

AI智算云平台 容器云实操培训课程

北京超级云计算中心/ 技术支持工程师 /徐子婷

2024年8月

CONTENTS

- 一 中心介绍
- 二 容器云介绍
- 三 容器实例
- 四 配置环境

一、中心介绍

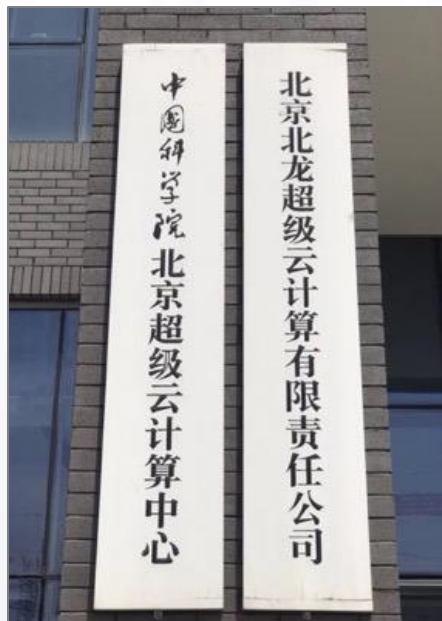


北京超级云计算中心
BEIJING SUPER CLOUD COMPUTING CENTER

北京市人民政府主导、院市共建的“北京超级云计算和国家重要信息化基础平台”

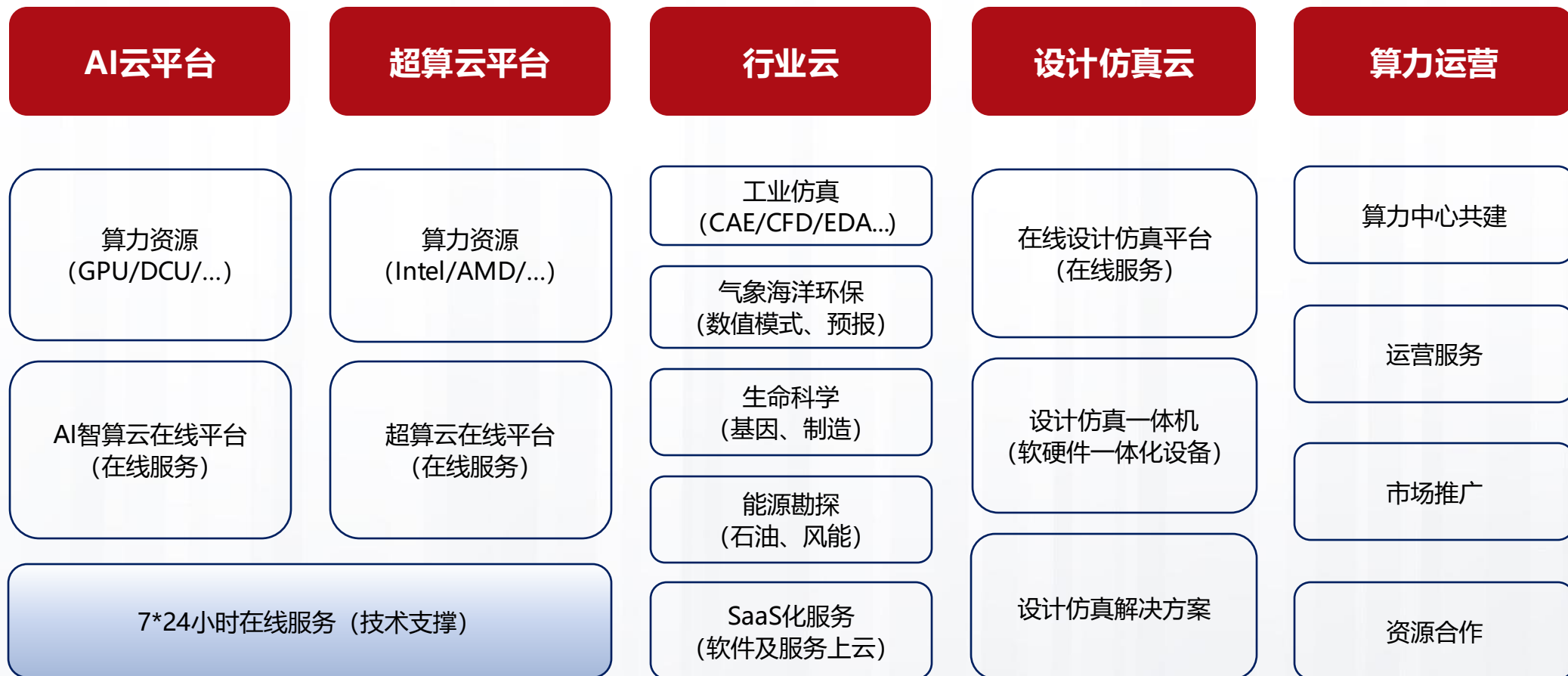
2011年11月，北京超级云计算中心奠基启动，并成立北京北龙超级云计算有限责任公司负责实体运营。

- 坐落北京怀柔综合性国家科学中心-怀柔科学城
- 《北京市通用人工智能产业创新伙伴计划》首批算力伙伴
- 科技部国家超算互联网联合体理事单位
- 连续4年荣获HPC TOP100通用CPU算力第一



面向不同场景的计算服务方案

产品矩阵



AI智算云

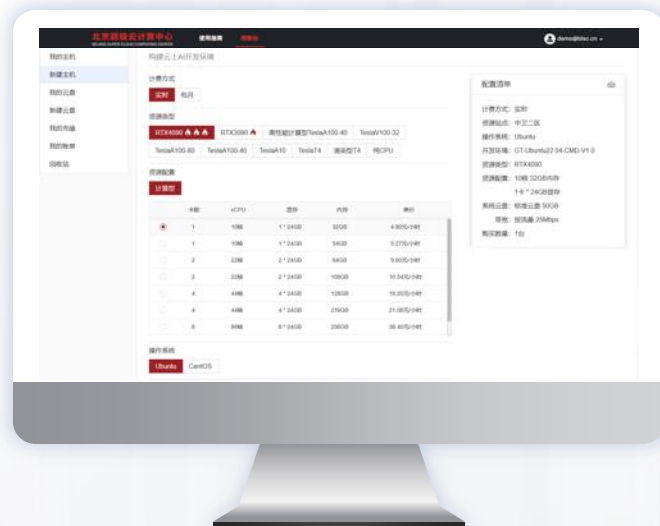
- 提供专业的GPU算力资源、以及简单易用的使用环境
- AI工程师可以轻松获取高性价比的智能算力资源
- 减少对算力环境建设、维护等工作，将更多精力用于科研项目

算力资源丰富

- H800、A800、V100、3090、A10、T4、国产DCU等主流型号的海量资源
- 支持多机多卡，满足推理、训练、科学计算等多种场景

支持多机多卡并行 HPC集群

主要用于训练、推理 AI云主机



高效易用

- 预置Tensorflow、PyTorch主流框架，内置数据集
- 通过简单易用的界面快速创建AI实例
- 裸金属服务器、独占显卡，性能强劲

高性价比

- 建设与运维“0”成本
- 按需灵活选用海量资源
- 用户将更多精力投入在科研

多元、稳定的大规模算力，支撑不同领域、规模、精度的计算诉求

H800-80

GPU: 8 * NVIDIA®Tesla®H800 SXM5
 显存: 8 * 80GB
 CPU: 2* Intel Sapphire Rapids 32 CPU
 内存: 2048GB DDR5
 NVLink: 双向通信400 GB/s
 节点互联: 8 * 200G IB (聚合1.6Tb)

A800-80

GPU: 8 * NVIDIA®Tesla®A800 SXM4
 显存: 8 * 80GB
 CPU: 2 * Intel Icelake 83系列 32 CPU
 内存: 2048GB
 NVLink: 双向通信400 GB/s
 节点互联: 8*100G IB (聚合800Gb)

A100-80

GPU: 8 * NVIDIA®Tesla®A100 SXM4
 显存: 8 * 80GB (2039 GB/s)
 CPU: Intel 83系列 112vCPU
 内存: 1960GB
 NVLink: 双向通信600 GB/s
 节点互联: 4 * 200G RoCE (RDMA协议)

V100-32

GPU: 8 * NVIDIA®Tesla®V100 SXM2
 显存: 8 * 32GB (897 GB/s)
 CPU: Platinum 82系列 80vCPU
 内存: 320GB
 NVLink: 双向通信300 GB/s
 节点互联: 100G RoCE (RDMA协议)

RTX 4090

GPU: 8 * NVIDIA®GeForce®RTX 4090
 显存: 8 * 24GB (936.2 GB/s)
 CPU: AMD EPYC™ 7402 48C
 内存: 512GB
 节点互联: 2 * 25G RoCE (RDMA协议)

RTX 3090

GPU: 8 * NVIDIA®GeForce®RTX 3090
 显存: 8 * 24GB (936.2 GB/s)
 CPU: AMD EPYC™ 7002 Series128
 内存: 512GB
 节点互联: 2 * 25G RoCE (RDMA协议)

国产加速卡

DCU: 4*国产加速卡, PCI-E G3 X16
 CPU: 国产CPU@2.0GHZ
 单节点32核
 内存: 128GB 节点互连: 200G

其他

A100-40 \ V100-16 \ A30 \ A10 \ T4 \ 2080Ti

二、容器云介绍



BSCC

北京超级云计算中心
BEIJING SUPER CLOUD COMPUTING CENTER

容器云

- 面向高校、科研、企业用户，提供弹性、按需的简便、易用、高效、快捷、高性价比产品
- 提供的资源类型包括：RTX3090、RTX4090
- 提供容器实例、网络、共享存储和自定义镜像能力，可满足绝大部分应用场景需求



AI容器云

满足各类需求，提供多种功能

- 提供多种卡数配置，以及开发环境
- 支持JupyterLab、WebSSH连接，多种工具使用
- 创建实例，存储，网络，镜像提供能力全面

平台易用，功能全面

- 预置PyTorch主流框架
- 通过简单易用的界面快速创建AI实例
- 保持用户使用习惯，操作零门槛

高效快捷，团队支持

- 省心：专业团队支持云服务，用户专注领域科研
- 省钱：消除高投入的计算资源建设成本门槛，选择先进的付费模式

容器云提供的能力



容器云实例

- 容器创建后自带30GB系统盘
- 开机、关机、重启、创建容器镜像



容器云网络

- 共享带宽 1Gbps
- 减少成本



容器云共享存储

- RTX3090、RTX4090实例共享存储互通
- 默认20GB免费额度



容器云镜像

- 公共镜像库提供多版本支持
- 可创建自定义镜像

三、容器实例



北京超级云计算中心
BEIJING SUPER CLOUD COMPUTING CENTER

创建实例

- 平台提供了多种GPU容器规格，按需求选择不同的配置，灵活搭配构建计算资源

使用公共镜像创建实例：

- 计费模式
- 站点
- 资源类型
- 资源配置
- 镜像类型
- 开发环境
- 容器名称
- 购买数量

The screenshot displays the '容器实例' (Container Instance) creation page. Key elements include:

- Navigation:** 北京超级云计算中心 AI 智算云 | 总览 云服务平台 容器云
- Left Sidebar:** 容器实例 (selected), 数据服务, 容器镜像
- Search:** 资源ID, 添加筛选条件, 多个关键字用“|”分隔
- Configuration Section:**
 - 计费模式: 按量计费 (适合需求量大峰谷波动业务)
 - 站点: 全部, 中卫一区
 - 资源类型: RTX3090
- Resource Configuration Table:**

配置	卡数	vCPU	显存	内存	本地系统盘
<input checked="" type="radio"/>	1	11核	1 * 24GB	60GB	30GB
<input type="radio"/>	2	22核	2 * 24GB	120GB	30GB
<input type="radio"/>	4	44核	4 * 24GB	240GB	30GB
<input type="radio"/>	8	88核	8 * 24GB	480GB	30GB
- Additional Settings:**
 - 镜像类型: 公共镜像 (selected), 自定义镜像
 - 开发环境: Pytorch 1.8.0, pytorch-20.12-py9 | Pytorch1.8.0-Ubuntu 22.04 |
 - 数据存储: 自动挂载共享存储
 - 共享存储: 将本组的容器挂载共享存储, 持久化存储数据, 目录: /home/pod/shared-nvme. 免费额度: 免费提供20GB, 如需更大容量, 可在创建容器后前往【数据服务】进行扩容。
 - 容器名称: kcs-ksqthjk
 - 购买数量: 1 台

其他实例相关操作

- 点击界面容器云，左侧的容器实例实现对容器的四种操作

- 启动容器
- 重启容器
- 关闭容器
- 删除容器



查看开启实例详情

- 点击界面容器云，左侧的容器实例查看已开启的容器具体配置

北京超级云计算中心 BEIJING SUPER CLOUD COMPUTING CENTER

AI 智算云 | 总览 云服务器 容器云

容器实例

容器存储

容器镜像

容器实例 1 严考

创建容器 资源ID 添加筛选条件, 多个关键字用 "|" 分隔

kcs-cbthvna 已关机

ID: ackcs-00gjdjtb

计费模式: 按量计费 站点: 中卫一区

起止时间: 2024-08-25 11:23:20

资源配置

资源类型: RTX3090 GPU: 1卡 * 24 GB显存

CPU: 10核 内存: 60GB

开发框架: PyTorch2.2.0-Ubuntu 20.04

开机 删除 查看详情

实例详情

基本信息

名称	kcs-cbthvna	ID	ackcs-00gjdjtb
计费模式	按量计费	站点	中卫一区
起止时间	2024-08-25 11:23:20		

资源配置

资源类型	RTX3090	GPU	1卡 * 24 GB显存
CPU	10核	内存	60GB
开发框架	PyTorch2.2.0-Ubuntu 20.04	操作系统	Ubuntu 20.04
镜像信息	镜像ID: ackci-0000qyv84 镜像名称: llama3-git		

可用连接方式一：ssh连接

- ssh连接方式分为两种：开机后平台webssh连接或第三方工具连接

kcs-cbthvna [🔗](#) ● 运行中

ID: ackcs-00gjdjtb [🔗](#)

计费模式: 按量计费 站点: 中卫一区

起止时间: 2024-08-25 11:23:20

资源配置

资源类型: RTX3090 GPU: 1卡 * 24 GB显存

CPU: 10核 内存: 60GB

开发框架: PyTorch2.2.0-Ubuntu 20.04

JupyterLab TensorBoard **WebSSH** ⋮

打开WebSSH，通过命令操作容器实例

ackcs-00gjdjtb [🔗](#)

ID: ackcs-00gjdjtb [🔗](#)

计费模式: 按量计费 站点: 中卫一区

起止时间: 2024-08-25 11:23:20

资源配置

资源类型: RTX3090 GPU: 1卡 * 24 GB显存

CPU: 10核 内存: 60GB

开发框架: PyTorch2.2.0-Ubuntu 20.04

JupyterLab TensorBoard WebSSH **⋮**

实例详情

基本信息

名称	kcs-cbthvna 🔗	ID	ackcs-00gjdjtb 🔗
计费模式	按量计费	站点	中卫一区
起止时间	2024-08-25 11:23:20		

资源配置

资源类型	RTX3090	GPU	1卡 * 24 GB显存
CPU	10核	内存	60GB
开发框架	PyTorch2.2.0-Ubuntu 20.04	操作系统	Ubuntu 20.04

镜像信息

镜像ID: ackci-0000qxv84
镜像名称: llama3-git

登录信息

IP地址	36.103.203.203	端口	31937
用户名	pod	密码	***** 👁

WebSSH

第三方工具，用实例的信息连接

可用连接方式二：JupyterLab连接

kcs-cbthvna [🔗](#)

● 运行中

ID: ackcs-00gdjtb [📄](#)

计费模式: 按量计费

站点: 中卫一区

起止时间: 2024-08-25 11:23:20

资源配置

资源类型: RTX3090

GPU: 1卡 * 24 GB显存

CPU: 10核

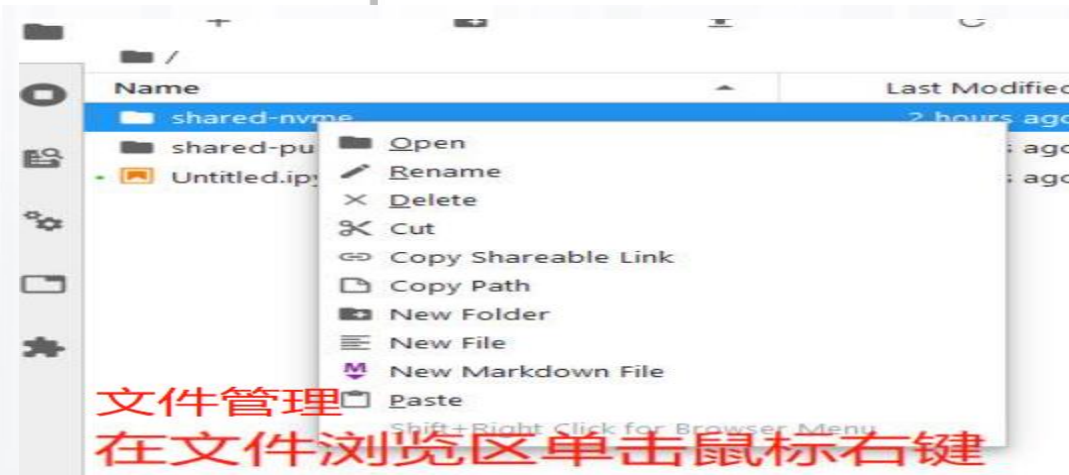
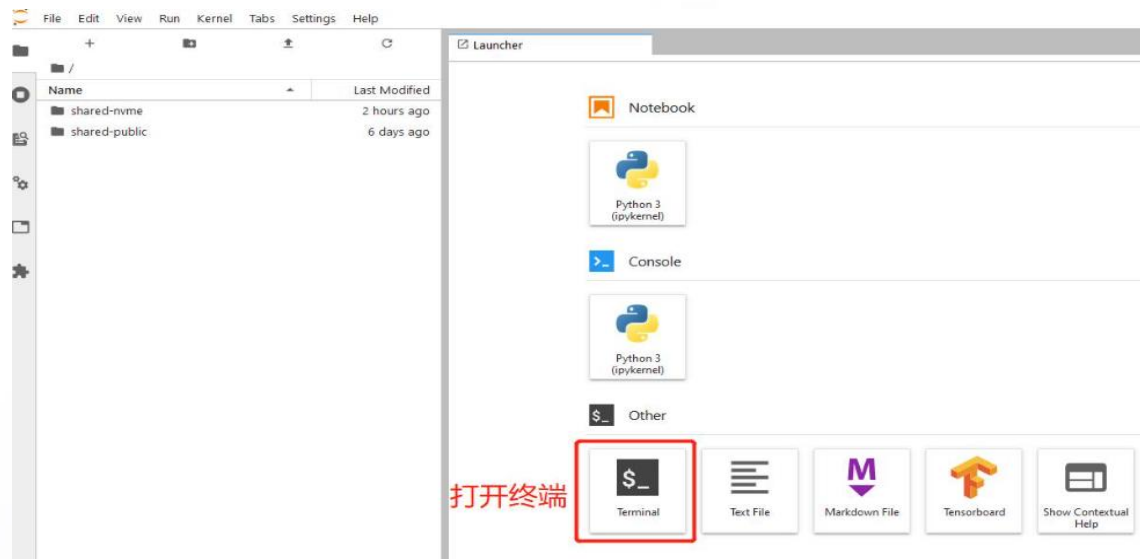
内存: 60GB

开发框架: PyTorch2.2.0-Ubuntu 20.04

JupyterLab

TensorBoard

WebSSH



JupyterLab 的工作目录为/root 目录

注: 访问打开的终端或 Notebook (JupyterLab 在关闭终端/Notebook 选项卡后默认仍在运行)

可用连接方式三：TensorBoard连接

kcs-cbthvna [✎](#) ● 运行中

ID: ackcs-00gjdjtb [📄](#)

计费模式: 按量计费 站点: 中卫一区

起止时间: 2024-08-25 11:23:20

资源配置

资源类型: RTX3090 GPU: 1卡 * 24 GB显存

CPU: 10核 内存: 60GB

开发框架: PyTorch2.2.0-Ubuntu 20.04

[JupyterLab](#) | **[TensorBoard](#)** | [WebSSH](#) | [⋮](#)

TensorBoard INACTIVE [⌵](#) [⚙️](#) [🔄](#) [⚙️](#) [?](#)

No dashboards are active for the current data set.

Probable causes:

- You haven't written any data to your event files.
- TensorBoard can't find your event files.

If you're new to using TensorBoard, and want to find out how to add data and set up your event files, check out the [README](#) and perhaps the [TensorBoard tutorial](#).

If you think TensorBoard is configured properly, please see [the section of the README devoted to missing data problems](#) and consider filing an issue on GitHub.

Last reload: Aug 26, 2024, 11:51:27 AM

Log directory: /home/pod

点击【TensorBoard】按钮，跳转至对应的TensorBoard页面

文件传输

- 文件传输方式分为三种：在线文件传输和本地文件传输

容器实例

创建容器

资源ID ▼ 添加筛选条件, 多个

kcs-cbhthvna [🔗](#)

ID: ackcs-00gjdjtb [📄](#)

计费模式: 按量计费 站点: 中卫一区

起止时间: 2024-08-25 11:23:20

资源配置

资源类型: RTX3090 GPU: 1卡 * 24 GB显

CPU: 10核 内存: 60GB

开发框架: PyTorch2.2.0-Ubuntu 20.04

JupyterLab

TensorBoard

WebSSH

⋮

平台在线文件传输

重启
关机
转包年包月
查看详情
创建镜像
调整配置
文件传输
开放端口
立即释放

新建会话属性

常规 选项

FTP 站点

名称(N): 新建会话

主机(H):

协议(R): SFTP

端口号(O): 22

代理服务器(X): <无>

说明(D):

登录

匿名登录(A)

使用身份验证代理(G)

方法(M): Password

用户名(U):

密码(P):

用户密钥(K):

密码(E):

Xftp, WinScp等本地传输工具连接, 同ppt15页实例信息连接方式获取端口,ip等信息

存储介绍

共享存储

- 自动创建共享存储并挂载，默认20GB
- 挂载目录：
/home/pod/shared-nvme
- 共享存储实现所有容器实例间共享

文件管理

- 通过顶栏【容器云】 - 【容器存储】-文件管理员功能
- 支持上传，删除，重命名文件，方便管理

扩容与删除

- 通过顶栏【容器云】 - 【容器存储】可看到扩容与删除
- 注：删除共享存储前需删除容器实例

调整配置和开放端口

- 容器实例在运行状态下可进行升级配置操作
- 实例支持开放对外端口，用以在外部访问实例内部的运行的指定端口的服务

The screenshot displays a cloud management console interface for container instances. The main content area shows details for an instance named 'kcs-cbhthvna'. Below the instance name, it lists the ID, billing mode (pay-as-you-go), location (Zhongwei Zone 1), and start time. A '资源配置' (Resource Configuration) section provides details on the instance's hardware and software stack, including RTX3090 GPU, 10-core CPU, 60GB memory, and a PyTorch2.2.0-Ubuntu 20.04 development framework. At the bottom, there are tabs for 'JupyterLab', 'TensorBoard', and 'WebSSH'. A context menu is open over the instance, listing various actions: '重启' (Restart), '关机' (Power Off), '转包年包月' (Convert to Annual/Monthly Billing), '查看详情' (View Details), '创建镜像' (Create Image), '调整配置' (Adjust Configuration), '文件传输' (File Transfer), '开放端口' (Open Ports), and '立即释放' (Release Immediately). The '调整配置' and '开放端口' options are highlighted with red boxes.

容器实例

[创建容器](#) 资源ID ▼ 添加筛选条件, 多个

kcs-cbhthvna [✎](#)

ID: ackcs-00gjdjtb [📄](#)

计费模式: 按量计费 站点: 中卫一区

起止时间: 2024-08-25 11:23:20

资源配置

资源类型: RTX3090 GPU: 1卡 * 24 GB显

CPU: 10核 内存: 60GB

开发框架: PyTorch2.2.0-Ubuntu 20.04

JupyterLab TensorBoard WebSSH ⋮

重启

关机

转包年包月

查看详情

创建镜像

调整配置

文件传输

开放端口

立即释放

四、配置环境



北京超级云计算中心
BEIJING SUPER CLOUD COMPUTING CENTER

Conda环境配置

平台提供的镜像不满足要求，可自行创建：

- 构建一个虚拟环境名为：**my_env**，Python 版本为 **3.9**

```
conda create -n my_env python=3.9
```

- 激活环境

```
source activate my_env
```

- 安装包(常见**pip**安装和**conda**安装)

```
pip install 包名 -i https://pypi.tuna.tsinghua.edu.cn/simple (清华源)
```

```
conda install -c xxxx 包名
```

参考命令网址(输入包名查看命令)：

```
https://pypi.org/project/
```

```
https://anaconda.org/
```

- **Torch**环境创建

可用 conda 或 pip 安装，推荐 pip

参考命令网址：

```
https://pytorch.org/get-started/previous-versions/
```

```
To restore this content, you can run the 'unminimize' command.
Last login: Mon Aug 26 11:10:50 2024 from 10.64.2.46
pod@p-4fcff2f2bda6-ackcs-00gjdjtb-0:~$ conda create -n nowcasting python=3.9 ^C
pod@p-4fcff2f2bda6-ackcs-00gjdjtb-0:~$ source activate nowcasting
(nowcasting) pod@p-4fcff2f2bda6-ackcs-00gjdjtb-0:~$ pip install numpy -i https://
/pypi.tuna.tsinghua.edu.cn/simple ^C
(nowcasting) pod@p-4fcff2f2bda6-ackcs-00gjdjtb-0:~$ conda install conda-forge::nu
umpy
```

```
(nowcasting) pod@p-4fcff2f2bda6-ackcs-00gjdjtb-0:~$ pip install torch==2.3.1 torc
hvision==0.18.1 torchaudio==2.3.1 --index-url https://download.pytorch.org/whl/cu
121
```

常见包安装

➤ Tensorflow

建议用pip安装,参考网址为<https://tensorflow.google.cn/install/pip?hl=en>
tensorflow 版本与 python 版本、CUDA、cuDNN 版本有对应关系

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow-2.15.0	3.9-3.11	Clang 16.0.0	Bazel 6.1.0	8.9	12.2
tensorflow-2.14.0	3.9-3.11	Clang 16.0.0	Bazel 6.1.0	8.7	11.8
tensorflow-2.13.0	3.8-3.11	Clang 16.0.0	Bazel 5.3.0	8.6	11.8
tensorflow-2.12.0	3.8-3.11	GCC 9.3.1	Bazel 5.3.0	8.6	11.8
tensorflow-2.11.0	3.7-3.10	GCC 9.3.1	Bazel 5.3.0	8.1	11.2
tensorflow-2.10.0	3.7-3.10	GCC 9.3.1	Bazel 5.1.1	8.1	11.2
tensorflow-2.9.0	3.7-3.10	GCC 9.3.1	Bazel 5.0.0	8.1	11.2
tensorflow-2.8.0	3.7-3.10	GCC 7.3.1	Bazel 4.2.1	8.1	11.2
tensorflow-2.7.0	3.7-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.6.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.5.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.4.0	3.6-3.8	GCC 7.3.1	Bazel 3.1.0	8.0	11.0
tensorflow-2.3.0	3.5-3.8	GCC 7.3.1	Bazel 3.1.0	7.6	10.1
tensorflow-2.2.0	3.5-3.8	GCC 7.3.1	Bazel 2.0.0	7.6	10.1
tensorflow-2.1.0	2.7, 3.5-3.7	GCC 7.3.1	Bazel 0.27.1	7.6	10.1
tensorflow-2.0.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10.0

```
(nowcasting) pod@p-4fcff2f2bda6-ackcs-00gjdjtb-0:~$ pip install tensorflow==版本
-i https://pypi.tuna.tsinghua.edu.cn/simple
```

GPU 架构信息

GPU	arch	CUDA gencode	支持的CUDA版本
RTX3090	Ampere	sm_86、compute_86	>=11.1
RTX4090	Ada Lovelace	sm_89、compute_89	>=11.8

注：GPU 架构对 cuda 版本有要求，见上图

安装测试

➤ Torch测试正确安装方法，并输出对应版本

- 进入环境中：source activate xxxx (环境名)
- 进入python :python
- import torch
- print(f"Python version: {torch.__version__}")
print(f"CUDA available: {torch.cuda.is_available()}")
print(f"CUDA version: {torch.version.cuda}")
print(f"cuDNN version: {torch.backends.cudnn.version()}")
print(f"Number of GPUs: {torch.cuda.device_count()}")

```
(nowcasting) pod@p-4fcff2f2bda6-ackcs-00gjdjtb-0:~$ python
Python 3.9.19 | packaged by conda-forge | (main, Mar 20 2024, 12:50:21)
[GCC 12.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> import torch
>>>
>>> print(f"Python version: {torch.__version__}")
Python version: 2.4.0+cu121
>>>
>>> print(f"CUDA available: {torch.cuda.is_available()}")
CUDA available: True
>>>
>>> print(f"CUDA version: {torch.version.cuda}")
CUDA version: 12.1
>>>
>>> print(f"cuDNN version: {torch.backends.cudnn.version()}")
cuDNN version: 90100
>>>
>>> print(f"Number of GPUs: {torch.cuda.device_count()}")
Number of GPUs: 1
>>> █
```

➤ Tensorflow测试正确安装方法

- 进入环境中：source activate xxxx (环境名)
- 进入python :python
- import tensorflow as tf
- tf.test.is_gpu_available()

```
>>> tf.test.is_gpu_available()
2024-08-26 13:09:50.412351: I tensorflow/core/common_runtime/gpu/gpu_device.cc:20
21] Created device /device:GPU:0 with 22455 MB memory: -> device: 0, name: NVIDI
A GeForce RTX 3090, pci bus id: 0000:01:00.0, compute capability: 8.6
True ←
```


conda注意事项：安装环境到共享存储

- 将虚拟环境安装到 `/home/pod/shared-nvme/envs`，包缓存到 `/home/pod/shared-nvme/pkg`s
 - `mkdir -p /home/pod/shared-nvme/pkg`s
 - `conda config --add pkg`s_dirs `/home/pod/shared-nvme/pkg`s
 - `mkdir -p /home/pod/shared-nvme/envs`
 - `conda config --add envs`_dirs `/home/pod/shared-nvme/envs`

- 通过修改 `/home/pod/.condarc` 配置文件修改（如在配置文件中加）
 - `envs`_dirs:
 - `- /home/pod/shared-nvme/env`
 - `pkg`s_dirs:
 - `- /home/pod/shared-nvme/pkg`s

JupyterLab注意事项：内核设置

➤ 安装 ipykernel 和 ipython_genutils

- pip install ipython_genutils==0.2.0
- pip install ipykernel==5.3.4
- 安装的版本与镜像JupyterLab保持一致

查看版本命令：

```
/opt/conda/bin/pip show ipykernel
```

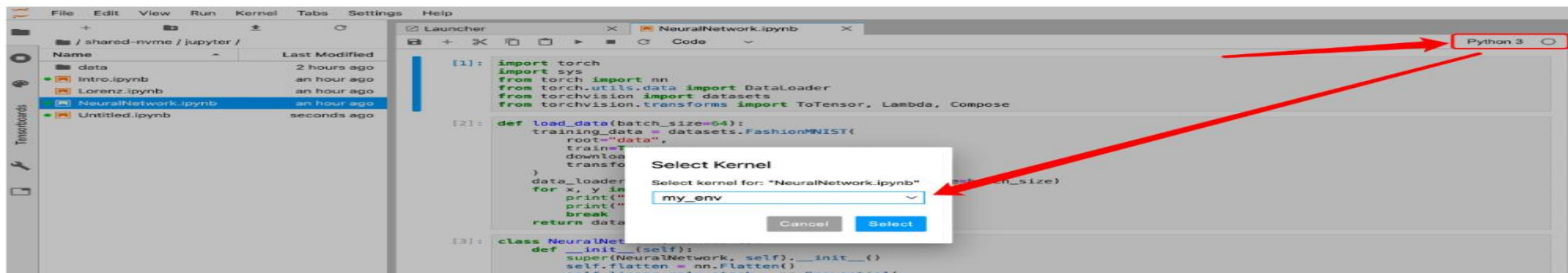
```
/opt/conda/bin/pip show ipython_genutils
```

➤ 命令行设置内核

- python -m ipykernel install --user --name [conda环境名称] --display-name [显示在jupyter的环境名称]

例：python -m ipykernel install --user --name my_env --display-name my_env

➤ 图形化设置内核



使用注意事项：学术加速

➤ 学术网络加速代理服务

➤ 使用方式：命令行执行

```
export https_proxy=http://u-UE25Z3:tXGJgV92@10.255.128.102:3128
```

```
export http_proxy=http://u-UE25Z3:tXGJgV92@10.255.128.102:3128
```

```
export no_proxy="127.0.0.0/8,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16,*.paracloud.com,*.paratera.com,*.blsc.c
```

```
- .github.com  
- .githubusercontent.com  
- .huggingface.co  
- .pypi.org  
- .wandb.ai  
- .gcr.io  
- .ghcr.io  
- .nvcr.io  
- .quay.io  
- .k8s.io  
- .pkg.dev  
- .googleapis.com  
- .pythonhosted.org  
- .githubassets.com
```



加速代理服务对以下网络进行加速

使用注意事项：终端复用

➤ 主要解决SSH断连的问题

• 使用方式：screen

新建一个session: `screen -S my`

`screen -ls` 查看所有的会话

`screen -r my` 进入到会话中

离开当前会话: `ctrl A + D`

• 使用方式：tmux

`sudo apt install tmux` (安装复用工具)

`tmux new -s download` (新建对话) `-s download` 指新会话的名称

`tmux ls`查看会话download

`tmux a -t download` (恢复download对话)

参考网址: <https://zhuanlan.zhihu.com/p/667646001>

演示教程：如何在AI智算云平台上部署Llama 3模型

 北京超级云计算中心



扫一扫，观看视频

#微信扫码，即可观看教程视频#

如果觉得视频还不错，点个赞吧！

实例演示

➤ 参考 torch 官网教程 提供的一个简单的 DDP 用例（此处仅展示部分）

```
import torch
import torch.distributed as dist
import torch.nn as nn
import torch.optim as optim

from torch.nn.parallel import DistributedDataParallel as DDP

class ToyModel(nn.Module):
    def __init__(self):
        super(ToyModel, self).__init__()
        self.net1 = nn.Linear(10, 10)
        self.relu = nn.ReLU()
        self.net2 = nn.Linear(10, 5)

    def forward(self, x):
        return self.net2(self.relu(self.net1(x)))
```

Vscode

<https://ai.blsc.cn/document/container/faq/remote/vscode>

实例演示

查看gpu利用率

```

[root@grid5 ~]# nvidia-smi
Mon Jun 26 23:42:23 2017
+-----+-----+-----+-----+-----+-----+
| NVIDIA-SMI 384.37 | Driver Version: 384.37 |
+-----+-----+-----+-----+-----+
| GPU  Name          Persistence-M | Bus-Id  Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap | 总线    Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+-----+-----+
|  0  Tesla M60      On          | 0000:04:00.0  Off  |             0%      Off |
| N/A   46C    P8     23W / 150W | 18MiB / 8191MiB |             Default |
+-----+-----+-----+-----+-----+-----+
|  1  Tesla M60      On          | 0000:05:00.0  Off  |             0%      Off |
| N/A   40C    P8     23W / 150W | 13MiB / 8191MiB |             Default |
+-----+-----+-----+-----+-----+-----+
|  2  Tesla P100-PCIE... On          | 0000:06:00.0  Off  |             0%      Off |
| N/A   47C    P0     31W / 250W | 8207MiB / 16383MiB |             Default |
+-----+-----+-----+-----+-----+-----+
| Processes:                 vGPU运行在第二块GPU上          GPU Memory |
| GPU      PID  Type  Process name                    Usage |
+-----+-----+-----+-----+-----+-----+
|  2          1933  C+G  /usr/lib64/xen/bin/vgpu          8192MiB GPU |
+-----+-----+-----+-----+-----+-----+

```

驱动常驻模式
 系统占用显存数量
 显存核心利用率
 GRID驱动版本
 显存总大小
 默认模式为计算
 GPU类型
 Persistence-M
 总线
 Memory-Usage
 GPU-Util
 Compute M.
 ECC关闭
 0代表第一块GPU
 当前GPU温度
 当前功率和总功率
 当前显存使用量
 vGPU显存为8GB
 GPU Memory Usage
 支持图形和计算

使用小结

1. 实例创建与平台相关实例操作 (12-14页)

2. 常用连接方式与文件传输 (15-18页)

SSH连接、JupyterLab连接、TensorBoard连接

3. 环境配置 (22-24页)

(1) conda环境配置

(2) 常见包安装

(3) 安装测试

4. 四个常见注意事项

(1) conda设置共享存储 25页

(2) JupyterLab内核设置 26页

(3) 学术加速 27页

(4) 终端复用 28页

算力资源使用问题请在微信群里@AI智算云技术工程师



AI智算云技术工程师

希望大家享受这次实操之旅！！

今晚19:00-21:00 进行上机实习❤️

构建云上科研工作环境 让计算更简单

