Use LibVirt to bootup VMs in a virtualization-agnostic fashion and on-demand as switches connect to the network

Grupo:

Andre Abade, Bruno Arndt, Jean Menossi, Luiz Pacini

Objetivos:

Possibilitar por meio do acoplamento da API LibVirt a inicialização de máquinas virtuais (VMs) independente do *hypervisor*.

Proposta:

Implementar uma integração por meio de um console o gerenciamento das VMs do RouteFlow. O console faria uso de uma API denominada LibVirt para gerenciar as máquinas virtuais, onde utilizando o conceito de *wizard* seria gerada toda configuração em um arquivo em formato XML. O formato XML permitiria uma flexibilidade quanto a administração das configurações das VMs portáveis para quaisquer *hypervisors* do RouteFlow.

Implementação:

Como proposto, foi implementado uma *wizard* na linguagem *bash script* que a partir de configurações definidas pelo usuário, a *wizard* gera um arquivo de configuração no formato XML. A partir do XML gerado, a API LibVirt configura a VM com as configurações definidas no arquivo XML e inicializa o hypervisor de acordo com sua correspondencia na organização do rftest.

As implementações foram todas feitas na linguagem *bash script* e foi organizada da seguinte maneira:

- A **wizard** (*libvirt.sh*) foi implementada em conjunto com a API *dialog*, que proporciona uma interatividade maior com o usuário comparado ao *bash script* modo texto. Nela, existe um fluxo na qual são definidas algumas configurações para a VM. Mais detalhes adiante.
- O **console de gerenciamento** (*rftest.sh*) foi implementado, assim como a *wizard*, em conjunto com a API *dialog*. Nele, é possível configurar uma nova VM a partir da *wizard* ou então reaproveitar um XML já existente. Este processo é executado para cada VM definida nos *rftests*. Mais detalhes adiante,
- O **teste** (*rftest2*) foi alterado do original para possibilitar a integração dos componentes

explicados anteriormente, de forma que a arquitetura permita a utilização de multiplos *hypervisors* ao invés de apenas LXC.

Para o funcionamento das implementações, os *bash scripts* citados acima devem ser colocados no diretório /home/routeflow/RouteFlow/rftest/. Além disso, são necessários a criação de dois diretórios:

- XML: Contém os arquivos XML das VMs configuradas pela *wizard*. Este diretório é criado automaticamente pela *wizard*.
- Images: Contém as imagens do SO das VMs.

Para executar a implementação, basta seguir o processo normal do RouteFlow e executar o script *rftest2* e este fará uso dos demais *bash scripts*.

Para o seu funcionamento, é necessário uma configuração prévia de uma imagem do SO de cada uma das VMs, na qual a configuração deve conter as configurações do ***quagga***, ***mongoDB***, ***ospf*** e **interfaces de rede** pertinentes a cada VM especificada no *rftest*. Por exemplo, no *rftest2* temos 4 VMs, sendo elas *rfvmA*, *rfvmB*, *rfvmC* e *rfvmD*, portanto, serão necessárias 4 imagens do SO. Por convenção, as imagens devem possuir o nome *h1*, *h2*, *h3* e *h4*.

Nos testes realizados, as configurações do *quagga*, *mongoDB* e *ospf* foram copiadas do script *rftest2* original e transferidas para cada uma das imagens. Já as interfaces de rede foram definidas no arquivo /etc/network/interfaces e no *rc.local*, sendo este último uma alternativa para a definição de interfaces virtuais com os MACs especificados.

Obs.: As imagens não foram submetidas devido o seu tamanho (aprox. 8Gb cada, ou seja, para as 4 VMs temos 32Gb).

Estrutura do XML:

Por definição, o conteúdo do XML definido em **negrito** é estático e independe dos dados de entrada fornecidos pelo usuário. Já o conteúdo definido em **vermelho** é dinâmico e depende das informações fornecidas pelo usuário.

O elemento chave requerido por todas as VMs é chamado **`domain`**. Ele possui dois atributos, **`type`** e **`id`**. O primeiro especifica o *hypervisor* usado e o segundo especifica um identificador inteiro e único para a VM.

A seguir, temos o metadado **`name`** que define um nome único para a VM. O arquivo XML gerado possui este mesmo nome.

Temos então a definição da inicialização do SO na tag **`os`**. O elemento **`type`** especifica o tipo do SO e o atributo **`arch`** define a arquitetura do SO. Caso o *hypervisor* seja LXC, temos que adicionar a tag **`init`** que define o caminho do binário de inicialização do SO. Caso contrário, devemos adicionar a tag **`boot`**, que define o dispositivo de inicialização **`dev`**, neste caso igual a hd.

Prosseguindo, tempos os elementos **`vcpu`**, que define a quantidade de CPUs, e

memory, que define a quantidade de memória (em MB).

Temos, então, algumas tags do libvirt para casos de **on_poweroff**, **on_reboot** e **on_crash**, na qual sua configuração é estática e irrelevante para o entendimento do XML, logo não entraremos em detalhes.

E finalmente temos a definição dos componentes da VM na tag **devices**, onde podemos definir disco, interfaces de rede, etc. Como as interfaces de rede são definidas estaticamente direto na imagem, então entraremos em detalhes apenas na tag **disk**, que define o HD da VM. Como a única opção definida na *wizard* é através de uma imagem, então os atributos **type** e **device** da tag **disk** são estáticos. Dentro da tag **disk** podemos definir o caminho onde se encontra a imagem no atributo **file** da tag **source** e o dispositivo destino da VM emulada no atributo **dev** da tag **target**. Neste último caso, o atributo é estático para *hda*, ou seja, o HD principal da VM.

A seguir, temos um modelo de XML e também o possíveis valores aceitos por cada atributo destacado.

```
<domain type='HYPERVISOR' id='ID'>
      <name>NAME</name>
      <os>
            <type arch='ARCH'>TYPEOS</type>
            <boot dev='hd'/>
            <init>/bin/bash</init>
      </os>
      <vcpu>CPU</vcpu>
      <memory unit='MB'>MEMORY</memory>
      <on_poweroff>destroy</on_poweroff>
      <on_reboot>restart</on_reboot>
      <on_crash>restart</on_crash>
      <devices>
            <disk type='file' device='disk'>
                  <source file='IMAGE'/>
                  <target dev='hda'/>
            </disk>
            <console type='pty'/>
      </devices>
</domain>
```

HYPERVISOR = {KVM, QEMU, LXC}
ID = número inteiro único
NAME = nome composto por caracteres alfa-numéricos, sendo único no escopo.
ARCH = {i686, x86_64}
TYPEOS = {hvm, linux, exe}
CPU = número inteiro que corresponde ao números de CPUs

MEMORY = número inteiro que corresponde à quantidade de memória em MB
IMAGE = caminho da localização da imagem do SO

Console de Gerenciamento:

O console de gerenciamento é responsável pela análise do XML e inicializar as VMs de acordo com seu *hypervisor* e *rfvm*. Inicialmente, é verificado se o usuário deseja configurar uma nova VM através da *wizard* implementada ou se deseja selecionar um XML existente. Caso seja configurado um novo XML, o console gera o XML e o utiliza em seguida para a rfvm correspondente da execução do console. Em ambos os casos, é feito um parser para verificar qual o *hypervisor* definido no XML.

A próxima etapa é definir qual imagem será utilizada. Por convenção, foi definido que para a VM *rfvmA* será atribuída a imagem cujo nome é *h1.img,* para a VM *rfvmB* será atribuída a imagem cujo nome é *h2.img*, e assim por diante, para as 4 VMs definidas no *rftest2*. As imagens *h1.img*, *h2.img*, *h3.img* e *h4.img* devem estar localizadas no diretório ./Images/.

Enfim, após definir as imagens é então realizada a inicialização das VMs de acordo com seu *hypervisor*, descoberto na primeira etapa através do parser. Para cada *hypervisor* é necessário uma sequência de comandos responsáveis por destruir um domínio caso ele já exista, indefinir um domínio caso ele já esteja definido, definir o domínio a partir do XML e iniciar a VM a partir do domínio definido pelo XML.

- **LXC:**

```
virsh -c lxc:/// destroy XML_NAME
virsh -c lxc:/// undefine XML_NAME
virsh -c lxc:/// define XML_SOURCE
virsh -c lxc:/// start XML_NAME
```

- **QEMU:**

```
virsh -c qemu:///system destroy XML_NAME
virsh -c qemu:///system undefine XML_NAME
virsh -c qemu:///system define XML_SOURCE
virsh -c qemu:///system start XML_NAME
```

- **KVM:**

```
virsh -c kvm:/// destroy XML_NAME
virsh -c kvm:/// undefine XML_NAME
virsh -c kvm:/// define XML_SOURCE
virsh -c kvm:/// start XML_NAME
```

Após a inicialização das VMs o script é finalizado e retomada a execução do rftest2 que retomará as configurações do RFClient, RFServer e RFProxy.

Como as interfaces de rede foram definidas nas imagens, ao iniciar as VMs, as

interfaces de rede são automaticamente inicializadas e ao relacioná-las com o ospf é possível virtualizar os túneis simulados no rftest.

Configuração da Imagem das VMs

Com a evolução do RouteFlow para possibilitar múltiplos hypervisors, todo a configuração antes realizada pelo LXC no script rftest2 passou a ser estática. Desta forma todo conteúdo antes copiado à cada execução do script de teste tornou-se única em cada uma das imagens, necessitanto portanto que sejam específicas para seu uso final. A necessidade de criar estas imagens fica evidente logo que, o script anterior, trabalhava de forma para-virtualizada, diferente dos outros hypervisors que a Libvirt tem a possibilidade de trabalhar. Desta forma, as imagens ficam padrão para sua utilização de acordo com a necessidade específica, no caso, o script rftest2. Dentro de cada uma das imagens, as configurações de Mongo, Quagga, Ospf são estáticas de sua respectiva funcionalidade na execução do script.