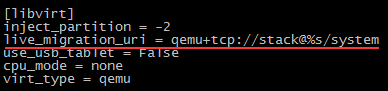


Migrate 操作会先将 instance 停掉，也就是所谓的“冷迁移”。而 Live Migrate 是“热迁移”，也叫“在线迁移”，instance不会停机。

Live Migrate 分两种：

1. 源和目标节点没有共享存储，instance 在迁移的时候需要将其镜像文件从源节点传到目标节点，这叫做 Block Migration（块迁移）
2. 源和目标节点共享存储，instance 的镜像文件不需要迁移，只需要将 instance 的状态迁移到目标节点。

源和目标节点需要满足一些条件才能支持 Live Migration：

1. 源和目标节点的 CPU 类型要一致。
2. 源和目标节点的 Libvirt 版本要一致。
3. 源和目标节点能相互识别对方的主机名称，比如可以在 /etc/hosts 中加入对方的条目。  
   http://7xo6kd.com1.z0.glb.clouddn.com/upload-ueditor-image-20160602-1464875385751012867.jpg
4. 在源和目标节点的 /etc/nova/nova.conf 中指明在线迁移时使用 TCP 协议。  
   
5. Instance 使用 config driver 保存其 metadata。在 Block Migration 过程中，该 config driver 也需要迁移到目标节点。由于目前 libvirt 只支持迁移 vfat 类型的 config driver，所以必须在 /etc/nova/nova.conf 中明确指明 launch instance 时创建 vfat 类型的 config driver。  
   http://7xo6kd.com1.z0.glb.clouddn.com/upload-ueditor-image-20160602-1464875386998099533.jpg
6. 源和目标节点的 Libvirt TCP 远程监听服务得打开，需要在下面两个配置文件中做一点配置。

**/etc/default/libvirt-bin**

start\_libvirtd="yes"

libvirtd\_opts="-d -l"

**/etc/libvirt/libvirtd.conf**

listen\_tls = 0

listen\_tcp = 1

unix\_sock\_group = "libvirtd"

unix\_sock\_ro\_perms = "0777"

unix\_sock\_rw\_perms = "0770"

auth\_unix\_ro = "none"

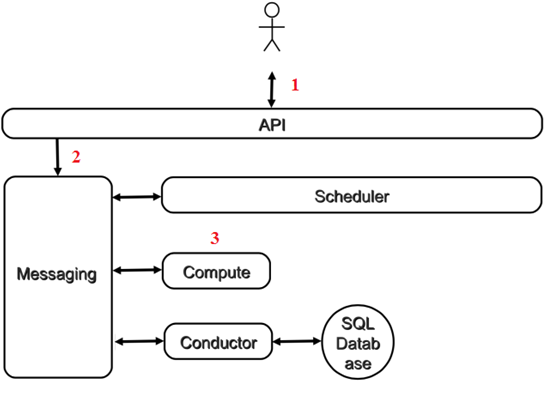
auth\_unix\_rw = "none"

auth\_tcp = "none"

然后重启 Libvirtd 服务  
**service libvirt-bin restart**

**非共享存储 Block Migration**

我们先讨论非共享存储的 Block Migration。流程图如下：

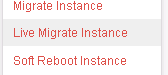


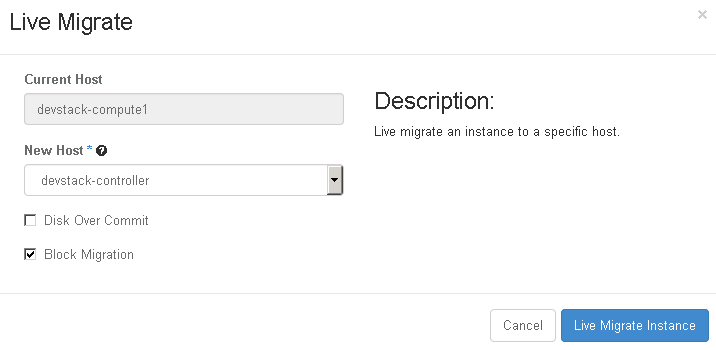
1. 向 nova-api 发送请求
2. nova-api 发送消息
3. nova-compute 执行操作

下面我们详细讨论每一个步骤。

**向nova-api发送请求**

客户（可以是 OpenStack 最终用户，也可以是其他程序）向API（nova-api）发送请求：“帮我将这个 Instance 从节点 A Live Migrate 到节点 B”

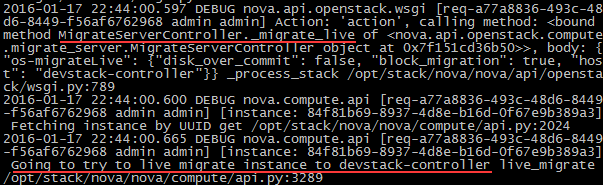




这里源节点是 devstack-compute1，目标节点是 devstack-controller，因为是非共享存储，记得将“Block Migration”勾选上。

这里还有一个“Disk Over Commit”选项，如果勾选了此选项，nova 在检查目标节点的磁盘空间是否足够时，是以 instance 磁盘镜像文件定义的最大容量为准；否则，以磁盘镜像文件当前的实际大小为准。

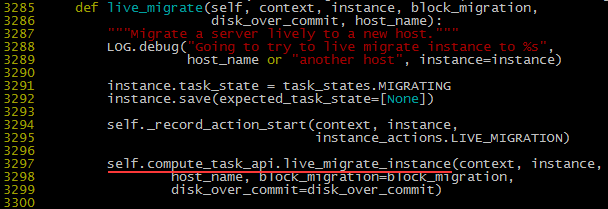
查看日志 /opt/stack/logs/n-api.log



nova-api 发送消息

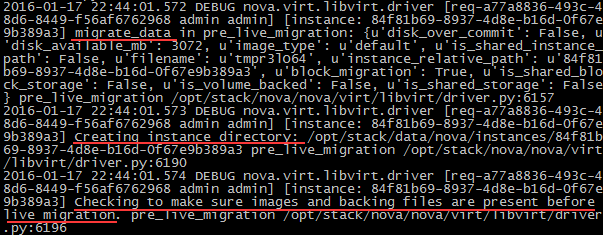
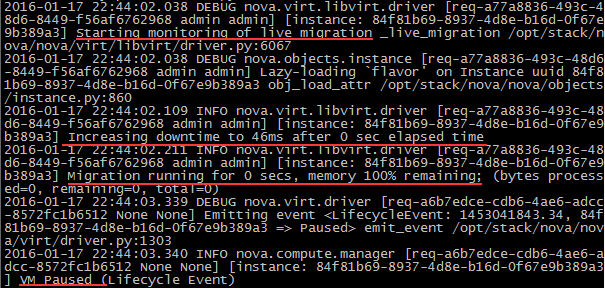
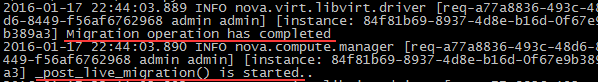
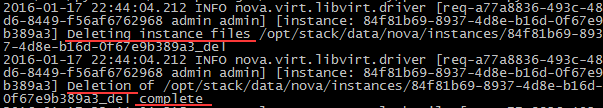
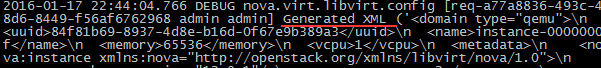
nova-api 向 Messaging（RabbitMQ）发送了一条消息：“Live Migrate 这个 Instance”

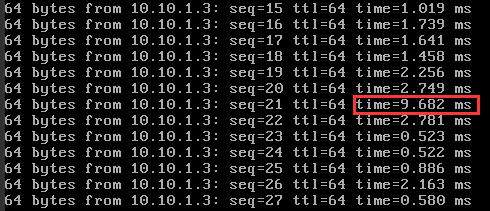
源代码在 /opt/stack/nova/nova/compute/api.py，方法是 live\_migrate。



**nova-compute 执行操作**

源和目标节点执行 Live Migrate 的操作过程如下：

1. 目标节点执行迁移前的准备工作，首先将 instance 的数据迁移过来，主要包括镜像文件、虚拟网络等资源，日志在 devstack-controller:/opt/stack/logs/n-cpu.log  
     
   http://7xo6kd.com1.z0.glb.clouddn.com/upload-ueditor-image-20160602-1464875388080027515.jpg
2. 源节点启动迁移操作，暂停 instance  
   
3. 在目标节点上 Resume instance  
   http://7xo6kd.com1.z0.glb.clouddn.com/upload-ueditor-image-20160602-1464875388381094037.jpg
4. 在源节点上执行迁移的后处理工作，删除 instance  
     
   
5. 在目标节点上执行迁移的后处理工作，创建 XML，在 Hypervisor 中定义 instance，使之下次能够正常启动。  
   

Instance 在 Live Migrate 的整个过程中不会停机，我们通过 Ping 操作来观察  


可见在迁移过程中，Ping 进程没有中断，只是有一个 ping 包的延迟增加了。

下面我们再来看源和目标节点共享存储下的 Live Migrate。

**共享存储 Live Migration**

有多种方式可以实现共享存储，比如可以将 instance 的镜像文件放在 NFS 服务器上，或者使用 NAS 服务器，或者分布式文件系统。

作为学习和实验，这里我们采用 NFS 方案。其他共享存储方案对于 Live Migration 本质上是一样的，只是在性能和高可用性上更好。

**搭建 NFS 环境**

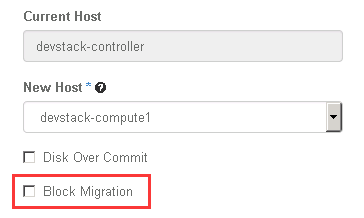
将 devstack-controller 作为 NFS 服务器，共享其目录 /opt/stack/data/nova/instances。

devstack-compute1 作为 NFS 客户端将此目录 mount 到本机，如下所示：

http://7xo6kd.com1.z0.glb.clouddn.com/upload-ueditor-image-20160602-1464875429765047272.png

这样，OpenStack 的 instance 在 devstack-controller 和 devstack-compute1 上就实现共享存储了。

共享存储的迁移过程与 Block Migrate 基本上一样，只是几个环节有点区别：

1. 向 nova-api 提交请求的时候，不能勾选“Block Migrate”  
   
2. 因为源和目标节点都能直接访问 instance 的镜像，所以目标节点在准备阶段不需要传输镜像文件，源节点在迁移后处理阶段也无需删除 instance 的目录。
3. 只有 instance 的状态需要从源节点传输到的目标节点，整个迁移速递比 Block Migration 快很多。

具体的日志分析留个大家练习。

以上是 Live Migrate 操作的详细分析，下一节我们讨论 Evacuate