

Technical White Paper

Ubuntu Enterprise Cloud Architecture

By Simon Wardley, Etienne Goyer & Nick Barcet – August 2009

© Copyright Canonical 2009



Overview

Ubuntu Enterprise Cloud (**UEC**) brings Amazon EC2-like infrastructure capabilities inside the firewall. The UEC is powered by Eucalyptus, an open source implementation for the emerging standard of the EC2 API. This solution is designed to simplify the process of building and managing an internal cloud for businesses of any size, thereby enabling companies to create their own self-service infrastructure.

As the technology is open source, it requires no licence fees or subscription for use, experimentation or deployment. Updates are provided freely and support contracts can be purchased directly from Canonical.

UEC is specifically targeted at those companies wishing to gain the benefits of self-service IT within the confines of the corporate data centre whilst avoiding either lock-in to a specific vendor, expensive re-occurring product licences or the use of non-standard tools that aren't supported by public providers.

This white paper tries to provide an understanding of the UEC internal architecture and possibilities offered by it in terms of security, networking and scalability.

Table of Contents

Overview.....	2
Vocabulary.....	4
Introduction.....	5
UEC elements.....	8
Cloud Controller.....	8
Walrus Storage Controller.....	9
Elastic Block Storage Controller.....	9
Cluster Controller.....	9
Node Controller.....	10
Security and networking	11
Authentication and Authorization.....	11
Authentication and authorization of users.....	11
Networking	12
SYSTEM Mode.....	12
STATIC Mode.....	12
MANAGED Mode.....	12
MANAGED-NOVLAN Mode.....	13
Machine InstanceIsolation.....	14
OS Isolation.....	14
Hypervisor machine Isolation.....	14
Amazon EC2 API.....	15
Hardware considerations.....	16
Minimal hardware requirements.....	17
References.....	18

Vocabulary

This section defines the various acronyms used throughout this document.

Acronym	Expansion	Definition
AMI	Amazon Machine Image	A file containing a predefined virtual machine to use as the base of an instance on the Amazon cloud.
API	Application Programming Interface	A set of functions that a programme or person can use to send request and receive results from another programme.
AWS	Amazon Web Services	The collection of services that Amazon provides which compose the Amazon cloud offering.
CC	Cluster Controller	See the “UEC Elements>Cluster Controller” section of this document.
CLC	Cloud Controller	See the “UEC Elements>Cloud Controller” section of this document.
EBS	Elastic Block Storage	See the “UEC Elements>Elastic Block Storage Controller” section of this document.
EC2	Elastic Cloud Compute	Name of the part of AWS that handles instantiating AMIs
IaaS	Infrastructure as a Service	One of the three base elements that form the common understanding of the word cloud, IaaS is the layer that provides virtual machine running and means for it to store data in a raw form.
MI	Machine Images	A file containing a predefined virtual machine to use as the base of an instance on UEC. UEC images are compatible in form and function with AMIs.
Minst	Machine Instance	A running instance of a virtual machine defined by an MI
NC	Node Controller	See the “UEC Elements>Node Controller” section of this document.
PaaS	Platform as a Service	One of the three base elements that form the common understanding of the word cloud, PaaS is the layer that provides advanced programming interfaces to manipulate data and present it to the outside world.
S3	Simple Storage Service	Amazon S3 (Simple Storage Service) is an online storage web service offered by Amazon Web Services. Amazon S3 provides storage through a simple web services interface and API.
SaaS	Software as a Service	One of the three base elements that form the common understanding of the word cloud, SaaS is the layer that provides a complete application service on a cloud that can be directly used by end users.
UEC	Ubuntu Enterprise Cloud	See this entire document for a complete definition.
WS3	Walrus Storage Controller	See the “UEC Elements>Walrus Storage Controller” section of this document. It can be used with the same API as Amazon's S3

Introduction

Ubuntu is the only Linux distribution to position itself as a true cloud OS with three initial components of our cloud strategy already released. Two of these components are aimed at the infrastructure layer of the computing stack (commonly called IaaS or Infrastructure as a Service) while one component is aimed at the software layer (commonly called SaaS).

The three components are known as:

- Ubuntu Server Edition on Amazon EC2 (**IaaS**)
- Ubuntu Enterprise Cloud powered by Eucalyptus (**IaaS**)
- UbuntuOne (**SaaS**)

Even though UbuntuOne is an important initiative for Canonical to deliver added functionality to its large user base, it should clearly be distinguished from the other two components as it focuses on consumer use of the cloud rather than the tools required to build cloud services. The rest of this document will focus on this latter case.

Ubuntu has a clearly stated mission; to select the best components from open source, to assemble and refine them, to encourage ecosystem development and to provide the best possible experience to our users. This approach enables Canonical to successfully leverage open source against many of the traditional software monopolies. Ubuntu provides a real and useable alternative to the operating system/productivity suite that once dominated the world.

Our cloud strategy follows this same mission; to select the best components from open source, to assemble and refine them, to encourage ecosystem development and to provide the best possible experience to our users whilst avoiding lock-in and the creation of monopolies in this new cloud industry.

Since John McCarthy & Douglas Parkhill in the 1960s, whilst many people, have predicted and described the development of the cloud computing industry, this transition from a product to a more service oriented and cloudy world has only begun in earnest during the last decade. This transition won't happen in one step and there exists many barriers to its current use such as governance, security of supply, lack of second sourcing options and trust. However, the shift has started and the overwhelming benefits of cloud computing (i.e. *elastic infrastructure, efficiency of resource utilisation, reduction of capital expenditure, focus on core activities*) is likely to accelerate this change. What this means is that more and more of the applications that we use today on our personal computers or servers will soon migrate to the cloud and self-service IT environments.

While the benefits of public clouds are numerous, free software advocates frequently warn of the risks that the use of proprietary software and data formats implies for the long term survival of their data. In the cloud, these risks are heightened and new risks such as a lack of transparency in relationships appear.

What will happen to your data/application if the cloud it is hosted on goes bust? Will you be able to shift it to another cloud swiftly? Who is actually running your cloud?

In the traditional software model, if your hardware manufacturer, O/S provider or software

vendor disappears then you may still have a few months or even years to migrate what is running to another platform. Your business continuity is not necessarily in immediate danger. In the cloud model, you need to be able shift almost instantaneously or else your business continuity is at stake. In the cloud world, business continuity can only be secured by having a variety of second sourcing options and ideally a marketplace of providers that you can switch between.

To create a truly competitive computing marketplace with portability and easy switching between providers requires a triumph of open APIs, open data formats and standards that are implemented through open-source reference models. This holds true whatever layer of the computing stack you are discussing.

Ubuntu is focused on building open-source software systems that will allow you and others to provide infrastructure, platform or application services. Ubuntu is not offering to host these services but instead we're defining an open-source stack (*an open-source reference model*) that any provider, whether internal to an organisation or external, can use to provide these services. By having multiple providers of the same service, risks to business continuity can be significantly reduced.

To be effective, these open-source reference models need to follow any emerging de facto cloud standards and allow for both switching from an internal infrastructure to an external one, and between external providers. However, cloud computing effects all layers of the computing stack from infrastructure to application. It is naïve to believe that a single reference model could apply to all layers of the stack equally, as each layer consists of subsystems built from lower layers. Instead, any attempt to create a standard with an open-source reference model would have to target a specific layer of the computing stack.

Because we had to start somewhere, we decided to focus on the infrastructure stack first. After a careful study of the current cloud market, we identified the EC2, EBS and S3 API as emerging de-facto industry standards. Our initial step in this space was therefore to offer official images of Ubuntu on the Amazon Web Service Elastic Compute Cloud (EC2).

Our second step was to select a software stack that would allow people to build their own cloud infrastructure which matched these de facto standards. Hence we selected Eucalyptus as the best open-source component, in terms of the quality of the implementation and in the willingness of the upstream project to share and work with us towards our vision.

With Ubuntu 9.04 Server Edition (*April 2009*), an enhanced version of Eucalyptus that uses the KVM hypervisor was integrated into the distribution. This allowed any user to deploy a cloud that matches the same API that AWS provides. This system is Ubuntu Enterprise Cloud (**UEC**). In conjunction with the release of UEC, we also created official Ubuntu images that would work both on AWS (*a Xen based hypervisor system*) and a UEC cloud (*a KVM based hypervisor system*)

UEC effectively demonstrated that the hypervisor layer does not need to be the same for portability to be achieved and that open source does provide viable tools for creation of a cloud. To help promote this, Canonical also announced a series of cloud specific services, from consulting to support and training, that are designed to help organisations get started with cloud computing.

In the next couple releases of Ubuntu, our goal is to have a complete infrastructure layer for the cloud that will let anyone deploy their own cloud whether it is for their own usage (*private cloud*), or public consumption.

The private cloud concept is worth highlighting as it provides many of the benefits of cloud computing (*e.g. efficient utilisation of resources*) whilst reducing many of the transitional risks associated with public provision. There are a number of specific use cases where private clouds are of particular interest, including :-

- A private cloud offers a company the ability to quickly develop and prototype cloud-aware applications behind the firewall. This includes the development of privacy-sensitive applications such as credit card processing, medical record database, classified data handling, etc.
- High-performance applications whose load varies over time will benefit from being run on a platform that is “elastic”. Instead of having your IT infrastructure built for the sum of all the peak loads of different application, you can build a cloud infrastructure for the aggregated peak load at a single point in time instead. Furthermore opportunities exist to burst from a private cloud to a public environment in times of peak load.
- Self-Service IT: Using a private cloud technology, organisations can now put together a pool of hardware inside the firewall, a set of standard base images that should be used, and provide a simple web interface for their internal users to create instances on the fly. This should maximise the speed of development and testing of new services whilst reducing the backlog on IT.

The industry is in a time of transition and the benefits of cloud computing (*both private and public*) promise to significantly disrupt existing models. UEC is designed to help companies navigate safely in this sea of change.

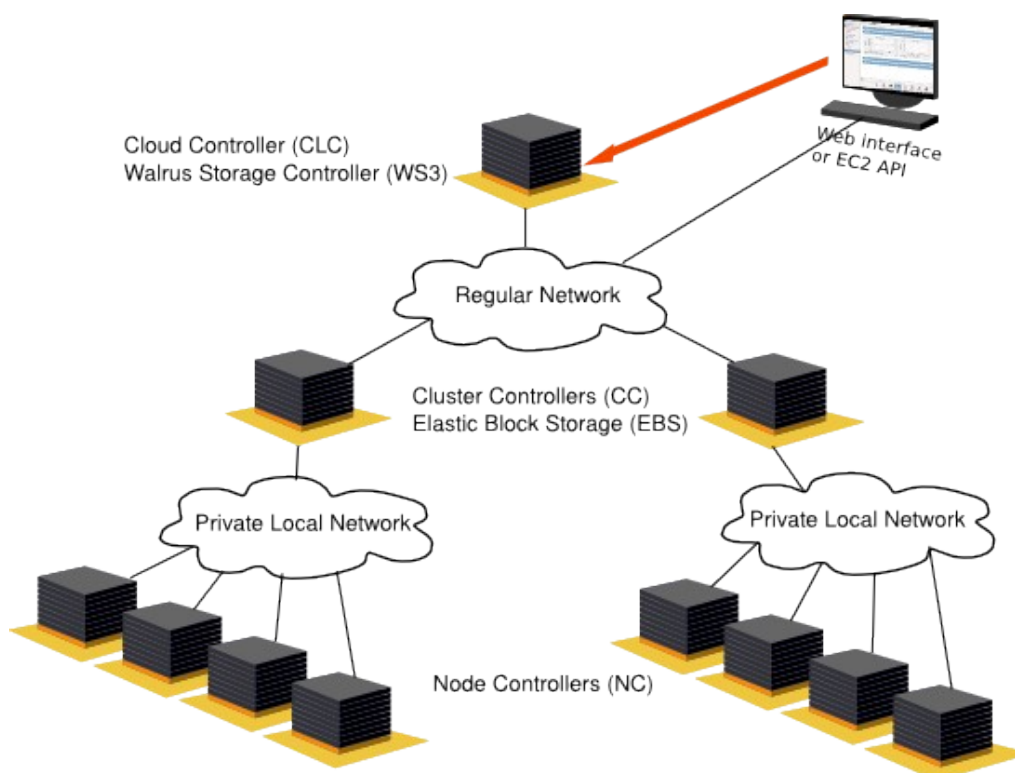
The following paper provides explains the UEC architecture as of Ubuntu 9.04.

UEC elements

The architecture of Eucalyptus, which is the main component of Ubuntu Enterprise Cloud, has been designed as modular set of 5 simple elements that can be easily scaled:

- Cloud Controller (CLC)
- Walrus Storage Controller (WS3)
- Elastic Block Storage Controller (EBS)
- Cluster Controller (CC)
- Node Controller (NC)

Each element is acting as an independent web service that exposes Web Service Description Language (WSDL) document defining the API to interact with it. It is a typical web service architecture.



Note: Regular network in this diagram does not necessarily mean the Internet, but a network that is generally accessible to the users of the cloud.

Cloud Controller

The Cloud Controller (CLC) is the most visible element of the Eucalyptus architecture, as it is providing the interface with which users of the cloud interact. This interface is comprised of a standard SOAP based API matching the Amazon EC2 API (see Amazon EC2 API below), a simpler "Query Interface" which euca2ools and ElasticFox uses and a traditional web interface for direct user interaction.

The CLC talks with the Cluster Controllers (CC) and makes the top level choices for allocating new instances. This element holds all information linking users to running instances, the collection of available machines to be run, as well as view of the load of the entire system.

Walrus Storage Controller

The Walrus Storage Controller (WS3) implements a REST (Representational State Transfer) and a SOAP (Simple Object Access Protocol) API which are compatible with Amazon Simple Storage Protocol (S3). It is used for:

- Storing the machine images (MI) that can be instantiated on our cloud;
- Accessing and storing data (either from a running instance or from anywhere on the web).

WS3 should be considered as a file level storage system. While it does not provide the ability to lock a file or portion of a file, users are guaranteed that a consistent copy of the file will be saved if there are concurrent writes to the same file. If a write to a file is encountered while there is a previous write in progress, the previous write is invalidated.

Currently, the machine on which the Cloud Controller runs also hosts the Walrus Storage Controller (WS3), but this limitation will be removed in a forthcoming version.

Elastic Block Storage Controller

The Elastic Block Storage Controller (EBS) runs on the same machine(s) as the Cluster Controller and is configured automatically when the Cluster Controller is installed. It allows to create persistent block devices that can be mounted on running machines in order to gain access to virtual hard drive. Storage volumes behave like raw, unformatted block devices, with user supplied device names and a block device interface. You can create a file system on top of EBS volumes, or use them in any other way you would use a block device.

EBS also provides the ability to create point-in-time snapshots of volumes, which are stored on WS3. These snapshots can be used as the starting point for new EBS volumes and protect data for long-term durability. The same snapshot can be used to instantiate as many volumes as you wish.

At the network level, the block device is accessed using ATA over Ethernet (AoE). Since packets cannot be routed, this requires that the EBS controller and the Nodes hosting machine images which are accessing it, to be on the same Ethernet segment. It is planned to add a more flexible protocol in a future version such as iSCSI.

Cluster Controller

The Cluster Controller (CC) operates as the go between between the Node Controller and the Cloud Controller. As such, it needs to have access to both the Node Controller and Cloud Controller networks. It will receive requests to allocate MI (*machine images*) from the Cloud Controller and in turn decides which Node Controller will run the Minst (*machine instance*). This decision is based upon status reports which the Cluster Controller receives from each of the Node Controllers. It can also answer requests from the Cloud Controller asking for its left over

capacity to run specific instance types, hence allowing the Cloud Controller to decide on which cluster to run new instances.

The Cluster Controller is also in charge of managing any virtual networks that the MInst run in and routing traffic to and from them. Its precise role greatly depends on the networking model chosen to run MInst, which we will describe later in this document in the Networking and Security section.

As described above, the Cluster Controller also runs the EBS Controllers.

As a whole, the group formed of one Cluster Controller and EBS Controller and a variable number of Node Controller constitutes the equivalent of Amazon's "availability zones".

Node Controller

The Node Controllers' (NC) software runs on the physical machines on which the MI will be instantiated. The NC software role is to interact with the OS and hypervisor running on the node, as instructed by the Cluster Controller.

The Node Controller's first task is to discover the environment on which it runs in term of available resources (disk space, type and number of cores, memory), as well as running VMs that could be started independently of the NC, CC, and CLC.

The Node Controller will then wait for and perform any requested tasks from the Cluster Controller (start and stop instances) or replies to availability queries. When requested to start a MI, it will:-

1. Verify the authenticity of the user request;
2. Download the image from WS3 (images are cached so that starting multiple instances of the same machine image only downloads that image once);
3. Create the requested virtual network interface;
4. Start the instance of the machine image running as a virtual machine (VM).

Stopping a virtual machine corresponds to performing the opposite operations in the the order 1, 4, 3.

Security and networking

Security in UEC can be summarised in three basic layers which closely interact with each other:

- Authentication and Authorization
- Network isolation
- MInst isolation

These are described in more details in this section.

Authentication and Authorization

Two type of “actors” need to be authenticated and authorized on UEC:

- The users or administrators of the system which have specific rights to modify the system and start and stop instances;
- The components of the system (NC, CLC, CC) which need to trust each other when transmitting requests.

On a general level, the authentication is performed using locally generated X509 certificates, as cryptographic keys to authenticate and secure communications between all actors with communication based on the WS-Security policy framework. This is true for all internal communication within the cloud. Users authenticate either with an X509 certificate or a Query type key.

The initial addition of Cluster Controller or Node Controller to the Cloud Controller environment requires using a password to exchange the cryptographic keys from the lower controller to the upper one. Once this is done, all operations rely on the trust provided by these keys in the communications between the controllers.

Authentication and authorization of users

Initial user account creation of user is performed in a two step operation:

- First, any user with access to the Cloud Controller user interface can fill a form to request an account;
- When a request is received, it is up to the administrator to grant access to the user according to its own policy.

Users which have been granted access retrieve the certificate and query type key that have been generated for them and which are used for all their requests performed through the APIs. The type of authentication (query key or certificate) will vary based on the tool used to access the cloud, their password only being used to access the web console that allows them to retrieve their certificate and query key.

Note: a new certificate and query key is generated each time the user ask to retrieve them.

The Cloud Controller holds the central right repository for each users. Every time a request arrives at a controller (CLC, CC or NC), it is its duty to verify that the user from which the

request originated is duly authorised to perform it. Users obviously only make requests to the Cloud Controller, so authentication is only performed there, but authorisation is verified at each level to prevent abuse.

Networking

Networking is a very important part of the overall security of the UEC system, as it might be crucial to prevent eavesdropping of network traffic from a machine run by one user to the machines run by other users. Depending on the level of security that a setup may require, the following networking modes are available at set-up time:

SYSTEM Mode

This is the simplest networking mode, but also offers the smallest number of networking features. In this mode, UEC simply assigns a random MAC address to the VM (virtual machine) instance before booting and attaches the VM instance's Ethernet device to the physical Ethernet through the node's local bridge. VM instances typically obtain an IP address using DHCP, the same way any non-VM machine using DHCP would obtain an address. Note that in this mode, the UEC administrator (or the administrator that manages the network to which UEC components are attached) must set up a DHCP server that has a dynamic pool of IP addresses to hand out as VMs boot. In other words, if your laptop/desktop/server gets an IP address using DHCP on the same network as the UEC nodes, then your VMs should similarly obtain addresses. This mode is most useful for users who want to try out UEC on their laptops/desktops.

STATIC Mode

This mode offers the UEC administrator more control over VM IP address assignment. Here, the administrator configures UEC with a 'map' of MAC address/IP Address pairs. When a VM is instantiated, UEC sets up a static entry within a UEC controlled DHCP server, takes the next free MAC/IP pair, assigns it to a VM, and attaches the VMs Ethernet device to the physical Ethernet through the bridge on the nodes (in a manner similar to SYSTEM mode). This mode is useful for administrators who have a pool of MAC/IP addresses that they wish to always assign to their virtual machines.

NB: Running UEC in SYSTEM or STATIC mode disables some key functionality such as the definition of ingress rules between collections of VMs (termed security groups in Amazon EC2), the user-controlled, dynamic assignment of IPs to instances at boot and run-time (Elastic IPs in Amazon EC2), isolation of network traffic between VMs (that is, the root user within VMs will be able to inspect and potentially interfere with network traffic from other VMs), and the availability of the meta-data service (use of the `http://169.254.169.254/` URL to obtain instance specific information).

MANAGED Mode

This mode is the most feature-full but also carries with it the most potential constraints on the setup of the UEC administrator's network. In MANAGED mode, the UEC administrator defines a large network (usually private and un-routable) from which VM instances will draw their IP addresses. As with STATIC mode, UEC will maintain a DHCP server with static mappings for

each VM instance that is created. UEC 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'. When a user runs a VM instance, they specify the name of such a network that a VM is to be a member of, and UEC selects a subset of the entire range of IPs that other VMs in the same 'network' can reside. A user can specify ingress rules that apply to a given 'network', such as allowing ping (ICMP) or ssh (TCP port 22) traffic to reach their VMs. This capability allows UEC expose a capability similar to Amazon's 'security groups'. In addition, the administrator can specify a pool of public IP addresses that users may allocate, then assign to VMs either at boot or dynamically at run-time. This capability is similar to Amazon's 'Elastic IPs'. UEC administrators that require security groups, Elastic IPs, and VM network isolation must use this mode.

MANAGED-NOVLAN Mode

This mode is identical to MANAGED mode in terms of features (Elastic IPs and security groups) but does not provide VM network isolation. Administrators who want Elastic IPs and the security groups, but are not running on a network that is 'VLAN clean'¹ or don't care if their VMs are isolated from one another on the network should choose this mode.

The following summarises the features available in each mode:

Mode	Isolation	Elastic IP	Security Group	Auto DHCP
SYSTEM				
STATIC				
MANAGED	✓	✓	✓	✓
MANAGED-NOVLAN		✓	✓	✓

¹We have often found that managed switches configured to strip VLAN tagging on frames, are still passing tags through, or are dropping all the frames entirely, which will render them incompatible with the MANAGED mode unless properly reconfigured.

Machine Instance Isolation

Machine instances (MInst) isolation can be provided at three levels:

- Networking isolation, which is described in the previous section and is available in manage mode;
- OS isolation, which can be provided by a Mandatory Access Control (MAC) system such as AppArmor;
- Hypervisor based machine isolation, which is intrinsic to the hypervisor technology used.

OS Isolation

The use of of Mandatory Access Control system, such as AppArmor which is the recommended default in Ubuntu, can greatly reduce the risk of MInst accessing resources which should normally be beyond the reach of a given VM. While the current 9.04 release of Ubuntu Server Edition does not provide default scripts to perform this isolation, an administrator can write and debug their own scripts using the AppArmor tools. Starting with 9.10 , a default AppArmor script will be automatically protecting the underlying OS and the other MInsts running on it by linking a default AppArmor script to the creation of any new instance using the libvirt library (the library that UEC calls to start new MInst).

Hypervisor machine Isolation

UEC's implementation of Eucalyptus is based on the KVM hypervisor, as it is the preferred and officially maintained hypervisor in Ubuntu since 8.04 LTS. KVM relies completely on the added hypervisor ring level isolation to processor that supports it (Intel-VT or AMD-V processor built-in technologies). While no technology is ever completely secure, the use of this ring level is normally limited to the hypervisor itself and prevents, by default, any interaction between running instances on the same physical machine.

Amazon EC2 API

As with any new technology, there are no industrial standards but instead emerging de facto standards. There currently exists a multitude of “standard” interfaces for the provision of Infrastructure as a Service (IaaS) like technology. The Eucalyptus project made no assumption on which of the existing standards was likely to dominate this field and hence built an abstraction layer which should allow it to be invoked from any other API. As the Amazon EC2 is currently the de facto standard, the decision was made to implement it.

Ubuntu's engineering and management fully endorses this approach and the view that Amazon EC2 has emerged as the de facto standard. Since Amazon's does not currently provide their API and AMI tools under a license that would allow its use with other clouds, UEC now operates with a full re-implementation of the toolset under a GPL compatible license.

These tools, named Euca2ools, are based on open-sourced python libraries (Boto and M2Crypto). As Euca2ools were not ready by the time 9.04 was released, Euca2ools have to be installed manually at the moment.

More information on using Euca2ools can be found at:

<http://open.eucalyptus.com/wiki/Euca2oolsGuide>

Hardware considerations

Hardware requirements of the Cloud Controller and the Node Controllers needs to be treated separately, as their roles, and hence requirements, are radically different.

The Cloud Controller requirements can be somewhat hard to pin. The web management UI and Web Services handling the cloud management API calls, are served by a relatively lightweight Java application; any modern hardware will do for that purpose. The WS3 service (running on the CLC) and the EBS service (running on the CC), however, are more demanding. Note that WS3 might be made independent in a future version.

As the Walrus service implements persistent storage, you want to have a fast storage subsystem and ample disk space. Also, in production use, you will want resiliency built into your storage (hardware RAID, for example). As the network will already be heavily taxed on the front-end, you may want to use a dedicated LAN iSCSI NAS or use fast local disk or Fibre Channel SAN. There is no hard and fast rule about storage, as the requirements will vary according to your application. The same NAS/SAN might be used as an attachment for the EBS based storage on the CC.

The regular network² capacity will be heavily taxed. All WS3 request will be going through the network. Moreover, if you choose to implement network ingress filtering and Elastic IP for static address, all network traffic between security groups, and between instances and network outside the cloud, will be routed through the front-end (the Cloud Controller, more precisely). If you do not want the front-end to become a bottleneck for network traffic, you will have to provision enough capacity for twice the projected amount of traffic between instances of different security groups (in and out), plus enough bandwidth for communication with hosts outside the cloud. To achieve that, you may want to consider 10G Ethernet or Ethernet interface bonding (sometime referred to as “Ethernet trunk” or “NIC teaming”) to ensure adequate network capacity. Also pay attention to the capacity of the switch to which the front-end is connected: it has to have a fabric fast enough for the throughput anticipated. The Cluster Controller follows the same rule, but its traffic will be limited to the nodes it controls.

The Node Controller’s requirements are much more straightforward. Essentially, the Node Controller needs to be able to run the KVM hypervisor in native virtualization mode. For that, the CPU on the Node Controller needs to have either the Intel VT or AMD-V extensions. Most server-class hardware built since 2006 should meet this requirement. Typically, these extension needs to be enabled in the machine’s BIOS. In term of disk capacity, you will need enough space to cache the MI locally and for the running instance disk image. Reasonably, 200GB of available space on the Node Controllers should be enough for most scenario, with the exception of cloud application where very large AMI are required. It is also worth noting that Node Controllers do not have to run on homogeneous hardware; you can use any mix of machine for your Node Controllers.

In typical EC2 fashion, you have to determine the resources allocated to an instance based on a profile, which dictate the amount of RAM, disk and CPU cycle it may use. Note that the largest

² Regular network in the sense depicted in the architecture diagram at the beginning of the section “UEC Elements”.

instance you will be able to run depends on resources available on the Node Controllers. For example, you cannot run an instance with 8 GB of RAM if none of your Node Controllers have that much RAM available. As such, if you plan to run a number of “large” instance, you will need to provision Node Controllers accordingly. This is why it is usually easier to run a large number of “small” instances instead of a small number of “large” instances; it is also more flexible.

If you wish to use ingress network filter (MANAGED mode), you will also need to verify that your network is “VLAN-clean”, that is, it passes Ethernet frame with VLAN tag intact. Managed switches will often be configured to strip to VLAN tag on frame it pass through, or to drop these frame entirely. As security groups is implemented in Eucalyptus using special VLAN interface, you will need to configure your switches to pass Ethernet frames between the Node Controller and the front-end as-is.

Minimal hardware requirements

Type	RAM	Disk	Intel-VT/AMD-v	Network
CLC	512 Mo	100 Go		Gigabit Ethernet
CC	512 Mo	100 Go		2 Gigabit Ethernet
NC	2 GB	20 Go	✓	Gigabit Ethernet

References

- The Eucalyptus Open-source Cloud-computing System, Daniel Nurmi, Rich Wolski, Chris Grzegorzczuk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, in Proceedings of 9th IEEE International Symposium on Cluster Computing and the Grid, Shanghai, China, <http://open.eucalyptus.com/documents/ccgrid2009.pdf>
- Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems, Daniel Nurmi, Rich Wolski, Chris Grzegorzczuk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, UCSB Computer Science Technical Report Number 2008-10 (August 2008) http://open.eucalyptus.com/documents/nurmi_et_al-eucalyptus_tech_report-august_2008.pdf
- Eucalyptus documentation: <http://open.eucalyptus.com/wiki/Documentation>
- Ubuntu Enterprise Cloud deployment guide Ubuntu's documentation Wiki: <https://help.ubuntu.com/community/Eucalyptus>
- Euca2ools guide: <http://open.eucalyptus.com/wiki/Euca2oolsGuide>
- Amazon EC2 documentation: <http://aws.amazon.com/ec2/>

Every effort has been made by Canonical to ensure the accuracy of this document but Canonical disclaims, to the extent possible at law, any liability for any error or omission.

© Canonical Limited 2008. Ubuntu and associated logos are registered trademarks of Canonical Ltd., all rights reserved. All other trademarks are the properties of their respective owners. Any information contained in this document may change without notice and Canonical is not held responsible for any such changes.