AgentSphere

最佳实践

文档版本	01
发布日期	2026-10-29



华为云计算技术有限公司

版权所有 © 华为云计算技术有限公司 2026。 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


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云计算技术有限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，华为云计算技术有限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。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 使用 AgentSphere 智能体沙箱部署 OpenClaw..... 1
2 沙箱指定路径日志采集最佳实践（Python）..... 12
3 按用户隔离挂载 SFS Turbo 文件系统..... 26
4 AgentSphere 细粒度权限管理最佳实践.....38

 版权所有 © 华为云计算技术有限公司

1 使用 AgentSphere 智能体沙箱部署 OpenClaw

操作场景

随着AI Agent在企业业务中的广泛应用，开发者需要一个安全、隔离的运行环境来部署和调试AI Agent应用。OpenClaw是一款开源的AI Agent网关，支持通过即时通讯软件与大模型进行对话交互，适用于智能客服、企业内部助手等场景。然而，直接在本地或生产服务器上运行OpenClaw存在安全风险和资源管理困难——Agent应用可能执行不可预测的代码操作，与宿主环境共享资源也容易引发冲突。

AgentSphere是华为云提供的AI Agent沙箱运行平台，能够为AI Agent提供安全隔离的代码执行环境。通过将OpenClaw部署在AgentSphere的沙箱中，既可以利用OpenClaw的Agent网关能力，又能借助沙箱的隔离机制保障宿主环境安全。

本文介绍如何在AgentSphere中创建OpenClaw沙箱实例，包括制作自定义镜像、通过E2B SDK创建沙箱以及在浏览器中访问OpenClaw的完整流程。

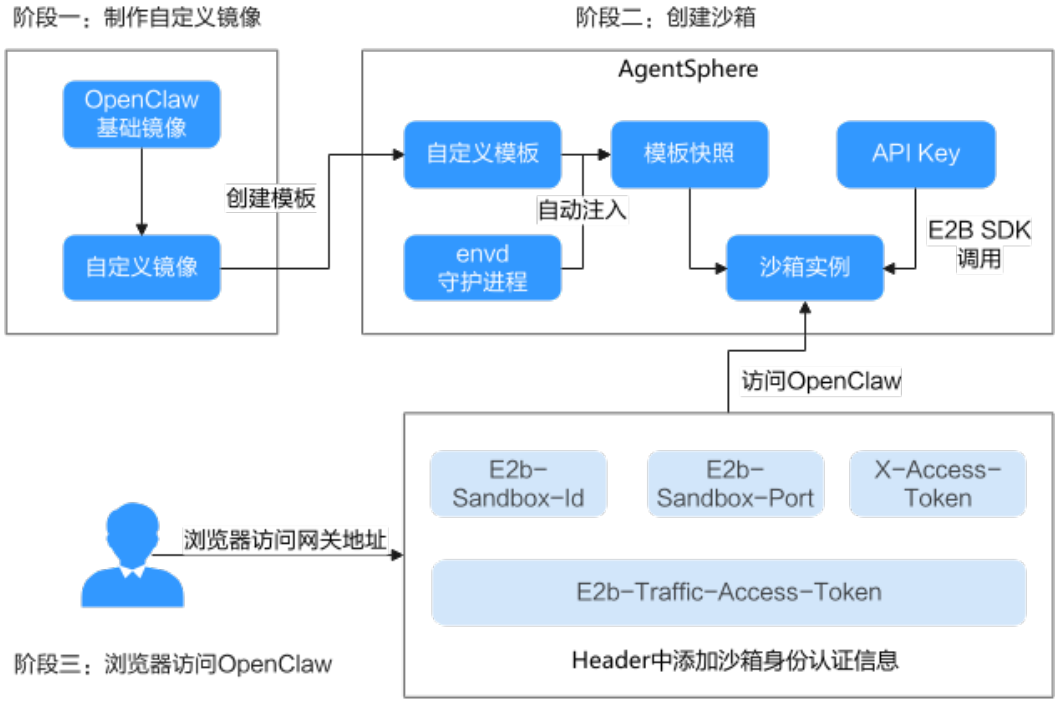
原理说明

在AgentSphere中创建OpenClaw的整体方案架构如下，分为镜像制作、沙箱创建和访问三个阶段：

1. **镜像制作阶段**：以OpenClaw官方镜像为基础，注入OpenClaw配置文件，构建自定义镜像，并使用该自定义镜像创建沙箱模板。
2. **沙箱模板创建阶段**：AgentSphere以自定义镜像为基础，自动注入ENVD，并完成设置健康检查、挂载外部存储等操作，构建出可以快速启动的模板快照。
3. **沙箱创建阶段**：通过E2B SDK调用AgentSphere的沙箱服务API，指定沙箱模板创建沙箱实例。
4. **访问阶段**：沙箱创建成功后，通过浏览器访问模板对应的网关域名，并在Header中注入沙箱身份认证信息，即可成功访问沙箱内的OpenClaw服务，进入OpenClaw对话界面。

版权所有 © 华为云计算技术有限公司

图 1-1 整体方案架构



前提条件

准备一个AgentSphere网关实例，并获取网关的公网访问域名。详情请参见**创建网关实例**。

阶段一：制作自定义镜像

本阶段以OpenClaw官方镜像为基础，集成ENVD守护进程和启动脚本，构建自定义Docker镜像并推送至SWR镜像仓库。

准备 OpenClaw 基础镜像

OpenClaw官方提供了基础Docker镜像，自定义镜像将以此为基础进行构建。

步骤1 在本地或构建服务器上执行以下命令，拉取OpenClaw最新镜像。

```
docker pull ghcr.io/openclaw/openclaw:latest
```

步骤2 验证镜像拉取成功。

```
docker images | grep openclaw
```

回显结果中能看到ghcr.io/openclaw/openclaw且TAG为latest，说明镜像拉取成功。

----结束

准备 OpenClaw 配置文件

OpenClaw启动时需要读取openclaw.json配置文件，用于定义Agent工作空间、消息渠道等运行参数。

步骤1 创建openclaw.json文件，示例配置如下：

```
{
 "gateway": {
```


版权所有 © 华为云计算技术有限公司

```
  "bind": "lan",
  "mode": "local",
  "port": 18789,
  "auth": {
    "mode": "token",
    "token": "testgatewaytoken"
  },
  "trustedProxies": [
    "127.0.0.1",
    "169.254.169.0/24"
  ],
  "allowRealIpFallback": true,
  "controlUi": {
    "allowedOrigins": [
      "http://localhost:18789",
      "http://127.0.0.1:18789",
      "https://<your-sandbox-url>.huaweicloud-agentnetwork.com"
    ],
    "allowInsecureAuth": true,
    "dangerouslyDisableDeviceAuth": true,
    "enabled": true
  }
 },
 "agents": {
  "defaults": {
    "workspace": "~/.openclaw/workspace"
  },
  "list": [
    {
      "id": "default",
      "name": "OpenClaw Assistant",
      "workspace": "~/.openclaw/workspace"
    }
  ]
 },
 "cron": {
  "enabled": false
 }
}
```

配置参数说明：

参数	说明	示例值
agents.defaults.workspace	Agent工作空间目录路径，用于存储Agent运行时产生的文件和数据。	~/.openclaw/workspace
gateway.bind	设置为 lan 表示网关将监听局域网内的所有可用 IP 地址，而不仅仅是本地回环地址。这意味着也可以从沙箱外部访问 OpenClaw。	lan
gateway.auth.mode	采用Token（令牌）认证方式。客户端在发起请求时，必须在请求头（如Authorization: Bearer <token>）中提供正确的令牌，否则会被拒绝。	token
gateway.auth.token	这是实际用于验证的Token（令牌）字符串。	testgatewaytoken

版权所有 © 华为云计算技术有限公司

参数	说明	示例值
gateway.trustedProxies	受信任的代理列表。	["127.0.0.1", "169.254.169.0/24"]
gateway.controlUi.allowedOrigins	允许的来源域名列表（CORS配置）。出于安全考虑，浏览器会限制跨域请求。这个列表指定了哪些域名可以访问OpenClaw的前端控制界面。	填写对应的AgentSphere网关实例域名 格式为： https://*<your-sandbox-url>*.huaweicloud-agentnetwork.com
gateway.controlUi.allowInsecureAuth	允许在没有HTTPS加密的情况下进行认证交互。在本地开发或非信任网络中方便使用，但在公网环境下极不安全，容易被中间人攻击窃取Token。本文中的设置仅作为示例。	true

步骤2 如需配置大模型对话能力，请在配置文件中补充模型相关参数。

详细配置方法请参考OpenClaw官方文档：**https://docs.openclaw.ai/gateway/configuration**

----结束

编写 Dockerfile 并构建镜像

自定义镜像需要在OpenClaw基础镜像上加入自定义的配置文件。

步骤1 在构建目录下创建Dockerfile文件，内容如下：

```
FROM ghcr.io/openclaw/openclaw:latest
RUN mkdir -p /home/node/.openclaw/workspace
chown -R node:node/home/node/.openclaw/workspace
COPY --chown=node:node openclaw.json /home/node/.openclaw/openclaw.json
```

步骤2 执行以下命令构建自定义镜像。

```
docker build -t openclaw-custom:latest .
```

步骤3 建议在推送前本地运行验证。

```
docker run --rm -p 49983:49983 openclaw-custom:latest
```

启动日志中出现“[gateway] ready”，OpenClaw网关正常启动。

----结束

推送镜像到 SWR

构建并验证通过后，需要将镜像推送到华为云容器镜像服务（SWR），供创建沙箱模板时使用。

以基础版SWR为例，步骤如下：

版权所有 © 华为云计算技术有限公司

步骤1 登录**SWR控制台**，选择左侧导航栏的“组织管理”，单击页面右上角的“创建组织”，创建一个组织。

如已有组织可跳过此步骤。

步骤2 在左侧导航栏选择“我的镜像”，单击右侧“客户端上传”，在弹出的页面中单击“生成登录指令”，单击 复制临时登录指令。

步骤3 在集群节点上执行上一步复制的登录指令，登录成功会显示“Login Succeeded”。

步骤4 给镜像打上SWR标签。

```
docker tag openclaw-custom:latest swr.<region>.myhuaweicloud.com/<namespace>/openclaw-custom:latest
```

其中，*swr.<region>.myhuaweicloud.com*为镜像仓库地址可在SWR控制台上查询，**登录指令**中末尾的域名即为镜像仓库地址；*<namespace>*为SWR中已创建的组织名称。

步骤5 推送镜像到 SWR：

```
docker push swr.<region>.myhuaweicloud.com/<namespace>/openclaw-custom:latest
```

推送完成后，可在SWR控制台对应组织的镜像列表中查看到该镜像。

----结束

阶段二：创建自定义模板并实例化沙箱

本阶段在AgentSphere控制台创建自定义沙箱模板，获取API Key，然后通过E2B-SDK创建沙箱实例并注入OpenClaw配置。

创建自定义沙箱模板

本步骤基于上一步推送至SWR的自定义镜像创建AgentSphere沙箱模板。

步骤1 登录**AgentSphere控制台**。

步骤2 在左侧导航栏中选择“沙箱模板”，并单击右上角的“创建沙箱模板”。

步骤3 填写**基础配置**。

参数	参数说明	示例
沙箱模板名称	自定义名称，需以小写字母开头，以小写字母或数字结尾，可以包含小写字母、数字和中划线，长度为1-48个字符。	openclaw-test
描述	您可以添加该沙箱模板的描述信息，最多1000字符。	-
选择镜像	单击选择镜像，选择SWR中的自定义镜像。	选择上一步已推送的镜像
资源规格	选择沙箱实例使用的CPU和内存配比。资源配额过小可能导致沙箱运行失败，过大则导致资源浪费。	2vCPUs 8GiB内存
启动命令	沙箱实例启动时执行的命令。此处可留空。	`node /app/openclaw.mjs gateway`

版权所有 © 华为云计算技术有限公司

参数	参数说明	示例
就绪检查命令	用于检测沙箱实例是否就绪。	curl -sf http://localhost:18789/healthz
环境变量	通过键值对为沙箱设置环境变量。	键：OPENCLAW_CONFIG_PATH 值：/home/node/.openclaw/openclaw.json

图 1-2 基础配置示例



步骤4 填写**网络与权限配置**。

版权所有 © 华为云计算技术有限公司

参数	参数说明	示例
入站（Inbound）网关	入站网关用于接收外部请求，配置后可通过网关访问沙箱内的服务。 如果没有可选网关，请单击“新建入站网关实例”进行创建，完成创建后单击刷新按钮。 **必填**，浏览器可通过网关域名访问OpenClaw。	选择已有网关实例
出站（Outbound）网络配置	选择**公网模式**（沙箱可通过公网对外访问，适用于调用第三方API、下载依赖等场景）或**VPC模式**（沙箱访问VPC内其他服务）。	公网模式（OpenClaw需访问外部大模型API）

步骤5 填写**日志配置**（可选）。建议开启日志对接LTS，方便后续排查问题。

步骤6 填写**高级配置**，开启健康检查，探测沙箱内业务进程存活状态，以判断运行中的沙箱实例是否可用。

本文中选择使用“HTTP请求检查”，并设置openclaw的健康检查端口为18789:/healthz，协议为HTTP。

图 1-3 设置 openclaw 的健康检查端口 18789:/healthz



步骤7 完成上述配置后，单击“创建”。

版权所有 © 华为云计算技术有限公司

模板创建成功后，在模板列表中记录**沙箱模板ID**，后续创建沙箱时需要使用。

----结束

获取 API Key

API Key是调用AgentSphere API和SDK的认证凭证，所有通过API或SDK操作沙箱资源的请求均需携带API Key进行身份鉴权。

步骤1 登录**AgentSphere控制台**。

步骤2 在左侧导航栏中选择“API Key管理”，并单击右上角的“创建API Key”。

步骤3 设置API Key参数。

参数	参数说明
有效期	根据需求设置API Key的有效期，过期后该API Key将不可用。
描述	您可以添加该API Key的描述信息。

步骤4 单击“确定”，系统将自动为您生成API Key。

⚠ 注意

为保障安全，生成的API Key将仅显示一次，请务必保存，关闭该页面后将无法再次查看。

您可以单击“下载API Key凭证”将API Key保存至本地。

----结束

使用 E2B SDK 创建沙箱

完成自定义模板和API Key准备后，通过E2B SDK的Python接口创建沙箱实例，并将OpenClaw配置文件注入沙箱。

步骤1 确保本地已安装E2B SDK和满足版本要求的Python，版本要求详见**e2b-code-interpreter**。若未安装SDK，请执行以下命令安装：

```
pip install e2b==2.42.0
pip install e2b-code-interpreter==2.9.1
```

步骤2 创建Python脚本create_sandbox.py，内容如下：

```
import os
from e2b_code_interpreter import Sandbox
from time import sleep

# ====== 请替换为AgentSphere控制台获取的真实值 ======
E2B_SANDBOX_URL = "https://<your-sandbox-url>.huaweicloud-agentnetwork.com"
E2B_API_URL = "https://agentsandbox.<region>.myhuaweicloud.com"
E2B_API_KEY = "<your_api_key>"
TEMPLATE_ID = "<template_id>"
# ================================================

# 创建沙箱实例，指定自定义模板和环境变量
sbx =  Sandbox.create(
    api_key=E2B_API_KEY,
    api_url=E2B_API_URL,
```


版权所有 © 华为云计算技术有限公司

```
    sandbox_url=E2B_SANDBOX_URL,
    template=TEMPLATE_ID,
)
print("Sandbox created:", sbx.sandbox_id)
print("Sandbox envd_access_token:",sbx._envd_access_token)
print("Sandbox traffic_access_token:",sbx.traffic_access_token)
```

代码关键参数说明：

参数	说明	注意事项
E2B_SANDBOX_URL	沙箱服务网关地址，用于浏览器访问时的流量入口。	请替换为AgentSphere控制台中实际分配的网关地址，可在控制台“沙箱模板”详情页查看其入站网关的“公网域名”获取。
E2B_API_KEY	沙箱服务API密钥，用于API调用身份认证。	请替换为实际密钥，注意妥善保管，避免泄露。
E2B_API_URL	沙箱服务API地址，用于SDK与AgentSphere平台通信。	请替换为实际API地址，请通过《AgentSphere SDK参考》“配置本地E2B SDK执行环境”章节查询当前区域的API访问地址。
template	自定义模板ID，决定沙箱的基础镜像和运行环境。	请替换为创建沙箱模板后获取的模板ID。

步骤3 执行脚本创建沙箱。

```
python create_sandbox.py
```

步骤4 验证沙箱创建成功。脚本执行后，控制台将输出**Sandbox ID**、**Traffic Access Token**和**Envd Access Token**，后续浏览器访问时需要使用。

```
[root@fg-sandbox-service-build-nc5yr openclaw-test]# python create.py
Sandbox created: 276bf64f-4b2f-4314-b61d-6ad640f456d6
Sandbox envd_access_token: 84c31b819cde6c5b6e0bf150f03be2cc3d9802ff2dbc6084d0a553856a8ea659
Sandbox traffic_access_token: 1a752fcda40dba495f3ed5fbdb7f5f1bf4cc9d8280dd7bc8734e1fa9fb857837
```

----结束

阶段三：在浏览器中访问 OpenClaw

沙箱创建成功后，通过浏览器访问OpenClaw对话界面。由于沙箱通过入站网关对外提供服务，浏览器需要通过Modify Header插件注入沙箱身份认证信息才能访问。

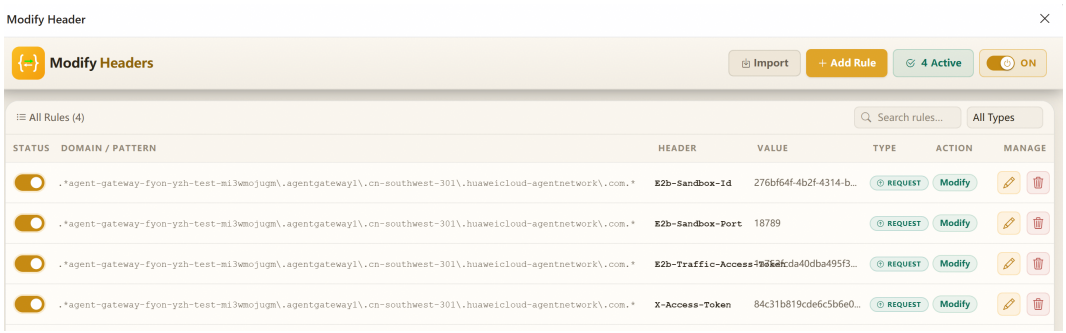
步骤1 在浏览器中安装请求头注入插件，用于在请求中注入沙箱身份认证信息。本文使用**Modify Header**插件。

步骤2 在Modify Header插件中添加以下四个请求头：

Header名称	Header值	说明
E2b-Sandbox-Id	阶段二输出的Sandbox ID	标识目标沙箱实例
E2b-Sandbox-Port	18789	OpenClaw监听端口

版权所有 © 华为云计算技术有限公司

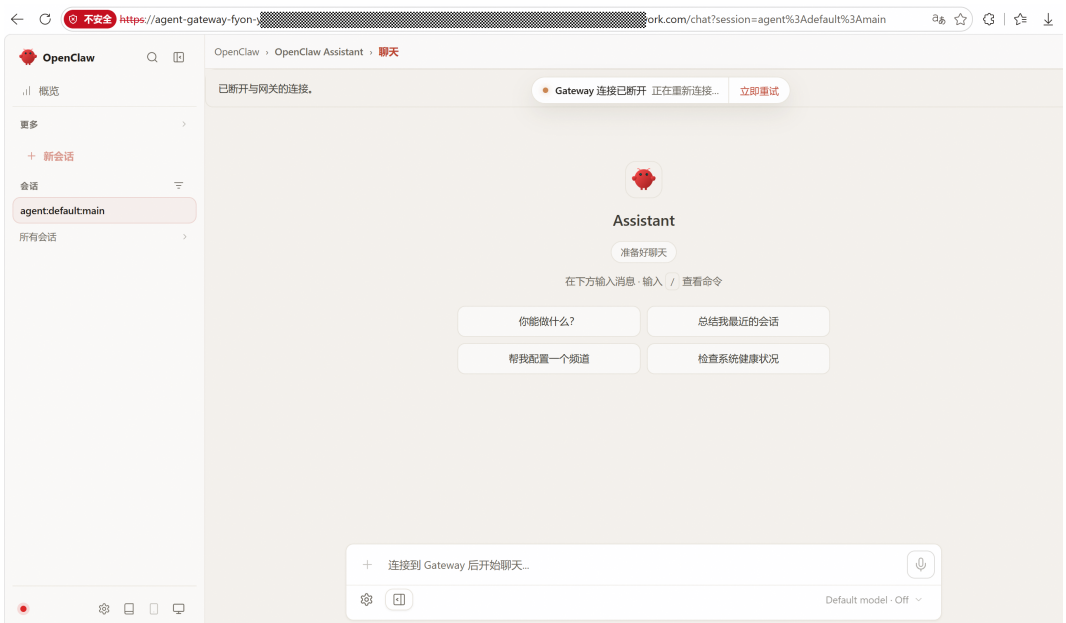
Header名称	Header值	说明
E2b-Traffic-Access-Token	阶段二输出的Traffic Access Token	沙箱流量认证令牌
X-Access-Token	阶段二输出的Envd Access Token	沙箱操作认证令牌



步骤3 在浏览器地址栏中输入沙箱网关域名（即*E2B_SANDBOX_URL*的值），按**Enter**键访问。

步骤4 在OpenClaw登录页面中输入**网关令牌**（配置文件中的gateway.auth.token值），单击**连接**。

步骤5 验证访问成功。若页面显示OpenClaw对话界面，说明已成功访问。若在openclaw.json中配置了大模型，即可在界面中进行对话交互。



----结束

常见问题

- **Q：浏览器访问OpenClaw提示连接失败怎么办？**
  A：请依次检查以下内容。

版权所有 © 华为云计算技术有限公司

a. Modify Header插件中的三个请求头是否填写正确。
b. 沙箱是否仍在运行（沙箱有生命周期限制，超时后会自动销毁）。
c. 网关令牌是否与Dockerfile中设置的OPENCLAW_GATEWAY_TOKEN一致。

- **Q：沙箱创建后ENVD启动失败怎么办?**

  A：请检查自定义镜像中的envd和envd-healthcheck.sh文件是否正确复制，以及sudoers配置是否允许node用户执行envd命令。可通过**docker run**本地启动镜像查看entrypoint.sh的输出日志进行排查。

- **Q：如何在OpenClaw中配置大模型对话?**

  A：在openclaw.json配置文件中添加模型相关参数（如API Key、模型名称等），具体配置方法请参考OpenClaw官方文档。配置完成后，重新执行阶段二中的E2B SDK脚本将更新后的配置文件写入沙箱即可。

相关文档

- **OpenClaw官方文档**：了解OpenClaw网关的完整配置项和功能说明，请参见**https://docs.openclaw.ai/gateway/configuration**。
- **AgentSphere用户指南**：了解沙箱模板、沙箱实例、API Key管理等详细操作，请参见**沙箱概述**。
- **E2B SDK文档**：了解沙箱API的完整接口说明，包括代码执行、文件操作等能力，请参见**E2B SDK官方文档**。

版权所有 © 华为云计算技术有限公司

2 沙箱指定路径日志采集最佳实践（Python）

在沙箱中运行的业务应用会将运行日志写入指定目录，但这些日志默认不会自动上报到日志服务。由于沙箱实例具有临时性，实例销毁后本地日志随之丢失，无法满足运维排障要求。您可以通过配置沙箱日志采集，将指定目录中的日志文件增量实时上报到华为云LTS（云日志服务），实现日志的持久化存储。

本文介绍如何通过Python SDK在沙箱自定义镜像中增加日志采集能力，将指定目录的日志文件增量上报到华为云LTS（云日志服务），实现日志的持久化存储和集中管理。

方案原理

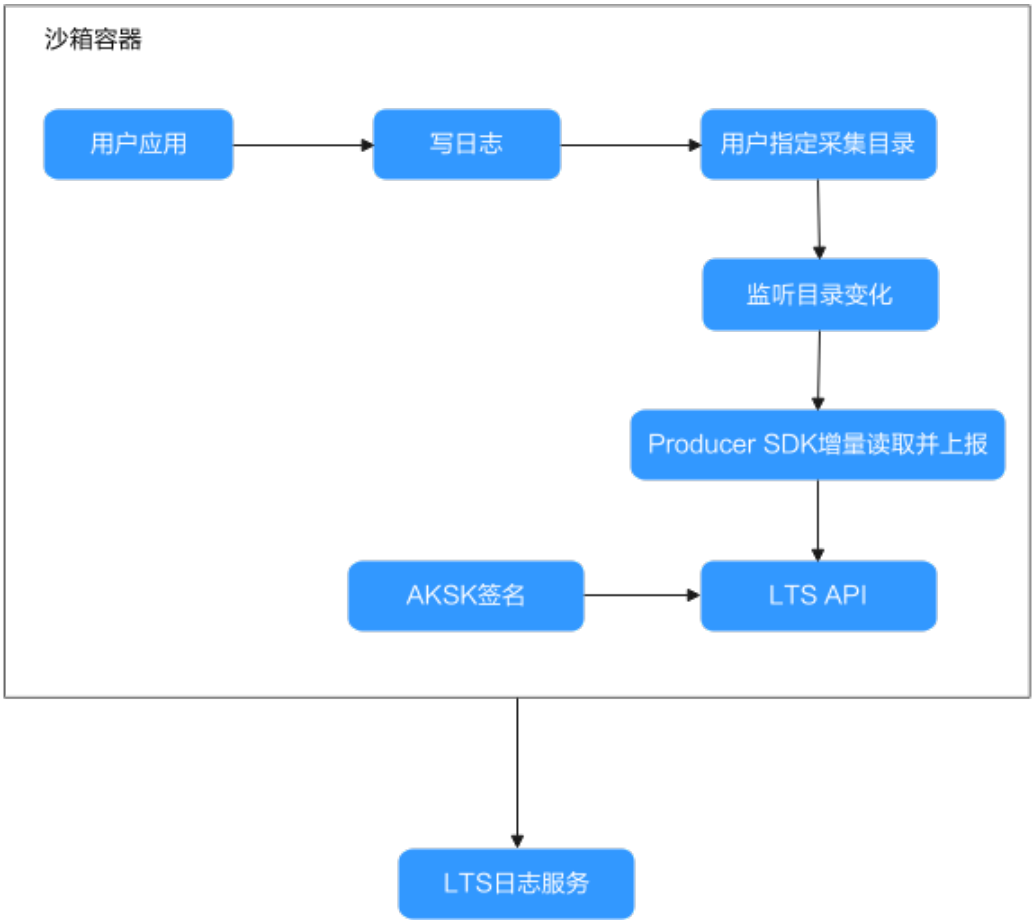
本方案在用户自定义Sandbox镜像基础上增加“日志采集”能力，通过旁路进程监听指定目录的文件变化，增量读取新增日志行，并通过LTS Producer SDK异步上报到华为云LTS。

整体工作流程如下：

1. **业务写入日志**：用户应用按原有方式将日志写入指定目录（如/var/log/app/）
2. **watchdog监听变化**：监听进程通过watchdog库检测到目录内文件的创建和修改事件
3. **增量读取**：从状态文件记录的上次偏移量位置读取新增的完整日志行
4. **异步上报**：通过Producer SDK将日志行批量发送到LTS API，由LTS完成存储和索引

版权所有 © 华为云计算技术有限公司

图 2-1 工作流程



前提条件

华为云资源准备

资源	说明
日志组	在LTS控制台中新建日志组，用于归类同一业务的日志
日志流	在LTS控制台中新建日志流，日志流是日志上报的最小单元
AK/SK	在“我的凭证”页面创建，用于LTS API认证
Project ID	在控制台右上角账号信息中获取

⚠ 注意

LTS服务按日志写入量和存储量计费，具体计费标准请参考**LTS产品价格**。建议根据日志量预估费用，并设置合理的日志保留周期。

版权所有 © 华为云计算技术有限公司

环境要求

要求	说明
Python 版本	>=3.9
基础镜像	需包含 tini（进程管理）和 setpriv（用户降权，来自 util-linux） **说明** 如果基础镜像没有tini或setpriv，可在 Dockerfile 中添加安装命令： *# Debian/Ubuntu 基础镜像* RUN apt-get update && apt-get install -y tini util-linux && rm -rf /var/lib/apt/lists/* *# CentOS/RHEL 基础镜像* RUN yum install -y tini util-linux && yum clean all
网络连通性	沙箱需能访问LTS接入点（如DNS不通，需手动配置hosts）

场景化示例

假设有一个AI推理沙箱，业务应用将推理日志写入/var/log/inference/目录，包含以下文件：

- app.log：主应用运行日志，持续追加
- tasks/：子目录，存放每个推理任务的独立日志
- audit.log：审计日志，记录每次API调用

配置LOG_WATCH_DIR=/var/log/inference后，watcher会递归监听该目录下所有文件，将新增日志行上报到LTS。在LTS控制台可以通过path字段区分日志来源（如/var/log/inference/app.log或/var/log/inference/tasks/task-001.log）。

步骤一：准备 SDK 源码和依赖包

获取LTS Python SDK源码和运行所需的Python依赖包，为补丁镜像提供日志上报和文件监听能力。

1. **获取LTS Python SDK源码**
   ```
   git clone https://gitee.com/lordstar-habile/huaweicloud-lts-python-sdk.git
   ```
   Producer组件位于源码的producer/目录下。将SDK源码目录（包含producer/、utils/、auth/等）放入构建上下文中。
2. **准备Python依赖**

   Producer SDK依赖loguru库，文件监听依赖watchdog库。根据沙箱的网络情况选择对应方式安装。
   - **方式A：沙箱可联网**——直接在Dockerfile中安装：
     ```
     RUN pip install watchdog loguru requests
     ```
   - **方式B：沙箱无法联网**——需要提前在能联网的机器上下载依赖。

     推荐使用--target目录方式：

     ```
     pip install watchdog loguru requests \
       --target ./python-packages \
       --no-cache-dir
     ```

     将python-packages/目录放入构建上下文，运行时通过PYTHONPATH指定即可。

版权所有 © 华为云计算技术有限公司

备选方式：下载whl包后离线安装

```
pip download loguru watchdog requests -d ./wheels
```

将wheels/目录上传后，在Dockerfile中离线安装：

```
COPY wheels/ /opt/sandbox-log/wheels/
RUN pip install --no-index /opt/sandbox-log/wheels/*.whl
```

说明

如果目标镜像的Python大版本或CPU架构与下载机器不同，请在与目标运行环境一致的机器上重新下载依赖，否则可能出现兼容性问题。

3. **验证方法**

在构建上下文目录中确认以下文件存在：

文件	说明
huaweicloud-lts-python-sdk/producer/	LTS SDK的Producer组件
python-packages/watchdog/	文件监听库
python-packages/loguru/	日志库

步骤二：编写日志服务配置文件

将LTS认证信息和采集参数集中在一个配置文件中，便于管理和权限控制。配置文件包含敏感凭据，权限设为0600，仅业务用户可读。

1. 创建logging.env文件，填入华为云配置信息：

```
# LTS认证配置（必填）
LTS_ACCESS_KEY='你的AK'
LTS_ACCESS_SECRET='你的SK'
LTS_PROJECT_ID='你的ProjectID'
LTS_LOG_GROUP_ID='你的日志组ID'
LTS_LOG_STREAM_ID='你的日志流ID'

# LTS接入点配置
LTS_ENDPOINT='https://lts-access.cn-south-1.myhuaweicloud.com:8102'
LTS_REGION_ID='cn-south-1'

# 采集配置
LOG_WATCH_DIR='/var/log/app'                    # 要监听的目录（递归监听）
LOG_STATE_FILE='/var/lib/log-watcher/state.json' # 状态文件路径
LOG_SOURCE='sandbox-file-watcher'               # 日志来源标识
LOG_TAIL_FROM_END='false'                       # true: 只采集启动后的新增内容
LOG_WATCH_USER='node'                           # 运行业务的用户（可选）
```

注意

高危操作提示：logging.env包含AK/SK等敏感凭据，泄露后可能导致云资源被未授权访问。请务必：

- 不要提交到源码仓库，建议在.dockerignore中排除
- 构建日志、watcher日志和诊断命令只输出配置是否存在，绝不输出配置值
- 配置发生变更时，重新构建并发布补丁镜像

2. **验证方法**

执行以下命令确认配置文件内容正确（不输出具体值）：

版权所有 © 华为云计算技术有限公司

```
# 确认必填项已设置（只检查变量是否存在，不输出值）
grep -c 'LTS_ACCESS_KEY\|LTS_ACCESS_SECRET\|LTS_PROJECT_ID\|LTS_LOG_GROUP_ID\|
LTS_LOG_STREAM_ID' logging.env
# 预期输出：5
```

步骤三：编写文件监听程序

监听程序是日志采集的核心组件，负责监听目录变化、增量读取日志行、通过Producer SDK异步上报到LTS。它使用watchdog库监听文件系统事件，通过状态文件记录每个文件的读取偏移量，确保重启后不重复发送。

创建log-watcher.py文件：

```
#!/usr/bin/env python3
"""Send appended lines from a Sandbox directory to Huawei Cloud LTS."""

import json
import logging
import os
import queue
import socket
import tempfile
import time
from pathlib import Path

from watchdog.events import FileSystemEvent, FileSystemEventHandler
from watchdog.observers import Observer

from producer.core.producer import Producer
from producer.model import lts_store
from producer.model.config import Config


REQUIRED = (
    "LTS_ACCESS_KEY",
    "LTS_ACCESS_SECRET",
    "LTS_PROJECT_ID",
    "LTS_LOG_GROUP_ID",
    "LTS_LOG_STREAM_ID",
)

class Events(FileSystemEventHandler):
    def __init__(self, changed: queue.Queue[str]):
        self.changed = changed

    def _add(self, path: str, is_directory: bool) -> None:
        if not is_directory:
            self.changed.put(path)

    def on_created(self, event: FileSystemEvent) -> None:
        self._add(event.src_path, event.is_directory)

    def on_modified(self, event: FileSystemEvent) -> None:
        self._add(event.src_path, event.is_directory)

    def on_moved(self, event: FileSystemEvent) -> None:
        self._add(event.dest_path, event.is_directory)


def configure_logging() -> None:
    logging.basicConfig(
        level=os.getenv("LOG_WATCH_LOG_LEVEL", "INFO").upper(),
        format="%(asctime)s %(levelname)s %(message)s",
    )


def required(name: str) -> str:
    value = os.getenv(name)
```


版权所有 © 华为云计算技术有限公司

```
    if not value:
        raise RuntimeError(f"missing required configuration: {name}")
    return value


def make_producer() -> tuple[Producer, str, str]:
    missing = [name for name in REQUIRED if not os.getenv(name)]
    if missing:
        raise RuntimeError("missing required configuration: " + ", ".join(missing))

    config = Config()
    config.endpoint = os.getenv("LTS_ENDPOINT", "https://lts-access.cn-south-1.myhuaweicloud.com:8102")
    config.region_id = os.getenv("LTS_REGION_ID", "cn-south-1")
    config.project_id = required("LTS_PROJECT_ID")
    config.access_key = required("LTS_ACCESS_KEY")
    config.access_secret = required("LTS_ACCESS_SECRET")
    producer = Producer.init_producer(config)
    producer.start_producer()
    return producer, required("LTS_LOG_GROUP_ID"), required("LTS_LOG_STREAM_ID")


def load_state(path: Path) -> dict[str, dict]:
    try:
        data = json.loads(path.read_text(encoding="utf-8"))
        return data.get("files", {}) if isinstance(data, dict) else {}
    except FileNotFoundError:
        return {}
    except (OSError, ValueError) as error:
        logging.warning("cannot read state file %s: %s", path, error)
        return {}


def save_state(path: Path, state: dict[str, dict]) -> None:
    path.parent.mkdir(parents=True, exist_ok=True)
    descriptor, temporary = tempfile.mkstemp(prefix=f".{path.name}.", dir=path.parent)
    try:
        with os.fdopen(descriptor, "w", encoding="utf-8") as handle:
            json.dump({"version": 1, "files": state}, handle, ensure_ascii=False)
            handle.write("\n")
        os.replace(temporary, path)
    finally:
        if os.path.exists(temporary):
            os.unlink(temporary)


def file_key(stat: os.stat_result) -> str:
    return f"{stat.st_dev}:{stat.st_ino}"


def read_complete_lines(path: Path, offset: int) -> tuple[bytes, int]:
    with path.open("rb") as handle:
        handle.seek(offset)
        data = handle.read()
    newline = data.rfind(b"\n")
    if newline < 0:
        return b"", offset
    complete = data[: newline + 1]
    return complete, offset + len(complete)


def forward(
    producer: Producer,
    group_id: str,
    stream_id: str,
    path: Path,
    state: dict[str, dict],
    tail_from_end: bool,
) -> bool:
    try:
```


版权所有 © 华为云计算技术有限公司

```
        stat = path.stat()
        if not path.is_file():
            return False
    except OSError:
        return False
    key = file_key(stat)
    entry = state.setdefault(
        key,
        {"offset": stat.st_size if tail_from_end else 0, "line_no": 0, "path": str(path)},
    )
    offset = int(entry["offset"])
    if stat.st_size < offset:
        offset = 0
        entry["line_no"] = 0
    try:
        data, next_offset = read_complete_lines(path, offset)
    except OSError as error:
        logging.warning("cannot read %s: %s", path, error)
        return False
    if not data:
        return False

    labels = {
        "source": os.getenv("LOG_SOURCE", "sandbox-file-watcher"),
        "path": str(path),
        "host_name": socket.gethostname(),
        "ip": os.getenv("LOG_SOURCE_IP", "127.0.0.1"),
    }
    line_no = int(entry["line_no"])
    for raw_line in data.splitlines():
        if not raw_line:
            continue
        line_no += 1
        message = raw_line.decode("utf-8", errors="replace")
        result = producer.send_log(
            group_id,
            stream_id,
            lts_store.generate_log([message], {**labels, "line_no": str(line_no)}),
        )
        if getattr(result, "error_code", 0):
            logging.warning("LTS enqueue failed for %s line %d", path, line_no)
            return False

    entry.update({"offset": next_offset, "line_no": line_no, "path": str(path)})
    logging.info("forwarded %s through line %d", path, line_no)
    return True


def drain(changed: queue.Queue[str]) -> set[Path]:
    paths: set[Path] = set()
    while True:
        try:
            paths.add(Path(changed.get_nowait()))
        except queue.Empty:
            return paths


def ensure_watch_dir(watch_dir: Path) -> bool:
    if watch_dir.is_dir():
        return True
    try:
        watch_dir.mkdir(parents=True, exist_ok=True)
        logging.info("created watch directory: %s", watch_dir)
        return True
    except OSError as error:
        logging.error("cannot create watch directory %s: %s", watch_dir, error)
        logging.error("please check permissions or create the directory manually")
        return False
```


版权所有 © 华为云计算技术有限公司

```
def run() -> None:
    configure_logging()
    watch_dir = Path(os.getenv("LOG_WATCH_DIR", "/var/log/app"))
    state_path = Path(os.getenv("LOG_STATE_FILE", "/var/lib/log-watcher/state.json"))
    tail_from_end = os.getenv("LOG_TAIL_FROM_END", "false").lower() in {"1", "true", "yes", "on"}
    producer, group_id, stream_id = make_producer()
    state = load_state(state_path)

    if not ensure_watch_dir(watch_dir):
        return

    while True:
        if not watch_dir.is_dir():
            logging.warning("watch directory removed: %s, attempting to recreate", watch_dir)
            if not ensure_watch_dir(watch_dir):
                return

        changed: queue.Queue[str] = queue.Queue()
        observer = Observer()
        observer.schedule(Events(changed), str(watch_dir), recursive=True)
        observer.start()
        logging.info("watching log directory: %s", watch_dir)
        try:
            pending = {path for path in watch_dir.rglob("*") if path.is_file()}
            while watch_dir.is_dir():
                try:
                    pending.add(Path(changed.get(timeout=1)))
                except queue.Empty:
                    pass
                pending.update(drain(changed))
                saved = False
                for path in list(pending):
                    try:
                        saved = forward(producer, group_id, stream_id, path, state, tail_from_end) or saved
                    except OSError as error:
                        logging.warning("error processing %s: %s", path, error)
                        pending.discard(path)
                pending.clear()
                if saved:
                    save_state(state_path, state)
        finally:
            observer.stop()
            observer.join(timeout=5)
if __name__ == "__main__":
    run()
```

核心机制说明：

- **文件追踪**：使用dev:ino（设备号:inode号）作为文件唯一标识，正确处理文件轮转和重命名
- **增量读取**：通过状态文件保存每个文件的读取偏移量，重启后从上次位置继续
- **完整行处理**：只发送以换行符结束的完整行，避免截断日志
- **目录自动创建**：如果监听目录不存在，程序会尝试自动创建
- **状态持久化**：状态文件使用原子写入（先写临时文件再重命名），避免数据损坏

步骤四：编写包装入口脚本

包装入口脚本是补丁镜像与业务镜像的桥梁。它在后台启动监听进程，然后启动原始业务入口，确保业务程序按原有方式运行，同时采集进程并行工作。当业务进程退出时，自动清理采集进程。

创建log-entrypoint.sh文件：

版权所有 © 华为云计算技术有限公司

须知

您需要将ORIGINAL_ENTRYPOINT的默认值改为基础镜像的原始入口脚本路径。也可通过环境变量覆盖，无需改脚本。

```
#!/usr/bin/env bash
set -euo pipefail
readonly LOGGING_ENV_FILE="/opt/sandbox-log/logging.env"
readonly ORIGINAL_ENTRYPOINT="${ORIGINAL_ENTRYPOINT:-/opt/onyxclaw/bin/entrypoint.sh}"
readonly RUN_AS_USER="${LOG_WATCH_USER:-node}"

if [[ ! -r "${LOGGING_ENV_FILE}" ]]; then
  echo "log collector configuration is missing: ${LOGGING_ENV_FILE}" >&2
  exit 1
fi

set -a
# shellcheck disable=SC1090
source "${LOGGING_ENV_FILE}"
set +a

watcher_pid=""
business_pid=""

stop_children() {
  for pid in "${watcher_pid}" "${business_pid}"; do
    if [[ -n "${pid}" ]] && kill -0 "${pid}" 2>/dev/null; then
      kill "${pid}" 2>/dev/null || true
    fi
  done
}

trap stop_children INT TERM

start_watcher() {
  echo "starting sandbox log watcher for ${LOG_WATCH_DIR}"
  setpriv --reuid="${RUN_AS_USER}" --regid="${RUN_AS_USER}" --init-groups \
    env PYTHONPATH="/opt/sandbox-log/python-packages:/opt/sandbox-log/lts-sdk${PYTHONPATH:+:$
{PYTHONPATH}}" \
    python3 -u /opt/sandbox-log/log-watcher.py &
  watcher_pid=$!
}

start_watcher
"${ORIGINAL_ENTRYPOINT}" "$@" &
business_pid=$!

set +e
wait "${business_pid}"
status=$?
set -e
stop_children
wait "${watcher_pid}" 2>/dev/null || true
exit "${status}"
```

步骤五：编写 Dockerfile

继承用户原始镜像，只叠加采集所需的文件和依赖，不删除、覆盖或替换原始业务程序。构建时自动验证依赖可导入，失败会中断构建，避免问题留到运行时。

创建Dockerfile文件：

```
ARG BASE_IMAGE
FROM ${BASE_IMAGE}

ARG APP_USER=node
```


版权所有 © 华为云计算技术有限公司

```
USER root

# 如果基础镜像没有tini和setpriv，取消下面的注释安装
# RUN apt-get update && apt-get install -y tini util-linux && rm -rf /var/lib/apt/lists/*

# 复制离线依赖和SDK
COPY python-packages/ /opt/sandbox-log/python-packages/
COPY huaweicloud-lts-python-sdk/ /opt/sandbox-log/lts-sdk/

# 复制采集脚本和入口脚本
COPY log-watcher.py /opt/sandbox-log/log-watcher.py
COPY log-entrypoint.sh /opt/sandbox-log/log-entrypoint.sh
COPY logging.env /opt/sandbox-log/logging.env

# 设置权限并验证依赖可导入
RUN chmod 0755 /opt/sandbox-log/log-watcher.py /opt/sandbox-log/log-entrypoint.sh \
 && chown -R ${APP_USER}:${APP_USER} /opt/sandbox-log \
 && chmod 0600 /opt/sandbox-log/logging.env \
 && PYTHONPATH=/opt/sandbox-log/python-packages:/opt/sandbox-log/lts-sdk \
    python3 -c "import watchdog; from producer.core.producer import Producer; print('OK')"

ENTRYPOINT ["tini", "-s", "--", "/bin/bash", "/opt/sandbox-log/log-entrypoint.sh"]
```

关键要点：

- FROM必须是用户需要保留的业务镜像
- 不要删除、覆盖或替换原始业务程序
- 构建时验证watchdog和LTS SDK可被导入，失败会中断构建
- logging.env权限设为0600，仅业务用户可读
- 使用tini作为init进程，正确处理信号转发

构建上下文的完整文件结构如下：

```
构建上下文/
├── Dockerfile                    # 补丁镜像构建文件
├── log-entrypoint.sh             # 包装入口脚本
├── log-watcher.py                # 文件监听程序
├── logging.env                   # 日志服务配置（不要提交到源码仓库）
├── python-packages/              # 离线Python依赖
│   ├── watchdog/
│   ├── loguru/
│   ├── requests/
│   ├── certifi/
│   ├── charset_normalizer/
│   ├── idna/
│   ├── urllib3/
│   └── six.py
└── huaweicloud-lts-python-sdk/   # LTS Python SDK源码
    ├── producer/
    ├── utils/
    └── auth/
```

步骤六：构建镜像

基于用户基础镜像构建包含日志采集能力的补丁镜像。构建过程中会自动验证依赖可用性，确保镜像可用后再发布。

执行构建命令：

```
docker build \
  --build-arg BASE_IMAGE='<用户基础镜像>:<版本>' \
  --build-arg APP_USER='node' \
  -t '<镜像仓库>/<名称>:<日志补丁版本>' \
  .
```


版权所有 © 华为云计算技术有限公司

验证方法：构建成功后，执行以下命令验证镜像可用：

```
# 验证镜像构建成功
docker images | grep '<镜像仓库>/<名称>'

# 验证依赖可导入
docker run --rm '<镜像仓库>/<名称>:<日志补丁版本>' \
  python3 -c "import watchdog; from producer.core.producer import Producer; print('OK')"
# 预期输出：OK
```

步骤七：发布并验证日志采集

将补丁镜像推送到镜像仓库，创建新的Sandbox模板，通过实际业务日志写入验证端到端的采集链路是否通畅。

1. 推送补丁镜像到镜像仓库
2. 基于补丁镜像创建新的Sandbox模板
3. 使用新模板创建全新的Sandbox
4. 触发真实业务日志写入到LOG_WATCH_DIR指定目录
5. 在LTS控制台查询日志，验证采集结果

说明

镜像变更后必须创建新的Sandbox。已有Sandbox不会自动获得新的入口脚本或新的依赖。

配置参数说明

LTS认证配置

参数名	参数解释	约束限制	取值范围	默认值	示例
LTS_ACCESS_KEY	华为云账号的访问密钥ID	不可修改，泄露后需立即禁用	字符串	-	your-access-key
LTS_ACCESS_SECRET	访问密钥的密钥	不可修改，泄露后需立即禁用	字符串	-	your-secret-key
LTS_PROJECT_ID	华为云项目ID	不可修改，需与AK/SK所属项目一致	字符串	-	0a123b4c5d6e7f8g9h0i
LTS_LOG_GROUP_ID	LTS日志组ID	不可修改，需在LTS控制台提前创建	字符串	-	abc123def456
LTS_LOG_STREAM_ID	LTS日志流ID	不可修改，需在日志组下创建	字符串	-	xyz789ghi012

LTS接入点配置

版权所有 © 华为云计算技术有限公司

参数名	参数解释	约束限制	取值范围	默认值	示例
LTS_ENDPOINT	LTS日志上报接入点地址	需与LTS_REGION_ID对应	URL格式	https://lts-access.<region>.myhuaweicloud.com:8102	https://lts-access.<region>.myhuaweicloud.com:8102
LTS_REGION_ID	LTS服务区域	需与LTS_ENDPOINT对应	华为云区域ID	cn-south-1	cn-north-4

采集行为配置

参数名	参数解释	约束限制	取值范围	默认值	示例
LOG_WATCH_DIR	要监听的目录路径	必须是绝对路径，支持递归监听子目录	绝对路径	/var/log/app	/var/log/inference
LOG_STATE_FILE	状态文件路径，保存读取偏移量	必须对业务用户可写	绝对路径	/var/lib/log-watcher/state.json	/mnt/sandbox/state.json
LOG_SOURCE	日志来源标识	用于在LTS中区分来源	字符串	sandbox-file-watcher	my-sandbox-collector
LOG_TAIL_FROM_END	是否只采集启动后的新增内容	仅接受true/false	true, false	false	true
LOG_WATCH_USER	运行业务的用户名	必须是基础镜像中存在的用户	系统用户名	node	appuser
LOG_WATCH_LOG_LEVEL	watcher自身的日志级别	级别从低到高	DEBUG, INFO, WARNING, ERROR	INFO	DEBUG
LOG_SOURCE_IP	来源IP标识	用于在LTS中标识来源IP	IP地址	127.0.0.1	192.168.1.100

日志记录格式

每条上报到LTS的记录包含以下字段：

版权所有 © 华为云计算技术有限公司

字段	说明	示例
message	原始日志行	2026-09-07 10:30:45 INFO User login successful
path	实际源文件的完整路径	/var/log/app/application.log
host_name	Sandbox主机名	sandbox-abc123
ip	来源IP	192.168.1.100
source	采集器名称	sandbox-file-watcher
line_no	文件内行号（递增）	1024

path必须是实际路径，不能是通配符，这样才能在LTS中按业务文件追踪来源。

状态文件与持久化策略

状态文件决定重启后是否重复发送或遗漏日志。

策略	状态位置	适用场景	优缺点
临时状态	容器本地目录（如/tmp/）	可接受重启后重新采集	简单，但重启后可能重复发送
持久状态	Sandbox挂载工作目录	需要跨重启保持消费位置	避免重复，但需要持久化存储

状态文件使用dev:ino（设备号:inode号）作为文件标识，正确处理文件轮转和重命名。状态文件不包含日志服务访问凭据。

常见问题

构建阶段问题

现象	常见原因	处理方式
tini: not found	基础镜像没有安装tini	在Dockerfile中安装：apt-get install -y tini或yum install -y tini
setpriv: not found	基础镜像没有安装util-linux	在Dockerfile中安装：apt-get install -y util-linux或yum install -y util-linux
import watchdog报错	离线依赖缺失或架构不匹配	重新准备与目标架构匹配的python-packages/
构建验证失败	SDK源码路径不正确或依赖未正确复制	检查构建上下文目录结构是否完整

版权所有 © 华为云计算技术有限公司

运行阶段问题

现象	常见原因	处理方式
业务服务未启动	ORIGINAL_ENTRYPOINT配置错误	核对基础镜像原始入口路径，或通过环境变量覆盖
watcher不存在	缺少依赖、权限错误或启动后立即退出	检查容器标准输出、业务用户与配置加载顺序
watcher启动失败	监听目录不存在且无权限创建	检查目录权限，或在Dockerfile中预创建目录
没有日志发送	配置缺失、SDK初始化失败或日志未以换行结束	检查配置存在性、SDK日志和文件内容
DNS解析失败	沙箱内DNS不通	手动配置hosts，联系LTS团队获取正确IP
上报返回非200	endpoint错误或网络不通	检查endpoint配置，确认网络连通性

日志质量问题

现象	常见原因	处理方式
日志重复	未保存偏移量或inode变化未处理	使用状态文件并识别截断/轮转
日志缺失	只依赖文件事件、watcher启动晚于写入	启动后扫描已有文件，并结合偏移量处理
日志路径含*	将通配符作为真实文件路径传递	监听目录，输出实际文件路径

DNS配置问题

如果沙箱内DNS无法解析lts-access域名，需要手动配置hosts：

```
echo "100.125.164.33 lts-access.<region>.myhuaweicloud.com" >> /etc/hosts
```


版权所有 © 华为云计算技术有限公司

3 按用户隔离挂载 SFS Turbo 文件系统

操作场景

在多租户的Agent应用场景下，您的业务系统（如企业服务端）会为每一个终端用户分配一个用户标识（userid），并为每个用户创建独立的运行环境（沙箱）。不同用户之间需要共享同一份SFS Turbo文件存储资源，但每个用户只能访问自己专属的存储目录，以避免用户间数据越权访问。

本文介绍如何通过Agent Identity与SFS Turbo的集成，实现SFS Turbo目录按用户隔离挂载：

- 根据指定的userid挂载对应的SFS Turbo隔离目录，若该目录不存在则自动创建。
- 限制该userid只能挂载其专属目录，无权限挂载其他用户的目录。

约束与限制

- 购买SFS Turbo实例后，需要提交工单开启该SFS Turbo实例的隔离挂载能力。
- SFS Turbo隔离挂载只支持Sandbox级别，不支持Template级别挂载。
- SFS Turbo隔离挂载场景下，Sandbox只能挂载1个目录。
- SFS Turbo实例开启隔离挂载后则不支持共享挂载，在使用Sandbox隔离挂载目录的同时，可以通过Template配置共享挂载目录。

步骤一：为 IAM 子用户配置临时凭证权限

授予IAM子用户访问Agent Identity服务，换取代表隔离用户的临时凭证（Workload Access Token, WAT）的权限。

说明

仅IAM子用户需要执行本步骤；主账号无需配置，默认拥有权限。

步骤1 创建身份策略

1. 使用管理员账号登录**统一身份认证服务新版控制台**。
2. 在左侧导航栏中选择“身份策略”，单击“创建自定义身份策略”。
3. 配置策略信息：

 版权所有 © 华为云计算技术有限公司

参数	说明
策略名称	本示例填写test-WAT-full。
策略配置方式	选择“JSON视图”。

策略内容	参考下方示例，允许IAM子用户换取隔离用户临时凭证。
	{
	"Version": "5.0",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": [
	"agentIdentity::createWorkloadAccessTokenForUserId"
	]
	}
	]
	}

步骤2 授权策略给IAM子用户

1. 单击上一步创建的test-WAT-full策略名称，切换至“授权记录”页签。
2. 单击“附加”，然后选择对应的IAM子用户，单击“确定”。

----结束

步骤二：授权 AgentSphere 获取 STS Token

创建运行时委托（Runtime agency），允许AgentSphere代表用户获取运行环境访问Agent Identity服务的临时凭证。

步骤1 创建身份策略

1. 使用用户A的账号登录**统一身份认证服务新版控制台**。
2. 在左侧导航栏中选择“身份策略”，单击“创建自定义身份策略”。
3. 配置策略信息：

参数	说明
策略名称	本示例填写policy-WAT。
策略配置方式	选择“JSON视图”。

策略内容	参考下方示例，允许沙箱获取访问Agent Identity服务的STS Token。
	{
	"Version": "5.0",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": [
	"agentIdentity::getResourceStsToken"
	]
	}
	]
	}

步骤2 创建委托

版权所有 © 华为云计算技术有限公司

1. 在左侧导航栏中选择“委托”，单击“创建信任委托”。
2. 配置委托信息：

参数	说明
委托名称	本示例填写test_metadata_trust_agency。
信任主体类型	选择“自定义信任策略”。

信任策略	参考下方示例，允许沙箱元数据服务使用该委托。
	{
	"Version": "5.0",
	"Statement": [
	{
	"Action": [
	"sts:agencies:assume",
	"sts::tagSession"
	],
	"Effect": "Allow",
	"Principal": {
	"Service": [
	"service.WorkloadSandboxMetadata"
	]
	}
	}
	]
	}

步骤3 为委托授权身份策略

1. 找到上一步创建的委托，单击“授权”。
2. 选择步骤1中创建的策略policy-WAT，单击“确定”。

----结束

步骤三：授权 Agent Identity 获取挂载 SFS Turbo 目录权限

允许Agent Identity获取SFS Turbo目录挂载权限，并限制只能挂载用户专属目录（通过${g:AgentIdentity/InboundUserId}动态解析，实现每个用户只能挂载以自身userid命名的目录）。

步骤1 创建身份策略

1. 使用用户A的账号登录**统一身份认证服务新版控制台**。
2. 在左侧导航栏中选择“身份策略”，单击“创建自定义身份策略”。
3. 配置策略信息：

参数	说明
策略名称	本示例填写policy-sfs1。
策略配置方式	选择“JSON视图”。

 版权所有 © 华为云计算技术有限公司

参数	说明

策略内容	参考下方示例，允许挂载SFS Turbo访问点下以userid命名的目录。
	{
	"Version": "5.0",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": [
	"sfsturbo:accesspoint:mount"
	],
	"Resource": [
	"sfsturbo:*:*:accesspoint:*eb2dba3c-bbd4-11f1-bb9d-fa163e188044|test/${g:AgentIdentity/InboundUserId}*"
	]
	}
	]
	}
	Resource字段格式为*sfsturbo:<region>:<account-id>:accesspoint:<share-id>/<dir-path>*，其中*<share-id>*为SFS Turbo实例ID，*<dir-path>*末段固定以*${g:AgentIdentity/InboundUserId}*标识用户专属目录。

步骤2 创建委托

1. 在左侧导航栏中选择“委托”，单击“创建信任委托”。
2. 配置委托信息：

参数	说明
委托名称	本示例填写agency-identity。
信任主体类型	选择“自定义信任策略”。

信任策略	参考下方示例，允许Agent Identity服务使用该委托。
	{
	"Version": "5.0",
	"Statement": [
	{
	"Action": [
	"sts:agencies:assume",
	"sts::SetContext",
	"sts::setSourceIdentity",
	"sts::tagSession"
	],
	"Effect": "Allow",
	"Principal": {
	"Service": [
	"service.AgentIdentity"
	]
	}
	}
	]
	}

步骤3 为委托授权身份策略

 版权所有 © 华为云计算技术有限公司

1. 找到上一步创建的委托，单击“授权”。
2. 选择步骤1中创建的策略policy-sfs1，单击“确定”。

----结束

步骤四：创建 Workload Identity

在Agent Identity服务中创建工作负载身份。工作负载身份是分配给沙箱模板、网关和Agent等工作负载的身份，用于沙箱访问Agent Identity服务时的身份验证。

步骤1 登录**Agent Identity控制台**。

步骤2 “工作负载身份”，单击“创建工作负载身份”。

步骤3 填写必要的工作负载身份信息：

参数	说明
身份名称	本示例填写workload-test。
入站身份认证	选择“API Key认证”，沙箱通过API Key完成身份认证。

步骤4 单击“立即创建”。

----结束

步骤五：创建出站身份并关联出站身份委托

在Agent Identity服务中创建出站身份（Credential Provider），将步骤2中创建的委托（agency-identity）关联到出站身份，使沙箱能够通过Agent Identity认证身份后，使用该委托获取SFS Turbo目录挂载临时凭证。

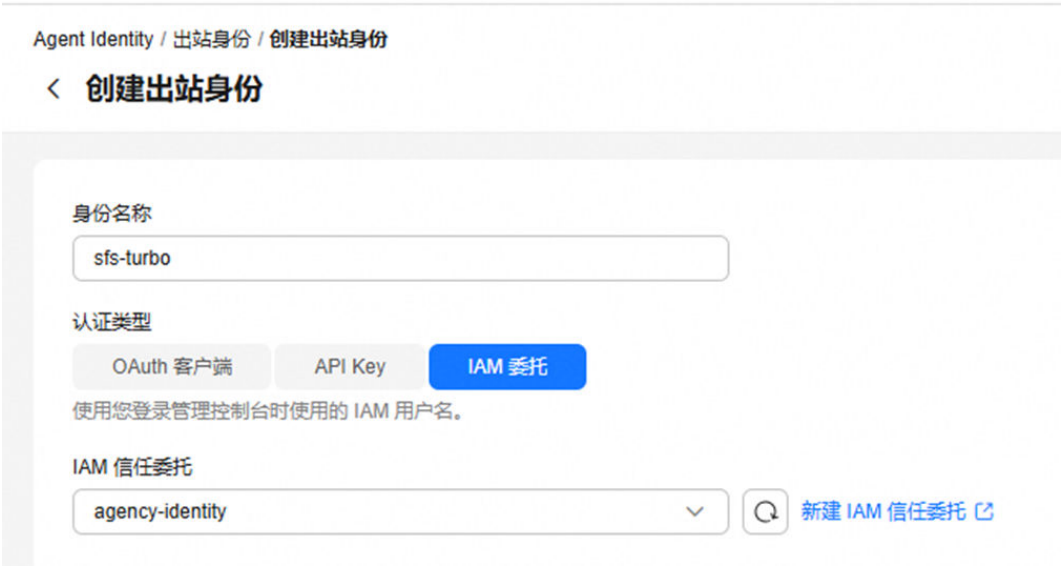
步骤1 登录**Agent Identity控制台**。

步骤2 “出站身份”，单击“创建出站身份”。

步骤3 填写必要的出站身份信息：

参数	说明
身份名称	本示例填写sfs-turbo。
认证类型	选择“IAM委托”。
IAM 信任委托	选择**步骤三**中创建的委托：agency-identity。

版权所有 © 华为云计算技术有限公司



步骤4 单击“立即创建”。

----结束

步骤六：检查 Agent Identity 策略集配置

检查Agent Identity策略集配置，防止STS Token换取失败。

步骤1 登录**Agent Identity控制台**。

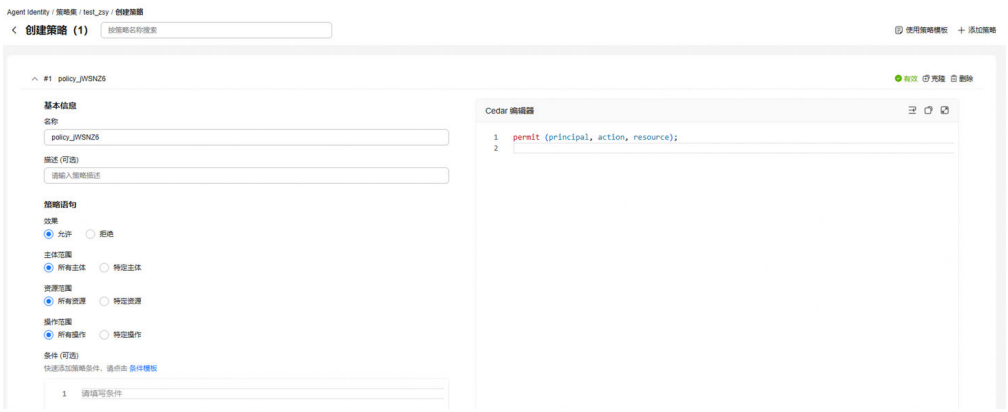
步骤2 在左侧导航栏中选择“策略集”，检查策略集配置情况：

- 如果无策略集，或策略集中无绑定实例，**可忽略本步骤**，无需额外配置。
- 如果已创建策略集且绑定了实例，且执行模式为**拦截模式**：



则需要添加策略放通STS获取权限，策略语句可配置为更细粒度的权限。

版权所有 © 华为云计算技术有限公司



----结束

步骤七：购买 VPCEP 打通 Sandbox 与 Agent Identity 的网络

当Sandbox出站网络为私有网络模式时，您需要通过VPCEP打通Sandbox与Agent Identity服务的网络。

步骤1 登录**VPC终端节点控制台**。

步骤2 在“终端节点”页面，单击“购买终端节点”。

步骤3 根据界面提示配置必选参数。更多参数说明，请参见**购买终端节点**。

表 3-1 终端节点配置参数

参数	说明
区域	终端节点所在区域，需要与AgentSphere沙箱所在区域保持一致。
计费模式	仅支持按需计费。按需计费是后付费模式，按终端节点的实际使用时长计费，可以随时开通/删除终端节点。
服务类别	选择“云服务”。
服务名称	搜索“agent-identity”，勾选对应的服务名称。
创建内网域名	如果您需要以域名的方式访问终端节点，则选择“创建内网域名”，终端节点创建完成后，即可通过内网域名直接访问终端节点。
虚拟私有云	需要选择AgentSphere沙箱所在的虚拟私有云。 **注意** 选择的VPC和子网需要与SFS Turbo实例和Sandbox保持一致。
路由表	选择一个已有的路由表。

步骤4 参数配置完成，单击“立即购买”，进行规格确认。

----结束

 版权所有 © 华为云计算技术有限公司

隔离挂载验证

前置配置完成后，您可以通过以下步骤验证按用户隔离挂载功能是否生效。

步骤1 创建沙箱模板（template）并完成模板快照构建，为创建沙箱实例准备模板配置，详情请参见**创建沙箱模板**。

其中VPC和子网需要与后续SFS Turbo实例的VPC相同，“运行时信任委托”配置为**步骤二**中创建的委托test_metadata_trust_agency。

说明

若要使用Sandbox隔离挂载目录的同时使用共享挂载目录，可以在模板中配置template级别的SFS Turbo共享目录挂载。

步骤2 通过API创建Sandbox，并携带以下3个隔离挂载Header头，实现按userid隔离挂载SFS Turbo目录。

表 3-2 隔离挂载 header 头说明

Header头	说明
X-HW-AgentSandbox-UserId	SFS Turbo隔离挂载的用户ID，用于标识SFS Turbo隔离目录。挂载目录时sharePath的末段须与该值一致，系统将基于该值获取此隔离目录的临时挂载凭证。
X-HW-AgentSandbox-Credential-Provider-Name	**步骤五**中创建的出站身份名称，本示例为sfs-turbo。
X-HW-AgentSandbox-Workload-Identity-Name	**步骤四**中创建的工作负载身份名称，本示例为workload-test。

注意

上述3个隔离挂载Header头必须齐全；若Header头不齐全，将按普通挂载方式创建沙箱，不会启用隔离挂载。

表 3-3 请求参数说明

参数	说明
templateID	验证**步骤1**中创建的模板ID（templateID）和快照ID（buildID），格式为templateID:buildID。
sfsTurboId	SFS Turbo实例ID，该实例VPC需要与沙箱模板中配置的VPC一致。
sharePath	挂载目录路径，子目录需与userid一致（本示例为sandbox-002），否则会校验报错。

版权所有 © 华为云计算技术有限公司

请求Header示例：

```
-H "X-HW-AgentSandbox-UserId: sandbox-002"
-H "X-HW-AgentSandbox-Credential-Provider-Name: sfs-turbo"
-H "X-HW-AgentSandbox-Workload-Identity-Name: workload-test"
```

body示例：

```
{
  "templateID": "00c73584-cb75-450e-a604-c692c5693473:c64f42d2-145e-4726-ab23-477c95c844de",
  "sfsTurboId": "eb2dba3c-bbd4-11f1-bb9d-fa163e188044",
  "sharePath": "/test/sandbox-002"
}
```

步骤3 创建成功后，沙箱激活，并挂载目录/test/sandbox-002，说明隔离挂载成功。

⚠ 注意

SFS Turbo隔离挂载若传入的sharePath不存在，会自动创建该sharePath。

----结束

挂载无访问权限目录拦截验证

验证隔离挂载的权限校验能力，确认Sandbox无法挂载无访问权限的目录。

- **场景1：sharePath根目录与出站身份关联的访问权限目录不一致**

  请求示例：

  ```
  -H "X-HW-AgentSandbox-UserId: sandbox-002"
  -H "X-HW-AgentSandbox-Credential-Provider-Name: sfs-turbo"
  -H "X-HW-AgentSandbox-Workload-Identity-Name: workload-test"
  -H "X-API-KEY: e2b_xxx"
  ```

  body示例：

  ```
  {
    "templateID": "00c73584-cb75-450e-a604-c692c5693473:c64f42d2-145e-4726-ab23-477c95c844de",
    "name": "sandbox-test2",
    "secure": true,
    "autoPause": false,
    "autoResume": {
      "enable": false
    },
    "storageConfig": {
      "sfsTurboMounts": [
        {
          "sfsTurboId": "eb2dba3c-bbd4-11f1-bb9d-fa163e188044",
          "sharePath": "/test-another/sandbox-002"
        }
      ]
    }
  }
  ```

  预期返回结果：挂载被拦截，提示无挂载权限。

  ```
  {
    "code": 403,
    "message": "no permission to mount SFS Turbo share path",
    "error_code": "AgentSphere.0108"
  }
  ```

- **场景2：sharePath子目录与userid不一致**

  body示例（sharePath子目录为sandbox-011，与userid sandbox-002不一致）：

 版权所有 © 华为云计算技术有限公司

```
{
  "templateID": "00c73584-cb75-450e-a604-c692c5693473:c64f42d2-145e-4726-ab23-477c95c844de",
  "name": "sandbox-test2",
  "secure": true,
  "autoPause": false,
  "autoResume": {
    "enable": false
  },
  "storageConfig": {
    "sfsTurboMounts": [
      {
        "sfsTurboId": "eb2dba3c-bbd4-11f1-bb9d-fa163e188044",
        "sharePath": "/test/sandbox-011"
      }
    ]
  }
}
```

预期返回结果：校验失败，sharePath与userId不匹配。

```
{
  "code": 400,
  "message": "sharePath does not match userId",
  "error_code": "AgentSphere.0505"
}
```

进阶配置：限制 IAM 子用户隔离挂载权限

上文中已授权IAM子用户换取WAT的权限。如果需要进一步限制IAM子用户只能使用指定的Workload Identity和userid，可以参考本节配置，实现IAM子用户级别的隔离挂载权限管控。

步骤1 限制IAM子用户WAT换取权限

创建身份策略，限制IAM子用户换取WAT的权限范围，可以从resource（workload identity）和condition（userid）两个维度进行配置。

1. 使用管理员账号登录**统一身份认证服务新版控制台**。
2. 在左侧导航栏中选择“身份策略”，单击“创建自定义身份策略”。
3. 配置策略信息：

参数	说明
策略名称	本示例填写test-WAT。
策略配置方式	选择“JSON视图”。

版权所有 © 华为云计算技术有限公司

参数	说明

策略内容	参考下方示例，仅允许使用sandbox-test1这个工作负载身份为sandbox-123这个userid换取Workload Access Token。
	{
	"Version": "5.0",
	"Statement": [
	{
	"Effect": "Allow",
	"Action": [
	"agentIdentity::createWorkloadAccessTokenForUserId"
	],
	"Resource": [
	"agentidentity:*:*:workloadIdentity:workloadIdentityDirectory/default/sandbox-test1",
	"agentidentity:*:*:workloadIdentityDirectory:default"
	],
	"Condition": {
	"StringEquals": {
	"agentIdentity:userId": [
	"sandbox-123"
	]
	}
	}
	}
	]
	}

4. 授权策略给IAM子用户
   a. 单击上一步创建的test-WAT策略名称，切换至“授权记录”页签。
   b. 单击“附加”，然后选择对应的IAM子用户，单击“确定”。

   说明

   授权子用户test-WAT时，需要确保前文中设置的test-WAT-full权限已移除，否则限制不生效。

步骤2 验证策略是否生效

- **验证场景1：userid与策略中的不一致**

  请求中携带的userid为sandbox-001，与策略中授权的sandbox-123不一致，预期返回无权限。

  ```
  -H "X-HW-AgentSandbox-UserId: sandbox-001"
  -H "X-API-KEY: {授权子用户的API-KEY}"
  ```

  预期返回结果：

  ```
  {
    "code": 403,
    "message": "AgentIdentity denied WAT access, please check identity policy permissions"
  }
  ```

- **验证场景2：workload identity与策略中授权的不一致**

  请求中携带的workload identity为sandbox-test2，与策略中授权的sandbox-test1不一致，预期返回无权限。

  ```
  -H "X-HW-AgentSandbox-UserId: sandbox-123"
  -H "X-HW-AgentSandbox-Workload-Identity-Name: sandbox-test2"
  -H "X-API-KEY: {授权子用户的API-KEY}"
  ```

  预期返回结果：

  ```
  {
    "code": 403,
  ```


版权所有 © 华为云计算技术有限公司

```
  "message": "AgentIdentity denied WAT access, please check identity policy permissions"
}
```

----结束

 版权所有 © 华为云计算技术有限公司

4 AgentSphere 细粒度权限管理最佳实践

在沙箱实验与仿真运行环境中，不同团队或不受信任的自动化脚本对计算资源的访问需求各不相同。由于沙箱内部运行的代码具有不可控性，如果鉴权范围过于宽泛，极易导致实验环境交叉污染、漏洞逃逸、高危误操作或底层资源恶意抢占。为了解决这些隐患，实现基于运行时上下文的细粒度权限控制尤为重要。

本沙箱权限管理指南是在华为云统一身份认证服务（IAM）的动态鉴权能力基础上打造的。系统支持基于IAM Token临时认证与纯Action级的动态条件控制，能够精准限制用户及临时进程对特定敏感接口（如网络变更、高危删除）的访问，从而有效地将风险锁死在沙箱边界内，确保多沙箱环境的绝对隔离与宿主机安全。

AgentSphere权限管理是在统一身份认证服务（IAM）的能力基础上，打造的细粒度权限管理功能。AgentSphere权限管理支持基于IAM的细粒度权限控制和IAM APIKEY认证，支持template级别的权限控制，从而有效地限制用户对特定资源的访问，确保资源使用的隔离性和安全性。

本示例主要介绍如何配置子用户的细粒度权限来创建沙箱。了解AgentSphere权限管理的更多内容，请参见**权限管理**。

AgentSphere 依赖的云服务

AgentSphere是面向Agent应用打造的一站式基础设施产品组合，为Agent强化学习（RL）训练和Agent在线服务（如智能助手）两大场景提供高性能沙箱运行、多层次安全防护、持久化数据底座等关键能力。

在使用AgentSphere的过程中，可能依赖的云服务如下：

云服务名称	服务简介
Agent Gateway（智能体网关）	Agent出入站安全代理网关，支持公私网接入、访问控制（黑白名单）、URL管控、MCP/API代理。
Agent Identity（智能体身份）	Agent身份与访问控制，支持入站/出站身份认证、安全策略管控、凭证管理、资源授权。
IAM（统一身份认证服务）	APIKEY生命周期管理及细粒度权限管理。
SWR (容器镜像服务)	容器镜像的生命周期管理，沙箱镜像需要托管到SWR仓库。

版权所有 © 华为云计算技术有限公司

云服务名称	服务简介
VPC（虚拟私有云）	沙箱的出入站网络控制。
OBS（对象存储服务）	可挂载用户已有的对象存储，用于加载已有skill等数据。
SFSTurbo（高性能文件弹性服务）	多个沙箱之间可以共享存储。
AOM（应用运维服务）	沙箱的监控指标上报和展示。
LTS（云日志服务）	沙箱的标准输出日志采集和上报。
NAT（NAT网关）	当前OBS/SFSTurbo等存储在沙箱模式下不支持公网访问，需要通过NAT代理至公网。
KMS（密钥管理服务）	identity服务被动依赖，用于凭证的加解密。
CSMS（云凭据管理服务）	identity服务被动依赖，用于凭证轮转。

AgentSphere在IAM中预置了以下系统身份策略，均遵循最小权限原则（仅授予完成对应操作所需的最小权限），可直接授予子用户使用：

系统身份策略名称	描述	策略类别
AgentSphereReadOnlyPolicy	访问AgentSphere只读权限。	系统身份策略
AgentSphereFullAccessPolicy	访问AgentSphere所有权限。	系统身份策略
AgentSphereRuntimeDeploymentAgencyPolicy	访问AgentSphere运行时需要的获取SWR镜像的相关权限。	系统身份策略

IAM 细粒度授权步骤

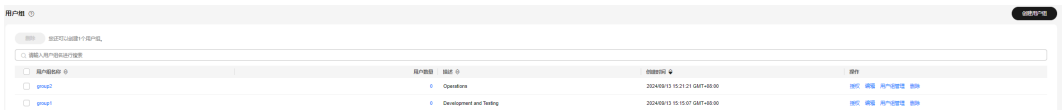
步骤一：创建用户与用户组

步骤1 登录统一身份认证服务新版控制台。

步骤2 在左侧导航栏中选择“用户组”，在页面右上角单击“创建用户组”。在弹出页面，输入“用户组名称”，单击“确定”。

本示例需要创建2个用户组，分别为，例如group1（用于开发与测试）和group2（用于运维）。

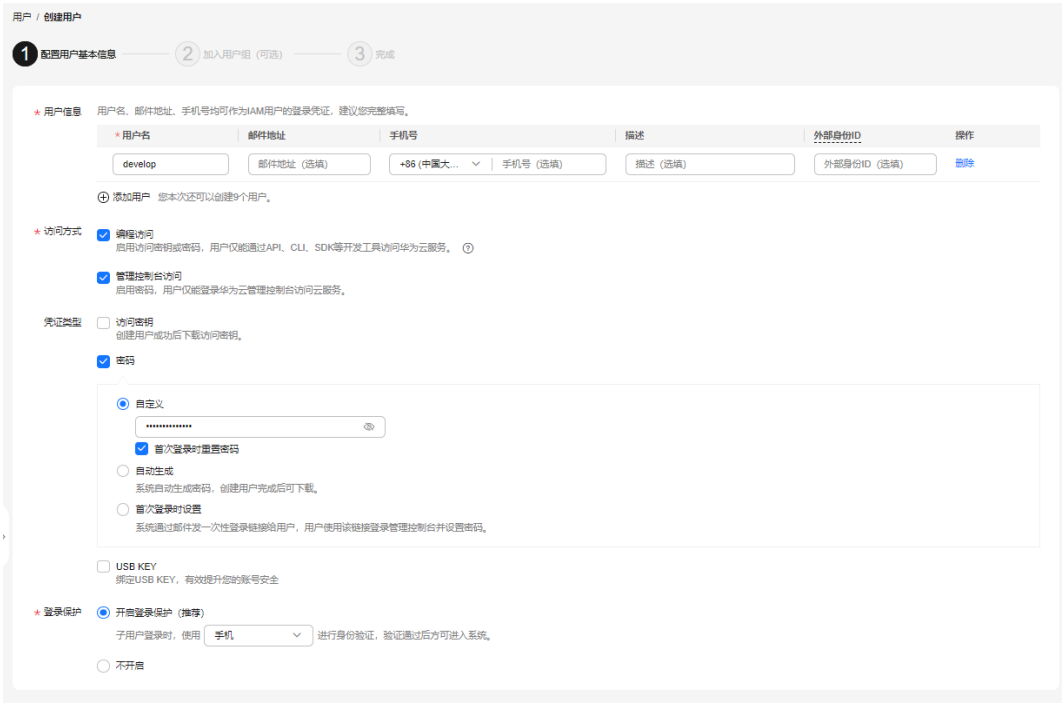
图 4-1 新建的用户组



版权所有 © 华为云计算技术有限公司

步骤3 在左侧导航栏中选择“用户”，在右上角单击“创建用户”。根据需求配置“用户基本信息”，单击“下一步”。

图 4-2 创建用户

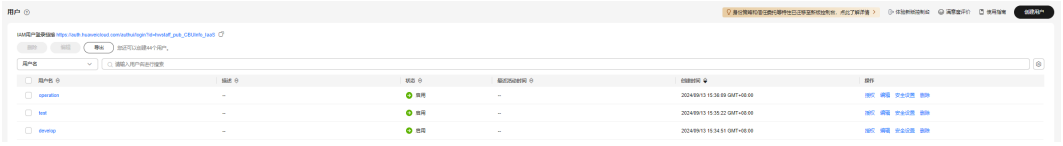


步骤4 根据需求选择加入的用户组，单击“创建用户”。

本示例共创建3个用户，分别代表开发、测试和运维，其中开发与测试加入group1，运维加入group2。

在用户列表中可以看到新建的3个用户，如图所示。

图 4-3 新建的用户



----结束

步骤二：对用户组进行 IAM 授权

管理员分别对用户组group1和group2进行IAM授权，基于云服务管理用户组的相关权限。

步骤1 在左侧导航栏中选择“用户组”，找到需要授权的用户组，单击“授权”。

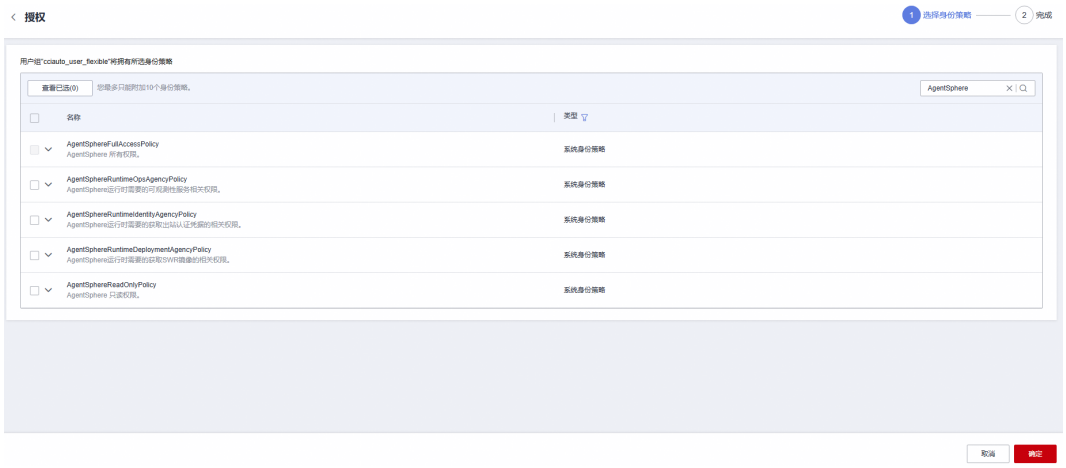
步骤2 在弹出页面中，在右上角输入“AgentSphere”，过滤出AgentSphere对应的系统身份策略。

根据需要选择策略，智能体沙箱（AgentSphere）具有2种策略，了解AgentSphereFullAccessPolicy和AgentSphereReadOnlyPolicy策略的更多内容，请参考**权限管理**。

版权所有 © 华为云计算技术有限公司

本示例中，group1选择AgentSphereFullAccessPolicy策略，group2选择AgentSphereReadOnlyPolicy策略。

图 4-4 IAM 授权



步骤3 单击“确定”后，在弹出的页面单击"完成"授权，返回到统一身份认证管理页面。

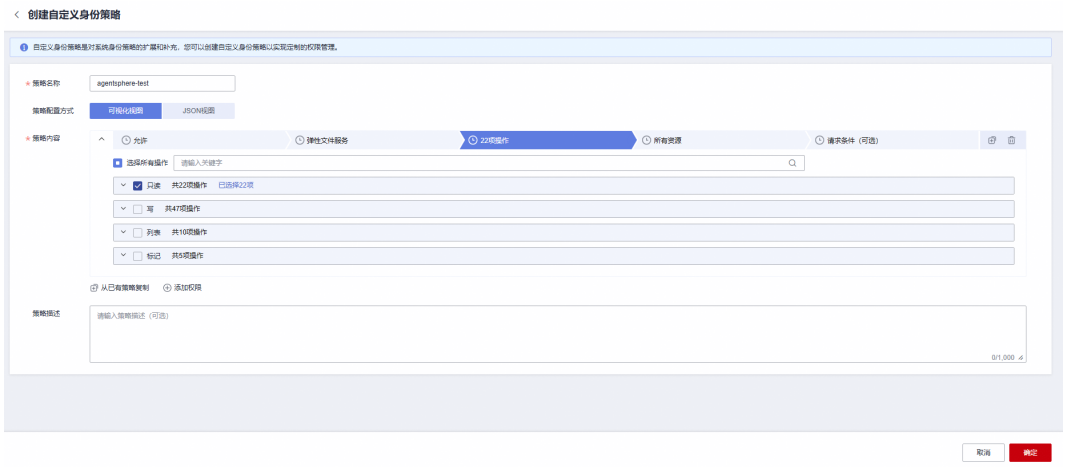


----结束

步骤三：自定义身份策略，按需添加额外所需权限

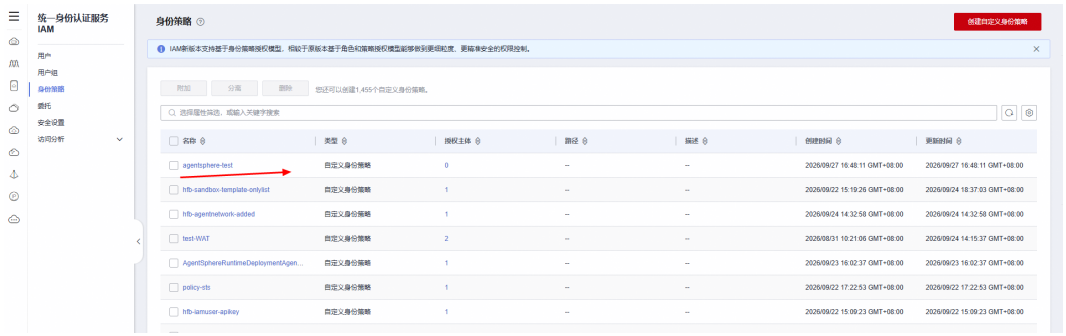
步骤1 在左侧导航栏中选择“身份策略”，单击右上角的“创建自定义身份策略”按钮。

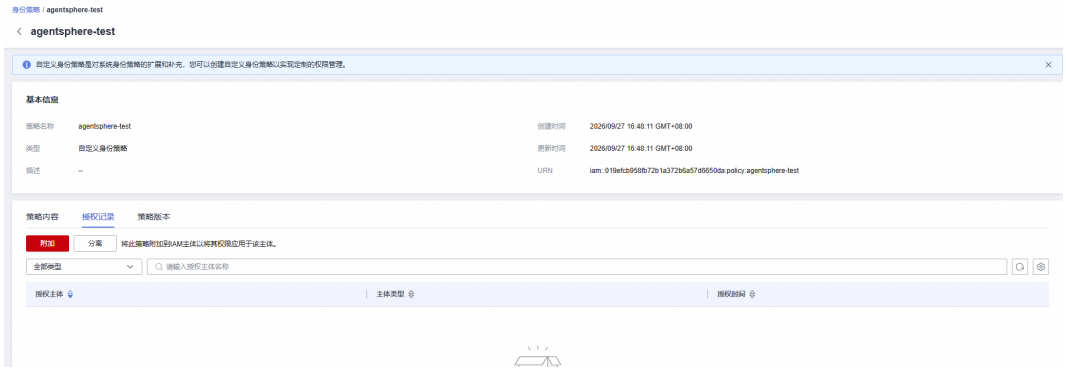
步骤2 在弹出页面中，选择所需要添加的云服务及对应的细粒度权限，单击“确认”完成自定义的身份策略创建。



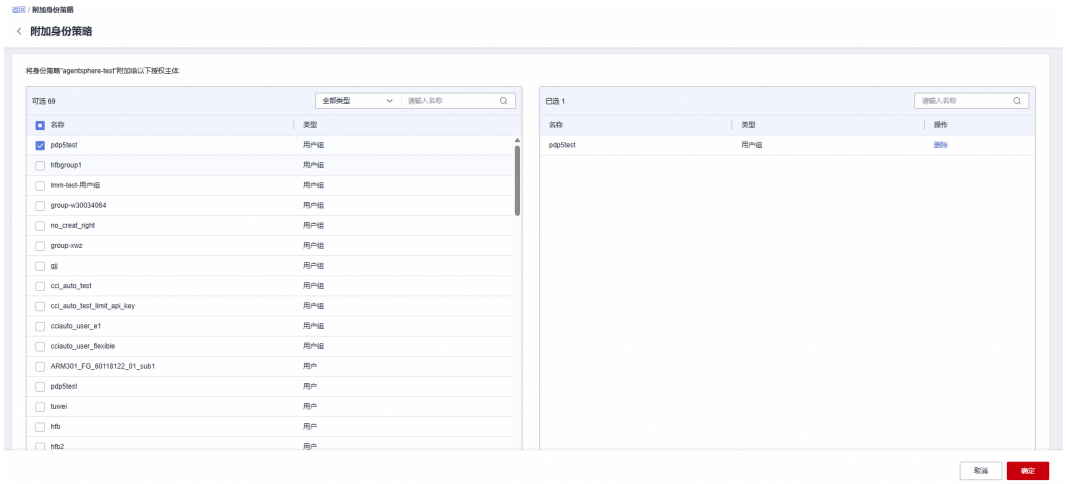
 版权所有 © 华为云计算技术有限公司

步骤3 在身份策略页面单击刚才创建的自定义身份策略查看策略详情，单击“授权记录”下的“附加”按钮，将自定义身份策略授权给需要的用户组。





步骤4 选择需要授权的用户组，完成授权。



----结束

场景示例

场景一：不授予子用户 te_admin 的权限，但是想让子用户能完全操控 AgentSphere 依赖的全部资源，进行业务快速验证

假设已新建用户组group1和子用户user1，且user1加入了group1。

步骤1 为group1授予除IAM外所有依赖服务的FullAccess系统策略包含：

 版权所有 © 华为云计算技术有限公司

- AgentSphereFullAccessPolicy
- SWRFullAccessPolicy
- VPCFullAccessPolicy
- VPCEPFullAccessPolicy
- AOMFullAccessPolicy
- LTSFullAccessPolicy

说明

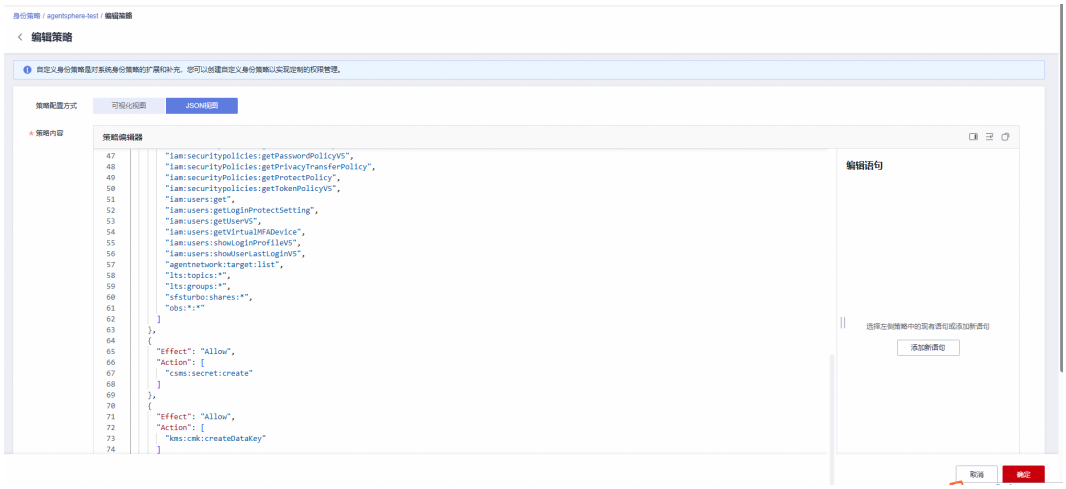
每个用户组最多授予10个策略。

步骤2 自定义身份策略agentsphere-test，授予如下细粒度权限，用于APIKEY的增删改查及系统策略查询。

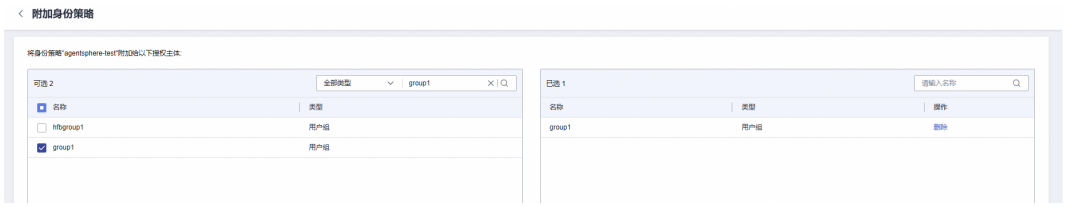
```
{
  "Version": "5.0",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:serviceSpecificCredentials:createV5",
        "iam:serviceSpecificCredentials:deleteV5",
        "iam:serviceSpecificCredentials:listV5",
        "iam:serviceSpecificCredentials:updateV5"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam::getAccessKey",
        "iam::getAsymmetricSignatureSwitchV5",
        "iam:agencies:checkRole",
        "iam:agencies:checkRoleOnDomain",
        "iam:agencies:checkRoleOnProject",
        "iam:agencies:get",
        "iam:agencies:getServiceLinkedAgencyDeletionStatusV5",
        "iam:agencies:getV5",
        "iam:agencies:listV5",
        "iam:agencies:createV5",
        "iam:agencies:attachPolicyV5",
        "iam:credentials:showAccessKeyLastUsedV5",
        "iam:groups:checkRole",
        "iam:groups:checkRoleOnDomain",
        "iam:groups:checkRoleOnProject",
        "iam:groups:checkUser",
        "iam:groups:get",
        "iam:groups:getGroupV5",
        "iam:identityProviders:get",
        "iam:identityProviders:getMapping",
        "iam:identityProviders:getOIDCConfig",
        "iam:identityProviders:getProtocol",
        "iam:identityProviders:getSAMLMetadata",
        "iam:oidcProviders:getV5",
        "iam:policies:getV5",
        "iam:policies:listV5",
        "iam:policies:getVersionV5",
        "iam:roles:get",
        "iam:samlProviders:getV5",
        "iam:securityPolicies:getApiAclPolicy",
        "iam:securityPolicies:getConsoleAclPolicy",
        "iam:securityPolicies:getLoginPolicy",
        "iam:securitypolicies:getLoginPolicyV5",
        "iam:securityPolicies:getPasswordPolicy",
        "iam:securitypolicies:getPasswordPolicyV5",
        "iam:securityPolicies:getPrivacyTransferPolicy",
```


版权所有 © 华为云计算技术有限公司

```
      "iam:securityPolicies:getProtectPolicy",
      "iam:securitypolicies:getTokenPolicyV5",
      "iam:users:get",
      "iam:users:getLoginProtectSetting",
      "iam:users:getUserV5",
      "iam:users:getVirtualMFADevice",
      "iam:users:showLoginProfileV5",
      "iam:users:showUserLastLoginV5",
      "agentnetwork:target:list",
      "lts:topics:*",
      "lts:groups:*",
      "sfsturbo:shares:*",
      "obs:*:*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "csms:secret:create"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "kms:cmk:createDataKey"
    ]
  }
 ]
}
```



步骤3 将自定义身份策略授权给用户组group1。



步骤4 如果沙箱需要挂载OBS存储，则需要创建一个信任委托，将OBS权限委托给“安全沙箱元数据”服务。您在创建沙箱模板时将其中“运行时信任委托”参数设置为该委托即可挂载OBS存储。本场景为了快速验证，授予OBSFullAccess权限。

1. 选择左侧导航栏中的“委托”，单击右上角的“创建信任委托”按钮。

 版权所有 © 华为云计算技术有限公司



2. 设置委托名称为agentsphere-delegation-test，信任主体类型选择云服务，云服务选择“安全沙箱元数据”，单击“完成”。



3. 找到agentsphere-delegation-test委托，单击“授权”，选择OBSFullAccess权限，单击“确定”。

步骤5 登录AgentSphere的页面进行后续的功能验证即可。

----结束

场景二：授予子用户可以操作 AgentSphere 资源的权限，依赖的云服务资源仅有查看权限

假设已新建用户组group1和子用户user1，且user1加入了group1。

步骤1 为group1授予AgentSphereFullAccessPolicy系统策略，该系统策略仅包含AgentSphere运行所需要的最小权限。

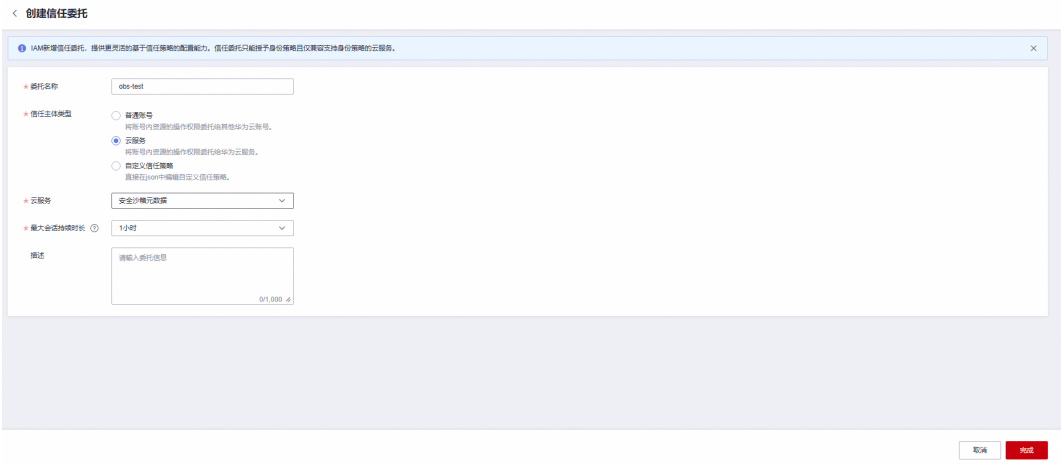
步骤2 如果需要为沙箱挂载OBS/SFSTurbo存储的话，则需要新建自定义策略agentsphere-extend添加相关权限后，授予group1，具体的策略内容如下：

```
{
  "Version": "5.0",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:agencies:getV5",
        "iam:agencies:listV5"
      ]
    },
    {
      "Effect": "Allow",
```

 版权所有 © 华为云计算技术有限公司

```
    "Action": [
      "sfsturbo:shares:getAllShares",
      "sfsturbo:shares:getShare"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "obs:bucket:listAllMyBuckets",
      "obs:bucket:listBucket",
      "obs:bucket:headBucket"
    ]
  }
 ]
}
```

步骤3 使用OBS和SFSTurbo挂载的时候，需要新建信任委托，委托名称为agentsphere-delegation-test，信任主体类型选择云服务，云服务选择“安全沙箱元数据”，单击“确认”即可。



步骤4 新建自定义信任策略agentsphere-delegation-policy，添加好OBS/SFSTurbo权限后，将自定义策略权限授予给agentsphere-delegation-test委托，自定义策略权限如下：

```
{
 "Version": "5.0",
 "Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sfsturbo:shares:getAllShares",
      "sfsturbo:shares:getShare"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "obs:bucket:listAllMyBuckets",
      "obs:bucket:listBucket",
      "obs:bucket:headBucket",
      "obs:object:*"
    ]
  }
 ]
}
```

----结束

 版权所有 © 华为云计算技术有限公司