# Introduzione ad OpenStack: i servizi e le funzionalità

Scuola di Cloud Computing

# Agenda

- Progetto OpenStack
- Componenti principali
- Risultati ultima user survey

# OpenStack

# Cosa è OpenStack

- È un insieme di software con lo scopo di fornire infrastrutture cloud pubbliche o private, largamente scalabili
  - servizi di cloud storage, compute, and networking
- Ha un disegno architetturale aperto e modulare, principalmente sviluppato in Python
- Interopera con altri sistemi di Cloud computing pubblici o privati
  - Ad es. VMware ESXi, Microsoft Hyper-V, Amazon EC2

# Principi

- Modello di sviluppo open source
  - dipendenze di tipo open source
  - può essere eseguito su piattaforme interamente open source (Linux)
- Processo di sviluppo aperto
  - design summit ogni 6 mesi, in cui gli sviluppatori ricevono requisiti e scrivono le specifiche per la release successiva
- Comunità aperta
  - decisioni prese con modello del tacito assenso
  - tutti i processi sono documentati e trasparenti

# Chi partecipa ad OpenStack

- Fondato da NASA e Rackspace nel 2010
- Collaborazione di sviluppatori e utenti di dimensioni mondiali
- Forte supporto da parte dell'industria
  - ad es. Rackspace, Intel, Cisco, Juniper, NetApp, HP, DELL, VMware, AT&T, IBM, Canonical, SUSE, RedHat, Yahoo!
- Governance interna ben definita che non è in mano a nessun singolo ente o impresa
- In forte e costante crescita in termini di funzionalità e di sviluppatori (cf. http://goo.gl/lBHzn per una comparazione con OpenNebula, CloudStack, Eucalyptus)

# Sponsor principali


8 platinum
($500K/y)

19 gold
($50K÷200K/y)

# Qualche numero

- Alla fine del 2012 (cf. http://goo.gl/d6vG8):
  - 6.695 membri della comunità di OpenStack in 87 paesi (oggi sono 17081 in 139 paesi)
  - Più di 550 sviluppatori sui vari progetti che compongono OpenStack
  - Più di 300.000 download di OpenStack dai repository centrali
  - Sponsorizzato da 155 industrie/compagnie
  - Solo nel 2012 sono stati lanciati 48 user groups in 33 paesi
  - La partecipazione al Design Summit di OpenStack in autunno 2012 è aumentata di tre volte rispetto al Design Summit di primavera
  - La comunità di OpenStack nei social media è stimata essere circa sei volte quella del prodotto concorrente open source più prossimo
- All'OpenStack Summit di novembre 2013:
  - 3500 partecipanti
  - Casi d'uso mostrati per ditte con numero di utenti nell'ordine di 450 *milioni* di persone
- La città con il più alto numero di sviluppatori OpenStack nel mondo?
  - Pechino

# Un prodotto in ascesa

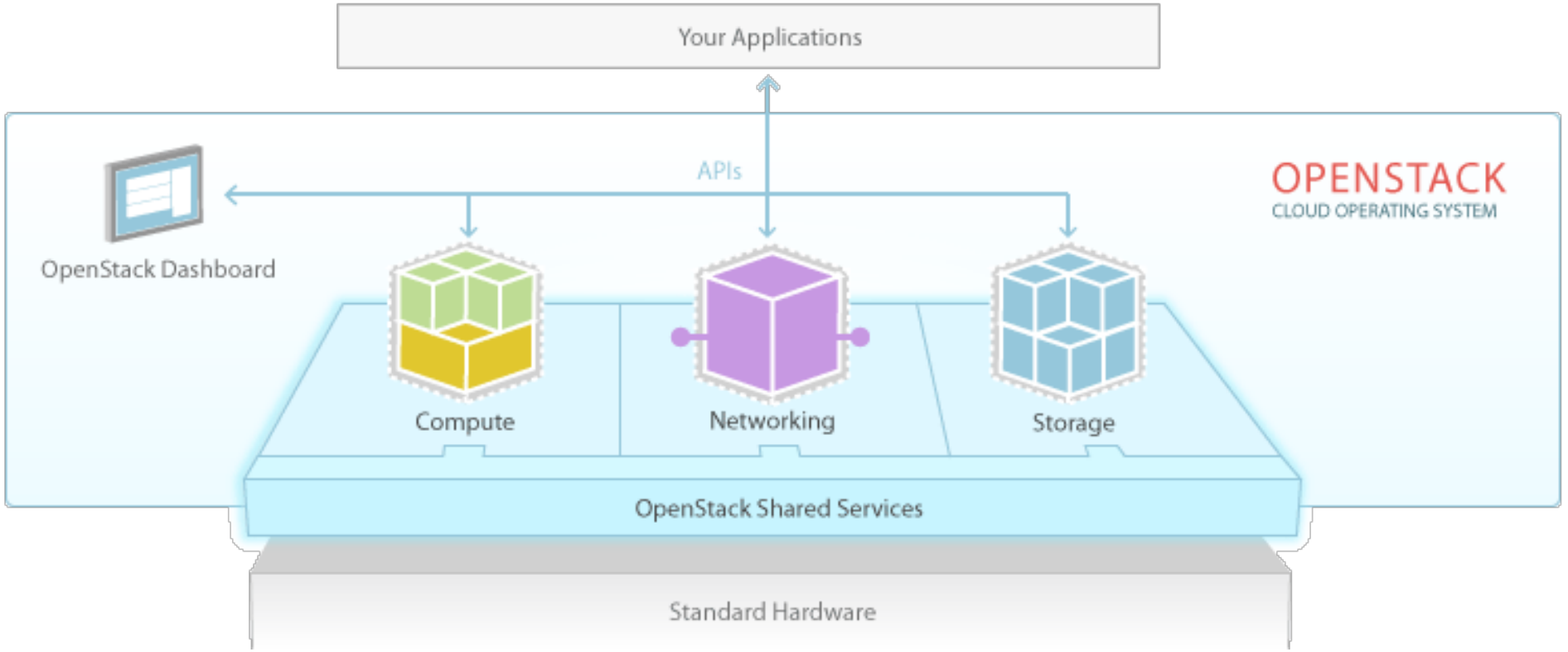

Interesse regionale

openstack | cloudstack | opennebula

Regione | Città

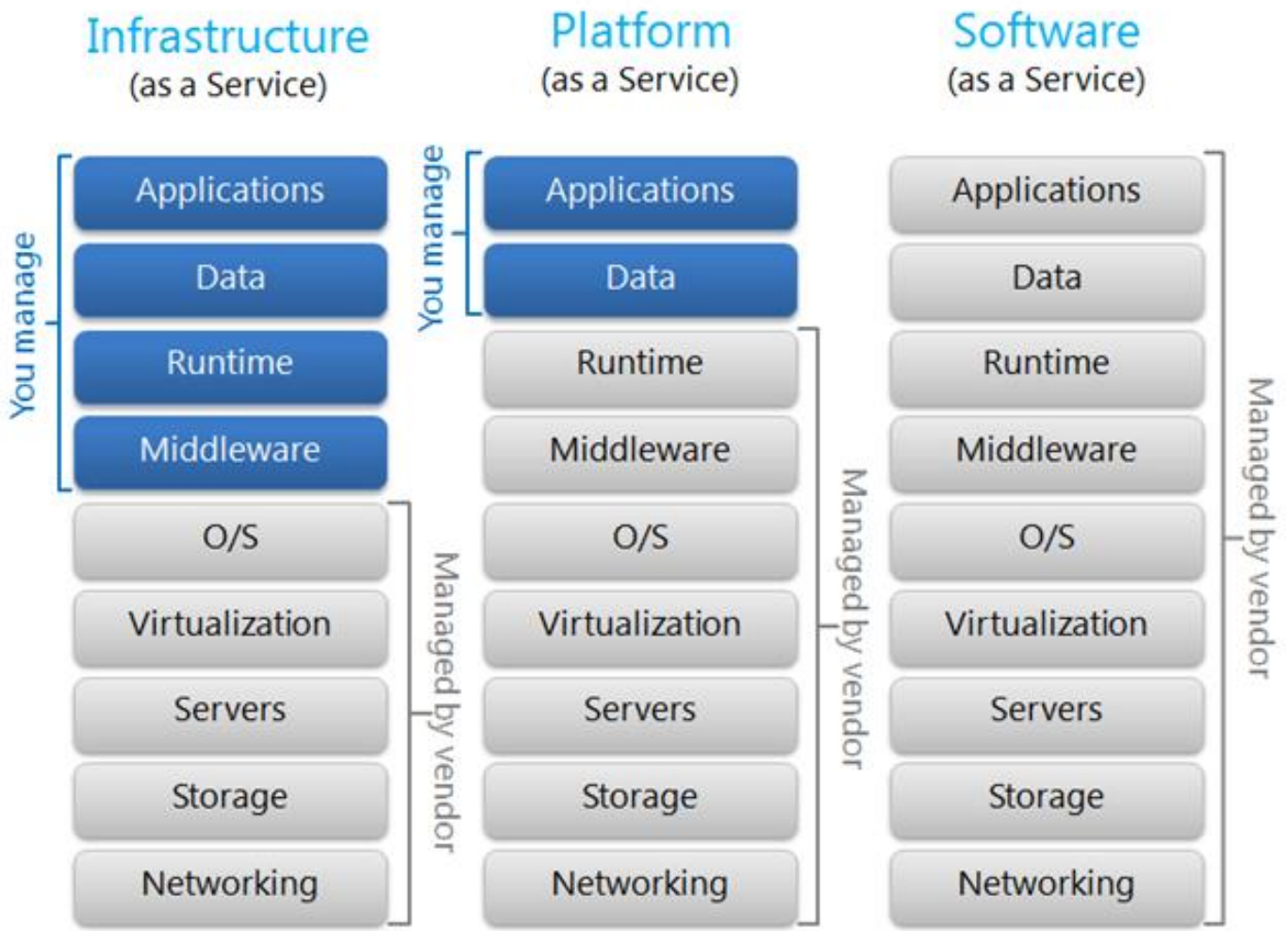
| | |
|---|---|
| Cina | 100 |
| Corea del Sud | 88 |
| Taiwan | 83 |
| Hong-Kong | 80 |
| India | 57 |
| Giappone | 44 |
| Stati Uniti | 42 |

# Visione d'insieme




Fonte: OpenStack

# OpenStack: piattaforma IaaS




Fonte: http://goo.gl/1jmkR

# I nomi delle release

- In OpenStack ogni *major release* del software ha un nome in codice
  - Il ciclo di rilascio delle major release è attualmente di 6 mesi (http://goo.gl/gMRhb)
  - La versione precedente all'attuale, rilasciata a ottobre 2013, è chiamata Havana. Questa è la ottava release di OpenStack (http://goo.gl/MlPbu)
  - La versione attuale, chiamata *Icehouse*, è uscita il 17 aprile 2014 (quella seguente uscirà ad ottobre 2014 e si chiamerà *Juno*)

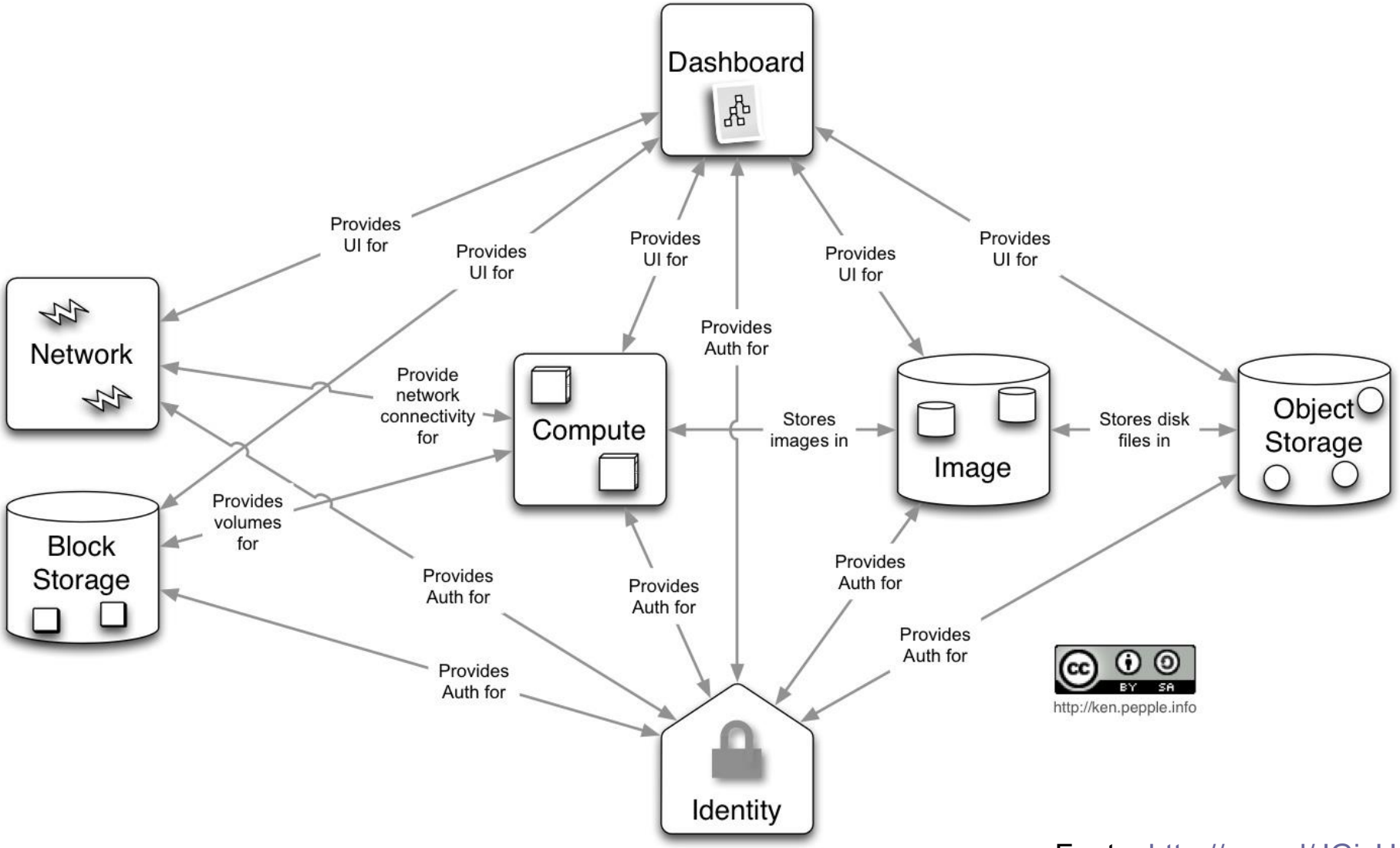
# I nomi dei componenti

- Ogni componente funzionale (modulo) di OpenStack ha un nome in codice. Questi sono i moduli principali:
  - Dashboard (web interface) → **Horizon**
  - Servizio di immagini (catalogo, repository) → **Glance**
  - Compute (server virtuali) → **Nova**
  - Network (gestione della rete) → **Neutron (Quantum)**
    - Estensibile con plugin, es. Load Balancer as a Service (LBaaS).
  - Identità (autenticazione, autorizzazione) → **Keystone**
  - Block storage (gestione dei volumi) → **Cinder**
  - Object storage (gestione di files) → **Swift**

# Progetti collegati

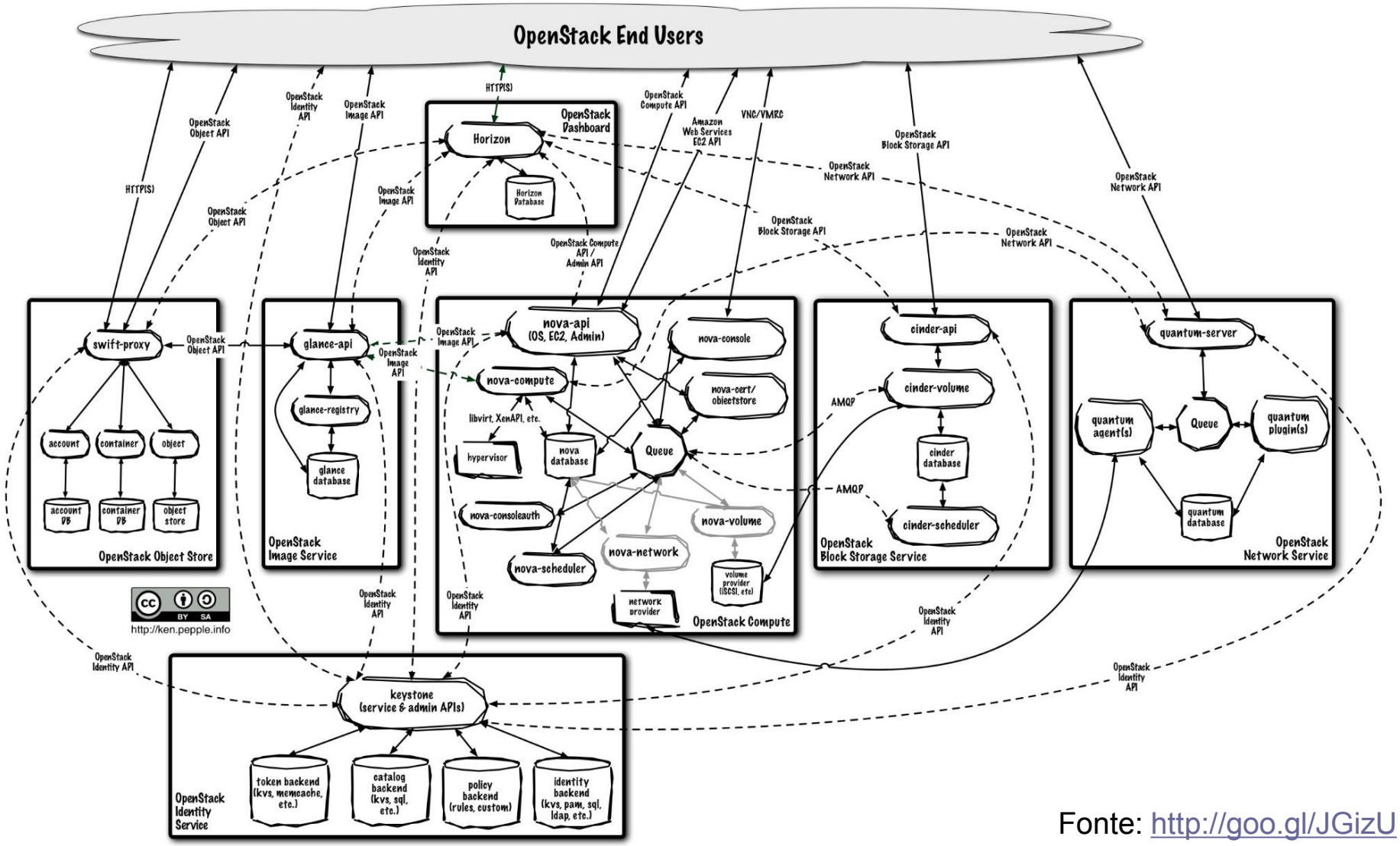
- Oltre ai moduli citati, esistono in OpenStack diversi altri “progetti”, alcuni già integrati in Havana, altri “incubati” per l’integrazione in release successive:
  - In Havana:
    - Metering (controllo d’uso delle risorse) → **Ceilometer**
    - Orchestration (descrizione e automazione deployment dell’infrastruttura) → **Heat**
  - Per release successive ad Havana:
    - Database service (in Icehouse, supporto di DBaaS per-tenant, sia relazionali che NoSQL) → **Trove**
    - Bare Metal (gestione di hardware non virtualizzato) → **Ironic**
    - Queue service (message queues as a service, simile ad Amazon SQS) → ~~**Marconi**~~ Zaqar
    - Data processing (fornitura di un cluster Hadoop su OpenStack) → **Sahara**

# L’architettura a moduli




Fonte: http://goo.gl/JGizU

# Il diavolo sta nei dettagli…




Fonte: http://goo.gl/JGizU

# Ma in sintesi…

- Gli utenti finali interagiscono con i servizi attraverso una interfaccia web comune, oppure attraverso API specifiche di ogni servizio.
- Tutti i servizi hanno una autenticazione comune.
- I singoli servizi interagiscono tra di loro attraverso le rispettive API pubbliche.

# Comparato ad Amazon Cloud

- Diversi dei servizi di OpenStack sono concettualmente simili a quelli forniti dal servizio Cloud di Amazon.

- **Nova** (la parte “compute”) corrisponde ad Amazon EC2 (“Elastic Compute Cloud”, http://goo.gl/2r8X).
  - Esistono API in OpenStack che consentono di interoperare direttamente con EC2 (http://goo.gl/lxh9T).
- **Swift** (object store) corrisponde ad Amazon S3 (“Simple Storage Service”, http://goo.gl/ailE).
- **Glance** (image service) corrisponde al catalogo AMI di Amazon.
- **Cinder** (volumi, o block storage) corrisponde ad Amazon EBS (“Elastic Block Storage”, http://goo.gl/Q7Fb).

# Horizon (dashboard)

- Horizon è il modulo che fornisce l'interfaccia utente via Web per amministratori e per utenti finali.
  - È scritto in Django (http://goo.gl/WWtun), un framework in Python per lo sviluppo di applicazioni web.
    - La personalizzazione è resa possibile dalla separazione tra la parte di presentazione e quella di interfacciamento con il resto di OpenStack
- Nota: non è indispensabile usare Horizon come dashboard.
  - Si può senza problemi sviluppare un proprio pannello di controllo (magari solo per gli utenti) che comunichi con le API di OpenStack.
  - Scelta adottata da alcuni fornitori di servizi Cloud.

# Utenti, tenant e ruoli

- **Utente**: una rappresentazione digitale di una persona, di un sistema o di un servizio che utilizza una parte di OpenStack.
- **Tenant**: un insieme che raggruppa risorse, oggetti, utenti.
  - Ad esempio: un'organizzazione, un progetto, una sede, un gruppo.
- **Ruolo**: i privilegi che vengono assegnati a un certo utente.
  - amministratore (globale, non del singolo tenant), o membro.
- Cf. http://goo.gl/LZCX4.

# Keystone (identità) 1/2

- Keystone implementa un unico componente di autenticazione per i diversi moduli di OpenStack. Fornisce autorizzazioni per credenziali di log-in multiple.
  - Identità: validazione delle credenziali per gli utenti.
  - Token: validazione e gestione dei token per l'autenticazione, una volta che le credenziali siano stati verificate.
  - Catalogo dei servizi, con informazioni sugli endpoint (indirizzi accessibili via rete).
  - Gestione delle policy di autorizzazione.

# Keystone (identità) 2/2

- Supporta diversi back-end come identity provider.
  - il back-end di default è MySQL
  - è possibile memorizzare credenziali e dati di autorizzazione in un back-end separato (ad es. LDAP)
- E' possibile utilizzare un meccanismo di autenticazione esterna.
- Può essere esteso per supportare ad esempio meccanismi di autenticazione federata di tipo single sign-on come Shibboleth, basati sul Security Assertion Markup Language (SAML), al fine ad esempio di facilitare una integrazione con eventuali sistemi SAML-based già in uso.

# Glance (gestione immagini)

- È un servizio autonomo (può essere installato anche indipendentemente da OpenStack) che fornisce un catalogo per la memorizzazione e la gestione di immagini virtuali.
- È diviso in tre blocchi fondamentali:
  - Le API.
  - Un database (spesso MySQL o SQlite).
  - Il registro delle immagini. Questo può essere (come nei diagrammi precedenti) Swift, ma anche un altro tipo di servizio di memorizzazione delle immagini.
- Supporta immagini di diversi formati
  - Raw, AMI, VHD (Hyper-V), VDI (VirtualBox), qcow2 (QEMU/KVM), VMDK (VMware), OVF (VMware, altri).
- Può servire per salvare snapshots delle VM

# Nova (compute)

- È un framework per la fornitura e la gestione su larga scala di istanze virtuali.
- Supporta diversi tipi di hypervisor (cf. http://goo.gl/mVdjG per dettagli).
  - XenServer, KVM, QEMU, LXC, ESXi, Hyper-V
- All'interno di nova, una parte di cloud controller si occupa dell'orchestrazione della comunicazione tra i vari componenti.
  - Utilizza un sistema di messaggistica basato su code che sfrutta l'Advanced Message Queuing Protocol (AMQP, http://goo.gl/WmxSO). L'implementazione di AMQP può essere data ad esempio da prodotti come Apache Qpid, RabbitMQ o ZeroMQ.
  - Utilizza un database per la memorizzazione delle configurazioni e degli stati a run-time dell'infrastruttura, ad esempio quali tipi di istanza sono disponibili, quali istanze sono in uso, i progetti, gli utenti, le reti, etc. Normalmente si usa MySQL o PostgreSQL.

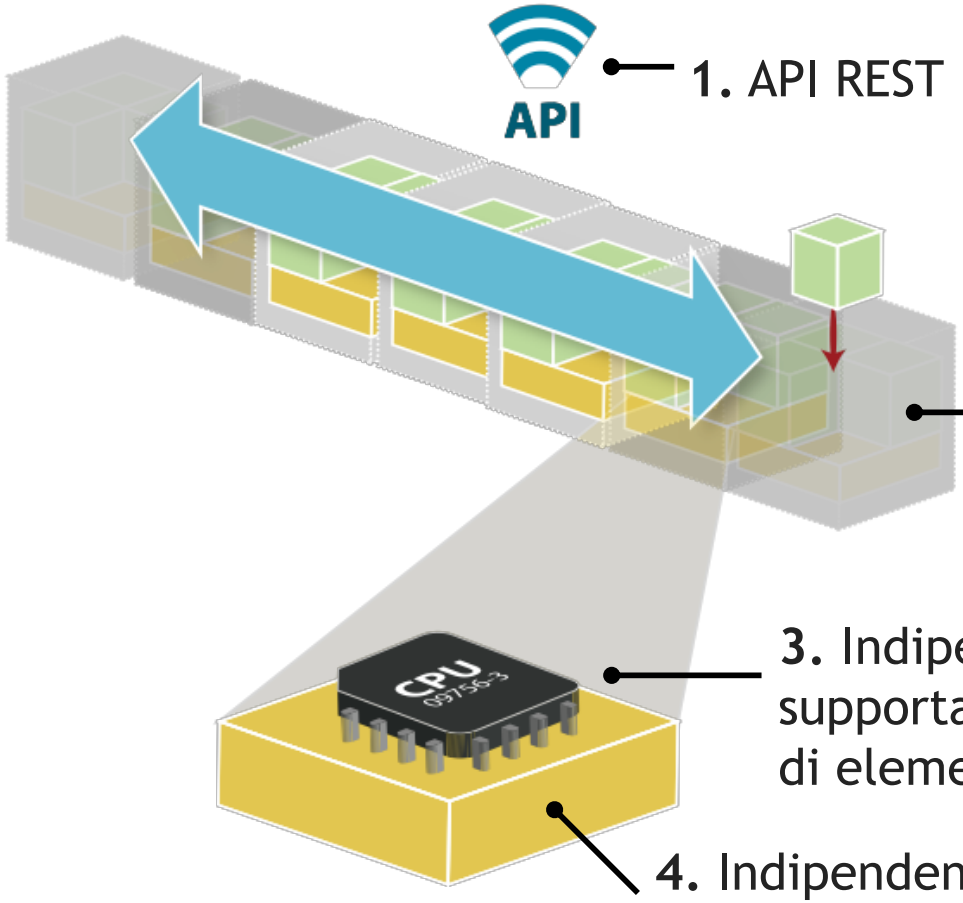
# Nova - servizi

- nova-api
  - accetta e risponde alle richieste di computazione da parte degli utenti. Supporta le API Compute di OpenStack, EC2 di Amazon e delle API Admin per permettere ad utenti privilegiati di effettuare operazioni di amministrazione.
- nova-scheduler
  - prende da una coda una richiesta di creazione di una macchina virtuale e determina su quale host fisico deve essere eseguita
  - sistema modulare personalizzabile con algoritmi complessi
- nova-compute
  - è principalmente un demone che crea e termina le istanze di macchine virtuali attraverso le API dell'hypervisor. Preleva i task da una coda, esegue i comandi di sistema relativi al task prelevato ed aggiorna nel database lo stato delle operazioni effettuate.
- nova-console, nova-vncproxy, nova-consoleauth
  - si occupano della gestione dell'accesso alle console delle macchine virtuali

# Nova - servizi deprecati

- nova-volume
  - gestione dei volumi di disco
  - ora deprecato, non esiste più da Grizzly in poi. Va utilizzata il componente specifico chiamato *Cinder.*
- nova-network
  - gestione della rete
  - probabilmente deprecato nelle prossime versioni di OpenStack
  - è stato rimpiazzato dal componente dedicato chiamato *Neutron* (ex *Quantum*)

# Nova - schema grafico (1/2)



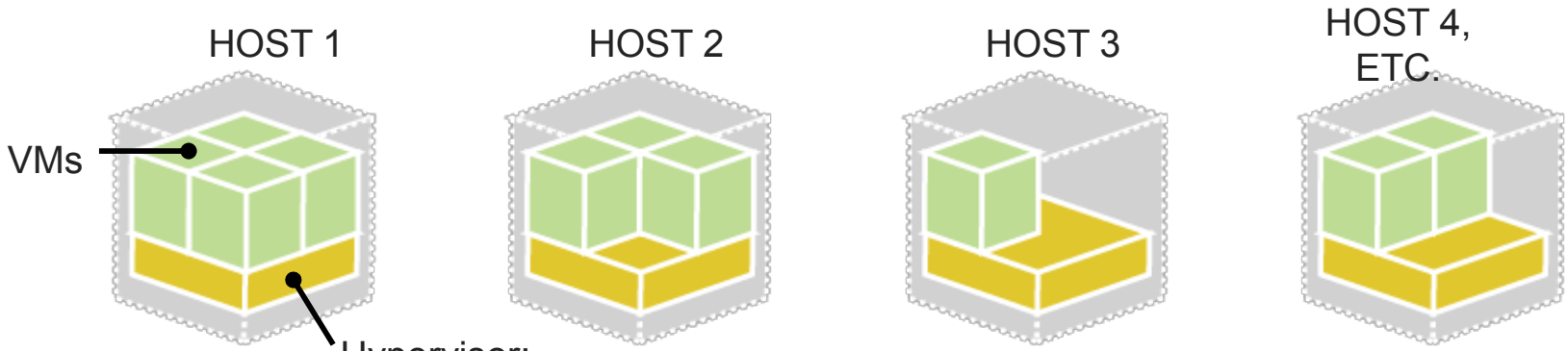

**1.** API REST

**2.** Scalabilità orizzontale

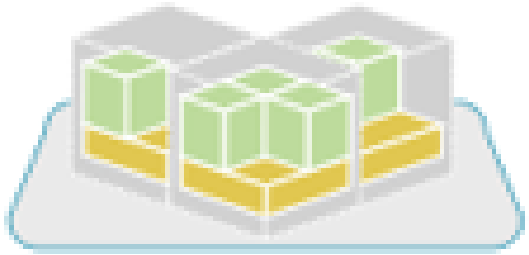
**3.** Indipendenza dall'hardware: supporta una grande quantità di elementi hardware

**4.** Indipendenza dall'hypervisor: supporta Xen, Citrix XenServer, Microsoft Hyper-V, KVM, UML, LXC and ESX

# Nova - schema grafico (2/2)




- gli hypervisors forniscono un livello di astrazione tra le applicazioni e l'hardware
- sistema operativo e applicazioni sono eseguiti sulle VM invece che su macchine fisiche

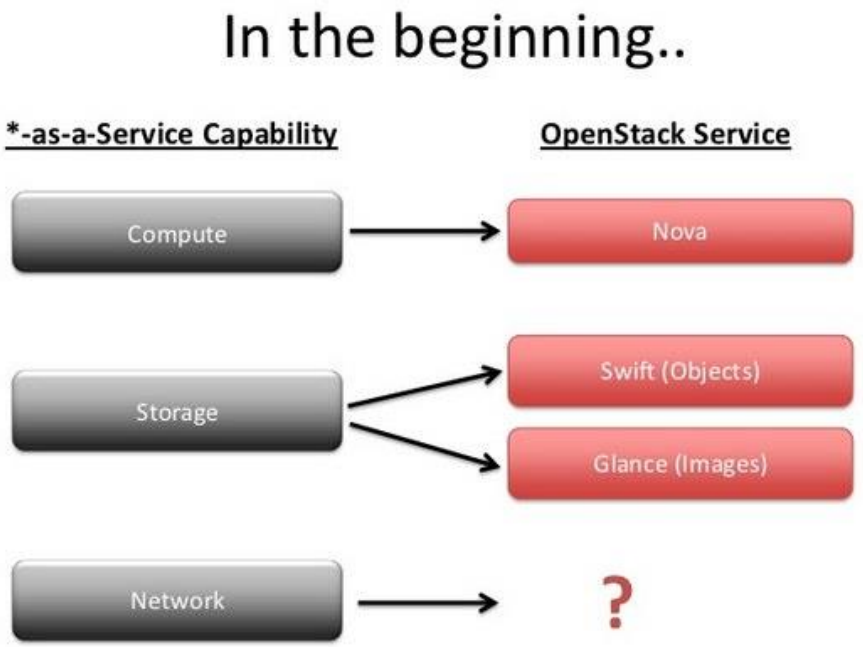


# Neutron (rete)

- In versioni precedenti a Folsom di OpenStack, la parte rete era gestita direttamente e solamente da nova, attraverso la parte `nova-network`.
  - `nova-network` è tuttora supportato ma si pensa che verrà deprecato a breve.
- A partire da Folsom, è disponibile il componente Neutron (si chiamava Quantum, è stato rinominato per questioni di copyright) che implementa il concetto di "Network as a Service".
  - "Servizio" tra interfacce (per esempio interfacce virtuali) gestite da Nova.
  - Come altri componenti in OpenStack, è composto da plug-in per la massima configurabilità.
  - E' possibile di scrittura di plug-in esterni per realizzare ad esempio servizi come:
    - Load Balancer as a Service (LBaaS), cf. http://goo.gl/mtmxsY
    - VPN as a Service (VPNaaS) per site-to-site IPSec VPNs, cf. http://goo.gl/da4NnM
    - Firewall as a Service (FWaaS), cf. http://goo.gl/dwO9KI
  - Tuttavia questo avviene a fronte di una notevole potenziale complessità (e Neutron è un componente relativamente giovane).

# Perché Neutron?

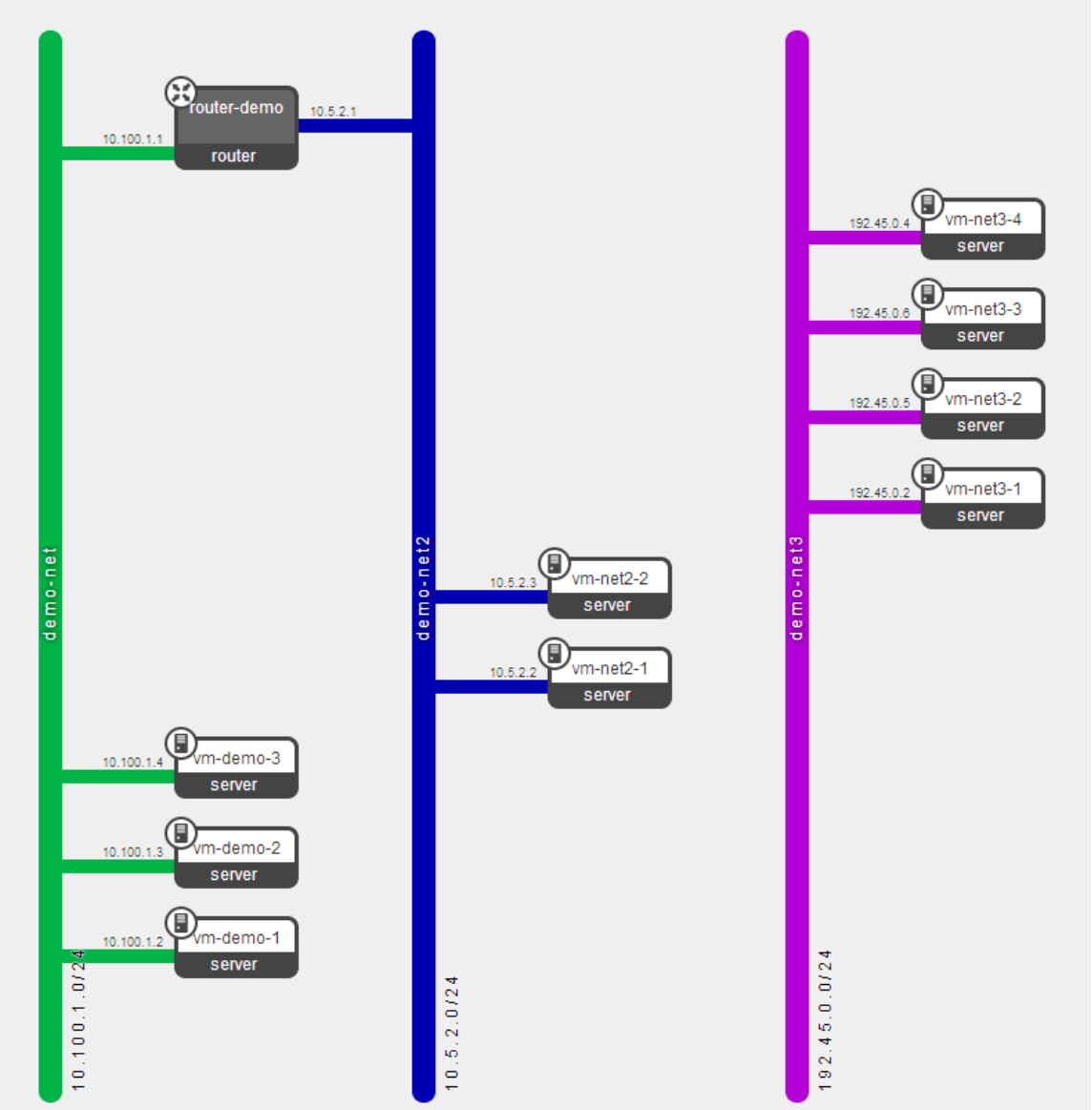
- Pre-Folsom, la rete è un sotto-componente di Nova
- Problemi:
  - Limitazioni dovute al modo in cui la rete è implementata in nova-network
  - Assenza di controllo della topologia di rete, impossibilità di gestire servizi avanzati di rete (ad esempio Dynamic Virtual Networks)
- Cf. http://goo.gl/dU59Y



# Tipologie di rete (1/3)

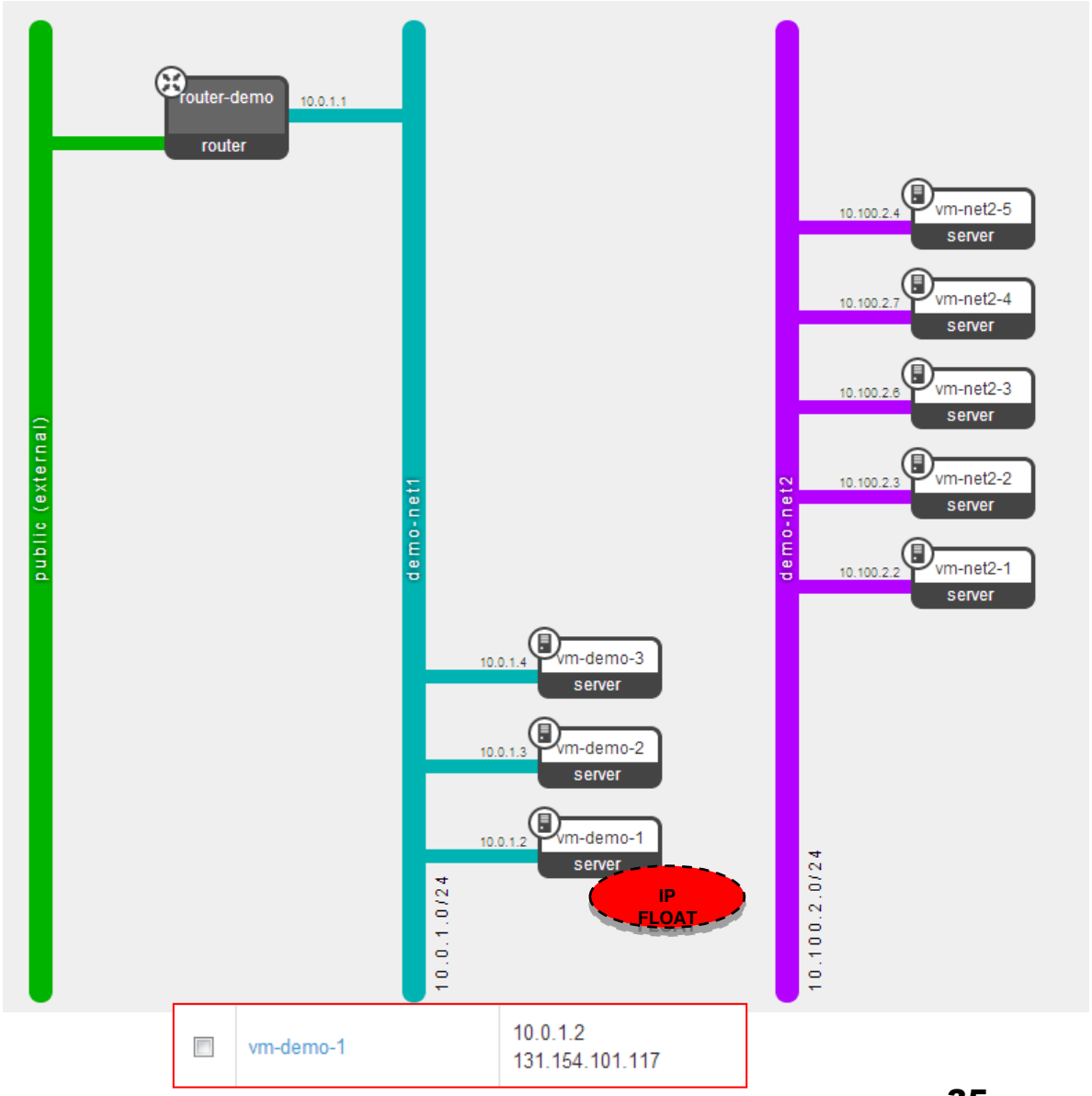
- Private Tenant Network
  - All'interno di ogni progetto (Tenant), gli utenti possono creare una o piu' reti private e uno o piu' apparati di rete (router) con cui connettere le reti in vario modo.
  - La definizione delle reti in ogni tenant non richiede l'intervento di un amministratore dell'infrastruttura ed e' garantito l'isolamento delle VM sia tra progetti diversi che tra reti diverse all'interno dello stesso progetto.
  - Le VM possono (o meno) avere outbound connectivity (Masquerade NAT), ma non sono raggiungibili dall'esterno se non hanno un floating IP assegnato (vedi slide successive).

# Private Tenant Network



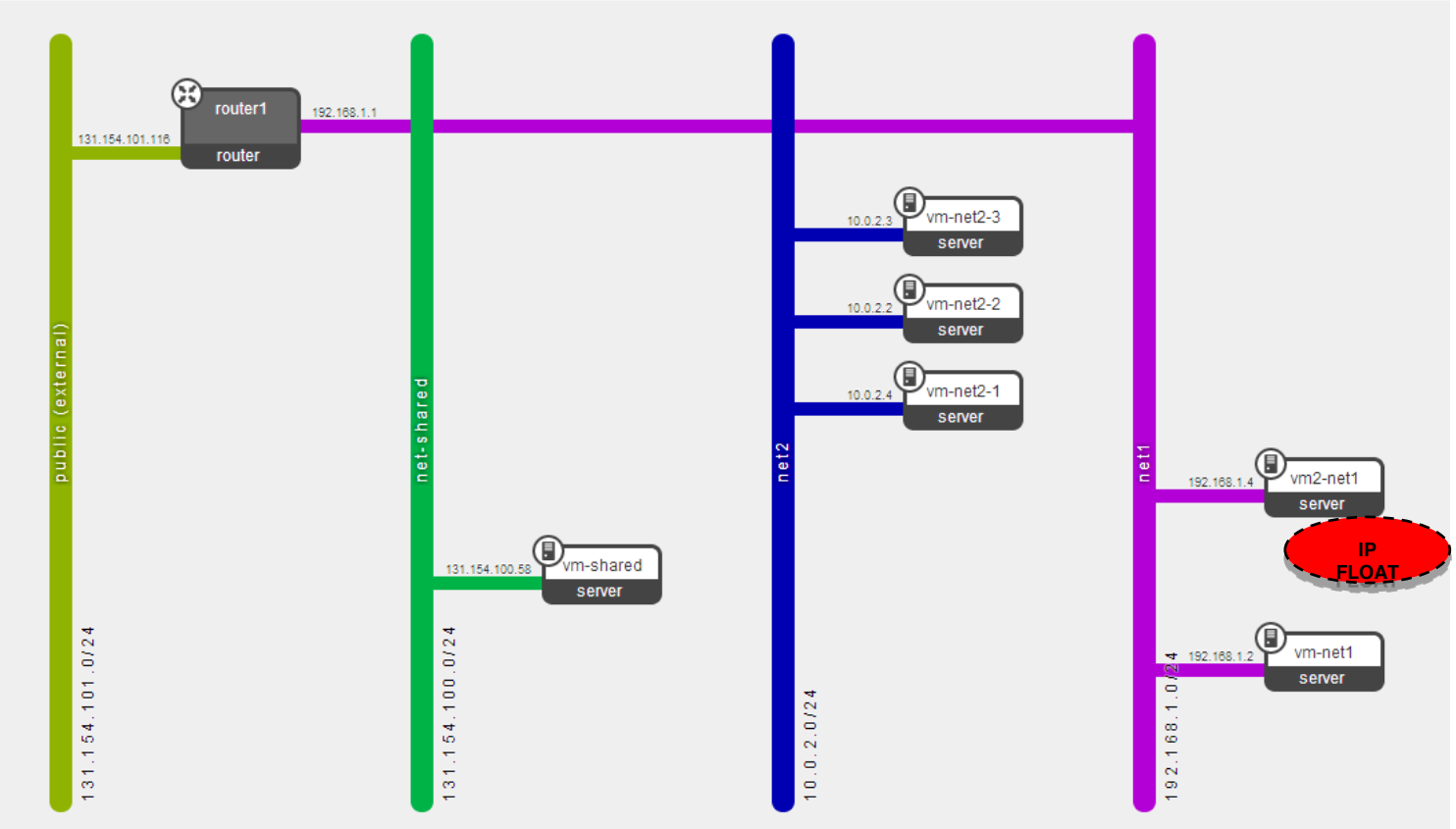
# Tipologie di rete (2/3)

- External Network
  - Tipologia di rete necessaria per assegnare floating IP a VM istanziate sulle Private Tenant Network e renderle quindi accessibili via NAT dall'esterno.
  - Le reti External sono condivise tra tutti i progetti e definite dall'amministratore dell'infrastruttura.
  - Le VM non possono partire con un ip assegnato su una rete External, deve esistere una rete Private.
- Workflow
  - L'utente crea una rete privata
  - L'utente crea un router che connette la rete External con la rete privata
  - L'utente fa partire una VM sulla rete privata a cui assegna anche un floating IP
  - La VM ha due IP: uno privato e uno pubblico



# Tipologie di rete (3/3)

- Shared Network
- La shared network e' una tipologia di rete attraverso la quale si mette a disposizione dell'infrastruttura OpenStack una rete esistente nel centro di calcolo per poter creare delle VM sulla rete stessa.
- Le VM possono essere instanziate sulla rete shared, l'IP viene loro fornito da un DHCP esterno.
- Le policy della rete shared non sono gestite da OpenStack. No NAT a nessun livello.



# OpenStack Storage

- Lo “storage” appare in diversi moduli di OpenStack ed è importante chiarirne i diversi tipi:

| On-instance / ephemeral | Volumes block storage (Cinder) | Object Storage (Swift) |
|---|---|---|
| Used for running Operating System and scratch space | Used for adding additional persistent storage to a virtual machine (VM) | Used for storing virtual machine images and data |
| Persists until VM is terminated | Persists until deleted | Persists until deleted |
| Access associated with a VM | Access associated with a VM | Available from anywhere |
| Implemented as a filesystem underlying OpenStack Compute | Mounted via OpenStack Block-Storage controlled protocol (for example, iSCSI) | REST API |
| Administrator configures size setting, based on flavors | Sizings based on need | Easily scalable for future growth |
| Example: 10GB first disk, 30GB/core second disk | Example: 1TB "extra hard drive" | Example: 10s of TBs of dataset storage |

Fonte: http://goo.gl/fud7nW

# Cinder (block storage)

- Nel *block storage* si creano dischi virtuali da legare alle singole macchine virtuali.
- Questa tipologia di storage permette di estendere la quantità di spazio di archiviazione che le macchine virtuali nell'infrastruttura di cloud computing hanno a disposizione, legando un numero arbitrario di dischi, secondo le specifiche necessità.
- Cinder è un servizio che precedentemente aveva il nome di *nova-volume*.
  - Separato da nova per l'aumentata complessità del servizio stesso, per l'introduzione di uno scheduler, per l'interazione con altri componenti.
- Pensiamo ad una chiavetta USB che possiamo di volta in volta montare su pc diversi; in maniera analoga un volume di Cinder puo' essere montato di volta in volta su VM diverse.

# Swift (object storage)

- Swift è un object store distribuito, scalabile, multi-tenant e ad alta disponibilità.
  - Il goal è la memorizzazione a basso costo di grandi moli di dati non strutturati attraverso API di tipo REST.
  - Supporta direttamente le API S3 di Amazon, gestisce quote, controllo accessi e si interfaccia a diversi sistemi di storage.
  - Non è:
    - Un file system (non si può montare sulle VM).
    - Pensato per memorizzare dei DB live o delle gerarchie di file.
- È pensato per scalabilità, alta affidabilità, gestione di elevata concorrenza nelle transazioni, storage replicato geograficamente.
  - Es. il San Diego Supercomputing Center ha una installazione basata su Swift con oltre 5 PB di spazio disco (http://goo.gl/jmxJK).
- Una possibile applicazione di Swift non legata direttamente a Cloud storage ad esempio può essere quella di utilizzarlo per la distribuzione geografica di immagini Glance.

# Ceilometer (telemetry)

- **Misurazione del consumo delle risorse**
  - per scopi di monitoring e accounting
- Questo aspetto è stato inizialmente lasciato fuori da OpenStack – la preferenza è stata di concentrarsi prima sulle caratteristiche IaaS di base
- Tuttavia il controllo dell'uso delle risorse è fondamentale → **Ceilometer**
  - Progetto iniziato a maggio 2012

# Ceilometer: workflow

- Raccolta delle metriche di utilizzo dai componenti Openstack
- Eventuale aggregazione delle metriche raccolte
- Pubblicazione delle metriche verso diverse destinazioni
  - incluso il collector di Ceilometer stesso
- Conservazione delle metriche
  - di default in un database MongoDB
- Lettura delle metriche attraverso API Rest

# Heat (orchestration)

- Servizio di **orchestrazione di componenti**, supportato a partire dalla release Grizzly
- Stack
  - Un insieme di risorse Cloud connesse tra loro (ad es. VM, volumi, reti, etc.)
- Gli stack vengono creati attraverso templates
  - Utilizzano le stesse strutture e le stesse astrazioni presenti in Amazon CloudFormation (http://goo.gl/4tvGl)
  - Si integrano con tool di configurazione automatizzata come Puppet
  - Possibilità di gestire alta affidabilità e autoscaling in maniera automatica, ricevendo dati da Ceilometer

# Distribuzione utenti


**1780** Survey Responses

**506** Deployments

**512** Companies

**293** UG Members

**Top 10 Countries (deployments)**

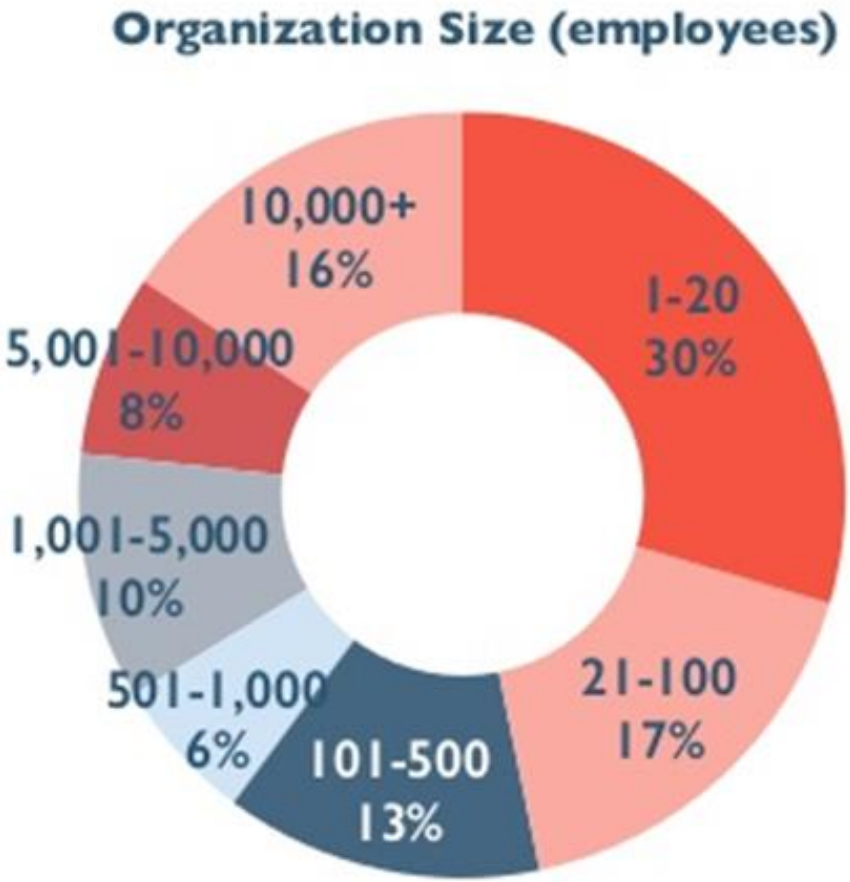
| | |
|---|---|
| US | 171 |
| CN | 36 |
| IN | 35 |
| FR | 31 |
| GB | 22 |
| CA | 21 |
| AU | 18 |
| CH | 14 |
| DE | 14 |
| BR | 9 |

**Top 10 Countries (all)**

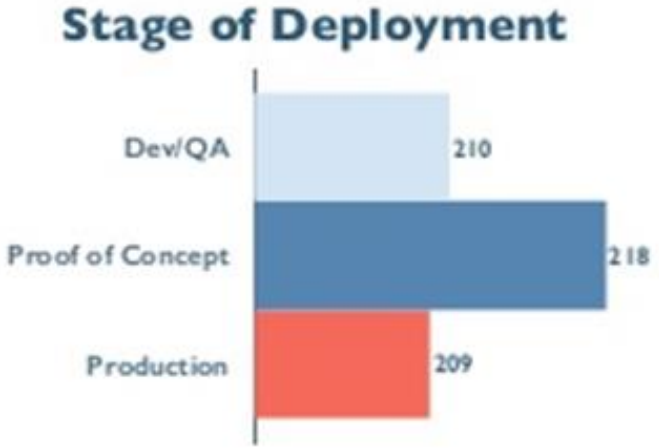
| | |
|---|---|
| US | 418 |
| CN | 83 |
| IN | 78 |
| FR | 53 |
| GB | 32 |
| CA | 36 |
| AU | 30 |
| DE | 30 |
| JP | 25 |
| BR | 19 |

Fonte: goo.gl/ZrrQJp

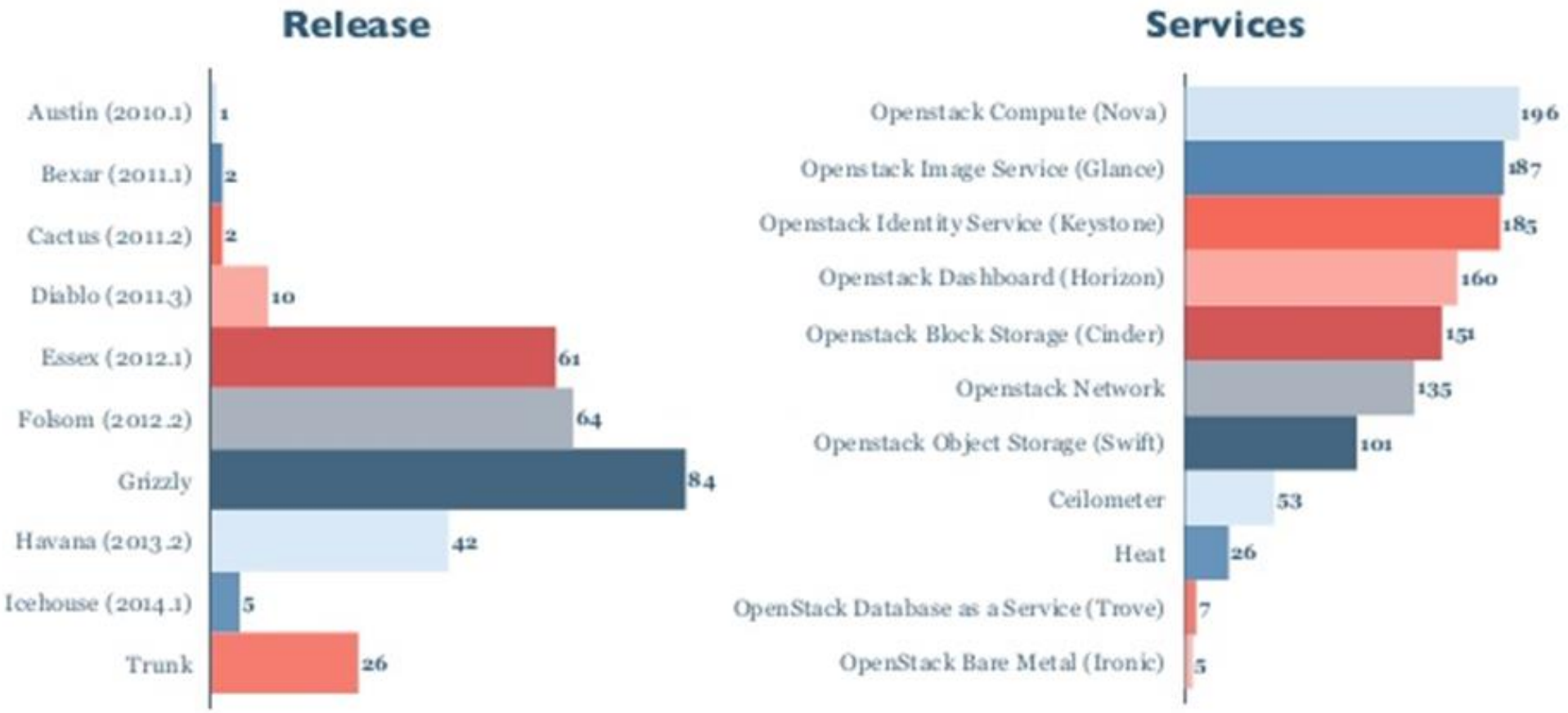
# Tipi di utilizzatori







# Release e servizi in produzione



# Cosa piace di OpenStack

- Community
- Open Source
- Flexibility
- Extendible
- Code is accessible and easy to understand
- 'Not a Lot'
- Scalable
- Easy automation
- 'It pays my salary'

# Cosa si vorrebbe in futuro

- Stability of core should be a priority above adding new functions
- Add how-to guides, problem management documentation, expire old documentation, end user guide **(but much less than previous surveys)**
- Zero downtime migrations
- Installation and configuration **(but much less than previous surveys)**
- Cross Project consistency with APIs, SDKs and CLIs
- High availability VMs
- Neutron stability, simplification, resilience, IPv6 and scalability
- Improved function and usability in Horizon
- Security, auditing **(but much less than previous survey)**
- AWS/EC2 compatibility

# Conclusioni

- **OpenStack** è un framework Cloud open
  - sta crescendo molto rapidamente, sia come maturità che come feature supportate
  - d'altra parte (e probabilmente è una cosa in qualche modo *voluta*) è **complicato** e richiede un certo numero di competenze, test ed ottimizzazioni
  - è il prodotto open di più grande prospettiva in ambito Cloud
  - presenta diverse opportunità come base per la creazione e lo sviluppo di applicazioni avanzate grazie alle sue possibilità di adattamento
  - è oggetto di diverse attività in corso in ambito INFN

# Riferimenti

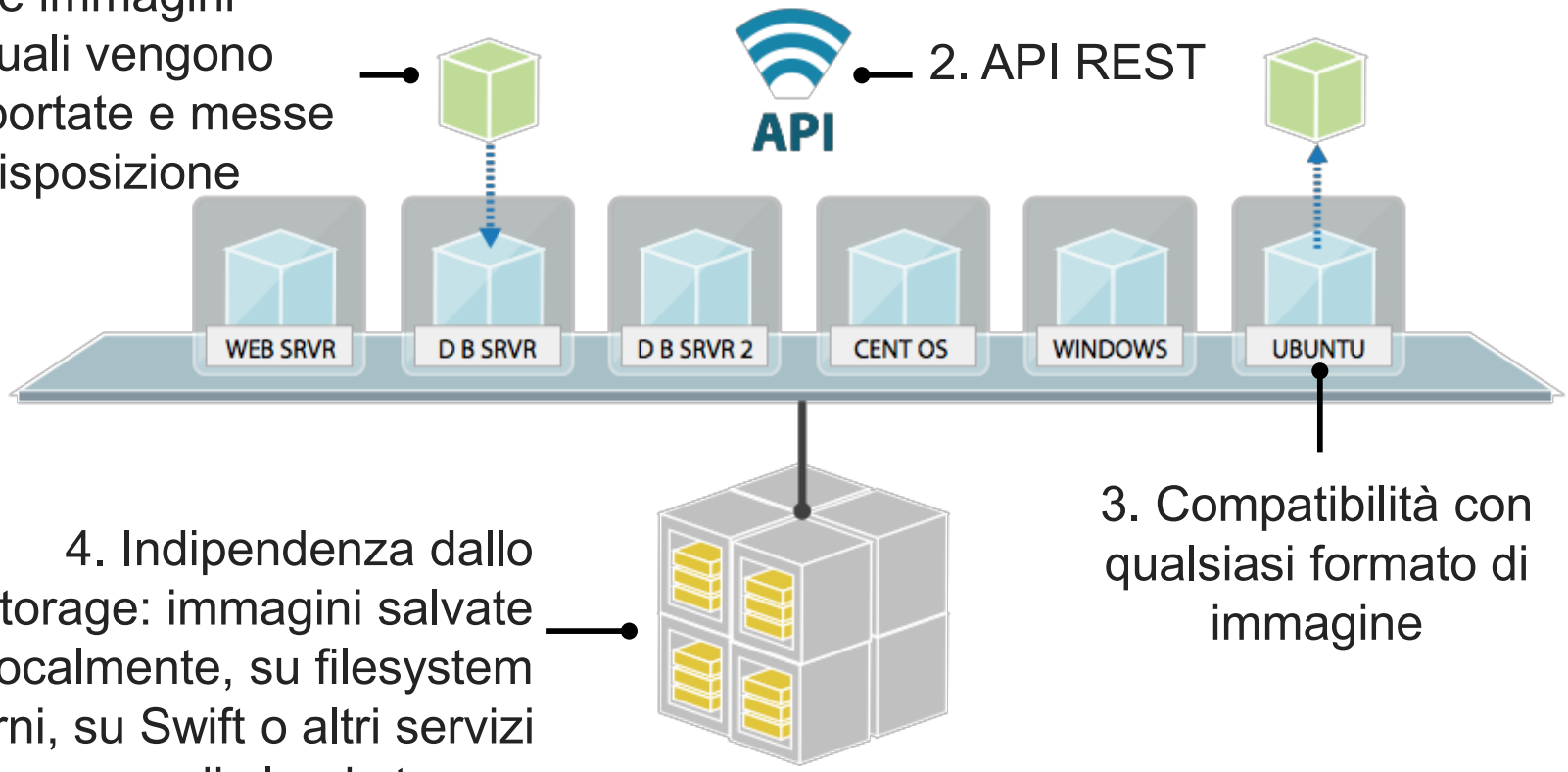
- Sito web ufficiale
  - http://www.openstack.org/
- Guida per amministratori
  - http://docs.openstack.org/admin-guide-cloud/content/
- Guida per utenti
  - http://docs.openstack.org/user-guide/content/

# Grazie per l'attenzione

per realizzare questa presentazione è stato utilizzato materiale di Paolo Veronesi ed Enrico Fattibene, INFN-CNAF

# Glance - schema grafico

1.Le immagini virtuali vengono importate e messe a disposizione




3. Compatibilità con qualsiasi formato di immagine

4. Indipendenza dallo storage: immagini salvate localmente, su filesystem esterni, su Swift o altri servizi di cloud storage