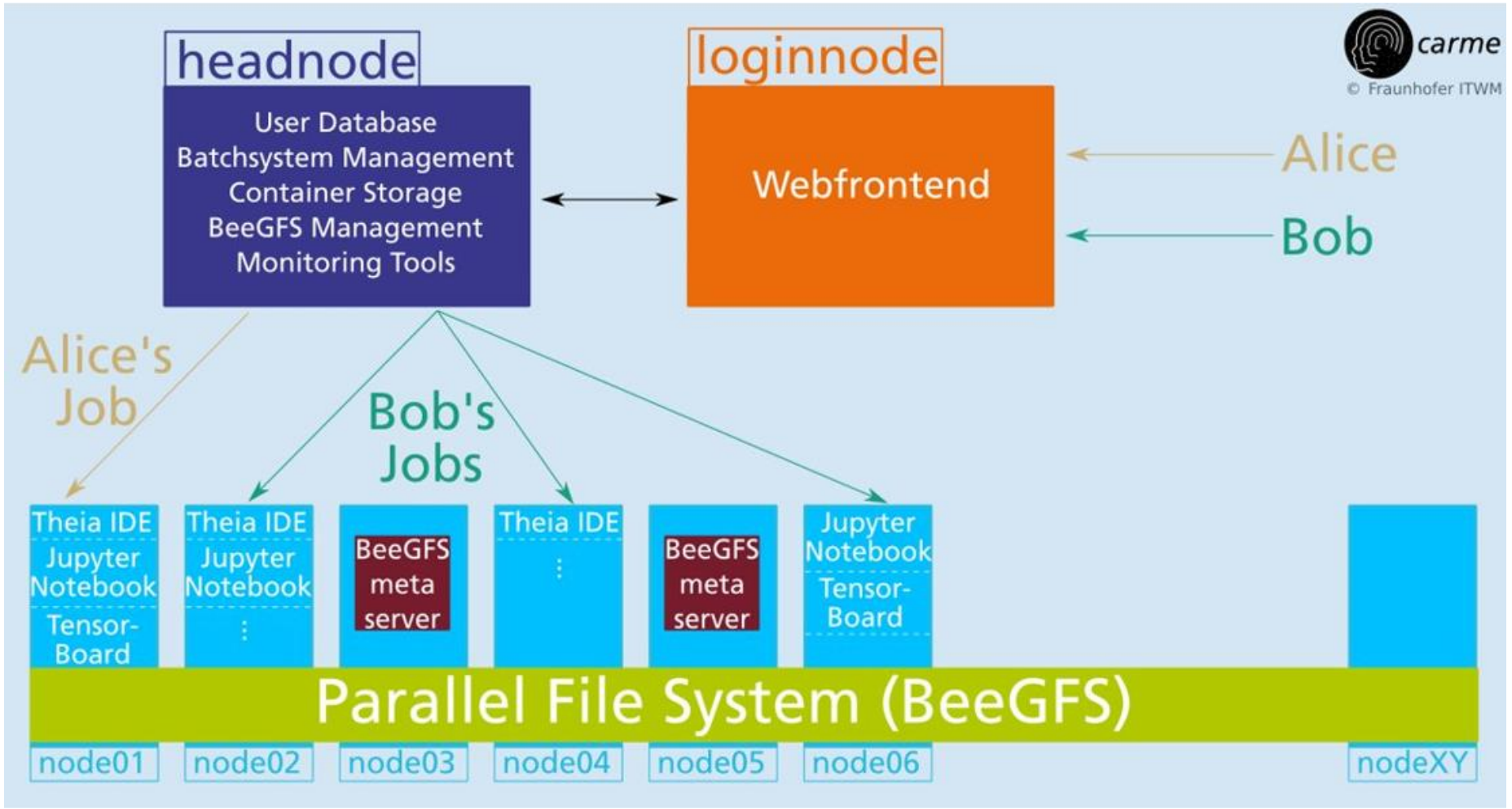




Job Scheduling

How Our Scheduling Works

Job Submission Scheme



Job Submission Scheme

- user login to the web interface
- user submits a job
  - request for new job is sent from the frontend to our HPC-backend
  - secure connection via rpyc and special keys
- our HPC-backend
  - gets the job information from the frontend and our data base
  - finally submits the job to the scheduler as original user
- job start script is handed over to the scheduler
  - scheduler checks handed over information (resource limits, user accounts, availability, …)
  - respective CARME prolog scripts run (only if the job belongs to us)
- job starts on the requested node(s)
- user is informed in the frontend that the job has successfully started


© freepik

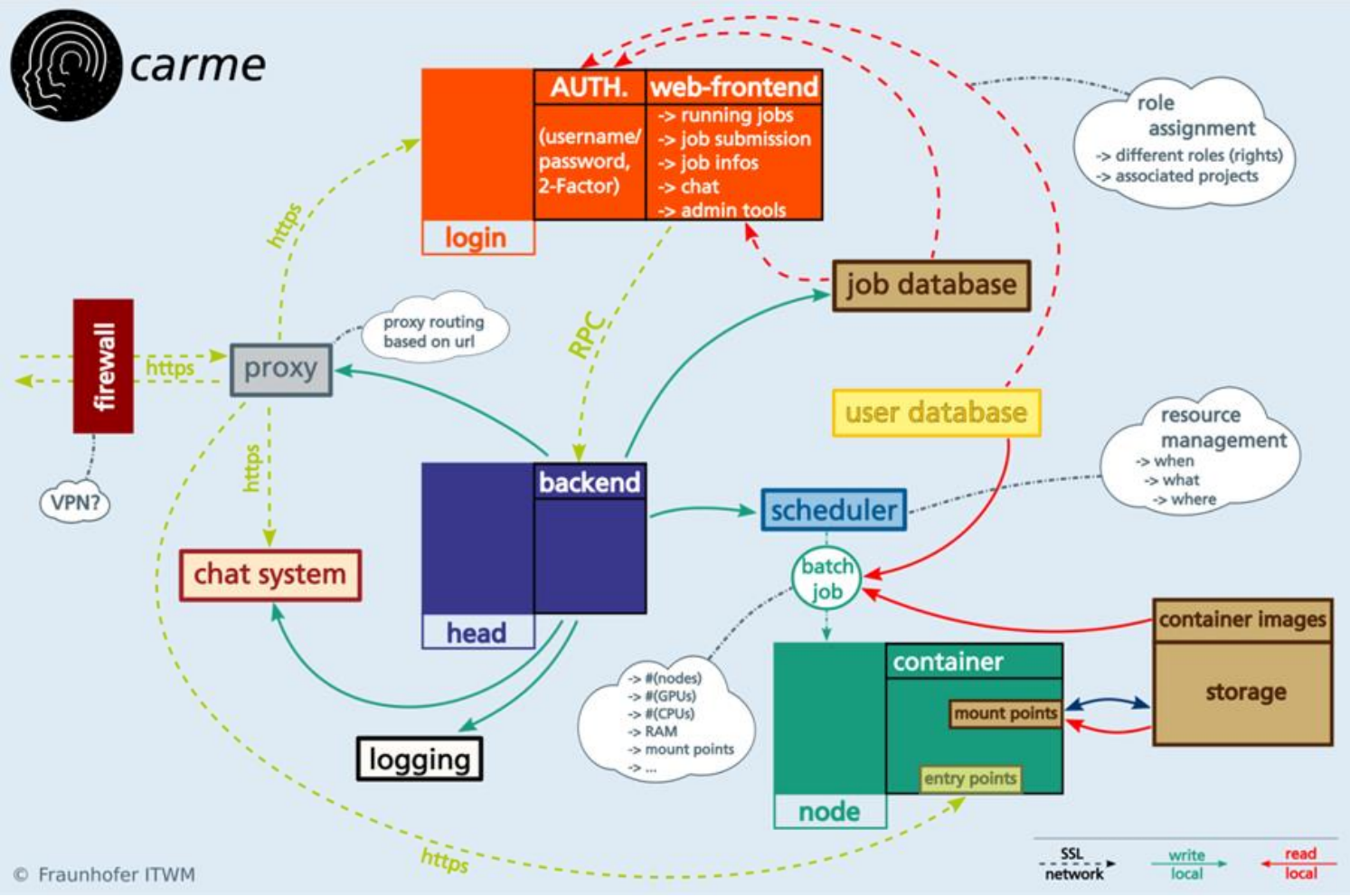
Security Concept

Our Security Idea

© freepik

Security Concept

Our Security Idea



Security Concept
Our Security Idea

- everything can run **behind your firewall**
- we utilize our **own proxy** and **frontend**
  (both are separated singularity containers)
- **secure connection** between frontend and backend
- access **requires authentication** and **2FA**
- define what is **mounted inside** a **job**
- fully **integrate** the **limitations** of your **scheduler**
- **additional limitations** in CARME
  (e.g., resource limits, running jobs and singularity images)
- everything **inside** a **job** runs in its **own namespace**
  (no chance to use other resources)

© freepik

Our Key Features

What We Have for You

© freepik / Fraunhofer ITWM

Our Key Features
In a Nutshell

- **seamless integration** into existing HPC setups
- user-friendly **web-interface** providing **flexible** and **os-independent** access from anywhere in the world
- **interactive jobs** to develop directly **on the cluster** with your favorite **deep learning tools**
- fully **separated jobs** with **custom resources**
- **intuitive abstraction** of complex cluster topologies
- **distributed multi-node/multi-gpu jobs** with direct access to GPI, GPI-Space, MPI, Tarantella and Horovod
- user maintained and **containerized environments** using **singularity** and **anaconda**

Our Key Features
In More Detail

- **seamless integration with available HPC tools**
  - job scheduling via SLURM
  - native LDAP support for user authentication
  - integrate existing distributed file systems like BeeGFS
- **access via web-interface**
  - OS independent (only web browser needed)
  - full user information (running jobs, cluster usage, news, messages)
  - start / stop jobs within the web-interface
- **interactive jobs**
  - flexible access to GPUs
  - access via web driven GUIs (TheiaIDE, JupyterLab)
- **distributed multi-node and/or multi-gpu jobs**
  - easy and intuitive job scheduling
  - directly use GPI, GPI-Space, MPI, Tarantella and Horovod within the jobs
- **full control about accounting and resource management**
  - job scheduling according to user specific roles
  - compute resources are user exclusive
- **user maintained, containerized environments**
  - singularity containers (runs as normal user, GPU, Ethernet and Infiband support)
  - anaconda environments (easy updates, project/user specific environments)
  - built-in matching between GPU driver and ML/DL tools

In Summary

What You get

© freepik

In Summary
What You Get

- **interactive development** access to HPC systems
  (keep your workflow with Jupyter Notebooks or full IDE)
- **easy** way to **submit** and **get jobs** on big clusters
  (simpified submission mask, no need to write batch scripts)
- guarantee that **only you** have **access** to **your resources**
  (no need to worry about other users)
- utilize **containers** and **conda environments** to get software
  (keep track with quickly chaning and diverse tool requirements)
- **access** your **HPC cluster** via a **web browser**
  (no need to install special or additional software)

In Summary
What You Get

The Administrator Side

- **easily integrate** CARME in your scheduler set up
  (a feature flag and our prolog/epilog scripts is all you need)
- **control** which **folders** user have access to
  (you can decide per image which folders are mounted or not)
- **provide** the **basic software** that ML/DL users need to do their work
  (keep the host installation small and clean)
- manage our **data base via** a simple **web interface**
- **simplify** the **support** for AI and Data Science users
- get **logs** and **error outputs** for CARME scripts and services

In Summary
What You Get

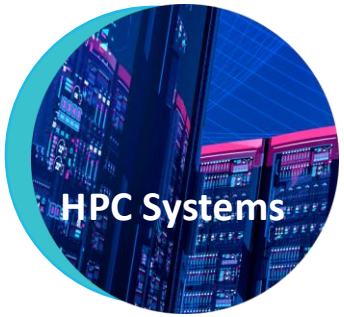
The Institution Side

- CARME was always and will be **developed in Germany**
  (following Fraunhofer standards and respecting data privacy)
- **license** that allows **commercial use**
- giving users a tool to **easily utilize GPU resources**
  (be sure that your investments in hardware pay off)
- **increase** the **productivitiy** of both your AI developers and researchers and your system administrators

Bringing AI Development / Education and HPC Systems Together

The CARME Way

Carme – an open-source framework to manage resources for multiple users running interactive jobs on an HPC Clusters



AI and Data Science Tools

- Code-Server
- JupyterLab
- Anaconda Environments (mambaforge/miniconda)

HPC Tools

- SLURM
- LDAP
- Singularity
- BeeGFS
- Zabbix

Open-Source Tools

enables

AI Development

- develop code
- schedule & track experiments
- debug model performance

AI Education

- hands-on sessions
- slides & homework
- easy Q&A

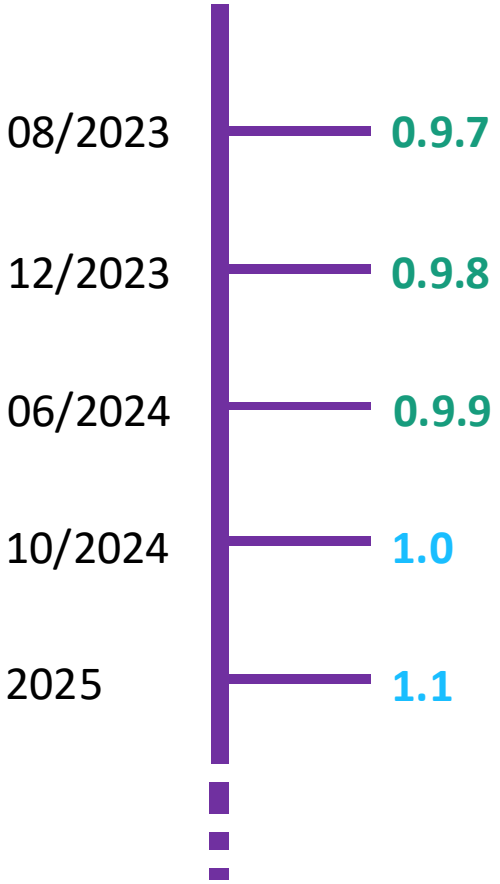
https://carmeteam.github.io/Carme/

Roadmap

Where to Go

Roadmap
Where to Go

- available **on GitHub** (https://carmeteam.github.io/Carme/)
- **Easy installation script** (including support)
- release cycle: **6–9 months**
- latest release: **r0.9.9** (06/2024)
- next release: **r1.0** (10/2024)



Contact

Ph.D. Christian Ortiz
Competence Center High Performance Computing
Fraunhofer ITWM
Fraunhofer-Platz 1
D-67663 Kaiserslautern, GERMANY
Tel. +49 631 31600 4134
christian.ortiz@itwm.fraunhofer.de

Fraunhofer
ITWM