

# Automating Ganeti

Automating Aspects of Ganeti administration.

© 2010-2011 Google

Use under GPLv2+ or CC-by-SA

Some images borrowed/modified from Lance Albertson and Iustin Pop

- What are the ways one can automate Ganeti? (shell and python)
- What is required to use the RAPI? (cluster name, password)
- Why is node homogeneity important?
- How can I use Puppet to configure a new node?

## Programatic control of Ganeti

Ganeti is all about automating the complex. You can write your own automation to control Ganeti.

## Use bash for small scripts

Ex: Find instances that are not using all the RAM allocated to them:

```
#!/bin/bash

ITEMS=$(gnt-instance list -o name,oper_ram,be/memory | awk '$2 != $3')
for i in $ITEMS ; do
    echo 'Why u no use your RAM, ' $i '?'
done
```

- "list" is faster, and easier to parse, than "info"
- gnt-\* commands don't return until the action is complete.
- Add --submit if waiting is not required.
- Submit long-running jobs with --priority=low

## RAPI

- RAPI is the Remote API.
- (not to be confused with the API used between masterd and noded)
- It is RESTful
- Client library hides all the details. You just need the cluster name and (for write access) credentials.
- <http://docs.ganeti.org/ganeti/current/html/rapi.html>

## Python Examples (1)

Read only requires no password:

```
import ganeti_rapi_client as grc

rapi = grc.GanetiRapiClient('cluster1.example.com')
```

```
print rapi.GetInfo()
print rapi.GetInstances(bulk=True)
```

Tip: Results are often long. Make them readable with pprint:

```
import pprint

pp = pprint.PrettyPrinter(indent=4).pprint
census = rapi.GetInstances(bulk=True)
pp(census)
```

## Python Examples (2)

Read/Write requires credentials:

```
import ganeti_rapi_client as grc

rapi = grc.GanetiRapiClient('cluster1.example.com')
rapi = grc.GanetiRapiClient(
    'cluster1', username='USERNAME', password='PASSWORD')

# Now "write" commands will work:
rapi.AddClusterTags(tags=['heuer'])
```

## ProTip: Your cluster is alive

**Bad:** Things could change between queries:

```
gnt-instance list -F 'pnode == "gntal"'
read -p 'Shut these down? ' ANS
if [[ $ANS == 'y' ]]; then
    gnt-instance list -F 'pnode == "gntal"' -o name \
        --no-headings |
        xargs -n1 gnt-instance shutdown
fi
```

**Good:** Make list and work from it:

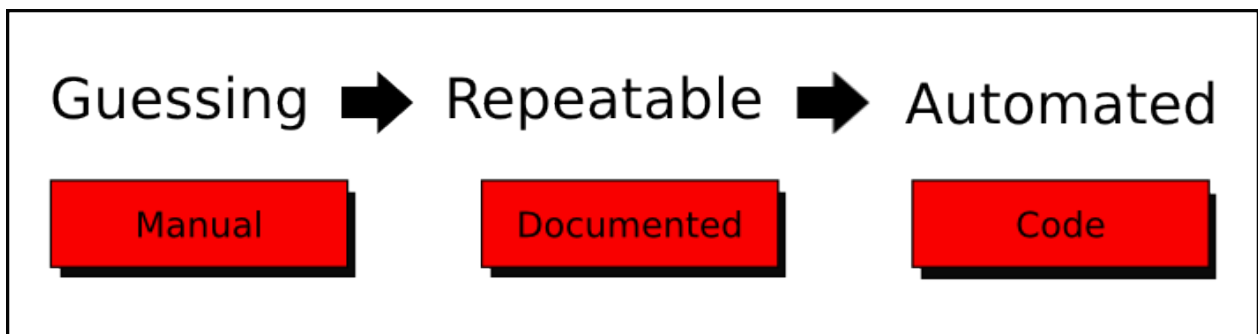
```
L=$(gnt-instance list -F 'pnode == "gntal"' -o name --no-headings)
echo $L ; read -p 'Shut these down? ' ANS
if [[ $ANS == 'y' ]]; then
    echo $L | xargs -n1 gnt-instance shutdown -f
fi
```

## Things to automate

- Adding instances of various types.
  - To insure all parameters are correct
- Periodic rebalances
  - Check to see if sufficiently unbalanced first

- Detect/fix DRBD issues
  - Find instances in degraded mode, stuck replication, etc.
- Workflow for evacuating a node
  - remove from monitoring system
  - evacuate primaries and secondaries
  - check to see if evacuation complete
  - print that it is safe to power off node for maintenance
- Configuring a node

## Automation Secret Formula



- Keep notes as you work
  - Once you "get it right", write a checklist
  - Automate the configuration of a node
    - Shell script
    - Configuration Management systems like CfEngine/Puppet/Chef.
- You'll be glad you did when you repeat a task months later.

## Automating node configuration

Configure nodes consistently: package versions, configuration files, network configuration

- Ganeti runs smoother: Fewer "UFO" problems.
- Easier to administer: Less to remember.

**Automate node configuration to achieve consistency.**

"A foolish consistency is the hobgoblin of little minds."

Ralph Waldo Emerson

"Don't be a fool, configure all nodes consistently."

Guido and Tom

## General strategy

1. Use PXE to install a "base OS"
2. Let installer partition the disks
3. Use CfEngine, Puppet or Chef to configure the host

## Puppet Tip 1:

Install specific version of a package, not 'latest':

- Reduces "surprise" upgrades in depot.
- Required for a "DEV -> QA -> PRODUCTION" strategy

```
package {  
  'xen-hypervisor-4.0-amd64': ensure => '4.0.1-5.2';  
  'ganeti2': ensure => '2.6.0-1';  
}
```

## Puppet Tip 2:

Add-ons like Augeas can edit complex configuration files:

```
augeas{"group_ganeti_settings" :  
  context => '/files/etc/default/grub',  
  changes => [  
    'set GRUB_DISABLE_OS_PROBER true',  
    'set GRUB_CMDLINE_XEN_DEFAULT \' "dom0_mem=512M" \',  
  ]  
}
```

Latest release understands the LISP-like format of xend-config.sxp and much, much more.

## Puppet Tip 3:

/etc/network/interfaces can be generated by template...

```
$primary_interface_name = ...  
$primary_interface_ip = ...  
$replication_interface_name = ...  
$replication_interface_ip = ...  
  
file {  
  path      => "/etc/network/interfaces",  
  owner     => root,  
  group     => root,  
  mode      => 644,  
  content   => template('interfaces-2nic.erb'),  
}
```

## Puppet Tip 3b:

...or use Augeas to edit /etc/network/interfaces in place:

```
augeas { "eth1":  
  context => "/files/etc/network/interfaces",  
  changes => [  
    "set auto[child::1 = 'eth1']/1 eth1",  
    "set iface[. = 'eth1'] eth1",  
    "set iface[. = 'eth1']/family inet",  
  ]  
}
```

```
"set iface[. = 'eth1']/method dhcp",  
],  
}
```

## Conclusion

Questions?

© 2010-2011 Google

Use under GPLv2+ or CC-by-SA

Some images borrowed/modified from Lance Albertson and Justin Pop

