Automated Provisioning

Me and surroundings

David Caro (dcaroest@redhat.com)

- Sysadmin transformed to CI engineer
- Working at Red Hat for Red Hat Enterprise Virtualization project
- Member of the infra team for the oVirt project

oVirt

RED HAT ENTERPRISE VIRTUALIZATION

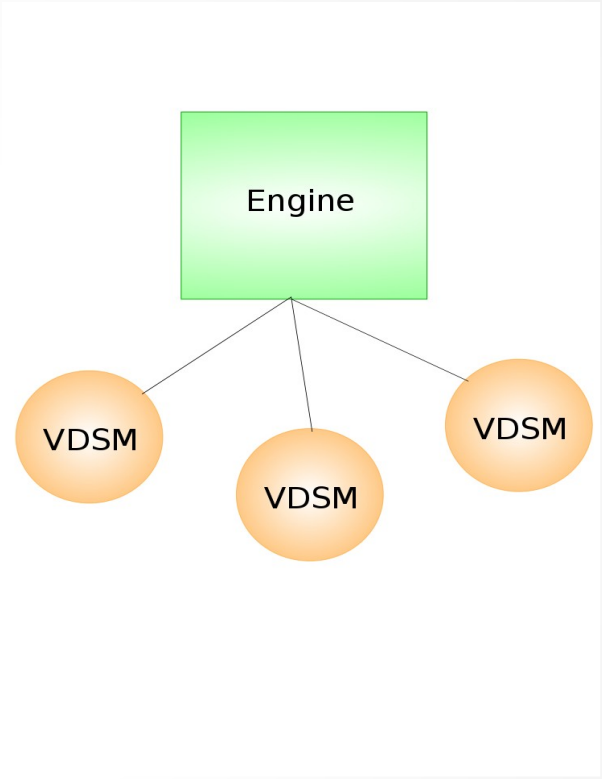
THE FOUNDATION FOR THE OPEN CLOUD

oVirt review

oVirt Application main componenets:

- Engine
  - Java based
  - Single instance
  - No need for hypervisor

- VDSM (Virtual Desktop and Server Manager)
  - Python based
  - Multiple instances (one per host)
  - Needs hypervisor (and physical host)



The issue

A few of the tests require:

- One engine

Most of the tests require:

- One engine
- One VDSM

Some of the tests require:

- One engine
- Multiple VDSM

This means having to provide lots of physical hosts.

The issue

Physical hosts are expensive, and the budget is tight...

Recycle!!

The Tools

The base we all know

- **Puppet**
  Finishes to configure the host once the base OS is installed.

- **Foreman**
  Orchestrates at a high level the Foreman Proxy and Puppet, used as external node classifier, stores the hostgroup definitions.

  - **Foreman Proxy**
    Orchestrates lower level services to allow PXE boot and provides the kickstart scripts that run puppet for the first time after installation.

The Tools

The extra spice:

- **foreman_reserve**
  Foreman plugin that adds host allocation capabilities to Foreman
  https://rubygems.org/gems/foreman_reserve

- **python-foreman**
  Python bindings for the Foreman API (V1 for now...)
  https://pypi.python.org/pypi/python-foreman/

- **Fabric**
  High level framework used directly by the user (in our case by jenkins) to wraps all the previous tools in a sensible manner
  http://fabric.readthedocs.org/

The Foreman

Installation:

Latest foreman (1.4, updated it lately, yay!)

Smart proxy, with:

- TFTP
- DHCP
- DNS (well... more or less)
- Puppet CA

Installation media, kickstart templates, operating systems and subnets properly configured.

For info on how to do that you can check the nice documentation:
http://theforeman.org/manuals/1.4/index.html#4.4Provisioning

foreman_reserve

Foreman plugin that provides host allocation capabilities

Right now it's very, very simple, two key concepts:

- Use the **RESERVED** host property to store the allocation status
- Block concurrent host allocations

Where is it?

- Gem:
  https://rubygems.org/gems/foreman_reserve
- Code:
  https://github.com/david-caro/foreman_reserve

To install it you must use the manual plugin installation procedure:

http://theforeman.org/manuals/1.4/index.html#6.Plugins

But the packaging is comming soon!

python-foreman

Python bindings for the Foreman API (V1)

- Autogenerated
  - Uses cached definitions by default to speed up the import, you can force it to regenerate using special module methods.
- Includes the documentation inside the function doc strings
- Basic functionality is stable and continuouslly used
- More complex scenarios are yet untested

You can find it on pypy or github:

- https://github.com/david-caro/automatic-provision-examples.git
- https://pypi.python.org/pypi/python-foreman

python-foreman

```
$> ipython
In [1]: from foreman import client as cli

In [2]: from getpass import getpass

In [3]: frm = cli.Foreman('http://my-foreman.com', auth=('awesome_user', getpass()))
Password:******

In [4]: frm.<tab>
Display all 142 possibilities? (y or n)
frm.bootfiles_operatingsystems           frm.index_models
frm.build_pxe_default_config_templates   frm.index_operatingsystems
frm.create_architectures                 frm.index_ptables
frm.create_auth_source_ldaps             frm.index_puppetclasses
frm.create_bookmarks                     frm.index_reports
frm.create_common_parameters             frm.index_roles
....
frm.index_config_templates               frm.update_ptables
frm.index_dashboard                      frm.update_puppetclasses
frm.index_domains                        frm.update_reserved_reason
frm.index_environments                   frm.update_roles
frm.index_fact_values                    frm.update_settings
frm.index_home                           frm.update_smart_proxies
frm.index_hostgroups                     frm.update_subnets
frm.index_hosts                          frm.update_usergroups
frm.index_images                         frm.update_users
frm.index_lookup_keys                    frm.url
frm.index_media                          frm.version
```

python-foreman

Foreman API provides 4 main method types, that allow you to execute the most common operations.

- index_*
  Shows a list of all the objects that match with a littel information for further reference.
- show_*
  Shows detailed information about the given object.
- create_*
  Creates a new entry of the object with the passed information
- update_*
  Updates an already existent object with the given information.

python-foreman

```
In [15]: frm.index_hosts()
Out[15]:
[{u'host': {u'hostgroup_id': 68,
   u'id': 400,
   u'name': u'builder02-rhel64.eng.lab.tlv.redhat.com',
   u'operatingsystem_id': 7}},
 {u'host': {u'hostgroup_id': 59,
   u'id': 417,
   u'name': u'buri03.ci.lab.tlv.redhat.com',
   u'operatingsystem_id': 7}},
 {u'host': {u'hostgroup_id': 59,
   u'id': 418,
   u'name': u'buri04.ci.lab.tlv.redhat.com',
   u'operatingsystem_id': 7}},
 {u'host': {u'hostgroup_id': 1,
   u'id': 508,
   u'name': u'buri07.ci.lab.tlv.redhat.com',
   u'operatingsystem_id': 11}},
...
 {u'host': {u'hostgroup_id': 82,
   u'id': 139,
   u'name': u'cinteg35.ci.lab.tlv.redhat.com',
   u'operatingsystem_id': 10}},
 {u'host': {u'hostgroup_id': 82,
   u'id': 140,
   u'name': u'cinteg36.ci.lab.tlv.redhat.com',
   u'operatingsystem_id': 10}}]
```

python-foreman

```
In [34]: frm.show_hosts(139)
Out[34]:
{u'host': {u'architecture_id': 1,
  u'build': False,
  u'certname': u'cinteg35.ci.lab.tlv.redhat.com',
  u'enabled': True,
  u'environment': {u'environment': {u'id': 1, u'name': u'production'}},
  u'hostgroup_id': 82,
  u'id': 139,
  u'installed_at': u'2013-09-29T09:38:36Z',
  u'ip': u'10.35.148.108',
  u'last_report': u'2014-02-01T21:12:03Z',
  u'mac': u'b4:99:ba:ab:ca:9c',
  u'managed': True,
  u'medium_id': 14,
  u'model_id': 11,
  u'name': u'cinteg35.ci.lab.tlv.redhat.com',
  u'operatingsystem_id': 10,
  u'owner_id': 1,
  u'owner_type': u'User',
  u'parameters': [{u'parameter': {u'created_at': u'2013-01-23T15:17:21Z',
     u'id': 297,
     u'name': u'RESERVED',
     u'priority': 4,
     u'reference_id': 139,
     u'updated_at': u'2014-01-31T18:29:16Z',
     u'value': u'false'}}],
  u'updated_at': u'2014-02-01T21:12:11Z',
  u'use_image': None,
  u'uuid': None}}
```

python-foreman

```
In [35]: frm.create_hosts?

Type:       instancemethod
String Form:<bound method Foreman.create_hosts of <foreman.client.Foreman object at
0x32150d0>>
File:       Dynamically generated function. No source code available.
Definition: frm.create_hosts(self, host)
Docstring:
:param host[hostgroup_id]: type number., optional
:param host[architecture_id]: type number., required
:param host[sp_subnet_id]: type number., optional
:param host[subnet_id]: type number., optional
:param host[owner_id]: type number., optional
:param host: type Hash, required
:param host[host_parameters_attributes]: type Array, optional
:param host[name]: type String, required
:param host[model_id_id]: type number., optional
:param host[puppet_ca_proxy_id]: type number., optional
:param host[domain_id]: type number., required
:param host[mac]: type String, required
:param host[ip]: type String, required
:param host[operatingsystem_id]: type String, required
:param host[ptable_id]: type number., optional
:param host[image_id]: type number., optional
:param host[medium_id]: type number., optional
:param host[puppet_proxy_id]: type number., required
:param host[environment_id]: type String, required
```

python-foreman

Rebuilding a host becomes as easy as:

```
In [40]: frm.update_hosts(id=139, host={'build': True})
Out[40]:
{u'host': {u'architecture_id': 1,
  u'build': True,
  u'managed': True,
  u'name': u'cinteg35.ci.lab.tlv.redhat.com',
...
  u'uuid': None}}
```

You can also use the internal host id number or its name:

```
In [41]: frm.show_hosts(id=139) == frm.show_hosts(id='cinteg35.ci.lab.tlv.redhat.com')
Out[41]:
True
```

Fabric

Fabric is a Python (2.5 or higher) library and command-line tool for streamlining the use of SSH for application deployment or systems administration tasks.

- Has very useful methods to handle remote command execution
- Provides parallelization and thread control ad-hoc, no need for extra code
- Very simple to use, only needs ssh running on the target hosts, no extra clients, no extra infrastructure.
- Python code (for us, that's a point as we all have python coding experience)
- Very easy to plug custom tasks for host selection
  Here's where python-foreman comes into play!
- Perfect for small/medium infrastructures

Custom tasks (fabric_ci)

You can find most of these examples and others at https://github.com/david-caro/automatic-provision-examples.git, we'll try to open the rest of custom tasks as soon as posible.

Fabric

Ewww...

that's just another ssh wrapper!!

Fabric

Example silly task:

```
from fabric.api import (
    run,
    task,
    settings,
    hide,
    env,
)
from fabric_ci.lib.utils import info

@task
def sync(ntp_server=None):
    """
    Sync the server with the given ntp server, stops and restarts the ntpd
    service if it was enabled.

    :param ntp_server:
        NTP server to use, default = env['NTP_SERVER']
    """
    if ntp_server is None:
        ntp_server = env['NTP_SERVER']
    start = False
    with settings(hide('running', 'status', 'stderr', 'stdout', 'warnings'),
                  warn_only=True):
        res = run("service ntpd status")
    if res.succeeded:
        info("Stopping ntpd")
        run("service ntpd stop")
        start = True
    info("Synching the time")
    run("ntpdate %s" % ntp_server)
    if start:
        info("Starting ntpd again")
        run("service ntpd start")
```

Fabric

```
$> fab -H cinteg15.ci.lab.tlv.redhat.com do.system.ntp.sync
[01/02/2014 23:24:36|by cascara@lanithro] [INFO] Stopping ntpd
[cinteg15.ci.lab.tlv.redhat.com] out: Shutting down ntpd:  [  OK  ]
[cinteg15.ci.lab.tlv.redhat.com] out:
[cinteg15.ci.lab.tlv.redhat.com] out:
[01/02/2014 23:24:37|by cascara@lanithro] [INFO] Synching the time
[cinteg15.ci.lab.tlv.redhat.com] out:  2 Feb 00:24:39 ntpdate[12830]: adjust time
server 10.11.160.238 offset 0.001003 sec
[cinteg15.ci.lab.tlv.redhat.com] out:
[01/02/2014 23:24:39|by cascara@lanithro] [INFO] Starting ntpd again
[cinteg15.ci.lab.tlv.redhat.com] out: Starting ntpd:       [  OK  ]
[cinteg15.ci.lab.tlv.redhat.com] out:
[cinteg15.ci.lab.tlv.redhat.com] out:
```

Ok that's nice, but you used one host and you specified it manually...

you cheater

Fabric

Selecting hosts from Foreman using Foreman's powerful search syntax:

```
$ fab on.foreman:"name ~ cinteg1%"
Query used:
     "name ~ cinteg1%"
Got 10 hosts:
     cinteg10.ci.lab.tlv.redhat.com
     cinteg11.ci.lab.tlv.redhat.com
     cinteg12.ci.lab.tlv.redhat.com
     cinteg13.ci.lab.tlv.redhat.com
     cinteg14.ci.lab.tlv.redhat.com
     cinteg15.ci.lab.tlv.redhat.com
     cinteg16.ci.lab.tlv.redhat.com
     cinteg17.ci.lab.tlv.redhat.com
     cinteg18.ci.lab.tlv.redhat.com
     cinteg19.ci.lab.tlv.redhat.com
Is what you expected? y|n [y]
```

Let's see a parallel execution of the previous task using Foreman to select the hosts it will run on

Fabric

```
$ fab -P -z 5 on.foreman:"name ~ cinteg1%" do.system.ntp.sync
Query used:
     "name ~ cinteg1%"
Got 10 hosts:
     cinteg10.ci.lab.tlv.redhat.com
...
     cinteg19.ci.lab.tlv.redhat.com
[0/0/10] finished, running, queued
[0/1/9] finished, running, queued
...
[0/5/5] finished, running, queued
[01/02/2014 23:28:54|on cinteg15.ci.la b.tlv.redhat.com|by cascara@lanithro] [INFO]
Stopping ntpd
[01/02/2014 23:28:54|on cinteg17.ci.lab.tlv.redhat.com|by cascara@lanithro] [INFO]
Stopping ntpd
01/02/2014 23:28:54|on cinteg18.ci.lab.tlv.redhat.com|by cascara@lanithro] [INFO]
Stopping ntpd
[01/02/2014 23:28:54|on cinteg16.ci.lab.tlv.redhat.com|by cascara@lanithro] [INFO]
Stopping ntpd
[01/02/2014 23:28:54|on cinteg19.ci.lab.tlv.redhat.com|by cascara@lanithro] [INFO]
Stopping ntpd
[cinteg15.ci.lab.tlv.redhat.com] out: Shutting down ntpd:  [  OK  ]
[cinteg15.ci.lab.tlv.redhat.com] out:
[cinteg15.ci.lab.tlv.redhat.com] out:  2 Feb 00:29:05 ntpdate[16283]: adjust time
server 10.11.160.238 offset 0.001131 sec

...
[3/2/5] finished, running, queued
[3/3/4] finished, running, queued
[3/4/3] finished, running, queued
...
[ 10 OK / 0 ERROR ] in 19.2893750668 seconds
```

Fabric

Take a close look to the passed options:

```
$ fab -P -z 5 on.foreman:"name ~ cinteg1%" do.system.ntp.sync
```

The '-P' option enables the parallel execution, and the '-z' one is to select the maximum size of the thread pool to use.

You can see that it started only 5 threads:

```
[0/5/5] finished, running, queued
```

And that it started adding tasks once the pool started finishing the current ones

```
[3/2/5] finished, running, queued
[3/3/4] finished, running, queued
[3/4/3] finished, running, queued
```

But I don't care about that bull$=|¬, I'm using another non-ssh framework, so just show me the code!

Fabric

The code for that task is a little more complicated (error handling mostly), this one is a simplified working version (you can see the full task in the examples repo, you can ping me anytime if you have questions)

```
from foreman.client import Foreman
from foreman_ci.lib.foreman import foreman_defaults

@task(default=True)
@runs_once #Do not run for each host, just once
@serial    #Do not run in parallel
@foreman_defaults #This decorator populates the foreman, user and passwd parameters from env
def search(searchstr='', foreman=None, user=None, passwd=None):
    """
    Use the given foreman search result as the hosts list.
    :param searchstr: Query to filter the hosts with
    :param foreman: URL to foreman
    :param user: Foreman username
    :param passwd: Foreman password
    """
    if user:
        auth = (user, passwd)
    else:
        auth = None
    frm = frm_cli.Foreman(foreman, auth)
    for host in frm.index_hosts(search=searchstr, per_page=999):
        env.hosts.append(host['host']['name'])
    print(yellow("Query used: \n\t\"%s\"" % searchstr))
    print(yellow("Got %d hosts: \n\t" % len(env.hosts)
                 + '\n\t'.join(env.hosts)))
    if not env.parallel:
        if prompt('Is what you expected? y|n', default='y').lower() == 'n':
            abort('Ended by user request.')
```

Provision flow example

```
$ fab -d do.provision
Displaying detailed information for task 'do.provision':

    Provision the given host or hosts
    :param profile: Use this profile
    :param query: Use this query when selecting the hosts
    :param change_profile: Change the profile of the hosts if no free hosts
        found for the given profile (false by default)
    :param force_rebuild: Rebuild the host from scratch (false by default)
    :param reason: New reason to put into
    :param amount: Reserve only this amount of hosts
    :param outfile: Write the csv list of provisioned hosts in that file
    :param add_tag: Add the [USER RESERVED] tag to the reason (true by default)
    :param tries: Times to try in case of failure (300 by default)
    :param timeout: How much time in seconds to wait between tries
        (60 by default)
    :param foreman: URL to the foreman server
    :param user: username to login into Foreman
    :param passwd: Password to use when logging in
```

Provision flow example

```
$ fab do.provision:profile=P-RHEVM-3.4-RHEL65-HOSTS,amount=1,reason="Doing some tests for
cfgmgmt",force_rebuild=true
[03/02/2014 16:58:07|by cascara@lanithro] [INFO] ########## Reserving hosts
[03/02/2014 16:58:07|by cascara@lanithro] [INFO] Seeing if we have free hosts for the given
profile.
[03/02/2014 16:58:17|by cascara@lanithro] [INFO] Trying to get enough hosts each 60s, have 0 of
1, 300 tries left
Connecting to cinteg08.ci.lab.tlv.redhat.com: up and running.
[03/02/2014 16:58:21|by cascara@lanithro] [INFO] ########## Provisioning hosts
[03/02/2014 16:58:21|by cascara@lanithro] [INFO] force_rebuild=true, rebuilding the hosts.
[0/0/1] finished, running, queued  ---> Here the parallel task kicks in
[0/1/0] finished, running, queued
[03/02/2014 16:58:29|on cinteg08.ci.lab.tlv.redhat.com|by cascara@lanithro] [INFO] Rebooting
host cinteg08.ci.lab.tlv.redhat.com
[03/02/2014 16:58:37|on cinteg08.ci.lab.tlv.redhat.com|by cascara@lanithro] [INFO] Waiting for
cinteg08.ci.lab.tlv.redhat.com to be built
...
[03/02/2014 17:10:16|on cinteg08.ci.lab.tlv.redhat.com|by cascara@lanithro] [INFO] Waiting for
cinteg08.ci.lab.tlv.redhat.com to be reachable through ssh
...
[03/02/2014 17:12:24|on cinteg08.ci.lab.tlv.redhat.com|by cascara@lanithro] [INFO] Host
cinteg08.ci.lab.tlv.redhat.com done
[1/0/0] finished, running, queued
[ 1 OK / 0 ERROR ] in 846.198055029 seconds  --> The parallel task finished
[03/02/2014 17:12:27|by cascara@lanithro] [INFO] ########## Everything went perfect.
Done B)
[03/02/2014 17:12:30|by cascara@lanithro] [INFO] ########## Cleaning up...

$ fab do.provision.show_reserved
Host                                Profile                         Reason
cinteg08.ci.lab.tlv.redhat.com      P-RHEVM-3.4-RHEL65-HOSTS        [03/02/2014 17:12:27|by
cascara@lanithro] Doing some tests for cfgmgmt
```

Simple dynamic pools

What we are currently using is:

As simple as:

Try to get a host from the given profile pool
If no free hosts for that profile (aka. hostgroup):
Broaden the search for the whole pool of hosts (limited by the foreman query passed)

That way you'll have dynamic pools, **BUT**:
No minimums or maximiums specified

Future development

foreman_reserve

- Packaging (should be out really soon!)
- UI extensions (on the works)
- Internal improvements (like improved locking mechanism)
- Add pool configuraion ot the hostgroups (minimums, maximums ...)

Python-foreman

- Foreman API v2 Support
- Improve api generation (the api capabilities were being extracted from the autogenerated foreman docs, now foreman can expose the API structure in json format so we can use that directly)
- Proper testing
- Add value outside mirroring the API (add some extra sugar to make it easier to use)

Fabric tasks

- Open all the tasks we use internally
- Add VM creation to the provision to autoscale (now that template support is on Foreman)
- Migrate from ssh-based tasks to any other message oriented task execution framework

A word of possibilities

Don't just use the tools!!

If they fit
-> **Extend and improve**

If they do not
-> **Build your own, it's easy**

But if you decide just to use these tools, let me know!!
I'd be really happy to hear about it and help you solve any problem you find.
I'll even buy you a beer next time we meet (or a tea, or a coffe, or a waffle....)*

* Subject to cash availability of the presenter at the moment of the meeting, might be exchanged by a handshake.

Questions?

If you have any other questions, ideas, complaints or just want to reach out you can find me here:

By e-mail:

- dcaroest at redhat.com, david.caro.estevez at gmail.com

IRC:

- dcaro at #ovirt or #theforeman at freenode