Cloud computing:
i sistemi Cloud e l'architettura OpenStack

Luca Foschini e
Alessandro Pernafini

DISI – Università di Bologna

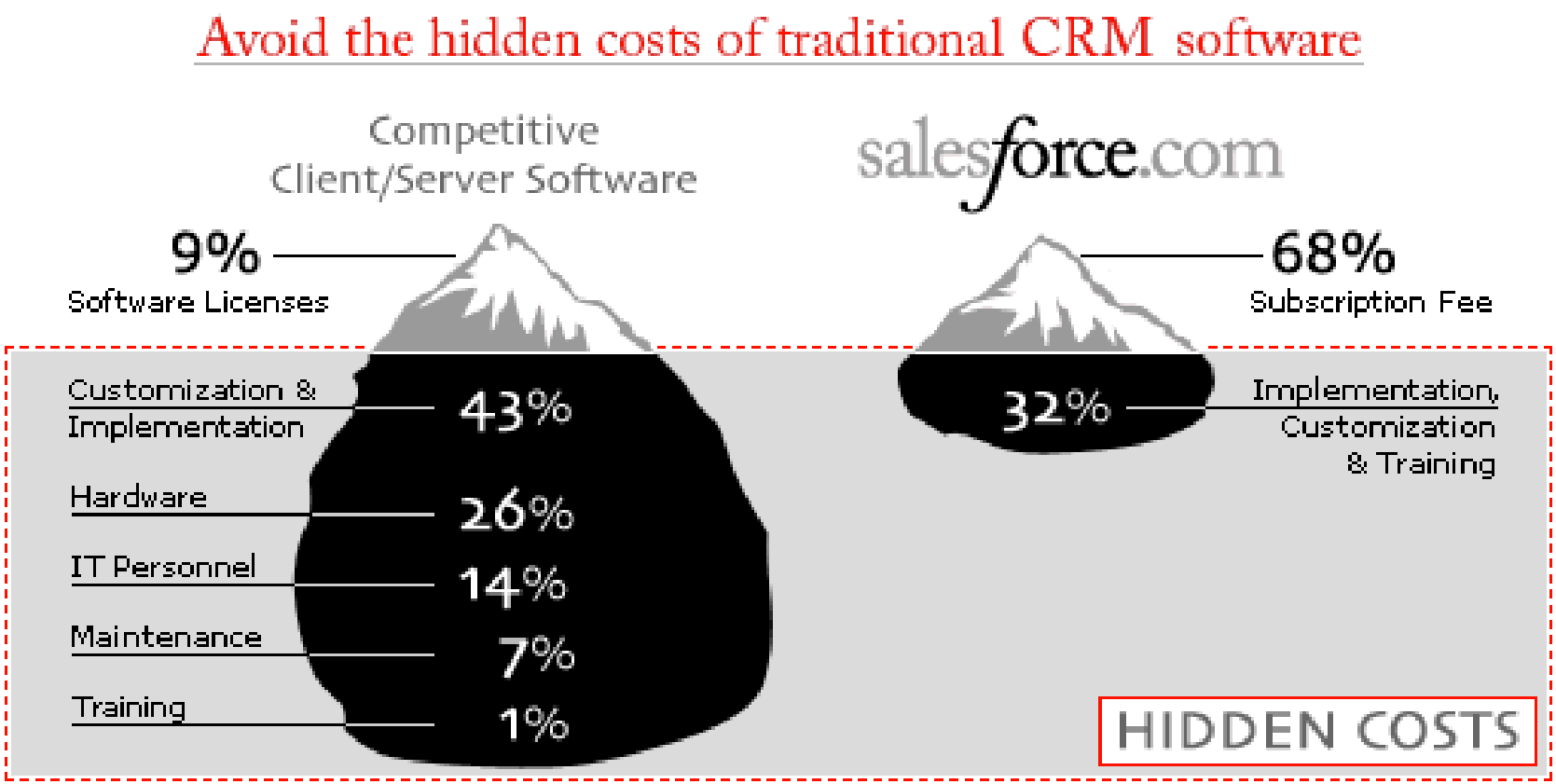
luca.foschini@unibo.it
alessandro.pernafini@studio.unibo.it

Cos'è il Cloud computing?

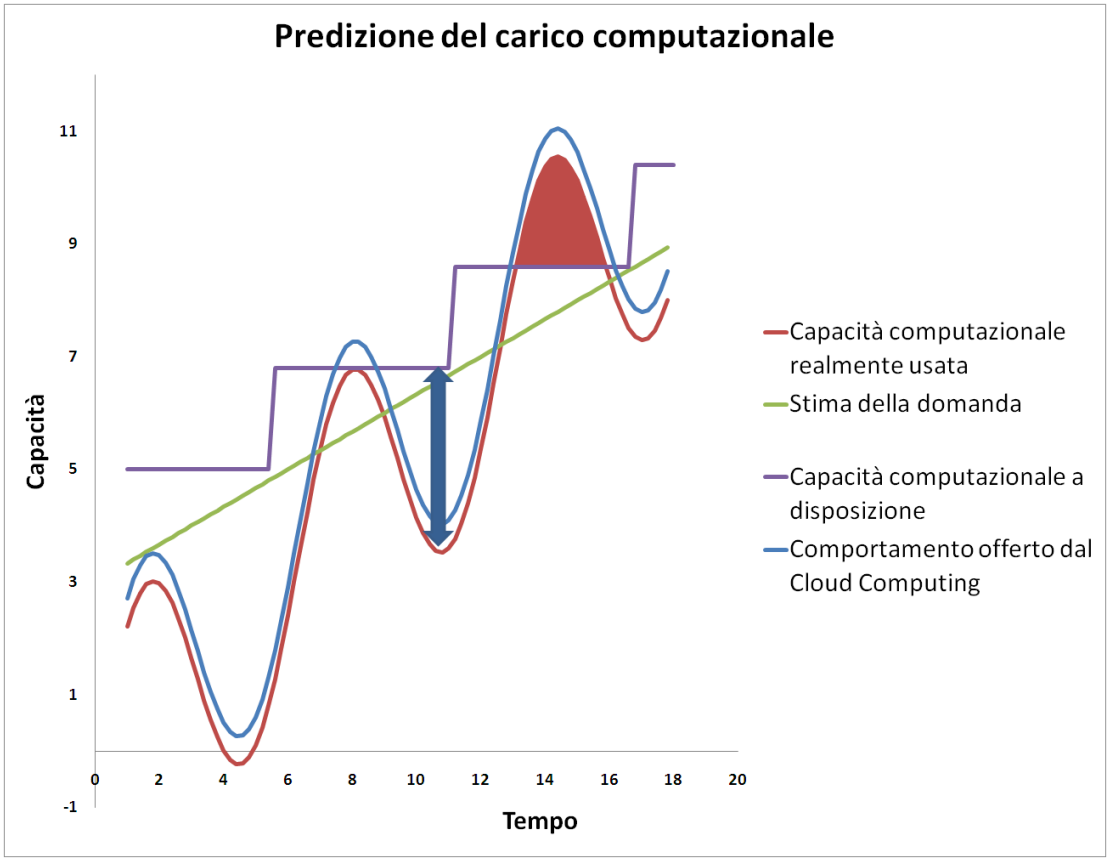
"The architecture and terminology of Cloud computing is as clearly and precisely ***defined as, well, a cloud****."*

Fonte:
www.openCloudmanifesto.org

Motivazioni: i costi dell'IT

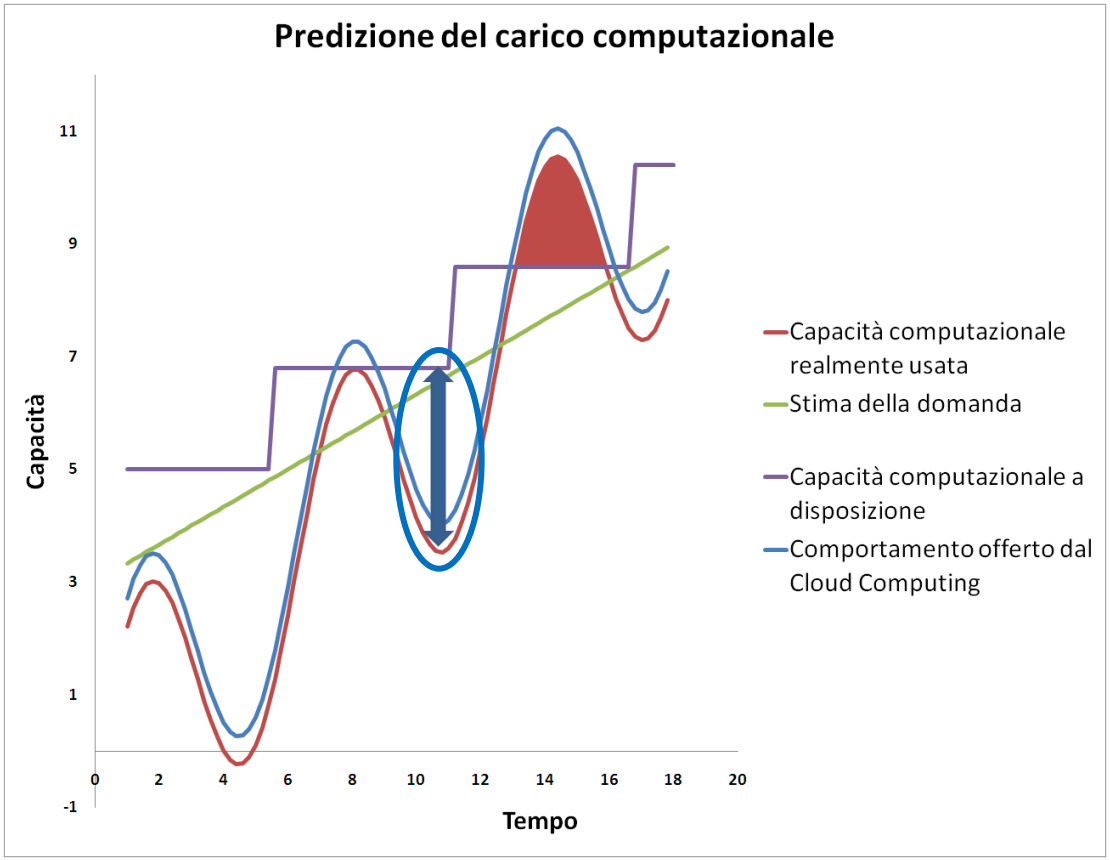


Motivazioni: gestione flessibile e in crescita dell'infrastruttura IT



L'interesse principale di un'azienda relativamente all'Information Technology è legato alla capacità di **fornire un servizio in maniera continuata, adattandosi rapidamente a cambiamenti** delle necessità economiche

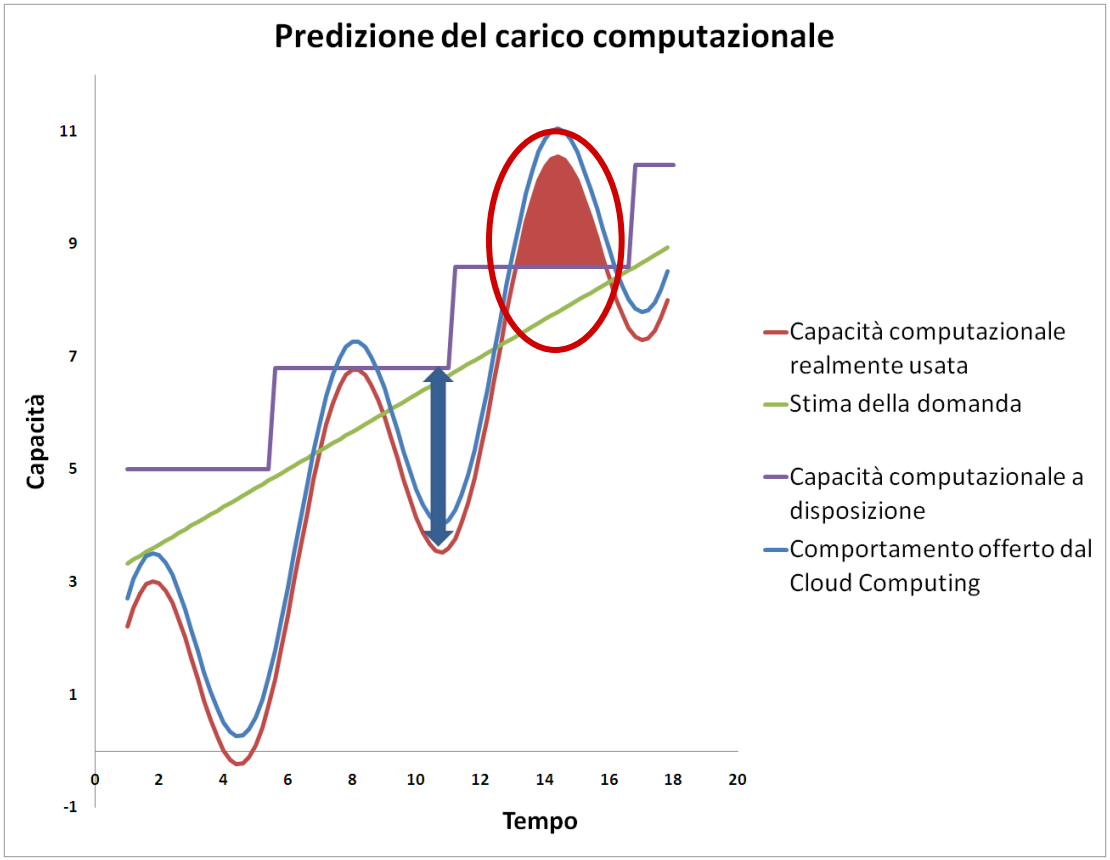
Motivazioni: gestione flessibile e in crescita dell'infrastruttura IT



L'area tra la linea rossa e la viola evidenzia la porzione di infrastruttura informatica che l'azienda ha pagato per avere ma che all'atto pratico non utilizza

Spreco di denaro, tempo e investimenti

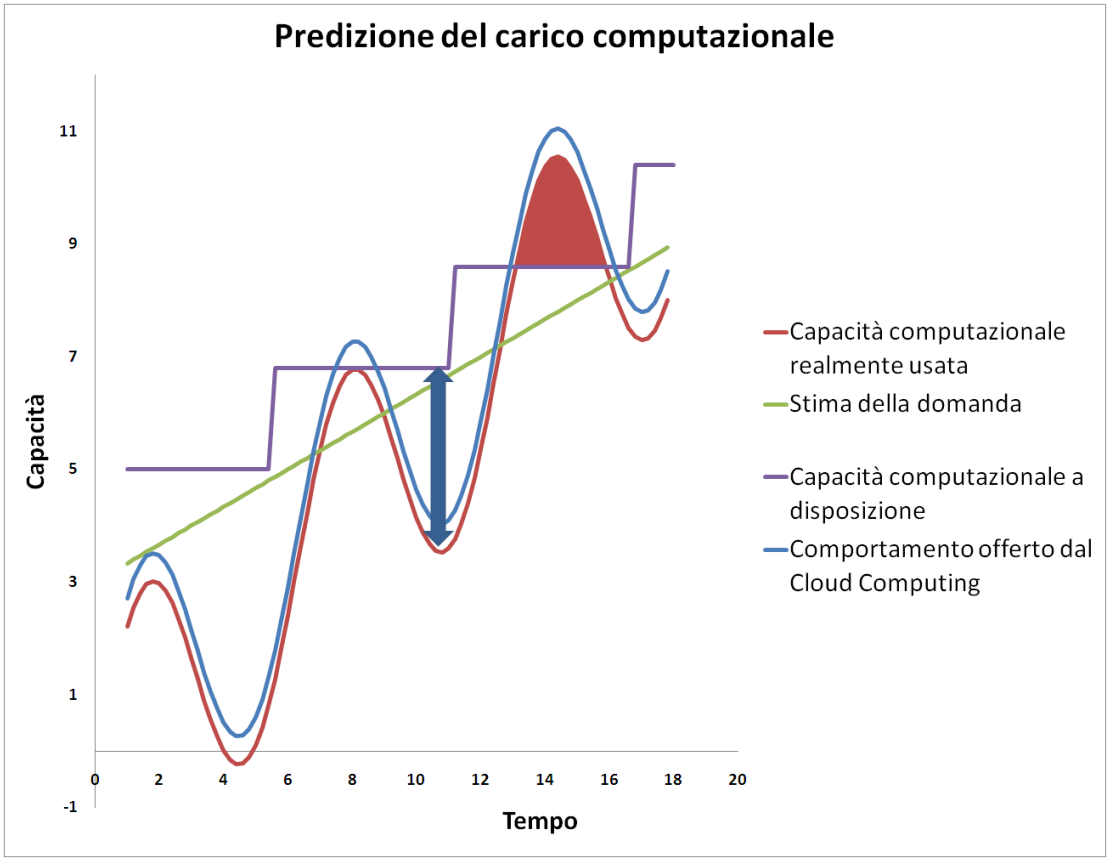
Motivazioni: gestione flessibile e in crescita dell'infrastruttura IT



Se al contrario l'impresa ha un successo inaspettato ed un conseguente aumento di clienti, la capacità computazionale non è in grado di scalare

Rallentamenti, blocchi nell'esecuzione e perdita di opportunità di guadagno

Motivazioni: gestione flessibile e in crescita dell'infrastruttura IT

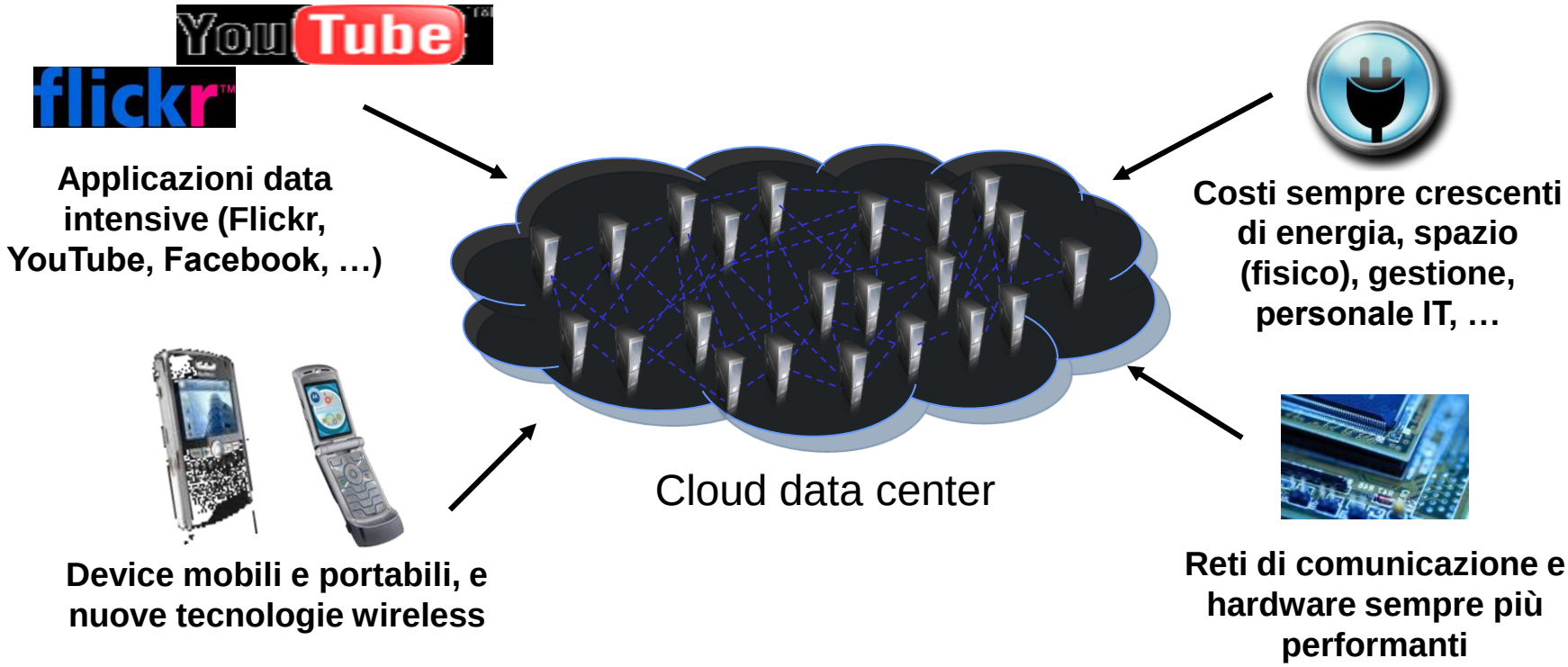


I problemi di previsione potrebbero essere risolti se fosse possibile avere un'infrastruttura informatica altamente flessibile, in grado di seguire la variazione di necessità di calcolo di giorno in giorno o di ora in ora (**elastica** ☺)

Cloud computing: problem space

"It starts with the premise that the **data services and architecture should be on servers**. We call it **Cloud computing** – **they should be in a 'Cloud' somewhere**. And that if you have the right kind of browser or the right kind of access, it doesn't matter whether **you have a PC or a Mac or a mobile phone or a BlackBerry or what have you – or new devices still to be developed – you can get access to the Cloud**…"

– Dr. Eric Schmidt, Google CEO, August 2006



Cloud computing: ingredienti di base

- Modello di costo IT on demand
- In un contesto affidabile (reliable)
- Pool di risorse di computing virtualizzate
- Provisioning rapido e su richiesta
- Sistemi su architetture scalabili

Cloud keywords

on demand, reliability, virtualizzazione, provisioning, scalabilità

Cloud computing: alcune prime definizioni

Il Cloud è in grado di effettuare **provisioning di risorse IT 'as a service'**

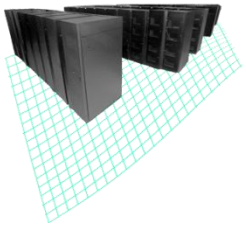
Il Cloud è un **servizio IT** con le seguenti proprietà:

- una **interfaccia utente** che rende l'infrastruttura sottostante il servizio trasparente per l'utente
- ridotti **costi incrementali di gestione** quando si aggiungono risorse IT ulteriori
- **architetture di gestione orientate ai service**
- **alta scalabilità**

Cloud computing in continuità

Grid Computing

- Risolve problemi di grosse dimensioni con infrastrutture di calcolo parallelo



Utility Computing

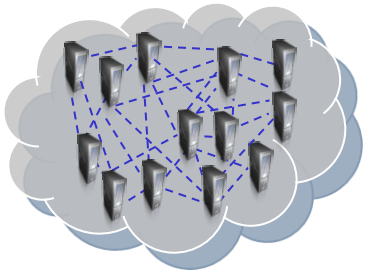
- Offre le risorse di computazione come servizio
- Tecnologie di virtualizzazione

Software as a Service

- Sottoscrizioni alle applicazioni di tipo network-based
- Service Oriented Architecture (SOA)
- Web 2.0

Cloud Computing

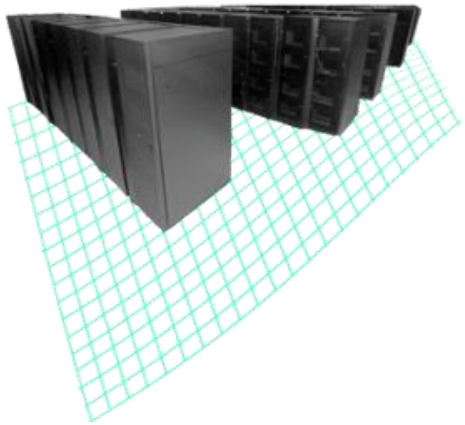
- Accesso ubiquo e in qualunque momento (anytime & anywhere) alle risorse IT erogate dinamicamente come servizio



Prima del Cloud: grid computing

Grid computing

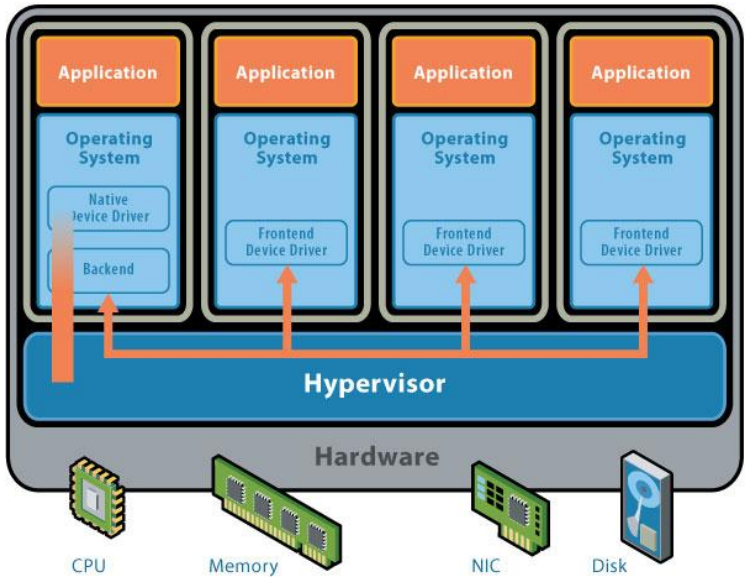
- Condivisione di **risorse eterogenee** (computer, software, dati, memoria e capacità computazionale, …) in **ambienti distribuiti altamente eterogenei** con l'obiettivo di **creare una virtual organization scalabile** (***secondo necessità!***)
- Interfacce (di gestione) spesso **a grana troppo fine, con basso livello di astrazione** e **non auto-contenute** ☹
- Ambiti applicativi **circoscritti** e **molto specifici** (calcolo parallelo per ambiti scientifico, ingegneristici, …)



Prima del Cloud: virtualizzazione

Virtualizzazione

- Tecnologie per la **virtualizzazione** (di sistema o ospitato), ad esempio in server farm**:** Vmware, Xen, …
- **Isolamento** e **personalizzazione infrastruttura e/o piattaforma SW** (O.S., ed eventuali applicativi)
- Strumenti per la **gestione** efficiente di **infrastrutture informatiche** (IBM Tivoli suite, Xen monitoring tools, …)



Prima del Cloud: utility computing

Utility computing

- Enormi capacità computazionali e di memorizzazione disponibili come **utility**, come ad esempio l'elettricità e l'acqua, e fornite secondo un modello di tipo "pay-per-use"
- "Computing may someday be organized as a public utility" - John McCarthy, MIT Centennial in 1961
- **Metered billing** (paghi per ciò che usi)
- Interfacce **semplici** e **standardizzate** per accedere alle risorse IT (es., come attaccarsi ad una presa)

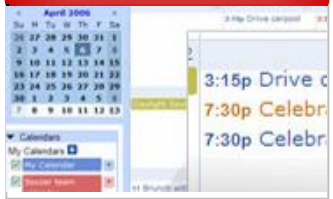
Risorse IT tradizionali: in-house con gestione interna

On-demand utility: plug-in, sottoscrizione pay-per-use

Web 2.0

- **Uso di protocolli asincroni non visibili all'utente per richiedere solo le informazioni necessarie e in modo asincrono: Asynchronous Javascript And XML (AJAX)**
- Nuovo modo di **usare i servizi Web** è stato anche accoppiato a **nuove applicazioni più facili da usare e collaborative** e **disponibili via Web in modo aperto, senza richiedere una installazione** da parte dei clienti interessati: arrivando a determinare un nuovo modo di lavoro, **molto cooperativo** (Software as a Service ☺)

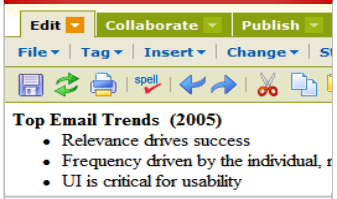
Office



https://www.google.com/hosted
http://smallbusiness.yahoo.com/email/
www.zimbra.com

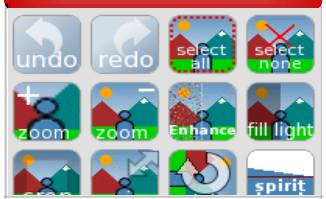
many others…

Word



www.writely.com
www.writeboard.com
www.inetword.com
many others…

Graphics



www.pxn8.com
www.pixoh.com
many others…

Database

www.dabbledb.com
www.Lazybase.com
www.quickbase.com
many others…

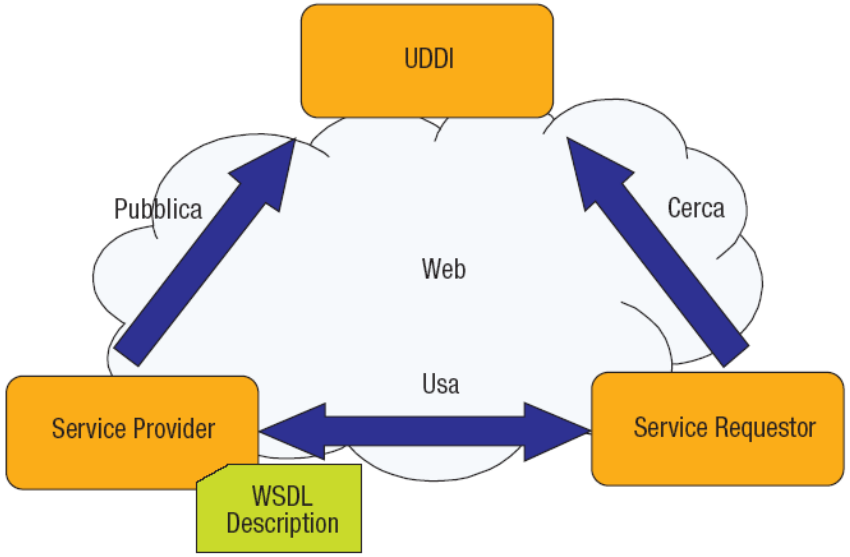
Contacts

www.
.com
others…

e molti altri…

Service oriented Architectures (SOA)

- **Interazione** tutta in termini di **contratto astratto** di **servizi offerti e richiesti**
- **Servizio** come **astrazione** di un **processo, risorsa o applicazione, di business** che possa essere **standardizzato** come **interfaccia** e possa essere **pubblicato e noto** (**riusabile e rintracciabile**, a **scarso accoppiamento** e **grana grossa**, a black box, senza stato, componibile, …)
- **Web Services** come architettura Web che fa propri e realizza diversi concetti di base delle architetture (astratte) SOA (WSDL, SOAP, UDDI, …)



Software as a Service (SaaS)

I servizi sono visti come **appliances disponibili via Web e disponibili in modo aperto, senza richiedere una installazione** da parte dei clienti interessati

- Sono **i provider dei servizi** che possiedono, gestiscono e forniscono le applicazioni, anche in modo congiunto
- I clienti possono **richiedere e consumare i servizi** in modo **privato o condiviso** in ogni momento ai provider che mantengono la responsabilità del codice e della definizione dei dati
- Le metriche di uso (e di accounting) sono tipicamente basate su sottoscrizione su base **pay-per-use**

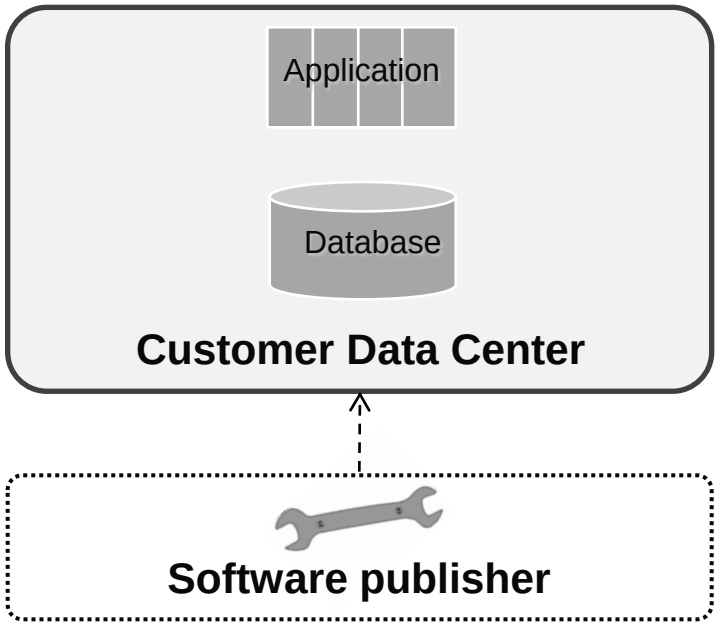
Vantaggi del Software as a Service

- **Responsabilità differenziata**: le organizzazioni si occupano del core business, e i provider si focalizzano sul servizio
- La **responsabilità del software** è limitata ai produttori: tutte le risorse, hardware, software, sono dalla parte del servizio
- **Sviluppo più efficiente**: la valutazione del cliente è immediata e si produce maggiore collaborazione
- **Interfacce efficienti Web 2.0**: migliore e più efficace uso delle risorse moderne del settore IT
- **Sempre versioni aggiornate**: i clienti ricevono la ultima versione del servizio sempre in linea
- Eliminazione del **costo della gestione prodotto per il cliente**: pay per use
- **Costi inferiori per il servizio**: il mantenimento della versione aggiornata costa meno in forma centralizzata sul produttore

SaaS: aree applicative

- **Applicazioni business verticali (es. ERP)**, anche specializzate e molto specifiche
- **Applicazioni general-purpose** senza particolari adattamenti (potenzialmente condivisibili)
  - provisioning self-service e personalizzazioni ad-hoc
  - applicazioni disponibili per molti utenti diversi
- **Applicazioni business B2B** e **domain-specific**
  - senza necessità di hosting e coinvolgimento di terze parti
- **Applicazioni Customer/Supplier**
  - Applicazioni dove la maggior parte degli utenti e dell'accesso avviene dall'esterno dell'organizzazionee dove l'accesso ubiquo via Web è critico e intrinseco
- **Applicazioni business even critical**,
  ma non quelle facenti parte del **core business**

Modelli: on-premise deployment lato client

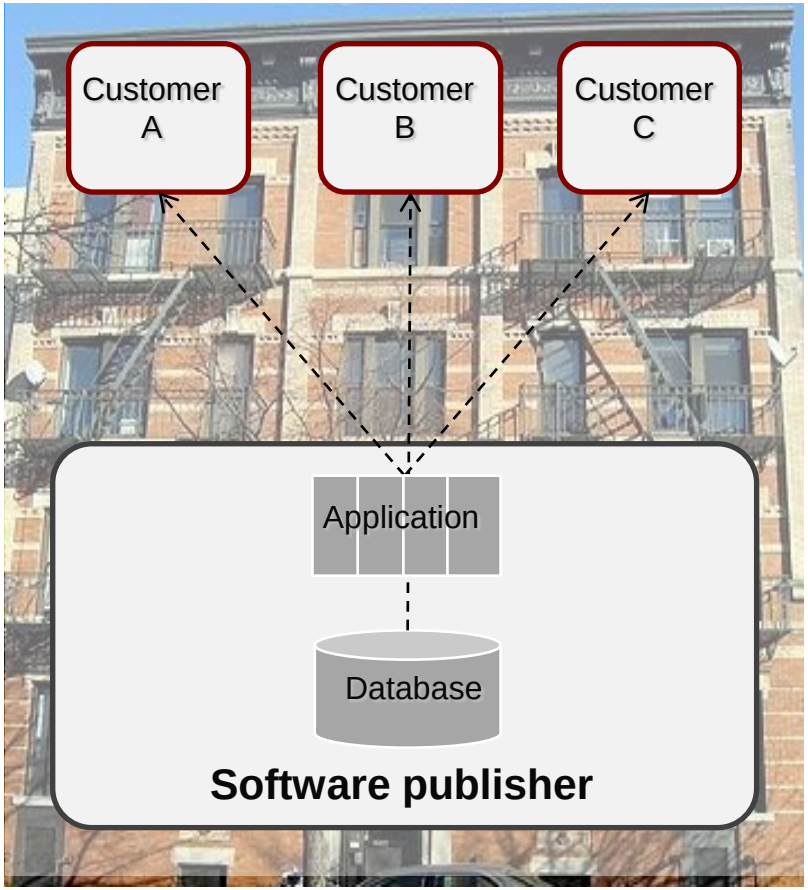


Caratteristiche

- Proprietà del software via acquisto o sviluppo
- Costi significativi di implementazione
- Adattabile e personalizzabile
- Difficile da modificare e da mantenere

Esempi gli sviluppi tradizionali presso il cliente

Modelli: applicazione SaaS condivisa (multi-tenant)



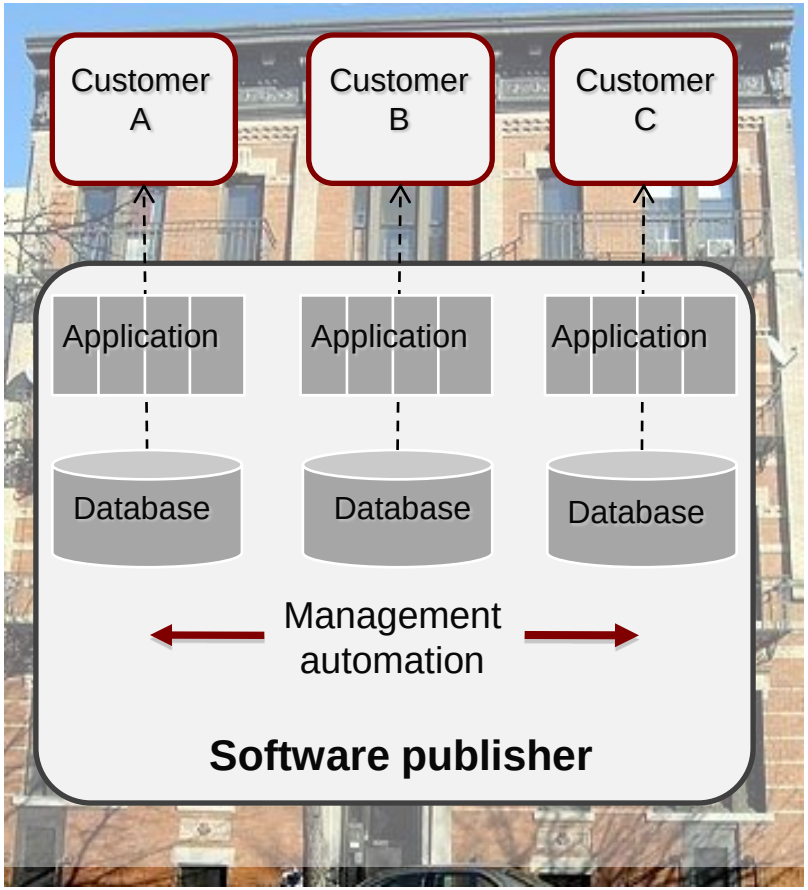
Caratteristiche

- Software mantenuto e sviluppato dal provider
- Molti clienti per una stessa applicazione
- Non troppo adattabile e personalizzabile

Esempi

- Salesforce.com
- Workday

Modelli: applicazione SaaS privata (single-tenant)



Caratteristiche

- Software mantenuto e sviluppato dal provider
- Ogni cliente ha una applicazione privata
- Servizio adattabile e personalizzabile con upgrade singoli

Esempi

- Service-now.com
- InteQ

Modelli SaaS in accrescimento

Possiamo considerare alcune caratterizzazioni ulteriori del servizio, con diverse possibilità di uso delle risorse

SaaS **S**oftware **a**s **a** **S**ervice

Le risorse sono le **applicazioni disponibili** attraverso accesso Web

PaaS **P**latform **a**s **a** **S**ervice

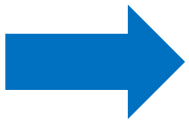
Le risorse sono le **intere piattaforme software** di esecuzione, per più programmi disponibili da utilizzare, anche in interazione tra loro

Iaas **I**nfrastructure **a**s **a** **S**ervice

Le risorse sono intese **in modo ampio e completo,** dalle **piattaforme hardware**, ai **sistemi operativi,** al **supporto** fino alle **applicazioni**: spesso usando virtualizzazione fino a Cloud Computing

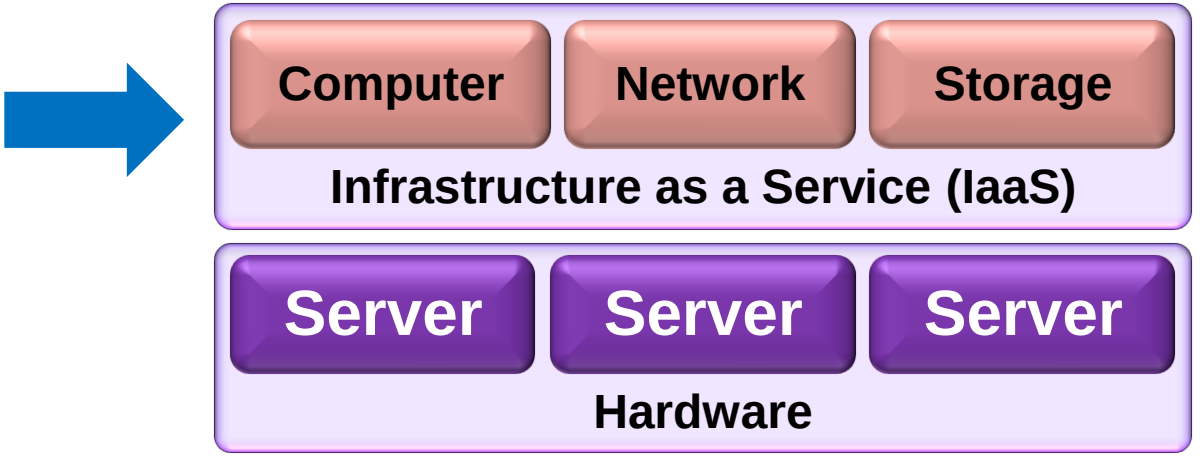
Architettura a livelli: IaaS, PaaS e SaaS

- Alla base del modello si trova l'architettura reale: componenti **hardware** e prodotti software



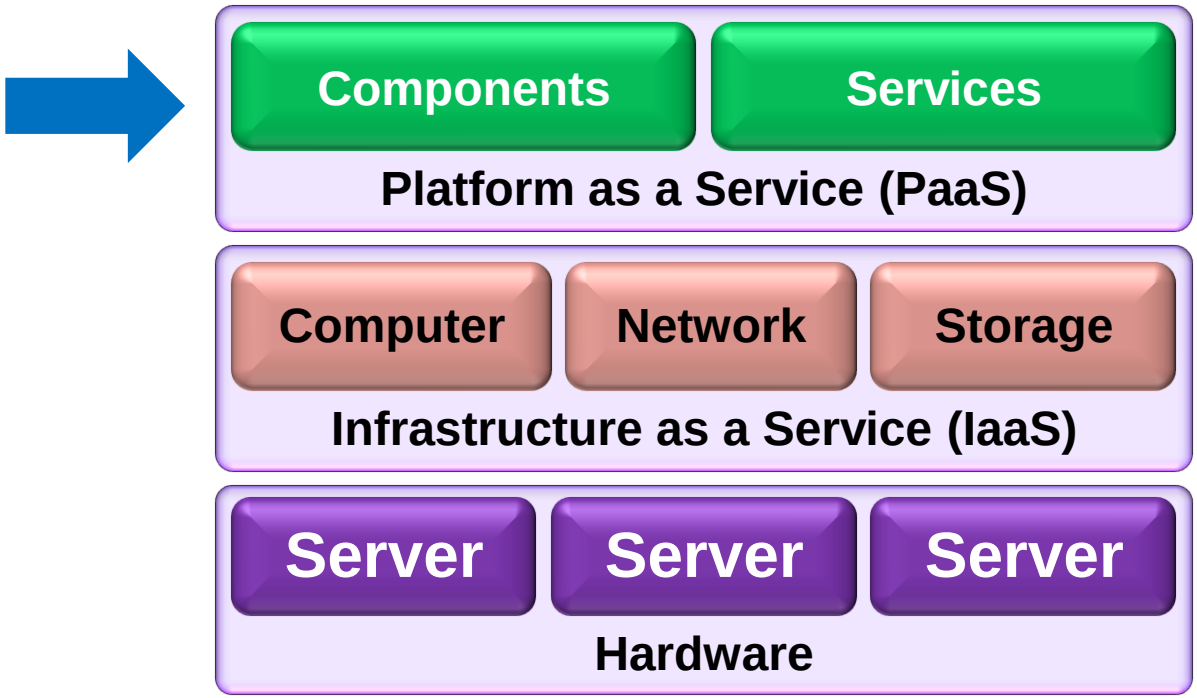
Architettura a livelli: IaaS, PaaS e SaaS

- **Infrastructure**: strato che permette di distribuire l'infrastruttura di servizio Cloud tipicamente realizzato dalla piattaforma di virtualizzazione



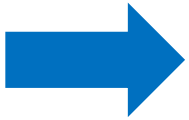
Architettura a livelli: IaaS, PaaS e SaaS

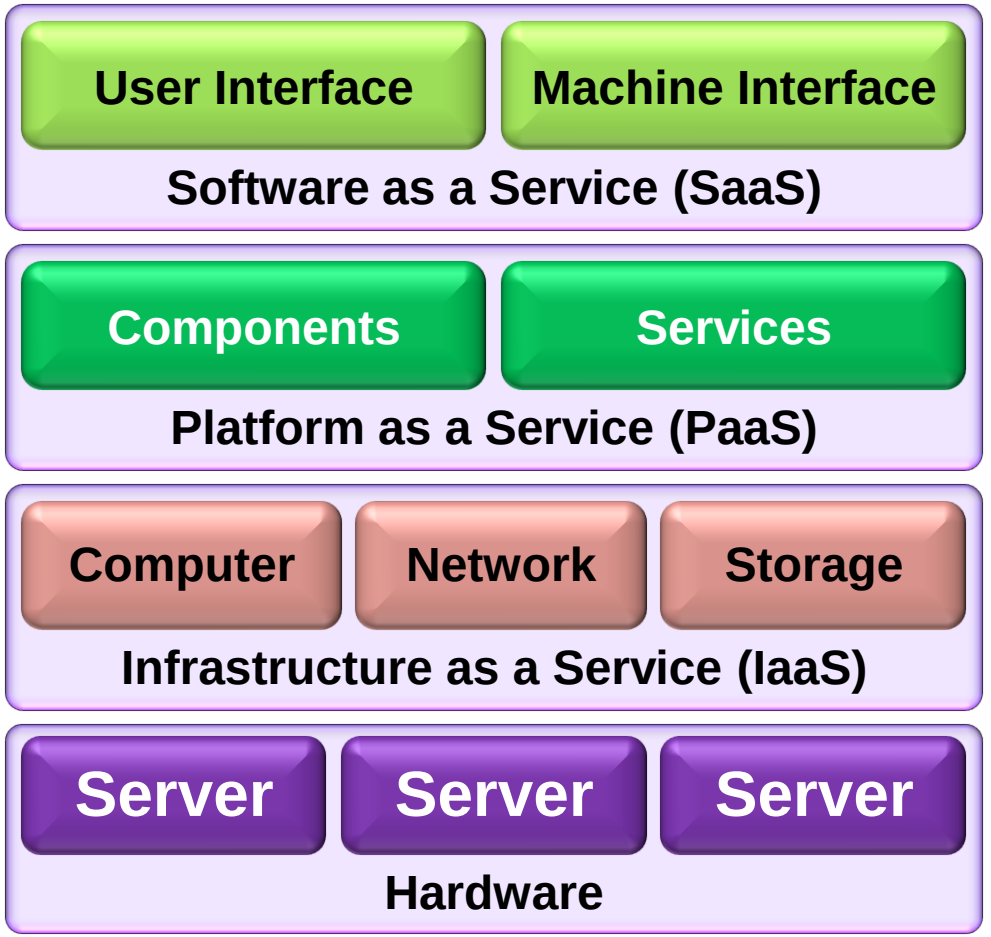
- **Platform:** strato che consente di fornire al livello superiore una serie di componenti e servizi accessibili da remoto



Architettura a livelli: IaaS, PaaS e SaaS

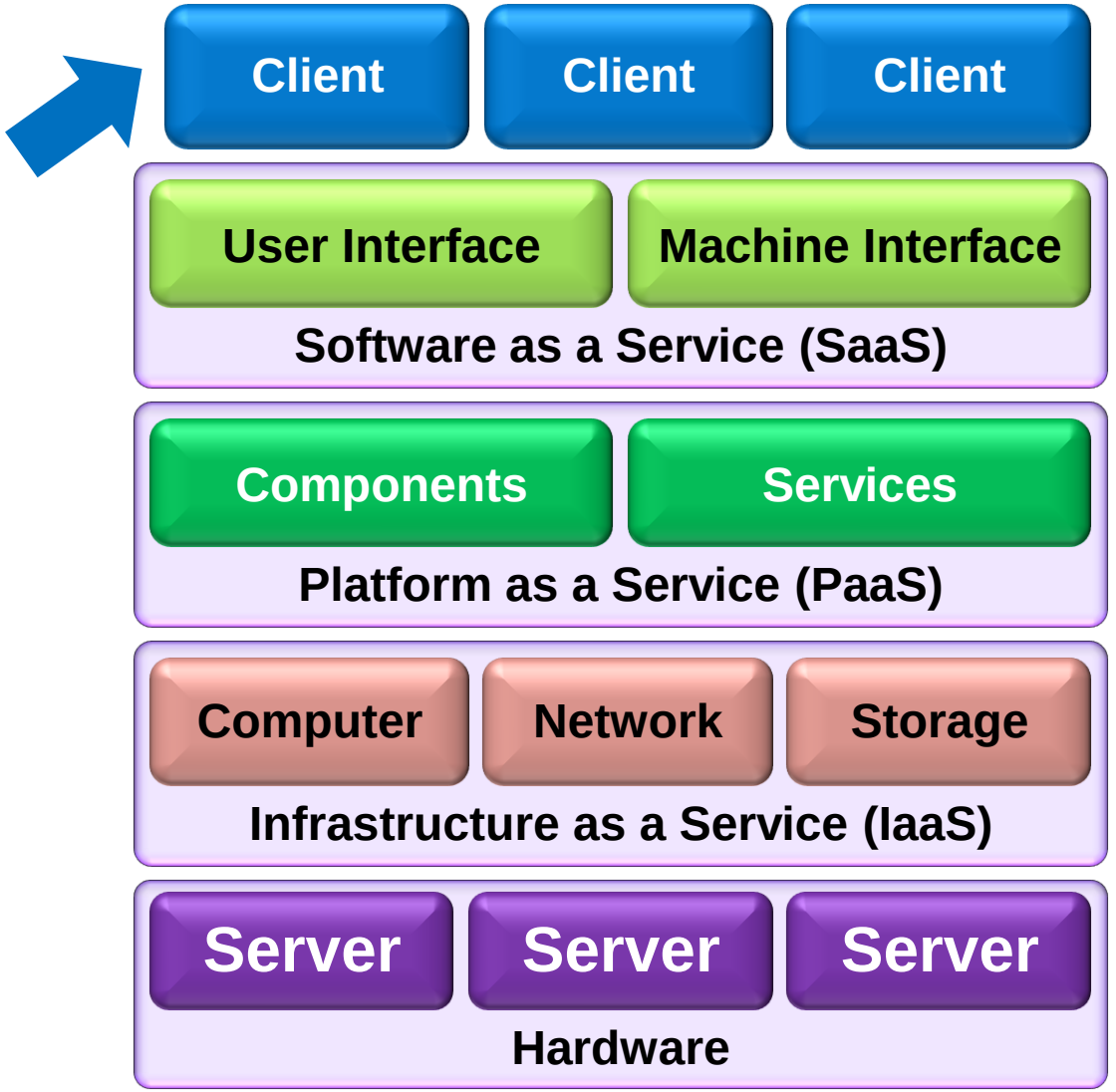
- **Application**: strato su cui possono esser installate alcune applicazioni, accessibili via Internet tramite lo stesso sistema Cloud



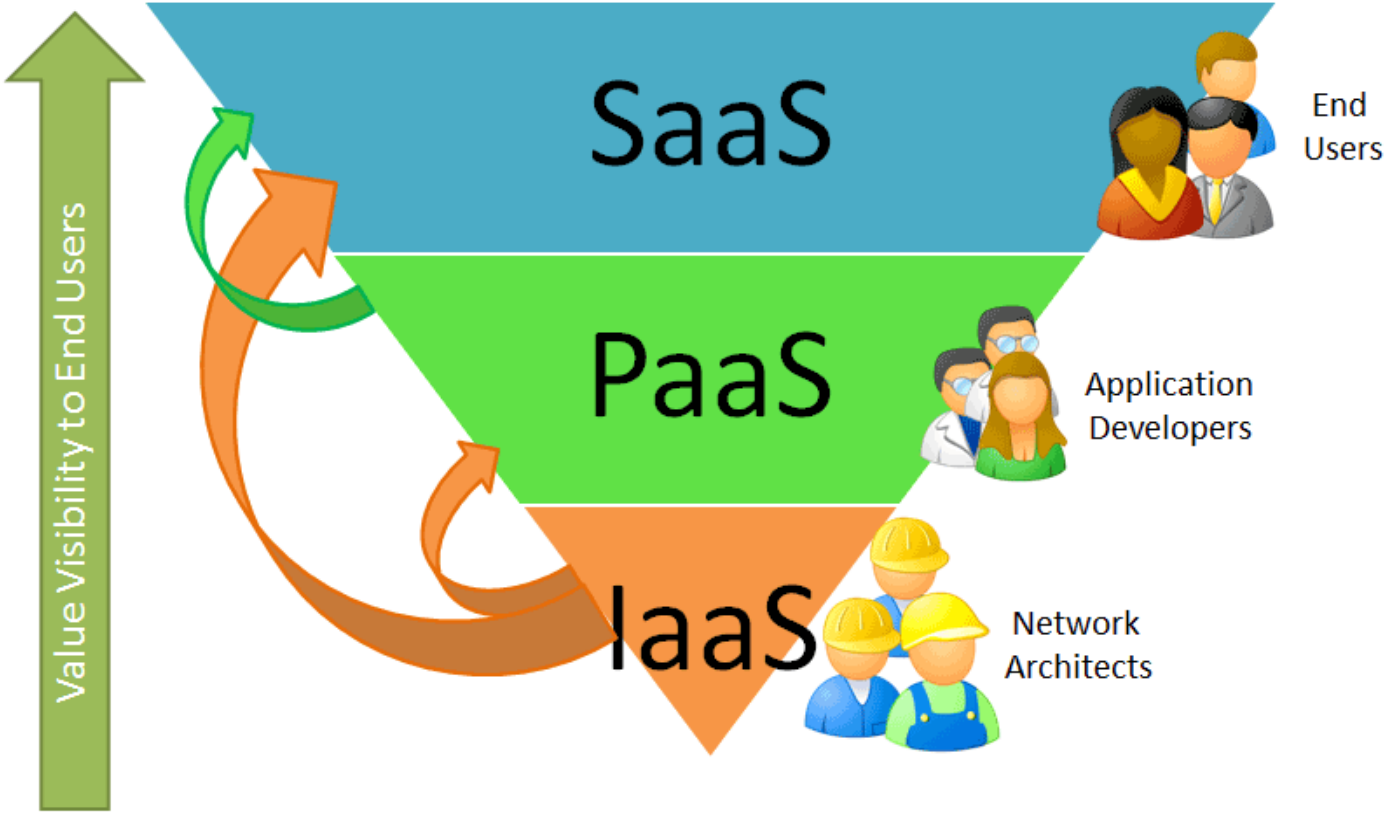


Architettura a livelli: IaaS, PaaS e SaaS

- Insieme di software **Client** per accedere al sistema. Tali applicazioni vengono eseguite sul computer fisico remoto di proprietà del cliente finale, e sono in grado di comunicare con le interfacce messe a disposizione dal Cloud



Architettura a livelli: principali attori



Alcuni esempi di SaaS e *aaS

Esempi molto diffusi e aperti

SaaS - servizi singoli anche con qualche coordinazione

Dai desktop come **Google Apps** (Gmail, Google calendar e docs), **Microsoft Window live** (Hotmail, Messenger, …) fino ai motori di ricerca, Google, Yahoo,
Ai **social network, come** Facebook, LinkedIn, Twitter, …

PaaS - tipicamente Web service e piattaforme sviluppo

Possono essere usati all'interno di e interagire con altre applicazioni, come **Google App Engine (GAE)**, **Google Maps**, **Microsoft Azure**

Iaas - emergono alcuni esempi sperimentali

Ci sono molti esempi anche avanzati di infrastrutture e server virtuali o meno, **Amazon Web Services** (Simple Storage Service, **S3**) e **Elastic Compute Cloud** (**EC2**), **OpenStack**, fino ad alcuni **desktop di controllo e monitoraggio** dell'esecuzione, come Sun global desktop, Zimdesk, …

Technology wrap up

SaaS

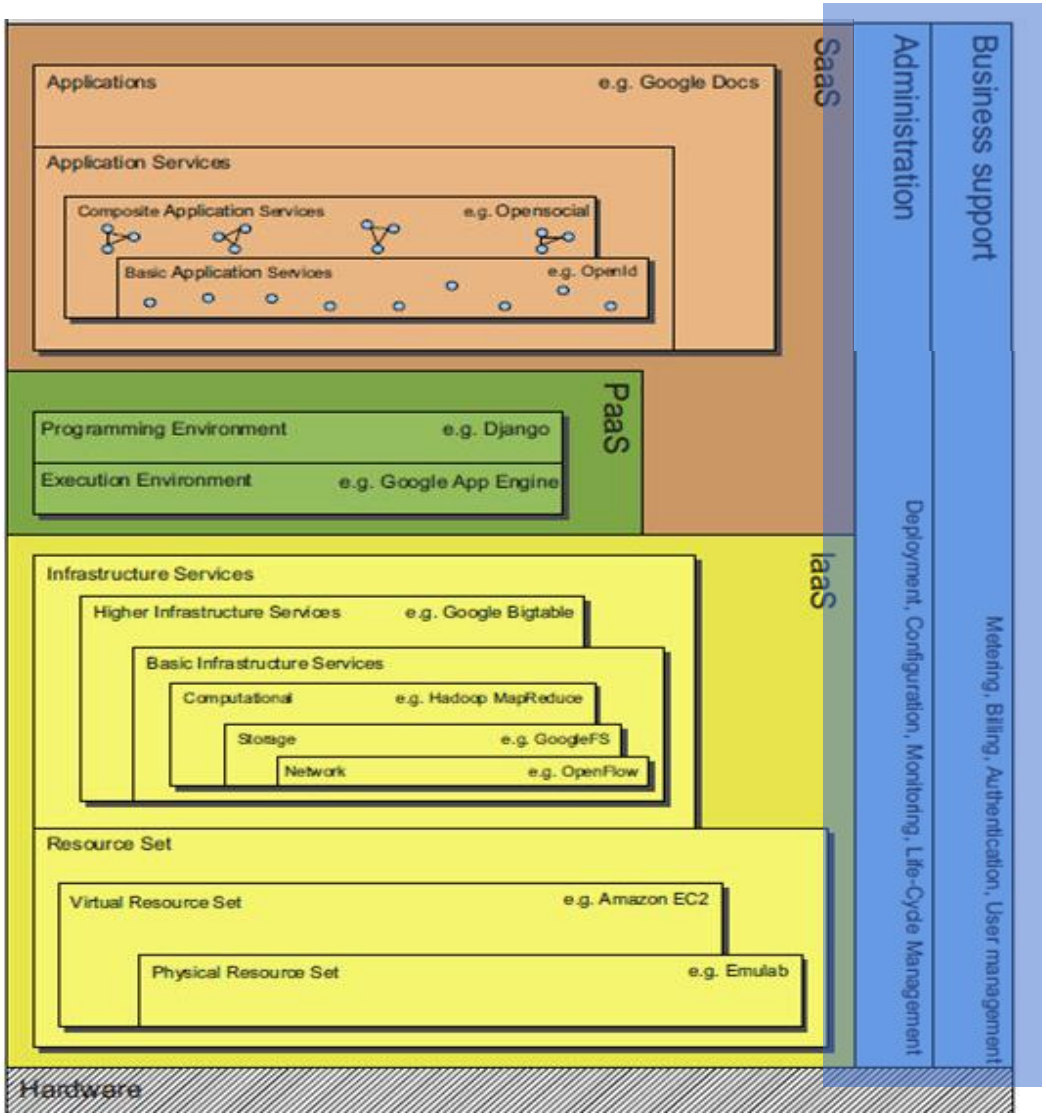
- Google docs

PaaS

- Google App Engine

IaaS

- Google Bigtable
- Hadoop MapReduce (Yahoo)
- GoogleFS
- Amazon EC2
- OpenStack



Funzionalità di supporto verticali a tutti i livelli

Il Cloud è… più della somma delle parti

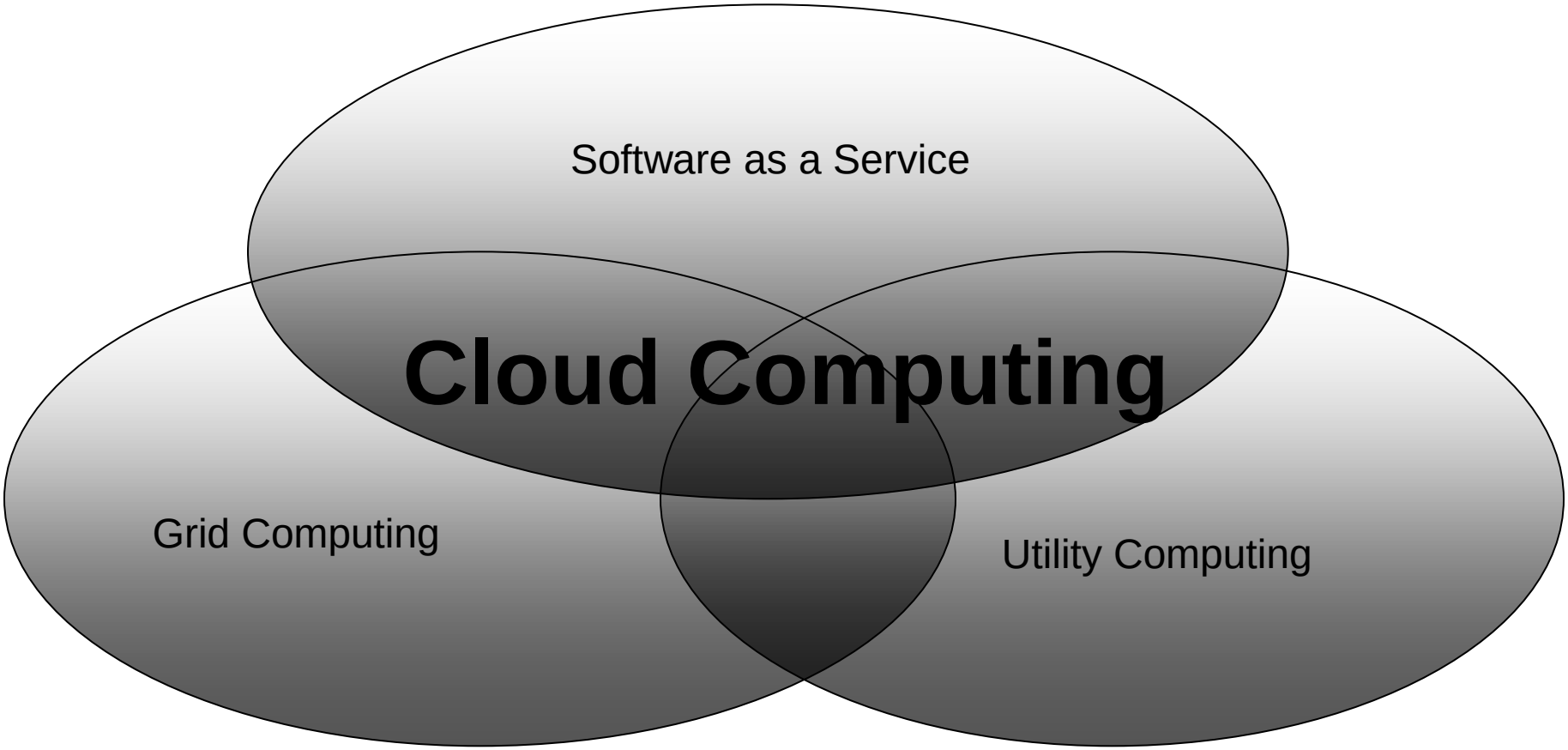
- **Grid Computing**
  - Cloud è più di un **insieme di risorse IT** perché offre i meccanismi per gestire quelle risorse
    - Provisioning, change requests, workload balancing, monitoring
  - Cloud computing è un'infrastruttura che si trova al di sopra del data center per **aumentarne l'efficienza**
- **Utility Computing**
  - Cloud facilita ulteriormente le funzionalità di **deploy, gestione, e scalabilità** dei servizi rendendoli disponibili "aaS"
  - Cloud richiede **disponibilità 24/7, anytime & anywhere** (e la rete??)
  - Cloud deve **adattarsi in modo elastico**
- **SaaS**
  - Cloud impone due modelli: chi **gestisce** SaaS **vs.** chi **usa** SaaS
  - Cloud **accorcia il time-to-market** e **facilita la composizione di ecosistemi di servizi**
  - Cloud può avere ricadute forti anche **a livello organizzativo** → modello più **cooperativo** e in **continuo accrescimento**

Il Cloud… introduce problemi nuovi

- **Affidabilità e disponibilità**
  - Per la parte tecnologica (nel public Cloud) ci si affida all'esterno
  - **Amazon US outage (21→24 Aprile 2011)**, anche alcuni articoli…
    - "Amazon EC2 Outage Hobbles Websites: Worst Cloud Computing Disaster"
    - "Amazon's lengthy cloud outage shows the danger of complexity"
- **Sicurezza e problematiche di tipo legislativo**
  - Esternalizzazione dei dati sul Cloud
  - Dematerializzazione → **dove sono i miei dati???**

  (con soluzioni parziali già emergenti…)
- **Gestione e integrazione dei servizi**
  - Si possono ricreare **silos SaaS**
  - Integrazione di sistemi Cloud diversi → **standardizzazione API**, protocolli per la migrazione delle VM, ecc.

Evoluzione del Cloud

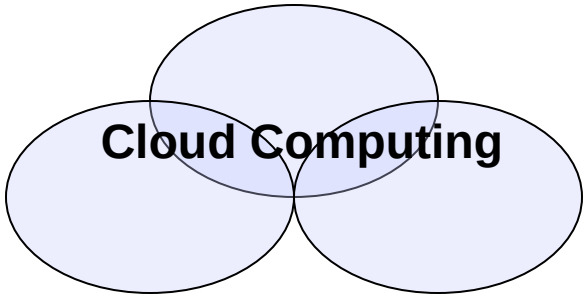


Tecnologia e Business

Virtualization
Scalability
Grid Computing
…

SaaS **UaaS**

Technology **Business** **End users**

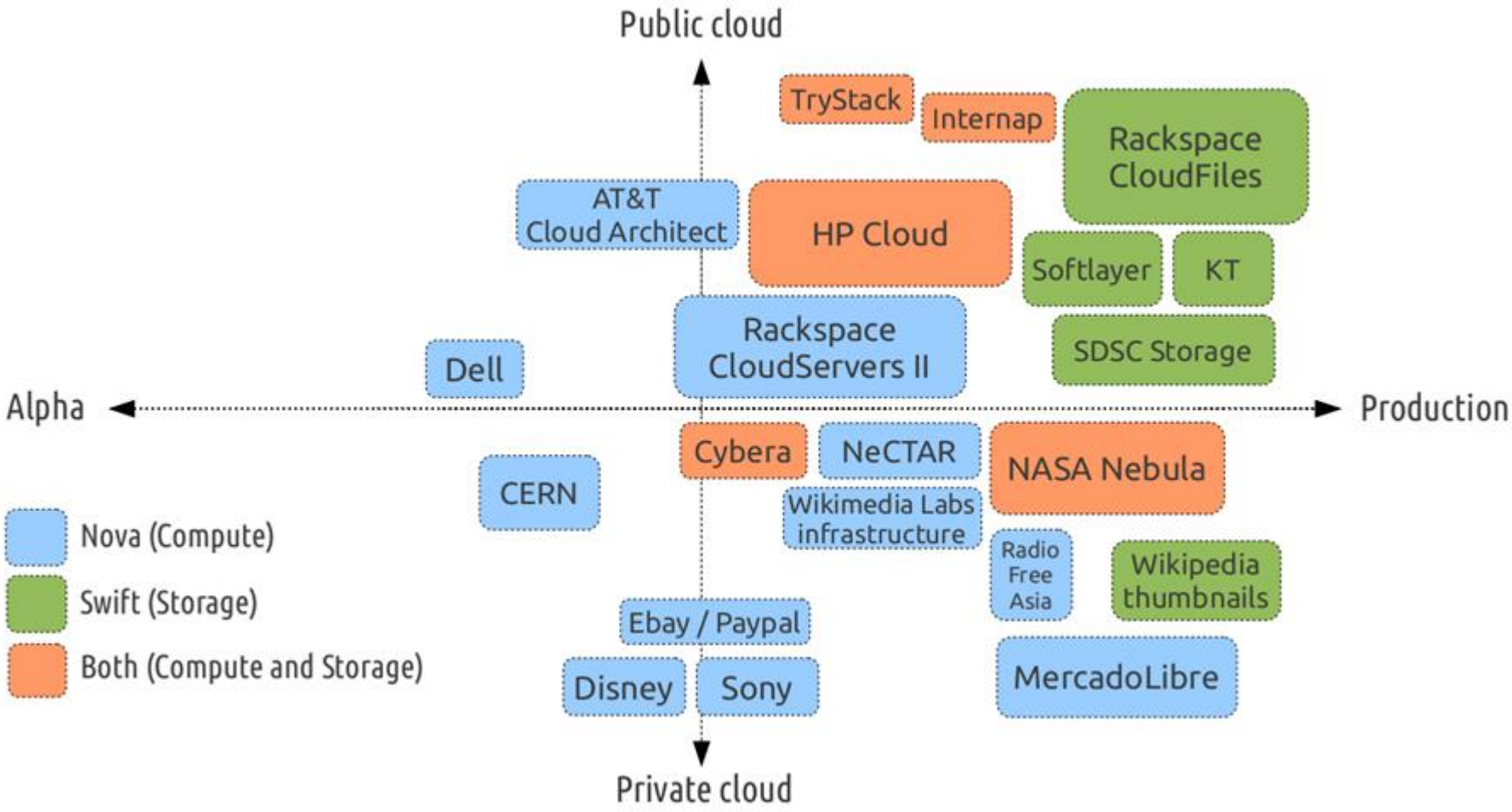


Modelli di deployment del Cloud

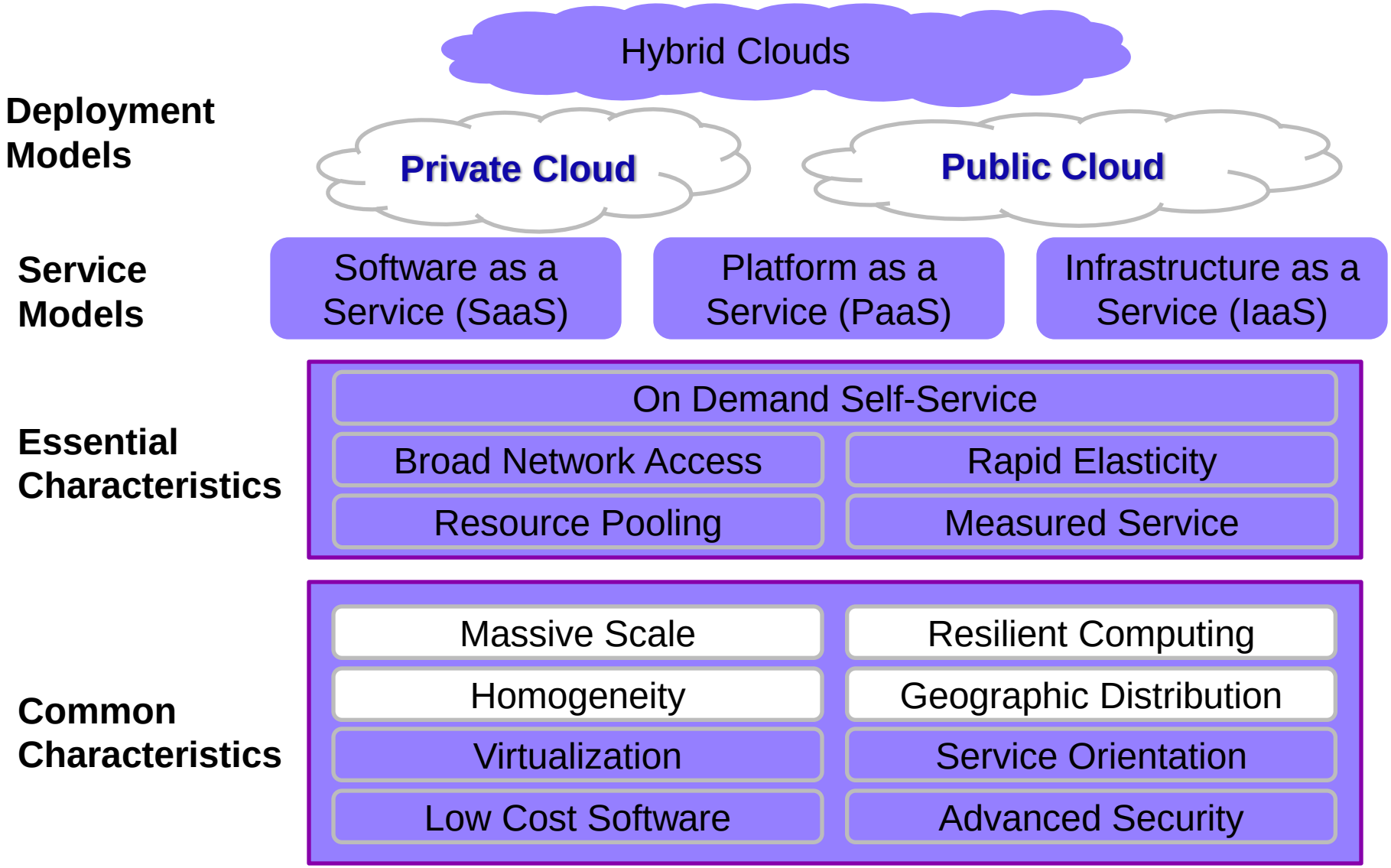
Tre modelli principali

- **Private Cloud**
  - l'azienda possiede o affitta l'infrastruttura
- **Public Cloud**
  - venduto al pubblico, mega-scale infrastructure
- **Hybrid Cloud**
  - Una compoizione di due o più Clouds (con modelli anche diversi)

Deployment conosciuti



NIST Cloud


Deployment Models
Hybrid Clouds
Private Cloud
Public Cloud
Service Models
Software as a Service (SaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)
Essential Characteristics
On Demand Self-Service
Broad Network Access
Rapid Elasticity
Resource Pooling
Measured Service
Common Characteristics
Massive Scale
Resilient Computing
Homogeneity
Geographic Distribution
Virtualization
Service Orientation
Low Cost Software
Advanced Security

Cloud computing: reality check

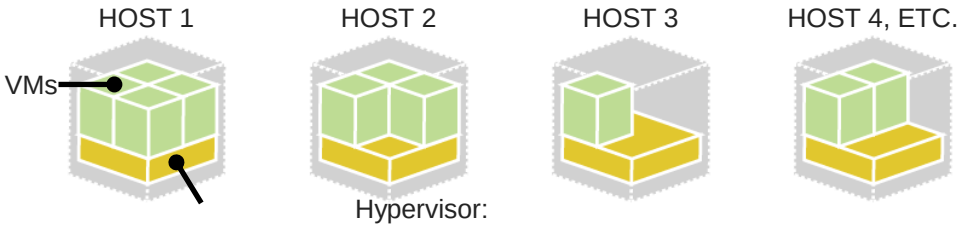
- **Google docs, salesForce, …** Web applications (Web 2.0) molto diffuse e usate
- **Google App Engine** Google App Engine, sandbox for management e security, vari linguaggi (Python e Java)
- **HP/Yahoo/Intel Open Cirrus Test Bed**: virtualized images, Xen, simple SLA console
- **Amazon Elastic Computing – EC2**: Amazon WS (SLA), Amazon Machine Image (DB+Software and middleware+OS) e Xen, Dynamo
- **Openstack**: virtualized images (DB+Software e middleware+OS), Xen, Swift (storage), Quantum (virtualized network)
- **IBM Cloud**: virtualized images (DB+Software and middleware+OS), Xen, Tivoli (monitoring and management), simple SLA console
- **Microsoft Azure**: soluzione Microsoft (*presentazioni seguenti* ☺)

Principali obiettivi del Cloud: punto di vista dell'infrastruttura

- Come si possono fornire in modo flessibile e veloce le risorse computazionali per promuovere **rapid prototyping**?
- Come organizzare il deployment delle per **scalare** in modo da fare fronte a domande crescenti nel tempo?
- Come gestire e monitorare 100,000 macchine con **intervento umano minimo**?
- Come possiamo aumentare l'**efficienza** nell'uso di tutte le risorse computazionali del data center?

Cloud: virtualizzazione delle risorse

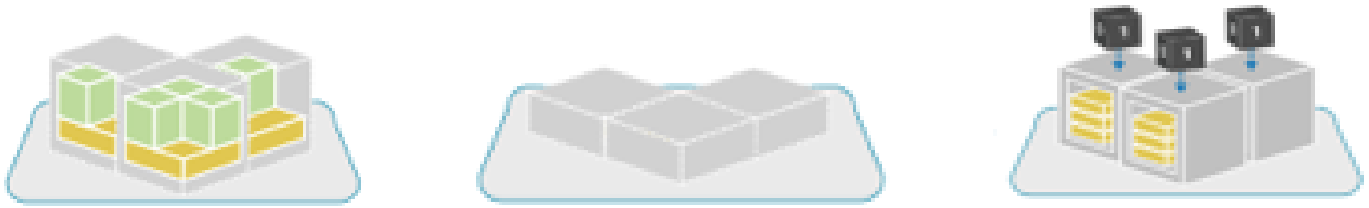
- Primo step: **virtualizzazione dei server** come Virtual Machine (VM)



- Gli Hypervisor forniscono uno strato di astrazione tra hardware e software
- Astrazione hardware per ciascun server
- **Migliore utilizzo di risorse** per ciascun server (fisico)

Cloud: virtualizzazione delle risorse

- Secondo step: **virtualizzazione** della **rete** e dello **storage**



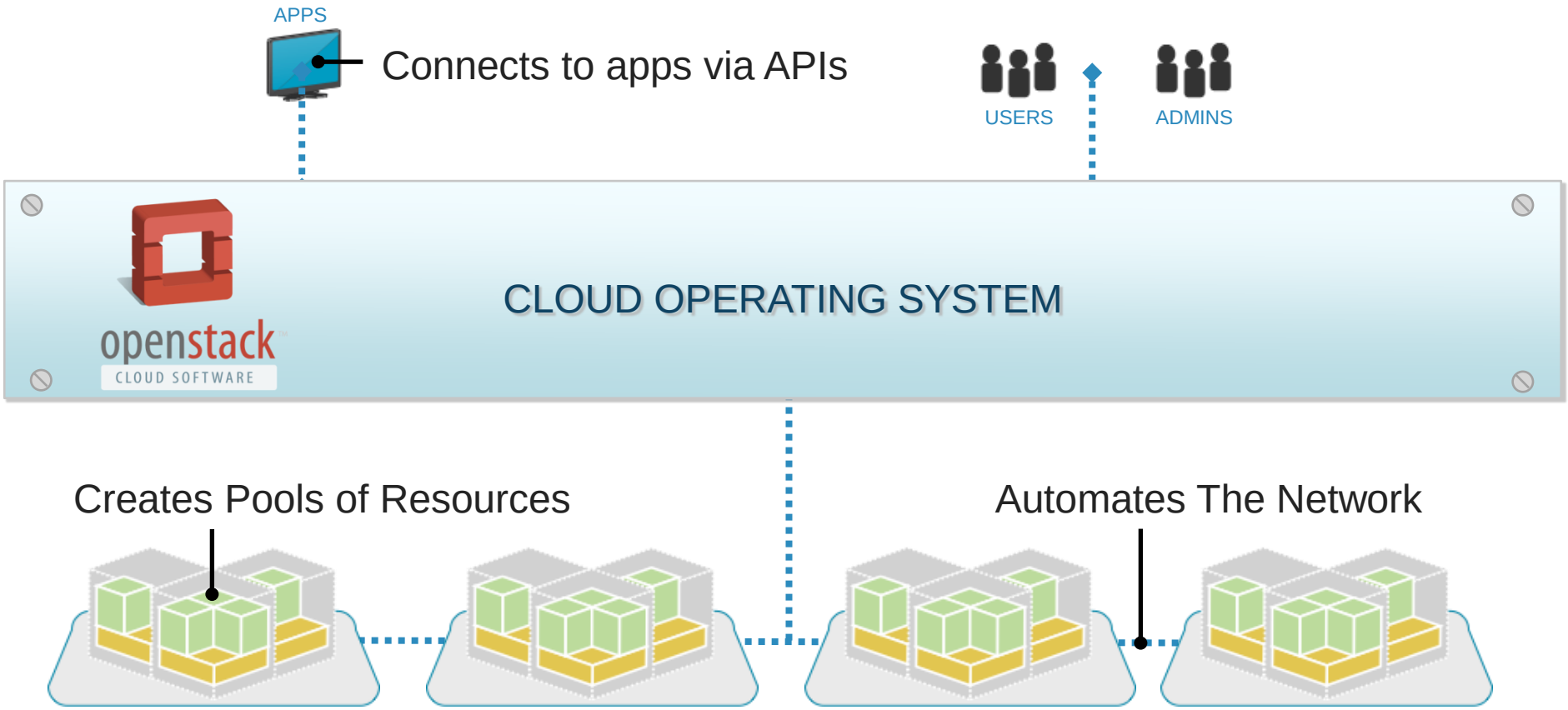
Compute Pool
Virtualized Servers

Network Pool
Virtualized Networks

Storage Pool
Virtualized Storage

- Pool di risorse disponibili per diverse applicazioni
- Flessibilità ed efficienza

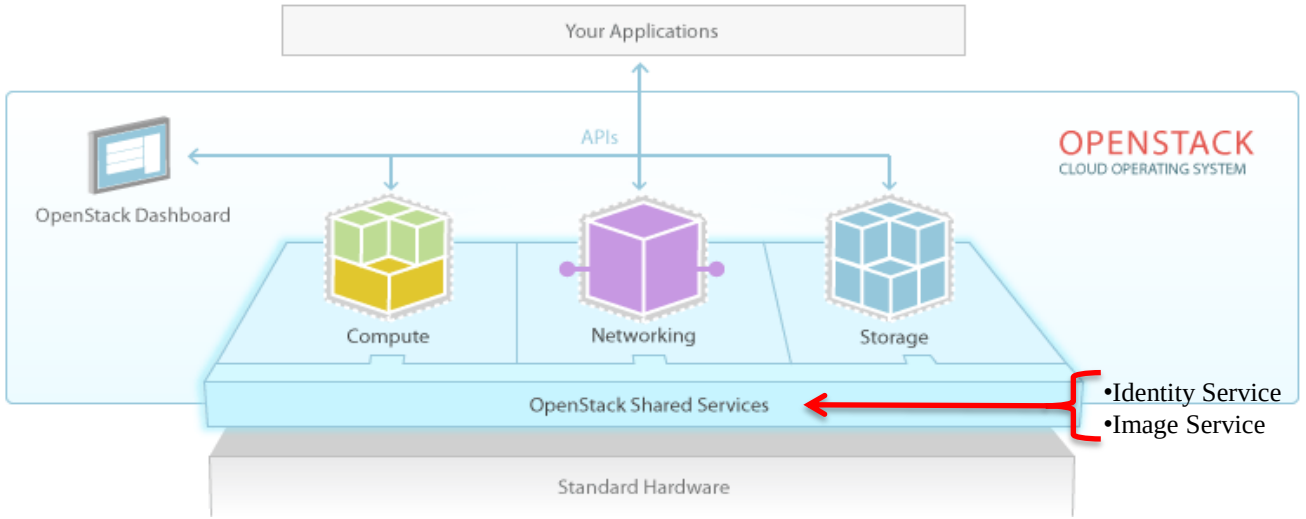
Architettura logica di un'infrastruttura Cloud



Un esempio concreto: OpenStack

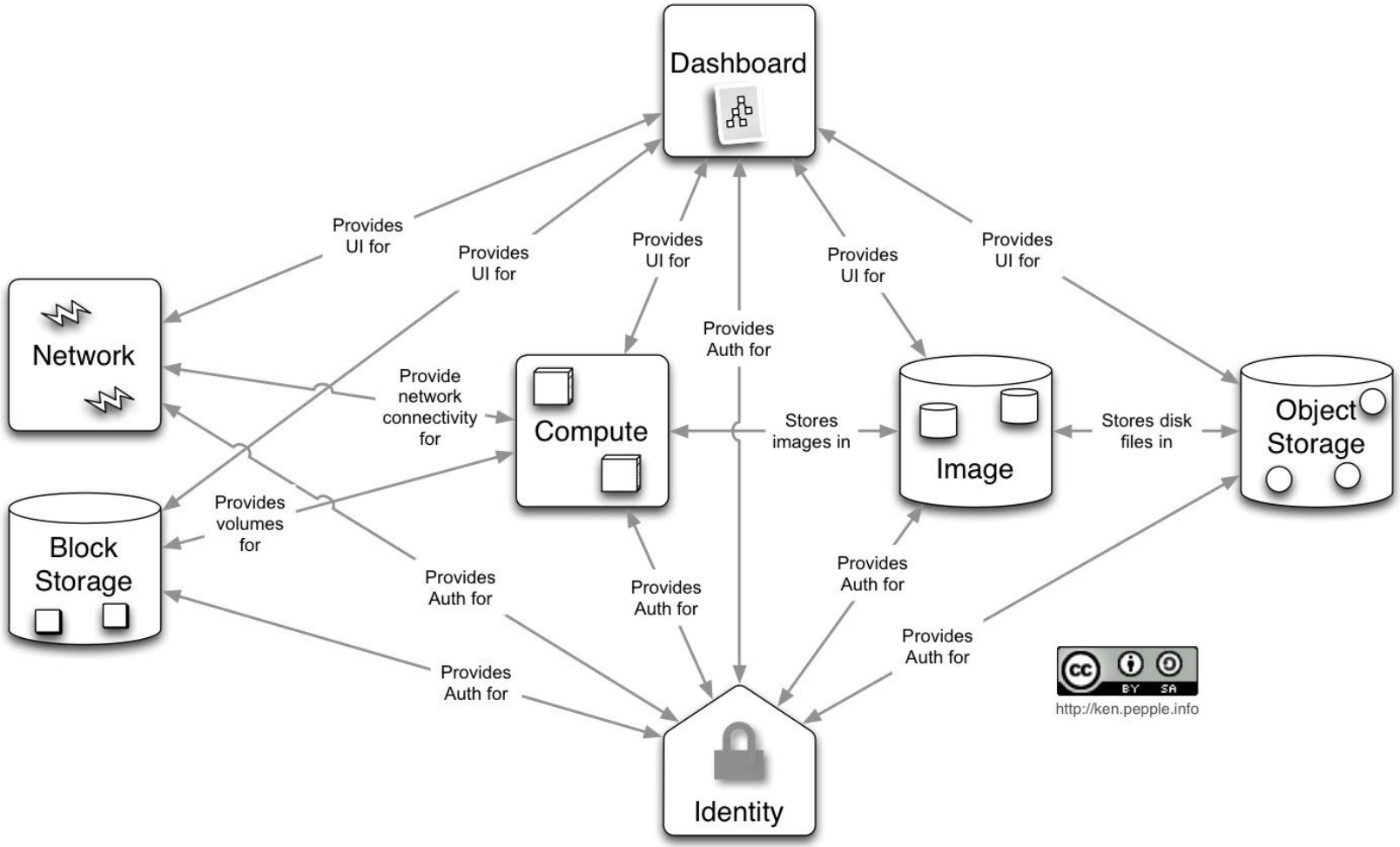
- **OpenStack**
  - Progetto fondato dalla NASA e Rackspace nel 2010
  - Attualmente supportato da 200 aziende e 9429 persone
  - Ultima release Grizzly, 4 Aprile 2013
- Allineato al ciclo di release di **Ubuntu**, Apr. / Ott.
- Open-source vs Amazon e Vmware
- Primi deployment anche pubblici (es. HP Cloud) basati su OpenStack e utilizzo da parte di grosse per ricerca aziende (es. IBM) → **prodotto open-source maturo**

Architettura a componenti di OpenStack

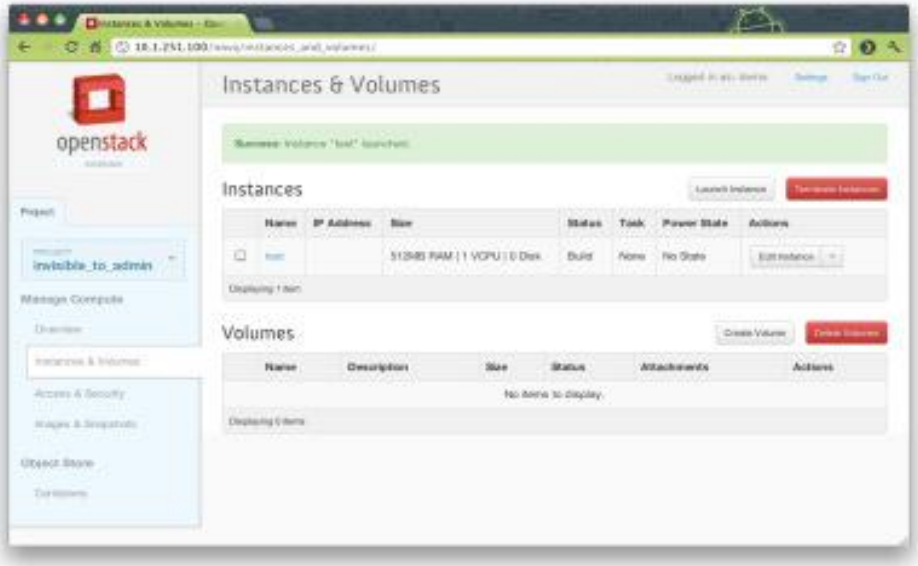


- **Dashboard**: **portale web di controllo** per amministratori e utenti
- **Identity**: servizio di **autenticazione** unificato su tutto il sistema
- **Object Storage**: sistema **storage ridondante** su larga scala di oggetti statici
- **Image Service**: servizio per la memorizzazione, il recupero, il discovery, la registrazione e la consegna di **immagini di VM**
- **Compute**: componente per il **deployment** e la gestione su larga scala di **VM**
- **Networking**: componente per la gestione della **virtualizzazione di rete**

Architettura logica di OpenStack



Horizon - Dashboard

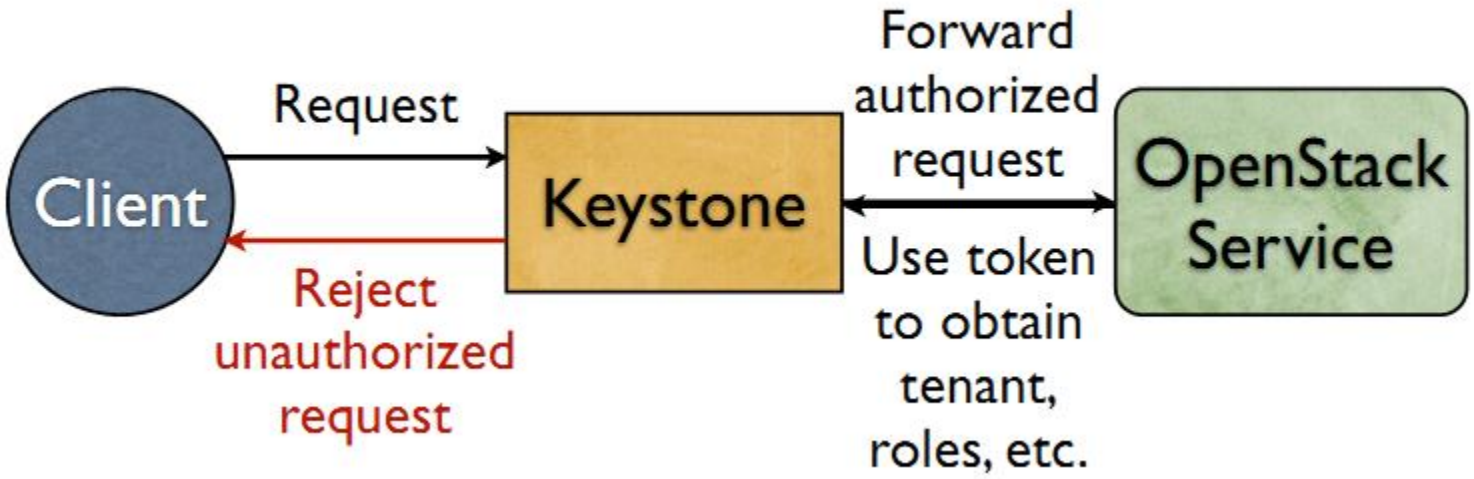


Fornisce un'interfaccia modulare **web-based** per tutti i servizi OpenStack, attraverso la quale è possibile **eseguire alcune operazioni** come lanciare un'istanza, assegnare indirizzi IP, caricare immagini di VM, settare politiche di controllo degli accessi, ecc.

Keystone – Authentication and Authorization

- Keystone è un framework per **l'autenticazione** e **l'autorizzazione** verso tutti i servizi di OpenStack
- Permette di aggiungere utenti a dei gruppi (anche chiamati **tenants**) e di **gestirne i relativi permessi**. I permessi, ad esempio la possibilità di lanciare e terminare VM
- Offre 4 servizi principali:
  - **Identity**: informazioni utente per l'autenticazione
  - **Token**: dopo il login, sostituisce il meccanismo di autenticazione basato su password
  - **Catalog**: mantiene un registro degli endpoint di servizi OpenStack
  - **Policy**: gestione di diversi livelli utente

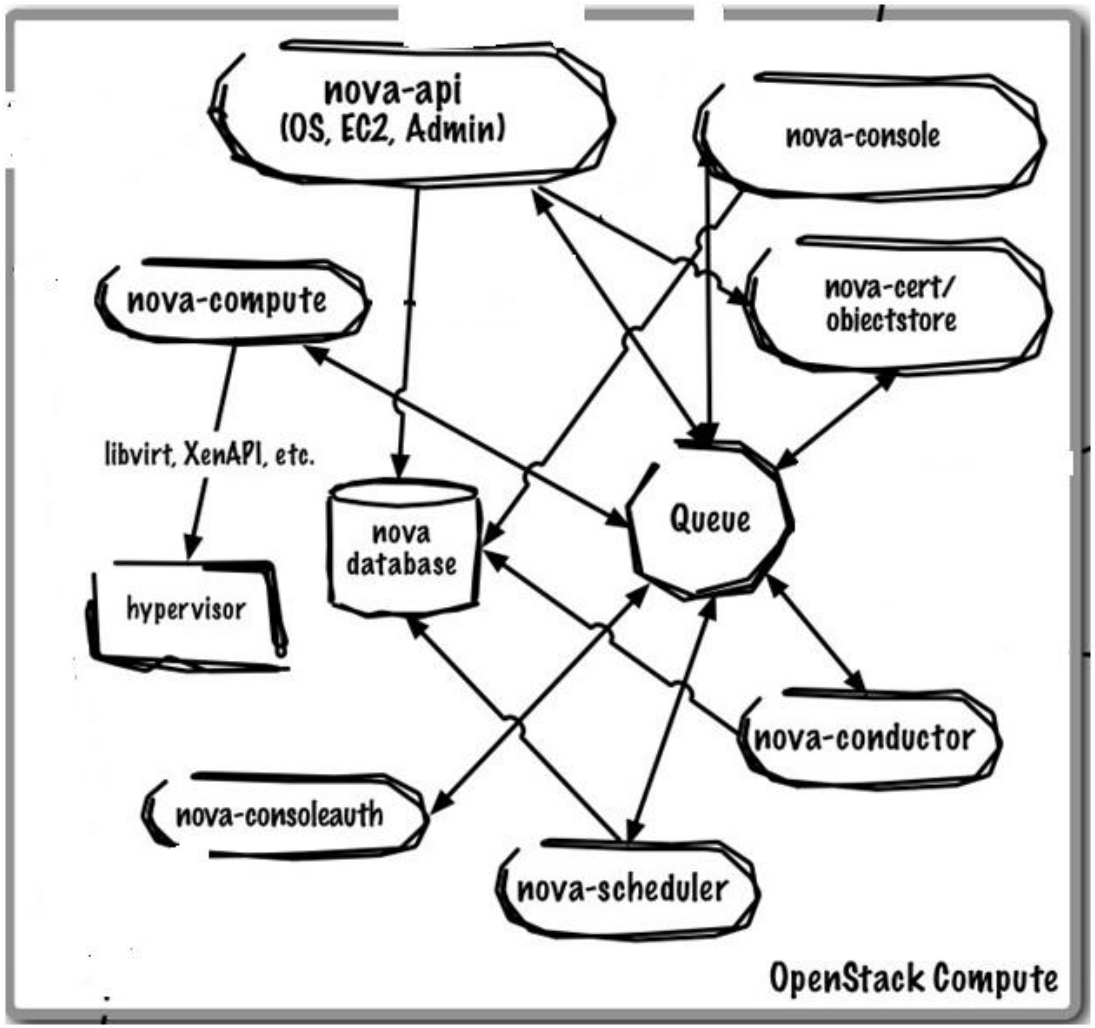
Keystone


Client
Request
Reject unauthorized request
Keystone
Forward authorized request
Use token to obtain tenant, roles, etc.
OpenStack Service

Nova - Compute

- Fornisce VM su richiesta
- Fornisce e gestisce grandi infrastruttura con **molte VM**
- Risorse computazionali **accessibili attraverso CLI dagli sviluppatori**
- Risorse computazionali **accessibili attraverso interfacce web** dagli **utenti** e dagli **amministratori**
- Architettura progettata per **scalare orizzontalmente** su hardware standard
- Supporto verso **numerosi hypervisor** (es. KVM, XenServer, ecc.)

Nova – Componenti (1)



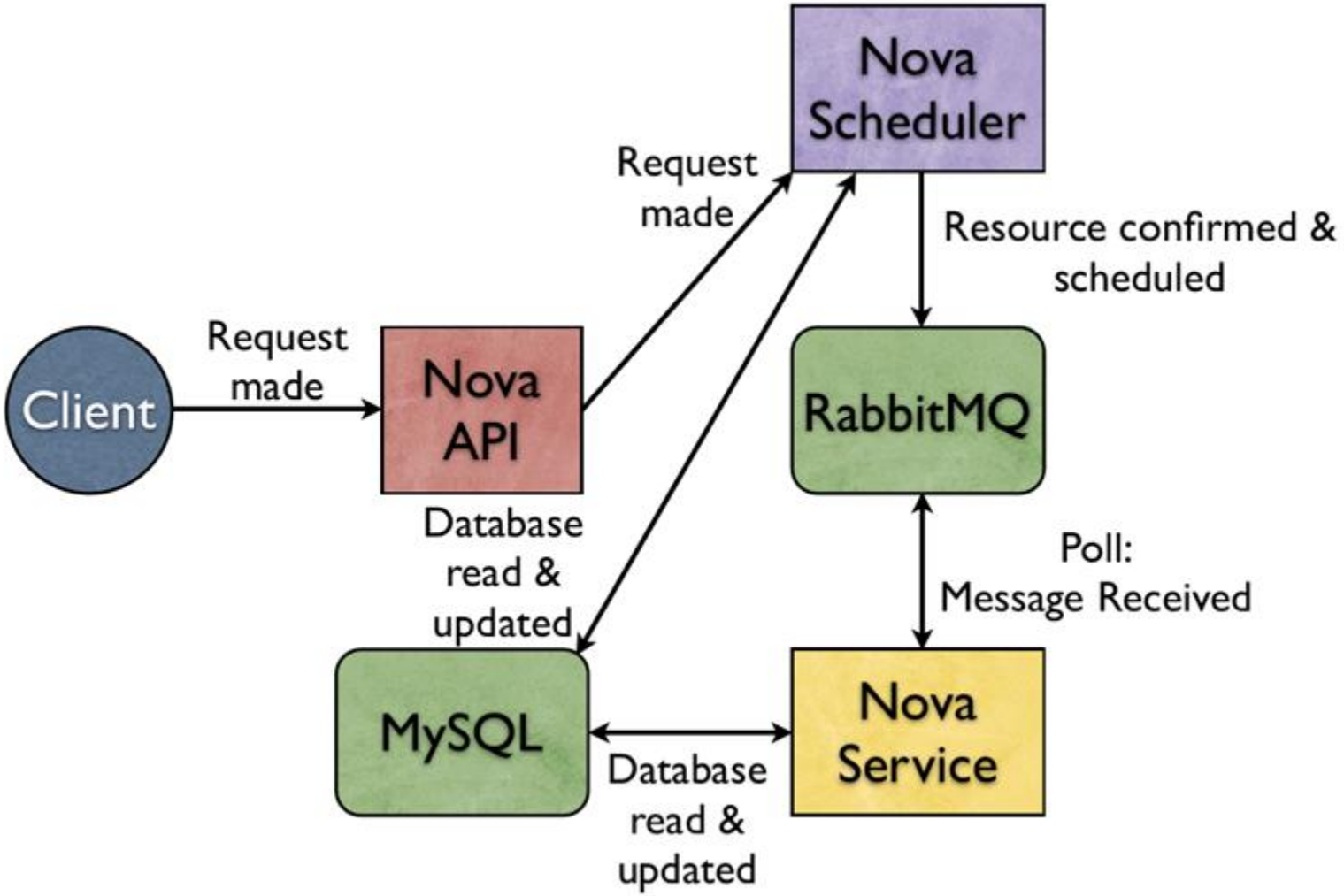
Nova – Componenti (2)

- **nova-api:** RESTful API web service per la **gestione dei comandi** per l'interazione con Nova
- **nova-compute: demone locale ai nodi fisici ospitanti le VM** per la **creazione** e la **distruzione** di istanze di VM attraverso le API dell'hypervisor
- **nova-scheduler:** coordina tutti i servizi e **determina il mapping "VM-host fisico"**, cioè quali host debbano fornire le risorse richieste
- **nova database:** memorizza parte degli stati configuration-time e run-time dell'infrastruttura Cloud
- **queue: servizio MOM** per l'invio e la consegna di messaggi tra i diversi demoni di OpenStack, di default realizzato con RabbitMQ

Nova – Componenti (3)

- **nova-console, nova-novncproxy e nova-consoleauth:** forniscono, attraverso un proxy, l'accesso via **CLI che permettono di eseguire chiamate alle API di OpenStack** e **alle console delle istanze virtuali**
- **nova-network: gestisce la configurazione della rete** (es. cambiare le regole di iptables, creare interfacce di bridging, ecc.) → questa funzionalità sta migrando verso il servizio Quantum
- **nova-volume:** gestisce la **creazione** e il **collegamento** di **volumi persistenti** ad istanze virtuali → questa funzionalità sta migrando verso il servizio Cinder.

Nova: schema generale di interazione

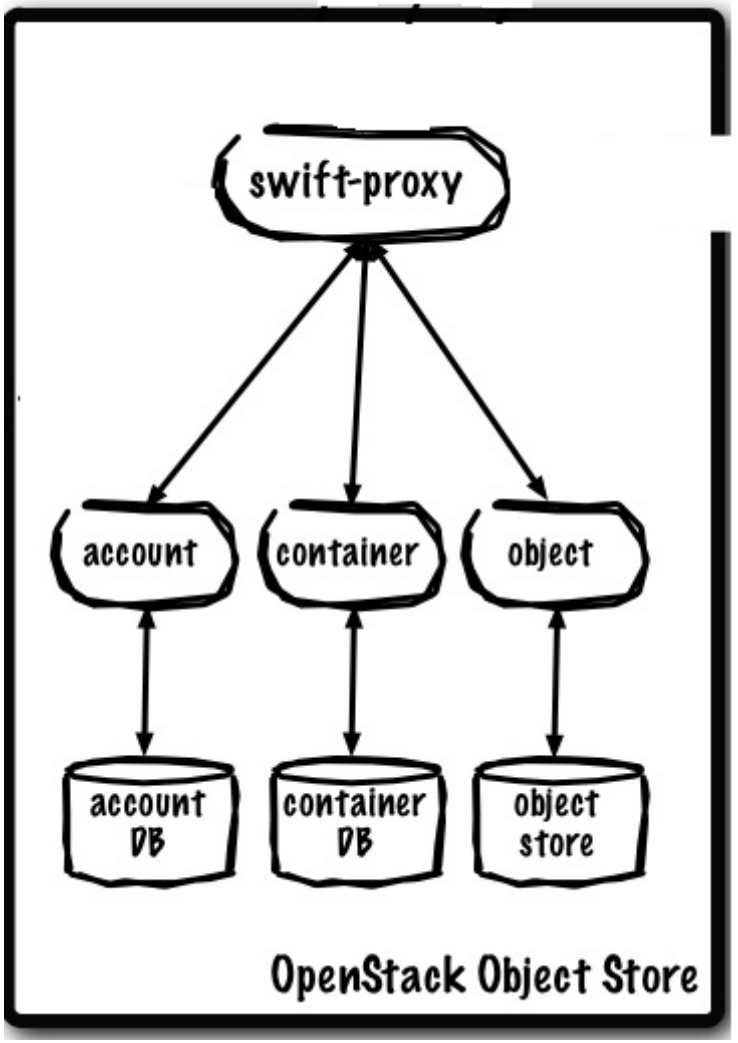


Swift - Storage

- Swift permette la **memorizzazione** e il recupero di file.
- Fornisce una piattaforma di **storage completamente distribuita**, accessibile tramite API, che può essere integrata direttamente all'interno delle applicazioni o utilizzata per l'archiviazione e il backup di dati
- **Non è** un filesystem tradizionale, ma **un sistema di storage distribuito per dati statici** come immagini VM, foto, email, backup e archivi
- Non esistendo un punto di controllo centralizzato, vengono garantite proprietà come **scalabilità**, **ridondanza** e **durabilità**

Swift - Componenti

- **Proxy Server:** accetta richieste come file da caricare, modifiche ai metadati o creazione di container
- **Accounts server:** gestisce gli account definiti attraverso il servizio di object storage
- **Container server:** gestisce il **mapping dei container** all'interno del servizio di object storage
- **Object server:** gestisce i file memorizzati sui vari nodi di storage

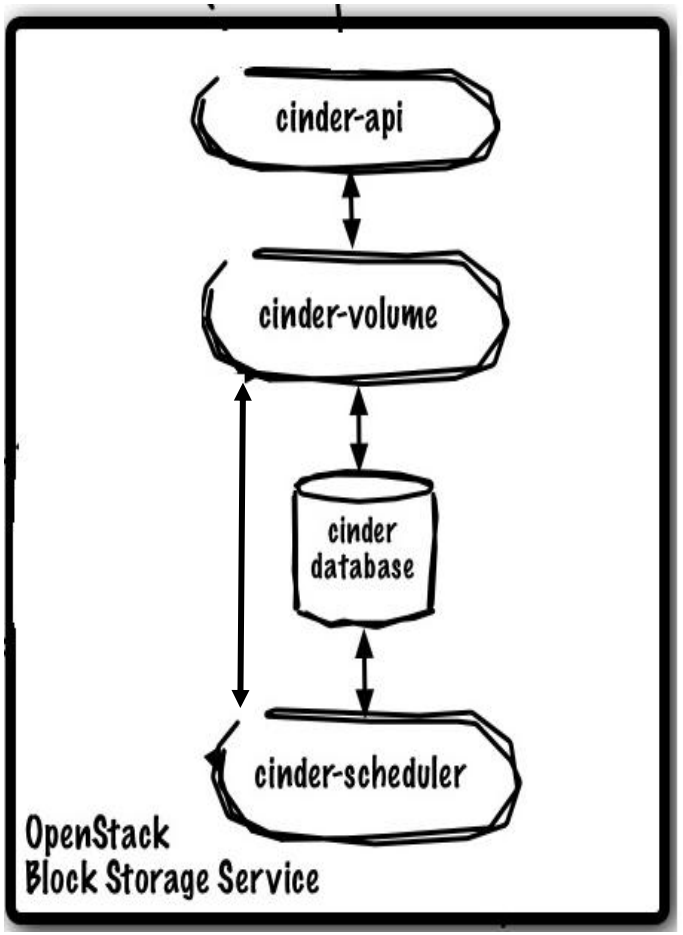


Cinder – Block Storage

- È stato recentemente lanciato come nuovo servizio di **block storage** (OpenStack è un **progetto in crescita**!) e fornisce dispositivi di storage che possono essere connessi alle istanze di VM
- Gestisce la **creazione** e la **dis-/connessione** dei volumi dalle istanze con supporto a diversi protocolli come iSCSI, NFS, FC, RBD, GlusterFS
- Supporta inoltre numerose **piattaforme di storage** come Ceph, NetApp, Nexenta, SolidFire e Zadara
- Possibilità di **eseguire snapshot** per il backup dei dati memorizzati nei volumi che possono poi essere ripristinati o utilizzati per creare un nuovo volume

Cinder - Componenti

- **cinder-api : accetta richieste** e le reindirizza al cinder-volume per il processamento
- **cinder-volume: è il sevizio vero e proprio** che risponde alle richieste andando a leggere e a scrivere sul cinder database per mantenere consistente lo stato del sistema, interagisce con gli altri componenti OpenStack attraverso una coda di messaggi e direttamente con i dispositivi di storage
- **cinder-scheduler: seleziona il dispositivo di storage** su cui creare il volume
- **cinder database:** memorizza lo stato dei volumi

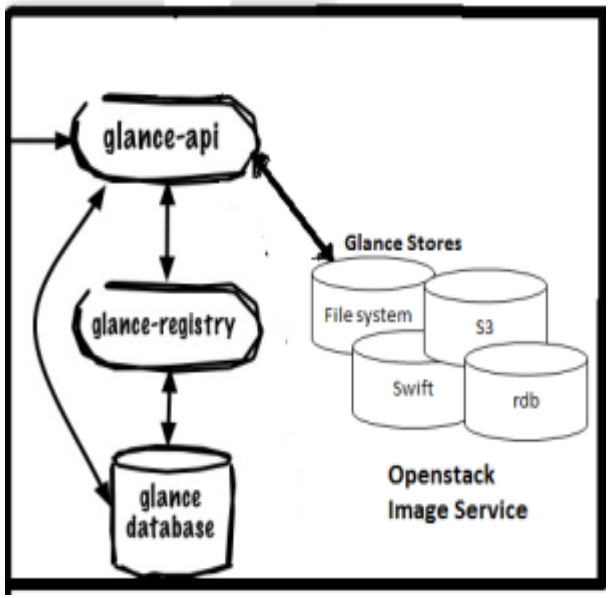


Glance – Image Service

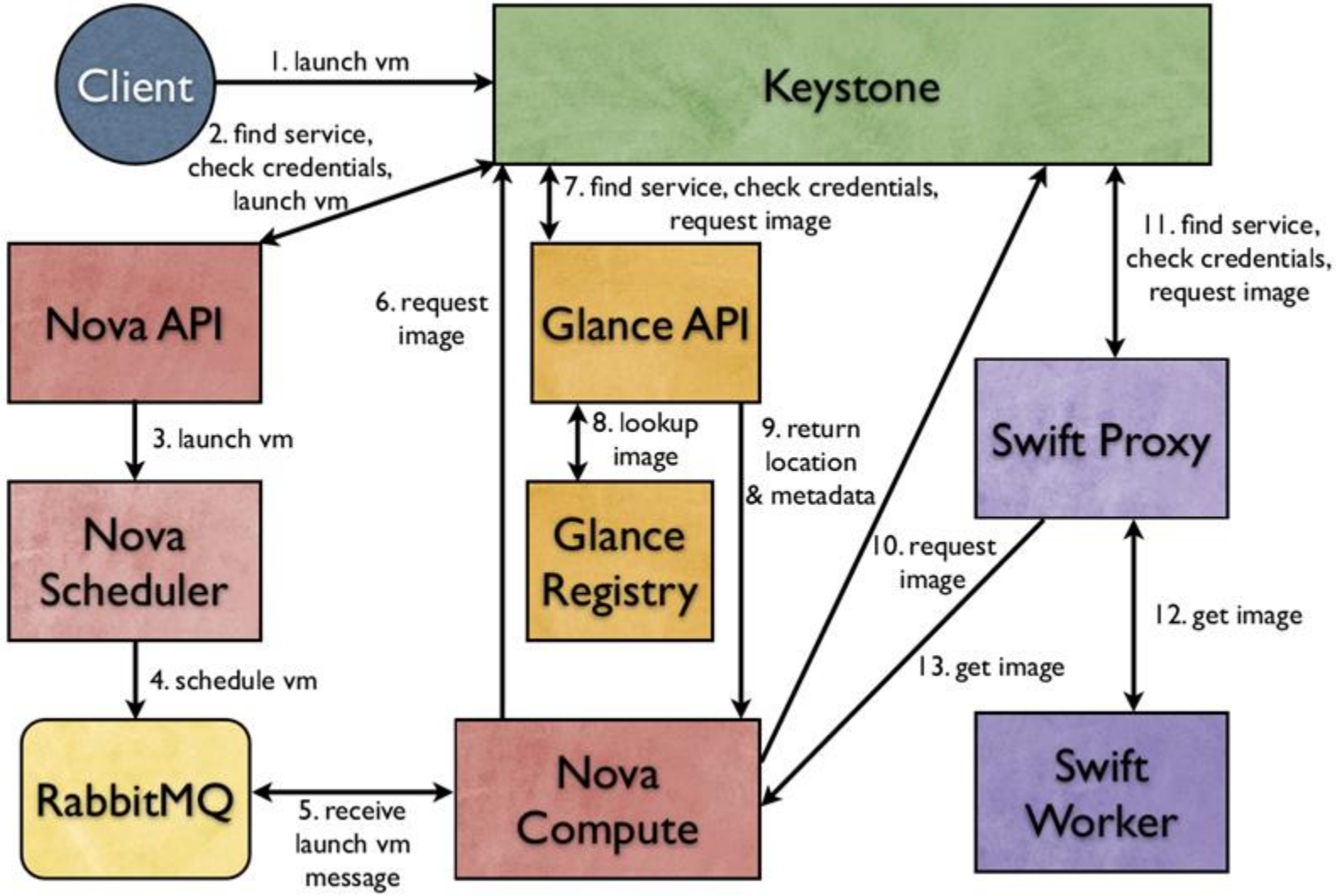
- Glance gestisce il **discovery**, la **registrazione** e la **consegna** di **immagini** di dischi e server virtuali
- Supporto per la memorizzazione di immagini su diversi sistemi di storage, tra cui Swift
- Supporta **molteplici formati di immagini** (es. Raw, qcow2, VMDK, ecc.)

Glance – Componenti

- **glance-api: gestisce le richieste** per il discovery di immagini, il recupero di immagini e la memorizzazione di immagini
- **glance-registry: gestione metadati** relativi alle immagini (dimensione, formato, ecc.)
- **glance database:** database per la **memorizzazione dei metadati** relativi alle immagini
- Glance si appoggia su di un **repository esterno** per la memorizzazione delle immagini. Per la memorizzazione vengono supportate diverse modalità come i normali **filesystem**, **Swift**, Amazon S3 e HTTP



Servizi OpenStack al lavoro: una visione d'insieme per l'avvio di una VM



Gestione della rete

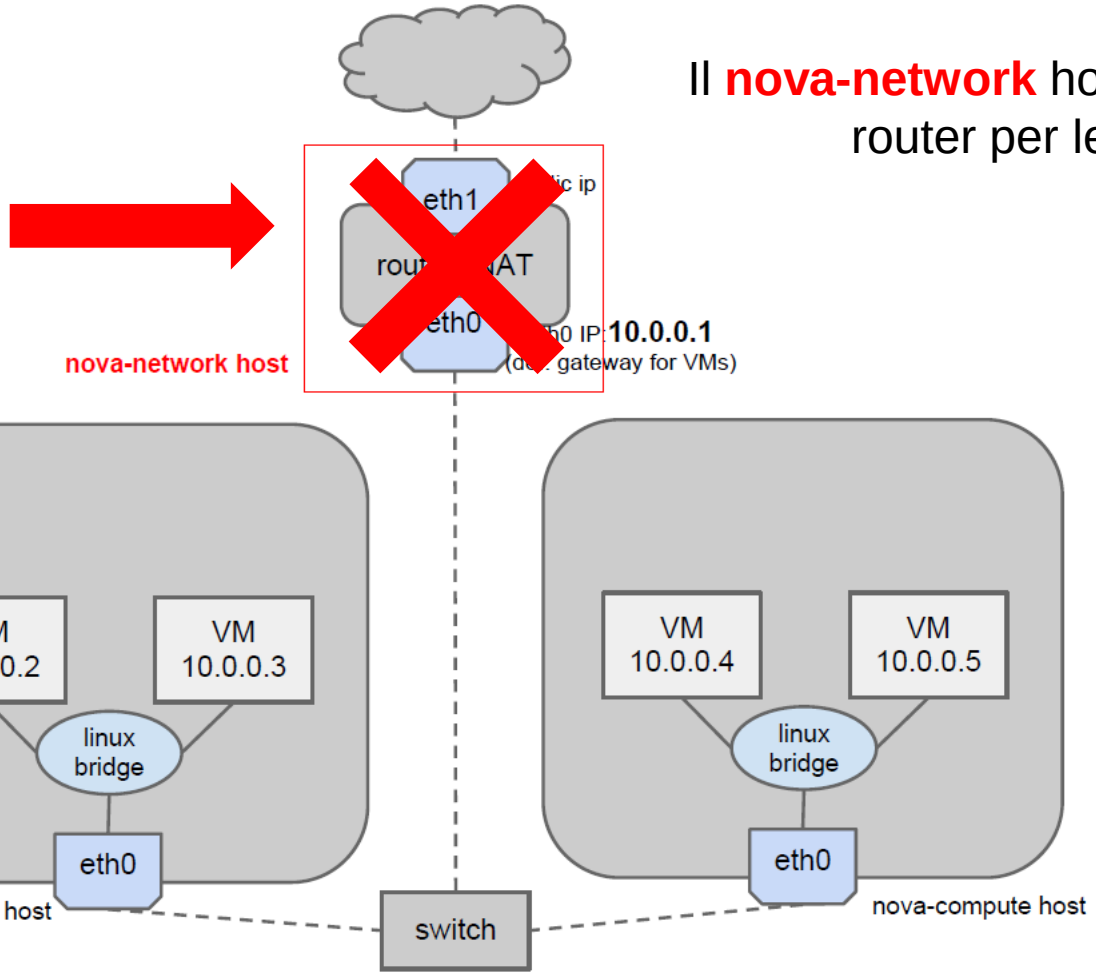
- Nelle **prime versioni di OpenStack** (e tuttora disponibile) basata sul servizio **nova-network** di Nova attraverso:
  - Linux bridge
  - Interfacce virtuali connesse alla rete attraverso un'interfaccia fisica
  - Assegnamento degli indirizzi IP alle VM
  - Fixed IP: restituito allo shutdown della VM
  - Floating IP: può essere riassegnato online
- Network Manager fornisce le **reti virtuali** per permettere alle VM di interagire tra loro e con la rete pubblica

Network manager

- Determina il **layout di rete** dell'infrastruttura Cloud
- **Ciascuna rete** a cui è collegata una VM viene **gestita da un host di network** (un host su cui è in esecuzione il demone nova-network, eventulamente anche replicato su più host)
- **Flat Network Manager:**
  - Un bridge Linux forma una sottorete
  - **Tutte le istanze** vengono **collegate allo stesso bridge**
  - Occorre **configurare manualmente** il server, il controller e gli indirizzi IP
- **Flat DHCP Network Manager:**
  - Viene aggiunto **un server DHCP** sullo **stesso bridge**
- **VLAN Network Manager:**
  - Viene creata una **VLAN e un bridge per ciascun progetto**

Single-host networking

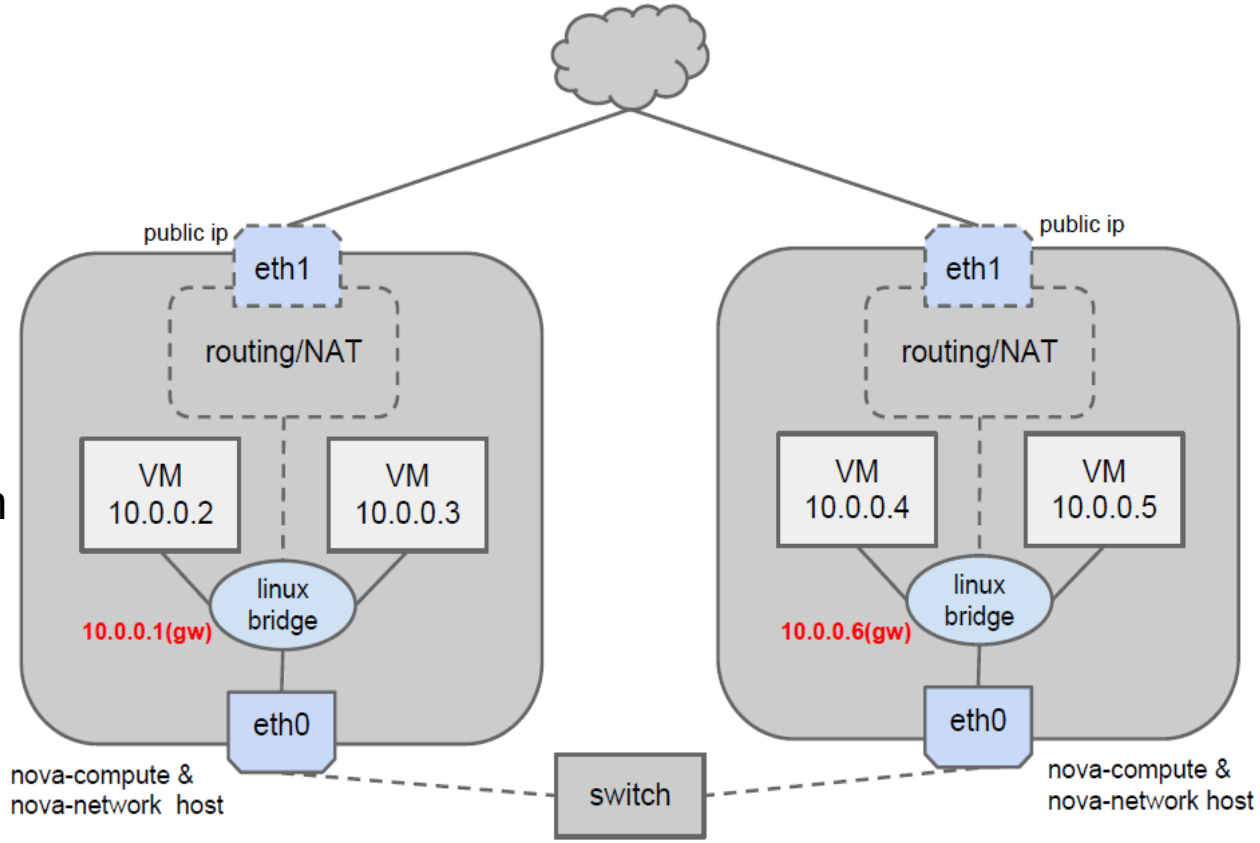
Single point of failure
se il nova-network host cade, le VM perdono la connettività con la rete pubblica!



Il **nova-network** host agisce da router per le VM

Multi-host networking (servizio replicato)

- Ciascun nodo è **indipendente dagli altri**, e in caso di caduta di altri mantiene accesso a Internet
- Su ciascun nodo sono in esecuzione i componenti **nova-compute** e **nova-network**

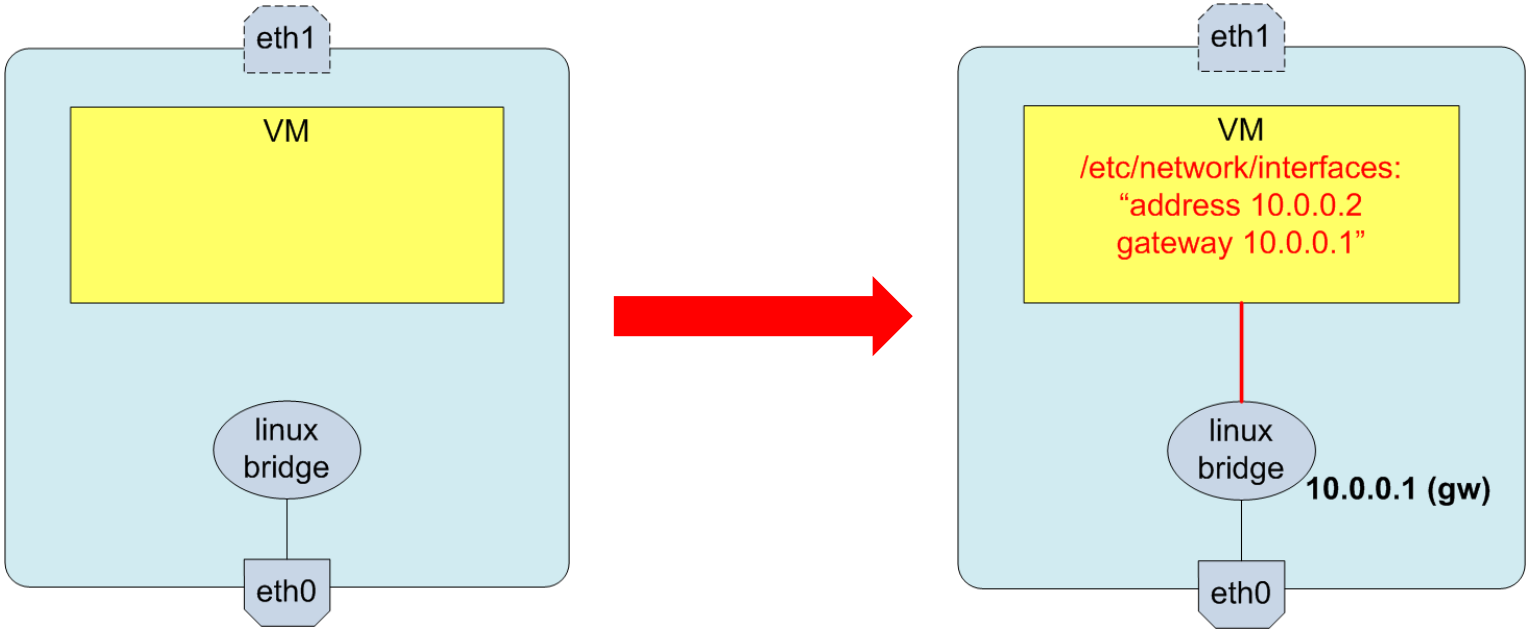


Multi-host networking

- **Traffico indipendente: le VM su tutti i nodi compute accedono alle reti esterne in modo indipendente**: per ciascuno di questi nodi, il bridge Linux locale viene utilizzato come default gateway
- **Routing:** vengono **controllate le tabelle di routing del kernel** per stabilire se debba essere eseguito il NAT del pacchetto su eth1 o se il pacchetto debba essere inviato su eth0
- Gestione degli **indirizzi IP:** nova-network **memorizza gli IP assegnati ai bridge e alle istanze nel database nova** (non solo DHCP…)

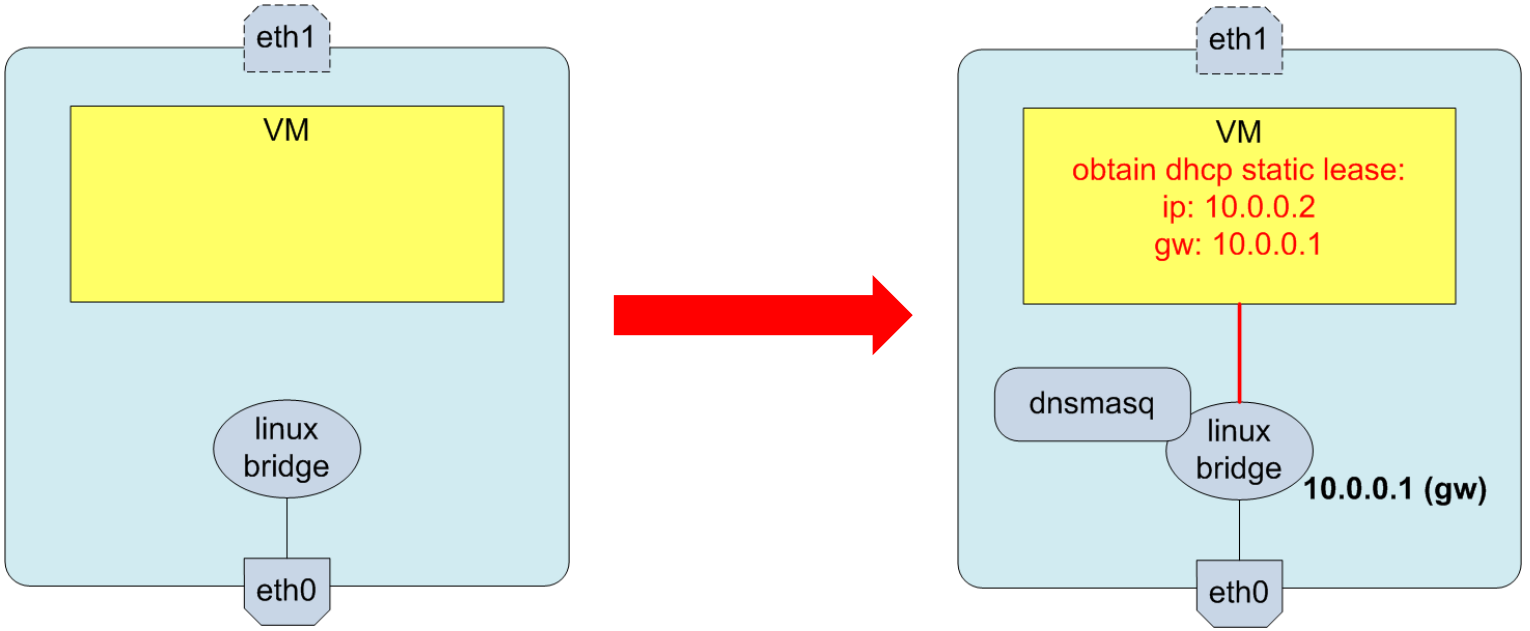
Flat Manager

- Opera su un **pool di indirizzi IP**: parti di questo pool sono condivise dai vari tenant
- **IP vengono allocati staticamente alle istanze (nel database nova)** al momento della loro creazione
- Collega le istanze su un bridge predefinito
- Configura le VM andando a modificare il file /etc/network/interfaces



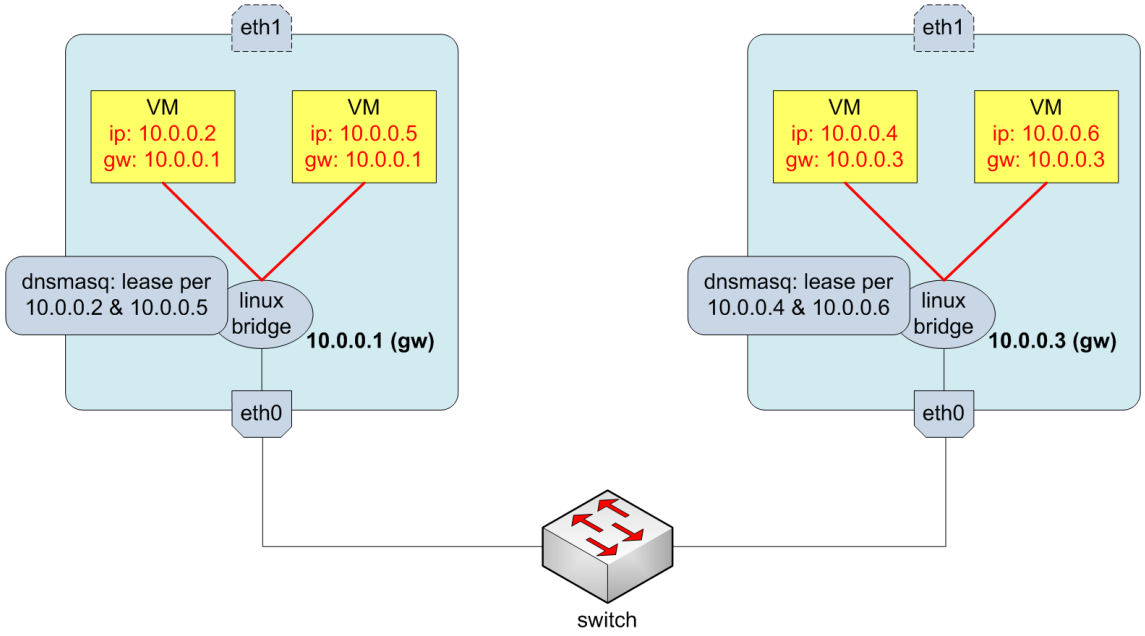
Flat DHCP Manager

- Opera su un **pool di indirizzi IP**: parti di questo pool sono condivisi dai vari tenant.
- **IP vengono allocati alle istanze** al momento della loro creazione **via DHCP** (poi salvate nel database nova)
- Collega le istanze su un bridge predefinito
- Mette in esecuzione un server DHCP (dnsmasq) attraverso il quale le varie istanze recuperano la configurazione di rete



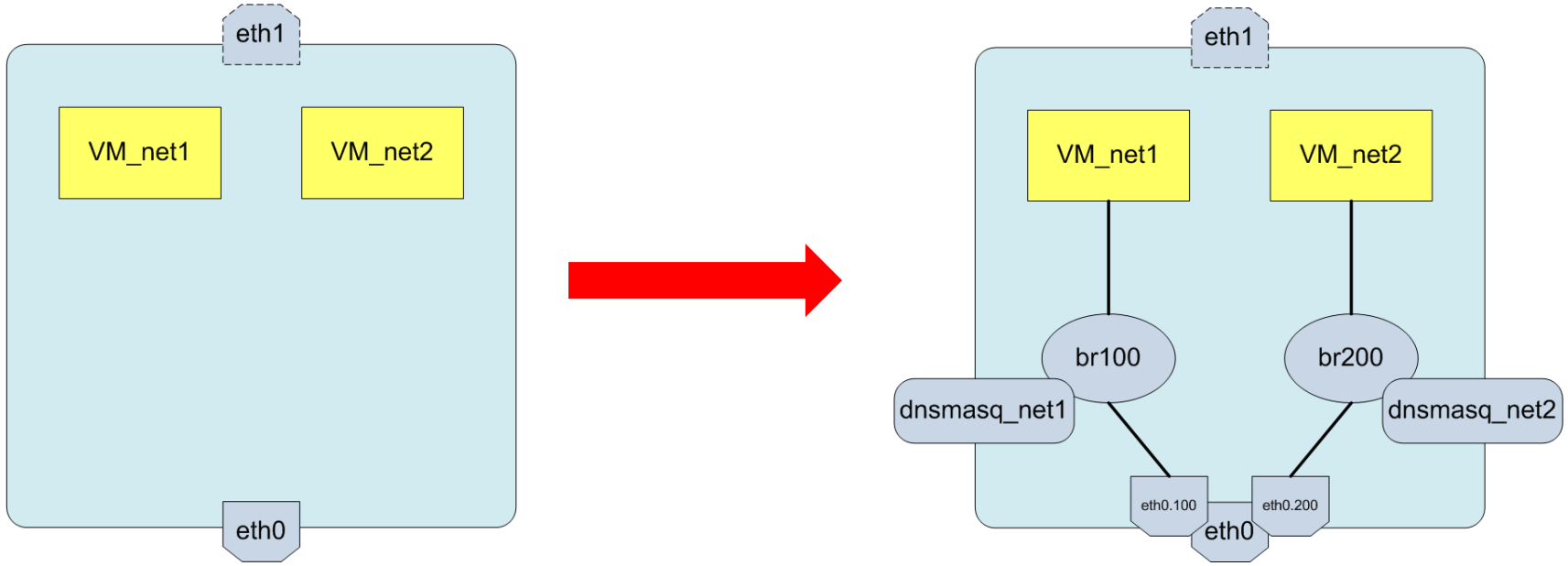
Flat DHCP – DHCP Server

- Gestito da nova-network
- Tiene traccia dei lease e dei release degli indirizzi IP
- Riusa e assegna indirizzi IP dinamicamente
- Nella **modalità multi-host viene eseguito su ciascun nodo compute** e **fornisce indirizzi solo alle istanze in esecuzione su quel nodo**



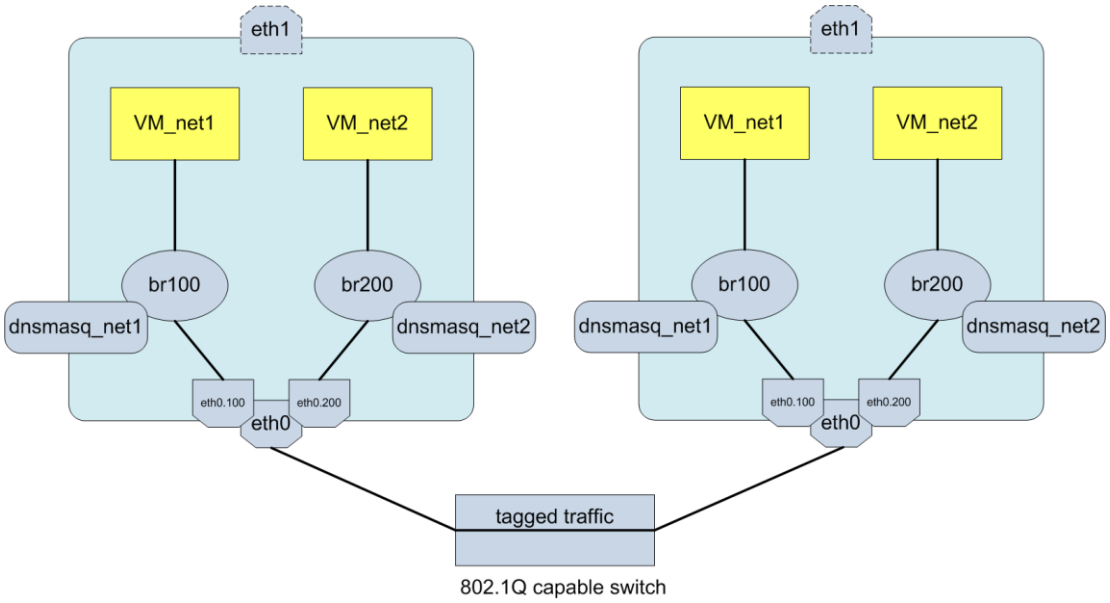
VLAN Manager

- Gestisce **più sottoreti IP**
- Utilizza un bridge dedicato per ciascuna rete
- **Mappa le reti sui tenant**
- Mette in esecuzione un server DHCP (dnsmasq) attraverso il quale le varie istanze recuperano la configurazione di rete
- Separa il traffico di rete con VLAN 802.1Q



VLAN Manager – Requisiti switch

Lo **switch** richiede il **supporto allo standard 802.1Q** per poter connettere istanze situate su nodi compute diversi



Problemi aperti:

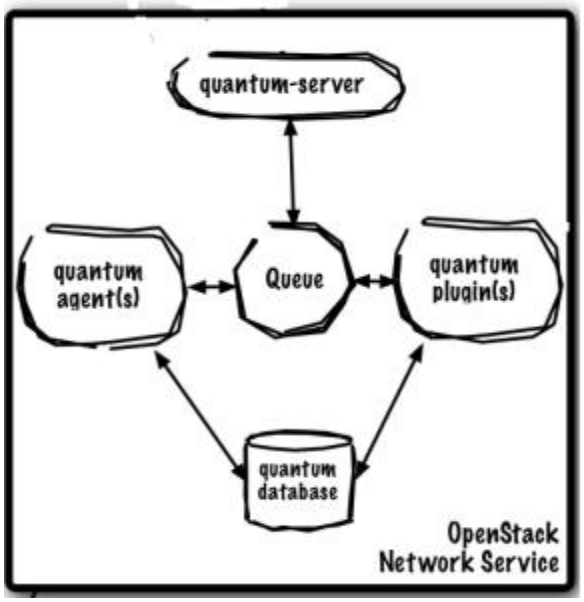
- **isolamento** logico/fisico
- facilità di **modifiche dinamiche**

Quantum Networking

- Sistema per la **gestione di reti e indirizzi IP** di tipo pluggable, scalable e API-driven
- “Network as a Service”
- Gli utenti possono **creare le proprie reti** e **connettervi le interfacce di rete virtuali**
- **Multitenancy: isolamento**, astrazione e pieno controllo su reti virtuali
- **Technology-agnostic:** le API specificano il servizio, il produttore fornisce la propria implementazione, estensioni per features vendor-specific
- **Loose coupling:** servizio standalone, non esclusivo di OpenStack

Quantum – Componenti

- **quantum-server: accetta richieste** tramite API e le indirizza al plugin corretto
- **Plugins e Agents: eseguono le reali azioni**, come collegare/scollegare porte, creare reti e sottoreti, creare router, ecc.
- **queue: MOM che smista i messaggi** tra quantum-server e vari agents
- **quantum database:** memorizza lo stato della rete per particolari plugin



Un po' di pratica

- Ambiente di test:
  - un nodo **controller** su cui saranno in esecuzione tutti i servizi principali di OpenStack
  - un nodo **compute** sul quale verranno messe in esecuzione istanze di macchine virtuali
  - **nova-network** con configurazione multi-host (sia su controller sia su compute) e Flat DHCP server
  - entrambi i nodi hanno a disposizione una sola interfaccia di rete eth0

Installazione di OpenStack

- Il metodo più semplice per installare OpenStack è utilizzare **DevStack**
- **DevStack** è uno script che permette di creare velocemente un ambiente di sviluppo OpenStack
- **DevStack** si occupa di:
  - scaricare tramite git i componenti di OpenStack
  - scaricare le varie dipendenze (es. mysql, RabbitMQ, ecc.)
  - configurare i componenti di OpenStack
- È possibile configurare **DevStack** andando a modificare il file **localrc**

Installazione di OpenStack

- Scarichiamo DevStack

```
% git clone git://github.com/openstack-dev/devstack.git
% cd devstack
```

- Prima di procedere al lancio dello script, bisogna opportunamente **creare** e **configurare il file localrc** (è possibile trovare un esempio di questo file nella cartella samples)

Nodo Controller – localrc

```
## Controller Host ##
HOST_IP=192.168.1.10
MULTI_HOST=1
## Network nova-network ##
FLAT_INTERFACE=eth0
FIXED_RANGE=10.0.0.0/24
FIXED_NETWORK_SIZE=254
FLOATING_RANGE=192.168.233.128/25
## Leaving Default Services Enabled ##
DISABLED_SERVICES=quantum
## Logs ##
LOGFILE=/opt/stack/logs/stack.sh.log
VERBOSE=True
LOG_COLOR=False
SCREEN_LOGDIR=/opt/stack/logs
## Auth ##
ADMIN_PASSWORD=password
MYSQL_PASSWORD=password
RABBIT_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=token
```

Il **floating range** rappresenta la rete pubblica, mentre il **fixed range** rappresenta la rete privata utilizzata dalle VM per comunicare tra loro.

Nodo Compute – localrc

```
## Compute Host ##
SERVICE_HOST=192.168.1.10
HOST_IP=192.168.1.11
MULTI_HOST=1
## Network nova-network ##
FLAT_INTERFACE=eth0
FIXED_RANGE=10.0.0.1/24
FIXED_NETWORK_SIZE=254
FLOATING_RANGE=192.168.1.128/25
## Compute Node Services ##
ENABLED_SERVICES=n-cpu,n-net,n-
api,n-vol
## API URIs ##
Q_HOST=$SERVICE_HOST
MYSQL_HOST=$SERVICE_HOST
RABBIT_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
KEYSTONE_AUTH_HOST=$SERVICE_HOST
KEYSTONE_SERVICE_HOST=$SERVICE_HOST
## Auth ##
ADMIN_PASSWORD=password
MYSQL_PASSWORD=password
RABBIT_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=token
DATABASE_TYPE=mysql
## Logs ##
LOGFILE=/opt/stack/logs/stack.sh.log
VERBOSE=True
LOG_COLOR=False
SCREEN_LOGDIR=/opt/stack/logs
```

Su questo nodo (compute) vanno messi in esecuzione **solo alcuni servizi di nova** (es. compute, network, ecc.).
Gli altri servizi devono, infatti, essere in esecuzione solo sul nodo controller

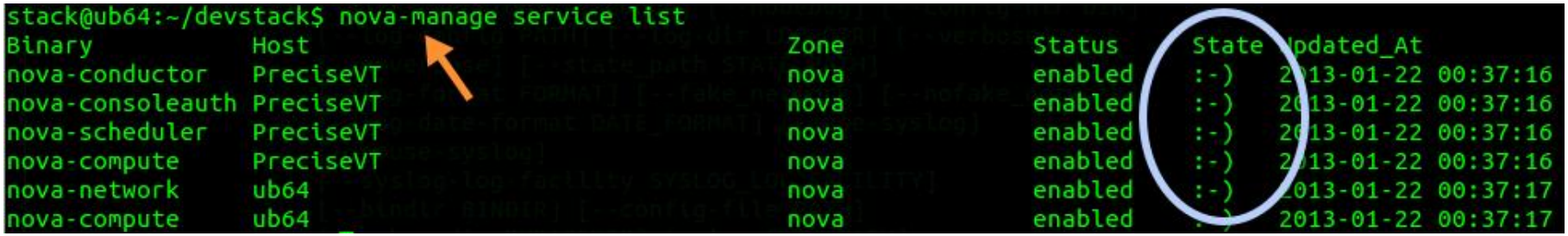
Installazione di OpenStack

- Possiamo ora eseguire lo script

```
% ./stack.sh
```

- Verifichiamo che l'installazione sia andata a buon fine e che i servizi siano attivi

```
% nova-manage service list
```



Caricamento di un'immagine

- Scarichiamo un'immagine

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

- Importiamo l'immagine tramite Glance

```
% glance add name=CirrOS-0.3.0 is_public=true
  container_format=ovf disk_format=qcow2 < cirros-0.3.0-x86_64-
  disk.img
```

- Verifichiamo che l'immagine sia stata correttamente importata

```
% glance index
```

Avvio di una VM

- Generiamo una chiave ssh per l'accesso remoto alla macchina

```
% nova keypair-add mykey > ~/mykey.pem
% chmod 0600 ~/mykey.pem
```

- Abilitiamo il ping e ssh verso le VM

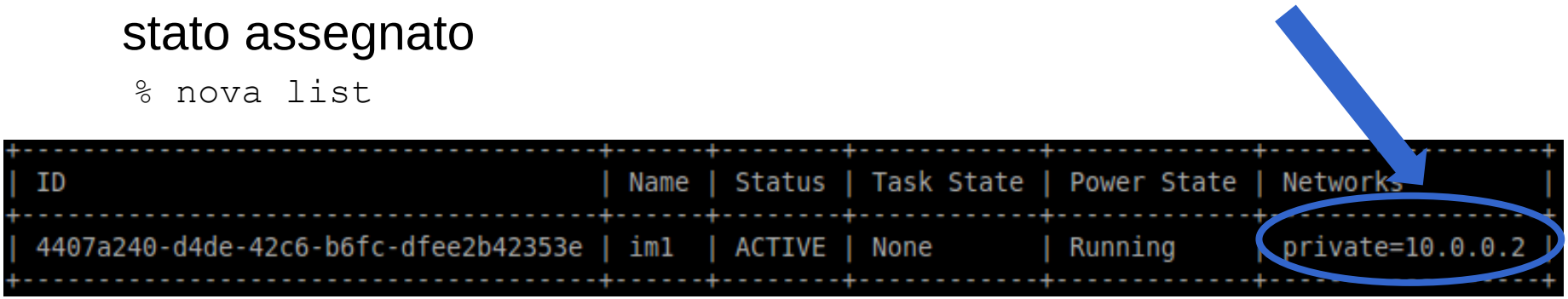
```
% nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
% nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

- Eseguiamo il boot di una VM

```
% nova boot --image CirrOS-0.3.0 --flavor 1 --key_name mykey
   im1
```

- Verifichiamo lo stato della VM e controlliamo l'IP che le è stato assegnato

```
% nova list
```



ID	Name	Status	Task State	Power State	Networks
4407a240-d4de-42c6-b6fc-dfee2b42353e	im1	ACTIVE	None	Running	private=10.0.0.2

Avvio di una VM

- Aspettiamo alcuni secondi e proviamo ad eseguire il ping verso la VM

```
% ping 10.0.0.2
```

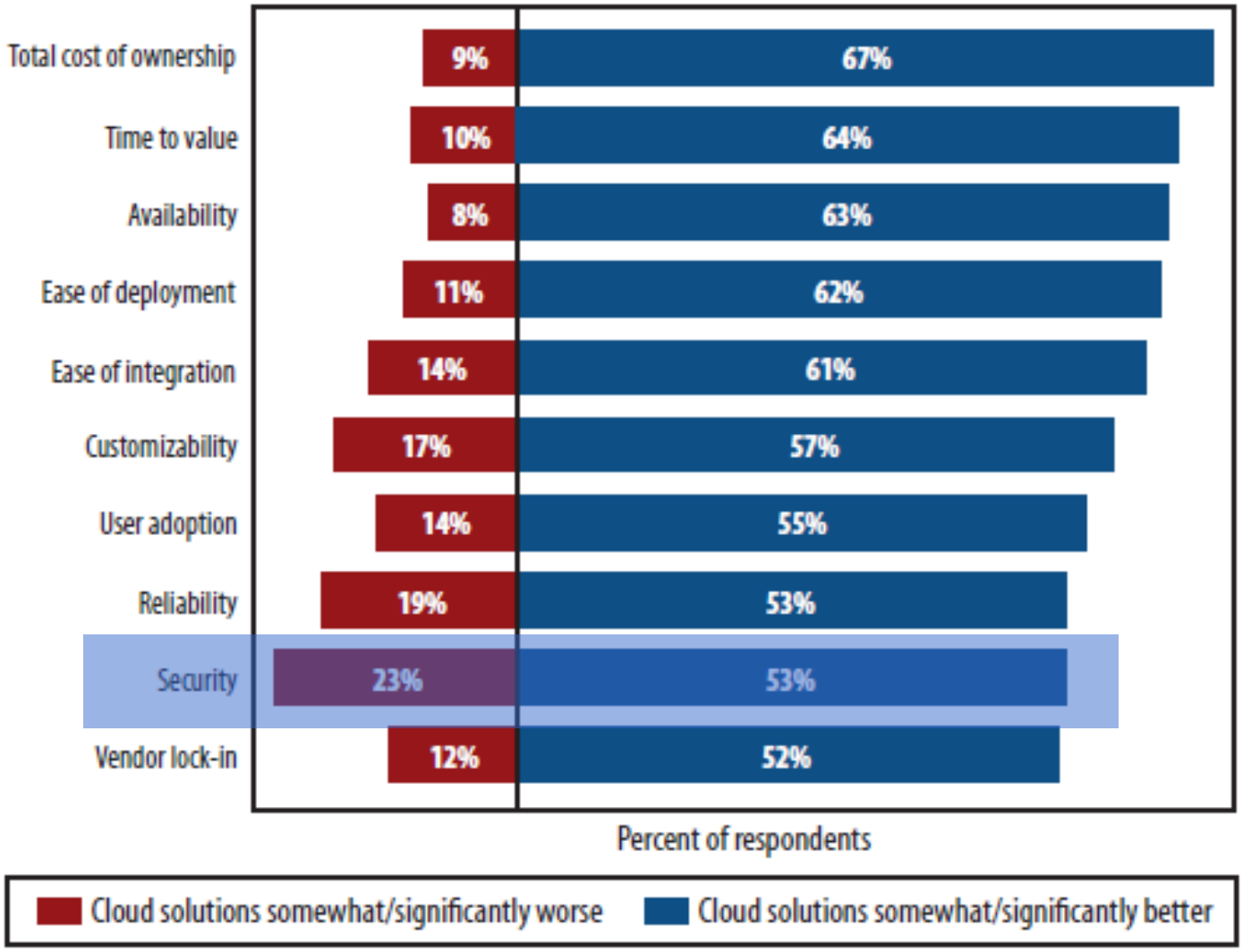
- Eseguiamo il login tramite ssh sulla VM

```
% ssh -i mykey.pem cirros@10.0.0.2
```

- Se ci sono problemi nell'avvio della VM, è possibile visualizzare il log della console

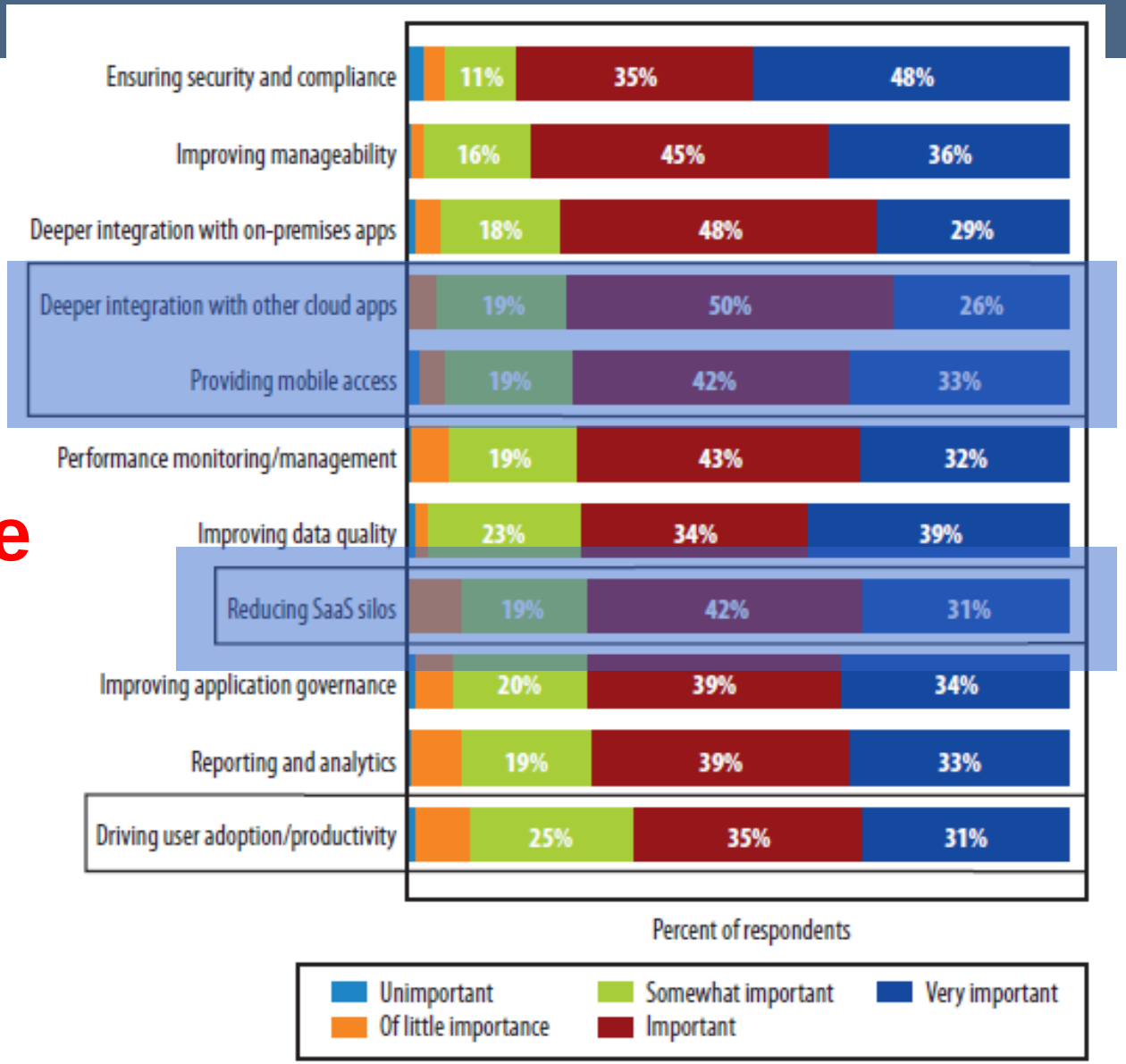
```
% sudo tail -F
  /opt/stack/data/nova/instances/<id-istanza>/console.log
```

Cloud computing: reality check



Cloud emerging challenges

Integrazione



Conclusioni e problemi aperti

Conclusioni

- Cloud computing è un nuovo paradigma computazionale col quale **ci stiamo già confrontando**
- Cloud computing è un nuovo **modello architetturale distribuito** emergente
  - **Non** esiste **definizione comune**
  - **MA** è **ben caratterizzato**, nelle sue varie declinazioni, **dal punto di vista tecnologico** (stack e componenti principali)

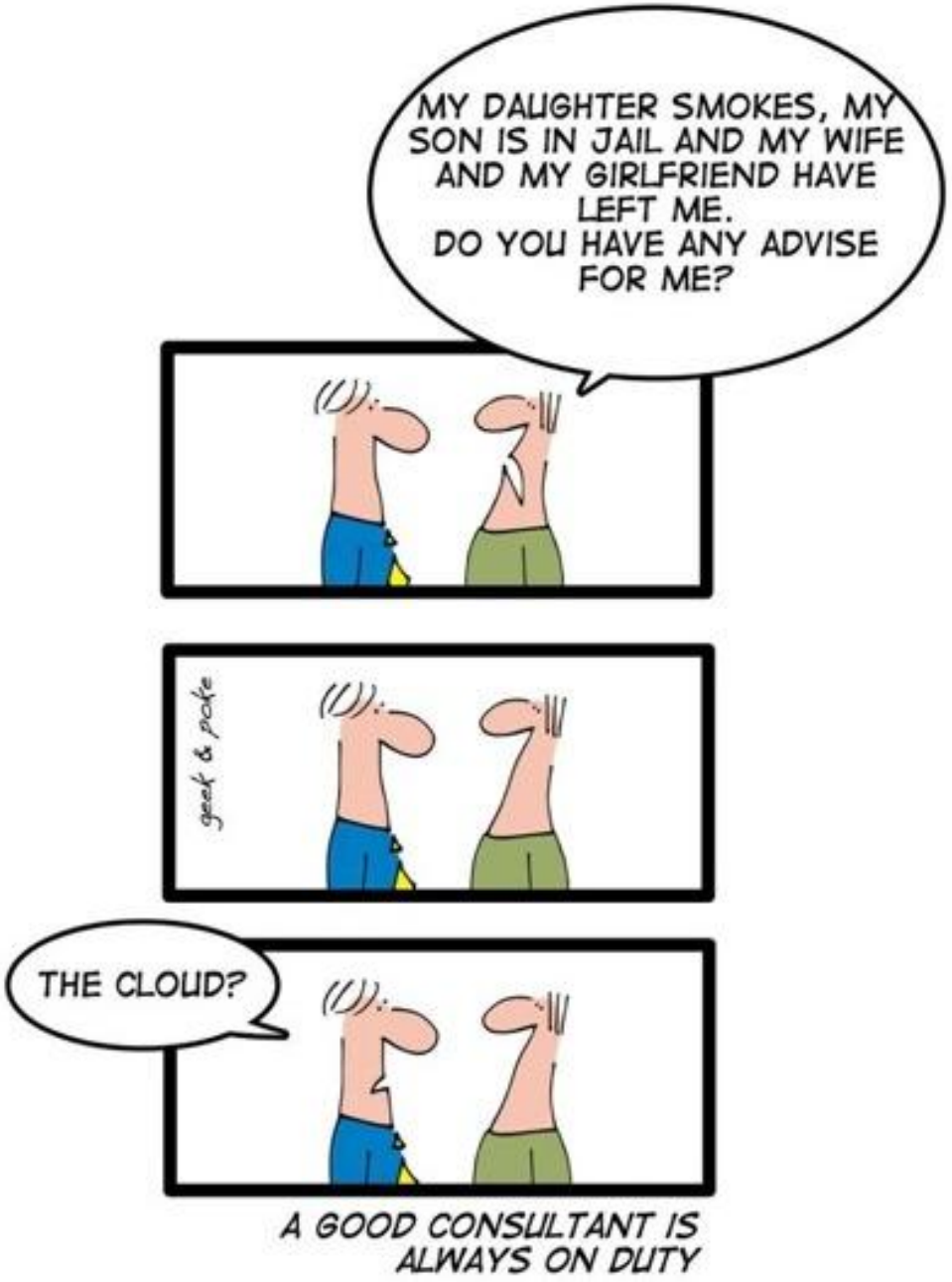
Problemi aperti

- Cloud computing e **modello economico**

  "Conviene affittare risorse di un Cloud esterno o acquistare e fornire un proprio Cloud?" → dipende, molta letteratura su questo aspetto
- Sicurezza e esternalizzazione gestione dati
  → ***necessità di nuovi modelli di trust***
- Integrazione fra Cloud e fra servizi SaaS

Cloud per tutto

Non esattamente per tutto ☺



The fog has gone…

… and Clouds are letting out the light!

Grazie per la vostra attenzione!

Riferimenti bibliografici

- Creeger M.
  Cloud Computing: An Overview
  ACM Queue, vol. 7, no. 5, pp. 3-4 (2009)
- Lenk, A., Klems, M., Nimis, J. et al.
  What's inside the Cloud? An architectural map of the Cloud landscape
  In ICSE Workshop on Software Engineering Challenges of Cloud Computing (2009)
- Vogels, W.
  Eventually consistent
  Communications of the ACM, vol. 52, no. 1, pp. 40-44 (2009)
- Vogels, W.
  Ahead in the Cloud - The Power of Infrastructure as a Service
  http://blip.tv/file/981534 (2010)
- Narasimhan, B.; Nichols, R.;
  State of Cloud Applications and Platforms: The Cloud Adopters' View
  IEEE Computer, vol. 44, no. 3, pp. 24-28 (Marzo 2011)

Riferimenti bibliografici

- ***OpenStack documentation***
  http://docs.openstack.org/
- ***OpenStack Wiki***
  https://wiki.openstack.org/wiki/Main_Page
- ***DevStack***
  http://devstack.org/
- HOU J. T. C.; Liao W.
  OpenStack Introduction
  http://www.slideshare.net/jasonhoutw/openstack-introduction
  (Giugno 2012)