# **PUPPET DICAS DIVERSAS**

# **testando sem aplicar (noop)**

Antes de aplicar uma configuração, podemos fazer um teste mais seguro, simulando as ações que o puppet vai executar durante a aplicação das configurações do catálogo.

Para fazer o teste usamos o parâmetro –noop, peguei o exemplo de trio postfix que já criamos para ilustrar o cenário.

puppetserverless:~/puppetlabs# puppet apply postfix.pp --noop

Veja a saída

notice: /Stage[main]//Package[postfix]/ensure: current\_value purged, should be present (noop)  
notice: /Stage[main]//File[main.cf]/ensure: current\_value absent, should be file (noop)  
err: /Stage[main]//Service[postfix]: Could not evaluate: Could not find init script for 'postfix'  
notice: /Stage[main]//Service[postfix]: Would have triggered 'refresh' from 1 events  
notice: Class[Main]: Would have triggered 'refresh' from 3 events  
notice: Stage[main]: Would have triggered 'refresh' from 1 events  
notice: Finished catalog run in 0.73 seconds

Observe que ele simula a execução mas não aplica nenhuma configuração. Esse parâmetro é essencialmente útil para avaliar por exemplo se algum pacote será removido devido a uma dependência ou se alguma modificação além do que foi declarado será feita como consequência da mudança. Na dúvida use o –noop antes de aplicar de fato.

Aplicando a configuração sem o noop

puppetserverless:~/puppetlabs# puppet apply postfix.pp

Saída

notice: /Stage[main]//Package[postfix]/ensure: ensure changed 'purged' to 'present'  
notice: /Stage[main]//File[main.cf]/content: content changed '{md5}6c2b81273db245fb47d3f5aacbc71c8e' to '{md5}fbffdbd6f85cdf5fd998c0c7c4a06ec6'  
notice: /Stage[main]//Service[postfix]: Triggered 'refresh' from 1 events  
notice: Finished catalog run in 22.25 seconds

No modo cliente/servidor você também poderá fazer testes antes de aplicar configurações, bastará usar o seguinte comando

puppet agent --test --noop

Mas isto, veremos na wiki puppet cliente/servidor.

# **debug/trace/log**

Se está ocorrendo algum problema, podemos utilizar alguns parâmetros que vão nos ajudar depurar o problema

## **debug**

Se quiser ativar o debug completo durante a execução do manifest.

Em modo autônomo

puppet apply manifest.pp --debug

Em modo cliente/servidor

puppet agent --test --debug

## **trace**

Se quiser debugar e imprimir stacl traces , primeiro habilite trace = true no puppet.conf e depois execute o comando abaixo:

Em modo autônomo

puppet apply manifest.pp --debug --trace

Em modo cliente/servidor

puppet agent --test --debug --trace

Mais info em<http://docs.puppetlabs.com/references/stable/configuration.html#trace>

## **redirecionando logs**

Se quiser direcionar a saída para um arquivo de log use os comandos abaixo:

Em modo autônomo

puppet apply -l /tmp/manifest.log manifest.pp

Em modo cliente/servidor

puppet agent --test -l /tmp/agent.log

# **filebucket (backup/restore)**

O filebucket é um recurso que nos permite salvar, recuperar e visualizar arquivos que foram modificados pelo puppet. Ele é um recurso essencial, é muito importante aprender a usá-lo corretamente.

## **restore**

eu tenho um arquivo main.cf com o seguinte conteúdo

# /etc/postfix/main.cf  
  
# HEADER: This file was autogenerated by puppet.  
# HEADER: While it can still be managed manually, it  
# HEADER: is definitely not recommended.  
  
smtpd\_banner = $myhostname ESMTP $mail\_name (detran)  
  
inet\_interfaces = all  
inet\_protocols = ipv4  
  
relayhost =  
recipient\_delimiter = +  
  
myorigin = agent-debian.hacklab  
mydestination = $myhostname, localhost.$mydomain, localhost, agent-debian.hacklab  
mynetworks = 127.0.0.0/8 10.138.26.0/24  
  
unknown\_local\_recipient\_reject\_code = 550  
  
alias\_maps = hash:/etc/aliases  
alias\_database = hash:/etc/aliases

na ultima linha eu vou adicionar um comentário

# teste de recuperacao do filebucket

e vou rodar o puppet

root@agent-debian:/tmp# puppet agent --testinfo: Retrieving plugin

saída

info: Loading facts in /var/lib/puppet/lib/facter/etcgroup.rb  
info: Loading facts in /var/lib/puppet/lib/facter/etcpasswd.rb  
No LSB modules are available.  
info: Caching catalog for agent-debian.hacklab  
info: Applying configuration version '1340384371'  
notice: /Stage[main]/Localmta/File[/etc/postfix/main.cf]/content:   
--- /etc/postfix/main.cf 2012-06-22 16:12:39.000000000 -0300  
+++ /tmp/puppet-file20120622-5649-1e4kghw-0 2012-06-22 16:13:36.000000000 -0300  
@@ -20,5 +20,3 @@  
   
 alias\_maps = hash:/etc/aliases  
 alias\_database = hash:/etc/aliases  
-  
-# teste de recuperacao do filebucket  
  
info: /Stage[main]/Localmta/File[/etc/postfix/main.cf]: Filebucketed /etc/postfix/main.cf to server with sum d59100e183899a5d11f435656c4d8f5b  
notice: /Stage[main]/Localmta/File[/etc/postfix/main.cf]/content: content changed '{md5}d59100e183899a5d11f435656c4d8f5b' to '{md5}c9bcb67fdf5c367dd40691aad07fa3b9'  
info: /Stage[main]/Localmta/File[/etc/postfix/main.cf]: Scheduling refresh of Service[postfix]  
notice: /Stage[main]/Localmta/Service[postfix]: Triggered 'refresh' from 1 events  
notice: Finished catalog run in 24.85 seconds

veja que o puppet modificou o arquivo pois ele estava diferente do que foi declarado, mas não se aflija, caso precise recuperá-la basta usar o comando abaixo

root@agent-debian:/tmp# puppet filebucket restore /etc/postfix/main.cf d59100e183899a5d11f435656c4d8f5b

observe que estamos usando o hash da modificação, este pode ser obtido no arquivo /var/log/puppet/puppet.log ou durante a execução do puppet como pode ve no log acima do comando, agora vamos conferir o arquivo realmente foi recuperado

root@agent-debian:/tmp# cat /etc/postfix/main.cf

saída

# /etc/postfix/main.cf  
  
# HEADER: This file was autogenerated by puppet.  
# HEADER: While it can still be managed manually, it  
# HEADER: is definitely not recommended.  
  
smtpd\_banner = $myhostname ESMTP $mail\_name (detran)  
  
inet\_interfaces = all  
inet\_protocols = ipv4  
  
relayhost =  
recipient\_delimiter = +  
  
myorigin = agent-debian.hacklab  
mydestination = $myhostname, localhost.$mydomain, localhost, agent-debian.hacklab  
mynetworks = 127.0.0.0/8 10.138.26.0/24  
  
unknown\_local\_recipient\_reject\_code = 550  
  
alias\_maps = hash:/etc/aliases  
alias\_database = hash:/etc/aliases  
  
# teste de recuperacao do filebucket

veja que confere, é o arquivo que adicionamos o comentário

## **get**

se você quiser apenas observar o arquivo que foi salvo, pode usar o parâmetro get

root@agent-debian:/tmp# puppet filebucket get d59100e183899a5d11f435656c4d8f5b

saída

# /etc/postfix/main.cf  
  
# HEADER: This file was autogenerated by puppet.  
# HEADER: While it can still be managed manually, it  
# HEADER: is definitely not recommended.  
  
smtpd\_banner = $myhostname ESMTP $mail\_name (detran)  
  
inet\_interfaces = all  
inet\_protocols = ipv4  
  
relayhost =  
recipient\_delimiter = +  
  
myorigin = agent-debian.hacklab  
mydestination = $myhostname, localhost.$mydomain, localhost, agent-debian.hacklab  
mynetworks = 127.0.0.0/8 10.138.26.0/24  
  
unknown\_local\_recipient\_reject\_code = 550  
  
alias\_maps = hash:/etc/aliases  
alias\_database = hash:/etc/aliases  
  
# teste de recuperacao do filebucket

## **backup**

agora vamos aprender a fazer backup de um arquivo, vou adicionar mais uma linha ao final do arquivo, apenas para ter um novo hash

# novo teste, agora de backup

e vamos fazer o backup

root@agent-debian:/tmp# puppet filebucket backup /etc/postfix/main.cf

saída

/etc/postfix/main.cf: 2de085d00496a249b5eaba083809ad08

veja que ele me deu um novo hash, com ele eu posso fazer um restore ou visualizar o conteúdo com o get, para testar vamos rodar o get

root@agent-debian:/tmp# puppet filebucket get 2de085d00496a249b5eaba083809ad08

saída

# /etc/postfix/main.cf  
  
# HEADER: This file was autogenerated by puppet.  
# HEADER: While it can still be managed manually, it  
# HEADER: is definitely not recommended.  
  
smtpd\_banner = $myhostname ESMTP $mail\_name (detran)  
  
inet\_interfaces = all  
inet\_protocols = ipv4  
  
relayhost =  
recipient\_delimiter = +  
  
myorigin = agent-debian.hacklab  
mydestination = $myhostname, localhost.$mydomain, localhost, agent-debian.hacklab  
mynetworks = 127.0.0.0/8 10.138.26.0/24  
  
unknown\_local\_recipient\_reject\_code = 550  
  
alias\_maps = hash:/etc/aliases  
alias\_database = hash:/etc/aliases  
  
# teste de recuperacao do filebucket  
# novo teste, agora de backup

bacana, o filebucket é bastante seguro e bastante prático, é essencial saber usá-lo.

# **uso de puppet autônomo**

depois de ver tudo isso pode ser que ainda fique a dúvida, quais usos teria para o puppet em modo autônomo/serverless?

## **em seu desktop**

pode usar isso para por exemplo configurar seu ambiente de usuário em seu Desktop Linux ou Unix, fazendo isto em segundos após a instalação do OS.

puppet apply meudesktop.pp

## **para instalar o puppet**

podes criar um manifest puppetmaster-passenger.pp para instalar e configurar o puppet em seu ambiente, fazendo isto em segundos.

puppet apply puppetmaster-passenger.pp

## **para instalar serviços**

podes criar um manifest para instalar e configurar por exemplo um ambiente tomcat completo, fazendo isto em segundos.

puppet apply tomcat-environment.pp

# **puppet modo cliente/servidor**

Agora que já entendemos como funciona o Puppet em modo autônomo, é hora de entender como é que funciona em modo cliente/servidor.

Acesse a Wiki …..

# **referências**

* <http://docs.puppetlabs.com/learning/ral.html>
* <http://docs.puppetlabs.com/learning/manifests.html>
* <http://docs.puppetlabs.com/learning/ordering.html>
* <http://docs.puppetlabs.com/learning/variables.html>
* <http://docs.puppetlabs.com/learning/modules1.html>
* <http://docs.puppetlabs.com/learning/templates.html>
* <http://docs.puppetlabs.com/learning/modules2.html>
* <http://docs.puppetlabs.com/learning/definedtypes.html>
* <http://docs.puppetlabs.com/guides/puppetlabs_package_repositories.html>
* <http://docs.puppetlabs.com/guides/language_guide.html>