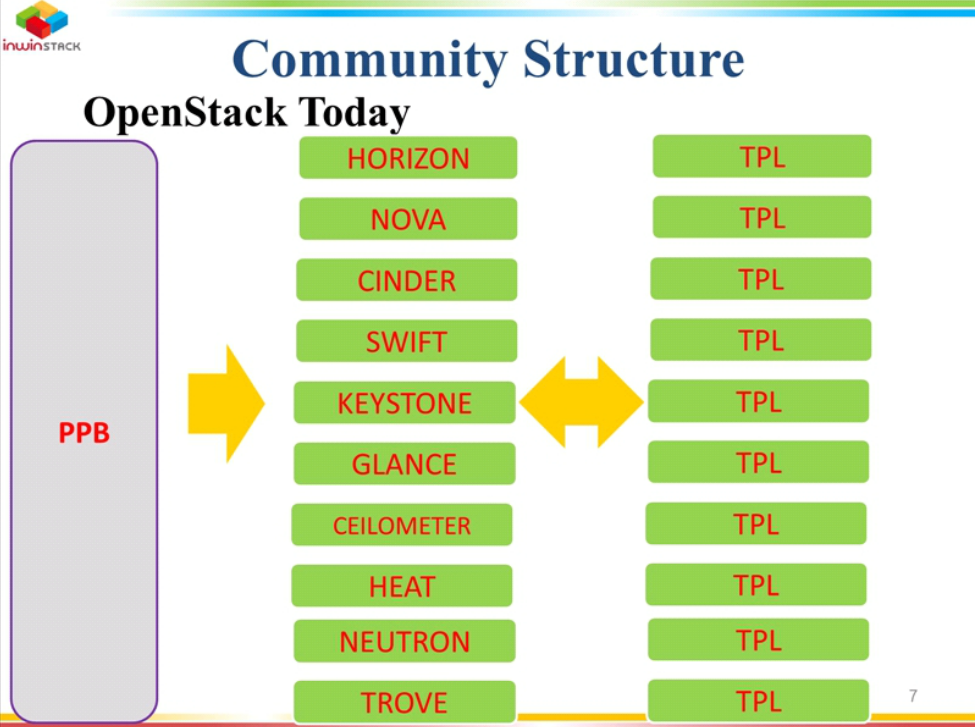
**OpenStack 學習**

第一章：OpenSatck基本介紹

1. 2012年獨立出來

2. RackSpance, Nasa, HP Support OpenStack

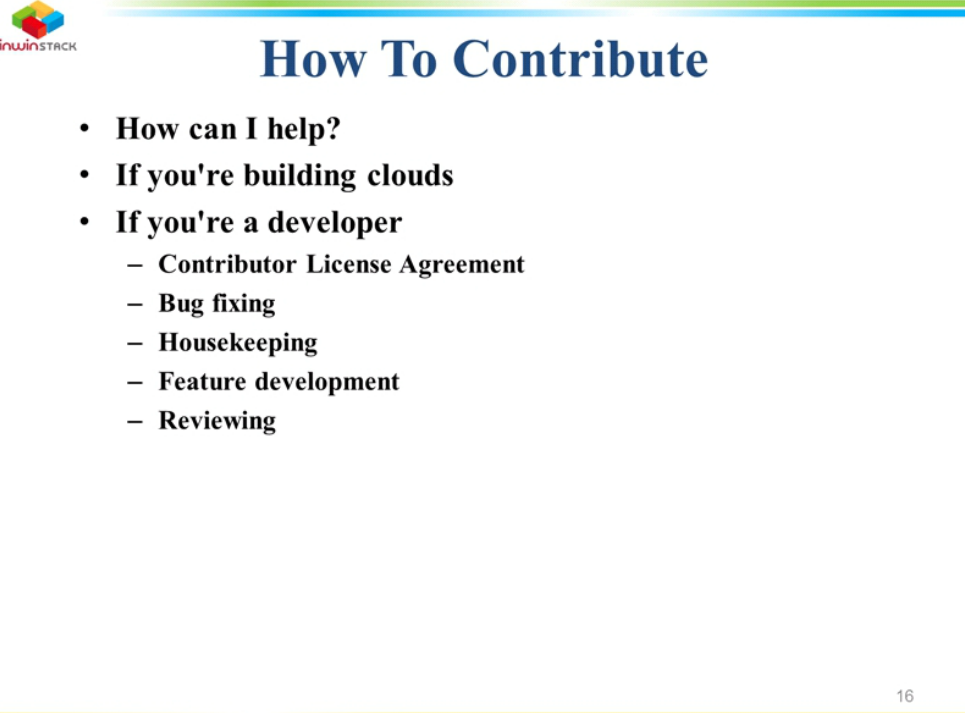
3. OpenStack version



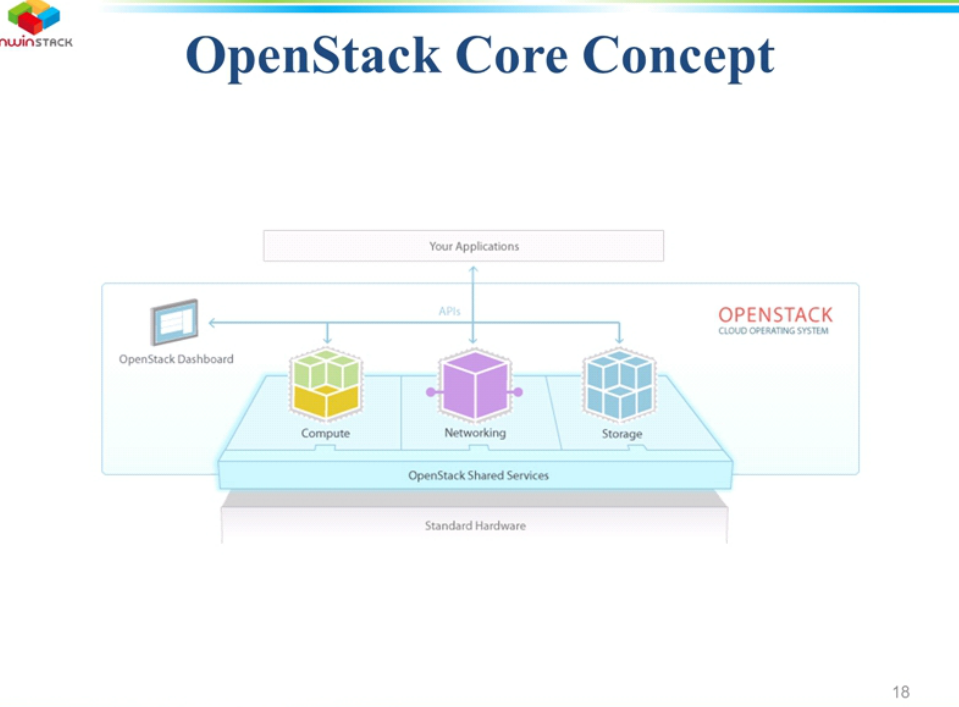
4. Governance

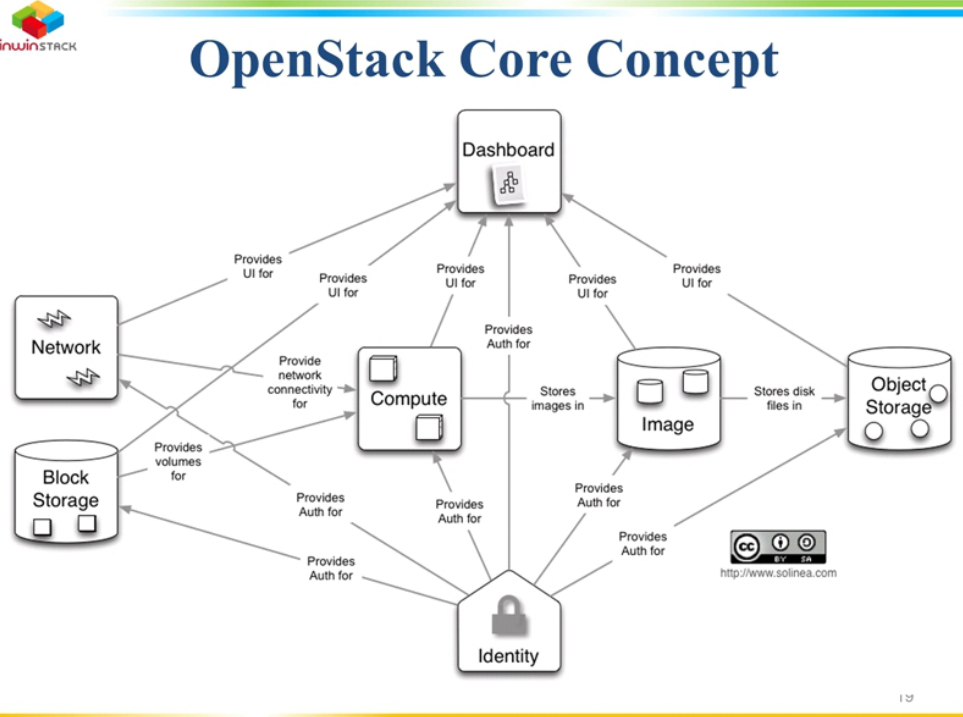


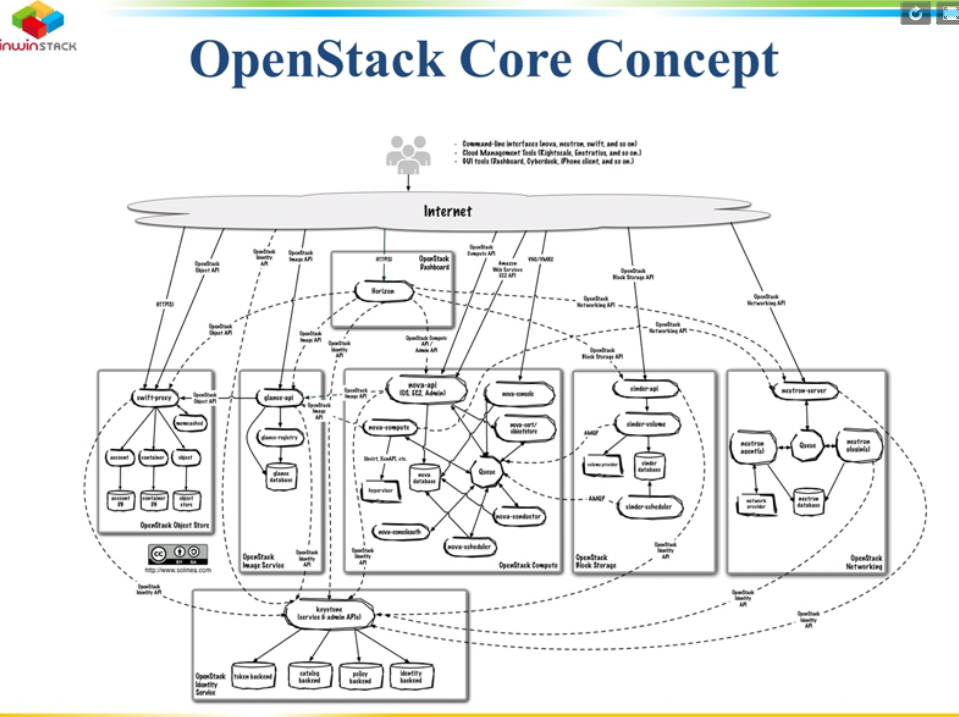
5. How to contribute



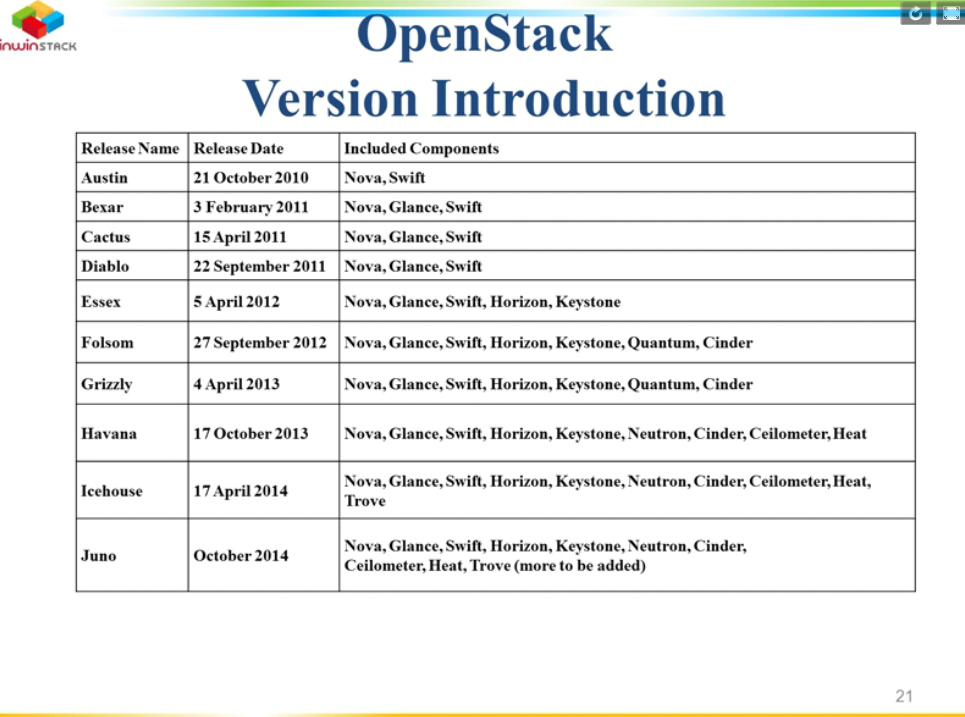
6. Three core API





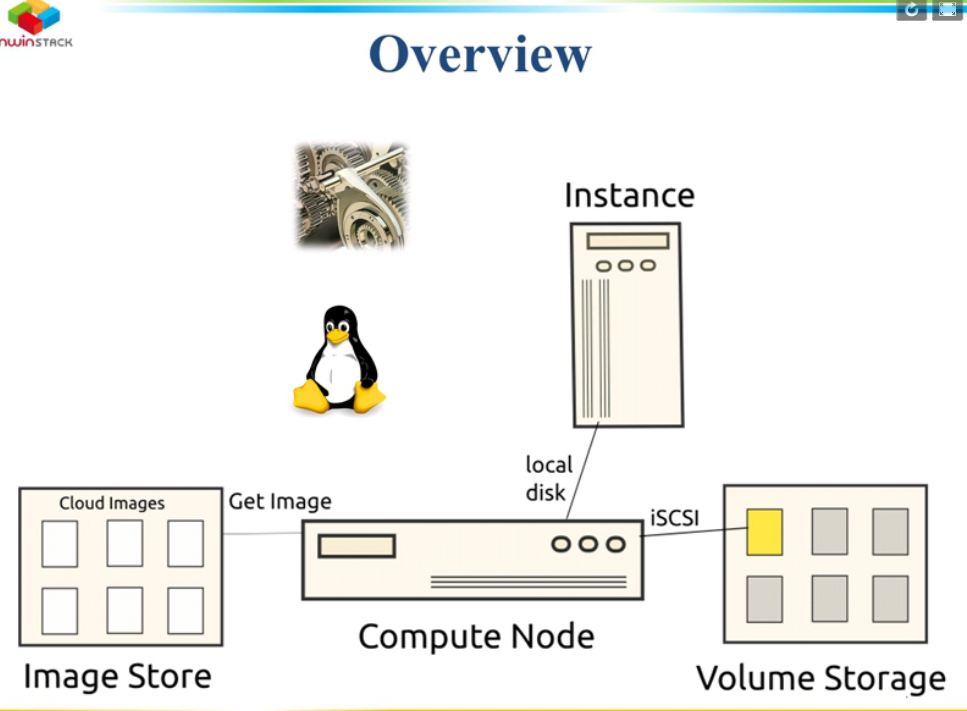


7. Version

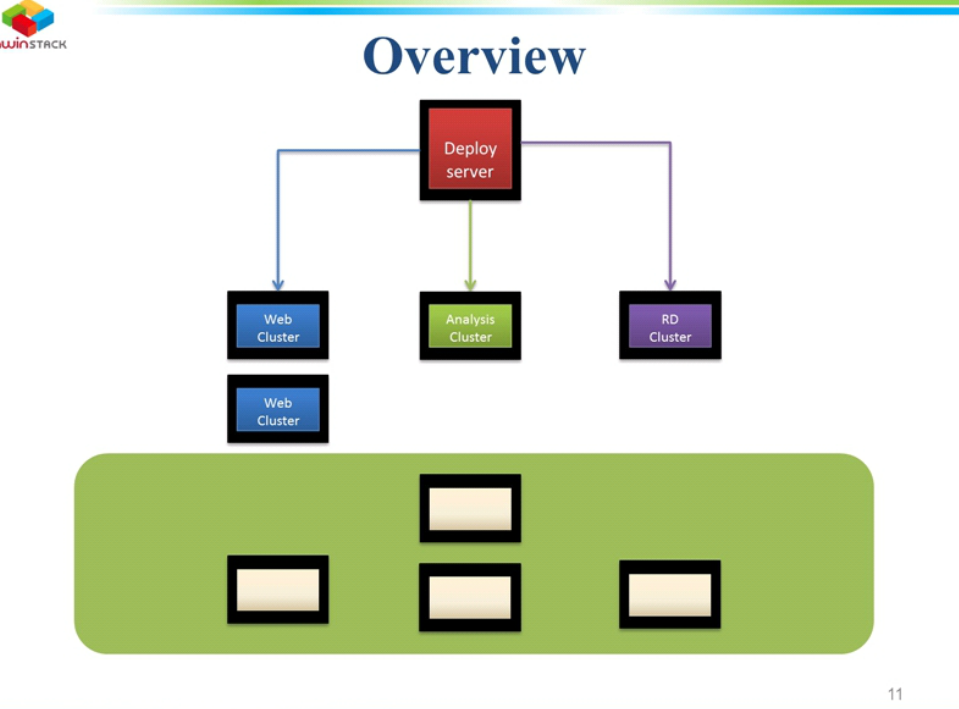


第二章 Controller

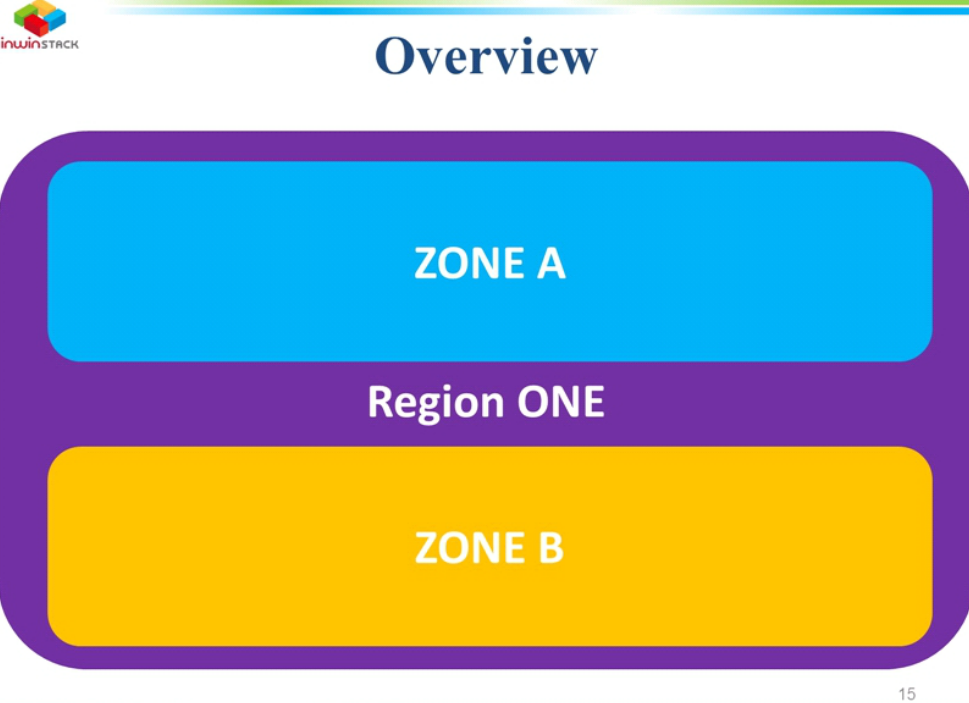
1. Overview, 先選擇要用的OS, 再架構基礎網路



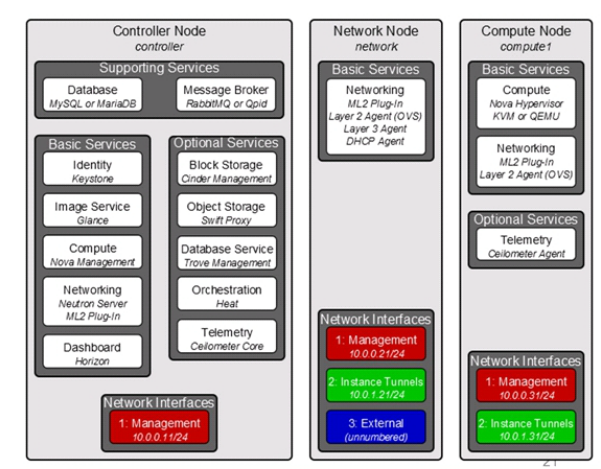
2. 在OpenStack內可以自由調整Node加入到所需的Server,



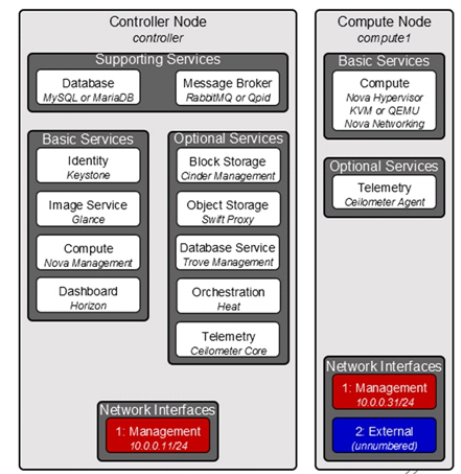
3. 用Openstack 在主機切出Region 供不同需求使用, 也可部署Zone A, Zone B(再切出其他虛擬機)



Openstack 提供三種Node



如果Neutron太難用, 可用Nona Network去架構簡單的基礎網路



如何操控Openstack??

1. Command Line

2. Dashboard

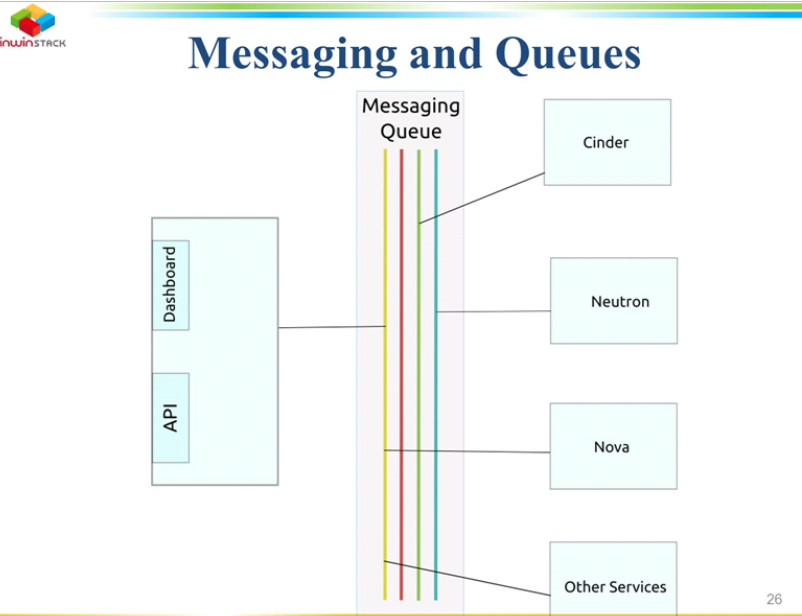
3. Script

4. APIs

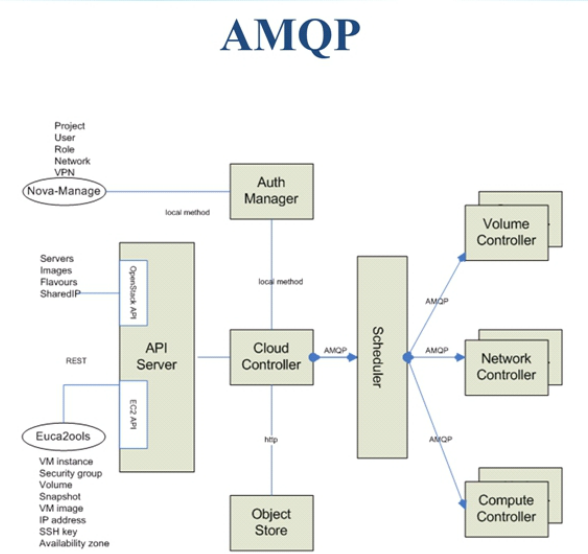
5. SDKs

6. Call Someone

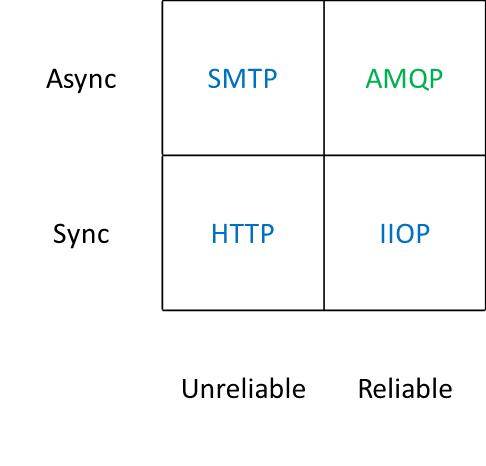
OpenSatck可以把很多service拆開, 要怎麼執行這些Service？使用Message ad Quene



提到的技術都被歸類為AMQP(Advanced Message Queuing Protocol) , 用Message Quene方式來解決

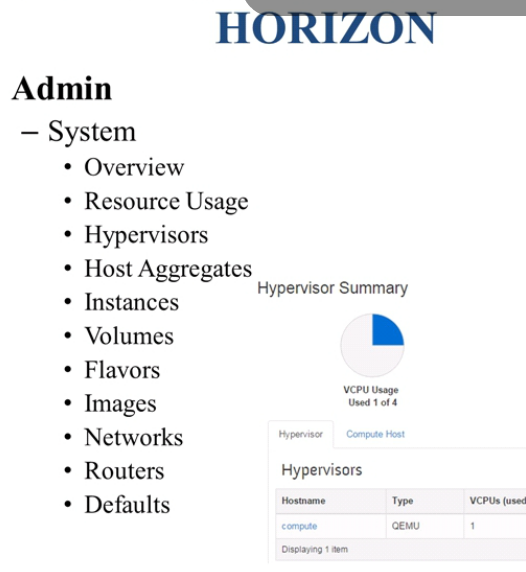
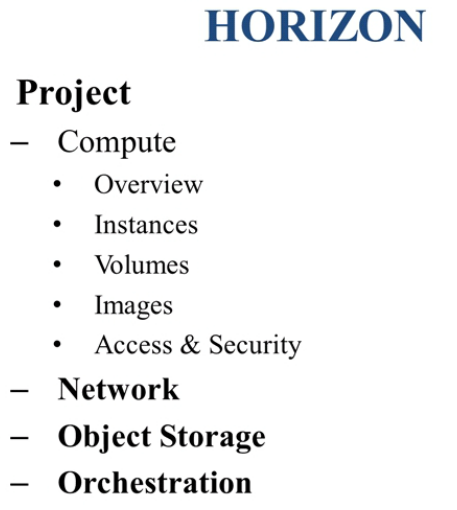


\*\*什麼時候會需要用到Message Queue，就如同下圖所示，如果你今天需要一個Reliable (保證傳達) 且 Async (允許非同步) 的訊息傳達和交換方式，這時候你就需要Message Queue，而首選就是AMQP。此外對於Cloud 和SOA 來說 AMQP也是佔了很重要的角色。(AMQP最主要就是多了Exchange的機制，除了 store-and-forward、publish/subscribe外還多了許多的Routing 方法

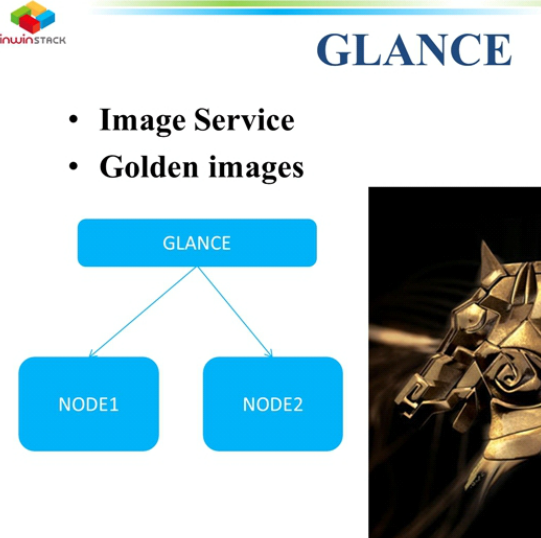


1. Horizon (Dashboard), Openstack以Python(Django)所寫, 架在Apache上用RESTful溝通

(登入後可使用的東西)



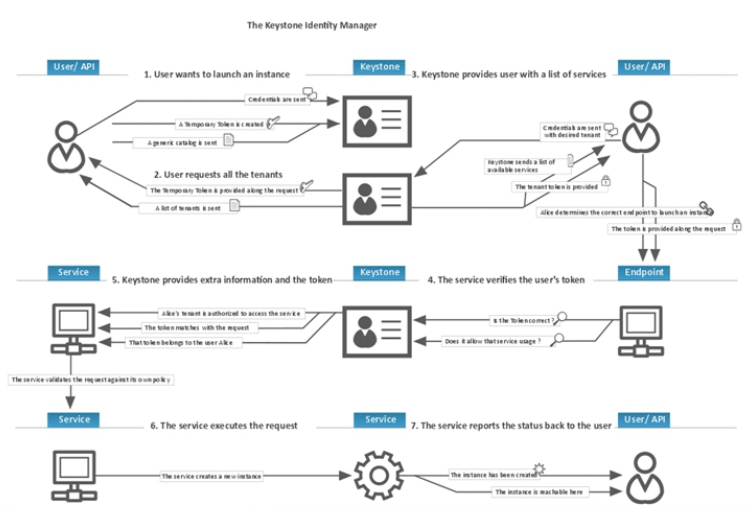
2. Glance (Images), Golden Image(直接複製一份) ->會佔用硬碟空間



3. Keystone 身分驗證機制(做任何事情都要驗證)

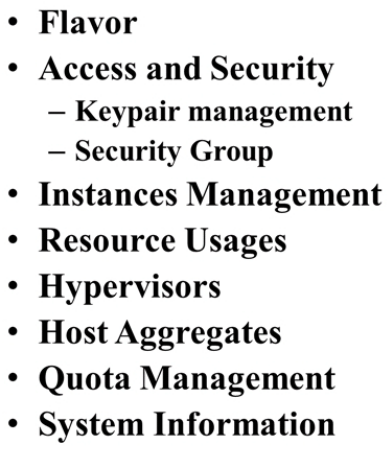


(帳號密碼輸入之後, 會產生Token, 用Token來認正



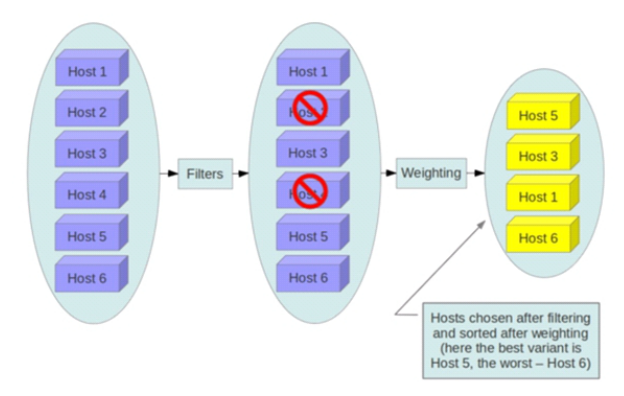
4. Nova (硬體)

\*\*啟動VM不能連線?通常是 Security Group沒有設定好 ->被允許link 的 Port!

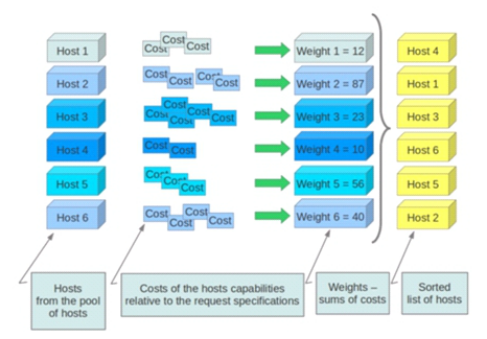


建立VM要怎麼建立? 會透過Filter和weight機制來進行

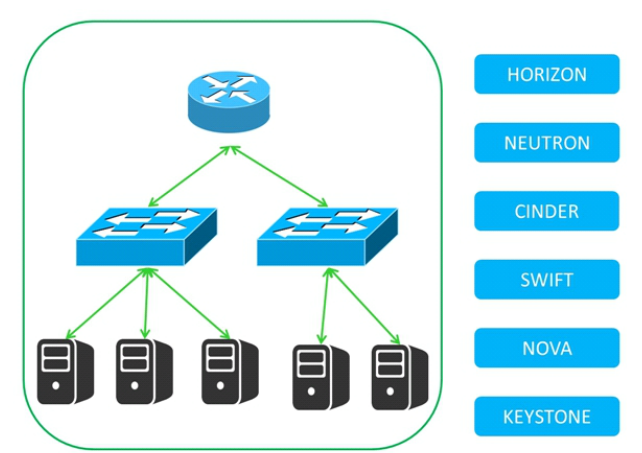
-> Filter會先過濾不行的Host ,接著Weight就是看裡面的資源進行權重比較



5. Weight



Basic Service (最基本 Dashboare, Nova, Keystone)



6. Cinder

(1) Block Storage Service

(2) NOVA-Volume

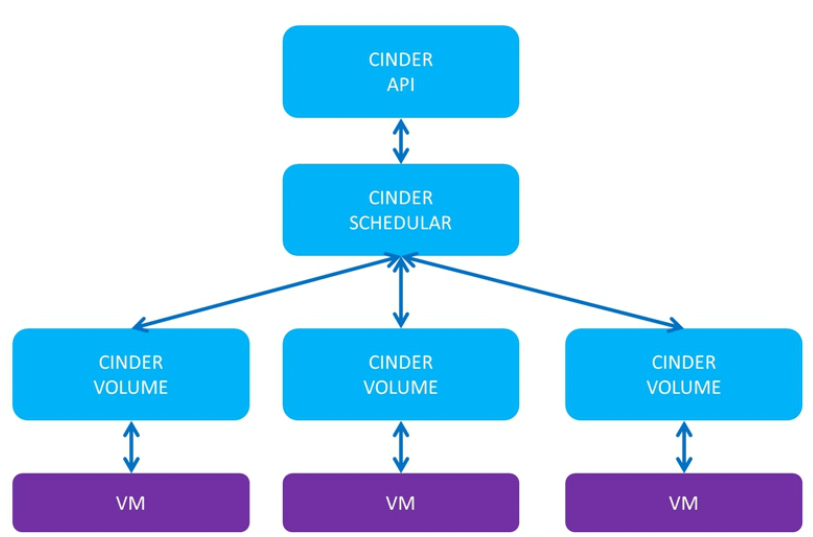
(3) LVM

(4) Drivers

(5) Support List

(6) Backup Strategy

呼叫API, 透过Shcedular来管理Cinder Volume

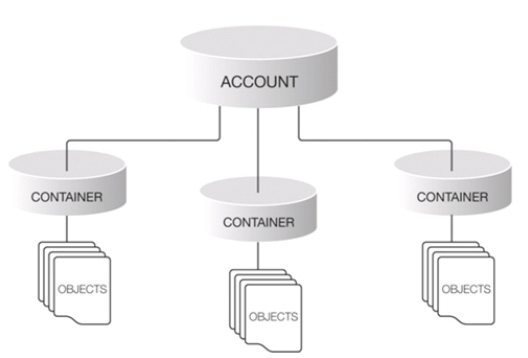


7.Swift

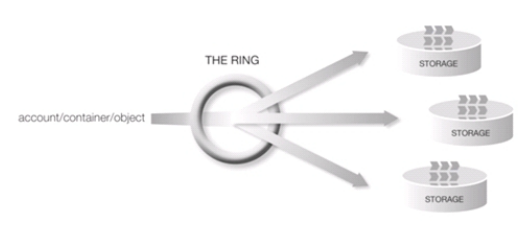
(1) Object Storage

(2) Backup Solution

資料上傳之後, 會分散儲存在不同的container



透過 “Ring”機制方式來分散



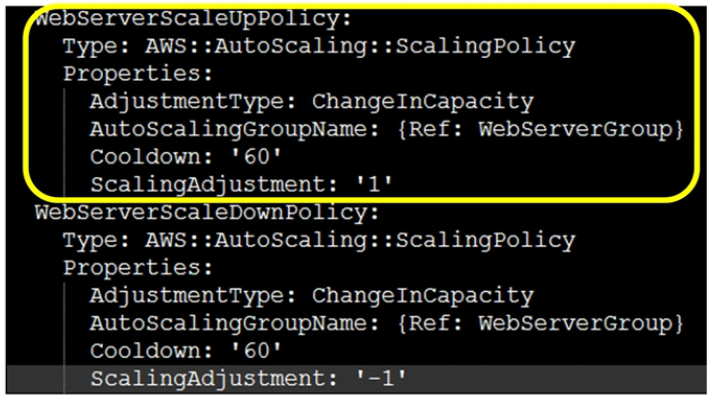
8. orchestration (heat) Auto-Scaling

(1) Template-Based

(2) AWS CloudFormation

(3)Hot and Yaml

設定Policy, 達到條件之後就會AutoScale~



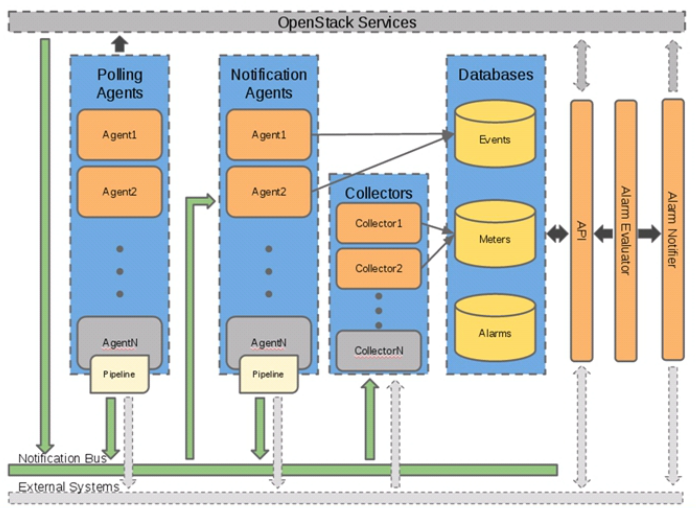
9. Ceilometer (計量的動作)

(1) Monitoring Notification

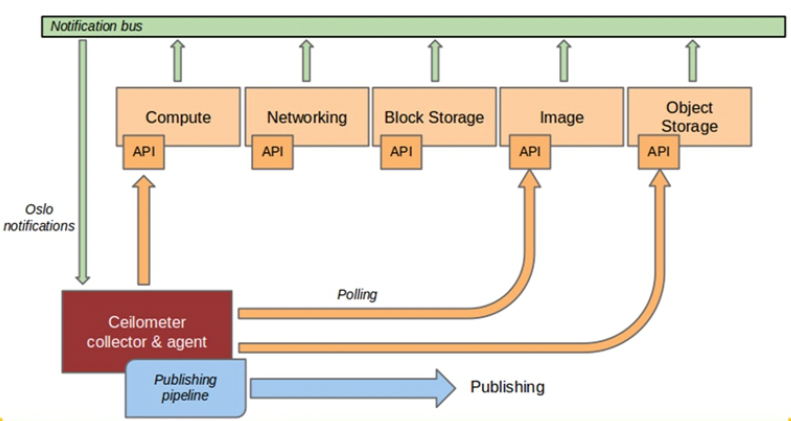
(2)Metering

(3)Database

運作的基礎架構



運作的原理(跟所有components要資料, 收集後放進資料庫, 再製作monitor部份)



10. Trove

(1) Trove- API, Receive job from AMQP

(快速建立一個資料庫環境, 需建立一個單一管理介面(不同的有不同介面))

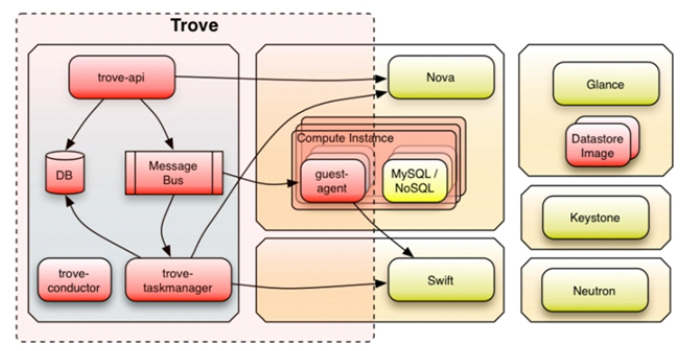
Trove-taskmanager – Provisioning, Managing, Performing operations

Trov-guestagent guest Agent

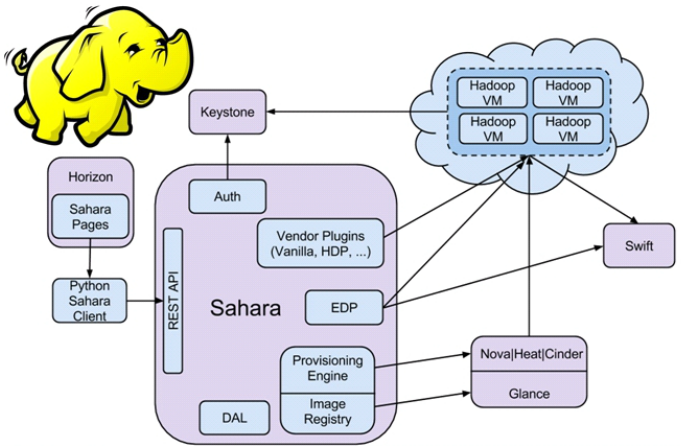
(2) 點幾個鍵就可以將Database Backup 到Swift上

(3) Self-service provisioning, Rapid provisioning, Single management interface, many database tech,

Single API, many database tech, Automated back-up and recovery, Built on OpenStack for Openstack



11. Sahara (Big data 快速環境部屬) – 也提供很多plugin的應用



**第三章 Compute**

|  |  |
| --- | --- |
| 解釋 | Nova : Nova-scheduler, VM placement, VM provisioning in-depth  Block Storage: Ephemeral Storage, Volume Storage, Location, Scaling |
| PPT |  |
| 解釋 | Nova 核心概念 |
| PPT |  |
| 解釋 | Nova-Scheduler, 根據條件尋找適當的主機  條件: (1)CPU (2) RAM (3)Disk  方式：(1) Filterring (2) Weights  先做第一次Filter掉後,  用Weight來計算實體主機的資源計算, 會把個主機打分數, 最高的就部數VM |
| PPT |  |
| 解釋 | VM Provisioning in-depth , 用Dashboard進行操作 |
| PPT |  |
| 解釋 | 儲存方式: Ephemetal Storage (instance 創出內部Disk)  並記得要做Raid的保護機制 |
| PPT |  |
| 解釋 | Volume Storage: 透過UI執行, 在特定位置建立Volume (透過Volume 指令建立, 以iSCSI的形式來使用), 可以外掛(mount), snapshot |
| PPT |  |
| 解釋 | Scaling: X86 hardware(Less price, ease configure) |
| PPT |  |

**第四章 Network**

|  |  |
| --- | --- |
| 解釋 | 1.Netwoking in OpenStack  - Four Networks  - Network  - Subnet  - Port (跟以前不太一樣)  - Plugin Architecture  2. Legacy Networking  3. OpenStack Networking  - Local  - Flat-Network  - VLAN  - GRE/VXLAN  - Data Flows  - DVR  - L3 Agent HA(VRRP) |
| PPT |  |
| 解釋 | Four Networks (有安全, performance考量, 建議用這個架構)  1. External Network – 提供給VM對外做存取使用  2.Data Network – 給VM 之間資料溝通  3. Management – 給各component之間用來溝通的網段  4. API network – 透過API做管理的動作 |
| PPT |  |
| 解釋 | (1)Network – L2, 可以去切Subnet網段,  (2)Port(以前像port號)-> 把VM的網卡當作Port  (3) Plugin Architecture (透過neutron plugin, 與SDN溝通, 不用一台台設定Switch) |
| PPT |  |
| 解釋 | Legacy Networking (Nova Network 都透過 Bridge把東西往外島)  \*\*到底Nova Network 和 Neutron 最大差異?  -> Nova: 不會有SPO FO的問題, 直接讓VM連線到網路外面  -> Neutron: L3 Routing 都集中在 Neutron Node上 |
| PPT |  |
| 解釋 | Nova Network 運作方式  (想要更複雜的功能就用Neutron) |
| PPT |  |
| 解釋 | Basic Node Deployment  (只是try的話, 一張網卡就可以符合需求) |
| PPT |  |
| 解釋 | Performance Node Deployment |
| PPT |  |
| 解釋 | 接下來將上面的圖拆開來就解釋的動作  - Controller HA 機制 (架兩台controller做串接) |
| PPT |  |
| 解釋 | Compute Node (VM 要對外, 透過Nova-compute, Neutron-open…做連接) |
| PPT |  |
| 解釋 | Network Node (HA機制, 使用PC maker機制來做資料同步) |
| PPT |  |
| 解釋 | Storage Node (準備什麼硬體? 用哪種儲存方式? (分散式) ) |
| PPT |  |
| 解釋 | 簡單的POC環境 (Local方式) |
| PPT |  |
| 解釋 | 用Flat Network方式進行POC |
| PPT |  |
| 解釋 | 可以切很多Vlan 跟subnet達到自己的需求 |
| PPT |  |
| 解釋 | GRE/VXLAN (同樣網段還是可以執行) |
| PPT |  |
| 解釋 | Network架構, 有Single, 混和,…甚至可以加一個Router |
| PPT |  |
| 解釋 | 更複雜, By tenant 又串router |
| PPT |  |
| 解釋 | Horizon 上 Network Topology的顯示 (external network, 切了subnet, 用router串, 當然也可以串很多個(綠色)) |
| PPT |  |
| 解釋 | 對應關係如下圖, 用外部IP連到內部instance(將IP bindle到VM), 用associate IP給VM (外面即可link), 藍色為外部IP ->要注意security設定! SSH link的話, 要把相關的Port打開, 不然無法連接進來 |
| PPT |  |
| 解釋 |  |
| PPT |  |
| 解釋 | Data Flow? 東西向: Server to Server的資料傳輸 對外:中間穿過L2 |
| PPT |  |
| 解釋 | 早期Neutron會有SPO Fla 問題, Neutron node 壞掉, 整個都無法對外  DVR ->分散式Routing , Handle ARP Table |
| PPT |  |
| 解釋 | 南北向用 L3HHA機制來解決問題, 假設L3 壞掉, 用kiplive來偵測, 壞掉轉到其他Neutron node (Connect check來檢查) |
| PPT |  |

**第五章 Storage**

|  |  |
| --- | --- |
| 解釋 | Commodity Storage Backend Technologies  (1) OpenStack Object Storage(Swift)  (2) Ceph  (3) Gluster  (4) LVM  (5) ZFS  CINDER  (1) Block Storage  (2) LVM  (3) Pool  SWIFT  (1) Architecture  (2) SDS  Shared Storage  (1) NFS  (2) SAN  (3) CEPH  - Why  - Success Case |
| PTT |  |
| 解釋 | SWIFT – Object Storage (分散儲存機制), Ceph, Gluster – Shared Storage  LVM, ZFS – Block Storage |
| PTT |  |
| 解釋 | CINDER, Block Storage(一個一個block 組成) – Image file or ISO file |
| PTT |  |
| 解釋 | CINDER API, CINDER Volume ->Pool  假設有instance(VM)要使用的時候, 會把它切出來, 做成iSCSI給VM使用 (Volume 或Disk應用) |
| PTT |  |
| 解釋 | CINDER  LVM = Logical Volume manager -> 多顆硬碟組成的(看怎麼切割再使用) |
| PTT |  |
| 解釋 | SWIFT Object Storage -> 不要把雞蛋放在同一個籃子,  假設用帳號登入, 把檔案上傳到Swift 的container, 透過container 幫忙儲存檔案, 並會把檔案儲在三個Container中 |
| PTT |  |
| 解釋 | SWIFT重要概念Ring -> 會對到Physical device ->透過此機制將檔案分散儲存在不同的地方 |
| PTT |  |
| 解釋 | SWIFT - > SDS -> Software defined storage  假設有不同的Storage, 管理上會有麻煩的問題, SDS可以用軟體的方式來管理, 中間疊了一個管理的功能(Unify Interface), 能整合不同廠商的Storage, 並且有UI看目前使用的情況 |
| PTT |  |
| 解釋 | SWIFT - >希望用五台以上的機器來使用Swift (安全HA的機制) |
| PTT |  |
| 解釋 | Shared Storage :  NFS(**N**etwork **F**ile **S**ystem) 一種使用於[分散式檔案系統](http://zh.wikipedia.org/wiki/%E5%88%86%E6%95%A3%E5%BC%8F%E6%AA%94%E6%A1%88%E7%B3%BB%E7%B5%B1)的協定，由[昇陽](http://zh.wikipedia.org/wiki/%E6%98%87%E9%99%BD)公司開發，於1984年向外公布。功能是透過[網路](http://zh.wikipedia.org/wiki/%E7%B6%B2%E8%B7%AF)讓不同的機器、不同的[作業系統](http://zh.wikipedia.org/wiki/%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1)能夠彼此分享個別的資料，讓[應用程式](http://zh.wikipedia.org/wiki/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)在客戶端通過[網路](http://zh.wikipedia.org/wiki/%E7%BD%91%E7%BB%9C)存取位於[伺服器](http://zh.wikipedia.org/wiki/%E6%9C%8D%E5%8A%A1%E5%99%A8)磁碟中的資料，是在[Unix](http://zh.wikipedia.org/wiki/Unix)系統間實作[磁碟](http://zh.wikipedia.org/wiki/%E7%A3%81%E7%A2%9F)檔案分享的一種方法。NFS的基本原則是「容許不同的[客戶端](http://zh.wikipedia.org/wiki/%E5%AE%A2%E6%88%B6%E7%AB%AF)及[伺服端](http://zh.wikipedia.org/wiki/%E4%BC%BA%E6%9C%8D%E7%AB%AF)通過一組[RPC](http://zh.wikipedia.org/wiki/RPC)分享相同的檔案系統」，它是獨立於[作業系統](http://zh.wikipedia.org/wiki/%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1)，容許不同[硬體](http://zh.wikipedia.org/wiki/%E7%A1%AC%E4%BB%B6)及作業系統的系統共同進行檔案的分享。  NFS提供了以下的服務：   * 在[目錄](http://zh.wikipedia.org/wiki/%E7%9B%AE%E9%8C%84)（directory）中尋找檔案 * 列出目錄中的檔案 * 管理目錄 * 取得各檔案的屬性（file attribute） * 檔案的讀／寫   -> 最常見的是SANBA (把目錄分出來給使用者使用) |
| PTT |  |
| 解釋 | Shared Storage :  SAN -> 用Fiber, Switch +HPA Card串起來, 高速的儲存位置 (較貴)  **儲存區域網絡**（[英文](http://zh.wikipedia.org/wiki/%E8%8B%B1%E6%96%87)：**Storage Area Network**，[縮寫](http://zh.wikipedia.org/wiki/%E7%B8%AE%E5%AF%AB)：**SAN**）是一種連接外接存儲設備和[伺服器](http://zh.wikipedia.org/wiki/%E6%9C%8D%E5%8A%A1%E5%99%A8)的架構。人們採用包括光纖通道技術、[磁碟陣列](http://zh.wikipedia.org/wiki/RAID)、[磁帶櫃](http://zh.wikipedia.org/wiki/%E7%A3%81%E5%B8%B6%E6%AB%83)、[光碟櫃](http://zh.wikipedia.org/w/index.php?title=%E5%85%89%E7%9B%98%E6%9F%9C&action=edit&redlink=1)的各種技術進行實現。該架構的特點是，連接到[伺服器](http://zh.wikipedia.org/wiki/%E6%9C%8D%E5%8A%A1%E5%99%A8)的存儲設備，將被[作業系統](http://zh.wikipedia.org/wiki/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F)視為[直接連接的存儲設備](http://zh.wikipedia.org/w/index.php?title=%E7%9B%B4%E6%8E%A5%E8%BF%9E%E6%8E%A5%E7%9A%84%E5%AD%98%E5%82%A8%E8%AE%BE%E5%A4%87&action=edit&redlink=1)。儘管SAN的複雜度和價格已經下降，但目前在大型[企業級存儲方案](http://zh.wikipedia.org/w/index.php?title=%E4%BC%81%E4%B8%9A%E7%BA%A7%E5%AD%98%E5%82%A8%E6%96%B9%E6%A1%88&action=edit&redlink=1)以外還應用不甚廣泛。  與SAN相比較，[網絡儲存設備](http://zh.wikipedia.org/w/index.php?title=%E7%BD%91%E7%BB%9C%E5%82%A8%E5%AD%98%E8%AE%BE%E5%A4%87&action=edit&redlink=1)使用的是基於文件的通信協議，例如[NFS](http://zh.wikipedia.org/wiki/NFS)或[SMB/CIFS](http://zh.wikipedia.org/wiki/SMB)通信協議就被明確的定義為遠程存儲設備，計算機請求訪問的是抽象文件的一段內容，而非對磁碟進行的塊設備操作。 |
| PTT |  |
| 解釋 | Shared Storage – CEPH  把資料切成很多小塊存在CEPH的Storage Cloud, 需要時候透過Metadata Cluster 尋找它的位置在哪, 再把資料找出來之後給instance使用 |
| PTT |  |
| 解釋 | Storage 考量 (1) Performce (2) Price (3)Scalability (4) no vendor lock-in (5) Standard (6)Stability |
| PTT |  |
| 解釋 | Success Case – Shared Storage - > CERN -> HLC (大型粒子對撞機), 產生的資料用tape, Compute用CEPH進行 |
| PTT |  |
| 解釋 | CERN上面用的架構 ->  Deploy node幫忙deploy controller or compute, controller node 有五個CEPH-mon  ->儲存資料 ->接著用1G網路串接  下面有Ceph storage Node, 用10G網路串接, 每個有24Disk, 總共47台 (提供儲存資料的流量需求) |
| PTT |  |
| 解釋 | 儲存考量的問題非常的多 (Volume 應用很廣- > CINDER 備份做在 SWIFT來做備援機制(相對的compute做設定即可) |
| PTT |  |

**第六章**

|  |  |
| --- | --- |
| 解釋 | OpenStack 與 AWS的對應服務 |
| PTT |  |
| 解釋 | Openstack 架構, |
| PTT |  |
| 解釋 | 傳統上做以下兩個HA都沒有比較好的解決方式  OpenStack 應用 -> Physical HA -> 有 Shared Storage , 將physical node A 移到 Node B (如果node A壞了, 可將VM 移動Node B, 壞的修好了後, 舊的資料會都清掉, 之後在加進去compute pool內)  VM web service HA -> 使用 Neutron的Load balancer來進行流量分散 |
| PTT |  |
| 解釋 | 安裝部屬方面, Cobber裝作業系統, 裝腳色用Puppet, |
| PTT |  |
| 解釋 | 利用Neutron Load balancer 配合 Ceilometer機制可以用來做Auto-Scaling  (需要用到Heat元件, if cpu 使用率超過多少…才增加VM) |
| PTT |  |
| 解釋 | 可以使用Neutron來做SDN (有OpenFlow controller) (搭配, EX: Solution 1  沒有OpenFlow controller的話, 可以傳統方式(用原本Neutron功能就好)  用SDN網卡也可以做! |
| PTT |  |
| 解釋 | 利用Neutron套件 -> VPN as a service -> 來用跟其他的雲做串接  以傳統架構上, 有三個分公司, 都是用VPN來做 (用openstack這個套件會快很多) |
| PTT |  |
| 解釋 | 而最近比較火紅是 HPC(Host Private Cloud)  假設公司想要用雲, 沒有相關人員可做? 可以買Server 請IDC控管 (提供monitor系統) |
| PTT |  |
| 解釋 | 部屬方面  1. 有Deploy node 來部屬 Openstack環境  2. 角色  3. Storage  4. 計算所需資源  5. 網路架構部屬 |
| PTT |  |
| 解釋 | 方案應用架構設計 |
| PTT |  |

**第七章 應用架構設計**

|  |  |
| --- | --- |
| 解釋 | Physical Resource Calculate  Instance stored location and difference  Application Model and Analysis |
| 解釋 | * Physical Resource Calculate   Flavor , 計算以下面圖片方式進行, 會先將CPU切xlarge(約四分之一), 然後才往下繼續切,記得VM真的run之後, 資源來自分割的實體主機(無法跨實體主機) |
| PTT |  |
| 解釋 | Computw node - CPU  CPU計算上, 假設要開一百個VM, 需要1.07CPU 雙核心 (100 x 1.07 x 2 /2.4(實體主機單顆cpu的功率) = 89 Physical Core  89/8 -> 得11 slot去插CPU -> 除2 因為一台server有2個slot-> 100/6 = 17(個VM) |
| PTT |  |
| 解釋 | 記憶體計算 (最後實體主機OS也有記憶體佔據) |
| PTT |  |
| 解釋 | 儲存空間計算, |
| PTT |  |
| 解釋 | IOPS計算, 2.5T local Storage, 17個VM同時對2.5TB進行存取會有Performance問題, 所以用RAID10或SSD(效能較高)解決 |
| PTT |  |
| 解釋 | 架構網路 |
| PTT |  |
| 解釋 | 為何捨去小數點？Overcommit(類似超頻)  - > OpenStack官網: CPU 16:1 RAM: 1.5:1 (比例太高, 畢竟平常的主機實際運作不那麼好)  建議 -> CPU 6:1 RAM: 1.2:1 |
| PTT |  |
| 解釋 |  |
| PTT |  |
| 解釋 |  |
| PTT |  |
| 解釋 |  |
| PTT |  |
| 解釋 |  |
| PTT |  |