Distributed CI and testing for CloudStack in a hybrid community

Daan Hoogland
Not senior. Not manager. Other job titles apply.
daan.hoogland@shapeblue.com
+DaanHoogland

https://www.slideshare.net/ShapeBlue/cccna17-distributed-ci-and-testing-for-cloudstack-in-a-hybrid-community
Contents

• About
  • You
  • Me
  • My employer

• Present – what is there at the moment
• Goal – a compatibility matrix
• Trillian (the shapeblue integration toolkit)
About Me

- **IT since 1990**
  Both operator and programmer since the beginning

- **2 adults as offspring**
  No kids

- **darts and triathlon**
  Active in the national federation(s)

- Likes rust better then go

- Schuberg Philis
- leaseweb
- Shape Blue
About You

• New Cloudstack user
• Infrastructure VAR or producer
• Integrator

• Interested in improvement of the platform
• Interested in contributing
  Quality insurance infrastructure
  Well tested code
“ShapeBlue are expert builders of public & private clouds. They are the leading global CloudStack services company.”
What’s in it

- Endgame (goal)
- Tools at the moment
- Trillian
• The goal of the community is to create a compatibility matrix of what works in CloudStack and in which environment, which will be easily maintainable and clear to users

• We have a hybrid community in terms of used hard- and software infrastructure. Testing everything is a challenge. (British understatement)
The compatibility matrix

• The compatibility matrix

• There must be a way to report
  • Compatible
  • Failing
  • Unstable
  • Untested (default)

• It must be reported per
  • ACS version
  • Feature
  • Context (whut?)
## Test run for PR 42

<table>
<thead>
<tr>
<th>Environment</th>
<th>Hypervisor</th>
<th>Networking</th>
<th>Extras</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShapeBlue 1</td>
<td>vmware</td>
<td>advanced</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Community env 1</td>
<td>xen</td>
<td>advanced</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cloudops test</td>
<td>xen</td>
<td>advanced</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interoute pre-production</td>
<td>vmware</td>
<td>advanced</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phil's betacloud</td>
<td>KVM</td>
<td>advanced</td>
<td>sdn/nicira</td>
<td></td>
</tr>
<tr>
<td>BT test bed</td>
<td>vmware</td>
<td>basic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PcXtreme</td>
<td>KVM</td>
<td>basic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anymous coward’s environment</td>
<td>KVM</td>
<td>Advanced</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*NOTA BENE: Do *not* test PRs, test merged PRs!*
Tools at the moment

• **Devcloud4**
• **Bubble** (kvm, yay)
• **Trillian** (vmware, but ansible \o/) more later

• **Github/Gitbox** (needs long and extensive explanation that will require an entirely new se
• **Bots** (blueorangutan, cloudmonger)
• **Upr** (https://github.com/cloudops/upr)
why

• Many, many, MANY permutations of a CloudStack deployment….

• Basic / Advanced
• Local / shared / mixed storage
• More than 8 common hypervisor types/versions
• 4 or 5 Management server OS possibilities

• That’s 144 combinations only looking the basics.

• Previous attempts at test automation haven’t been able to get coverage or capacity.
• Ability to support multiple concurrent environments
• No hardcoding to a specific (our) lab/environment
• As few separate components as possible
• Separation of ‘code’ and ‘configuration’
• Scripted command-line use fully supported
• Ability to support as many hypervisors as possible
• As flexible as possible
• Best ‘performance’ possible
• Best utilization of hardware resources possible
• **Trillian**
• Ansible based environment builder

• Multi-tenant (and multiple concurrent environments)
• CloudStack builds the Virtual Infrastructure
• Hypervisor hosts are CloudStack templates
• Supports multiple and mixed hypervisors
• Supports Advanced and Basic networks
• Open source
• **Blue Orangutan**
  • Build-bot

  • Automation endpoint to send build jobs to Jenkins for CentOS 6/7 and Debian packages from CloudStack source
  • Provides an integrated way to create packages from pull requests in Github.
  • Can kick-off Jenkins jobs to build and test environments based on packages which it builds.
when trilian result finish we use this to build the reports:
we build the reports here:
https://github.com/shapeblue/Trillian/blob/master/Ansible/roles/marvin/templates/smoketests.sh.j2
for posting comments, we use github apis keys etc.
that part's in the secret file some-scheme://some/path/to/blueorangutan/bo.py
conclusions

• **Add upr to bots**  Or behaviour like it
• **Several ways to our ideal**
• **Trillian is there as tool, proper reporting is key**
• **Next we need to orchestrate what test runs where**
• **Winston**
  • Test Orchestrator (Future project)

• Receives ‘offers’ of test type/capacity from remote test labs
• Takes request from Blue Orangutan to test packages and distributes tests to suitable remote labs.
• Offers might include:
  • Use of local storage
  • Use of basic vs advanced networks
  • Use of particular hypervisors/versions
Apache Cloudstack needs you
Guru tell me …?