Procedimento de instalação do OpenStack (IceHouse)

Guia baseado na documentação presente para o OpenStack Icehouse.

Sumário

Configurações Gerais de Ambiente

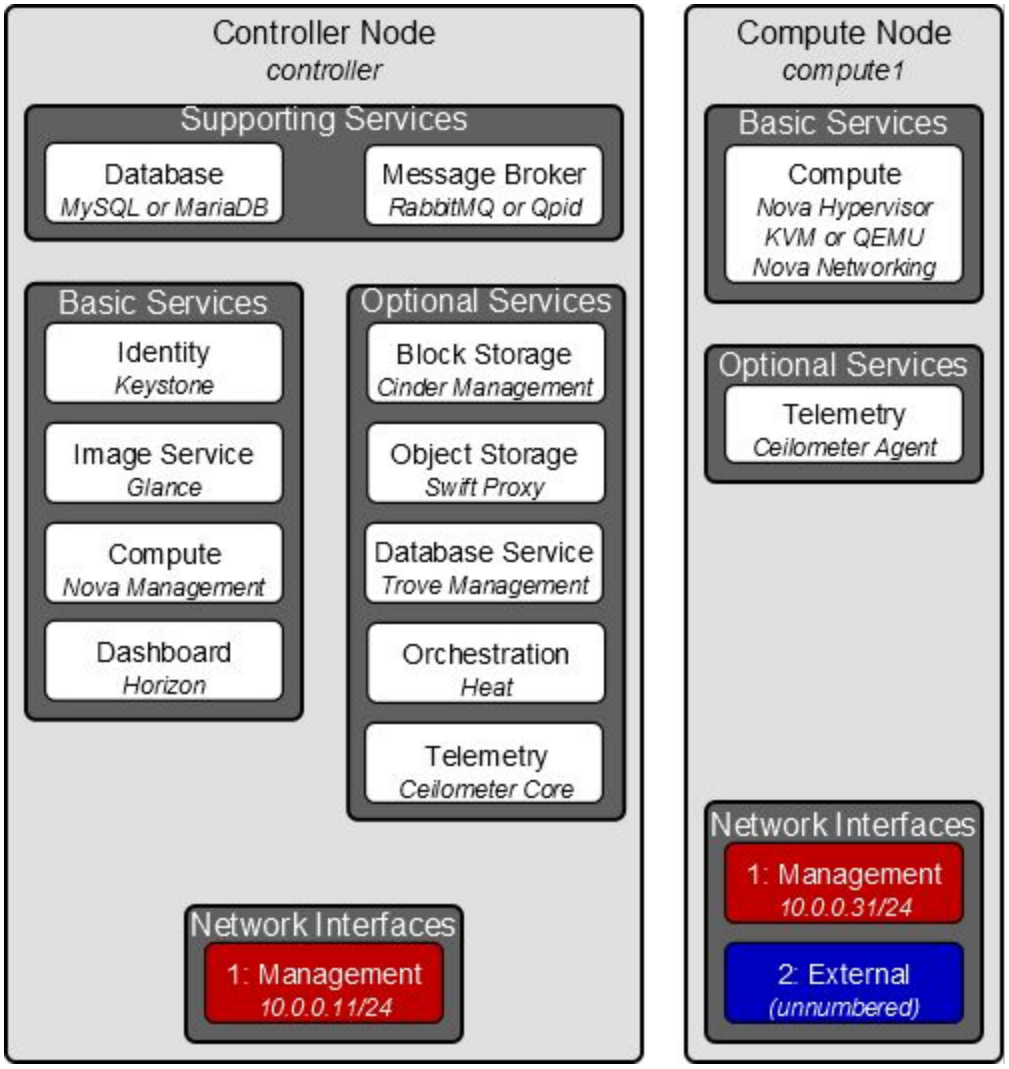
Configurações de Sistema

Embora um ambiente de produção necessite de uma especificação mais robusta, uma configuração funcional pode usar uma quantidade modesta de recursos, como:
Nó **controller**: 1 processador, 2 GB de memória RAM, e um disco de 5 GB.
Nó de **Rede**: 1 processador, 512 MB de memória RAM, e um disco de 5 GB.
Nó de **compute**: 1 processador, 2 GB de memória, e um disco de 10 GB.

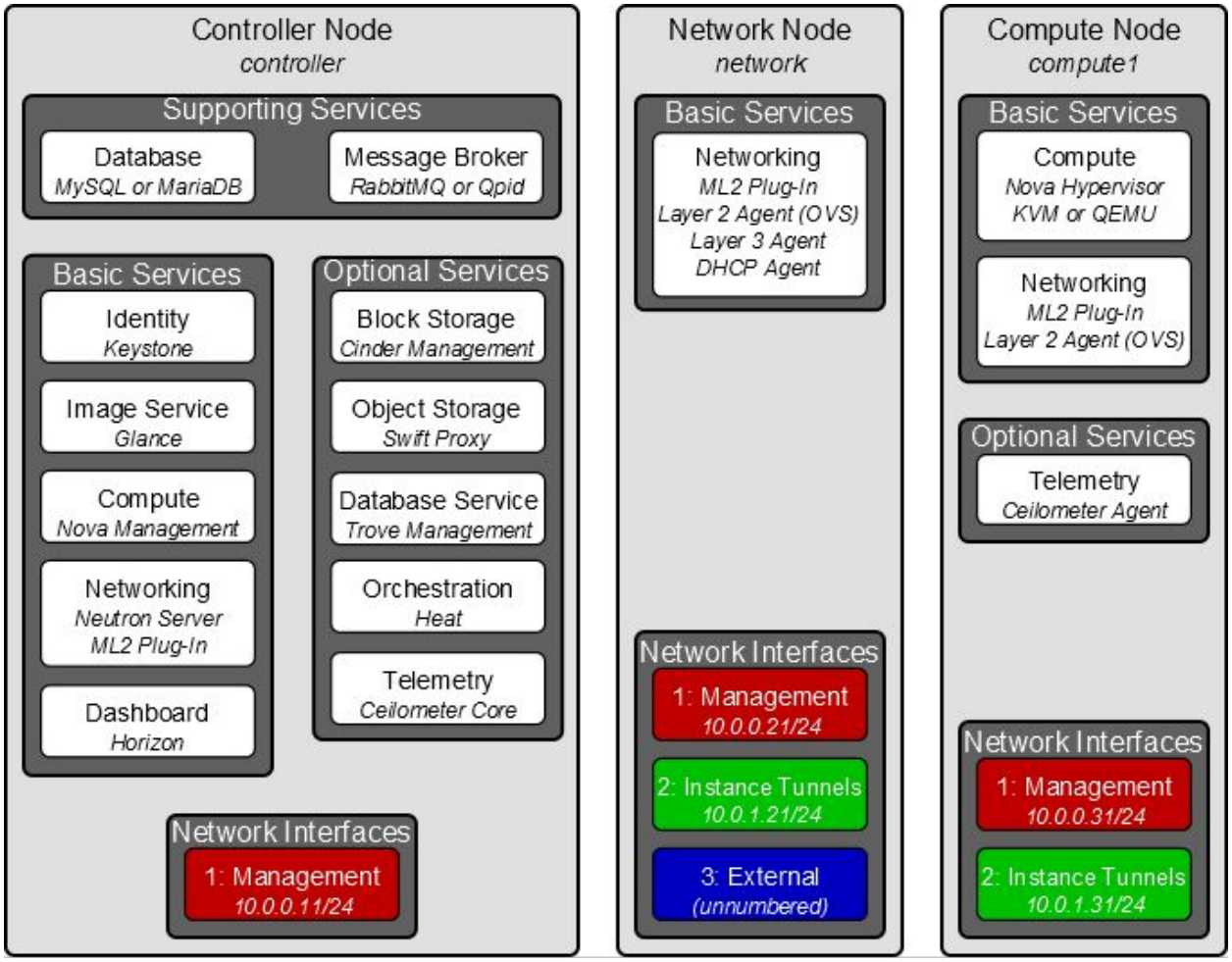
Para serviços de armazenamento, podem ser necessários discos adicionais, com tamanhos maiores, a depender da necessidade, tanto do serviço em si, como do grau de necessidade. No caso do serviço de armazenamento de bloco (o que cria Volumes para as máquinas virtuais), também é necessário usar uma partição que use um Gerenciador de Volumes Lógicos (do inglês *Logical Volume Manager* - **LVM**)

Além disso, é recomendado que seja instalado um sistema Linux de 64 bits, pois com um sistema de 32 bits não é possível iniciar uma instância a partir de uma imagem de 64 bits.

1- Rede e Arquitetura do Serviço
Normalmente, é recomendável que **todas** as máquinas usem duas interfaces de rede: uma para tráfego externo (para ter Internet, por exemplo), e uma para tráfego interno (para se comunicar com a nuvem). Note que isso se aplica tanto para o nó **controller**, quanto para os possíveis múltiplos nós de **compute**. A configuração abaixo é um exemplo, caso seja usado o serviço de rede nativo do OpenStack (**nova-network**):



Mas, para a arquitetura clássica do OpenStack, é recomendado usar o serviço de rede dedicado, conhecido pelo nome **neutron**. Esta seria a disposição dos serviços e seus endereços de rede para este caso:



OBS: Atentar aos endereços de rede.
OBS2: Se você estiver usando uma configuração de dois nós, então as interfaces que serão usadas no nó de Rede também serão aplicadas em seu nó de controller (caso seja este seu *front-end*).

Se baseando nesta disposição, cada um dos nós teria sua configuração de rede (baseando-se em uma instalação em um Sistema Operacional Linux) como:

Nó **controller**:

```
# Rede de gerenciamento
auto eth0
iface eth0 inet static
        address 10.0.0.11
        netmask 255.255.255.0
        gateway 10.0.0.1
```

E, no arquivo **/etc/hosts** , para que seja possível resolver os nomes dos outros nós:

```
# controller
10.0.0.11       controller

# network
10.0.0.21       network

# compute1
10.0.0.31       compute1
```

OBS: Comente ou remova a linha que referencie a esta mesma máquina pelo endereço 127.0.1.1

Nó de **Rede**:

```
# Rede de gerenciamento
auto eth0
iface eth0 inet static
        address 10.0.0.21
        netmask 255.255.255.0
        gateway 10.0.0.1
# Rede de tunel para as instancias
auto eth1
iface eth1 inet static
        address 10.0.1.21
        netmask 255.255.255.0
# Para fornecer acesso externo as instancias
auto eth2
iface eth2 inet manual
        up ip link set dev $IFACE up
        up ip link set dev $IFACE down
```

E, no arquivo **/etc/hosts** :

```
# network
10.0.0.21       network

# controller
10.0.0.11       controller

# compute1
10.0.0.31       compute1
```

OBS: Comente ou remova a linha que referencie a esta mesma máquina pelo endereço 127.0.1.1

Nó de **Compute**
Rede de Gerenciamento
auto eth0
iface eth0 inet static
        address 10.0.0.31
        netmask 255.255.255.0
        gateway 10.0.0.1
Rede de tunel para as instancias
auto eth1
iface eth1 inet static
        address 10.0.1.31
        netmask 255.255.255.0

compute1
10.0.0.31   compute1

controller
10.0.0.11   controller

network
10.0.0.21   network

OBS: Comente ou remova a linha que referencie a esta mesma máquina pelo endereço 127.0.1.1
OBS2: Perceba que a este nó inicial de **compute** se deu o nome compute1, de maneira a enumerar nós conseguintes que rodem este serviço.

Depois que configurar a rede para todas as máquinas, reinicie o serviço de rede com:
`# service networking restart`

Após este procedimento, pode ser testada a conectividade com o comando **ping** a partir de qualquer uma das máquinas com um endereço atribuído, para quaisquer outros dos endereços das outras máquinas.

2- Network Time Protocol (NTP)
Depois disso, precisa instalar o NTP, usado para sincronizar o horário perfeitamente entre as máquinas da nuvem. Instale com:
`# apt-get install ntp`

Note que isso significa que o **controller** normalmente é o *host*, e usará um local de fora como referência, mas as outras máquinas usarão **controller** como servidor de NTP. Isso é configurado no arquivo /etc/ntp.conf, mais especificamente a partir da linha que diz:
Use servers from the NTP Pool Project.

OBS: Referência para instalação completa do ntp
http://www.ntp.br/NTP/MenuNTPLinuxBSD

3- Notações e informes sobre senhas

Os vários serviços usados no OpenStack, como por exemplo os servidores de Banco de Dados e de Mensagens, vão exigir em algum momento uma autenticação. Uma senha aleatória para cada serviço deverá ser determinada durante a configuração de cada serviço, e para gerar senhas, pode ser usado o software **pwgen** (no apt-get o nome do pacote é este mesmo), ou pegar a saída deste comando:

```
$ openssl rand -hex 10
```

para usar como senha.

O guia usado adotou que SERVICE_PASSWORD será a senha de um serviço SERVICE, e SERVICE_DBPASS é a senha do Banco de Dados usada pelo serviço SERVICE para acessá-la. Um esquema é mostrado abaixo:

Tabela 1 - Identificação das Senhas

Nome da Senha	**Descrição**
Database password (no variable used)	Senha de *root* no banco de dados (dbpa22w0rd)
RABBIT_PASS	Senha do usuário *guest* do RabbitMQ (rabb1tdbpa22)
KEYSTONE_DBPASS	Senha do BD no Serviço de Identidade (k3yston3dbpa22)
DEMO_PASS	Senha do usuário demo (d3mopa22)
ADMIN_PASS	Senha do usuário *admin* (us3radm1npa22**)**
GLANCE_DBPASS	Senha do BD no Serviço de Imagem (glanc3dbpa22)
GLANCE_PASS	Senha do usuário *glance* no Serviço de Imagem (us3rglanc3pa22)
NOVA_DBPASS	Senha do BD no Serviço de *Compute* (n0vadbpa22)

NOVA_PASS	Senha do usuário *nova* no Serviço de *Compute* (us3rn0vapa22)
DASH_DBPASS	Senha do BD do *dashboard* (dah2dbpa22)
CINDER_DBPASS	Senha do BD no Serviço de Armazenamento em Bloco (c1nd3rdbpa22)
CINDER_PASS	Senha do usuário *cinder* no Serviço de Armazenamento em Bloco (us3rc1nd3rpa22)
SWIFT_PASS	Senha do usuario swift do Serviço de Armazenamento de Objeto (us3rsw1ftpa22)
NEUTRON_DBPASS	Senha do BD no Serviço de Rede (n3utrondbpa22)
NEUTRON_PASS	Senha do usuário *neutron* no Serviço de Rede (us3rn3utronpa22)
HEAT_DBPASS	Senha do BD no Serviço de Orquestramento (h3atdbpa22)
HEAT_PASS	Senha do usuário *heat* no Serviço de Orquestramento (us3rh3atpa22)
CEILOMETER_DBPASS	Senha do BD no Serviço de Telemetria (*CEILOMETER_DBPASS*) citta (c31l0m3t3rdbpa22)
CEILOMETER_PASS	Senha do usuário *ceilometer* no Serviço de Telemetria (us3rc3ilom3t3rpa22)
TROVE_DBPASS	Senha do Banco de Dados para o serviço de Banco de Dados (tr0v3dbpa22)
TROVE_PASS	Senha do usuário do serviço **database** *Trove* (us3rtr0v3pa22)

4- Banco de Dados

A maioria dos serviços precisa de um Banco de Dados para armazenar informação. Neste caso, o MySQL será usado, e o *host* será no **controller**. Os outros nós usarão um cliente para acessar este Banco de Dados. A configuração para cada um segue abaixo.

Para o controller
No **controller**, os pacotes com o **cliente** e **servidor** MySQL, e a biblioteca Python são instalados com:
```
# apt-get install python-mysqldb mysql-server
```

OBS: Quando estiver instalando o pacote do servidor MySQL, será pedido para elaborar uma senha de root para o Banco de Dados.

Depois, abra o arquivo /etc/mysql/my.cnf e edite o bind-address para o endereço interno do **controller**, para que este possa ser acessado pelos outros nós.
Exemplo:
[mysqld]
...
bind-address = 10.0.0.11

E, ainda nesta seção, habilite o InnoDB, o conjunto de caracteres UTF-8 e a colação UTF-8 por padrão:
[mysqld]
...
default-storage-engine = innodb
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8

Reinicie o serviço do MySQL para que as mudanças tenham efeito, com:
```
# service mysql restart
```

Quando o Banco de Dados é iniciado pela primeira vez, são criados uns usuários anônimos, que devem ser removidos, senão pode gerar problemas ao seguir o guia. Use os comandos:
```
# mysql_install_db
# mysql_secure_installation
```

E durante o procedimento que se iniciará, responda **yes** em tudo, a menos que haja uma razão específica para que não seja o caso.

Para os nós - ***Compute Nodes***

Em nós que não sejam o **controller**, os pacotes com o cliente MySQL e a biblioteca Python são instalados, com:

```
# apt-get install python-mysqldb
```

5- Pacotes e repositórios para o OpenStack

Os pacotes **OpenStack** podem ser disponibilizados nas distribuições **Ubuntu** em épocas diferentes em que são elaborados, e portanto deve ser necessário trabalhar com um repositório específico, chamado **Ubuntu Cloud Archive,** que no caso da versão em questão do **Openstack Icehouse**, é instalado com os comandos:

```
# apt-get install python-software-properties
# add-apt-repository cloud-archive:icehouse
```

Depois, é necessário atualizar a base de dados dos pacotes, atualizar o sistema, e reiniciar para as mudanças fazerem efeito, com os comandos:

```
# apt-get update
# apt-get dist-upgrade
# reboot
```

OBS: Devem ser usadas apenas versões de longo termo do Ubuntu. Neste caso, a versão **14.04 LTS** é usada.

6- Servidor de Mensagens (RabbitMQ)

No **controller**, deve ser instalado o servidor de escalonamento de mensagens. O mais comum é o RabbitMQ, mas também podem ser usados Qpid ou ZeroMQ (0MQ). Para o caso do primeiro, podemos instalar com:

```
# apt-get install rabbitmq-server
```

OBS: O RabbitMQ deixa o serviço iniciando junto com o Sistema Operacional, e também por padrão já ativa um usuário padrão, com senha padrão. Em um sistema com endereço IPv6, se não mudar a senha do *guest*, vai ser possível para qualquer um acessar de fora. Sendo assim, a senha do usuário *guest* pode ser mudada com:

```
# rabbitmqctl change_password guest RABBIT_PASS
```

1- Serviço de Identidade (Identity Server) - KEYSTONE

Os conceitos que serão usados podem ser conferidos aqui.

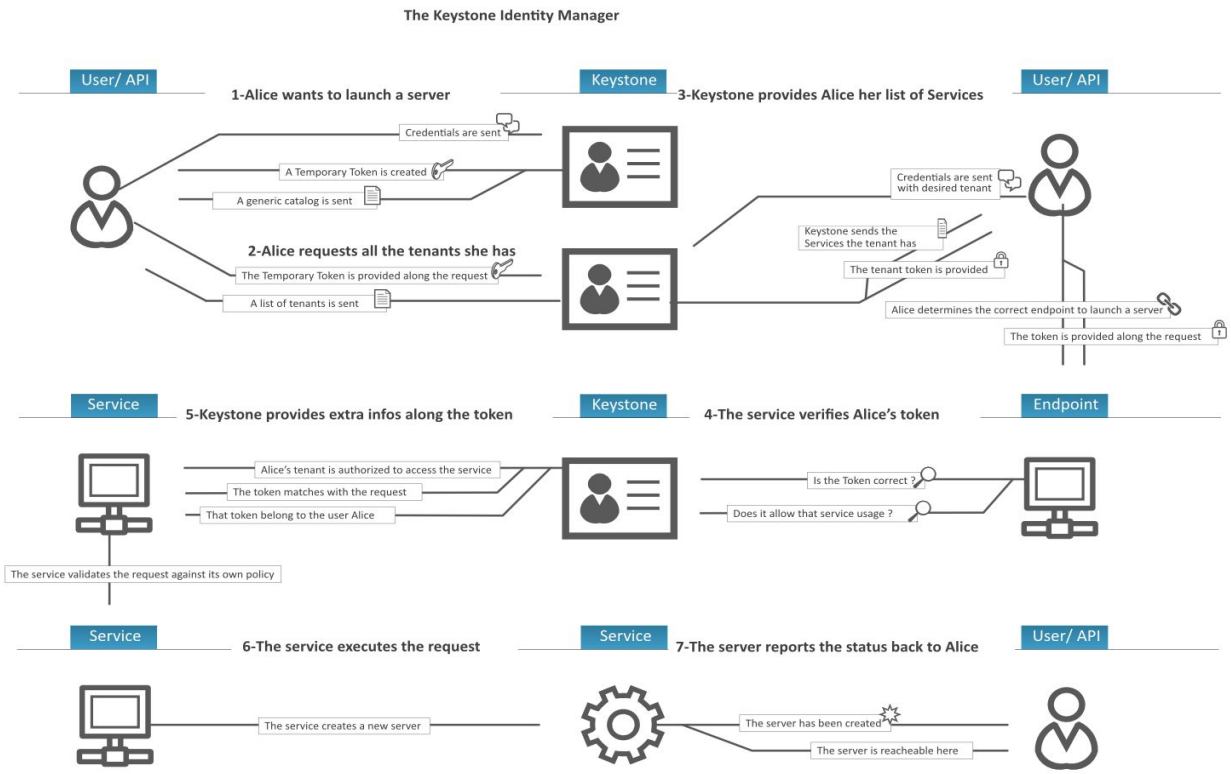
O Serviço de Identidade, chamado de **Keystone**, realiza duas funções: a primeira é gerenciar os usuários, acompanhando seu uso dos serviços, bem como suas permissões; a segunda é catalogar os serviços, fornecendo uma lista dos serviços disponíveis, bem como seus pontos de terminação na API.

Este serviço trabalha com uma terminologia para se referir a determinados elementos, que resumidamente trata com:

usuário (user) - para todos os fins, é uma pessoa que está usando os serviços do OpenStack;

credenciais (credentials) - dados que são conhecidos pelo usuário e provam que este é quem diz ser. Pode ser nome_de_usuário/senha, nome_de_usuário/chave_da_API, ou uma ficha de autenticação fornecida pelo Serviço de Identidade.
autenticação (authentication) - quando um usuário confirma sua identidade com suas credenciais. Após fazer isso, o Serviço de Identidade fornece uma ficha, que é então usada para fazer as requisições de uso para outros serviços.
ficha (token) - um texto que é usado para acessar os recursos. Cada ficha possui um escopo dos recursos que é capaz de acessar, e esta pode ser revogada a qualquer momento, assim como possui uma validade (em tempo) limitada.
alocador (tenant) - um contâiner que pode agrupar ou isolar recursos e/ou objetos de identidade. Pode levar a um cliente, conta, organização ou projeto.
serviço (service) - É algum dos serviços que constitui o OpenStack, que podem fornecer acesso a um ou mais pontos de terminação, por onde os usuários acessam recursos e realizam operações.
endpoint - um endereço acessível pela rede, geralmente através de uma URL, por onde um serviço pode ser acessado.
função (role) - uma personalidade que um usuário pode adotar que o permite realizar um conjunto bem definido de operações. Isso se traduz como um conjunto de direitos e privilégios.

Visualmente, o funcionamento do Serviço de Identidade é como descrito na imagem abaixo ou neste link.



1.1- Instalando o Serviço de Identidade

No **controller**, instale o Serviço de Identidade do OpenStack, junto com a dependência python-keystoneclient:

```
# apt-get install keystone
```

Agora, deve ser especificado o caminho do Banco de Dados no arquivo de configuração. Procure pela linha "connection =" no arquivo /etc/keystone/keystone.conf e use os dados do serviço keystone:

```
...
[database]
connection = mysql://keystone:k3yston3dbpa22@controller/keystone
…
```

O Ubuntu por padrão cria um Banco de Dados SQLite. É melhor remover (ou renomear) o arquivo keystone.db em /var/lib/keystone/ para ele não ser acessado por engano.
OBS: Em **guaricema** esse arquivo não existia.

Depois, use a senha que foi configurada ali em cima para o usuário root, e crie um usuário “keystone” no Banco de Dados:

```
# mysql -u root -p
mysql> CREATE DATABASE keystone;
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'KEYSTONE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY '*KEYSTONE_DBPASS*';
```

Cria as tabelas para o Serviço de Identidade, com:

```
# keystone-manage db_sync
```

Depois, precisa criar uma ficha para autorização secreta entre o Serviço de Identidade e os Serviços do OpenStack. A ficha pode ser criada com:

```
# openssl rand -hex 10
```

E edita o arquivo de configuração /etc/keystone/keystone.conf e muda a seção [DEFAULT] substituindo ADMIN_TOKEN pela ficha que vamos usar (no caso, o resultado do comando aí em cima):

```
[DEFAULT]
# A "shared secret" between keystone and other openstack services
admin_token = 6611276e30b859e6ef97
…
```

E então reinicia o Serviço de Identidade:

```
# service keystone restart
```

Por default, o keystone guarda os tokens expirados indefinidamente no banco de dados. Este acumulo pode aumentar consideravelmente o espaço de armazenamento do BD.
É recomendado configurar uma tarefa periódica usando cron para limpar tokens expirados a cada hora.

```
# (crontab -l 2>&1 | grep -q token_flush) || \
echo '@hourly /usr/bin/keystone-manage token_flush
>/var/log/keystone/keystone-tokenflush.log 2>&1' >>
/var/spool/cron/crontabs/root
```

OBS: Execute como *root* (isto é, use **sudo su** antes, no caso do Ubuntu), antes de executar este comando.

1.2- Definindo usuários, alocadores e funções

Antes de criar usuários que possam se autenticar no Serviço de Identidade, é necessário especificar onde este se encontra, mediante o uso do ADMIN_TOKEN criado anteriormente. Assim:

```
# export OS_SERVICE_TOKEN=ADMIN_TOKEN
# export OS_SERVICE_ENDPOINT=http://controller:35357/v2.0
```

Criar um usuário administrativo:

```
$ keystone user-create --name=admin --pass=ADMIN_PASS --email=ADMIN_EMAIL
```

Criando um papel admin

```
$ keystone role-create --name=admin
```

Criando um admin tenant

```
$ keystone tenant-create --name=admin --description="Admin Tenant"
```

É necessario agora ligar admin user, admin role e admin tenant juntos atraves do comando:

```
$ keystone user-role-add --user=admin --tenant=admin --role=admin
```

Ligar admin user, _member_role e admin tenant:

```
$ keystone user-role-add --user=admin --role=_member_ --tenant=admin
```

Criar um usuário comum

Esta conta irá utilizar tarefas não administrativas na nuvem. Este processo pode ser repetido para criar novos usuários não administrativos:

```
$ keystone user-create --name=demo --pass=DEMO_PASS --email=DEMO_EMAIL
```

Criar um demo tenant

```
$ keystone tenant-create --name=demo --description="Demo Tenant"
```

Ligar demo user, _member_role e demo tenant

```
$ keystone user-role-add --user=demo --role=_member_ --tenant=demo
```

Criando um service tenant

```
$ keystone tenant-create --name=service --description="Service Tenant"
```

1.3- Definindo os *endpoints* dos serviços e das APIs

Para que o Serviço de Identidade acompanhe os outros serviços do OpenStack, é necessário fazer a associação, registrando cada serviço no OpenStack.
Primeiro, criar uma entrada de serviço para o Serviço de Identidade:
$ keystone service-create --name=keystone --type=identity --description="Keystone Identity Service"

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | Keystone Identity Service        |
| id          | 15c11a23667e427e91bc31335b45f4bd |
| name        | keystone                         |
| type        | identity                         |
+-------------+----------------------------------+
```

O valor do campo id é gerado aleatoriamente . Esse id será usado no próximo comando:

```
$ keystone endpoint-create \
  --service-id=$(keystone service-list | awk '/ identity / {print
$2}') \
  --publicurl=http://controller:5000/v2.0 \
  --internalurl=http://controller:5000/v2.0 \
  --adminurl=http://controller:35357/v2.0
```

```
+-------------+----------------------------------------------+
|   Property  |            Value                             |
+-------------+----------------------------------------------+
| adminurl    | http://controller:35357/v2.0                 |
| id          | 11f9c625a3b94a3f8e66bf4e5de2679f             |
| internalurl | http://controller:5000/v2.0                  |
| publicurl   | http://controller:5000/v2.0                  |
| region      | regionOne                                    |
| service_id  | 15c11a23667e427e91bc31335b45f4bd             |
+-------------+----------------------------------------------+
```

Esses comandos vão ser usados quando novos serviços do OpenStack forem adicionados, para poder registrá-los.

1.4- Verificando a instalação do serviço de identidade (DONE!)

Para verificar se o Serviço de Identidade está funcionando corretamente, primeiro, deve ser tirada a atribuição das variáveis de ambiente OS_SERVICE_TOKEN e OS_SERVICE_ENDPOINT, que na verdade foram atribuídas unicamente para criar o usuário administrativo e registrar o Serviço de Identidade.

```
$ unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT
```

Agora, a autenticação normal baseada em nome de usuário pode ser realizada. Peça uma ficha de autenticação com o usuário administrativo e a senha dele que foi criada anteriormente:

```
$ keystone --os-username=admin --os-password=ADMIN_PASS \
  --os-auth-url=http://controller:35357/v2.0 token-get
```

Uma ficha deverá ser passada em resposta, juntamente com o Identificador (ID) do usuário. Isso verifica que o *keystone* está rodando no *endpoint* esperado, e que a conta de usuário está estabelecida com as devidas credenciais.
Agora, verifique que a autorização está se comportando como deveria ao pedir autorização em um alocador:

```
$ keystone --os-username=admin --os-password=ADMIN_PASS \
  --os-tenant-name=admin --os-auth-url=http://controller:35357/v2.0
token-get
```

Assim, você deve receber uma nova ficha, mas desta vez ela incluirá o identificador do alocador requisitado. Isso comprova que sua conta de usuário tem uma função explicitamente definida no alocador especificado, e que este existe, conforme deveria ser.

Opcionalmente, você pode deixar os argumentos --os-* definidos já direto nas variáveis de ambiente para simplificar o uso de alguns comandos. Crie um arquivo openrc.sh com as credenciais de usuário e o *endpoint* do administrador, com:
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://controller:35357/v2.0

Então, você pode dar um *source* no arquivo para que ele seja lido pelas variáveis de ambiente:
```
$ source openrc.sh
```

Verifique que seu arquivo openrc.sh está configurado corretamente com o mesmo comando acima, mas sem a parte do --os-* :
```
$ keystone token-get
```

O comando retorna uma ficha e um identificador do alocador especificado. Isso confirma que as variáveis de ambiente foram configuradas corretamente.
Agora, confirme que sua conta de administrador possui autorização para usar comandos de administrador:
```
$ keystone user-list
```

```
+----------------------------------------------------+----------+-----------------------------+----------+
|                 id                                 | enabled | email                        |  name  |
+----------------------------------------------------+----------+-----------------------------+----------+
| a4c2d43f80a549a19864c89d759bb3fe | True      | admin@example.com  | admin  |
```

Isso confirma que sua conta de administrador possui função de administrador, o que bate com a configuração que foi definida no arquivo. json, na política usada no Serviço de Identidade.

2- Instalar e Configurar os clientes OpenStack

A ideia é passar comandos e variáveis de ambiente que os usuários da nuvem terão acesso. Deste modo, os passos desta seção são executados em uma máquina que supostamente serviria de cliente, para que haja uma experiência de usuário similar a outros indivíduos que eventualmente irão fazer uso.

2.1- Visão Geral

Você pode usar clientes por linha de comando do Openstack que fazem chamadas de API. Você pode rodar estes comandos por linha de comando ou por *scripts* para automatizar tarefas. Se você provar credenciais do OpenStack, você pode rodar estes comandos de qualquer máquina. Internamente, cada comando de cliente roda comandos **cURL** que incorporam as

chamadas de API. As APIs do OpenStack são APIs *RESTful* que fazem uso do protocolo HTTP, incluindo métodos, URIs, tipos de mídia, e códigos de resposta.
Estes clientes de código aberto em Python rodam em sistemas Linux ou Mac OS X e são fáceis de aprender e usar. Cada serviço do OpenStack possui seu próprio cliente de linha de comando. Em alguns comandos de cliente, você pode especificar um parâmetro de **debug** para mostrar a requisição API por baixo do comando. Esta é uma boa maneira de se familiarizar com as chamadas de API do OpenStack. A tabela a seguir lista os clientes por linha de comando para cada serviço do OpenStack e seu nome de pacote e descrição.

Serviço	Cliente	Pacote	Descrição
Armazenamento em Bloco	**cinder**	python-cinderclient	Cria e gerencia volumes
Compute	**nova**	python-novaclient	Cria e gerencia imagens, instâncias, e *flavors*.
Serviço de Banco de Dados	**trove**	python-troveclient	Cria e gerencia Banco de Dados.
Identidade	**keystone**	python-keystoneclient	Cria e gerencia usuários, alocadores, funções, *endpoints*, e credenciais.
Serviço de Imagem	**glance**	python-glanceclient	Cria e gerencia imagens.
Rede	**neutron**	python-neutronclient	Configura redes para servidores visitantes. Este cliente era chamado antes de **quantum**
Armazenamento de Objetos	**swift**	python-swiftclient	Coleta estatísticas, lista itens, atualiza metadados, e faz *upload*, *download*, e remoção de arquivos armazenados pelo Serviço de Armazenamento de Objetos. Ganha acesso a uma instalação de Armazenamento de Objetos para processamento *ad hoc*.
Orquestramento	**heat**	python-heatclient	Inicia pilhas de templates, visualiza detalhes de pilhas em execução incluindo eventos e

			recursos, e faz *update* e remoção de pilhas.
Telemetria	**ceilometer**	python-ceilometerclient	Cria e coleta medições através do OpenStack

2.2- Instalar os clientes por linha de comando do OpenStack

2.2.1- Instalando *software* de pré-requisito

A tabela a seguir mostra os *softwares* que serão necessários para rodar os clientes por linha de comando, e fornece instruções de instalações para estes, conforme necessário.

OBS: Existem instruções para instalação em Mac OS X e Windows, além de Linux, mas neste manual se assume que o usuário está usando uma distribuição Linux.

Pré-requisito	Descrição
Python 2.6 em diante	Atualmente, os clientes não suportam o Python 3.
pacote **setuptools**	Existem pacotes nas distribuições Linux que fazem o setuptools ser fácil de instalar. Se no seu gerenciador de pacotes não existir algo pelo nome *setuptools*, baixe o pacote diretamente de http://pypi.python.org/pypi/setuptools.
pacote **pip**	No Ubuntu 12.04/14.04, existe uma versão por pacote que permite usar dpkg ou aptitude para instalar o python-novaclient: # aptitude install python-novaclient OU # apt-get install python-novaclient Então, para o Ubuntu ou Debian, instale o pacote pip com este comando: # aptitude install python-pip

2.2.2- Instalando os clientes

Os clientes foram elaborados para serem rodados a partir de outras máquinas que não sejam os servidores (isto é, se tratando dos usuários da nuvem, e não os administradores).
Nas instruções a seguir nesta seção, substitua PROJECT pelo nome do cliente que deseja instalar, como **nova**, por exemplo. Repita para cada cliente. Os seguintes valores são válidos:

- ceilometer - API de Telemetria
- cinder - API do Armazenamento de Bloco e extensões

- glance - API do Serviço de Imagem
- heat - API de Orquestramento
- keystone - API do Serviço de Identidade e extensões
- neutron - API de Rede
- nova - API do *Compute* e extensões
- swift - API do Armazenamento de Objetos
- trove - API do Serviço de Banco de Dados

Este é um exemplo de comando para instalar o cliente **nova** com o *pip*:
```
# pip install python-novaclient
```

OBS: Atentar para a instalação do pacote do **python-pip**.

Instalando com o pip
Use o pip para instalar os clientes OpenStack em um sistema Linux, Mac OS X, ou Windows. É fácil de usar e garante que você irá possuir a versão mais recente do cliente pelo Python Package Index. Além disso, o pip permite que você atualize ou remova um pacote.
Instale cada cliente separadamente usando o comando a seguir:
```
# pip install python-PROJECTclient
```

Atualizando ou removendo pacotes
Para atualizar um cliente, adicione a opção --upgrade ao comando **pip install**:
```
# pip install --upgrade python-PROJECTclient
```

E para remover um cliente, use o comando **pip uninstall**:
```
# pip uninstall python-PROJECTclient
```

2.3- Definindo variáveis de ambiente usando o arquivo de RC do OpenStack
Para definir as variáveis de ambiente necessárias para os clientes do OpenStack por linha de comando, você precisa criar um arquivo de ambiente chamado de arquivo de RC do OpenStack, ou arquivo openrc.sh. Este arquivo de ambiente específico por projeto contém as credenciais que todos os serviços do OpenStack usam.
Quando você realiza um *source* no arquivo, as variáveis de ambiente são definidas para seu *shell* atual. As variáveis permitem os comandos de cliente do OpenStack se comunicarem com os serviços do OpenStack que rodam na nuvem.

2.3.1- Criar e realizar source em um arquivo de RC do OpenStack
1- Em um editor de texto, crie um arquivo chamado PROJECT-openrc.sh e adiciona a seguinte informação de autenticação:
O exemplo a seguir mostra a informação para um projeto chamado **admin**, onde o usuário do SO também se chama **admin**, e o serviço de identidade está localizado no **controller**.
export OS_USERNAME=admin

export OS_PASSWORD=ADMIN_PASS
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://controller:35357/v2.0

2- Em qualquer *shell* que você deseje rodar os comandos do OpenStack, realize um *source*, do arquivo PROJECT-openrc.sh do respectivo projeto. Neste exemplo, você está realizando *source* do arquivo admin-openrc.sh para o projeto *admin*:

```
$ source admin-openrc.sh
```

2.3.2- Sobre-escrever valores de variáveis de ambiente

Quando você roda os comandos de cliente do OpenStack, você sobre-escreve algumas configurações de variáveis de ambiente usando as opções que são listadas no fim da saída de **ajuda** para os vários comandos de cliente. Por exemplo, você pode sobre-escrever a configuração OS_PASSWORD no arquivo PROJECT-openrc.sh ao especificar uma senha em um comando do **keystone**, como em:

```
$ keystone --os-password PASSWORD service-list
```

sendo PASSWORD a sua senha.

2.4- Criar arquivos openrc.sh

Conforme explicado na seção **Criar e realizar source em um arquivo de RC do OpenStack**, use as credenciais da seção chamada **Definindo usuários, alocadores e funções** e crie os seguintes arquivos PROJECT-openrc.sh:

- admin-openrc.sh para um usuário administrativo
- demo-openrc.sh para um usuário normal

export OS_USERNAME=demo
export OS_PASSWORD=DEMO_PASS
export OS_TENANT_NAME=demo
export OS_AUTH_URL=http://controller:35357/v2.0

3- Configurar o Serviço de Imagem - GLANCE

O Serviço de Imagem do OpenStack permite os usuários descobriem, registrarem e recuperarem imagens de máquinas virtuais. Também conhecido como o projeto *glance*, o Serviço de Imagem oferece uma API em REST que permite requisitar metadados de imagens de máquinas virtuais e recuperar uma imagem real.
Imagens de máquinas virtuais disponibilizadas pelo Serviço de Imagem podem ser armazenadas em uma variedade de maneiras, o que inclui tipos de arquivos simples, ou até sistemas de armazenamento de objetos, como o Armazenamento de Objetos do OpenStack.

3.1- Visão Geral do Serviço de Imagem

Este contém os seguintes componentes:

- glance-api - Aceita chamadas de API para Imagens, para obter descoberta, recuperação e armazenamento de imagens;
- glance-registry - Armazena, processa, e recupera metadados sobre imagens. Os metadados incluem detalhes como tamanho, tipo, entre outros;
- Banco de Dados - Armazena metadados de imagens. Você pode escolher o banco de dados de acordo com sua preferência. A maioria das instalações usa MySQL ou SQLite;
- Repositório de armazenamento para arquivos de imagens. De acordo com esta figura, o Serviço de Armazenamento de Objetos é o que está sendo utilizado, mas você pode configurar um repositório diferente. O Serviço de Imagem suporta sistemas normais de arquivos, dispositivos de bloco RADOS, *Amazon S3*, e HTTP. Algumas escolhas permitem apenas leitura.

3.2- Instalando o Serviço de Imagem

O Serviço de Imagem age como um registro de imagens de máquinas virtuais. Os usuários podem adicionar novas imagens, ou pedir um *snapshot*, que armazena o estado de uma imagem.

1- Instale o pacote com o Serviço de Imagem no **controller**:

```
# apt-get install glance python-glanceclient
```

2- O Serviço de Imagem armazena informações sobre imagens em um Banco de Dados. Baseando-se que esse guia usa o MySQL, a configuração que irá localizar o Banco de Dados ficará assim nos arquivos de configuração /etc/glance/glance-api.conf (**glance-api**) e /etc/glance/glance-registry.conf (**glance-registry**):

```
....
[database]
connection = mysql://glance:GLANCE_DBPASS@controller/glance
```

GLANCE_DBPASS será a senha usada pelo Banco de Dados no Serviço de Imagem.

3- Configurar o Serviço de Imagem para usar o *broker* de mensagens: Edite o arquivo /etc/glance/glance-api.conf na seção [DEFAULT]. Substitua RABBIT_PASS pela senha para a conta *guest* do RabbitMQ.

```
[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

4- O Ubuntu por padrão cria um Banco de Dados SQLite. É melhor remover (ou renomear) o arquivo glance.sqlite em /var/lib/glance/ para ele não ser acessado por engano.

5- Acesse o Banco de Dados, e use a senha que foi criada para adicionar um usuário *glance* do banco de dados:

```
# mysql -u root -p
mysql> CREATE DATABASE glance;
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
IDENTIFIED BY 'GLANCE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
IDENTIFIED BY 'GLANCE_DBPASS';
```

6- Crie as tabelas do banco de dados para o Serviço de Imagem:

```
# su -s /bin/sh -c "glance-manage db_sync" glance
```

OBS [BUGFIX]: Caso ocorra uma mensagem de erro do tipo **CRITICAL glance [-] ValueError: Tables "migrate_version" have non utf8 collation, please make sure all tables are CHARSET=utf8**, adicione as seguintes linhas no arquivo /etc/my.cnf , na seção [mysqld] :

```
collation-server = utf8_general_ci
init-connect='SET NAMES utf8'
character-
set-server = utf8
```

- Em seguida, alterar a tabela **migrate_version**, com:

```
ALTER TABLE migrate_version CONVERT TO CHARACTER SET 'utf8';
```

E então, tente novamente o comando **su -s /bin/sh -c "glance-manage db_sync" glance**. A senha que irá ser pedida deve ser a do Banco de Dados ou a do GLANCE_DB. Caso não seja possível conectar ao BD, reinicie o sistema antes de tentar o comando.

7- Crie um usuário *glance* para que o Serviço de Imagem possa autenticar com o Serviço de Identidade. Escolha uma senha, e especifique um endereço de e-mail para o usuário *glance*. Use o alocador de serviço, e atribua a ele a função de administrador.

```
# keystone user-create --name=glance --pass=GLANCE_PASS \
   --email=glance@example.com

# keystone user-role-add --user=glance --tenant=service --role=admin
```

8- Configure o Serviço de Imagem para usar o Serviço de Identidade na autenticação. Para isso, edite os arquivos /etc/glance/glance-api.conf e /etc/glance/glance-registry.conf , substituindo *GLANCE_PASS* pela senha que você escolheu para o usuário *glance* no Serviço de Identidade.

a. Adicione as seguintes chaves na seção [keystone_authtoken]:

```
[keystone_authtoken]
…
```

auth_uri = http://controller:5000
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = GLANCE_PASS

b. Adicione a seguinte chave na seção [paste_deploy]:
[paste_deploy]
...
flavor = keystone

9- Adicione as credenciais nos arquivos /etc/glance/glance-api-paste.ini e /etc/glance/glance-registry-paste.ini . Deixe assim as opções na seção [filter:authtoken], e o resto, deixe como está:
[filter:authtoken]
paste.filter_factory=keystoneclient.middleware.auth_token:filter_factory
auth_host=controller
admin_user=glance
admin_tenant_name=service
admin_password=GLANCE_PASS

10- Registre o Serviço de Imagem com o Serviço de Identidade para que os outros serviços OpenStack possam localizá-lo. Registre o serviço e crie o *endpoint*:

```
# keystone service-create --name=glance --type=image \
  --description="Glance Image Service"
```

11- Use o identificador fornecido pelo serviço para criar o *endpoint*:

```
# keystone endpoint-create \
  --service-id=the_service_id_above \
  --publicurl=http://controller:9292 \
  --internalurl=http://controller:9292 \
  --adminurl=http://controller:9292
```

12- Reinicie o serviço *glance* com as novas configurações:

```
# service glance-registry restart
# service glance-api restart
```

3.3- Verifique a instalação do Serviço de Imagem

Para testar o serviço é bom usar uma imagem pequena, e que é conhecida por funcionar no OpenStack. Um esquema é usar uma imagem de 64-bit do CirrOS QCW2.

1- Baixe a imagem em um diretório dedicado com wget ou curl:

```
$ mkdir images
$ cd images/
$ wget
http://cdn.download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86_64-disk.img
```

2- Faça *upload* da imagem para o Serviço de Imagem:

```
# glance image-create --name=imageLabel --disk-format=fileFormat \
  --container-format=containerFormat --is-public=accessValue < imageFile
```

OBS:

- *imageLabel* = É um nome arbitrário pelo qual o usuário vai chamar a imagem.
- *fileFormat* = Especifica o formato do arquivo de imagem, o que inclui qcow2, raw, vhd, vmdk, vdi, iso, aki, ari, e ami.
- *containerFormat* = Especifica o formato do contâiner, que de formatos válidos inclui bare, ovf, aki, ari e ami.
- *accessValue* = Especifica o acesso á imagem. Se for **true**, todos os usuários podem ver e acessar a imagem, e se for **false**, somente administradores podem ver e acessar a imagem.
- *imageFile* = Especifica o nome do arquivo de imagem baixado.

Exemplo:

```
$ source admin-openrc.sh
$ glance image-create --name "cirros-0.3.2-x86_64" --disk-format qcow2 \
  --container-format bare --is-public True --progress < cirros-0.3.2-x86_64-disk.img
```

A resposta é algo do tipo:

```
+------------------+--------------------------------------+

| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 64d7c1cd2b6f60c92c14662941cb7913     |
| container_format | bare                                 |
| created_at       | 2014-04-08T18:59:18                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | acafc7c0-40aa-4026-9673-b879898e1fc2 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | cirros-0.3.2-x86_64                  |
| owner            | efa984b0a914450e9a47788ad330699d     |
| protected        | False                                |
| size             | 13167616                             |
| status           | active                               |
| updated_at       | 2014-01-08T18:59:18                  |
+------------------+--------------------------------------+
```

OBS: O id obtido será aleatório, então certamente não será igual ao mostrado na figura.

Exemplo:

```
$ glance image-create --name="CirrOS 0.3.1" --disk-format=qcow2
--container-format=bare --is-public=true < cirros-0.3.1-x86_64-disk.img
```

saída:

```
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | d972013792949d0d3ba628fbe8685bce     |
| container_format | bare                                 |
| created_at       | 2014-03-26T20:43:09                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | a54ff28e-d178-483c-9765-6f2e876bd22c |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | CirrOS 0.3.1                         |
| owner            | 16875daf2d5b4bc5ab1628d97a2a4c38     |
```

```
| protected        | False                                |
| size             | 13147648                             |
| status           | active                               |
| updated_at       | 2014-03-26T20:43:09                  |
+------------------+--------------------------------------+
```

3- Confirme que o *upload* da imagem foi feito com sucesso, e veja seus atributos:

```
# glance image-list
```

```
+--------------------------------------+------------------+-------------+------------------+----------+--------+
| ID                                   | Name             | Disk Format | Container Format | Size     | Status |
+--------------------------------------+------------------+-------------+------------------+----------+--------+
| acafc7c0-40aa-4026-9673-b879898e1fc2 | CirrOS 0.3.1     | qcow2       | bare             | 13147648 | active |
+--------------------------------------+------------------+-------------+------------------+----------+--------+
```

OBS: Se quando tentar adicionar a imagem receber um erro do tipo HTTPInternalServerError (HTTP 500), tente definir 'db_auto_create = True' in glance-registry.conf para resolver, senão ele não vai criar a imagem no BD.

Também é possível fazer *upload* da imagem para o Glance sem necessariamente usar um arquivo local, com o parâmetro --copy-from, como em:

```
$ glance image-create --name="cirros-0.3.2-x86_64" --disk-format=qcow2 \
  --container-format=bare --is-public=true \
  --copy-from http://cdn.download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86_64-disk.img
```

4- Serviço de *Compute* - NOVA

O Serviço de *Compute* é o controlador do funcionamento da nuvem, o qual é a parte principal em um sistema IaaS. É usado então para hospedar e gerenciar sistemas de computação em nuvem, e seus módulos são implementados em Python.
Sua interação com os outros componentes se dá tal como: com o ***Keystone*** para autenticação, ***Glance*** para as imagens, e o ***Horizon*** para a interface administrativa e de usuário. O acesso às imagens é limitado pelo projeto e pelo usuário; as cotas, são limitadas pelo projeto (por exemplo, pelo número de instâncias). O serviço de *Compute* escala horizontalmente com *hardware* convencional, e baixa imagens para lançar instâncias conforme exigido.

Suas áreas, com seus respectivos componentes, seguem abaixo:
API

- nova-api service - Aceita e responde chamadas de usuários finais para a *compute* API. Suporta o **OpenStack Compute API**, **Amazon EC2 API**, e uma API especial para usuários privilegiados para realizar ações administrativas. Também realiza a maioria das ações orquestrais, como iniciar uma instância, e monitora as políticas.
- nova-api-metadata service - Aceita requisições de metadados das instâncias. Este serviço é geralmente utilizado apenas em um modo multi-host com múltiplas instalações do nova-network.

Compute Core

- nova-compute process. Um monitor que inicia e termina instâncias de máquinas virtuais por meio de chamadas nas APIs de hipervisores, como XenAPI (Xen), libvirt (KVM ou QEMU), etc.
- nova-scheduler process. É o escalonador, que pega uma requisição da fila, e determina em que host a instância deve rodar.
- nova-conductor module. Intermedia o acesso entre a BD da nuvem e o nova-compute.

Rede para as VMs

- nova-network daemon. Um monitor que opera similarmente ao nova-compute, mas que nesse caso, realiza operações nas regras do iptables para adequar as requisições de uma instância.
- nova-dhcpbridge script. Rastreia *leases* de IPs usados por VMs e guarda no BD ao usar o dnsmasq dhcp-script.

Interface de Console

- nova-consoleauth daemon.
- nova-novncproxy daemon. Faz um esquema de proxy para acessar as instâncias que estão rodando por meio de VNC.
- nova-console daemon. Depreciado, e foi substituído pelo nova-novncproxy.
- nova-cert daemon. Gerencia certificados x509.

Gerenciamento de imagem (Caso do EC2)

- nova-objectstore daemon. Fornece uma interface S3 para registrar imagens com o Serviço de Imagem. Usado principalmente para instalações que suportam o **euca2ools**. As ferramentas **euca2ools** se comunicam com o nova-objectstore em uma linguagem do S3, e o nova-objectstore traduz as requisições S3 para requisições do Serviço de Imagem.
- euca2ools client. Um conjunto de comandos intepretados por linha de comando para gerenciar os recursos da nuvem. Apesar de não ser um módulo do OpenStack, você pode configurar o **nova-api** para suportar esta interface EC2. Para mais informações veja Eucalyptus 3.4 Documentation.

Clientes por linha de Comando e outras interfaces

- nova client. Permite que os usuários usem comandos como um administrador de alocador ou usuário final.
- nova-manage client. Permite que os administradores da nuvem entrem com comandos.

Outros componentes

The queue (fila). Um concentrador que repassa as mensagens entre os monitores (daemon). É geralmente implementado em **RabbitMQ**, mas funcionaria com outros *queues* AMQP.

Banco de Dados SQL. Suporta os múltiplos serviços que precisam de um banco de dados, como aqueles que usam metadados, dados de instâncias em uso, redes disponíveis e projetos. Teoricamente, o OpenStack suporta qualquer banco de dados que o **SQL-Alchemy** suporte, mas os mais comumente usados são o sqlite3, MySQL, e PostgreSQL.

4.1- Instalando serviços *Compute* do *controller*

O *Compute*, sendo um dos serviços principais da nuvem, vai fazer uma quantidade considerável de tarefas. Considerando que este está instalado no **controller**, esta é a sequência para instalar e configurar os serviços:

1- Instale os pacotes do *Compute*:

```
# apt-get install nova-api nova-cert nova-conductor nova-consoleauth
nova-novncproxy nova-scheduler python-novaclient
```

2- Como o Compute guarda informação em um Banco de Dados, e considerando que o MySQL é o BD utilizado, é preciso configurar a localização deste:
Substitua a senha NOVA_DBPASS pela senha escolhida para o acesso ao BD. Edite o arquivo /etc/nova/nova.conf na seção [database], adicionando caso necessário:

```
[database]
connection = mysql://nova:NOVA_DBPASS@controller/nova
```

3- Configure o Serviço *Compute* para usar o servidor de mensagens **RabbitMQ** no arquivo /etc/nova/nova.conf , na seção [DEFAULT], ao adicionar estas chaves de configuração:

```
[DEFAULT]
rpc_backend = nova.rpc.impl_kombu
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

4- Configure os endereços de **my_ip**, **vncserver_listen**, e **vncserver_proxyclient_address** para o endereço da rede interna do **controller**, no arquivo de configuração /etc/nova/nova.conf , na seção [DEFAULT]

```
[DEFAULT]
my_ip=10.0.0.11
vncserver_listen=10.0.0.11
vncserver_proxyclient_address=10.0.0.11

glance_host=controller
```

OBS: As linhas na verdade parece que não existem, e vão ser adicionadas (criadas) no arquivo.
OBS2: A linha glance_host é importante! Atraves dela o nova se comunica com o glance.

5- O Ubuntu por padrão cria um Banco de Dados SQLite. É melhor remover (ou renomear) o arquivo nova.sqlite em /var/lib/nova/ para ele não ser acessado por engano.

```
# rm /var/lib/nova/nova.sqlite
```

6- Use a senha do BD que foi criada anteriormente para logar como *root* , e crie um novo usuário *nova*:

```
$ mysql -u root -p
mysql> CREATE DATABASE nova;
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
IDENTIFIED BY 'NOVA_DBPASS';
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
IDENTIFIED BY 'NOVA_DBPASS';
```

7- Crie as tabelas do Serviço de *Compute*:

```
# su -s /bin/sh -c "nova-manage db sync" nova
```

8- Crie um novo usuário *nova* para que o Serviço de *Compute* autentique no Serviço de Identidade, usando o alocador *service*, e dando ao usuário a função de *admin*:

```
$ keystone user-create --name=nova --pass=NOVA_PASS
--email=nova@example.com
```

saída:

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   | guilherme.pimentel@ccc.ufcg.edu.br |
| enabled  |              True                |
|    id    |  13ee333ee24e481fa1be677e08c4a7ef  |
|   name   |              nova                |
| username |              nova                |
+----------+----------------------------------+
```

```
$ keystone user-role-add --user=nova --tenant=service --role=admin
```

9- Configure o *Compute* para usar essas credenciais com o Serviço de Identidade rodando no **controller**. Substitua NOVA_PASS pela senha criada para o *Compute*. Edite o arquivo de configuração /etc/nova/nova.conf na seção [DEFAULT] para adicionar esta chave:

```
[DEFAULT]
...
auth_strategy=keystone
```

Adicione estas chaves à seção [keystone_authtoken]:

```
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = NOVA_PASS
```

10- Você precisa registrar o Serviço de *Compute* com o Serviço de Identidade para que os outros serviços do OpenStack possam localizá-lo. Registre o serviço e especifique o *endpoint*:

```
$ keystone service-create --name=nova --type=compute \
  --description="OpenStack Compute"
```

saída:

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |          OpenStack Compute       |
|   enabled   |              True                |
|      id     | 29431518a9c1421291d39ff7f01676e9 |
|      name   |              nova                |
|      type   |           compute                |
+-------------+----------------------------------+
```

```
$ keystone endpoint-create \
  --service-id=$(keystone service-list | awk '/ compute / {print $2}') \
  --publicurl=http://controller:8774/v2/%\(tenant_id\)s \
  --internalurl=http://controller:8774/v2/%\(tenant_id\)s \
  --adminurl=http://controller:8774/v2/%\(tenant_id\)s
```

saída:

```
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
|   adminurl  | http://150.165.85.226:8774/v2/%(tenant_id)s |
|      id     |          ca3b774988aa49aaae0d350d88687216       |
| internalurl | http://150.165.85.226:8774/v2/%(tenant_id)s |
|  publicurl  | http://150.165.85.226:8774/v2/%(tenant_id)s |
```

```
|         region   |                  regionOne                  |
| service_id |           29431518a9c1421291d39ff7f01676e9         |
+------------+---------------------------------------------+
```

11- Reinicie os serviços de *Compute*:

```
# service nova-api restart
# service nova-cert restart
# service nova-consoleauth restart
# service nova-scheduler restart
# service nova-conductor restart
# service nova-novncproxy restart
```

12- Para verificar a configuração, mande listar as imagens disponíveis:

$ nova image-list

```
+--------------------------------------+-----------------+--------+--------+
| ID                                   | Name            | Status | Server |
+--------------------------------------+-----------------+--------+--------+
| acafc7c0-40aa-4026-9673-b879898e1fc2 | CirrOS 0.3.1    | ACTIVE |        |
|                                      |                 |        |        |
```

4.2- Configure um *Compute Node*

Depois de configurar o Serviço de *Compute* no controller, agora podem ser adicionados **Compute Nodes**. Esta disposição na verdade pode ser feita em apenas uma máquina, mas este guia adota o padrão de separar os **Compute Nodes** em outras máquinas, para auxiliar na escalabilidade horizontal.

Como o Serviço de *Compute* depende de um hipervisor para rodar as instâncias das máquinas virtuais, e o OpenStack pode adotar vários hipervisores, neste guia o KVM foi o escolhido.

1- No nó adicional, precisa obviamente configurar o sistema também, o que está descrito no começo do guia, na parte de **Rede**.

- Use endereços IP diferentes quando configurar eth0. Este guia assume 10.0.0.31 como o endereço de rede de gerenciamento no primeiro nó de ***Compute***.
- Se você está usando a Rede do OpenStack (**neutron**), configure eth1 como a interface dos túneis das instâncias, com endereço IP 10.0.1.31 para o primeiro nó de ***Compute***. Para detalhes, veja as instruções na seção chamada **Nó de Compute;**
- **Se você está uasndo Rede legada (nova-network), não configure um endereço IP estático para eth1. O componente de Rede do OpenStack atribui e configura um endereço IP. Para detalhes, veja as instruções na seção chamada Nó de Compute;**
- **Defina o nome de *host* para compute1**. Para verificar, use o parâmetro **uname -n**. Certifique-se que os endereços IP e nomes de *host* de ambos os nós estão listados no arquivo **/etc/hosts** de cada sistema.

- Sincronize com o nó **controller**. Siga as instruções da seção **Network Time Protocol (NTP)**.
- Instale as bibliotecas de cliente MySQL. Você não precisa instalar o servidor de Banco de Dados MySQL ou iniciar o serviço de MySQL.
- Ative os pacotes OpenStack para a distribuição que você está usando. Veja a seção **Pacotes e repositórios para o OpenStack**.

OBS: Não esqueça de reiniciar para poder realizar todas as mudanças após estas configurações.

2- Quando tudo estiver pronto, instale os pacotes necessários para o Serviço de *Compute*:
```
# apt-get install nova-compute-kvm python-guestfs
```

OBS: Quando/se perguntar algo sobre criar uma *supermin appliance*, responda com **yes**.

3- Por questões de segurança, o Ubuntu impede usuários não-root de ler o kernel, o que restringe serviços de hipervisor, como o qemu e libguestfs. Para tornar o kernel atual legível, use:
```
# dpkg-statoverride  --update --add root root 0644 /boot/vmlinuz-$(uname
-r)
```

Também é possível aplicar essa solução para atualizações futuras do *kernel*, criando o seguinte script: /etc/kernel/postinst.d/statoverride
```
#!/bin/sh
version="$1"
# passing the kernel version is required
[ -z "${version}" ] && exit 0
dpkg-statoverride --update --add root root 0644 /boot/vmlinuz-${version}
```

Torne o script executável, com:
```
# chmod +x /etc/kernel/postinst.d/statoverride
```

4- Edite o arquivo de configuração /etc/nova/nova.conf e adicione as seguintes linhas às seções em questão:
```
[DEFAULT]
...
auth_strategy=keystone
...
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://nova:NOVA_DBPASS@controller/nova
```

[keystone_authtoken]
auth_uri = http://controller:5000
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = NOVA_PASS

5- Configure o Serviço de *Compute* para usar o servidor de mensagens **RabbitMQ** no arquivo /etc/nova/nova.conf , na seção [DEFAULT], ao adicionar estas chaves de configuração:

[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS

6- Configure o *Compute* para permitir acesso remoto às instâncias. Edite o arquivo /etc/nova/nova.conf , na seção [DEFAULT], e adicione as seguintes chaves:

[DEFAULT]
...
my_ip = 10.0.0.31
vnc_enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = 10.0.0.31
novncproxy_base_url = http://controller:6080/vnc_auto.html

7- Especifique o host que está rodando o Serviço de Imagem. Edite o o arquivo /etc/nova/nova.conf , e adicione estas linhas à seção [DEFAULT]:
[DEFAULT]
...
glance_host=controller

8- (VEJA O CASO) Se você instalar o *Compute* em uma máquina virtual para fins de testes, você deve determinar se seu hipervisor e/ou CPU suportam aceleração de *hardware* aninhada (***nested hardware acceleration***) usando o comando a seguir:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

Se este comando retornar um valor de **um ou maior**, seu hipervisor e/ou CPU suportam aceleração de hardware aninhada e não precisam de configuração adicional.

Se o comando retornar um valor de 0, seu hipervisor e/ou CPU não suportam aceleração de *hardware* aninhada e o libvirt precisa usar o QEMU no lugar do KVM. Edite o arquivo /etc/nova/nova-compute.conf na seção [libvirt] para modificar esta chave:

```
[libvirt]
...
virt_type = qemu
```

9- Remova (ou renomeie) o Banco de Dados SQLite criado pelos pacotes:

```
# rm /var/lib/nova/nova.sqlite
```

10- Reinicie o Serviço de *Compute*:

```
# service nova-compute restart
```

5- Adicionando um Serviço de Rede

A Rede do OpenStack pode ser configurada de duas maneiras. O Serviço de Rede nativo (nova-network) é mais simples e limitado, enquanto que o serviço de Rede dedicado (neutron) é mais completo, e apresenta melhor compatibilidade à tecnologias de rede para as instâncias. Este guia aborda como configurar das duas maneiras.

5.1- Serviço dedicado de Rede do OpenStack (Neutron)

5.1.1- Conceitos de Rede

O serviço dedicado de Rede do OpenStack (Neutron) gerencia todas as facetas de rede para a infraestrutura de rede virtual (VNI) e aspectos de camadas de acesso da infraestrutura de rede física (PNI) no seu ambiente OpenStack. Ele permite os alocadores criarem topologias avançadas de rede, incluindo serviços como firewall, balanceadores de carga, e redes privadas virtuais (VPNs).

A Rede fornece as seguintes abstrações de objetos: redes, subredes e roteadores. Cada um possui a funcionalidade que imita sua versão física: redes contém subredes, e roteadores direcionar tráfego entre diferentes subredes e redes.

Qualquer configuração do **Neutron** possui pelo menos uma rede externa. Esta rede, diferente de outras redes, não é meramente uma rede definida virtualmente, e representa uma visualização de uma fatia da rede externa que é acessível por fora da instalação do OpenStack. Os endereços IP da rede externa do Neutron são acessíveis por qualquer um fisicamente na rede externa. Como esta rede meramente representa uma fatia da rede externa, o DHCP é desativado nesta rede.

Em adição às redes externas, qualquer configuração do Neutron possui uma ou mais redes internas. Estas redes definidas por software conectam diretamente com as MVs. Apenas as MVs de uma certa rede interna, ou aquelas em subredes conectadas através de interfaces para um roteador similar, poderão acessar as MVs conectadas a esta rede diretamente.

Para que a rede externa acesse as MVs e vice-versa, são necessários roteadores entre as redes. Cada roteador possui um **gateway** que está conectado à rede e múltiplas interfaces que

estão conectadas à subredes. Como um roteador físico, as subredes podem acessar máquinas em outras subredes que estão conectadas ao mesmo roteador, e máquinas podem acessar a rede externa pelo gateway do roteador.

Adicionalmente, você pode alocar endereços IP em redes externas para portas na rede interna. Quando algo está conectado a uma subrede, esta conexão é chamada de porta. Você pode associar endereços IP da rede externa com portas para MVs. Desta maneira, entidades de fora da rede podem acessar as MVs.

O Neutron também suporta grupos de segurança, que permitem que administradores definam regras de firewall em grupos. Uma MV pertence a um ou mais grupos de segurança, e o Neutron aplica as regras a estes grupos para bloquear/desbloquear portas, faixas de portas, ou tipos de tráfego para esta MV.

Cada **plugin** que o Neutron usa possui seus próprios conceitos. Ainda que não seja vital para operar o Neutron, entender estes conceitos pode ajudá-lo a configurar o Neutron. Todas as instalações do Neutron usam um plugin central e um plugin de segurança de grupo (ou apenas o plugin de segurança **No-OP**). Adicionalmente, plugins de Firewall-as-a-Service (FWaaS) e Load-balancing-as-a-service (LBaaS) estão disponíveis.

5.1.2- Plugin modular de camada 2 (ML2)

5.1.2.1- Configurando o nó controller

Pré-requisitos

Antes de configurar a Rede do OpenStack pelo **neutron**, você deve criar um Banco de Dados e credenciais do serviço de Identidade incluindo um usuário e serviço.

1- Conecte ao Banco de Dados como usuário *root*, crie o Banco de Dados *neutron*, e forneça o devido acesso a este:

Substitua NEUTRON_DBPASS pela senha desejada.

```
$ mysql -u root -p
mysql> CREATE DATABASE neutron;
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
IDENTIFIED BY 'NEUTRON_DBPASS';
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
IDENTIFIED BY 'NEUTRON_DBPASS';
```

2- Crie as credenciais do Serviço de Identidade para a Rede:

a) Crie o usuário *neutron*:

Substitua NEUTRON_PASS por uma senha desejada e neutron@example.com com um endereço de e-mail adequado.

```
$ keystone user-create --name neutron --pass NEUTRON_PASS --email
neutron@example.com
```

b) Associe o usuário *neutron* ao alocador *service* e função *admin*:

```
$ keystone user-role-add --user neutron --tenant service --role admin
```

c) Crie o serviço *neutron*:

```
$ keystone service-create --name neutron --type network --description
"OpenStack Networking"
```

d) Crie o *endpoint* do serviço:

```
$ keystone endpoint-create \
  --service-id $(keystone service-list | awk '/ network / {print $2}') \
  --publicurl http://controller:9696 \
  --adminurl http://controller:9696 \
  --internalurl http://controller:9696
```

Para instalar os componentes de Rede

```
# apt-get install neutron-server neutron-plugin-ml2
```

Para configurar o componente de servidores* de Rede

A configuração do componente de servidores de Rede inclui o Banco de Dados, mecanismo de autenticação, *broker* de mensagens, notificador de mudanças de topologia, e plugin.

1- Configure a Rede para usar o Banco de Dados:

a) Edite o arquivo /etc/neutron/neutron.conf e adicione a seguinte chave na seção [database]:

Substitua NEUTRON_DBPASS pela senha estabelecida anteriormente.

```
[database]
...
connection = mysql://neutron:NEUTRON_DBPASS@controller/neutron
```

2- Configure a Rede para usar o Serviço de Identidade para autenticação:

a) Edite o arquivo /etc/neutron/neutron.conf e adicione a seguinte chave à seção [DEFAULT]:

```
[DEFAULT]
...
auth_strategy = keystone
```

b) Adicione as seguintes chaves à seção [keystone_authtoken]:

Substitua NEUTRON_PASS pela senha estabelecida anteriormente para neutron usar com o Serviço de Identidade.

```
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_host = controller
auth_protocol = http
auth_port = 35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
```

3- Configure a Rede para usar o *broker* de mensagens:

a) Edite o arquivo /etc/neutron/neutron.conf e adicione as seguintes chaves à seção [DEFAULT]:

Substitua RABBIT_PASS pela senha que foi estabelecida para a conta guest no RabbitMQ.

```
[DEFAULT]
...
rpc_backend = neutron.openstack.common.rpc.impl_kombu
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

4- Configure a Rede para notificar o *Compute* sobre as mudanças de topologia de rede:

Substitua SERVICE_TENANT_ID pelo identificador do alocador *service* no Serviço de Identidade e NOVA_PASS pela senha que foi estabelecida para o usuário *nova* no Serviço de Identidade.

a) Edite o arquivo /etc/neutron/neutron.conf e adicione as seguintes chaves à seção [DEFAULT]:

```
[DEFAULT]
...
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://controller:8774/v2
nova_admin_username = nova
nova_admin_tenant_id = SERVICE_TENANT_ID
nova_admin_password = NOVA_PASS
nova_admin_auth_url = http://controller:35357/v2.0
```

OBS: Para obter o identificador do alocador *service:*

```
$ source admin-openrc.sh
$ keystone tenant-get service
```

5- Configure a Rede para usar o plugin modular de camada 2 (ML2) e os serviços associados:

a) Edite o arquivo /etc/neutron/neutron.conf e adicione as seguintes chaves à seção [DEFAULT]:

```
[DEFAULT]
...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
```

OBS: É recomendável adicionar **verbose = True** na seção [DEFAULT] do arquivo /etc/neutron/neutron.conf para auxiliar na correção de problemas.

Para configurar o plugin Modular de Camada 2 (ML2)
O plugin ML2 usa o mecanismo (agente) Open vSwitch (OVS) para construir um arcabouço de rede para as instâncias. Entretanto, o **controller** não precisa do agente ou serviço OVS porque ele não controla o tráfego de rede das instâncias.

- Edite o arquivo /etc/neutron/plugins/ml2/ml2_conf.ini :
  Adicione as seguintes chaves à seção [ml2]:
  [ml2]
  ...
  type_drivers = gre
  tenant_network_types = gre
  mechanism_drivers = openvswitch

  Adicione a seguinte chave à seção [ml2_type_gre]:
  [ml2_type_gre]
  ...
  tunnel_id_ranges = 1:1000

  Adicione a seção [securitygroup] e adicione as seguintes chaves a ela:
  [securitygroup]
  ...
  firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
  enable_security_group = True

Para configurar o Compute para usar a Rede
Por padrão, a maioria das distribuições configuram o Compute para usar a rede legada. Você deve reconfigurar o Compute para gerenciar redes pelo serviço dedicado de Rede.
Edite o arquivo /etc/nova/nova.conf e adicione as seguintes chaves à seção [DEFAULT]:
[DEFAULT]
...
network_api_class = nova.network.neutronv2.api.API
neutron_url = http://controller:9696
neutron_auth_strategy = keystone
neutron_admin_tenant_name = service
neutron_admin_username = neutron
neutron_admin_password = NEUTRON_PASS
neutron_admin_auth_url = http://controller:35357/v2.0
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
security_group_api = neutron

OBS: Por padrão, o Compute usa um serviço interno de firewall. Uma vez que a Rede inclui um serviço de firewall, você deve desabilitar o serviço de firewall do Compute ao usar o driver de firewall nova.virt.firewall.NoopFirewallDriver

Para finalizar a instalação

1- Reinicie os serviços de Compute

```
# service nova-api restart
# service nova-scheduler restart
# service nova-conductor restart
```

2- Reinicie o serviço de Rede

```
# service neutron-server restart
```

5.1.2.2- Configurando o nó de Rede

Pré-requisitos

Antes de configurar a Rede do OpenStack, é necessário ativar certas funções de rede do kernel.

1- Edite o arquivo /etc/sysctl.conf para conter o seguinte:

```
net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
```

2- Aplique as mudanças:

```
# sysctl -p
```

Para instalar os componentes de Rede

```
# apt-get install neutron-plugin-ml2 neutron-plugin-openvswitch-agent
openvswitch-datapath-dkms \
  neutron-l3-agent neutron-dhcp-agent
```

OBS: Instalações do Ubuntu usando a versão de kernel 3.11 ou mais recentes não precisam instalar o pacote *openvswitch-datapath-dkms*.

Para configurar os componentes comuns de Rede

A configuração de componentes comuns da Rede inclui o mecanismo de autenticação, o *broker* de mensagens, e plugin.

1- Configure a Rede para usar o Serviço de Identidade para autenticação:

a) Edite o arquivo /etc/neutron/neutron.conf e adicione a seguinte chave à seção [DEFAULT]:

```
[DEFAULT]
...
auth_strategy = keystone
```

b) Adicione as seguintes chaves na seção [keystone_authtoken]:
Substitua NEUTRON_PASS pela senha definida para o usuário *neutron* no **Keystone**.
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_host = controller
auth_protocol = http
auth_port = 35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS

2- Configure a Rede para usar o *broker* de mensagens.
a) Edite o arquivo sdgt e adicione as seguintes chaves à seção [DEFAULT]:
Substitua RABBIT_PASS pela senha escolhida para o usuário *guest* do RabbitMQ.
[DEFAULT]
...
rpc_backend = neutron.openstack.common.rpc.impl_kombu
rabbit_host = controller
rabbit_password = RABBIT_PASS

3- Configure a Rede para usar o plugin modular da camada 2 (ML2) e seus serviços associados:

a) Edite o arquivo /etc/neutron/neutron.conf e adicione as seguintes chaves à seção [DEFAULT]:
[DEFAULT]
...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True

OBS: É recomendável adicionar verbose = True na seção [DEFAULT] do arquivo /etc/neutron/neutron.conf para auxiliar na resolução de problemas.

4- Deixe todas as linhas da seção [service_providers] **comentadas**.

Para configurar o agente da camada 3 (L3)
O agente da camada 3 fornece serviços de roteamento para as redes virtuais das instâncias.

- Edite o arquivo /etc/neutron/l3_agent.ini e adicione as seguintes chaves à seção [DEFAULT]:
  [DEFAULT]
  ...

interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
use_namespaces = True

OBS: É recomendável adicionar a opção verbose = True na seção [DEFAULT] do arquivo /etc/neutron/l3_agent.ini para auxiliar na resolução de problemas.

Para configurar o agente DHCP

O agente DHCP fornece serviços de DHCP para as redes virtuais das instâncias.

- Edite o arquivo /etc/neutron/dhcp_agent.ini e adicione as seguintes chaves à seção [DEFAULT]:
  [DEFAULT]
  ...
  interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
  dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
  use_namespaces = True

OBS: É recomendável adicionar a opção **verbose = True** na seção [DEFAULT] do arquivo /etc/neutron/dhcp_agent.ini para auxiliar na resolução de problemas.

Para configurar o agente de metadados

O agente de metadados providencia informação de configuração, como credenciais para acesso remoto das instâncias.

1- Edite o arquivo /etc/neutron/metadata_agent.ini e adicione as seguintes chaves à seção [DEFAULT]:

Substitua NEUTRON_PASS pela senha que foi estabelecida para o usuário *neutron* do **Keystone**. Substitua METADATA_SECRET por um segredo para o *proxy* de metadados.

[DEFAULT]
...
auth_url = http://controller:5000/v2.0
auth_region = regionOne
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
nova_metadata_ip = controller
metadata_proxy_shared_secret = METADATA_SECRET

OBS: No caso do LSD, foi gerado um usando o openssl, com resultado:
metadata_proxy_shared_secret = ea0a45ba213d7e5b11ad

OBS: É recomendável adicionar a opção verbose = True na seção [DEFAULT] do arquivo /etc/neutron/metadata_agent.ini para auxiliar na resolução de problemas.

OBS2: Estes passos acima eram no nó de Rede. Os próximos **dois passos** devem ser realizados no nó **controller**.

2- No nó **controller**, edite o arquivo /etc/nova/nova.conf e adicione as seguintes chaves à seção [DEFAULT]:
Substitua METADATA_SECRET pelo segredo que foi escolhido para o proxy de metadados.

```
[DEFAULT]
...
service_neutron_metadata_proxy = true
neutron_metadata_proxy_shared_secret = METADATA_SECRET
```

3- No nó **controller**, reinicie o serviço de **API do Compute**.

Para configurar o plugin modular de camada 2 (ML2)
O plugin ML2 usa o mecanismo (agente) Open vSwitch (OVS) para construir arcabouços de redes virtuais para as instâncias.
Edite o arquivo /etc/neutron/plugins/ml2/ml2_conf.ini e:
Adicione as seguintes chaves à seção [ml2]:

```
[ml2]
...
type_drivers = gre
tenant_network_types = gre
mechanism_drivers = openvswitch
```

As seguintes chaves á seção [ml2_type_gre]:

```
[ml2_type_gre]
...
tunnel_id_ranges = 1:1000
```

Adicione a seção [ovs] ao arquivo e insira as seguintes chaves nela:
Substitua INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS pelo endereço IP da interface que foi escolhida para criar os túneis no seu nó de rede.

```
[ovs]
...
local_ip = INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS
tunnel_type = gre
enable_tunneling = True
```

Adicione a seção [securitygroup] ao arquivo e insira as seguintes chaves nela:

```
[securitygroup]
...
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
enable_security_group = True
```

Para configurar o serviço Open vSwitch (OVS)
O serviço OVS providencia o arcabouço das redes virtuais inferiores para as instâncias. A ponte de integração **br-int** controla o tráfego de rede interno das instâncias no OVS. A porta externa **br-ex** controla o tráfego externo das instâncias no OVS. A ponte externa precisa de uma porta na interface física externa para fornecer acesso externo para as instâncias. Em essência, esta porta faz a ponte das redes física externa e a virtual em seu ambiente.
1- Reinicie o serviço OVS:

```
# service openvswitch-switch restart
```

2- Adicione a ponte de integração:

```
# ovs-vsctl add-br br-int
```

3- Adicione a ponte externa:

```
# ovs-vsctl add-br br-ex
```

4- Adicione uma porta para a ponte externa que conecta à interface física externa (no exemplo da figura do começo que mostra a arquitetura é **eth2**):

```
# ovs-vsctl add-port br-ex eth2
```

OBS: Dependendo do driver da interface de rede, você pode precisar desligar o *Generic Receive Offload (GRO)* para obter uma vazão aceitável entre suas instâncias e a rede externa. Para desligar temporariamente o GRO na interface de rede externa enquanto testa o ambiente:
ethtool -K eth2 gro off

Para finalizar a instalação
Reinicie os serviços de rede:

```
# service neutron-plugin-openvswitch-agent restart
# service neutron-l3-agent restart
# service neutron-dhcp-agent restart
# service neutron-metadata-agent restart
```

5.1.2.3- Configurando o nó de *compute*
Pré-requisitos
Antes de configurar a Rede do OpenStack, você precisa habilitar certas funções de rede do kernel.
1- Edite o arquivo /etc/sysctl.conf para conter o seguinte:
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0

2- Aplique as mudanças:

```
# sysctl -p
```

Para instalar os componentes de Rede

```
# apt-get install neutron-common neutron-plugin-ml2 neutron-plugin-openvswitch-agent \
  openvswitch-datapath-dkms
```

OBS: Instalações do Ubuntu usando versões de kernel 3.11 em diante não precisam instalar o pacote *openvswitch-datapath-dkms*.

Para configurar os componentes comuns de Rede

A configuração dos componentes comuns de Rede incluem um mecanismo de autenticação, um *broker* de mensagens, e plugin.

1- Configure a Rede para usar o Serviço de Identidade para autenticação:

a) Edite o arquivo /etc/neutron/neutron.conf e adicione a seguinte chave à seção [DEFAULT]:

[DEFAULT]
...
auth_strategy = keystone

b) Adicione as seguintes chaves à seção [keystone_authtoken]:

Substitua NEUTRON_PASS pela senha que foi escolhida para o usuário *neutron* no Serviço de Identidade.

[keystone_authtoken]

...
auth_uri = http://controller:5000
auth_host = controller
auth_protocol = http
auth_port = 35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS

2- Configure a Rede para usar o *broker* de mensagens.

a) Edite o arquivo /etc/neutron/neutron.conf e adicione as seguintes chaves à seção [DEFAULT]:

Substitua RABBIT_PASS pela senha escolhida para o usuário *guest* do RabbitMQ.

[DEFAULT]
...
rpc_backend = neutron.openstack.common.rpc.impl_kombu
rabbit_host = controller
rabbit_password = RABBIT_PASS

3- Configure a Rede para usar o plugin modular da camada 2 (ML2) e seus serviços:

a) Edite o arquivo /etc/neutron/neutron.conf e adicione as seguintes chaves à seção [DEFAULT]:

[DEFAULT]
...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True

OBS: É recomendável adicionar verbose = True na seção [DEFAULT] do arquivo /etc/neutron/neutron.conf para auxiliar na resolução de problemas.

4- Comente todas as linhas da seção [service_providers] do mesmo arquivo.

Para configurar o plugin modular da camada 2 (ML2)
O plugin ML2 usa o mecanismo (agente) Open vSwitch (OVS) para construir o arcabouço da rede virtual para as instâncias.

- Edite o arquivo /etc/neutron/plugins/ml2/ml2_conf.ini :
  Adicione as seguintes chaves à seção [ml2]:
  [ml2]
  ...
  type_drivers = gre
  tenant_network_types = gre
  mechanism_drivers = openvswitch

  Adicione as seguintes chaves à seção [ml2_type_gre]:
  [ml2_type_gre]
  ...
  tunnel_id_ranges = 1:1000

  Adicione a seção [ovs] ao arquivo e insira as seguintes chaves nela:
  Substitua INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS com o endereço IP da interface de rede dos túneis de instâncias no seu nó de **compute**.
  [ovs]
  ...
  local_ip = INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS
  tunnel_type = gre
  enable_tunneling = True

  Adicione a seção [securitygroup] à seção e insira as seguintes chaves nela:
  [securitygroup]
  ...
  firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
  enable_security_group = True

Para configurar o serviço Open vSwitch (OVS)

O serviço OVS providencia o arcabouço de redes virtuais para as instâncias. A ponte de integração **br-int** controla o tráfego de rede interno no OVS.
1- Reinicie o serviço OVS:

```
# service openvswitch-switch restart
```

2- Adicione a ponte de integração:

```
# ovs-vsctl add-br br-int
```

Para configurar o *compute* para usar o Serviço de Rede
Por padrão, a maioria das distribuições configura o **compute** para usar a rede legada. Você deve reconfigurar o **compute** para gerenciar as redes pelo Serviço de Rede
Edite o arquivo /etc/nova/nova.conf e adicione as seguintes chaves à seção [DEFAULT]:
Substitua NEUTRON_PASS pela senha que foi escolhida para o usuário *neutron* no **Keystone**.

```
[DEFAULT]
...
network_api_class = nova.network.neutronv2.api.API
neutron_url = http://controller:9696
neutron_auth_strategy = keystone
neutron_admin_tenant_name = service
neutron_admin_username = neutron
neutron_admin_password = NEUTRON_PASS
neutron_admin_auth_url = http://controller:35357/v2.0
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
security_group_api = neutron
```

OBS: Por padrão, o **compute** usa um serviço interno de firewall. Uma vez que o Serviço de Rede inclui um serviço de firewall, você deve desabilitar o serviço de firewall do **compute** ao usar o driver de firewall nova.virt.firewall.NoopFirewallDriver.

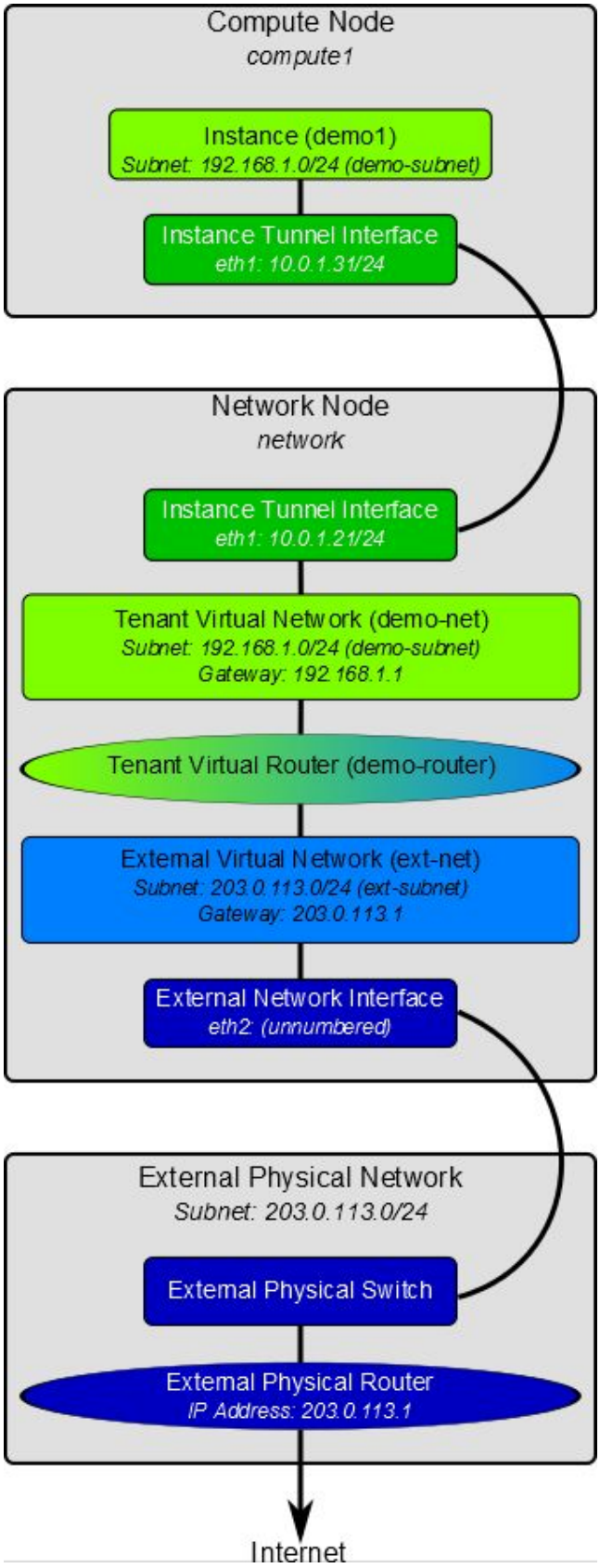
Para finalizar a instalação
1- Reinicie o serviço de **compute**:

```
# service nova-compute restart
```

2- Reinicie o agente Open vSwitch (OVS):

```
# service neutron-plugin-openvswitch-agent restart
```

5.1.2.4- Criando as redes iniciais
Antes de iniciar a primeira instância, você deve criar a infraestrutura de rede virtual para a qual a instância vai conectar, incluindo a rede externa e a rede do alocador. Veja a Figura abaixo. Depois de criar esta infraestrutura, é recomendável verificar a conectividade e resolver quaisquer problemas pendentes antes de prosseguir.


Compute Node
compute1
Instance (demo1)
Subnet: 192.168.1.0/24 (demo-subnet)
Instance Tunnel Interface
eth1: 10.0.1.31/24
Network Node
network
Instance Tunnel Interface
eth1: 10.0.1.21/24
Tenant Virtual Network (demo-net)
Subnet: 192.168.1.0/24 (demo-subnet)
Gateway: 192.168.1.1
Tenant Virtual Router (demo-router)
External Virtual Network (ext-net)
Subnet: 203.0.113.0/24 (ext-subnet)
Gateway: 203.0.113.1
External Network Interface
eth2: (unnumbered)
External Physical Network
Subnet: 203.0.113.0/24
External Physical Switch
External Physical Router
IP Address: 203.0.113.1
Internet

5.1.2.4.1- Rede externa

A rede externa tipicamente fornece acesso à Internet para suas instâncias. Por padrão, esta rede permite acesso à internet apenas *a partir* das instâncias usando *Network Address Translation **(NAT)***. Você pode habilitar o acesso à Internet para instâncias individuais usando um endereço IP flutuante (*floating IP*) e regras de grupos de segurança adequados. O alocador *admin* é o dono desta rede porque providencia acesso externo de rede para múltiplos alocadores. Você também deve habilitar o compartilhamento para permitir acesso por estes outros alocadores.

OBS: Os comandos a seguir são executados no nó **controller**.

Para criar a rede externa

1- Faça *source* nas credencias do alocador *admin*:

```
$ source admin-openrc.sh
```

2- Crie a rede:

```
$ neutron net-create ext-net --shared --router:external=True
```

Created a new network:

```
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 893aebb9-1c1e-48be-8908-6b947f3237b3 |
| name                      | ext-net                              |
| provider:network_type     | gre                                  |
| provider:physical_network |                                      |
| provider:segmentation_id  | 1                                    |
| router:external           | True                                 |
| shared                    | True                                 |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 54cd044c64d5408b83f843d63624e0d8     |
+---------------------------+--------------------------------------+
```

Assim como em uma rede física, uma rede virtual precisa de uma subrede atribuída a ela. A rede externa compartilha a mesma subrede e *gateway* associada com a rede física conectada a interface de rede externa no nó de Rede. Você deve especificar uma fatia exclusiva desta subrede para os endereços IP do roteador e endereços IP flutuantes para evitar interferência com outros dispositivos na rede externa.

Substitua FLOATING_IP_START e FLOATING_IP_END pelos primeiro e último endereços daquela faixa que você deseja alocar para os endereços de IP flutuante. Substitua EXTERNAL_NETWORK_CIDR pela subrede associada à rede física. Substitua

EXTERNAL_NETWORK_GATEWAY pelo *gateway* associado à rede física, geralmente o endereço IP ".1". Você deve desabilitar o DHCP nesta subrede, porque as instâncias não se conectam diretamente à rede externa, e os endereços IP flutuantes exigem a atribuição manual.

Para criar uma subrede na rede externa
Crie a subrede:

```
$ neutron subnet-create ext-net --name ext-subnet \
  --allocation-pool start=FLOATING_IP_START,end=FLOATING_IP_END \
  --disable-dhcp --gateway EXTERNAL_NETWORK_GATEWAY EXTERNAL_NETWORK_CIDR
```

Por exemplo, usando 203.0.113.0/24 com a faixa de IPs flutuantes de 203.0.113.101 até 203.0.113.200:

```
$ neutron subnet-create ext-net --name ext-subnet \
  --allocation-pool start=203.0.113.101,end=203.0.113.200 \
  --disable-dhcp --gateway 203.0.113.1 203.0.113.0/24
```

Outro exemplo:

```
$ neutron subnet-create ext-net --name ext-subnet \
  --allocation-pool start=10.2.0.10,end=10.2.0.40 \
  --disable-dhcp --gateway 10.2.0.4 10.2.0.0/16
```

Created a new subnet:

```
+-------------------+------------------------------------------------------+
| Field             | Value                                                |
+-------------------+------------------------------------------------------+
| allocation_pools  | {"start": "203.0.113.101", "end": "203.0.113.200"}   |
| cidr              | 203.0.113.0/24                                       |
| dns_nameservers   |                                                      |
| enable_dhcp       | False                                                |
| gateway_ip        | 203.0.113.1                                          |
| host_routes       |                                                      |
| id                | 9159f0dc-2b63-41cf-bd7a-289309da1391                 |
| ip_version        | 4                                                    |
| ipv6_address_mode |                                                      |
| ipv6_ra_mode      |                                                      |
| name              | ext-subnet                                           |
| network_id        | 893aebb9-1c1e-48be-8908-6b947f3237b3                 |
| tenant_id         | 54cd044c64d5408b83f843d63624e0d8                     |
+-------------------+------------------------------------------------------+
```

5.1.2.4.2- Rede de alocador (tenant network)
A rede do alocador fornece acesso de rede interno para as instâncias. Esta arquitetura isola este tipo de rede de outros alocadores. O alocador *demo* é o dono desta rede porque permite acesso de rede apenas para as instâncias dentro deste.

OBS: Execute os comandos a seguir no nó **controller**.

Para criar a rede do alocador

1- Faça *source* nas credenciais do alocador *demo*:

```
$ source demo-openrc.sh
```

2- Crie a rede

```
$ neutron net-create demo-net
```

Created a new network:

Field	Value
admin_state_up	True
id	ac108952-6096-4243-adf4-bb6615b3de28
name	demo-net
shared	False
status	ACTIVE
subnets	
tenant_id	cdef0071a0194d19ac6bb63802dc9bae

Assim como em uma rede externa, a rede do alocador também precisa de uma subrede associada a ela. Você pode especificar qualquer subrede válida, porque esta arquitetura isola redes de alocadores. Substitua TENANT_NETWORK_CIDR pela subrede que você deseja associar à rede do alocador. Substitua TENANT_NETWORK_GATEWAY pelo *gateway* que você deseja associar a esta rede, normalmente o endereço ".1". Por padrão, esta subrede vai usar DHCP para atribuir endereços IP a suas instâncias.

Para criar uma subrede na rede do alocador

- Crie a subrede:

```
$ neutron subnet-create demo-net --name demo-subnet \
  --gateway TENANT_NETWORK_GATEWAY TENANT_NETWORK_CIDR
```

Por exemplo, usando 192.168.1.0/24:

```
$ neutron subnet-create demo-net --name demo-subnet \
  --gateway 192.168.1.1 192.168.1.0/24
```

OBS: No nosso caso:

```
$ neutron subnet-create demo-net --name demo-subnet \
  --gateway 192.168.0.1 192.168.0.0/24
```

Created a new subnet:

```
+-------------------+-----------------------------------------------------+
| Field             | Value                                               |
+-------------------+-----------------------------------------------------+
| allocation_pools  | {"start": "192.168.1.2", "end": "192.168.1.254"}    |
| cidr              | 192.168.1.0/24                                      |
| dns_nameservers   |                                                     |
| enable_dhcp       | True                                                |
| gateway_ip        | 192.168.1.1                                         |
| host_routes       |                                                     |
| id                | 69d38773-794a-4e49-b887-6de6734e792d                |
| ip_version        | 4                                                   |
| ipv6_address_mode |                                                     |
| ipv6_ra_mode      |                                                     |
| name              | demo-subnet                                         |
| network_id        | ac108952-6096-4243-adf4-bb6615b3de28                |
| tenant_id         | cdef0071a0194d19ac6bb63802dc9bae                    |
+-------------------+-----------------------------------------------------+
```

Um roteador virtual repassa o tráfego entre duas ou mais redes virtuais. Cada roteador precisa de uma ou mais interfaces e/ou *gateways* que fornecem acesso à redes específicas. Neste caso, você pode criar um roteador e associar seu alocador e redes externas a ele.

Para criar um roteador na rede do alocador e associar as redes externas e do alocador nele

1- Crie o roteador:

```
$ neutron router-create demo-router
```

Created a new router:

```
+-----------------------+----------------------------------------+
| Field                 | Value                                  |
+-----------------------+----------------------------------------+
| admin_state_up        | True                                   |
| external_gateway_info |                                        |
| id                    | 635660ae-a254-4feb-8993-295aa9ec6418   |
| name                  | demo-router                            |
| status                | ACTIVE                                 |
| tenant_id             | cdef0071a0194d19ac6bb63802dc9bae       |
+-----------------------+----------------------------------------+
```

2- Associe o roteador à subrede do alocador *demo*:

```
$ neutron router-interface-add demo-router demo-subnet
```

Added interface b1a894fd-aee8-475c-9262-4342afdc1b58 to router demo-router.

3- Associe o roteador à rede externa ao definí-lo como *gateway*:

```
$ neutron router-gateway-set demo-router ext-net
```

Set gateway for router demo-router

5.1.2.4.3- Verificando conectividade

É recomendável que você verifique a conectividade de rede e resolva quaisquer problemas antes de prosseguir. Seguindo o exemplo de subrede da rede externa usando 203.0.113.0/24, o *gateway* do roteador do alocador deve ocupar o primeiro endereço IP da faixa de endereços, 203.0.113.101. Se você configurou sua rede física externa e redes virtuais corretamente, você deve ser capaz de fazer um ***ping*** para este endereço IP de qualquer *host* na sua rede física externa.

OBS: Se você está construindo seus nós como máquinas virtuais, você deve configurar o hipervisor para permitir o modo promíscuo na rede externa.

Para verificar a conectividade de rede

Faça um *ping* para o *gateway* do roteador:

```
$ ping -c 4 203.0.113.101
PING 203.0.113.101 (203.0.113.101) 56(84) bytes of data.
64 bytes from 203.0.113.101: icmp_req=1 ttl=64 time=0.619 ms
64 bytes from 203.0.113.101: icmp_req=2 ttl=64 time=0.189 ms
64 bytes from 203.0.113.101: icmp_req=3 ttl=64 time=0.165 ms
64 bytes from 203.0.113.101: icmp_req=4 ttl=64 time=0.216 ms
```

5.2- Rede legada - serviço nativo (nova-network)

5.2.1- Configurar o nó controller

A rede legada envolve primariamente os nós de **compute**. Entretanto, você deve configurar o nó **controller** para usá-la.

Para configurar a rede legada

1- Edite o arquivo /etc/nova/nova.conf e adicione as seguintes chaves à seção [DEFAULT]:

```
[DEFAULT]
...
network_api_class = nova.network.api.API
        security_group_api = nova
```

2- Reinicie os serviços de compute:

```
# service nova-api restart
# service nova-scheduler restart
# service nova-conductor restart
```

5.2.2- Configure o nó de compute

Esta seção cobre a implantação de uma rede plana que fornece endereços IP para suas instâncias por DHCP. Se seu ambiente inclui múltiplos nós de ***compute***, a funcionalidade *multi-host* providencia redundância ao espalhar as funções de rede pelos nós de ***compute***.

Para instalar os componentes de rede legada

```
# apt-get install nova-network nova-api-metadata
```

Para configurar a rede legada
1- Edite o arquivo /etc/nova/nova.conf e adicione as seguintes chaves à seção [DEFAULT]:
[DEFAULT]
...
network_api_class = nova.network.api.API
security_group_api = nova
firewall_driver = nova.virt.libvirt.firewall.IptablesFirewallDriver
network_manager = nova.network.manager.FlatDHCPManager
network_size = 254
allow_same_net_traffic = False
multi_host = True
send_arp_for_ha = True
share_dhcp_address = True
force_dhcp_release = True
flat_network_bridge = br100
flat_interface = eth1
public_interface = eth1

2- Reinicie os serviços:
```
# service nova-network restart
# service nova-api-metadata restart
```

5.2.3- Crie a rede inicial
Antes de iniciar sua primeira instância, você deve criar a infraestrutura de rede virtual necessária para qual a instância vai se conectar. Esta rede tipicamente fornece acesso à Internet *a partir das* instâncias. Você pode ativar o acesso à Internet *para* instâncias individuais usando um endereço IP flutuante e as regras de grupos de segurança adequadas. O alocador *admin* é o dono desta rede porque provê acesso à rede externa para múltiplos alocadores.

Esta rede compartilha a mesma subrede associada com a rede física conectada à interface externa no nó de ***compute***. Você deve especificar uma fatia exclusiva desta subrede para evitar interferência com outros dispositivos na rede externa.

OBS: Realize este procedimento no nó **controller**.

Para criar a rede
1- Faça um *source* nas credenciais do alocador *admin*:
```
$ source admin-openrc.sh
```

2- Crie a rede:
Substitua NETWORK_CIDR com a subrede associada a rede física.

```
$ nova network-create demo-net --bridge br100 --multi-host T \
  --fixed-range-v4 NETWORK_CIDR
```

Por exemplo, usando uma fatia exclusiva da rede 203.0.113.0/24 com a faixa de endereços IP de 203.0.113.24 até 203.0.113.32:

```
$ nova network-create demo-net --bridge br100 --multi-host T \
  --fixed-range-v4 203.0.113.24/29
```

OBS: Este comando não apresenta uma saída na tela.

3- Verifique a criação da rede:

```
$ nova net-list
+--------------------------------------+----------+------------------+
| ID                                   | Label    | CIDR             |
+--------------------------------------+----------+------------------+
| 84b34a65-a762-44d6-8b5e-3b461a53f513 | demo-net | 203.0.113.24/29  |
+--------------------------------------+----------+------------------+
```

5.3- Próximos passos

Seu ambiente OpenStack agora inclui todos os componentes centrais necessários para iniciar uma instância básica. Você pode iniciar uma instância para testar seu ambiente, ou adicionar mais serviços no seu ambiente nas seções seguintes.

6- Adicionando o painel - *Dashboard* - HORIZON

6.1- Requisitos de Sistema

Os seguintes requisitos devem ser atendidos:

- Instalação do **OpenStack Compute**. Ative o Serviço de Identidade para o gerenciamento de usuário e de projeto. Atentar aos *endpoints* usados;
- Serviço de Identidade com privilégios sudo. Como o Apache não fornece conteúdo a partir de um usuário *root*, os usuários devem rodar o painel como um **usuário do Serviço de Identidade** com privilégios sudo;
- Versão **2.6** ou **2.7** do **Python**. A versão do Python deve suportar **Django**. A versão do Python deve ser possível de rodar em qualquer sistema, incluindo Mac OSX. Os requisitos de instalação podem mudar com a plataforma.

Os usuários do **Horizon** vão precisar do IP público e de um usuário/senha para acessar o painel.
O navegador web deve suportar **HTML5**, e ter ativado **cookies** e **JavaScript.**

OBS: Para usar o cliente VNC a partir do *dashboard*, o navegador deve suportar o HTML5 Canvas e HTML5 WebSockets. Para detalhes sobre os navegadores que suportam noVNC, veja este e este link, respectivamente.

6.2- Instalando o painel

1- Instale o *dashboard* em um nó que possa contactar o Serviço de Identidade como *root*:

```
# apt-get install  apache2 memcached libapache2-mod-wsgi
openstack-dashboard
```

OBS: No Ubuntu é bom remover o pacote openstack-dashboard-ubuntu-theme , para evitar que ocorram traduções, e que vários menus e o mapa de rede de serem renderizados corretamente. Use o comando abaixo:

```
# apt-get remove --purge openstack-dashboard-ubuntu-theme
```

2- Modifique os campos CACHES[‘default’] e [‘LOCATION’] no arquivo /etc/openstack-dashboard/local_settings.py para bater com aqueles em /etc/memcached.conf .

Procure esta linha no arquivo local_settings.py:

```
CACHES = {
'default': {
'BACKEND' : 'django.core.cache.backends.memcached.MemcachedCache',
'LOCATION' : '127.0.0.1:11211'
}
}
```

OBS:

- Os endereços e portas devem ser os mesmos que em /etc/memcached.conf . Se alguma configuração for mudada neste arquivo, o servidor web do Apache deve ser reiniciado para que as mudanças tenham efeito;
- Você pode usar mais opções além da *memcached* para armazenamento de sessão. Configure o *back-end* da sessão com a opção **SESSION_ENGINE**.
- Para mudar a *timezone*, use o **dashboard** ou edite o arquivo /etc/openstack-dashboard/local_settings.py , no parâmetro TIME_ZONE = "UTC" .

3- Atualize o campo ALLOWED_HOSTS no arquivo /etc/openstack-dashboard/local_settings.py para incluir os endereços que poderão ter acesso ao *Dashboard*.
ALLOWED_HOSTS = ['localhost', 'my-desktop']

Exemplo: ALLOWED_HOSTS = "*" permite todo tipo de máquina conectar ao **Horizon**.

4- Esse guia assume que você está rodando o serviço do *Dashboard* no nó **controller**. Você pode rodar o *Dashboard* em outro servidor, ao mudar o parâmetro OPENSTACK_HOST no arquivo de configuração /etc/openstack-dashboard/local_settings.py .
OPENSTACK_HOST = "controller"

5- Inicie o servidor web **Apache** e o **memcached**:

```
# service apache2 restart
# service memcached restart
```

6- Agora, você pode acessar o dashboard em http://controller/horizon . Use as credenciais que você criou com o Serviço de Identidade do OpenStack.

6.3- Configurando o armazenamento de Sessão para o Dashboard

O Dashboard faz uso do arcabouço *Django sessions* para lidar com os dados de sessão de usuário. Você pode usar outros *back end* de sessão, editando o arquivo /etc/openstack-dashboard/local_settings.py , para o caso do Ubuntu. Esta sessão apresenta as vantagens e desvantagens de cada opção para *back end*.

6.3.1- Cache de memória local

Esta é a opção padrão do OpenStack, e possui a vantagem de não possuir dependência alguma. Entretanto, também não possui persistência, e portanto, não é recomendado para um ambiente de produção ou de uso em si. Ele é ativado em:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
        'BACKEND': 'django.core.cache.backends.locmem.LocMemCache'
}
```

6.3.2- Armazenamento por valor de chave (Key-value)

É possível usar aplicações para cache externo, como **Memcached** ou **Redis**. Essas aplicações permitem que se obtenha persistência e armazenamento compartilhado para um uso mais sério do OpenStack.

Memcached

O memcached possui alto desempenho e um sistema de armazenamento em cache de objetos distribuído. Os requisitos para adotá-lo são:

- O serviço memcached funcionando e acessível;
- O módulo Python **python-memcached** instalado.

Ele pode ser ativado por:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache'
        'LOCATION': 'my_memcached_host:11211',
}
```

Redis

Redis é um sistema em código aberto, e licenciado para o BSD. É comumente referenciado por ser um servidor de estrutura de dados. Os requisitos para adotá-lo são:

- Serviço Redis rodando e acessível;
- Os módulos Python **redis** e **django-redis** instalados.

Ele pode ser ativado por:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
        "default": {
                "BACKEND": "redis_cache.cache.RedisCache",
                "LOCATION": "127.0.0.1:6379:1",
                "OPTIONS": {
                        "CLIENT_CLASS": "redis_cache.client.DefaultClient",
                }
        }
}
```

6.3.3- Inicializar e configurar o Banco de Dados

Sessões apoiadas por Banco de Dados são escaláveis, persistentes e podem ser definidas como alta-concorrência e alta-disponibilidade. Entretanto, trata-se de um dos armazenamentos mais lentos e, sob alta carga, podem gerar uma quantidade alta de *overhead*. Uma configuração adequada da disposição do Banco de Dados pode ser bastante trabalhosa e está além do escopo deste guia.

1- Inicie o cliente de mysql por linha de comando:

```
$ mysql -u root -p
```

2- Entre com os dados do usuário *root*.

3- Para configurar o Banco de Dados MySQL, crie o BD chamado *dash*:

```
mysql> CREATE DATABASE dash;
```

4- Crie um usuário MySQL para o Banco de Dados que acabou de ser criado, que tenha controle total do BD. Substitua *DASH_DBPASS* por uma senha para o novo usuário:

```
mysql> GRANT ALL ON dash.* TO 'dash'@'%' IDENTIFIED BY 'DASH_DBPASS';
mysql> GRANT ALL ON dash.* TO 'dash'@'localhost' IDENTIFIED BY 'DASH_DBPASS';
```

5- Saia do cliente usando o comando **quit**;

6- No arquivo de configuração /etc/openstack-dashboard/local_settings.py , mude estas opções:

```
SESSION_ENGINE = 'django.core.cache.backends.db.DatabaseCache'
DATABASES = {
        'default': {
```

```
                # Database configuration here
                'ENGINE': 'django.db.backends.mysql',
                'NAME': 'dash',
                'USER': 'dash',
                'PASSWORD': 'DASH_DBPASS',
                'HOST': 'localhost',
                'default-character-set': 'utf8'
        }
}
```

7- Depois de configurar o arquivo local_settings conforme acima, rode o comando **manage.py syncdb** para povoar este BD que acabou de ser criado.

```
$ /usr/share/openstack-dashboard/manage.py syncdb
```

Exemplo de Saida no Ubuntu:

openstack@guaricema:~$ sudo /usr/share/openstack-dashboard/manage.py syncdb
Creating tables ...
Creating table django_content_type
Creating table auth_permission
Creating table auth_group_permissions
Creating table auth_group
Creating table auth_user_groups
Creating table auth_user_user_permissions
Creating table auth_user
Creating table django_session

You just installed Django's auth system, which means you don't have any superusers defined.
Would you like to create one now? (yes/no): no
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)

OBS: É mostrada uma saída, mas aparentemente é para o openSUSE. A saída segue abaixo:
Installing custom SQL ...
Installing indexes ...
DEBUG:django.db.backends:(0.008) CREATE INDEX `django_session_c25c2c28` ON `django_session` (`expire_date`);; args=()
No fixtures found.

8- Para o **Ubuntu**: reiniciar o apache2 pode gerar um *warning*. Isto pode ser evitado, ao criar um diretório *blackhole* no diretório do *dashboard*, como a seguir:

```
# sudo mkdir -p /var/lib/dash/.blackhole
```

9- Reinicie o Apache para apanhar as configurações de *site* default e *link* simbólico:
```
# /etc/init.d/apache2 restart
```

10- Para o **Ubuntu**: reinicie também o serviço **nova-api** para garantir que o servidor de API possa conectar ao **dashboard** sem erro:
```
# sudo restart nova-api
```

6.3.4- Banco de Dados em cache

Para mitigar os problemas de desempenho nas requisições do BD, você pode usar o *back end* de sessão **Django** cached_db, que utiliza tanto o BD quanto a estrutura em cache para realizar escritas em cache e recuperação eficiente.
Esta opção pode ser habilitada híbrida ao configurar tanto o BD quanto o cache (que foi feito anteriormente), e depois adicionando o seguinte valor ao campo SESSION_ENGINE:
SESSION_ENGINE = "django.contrib.sessions.backends.cached_db"

OBS: Esse campo vai ficar assim provavelmente em ambos os arquivos de configuração.

6.3.5- Cookies

Se a versão 1.4 ou superior do Django estiver sendo utilizada, o *back end* **signed_cookies** evita sobrecarga de servidor e problemas de escalabilidade. Ao usar armazenamento via *Cookies*, os dados de sessão ficam salvos no navegador do usuário, com uma técnica de assinatura criptográfica, que embora não garanta que o conteúdo não possa ser lido, faz com que este esquema não possua nenhuma dependência para ser adotado. Uma outra grande desvantagem é que os dados terão que ser passados pela conexão, e que a capacidade de armazenamento é limitada.
Para mais detalhes, consulte este link: cookie-based sessions.

6.4- Próximos passos

Seu ambiente OpenStack agora inclui o *dashboard*. Você pode iniciar uma instância ou adicionar mais serviços ao ambiente, mencionados nas seções a seguir.

7- Adicionar o Serviço de Armazenamento em Bloco - CINDER

O Serviço de Armazenamento em Bloco funciona por meio da interação de uma série de monitores chamados **cinder-*** que residem persistentemente nos *hosts*.

7.1- Componentes do Serviço de Armazenamento em Bloco

O Serviço de Armazenamento em Bloco permite o gerenciamento de **volumes** (em termos de discos), **volume snapshots**, e **volume types**. Os seguintes componentes estão inclusos:

- **cinder-api** - Aceita requisições de API e as direciona para **cinder-volume** para tomar uma ação;

- **cinder-volume**- Responde a requisições de leitura e gravação para o BD do Armazenamento de Bloco para manter estado, interagindo com outros processos através de uma fila de mensagens e diretamente com o armazenamento de bloco provendo *hardware* ou *software*. Pode interagir com uma variedade de provedores de armazenamento através de uma arquitetura de *driver*.
- **cinder-scheduler** *daemon*- Assim como o **nova-scheduler**, seleciona o nó provedor de armazenamento em bloco mais adequado para criar o *volume;*
- **Fila de mensagens**- Direciona informação entre os processos do Serviço de Armazenamento em Bloco.

O Serviço de Armazenamento em Bloco interage com o **Compute** para fornecer *volumes* para as instâncias.

7.2- Configurando um controlador do Serviço de Armazenamento em Bloco

OBS: Este guia assume que você está instalando os serviços principais do **Cinder** no nó **controller**, e que um segundo nó é utilizado para o armazenamento com o serviço **cinder-volume.** O procedimento para configurar o nó adicional está na próxima sessão.

OBS: *No caso do CITTA, temos um nó cinder-server que ira conter os serviços cinder-api e cinder scheduler. O que será feito no controller é a criação db, de usuarios e afins do keystone e redirecionamento do endpoint para o endereço do no cinder-server.*

O OpenStack pode utilizar vários sistemas de armazenamento. Este exemplo usa o **LVM**.

1- Instale os devidos pacotes do Serviço de Armazenamento em Bloco:

```
# apt-get install cinder-api cinder-scheduler
```

2- Configure o Armazenamento em Bloco para usar o seu BD em MySQL. Edite o arquivo de configuração /etc/cinder/cinder.conf e adicione a chave a seguir na seção [database]. Substitua *CINDER_DBPASS* pela senha para o BD do Armazenamento em Bloco que irá criar num passo adiante.

```
[database]
...
connection = mysql://cinder:CINDER_DBPASS@controller/cinder
```

OBS: Em algumas distribuições, no arquivo de configuração não irá existir a seção [database]. Então você vai lá no final do arquivo e cria ela.

3- Use a senha que você criou para acessar o BD como *root* e criar um Banco de Dados *cinder*:

```
# mysql -u root -p
mysql> CREATE DATABASE cinder;
mysql> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
```

IDENTIFIED BY '*CINDER_DBPASS*';
mysql> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' \
IDENTIFIED BY '*CINDER_DBPASS*';

4- Crie as tabelas do BD para o Serviço de Armazenamento em Bloco:
```
# sudo cinder-manage db sync
```

OBS: Pode ser necessário instalar o pacote python-mysqldb
(`sudo apt-get install python-mysqldb`) para poder rodar o db sync do cinder-server para popular o db do controller.

5- Crie um usuário *cinder*.
O Serviço do Armazenamento em Bloco faz uso deste usuário para autenticar com o **Serviço de Identidade**.
Use o alocador *service* e dê ao usuário a função *admin*.
```
# keystone user-create --name=cinder --pass=CINDER_PASS
--email=cinder@example.com
# keystone user-role-add --user=cinder --tenant=service --role=admin
```

6- Edite o arquivo de configuração /etc/cinder/cinder.conf e adicione esta seção para as credenciais do **Keystone**:
[keystone_authtoken]
auth_uri = http://controller:5000
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = cinder
admin_password = CINDER_PASS

7- Configure o Armazenamento em Bloco para usar o *broker* de mensagens **RabbitMQ**, ao definir estas chaves no arquivo de configuração /etc/cinder/cinder.conf , na seção [DEFAULT]. Substitua *RABBIT_PASS* pela senha que você escolheu para o RabbitMQ.
[DEFAULT]
rpc_backend = cinder.openstack.common.rpc.impl_kombu
rabbit_host = controller
rabbit_port = 5672
rabbit_userid = guest
rabbit_password = RABBIT_PASS

8- Registre o Serviço de Armazenamento em Bloco no **Serviço de Identidade**, para que os outros serviços do OpenStack possam localizá-lo.

```
# keystone service-create --name=cinder --type=volume \
  --description="Cinder Volume Service"

$ keystone endpoint-create \
  --service-id=$(keystone service-list | awk '/ volume / {print $2}') \
  --publicurl=http://controller:8776/v1/%\(tenant_id\)s \
  --internalurl=http://controller:8776/v1/%\(tenant_id\)s \
  --adminurl=http://controller:8776/v1/%\(tenant_id\)s
```

9- Também registre um serviço e *endpoint* para a versão 2 da API do Serviço de Armazenamento em Bloco:

```
# keystone service-create --name=cinderv2 --type=volumev2 \
  --description="Cinder Volume Service V2"

$ keystone endpoint-create \
  --service-id=$(keystone service-list | awk '/ volumev2 / {print $2}') \
  --publicurl=http://controller:8776/v2/%\(tenant_id\)s \
  --internalurl=http://controller:8776/v2/%\(tenant_id\)s \
  --adminurl=http://controller:8776/v2/%\(tenant_id\)s
```

10- Reinicie o serviço **cinder** com as novas configurações:

```
# service cinder-scheduler restart
# service cinder-api restart
```

7.3- Configure um nó do Serviço de Armazenamento em Bloco

Quando as configurações do nó **controller** estiverem completas, configure um segundo sistema para ser um nó do Serviço de Armazenamento. Este nó contém os discos que servirão **volumes**.

Uma vez mais, o sistema de armazenamento que será descrito a seguir é o LVM, mas o OpenStack pode ser configurado para ser usado com outros sistemas de armazenamento.

1- Configure o sistema como está neste guia no começo, da seção 1 (**Rede**) até a seção 5 (**Servidor de Mensagens**)

O nome do *host* será block1. Seu endereço IP será 10.0.0.41 na interface de rede para rede de controle. Certifique-se que o os nomes dos hosts e seus endereços IPs estejam todos listados no arquivo /etc/hosts de todas as máquinas envolvidas.

2- Instale os pacotes referentes ao LVM, se não já estiverem instalados:

```
# apt-get install lvm2
```

Fonte de consulta:

http://eriberto.pro.br/wiki/index.php?title=Linux_Volume_Manager_%28LVM%29

3- Crie *volumes* LVM **físicos** e **lógicos.** Este guia assume que existe um segundo disco rígido /dev/sdb usado para este propósito.

OBS: A exemplo do caso do CITTA, iremos utilizar a partição /dev/vda2 (/local) com apenas 160GB
OBS2: a partição /dev/vda2 deve estar desmontada para criação das partições lógicas. Por precaução, a linha do /etc/fstab relacionada a /local será comentada

```
# pvcreate /dev/sdb
# vgcreate cinder-volumes /dev/sdb
```

4- Crie uma entrada de filtros no arquivo /etc/lvm/lvm.conf na parte de dispostivos, para impedir que o LVM faça varredura em dispositivos utilizados pelas máquinas virtuais.

Cada item do vetor de filtros começa ou com um 'a' que significa "**accept**", ou 'r' que significa "**reject**". Os volumes físicos requeridos no *host* no Armazenamento em Bloco tem nomes que começam com 'a'. O vetor deve terminar com um "r/.*/" para rejeitar todos os dispositivos que não foram listados.
Exemplo: O dispositivo /dev/sda1 fica com volumes que são usados pelo Sistema Operacional do nó, e /dev/sdb é reservado para os volumes do **cinder-volumes**.

```
devices {
...
filter = [ "a/sda1/", "a/sdb/", "r/.*/"]
...
}
```

OBS: Você deve adicionar os volumes físicos requeridos para o LVM no *host* do Armazenamento em Bloco. Use o comando **pvdisplay** para obter uma lista ou os volumes requeridos.

5- Após configurar o Sistema Operacional, instale os pacotes apropriados para o Serviço de Armazenamento em Bloco.

```
# apt-get install cinder-volume
```

6- Edite o arquivo de configuração /etc/cinder/cinder.conf e adicione esta seção para as credenciais do **Keystone**:

```
[keystone_authtoken]
auth_uri = http://controller:5000
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = cinder
admin_password = CINDER_PASS
```

7- Configure o Armazenamento em Bloco para usar o *broker* de mensagens **RabbitMQ** ao atribuir estas configurações no arquivo /etc/cinder/cinder.conf , na seção [DEFAULT]. Substitua *RABBIT_PASS* pela senha que você escolheu para o RabbitMQ.

```
[DEFAULT]
rpc_backend = cinder.openstack.common.rpc.impl_kombu
rabbit_host = controller
rabbit_port = 5672
rabbit_userid = guest
rabbit_password = RABBIT_PASS
```

8- Configure o Armazenamento em Bloco para usar o BD MySQL. Edite o arquivo /etc/cinder/cinder.conf e adicione a seguinte chave na seção [database]. Substitua *CINDER_DBPASS* pela senha que você escolheu para o BD do Armazenamento em Bloco:

```
[database]
...
connection = mysql://cinder:CINDER_DBPASS@controller/cinder
```

OBS: também é necessário instalar o pacote “sudo apt-get install python-mysqldb”

OBS: O arquivo de configuração /etc/cinder/cinder.conf em algumas distribuições pode ter a seção [database] faltando. Você deverá adicioná-la ao fim do arquivo.

9- Configure o Armazenamento em Bloco para usar o Serviço de Imagens. O Armazenamento em Bloco precisa de acesso às imagens para criar volumes que possam fazer *boot*. Edite o arquivo /etc/cinder/cinder.conf e atualize a opção glance_host na seção [DEFAULT]:

```
[DEFAULT]
...
glance_host = controller
```

10- Reinicie o serviço **cinder** com as novas configurações:

```
# service cinder-volume restart
# service tgt restart
```

7.4 - Verificar a instalação do Armazenamento em Bloco

Para verificar que o Armazenamento em Bloco está instalado e configurado devidamente, crie um novo volume. Para mais informações sobre como gerenciar volumes, veja o Manual de Usuário do OpenStack.

1- Faça *source* no arquivo demo-openrc.sh:

```
$ source demo-openrc.sh
```

2- Use o comando **cinder create** para criar um novo volume:

```
$ cinder create --display-name myVolume 1
```

Property	Value
attachments	[]
availability_zone	nova
bootable	false
created_at	2014-04-17T10:28:19.615050
display_description	None
display_name	myVolume
encrypted	False
id	5e691b7b-12e3-40b6-b714-7f17550db5d1
metadata	{}
size	1
snapshot_id	None
source_volid	None
status	creating
volume_type	None

3- Certifique-se que o volume foi criado corretamente, com o comando **cinder list**:

```
$ cinder list
```

ID	Status	Display Name	Size	Volume Type	Bootable	Attached to
5e691b7b-12e3-40b6-b714-7f17550db5d1	available	myVolume	1	None	false	

OBS: Se o valor em **Status** for "not available", a criação do volume **falhou**. Veja os arquivos de log no diretório /var/log/cinder/ no **controller** e nós de volume para obter detalhes sobre o fracasso.

7.4.1- Testando o funcionamento do Serviço de Armazenamento em Bloco

1- Inicie uma instância pelo **Horizon** ou pela interface de linha de comando:

```
$ nova boot --flavor m1.tiny --image cirros-0.3.2-x86_64 --nic net-id=DEMO_NET_ID \
  --security-group default --key-name demo-key demo-instance1
```

OBS: Este exemplo assume que a rede usada é pelo Serviço de Rede dedicada do OpenStack (neutron).

2- Verifique que a instância está pronta para ser acessada:

```
$ nova list
```

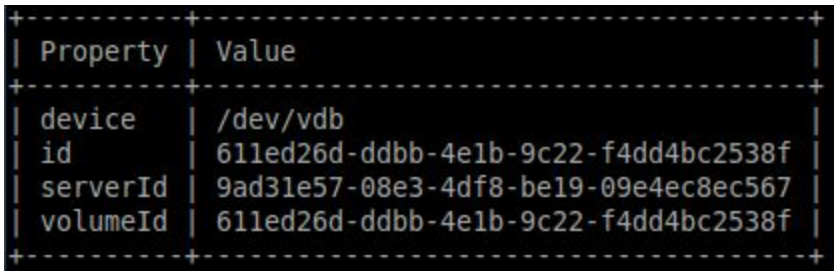
3- Crie um volume pelo **Cinder**:

```
$ cinder create --display-name nomeDoVolume 1
```

4- Verifique que o volume está pronto para o uso:
```
$ cinder list
```

5- Agora, monte o volume na instância. Este exemplo considera que a imagem do CirrOS está sendo usada:
a) Associe o volume à instância, com:
```
$ nova volume-attach id_da_instância id_do_volume
```

b) O nova irá avisar em que local da instância o volume foi montado, algo como:



Property	Value
device	/dev/vdb
id	611ed26d-ddbb-4e1b-9c22-f4dd4bc2538f
serverId	9ad31e57-08e3-4df8-be19-09e4ec8ec567
volumeId	611ed26d-ddbb-4e1b-9c22-f4dd4bc2538f

Assim, neste exemplo, o "disco" com o Volume criado foi inserido em /dev/vdb na instância.

c) É necessário formatar o Volume para que se adeque ao SO da instância usada, como por exemplo:
```
# mkfs.ext4 /dev/vdb
```

d) Agora ela pode ser montada em um diretório de sua escolha, tal como:
```
#mkdir -p /mnt/VolumeInstancia
#mount /dev/vdb /mnt/VolumeInstancia
```

6- Faça um teste, criando ou movendo um arquivo para esta partição que foi montada:
```
$ touch sabao.txt
$ echo "Teste do texto persistente." > /mnt/VolumeInstancia/sabao.txt
```

7- Desmonte o Volume nesta instância, com:
```
# umount /mnt/VolumeInstancia
```
OBS: Saia do diretório, se estiver nele.

8- Para verificar que a persistência existe, inicie uma outra instância desta mesma imagem, e monte este Volume nessa nova instância, essencialmente repetindo os passos de 1 a 5 (até a letra 'b'.
```
#mkdir -p /mnt/VolumeNovaInstancia
#mount /dev/vdb /mnt/VolumeNovaInstancia
```

OBS: Assumindo que o Volume foi montado em /dev/vdb nesta nova instância.

9- Verifique que o arquivo previamente criado está lá, com:

```
$ cat /mnt/VolumeNovaInstancia/sabao.txt
```
Teste do texto persistente.

7.5- Próximos passos

Seu ambiente OpenStack agora inclui o Armazenamento em Bloco. Você pode iniciar uma instância ou adicionar mais serviços ao ambiente, mencionados nas seções a seguir.

8- Adicione o Armazenamento de Objetos - SWIFT

Os serviços do Armazenamento de Objetos do OpenStack trabalham juntos para prover armazenamento em bloco e recuperação por meio de uma API em **REST**.

8.1- Serviço de Armazenamento de Objetos

O Serviço de Armazenamento de Objetos é um sistema altamente escalável e com múltiplos alocadores para grandes quantidades de dados desestruturados a um baixo custo por meio de suma API em HTTP *RESTful.*

Ele inclui os seguintes componentes:

- Servidores proxy (**swift-proxy-server**) - Aceita requisições de API do Armazenamento de Objetos e requisições HTTP puras para fazer *upload* de dados, modificar metadados, abrir contâineres. Ele também fornece listagens de arquivos ou de contâiners pelos navegadores web. Para melhorar o desempenho, o servidor *proxy* pode usar um cache adicional geralmente implantado com o **memcache**.
- Servidores de conta (**swift-account-server**) - Gerencia contas definidas com o Serviço de Armazenamento de Objetos.
- Servidores de contâineres (**swift-container-server**) - Gerencia um mapeamento de contâineres, ou pastas, dentro do Serviço de Armazenamento de Objetos.
- Servidores de Objetos (**swift-object-server**) - Gerencia os objetos de fato, como arquivos, nos nós de armazenamento.
- Uma quantidade de processos periódicos. Realiza tarefas de manutenção no armazenamento de grandes dados. Serviços de replicação garantem consistência e disponibilidade pelo *cluster*. Outros processos periódicos incluem *auditors*, *updaters* e *reapers*.
- Um *middleware* WSGI configurável que controla a autenticação. Geralmente o Serviço de Identidade.

8.2- Recomendações de Sistema

O Armazenamento de Objetos do OpenStack foi projetado para trabalhar com *hardware* comum.

OBS: Quando você instala apenas o Armazenamento de Objetos e o Serviço de Identidade, você não pode usar o *dashboard* até ter instalado o ***Compute*** e o Serviço de Imagem.

Segue abaixo uma tabela contendo uma descrição geral de cada componente e seus respectivos requisitos de sistema:

Tabela 2 - Recomendações de *Hardware*

Servidor	Hardware Recomendado	Observações
Servidores de Armazenamento de Objetos	Processador: dual ou quad core Memória: 8 ou 12 GB de RAM Espaço em Disco: O que tiver o menor custo por GB Rede: uma interface de rede de 1 Gbps	A quantidade de disco vai depender do quanto é possível caber no rack do servidor, eficientemente. O melhor caso é usar discos que apresentem taxas de leitura/escrita e taxas de erro padrão. Segundo a comunidade no Rackspace, os servidores de armazenamento estão usando servidores genéricos de 4U com 24 discos SATA de 2 TB, com 8 núcleos de processamento. O uso de RAID não é exigido e não-recomendado, pois o padrão de uso de disco do Swift age da pior maneira possível para RAID, e em disposições de RAID 5 ou 6 o desemepenho cai drasticamente.
Servidores de contas/contâineres	Processador: dual ou quad core Memória: 8 ou 12 GB de RAM Rede: uma interface de rede de 1 Gbps	Otimizado para IOPS por conta do rastreamente com os Banco de Dados SQLite.
Servidor de proxy do Armazenamento	Processador: dual ou quad core Rede: uma interface de rede de 1 Gbps	Quanto maior a capacidade de vazão, melhor para lidar com as requisições de API. Otimize os servidores de proxy para o melhor desempenho de CPU. Os Serviços de Proxy são mais exigentes em CPU e entrada/saída de rede. Se estiver usando uma rede de 10 Gbps para o proxy, ou o tráfego SSL termina no proxy, um processador mais robusto será necessário.

Sistema Operacional: O Armazenamento de Objetos do OpenStack atualmente roda em Ubuntu, RHEL, CentOS, Fedora, openSUSE ou SLES.
Rede: 1 Gbps ou 10 Gbps para uso interno. Para o Armazenamento de Bloco do OpenStack, uma rede externa deve conectar o mundo exterior ao proxy, e a rede de armazenamento deve estar em uma ou mais redes internas privadas.
Banco de Dados: Para o Armazenamento de Objetos do OpenStack, o Banco de Dados SQLite é parte dos processos de gerenciamento de contâineres e contas do Armazenamento de Bloco do OpenStack.
Permissões: Você pode instalar o Armazenamento de Objetos do OpenStack tanto como *root* quanto um usuário com permissões **sudo**, se você configurar o arquivo **sudoers** para ativar todas as permissões.

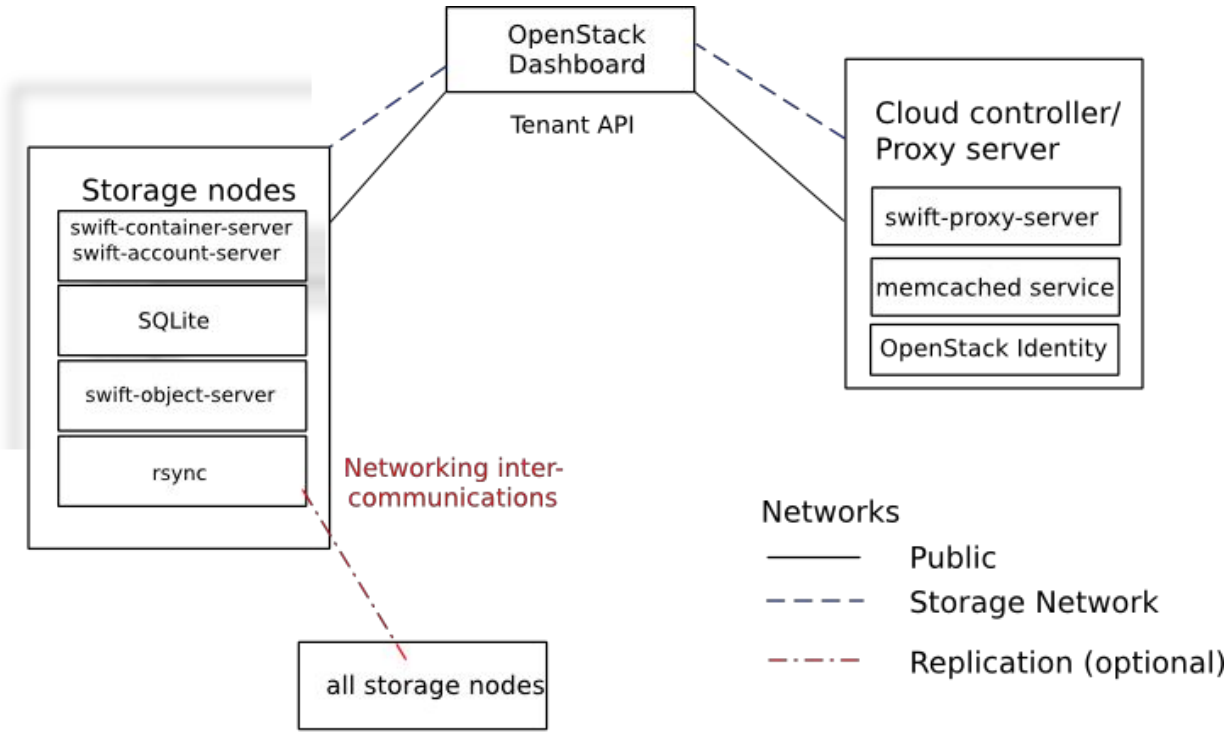
8.3- Planeje a comunicação de rede para o Armazenamento de Objetos

Tanto para conservar os recursos da rede quanto para garantir que os administradores de rede entendam as necessidades de redes e IPs públicos para prover acesso para as APIs e rede de armazenamento quanto necessário, seguem abaixo as recomendações e tamanhos mínimos. É recomendada uma rede com vazão de pelo menos 1000 Mbps.

O guia descreve as seguintes redes:

- Uma **rede pública** obrigatória, que conecta ao servidor *proxy*;
- Uma **rede de armazenamento** obrigatória, não acessível de fora do *cluster*, para se conectar a todos os nós que pertencem a este;
- Uma **rede de replicação** opcional, não acessível de fora do cluster, dedicada a replicar tráfego entre os nós de armazenamento. Deve ser configurada no anel.

A Figura abaixo mostra a arquitetura básica para a rede pública, rede de armazenamento, e rede de replicação opcional.



Por padrão, todos os serviços de Armazenamento de Objetos do OpenStack, bem como o monitor **rsync** dos nós de armazenamento são configurados para ouvirem nos seus endereços IP de STORAGE_LOCAL_NET.
Se você configurar uma rede de replicação no anel, os servidores de Objeto e Contâineres ouvem nos IPs de STORAGE_LOCAL_NET e STORAGE_REPLICATION_NET , respectivamente. O monitor **rsync** ouve apenas no IP STORAGE_REPLICATION_NET.

Rede pública (faixa de IPs publicamente roteável):

- Provê acessibilidade IP pública para os pontos de terminação da API dentro da infraestrutura de nuvem.
- Tamanho mínimo: Um endereço IP para cada servidor proxy.

Rede de Armazenamento (faixa de IP pela RFC1918, não roteável publicamente)

- Gerencia todas as comunidações entre servidores dentro da infraestrutura de Armazenamento de Objetos.
- Tamanho mínimo: Um endereço IP para cada nó de armazenamento e servidor proxy.
- Tamanho recomendado: Conforme supracitado, com espaço para expansão para o maior tamanho de *cluster* disponível, como por exemplo, 255 ou CIDR /24.

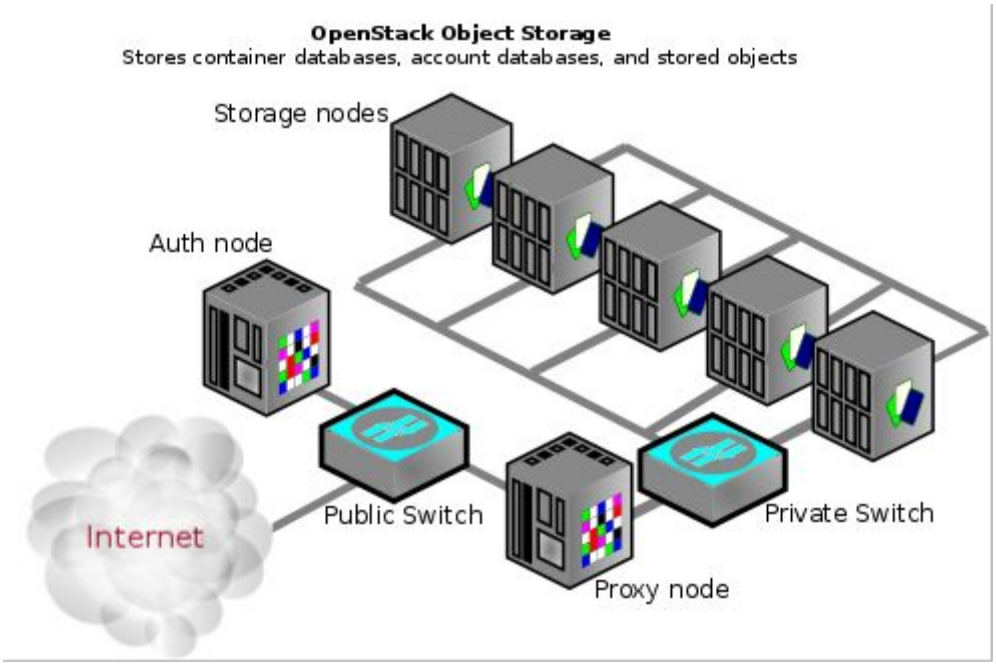
Rede de replicação (faixa de IP pela RFC1918, não roteável publicamente)

- Gerencia as comunicações relacionadas à replicação entre os servidores de armazenamento dentro da infraestrutura de Armazenamento de Objetos.
- Tamanho recomendado: conforme o parâmetro STORAGE_LOCAL_NET.

8.4- Exemplo de arquitetura da instalação do Armazenamento de Objetos

- Nó: Um sistema com um ou mais serviços de Armazenamento de Objeto do OpenStack.
- Nó *proxy*: Roda serviços de *Proxy*.
- Nó de Armazenamento: Roda serviços de Conta, Contâineres e Objetos.
- Anel: Um conjunto de mapeamentos entre os dados do Armazenamento de Objetos do OpenStack e os dispositivos físicos.
- Réplica: Uma cópia de um objeto. Por padrão, três cópias são mantidas no *cluster*.
- Zona: Uma seção lógica separada do cluster, relacionada às características de falha independentes.
- Região (opcional). Uma seção lógica separada do *cluster*, representando locais físicos distintos, como cidades ou países. Funciona como as Zonas, mas representando locais físicos em vez de segmentos lógicos.

Para aumentar a confiabilidade e desempenho, mais servidores *proxy* podem ser adicionados. Este documento descreve cada nó de armazenamento como uma zona separada no anel. No mínimo, cinco zonas são recomendadas. Uma zona é um grupo de nós que está o mais isolado dos outros nós (servidores separados, redes, alimentação e até mesmo geografia). O anel garante que cada réplica é armazenada em uma zona diferente. O diagrama abaixo mostra uma possível configuração para a instalação mínima.



8.5- Instalar o Armazenamento de Objetos

Embora seja possível instalar o Armazenamento de Objetos do OpenStack para propósitos de desenvolvimento em um servidor, uma instalação com múltiplos servidores permite obter redundância e alta disponibilidade, desejável em uma estrutura de produção.

Os passos a seguir, assumem que existe um servidor para o proxy rodando os processos **swift-proxy-server**, e cinco outros nós para o armazenamento, rodando os processos **swift-account-server**, **swift-container-server**, e **swift-object-server**.

Para realizar a instalação de um único nó para propósitos de desenvolvimento a partir do código fonte, use as Instruções ***All In One*** do Swift (Ubuntu) ou o DevStack (múltiplas distribuições). Veja http://swift.openstack.org/development_saio.html para instruções manuais ou http://devstack.org para o *all-in-one*, incluindo autenticação com o Serviço de Autenticação (**Keystone**).

8.5.1- Antes de Começar

Tenha uma cópia da mídia de instalação do Sistema Operacional disponível se você estiver instalando em um servidor novo.
Estes passos assumem que você definiu os repositórios para pacotes no seu SO conforme está na seção **Pacotes e repositórios para o OpenStack**.
Este documento demonstra como instalar um *cluster* ao usar os seguintes tipos de nós:

- Um nó de *proxy* que roda os processos **swift-proxy-server**. O servidor de *proxy* realiza requisições de *proxy* para os nós de armazenamento apropriados.
- Cinco nós de armazenamento que rodam os processos **swift-account-server**, **swift-container-server**, e **swift-object-server**, que controlam o armazenamento dos Banco de Dados das contas, dos contâineres, e também dos objetos de fato.

OBS: Podem ser usados menos nós de armazenamento inicialmente, mas um mínimo de cinco é recomendado para um cluster de produção.

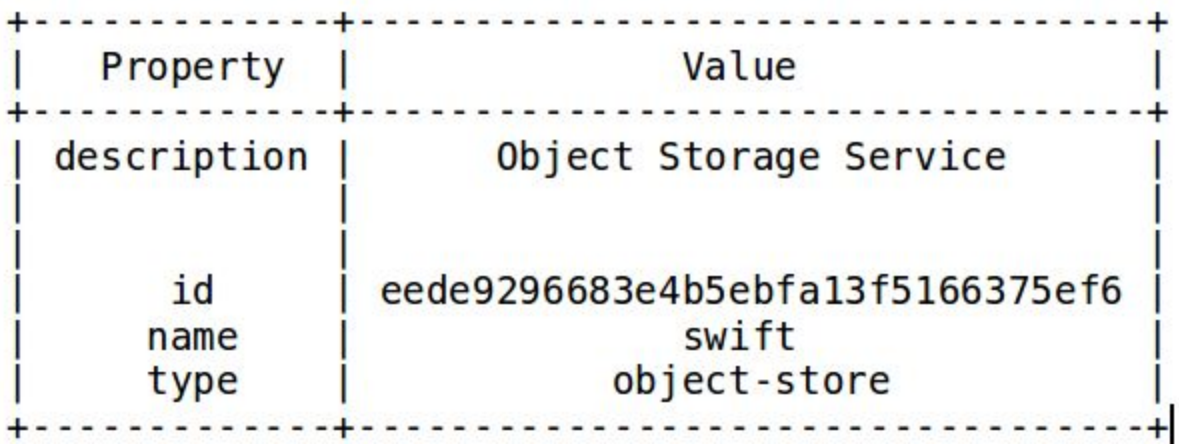
8.5.2- Passos gerais para instalação

1- Crie um usuário *swift* para que o serviço de Armazenamento de Objetos possa se comunicar com o Serviço de Identidade. Escolha uma senha e especifique um endereço de e-mail para o usuário *swift*. Use o alocador *service* e dê ao usuário a função *admin*:

```
# keystone user-create --name=swift --pass=SWIFT_PASS \
  --email=swift@example.com
# keystone user-role-add --user=swift --tenant=service --role=admin
```

2- Crie uma entrada de serviço para o serviço de Armazenamento de Objetos:

```
# keystone service-create --name=swift --type=object-store \
  --description="Object Storage Service"
```



Property	Value
description	Object Storage Service
id	eede9296683e4b5ebfa13f5166375ef6
name	swift
type	object-store

OBS: Prestar atenção no **id**, que obviamente será outro.

3- Especifique o *endpoint* da API para o serviço de Armazenamento de Objetos, usando o **id** obtido. Neste caso, você irá fornecer URLs para a API pública, a API interna, a API de administrador. Por coincidência, atentar que o nome de host **controller** foi escolhido:

```
$ keystone endpoint-create \
  --service-id=$(keystone service-list | awk '/ object-store / {print $2}')
\
  --publicurl='http://controller:8080/v1/AUTH_%(tenant_id)s' \
  --internalurl='http://controller:8080/v1/AUTH_%(tenant_id)s' \
  --adminurl=http://controller:8080
```

Property	Value
adminurl	http://controller:8080/
id	9e3ce428f82b40d38922f242c095982e
internalurl	http://controller:8080/v1/AUTH_%(tenant_id)s
publicurl	http://controller:8080/v1/AUTH_%(tenant_id)s
region	regionOne
service_id	eede9296683e4b5ebfa13f5166375ef6

OBS: O endereço do endpoint deve apontar para onde estará rodando o proxy-server. Neste caso pode ser usado a mesma máquina de controller.

4- Crie um diretório de configuração em todos os nós:

```
# mkdir -p /etc/swift
```

5- Crie um arquivo /etc/swift/swift.conf em todos os nós:

[swift-hash]
# random unique string that can never change (DO NOT LOSE)
swift_hash_path_suffix = fLIbertYgibbitZ

OBS: O valor no campo **swift_hash_path_suffix** deve ser uma string aleatória de texto, para ser usada como *sal* no processo de *hashing* para realizar mapeamentos no anel. Este arquivo deve ser **idêntico em todos os nós**.

A seguir, configure os nós de armazenamento e de *proxy*. Este exemplo usa o Serviço de Identidade para realizar autenticação comum.

8.6- Instale e configure os nós de armazenamento

OBS: O Armazenamento de Objetos funciona com qualquer formato de arquivos que suporte o Extended Attributes (XATTRS). O formato **XFS** mostrou ter o melhor desempenho para o swift, de acordo com a comunidade do **Rackspace**.

OBS: no caso do citta, nosso nó de armazenamento (swift-node) tem apenas uma particao disponivel (/local 200GB ---> /dev/vda2). Tal partição foi comentada em /etc/fstab e vai ser formatada em XFS.

1- Instale os pacotes dos nós de Armazenamento:

```
# apt-get install swift-account swift-container swift-object xfsprogs
```

2- Para cada dispositivo em um nó que você quiser usar para armazenamento, crie um volume em XFS (neste exemplo foi o /dev/sdb). Use uma única partição por disco. Por exemplo, em uma máquina com 12 discos, um ou dois podem ser usados para o sistema operacional (que não são tocados neste procedimento), e todos os restantes são formatados com uma única partição, com formato XFS.

```
# fdisk /dev/sdb
# mkfs.xfs /dev/sdb1
# echo "/dev/sdb1 /srv/node/sdb1 xfs noatime,nodiratime,nobarrier,logbufs=8
0 0" >> /etc/fstab
# mkdir -p /srv/node/sdb1
# mount /srv/node/sdb1
# chown -R swift:swift /srv/node
```

3- Crie o arquivo /etc/rsyncd.conf :

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

[account]
max connections = 2
path = /srv/node/
```

read only = false
lock file = /var/lock/account.lock

[container]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/container.lock

[object]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/object.lock

4- (OPCIONAL) Se você deseja separar os tráfegos de **rsync** e de replicação, atribua STORAGE_REPLICATION_NET_IP no lugar de STORAGE_LOCAL_NET_IP:
address = STORAGE_REPLICATION_NET_IP

5- Edite a seguinte linha no arquivo /etc/default/rsync :
RSYNC_ENABLE=true

6- Inicie o serviço **rsync**:
```
# service rsync start
```

OBS: Ele não exige nenhuma autenticação, então rode em uma rede privada interna.

7- Crie o diretório de **cache recon** do *swift* e determine suas permissões:
```
# mkdir -p /var/swift/recon
# chown -R swift:swift /var/swift/recon
```

8.7- Instale e configure o nó de proxy

O servidor de proxy pega cada requisição e procura locais para conta, contâiner ou objeto, e as direciona corretamente. O servidor de proxy também atende requisições de API. Você ativa o gerenciamento de conta ao configurar o arquivo /etc/swift/proxy-server.conf .

OBS: Os processos do Armazenamento de Objeto rodam com outro usuário e grupo, atribuídos por configurações, e por padrão, é swift:swift.

1- Instale o serviço swift-proxy:
```
# apt-get install swift swift-proxy memcached python-keystoneclient python-swiftclient
python-webob
```

2- Modifique o **memcached** para ouvir na interface local em uma máquina de rede local e não pública. Edite esta linha no arquivo /etc/memcached.conf :
-l 127.0.0.1

mudando para:
-l *PROXY_LOCAL_NET_IP*

3- Reinicie o serviço **memcached**:
```
# service memcached restart
```

4- Crie o arquivo /etc/swift/proxy-server.conf:
```
[DEFAULT]
bind_port = 8080
user = swift

[pipeline:main]
pipeline = healthcheck cache authtoken keystoneauth proxy-server

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

[filter:keystoneauth]
use = egg:swift#keystoneauth
operator_roles = Member,admin,swiftoperator

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
# Delaying the auth decision is required to support token-less
# usage for anonymous referrers ('.r:*').
delay_auth_decision = true
# cache directory for signing certificate
signing_dir = /home/swift/keystone-signing
# auth_* settings refer to the Keystone server
auth_protocol = http
auth_host = controller
auth_port = 35357
# the service tenant and swift username and password created in Keystone
admin_tenant_name = service
admin_user = swift
admin_password = SWIFT_PASS
```

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

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

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

OBS: Se você rodar múltiplos servidores **memcache**, coloque as múltiplas tuplas IP:porta no arquivo /etc/swift/proxy-server.conf , na seção [filter:cache]:
10.1.2.3:11211,10.1.2.4:11211
OBS2: Apenas o servidor de **proxy** usa o **memcache**.

5- Crie os anéis de conta, contâiner e objeto. O comando **builder** cria um arquivo builder com alguns parâmetros. O parâmetro com o valor 18 representa 2^18º, o valor que indica o tamanho da partição. Mude este valor de "potência de partição" baseado na quantidade total de espaço para armazenamento que você espera que o anel precise. O valor 3 representa o número de réplicas para cada objeto, com o último valor sendo o número de horas que deve ser aguardado para mover uma partição.

```
# cd /etc/swift
# swift-ring-builder account.builder create 18 3 1
# swift-ring-builder container.builder create 18 3 1
# swift-ring-builder object.builder create 18 3 1
```

6- Para cada dispositivo de armazenamento, em cada nó, adicione entradas para cada anel:

```
# swift-ring-builder account.builder add
zZONE-STORAGE_LOCAL_NET_IP:6002[RSTORAGE_REPLICATION_NET_IP:6005]/DEVICE
100
# swift-ring-builder container.builder add
zZONE-STORAGE_LOCAL_NET_IP_1:6001[RSTORAGE_REPLICATION_NET_IP:6004]/DEVICE
100
# swift-ring-builder object.builder add
zZONE-STORAGE_LOCAL_NET_IP_1:6000[RSTORAGE_REPLICATION_NET_IP:6003]/DEVICE
100
```

OBS: Omita o parâmetro opcional STORAGE_REPLICATION_NET_IP, caso não esteja usando uma rede dedicada para replicação.

Por exemplo, se um nó de armazenamento tem uma partição na Zona 1, com IP 10.0.0.1, o nó de armazenamento tem endereço 10.0.1.1 para a rede de replicação. O ponto de montagem esta partição é /srv/node/sdb1 , e o caminho no arquivo de configuração /etc/rsyncd.conf é /srv/node/ , o dispositivo seria **sdb1**, e os comandos seriam:

```
# swift-ring-builder account.builder add z1-10.0.0.1:6002R10.0.1.1:6005/sdb1 100
```

CITTA: `swift-ring-builder account.builder add z1-10.0.0.18:6002/vda2 100`

```
# swift-ring-builder container.builder add z1-10.0.0.1:6001R10.0.1.1:6005/sdb1 100
# swift-ring-builder object.builder add z1-10.0.0.1:6000R10.0.1.1:6005/sdb1 100
```

OBS: Caso sejam assumidas cinco zonas, com um nó para cada zona, comece pela **ZONE 1**. Para cada nó adicional, incremente **ZONE** em 1.

7- Verifique o conteúdo do anel de cada anel:

```
# swift-ring-builder account.builder
# swift-ring-builder container.builder
# swift-ring-builder object.builder
```

8- Rebalanceie os anéis:

```
# swift-ring-builder account.builder rebalance
# swift-ring-builder container.builder rebalance
# swift-ring-builder object.builder rebalance
```

OBS: Isso pode levar algum tempo.

9- Copie os arquivos **account.ring.gz**, **container.ring.gz**, e **object.ring.gz** para cada nó Proxy e de Armazenamento em /etc/swift .

10- Certifique-se que o usuário *swift* é o dono de todos os arquivos de configuração:

```
# chown -R swift:swift /etc/swift
```

11- Reinicie o serviço de Proxy:

```
# service swift-proxy restart
```

8.8- Inicie serviços nos nós de Armazenamento

Agora que os arquivos de anel estão em cada nó de armazenamento, você pode iniciar os serviços. Para cada nó de armazenamento, use o comando:

```
# for service in \
  swift-object swift-object-replicator swift-object-updater swift-object-auditor \
  swift-container swift-container-replicator swift-container-updater
swift-container-auditor \
  swift-account swift-account-replicator swift-account-reaper swift-account-auditor; do
\
      service $service start; done
```

OBS: Para iniciar todos os serviços do **swift** ao mesmo tempo, use:

```
# swift-init all start
```

OBS2: Mais detalhes sobre este comando podem ser achados no manual:

```
# man swift-init
```

8.9- Verifique a instalação

Estes comandos podem ser rodados no servidor Proxy, ou qualquer máquina que tenha acesso ao Serviço de Identidade.

1- Confira se as credenciais estão definidas corretamente no arquivo **admin-openrc.sh** e dê um *source* nele:

```
$ source admin-openrc.sh
```

2- Rode o comando **swift** a seguir:

```
$ swift stat
Account: AUTH_11b9758b7049476d9b48f7a91ea11493
Containers: 0
  Objects: 0
    Bytes: 0
Content-Type: text/plain; charset=utf-8
X-Timestamp: 1381434243.83760
X-Trans-Id: txdcdd594565214fb4a2d33-0052570383
X-Put-Timestamp: 1381434243.83760
```

3- Rode os comandos **swift** a seguir para fazer *upload* de arquivos para um contâiner. Crie os arquivos **test.txt** e **test2.txt** localmente se necessário.

```
$ swift upload myfiles test.txt
$ swift upload myfiles test2.txt
```

4- Rode o comando **swift** a seguir para fazer *download* de todos os arquivos no contâiner **myfiles**:

```
$ swift download myfiles
test2.txt [headers 0.267s, total 0.267s, 0.000s MB/s]
test.txt [headers 0.271s, total 0.271s, 0.000s MB/s]
```

8.10- Adicione outro servidor proxy

Para obter (ou aumentar) confiabilidade, você adiciona servidores proxy, assim como fez com o primeiro, salvo por algumas configurações adicionais:

Se existirem mais que dois servidores proxy, você precisa fazer o balanceamento entre eles; seu *endpoint* para armazenamento (que os clientes usam para se conectar ao armazenamento) também muda. Podem ser usadas várias estratégias para fazer o balanceamento. Por exemplo, você pode deixar um **DNS** em *round-robin*, ou um escalonador de carga por *software* ou *hardware*, como o **pound**, na frente dos dois servidores proxy, e colocar o endereço URL do balanceador de carga para o armazenamento.

Configure o proxy inicial. Então, complete estas etapas para adicionar os proxies adicionais:

1- Atualize a lista dos servidores **memcache** no arquivo de configuração /etc/swift/proxy-server.conf para os servidores *proxy* adicionais. Se você roda múltiplos

servidores memcache, use este padrão para as mútiplas listagens IP:porta em cada arquivo de configuração do servidor proxy:

10.1.2.3:11211,10.1.2.4:11211

[filter:cache]
use = egg:swift#memcache
memcache_servers = *PROXY_LOCAL_NET_IP*:11211

2- Copie a informação do anel para todos os nós, incluindo os nós **proxy** novos. Certifique-se também que a informação do anel chegue em todos os nós de armazenamento.

3- Após sincronizar todos os nós, certifique-se que o administrador tem as chaves em /etc/swift e que o dono do arquivo do anel está correto.

8.11- Próximos passos

Seu ambiente OpenStack agora inclui o Armazenamento de Objetos. Você pode iniciar uma instância ou adicionar mais serviços ao ambiente, mencionados nas seções a seguir.

9- Adicionando o Serviço de Orquestramento - HEAT

Use o módulo de Orquestramento para criar recursos de nuvem usando uma linguagem de modelos chamada **HOT**. O nome do projeto integrado é **Heat**.

9.1- Visão geral do serviço de Orquestramento

O serviço de Orquestramento fornece um orquestramento baseado em modelos para descrever uma aplicação da nuvem ao usar chamadas de API do OpenStack para gerar aplicações de nuvem em execução. O *software* se integra com outros componentes centrais do OpenStack em um sistema de um arquivo de modelo. Os modelos permitem que você crie a maioria dos tipos de recurso do OpenStack, como instâncias, IPs flutuantes, volumes, grupos de segurança, usuário, entre outros. Além disso, providencia mais funcionalidades avançadas, como alta disponibilidade de instância, auto escalonamento de instância, e pilhas aninhadas. Ao fornecer uma integração bem próxima com outros projetos centrais do OpenStack, todos os projetos do OpenStack poderiam ter uma base de usuários maior.
O serviço permite implantadores integrarem com o serviço de Orquestramento diretamente ou pelo uso de *plugins* personalizados.
O serviço de Orquestramento consiste dos seguintes componentes:

- Cliente por linha de comando **heat**. Uma **interface por linha de comando** (*command line interface* - CLI) que se comunica com o **heat-api** para rodar APIs de **CloudFormation** do AWS (***Amazon Web Services***). Desenvolvedores finais podem usar também a API em REST de Orquestramento diretamente.
- Componente **heat-api**. Providencia uma API em REST nativa para OpenStack que processa requisições de API ao enviá-las para o **heat-engine** por RPC.

- Componente **heat-api-cfn**. Providencia uma API de consulta para AWS que é compatível com o **AWS CloudFormation** e processa requisições de API ao enviá-las para o **heat-engine** por RPC.
- **heat-engine**. Orquestra o lançamento de modelos e providencia eventos de volta para o consumidor de API.

9.2 - Instalar o serviço de Orquestramento

1- Instale o módulo de Orquestramento no nó **controller**:

```
# apt-get install heat-api heat-api-cfn heat-engine
```

2- No arquivo de configuração, especifique a localização do Banco de Dados onde o serviço de Orquestramento guarda os dados. Estes exemplos usam o BD do MySQL com um usuário *heat* no nó **controller**. Substitua HEAT_DBPASS pela senha criada para o usuário do BD:
Edite o arquivo /etc/heat/heat.conf na seção [database]:

```
[database]
# The SQLAlchemy connection string used to connect to the database
sql_connection = mysql://heat:HEAT_DBPASS@controller/heat
...
```

3- Por padrão, os pacotes do Ubuntu criam um BD SQLite. Remova (ou renomeie) o arquivo **heat.sqlite** que foi criado no diretório /var/lib/heat/ para que ele não seja usado por engano.

4- Use a senha que você definiu anteriormente para fazer *login* como *root* e crie um usuário *heat* do BD:

```
# mysql -u root -p
mysql> CREATE DATABASE heat;
mysql> GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'localhost' \
IDENTIFIED BY 'HEAT_DBPASS';
mysql> GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'%' \
IDENTIFIED BY 'HEAT_DBPASS';
```

5- Crie as tabelas de serviço *heat*:

```
# su -s /bin/sh -c "heat-manage db_sync" heat
```

OBS: Ignore erros do tipo **DeprecationWarning**.

6- Os pacotes do Ubuntu não configuram a gravação de **log** corretamente. Edite o arquivo /etc/heat/heat.conf e mude a seção [DEFAULT]:

```
[DEFAULT]
...
# Print more verbose output (set logging level to INFO instead
# of default WARNING level). (boolean value)
verbose = True
```

...
# (Optional) The base directory used for relative --log-file
# paths (string value)
log_dir=/var/log/heat

7- Configure o Serviço de Orquestramento para usar o RabbitMQ como o *broker* de mensagens. Edite o arquivo /etc/heat/heat.conf na seção [DEFAULT]:
rabbit_host = controller
rabbit_password = RABBIT_PASS

8- Crie um usuário *heat* para que o serviço de Orquestramento possa se autenticar com o Serviço de Identidade. Use o alocador **service** e dê ao usuário a função de *admin*:

```
# keystone user-create --name=heat --pass=HEAT_PASS
--email=heat@example.com
# keystone user-role-add --user=heat --tenant=service --role=admin
```

9- Edite o arquivo /etc/heat/heat.conf para mudar as seções [keystone_authtoken] e [ec2authtoken] para adicionar credenciais para o Serviço de Orquestramento:
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000/v2.0
admin_tenant_name = service
admin_user = heat
admin_password = HEAT_PASS

[ec2authtoken]
auth_uri = http://controller:5000/v2.0

OBS: Estas seções normalmente não existem, e precisam ser criadas.

10- Registre as APIs **Heat** e **CloudFormation** com o Serviço de Identidade para que outros serviços do OpenStack possam localizar estas APIs. Registre o serviço e especifique um *endpoint*:

```
$ keystone service-create --name=heat --type=orchestration \
  --description="Orchestration"

$ keystone endpoint-create \
  --service-id=$(keystone service-list | awk '/ orchestration / {print
$2}') \
  --publicurl=http://controller:8004/v1/%\(tenant_id\)s \
```

```
  --internalurl=http://controller:8004/v1/%\(tenant_id\)s \
  --adminurl=http://controller:8004/v1/%\(tenant_id\)s

$ keystone service-create --name=heat-cfn --type=cloudformation \
  --description="Orchestration CloudFormation"

$ keystone endpoint-create \
  --service-id=$(keystone service-list | awk '/ cloudformation / {print $2}') \
  --publicurl=http://controller:8000/v1 \
  --internalurl=http://controller:8000/v1 \
  --adminurl=http://controller:8000/v1
```

11- Reinicie o serviço com as novas configurações:

```
# service heat-api restart
# service heat-api-cfn restart
# service heat-engine restart
```

9.3 - Verifique a instalação do Serviço de Orquestramento

Para verificar que o serviço de Orquestramento está instalado e configurado corretamente, certifique-se que suas credenciais estão definidas corretamente no arquivo openrc.sh. Faça um *source* no arquivo, como a seguir:

```
$ source demo-openrc.sh
```

O módulo de Orquestramento usa *templates* para descrever pilhas. Para conhecer as linguagens de *template*, veja o Guia de Templates na documentação de desenvolvedor do Heat. Crie um *template* de teste no arquivo **test-stack.yml** com o conteúdo a seguir:

```
heat_template_version: 2013-05-23

description: Test Template

parameters:
  ImageID:
    type: string
    description: Image use to boot a server
  NetID:
    type: string
    description: Network ID for the server

resources:
  server1:
    type: OS::Nova::Server
    properties:
      name: "Test server"
```

```
    image: { get_param: ImageID }
    flavor: "m1.tiny"
    networks:
    - network: { get_param: NetID }

outputs:
  server1_private_ip:
    description: IP address of the server in the private network
    value: { get_attr: [ server1, first_address ] }
```

Use o comando **heat stack-create** para criar uma pilha a partir deste template:

```
$ NET_ID=$(nova net-list | awk '/ demo-net / { print $2 }')
$ heat stack-create -f test-stack.yml \
  -P "ImageID=cirros-0.3.2-x86_64;NetID=$NET_ID" testStack
```

```
+--------------------------------------+------------+--------------------+----------------------+
| id                                   | stack_name | stack_status       | creation_time        |
+--------------------------------------+------------+--------------------+----------------------+
| 477d96b4-d547-4069-938d-32ee990834af | testStack  | CREATE_IN_PROGRESS | 2014-04-06T15:11:01Z |
+--------------------------------------+------------+--------------------+----------------------+
```

Verifique que a pilha foi criada com sucesso com o comando **heat stack-list**:

```
$ heat stack-list
```

```
+--------------------------------------+------------+-----------------+----------------------+
| id                                   | stack_name | stack_status    | creation_time        |
+--------------------------------------+------------+-----------------+----------------------+
| 477d96b4-d547-4069-938d-32ee990834af | testStack  | CREATE_COMPLETE | 2014-04-06T15:11:01Z |
+--------------------------------------+------------+-----------------+----------------------+
```

9.4 - Próximos Passos

Seu ambiente OpenStack agora inclui a Orquestração. Você pode iniciar uma instância ou adicionar mais serviços a seu ambiente, descritos nas próximas seções.

10- Adicione o módulo de Telemetria - CEILOMETER

10.1- Telemetria

O módulo de Telemetria é capaz de:

- Coleta eficientemente dados de métrica sobre custos de CPU e rede.
- Coleta dados ao monitorar notificações enviadas pelos serviços ou por realizar *polling* na infraestrutura.
- Configura o tipo de dados coletados para atender a vários requisitos de operação. Acessar e inserir dados de métricas com a API em REST.
- Expande o arcabouço para coletar dados de uso personalizados por plugins adicionais.
- Produz mensagens de métrica assinados que não podem ser repudiados.

O sistema consiste dos seguintes componentes básicos:

- Um agente de **compute** (**ceilometer-agent-compute**). Roda em cada nó de **compute** e realiza coletas de estatísticas de utilização de recursos. Podem haver outros tipos de agentes no futuro, mas por ora o foco está em criar o agente de **compute**.
- Um agente central (**ceilometer-agent-central**). Roda em um servidor de gerenciamento central para realizar *polls* de estatísticas de utilização de recursos para recursos não associados a instâncias ou nós de **compute**.
- Um coletor (**ceilometer-collector**). Roda em um ou mais servidores de gerenciamento centrais para monitorar as filas de mensagens (para notificações e para dados de métricas vindos do agente). Mensagens de notificação são processadas e transformadas em mensagens de métricas e enviadas de volta para o canal de mensagens usando o tópico apropriado. Mensagens de telemetria são escritas para o armazenamento de dados sem modificações.
- Um notificador de alarme (**ceilometer-alarm-notifier**). Roda em um ou mais servidores de gerenciamento centrais para permitir definições de alarmes baseado na avaliação de limiares para uma coleção de amostras.
- Um armazenamento de dados. Um Banco de Dados capaz de controlar escritas (de uma ou mais instâncias coletoras) e leituras concorrentes (do servidor de API).
- Um servidor de API (**ceilometer-api**). Roda em um ou mais servidores de gerenciamento para prover acesso a dados a partir do armazenamento de dados.

Estes serviços se comunicam usando o canal de mensagens padrão do OpenStack. Apenas o servidor de coleta e de API têm acesso ao armazenamento de dados.

10.2- Instale o módulo de Telemetria

Telemetria fornece um serviço de API que providencia um **coletor** e um conjunto de agentes disparates. Antes de instalar estes agentes nos nós, como o nó de **compute**, você deve usar este procedimento para instalar os componentes centrais no nó **controller**.

1- Instale o Serviço de Telemetria no nó **controller**:

```
# apt-get install ceilometer-api ceilometer-collector ceilometer-agent-central \
  ceilometer-agent-notification ceilometer-alarm-evaluator ceilometer-alarm-notifier
python-ceilometerclient
```

2- O Serviço de Telemetria usa um Banco de Dados para armazenar informação. Especifique a localização do Banco de Dados no arquivo de configuração. Os exemplos usam o Banco de Dados **MongoDB** no nó **controller**:

```
# apt-get install mongodb-server
```

OBS: Por padrão, o MongoDB é configurado para criar vários arquivos de 1 GB no diretório /var/lib/mongodb/journal/ para suportar o *database journaling*.

Se você precisa minimizar o espaço alocado para *database journaling*, então defina a chave de configuração *smallfiles* para **true** no arquivo de configuração /etc/mongodb.conf. Esta configuração reduz o tamanho de cada arquivo de journaling para 512 MB.

Como os arquivos são criados na primeira vez que o serviço do MongoDB inicia, você deve parar o serviço e remover os arquivos para esta mudança ter efeito:

```
# service mongodb stop
# rm /var/lib/mongodb/journal/prealloc.*
# service mongodb start
```

Para mais informação da chave de configuração *smallfiles*, veja a documentação do MongoDB. Para instruções detalhando os passos de como desligar o database journaling completamente, veja o link http://docs.mongodb.org/manual/tutorial/manage-journaling/.

3- Configure o MongoDB para ouvir no endereço de IP público do **controller**. Edite o arquivo /etc/mongodb.conf e modifique a chave **bind_ip**:

bind_ip = 10.0.0.11

OBS: No nosso caso, bind_ip=150.165.85.226

4- Reinicie o serviço MongoDB para aplicar a mudança de configuração:

```
# service mongodb restart
```

5- Crie um Banco de Dados e um usuário *ceilometer* do BD:

```
# mongo --host controller --eval '
db = db.getSiblingDB("ceilometer");
db.addUser({user: "ceilometer",
        pwd: "CEILOMETER_DBPASS",
        roles: [ "readWrite", "dbAdmin" ]})'
```

6- Configure o Serviço de Telemetria para usar o Banco de Dados. Edite o arquivo /etc/ceilometer/ceilometer.conf e mude a seção [database]:

```
[database]
# The SQLAlchemy connection string used to connect to the
# database (string value)
connection = mongodb://ceilometer:CEILOMETER_DBPASS@controller:27017/ceilometer
```

7- Você precisa definir uma chave secreta que será compartilhada entre os nós do Serviço de Telemetria. Use o **openssl** para gerar uma ficha aleatória e armazenar no arquivo de configuração:

```
# openssl rand -hex 10
```

Edite o arquivo /etc/ceilometer/ceilometer.conf e mude a seção [publisher]. Substitua *CEILOMETER_TOKEN* com os resultados do comando openssl:

```
[publisher]
# Secret value for signing metering messages (string value)
```

metering_secret = CEILOMETER_TOKEN
...

8- Configure o acesso ao RabbitMQ. Edite o arquivo /etc/ceilometer/ceilometer.conf e atualize a seção [DEFAULT]:
rabbit_host = controller
rabbit_password = RABBIT_PASS

9- Configure o diretório para **logs**. Edite o arquivo /etc/ceilometer/ceilometer.conf e atualize a seção [DEFAULT]:
[DEFAULT]
log_dir = /var/log/ceilometer

10- Crie um usuário *ceilometer* que o Serviço de Telemetria usa para se autenticar com o Serviço de Identidade. Use o alocador **service** e dê ao usuário a função *admin*:
```
# keystone user-create --name=ceilometer --pass=CEILOMETER_PASS
--email=ceilometer@example.com
# keystone user-role-add --user=ceilometer --tenant=service --role=admin
```

11- Configure o serviço de Telemetria para se autenticar com o Serviço de Identidade. Defina o valor em *auth_strategy* para **keystone** no arquivo /etc/ceilometer/ceilometer.conf.
[DEFAULT]
...
auth_strategy = keystone

12- Adicione as credenciais aos arquivos de configuração para o Serviço de Telemetria. Edite o arquivo /etc/ceilometer/ceilometer.conf e mude a seção [keystone_authtoken]:
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000
admin_tenant_name = service
admin_user = ceilometer
admin_password = CEILOMETER_PASS

Também defina a seção [service_credentials]:
[service_credentials]
os_auth_url = http://controller:5000/v2.0
os_username = ceilometer
os_tenant_name = service
os_password = CEILOMETER_PASS

13- Registre o Serviço de Telemetria com o Serviço de Identidade para que outros serviços do OpenStack possam localizá-lo. Use o comando **keystone** para registrar o serviço e especificar o *endpoint*:

```
$ keystone service-create --name=ceilometer --type=metering \
  --description="Ceilometer Telemetry Service"
$ keystone endpoint-create \
  --service-id=$(keystone service-list | awk '/ metering / {print $2}') \
  --publicurl=http://controller:8777 \
  --internalurl=http://controller:8777 \
  --adminurl=http://controller:8777
```

14- Reinicie os serviços com suas novas configurações:

```
# service ceilometer-agent-central restart
# service ceilometer-agent-notification restart
# service ceilometer-api restart
# service ceilometer-collector restart
# service ceilometer-alarm-evaluator restart
# service ceilometer-alarm-notifier restart
```

10.3- Instalar o agente de *Compute* para Telemetria

Telemetria fornece um serviço de API que providencia um coletor e um conjunto de agentes disparates. Este procedimento detalha como instalar o agente que roda no nó **compute**.

1- Instale o Serviço de Telemetria no nó **Compute**:

```
# apt-get install ceilometer-agent-compute
```

2- Edite o arquivo /etc/nova/nova.conf e adicione as seguintes linhas à seção [DEFAULT]:

```
[DEFAULT]
...
instance_usage_audit = True
instance_usage_audit_period = hour
notify_on_state_change = vm_and_task_state
notification_driver = nova.openstack.common.notifier.rpc_notifier
notification_driver = ceilometer.compute.nova_notifier
```

3- Reinicie o serviço de **Compute**:

```
# service nova-compute restart
```

4- Você deve atribuir a chave secreta que você definiu previamente. Os nós do serviço de Telemetria compartilham esta chave secretamente. Edite o arquivo /etc/ceilometer/ceilometer.conf e mude estas linhas na seção [publisher]. Substitua *CEILOMETER_TOKEN* pela ficha de *ceilometer* que você criou previamente:

```
...
```

[publisher]
# Secret value for signing metering messages (string value)
metering_secret = CEILOMETER_TOKEN

5- Configure o acesso ao RabbitMQ. Edite o arquivo /etc/ceilometer/ceilometer.conf e atualize a seção [DEFAULT]:
[DEFAULT]
rabbit_host = controller
rabbit_password = RABBIT_PASS

6- Adicione as credenciais do serviço de Identidade. Edite o arquivo /etc/ceilometer/ceilometer.conf e mude a seção [keystone_authtoken]:
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = ceilometer
admin_password = CEILOMETER_PASS

Também defina a seção [service_credentials]:
[service_credentials]
os_auth_url = http://controller:5000/v2.0
os_username = ceilometer
os_tenant_name = service
os_password = CEILOMETER_PASS

6- Configure o diretório de **logs**. Edite o arquivo /etc/ceilometer/ceilometer.conf e atualize a seção [DEFAULT]:
[DEFAULT]
log_dir = /var/log/ceilometer

7- Reinicie o serviço com suas novas configurações:
```
# service ceilometer-agent-compute restart
```

10.4- Configure o Serviço de Imagem para Telemetria

1- Para obter amostras de imagem, você deve configurar o Serviço de Imagem para enviar notificações para o canal. Edite o arquivo /etc/glance/glance-api.conf e modifique a seção [DEFAULT]:
notification_driver = messaging
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS

2- Reinicie os Serviços de Imagem com suas novas configurações:

```
# service glance-registry restart
# service glance-api restart
```

10.5- Adicione o agente do Serviço de Armazenamento em Bloco para Telemetria

1- Para obter amostras de volumes, você deve configurar o Serviço de Armazenamento em Bloco para enviar notificações pelo canal. Edite o arquivo /etc/cinder/cinder.conf e adicione na seção [DEFAULT]:

control_exchange = cinder
notification_driver = cinder.openstack.common.notifier.rpc_notifier

2- Reinicie os Serviços de Armazenamento em Bloco com suas novas configurações:

No nó **controller**:

```
# service cinder-api restart
# service cinder-scheduler restart
```

No nó **compute**:

```
# service cinder-volume restart
```

10.6- Adicione o agente do Armazenamento de Objetos para o serviço de Telemetria

1- Para obter estatísticas do armazenamento de objetos, o serviço de Telemetria precisa ter acesso ao Armazenamento de Objetos com a função **ResellerAdmin**. Dê esta função para o seu usuário **os_username** para o alocador **os_tenant_name**:

```
$ keystone role-create --name=ResellerAdmin
```

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 462fa46c13fd4798a95a3bfbe27b5e54 |
|   name   |           ResellerAdmin          |
+----------+----------------------------------+
```

```
$ keystone user-role-add --tenant service --user ceilometer \
      --role 462fa46c13fd4798a95a3bfbe27b5e54
```

2- Você também deve adicionar o *middleware* de Telemetria ao Armazenamento de Objetos para controlar o tráfego de entrada e saída. Adicione estas linhas ao arquivo /etc/swift/proxy-server.conf :

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

3- Adicione o **ceilometer** para o parâmetro de ***pipeline*** no mesmo arquivo:

[pipeline:main]
pipeline = healthcheck cache authtoken keystoneauth ceilometer proxy-server

4- Reinicie o serviço com suas novas configurações:
```
# service swift-proxy restart
```

10.7- Verifique a instalação da Telemetria

Para testar a instalação da Telemetria, baixe uma imagem do Serviço de Imagem, e use o comando **ceilometer** para mostrar as estatísticas de uso.

1- Use o comando **ceilometer meter-list** para testar o acesso à Telemetria:
```
$ ceilometer meter-list
```
```
+------------+-------+-------+--------------------------------------+---------+----------------------------------+
| Name       | Type  | Unit  | Resource ID                          | User ID | Project ID                       |
+------------+-------+-------+--------------------------------------+---------+----------------------------------+
| image      | gauge | image | 9e5c2bee-0373-414c-b4af-b91b0246ad3b | None    | e66d97ac1b704897853412fc8450f7b9 |
| image.size | gauge | B     | 9e5c2bee-0373-414c-b4af-b91b0246ad3b | None    | e66d97ac1b704897853412fc8450f7b9 |
+------------+-------+-------+--------------------------------------+---------+----------------------------------+
```

2- Baixe uma imagem do Serviço de Imagem:
```
$ glance image-download "cirros-0.3.2-x86_64" > cirros.img
```

OBS: Atentar ao nome da imagem que está sendo usada.

3- Chame o comando **ceilometer meter-list** novamente para validar que o *download* foi detectado e armazenado na Telemetria:
```
$ ceilometer meter-list
```
```
+----------------+-------+-------+--------------------------------------+---------+----------------------------------+
| Name           | Type  | Unit  | Resource ID                          | User ID | Project ID                       |
+----------------+-------+-------+--------------------------------------+---------+----------------------------------+
| image          | gauge | image | 9e5c2bee-0373-414c-b4af-b91b0246ad3b | None    | e66d97ac1b704897853412fc8450f7b9 |
| image.download | delta | B     | 9e5c2bee-0373-414c-b4af-b91b0246ad3b | None    | e66d97ac1b704897853412fc8450f7b9 |
| image.serve    | delta | B     | 9e5c2bee-0373-414c-b4af-b91b0246ad3b | None    | e66d97ac1b704897853412fc8450f7b9 |
| image.size     | gauge | B     | 9e5c2bee-0373-414c-b4af-b91b0246ad3b | None    | e66d97ac1b704897853412fc8450f7b9 |
+----------------+-------+-------+--------------------------------------+---------+----------------------------------+
```

4- Agora você pode obter estatísticas de uso para várias métricas:
```
$ ceilometer statistics -m image.download -p 60
```
```
+--------+---------------------+---------------------+-------+-----------+-----------+-----------+-----------+----------+---------------------+---------------------+
| Period | Period Start        | Period End          | Count | Min       | Max       | Sum       | Avg       | Duration | Duration Start      | Duration End        |
+--------+---------------------+---------------------+-------+-----------+-----------+-----------+-----------+----------+---------------------+---------------------+
| 60     | 2013-11-18T18:08:50 | 2013-11-18T18:09:50 | 1     | 13147648.0 | 13147648.0 | 13147648.0 | 13147648.0 | 0.0      | 2013-11-18T18:09:05.334000 | 2013-11-18T18:09:05.334000 |
+--------+---------------------+---------------------+-------+-----------+-----------+-----------+-----------+----------+---------------------+---------------------+
```

10.8- Próximos passos

Seu ambiente OpenStack agora inclui Telemetria. Você pode iniciar uma instância ou adicionar mais serviços ao ambiente das seções anteriores.

11 - Adicionar o Serviço de Banco de Dados - TROVE

OBS: Esta seção está sendo atualizada constantemente, por conta do guia oficial estar igualmente incompleto.

11.1 - Visão geral do Serviço de Banco de Dados

O serviço de Banco de Dados provê uma funcionalidade escalável e confiável de provisionamento de nuvem tanto para os mecanismos relacionais quanto os não-relacionais. Usuários podem usar funcionalidades do Banco de Dados fácil e rapidamente sem o problema de lidar com tarefas administrativas compelxas. Usuários da nuvem e administradores do BD podem provisionar e gerenciar múltiplas instâncias de BDs conforme necessário.
O serviço de Banco de Dados fornece o isolamento de recursos em altos níveis de desempenho, e automatiza tarefas administrativas complexas, como implantação, configuração, *patching*, *backups*, restauração, e monitoramento.

Exemplo de fluxo de processo. Aqui está um fluxo de processo de alto nível para usar os serviços de Banco de Dados:
1 - O administrador define a instraestrutura:

a) O administrador do OpenStack instala o serviço de Banco de Dados.
b) Ele(a) cria uma imagem para cada tipo de Banco de Dados que o administrador deseja possuir (um para o MySQL, um para o MongoDB, e assim vai).
c) O administrador do OpenStack atualizar o armazenamento de dados para usar as novas imagens, usando o comando **trove-manage**.

2- Usuários finais usam o serviço de Banco de Dados:

a) Agora que a infraestrutura básica está definida, um usuário final pode criar uma instância **Trove** (banco de dados) quando o usuário desejar, usando o comando **trove create**.
b) O usuário final pega um endereço IP da instância do **Trove** ao usar o comando **trove list** para pegar o ID da instância, e então usando o comando **trove show instanceID** para pegar o endereço IP.
c) O usuário final agora pode acessar a instância Trove usando comandos típicos de acesso a BD. Por exemplo, para o MySQL:
$ mysql -u myuser -pmypass -h *trove_ip_address* mydb

Componentes: O serviço de Banco de Dados inclui os seguintes componentes:

- cliente por linha de comando **python-troveclient**. Uma CLI que se comunica com o componente **trove-api**.
- componente **trove-api**. Providencia uma API em REST nativa para o OpenStack que suporta JSON para o provisionamento e gerencia instâncias **Trove**.
- serviço **trove-conductor**. Roda no *host*, e recebe mensagens das instâncias visitantes que desejam atualizar a informação no *host*.
- serviço **trove-taskmanager**. Instrumenta os complexos fluxos de sistema que suportam o provisionamento de instâncias, gerenciam o ciclo de vida das instâncias, e realizam operações nas instâncias.
- serviço **trove-guestagent**. Roda por dentro da instância visitante. Gerencia e realiza operações no próprio Banco de Dados.

11.2 - Instalando o serviço de Banco de Dados

Este procedimento instala o módulo de Banco de Dados no nó **controller**.

Pré-requisitos

Esta seção assume que já existe um ambiente OpenStack funcionando com pelo menos os seguintes componentes instalados: **Compute**, Serviço de **Imagem**, e **Identidade**.

OBS: Apenas no Ubuntu **14.04** - O módulo de Banco de Dados está disponível apenas no Ubuntu 14.04. Os pacotes não estão disponíveis para o 12.04, ou pelo **Ubuntu Cloud Archive**.

Instalando o módulo de Banco de Dados no controller

1- Instale os pacotes necessários:

```
# apt-get install python-trove python-troveclient python-glanceclient \
  trove-common trove-api trove-taskmanager
```

2- Prepare o OpenStack:

a) Faça um source no arquivo **admin-openrc.sh**.

```
$ source ~/admin-openrc.sh
```

b) Crie um usuário *trove* que o **Compute** usa para autenticar no Serviço de Identidade. Use o alocador *service* e dê ao usuário a função *admin*:

```
$ keystone user-create --name=trove --pass=TROVE_PASS \
  --email=trove@example.com
$ keystone user-role-add --user=trove --tenant=service --role=admin
```

3- Edite os seguintes arquivos de configuração, realizando as seguintes ações para cada arquivo:

- trove.conf
- trove-taskmanager.conf
- trove-conductor.conf

a) Edite a seção [DEFAULT] de cada arquivo e defina os valores apropriados das URLs, configuração de logs e mensagens (possivelmente de erro e avisos), e conexões SQL:

```
[DEFAULT]
log_dir = /var/log/trove
trove_auth_url = http://controller:5000/v2.0
nova_compute_url = http://controller:8774/v2
cinder_url = http://controller:8776/v1
swift_url = http://controller:8080/v1/AUTH_
sql_connection = mysql://trove:TROVE_DBPASS@controller/trove
notifier_queue_hostname = controller
```

b) Configure o módulo de Banco de Dados para usar o *broker* de mensagens RabbitMQ ao configurar **rabbit_password** no grupo de configuração [DEFAULT] de cada arquivo:

[DEFAULT]
...
rabbit_password = RABBIT_PASS
...

4- Edite a seção [filter:authtoken] do arquivo **api-paste.ini** para que seja igual a como está abaixo:
[filter:authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_user = trove
admin_password = ADMIN_PASS
admin_token = ADMIN_TOKEN
admin_tenant_name = service
signing_dir = /var/cache/trove

5- Edite o arquivo **trove.conf** para que ele inclua os valores apropriados para o armazenamento de dados padrão e as regras de rede, como mostrado abaixo:
[DEFAULT]
default_datastore = mysql
....
# Config option for showing the IP address that nova doles out
add_addresses = True
network_label_regex = ^NETWORK_LABEL$
....

6- Edite o arquivo **trove-taskmanager.conf** para que ele inclua as credenciais de serviço apropriadas para conectar ao serviço de **Compute** do OpenStack, conforme mostrado abaixo:
[DEFAULT]
....
# Configuration options for talking to nova via the novaclient.
# These options are for an admin user in your keystone config.
# It proxy's the token received from the user to send to nova via this admin users creds,
# basically acting like the client via that proxy token.
nova_proxy_admin_user = admin
nova_proxy_admin_pass = ADMIN_PASS
nova_proxy_admin_tenant_name = service
...

7- Prepare o Banco de Dados de administrador **trove**:
```
$ mysql -u root -p
```
mysql> CREATE DATABASE trove;

mysql> GRANT ALL PRIVILEGES ON trove.* TO trove@'localhost' IDENTIFIED BY 'TROVE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON trove.* TO trove@'%' IDENTIFIED BY 'TROVE_DBPASS';

8- Prepare o serviço de Banco de Dados:
a) Inicialize o Banco de Dados:

```
# su -s /bin/sh -c "trove-manage db_sync" trove
```

b) Crie o armazenamento de dados. Você precisa criar um armazenamento de dados separado para cada tipo de Banco de Dados que você deseja usar, por exemplo, MySQL, MongoDB, Cassandra. Este exemplo mostra como criar um armazenamento de dados para um Banco de Dados MySQL:

```
# su -s /bin/sh -c "trove-manage datastore_update mysql '' " trove
```

9- Crie uma imagem **trove**.
Crie uma imagem para o tipo de Banco de Dados que você deseja usar, por exemplo, MySQL, MongoDB, Cassandra.
Esta imagem deve possuir um agente visitante do **trove** instalado, e deve possuir o arquivo **trove-guestagent.conf** configurado para conectar ao seu ambiente OpenStack. Para configurar corretamente o arquivo trove-guestagent.conf, siga estes passos na instância visitante que você está usando para construir sua imagem:
a) Adicione as seguintes linhas ao arquivo **trove-guestagent.conf**:
rabbit_host = controller
rabbit_password = RABBIT_PASS
nova_proxy_admin_user = admin
nova_proxy_admin_pass = ADMIN_PASS
nova_proxy_admin_tenant_name = service
trove_auth_url = http://controller:35357/v2.0

10- Atualize o armazenamento de dados para usar a nova imagem, usando o comando **trove-manage**.
Este exemplo mostra como criar um armazenamento de dados MySQL 5.5:

```
# trove-manage --config-file=/etc/trove/trove.conf datastore_version_update
\
  mysql mysql-5.5 mysql glance_image_ID mysql-server-5.5 1
```

11- Você deve registrar o módulo do Banco de Dados com o Serviço de Identidade para que os outros serviços do OpenStack possam localizá-lo. Registre o serviço e especifique o *endpoint*:

```
$ keystone service-create --name=trove --type=database \
  --description="OpenStack Database Service"
$ keystone endpoint-create \
  --service-id=$(keystone service-list | awk '/ trove / {print $2}') \
```

```
  --publicurl=http://controller:8779/v1.0/%\(tenant_id\)s \
  --internalurl=http://controller:8779/v1.0/%\(tenant_id\)s \
  --adminurl=http://controller:8779/v1.0/%\(tenant_id\)s
```

12- Reinicie os serviços de Banco de Dados:

```
# service trove-api restart
# service trove-taskmanager restart
# service trove-conductor restart
```

11.3 - Verifique a instalação do serviço de Banco de Dados

Para verificar que o serviço de Banco de Dados está instalado e configurado corretamente, tente executar um comando Trove:

1- Faça um *source* no arquivo **demo-openrc.sh**.

```
$ source ~/demo-openrc.sh
```

2- Recupere a lista de instâncias do **Trove**:

```
$ trove list
```

Deve ser obtida uma saída deste tipo:

```
+----+------+-----------+-------------------+--------+-----------+------+

| id | name | datastore | datastore_version | status | flavor_id | size |
+----+------+-----------+-------------------+--------+-----------+------+
+----+------+-----------+-------------------+--------+-----------+------+
```

3- Assumindo que você criou uma imagem para o tipo de Banco de Dados que você deseja, e atualizou o armazenamento de dados para usar esta imagem, você agora pode criar uma instância **Trove** (BD). Para fazer isso, use o comando **trove create**.

Este exemplo mostra como criar um Banco de Dados MySQL 5.5:

```
$ trove create name 2 --size=2 --databases=DBNAME \
  --users USER:PASSWORD --datastore_version mysql-5.5 \
  --datastore mysql
```

12 - Iniciando uma Instância

Uma instância é uma Máquina Virtual que o OpenStack provisiona no nó de **compute**. Este guia mostra como iniciar uma instância mínima usando a imagem CirrOS que foi adotada no guia como cobaia. Nestas etapas, serão usados comandos da interface por linha de comando no nó **controller** ou em qualquer outro sistema com as bibliotecas de cliente do OpenStack. O Horizon pode ser usado, alternativamente também.

A instância será iniciada, seja usando o serviço de Rede dedicado do OpenStack (**neutron**), ou a rede legada (**nova-network**).

OBS: Estes procedimentos fazem referência de componentes criados em seções anteriores. Você deve ajustar certos valores (como endereços IP) para se adequar ao seu ambiente usado.

12.1 - Iniciando uma instância com o Serviço de Rede do OpenStack (neutron)

Gerando um par de chaves

A maioria das imagens de nuvem suportam autenticação por chave pública ao invés da autenticação típica por usuário/senha. Antes de iniciar uma instância, você deve gerar um par de chaves pública/privada usando o **ssh-keygen** e adicionando a chave pública em seu ambiente do OpenStack.

1- Faça source nas credenciais do alocador *demo*:

```
$ source demo-openrc.sh
```

2- Gere um par de chaves:

```
$ ssh-keygen
```

3- Adicione a chave pública em seu ambiente OpenStack:

```
$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub demo-key
```

OBS: Este comando não gera nenhuma saída visual.

4- Verifique a adição da chave pública:

```
$ nova keypair-list
+----------+-------------------------------------------------+
| Name     | Fingerprint                                     |
+----------+-------------------------------------------------+
| demo-key | 6c:74:ec:3a:08:05:4e:9e:21:22:a6:dd:b2:62:b8:28 |
+----------+-------------------------------------------------+
```

Para iniciar uma instância

Para iniciar uma instância, você deve pelo menos especificar o *flavor*, nome da imagem, rede, grupo de segurança, chave, e nome da instância.

1- Um *flavor* especifica um perfil virtual de alocação de recursos que inclui processador, memória, e armazenamento.

Liste os *flavors* disponíveis:

```
$ nova flavor-list
```

```
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| 1  | m1.tiny   | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
```

Sua primeira instância usa o *flavor* **m1.tiny**.

OBS: O *flavor* também pode ser referenciado pelo seu **id**.

2- Liste as imagens disponíveis:

```
$ nova image-list
+--------------------------------------+--------------+--------+--------+
| ID                                   | Name         | Status | Server |
+--------------------------------------+--------------+--------+--------+
| 9e5c2bee-0373-414c-b4af-b91b0246ad3b | CirrOS 0.3.1 | ACTIVE |        |
+--------------------------------------+--------------+--------+--------+
```

OBS: Segundo este guia, a imagem cirros-0.3.2-x86_64 será usada, mas dependendo da saída deste comando, você pode usar uma outra versão da imagem do CirrOS.

3- Liste as redes disponíveis:

```
$ neutron net-list
+--------------------------------------+----------+-------------------------------------------------------+
| id                                   | name     | subnets                                               |
+--------------------------------------+----------+-------------------------------------------------------+
| 3c612b5a-d1db-498a-babb-a4c50e344cb1 | demo-net | 20bcd3fd-5785-41fe-ac42-55ff884e3180 192.168.1.0/24   |
| 9bce64a3-a963-4c05-bfcd-161f708042d1 | ext-net  | b54a8d85-b434-4e85-a8aa-74873841a90d 203.0.113.0/24   |
+--------------------------------------+----------+-------------------------------------------------------+
```

Sua primeira instância usa a rede de alocador **demo-net**. Entretanto, você deve referenciar esta rede usando seu **ID** no lugar de seu nome.

4- Liste os grupos de segurança disponíveis:

```
$ nova secgroup-list
+--------------------------------------+---------+-------------+
| Id                                   | Name    | Description |
+--------------------------------------+---------+-------------+
| ad8d4ea5-3cad-4f7d-b164-ada67ec59473 | default | default     |
+--------------------------------------+---------+-------------+
```

Sua primeira instância usa o grupo de segurança padrão. Por padrão, este grupo de segurança implementa um firewall que bloqueia o acesso remoto às instâncias. Se você deseja permitir acesso remoto a sua instância, então configure o acesso remoto.

5- Inicie a instância:
Substitua DEMO_NET_ID pelo ID da rede de alocador demo-net.

```
$ nova boot --flavor m1.tiny --image cirros-0.3.2-x86_64 --nic net-id=DEMO_NET_ID \
  --security-group default --key-name demo-key demo-instance1
```

Property	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	nova
OS-EXT-STS:power_state	0
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	-
OS-SRV-USG:terminated_at	-
accessIPv4	
accessIPv6	
adminPass	vFW7Bp8PQGNo
config_drive	
created	2014-04-09T19:24:27Z
flavor	m1.tiny (1)
hostId	
id	05682b91-81a1-464c-8f40-8b3da7ee92c5
image	cirros-0.3.2-x86_64 (acafc7c0-40aa-4026-9673-b879898e1fc2)
key_name	demo-key
metadata	{}
name	demo-instance1
os-extended-volumes:volumes_attached	[]
progress	0
security_groups	default
status	BUILD
tenant_id	7cf50047f8df4824bc76c2fdf66d11ec
updated	2014-04-09T19:24:27Z
user_id	0e47686e72114d7182f7569d70c519c9

6- Verifique o estado de sua instância:

```
$ nova list
```

ID	Name	Status	Task State	Power State	Networks
05682b91-81a1-464c-8f40-8b3da7ee92c5	demo-instance1	ACTIVE	-	Running	demo-net=192.168.1.3

O **status** muda de BUILD para ACTIVE quando sua instância terminar de ser construída.

Para acessar sua instância usando o console virtual

Obtenha uma URI de sessão de Virtual Network Computing (VNC) para sua isntância e a acesse a partir de um browser da web.

```
$ nova get-vnc-console demo-instance1 novnc
```

Type	Url
novnc	http://controller:6080/vnc_auto.html?token=2f6dd985-f906-4bfc-b566-e87ce656375b

OBS: Se seu browser da web roda em um *host* que não pode resolver o nome do *host* de **controller**, você pode substituir *controller* pelo endereço IP da interface de gerenciamento do seu nó **controller**.

A imagem CirrOS inclui autenticação convencional por usuário/senha e providencia estas credenciais no prompt de login. Após logar no CirrOS, é recomendável que você verifique a conectividade de rede usando **ping**.
Verifique o gateway da rede de alocador demo-net:

```
$ ping -c 4 192.168.1.1
```

PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=64 time=0.357 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=64 time=0.473 ms
64 bytes from 192.168.1.1: icmp_req=3 ttl=64 time=0.504 ms
64 bytes from 192.168.1.1: icmp_req=4 ttl=64 time=0.470 ms

--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.357/0.451/0.504/0.055 ms

Verifique a rede externa ext-net:

```
$ ping -c 4 openstack.org
```

PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_req=1 ttl=53 time=17.4 ms
64 bytes from 174.143.194.225: icmp_req=2 ttl=53 time=17.5 ms
64 bytes from 174.143.194.225: icmp_req=3 ttl=53 time=17.7 ms
64 bytes from 174.143.194.225: icmp_req=4 ttl=53 time=17.5 ms

--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 17.431/17.575/17.734/0.143 ms

Para acessar sua instância remotamente
1- Adicione regras ao grupo de segurança padrão:
a) Permita ICMP (ping):

```
$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

b) Permita acesso de *secure shell* (SSH):

```
$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

```
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

2- Crie um endereço de IP flutuante na rede externa ext-net:

```
$ neutron floatingip-create ext-net
```

Created a new floatingip:

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    |                                      |
| floating_ip_address | 203.0.113.102                        |
| floating_network_id | 9bce64a3-a963-4c05-bfcd-161f708042d1 |
| id                  | 05e36754-e7f3-46bb-9eaa-3521623b3722 |
| port_id             |                                      |
| router_id           |                                      |
| status              | DOWN                                 |
| tenant_id           | 7cf50047f8df4824bc76c2fdf66d11ec     |
+---------------------+--------------------------------------+
```

3- Associe o endereço de IP flutuante a sua instância:

```
$ nova floating-ip-associate demo-instance1 203.0.113.102
```

OBS: Este comando não apresenta nenhuma saída visual.

4- Verifique o estado do seu endereço de IP flutuante:

```
$ nova list
```

```
+--------------------------------------+----------------+--------+------------+-------------+------------------------------------+
| ID                                   | Name           | Status | Task State | Power State | Networks                           |
+--------------------------------------+----------------+--------+------------+-------------+------------------------------------+
| 05682b91-81a1-464c-8f40-8b3da7ee92c5 | demo-instance1 | ACTIVE | -          | Running     | demo-net=192.168.1.3, 203.0.113.102 |
+--------------------------------------+----------------+--------+------------+-------------+------------------------------------+
```

5- Verifique a conectividade de rede usando **ping** a partir do nó **controller** ou qualquer outro *host* na rede externa:

```
$ ping -c 4 203.0.113.102
```

PING 203.0.113.102 (203.0.113.112) 56(84) bytes of data.
64 bytes from 203.0.113.102: icmp_req=1 ttl=63 time=3.18 ms
64 bytes from 203.0.113.102: icmp_req=2 ttl=63 time=0.981 ms
64 bytes from 203.0.113.102: icmp_req=3 ttl=63 time=1.06 ms
64 bytes from 203.0.113.102: icmp_req=4 ttl=63 time=0.929 ms

--- 203.0.113.102 ping statistics ---

4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.929/1.539/3.183/0.951 ms

6- Acesse sua instância por SSH a partir do nó **controller** ou qualquer outro *host* na rede externa:
```
$ ssh cirros@203.0.113.102
```
The authenticity of host '203.0.113.102 (203.0.113.102)' can't be established.
RSA key fingerprint is ed:05:e9:e7:52:a0:ff:83:68:94:c7:d1:f2:f8:e2:e9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '203.0.113.102' (RSA) to the list of known hosts.
$

OBS: Se seu *host* não contiver nenhum par de chaves pública/privada criado em uma etapa anterior, o SSH vai pedir a senha padrão associada ao usuário *cirros*.

Se sua instância não for iniciada ou não parecer funcionar como deveria, veja o Guia de Operações do Openstack para mais informações ou use uma das muitas outras opções para buscar assistência.

12.2 - Iniciando uma instância com a rede legada do OpenStack (nova-network)

Gerando um par de chaves

A maioria das imagens de nuvem suportam autenticação por chave pública ao invés da autenticação típica por usuário/senha. Antes de iniciar uma instância, você deve gerar um par de chaves pública/privada usando o **ssh-keygen** e adicionando a chave pública em seu ambiente do OpenStack.

1- Faça source nas credenciais do alocador *demo*:
```
$ source demo-openrc.sh
```

2- Gere um par de chaves:
```
$ ssh-keygen
```

3- Adicione a chave pública em seu ambiente OpenStack:
```
$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub demo-key
```

OBS: Este comando não gera nenhuma saída visual.

4- Verifique a adição da chave pública:

```
$ nova keypair-list
+----------+-------------------------------------------------+
| Name     | Fingerprint                                     |
+----------+-------------------------------------------------+
| demo-key | 6c:74:ec:3a:08:05:4e:9e:21:22:a6:dd:b2:62:b8:28 |
+----------+-------------------------------------------------+
```

Para iniciar uma instância

Para iniciar uma instância, você deve pelo menos especificar o *flavor*, nome da imagem, rede, grupo de segurança, chave, e nome da instância.

1- Um *flavor* especifica um perfil virtual de alocação de recursos que inclui processador, memória, e armazenamento.

Liste os *flavors* disponíveis:

```
$ nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| 1  | m1.tiny   | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
```

Sua primeira instância usa o *flavor* **m1.tiny**.

OBS: O *flavor* também pode ser referenciado pelo seu **id**.

2- Liste as imagens disponíveis:

```
$ nova image-list
+--------------------------------------+--------------+--------+--------+
| ID                                   | Name         | Status | Server |
+--------------------------------------+--------------+--------+--------+
| 9e5c2bee-0373-414c-b4af-b91b0246ad3b | CirrOS 0.3.1 | ACTIVE |        |
+--------------------------------------+--------------+--------+--------+
```

OBS: Segundo este guia, a imagem cirros-0.3.2-x86_64 será usada, mas dependendo da saída deste comando, você pode usar uma outra versão da imagem do CirrOS.

3- Liste as redes disponíveis:

OBS: Você deve fazer um *source* nas credenciais do alocador *admin* para esta etapa **apenas, e daí usar o *tenant demo* para os passos restantes.**

```
$ source admin-openrc.sh
```

```
$ nova net-list
```

```
+--------------------------------------+----------+------------------+
| ID                                   | Label    | CIDR             |
+--------------------------------------+----------+------------------+
| 7f849be3-4494-495a-95a1-0f99ccb884c4 | demo-net | 203.0.113.24/29  |
+--------------------------------------+----------+------------------+
```

Sua primeira instância usa a rede de alocador **demo-net**. Entretanto, você deve referenciar esta rede usando seu **ID** no lugar de seu nome.

4- Liste os grupos de segurança disponíveis:

```
$ nova secgroup-list
+--------------------------------------+---------+-------------+
| Id                                   | Name    | Description |
+--------------------------------------+---------+-------------+
| ad8d4ea5-3cad-4f7d-b164-ada67ec59473 | default | default     |
+--------------------------------------+---------+-------------+
```

Sua primeira instância usa o grupo de segurança padrão. Por padrão, este grupo de segurança implementa um firewall que bloqueia o acesso remoto às instâncias. Se você deseja permitir acesso remoto a sua instância, então configure o acesso remoto.

5- Inicie a instância:

Substitua DEMO_NET_ID pelo ID da rede de alocador demo-net.

```
$ nova boot --flavor m1.tiny --image cirros-0.3.2-x86_64 --nic net-id=DEMO_NET_ID \
  --security-group default --key-name demo-key demo-instance1
+--------------------------------------+--------------------------------------------------------------+
| Property                             | Value                                                        |
+--------------------------------------+--------------------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                                       |
| OS-EXT-AZ:availability_zone          | nova                                                         |
| OS-EXT-STS:power_state               | 0                                                            |
| OS-EXT-STS:task_state                | scheduling                                                   |
| OS-EXT-STS:vm_state                  | building                                                     |
| OS-SRV-USG:launched_at               | -                                                            |
| OS-SRV-USG:terminated_at             | -                                                            |
| accessIPv4                           |                                                              |
| accessIPv6                           |                                                              |
| adminPass                            | vFW7Bp8PQGNo                                                 |
| config_drive                         |                                                              |
| created                              | 2014-04-09T19:24:27Z                                         |
| flavor                               | m1.tiny (1)                                                  |
| hostId                               |                                                              |
| id                                   | 05682b91-81a1-464c-8f40-8b3da7ee92c5                         |
| image                                | cirros-0.3.2-x86_64 (acafc7c0-40aa-4026-9673-b879898e1fc2)   |
| key_name                             | demo-key                                                     |
| metadata                             | {}                                                           |
| name                                 | demo-instance1                                               |
| os-extended-volumes:volumes_attached | []                                                           |
| progress                             | 0                                                            |
| security_groups                      | default                                                      |
| status                               | BUILD                                                        |
| tenant_id                            | 7cf50047f8df4824bc76c2fdf66d11ec                             |
| updated                              | 2014-04-09T19:24:27Z                                         |
| user_id                              | 0e47686e72114d7182f7569d70c519c9                             |
+--------------------------------------+--------------------------------------------------------------+
```

6- Verifique o estado de sua instância:

```
$ nova list
+--------------------------------------+---------------+--------+------------+-------------+-----------------------+
| ID                                   | Name          | Status | Task State | Power State | Networks              |
+--------------------------------------+---------------+--------+------------+-------------+-----------------------+
| 05682b91-81a1-464c-8f40-8b3da7ee92c5 | demo-instance1 | ACTIVE | -          | Running     | demo-net=192.168.1.3  |
+--------------------------------------+---------------+--------+------------+-------------+-----------------------+
```

O **status** muda de BUILD para ACTIVE quando sua instância terminar de ser construída.

Para acessar sua instância usando o console virtual

- Obtenha uma URL de sessão de Virtual Network Computing (VNC) para sua isntância e a acesse a partir de um browser da web.

```
$ nova get-vnc-console demo-instance1 novnc
+-------+-----------------------------------------------------------------------------------------+
| Type  | Url                                                                                     |
+-------+-----------------------------------------------------------------------------------------+
| novnc | http://controller:6080/vnc_auto.html?token=2f6dd985-f906-4bfc-b566-e87ce656375b    |
+-------+-----------------------------------------------------------------------------------------+
```

OBS: Se seu browser da web roda em um *host* que não pode resolver o nome do *host* de **controller**, você pode substituir *controller* pelo endereço IP da interface de gerenciamento do seu nó **controller**.

A imagem CirrOS inclui autenticação convencional por usuário/senha e providencia estas credenciais no prompt de login. Após logar no CirrOS, é recomendável que você verifique a conectividade de rede usando **ping**.

Verifique o gateway da rede de alocador demo-net:

```
$ ping -c 4 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=64 time=0.357 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=64 time=0.473 ms
64 bytes from 192.168.1.1: icmp_req=3 ttl=64 time=0.504 ms
64 bytes from 192.168.1.1: icmp_req=4 ttl=64 time=0.470 ms

--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.357/0.451/0.504/0.055 ms
```

Verifique a rede externa ext-net:

```
$ ping -c 4 openstack.org
PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_req=1 ttl=53 time=17.4 ms
64 bytes from 174.143.194.225: icmp_req=2 ttl=53 time=17.5 ms
64 bytes from 174.143.194.225: icmp_req=3 ttl=53 time=17.7 ms
64 bytes from 174.143.194.225: icmp_req=4 ttl=53 time=17.5 ms
```

--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 17.431/17.575/17.734/0.143 ms

Para acessar sua instância remotamente

1- Adicione regras ao grupo de segurança padrão:

a) Permita ICMP (ping):

```
$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

b) Permita acesso de *secure shell* (SSH):

```
$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

2- Verifique a conectividade de rede usando **ping** a partir do nó **controller** ou qualquer outro *host* na rede externa:

```
$ ping -c 4 203.0.113.102
PING 203.0.113.102 (203.0.113.112) 56(84) bytes of data.
64 bytes from 203.0.113.102: icmp_req=1 ttl=63 time=3.18 ms
64 bytes from 203.0.113.102: icmp_req=2 ttl=63 time=0.981 ms
64 bytes from 203.0.113.102: icmp_req=3 ttl=63 time=1.06 ms
64 bytes from 203.0.113.102: icmp_req=4 ttl=63 time=0.929 ms

--- 203.0.113.102 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.929/1.539/3.183/0.951 ms
```

3- Acesse sua instância por SSH a partir do nó **controller** ou qualquer outro *host* na rede externa:

```
$ ssh cirros@203.0.113.102
The authenticity of host '203.0.113.102 (203.0.113.102)' can't be established.
RSA key fingerprint is ed:05:e9:e7:52:a0:ff:83:68:94:c7:d1:f2:f8:e2:e9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '203.0.113.102' (RSA) to the list of known hosts.
```

OBS: Se seu *host* não contiver nenhum par de chaves pública/privada criado em uma etapa anterior, o SSH vai pedir a senha padrão associada ao usuário *cirros*.

Se sua instância não for iniciada ou não parecer funcionar como deveria, veja o Guia de Operações do Openstack para mais informações ou use uma das muitas outras opções para buscar assistência.

Apêndice A - Suporte da Comunidade

Troubleshooting - problemas encontrados

Problema: Tentar lançar uma instância gerou esta mensagem no /var/log/nova/nova-compute.log correspondente ao nó que hospedaria a instância (no caso, o *compute node*):

Error: Failed to launch instance "testeNeutron": Please try again later [Error: Error during following call to agent: ['ovs-vsctl', '--timeout=120', 'del-port', 'br-int', u'qvo087b0693-88']].

2014-06-06 16:23:47.945 1820 ERROR oslo.messaging._drivers.common [-] ['Traceback (most recent call last):\n', ' File "/usr/lib/python2.7/dist-packages/oslo/messaging/rpc/dispatcher.py", line 133, in _dispatch_and_reply\n incoming.message))\n', ' File "/usr/lib/python2.7/dist-packages/oslo/messaging/rpc/dispatcher.py", line 176, in _dispatch\n return self._do_dispatch(endpoint, method, ctxt, args)\n', ' File "/usr/lib/python2.7/dist-packages/oslo/messaging/rpc/dispatcher.py", line 122, in _do_dispatch\n result = getattr(endpoint, method)(ctxt, **new_args)\n', ' File "/usr/lib/python2.7/dist-packages/oslo/messaging/rpc/server.py", line 139, in inner\n return func(*args, **kwargs)\n', ' File "/usr/lib/python2.7/dist-packages/nova/exception.py", line 88, in wrapped\n payload)\n', ' File "/usr/lib/python2.7/dist-packages/nova/openstack/common/excutils.py", line 68, in __exit__\n six.reraise(self.type_, self.value, self.tb)\n', ' File "/usr/lib/python2.7/dist-packages/nova/exception.py", line 71, in wrapped\n return f(self, context, *args, **kw)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 280, in decorated_function\n pass\n', ' File "/usr/lib/python2.7/dist-packages/nova/openstack/common/excutils.py", line 68, in __exit__\n six.reraise(self.type_, self.value, self.tb)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 266, in decorated_function\n return function(self, context, *args, **kwargs)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 333, in decorated_function\n function(self, context, *args, **kwargs)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 309, in decorated_function\n e, sys.exc_info())\n', ' File "/usr/lib/python2.7/dist-packages/nova/openstack/common/excutils.py", line 68, in __exit__\n six.reraise(self.type_, self.value, self.tb)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 296, in decorated_function\n return function(self, context, *args, **kwargs)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 2075, in run_instance\n do_run_instance()\n', ' File "/usr/lib/python2.7/dist-packages/nova/openstack/common/lockutils.py", line 249, in inner\n return f(*args, **kwargs)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 2074, in do_run_instance\n legacy_bdm_in_spec)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 1207, in _run_instance\n notify("error", fault=e) # notify that build failed\n', ' File "/usr/lib/python2.7/dist-packages/nova/openstack/common/excutils.py", line 68, in __exit__\n six.reraise(self.type_, self.value, self.tb)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 1191, in _run_instance\n

instance, image_meta, legacy_bdm_in_spec)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 1355, in _build_instance\n filter_properties, bdms, legacy_bdm_in_spec)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 1401, in _reschedule_or_error\n self._log_original_error(exc_info, instance_uuid)\n', ' File "/usr/lib/python2.7/dist-packages/nova/openstack/common/excutils.py", line 68, in __exit__\n six.reraise(self.type_, self.value, self.tb)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 1396, in _reschedule_or_error\n bdms, requested_networks)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 2125, in _shutdown_instance\n requested_networks)\n', ' File "/usr/lib/python2.7/dist-packages/nova/openstack/common/excutils.py", line 68, in __exit__\n six.reraise(self.type_, self.value, self.tb)\n', ' File "/usr/lib/python2.7/dist-packages/nova/compute/manager.py", line 2115, in _shutdown_instance\n block_device_info)\n', ' File "/usr/lib/python2.7/dist-packages/nova/virt/libvirt/driver.py", line 953, in destroy\n destroy_disks)\n', ' File "/usr/lib/python2.7/dist-packages/nova/virt/libvirt/driver.py", line 989, in cleanup\n self.unplug_vifs(instance, network_info)\n', ' File "/usr/lib/python2.7/dist-packages/nova/virt/libvirt/driver.py", line 860, in unplug_vifs\n self.vif_driver.unplug(instance, vif)\n', ' File "/usr/lib/python2.7/dist-packages/nova/virt/libvirt/vif.py", line 783, in unplug\n self.unplug_ovs(instance, vif)\n', ' File "/usr/lib/python2.7/dist-packages/nova/virt/libvirt/vif.py", line 667, in unplug_ovs\n self.unplug_ovs_hybrid(instance, vif)\n', ' File "/usr/lib/python2.7/dist-packages/nova/virt/libvirt/vif.py", line 661, in unplug_ovs_hybrid\n v2_name)\n', ' File "/usr/lib/python2.7/dist-packages/nova/network/linux_net.py", line 1317, in delete_ovs_vif_port\n _ovs_vsctl([\'del-port\', bridge, dev])\n', ' File "/usr/lib/python2.7/dist-packages/nova/network/linux_net.py", line 1302, in _ovs_vsctl\n raise exception.AgentError(method=full_args)\n', "AgentError: Error during following call to agent: ['ovs-vsctl', '--timeout=120', 'del-port', 'br-int', u'qvo84d1a3a9-52']\n"]

Solução: como no compute havia um diretório /var/lib/nova/instances onde foi realizado um **bind** (vinculado a outro diretório, de outra partição), as permissões deste diretório não haviam sido atualizadas para que o OpenStack tivesse permissão para realizar alterações neste. Então, foi usado o comando:
$ chown nova.nova -R instances/

Problema: Baixa vazão nas conexões das instâncias; falha de comunicação de ssh para instâncias, e a partir de instâncias:

Solução:

Trata-se de um problema recorrente do Quantum, reportado em:

https://ask.openstack.org/en/question/6140/quantum-neutron-gre-slow-performance/

Passos:

Editar no arquivo /etc/neutron/dhcp_agent.ini a linha

...

dnsmasq_config_file=/etc/neutron/dnsmasq-neutron.conf

...

Em seguida criar o arquivo /etc/neutron/dnsmasq-neutron.conf e dentro dele colocar

dhcp-option-force=26,1400

Apos isso, reiniciar o serviço neutron-dhcp-agent .

OBS: Este procedimento funciona para outros tipos de imagem, mas não é garantido solucionar o caso para a imagem-exemplo do Cirros 0.3.2, usada neste manual. O procedimento foi validado usando uma imagem pronta para cloud do Ubuntu 14.04.