云手机服务器

最佳实践

文档版本	01
发布日期	2023-10-31

华为技术有限公司

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

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

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

目 录

1 云手机接入最佳实践

1.1 创建云手机服务器

操作步骤

1. 登录管理控制台。
2. 在服务列表页，选择“计算 > 云手机服务器CPH”。
3. 在左侧导航栏选择“服务器管理”，单击右上角的“购买服务器实例”。
4. 根据界面提示，完成基础配置，如表1-1。

表 1-1 参数说明

参数	参数说明	样例
计费模式	服务器仅支持一种计费类型：包年/包月。	包年/包月
区域	不同区域的云服务产品之间内网互不相通。建议您选择最靠近您业务的区域，这样可以减少网络时延、提高访问速度。 需要注意：服务器购买成功后不能更换区域。	华东-上海一
可用区	指在同一区域下，电力、网络隔离的物理区域，可用区之间内网互通，不同可用区之间物理隔离。 • 如果您需要提高应用的高可用性，建议您将服务器创建在不同的可用区。 • 如果您需要较低的网络时延，建议您将服务器创建在相同的可用区。	可用区1

参数	参数说明	样例
服务器类型	包括云手机服务器和云手游服务器，请根据业务场景进行选择。详情请参见“**云手机服务器规格**和**云手游服务器规格**”。	云手机服务器 physical.rx1.xlarge
实例规格	请根据业务场景进行选择。	rc1.se
手机镜像	云手机实例运行的操作系统，目前只支持Android系统，其他手机系统由于商业授权缘故，不能提供。 **说明** 查看私有手机镜像列表需要具有对应的细粒度权限：ims:images:list。	AOSP7.1.1
购买量	• 服务器的数量，一次最多可购买10台服务器。 • 购买时长：可选取的时间范围为1个月~3年。	服务器数量：1 购买时长：6个月

5. 单击“下一步：网络配置”。根据界面提示，完成网络配置。
   推荐您使用自定义网络配置，如**表1-2**所示。

表 1-2 自定义网络配置

参数	参数说明	样例
网络	在下拉列表中选择可用的虚拟私有云、子网，并设置私有IP地址的分配方式。 请确定您的登录用户至少具有“VPC ReadOnlyAccess”权限。 云手机网络使用虚拟私有云（VPC）提供的网络，包括子网、安全组等。您可以选择使用已有的虚拟私有云网络，或者创建新的虚拟私有云。 ● 暂不分配IPv6地址/自动分配IPV6地址：当且仅当选择部分区域、部分规格的云手机服务器、且VPC子网开启了IPv6功能时，该参数可见。子网如何开启IPv6功能，请参见**IPv4/IPv6双栈网络**。 系统默认分配IPv4地址，当选择“自动分配IPV6地址”后，网卡的IP地址为IPv4/IPv6双栈类型。 ● 暂不配置/选择需要的共享带宽：在同一VPC内，云手机服务器可通过IPv6地址在双栈服务器之间进行内网访问。如需访问外网，您需要在下拉列表选择您的共享带宽，将IPv6地址加入您的共享带宽。此时云手机服务器可以通过IPv6地址与互联网上的IPv6网络进行访问。 若创建时未选择共享带宽，后续也可在VPC服务参考**（可选）步骤3：购买和加入共享带宽**手动将您的IPv6地址加入共享带宽。 **说明** ● 在创建云手机服务器时，一旦开启IPv6功能，开启成功后，不能修改。 ● 由于VPC限制，上海二不支持开启ipv4/ipv6双栈。 ● IPv6暂不支持使用独享带宽。 ● 单个共享带宽默认能加入的IPv6地址数量最多为20个，双栈云手机服务器IPv6网卡数量和VIP数量相同。若您要购买e0v100等多vip规格的云手机服务器时，建议您提前申请扩大共享带宽的容量配额。 ● IPV6不支持规格为RX1的服务器。	无

参数	参数说明	样例
安全组授权	云手机服务器服务将为您创建一个“cph_admin_trust”委托，该委托包含的权限为：“VPC FullAccess”。 授权云手机服务为您创建委托，需确定您的登录用户具有“Security Administrator”权限。 了解更多请参考**权限管理**。 云手机服务将使用该委托完成如下操作： ● 为云手机实例创建弹性网卡、弹性公网IP、虚拟IP。 ● 为云手机服务器创建默认安全组，并设置安全组，开放端口范围，该端口开放范围将映射到每个云手机/云游戏实例以支持实例开放应用访问端口。 **说明** 同一VPC下的弹性云服务器默认无法通过1-9999端口访问云手机/云游戏实例，若想放开此限制，请参考**使用自定义网络的安全组授权规则**添加高优先级的安全组规则。	无
弹性公网IP	● 现在购买：为云手机购买新的弹性公网IP。 ● 使用已有：为云手机分配已有弹性公网IP。	现在购买
线路	● 静态BGP中的网络结构发生变化，运营商无法在第一时间自动调整网络设置以保障用户的体验度。 ● 全动态BGP可根据设定的寻路协议第一时间自动优化网络结构，以保持客户使用的网络持续稳定、高效。	全动态BGP
公网带宽	仅在新购买弹性公网IP场景下可选择： ● 按流量计费：按照实际使用的流量来计费。 ● 加入共享带宽：一个带宽中可以加入多个弹性公网IP，多个弹性公网IP共用一个带宽。	加入共享带宽
带宽大小	可选带宽范围：1-2000 Mbit/s。	300 Mbit/s
带宽名称	“公网带宽”选择“加入共享带宽”时，请在下拉框中选择已有的共享带宽名称。	bandwidth-001

6. 单击“下一步：高级配置”。根据界面提示，完成高级配置，如**表1-3**所示。

表 1-3 参数说明

参数	参数说明	样例
名称	为您购买的服务器和云手机实例命名，名称不可重复。 命名规则：购买服务器后，名称自动按序增加数字后缀；服务器对应的云手机实例则自动按序增加5位数字后缀。 例如，您购买了1台服务器，该服务器对应60台云手机，名称输入CPH。则，服务器名称为CPH-1，云手机实例的名称为CPH-1-00001~CPH-1-00060。	CPH
密钥对	使用密钥对（Key Pair）进行远程登录身份验证。 ● 如果您已经创建过密钥对，并且本地已妥善保存私钥文件（.pem格式），可以在下拉列表选择已有密钥对。 ● 如果您从未创建过密钥对，请单击“新建密钥对”，跳转至云服务器控制台进行新建。然后返回该页面，刷新下拉列表，选择创建好的密钥对。 私钥用于远程登录身份认证，为保证安全，私钥文件（.pem格式）只能下载一次，请妥善保管。更多关于密钥对的介绍请参见“**创建密钥对**”。 **说明** ● 请确保账号具有查询密钥对的细粒度权限ecs:serverKeypairs:list。 ● 若账号需要创建密钥对，请确保账号具有创建密钥对的细粒度权限ecs:serverKeypairs:create。	KeyPair-test

参数	参数说明	样例
应用端口	高级配置勾选“应用端口”时，该参数可配。适用于云手机需要对外提供服务的场景。 • 应用名称：支持大小写英文字母。不能为关键字“ADB”，大小写的任意组合都不行。 • 端口号：在0~65535之间。 • 公网访问 – 勾选时，表示可通过公网访问云手机该应用端口，即云手机对应的端口以及相对应的服务器公网端口直接暴露在公网，无需鉴权即可互通。 – 不勾选时，表示只能通过租户的私有网络访问。 **注意** • 请谨慎勾选“公网访问”，确保在勾选前已做好安全控制。 • 云手机服务未对客户选择打开的端口做任何安全鉴权。	key 10001 不勾选

7. 单击“下一步：确认订单”，您可再次核对信息。
   – 确认无误后，单击“立即购买”。
   – 如果还需要修改，单击“上一步”，修改参数。
8. 根据界面提示，完成支付。
   大约需要20~30分钟，购买服务器成功后，系统会自动创建好云手机。
   云手机状态变为“运行中”时，表示云手机实例可使用。

1.2 连接云手机并获取云手机画面

Airtest是跨平台UI自动化编译器，可使用Airtest工具快速获取云手机画面。

前提条件

- 已购买并通过ADB方式登录云手机。详细操作请参见“**购买云手机服务器**”。
- 已在本地PC安装Airtest工具。

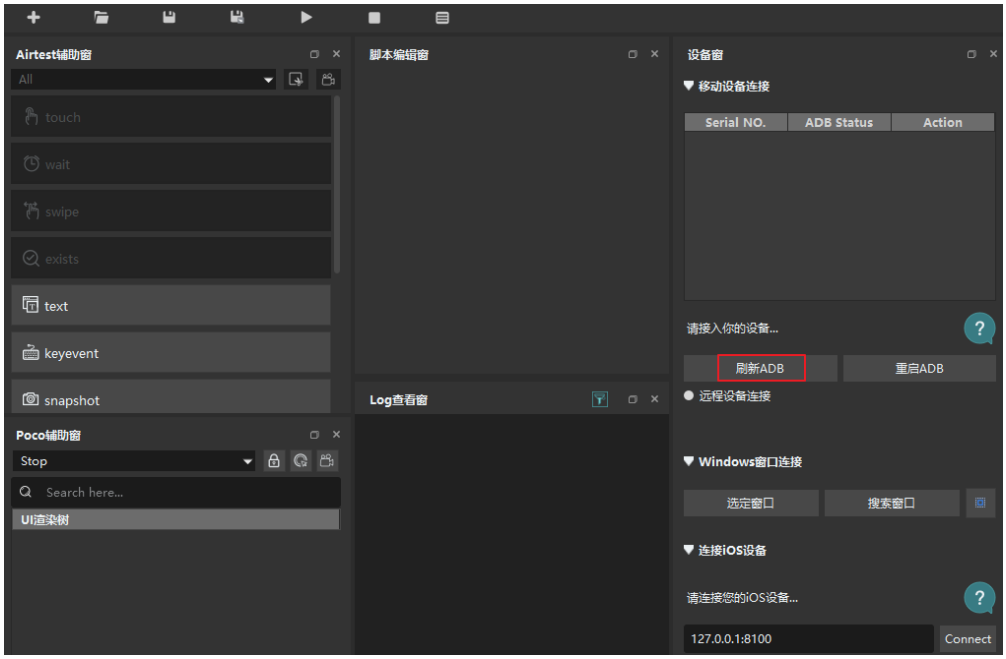
  说明

  登录Airtest官网（**https://airtest.netease.com/**），下载符合您操作系统的版本并安装。
- 已关闭ADB连接的命令行窗口，并保证SSH隧道建立成功。

操作步骤

1. 在Airtest主页单击“刷新ADB”，出现已连接的移动设备：

图 1-1 Airtest 主页



2. 如果没有出现您想要连接的设备，可选择下方的“远程设备连接”，然后手动输入对应云手机的ADB连接命令，如图1-2。

   adb connect 127.0.0.1:*1234*

   其中，*1234*为建立SSH隧道时所使用的本地空闲端口。

图 1-2 远程设备连接

单击右侧的“连接”后，“移动设备链接”中即会出现所需连接的云手机。

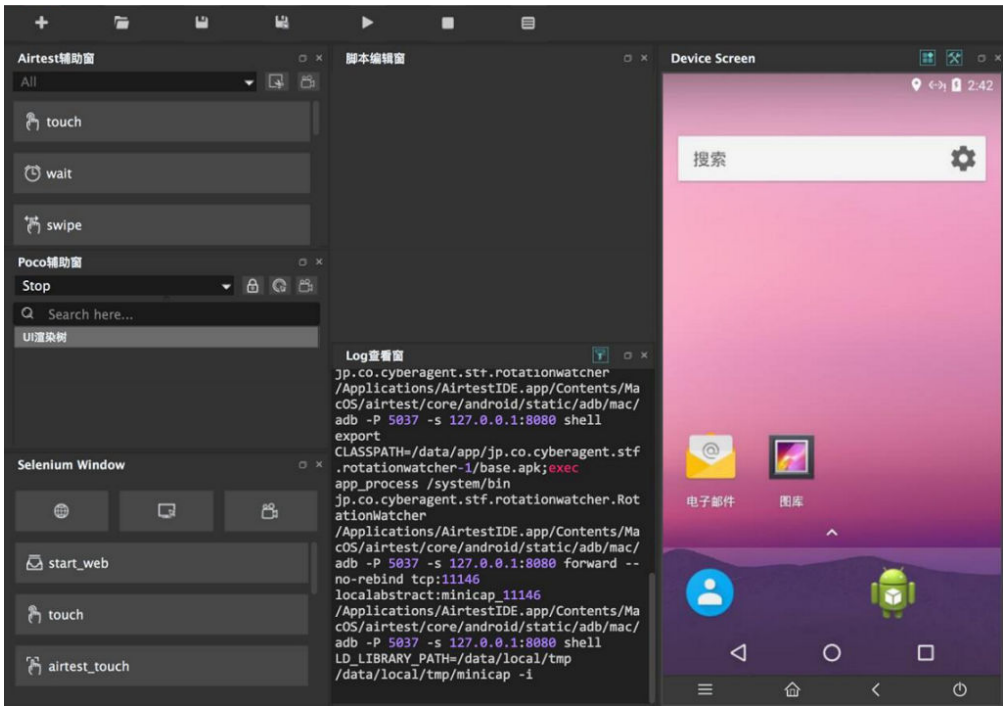
注意

请确保ADB连接的命令行窗口已关闭，否则会连接失败；并且保证SSH隧道建立成功，否则即使已识别出移动设备，“ADB Status”也会出现“offline”状态，导致无法获取云手机画面。

3. 在已识别的移动设备列表中单击对应设备右侧的“connect”，即可获取云手机画面：

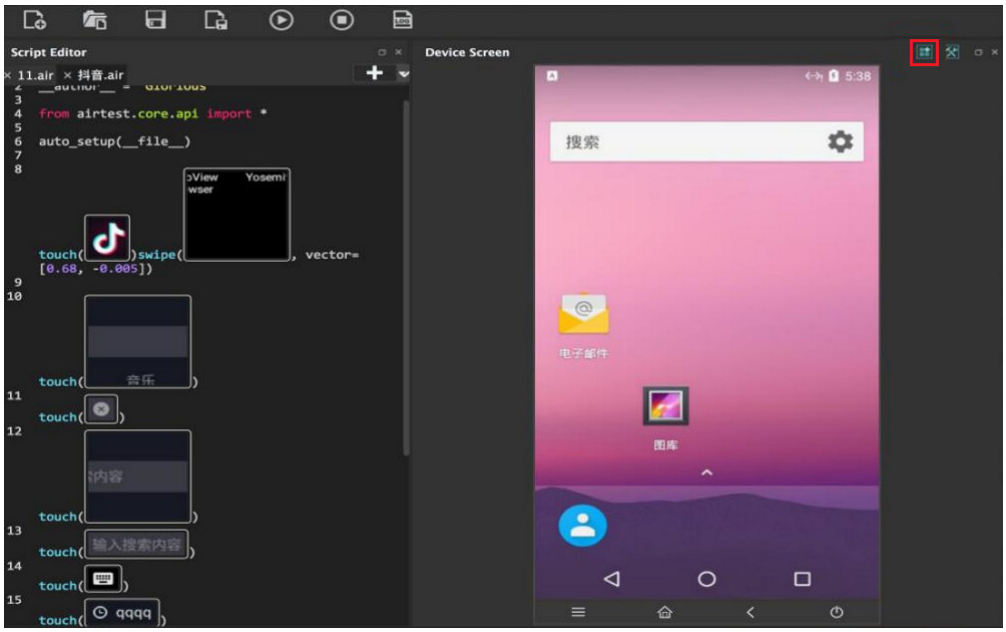
图 1-3 移动设备列表

图 1-4 云手机画面



4. 若通过ADB连接了多台云手机，需要切换画面，可单击右上角的切换图标进行切换。

图 1-5 切换云手机



1.3 云手机内管理程序保活方案

管理程序形态分类

在云手机内的管理程序，基本分为两类：

Android APK Service 形态，即以Android无UI的Service的形式在云手机后台运行。

Android JNI Native 形态，即用Android JNI开发的Executable的二进制可执行程序。

管理程序保活方案

两种形态的管理程序保活方案都依赖一个关键点：extend_custom.sh，即在云手机可内置的钩子脚本。

钩子脚本 extend_custom.sh 的机制说明：

云手机在系统 boot_complete 之后会检查 /data/local/tmp 目录下是否有 extend_custom.sh 脚本，如果有则会执行此脚本。客户可以在此脚本中执行自己文件的操作和移动，也可以启动和管理自己的程序等，但是有个限制条件是脚本执行超时时间是10s。

1. Service形态的管理程序保活

Service形态的程序由于系统机制限制，首次安装之后，如果不启动是属于处于停止状态的，而开机启动完成的广播使用了FLAG_EXCLUDE_STOPPED_PACKAGES ，会使得处于停止状态的应用收不到开机启动完成广播。因此，service管理程序的的安装和首次启动都需要依赖在云手机 /data/local/tmp 目录下内置 extend_custom.sh 脚本来实现，后续重启开机就可以通过接收开机启动完成广播来自启动。

自启动可以在AndroidManifest.xml里配置接收开机广播并在对应的广播处理里启动Service。

```
<!-- example code-->
```



```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<!-- 适配 8.0及以上版本 -->
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>

<receiver android:name=".DemoServiceReceiver">
    <intent-filter android:priority="1000">
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
 </receiver>
```

开机广播处理并启动Service：

```
// example code
public class DemoServiceReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (!intent.getAction().equals("android.intent.action.BOOT_COMPLETED")) {
            return;
        }
        Intent serviceIntent = new Intent(context, DemoService.class);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            context.startForegroundService(serviceIntent);
        } else {
            context.startService(serviceIntent);
        }
    }
}
```

service 程序的保活可以在对应的 Service 类里的 onStartCommand 函数里返回 START_STICKY，这样的话，当 Service 进程被系统 kill 后可以被重新拉起。

```
// example code
public class DemoService extends Service {
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        //other todo...

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) { //适配8.0及以上
            String channel_id = "MyTestService-id";
            String channel_name = "MyTestService-name";
            NotificationManager manager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
            NotificationChannel Channel = new NotificationChannel(channel_id, channel_name,
NotificationManager.IMPORTANCE_HIGH);
            if (manager != null) {
                manager.createNotificationChannel(Channel);
            }
            Notification notification = new
Notification.Builder(this).setChannelId(channel_id).setSmallIcon(R.mipmap.ic_launcher).build();
            startForeground(100, notification);
        }
        return START_STICKY;
    }
}
```

安装和首次启动依赖的 extend_custom.sh 脚本实现，示例如下（示例中默认apk文件也在 /data/local/tmp 目录下）：

```
#  example code

# !/system/bin/sh

# APK名称
ApkName=service.apk
# APK包名
PackageName=com.huawei.myapplication
# Service
ServiceName=com.huawei.myapplication/.MyService
```

```
InstallService() {
        count=`pm list packages | grep $PackageName |wc -l`
        if [ $count -le 0 ]; then
                echo "$ApkName is not installed, now install."
                ret=`pm install $ApkName`
                Succ="Success"
                if [ "$ret" == "$Succ" ]; then
                        echo "install succeed."
                else
                        echo "install failed."
                        exit 1
                fi
        else
                echo "$ApkName is already installed."
                echo "Service start by boot_complete broadcast."
                exit 1
        fi
        return 0
}

StartService() {
    ret=`am startservice -n $ServiceName`
    Err="Error: Not found; no service started."
        contain=$(echo $ret | grep "${Err}")
        if [[ "$contain" != "" ]]; then
                echo "Service start failed."
                exit 1
        else
                echo "Service start succeed."
        fi
        return 0
}

main() {
        # 安装
        InstallService

        echo "To start when first installed."
        # 启动
        StartService
}
```

2. Native形态的管理程序保活

Native 管理程序需要被放在云手机 /system/bin 目录下，并给予可执行权限，Native 程序依赖的二进制so库需要放在 system/lib 和 system/lib64 目录下。

Native程序、二进制so库文件等都可以通过共享存储或共享应用的方式推进手机的data目录，然后利用 extend_custom.sh 脚本将自己的文件放置在对应的目录下。针对使用诉求不同，有以下几种方式：

（1）系统级保活

可以实现 init_custom.rc 文件，并将文件移动到 /data/local/目录下，然后需要重启手机生效，重启手机后系统会扫描到 init_custom.rc 文件并在 boot_completed 之后启动 NativeDemo（示例中 Native 管理程序名），如果 NativeDemo 进程异常退出或被kill都会被系统再次拉起。

```
on property:sys.boot_completed=1
    start NativeDemo

service NativeDemo /system/bin/NativeDemo
    user root
    group root
    disabled
writepid /dev/cpuset/system-background/tasks
```

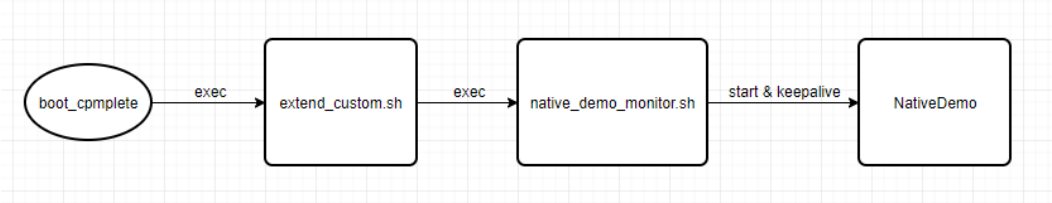
系统级保活方式有以下优缺点：

优点：系统级的自启动和保活，可靠性高，拉起实时性高。

缺点：需要重启一次手机才可生效。

（2）用户级保活

如果既需要开机自启又需要保活管理程序，而且不希望重启手机生效，可以先在extend_custom.sh 脚本中执行一个管理程序的检测保活脚本native_demo_monitor.sh，由管理程序的检测保活脚本真正实现 NativeDemo 管理程序的启动和保活。



extend_custom.sh 钩子脚本实现示例：

```
#  example code

# !/system/bin/sh

[[ -f /data/local/tmp/native_demo_monitor.sh ]] && sh /data/local/tmp/native_demo_monitor.sh &
```

native_demo_monitor.sh 脚本实现示例：

```
#  example code
# !/system/bin/sh

file_dir="/data/demo"

CopyFile() {
	cp -rf $file_dir/NativeDemo /system/bin/
	chmod 755 /system/bin/NativeDemo

	cp $file_dir/lib*.so /system/lib64/
}

CheckIfCopyFile() {
	if [ -s $file_dir ]; then
		echo "file exist"
		CopyFile
	else
		echo "file not exist"
	fi
}

Start() {
	nohup NativeDemo &
}

KeepAlive() {
	while do
		echo "check NativeDemo proc"
		proc_count=`ps | grep NativeDemo | wc -l`
		if [ $proc_count -le 0 ]; then
			echo "start NativeDemo"
			Start
		else
			echo "NativeDemo already started"
		fi
		sleep 5
	done
}
```

```
main() {
        CheckIfCopyFile
        KeepAlive
}

main
```

用户级保活方式有以下优缺点：

优点：不需要重启手机即可生效。

缺点：拉起的实时性不如系统级，依赖检查的循环间隔时间。

1.4 云手机应用部署

1.4.1 获取云手机列表

参考《云手机服务器API参考》的“**查询云手机列表**”章节获取云手机列表。

接口示例

```
GET https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones?
phone_name={phone_name}&server_id={server_id}&status={status}&offset={offset}&limit={limit}&type={type}
Header:
Content-Type: application/json
X-Auth-Token: ${token}
```

其中，

- CPH Endpoint为Endpoint列表中CPH对应区域的终端节点，如华北-北京四为cph.cn-north-4.myhuaweicloud.com。
- project_id为云手游服务器所属区域对应的项目ID，如083e9f825e80f50c2f96c0045edc70e8。可通过如下方式获取：
  a. 登录管理控制台。
  b. 单击右上角用户名下的“我的凭证”。
  c. 在“API凭证”页面的项目列表中获取项目ID。

  图 1-6 获取项目 ID

- URL中"?"之后的部分为可选参数。
- $token为**获取token**接口的响应结果。

接口使用示例

```
GET https://cph.cn-north-4.myhuaweicloud.com/v1/083e9f825e80f50c2f96c0045edc70e8/cloud-phone/phones
Header:
Content-Type: application/json
X-Auth-Token: ${token}
```

说明

${token}需要换成实际获取到的token值。

1.4.2 为单台云手机安装应用

参考《云手机服务器API参考》的“**安装apk**”章节为单台云手机安装应用。

前提条件

- 云手机服务器所在Region的OBS桶中已经存放好对应的APK安装包。否则，需要上传安装包，可参考**场景二：快速通过OBS Browser+上传下载文件**进行上传。
- 确保已经配置好OBS桶的授权策略，若未配置请参考**委托CPH操作OBS桶**章节来完成配置。

接口示例

```
POST https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones/commands
Header:
Content-Type: application/json
X-Auth-Token: ${token}
Body:
{
   "command": "install",
   "content": "-t -r obs://{bucket_name}/{object_path}",
   "phone_ids": [
      "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
   ]
}
```

其中，

- CPH Endpoint、project_id、${token}等参数取值可以参考**获取云手机列表**章节的说明获取。
- bucket_name为对象存储服务OBS的桶名，object_path为apk安装包的存放路径。
- phone_ids为需要安装应用的云手机ID（通过**获取云手机列表**获得，可以填多个，填多个则为多台云手机安装apk应用）。

接口使用示例

```
POST https://cph.cn-east-3.myhuaweicloud.com/v1/081ceeb7fb800f0c2f4cc004bb39c2f7/cloud-phone/phones/commands
Content-Type: application/json
X-Auth-Token: ${token}
{
   "command": "install",
   "content": "-t -r obs://yzw-apk-install/apk/com.hermes.bgame.apk",
   "phone_ids": [
      "bdc2f2e960164dd9a2765374afeea300"
   ]
}
```

- yzw-apk-install为OBS桶名，apk/com.hermes.bgame.apk为安装包存放路径，obs://yzw-apk-install/apk/com.hermes.bgame.apk为安装包全路径。

- ${token}需要换成实际获取到的token值。

1.4.3 生成应用版本 tar 包并推至 OBS 桶

前提条件

- 需确保云手机已安装对应的应用。
- 确保已配置好OBS桶的授权策略，具体操作请参考**委托CPH操作OBS桶**章节。

接口示例

```
POST https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones/batch-storage
Header:
Content-Type: application/json
X-Auth-Token: ${token}
Body:
{
    "storage_infos": [{
        "phone_id": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
        "include_files": [
            "/data/app/${package_name}-*",
            "/data/data/${package_name}",
            "/data/media/0/Android/data/${package_name}"
        ],
        "bucket_name": "${bucket_name}",
        "object_path": "apk/${package_name}_${version_name}.tar"
    }]
}
```

其中，

- CPH Endpoint、project_id、${token}、bucket_name、object_path等参数可以参考**为单台云手机安装应用**章节的说明获取。
- phone_id为安装了对应应用的云手机ID。
- include_files中的三个元素需要填写手机中的绝对路径。
- 如果该安装包为xapk类型，则需要在include_files中增加“/data/media/obb/${package_name}”路径。
- object_path为tar包上传至OBS桶的目标路径。

须知

1. object_path中apk为obs桶中已存在文件夹，${package_name}为当前应用的包名，${version_name}为当前应用的版本号，版本号可自行定义。
2. 针对共享应用场景，部分应用在启动后，会进行资源文件的在线下载。对于此类应用，在执行当前操作前，建议先启动一次应用，待资源文件和补丁包下载完成后，再执行当前操作。后续以共享应用的方式安装到云手机后，该应用的启动过程可省去资源文件下载的过程。

1.4.4 部署应用

存储 2.0 机器（推荐）

推送tar包至服务器，即将文件“apk/${package_name}_${version_name}.tar”推送到服务器（${server_id1}和${server_id2}）的共享应用中。

- 接口示例

```
POST https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones/share-apps
Header:
Content-Type: application/json
X-Auth-Token: ${token}
Body:
{
    “package_name”：“${package_name}”
   "bucket_name": "${bucket_name}",
   "object_path": "apk/${package_name}_${version_name}.tar",
   "server_ids": [
      "${server_id1}",
      "${server_id2}"
   ]
}
```

其中，

  - CPH Endpoint、project_id、${token}、bucket_name、object_path等参数可以参考**1.4.2-为单台云手机安装应用**的说明获取。
  - package_name为应用在安卓系统中的包名，例如：com.miniteck.miniworld。
  - object_path为tar包上传的目的路径。
  - package_name为当前应用的包名，version_name为当前应用的版本号。

    说明

    apk为任意已存在文件夹，${package_name}_${version_name}.tar中除.tar后缀外，其余内容需根据实际名称来修改。
  - server_ids为部署应用版本的目标服务器ID列表，可以填写多个，服务器ID可通过“**查询云手机服务器列表**”接口获得。

- 使用示例

  参考《 云手机服务器API参考 》的“**推送共享应用**”章节。

存储 1.0 机器

推送tar包至服务器，即将文件“apk/${package_name}_${version_name}.tar”推送到服务器（ ${server_id1}和${server_id2} ）的共享存储中。

- 接口示例

```
POST https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones/share-files
Header:
Content-Type: application/json
X-Auth-Token: ${token}
Body:
{
   "bucket_name": "${bucket_name}",
   "object_path": "apk/${package_name}_${version_name}.tar",
   "server_ids": [
      "${server_id1}",
      "${server_id2}"
   ]
}
```

其中，

  - CPH Endpoint、project_id、${token}、bucket_name、object_path等参数可以参考**4.2 为单台云手机安装应用**的说明获取。
  - object_path为tar包上传的目的路径。
  - package_name为当前应用的包名，version_name为当前应用的版本号。

> **说明**
>
> apk为任意已存在文件夹，${package_name}_${version_name}.tar中除.tar后缀外，其余内容需根据实际名称来修改。

  - server_ids为部署应用版本的目标服务器ID列表，可以填写多个，服务器ID可通过“**查询云手机服务器列表**”接口获得。

- 使用示例

  参考《云手机服务器API参考》的“**推送共享存储文件**”章节。

- 后续操作

  参考《云手机服务器API参考》“**重置云手机**”接口，批量重置所有云手机。

1.4.5 更新应用版本

存储 2.0 的机器（推荐）

确保手机已经卸载应用

推送最新的应用

随后，在需要时调用appctrl start会启动最新版本应用

存储 1.0 的机器

更新应用版本，需要先删除服务器上的旧版本应用，然后重新部署新版本应用。

删除旧版本应用

- 接口示例：

```
POST https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones/share-files
Header:
Content-Type: application/json
X-Auth-Token: ${token}
Body:
{
   "file_paths": [
      "/data/app/${package_name}-1",
      "/data/app/${package_name}-2",
      "/data/data/${package_name}",
      "/data/media/0/Android/data/${package_name}"
   ],
   "server_ids": [
      "${server_id1}",
      "${server_id2}"
   ]
}
```

  删除服务器（${server_id1}和${server_id2}）共享存储中的文件，文件列表包含："/data/app/${package_name}-1", "/data/app/${package_name}-2", "/data/data/${package_name}", "/data/media/0/Android/data/${package_name}"。

  其中，

  - CPH Endpoint、project_id、${token}等参数可以参考**为单台云手机安装应用**的说明获取。
  - file_paths的内容等同于**生成应用版本tar包并推至OBS桶**中的include_files，其中package_name为当前应用的包名。

  - server_ids为部署应用版本的目标服务器ID列表,可以填写多个，服务器ID可通过“**查询云手机服务器列表**”接口获得。

- 使用示例：
  参考《 云手机服务器API参考 》的“**删除共享存储文件**”章节。

部署新版本应用

参考**部署应用**来部署应用。

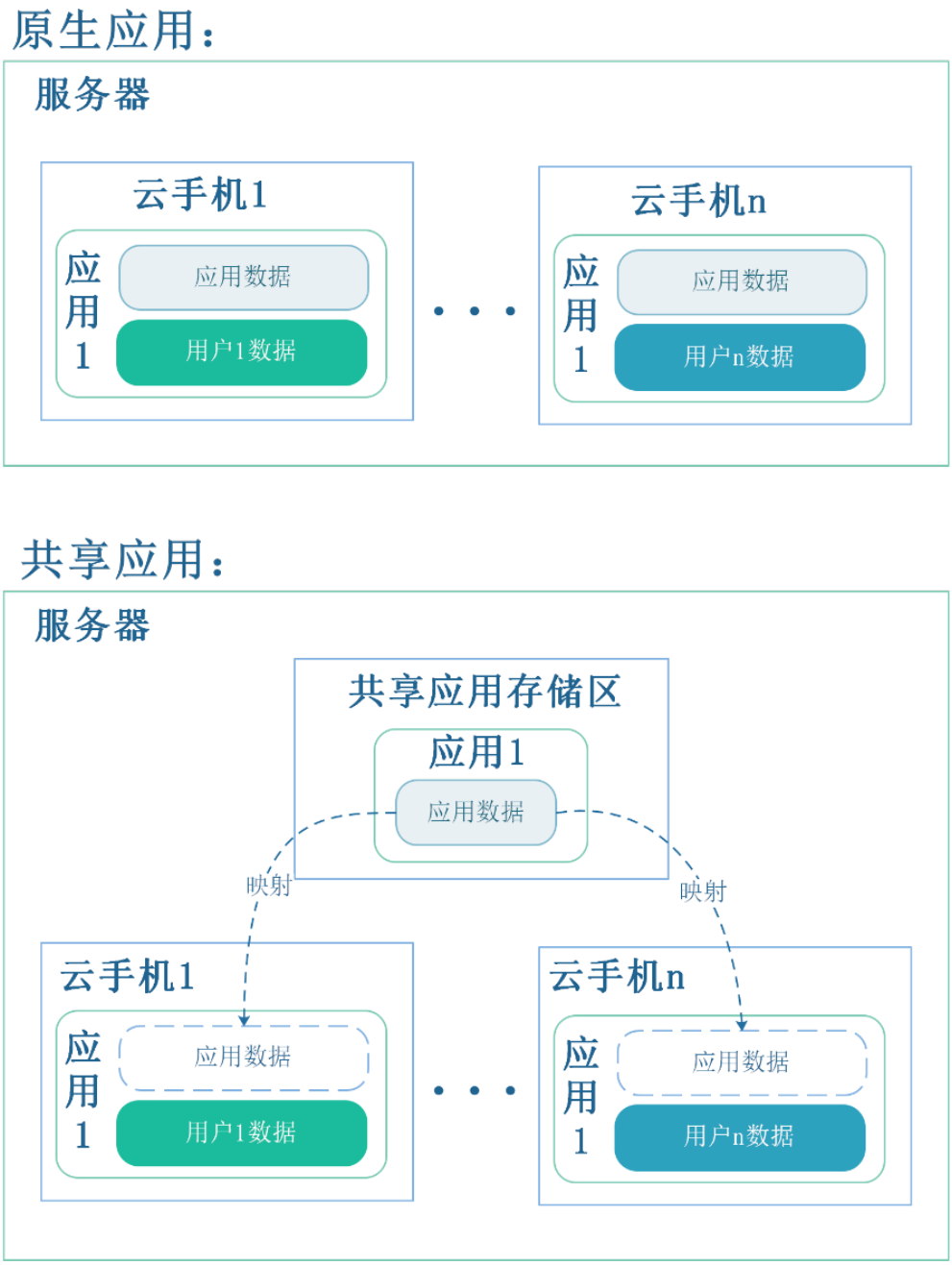
2 云手机共享应用管理最佳实践

2.1 共享应用特性综述

共享应用是什么?

云手机服务器提供一块共享应用存储区（简称共享空间），服务器所属用户只需要下载/更新一次应用，然后将应用数据打包上传至指定服务器的共享空间，即可以实现该服务器上所有的云手机实例共享该应用。

图 2-1 共享应用



对于云手机服务器用户而言，共享应用可以避免同样的应用数据在多台云手机里重复下载、存储、更新，降低存储和网络成本。除此之外，共享应用还有以下优点：

- 部署方式灵活可控：云手机应用既可批量部署，也可按云手机实例单台部署。
- 方便快捷极致体验：将更新后的应用直接共享，简化安装流程，免除更新操作，真正实现即连即用。

共享应用有哪些典型场景?

1. 对于需要为服务器上的云手机实例按需安装、更新应用的场景，请参考**云手机按需安装**。

2. 对于需要为整台服务器上所有的云手机实例预安装、更新相同应用的场景，请参考**云手机预装应用**
3. 对于非应用类文件，可以使用共享应用的能力，实现文件的批量部署的场景，请参考**云手机部署文件**。

2.2 共享空间中的应用包管理

在使用共享应用前，首先需要确保为云手机服务器配置了合适的共享空间，共享空间大小在购买服务器时进行配置；其次，需要将应用的相关文件打包推送到服务器的共享空间中；当某应用不再使用时，可以将共享空间中的应用删除，以便释放共享空间来推送其他应用。

2.2.1 配置共享空间

在购买服务器实例页面，选定区域、服务器类型、实例规格后，配置共享空间大小。建议在实际业务共享应用所需空间大小的基础上，上浮30%来配置共享空间的大小，以满足后续应用更新对共享空间的需求。

图 2-2 配置共享空间

2.2.2 准备共享应用 tar 包

2.2.2.1 获取云手机列表

参考《云手机服务器API参考》的“**查询云手机列表**”章节获取云手机列表。

接口示例

```
GET https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones?
phone_name={phone_name}&server_id={server_id}&status={status}&offset={offset}&limit={limit}&type={type}
Header:
Content-Type: application/json
X-Auth-Token: ${token}
```

其中，

- CPH Endpoint为Endpoint列表中CPH对应区域的终端节点，如华北-北京四为cph.cn-north-4.myhuaweicloud.com。

- project_id为云手游服务器所属区域对应的项目ID，如083e9f825e80f50c2f96c0045edc70e8。可通过如下方式获取：
  a. 登录管理控制台。
  b. 单击右上角用户名下的“我的凭证”。
  c. 在“API凭证”页面的项目列表中获取项目ID。

  图 2-3 获取项目 ID

- URL中"?"之后的部分为可选参数。
- $token为**获取Token**接口的响应结果。

接口使用示例

```
GET https://cph.cn-north-4.myhuaweicloud.com/v1/083e9f825e80f50c2f96c0045edc70e8/cloud-phone/phones
Header:
Content-Type: application/json
X-Auth-Token: ${token}
```

说明

${token}需要换成实际获取到的token值。

2.2.2.2 为单台云手机安装应用

参考《云手机服务器API参考》的“**安装apk**”章节为单台云手机安装应用。

前提条件

- 云手机服务器所在Region的OBS桶中已经存放好对应的APK安装包。否则，需要上传安装包，可参考**场景二：快速通过OBS Browser+上传下载文件**进行上传。
- 确保已经配置好OBS桶的授权策略，若未配置请参考**委托CPH操作OBS桶**章节来完成配置。

接口示例

```
POST https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones/commands
Header:
Content-Type: application/json
X-Auth-Token: ${token}
Body:
{
   "command": "install",
   "content": "-t -r obs://{bucket_name}/{object_path}",
   "phone_ids": [
      "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

```
    ]
}
```

其中，

- CPH Endpoint、project_id、${token}等参数取值可以参考**获取云手机列表**章节的说明获取。
- bucket_name为对象存储服务OBS的桶名，object_path为apk安装包的存放路径。
- phone_ids为需要安装应用的云手机ID（通过**获取云手机列表**获得，可以填多个，填多个则为多台云手机安装apk应用）。

接口使用示例

```
POST https://cph.cn-east-3.myhuaweicloud.com/v1/081ceeb7fb800f0c2f4cc004bb39c2f7/cloud-phone/phones/commands
Content-Type: application/json
X-Auth-Token: ${token}
{
    "command": "install",
    "content": "-t -r obs://yzw-apk-install/apk/com.hermes.bgame.apk",
    "phone_ids": [
        "bdc2f2e960164dd9a2765374afeea300"
    ]
}
```

- yzw-apk-install为OBS桶名，apk/com.hermes.bgame.apk为安装包存放路径，obs://yzw-apk-install/apk/com.hermes.bgame.apk为安装包全路径。
- ${token}需要换成实际获取到的token值。

2.2.2.3 生成应用版本 tar 包并推至 OBS 桶

前提条件

- 需确保云手机已安装对应的应用。
- 确保已配置好OBS桶的授权策略，具体操作请参考**委托CPH操作OBS桶**章节。

接口示例

```
POST https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones/batch-storage
Header:
Content-Type: application/json
X-Auth-Token: ${token}
Body:
{
    "storage_infos": [{
        "phone_id": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
        "include_files": [
            "/data/app/${package_name}-*",
            "/data/data/${package_name}",
            "/data/media/0/Android/data/${package_name}"
        ],
        "bucket_name": "${bucket_name}",
        "object_path": "apk/${package_name}_${version_name}.tar"
    }]
}
```

其中，

- CPH Endpoint、project_id、${token}、bucket_name、object_path等参数可以参考**为单台云手机安装应用**章节的说明获取。

- phone_id为安装了对应应用的云手机ID。
- include_files中的三个元素需要填写手机中的绝对路径。
- 如果该安装包为xapk类型，则需要在include_files中增加“/data/media/obb/${package_name}”路径。
- object_path为tar包上传至OBS桶的目标路径。

须知

1. object_path中apk为obs桶中已存在文件夹，${package_name}为当前应用的包名，${version_name}为当前应用的版本号，版本号可自行定义。
2. 针对共享应用场景，部分应用在启动后，会进行资源文件的在线下载。对于此类应用，在执行当前操作前，建议先启动一次应用，待资源文件和补丁包下载完成后，再执行当前操作。后续以共享应用的方式安装到云手机后，该应用的启动过程可省去资源文件下载的过程。

2.2.3 推送应用 tar 包至服务器共享空间

调用接口将OBS桶内的应用tar包推送至云手机服务器共享空间。

调用示例

```
POST https://${CPH Endpoint}/v1/${projectId}/cloud-phone/phones/share-apps
Header:
Content-Type: application/json
X-Auth-Token: ${token}
Body:
{
    "package_name": "${package_name}",
    "bucket_name": "${bucket_name}",
    "object_path": "apk/${package_name}_${version_name}.tar",
    "pre_install_app": 0,
    "server_ids": [
        "${server_id1}",
        "${server_id2}"
    ]
}
```

其中，

- ${bucket_name}为对象存储服务OBS的桶名，object_path为应用tar包在OBS桶中的存放路径。
- pre_install_app 为1表示将待推送的应用设置为预装应用，0表示该应用为非预装。（预装应用的具体说明请参考**云手机预装应用**）。如果应用被设置为预装应用，当云手机重置后，应用会自动安装到云手机中。
- server_ids为准备接受应用tar包推送的服务器ID列表。一次可指定多个服务器ID。
- 该接口的更多说明，请参考**推送共享应用**。

须知

1. 同一款应用可多次推送，后推送的版本作为该应用的最新版本，多个版本可同时存在于共享空间中。
2. package_name为应用的真实包名，不能被修改。如果是渠道包，可能会存在渠道包后缀，可以在**获取云手机列表**中已安装该应用的云手机中查看真实包名。

   例如：

   官网下载包名：com.huawei.xxxx

   其他渠道下载包名：com.huawei.xxxx.huawei

2.2.4 从服务器共享空间删除应用

当云手机服务器上的所有云手机均不再使用某一共享应用时，需要从云手机服务器的共享空间中删除该共享应用，以释放存储空间。

约束与限制

云手机服务器上所有云手机均已卸载该共享应用。

调用示例

```
DELETE "https://${CPH Endpoint}/v1/${projectId}/cloud-phone/phones/share-apps"
Header:
Content-Type: application/json
X-Auth-Token: ${token}
Body:
{
    "package_name": "com. huawei.xxxx",
    "server_ids": ["1678567b8bab40f937112xxxxx","1234567b8bab40ffb711xxxxx"]
}
```

其中，package_name 为应用的真实包名，server_ids为服务器ID列表，可以指定多个服务器ID。

详细步骤请参考**删除共享应用**。

须知

1. 在删除共享应用前，请确认服务器上所有云手机均已卸载该共享应用。如果没有卸载，会导致共享空间中的应用无法删除，调用删除接口失败时会返回未卸载该应用的云手机列表信息。
2. 如果共享空间中存在某款共享应用的多个版本，并且旧版本没有被任何一台云手机安装，该旧版本会被定期自动清理。

2.3 共享应用使用场景

共享应用tar包推送到共享空间后，可通过共享应用命令实现共享应用的快速安装、更新、卸载。

2.3.1 云手机按需安装

2.3.1.1 应用安装

业务场景示例

- 云游戏场景：用户接入后直接进入某款游戏。
- 个人云手机场景：用户接入后先看到云手机画面，按需选择安装共享空间中已有的应用。

操作步骤

在云手机中执行appctrl命令（参考**appctrl命令**）。

1. 使用appctrl install命令安装应用
   使用指导：appctrl install {包名}
   调用示例：appctrl install com. huawei.xxxx
2. 使用appctrl start命令安装并启动应用
   使用指导：appctrl start {包名}
   调用示例：appctrl start com. huawei.xxxx

须知

当共享空间某个应用有多个版本时，appctrl install 和appctrl start会自动选择该应用最新版本进行安装。

2.3.1.2 应用更新

业务场景示例

- 当某款应用发布新版本，需对共享应用版本进行更新。
- 当某款游戏发布了新的在线资源更新，需对共享应用中的资源提前进行更新，避免云游戏场景下用户等待在线资源的下载和安装。

操作步骤

1. 参考**获取云手机列表**、**为单台云手机安装应用**在单台手机上安装应用最新的版本。
2. 如果应用存在资源更新，请启动应用并完成资源文件的下载和更新。
3. 参考**生成应用版本tar包并推至OBS桶**和**推送应用tar包至服务器共享空间**生成并推送tar包到服务器共享空间。
4. 执行appctrl命令进行更新（参考**appctrl命令**）。
   - 执行appctrl start 命令，云手机会安装该应用共享空间中最新版本并启动。
   - 执行appctrl install命令，云手机会安装该应用共享空间中最新版本。

说明

1. 当共享空间中某应用没有更新版本时，appctrl start和 appctrl install不会对已安装版本进行重复安装。

2. 对于采用共享方式安装的应用，如果启动应用后采用在线更新的方式更新资源，该更新后的资源会占用云手机的存储空间。因此当应用存在必要的在线更新时，建议提前对共享空间中的应用版本和在线资源进行更新。

2.3.1.3 应用卸载

操作步骤

在云手机中执行appctrl命令，对已安装在云手机中的共享应用进行卸载（参考**appctrl命令**）。

使用指导：在云手机中执行appctrl uninstall {包名}

例如：卸载应用

appctrl uninstall com. huawei.xxxx

须知

appctrl uninstall命令，仅适用于共享应用，通过原生接口安装的应用无法使用该命令卸载。

2.3.1.4 应用批量卸载

业务场景示例

客户分配的云手机时间到期，需要回收进入待分配区，并进行共享应用批量卸载，卸载上一个玩家安装的共享应用。

操作步骤

在云手机中执行appctrl命令，对使用appctrl命令安装的所有共享应用进行卸载（参考**appctrl命令**）。

使用指导：在云手机中执行appctrl clear

须知

appctrl clear命令，仅适用于使用appctrl install和appctrl start命令安装的所有共享应用，原生接口安装的应用和预装的共享应用无法使用该命令卸载。

2.3.2 云手机预装应用

2.3.2.1 预装应用安装

业务场景示例

服务器上所有云手机都要安装某些应用，并且云手机重置后默认安装这些应用。

操作步骤

1. 参考**推送应用tar包至服务器共享空间**，推送需要预装的应用tar包到服务器共享空间，将pre_install_app设置为1。
2. 重置云手机，所有预装应用会自动安装。

须知

1. 预装应用的安装过程会占用一定的手机重置时间。
2. 若云手机正在使用中，无法重置进行重置，也可使用appctrl install命令直接安装应用。
3. 取消应用的预装，参考**推送应用tar包至服务器共享空间**，再次推送该应用的tar包并将pre_install_app参数设置为0；再次重置云手机时，该应用不会自动安装到云手机中。

2.3.2.2 预装应用更新

操作步骤

1. 参考**为单台云手机安装应用**，安装新的应用版本。
2. 启动应用更新下载资源。
3. 参考**生成应用版本tar包并推至OBS桶**，生成共享应用tar包并推送到OBS桶中。
4. 参考**推送应用tar包至服务器共享空间**，推送需要预装的应用tar包到服务器共享空间，将pre_install_app设置为1。
5. 重启或重置云手机，所有预装应用会自动更新。

须知

预装应用的更新过程会占用一定的手机重启和重置时间。

2.3.2.3 预装应用卸载

业务场景示例一

单台或部分手机不再使用预装的应用。

操作步骤

1. 参考**应用卸载**，卸载手机上的对应应用。

须知

卸载应用后，重置手机应用会重新自动安装。

业务场景示例二

服务器上所有的手机都不再使用预装的应用。

操作步骤

1. 参考**应用卸载**，卸载服务器上所有手机的对应应用。
2. 参考**从服务器共享空间删除应用**，从服务器共享应用区域删除应用。

2.3.3 云手机部署文件

除了共享应用的使用场景外，部分用户业务上需要在云手机中部署文件，部署前需要将这类文件打成tar包，下文中统称这种tar包为配置文件包。

2.3.3.1 文件预置

业务场景示例

需要在服务器上的所有云手机中预置文件。

例如：推流二进制文件、保活脚本文件、应用行为监控文件等。

操作步骤

1. 参考**获取云手机列表**，从列表中选定一台云手机，将需要打包的文件放入到该云手机的目标路径下。

   例如：服务器上云手机需要在/data/local/huawei/、/data/local/tmp/路径下分别部署test.txt和test.sh文件。将test.txt、test.sh放入选定手机的/data/local/huawei/、/data/local/tmp/目录下。
2. 将文件打包成配置文件tar包并上传到OBS桶。

   调用示例：

```
POST https://${CPH Endpoint}/v1/${project_id}/cloud-phone/phones/batch-storage
Header:
Content-Type: application/json
X-Auth-Token: ${token}
Body:
{
    "storage_infos": [{
        "phone_id": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
        "include_files": [
            "/data/local/huawei/test.txt",
            "/data/local/tmp/test.sh"
        ],
        "bucket_name": "${bucket_name}",
        "object_path": "apk/${package_name}_${version_name}.tar"
    }]
}
```

   其中，

   - include_files中的元素为需要打包的文件在云手机中的绝对路径。

- object_path中${package_name}仅支持com.cph.config、com.cph.config.level1、com.cph.config.level2中的一种，${ version_name }用于识别配置包版本，可自行定义。

3. 参考**推送应用tar包至服务器共享空间**，将配置文件tar包推送到目标云手机服务器。

   调用示例：

   ```
   POST https://${CPH Endpoint}/v1/${projectId}/cloud-phone/phones/share-apps
   Header:
   Content-Type: application/json
   X-Auth-Token: ${token}
   Body:
   {
      "package_name": "${package_name}",
      "bucket_name": "${bucket_name}",
      "object_path": "apk/${package_name}_${version_name}.tar",
      "pre_install_app": 1,
      "server_ids": [
         "${server_id1}",
         "${server_id2}"
      ]
   }
   ```

   推送时需将"pre_install_app"参数设置为1。

4. 重置或重启云手机，自动部署配置文件包中的文件到云手机对应位置。

须知

1. 配置文件包部署到云手机会占用云手机存储空间，建议配置文件不要过大。
2. 建议仅使用一个配置文件包完成业务部署，将所需的文件都打包在com.cph.config包中，并且文件放置在云手机固定目录，方便文件统一管理。
3. 如果当前支持的三个配置文件包均需预置，云手机在重置或重启时，会按照com.cph.config、com.cph.config.level1、com.cph.config.level2的顺序依次部署，如果有相同文件，则会依次覆盖。

2.3.3.2 文件按需部署

业务场景示例

服务器上只有某些云手机需要部署文件，云手机正在运行状态，直接部署文件到云手机。

操作步骤

1. 参考**文件预置**的**1**和**2**，将配置文件打成配置文件tar包并上传到OBS桶。
2. 参考**推送应用tar包至服务器共享空间**，将配置文件tar包推送到目标云手机服务器。

   推送时将"package_name"配置为支持的配置文件包包名，将"pre_install_app"配置为0。

3. 参考**appctrl命令**，在云手机中调用appctrl applyConfig {配置文件包名}命令。

2.3.3.3 已部署文件更新

业务场景示例

该服务器上云手机需要更新已部署过的文件。

如：/data/local/tmp/test.sh脚本有更新，云手机需要部署最新的脚本。

操作步骤

1. 参考**文件预置**的**1**、**2**、**3**，将新配置文件打成tar包并推送到服务器共享空间。
2. 更新文件。

   方式一：重启云手机，自动更新文件。

   方式二：执行appctrl applyConfig命令更新文件（参考**appctrl命令**）。

须知

1. 如果配置文件和自身业务有关联，如：配置文件与运行应用的版本有相互依赖关系，当应用更新时需要对配置文件进行更新。
2. 由于重置云手机只会部署最后一次推送的配置文件包中的文件，更新配置文件包时，需要将所有期望生效的配置文件全量打包。

   例如：已部署的配置文件包中有1.txt、2.txt，当前需要更新2.txt。您需要将1.txt和已更新的2.txt全量打包并推送。如果仅打包更新后的2.txt并推送，重置仅会部署2.txt。

2.3.3.4 取消文件预置

业务场景示例

服务器上的所有云手机重置时均不再需要部署文件。

操作步骤

1. 参考**文件预置**的**1**、**2**、**3**，将配置文件tar包推送到目标云手机服务器。

   推送时将**3**中的"pre_install_app"参数设置为0。
2. 重置云手机，该包中文件将不再部署。

2.3.3.5 使用多个配置文件包部署文件

业务场景示例一

所有服务器需要预置公共文件，某些服务器需要预置特殊配置文件，处理不同业务。

例如：1.txt、2.txt为公共配置文件，需要预置到所有服务器的云手机中。同时需要预置特定配置文件a.txt，仅对服务器组A生效；特定配置文件b.txt，仅对服务器组B生效。

此时需将1.txt、2.txt打包，并以com.cph.config为包名推送到所有服务器；

将a.txt单独打包，并以com.cph.config.level1为包名推送到服务器组A的服务器；

将b.txt单独打包，并以com.cph.config.level1为包名推送到服务器组B的服务器；

如果需要更新1.txt、2.txt，则更新该文件后重新打包，并以com.cph.config为包名推送的服务器。

如果需要更新a.txt/b.txt，则更新该文件后重新打包，并以com.cph.config.level1为包名推送到对应服务器。

操作步骤

1. 参考**文件预置**的**1**、**2**、**3**，将配置文件tar包推送到目标云手机服务器。
   推送时将**3**中的"pre_install_app"参数设置置为1。
2. 参考**文件预置**的**4**完成文件部署到云手机。

业务场景示例二

服务器上的所有云手机已预置公共文件，其中某些云手机需要额外增加特定配置文件。

例如：服务器中所有云手机已预置公共配置文件1.txt、2.txt，某些云手机需要额外部署特定配置文件3.txt、4.txt。

此时需将1.txt、2.txt打包，并以com.cph.config为包名推送到服务器；将特定配置文件3.txt、4.txt打包，并以com.cph.config.level1为包名推送到服务器。

操作步骤：

1. 参考**文件按需部署**的**1**、**2**，将配置文件tar包推送到目标云手机服务器。

推送com.cpm.config.level1时，需将**2**中的"pre_install_app"云手机单独配置设置为0。

2. 参考**文件按需部署**的**3**完成文件部署到云手机。

2.4 appctrl 命令

2.4.1 appctrl 命令的执行方法

可通过以下三种方式在云手机中执行appctrl命令：

1. 通过ADB连接云手机执行命令，请参考**一键式ADB连接** 。
2. 调用API接口在云手机中执行命令，请参考**执行异步adb命令** 。
3. 云手机中的业务程序或脚本直接调用appctrl命令。

   说明

   需确保业务程序或脚本具备root用户权限。

2.4.2 appctrl install 安装应用

使用场景

安装应用到云手机。

前置条件

应用的tar包已经推送到云手机服务器。

使用指导

appctrl install {包名}

例如：安装应用：appctrl install com.huawei.xxxx

2.4.3 appctrl start 安装并启动应用

使用场景

安装应用到云手机，并启动应用。

前置条件

应用的tar包已经推送到云手机服务器。

使用指导

appctrl start {包名}

例如：安装并启动应用：appctrl start com.huawei.xxxx

须知

对于服务类应用，没有activity启动入口，须先使用appctrl install命令先安装应用，再使用am命令启动前后台服务。

2.4.4 appctrl uninstall 卸载应用

使用场景

云手机中不再使用某款共享应用时，从云手机中卸载该应用。

使用指导

在云手机中执行appctrl uninstall {包名}

例如：卸载应用：appctrl uninstall com.huawei.xxxx

2.4.5 appctrl clear 批量卸载应用

使用场景

批量卸载所有非预安装的共享应用。

使用指导

调用appctrl clear

例如：云手机中有2个非预安装共享应用，执行appctrl clear，可将2个非预安装应用批量卸载。

2.4.6 appctrl applyConfig 部署文件

使用场景

直接部署文件至云手机。

前置条件

- 配置文件tar包已经推送到云手机服务器，并且为固定支持的包名。
- 仅支持包名为com.cph.config、com.cph.config.level1、com.cph.config.level2。

使用指导

调用appctrl applyConfig {配置文件包名}

例如：appctrl applyConfig com.cph.config.level2

3 批量安装应用至云手机

在一台云手机内安装APP后，可以通过调用接口的方式将此APP共享安装至多台云手机，省去重复安装的时间。

说明

假设这台安装了APP的云手机为种子云手机。

约束与限制

- 该方案仅适用于**非**qemu规格的云手机，即实例规格名称中**不包含**“qemu”字样。

  图 3-1 实例规格

- 种子云手机必须为一台未进行过任何操作的云手机。如果云手机上存在历史操作，请**重置云手机**。

操作步骤

1. 登录管理控制台，选择“存储 > 对象存储服务 OBS”，参考“**批量控制**”创建一个用来存放文件的OBS桶，并授权CPH服务操作OBS桶权限。

   注意：为了成功上传文件到桶，请确保已授权CPH对应桶的读写权限。
2. 使用ADB方式连接种子云手机，并为其安装需要的APP。

   详细指导请参考“**如何在单台云手机中安装APP?** ”。
3. 通过调用接口的方式，导出种子云手机数据并打包上传至步骤**1**中的OBS桶。

   curl命令如下：

   ```
   curl -i -k -X POST "https://${CPH Endpoint}/v1/${projectId}/cloud-phone/phones/batch-storage" -H "Content-Type: application/json" -H "X-Auth-Token: $token" -d '
   ```

```
{
   "storage_infos": [
      {
         "phone_id": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
         "include_files": [
            "/data/app/${package-name}-1",
            "/data/data/${package-name}",
            "/data/media/0/Android/data/${package-name}"
         ],
         "bucket_name": "${bucket_name}",
         "object_path": "${your_dir}/${package-name}.tar"
      }
   ]
}'
```

其中，

- phone_id为种子云手机的ID。
- bucket_name为导出数据所存储的OBS桶名。
- object_path为导出数据所存储的OBS路径。

⚠ 注意

如果想全量打包云手机APP数据，必须要包含以下3个路径：

- /data/app/${package-name}-1

  其中${package-name}后面可能接的不是-1，需要根据云手机中的实际情况填写。
- /data/data/${package-name}
- /data/media/0/Android/data/${package-name}

示例：

```
curl -i -k -X POST "https://${CPH Endpoint}/v1/${projectId}/cloud-phone/phones/batch-storage" -H
"Content-Type: application/json" -H "X-Auth-Token: $token" -d '
{
   "storage_infos": [
      {
         "phone_id": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
         "include_files": [
            "/data/app/com.taptap-1",
            "/data/data/com.taptap",
            "/data/media/0/Android/data/com.taptap"
         ],
         "bucket_name": "your-bucket-name",
         "object_path": "your/dir/taptap.tar"
      }
   ]
}'
```

4. 进入步骤1中的OBS桶，在“对象”中查看并确认步骤3中打包上传的文件是否已全部上传成功。确认成功后执行下一步。
5. 此步骤需区分“存储1.0”和“存储2.0”服务器
   - 存储2.0服务器（推荐）

     通过调用接口的方式，将OBS桶内的文件推送至服务器。

     curl命令示例如下：

     ```
     curl -i -k -X POST "https://${CPH Endpoint}/v1/${projectId}/cloud-phone/phones/share-apps" -H
     "Content-Type: application/json" -H "X-Auth-Token: $token" -d '
     {
        "package_name": "com.miniteck.miniworld",
     ```



```
    "bucket_name": "your-bucket-name",
    "object_path": "your/dir/miniworld.tar",
    "server_ids": ["1678567b8bab40f93711234cb8","1234567b8bab40ffb711234cb"]
}'
```

其中，

- bucket_name、object_path分别对应步骤**3**中的内容。
- server_ids为接受文件推送的服务器ID列表。指定多个服务器ID，可以实现多台服务器上的云手机均安装APP的诉求。

说明

该接口的更多说明，请参考”**推送共享应用**”。

– 存储1.0服务器

通过调用接口的方式，将OBS桶内的文件推送至服务器的共享存储目录。

curl命令示例如下：

```
curl -i -k -X POST "https://${CPH Endpoint}/v1/${projectId}/cloud-phone/phones/share-files" -H "Content-Type: application/json" -H "X-Auth-Token: $token" -d '
{
    "bucket_name": "your-bucket-name",
    "object_path": "your/dir/taptap.tar",
    "server_ids": ["1678567b8bab40f93711234cb8","1234567b8bab40ffb711234cb"]
}'
```

其中，

- bucket_name、object_path分别对应步骤**3**中的内容。
- server_ids为接受文件推送的服务器ID列表。指定多个服务器ID，可以实现多台服务器上的云手机均安装APP的诉求。

说明

该接口的更多说明，请参考“**推送共享存储文件**”。

6. 此步骤需区分“存储1.0”和“存储2.0”服务器

– 存储2.0的服务器（推荐）

在云手机中执行安装应用命令：appctrl install package_name，其中的包名对应**第五步**中的package_name，例如：com.miniteck.miniworld。

– 存储1.0的服务器

进入云手机控制台，单击已接受文件推送的服务器名称，在“实例管理”中，勾选所有需要安装以上APP的云手机实例，单击“重置”。

注意

此步骤为重置，并非重启。

执行结果

存储1.0的机器在重置手机成功后均已安装APP；存储2.0的机器无需重置，后续需要时直接调用 appctrl start package_name来启动应用。

4 修改云手机的 GPS 定位信息

云手机的GPS定位信息是模拟GPS卫星获取的经纬度值，数值单位为度，使用十进制小数形式表示，遵循国际惯例，东经为正，西经为负，北纬为正，南纬为负。本文指导您如何修改云手机的GPS定位信息。

前提条件

已购买并通过ADB方式登录云手机。详细操作请参见“**购买云手机服务器**”。

操作步骤

假设需修改的位置是东经114.055939度，北纬22.657501度。

在本地设备的ADB安装目录中，执行如下命令，修改GPS定位信息。

adb -s 127.0.0.1:*本地空闲端口* shell "echo 'longitude=114.055939:latitude=22.657501' > /data/gps/fifo"

说明

其中，*本地空闲端口*是建立SSH隧道时所使用的本地空闲端口。

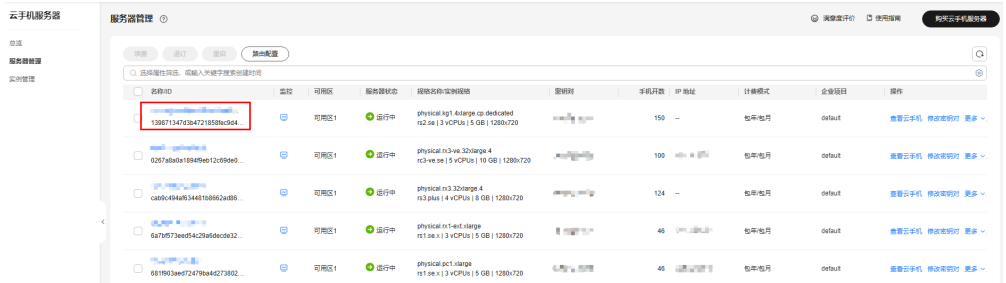
命令执行完立即生效，可使用地图类软件查看修改结果。例如：使用某社交软件发布动态，添加位置时可以看到GPS定位信息。

5 使用云手机摄像头

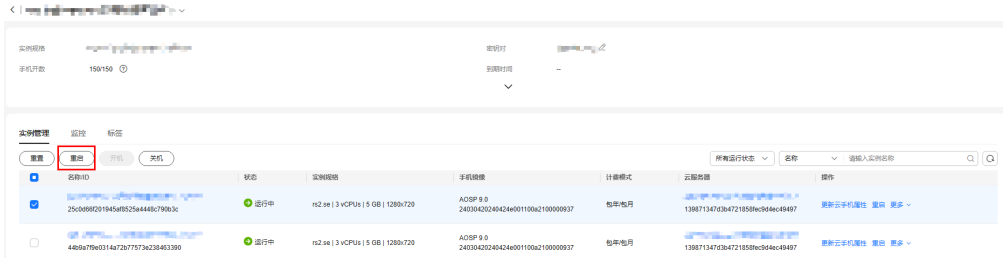
步骤 1：替换手机镜像

1. 查看云手机镜像**最新动态**，选择一个2020年10月9日（包含）之后的镜像，复制镜像ID。
2. 登录华为云控制台，切换到您的资源所在region，选择云手机服务器服务。

3. 单击服务器管理，查看云手机服务器列表。



4. 进入云手机服务器详情页，选择其中一台云手机，单击“重启”按钮。



5. 勾选更新手机镜像复选框，然后填入**第一步**所选的镜像ID。

6. 单击“确定”，完成单台云手机更新镜像操作。

步骤 2：上传图片到手机

上传一张图片到云手机的“/data/local/tmp/”目录下。您有两种方式上传图片，此处以“/path/to/local”目录下的“pic.jpeg”为例。

- **方式一：通过adb push命令推送图片**
  首先通过adb连接到云手机，然后执行以下命令：
  adb push <*/path/to/local/pic.jpeg*> /data/local/tmp/*pic.jpeg*
  adb shell chmod 644 /data/local/tmp/*pic.jpeg*
- **方式二：调用云手机api接口推送图片**
  请参考**ADB命令推送文件**来推送图片文件。

须知

- 上传的图片尺寸保证比例为 480(宽) * 640(高)，比例不为480*640时，图片在摄像头中可能会被缩放。
- 图片只支持jpeg和png格式，请不要用其他格式的图片。并且路径一定要是“/data/local/tmp/”目录下。
- 图片权限至少为644(rw-r--r--)。

步骤 3：设置手机属性

您有两种方式设置手机属性。

方式一：adb连接到云手机，然后执行adb命令

adb shell setprop com.cph.cam_local_pic_path /data/local/tmp/*pic.jpeg*

使用此这种方式，重启手机后属性失效。

方式二：调用云手机api接口设置属性

参考**更新云手机属性**来设置，将”com.cph.cam_local_pic_path”:”/data/local/tmp/pic.jpeg”属性设置到手机中。属性将被持久化，重启手机属性依然保留。

步骤 4：测试摄像头

安装任意一款需要调用摄像头的APP，打开APP查看取景框是否成功显示您设置的图片。

说明

当前云手机只支持后置摄像头。

6 通过 STF 批量管理云手机

操作场景

STF，全称Smartphone Test Farm，一个开源的web架构应用，用于移动设备管理控制。本质上通过浏览器控制和管理Android设备，实现真正意义的云端使用、调试和测试。本小节通过在一台ECS上部署STF的相关组件，实现快速批量管理云手机的功能。

约束与限制

- STF实测支持同时管理约160台云手机，更大规模的接入管理需要结合业务进行二次开发。
- STF可靠运行依赖稳定的网络环境，网络状态不佳时云手机的操作时延会显著增大。

前提条件

- 已购买一台绑定EIP的云手机服务器。
- 已购买一台绑定EIP的弹性云服务器。

说明

云手机服务器和弹性云服务器参考以下规格，具体规格可结合业务场景决定。

- 云手机服务器规格为：physical.kg1.4xlarge.cp | kg1.cp.c60.d16SSD.e1v1
- 弹性云服务器规格为：通用计算型 | s6.large.2 | 2vCPUs | 4GiB | Ubuntu 18.04 server 64bit(40GB)

操作步骤

在弹性云服务器上部署STF依赖的相关组件，并借助ADB工具连接云手机，最后通过浏览器访问STF的地址，实现云手机的批量管理。

1. 安装ADB，并验证安装结果。

```
sudo apt install android-tools-adb android-tools-fastboot
adb --version
```

正确回显版本即安装成功。

图 6-1 ADB 安装成功

```
root@ecs-stf:~# adb --version
Android Debug Bridge version 1.0.39
Version 1:8.1.0+r23-5~18.04
Installed as /usr/lib/android-sdk/platform-tools/adb
```

2. 更新源，然后安装RethinkDB，用于STF数据存储。

```
source /etc/lsb-release && echo "deb https://download.rethinkdb.com/repository/ubuntu-$DISTRIB_CODENAME $DISTRIB_CODENAME main" | sudo tee /etc/apt/sources.list.d/rethinkdb.list
wget -qO- https://download.rethinkdb.com/repository/raw/pubkey.gpg | sudo apt-key add -
sudo apt-get update
sudo apt-get install rethinkdb
rethinkdb -v
```

正确回显版本即安装成功。

图 6-2 RethinkDB 安装成功

```
root@ecs-stf:~# rethinkdb -v
rethinkdb 2.4.1~0bionic (CLANG 6.0.0 (tags/RELEASE_600/final))
```

RethinkDB官网支持x86架构，对于arm架构仅为实验性支持。

3. 安装ZeroMQ，用于消息传递。

```
sudo apt-get install libzmq3-dev
```

图 6-3 ZeroMQ 安装成功

```
root@ecs-stf:~# sudo apt-get install libzmq3-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libnorm1 libpgm-5.2-0 libsodium23 libzmq5
The following NEW packages will be installed:
  libnorm1 libpgm-5.2-0 libsodium23 libzmq3-dev libzmq5
0 upgraded, 5 newly installed, 0 to remove and 96 not upgraded.
Need to get 1,145 kB of archives.
After this operation, 4,150 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://repo.huaweicloud.com/ubuntu bionic/universe amd64 libnorm1 amd64 1.5r6+dfsg1-6 [224 kB]
Get:2 http://repo.huaweicloud.com/ubuntu bionic/universe amd64 libpgm-5.2-0 amd64 5.2.122~dfsg-2 [157 kB]
Get:3 http://repo.huaweicloud.com/ubuntu bionic/main amd64 libsodium23 amd64 1.0.16-2 [143 kB]
Get:4 http://repo.huaweicloud.com/ubuntu bionic-updates/universe amd64 libzmq5 amd64 4.2.5-1ubuntu0.2 [221 kB]
Get:5 http://repo.huaweicloud.com/ubuntu bionic-updates/universe amd64 libzmq3-dev amd64 4.2.5-1ubuntu0.2 [400 kB]
Fetched 1,145 kB in 0s (10.3 MB/s)
Selecting previously unselected package libnorm1:amd64.
(Reading database ... 111853 files and directories currently installed.)
Preparing to unpack .../libnorm1_1.5r6+dfsg1-6_amd64.deb ...
Unpacking libnorm1:amd64 (1.5r6+dfsg1-6) ...
Selecting previously unselected package libpgm-5.2-0:amd64.
Preparing to unpack .../libpgm-5.2-0_5.2.122~dfsg-2_amd64.deb ...
Unpacking libpgm-5.2-0:amd64 (5.2.122~dfsg-2) ...
Selecting previously unselected package libsodium23:amd64.
Preparing to unpack .../libsodium23_1.0.16-2_amd64.deb ...
Unpacking libsodium23:amd64 (1.0.16-2) ...
Selecting previously unselected package libzmq5:amd64.
Preparing to unpack .../libzmq5_4.2.5-1ubuntu0.2_amd64.deb ...
Unpacking libzmq5:amd64 (4.2.5-1ubuntu0.2) ...
Selecting previously unselected package libzmq3-dev:amd64.
Preparing to unpack .../libzmq3-dev_4.2.5-1ubuntu0.2_amd64.deb ...
Unpacking libzmq3-dev:amd64 (4.2.5-1ubuntu0.2) ...
Setting up libpgm-5.2-0:amd64 (5.2.122~dfsg-2) ...
Setting up libnorm1:amd64 (1.5r6+dfsg1-6) ...
Setting up libsodium23:amd64 (1.0.16-2) ...
Setting up libzmq5:amd64 (4.2.5-1ubuntu0.2) ...
Setting up libzmq3-dev:amd64 (4.2.5-1ubuntu0.2) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
```

4. 安装Protocol Buffers，作为消息传递的数据格式。

```
sudo apt-get install libprotobuf-dev protobuf-compiler
protoc --version
```

正确回显版本即安装成功。

图 6-4 Protocol Buffers 安装成功

```
root@ecs-stf:~# protoc --version
libprotoc 3.0.0
```

5. 安装GraphicsMagick，用于处理图像的读取、写入和操作。

```
sudo apt-get install graphicsmagick
gm version
```

正确回显版本即安装成功。

图 6-5 GraphicsMagick 安装成功

```
root@ecs-stf:~# gm version
GraphicsMagick 1.3.28 2018-01-20 Q16 http://www.GraphicsMagick.org/
Copyright (C) 2002-2018 GraphicsMagick Group.
Additional copyrights and licenses apply to this software.
See http://www.GraphicsMagick.org/www/Copyright.html for details.
```

6. 安装pkg-config，用于编译Nodejs第三方库。

```
sudo apt-get install pkg-config
pkg-config --version
```

正确回显版本即安装成功。

图 6-6 pkg-config 安装成功

```
root@ecs-stf:~# pkg-config --version
0.29.1
```

7. 安装yasm，用于编译STF的依赖库。

```
sudo apt-get install yasm
yasm --version
```

正确回显版本即安装成功。

图 6-7 yasm 安装成功

```
root@ecs-stf:~# yasm --version
yasm 1.3.0
Compiled on Apr  3 2018.
Copyright (c) 2001-2014 Peter Johnson and other Yasm developers.
Run yasm --license for licensing overview and summary.
```

8. 安装Nodejs，用于部署STF运行环境。

```
##STF运行只支持Node.js 8.x版本
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash  -
sudo apt-get install -y nodejs
node -v
npm -v
```

正确回显版本即安装成功。

图 6-8 node 和 npm 安装成功

```
root@ecs-stf:~# node -v
v8.17.0
root@ecs-stf:~# npm -v
6.13.4
```

9. 安装STF。

```
sudo npm install -g cnpm --registry=https://registry.npm.taobao.org
sudo cnpm install -g stf
stf -V
```

正确回显版本即安装成功。

图 6-9 STF 安装成功

```
root@ecs-stf:~# stf -V
3.4.1
```

10. 检查STF依赖环境是否满足。

```
stf doctor
```

正确回显各组件版本即满足。

图 6-10 STF 检测启动环境

```
root@ecs-stf:~# stf doctor
2021-08-18T01:47:35.484Z INF/cli:doctor 20873 [*] OS Arch: x64
2021-08-18T01:47:35.486Z INF/cli:doctor 20873 [*] OS Platform: linux
2021-08-18T01:47:35.486Z INF/cli:doctor 20873 [*] OS Platform: 4.15.0-136-generic
2021-08-18T01:47:35.486Z INF/cli:doctor 20873 [*] Using Node 8.17.0
2021-08-18T01:47:35.495Z INF/cli:doctor 20873 [*] Using ZeroMQ 4.2.5
2021-08-18T01:47:35.512Z INF/cli:doctor 20873 [*] Using RethinkDB 2.4.1~0bionic
2021-08-18T01:47:35.512Z INF/cli:doctor 20873 [*] Using GraphicsMagick 1.3.28
2021-08-18T01:47:35.512Z INF/cli:doctor 20873 [*] Using ProtoBuf 3.0.0
2021-08-18T01:47:35.513Z INF/cli:doctor 20873 [*] Using ADB 1.0.39
```

11. 基于ADB连接云手机实例，具体连接方式参考**ADB连接**。
12. 启动RethinkDB。

```
rethinkdb
```

回显如**图6-11**即启动成功。

图 6-11 启动 RethinkDB

```
root@ecs-stf:~# rethinkdb
Recursively removing directory /root/rethinkdb_data/tmp
Initializing directory /root/rethinkdb_data
Running rethinkdb 2.4.1~0bionic (CLANG 6.0.0 (tags/RELEASE_600/final))...
Running on Linux 4.15.0-136-generic x86_64
Loading data from directory /root/rethinkdb_data
Listening for intracluster connections on port 29015
Listening for client driver connections on port 28015
Listening for administrative HTTP connections on port 8080
Listening on cluster addresses: 127.0.0.1, ::1
Listening on driver addresses: 127.0.0.1, ::1
Listening on http addresses: 127.0.0.1, ::1
To fully expose RethinkDB on the network, bind to all addresses by running rethinkdb with the `--bind all` command line option.
Server ready, "ecs_stf_qnb" afd469a8-a055-43c0-bbf1-a1334d1de3c1
```

13. 基于local模式启动STF，并基于浏览器访问。

```
##下方请填写实际的弹性云服务器EIP地址
stf local --public-ip {EIP地址} --allow-remote
##访问方式
http://{EIP地址}:7100/
```


图 6-12 输入 STF 默认账户和密码

图 6-13 云手机实例

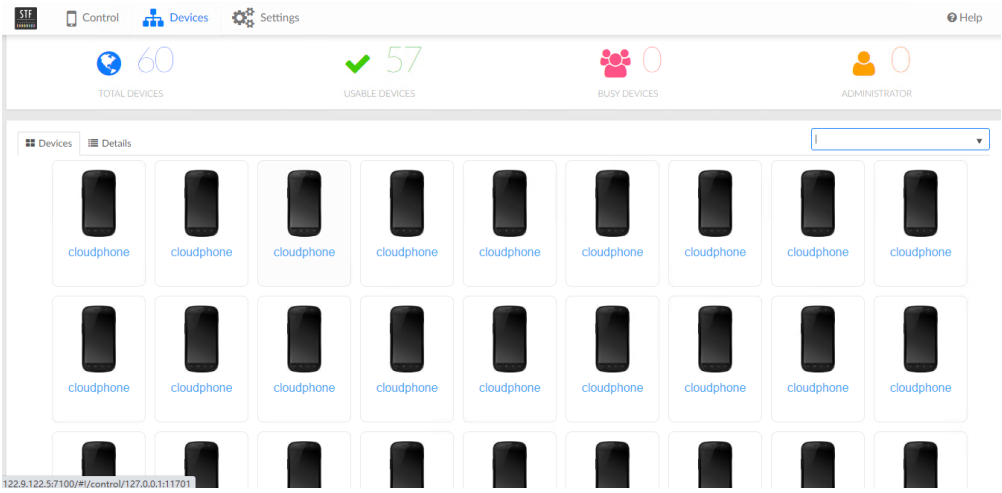
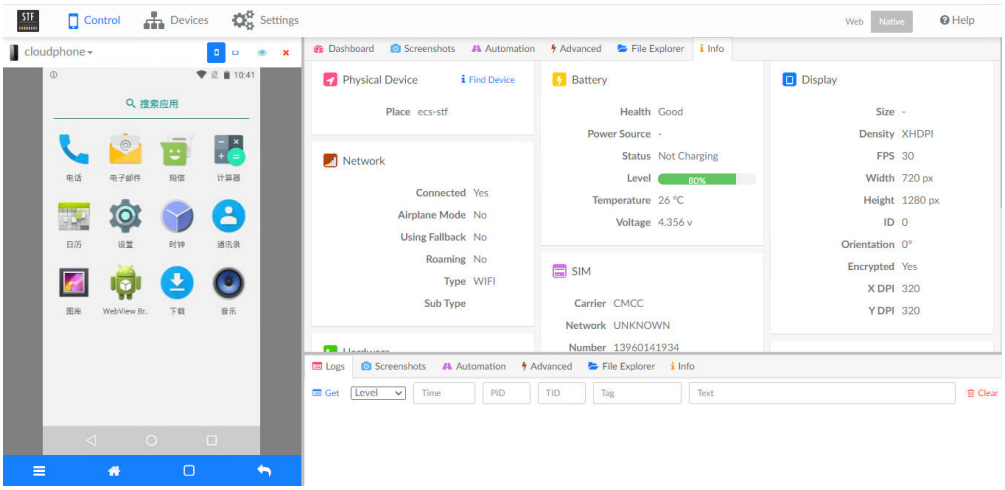
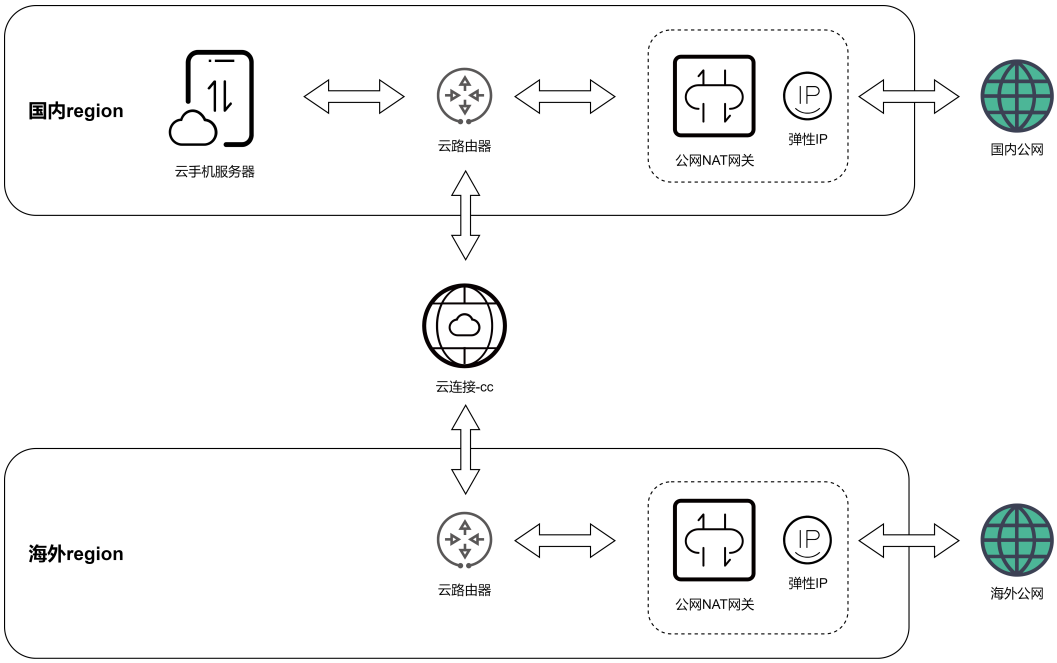


图 6-14 云手机控制界面



7 国内云手机服务器导流海外

导流方案整体结构如图：



约束与限制

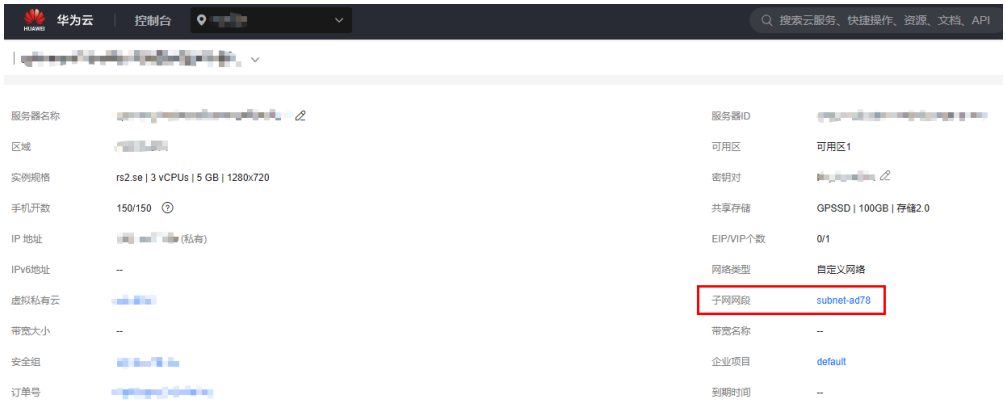
- 该方案仅适用于没有EIP的云手机服务器，即实例规格中EIP个数为0的规格（VIP个数不限）。

规格名称	vCPUs | 运行内存 | 机身存储	屏幕分辨率	手机开数	EIP/VIP个数
rx1.cp.c15.d46.e1v1	4 vCPUs | 16 GB | 46 GB	1280x720	15	1/1
rx1.cp.c60.d10.e0v60	2 vCPUs | 3.5 GB | 10 GB	1280x720	60	0/60
rx1.cp.c60.d10.e1v1	2 vCPUs | 3.5 GB | 10 GB	1280x720	60	1/1

操作步骤

1. 参考云连接“**跨境申请管理**”申请跨境专线资质。待审批通过后，执行后续步骤。

2. 登录管理控制台，进入云手机服务器管理，单击需要导流的云手机服务器，进入详情页面，找到“子网网段”。



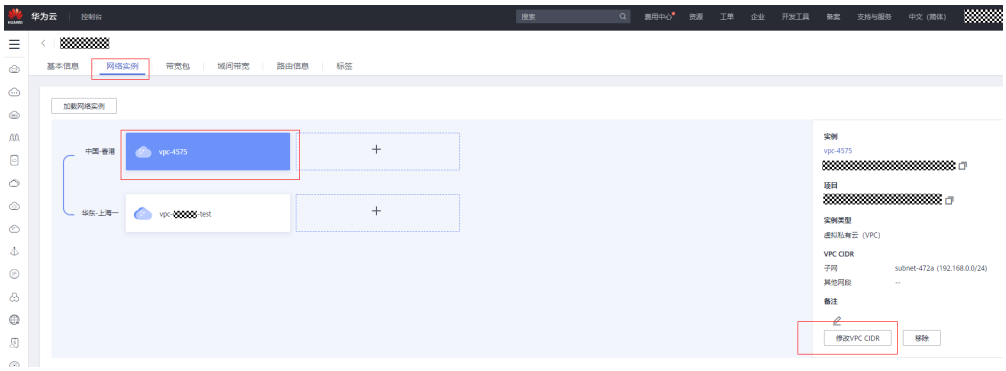
3. 单击子网名称，进入子网页面，找到“子网IPv4网段”，记录下云手机服务器所在的子网地址，例如192.168.0.0/24。

4. 选择“网络 > NAT网关”。
5. 切换到想要出海外公网的Region，例如“中国-香港”。
6. 参考NAT网关“**使用SNAT访问公网**”，购买EIP、公网NAT网关，并配置好SNAT规则、路由表增加目的地址0.0.0.0/0到NAT网关。

   其中配置SNAT规则时，使用场景选择“云专线/云连接”，并输入第**3**步记录的云手机服务器子网网段。

7. 选择“网络 > 云连接CC”。
8. 参考云连接CC“**跨区域同账号VPC互通**”，创建云连接实例，并配置加载云手机所在子网和第**6**步购买的NAT网关所在子网。购买“跨大区互通”带宽包，例如“中国大陆”到“亚太”，并绑定到云连接实例。示例如下图：

9. 在云连接“网络实例”页面选中已加载的海外vpc（示例中为“中国-香港”），然后单击“修改VPC CIDR”。



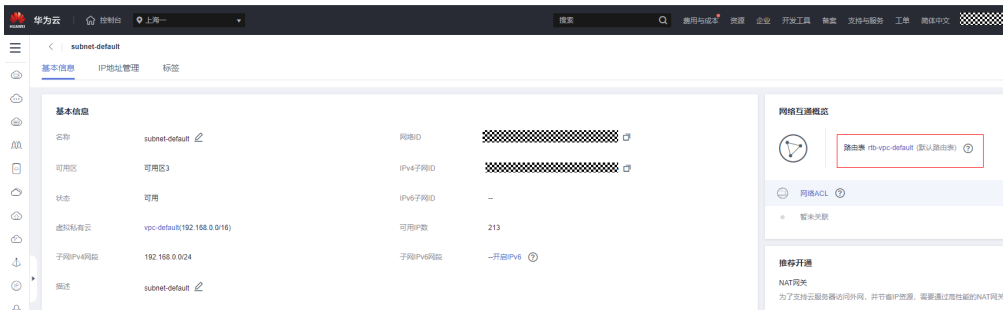
在弹出的对话框中选择“高级配置”，输入0.0.0.0/0，单击“添加”按钮，然后确定。

以上步骤完成后，国内云手机服务器的流量就已经全部导流到云连接，使用海外EIP出口。可以在云手机服务器内测试公网出口IP，验证配置是否成功。

如果不需要国内流量分流，以下步骤无需执行。

配置云手机服务器国内流量分流（可选）

1. 参考以上“操作步骤”第6步，在国内云手机所在Region，购买EIP、公网NAT网关，并配置好SNAT规则。此处有一个差异，不需要配置路由表目的地址0.0.0.0/0到NAT网关。
2. 参考以上“操作步骤”第2、3步，进入云手机服务器详情页面，找到“路由表”。



3. 单击路由表名称，进入路由表页面，单击“添加路由”。

4. 在弹出的对话框中，目的地址输入需要分流到国内的IP地址或网段，下一跳类型选择“NAT网关”，下一跳选择第1步购买的公网NAT网关，然后确定。

5. 如果有其它IP地址或网段需要分流，重复第4步添加。

以上步骤完成后，从云手机服务器内访问配置了分流的IP地址时，流量会从国内的EIP出口，其它流量则会导流到云连接从海外EIP出口。

8 委托 CPH 操作 OBS 桶

管理员可以通过IAM创建自定义策略，给OBS桶授予自定义策略来进行精细的访问控制，并将此策略委托给CPH服务，以完成云手机数据备份恢复、应用安装等功能。

操作步骤

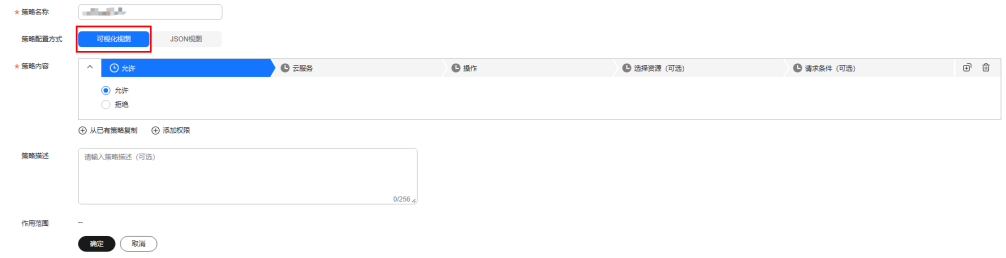
- 创建自定义策略

1. 登录管理控制台。
2. 在服务列表页，选择“管理与监管 > 统一身份认证服务 IAM”。
3. 在统一身份认证服务，左侧导航窗格中，选择“权限管理>权限”页签，单击右上方的“创建自定义策略”。

图 8-1 创建自定义策略

4. 输入“策略名称”。
   “策略配置方式”选择“可视化视图”。

图 8-2 可视化视图



5. 在“策略内容”下配置策略。
   a. 选择“允许”。
   b. 单击“云服务”，选择“对象存储服务 OBS”。

图 8-3 配置策略

c. 选择“操作”，根据需求勾选授权项。关于授权项请参考**对象相关授权项**。
委托CPH操作OBS桶，至少指定以下四项操作：
"obs:object:GetObject",
"obs:object:PutObject",
"obs:object:PutObjectVersionAcl",
"obs:object:PutObjectAcl"

图 8-4 勾选授权项-01

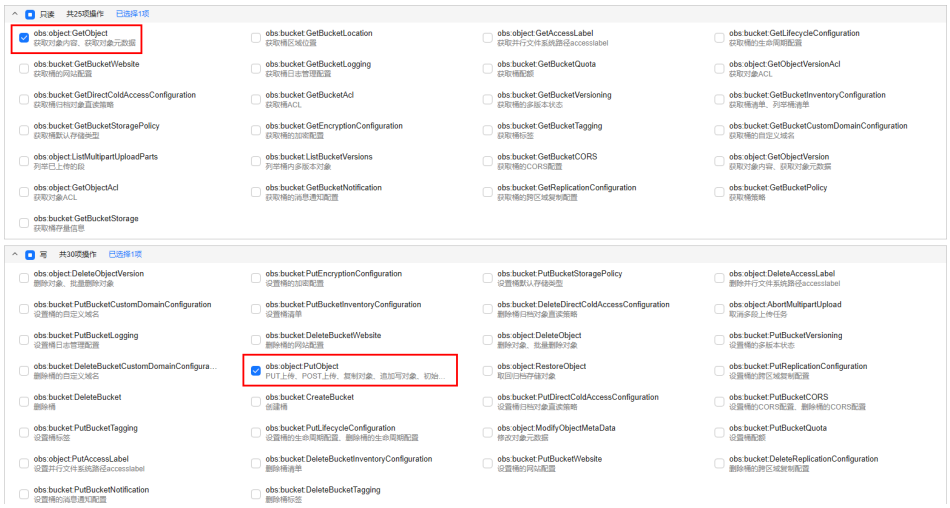
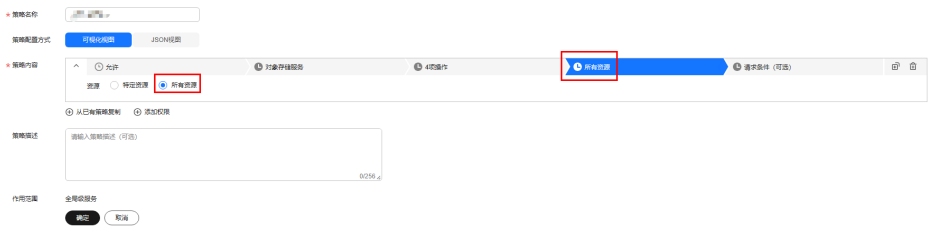


图 8-5 勾选授权项-02

d. 选择“所有资源”。如需选择“特定资源”，请参考**创建自定义策略**章节中关于“特定资源”的说明。

图 8-6 选择资源



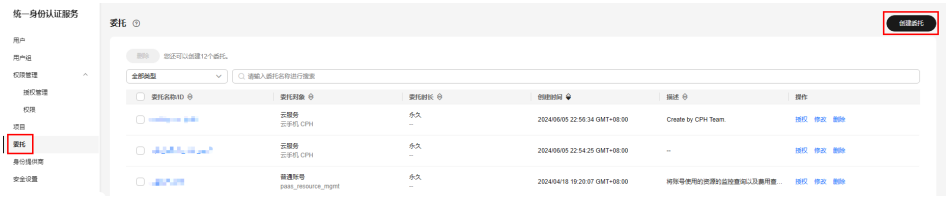
6. 单击“确定”，完成创建。

- 创建委托

1. 登录统一身份认证服务控制台。

   在统一身份认证服务的左侧导航窗格中，选择“委托”页签，单击“创建委托”。

图 8-7 委托



2. 参照下图填写参数内容，单击“下一步”。

图 8-8 创建委托

注意

委托名称必须填写“**cph_obs_agency**”。

3. 选中步骤**创建自定义策略**中创建的自定义策略，单击“下一步”。

图 8-9 选择策略

4. 选择“全局服务资源”，单击“确定”委托创建完成。

9 云手机 AOSP 版本切换

本文将指导您如何切换云手机AOSP镜像版本。

前提条件

仅手机镜像为AOSP7的服务器支持切换AOSP镜像版本。

注意事项

1. 版本切换是高风险操作，切换前请参考**导出云手机数据**备份用户数据。如需回退，重置原有镜像后参考**恢复云手机数据**恢复数据即可。
2. 版本切换时，Android版本号相关的属性值（如ro.build.version.release属性和ro.build.fingerprint中Android版本字段）会自动变更为目标镜像对应的版本号。

升级 AOSP 版本

方法一（保留用户数据）

云手机重启接口支持切换AOSP镜像版本，同时保留用户数据，操作详情请参考**重启云手机实例**。

注意：

- 重启接口支持将低版本镜像升级成高版本，但不支持将高版本镜像降级为低版本。
- 如果您没有保留用户数据的强烈诉求，推荐您使用方法二切换AOSP版本，应用不兼容的风险更小。

方法二（不保留用户数据）

云手机重置接口支持切换AOSP镜像版本，操作详情请参考**重置云手机实例**。

注意：

- 重置接口支持将低版本镜像升级成高版本，也支持将高版本镜像回退为低版本。

回退 AOSP 版本

因为应用兼容原因，只支持低版本切换高版本时保留用户数据。回退低版本只能使用不保留用户数据的重置接口，操作详情请参考**重置云手机实例**。

说明

若您通过重置云手机实例的方式更换低版本镜像，某些版本adb客户端需要先执行断开命令：adb disconnect ip:port，再重新执行连接命令：adb connect ip:port，才能看到手机画面。

