Run:ai

Effiziente KI-Nutzung für Lehre und Forschung

Präsentation: Ferhat Basagac
Im Rahmen des Projekts ZAKKI mit
Prof. Dr.-Ing. Sebastian von Enzberg & David Döring

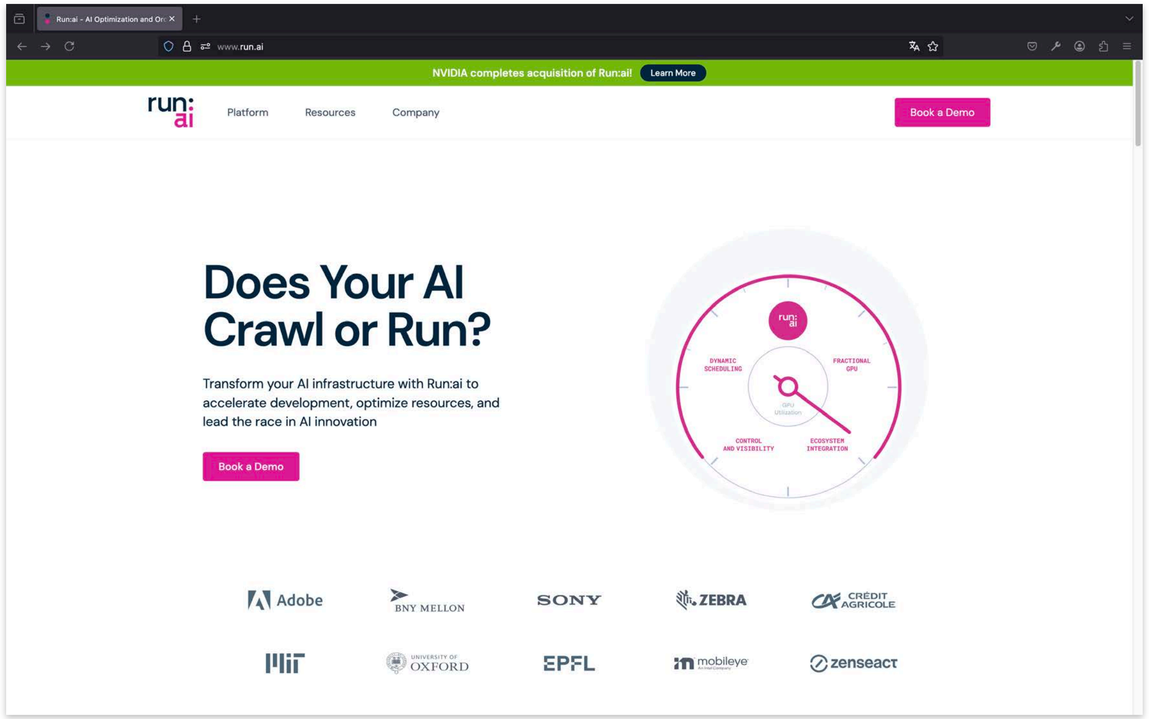
Mittwoch, 19.03.2025 // Hybrid // Lehr-Angebote
auf KITT - der KI-Infrastruktur der h2

Inhaltsverzeichnis

Run:ai ist eine Plattform zur Optimierung und Orchestrierung von KI-Infrastrukturen.

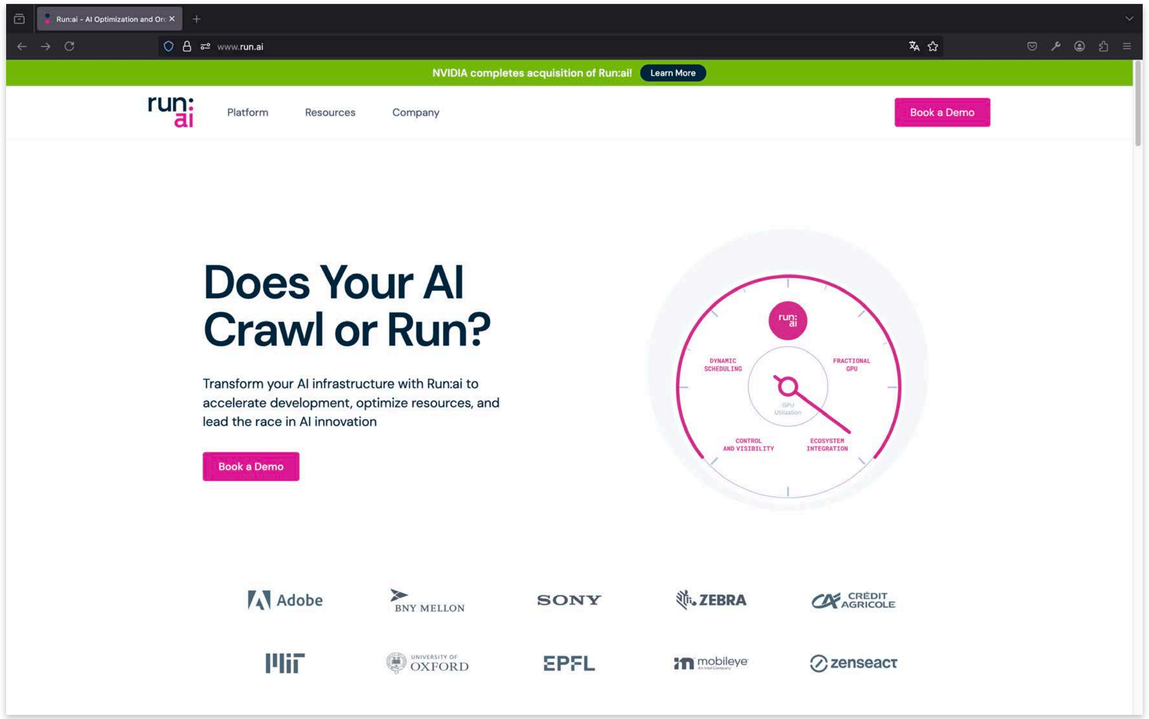
- Run:ai ermöglicht Unternehmen und Forschungseinrichtungen die effiziente Verwaltung und Skalierung ihrer GPU-Ressourcen.

👉 **Ziel ist die Beschleunigung der Entwicklung und Implementierung von KI-Anwendungen.**



Quelle: https://www.run.ai/

- **Gründung:** Run:ai wurde 2018 von Omri Geller und Ronen Dar gegründet.
- **Unternehmenssitz:** Der Hauptsitz des Unternehmens befindet sich in Tel Aviv, Israel.
- **Übernahme durch Nvidia:** Im Dezember 2024 wurde Run:ai von Nvidia für 700 Millionen US-Dollar übernommen.



Quelle: https://www.run.ai/blog/run-ai-joins-nvidia, https://www.heise.de/news/Nvidia-uebernimmt-KI-Softwarefirma-Run-ai-und-plant-Offenlegung-der-Software-10222071.html

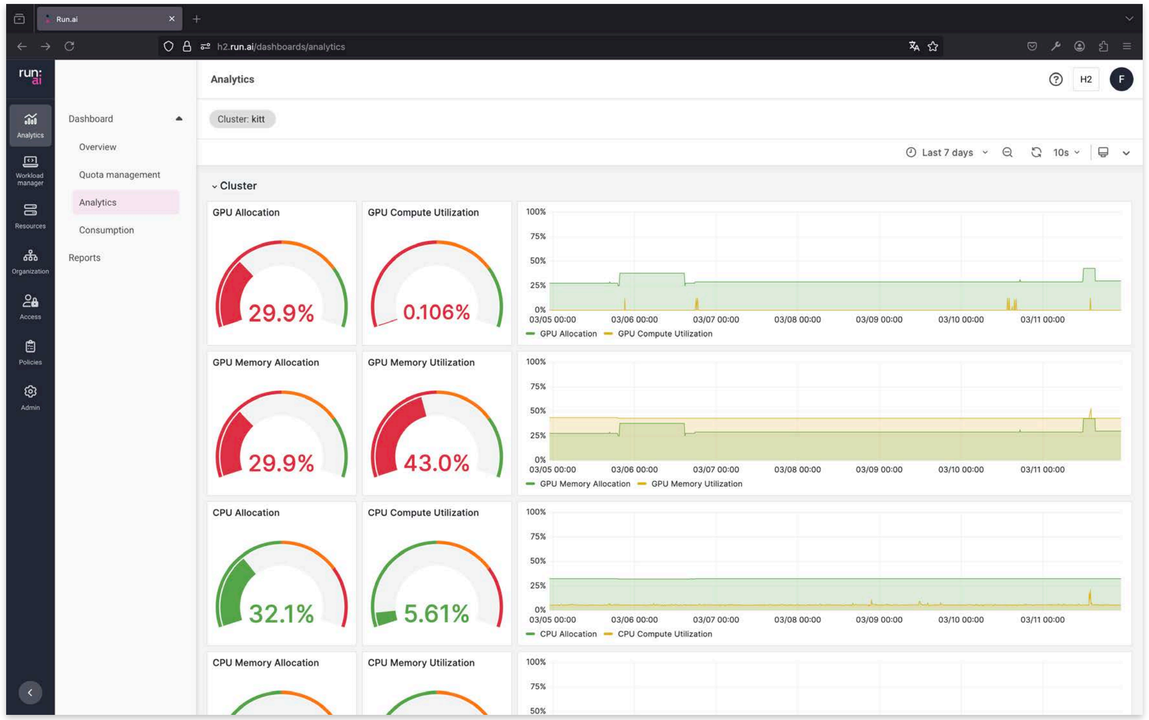
Run:ai bietet Funktionen wie:

1. **Dynamisches Scheduling:** Flexible Zuweisung von Aufgaben oder Ressourcen in Echtzeit, basierend auf der aktuellen Systemauslastung und den Prioritäten.

2. **GPU-Pooling**: Zusammenfassen mehrerer GPUs zu einem gemeinsamen Ressourcenpool, sodass verschiedene Workloads effizient darauf zugreifen können.

3. **GPU-Fractioning:** Aufteilen einer einzelnen GPU in mehrere virtuelle Einheiten, sodass mehrere Prozesse oder Nutzer gleichzeitig dieselbe GPU nutzen können.

👉 **Optimale Auslastung der vorhandenen Hardware.**



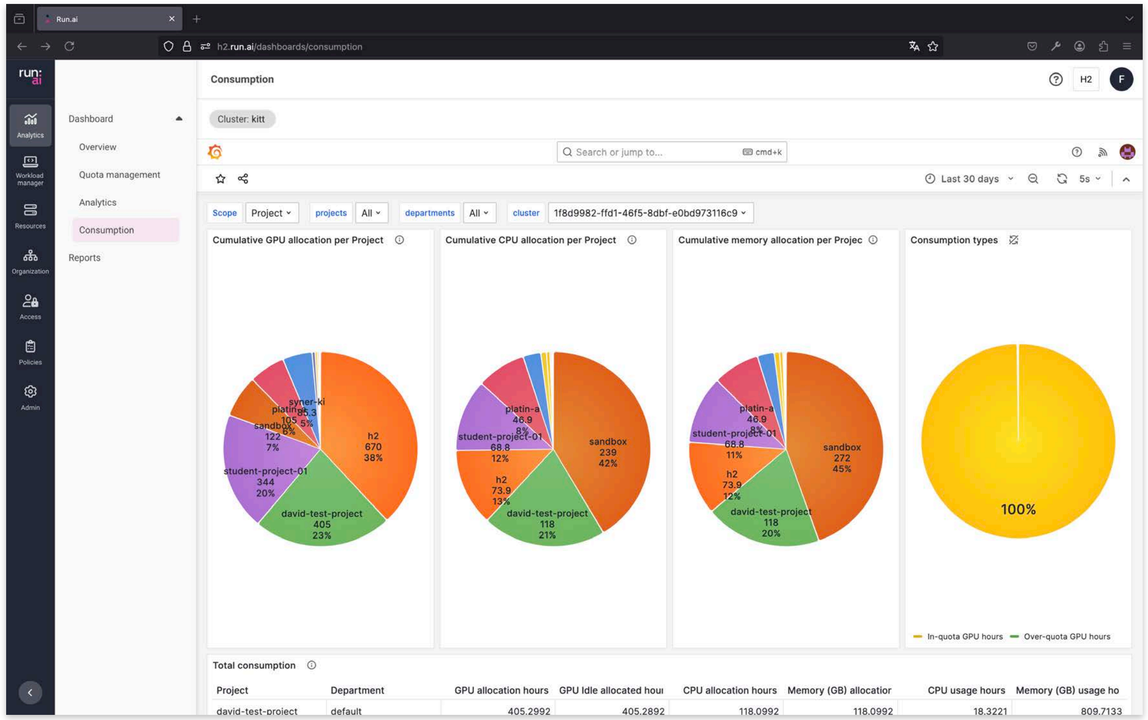
Quelle: https://www.run.ai/, https://www.run.ai/blog/maximize-the-potential-of-your-gpus-a-guide-to-dynamic-gpu-fractions-node-level-scheduler

In Bildungs- und Forschungseinrichtungen, in denen oft begrenzte Ressourcen auf viele Projekte verteilt werden müssen, bietet Run:ai signifikante Vorteile:

1. **Effiziente Ressourcennutzung:** Durch die dynamische Zuweisung von GPU-Ressourcen können mehrere Projekte gleichzeitig durchgeführt werden, ohne dass es zu Engpässen kommt.

2. **Kosteneffizienz:** Eine optimierte Auslastung der vorhandenen Hardware reduziert den Bedarf an zusätzlichen Investitionen in neue Geräte.

3. **Verbesserte Forschungsproduktivität:** Schnellere Rechenzeiten ermöglichen es Forschern, mehr Experimente in kürzerer Zeit durchzuführen, was den Innovationsprozess beschleunigt.

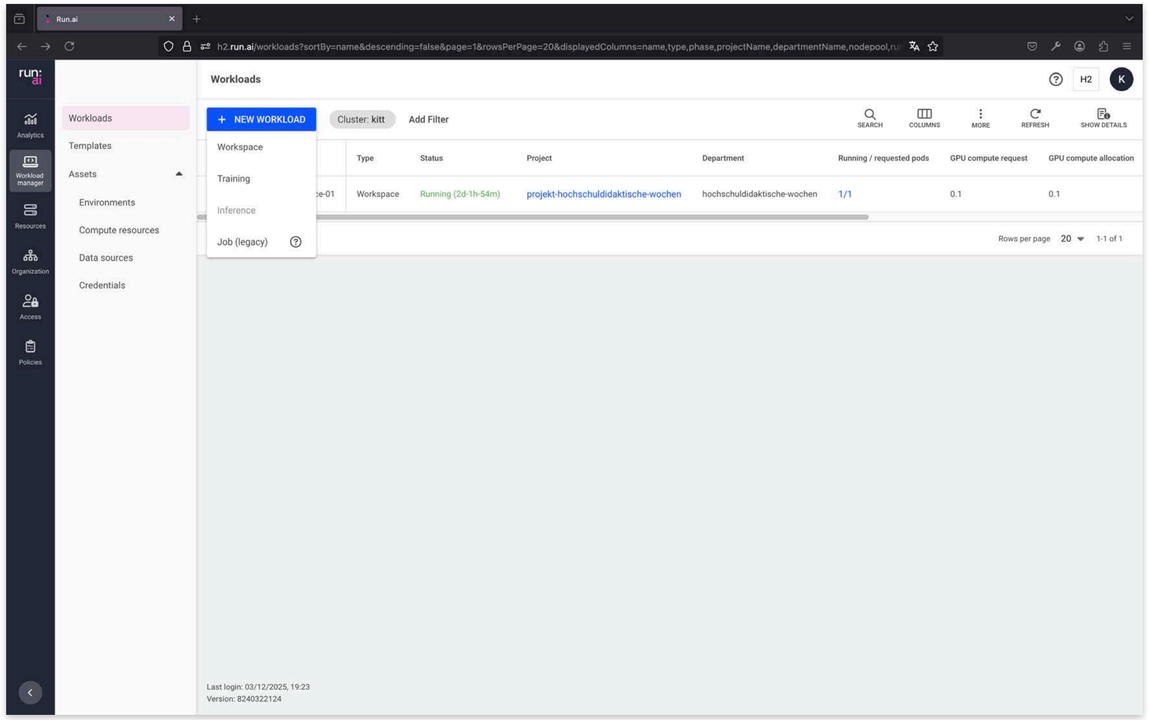


Quelle: https://h2.run.ai/

Run:ai optimiert die Nutzung von GPU-Ressourcen in verschiedenen Bereichen des Machine Learning (ML) und Deep Learning (DL):

1. **Modelltraining:** Effiziente Ressourcenzuweisung → schnellere Trainingszeiten (Reduziert Engpässe bei großen Modellen)

👉 Ohne Run:ai muss die Zuweisung von GPUs manuell oder mit statischen Regeln erfolgen.

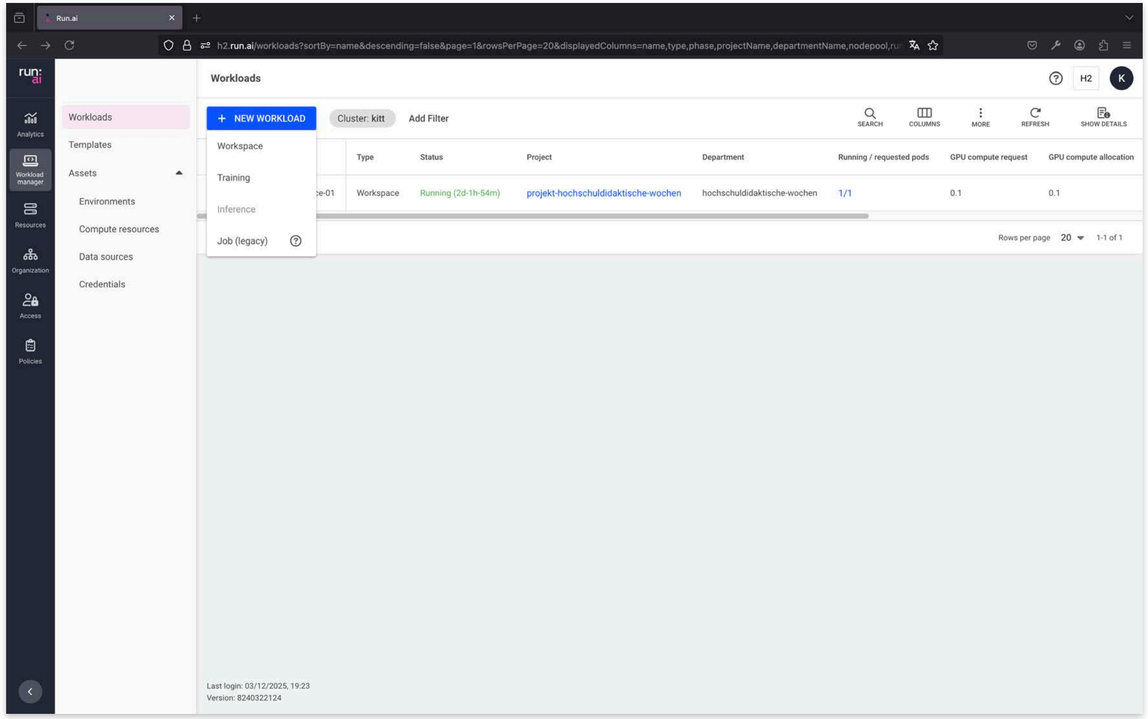


Quelle: https://www.run.ai/guides/cloud-deep-learning/ai-inference, https://www.run.ai/guides/multi-gpu/automate-hyperparameter-tuning-across-multiple-gpu

Run:ai optimiert die Nutzung von GPU-Ressourcen in verschiedenen Bereichen des Machine Learning (ML) und Deep Learning (DL):

2. **Hyperparameter-Tuning:** Automatisierte Tests für optimale Parameter (Keine manuellen Anpassungen nötig)

👉 Ohne Run:ai müsste man Tests manuell ausführen oder mit Skripten automatisieren.

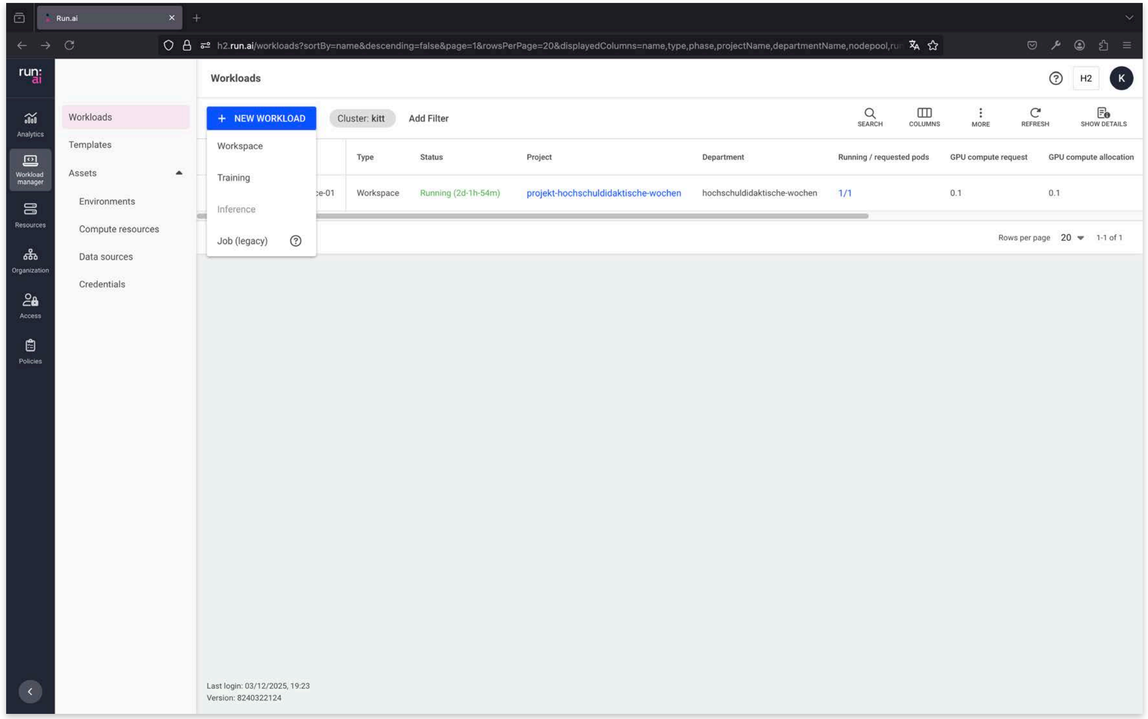


Quelle: https://www.run.ai/guides/cloud-deep-learning/ai-inference, https://www.run.ai/guides/multi-gpu/automate-hyperparameter-tuning-across-multiple-gpu

Run:ai optimiert die Nutzung von GPU-Ressourcen in verschiedenen Bereichen des Machine Learning (ML) und Deep Learning (DL):

3. **Inference:** Optimierte Lastverteilung → schnellere Verarbeitung (Zuverlässige Nutzung für neue Daten)

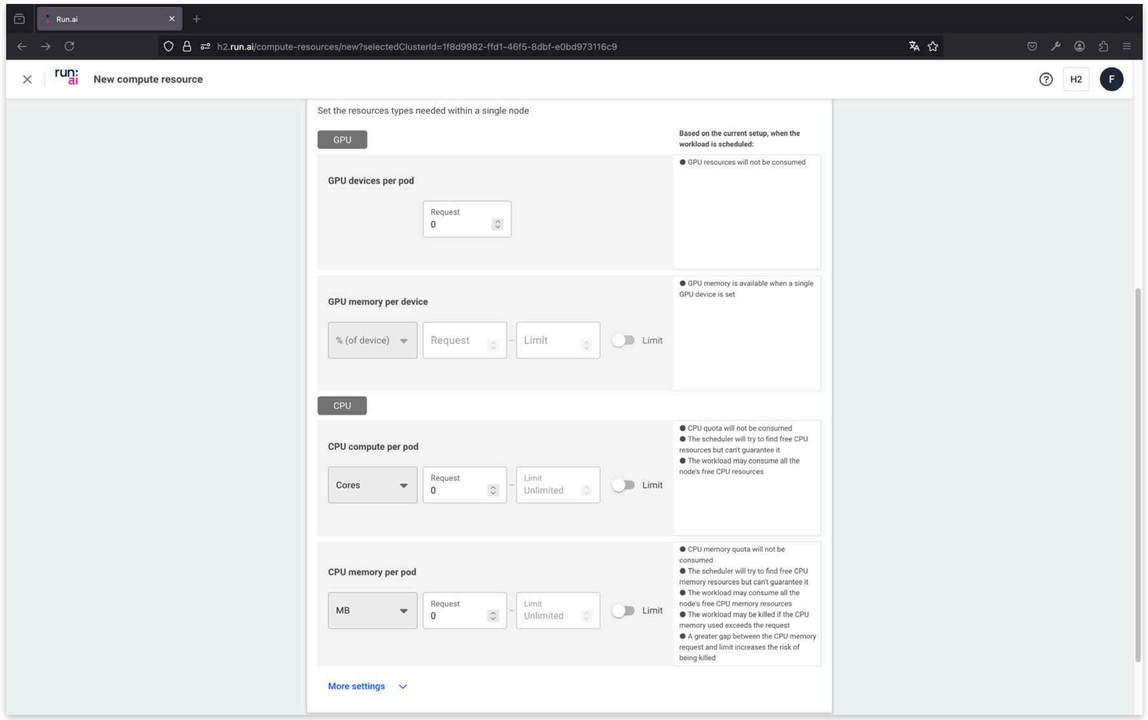
👉 Ohne Run:ai kann die GPU-Auslastung bei Inference unausgeglichen sein (z. B. Überlastung einer GPU, während andere ungenutzt bleiben).



Quelle: https://www.run.ai/guides/cloud-deep-learning/ai-inference, https://www.run.ai/guides/multi-gpu/automate-hyperparameter-tuning-across-multiple-gpu

1. **Kontinuierliche Überwachung des Ressourcenverbrauchs:** Laufende Analyse von GPU-Auslastung, Speicherverbrauch und Job-Wartezeiten.

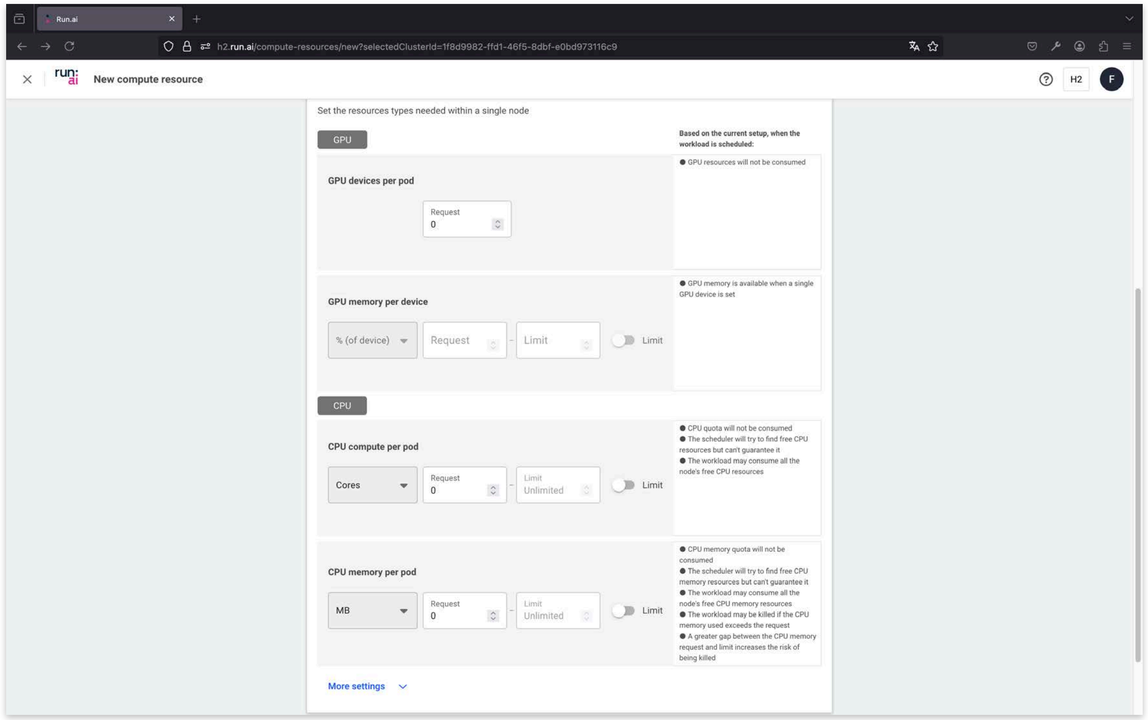
👉 Vorteile: Frühzeitige Erkennung von Engpässen und Sicherstellung einer optimalen Ressourcennutzung.



Quelle: https://www.run.ai/gpu-optimization

2. **Dynamische GPU-Zuweisung & Priorisierung von Workloads:** Ein intelligenter Scheduler (Systemkomponente, die Aufgaben zeitlich plant und Ressourcen zuteilt) verteilt GPUs bedarfsgerecht und priorisiert kritische Workloads.

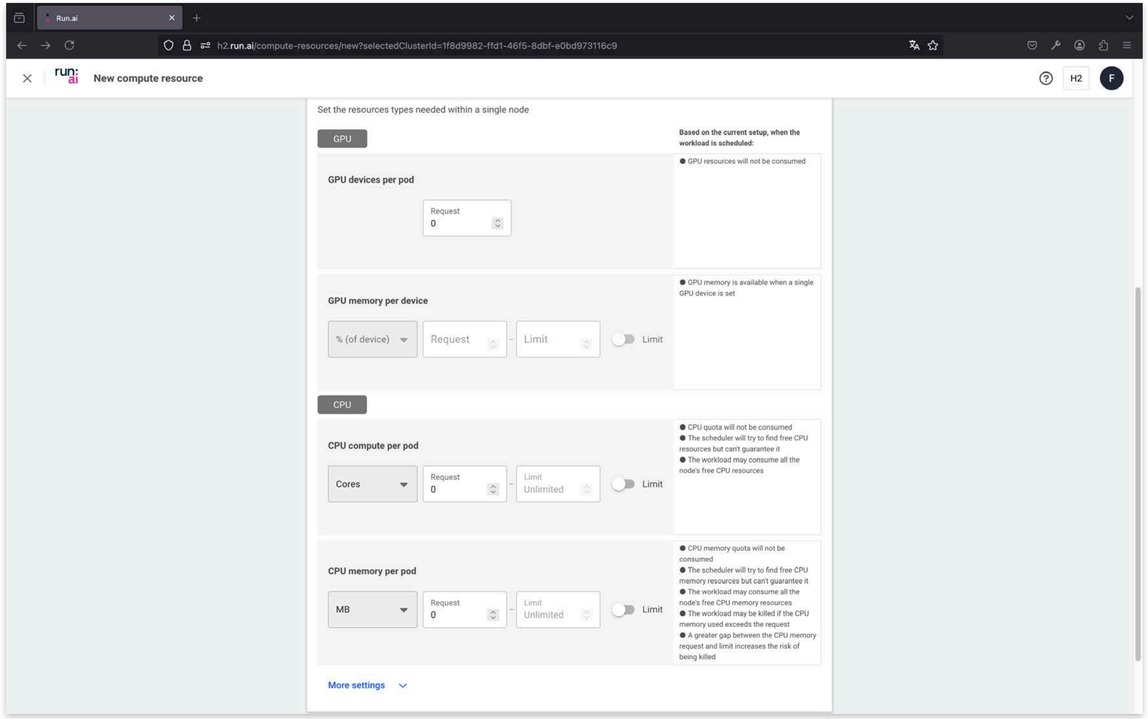
👉 Vorteile: Maximierung der Effizienz durch Anpassung an aktuelle Anforderungen.



Quelle: https://www.run.ai/gpu-optimization

3. **Maximale Hardware-Auslastung durch GPU-Sharing:** Mehrere Jobs teilen sich GPUs, wodurch Leerlaufzeiten minimiert werden.

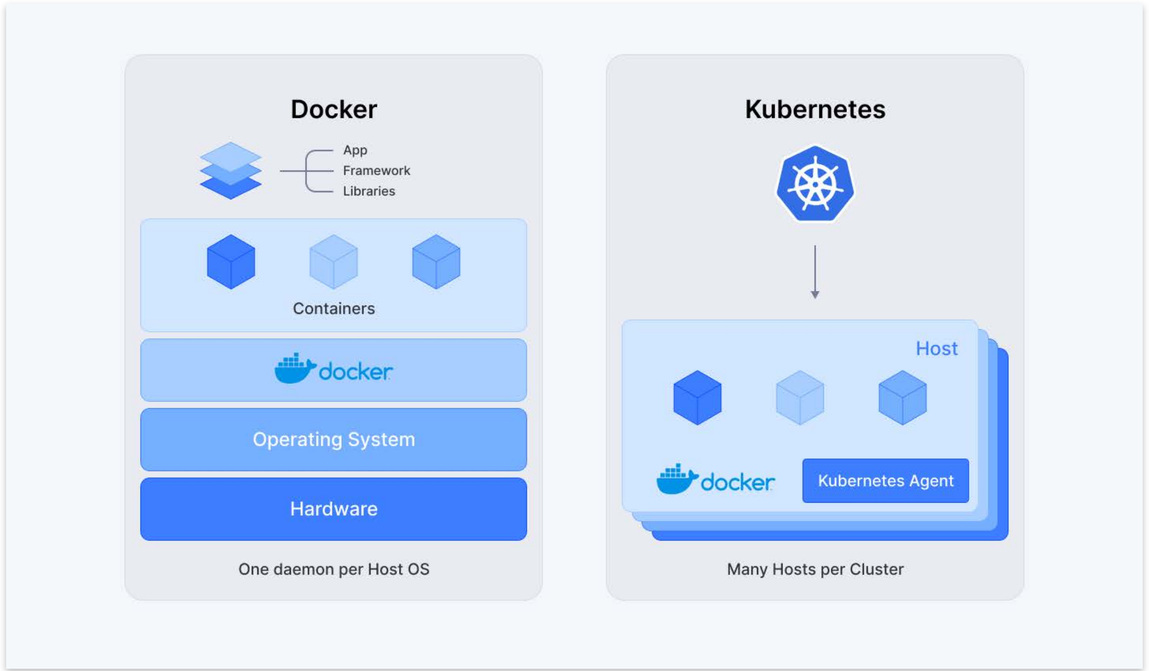
👉 Vorteile: Erhöhte Auslastung der Hardware und verbesserte Ressourceneffizienz.



Quelle: https://www.run.ai/gpu-optimization

- **Einsatz von Container-Technologie (Docker):** KI-Modelle und Anwendungen werden in Containern isoliert, wodurch sie unabhängig von der zugrunde liegenden Hardware und Software-Umgebung betrieben werden können. Dies erleichtert das Deployment und verbessert die Skalierbarkeit.

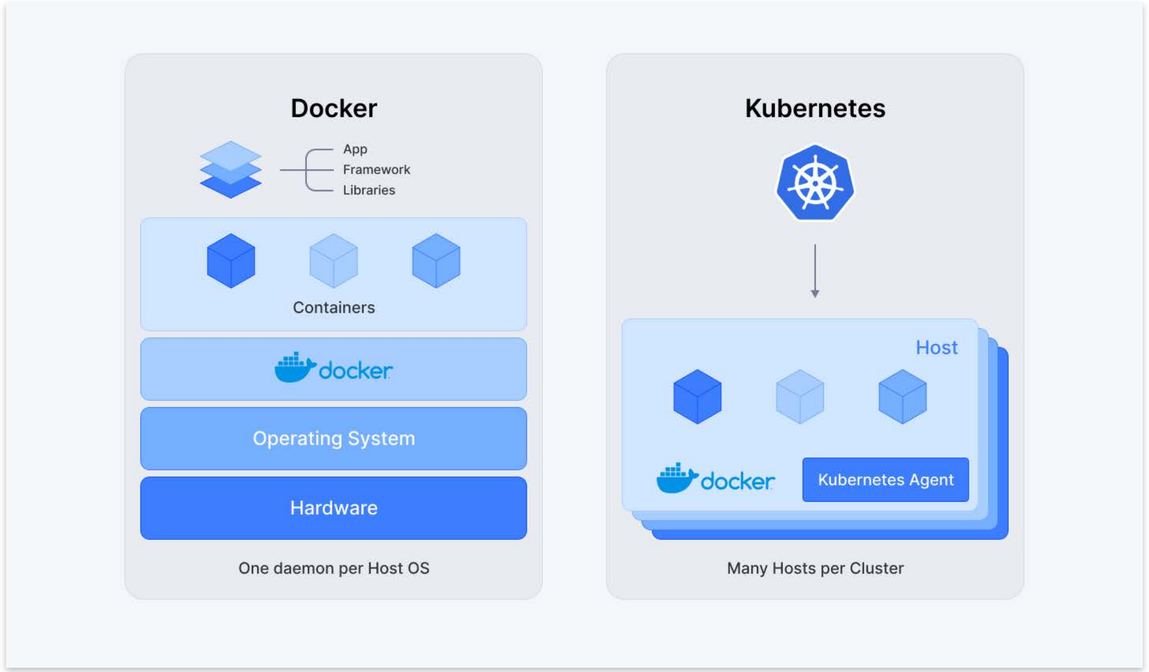
👉 Docker ermöglicht es, Anwendungen in Containern auszuführen, die unabhängig von der Hardware oder dem Betriebssystem sind.



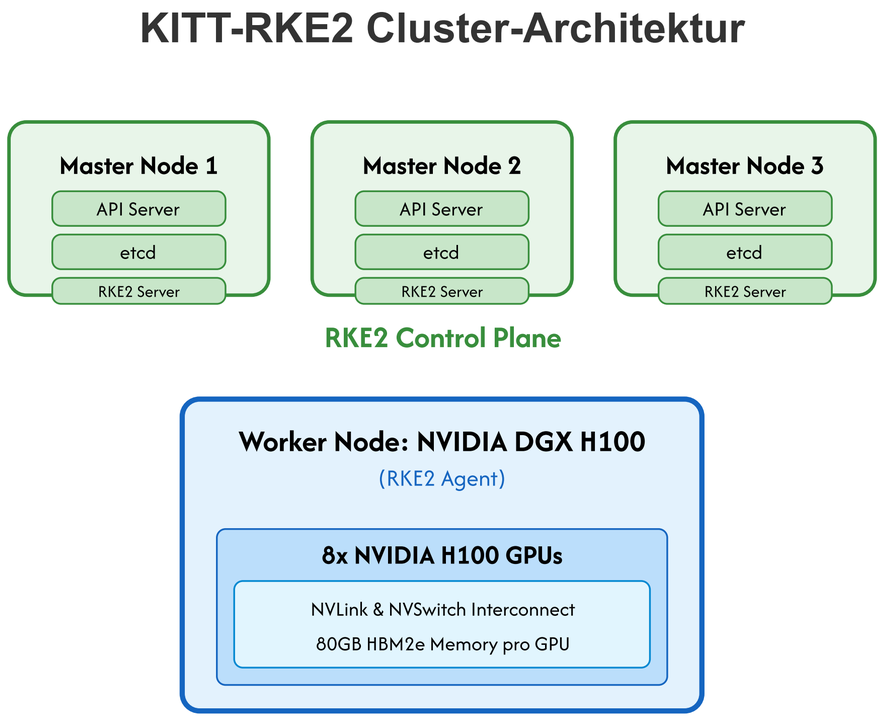
Quelle: https://aws.amazon.com/de/compare/the-difference-between-kubernetes-and-docker/, https://www.index.dev/blog/kubernetes-vs-docker

- **Kubernetes-Integration:** Run:ai integriert sich nahtlos in Kubernetes, um die Bereitstellung, Skalierung und Verwaltung von GPU-gestützten Workloads zu automatisieren. Als Plug-in für Kubernetes erfordert der Run:ai-Scheduler keine aufwendige Einrichtung und ist zertifiziert, sich in verschiedene Kubernetes-Varianten zu integrieren.

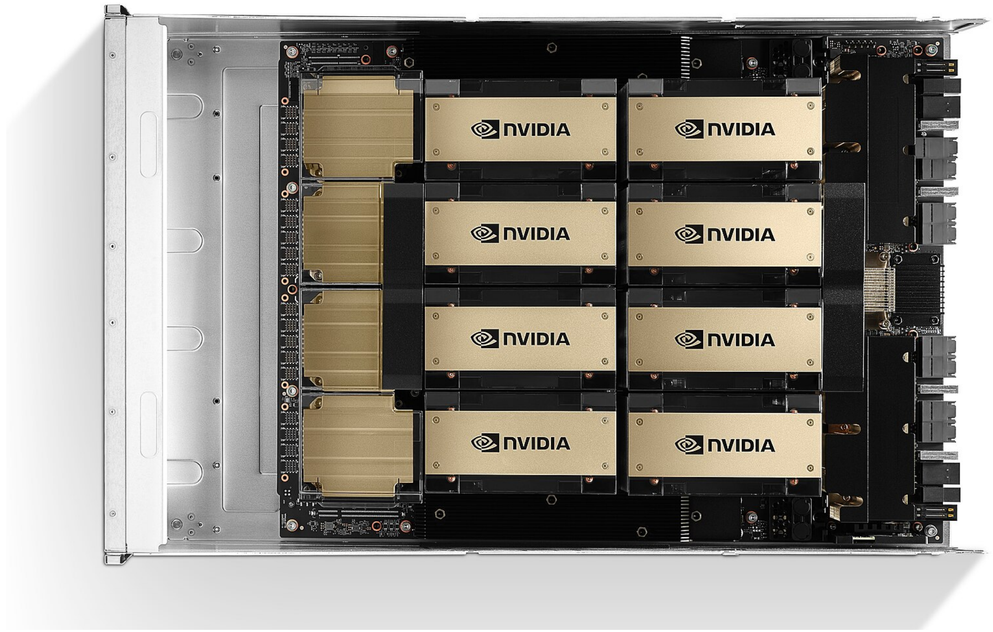
👉 Kubernetes orchestriert und verwaltet Docker-Container über mehrere Server hinweg und sorgt für Skalierung und Lastverteilung.



Quelle: https://aws.amazon.com/de/compare/the-difference-between-kubernetes-and-docker/, https://www.index.dev/blog/kubernetes-vs-docker



Der KITT-Kubernetes-Cluster besteht aus drei Master-Knoten und einem NVIDIA DGX H100 als Worker-Node, auf dem unter anderem auch Run:ai betrieben wird.



DGX H100 Top View: Hopper-Architektur für maximale KI- und HPC-Leistung.

Quelle: https://www.wikiwand.com/en/articles/Nvidia_DGX

Run:ai kann eine GPU in mehrere vGPUs aufteilen und diese verschiedenen Workloads zuweisen.

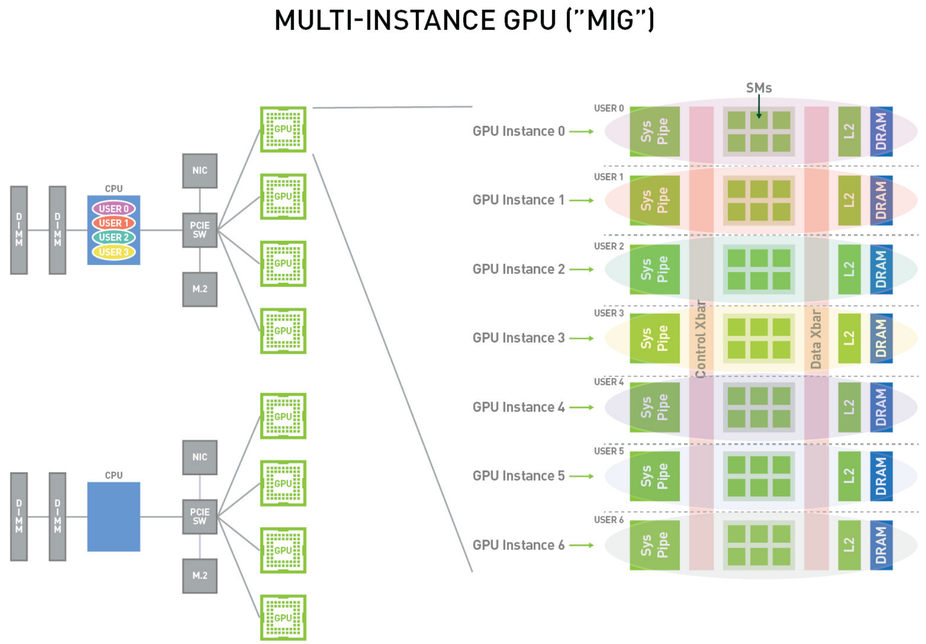
Was sind vGPUs?

- Eine **vGPU (virtuelle GPU)** ist eine logische Partition einer physischen GPU. Sie ermöglicht es, eine einzelne GPU auf mehrere Nutzer oder Workloads aufzuteilen.

Warum ist das nützlich?

- Ideal für Szenarien, in denen nicht jeder Job eine komplette GPU benötigt

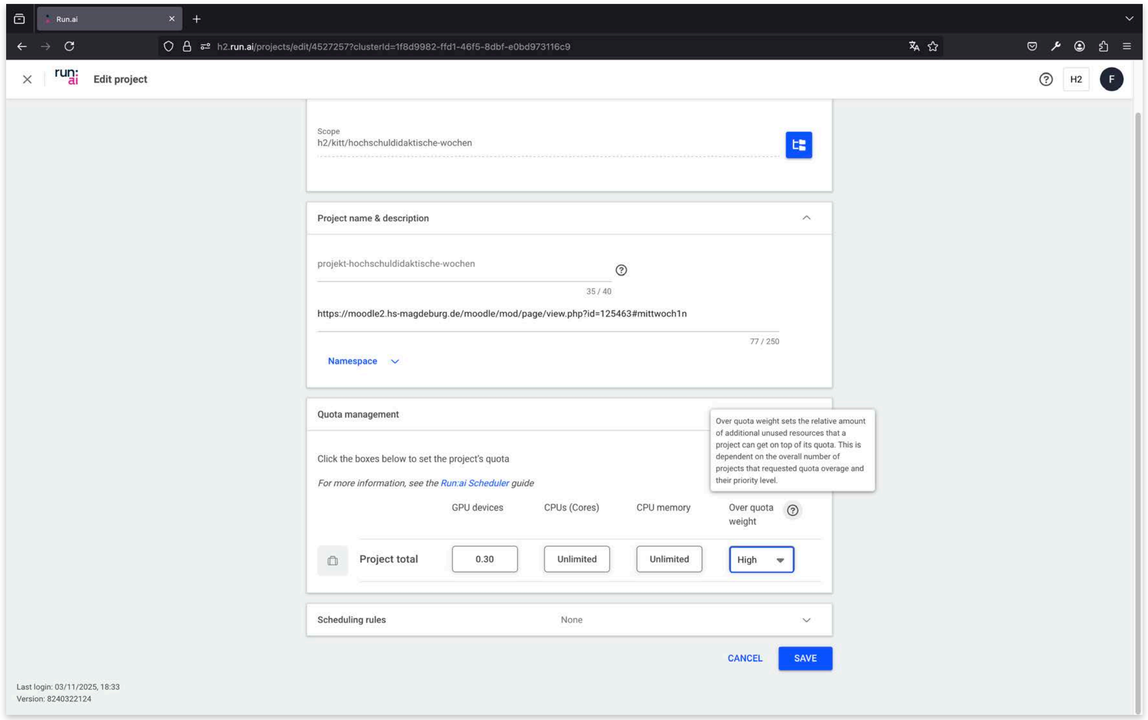
👉 **Effizientere Ressourcennutzung** und **bessere Skalierbarkeit**, da GPUs nicht mehr exklusiv blockiert werden.



Quelle: https://docs.nvidia.com/datacenter/tesla/mig-user-guide/

Wie verwaltet Run:ai vGPUs?

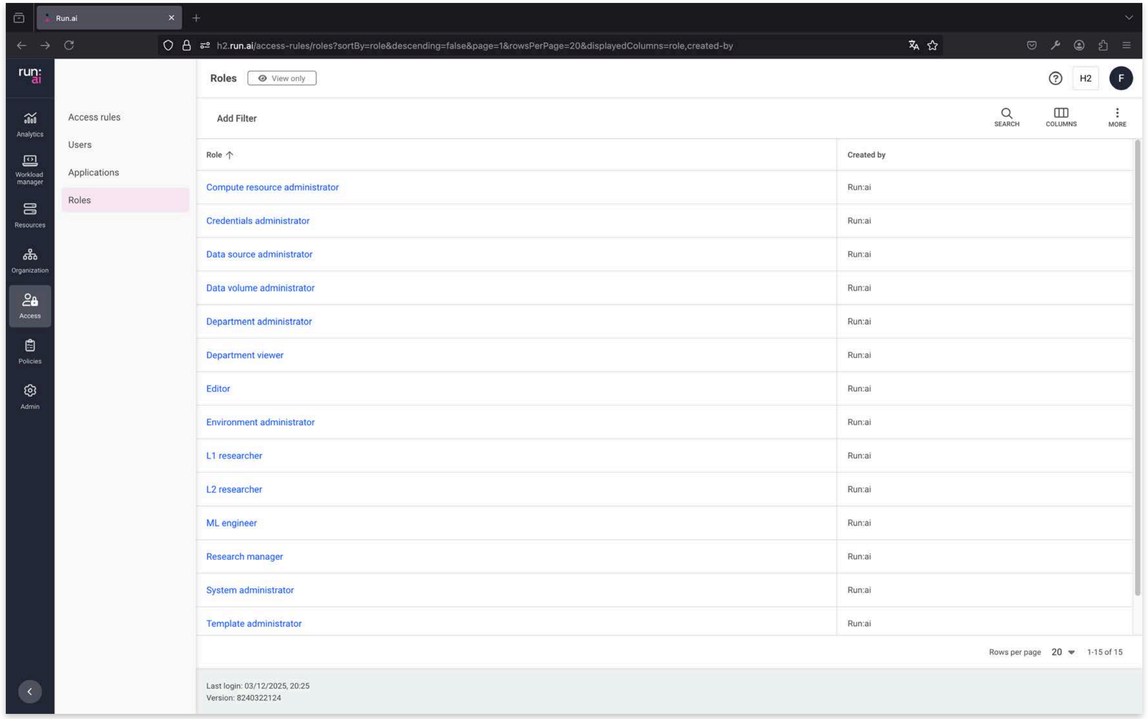
- **GPU-Sharing & Overcommitment:** Mehrere Nutzer oder Jobs teilen sich eine GPU, wobei priorisierte Aufgaben bevorzugt werden.
- **Automatische Ressourcenzuweisung:** Run:ai verteilt vGPUs dynamisch nach Prioritäten, Ressourcenbedarf und Verfügbarkeit.
- **Transparente Nutzungskontrolle:** Administratoren verwalten die GPU-Zuteilung und Priorisierung über das Dashboard oder APIs.



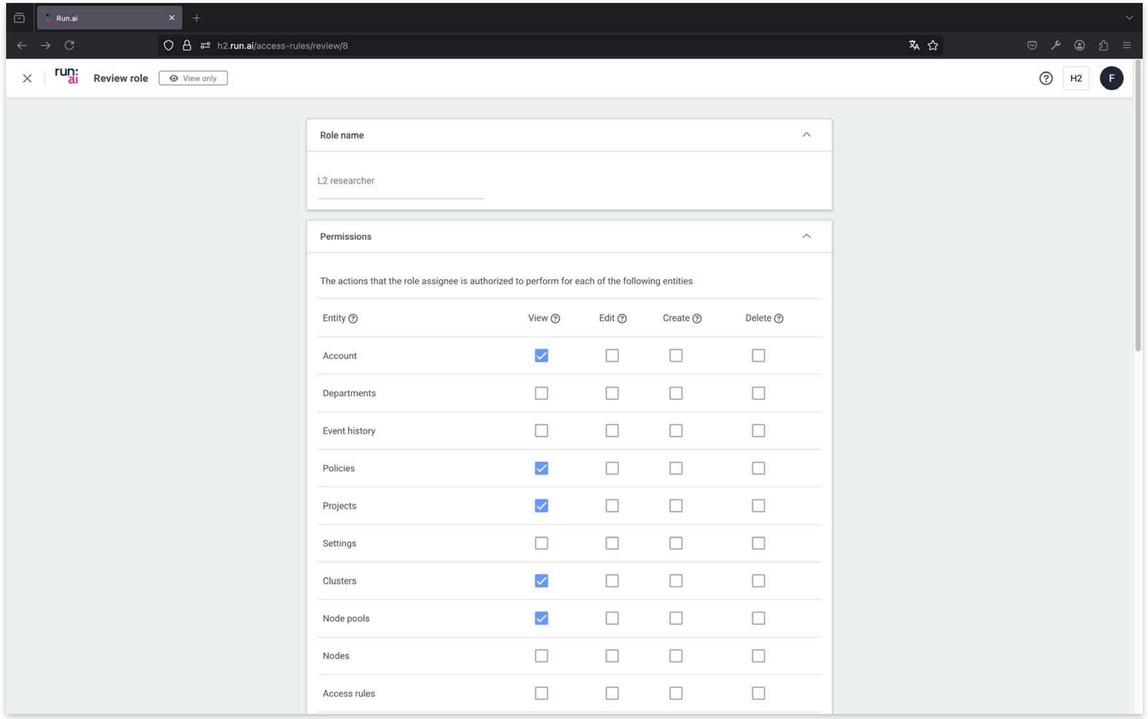
Quelle: https://www.run.ai/blog/runai-nvidia-mig-multi-instance-gpu-scheduling, https://www.run.ai/gpu-optimization

Kriterium	Klassische GPU-Zuweisung	Run:ai mit vGPUs
GPU-Zuweisung	Statisch (ein Job/Nutzer pro GPU)	Dynamisch (mehrere Jobs pro GPU)
Flexibilität	Wenig, GPU wird komplett blockiert	Hoch, GPUs werden je nach Bedarf geteilt
Effizienz	Oft suboptimal (z. B. 20% Nutzung, 80% Leerlauf)	Hohe Auslastung durch Sharing
Priorisierung	Keine Priorisierung möglich	Wichtige Jobs haben Vorrang
Skalierbarkeit	Begrenzte Anzahl an parallelen Nutzern	Mehr Nutzer können dieselben GPUs teilen

Quelle: https://www.run.ai/guides/multi-gpu/simplify-gpu-sharing-part-1, https://www.exxactcorp.com/blog/Deep-Learning/run-ai-you-ve-got-idle-gpus-we-guarantee-it



Die effiziente Nutzung von GPU-Ressourcen in gemeinsam genutzten Umgebungen erfordert eine klare Strukturierung von Rollen und Verantwortlichkeiten sowie eine gezielte Ressourcenverteilung.

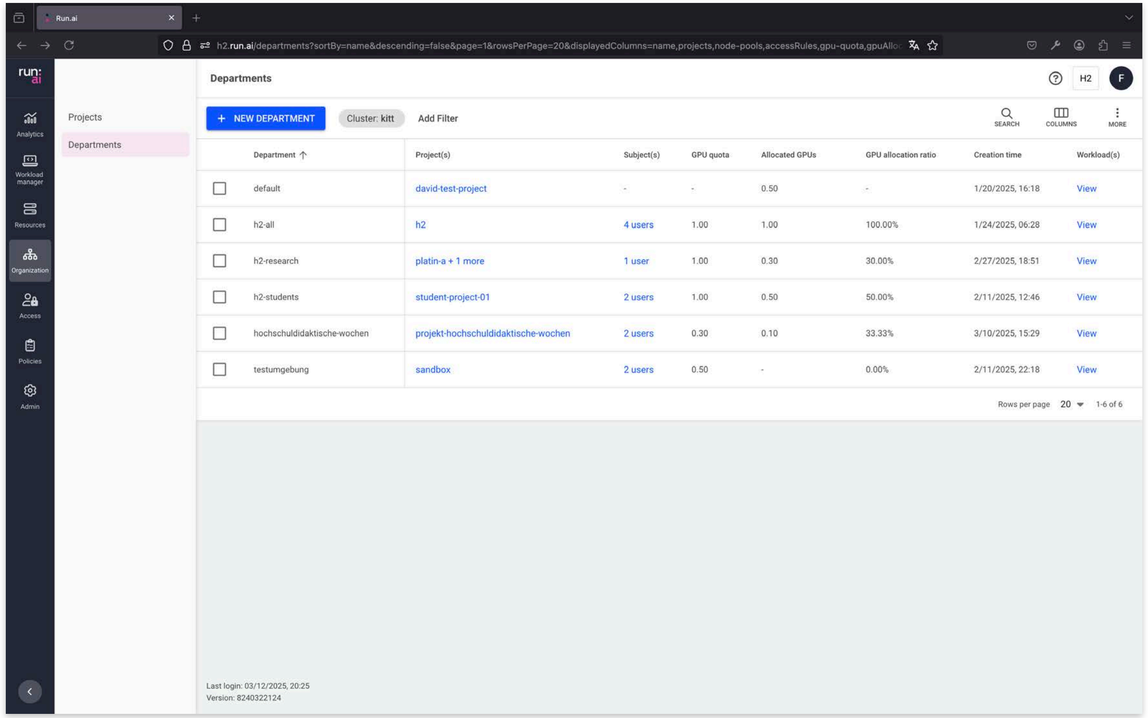


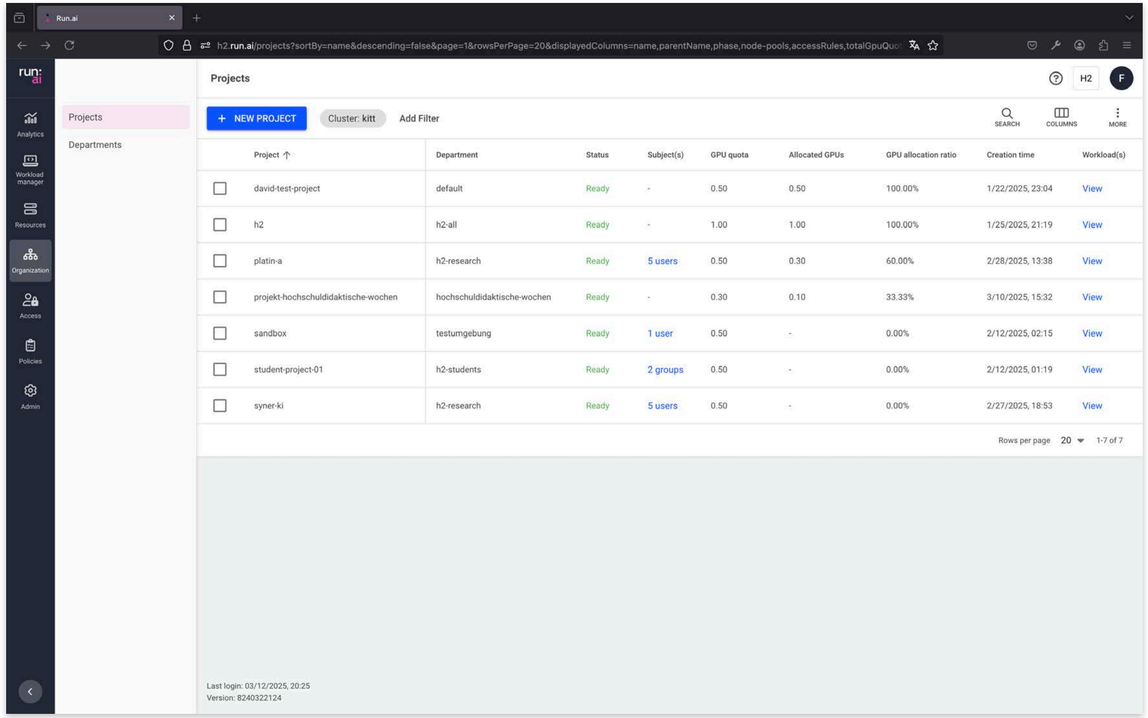
Run:ai bietet ein Rollen- und Berechtigungssystem, das Administratoren eine präzise Steuerung des Zugriffs auf Systemressourcen ermöglicht.

Quelle: https://docs.run.ai/v2.18/admin/authentication/roles/

Role	Description
Environment administrator	Create, view, edit, and delete Environments. View Jobs, Workspaces, Dashboards, Data sources, Compute resources, and Templates.
Credentials administrator	Create view, edit, and delete Credentials. View Jobs, Workspaces, Dashboards, Data sources, Compute resources, Templates*, and environments.
Data source administrator	Create, view, edit, and delete Data sources. View Jobs, Workspaces, Dashboards, Environments, Compute resources, and Templates.
Compute resource administrator	Create, view, edit, and delete Compute resources. View Jobs, Workspaces, Dashboards, Environments, Data sources, and Templates.
System administrator	Controls all aspects of the system. This role has global system control and should be limited to a small group of skilled IT administrators.
Department administrator	Create, view, edit, and delete: Departments and Projects. Assign Roles (Researcher, ML engineer, Research manager, Viewer) within those departments and projects. View Dashboards (including the Consumption dashboard).
Editor	View Screens and Dashboards, Manage Departments and Projects.
Research manager	Create, view, edit, and delete: Environments, Data sources, Compute resources, and Templates. View Projects, related Jobs and Workspaces, and Dashboards.
L1 researcher	Create, view, edit, and delete Jobs, Workspaces, Environments, Data sources, Compute resources, Templates, Deployments. View Dashboards.
ML engineer	Create, edit, view, and delete Deployments. View Departments, Projects, Clusters, Node-pools, Nodes, Dashboards.
Viewer	View Departments, Projects, Respective subordinates (Jobs, Deployments, Workspaces, Environments, Data sources, Compute resources, Templates), Dashboards. A viewer cannot edit Configurations.
L2 researcher	Create, view, edit, and delete Jobs, Workspaces. An L2 researcher cannot create, edit, or delete Environments, Data sources, Compute resources, and Templates.
Template administrator	Create, view, edit, and delete Templates. View Jobs, Workspaces, Dashboards, Environments, Compute resources, and Data sources.
Department viewer	View Departments, Projects, assigned subordinates (Jobs, Deployments, Workspaces, Environments, Data sources, Compute resources, Templates), and Dashboards.

Quelle: https://docs.run.ai/v2.18/admin/authentication/roles/



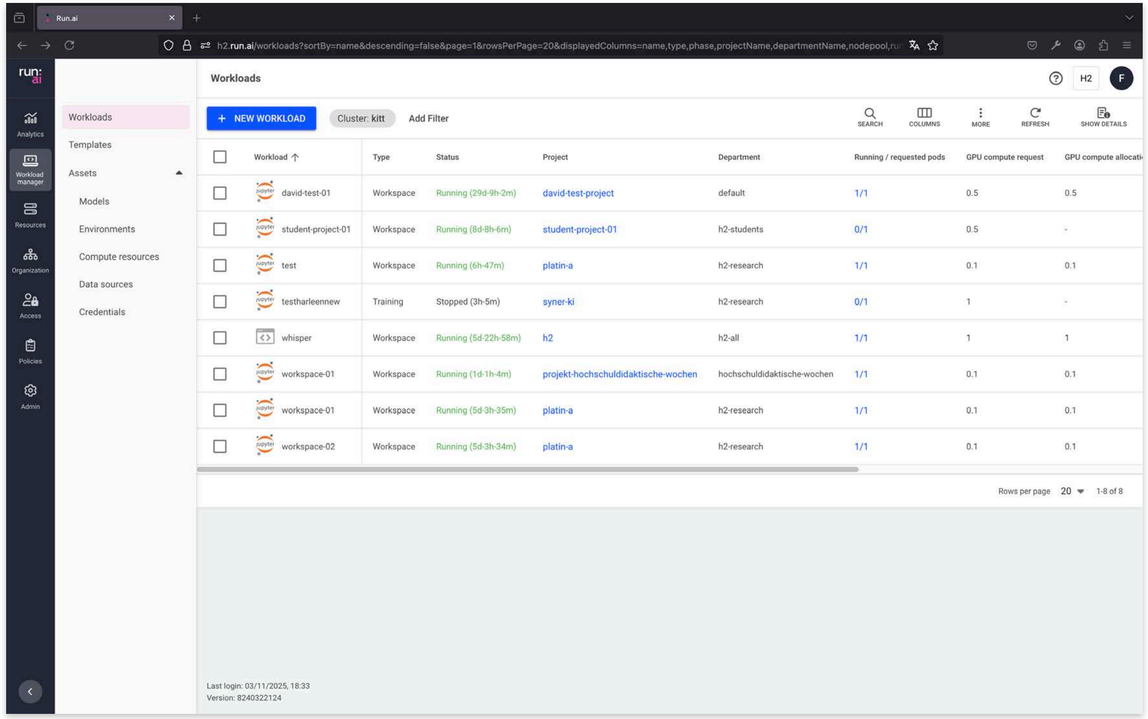


Die verfügbaren Ressourcen wie CPU, GPU und Speicher können den jeweiligen Departments sowie deren zugehörigen Projekten zugewiesen werden.

Quelle: https://h2.run.ai/

Sicherheit & Zugriffskontrollen:

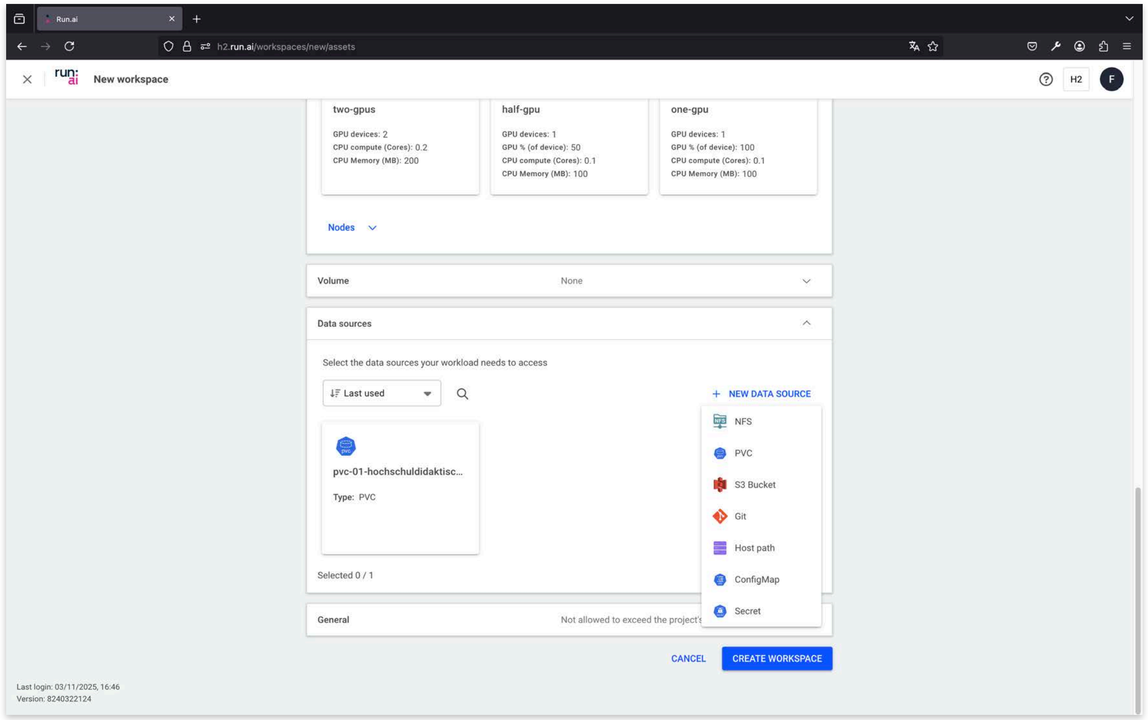
- **Isolierung von Workloads:** Teams haben die Möglichkeit, in getrennten Workspaces zu arbeiten, ohne Zugriff auf fremde Ressourcen oder Daten.
- **Zugriffskontrollen:** Nur autorisierte Nutzer erhalten spezifische Berechtigungen.
- **Netzwerksicherheit:** Die Anbindung an das Hochschulnetzwerk stellt sicher, dass nur autorisierte Nutzer Zugriff erhalten und unbefugter Zugang verhindert wird (Run:ai ist über VPN auch außerhalb der Hochschule nutzbar).



Quelle: https://pages.run.ai/hubfs/PDFs/Building Enterprise-class AI Infrastructure - Datasheet.pdf

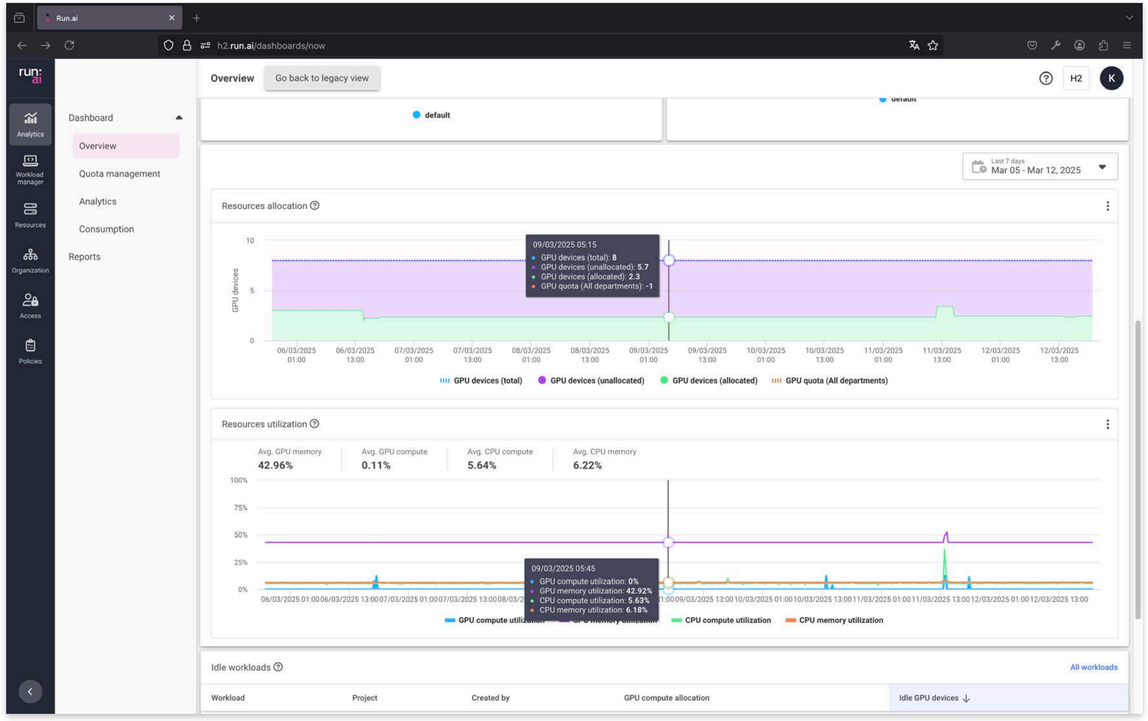
Cloud-Speicher/externe Datenspeicher:

- **S3-Buckets:** Integration von Amazon S3-kompatiblen Speichern.
- **NFS (Network File System):** Einbindung von NFS-Servern für den Datenaustausch.
- **Persistent Volume Claims (PVCs):** Ermöglichen die Anforderung von persistentem Speicher innerhalb von Kubernetes, wodurch Daten über den Lebenszyklus einzelner Pods hinaus bestehen bleiben.
- **Git-Integration:** Run:ai ermöglicht die Integration von Git-Repositories als Datenquellen, wodurch Code direkt aus einem Git-Branch in ein dediziertes Verzeichnis im Container kopiert wird.
- **Host Path:** Zugriff auf Verzeichnisse des Host-Systems.
- **ConfigMap:** Speicherung von Konfigurationsdaten.
- **Secret:** Sichere Speicherung sensibler Informationen.



Quelle: https://docs.run.ai/v2.19/Researcher/workloads/assets/datasources/

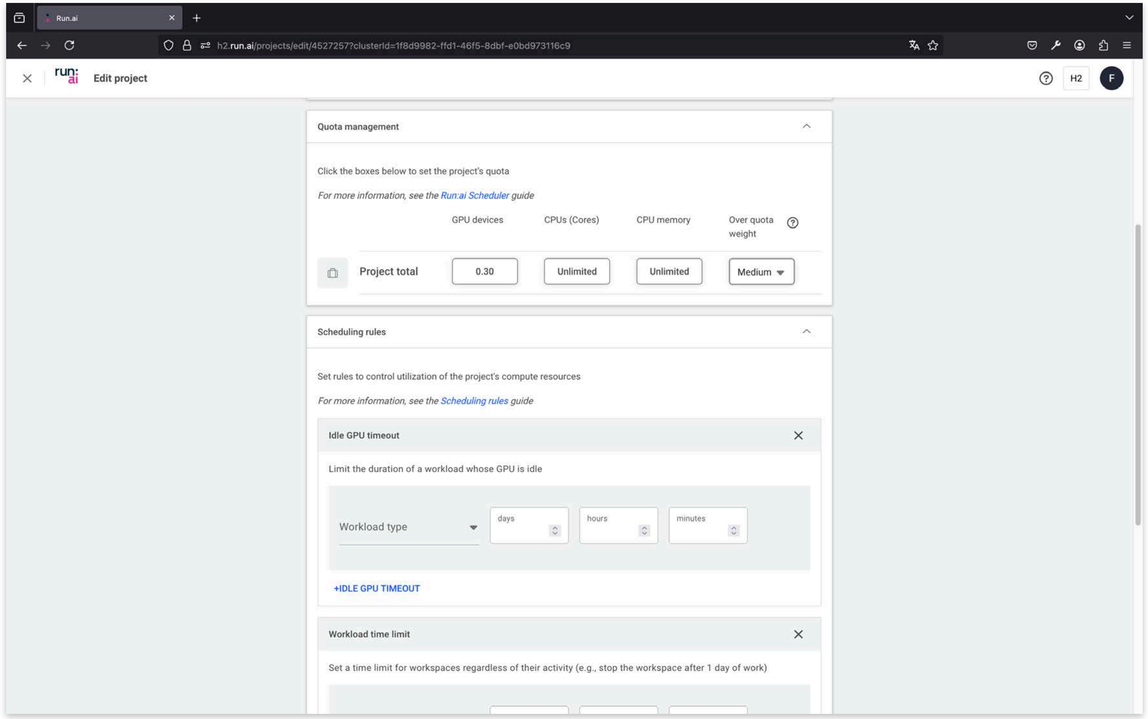
1. **Department anlegen:** Eine neue Abteilung mit den entsprechenden Parametern erstellen.
2. **Projekt erstellen:** Ein Projekt innerhalb des Departments anlegen und konfigurieren.
3. **JupyterLab Workspaces einrichten:** Drei JupyterLab Workspaces erstellen und dem Projekt zuweisen.
4. **Benutzer hinzufügen:** kittguest2@icloud.com als L2 Researcher hinzufügen.
   - **Department:** hochschuldidaktische-wochen
   - **Editor:** kittguest1@icloud.com
   - **L2 researcher:** kittguest2@icloud.com



Quelle: https://h2.run.ai/

Ressourcenaufteilung der drei Departments:

- **h2-students:** 1 GPU für studentische Projekte, Abschlussarbeiten etc.
- **h2-research:** 1 GPU für Forschungsprojekte
- **h2-all:** 1 GPU für weitere Dienste



Quelle: https://h2.run.ai/

Projektlaufzeiten:

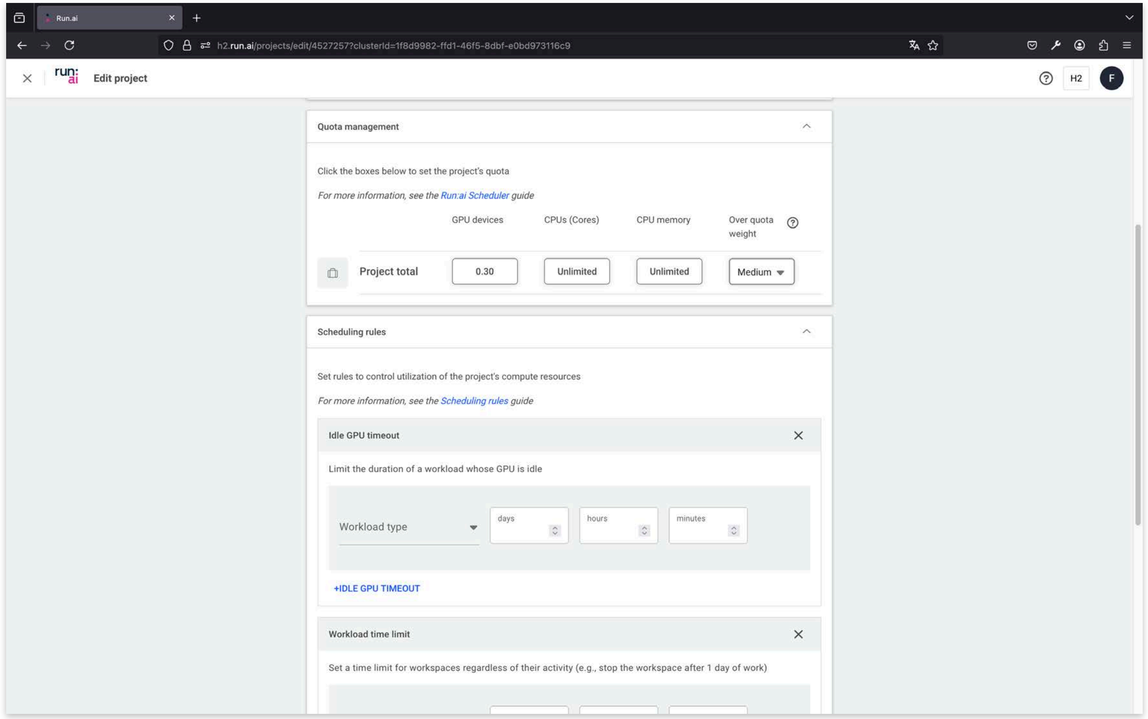
1. **Projekte im Bereich h2-students:** Gültigkeit 6 Monate

👉 Alle 6 Monate wird eine Nachricht per E-Mail versendet, und das Projekt wird ggf. gelöscht.

2. **Projekte im Bereich h2-research:** Gültigkeit 12 Monate

👉 Alle 12 Monate wird eine Nachricht per E-Mail versendet, und das Projekt wird ggf. gelöscht.

3. **Projekte im Bereich h2-all:** Kein Ablauf



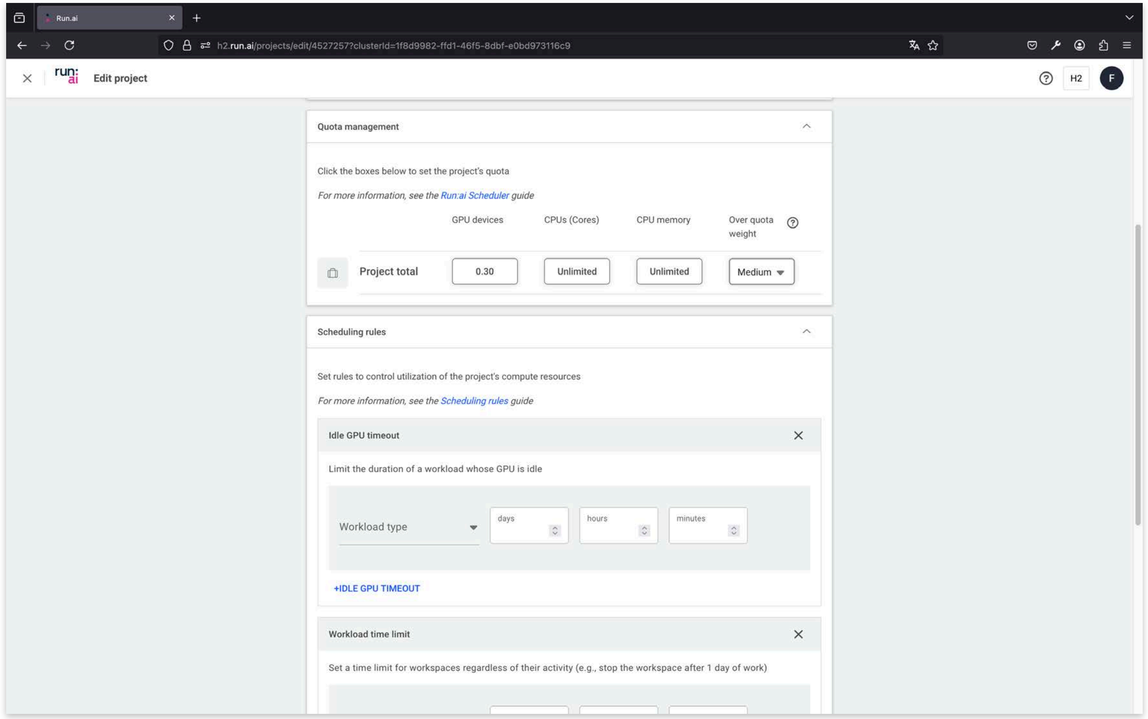
Quelle: https://h2.run.ai/

Workload-Laufzeiten:

1. **h2-students:** Laufzeit 48 Stunden

👉 Automatische Schließung: 48 Stunden, falls der Workload durchgehend aktiv genutzt wurde.

👉 Leerlauf-Erkennung: Falls die GPU 6 Stunden lang nicht genutzt wird, wird der Workload gestoppt.



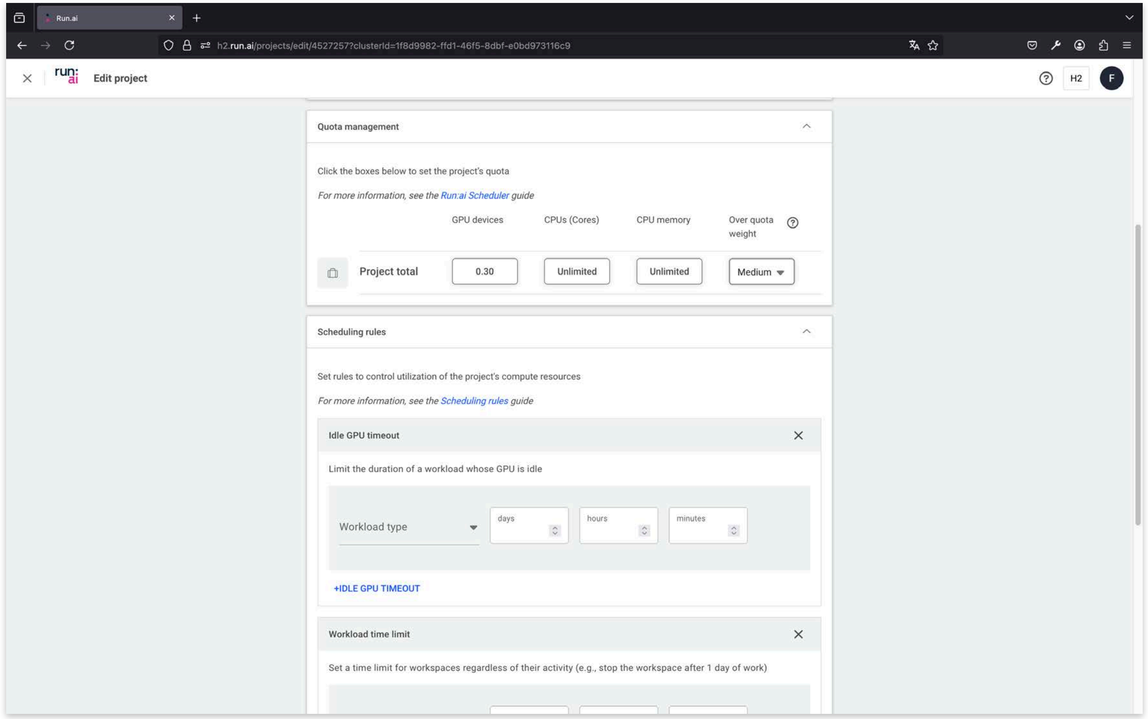
Quelle: https://h2.run.ai/

Workload-Laufzeiten:

2. **h2-research:** Laufzeit: 10 Tage

👉 Automatische Schließung: Nach 10 Tagen, falls der Workload durchgehend aktiv genutzt wurde.

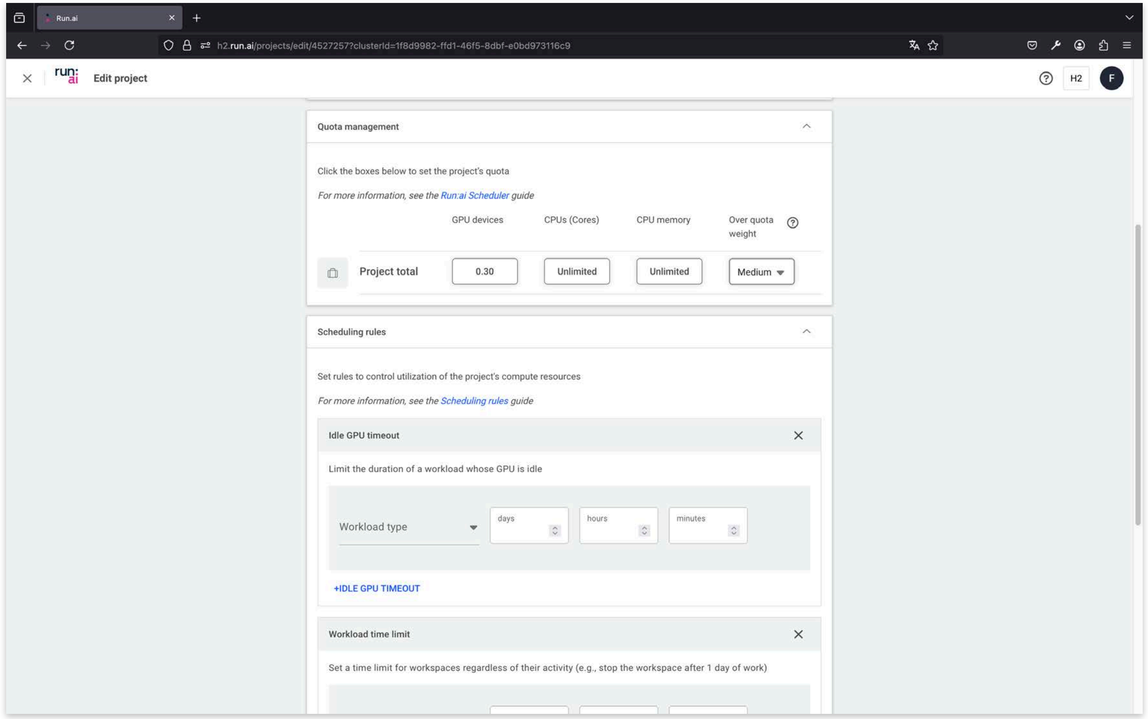
👉 Leerlauf-Erkennung: Falls die GPU 12 Stunden lang nicht genutzt wird, wird der Workload gestoppt.



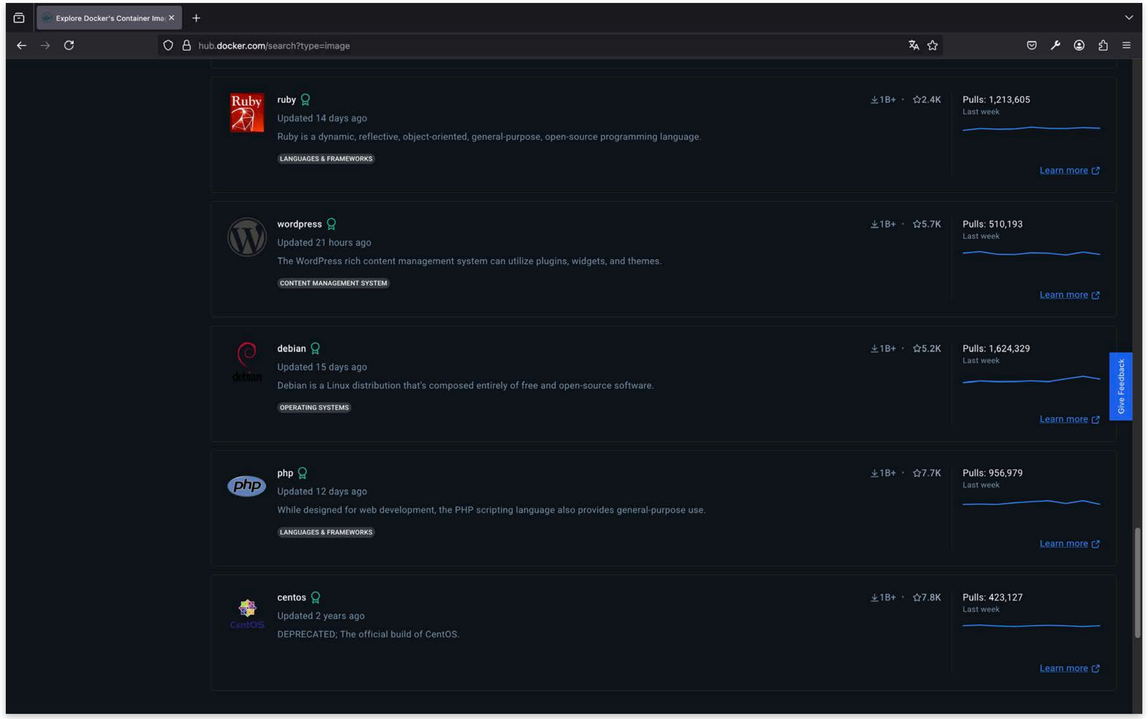
Quelle: https://h2.run.ai/

Workload-Laufzeiten:

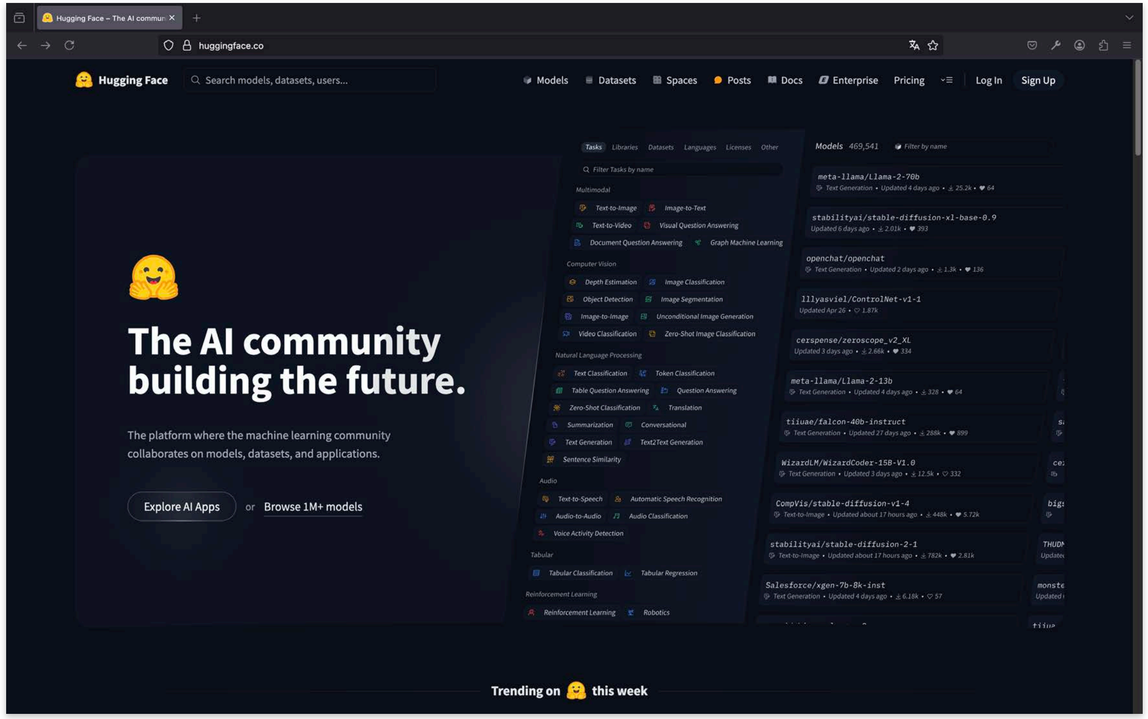
3. **h2-all:** Kein Ablauf



Quelle: https://h2.run.ai/

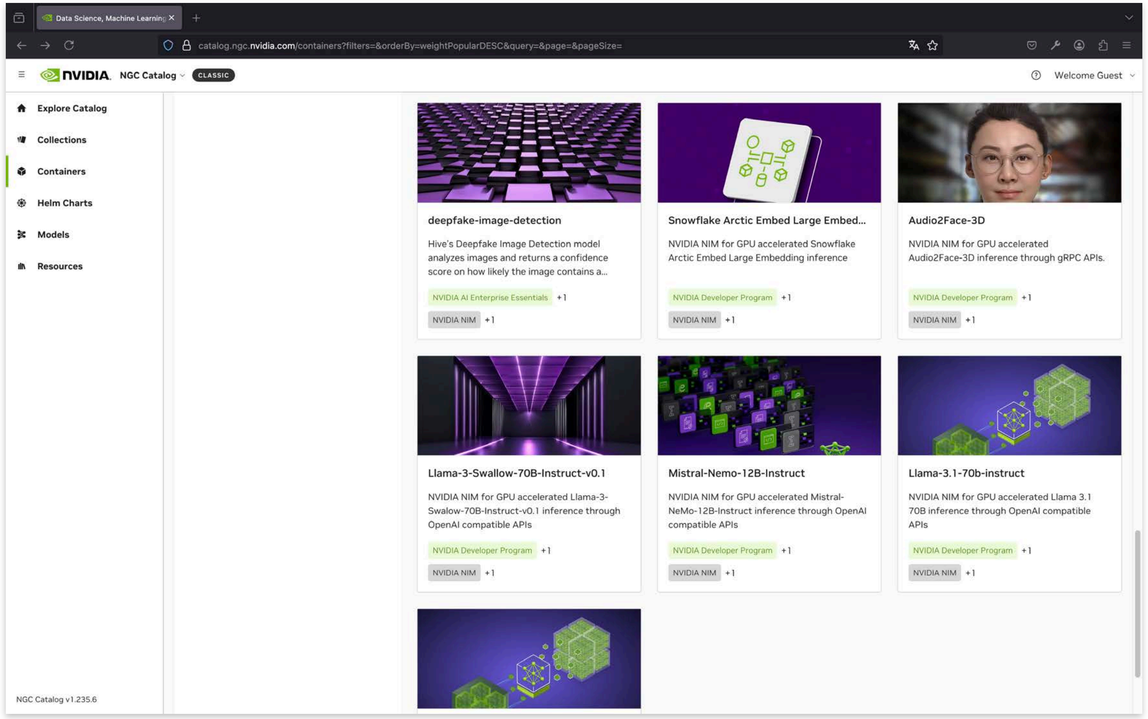


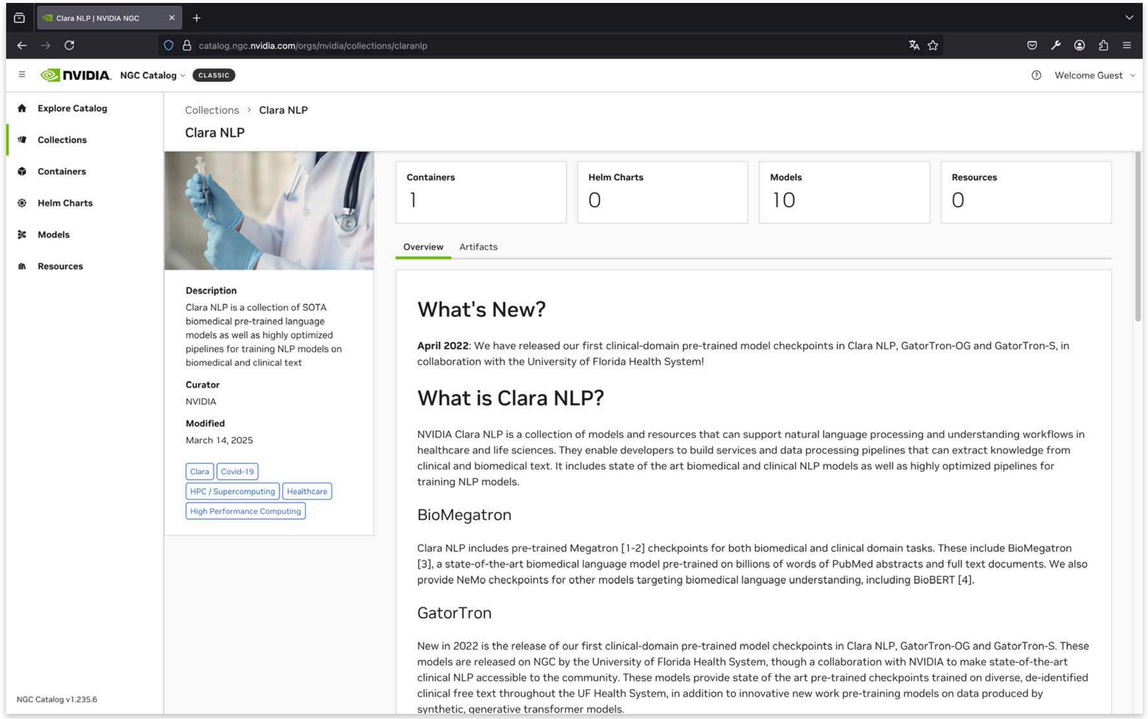
Mithilfe von Docker Hub lassen sich individuelle Images in Run:ai integrieren und für Lehre und Forschung nutzen.



Der Inferenzmodus in Run:ai ermöglicht den Betrieb von Modellen aus Hugging Face und dem NGC Catalog.

Quelle: https://hub.docker.com/, https://huggingface.co/





NVIDIA NGC: GPU Accelerated AI models and SDKs that help you infuse AI into your applications at speed of light

Quelle: https://catalog.ngc.nvidia.com/?filters=&orderBy=weightPopularDESC&query=&page=&pageSize=

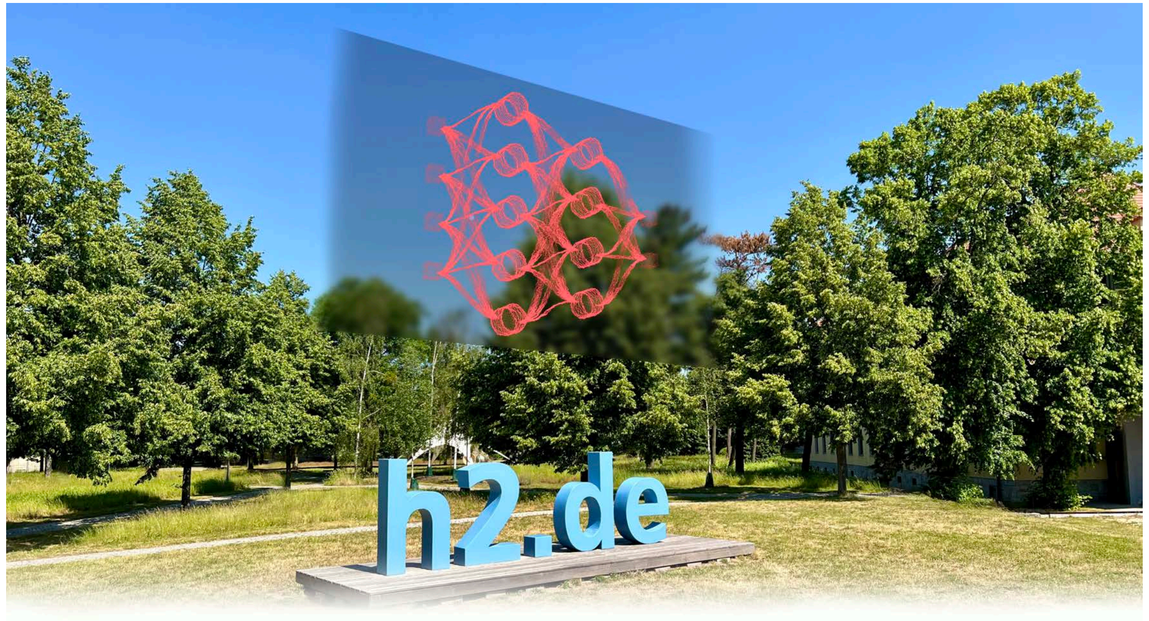
🚀 **KITT Service Desk:** https://kitt.h2.de/

🚀 **KITT Mail:** kitt@h2.de

🚀 **Run:ai:** http://h2.run.ai

Overview: Researcher Documentation: https://docs.run.ai/v2.17/Researcher/overview-researcher/

Run:ai Official (Acquired by NVIDIA): https://www.youtube.com/@runai_



Quellen

Titelbild generiert mithilfe von Midjourney v6 (2025) basierend auf folgendem Prompt:

"Abstract digital artwork representing the raw power of GPUs and artificial intelligence. A dynamic explosion of vibrant energy, featuring interconnected neural networks, glowing data streams, and fractal-like circuits weaving through an ethereal space. Electric blue, neon purple, and fiery orange tones blend seamlessly, symbolizing deep learning computations and high-performance processing. Floating geometric shapes and swirling particles create a sense of speed, precision, and complexity. The composition conveys an otherworldly fusion of technology and creativity, with a futuristic, hyper-detailed style. High contrast, cinematic lighting, and a sense of infinite depth."