HA with Pacemaker Cloud

High availability management for cloud providers
http://pacemaker-cloud.org/

Pádraig Brady, Red Hat
Feb 4th, 2012

Overview

- Define Pacemaker Cloud's role using availability parameters
- Give a practical example of using Pacemaker Cloud along with other interesting technologies

High Availability

- A = MTBF / (MTBF + MTTR)
  - MTBF = Mean Time Between Failures
  - MTTR = Mean Time To Repair

  = Probability that system is operable at an unspecified time

  = 0 .. 1

- High Availability is achieved through the manipulation of MTBF and MTTR parameters of system design to meet availability requirements.

Techniques to increase Availability

- **Increase MTBF**
  - system specific, so outside scope
  - Just improve your software :)
- **Decrease MTTR**
  - React better to failure
  - Can benefit from automation
- **active-active**
  - Can also run systems in parallel to increase A
  - Used for very specialized apps, or low level like RAID
  - But complex and invasive to your stack
- **active-passive**
  - Passive system is used to decrease MTTR
  - Essentially the case we're considering
- **online calculator** http://www.pixelbeat.org/docs/reliability_calculator/
  - Example a VM needs a restart once a week, and takes around 30s to restart.
  - I.E. MTBF=168h and MTTR=0.01h (low MTBF, but also low MTTR)
  - active-active is parallel case
  - active-passive is single (not series) case
    - A = 0.99994 (4 nines)

active - passive

- **Traditionally the passive standby was "Hot" or "Cold"**
- **Hot standby**
  - Machine running in parallel
  - Can quickly assume the last known state of active
  - But...
    - Consumes resources for power and under utilized hardware
    - Implementation complexity for auto failover
  - Reduce MTTR (to minute range)
- **Cold standby**
  - Identical system in storage
  - Requires support staff to provision
    - For a standard computer: rack, swap disks, etc.
  - Reduce MTTR (to hour range)
- **Warm standby**
  - Cloud blurs the distinction between "hot" and "cold"
  - Can consider a new VM as a provisioned cold standby
  - Much reduced MTTR, as hardware is abstracted away
  - Swapping disks is now reconnect to shared storage etc.
  - Also have reduced resource usage as the standbys share hardware.
- **So there is a natural synergy between HA management and Cloud!**

Pacemaker Cloud

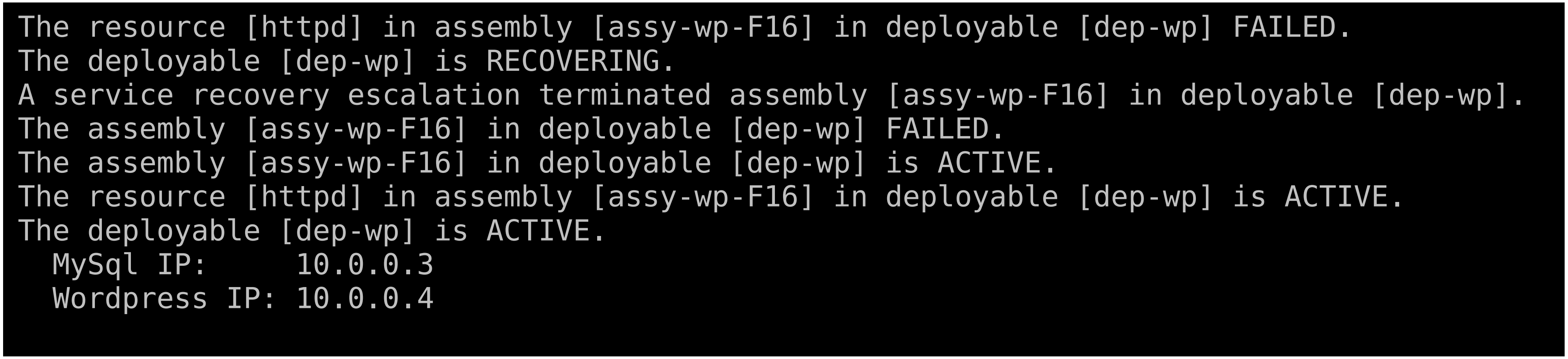
- **HA management can be modeled as a rules engine.**
  - events -> rules -> actions
- **events**
  - fault detection
  - Matahari agents currently used
- **rules**
  - modeled entities
    - cloud provider
    - group of VMs (deployable)
    - VM (assembly)
    - software service (application)
  - escalation
    - restart VM if app fails 3 times in 1 hour
    - restart deployable on new cloud provider, if...
  - Central rules (policy) engine from the pacemaker
    - Mature engine used in traditional clusters
- **actions**
  - Restart entity (provision the warm standby)
    - isolate, terminate, start new
  - Application control
    - matarhari agents currently used
  - VM and cloud control
    - Openstack, Aeolus, oVirt

Pacemaker Cloud + Openstack demo

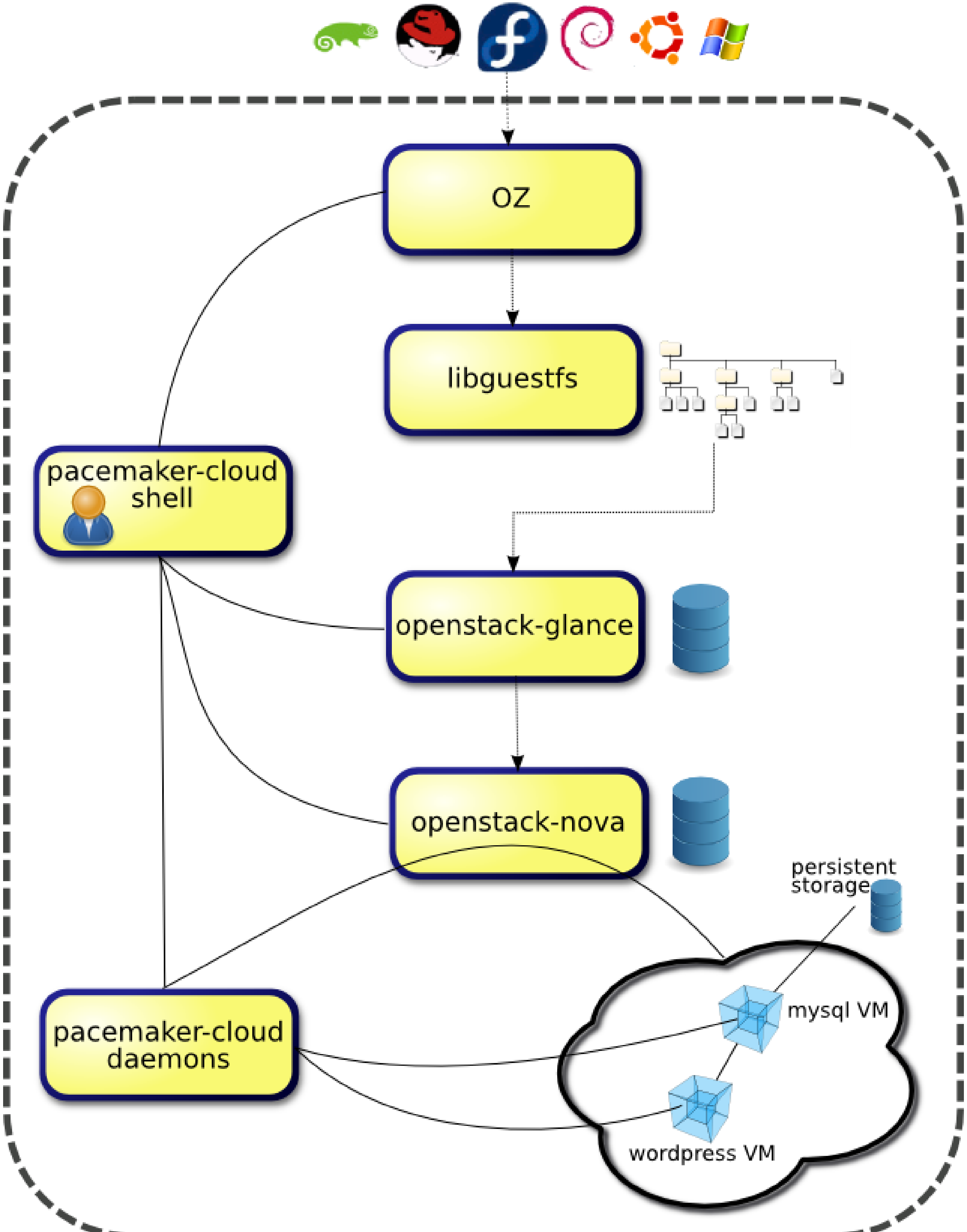
- http://www.pixelbeat.org/docs/pacemaker-cloud/
    - These instructions are an easy way to try the following in a practical way on Fedora 16
        - http://pacemaker-cloud.org/
        - http://openstack.org/
        - http://libvirt.org/
        - http://libguestfs.org/
        - http://aeolusproject.org/oz.html
    - Shows httpd restart, and escalation to VM restart

Demo overview

- Recipes for building VM images from upstream installation images
- Example is wordpress using 2 VMs and shared storage
- Monitoring and Notification for instances of those



```
The resource [httpd] in assembly [assy-wp-F16] in deployable [dep-wp] FAILED.
The deployable [dep-wp] is RECOVERING.
A service recovery escalation terminated assembly [assy-wp-F16] in deployable [dep-wp].
The assembly [assy-wp-F16] in deployable [dep-wp] FAILED.
The assembly [assy-wp-F16] in deployable [dep-wp] is ACTIVE.
The resource [httpd] in assembly [assy-wp-F16] in deployable [dep-wp] is ACTIVE.
The deployable [dep-wp] is ACTIVE.
  MySql IP:     10.0.0.3
  Wordpress IP: 10.0.0.4
```


OZ
libguestfs
pacemaker-cloud shell
openstack-glance
openstack-nova
persistent storage
pacemaker-cloud daemons
mysql VM
wordpress VM

Pacemaker Cloud Timeline

- **0.0.0** March 2011
  - Empty repo
- **0.4.0** July 2011
  - Released in Fedora 15
  - First basic implementation and architecture
  - F14 guest image support
- **0.5.0** Nov 2011
  - Released in Fedora 16
  - REST API to cped process for integration with other IAAS platforms
  - F15, F16 guest image support
- **0.6.0** Jan 2012
  - Openstack integration
  - resource and assembly escalation recovery
  - development of a multi-instance deployable with Mysql/Wordpress
  - U10, U11, RHEL guest image support
- **0.7.0** March 2012 - completed infrastructure
  - ssh-only monitoring dped version
  - direct integration with libdeltacloud in the dpe process
  - dependencies between resources/assemblies
  - reimplement cped into python for simpler IAAS platform integration
  - F17 guest image support
- **0.9.0+**
  - focus on integration with IAAS platforms
  - merge with distros beyond Fedora

Summary

Reliability modeling is easy and generally useful

http://www.pixelbeat.org/docs/reliability_calculator/

http://pacemaker-cloud.org/ reduces MTTR

The demo is an easy way to try OZ & Openstack etc.

http://www.pixelbeat.org/docs/pacemaker-cloud/