Ganeti e Xen

Virtualização com Ganeti e Xen

Aprenda como utilizar o poderoso Ganeti, que em conjunto com o Xen irá lhe auxiliar no gerenciamento de máquinas virtuais com alta disponibilidade.

por Sérgio Pelissari Junior

Temos atualmente no mercado, algumas soluções Open Source para virtualização, como por exemplo, Xen e KVM, que fazem muito bem suas funções para as quais foram designadas, e já se mostraram suficientemente maduras para serem colocadas em um ambiente de produção. O único problema dessas soluções encontra-se nas complicadas configurações necessárias para gerenciamento do ambiente virtual. Pensando nisto, o Google disponibilizou uma ferramenta Open Source chamada Ganeti que elimina boa parte de toda a preocupação de qualquer SysAdmin.

O Ganeti é uma ferramenta para gerenciamento de virtualização baseada em cluster, e que nas versões 1.x comportava apenas o Hypervisor Xen. Atualmente, O Ganeti já está disponível na versão 2.x incluindo também o Hypervisor KVM. Este artigo irá demonstrar como utilizar esta poderosa ferramenta com o intuito de gerenciar suas máquinas virtuais e conseguir recursos de alta disponibilidade.

O Ganeti é uma aplicação criada para cuidar da parte de gerenciamento dos Hypervisors Xen e KVM. Além desse processo de gerenciamento, o Ganeti trabalha em conjunto com outra ferramenta bastante interessante e Open Source chamada *DRBD*.

O DRBD é uma aplicação Open Source que aplica o conceito Raid-1 na rede, ou seja, caso você tenha duas partições KVM, uma em cada um servidor, com o `drbd` conseguirá mantê-las sincronizadas replicando cada mudança de uma para a outra, ou para ambas simultaneamente.

No cenário apresentado neste artigo foi utilizada a distribuição Ubuntu Hardy 8.04 64bits, Xen 3.3.0, DRBD8.02 e Ganeti 2, lembrando que os procedimentos de instalação devem ser executados em todos os nodes onde é desejado usar o Ganeti.

Antes de prosseguir, precisamos entender alguns conceitos: o Ganeti funciona de forma hierárquica, sendo assim, todas as configurações realizadas pelo `gnt-cluster`, serão herdadas pelos nodes. Configurações informadas ao sistema com o `gnt-node` afetarão apenas o node designado, e os comandos utilizados pelo `gnt-instance` serão acatadados apenas pelas instâncias designadas. Resumindo, o segmento é o de cluster, node e instância, onde cluster é todo o ambiente integrado com cada servidor disponibilizando seus recursos, nodes são os servidores que neste caso estarão executando o Xen Hypervisor e por último as instâncias, também conhecidas como máquinas virtuais.

Instalação

Para a instalação do Ganeti, consulte o manual disponibilizado pela equipe de desenvolvimento e que pode ser encontrada em [1]. Os pré-requisitos para a instalação, são:

1. Xen compilado e instalado;
2. Código fonte do do kernel.

Para cada máquina virtual ou node adicionado ao cluster é preciso uma refêrencia ao `dns`. Dependendo do tamanho do cluster, as entradas podem ser adicionadas no arquivo `hosts`.

Após a instalação e início do cluster é preciso adicionar nodes ao cluster, executando o seguinte comando (lembrando que o node02 deve estar na entrada `/etc/hosts`):

```
root@node01:/# gnt-node add node02
```

Agora verifique se o novo node foi adicionado ao cluster (**listagem 1**).

Como podemos ver, ambos estão no ar e já temos suas informações. O node01 é o primeiro node ao qual iniciamos o cluster e o node02 foi o recém adicionado. Neste momento vamos adicionar uma máquina virtual standalone com 256 de memória e com 2GB em disco.

```
gnt-instance add -t plain --disk
➥ 0:size=2g -B memory=256 -o
➥ debootstrap -n node01 instance01
```

O comando `gnt-instance` é destinado ao gerenciamento de máquinas virtuais, e conforme a sintaxe acima estamos usando em conjunto o comando `gnt-instance` e a opção `add` para adicionar uma máquina virtual com seu tipo (`-t`) *plain* que neste caso, é uma máquina normal criada em um volume lógico. Podemos usar outros tipos de instances como `drbd` para alta disponibilidade e também o modo `diskless`.

Com a opção `--disk`, podemos criar quantas partições quisermos, por exemplo, se deserjamos duas partições de 2GB, basta informar na sintaxe `--disk 0:size=2g --disk 1:size=2g` e adicionando a opção `-B`, o backend padrão será utilizado. A opção *memory*, como o próprio nome já diz, é destinada a quantidade de memória, a opção `-o` nos indica qual script para a criação da máquina virtual será utilizado (e neste caso é o script `debootstrap`, a opção `-n node01` indica onde será criada a máquina virtual e por último o nome da nova máquina virtual, que deverá corresponder ao mesmo nome presente em `/etc/hosts`.

Após a saída do script vamos verificar o status da máquina virtual com o comando:

```
root@node01:/# gnt-instance list
Instance              Hypervisor
➥ OS          Primary_node
➥Status  Memory
instance01.ganeti.lab2 xen-pvm
➥ debootstrap node01.ganeti.lab
➥ running   256M
```

Listagem 1: Node adicionado ao cluster

```
root@node01:/# gnt-node list
Node              DTotal  DFree MTotal MNode MFree Pinst Sinst
node01.ganeti.lab2 139.2G 133.1G   1.9G  512M  1.3G     1     1
node02.ganeti.lab2 139.7G 136.6G   3.9G  512M  3.2G     1     0
```

Listagem 2: Sincronização pelo DRBD

```
[root@node01 ~]# cat /proc/drbd
version: 8.3.0 (api:88/proto:86-89)
GIT-hash: 9ba8b93e24d842f0dd3fb1f9b90e8348ddb95829 build by root@
➥node01.labcloud.int, 2009-02-13 11:24:41
0: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C r-
ns:4166540 nr:0 dw:0 dr:4174560 al:0 bm:254 lo:1 pe:5 ua:252 ap:0
➥ ep:1 wo:b oos:1076284
[==============>.....] sync'ed: 79.6% (1051/5119)M
finish: 0:01:22 speed: 25,932 (25,352) K/sec
```

Listagem 3: Failover

```
[root@node01 ~]#gnt-instance failover instance02
node1:~# gnt-instance failover instance02.ganeti.lab
Failover will happen to image inst1.example.com. This requires a
shutdown of the instance. Continue?
y/[n]/?: <-- y
* checking disk consistency between source and target
Node node2.ganeti.lab: Disk degraded, not found or node down
Failure: command execution error:
Disk sda is degraded on target node, aborting failover.
[root@node1 ~]#
```

Listagem 4: Sincronização do disco

```
 [root@node02 ~]# cat /proc/drbd
 version: 8.3.0 (api:88/proto:86-89)
  IT-hash: 9ba8b93e24d842f0dd3fb1f9b90e8348ddb95829 build by root@
➥node02.labcloud.int, 2009-02-13 11:24:41
 0: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C r-
  ns:4166540 nr:0 dw:0 dr:4174560 al:0 bm:254 lo:1 pe:5 ua:252 ap:0
➥ ep:1 wo:b oos:1076284
  [========>...........] sync'ed: 79.6% (1051/5119)M
  inish: 0:01:22 speed: 25,932 (25,352) K/sec
```

Alta disponibilidade

A partir deste momento, é possível utilizar a nova máquina virtual, instalar programas etc. Execute `gnt-instance console instance01` e você será direcionado ao shell de logon da máquina. Porém, ainda não existe redundância da máquina virtual caso ela ou o node ao qual foi instanciada apresente algum problema, e em ambientes que precisam ter seus sistemas sem pausas de serviço não programadas, seria motivo de muitas dores de cabeça, por isso é interessante incluir um trecho de *failover* e alta disponibilidade.

Para começar, precisamos de pelo menos dois nodes no cluster, configur devidamente o DRBD e fazer uma pequena alteração no arquivo de configuração do Xen, adicionando ou apenas descomentando as seguintes linhas:

```
(xend-relocation-server yes)
(xend-relocation-port 8002)
(xend-relocation-hosts-allow
➥ '^localhost$ ^localhost\\.
```

```
➥localdomain$ ^192\\.
➥168\\.1\\.65$')
```

A primeira linha habilita a *relocation* (migração), a segunda linha especifica a porta utilizada para o *relocation* e por último os hosts que podem alocar suas máquinas virtuais, e que pode ser configurado por host, domínio ou intervalo de IPs da rede. Após a inclusão dessas linhas reinicie o serviço do Xen:

```
/etc/init.d/xend restart
```

Crie agora a primeira instância do tipo DRBD, através do comando:

```
gnt-instance add -t drbd --disk
➥ 0:size=2g -B memory=256 -o
➥ debootstrap -n node01:node02
➥ instance02
```

Lembre-se também de criar uma entrada para esta intância no arquivo `/etc/hosts`. Após confirmar o comando, será exibida uma sincronização entre os nodes, criando a máquina virtual. Caso queira acompanhar o processo de sincronização pelo DRBD utilize o comando `watch -n1 cat /proc/drbd` e você verá uma tela semelhante a msotrada na **listagem 2**.

Após o término da sincronização, a máquina virtual terá seu status marcado como *running* (rodando ou em funcionamento). Para verificar os status das máquinas virtuais, execute:

```
[root@node01 ~]#gnt-instance list
Instance              Hypervisor OS
   Primary_node       Status  Memory
instance02.ganeti.lab xen-pvm
➥debootstrap node01.ganeti.lab
➥ running   256M
```

E então é possível testar a alta disponibilidade e o *failover* através do seguinte comando:

```
[root@node01 ~]#gnt-instance
➥ migrate instance02
```

É possível acompanhar o processo de migração para outra estação sem a queda do serviço, e com isso é possível perceber a agilidade em possíveis futuras manutenções ou implementações sem se preocupar com a disponibilidade do sistema. E se um node porventura ficar indisponível? Então teríamos de avisar para o Ganeti de que precisamos de um *failover* para a instância em questão. Vamos fazer uma pequena simulação.

A instância atual foi migrada para o node02, sendo assim, vamos puxar o cabo de rede do node02 e dizer ao node01 fazer um *failover* desta instância (**listagem 3**).

Note que o `gnt-instance` reclamou da checagem de consistência de discos porque não conseguiu fazer a consulta ao outro node e então ativou a instance02 no node1. Ao colocarmos novamente o cabo ao node2 não será possível fazer a migração da instance02 pois serão exibidos avisos de inconsistência de disco. Neste caso será necessário avisar para o DRBD que o dispositivo é inválido e precisa de uma nova sincronização.

O node02, do qual puxamos o cabo, não está mais sincronizado com o dispositivo do node01 então vamos selecionar este dispositivo invalidá-lo utilizado o comando:

```
[root@node01 ~]#drbd primary /dev/
➥drbd0   # Deixe o dispositivo como
➥ primário para que este receba
➥ as devidas mudanças
```

```
[root@node01 ~]#drbd invalidate
➥/dev/drbd0  # Tornando o
➥ dispositivo inválido
```

Após executar este comando você pode conferir através do comando `watch -n1 cat /proc/drbd` que será feita uma nova sincronização de todo o disco novamente (**listagem 4**).

Finalizando a migração

Ao finalizar a migração do sistema utilizando o comando `gnt-instance migrate instance02`, esta instância estará disponível novamente. É possível ainda efetuar outras mudanças nos nodes primário e secundário das instâncias, caso seja necessário. Note que as máquinas virtuais do tipo DRBD trabalham em duplas, sendo assim, toda a mudança feita em uma instância é replicada para a segunda.

Para recuperar-se de uma falha do node master ou principal, você pode designar outro node para se tornar o master, até que resolva o problema. Neste caso vamos alterar o master para o node02:

```
[root@node02 ~]#gnt-cluster
➥ masterfailover
```

E então o novo cluster master passará a ser o node02 e continuará o gerenciamento até que o node01 volte a funcionar corretamente. ■

Mais informações

[1] Instalação do Ganeti:
http://ganeti-doc.googlecode.com/svn/ganeti-1.2/install.html

Gostou do artigo?

Queremos ouvir sua opinião. Fale conosco em
cartas@linuxmagazine.com.br

Este artigo no nosso site:
http://lnm.com.br/article/3852