Foreman - Feature #1561

Add new provision template types for EC2

03/30/2012 02:57 PM - David Swift

Status:	New		
Priority:	Normal		
Assignee:			
Category:	Unattended installations		
Target version:			
Difficulty:		**Fixed in Releases:**	
Triaged:	No	**Found in Releases:**	
Bugzilla link:		**Red Hat JIRA:**	
Pull request:			

Description

EC2 uses cloud-init to bootstrap machines. This can be configured to run custom cloud-init scripts, as well as even Bash scripts.

To facilitate using EC2, I would like to add 2 new provisioning templates.

The first is called 'cloud-init', and it is available based upon the same rules as any other template. When the EC2 instance is built this script is passed as userdata via fog.

The second is called 'cloud-include', and it is available at a URL on the Foreman host to be used by the 'cloud-init' script if required. However, this one requires a bit of forethought, as the Foreman host must be available to the EC2 instance during bootstrap via security group rules.

Related issues:

Related to Foreman - Feature #1223: Provisioning engine for EC2	**Closed**	**10/12/2011**

History

#1 - 05/09/2012 08:12 AM - Ohad Levy

I progressed quite a lot on my ec2 integeration, and now you can
1. create instances
2. scp a template and ssh to an instance configure it to run puppet.

at the moment, for 2. i use a finish script type, and I'm not 100% sure if its required to add another type.

for cloud init, we could add another provisioning method, but we need to figure out a way of how to sign the certs automatically etc.

another question that raised during this, is how we should handle hostnames in ec2, should we rename them to the host entered in foreman etc..

so, my question is, whats the last state of this ticket ;)

#2 - 05/10/2012 03:19 PM - David Swift

I think a user-data type would make it clear where to start custom bootstrapping for people that are using EC2.

I think a cloud-init type would also be very useful, as you can do advanced provisioning of drives, repositories, and other details.

The bit that is currently hard for me, is being able to deliver a user-data or cloud-init script via fog to an instance with no current IP. When we provision machines via puppet, they are already partially provisioned, either via Rackspace, Internap, or AWS. So, we might or might not know the IP when we assign a machine to an environment and hostgroup, as well as assign classes. I call this "provisioning a presented machine", instead of "provisioning a new machine".

In the 0.4.2 Foreman, we cannot add instances without an IP, so all the automatic decision-making that goes on for provisioning scripts is not usable for presented machines. What I do is a hack, using the "spoof" parameter, to allow a new puppet client in EC2 to download some Foreman URLs that configure it. Puppet and Foreman are configured to auto-sign certificates, since only clients that are in my security group can see the puppetmaster this works. After I get it recognized by Puppet & Foreman, I can then assign it classes, and one of those classes takes care of renaming the host based upon our internal naming scheme.

What I want to do, is be able to have user-data and cloud-init provisioning scripts work for "presented" machines just like kickstart provisioning scripts work for PXE clients in Foreman. This would then let me assign environments, hostgroups, classes, and parameters to a machine that Fog would provision. Fog would get passed the user-data and cloud-init scripts based upon the rules the same way kickstart provisioning works, just without knowing the client IP beforehand..

Here are ugly details of what I am doing currently:

- using ec2-tools, I pass a user-data script that looks like this (where PMIP is the puppetmaster IP):

```
--user-data '#include'$'\n'"http://${PMIP}/unattended/script/SPI_aws_cloud_config?spoof=${PMIP}"$'\n'"http
://${PMIP}/unattended/finish/SPI_aws_client_userdata?spoof=${PMIP}"
```

- ec2-tools delivers the user-data securely to the new instance
- This gives the new EC2 instance access to a cloud-init script with two URLs to use on the new client instance
- The first URL (unattended/script/SPI_aws_cloud_config) reconfigures the EC2 instance to lock it's cloud-init (amzn-main) repo to 2011.09
- The 2nd URL then supplies a script that does the following:
    1. gets the address of the new instance
    2. Via curl, tells the Foreman instance on the puppetmaster that the new client is alive
    3. Calls a snippet that then proceeds with most of a custom "kickstart-style" configuration run

unattended/finish/SPI_aws_client_userdata

```
#!/bin/sh
#
#  Run this script upon creation of AWS puppet client instances to simulate the end resu
lt of the internap kickstart
#  Run it by:
#   ec2-tools, ex:  ec2-run-instances --key KEYPAIR --user-data-file <%= foreman_url %>/
unattended/script/SPI_aws_client_install ami-cbc12fa2

#
#  tell foreman we are here - this is a hack, as foreman will support this in a much bet
ter way soon
#
IPADDR=`/sbin/ifconfig eth0 |  /bin/awk '/inet addr:/ {print $2}' | /bin/cut -d : -f 2`

curl -H "Accept:application/json"  -d "host[name]=`hostname -f`"  -d "host[architecture_
id]=0" -d "host[domain_id]=1" -d "host[environment_id]=2" \
-d "host[ip]=${IPADDR}" -d "host[mac]=`/sbin/ifconfig eth0 | /usr/bin/head -n 1 |\
/bin/awk '{print $5}'`" -d "host[operatingsystem_id]=1" -d "host[ptable_id]=1" -d "host[
puppetmaster]=<%= @host.puppetmaster %>" https://<%= @host.puppetmaster %>/hosts -k

echo "<%= @host.puppetmaster %>  puppet" >> /etc/hosts
echo <%= snippet("SPI_client_bootstrap") %>  | sh -s
exit 0
```

snippet("SPI_client_bootstrap")
This snippet :

1. **Adjusts some packages**
2. **injects a puppet configuration**
3. **runs the first puppet run, to make the CSR**
4. **Informs foreman it is done**

```
#!/bin/sh
#
#  This script is called via curl for AWS clients, it could also be used to handle in
ternap boxes
#
#  Prevent amazon from foisting the latest packages for their AMI on us
#
sed -i -e 's/releasever/\#releasever/' /etc/yum.conf

#
```

```
#  Replace the GID of the ec2 user
#
groupmod -g 502 ec2-user

#
#  First, remove some packages that we will explicitly replace
#
/usr/bin/yum -y remove ruby ruby-libs augeas-libs

#
# bad hack to support using small (32-bit) and large (64-bit) instances w/o requiring
 the user to do much
#
rpm -ivh http://apt.sw.be/redhat/el6/en/x86_64/extras/RPMS/augeas-libs-0.9.0-1.el6.rf
x.x86_64.rpm
rpm -ivh http://apt.sw.be/redhat/el6/en/i386/extras/RPMS/augeas-libs-0.9.0-1.el6.rfx.
i686.rpm
#
# get to the version of ruby that passenger currently supports
#
/usr/bin/yum -y install ruby-1.8.7.352-1.8.amzn1 ruby-irb-1.8.7.352-1.8.amzn1 ruby-rd
oc-1.8.7.352-1.8.amzn1 rubygems ruby-shadow ruby-augeas aws-amitools-ec2

#
#  Correct password setup problem with puppet
#
ln -s /usr/lib64/ruby/site_ruby/1.8/x86_64-linux/shadow.so /usr/lib/ruby/site_ruby/1.
8/

#
# make it a bit easier for RHEL packages to install
#
sh -c 'cat > /etc/yum/vars/releasever << RELEASEVER
6
RELEASEVER'
sh -c 'cat > /etc/yum/vars/amznreleasever << AMZNRELEASEVER
latest
AMZNRELEASEVER'
/usr/bin/perl -i -p -e 's/releasever/amznreleasever/g' /etc/yum.repos.d/amzn-*

sh -c 'cat > /etc/yum.repos.d/puppetlabs.repo << PUPPETREPO
[puppetlabs]
name=Puppet Labs Packages
baseurl=http://yum.puppetlabs.com/el/\$releasever/products/\$basearch
enabled=0
gpgcheck=0
PUPPETREPO'

#
#  Finish package installation
#
/usr/bin/yum -y install libselinux-ruby dmidecode pciutils facter
/usr/bin/yum -y --enablerepo=epel install ruby-shadow
/usr/bin/yum -y --enablerepo=puppetlabs --disablerepo=amzn\* install puppet

sh -c 'cat > /etc/puppet/puppet.conf << PUPPETCONF
<%= snippet("puppet.conf") %>
PUPPETCONF'

/usr/sbin/puppetd --config /etc/puppet/puppet.conf -o --tags no_such_tag --server <%=
```

```
 @host.puppetmaster %>  --no-daemonize

sync

# Inform the build system that we are done.
echo "Informing Foreman that we are built"
wget -q -O /dev/null --no-check-certificate <%= foreman_url %>
exit 0
```

#3 - 05/13/2012 02:30 PM - Ohad Levy

test branch now feedback welcomed - https://github.com/ohadlevy/foreman/tree/ec2_provisioning

#4 - 06/07/2012 07:51 PM - David Swift

I am putting together a pull request, I have this working on my dev instance. While my original needs are pretty well met by the SSH provisioning and a 'finish' script, access to the user_data gives us the chance to use cloud-init, if nothing else to lock the amzn repo. However, there is a lot of buried power there I am just staring to use in our deployment, and the ability to have a templated cloud-init script really works well with our hostgroups breakdown. For instance, using cloud-init, you could resize the default root partition size, among other things.

If you accept the pull request, this can be tested/closed. Thanks a million for all your effort and help.

#5 - 06/12/2012 02:17 AM - Ohad Levy

any chance we can discuss this a bit over irc? I'm still not 100% sure on the flow that you are suggesting

thanks!

#6 - 06/21/2012 04:40 AM - Ohad Levy

- Target version deleted (1.0)

#7 - 07/12/2012 09:22 PM - David Swift

Created pull request https://github.com/theforeman/foreman/pull/107

#8 - 08/16/2012 04:21 PM - Brian Gupta

Ohad,

Thinking about this more and more... I think the way to handle this in a method that's relatively secure is to create a new type of role or user that supports only grabbing a template/script, and has a separate set of credentials. These credentials would probably also need to allow a node to self-register in foreman.

-Brian

#9 - 07/12/2018 07:53 AM - Ohad Levy

- Assignee deleted (Ohad Levy)

- Triaged set to No