d.run 算力云 · 算力市场

算力云用户手册

容器实例 · 文件存储 · 我的镜像 全流程使用指南

文档名称	算力云用户手册
产品名称	d.run 算力云（Zestu）
所属平台	d.run 算力平台
产品定位	按需购买 GPU 算力、快速创建容器实例的开放式算力服务平台
内容范围	算力市场、容器实例、包年包月、磁盘存储、开发环境、Docker、文件存储、我的镜像、DeepSeek 部署、常见问题
整理来源	https://docs.daocloud.io/tf/zestu/
更新时间	2026 年 9 月

本手册基于 DaoCloud 官方文档中心「算力云」全站页面整理，供实施与运维人员参考。

目录

一、算力云介绍

算力云（Zestu）是 d.run 搭建的开放式算力服务平台，聚合来自独立算力资源供应商的多种 GPU 计算资源与配套服务。您可以在**算力市场**中按需浏览、筛选并购买不同规格的 GPU 资源，选择合适的计费模式，快速创建满足业务需求的**容器实例**，实现高效、灵活的算力获取与使用。

算力云围绕容器实例提供了一整套配套能力：

- **容器实例**：轻量级、可弹性伸缩的计算资源单元，广泛用于算法开发、模型训练与推理等场景。
- **文件存储**：基于实例的共享存储服务，同一用户创建的所有容器实例均可访问并共享同一套文件数据。
- **我的镜像**：基于 Harbor 的专属镜像仓库服务，可将容器实例的系统盘打包为镜像并持久保存、一键复用。

（一）算力市场

算力市场是您进入算力云的默认页面。进入 d.run 平台后，默认即进入**算力市场**，可在此按 GPU 型号、地区、计费方式等维度筛选资源，选择合适的规格后点击**立即购买**即可进入容器实例创建流程。

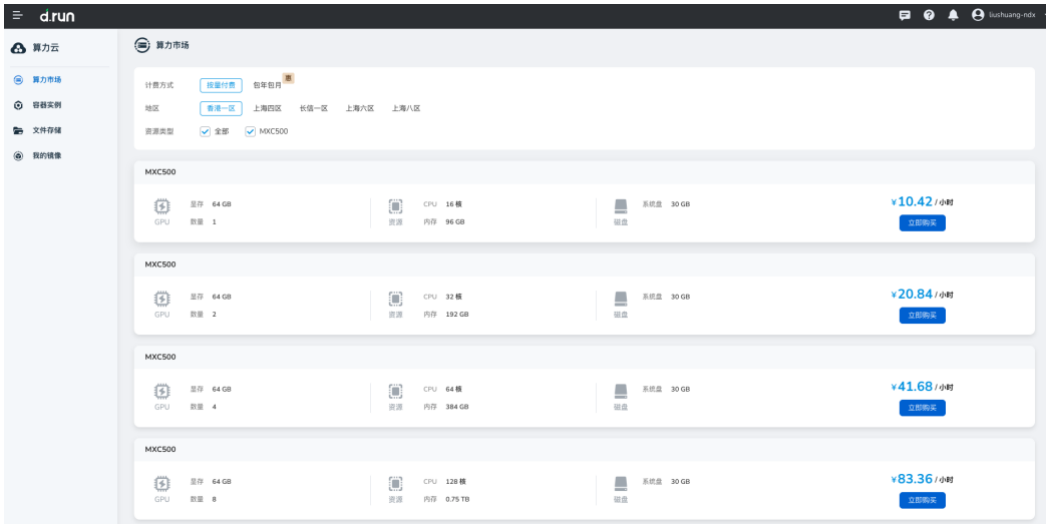

图 1　算力市场

（二）计费模式与资源类型

算力云提供**按量付费**与**包年包月**两种计费模式，并支持多种 GPU 资源类型，可在库存充足时按需选择。

项目	说明

按量付费	费用 = 时长 × 单价。时长 = 关机时间 – 开机时间，精确到秒，开机或等待过程中不计费。支持按小时计费，不足一小时将按实际使用时长扣费，同样精确到秒。
包年包月	按月或按年预留资源，单卡价格更优惠，租用时一次性付清。适合长期稳定使用算力的场景。
资源类型	即 GPU 的资源类型（型号）。当库存充足时，可根据业务需求选择购买；库存不足时需更换型号或稍后重试。

> **两种计费模式的差别**
> 按量付费按实际运行时长计费，关机即停止计费（不预留 GPU 资源）；包年包月为预付费订阅，**关机后 GPU 资源仍为该实例预留**，因此关机状态下依旧正常计费，但重新开机无需等待资源分配。

在算力市场选定规格后，点击**立即购买**即可前往创建容器实例，具体步骤请参见二、创建容器实例。

二、创建容器实例

容器实例是一种轻量级、可弹性伸缩的计算资源单元，广泛应用于算法开发、模型训练和推理等场景。通过容器实例，您能够快速启动独立的运行环境，实现算力的按需分配与高效利用。

创建容器实例时，支持多种 GPU 类型选择，满足不同算力需求，从入门级到高性能计算均有覆盖。实例会自动关联文件存储，方便统一管理数据和模型文件，提升数据处理效率。此外，容器实例支持使用系统预置的标准镜像，帮助您快速搭建开发环境，减少环境配置时间；开发者还可以通过 Jupyter Notebook、VS Code 远程插件及 SSH 等多样化方式访问实例，轻松进行代码调试与算法实验。

（一）前提条件

- 已登录 d.run 账号
- 账户余额大于等于所选资源类型的单价

（二）操作步骤

1. 登录 d.run 平台，默认进入**算力市场**，可选择所需 GPU 资源类型后点击**立即购买**；或切换到**容器实例**，点击**创建**按钮进行创建。

2. 参考下列要求填写基本信息，并点击**确定**。

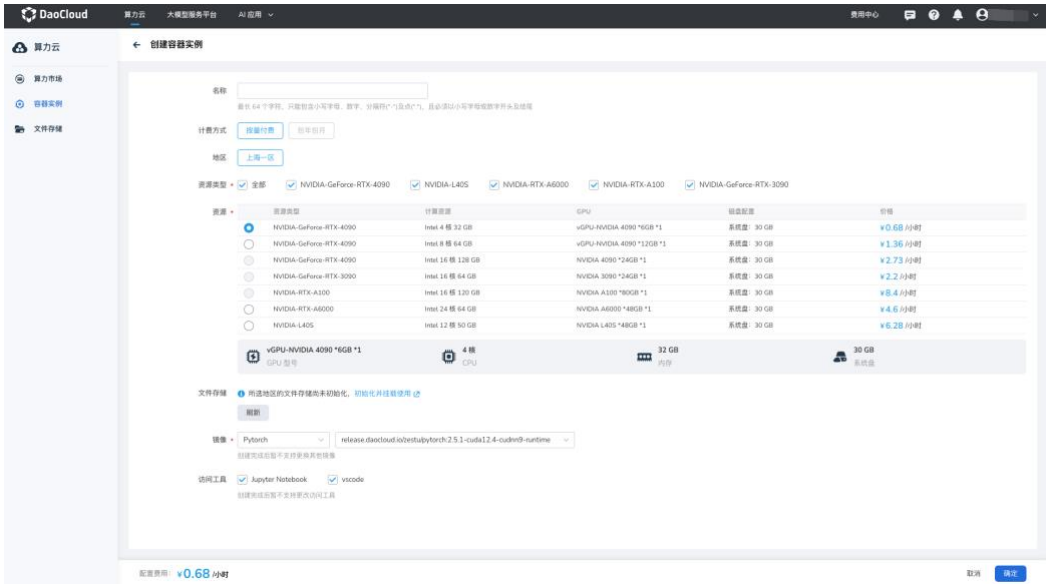

图 2　填写创建容器实例的基本信息

各配置项的填写要求详见（三）配置参数说明。

3. 点击**确定**，完成实例创建。

（三）配置参数说明

名称	说明
实例名称	最长 64 个字符，只能包含小写字母、数字、分隔符（"-"）及点（"."），且必须以小写字母或数字开头及结尾，非必填。
计费方式	— 按量付费：费用 = 时长 × 单价。时长 = 关机时间 – 开机时间，精确到秒，开机或等待过程中不计费。按小时计费，不足一小时将按实际使用时长扣费，精确到秒。 — 包年包月：按月/年预留资源，租用时一次性付清。若选择该计费模式，需配置购买时长、是否自动续费及续费时长等参数。
地区	建议选择与自己相近的地区创建实例和初始化文件存储，文件上传和下载速度更快。
资源类型	GPU 资源型号。
资源	支持整卡 GPU 和 vGPU 资源，根据需求选择。
系统盘	免费容量为 30 GB，不支持扩容。系统盘可保存至我的镜像。
数据盘	— 免费容量为 50 GB，支持付费扩容，实例关机状态下不停止计费。 — 可扩容容量默认最大为 200 GB，更大配额请联系 400-002-6898。扩容后无法进行缩容。 — 与容器实例的生命周期强绑定，实例创建时自动挂载，实例删除时自动删除。 — 不支持实例间的数据共享。
文件存储	初始化文件存储后，将自动挂载到实例中，默认挂载路径为：/root/data。

镜像	— 基础镜像：支持 Pytorch、TensorFlow、Paddle、vLLM 和 SGLang 等。 — 应用镜像：支持 DeepSeek、ComfyUI 等热门 AI 应用镜像。 — 我的镜像：支持使用我的镜像中保存的镜像创建容器实例。 — 镜像地址：公有或用户私有镜像地址，私有镜像需输入用户名和密码。
访问工具	可选择通过 Jupyter Notebook 和 VS Code 访问该容器实例。若取消勾选，则不会安装相关组件，创建完成后暂不支持更改访问工具。
定时关机	支持指定时间关机。

（四）计费说明

> 容器处于**启动中、排队中、已关机、删除中**等状态下时不产生任何费用，仅处于**运行中**和**关机中**时正常计费。
> 按量计费的资源单价为运行该实例每小时产生的费用，实际使用不满一小时，则按实际使用时长计费，精确到秒。

三、包年包月容器实例

包年包月容器实例是一种预付费的计算资源订购模式，为需要长期稳定算力的用户提供更具成本效益的解决方案。通过包年包月计费方式，用户可以享受相比按量付费更优惠的价格，同时获得资源预留保障，确保所需的 GPU 算力始终可用。

相较于按量付费模式，包年包月实例在关机状态下仍会保留 GPU 资源，用户可以随时重新启动实例而无需等待资源分配。此外，d.run 平台还提供灵活的自动续订方式和实例管理功能，帮助用户更好地控制成本和资源使用。

（一）适用场景

包年包月模式特别适合以下场景：

- **长期项目开发**：持续进行的算法研发、模型训练等需要稳定算力支持的项目。
- **生产环境部署**：需要 7×24 小时稳定运行的推理服务和应用部署。
- **成本优化需求**：通过预付费方式降低长期使用成本，获得更好的资源性价比。
- **资源保障要求**：确保关键业务所需的 GPU 资源得到预留，避免因资源紧张导致的业务中断。

（二）操作步骤

前提条件

- 已登录 d.run 账号
- 账户余额大于等于所选订阅套餐的总价
- 了解所需的 GPU 资源类型和配置要求

1. 登录 d.run 平台，选择**算力云**，默认进入**算力市场**，现已支持选择**包年包月**。然后选择所需 GPU 资源类型后点击**立即购买**；或切换到**容器实例**，点击**创建**按钮进行创建。

图 3　在算力市场选择包年包月

2. 在创建容器实例页面中，将**计费方式**选择为**包年包月**；配置包年包月相关参数，并填写其他实例信息（如实例名称、地区、资源类型、镜像等），点击**确定**完成购买。

图 4　选择包年包月计费并配置订阅参数

> 关于其他实例配置参数（如地区、资源类型、系统盘、文件存储、镜像、访问工具等）的详细说明，请参考二、（三）配置参数说明。

（三）配置参数说明

选择包年包月计费方式后，需要配置以下订阅相关参数：

参数名称	说明
购买时长	支持选择 1 - 12 个月的订阅期限。系统会根据选择的时长显示相应的折扣信息（如有）。时长越长，通常享受的折扣越大。

自动续订方式	支持三种到期处理方式： — **到期后自动续期（包年包月）**：到期时自动按相同计费模式续费。 — **到期后自动关机**：到期时自动关闭实例，停止计费。 — **到期后转按需计费**：到期时自动转换为按量付费模式。
续费时长	当选择"到期后自动续期"时，可配置自动续费的时长，支持 1 - 12 个月选择。

购买时长越长，享受的价格折扣通常越大，适合长期稳定使用的场景。
建议根据项目周期合理选择购买时长，避免资源浪费或频繁续费。
自动续订方式可在实例创建后随时修改，提供灵活的管理方式。

（四）包年包月实例管理

1. 修改自动续订方式

包年包月实例创建成功后，支持随时修改自动续订方式：

1. 进入**算力云 → 容器实例**，找到目标实例。
2. 点击实例操作菜单，选择**更换续订方式**。
3. 根据需要调整续费方式和续费时长。

图 5　修改自动续订方式

2. 释放实例

包年包月实例支持提前释放，系统会自动计算退款金额：

- **退款计算**：根据剩余使用时长按比例计算退款金额。
- **手续费扣除**：提前退款需要扣除一定的手续费。
- **退款方式**：退款金额将自动返还到账户钱包中。

图 6　释放实例并确认退款金额

3. 关机与资源预留

包年包月实例的关机行为与按量付费实例有重要区别：

- **资源预留**：关机后 GPU 资源将继续为该实例预留，确保随时可以重新启动。
- **费用计算**：关机状态下仍按订阅费用正常计费，不会停止扣费。
- **快速启动**：由于资源已预留，重新开机时无需等待资源分配，可立即启动。

（五）续订机制

如包年包月实例的续订配置为**到期后自动续期（包年包月）**，平台为确保服务的连续性，会提前进行续费。

1. 时间安排

- **提前扣费**：系统会在实例到期前 7 天开始尝试自动扣费。
- **每日尝试**：从到期前 7 天开始，系统每天都会根据设置的续订方式尝试扣费。
- **多次保障**：通过多次尝试机制，最大程度避免因临时余额不足导致的服务中断。

2. 扣费成功与失败处理

扣费成功：

- 按照设置的续订方式自动完成续费。
- 实例继续正常运行，无需人工干预。
- 系统发送续费成功通知。

扣费失败：

- 如果连续 7 天扣费都失败，实例将在到期后自动关机。
- 关机后停止计费，但会保留实例配置一段时间。
- 用户充值后可手动重新启动实例（启动实例时，可以选择继续包年包月或直接转按需计费）。

包年包月实例一旦创建，订阅期内将持续计费。
请确保账户余额充足，避免因余额不足导致的自动关机。
可随时修改续订方式，灵活应对业务需求变化；修改续订方式不会影响当前订阅期，仅在下次续费时生效。
释放实例操作不可撤销，请谨慎操作并确认数据已备份。

包年包月实例从创建成功时开始计费，直到订阅期满或主动释放。
关机状态下仍会继续计费，但保证资源预留和快速启动能力。
提前释放实例会按剩余时长比例退款，但需扣除相应手续费。
如需更多帮助或遇到问题，请联系客服热线：**400-002-6898**。

四、容器实例日常管理

容器实例创建完成后，日常使用中最常见的操作包括**开关机**、**更换配置**以及**磁盘存储**管理。合理使用这些能力，可以在保证业务连续性的同时有效控制算力成本。

（一）容器实例开关机

容器实例在创建完成后，支持通过 UI 界面、控制台或 API 执行**开机**与**关机**操作，以便灵活管理算力资源的使用与成本。

- **开机**：启动容器实例后，计算资源（如 CPU、GPU、内存等）将被分配并投入运行，系统开始按照所选计费模式计费。
- **关机**：关机操作会释放计算资源，同时**停止计费**，但保留实例的系统盘数据和配置，方便后续快速恢复运行。

对于**间歇性运行任务**或**按需使用算力**的场景，例如模型训练、批量计算或测试环境部署，通过合理安排开关机时间，可在保证业务连续性的同时有效降低资源成本。

1. 关机

前提条件：已创建容器实例，且容器实例处于运行中。

操作步骤如下：

1. 登录 d.run，进入**算力云 → 容器实例**，选择一个运行中的实例，点击列表右侧的 ⋮，在下拉列表中选择**关机**。

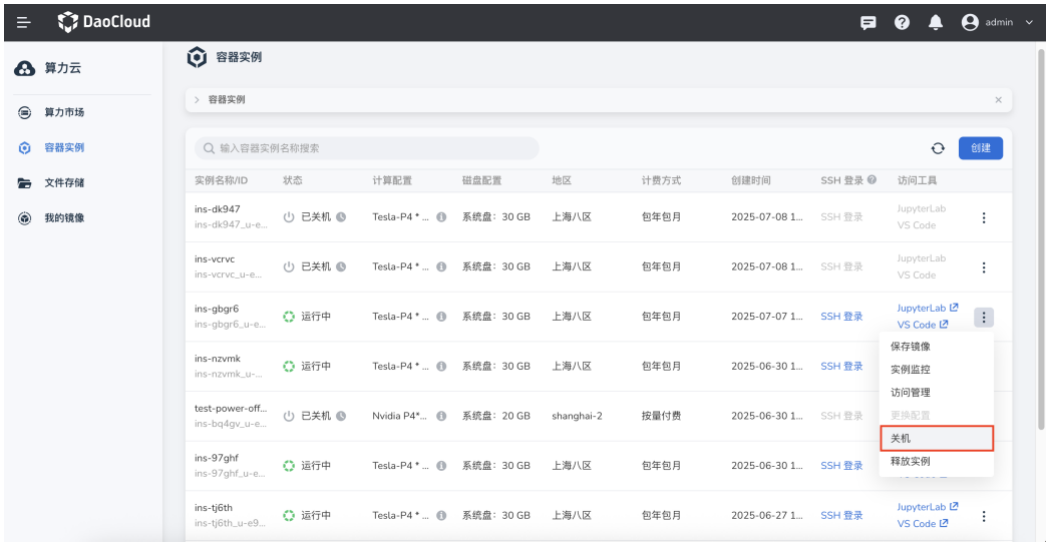

图 7　在实例列表中选择关机

2. 在弹出的提示框中，仔细阅读关机须知后，选择是否保存系统盘，然后点击**确定**。

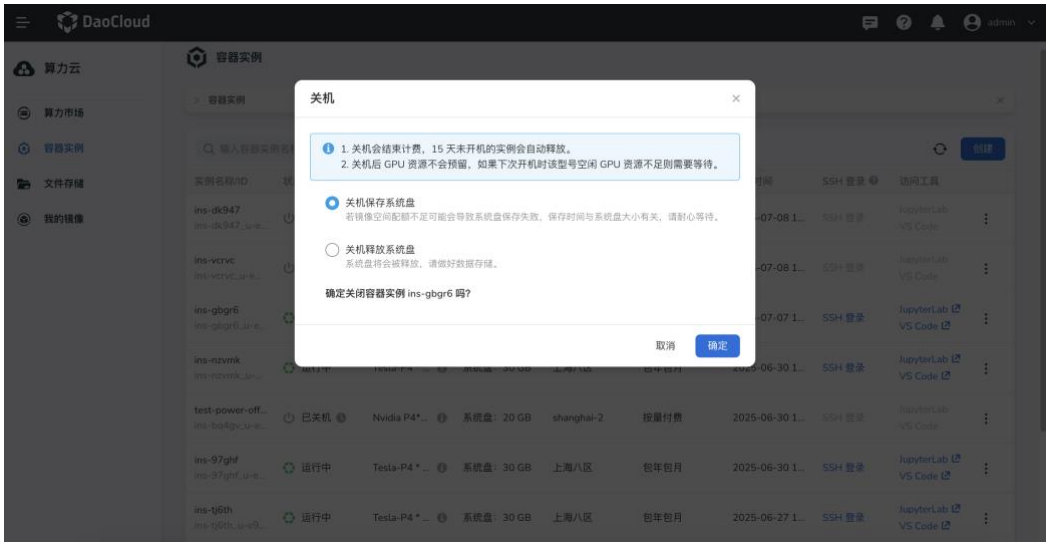

图 8　阅读关机须知并确认

15 天未开机的实例会自动释放。
关机后 GPU 资源不会预留，如果下次开机时该型号空闲的 GPU 资源不足则需要等待。
若镜像空间配额不足可能会导致系统盘保存失败，请确保镜像空间配额充足。

2. 开机

前提条件：已创建容器实例，且容器实例处于关机中。

操作步骤如下：

1. 登录 d.run，进入**算力云 → 容器实例**，选择需要启动的实例，点击列表右侧的 ⋮，在下拉列表中选择**启动实例**。

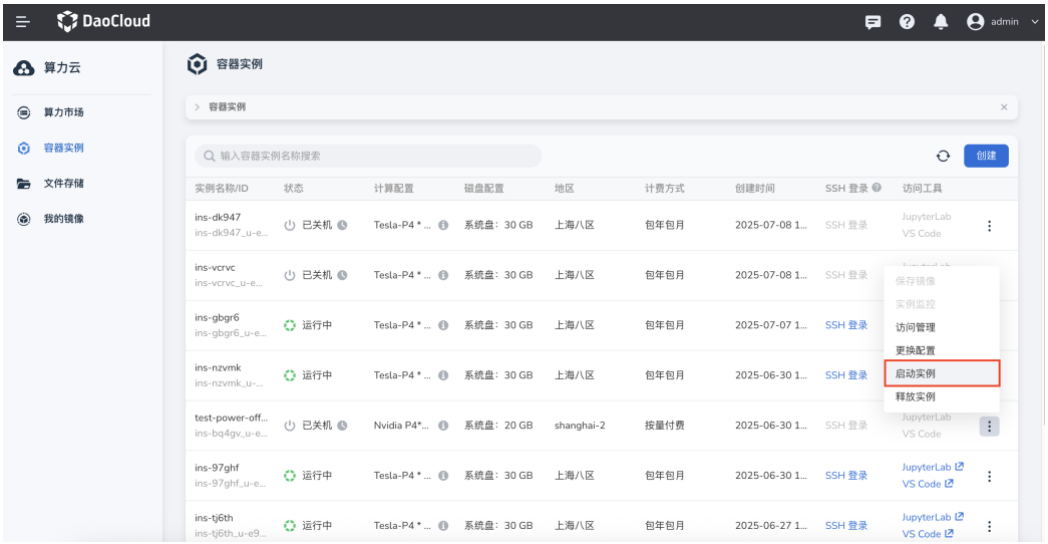

图 9　在实例列表中选择启动实例

2. 若所需的 GPU 资源充足，点击**确定**即可启动。资源不足时可能需要等待。

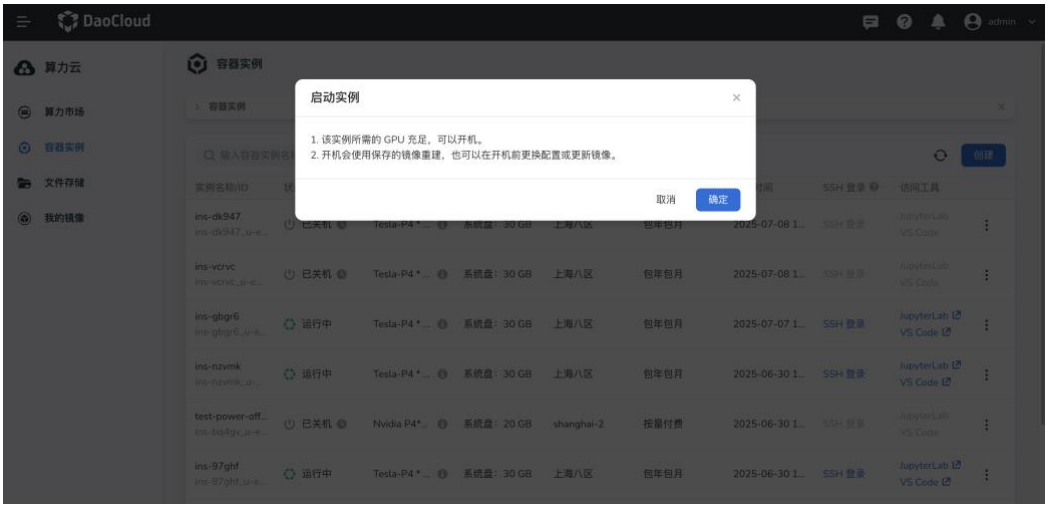

图 10　确认启动实例

> 容器实例处于排队中或者启动中时不计费，运行状态下开始计费。
> 容器实例不会主动刷新状态，需手动刷新查看容器实例的真实状态。

（二）容器实例更换配置

容器实例关机后，可以通过**更换配置**选择新的 SKU，执行升配或降配操作。

前提条件：已创建容器实例，且容器实例处于**已关机**状态。

1. 进入**算力云 → 容器实例**，找到状态为**已关机**的实例，点击列表右侧的 ⋮，在下拉列表中选择**更换配置**。

图 11　更换配置入口

2. 在**更换配置**页面选择新的 SKU，确认 GPU 类型、显存、数量、计算资源、磁盘配置及价格等信息后，点击**确定**完成升降配。

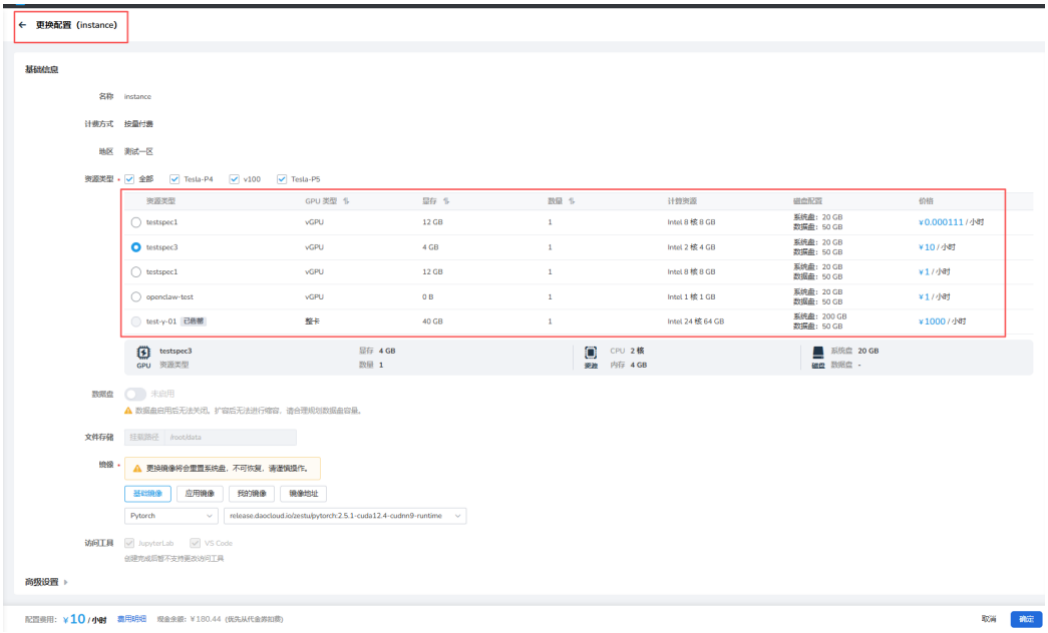

图 12　选择新的 SKU 并确认价格

> 更换配置仅支持处于关机状态的容器实例。
> 升降配完成后，容器实例不会自动开机，需要手动选择**启动实例**。

（三）磁盘存储

磁盘存储为容器实例提供了基础的存储空间，磁盘分为**系统盘**和**数据盘**。

1. 系统盘

系统盘是容器根文件系统存储（rootfs），d.run 为每个容器实例提供 30 GB 免费容量，主要功能如下：

- **基础功能**：系统盘可通过镜像的方式保存，详情可参考七、（五）保存镜像；重启后会使用保存的镜像。

- **高级功能**：开发机的系统盘提供持久化存储，开发机关机、重启后仍保存开发环境和数据。开发机删除后，系统盘存储空间将被清理。（如需使用请联系管理员）

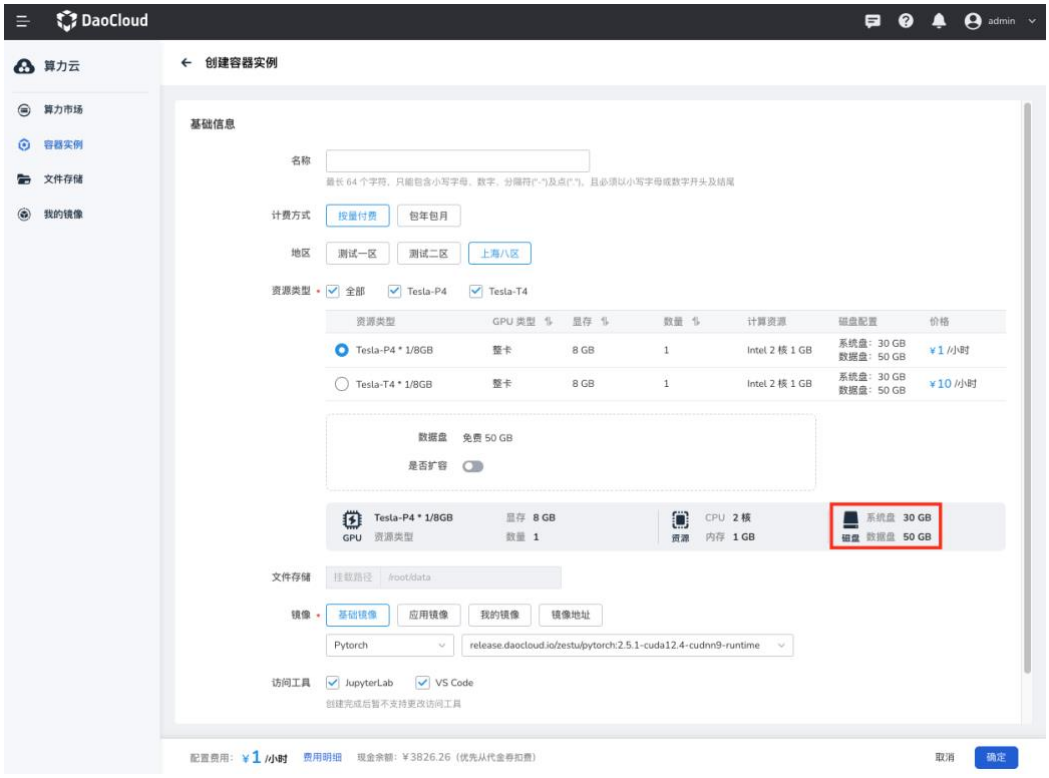

图 13　系统盘容量展示

2. 数据盘

数据盘是基于分布式块存储的持久化存储产品，免费容量为 50 GB。创建时自动挂载至 /root/tmp 路径下，具有强关联实例的生命周期，跟随实例一起创建及释放。数据盘支持付费扩容，不支持实例间的数据共享。

数据盘扩容

数据盘可扩容容量默认最大为 200 GB，更大配额请联系 400-002-6898。扩容后无法进行缩容，实例关机状态下不停止计费。

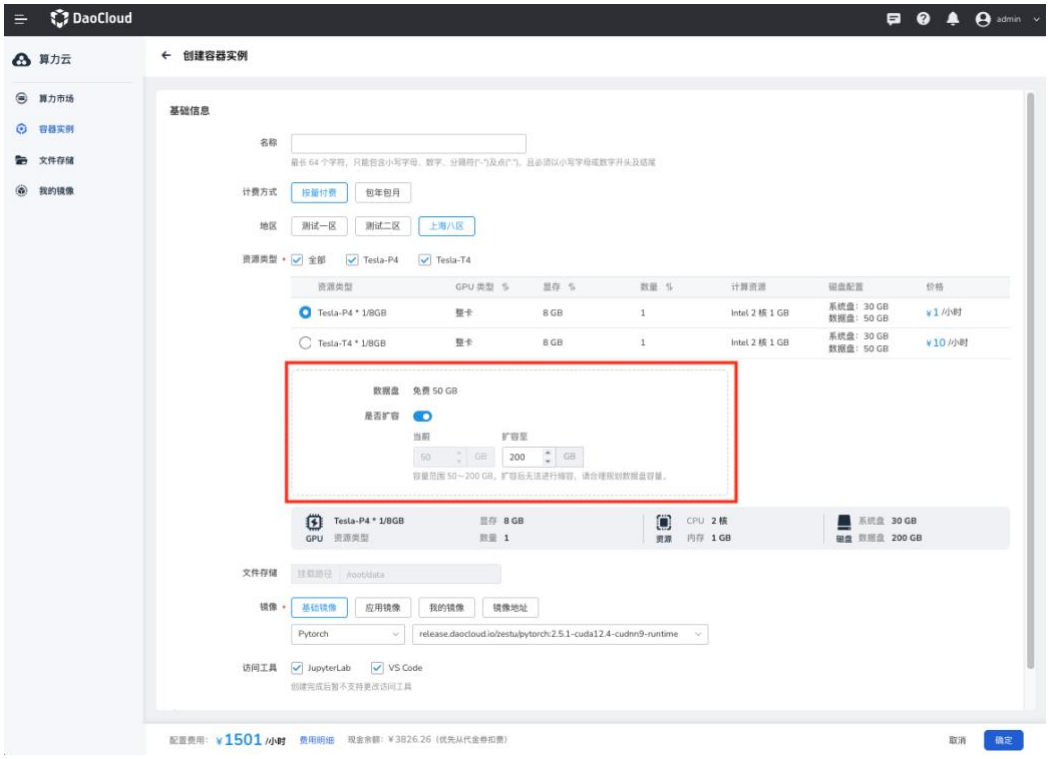

图 14　数据盘扩容配置

容器实例创建后，支持通过**更换配置**对数据盘进行多次扩容。

五、开发环境与访问方式

容器实例提供了多种访问与开发方式，您可以根据使用习惯选择 JupyterLab、VS Code 或 SSH 登录实例，也可以通过**访问管理**将实例内部服务暴露到公网。

访问方式	适用场景
JupyterLab	基于 Web 的交互式开发环境，适合运行 Notebook、执行 Shell 命令与 Python 代码。
VS Code	既支持云端快捷使用，也支持本地 VS Code 通过 Remote-SSH 插件连接远程开发。
SSH	命令行远程登录，支持用户名/密码与公钥免密两种方式，适合脚本化与自动化操作。
访问管理	将实例内任意端口暴露到公网，供外部人员访问实例中运行的服务。

（一）JupyterLab

JupyterLab 是基于 Web 的新一代 Jupyter 交互式开发环境，可以通过 Web 管理文件并执行 Shell 命令和 Python 代码，还支持插件扩展。JupyterLab 包含了 Jupyter Notebook 的全部功能。

下文介绍如何在 d.run 上使用 JupyterLab。

1. JupyterLab 基本操作

1). 创建容器实例时勾选访问工具 JupyterLab（默认勾选）。

图 15　创建实例时勾选 JupyterLab

2). 容器启动后，在容器实例列表中点击快捷访问链接打开 JupyterLab。

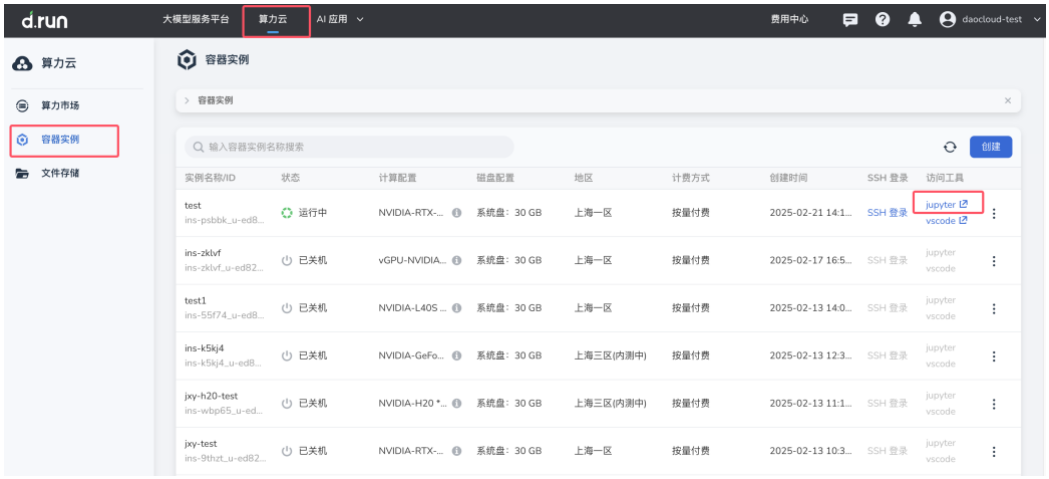

图 16　在实例列表中打开快捷访问链接

3). 进入启动页，左侧为文件浏览区，可以查看实例内的目录文件，右侧为工作区。

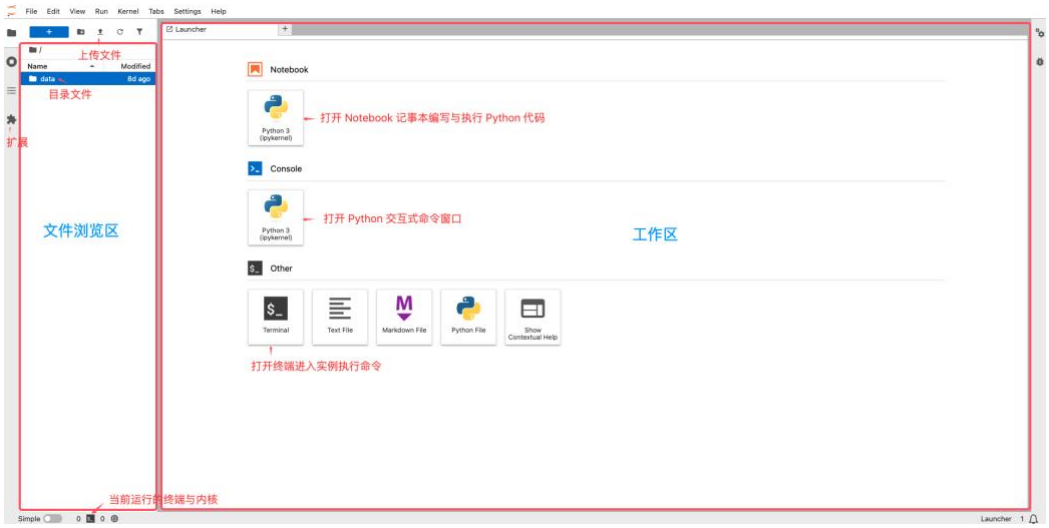

图 17　JupyterLab 启动页

2. 使用 JupyterLab

1). 在启动页中打开终端。

图 18　在 JupyterLab 中打开终端

2). 打开终端后可以直接执行命令。

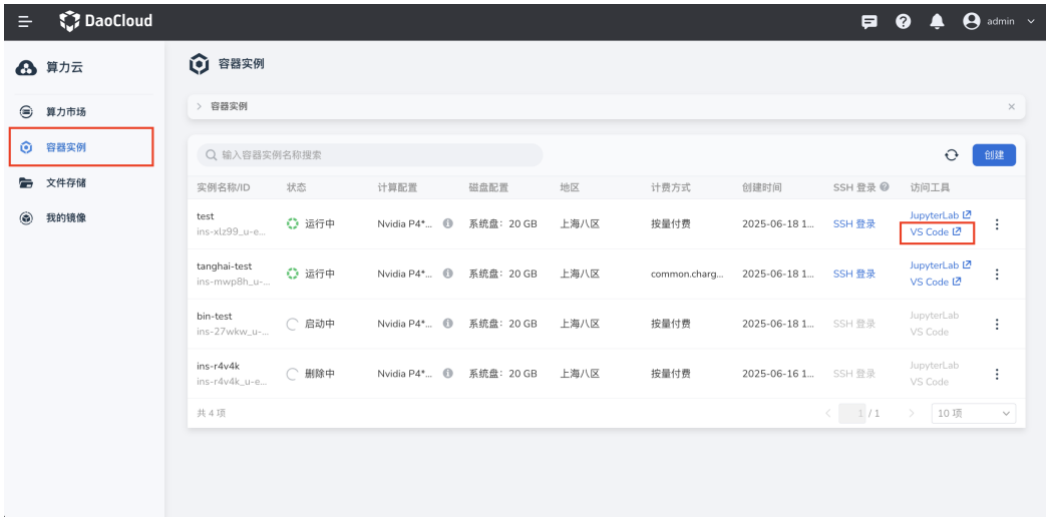

图 19　在终端中执行命令

使用完成后在终端内执行 logout 或按下 Ctrl+D 正常退出终端。

如果直接关闭了终端窗口，这个终端仍然会在后台继续运行，包括正在执行的命令任务。您可以在左侧菜单栏点击正在运行终端和内核按钮，查看运行中的终端。

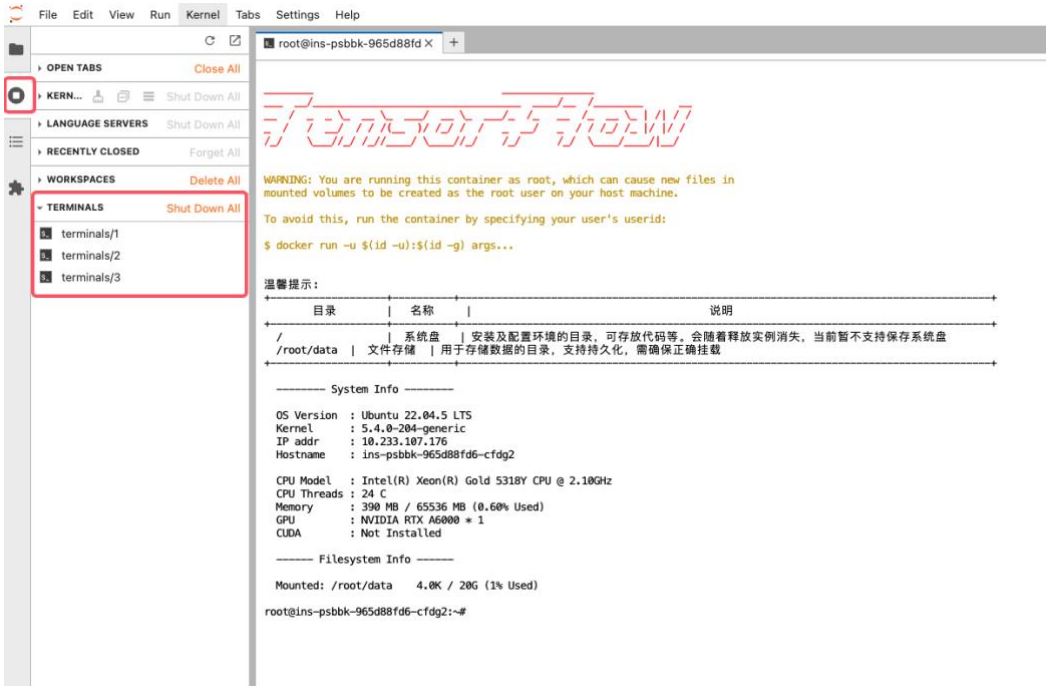

图 20　查看运行中的终端

3. 运行代码

1). 首先创建一个为 TensorFlow 2 的实例。

2). 下载 beginner.ipynb 记事本文件并通过 JupyterLab 上传到服务器中。双击打开记事本，在右方工作区中可以看到笔记本中的代码内容。

图 21　打开上传的 Notebook 文件

3). 在菜单中选择**运行 → 运行所有单元格**，即可运行所有代码。

图 22　运行所有单元格

4). 在工作区每个单元格下方可以看到程序的输出。

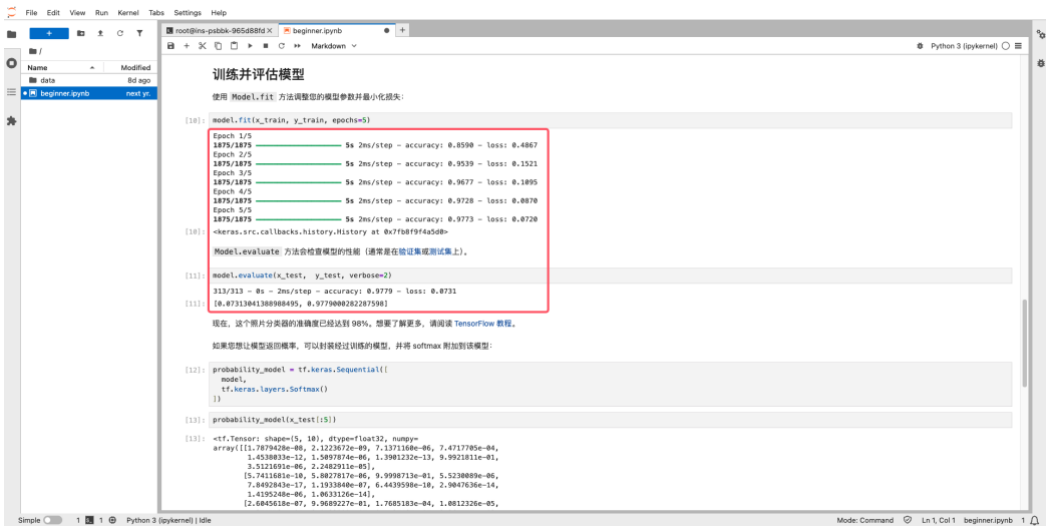

图 23　查看单元格执行输出

（二）VS Code

VS Code 是微软开发的一款跨平台代码编辑器，免费、开源并且支持扩展插件。平台既支持云端快捷使用 VS Code，也支持在本地使用 Remote 插件连接到远程服务器上进行开发。

1. 在 d.run 上使用 VS Code

1). 创建容器实例时勾选访问工具 VS Code（默认勾选）。

图 24　创建实例时勾选 VS Code

2). 容器启动后，在容器实例列表中点击快捷访问链接打开 VS Code。

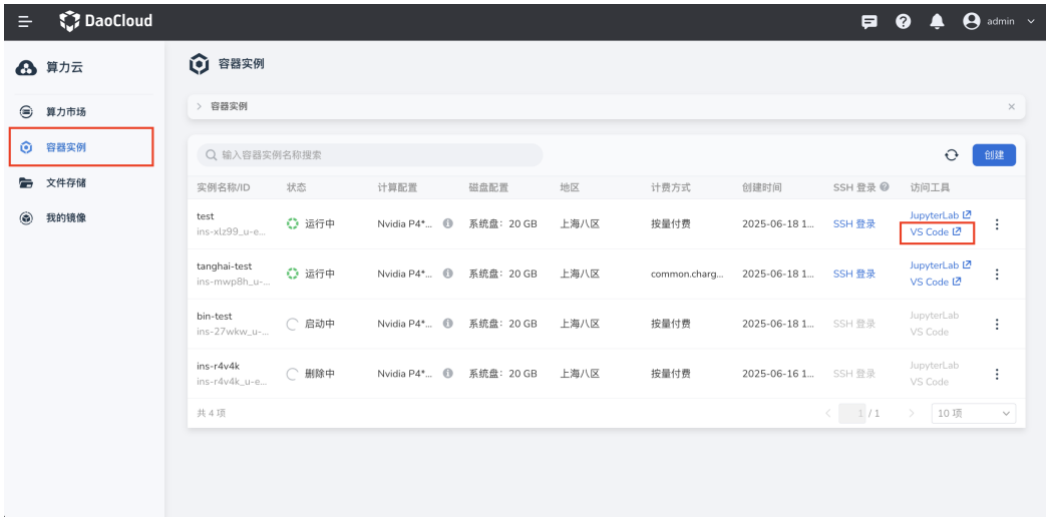

图 25　在实例列表中打开 VS Code

3). 进入 VS Code UI 页面，点击右上角 **Toggle Panel**，可打开 VS Code 底部的 Terminal 面板，用于输入并执行 Terminal 命令。左侧相关菜单栏，可进行**新建文件**等操作。

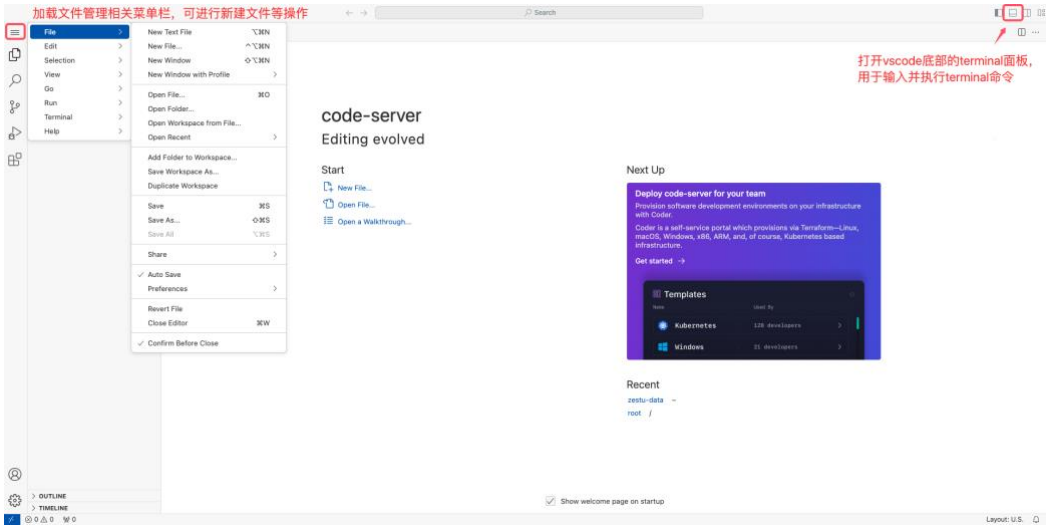

图 26　VS Code 界面与 Toggle Panel

4). 在左侧导航栏中，点击**扩展**，进入扩展列表，用户可根据实际需要安装相应扩展组件。在扩展列表中，选择待安装的组件（以 Python 扩展为例），打开其介绍页面，点击 **Install**，等待组件安装完成即可。

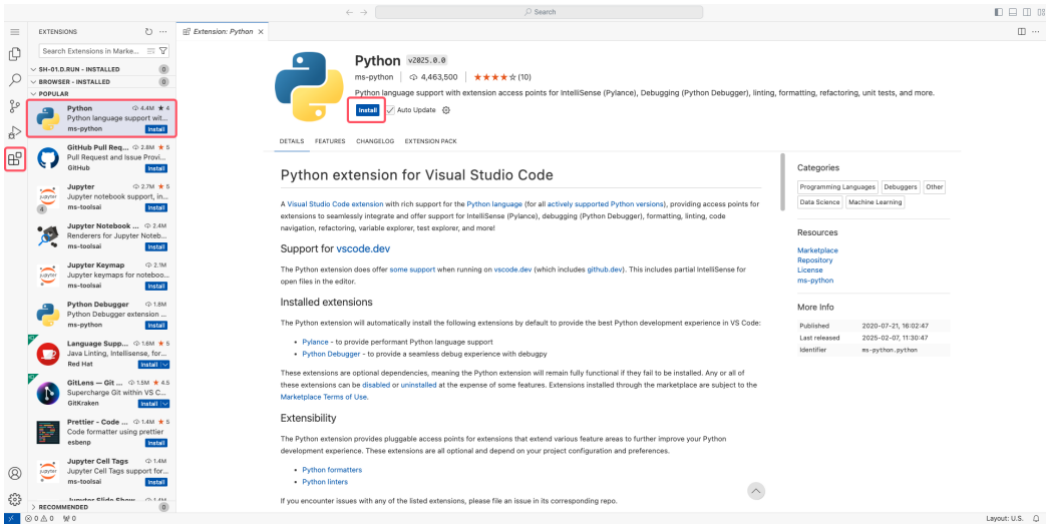

图 27　在扩展市场中安装 Python 扩展

2. 本地 VS Code 使用

1). 本地 VS Code 配置 Remote-SSH。

> 如果您本地 VS Code 开发工具已安装 Remote-SSH，可跳过此步骤，直接参考下一步。

打开本地的 VS Code 开发插件菜单，在扩展程序中搜索 Remote-SSH 并安装：

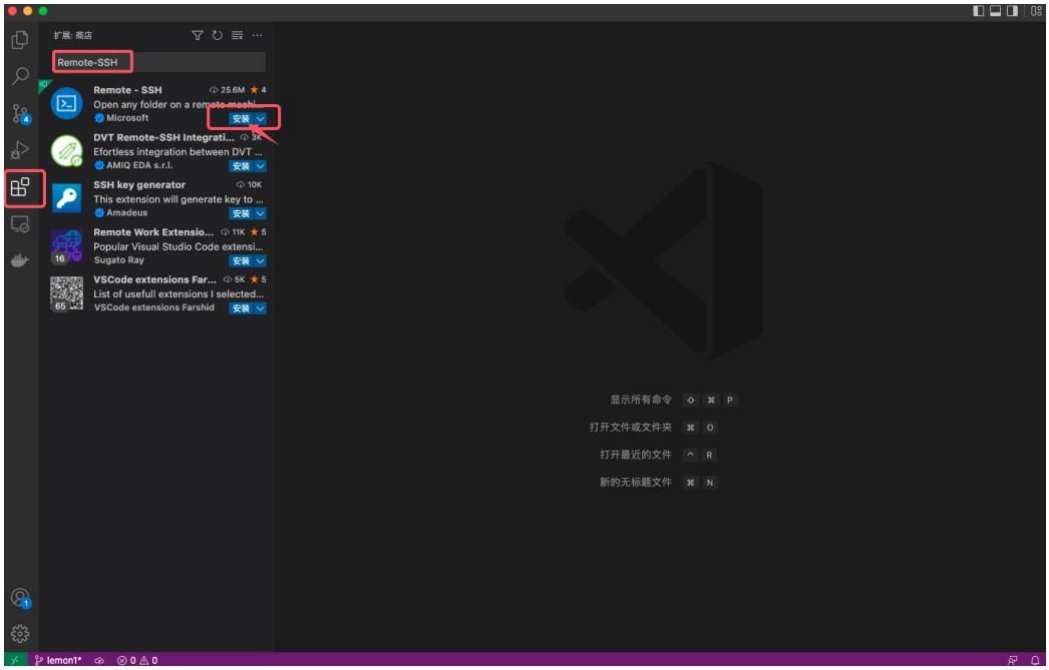

图 28　在本地 VS Code 中安装 Remote-SSH

2). SSH 连接并登录您平台上租用的实例。按照 ① ② ③ 顺序进行点击，完成添加 SSH 主机。

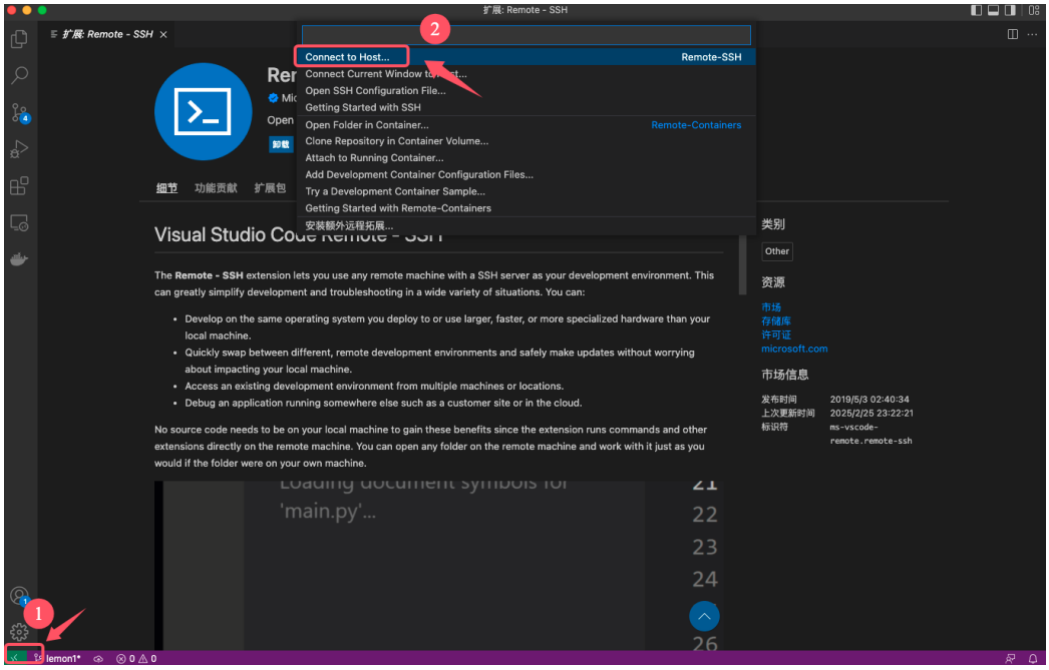

图 29　添加 SSH 主机

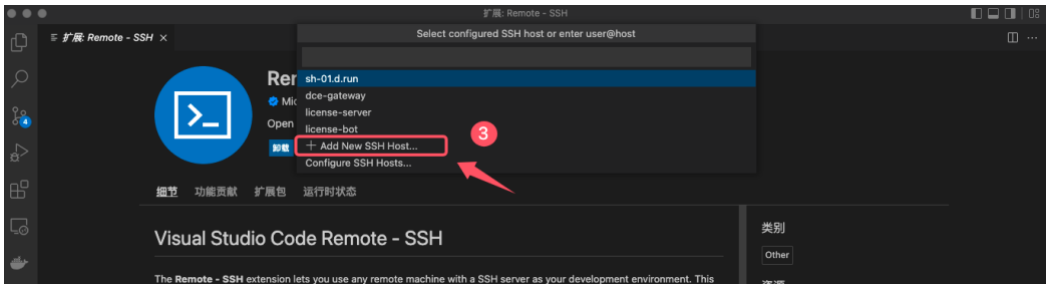

图 30　输入 SSH 主机连接信息

3). 复制并获取您实例的登录信息。

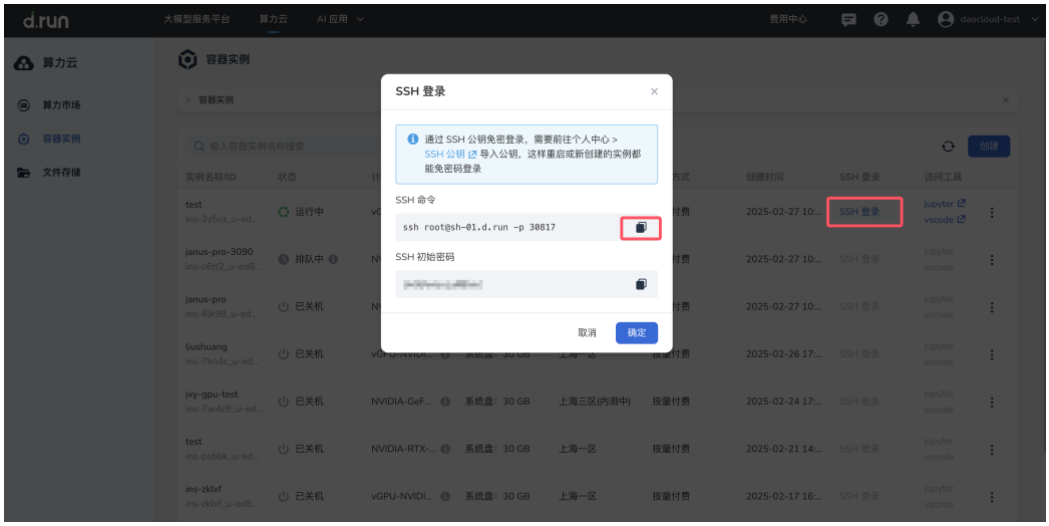

图 31　复制实例的登录信息

4). 下图以 ssh root@sh-01.d.run -p 30817 为例。回车键继续操作。

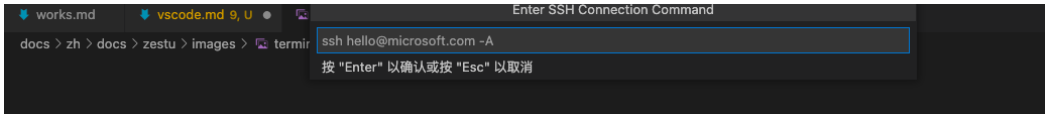

图 32　确认连接远程主机

远程主机信息会保存在本地配置文件中，选择第 1 个配置文件完成添加。

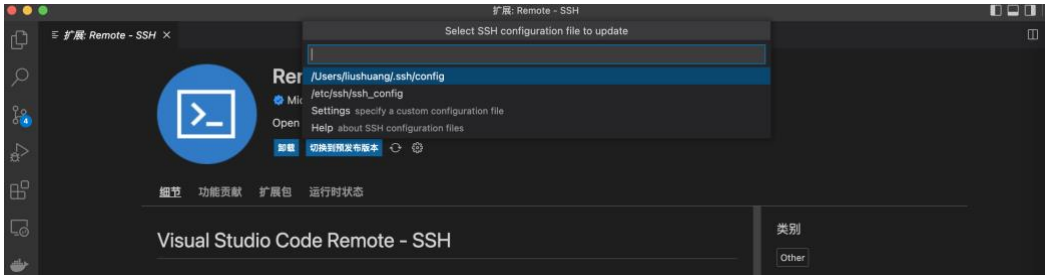

图 33　选择本地配置文件

5). 连接远程主机。添加完成后在**远程资源管理器**中显示刚添加的远程主机。点击在新窗口中打开连接的按钮。

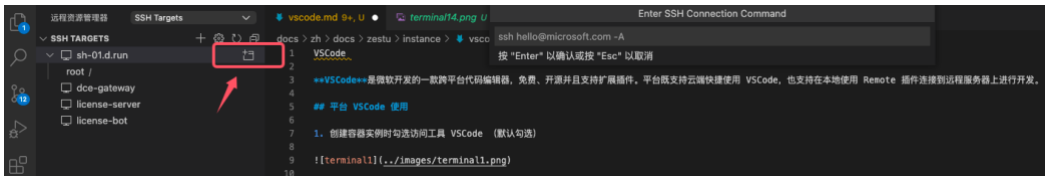

图 34　在远程资源管理器中打开连接

6). 复制并获取实例的登录密码，参考步骤 3 粘贴复制的登录密码后敲击回车键。

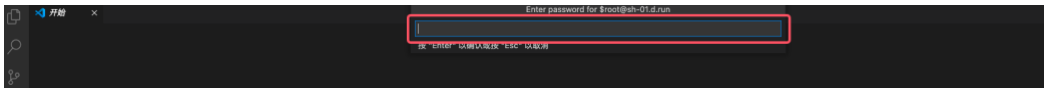

图 35　粘贴登录密码

7). 成功连接后点击**资源管理器**标签会显示已连接到远程。同时左下角提示已连接的远程主机名称。

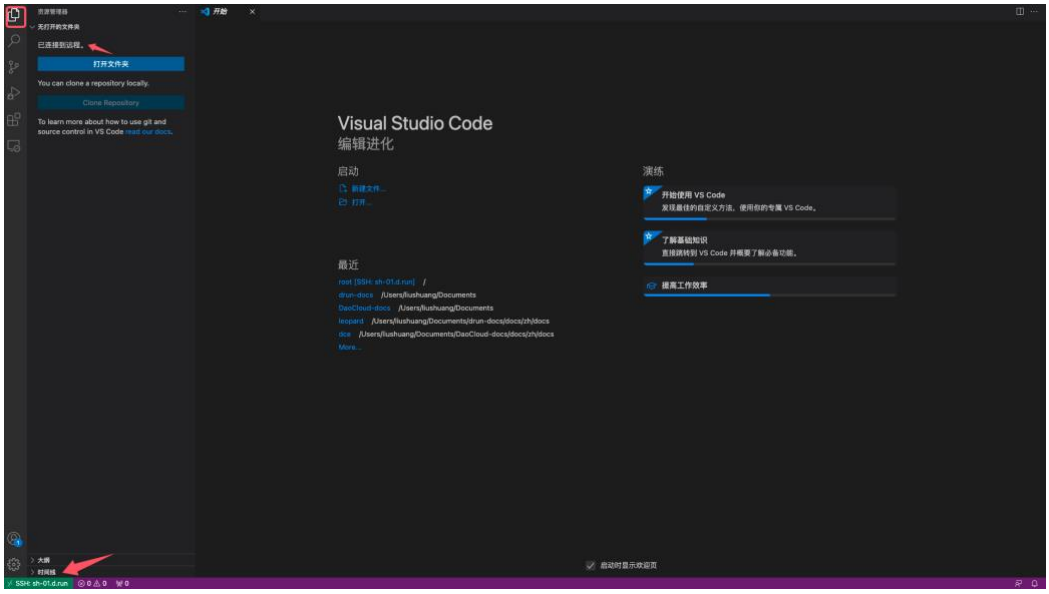

图 36　成功连接到远程实例

（三）SSH

SSH（Secure Shell）是一种网络协议，用于在不安全的网络中安全地进行远程登录和执行命令。通过 SSH，用户可以从本地计算机安全地访问平台上的容器实例。

1. 使用 SSH 用户名/密码登录容器实例

1). 在打开的 Terminal 终端中，安装 openssh-server 服务。若已安装可跳过此步骤，直接进入步骤 2。

（1）执行如下命令，安装 openssh-server 服务。

```
apt-get update && apt install openssh-server
```

（2）检查安装是否成功。

检查 ssh 进程：

```
ps -e | grep ssh
```

检查安装包：

```
dpkg -l | grep ssh
```

2). 容器启动后，在容器实例列表中点击 SSH 登录按钮，打开弹窗复制 SSH 的登录信息。

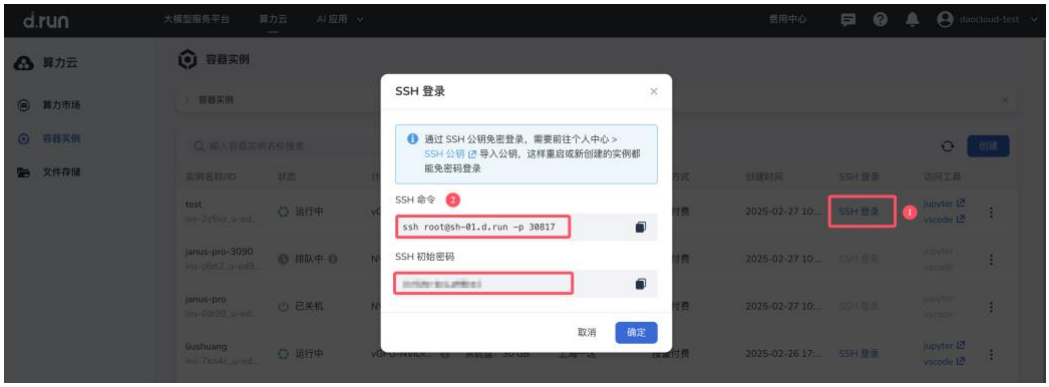

图 37　复制 SSH 登录信息

3). 在本地的 Terminal 终端中输入用户名/密码，返回如下信息表示登录成功。

图 38　SSH 登录成功

2. 使用 SSH 免密登录容器实例

在**个人中心 → SSH 公钥**导入公钥后，重启或新创建的实例都能免密码登录容器实例。

步骤一：查看 SSH 公钥

在生成新的 SSH 密钥前，请先确认是否需要使用本地已生成的 SSH 密钥，SSH 密钥对一般存放在本地用户的根目录下。Linux、Mac 请直接使用以下命令查看已存在的公钥，Windows 用户在 WSL（需要 Windows 10 或以上）或 Git Bash 下使用以下命令查看已生成的公钥。

ED25519 算法：

```
cat ~/.ssh/id_ed25519.pub
```

RSA 算法：

```
cat ~/.ssh/id_rsa.pub
```

如果返回一长串以 ssh-ed25519 或 ssh-rsa 开头的字符串，说明已存在本地公钥，您可以跳过步骤二（生成 SSH 密钥），直接进入步骤三（拷贝公钥）。

步骤二：生成 SSH 密钥

若步骤一未返回指定的内容字符串，表示本地暂无可用 SSH 密钥，需要生成新的 SSH 密钥，请按如下步骤操作：

1. 访问终端（Windows 请使用 WSL 或 Git Bash），运行 ssh-keygen -t。

2. 输入密钥算法类型和可选的注释。注释会出现在 .pub 文件中，一般可使用邮箱作为注释内容。

基于 ED25519 算法，生成密钥对命令如下：

```
ssh-keygen -t ed25519 -C "<注释内容>"
```

基于 RSA 算法，生成密钥对命令如下：

```
ssh-keygen -t rsa -C "<注释内容>"
```

3. 点击回车，选择 SSH 密钥生成路径。以 ED25519 算法为例，默认路径如下：

```
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/user/.ssh/id_ed25519):
```

密钥默认生成路径：/home/user/.ssh/id_ed25519，公钥与之对应为：

/home/user/.ssh/id_ed25519.pub。

4. 设置一个密钥口令。

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

口令默认为空，您可以选择使用口令保护私钥文件。如果您不想在每次使用 SSH 协议访问仓库时都要输入用于保护私钥文件的口令，可以在创建密钥时输入空口令。

5. 点击回车，完成密钥对创建。

步骤三：拷贝公钥

除了在命令行打印出已生成的公钥信息手动复制外，可以使用命令拷贝公钥到粘贴板下，请参考操作系统使用以下命令进行拷贝。

Windows（在 WSL 或 Git Bash 下）：

```
cat ~/.ssh/id_ed25519.pub | clip
```

Mac：

```
tr -d '\n'< ~/.ssh/id_ed25519.pub | pbcopy
```

GNU/Linux（需要 xclip）：

```
xclip -sel clip < ~/.ssh/id_ed25519.pub
```

步骤四：在平台上设置公钥

1. 登录 d.run UI 页面，在页面右上角选择**个人中心 → SSH 公钥**。

图 39　进入个人中心的 SSH 公钥

2. 添加生成的 SSH 公钥信息。

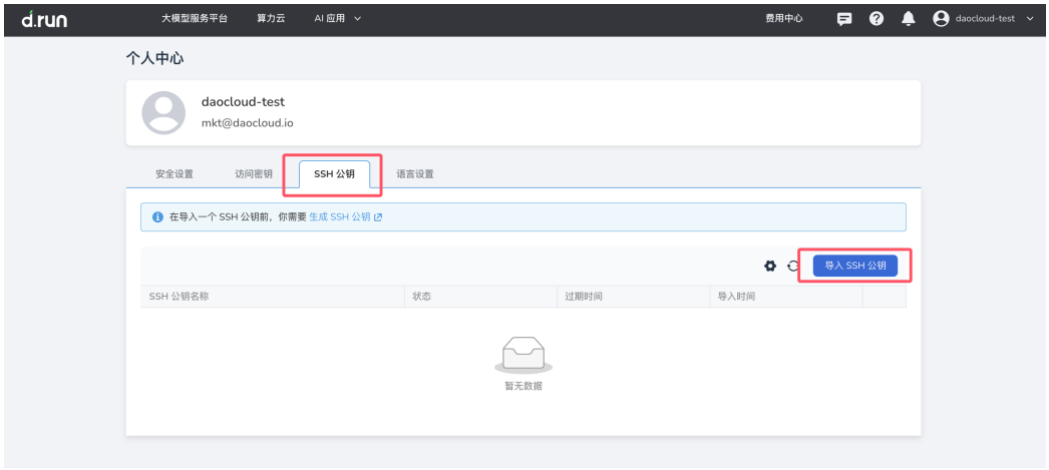

图 40　添加 SSH 公钥

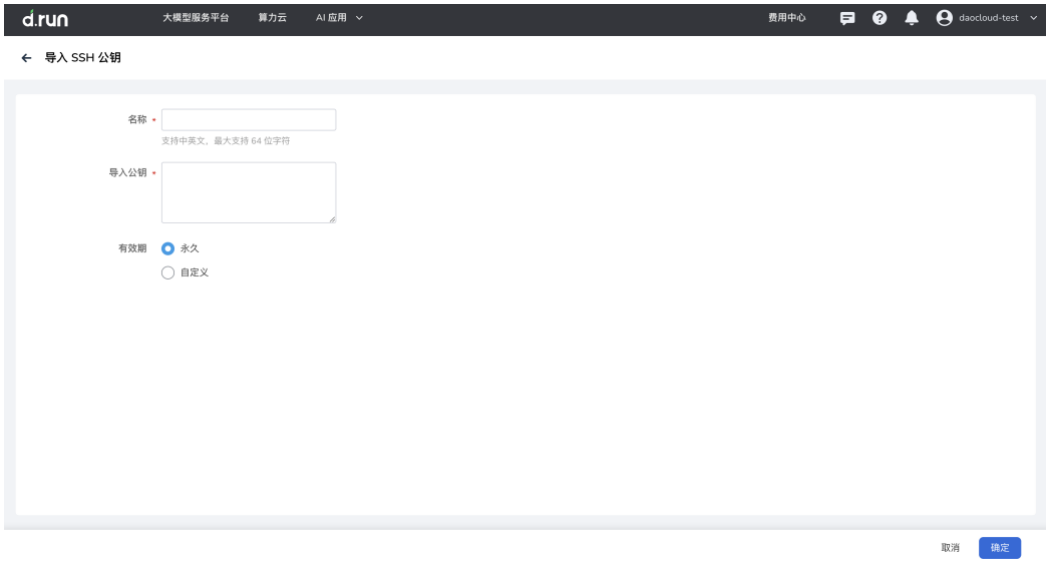

图 41　填写公钥标题与过期时间

- **公钥标题**：支持自定义公钥名称，用于区分管理。
- **过期时间**：设置公钥过期时间，到期后公钥将自动失效，不可使用；如果不设置，则永久有效。

（四）访问管理

访问管理是 d.run 平台给用户暴露服务到公网访问的一个功能，允许将任何端口暴露到公网进行访问（不包含已经预留服务的端口：22/5555/6666/10001/10002）。

本文以 nginx 为例，介绍如何使用容器实例的访问管理能力开放端口。开放端口生成的外部访问地址可供外部人员访问。

前提条件：

- 已登录 d.run 账号
- 已通过算力云创建容器实例，且容器实例状态为**运行中**

1. 登录 d.run 账号，在顶部导航栏点击**算力云**，然后点击左侧导航栏的**容器实例**，在容器列表中选择您要操作的容器实例。

图 42　容器实例列表

2. 在容器实例列表页面，点击 **SSH 登录**，通过 SSH 命令登录容器实例。

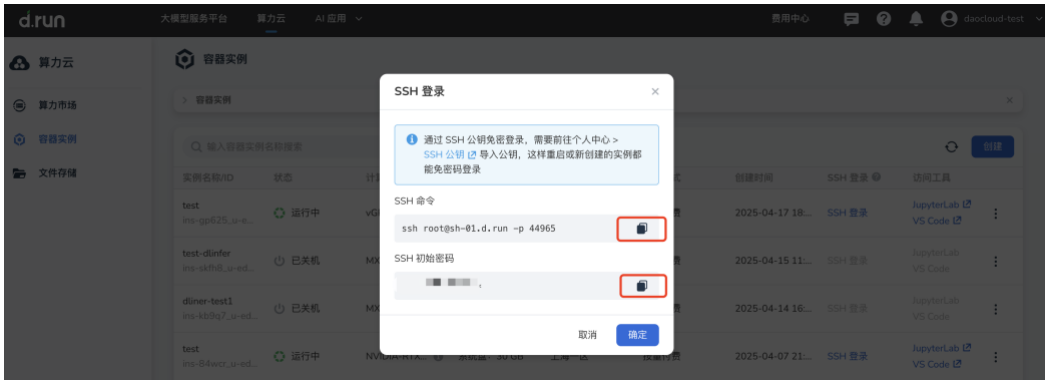

图 43　SSH 登录入口

图 44　在本地终端完成 SSH 登录

3. 安装并启动 nginx 服务。

```
apt update
apt install -y nginx && nginx && ps -ef|grep master
apt install lsof # 安装 lsof
```

4. nginx 默认端口是 80，检查一下 nginx 是否正常启动。

```
lsof -i:80
```

返回如下信息，表明端口被 nginx 正常占用。

```
COMMAND  PID USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
nginx   2693 root    6u  IPv4 723250562      0t0  TCP *:80 (LISTEN)
nginx   2693 root    7u  IPv6 723250563      0t0  TCP *:80 (LISTEN)
```

5. 将 nginx 的端口添加到访问管理，并等待外部端口的分配（约 1 分钟）。

图 45　在访问管理中添加端口

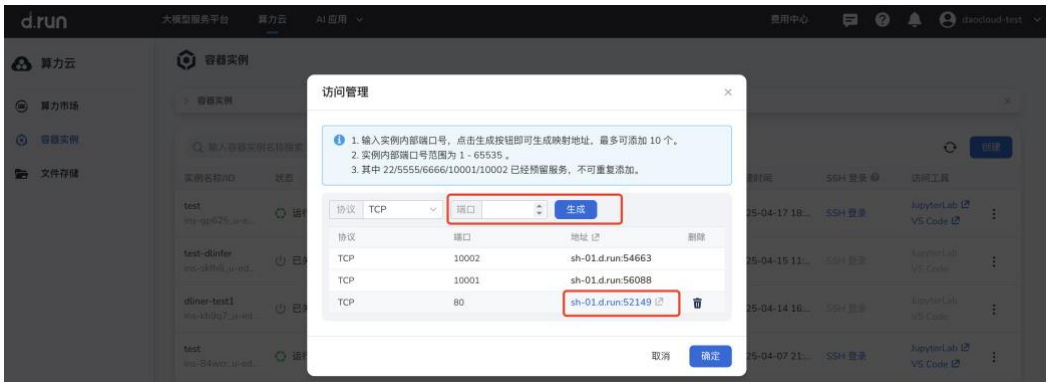

图 46　外部端口分配完成

6. 点击生成的外部访问链接，即可访问 nginx 服务。

图 47　通过外部链接访问 nginx 服务

（五）容器实例内服务开机自启

在每一个算力云的容器实例中，都有一个 s6 守护进程，它会在开机时，将指定的服务自动拉起。如果用户需要开启自启服务，只需要向 s6 中注册自己的自定义服务。

前提条件：

- 已登录 d.run 账号
- 已通过算力云创建容器实例，且容器实例状态为**运行中**

注册自定义服务

在 /etc/s6/ 目录下创建一个目录 <Your_service_name>，然后在这个目录下创建一个固定名称 run 的 Bash 脚本，然后在 run 文件中填写服务启动命令。

> 注册自定义镜像后，需要关机保存镜像，这样才能将配置进行持久化。下次开机后，注册的自定义服务就会自动启动了。

示例一：以 Nginx 举例。

```
mkdir /etc/s6/nginx    # 注册自定义服务
cat <<EOF > /etc/s6/nginx/run  # 注册自定义服务的启动脚本
#!/bin/bash

echo "Starting Nginx..."

exec nginx        # 启动 nginx
EOF
```

示例二：以 Python 的 http 程序举例。

```
mkdir /etc/s6/python_http    # 注册自定义服务
cat <<EOF > /etc/s6/python_http/run  # 注册自定义服务的启动脚本
#!/bin/bash

echo "Starting python http..."

exec python /root/data/http.py
EOF
```

六、容器实例 Docker 功能

容器实例的 Docker 功能为开发者提供了在算力云环境中进行容器化开发的强大能力。通过启用 Docker 功能，用户可以在容器实例内部创建、管理和运行 Docker 容器，实现更灵活的开发环境配置和应用部署。

Docker 功能支持完整的容器生命周期管理，包括镜像拉取、容器创建、启动停止等基本操作，同时提供存储挂载、GPU 加速、网络配置等高级功能。开发者可以利用这些功能构建复杂的容器化应用，进行 AI 模型训练、算法开发和服务部署等工作。

此外，Docker 功能还支持镜像制作和管理，用户可以通过 docker build、save 等方式创建和保存自定义镜像，并支持 Docker buildx 和 Compose 等高级工具，满足复杂场景下的容器化开发需求。

（一）启用 Docker 功能

在创建容器实例时启用 Docker 功能：

1. 登录 d.run 平台，进入**算力云** → **容器实例**，点击**创建**按钮。

2. 在创建页面中找到**高级配置**选项，展开高级配置面板。

3. 在高级配置中勾选**启用 Docker** 选项。

4. 完成其他配置后，点击**确定**创建实例。

> 启用 Docker 功能的容器实例创建后，Docker 服务会自动启动，无需手动配置。

（二）容器的基本操作

Docker 功能提供完整的容器生命周期管理能力，支持容器的创建、启动、停止、删除等基本操作。

1. 创建和运行容器

使用 docker run 命令创建并启动容器：

```
# 基本语法
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

# 运行一个简单的 Ubuntu 容器
docker run -it ubuntu:20.04 /bin/bash

# 后台运行容器
docker run -d --name my-app nginx:latest

# 指定端口映射
docker run -d -p 8080:80 --name web-server nginx:latest
```

参数	说明
-d	后台运行容器
-it	交互式运行，分配伪终端
--name	指定容器名称
-p	端口映射，格式为 宿主机端口:容器端口
-v	挂载数据卷

2. 管理容器

查看容器状态：

```
# 查看正在运行的容器
docker ps

# 查看所有容器（包括已停止的）
docker ps -a

# 查看容器详细信息
docker inspect container_name
```

启动和停止容器：

```
# 启动已停止的容器
docker start container_name

# 停止运行中的容器
docker stop container_name

# 重启容器
docker restart container_name
```

3. 进入容器

进入正在运行的容器：

```
# 进入容器的交互式终端
docker exec -it container_name /bin/bash

# 执行单个命令
docker exec container_name ls -la /app
```

> 建议使用 docker exec 而不是 docker attach 来进入容器，因为 exec 会创建新的进程，退出时不会影响容器的运行状态。

（三）挂载存储

容器实例支持将文件存储和数据盘挂载到 Docker 容器中，实现数据的持久化和共享。

1. 挂载文件存储

文件存储默认挂载在容器实例的 /root/data 目录下，可以将此目录挂载到 Docker 容器中：

```
# 挂载文件存储到容器
docker run -d -v /root/data:/data --name my-app ubuntu:20.04

# 挂载到指定目录
docker run -d -v /root/data:/workspace/data --name dev-env python:3.9

# 只读挂载
docker run -d -v /root/data:/data:ro --name readonly-app ubuntu:20.04
```

文件存储特点：

- 支持多个容器实例间共享数据。
- 数据持久化，容器删除后数据仍然保留。
- 免费提供 20 GB 空间，支持弹性扩容。

2. 挂载数据盘

数据盘与容器实例生命周期绑定，提供高性能的本地存储：

```
# 挂载数据盘（假设数据盘挂载在 /mnt/disk）
docker run -d -v /mnt/disk:/app/data --name high-perf-app ubuntu:20.04

# 同时挂载文件存储和数据盘
docker run -d \
  -v /root/data:/shared \
```

```
  -v /mnt/disk:/local \
  --name multi-storage ubuntu:20.04
```

数据盘特点：

- 免费提供 50 GB 空间，支持扩容至 200 GB。
- 高性能本地存储，适合临时数据和缓存。
- 与容器实例生命周期绑定，实例删除时数据会丢失。
- 不支持实例间数据共享。

（四）使用 GPU

容器实例的 Docker 功能支持将 GPU 资源挂载到容器中，为 AI 训练和推理提供硬件加速能力。

1. 挂载 GPU

使用 --gpus 参数将 GPU 设备挂载到容器：

```
# 挂载所有 GPU
docker run --gpus all -it pytorch/pytorch:latest python

# 挂载指定数量的 GPU
docker run --gpus 2 -it tensorflow/tensorflow:latest-gpu python

# 挂载指定的 GPU
docker run --gpus device=0 -it nvidia/cuda:11.8-devel-ubuntu20.04
```

2. 支持的 GPU

d.run 算力云支持多种 GPU 供应商：

- **Nvidia GPU**：支持 CUDA 加速，兼容主流深度学习框架。
- **沐曦 GPU**：国产 GPU 解决方案，支持 AI 计算加速。

（五）网络配置

Docker 容器可以通过多种网络模式与宿主机和外部网络进行通信。将容器端口映射到宿主机端口，实现外部访问：

```
# 映射单个端口
docker run -d -p 8080:80 --name web-app nginx:latest

# 映射多个端口
docker run -d \
  -p 8080:80 \
  -p 8443:443 \
  --name web-server nginx:latest

# 映射到指定 IP
docker run -d -p 127.0.0.1:8080:80 --name local-app nginx:latest

# 映射随机端口
```

```
docker run -d -P --name random-port nginx:latest
```

（六）制作镜像

Docker 功能支持多种方式制作和保存自定义镜像，满足不同场景的需求。

1. docker build

使用 Dockerfile 构建镜像是最常用的方式：

```
# 基本构建命令
docker build -t my-app:latest .

# 指定 Dockerfile 路径
docker build -f /path/to/Dockerfile -t my-app:v1.0 .
```

2. docker save

将镜像导出为 tar 文件：

```
# 导出单个镜像
docker save -o my-image.tar my-app:latest
```

导出的镜像可以通过 docker load 命令导入：

```
# 导入镜像
docker load -i my-image.tar
```

（七）高级功能

容器实例的 Docker 功能支持 buildx 和 Compose 等高级工具，满足复杂场景下的容器化开发需求。

1. Docker buildx

Docker buildx 是 Docker 的扩展构建功能，支持多平台构建和高级构建特性。

基本用法

```
# 查看 buildx 版本
docker buildx version

# 查看可用的构建器
docker buildx ls

# 创建新的构建器
docker buildx create --name mybuilder --use

# 启动构建器
docker buildx inspect --bootstrap
```

多平台构建

```
# 构建多平台镜像
```

```
docker buildx build --platform linux/amd64,linux/arm64 -t my-app:latest .

# 构建并推送到仓库
docker buildx build --platform linux/amd64,linux/arm64 -t my-app:latest --push .

# 构建特定平台
docker buildx build --platform linux/amd64 -t my-app:amd64 .
```

2. Docker Compose

Docker Compose 用于定义和运行多容器应用程序。

安装和基本用法

```
# 检查 Compose 版本
docker compose version

# 启动服务
docker compose up -d

# 查看服务状态
docker compose ps

# 停止服务
docker compose down

# 查看日志
docker compose logs
```

（八）访问镜像仓库

容器实例的 Docker 功能支持访问算力云镜像仓库以及其他公有和私有镜像仓库。

1. 算力云镜像仓库

算力云提供内置的镜像仓库服务，用户可以存储和管理自定义镜像。

访问仓库

以下是查看仓库地址的一个示例。实际地址请参考平台提供的**我的镜像**中仓库信息。

```
REGISTRY_URL="harbor.d.run"

# 拉取镜像
docker pull ${REGISTRY_URL}/my-namespace/my-app:latest

# 推送镜像
docker push ${REGISTRY_URL}/my-namespace/my-app:latest
```

认证配置

> 当前版本暂不支持自动注入用户名和密码，需要手动配置认证信息。

手动配置认证：

```
# 登录到算力云镜像仓库
docker login registry.d.run -u your-username
```

```
# 输入密码
Password: your-password

# 验证登录状态
docker info | grep -A 5 "Registry Mirrors"
```

（九）故障排查

常见问题及解决方法：

1. 容器启动失败

```
# 查看容器日志
docker logs container_name

# 查看容器详细信息
docker inspect container_name
```

2. 端口访问问题

```
# 检查端口映射
docker port container_name

# 检查防火墙设置
netstat -tlnp | grep :8080
```

3. 存储挂载问题

```
# 检查挂载点
docker inspect container_name | grep -A 10 "Mounts"

# 验证宿主机路径权限
ls -la /root/data
```

4. GPU 不可用

```
# 检查 GPU 状态
nvidia-smi

# 验证容器内 GPU 访问
docker exec container_name nvidia-smi
```

重要提醒
容器实例关机时，运行中的 Docker 容器会被停止。
重启容器实例后，需要手动重启 Docker 容器。
删除容器实例会同时删除所有 Docker 容器和未持久化的数据。

最佳实践
使用 Docker Compose 管理复杂的多容器应用。
定期清理未使用的镜像和容器以节省存储空间。
为生产环境的容器配置健康检查和重启策略。
使用标准化的镜像命名和版本管理规范。

通过合理使用容器实例的 Docker 功能，开发者可以构建灵活、高效的容器化开发和部署环境，充分利用算力云平台的计算资源和存储能力。

七、文件存储与我的镜像

文件存储与我的镜像是容器实例最重要的两项配套能力：前者解决**数据与代码的共享持久化**问题，后者解决**运行环境的打包与复用**问题。本章介绍两者的初始化、使用与计费规则。

（一）初始化文件存储

算力云的文件存储是一种基于实例的共享存储服务，同一个用户创建的所有容器实例都可以访问并共享同一套文件数据。文件存储主要用来存储数据、源代码等文件，可将重要文件定期上传到文件存储中。

> d.run 为每个用户提供免费 20 GB 空间，同时支持弹性扩缩容，扩大的容量按天计费，不足一天则按实际使用时间扣费，精确到秒。

初始化文件存储是指在创建文件存储资源后，为其进行首次配置和准备，以便容器实例或应用能够正常访问和使用。该过程通常包括分配存储路径、设置访问权限、挂载至指定的容器实例或节点，以及根据业务需求进行目录结构或初始数据的导入。

初始化完成后，文件存储即可作为共享数据源，供同一用户空间下的多个容器实例同时访问，确保数据一致性和可见性。这一操作通常在新建项目、部署应用或替换存储资源时进行，是保障后续业务稳定运行的前提步骤。

前提条件：

- 已登录您的 d.run 账号
- 账户余额 ≥ 0 元

操作步骤：

1. 登录 d.run，进入**算力云 → 文件存储**，选择相近的区域，点击**初始化文件存储**开始初始化。

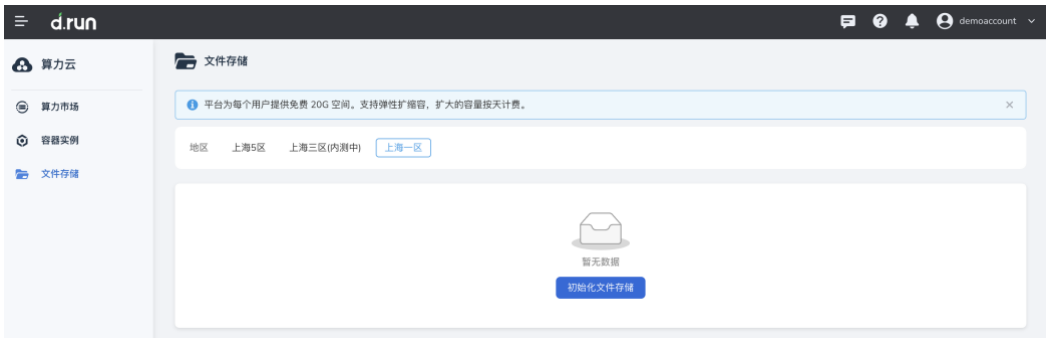

图 48　初始化文件存储

图 49　文件存储初始化完成

2. 上传文件：点击**上传文件**选择本地文件进行上传。

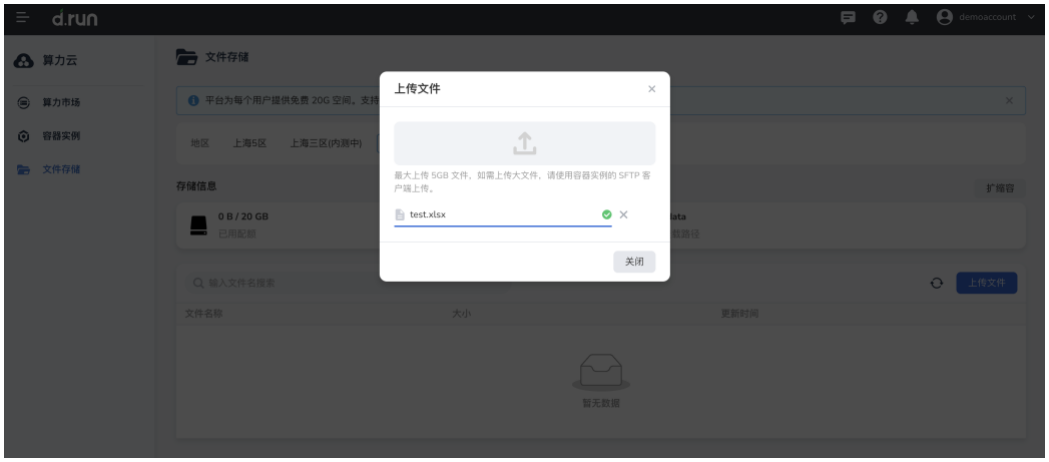

图 50　上传文件到文件存储

3. 执行更多操作。

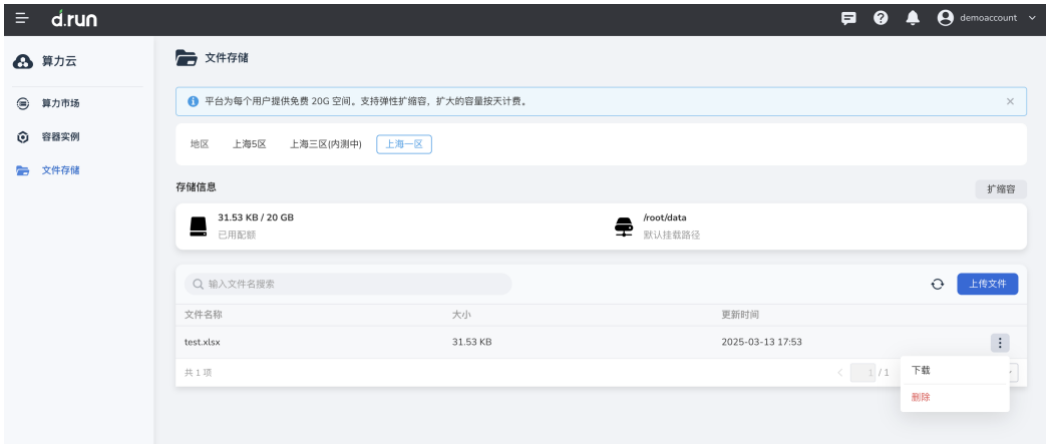

图 51　下载与删除文件

- **下载文件**：选择需要下载的文件，点击列表右侧的 ⋮，在下拉列表中选择**下载**。
- **删除文件**：选择需要删除的文件，点击列表右侧的 ⋮，在下拉列表中选择**删除**。删除后不可恢复，请谨慎操作！

（二）文件存储扩缩容

在算力云或容器化应用环境中，文件存储扩缩容指的是根据业务需求动态调整共享文件存储的容量大小。通过扩容，用户可以在数据增长时快速增加存储空间，避免因容量不足导致业务中断；通过缩容，则可以在数据清理或业务规模下降后，释放多余的存储资源，降低成本。

文件存储扩缩容功能支持不中断服务在线调整，在调整过程中，挂载该存储的所有容器实例依然可以正常读写数据。这一能力适用于数据量波动较大、或需要长期稳定访问共享文件的场景，例如日志归档、模型训练数据集、媒体文件处理等。

以下演示如何在 d.run 中对文件存储执行扩缩容操作。

前提条件：

- 所选地区已完成文件存储初始化。
- 账户余额大于扩容所需费用。

操作步骤：

1. 登录 d.run，进入**算力云 → 文件存储**。
2. 在**文件存储**页面，点击右上角的**扩缩容**。

图 52　文件存储页面的扩缩容入口

3. 在弹出的**扩缩容**窗口中，输入**存储配额**，点击**确定**。

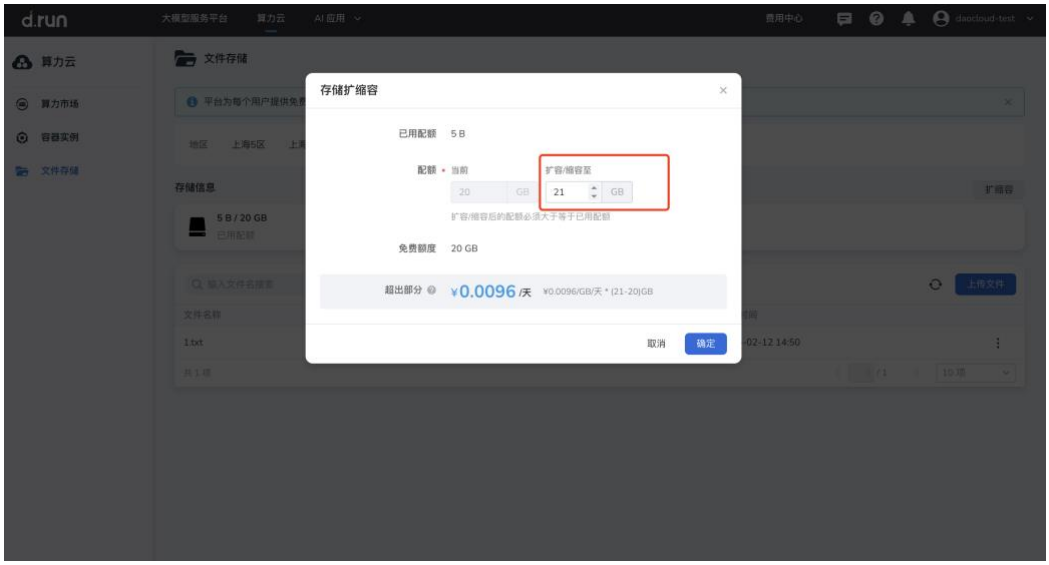

图 53　输入新的存储配额

（三）文件存储计费规则

项目	规则
计费规则	超出免费容量 20 GB 的部分按天计费，不足一天则按实际使用时间扣费，精确到秒。
计费时间	扩容开始计费，直到下一次扩容/缩容结束该订单。若缩容到 20 GB 则不再计费，若扩容/缩容到其他规格则开启新订单继续计费。
欠费处理	若账户余额 < 0，且欠费前已扩容至付费容量（大于 20 GB），则系统将继续保存 15 天，期间继续扣费。欠费后支持手动缩容，若手动缩容至 20 GB，则不再继续扣费。若欠费超过 15 天则自动缩容至 20 GB，由于缩容造成的文件丢失无法找回，请及时充值。
长时间无消费记录	若 30 天内未在平台产生消费记录（包括但不限于在算力云、大模型服务等任何模块产生的费用），则平台会自动删除文件存储，下次使用时需要重新初始化。

（四）我的镜像

我的镜像是基于 Harbor 构建的**专属镜像仓库服务**，用于集中存储和管理用户的自定义镜像。它支持将容器实例的系统盘内容打包为镜像并保存至仓库，不仅能有效防止因实例重建、意外删除或硬件故障导致的数据丢失，还能在需要时**一键复用**，快速创建相同环境的容器实例。

通过"我的镜像"，用户可以：

- 持久化保存容器运行环境及配置，便于跨项目或跨环境复用。
- 快速分发和部署镜像，提高开发与运维效率。
- 配合版本管理，追踪环境变更历史，便于回滚与调试。

> d.run 为每位用户默认提供 **50 GB 免费镜像空间**，如需更大配额，请联系 **400-002-6898** 申请扩容。

前提条件：已登录您的 d.run 账号。

1. 初始化镜像仓库

1. 登录 d.run，进入**算力云 → 我的镜像**，选择区域后点击**初始化镜像仓库**。

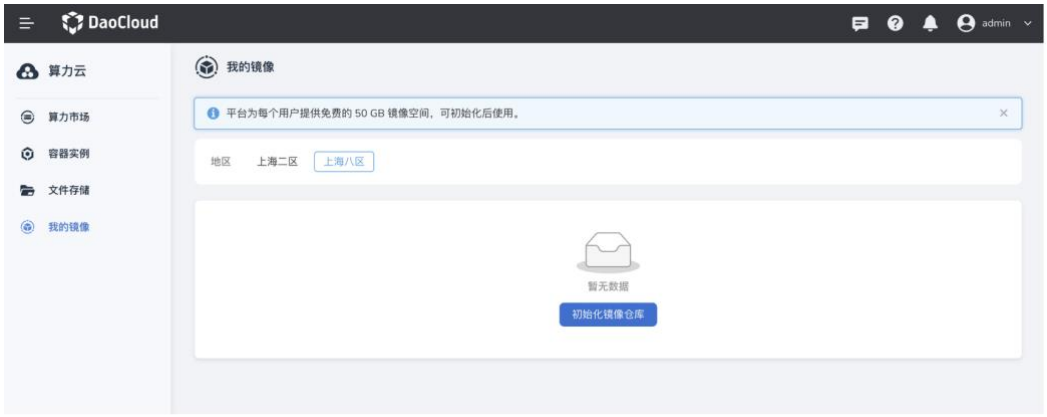

图 54　初始化镜像仓库

2. 设置镜像仓库的用户名和密码，然后点击**开始初始化**初始化镜像仓库。

图 55　设置镜像仓库的用户名和密码

3. 使用镜像仓库地址即可访问 Harbor 镜像仓库。

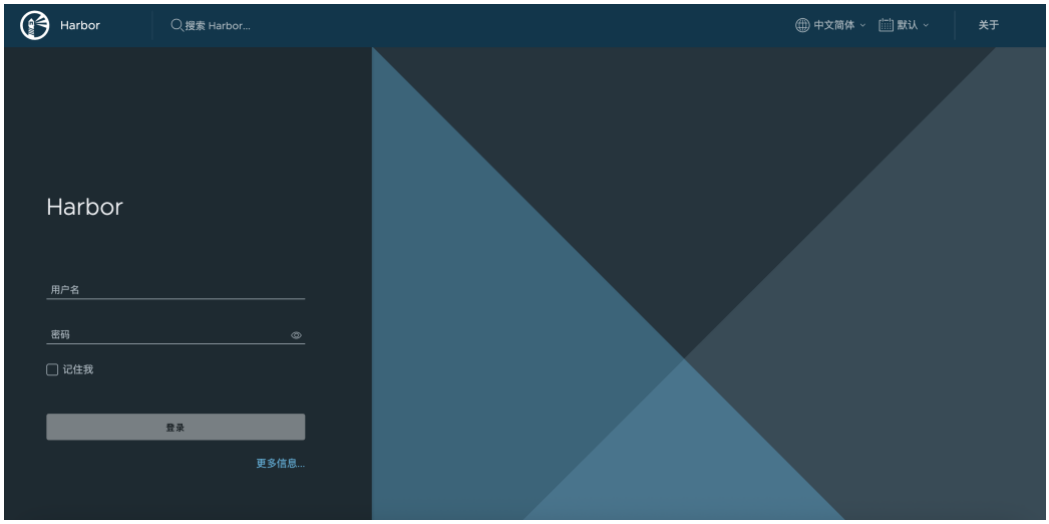

图 56　获取 Harbor 镜像仓库地址

2. 使用镜像

点击**算力市场 → 立即购买**，或**容器实例 → 创建**，进入容器实例的创建页面，选择镜像为**我的镜像**即可使用镜像仓库中已保存的镜像创建容器实例。

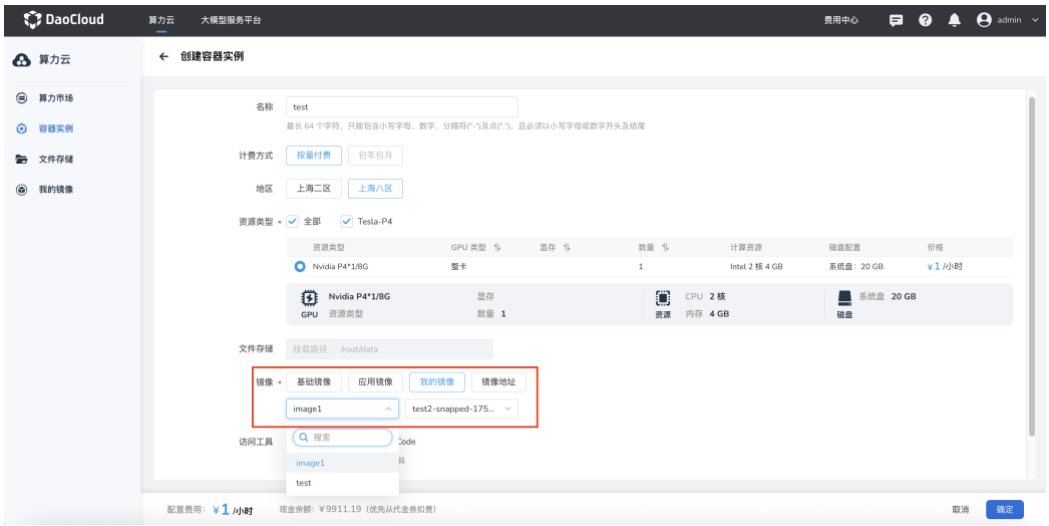

图 57　选择「我的镜像」创建容器实例

3. 删除镜像/镜像版本

1). 在**算力云 → 我的镜像**，找到需要删除的镜像，在右侧操作栏选择**删除**按钮。

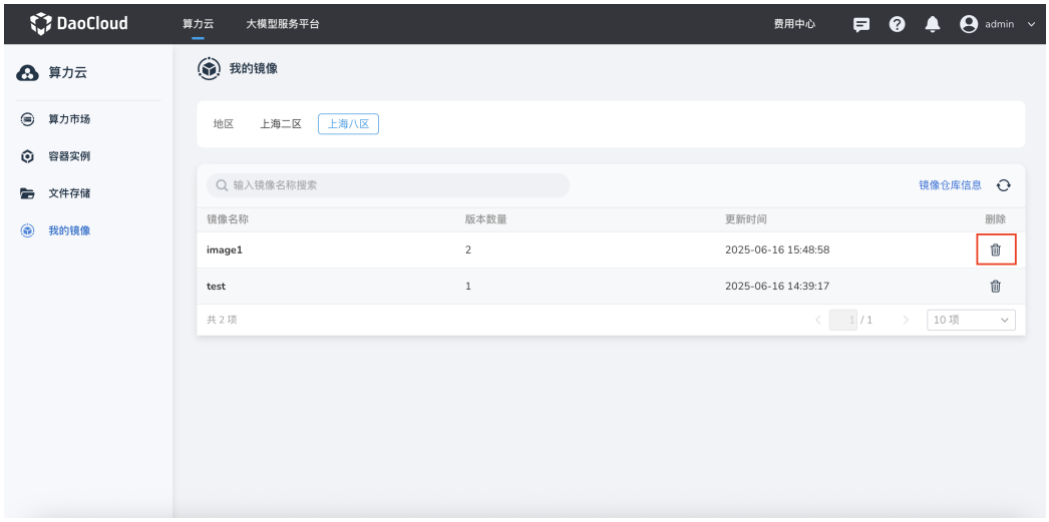

图 58　删除镜像

2). 在二次确认弹窗中点击**确认**，删除镜像的所有版本。

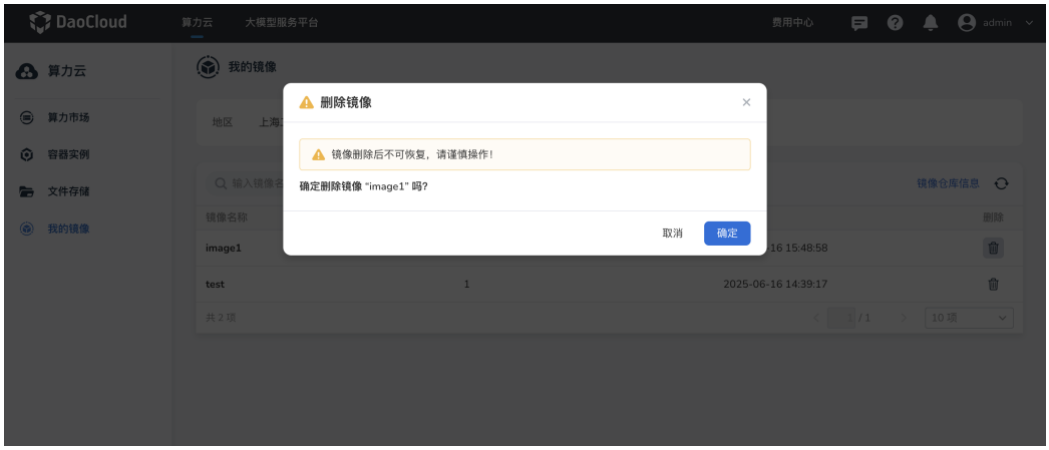

图 59　二次确认删除

3). 在**算力云 → 我的镜像**，点击镜像详情页，找到需要删除的镜像版本，在右侧操作栏选择**删除**按钮。

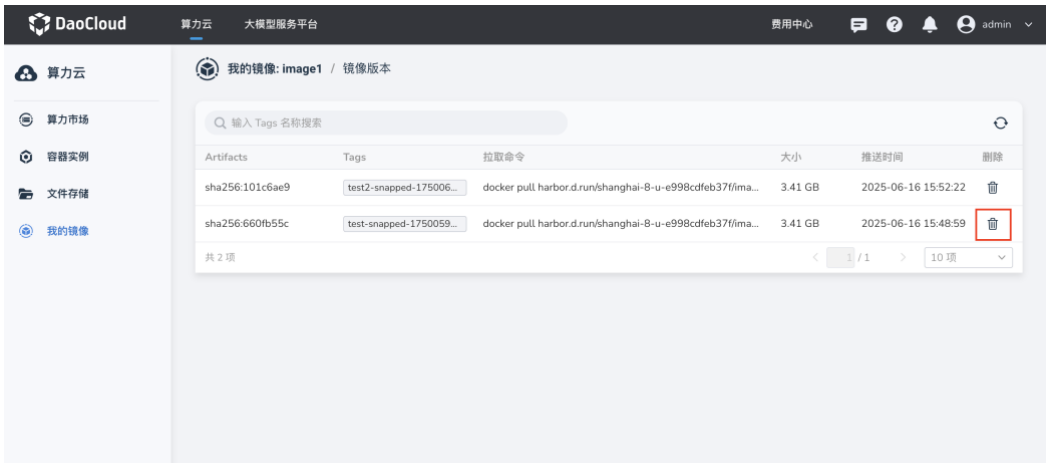

图 60　删除指定镜像版本

镜像/镜像版本删除后不可恢复，请谨慎操作！

（五）保存镜像

本节介绍在 d.run 中如何保存容器实例系统盘至**我的镜像**，主要有手动保存、关机保存、定时关机自动保存及欠费自动保存共四种方式。

前提条件：

- 所选地区已完成镜像仓库初始化
- 镜像空间配额充足

1. 手动保存镜像

进入**算力云 → 容器实例**列表，选择需要保存镜像的容器实例，点击列表右侧的 ⋮，在下拉列表中选择**保存镜像**，填写镜像名称和 Tags 后点击**确定**。

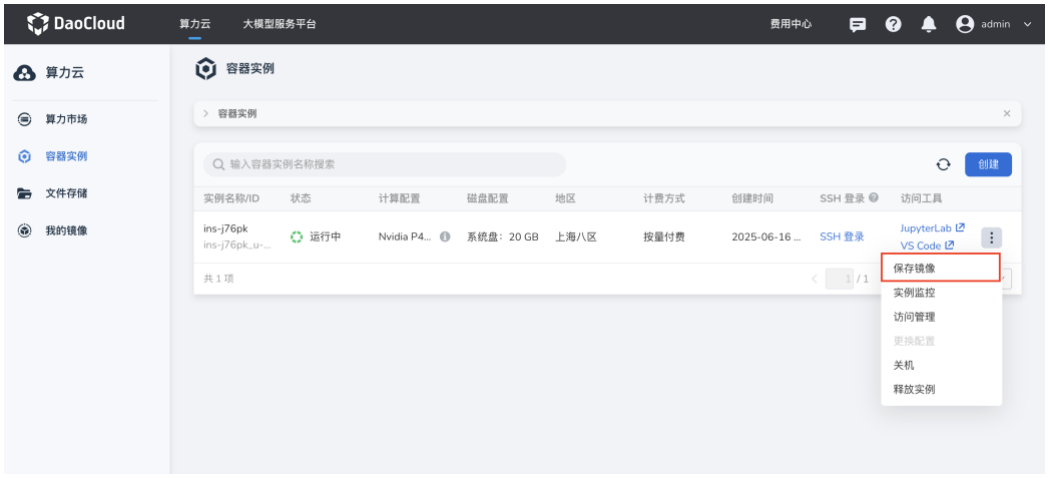

图 61　手动保存镜像入口

图 62　填写镜像名称与 Tags

- **已有镜像**：选择已有镜像时，镜像将以新版本的形式存放在所选镜像的详情中。
- **新建镜像**：选择新建镜像时，镜像将以新镜像的形式存放在我的镜像列表中。

2. 关机保存镜像

对容器实例执行关机操作时，选择**关机保存系统盘**保存镜像。

图 63　关机时保存系统盘

3. 自动保存镜像

当容器实例**定时关机**或因欠费关机时，平台将为用户自动保存系统盘至**我的镜像**。

> 镜像空间配额不足时将会导致系统盘保存失败，数据丢失，请确保配额充足！

八、实战：五步部署 DeepSeek

本章介绍如何在算力云模块通过容器实例直接使用 DeepSeek，以及通过 Jupyter、VS Code、SSH 进行算法二次开发使用。

> 算力云上线的 DeepSeek-ai/Janus-Pro-7B-ComfyUI WebUI 版文生图模型，支持 Multimodal Understanding 和 Text-to-Image Generation 两种服务，部署后可通过 **10002 端口**直接访问并使用。

前提条件：

- 已注册并登录 d.run
- 账户余额大于等于所选资源类型的单价

（一）创建 DeepSeek 实例

1. 登录 d.run 平台，选择**算力云**进入**算力市场**。在算力市场页面中选择算力规格，点击**立即购买**，进入创建容器实例页面。

运行 Janus-Pro-7B 模型需约 20 GB 显存，请确保所选算力规格满足此要求。

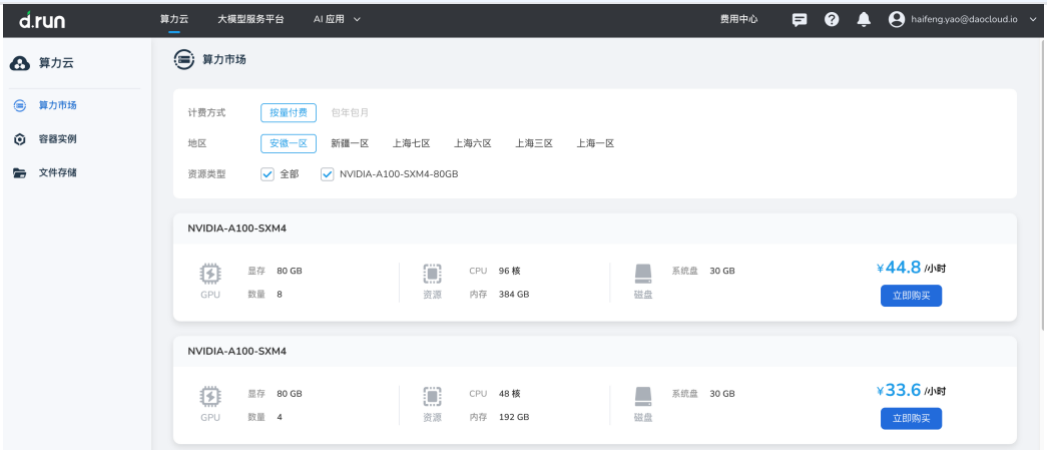

图 64　在算力市场选择算力规格

2. 填写实例名称，选择 DeepSeek 的镜像为 DeepSeek-ai/Janus-Pro-7B-ComfyUI。（初次使用建议初始化文件存储并挂载），点击**确定**后完成部署。

图 65　选择 DeepSeek 镜像并完成部署

3. 刷新页面当容器实例状态为运行中时，点击**访问管理**，打开 **10002 端口**即可访问 Janus-Pro-7B 的 UI 界面。

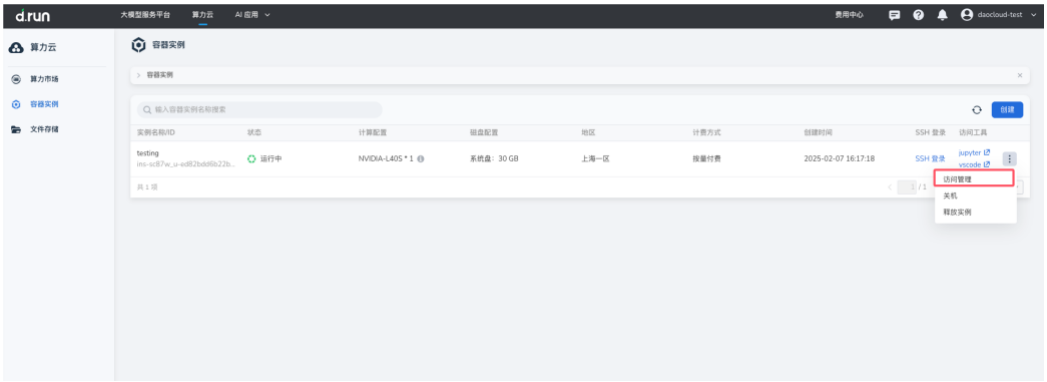

图 66　在访问管理中开放 10002 端口

4. 使用 DeepSeek Janus-Pro 提供的多模态理解和文生图服务。

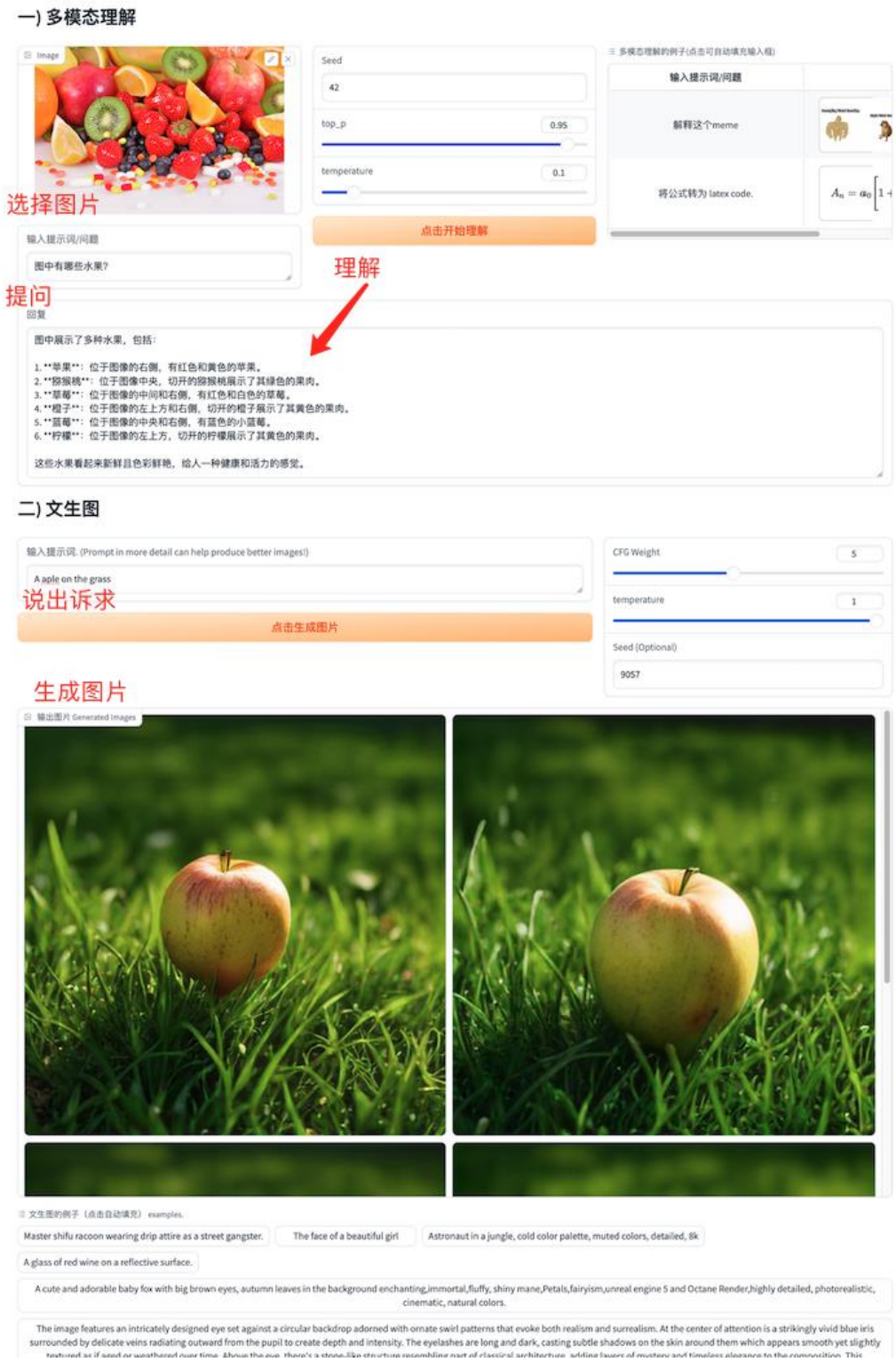

图 67　使用文生图服务

（二）通过 Jupyter、VS Code 进行算法二次开发

通过此方法部署的 DeepSeek Janus-Pro 7B，其模型文件通常存放在容器的 /JANUS/deepseek-ai/Janus-Pro-7B/ 目录下，如有需要可通过 Jupyter 或 VS Code 进行二次开

发。通过 Jupyter、VS Code 进行算法二次开发使用，可在页面上直接打开。SSH 登录提供了用户名密码和 SSH 公钥免密登录两种方式。

SSH 公钥免密登录可前往个人中心导入公钥，这样重启后或新创建的实例都能免密码登录。

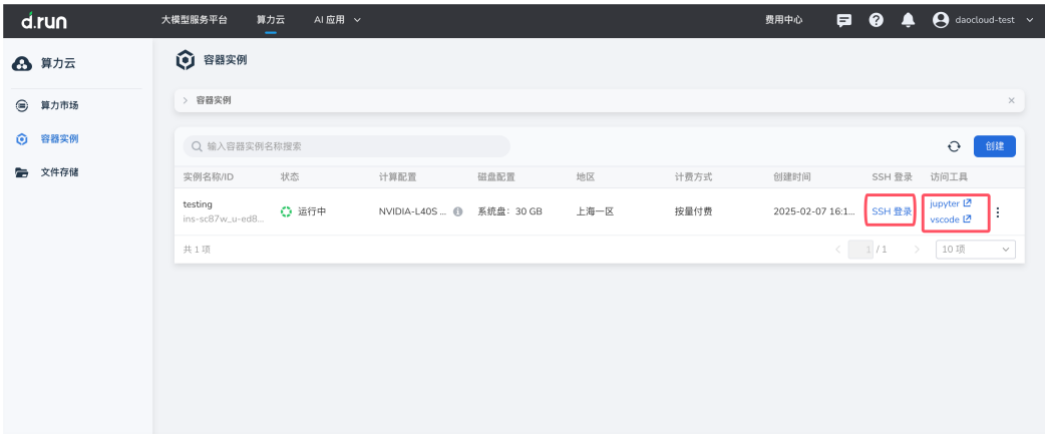

图 68　实例的多种访问方式

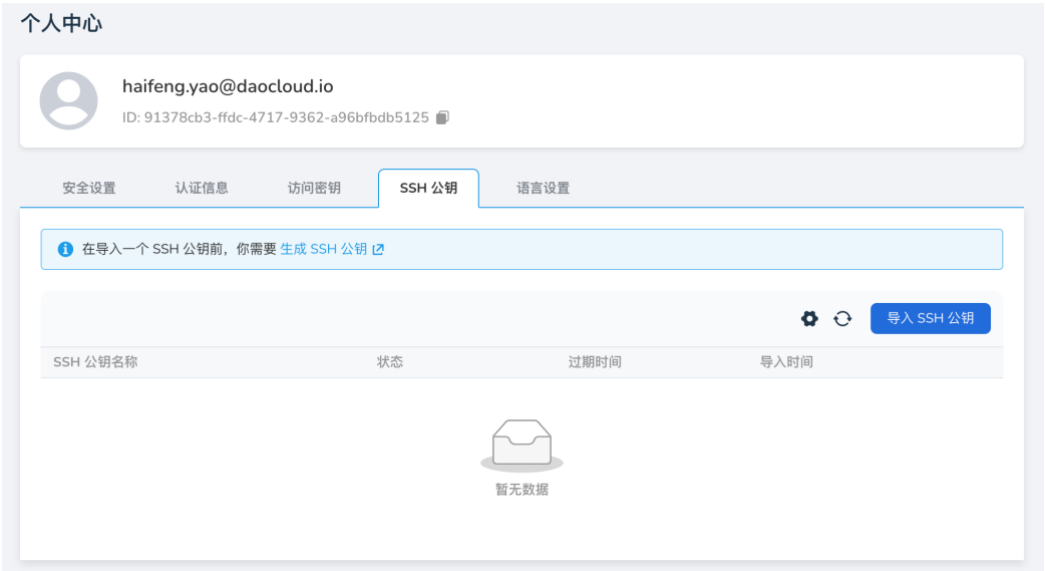

图 69　通过 SSH 访问实例

九、常见问题

本章列出了一些在算力云（Zestu）中可能遇到的问题，为您提供便利的故障排除解决办法。

（一）显存规格不一致的问题

现象：为什么在容器实例中，命令行查询的显存规格和页面显示的显存规格不一致？

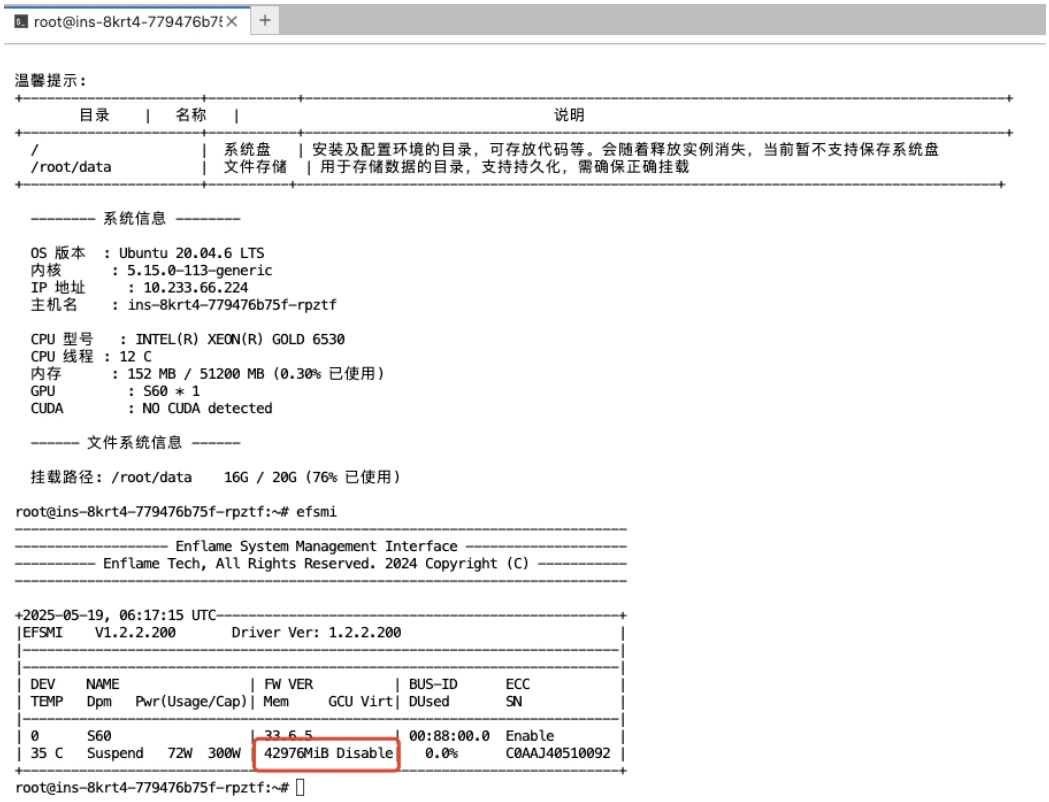

图 70　命令行查询到的显存规格

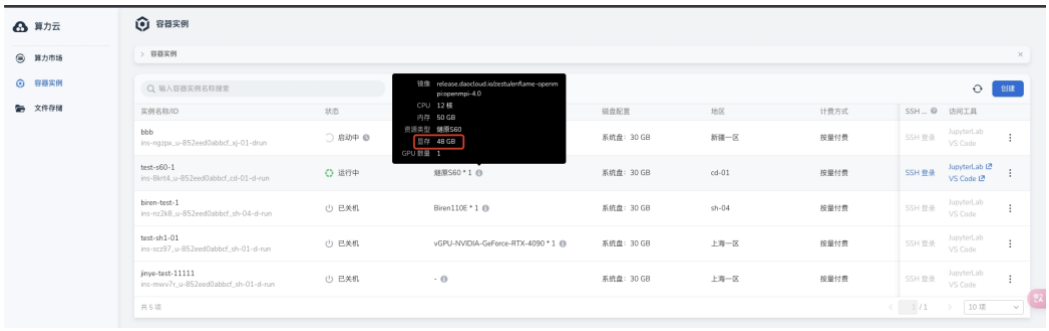

图 71　页面显示的显存规格

原因：这是由于 GPU 卡开启了 ECC（Error Correcting Code，错误检查和纠正）功能，该功能可以提高数据的正确性，随之而来的是可用内存的减少和性能上的损失，故 ECC 功能导致消耗了一部分实际显存。ECC 默认在企业级显卡中开启，在消费级显卡中关闭。