SENADO FEDERAL
Secretaria Especial do Interlegis - SINTER
Subsecretaria de Tecnologia da Informação - SSTIN

Relatório IV

Manual de Instalação e Transferência de Tecnologia

Consultor: Felipe Luis de Souza Vieira (felipe.tio@gmail.com)

Produto: Metabus

Contrato:2008/000750

Brasília - DF

Novembro / 2008

Sumário

1 Introdução

Este documento tem por objetivo principal apresentar as instruções de instalação e configuração do produto Metabus além de informações para obter o código fonte e uma breve introdução à ferramenta utilizada para auxiliar na tarefa de instalação do sistema em diferentes servidores. Ainda, são apresentadas as saídas dos scripts de instalação para ilustrar os procedimentos realizados.

Este manual de instalação é específico para sistemas Linux, os comandos aqui apresentados foram efetuados na distribuição Ubuntu 8.04 LTS.

1.1. Como obter o Metabus

O Metabus pode ser obtido a partir do Subversion associado a este projeto. O procedimento é simples e é melhor detalhado em http://code.google.com/p/metabus/source/checkout . Porém é necessário ter um cliente Subversion instalado. Para isso basta executar:

```
sudo apt-get install subversion
```

A partir daí os procedimentos descritos no link acima são explicativos o suficiente.

1.2. Instalação com o Fabric

Fabric é uma ferramenta Python que auxilia a instalação e configuração de sistemas em servidores. Essa ferramenta é capaz de fazer upload de arquivos e executar comandos em servidores a fim de instalar um sistema remotamente em um ou diversos servidores.

Os comandos de instalação são organizados em funções python dentro do arquivo 'fabfile.py' Este arquivo possui a seguinte estrutura:

```
set(
  project = 'mbclient',
  project_log_path = '/home/metabus/var/log/$(project)',
  (...)
  fab_user  = 'metabus',
  fab_hosts  = ['127.0.0.1'],
  (...)
)
def deploy():
  "Deploy the project to the production server."
  (...)
def build():
  "Build the project. Prepare the packages to deploy."
  (...)
def clean():
  "Remove the build directory."
  (...)
```

Cada método definido neste arquivo é uma etapa da instalação. No início uma tupla 'set' permite configurar as variáveis necessárias para o script. Veja que, nesta tupla, indicamos na variável fab_hosts os servidores nos quais os comandos remotos devem ser executados. E na variável fab_user, o usuário que executará esses comandos nos servidores. Pode-se também criar variavéis próprias e inseri-las na tupla 'set'(como a variável project_log_path).

Além de poder utilizar todos os comandos python esta ferramenta fornece quatro principais métodos, são eles:

- local: Executa comandos na máquina local com o usuário que roda o script. Ex:

```
local('mkdir -p $(build_path)/$(project)/')
```

- run: Mesma funcionalidade do *local*, entretanto executa os comandos nas máquinas remotas (indicadas na variável fab_hosts) com o usuário especificado na variável fab_user. Ex:

```
run('cd $(destination)/; tar xf $(project_build); rm $(project_build)')
```

- sudo: Igual ao run, porém executa o comando remotamente como super-usuário Ex:

```
sudo('/etc/init.d/apache2 restart')
```

- put: Envia um arquivo para as máquinas remotas. Ex:

```
put('$(build_path)/$(project_build)', '$(destination)/$(project_build)')
```

Quando uma variável é utilizada dentro de um comando, isso significa que o Fabric irá parsear a string, para só então executar tal comando.

O principal motivo pelo qual decidiu-se pelo uso dessa ferramenta é organizar os arquivos dos projetos para que estes fiquem dispostos da mesma forma utilizada pelo servidor de produção, enviando os arquivos dentro de um pacote tar.gz para os servidores, descompactando e movendo para o local correto, além de reiniciar o servidor web para recarregar o projeto.

2 Pré Requisitos

Para realizar a instalação do Metabus com sucesso as seguintes dependências devem ser instaladas:

2.1. Pacotes Debian

Os pacotes listados abaixo podem ser instalados com o comando: *apt-get*.

- python2.5
- python-psycopg2
- postgresql
- apache2
- libapache2-mod-wsgi
- buid_essencial

2.2. Python Eggs

Os módulos listados abaixo podem ser instalados com o comando: *easy_install*. Para instalar este comando pode-se instalar o pacote: python-setuptolls.

- fabric
- gettext
- simplejson

2.3. Django

O processo de instalação do Django é simples, primeiramente, deve-se obter a versão 1.0 do site do próprio framework.

http://www.djangoproject.com/download/1.0/tarball/

Após baixar o Django descompacte o arquivo com o comando:

```
tar xzf Django-1.0.tar.gz
```

Entre no diretório criado pelo comando anterior e execute o comando abaixo, que instalará o Django em seu diretório site-packages da instalação do Python.

```
sudo python setup.py install
```

3 Instalação do Metabus

Para instalar o Metabus em uma máquina é necessário ter todas as depedências instaladas. As instruções que seguem visam preparar o ambiente e rodar os scripts necessários para copiar todos os arquivos para os lugares corretos.

Em ambos projetos (mbserver ou mbcliente) podemos distinguir os arquivos de configuração que ficam no diretório deploy (dentro do diretório do projeto, exemplo: mbserver/deploy/). Este diretório contém (além do script de instalação automática) os arquivos de configuração, que definem as propriedades e o funcionamento dos softwares que acompanham este sistema, como por exemplo: Apache, Lighttpd e Postgresql. Estes arquivos podem eventualmente ser alterados, entretanto os scripts de instalação apresentados nas próximas seções executarão os comandos necessários para mover estes arquivos para os servidores e diretórios adequados reiniciando esses softwares, caso seja necessário.

3.1. Ambiente

Para iniciar a instalação do sistema é necessário criar um usuário metabus, de maneira que todos os arquivos do projetos ficarão dentro do diretório home deste usuário. Para criar um usuário metabus basta executar:

```
sudo adduser metabus
```

Após criar o novo usuário é necessário dar privilégios para que este possa executar comandos como administrador do sistema. Para isso deve-se editar o arquivo */etc/sudoers* e adicionar o metabus na lista de sudoers como no exemplo:

```
# User privilege specification
root  ALL=(ALL) ALL
metabus ALL=(ALL) ALL
```

3.2. Instalação do Servidor (mbserver)

Para instalar um servidor é necessário já ter um usuário criado (como especificado na seção acima). Para realizar a instalação automática é necessário também ter acesso através de ssh à essa máquina. Para instalar ssh na máquina que rodará o servidor pode-se executar:

```
sudo apt-get install openssh-server
```

Para configurar a instalação temos também que editar o script localizado na pasta deploy do projeto. Para inserir a lista de IPs das máquinas em que este sistema será instalado.

No exemplo abaixo vamos configurar a instalação para copiar os arquivos para a mesma máquina que está executando a instalação (localhost). Para isto, editamos a variável *fab_hosts* no arquivo *mbclient/deploy/fabfile.py* como mostra o código abaixo:

```
set(
  project = 'mbserver',
  (...)
  fab_user  = 'metabus',
  fab_hosts  = ['127.0.0.1'], #localhost
  destination = '/home/metabus/$(project)',
)
(...)
```

Agora com as depedências instaladas, ambiente criado e arquivos de configuração editados podemos executar o script de instalação. O script de instalação pode ser analisado digitando o comando *fab* no diretório *mbserver/deploy/* , como podemos ver abaixo:

```
Available commands are:
 build  : Build the project. Prepare the packages to deploy.
 clean  : Remove the build directory.
 deploy  : Deploy the project to the production server.
 help   : Display Fabric usage help, or help for a given command.
 license : Display the Fabric distribution license text.
 list   : Display a list of commands with descriptions.
 set   : Set a Fabric variable.
 shell  : Start an interactive shell connection to the specified hosts.
 warranty : Display warranty information for the Fabric software.
```

Dentre estes comandos apenas três deles são diretamente responsáveis pela instalação do servidor, são eles:

- **deploy**: responsável por instalar o servidor de busca nos IPs determinados. Esta instalação é feita enviando os pacotes tar.gz (criados pelo comando build) para os servidores, descompactando e movendo para os locais adequados.
- **build**: seleciona e agrupa os arquivos do servidor para serem utilizados pelo comando deploy. Este comando cria 2 pacotes: mbserver.tar.gz (arquivos django do sistema) e mbserver_config (arquivos de configuração do Apache)
- **clean**: limpa os pacotes e diretórios de instalações antigas criados pelo comando build.

Portanto para instalar o mbserver e copiar os arquivos para as pastas corretas basta executar:

```
fab deploy
```

Este comando além de copiar os arquivos para as pastas reinicia o sevidor web antes de finalizar a instalação, de modo que o serviço do Metabus (mbserver) já possa ser acessado através do servidor web. Abaixo é exemplificado uma execução desta instalação:

```
Running deploy...
[localhost] run: mkdir -p /tmp/build/mbserver/
[localhost] run: cp -R ../../mbserver/* /tmp/build/mbserver/
[localhost] run: find /tmp/build -name *.pyc -delete
[localhost] run: find /tmp/build -path *.svn* -delete
[localhost] run: cd /tmp/build/mbserver/; tar -zcf ../mbserver.tar.gz .; cd ../../
[localhost] run: rm -fR /tmp/build/mbserver/
Logging into the following hosts as metabus:
  127.0.0.1
Password for metabus@127.0.0.1:
[127.0.0.1] run: mkdir -p /home/metabus/mbserver
[127.0.0.1] put: /tmp/build/mbserver.tar.gz -> /home/metabus/mbserver/mbserver.tar.gz
[127.0.0.1] run: cd /home/metabus/mbserver; tar xf mbserver.tar.gz; rm mbserver.tar.gz
[127.0.0.1] sudo: /etc/init.d/apache2 restart
[127.0.0.1] out: * Restarting web server apache2
[127.0.0.1] out:  ...done.
Done.
```

3.3. Instalação do Cliente (mbclient)

A instalação do Cliente - módulo que contém a interface com o usuário - é muito similar à instalação dos servidores, de forma que, também é necessário já ter um usuário criado e acesso ssh à máquina em que se deseja efetuar a instalação.

Para configurar a instalação temos também que editar o script localizado na pasta deploy do projeto. Entretanto, diferentemente da instalação dos servidores o cliente deve ser instalado em apenas uma máquina. Para isto, editamos a variável *fab_hosts* no arquivo *mbclient/deploy/fabfile.py* como mostra o código abaixo:

```
set(
  project = 'mbclient',
  (...)
  fab_user  = 'metabus',
  fab_hosts  = ['127.0.0.1'], #localhost
  destination = '/home/metabus/$(project)',
)
(...)
```

O script de instalação do cliente tem a mesma estrutura do script de instalação dos servidores. Portanto se digitarmos o comando *fab* no diretório *mbclient/deploy/* no terminal teremos a mesma saída da seção anterior :

```
Available commands are:
 build  : Build the project. Prepare the packages to deploy.
 clean  : Remove the build directory.
 deploy  : Deploy the project to the production server.
 help   : Display Fabric usage help, or help for a given command.
 license : Display the Fabric distribution license text.
 list   : Display a list of commands with descriptions.
 set   : Set a Fabric variable.
 shell  : Start an interactive shell connection to the specified hosts.
 warranty : Display warranty information for the Fabric software.
```

Da mesma forma apenas três opções são responsáveis pela instalação da interface gráfica, são elas:

- **deploy**: responsável por instalar o projeto no IP determinado (executa o comando build). Esta instalação é feita enviando os pacotes tar.gz (criados pelo comando build) para o servidor, descompactando e movendo para os locais adequados.

- **build**: divide, seleciona e agrupa os arquivos do projeto para serem utilizados pelo comando deploy. Este comando cria 3 pacotes: mbclient.tar.gz (arquivos django do sistema), mbclient_media.tar.gz (arquivos estáticos utilizado pelo sistema) e mbserver_config (arquivos de configuração do Apache, Lighttpd e Postgresql)
- **clean**: limpa os pacotes e diretórios de instalações antigas criados pelo comando build.

Portanto para instalar o cliente do sistema e copiar os arquivos para as pastas corretas basta executar:

```
fab deploy
```

Este comando além de copiar os arquivos para as pastas reinicia os sevidores antes de finalizar a instalação, de modo que a interface gráfica já possa ser acessada pelos usuários através do browser. Na próxima página é exemplificado uma execução desta instalação:

```
Running deploy...
[localhost] run: cd ../../mbserver/deploy; fab build; cd -;
 Running build...
 [localhost] run: mkdir -p /tmp/build/mbserver/
 [localhost] run: cp -R ../../mbserver/* /tmp/build/mbserver/
 [localhost] run: find /tmp/build -name *.pyc -delete
 [localhost] run: find /tmp/build -path *.svn* -delete
 [localhost] run: cd /tmp/build/mbserver/; tar -zcf ../mbserver.tar.gz .; cd ../../
 [localhost] run: rm -fR /tmp/build/mbserver/
 Done.
/home/felipevieira/metabus/mbclient/deploy
[localhost] run: mkdir -p /tmp/build/mbclient/
[localhost] run: cp -R ../../mbclient/* /tmp/build/mbclient/
[localhost] run: find /tmp/build -name *.pyc -delete
[localhost] run: find /tmp/build -path *.svn* -delete
[localhost] run: cd /tmp/build/mbclient/; tar -zcf ../mbclient_media.tar.gz ui/media;
cd ../../
[localhost] run: rm -fR /tmp/build/mbclient/ui/media
[localhost] run: cd /tmp/build/mbclient/; tar -zcf ../mbclient.tar.gz .; cd ../../
[localhost] run: rm -fR /tmp/build/mbclient/
Logging into the following hosts as metabus:
  127.0.0.1
Password for metabus@127.0.0.1:
[127.0.0.1] run: mkdir -p /home/metabus/mbclient
[127.0.0.1] run: mkdir -p /home/metabus/mbserver
[127.0.0.1] put: /tmp/build/mbclient.tar.gz -> /home/metabus/mbclient/mbclient.tar.gz
[127.0.0.1] put: /tmp/build/mbclient_media.tar.gz ->
/home/metabus/mbclient/mbclient_media.tar.gz
[127.0.0.1] put: /tmp/build/mbserver.tar.gz -> /home/metabus/mbserver/mbserver.tar.gz
[127.0.0.1] run: cd /home/metabus/mbclient/; tar xf mbclient.tar.gz; rm mbclient.tar.gz
[127.0.0.1] run: cd /home/metabus/mbclient/; tar xf mbclient_media.tar.gz; rm
mbclient_media.tar.gz
[127.0.0.1] run: cd /home/metabus/mbserver/; tar xf mbserver.tar.gz; rm mbserver.tar.gz
[127.0.0.1] sudo: /etc/init.d/apache2 restart
[127.0.0.1] out: * Restarting web server apache2
[127.0.0.1] out:  ...done.
Done.
```

4 Escalabilidade e Paralelismo

O Metabus foi projetado para ser totalmente escalável, principalmente por que para realizar uma busca, que não está em cache, é necessário consultar muitas fontes (talvez milhares) em um período de tempo aceitável para um usuário que está acostumado a utilizar mecanismos de busca convêncionais que retornam os resultados em menos de 1 segundo. Dessa forma, quanto mais tarefas possam ser executadas paralelamente, melhor, e ainda, aproveitando todos os recursos disponíveis, sejam recursos de rede ou de processamento.

Para viabilizar essa idéia o Metabus deve ser capaz de:

1. Utilizar mais de uma rede. Uma vez que a conexão com a internet disponível em um servidor pode não ser rápida o suficiente para consultar todas as fontes em tempo hábil.
2. Ampliar ao máximo o paralelismo da realização das consultas de maneira escalável. O principal gargalo dessa aplicação é a realização de uma busca em todas as fontes, se o tempo para finalizar essa busca não for aceitável para o usuário, deve existir a possibilidade de adicionar servidores para agilizar as buscas.
3. Utilizar *threads* para efetuar cada uma das buscas. A busca realizada em cada uma das fontes não pode ser sequencial dentro de um processo de forma alguma.

4.1. Registro de Servidores

O Metabus é escalável no nível de servidores. É possivel adicionar quantos servidores forem necessários para suprir a necessidade de processamento do sistema. Assim, cada servidor pode possuir recursos de rede diferentes e independentes (a utilização de uma rede não influencia na performance da outra) que serão somados e compartilhados como recursos do sistema Metabus.

Para adicionar novos servidores é necessário apenar efetuar o processo de instalaçao do mbserver na nova máquina e cadastrá-lo no mbclient (módulo serverlib), editando o arquivo settings.py e adicionando o IP ou nome do servidor na lista de servidores chamada SERVERS, como no exemplo abaixo:

```
SERVERS = [
  'http://localhost:9123/',
  'http://200.201.122.32:80/',
]
```

A identificação do servidor vem seguida da porta que o Metabus esta rodando.

4.2. Processo por processador

O Interpretador *Python* não é capaz de processar mais de uma *thread* de um mesmo processo e processadores diferentes simultaneamente. Dessa forma, o número que processos é configurável para prover a possibilidade de utilização de todos os processadores disponíveis em um servidor. O número de processos por servidor é normalmente configurado para ser igual ao número de processadores para que cada processo possa ser escalado para um processador ao mesmo tempo.

Para possibilitar essa configuração de número de processos utilizados por servidor foi utilizado o WSGI (http://wsgi.org/) e Apache2 que permitem a configuração de um pool de processos do serviço Metabus. O arquivo de configuração do Apache2, utilizado pelo Metabus, que contém os parametros relativos ao número de processos pode ser encontrado em mbserver/deploy/apache2/etc/apache2/sites-available/mbserver. Segue um trecho desse arquivo:

```
NameVirtualHost *:9123
<VirtualHost *:9123>
  ErrorLog /var/log/apache2/error.log
  # Possible values include: debug, info, notice, warn, error, crit,
  # alert, emerg.
  LogLevel error
  CustomLog /var/log/apache2/access.log combined
  ServerSignature On
  WSGIDaemonProcess mbserver user=metabus group=www-data processes=8 threads=1
  WSGIProcessGroup mbserver
  WSGIScriptAlias / /home/metabus/mbserver/mbserver.wsgi
</VirtualHost>
```

Para alterar o número de processos basta modificar o parametro *processes*.

4.3. Threads de Buscas

Dividir as requisições de buscas em *threads* é uma maneira de evitar que a execução do sistema fique interrompida enquanto se obtém os resultados de uma pesquisa em uma fonte de busca. Usando *threads*, as requisições podem ser feitas paralelamente, em outras palavas, é possivel fazer a requisição de uma pesquisa em mais de uma fonte antes que a primeira fonte retorne o resultado da busca.

O número de threads de cada processo também é configurável, o que possibilitará o desenvolvimento de testes automáticos para verificar o número ótimo de *threads* por processo em cada ambiente. Esse parâmetro pode ser editado no arquivo mbserver/server/settings.py:

```
THREADS_NUMBER = 10
```

5 Ajuda

Se foram encontrados problemas ou dúvidas durante a instalação, por favor envie um email para metabus@googlegroups.com.

Os arquivos de log podem ajudar a resolver a maioria dos problemas encontrados durante a instalação e execução do sistema. Os logs do sistema metabus são:

```
/home/metabus/var/log/metabus/mbserver.log
/home/metabus/var/log/metabus/mbclient.log
```

Além dos dados encontrados nos arquivos de logs do Metabus, outros dados sobre erros do sistema podem ser encontrados nos logs de erros dos servidores, por exemplo:

```
/var/log/apache2/error.log
```