

一、采用DevStack安装

DevStack是一个自动安装OpenStack的工具，使用OpenStack的源码进行安装。采用DevStack部署OpenStack的顺序如下：

1.安装Git、pip

```
yum -y install git
###安装yum扩展源
yum -y install epel-release
yum -y install python-pip
```

可以使用浙大的epel源

```
1 # 所有节点安装浙大、交大的epel源
2 rpm -Uvh http://ftp.sjtu.edu.cn/fedora/epel/epel-release-latest-7.noarch.r
3 rpm -Uvh http://mirrors.zju.edu.cn/epel/epel-release-latest-7.noarch.rpm
```

配置pip：

在指定用户（root和随后创建的stack）的家目录下创建.pip/pip.conf文件配置国内的python源信息

```
1 [global]
2 timeout = 6000
3 index-url = http://pypi.douban.com/simple/
4 trusted-host = pypi.douban.com
```

2.下载安装DevStack

进入home目录clone源代码

```
1 #####主干代码:
```

```
2 git clone http://git.trystack.cn/openstack-dev/devstack.git -b master
3 #####指定分支
4 git clone http://git.trystack.cn/openstack-dev/devstack.git -b stable/
```

3.为OpenStack创建Stack用户

由于DevStack提供了创建该用户的脚本，我们无需手工创建（第一次运行脚本会安装：redhat-lsb-core.x86_64 0:4.1-27.el7.centos.1）

```
1 cd /home/devstack/tools
2 ./create-stack-user.sh
3 chmod 777 /dev/pts/0
4 chown -R stack:stack /home/devstack
5 passwd stack
```

可以考虑个stack用户添加sudo权限

3.配置devstack的安装信息

```
/home/devstack/local.conf
```

由于devstack通过源代码安装，因此我们在local.conf中需要配置一个git的国内镜像

```
1 # use TryStack git mirror
2 GIT_BASE=http://git.trystack.cn
3 NOVNC_REPO=http://git.trystack.cn/kanaka/noVNC.git
4 SPICE_REPO=http://git.trystack.cn/git/spice/spice-html5.git
5 LOG_COLOR=False
```

配置文件local.conf的安装说明：

<http://www.chenshake.com/local-conf-devstack-profile-parameter-description/>

```
1 修改stackrc中的GIT_BASE为：
```

```
2 GIT_BASE=${GIT_BASE:-git://git.trystack.cn}
```

4.执行安装

```
/home/devstack/stack.sh
```

5.安装完成后

要关闭防火墙，否则dashboard无法访问。执行命令

```
service iptables stop
```

安装完成后修改stackrc，防止每次执行stack.sh都进行依赖下载。

```
1 #OFFLINE=$(trueorfalse False OFFLINE)
2 OFFLINE=True
```

6.重启方案

```
screen -c stack-screenrc
systemctl restart httpd
```

二、通过dashboard创建VM

dashboard是OpenStack自带的管理UI，通过他可以在浏览器上管理OpenStack集群。

创建VM步骤

1.登录dashboard，其中\${ip}为在local.conf中配置的HOST_IP

```
http://${ip}/dashboard
```

devstack为我们默认创建了两个用户（admin、demo），密码均为nomoresecret
登录后，可以看到页面中包含项目、管理员、身份管理三个tab页签。Octave的界面的具体功能可以查询官方的说明文档（<https://docs.openstack.org/admin->

2.单击项目->计算->实例->创建实例，依次填写以下信息

创建实例

详情

源 *

实例类型 *

网络 *

网络接口

安全组

密钥对

配置

服务器组

scheduler hint

元数据

请提供实例的主机名，欲部署的可用区域和数量。 增大数量以创建多个同样配置的实例。

实例名称 *
linqing

可用域
nova

数量 *
1

实例总计 (最大10)
10%
0 当前用量
1 已添加
9 剩余量

取消

返回

下一步

创建实例

创建实例

详情

源

实例类型 *

网络 *

网络接口

安全组

密钥对

配置

服务器组

scheduler hint

元数据

实例的源是用来创建实例的模板。 可以使用一个镜像、一个实例的快照（镜像快照）、一个卷或一个卷快照（如果启用这个功能）。 您也可以通过创建一个新卷来选择使用具有持久性的存储。

选择源
镜像

创建新卷
是 不

存储卷大小(GB) *
1

删除实例时删除卷
是 不

已分配

名称	已更新	大小	类型	可见性
> cirros-0.3.5-x86_64-disk	4/18/17 5:25 PM	12.65 MB	qcow2	公有

可用配额 0

选择一个

Q 点击这里过滤

名称	已更新	大小	类型	可见性
没有可选项				

取消

返回

下一步

创建实例

创建实例

详情

源

实例类型

网络

网络接口

安全组

密钥对

配置

服务器组

scheduler hint

元数据

类型管理实例的计算、内存和存储容量的大小。

已分配

名称	虚拟内核	内存	磁盘总计	根磁盘	临时磁盘	公有
> m1.tiny	1	512 MB	1 GB	1 GB	0 GB	是

可用配额 11

选择一个

点击这里过滤

名称	虚拟内核	内存	磁盘总计	根磁盘	临时磁盘	公有
> m1.small	1	2 GB	20 GB	20 GB	0 GB	是
> m1.medium	2	4 GB	40 GB	40 GB	0 GB	是
> m1.large	4	8 GB	80 GB	80 GB	0 GB	是
> m1.nano	1	64 MB	0 GB	0 GB	0 GB	是
> m1.xlarge	8	16 GB	160 GB	160 GB	0 GB	是
> m1.micro	1	128 MB	0 GB	0 GB	0 GB	是
> cirros256	1	256 MB	0 GB	0 GB	0 GB	是
> ds512M	1	512 MB	5 GB	5 GB	0 GB	是

取消

返回

下一步

创建实例

3.按照上面步骤配置之后可以看到VM信息

openstack admin

项目 / 计算 / 实例

实例

显示 1 个项

实例名称	镜像名称	IP 地址	实例类型	密钥对	状态	可用域	任务	电源状态	创建后的时间	动作
linqing	cirros-0.3.5-x86_64-disk	172.24.4.8 2001:db8::3	m1.tiny	-	运行	nova	无	运行中	1 小时, 53 分钟	创建快照

正在显示 1 项

三、VM启动失败时日志信息

1.单击实例名称可以在前台获得日志信息

openstack

admin

项目

访问API

计算

概况

实例

镜像

密钥对

卷

网络

管理员

身份管理

项目 / 计算 / 实例 / linqing

linqing

创建快照

概况

日志

控制台

操作日志

名称

ID

状态

可用域

已创建

创建后的时间

主机

linqing

ec4cbbd5-e065-4e8a-94d4-21cd3848b60

运行

nova

2017年4月18日 09:57

2 小时 · 3 分钟

localhost.localdomain

规格

实例类型名称

实例类型 ID

内存

VCPU数量

磁盘

m1.tiny

1

512MB

1 VCPU

1GB

IP地址

Public

172.24.4.8, 2001:db8::3

安全组

default

允许 IPv4 to 0.0.0.0/0

允许 IPv6 to ::/0

允许 IPv6 from default

允许 IPv4 from default

元数据

2.查看后台Scheduler日志

日志位置：/opt/stack/logs/n-sch.log

3.日志的查看方式

OpenStack基于python，典型的python错误trace如下：

```
2017-04-18 17:29:42.868 [[{"[01:31mERROR oslo_messaging.rpc.server "[01:36mreq-b26530da-4e86-4e7c-aef9-a706c6a0d916 "[00:36mNone None"[01:31m "[01:35m"[01:31mException during message handling"[00m
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00mTraceback (most recent call last):
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m File "/usr/lib/python2.7/site-packages/oslo_messaging/rpc/server.py", line 157, in _process_incoming
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m res = self.dispatcher.dispatch(message)
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m File "/usr/lib/python2.7/site-packages/oslo_messaging/rpc/dispatcher.py", line 213, in dispatch
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m return self._do_dispatch(endpoint, method, ctxt, args)
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m File "/usr/lib/python2.7/site-packages/oslo_messaging/rpc/dispatcher.py", line 183, in _do_dispatch
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m result = func(ctxt, **new_args)
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m File "/opt/stack/nova/nova/scheduler/manager.py", line 139, in sync_instance_info
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m instance_uuids)
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m File "/usr/lib/python2.7/site-packages/oslo_concurrency/lockutils.py", line 271, in inner
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m return f(*args, **kwargs)
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m File "/opt/stack/nova/nova/scheduler/host_manager.py", line 811, in sync_instance_info
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m self._recreate_instance_info(context, host_name)
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m File "/opt/stack/nova/nova/scheduler/host_manager.py", line 738, in _recreate_instance_info
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m inst_dict = self._get_instances_by_host(context, host_name)
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m File "/opt/stack/nova/nova/scheduler/host_manager.py", line 710, in _get_instances_by_host
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m hm = objects.HostMapping.get_by_host(context, host_name)
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m File "/usr/lib/python2.7/site-packages/oslo_versionedobjects/base.py", line 184, in wrapper
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m result = fn(cls, context, *args, **kwargs)
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m File "/opt/stack/nova/nova/objects/host_mapping.py", line 190, in get_by_host
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m db_mapping = cls._get_by_host_from_db(context, host)
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m File "/usr/lib/python2.7/site-packages/oslo_db/sqlalchemy/enginefacade.py", line 963, in wrapper
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m return fn(*args, **kwargs)
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m File "/opt/stack/nova/nova/objects/host_mapping.py", line 95, in _get_by_host_from_db
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m raise exception.HostMappingNotFound(name=host)
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00mHostMappingNotFound: Host 'localhost.localdomain' is not mapped to any cell
[[{"[01:31m2017-04-18 17:29:42.868 TRACE oslo_messaging.rpc.server "[01:35m"[00m
```