OpenStack 完整安装手册

作者:	yz
联系方式:	Mail: atkisc@gmail.com QQ: 949587200
日期:	2012-4-6
版本:	Essex 最终版

目录

实验环境

硬件:

DELL R710(1 台)

CPU : Intel(R) Xeon(R) CPU E5620 @ 2.40GHz * 2

内存 : 48GB

硬盘 : 300GB

网卡 : Broadcom Corporation NetXtreme II BCM5716 Gigabit Ethernet * 4

DELL R410(1 台)

CPU : Intel(R) Xeon(R) CPU E5606 @ 2.13GHz * 2

内存 : 8GB

硬盘 : 1T * 4

网卡 : Broadcom Corporation NetXtreme II BCM5709 Gigabit Ethernet * 4

系统 :

CentOS 6.2 x64

Openstack 版本 :

Essex release (2012.1)

架构部署

机器型号/主机名	外网 IP	内网 IP	作用
DELL R410/Control	60.12.206.105	192.168.1.2	控制节点
DELL R710/Compute	60.12.206.99	192.168.1.3	计算节点
实例网段为 10.0.0.0/24，Floating IP 为 60.12.206.110,实例网段桥接在内网网卡上，网络模式采用 FlatDHCP 控制节点 /dev/sda 为系统盘，/dev/sdb 为 nova-volume 盘，/dev/sdc、/dev/sdd 为 swift 存储用			

服务器系统安装

1. CentOS 6.2 x64 使用最小化安装方式
2. 服务器外网使用 eth0
3. 服务器内网使用 eth1
4. 所有服务均监听 0.0.0.0

控制节点安装

前提工作

1. 导入第三方软件源

   rpm -Uvh http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-5.noarch.rpm

   rpm -Uvh http://pkgs.repoforge.org/rpmforge-release/rpmforge-release-0.5.2-2.el6.rfx86_64.rpm

2. 安装依赖包

   yum -y install swig libvirt-python libvirt qemu-kvm python-pip gcc make gcc-c++ patch m4 python-devel libxml2-devel libxslt-devel libgsasl-devel openldap-devel sqlite-devel openssl-devel wget telnet gpxe-bootimgs gpxe-roms gpxe-roms-qemu dmidecode git scsi-target-utils kpartx socat vconfig aoetools

   rpm -Uvh http://mirrors.163.com/centos/6.2/os/x86_64/Packages/kernel-devel-2.6.32-220.el6.x86_64.rpm

   rpm -Uvh http://veillard.com/libvirt/6.3/x86_64/dnsmasq-utils-2.48-6.el6.x86_64.rpm

   ln -sv /usr/bin/pip-python /usr/bin/pip

NTP 时钟服务安装

1. 安装 NTP 时钟同步服务器

   yum install -y ntp

2. 编辑/etc/ntp.conf，将文件内容替换为如下：

   ```
   restrict default ignore
   restrict 127.0.0.1
   restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
   server ntp.api.bz
   server 127.127.1.0
   fudge 127.127.1.0 stratum 10
   driftfile /var/lib/ntp/drift
   keys /etc/ntp/keys
   ```

3. 重启 ntp 服务

   /etc/init.d/ntpd start

MYSQL 数据库服务安装

1. 安装 MYSQL 数据库服务

   yum install -y mysql-server

2. 更改 MYSQL 数据库服务监听内网网卡 IP

   sed -i '/symbolic-links=0/a bind-address = 192.168.1.2' /etc/my.cnf

3. 启动 MYSQL 数据库服务

   /etc/init.d/mysqld start

4. 设置 MYSQL 的 root 用户密码为 openstack

   mysqladmin -uroot password 'openstack';history -c

5. 检测服务是否正常启动

   通过 netstat -ltunp 查看是否有 tcp 3306 端口监听

   如果没有正常启动请查看/var/log/mysqld.log 文件排错

RABBITMQ 消息队列服务安装

1. 安装 RABBITMQ 消息队列服务

   yum -y install rabbitmq-server

2. 启动 RABBITMQ 消息队列服务

   /etc/init.d/rabbitmq-server start

3. 更改 RABBITMQ 消息队列服务 guest 用户默认密码为 openstack

   rabbitmqctl change_password guest openstack

PYTHON-NOVACLIENT 库安装

1. 下载源码包

   wget https://launchpad.net/nova/essex/2012.1/+download/python-novaclient-2012.1.tar.gz -P /opt

2. 安装依赖包

   yum -y install python-simplejson python-prettytable python-argparse python-nose1.1 python-httplib2 python-virtualenv MySQL-python

3. 解压并安装 PYTHON-NOVACLIENT 库

   cd /opt

   tar xf python-novaclient-2012.1.tar.gz

   cd python-novaclient-2012.1

   python setup.py install

   rm -f ../python-novaclient-2012.1.tar.gz

KEYSTONE 身份认证服务安装

1. 下载源码包

   wget https://launchpad.net/keystone/essex/2012.1/+download/keystone-2012.1.tar.gz -P /opt

2. 安装依赖包

   yum install -y python-eventlet python-greenlet python-paste python-passlib

   pip install routes==1.12.3 lxml==2.3 pam==0.1.4 sqlalchemy-migrate==0.7.2 PasteDeploy==1.5.0 SQLAlchemy==0.7.3 WebOb==1.0.8

3. 解压并安装 KEYSTONE 身份认证服务

   cd /opt

   tar xf keystone-2012.1.tar.gz

   cd keystone-2012.1

   python setup.py install

   rm -f ../keystone-2012.1.tar.gz

PYTHON-KEYSTONECLIENT 库安装

1. 下载源码包

   wget https://launchpad.net/keystone/essex/2012.1/+download/python-keystoneclient-2012.1.tar.gz -P /opt

2. 解压并安装 PYTHON-KEYSTONECLIENT 库

   cd /opt

   tar xf python-keystoneclient-2012.1.tar.gz

   cd python-keystoneclient-2012.1

   python setup.py install

   rm -f ../python-keystoneclient-2012.1.tar.gz

SWIFT 对象存储服务安装

1. 下载源码包

   wget https://launchpad.net/swift/essex/1.4.8/+download/swift-1.4.8.tar.gz -P /opt

2. 安装依赖包

   yum install -y memcached

   pip install configobj==4.7.1 netifaces==0.6

3. 解压并安装 SWIFT 对象存储服务

   cd /opt

tar xf swift-1.4.8.tar.gz

cd swift-1.4.8

python setup.py install

rm -f ../swift-1.4.8.tar.gz

GLANCE 镜像存储服务安装

1. 下载源码包

   wget https://launchpad.net/glance/essex/2012.1/+download/glance-2012.1.tar.gz -P /opt

2. 安装依赖包

   yum install -y python-anyjson python-kombu m2crypto

   pip install xattr==0.6.0 iso8601==0.1.4 pysendfile==2.0.0 pycrypto==2.3 wsgiref boto==2.1.1

3. 解压并安装 GLANCE 镜像存储服务

   cd /opt

   tar xf glance-2012.1.tar.gz

   cd glance-2012.1

   python setup.py install

   rm -f ../glance-2012.1.tar.gz

NOVA 计算服务安装

1. 下载源码包

   wget https://launchpad.net/nova/essex/2012.1/+download/nova-2012.1.tar.gz -P /opt

2. 安装依赖包

   yum install -y python-amqplib python-carrot python-lockfile python-gflags python-netaddr python-suds python-paramiko python-feedparser

   pip install pycrypto==2.3 Cheetah==2.4.4 python-daemon==1.5.5 Babel==0.9.6

3. 解压并安装 NOVA 计算服务

   cd /opt

   tar xf nova-2012.1.tar.gz

   cd nova-2012.1

   python setup.py install

   rm -f ../nova-2012.1.tar.gz

HORIZON 管理面板安装

1. 下载源码包

wget https://launchpad.net/horizon/essex/2012.1/+download/horizon-2012.1.tar.gz -P /opt

2. 安装依赖包

yum install -y python-django-nose python-dateutil python-cloudfiles python-django python-django-integration-apache httpd

3. 解压并安装 HORIZON 管理面板

cd /opt

tar xf horizon-2012.1.tar.gz

cd horizon-2012.1

python setup.py install

rm -f ../horizon-2012.1.tar.gz

NOVNC WEB 访问安装

1. 下载源码包

git clone https://github.com/cloudbuilders/noVNC.git /opt/noVNC

2. 安装依赖包

yum install -y python-numdisplay

KEYSTONE 身份认证服务配置

建立 KEYSTONE 服务数据库

mysql -uroot -popenstack -e 'create database keystone'

建立 KEYSTONE 服务配置文件存放目录

mkdir /etc/keystone

建立 KEYSTONE 服务启动用户

useradd -s /sbin/nologin -m -d /var/log/keystone keystone

在/etc/keystone 建立 default_catalog.templates 作为 KEYSTONE 服务服务点配置文件，内容如下：

catalog.RegionOne.identity.publicURL = http://60.12.206.105:$(public_port)s/v2.0

catalog.RegionOne.identity.adminURL = http://60.12.206.105:$(admin_port)s/v2.0

catalog.RegionOne.identity.internalURL = http://60.12.206.105:$(public_port)s/v2.0

catalog.RegionOne.identity.name = Identity Service

catalog.RegionOne.compute.publicURL = http://60.12.206.105:8774/v2/$(tenant_id)s

catalog.RegionOne.compute.adminURL = http://60.12.206.105:8774/v2/$(tenant_id)s

catalog.RegionOne.compute.internalURL = http://60.12.206.105:8774/v2/$(tenant_id)s

catalog.RegionOne.compute.name = Compute Service

```
catalog.RegionOne.volume.publicURL = http://60.12.206.105:8776/v1/$(tenant_id)s
catalog.RegionOne.volume.adminURL = http://60.12.206.105:8776/v1/$(tenant_id)s
catalog.RegionOne.volume.internalURL = http://60.12.206.105:8776/v1/$(tenant_id)s
catalog.RegionOne.volume.name = Volume Service

catalog.RegionOne.ec2.publicURL = http://60.12.206.105:8773/services/Cloud
catalog.RegionOne.ec2.adminURL = http://60.12.206.105:8773/services/Admin
catalog.RegionOne.ec2.internalURL = http://60.12.206.105:8773/services/Cloud
catalog.RegionOne.ec2.name = EC2 Service

catalog.RegionOne.s3.publicURL = http://60.12.206.105:3333
catalog.RegionOne.s3.adminURL = http://60.12.206.105:3333
catalog.RegionOne.s3.internalURL = http://60.12.206.105:3333
catalog.RegionOne.s3.name = S3 Service

catalog.RegionOne.image.publicURL = http://60.12.206.105:9292/v1
catalog.RegionOne.image.adminURL = http://60.12.206.105:9292/v1
catalog.RegionOne.image.internalURL = http://60.12.206.105:9292/v1
catalog.RegionOne.image.name = Image Service

catalog.RegionOne.object_store.publicURL = http://60.12.206.105:8080/v1/AUTH_$(tenant_id)s
catalog.RegionOne.object_store.adminURL = http://60.12.206.105:8080/
catalog.RegionOne.object_store.internalURL = http://60.12.206.105:8080/v1/AUTH_$(tenant_id)s
catalog.RegionOne.object_store.name = Swift Service

# 在/etc/keystone 建立 policy.json 作为 KEYSTONE 服务策略文件，内容如下：
{
    "admin_required": [["role:admin"], ["is_admin:1"]]
}

# 在/etc/keystone 建立 keystone.conf 作为 KEYSTONE 服务配置文件，内容如下：
[DEFAULT]
public_port = 5000
admin_port = 35357
admin_token = ADMIN
compute_port = 8774
verbose = True
debug = True
log_file = /var/log/keystone/keystone.log
use_syslog = False
syslog_log_facility = LOG_LOCAL0

[sql]
```

```
connection = mysql://root:openstack@localhost/keystone
idle_timeout = 30
min_pool_size = 5
max_pool_size = 10
pool_timeout = 200

[identity]
driver = keystone.identity.backends.sql.Identity

[catalog]
driver = keystone.catalog.backends.templated.TemplatedCatalog
template_file = /etc/keystone/default_catalog.templates

[token]
driver = keystone.token.backends.kvs.Token

[policy]
driver = keystone.policy.backends.simple.SimpleMatch

[ec2]
driver = keystone.contrib.ec2.backends.sql.Ec2

[filter:debug]
paste.filter_factory = keystone.common.wsgi:Debug.factory

[filter:token_auth]
paste.filter_factory = keystone.middleware:TokenAuthMiddleware.factory

[filter:admin_token_auth]
paste.filter_factory = keystone.middleware:AdminTokenAuthMiddleware.factory

[filter:xml_body]
paste.filter_factory = keystone.middleware:XmlBodyMiddleware.factory

[filter:json_body]
paste.filter_factory = keystone.middleware:JsonBodyMiddleware.factory

[filter:crud_extension]
paste.filter_factory = keystone.contrib.admin_crud:CrudExtension.factory

[filter:ec2_extension]
paste.filter_factory = keystone.contrib.ec2:Ec2Extension.factory

[filter:s3_extension]
```

```
paste.filter_factory = keystone.contrib.s3:S3Extension.factory

[app:public_service]
paste.app_factory = keystone.service:public_app_factory

[app:admin_service]
paste.app_factory = keystone.service:admin_app_factory

[pipeline:public_api]
pipeline = token_auth admin_token_auth xml_body json_body debug ec2_extension s3_extension public_service

[pipeline:admin_api]
pipeline = token_auth admin_token_auth xml_body json_body debug ec2_extension crud_extension admin_service

[app:public_version_service]
paste.app_factory = keystone.service:public_version_app_factory

[app:admin_version_service]
paste.app_factory = keystone.service:admin_version_app_factory

[pipeline:public_version_api]
pipeline = xml_body public_version_service

[pipeline:admin_version_api]
pipeline = xml_body admin_version_service

[composite:main]
use = egg:Paste#urlmap
/v2.0 = public_api
/ = public_version_api

[composite:admin]
use = egg:Paste#urlmap
/v2.0 = admin_api
/ = admin_version_api

# 在/etc/init.d/下建立名为 keystone 的 KEYSTONE 服务启动脚本，内容如下：
#!/bin/sh
#
# keystone   OpenStack Identity Service
#
# chkconfig:    - 20 80
```

```
    # description: keystone works provide apis to   \
    #                   * Authenticate users and provide a token \
    #                   * Validate tokens
    ### END INIT INFO

    . /etc/rc.d/init.d/functions

    prog=keystone
    prog_exec=keystone-all
    exec="/usr/bin/$prog_exec"
    config="/etc/$prog/$prog.conf"
    pidfile="/var/run/$prog/$prog.pid"

    [ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

    lockfile=/var/lock/subsys/$prog

    start() {
        [ -x $exec ] || exit 5
        [ -f $config ] || exit 6
        echo -n $"Starting $prog: "
        daemon --user keystone --pidfile $pidfile "$exec --config-file=$config &>/dev/null & echo \$!
> $pidfile"
        retval=$?
        echo
        [ $retval -eq 0 ] && touch $lockfile
        return $retval
    }

    stop() {
        echo -n $"Stopping $prog: "
        killproc -p $pidfile $prog
        retval=$?
        echo
        [ $retval -eq 0 ] && rm -f $lockfile
        return $retval
    }

    restart() {
        stop
        start
    }

    reload() {
```

```
    restart
}

force_reload() {
    restart
}

rh_status() {
    status -p $pidfile $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
```

```
        exit 2
esac
exit $?

# 配置启动脚本：
chmod 755 /etc/init.d/keystone
mkdir /var/run/keystone
mkdir /var/lock/keystone
chown keystone:root /var/run/keystone
chown keystone:root /var/lock/keystone

# 启动 KEYSTONE 服务
/etc/init.d/keystone start

# 检测服务是否正常启动
通过 netstat -ltunp 查看是否有 tcp 5000 和 tcp 35357 端口监听
如果没有正常启动请查看/var/log/keystone/keystone.log 文件排错

# 建立 KEYSTONE 服务初始化数据脚本 keystone_data.sh，内容如下：
#!/bin/bash
# Variables set before calling this script:
# SERVICE_TOKEN - aka admin_token in keystone.conf
# SERVICE_ENDPOINT - local Keystone admin endpoint
# SERVICE_TENANT_NAME - name of tenant containing service accounts
# ENABLED_SERVICES - stack.sh's list of services to start
# DEVSTACK_DIR - Top-level DevStack directory

ADMIN_PASSWORD=${ADMIN_PASSWORD:-secrete}
SERVICE_PASSWORD=${SERVICE_PASSWORD:-service}
export SERVICE_TOKEN=ADMIN
export SERVICE_ENDPOINT=http://localhost:35357/v2.0
SERVICE_TENANT_NAME=${SERVICE_TENANT_NAME:-tenant}

function get_id () {
    echo `$@ | awk '/ id / { print $4 }'`
}

# Tenants
ADMIN_TENANT=$(get_id keystone tenant-create --name=admin)
SERVICE_TENANT=$(get_id keystone tenant-create --name=$SERVICE_TENANT_NAME)
DEMO_TENANT=$(get_id keystone tenant-create --name=demo)
INVIS_TENANT=$(get_id keystone tenant-create --name=invisible_to_admin)

# Users
```

```
ADMIN_USER=$(get_id keystone user-create --name=admin \
                                         --pass="$ADMIN_PASSWORD" \
                                         --email=admin@example.com)
DEMO_USER=$(get_id keystone user-create --name=demo \
                                        --pass="$ADMIN_PASSWORD" \
                                        --email=demo@example.com)

# Roles
ADMIN_ROLE=$(get_id keystone role-create --name=admin)
KEYSTONEADMIN_ROLE=$(get_id keystone role-create --name=KeystoneAdmin)
KEYSTONESERVICE_ROLE=$(get_id keystone role-create --name=KeystoneServiceAdmin)
ANOTHER_ROLE=$(get_id keystone role-create --name=anotherrole)

# Add Roles to Users in Tenants
keystone user-role-add --user $ADMIN_USER --role $ADMIN_ROLE --tenant_id $ADMIN_TENANT
keystone user-role-add --user $ADMIN_USER --role $ADMIN_ROLE --tenant_id $DEMO_TENANT
keystone user-role-add --user $DEMO_USER --role $ANOTHER_ROLE --tenant_id $DEMO_TENANT

# TODO(termie): these two might be dubious
keystone user-role-add --user $ADMIN_USER --role $KEYSTONEADMIN_ROLE --tenant_id $ADMIN_T
ENANT
keystone user-role-add --user $ADMIN_USER --role $KEYSTONESERVICE_ROLE --tenant_id $ADMIN_
TENANT

# The Member role is used by Horizon and Swift so we need to keep it:
MEMBER_ROLE=$(get_id keystone role-create --name=Member)
keystone user-role-add --user $DEMO_USER --role $MEMBER_ROLE --tenant_id $DEMO_TENANT
keystone user-role-add --user $DEMO_USER --role $MEMBER_ROLE --tenant_id $INVIS_TENANT

NOVA_USER=$(get_id keystone user-create --name=nova \
                                        --pass="$SERVICE_PASSWORD" \
                                        --tenant_id $SERVICE_TENANT \
                                        --email=nova@example.com)
keystone user-role-add --tenant_id $SERVICE_TENANT \
                       --user $NOVA_USER \
                       --role $ADMIN_ROLE

GLANCE_USER=$(get_id keystone user-create --name=glance \
                                          --pass="$SERVICE_PASSWORD" \
                                          --tenant_id $SERVICE_TENANT \
                                          --email=glance@example.com)
keystone user-role-add --tenant_id $SERVICE_TENANT \
                       --user $GLANCE_USER \
                       --role $ADMIN_ROLE
```

```
SWIFT_USER=$(get_id keystone user-create --name=swift \
                                         --pass="$SERVICE_PASSWORD" \
                                         --tenant_id $SERVICE_TENANT \
                                         --email=swift@example.com)
keystone user-role-add --tenant_id $SERVICE_TENANT \
                       --user $SWIFT_USER \
                       --role $ADMIN_ROLE

RESELLER_ROLE=$(get_id keystone role-create --name=ResellerAdmin)
keystone user-role-add --tenant_id $SERVICE_TENANT \
                       --user $NOVA_USER \
                       --role $RESELLER_ROLE

# 建立 KEYSTONE 服务数据库结构
keystone-manage db_sync

# 执行初始化数据脚本
bash keystone_data.sh
```

GLANCE 镜像存储服务配置

```
# 建立 GLANCE 服务数据库
mysql -uroot -popenstack -e 'create database glance'

# 建立 GLANCE 服务配置文件存放目录
mkdir /etc/glance

# 建立 GLANCE 服务启动用户
useradd -s /sbin/nologin -m -d /var/log/glance glance

# 在/etc/glance 建立 glance-api.conf 作为 GLANCE-API 服务配置文件，内容如下：
[DEFAULT]
# Show more verbose log output (sets INFO log level output)
verbose = True

# Show debugging output in logs (sets DEBUG log level output)
debug = True

# Which backend store should Glance use by default is not specified
# in a request to add a new image to Glance? Default: 'file'
# Available choices are 'file', 'swift', and 's3'
default_store = file
```

```
# Address to bind the API server
bind_host = 0.0.0.0

# Port the bind the API server to
bind_port = 9292

# Address to find the registry server
registry_host = 0.0.0.0

# Port the registry server is listening on
registry_port = 9191

# Log to this file. Make sure you do not set the same log
# file for both the API and registry servers!
log_file = /var/log/glance/api.log

# Send logs to syslog (/dev/log) instead of to file specified by `log_file`
use_syslog = False

# ============ Notification System Options =====================

# Notifications can be sent when images are create, updated or deleted.
# There are three methods of sending notifications, logging (via the
# log_file directive), rabbit (via a rabbitmq queue) or noop (no
# notifications sent, the default)
notifier_strategy = noop

# Configuration options if sending notifications via rabbitmq (these are
# the defaults)
rabbit_host = localhost
rabbit_port = 5672
rabbit_use_ssl = false
rabbit_userid = guest
rabbit_password = openstack
rabbit_virtual_host = /
rabbit_notification_topic = glance_notifications

# ============ Filesystem Store Options ========================

# Directory that the Filesystem backend store
# writes image data to
filesystem_store_datadir = /var/lib/glance/images/
```

```
# ============  Swift Store Options ============================

# Address where the Swift authentication service lives
swift_store_auth_address = 127.0.0.1:8080/v1.0/

# User to authenticate against the Swift authentication service
swift_store_user = jdoe

# Auth key for the user authenticating against the
# Swift authentication service
swift_store_key = a86850deb2742ec3cb41518e26aa2d89

# Container within the account that the account should use
# for storing images in Swift
swift_store_container = glance

# Do we create the container if it does not exist?
swift_store_create_container_on_put = False

# What size, in MB, should Glance start chunking image files
# and do a large object manifest in Swift? By default, this is
# the maximum object size in Swift, which is 5GB
swift_store_large_object_size = 5120

# When doing a large object manifest, what size, in MB, should
# Glance write chunks to Swift? This amount of data is written
# to a temporary disk buffer during the process of chunking
# the image file, and the default is 200MB
swift_store_large_object_chunk_size = 200

# Whether to use ServiceNET to communicate with the Swift storage servers.
# (If you aren't RACKSPACE, leave this False!)
#
# To use ServiceNET for authentication, prefix hostname of
# `swift_store_auth_address` with 'snet-'.
# Ex. https://example.com/v1.0/ -> https://snet-example.com/v1.0/
swift_enable_snet = False

# ============  S3 Store Options ============================

# Address where the S3 authentication service lives
s3_store_host = 127.0.0.1:8080/v1.0/

# User to authenticate against the S3 authentication service
```

```
s3_store_access_key = <20-char AWS access key>

# Auth key for the user authenticating against the
# S3 authentication service
s3_store_secret_key = <40-char AWS secret key>

# Container within the account that the account should use
# for storing images in S3. Note that S3 has a flat namespace,
# so you need a unique bucket name for your glance images. An
# easy way to do this is append your AWS access key to "glance".
# S3 buckets in AWS *must* be lowercased, so remember to lowercase
# your AWS access key if you use it in your bucket name below!
s3_store_bucket = <lowercased 20-char aws access key>glance

# Do we create the bucket if it does not exist?
s3_store_create_bucket_on_put = False

# ============ Image Cache Options ========================

image_cache_enabled = False

# Directory that the Image Cache writes data to
# Make sure this is also set in glance-pruner.conf
image_cache_datadir = /var/lib/glance/image-cache/

# Number of seconds after which we should consider an incomplete image to be
# stalled and eligible for reaping
image_cache_stall_timeout = 86400

# ============ Delayed Delete Options =============================

# Turn on/off delayed delete
delayed_delete = False

# Delayed delete time in seconds
scrub_time = 43200

# Directory that the scrubber will use to remind itself of what to delete
# Make sure this is also set in glance-scrubber.conf
scrubber_datadir = /var/lib/glance/scrubber

# 在/etc/glance 建立 glance-api-paste.ini 作为 GLANCE-API 服务认证配置文件，内容如下：
[pipeline:glance-api]
#pipeline = versionnegotiation context apiv1app
```

```
# NOTE: use the following pipeline for keystone
pipeline = versionnegotiation authtoken context apiv1app

# To enable Image Cache Management API replace pipeline with below:
# pipeline = versionnegotiation context imagecache apiv1app
# NOTE: use the following pipeline for keystone auth (with caching)
# pipeline = versionnegotiation authtoken auth-context imagecache apiv1app

[app:apiv1app]
paste.app_factory = glance.common.wsgi:app_factory
glance.app_factory = glance.api.v1.router:API

[filter:versionnegotiation]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.api.middleware.version_negotiation:VersionNegotiationFilter

[filter:cache]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.api.middleware.cache:CacheFilter

[filter:cachemanage]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.api.middleware.cache_manage:CacheManageFilter

[filter:context]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.common.context:ContextMiddleware

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
service_host = 60.12.206.105
service_port = 5000
service_protocol = http
auth_host = 60.12.206.105
auth_port = 35357
auth_protocol = http
auth_uri = http:/60.12.206.105:500/
admin_tenant_name = tenant
admin_user = glance
admin_password = service

# 在/etc/glance 建立 glance-registry.conf 作为 GLANCE-REGISTRY 服务配置文件，内容如下：
[DEFAULT]
# Show more verbose log output (sets INFO log level output)
```

```
verbose = True

# Show debugging output in logs (sets DEBUG log level output)
debug = True

# Address to bind the registry server
bind_host = 0.0.0.0

# Port the bind the registry server to
bind_port = 9191

# Log to this file. Make sure you do not set the same log
# file for both the API and registry servers!
log_file = /var/log/glance/registry.log

# Where to store images
filesystem_store_datadir = /var/lib/glance/images

# Send logs to syslog (/dev/log) instead of to file specified by `log_file`
use_syslog = False

# SQLAlchemy connection string for the reference implementation
# registry server. Any valid SQLAlchemy connection string is fine.
# See: http://www.sqlalchemy.org/docs/05/reference/sqlalchemy/connections.html#sqlalchemy.create_
engine
sql_connection = mysql://root:openstack@localhost/glance

# Period in seconds after which SQLAlchemy should reestablish its connection
# to the database.
#
# MySQL uses a default `wait_timeout` of 8 hours, after which it will drop
# idle connections. This can result in 'MySQL Gone Away' exceptions. If you
# notice this, you can lower this value to ensure that SQLAlchemy reconnects
# before MySQL can drop the connection.
sql_idle_timeout = 3600

# Limit the api to return `param_limit_max` items in a call to a container. If
# a larger `limit` query param is provided, it will be reduced to this value.
api_limit_max = 1000

# If a `limit` query param is not provided in an api request, it will
# default to `limit_param_default`
limit_param_default = 25
```

```
# 在/etc/glance 建立 glance-registry-paste.ini 作为 GLANCE-REGISTRY 服务认证配置文件，内容如下：
[pipeline:glance-registry]
#pipeline = context registryapp
# NOTE: use the following pipeline for keystone
pipeline = authtoken context registryapp

[app:registryapp]
paste.app_factory = glance.common.wsgi:app_factory
glance.app_factory = glance.registry.api.v1:API

[filter:context]
context_class = glance.registry.context.RequestContext
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.common.context:ContextMiddleware

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
service_host = 60.12.206.105
service_port = 5000
service_protocol = http
auth_host = 60.12.206.105
auth_port = 35357
auth_protocol = http
auth_uri = http:/60.12.206.105:500/
admin_tenant_name = tenant
admin_user = glance
admin_password = service

# 在/etc/glance 建立 policy.json 作为 GLANCE 服务策略文件，内容如下：
{
    "default": [],
    "manage_image_cache": [["role:admin"]]
}

# 在/etc/init.d/下建立名为 glance-api 的 GLANCE-API 服务启动脚本，内容如下：
#!/bin/sh
#
# glance-api OpenStack Image Service API server
#
# chkconfig:   - 20 80
# description: OpenStack Image Service (code-named Glance) API server

### BEGIN INIT INFO
# Provides:
```

```
    # Required-Start: $remote_fs $network $syslog
    # Required-Stop: $remote_fs $syslog
    # Default-Stop: 0 1 6
    # Short-Description: Glance API server
    # Description: OpenStack Image Service (code-named Glance) API server
    ### END INIT INFO

    . /etc/rc.d/init.d/functions

    suffix=api
    prog=openstack-glance-$suffix
    exec="/usr/bin/glance-$suffix"
    config="/etc/glance/glance-$suffix.conf"
    pidfile="/var/run/glance/glance-$suffix.pid"

    [ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

    lockfile=/var/lock/subsys/$prog

    start() {
        [ -x $exec ] || exit 5
        [ -f $config ] || exit 6
        echo -n $"Starting $prog: "
        daemon --user glance --pidfile $pidfile "$exec --config-file=$config &>/dev/null & echo \$! >
$pidfile"
        retval=$?
        echo
        [ $retval -eq 0 ] && touch $lockfile
        return $retval
    }

    stop() {
        echo -n $"Stopping $prog: "
        killproc -p $pidfile $prog
        retval=$?
        echo
        [ $retval -eq 0 ] && rm -f $lockfile
        return $retval
    }

    restart() {
        stop
        start
    }
```

```
reload() {
    restart
}

force_reload() {
    restart
}

rh_status() {
    status -p $pidfile $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
```

```
        *)
            echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
            exit 2
    esac
    exit $?

    # 在/etc/init.d/下建立名为 glance-registry 的 GLANCE-REGISTRY 服务启动脚本，内容如下：
    #!/bin/sh
    #
    # glance-registry OpenStack Image Service Registry server
    #
    # chkconfig:    - 20 80
    # description: OpenStack Image Service (code-named Glance) Registry server

    ### BEGIN INIT INFO
    # Provides:
    # Required-Start: $remote_fs $network $syslog
    # Required-Stop: $remote_fs $syslog
    # Default-Stop: 0 1 6
    # Short-Description: Glance Registry server
    # Description: OpenStack Image Service (code-named Glance) Registry server
    ### END INIT INFO

    . /etc/rc.d/init.d/functions

    suffix=registry
    prog=openstack-glance-$suffix
    exec="/usr/bin/glance-$suffix"
    config="/etc/glance/glance-$suffix.conf"
    pidfile="/var/run/glance/glance-$suffix.pid"

    [ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

    lockfile=/var/lock/subsys/$prog

    start() {
        [ -x $exec ] || exit 5
        [ -f $config ] || exit 6
        echo -n $"Starting $prog: "
        daemon --user glance --pidfile $pidfile "$exec --config-file=$config &>/dev/null & echo \$! >
$pidfile"
        retval=$?
        echo
        [ $retval -eq 0 ] && touch $lockfile
```

```
    return $retval
}

stop() {
    echo -n $"Stopping $prog: "
    killproc -p $pidfile $prog
    retval=$?
    echo
    [ $retval -eq 0 ] && rm -f $lockfile
    return $retval
}

restart() {
    stop
    start
}

reload() {
    restart
}

force_reload() {
    restart
}

rh_status() {
    status -p $pidfile $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
```

```
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
        exit 2
esac
exit $?

# 配置启动脚本：
chmod 755 /etc/init.d/glance-api
chmod 755 /etc/init.d/glance-registry
mkdir /var/run/glance
mkdir /var/lock/glance
mkdir -p /var/lib/glance/images
chown glance:root /var/run/glance
chown glance:root /var/lock/glance
chown glance:glance /var/lib/glance

# 启动 GLANCE-API 和 GLANCE-REGISTRY 服务
/etc/init.d/glance-api start
/etc/init.d/glance-registry start

# 检测服务是否正常启动
通过 netstat -ltunp 查看是否有 tcp 9292 和 tcp 9191 端口监听
如果没有正常启动请查看/var/log/glance 目录下相关文件排错
```

NOVA 计算服务配置

```
# 建立 NOVA 服务数据库
```

```
mysql -uroot -popenstack -e 'create database nova'

# 建立 NOVA 服务配置文件存放目录
mkdir /etc/nova

# 建立 NOVA 服务启动用户
useradd -s /sbin/nologin -m -d /var/log/nova nova

# 在/etc/nova 建立 nova.conf 作为 NOVA 服务配置文件，内容如下：
[DEFAULT]

debug=True
log-dir=/var/log/nova
pybasedir=/var/lib/nova
use_syslog=False
verbose=True
api_paste_config=/etc/nova/api-paste.ini
auth_strategy=keystone
bindir=/usr/bin
glance_host=$my_ip
glance_port=9292
glance_api_servers=$glance_host:$glance_port
image_service=nova.image.glance.GlanceImageService
lock_path=/var/lock/nova
my_ip=60.12.206.105
rabbit_host=localhost
rabbit_password=openstack
rabbit_port=5672
rabbit_userid=guest
root_helper=sudo
sql_connection=mysql://root:gamewave@localhost/nova
keystone_ec2_url=http://$my_ip:5000/v2.0/ec2tokens
novncproxy_base_url=http://$my_ip:6080/vnc_auto.html
vnc_enabled=True
vnc_keymap=en-us
vncserver_listen=$my_ip
vncserver_proxyclient_address=$my_ip
dhcpbridge=$bindir/nova-dhcpbridge
dhcpbridge_flagfile=/etc/nova/nova.conf
public_interface=eth0
routing_source_ip=$my_ip
fixed_range=10.0.0.0/24
flat_interface=br1
flat_network_bridge=eth1
```

```
floating_range=60.12.206.115
force_dhcp_release=True
target_host=$my_ip
target_port=3260
console_token_ttl=600
iscsi_helper=ietadm
iscsi_ip_address=$my_ip
iscsi_num_targets=100
iscsi_port=3260
volume_group=nova-volumes
ec2_listen=0.0.0.0
ec2_listen_port=8773
metadata_listen=0.0.0.0
metadata_listen_port=8775
osapi_compute_listen=0.0.0.0
osapi_compute_listen_port=8774
osapi_volume_listen=0.0.0.0
osapi_volume_listen_port=8776

# 在/etc/nova 建立 api-paste.ini 作为 NOVA 服务认证配置文件，内容如下：
############
# Metadata #
############
[composite:metadata]
use = egg:Paste#urlmap
/: metaversions
/latest: meta
/1.0: meta
/2007-01-19: meta
/2007-03-01: meta
/2007-08-29: meta
/2007-10-10: meta
/2007-12-15: meta
/2008-02-01: meta
/2008-09-01: meta
/2009-04-04: meta

[pipeline:metaversions]
pipeline = ec2faultwrap logrequest metaverapp

[pipeline:meta]
pipeline = ec2faultwrap logrequest metaapp

[app:metaverapp]
```

```
paste.app_factory = nova.api.metadata.handler:Versions.factory

[app:metaapp]
paste.app_factory = nova.api.metadata.handler:MetadataRequestHandler.factory

#######
# EC2 #
#######

[composite:ec2]
use = egg:Paste#urlmap
/services/Cloud: ec2cloud

[composite:ec2cloud]
use = call:nova.api.auth:pipeline_factory
noauth = ec2faultwrap logrequest ec2noauth cloudrequest validator ec2executor
deprecated = ec2faultwrap logrequest authenticate cloudrequest validator ec2executor
keystone = ec2faultwrap logrequest ec2keystoneauth cloudrequest validator ec2executor

[filter:ec2faultwrap]
paste.filter_factory = nova.api.ec2:FaultWrapper.factory

[filter:logrequest]
paste.filter_factory = nova.api.ec2:RequestLogging.factory

[filter:ec2lockout]
paste.filter_factory = nova.api.ec2:Lockout.factory

[filter:totoken]
paste.filter_factory = nova.api.ec2:EC2Token.factory

[filter:ec2keystoneauth]
paste.filter_factory = nova.api.ec2:EC2KeystoneAuth.factory

[filter:ec2noauth]
paste.filter_factory = nova.api.ec2:NoAuth.factory

[filter:authenticate]
paste.filter_factory = nova.api.ec2:Authenticate.factory

[filter:cloudrequest]
controller = nova.api.ec2.cloud.CloudController
paste.filter_factory = nova.api.ec2:Requestify.factory
```

```
[filter:authorizer]
paste.filter_factory = nova.api.ec2:Authorizer.factory

[filter:validator]
paste.filter_factory = nova.api.ec2:Validator.factory

[app:ec2executor]
paste.app_factory = nova.api.ec2:Executor.factory

##############
# Openstack #
##############

[composite:osapi_compute]
use = call:nova.api.openstack.urlmap:urlmap_factory
/: oscomputeversions
/v1.1: openstack_compute_api_v2
/v2: openstack_compute_api_v2

[composite:osapi_volume]
use = call:nova.api.openstack.urlmap:urlmap_factory
/: osvolumeversions
/v1: openstack_volume_api_v1

[composite:openstack_compute_api_v2]
use = call:nova.api.auth:pipeline_factory
noauth = faultwrap noauth ratelimit osapi_compute_app_v2
deprecated = faultwrap auth ratelimit osapi_compute_app_v2
keystone = faultwrap authtoken keystonecontext ratelimit osapi_compute_app_v2
keystone_nolimit = faultwrap authtoken keystonecontext osapi_compute_app_v2

[composite:openstack_volume_api_v1]
use = call:nova.api.auth:pipeline_factory
noauth = faultwrap noauth ratelimit osapi_volume_app_v1
deprecated = faultwrap auth ratelimit osapi_volume_app_v1
keystone = faultwrap authtoken keystonecontext ratelimit osapi_volume_app_v1
keystone_nolimit = faultwrap authtoken keystonecontext osapi_volume_app_v1

[filter:faultwrap]
paste.filter_factory = nova.api.openstack:FaultWrapper.factory

[filter:auth]
paste.filter_factory = nova.api.openstack.auth:AuthMiddleware.factory
```

```
[filter:noauth]
paste.filter_factory = nova.api.openstack.auth:NoAuthMiddleware.factory

[filter:ratelimit]
paste.filter_factory = nova.api.openstack.compute.limits:RateLimitingMiddleware.factory

[app:osapi_compute_app_v2]
paste.app_factory = nova.api.openstack.compute:APIRouter.factory

[pipeline:oscomputeversions]
pipeline = faultwrap oscomputeversionapp

[app:osapi_volume_app_v1]
paste.app_factory = nova.api.openstack.volume:APIRouter.factory

[app:oscomputeversionapp]
paste.app_factory = nova.api.openstack.compute.versions:Versions.factory

[pipeline:osvolumeversions]
pipeline = faultwrap osvolumeversionapp

[app:osvolumeversionapp]
paste.app_factory = nova.api.openstack.volume.versions:Versions.factory

##########
# Shared #
##########

[filter:keystonecontext]
paste.filter_factory = nova.api.auth:NovaKeystoneContext.factory

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
service_protocol = http
service_host = 60.12.206.105
service_port = 5000
auth_host = 60.12.206.105
auth_port = 35357
auth_protocol = http
auth_uri = http://60.12.206.105:5000/
admin_tenant_name = tenant
admin_user = nova
admin_password = service
```

在/etc/nova 建立 policy.json 作为 NOVA 服务策略文件，内容如下：

```
{
    "admin_or_owner":  [["role:admin"], ["project_id:%(project_id)s"]],
    "default": [["rule:admin_or_owner"]],


    "compute:create": [],
    "compute:create:attach_network": [],
    "compute:create:attach_volume": [],
    "compute:get_all": [],


    "admin_api": [["role:admin"]],
    "compute_extension:accounts": [["rule:admin_api"]],
    "compute_extension:admin_actions": [["rule:admin_api"]],
    "compute_extension:admin_actions:pause": [["rule:admin_or_owner"]],
    "compute_extension:admin_actions:unpause": [["rule:admin_or_owner"]],
    "compute_extension:admin_actions:suspend": [["rule:admin_or_owner"]],
    "compute_extension:admin_actions:resume": [["rule:admin_or_owner"]],
    "compute_extension:admin_actions:lock": [["rule:admin_api"]],
    "compute_extension:admin_actions:unlock": [["rule:admin_api"]],
    "compute_extension:admin_actions:resetNetwork": [["rule:admin_api"]],
    "compute_extension:admin_actions:injectNetworkInfo": [["rule:admin_api"]],
    "compute_extension:admin_actions:createBackup": [["rule:admin_or_owner"]],
    "compute_extension:admin_actions:migrateLive": [["rule:admin_api"]],
    "compute_extension:admin_actions:migrate": [["rule:admin_api"]],
    "compute_extension:aggregates": [["rule:admin_api"]],
    "compute_extension:certificates": [],
    "compute_extension:cloudpipe": [["rule:admin_api"]],
    "compute_extension:console_output": [],
    "compute_extension:consoles": [],
    "compute_extension:createserverext": [],
    "compute_extension:deferred_delete": [],
    "compute_extension:disk_config": [],
    "compute_extension:extended_server_attributes": [["rule:admin_api"]],
    "compute_extension:extended_status": [],
    "compute_extension:flavorextradata": [],
    "compute_extension:flavorextraspecs": [],
    "compute_extension:flavormanage": [["rule:admin_api"]],
    "compute_extension:floating_ip_dns": [],
    "compute_extension:floating_ip_pools": [],
    "compute_extension:floating_ips": [],
    "compute_extension:hosts": [["rule:admin_api"]],
    "compute_extension:keypairs": [],
```

```
"compute_extension:multinic": [],
"compute_extension:networks": [["rule:admin_api"]],
"compute_extension:quotas": [],
"compute_extension:rescue": [],
"compute_extension:security_groups": [],
"compute_extension:server_action_list": [["rule:admin_api"]],
"compute_extension:server_diagnostics": [["rule:admin_api"]],
"compute_extension:simple_tenant_usage:show": [["rule:admin_or_owner"]],
"compute_extension:simple_tenant_usage:list": [["rule:admin_api"]],
"compute_extension:users": [["rule:admin_api"]],
"compute_extension:virtual_interfaces": [],
"compute_extension:virtual_storage_arrays": [],
"compute_extension:volumes": [],
"compute_extension:volumetypes": [],


"volume:create": [],
"volume:get_all": [],
"volume:get_volume_metadata": [],
"volume:get_snapshot": [],
"volume:get_all_snapshots": [],


"network:get_all_networks": [],
"network:get_network": [],
"network:delete_network": [],
"network:disassociate_network": [],
"network:get_vifs_by_instance": [],
"network:allocate_for_instance": [],
"network:deallocate_for_instance": [],
"network:validate_networks": [],
"network:get_instance_uuids_by_ip_filter": [],

"network:get_floating_ip": [],
"network:get_floating_ip_pools": [],
"network:get_floating_ip_by_address": [],
"network:get_floating_ips_by_project": [],
"network:get_floating_ips_by_fixed_address": [],
"network:allocate_floating_ip": [],
"network:deallocate_floating_ip": [],
"network:associate_floating_ip": [],
"network:disassociate_floating_ip": [],

"network:get_fixed_ip": [],
```

```
    "network:add_fixed_ip_to_instance":  [],
    "network:remove_fixed_ip_from_instance":  [],
    "network:add_network_to_project":  [],
    "network:get_instance_nw_info":  [],

    "network:get_dns_domains":  [],
    "network:add_dns_entry":  [],
    "network:modify_dns_entry":  [],
    "network:delete_dns_entry":  [],
    "network:get_dns_entries_by_address":  [],
    "network:get_dns_entries_by_name":  [],
    "network:create_private_dns_domain":  [],
    "network:create_public_dns_domain":  [],
    "network:delete_dns_domain":  []
}

# 在/etc/init.d/下建立名为 nova-api 的 NOVA-API 服务启动脚本，内容如下：
#!/bin/sh
#
# openstack-nova-api    OpenStack Nova  API Server
#
# chkconfig:    - 20 80
# description: At the heart of the cloud framework is an API Server. \
#                  This API Server makes command and control of the       \
#                  hypervisor, storage, and networking programmatically  \
#                  available to users in realization of the definition    \
#                  of cloud computing.

### BEGIN INIT INFO
# Provides:
# Required-Start: $remote_fs $network $syslog
# Required-Stop: $remote_fs $syslog
# Default-Stop: 0 1 6
# Short-Description: OpenStack Nova API Server
# Description: At the heart of the cloud framework is an API Server.
#                  This API Server makes command and control of the
#                  hypervisor, storage, and networking programmatically
#                  available to users in realization of the definition
#                  of cloud computing.
### END INIT INFO

. /etc/rc.d/init.d/functions

suffix=api
```

```
        prog=openstack-nova-$suffix
        exec="/usr/bin/nova-$suffix"
        config="/etc/nova/nova.conf"
        pidfile="/var/run/nova/nova-$suffix.pid"
        logfile="/var/log/nova/$suffix.log"

        [ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

        lockfile=/var/lock/nova/$prog

        start() {
            [ -x $exec ] || exit 5
            [ -f $config ] || exit 6
            echo -n $"Starting $prog: "
            daemon --user nova --pidfile $pidfile "$exec --config-file=$config --logfile=$logfile &>/dev/nu
ll & echo \$! > $pidfile"
            retval=$?
            echo
            [ $retval -eq 0 ] && touch $lockfile
            return $retval
        }

        stop() {
            echo -n $"Stopping $prog: "
            killproc -p $pidfile $prog
            retval=$?
            echo
            [ $retval -eq 0 ] && rm -f $lockfile
            return $retval
        }

        restart() {
            stop
            start
        }

        reload() {
            restart
        }

        force_reload() {
            restart
        }
```

```
rh_status() {
    status -p $pidfile $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
        exit 2
esac
exit $?

# 在/etc/init.d/下建立名为 nova-network 的 NOVA-NETWORK 服务启动脚本，内容如下：
#!/bin/sh
#
```

```
# openstack-nova-network   OpenStack Nova Network Controller
#
# chkconfig:    - 20 80
# description: The Network Controller manages the networking resources \
#              on host machines. The API server dispatches commands    \
#              through the message queue, which are subsequently       \
#              processed by Network Controllers.                       \
#              Specific operations include:                            \
#              * Allocate Fixed IP Addresses                           \
#              * Configuring VLANs for projects                        \
#              * Configuring networks for compute nodes                \

### BEGIN INIT INFO
# Provides:
# Required-Start: $remote_fs $network $syslog
# Required-Stop: $remote_fs $syslog
# Default-Stop: 0 1 6
# Short-Description: OpenStack Nova Network Controller
# Description: The Network Controller manages the networking resources
#              on host machines. The API server dispatches commands
#              through the message queue, which are subsequently
#              processed by Network Controllers.
#              Specific operations include:
#              * Allocate Fixed IP Addresses
#              * Configuring VLANs for projects
#              * Configuring networks for compute nodes
### END INIT INFO

. /etc/rc.d/init.d/functions

suffix=network
prog=openstack-nova-$suffix
exec="/usr/bin/nova-$suffix"
config="/etc/nova/nova.conf"
pidfile="/var/run/nova/nova-$suffix.pid"
logfile="/var/log/nova/$suffix.log"

[ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

lockfile=/var/lock/nova/$prog

start() {
    [ -x $exec ] || exit 5
    [ -f $config ] || exit 6
```

```
        echo -n $"Starting $prog: "
        daemon --user nova --pidfile $pidfile "$exec --config-file=$config --logfile=$logfile &>/dev/nu
ll & echo \$! > $pidfile"
        retval=$?
        echo
        [ $retval -eq 0 ] && touch $lockfile
        return $retval
    }

    stop() {
        echo -n $"Stopping $prog: "
        killproc -p $pidfile $prog
        retval=$?
        echo
        [ $retval -eq 0 ] && rm -f $lockfile
        return $retval
    }

    restart() {
        stop
        start
    }

    reload() {
        restart
    }

    force_reload() {
        restart
    }

    rh_status() {
        status -p $pidfile $prog
    }

    rh_status_q() {
        rh_status >/dev/null 2>&1
    }


    case "$1" in
        start)
            rh_status_q && exit 0
            $1
```

```
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
        exit 2
esac
exit $?

# 在/etc/init.d/下建立名为 nova-objectstore 的 NOVA-OBJECTSTORE 服务启动脚本，内容如下：
#!/bin/sh
#
# openstack-nova-objectstore  OpenStack Nova Object Storage
#
# chkconfig:    - 20 80
# description: Implementation of an S3-like storage server based on local files.

### BEGIN INIT INFO
# Provides:
# Required-Start: $remote_fs $network $syslog
# Required-Stop: $remote_fs $syslog
# Default-Stop: 0 1 6
# Short-Description: OpenStack Nova Object Storage
# Description: Implementation of an S3-like storage server based on local files.
### END INIT INFO
```

```
. /etc/rc.d/init.d/functions

suffix=objectstore
prog=openstack-nova-$suffix
exec="/usr/bin/nova-$suffix"
config="/etc/nova/nova.conf"
pidfile="/var/run/nova/nova-$suffix.pid"
logfile="/var/log/nova/$suffix.log"

[ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

lockfile=/var/lock/nova/$prog

start() {
    [ -x $exec ] || exit 5
    [ -f $config ] || exit 6
    echo -n $"Starting $prog: "
    daemon --user nova --pidfile $pidfile "$exec --config-file=$config --logfile=$logfile &>/dev/nu
ll & echo \$! > $pidfile"
    retval=$?
    echo
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}

stop() {
    echo -n $"Stopping $prog: "
    killproc -p $pidfile $prog
    retval=$?
    echo
    [ $retval -eq 0 ] && rm -f $lockfile
    return $retval
}

restart() {
    stop
    start
}

reload() {
    restart
}
```

```
force_reload() {
    restart
}

rh_status() {
    status -p $pidfile $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
        exit 2
esac
exit $?
```

```
# 在/etc/init.d/下建立名为 nova-scheduler 的 NOVA-SCHEDULER 服务启动脚本，内容如下：
#!/bin/sh
#
# openstack-nova-scheduler  OpenStack Nova Scheduler
#
# chkconfig:   - 20 80
# description: Determines which physical hardware to allocate to a virtual resource

### BEGIN INIT INFO
# Provides:
# Required-Start: $remote_fs $network $syslog
# Required-Stop: $remote_fs $syslog
# Default-Stop: 0 1 6
# Short-Description: OpenStack Nova Scheduler
# Description: Determines which physical hardware to allocate to a virtual resource
### END INIT INFO

. /etc/rc.d/init.d/functions

suffix=scheduler
prog=openstack-nova-$suffix
exec="/usr/bin/nova-$suffix"
config="/etc/nova/nova.conf"
pidfile="/var/run/nova/nova-$suffix.pid"
logfile="/var/log/nova/$suffix.log"

[ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

lockfile=/var/lock/nova/$prog

start() {
    [ -x $exec ] || exit 5
    [ -f $config ] || exit 6
    echo -n $"Starting $prog: "
    daemon --user nova --pidfile $pidfile "$exec --config-file=$config --logfile=$logfile &>/dev/nu
ll & echo \$! > $pidfile"
    retval=$?
    echo
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}

stop() {
```

```
    echo -n $"Stopping $prog: "
    killproc -p $pidfile $prog
    retval=$?
    echo
    [ $retval -eq 0 ] && rm -f $lockfile
    return $retval
}

restart() {
    stop
    start
}

reload() {
    restart
}

force_reload() {
    restart
}

rh_status() {
    status -p $pidfile $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        rh_status_q || exit 7
```

```
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
        exit 2
esac
exit $?

# 在/etc/init.d/下建立名为 nova-volume 的 NOVA-VOLUME 服务启动脚本，内容如下：
#!/bin/sh
#
# openstack-nova-volume   OpenStack Nova Volume Worker
#
# chkconfig:   - 20 80
# description:  Volume Workers interact with iSCSI storage to manage    \
#               LVM-based instance volumes. Specific functions include: \
#               * Create Volumes                                        \
#               * Delete Volumes                                        \
#               * Establish Compute volumes

### BEGIN INIT INFO
# Provides:
# Required-Start: $remote_fs $network $syslog
# Required-Stop: $remote_fs $syslog
# Default-Stop: 0 1 6
# Short-Description: OpenStack Nova Volume Worker
# Description:  Volume Workers interact with iSCSI storage to manage
#               LVM-based instance volumes. Specific functions include:
#               * Create Volumes
#               * Delete Volumes
#               * Establish Compute volumes
### END INIT INFO

. /etc/rc.d/init.d/functions
```

```
        suffix=volume
        prog=openstack-nova-$suffix
        exec="/usr/bin/nova-$suffix"
        config="/etc/nova/nova.conf"
        pidfile="/var/run/nova/nova-$suffix.pid"
        logfile="/var/log/nova/$suffix.log"

        [ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

        lockfile=/var/lock/nova/$prog

        start() {
            [ -x $exec ] || exit 5
            [ -f $config ] || exit 6
            echo -n $"Starting $prog: "
            daemon --user nova --pidfile $pidfile "$exec --config-file=$config --logfile=$logfile &>/dev/nu
ll & echo \$! > $pidfile"
            retval=$?
            echo
            [ $retval -eq 0 ] && touch $lockfile
            return $retval
        }

        stop() {
            echo -n $"Stopping $prog: "
            killproc -p $pidfile $prog
            retval=$?
            echo
            [ $retval -eq 0 ] && rm -f $lockfile
            return $retval
        }

        restart() {
            stop
            start
        }

        reload() {
            restart
        }

        force_reload() {
            restart
```

```
}

rh_status() {
    status -p $pidfile $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
        exit 2
esac
exit $?

# 配置启动脚本：
```

```
chmod 755 /etc/init.d/nova-api
chmod 755 /etc/init.d/nova-network
chmod 755 /etc/init.d/nova-objectstore
chmod 755 /etc/init.d/nova-scheduler
chmod 755 /etc/init.d/nova-volume
mkdir /var/run/nova
mkdir -p /var/lib/nova/instances
mkdir /var/lock/nova
chown nova:root /var/run/nova
chown -R nova:nova /var/lib/nova
chown nova:root /var/lock/nova

# 配置 sudo
在/etc/sudoers.d/建立 nova 文件，内容如下：
Defaults:nova !requiretty

Cmnd_Alias NOVACMDS = /bin/aoe-stat,                                    \
                      /bin/chmod,                                       \
                      /bin/chmod /var/lib/nova/tmp/*/root/.ssh,         \
                      /bin/chown,                                       \
                      /bin/chown /var/lib/nova/tmp/*/root/.ssh,         \
                      /bin/dd,                                          \
                      /bin/kill,                                        \
                      /bin/mkdir,                                       \
                      /bin/mount,                                       \
                      /bin/umount,                                      \
                      /sbin/aoe-discover,                               \
                      /sbin/ifconfig,                                   \
                      /sbin/ip,                                         \
                      /sbin/ip6tables-restore,                          \
                      /sbin/ip6tables-save,                             \
                      /sbin/iptables,                                   \
                      /sbin/iptables-restore,                           \
                      /sbin/iptables-save,                              \
                      /sbin/iscsiadm,                                   \
                      /sbin/kpartx,                                     \
                      /sbin/losetup,                                    \
                      /sbin/lvcreate,                                   \
                      /sbin/lvdisplay,                                  \
                      /sbin/lvremove,                                   \
                      /sbin/pvcreate,                                   \
                      /sbin/route,                                      \
                      /sbin/tune2fs,                                    \
                      /sbin/vconfig,                                    \
```

```
                /sbin/vgcreate,                 \
                /sbin/vgs,                      \
                /usr/bin/fusermount,            \
                /usr/bin/guestmount,            \
                /usr/bin/socat,                 \
                /bin/cat,                       \
                /usr/bin/tee,                   \
                /usr/bin/qemu-nbd,              \
                /usr/bin/virsh,                 \
                /usr/sbin/brctl,                \
                /usr/sbin/dnsmasq,              \
                /usr/sbin/ietadm,               \
                /usr/sbin/radvd,                \
                /usr/sbin/tgtadm,               \
                /usr/sbin/vblade-persist

nova ALL = (root) NOPASSWD: SETENV: NOVACMDS
chmod 0440 /etc/sudoers.d/nova

# 配置 polkit 策略
在/etc/polkit-1/localauthority/50-local.d/建立 50-nova.pkla，内容如下：
[Allow nova libvirt management permissions]
Identity=unix-user:nova
Action=org.libvirt.unix.manage
ResultAny=yes
ResultInactive=yes
ResultActive=yes

# 建立 NOVA 服务数据库结构
nova-manage db sync

# 安装 iscsitarget
wget http://sourceforge.net/projects/iscsitarget/files/iscsitarget/1.4.20.2/iscsitarget-1.4.20.2.tar.gz/dow
nload -P /opt
cd /opt
tar xf iscsitarget-1.4.20.2.tar.gz
cd iscsitarget-1.4.20.2
make
make install
/etc/init.d/iscsi-target start
netstat -ltnp 查看是否有 tcp 3260 端口监听

# 建立 nova-volumes 卷
fdisk /dev/sdb
```

n

p

1

两次回车

t

83

w

```
mkfs.ext4  /dev/sdb1
vgcreate nova-volumes  /dev/sdb1

# 启动 NOVA 相关服务
/etc/init.d/nova-api  start
/etc/init.d/nova-network  start
/etc/init.d/nova-objectstore start
/etc/init.d/nova-scheduler  start
/etc/init.d/nova-volume  start
```

检测服务是否正常启动

通过 netstat -ltunp 查看是否有 tcp 3333、8773、8774、8775、8776 端口监听

如果没有正常启动请查看/var/log/nova 目录下相关文件排错

SWIFT 对象存储服务配置

```
# 建立 SWIFT 服务配置文件存放目录
mkdir  /etc/swift

# 建立 SWIFT 服务启动用户
useradd  -s  /sbin/nologin  -m  -d  /var/log/swift  swift

# 格式化硬盘及挂载
yum  -y  install xfsprogs
mkfs.xfs  -f  -i  size=1024 /dev/sdc
mkfs.xfs  -f  -i  size=1024 /dev/sdd
mkdir  -p  /swift/drivers/sd{c,d}
mount  -t  xfs  -o  noatime,nodiratime,nobarrier,logbufs=8  /dev/sdc  /swift/drivers/sdc
mount  -t  xfs  -o  noatime,nodiratime,nobarrier,logbufs=8  /dev/sdd  /swift/drivers/sdd
echo  -e  '/dev/sdc\t/swift/drivers/sdc\txfs\tnoatime,nodiratime,nobarrier,logbufs=8\t0  0' >>/etc/fstab
echo  -e  '/dev/sdd\t/swift/drivers/sdd\txfs\tnoatime,nodiratime,nobarrier,logbufs=8\t0  0' >>/etc/fstab

# swift 同步相关配置
mkdir  -p  /swift/node/sd{c,d}
ln  -sv  /swift/drivers/sdc  /swift/node/
```

ln -sv /swift/drivers/sdd /swift/node/

在/etc 下建立 rsyncd.conf 文件，内容如下：

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = 192.168.1.2

[account6012]
max connections = 50
path = /swift/node/sdc
read only = false
lock file = /var/lock/account6012.lock

[account6022]
max connections = 50
path = /swift/node/sdd
read only = false
lock file = /var/lock/account6022.lock

[container6011]
max connections = 50
path = /swift/node/sdc
read only = false
lock file = /var/lock/container6011.lock

[container6021]
max connections = 50
path = /swift/node/sdd
read only = false
lock file = /var/lock/container6021.lock

[object6010]
max connections = 50
path = /swift/node/sdc
read only = false
lock file = /var/lock/object6010.lock

[object6020]
max connections = 50
path = /swift/node/sdd
read only = false
lock file = /var/lock/object6020.lock
```

```
yum -y install xinetd
sed -i '/disable/s#yes#no#g' /etc/xinetd.d/rsync
/etc/init.d/xinetd start
mkdir -p /etc/swift/{object,container,account}-server
```

在/etc/swift 下建立 swift.conf 文件，内容如下：

```
[swift-hash]
swift_hash_path_suffix = changeme
```

在/etc/swift 下建立 proxy-server.conf 文件，内容如下：

```
[DEFAULT]
bind_port = 8080
user = swift
swift_dir = /etc/swift
workers = 8
log_name = swift
log_facility = LOG_LOCAL1
log_level = DEBUG

[pipeline:main]
pipeline = healthcheck cache swift3 s3token authtoken keystone proxy-server

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true

[filter:keystone]
paste.filter_factory = keystone.middleware.swift_auth:filter_factory
operator_roles = Member,admin,SwiftOperator

# NOTE(chmou): s3token middleware is not updated yet to use only
# username and password.
[filter:s3token]
paste.filter_factory = keystone.middleware.s3_token:filter_factory
service_port = 60.12.206.105
service_host = 5000
auth_host = 60.12.206.105
auth_port = 35357
auth_protocol = http
auth_token = ADMIN
admin_token = ADMIN
```

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host = 60.12.206.105
auth_port = 35357
auth_protocol = http
auth_uri = http://60.12.206.105:5000/
admin_tenant_name = tenant
admin_user = swift
admin_password = service

[filter:swift3]
use = egg:swift#swift3

[filter:healthcheck]
use = egg:swift#healthcheck

[filter:cache]
use = egg:swift#memcache

在/etc/swift/account-server 下建立 sdc.conf 和 sdd.conf 文件，内容如下：
----------sdc.conf----------
[DEFAULT]
devices = /swift/node/sdc
mount_check = false
bind_port = 6012
user = swift
log_facility = LOG_LOCAL0
swift_dir = /etc/swift

[pipeline:main]
pipeline = account-server

[app:account-server]
use = egg:swift#account

[account-replicator]
vm_test_mode = yes

[account-auditor]

[account-reaper]

----------sdd.conf----------
[DEFAULT]

devices = /swift/node/sdd
mount_check = false
bind_port = 6022
user = swift
log_facility = LOG_LOCAL0
swift_dir = /etc/swift

[pipeline:main]
pipeline = account-server

[app:account-server]
use = egg:swift#account

[account-replicator]
vm_test_mode = yes

[account-auditor]

[account-reaper]

在/etc/swift/container-server 下建立 sdc.conf 和 sdd.conf 文件，内容如下：
----------sdc.conf----------
[DEFAULT]
devices = /swift/node/sdc
mount_check = false
bind_port = 6011
user = swift
log_facility = LOG_LOCAL0
swift_dir = /etc/swift

[pipeline:main]
pipeline = container-server

[app:container-server]
use = egg:swift#container

[container-replicator]
vm_test_mode = yes

[container-updater]

[container-auditor]

[container-sync]

----------sdd.conf----------

```
[DEFAULT]
devices = /swift/node/sdd
mount_check = false
bind_port = 6021
user = swift
log_facility = LOG_LOCAL0
swift_dir = /etc/swift

[pipeline:main]
pipeline = container-server

[app:container-server]
use = egg:swift#container

[container-replicator]
vm_test_mode = yes

[container-updater]

[container-auditor]

[container-sync]
```

在/etc/swift/object-server 下建立 sdc.conf 和 sdd.conf 文件，内容如下：

----------sdc.conf----------

```
[DEFAULT]
devices = /swift/node/sdc
mount_check = false
bind_port = 6010
user = swift
log_facility = LOG_LOCAL0
swift_dir = /etc/swift

[pipeline:main]
pipeline = object-server

[app:object-server]
use = egg:swift#object

[object-replicator]
vm_test_mode = yes
```

```
[object-updater]

[object-auditor]

[object-expirer]

--------------------sdd.conf--------------------
[DEFAULT]
devices = /swift/node/sdd
mount_check = false
bind_port = 6020
user = swift
log_facility = LOG_LOCAL0
swift_dir = /etc/swift

[pipeline:main]
pipeline = object-server

[app:object-server]
use = egg:swift#object

[object-replicator]
vm_test_mode = yes

[object-updater]

[object-auditor]

[object-expirer]

# 建立 ring
cd /etc/swift
swift-ring-builder object.builder create 8 2 1
swift-ring-builder object.builder add z1-192.168.1.2:6010/sdc 1
swift-ring-builder object.builder add z2-192.168.1.2:6020/sdd 1
swift-ring-builder object.builder rebalance

swift-ring-builder container.builder create 8 2 1
swift-ring-builder container.builder add z1-192.168.1.2:6011/sdc 1
swift-ring-builder container.builder add z2-192.168.1.2:6021/sdd 1
swift-ring-builder container.builder rebalance

swift-ring-builder account.builder create 8 2 1
swift-ring-builder account.builder add z1-192.168.1.2:6012/sdc 1
```

```
swift-ring-builder account.builder add z2-192.168.1.2:6022/sdd 1
swift-ring-builder account.builder rebalance

# 建立各服务启动脚本
在/etc/swift 下建立 functions 文件，内容如下：
. /etc/rc.d/init.d/functions

swift_action() {
  retval=0
  server="$1"
  call="swift_$2"

  if [[ -f "/etc/swift/$server-server.conf" ]]; then
    $call "$server" \
          "/etc/swift/$server-server.conf" \
          "/var/run/swift/$server-server.pid"
    [ $? -ne 0 ] && retval=1
  elif [[ -d "/etc/swift/$server-server/" ]]; then
    declare -i count=0
    for name in $( ls "/etc/swift/$server-server/" ); do
      $call "$server" \
            "/etc/swift/$server-server/$name" \
            "/var/run/swift/$server-server/$count.pid"
      [ $? -ne 0 ] && retval=1
      count=$count+1
    done
  fi
  return $retval
}

swift_start() {
  name="$1"
  long_name="$name-server"
  conf_file="$2"
  pid_file="$3"

  ulimit -n ${SWIFT_MAX_FILES-32768}
  echo -n "Starting swift-$long_name: "
  daemon --pidfile $pid_file \
    "/usr/bin/swift-$long_name $conf_file &>/var/log/swift-startup.log & echo \$! > $pid_file"
  retval=$?
  echo
  return $retval
}
```

```
swift_stop() {
    name="$1"
    long_name="$name-server"
    conf_name="$2"
    pid_file="$3"

    echo -n "Stopping swift-$long_name: "
    killproc -p $pid_file -d ${SWIFT_STOP_DELAY-15} $long_name
    retval=$?
    echo
    return $retval
}

swift_status() {
    name="$1"
    long_name="$name-server"
    conf_name="$2"
    pid_file="$3"

    status -p $pid_file $long_name
}
```

在/etc/init.d 下建立 swift-proxy 文件，内容如下：

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:          openstack-swift-proxy
# Required-Start:    $remote_fs
# Required-Stop:     $remote_fs
# Default-Stop:      0 1 6
# Short-Description: Swift proxy server
# Description:       Account server for swift.
### END INIT INFO

# openstack-swift-proxy: swift proxy server
#
# chkconfig: - 20 80
# description: Proxy server for swift.

. /etc/rc.d/init.d/functions
. /etc/swift/functions

name="proxy"
```

```
[ -e "/etc/sysconfig/openstack-swift-$name" ] && . "/etc/sysconfig/openstack-swift-$name"

lockfile="/var/lock/swift/openstack-swift-proxy"

start() {
    swift_action "$name" start
    retval=$?
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}

stop() {
    swift_action "$name" stop
    retval=$?
    [ $retval -eq 0 ] && rm -f $lockfile
    return $retval
}

restart() {
    stop
    start
}

rh_status() {
    swift_action "$name" status
}

rh_status_q() {
    rh_status &> /dev/null
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
```

```
        ;;
    reload)
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart}"
        exit 2
esac
exit $?
```

在/etc/init.d 下建立 swift-account 文件，内容如下：

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:          openstack-swift-account
# Required-Start:    $remote_fs
# Required-Stop:     $remote_fs
# Default-Stop:      0 1 6
# Short-Description: Swift account server
# Description:       Account server for swift.
### END INIT INFO

# openstack-swift-account: swift account server
#
# chkconfig: - 20 80
# description: Account server for swift.

. /etc/rc.d/init.d/functions
. /etc/swift/functions

name="account"

[ -e "/etc/sysconfig/openstack-swift-$name" ] && . "/etc/sysconfig/openstack-swift-$name"

lockfile="/var/lock/swift/openstack-swift-account"

start() {
    swift_action "$name" start
```

```
    retval=$?
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}

stop() {
    swift_action "$name" stop
    retval=$?
    [ $retval -eq 0 ] && rm -f $lockfile
    return $retval
}

restart() {
    stop
    start
}

rh_status() {
    swift_action "$name" status
}

rh_status_q() {
    rh_status &> /dev/null
}

case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
```

```
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart}"
        exit 2
esac
exit $?
```

在/etc/init.d 下建立 swift-container 文件，内容如下：

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:          openstack-swift-container
# Required-Start:    $remote_fs
# Required-Stop:     $remote_fs
# Default-Stop:      0 1 6
# Short-Description: Swift container server
# Description:       Container server for swift.
### END INIT INFO

# openstack-swift-container: swift container server
#
# chkconfig: - 20 80
# description: Container server for swift.

. /etc/rc.d/init.d/functions
. /etc/swift/functions

name="container"

[ -e "/etc/sysconfig/openstack-swift-$name" ] && . "/etc/sysconfig/openstack-swift-$name"

lockfile="/var/lock/swift/openstack-swift-container"

start() {
    swift_action "$name" start
    retval=$?
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}

stop() {
    swift_action "$name" stop
    retval=$?
```

```
    [ $retval -eq 0 ] && rm -f $lockfile
    return $retval
}

restart() {
    stop
    start
}

rh_status() {
    swift_action "$name" status
}

rh_status_q() {
    rh_status &> /dev/null
}

case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart}"
        exit 2
esac
exit $?
```

在/etc/init.d 下建立 swift-object 文件，内容如下：

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:          openstack-swift-object
# Required-Start:    $remote_fs
# Required-Stop:     $remote_fs
# Default-Stop:      0 1 6
# Short-Description: Swift object server
# Description:       Object server for swift.
### END INIT INFO

# openstack-swift-object: swift object server
#
# chkconfig: - 20 80
# description: Object server for swift.

. /etc/rc.d/init.d/functions
. /etc/swift/functions

name="object"

[ -e "/etc/sysconfig/openstack-swift-$name" ] && . "/etc/sysconfig/openstack-swift-$name"

lockfile="/var/lock/swift/openstack-swift-object"

start() {
    swift_action "$name" start
    retval=$?
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}

stop() {
    swift_action "$name" stop
    retval=$?
    [ $retval -eq 0 ] && rm -f $lockfile
    return $retval
}

restart() {
    stop
    start
}
```

```
rh_status() {
    swift_action "$name" status
}

rh_status_q() {
    rh_status &> /dev/null
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart}"
        exit 2
esac
exit $?

# 设置权限
chown -R swift:swift /swift /etc/swift

# 配置启动脚本：
chmod 755 /etc/init.d/swift-proxy
chmod 755 /etc/init.d/swift-account
chmod 755 /etc/init.d/swift-container
```

```
chmod 755 /etc/init.d/swift-object
mkdir /var/run/swift
mkdir /var/lock/swift
chown swift:root /var/run/swift
chown swift:root /var/lock/swift

# 启动服务
/etc/init.d/swift-proxy start
/etc/init.d/swift-account start
/etc/init.d/swift-container start
/etc/init.d/swift-object start
```

HORIZON 管理面板配置

```
# 建立 KEYSTONE 服务数据库
mysql -uroot -popenstack -e 'create database dashboard'

# 配置 apache
编辑/etc/httpd/conf.d/django.conf，更改成如下内容：
WSGISocketPrefix /tmp/horizon
<VirtualHost *:80>
    WSGIScriptAlias / /opt/horizon-2012.1/openstack_dashboard/wsgi/django.wsgi
    WSGIDaemonProcess horizon user=apache group=apache processes=3 threads=10
    SetEnv APACHE_RUN_USER apache
    SetEnv APACHE_RUN_GROUP apache
    WSGIProcessGroup horizon

    DocumentRoot /opt/horizon-2012.1/.blackhole/
    Alias /media /opt/horizon-2012.1/openstack_dashboard/static

    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>

    <Directory /opt/horizon-2012.1/>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
    </Directory>

    ErrorLog /var/log/httpd/error.log
```

```
    LogLevel warn
    CustomLog /var/log/httpd/access.log combined
</VirtualHost>

mkdir /opt/horizon-2012.1/.blackhole

# 配置 HORIZON
在/opt/horizon-2012.1/openstack_dashboard/local 下建立 local_settings.py 文件，内容如下：
import os

DEBUG = False
TEMPLATE_DEBUG = DEBUG
PROD = False
USE_SSL = False

LOCAL_PATH = os.path.dirname(os.path.abspath(__file__))

# FIXME: We need to change this to mysql, instead of sqlite.
DATABASES = {
    'default': {
    'ENGINE': 'django.db.backends.mysql',
      'NAME': 'dashboard',
      'USER': 'root',
      'PASSWORD': 'openstack',
      'HOST': 'localhost',
      'PORT': '3306',
    },
}

# The default values for these two settings seem to cause issues with apache
CACHE_BACKEND = 'dummy://'
SESSION_ENGINE = 'django.contrib.sessions.backends.cached_db'

# Send email to the console by default
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
# Or send them to /dev/null
#EMAIL_BACKEND = 'django.core.mail.backends.dummy.EmailBackend'

# django-mailer uses a different settings attribute
MAILER_EMAIL_BACKEND = EMAIL_BACKEND

# Configure these for your outgoing email host
# EMAIL_HOST = 'smtp.my-company.com'
# EMAIL_PORT = 25
```

```
# EMAIL_HOST_USER = 'djangomail'
# EMAIL_HOST_PASSWORD = 'top-secret!'

HORIZON_CONFIG = {
    'dashboards': ('nova', 'syspanel', 'settings',),
    'default_dashboard': 'nova',
    'user_home': 'openstack_dashboard.views.user_home',
}

# TODO(tres): Remove these once Keystone has an API to identify auth backend.
OPENSTACK_KEYSTONE_BACKEND = {
    'name': 'native',
    'can_edit_user': True
}

OPENSTACK_HOST = "60.12.206.105"
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v2.0" % OPENSTACK_HOST
# FIXME: this is only needed until keystone fixes its GET /tenants call
# so that it doesn't return everything for admins
OPENSTACK_KEYSTONE_ADMIN_URL = "http://%s:35357/v2.0" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "Member"

SWIFT_PAGINATE_LIMIT = 100

# If you have external monitoring links, eg:
# EXTERNAL_MONITORING = [
#     ['Nagios','http://foo.com'],
#     ['Ganglia','http://bar.com'],
# ]

#LOGGING = {
#         'version': 1,
#         # When set to True this will disable all logging except
#         # for loggers specified in this configuration dictionary. Note that
#         # if nothing is specified here and disable_existing_loggers is True,
#         # django.db.backends will still log unless it is disabled explicitly.
#         'disable_existing_loggers': False,
#         'handlers': {
#             'null': {
#                 'level': 'DEBUG',
#                 'class': 'django.utils.log.NullHandler',
#                 },
#             'console': {
#                 # Set the level to "DEBUG" for verbose output logging.
```

```
#                     'level': 'INFO',
#                     'class': 'logging.StreamHandler',
#                     },
#                 },
#             'loggers': {
#                 # Logging from django.db.backends is VERY verbose, send to null
#                 # by default.
#                 'django.db.backends': {
#                     'handlers': ['null'],
#                     'propagate': False,
#                     },
#                 'horizon': {
#                     'handlers': ['console'],
#                     'propagate': False,
#                 },
#                 'novaclient': {
#                     'handlers': ['console'],
#                     'propagate': False,
#                 },
#                 'keystoneclient': {
#                     'handlers': ['console'],
#                     'propagate': False,
#                 },
#                 'nose.plugins.manager': {
#                     'handlers': ['console'],
#                     'propagate': False,
#                 }
#             }
#}
```

静态化 django

编辑/opt/horizon-2012.1/openstack_dashboard/urls.py，在最下面添加如下内容：

```
if settings.DEBUG is False:
    urlpatterns += patterns('',
        url(r'^static/(?P<path>.*)$', 'django.views.static.serve', {
            'document_root': settings.STATIC_ROOT,
        }),
   )
```

python /opt/horizon-2012.1/manage.py collectstatic，选择 yes

建立 HORIZON 数据库结构

python /opt/horizon-2012.1/manage.py syncdb

从启 apache 服务

chown -R apache:apache /opt/horizon-2012.1

/etc/init.d/httpd restart

NOVNC WEB 访问配置

编辑/etc/nova/nova.conf 文件，添加如下内容：

novncproxy_base_url=http://$my_ip:6080/vnc_auto.html

vnc_enabled=true

vnc_keymap=en-us

vncserver_listen=$my_ip

vncserver_proxyclient_address=$my_ip

将 NOVNC 执行程序添加到环境变量中

ln -sv /opt/noVNC/utils/nova-novncproxy /usr/bin/

在/etc/init.d/下建立名为 nova-novncproxy 的 NOVNC 服务启动脚本，内容如下：

```
#!/bin/sh
#
# openstack-nova-novncproxy     OpenStack Nova VNC Web Console
#
# chkconfig:      - 20 80
# description: OpenStack Nova VNC Web Console

### BEGIN INIT INFO
# Provides:
# Required-Start: $remote_fs $network $syslog
# Required-Stop: $remote_fs $syslog
# Default-Stop: 0 1 6
# Short-Description: OpenStack Nova VNC Web Console
# Description: OpenStack Nova VNC Web Console
### END INIT INFO

. /etc/rc.d/init.d/functions

suffix=novncproxy
prog=openstack-nova-$suffix
web="/opt/noVNC"
exec="/usr/bin/nova-$suffix"
config="/etc/nova/nova.conf"
pidfile="/var/run/nova/nova-$suffix.pid"
logfile="/var/log/nova/$suffix.log"
```

```
    [ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

    lockfile=/var/lock/nova/$prog

    start() {
        [ -x $exec ] || exit 5
        [ -f $config ] || exit 6
        echo -n $"Starting $prog: "
        daemon --user nova --pidfile $pidfile "$exec --config-file=$config --web $web --logfile=$logfile
--daemon &>/dev/null & echo \$! > $pidfile"
        retval=$?
        echo
        [ $retval -eq 0 ] && touch $lockfile
        return $retval
    }

    stop() {
        echo -n $"Stopping $prog: "
        killproc -p $pidfile $prog
        retval=$?
        echo
        [ $retval -eq 0 ] && rm -f $lockfile
        return $retval
    }

    restart() {
        stop
        start
    }

    reload() {
        restart
    }

    force_reload() {
        restart
    }

    rh_status() {
        status -p $pidfile $prog
    }

    rh_status_q() {
        rh_status >/dev/null 2>&1
```

```
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
        exit 2
esac
exit $?

#在/etc/init.d/下建立名为 nova-consoleauth 的控制台认证启动脚本，内容如下：
#!/bin/sh
#
# openstack-nova-novncproxy     OpenStack Nova Console Auth
#
# chkconfig:    - 20 80
# description: OpenStack Nova Console Auth

### BEGIN INIT INFO
```

```
        # Provides:
        # Required-Start: $remote_fs $network $syslog
        # Required-Stop: $remote_fs $syslog
        # Default-Stop: 0 1 6
        # Short-Description: OpenStack Nova Console Auth
        # Description: OpenStack Nova Console Auth
        ### END INIT INFO

        . /etc/rc.d/init.d/functions

        suffix=consoleauth
        prog=openstack-nova-$suffix
        exec="/usr/bin/nova-$suffix"
        config="/etc/nova/nova.conf"
        pidfile="/var/run/nova/nova-$suffix.pid"
        logfile="/var/log/nova/$suffix.log"

        [ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

        lockfile=/var/lock/nova/$prog

        start() {
            [ -x $exec ] || exit 5
            [ -f $config ] || exit 6
            echo -n $"Starting $prog: "
            daemon --user nova --pidfile $pidfile "$exec --config-file=$config --logfile=$logfile &>/dev/null &
echo \$! > $pidfile"
            retval=$?
            echo
            [ $retval -eq 0 ] && touch $lockfile
            return $retval
        }

        stop() {
            echo -n $"Stopping $prog: "
            killproc -p $pidfile $prog
            retval=$?
            echo
            [ $retval -eq 0 ] && rm -f $lockfile
            return $retval
        }

        restart() {
            stop
```

```
    start
}

reload() {
    restart
}

force_reload() {
    restart
}

rh_status() {
    status -p $pidfile $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
```

```
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
        exit 2
esac
exit $?

# 配置启动脚本
chmod  755  /etc/init.d/nova-novncproxy
chmod 755 /etc/init.d/nova-consoleauth

# 启动 GLANCE-API 和 GLANCE-REGISTRY 服务
/etc/init.d/nova-novncproxy  start
/etc/init.d/nova-consoleauth  start

# 检测服务是否正常启动
通过 netstat -ltunp 查看是否有 tcp 6080 端口监听
如果没有正常启动请查看/var/log/nova 目录下相关文件排错
```

计算节点安装

前提工作

1. 导入第三方软件源

   rpm -Uvh http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-5.noarch.rpm

   rpm -Uvh http://pkgs.repoforge.org/rpmforge-release/rpmforge-release-0.5.2-2.el6.rf.x86_64.rpm

2. 安装依赖包

   yum -y install swig libvirt-python libvirt qemu-kvm gcc make gcc-c++ patch m4 python-devel libxml2-devel libxslt-devel libgsasl-devel openldap-devel sqlite-devel openssl-devel wget telnet gpxe-bootimgs gpxe-roms gpxe-roms-qemu dmidecode git scsi-target-utils kpartx socat vconfig aoetools python-pip

   rpm -Uvh http://veillard.com/libvirt/6.3/x86_64/dnsmasq-utils-2.48-6.el6.x86_64.rpm

   ln -sv /usr/bin/pip-python /usr/bin/pip

NTP 时钟同步配置

1. 安装 NTP 相关命令包

   yum -y install ntpdate

   跟控制节点同步时间并写入硬件

   ntpdate 192.168.1.2

   hwclock -w

2. 将时间同步添加到计划任务

   echo '30 8 * * * root /usr/sbin/ntpdate 192.168.1.2;hwclock -w' >>/etc/crontab

PYTHON-NOVACLIENT 库安装

4. 下载源码包

   wget https://launchpad.net/nova/essex/2012.1/+download/python-novaclient-2012.1.tar.gz -P /opt

5. 安装依赖包

   yum -y install python-simplejson python-prettytable python-argparse python-nose1.1 python-httplib2 python-virtualenv MySQL-python

6. 解压并安装 PYTHON-NOVACLIENT 库

cd /opt

tar xf python-novaclient-2012.1.tar.gz

cd python-novaclient-2012.1

python setup.py install

rm -f ../python-novaclient-2012.1.tar.gz

GLANCE 库文件安装

4. 下载源码包

wget https://launchpad.net/glance/essex/2012.1/+download/glance-2012.1.tar.gz -P /opt

5. 安装依赖包

yum install -y python-anyjson python-kombu

pip install xattr==0.6.0 iso8601==0.1.4 pysendfile==2.0.0 pycrypto==2.3 wsgiref boto==2.1.1

6. 解压并安装 GLANCE 镜像存储服务

cd /opt

tar xf glance-2012.1.tar.gz

cd glance-2012.1

python setup.py install-lib

rm -f ../glance-2012.1.tar.gz

NOVA 计算服务安装

4. 下载源码包

wget https://launchpad.net/nova/essex/2012.1/+download/nova-2012.1.tar.gz -P /opt

5. 安装依赖包

yum install -y python-amqplib python-carrot python-lockfile python-gflags python-netaddr python-suds python-paramiko python-feedparser python-eventlet python-greenlet python-paste

pip install pycrypto==2.3 Cheetah==2.4.4 python-daemon==1.5.5 Babel==0.9.6 routes==1.12.3 lxml==2.3 PasteDeploy==1.5.0 sqlalchemy-migrate==0.7.2 SQLAlchemy==0.7.3 WebOb==1.0.8

6. 解压并安装 NOVA 计算服务

cd /opt

tar xf nova-2012.1.tar.gz

cd nova-2012.1

python setup.py install

rm -f ../nova-2012.1.tar.gz

NOVA 计算服务配置

建立 NOVA 服务配置文件存放目录

mkdir /etc/nova

建立 NOVA 服务启动用户

useradd -s /sbin/nologin -m -d /var/log/nova nova

在/etc/nova 建立 nova.conf 作为 NOVA 服务配置文件，内容如下：

[DEFAULT]

auth_strategy=keystone

bindir=/usr/bin

pybasedir=/var/lib/nova

connection_type=libvirt

debug=True

lock_path=/var/lock/nova

log-dir=/var/log/nova

my_ip=60.12.206.99

ec2_host=60.12.206.105

ec2_path=/services/Cloud

ec2_port=8773

ec2_scheme=http

glance_host=60.12.206.105

glance_port=9292

glance_api_servers=$glance_host:$glance_port

image_service=nova.image.glance.GlanceImageService

metadata_host=60.12.206.105

metadata_port=8775

network_manager=nova.network.manager.FlatDHCPManager

osapi_path=/v1.1/

osapi_scheme=http

rabbit_host=192.168.1.2

rabbit_password=openstack

rabbit_port=5672

rabbit_userid=guest

root_helper=sudo

s3_host=60.12.206.105

s3_port=3333

sql_connection=mysql://root:openstack@192.168.1.2/nova

state_path=/var/lib/nova

use_ipv6=False

use-syslog=False

verbose=True

ec2_listen=60.12.206.105

```
ec2_listen_port=8773
metadata_listen=60.12.206.105
metadata_listen_port=8775
osapi_compute_listen=60.12.206.105
osapi_compute_listen_port=8774
osapi_volume_listen=60.12.206.105
osapi_volume_listen_port=8776
keystone_ec2_url=http://60.12.206.105:5000/v2.0/ec2tokens
dhcpbridge=$bindir/nova-dhcpbridge
dhcpbridge_flagfile=/etc/nova/nova.conf
public_interface=eth0
routing_source_ip=$my_ip
fixed_range=10.0.0.0/24
flat_interface=br1
flat_network_bridge=eth1
force_dhcp_release=True
libvirt_type=kvm
libvirt_use_virtio_for_bridges=True
iscsi_helper=ietadm
iscsi_ip_address=60.12.206.105
vncserver_listen=$my_ip
vncserver_proxyclient_address=$my_ip
novncproxy_base_url=http://60.12.206.105:6080/vnc_auto.html

# 在/etc/init.d/下建立名为 nova-compute 的 NOVA-COMPUTE 服务启动脚本，内容如下：
#!/bin/sh
#
# openstack-nova-compute   OpenStack Nova Compute Worker
#
# chkconfig:    - 20 80
# description: Compute workers manage computing instances on host \
#                machines. Through the API, commands are dispatched \
#                to compute workers to:                              \
#                * Run instances                                     \
#                * Terminate instances                               \
#                * Reboot instances                                  \
#                * Attach volumes                                    \
#                * Detach volumes                                    \
#                * Get console output

### BEGIN INIT INFO
# Provides:
# Required-Start: $remote_fs $network $syslog
# Required-Stop: $remote_fs $syslog
```

```
        # Default-Stop: 0 1 6
        # Short-Description: OpenStack Nova Compute Worker
        # Description: Compute workers manage computing instances on host
        #                  machines. Through the API, commands are dispatched
        #                  to compute workers to:
        #                  * Run instances
        #                  * Terminate instances
        #                  * Reboot instances
        #                  * Attach volumes
        #                  * Detach volumes
        #                  * Get console output
        ### END INIT INFO

        . /etc/rc.d/init.d/functions

        suffix=compute
        prog=openstack-nova-$suffix
        exec="/usr/bin/nova-$suffix"
        config="/etc/nova/nova.conf"
        pidfile="/var/run/nova/nova-$suffix.pid"
        logfile="/var/log/nova/$suffix.log"

        [ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

        lockfile=/var/lock/nova/$prog

        start() {
            [ -x $exec ] || exit 5
            [ -f $config ] || exit 6
            echo -n $"Starting $prog: "
            daemon --user nova --pidfile $pidfile "$exec --config-file=$config --logfile=$logfile &>/dev/nu
ll & echo \$! > $pidfile"
            retval=$?
            echo
            [ $retval -eq 0 ] && touch $lockfile
            return $retval
        }

        stop() {
            echo -n $"Stopping $prog: "
            killproc -p $pidfile $prog
            retval=$?
            echo
            [ $retval -eq 0 ] && rm -f $lockfile
```

```
    return $retval
}

restart() {
    stop
    start
}

reload() {
    restart
}

force_reload() {
    restart
}

rh_status() {
    status -p $pidfile $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
```

```
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
        exit 2
esac
exit $?

# 在/etc/init.d/下建立名为 nova-network 的 NOVA-NETWORK 服务启动脚本，内容如下：
#!/bin/sh
#
# openstack-nova-network   OpenStack Nova Network Controller
#
# chkconfig:    - 20 80
# description: The Network Controller manages the networking resources \
#              on host machines. The API server dispatches commands     \
#              through the message queue, which are subsequently        \
#              processed by Network Controllers.                        \
#              Specific operations include:                             \
#              * Allocate Fixed IP Addresses                            \
#              * Configuring VLANs for projects                         \
#              * Configuring networks for compute nodes                 \

### BEGIN INIT INFO
# Provides:
# Required-Start: $remote_fs $network $syslog
# Required-Stop: $remote_fs $syslog
# Default-Stop: 0 1 6
# Short-Description: OpenStack Nova Network Controller
# Description: The Network Controller manages the networking resources
#              on host machines. The API server dispatches commands
#              through the message queue, which are subsequently
#              processed by Network Controllers.
#              Specific operations include:
#              * Allocate Fixed IP Addresses
#              * Configuring VLANs for projects
#              * Configuring networks for compute nodes
### END INIT INFO
```

```
. /etc/rc.d/init.d/functions

suffix=network
prog=openstack-nova-$suffix
exec="/usr/bin/nova-$suffix"
config="/etc/nova/nova.conf"
pidfile="/var/run/nova/nova-$suffix.pid"
logfile="/var/log/nova/$suffix.log"

[ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

lockfile=/var/lock/nova/$prog

start() {
    [ -x $exec ] || exit 5
    [ -f $config ] || exit 6
    echo -n $"Starting $prog: "
    daemon --user nova --pidfile $pidfile "$exec --config-file=$config --logfile=$logfile &>/dev/nu
ll & echo \$! > $pidfile"
    retval=$?
    echo
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}

stop() {
    echo -n $"Stopping $prog: "
    killproc -p $pidfile $prog
    retval=$?
    echo
    [ $retval -eq 0 ] && rm -f $lockfile
    return $retval
}

restart() {
    stop
    start
}

reload() {
    restart
}

force_reload() {
```

```
    restart
}

rh_status() {
    status -p $pidfile $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}


case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart)
        $1
        ;;
    reload)
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload}"
        exit 2
esac
exit $?
```

配置 sudo

在/etc/sudoers.d/建立 nova 文件，内容如下：

```
Defaults:nova !requiretty

Cmnd_Alias NOVACMDS = /bin/aoe-stat,                                    \
                      /bin/chmod,                                       \
                      /bin/chmod /var/lib/nova/tmp/*/root/.ssh,         \
                      /bin/chown,                                       \
                      /bin/chown /var/lib/nova/tmp/*/root/.ssh,         \
                      /bin/dd,                                          \
                      /bin/kill,                                        \
                      /bin/mkdir,                                       \
                      /bin/mount,                                       \
                      /bin/umount,                                      \
                      /sbin/aoe-discover,                               \
                      /sbin/ifconfig,                                   \
                      /sbin/ip,                                         \
                      /sbin/ip6tables-restore,                          \
                      /sbin/ip6tables-save,                             \
                      /sbin/iptables,                                   \
                      /sbin/iptables-restore,                           \
                      /sbin/iptables-save,                              \
                      /sbin/iscsiadm,                                   \
                      /sbin/kpartx,                                     \
                      /sbin/losetup,                                    \
                      /sbin/lvcreate,                                   \
                      /sbin/lvdisplay,                                  \
                      /sbin/lvremove,                                   \
                      /sbin/pvcreate,                                   \
                      /sbin/route,                                      \
                      /sbin/tune2fs,                                    \
                      /sbin/vconfig,                                    \
                      /sbin/vgcreate,                                   \
                      /sbin/vgs,                                        \
                      /usr/bin/fusermount,                              \
                      /usr/bin/guestmount,                              \
                      /usr/bin/socat,                                   \
                      /bin/cat,                                         \
                      /usr/bin/tee,                                     \
                      /usr/bin/qemu-nbd,                                \
                      /usr/bin/virsh,                                   \
                      /usr/sbin/brctl,                                  \
                      /usr/sbin/dnsmasq,                                \
```

```
                /usr/sbin/ietadm,                \
                /usr/sbin/radvd,                 \
                /usr/sbin/tgtadm,                \
                /usr/sbin/vblade-persist

nova ALL = (root) NOPASSWD: SETENV: NOVACMDS
chmod 0440 /etc/sudoers.d/nova

# 配置 polkit 策略
在/etc/polkit-1/localauthority/50-local.d/建立 50-nova.pkla，内容如下：
[Allow nova libvirt management permissions]
Identity=unix-user:nova
Action=org.libvirt.unix.manage
ResultAny=yes
ResultInactive=yes
ResultActive=yes

# 配置启动脚本：
chmod 755 /etc/init.d/nova-compute
chmod 755 /etc/init.d/nova-network
mkdir /var/run/nova
mkdir -p /var/lib/nova/instances
mkdir /var/lock/nova
chown nova:root /var/run/nova
chown -R nova:nova /var/lib/nova
chown nova:root /var/lock/nova

# 配置 MYSQL 数据库
在控制节点 mysql 执行如下语句：
grant all on nova.* to root@'192.168.1.%' identified by 'openstack';

# 启动 NOVA 相关服务
/etc/init.d/nova-compute start
/etc/init.d/nova-network start

# 更改 iptables 允许 vnc 连接
iptables -I INPUT -d 60.12.206.99 -p tcp -m multiport --dports 5900:6000 -j ACCEPT
```