目 录

1 linux 系统启动过程

首先介绍 linux 系统启动过程中的关键文件：

1. 引导加载程序是系统加电后运行的第一段软件代码。PC 机中的引导加载程序由 BIOS 和位于硬盘 MBR 中的 OS BootLoader(比如，LILO 和 GRUB 等) 一起组成。BIOS 在完成硬件监测和资源分配后，将硬盘 MBR 中的 BootLoader 读到系统的 RAM 中，然后将控制器交给 OS BootLoader。

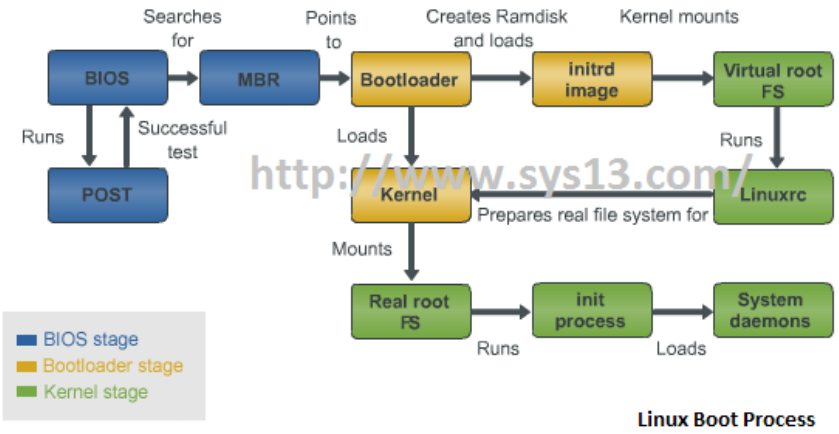
2. bootloader，负责将 kernel 和 ramdisk 从硬盘读到内存中，然后跳转到内核的入口去运行。

3. kernel 是 linux 的内核，包含最基本的程序。

4. ramdisk 是一种基于内存的虚拟文件系统，就好像你又有一个硬盘，你可以对它上面的文件添加修改删除等等，但一调电，就什么也没有了，无法保存。一般驱动程序放在这里面。

5. initrd 是 boot loader initialized RAM disk，是在系统初始化引导时候用的 ramdisk。也就是由启动加载其所初始化的 ramdisk 设备，它的作用是完善内核的模块机制，让内核的初始化流程更具弹性。内核以及 initrd 都由 bootloader 在系统启动后被加载到内存的指定位置，主要功能为按需加载模块以及按需改变根文件系统。initramfs 与 initrd 功能类似，是 initrd 的改进版本，改进了 initrd 大小不可变等缺点。

系统启动流程图如下所示：



1.1 initrd 的作用

如果可以把需要的功能全都编译到内核，那么只需要一个内核文件就可以了，不需要 initrd。

可是如果文件系统没有编译到内核中，而是以模块方式存放在内核中，而启动阶段的驱动模块放在这些文件系统上，那么内核是无法读取文件系统的。

所幸的是，Grub 是 filesystem sensitive 的，能够识别常见的文件系统，所以内核的文件就可以装入内存中，通用的安装流程如下：

1. 开机启动，BIOS 完成硬件检测和资源分配，选择操作系统的启动模式。
2. 根据不同的启动模式，寻找操作系统的引导程序。
3. 引导程序加载文件系统初始化程序 (initrd) 和内核初始化镜像 (vmlinuz)，完成操作系统启动前的的初始化。
4. 操作系统开始安装相关的系统和应用程序。

2 Ironic 部署过程

2.1 文字描述

部署过程描述如下：

1. 部署物理机的请求通过 Nova API 进入 Nova。
2. Nova Scheduler 根据请求参数中的信息选择合适的物理节点。
3. Nova 创建一个 spawn 任务，并调用 Ironic API 部署物理节点，Ironic 将此次任务中所需要的硬件资源保留，并更新数据库。
4. Ironic 与 openstack 的其他服务交互，从 Glance 服务获取部署物理节点所需的镜像资源，并调用 Neutron 服务为物理机创建网络端口。
5. Ironic 开始部署物理节点，PXE driver 准备 tftp booloader，IPMI driver 设置物理机启动模式并将机器上电。
6. 物理机启动后，通过 DHCP 获得 ironic Conductor 的地址并尝试通过 tftp 协议从 Conductor 获取镜像，Conductor 将部署镜像部署到物理节点上后，通过 iSCSI 协议将物理节点的硬盘暴露出来，随后写入用户镜像，成功部署用户镜像后，物理节点的部署就完成了。

2.2 代码描述

首先在 nova.conf 中修改 scheduler_host_manager 和 compute_driver，内容如下：

```
# /etc/nova/nova.conf
[DEFAULT]
scheduler_host_manager = nova.scheduler.ironic_host_manager.IronicHostManager
compute_driver = nova.virt.ironic.driver.IronicDriver
compute_manager = ironic.nova.compute.manager.ClusteredComputeManager
```

随后开始介绍裸机的部署过程：

1. nova-api 接收到 nova boot 的请求，通过消息队列到达 nova-scheduler。
2. nova-scheduler 收到请求后，在 scheduler_host_manager 里面处理。scheduler_host_manager 找到相匹配的物理节点后，发送 RPC 消息到 nova-compute。
3. nova-compute 拿到消息调用 compute_driver 的 spawn 方法进行部署，也就是调用 nova.virt.ironic.driver.IronicDriver.spawn()。

   spawn() 方法中的代码思路如下：

   - 获取节点。
   - 配置网络信息。
   - 配置驱动信息。
   - 触发部署，设置 ironic 的 provision_state 为 ACTIVE。
   - 然后等待 ironic 的 node provision_state 为 ACTIVE。

   spawn() 方法的代码分析如下;

```
# nova/virt/ironic/driver.py IronicDriver类
def spawn(self, context, instance, image_meta, injected_files,
          admin_password, network_info=None, block_device_info=None):

    # 获取镜像信息
    image_meta = objects.ImageMeta.from_dict(image_meta)

    ......

    # 调用ironic的node.get方法查询node的详细信息,锁定物理机，获取该物理机
      的套餐信息
    node = self.ironicclient.call("node.get", node_uuid)
    flavor = instance.flavor

    # 将套餐里面的baremetal:deploy_kernel_id和baremetal:deploy_ramdisk_id
      信息
    # 更新到driver_info，将image_source、root_gb、swap_mb、ephemeral_gb、
    # ephemeral_format、preserve_ephemeral信息更新到instance_info中，
    # 然后将driver_info和instance_info更新到ironic的node节点对应的属性上。
```

```
        self._add_driver_fields(node, instance, image_meta, flavor)

        .......

        # 验证是否可以部署，只有当deply和power都准备好了才能部署
        validate_chk = self.ironicclient.call("node.validate", node_uuid)
        .....

        # 准备部署
        try:
            # 将节点的虚拟网络接口和物理网络接口连接起来并调用ironic API
            # 进行更新，以便neutron可以连接
            self._plug_vifs(node, instance, network_info)
            self._start_firewall(instance, network_info)
        except Exception:
            ....

        # 配置驱动
        onfigdrive_value = self._generate_configdrive(
                instance, node, network_info, extra_md=extra_md,
                files=injected_files)


        # 触发部署请求
        try:
            # 调用ironic API，设置provision_state的状态ACTIVE
            self.ironicclient.call("node.set_provision_state", node_uuid,
                                   ironic_states.ACTIVE,
                                   configdrive=configdrive_value)
        except Exception as e:
            ....

        # 等待node provision_state为ATCTIVE
        timer = loopingcall.FixedIntervalLoopingCall(self._wait_for_active,
                                                     self.ironicclient,
                                                     instance)
        try:
            timer.start(interval=CONF.ironic.api_retry_interval).wait()
        except Exception:
              ...
```

4. 设置 ironic node 的 provision_state 为 ACTIVE 相当于发了一个 PUT 请求：“PUT /v1/nodes/(node_uuid)/states/provision”。随后，根据 openstack 的 wsgi 框架，注册了 app 为 ironic.api.app.VersionSelectorApplication 的方法开始处理这个 PUT 请求。

   ironic/api/controllers/v1/node/NodeStatesController 的 provision() 注册了 ironic.api.app.VersionSelectorAppl
   所以 provision() 执行，代码如下：

```
    # ironic/api/controllers/v1/node.py NodeStatesController类
    @expose.expose(None, types.uuid_or_name, wtypes.text,
                   wtypes.text, status_code=http_client.ACCEPTED)
    def provision(self, node_ident, target, configdrive=None):
        ....

```

```
        if target == ir_states.ACTIVE:
            #RPC调用do_node_deploy方法
            pecan.request.rpcapi.do_node_deploy(pecan.request.context,
                                                rpc_node.uuid, False,
                                                configdrive, topic)
    ...
```

5. RPC 调用 do_node_deploy() 方法，方法的代码思路如下：

   - 先检查电源和部署信息。
   - 检查完之后调用 ironic.condutor.manager.do_node_deploy() 方法。

ironic/condutor/manager/do_node_deploy() 方法代码分析如下：

```
# ironic/condutor/manager.py
def do_node_deploy(task, conductor_id, configdrive=None):
    """Prepare the environment and deploy a node."""
    node = task.node
    ...

    try:
        try:
            if configdrive:
                _store_configdrive(node, configdrive)
        except exception.SwiftOperationError as e:
            with excutils.save_and_reraise_exception():
                handle_failure(
                    e, task,
                    _LE('Error while uploading the configdrive for '
                        '%(node)s to Swift'),
                    _('Failed to upload the configdrive to Swift. '
                    'Error: %s'))

        try:
            #调用驱动的部署模块的prepare方法，不同驱动的动作不一样
            #1. pxe_* 驱动使用的是iscsi_deploy.ISCSIDeploy.prepare，
            #然后调用pxe.PXEBoot.prepare_ramdisk()准备部署进行和环境，包括
                cache images、 update DHCP、
            #switch pxe_config、set_boot_device等操作
            #cache images 是从glance上取镜像缓存到condutor本地，
            #update DHCP指定bootfile文件地址为condutor
            #switch pxe_config将deploy mode设置成service mode
            #set_boot_device设置节点pxe启动
            #2. agent_* 生成镜像swift_tmp_url加入节点的instance_info中
            #然后调用pxe.PXEBoot.prepare_ramdisk()准备部署镜像和环境
            task.driver.deploy.prepare(task)
        except Exception as e:
            ...

        try:
            #调用驱动的deploy方法，不同驱动动作不一样
            #1. pxe_* 驱动调用iscsi_deploy.ISCSIDeploy.deploy ()
            #进行拉取用户镜像，然后重启物理机
            #2. agent_*驱动，直接重启
```

```
40                  new_state = task.driver.deploy.deploy(task)
41              except Exception as e:
42                  ...
43
44              # NOTE(deva): Some drivers may return states.DEPLOYWAIT
45              #             eg. if they are waiting for a callback
46              if new_state == states.DEPLOYDONE:
47                  task.process_event('done')
48              elif new_state == states.DEPLOYWAIT:
49                  task.process_event('wait')
50
51          finally:
52              node.save()
```

6. 在上一步我们已经设置好了机器启动方式和相关网络并给机器上电了，那么下一步就是部署机器。有两种部署方式，分别是 PXE 和 agent。

2.2.1 PXE 部署方式

PXE 的部署流程如下：

1. 物理机上电后，BIOS 把 PXE client 调入内存执行，客户端广播 DHCP 请求。
2. DHCP 服务器给客户机分配 IP 并给定 bootstrap 文件的放置位置。
3. 客户机向本网络中的 TFTP 服务器索取 bootstrap 文件。
4. 客户机取得 bootstrap 文件后启动这个文件。
5. 根据 bootstrap 的执行结果，通过 TFTP 服务器加载内核和文件系统。
6. 在内存中启动安装，此时运行 init 脚本。

这个 init 脚本主要做了以下几个动作：

1. 找到磁盘，以该磁盘启动 iSCSI 设备。
2. TFTP 获取到 ironic 准备的 token 文件。
3. 调用 ironic 的 api，发送“POST v1/nodes/node-id/vendor_passthru/pass_deploy_info”，向 Ironic 请求部署镜像。
4. 启动 iSCSI 设备，开启 socket 端口 10000 等待通知 PXE 结束。
5. 结束后停止 iSCSI 设备。

init 脚本文件内容如下：

```
# 安装bootloader
function install_bootloader {
    #此处省略很多
    ...
}


#向Ironic Condutor发送消息，开启socket端口10000等待通知PXE结束
function do_vendor_passthru_and_wait {

    local data=$1
    local vendor_passthru_name=$2

    eval curl -i -X POST \
        "$TOKEN_HEADER" \
        "-H 'Accept: application/json'" \
        "-H 'Content-Type: application/json'" \
        -d "$data" \
        "$IRONIC_API_URL/v1/nodes/$DEPLOYMENT_ID/vendor_passthru/
            $vendor_passthru_name"

    echo "Waiting for notice of complete"
    nc -l -p 10000
}


readonly IRONIC_API_URL=$(get_kernel_parameter ironic_api_url)
readonly IRONIC_BOOT_OPTION=$(get_kernel_parameter boot_option)
readonly IRONIC_BOOT_MODE=$(get_kernel_parameter boot_mode)
readonly ROOT_DEVICE=$(get_kernel_parameter root_device)

if [ -z "$ISCSI_TARGET_IQN" ]; then
    err_msg "iscsi_target_iqn is not defined"
    troubleshoot
fi

#获取当前linux的本地硬盘
target_disk=
if [[ $ROOT_DEVICE ]]; then
    target_disk="$(get_root_device)"
else
    t=0
    while ! target_disk=$(find_disk "$DISK"); do
    if [ $t -eq 60 ]; then
        break
    fi
    t=$(($t + 1))
    sleep 1
    done
fi

if [ -z "$target_disk" ]; then
    err_msg "Could not find disk to use."
    troubleshoot
fi
```

```
55
56      #将找到的本地磁盘作为iSCSI磁盘启动，暴露给Ironic Condutor
57      echo "start iSCSI target on $target_disk"
58      start_iscsi_target "$ISCSI_TARGET_IQN" "$target_disk" ALL
59      if [ $? -ne 0 ]; then
60          err_msg "Failed to start iscsi target."
61          troubleshoot
62      fi
63
64      #获取到相关的token文件，从tftp服务器上获取，token文件在ironic在prepare阶段就生
            成好的。
65      if [ "$BOOT_METHOD" = "$VMEDIA_BOOT_TAG" ]; then
66          TOKEN_FILE="$VMEDIA_DIR/token"
67          if [ -f "$TOKEN_FILE" ]; then
68              TOKEN_HEADER="-H 'X-Auth-Token: $(cat $TOKEN_FILE)'"
69          else TOKEN_HEADER=""
70          fi
71          else
72          TOKEN_FILE=token-$DEPLOYMENT_ID
73
74          # Allow multiple versions of the tftp client
75          if tftp -r $TOKEN_FILE -g $BOOT_SERVER || tftp $BOOT_SERVER -c get
                $TOKEN_FILE; then
76              TOKEN_HEADER="-H 'X-Auth-Token: $(cat $TOKEN_FILE)'"
77          else
78              TOKEN_HEADER=""
79          fi
80      fi
81
82
83      #向Ironic请求部署镜像，POST node的/vendor_passthru/pass_deploy_info请求
84      echo "Requesting Ironic API to deploy image"
85      deploy_data="'{\"address\":\"$BOOT_IP_ADDRESS\",\"key\":\"$DEPLOYMENT_KEY\",\"
            iqn\":\"$ISCSI_TARGET_IQN\",\"error\":\"$FIRST_ERR_MSG\"}'"
86      do_vendor_passthru_and_wait "$deploy_data" "pass_deploy_info"
87
88      #部署镜像下载结束，停止iSCSI设备
89      echo "Stopping iSCSI target on $target_disk"
90      stop_iscsi_target
91
92      #如果是本地启动，安装bootloarder
93      # If localboot is set, install a bootloader
94      if [ "$IRONIC_BOOT_OPTION" = "local" ]; then
95          echo "Installing bootloader"
96
97          error_msg=$(install_bootloader)
98          if [ $? -eq 0 ]; then
99              status=SUCCEEDED
100         else
101             status=FAILED
102         fi
103
104         echo "Requesting Ironic API to complete the deploy"
105     bootloader_install_data="'{\"address\":\"$BOOT_IP_ADDRESS\",\"status\":\"
            $status\",\"key\":\"$DEPLOYMENT_KEY\",\"error\":\"$error_msg\"}'"
106         do_vendor_passthru_and_wait "$bootloader_install_data" "
```

```
            pass_bootloader_install_info"
107     fi
```

init 脚本调用了 ironic.drivers.modules.iscsi_deploy.AgentDeployMixin.continue_deploy() 方法，这个方法的代码思路如下;

1. 调用 do_agent_iscsi_deploy() 方法，解析 iscsi 部署的信息，然后在进行分区、格式化、写入镜像到磁盘。
2. 然后调用 prepare_instance() 在设置一遍 PXE 环境，为进入系统做准备，我们知道在 instance_info 上设置了 ramdisk、kernel、image_source3 个镜像。这里就是设置了 ramdisk 和 kernel 两个镜像文件，磁盘镜像在第一步中已经写到磁盘中去了，调用 switch_pxe_config() 方法将当前的操作系统的启动项设置为 ramdisk 和 kernel 作为引导程序。
3. 最后向节点的 10000 发送一个‘done’通知节点关闭 iSCSI 设备，最后节点重启安装用户操作系统，至此部署结束。

分析如下:

```
1   # ironic/drivers/modules/iscsi_deploy.py AgentDeployMixin类
2   def continue_deploy(self, task):
3       """Method invoked when deployed using iSCSI.
4
5       This method is invoked during a heartbeat from an agent when
6       the node is in wait-call-back state. This deploys the image on
7       the node and then configures the node to boot according to the
8       desired boot option (netboot or localboot).
9
10      :param task: a TaskManager object containing the node.
11      :param kwargs: the kwargs passed from the heartbeat method.
12      :raises: InstanceDeployFailure, if it encounters some error during
13          the deploy.
14      """
15      task.process_event('resume')
16      node = task.node
17      LOG.debug('Continuing the deployment on node %s', node.uuid)
18
19      # 继续部署的函数，连接到iSCSI设备，将用户镜像写到iSCSI设备上，退出删除iSCSI
            设备，
20      # 然后在Condutor上删除镜像文件
21      uuid_dict_returned = do_agent_iscsi_deploy(task, self._client)
22      root_uuid = uuid_dict_returned.get('root uuid')
23      efi_sys_uuid = uuid_dict_returned.get('efi system partition uuid')
24
25      # 再一次设置PXE引导，为准备进入用户系统做准备
26      self.prepare_instance_to_boot(task, root_uuid, efi_sys_uuid)
27
28      # 结束部署，通知ramdisk重启，将物理机设置为ative
29      self.reboot_and_finish_deploy(task)
```

2.2.2 使用 agent 驱动进行部署

使用agent驱动进行部署，在内存中启动安装，此时运行的是ironic-python-agent.agent.run()执行，代码如下：

```
    def run(self):
        """Run the Ironic Python Agent."""
        # Get the UUID so we can heartbeat to Ironic. Raises LookupNodeError
        # if there is an issue (uncaught, restart agent)
        self.started_at = _time()

        #加载hardware manager
        # Cached hw managers at runtime, not load time. See bug 1490008.
        hardware.load_managers()

        if not self.standalone:
            # Inspection should be started before call to lookup, otherwise
            # lookup will fail due to unknown MAC.
            uuid = inspector.inspect()

            # 利用Ironic API给Condutor发送lookup()请求，用户获取UUID
            # 发送一个“GET /{api_version}/drivers/{driver}/vendor_passthru/lookup
                ”
            content = self.api_client.lookup_node(
                hardware_info=hardware.dispatch_to_managers(
                                'list_hardware_info'),
                timeout=self.lookup_timeout,
                starting_interval=self.lookup_interval,
                node_uuid=uuid)

            self.node = content['node']
            self.heartbeat_timeout = content['heartbeat_timeout']

        wsgi = simple_server.make_server(
            self.listen_address[0],
            self.listen_address[1],
            self.api,
            server_class=simple_server.WSGIServer)

        # 发送心跳包
        if not self.standalone:
            # Don't start heartbeating until the server is listening
            self.heartbeater.start()

        try:
            wsgi.serve_forever()
        except BaseException:
            self.log.exception('shutting down')
        # 部署完成后停止心跳包
        if not self.standalone:
            self.heartbeater.stop()
```

heartbeat() 函数如下：

```
    @base.passthru(['POST'])
```

```
    def heartbeat(self, task, **kwargs):
        """Method for agent to periodically check in.

        The agent should be sending its agent_url (so Ironic can talk back)
        as a kwarg. kwargs should have the following format::

         {
             'agent_url': 'http://AGENT_HOST:AGENT_PORT'
         }

        AGENT_PORT defaults to 9999.
        """
        node = task.node
        driver_internal_info = node.driver_internal_info
        LOG.debug(
            'Heartbeat from %(node)s, last heartbeat at %(heartbeat)s.',
            {'node': node.uuid,
             'heartbeat': driver_internal_info.get('agent_last_heartbeat')})
        driver_internal_info['agent_last_heartbeat'] = int(_time())
        try:
            driver_internal_info['agent_url'] = kwargs['agent_url']
        except KeyError:
            raise exception.MissingParameterValue(_('For heartbeat operation, '
                                                    '"agent_url" must be '
                                                    'specified.'))

        node.driver_internal_info = driver_internal_info
        node.save()

        # Async call backs don't set error state on their own
        # TODO(jimrollenhagen) improve error messages here
        msg = _('Failed checking if deploy is done.')
        try:
            if node.maintenance:
                # this shouldn't happen often, but skip the rest if it does.
                LOG.debug('Heartbeat from node %(node)s in maintenance mode; '
                          'not taking any action.', {'node': node.uuid})
                return
            elif (node.provision_state == states.DEPLOYWAIT and
                  not self.deploy_has_started(task)):
                msg = _('Node failed to get image for deploy.')
                # 调用continue_deploy函数，下载镜像
                self.continue_deploy(task, **kwargs)

            # 查看IPA执行下载镜像是否结束
            elif (node.provision_state == states.DEPLOYWAIT and
                  self.deploy_is_done(task)):
                msg = _('Node failed to move to active state.')
                # 如果镜像已经下载完成，即部署完成，设置从disk启动，重启进入用户系
                    统
                self.reboot_to_instance(task, **kwargs)

            elif (node.provision_state == states.DEPLOYWAIT and
                  self.deploy_has_started(task)):
                #更新数据库，将节点的设置为alive
                node.touch_provisioning()
```

```
57
58              # TODO(lucasagomes): CLEANING here for backwards compat
59              # with previous code, otherwise nodes in CLEANING when this
60              # is deployed would fail. Should be removed once the Mitaka
61              # release starts.
62              elif node.provision_state in (states.CLEANWAIT, states.CLEANING):
63                  node.touch_provisioning()
64                  if not node.clean_step:
65                      LOG.debug('Node %s just booted to start cleaning.',
66                                node.uuid)
67                      msg = _('Node failed to start the next cleaning step.')
68                      manager.set_node_cleaning_steps(task)
69                      self._notify_conductor_resume_clean(task)
70                  else:
71                      msg = _('Node failed to check cleaning progress.')
72                      self.continue_cleaning(task, **kwargs)
73
74          except Exception as e:
75              err_info = {'node': node.uuid, 'msg': msg, 'e': e}
76              last_error = _('Asynchronous exception for node %(node)s: '
77                             '%(msg)s exception: %(e)s') % err_info
78              LOG.exception(last_error)
79              if node.provision_state in (states.CLEANING, states.CLEANWAIT):
80                  manager.cleaning_error_handler(task, last_error)
81              elif node.provision_state in (states.DEPLOYING, states.DEPLOYWAIT):
82                  deploy_utils.set_failed_state(task, last_error)
```

上述函数中的 continue_deploy() 方法如下：

```
1   @task_manager.require_exclusive_lock
2   def continue_deploy(self, task, **kwargs):
3       task.process_event('resume')
4       node = task.node
5       image_source = node.instance_info.get('image_source')
6       LOG.debug('Continuing deploy for node %(node)s with image %(img)s',
7                 {'node': node.uuid, 'img': image_source})
8
9       image_info = {
10          'id': image_source.split('/')[-1],
11          'urls': [node.instance_info['image_url']],
12          'checksum': node.instance_info['image_checksum'],
13          # NOTE(comstud): Older versions of ironic do not set
14          # 'disk_format' nor 'container_format', so we use .get()
15          # to maintain backwards compatibility in case code was
16          # upgraded in the middle of a build request.
17          'disk_format': node.instance_info.get('image_disk_format'),
18          'container_format': node.instance_info.get(
19              'image_container_format')
20      }
21
22      # 通知IPA下载swift上的镜像，并写入本地磁盘
23      # Tell the client to download and write the image with the given args
24      self._client.prepare_image(node, image_info)
25
26      task.process_event('wait')
```

3 参考学习的网站

以上笔记主要参考了这个网站：

https://doodu.gitbooks.io/openstack-ironic/content/bu_shu_liu_cheng.html