THE INTEGRATED CLOUDS

Linking a Complex Infrastructure & Making it Work

Phil Avery
Architect
BJ's Wholesale
pavery@bjs.com
Blog: http://averytechguy.com

Matt St. Onge
Sr. Solutions Architect
Red Hat
mstonge@redhat.com
Twitter: @MattTheITGuru
Blog: http://MattTheITGuru.com

May 4th, 2017
Introduction
About BJ’s Wholesale

BJ’s Wholesale Club first introduced the warehouse club concept to the northeastern U.S. in 1984. Since then we’ve expanded, moving north to Maine, south to Florida and as far west as Ohio. Today, we operate over 213 Clubs, 131 Gas Stations, and our locations are across 15 States.

In-Club and online, BJ’s offers Members a huge selection of the very best products for home and business — from groceries, cleaning supplies and health & beauty to home goods, computers, electronics and more — at incredibly low prices every day.

BJ’s is ranked first for retail customer experience according to the 2017 Temkin Experience Ratings.
The Integrated Clouds
Linking a Complex Infrastructure & Making it Work

How BJ’s Wholesale found success in automation and integration

- Challenges
- The Solution
- Demo / Walkthru
- Q&A
The Challenges
Centralized Control and Governance
Centralized Control and Governance

Networking Management
Centralized Control and Governance

Drift Control & Config Management

Networking Management
Centralized Control and Governance

Networking Management

Patching / Errata

Rapid Scale-out of New Services

Drift Control & Config Management

Networking Management
Centralized Control and Governance

Rapid Scale-out of New Services

Drift Control & Config Management

Patching / Errata

Networking Management

Access Controls
We Need Some Better Tools...
All we’ve done is create NEW SILOS!
The Solution: Centralize The Automation Tools

- CONFIGURATION MANAGEMENT
- PATCH MANAGEMENT
- IPAM
- CENTRALIZED AUTHENTICATION
- SECURITY COMPLIANCE
- MONITORING
- TEMPLATES
- CLOUD-INIT
- CLOUDFORMATIONS
- SHOWBACK
CloudForms Delivers IT Management Across Platforms

- RBAC Enforced - Multi-group - Multi-cloud Management
- Provisioning with Quota Controls
- Automatic Retirement (Ability to over rule)
- Modification of Deployed Systems
- Capacity Planning
- Security / Compliance
- Approval Workflows
- Drift Management
- User/Self-service Portal with Extensive Service Catalog
- Centralized Cross-system Tagging
- Reporting
- Event-based Triggers (Especially when changes happen outside CF)
Our Third Party Integrations

- PHP IPAM
- Puppet Enterprise
- Red Hat Satellite 6
- Centrify
- Microsoft Azure
- AWS
- VMware
- RHEV
- Tenable/Nessus
- Ansible Core (On the CloudForms Appliance Itself)
RBAC Restrict Views

- EvmGroup-user
- EvmGroup-user_limited_self_service
- EvmGroup-user_self_service
- EvmGroup-vm_user

IT - UNIX

- Roles
- Tenants

- Diagnostics

- Database

- Parent Folder Path (VMs & Templates)
  - Datacenters/Somerville/vm/Discovered virtual machine
  - Datacenters/Somerville/vm/Linux/Dev
  - Datacenters/Somerville/vm/Linux/QA
  - Datacenters/Somerville/vm/Linux/Perf
  - Datacenters/Somerville/vm/Linux
  - Datacenters/Somerville/vm/Windows
  - Datacenters/Somerville/vm/Linux/Sandbox
  - Datacenters/Somerville/vm
  - Datacenters/Somerville/vm
  - Datacenters/Somerville/vm/Citrix
  - Datacenters/Somerville/vm/Appliance
Granular control over what groups or users can or cannot do
## Service Catalog

### 22 Results

<table>
<thead>
<tr>
<th>Service</th>
<th>Catalog Name</th>
<th>Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP_HANA</td>
<td>AWS SAP PROD</td>
<td>SAP Hana</td>
</tr>
<tr>
<td>SAP_RHEL7</td>
<td>AWS SAP PROD</td>
<td>SAP Business Objects</td>
</tr>
<tr>
<td>DB2_DEV</td>
<td>AWS-Ecom</td>
<td>IBM DB2</td>
</tr>
<tr>
<td>Ecom_dev_stack</td>
<td>AWS-Ecom</td>
<td>ENDECA_DEV</td>
</tr>
<tr>
<td>IHS_DEV</td>
<td>AWS-Ecom</td>
<td>IBM WebSphere</td>
</tr>
<tr>
<td>WAS_DEV</td>
<td>AWS-Ecom</td>
<td>Red Hat</td>
</tr>
<tr>
<td>CS_DEV_RHEL6.7</td>
<td>Bjs, Development</td>
<td>CentOS</td>
</tr>
<tr>
<td>CS_DEV_RHEL7.2</td>
<td>Bjs, Development</td>
<td>CentOS</td>
</tr>
<tr>
<td>CS_DEV_WIN2K12</td>
<td>Bjs, Development</td>
<td>Windows Server 2012</td>
</tr>
</tbody>
</table>
Default AWS Stack Provisioning Code

# Description: This method prepares arguments and parameters for orchestration provisioning

$evm.log("info", "Starting Orchestration Pre-Provisioning")

service = $evm.root["service_template_provision_task"].destination

# Through service you can examine the orchestration template, manager (i.e., provider)
# stack name, and options to create the stack
# You can also override these selections through service

$evm.log("info", "manager = #{service.orchestration_manager.name}(#{service.orchestration_manager.id})")
$evm.log("info", "template = #{service.orchestration_template.name}(#{service.orchestration_template.id})")
$evm.log("info", "stack name = #{service.stack_name}")

# Caution: stack options may contain passwords:
# $evm.log("info", "stack options = #{service.stack_options.inspect}")

# Example how to programmatically modify stack options:
#service.stack_name = 'new name'
#stack_options = service.stack_options
#stack_options[:disable_rollback] = false
#stack_options[:timeout_mins] = 2 # this option is provider dependent
#stack_options[:parameters]["flavor"] = 'ml.small'
# # Important: set stack_options
#service.stack_options = stack_options

# BJ's custom code
service.stack_name = "miq=#{service.id}"
## Results in AWS

<table>
<thead>
<tr>
<th>Stack Name</th>
<th>Created Time</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>miq-1000000000184</td>
<td>2017-04-03 21:53:38 UTC-0400</td>
<td>CREATE_COMPLETE</td>
<td>AWS CloudFormation Template for Ecommerce Development</td>
</tr>
<tr>
<td>miq-1000000000182</td>
<td>2017-04-03 21:52:25 UTC-0400</td>
<td>CREATE_COMPLETE</td>
<td>AWS CloudFormation Template for Ecommerce Development</td>
</tr>
<tr>
<td>miq-1000000000183</td>
<td>2017-04-03 21:52:25 UTC-0400</td>
<td>CREATE_COMPLETE</td>
<td>AWS CloudFormation Template for Ecommerce Development</td>
</tr>
<tr>
<td>miq-1000000000181</td>
<td>2017-04-03 21:50:47 UTC-0400</td>
<td>CREATE_COMPLETE</td>
<td>AWS CloudFormation Template for Ecommerce Development</td>
</tr>
</tbody>
</table>
CLOUDFORMS PROVISION FLOW

Start

- CustomizeRequest
  - If set to allow user to change
- Auto Approval or wait for Approval
- AcquireIPAddress
  - Request next free IP from IPAM and update database
- PreProvision
  - Detect best fit host based on various factors
- Provision
  - Initiate the instance creations and wait for completion

CheckProvision
- Ensure Server is built as expected

PostProvision
- Standard Post tasks like power on Vm

Configure VM, Cloud-init / VMWare
- Cloudformations. Set Hostname, I.P etc

Trigger Ansible Script for any pre orchestration work

Initial Puppet run, AD Integration/ Application Buildup

Email/Owner

End
CLOUDFORMS RETIREMENT FLOW

Start

- Triggered By Either User or Automated Retirement
- Run Local tasks on guest VMWare / SSH / Ansible
- Shutdown machine
- Release IP Address from IPAM
- Delete from Provider

- Email Owner that Box is beginning to be deleted
- Check Box is deleted
- Ansible Cleanup Playbook. (Puppet certs etc)
- Email Owner Retirement is completed
- Remove reference from Cloudforms

End
Demo / Walkthru
Q & A
Additional Resources

Phil Avery’s Blog
http://averytechguy.com

Peter McGowan’s Book (Mastering CloudForms Automation)
https://www.gitbook.com/@pemcg

Christian Jung’s Blog
http://www.jung-christian.de

The Official Red Hat CloudForms Blog
http://CloudFormsBlog.com

Matt St. Onge’s Blog
http://MattTheITGuru.com

Kevin Morey’s GitHub Page
https://github.com/ramrexx
THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHatNews
LEARN. NETWORK. EXPERIENCE OPEN SOURCE.