

MindX DL
5.0.0.2

Ascend Docker Runtime 用户指南

文档版本	01
发布日期	2024-11-25



版权所有 © 华为技术有限公司 2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址：<https://www.huawei.com>

客户服务邮箱：support@huawei.com

客户服务电话：4008302118

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：

<https://www.huawei.com/cn/psirt/vul-response-process>

如企业客户须获取漏洞信息，请参见如下网址：

<https://securitybulletin.huawei.com/enterprise/cn/security-advisory>

目 录

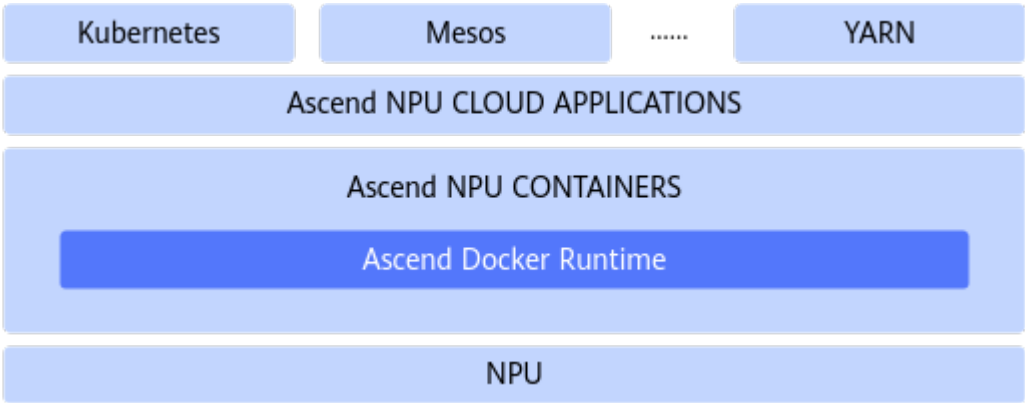
1 简介	1
2 支持的产品形态	2
3 安装部署	4
3.1 手动安装	4
3.1.1 环境准备	4
3.1.2 获取软件包	4
3.1.3 安装 Ascend Docker Runtime	6
3.1.4 确认组件状态	9
3.2 使用工具安装	10
4 使用指导	11
4.1 场景介绍	11
4.2 （可选）配置自定义挂载内容	11
4.3 在 Docker 客户端使用	12
4.4 在 K8s 集成 Docker 使用	18
4.5 在 Containerd 客户端使用	18
4.6 在 K8s 集成 Containerd 使用	22
5 升级	24
6 卸载	26
7 FAQ	28
7.1 系统日志出现 ascend-docker-r[xxx]:syscall 439 相关打印	28
8 参考信息	29
8.1 边缘容器日志输出指导	29
8.2 默认挂载内容	31
8.3 命令说明	35
A 附录	37
A.1 用户信息列表	37
A.2 修订记录	38

1 简介

产品定义

Ascend Docker Runtime（又称Ascend Docker，又称昇腾容器运行时）是MindX DL的基础组件，用于为所有的训练或推理作业提供昇腾AI处理器（Ascend NPU）容器化支持，使用户AI作业能够以Docker容器的方式平滑运行在昇腾设备之上，Ascend Docker Runtime逻辑接口如图1-1所示。

图 1-1 Ascend Docker



产品价值

- 充分解耦：与Docker解耦，无需修改Docker代码，Ascend Docker Runtime可以独立运行。
- 后向兼容：提供可选装的Ascend Docker Runtime，不影响原生Docker使用方式。
- 易适配：与用户现有平台和系统平滑适配，不影响原Docker的命令接口。
- 易部署：提供run包部署，用户安装后即可用Docker创建挂载NPU的容器。

2 支持的产品形态

表 2-1 产品说明

产品系列	产品类型	产品名称
Atlas 训练系列产品	标卡	Atlas 300T 训练卡（型号 9000）
		Atlas 300T Pro 训练卡（型号 9000）
	训练服务器	Atlas 800 训练服务器（型号 9000）
		Atlas 800 训练服务器（型号 9010）
		Atlas 900 PoD（型号 9000）
Atlas A2 训练系列产品	训练服务器	Atlas 800T A2 训练服务器
		Atlas 900 A2 PoD 集群基础单元
Atlas 200/300/500 推理产品	边缘节点	Atlas 200 AI加速模块（RC场景）
	标卡	Atlas 300I 推理卡（型号 3000）
		Atlas 300I 推理卡（型号 3010）
Atlas 推理服务器系列产品	推理服务器	Atlas 800 推理服务器（型号 3000）支持插入Atlas 300I 推理卡（型号 3000）、Atlas 300I 推理卡（型号 3010）、Atlas 300I Pro 推理卡、Atlas 300V 视频解析卡、Atlas 300V Pro 视频解析卡、Atlas 300I Duo 推理卡使用
		Atlas 800 推理服务器（型号 3010）支持插入Atlas 300I 推理卡（型号 3000）、Atlas 300I Pro 推理卡、Atlas 300V 视频解析卡、Atlas 300V Pro 视频解析卡、Atlas 300I Duo 推理卡使用

产品系列	产品类型	产品名称
Atlas 200I/500 A2 推理产品	边缘节点	Atlas 200I A2 加速模块
		Atlas 200I DK A2 开发者套件
		Atlas 500 A2 智能小站
Atlas 推理系列产品	推理服务器	Atlas 200I SoC A1 核心板
	标卡	Atlas 300I Pro 推理卡
		Atlas 300V Pro 视频解析卡
		Atlas 300I Duo 推理卡
		Atlas 300V 视频解析卡
其他	边缘节点	Atlas 500 Pro 智能边缘服务器（型号 3000）支持插入Atlas 300I 推理卡（型号 3000）、Atlas 300I Pro 推理卡、Atlas 300V 视频解析卡、Atlas 300V Pro 视频解析卡使用

支持的操作系统

表 2-2 支持的操作系统

操作系统名称	操作系统版本	说明
Ubuntu	<ul style="list-style-type: none">Ubuntu 18.04.1Ubuntu 18.04.5Ubuntu 20.04Ubuntu 22.04	Ubuntu 22.04仅支持Atlas A2 训练系列产品
CentOS	CentOS 7.6	-
openEuler	<ul style="list-style-type: none">openEuler 20.03 LTSopenEuler 22.03 LTS	-
KylinOS	<ul style="list-style-type: none">KylinOS V10SP1KylinOS V10SP1	-
Debian	Debian 10.0	仅Atlas 200T A2 Box16 异构子框支持
UOS	UOS V20 （ 1020e ）	-

3 安装部署

- 3.1 手动安装
- 3.2 使用工具安装

3.1 手动安装

3.1.1 环境准备

- 当前环境的Docker版本需要为18.03及以上版本。
- 宿主机已安装驱动和固件，详情请参见《[CANN 软件安装指南](#)》的“安装驱动和固件”章节。
- Atlas 500 A2 智能小站安装Ascend Docker Runtime需要修改Docker配置。执行 **vi /etc/sysconfig/docker**命令，将`--config-file=""`参数删除；并执行**systemctl restart docker**使配置生效。
- Atlas 500 A2 智能小站预置的MEF服务会对Docker进行安全加固配置，Ascend Docker Runtime不支持在安全加固后的Docker环境下使用。若需要使用Ascend Docker Runtime，请手动卸载MEF服务，参考《[MindX Edge MEF用户指南](#)》中的“卸载 > 卸载MEF Edge”章节进行操作。

说明

执行**systemctl status docker**命令，如果返回信息里包含“/docker_entrypoint.sh”字段，则为MEF服务安全加固后的docker。

3.1.2 获取软件包

下载软件

Ascend Docker Runtime为开源软件，不需要申请下载权限就可以直接下载，但不能用于商业用途。请参考[表3-1](#)获取所需软件包。下载的开源软件包可以通过软件SUM值验证。

下载本软件即表示您同意[华为企业业务最终用户许可协议（EULA）](#)的条款和条件。

表 3-1 Ascend Docker Runtime 软件包说明

软件包名称	包内文件	说明	获取链接
Ascend-docker-runtime_{version}_linux-{arch}.run	ascend-docker-cli	Ascend Docker Runtime运行所必须的可执行程序，不建议用户直接运行。	获取链接
	ascend-docker-destroy		
	ascend-docker-hook		
	ascend-docker-plugin-install-helper		
	ascend-docker-runtime		
	assets	说明资料的图片资源。	
	base.list*	默认的挂载列表，安装时，程序会根据install-type，安装不同的挂载列表。	
	run_main.sh	安装脚本，不建议用户直接使用。	
	uninstall.sh	卸载脚本。	
	README.md	Ascend Docker Runtime说明资料，包含设计原理。	

软件包完整性校验

在安装软件包时执行./Ascend-docker-runtime_{version}_linux-{arch}.run --check命令，校验软件包安装文件的一致性和完整性。

获取开源软件

集群调度提供Ascend Docker Runtime组件的开源代码。如果用户需要了解源码或定制开发组件，可以通过[表3-2](#)获取。

表 3-2 获取组件源码

组件名	源码地址
Ascend Docker Runtime	https://gitee.com/ascend/ascend-docker-runtime/tree/v5.0.0-Patch2/

3.1.3 安装 Ascend Docker Runtime

安装场景

目前只支持root用户安装、升级Ascend Docker Runtime，请根据实际情况选择对应的安装方式。

用户需要根据环境的容器引擎的安装和使用情况决定Ascend Docker Runtime的安装场景；常见的Ascend Docker Runtime安装场景如下。

- [Docker场景](#)
- [Containerd场景](#)

Docker 场景下安装 Ascend Docker Runtime

步骤1 安装包下载完成后，首先进入安装包（run包）所在路径。

```
cd <path to run package>
```

步骤2 执行如下命令，校验软件包安装文件的一致性和完整性。

```
./Ascend-docker-runtime_{version}_linux-{arch}.run --check
```

回显示例如下：

```
[WARNING]: --check is meaningless for Ascend-docker-runtime and will be discarded in the future
Verifying archive integrity... ./Ascend-docker-runtime_5.0.0.2_linux-x86_64.run does not contain an
embedded SHA256 checksum.
...
All good.
```

步骤3 执行以下命令，为软件包添加可执行权限。

```
chmod u+x Ascend-docker-runtime_{version}_linux-{arch}.run
```

步骤4 可通过以下命令安装Ascend Docker Runtime。

- 安装到默认路径下，执行以下命令。

```
./Ascend-docker-runtime_{version}_linux-{arch}.run --install
```
- 安装到指定路径下，执行以下命令，“--install-path”参数为指定的安装路径。

```
./Ascend-docker-runtime_{version}_linux-{arch}.run --install --install-path=<path>
```

回显示例如下，表示安装成功。

```
Uncompressing ascend-docker-runtime 100%
[INFO]: installing ascend docker runtime
...
[INFO] Ascend Docker Runtime install success
```

步骤5 执行以下命令，使Ascend Docker Runtime生效。

```
systemctl daemon-reload && systemctl restart docker
```

如果用户在Docker的daemon.json文件中配置了“{“live-restore”: true }”选项，且此时已部署了Ascend Device Plugin，则Ascend Docker Runtime安装之后重启容器时，不能成功重启Ascend Device Plugin组件。请在安装Ascend Docker Runtime后，重新安装部署Ascend Device Plugin组件。

----结束

Containerd 场景下安装 Ascend Docker Runtime

步骤1 安装包下载完成后，首先进入安装包（run包）所在路径。

```
cd <path to run package>
```

步骤2 执行如下命令，校验软件包安装文件的一致性和完整性。

```
./Ascend-docker-runtime_{version}_linux-{arch}.run --check
```

回显示例如下：

```
[WARNING]: --check is meaningless for Ascend-docker-runtime and will be discarded in the future
Verifying archive integrity... ./Ascend-docker-runtime_5.0.0.2_linux-x86_64.run does not contain an
embedded SHA256 checksum.
...
All good.
```

步骤3 执行以下命令，为软件包添加可执行权限。

```
chmod u+x Ascend-docker-runtime_{version}_linux-{arch}.run
```

步骤4 可通过以下命令安装Ascend Docker Runtime。

- 安装到默认路径下。
./Ascend-docker-runtime_{version}_linux-{arch}.run --install
- 安装到指定路径下，执行以下命令，“--install-path”参数为指定的安装路径。
./Ascend-docker-runtime_{version}_linux-{arch}.run --install --install-path=<path>

回显示例如下，表示安装成功。

```
Uncompressing ascend-docker-runtime 100%
[INFO]: installing ascend docker runtime
...
[INFO] Ascend Docker Runtime install success
```

步骤5 执行以下命令，修改Containerd配置文件。

- Containerd无默认配置文件时，依次执行以下命令，创建并修改配置文件。
mkdir /etc/containerd
containerd config default > /etc/containerd/config.toml
vim /etc/containerd/config.toml
- Containerd已有配置文件时，打开并修改配置文件。
vim /etc/containerd/config.toml

步骤6 根据实际情况修改config.toml的“runtime”为安装路径，示例如下加粗部分所示。

```
...
[plugins."io.containerd.monitor.v1.cgroups"]
  no_prometheus = false
[plugins."io.containerd.runtime.v1.linux"]
  shim = "containerd-shim"
  runtime = "/usr/local/Ascend/Ascend-Docker-Runtime/ascend-docker-runtime"
  runtime_root = ""
  no_shim = false
  shim_debug = false
[plugins."io.containerd.runtime.v2.task"]
  platforms = ["linux/amd64"]
...
```

步骤7 执行以下命令，重启Containerd。

```
systemctl daemon-reload && systemctl restart containerd
```

----结束

说明

Ascend Docker Runtime日志路径为“/var/log/ascend-docker-runtime/”。

Ascend Docker Runtime 安装包命令行参数说明

参数说明如表3-3所示。

表 3-3 安装包支持的参数说明

参数	说明
--help -h	查询帮助信息。
--info	查询软件包构建信息。在未来某个版本将废弃该参数。
--list	查询软件包文件列表。在未来某个版本将废弃该参数。
--check	检查软件包的一致性和完整性。在未来某个版本将废弃该参数。
--quiet	静默安装，跳过交互式信息，需要配合install、uninstall或者upgrade使用。在未来某个版本将废弃该参数。
--nox11	不使用x11模式运行。在未来某个版本将废弃该参数。
--noexec	解压软件包到当前目录，但不执行安装脚本。配套--extract=<path>使用，格式为--noexec --extract=<path>。在未来某个版本将废弃该参数。
--extract=<path>	解压软件包中文件到指定目录，需要配合--noexec参数一起使用。在未来某个版本将废弃该参数。
--tar arg1 [arg2 ...]	对软件包执行tar命令，使用tar后面的参数作为命令的参数。例如执行--tar xvf命令，解压run安装包的内容到当前目录。在未来某个版本将废弃该参数。
--install	安装软件包。可以指定安装路径--install-path=<path>，也可以不指定安装路径，直接安装到默认路径下。
--install-path=<path>	指定安装路径。当环境上存在全局配置文件“ascend_docker_runtime_install.info”时，指定的安装路径必须与全局配置文件中保存的安装路径保持一致。如用户想更换安装路径，需先卸载原路径下的Ascend Docker Runtime软件包并确保全局配置文件“ascend_docker_runtime_install.info”已被删除。 若5.0.RC1版本之前的Ascend Docker Runtime是通过ToolBox安装包安装的，则该文件不存在，不需要删除。 若不指定安装路径，将安装到默认路径“/usr/local/Ascend”。 若通过该参数指定了安装目录，运行用户需要对指定的安装路径有可读写权限。
--uninstall	卸载已安装的软件。如果安装时指定了安装路径，那么卸载时也需要指定安装路径，安装路径的参数为--install-path=<path>。
--upgrade	升级软件。如果安装时指定了安装路径，那么升级时也需要指定安装路径，安装路径的参数为--install-path=<path>。

参数	说明
--install-type=<type>	<p>仅支持在以下产品安装或升级Ascend Docker Runtime包时使用该参数：</p> <ul style="list-style-type: none">• Atlas 200 AI加速模块（RC场景）• Atlas 200I SoC A1 核心板• Atlas 500 智能小站（型号 3000）• Atlas 500 A2 智能小站• Atlas 200I A2 加速模块• Atlas 200I DK A2 开发者套件 <p>该参数用于设置Ascend Docker Runtime的默认挂载内容，且需要配合“--install”一起使用，格式为--install --install-type=<type>。<type>可选值为：</p> <ul style="list-style-type: none">• A200• A200ISoC• A500• A500A2• A200IA2（支持Atlas 200I A2 加速模块和Atlas 200I DK A2 开发者套件）
--ce=<ce>	<p>仅在使用iSula启动容器时需要指定该参数，参数值为isula，并且需要配合--install或者--uninstall一起使用，不能单独使用。</p>
--version	<p>打印Ascend Docker Runtime版本。</p>

3.1.4 确认组件状态

用户在安装后验证Ascend Docker Runtime是否安装成功或需要确认Ascend Docker Runtime状态时可参考本章节进行操作。

操作步骤

步骤1 Ascend Docker Runtime安装后，会修改配置文件daemon.json中的代码，如下加粗部分所示（Containerd场景不涉及）。

```
{
  "default-runtime": "ascend",
  "exec-opts": [
    "native.cgroupdriver=systemd"
  ],
  "registry-mirrors": [
    "https://docker.mirrors.ustc.edu.cn"
  ],
  "runtimes": {
    "ascend": {
      "path": "/usr/local/Ascend/Ascend-Docker-Runtime/ascend-docker-runtime",
      "runtimeArgs": []
    }
  }
}
```

```
...  
}
```

同时自动生成默认挂载内容配置文件“/etc/ascend-docker-runtime.d/base.list”。

说明

Docker场景中daemon.json路径为“/etc/docker/daemon.json”。

步骤2（仅Docker场景）执行**docker info**查看默认Runtime是否为ascend，若是则表示Ascend Docker Runtime安装成功，示例如下加粗部分所示。

```
...  
Swarm: inactive  
Runtimes: ascend runc  
Default Runtime: ascend  
Init Binary: docker-init
```

----结束

3.2 使用工具安装

ascend-deployer工具提供Ascend Docker Runtime的离线安装，用户可以通过工具快速和批量的安装Ascend Docker Runtime。如需使用ascend-deployer工具安装，请参考《[ascend-deployer用户指南](#)》进行操作。

说明

Ascend Docker Runtime日志路径为“/var/log/ascend-docker-runtime/”。

4 使用指导

- 4.1 场景介绍
- 4.2 （可选）配置自定义挂载内容
- 4.3 在Docker客户端使用
- 4.4 在K8s集成Docker使用
- 4.5 在Containerd客户端使用
- 4.6 在K8s集成Containerd使用

4.1 场景介绍

Ascend Docker Runtime支持以下场景使用，支持用户AI作业能够以Docker容器的方式在昇腾设备平滑运行、为用户默认挂载驱动以及设备相关的文件。

- Docker客户端
- 在K8s集成Docker
- K8s集成Docker
- Containerd客户端
- K8s集成Containerd

4.2 （可选）配置自定义挂载内容

Ascend Docker Runtime会为用户默认挂载驱动以及基础配置文件base.list中的全部内容。若用户需要挂载文件里的全部路径，则跳过本小节；若用户不需要挂载基础配置文件base.list中的全部内容时，可新增自定义配置文件，减少挂载的内容。自定义配置文件挂载内容须基于base.list文件，操作如下：

步骤1 进入配置文件目录。

```
cd /etc/ascend-docker-runtime.d/
```

该目录下已存在基础配置文件base.list，内容即Ascend Docker Runtime默认挂载内容，具体可参见[8.2 默认挂载内容](#)，原则上不允许用户修改base.list文件。

步骤2 创建新的配置文件，文件名可自定义（如hostlog.list）。

```
vi hostlog.list
```

步骤3 将需要挂载的文件或目录写入hostlog.list，保存并退出。

步骤4 执行命令，使自定义配置文件hostlog.list生效。示例如下：

```
docker run --rm -it -e ASCEND_VISIBLE_DEVICES=0 -e ASCEND_RUNTIME_MOUNTS=hostlog image-  
name:tag /bin/bash
```

----结束

4.3 在 Docker 客户端使用

Docker客户端使用Ascend Docker Runtime的启动容器命令示例如下。

- 示例1：启动容器时，挂载物理芯片ID为0的芯片。

```
docker run -it -e ASCEND_VISIBLE_DEVICES=0 image-name:tag /bin/bash
```
- 示例2：启动容器时，仅挂载NPU设备和管理设备，不挂载驱动相关目录。

```
docker run --rm -it -e ASCEND_VISIBLE_DEVICES=0 -e ASCEND_RUNTIME_OPTIONS=NODRV image-  
name:tag /bin/bash
```
- 示例3：启动容器时，挂载物理芯片ID为0的芯片，读取自定义配置文件hostlog中的挂载内容。

```
docker run --rm -it -e ASCEND_VISIBLE_DEVICES=0 -e ASCEND_RUNTIME_MOUNTS=hostlog image-  
name:tag /bin/bash
```
- 示例4：启动容器时，挂载虚拟芯片ID为100的芯片。

```
docker run -it -e ASCEND_VISIBLE_DEVICES=100 -e ASCEND_RUNTIME_OPTIONS=VIRTUAL image-  
name:tag /bin/bash
```
- 示例5：启动容器时，从物理芯片ID为0的芯片上，切分出4个AI Core作为虚拟设备并挂载至容器中。

```
docker run -it --rm -e ASCEND_VISIBLE_DEVICES=0 -e ASCEND_VNPU_SPECS=vir04 image-  
name:tag /bin/bash
```
- 示例6：启动容器时，挂载物理芯片ID为0的芯片，并且允许挂载的驱动文件中存在软链接（仅适用于Atlas 500 A2 智能小站、Atlas 200I A2 加速模块和Atlas 200I DK A2 开发者套件场景）：

```
docker run --rm -it -e ASCEND_VISIBLE_DEVICES=0 -e ASCEND_ALLOW_LINK=True image-  
name:tag /bin/bash
```

容器启动后，可执行以下命令检查相应设备和驱动是否挂载成功，每台机型具体的挂载目录参考[8.2 默认挂载内容](#)。命令示例如下：

```
ls /dev | grep davinci* && ls /dev | grep devmm_svm && ls /dev | grep hisi_hdc && ls /usr/local/Ascend/  
driver && ls /usr/local/ | grep dcmi && ls /usr/local/bin
```

使用 Ascend Docker Runtime

使用Ascend Docker Runtime支持容器运行训练、推理或其他任务。

- 以Atlas 200I SoC A1 核心板运行推理容器为例，用户请根据实际情况修改。示例如下，相关参数如[表4-1](#)和[表4-2](#)所示：

```
docker run -it -e ASCEND_VISIBLE_DEVICES=0 --device=/dev/xsmem_dev --device=/dev/event_sched --  
device=/dev/svm0 --device=/dev/sys --device=/dev/vdec --device=/dev/vpc --device=/dev/log_drv --  
device=/dev/spi_smbus --device=/dev/upgrade --device=/dev/user_config --device=/dev/ts_aisle --  
device=/dev/memory_bandwidth -v /var/dmp_daemon:/var/dmp_daemon:ro -v /var/slogd:/var/  
slogd:ro -v /var/log/npu/conf/slog/slog.conf:/var/log/npu/conf/slog/slog.conf:ro -v /usr/local/Ascend/  
driver/tools:/usr/local/Ascend/driver/tools -v /usr/local/Ascend/driver/lib64:/usr/local/Ascend/driver/  
lib64 -v /usr/lib64/aicpu_kernels:/usr/lib64/aicpu_kernels:ro -v /usr/lib64/libtensorflow.so:/usr/lib64/  
libtensorflow.so:ro -v /sys/fs/cgroup/memory:/sys/fs/cgroup/memory:ro -v /usr/lib64/  
libyaml-0.so.2:/usr/lib64/libyaml-0.so.2:ro -v /etc/ascend_install.info:/etc/ascend_install.info -v /usr/  
local/Ascend/driver/version.info:/usr/local/Ascend/driver/version.info workload-image:v1.0 /bin/bash
```

说明

- 如果Atlas 200I SoC A1 核心板的驱动是1.0.0（Ascend HDK 22.0.0）及之前的版本，则需要挂载/dev/xsmem_dev和/dev/event_sched这两个设备。
- 如果Atlas 200I SoC A1 核心板的驱动是1.0.0（Ascend HDK 22.0.0）之后的版本，则不需要挂载/dev/xsmem_dev和/dev/event_sched这两个设备。
- 以Atlas 500 A2 智能小站运行推理容器为例，用户请根据实际情况修改。示例如下，相关参数如表4-1和表4-2所示：

```
docker run --rm -it -e ASCEND_VISIBLE_DEVICES=0 -e ASCEND_ALLOW_LINK=True workload-image:v1.0 /bin/bash
```

不使用 Ascend Docker Runtime

- 以Atlas 200I SoC A1 核心板运行推理容器为例，用户请根据实际情况修改。示例如下，相关参数如表4-1和表4-2所示：

```
docker run -it --device=/dev/davinci0 --device=/dev/xsmem_dev --device=/dev/event_sched --device=/dev/svm0 --device=/dev/sys --device=/dev/vdec --device=/dev/venc --device=/dev/vpc --device=/dev/davinci_manager --device=/dev/spi_smbus --device=/dev/upgrade --device=/dev/user_config --device=/dev/ts_aisle --device=/dev/memory_bandwidth -v /etc/sys_version.conf:/etc/sys_version.conf:ro -v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi:ro -v /var/dmp_daemon:/var/dmp_daemon:ro -v /var/slogd:/var/slogd:ro -v /var/log/npu/conf/slog/slog.conf:/var/log/npu/conf/slog/slog.conf:ro -v /etc/hdcBasic.cfg:/etc/hdcBasic.cfg:ro -v /usr/local/Ascend/driver/tools:/usr/local/Ascend/driver/tools -v /usr/local/Ascend/driver/lib64:/usr/local/Ascend/driver/lib64 -v /usr/lib64/aicpu_kernels:/usr/lib64/aicpu_kernels:ro -v /usr/lib64/libtensorflow.so:/usr/lib64/libtensorflow.so:ro -v /sys/fs/cgroup/memory:/sys/fs/cgroup/memory:ro -v /usr/lib64/libyaml-0.so.2:/usr/lib64/libyaml-0.so.2:ro -v /etc/ascend_install.info:/etc/ascend_install.info -v /usr/local/Ascend/driver/version.info:/usr/local/Ascend/driver/version.info workload-image:v1.0 /bin/bash
```

说明

- 如果Atlas 200I SoC A1 核心板的驱动是1.0.0（Ascend HDK 22.0.0）及之前的版本，则需要挂载/dev/xsmem_dev和/dev/event_sched这两个设备。
- 如果Atlas 200I SoC A1 核心板的驱动是1.0.0（Ascend HDK 22.0.0）之后的版本，则不需要挂载/dev/xsmem_dev和/dev/event_sched这两个设备。
- Atlas 500 A2 智能小站不使用Ascend Docker Runtime运行推理任务请参见《Atlas 500 A2 智能小站 昇腾软件快速安装指南》中的“部署昇腾软件（预置系统场景）> 制作容器镜像”章节中的“启动容器”，运行推理任务。

参数说明

表 4-1 Ascend Docker Runtime 运行参数解释

参数	说明	举例
ASCEND_VISIBLE_DEVICES	必选使用 ASCEND_VISIBLE_DEVICES环境变量指定被挂载至容器中的NPU设备，否则挂载NPU设备失败；使用设备序号指定设备，支持单个和范围指定且支持混用；使用芯片名称指定设备，支持同时指定多个同类型的芯片名称。	<ul style="list-style-type: none">• ASCEND_VISIBLE_DEVICES=0表示将0号设备（ /dev/davinci0 ）挂载入容器中。• ASCEND_VISIBLE_DEVICES=1,3 表示将1、3号设备挂载入容器中。• ASCEND_VISIBLE_DEVICES=0-2 表示将0号至2号设备（ 包含0号和2号 ）挂载入容器中，效果同-e ASCEND_VISIBLE_DEVICES=0,1,2。• ASCEND_VISIBLE_DEVICES=0-2,4 表示将0号至2号以及4号设备挂载入容器，效果同-e ASCEND_VISIBLE_DEVICES=0,1,2,4。• ASCEND_VISIBLE_DEVICES=AscendXXX-Y，其中XXX表示芯片的型号，支持的取值为910，310和310P；Y表示物理设备ID。<ul style="list-style-type: none">- ASCEND_VISIBLE_DEVICES=Ascend910-1，表示把1号芯片挂载进容器。- ASCEND_VISIBLE_DEVICES=Ascend910-1,Ascend910-3，表示把1号芯片和3号芯片挂载进容器。

参数	说明	举例
		说明 <ul style="list-style-type: none">芯片类型需要和实际环境的芯片类型保持一致，否则将会挂载失败。不支持在一个参数里既指定设备序号又指定芯片名称，即不支持 ASCEND_VISIBLE_DEVICES=0, Ascend910-1。
ASCEND_ALLOW_LINK	<p>是否允许挂载的文件或目录中存在软链接，在Atlas 500 A2 智能小站、Atlas 200I A2 加速模块和Atlas 200I DK A2 开发者套件场景下必须指定该参数。</p> <p>说明 其他设备如Atlas 训练系列产品、Atlas A2 训练系列产品和Atlas 200I SoC A1 核心板等产品可以使用该参数，但因其默认挂载内容中不存在软链接，所以无需额外指定该参数。</p>	<ul style="list-style-type: none">ASCEND_ALLOW_LINK=True，表示在Atlas 500 A2 智能小站、Atlas 200I A2 加速模块和Atlas 200I DK A2 开发者套件场景下允许挂载带有软链接的驱动文件。ASCEND_ALLOW_LINK=False或者不指定该参数，Atlas 500 A2 智能小站、Atlas 200I A2 加速模块和Atlas 200I DK A2 开发者套件将无法使用Ascend Docker Runtime。
ASCEND_RUNTIME_OPTIONS	<p>对参数ASCEND_VISIBLE_DEVICES中指定的芯片ID作出限制：</p> <ul style="list-style-type: none">NODRV：表示不挂载驱动相关目录。VIRTUAL：表示挂载的是虚拟芯片。NODRV,VIRTUAL：表示挂载的是虚拟芯片，并且不挂载驱动相关目录。	<ul style="list-style-type: none">ASCEND_RUNTIME_OPTIONS=NODRVASCEND_RUNTIME_OPTIONS=VIRTUALASCEND_RUNTIME_OPTIONS=NODRV,VIRTUAL

参数	说明	举例
ASCEND_RUNTIME_MOUNTS	读取配置文件中的挂载内容。	<ul style="list-style-type: none">ASCEND_RUNTIME_MOUNTS=hostlogASCEND_RUNTIME_MOUNTS=hostlog,hostlog1,hostlog2 说明 <ul style="list-style-type: none">hostlog请根据实际自定义配置文件名修改。支持读取多个配置文件。
ASCEND_VNPU_SPECS	<p>从物理NPU设备中切分出一定数量的AI Core，指定为虚拟设备。支持的取值为（vir01，vir02，vir02_1c，vir04，vir04_3c，vir08，vir16，vir04_4c_dvpp、vir04_3c_ndvpp）。</p> <ul style="list-style-type: none">Atlas 推理系列产品仅支持vir01、vir02、vir02_1c、vir04、vir04_3c、vir04_4c_dvpp、vir04_3c_ndvpp。Atlas 训练系列产品仅支持vir02、vir04、vir08和vir16。 <p>需配合参数“ASCEND_VISIBLE_DEVICES”一起使用，参数“ASCEND_VISIBLE_DEVICES”指定用于切分的物理NPU设备。</p> <p>说明 参数 ASCEND_RUNTIME_OPTIONS的取值包含VIRTUAL时，ASCEND_VNPU_SPECS参数将不再生效。</p>	<p>ASCEND_VNPU_SPECS=vir04表示切分4个AI Core作为虚拟设备，挂载至容器中。</p> <p>说明</p> <ul style="list-style-type: none">vir后面的数字表示AI Core数量。c前面的数字表示AI CPU数量。dvpp表示虚拟化时包含所有数字视觉预处理模块（即VPC，VDEC，JPEGD，PNGD，VENC，JPEGE）。ndvpp表示虚拟化时没有数字视觉预处理硬件资源。

表 4-2 其他参数解释

参数	参数说明
/dev/xsmem_dev	将内存设备管理挂载到容器。

参数	参数说明
/dev/event_sched	将事件调度的设备挂载到容器。
/dev/ts_aisle	将aicpudrv驱动对应的设备挂载到容器。
/dev/svm0	将内存管理的设备挂载到容器。
/dev/sys	将dvpp相关的设备挂载到容器。
/dev/vdec	将dvpp相关的设备挂载到容器。
/dev/vpc	将dvpp相关的设备挂载到容器。
/dev/log_drv	将日志记录相关的设备挂载到容器。
/dev/upgrade	将获取昇腾系统相关配置、固件设备挂载到容器。
/dev/spi_smbus	将设备带外spi通信相关的设备挂载到容器。
/dev/user_config	将管理用户配置相关的设备挂载到容器。
/dev/memory_bandwidth	将内存带宽相关的设备挂载到容器。
-v /var/slogd:/var/slogd	将宿主机日志进程文件以只读方式挂载到容器。
-v /var/dmp_daemon:/var/dmp_daemon	将dmp守护进程挂载到容器。
-v /var/log/npu/conf/slog:/var/log/npu/conf/slog	将npu日志模块挂载到容器。
-v /usr/lib64/libyaml-0.so.2:/usr/lib64/libyaml-0.so.2:ro	将宿主机libyaml的.so文件挂载到容器中。
-v /usr/local/Ascend/driver/tools:/usr/local/Ascend/driver/tools	将驱动相关工具目录“/usr/local/Ascend/driver/tools”挂载到容器中。
-v /usr/local/Ascend/driver/lib64:/usr/local/Ascend/driver/lib64	将驱动依赖动态库目录“/usr/local/Ascend/driver/lib64”挂载到容器中。
-v /usr/lib64/libtensorflow.so:/usr/lib64/libtensorflow.so	将tensorflow的aicpu算子库文件“/usr/lib64/libtensorflow.so”挂载到容器中。
-v /usr/lib64/aicpu_kernels:/usr/lib64/aicpu_kernels	将aicpu lib库目录“/usr/lib64/aicpu_kernels”挂载到容器。
-v /sys/fs/cgroup/memory:/sys/fs/cgroup/memory:ro	将宿主机查询内存占用率所需依赖目录“/sys/fs/cgroup/memory”以只读方式挂载到容器中。
-v /etc/ascend_install.info:/etc/ascend_install.info	将宿主机安装信息文件“/etc/ascend_install.info”挂载到容器中。

参数	参数说明
<code>-v /usr/local/Ascend/driver/version.info:/usr/local/Ascend/driver/version.info</code>	将宿主机版本信息文件“/usr/local/Ascend/driver/version.info”挂载到容器中，请根据实际情况修改。
<code>workload-image:v1.0 /bin/bash</code>	生成的镜像文件。

4.4 在 K8s 集成 Docker 使用

K8s集成Docker场景下，用户只需要安装Ascend Docker Runtime即可，无需进行额外配置。使用任务yaml下发训练或推理任务时，NPU芯片的分配由Volcano和Ascend Device Plugin组件自动完成，NPU芯片及相关文件目录的挂载由Ascend Docker Runtime组件自动完成。

4.5 在 Containerd 客户端使用

Containerd客户端使用cgroup v1的用户，启动容器命令示例如下。

说明

如果是使用cgroup v2的用户，需要将下文的runtime io.containerd.runtime.v1.linux修改成runtime io.containerd.runc.v2 --runc-binary /usr/local/Ascend/Ascend-Docker-Runtime/ascend-docker-runtime。

如示例1修改后为`ctr run --runtime io.containerd.runc.v2 --runc-binary /usr/local/Ascend/Ascend-Docker-Runtime/ascend-docker-runtime -t --env ASCEND_VISIBLE_DEVICES=0 image-name:tag containerID`。示例2~6修改方式与示例1相同。

- 示例1：启动容器时，挂载物理芯片ID为0的芯片。
`ctr run --runtime io.containerd.runtime.v1.linux -t --env ASCEND_VISIBLE_DEVICES=0 image-name:tag containerID`
- 示例2：启动容器时，仅挂载NPU设备和管理设备，不挂载驱动相关目录。
`ctr run --runtime io.containerd.runtime.v1.linux -t --env ASCEND_VISIBLE_DEVICES=0 --env ASCEND_RUNTIME_OPTIONS=NODRV image-name:tag containerID`
- 示例3：启动容器时，挂载物理芯片ID为0的芯片，读取自定义配置文件hostlog中的挂载内容。
`ctr run --runtime io.containerd.runtime.v1.linux -t --env ASCEND_VISIBLE_DEVICES=0 --env ASCEND_RUNTIME_MOUNTS=hostlog image-name:tag containerID`
- 示例4：启动容器时，挂载虚拟芯片ID为100的芯片。
`ctr run --runtime io.containerd.runtime.v1.linux -t --env ASCEND_VISIBLE_DEVICES=100 --env ASCEND_RUNTIME_OPTIONS=VIRTUAL image-name:tag containerID`
- 示例5：启动容器时，从物理芯片ID为0的芯片上，切分出4个AI Core作为虚拟设备并挂载至容器中。
`ctr run --runtime io.containerd.runtime.v1.linux -t --env ASCEND_VISIBLE_DEVICES=0 --env ASCEND_VNPU_SPECS=vir04 image-name:tag containerID`
- 示例6：启动容器时，挂载物理芯片ID为0的芯片，并且允许挂载的驱动文件中存在软链接（仅适用于Atlas 500 A2 智能小站、Atlas 200I A2 加速模块和Atlas 200I DK A2 开发者套件场景）：
`ctr run --runtime io.containerd.runtime.v1.linux -t --env ASCEND_VISIBLE_DEVICES=0 -e ASCEND_ALLOW_LINK=True image-name:tag /bin/bash`

启动命令相关参数如[表4-3](#)解释。

容器启动后，可执行以下命令检查相应设备和驱动是否挂载成功，每台机型具体的挂载目录参考[8.2 默认挂载内容](#)。命令示例如下：

```
ls /dev | grep davinci* && ls /dev | grep devmm_svm && ls /dev | grep hisi_hdc && ls /usr/local/Ascend/  
driver && ls /usr/local/ | grep dcmi && ls /usr/local/bin
```

说明

- 可用的芯片ID可通过如下方式查询确认：
 - 物理芯片ID：
`ls /dev/davinci*`
 - 虚拟芯片ID：
`ls /dev/vdavinci*`
- image-name:tag：镜像名称与标签，请根据实际情况修改。如“ascend-tensorflow:tensorflow_TAG”。
- containerID：容器ID。使用ctr启动容器需要指定容器ID，如“c1”。
- 若用户不需要挂载基础配置文件base中的所有内容时，可指定自定义配置文件hostlog减少挂载的内容，hostlog请根据实际创建的配置文件名修改。
- 用户在使用过程中，请勿重复定义和在容器镜像中固定ASCEND_VISIBLE_DEVICES、ASCEND_RUNTIME_OPTIONS、ASCEND_RUNTIME_MOUNTS和ASCEND_VNPU_SPECS等环境变量。

表 4-3 Ascend Docker Runtime 运行参数解释

参数	说明	举例
ASCEND_VISIBLE_DEVICES	必须使用 ASCEND_VISIBLE_DEVICES 环境变量指定被挂载至容器中的NPU设备，否则挂载NPU设备失败；使用设备序号指定设备，支持单个和范围指定且支持混用；使用芯片名称指定设备，支持同时指定多个同类型的芯片名称。	<ul style="list-style-type: none">• ASCEND_VISIBLE_DEVICES=0表示将0号设备（ /dev/davinci0 ）挂载入容器中。• ASCEND_VISIBLE_DEVICES=1,3 表示将1、3号设备挂载入容器中。• ASCEND_VISIBLE_DEVICES=0-2 表示将0号至2号设备（ 包含0号和2号 ）挂载入容器中，效果同-e ASCEND_VISIBLE_DEVICES=0,1,2• ASCEND_VISIBLE_DEVICES=0-2,4 表示将0号至2号以及4号设备挂载入容器，效果同-e ASCEND_VISIBLE_DEVICES=0,1,2,4• ASCEND_VISIBLE_DEVICES=AscendXXX-Y，其中XXX表示芯片的型号，支持的取值为910，310和310P；Y表示物理设备ID。<ul style="list-style-type: none">- ASCEND_VISIBLE_DEVICES=Ascend910-1，表示把1号芯片挂载进容器。- ASCEND_VISIBLE_DEVICES=Ascend910-1,Ascend910-3，表示把1号芯片和3号芯片挂载进容器。 <p>说明</p> <ul style="list-style-type: none">• 芯片类型需要和实际环境的芯片类型保持一致，否则将会挂载失败。• 不支持在一个参数里既指定设备序号又指定芯片名称，即不支持 ASCEND_VISIBLE_DEVICES=0，Ascend910-1。

参数	说明	举例
ASCEND_ALLOW_LINK	<p>是否允许挂载的文件或目录中存在软链接，在Atlas 500 A2 智能小站、Atlas 200I A2 AI加速模块和Atlas 200I DK A2 开发者套件场景下需要指定该参数。</p> <p>说明 其他设备如Atlas 训练系列产品、Atlas A2 训练系列产品和Atlas 200I SoC A1 核心板等产品可以使用该参数，但因其默认挂载内容中不存在软链接，所以无需额外指定该参数。</p>	<ul style="list-style-type: none"> ASCEND_ALLOW_LINK=True，表示在Atlas 500 A2 智能小站、Atlas 200I A2 AI加速模块和Atlas 200I DK A2 开发者套件场景下允许挂载带有软链接的驱动文件。 ASCEND_ALLOW_LINK=False或者不指定该参数，Atlas 500 A2 智能小站、Atlas 200I A2 AI加速模块和Atlas 200I DK A2 开发者套件将无法使用Ascend Docker Runtime。
ASCEND_RUNTIME_OPTIONS	<p>对参数ASCEND_VISIBLE_DEVICES中指定的芯片ID作出限制：</p> <ul style="list-style-type: none"> NODRV：表示不挂载驱动相关目录。 VIRTUAL：表示挂载的是虚拟芯片。 NODRV,VIRTUAL：表示挂载的是虚拟芯片，并且不挂载驱动相关目录。 	<ul style="list-style-type: none"> ASCEND_RUNTIME_OPTIONS=NODRV ASCEND_RUNTIME_OPTIONS=VIRTUAL ASCEND_RUNTIME_OPTIONS=NODRV,VIRTUAL
ASCEND_RUNTIME_MOUNTS	<p>读取配置文件中的挂载内容。</p>	<ul style="list-style-type: none"> ASCEND_RUNTIME_MOUNTS=hostlog ASCEND_RUNTIME_MOUNTS=hostlog,hostlog1,hostlog2 <p>说明</p> <ul style="list-style-type: none"> hostlog请根据实际自定义配置文件名修改。 支持读取多个配置文件。

参数	说明	举例
ASCEND_VNPU_SPECS	<p>从物理NPU设备中切分出一定数量的AI Core，指定为虚拟设备。支持的取值为（vir01，vir02，vir02_1c，vir04，vir04_3c，vir08，vir16，vir04_4c_dvpp、vir04_3c_ndvpp）。</p> <ul style="list-style-type: none">Atlas 推理系列产品系列处理器仅支持vir01、vir02、vir02_1c、vir04、vir04_3c、vir04_4c_dvpp、vir04_3c_ndvpp。Atlas 训练系列产品仅支持vir02、vir04、vir08和vir16。 <p>需配合参数 “ASCEND_VISIBLE_DEVICES”一起使用，参数 “ASCEND_VISIBLE_DEVICES”指定用于切分的物理NPU设备。</p> <p>说明 参数 ASCEND_RUNTIME_OPTIONS的取值包含VIRTUAL时，ASCEND_VNPU_SPECS参数将不再生效。</p>	<p>ASCEND_VNPU_SPECS=vir04表示切分4个AI Core作为虚拟设备，挂载至容器中。</p> <p>说明</p> <ul style="list-style-type: none">vir后面的数字表示AI Core数量。c前面的数字表示AI CPU数量。dvpp表示虚拟化时包含所有数字视觉预处理模块（即VPC，VDEC，JPEGD，PNGD，VENC，JPEGE）。ndvpp表示虚拟化时没有数字视觉预处理硬件资源。

4.6 在 K8s 集成 Containerd 使用

K8s集成Containerd场景下，用户需要安装Ascend Docker Runtime，并参照本章节完成相应修改。使用任务yaml下发训练或推理任务时，NPU芯片的分配由Volcano和Ascend Device Plugin组件自动完成；NPU芯片及相关文件目录的挂载由Ascend Docker Runtime组件自动完成。

操作步骤

步骤1 修改Containerd配置文件“/etc/containerd/config.toml”。

- 使用cgroup v1的用户，根据实际情况修改返回类型和runtime安装路径，示例如下加粗部分所示。

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes]
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]
  runtime_type = "io.containerd.runtime.v1.linux"
  runtime_engine = ""
  runtime_root = ""
  privileged_without_host_devices = false
```

```
base_runtime_spec = ""
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
[plugins."io.containerd.grpc.v1.cri".cni]
bin_dir = "/opt/cni/bin"
conf_dir = "/etc/cni/net.d"
max_conf_num = 1
conf_template = ""
[plugins."io.containerd.grpc.v1.cri".registry]
[plugins."io.containerd.grpc.v1.cri".registry.mirrors]
[plugins."io.containerd.grpc.v1.cri".registry.mirrors."docker.io"]
endpoint = ["https://registry-1.docker.io"]
[plugins."io.containerd.grpc.v1.cri".image_decryption]
key_model = ""
...
[plugins."io.containerd.monitor.v1.cgroups"]
no_prometheus = false
[plugins."io.containerd.runtime.v1.linux"]
shim = "containerd-shim"
runtime = "/usr/local/Ascend/Ascend-Docker-Runtime/ascend-docker-runtime"
runtime_root = ""
no_shim = false
shim_debug = false
[plugins."io.containerd.runtime.v2.task"]
platforms = ["linux/amd64"]
...
```

- 若使用的是cgroup v2，需要做如下加粗部分的修改，示例如下。

```
[plugins."io.containerd.grpc.v1.cri".containerd.default_runtime.options]
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes]
[plugins."io.containerd.grpc.v2.cri".containerd.runtimes.runc]
base_runtime_spec = ""
cni_conf_dir = ""
cni_max_conf_num = 0
container_annotations = []
pod_annotations = []
privileged_without_host_devices = false
runtime_engine = ""
runtime_path = ""
runtime_root = ""
runtime_type = "io.containerd.runc.v2"
[plugins."io.containerd.grpc.v2.cri".containerd.runtimes.runc.options]
  BinaryName = "/usr/local/Ascend/Ascend-Docker-Runtime/ascend-docker-runtime"
  CriuImagePath = ""
  CriuPath = ""
  CriuWorkPath = ""
  IoGid = 0
  IoUid = 0
  NoNewKeyring = false
  NoPivotRoot = false
  Root = ""
  ShimCgroup = ""
  SystemdCgroup = true
...
```

步骤2 修改节点上kubelet的配置文件kubeadm-flags.env。

```
vi /var/lib/kubelet/kubeadm-flags.env
```

增加如下加粗内容，其他参数已省略。

```
KUBELET_KUBEADM_ARGS="... --container-runtime=remote --container-runtime-endpoint=/run/  
containerd/containerd.sock ..."
```

步骤3 执行命令，重启Containerd和kubelet，示例如下。

```
systemctl daemon-reload && systemctl restart containerd kubelet
```

----结束

5 升级

版本要求

表 5-1 升级路径说明

源版本	目标版本	是否支持升级
MindX 3.0.0	MindX 5.0.0.2	是
MindX 5.0.RC1	MindX 5.0.0.2	是
MindX 5.0.RC2	MindX 5.0.0.2	是
MindX 5.0.RC3	MindX 5.0.0.2	是
MindX 5.0.0	MindX 5.0.0.2	是
MindX 5.0.0.1	MindX 5.0.0.2	是

目前只支持root用户升级Ascend Docker Runtime，执行如下步骤进行升级操作。

步骤1 安装包下载完成后，首先进入安装包（run包）所在路径。

```
cd <path to run package>
```

步骤2 执行以下命令，为软件包添加可执行权限。

```
chmod u+x Ascend-docker-runtime_{version}_linux-{arch}.run
```

步骤3 通过以下命令升级Ascend Docker Runtime。

- 在默认路径下升级Ascend Docker Runtime，需要依次执行以下命令。
./Ascend-docker-runtime_{version}_linux-{arch}.run --upgrade
- 在指定路径下升级Ascend Docker Runtime，需要依次执行以下命令，“--install-path”参数为指定的升级路径。
./Ascend-docker-runtime_{version}_linux-{arch}.run --upgrade --install-path=<path>

回显示例如下，表示升级成功。

```
Uncompressing ascend-docker-runtime 100%
...
[INFO] Ascend Docker Runtime upgrade success
```

步骤4 执行以下命令，使新版Ascend Docker Runtime生效。

- Docker场景
systemctl daemon-reload && systemctl restart docker
- Containerd场景
systemctl daemon-reload && systemctl restart containerd

----结束

说明

若当前的Ascend Docker Runtime是通过ToolBox安装包安装的，升级时需要从ToolBox安装路径中找到Ascend Docker Runtime的script路径，执行**uninstall.sh**脚本将3.0.0版本之前的Ascend Docker Runtime卸载，然后下载新的Ascend Docker Runtime安装包进行安装，安装完成后需要重启服务器。

6 卸载

- 情况一：使用不同安装路径。
用户在卸载Ascend Docker Runtime时需要针对不同容器引擎，根据[步骤1](#)进行两次卸载操作，每次卸载需要指定相应的安装路径，即--install-path参数。
- 情况二：使用相同安装路径。
用户在卸载Ascend Docker Runtime时，只需根据[步骤1](#)进行一次卸载操作，卸载完成之后需要手动将另一引擎的daemon.json文件还原为Ascend Docker Runtime安装之前的内容。

若用户需要保留其中一个容器引擎，需要在Ascend Docker Runtime卸载之后，针对相应场景进行重新安装。

步骤1 可以选择以下方式中的一种卸载Ascend Docker Runtime软件。

- 方式一：（推荐）使用软件包卸载
 - a. 首先进入安装包（run包）所在路径。

```
cd <path to run package>
```
 - b. 执行以下卸载命令。
 - 执行以下命令，在默认路径下卸载Ascend Docker Runtime。

```
./Ascend-docker-runtime_{version}_linux-{arch}.run --uninstall
```
 - 执行以下命令，在指定路径下卸载Ascend Docker Runtime。

```
./Ascend-docker-runtime_{version}_linux-{arch}.run --uninstall --install-path=<path>
```

回显示例如下，表示卸载成功。

```
Uncompressing ascend-docker-runtime 100%
...
[INFO] Ascend Docker Runtime uninstall success
```
- 方式二：使用脚本卸载
 - a. 首先进入Ascend Docker Runtime的安装路径下的“script”目录（默认安装路径为：“/usr/local/Ascend/Ascend-Docker-Runtime”）：

```
cd /usr/local/Ascend/Ascend-Docker-Runtime/script
```
 - b. 运行卸载的脚本进行卸载。

```
./uninstall.sh
```

回显示例如下，表示卸载成功。

```
[INFO]: You will recover Docker's daemon
...
[INFO] uninstall.sh exec success
```

步骤2 Containerd场景需要修改“/etc/containerd/config.toml”文件，将文件还原成使用Ascend Docker Runtime之前的配置。

步骤3 重启服务。

- Docker场景
systemctl daemon-reload && systemctl restart docker
- Containerd场景
systemctl daemon-reload && systemctl restart containerd

----结束

7 FAQ

7.1 系统日志出现ascend-docker-r[xxx]:syscall 439相关打印

7.1 系统日志出现 ascend-docker-r[xxx]:syscall 439 相关打印

问题描述

系统日志出现ascend-docker-r[xxx]:syscall 439相关打印或journalctl -f | grep -E 'syscall'不断输出关于syscall 439和437的调用日志记录。

原因分析

对于低于4.16-rc3版本之前的Linux（Arm）内核而言，如果进程使用了自定义用户态系统调用号，该调用日志就会被记录到系统日志，并且被kdump捕获记录。

解决方案

说明

Linux内核开发人员认为这些打印并不一定是问题，没有必要进行记录，因此4.16-rc3及以上版本都已经删除相关代码。

用户可以通过升级内核版本解决该日志打印问题。

8 参考信息

8.1 边缘容器日志输出指导

8.2 默认挂载内容

8.3 命令说明

8.1 边缘容器日志输出指导

使用背景

由于边缘设备（如Atlas 500 A2 智能小站）存储空间有限，并且边缘设备多采用eMMC等flash作为存储介质，该介质存在使用寿命的限制。为避免存储空间过快被写满从而影响业务或存储介质过快达到使用寿命，用户可以参考本章节边缘容器日志的输出建议，使边缘容器以合适的方式输出日志。

输出方式

当前Atlas硬件上运行的边缘容器应用一般是通过K8s兼容的边缘管理平台来进行管理，如华为云IEF或基于KubeEdge搭建的第三方边缘平台等。在该平台下，容器日志的输出方式主要分为以下三种：

- 容器控制台标准输出（STDOUT和STDERR）方式
- （推荐）挂载到主机目录方式
- 容器日志直接输出到日志服务

说明

如果系统中有日志服务器，建议直接在容器中将日志输出到日志服务中；如果没有，建议采用挂载到主机目录的方式输出日志，减少日志对硬件和其他业务影响的风险。

容器控制台标准输出方式

在这种方式下，应用将容器的日志输出到标准输出。缺省情况下，Docker引擎捕捉所有容器的标准输出，使用JSON格式写入到文件里，该文件会保存到主机的“/var/lib/docker/containers/containerid”目录下，如图8-1所示。

图 8-1 containerid-json.log 文件所在路径示例

```
Euler:/var/lib/docker/containers/579f4d6e454417517b70fb35b7fb7d79d2b54c38a93ad47f7a959455295e2826 # pwd
/var/lib/docker/containers/579f4d6e454417517b70fb35b7fb7d79d2b54c38a93ad47f7a959455295e2826
Euler:/var/lib/docker/containers/579f4d6e454417517b70fb35b7fb7d79d2b54c38a93ad47f7a959455295e2826 # ll
total 28
-rw-r----- 1 root root    0 Mar 15 08:53 579f4d6e454417517b70fb35b7fb7d79d2b54c38a93ad47f7a959455295e2826-json.log
```

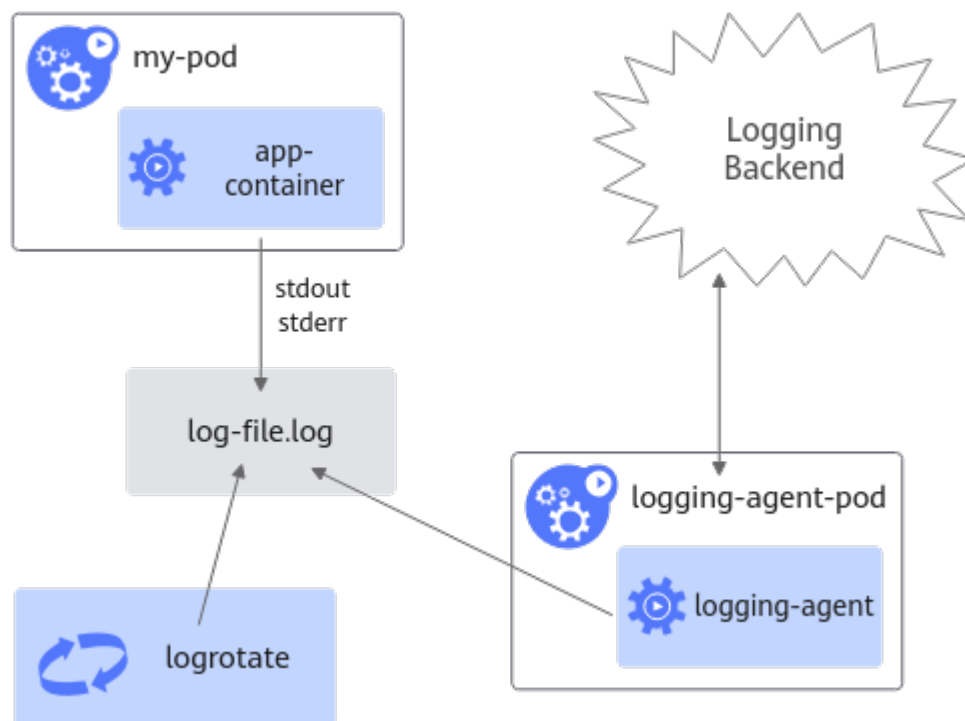
说明

如果边缘管理平台不支持该目录下日志文件的绕接或日志绕接配置错误，会导致/var/lib/docker被占满，从而影响新容器的部署及其他容器业务的正常运行。故不建议采用该方式。

（推荐）挂载到主机目录方式

该方式下边缘平台日志收集的方式如图8-2所示。

图 8-2 方案架构



应用将容器日志挂载到边缘主机上。边缘管理平台提供主机上日志收集能力，并将主机文件日志进行绕接。

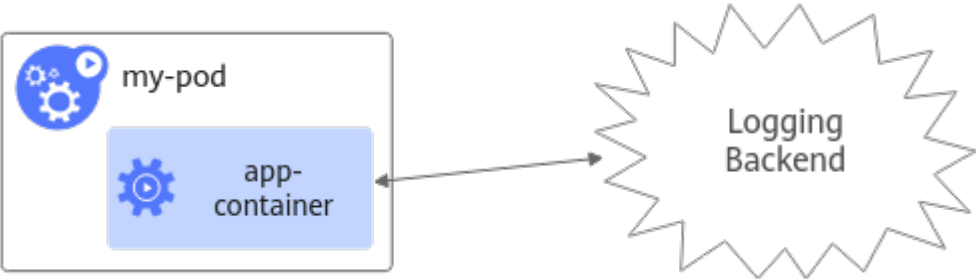
说明

- 应用可以将容器日志挂载到主机上的非关键大容量目录，建议不要挂载到eMMC等存储介质上，避免影响硬件整体寿命。
- 边缘容器管理平台一般会支持该能力，以减少对系统目录var/lib/docker的影响。基于安全性考虑，该配置需要符合所在组织的安全要求。

容器日志直接输出到日志服务

如图8-3所示，应用环境里如果有日志服务器，可以将日志直接输出到外部日志服务器，使日志不在边缘环境里落盘，最大限度减少对硬件和其他业务的影响。

图 8-3 方案架构



8.2 默认挂载内容

Ascend Docker Runtime 默认挂载内容

Ascend Docker Runtime会根据实际环境情况默认以只读方式挂载以下目录和文件到容器中。

表 8-1 默认挂载目录和文件（Atlas 200 AI 加速模块 RC 场景）

路径	说明
/dev/davinciX	NPU设备，X是ID号。例如：davinci0。
/dev/davinci_manager	管理设备。
/usr/local/Ascend/driver/tools	目录，驱动提供的工具包。
/usr/local/Ascend/driver/lib64	目录，驱动提供的用户态库。
/usr/local/sbin/npu-smi	文件，NPU-SMI工具。
/etc/hdcBasic.cfg	文件，hdc基础文件。
/etc/sys_version.conf	文件，驱动的版本信息。

表 8-2 默认挂载目录和文件（Atlas 200I SoC A1 核心板）

路径	说明
/dev/davinciX	NPU设备，X是ID号。例如：davinci0。
/dev/davinci_manager	davinci相关的设备管理的设备。
/usr/local/bin/npu-smi	文件，NPU-SMI工具。
/etc/hdcBasic.cfg	文件，hdc基础文件。
/etc/sys_version.conf	文件，驱动的版本信息。

表 8-3 默认挂载目录和文件（Atlas 200I A2 加速模块和 Atlas 200I DK A2 开发者套件）

路径	说明
/dev/davinciX	NPU设备，X是ID号。例如：davinci0。
/dev/davinci_manager	davinci相关的设备管理的设备。
/dev/svm0	内存管理的设备。
/dev/ts_aisle	aicpudrv驱动设备，为任务调度提供事件驱动的渠道接口。
/dev/upgrade	驱动设备。
/dev/sys	
/dev/vdec	设备文件，支撑推理业务。
/dev/vpc	
/dev/pngd	
/dev/venc	
/dev/dvpp_cmdlist	
/dev/log_drv	日志驱动设备。
/etc/sys_version.conf	文件，驱动的版本信息。
/etc/hdcBasic.cfg	文件，hdc基础文件。
/usr/local/sbin/npu-smi	文件，NPU-SMI工具。
/usr/local/Ascend/driver/lib64	目录，驱动提供的用户态库。
/usr/lib64/aicpu_kernels/	
/var/slogd	文件，日志组件。
/var/dmp_daemon	文件，dmp守护进程。
/usr/lib64/libcrypto.so.1.1	文件，驱动所需动态库。
/usr/lib64/libyaml-0.so.2	openEuler 22.03需要挂载该路径。
/usr/lib/aarch64-linux-gnu/libcrypto.so.1.1	文件，驱动所需动态库。 Ubuntu 22.04需要挂载该路径。
/usr/lib/aarch64-linux-gnu/libyaml-0.so.2	
/usr/lib64/libaicpu_processor.so	文件，驱动所需动态库。
/usr/lib64/libaicpu_prof.so	
/usr/lib64/libaicpu_sharder.so	
/usr/lib64/libadump.so	

路径	说明
/usr/lib64/libtsd_eventclient.so	
/usr/lib64/libaicpu_scheduler.so	
/usr/lib64/libdcmi.so	
/usr/lib64/libmpi_dvpp_adapter.so	
/usr/lib64/libstackcore.so	

表 8-4 默认挂载目录和文件（Atlas 500 智能小站（型号 3000））

路径	说明
/dev/davinciX	NPU设备，X是ID号。例如：davinci0。
/dev/davinci_manager	管理设备。
/dev/hisi_hdc	管理设备。
/dev/devmm_svm	管理设备。
/home/data/miniD/driver/lib64	目录，驱动提供的用户态库。
/usr/local/lib/libdcmi.so	文件，DCMI动态库。
/usr/local/bin/npu-smi	文件，NPU-SMI工具。

表 8-5 默认挂载目录和文件（Atlas 500 A2 智能小站）

路径	说明
/dev/davinciX	NPU设备，X是ID号。例如：davinci0。
/dev/davinci_manager	davinci相关的设备管理的设备。
/dev/svm0	内存管理的设备。
/dev/ts_aisle	aicpudrv驱动设备，为任务调度提供事件驱动的渠道接口。
/dev/upgrade	驱动设备。
/dev/sys	
/dev/vdec	设备文件，支撑推理业务。
/dev/vpc	
/dev/pngd	
/dev/venc	

路径	说明
/dev/dvpp_cmdlist	
/dev/log_drv	日志驱动设备。
/usr/local/Ascend/driver/lib64	目录，驱动提供的用户态库。
/usr/lib64/aicpu_kernels	
/usr/local/sbin/npu-smi	文件，NPU-SMI工具。
/etc/sys_version.conf	文件，驱动的版本信息。
/etc/ld.so.conf.d/mind_so.conf	驱动动态库路径配置文件
/etc/hdcBasic.cfg	文件，hdc基础文件。
/var/dmp_daemon	文件，dmp守护进程。
/var/slogd	文件，日志组件。
/usr/lib64/libcrypto.so.1.1	文件，驱动所需动态库。 openEuler 22.03或EulerOS 2.11及以上需要挂载该路径。
/usr/lib64/libyaml-0.so.2	
/usr/lib/aarch64-linux-gnu/libcrypto.so.1.1	文件，驱动所需动态库。 Ubuntu 22.04需要挂载该路径。
/usr/lib/aarch64-linux-gnu/libyaml-0.so.2	
/usr/lib64/libsemanage.so.2	文件，驱动所需动态库。
/usr/lib64/libmmpa.so	
/usr/lib64/libdrvdsmi.so	
/usr/lib64/libdcmi.so	
/usr/lib64/libstackcore.so	
/usr/lib64/libmpi_dvpp_adapter.so	
/usr/lib64/libaicpu_scheduler.so	
/usr/lib64/libaicpu_processor.so	
/usr/lib64/libaicpu_prof.so	
/usr/lib64/libaicpu_sharder.so	
/usr/lib64/libadump.so	
/usr/lib64/libtsd_eventclient.so	

表 8-6 默认挂载目录和文件（其他设备）

路径	说明
/dev/davinciX	NPU设备，X是ID号。例如：davinci0。
/dev/davinci_manager	管理设备。
/dev/hisi_hdc	管理设备。
/dev/devmm_svm	管理设备。
/usr/local/Ascend/driver/lib64	目录，驱动提供的用户态库。
/usr/local/Ascend/driver/include	目录，驱动提供的头文件。
/usr/local/dcmi	目录，DCMI头文件和库。
/usr/local/bin/npu-smi	文件，NPU-SMI工具。

8.3 命令说明

Ascend Docker Runtime安装后，会在安装目录生成可执行工具，涉及到的指令为内部命令，用户请勿直接使用，相关指令如表8-7所示。

表 8-7 命令说明

工具名	短指令	长指令	其他参数类型	其他参数位置
ascend-docker-cli	p	pid	-	-
	r	rootfs	-	-
	o	options	-	-
	f	mount-file	-	-
	l	allow-link	-	-
	i	mount-dir	-	-
ascend-docker-plugin-install-helper	-	add	-	1
	-	rm	-	1
	h	-	-	-
	-	-	destPath	2
	-	-	srcPath	3
	-	-	installPath	安装时为4
	-	-	reserveDefault	安装时为5，卸载时为4

工具名	短指令	长指令	其他参数类型	其他参数位置
ascend-docker-runtime	0	create	-	-
	b	bundle	-	-
ascend-docker-destroy	-	-	cardId	1
-	-	-	deviceId	2
-	-	-	vDevicId	3

 说明

- 因为Ascend Docker Runtime会将输入参数直接传递至runc或者docker-runc，所以runc/docker-runc的相关参数也会被Ascend Docker Runtime接受，用户请自行参考所在环境的runc/docker-runc的命令行选项使用相关参数。
- ascend-docker-hook工具会忽略参数运行，运行时会接受标准输入。

A 附录

A.1 用户信息列表

请周期性的更新用户的密码，避免长期使用同一个密码带来的风险。

系统用户

用户	描述	初始密码	密码修改方法
root	超级用户。	用户自定义	使用passwd命令修改。
HwHiAiUser	驱动run包的运行用户。	用户自定义	使用passwd命令修改。
hwMindX	集群调度组件默认的运行用户，默认设置为nologin。	无	-
HwBaseUser	Atlas 200I SoC A1 核心板上驱动相关设备运行用户，安装驱动时由驱动run包或者用户自行创建，默认设置为nologin。	无	-
HwDmUser	Atlas 200I SoC A1 核心板上驱动相关设备运行用户，安装驱动时由驱动run包或者用户自行创建，默认设置为nologin。	无	-

集群调度组件容器内用户

用户	描述	初始密码	密码修改方法
root	超级用户。	无	-
HwHiAiUser	驱动run包的运行用户，非Atlas 200I SoC A1 核心板上的集群调度组件容器内默认为nologin，该用户不可登录。	无	
hwMindX	集群调度组件容器内默认的运行用户，默认设置为nologin。	无	
HwBaseUser	Atlas 200I SoC A1 核心板上驱动相关设备运行用户，集群调度组件容器内由用户自行创建，建议设置为nologin。	无	
HwDmUser	Atlas 200I SoC A1 核心板上驱动相关设备运行用户，集群调度组件容器内由用户自行创建，该用户不可登录。	无	

A.2 修订记录

发布日期	修订说明
2023-12-29	第一次正式发布。
2024-02-07	5.0.0.2补丁版本正式发布。