

AircityCloudForLinuxVersion 安装文档

- [一、安装包准备](#)
- [二、初始化服务器](#)
 - [1. 创建 freedo 用户，并授予 sudo 权限](#)
 - [2. 【可选，如果网络配置正常请跳过此步骤】网卡自动启动](#)
 - [3. 处理系统防火墙](#)
 - [可选方法一：关闭防火墙服务](#)
 - [可选方法二：通过防火墙放行相关端口](#)
 - [4. 【可选】配置 tigervnc](#)
 - [1. 安装配置 vncserver 服务](#)
 - [2. 设置 freedo 用户的 vnc 密码](#)
 - [3. 重启系统，确认 vnc 服务正常启动](#)
 - [4. 使用 VNC Viewer 访问](#)
 - [5. 安装配置 nvidia 显卡驱动](#)
 - [1. 安装依赖包](#)
 - [2. 屏蔽自带驱动](#)
 - [3. 显卡支持](#)
 - [4. 重建 initramfs image](#)
 - [5. 重启](#)
 - [6. 停止 x windows 相关服务](#)
 - [7. 上传下载后的驱动文件到服务器](#)
 - [8. 为驱动文件添加执行权限](#)
 - [9. 运行文件安装驱动](#)
 - [10. 再次重启](#)
 - [11. 重启之后执行命令`nvidia-smi`确定显卡驱动状态](#)
- [三、上传安装包到服务器任意目录并解压](#)
 - [1. 创建部署目录](#)
 - [2. 上传安装包 AirCityCloud_x86_0615.zip 到/deploy](#)
 - [3. 解压安装包](#)
- [四、申请 License](#)
 - [1. 使用 PCIdentifier 工具生成机器码](#)
 - [2. 提供机器码给飞渡，申请授权文件](#)
 - [3. 拿到授权文件，重命名并复制到指定目录](#)
- [五、拷贝依赖库到指定目录](#)
 - [拷贝依赖库到指定目录](#)
 - [X86架构](#)
 - [ARM64架构](#)
- [六、执行脚本 startCloudServer.sh 启动集群管理服务 CloudServer](#)
 - [1. 修改配置文件 AirCityCloud/CloudServer.conf](#)
 - [2. 执行脚本，启动集群管理服务](#)
 - [3. 【可选】配置为systemd服务，并开机启动](#)
 - [3. 访问实例管理页面\[无渲染实例\]](#)
- [七、执行脚本 startNodeServcie.sh 启动单个渲染实例](#)
 - [1. 修改配置文件 AirCityCloud/NodeService.conf，使渲染节点加入到集群管理服务中](#)
 - [2. 执行脚本，启动渲染实例](#)
 - [3. 【可选】配置为systemd服务，并开机启动](#)
 - [4. 添加工程数据并访问](#)
- [八、【可选项】安装中继服务\[在需要跨网段访问的 cloud 服务的环境下，就需要部署中继 coturn 服务\]](#)
 - [1. 安装 docker 服务](#)
 - [2. 上传镜像到服务器](#)
 - [3. 挂载镜像](#)
 - [4. 编辑启动脚本，然后启动服务](#)
 - [5. 配置 AirCityCloud/Config/TurnServer.conf 配置文件，启用中继服务](#)
 - [6. 重启服务使中继服务配置生效](#)

- 九、【可选】配置Https访问
 - 1. 申请证书，推荐[腾讯云免费证书](https://console.cloud.tencent.com/ssl)
 - 2. 申请完成之后，下载nginx证书
 - 3. 上传下载的证书到服务器任意目录，并解压
 - 4. 配置AirCityCloud/Config/Https.conf配置文件，启用Https服务
 - 5. 重启服务使Https配置生效
 - 6. 使用https访问集群管理页面
 - 7. NodeService服务配置修改【所有节点服务配置都要修改】
 - 8. 重启NodeService服务
- 十、公网端口映射配置
 - 1. 在出口路由器上映射`8087/tcp`和`3478/udp`端口到公网
 - 2. 根据端口映射配置，修改AirCityCloud/Config/TurnServer.conf
 - 3. 重启CloudServer服务使端口映射配置生效
- 十一、【可选】KylinV10 系统配置 nfs
 - 1、环境准备
 - 2、启动 NFS 服务
 - 3、客户端挂载 NFS 共享目录，并配置开机自动挂载
- 十二、常见问题
 - 1. 实例无法启动
 - 2. 无法访问页面
 - 3. player 页面正常的情况下无法获取视频流
 - 问题描述： 卡在以下画面无法获取视频流
 - 排查步骤：
 - 4. 实例卡死、性能问题
 - 5. 中继服务无法访问问题

PDF版本文档	下载地址
0615[最新版]	点击查看下载
0614	点击查看下载
0606	点击查看下载
0529	点击查看下载
0511	点击查看下载

实测环境
系统： Kylin Linux Advanced Server V10 （Sword）(SP2)
架构： X86_64
用户： freedo[具有sudo权限]

一、安装包准备

系统架构	组件	安装包名称	下载地址	备注
x86_64	NVIDIA 显卡驱动	NVIDIA-Linux-x86_64-525.116.03.run	https://installpackages.gbim.vip/Nvidia-Drivers/ForLinux/NVIDIA-Linux-x86_64-525.116.03.run	
x86_64	中继服务 coturn 镜像	coturn-latest-amd64-20230510.tar.gz	https://installpackages.gbim.vip/docker_update/coturn-latest-amd64-20230510.tar.gz	
x86_64	AirCityCloud	AirCityCloud_x86_0615.zip	https://pan.baidu.com/s/1oF8NCCcLr5-Bq5bdnB6bwQ?pwd=ytjy	
ARM64	NVIDIA 显卡驱动	NVIDIA-Linux-aarch64-525.116.04.run	https://installpackages.gbim.vip/Nvidia-Drivers/ForLinux/ARM64/NVIDIA-Linux-aarch64-525.116.04.run	
ARM64	中继服务 coturn 镜像	coturn-latest-arm64-20230511.tar.gz	https://installpackages.gbim.vip/docker_update/coturn-latest-arm64-20230511.tar.gz	
ARM64	AirCityCloud	AirCityCloud_arm64_0615.zip	https://pan.baidu.com/s/1e1NeQ4hQTiPi6eIZxPJUJg?pwd=rdiw	

二、初始化服务器

1. 创建 freedo 用户，并授予 sudo 权限

```
1 useradd -u 9988 freedo #建立用户名为 freedo 的一般用户
2 passwd freedo #为用户 freedo 设置密码
3 #Changing password for user freedo.
4 #New UNIX password:      ← 输入密码（密码不会被显示）
5 #Retype new UNIX password:  ← 再次输入密码确认两次密码一致
6 #passwd: all authentication tokens updated successfully.  ← 密码设置成功
7
8 #添加freedo用户到wheel组
9 usermod -G wheel freedo
```

【以下步骤都登录到 freedo 用户进行操作】

2. 【可选，如果网络配置正常请跳过此步骤】网卡自动启动

Kylin V10 Server SP2 安装成功之后，默认网卡是禁用状态，需要配置网卡自动启用。

```
1 # 查看网卡信息
2 sudo ifconfig
3
4 # 根据获取的网卡信息，把以下命令中的ens192改为实际中的网卡名称
5 sudo nmcli con mod ens192 connection.autoconnect yes
```

3. 处理系统防火墙

选择下面的任意一种方法，处理系统自带 firewalld 防火墙

可选方法一：关闭防火墙服务

```
1 sudo systemctl stop firewalld
2 sudo systemctl disable firewalld
```

可选方法二：通过防火墙放行相关端口

```
1 # 允许vnc访问
2 sudo firewall-cmd --zone=public --add-port=5900-5902/tcp --permanent
3
4 # 允许Cloud页面访问,默认8087/tcp端口，根据实际情况修改
5 sudo firewall-cmd --zone=public --add-port=8087/tcp --permanent
6
7 # 渲染程序连接的端口
8 sudo firewall-cmd --zone=public --add-port=8088/tcp --permanent
9 # 实例管理服务使用的端口
10 sudo firewall-cmd --zone=public --add-port=8089/tcp --permanent
11
12
13 # 允许P2P视频流传输
14 sudo firewall-cmd --zone=public --add-port=50000-65535/udp --permanent
15
16
17 # 允许中继服务访问,默认3478/tcp 3478/udp端口，根据实际情况修改
18 sudo firewall-cmd --zone=public --add-port=3478/tcp --permanent
19 sudo firewall-cmd --zone=public --add-port=3478/udp --permanent
20
21 # 重载配置，然后查看配置状态
22 sudo firewall-cmd --reload
23 sudo firewall-cmd --list-port
```

```
[freedo@kylinV10Sp2Server-01 dep]$ sudo firewall-cmd --zone=public --add-port=8087/tcp --permanent
success
[freedo@kylinV10Sp2Server-01 dep]$ sudo firewall-cmd --zone=public --add-port=3478/tcp --permanent
success
[freedo@kylinV10Sp2Server-01 dep]$ sudo firewall-cmd --zone=public --add-port=3478/udp --permanent
success
[freedo@kylinV10Sp2Server-01 dep]$ sudo firewall-cmd --reload
success
[freedo@kylinV10Sp2Server-01 dep]$ sudo firewall-cmd --list-port
5900-5902/tcp 8087/tcp 3478/tcp 3478/udp
```

4. 【可选】配置 tigervnc

1. 安装配置 vncserver 服务

```
1 # 安装tigervnc服务端
2 sudo dnf makecache
3 sudo dnf install tigervnc-server tigervnc-server-module
4
5 # 配置为系统服务
6 sudo cp /lib/systemd/system/vncserver@.service /etc/systemd/system/
7 sudo sed -i 's+<USER>+freedo+g' /etc/systemd/system/vncserver@.service
8 sudo systemctl daemon-reload
9 sudo systemctl enable vncserver@:1.service
```

2. 设置 freedo 用户的 vnc 密码

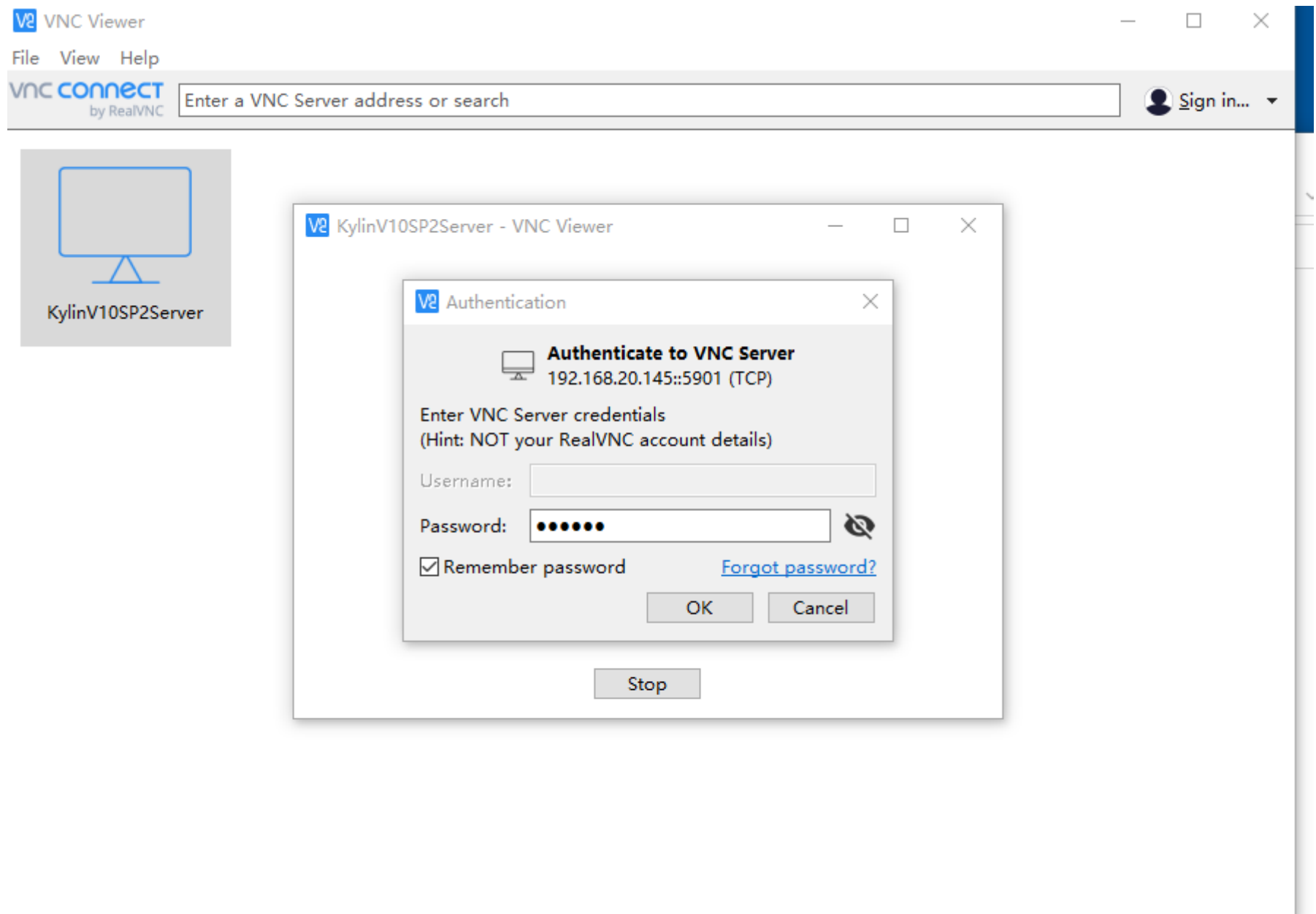
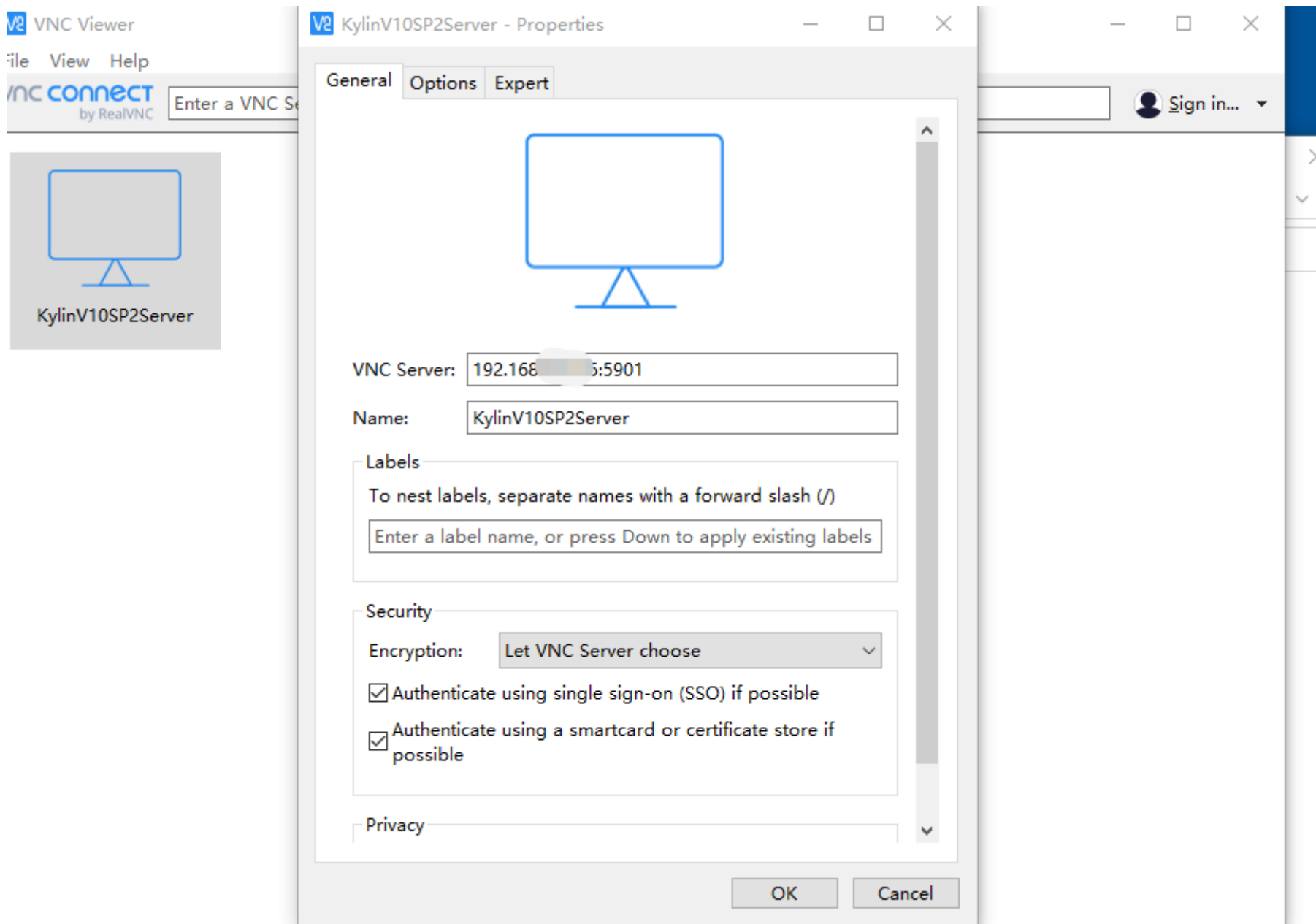
```
1 vncpasswd
2 #Password:
3 #Verify:
4 #Would you like to enter a view-only password (y/n)? n
```

3. 重启系统，确认 vnc 服务正常启动

```
[freedo@kylinV10Sp2Server-01 ~]$ sudo systemctl status vncserver@:1.service
● vncserver@:1.service - Remote desktop service (VNC)
   Loaded: loaded (/etc/systemd/system/vncserver@.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2023-05-08 10:18:05 CST; 9min ago
     Process: 1730 ExecStartPre=/bin/sh -c /usr/bin/vncserver -kill :1 > /dev/null 2>&1 || : (code=exited, status=0/SUCCESS)
     Process: 1845 ExecStart=/usr/bin/vncserver -autokill :1 (code=exited, status=0/SUCCESS)
    Main PID: 1855 (Xvnc)
      Tasks: 221
     Memory: 519.4M
    CGroup: /system.slice/system-vncserver.slice/vncserver@:1.service
            └─1855 /usr/bin/Xvnc :1 -auth /home/freedo/.Xauthority -desktop kylinV10Sp2Server-01:1 (freedo) -fp catalogue:/etc/X11/fontpath.d -geometry 1024x768 -pn -rfbauth /home/freedo/.vnc/passwd -rfbport 5900
            └─2261 sh -c (/home/freedo/.vnc/xstartup; /usr/bin/vncserver -kill :1) >> '/home/freedo/.vnc/kylinV10Sp2Server-01:1.log' 2>&1 &
            └─2262 /usr/bin/mate-session
            └─2265 dbus-launch --exit-with-session /usr/bin/mate-session
            └─2304 /usr/bin/dbus-daemon --syslog --fork --print-pid 6 --print-address 8 --session
            └─2310 /usr/libexec/dconf-service
            └─2314 gnome-keyring-daemon --start
            └─2318 /usr/libexec/mate-settings-daemon
            └─2323 marco
            └─2345 caja
            └─2349 mate-volume-control-applet
            └─2351 /usr/bin/python3 /usr/local/lib64/mate-indicators/libexec/reset_applet_position.py
            └─2355 nm-applet
            └─2361 ukui-screensaver-backend
            └─2363 /usr/libexec/gvfsd
            └─2373 /usr/libexec/gvfsd-fuse /home/freedo/.cache/gvfs -f -o big_writes
            └─2374 /usr/libexec/geoclue-2.0/demos/agent
            └─2497 /usr/libexec/imsettings-daemon
            └─2512 /usr/libexec/at-spi-bus-launcher
            └─2517 /usr/bin/dbus-daemon --config-file=/usr/share/defaults/at-spi2/accessibility.conf --nofork --print-address 3
            └─2525 /usr/libexec/gvfsd-trash --spawner :1.10 /org/gtk/gvfs/exec_spaw/0
            └─2532 /usr/bin/pulseaudio --start --log-target=syslog
            └─2571 /usr/libexec/gvfs-udisks2-volume-monitor
            └─2605 /usr/libexec/at-spi2-registryd --use-gnome-session
            └─2635 /usr/libexec/gvfs-goa-volume-monitor
            └─2654 /usr/libexec/goa-daemon
            └─2744 mate-panel
            └─2802 /usr/bin/fcitx -D
            └─2806 /usr/libexec/pulse/gsettings-helper
            └─2811 /usr/libexec/pulse/gconf-helper
```

4. 使用 VNC Viewer 访问

VNC Viewer 下载地址: <https://www.realvnc.com/en/connect/download/viewer/windows/> 



5. 安装配置 nvidia 显卡驱动

驱动下载地址：<https://www.nvidia.com/Download/Find.aspx?lang=en-us>

1. 安装依赖包

```
sudo dnf install elfutils-libelf-devel libglvnd libglvnd-devel vulkan-loader vulkan-loader-devel
```

2. 屏蔽自带驱动

```
1 | sudo touch /etc/modprobe.d/blacklist-nvidia-nouveau.conf
2 | sudo bash -c "cat > /etc/modprobe.d/blacklist-nvidia-nouveau.conf" << EOF
3 | blacklist nouveau
4 | options nouveau modeset=0
5 | EOF
```

3. 显卡支持

```
1 | sudo touch /etc/modprobe.d/nvidia.conf
2 | sudo bash -c "cat > /etc/modprobe.d/nvidia.conf" << EOF
3 | options nvidia NVreg_OpenRmEnableUnsupportedGpus=1
4 | EOF
```

4. 重建 initramfs image

```
1 | sudo mv /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).img.bak
2 | sudo dracut /boot/initramfs-$(uname -r).img $(uname -r)
```

5. 重启

```
sudo systemctl reboot
```

6. 停止 x windows 相关服务

```
sudo systemctl stop lightdm vncserver@\:1
```

7. 上传下载后的驱动文件到服务器

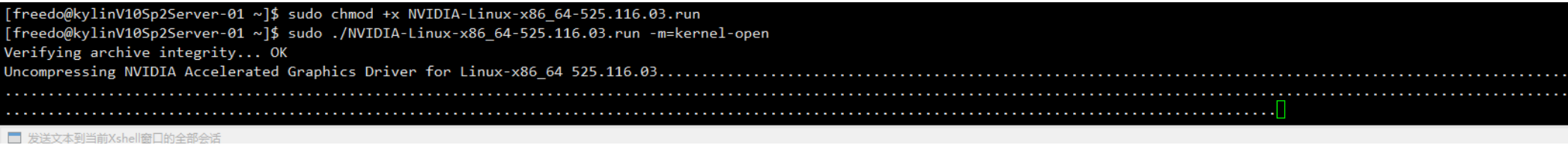
略

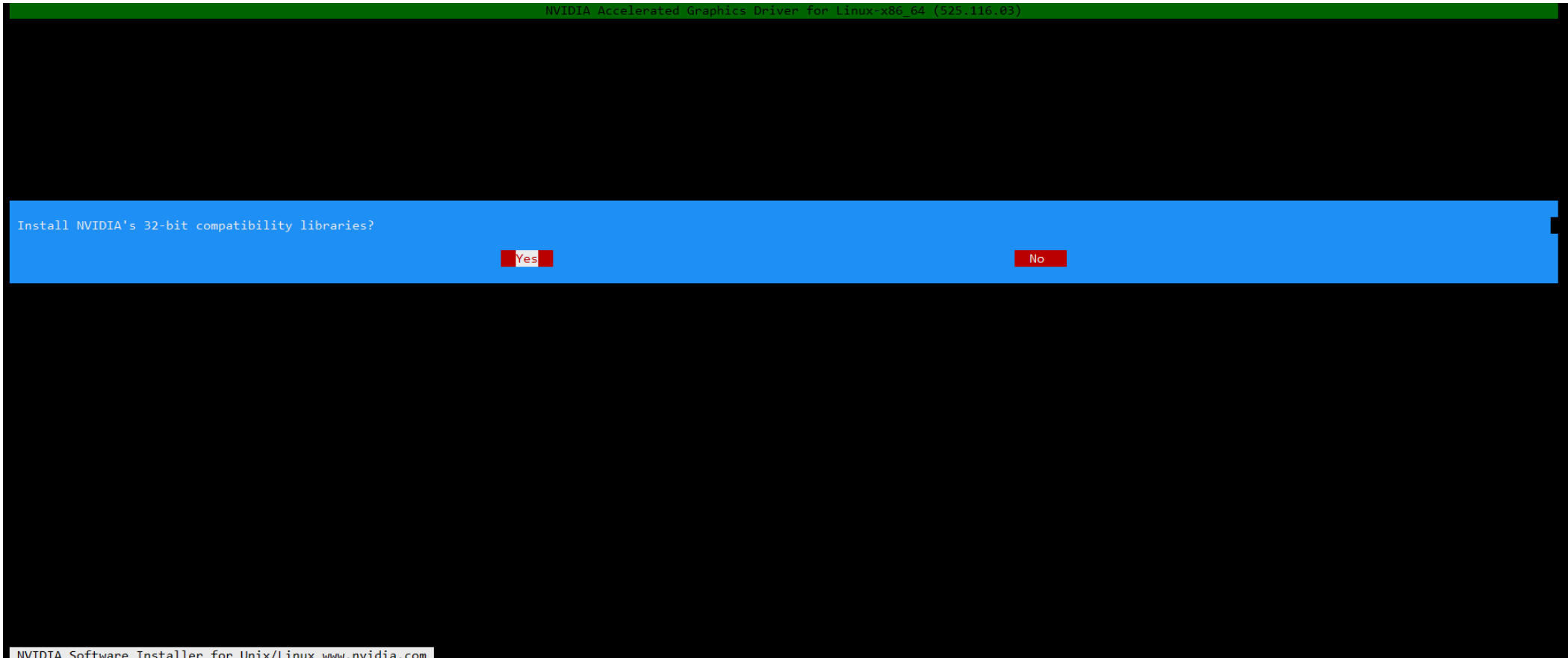
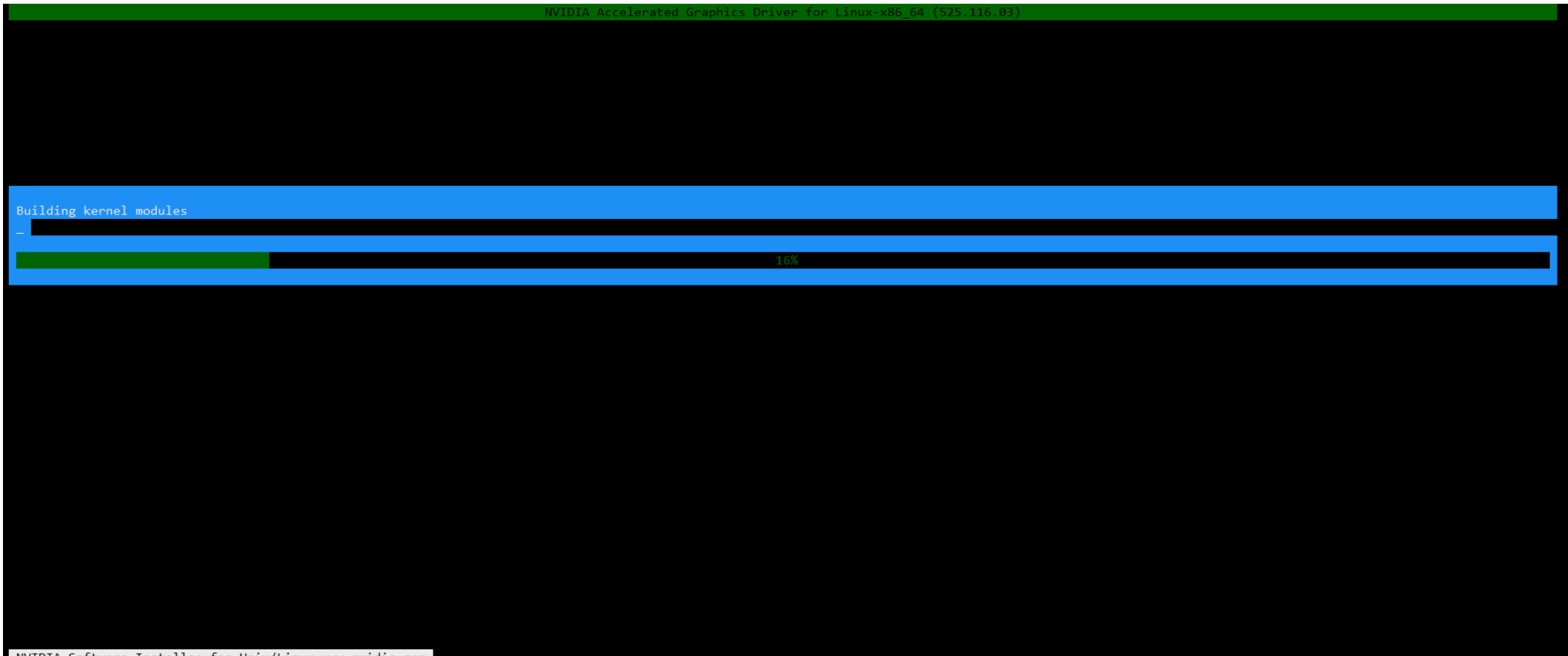
8. 为驱动文件添加执行权限

```
sudo chmod +x NVIDIA-Linux-x86_64-525.116.03.run
```

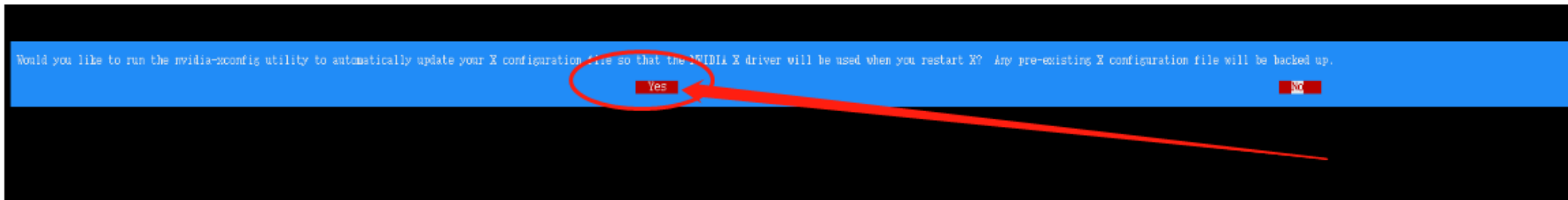
9. 运行文件安装驱动

```
sudo ./NVIDIA-Linux-x86_64-525.116.03.run -m=kernel-open
```





注意： 最后一步，提示是否自动配置 *X Window* 的时候，请选择"yes"，否则会导致 *vnc* 无法调用独立显卡，从而无法运行 *AirCityExplorer*



10. 再次重启

```
sudo systemctl reboot
```

11. 重启之后执行命令 `nvidia-smi` 确定显卡驱动状态

```
[freedo@kylinV10Sp2Server-01 ~]$ nvidia-smi
Mon May  8 10:18:56 2023

+-----+
| NVIDIA-SMI 525.116.03   Driver Version: 525.116.03   CUDA Version: 12.0   |
+-----+-----+-----+
| GPU   Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                                           MIG M.         |
+-----+-----+-----+
|  0 NVIDIA GeForce ...    Off   | 00000000:13:00.0 Off |             N/A     |
| 39%   44C   P8     22W / 350W |  5MiB / 24576MiB |      0%      Default |
|                                           N/A              |
+-----+-----+-----+

+-----+
| Processes: |
| GPU   GI    CI          PID    Type   Process name                      GPU Memory |
|          ID    ID                                   Usage          |
+-----+-----+
|  0   N/A   N/A         1833      G   /usr/libexec/Xorg                  4MiB      |
+-----+
```

三、上传安装包到服务器任意目录并解压

1. 创建部署目录

```
1 | sudo mkdir /deploy
2 | sudo chown -R freedo.freedo /deploy
```

2. 上传安装包 AirCityCloud_x86_0615.zip 到/deploy

3. 解压安装包

```
unzip AirCityCloud_x86_0615.zip
```

四、申请 License

1. 使用 PCIdentifier 工具生成机器码

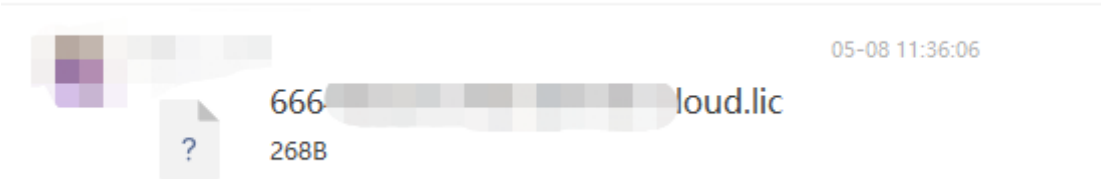
```
1 | cd /deploy/AirCityCloud/
2 | ./PCIdentifier
```

```
[freedo@kylinV10Sp2Server-01 AirCityCloud]$ ./PCIdentifier
机器码: 666[REDACTED]9B
```

2. 提供机器码给飞渡，申请授权文件



3. 拿到授权文件，重命名并复制到指定目录



```
1 | # 假如授权文件名称为68887736D6C4901B9F99B-cloud.lic，并且已上传到/deploy目录
2 |
3 | cd /deploy
```

```
4
5 # 重命名授权文件为License.lic
6 mv 68887736D6C4901B9F99B-cloud.lic License.lic
7
8 # 放置授权文件到指定目录
9 mv License.lic AirCityCloud/CloudServer/
```

五、拷贝依赖库到指定目录

拷贝依赖库到指定目录

X86架构

```
1 cd /deploy/
2 cd AirCityCloud/dep/dll/
3 cp lib* ../../CloudRendererer/AirCityExplorer/Binaries/Linux/
```

ARM64架构

```
1 cd /deploy/
2 cd AirCityCloud/dep/dll/
3 cp lib* ../../CloudRendererer/AirCityExplorer/Binaries/LinuxAArch64/
```

六、 执行脚本 [startCloudServer.sh](#) 启动集群管理服务 CloudServer

1. 修改配置文件 AirCityCloud/CloudServer.conf

1. 修改 serverIP 变量为本机 IP 地址

```
#!/usr/bin/env bash
# 集群管理服务配置文件

# 服务器IP地址
serverIP="192.168.20.145"
# 脚本所在目录，AircityCloud所在的目录
Aircity_Root=$PWD
```

2. 【可选】修改服务端口，默认 8087/tcp

```
# 网页与API访问端口
playerPort="8087"
```

2. 执行脚本， 启动集群管理服务

```
1 cd /deploy/
2 cd AirCityCloud/
3
4 ./startCloudServer.sh
```

```
[f...r-01 AirCityCloud]$ ./startCloud.sh
正在强制停止cloudWeb进程
正在强制停止cloudDataserer进程
-----等待web服务启动成功-----
-----5-----
-----4-----
-----3-----
-----2-----
-----1-----
-----0-----
-----CloudWeb服务已启动，可以继续-----
-----CloudDataserer服务已启动，可以继续-----
^_
^_ 服务启动成功！
^_ 请使用浏览器[推荐Chrome]打开网址： http://...87/samples/locale_zh/player.html 访问
```

3. 【可选】配置为systemd服务，并开机启动

```
1 cd /deploy/
2 cd AirCityCloud/
3
4 # 先停止脚本直接启动的服务
5 ./stopCloudServer.sh
6
7 # 执行脚本自动配置服务
8 cd dep/
9 ./makeService_CloudServer.sh
10
11 # 启动CloudServer服务
12 sudo systemctl start AirCityCloudServer.service
13
14 # 查看CloudServer服务状态
15 sudo systemctl status AirCityCloudServer.service
```

```
[freedo@freedo-01 dep]$ sudo systemctl start AirCityCloudServer.service
[freedo@freedo-01 dep]$ sudo systemctl status AirCityCloudServer.service
● AirCityCloudServer.service - Freedo AirCityCloud Service
   Loaded: loaded (/etc/systemd/system/AirCityCloudServer.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2023-05-25 15:17:31 CST; 4s ago
     Process: 297931 ExecStart=/home/freedo/DTS/AirCityCloudPro/startCloudServer.sh (code=exited, status=0/SUCCESS)
    Main PID: 297945 (CloudServer_lin)
       Tasks: 10
      Memory: 26.9M
     CGroup: /system.slice/AirCityCloudServer.service
            └─297945 /home/freedo/DTS/AirCityCloudPro/CloudServer/CloudServer_lin --httpRoot="/home/freedo/DTS/AirCityCloudPro/SDK" --playerPort=8087 --license="/home/freedo/DTS/AirCityCloudPro/CloudServer/License.lic"

5月 25 15:17:28 freedo-01 startCloudServer.sh[297931]: -----4-----
5月 25 15:17:28 freedo-01 startCloudServer.sh[297931]: -----3-----
5月 25 15:17:29 freedo-01 startCloudServer.sh[297931]: -----2-----
5月 25 15:17:30 freedo-01 startCloudServer.sh[297931]: -----1-----
5月 25 15:17:31 freedo-01 startCloudServer.sh[297931]: -----0-----
5月 25 15:17:31 freedo-01 startCloudServer.sh[297931]: -----CloudServer已启动，可以继续-----
5月 25 15:17:31 freedo-01 startCloudServer.sh[297931]: ^_^
5月 25 15:17:31 freedo-01 startCloudServer.sh[297931]: ^_^ 服务启动成功！
5月 25 15:17:31 freedo-01 startCloudServer.sh[297931]: ^_^ 请使用浏览器[推荐Chrome]打开网址： http://192.168.20.145:8087/samples/locale_zh/manage/ 管理节点
5月 25 15:17:31 freedo-01 systemd[1]: Started Freedo AirCityCloud Service.
```

3. 访问实例管理页面[无渲染实例]

← ↻ ⚠ 不安全 | 192.168.20.145:8087/samples/locale_zh/manage/ 🔍 ⭐ 🏠 🌐 📱 🖨

 **CloudMaster**

实例管理：（0个实例）

实例标识符	运行状态	运行时长	渲染节点	分辨率	帧率	模式	连接	次数	缩放	限制	显卡	锁定	工程	操作

测试页面：[视频流测试](#) [API示例](#) [实例管理接口](#) [实时运行状态](#) [样例集锦](#) [Hello World](#) [二三维坐标转换](#)

Initializing...
connecting (ws://192.168.20.145:8087/manager?category=NodeService) ...
WebSocket Connected!

Call GetStatus
{ "command": 1, "authorization": null }
Result: { "command": -1004, "data": { "enableLog": true, "logLevel": "Debug", "udpCustomPorts": true, "udpMinPort": 50000, "udpMaxPort": 65535, "pauseWhenIdle": false, "hideCrashDialogs": false, "resetWhenNotResponding": 5, "hideSidebar": false } }

七、 执行脚本 startNodeServcie.sh 启动单个渲染实例

子节点只需要启动本 NodeService 服务，不需要启动 CloudServer 以及中继服务

1. 修改配置文件 AirCityCloud/NodeService.conf，使渲染节点加入到集群管理服务中

- 1、 修改 serverIP 变量为集群管理服务的 IP 地址
- 2、 修改 serverPort 变量为集群管理服务的端口
 - 【默认8087端口；启用Https时，默认8089端口】

```
#!/usr/bin/env bash
# 渲染节点服务配置文件

# 服务器IP地址
serverIP="192.168.20.145"

# 服务器端口
serverPort="8087"

# 脚本所在目录，AircityCloud所在的目录
Aircity_Root=$PWD
```

2. 执行脚本，启动渲染实例

```
1 | cd /deploy/
2 | cd AirCityCloud/
3
4 | ./startNodeService.sh
```



实例管理：（4个实例）

实例标识符	运行状态	运行时长	渲染节点	分辨率	帧率	模式	连接	次数	缩放	限制	显卡	锁定	工程	操作
2511176939984	Running	00:25:59	192.168.20.145	1920x937	25	1	0	3	Y	Y	0	N	/home/freedom/DTS/DTSPProject/vtpkwjnvtpkwjn.acp	设置参数 取消锁定 启动 停止
2511176939996	Running	00:25:52	192.168.20.145	1920x1080	25	1	0	0	Y	Y	1	N	/home/freedom/DTS/DTSPProject/vtpkwjnvtpkwjn.acp	设置参数 取消锁定 启动 停止
2511176940008	Running	00:25:46	192.168.20.145	1920x1080	25	1	0	0	Y	Y	0	N	/home/freedom/DTS/DTSPProject/vtpkwjnvtpkwjn.acp	设置参数 取消锁定 启动 停止
2511176940020	Running	00:25:39	192.168.20.145	1920x1080	25	1	0	0	Y	Y	1	N	/home/freedom/DTS/DTSPProject/vtpkwjnvtpkwjn.acp	设置参数 取消锁定 启动 停止

测试页面：[视频流测试](#) [API示例](#) [实例管理接口](#) [实时运行状态](#) [样例集锦](#) [Hello World](#) [二三坐标转换](#)

3. 【可选】配置为systemd服务，并开机启动

```
1 | cd /deploy/
2 | cd AirCityCloud/
3
4 | # 先停止脚本直接启动的服务
5 | ./stopNodeService.sh
6
7 | # 执行脚本自动配置服务
8 | cd dep/
9 | ./makeService_NodeService.sh
10
11 | # 启动NodeService服务
12 | sudo systemctl start AirCityNodeService.service
13
14 | # 查看NodeService服务运行状态
15 | sudo systemctl status AirCityNodeService.service
```

```
[freedom@freedom ~]$ sudo systemctl status AirCityNodeService.service
● AirCityNodeService.service - Freedom AirCityCloud Service
   Loaded: loaded (/etc/systemd/system/AirCityNodeService.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2023-05-25 15:13:48 CST; 8s ago
     Process: 296738 ExecStart=/home/freedom/DTS/AirCityCloudPro/startNodeService.sh (code=exited, status=0/SUCCESS)
    Main PID: 296761 (mono)
      Tasks: 439
     Memory: 2.3G
    CGroup: /system.slice/AirCityNodeService.service
           └─296761 mono /home/freedom/DTS/AirCityCloudPro/CloudRenderer/RendererAgent/NodeService.exe --ip=192.168.20.145 --port=8089 --dataRoot="/home/freedom/DTS/AirCityCloudPro/CloudRenderer/RendererAgent/No
           └─296782 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/AirCityExplorer/Binaries/Linux/AirCityExplorer-Linux-Shipping -id=2512210311074 -projectpath=/home/freedom/DTS/AirCityCloudPro/SDK/media/projec
           └─296785 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/AirCityExplorer/Binaries/Linux/AirCityExplorer-Linux-Shipping -id=2512210311085 -projectpath=/home/freedom/DTS/AirCityCloudPro/SDK/media/projec
           └─296788 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/AirCityExplorer/Binaries/Linux/AirCityExplorer-Linux-Shipping -id=2512210311096 -projectpath=/home/freedom/DTS/AirCityCloudPro/SDK/media/projec
           └─296791 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/AirCityExplorer/Binaries/Linux/AirCityExplorer-Linux-Shipping -id=2512210311107 -projectpath=/home/freedom/DTS/AirCityCloudPro/SDK/media/projec
           └─296933 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=zygote --no-zygote-sandbox --no-sandbox --locales-dir-path=/home/freedom/DTS/AirCityCloudPro/CloudR
           └─296935 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=zygote --no-zygote-sandbox --no-sandbox --locales-dir-path=/home/freedom/DTS/AirCityCloudPro/CloudR
           └─296941 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=zygote --no-sandbox --locales-dir-path=/home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binar
           └─296942 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=zygote --no-sandbox --locales-dir-path=/home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binar
           └─296945 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=zygote --no-zygote-sandbox --no-sandbox --locales-dir-path=/home/freedom/DTS/AirCityCloudPro/CloudR
           └─296946 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=zygote --no-zygote-sandbox --no-sandbox --locales-dir-path=/home/freedom/DTS/AirCityCloudPro/CloudR
           └─296947 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=zygote --no-sandbox --locales-dir-path=/home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binar
           └─296948 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=zygote --no-sandbox --locales-dir-path=/home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binar
           └─296973 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=gpu-process --field-trial-handle=11900335550845050427,18213647869669702984,131072 --no-sandbox --o
           └─296974 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=gpu-process --field-trial-handle=1501349583957061839,5662202684715684194,131072 --no-sandbox --ozo
           └─296999 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=gpu-process --field-trial-handle=12972911339298957415,7373245993841430247,131072 --no-sandbox --oz
           └─297005 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=gpu-process --field-trial-handle=7149277571050211753,4606713364906240720,131072 --no-sandbox --ozo
           └─297200 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=utility --utility-sub-type=network.mojom.NetworkService --field-trial-handle=11900335550845050427,
           └─297201 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=utility --utility-sub-type=network.mojom.NetworkService --field-trial-handle=1501349583957061839,5
           └─297230 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=utility --utility-sub-type=network.mojom.NetworkService --field-trial-handle=7149277571050211753,4
           └─297231 /home/freedom/DTS/AirCityCloudPro/CloudRenderer/Engine/Binaries/Linux/CloudWebHelper --type=utility --utility-sub-type=network.mojom.NetworkService --field-trial-handle=12972911339298957415,

5月 25 15:13:48 Server-01 systemd[1]: Starting Freedom AirCityCloud Service...
5月 25 15:13:48 Server-01 sudo[296739]: freedo : TTY=unknown ; PWD=/home/freedom/DTS/AirCityCloudPro ; USER=root ; COMMAND=/usr/bin/chmod -R a+x ./CloudRenderer/RendererAgent/BatchFiles
5月 25 15:13:48 Server-01 startNodeService.sh[296738]: -----Mono已安装-----
5月 25 15:13:48 kylli Server-01 sudo[296755]: freedo : TTY=unknown ; PWD=/home/freedom/DTS/AirCityCloudPro ; USER=root ; COMMAND=/usr/bin/ps -ef
5月 25 15:13:48 Server-01 startNodeService.sh[296738]: 开始启动NodeService.
5月 25 15:13:48 Server-01 sudo[296762]: freedo : TTY=unknown ; PWD=/home/freedom/DTS/AirCityCloudPro ; USER=root ; COMMAND=/usr/bin/ps -ef
5月 25 15:13:48 Server-01 startNodeService.sh[296738]: -----NodeService已启动-----
5月 25 15:13:48 Server-01 systemd[1]: Started Freedom AirCityCloud Service.

freedom@freedom ~$
```

4. 添加工程数据并访问

- 1. 上传工程数据到服务器上
- 2. 点击"设置参数", 打开实例设置界面

CloudMaster

实例管理：（2个实例）

实例标识符	运行状态	运行时长	渲染节点	分辨率	帧率	模式	连接	次数	缩放	限制	显卡	锁定	工程	操作
 2511308762465	Stopped	00:00:00	192.168.20.139	1920x1080	25	1	0	0	Y	Y	-1	N		设置参数 取消锁定 启动 停止
 2511308762477	Stopped	00:00:00	192.168.20.139	1920x1080	25	1	0	0	Y	Y	-1	N		设置参数 取消锁定 启动 停止

测试页面（在实例列表选中实例，可访问指定的实例，如果没有选中实例，则自动分配）：

[视频流测试](#) [API示例](#) [实例管理接口](#) [实时运行状态](#) [样例集锦](#) [Hello World](#) [二三维坐标转换](#)

Initializing...
connecting (ws://192.168.20.139:8087/manager?category=NodeService) ...
WebSocket Connected!

Call GetStatus
{ "command":1,"authorization":null}
Result: { "command":-1004,"data":{"enableLog":true,"logLevel":"Debug","udpCustomPorts":true,"udpMinPort":50000,"udpMaxPort":65535,"pauseWhenIdle":false,"hideCrashDialogs":false,"resetWhenNotResponding":5,"hideSidebar":false}}

实例参数设置

实例ID: 2511308762465

工程:

☐ 锁定工程

分辨率: 1920 X 1080 [HD](#) [FHD](#) [2K](#) [4K](#)

☒ 三维自适应客户端分辨率 ☐ 限制最大分辨率

视频模式: ☐ 高画质 ☒ 均衡 ☐ 流畅

高级参数

确定

取消

- 3. 复制粘贴之前上传的 acp 工程的绝对路径到工程空白框中

CloudMaster

实例管理：（2个实例）

实例标识符	运行状态	运行时长	渲染节点	分辨率	帧率	模式	连接	次数	缩放	限制	显卡	锁定	工程	操作
 2511308762465	Stopped	00:00:00	192.168.20.139	1920x1080	25	1	0	0	Y	Y	-1	N		设置参数 取消锁定 启动 停止
 2511308762477	Stopped	00:00:00	192.168.20.139	1920x1080	25	1	0	0	Y	Y	-1	N		设置参数 取消锁定 启动 停止

测试页面（在实例列表选中实例，可访问指定的实例，如果没有选中实例，则自动分配）：

[视频流测试](#) [API示例](#) [实例管理接口](#) [实时运行状态](#) [样例集锦](#) [Hello World](#) [二三维坐标转换](#)

Initializing...
connecting (ws://192.168.20.139:8087/manager?category=NodeService) ...
WebSocket Connected!

Call GetStatus
{ "command":1,"authorization":null}
Result: { "command":-1004,"data":{"enableLog":true,"logLevel":"Debug","udpCustomPorts":true,"udpMinPort":50000,"udpMaxPort":65535,"pauseWhenIdle":false,"hideCrashDialogs":false,"resetWhenNotResponding":5,"hideSidebar":false}}

实例参数设置

实例ID: 2511308762465

工程: /deploy/DemoData/vtpkwjn/vtpkwjn.acp

☒ 锁定工程

分辨率: 1920 X 1080 [HD](#) [FHD](#) [2K](#) [4K](#)

☒ 三维自适应客户端分辨率 ☐ 限制最大分辨率

视频模式: ☐ 高画质 ☒ 均衡 ☐ 流畅

高级参数

确定

取消

- 4. 点击确定，然后本实例就会自动启动

CloudMaster

实例管理：（2个实例）

实例标识符	运行状态	运行时长	渲染节点	分辨率	帧率	模式	连接	次数	缩放	限制	显卡	锁定	工程	操作
 2511308762465	Running	00:00:22	192.168.20.139	1920x1080	25	1	0	0	Y	Y	-1	Y	/deploy/DemoData/vtpkwjn/vtpkwjn.acp	设置参数 取消锁定 启动 停止
 2511308762477	Stopped	00:00:00	192.168.20.139	1920x1080	25	1	0	0	Y	Y	-1	N		设置参数 取消锁定 启动 停止

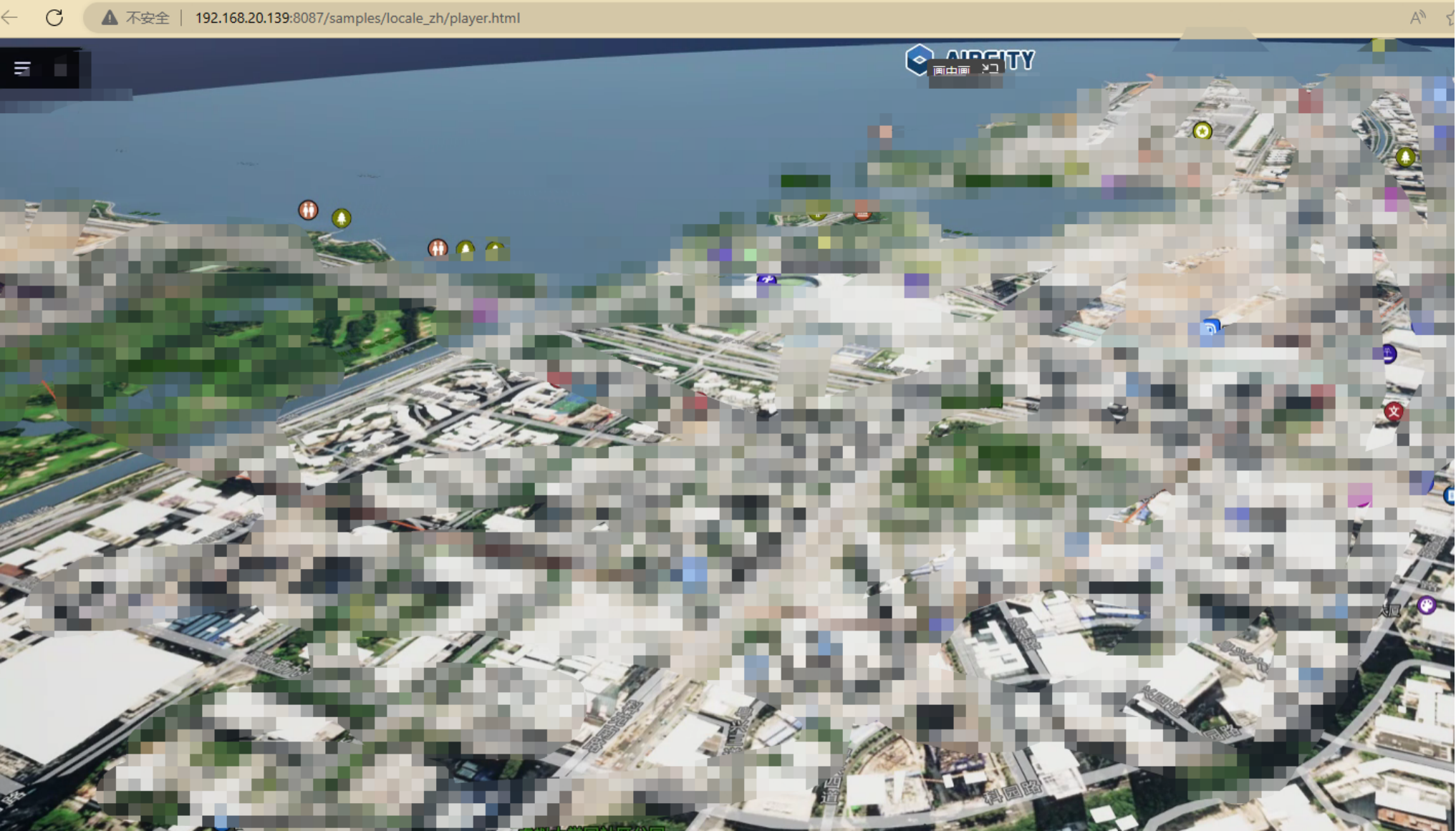
测试页面（在实例列表选中实例，可访问指定的实例，如果没有选中实例，则自动分配）：

[视频流测试](#) [API示例](#) [实例管理接口](#) [实时运行状态](#) [样例集锦](#) [Hello World](#) [二三维坐标转换](#)

Initializing...
connecting (ws://192.168.20.139:8087/manager?category=NodeService) ...
WebSocket Connected!

Call GetStatus
{ "command":1,"authorization":null}
Result: { "command":-1004,"data":{"enableLog":true,"logLevel":"Debug","udpCustomPorts":true,"udpMinPort":50000,"udpMaxPort":65535,"pauseWhenIdle":false,"hideCrashDialogs":false,"resetWhenNotResponding":5,"hideSidebar":false}}
Call SetInstanceParams
{ "async":false,"staticInstance":{"videoMode":1,"encodeRateControl":"VBR","multipass":0,"encodeMaxQP":35,"keyframeInterval":250,"maxBitrate":35,"id":"2511308762465","resX":1920,"resY":1080,"project":"/deploy/DemoData/vtpkwjn/vtpkwjn.acp","adjustResolution":true,"limitMaxResolution":false,"locked":true},"command":104,"authorization":null}
Instance is starting, please wait...
Result: { "command":104,"result":0} 操作成功
{ "command":104,"result":0}

5. 点击"视频流测试", 然后就可以跳转到视频流页面



八、【可选项】安装中继服务[在需要跨网段访问的 cloud 服务的环境下，就需要部署中继 coturn 服务]

coturn 是一个基于 TURN（Traversal Using Relay NAT）协议实现的用于在 NAT 网络下进行实时音视频通信的服务器软件。在网络层中，NAT 路由器会将源 IP 地址重写为自己的公网 IP 地址，而 TURN 服务器则可以通过中继数据包的方式，将经过 NAT 路由器的音视频数据转发到目标客户端，实现穿越 NAT 的效果，从而保证实时音视频通信的顺畅和稳定。

coturn提供了一个开源的免费实现，支持以下特性：

- 1. IPv4和IPv6网络的支持。
- 2. 高可用性和负载均衡的支持。
- 3. 证书验证和加密传输的支持。
- 4. 远程管理和监控的支持。
- 5. WebRTC（Web Real-Time Communications）的支持。
- 6. 简单可扩展的支持。

coturn适用于WebRTC、VoIP等网络通信应用，也可以作为一个代理服务器提供中间服务，增强网络连接稳定性与可靠性。

1. 安装 docker 服务

```
1  sudo dnf install docker
2  sudo systemctl enable docker
3  sudo systemctl start docker
4
5  # 使freedo用户可运行docker
6  sudo usermod -aG docker freedo
7  newgrp docker
8
9  # 设置内核支持ipv4转发
10 sudo sed -i "s+net.ipv4.ip_forward=0+net.ipv4.ip_forward=1+g" /etc/sysctl.conf
11 sudo sysctl -p
```

```
[root@AirCityCloud ~]# sudo dnf install docker
上次元数据过期检查: 1:32:11 前, 执行于 2023年05月09日 星期二 07时18分30秒。
依赖关系解决。
=====
Package                                Architecture          Version               Repository            Size
=====
安装:
docker-engine                          x86_64                18.09.0-202.ky10     ks10-sp2-iso          51 M

事务概要
=====
安装      1 软件包

总下载: 51 M
安装大小: 216 M
确定吗? [y/N]: y
下载软件包:
docker-engine-18.09.0-202.ky10.x86_64.rpm                                68 MB/s | 51 MB    00:00
-----
总计
运行事务检查
事务检查成功。
运行事务测试
事务测试成功。
运行事务
 准备中 :
 安装    : docker-engine-18.09.0-202.ky10.x86_64                1/1
 运行脚本: docker-engine-18.09.0-202.ky10.x86_64                1/1
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.

/sbin/ldconfig: /usr/lib64/libLLVM-7.so 不是符号链接

 验证    : docker-engine-18.09.0-202.ky10.x86_64                1/1

已安装:
docker-engine-18.09.0-202.ky10.x86_64

完毕!
```

2. 上传镜像到服务器

x86 架构镜像：https://installpackages.gbim.vip/docker_update/coturn-latest-amd64-20230510.tar.gz
aarch64 架构镜像：https://installpackages.gbim.vip/docker_update/coturn-latest-arm64-20230511.tar.gz

3. 挂载镜像

```
1 tar zxvf coturn-latest-amd64-20230510.tar.gz
2 docker load < coturn-latest-amd64-20230510.tar
```

```
[root@freedom01 ~]# cd /deploy
[root@freedom01 ~]# ls
coturn-latest-20230213.tar.gz
[root@freedom01 ~]# tar zxvf coturn-latest-20230213.tar.gz
coturn-latest-20230213.tar
[root@freedom01 ~]# sudo docker load < coturn-latest-20230213.tar
4695cdfb426a: Loading layer [=====>] 84MB/84MB
46e2019b70bf: Loading layer [=====>] 49.25MB/49.25MB
6b66680faf51: Loading layer [=====>] 4.193MB/4.193MB
fe03f45141bf: Loading layer [=====>] 1.591MB/1.591MB
Loaded image: coturn/coturn:latest
```

4. 编辑启动脚本，然后启动服务

```
默认使用3478/tcp 和 3478/udp 端口
如果要修改中继服务为其他端口，比如9999，替换下面脚本中的
-p 3478:3478 \
-p 3478:3478/udp \
为
-p 9999:3478 \
-p 9999:3478/udp \
```

```
1 cd /deploy
2 mkdir coturn
3 cat > start_coturn.sh << "EOF"
4 docker run -d \
5 --name freedo-coturn-01 \
6 --restart=always \
7 -e DETECT_EXTERNAL_IP=yes \
8 -e DETECT_RELAY_IP=yes \
9 -p 3478:3478 \
10 -p 3478:3478/udp \
11 -p 5349:5349 \
12 -p 5349:5349/udp \
13 -p 48100-48200:48100-48200/udp \
14 coturn/coturn:latest-amd64 \
15 -n --log-file=stdout \
16 --min-port=48100 \
17 --max-port=48200 \
18 --user=freedo:freedo \
```

```
19  --realm=feidu \  
20  --lt-cred-mech \  
21  EOF  
22  
23  # 授予脚本执行权限  
24  chmod +x start_coturn.sh  
25  
26  # 启动脚本  
27  ./start_coturn.sh  
28  
29  # 查看服务是否正常  
30  sudo docker ps
```

若架构为ARM64,修改start_coturn.sh中的 coturn/coturn:latest-amd64 为 coturn/coturn:latest-arm64 ,然后执行命令 docker rm -f freedo-coturn-01 , 重新执行脚本 ./start_coturn.sh

```
[freedom@11-M005-00-Server-01 coturn]$ cat >> start_coturn.sh << EOF  
> sudo docker run -d \  
> --name freedo-coturn-01 \  
> --restart=always \  
> -p 3478:3478 \  
> -p 3478:3478/udp \  
> -p 5349:5349 \  
> -p 5349:5349/udp \  
> -p 48100-48200:48100-48200/udp \  
> coturn/coturn:latest \  
> -n --log-file=stdout \  
> --min-port=48100 \  
> --max-port=48200  
> EOF  
[freedom@11-M005-00-Server-01 coturn]$ chmod +x start_coturn.sh  
[freedom@11-M005-00-Server-01 coturn]$ ./start_coturn.sh  
7f6cb19f68481951234a015d68d53689bd59c07e13e50b58a6fc5c3767a9e4de  
[freedom@11-M005-00-Server-01 coturn]$ sudo docker ps  
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS  
7f6cb19f6848        coturn/coturn:latest  "docker-entrypoint.s..."  9 seconds ago       Up 6 seconds       0.0.0.0:3478->3478/tcp, 0.0.0.0:3478->3478/udp, 0.0.0.0:5349->5349/udp, 0.0.0.0:5349->5349/tcp, 0.0.0.0:48100-48200->48100-48200/udp  
freedo-coturn-01
```

5. 配置 AirCityCloud/Config/TurnServer.conf 配置文件，启用中继服务

```
#!/usr/bin/env bash  
# 中继服务配置文件  
  
# 是否使用中继服务连接配置，默认不使用  
useTurnServer="yes"  
  
# 中继服务访问端口  
turnServerPort="3478"  
  
# 当中继服务无法使用udp端口时，使用tcp端口，默认使用udp  
useTCP="no"  
  
# 中继服务连接配置  
stunServer="stun:$serverIP:$turnServerPort"  
if [ "$useTCP" = "yes" ]; then  
    turnProtocol="tcp"  
    turnServer="turn:$serverIP:$turnServerPort?transport=$turnProtocol"  
else  
    turnServer="turn:$serverIP:$turnServerPort"  
fi  
turnUsername="freedo"  
turnPassword="freedo"  
  
# 视频流中继方式  
iceTransportPolicy="relay"
```

- 1. 修改turnServerPort变量为上一步启动的中继服务的端口，默认3478
- 2. 修改useTurnServer变量，yes表示启用中继服务，no表示不启用
- 3. 修改useTCP变量，默认使用udp协议进行数据转发,yes表示使用TCP协议

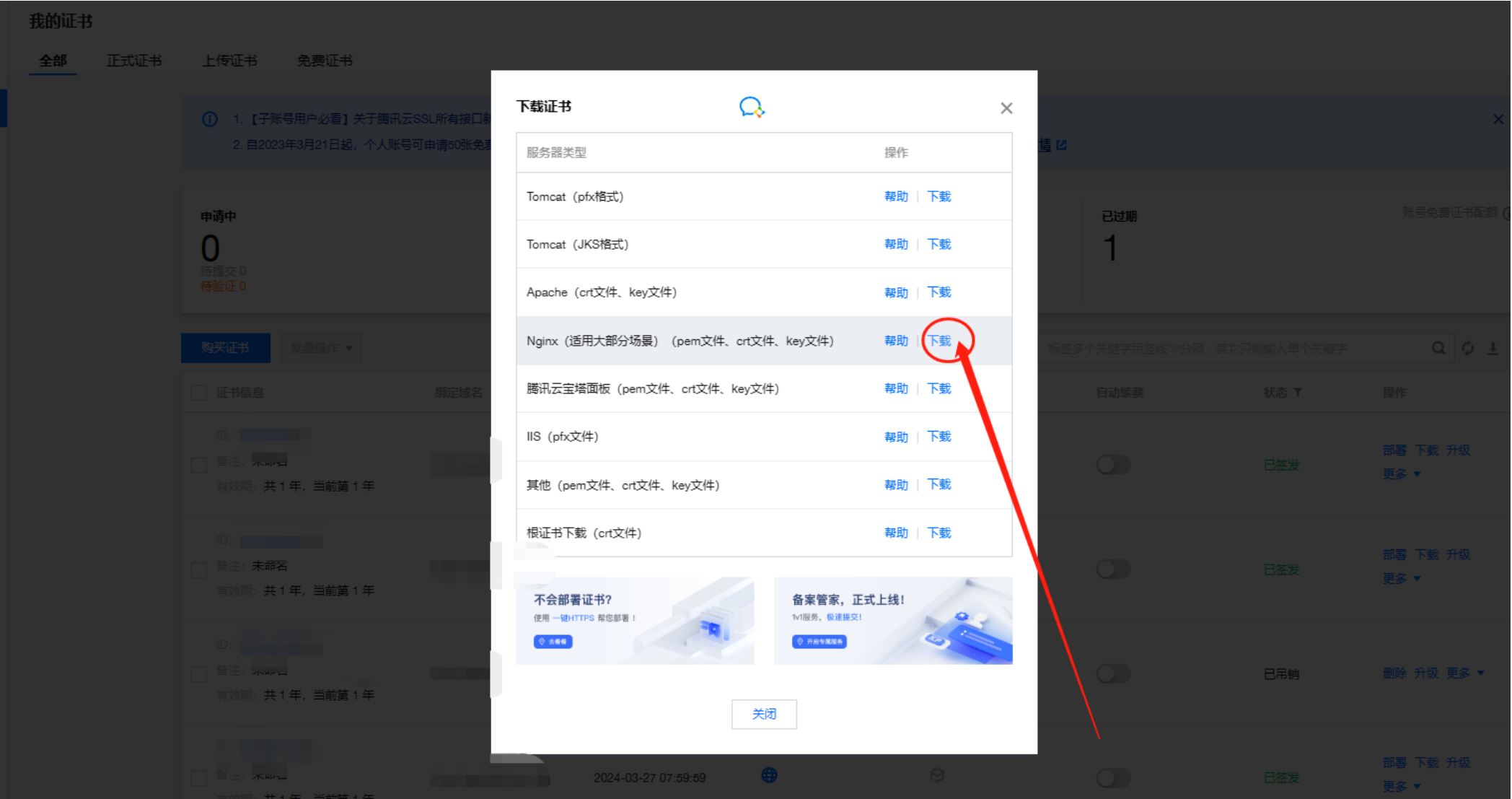
6. 重启服务使中继服务配置生效

```
sudo systemctl restart AirCityCloudServer.service
```

九、【可选】配置Https访问

1. 申请证书，推荐[腾讯云免费证书](#)

2. 申请完成之后，下载nginx证书



3. 上传下载的证书到服务器任意目录，并解压

```
[free@freedom ~]$ tree -L 2 .
.
├── b[redacted]_nginx
│   ├── b[redacted].com_bundle.crt
│   ├── b[redacted].com_bundle.pem
│   ├── b[redacted].com.csr
│   └── b[redacted].com.key
└── b[redacted].com_nginx.zip

1 directory, 5 files
```

4. 配置AirCityCloud/Config/Https.conf配置文件，启用Https服务

```
#!/usr/bin/env bash
# HTTPS配置文件

# 是否使用HTTPS访问，默认不使用，如果使用HTTPS访问请对此配置中的各项指定正确的参数
useHttps="yes"

# 指定渲染程序连接的端口
streamerPort="8088"
# 指定实例管理服务使用的端口
managerPort="8089"

#证书，如果提供了有效的证书参数，则提供https服务以代替http
ssl_cert="/home/freedom/DTS/AirCityCloudPro/Config/cert/b[redacted].com_nginx/[redacted].com_bundle.pem"

#私钥文件
ssl_key="/home/freedom/DTS/AirCityCloudPro/Config/cert/[redacted].com_nginx/[redacted].com.key"

#证书密码（可选，创建证书时设置了密码则此处就要设置对应密码）
ssl_passphrase=""
```

5. 重启服务使Https配置生效

```
sudo systemctl restart AirCityCloudServer.service
```

6. 使用https访问集群管理页面



你的连接不是专用连接

攻击者可能试图从 **192.168.20.145** 窃取你的信息(例如, 密码、消息或信用卡)。

NET::ERR_CERT_COMMON_NAME_INVALID

隐藏高级

[返回](#)

此服务器无法证明它是 192.168.20.145；它的安全证书来自 [REDACTED].com。
这可能是由错误配置或者有攻击者截获你的连接而导致的。

[继续访问 192.168.20.145 \(不安全\)](#)



渲染节点

ID	主机名称	IP地址	Nvidia显卡数	支持的实例数	已启动的实例数
 N2511308762192	kylinV10SP1Server-01	192.168.20.139	1	2	1
 N2512210310871	kylinV10Sp2Server-01	192.168.20.145	2	4	1

实例管理

实例标识符	运行状态	运行时长	渲染节点	分辨率	帧率	模式	连接	次数	缩放	限制	显卡	锁定	工程	操作
 2511308762477	Running	00:02:32	192.168.20.139	1207x543	40	均衡	1	1	Y	N	-1	Y	/deploy/DemoData/demo60/demo60.acp	设置参数 动 取消锁定 停止 启
 2511308762465	Stopped	00:00:00	192.168.20.139	2560x1440	39	均衡	0	0	Y	N	-1	N	/deploy/DemoData/vtpkwjn/vtpkwjn.acp	设置参数 动 取消锁定 停止 启
 2512210311074	Stopped	00:00:00	192.168.20.145	1920x1080	25	均衡	0	0	Y	N	0	Y	/home/freedom/DTS/AirCityCloudPro/SDK/media/project/demo.acp	设置参数 动 取消锁定 停止 启
 2512210311085	Stopped	00:00:00	192.168.20.145	1920x1080	25	均衡	0	0	Y	N	1	Y	/home/freedom/DTS/DTSProject/qd-acp/yj_test.acp	设置参数 动 取消锁定 停止 启
 2512210311096	Stopped	00:00:00	192.168.20.145	1920x1080	25	均衡	0	0	Y	N	0	Y	/home/freedom/DTS/AirCityCloudPro/SDK/media/project/demo.acp	设置参数 动 取消锁定 停止 启
 2512210311107	Running	00:21:15	192.168.20.145	1207x543	25	均衡	0	2	Y	N	1	Y	/home/freedom/DTS/AirCityCloudPro/SDK/media/project/demo.acp	设置参数 动 取消锁定 停止 启

测试页面（在实例列表中选中实例，可访问指定的实例，如果没有选中实例，则自动分配）：

[视频流测试](#) [API示例](#) [实例管理接口](#) [实时运行状态](#) [样例集锦](#) [Hello World](#) [二三维坐标转换](#)

```
Initializing...
connecting (wss://192.168.20.145:8087/manager?category=CloudMaster) ...
WebSocket Connected!
```

Call `GetStatus`

```
{"command":1,"authorization":null}
```

7. NodeService服务配置修改【所有节点服务配置都要修改】

当启用Https的时候，实例管理服务的端口为8089端口，所以需要修改配置文件中的端口

```
#!/usr/bin/env bash
# 渲染节点服务配置文件

# CloudServer服务器IP地址
serverIP="192.168.20.145"

# 服务器端口
managerPort="8089"

# 脚本所在目录，AircityCloud所在的目录
Aircity_Root=$PWD

# dataRoot路径
NodeServiceDataRoot="$Aircity_Root/CloudRenderer/RendererAgent/NodeServiceDataRoot"

# 是否后台运行，默认后台运行，只有运行出错的情况下才关闭后台运行便于观察错误
BackgroundProcess="yes"
```

8. 重启NodeService服务

```
sudo systemctl restart AirCityNodeService.service
```

十、 公网端口映射配置

1. 在出口路由器上映射 8087/tcp 和 3478/udp 端口到公网

8087/tcp 为 player页面访问端口，保证页面和接口的正常访问
3478/udp 为 coturn中继服务访问端口，保证视频流正常传输

本文档测试环境映射配置

内网IP	内网端口	映射IP	映射端口
192.168.20.145	8087/tcp	43.227.255.154	20535/tcp
192.168.20.145	3478/udp	43.227.255.154	13489/udp

2. 根据端口映射配置，修改AirCityCloud/Config/TurnServer.conf

```
useTCP="no"

# 中继服务连接配置
stunServer="stun:$serverIP:$turnServerPort"
if [ "$useTCP" = "yes" ]; then
    turnProtocol="tcp"
    turnServer="turn:$serverIP:$turnServerPort?transport=$turnProtocol"
else
    turnServer="turn:$serverIP:$turnServerPort"
fi
turnUsername="freedo"
turnPassword="freedo"

# 视频流中继方式
iceTransportPolicy="relay"

# 域名和ip映射
# 只有在开启中继服务之后，在需要的情况下才设置useMappingServer
useMappingServer="yes"

# 指定端口映射服务器的IP地址或域名
mappingServerIP="43.227.255.154"

# 指定中继服务的映射端口
portTurnMapping="13489"

# playerPort的映射端口
playerPortMapping="20535"

changeConf_MappingServer(){
    # $1 传进来的ac_conf.js的路径
    if [ "$useTurnServer" = "yes" -a "$useMappingServer" = "yes" -a -n "$mappingServerIP" ]; then
        sed -i "s+\"Path\"+\"PlayerMapping\":\ \ \"$mappingServerIP:$playerPortMapping\",\\n\\ \ \"Path\"+g" $1
    fi
}
```

- 1. 修改 useMappingServer 为 yes
- 2. 修改 mappingServerIP 为映射后的IP
- 3. 修改 portTurnMapping 为 3478/udp 映射后的端口
- 4. 修改 playerPortMapping 为 8087/tcp 映射后的端口

3. 重启CloudServer服务使端口映射配置生效

```
sudo systemctl restart AirCityCloudServer.service
```

十一、【可选】KylinV10 系统配置 nfs

本文档中的用户的 uid 为 9988 gid 为 9988
共享目录和挂载目录均为 /deploy/shareData
KylinV10Server 默认已安装 nfs-utils 和 rpcbind

以上参数根据系统实际情况进行灵活修改

1、环境准备

IP	功能
192.168.20.145	NFS 服务端
192.168.20.139	NFS 客户端

2、启动 NFS 服务

- 1. 配置防火墙【若 firewalld 已禁用，跳过本步骤】

```
1 | sudo firewall-cmd --permanent --add-service=nfs
2 | sudo firewall-cmd --permanent --add-service=mountd
3 | sudo firewall-cmd --permanent --add-service=rpc-bind
4 | sudo firewall-cmd --reload
```

2. 创建共享目录并授权

```
1 | mkdir /deploy/shareData
2 | chmod 777 /deploy/shareData
```

3. 修改 nfs 配置文件

```
1 | # anonuid=9988,anongid=9988 若客户端用户的uid/gid有变化，请自行修改
2 | sudo bash -c "cat >> /etc/exports" << "JIESHU"
3 | /deploy/shareData 192.168.20.0/24(rw,sync,insecure,no_subtree_check,no_root_squash,anonuid=9988,anongid=9988)
4 | JIESHU
```

4. 启动 nfs 服务，并配置开机启动

```
1 | sudo systemctl start nfs.service
2 | sudo systemctl enable nfs.service
```

3、客户端挂载 NFS 共享目录，并配置开机自动挂载

1. 确认可正常访问共享

```
showmount -e 192.168.20.145
```

2. 配置开机自动挂载

```
1 | mkdir /deploy/shareData
2 |
3 | sudo bash -c "cat >> /etc/fstab" << "JIESHU"
4 | 192.168.20.145:/deploy/shareData /deploy/shareData nfs4 defaults 0 0
5 | JIESHU
```

3. 挂载目录

```
1 | sudo mount -a
2 | sudo df -hT
```

4. 测试

```
1 | # 服务端
2 | cd /deploy/shareData
3 | touch ThisIsServer
4 |
5 | # 客户端
6 | cd /deploy/shareData
7 | touch ThisIsClient
```

十二、常见问题

1. 实例无法启动

- 1. 检查是否已申请授权，并重命名为 License.lic，并放置到指定目录
- 2. 检查是否使用拥有 sudo 权限的普通用户运行 CloudServer 和 NodeService
- 3. 检查是否安装 vulkan 相关 rpm 包
- 4. 集群环境下，若实例一直处于 starting 状态，尝试先停止实例，再启动实例

2. 无法访问页面

1. 检查服务是否正常启动 检查命令 ps -ef|grep AirCityCloud

```
[freedo@kylinV10Sp2Server-01 AirCityCloudPro]$ ps -ef|grep AirCityCloudPro
freedo  59194      1  0 13:59 pts/1    00:00:03 /home/freedo/DTS/AirCityCloudPro/AirCityCloudServer/CloudServer_lin --httpRoot=/home/freedo/DTS/AirCityCloudPro/AirCityCloudSDK/JS --playerPort=8087 --fps=25
--license=/home/freedo/DTS/AirCityCloudPro/AirCityCloudServer/License.lic --enableFileLog=1 --logLevel=Debug --logSubDir=v6.0 --stun=stun:192.168.20.145:3478 --turn=turn:192.168.20.145:3478?transport=udp --turn
Username=freedo --turnPassword=freedo --iceTransportPolicy=relay
freedo  59386      1  0 14:01 pts/2    00:00:00 mono /home/freedo/DTS/AirCityCloudPro/NodeService/NodeService.exe --ip=192.168.20.145 --port=8087 --dataRoot=/home/freedo/DTS/AirCityCloudPro/NodeService/Node
ServiceDataRoot
freedo  59698      1  0 14:08 pts/1    00:00:00 grep AirCityCloudPro
```

2. 检查系统防火墙是否关闭，或者放行 cloud api 的端口，默认 8087/tcp

```
[freedo@kylinV10Sp2Server-01 AirCityCloudPro]$ sudo systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: active (running) since Tue 2023-05-09 09:46:41 CST; 2 days ago
     Docs: man:firewalld(1)
   Main PID: 34937 (firewalld)
      Tasks: 2
     Memory: 24.1M
    CGroup: /system.slice/firewalld.service
            └─34937 /usr/bin/python3 /usr/sbin/firewalld --nofork --nopid

5月 09 09:46:48 kylinV10Sp2Server-01 firewalld[34937]: WARNING: COMMAND_FAILED: '/usr/sbin/iptables -w10 -t filter -X DOCKER' failed: iptables: No chain/target/match by that name.
5月 09 09:46:48 kylinV10Sp2Server-01 firewalld[34937]: WARNING: COMMAND_FAILED: '/usr/sbin/iptables -w10 -t filter -F DOCKER-ISOLATION-STAGE-1' failed: iptables: No chain/target/match by that name.
5月 09 09:46:48 kylinV10Sp2Server-01 firewalld[34937]: WARNING: COMMAND_FAILED: '/usr/sbin/iptables -w10 -t filter -X DOCKER-ISOLATION-STAGE-1' failed: iptables: No chain/target/match by that name.
5月 09 09:46:48 kylinV10Sp2Server-01 firewalld[34937]: WARNING: COMMAND_FAILED: '/usr/sbin/iptables -w10 -t filter -F DOCKER-ISOLATION-STAGE-2' failed: iptables: No chain/target/match by that name.
5月 09 09:46:48 kylinV10Sp2Server-01 firewalld[34937]: WARNING: COMMAND_FAILED: '/usr/sbin/iptables -w10 -t filter -X DOCKER-ISOLATION-STAGE-2' failed: iptables: No chain/target/match by that name.
5月 09 09:46:48 kylinV10Sp2Server-01 firewalld[34937]: WARNING: COMMAND_FAILED: '/usr/sbin/iptables -w10 -t filter -F DOCKER-ISOLATION' failed: iptables: No chain/target/match by that name.
5月 09 09:46:48 kylinV10Sp2Server-01 firewalld[34937]: WARNING: COMMAND_FAILED: '/usr/sbin/iptables -w10 -t filter -X DOCKER-ISOLATION' failed: iptables: No chain/target/match by that name.
5月 09 09:46:48 kylinV10Sp2Server-01 firewalld[34937]: WARNING: COMMAND_FAILED: '/usr/sbin/iptables -w10 -D FORWARD -i docker0 -o docker0 -j DROP' failed: iptables: Bad rule (does a matching rule exist in that
5月 09 10:46:18 kylinV10Sp2Server-01 firewalld[34937]: WARNING: COMMAND_FAILED: '/usr/sbin/iptables -w10 -D FORWARD -i docker0 -o docker0 -j DROP' failed: iptables: Bad rule (does a matching rule exist in that
5月 09 10:48:19 kylinV10Sp2Server-01 firewalld[34937]: WARNING: COMMAND_FAILED: '/usr/sbin/iptables -w10 -D FORWARD -i docker0 -o docker0 -j DROP' failed: iptables: Bad rule (does a matching rule exist in that
[freedo@kylinV10Sp2Server-01 AirCityCloudPro]$ sudo firewall-cmd --list-ports
5900-5902/tcp 8087/tcp 3478/tcp 3478/udp
```

3. player 页面正常的情况下无法获取视频流

问题描述： 卡在以下画面无法获取视频流

```
← ↻ ⚠ 不安全 | 192.168.20.145:8087/samples/locale_zh/player.html?iid=2511176939984
[14:32:15.828] sdk version: 6.0.0509
[14:32:15.830] uid: 1
[14:32:15.830] host: 192.168.20.145:8087
[14:32:16.033] connecting with ws...
[14:32:16.040] connected
[14:32:16.041] process started
[14:32:16.042] setting up...
[14:32:16.045] channel created
[14:32:16.051] __signaling: have-local-offer
[14:32:16.051] __ice_connection: new
[14:32:16.051] __ice_gathering: new
[14:32:16.052] __ice_gathering: gathering
[14:32:16.140] received answer: 6
[14:32:16.142] __signaling: stable
[14:32:16.143] track: video
[14:32:16.174] candidate added
[14:32:16.241] candidate added
```

```
← ↻ ⚠ 不安全 | 192.168.20.139:8087/samples/locale_zh/player.html
[09:42:52.919] sdk version: 6.0.0511
[09:42:52.928] uid: 1
[09:42:52.929] host: 192.168.20.139:8087
[09:42:53.132] connecting with ws...
[09:42:53.140] connected
[09:42:53.141] process started
[09:42:53.142] setting up...
[09:42:53.147] channel created
[09:42:53.154] __signaling: have-local-offer
[09:42:53.154] __ice_connection: new
[09:42:53.155] __ice_gathering: new
[09:42:53.155] __ice_gathering: gathering
[09:42:53.294] __ice_gathering: complete
[09:42:53.306] received answer: 1
[09:42:53.309] __signaling: stable
[09:42:53.309] track: video
[09:42:53.340] candidate added
[09:42:53.341] __ice_connection: checking
[09:42:53.373] candidate added
[09:43:14.423] __ice_connection: disconnected
[09:43:14.424] 连接建立失败，请检查服务器端口设置
```

排查步骤：

- 1. 未使用中继服务的情况下，请确认已完成系统防火墙的配置

2. 通过浏览器打开 webrtc 调试地址 chrome://webrtc-internals/ , 获取 webrtc 调试信息, 确认访问中继服务的地址是否正确

▶ Create Dump

Read stats From:

Standardized (promise-based) getStats() API

Note: computed stats are in []. Experimental stats are marked with an * at the end and do not show up in the getStats result.

http://192.168.20.145:8087/samples/locale_zh/player.html?iid=2511176939984, { iceServers: [stun:192.168.20.145:3478, turn:192.168.20.145:3478?transport=tcp], iceTransportPolicy: relay, bundlePolicy: balanced, iceCandidatePoolSize: 0 }

ICE connection state: new
Connection state: new
Signaling state: new => have-local-offer => stable
ICE Candidate pair: (not connected)
▶ ICE candidate grid

TimeEvent

2023/5/11 14:32:16▶ transceiverAdded

2023/5/11 14:32:16▶ createDataChannel

2023/5/11 14:32:16▶ createOffer

2023/5/11 14:32:16negotiationneeded

2023/5/11 14:32:16▶ createOfferOnSuccess (type: "offer", 3 sections)

2023/5/11 14:32:16▶ setLocalDescription (type: "offer", 3 sections)

2023/5/11 14:32:16setLocalDescriptionOnSuccess

2023/5/11 14:32:16▶ signalingstatechange

2023/5/11 14:32:16▶ transceiverModified

2023/5/11 14:32:16▶ icegatheringstatechange

2023/5/11 14:32:16▶ icecandidateerror

2023/5/11 14:32:16▶ icecandidateerror

2023/5/11 14:32:16▶ setRemoteDescription (type: "answer", 3 sections)

2023/5/11 14:32:16setRemoteDescriptionOnSuccess

2023/5/11 14:32:16▶ signalingstatechange

2023/5/11 14:32:16▶ transceiverModified

2023/5/11 14:32:16▶ addIceCandidate(sdpMid: 0, sdpMLineIndex: 0, type: host)

2023/5/11 14:32:16▶ addIceCandidate(sdpMid: 0, sdpMLineIndex: 0, type: host)

2023/5/11 14:32:55▶ icecandidateerror

Stats Tables

Filter statistics by type including

separate multiple values by ` , `

▶ media-playout (kind=audio, id=AP)

▶ certificate (id=CFC8:2F:AF:D9:22:E4:7F:A5:02:80:3E:F2:B3:E4:B8:B9:F1:F2:F8:43:D5:E7:88:10:67:4E:21)

▶ data-channel (id=D13)

▶ inbound-rtp (kind=video, mid=0, ssrc=3838364922, id=IT01V3838364922)

▶ peer-connection (id=P)

▶ transport (iceState=new, dtlsState=new, id=T01)

3. 确认中继服务正常启动

[root@kylinV10Sp2Server-01 ~]# docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
7f6cb19f6848	coturn/coturn:latest	"docker-entrypoint.s..."	2 days ago	Up 4 seconds	0.0.0.0:3478->3478/tcp, 0.0.0.0:3478->3478/udp, 0.0.0.0:5349->5349/udp, 0.0.0.0:5349->5349/tcp, 0.0.0.0:48100-48200->48100-48200/udp

[root@kylinV10Sp2Server-01 ~]#

4. 确认从服务器本地可以访问映射之后的中继服务端口

[fr-145@kylinV10Sp2Server-01 coturn]\$ telnet 192.168.20.145 3478

Trying 192.168.20.145...

Connected to 192.168.20.145.

Escape character is '^'].

^]

telnet> quit

Connection closed.

5. 确认客户机可以联通中继服务端口而不是被防火墙拦截

C:\Users\zhongxin>telnet 192.168.20.145 3478

正在连接192.168.20.145...无法打开到主机的连接。 在端口 3478: 连接失败

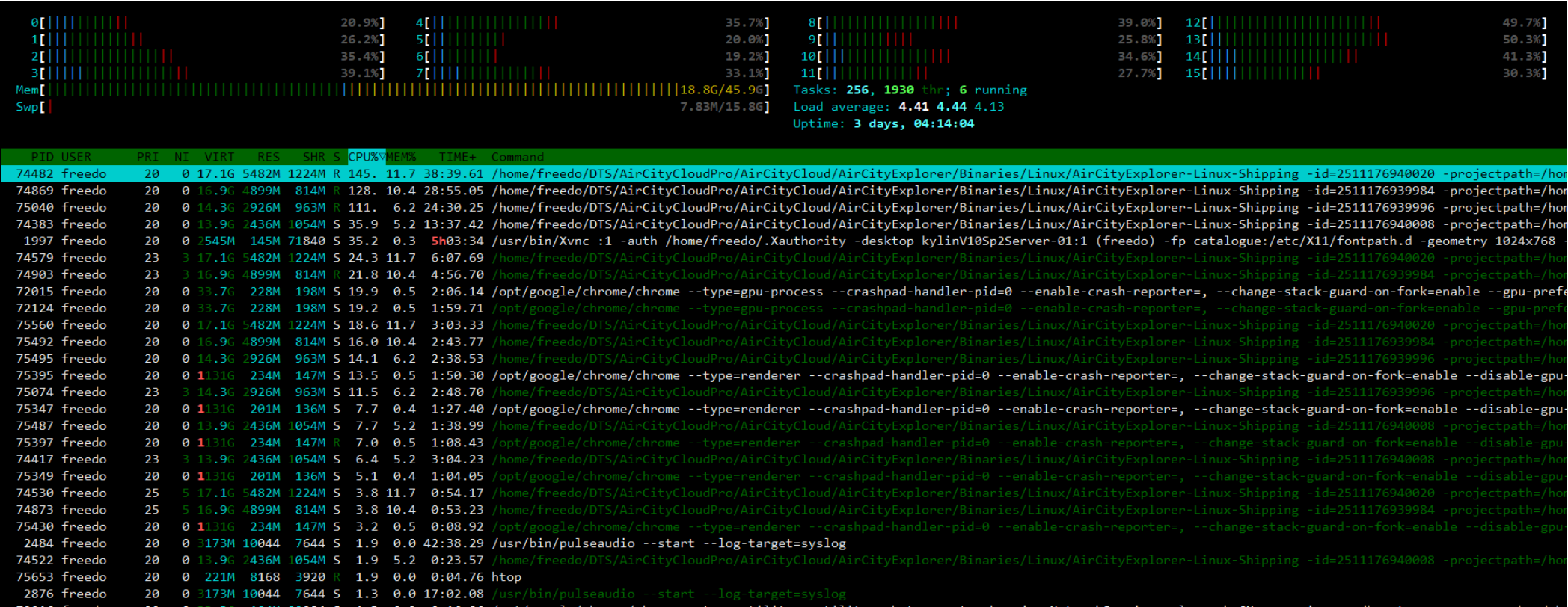
4. 实例卡死、性能问题

1. 使用 htop 命令排查 cpu, 查看各个 cpu 核心使用率

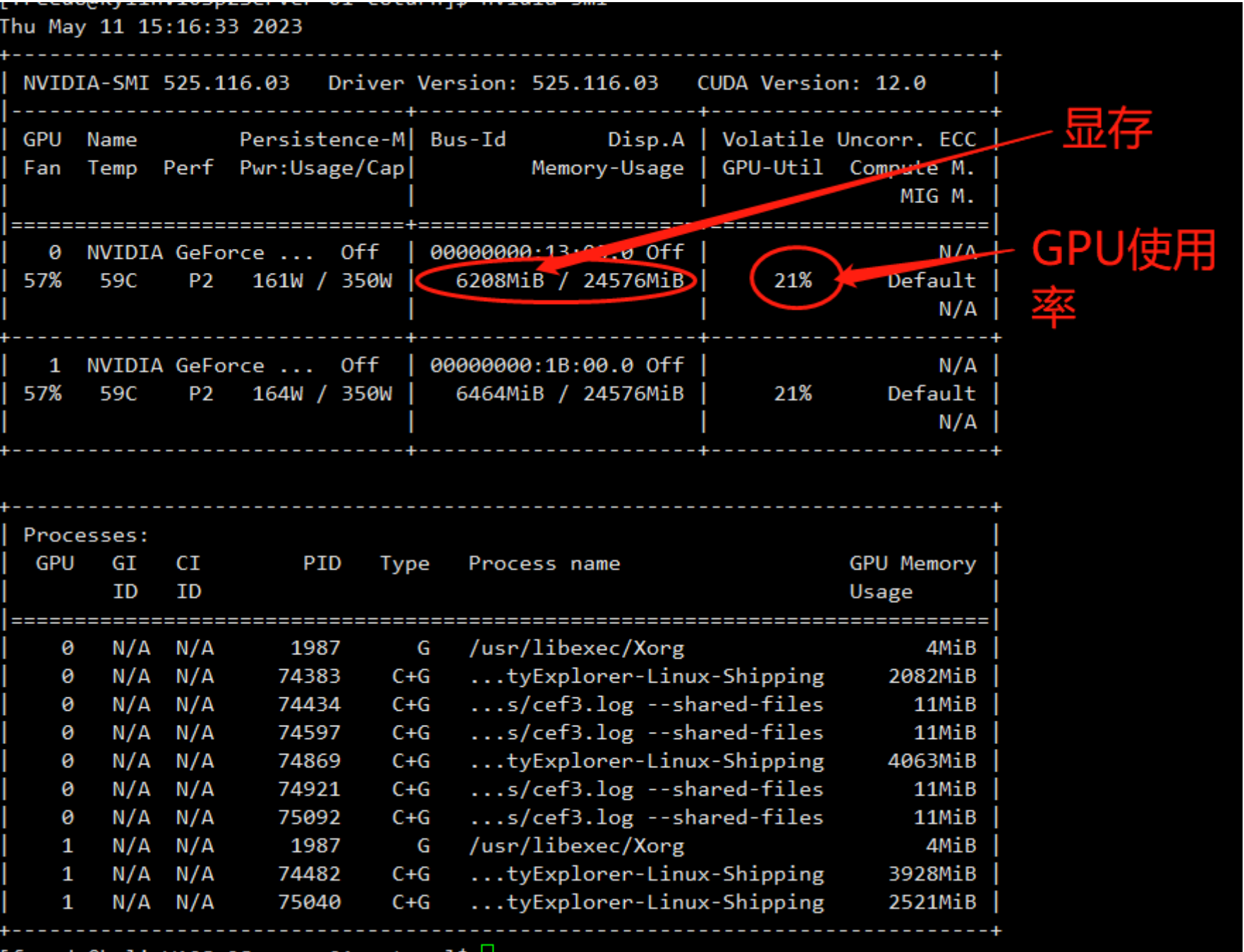
htop 命令系统没有自带, 请单独安装。

1 wget https://update.cs2c.com.cn/NS/V10/V10SP2/os/adv/lic/base/x86_64/Packages/htop-3.0.5-4.ky10.x86_64.rpm

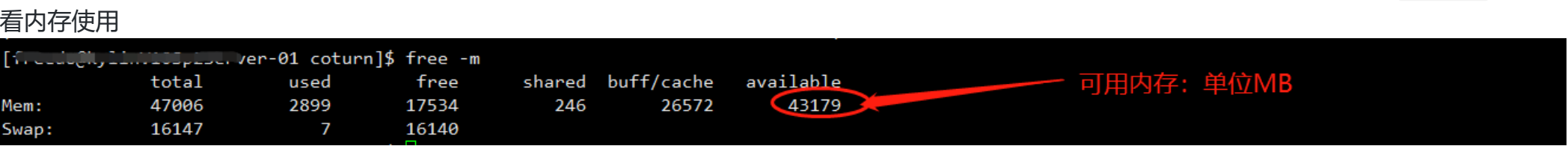
2 sudo dnf localinstall ./htop-3.0.5-4.ky10.x86_64.rpm htop



2. 使用 nvidia-smi 命令查看显卡运行状态



3. 使用 free -m 命令查看内存使用



5. 中继服务无法访问问题

有些服务器修改了内核参数 net.ipv4.ip_forward=0，关闭了IPv4转发，导致无法访问容器端口，从而无法访问容器部署的中继服务

- 修复方式

1. 修改/etc/sysctl.conf

```
1 sudo vi /etc/sysctl.conf
2
3 net.ipv4.ip_forward=1
```

2. 执行命令重载内核参数

```
sudo sysctl -p
```

3. 重启docker服务

```
sudo systemctl restart docker
```