ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA SEDE DI CESENA

SECONDA FACOLTA' DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria informatica

STRUMENTI OPEN-SOURCE PER LA GESTIONE INTEGRATA DI INFRASTRUTTURE PER CLOUD COMPUTING

Tesi di Laurea in Laboratorio di Reti di Telecomunicazioni

Relatore:
Prof. WALTER CERRONI

Realizzata da:
LORENZO FORCELLINI REFFI

Sessione II
Anno Accademico 2011/2012

...ai miei genitori, Claudio e Francesca
alla mia famiglia
agli amici con cui ho vissuto a Bologna
a Rachele...

"Se vuoi farti buono, pratica queste tre cose e tutto andrà bene: allegria, studio, preghiera. E' questo il grande programma per vivere felice, e fare molto bene all'anima tua e agli altri"

-Don Bosco-

INTRODUZIONE

La corsa alle informazioni, la possibilità di accedere a dati in quantità illimitata nel tempo e nello spazio, aggiornamenti real-time, servizi efficienti, completi, adatti alle esigenze di chiunque, sincronizzazione avanzata sono solo alcuni aspetti di un ambito sempre più alla base del supporto alla società moderna: le tecnologie dell'informazione e della comunicazione, più comunemente sintetizzate in ICT.

Se per tecnologie dell'informazione si intendono tutti quei meccanismi atti a gestire dati di qualsiasi tipo od uso, classificati in due macro-gruppi quali hardware e software, si sfocia nell'ambito delle telecomunicazioni nel momento in cui le informazioni necessitano di essere trasferite, ovvero quando i dispositivi devono comunicare fra loro.

In origine i sistemi informatici erano prevalentemente sviluppati attorno ad un uso "locale" ma l'esplosione delle enormi potenzialità relative alle comunicazioni, favorite dalla nascita di Internet, non poté che spostare le attenzioni sull'aspetto relazionale dei sistemi, visto e considerato l'insieme di benefici derivanti difficilmente confutabili.

Al giorno d'oggi la spinta in questa direzione è talmente elevata da richiedere un cambio di passo importante a tutto l'ambiente, il quale richiede tecnologie in grado di modificare radicalmente la concezione dei sistemi spostandone il cuore, la base, ma anche la progettazione da un impostazione locale, e conseguentemente condivisa ad un impostazione prettamente votata alla rete.

A livello pratico questo è traducibile per esempio mediante applicazioni, fino a ieri installate e lanciate sul proprio pc, unicamente eseguite sul web fino all'uso dell'intero sistema operativo a distanza comportando un' approccio differente alla programmazione, allo sviluppo di software, ma

anche al potenziamento delle infrastrutture e alla progettazione dei dispositivi hardware.

Lo sviluppo delle tecnologie legate ad Internet ha raggiunto quindi un grado talmente elevato da richiedere il passaggio a strumenti più avanzati, apportando innumerevoli benefici ed introducendo d'altro canto problematiche da studiare, analizzare e migliorare. Non è difficile pensare ad esempio ai limiti delle infrastrutture. Cosa succederà quando la maggior parte delle attività computazionali utili ad un qualsiasi cittadino si riverseranno sulla rete? Come è possibile sfruttare al meglio i canali comunicativi esistenti per fornire servizi veloci ed efficienti?

La presente tesi di Laurea, dal titolo "Strumenti Open-source per la gestione integrata di infrastrutture per cloud computing" nasce dalla necessità di studiare l'impatto derivante da una delle tecnologie ascendenti del panorama descritto, e quindi in parte in grado di dare una risposta o meglio una soluzione alle domande di cui sopra: il cloud computing.

Il cloud computing è un nuovo approccio per la fornitura di risorse IT quali ad esempio spazio di memorizzazione o capacità computazionali sotto forma di servizi via rete.

In altri termini si tratta di un servizio in grado di mettere a disposizione applicazioni o spazi di archiviazione direttamente via web rendendo piuttosto agevole l'accesso e la condivisione di dati regolando la fornitura di servizi in base ad un contratto stipulato fra gestore e cliente. Questo significa esattamente ciò che è stato precedentemente descritto: tutto il carico pendente sulle risorse normalmente in locale si sposta in rete rendendo necessario lo studio, l'ottimizzazione e l'efficienza di quest'ultima in modo da garantire un servizio di qualità.

Il progetto universitario, portato avanti dal Centro Interdipartimentale di Ricerca Industriale sull'ICT (CIRI ICT) dell'università di Bologna, il quale lavora alla *"promozione del trasferimento tecnologico e il sostegno*

all'innovazione per le imprese del territorio grandi, medie e piccole" [13], si propone così di effettuare esperimenti di ottimizzazione dei servizi web mediante Openstack, la piattaforma cloud open-source attualmente più avanzata.

Il primo passo in tale direzione, scopo di questa tesi, si riassume nella messa a punto dei sistemi, delle architetture e quindi del testbed Openstack facente parte del CIRI ICT presso la Facoltà di Ingegneria di Cesena per abilitare la fase sperimentale.

Proprio quest'ultima sarà effettuata in seguito attraverso l'interazione e la collaborazione con un sistema simile installato nella sede universitaria di Bologna.

La redazione del presente documento si articola fornendo nel primo capitolo una panoramica informativa inerente la tecnologia cloud, proseguendo con il secondo capitolo atto alla descrizione dello strumento utilizzato nel caso di studio quindi Openstack per terminare con il terzo capitolo in cui vengono descritti i passi necessari all'installazione e all'esecuzione del sistema stesso.

INDICE

Capitolo 1

SISTEMI DI CLOUD COMPUTING

"Cloud computing is a model for enabling convenient,on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability." *[1]*

Il Cloud computing è un modello per abilitare un accesso conveniente su richiesta via rete ad un insieme configurabile di risorse computazionali condivise (es. rete, server, archiviazione, applicazioni, servizi) che possono essere rapidamente rilasciate con una gestione degli sforzi o un interazione col provider minima.

In questo modo il NIST (National Institute of Standards Technology) definisce ufficialmente il cloud computing, gettando le basi in grado di distinguere servizi cloud dal più tradizionale Internet.
Tentando brevemente di semplificare la definizione al fine di una maggiore comprensione è possibile descrivere un sistema cloud come un sistema che mette a disposizione applicazioni e servizi, che un tempo risiedevano su pc o server aziendali, direttamente sul web offrendo le proprie funzionalità in base alla necessità del cliente e regolamentando le prestazioni mediante un contratto.

La definizione data dal NIST si propone quindi di mettere in luce le principali caratteristiche di tali sistemi riassumibili in questo modo *[2] [3]*:

On-demand - I servizi disponibili sono rilasciati su richiesta in base alla necessità del cliente e regolamentati da un contratto, il quale stabilisce i costi in relazione, facendo un esempio, al tempo di utilizzo di questi ultimi o alla quantità di risorse utilizzate.

Network access - L'ambiente di esecuzione di un servizio cloud è indubbiamente la rete che si trova nelle condizioni di gestire un carico elevato in termini di traffico dati, di conseguenza l'accesso a banda larga diventa un aspetto fondamentale. Quest'ultimo è reso disponibile ad una grande varietà di dispositivi soprattutto mobili che devono avere la possibilità quindi di interfacciarsi nonostante la loro eterogeneità.

Shared resources - Le risorse che il gestore del servizio mette a disposizione, assegnate come spiegato in base alla reale necessità, sono condivise con tutti gli utenti. La loro locazione fisica non è di interesse del cliente, il quale, tuttavia, può ottenere il rispetto di vincoli sull'area geografica utilizzata in particolare in merito all'archiviazione di file.

Flexibility - Le risorse vengono fornite e rilasciate agilmente, elasticamente, garantendo grande efficienza e dando l'impressione di essere illimitate.

Monitored services - I sistemi cloud sono in grado di monitorare ed analizzare l'utilizzo delle risorse in modo da adattarsi al variare delle condizioni, ad esempio in caso di picchi di richiesta di banda, continuando a garantire dunque la qualità del servizio.

1.1 CLOUD: LA NUVOLA

Dopo aver rapidamente introdotto le principali caratteristiche concettuali è fondamentale analizzare come queste vengano rispettate e realizzate cominciando dall'aspetto più generale.

Che cosa si intende quindi per "nuvola"?
La nuvola può essere vista come un infrastruttura scalabile che supporta ed interconnette i servizi di cloud computing, ovvero è un insieme di computer connessi e virtualizzati considerati come una risorsa computazionale unica ed unificata.
Nel cloud computing infatti non esiste un server singolo che nelle vesti di fornitore permette di essere raggiunto dai clienti come tradizionalmente accade, bensì esiste un gruppo distribuito di server interconnessi, la nuvola appunto, che gestiscono servizi, eseguono applicazioni ed archiviano documenti in modo totalmente trasparente agli utenti distribuendosi compiti e carichi di lavoro.

Esistono molte società che forniscono tale tecnologia fra le quali Microsoft, Google, Apple o Amazon citandone alcune, le quali hanno seguito nel tempo strade alquanto diverse nello sviluppo e progettazione dei propri sistemi.
Proprio questa differenziazione, riassumibile nel tipo di servizio offerto, lo spazio di archiviazione disponibile, le piattaforme di sviluppo software o le applicazioni online messe a disposizione ha portato inevitabilmente a problemi sia di interazione fra le varie "nuvole", creando la necessità di una standardizzazione, sia di confronto fra le stesse. Quest'ultimo aspetto non è da sottovalutare considerando l'importanza di distinguere velocemente i

vari servizi cloud esistenti, esaminarli specificandone le caratteristiche, compararli facilmente e fornire una terminologia tecnica.
Tutti questi aspetti hanno portato quindi all'introduzione di una classificazione definitiva di tali sistemi.

1.2 – CLASSIFICAZIONE: I LIVELLI IaaS, PaaS, SaaS

Come precedentemente anticipato la necessità di confrontare i diversi sitemi cloud ha indotto il NIST a creare una classificazione articolata in tre livelli *[4]*:

- Infrastructure as a service (IaaS)
- Platform as a service (PaaS)
- Software as a service (SaaS)

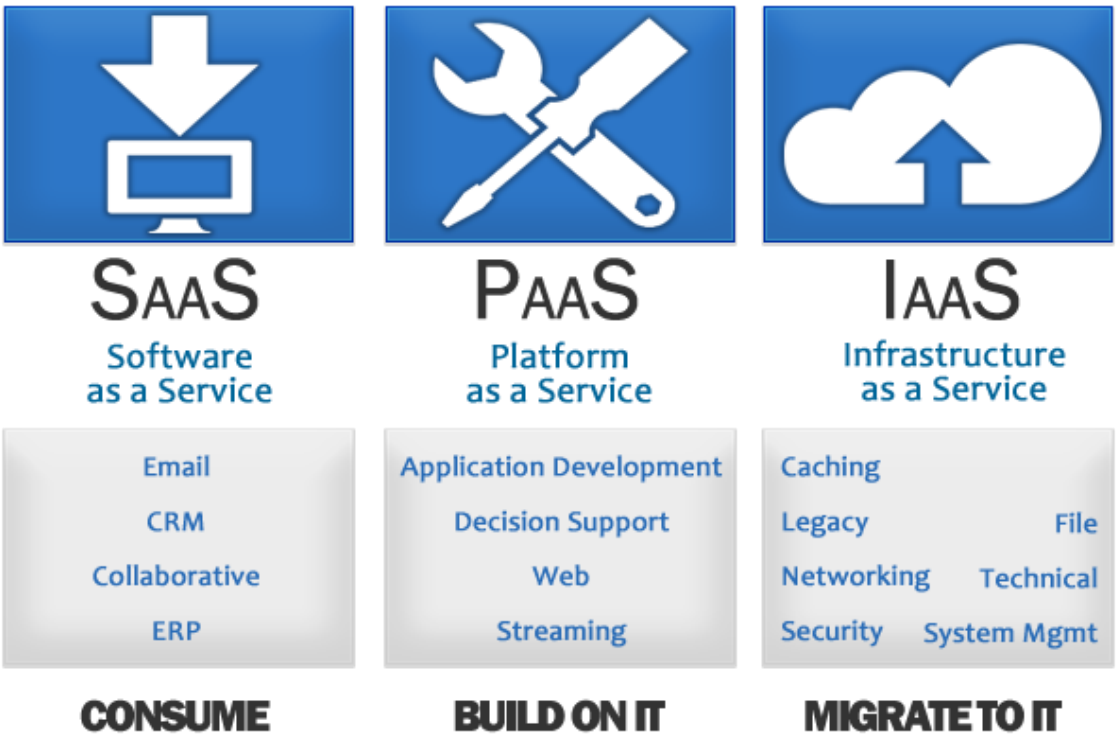

Figura 1.1: I tre livelli di classificazione dei servizi cloud

- Infrastructure as a service (IaaS)

Modello di servizio che prevede l'offerta di risorse hardware remote con capacità computazionali e di archiviazione sulle quali è possibile installare ed eseguire software.

Da sottolineare che le risorse rese disponibili possono essere virtuali, caratteristica fondamentale che ne rende possibile l'utilizzo su richiesta nel momento in cui un cliente ne ha bisogno, e non assegnate a prescindere dall'utilizzo effettivo. Di conseguenza il costo economico risulta tarato sull'impiego e sulle risorse realmente utilizzate.

Il collegamento fra le risorse virtuali e le reti aziendali che sfruttano i servizi offerti può essere inoltre gestito mediante VPN, reti virtuali private implementate su linee pubbliche. Tale soluzione, denominata Hybrid cloud, viene descritta successivamente.

Volendo infine citare una società che ha investito sulla virtualizzazione nel modello Iaas è giusto fare il nome di Amazon con Elastic Compute Cloud (EC2), uno fra i più conosciuti sistemi di gestione ed utilizzo di virtual machine in ambito cloud.

- Platform as a service (PaaS)

Modello di servizi inerenti alla distribuzione di piattaforme di elaborazione che permettono di sviluppare, testare e gestire le applicazioni tramite un interfaccia di programmazione. Il grande vantaggio riscontrato risiede nell'abbattimento dei costi e delle complessità associate all'acquisto, la configurazione, e la gestione dell'hardware e del software di base.

Queste piattaforme sono viste come infrastrutture ad alto livello che forniscono tools e linguaggi di programmazione, Python o Java citando

alcuni esempi, molto adatti appunto allo sviluppo di applicazioni web. In questo caso è possibile citare Google App Engine o la piattaforma Azure di Microsoft rimanendo nella sfera delle multinazionali.

- Software as a service (SaaS)

In questo caso si tratta infine di un modello di distribuzione del software applicativo dove un produttore mette applicazioni eseguibili via internet a disposizione dei propri clienti.

Un esempio molto diffuso sono i sistemi mail via web, oppure le applicazioni di Google quali Google Documents. Risulta chiaro che questo livello è estremamente importante essendo il più vicino alle esigenze dell'utente privato. *[5]*

1.2.1 – TIPOLOGIE

Accanto alla classificazione del NIST vengono poi presentate le diverse tipologie di sistemi cloud, le quali si differenziano tramite caratteristiche valutabili in base all'utilizzo finale *[3]*.

PUBLIC CLOUD:

Un fornitore mette a disposizione le risorse a chiunque stipuli un contratto ed i clienti utilizzano i servizi, in relazione alle proprie necessità, potendo risparmiare i costi di acquisto e manutenzione. Non è d'altro canto possibile definire le politiche di sicurezza ne sapere dove effettivamente risiedono i dati, mentre il fornitore si impegna ad utilizzare meccanismi di accesso da parte di clienti diversi in modo da mantenere separati i dati di ciascuno risedenti sulla stessa macchina fisica.

PRIVATE CLOUD:

Il cloud privato è installato dall'utente nel proprio data center ed è a suo uso esclusivo. Questo determina ovviamente il superamento dei problemi legati al cloud pubblico, quali la locazione dei dati o la gestione della sicurezza. Normalmente è un tipo di sistema sfruttato dalle grandi aziende.

COMMUNITY CLOUD:

Sistema utilizzato da enti che condividono l'infrastruttura su cui sono installati i servizi, avendo obiettivi ed esigenze comuni. Si tratta di una tipologia utile nel supporto ad esempio degli enti governativi, e potrebbe essere gestita da un ente esterno agli utilizzatori. Combina i vantaggi del cloud pubblico aumentandone gli standard di sicurezza.

HYBRID CLOUD:

Il sistema ibrido è il più adatto a correlare le caratteristiche di cloud pubblico e privato fornendo un servizio completo permettendo al cliente di utilizzare sia il primo che il secondo a discrezione delle necessità.
Ad esempio l'utilizzatore potrebbe lavorare su dati importanti all'interno del cloud privato aumentandone la sicurezza per poi spostarsi sul pubblico nel resto delle proprie attività, oppure potrebbe risultare utile nella gestione di picchi di carico spostandosi in ambiente pubblico solo al momento di necessità dovuto alla grande richiesta, che verrebbe così ridistribuita. La possibilità di scelta dell'ambiente su cui indirizzare i propri dati porta quindi notevoli vantaggi.

Terminata la descrizione delle classificazioni dei sistemi presi in esame è opportuno affrontare un tema fondamentale proprio del cloud computing. La tecnica della virtualizzazione.

1.3 – VIRTUALIZZAZIONE

La virtualizzazione dei server è una tecnica fondamentale per i sistemi cloud.

Questa tecnologia permette infatti l'astrazione dei servizi IT dalle risorse fisiche in modo da poter eseguire più sistemi operativi distinti sulla stessa macchina, creando la possibilità di servire simultaneamente più clienti con le stesse risorse. Il sistema operativo installato sul server o sul computer host è legato ad un layer di virtualizzazione, l'hypervisor, che gestisce l'emulazione dei sistemi virtuali.

E' ovvio quindi che i sistemi operativi non hanno accesso diretto alle risorse fisiche bensì è la macchina virtuale che li gestisce.

In ambito cloud questo aspetto diventa quindi imprescindibile, dato che viene incontro a molte esigenze permettendo la gestione di più utenti separatamente con un numero di macchine fisiche non vincolato al rapporto uno a uno (una macchina = un cliente).

I vantaggi sono considerevoli. I costi si riducono notevolmente grazie alla diminuzione di macchine fisiche così come lo spazio utilizzato da queste ultime risulta ovviamente minore rendendo così possibile l'espansione anche di piccole aziende con poco spazio a disposizione. *[6]*

Sintetizzando i vantaggi della virtualizzazione risultano:

- **Riduzione di costi di gestione hardware**
- **Riduzione del consumo energetico**
- **Allocazione di risorse dinamico in base alla necessità**
- **Riduzione del tempo necessario all'installazione ed avviamento di nuovi sistemi**

- **Isolamento dell'architettura da problemi a livello di sistema operativo e applicativo**
- **Possibilità di semplificazione nella gestione di risorse eterogenee**
- **Riduzione dello spazio fisico necessario**
- **Riduzione della possibilità di guasti tecnici**

Per finire un importante punto a favore della struttura virtualizzata si manifesta nella possibilità di salvare agilmente un backup completo della macchina, comprese quindi le impostazioni del sistema operativo, normalmente molto critiche da ripristinare.
Questo permette inoltre la grande semplicità con cui diventa possibile gestire l'avanzamento tecnologico dei sistemi. Se l'hardware diventa obsoleto rimane piuttosto semplice migrare i server su macchine di ultima generazione.

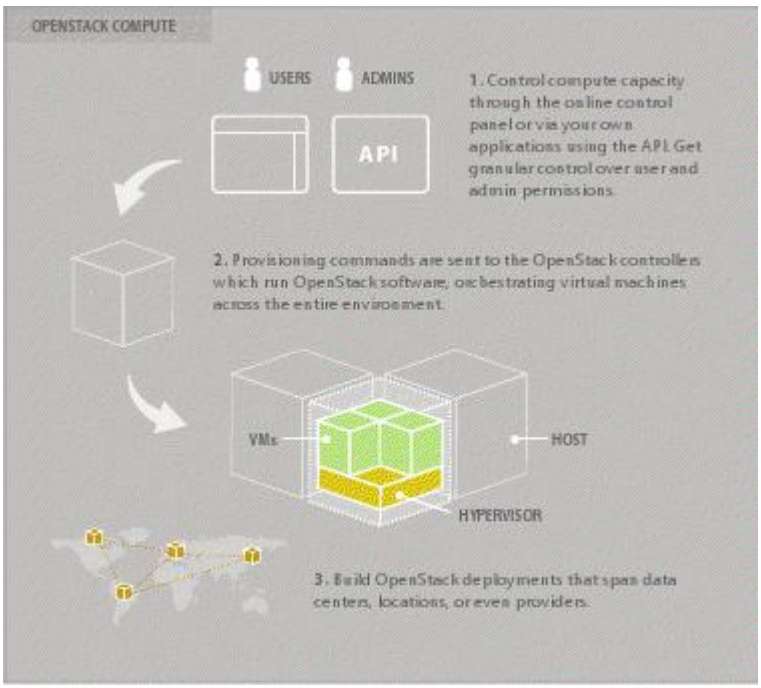

Figura 1.2: La virtualizzazione all'interno del sistema Openstack

E’ importante a questo punto sottolineare però, quale sia il principale ostacolo all’adozione di tale tecnologia. Le industrie sono ancora restie ad investire sulla virtualizzazione a causa della sicurezza. Le tecniche relative a questo tema nei sistemi virtuali sono molto meno avanzate rispetto ai sistemi fisici anche se le problematiche sono analoghe partendo dalle vulnerabilità e stabilità dei sistemi ed arrivando alle minacce web.

E’ ipotizzabile in ogni caso che nel tempo, mediante l’affinamento dei principali meccanismi di sicurezza, supportata dagli evidenti vantaggi dimostrati se non necessità soddisfatte, sospinta dalla diffusione del cloud computing, questa tecnologia verrà adottata dalla maggior parte di fornitori di servizi informatici e delle telecomunicazioni.

1.4 - SICUREZZA

Introdotta precedentemente solo a livello di virtualizzazione è importante concentrarsi ora sulla sicurezza, un aspetto critico anche dei sistemi presi in considerazione *[7][3]*.

Gestire quantità rilevanti di dati in un ambito condiviso introduce due problematiche essenziali: i meccanismi di protezione riguardante la privacy, ossia la violazione dei dati, e la tutela contro eventuali rotture hardware e quindi la perdita delle informazioni.

A tal proposito diventano significative leggi e regolamentazioni, come ad esempio quelle dei paesi inseriti nell'Unione Europea. In questi Stati vigono direttive piuttosto precise in merito alla protezione dei dati come l'obbligo di conoscere l'esatta locazione di questi ultimi che un sistema cloud non riesce sempre a garantire.

Data la mancanza di standardizzazione ogni sistema affronta in modo autonomo la gestione della sicurezza ma ognuno di essi utilizza fondamentalmente due tecniche come la cifratura delle informazioni, e l'autenticazione. La prima permette di tutelarsi dalle intercettazioni la seconda regolamenta gli accessi rientrando entrambe nella sfera della sicurezza esterna del sistema. Occupandosi invece della sicurezza interna, l'attenzione si sposta sulla separazione dei dati fra istanze virtuali e clienti diversi. E' obbligatorio garantire infatti che gli utenti siano liberi di accedere solamente ai propri file e non riescano in nessun caso, erroneamente o illecitamente mediante attacchi, ad entrare in possesso di dati altrui conservati nella nuvola. Lo sviluppo della sicurezza si muove quindi in questi termini, momentaneamente, dando uno sguardo ai protocolli comunemente utilizzati si trova l'HTTP per quanto riguarda

l’accesso e l’utilizzo dei servizi mentre la cifratura dei dati è affidata alla crittografia SSL/TLS.
In ogni caso il problema della sicurezza pone in essere un doppio risvolto per quel che riguarda l’utente ma ancor meglio l’azienda:
il rischio di affidare i propri dati ad un componente esterno al quale dover dare fiducia, sia in merito alla qualità del servizio sia al rischio di compromissione dei dati, è contrapposto agli innumerevoli vantaggi in termini si di efficienza ma soprattutto, prendendo ad esempio le piccole aziende le quali non possono permettersi staff o servizi di sicurezza avanzati per questioni prettamente economiche, in termini di realizzazione di un ampio miglioramento del proprio standard di sicurezza. Tutto questo comprendendo anche la risoluzione di tutti quegli aspetti riguardanti l’installazione, la gestione e quindi il funzionamento ottimale dei numerosi e complicati software specifici dedicati proprio alle aziende. L’unico compito del cliente quindi è utilizzare il servizio come meglio crede.

E’ possibile infine riassumere i rischi legati alla sicurezza in punti ben precisi:

- **Aumento del rischio delle esposizioni durante la migrazione dei dati. Sviluppare la crittografia è importante**
- **Controllo approssimativo della locazione dei dati**
- **Controllo dell’ accesso ai dati**
- **Monitoraggio migliorabile, ovvero è necessario sapere cosa accade nei momenti di assenza**
- **Rischi di interruzione del servizio**
- **Isolamento dei dati condivisi. Ogni utente deve accedere solo ai propri dati.**

- **Utilizzo obbligatorio di interfacce in rete con conseguente esposizione a rischi.**
- **Le risorse possono essere effettivamente rimosse ma non distrutte. La distruzione, visto la condivisione di supporti hardware è molto complicata.**

La risoluzione di tutti questi aspetti è fondamentale alla conquista della fiducia delle aziende, punto critico dell'utilizzo di sistemi cloud.
E' già stata descritta la variazione del grado di sicurezza a seconda della tipologia di sistema: privato, ibrido e pubblico.
In un ambiente privato, ovvero un infrastruttura creata e gestita unicamente per un azienda, è quest'ultima ad essere responsabile del sistema, diventando in grado di personalizzare i controlli di sicurezza, monitorare gli accessi e le operazioni. Ciò non è possibile in un infrastruttura pubblica nella quale i dati sono condivisi e a disposizione del pubblico o di un grande gruppo industriale. In questo caso è importante sottolineare nuovamente l'importanza dei meccanismi di isolamento dati, i quali garantiscono l'uso unicamente al proprietario.
Infine le infrastrutture ibride, che comprendono ambienti separati ma interconnessi, si pongono come già visto a metà fra i due sistemi tentando di coniugare i vantaggi di entrambe le strutture limitandone le criticità.

Concludendo è possibile riassumere la sicurezza dei servizi cloud come l'aspetto più critico in termini di appetibilità per le società contrapponendo considerevoli vantaggi in termini di investimenti iniziali e di gestione, innalzamento dello standard tecnologico ad un costo accettabile e di conseguenza l'apertura anche alle piccole aziende.

1.5 – BENEFICI

Il cambio tecnologico promosso dai servizi cloud è indubbiamente di grande impatto. E' possibile soffermarsi sui vantaggi derivanti da questa tecnologia focalizzandosi sulla capacità, ad esempio per le aziende, di accedere a software e servizi eseguibili esclusivamente via web potendo fruirne rivolgendosi a società esterne. *[3]*

I costi da sostenere riguardano quindi la locazione, non più, come in passato, l'acquisto di hardware e software che venivano ospitati materialmente presso la propria sede aziendale (es. server aziendali). Questo si traduce effettivamente nel pagare in relazione al consumo quindi in termini di cicli cpu/ram/spazio disco, proprio come accade ad esempio per l'energia elettrica, l'acqua, o il gas nelle case di chiunque. I costi di avviamento dell'infrastruttura risultano quasi annullati e si rende possibile un ampia ma graduale scalabilità dei sistemi.

Questa sorta di elasticità permette di eseguire operazioni complesse, ordinariamente ostacolate da costi e tempistiche molto elevate, rapidamente e con un impatto contenuto. Proprio flessibilità e scalabilità, come precedentemente sottolineato, vengono supportate dalla virtualizzazione, occupante un ruolo primario nel comparto tecnologico richiesto, mentre la diminuzione dei costi riduce drasticamente il fattore di rischio sui nuovi investimenti favorendo lo sviluppo e tutelando l'azienda.

Locazione, scalabilità e portabilità sono le qualità su cui il cloud computing getta le basi del proprio successo. Ovviamente anche a livello di gestione delle applicazioni vengono riscontrati discreti vantaggi in quanto questa è delegata alla società che fornisce il servizio liberando il cliente da aggiornamenti e gestione di minacce quali i virus.

Infine è importante soffermarsi sul fatto che i benefici non si limitano all'aspetto economico o di gestione degli spazi ma impattano direttamente

sul settore che concerne i sistemi informativi, un area tecnologica ampiamente diffusa ed utilizzata nella società moderna aprendo le porte a quelle che sono grandi prospettive.

1.6 – GLI UTENTI

Ritenendo conclusa l'analisi delle caratteristiche fondamentali di un sistema cloud è necessario considerare i destinatari di tale tecnologia. *[3]* Questa infatti, come intuibile, si rivela funzionale sia all'ambito privato, sia all'ambito lavorativo in particolare aziendale.

Privati - Gli utenti diventano in grado di accedere facilmente da qualsiasi dispositivo dotato di connessione Internet, (PC, tablet, smartphone..) ai propri dati. Viene superata la necessità di replicare le informazioni fra più dispositivi, con una semplificazione della sincronizzazione, consentendo di eseguire anche applicazioni ovunque ci si trovi.

Aziende - In ambito lavorativo i servizi cloud possono essere usati per condividere dati fra più sedi aziendali, sgravano le società da problemi manutentivi e costi di avviamento/installazione di software complessi, innalzano lo standard di sicurezza per piccole imprese, favoriscono lo sviluppo e l'efficienza come precedentemente descritto.

Enti Governativi - Sicuramente uno dei destinatari più interessanti dato che la necessità di informatizzazione all'interno degli organismi statali, quali ad esempio Pubblica Amministrazione o sanità, si coniuga perfettamente con i sistemi cloud i quali sono in grado di rispondere alle richieste di efficienza e rapidità fondamentali all'interno di uno Stato. Lo stesso si potrebbe dire, a livello più limitato, di società e organizzazioni.

1.7 - LE GRANDI SOCIETA'

Da alcuni anni diverse aziende hanno incominciato ad offrire le proprie infrastrutture di cloud computing. Nel 2006 Amazon lanciava la piattaforma S3 (Simple Storage Service), con la quale consentiva l'accesso ad uno storage attraverso una semplice interfaccia web, calcolando mensilmente i consumi in termini di gigabyte e di banda utilizzata. Ad un costo davvero basso vengono tuttora garantite scalabilità, affidabilità e alta disponibilità.

Successivamente diversi altri servizi come DropBox si sono appoggiati ad Amazon S3, utilizzandone lo storage come base di dati per le proprie soluzioni di file sharing. Poco dopo venne lanciato Amazon EC2 (Elastic Computer Cloud) il quale consentiva di utilizzare istanze di servizi per l'esecuzione di macchine virtuali all'interno di un'infrastruttura di cloud dando inizio alla cooperazione fra questa e la virtulizzazione.
Al giorno d'oggi anche Google, con Google Storage o Google App Engine, fornisce la sua infrastruttura di cloud computing, non a caso sono già numerose le applicazioni utilizzate quotidianamente da utenti privati, partendo da servizi mail, calendari, contatti, documenti fino ad arrivare a Google Drive. Quest'ultimo, presentato come soluzione cloud completa, permette appunto la gestione e l'archiviazione di file direttamente sul web avendo anche a disposizione numerose soluzioni che si occupano della sincronizzazione dei dispositivi. L'alter ego in casa Microsoft si propone sotto il nome di Sky Drive, mentre per lo sviluppo di applicazioni si fa strada Microsoft Azure. Apple ha rilasciato invece iCloud ma esistono diversi altri vendor che offrono soluzioni più o meno simili quali Rackspace, IBM, Telecom Italia, HP, Sun/Oracle. Quelle citate sono

comunque solo alcune delle principali società che stanno investendo nel cloud computing, e da qui la nacita del problema riguardante l'eterogeneità. Pur consentendo ampia varietà di scelta stimolando la competizione fra gestori traducibile in offerte vantaggiose per i clienti, si risolleva la questione già trattata della mancanza di standardizzazione. Ogni vendor infatti interpreta e definisce un modello di cloud computing secondo la propria visione complicando le interazioni e rendendo necessaria una soluzione in tempi brevi vista l'ascesa e lo sviluppo di questa tecnologia.

1.8 - PROSPETTIVE

Ciò che è stato scritto in precedenza in termini di appetibilità dei servizi cloud viene confermato da una ricerca condotta da Alcatel-Lucent su circa tremila responsabili IT in sette Paesi differenti. Tale ricerca mostra come sicurezza e prestazioni siano gli elementi che frenano la diffusione della "nuvola" all'interno delle grandi imprese.

Un importante sviluppo della tecnologia legata alle telecomunicazioni, nel quale rientrano di diritto i sistemi cloud, passa obbligatoriamente dall'ottimizzazione delle infrastrutture. Questo si traduce in una velocità ed efficienza aumentata, garantita molto più dalla fibra otttica che dalle tecniche tradizionali. Punto chiave rimane inoltre la simmetria ovvero la concessione di alte velocità di traffico sia in download che in upload. Nonostante l'arretratezza vigente in quest'ambito, le grandi aziende credono e spingono in questa direzione, tesi dimostrata dal fatto che esse tentano di espandersi acquisendo società minori nonostante la crisi dei bilanci. L'obiettivo è riassumibile nella scalata a posizioni di leadership nella fornitura di servizi cloud come potrebbe essere ad esempio lo storage. Significativa all'utilizzo e alla prospettiva di espansone di tali servizi è anche lo stimolo dato dalla necessità di impiegare continuamente i grandi centri di calcolo, come nel caso di Amazon ad esempio, società di acquisti online e pioniera della tecnologia cloud che soffriva l'inutilizzo della potenza a disposizione nei periodi dell'anno meno adatti agli acquisti, lontani perciò dalle grandi festività.

La spinta decisiva per convincere le aziende utilizzatrici a far cadere gli ultimi dubbi verso il cloud sarà in ogni caso l'apertura verso la mobilità. Smartphone e tablet stanno diventando un mezzo largamente utilizzato di accesso al Web tanto che compagnie come IBM hanno annunciato soluzioni in questo senso.

Oltre all'ambito aziendale il cloud computing sarà un valido strumento sempre più evoluto. Considerando l'ambiente privato, si nota si la disponibilità di eseguire qualsiasi applicazione fornita ai clienti ovunque, memorizzare dati o sincronizzarli, ma anche di disporre di sistemi operativi completi, Chrome OS ne è il primo esempio, e software "pesanti" quali ad esempio videogiochi. Allo stesso modo l'ambiente pubblico, comprenderà, oltre alle già citate Pubbliche amministrazioni o gestione della Sanità, anche le banche e qualsiasi tipo di organizzazione.

I primi esperimenti sono già in atto. E' opportuno citare ad esempio l'edizione 2012 del Roland-Garros, prestigioso torneo di tennis nel quale IBM ha utilizzato un infrastruttura cloud per gestirne il sito internet costretto a far fronte ad un carico di rete elevatissimo, consentendo agli utenti di seguire le gare in una particolare modalità streaming avendo a disposizione numerosissimi dati in particolare accessibili con dispositivi mobili.

Parlando in termini numerici, attraverso la tecnologia di virtualizzazione di IBM, "la Federazione Francese di Tennis è riuscita a **ridurre la sua domanda di energia del 40%** e la **domanda di raffreddamento del 48%**. Nel 2006, 60 server erano delegati a far fronte all'aumento di 100 volte nel traffico web del sito del Roland-Garros durante il torneo. Ora, il tutto è affidato ai servizi cloud-based di IBM, che assegnano la quantità di servizi a supporto necessari." [8]

Ora che è stata velocemente presentata la tecnologia relativa al cloud computing è possibile passare ad un punto di vista più tecnico mediante il caso di studio oggetto di questa tesi concentrandosi sulla piattaforma Openstack.

Dopo averne brevemente descritto la storia, le caratteristiche e l'architettura si proseguirà con la descrizione della procedura di installazione ed al compimento di qualche esperimento di migrazione.

Capitolo 2

PREPARAZIONE AL CASO DI STUDIO: OPENSTACK

Nel panorama dei sistemi cloud tanti sono i progetti indipendenti portati avanti dalle società più disparate, molti meno quelli inerenti licenze open-source.

Openstack, adottato come caso di studio di questa tesi, si pone proprio come possibile soluzione a questa lacuna essendo una fra le più innovative ed avanzate piattaforme open-source per cloud-computing esistente. In dettaglio è un framework organizzato in moduli utile a creare il proprio servizio di cloud computing il quale ci permetterà, come precedentemente descritto, di memorizzare file, eseguire e migrare macchine virtuali o lanciare software su richiesta senza l'installazione in locale.

2.1 - IL PROGETTO

Nato da una collaborazione fra NASA e Rockspace cloud, il progetto risale al 2006 quando le due compagnie si unirono allo sviluppo di questa piattaforma, come spiegato in *[9]*.

L'agenzia spaziale Americana infatti, era alla ricerca di un sistema utile a possedere tutto lo spazio necessario per conservare le proprie immagini ad alta risoluzione, senza chiedere aiuto ad enti esterni. Grazie a vari consulenti aveva così dato vita ad un progetto di cloud computing. Ben presto si manifestarono però stringenti difficoltà dovute all'utilizzo di una piattaforma di sviluppo chiamata Eucalyptus, la quale, essendo sviluppata da ricercatori dell' Università della California di Santa Barbara, non era

completamente open-source ovvero alcuni codici sorgenti non erano accessibili ne modificabili.

Per poter ovviare all'inconveniente entrò in gioco la collaborazione con Rockspace, la quale lavorando ad un progetto simile di sistemi di storage e cloud computing, riuscì a creare nova, una piattaforma alternativa ad Eucalyptus adottata appunto dalla NASA. In questo modo l'agenzia spaziale potè terminare il proprio sistema cloud. L'intero progetto fu successivamente rilasciato sotto licenza Apache, la quale obbligando gli utenti a preservare l'informativa di diritto d'autore e d'esclusione di responsabilità nelle versioni modificate, veniva considerata la via migliore per garantirne la diffusione.

Recentemente lo sviluppo di Openstack ha trovato sostegno in una nutrita community formata da end-user e da grandi compagnie che hanno aderito al progetto, permettendo il frequente rilascio di aggiornamenti utili ad affinare e migliorare il sistema e alla creazione di una nutrita documentazione.

Esistono così già cinque versioni del sistema chiamate Austin, Bexar, Cactus, Diablo mentre l'ultima, la più aggiornata è Essex risalente all'Aprile 2012.

2.2 – IL FUNZIONAMENTO

Per comprendere al meglio Openstack, è opportuno riassumerne innanzitutto lo scopo ed il funzionamento, descriverne i componenti, per poi affrontare tali argomenti in dettaglio.

Openstack è un sistema formato da un cloud controller e vari host che lavorano insieme per poter eseguire macchine virtuali create dinamicamente e collegate in rete tramite le quali lanciare software archiviato in spazi chiamati volumi ed eseguire quindi operazioni computazionali bilanciando il carico fra gli elaboratori disponibili potendo così servire efficientemente più utenti.

Cloud controller - E' il componente principale, ovvero il server che gestisce gli utenti, la rete virtuale, e le immagini dei dischi.

Host - Macchine connesse al controller che formano così la nuvola. Hanno a disposizione API e una dashboard per comunicare e fungono da vero e proprio componente computazionale sul quale vengono eseguite le macchine virtuali.

Rete pubblica – La rete esterna al quale è collegato il controller

Rete fisica – La rete privata alla quale sono connessi gli host e il controller

Rete virtuale – La rete che collega le virtual machine

2.3 – I MODULI

Per poter arrivare ad offrire le funzionalità descritte nel paragrafo precedente l'intero sistema è stato suddiviso in moduli. Questi inoltre permettono anche la realizzazione dei punti di forza di Openstack, fra i quali la flessibilità che permette l'apertura ad altre tecnologie nello sviluppo del sistema, la scalabilità che consente di espandere o meno il sistema in base alle necessità, la partecipazione di grandi aziende che garantisce il rispetto degli standard aziendali o la compatibilità con innumerevoli "nuvole" esistenti.
In ogni caso i principali moduli, come descritto in *[10]*, si identificano in:

Openstack Identity – Keystone:

Keystone è il componente che assume essenzialmente due funzioni:

- la gestione degli utenti
- utilizzo di un catalogo dei servizi

Il primo punto implica l'inserimento o la modifica di nuovi utenti, la definizione di privilegi, il controllo di questi ultimi e quindi l'autenticazione tramite user e password ma anche la registrazione dei componenti dei vari host.
Il secondo si riferisce alla compilazione del catalogo tenente traccia dei servizi avviabili con l'indirizzo dei rispettivi endpoint.

Openstack Imaging Service - Glance:

Glance è il componente dedicato a catalogare grandi raccolte di immagini di dischi virtuali, all'interno di blocchi o volumi successivamente descritti, i quali contengono software da avviare sulle macchine virtuali. La divisione dei compiti fra i vari componenti della nuvola implica anche che non

necessariamente la macchina contenente le immagini dei dischi sia anche la macchina che le gestisce o che ne invoca l'esecuzione.

Openstack Compute - Nova:

Nova è il componente fondamentale del sistema. Esso realizza tramite i rispettivi servizi tre principali funzioni:

- Gestisce la creazione e l'invocazione di macchine virtuali tramite hypervisor così come il montaggio di volumi di memorizzazione
- Crea e gestisce la rete di macchine virtuali
- Gestisce blocchi o volumi extra di memoria agganciabili alle istanze di macchine virtuali in grado di rendere persistenti i dati salvati.

Esso mette quindi a disposizione gli strumenti di gestione di larghe reti di macchine virtuali, realizzando una piattaforma estremamente scalabile, attraverso un pannello di controllo e API dedicate. Vari processi appartenenti al servizio concretizzano quindi le attività sopra indicate: creare, gestire ed utilizzare una rete di macchine virtuali (nova-network), imbastire le condizioni che permettano l'archiviazione di dati relativi alle istanze virtuali in volumi, (nova-volume) gestire le macchine virtuali stesse ovvero l'interazione con l'hypervisor che permette il richiamo e l'esecuzione di queste (nova-compute), assegnare il carico computazionale (nova-scheduler).

Openstack Object Storage - Swift:

Swift è il componente utile a realizzare l'archiviazione di grandi quantità di dati, utilità sfruttata dalla maggior parte della clientela. La sicurezza di non perdere gli stessi in seguito a guasti è garantita dalla loro replicazione, la

quale varia con un costo correlato. Maggior sicurezza, maggior replica, maggior costo.

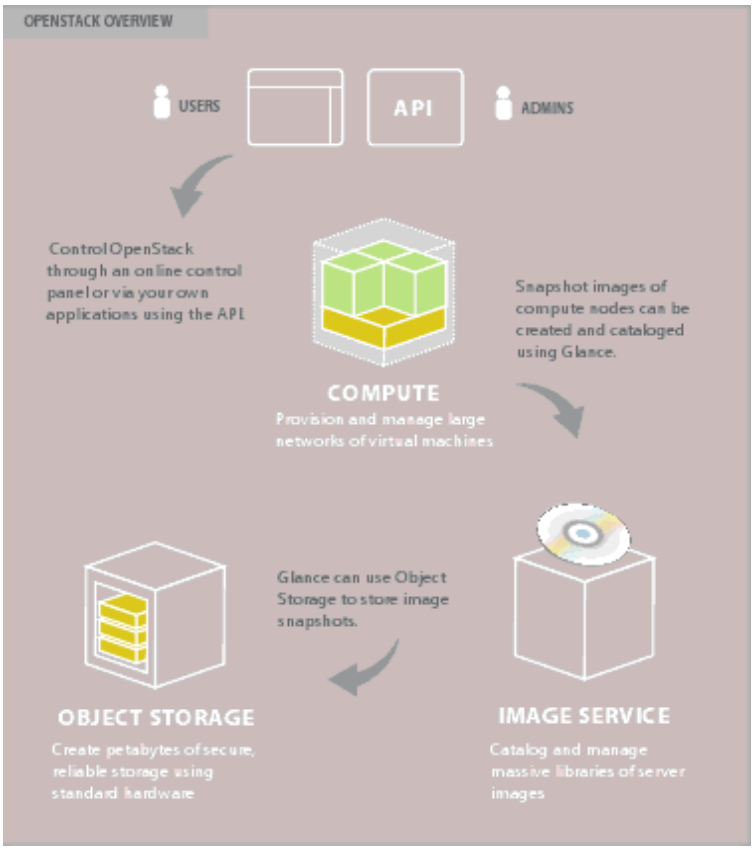

Figura 2.1: I tre moduli Openstack

2.4 – I PRINCIPALI SERVIZI

Analizzando quindi più nel dettaglio i servizi disponibili è importante soffermarsi soprattutto su nova, il componente principale che come già detto interessa tutto ciò che riguarda le macchine virtuali.

nova-compute: è il servizio che lavora insieme al driver di default, libvirt, il quale guida l'hypervisor KVM (Kernel-based Virtual Machine) utile a creare e gestire le macchine virtuali. Nel caso si utilizzi Xen come hypervisor esiste invece un altro driver dedicato. La possibilità da parte di nova-compute di montare volumi in remoto si realizza mediante Open-iscsi che gestisce il protocollo ISCSI descritto successivamente.

nova-volume: è il servizio che gestisce blocchi o volumi extra di memoria collegabili, anche in remoto, alle istanze di macchine virtuali utili a salvare dati in modo persistente. Questo risulta fondamentale essendo quindi l'unico modo per non perdere i dati in seguito allo spegnimento della virtual machine. Il componente principale che realizza le suddette funzionalità è LVM (lvm2), logical volume manager o gestore logico dei volumi. Questo è un software di gestione dei volumi di memorizzazione il quale riesce a creare più blocchi virtuali (partizioni logiche o volumi logici) da un disco fisico. In particolare crea, all'interno di un volume group, un volume logico LV che potrà essere collegato ad un istanza. I blocchi sono raggiungibili mediante ISCSI. Quest'ultimo è un protocollo che permette di utilizzare dispositivi di memoria SCSI collegati al server, ma soprattutto dispositivi virtuali in rete come nel caso di Openstack.

nova-network: è il servizio che crea e gestisce reti di macchine virtuali usando bridge per collegare reti virtuali a reti fisiche. Openstack permette

due tecniche di assegnazione ip alle macchine virtuali: flatdhcp ovvero quando una macchina virtuale si avvia ottiene l'indirizzo mediante dhcp, oppure tramite network manager avviene l'injiection, cioè la modifica diretta delle interfacce virtuali. Nel caso di studio è stato utilizzato il flatdhcp.

Da sottolineare che questi tre servizi non devono necessariamente risiedere sul server. Nel caso di studio affrontato ad esempio viene eseguito nova-network sul server mentre nova-compute e nova-volume sugli host in modo che la gestione delle macchine virtuali sia relegata a questi.

nova-scheduler: il servizio che affida la computazione ai vari host

Openstack-dashboard: Horizon è la dashboard basata su Django che gira su un server Apache fornendo un interfaccia grafica semplificata all'utilizzatore.

2.5 – LA TECNOLOGIA DI BASE

Oltre all'esemplificazione di tutte quelle tecnologie utili all'esecuzione del sistema vengono introdotti ai fini della comprensione termini ricorrenti in Openstack così come successivamente nella guida.

User – Rappresentazione digitale di una persona che utilizza i servizi di Openstack

Credenziali – Dati noti solo all'utente che permettono l'accesso ai servizi di Openstack nel procedimento chiamato autenticazione.

Token – Sequenza di caratteri usata per accedere solo a determinate risorse ed è valido per un periodo limitato di tempo.

Tenant – Una sorta di gruppo che contiene solo determinate risorse avviabili.

Role – Caratteristica assunta da un utente che ne definisce i privilegi. In Openstack il token include la lista di ruoli che l'utente può assumere.

Endpoint – Indirizzo di rete che identifica un servizio

Generalmente, in ambiente Linux, un servizio è un programma eseguito in background che rimane in ascolto su una porta rispondendo alle richieste che arrivano. Detto questo si può descrivere un servizio di Openstack come un insieme di servizi Linux, tanto che il sistema è sviluppato proprio sull'interazione fra le due tipologie. Andando ad analizzare nel dettaglio risulta utile mostrare lo schema seguente:

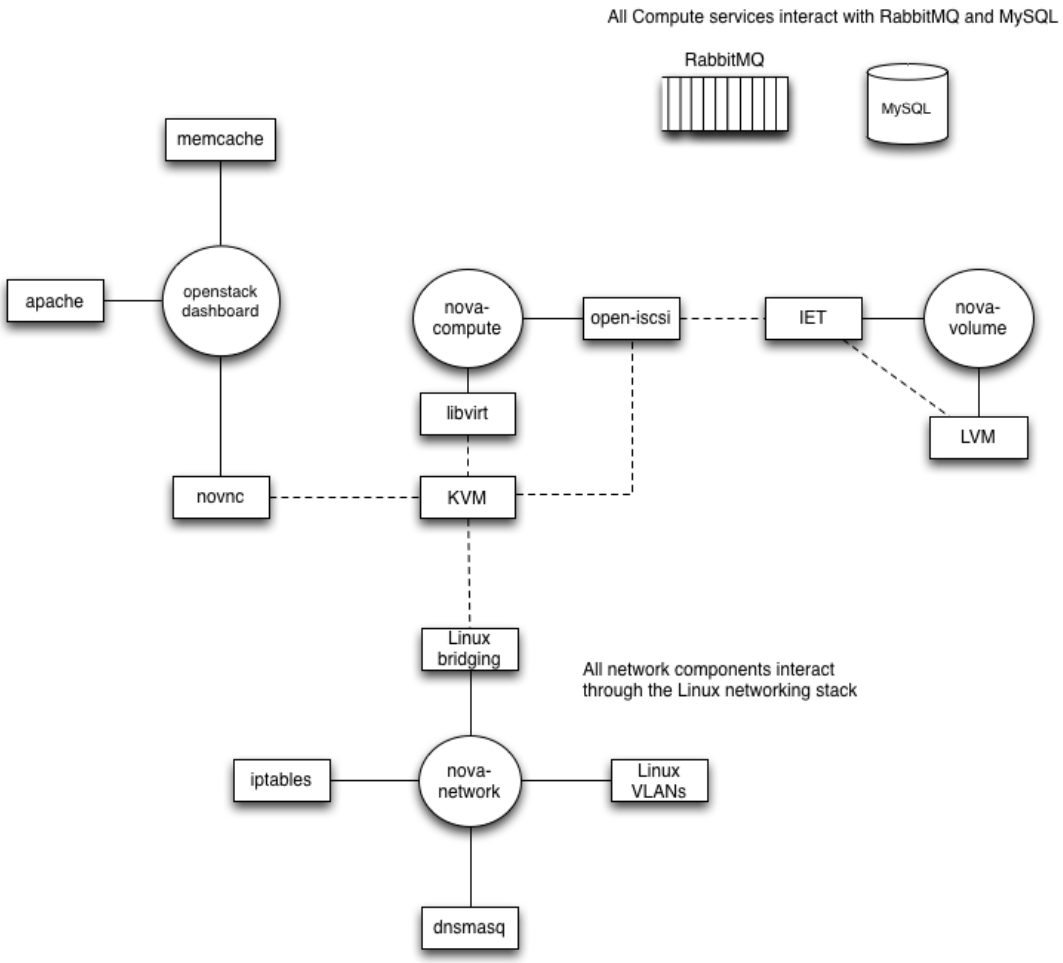

Figura 2.2: Schema informativo sulle tecnologie di Openstack

Nello schema i cerchi rappresentano servizi Linux facenti parte di Openstack, i rettangoli identificano servizi esterni a quest'ultimo, le linee continue mostrano l'interazione fra le due tipologie, le linee tratteggiate le interazioni fra servizi Linux.

L'intero sistema si appoggia di default su database SQL e su sistemi di messaggistica fra server come RabbitMQ in caso di Ubuntu, Qpid su distribuzioni REDHAT.

Le tecnologie utilizzate vengono riassunte e successivamente descritte in:

TECNOLOGIA	**IMPLEMENTAZIONE SUPPORTATA**
Messaggistica	**RabbitMQ**, Qpid
Virtualizzazione	xapi+XCP, xapi+XenServer, **libvirt+KVM**, libvirt+QEMU, libvirt+LXC, libvirt+VMWare
Database	**MySQL**, PostgreSQL, sqlite
Web server	**Apache**, Nginx
Session cache	**memcache**, any Django-supported database backend (e.g., MySQL, PostgreSQL, sqlite)
iSCSI back-end	LVM+IET, LVM+tgt, Xen Storage Manager, SAN (Solaris, HP, SolidFire), NexentaStor, NetApp, Ceph, Sheepdog

Proseguendo, un breve accenno ai volumi di memorizzazione più volte citati che vanno catalogati in due differenti tipologie.

Object storage – I quali contengono dati memorizzati da swift il componente utile all'archiviazione dei file degli utenti

Block storage (SAN) – I quali, paragonabili agli EBS di Amazon contengono i dati delle istanze come descritto precedentemente potendo memorizzare anche le immagini stesse di dischi virtuali. Sono collegati fisicamente al server mediante ATA o SCSI e acceduti in remoto mediante iSCSI. Quest'ultimo è un protocollo che permette di utilizzare dispositivi di memoria SCSI collegati al server. I blocchi sono quindi sviluppati mediante architettura SAN la quale permette di renderli disponibili a tutti i dispositivi in rete realizzando cio che serve ad un sistema cloud.

Incrociando quindi componenti moduli e tecnologie nel caso di studio affrontato:

il cloud controller gestisce la base del sistema ovvero contiene:

- MySQL con il quale crea e gestisce tutti i database di tutti i moduli, keystone per gli utenti, nova e glance per le macchine virtuali
- RabbitMQ con il quale gestisce le comunicazioni con gli host che devono ad esempio accedere ai database
- Modulo keystone che si occupa di tutta la parte di gestione ed autenticazione degli utenti
- Modulo Glance che gestisce la parte di archiviazione delle macchine virtuali che in seguito verranno richieste
- Modulo nova in particolare il servizio nova-network che gestisce la creazione e gestione delle macchine virtuali.

Il computer host invece contiene:

- RabbitMQ per poter comunicare col server controller
- Nova ed in particolare il servizio nova-compute che agendo sull'hypervisor esegue le macchine virtuali mentre nova-volume gestisce i volumi.

2.6 – REQUISITI

CONTROLLER - HARDWARE CONSIGLIATO:

- **Processore:** quad-core 64 bit x86 abilitato al supporto della virtualizzazione (*)
- **Memoria:** 12 GB (**)
- **Spazio su disco:** 30 GB (SATA - SAS - SSD)
- **Spazio di archiviazione:** 2 Dischi X 2TB (SATA) (***)
- **Interfaccia di rete:** 2 X 1 GB Network Interface Card (NIC) (****)

(*) Il controller funziona anche con processore a 32 bit, Non è possibile in questo caso avviare istanze di virtual machine a 64 bit
(**) Processore e memoria consigliati sono molto superiori ai requisiti minimi.
(***) Nel caso si intenda utilizzare il controller anche per archiviare volumi di macchine virtuali
(****) Due interfacce di rete sono fortemente consigliate. Il controller funziona anche con una singola scheda di rete. WARNING: In questo caso si possono riscontrare problemi di connettività

HOST - HARDWARE CONSIGLIATO:

- **Processore:** quad-core 64 bit x86 abilitato al supporto della virtualizzazione (*)
- **Memoria:** 32 GB (**)
- **Spazio su disco:** 30 GB (SATA - SAS - SSD)
- **Interfaccia di rete:** 2 X 1 GB Network Interface Card (NIC)

(*) Il controller funziona anche con processore a 32 bit, Non è possibile in questo caso avviare istanze di virtual machine a 64 bit
(**) Con 2GB RAM è comunque possibile avviare poche istanze di piccole dimensioni

REQUISITI SOFTWARE NECESSARI:

- **Sistema Operativo:** CentOS, Debian, Fedora, RHEL, Ubuntu
- **Database:** PostgreSQL, MySQL (*)
- **Network Time Protocol:** NTP (**)
- **Altro:** Python, pip, rabbitMQ-server

(*) SQLite necessario per utilizzare il servizio ObjectStorage di Openstack
(**) Controller e nodi necessitano di essere sincronizzati temporalmente.

Capitolo 3

CASO DI STUDIO

Il caso di studio affrontato si pone l'obiettivo di installare ed inizializzare Openstack creando un sistema di cloud computing all'interno della facoltà di Ingegneria nella sede di Cesena che consentirà, nel tempo, di effettuare operazioni di migrazione di macchine virtuali sia fra gli host della nuvola locale, sia con gli host della nuvola situata nella sede di Bologna. Tutto questo servirà ad approfondire le operazioni di ottimizzazione dinamica, trasparente all'utente, dei servizi utilizzati e della rete.

Il lavoro è articolato installando innanzitutto il sistema operativo sul server che funge da controller e sugli host, semi-automatizzando la procedura mediante file kickstart. In seguito si è poi passati all'installazione vera e propria di Openstack. Quest'ultima fase è stata effettuata secondo due modalità differenti, approcciandosi quindi in un primo momento all'installazione manuale di prova su una singola macchina utilizzando sorgenti installabili tramite python. In questo caso sono sorti numerosissimi problemi di compatibilità con il sistema ospitante risolti mediante la sistemazione di dipendenze errate fra i vari componenti. In seguito si è affrontata l'installazione sul server e sugli host mediante EPEL (Extra Package Enterprise Linux) repository, la raccolta di software ottimizzata per Linux che include distribuzioni RHEL. La procedura descritta di seguito è proprio quest'ultima, lasciando all'appendice la descrizione della prima modalità.

3.1 – ARCHITETTURA

I componenti principali del sistema possono essere suddivisi in tre macro categorie: elaboratori, rete, software.

ELABORATORI:

- **1 SERVER** - UTILIZZATO COME CLOUD CONTROLLER
- **6 PC** - UTILIZZATI COME HOST CHE ASSUMERANNO IL RUOLO DI DEPOSITO DELLE IMMAGINI DEI DISCHI ED ESECUZIONE DELLE MACCHINE VIRTUALI

RETE:

- **10.10.0.0/24** - RETE FISICA PRIVATA AL QUALE SONO COLLEGATI CONTROLLER ED HOST
- **10.20.0.0/24** - SOTTORETE VIRTUALE AL QUALE SI COLLEGANO LE VIRTUAL MACHINE
- **RETE PUBBLICA**

SOFTWARE:

- CENTOS 6.3
- OPENSTACK - ESSEX

Il lavoro quindi comincia dalla scelta del sistema operativo su cui si lavorerà.

La decisione verte appunto su CentOS, una distribuzione enterprise di Linux fornita da Red Hat compatibile con Openstack.

3.2 – IL SISTEMA OPERATIVO

Trattandosi di un paragrafo secondario viene descritto il sistema operativo semplicemente citando *[11]*.

“ CentOS (acronimo di Community enterprise Operating System) è un sistema operativo concepito in modo da fornire una piattaforma di classe enterprise per chiunque intenda utilizzare GNU/Linux con scopi professionali. Si tratta di una distribuzione Linux che deriva da Red Hat Enterprise Linux con la quale si impegna ad essere completamente compatibile. Per questo motivo CentOS può essere una soluzione ottimale per tutti coloro che vogliono ottenere velocemente un sistema GNU/Linux di classe superiore con al seguito innumerevoli vantaggi.

Red Hat Enterprise Linux è composta interamente da software libero, ma è resa disponibile in una forma usabile (come CD-ROM di binari) solo a pagamento. Come richiesto dalla GNU General Public License e dalle altre licenze, tutto il codice sorgente è reso disponibile pubblicamente dalla Red Hat. Gli sviluppatori di CentOS usano quindi questo codice per creare un prodotto molto simile a Red Hat Enterprise Linux rendendolo accessibile gratuitamente per il download e l'uso, senza in ogni caso il supporto offerto da Red Hat.

Sostanzialmente la differenza tra Red Hat e CentOS risiede principalmente nell'assenza di assistenza, il principale motivo per cui si paga, e il cambio del logo, dato che Red Hat è un trademark. Altre differenze seppur minimali risiedono nella garanzia ad esempio, in ogni caso esistono molte altre distribuzioni derivate dal codice sorgente di Red Hat.”

Avendo a disposizione più macchine risulta estremamente efficiente adottare un metodo semiautomatizzato per procedere all'installazione del sistema operativo. Per questo motivo verrà descritta l'installazione mediante file kickstart.

3.3 – INSTALLAZIONE KICKSTART

L’installazione kickstart è un metodo che permette l'installazione di sistemi operativi mediante un apposito file, appunto il kickstart. L'idea di base consiste nel creare il suddetto file con tutte le impostazioni necessarie richieste in fase di installazione del SO (Lingua, rete, dispositivi I/O, partizionamento disco..) in modo da evitare all’utente l’inserimento manuale di queste ultime a ripetizione. Questa tecnica risulta molto utile quindi nel caso di installazioni su più macchine che possono così essere eseguite parallelamente con un notevole risparmio di tempo.
La procedura si riassume nel rendere disponibile il file kickstart su un server in modo che possa essere letto dalle macchine automaticamente.
Nel caso particolare analizzato questa metodologia sarà doppiamente utilizzata:

- Installazione SO sul cloud controller, mediante file kickstart disponibile su server esterno
- Installazione SO sulle macchine host parallelamente mediante file kickstart opportunamente salvato e configurato sul controller.

Al primo avvio della macchina vergine (controller od host) è necessario quindi configurare l'interfaccia di rete in modo che si connetta ad un servizio DHCP ottenendo così un indirizzo IP e diventando in grado di raggiungere il file necessario.
Come già detto, nel caso specifico affrontato il cloud controller sarà abilitato a collegarsi ad un server esterno contenente un file kickstart preconfigurato. Una volta terminata la procedura di installazione verranno creati al suo interno tutti i file kickstart, ciascuno utile a configurare uno specifico host. Questi ultimi abiliteranno quindi la propria interfaccia di

rete mediante servizio DHCP fornito dal controller stesso, ed accederanno ai rispettivi file kickstart.

METODOLOGIA DI LAVORO:

- SERVER ESTERNO: CONFIGURAZIONE SERVIZIO DHCP E FILE KICKSTART UTILE AL CONTROLLER
- CONTROLLER: INSTALLAZIONE SISTEMA OPERATIVO
- CONTROLLER: CONFIGURAZIONE DHCP
- CONTROLLER: CREAZIONE FILE KICKSTART PER MACCHINE CLIENTE
- HOST: INSTALLAZIONE SISTEMA OPERATIVO

3.3.1 – CLOUD CONTROLLER

Per poter installare il sistema operativo sul cloud controller è necessario preventivamente configurare un server esterno che possa fornire un indirizzo IP statico mediante servizio DHCP e un file kickstart .

SERVER ESTERNO: CONFIGURAZIONE SERVIZIO DHCP E FILE KICKSTART UTILE AL CONTROLLER

Ottenuto l'indirizzo MAC della scheda di rete del cloud controller dalle opzioni accessibili in fase di boot, viene configurato il servizio DHCP del server esterno aggiornando come segue il file dhcpd.conf.

```
$ vim /etc/dhcpd.conf
```

Aggiungere:

```
ddns-update-style none;

        subnet <indirizzo sottorete> netmask <indirizzo netmask>{
        …
        host server1 {
                hardware ethernet <indirizzo MAC cloud controller>
                fixed-address <indirizzo IP>
        }
}
```

In questo modo ogni volta che la scheda di rete relativa all' indirizzo MAC inserito si connette al servizio DHCP del server si vedrà assegnato l'indirizzo IP specificato.

Nel caso di studio affrontato:

```
ddns-update-style none;

        subnet 10.10.0.0 netmask 255.255.255.0 {
        …
        host server1 {
                hardware ethernet <indirizzo MAC cloud
controller>
                fixed-address 10.10.0.1
        }
}
```

Il cloud controller otterrà così l'IP 10.10.0.1

CLOUD CONTROLLER: INSTALLAZIONE SISTEMA OPERATIVO

Una volta ottenuto l'IP statico è possibile passare alla fase di installazione del SO. Assodato lo scopo e l'utilizzo del kickstart file per prima cosa è necessario modificarlo secondo le proprie esigenze.

CentOS è un sistema operativo che salva il suddetto file in:

/root/anaconda-ks.cfg

quindi non resta che accedere al server esterno, replicare il file all'interno del percorso

/var/html/www

e modificarlo.

Nel nostro caso viene chiamato ccloud01.cfg.

KICKSTART FILE: STRUTTURA

(In appendice è possibile trovare un esempio di file kickstart)

Ora è possibile iniziare l'installazione.

Dopo aver inserito il cd alla comparsa del menù premere TAB inserendo così il comando "ks" seguito dall'indirizzo del Server in cui è contenuto il file kickstart.

Esempio:

"ks=http://<indirizzo_server_esterno>/ccloud01.cfg"

In questo modo il programma di installazione cerca il kickstart file sul server esterno nel percorso inserito usando il protocollo DHCP per configurare la scheda di rete. L'installazione può partire.

CONTROLLER: CONFIGURAZIONE DHCP

Terminata l'installazione del SO sul server è necessario ripetere la configurazione in modo da renderlo in grado di fornire un servizio DHCP utile in questo caso agli host. Questi ultimi infatti si collegheranno al

controller per essere in grado di accedere alla rete ed ottenere il kickstart file.

La procedura, come nella situazione precedente, prevede la modifica del file */etc/dhcpd.conf.*

In questo caso tuttavia non è necessario associare l'indirizzo MAC delle interfacce di rete in stato di collegamento all'IP da assegnargli visto che la rete di host è interna e l'indirizzo verrà impostato direttamente in fase di installazione mediante le opzioni specificate nel kickstart file. L'unica necessità risiede nella specifica della rete che verrà utilizzata.

Viene così installato il servizio dhcp:

```
$ yum install dhcp
```

ed aprendo il file:

```
$ vim /etc/dhcpd.conf
```

si noterà che è vuoto a meno di una stringa. Il file stesso infatti suggerisce il percorso di un file esempio che verrà quindi copiato e modificato con le informazioni utili:

Subnet 10.10.0.0 netmask 255.255.255.0 {

range 10.10.0.100 10.10.0.120; [range di indirizzi]

option routers 10.10.0.1 [impostare il router]

option domain name server 137... [DNS]

}

Si avvia il servizio: `/etc/init.d/dhcpd start`

E si imposta:

- forwarding:
  - Aprire il file */etc/sysctl.conf* e impostare net.ipv4 = 1
- NAT:

- modificando le tabelle di NAT secondo la necessità

Infine viene avviato il servizio di server web:
/etc/init.d/httpd start

CONTROLLER: CREAZIONE FILE KICKSTART PER MACCHINE CLIENTE

Il kickstart file contenuto all'interno del server è il file da replicare e modificare in modo che le macchine cliente possano utilizzarlo. Non resta che crearne tanti quante macchine si desidera installare adattandoli secondo le esigenze.

Nel caso esaminato copiamo piu volte il file kickstart del controller */root/anaconda-ks.cfg* in */var/www/html/* rendendo così tutte le copie accessibili e rinominandole in ccloud02,ccloud03...

La modifica del file riguarda essenzialmente le impostazioni dell'interfaccia eth0. Viene quindi settato l'IP desiderato, viene aggiunto il router di default e dns semplicemente copiando la sintassi dall'interfaccia eth2 posta più in basso e viene cambiato l'host name.

Le interfacce diverse da eth0 vengono poi cancellate dato che inutilizzate mentre l'ultimo passaggio riguarda la password.

Viene impostata la password di root all'interno del server:

passwd <password>

e ne viene letta la codifica all'interno di

/etc/shadow.

Questa viene copiata e incollata all'interno di tutti i file kickstart in modo da impostare la stessa password utente root per ogni macchina compreso appunto il controller.

Infine, per rendere ogni file accessibile dalla rete impostiamo i permes*si chmod 644 *.*

(In appendice è possibile trovare un esempio di file kickstart)

3.3.2 – COMPUTER HOST

- Controllare che sul controller siano attivi i servizi di rete e DHCP

  Accedere quindi al server via ssh e procedendo come segue:

```
$ ssh root@10.10.0.1
$ /etc/init.d/dhcpd start
$ /etc/init.d/httpd start
```

- Entrare poi nel BIOS dell'host da inizializzare (ESC+computer setup) :
  - Impostare l' avvio da CD
  - Abilitare virtualizzazione (Security → System Security → Virtualization Tecnology)
  - Abilitare avvio da rete (Security → Network Boot)

- Riavviare col CD inserito

Alla comparsa del menù di installazione premere TAB, e scrivere dopo uno spazio:

ks=http://ip_server/ccloud02.cfg

(ccloud02.cfg è il file kickstart relativo al primo computer host che vogliamo inizializzare)

- Selezionare *"Use all Space"* e in basso *"Review and modify partitioning layout"* per gestire le partizioni.
- Cancellare la partizione *"home"*
- Creare una partizione per il sistema, una di 100 GB per memorizzare le immagini dei dischi virtuali impostando la tipologia in LVM e una partizione di SWAP da 200 MB.

** Su una delle 6 macchine si è presentato un errore. Il disco non veniva letto causa precedenti impostazioni RAID. E' stato sufficiente da console di ripristino lanciare il comando*: dmraid -r -E /dev/sda*

3.4 – INSTALLAZIONE OPENSTACK

Prima di cominciare l'installazione vengono definiti alcuni punti imprescindibili:

- Il cloud controller è delegato ad eseguire sia processi Linux di supporto come MySQL e RabbitMQ sia i principali processi dei moduli keystone (keystone-all), glance (glance-api, glance-registry) e nova (nova-api, nova-network, nova-scheduler). Gli host si occupano invece di eseguire i processi nova-compute e nova-volume per la gestione delle macchine virtuali e dei volumi di memorizzazione.
- Gli host possono essere collegati ai volumi logici di una partizione gestita mediante LVM in un gruppo LVM chiamato nova-volumes
- Assicurarsi che il server non abbia problemi con localhost al fine di non incontrare problemi con RabbitMQ, sistema di messaggistica di default della distribuzione Ubuntu. Qpid per quel che riguarda Fedora. Nel caso di studio descritto è stato utilizzato in ogni caso RabbitMQ.
- Verrà creato un alias sull'interfaccia principale per evitare conflitti di IP fra macchine fisiche e virtual machine. Da notare la possibilità, alternativa, di creare due reti completamente separate
- 10.10.0.0/24 è la rete privata resa disponibile sull'interfaccia eth0.
- 10.20.0.0/24 è la rete di virtual machine resa disponibile sull'alias eth0:virt
- br100 è l'interfaccia virtuale di comunicazione con la rete di virtual machine
- Gli IP delle macchine virtuali sono assegnati dal controller mediante FlatDHCP
- KVM è utilizzato come Hypervisor.

3.4.1 – CLOUD CONTROLLER

Come descritto precedentemente verranno installati sul server i componenti Keystone, Glance e Nova. Il primo gestirà l'autenticazione degli utenti e dei servizi, il secondo si occuperà di tutta la gestione dei dischi virtuali salvati sugli host, il terzo eseguirà solamente la creazione della rete virtuale e la schedulazione delle virtual machine sui vari host ai quali è riservata la parte computazionale.

3.4.1.1 – CARATTERISTICHE TECNICHE

Verrà affrontata l'installazione di Openstack su cloud controller rispondente alla seguente configurazione:

- Hardware:

- Processore: *Intel Xeon E31230 3.20 GHz quadCore (4 CPU fisiche + 4 CPU virtualizzate)*
- RAM: *8GB*
- HD: *500 GB*
- Interfaccia di rete: *eth0, eth0:virt, eth2*

eth0 -> Rete interna: 10.10.0.0/24 - IP Controller: 10.10.0.1
eth0:virt -> Rete virtuale: 10.20.0.0/24 – IP Controller: 10.20.0.1
eth2 -> Rete pubblica

- Software:

- SO: *CentOS*
- Database: *MySQL*
- Network Time Protocol: *NTP*
- Openstack: *Essex - ultima versione disponibile*
- Altro: *Python, pip, rabbitMQ, Apache*

3.4.1.2 - CONTROLLI PRE-INSTALLAZIONE

- **Controllo dell'abilitazione del processore al supporto della virtualizzazione:** `$ lsmod |grep kvm`

  Vengono mostrati i file "kvm_intel" e "kvm". La virtualizzazione per processori Intel è attivata

- **Attivare EPEL repository mediante:**

```
$ sudo rpm –Uvh
http://download.fedoraproject.org/pub/epel/6/i386/
epel-release-6-7.noarch.rpm
```

- **Installazione Network Time Protocol (NTP)**

```
$ sudo yum install -y ntp
$ sudo chkconfig ntpd on
```

- **Installazione MySQL:**

```
$ yum install mysql mysql-devel mysql-server MySQL-python
$ sed -i 's/127.0.0.1/0.0.0.0/g' /etc/my.cnf
$ chkconfig --levels 235 mysqld on
```

  Impostazione password:

```
mysqladmin -u root password <PASSWORD>(*)
```

  (*) Selezione password root : Se è necessario inserire caratteri speciali come "$" è necessario utilizzare il carattere di escape "\".

  ESEMPIO (Password = cloud$):

  "mysqladmin –u root password cloud\$"

==========

Se la password crea problemi (ad esempio usando $) potrebbe essere necessario disinstallare mysql:
- rpm -aq | grep mysql | xargs rpm -e --nodeps --allmatches #removes existing MySQL- rm -rf /var/lib/mysql

==========

- **Installazione sistema di messaggistica rabbitMQ**

  $ yum install rabbitmq-server

  $ sudo chkconfig rabbitmq-server on

  Modificare la porta su cui offrire il servizio dalla 5672 alla 5673 creando se non esiste e modificando il file /etc/rabbitmq/rabbitmq.config in questo modo:

```
[
   {mnesia, [{dump_log_write_threshold, 1000}]},
   {rabbit, [{tcp_listeners, [5673]}]}
]
```

  Questo è necessario perchè di default centOS utilizza il sistema di messaggistica qpid in ascolto sulla porta 5672.

- **Installare servizi iscsi**

  Initiator:

  $ yum install iscsi-initiator-utils

  iscsi-target :

  $ yum install netbsd-iscsi

- **Creazione alias**

  Copiare rinominandolo il file */etc/sysconfig/network-scripts/ifcfg-eth0*

  $ cp ifcfg-eth0 ifcfg-eth0:virt

  Modificare il file appena creato:

```
...

DEVICE=eth0:virt
```

```
IPADDR=<IP_DESIDERATO> (Nel caso 10.20.0.1)

...
```

Riavviare i servizi di rete e abilitare l'interfaccia

```
$ service network restart
$ ifup eth0:virt up
```

Potrebbe essere necessario ricollegarsi all'interfaccia eth0

- **Avviare tutti i servizi:**

```
$ /etc/init.d/httpd start

$ sudo chkconfig httpd on

$ sudo service ntpd start

$ sudo chkconfig ntpd on

/etc/init.d/mysqld start

$ sudo service rabbitmq-server start

$ /etc/init.d/iscsi start

$ service  netbsd-iscsi start

$ /etc/init.d/tgtd restart
```

3.4.1.3 - INIZIALIZZAZIONE DATABASE

E' necessario ora creare un database per ogni modulo di Openstack e relativi amministratore e password.

(NOTA: Per maggior semplicità è possibile installare phpmyadmin, software in grado di gestire datatbase in modalità grafica. Vedi appendice)

```
$ mysql -u root -p
```

Keystone (user: keystoneAdmin - password: …) :

- Crea un nuovo database per keystone:

```
CREATE DATABASE keystone;
```

- Crea un nuovo utente che gestisca il database keystone con tutti i privilegi

```
GRANT ALL ON keystoneAdmin.* TO 'keystone'@'%'
IDENTIFIED BY '<PASSWORD>';
```

Nova (user: novaAdmin - password: …):

- Crea un nuovo database per nova:

```
CREATE DATABASE nova;
```

- Crea un nuovo utente che gestisca il database nova con tutti i privilegi

```
GRANT ALL ON novaAdmin.* TO 'nova'@'%' IDENTIFIED
BY '<PASSWORD>';
```

Glance (user: glanceAdmin - password: …):

- Crea un nuovo database per nova:

```
CREATE DATABASE glance;
```

- Crea un nuovo utente che gestisca il database glance con tutti i privilegi

```
GRANT ALL ON glanceAdmin.* TO 'glance'@'%'
IDENTIFIED BY '<PASSWORD>';
```

horizon (user: horizonAdmin - password=…)
- Crea un nuovo database per horizon:

```
CREATE DATABASE horizon;
```

- Crea un nuovo utente che gestisca il database horizon con tutti i privilegi

```
GRANT ALL ON horizonAdmin.* TO 'horizon'@'%'
IDENTIFIED BY '<PASSWORD>';
```

-Uscire: "\q"

3.4.1.4 –KEYSTONE

- **Installazione:**

```
$ yum install openstack-utils openstack-keystone
python-keystoneclient
```

- **Configurazione:**

- Modificare /etc/keystone/keystone.conf come mostrato in appendice. In particolare modificare il campo “connection” con

```
connection = mysql://<user_database>:<password_database>@<IP>/keystone
```

completare il campo “admin_token” per l’autenticazione dei servizi a keystone scegliendo una parola di sicurezza e controllare che sia impostato:

```
[identity]
driver = keystone.identity.backends.sql.Identity
```

Aggiungere in fondo al file:

```
[filter:s3_extension]
paste.filter_factory = keystone.contrib.s3:S3Extension.factory
```

e modificare

```
[pipeline:admin_api]
pipeline = token_auth admin_token_auth xml_body json_body debug
ec2_extension crud_extension admin_service
```

con

```
[pipeline:admin_api]
pipeline = token_auth admin_token_auth xml_body json_body debug
ec2_extension s3_extension crud_extension admin_service
```

- Esportare le variabili d’ambiente ed aggiungerle eventualmente al file /bashrc per averle a disposizione anche dopo il riavvio:

```
$ export ADMIN_TOKEN= <admin_token>
$ export SERVICE_ENDPOINT="http://localhost:35357/v2.0"
```

- **Avviare Keystone ed impostarne l’autostart:**

```
$ sudo service keystone start && sudo chkconfig
keystone on
```

A causa di un malfunzionamento, probabilmente a livello di permessi sui file, nel caso di studio affrontato diventa necessario avviare il singolo processo mediante:

```
$ keystone-all &>/dev/null &
$ sudo chkconfig keystone-all on
```

- **Sincronizzare il database:**

```
$ keystone-manage db_sync
```

- **CREAZIONE TENANT**

```
$ keystone tenant-create --name admin
$ keystone tenant-create --name service
```

- **CREAZIONE UTENTI**

```
$ keystone user-create --name admin --pass admin
$ keystone user-create --name nova --pass nova
$ keystone user-create --name glance --pass glance
$ keystone user-create --name swift --pass swift
```

- **CREAZIONE RUOLI**

```
$ keystone role-create --name admin
$ keystone role-create --name Member
```

Per visualizzare le informazioni e gli id appena creati:

```
$ keystone tenant-list
$ keystone user-list
$ keystone role-list
```

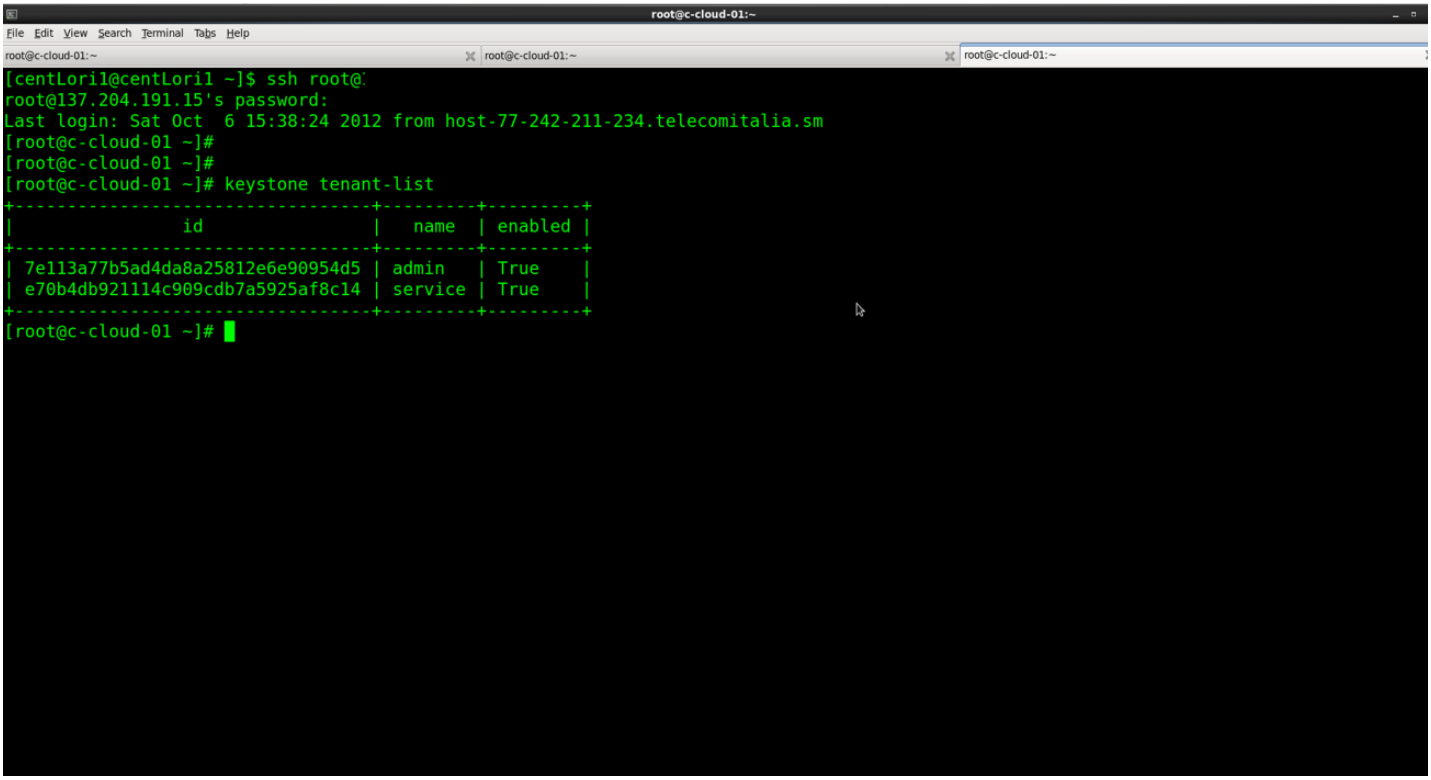

Figura 3.1: Screenshot - Lista di tenant.Comando keystone tenant-list

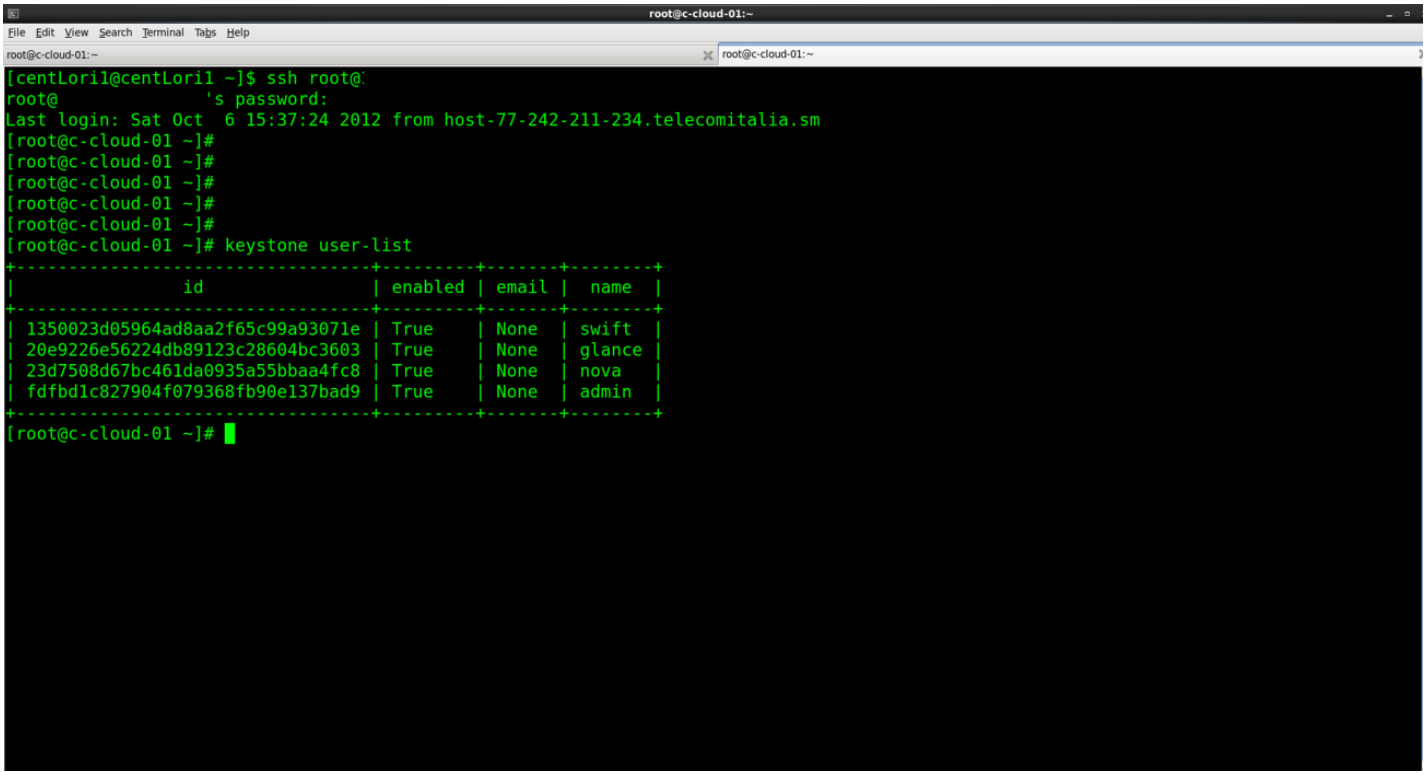

Figura 3.2: Screenshot - Lista di user.Comando keystone user-list

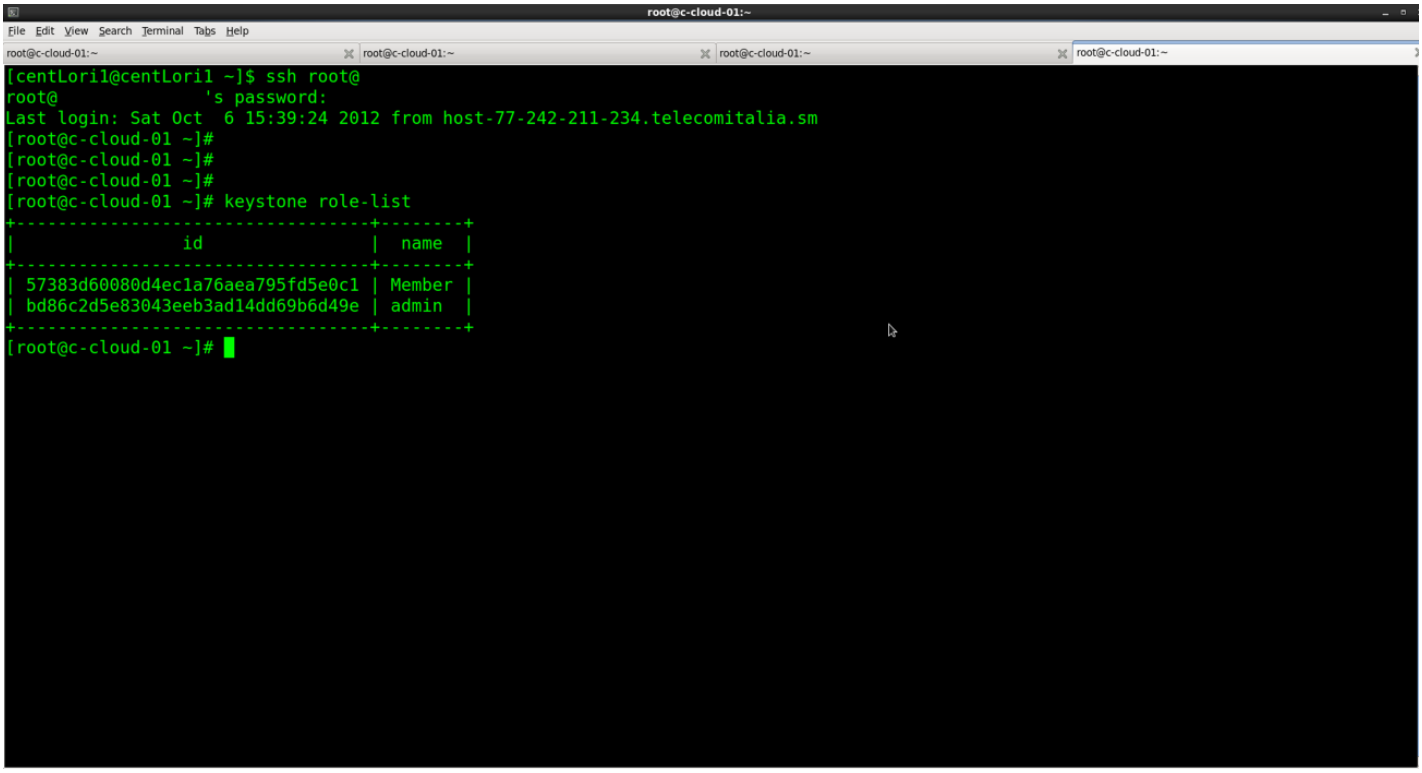

Figura 3.3: Screenshot – Lista di ruoli.Comando keystone role-list

- **ASSEGNAMENTO RUOLI-UTENTI-TENANT**

```
keystone  --token <admin_token> --endpoint
http://<IP>:35357/v2.0 user-role-add --user_id
<user_id> --role_id <role_id> --tenant_id
<tenant_id>
```

Utilizzare il comando soprastante per le impostare le seguenti relazioni:
- Ruolo: admin – User: admin – Tenant: admin
- Ruolo: admin – User: nova – Tenant: service
- Ruolo: admin – User: glance – Tenant: service
- Ruolo: admin – User: swift – Tenant: service
- Ruolo: Member – User: admin – Tenant: admin

- **CREAZIONE SERVIZI PER OGNI MODULO**

nova:

```
$ keystone service-create --name nova --type
compute --description 'OpenStack Compute Service'
```

nova-volume:

```
$ keystone service-create --name volume --type
volume --description 'OpenStack Volume Service'
```

glance:

```
$ keystone service-create --name glance --type
image --description 'OpenStack Image Service'
```

swift: (Non utilizzato nel caso di studio)

```
$ keystone service-create --name swift --type
object-store --description 'OpenStack Storage
Service'
```

keystone:

```
$ keystone service-create --name keystone --type
identity --description 'OpenStack Identity
Service'
```

ec2:

```
$ keystone service-create --name ec2 --type ec2 --
description 'EC2 Service'
```

- **CREAZIONE ENDPOINT**

```
$ keystone endpoint-create --region region_name --
service_id service_id --publicurl public_url --
adminurl admin_url --internalurl internal_url
```

Utilizzare il comando soprastante per creare endpoint di:

- nova-compute
- nova-volume
- glance
- swift (non utilizzato nel caso di studio)
- keystone
- ec2

Un esempio è disponibile all'indirizzo *[12]* .

Visualizzare gli endpoint creati:

```
$ keystone endpoint-list
```

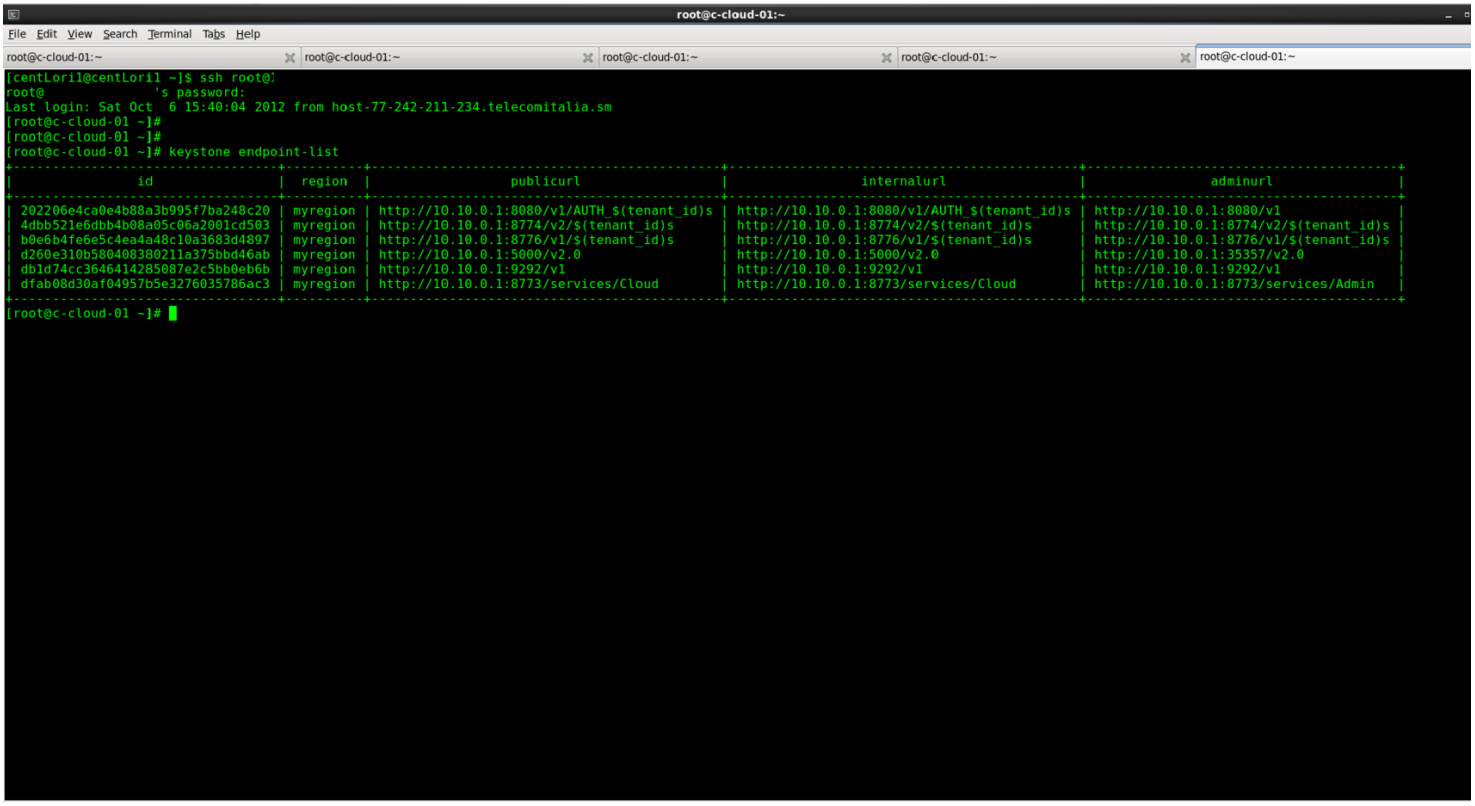

Figura 3.4: Screenshot – Lista di ruoli.Comando keystone role-list

3.4.1.5 – GLANCE

- **Installazione**

```
$ sudo yum install openstack-nova openstack-glance
```

- **Configurazione**

Modificare i seguenti file come in appendice.

In particolare /etc/glance/glance-api-paste.ini

```
[filter:authtoken]
admin_tenant_name = service
admin_user = glance
admin_password = glance
```

/etc/glance/glance-api.conf aggiungere:

```
[paste_deploy]
flavor = keystone
```

/etc/glance/glance-registry.conf inserisci:

```
[paste_deploy]
flavor = keystone
```

/etc/glance/glance-registry-paste.ini aggiorna:

```
[filter:authtoken]
admin_tenant_name = service
admin_user = glance
admin_password = glance
```

/etc/glance/glance-registry.conf aggiorna:

```
sql_connection =
mysql://<user_database>:<password_database>@<IP>/glance

Esempio:

sql_connection = mysql://glanceAdmin:<password_database>@10.10.0.1/glance
```

- **Avviare i servizi**

```
$ service openstack-glance-api start
$ service openstack-glance-registry start
$ sudo chkconfig glance-api on
$ sudo chkconfig glance-registry on
```

Anche in questo caso per avviare i processi vengono utilizzati i comandi manuali non necessari normalmente.

```
$ glance-api &>/dev/null &
$ glance-registry &>/dev/null &
$ sudo chkconfig glance-api on
$ sudo chkconfig glance-registry on
```

- **Sincronizzare i database**

```
glance-manage db_sync
```

- **Testare il servizio**

Caricare un'immagine di prova:

```
$ wget https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-disk.img

$ name=cirros-0.3-x86_64

$ image=cirros-0.3.0-x86_64-disk.img

$ glance add name=$name is_public=true container_format=bare disk_format=qcow2 < $image
```

Verificare che l'immagine sia stata caricata correttamente

```
$ glance index
```

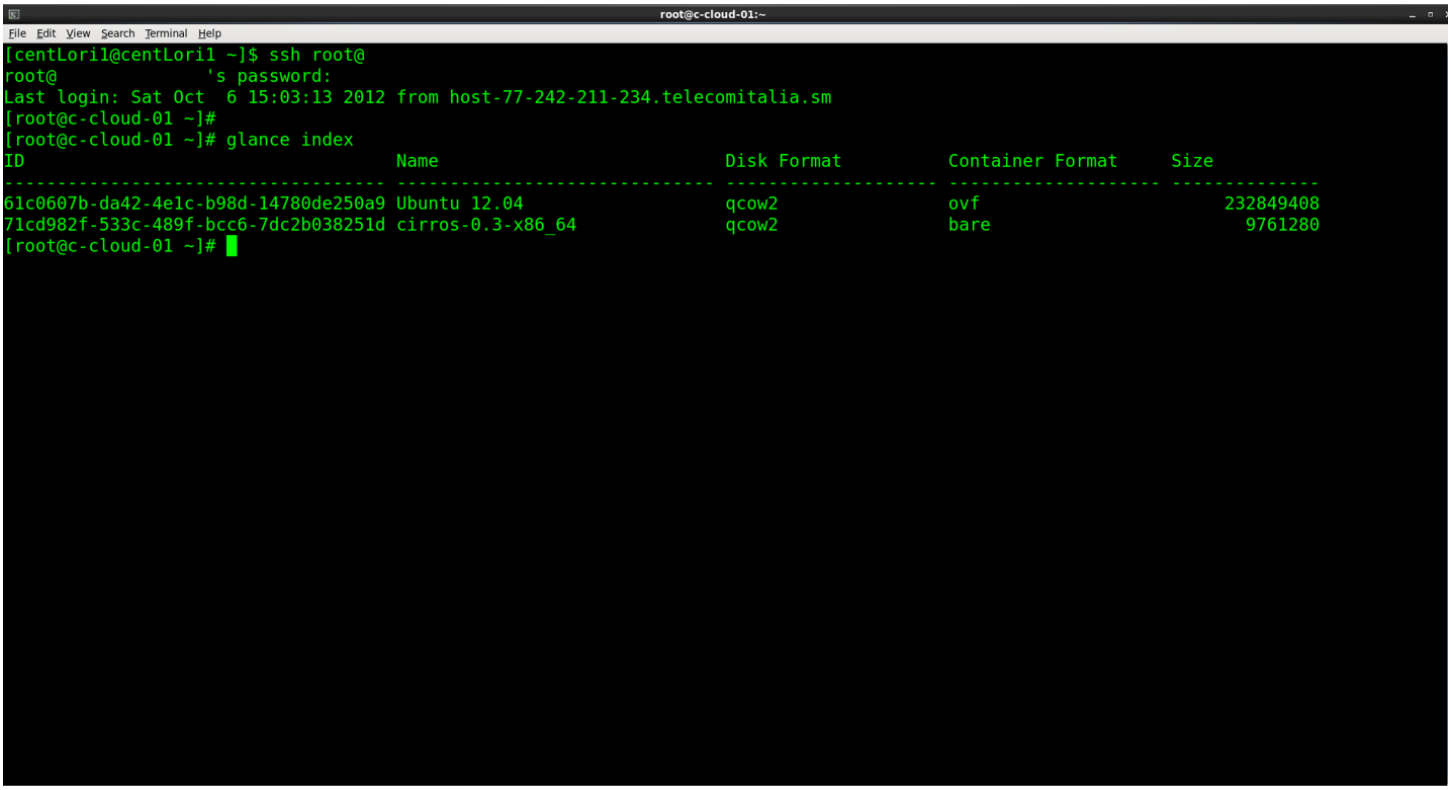

Figura 3.5: Screenshot – Lista di immagini archiviate.Comando glance index

3.4.1.6 – NOVA

Prima di tutto vengono installate le librerie utili all'uso del bridge con la creazione di quest'ultimo.

```
$ yum install bridge-utils
$ sudo brctl addbr br100
$ sudo /etc/init.d/networking restart
```

==========

Solamente nel caso di centos 6.3:

```
$ sudo yum install dnsmasq-utils
```

==========

- **Installazione**

```
$ sudo yum install openstack-utils memcached qpid-cpp-server
```

```
$ sudo yum install openstack-nova
```

- **Configurazione**

Il file di configurazione di nova è il più importante fra tutti.

Modificare quindi /etc/nova/nova.conf.
Avendone visti più di una tipologia viene riportato in appendice un esempio semplificato e funzionante.
Di particolare rilievo le opzioni:

- `sql_connection=mysql://<user_db>:<user_password>@<IP>/nova`
- `fixed_range` = range di indirizzi della rete virtuale
- `flat_interface` = interfaccia alla rete locale

Modificare il file /etc/nova/api-paste.ini

```
admin_tenant_name = service
admin_user = nova
admin_password = nova
```

In /etc/qpidd.conf. impostare "auth=no".

Creare il gruppo nova:

```
$ groupadd nova
```

Infine assegnare la directory di configurazione al gruppo nova ed impostarne i permessi:

```
$ chown -R root:nova /etc/nova
$ chmod 644 /etc/nova/nova.conf
```

- **Sincronizzazione database**

```
$ nova-manage db sync
```

- **Creazione della rete per le virtual machine**

```
nova-manage network create openstackVNet --
fixed_range_v4=10.20.0.0/24 --num_networks=1  --
bridge_interface=br100 --network_size=256
```

** Controllare nel database che la rete sia identificata dall'ID = 1 altrimenti non funziona

- **Avviare i servizi**

```
$ for svc in api objectstore compute network
volume scheduler cert; do sudo service openstack-
nova-$svc start ; sudo chkconfig openstack-nova-
$svc on ;  done
```

Anche in questo caso vengono avviati i processi singolarmente per problemi di permessi e viene impostato l'autostart:

```
$ nova-api &>/dev/null &
$ nova-network &>/dev/null &
$ nova-scheduler &>/dev/null &
$ nova-compute &>/dev/null &
$ sudo chkconfig nova-api on
$ sudo chkconfig nova-network on
$ sudo chkconfig nova-scheduler on
```

- **Controllare l'avvio dei servizi (verificare: state :-))**

```
$ nova-manage service list
```

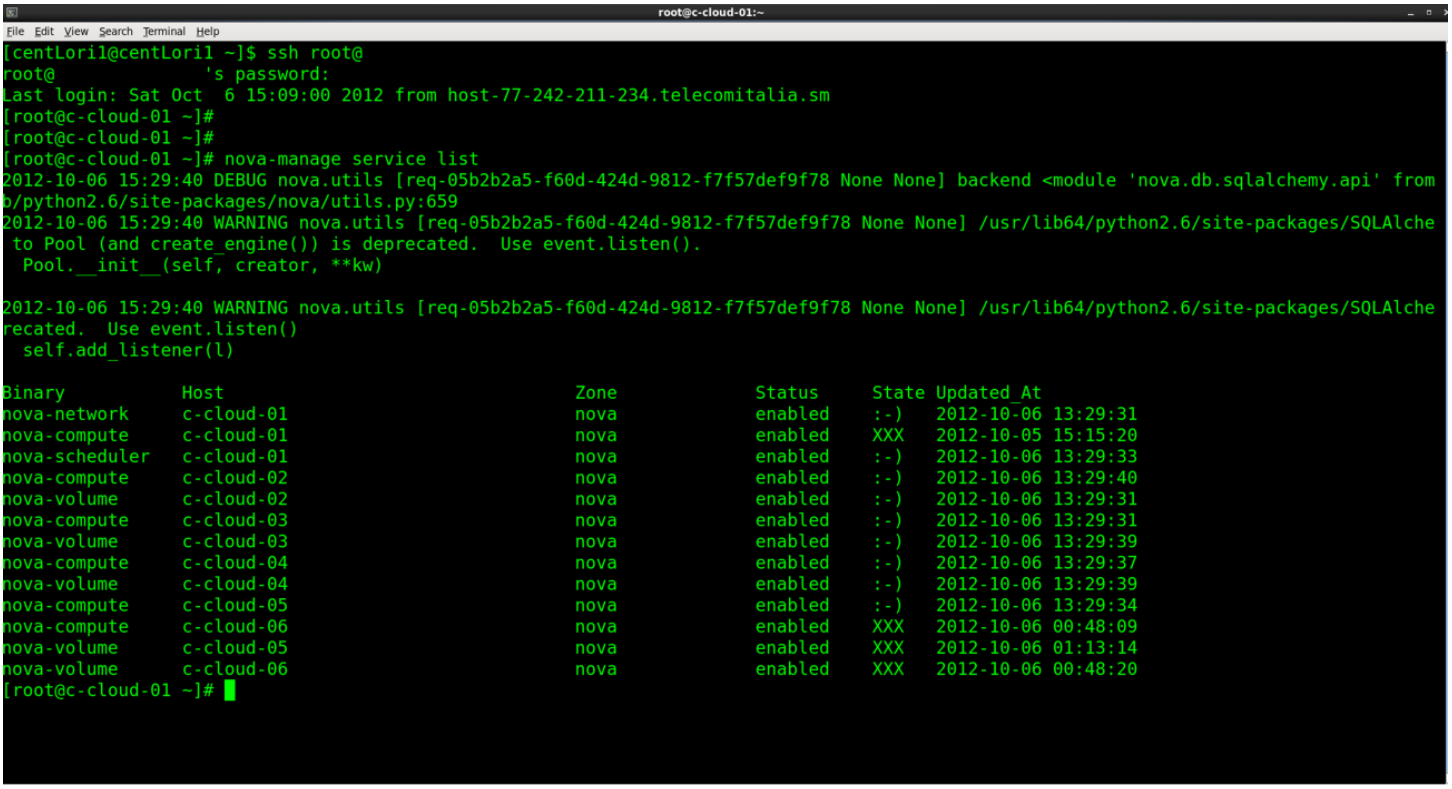

Figura 3.6: Screenshot – Lista di servizi avviati.Comando nova-manage service list

- **Problemi**

In caso di problemi aggiungere in /etc/nova/nova.conf
`dhcpbridge=/usr/bin/nova-dhcpbridge`

Potrebbe essere necessario creare le cartelle specificate in nova.conf

es. `$ mkdir /var/lock/nova`

ed installare:

```
$ yum install lvm2
$ yum install qemu-kvm qemu-img
$ yum install virt-manager libvirt libvirt-python
python-virtinst libvirt-client
```

3.4.1.7 – HORIZON

- **Installazione:**

```
yum install -y memcached mod-wsgi openstack-dashboard
```

- **Configurazione:**

Modificare /usr/share/openstack-dashboard/openstack_dashboard/local/local_settings aggiungendo:

```
SECRET_KEY = “<admin_token>”
OPENSTACK_ADMIN_TOKEN = "<admin_token>"

DATABASES = {
'default': {
'ENGINE': 'django.db.backends.mysql',
'NAME': 'horizon',
'USER': '<user_database>',
'PASSWORD': ‘<password_database'>,
'HOST': '<IP_database',
'default-character-set': 'utf8'
},
}
```

- **Sincronizzare Database:**

```
/usr/share/openstack-dashboard/manage.py syncdb
```

- **Avviare il servizio:**

```
/usr/share/openstack-dashboard/manage.py runserver 0.0.0.0:8000
```

3.4.2 – COMPUTER HOST

Data l'architettura selezionata i computer host si collegano al controller per autenticarsi ma eseguono solamente i processi nova-compute e nova-volume per gestire i volumi in cui salvare i dati persistenti relativi alle istanze di macchine virtuali ed effettuare la parte effettiva della computazione.

3.4.2.1 – CARATTERISTICHE TECNICHE

Verrà affrontata l'installazione di Openstack riferita alla seguente configurazione:

- Hardware:

- Processore: Intel pentium G620
- RAM: 4GB DDR3
- HD: 250 GB
- Interfaccia di rete: *eth0, eth0:virt, br100*

eth0 -> Rete interna: 10.10.0.0/24

eth0:virt -> Rete virtuale: 10.20.0.0/24

br100 -> Rete virtuale

Software:

- SO: *CentOS*
- Network Time Protocol: *NTP*
- Openstack*: Essex - ultima versione disponibile*
- Altro*: Python, pip, rabbitMQ,Apache*

3.4.2.2 – CONTROLLI PRE-INSTALLAZIONE

- **Controllo dell'abilitazione del processore al supporto della virtualizzazione:**

  ```
  $ lsmod |grep kvm
  ```

  Vengono mostrati i file "kvm_intel" e "kvm". La virtualizzazione per processori Intel è attivata

- **Attivare EPEL repository mediante:**

  ```
  $ sudo rpm –Uvh
  http://download.fedoraproject.org/pub/epel/6/i386/
  epel-release-6-7.noarch.rpm
  ```

- **Installazione Network Time Protocol (NTP)**

  ```
  sudo yum install -y ntp
  sudo chkconfig ntpd on
  ```

- **Installazione sistema di messaggistica rabbitMQ**

  ```
  yum install rabbitmq-server
  sudo chkconfig rabbitmq-server on
  ```

  Modificare la porta su cui offrire il servizio dalla 5672 alla 5673 creando se non esiste e modificando il file /etc/rabbitmq/rabbitmq.config in questo modo:

  ```
  [
     {mnesia, [{dump_log_write_threshold, 1000}]},
     {rabbit, [{tcp_listeners, [5673]}]}
  ]
  ```

Questo è necessario perchè di default centOS utilizza il sistema di messaggistica qpid in ascolto sulla porta 5672.

- **Installare servizi iscsi**

  Initiator:

  ```
  $ yum install iscsi-initiator-utils
  ```

  iscsi-target :

  ```
  $ yum install netbsd-iscsi
  ```

- **Creazione alias**

  Copiare rinominandolo il file */etc/sysconfig/network-scripts/ifcfg-eth0*

  ```
  $ cp ifcfg-eth0 ifcfg-eth0:virt
  ```

  Modificare il file appena creato:

  ```
  ...

  DEVICE=eth0:virt

  IPADDR=<IP_DESIDERATO> (Nel caso 10.20.0.X)

  ...
  ```

  Riavviare i servizi di rete e abilitare l'interfaccia

  ```
  $ service network restart
  $ ifup eth0:virt up
  ```

  Potrebbe essere necessario ricollegarsi all'interfaccia eth0

- **Esportare le variabili di ambiente**

  ```
  export OS_TENANT_NAME=admin
  export OS_USERNAME=admin
  ```

```
export OS_PASSWORD=admin
export OS_AUTH_URL=http://10.10.0.1:5000/v2.0/
```

Riportarle anche all'interno di /.bashrc

- **Avviare tutti i servizi:**

```
$ /etc/init.d/httpd start
$ sudo chkconfig httpd on
$ sudo service ntpd start
$ sudo chkconfig ntpd on
$ sudo service rabbitmq-server start
$ sudo chkconfig rabbitmq-server on
$ /etc/init.d/iscsi start
$ service  netbsd-iscsi start
$ /etc/init.d/tgtd restart
```

3.4.2.3 – NOVA

Prima di tutto vengono installate le librerie utili all'uso del bridge con la creazione di quest'ultimo.

```
$ yum install bridge-utils
$ sudo brctl addbr br100
$ sudo /etc/init.d/networking restart
```

==========

Solamente nel caso di centos 6.3:

```
$ sudo yum install dnsmasq-utils
```

==========

- **Installazione**

```
$ sudo yum install openstack-utils memcached qpid-cpp-server
```

```
$ sudo yum install openstack-nova
```

- **Configurazione**

Il file di configurazione di nova è il più importante fra tutti.

Modificare quindi /etc/nova/nova.conf semplicemente importandolo dal cloud controller e cambiando:

  - my_ip: inserire l'IP dell'host
  - tutte le impostazioni rigurdanti vnc immettendo l'IP dell'host

Importare dal cloud controller il file /etc/nova/api-paste.ini

In /etc/qpidd.conf. impostare "auth=no".

- **Creare il gruppo di volumi:**

Mostrare le partizioni:

```
$ fdisk -l
```

Formattare la partizione dedicata ai volumi (sda3 nel caso):

```
$ pvcreate /dev/sda3
```

Creare il gruppo di volumi “nova-volumes” :

```
$ vgcreate nova-volumes /dev/sda3
```

- **Avviare i servizi:**

```
$ sudo service openstack-nova-compute
$ sudo service openstack-nova-volume
$ sudo chkconfig nova-compute on
$ sudo chkconfig nova-volume on
```

- **Abilitare l’accesso via ssh e il comando ping verso le macchine virtuali:**

```
$ nova secgroup-add-rule default icmp -1 -1
0.0.0.0/0
$ nova secgroup-add-rule default tcp 22 22
0.0.0.0/0
```

- **Problemi**

In caso di problemi aggiungere in /etc/nova/nova.conf
`dhcpbridge=/usr/bin/nova-dhcpbridge`

Potrebbe essere necessario creare le cartelle specificate in nova.conf

es `$ mkdir /var/lock/nova`

ed installare:

```
$ yum install lvm2
$ yum install qemu-kvm qemu-img
```

```
$ yum install virt-manager libvirt libvirt-python
python-virtinst libvirt-client
```

3.5 – UTILIZZO DEL SISTEMA

Una volta completata l'installazione del cloud controller e dei sei host si può procedere con il caricamento delle immagini di dischi virtuali, il lancio delle istanze delle macchine virtuali e la creazione dei volumi logici di memoria.

3.5.1 – AVVIO DELLE ISTANZE DELLE MACCHINE VIRTUALI

Viene illustrato un esempio ed in seguito il test effetutato.

Scaricare sul controller, in una cartella temporanea l'immagine desiderata:

```
wget https://cloud-
images.ubuntu.com/precise/current/precise-server-
cloudimg-amd64-disk1.img
```

Caricarla su glance:

```
glance add name="Ubuntu 12.04" is_public=true
container_format=ovf disk_format=qcow2 < precise-
server-cloudimg-amd64-disk1.img
```

controllarne la presenza:

```
glance-index
```

creare chiavi login:

```
nova keypair-add ssh_key > ssh_key.pem
```

impostare permessi sulle chiavi:

```
chmod 0600 ssh_key.pem
```

lanciare macchina virtuale su host (c-cloud-02):

```
$nova flavor-list
$nova image-list
```

```
$nova boot --image <IMAGE_ID> --flavor <FLAVOR_ID>
--key_name ssh_key <VM_NAME> --hint
force_hosts=<HOST>
```

nel caso:

```
nova boot --image 61c0607b-da42-4e1c-b98d-
14780de250a9 --flavor 2 --key_name ssh_key
Ubuntu_12.04 --hint force_hosts=c-cloud-02
```

controllarne lo stato verificando la dicitura ACTIVE:

```
$ nova list
```

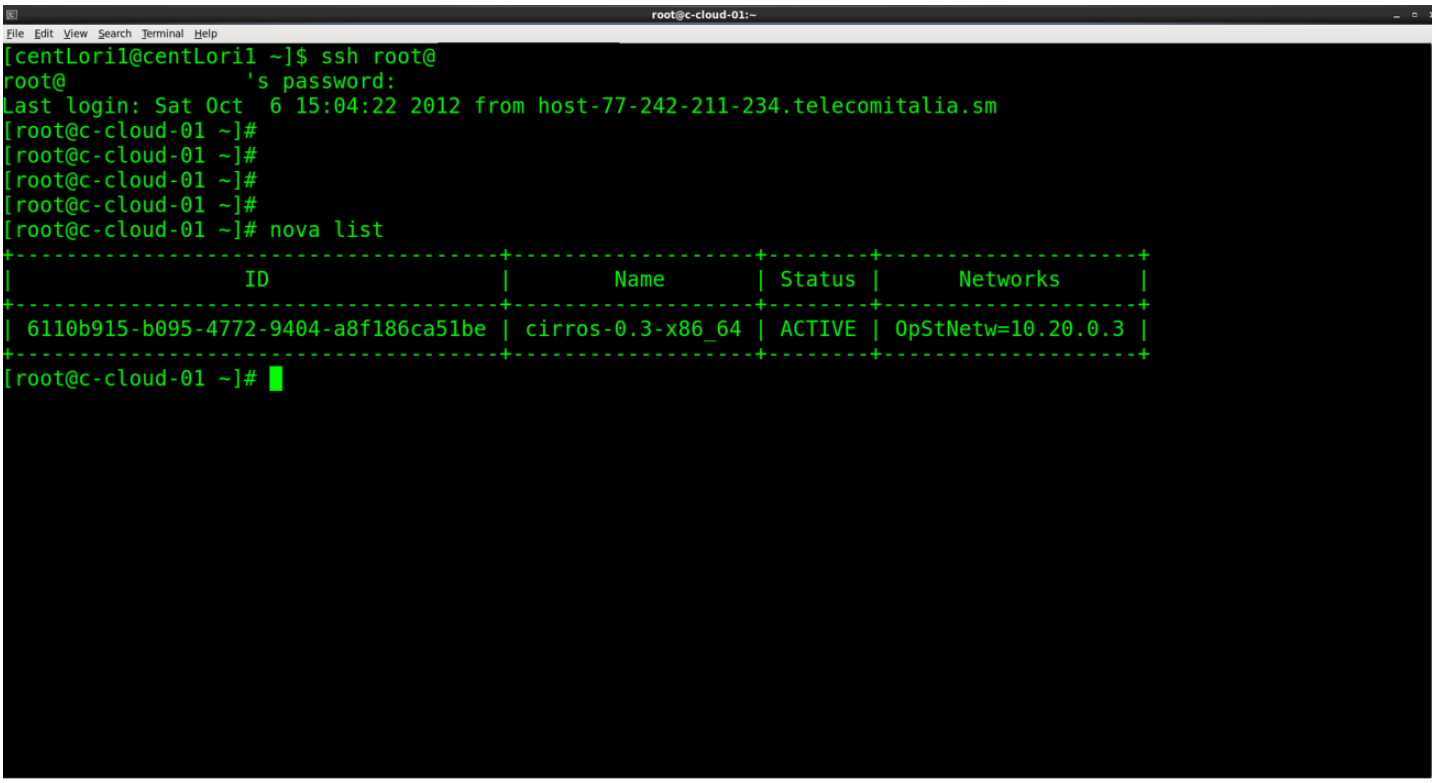

Figura 3.7: Screenshot – Lista di macchine virtuali avviate.Comando nova list

In ogni caso è possibile collegarsi ad un host, dopo aver caricato l'immagine disco, per avviare direttamente la macchina virtuale:

```
$nova boot --image <IMAGE_ID> --flavor <FLAVOR_ID>
--key_name ssh_key <VM_NAME>
```

Una volta avviata ci si connette alla macchina virtuale mediante ssh
Nel caso affrontato il test è stato effettuato su un immagine cirros le cui credenziali erano: user: cirros – password: cubswin:)
Dal controller:

```
$ wget
https://launchpad.net/cirros/trunk/0.3.0/+download
/cirros-0.3.0-x86_64-disk.img
$ name=cirros-0.3-x86_64
$ image=cirros-0.3.0-x86_64-disk.img
$ glance add name=$name is_public=true
container_format=bare disk_format=qcow2 <$image
```

```
$ nova keypair-add ssh_key > ssh_key.pem
$ chmod 0600 ssh_key.pem
```

Dall'host:

```
$ nova flavor-list
$ nova image-list
```

```
$ nova boot --image 71cd982f-533c-489f-bcc6-
7dc2b038251d --flavor 2 --key_name ssh_key cirros-
0.3-x86_64
```

```
$ nova list
```

Login:

```
ssh cirros@<IP>
```

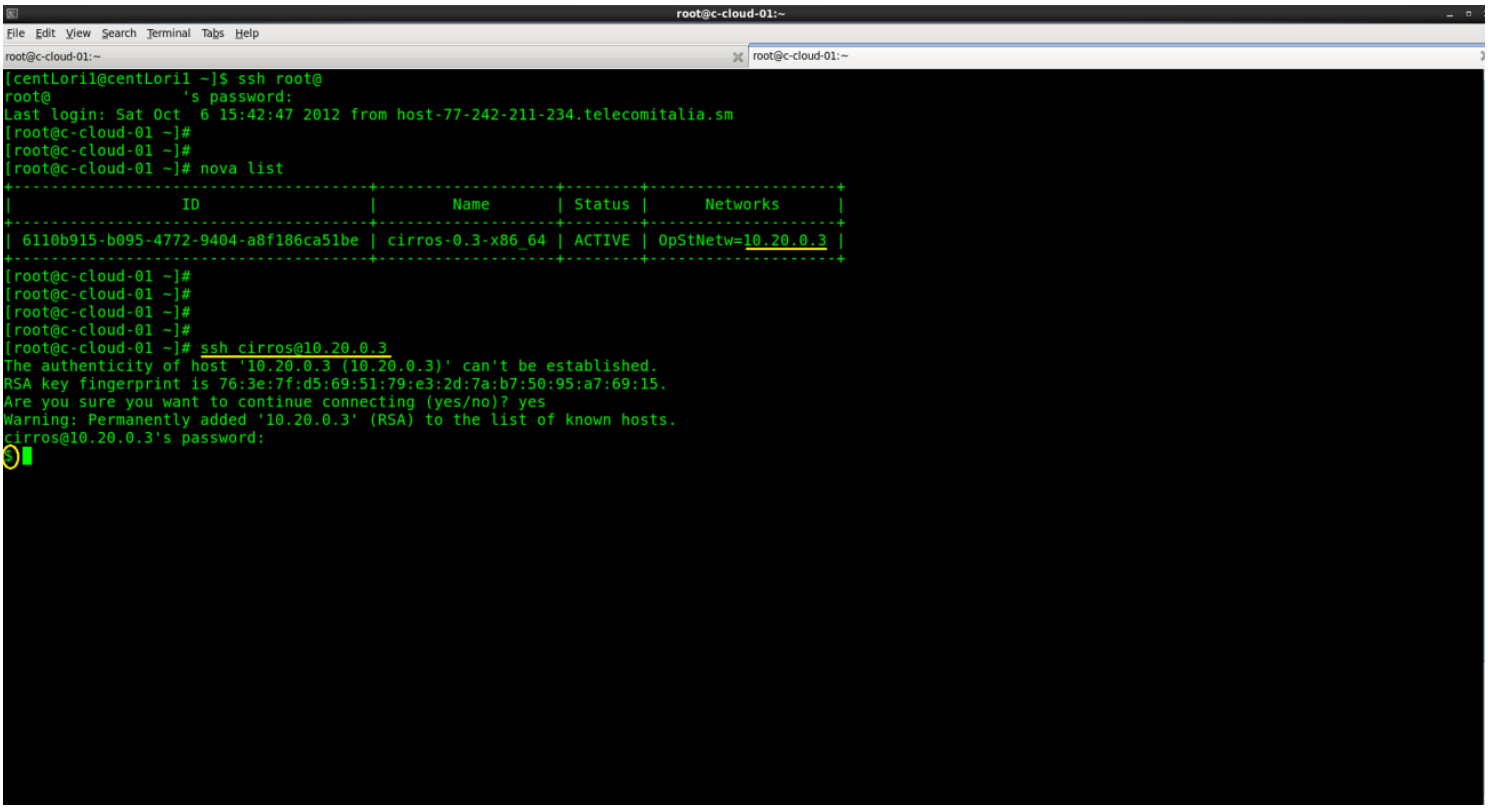

Figura 3.8: Screenshot – Login

3.5.2 - CREAZIONE E COLLEGAMENTO DEI VOLUMI

Dall'host su cui si vuole creare un volume controllare che nova-volume sia attivo:

```
$ nova-manage service list
```

Creare il volume:

```
$ nova volume-create --display_name <VOLUME_NAME>
<SIZE>
TEST: $ nova volume-create --display_name
ccloud02vol1 10
```

Controllare:

```
$ nova volume-list
```

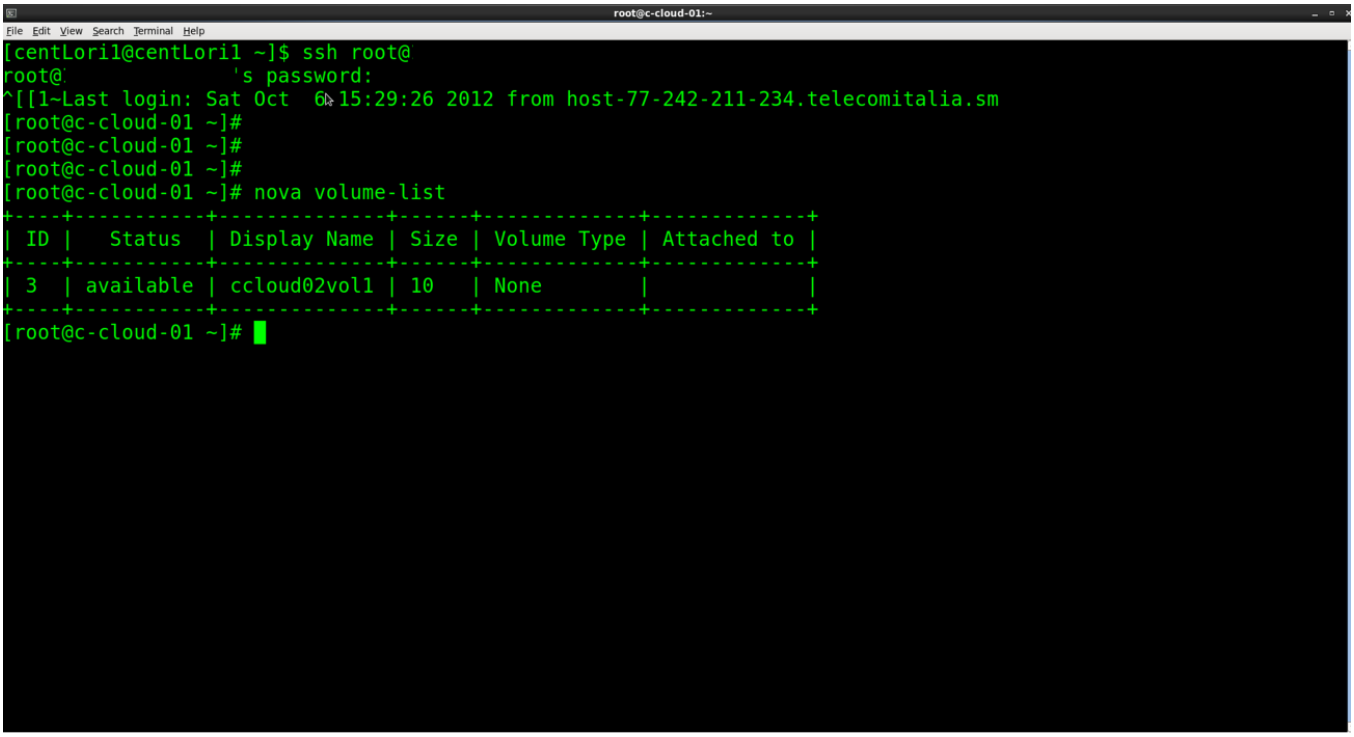

Figura 3.9: Screenshot – Lista dei volumi creati.Comando nova volume-list

Se si desidera eliminarlo

```
$ nova volume-delete <ID>
```

Collegarlo all'istanza:

```
$ nova list
$ nova volume-attach <ID_ISTANZA> <ID_VOLUME>
<DISPOSITIVO>
```

TEST:

```
$nova volume-attach 9d6afa10-cb81-4a98-a454-
7db751343c25 3 /dev/vdc
```

3.5.3 - LIVE MIGRATION

Per poter migrare una macchina virtuale agire come segue su tutti gli host.

Modificare */etc/libvirt/libvirtd.conf*

disabilitando il commento togliendo # e

```
listen_tls = 0
listen_tcp = 1
auth_tcp = "none"
```

riavviare il driver libvirt:

```
$ sudo service libvirtd restart
```

Avviare la migrazione controllando l'ID dell'istanza mediante commando *nova list*:

```
$ sudo nova live-migration --block_migrate <ID_VM>
<NOME_HOST_DESTINAZIONE>
```

Nel caso di studio:

```
$ sudo nova --os_username admin --os_password
admin --os_tenant_name admin --os_auth_url
"http://10.10.0.1:5000/v2.0/" live-migration --
block_migrate 6110b915-b095-4772-9404-a8f186ca51be
c-cloud-03
```

Nota 1:

--block_migrate è un opzione che permette la migrazione senza condividere la directory istances fra i vari host. Nel caso si volesse farne a meno configurare file system nfs come nella guida *[14]*

Nota 2:

Nel caso si presenti un errore di policy, dovuto a permessi non rilasciati, modificare */etc/nova/policy.json* aggiungendo `rule:admin_or_owner` alla stringa *"migrateLive"*

4- GUIDA RAPIDA ALL'AVVIO DEL SISTEMA DEL CASO DI STUDIO

CONTROLLER:

```
$ /etc/init/.d/httpd start
$ /etc/init.d/dhcpd start
$ /etc/init/.d/ntpd start
$ sudo service rabbitmq-server start
$ /etc/init.d/mysqld start
$ /etc/init.d/iscsi start
$ service  netbsd-iscsi start
$ /etc/init.d/tgtd restart
```

```
$ keystone-all &>/dev/null &
$ glance-api &>/dev/null &
$ glance-registry &>/dev/null &
$ nova-api &>/dev/null &
$ nova-network &>/dev/null &
$ nova-scheduler &>/dev/null &
$ nova-compute &>/dev/null &
```

HOST:

```
$ /etc/init/.d/httpd start
$ /etc/init/.d/ntpd start
$ sudo service rabbitmq-server start
$ /etc/init.d/iscsi start
$ service  netbsd-iscsi start
$ /etc/init.d/tgtd restart
```

```
$ nova-compute &>/dev/null &
$ nova-volume &>/dev/null &
```

5- CONCLUSIONI

Giunti al termine di questo lavoro di tesi è opportuno riassumere la situazione attuale concludendo infine con alcune considerazioni. La procedura di installazione eseguita inizialmente, la quale prevedeva il download dei singoli moduli e l'inizializzazione degli stessi tramite python si è rivelata eccessivamente problematica su sistema CentOS. Ogni componente ha necessitato infatti di numerosi interventi al fine di soddisfare la compatibilità con il sistema ospitante e, nonostante l'ausilio della community di Openstack, il completamento delle operazioni ha richiesto tantissimo tempo. La stessa dashboard una volta avviata ha presentato problemi di template rendering che dovevano essere corretti attraverso piccole modifiche nei sorgenti. La situazione si è completamente capovolta una volta adottato il secondo approccio, ovvero mediante repository EPEL. La documentazione si è dimostrata corrispondente al caso reale e l'unica difficoltà si è presentata nel settaggio dei file di configurazione e nella gestione della rete virtuale. Il problema di conflittualità fra indirizzi fisici e logici è stato risolto mediante la creazione di un alias sull'interfaccia principale del cloud controller.

La situazione attuale presenta tutti i servizi attivi su controller e computer host, il test di avvio delle macchine virtuali è andato a buon fine, la dashboard con la quale gestire il sistema funziona correttamente.

Il prossimo passo consisterà nella migrazione di istanze di macchine virtuali fra due host della rete locale per poi tentare una migrazione fra il sistema installato a Cesena e quello risiedente a Bologna. Sono già in atto test di live-migration da perfezionare.

Concludendo è doveroso sottolineare le ottime potenzialità di Openstack, in particolare dopo aver citato i punti di forza di un sistema cloud, anche se la

discreta complessità del sistema soprattutto in fase di configurazione lascia qualche perplessità sulla distribuzione in tempi brevi. D'altro canto, la community e le società alla base del progetto sembrano operare celermente nel rilascio di patch e aggiornamenti offrendo un supporto in via di miglioramento aumentando così anche le possibilità, se mai ce ne fosse bisogno, di riuscita degli esperimenti previsti nel progetto universitario del CIRI ICT.

Le potenzialità concretizzabili da questi ultimi, riassumibili nell'ottimizzazione degli strumenti alla base dei sistemi sopra descritti quali ad esempio la rete, rischiano di avere un impatto molto importante data la rilevanza che questa tecnologia ricoprirà nei prossimi anni. Senza ombra di dubbio si può indicare il "cloud-computing" come uno degli scenari protagonisti dell'ambito ICT. Ad un livello molto più concreto infatti esso influenzerà sia l'ambito professionale che privato entrando a far parte, come solo le maggiori innovazioni tecnologiche hanno saputo fare, degli strumenti usati quotidianamente da chiunque.

Bibliografia e sitografia:

[1] *The NIST definition of Cloud Computing* - http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] *Il cloud computing spiegato da Oracle* - http://cristiancudizio.wordpress.com/2010/03/24/il-cloud-computing-spiegato-da-oracle/

[3] *Raccomandazioni e proposte sull'utilizzo del cloud computing nella pubblica amministrazione* - http://www.digitpa.gov.it/notizie/uso-del-cloud-computing-nella-pa

[4] *C.N Hofer- G. Karagiannis* - Computing services: Taxonomy and comparison

[5] *Cos'è il cloud computing: dati tra le nuvole* - http://www.comefarea.it/internet/cloud/cloudcomputing/

[6] *Virtualizzazione server* - http://www.starsystem.biz/virtualizzazione-server/

[7] *Cloud computing: la sicurezza è ancora un problema* - http://www.consulenti-ict.it/Area-Tecnica/Virtualizzazione/cloud-computing-la-sicurezza-e-ancora-un-problema.html

[8] *Il cloud computing di IBM per il tennis al Roland-Garros* - http://www.cloudeconomy.it/il-cloud-computing-di-ibm-per-il-tennis-al-roland-garros

[9] *Openstack, il software open che rivoluziona il cloud computing* - http://daily.wired.it/news/internet/2012/04/06/openstack-open-source-cloud-36241.html

[10] *Openstack: storia e architettura del progetto cloud open source* - http://www.cloudtalk.it/openstack-storia-e-architettura-del-progetto-cloud-open-source/4

[11] *Descrizione centOS* - http://it.wikipedia.org/wiki/CentOS

[12] *Esempio creazione endpoint* - http://docs.openstack.org/essex/openstack-compute/starter/content/Creating_Endpoints-d1e469.html

[13] *CIRI ICT:* http://www.ciri-ict.unibo.it

[14] *Configurazione file system nfs:* http://docs.openstack.org/trunk/openstack-compute/admin/content/configuring-live-migrations.html

Lista immagini:

RINGRAZIAMENTI

Ora che il lavoro è terminato, augurandomi che possa essere in qualche modo utile a coloro i quali si avvicinano al mondo del cloud computing ed in particolare ad Openstack, desidero che rimangano impressi alcuni ringraziamenti. Il primo pensiero non può che andare alla mia famiglia, in primo luogo ai miei genitori, Claudio e Francesca, che mi hanno dato si la possibilità economica ma soprattutto il sostegno affettivo a me necessario per arrivare in fondo al percorso universitario, culminato con questa tesi, credendo in me quando io stesso ne dubitavo. Dire che quello che sono è merito loro non è certo esagerato. Non posso che continuare con Alessandra e Carlotta, le mie sorelle così come i miei nonni e mia zia Paola che non hanno mai perso occasione di dimostrare il loro interesse e la loro vicinanza. Un pensiero è naturale verso Luca, Zebe e Bene gli amici con i quali condividerò per sempre i primi passi nel mondo universitario iniziato a Bologna poi proseguito con una persona fantastica, Ezio, a Cesena per finire poi con tutti gli amici che mi hanno spinto e sostenuto anche nella redazione di questa tesi.
Un sentito ringraziamento va al mio relatore, Walter Cerroni, per la disponibilità e cortesia dimostratami nelle fasi più critiche del lavoro svolto e per la pazienza nel replicare a tutte le mie domande. E' stato un piacere lavorare insieme.
Infine un buon pensiero anche per chi in origine non credeva in me, per aver in qualche modo contribuito ad insegnarmi a guardare più in alto delle mie potenzialità portandomi ad affrontare difficoltà che oggi riconosco come parte della mia crescita.
Chiudo con un incoraggiamento agli universitari che leggeranno queste righe consigliandovi di valutare relativamente i vostri limiti considerando piuttosto che la volontà di superarli è già il primo passo tangibile per riuscirci. Questo, unito alle persone sopra citate, è ciò che mi ha permesso di trovare una parte della mia strada, della mia dimensione arrivando a redigere una tesi di Ingegneria Informatica e potendo guardare il futuro con entusiasmo, conscio finalmente delle mie qualità. Grazie

APPENDICE A

/etc/keystone/keystone.conf

```
[DEFAULT]
bind_host = 0.0.0.0
public_port = 5000
admin_port = 35357
admin_token = <admin_token>
compute_port = 8774
verbose = True
debug = True
#log_config = ./etc/logging.conf.sample

# ================= Syslog Options ============================
# Send logs to syslog (/dev/log) instead of to file specified
# by `log-file`
use_syslog = False
log_file = /var/log/keystone/keystone.log

# Facility to use. If unset defaults to LOG_USER.
# syslog_log_facility = LOG_LOCAL0

[sql]
connection =
mysql://keystoneAdmin:<keystone_password>@10.10.0.1/keystone
idle_timeout = 200

[ldap]
#url = ldap://localhost
#tree_dn = dc=example,dc=com
#user_tree_dn = ou=Users,dc=example,dc=com
#role_tree_dn = ou=Roles,dc=example,dc=com
#tenant_tree_dn = ou=Groups,dc=example,dc=com
#user = dc=Manager,dc=example,dc=com
#password = freeipa4all
#suffix = cn=example,cn=com

[identity]
driver = keystone.identity.backends.sql.Identity

[catalog]
driver = keystone.catalog.backends.sql.Catalog
template_file = /etc/keystone/default_catalog.templates

[token]
driver = keystone.token.backends.sql.Token

# Amount of time a token should remain valid (in seconds)
expiration = 86400
```

```
[policy]
driver = keystone.policy.backends.rules.Policy

[ec2]
driver = keystone.contrib.ec2.backends.sql.Ec2

[filter:debug]
paste.filter_factory = keystone.common.wsgi:Debug.factory

[filter:token_auth]
paste.filter_factory =
keystone.middleware:TokenAuthMiddleware.factory

[filter:admin_token_auth]
paste.filter_factory =
keystone.middleware:AdminTokenAuthMiddleware.factory

[filter:xml_body]
paste.filter_factory =
keystone.middleware:XmlBodyMiddleware.factory

[filter:json_body]
paste.filter_factory =
keystone.middleware:JsonBodyMiddleware.factory

[filter:crud_extension]
paste.filter_factory =
keystone.contrib.admin_crud:CrudExtension.factory

[filter:ec2_extension]
paste.filter_factory = keystone.contrib.ec2:Ec2Extension.factory

[app:public_service]
paste.app_factory = keystone.service:public_app_factory

[app:admin_service]
paste.app_factory = keystone.service:admin_app_factory

[filter:debug]
paste.filter_factory = keystone.common.wsgi:Debug.factory

[filter:token_auth]
paste.filter_factory =
keystone.middleware:TokenAuthMiddleware.factory

[filter:admin_token_auth]
paste.filter_factory =
keystone.middleware:AdminTokenAuthMiddleware.factory

[filter:xml_body]
paste.filter_factory =
```

```
keystone.middleware:XmlBodyMiddleware.factory

[filter:json_body]
paste.filter_factory =
keystone.middleware:JsonBodyMiddleware.factory

[filter:crud_extension]
paste.filter_factory =
keystone.contrib.admin_crud:CrudExtension.factory

[filter:ec2_extension]
paste.filter_factory = keystone.contrib.ec2:Ec2Extension.factory

[app:public_service]
paste.app_factory = keystone.service:public_app_factory

[app:admin_service]
paste.app_factory = keystone.service:admin_app_factory

[pipeline:public_api]
pipeline = token_auth admin_token_auth xml_body json_body debug
ec2_extension public_service

[pipeline:admin_api]
pipeline = token_auth admin_token_auth xml_body json_body debug
ec2_extension s3_extension crud_extension admin_service

[app:public_version_service]
paste.app_factory = keystone.service:public_version_app_factory

[app:admin_version_service]
paste.app_factory = keystone.service:admin_version_app_factory

[pipeline:public_version_api]
pipeline = xml_body public_version_service

[pipeline:admin_version_api]
pipeline = xml_body admin_version_service

[composite:main]
use = egg:Paste#urlmap
/v2.0 = public_api
/ = public_version_api

[composite:admin]
use = egg:Paste#urlmap

/v2.0 = admin_api
/ = admin_version_api

[filter:s3_extension]
paste.filter_factory = keystone.contrib.s3:S3Extension.factory
```

/etc/nova/nova.conf

```
[DEFAULT]

# LOGS/STATE
verbose=True
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova

# AUTHENTICATION
auth_strategy=keystone

# SCHEDULER
compute_scheduler_driver=nova.scheduler.filter_scheduler.FilterS
cheduler

# VOLUMES
volume_group=nova-volumes
volume_name_template=volume-%08x
iscsi_helper=tgtadm

# DATABASE
sql_connection=mysql://novaAdmin:<nova_password>@10.10.0.1/nova

# COMPUTE
libvirt_type=kvm
connection_type=libvirt
instance_name_template=instance-%08x
api_paste_config=/etc/nova/api-paste.ini
allow_resize_to_same_host=True

# APIS
osapi_compute_extension=nova.api.openstack.compute.contrib.stand
ard_extensions
ec2_dmz_host=10.10.0.1
s3_host=10.10.0.1

# RABBITMQ
rabbit_host=10.10.0.1
rabbit_port = 5673

# GLANCE
image_service=nova.image.glance.GlanceImageService
glance_api_servers=10.10.0.1:9292

# NETWORK
network_manager=nova.network.manager.FlatDHCPManager
force_dhcp_release=True
dhcpbridge_flagfile=/etc/nova/nova.conf
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDrive
r
```

```
# Change my_ip to match each host
my_ip=10.10.0.1
#public_interface=virbr0
#vlan_interface=eth0
flat_network_bridge=br100
flat_interface=eth0
fixed_range=10.10.0.0/24
dhcpbridge=/usr/bin/nova-dhcpbridge

# NOVNC CONSOLE
novncproxy_base_url=http://10.10.0.1:6080/vnc_auto.html
# Change vncserver_proxyclient_address and vncserver_listen to
match each compute host
vncserver_proxyclient_address=10.10.0.1
vncserver_listen=10.10.0.1
```

File kickstart (solo parte significativa):

```
# Kickstart file automatically generated by anaconda.

#version=DEVEL
install
url --url=http://mi.mirror.garr.it/mirrors/CentOS/6/os/x86_64
lang en_US.UTF-8
keyboard it
network --onboot yes --device eth0 --bootproto static --ip
10.10.0.2 --netmask 255.255.255.0 --gateway 10.10.0.1 --ipv6
auto --nameserver <name_server> --hostname c-cloud-02
rootpw  --iscrypted
$6$1kQ2gSwweg$y6JCu2f.gaUVcXjV1jwMyA/YAxMsE6XwSrU7W4CZzvIzx.njCa
HruUweZSoTSiNoXAZeeeddMYDE4amwzoHbPYe.7k.

firewall --disabled
authconfig --enableshadow --passalgo=sha512
selinux --disabled
timezone --utc Europe/Rome
bootloader --location=mbr --driveorder=sda --append=" rhgb
crashkernel=auto   rhgb crashkernel=auto quiet crashkernel=auto
quiet rhgb quiet"
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
#clearpart --all --drives=sda

#part /boot --fstype=ext4 --size=500
#part pv.008002 --grow --size=1

#volgroup vg_ccloud01 --pesize=4096 pv.008002
#logvol / --fstype=ext4 --name=lv_root --vgname=vg_ccloud01 --
size=466388
```

```
#logvol swap --name=lv_swap --vgname=vg_ccloud01 --size=10048

repo --name="CentOS"  --
baseurl=http://mi.mirror.garr.it/mirrors/CentOS/6/os/x86_64 --
cost=100
repo --name="CentOS"  --
baseurl=http://mi.mirror.garr.it/mirrors/CentOS/6/os/x86_64 --
cost=100
```

Installazione - phpmyadmin:

```
$ yum install php
$ mkdir phpmyadmin
$ cd phpmyadmin
$ wget http://dag.wieers.com/rpm/packages/phpmyadmin/phpmyadmin-
2.11.1.2-1.el5.rf.noarch.rpm
$ rpm -ivh phpmyadmin-2.11.1.2-1.el5.rf.noarch.rpm
```

Aprire*: /usr/share/phpmyadmin/config.inc.php*

modificare:

```
cfg['Servers'][$i]['auth_type'] = 'cookies';
```

con:

```
cfg['Servers'][$i]['auth_type'] = 'http';
```

Se si necessita di accedere ai database da remoto modificare:

```
nano /etc/httpd/conf.d/phpmyadmin.conf
```

mettendo vicino ad "allow" la lista degli IP consentiti

Order Deny,Allow
Deny from all
Allow from 127.0.0.1 <IP1> <IP2>

Cancellare anche gli "any" altrimenti non funziona localhost

Comando per copiare un file da remoto:

```
scp <user>@<IP>:<path_remoto file> <path_locale file>
```

Comando per copiare una directory da remoto:

```
scp -r <user>@<IP>:<path_remoto dir> <path_locale dir>
```

APPENDICE B

L'installazione manuale è stata eseguita attraverso le guide ufficiali disponibili ma soprattutto grazie all'ausilio fornito dalla guida dell'Ing. Massimiliano Mattetti. Il lavoro svolto ricalca quelle linee guida con l'aggiunta dei principali problemi rilevati.

CONTROLLER - Installazione di openstack manuale mediante python:

La parte di configurazione iniziale rimane uguale alla guida precedente. Viene creata una cartella la quale conterrà tutti i sorgenti scaricati.

```
$ mkdir /root/Openstack
```

- Installazione python e pip:

```
$ yum install python
$ yum install gcc python-devel
```

in questo caso python era gia installato. Questo però non comprende il comando pip. Si procede controllando da "http://pypi.python.org/packages/source/p/pip/" l'utlima versione disponibile la quale viene scaricata in una directory temporanea:

```
$ cd Openstack
$ mkdir pipTemp
$ cd pipTemp
$ wget http://pypi.python.org/packages/source/p/pip/pip-1.1.tar.gz
```

scompattata:

```
$ tar -xzf pip-1.1.tar.gz
```

ed installata

```
$ cd pip-1.1
$ python setup.py install
```

- Installazione Django > 1.4:

```
$ yum install subversion
$ cd /var/www
$ mkdir django-src
$ cd django-src
$ svn co http://code.djangoproject.com/svn/django/trunk/
```

```
$ cd trunk
$ python setup.py install
```

***** Scaricare con git tutti i moduli subito. Farlo in tempi diversi potrebbe significare incompatibilità di versione *****

- CREAZIONE DEI DATABASE

Come guida precedente

- INSTALLAZIONE E CONFIGURAZIONE KEYSTONE

All'interno della cartella "Openstack":

```
$ git clone http://github.com/openstack/keystone.git
$ cd keystone
$ sudo pip install -r tools/pip-requires
$ python setup.py build
$ python setup.py install
```

E' possibile che a questo punto sorgano errori dovuti alla mancanza/incompatibilità di librerie. Aggiornare il sistema in base all'output mostrato. Nel caso il sistema potrebbe richiedere l'aggiornamento di SQLAlchemy >0.6.

```
$ yum install python-sqlalchemy
```

Installare SQLAlchemy 0.7.

```
$ wget
http://sourceforge.net/projects/sqlalchemy/files/sqlalchemy/0.7.
8/SQLAlchemy-0.7.8.tar.gz/download
$ tar -xzf SQLAlchemy-0.7.8.tar.gz
$ cd SQLAlchemy-0.7.8
$ python setup.py install
```

Creare la directory /etc/keystone

```
$ mkdir /etc/keystone
$ cp etc/* /etc/keystone
```

Configurare keystone.conf come nella guida principale
Eseguire il servizio

```
$ keystone-all [&>/dev/null &]
```

Sincronizzare il database

```
$ keystone-manage db_sync
```

Il comando potrebbe ritornare un errore “filedecoder”.

RISOLTO: easy_install anyjson==0.2.4

**Con la versione 0.2.4 si avvia keystone. Con la 0.3.3 si sincronizza il database

```
$ export SERVICE_ENDPOINT="http://localhost:35357/v2.0"
$ export SERVICE_TOKEN=012345SECRET99TOKEN012345
```

Aggiungere le stesse variabili nel file ~/.bashrc

- PYTHON-KEYSTONECLIENT

```
$ git clone https://github.com/openstack/python-keystoneclient
$ cd python-keystoneclient
$ pip install -r tools/pip-requires
$ python setup.py build
$ python setup.py install
```

e configurare tenant, user, role come nella precedente guida.

- INSTALLAZIONE GLANCE

```
$ git clone git://github.com/openstack/glance
$ cd glance
$ pip install -r tools/pip-requires
$ python setup.py build
$ python setup.py install

$ mkdir /etc/glance
$ cp etc/* /etc/glance
```

Configurare i file di glance come nella guida precedente

```
$ glance-manage version control 0
```

(Potrebbe essere necessario aggiornare python crypto alla versione 2.6 e commentare il log file dentro glance-registry.conf e glance-api.conf)

```
$glance-manage db_sync
```

avviare i servizi

```
$ glance-registry &>/dev/null &
$ glance-api &>/dev/null &
```

esportare le seguenti variabili

```
$ export OS_TENANT_NAME=admin
$ export OS_USERNAME=admin
$ export OS_PASSWORD=admin
$ export OS_AUTH_URL=http://localhost:5000/v2.0/
```

anche in questo caso memorizzarle all'interno del file ~/.bashrc

Caricare un'immagine di prova

```
$ wget
https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-
0.3.0-x86_64-disk.img
$ name=cirros-0.3-x86_64
$ image=cirros-0.3.0-x86_64-disk.img
$ glance add name=$name is_public=true
container_format=bare disk_format=qcow2 < $image
```

e verificarla

```
$ glance index
```

PYTHON GLANCE-CLIENT

```
$ git clone https://github.com/openstack/python-
glanceclient
$ cd python-glanceclient
$ pip install -r tools/pip-requires
$ python setup.py build
$ python setup.py install
```

INSTALLAZIONE NOVA

Preconfigurazione:

```
$ yum install bridge-utils
$ yum install dnsmasq-utils
```

Impostare auth=no in /etc/qpidd.conf.
sudo setenforce permissive

Installazione:

```
$ git clone https://github.com/openstack/nova.git
$ cd nova
$ pip install -r tools/pip-requires
$ python setup.py build
$ python setup.py install

$ mkdir /etc/nova
$ cp etc/nova/* /etc/nova
```

Configurare nova come nella guida precedente

```
$ sudo yum install openstack-utils memcached qpid-cpp-
server
$ yum install qemu-kvm qemu-img
$ yum install virt-manager libvirt libvirt-python python-
virtinst libvirt-client
```

Creare il gruppo nova

```
$ groupadd nova
$ /etc/nova/nova.conf
$ chown -R root:nova /etc/nova
$ chmod 644 /etc/nova/nova.conf
```

Sincronizzare il db

```
$ nova-manage db sync
```

A questo punto si è manifestato un errore grave, non funzionava nemmeno keystone. Reinstallando glance è andato tutto a posto

```
nova-manage network create openstackVNet --
fixed_range_v4=10.0.0.0/24 --num_networks=1  --
bridge_interface=br100 --network_size=256

$ sudo yum install kvm libvirt
```

Eseguire i servizi:

```
$ nova-api &>/dev/null &
$ nova-network &>/dev/null &
$ nova-scheduler &>/dev/null &
```

Il commando seguente permette di visualizzare lo stato dei servizi

```
$ nova-manage service list
```

PYTHON-NOVACLIENT

```
$ git clone https://github.com/openstack/python-novaclient
$ cd python-novaclient
$ pip install -r tools/pip-requires
$ python setup.py build
```

```
$ python setup.py install
```

INSTALLAZIONE HORIZON

```
$ git clone git clone
https://github.com/openstack/horizon.git
$ cd horizon
$ pip install -r tools/pip-requires
```

Configurare horizon come nella guida precedente

```
$ ~/Openstack/horizon/manage.py syncdb
```

Installare nodejs

```
$ wget http://nodejs.tchol.org/repocfg/el/nodejs-stable-
release.noarch.rpm
$ yum localinstall --nogpgcheck nodejs-stable-
release.noarch.rpm
$ yum install nodejs-compat-symlinks npm
$ rm nodejs-stable-release.noarch.rpm
```

Lanciare la dashboard

```
$ ~/Openstack/horizon/manage.py runserver 0.0.0.0:8000
```

Accedendo alla dashboard si sono verificati numerosi problem di rendering. La soluzione risiedeva nel'aggiungere " nei file segnalati.

ESEMPIO:
{% trans "Settings" %}
si risolve con
{% trans "Settings" %}

mentre l'errore riguardante "refresh..index" viene risolto cancellando quella parte