

GERENCIAMENTO REMOTO DE SERVIDORES COM A FERRAMENTA REX

Breno Farias Fonseca¹
Aluizio Ferreira da Rocha Neto²

RESUMO

Este artigo foi desenvolvido com o intuito de elucidar os conhecimentos referentes ao gerenciamento remoto de servidores, utilizando um framework denominado “Rex”, que foi concebido com a linguagem de programação Perl. O gerenciamento remoto utilizando essa ferramenta possibilita uma centralização das operações executadas no servidor, consequentemente um maior ganho de desempenho, visando simplificar a rotina dos administradores de sistemas.

Palavras-chave: Gerenciamento Remoto. Framework Rex. Linguagem Perl.

REMOTE MANAGEMENT OF SERVERS WITH REX TOOL

ABSTRACT

This article was written with the aim of elucidating the knowledge concerning the management of remote servers, using a framework called “Rex”, which was designed with the Perl programming language. The remote management using this tool enables centralization of the operations performed on the server, hence a greater performance gain, aiming to simplify the routine of server administrators.

Keywords: Remote Management. Rex Framework. Perl.

1 INTRODUÇÃO

A ferramenta de gerenciamento demonstrada neste artigo, permite que você gerencie seus servidores através de um ponto central. Em um único arquivo de configuração, o administrador poderá comunicar-se com todos seus servidores, se necessário.

Com o gerenciamento remoto, além de centralizar as operações do administrador, também ocorre que, em qualquer máquina com conectividade aos servidores, pode ser feito o gerenciamento. Possibilitando assim uma maior mobilidade ao responsável pelos servidores.

¹Discente do Curso de Especialização em Redes de Computadores pelo Centro Universitário do Rio Grande do Norte - UNI-RN. Bacharelado em Sistemas de Informação pelo Centro Universitário do Rio Grande do Norte - UNI-RN.
E-mail: brenofarias87@yahoo.com.br

²Docente do Curso de Especialização em Redes de Computadores do Centro Universitário do Rio Grande do Norte - UNI-RN. Mestrado em Ciência da Computação pela UNICAMP, Brasil.
E-mail: aluiziorocha@gmail.com

Existem várias ferramentas de gerenciamento disponíveis para os administradores de sistemas. Este artigo não tem a intenção de apoiar ou influenciar o uso de uma ferramenta de gerenciamento específica, mas sim de despertar o interesse e a pesquisa, permitindo assim que o leitor escolha a ferramenta que achar conveniente.

Em primeiro plano, será introduzido um guia referencial para a utilização e compreensão da ferramenta de gerenciamento, e será apresentada outras alternativas de ferramentas que possam ser utilizadas. Em seguida, será exposta à ferramenta de gerenciamento utilizada. E por fim, será demonstrado modelos de gerenciamento, a utilização da ferramenta em si.

2 DESCRIÇÃO DETALHADA DO PROBLEMA

Em uma intranet corporativa a demanda por múltiplos servidores é muito alta, não somente por questões de quantidade de usuários, mas por questões que envolvem o desempenho, a disponibilidade e a segurança dos dados, atributos esses ligados a estrutura da rede. Por trás de toda a parte física, é montada toda uma estrutura de controle - que pode ser planejada através de uma ferramenta de gerenciamento - para que se consolide o acesso.

Quanto a utilização de múltiplos servidores, existe a questão da padronização, as empresas buscam ferramentas que permitam definir modelos de estruturas para seus servidores e replicá-los de forma rápida e eficiente.

Com uma grande frequência, se faz necessária a execução de comandos nesses servidores, mesmo que pré configurados. A ferramenta de gerenciamento demonstrada nesse artigo, possibilita uma estrutura de comandos a ser executado em blocos, blocos esses compostos de instruções que serão enviadas para o servidor à medida e ao modo que o administrador achar necessário. Por possuir toda essa estrutura, a ferramenta de gerenciamento permite a automação de diversos serviços.

O gerenciamento garante que uma pessoa, com o conhecimento correto, entregue e ofereça suporte aos serviços exigidos pela empresa.

3 GUIA DE REFERÊNCIA EM PERL

Para começar a parte relativa ao uso da ferramenta é necessário uma parte introdutória da linguagem de programação em que o programa trabalha (Perl). Apenas será apresentado o necessário para o uso ou o entendimento de como a ferramenta trabalha.

3.1 PERL - VARIÁVEIS

Em Perl existem 3 tipos de variáveis: escalares, *arrays* e *hash*. Uma variável escalar é assim denominada porque contém um valor escalar. Nomes de variáveis escalares sempre começam com um cifrão(\$). Uma variável *array* armazena uma lista de escalares. Cada array pode guardar 0 ou mais escalares individuais. Cada elemento individual de um array tem um número de posição (ou índice), que é usado para acessar (ou fazer referência a) aquele elemento. Um variável *array* sempre começa com “@” e seus elementos são acessados através de colchetes ([]). O segundo tipo de dados que armazena vários valores (não-escalar) que é fundamental para a linguagem Perl é a tabela *hash* ou *array* associativo. Uma tabela *hash* é uma coleção não-ordenada de pares chave-valor. Em vez de acessar um elemento da tabela *hash* com um subscrito, como em um array, elementos de uma tabela *hash* são acessados com um *string* conhecido como uma chave. Cada chave deve ser única. Tabelas *hash* às vezes são chamadas de *array* associativos, porque elas definem associações entre chaves e seus valores. Uma variável *hash* sempre começa com um “%” e seus elementos são acessados através de chaves ({ }) [1].

3.2 PERL - SUB ROTINAS

A chamada de sub-rotinas é muito frequente no uso da ferramenta de gerenciamento tratada neste artigo, já que todas as outras funções que usamos estão escritas, bastando apenas serem chamadas pelas sub-rotinas. A chamada de sub-rotinas começa com a expressão “sub” e logo após vem um bloco de comandos colocados entre chaves. Quando a execução atinge a chave de fechamento da sub-rotina, o controle do programa retorna para o primeiro comando [1].

3.3 PERL - ARGUMENTOS

Para ajudar as funções a executar suas tarefas, argumentos são passados para essas funções. A lista de argumentos passada para uma função é armazenada em uma variável *array*. Um função poderá ter quantos argumentos forem necessários e retornar um valor [1].

3.4 PERL – MÓDULOS

Um módulo em Perl são conjuntos de programas que são utilizados por *scripts*. Utilizando a ferramenta de gerenciamento abordada neste artigo, diversas vezes se fez necessário exportar um módulo para que as funções escritas no módulo exportado possam funcionar.

A estrutura para o uso e a instalação de um módulo em Perl é dividido em “::”. Por exemplo, caso queria utilizar o módulo principal da ferramenta de gerenciamento, exporto apenas o “Rex”, mas se quiser fazer o uso do módulo de *hardware* da ferramenta, utilizo “Rex::Hardware”, e assim acontece para todos os módulos dependentes do Rex.

4 OUTRAS ALTERNATIVAS PARA SOFTWARES DE GERENCIAMENTO

A ferramenta de gerenciamento exposta neste artigo é relativamente nova e está em processo de constante desenvolvimento. Existem outras ferramentas mais robustas e sólidas que desempenham o mesmo papel de diferentes formas. Este tópico tem o propósito de expor outras ferramentas de gerenciamento mais utilizadas e suas características que as diferenciam uma das outras. Pelo fato da ferramenta de gerenciamento utilizada neste artigo ser de código aberto, será demonstrado apenas ferramentas de código aberto.

4.1 CHEF

O Chef é uma ferramenta de gerenciamento feita em Ruby. A comunicação feita entre o servidor e o cliente é por HTTPS utilizando REST - uma interface que separa o cliente do servidor. O administrador poderá acessar seu servidor com as funções - chamadas pelos usuários da ferramenta de “receitas”. Existem muitas receitas já criadas, sendo assim, em muitos casos não é necessário a criação à partir de um ponto inicial, bastando apenas consultar o repositório do Chef para localizar uma função desejada.

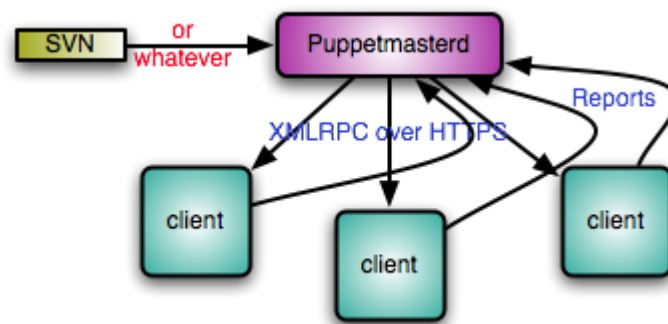
Você começa com um máquina “vazia”, instala os programas necessários para que o “chef-client” possa ser instalado e depois executa o cliente Chef. É considerado o melhor para os desenvolvedores, pois os mesmos podem criar e implementar seus próprios *scripts* [4].

4.2 PUPPET

O Puppet é uma ferramenta feita para gerenciar a configuração de sistemas Unix e Microsoft Windows declarativamente. Assim como o Chef ele também foi feito em Ruby.

Cada cliente conecta-se ao servidor periodicamente para fazer o *download* das últimas configurações e certificar se está sincronizado com a configuração atual. Depois, o cliente pode enviar uma resposta ao servidor indicando se algo precisa ser modificado. O servidor é chamado de “Puppet Master”. O Puppet sempre foi mais ligado ao administrador de sistemas do que com o desenvolvedor, pois ele é um conjunto de regras e não scripts desenvolvidos puramente em *Ruby* [5].

Figura 1 - Estrutura de acesso à ferramenta Puppet



Fonte: http://docs.puppetlabs.com/images/Puppet_Star.png

A figura 1 representa a estrutura do uso da ferramenta Puppet. Primeiramente, obtêm-se o Puppet através de um repositório SVN, ou por outro meio. Depois ocorre a instalação no servidor e a conexão é feita através de XMLRPC ou HTTPS com os clientes.

4.3 CFENGINE

O CFEngine é uma ferramenta de gerenciamento bastante leve desenvolvida em C. Assim como o Puppet e o Chef, ele suporta uma gama imensa de sistemas. Ele possui duas edições: uma edição da comunidade e uma edição corporativa comercial. Em sistemas Microsoft Windows, o CFEngine exige um ambiente Cygwin. Para desenvolver utilizando o CFEngine é necessária a instalação de ferramentas adicionais como: gcc, flex e bison [3].

Existem 2 tipos de diretórios para configuração usados pelo CFEngine no servidor. Os seus arquivos base são instalados no seu diretório de trabalho: “/var/cfengine” – por essa

estrutura de diretórios, onde estão localizados seus principais arquivos, é necessário em sistemas Windows a instalação de um emulador que simule um ambiente Unix. À medida que os arquivos de configuração são codificados, são colocados em um ponto de distribuição de configuração “/var/cfmasterfiles” em todos os servidores [3].

Para o recebimento de conexões de clientes remotos, é executado a sua aplicação de servidor, através do comando “start cf-server”. O CFEngine possui uma ferramenta própria para a geração de chaves, que servirão para a autenticação. Em cada cliente é criado uma par de chaves: pública e privada [3].

5 REX

O Rex é uma ferramenta de gerenciamento remoto desenvolvida em Perl, que pode ser instalada através de um módulo de Perl ou através dos repositórios de algumas distribuições Linux. É um projeto feito através da licença Apache – ela permite o uso e a distribuição do código fonte tanto no *software open source* como no proprietário [2].

Pelo fato de Perl ser uma linguagem interpretada, a ferramenta também pode ser utilizada por usuários de Windows, através do Strawberry Perl. Grande parte dos sistemas baseados em UNIX já vem com Perl por padrão.

Toda sua rotina e chamadas de funções são feitas em Perl, porém não é necessário desenvolver em Perl. A sua sintaxe é bastante clara, e grande parte das funções já estão dispostas no próprio site do projeto - que também é o principal guia de referência - bastando apenas adequá-las ao seu ambiente.

5.1 REX - DIFERENCIAL

O grande diferencial do Rex em relação a outras ferramentas de gerenciamento se dá pelo fato de não executar nenhum serviço no servidor. Todo o serviço é executado apenas na máquina que irá se comunicar com o servidor.

Por suas funções serem desenvolvidas em Perl, quando utilizado por clientes que utilizem sistemas baseados em Unix, não há necessidade de desenvolvedores instalarem algum programa para que ocorra a execução do código escrito, pois o interpretador Perl já é instalado por padrão.

5.2 REX - COMPATIBILIDADE

O Rex foi testado em várias distribuições, segue a lista retirada do site do desenvolvedor - no momento da realização deste artigo – contendo as distribuições Linux suportadas e as tarefas que podem ser executadas – que serão abordadas no decorrer deste artigo.

Figura 2 - Compatibilidade das tarefas em distribuições Linux

	CentOS 5	CentOS 6	Debian 5	Debian 6	Fedora	Gentoo	Mageia	OpenSUSE	RHEL 5	RHEL 6	ScientificLinux	Ubuntu 10.04 LTS	Ubuntu 11.04 LTS	Ubuntu 12.04 LTS
Cron														
File Transfer														
File Manipulation / Configuration Management														
Filesystem Operations														
Hardware Gathering														
Manage /etc/hosts														
Inventory														
Network Management														
IPTables														
Kernel Operations														
LVM														
Package Management														
Process Management														
Rsync														
Execute Commands														
Manage Services														
Manage Sysctl Entries														
Manage User- and Groupaccounts														
Virtualization			not tested			not tested	not tested	not tested	not tested	not tested	not tested			not tested

Fonte: <http://rexify.org/compatibility.html>

Além das distribuições Linux citadas na figura 2, o Rex também suporta: Solaris, BSD, MacOS e Windows (MacOS com MacPorts Perl e Windows com ActiveState Perl).

5.3 REX - ESTABELECIMENTO DA CONEXÃO

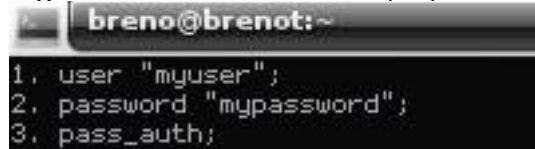
Você pode definir pequenas tarefas, tarefas em cadeia para lotes, ligá-los com os servidores ou grupos de servidores e executá-las facilmente em seu terminal. Mas para que isso ocorra é necessário que haja uma conexão segura entre ambas as partes. O Rex utiliza o SSH – um protocolo que permite a conexão com outro computador na rede de forma a permitir a execução de comandos de uma unidade remota – para fazer a conexão entre ambas as partes [2].

Existem 2 métodos para se autenticar utilizando o SSH. O primeiro método é a autenticação por senha. Nesse método passam-se dois parâmetros: o nome de usuário(login) e

a senha. O segundo método é a autenticação por chaves, através do processo de chave privada e chave pública.

Todos os comandos definidos pelo Rex são atribuídos a um único arquivo denominado “*Rexfile*”. Para “materializar” a situação do estabelecimento de conexão, abaixo está o código para as situações de autenticação por senha e autenticação por chaves.

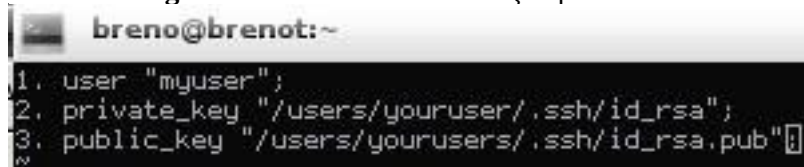
Figura 3 - Método de autenticação por senha



```
breno@brenot:~  
1. user "myuser";  
2. password "mypassword";  
3. pass_auth;
```

Fonte: Gerado pelo autor

Figura 4 - Método de autenticação por chaves



```
breno@brenot:~  
1. user "myuser";  
2. private_key "/users/youruser/.ssh/id_rsa";  
3. public_key "/users/yourusers/.ssh/id_rsa.pub";
```

Fonte: Gerado pelo autor

Na figura 3 é passado ao *Rexfile* na linha 1 o método que será usado, que é o método de senha, na linha 2 e 3 respectivamente estão o nome de usuário e a senha. A figura 4 é o método de autenticação por chaves. Na linha 1 é passado o nome de usuário, e nas linhas subsequentes, estão sendo passadas a chave privada e a chave pública respectivamente.

5.4 REX - IMPLANTAÇÃO

Como um passo inicial para a real utilização da ferramenta de gerenciamento, é necessário passar ao *Rexfile* o endereço do servidor a ser gerenciado, ou caso haja mais de um servidor, passar o endereço de múltiplos servidores.

Para esclarecer o processo de implantação, abaixo segue o código passado ao *Rexfile*, para a gerência de múltiplos servidores:

Figura 5 - Implantação em múltiplos servidores



```
breno@brenot:~  
1. user "myuser";  
2. password "mypassword";  
3. pass_auth;  
4. group myserver => "mywebserver", "mymailserver", "myfileserv" |
```

Fonte: Gerado pelo autor

Na figura 5, a linha 4 está sendo criado um grupo denominado “myserver” e dentro desse grupo, três servidores são alojados. No exemplo citado os servidores “mywebserver”, “mymailserver” e “myfileserver”.

Percebe-se também que o *Rexfile* está utilizando o método de autenticação por senha – esse método será exposto em todas as situações neste artigo.

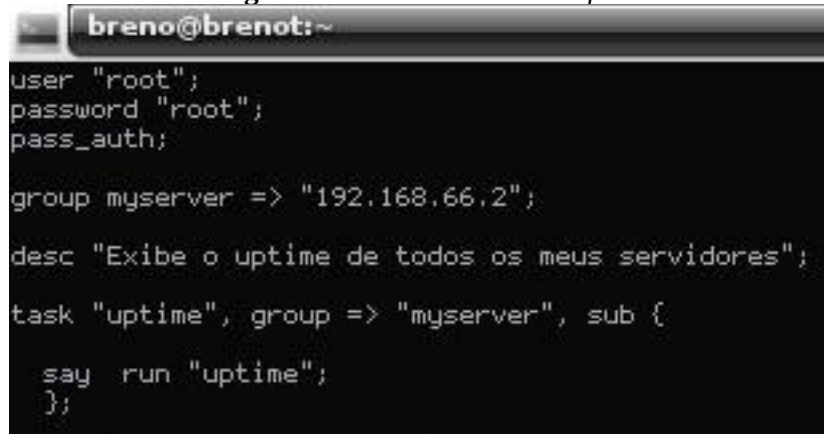
5.5 REX – UTILIZAÇÃO DOS SERVIÇOS DE GERENCIAMENTO

À partir deste tópico, tudo relacionado à ferramenta de gerenciamento será em relação aos serviços suportados, anteriormente listados na figura 2.

5.5.1 REX – UPTIME

Como primeiro passo, foi testado o comando de *uptime* do sistema. Visto que é a tarefa mais simples a ser feita, e não requer requisitos de compatibilidade.

Figura 6 - Estrutura de um *Rexfile*



```
user "root";
password "root";
pass_auth;

group myserver => "192.168.66.2";

desc "Exibe o uptime de todos os meus servidores";
task "uptime", group => "myserver", sub {
  say run "uptime";
};
```

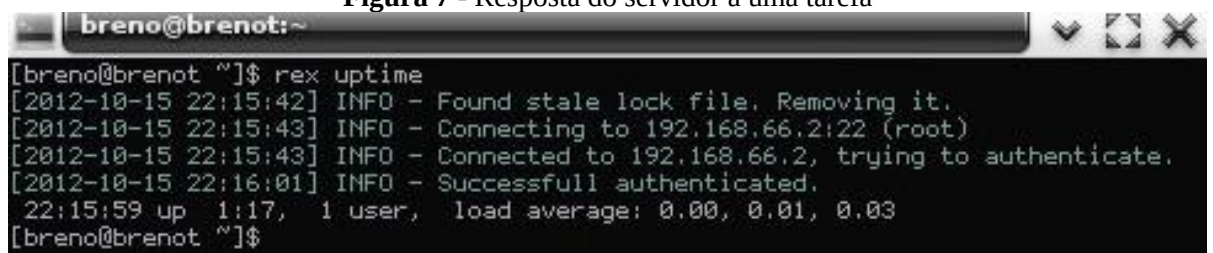
Fonte: Gerado pelo autor

Através da Figura 6, são apresentados mais três conceitos. A função “desc” permite descrever o que cada comando faz – Isso é bastante útil quando se quer ter um controle sobre todas as tarefas, a ferramenta possibilita fazer a listagem de todas as tarefas registradas no *Rexfile* através do comando “Rex -T”. A função task define uma nova tarefa com o nome de “uptime” e o comando “group => “myserver”” estabelece que a tarefa *uptime* será executada no grupo de servidores “myserver”. Ao final é escrito “say run “uptime”” que será

interpretado no meu servidor – O interpretador de comandos(shell) em sistemas baseados em UNIX é uma ferramenta bastante robusta, através dela, os comandos passados pelo usuário são interpretados para que o sistema operacional execute a tarefa solicitada e mostre na tela.

O comando “uptime” mostra na tela o tempo em que o computador está ligado, além de outras informações, como o número de usuários logados no sistema. Ele foi tomado como exemplo, mas poderia ser qualquer outro comando da base de dados do interpretador de comando.

Figura 7 - Resposta do servidor a uma tarefa



```
breno@brenot:~$ rex uptime
[2012-10-15 22:15:42] INFO - Found stale lock file. Removing it.
[2012-10-15 22:15:43] INFO - Connecting to 192.168.66.2:22 (root)
[2012-10-15 22:15:43] INFO - Connected to 192.168.66.2, trying to authenticate.
[2012-10-15 22:16:01] INFO - Successfull authenticated.
22:15:59 up 1:17, 1 user, load average: 0.00, 0.01, 0.03
breno@brenot ~]$
```

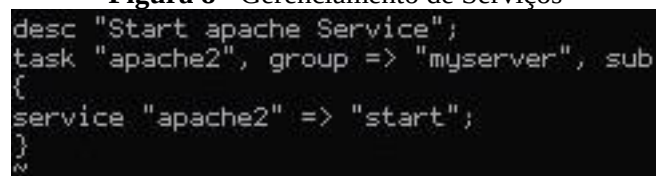
Fonte: Gerado pelo autor

Na figura 7, podemos verificar que a autenticação e a execução da tarefa “uptime” foi realizada com sucesso, retornando que o sistema está ligado a uma hora e dezessete minutos e que no momento está com 1 usuário logado no sistema.

5.5.2 REX – Gerenciando os Serviços

O processo de gerenciamento de serviços permite que você manipule todos os serviços do seu sistema. Existem 5 principais tarefas relacionadas ao gerenciamento de serviços: inicializar, reinicializar, parar, recarregar e checar o *status* do serviço.

Figura 8 - Gerenciamento de Serviços



```
desc "Start apache Service";
task "apache2", group => "myserver", sub
{
  service "apache2" => "start";
}
~
```

Fonte: Gerado pelo autor

Na figura 8, é apresentada uma nova função denominada “service”. A função “service” aceita 2 parâmetros. O primeiro é o nome do serviço e o segundo é a ação desejada. Na figura

8 esses parâmetros são representados respectivamente por “apache2” e por “start”, isso quer dizer que a intenção da função é de inicializar o serviço do apache.

Figura 9 - Inicialização do Apache

```
[breno@brenot ~]$ rex apache2
[2012-10-16 01:01:04] INFO - Connecting to 192.168.66.4:22 (root)
[2012-10-16 01:01:04] INFO - Connected to 192.168.66.4, trying to authenticate.
[2012-10-16 01:01:04] INFO - Successfull authenticated.
[2012-10-16 01:01:07] INFO - Service apache2 started.
[breno@brenot ~]$
```

Fonte: Gerado pelo autor

Na figura 9, é obtido o resultado do *Rexfile* da figura anterior, onde iniciamos o serviço do apache. Somos informados que a autenticação ocorreu com sucesso e que o serviço do apache foi inicializado.

Os outros parâmetros da função possuem o mesmo escopo, bastando substituir pela ação desejada.

5.5.3 REX – Informação de *Hardware*

Um módulo bastante utilizado é o de informações de *hardware*, nele é armazenado em um *hash* – Uma tabela hash é uma coleção não-ordenada de pares chave-valor. Elementos de uma tabela hash são acessados com um *string* conhecido como uma chave. Cada chave deve ser única. Tabelas *hash* às vezes são chamadas de arrays associativos, porque elas definem associações entre chaves e seus valores. Uma variável que armazena um *hash* é iniciada com um sinal de porcentagem(%) [1].

Figura 10 - Uso do módulo de informação de *hardware*

```
use Rex::Hardware;
use Data::Dumper;
task "get-info", group => "myserver", sub {
    my %hw_info = Rex::Hardware->get(qw/ All /);
    say Dumper(\%hw_info);

    say "IP of eth2 is: " . $hw_info{Network}->{networkconfiguration}->{eth2}->{ip};

    my $hostname = run "hostname";
    say "The hostname is: $hostname";
};
```

Fonte: Gerado pelo autor

Na figura 10, são apresentados novos conceitos da própria linguagem. O código passado ao *Rexfile* começa com a expressão “use”. Essa expressão indica que algum módulo está sendo passado para o *Rexfile*. Estes módulos são responsáveis pela obtenção da informação de *hardware* do computador e por pegar esses dados e mostrá-los na tela. São representados por *Rex::Hardware* e *Data::Dumper* respectivamente.

Figura 11 – Exibição de informações da memória

```
'Memory' => {
  'shared' => '0',
  'buffers' => '3',
  'free' => '415',
  'used' => '87',
  'total' => '502',
  'cached' => '42'
```

Fonte: Gerado pelo autor

Como exemplo, a figura 11 representa a resposta com a correspondente informação da memória. O módulo de detecção de *hardware* é dividido em 5 partes: Módulo de memória, Kernel, Swap, *Host* e *Network*. A informação pode ser obtida separadamente ou em seu total.

Para o módulo “Network” é passado como parâmetro o nome da interface, mostrada com “eth2” para a obtenção do IP da interface.

5.5.4 REX – Firewall

O módulo de Firewall é usado para manipular as regras básicas de iptables - na maioria das distribuições baseadas em Unix o iptables está incorporado ao Kernel.

Por regras básicas entende-se:

- Abrir e fechar portas
- Redirecionamento de portas
- Habilitar *gateway*
- Definir regras padrão
- Listar regras
- Apagar regras

Figura 12 - Função de fechar a porta pelo módulo do Iptables

```
use Rex::Commands::Iptables;
task "firewall", group => "myserver", sub {
    close_port [22, 80] => { dev => "eth2", };
};
```

Fonte: Gerado pelo autor

Na figura 12, é representada a ação de fechar a porta. Como primeiro passo, o módulo do iptables é exportado. A Função “close_port” está recebendo como parâmetro um *array* de portas e a interface a qual a função será aplicada. O código passado ao *Rexfile* está dizendo a ferramenta que rejeite pacotes destinados a porta 22 e 80.

O módulo de Firewall é bastante utilizado em cadeia (conjunto de comandos).

O código passado ao *Rexfile*, no exemplo da figura 12, seria equivalente a executar manualmente no iptables as seguintes instruções: “iptables -t filter -A input -i eth2 -p tcp -m tcp -j DROP 22”. O código implicaria no acesso bloqueado ao SSH.

A mesma estrutura aplica-se a função de abrir portas. Bastando apenas substituir a função “close_port” pela função “open_port”.

Figura 13 - Redirecionamento de porta

```
task "redirects", group => "myserver", sub {
    redirect_port 80 => {
        to => 10080,
        dev => "eth0",
    };
};
```

Fonte: Gerado pelo autor

Na figura 13, é demonstrada a estrutura da função do redirecionamento de portas. A função “redirect_port” recebe três parâmetros. O primeiro parâmetro é a porta em que se quer fazer o redirecionamento. O segundo parâmetro é a porta em que vai receber o redirecionamento. E o terceiro parâmetro é a interface em que o redirecionamento vai atuar. A função “redirect_port” demonstrada na figura 13 seria equivalente a executar no iptables as seguintes instruções: “iptables -t nat -I PREROUTING -i eth0 -p tcp -m tcp -dport 80 -j REDIRECT -to-ports 10080”.

Para a listagem e a remoção das regras utilizando a ferramenta de gerenciamento, são utilizadas as funções “iptables_list” e “iptables_clear” respectivamente.

Devido toda a regra do iptables está incorporada ao Kernel do sistema, pode ser escrito no *Rexfile* as regras escritas na forma primitiva, manualmente, como anteriormente citado. O interpretador irá ler as instruções da mesma forma.

5.5.5 REX – Alteração do Arquivo *Hosts*

Com esse módulo é possível adicionar, apagar e exibir entradas no arquivo “/etc/hosts” – arquivo que define um endereço IP a determinado host. As funções são representadas por “create_host”, “delete_host” e “get_host” respectivamente.

Na função adicionar são recebidos 3 parâmetros: o nome do host, o IP e a aliases em que o IP vai receber – o nome do host que quer ser acessado.

5.5.6 REX – Retornar Nome do Sistema Operacional

Esse módulo é bem simples, pois só recebe um valor escalar. Existe outra função semelhante que exibe a versão do sistema instalado. O código passado ao *Rexfile* para ambas as funções é representado na figura 14 abaixo:

Figura 14 - Função para exibir nome do Sistema Operacional

```
task "get-os", group => "myserver", sub {  
  say get_operating_system();  
  say operating_system_version();  
};
```

Fonte: Gerado pelo autor

5.5.7 REX – Instalação de Pacotes

O módulo de instalação de pacotes só tem suporte para as distribuições Linux: CentOS, OpenSuSE e Debian. Ele pode receber apenas um valor ou um *array* contendo mais de um programa a ser instalado.

Figura 15 - Função para instalação de pacotes

```
task "instalar", group => "myserver", sub {  
  install package => "perl";  
};
```

Fonte: Gerado pelo autor

A figura 15 contém o código passado ao *Rexfile* em que a função “install” está instalando por via do gerenciador de pacotes o programa “perl”.

Seguindo a mesma estrutura, também existe a função “remove”, que remove um pacote ou uma lista de pacotes a sua escolha.

Para a atualização da base de dados, existe a função “update_package_db”, que não requer parâmetro algum. Em um sistema Debian isso seria equivalente ao comando “apt-get update” e no CentOS ao comando “yum makecache” [2].

5.5.8 REX – Instalação Arquivos

O módulo de instalação de programas do Rex requer que exista um arquivo instalado no sistema em que a ferramenta está sendo executada. O próximo passo será o destino da instalação, seguido das permissões – sistemas baseados em Unix possuem um sistema de permissões que definem os direitos dos usuários e dos programas sobre todos os arquivos e diretórios, tendo o usuário “root” acesso a todos os arquivos.

Sistemas baseados em Unix consideram tudo como arquivo. Até mesmo os diretórios são considerados arquivos.

Figura 16 - Estrutura da função de instalação de arquivos

```
task "arquivo", "server01", sub {
  install file => "/etc/httpd/apache2.conf", {
    source      => "/export/files/etc/httpd/apache2.conf",
    owner       => "root",
    group       => "root",
    mode        => 644,
  };
};
```

Fonte: Gerado pelo autor

Na figura 16 observamos a existência de 4 parâmetros na função “install”. O primeiro parâmetro é onde o arquivo será instalado, o segundo é o arquivo que será instalado, seguido das permissões para o dono do arquivo e do grupo – quando é setada uma permissão em sistemas baseados em UNIX, essa permissão é imposta ao usuário e ao grupo.

5.5.9 REX – Checagem de Portas

O módulo de checagem de portas reporta ao administrador do sistema se uma porta está aberta por meio de expressão condicional.

A estrutura da função de checagem de portas é: “if (is_port_open)(\$remote_host, \$port)”. Percebe-se que a função recebe 2 parâmetros em valor escalar – tipos de variáveis em Perl são tratados no tópico 3.3 – o primeiro parâmetro é o endereço do servidor ao qual se quer fazer a checagem e o segundo parâmetro é a porta que será verificada.

5.5.10 REX – Gerenciando Usuários

O módulo de gerenciamento de usuários é o que possui o maior número de parâmetros, pois em uma distribuição baseada em UNIX, para adicionar usuários, se faz necessário. Além da criação, também é possível a alteração desses parâmetros em usuários já registrados, assim como o retorno do seu UID - o UID representa a identificação do usuário em um número para fins de monitoramento - o retorno de quais grupos o usuário participa, uma lista de usuários cadastrados no sistema, a remoção de determinado usuário, a criação de grupos, o retorno do GID - o GID representa a identificação do grupo em um número para fins de monitoramento - pode também retornar informações sobre determinado grupo e a remoção do grupo.

Figura 17 - Utilização do módulo para criação de usuários

```
use Rex::Commands::User;  
task "create-user", "remoteserver", sub {  
    create_user "root",  
        uid => 0,  
        home => '/root',  
        comment => 'Root Account',  
        expire => '2011-05-30',  
        groups => ['root', '...'],  
        password => 'blahblah',  
        system => 1,  
};
```

Fonte: Gerado pelo autor

Na figura 17, é representado o código passado ao *Rexfile* para a criação de um usuário. Todos os argumentos recebem um valor escalar, exceto o argumento “groups”, pois um usuário pode fazer parte de mais de um grupo. Para que as opções possam ser executadas, se faz necessário exportar o módulo de comandos para criação de um usuário chamado de “Rex::Commands::User”.

As outras funções que englobam a manipulação de uma conta de usuário são funções que ao serem acionadas retornam algum dos valores passados como argumentos para a criação do usuário - e também a função que remove o usuário - por fim as funções que interagem com os grupos.

5.5.11 REX – Download de Arquivos

A ferramenta de gerenciamento Rex também permite a ação de fazer o *download* de determinado arquivo. A transferência pode ocorrer utilizando os seguintes protocolos: HTTP, SFTP e FTP.

Através dos protocolos de FTP é feita uma transferência local. Nessa situação pode ser feita políticas de *backup* utilizando todo o poder da linguagem Perl, para criar condições e tempos necessários para ocorrência do backup. Esse backup poderá ocorrer também como uma segurança extra, pois pode ser feito a transferência para computadores “normais”, ao qual que não se ocupam da função de um servidor de arquivos.

Por meio da utilização do protocolo HTTP, pode ser feita a transferência do conteúdo da página e de arquivos hospedados no servidor WEB. Um exemplo de utilização no dia-a-dia dessa função seria a de copiar toda a estrutura de um site da intranet corporativa para fins de backup ou alteração.

Em todos protocolos as funções passadas ao *Rexfile* requerem dois parâmetros. O primeiro parâmetro é o arquivo em que se deseja baixar. O segundo parâmetro é o nome do arquivo que será salvo no computador que está sendo executado a ferramenta. O bloco de códigos é representado na figura abaixo:

Figura 18 - Estrutura do módulo de download

```
task "download", group => "myserver", sub {
  download "/root/teste", "teste";
};
```

Fonte: Gerado pelo autor

Na figura 18, o código passado ao *Rexfile* resulta na transferência do arquivo de nome “teste”, localizado em “/root”, que será salvo com o mesmo nome.

5.5.12 REX – Particionamento

Através do módulo de particionamento é possível particionar discos rígidos e limpar as partições da unidade.

O módulo de particionamento - juntamente com o módulo de gerenciamento de usuário - possui uma grande quantidade de parâmetros. Toda mudança efetuada pela função de particionamento tem efeito no arquivo “/etc/fstab” - arquivo de configuração do sistema de arquivos. É por esse arquivo que se diz ao sistema de que modo os discos e partições serão inicializados.

Figura 19 - Estrutura do módulo de particionamento

```

use Rex::Commands::Partition;
task "particionar", group => "myserver", sub {
    partition "swap",
        fstype => "swap",
        type   => "logical",
        ondisk => "sda",
        size   => 100;
};

```

Fonte: Gerado pelo autor

Na figura 19, está sendo representada a criação de uma partição. Essa partição é do tipo “swap” - memória virtual - e é uma partição lógica – podem haver de uma a quatro partições lógicas em um disco. Os outros 2 parâmetros são o disco em que será feito o particionamento e o tamanho da partição. Respectivamente representados por “sda” e “100”.

5.5.13 REX – Virtualização

Através do módulo de virtualização é possível a criação de máquinas virtuais(KVM). Uma máquina virtual KVM é um *hypervisor* – gerenciador de máquinas virtuais - embutido no Kernel do Linux. Para começar a usar o *hypervisor* é necessário apenas carregar os módulos apropriados.

Assim como *softwares* para virtualização como VMware e VirtualBox que possuem um robusto gerenciamento gráfico, o módulo de virtualização também permite através de suas funções o gerenciamento de várias ações depois que a máquina é criada: inicialização, remoção, desligamento, reinicialização e edição.

Todas as ações recebem um único argumento que é o nome da máquina virtual, exceto a função de edição que permite passar novos valores ou alterá-los. Esses valores são dados do *hardware* da máquina virtual como: quantidade máxima de memória, número de processadores, arquitetura, tipo de interface de rede e outros valores mais específicos.

5.5.14 REX - Exibindo Log de Múltiplos Servidores

Através do módulo de exibição de LOGS é possível o monitoramento de um servidor ou de múltiplos servidores ao mesmo tempo – para se monitorar múltiplos servidores adiciona-se um recurso no próprio *Rexfile* chamado de paralelismo.

Figura 20 - Exibição de LOGS para um grupo de servidores

```
use Rex::Commands::Tail;

parallelism 20;

desc "Tail Frontend Apache Access Logs";
task "tail-fe", group => "myserver", sub {
    tail "/opt/instances/httpd/fe-i01/logs/access_log";
cd
};
```

Fonte: Gerado pelo autor

Na figura 20, é representada a tarefa de exibição de LOGS utilizando paralelismo. Será exibido o arquivo de LOG da requisição de acesso do servidor Apache de todos os servidores do grupo “myserver”.

5.5.15 REX – Banco de Dados

Uma linguagem de programação se conecta a, e interage com, bancos de dados relacionais através de uma *interface - software* que facilita as comunicações entre um sistema de gerenciamento de banco de dados e um programa. A interface mais popular usada por programadores Perl é a Perl Database Interface (DBI) - uma interface independente de banco de dados que permite à Perl interagir com bando de dados [1].

Este módulo permite um simples acesso ao banco de dados. As operações permitidas são: selecionar, apagar, inserir e atualizar.

Os argumentos passados para as 4 funções suportadas são: o nome da tabela, o campo a ser modificado ou selecionado e a condição.

Figura 21 - Inserção de valores através do módulo de Banco de Dados

```
db insert => "table", {
    campo1 => "valor1",
    campo2 => "valor2",
    campo3 => 5,
};
```

Fonte: Gerado pelo autor

A figura 21 representa a estrutura da função “insert” do módulo de banco de dados. Os campos da tabela “table” foram atualizados para novos valores. As outras funções também possuem a mesma estrutura.

6 CONSIDERAÇÕES FINAIS

Ao final deste artigo, pode-se concluir que as ferramentas de gerenciamento possuem um papel de grande importância para os administradores de sistemas, apesar de não serem tratadas como ferramentas essenciais. Elas tem o principal dever de centralizar as operações, primando pela organização e pela automação.

O grande objetivo das empresas para a adoção de uma ferramenta de gerenciamento é o seu uso em múltiplos servidores, com o intuito de padronizar instalações, arquivos, contas de usuários, grupos, compartilhamentos, regras de segurança e acesso. Tudo isso com o uso adequado da própria linguagem de desenvolvimento da ferramenta pode-se construir um ambiente bastante padronizado, diminuindo consideravelmente o risco à falhas. Para se ter um exemplo, em uma situação onde o UID do usuário é alterado acidentalmente, é possível reverter para a situação inicial em uma única instrução pela ferramenta de gerenciamento.

Em relação à ferramenta específica abordada neste artigo, ela é relativamente nova e ainda está em constante processo de desenvolvimento. Com base nisso, a ferramenta precisa passar por um processo de testes em larga escala. É algo arriscado para grandes e médias empresas que necessitam de algo sólido.

Para o futuro é esperado que as ferramentas de gerenciamento suportem mais sistemas, garantindo assim uma maior compatibilidade. Essa grande gama de sistemas suportados será o grande atrativo dessas ferramentas. Imagina-se que assim ocorra uma maior centralização das operações e que tudo seja mais simplificado, já que um único arquivo poderá ser englobado todos os sistemas existentes que necessitem de um gerenciamento remoto.

REFERÊNCIAS

[1] DEITEL, Harvey et al. **Como Programar**. Rio de Janeiro: Bookman, 2001.

[2] GEHRING, Jan. **Rex Documentation**. Disponível em: <<http://rexify.org/howtos/index.html>>. Acesso em: 15 out. 2012.

[3] IBM. **Automatizando o Gerenciamento de infraestrutura com CFEngine**. Disponível em: <<http://www.ibm.com/developerworks/br/library/os-cfengine1/index.html>>. Acesso em: 25 out. 2012.

[4] Opscode. **Chef Documentation**. Disponível em:
<<http://wiki.opscode.com/display/chef/Documentation>>. Acesso em: 20 out. 2012.

[5] PUPPET LABS. **Puppet Labs Documentation**. Disponível em:
<<http://docs.puppetlabs.com>>. Acesso em: 20 out. 2012.