Ironic裸机布署

王凯峰

2017–03–15

Abstract

本文介绍Ironic布署裸机的现有流程、社区流程以及Windows的布署。

1 目前布署流程

1.1 Linux布署

我们实现的布署流程是在社区PXEAndIPMIToolDriver驱动的基础上修改实现的Partition布署类型。社区在Partition部署时，需要五个映像：两个布署映像deploy vmlinuz和deploy ramdisk，三个用户映像vmlinuz, ramdisk及Linux用户映像。

我们将Linux用户映像替换为原版Linux发行版ISO上传到Glance，其它四个映像实际并不使用，为了通过公共流程的检查，布署前仍然需要保证Glance上有五个映像。

布署前，首先注册Ironic节点，并填好映像信息，由于不使用vmlinuz和ramdisk，我们设置节点为本地启动类型。nova boot触发Ironic API的布署接口，进而进入ironic conductor的do_node_deploy。do_node_deploy主要执行两个操作，一个是布署准备，另一个是启动布署第一个阶段，社区的布署第二阶段由IPA触发，而我们的流程布署只有一个阶段。

社区的准备阶段包括TFTP配置，生成NBP(pxelinux.cfg)配置文件和布署映像下载，通过IPMI设置下次启动类型为PXE等工作，我们的流程去掉了布署映像下载，而改为在布署阶段时下载用户映像，并根据Glance上用户映像新增的tag字段，区分是Linux或是Windows。对于Linux，将ISO挂载到本地，拷出发行版的vmlinuz和initrd用作布署映像，放置到TFTP对应的位置，从kickstart模板生成ks文件，配置PXE菜单向内核传kickstart配置文件路径，并重启裸机。

裸机重启后通过PXE和conductor的DHCP服务进入Linux发行版安装，kickstart指定安装日志输出到conductor，conductor添加了定时任务来分析日志，并根据特定字段记录进度，当进度为100%时，手动触发状态机处理resume，done事件然后进入Active状态，布署至此完成。

1.2 Windows布署

Windows布署是在上述流程已经实现的基础上添加的，流程差异在于Windows安装ISO里没有可以布署用的映像，需要另外提供。

从Windows PE安装Windows是微软支持的布署方式，Windows PE即Windows Preboot Environment，角色与deploy ramdisk类似。与Linux一样，裸机通过PXE进入WinPE命令行环境，WinPE自动执行wpeinit初始化系统然后等待用户操作。我们可以手工挂载Conductor提供的samba服务路径，运行Windows安装映像的Setup.exe来启动Windows安装。

Windows安装过程要重新启动三到四次，这个阶段不能通过PXE启动，因此在布署Windows时，需要通过IPMI将裸机设为硬盘启动。Windows安装完成后，我们手工挂载Conductor的Samba服务路径，输出包含Windows安装完成的信息，Conductor监测到日志变化，发现安装完成后，执行与Linux安装相同的流程，触发节点进入Active状态，布署完成。

以上是手动安装Windows的过程，需要人工干预，下面就来讲讲怎么解决。

1.3 WinPE自动化

首先我们要制作WinPE映像，制作PE映像需要使用微软的AIK(Windows Automated Installation Kit)或者新的ADK(Assessment and Deployment Kit)工具。制作PE这两者差别不大，以AIK为例：

1. copype拷贝相关文件

   ```
   copype.cmd arch dest
   ```

   arch 指架构，可以指定x86, amd64等，可查看copype.cmd的脚本，这里以amd64为例。dest指定目标路径，copype.cmd将相关文件拷贝至该路径，下面以C盘winpe_amd64目录为例。

2. 拷贝winpe.wim

   ```
   copy "C:\Program Files\Windows AIK\Tools\PETools\amd64\winpe.wim"
        C:\winpe_amd64\ISO\Sources\Boot.wim
   ```

   实际上copype.cmd已经将wim拷贝到了目标路径的根目录，直接移动到ISO/sources就可以了。

   ```
   copy "C:\Program Files\Windows AIK\Tools\amd64\Imagex.exe" C:\
       winpe_amd64\ISO\
   ```

   这个文件是用来操作wim的，如果布署过程不需要用可以不拷贝。imagex和dism都可以操作wim文件，用法稍有不同，ADK所带的dism工具功能更多。

3. 用oscdimg生成PE ISO

   ```
   oscdimg -n -bc:\winpe_amd64\etfsboot.com c:\winpe_amd64\ISO c:\
       winpe_amd64.iso
   ```

生成winpe_amd64.iso文件

这样就制成了不经修改的PE，可以上传到Glance作为deploy ramdisk使用，类型指定为iso。memdisk可以从linux下取，比如/usr/share/syslinux/memdisk，用作deploy kernel。

但是不经定制的PE启动后，需要人工干预以启动Windows安装，无法自动化，主要有三个问题：

1. 引导时的“按任意键从DVD启动”的提示
2. 自带驱动若不支持物理网卡则无法连通Samba服务器来访问Windows安装文件
3. 进入PE环境后需要人工输入命令

这三个问题都需要修改PE镜像中的boot.wim，需要用dism工具先将其挂载到目录，完成修改后提交并卸载，修改wim完成后，需要重新制作PE ISO。命令用法如下：

```
Dism /Mount-Wim /WimFile:C:\winpe_amd64\ISO\sources\boot.wim /index:1 /
    MountDir:C:\winpe_amd64\mount
Dism /unmount-Wim /MountDir:C:\winPE_amd64\mount /Commit
```

修改步骤如下：

1. 挂载wim文件
2. 删掉boot.wim内的BOOT\bootfix.bin文件，就去除了启动时选择是否从光盘启动的交互。
3. 添加网卡驱动
4. 插入自动化脚本
5. 卸载wim文件
6. 生成ISO

1.3.1 添加网卡驱动

给PE添加驱动有两种方式，分为Offline和Online两种方式。Offline是指在制作PE镜像时，通过dism工具将驱动打进镜像，Online方式是在PE启动后，在命令行模式下通过drvload命令加载。目前采用的是Offline方式：

```
dism /image:C:\winPE_amd64\mount /add-driver /driver:C:\winPE_amd64\
    drivers /recurse /forceunsigned
```

查看驱动

```
dism /image:C:\winPE_amd64\mount /get-drivers /format:table > C:\wim-
    table.txt
```

采用Online方式有可能使PE的定制完全在Linux下完成，但官方提到drvload不支持重启，如果驱动有安装重启的需求，则无法通过该工具安装。

1.3.2 自动启动Windows安装

向PE插入定制操作有三种方式：

1. 定制 Winpeshl.ini
2. 修改startnet.cmd

```
%SYSTEMROOT%\System32\startnet.cmd
```

3. 使用 unattend 文件

具体可参考 https://technet.microsoft.com/en-us/library/cc766521(v=ws.10).aspx。目前使用第二种方式，例如：

```
wpeinit
net use z: \\192.168.1.1\install\x64
z:
setup.exe
```

PE启动后自动执行脚本，然后启动Windows安装。

1.3.3 Linux下的实现

因为ironic运行在Linux上，因此需要能在Linux定制wim文件，幸好Linux下确实有一个wimtools可以用来做部分定制工作，ubuntu可以直接搜索wimtools工具包，其它系统需要找别的源来下载，比如

```
http://li.nux.ro/download/nux/misc/el7/x86_64/wimtools-1.9.2-1.el7.nux.
    x86_64libwim15-1.9.2-1.el7.nux.x86_64
```

有了wimtools的支持，Linux下除了没有办法添加驱动外，其它两项可以实现。我们首先需要在Windows下用AIK/ADK工具生成一个模板WinPE，并添加所需要的驱动，主要是网卡和存储驱动。很多厂商驱动已经不支持AIK的WinPE版本了，建议用ADK生成WinPE并打入厂商驱动。

现在有了带驱动的模板WinPE，将其上传到Glance用作deploy ramdisk。与Linux流程不同的是，对于Windows布署，要恢复在prepare阶段去掉的deploy映像下载社区流程，其实这里也可以和Linux统一处理，在布署阶段去做，只是目前流程是这样实现了。

对于Windows类型的布署，在布署阶段，先要修改已下载到本地的deploy ramdisk（即WinPE ISO），然后执行定制操作，以映像名winpe_amd64.iso为例，如下是对应的bash操作，目前已经用Python实现并合到ironic里了：

```
mkdir media tmp_media tmp_wim
mount -o loop winpe_amd64.iso media/
cp -r media/* tmp_media/
# for file is read only
chmod +w tmp_media/sources/boot.wim
chmod a+w tmp_media/sources
# don't use wimmount as it's readonly mount
wimmountrw tmp_media/sources/boot.wim tmp_wim
# wpeinit is already included in startnet.cmd by default, so no need to
    append.
# \\ is translated to \ by bash, OK with python.
```

```
echo "net use z: \\192.168.1.1\install\x64" >> tmp_wim/Windows/System32
    /startnet.cmd
...add other commands if needed...
wimunmount --commit tmp_wim
# can use recursive option, but not python
chmod +w tmp_media/boot
chmod +w tmp_media/boot/bootfix.bin
# Remove Boot from CD ... wait message
rm tmp_media/boot/bootfix.bin
# create iso file, note that the boot image path following -b is
    relative path from CDImage root directory
mkisofs -r -o winpe_amd64_2.iso -b boot/etfsboot.com -no-emul-boot -
    boot-load-size 4 -J -l tmp_media/
```

写入startnet.cmd的内容，是通过模板定制的：

```
wpeinit
net use Z: \\{{ pxe_options.tftp_server }}\install
net use P: \\{{ pxe_options.tftp_server }}\log\remote-logs
Z:
cd {{ pxe_options.deployment_id }}
echo Windows Installation Start > P:\{{ pxe_options.pxe_ip }}.log
setup.exe /unattend:Z:\unattend\{{ pxe_options.deployment_id }}.xml
```

Conductor定时任务检测到"Windows Installation Start"后，将进度设为10%。

1.4 Unattend安装自动化

Windows安装过程中需要用户输入信息，如硬盘分区，序列号，用户创建等等，这些可以通过unattend文件实现应答，与Linux的kickstart一样。

应答文件unattend需要用AIK/ADK工具包的WSIM(Windows System Image Manager)图形化工具生成。首先要以管理员权限打开一个Windows安装镜像，生成编录文件，后续操作便可以脱离安装镜像完成。我们可以先生成一个模板XML，然后在布署时替换掉里面的信息来实现Windows自动安装。

在制作Windows Server 2012R2时，AIK工具不能生成编录文件，32位Windows 7运行ADK也会失败，最后在Windows Server 2012R2上安装ADK并生成编录文件。ADK对宿主机有要求，需要一个WEI 8.1的支持（尚不清楚是什么，也许需要Windows 8.1）。另WSIM工具还有32位和64位的限制，官方说明32位的WSIM可以生成32和64位安装镜像的编录，而64位WSIM只能生成64位安装镜像的编录，在此说明一下。

Windows启动安装后会自动查找安装映像sources目录下名为autounattend.xml的应答文件，也可以通过setup.exe的参数来明确指定：

```
setup /unattend:unattend.xml
```

使用默认应答文件名需要修改Windows安装镜像，我们采用后一种方式。

Windows安装分为几个阶段，称为Configuration Pass，在每个阶段，允许配置一些特定的选项以进行系统定制，这些选项从应答文件里获得，在Configuration Pass以外的过程则无法干预。可以参考AIK附带的帮助文档。我们主要定制的是以下几个内容：

- 提供驱动。为PE添加的驱动只在PE阶段安装Windows时使用，Windows安装到硬盘后会重启，便脱离了PE环境，如果不为Windows提供驱动，对于不支持的网卡，同样不能驱动，这样就不能回传日志。配置在PnpCustomizationsWinPE或PnpCustomizationsNoWinPE添加驱动路径，因为WinPE阶段添加的驱动会被传给NoWinPE阶段，为了能尽早使用网络，建议配置在WinPE阶段（这里的WinPE指的应该是安装镜像中的PE阶段），参见https://technet.microsoft.com/en-us/library/cc766485(v=ws.10).aspx。

- 用户创建与自动登录。为了在自动化安装完成后，需要支持自动登录，以便使用FirstLogonCommands执行一些命令，需要有用户。按照文档，如果设置用户为Administrator，则管理员帐户会被开启，可以不设置密码。若不设置密码，在AutoLogon配置为管理员帐户时也无需填写密码。这样可以按管理员帐户自动登录。

- 首次登录命令执行。通过FirstLogonCommands实现，Windows完成安装，在用户首次登录并未进入桌面环境时，调用配置的命令，可以在该处添加上传安装日志的操作，或者向Ironic汇报安装完成。该方法只需修改应答文件，利于脚本处理。Windows安装完成后，会调用SetupComplete.cmd，位置在：

  ```
  %WINDIR%\Setup\Scripts\SetupComplete.cmd
  ```

  但该方式需要修改Windows原版安装映像，所以优先考虑上面的方法。后来也试过修改SetupComplete.cmd但没有成功，Windows没有调用，这种黑盒子，唉，真真是，处处都是坑。还要注意类似echo, copy等命令属于shell命令，只能在cmd.exe下执行，要用cmd /c xxx的方式填写。

相比Linux，Windows就需要更多的信息来布署了：

- ImageName。现在发布的安装镜像都是几种版本合一的，如Windows 7 Ultimate, Windows 7 Standard会包含在一个wim里，需要指定安装哪一个镜像。也就是前面命令里出现的Index参数，可以用imagex/dism工具查看：

  ```
  imagex /info install.wim
  ```

- ProductKey。填充Microsoft-Windows-Setup\UserData\ProductKey\Key 字段。

分区信息目前是写死在unattend文件里的，Windows Server 2012R2需要至少两个区，一个最少350M的Boot区和一个普通分区，Windows 7无此要求。

Windows安装日志位于%WINDIR%的Panther目录中。setupact.log和setuperr.log是安装过程的完整日志和出错信息，子目录UnattendGC下的setupact.log和setuperr.log是执行unattend相关的日志。因为Windows安装会多次重启，无法像Linux一样用syslog返回日志，因此实时日志分析也没

有意义。我们是通过设置unattend首次登录命令，来挂载samba服务器，输出Windows Installation Complete字符串到以本机IP为文件名的文件。Conductor检测到安装完成，切换节点到Active状态。

2 社区布署流程

2.1 M版本PXE驱动布署代码流程

nova boot的请求最终通过ironic api传递到ironic conductor的manager.py的ConductorManager.do_node_deploy，它分给了task管理，实际还是传给该文件的do_node_deploy函数处理，它调用driver做两个主要的操作：driver.deploy.prepare和driver.deploy.deploy，对应到iscsi驱动就是ISCSIDeploy.prepare和ISCSIDeploy.deploy。prepare阶段主要是生成tftp配置，下载所需镜像，deploy则启动第一阶段的布署。

nova boot也携带了一些instance信息过来，还没有具体分析过。

prepare的具体调用流程为：

```
build_deploy_ramdisk_options
build_agent_options
driver.boot.prepare_ramdisk
    pxe_utils.dhcp_options_for_instance
    _get_deploy_image_info
    _build_pxe_config_options
    pxe_utils.create_pxe_config
    deploy_utils.set_boot_device
    _cache_ramdisk_kernel
```

build_deploy_ramdisk_options生成deploykey（随机字符），创建deploy_options并返回。

build_agent_options生成ipa-api_url，ipa-driver-name等配置信息到deploy_opts。这个信息被传到prepare_ramdisk。

prepare_ramdisk主要工作是准备好deploy ramdisk和deploy kernel，配置PXE启动环境，如设置tftp和dhcp配置等。iPXE走http，PXE走tftp。

dhcp_options_for_instance得到dhcp_factory.DHCPFactory单实例，并设置参数，参数为字典形式的形表：opt_name:xx，opt_value:xx。主要设置了bootfile名称，tftpserver的IP地址信息。目前dhcp只有neutron这个驱动。

_get_deploy_image_info先取driver_info里的deploy_kernel和deploy_ramdisk，并检查是否有无值的情况。然后构造pxe_info字典

```
deploy_kernel: (driverinfo, path)
deploy_ramdisk: (driverinfo, path)
```

它包含deploy_kernel和deploy_ramdisk的driver信息和绝对路径，但此时文件并不存在，路径形式为：

```
rootdir/uuid/deploy_kernel
rootdir/uuid/deploy_ramdisk
```

_build_pxe_config_options创建pxe_options，并将前面的deploy_opts加入。从配置文件加入pxe_append_params

pxe_utils.create_pxe_config通过预定义的配置模板，使用jinja2库填入参数生成配置文件，然后写入pxe_config_file_path，配置文件为rootdir/uuid/config，并创建软链接，链接文件名是MAC地址，位于rootdir/pxelinux.cfg目录下。

deploy_utils.set_boot_device通过IPMIManagement设置PXE启动方式。

_cache_ramdisk_kernel通用deploy_utils.fetch_images转给TFTPImageCache下载布署映像到指定位置。

prepare_ramdisk至此结束，然后进入deploy流程。

iscsi_deploy的具体调用流程为：

```
cache_instance_image
check_image_size
conductor.utils.node_power_action(REBOOT)
```

cache_instance_image下载用户映像，对应本地名字是disk，由InstanceImageCache来管理。

check_image_size检查映像大小，如果映像超过了分区大小，则抛异常退出流程。

接下来重启裸机，然后返回DEPLOYWAIT状态。DEPLOYWAIT状态反映在节点状态就是wait call-back状态。开始第二阶段布署时，节点状态会切换到deploying。conductor维护了一个状态机fsm，由TaskManager事件来驱动。

裸机重启后，按照此前设定的PXE方式启动，从Conductor要到布署映像，启动后IPA运行，向ironic api发送POST请求：

```
/vendor\_passthrough?method=pass\_deploy\_info
```

最终进入iscsi_deploy驱动的pass_deploy_info入口。这种驱动透传的方式，可能会逐步被agent方式取代。

pass_deploy_info流程比较简单：

```
continue_deploy
driver.boot.prepare_instance
finish_deploy
```

continue_deploy使用deploy_utils辅助写盘，会区分whole disk和非whole disk流程，whole disk流程使用deploy_disk_image写盘，非whole disk使用deploy_partition_image写盘。

deploy_utils主要是封装iscsi连接操作，_iscsi_setup_and_handle_errors提供iscsi建立和退出连接的环境管理器，具体写盘操作是ironic-lib项目提供的disk_utils实现。非whole disk流程由work_on_disk处理，whole disk流程是用populate_image。写盘完成后，调用destroy_images unlink用户映像并删掉PXE本地映像目录。

prepare_instance对于本地启动，清除PXE配置并设置为硬盘启动。对于PXE启动，更新DHCP配置，得到root uuid用于之后的网络启动，更新修改PXE配置。因为启动位置变化了，可能是指定PXE配置里的内核启动参数，如根文件系统的uuid之类的吧，然后设置启动方式为PXE。

finish_deploy完成收尾工作，它等3秒，调用notify与ipa通信，用的是socket发了一个'done'，若本地启动，则直接重启裸机。task.process_event('done') 驱动状态机进入active状态。

第二阶段的步骤主要是Conductor与IPA的交互，需要结合IPA代码来看。

2.2 O版本Agent驱动布署代码流程

PXE与Agent驱动的主要区别是，PXE通过iSCSI暴露硬盘，由Conductor来执行写盘，而Agent则主动下载映像写盘，不暴露硬盘。两个版本总的流程差别不大，走读O版本的目的主要是分析一下whole disk的流程和Agent的流程，因此差不多的就不再描述。

同M版本PXE布署流程一样，最终进入Conductor的do_node_deploy，这里先通过is_whole_disk_image判断了是否为全盘映像，它获取image property，看有没有kernel_id和ramdisk_id，如果没有的话，就把driver internal info里设置is_whole_disk_image=True并保存到节点数据库供后面的流程使用，但其实whole disk流程与partition流程差异很小。

iSCSIDeploy.validate调用PXEBoot.validate检查参数是否完备，如果是whole disk映像就不检查ramdisk和kernel了。检查通过后创建do_node_deploy任务。

do_node_deploy任务的具体调用链为：

```
ISCSIDeploy.prepare
    build_deploy_ramdisk_options
    build_agent_options
    PXEBoot.prepare_ramdisk
        pxe_utils.dhcp_options_for_instance
        _get_deploy_image_info
        _build_pxe_config_options
        pxe_utils.create_pxe_config
        deploy_utils.set_boot_device
        _cache_ramdisk_kernel
ISCSIDeploy.deploy
    cache_instance_image
    check_image_size
    conductor.utils.node_power_action(REBOOT)
```

与前面流程基本上是一样的。第二个阶段的步骤仍由IPA触发，不同的是Agent不走vendor_passthrough接口，而是/v1/lookup接口，query string带上MAC地址（或者加上节点uuid），这个API接口实现在api/controllers/ramdisk.py中的LookupController里。它读取节点相关的driver信息，存在node.links属性里，node是新创建的对象，然后填入一些属性，它并不是当前node对象的完整克隆，可能是出于安全的考虑，所以只写入了布署必要的信息。

lookup api没有直接触发conductor调用，推测是IPA从ironic api获得各类请求链接后，后面又发生一些交互，获取各类部署所需信息，就绪后再向ironic api发送heartbeat请求并携带uuid，api当然还是把请求转给了conductor，conductor转给驱动，然后进入iscsi_deploy的heartbeat接口，

但iscsi_deploy并没有实现heartbeat，它是继承自agent_base_vendor，通用的流程提取到父类实现了。

heartbeat会判断状态，因为还没有开始布署，所以会执行AgentDeployMixin的continue_deploy接口，这是个接口，所以流程又回到iscsi_deploy的continue_deploy。具体调用流程为：

```
do_agent_iscsi_deploy
    continue_deploy (iscsi_deploy)
prepare_instance_to_boot
reboot_and_finish_deploy
```

continue_deploy做实际的写盘操作，对于非whole disk使用deploy_ partition_image，whole disk则使用deploy_disk_image。写完后用destroy_images删除用户映像。两种写盘方式的区别在于：partition方式会对磁盘进行分区，分区信息可以通过configdrive参数传入（configdrive是放到swift的），然后调用populate_image写根分区，whole disk则不做分区操作，直接调用populate_image。

populate_image的逻辑也很简单，如果用户映像是raw格式，直接用dd写盘，其它格式就用qemu-img convert直接输入raw格式到iscsi设备。

prepare_instance_to_boot调用driver.boot.prepare_instance与之前流程类似，清理环境配置，然后设置启动方式。如果指定了本地启动，则安装bootloader，然后设置为本地启动。但这里会判断如果是whole disk就跳过bootloader安装。

reboot_and_finish_deploy先关闭裸机，与M版本不同，这里切换了网络：

```
task.driver.network.remove_provisioning_network(task)
task.driver.network.configure_tenant_networks(task)
```

然后控制裸机上电，节点状态切换为Active，布署结束。

这个改动有一个spec提到引入network provider，看起来是实现到一个布署网络和一个租户网络的程度，也就是说租户网络只有一个，可能是使用的网络翻转机制，目前社区正在做的multi-node multi-tenant应该就是为了完整支持网络功能的方案。网络部分还有待深入分析。

2.3 Whole Disk布署

从上面的流程来看，whole disk与partition的流程差异并不大，主要有两点：

- whole disk不写bootloader
- 写盘实现不同，whole disk方式没有分区操作，partition方式会先分区

最初认为whole disk与partition在启动方式上有不同，但目前从代码来看，两种方式既可以网络启动也可以本地启动，是可共存的配置。

但对于Windows来说，就只能采用本地启动的方式了。以布署Windows Server 2012 R2为例，先用KVM安装首个系统，然后将虚机的磁盘文件上传到Glance作为用户映像，获取布署映像最方便的是直接下载coreos的对

应版本，也上传到Glance，节点设为本地启动，然后nova boot启动布署。ironic判断没有kernel和ramdisk，走whole disk流程，将预装好的虚机映像直接写入裸机磁盘，并且不安装bootloader，布署完成后，裸机磁盘的分区布局与虚机是一样的，由于Windows有自己的bootloader，所以从本地启动是没有疑问的。