指令：

> usage: cinder extend
>
> Attempts to extend size of an existing volume.
>
> Positional arguments:
>
> Name or ID of volume to extend.
>
> New size of volume, in GiBs.

最新的openstack(3.42)代码支持了热扩容功能：

> Add ability to extend 'in-use' volume. User should be aware of the
>
> whole environment before using this feature because it's dependent
>
> on several external factors below:
>
> 1. nova-compute version - needs to be the latest for Pike.
>
> 2. only the libvirt compute driver supports this currently.
>
> 3. only iscsi and fibre channel volume types are supported
>
> on the nova side currently.
>
> Administrator can disable this ability by updating the
>
> 'volume:extend_attached_volume' policy rule. Extend in reserved
>
> state is intentionally NOT allowed.

目前只有 *libvirt* 驱动能支持，且只有 *iscsi* 和 *fibre channel* 协议的卷类型能支持！

cinder.api.contrib.volume_actions.VolumeActionsController#_extend：

《openstack 官网对于热扩容在 CINDER 方面的说明》

《openstack 官网对于热扩容在 NOVA 方面的说明》

Cinder方面

Cinder 方面其实没有什么改动，跟冷扩容做的事一样，扩容后的size必须是整数GB且大于扩容前size。核心实现是调用卷驱动里的方法，如netapp，调用的是cinder.volume.drivers.netapp.dataontap.client.client_base.Client#do_direct_resize实现。唯一的区别是，扩容后，会调用nova api。

1.

```
# 如果版本是3.42 且 卷状态是in-use，则走热扩容
if req_version.matches("3.42") and volume.status in ['in-use']:
    self.volume_api.extend_attached_volume(context, volume, size)
else: # 否则，冷扩展
    self.volume_api.extend(context, volume, size)

```

1. cinder/volume/manager.py里调用nova api，通知nova要扩展卷：

```
if orig_volume_status == 'in-use':
        nova_api = compute.API()
        instance_uuids = [attachment.instance_uuid
                          for attachment in attachments]
        nova_api.extend_volume(context, instance_uuids, volume.id)
```

这里调用nova client 里的：novaclient.v2.server_external_events.ServerExternalEventManager#create:

```
class ServerExternalEventManager(base.Manager):
    resource_class = Event

    def create(self, events):
        """Create one or more server events.

        :param:events: A list of dictionaries containing 'server_uuid', 'r
                       'status', and 'tag' (which may be absent)
        """

        body = {'events': events}
        return self._create('/os-server-external-events', body, 'events',
                            return_raw=True)

```

'events'：

```
{'name': 'volume-extended',
'server_uuid': server_id,
'tag': volume_id}
```

Nova方面

Nova将使用现有的外部事件API端点监听来自Cinder的附加卷扩展通知。收到通知后，Nova将使用os-brick触发主机上的设备重新扫描，以发现卷大小的变化。

nova api '/os-server-external-events'：

- request：

```
    {
        "events": [
            {
                "name": "volume-extended",
                "server_uuid": "3df201cf-2451-44f2-8d25-a4ca826fc1f3",
                "tag": "0e63d806-6fe4-4ffc-99bf-f3dd056574c0"
            }
        ]
    }
```

- response：

```
    {
        "events": [
            {
                "name": "volume-extended",
                "status": "completed",
                "code": 200,
                "server_uuid": "3df201cf-2451-44f2-8d25-a4ca826fc1f3",
                "tag": "0e63d806-6fe4-4ffc-99bf-f3dd056574c0"
            }
        ]
    }
```

1. 检查nova驱动 `compute_driver = libvirt.LibvirtDriver` 是否能支持卷扩展 `"supports_extend_volume": True`
   驱动的支持功能定义在**nova.virt.libvirt.driver.LibvirtDriver**：

```
class LibvirtDriver(driver.ComputeDriver):
    capabilities = {
        "has_imagecache": True,
        "supports_recreate": True,
        "supports_migrate_to_same_host": False,
        "supports_attach_interface": True,
        "supports_device_tagging": True,
        "supports_tagged_attach_interface": True,
        "supports_tagged_attach_volume": True,
        "supports_extend_volume": True,
    }

```

1. 根据卷connection_info找对应的驱动，然后调用驱动的extend_volume方法。
   def _extend_volume(self, connection_info, instance):
   vol_driver = self._get_volume_driver(connection_info)
   return vol_driver.extend_volume(connection_info, instance)
2. 比如Iscsi，就会找**nova.virt.libvirt.volume.iscsi.LibvirtISCSIVolumeDriver#extend_volume**：

```
        new_size = self.connector.extend_volume(connection_info['data'])
```

1. 在调用ISCSIConnector的extend_volume
   os_brick.initiator.connectors.iscsi.ISCSIConnector#extend_volume：

```
        volume_paths = self.get_volume_paths(connection_properties)
        if volume_paths:
            return self._linuxscsi.extend_volume(volume_paths)
```

1. 调用linuxscsi **os_brick.initiator.linuxscsi.LinuxSCSI#extend_volume**:

```
    def extend_volume(self, volume_paths):
        """Signal the SCSI subsystem to test for volume resize.

        This function tries to signal the local system's kernel
        that an already attached volume might have been resized.
        """
        LOG.debug("extend volume %s", volume_paths)

        for volume_path in volume_paths:
            device = self.get_device_info(volume_path)
            LOG.debug("Volume device info = %s", device)
            device_id = ("%(host)s:%(channel)s:%(id)s:%(lun)s" %
                         {'host': device['host'],
                          'channel': device['channel'],
                          'id': device['id'],
                          'lun': device['lun']})

            scsi_path = ("/sys/bus/scsi/drivers/sd/%(device_id)s" %
                         {'device_id': device_id})

            size = self.get_device_size(volume_path)
            LOG.debug("Starting size: %s", size)

            # now issue the device rescan
            rescan_path = "%(scsi_path)s/rescan" % {'scsi_path': scsi_path

            # 在rescan_path文件里写入1
            self.echo_scsi_command(rescan_path, "1")
            new_size = self.get_device_size(volume_path)
            LOG.debug("volume size after scsi device rescan %s", new_size)

        scsi_wwn = self.get_scsi_wwn(volume_paths[0])
        mpath_device = self.find_multipath_device_path(scsi_wwn)
        if mpath_device:
            # Force a reconfigure so that resize works
            self.multipath_reconfigure()

            size = self.get_device_size(mpath_device)
            LOG.info("mpath(%(device)s) current size %(size)s",
                     {'device': mpath_device, 'size': size})

            # 调用指令 multipathd resize map multipath_device
            result = self.multipath_resize_map(scsi_wwn)
            if 'fail' in result:
                LOG.error("Multipathd failed to update the size mapping of
                          "multipath device %(scsi_wwn)s volume %(volume)s
                          {'scsi_wwn': scsi_wwn, 'volume': volume_paths})
                return None

            new_size = self.get_device_size(mpath_device)
            LOG.info("mpath(%(device)s) new size %(size)s",
                     {'device': mpath_device, 'size': new_size})

        return new_size
```

os_brick.initiator.linuxscsi.LinuxSCSI#multipath_resize_map：

```
    def multipath_resize_map(self, mpath_id):
        """Issue a multipath resize map on device.

        This forces the multipath daemon to update it's
        size information a particular multipath device.
        """
        (out, _err) = self._execute('multipathd', 'resize', 'map', mpath_i
                                    run_as_root=True,
                                    root_helper=self._root_helper)
        return out

```

流程简要：

1. 向scsi扫描文件写入 1： `tee -a 1 "%(scsi_path)s/rescan"`
2. `/lib/udev/scsi_id --page 0x83 --whitelisted` 得到scsi_wwn
3. `multipathd resize map scsi_wwn` 重新设置设备大小

FC和Iscsi驱动都是这样流程。

附：

centos 如果要使用 `multipathd` 指令，需要安装device-mapper-multipath， `sudo yum install device-mapper-multipath` 。

redhat官网上介绍了修改在线多路径设备容量的方法：

RESIZING AN ONLINE MULTIPATH DEVICE

If you need to resize an online multipath device, use the following procedure.

1. Resize your physical device.
2. Execute the following command to find the paths to the LUN:
   `# multipath -l`
3. Resize your paths. For SCSI devices, writing a 1 to the rescan file for the device causes the SCSI driver to rescan, as in the following command:
   `# echo 1 > /sys/block/path_device/device/rescan`
   Ensure that you run this command for each of the path devices. For example, if your path devices are sda, sdb, sde, and sdf, you would run the following commands:
   ```
   # echo 1 > /sys/block/sda/device/rescan
   # echo 1 > /sys/block/sdb/device/rescan
   # echo 1 > /sys/block/sde/device/rescan
   # echo 1 > /sys/block/sdf/device/rescan
   ```
4. Resize your multipath device by executing the multipathd resize command:
   `# multipathd resize map multipath_device`
5. Resize the file system (assuming no LVM or DOS partitions are used):
   `# resize2fs /dev/mapper/mpatha`

看的出，跟nova的处理流程差不多。