東海大學

資訊工程研究所

碩士論文

指導教授: 楊朝棟博士

以虛擬化叢集及 DRBD 建構雙重高可用性雲端服務

The Implementation of Double High Availability Cloud Service with Cluster Virtualization and DRBD

研究生: 張廣欽

中華民國一〇四年七月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 張 廣 欽 所提之論文

以虛擬化叢集及 DRBD 建構雙重高可用性雲端服務

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 楊武 簽章

委 員 劉榮春

許慶賢

指 導 教 授 楊朝棟 簽章

中華民國 104 年 7 月 7 日

摘　要

隨著雲端資訊服務上的發展，為了能不間斷地提供各項資訊服務，對伺服器及儲存空間的可用性需求就越來越顯得重要。當在發生軟硬體錯誤時，如何盡可能縮短停機時間，於系統當機和失敗期間持續維持服務可用，提昇資訊系統的可靠程度，將有問題的執行個體重新導向正常運作的執行個體，使系統服務不會停頓並能夠穩定持續地提供資訊服務就成了很重要的課題。隨著虛擬化技術的日新月異，將資訊系統主機虛擬化叢集化，可以確保硬體的高可用性。若能在某實體機發生問題時，可以將上方的資料備分至其他實體機上，讓備用機器能夠即時恢復已經停止服務的虛擬機。加上 DRBD（Distributed Replicated Block Device），可即時同步所有的資料，當軟體發生異常時，可立即啟用另一個資訊系統來取代原先提供服務的系統，如此即可達到雙重保障的效用。本論文將介紹透過結合 Virtualization Cluster 及 DRBD 技術，以達成資訊服務不中斷的雙重保障。在實作中，分別探討以 VMware 及 OpenStack 作為虛擬化叢集平台，在容錯轉移機制下來處理實體主機失效，結合 DRBD 處理軟體失效後，在各種軟硬體失效的情況下的差異。並驗證本論文提出的架構在實際應用上，此雙重高可用性雲端服務可在硬體失效及軟體失效中自動回復，以確保雲端服務不會受到影響，縮短服務中斷的時間。

關鍵字：高可用性, 虛擬化叢集, 容錯轉移

Abstract

More and more campus information services are now become cloud service. In order to let the information services uninterrupted, the need for the availability of servers and storage space is getting more important. To achieve that, following subjects need to be discuss: How to minimize downtime, when a hardware failure occurs. How to improve the reliability of information systems. How to build a robust system, with stabilize service, and be able to provide IT service all the time. To figure out solutions of those questions has become very important topics. With host virtualization cluster, it can ensure high availability of hardware. If a physical machine occurs problem, the virtual machine can move to the other entities of the physical machine. With DRBD (Distributed Replicated Block Device), instantly synchronize all the data, when an exception occurs the software can be immediately transfer to replace the system previously provided services, so you can achieve the double protection of the utility. This article would like to introduce a combination technology of Virtualization Cluster with High Availability and DRBD. With this technology, the researchers can build a demonstration of dual-high availability cloud service. In this work, the researchers compare the performance of two kind of virtualization clusters: VMware and OpenStack. To find out the differences and verify the efficiency of the dual-high availability cloud service.

Keyword：High Availability, Virtualization, Virtualization Fault Tolerance (VFT), DRBD

致謝詞

在長久的努力之後，終於能夠順利取得夢想中的學位，完成人生中的目標。除了感到非常的歡欣外。要非常感謝我的指導老師楊朝棟教授，指引我研究的方向，教導我雲端運算領域的相關知識。對於已經在工作的我來說，兼顧學業及工作本來就是不易的一件事，幸而有老師在工作及學業上給予我許多的幫助。並讓我得以將這些所學運用在工作之上，大大減輕了工作上的壓力，使得念研究所的過程及最終的收穫更顯珍貴。

在此也要感謝國立交通大學資訊工程學系楊武教授、國立台北大學資訊工程學系張玉山教授、中華大學資訊工程學系許慶賢教授、東海大學資訊工程學系劉榮春助理教授在口試期間，不厭其煩地給予學生諸多指導及許多修改建議，讓我的論文能夠更臻於完善，也讓我有茅塞頓開之感，啟發了許多不一樣的想法。

在求學過程中，更認識了許多的好朋友好同學，一同成長並分享彼此不管是在生活上或是學業上的經驗，更希望這個緣分不會隨著研究所的畢業而消散。

最後要深深地感謝的是在背後默默支持我、一直陪伴在身邊的家人及父母，讓我的生命更加精彩，是支撐我完成這個目標的最大力量。謝謝！

Table of Contents

List of Figures

List of Tables

Chapter 1

簡介

1.1 研究動機

隨著雲端資訊服務上的發展，為了能不間斷地提供各項資訊服務，對伺服器及儲存空間的可用性需求就越來越顯得重要。當在發生軟硬體錯誤時，如何盡可能縮短停機時間，於系統當機和失敗期間持續維持服務可用，提昇資訊系統的可靠程度，將有問題的執行個體重新導向正常運作的執行個體，使系統服務不會停頓並能夠穩定持續地提供資訊服務就成了很重要的課題。雲端服務仰賴硬體的強固性，穩定的主機及儲存空間決定了服務的可用性。隨著虛擬化技術的日新月異，將資訊系統主機虛擬化叢集化，可以確保硬體的高可用性。

身為服務提供者，服務的可用性 (Availability) 是一項重要的指標，若是服務中斷則將會違反使用者與我們訂立的契約，同時也造成使用者對此服務的不信任。使用虛擬化叢集技術可相當程度的免除因硬體故障或維修所造成的服務中斷。

為了實現主機硬體的高可用性，可採用各種虛擬化技術，開啟容錯轉移機制來處理實體主機的錯誤。但以目前現有的容錯技術來說，在虛擬機的管理上，只能對於所有已列入管理清單的虛擬機操作和支配，但無法了解虛擬機和實體機器上的主從關係，若有一台實體機器發生故障，其上方的無法提供服務的虛

擬機只能選擇其他虛擬機啟動來彌補發生錯誤實體機上的虛擬機。若能在某實體機發生問題時，可以將上方的資料備分至其他實體機上，讓備用機器能夠即時恢復已經停止服務的虛擬機。

1.2 論文目標與貢獻

本論文旨在運用虛擬化叢集及 DRBD 實作雙重高可用性雲端服務。在此架構中，分別採用了商業軟體 VMware® 的產品及 OpenStack™ 來搭建虛擬化叢集，搭配 DRBD 來實現雙重高可用性。此雙重高可用性雲端服務可在硬體失效及軟體失效中自動回復，以確保雲端服務不會受到影響，縮短服務中斷的時間，並比較商業軟體及 Open source 在此架構中之間的差異。

1.3 論文架構

第二章將說明本論文的研究背景，介紹目前虛擬化及 DRBD 相關技術，並探討現有的高可用性技術及可能面對的問題。第三章則描述系統的架構、建置實作。第四章說明實驗的環境、方法以及實驗的結果，並依照實驗的結果討論 VMware® 及 OpenStack™ 兩個系統的差異。在最後，第五章將說明本論文依照實驗結果做出的結論及未來可能的研究方向。

Chapter 2

研究背景與相關研究

2.1 研究背景

2.1.1 High availability

高可用性 HA(High Availability)，是指針對在軟硬體產生錯誤或損毀時，仍能繼續維持正常運作的技術。避免長時間之硬體維修或系統重設，縮短停機時間，並能快速回復系統服務，減少後續引起的損失。而其運作之方式主要為在錯誤發生時，HA 會自動接管資源，並且啟動備援軟硬體接替，透過一連串預先制訂好的程序將資訊服務回復。[21] [20] 高可用性的主要目的為減少停機時間，當單點故障 (Single Point Of Failure) 時，要讓整個系統繼續運作，以目前的技術來說，可以透過主機硬體虛擬化及建立容錯移轉叢集來解決實體主機的故障。以目前主流的 Server 來說，以機架式的 1U 及 2U 為主，配備雙電源供應器分別接至兩條獨立的電力系統迴路，避免單條迴路損壞導致伺服器停擺。網路部分多配備有 4 Port(1Gb) 乙太網路卡，根據需求分別接至不同的 Switch，將不同功能的線路分散在不同網路卡上，並可互相備援。[15] [16] [27] [19] [18]

2.1.2 虛擬化技術

虛擬化，指的是從實際提供服務的實體資源上，將資源做邏輯上的分割再提供給需求者的技術。具體而言，虛擬化是在邏輯分割出的系統環境上執行作業系統或系統服務的能力，並且無關乎特定的實體電腦系統。我們都知道，實際上作業系統或系統服務等都必須在實際的電腦系統上運行，然而虛擬化提供了一個邏輯上的抽象層，抽象層能與應用程式、系統服務，甚至作業系統之特定的硬體區塊綁定。虛擬化，專注於邏輯上的運作環境，但實體上，能讓應用、服務以及在作業系統中的實體，易於在不同的實體系統上搬移。虛擬化後硬體能夠執行許多作業系統下的應用程式，更有效率地管理 IT，並且與其他電腦分配計算資源 [26] [25]。透過虛擬機器的監控器，虛擬化將硬體模仿為更多的硬體，每一個虛擬機可以視為一個完整的個體單位。作為虛擬機，其也有記憶體、CPU 等，視為獨立且完整的硬體設備，它可以執行任何統稱為 Guest OS 系統的作業系統，而不影響其他的虛擬機。在一般情況下，大多數的虛擬化可以分為兩個主要類別：[27] [25] [23]

2.1.2.1 Bare-Metal hypervisor

半虛擬化，虛擬機器監控器提供底層實體硬體修改後的版本。其導出的虛擬機是相同的架構，這使其並不需要模擬。相反的，有針對性的修改，使其更簡單，更快，支援多個 Guest OS。舉例來說，Guest OS 可能會被修改，使用一種特殊的高級呼叫，應用程式二進制界面（ABI），而不是使用某些在常見架構中常用的指令。這意味著，通常只有微小的變化需要在 GuestOS 修改，但任何變化，使得它很難支援原始碼非開放的作業系統，如 Microsoft Windows，皆只發布二進制形式。如同全虛擬化，應用程式運作上仍然無需進行修改。半虛擬化與全虛擬化等使用一個虛擬機器監控器和虛擬機，這個詞指的是虛擬化的作業系統。然而，不像全虛擬化，半虛擬化需要更改的虛擬作業系統。這允許 VM 與虛擬機器監控器的協調和減少使用特權指令，這通常是全虛擬化需要地，但也使其效能損失。半虛擬化虛擬機它的優點是通常比全虛擬化的虛擬機有更好的效能。然而，缺點是需要對修改的半虛擬化虛擬機或作業系統的了解。半

虛擬化的架構如圖 2-1 所示 [28] [29]

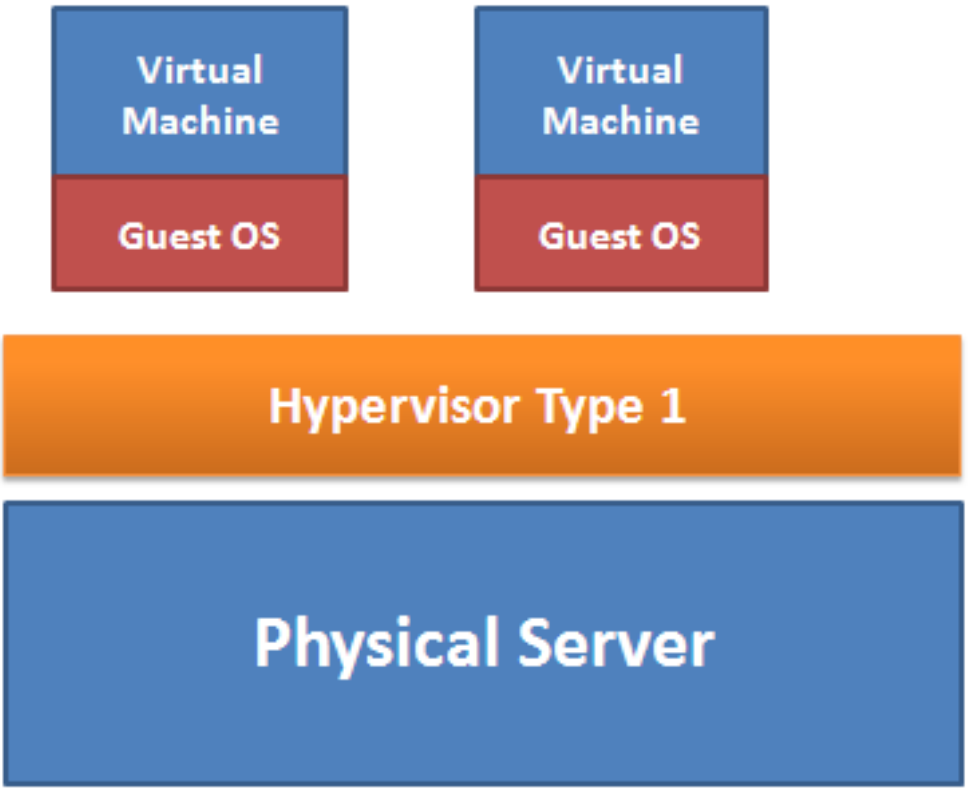

FIGURE 2.1: Bare-Metal VM 系統架構圖

2.1.2.2 Hosted hypervisor

全虛擬化（也稱為本機虛擬化）是模擬相似的硬體；未修改的作業系統和應用程式，可在虛擬機內運行。全虛擬化與模擬不同的是，模擬是設計為在基礎的實體機上運行相同的架構，作業系統和應用程式。這使得一個完整的虛擬化系統直接運行許多指令在原始硬體上。在這種情況下，虛擬機器監控器監控底層硬體的存取，並給每個 Guest OS 有它自己的副本的錯覺。它不再使用軟體來模擬不同的基礎架構如圖 2-2 所示。[28] [29]

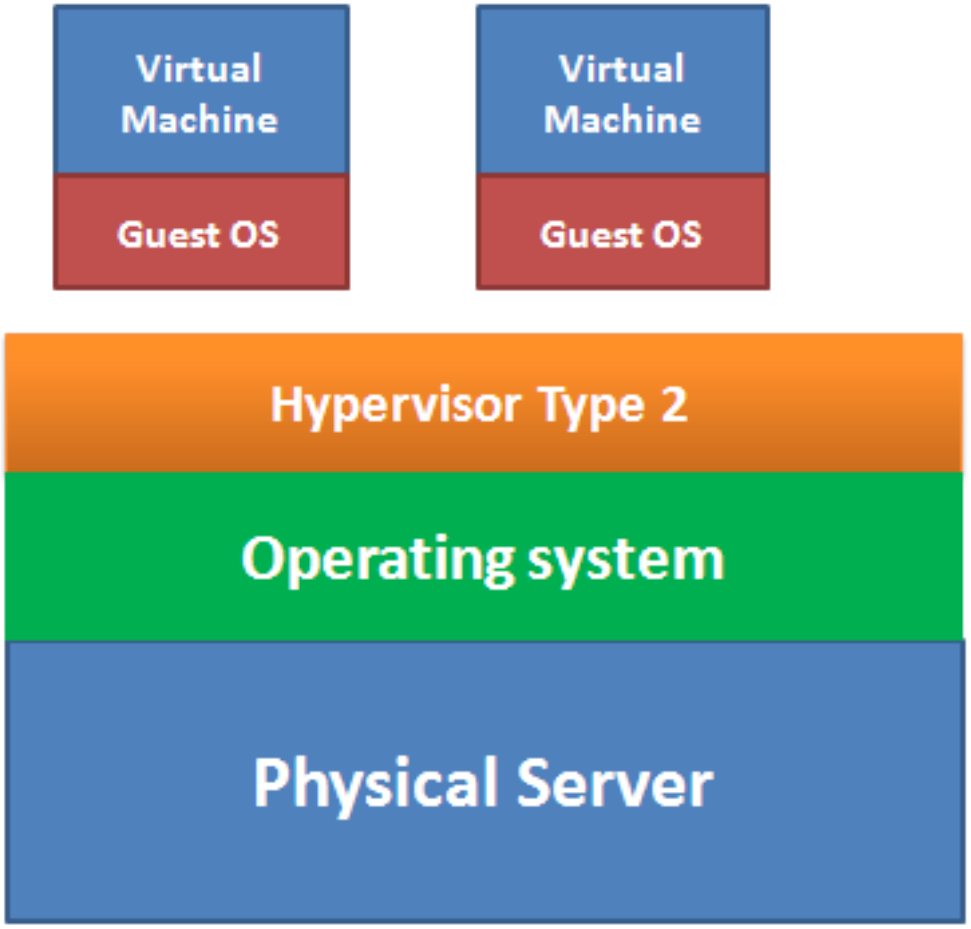

FIGURE 2.2: Hosted VM 系統架構圖

2.1.3 Distributed Replicated Block Device

DRBD (Distributed Replicated Block Device) 是 Linux 平台上的分散式儲存系統。是一種透過網路傳輸，即時同步主機間區塊裝置（Block Device）資料內容的技術。其中包含了核心模組，數個使用者空間管理程式及 shell scripts，通常用於高可用性叢集。DRBD 類似磁碟陣列的 RAID 1（鏡像），只不過 RAID 1 是在同一台電腦內，而 DRBD 是透過網路。[32] DRBD 的特點與優勢是：即時傳輸、透明運作、支援同步及非同步機制，透過核心模組運作，並整合至 System I/O Stack，非常接近硬體底層，所以，具有通用性跟靈活性，適用於任何應用程式。基礎架構如圖 2-3 所示。

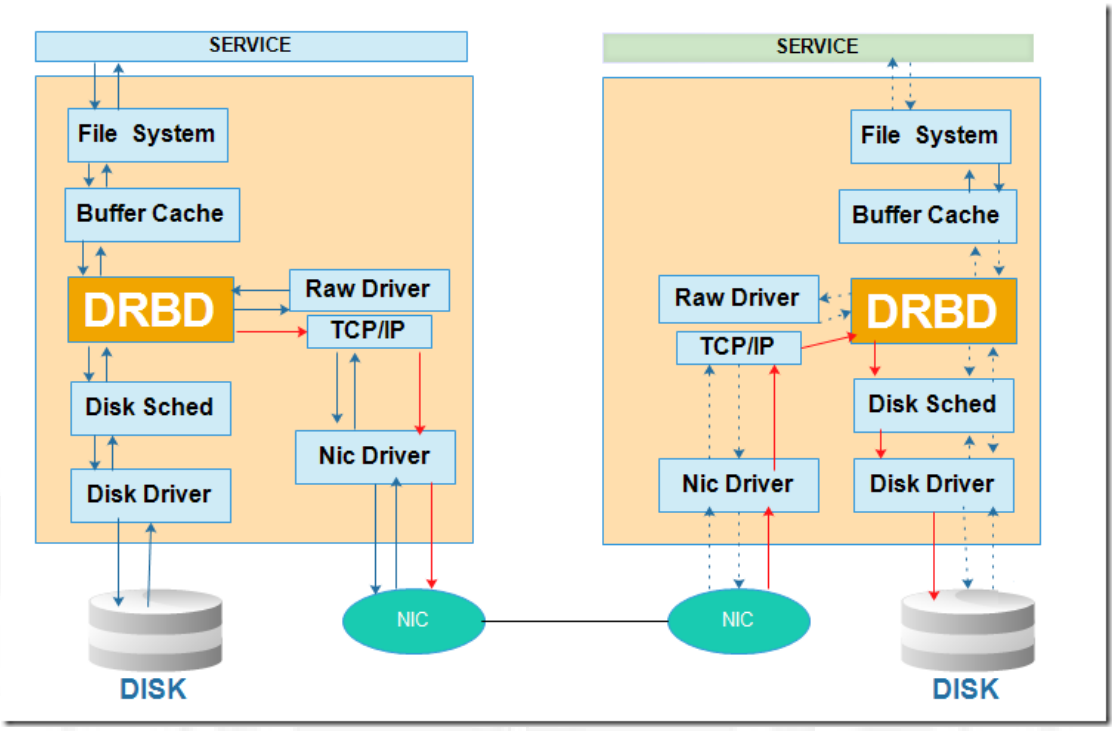

FIGURE 2.3: DRBD 系統架構圖

DRBD 的網路需求官方建議要用 Gigabit 以上等級的網卡，這個以現今的伺服器來說已是標準備配。此外，網路架構要透過 Back-to-Back 對接方式，並避免避免經過路由器。其主要原因是需要高網路吞吐量，以及需要低延遲時間的網路連線。

2.1.3.1 DRBD 工作原理

每個節點 (主機) (DRBD 提供了不止一台節點) 都有一種狀態，可能是主要 (primary) 或次要 (secondary)。在主要節點上，DRBD 程式能執行和走訪 DRBD 磁區內部 (/dev/drbd*)。每次寫入資料都會同時傳送至自身硬碟和透過網路至次要節點的硬碟中。從次要節點只能簡單地把資料寫入自己的硬碟上。讀取資料通常本機執行。如果主要節點發生故障，心跳 (heartbeat) 會將次要節點轉換到主要的狀態，並啟動前面所提的主要節點執行程式。(如果將它和無日誌檔案系統使用，則需要執行 fsck)。如果發生故障的節點恢復工作，那就會轉成為新的次要節點，而且必須使自己的內容與主要節點保持同步。同時，這些動作並不會干擾到其他背景服務。[32] Heartbeat 自身包含了幾個套件，分別

是 ipfail、Stonith 和 Ldirectord，介紹如下。ipfail 的功能直接包含在 Heartbeat 裡面，主要用於檢測網路故障，並做出合理的反應。為了實現這個功能，ipfail 使用 ping 節點或者 ping 節點組來檢測網路連接是否出現故障，從而及時做出轉移措施。Stonith 套件可以在一個沒有回應的節點恢復後，合理接管集群服務資源，防止資料衝突。當一個節點失效後，會從群組中刪除。如果不使用 Stonith 外掛程式，那麼失效的節點可能會導致集群服務在多於一個節點運行，從而造成資料衝突甚至是系統崩潰。因此，使用 Stonith 套件可以保證共用存儲環境中的資料完整性。[30] [31] Ldirector 是一個監控群組服務節點運行狀態的套件。Ldirector 如果監控到集群節點中某個服務出現故障，就遮罩此節點的對外連接功能，同時將後續請求轉移到正常的節點提供服務。此套件經常用在 LVS 負載均衡集群中。[14] [17]

2.1.4 VMware®

2.1.4.1 VMware ESXi

VMware vSphere 是一商業版本的虛擬化平台，作為雲端環境的核心基礎建構區塊。這套平台具有可用性、富彈性的隨選基礎架構。其中重要功能與元件主要有 VMware vSphere Hypervisor(VMware ESXi)、VMware vSphere Virtual Machine File System(VMFS)、VMware vSphere High Availability。

2.1.4.2 VMware vCenter

VMware vCenter Server 提供可管理 VMware vSphere 環境的集中式平台，透過 VMware vSphere Web Client 可以集中管理所有的 VM 及實體伺服器資源，並將 ESXi 主機組成資源叢集，動態分配資源及透過 HA 機制降低硬體故障造成的損失。基礎架構如圖 2-4 所示

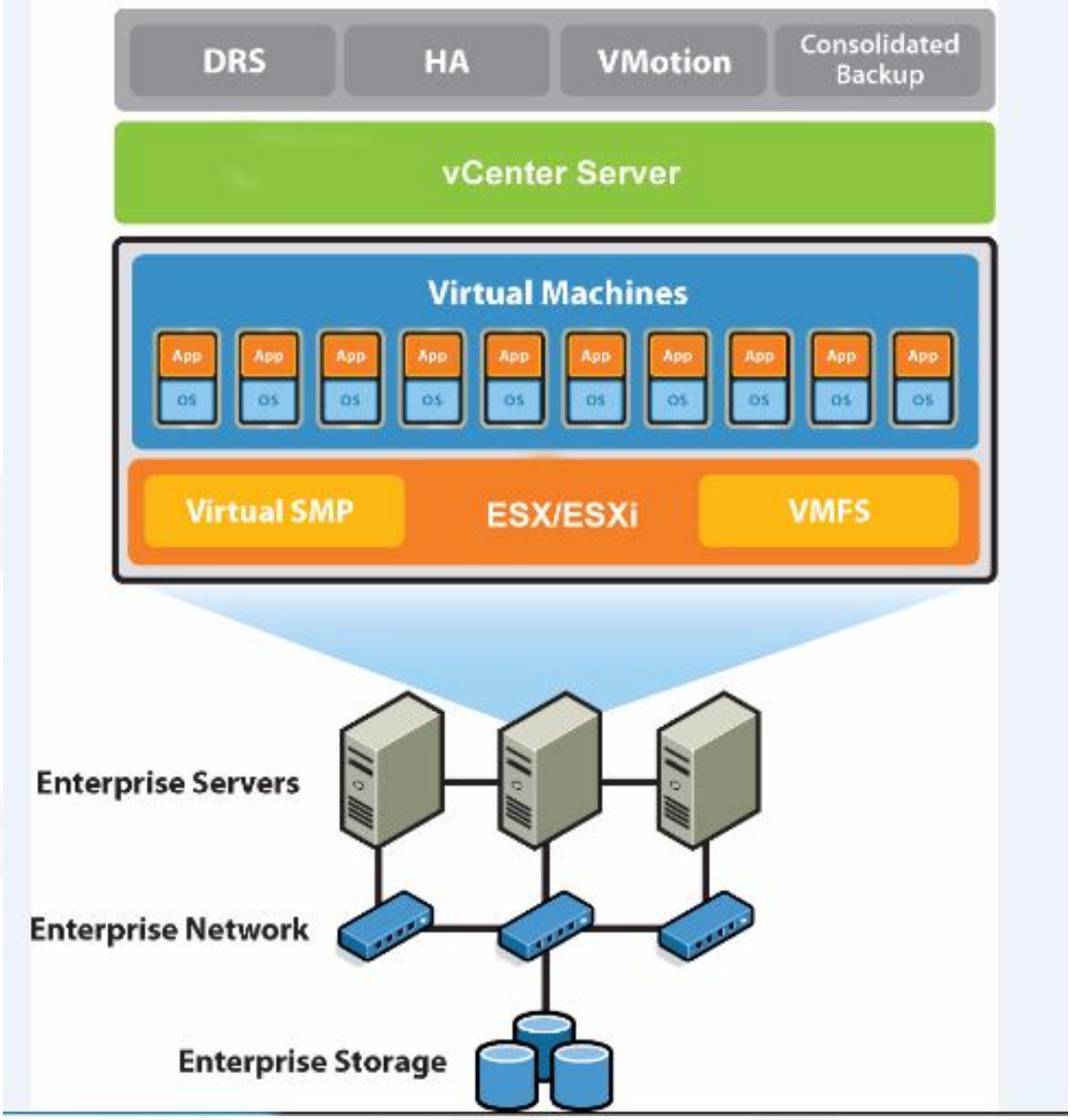

FIGURE 2.4: VMware vSphere 系統架構圖

2.1.4.3 VMware Integrated OpenStack

VMware Integrated OpenStack 使用 vSphere Web Client 進行其部署。vSphere Web Client，同時會部署所有的必要虛擬機與元件，可建立並直接使用 OpenStack 基礎架構。由於這個架構完整地支援 Nova、Neutron、Cinder、Glance、Horizon、Keystone 與 Heat。可以簡化整個的安裝流程。圖 2-5 為 VMware Integrated OpenStack 系統架構圖。

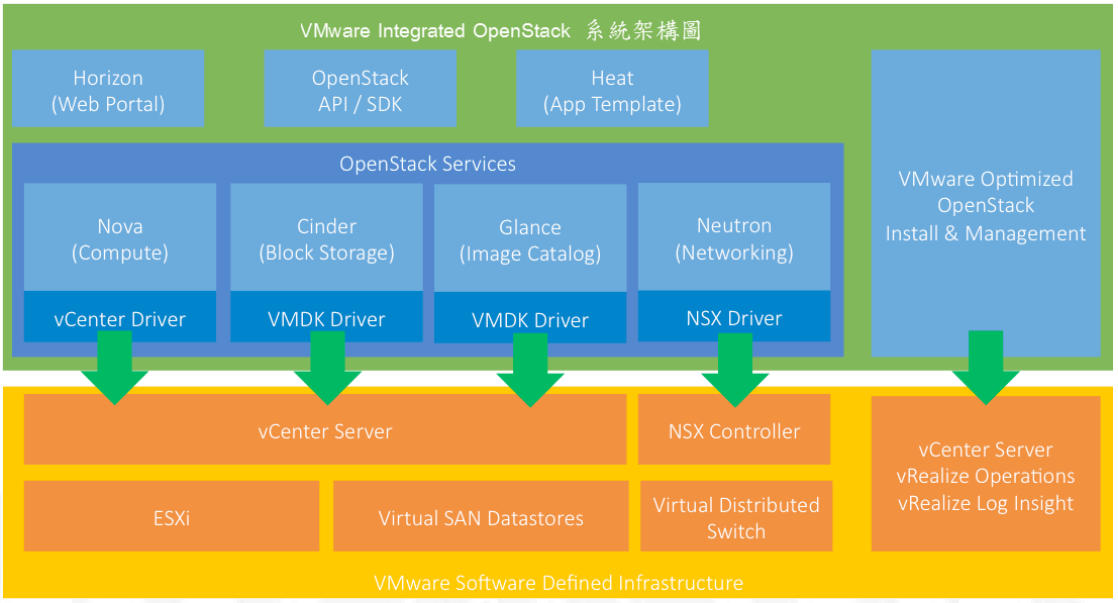

FIGURE 2.5: VMware Integrated OpenStack 系統架構圖

2.1.5 OpenStack

OpenStack 是由好幾個不同的功能套件所構成的開源雲端軟體，分別是運算 (compute) 套件 Nova、儲存 (object-storage) 套件 Swift、區塊儲存 (Block Storage) 套件 Cinder、網路拓撲 (networking) 套件 Neutron、身分識別（identity）套件 Keystone、映象檔（image）管理套件 Glance、提供管理介面的儀表板 (dashboard) 套件 Horizon。套件數目及名稱隨著版本不同有所增減和改變。基礎架構如圖 2-6 所示。[7] [10]

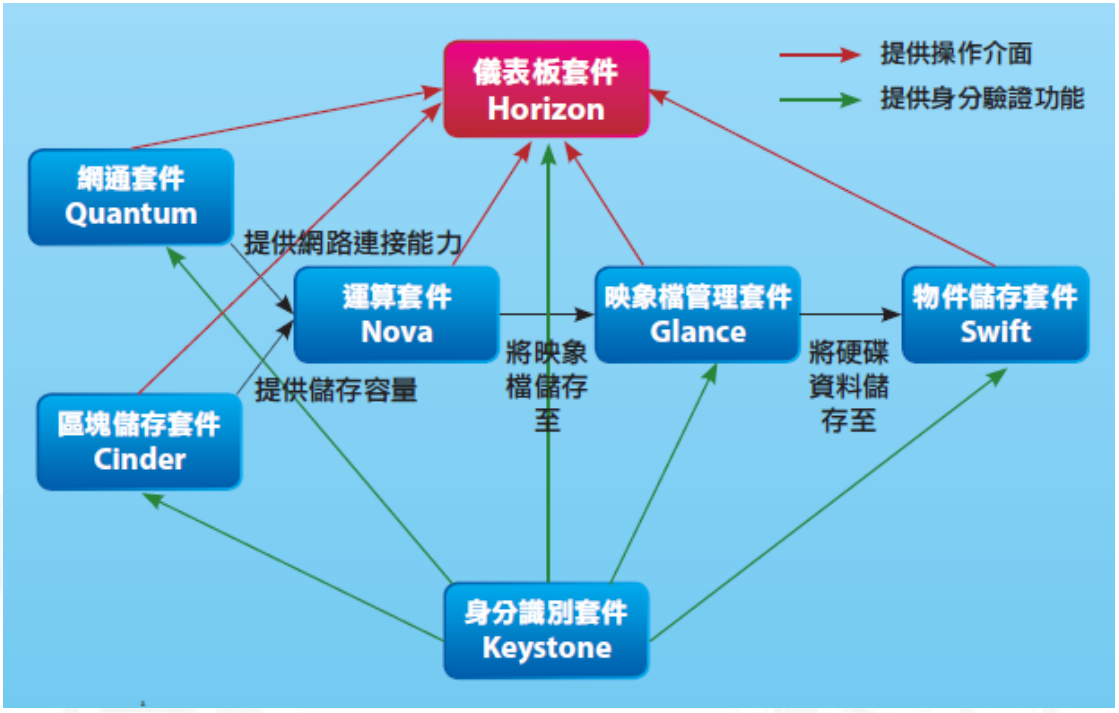

FIGURE 2.6: OpenStack 系統架構圖

2.2 相關研究

本論文參考了 [7] Arvinder Kaur 提出了一種測試高可用性的方法，分別測試 30 次的模擬硬體失效，並評估平均的回復時間、重啟時間。[11] 為了增加測試的真實性，採用了 TPC-C 的標準來模擬產生 MySQL 的流量。TPC(Tracsaction Processing Performance Council) 事務處理性能協會是一個評價大型數據庫系統軟硬件性能的非盈利的組織,TPC-C 則是 TPC 協會制定的，用來測試典型的複雜系統的性能；Tpcc-mysql 是 percona 基於 tpcc 衍生出來的產品，專用於 mysql 基準測試。測試原理是建立一個完整的訂單系統，測試更接近實際運作的系統。圖 2-7 為整個資料表的內容範例 [6] [5] [4] [3]

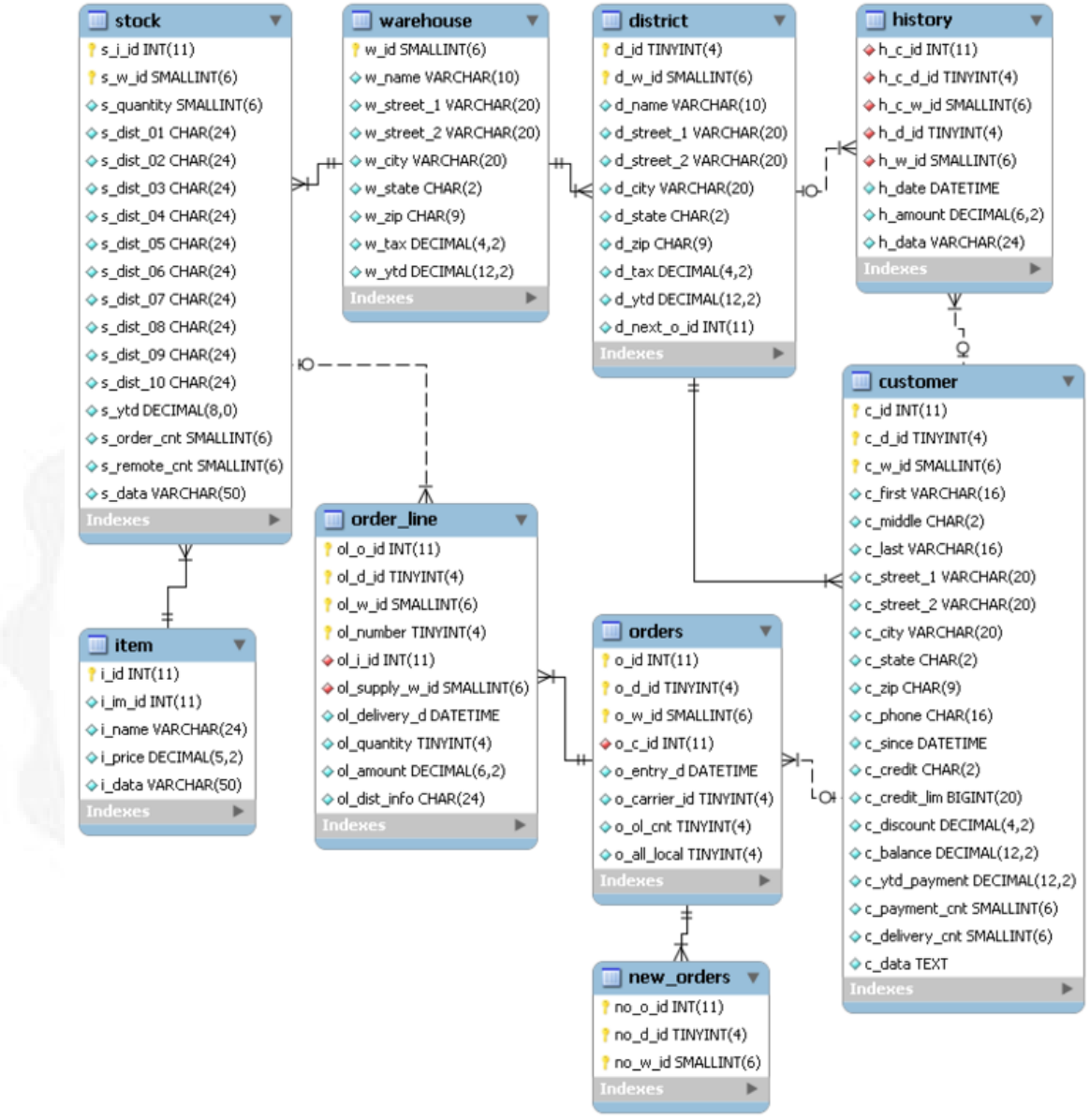

FIGURE 2.7: tpcc-mysql 所建立的訂單系統資料表

網站服務方面則使用 ApacheBench 來模擬網站訪問。ApacheBench 工具程式是 Apache 網站伺服器軟體的一個附帶的工具軟體，專門用來執行網站伺服器的運行效能，特別是針對 Apache 網站伺服器的效能分析。這支程式原本是用來檢測 Apache 網站伺服器 (Web Server) 所能夠提供的效能，特別是可以看出 Apache 網站伺服器能提供每秒能送出多少網頁。在本論文中，我們用來模擬正常運作的網站會產生的連線狀況。[2] [1] [2] [11]

Chapter 3

系統設計與實作

3.1 建置環境

建置的環境我們是以三台 IBM System X3650 M4 來組成整個 VMware vSphere ESXi 實體主機的環境。在圖 3-1 詳列了硬體的規格。

硬體設備型號：IBM System x3650 M4		
CPU	Intel Xeon E5-2620 6Core 2.0 GHz	2顆
RAM	8GB DDR3-1333 ECC Register	16片
HDD	600GB 10K 6Gbps SAS 2.5”	8顆
RAID	IBM ServeRAID M5110e RAID	1張
LAN	Intel 1GbE	4個
Power Supply	750W DC	2顆

FIGURE 3.1: 硬體規格

3.1.1 VMware Integreated OpenStack

使用 VMware Integreated OpenStack（VIO）將建立好的 OpenSatck 部署於 VMware vSphere 中，建立統一的實驗環境，圖 3-2 為 OpenStack 於 VIO 部署

的狀況。這樣可以建立統一的測試環境，最後用以比較 VMware 與 OpenStack 之間的差異。

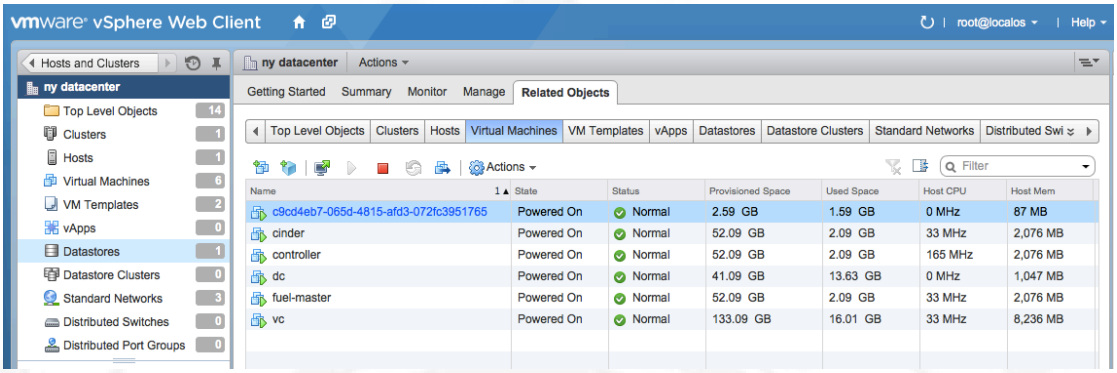

FIGURE 3.2: VMware Integreated OpenStack 部署狀況

3.1.2 OpensStack HA 研究

OpensStack HA 研究由於高可用性系統力求最大限度地減少了兩件事：系統維修的停機時間，資料意外被刪除或銷毀。大部分的高可用性系統中只有一個且單一故障時保證防止系統停機時間和數據丟失的功能，當然也防止一連串錯誤所惡化的連鎖效應。高可用性的一個關鍵方面是消除單點故障（single points of failure,SPOF ）。SPOF 是指一件單獨的設備或軟體如果失效，這將導致系統停機或數據消失。為了消除 SPOFS 的檢查機制，冗餘 (redundancy) 存在：網絡元件，例如交換機和路由器，應用程式和自動服務遷移，儲存套件，硬體設施服務，如電力，空調和消防。大部分的高可用性系統將在多個獨立（非必要）故障時失敗。在這種情況下，大多數系統將保護數據在保持可用性。目前 OpenStack 的可用性可以滿足的基礎功能服務，這意味著 OpenStack 能 99.99% 的正常運行時間執行基礎服務。不過，OpenStack 並不能保證每位使用者的虛擬機都能擁有 99.99 %的可用性。[9]

3.1.3 VMware HA 研究

vSphere High Availability 能針對虛擬機所執行的眾多應用程式提供所需的可用性，不受執行的作業系統及應用程式影響。HA 能提供統一、具成本效益

的容錯移轉保護，以保護虛擬化的 IT 環境免於硬體及作業系統中斷。HA 能達成：

(1) 監控 vSphere 主機和虛擬機，藉此偵測硬體和 Guest 作業系統中斷。

(2) 偵測到伺服器中斷時，能將叢集中其他 vSphere 主機內的虛擬機重新啟動，無需手動介入。

(3) 在偵測到作業系統中斷時，會自動重新啟動虛擬機，縮短應用程式停機時間。

3.2 系統架構與實作

首先，分別在三台主機上安裝 ESXi，之後再於虛擬環境中立一台虛擬機器，並安裝 Windows Server 2012。最後，將 VMware vCenter 安裝於此 Windows 環境中。透過 vCenter，我們建立起一個容錯轉移叢集，並將兩台實體主機納入管理。此一基本的環境即為 VMware 高可用性雲端服務之平台。

另外，使用 vSphere Web Client 進行 OpenStack 的部署，並透過 Pacemaker 建立 Cluster。此即為 OpenStack 高可用性雲端服務平台。

在上述兩平台中可以實現當其中一台實體主機發生硬體故障時，所有的虛擬機器可以轉移至另一台主機而不致中斷。[24] [22]

接下來，依照架構不同分別實作：

(1) 單節點的架構。

(2) DRBD+Heartbeat 架構。

(3) DRBD+HeartBeat 與 OpenStack 合併架構。

於建置的環境上安裝 Mysql 及 Apache 以及以便模擬資訊服務，並一一進行測試。

3.2.1 單節點架構

該架構主要的問題點就在於該節點上的所有資料都儲存於同一個硬體裝置上，當硬體裝置損毀時會造成資料的遺失。圖 3-3 為此架構的示意圖。

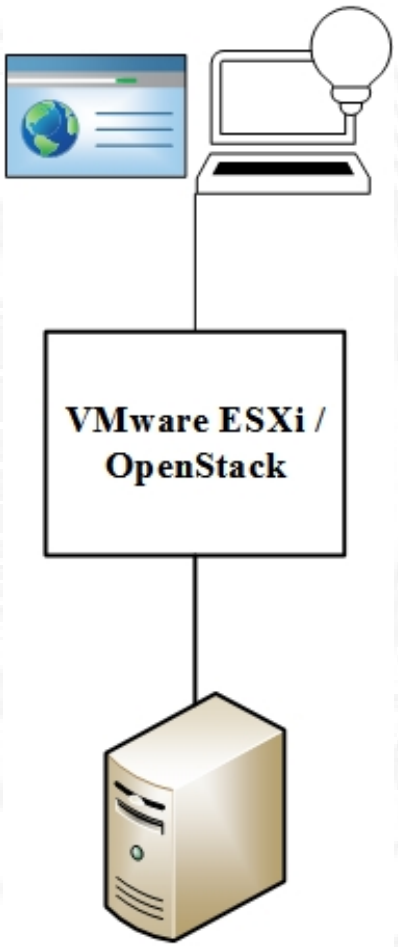

FIGURE 3.3: 單節點架構圖

3.2.2 DRBD+Heartbeat 架構

針對該節點上資料會因為硬體損毀的問題，因此需要對於單個節點做出一個備份節點，因此在這裡我們使用 DRBD 作為備份資料以確保資料的完整性。接著為了使 DRBD 有即時移轉 (live migration) 的能力，使得 OpenStack 的服務不會因為當下主伺服器故障而中斷，因此在 DRBD 架設完後，加入 Heartbeat 使用心跳線頻率性的去偵測硬體狀況，以達到在硬體故障的情況下能夠即時的主從轉換，使得服務不會因此而中斷。圖 3-4 為此架構的示意圖。[12] [14] [17]

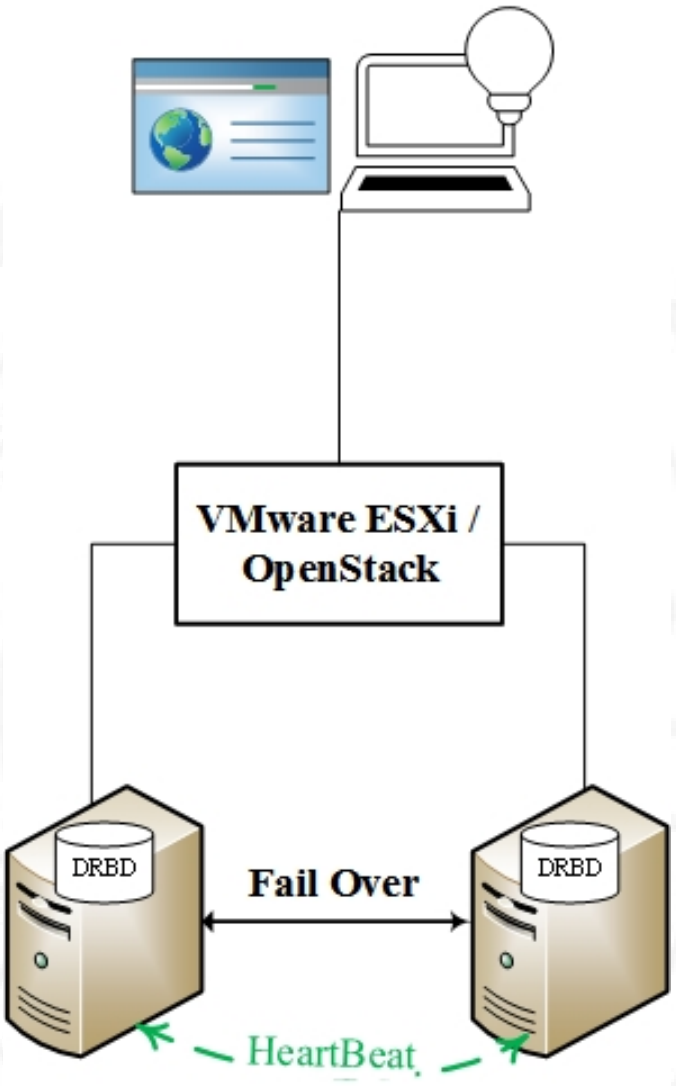

FIGURE 3.4: DRBD+Heartbeat 架構圖

3.2.3 DRBD+HeartBeat 與 Cluster HA 合併架構

將架設好的 DRBD+Heartbeat 節點作為 OpenSatck 的單一節點使用，藉此來達到 OpenStack 的服務就算硬體故障，也能透過即時移轉來達到服務以及資料的高可用性。圖 3-5 為此架構的示意圖，圖 3-6 則說明了當硬體發生故障時如何達到即時移轉。

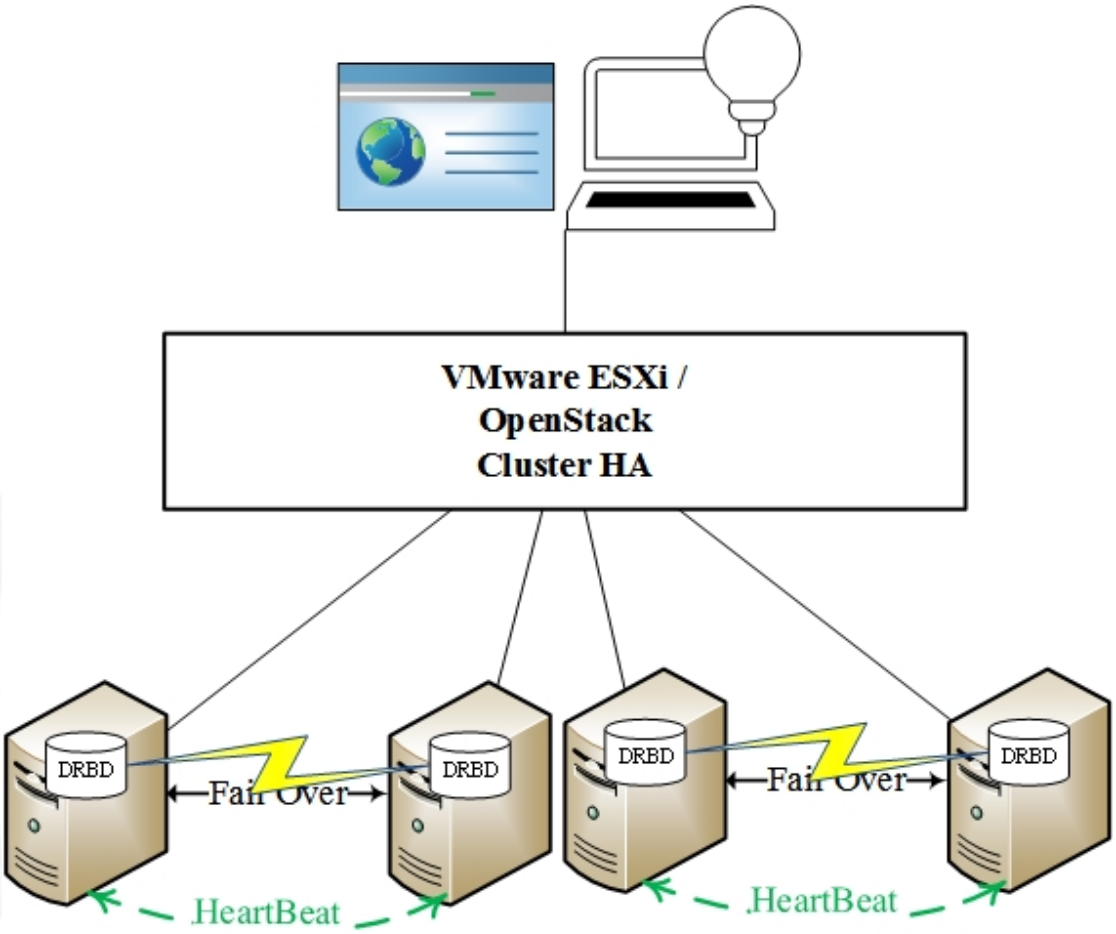

FIGURE 3.5: DRBD+HeartBeat 與 Cluster HA 合併架構圖

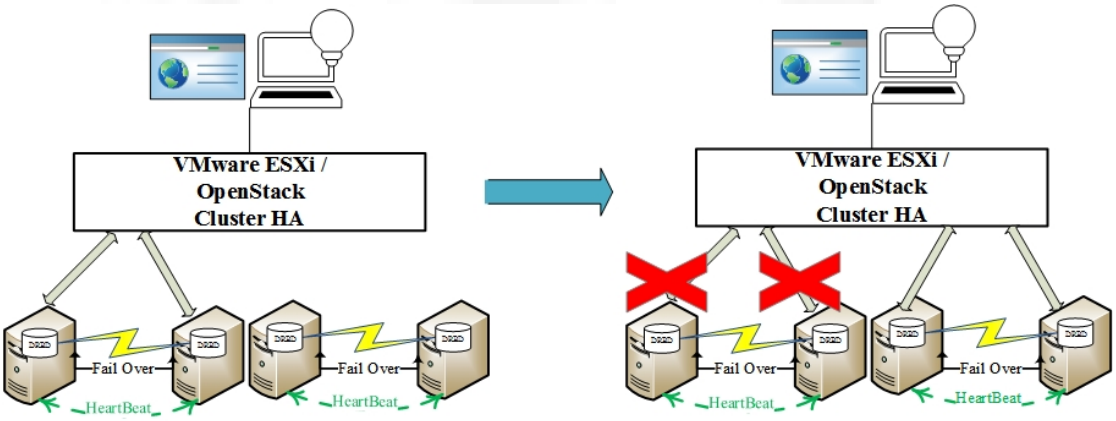

FIGURE 3.6: 即時轉移示意圖

Chapter 4

實驗環境與結果

4.1 實驗環境

實作 OpenStack 及 VMware HA 平台，觀察其運作原理，了解此系統如何達到高穩定和可靠性。是否有能提升此系統穩定性和可靠性的方法，加強性能。將提升方法實際套入其中，並且使之運行觀察其結果。

OpenStack 在虛擬機的管理上，只能對於所有已列入管理清單的虛擬機操作和支配，但無法了解虛擬機和實體機器上的主從關係，若有一台實體機器發生故障，其上方的無法提供服務的虛擬機只能選擇其他虛擬機啟動來彌補發生錯誤實體機上的虛擬機。若能在實體機發生問題時，可以將上方的虛擬機資料備分至其他實體機上，讓備用機器能夠即時恢復已經停止服務的虛擬機。因此若使用 DRBD（Distributed Replicated Block Device）能夠即時複製（Replicated）區塊（Block Device）裝到網路上另一台電腦的 Block Device，簡單的說就是「網路版的 RAID1」，以達到高可用性的目的。而在 VMware vSphere 的 Cluster 環境中，若無 share storage 的情況下，無法即時做到線上即時遷移虛擬機，而購置 SAN 或是 NAS 都會是一筆龐大的開銷。因此，同樣的嘗試透過 DRBD 的方式來建構不中斷的雲端服務。

4.2 實驗架構

使用 VMware vSphere 6.0 建立一虛擬基礎平台，並在此平台上分別建立 OpenStack Cluster 及 VMware Cluster 環境，再於這兩個實驗環境上分別實作下述三種架構。

(1) 單節點的架構。
(2) DRBD+Heartbeat 架構。
(3) DRBD+HeartBeat 與 Cluster HA 合併架構。

4.2.1 單節點架構

該架構主要的問題點就在於 VMware vSphere 該節點上的所有資料都儲存於同一個硬體裝置上，當硬體裝置損毀時會造成資料的遺失。圖 4-1 為此架構的示意圖。

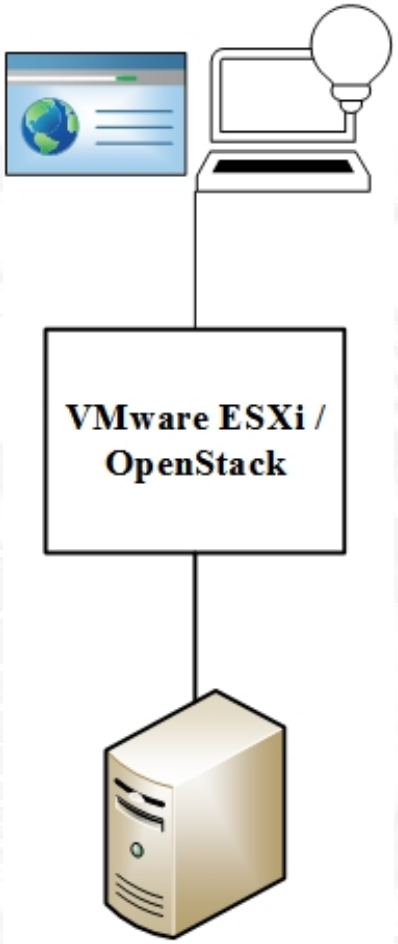

FIGURE 4.1: 單節點架構圖

4.2.2 DRBD+Heartbeat 架構

針對該節點上資料會因為硬體損毀的問題，因此需要對於單個節點做出一個備份節點，因此在這裡我們使用 DRBD 作為備份資料以確保資料的完整性。接著為了使 DRBD 有即時移轉 (live migration) 的能力，使得 VMware vSphere 的服務不會因為當下主伺服器故障而中斷，因此在 DRBD 架設完後，加入 Heartbeat 使用心跳線頻率性的去偵測硬體狀況，以達到在硬體故障的情況下能夠即時的主從轉換，使得服務不會因此而中斷。圖 4-2 為此架構的示意圖。

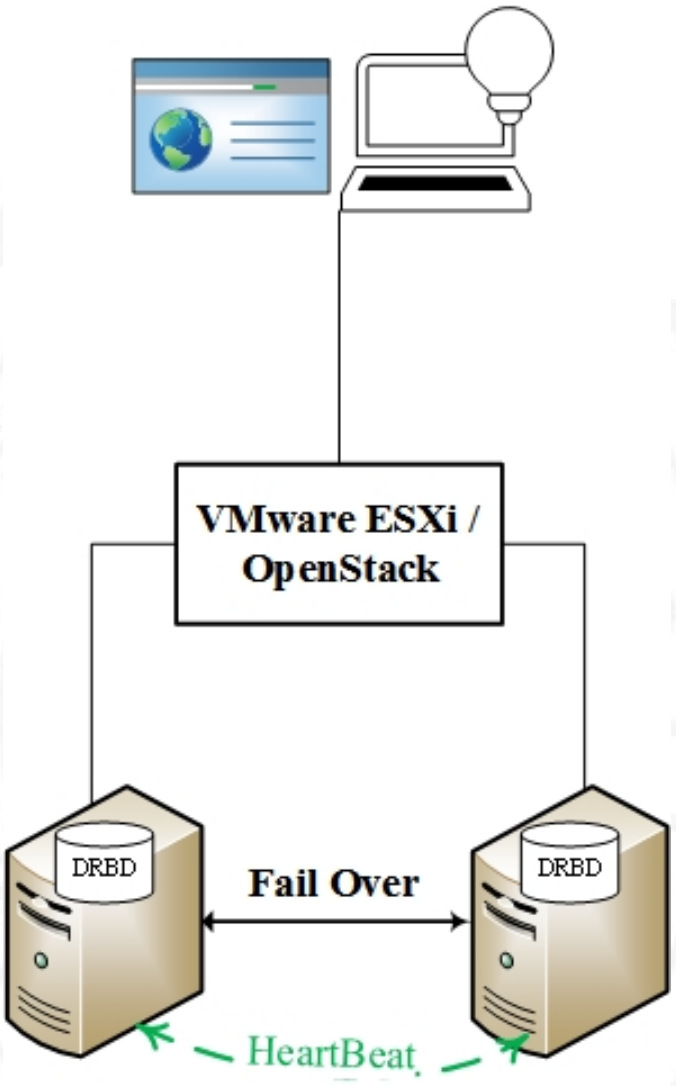

FIGURE 4.2: DRBD+Heartbeat 架構圖

4.2.3 DRBD+HeartBeat 與 Cluster HA 合併架構

將架設好的 DRBD+Heartbeat 節點作為 VMware vSphere 的單一節點使用，藉此來達到 VMware vSphere 的服務就算硬體故障，也能透過即時移轉來達到服務以及資料的高可用性。圖 4-3 為此架構的示意圖，圖 4-4 則說明了當硬體發生故障時如何達到即時移轉。

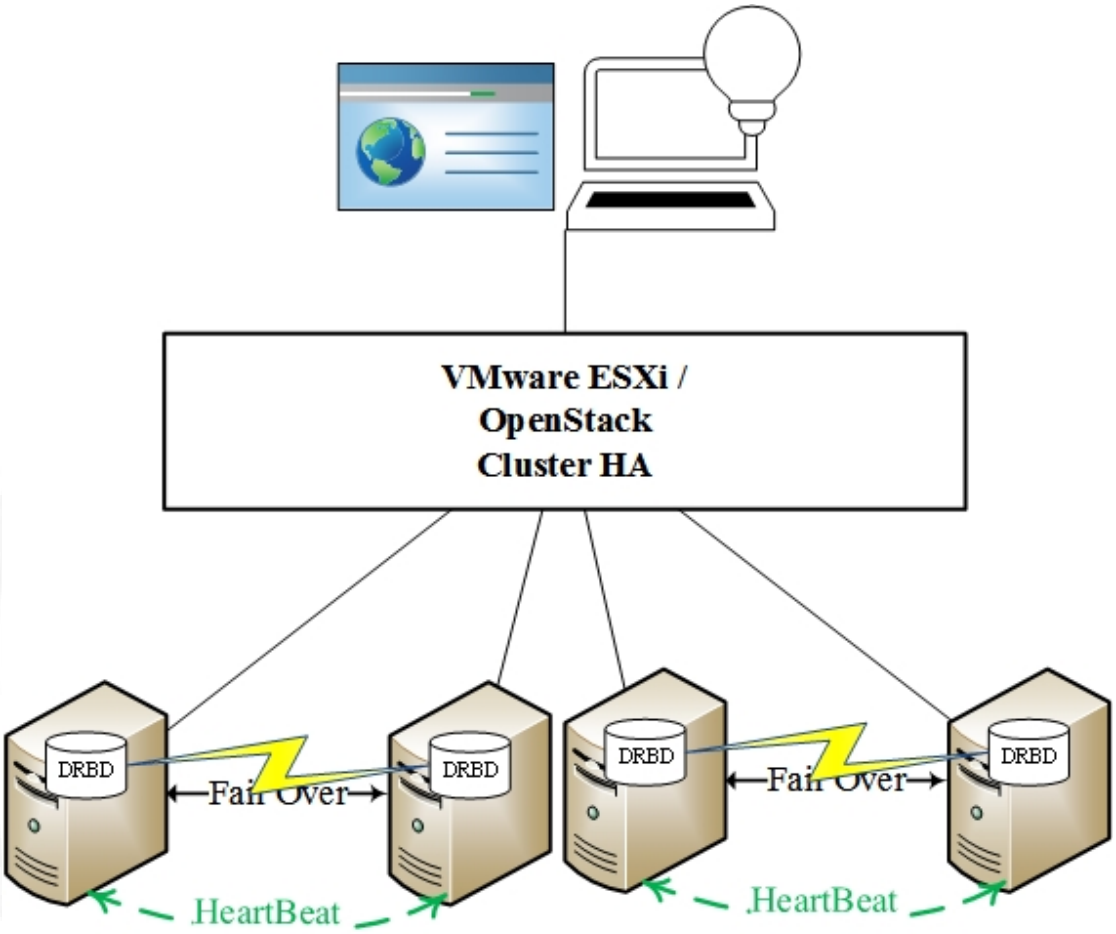

FIGURE 4.3: DRBD+HeartBeat 與 Cluster HA 合併架構圖

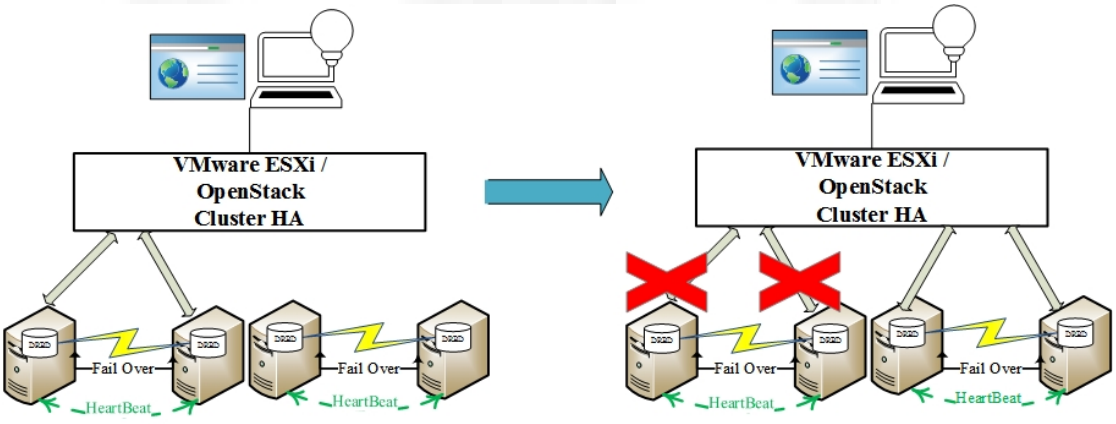

FIGURE 4.4: 即時轉移示意圖

另外，另外使用了一台筆電，對整個 HA 環境進行測試。透過 tpcc-mysql 來產生大量的 mysql 流量，以模擬實際的服務狀況。而 Apache 部分則使用

ApacheBench 來進行網站訪問的模擬。而在系統上則安裝了 Monit 來監測服務的運行狀態，進而計算出失效的時間以及服務重啟的時間。圖 4-5 為整個實驗的架構圖。

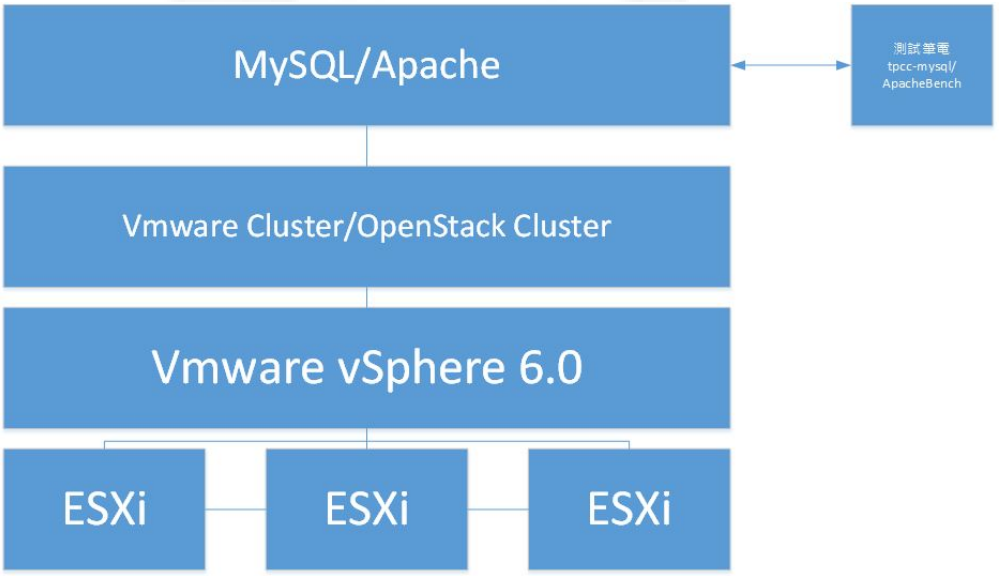

FIGURE 4.5: 實驗架構圖

4.3 實驗方法

4.3.1 確認 DRBD 運作狀態

在測試 HA 機制時您可使用下列指令來即時 (每 1 秒更新) 觀察 DRBD 的 Primary/Secondary 狀態。

```
watch -n 1 service drbd status
```

確認 DRBD 同步完成兩邊同時擁有相同且最新資料。如圖 4-6 及所示。

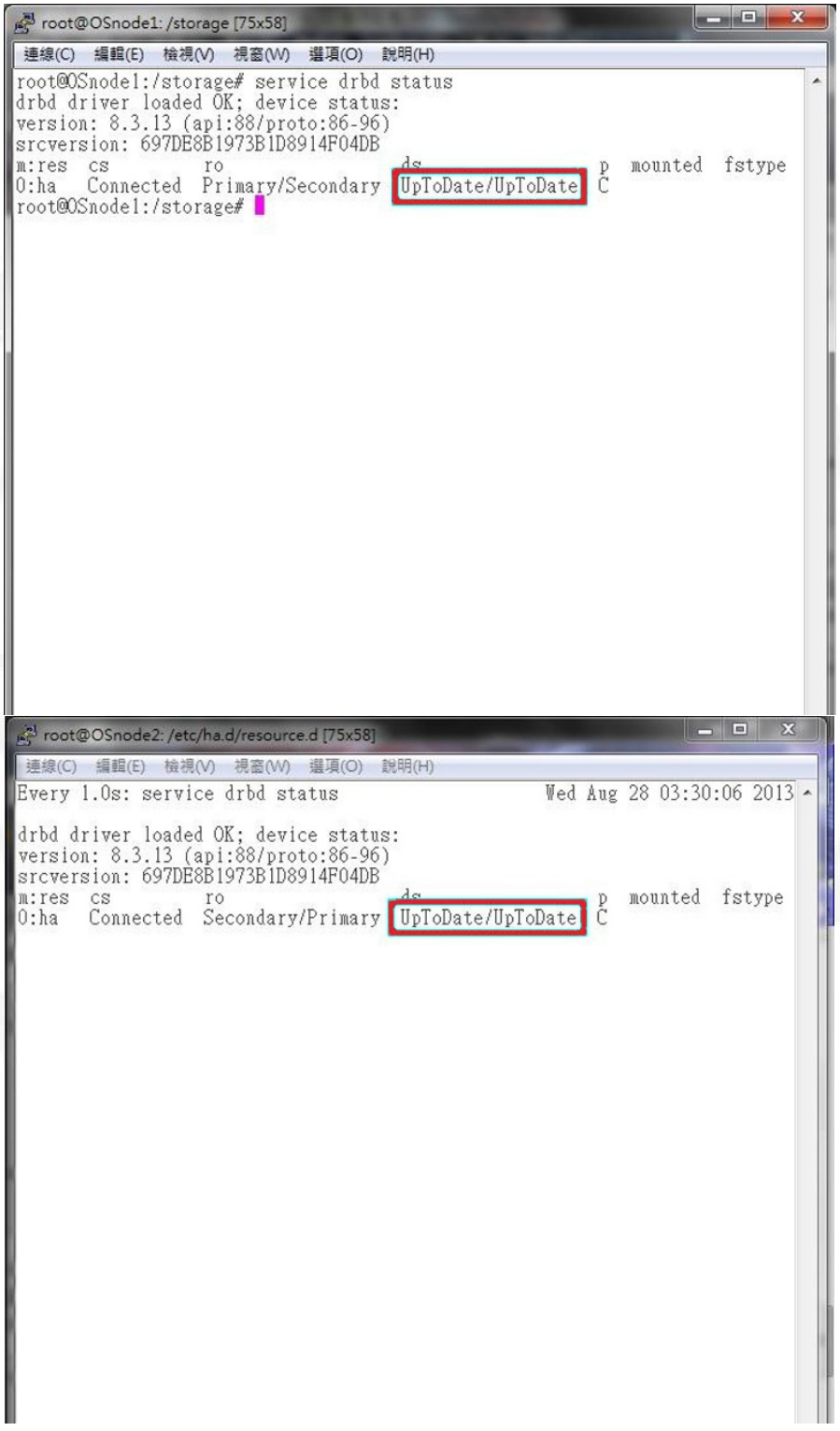

FIGURE 4.6: Node1 與 Node2 的 DRBD 狀態

測試同步: 在 NODE1 用 dd 指令創造一個檔案，自動同步會在 NODE2 出現相同的檔案，如圖 4-7 所示。

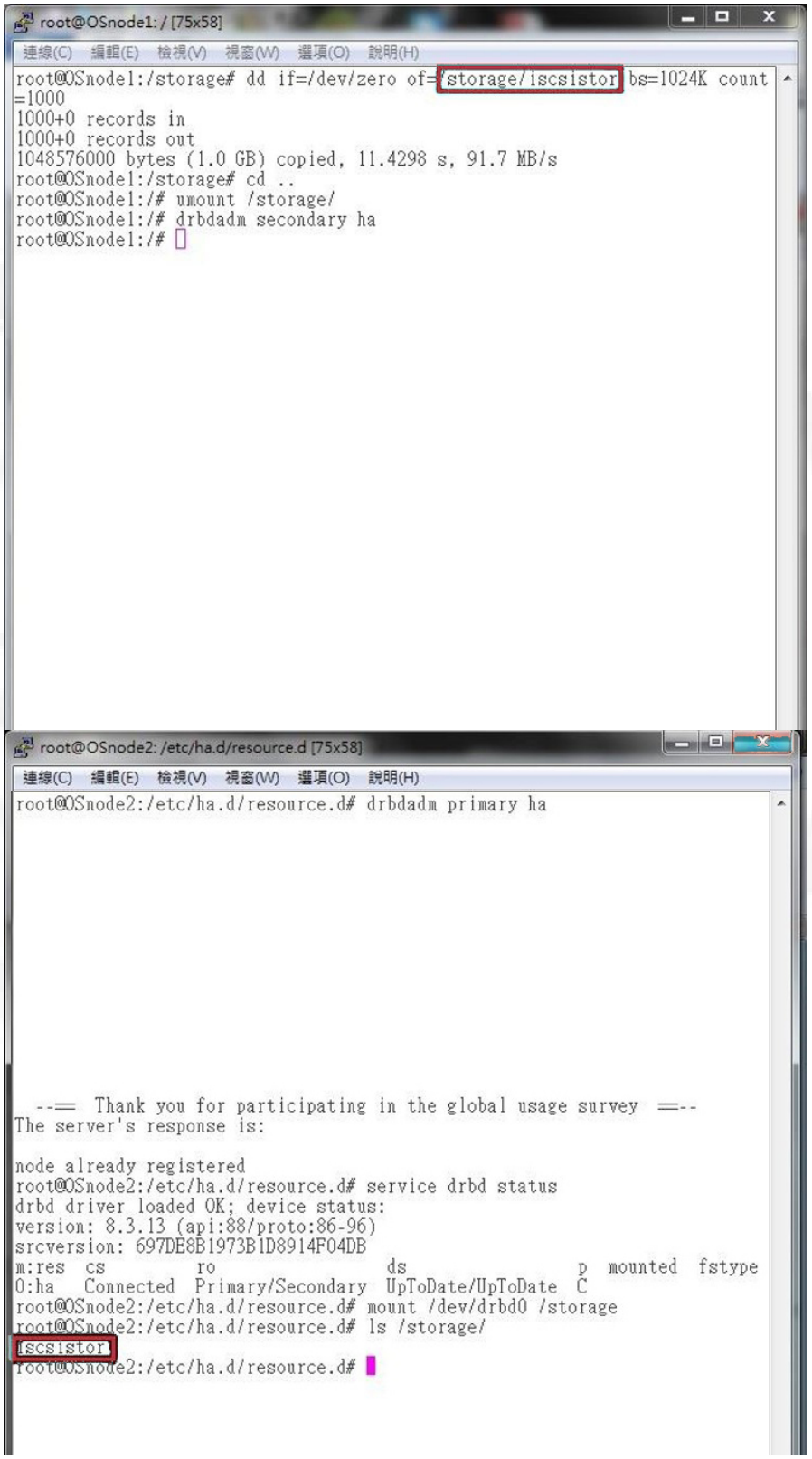

FIGURE 4.7: 在 Node1 建立檔案，Node2 會出現相同檔案

4.3.2 確認 Hearbeat 運作狀態

準備測試 HA 交換機制先 NODE1 下好暫停指令未送出，NODE2 則監控同步狀態（圖 4-8）。

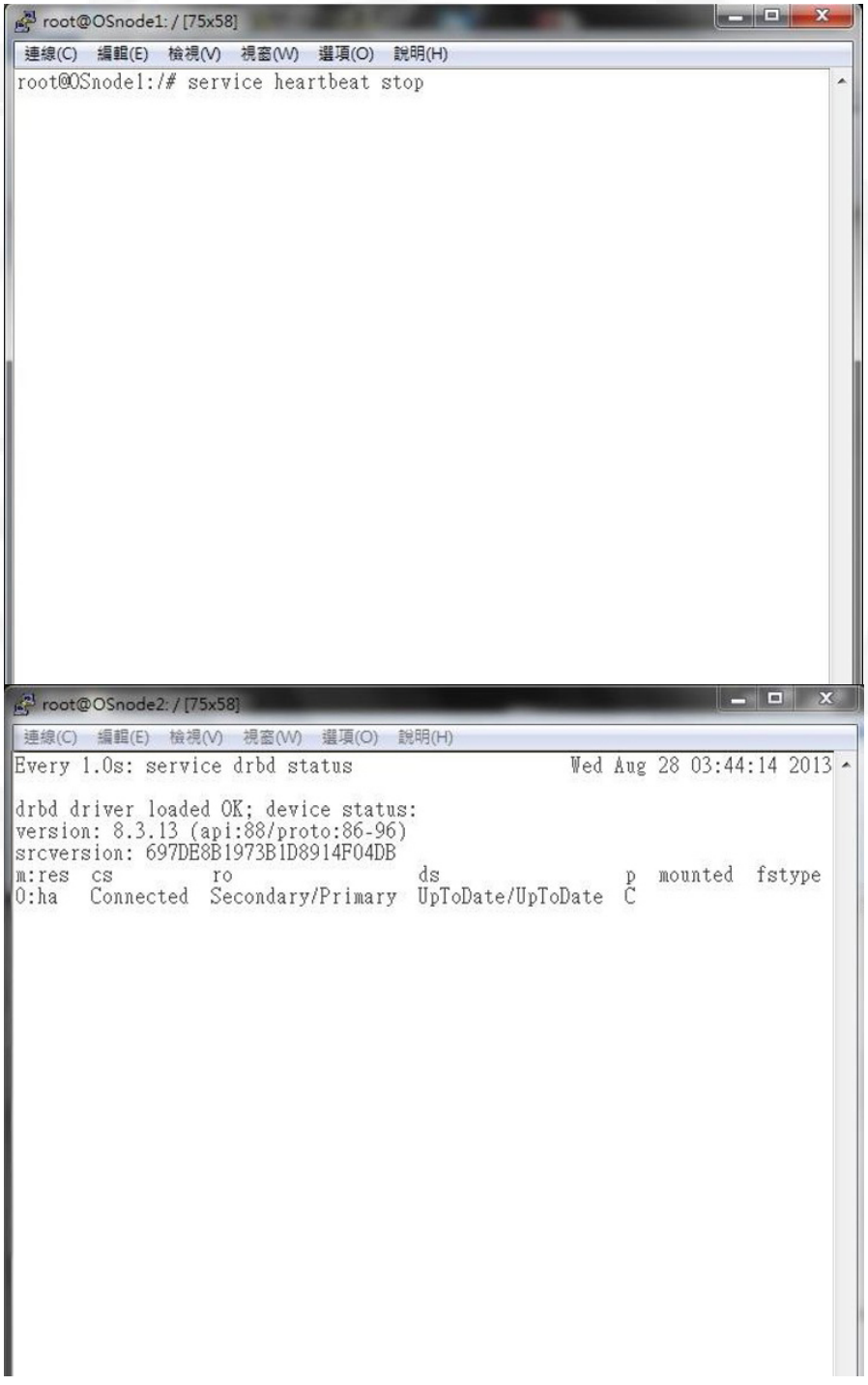

FIGURE 4.8: 在 Node1 打停止服務的指令暫不送出，監控 NODE2 的同步狀態

NODE1 服務暫停指令送出，HeartBeat 自動啟用 HA 機制 NODE2 由 Secondary 狀態切換為 Primary 狀態接管（圖 4-9）。

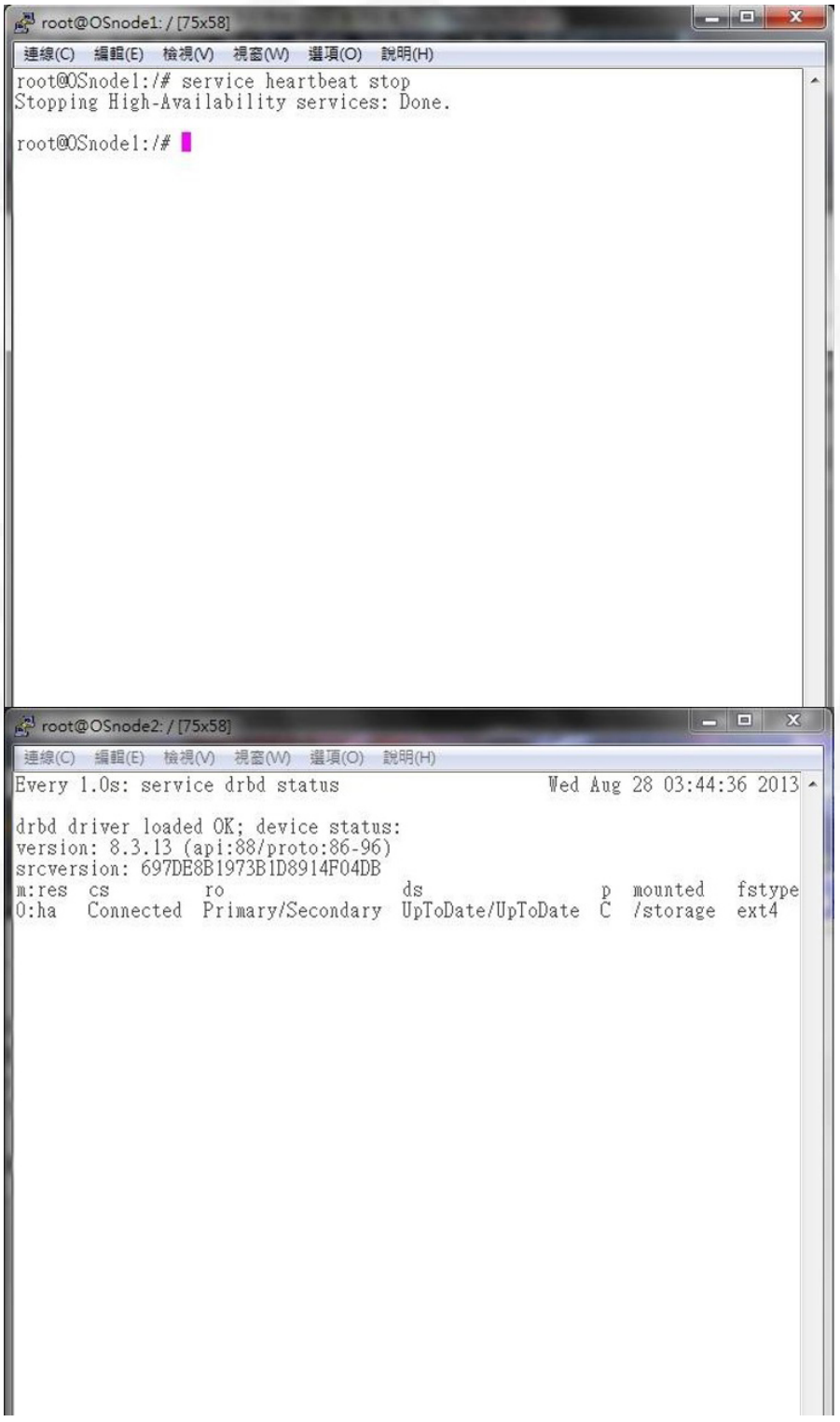

FIGURE 4.9: 在 Node1 送出停止服務的指令，檢查 Node2 切換為 Primary 狀態

4.3.3 建立 VMware Cluster

透過 VMware vSphere Client 連入實驗平台，並建立 Cluster，開啟 HA。圖 4-10 中可以看到整個 VMware Cluster 的狀態。

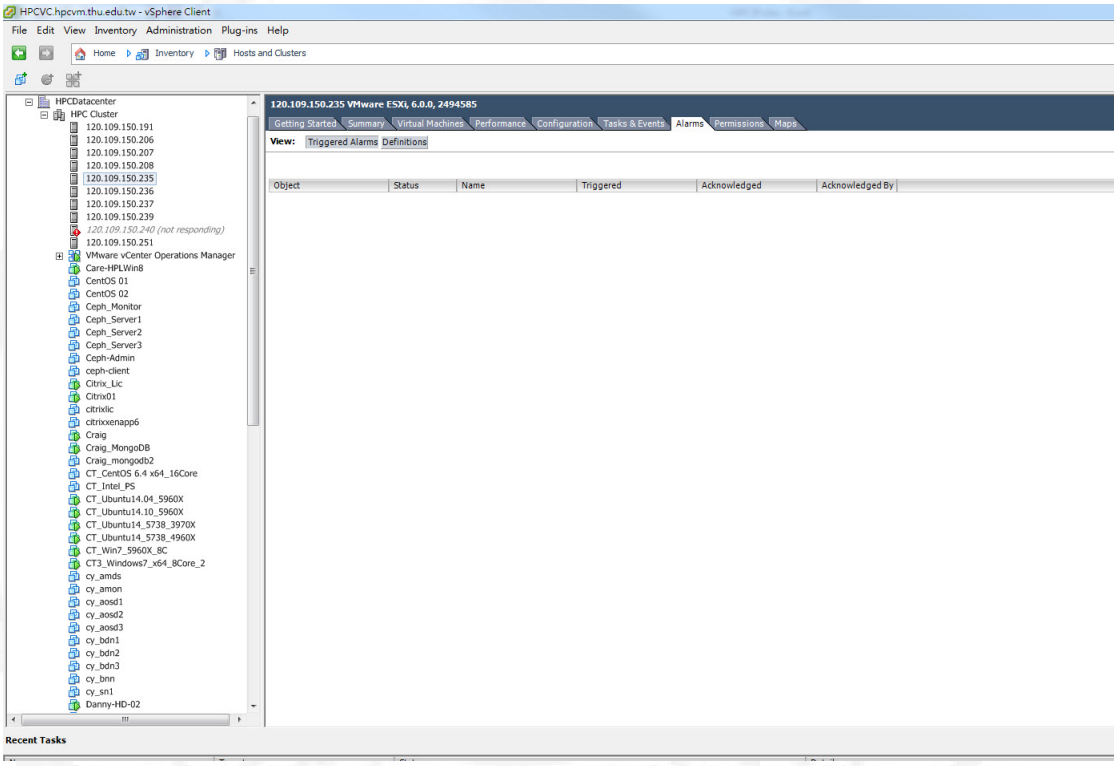

FIGURE 4.10: VMware Cluster 狀態

4.3.4 建立 OpenSatck Cluster

OpenStack 的基礎設施高可用性依賴了 Pacemaker 叢集，透過安裝 Corosync 來控制叢集內各實體主機的通信，以達到 Active/Passive 模式的高可用性叢集。圖 4-11 為整個 Pacemaker 的架構。

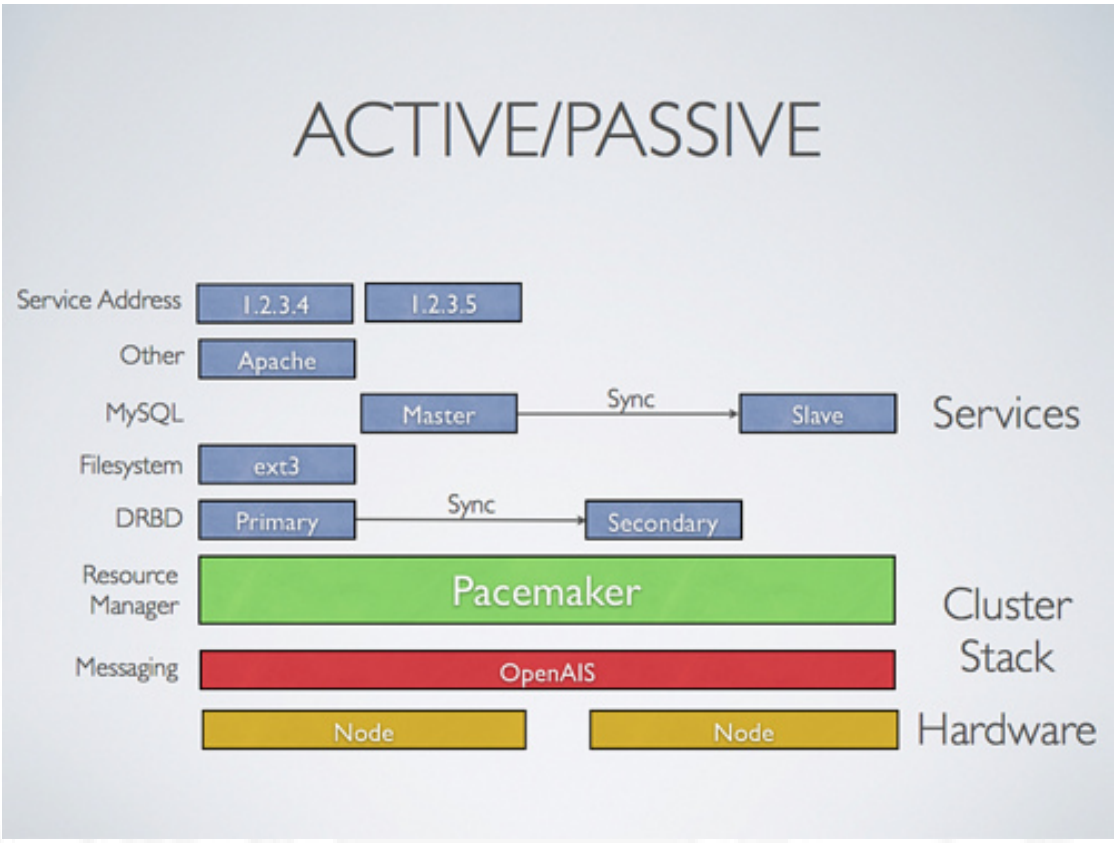

FIGURE 4.11: Pacemaker 架構圖

建構完成後，可以用下列指令來確認 corosync 的連通性，圖 4-12 為實驗環境的檢測狀況

corosync-cfgtool -s

```
# corosync-cfgtool -s
Printing ring status.
Local node ID 435324542
RING ID 0
        id      = 10.0.42.100
        status  = ring 0 active with no faults
RING ID 1
        id      = 10.0.42.101
        status  = ring 1 active with no faults
```

FIGURE 4.12: corosync 連通性狀況

一旦 Pacemaker 啟動完成，可以用 crm_mon 來查詢運行的狀態，圖 4-13 為 Pacemaker 運行的狀態

```
============
Last updated: Sun Jul 5 23:00:52 2015
Last change: Sun Jul 5 20:46:00 2015 via cibadmin on node2
Stack: openst
Current DC: node2 - partition with quorum
Version: 1.1.6-9971ebba4494012a93c03b40a2c58ec0eb60f50c
2 Nodes configured, 2 expected votes
0 Resources configured.
============

Online: [ node2 node1 ]
#_
```

FIGURE 4.13: Pacemake 運行狀況

4.3.5 tpcc-mysql 測試

透過圖 4-14 的指令可以創建一個測試的資料表，圖 4-15 為資料表載入的狀態

```
# mysqladmin -h 10.0.42.100 -u root -p tpcctest create tpcc
# mysql -h 10.0.42.100 -u root -p tpcctest tpcc < ./create_table.sql
# mysql -h 10.0.42.100 -u root -p tpcctest tpcc < ./add_fkey_idx.sql
# ./tpcc_load -h 10.0.42.100 tpcc "tpcctest" 1000
#
#_
```

FIGURE 4.14: 創建測試環境

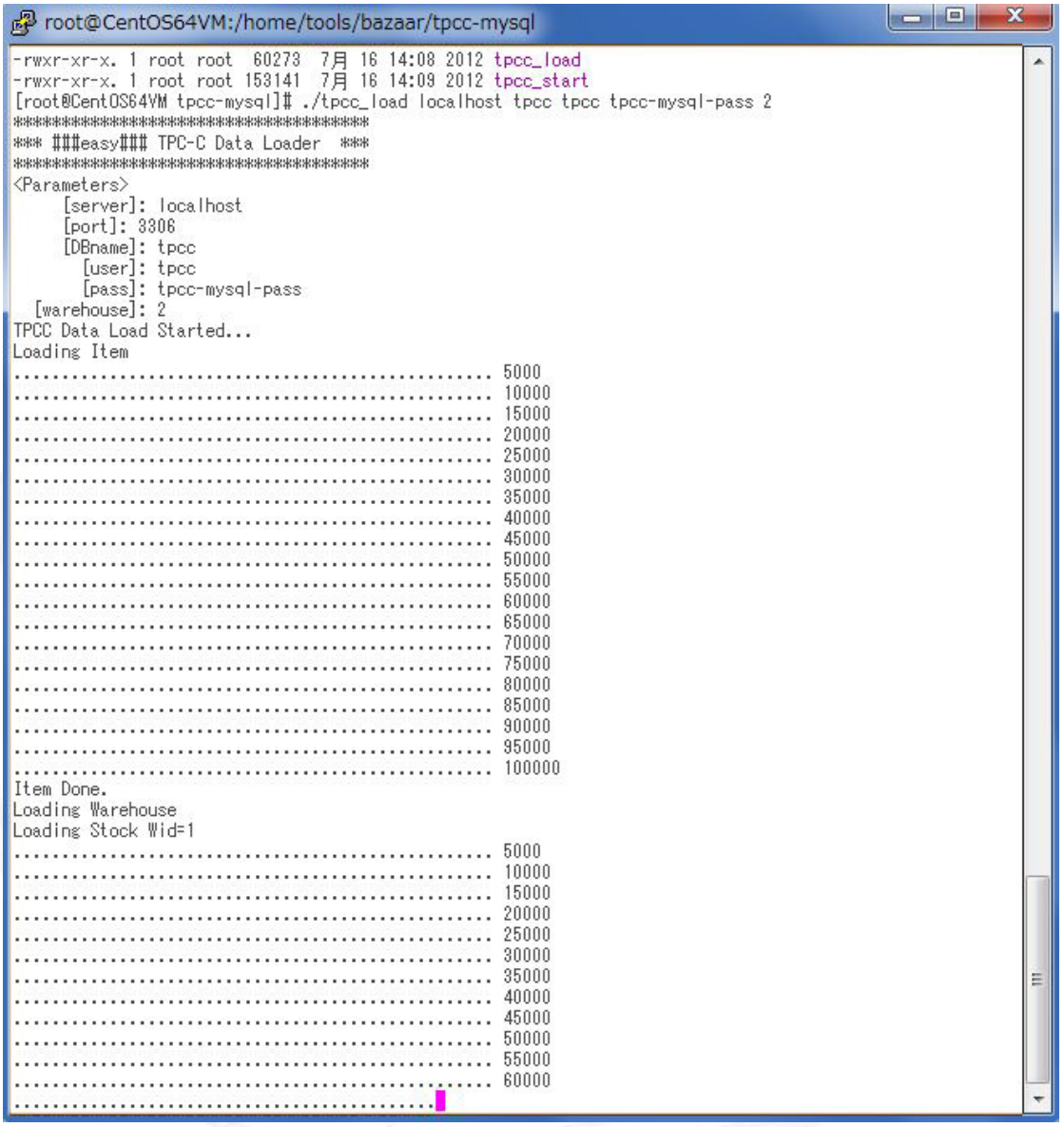

FIGURE 4.15: tpcc_load 載入資料狀態

完成上述資料載入後，即可使用 tpcc_start 來產生 MySQL 的實際流量，圖 4-16 是 tpcc_start 運行的結果

```
[root@ABPNOSS01 tpcc-mysql]# ./tpcc_start -h10.0.42.100 -d tpcc -u root -w 10 -c 8 -r 2 -l 300
****************************************
*** ###easy### TPC-C Load Generator ***
****************************************
option h with value '10.0.42.100'
option d with value 'tpcc'
option u with value 'root'
option w with value '10'
option c with value '8'
option r with value '2'
option l with value '300'
<Parameters>
     [server]: 10.0.42.100
     [port]: 3306
     [DBname]: tpcc
       [user]: root
       [pass]:
  [warehouse]: 10
 [connection]: 8
     [rampup]: 2 (sec.)
    [measure]: 300 (sec.)

RAMP-UP TIME.(2 sec.)

MEASURING START.

  10, 6104(0):2.168|3.596, 6103(0):0.443|0.723, 611(0):0.226|0.318, 610(0):2.932|4.293, 611(0):7.759|9.642
  20, 5557(0):2.308|3.623, 5558(0):0.517|0.705, 555(0):0.248|0.287, 556(0):3.345|4.046, 556(0):7.425|9.947
  30, 2628(0):2.402|3.598, 2631(0):0.570|0.738, 264(0):0.271|0.335, 262(0):3.353|4.059, 262(0):7.664|9.717
  40, 6048(0):2.363|3.654, 6046(0):0.507|0.709, 605(0):0.261|0.316, 606(0):3.392|4.638, 605(0):7.672|9.378
  50, 4701(0):2.316|3.469, 4703(0):0.534|0.703, 470(0):0.269|0.315, 469(0):3.312|4.164, 471(0):7.517|8.714
  60, 4539(0):2.333|3.629, 4535(0):0.532|0.746, 454(0):0.263|0.338, 455(0):3.156|4.056, 453(0):9.582|11.027
  70, 5540(0):2.285|3.802, 5544(0):0.465|0.656, 553(0):0.208|0.284, 553(0):3.239|3.675, 555(0):8.065|9.772
  80, 4205(0):2.306|3.561, 4202(0):0.503|0.686, 421(0):0.233|0.319, 420(0):3.230|3.754, 421(0):7.890|10.382
  90, 5442(0):2.426|3.859, 5444(0):0.500|0.709, 544(0):0.259|0.327, 545(0):3.371|3.926, 542(0):7.416|9.846
 100, 4716(0):2.360|3.577, 4713(0):0.512|0.688, 472(0):0.231|0.307, 471(0):2.852|4.221, 472(0):8.217|8.971
 110, 4203(0):2.413|4.013, 4207(0):0.477|0.694, 420(0):0.227|0.304, 421(0):3.299|3.985, 420(0):8.557|11.166
 120, 5223(0):2.360|4.101, 5222(0):0.483|0.695, 522(0):0.244|0.345, 523(0):3.377|4.077, 523(0):8.236|10.825
 130, 4273(0):2.351|3.646, 4273(0):0.518|0.657, 428(0):0.277|0.323, 427(0):2.914|3.657, 429(0):7.987|9.654
 140, 2722(0):2.247|3.384, 2715(0):0.514|0.738, 271(0):0.242|0.313, 272(0):2.821|3.560, 272(0):7.397|8.002
 150, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000
 160, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000
 170, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000
 180, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000
neword 6:2
payment 1:1
1205, HY000, Lock wait timeout exceeded; try restarting transaction
1205, HY000, Lock wait timeout exceeded; try restarting transaction
payment 4:3
1205, HY000, Lock wait timeout exceeded; try restarting transaction
payment 7:1
1205, HY000, Lock wait timeout exceeded; try restarting transaction
 190, 8(0):0.862|1.619, 11(0):0.418|0.422, 1(0):0.000|0.159, 2(0):0.809|1.531, 0(0):0.000|0.000
 200, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000, 0(0):0.000|0.000
```

FIGURE 4.16: tpcc-mysql 測試

4.3.6 高可用性測試

測試方式主要分為：

(1) 模擬軟體失效：以手動方式將運行中的服務關閉，測試是否能夠回復正常服務，並測量回復時間。

(2) 模擬硬體失效：手動方式將實體主機電源關閉，模擬硬體發生錯誤的狀況，測試是否能夠回復正常服務，並測量回復時間。

(3) 模擬 DRBD+HeartBeat 與 Cluster HA 合併架構中其中一整個 Cluster 節點損壞的狀況，並測量回復時間。

4.4 實驗結果

4.4.1 軟體失效

從表 4-1、4-2 我們可以看出除了單節點架構因為沒有任何 HA 機制，以致於在軟體失效時完全無法回復外，其他架構在連續進行 30 次測試後皆可以正常回復服務。

TABLE 4.1: 軟體失效 Apache HA 測試結果

架構	VMware Cluster	OpenStack Cluster
單節點架構	N	N
DRBD+Heartbeat 架構	Y	Y
DRBD+HeartBeat 與 Cluster HA 合併架構	Y	Y

TABLE 4.2: 軟體失效 MySQL HA 測試結果

架構	VMware Cluster	OpenStack Cluster
單節點架構	N	N
DRBD+Heartbeat 架構	Y	Y
DRBD+HeartBeat 與 Cluster HA 合併架構	Y	Y

圖 4-17、圖 4-18 分別是 Apache 及 MySQL 在軟體失效回復測試中的結果，在此圖表中，排除了單節點架構的數據，因為此架構無法自軟體失效中回復。從圖表中可以看出在軟體失效測試中，個別環境之間的差異很小，Apache 回復時間皆在 45 60 秒的範圍內；而 MySQL 則在 70 100 秒內。這是因為服務的狀態由 Hearbeat 這個套建在監控，當發現服務停止的狀況，將會自動將服務切換至存活的 Node，並利用 DRBD 已同步的資料繼續進行服務。

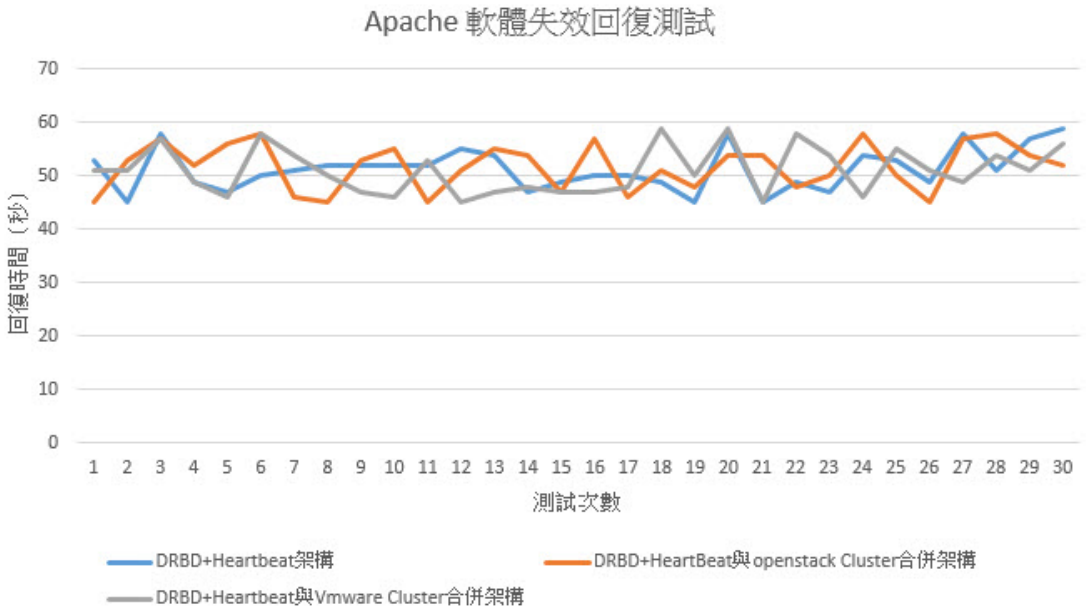

FIGURE 4.17: Apache 軟體失效回復測試時間

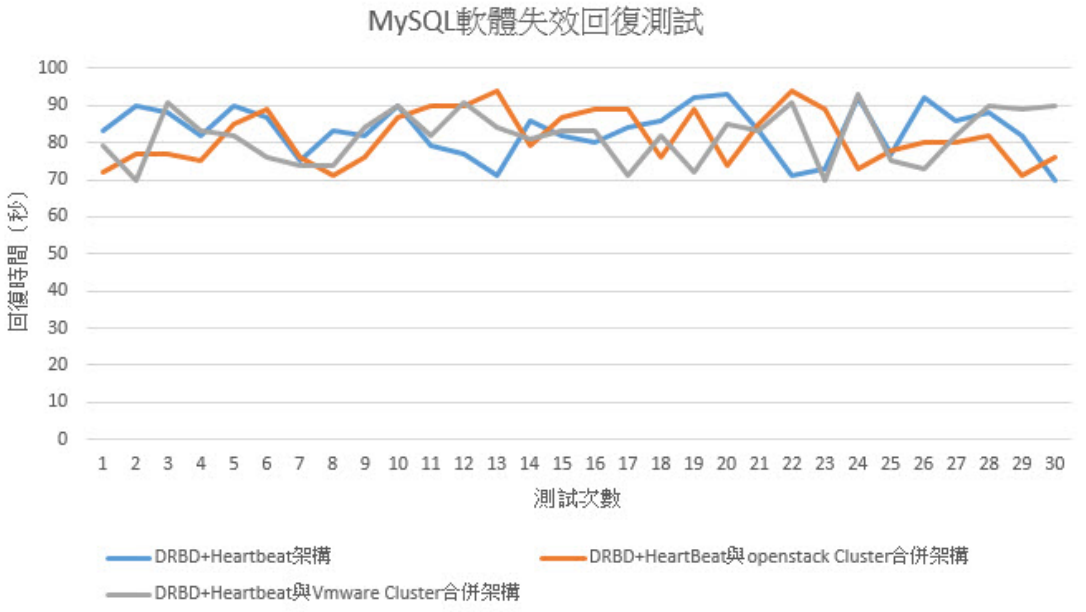

FIGURE 4.18: MySQL 軟體失效回復測試時間

4.4.2 硬體失效

從表 4-3、4-4 我們再次可以看到單節點架構因為沒有任何 HA 機制，導致無法回復的狀況，其他架構在連續進行 30 次測試後皆可以正常回復服務。

TABLE 4.3: 硬體失效 Apache HA 測試結果

架構	VMware Cluster	OpenStack Cluster
單節點架構	X	X
DRBD+Heartbeat 架構	O	O
DRBD+HeartBeat 與 Cluster HA 合併架構	O	O

TABLE 4.4: 硬體失效 MySQL HA 測試結果

架構	VMware Cluster	OpenStack Cluster
單節點架構	X	X
DRBD+Heartbeat 架構	O	O
DRBD+HeartBeat 與 Cluster HA 合併架構	O	O

圖 4-19、圖 4-20 分別是 Apache 及 MySQL 在硬體失效回復測試中的結果，在此圖表中，同樣排除了單節點架構的數據，因為此架構無法也從硬體失效中回復。從圖表中可以看出 DRBD+Heartbeat 架構明顯地回復時間較其他兩者短，原因是在此架構中，當硬體發生失效服務是立刻轉移至另外一個節點。但在混合架構中，Cluster 必須進行即時轉移，之後其上的 VM 會恢復正常運作，此時整個服務才會重新運作。而 VMware 回復時間則又短於 OpenStack，這是因為即時轉移的速度所影響。

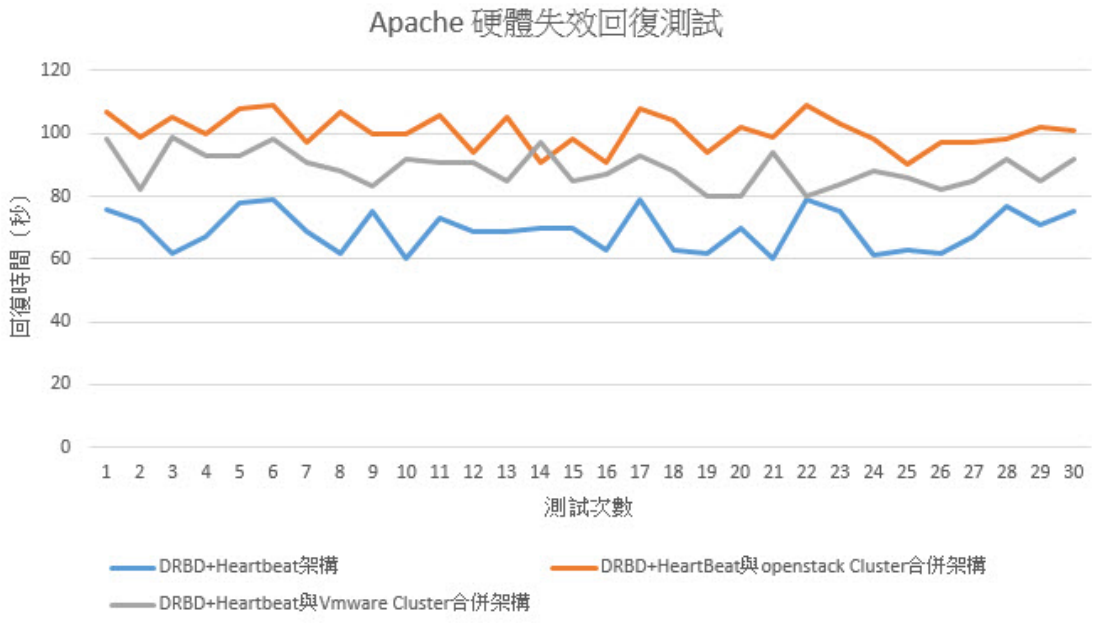

FIGURE 4.19: Apache 硬體失效回復測試時間

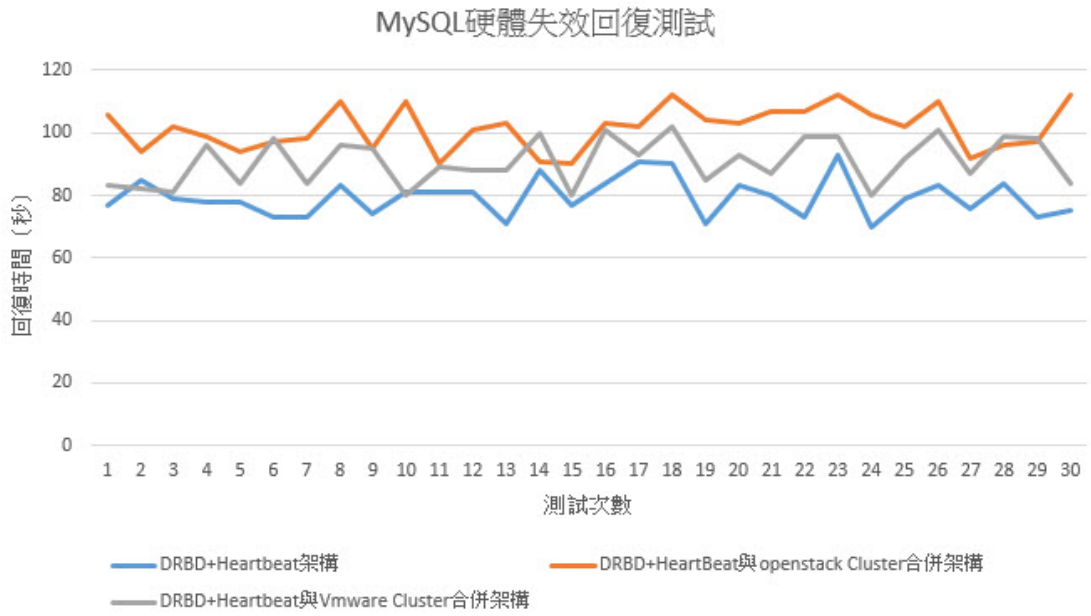

FIGURE 4.20: MySQL 硬體失效回復測試時間

4.4.3 Cluster 硬體失效

從表 4-5、4-6 我們可以看到 DRBD+Heartbeat 架構中，因為整個 Cluster 已經完全失效，其上的所有虛擬機器也隨之無法使用，而合併架構在連續進行 30 次測試後皆可以正常回復服務。

TABLE 4.5: Cluster 失效 Apache HA 測試結果

架構	VMware Cluster	OpenStack Cluster
DRBD+Heartbeat 架構	X	X
DRBD+HeartBeat 與 Cluster HA 合併架構	O	O

TABLE 4.6: Cluster 失效 MySQL HA 測試結果

架構	VMware Cluster	OpenStack Cluster
DRBD+Heartbeat 架構	X	X
DRBD+HeartBeat 與 Cluster HA 合併架構	O	O

圖 4-21、圖 4-22 分別是 Apache 及 MySQL 在 Cluster 失效回復測試中的結果，在此圖表中，排除了 DRBD+Heartbeat 的數據，因為整個 Cluster 已經完全失效，其上的所有虛擬機器也隨之無法使用。在此一模擬測試中，可以發

現 VMware Cluster 明顯回復時間都較短、個別測試回復時間的差異也較小。而 Openstack 的回復時間不但比較長，每次測試的時間也較為不穩定、變動較大。

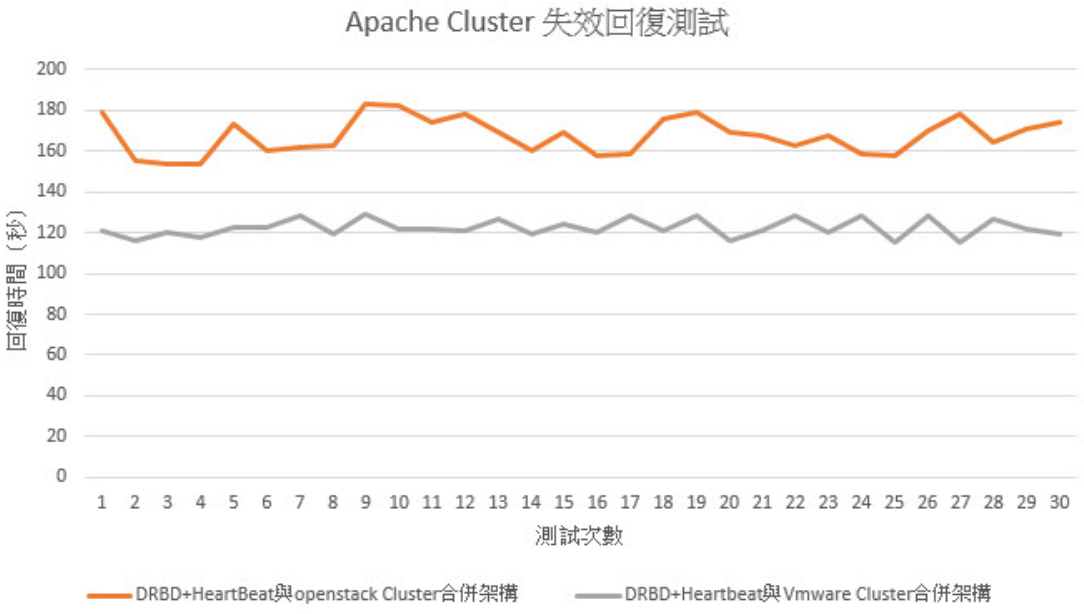

FIGURE 4.21: Apache Cluster 失效回復測試時間

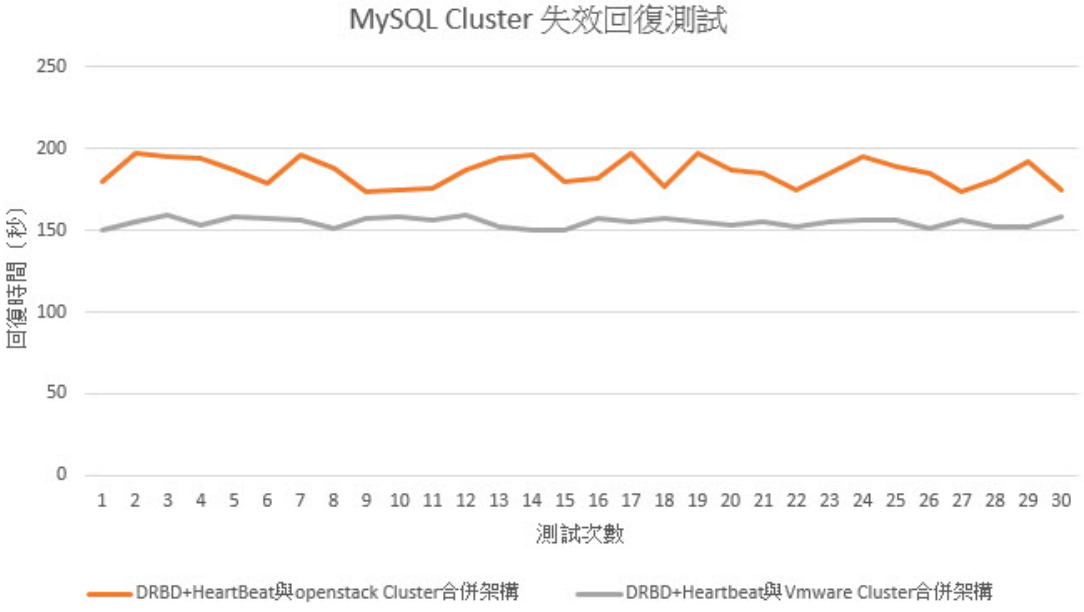

FIGURE 4.22: MySQL Cluster 失效回復測試時間

Chapter 5

結論與未來方向

5.1 結論

雲端服務對於現代資訊社會來說日益重要，因為雲端服務可帶來極大的經濟效應。它們可幫助企業降低資本支出和營運開支，共用硬體，並提供動態、按照需求的服務調配方式。此外，還提供靈活的資源分配模式，可以更快速、有效地啟動擴展服務。隨著企業及個人都越來越仰賴這些各項雲端服務。因此，身為雲端服務的提供者，如何維持系統穩定責任也就越來越重大，思考如何穩定地提供 24 小時不間斷的資訊服務也就成為了重要的議題。在本文中，嘗試以 mysql 及 apache 兩項常見的網路服務作為實驗工具，藉以模擬實際的雲端服務，發現到不管是 VMware 或是 OpenStack 環境下，透過 Cluster HA 的技術建構硬體的容錯叢集確實可以達到軟硬體上的服務不間斷；然而在沒有共享儲存的情況下，服務仍舊會因為虛擬機器無法順利遷移而造成問題。如果加上了 DRBD 儲存備援以及 Heartbeat 的軟體備援監控，確實更進一步提供了較為完整的軟硬體 HA 機制。並且可以在不用耗費大筆金錢購置額外的共用儲存空間下達成這樣的目的。在本文中更探討了以 DRBD 結合 VMware 或 Openstack 的硬體 HA，讓 Cluster 也發生失效時，還能夠順利轉移至另外一個備援的 Cluster 上繼續運作。因此，透過結合 Virtualization Cluster 及 DRBD 技術，確實能達成資訊服務不中斷的雙重保障。在實作中，也驗證了以 VMware 及 OpenStack

作為虛擬化叢集平台，在容錯轉移機制下來處理實體主機失效，結合 DRBD 處理軟體失效後，在各種軟硬體失效的情況下，在實際應用中，此雙重高可用性雲端服務可在硬體失效及軟體失效中自動回復，以確保雲端服務不會受到影響，縮短服務中斷的時間。然而這樣的架構有著回復時間較長的問題存在，這是未來可以進一步思考如何找出架構上的不足，修改架構或是增進效率。

5.2 未來方向

虛擬化平台對於儲存設備 IOPS (Input/Output Operations Per Second, pronounced eye-ops) 越來越高，目前也有越來越多技術是針對儲存設備在效率上的提升。目前最新的技術是透過儲存虛擬化的方式將各實體主機間的磁碟模擬成一塊完整的共享儲存。在探討如何提高虛擬化平台穩定性、可用性之後，或許未來可以思考如何結合穩定與效能，找出更快更穩定的解決方案。

參考文獻

[1] Justin F. Brunelle and Michael L. Nelson. Evaluating the sitestory transactional web archive with the apachebench tool. *CoRR*, abs/1209.1811, 2012.

[2] Justin F. Brunelle, Michael L. Nelson, Lyudmila Balakireva, Robert Sanderson, and Herbert Van de Sompel. Evaluating the sitestory transactional web archive with the apachebench tool. In *Research and Advanced Technology for Digital Libraries - International Conference on Theory and Practice of Digital Libraries, TPDL 2013, Valletta, Malta, September 22-26, 2013. Proceedings*, pages 204–215, 2013.

[3] Scott T. Leutenegger and Daniel M. Dias. A modeling study of the TPC-C benchmark. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993.*, pages 22–31, 1993.

[4] Judith A. Piantedosi, Archana S. Sathaye, and D. John Shakshober. Performance measurement of trucluster systems under the TPC-C benchmark. *Digital Technical Journal*, 8(3), 1996.

[5] Yanpei Chen, Francois Raab, and Randy H. Katz. From TPC-C to big data benchmarks: A functional workload model. In *Specifying Big Data Benchmarks - First Workshop, WBDB 2012, San Jose, CA, USA, May 8-9, 2012, and Second Workshop, WBDB 2012, Pune, India, December 17-18, 2012, Revised Selected Papers*, pages 28–43, 2012.

[6] Jinhui Yao, Alex Ng, Shiping Chen, Dongxi Liu, Carsten Friedrich, and Surya Nepal. A performance evaluation of public cloud using TPC-C. In *Service-Oriented Computing - ICSOC 2012 Workshops - ICSOC 2012, International Workshops ASC, DISA, PAASC, SCEB, SeMaPS, WESOA, and Satellite Events, Shanghai, China, November 12-15, 2012, Revised Selected Papers*, pages 3–13, 2013.

[7] Robayet Nasim and Andreas J. Kassler. Deploying openstack: Virtual infrastructure or dedicated hardware. In *IEEE 38th Annual Computer Software and Applications Conference, COMPSAC Workshops 2014, Vasteras, Sweden, July 21-25, 2014*, pages 84–89, 2014.

[8] Juan Angel Lorenzo del Castillo, Kate Mallichan, and Yahya Al-Hazmi. Openstack federation in experimentation multi-cloud testbeds. In *IEEE 5th International Conference on Cloud Computing Technology and Science, CloudCom 2013, Bristol, United Kingdom, December 2-5, 2013, Volume 2*, pages 51–56, 2013.

[9] Xiaoen Ju, Livio Soares, Kang G. Shin, Kyung Dong Ryu, and Dilma Da Silva. On fault resilience of openstack. In *ACM Symposium on Cloud Computing, SOCC '13, Santa Clara, CA, USA, October 1-3, 2013*, pages 2:1–2:16, 2013.

[10] Tiago Rosado and Jorge Bernardino. An overview of openstack architecture. In *18th International Database Engineering & Applications Symposium,IDEAS 2014, Porto, Portugal, July 7-9, 2014*, pages 366–367, 2014.

[11] Arvinder Kaur and Shraddha Verma. Performance measurement and analysis of high-availability clusters. *ACM SIGSOFT Software Engineering Notes*, 40(2):1–7, 2015.

[12] Salvatore Distefano. Using DRBD in the design of redundant distributed computing system. In *2014 International Conference on Intelligent Networking and Collaborative Systems, Salerno, Italy, September 10-12, 2014*, pages 274–281, 2014.

[13] Salvatore Distefano. Standby system reliability through DRBD. In *2014 IEEE International Parallel & Distributed Processing Symposium Workshops, Phoenix, AZ, USA, May 19-23, 2014*, pages 1330–1337, 2014.

[14] Salvatore Distefano and Antonio Puliafito. Reliability and availability analysis of dependent-dynamic systems with drbds. *Rel. Eng. & Sys. Safety*, 94(9): 1381–1393, 2009.

[15] Thandar Thein, Sung-Do Chi, and Jong Sou Park. Improving fault tolerance by virtualization and software rejuvenation. In *Second Asia International Conference on Modelling and Simulation, AMS 2008, Kuala Lumpur, Malaysia, May 13-15, 2008*, pages 855–860, 2008.

[16] Gracjan Jankowski, Rafal Mikolajczak, Radoslaw Januszewski, Norbert Meyer, and Maciej Stroinski. Resources virtualization in fault-tolerance and migration issues. In *Computational Science - ICCS 2004, 4th International Conference, Kraków, Poland, June 6-9, 2004, Proceedings, Part I*, pages 449–452, 2004.

[17] Salvatore Distefano, Marco Scarpa, and Antonio Puliafito. Modeling distributed computing system reliability with DRBD. In *25th IEEE Symposium on Reliable Distributed Systems (SRDS 2006),2-4 October 2006, Leeds, UK*, pages 106–118, 2006.

[18] Chao-Tung Yang, Jung-Chun Liu, Ching-Hsien Hsu, and Wei-Li Chou. On improvement of cloud virtual machine availability with virtualization fault tolerance mechanism. *The Journal of Supercomputing*, 69(3):1103–1122, 2014.

[19] Chao-Tung Yang, Cheng-Ta Kuo, Wen-Hung Hsu, and Wen-Chung Shih. A medical image file accessing system with virtualization fault tolerance on cloud. In *Advances in Grid and Pervasive Computing - 7th International Conference, GPC 2012, Hong Kong, China, May 11-13, 2012. Proceedings*, pages 338–349, 2012.

[20] William von Hagen. Chapter 1: Overview of virtualization. In *Professional Xen Virtualization*, pages 1–26, 2008.

[21] Chao-Tung Yang, Wei-Li Chou, Ching-Hsien Hsu, and Alfredo Cuzzocrea. On improvement of cloud virtual machine availability with virtualization fault tolerance mechanism. In *IEEE 3rd International Conference on Cloud Computing Technology and Science, CloudCom 2011, Athens, Greece, November 29 - December 1, 2011*, pages 122–129, 2011.

[22] Michail D. Flouris, Renaud Lachaize, Konstantinos Chasapis, and Angelos Bilas. Extensible block-level storage virtualization in cluster-based systems. *Journal of Parallel and Distributed Computing*, 70(8):800 – 824, 2010.

[23] Rajkumar Buyya, Christian Vecchiola, and S. Thamarai Selvi. Chapter 3 - virtualization. In *Mastering Cloud Computing*, pages 71 – 109. Morgan Kaufmann, 2013.

[24] Qian Lin, Zhengwei Qi, Jiewei Wu, Yaozu Dong, and Haibing Guan. Optimizing virtual machines using hybrid virtualization. *Journal of Systems and Software*, 85(11):2593 – 2603, 2012.

[25] Simon Grinberg and Shlomo Weiss. Architectural virtualization extensions: A systems perspective. *Computer Science Review*, 6(5–6):209 – 224, 2012.

[26] Yaozu Dong, Xiaowei Yang, Jianhui Li, Guangdeng Liao, Kun Tian, and Haibing Guan. High performance network virtualization with sr-iov. *Journal of Parallel and Distributed Computing*, 72(11):1471 – 1480, 2012. Communication Architectures for Scalable Systems.

[27] Fernando Rodríguez-Haro, Felix Freitag, Leandro Navarro, Efraín Hernánchez-sánchez, Nicandro Farías-Mendoza, Juan Antonio Guerrero-Ibáñez, and Apolinar González-Potes. A summary of virtualization techniques. *Procedia Technology*, 3(0):267 – 272, 2012. The 2012 Iberoamerican Conference on Electronics Engineering and Computer Science.

[28] Gary Lee. Chapter 6 - server virtualization and networking. In Gary Lee, editor, *Cloud Networking*, pages 103 – 120. Morgan Kaufmann, Boston, 2014.

[29] Diane Barrett and Gregory Kipper. 2 - server virtualization. In Diane Barrett and Gregory Kipper, editors, *Virtualization and Forensics*, pages 25 – 36. Syngress, Boston, 2010.

[30] Zhaojun Li, Haijiang Li, Xicheng Wang, and Keqiu Li. A generic cloud platform for engineering optimization based on openstack. *Advances in Engineering Software*, 75(0):42 – 57, 2014.

[31] Antonio Corradi, Mario Fanelli, and Luca Foschini. {VM} consolidation: A real case based on openstack cloud. *Future Generation Computer Systems*, 32(0):118 – 127, 2014. Special Section: The Management of Cloud Systems, Special Section: Cyber-Physical Society and Special Section: Special Issue on Exploiting Semantic Technologies with Particularization on Linked Data over Grid and Cloud Architectures.

[32] Wikipedia. Distributed replicated block device. `http://en.wikipedia.org/wiki/Distributed_Replicated_Block_Device`, 2015. [Online; accessed 19-May-2015].

附錄 A

OpenSTACK Installation on Ubuntu

一、Lab esnvironment

- CPU: Intel Core i5-3210M 2.5GHz
- RAM: 8 GB
- HD: 500GB
- Network: 100M/1000M bps Ethernet
- OS: Windows 8.1 Professional 64-bit
- VM Platform: VMware Workstation 10.0.0
- VM Guest OS: Ubuntu 12.04.1 Server 64-bit
- VM RAM: 4GB
- VM HD: 20GB

PS: Rackspace recommends that the Chef server hardware meet the following requirements:

- 4 GB VM RAM
- 20 GB VM disk space or more
- Single socket dual core, or dual socket CPU with dual core, or single socket quad core

二、OpenStack and Rackspace Private Cloud Software presentation

1. Understanding the OpenStack Framework

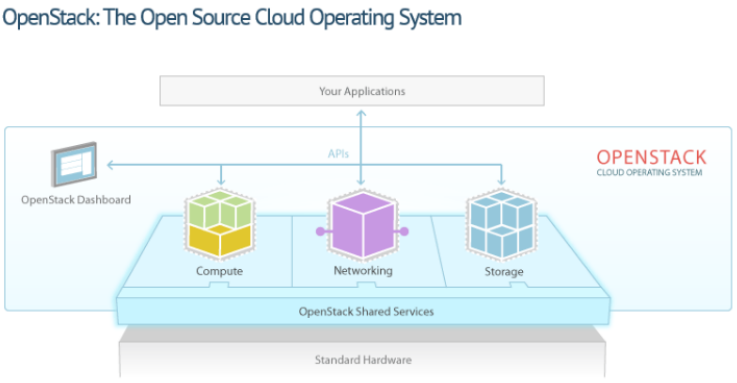

FIGURE A.1: Openstack Cloud Architecture

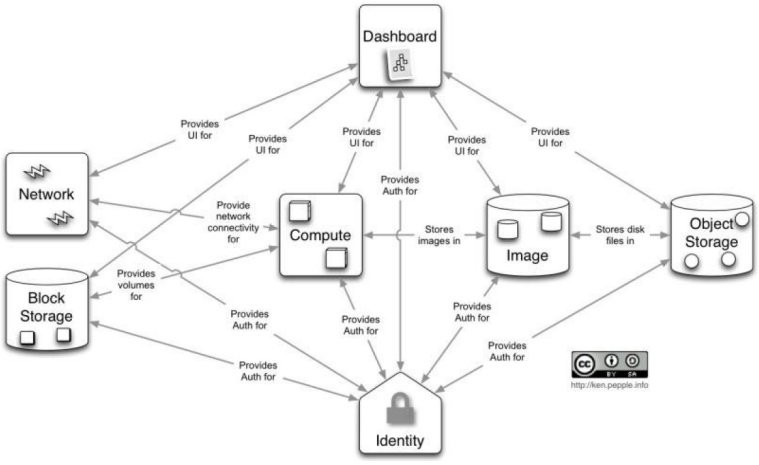

FIGURE A.2: OpenSatck Conceptual Diagram

來源: http://docs.openstack.org

快速瀏覽OpenStack 7大套件

套件名稱	套件功能
運算套件Nova	部署與管理虛擬機器的功能
物件儲存套件 Swift	可擴展的分布式儲存平臺，以防止單點故障的情況產生，可存放非結構化的資料
區塊儲存套件 Cinder	整合了運算套件，可讓IT人員查看儲存設備的容量使用狀態，具有快照功能
網通套件 Quantum	可擴展、隨插即用，透過API來管理的網路架構系統，以確保IT人員在部署雲端服務時，網路服務不會出現瓶頸，或是成為無法部署的因素之一
身分識別套件 Keystone	具有中央目錄，能查看哪位使用者可存取哪些服務，並且，提供了多種驗證方式
映象檔管理套件 Glance	硬碟或伺服器的映象檔尋找、註冊以及服務交付等功能
儀表板套件 Horizon	圖形化的網頁介面，讓IT人員可以綜觀雲端服務目前的規模與狀態，並能夠統一存取、部署與管理所有雲端服務所使用到的資源。

FIGURE A.3: OpenSatck Key Element

2. Understanding the Rackspace Private Cloud Software Framework Start building your private cloud today. Rackspace Private Cloud combines the power of OpenStack—the leading open-source cloud operating system—with a streamlined

deployment process that allows you to get your private cloud up and running in less than an hour.

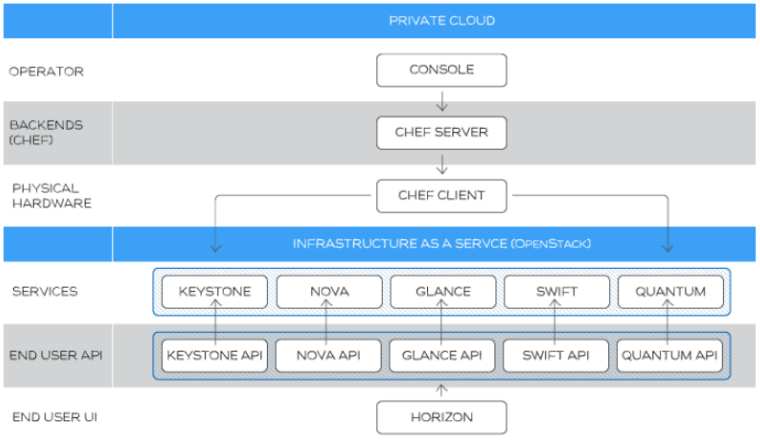

FIGURE A.4: Openstack with Rackspace Private Cloud

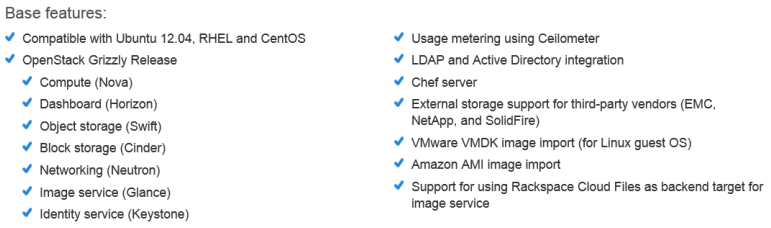

FIGURE A.5: Base features

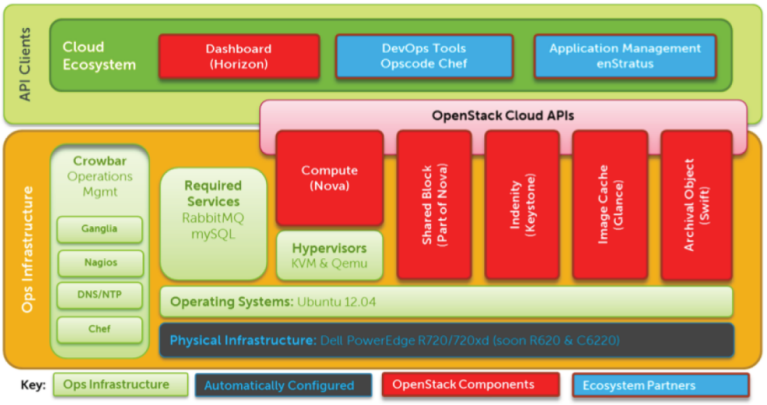

FIGURE A.6: OpenStack-Powered Cloud

來源: http://www.rackspace.com/cloud/private/openstack_software/ http://www.yet.org/2013/06/crowbar-rc1/

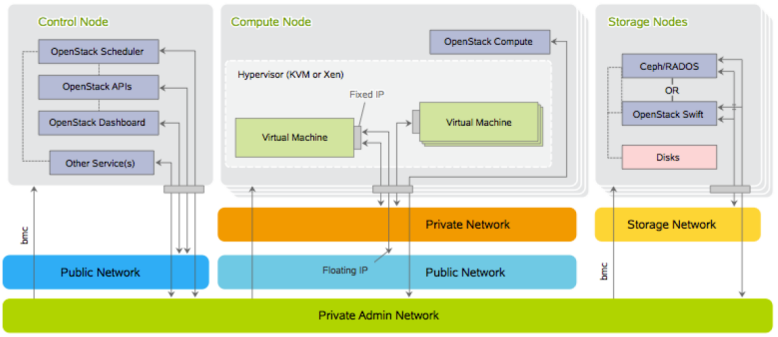

FIGURE A.7: Openstack Network Architecture

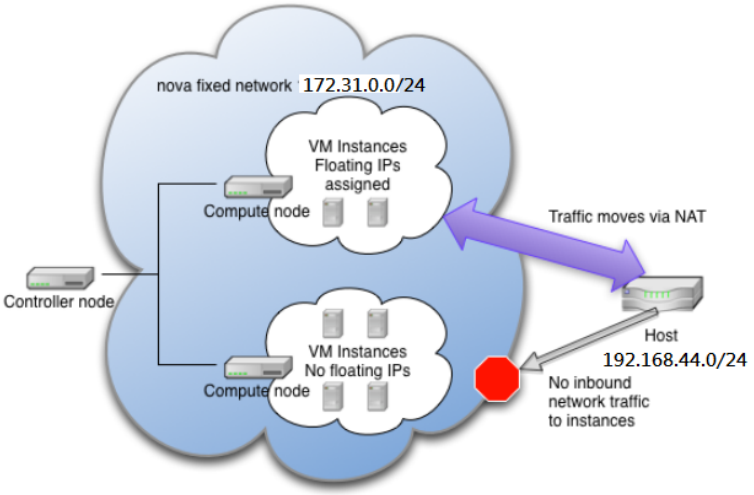

FIGURE A.8: NAT access to instances on the cloud

三、BIOS setup

檢查電腦開機前 BIOS SETUP，有無存在 Virtual Machine Mode 為 Enabled

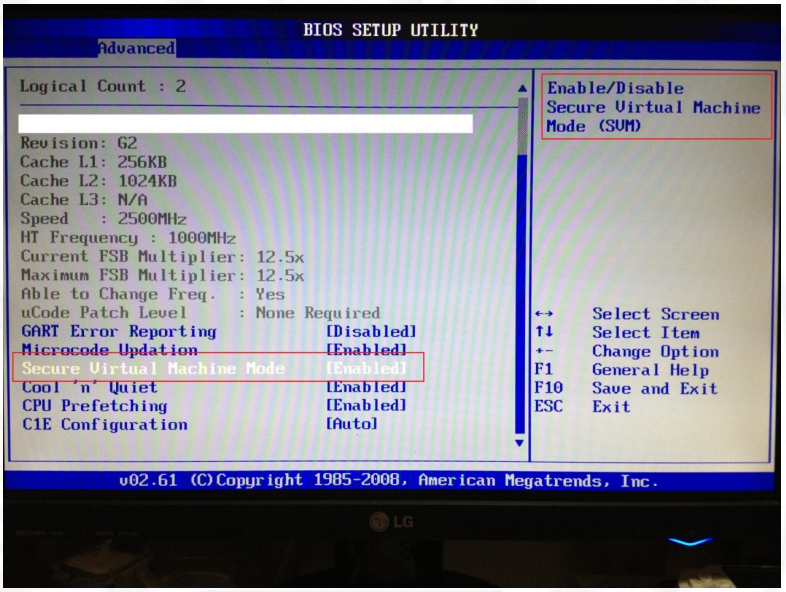

FIGURE A.9: BIOS Setup

四、Installation of Rackspace Private Cloud Software powerd by OpenSTACK

1. 開始執行 Rackspace Private Cloud installation on Virtual machine

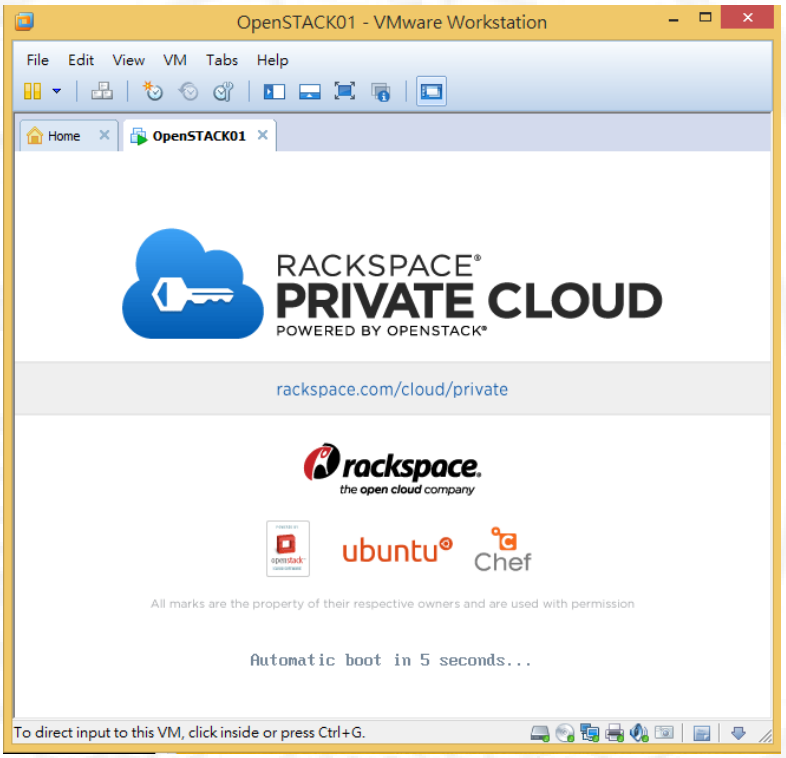

FIGURE A.10: Rackspace Private Cloud installation

2. 正在載入安裝附加元件

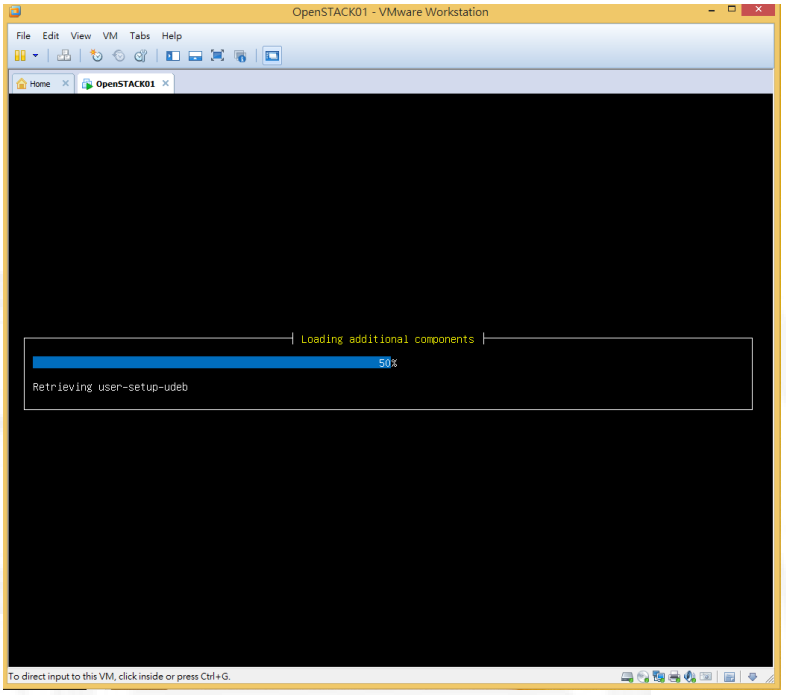

FIGURE A.11: Rackspace Private Cloud installation-載入安裝附加元件

3. 進入 Rackspace Private Cloud Software License Agreement

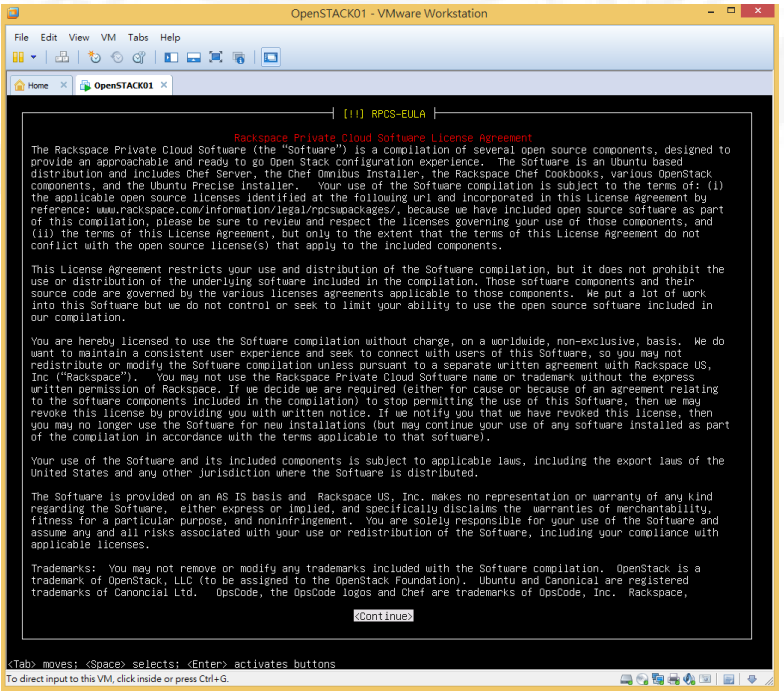

FIGURE A.12: Rackspace Private Cloud Software License Agreement

4. RPCS Pre-Networking - Choose the role for this node: All-In-One.

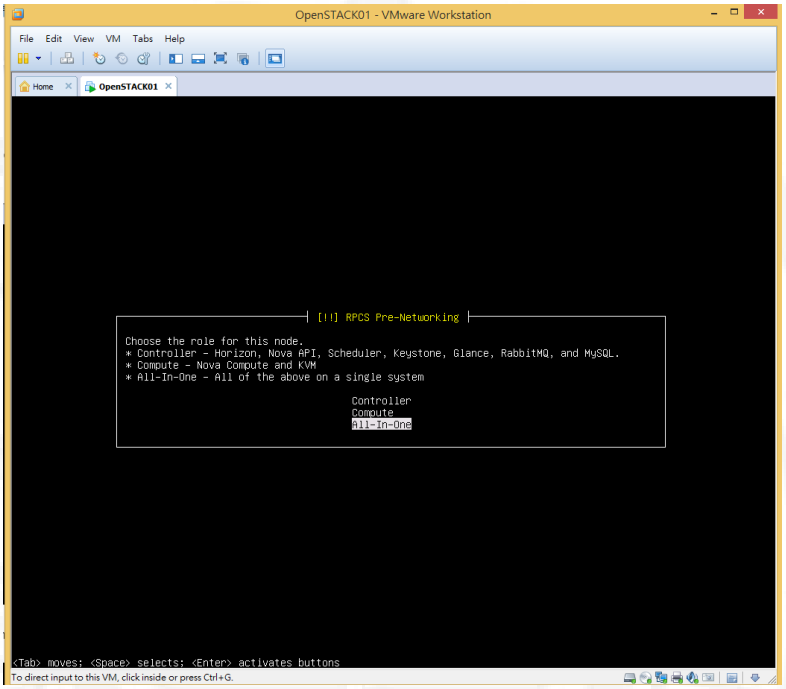

FIGURE A.13: RPCS Pre-Networking

5. Configure the network - IP Address: 192.168.44.80/24

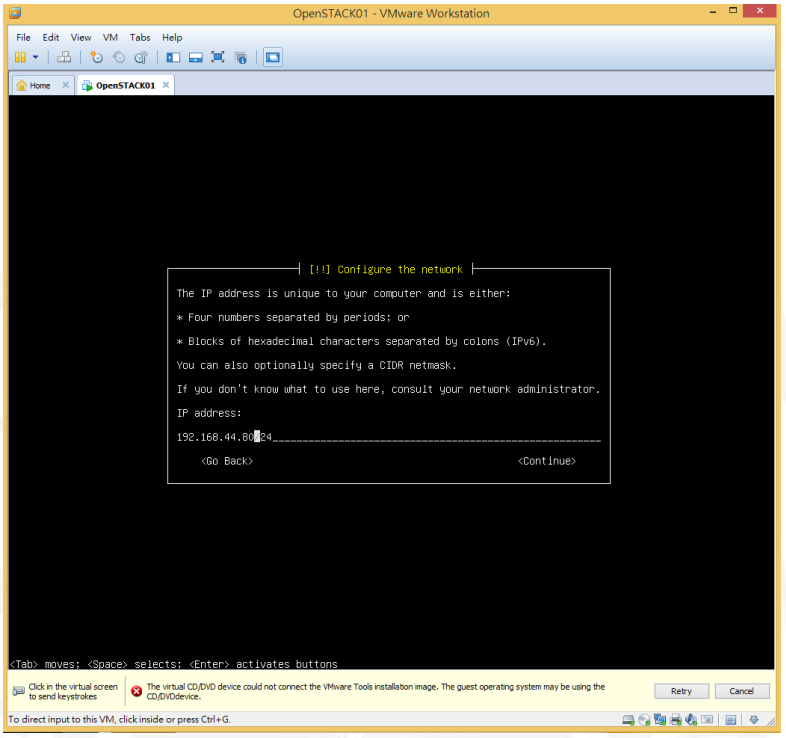

FIGURE A.14: Configure the network(IP Address)

6. Configure the network - Getway: 192.168.44.2

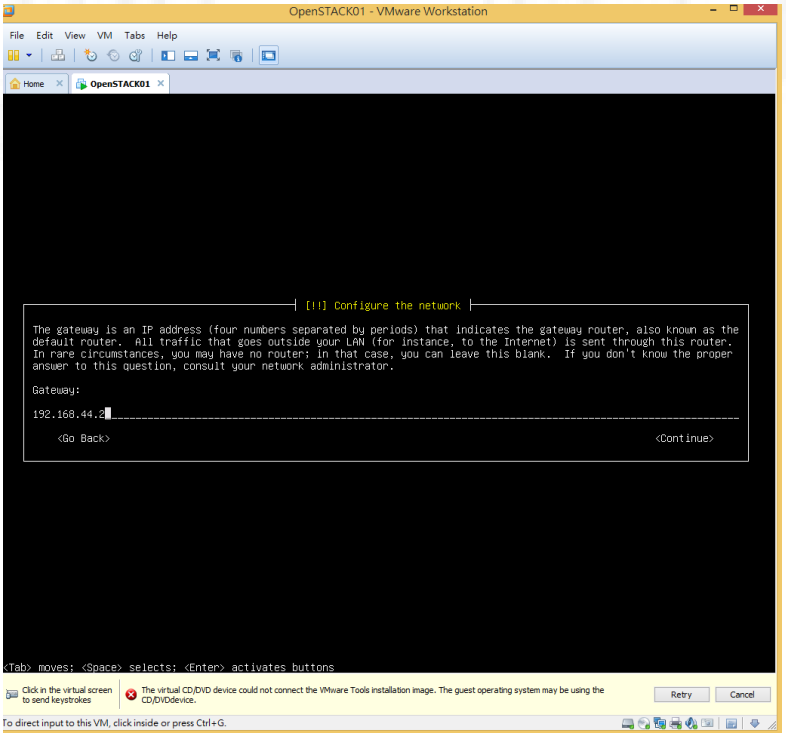

FIGURE A.15: Configure the network(Gateway)

7. Configure the network - Name server address: 8.8.8.8

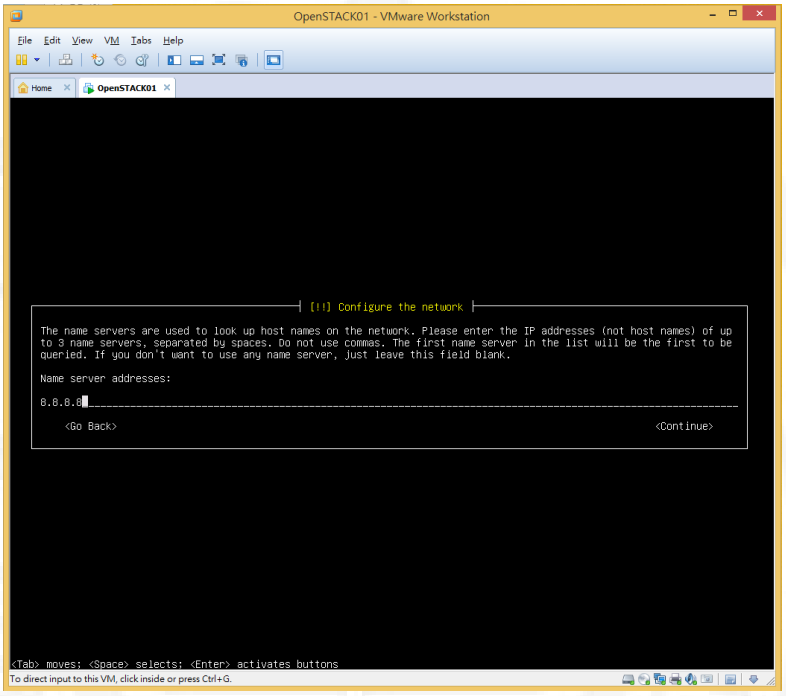

FIGURE A.16: Configure the network(Name Server)

8. Configure the network - Hostname: openstack01

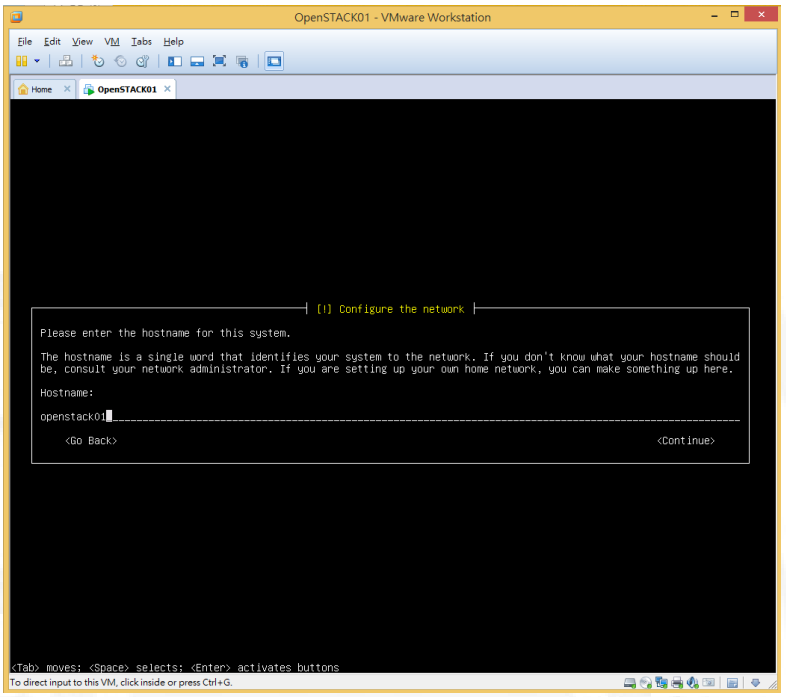

FIGURE A.17: Configure the network(Hostname)

9. Configure the network - Domain name: openstack01.thu.edu.tw

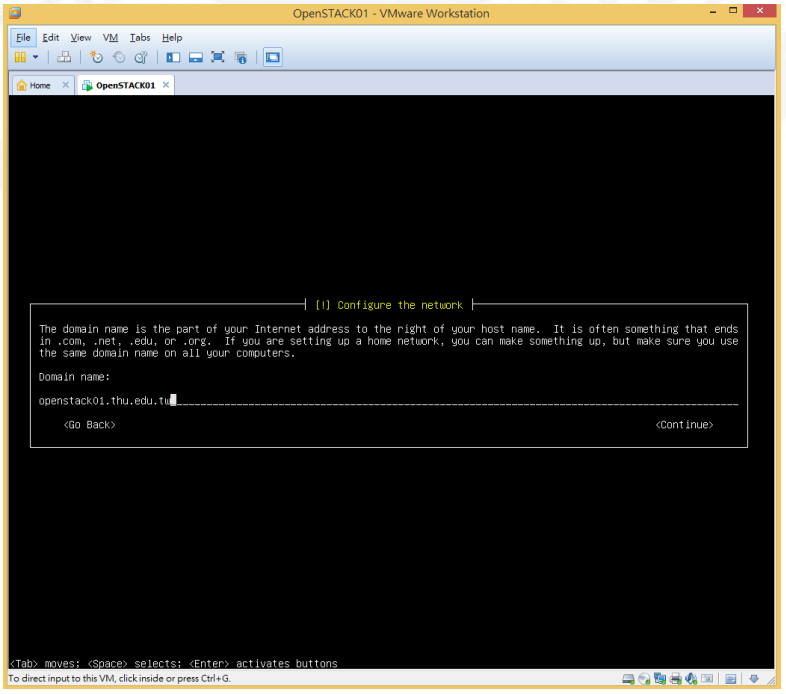

FIGURE A.18: Configure the network(Domanin name)

10. Enter CIDR block for Nova Fixed (VM) network: 172.31.0.0/24

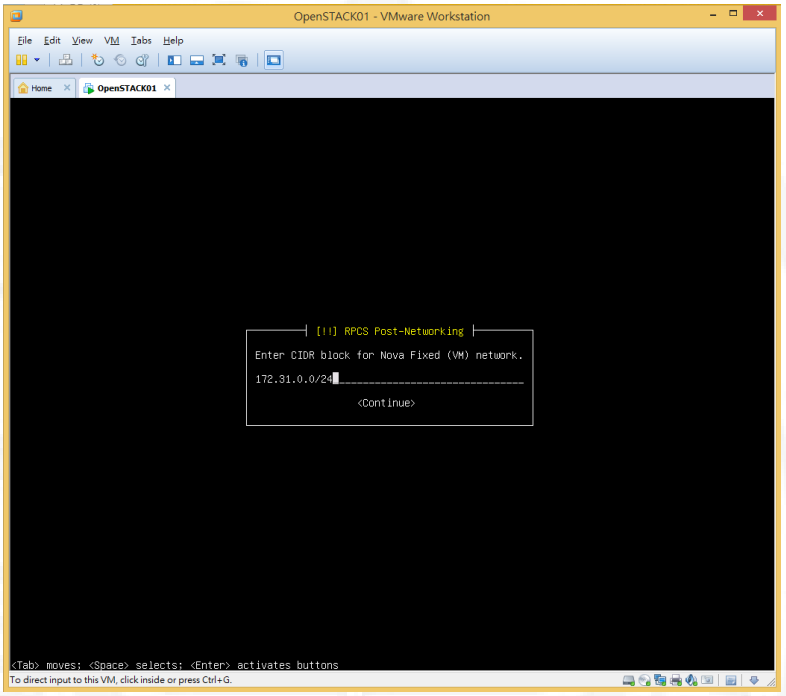

FIGURE A.19: Enter CIDR block for Nova Fixed (VM) network

11. Enter a password for the Openstack "admin" user: openstack

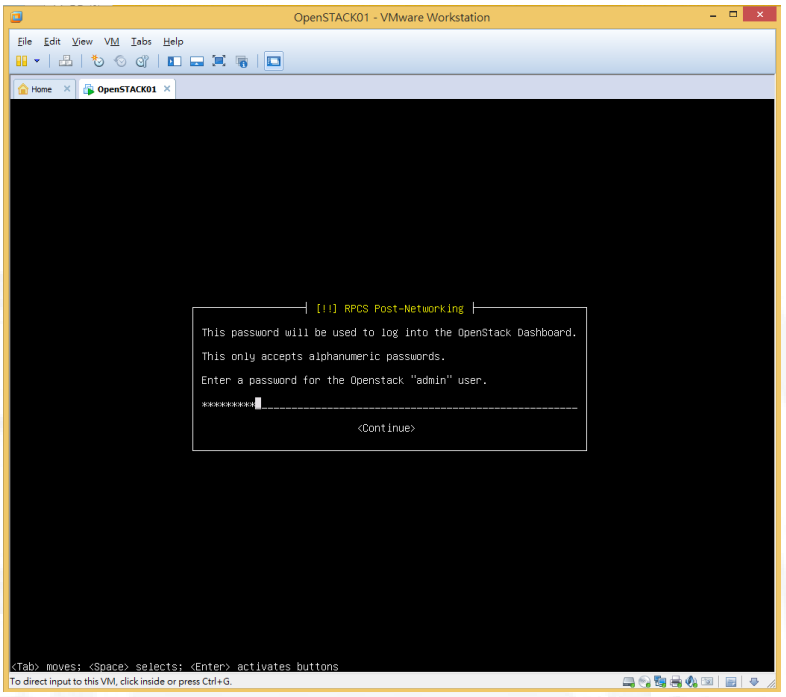

FIGURE A.20: Enter a password for the Openstack "admin"

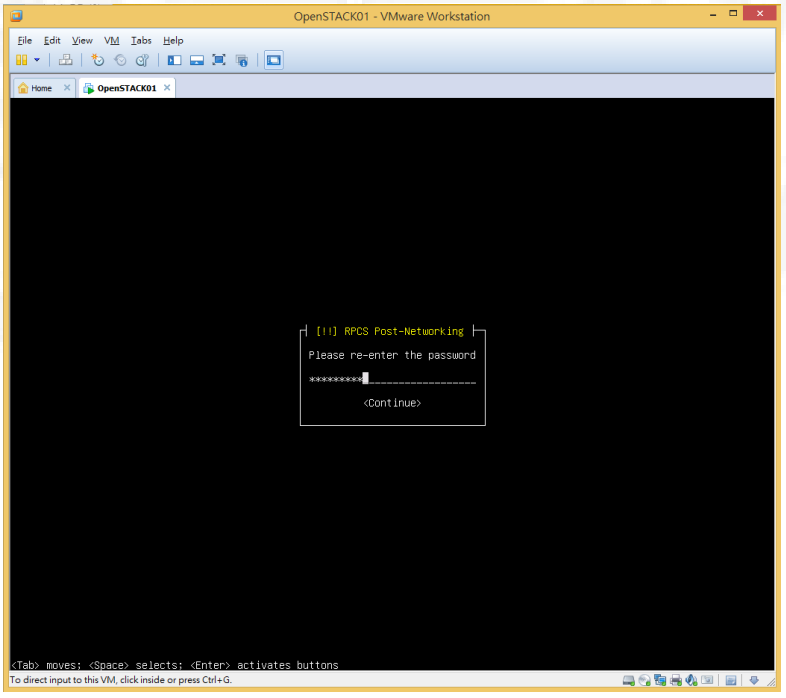

FIGURE A.21: Re-enter a password for the Openstack "admin"

12. Enter a username for a normal Openstack user.

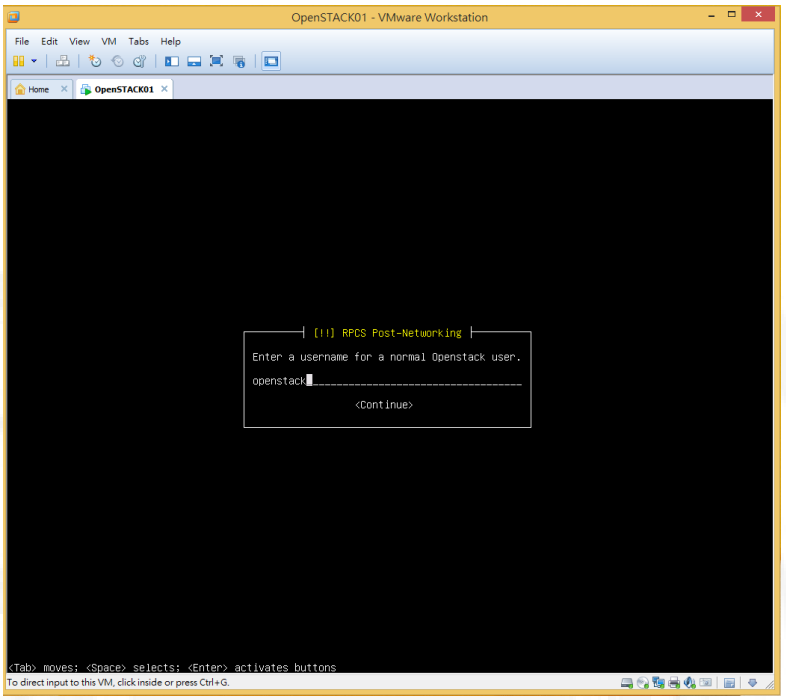

FIGURE A.22: Enter a username for a normal Openstack user

13. Enter a password for a normal Openstack user: openstack

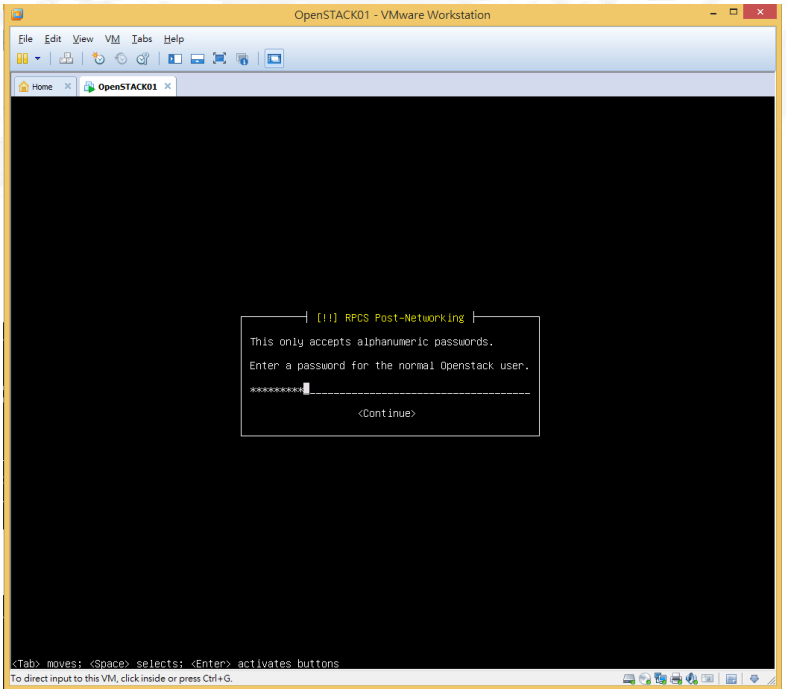

FIGURE A.23: Enter a password for a normal Openstack user

14. Re-enter the password.

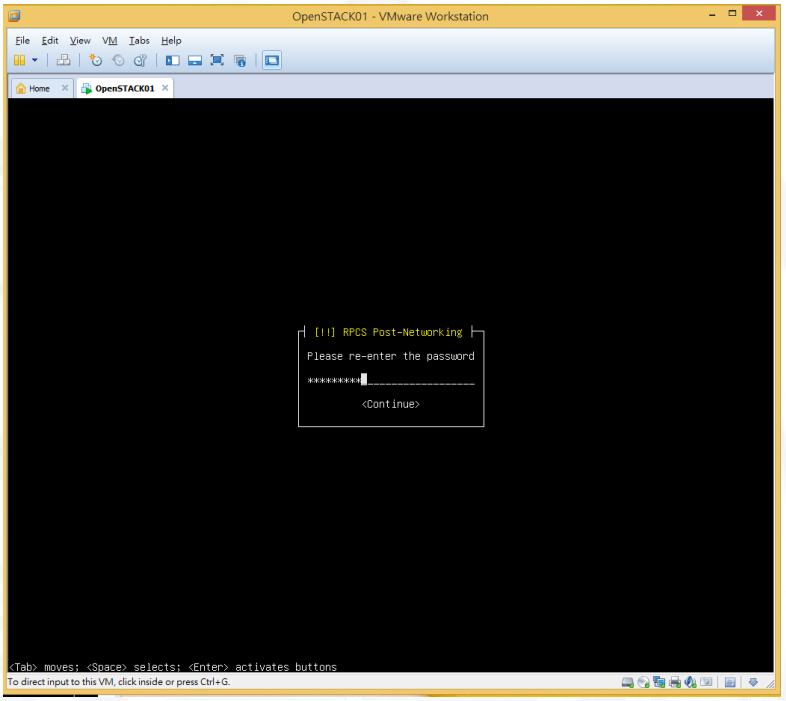

FIGURE A.24: Re-enter the password

15. Enter full name of the new user.

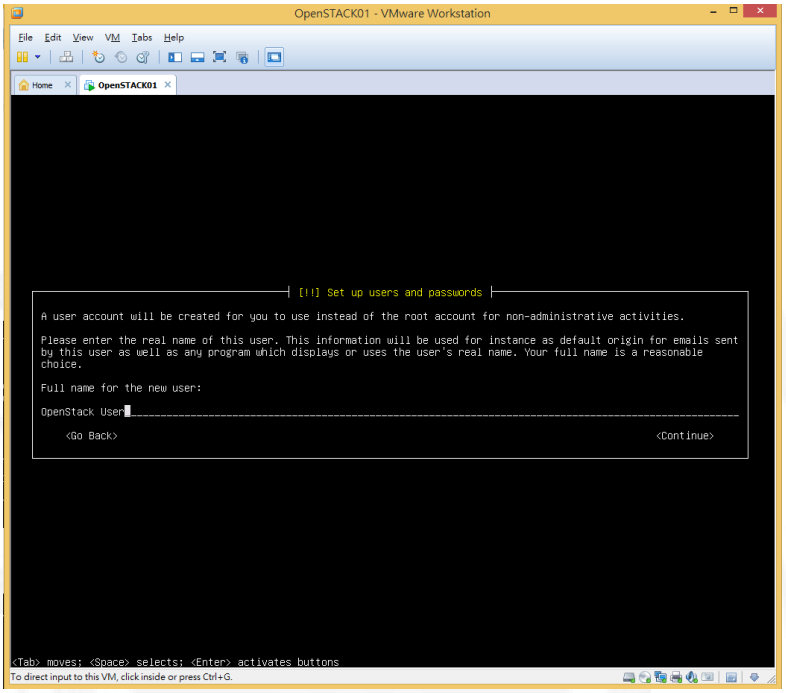

FIGURE A.25: Enter full name of the new user

16. Enter a account.

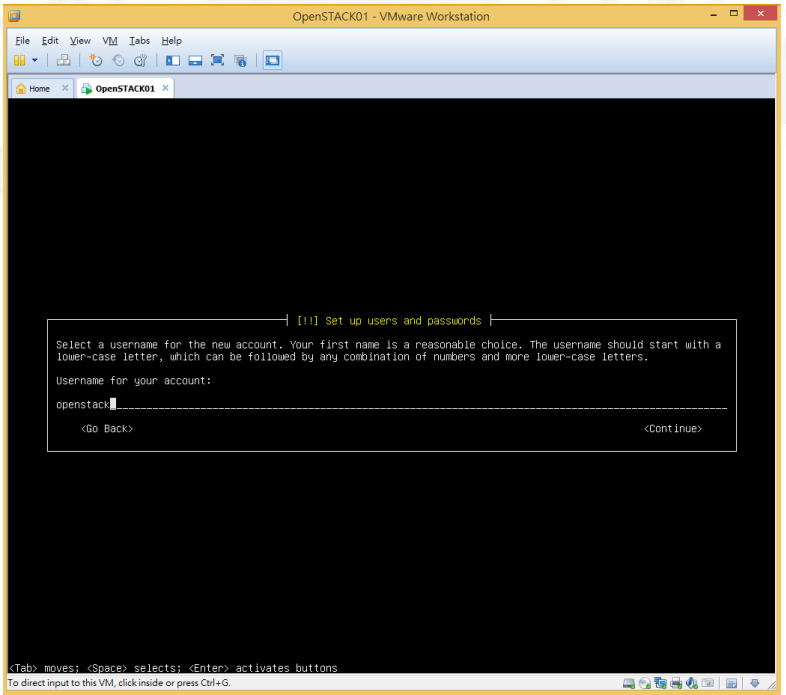

FIGURE A.26: Enter a account

17. Enter a password for the account: openstack

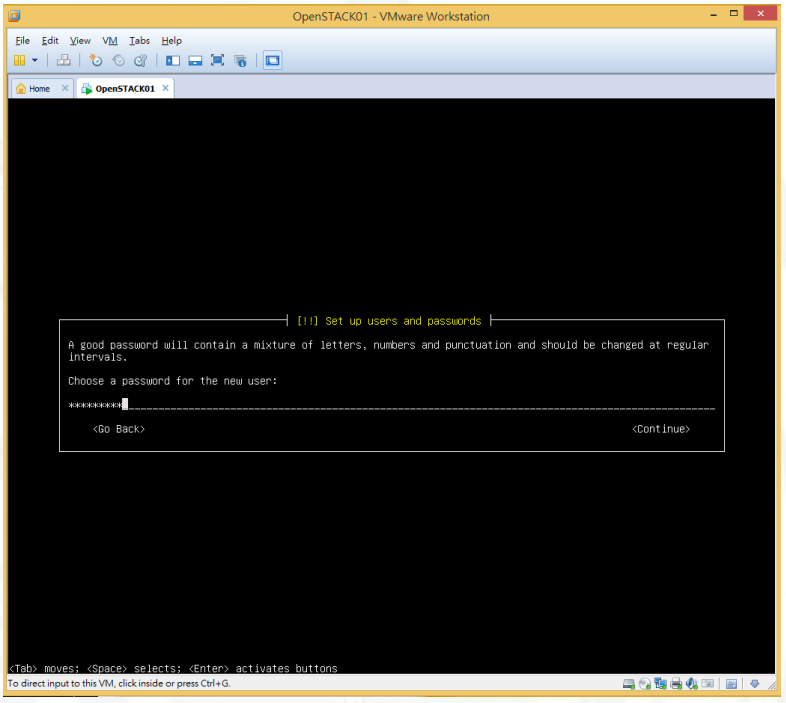

FIGURE A.27: Enter a password for the account

18. Re-enter password to verify.

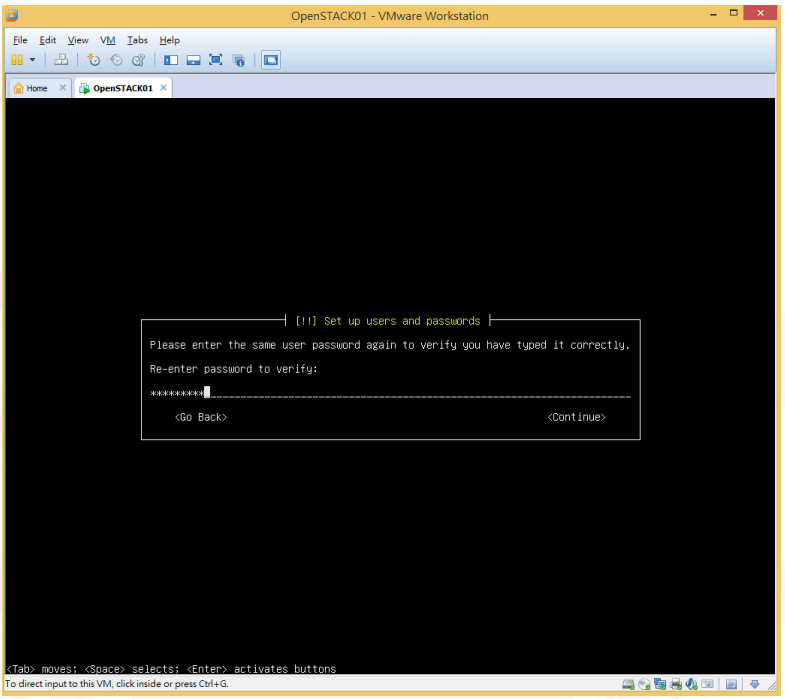

FIGURE A.28: Re-enter password to verify

19. 開始 Installing the base system，稍後片刻...

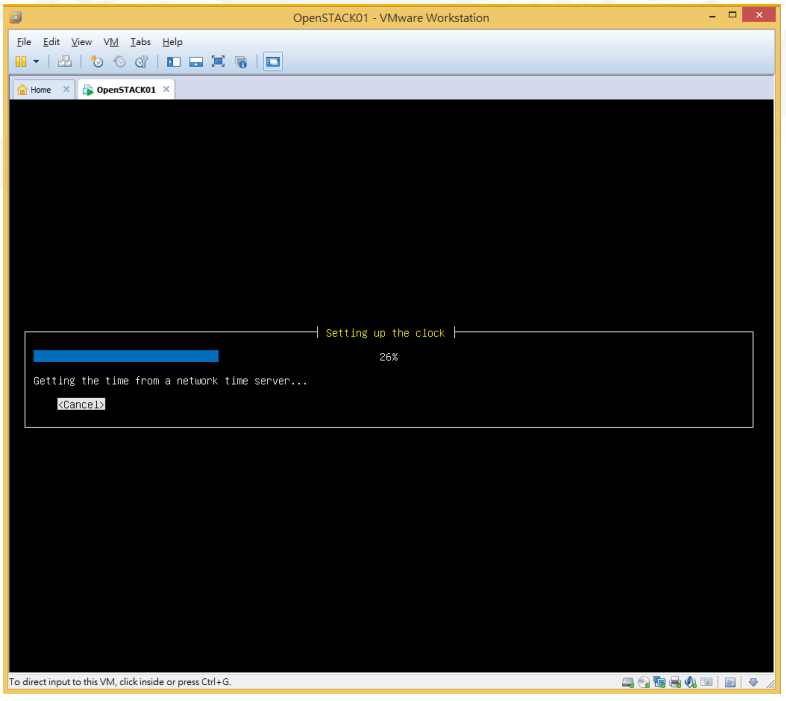

FIGURE A.29: Installing the base system

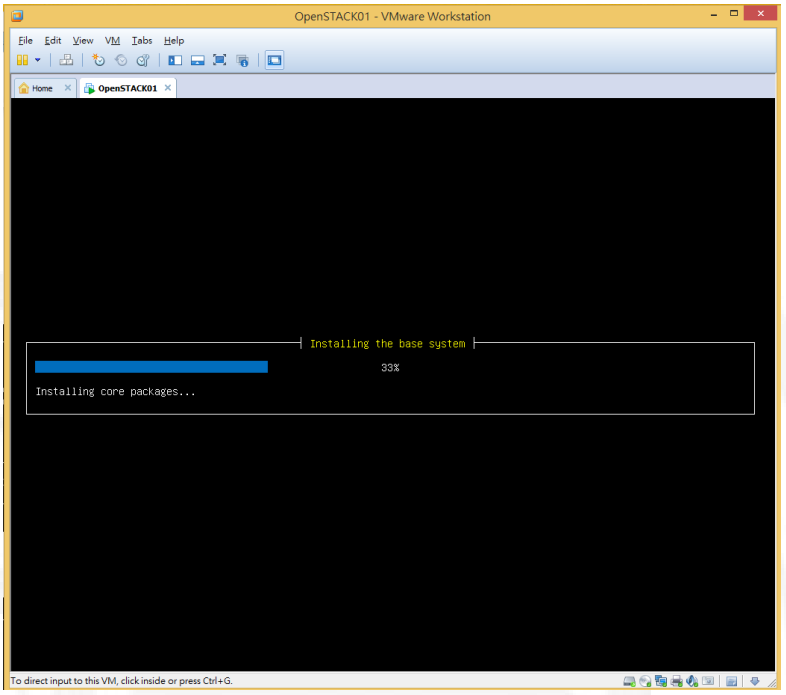

FIGURE A.30: Rackspace Private Cloud installation

20. Enter HTTP proxy information，預設為空.

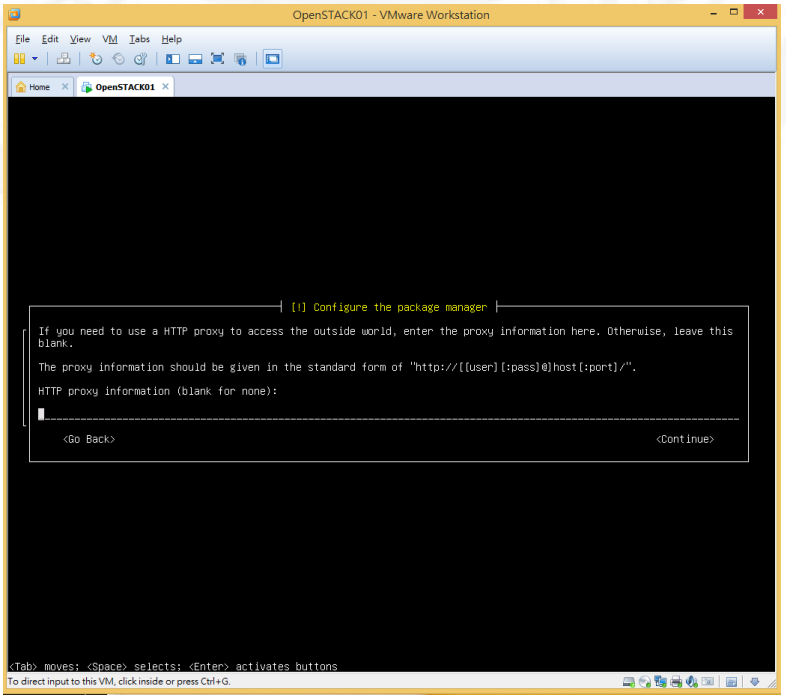

FIGURE A.31: Enter HTTP proxy information

21. 繼續 Configuring 設定和安裝，稍後片刻...

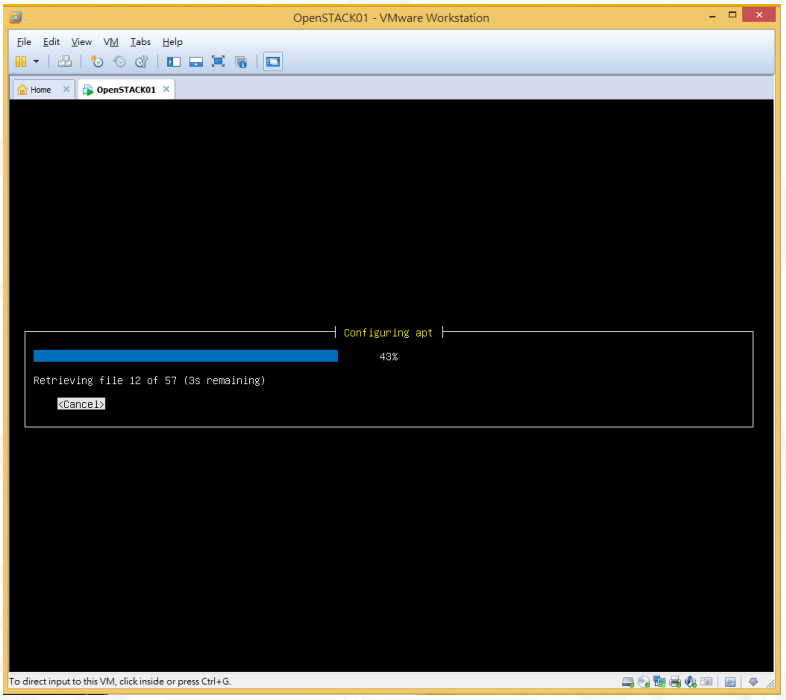

FIGURE A.32: 繼續 Configuring 設定和安裝

22. 即將完成安裝. (當安裝完成會自動重新開機)

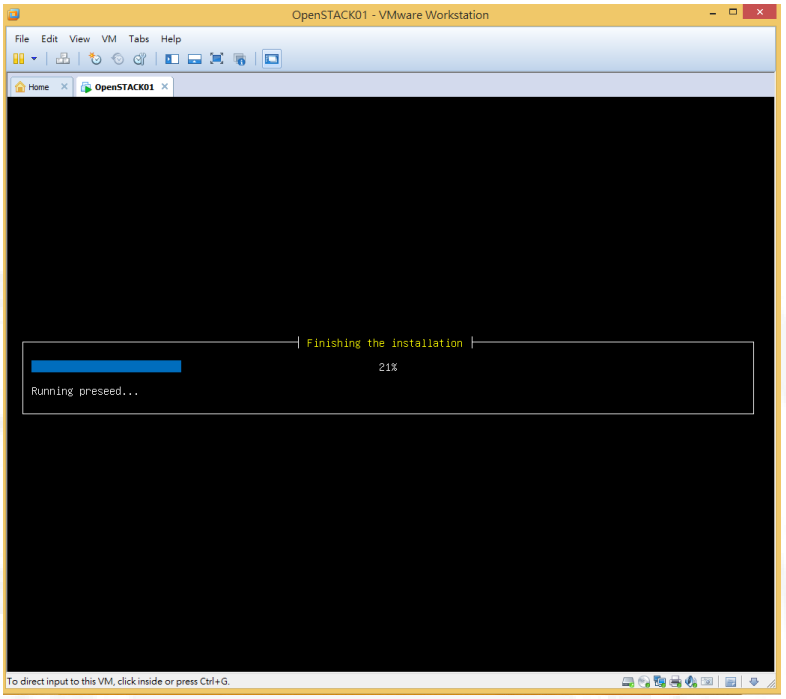

FIGURE A.33: 即將完成安裝

23. 重新開機後，繼續自動設定和安裝.

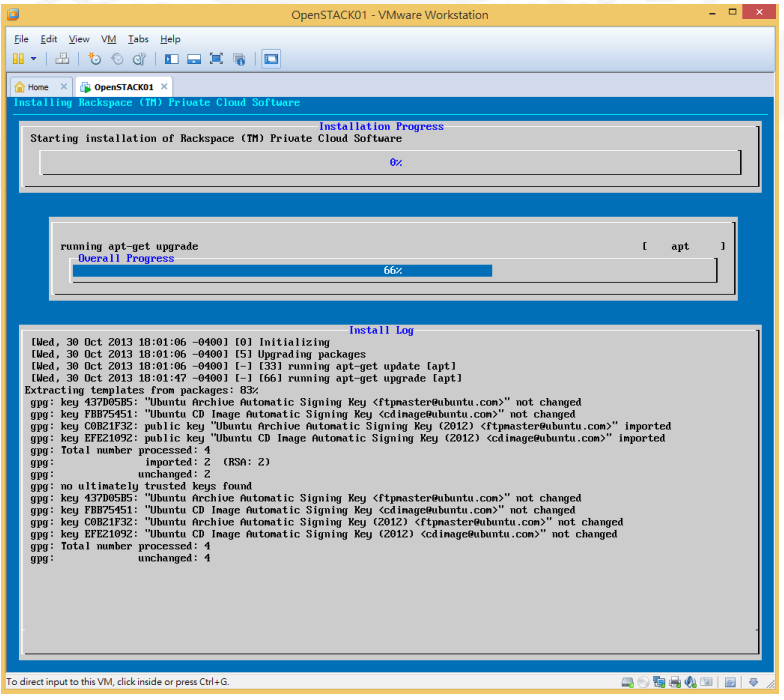

FIGURE A.34: 重新開機後，繼續自動設定和安裝

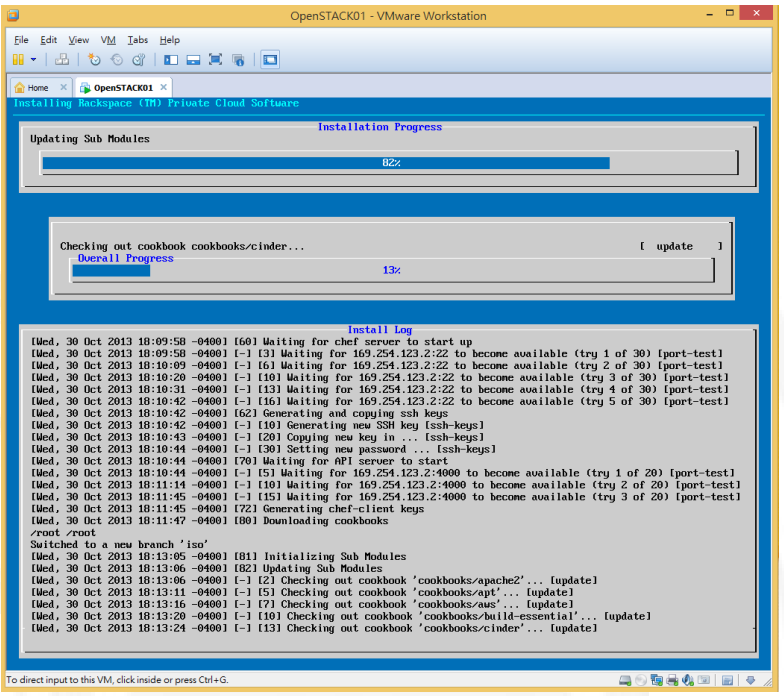

FIGURE A.35: Rackspace Private Cloud installation

24. 安裝完成，顯示目前的狀態。

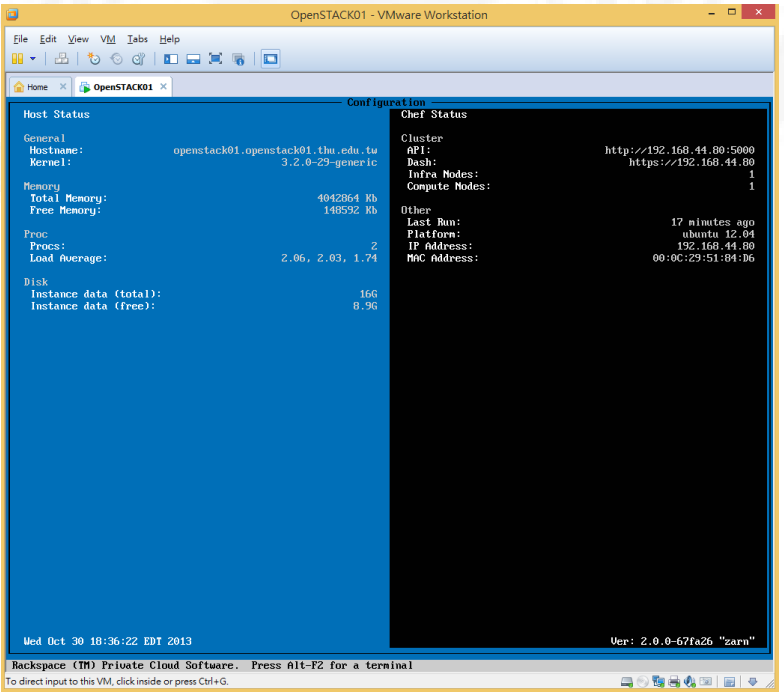

FIGURE A.36: 安裝完成

25. 同時按下 Alt-F2 鍵，進入 terminal on Ubuntu OS，表示大功告成。

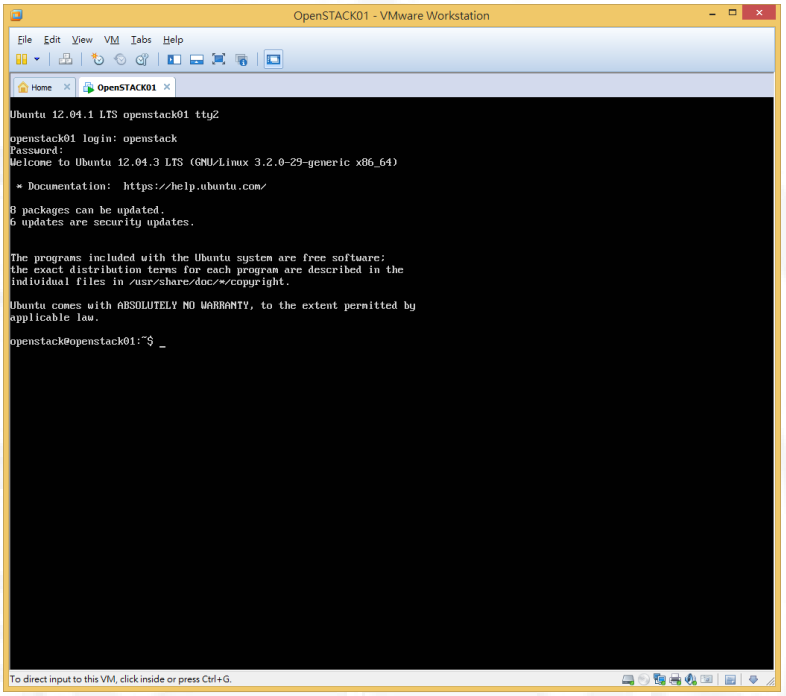

FIGURE A.37: 進入 terminal on Ubuntu OS

五、OpenSTACK web user interface

1. 打開瀏覽器 (如:IE)，輸入 https://192.168.44.80，即顯示如下 Rackspace Private Cloud Software powerd by OpenSTACK 畫面。若出現安全憑證問題，可先略過進入即可。

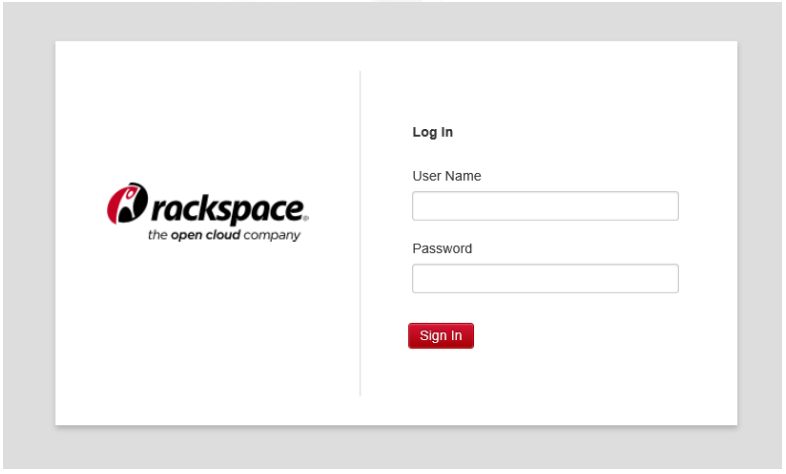

FIGURE A.38: Web UI

2. 請登入帳號: openstack，密碼: openstack

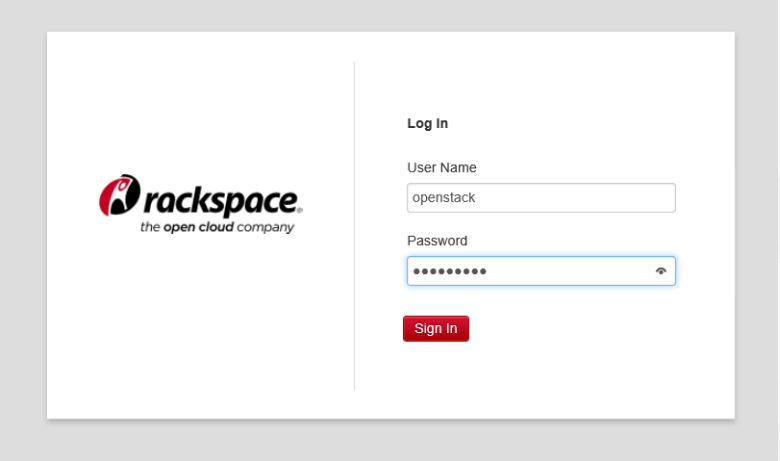

FIGURE A.39: Login Web UI

3. 成功進入 Rackspace Private Cloud Software powerd by OpenSTACK

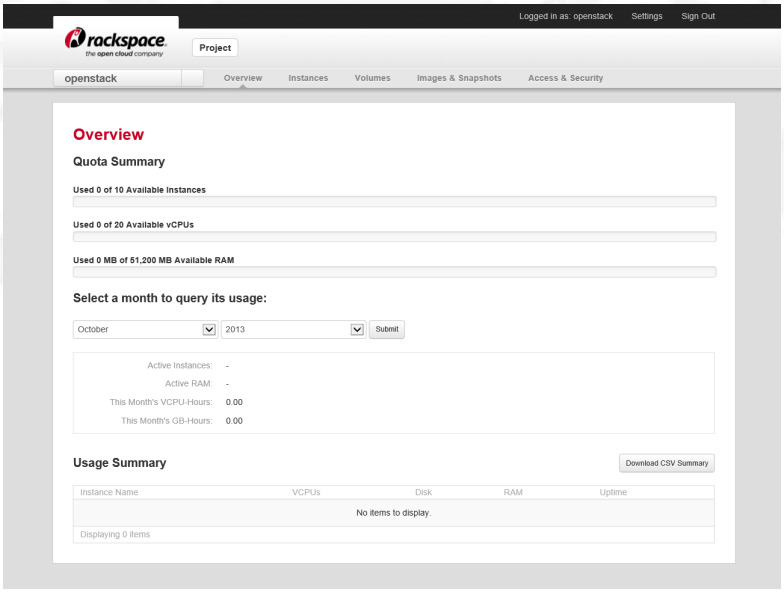

FIGURE A.40: 成功進入 Rackspace Private Cloud Software

附錄 B

Setting Up Network RAID1 With DRBD On Ubuntu

一、建置所需條件

• 二台以上的伺服器 • 每台伺服器中都要再多一個分割區給 DRBD 同步資料。

二、建置規格

• Intel® Core™ i7 • Memory：8G • HDD：1TB • OpenStack • Server1 and Server2 • OS：Ubuntu 12.04 • Memory：2G • HDD：20G+10G

三、建置前的準備

• 將兩台虛擬機網路建好 • DRBD 運作時只有一台為 Primary 並且這台才有權將 DRBD 的磁區掛載，其餘機器只能同步。

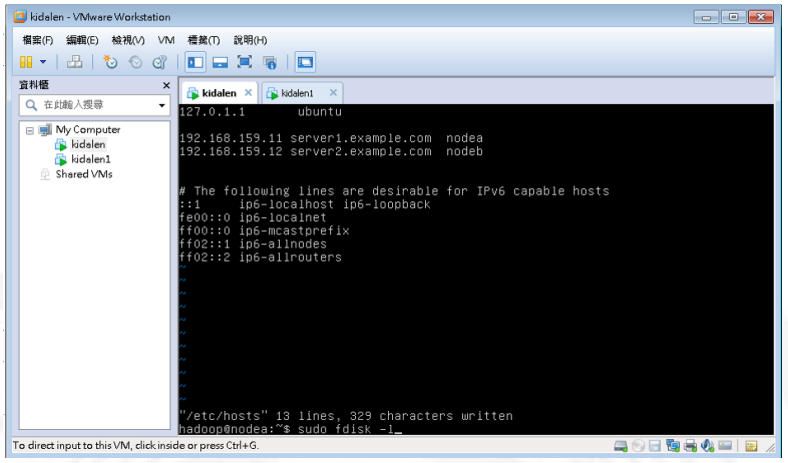

FIGURE B.1: Network setup

- Load the DRBD kernel module: # modprobe drbd
- To check if it is loaded, run: #lsmod | grep drbd

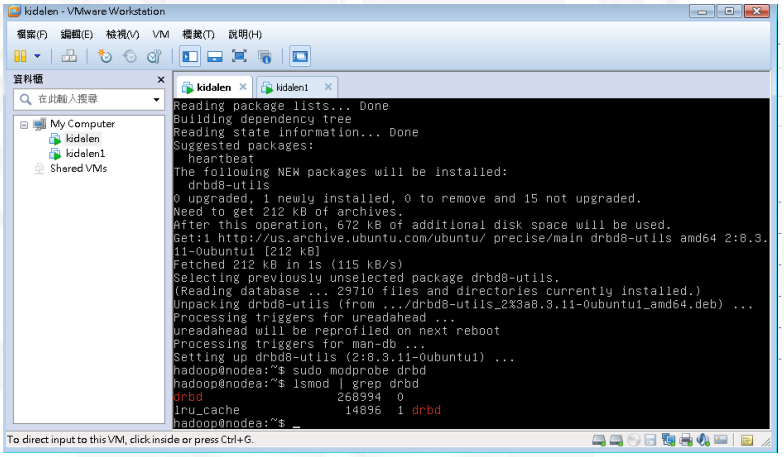

FIGURE B.2: DRDB Installtion

- Backup the original /etc/drbd.conf file and create a new one on both nodes

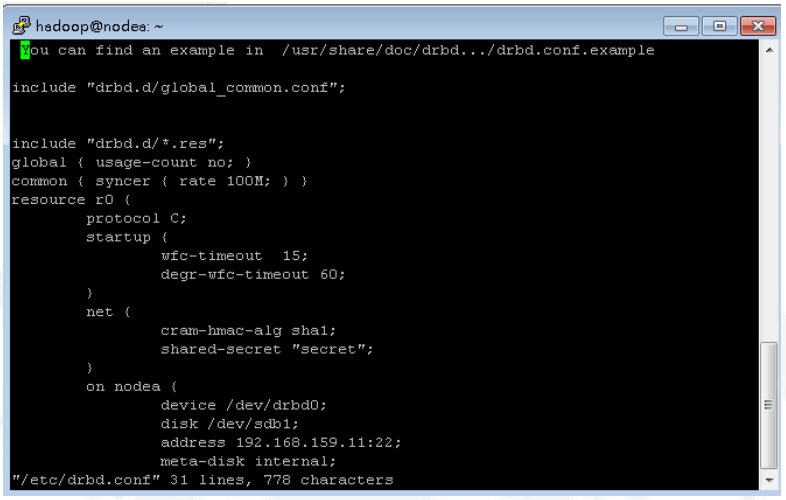

FIGURE B.3: Edit drbd.conf

- The next step has to be carried out on server1 only. Now make server1 the primary node

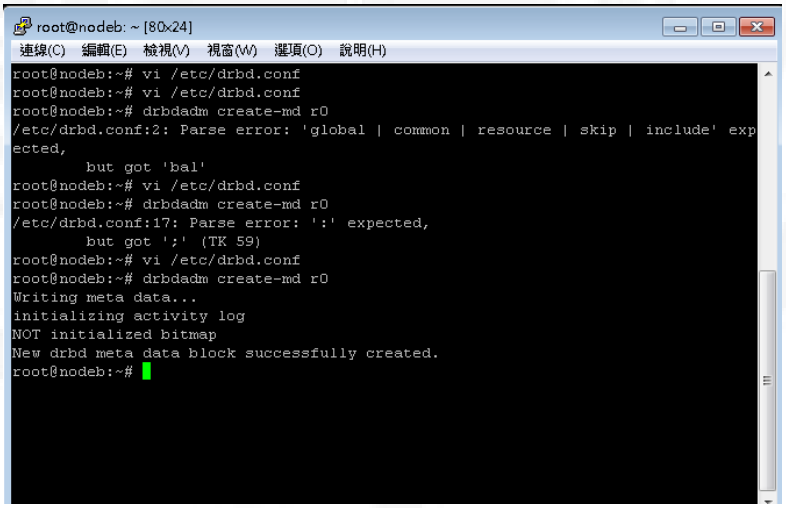

FIGURE B.4: Testing DRDB Installtion