Build a “Domestic Cloud”

Fabrizio Manfredi Furuholmen

Beolink.org
Agenda

- Introduction
  - Definition
  - Characteristics
  - Deployment model
  - Delivery model

- openQRM
  - Components
  - Usage

- Eucalyptus
  - Components
  - Usage
What is cloud computing

- **Wikipedia (2007)**
  Cloud computing is a style of computing in which dynamically scalable and often Virtualized resources are provided as a Service over the internet.

- **Gardner (2009)**
  Cloud computing is a style of computing in which scalable and elastic IT-enabled capabilities are delivered as a service to external customers using Internet technologies.
Definition of Cloud Computing (version 14)

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three delivery models, and four deployment models.
Essential Characteristics

On-demand self-service

Ubiquitous network access.

Location independent resource pooling.

Rapid elasticity.

Measured Service.
Delivery model

- **Cloud Software as a Service (SaaS)**
- **Cloud Platform as a Service (PaaS)**
- **Cloud Infrastructure as a Service (IaaS)**
Deployment Models

- **Private cloud**
  The cloud infrastructure is operated solely for an organization

- **Public cloud**
  The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services

- **Hybrid cloud**
  The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities
Why setup a private cloud?

- Control
- Privacy and Security
- Utility Chargeback System
- Multi-tenancy
- Instant (Self) Provisioning Framework
- Temporary Infrastructure for Proof of Concept
- Cloud Management Infrastructure
Cloud comparison parameters

Virtualization / image format

Storage type/Access

Networking

Interface (API, Communication, ..)

Scheduler
Open-source data center management platform, for a automatic and appliance-based deployment,

- fully pluggable architecture
- monitoring
- high-availability
- cloud computing
- multiple virtualization technologies
- single-management console
- API
- Different storage support
openQRM Concepts

OpenQRM consists of:

- **Storage server**
  which can export volumes to your clients such as iSCSI or NFS volumes

- **A filesystem image**
  captured by openQRM, created, or generated yourself.

- **A Boot Image**
  from which the node boots, consisting of a kernel it's initrd and a small filesystem containing openQRM tools

- **A Virtual environment**
  which is actually the combination of a bootimage and a filesystem
openQrm: booting

- PXE Boot
- DHCP
- TFTP Server
- Kernel+initrd
- Openqrm script
- qrm agents
- Mount image/storage
openQrm: Deploy

- **Single deployment**
  1 image running on 1 machine

- **Shared deployment**
  Deploy the same filesystem on multiple machines, define pools where you need a number of resources per type filesystem and load balance between those instances

- **Partitioned deployment**
  Rather than using full machines you can partition a machine with different virtual machines. This can be done with the same filesystem images.
openQrm: Storage

- NFS
- iSCSI
- Aoe/Coraid
- NetApp
- Local-disk
  transferring server-images to the local-disk
- LVM-Nfs/iScsi/Aoe
  NFS on top of LVM2 to allow fast-cloning
  iSCSI on top of LVM2 to allow fast-cloning
  Aoe on top of LVM2 to allow fast-cloning
**openQrm: failover**

- **Automatic Hardware Failover (AHF)**
  AHF is a system level monitoring, when openQrm lose its heartbeat from a node for a specified period of time, openQRM will provision a new resource with that same image and try to kill the original resource to prevent conflict.

- **Automatic Application Recovery (AAR)**
  AAR uses application level checks to determine the health of the application. The actions AAR can trigger are either to reboot a particular resource, reboot all resources that belong to that Virtual Environment, or simply report the error.

- **openQRM High Availability**
  The HA pool can have multiple standbys.
openQrm: Policy Scheduler

- **Internal Policies**
  Internal thresholds of average utilization that determine when to add or remove resources from a VE.

- **External Policies**
  Drived by the user external policy can call 'qrm-cli virtual-environment' to add or remove resources.

- **Provisioning Request**
  Provisioning requests enables requesting resources and services to be a standard procedure. Once approved, a VE is created on a specific start date with the requested image and resources assigned. At the end date, the VE is stopped.
openQrm: Monitoring

- **Internal**
  - User define metrics gathered

- **Nagios (now also Zabbix)**
  - Integrates Nagios reports into the openQRM GUI.
  - Automatically configures Nagios sensors on resources. This is done by having a boot service which receives the list of sensors for each VE, as configured in the openQRM GUI, and activates them on the resource.
  - Reconfigures the Nagios server when assignments/de-assignments occur so it is aware of services running in the DC. The plug-in does this by registering event listeners.

- **Events**
  - server creates notifications to the user/plug-ins about various events in the managed resources
openQrm: Cloud interface

- **WebService API**
  - Provisioning / deprovisioning
  - Remote control openQRM cloud

- **Cloud Interface**
  - Fully automated provisioning cycle
  - External User portal for cloud-requests
  - Manual or automatic approval
  - Deployment to physical- or virtual machines
  - Sends mail in case of cloud-events
  - Integrate with Puppet

- **AWS integration (in development)**
Good for...

- Keep separation of "hardware" from "software"
- Consolidation and centralized management
- Fast prototyping and deploy
- High availability environments
Eucalyptus started as a research project by *University of California, (SB)*

- Open source software infrastructure that implements IaaS-style cloud computing. The goal of Eucalyptus is to allow sites with existing clusters and server infrastructure to host an elastic computing.
- Implementing on-premise private and hybrid clouds using the hardware and software infrastructure that is in place, without modification.
- API-compatible open-source clone of the Amazon AWS (Amazon Web Services) cloud platform
- Now Eucalyptus Systems (5.5 $ milion)
Eucalyptus: infrastructure

- **Cloud Controller (CLC)**
  includes the front-end services (Cloud Controller) as well as the Walrus storage system.

- **Cluster Controller (CC)**
  provides support for the virtual network overlay

- **Node Controller (NC)**
  interacts with KVM/XEN to manage individual VMs
**Eucalyptus : storage**

- **Walrus**
  is a storage service included with Eucalyptus that is interface compatible with Amazon's S3. Walrus allows users to store persistent data, organized as buckets and objects.

- **Elastic Block Storage**
  The Block Storage Service in Eucalyptus is interface-compatible with Amazon's Elastic Block Store.
Eucalyptus: Walrus

- **Supported Features**
  - Rest authentication
  - Operation on the service buckets and object (including acl and metadata)

- **Additional**
  - Image management (upload, decryption, download)
  - Web based configuration

- **Unsupported Features**
  - Virtual hosting of buckets
  - Location constraints (no replication, ha,..)
  - BitTorrent support
sub getAWSAuthorization
{
    my ($HTTPVerb,$bucket)=@_;  
    my $template="%a, %d %b %Y %T %z";
    my $Date = time2str($template, time);
    my $aws_access_key_id = "your id here";
    my $aws_secret_access_key = "your key here";
    my $host = "s3.amazonaws.com";
    my $ContentMD5="";
    my $CanonicalizedResource = "/";
    $CanonicalizedResource.==$bucket if $opt{b};
    $CanonicalizedResource.==$key if $opt{k};
    my $stringToSign;
    $stringToSign =$HTTPVerb."\n".$ContentMD5."\n";
    $stringToSign.==$ContentType if $ContentType;
    $stringToSign."\n";
    $stringToSign.==$Date."\n";
    $stringToSign.==$CanonicalizedAmzHeaders if $CanonicalizedAmzHeaders;
    $stringToSign.==$CanonicalizedResource;

    print "$stringToSign:".$stringToSign."\n" if $opt{d};

    my $hmac = Digest::HMAC_SHA1->new($aws_secret_access_key);
    $hmac->add($stringToSign);
    my $signature = $hmac->b64digest;
    my $authorization = "$AWS"." "$aws_access_key_id." "$signature."=";
    ($authorization,$Date);
}

sub doAWSRequest
{
    my ($authorization,$Date,$HTTPVerb,$bucket)=@_;  
    my $ua=LWP::UserAgent->new;
    my $url="http://s3.amazonaws.com/";
    $url.==$bucket if $bucket;
    $url.==$key if $key;
    print STDERR "URL: $url\n" if $opt{d};
    my $req = HTTP::Request->new($HTTPVerb=>$url);
    $req->header( "Date" => $Date );
    $req->header( "Authorization" => $authorization);
    $req->header( "Content-Type" => $ContentType );
    $req->header("x-amz-acl" => "public-read") if $opt{p};
    if ($HTTPVerb eq "PUT" && $opt{f} && -e $opt{f}){
        my $data=read_file($opt{f});
        $req->content($data);
    }

    print STDERR "request->as_string: ".$req->as_string if ($opt{d});

    my $res=$ua->request($req);
    if ($res->is_success) {
        print STDERR localtime()." SUCCESS: ".$res->status_line."\n" if ($opt{d});
    }else{
        warn localtime()." FAILED: ".$res->status_line."\n";
        print STDERR $res->content."\n";
    }
    $res;
}
Unlike S3, which only provides persistent file storage, EBS provides persistent block storage. Now you have random access to all the content in a volume's file system with EBS.

**User can:**
- defined storage size of each EBS Volume
- create an EBS Snapshot of an EBS Volume at any point in time.
- Volumes can only be mounted by one instance at any time
- Instances can only attach EBS Volumes that are in the same broadcast network of CLC.
- Multiple EBS Volumes can be attached to the same instance.
Eucalyptus: networking

- **SYSTEM Mode**
  In this mode, Eucalyptus simply assigns a random MAC address to the VM instance before booting and attaches the VM instance's ethernet device to the physical ethernet.

- **STATIC Mode**
  'map' of MAC address/IP Address pairs.

- **MANAGED Mode**
  Eucalyptus will maintain a DHCP server with static mappings for each VM instance that is created. Eucalyptus users can define a number of 'named networks', or 'security groups', to which they can apply network ingress rules that apply to any VM that runs within that 'network'.

- **MANAGED-NOVLAN Mode**
  This mode is identical to MANAGED mode in terms of features (dynamic IPs and security groups) but does not provide VM network isolation.
Eucalyptus: Scheduler policy

- **GREEDY**
  first node that is found that can run the VM will be chosen

- **Round Robin**
  nodes are selected one after another until one is found that can run the VM.
Good for ..

- Rapid prototyping and deploy infrastructure
- Debugging and/or application development purposes.
- Commercial web development services that target AWS (such as Rightscale)
- Simple installation and Administration (Ubuntu Enterprise Cloud)
Don’t forget...

Eucalyptus is Not as Scalable as Amazon

No enterprise ready

No fail-over, redundancy, load-balancing, etc.

Eucalyptus implements most of EC2 and a little of s3
## Comparison

<table>
<thead>
<tr>
<th>Cloud</th>
<th>VM type</th>
<th>Storage</th>
<th>Networking</th>
<th>Interface</th>
<th>Scheduler</th>
<th>VM Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eucalyptus</td>
<td>Xen/KVM</td>
<td>Wirlus/Ebs</td>
<td>Security Group, elastic IP</td>
<td>CLI, Web portal</td>
<td>First Fit, round robin</td>
<td>start, list, stop, reboot, get console output</td>
</tr>
<tr>
<td>openNebula</td>
<td>Xen,KVM</td>
<td>Local file over ssh / nfs</td>
<td>none</td>
<td>CLI</td>
<td>Queuing system, advanced reservation preemption</td>
<td>Save, pause, resize, live migrate</td>
</tr>
<tr>
<td>Nimbus</td>
<td>Xen</td>
<td>? (file copy)</td>
<td>none</td>
<td>CLI</td>
<td>?</td>
<td>save</td>
</tr>
<tr>
<td>Aspen</td>
<td>Xen</td>
<td>CoW, snapshots distribution over nfs</td>
<td>none</td>
<td>CLI</td>
<td>Save, pause, resize</td>
<td>Policy, priority queuing system</td>
</tr>
</tbody>
</table>
The others

- Enomalism
  http://wiki.xensource.com/xenwiki/Enomalism

- Reservoir
  http://www.reservoir-fp7.eu/
I look forward to meeting you...

XVI European AFS meeting 2009
Rome: September 28-30

Who should attend:

- Everyone interested in deploying a globally accessible file system
- Everyone interested in learning more about real world usage of Kerberos authentication in single realm and federated single sign-on environments
- Everyone who wants to share their knowledge and experience with other members of the AFS and Kerberos communities
- Everyone who wants to find out the latest developments affecting AFS and Kerberos

More Info:

www.openafs.it
Thank you!

manfred@freemails.ch