Consortium GARR | THE ITALIAN EDUCATION & RESEARCH NETWORK

# Workshop on GARR federated cloud II


Alex Barchiesi
Alberto Colla
Fulvio Galeazzi
Claudio Pisa

Matteo Di Fazio
Marco Lorini
Delia Passalacqua

a.k.a. GARR CSD department - cloud-support@garr.it


# Outline

## Day 1

- Where are we (since 2017)?
- **GARR Cloud**:
  - Architecture
  - How we built it: **MAAS - juju**
  - How we extended to container platform **k8s**
  - **Kubernetes** OpenStack keystone integration
- Charm tutorial part 1
- **lunch break**
- Charm tutorial part 2

## Day 2

- **Roundtable**:
  - backup & recovery
  - monitoring
- **Roundtable**:
  - accounting
  - administrative access control
  - security
- **lunch break**
- Cloud **Federation**
  - use cases
  - administrative delegation, access policies
- Advanced use of **CEPH** storage

# OUR idealistic starting point

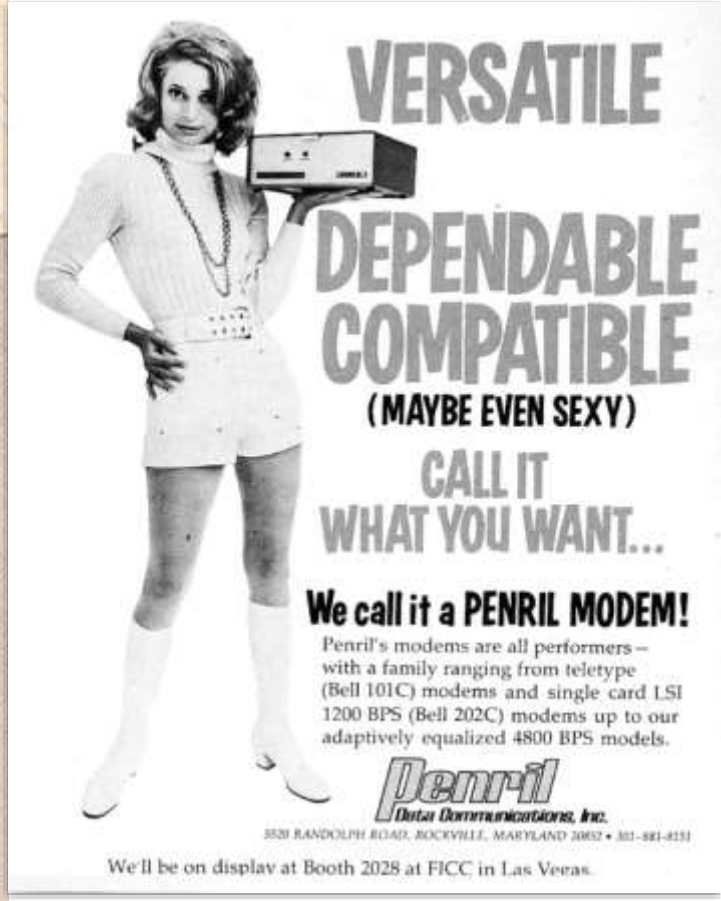
"You can never change things by fighting the existing reality.

To change something, build a new model that makes the existing obsolete."

[R. Buckminster Fuller]

M.Rangaswami (Corporate Eco Forum) comments over on his blog:

**Open Source as the antidote to Cloud Monopolies** [...]


where we come from

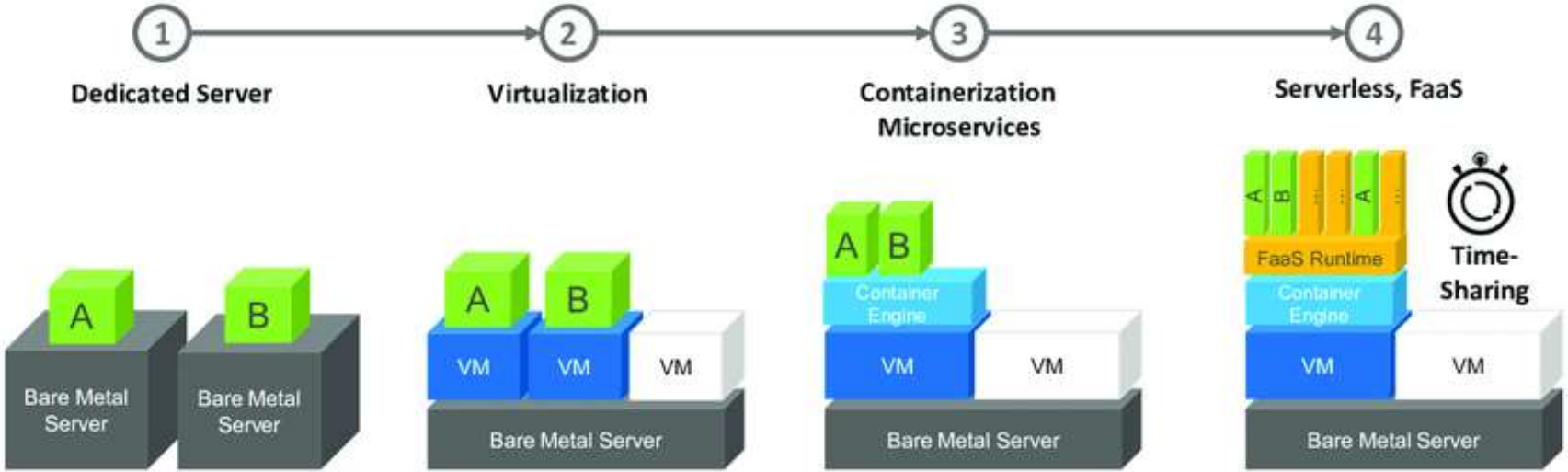
# where we are going

samsung prototype 2016

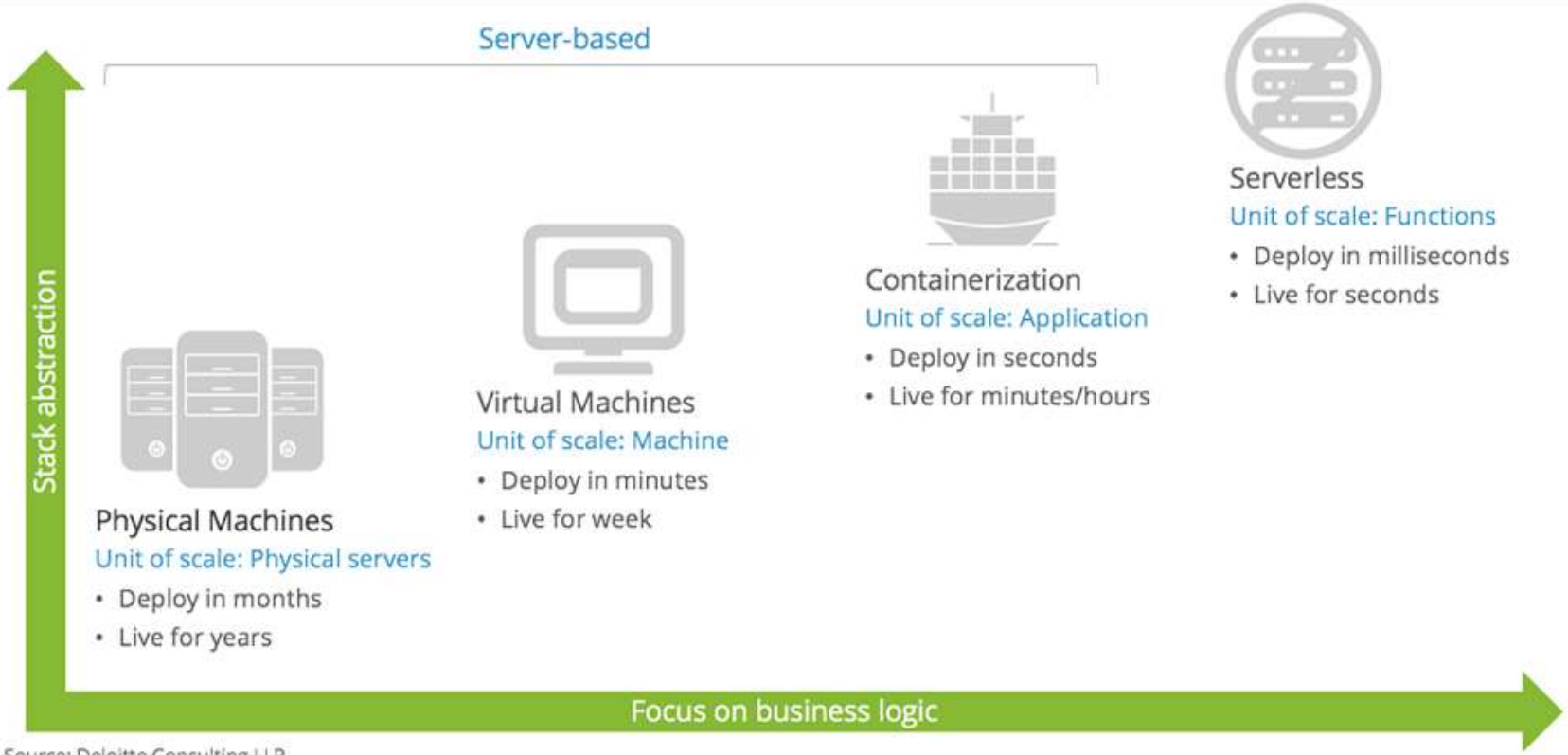
how

# A Brief History of Cloud Architectureal evolution

(from a resource utilization point of view)



# paradigm shift in computing



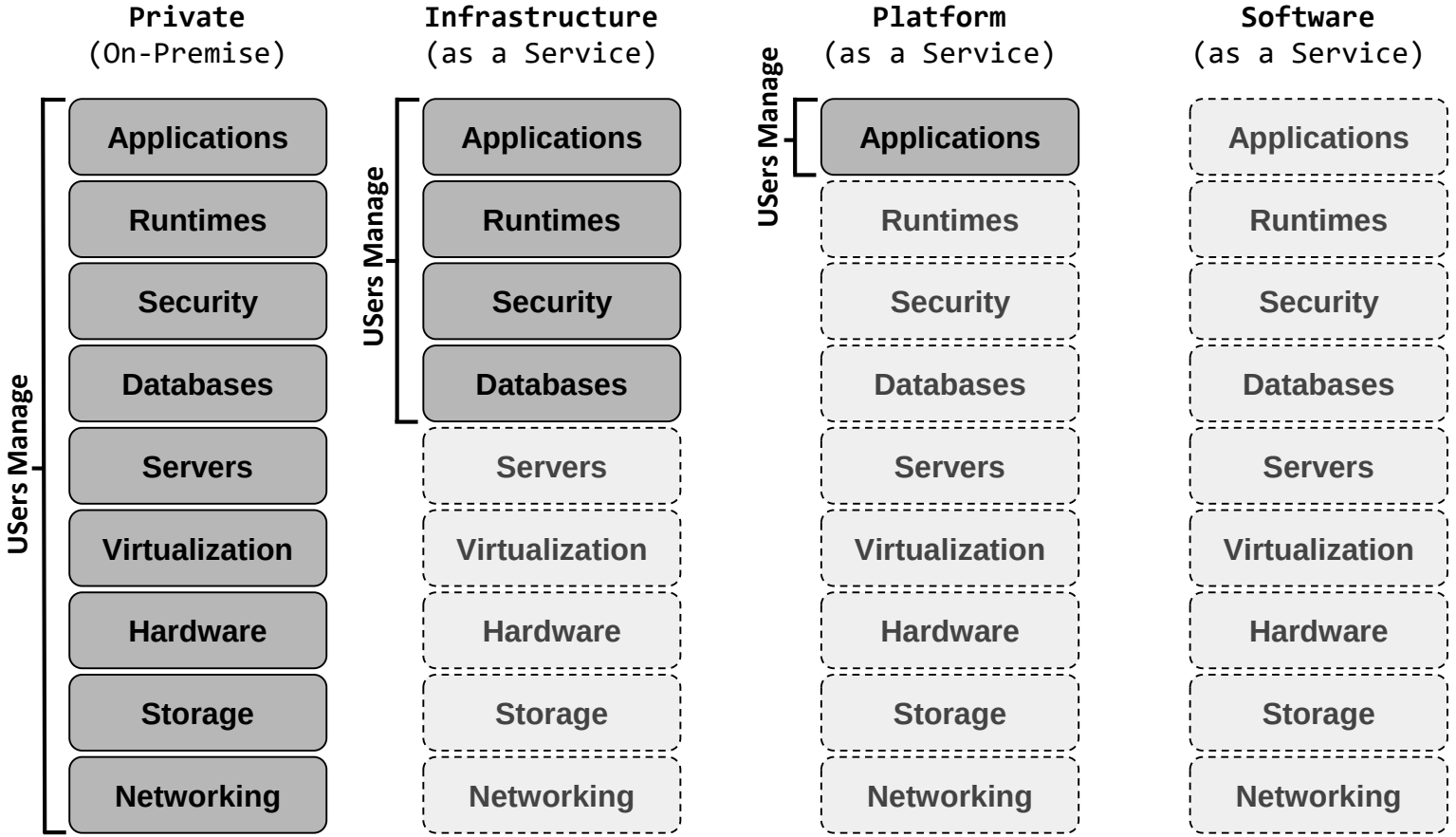

Source: Deloitte Consulting LLP

## VIRTUAL MACHINES


WHATS *the* DIFF?

## CONTAINERS

# Cloud Service Models (user view)



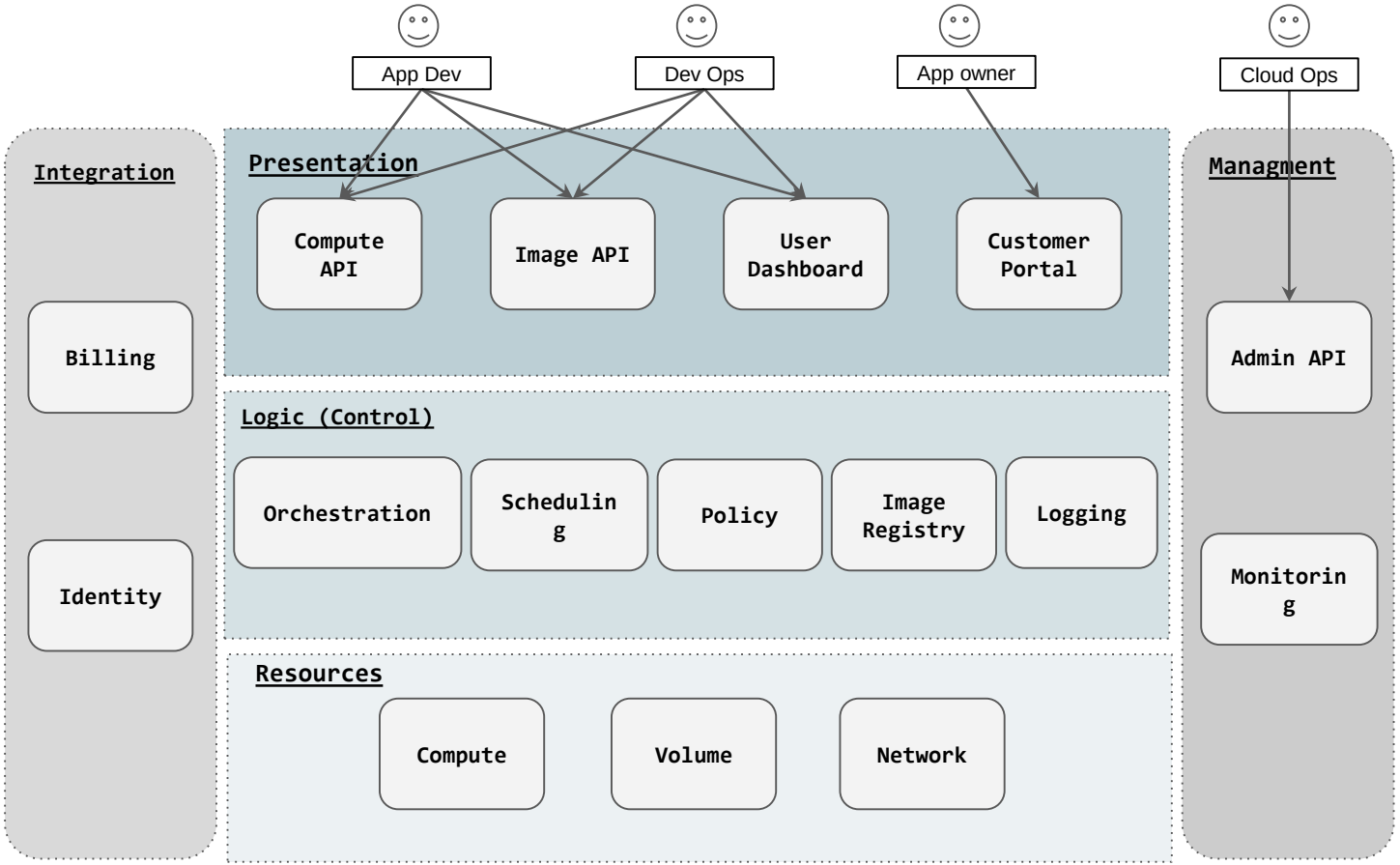
# GARR Cloud Services


- Infrastructure as a Service:
  - GARR Cloud (*OpenStack*)


- Platform as a Service:
  - GARR Container Platform (*Kubernetes*)
  - Deployment as a Service (*Juju*)


- Software as a Service:
  - GARR Workplace (*OnlyOffice*)

# Conceptual Cloud Architecture

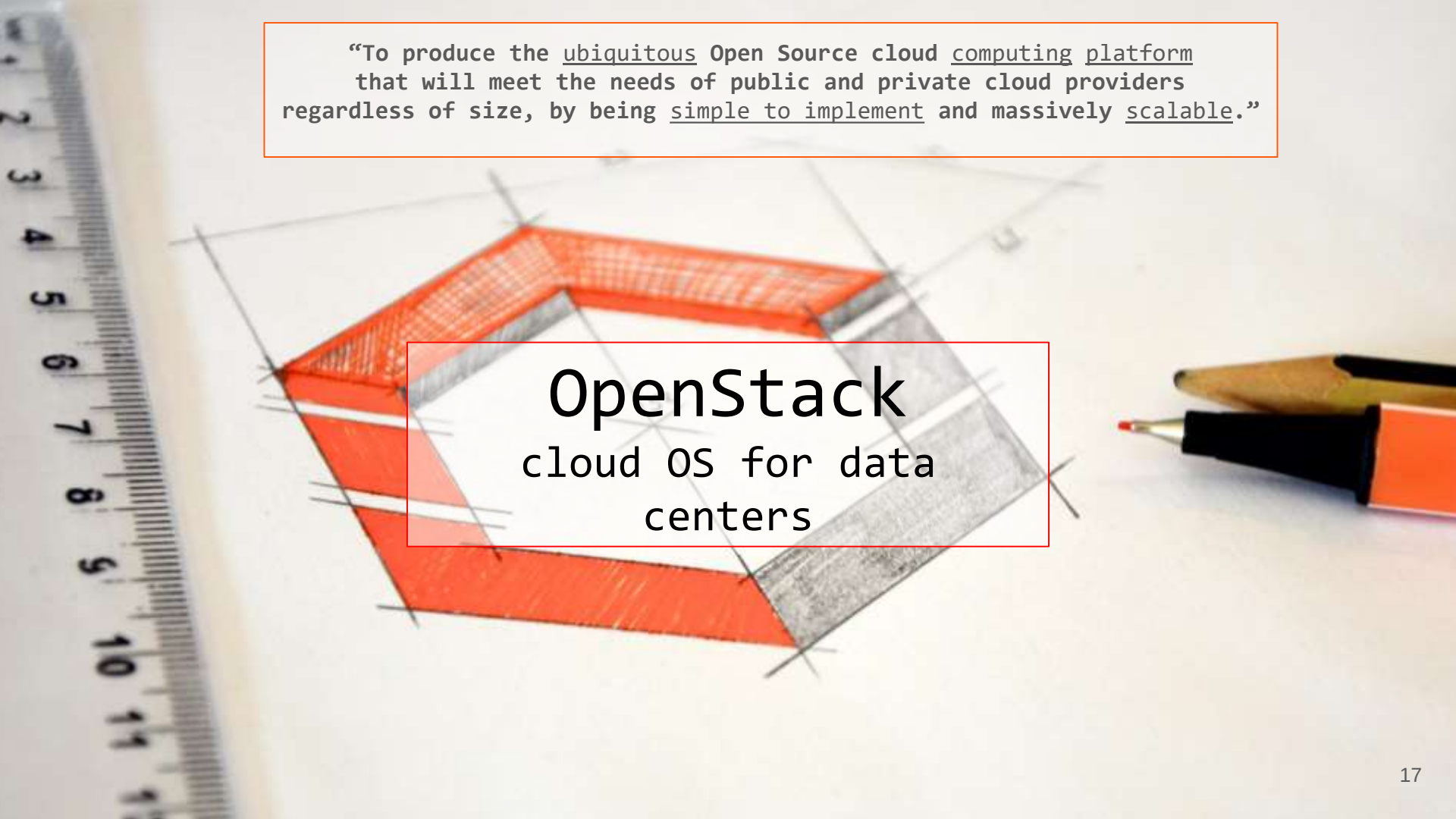



BETTY GILLIS

# (our) goals and requirements

- open-source
- reduced manpower *efforts*
- sharing resources
- simplify provisioning of storage and computing services
- serve different organizations
- unified access (SSO)
- always on
- replicable and scalable
- *self* deploying and *self* healing
- elastic
- separation / flexible security policies
- Empower users with something more than a PAAS and something easier than a IAAS

**“To produce the ubiquitous Open Source cloud computing platform that will meet the needs of public and private cloud providers regardless of size, by being simple to implement and massively scalable.”**

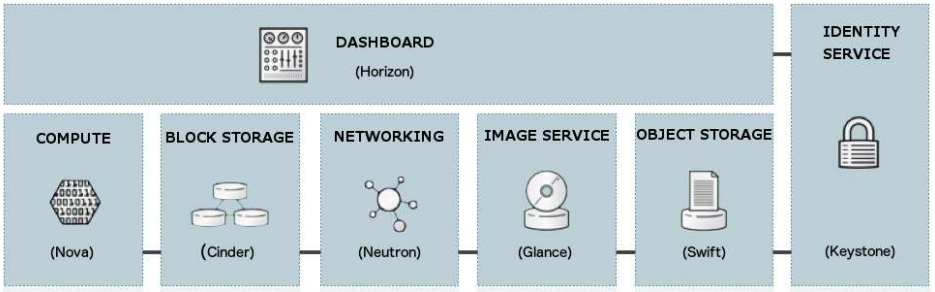


# OpenStack

## cloud OS for data centers

# OpenStack Architecture recap

- Users log into **Horizon** and initiate a VM create
- **Keystone** authorizes
- **Nova** initiates provisioning and saves state to DB
- **Nova Scheduler** finds appropriate host
- **Neutron** configures networking
- **Cinder** provides block device
- Image URI is looked up through **Glance**
- Image is retrieved via **Swift**
- VM is rendered by Hypervisor



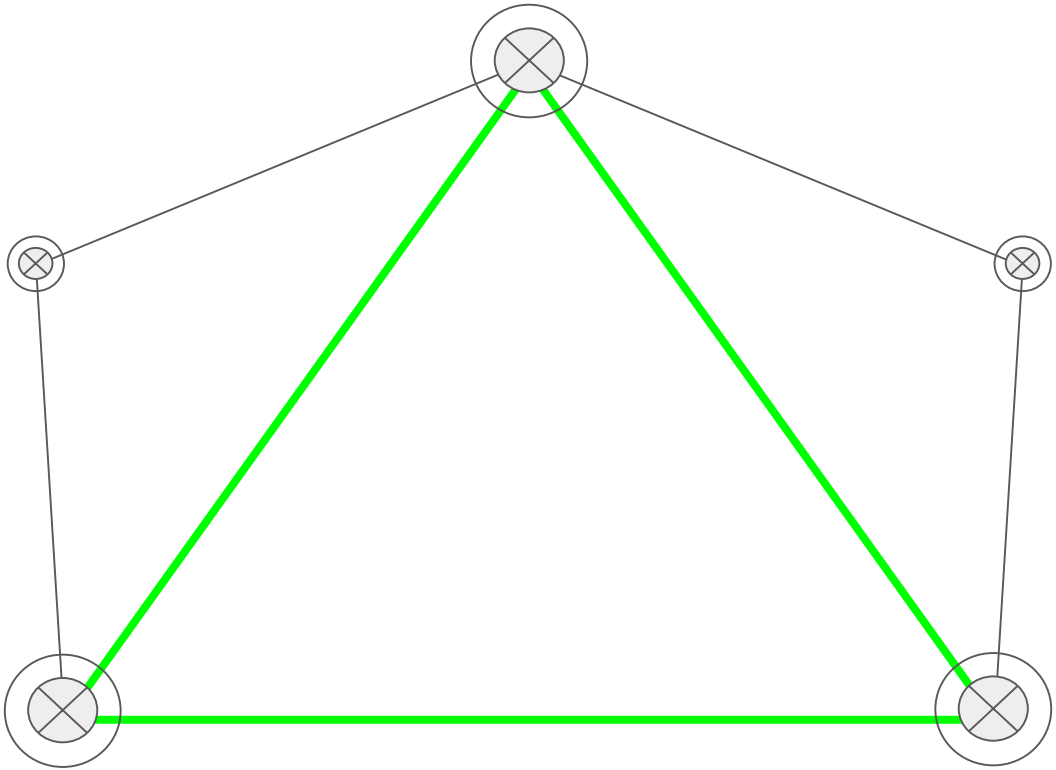
# GARR Cloud Infrastructure

# 8500 core
# 10 PB

... 11 rack/CSD-modules

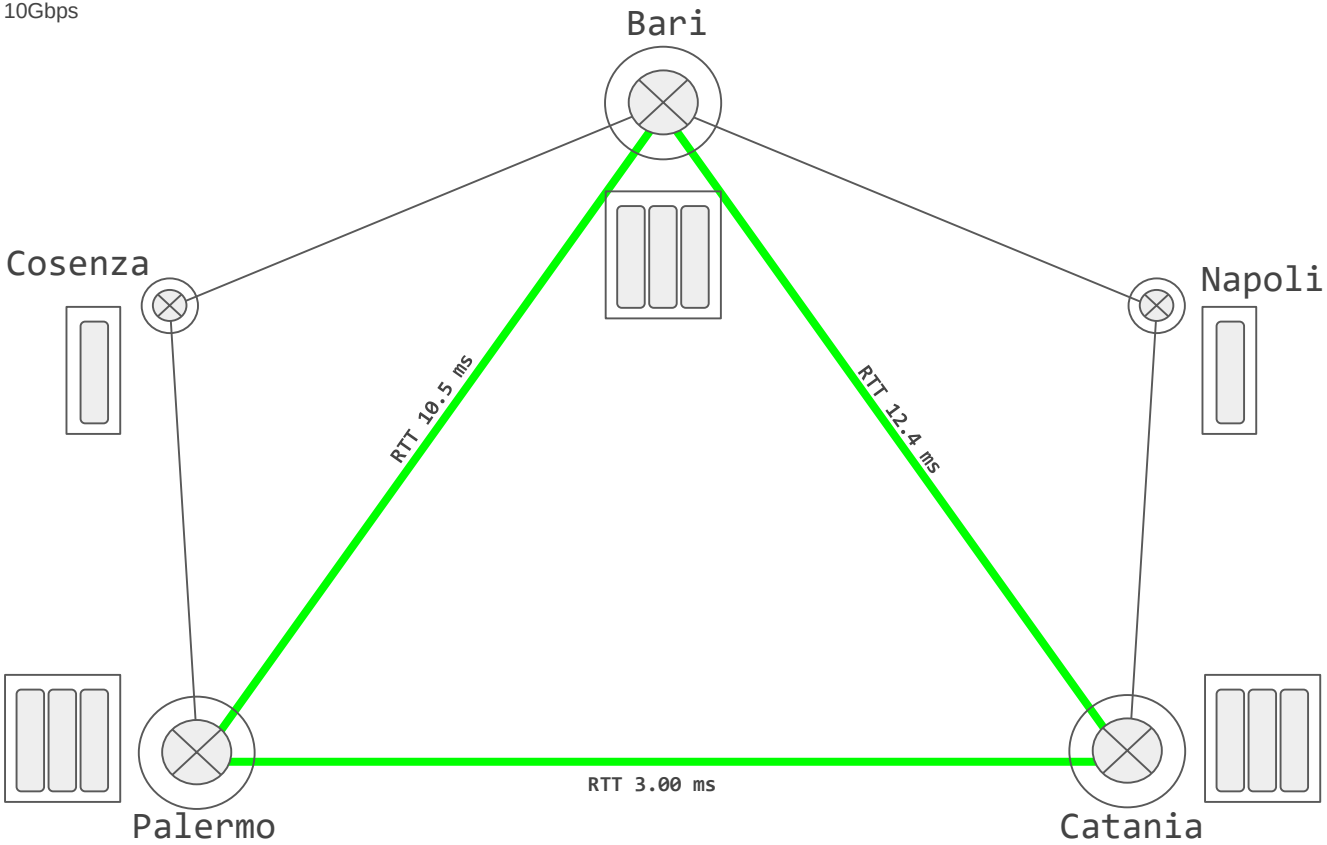
# Network

40Gbps
10Gbps

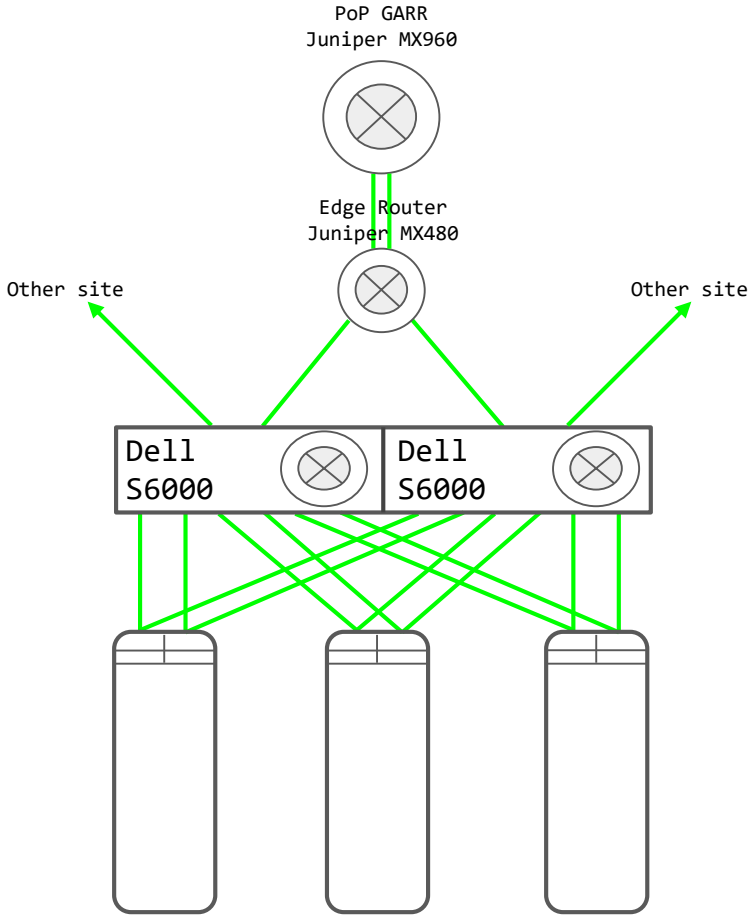


# Network

40Gbps
10Gbps



# Sito




## Data Network 40 Gbps

- 2 Switch ToR Dell MXL in each module-CSD
- 2 star Router/Switch Dell S6000
  - 32 ports x 40 Gbps

## Managment network (“ILO”) separated from data

- 2 Switch management ToR Dell S55 each module-CSD
- 2 Switch management star Dell S4810
  - 48 ports x 1 Gbps

Chassis Blade **Dell M1000e:**

- 16 server (lame) Dell Poweredge M620
- 2 switch integrati Ethernet (Dell MXL)
  - 2x16 porte 10 Gbps -> server
  - 4 uplink 40 Gbps -> centro stella;
- 2 switch Fibre Channel (Brocade M6505)
  - 16 porte a 16 Gbps verso i server
  - 8 uplink a 16 Gbps verso gli storage controller;

**2** Storage Array **MD3860f FC:**

- Dischi SAS 116x4TB + 4xSSD 1.6TB
- FiberChannel brocade controller 2x16 Gbps (2x4 porte)

# Modulo CSD

-Sistema operativo: **ubuntu** Xenial

-4 schede di rete - link aggregation **40Gbps**

**-vlan** + **Bridge** linux

**-lama** fisica fa da **GW per LXC**

**-iptables** su lama per:

-fwd, nat + sicurezza LXC

-indistinguibilità lame x LXC

LinuXContainer (LXC with LXD managment)

-opensource

-supportati da kernel moderni

-Uso efficiente HW (rispetto VM)

-Near Bare Metal runtime performance

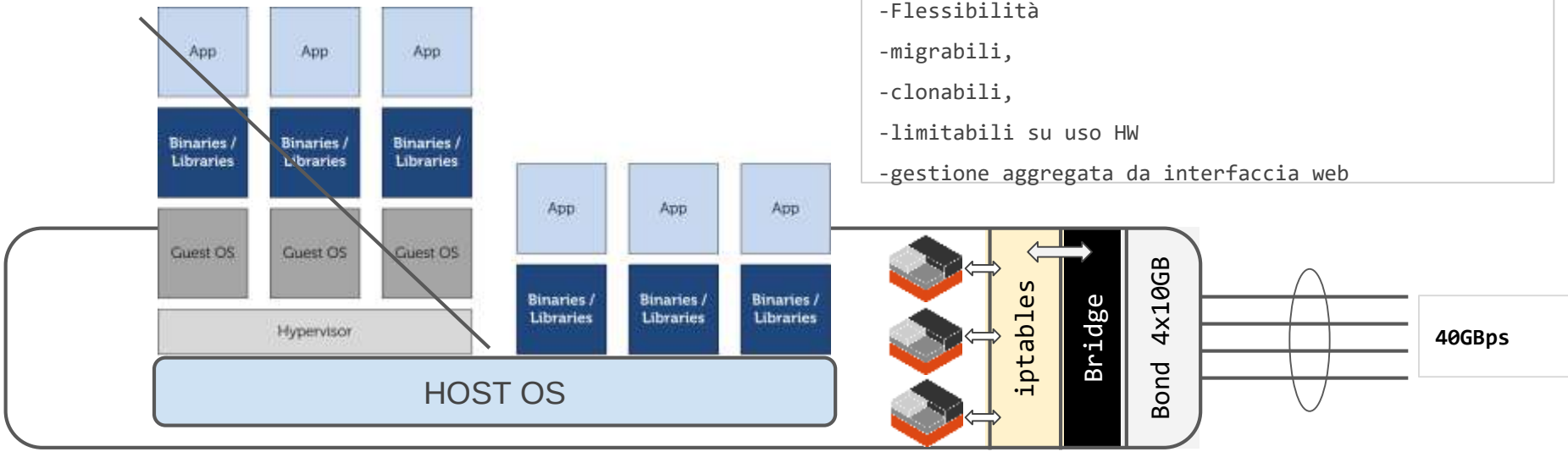
-Flessibilità

-migrabili,

-clonabili,

-limitabili su uso HW

-gestione aggregata da interfaccia web




## Lama


Federated Cloud Architecture

# multi-region (OpenStack) model

**Region**

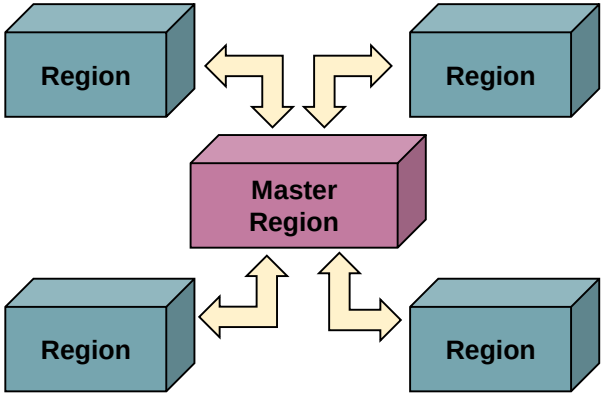
has its own deployment of OpenStack, is linked to other regions using Identity and dashboard.

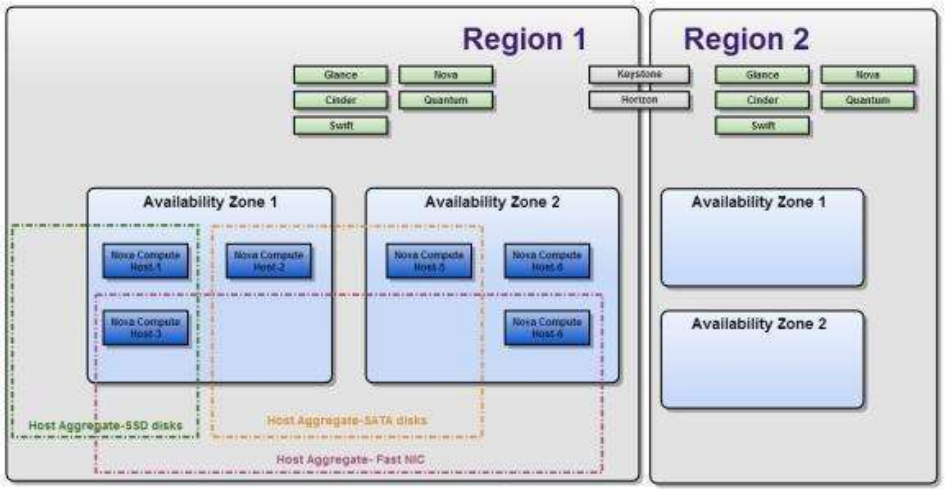
**Availability Zone**

Within each Region, nodes can be logically grouped into Availability Zones (AZ)

**Host Aggregate**

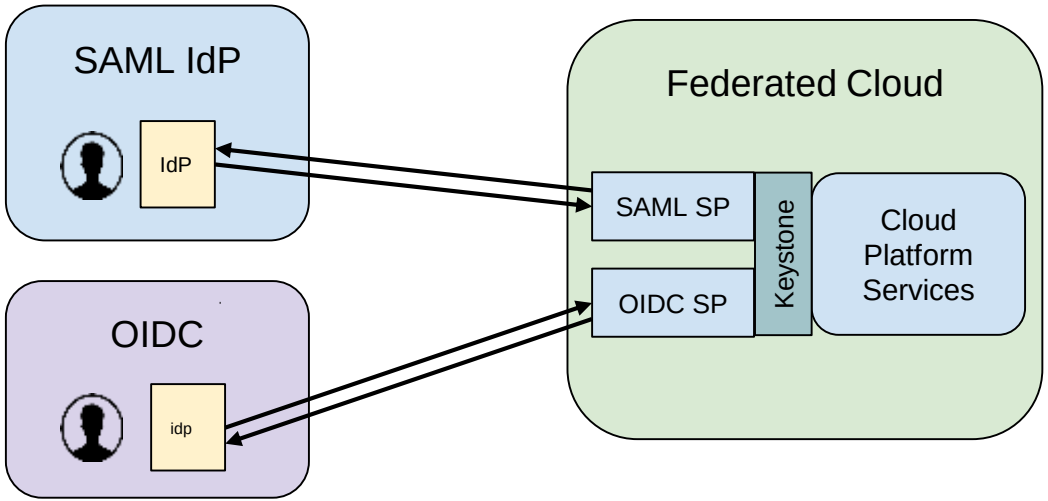
Within a Region machines can be grouped into Host aggregates. A machine may belong to multiple Host aggregates.



# Federated authentication/authorization

1. **Separation of roles**: cloud administrator and the domain administrators.
2. The federated Identity providers are **delegated** only for **authentication**
3. **No authorization information stored outside of keystone**, in order to avoid:
    a. Having to check reliability and consistency of such information
    b. Having to map it to internal keystone entities
    c. Force users to act on an IdP not under their personal control
4. **Users can be granted rights on any project** of the federation, irrespective of their affiliation and under the sole control of the administrator for that project
5. Deploy the simplest solution, relying **as much as possible on native OpenStack** capabilities avoiding any extra non necessary component.



# Building the GARR federated cloud...

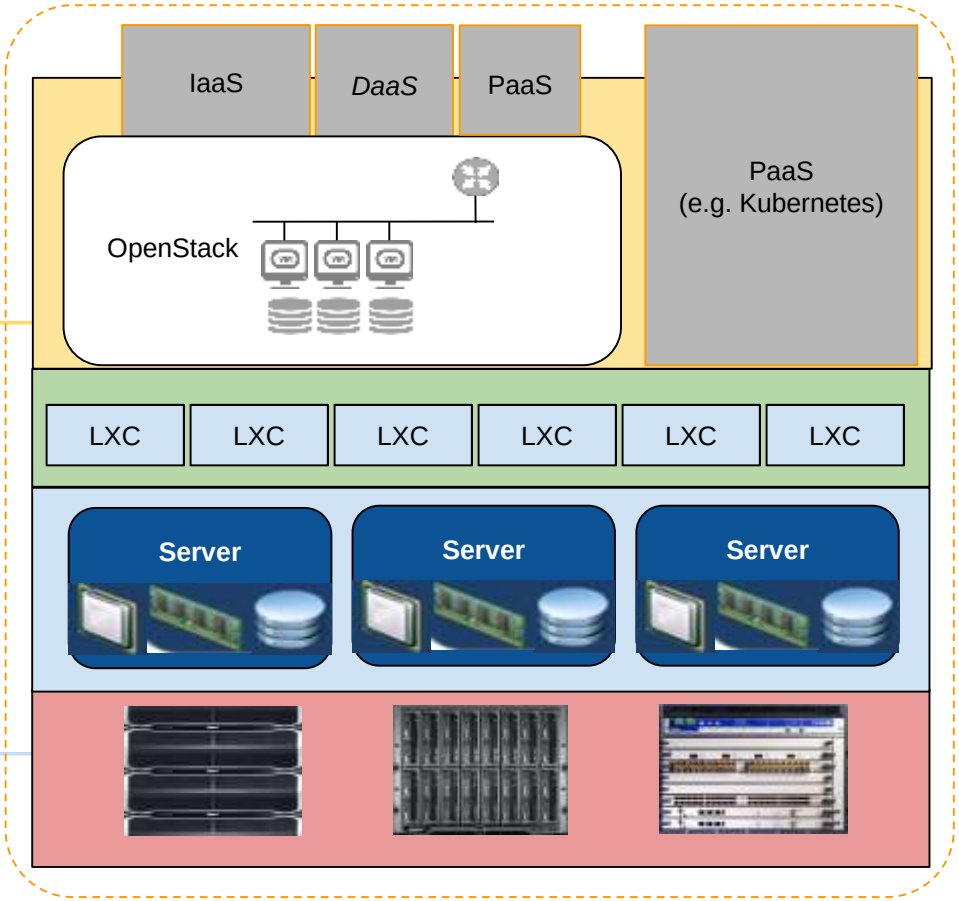
...being an extremely busy team...

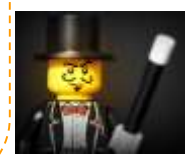
# Building the GARR federated cloud...

...being an extremely busy team...

# 4 Layers recipe:

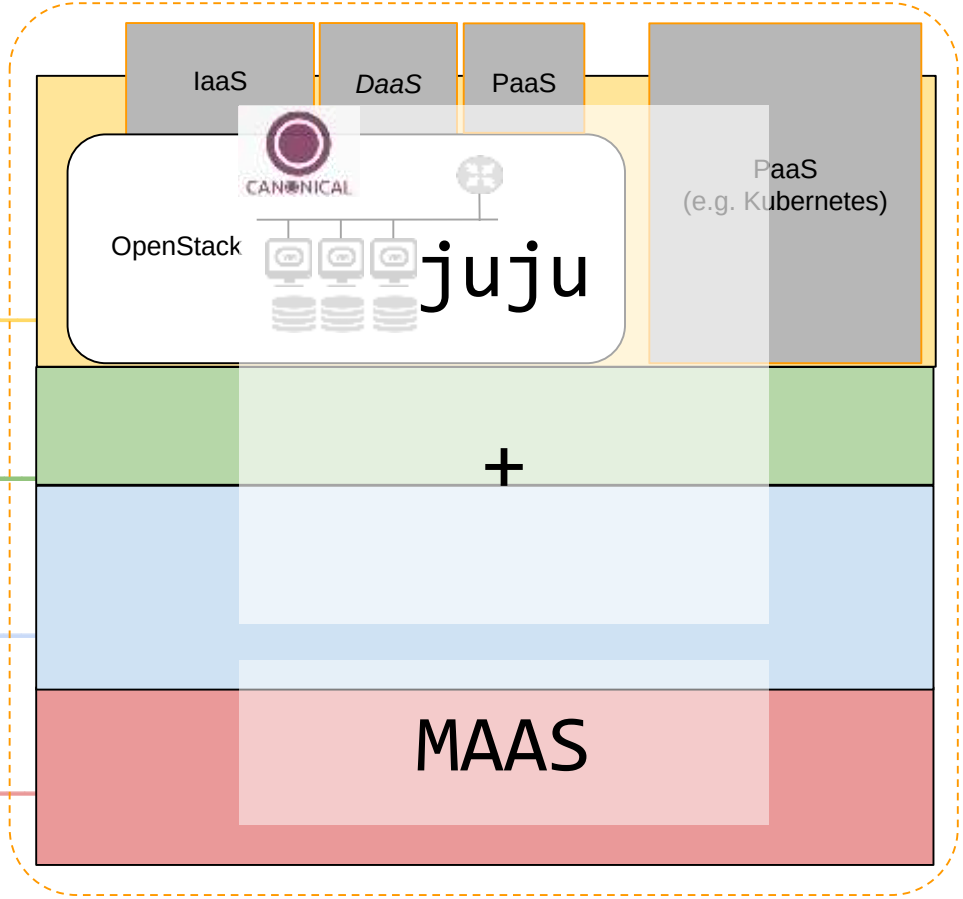
1. Application Services
2. Infrastructure *Virtualization*
3. Operating System
4. Physical resources

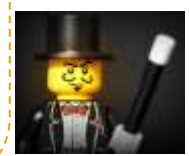


# 4 Layers recipe:

1. Application Services
2. Infrastructure *Virtualization*
3. Operating System
4. Physical resources



-Operating system: **Ubuntu** Xenial

-4 NIC - link aggregation **40Gbps**

**-vlan** + **Bridge** linux

**-blade** is the **GW** for **LXC**

**-iptables**:

-fwd, nat + security LXC

-blade interchangeable from p.v. LXC

**MAAS + juju ***

**LinuXContainer** (LXC with LXD managment)

-opensource

-modern kernel support

-Efficient resource usage (compared to VM)

-Near Bare Metal runtime performance

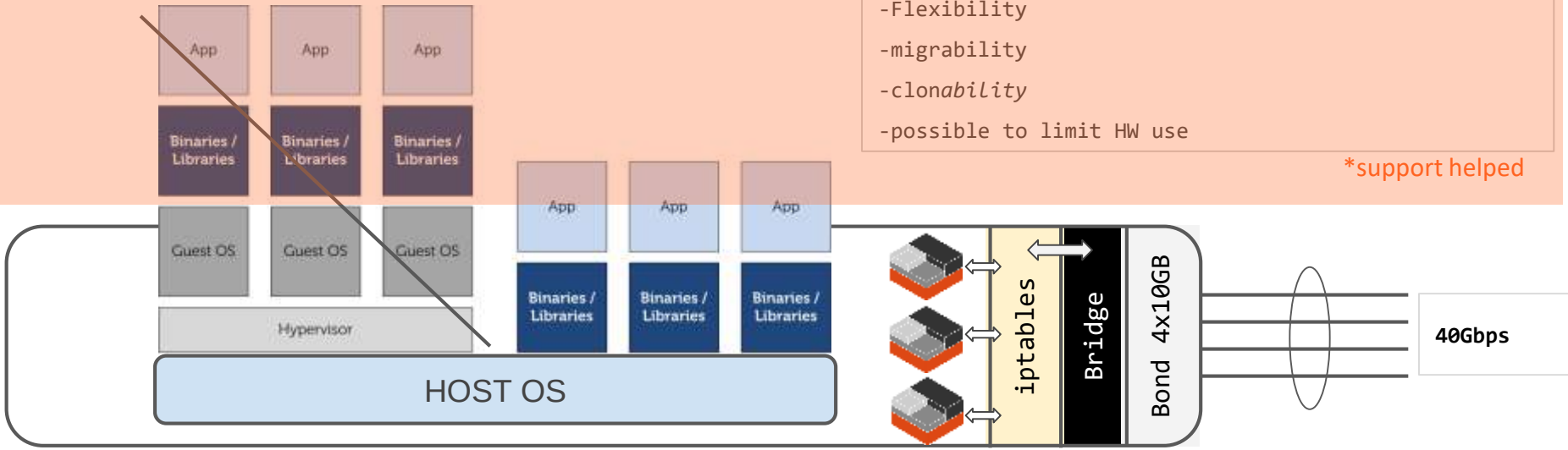
-Flexibility

-migrability

-clon*ability*

-possible to limit HW use

*support helped



# 4 Layers recipe:

1. Application Services
2. Infrastructure *Virtualization*
3. Operating System
4. Physical resources

IaaS | *DaaS* | PaaS | PaaS (e.g. Kubernetes)

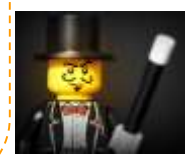
CANONICAL

OpenStack

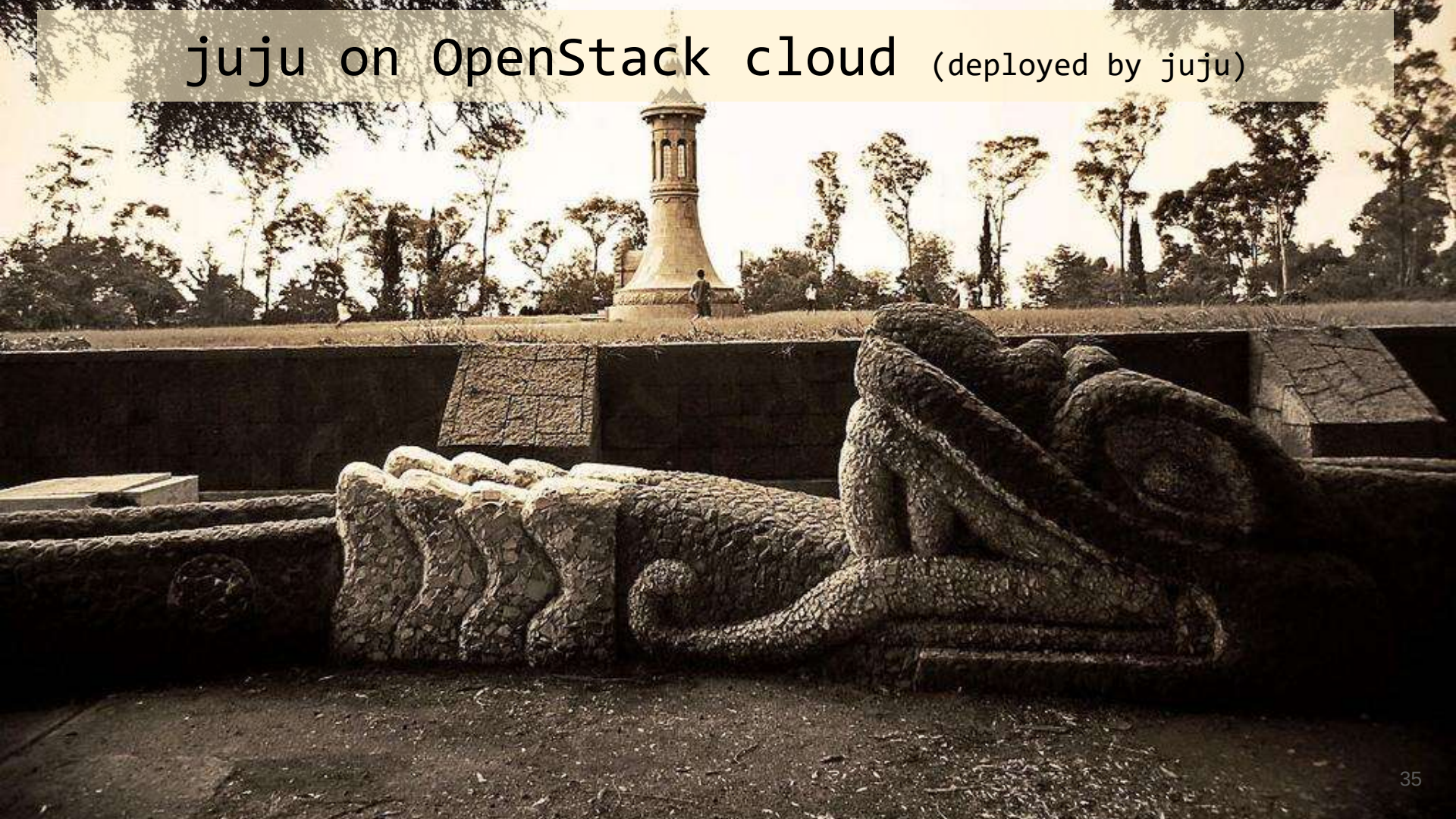
juju

+

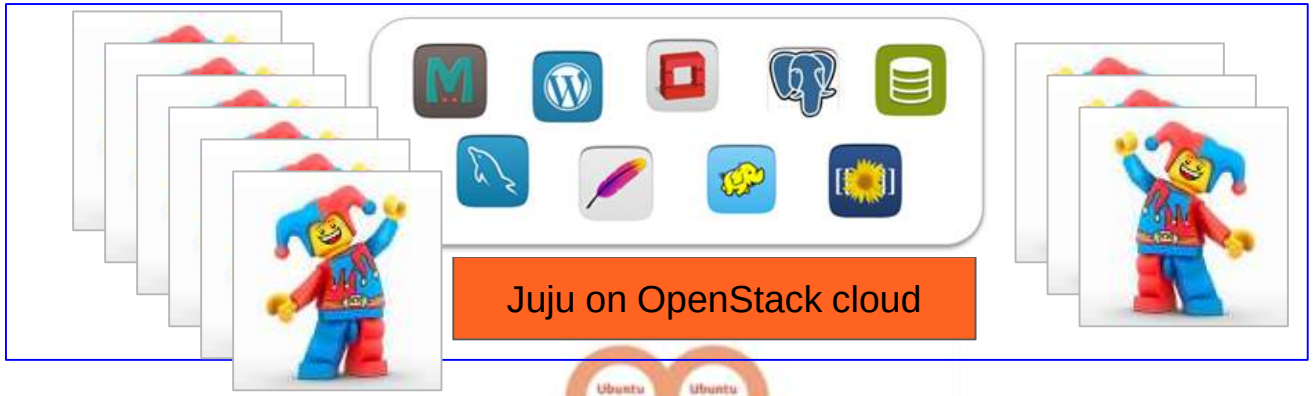
MAAS

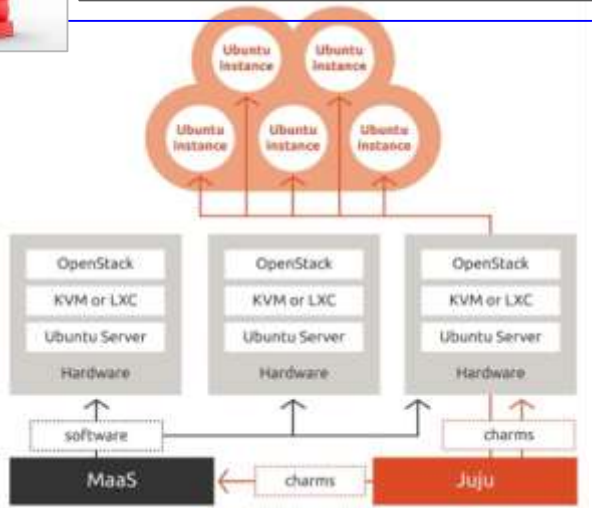


# juju on OpenStack cloud (deployed by juju)



# More flexible than a PAAS, easier than a IAAS




=

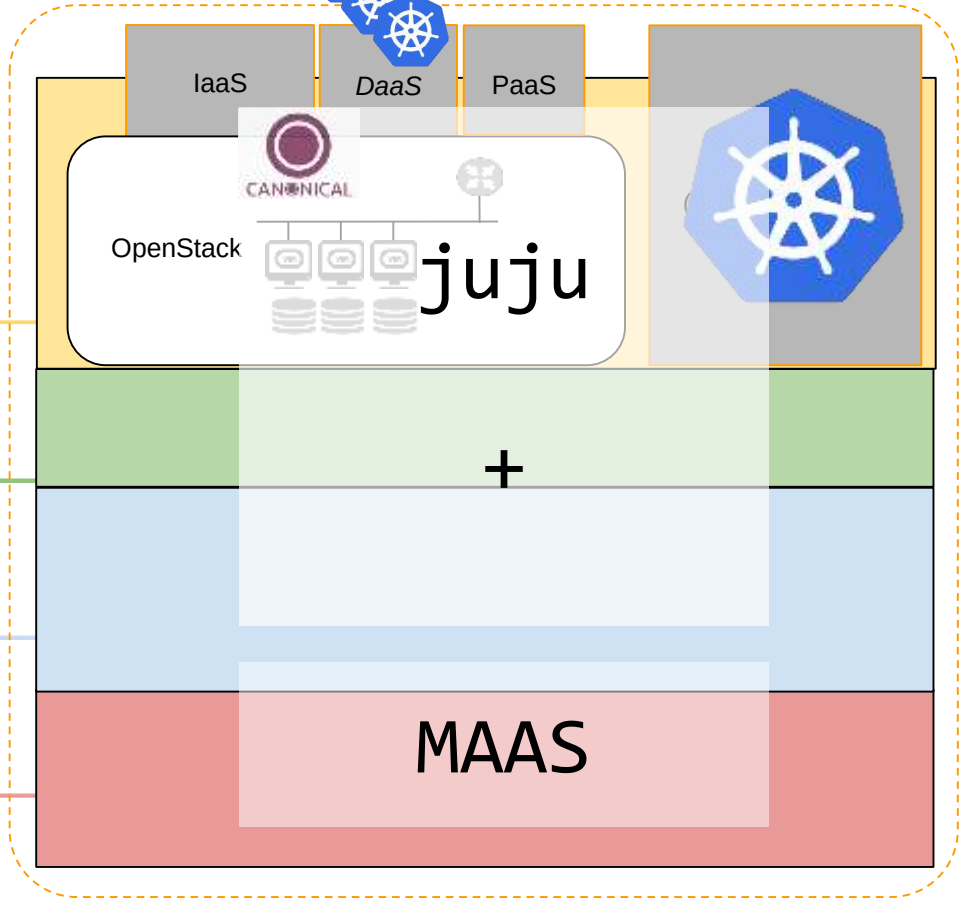
GARR Cloud

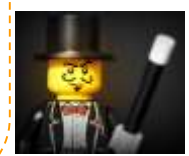
DaaS :)

# 4 Layers recipe:

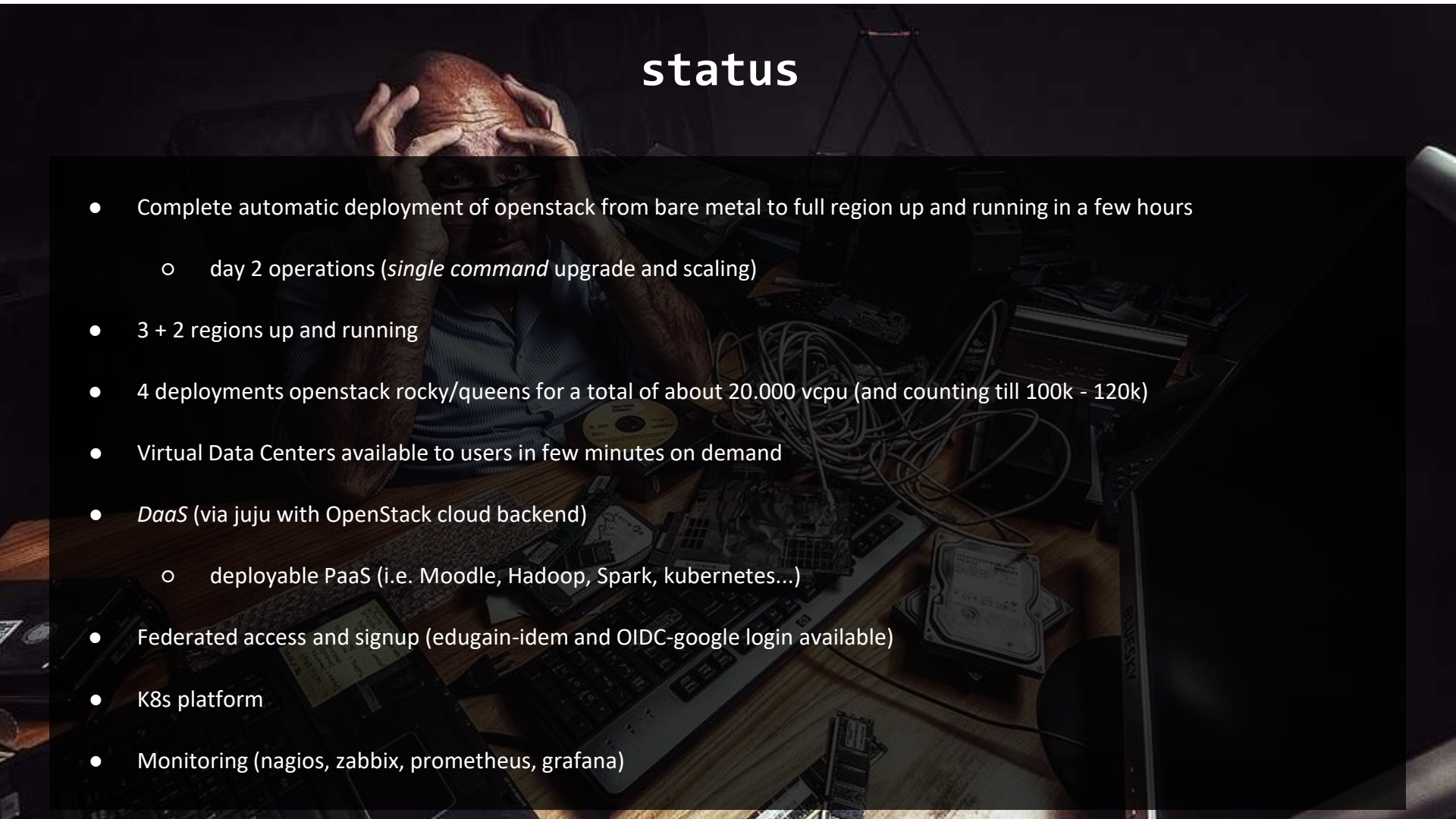
1. Application Services
2. Infrastructure *Virtualization*
3. Operating System
4. Physical resources





# status

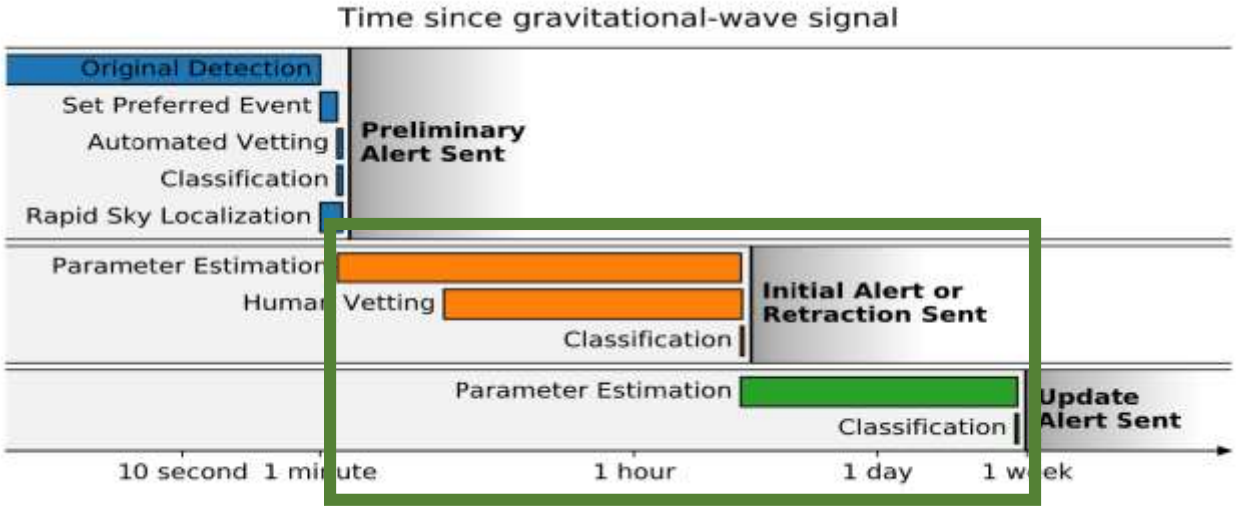


- Complete automatic deployment of openstack from bare metal to full region up and running in a few hours
  - day 2 operations (*single command* upgrade and scaling)
- 3 + 2 regions up and running
- 4 deployments openstack rocky/queens for a total of about 20.000 vcpu (and counting till 100k - 120k)
- Virtual Data Centers available to users in few minutes on demand
- *DaaS* (via juju with OpenStack cloud backend)
  - deployable PaaS (i.e. Moodle, Hadoop, Spark, kubernetes...)
- Federated access and signup (edugain-idem and OIDC-google login available)
- K8s platform
- Monitoring (nagios, zabbix, prometheus, grafana)

Consortium GARR
THE ITALIAN EDUCATION & RESEARCH NETWORK

# GARR Cloud Use Cases

# Virgo - H2020 Asterics - Gravitational Wave alert system



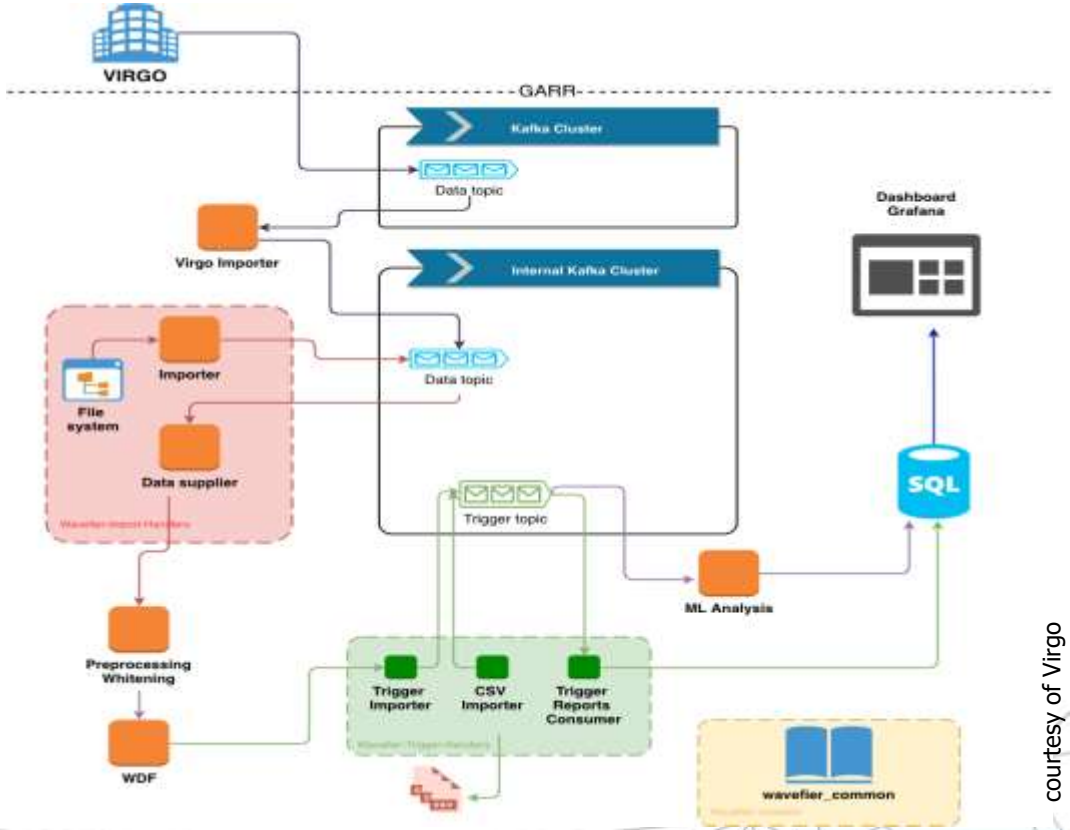

https://emfollow.docs.ligo.org/userguide/index.html

courtesy of Virgo

# Virgo - H2020 Asterics - Wavefier
# Gravitational Wave Detection over the GARR Container Platform


Consortium GARR

Wavefier:
real time pipeline for the detection of transient signals and their automatic classification based on big data / machine learning




courtesy of Virgo

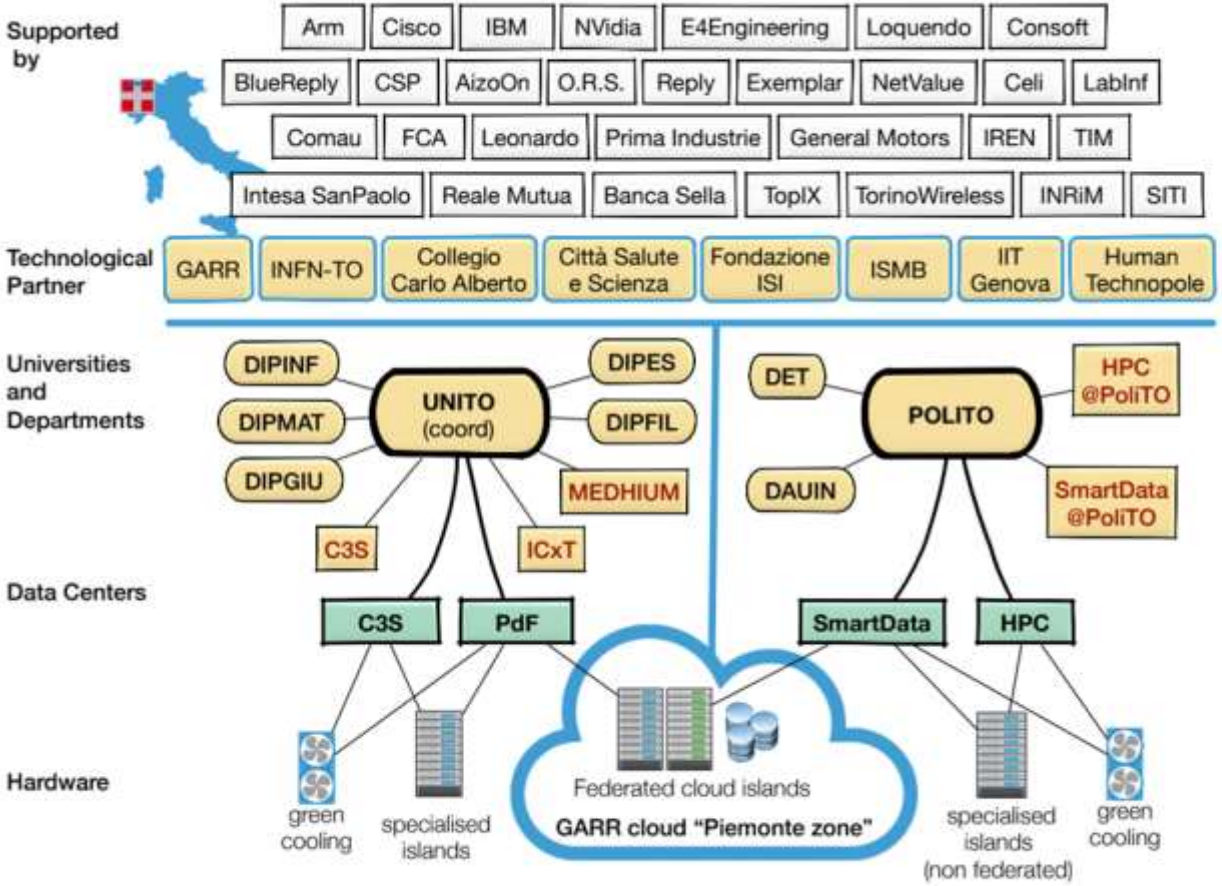


Consortium GARR THE ITALIAN EDUCATION & RESEARCH NETWORK

# INGV - Seismic e-Infrastructure

- Seismic measurement, alerting and analysis infrastructure
- Federated cloud architecture, based on the GARR Cloud model
  - Work in progress

# Università di Torino




HPC4AI

AI-in-demand platform

## Facts

- INFRA-P call Nov. 2017
- Ranked 1st on ~30 submitted projects
- Kick-off mid apr 2018
- 4.5M€ funding
- 2 partners
- 8 associated partners
- Coord. M. Aldinucci
- Many industrial stakeholders

courtesy of UniTo

# Politecnico di Torino - GARR Cloud OpenStack Region

- One of the first GARR Cloud federated regions
- Used for:
  - IaaS - user self service
  - Cloud native application development space
  - Experiment cross-region highly available services
  - Data backup
    - right now 60TB/day of deduplicated and encrypted data


1,500+
CPU cores

400 TB
Storage

9 TB
RAM

ceph

openstack.

courtesy of PoliTo

# ISTI CNR - D4Science + OpenAire

- D4Science.org
  - Integrated technologies that provide elastic access and usage of data and data-management capabilities
    - Virtual Research Environments (VRE)
    - Data discovery, accessing, analysis, and transformation
- OpenAIRE
  - A pan-European (and global) network for Open Science
- Infrastructure:
  - 500+ VMs, ~100 of them hosted by the GARR cloud
  - 75% of them are production services (they must work)
  - Migration, in 2018, from Xen/Aoe to OpenStack/Ceph

courtesy of ISTI-CNR

# Università degli Studi di Salerno - HPC Bioinformatics

- GARR Cloud used for alignment-free sequence comparison
  - Map/Reduce
    - Spark
    - Hadoop
- GARR Cloud credited in scientific publications

Ferraro Petrillo U., Roscigno G., Cattaneo G., Giancarlo R. "*FASTdoop: a versatile and efficient library for the input of FASTA and FASTQ files for MapReduce Hadoop bioinformatics applications.*", Bioinformatics. 2017 May 15;33(10):1575-1577. doi: 10.1093/bioinformatics/btx010.

Ferraro Petrillo U., Roscigno G., Cattaneo G., Giancarlo R., "*Informational and linguistic analysis of large genomic sequence collections via efficient Hadoop cluster algorithms.*", Bioinformatics. 2018 Jun 1;34(11):1826-1833. doi: 10.1093/bioinformatics/bty018.

Invited Talk, Raffaele Giancarlo, "*Methods, tools & platforms for Personalized Medicine in the Big Data Era*", NETTAB 2017 Workshop, Palermo, October 16-18, 2017

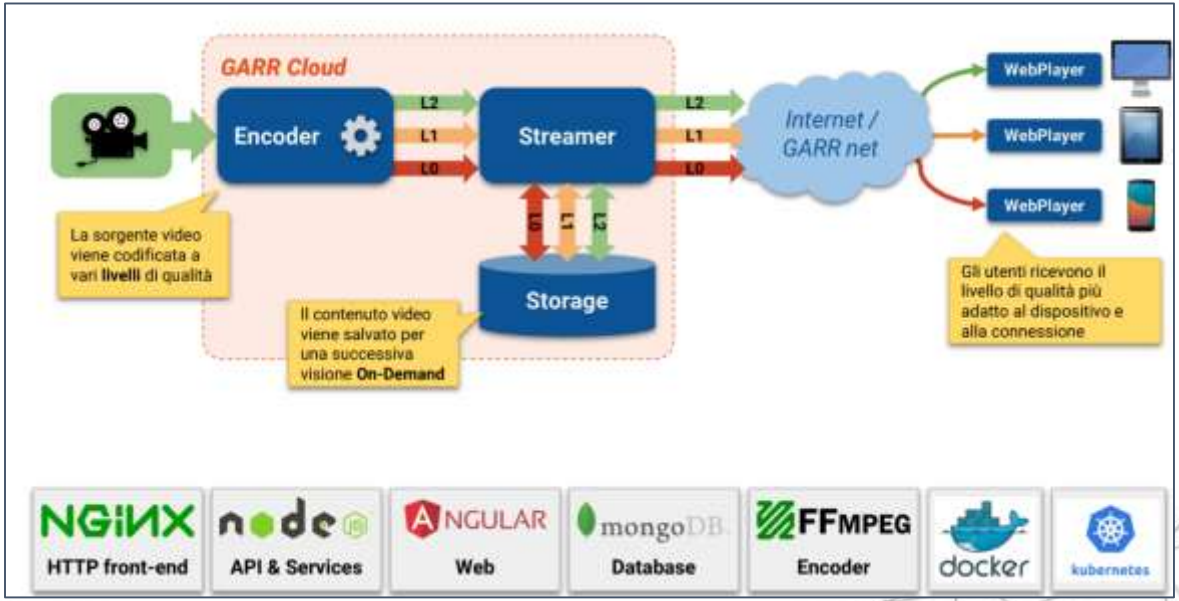
# IEO - Bioinformatics

- container-based bioinformatic research applications
  - reproducible results
- GARR Cloud advantages
  - different research groups can access the same infrastructure
  - dedicated network access/VLANs
  - data doesn’t need to cross different networks

courtesy of IEO

# GARR TV

- Video streaming platform - https://www.garr.tv
  - adaptive video encoding
  - deployed with Kubernetes

# GARR IdP in the Cloud

- Federated identity as a service
  - IDEM and eduGAIN compliant
  - Targeted to small research and education organizations
  - Manage user digital identities without the need to manage also the technological aspects

# GARR Mirror

GARR MIRROR

- Online since 2001
- 200 TB in two network locations
- Backup in the GARR Cloud

# EAPConnect

- you tell us :)

Consortium GARR
THE ITALIAN EDUCATION & RESEARCH NETWORK

# GARR federation

alex.barchiesi@garr.it - Rome 2019


# Re-cap of GARR mission and infrastructure

- [...] fornire servizi per favorire **l’armonizzazione, l’implementazione e la gestione delle e-Infrastructure a vantaggio della comunità scientifica e accademica** nazionale; [...]
- [...] sostenere e **stimolare lo sviluppo di strumenti atti a facilitare l’accesso** alle risorse di calcolo, supercalcolo e storage a livello nazionale ed internazionale, **fornendo gli opportuni metodi, interventi e funzionalità** necessari a mantenere le e-Infrastructure ai livelli degli standard internazionali; [...]
- [...] svolgere le connesse attività di ricerca tecnologica, sperimentazione, trasferimento tecnologico e **formazione del personale**. [...]

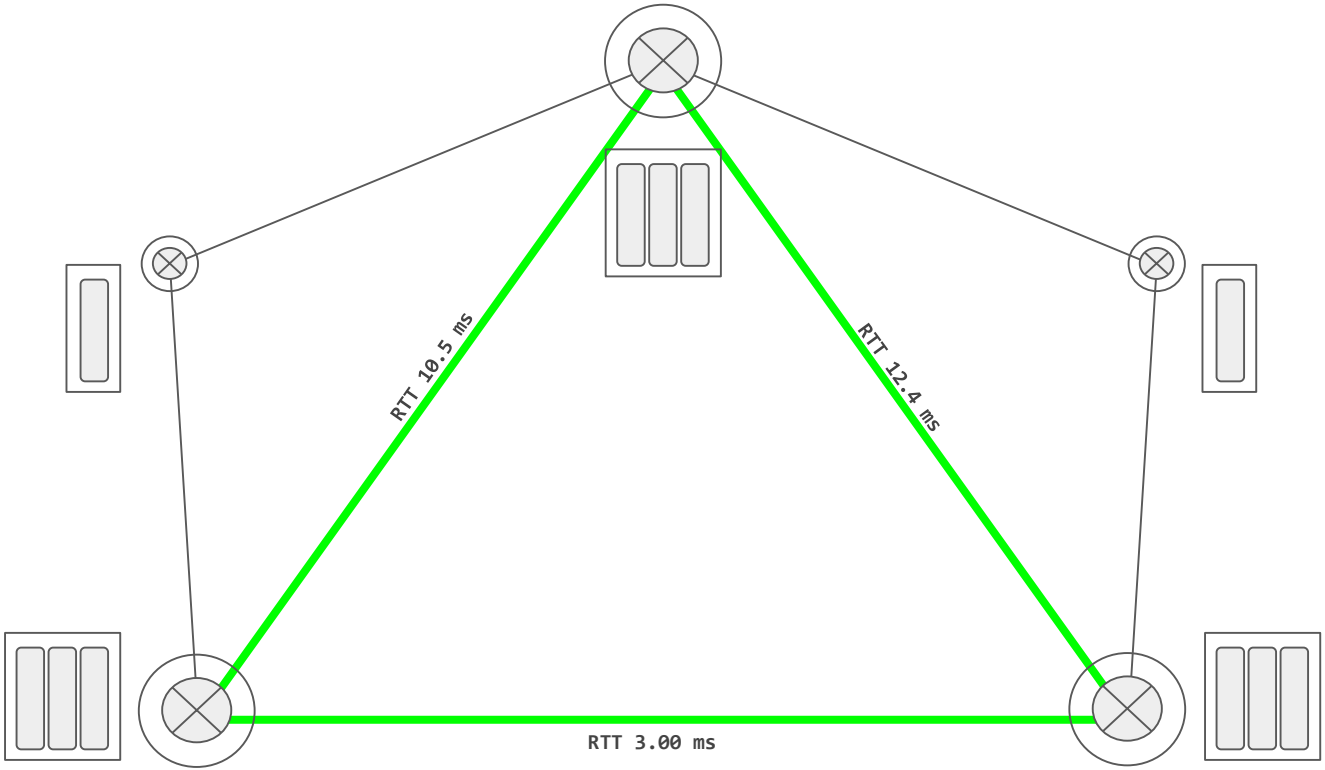
- **Who:**
  - no profit association: **CNR, ENEA, INFN,** Fondazione **CRUI** (universities)
- **Role**:
  - resource **aggregator** (federation)
- **Goals**:
  - **simplify** provisioning of storage and computing
  - serve **different** organizations
  - **Simplify** experience (empower administrators)

# 8500 core
# 10 PB

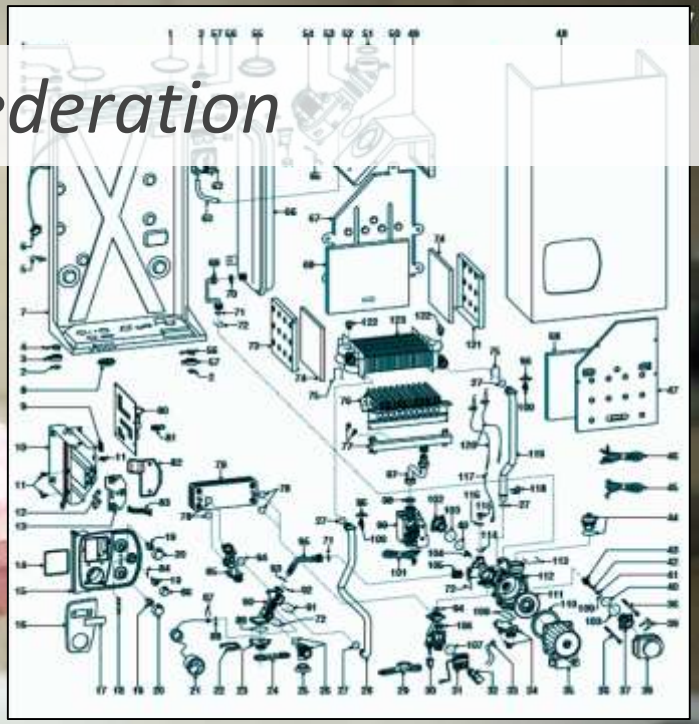
... 11 rack/CSD-modules

# Network

40Gbps
10Gbps



# our concept of *Federation*



- the simplest (for federated region admin) the better
  - the less requirements the more inclusive

BETTY GILLIS

# multi-region/multi-domain model

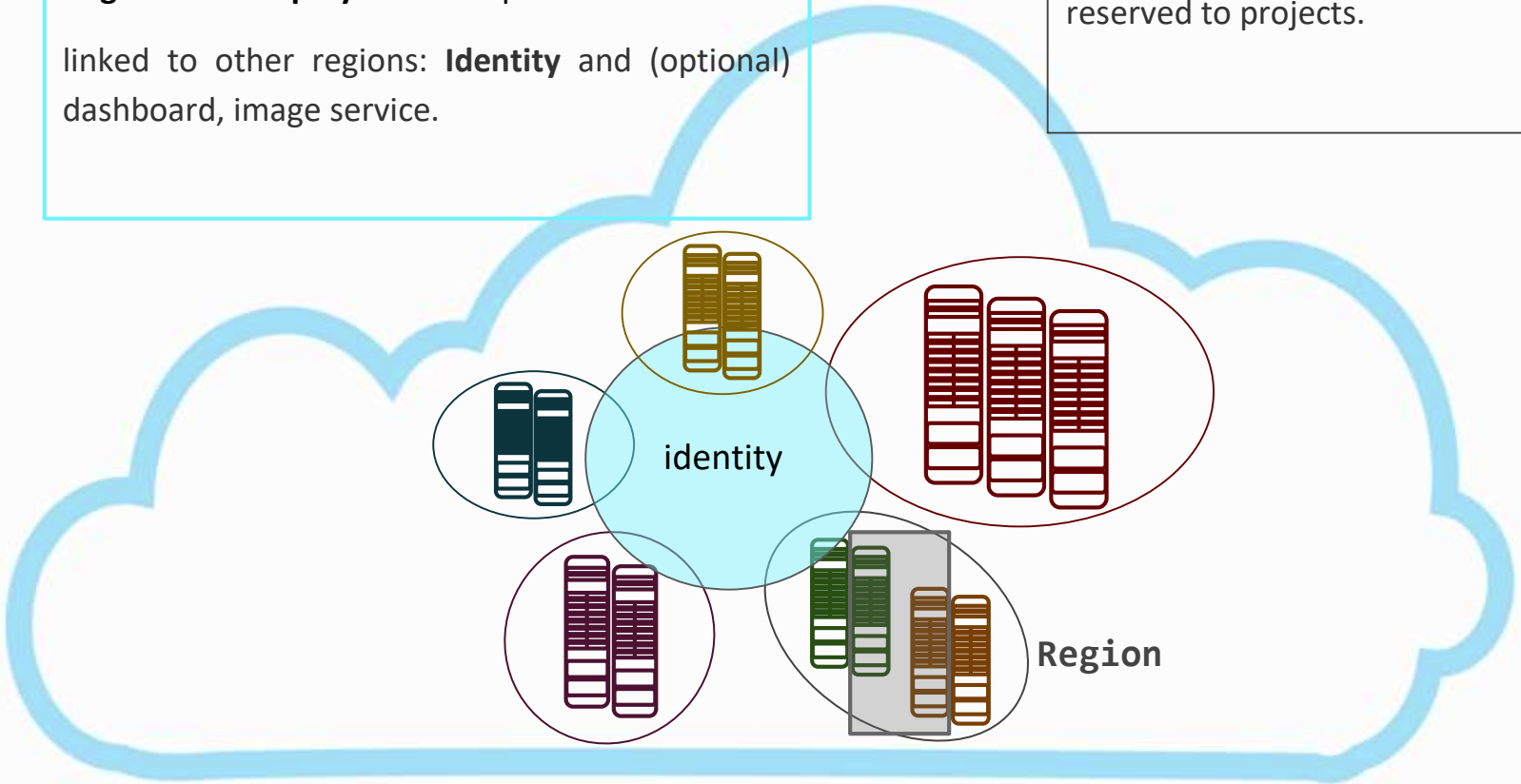
**Region: own deployment** of OpenStack

linked to other regions: **Identity** and (optional) dashboard, image service.

Inside a **Region**: advanced scheduling

**Availability Zone:**

nodes can be logically grouped into AZ and reserved to projects.



# joining the Federation

Procedure of inclusion

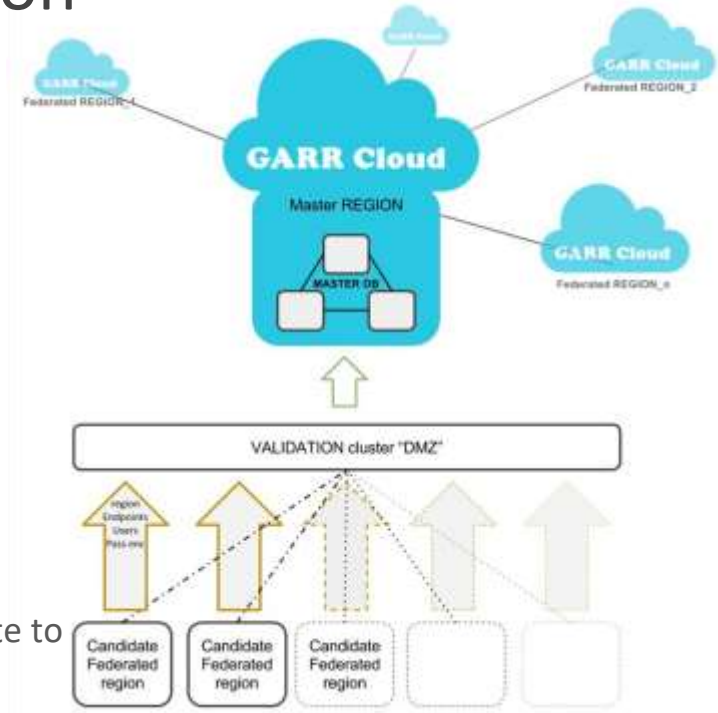
- Bundle OpenStack - attaches to validation cluster
- Validation in “DMZ” cluster
- **No cleartext credentials exchange**

**for research institutes:**

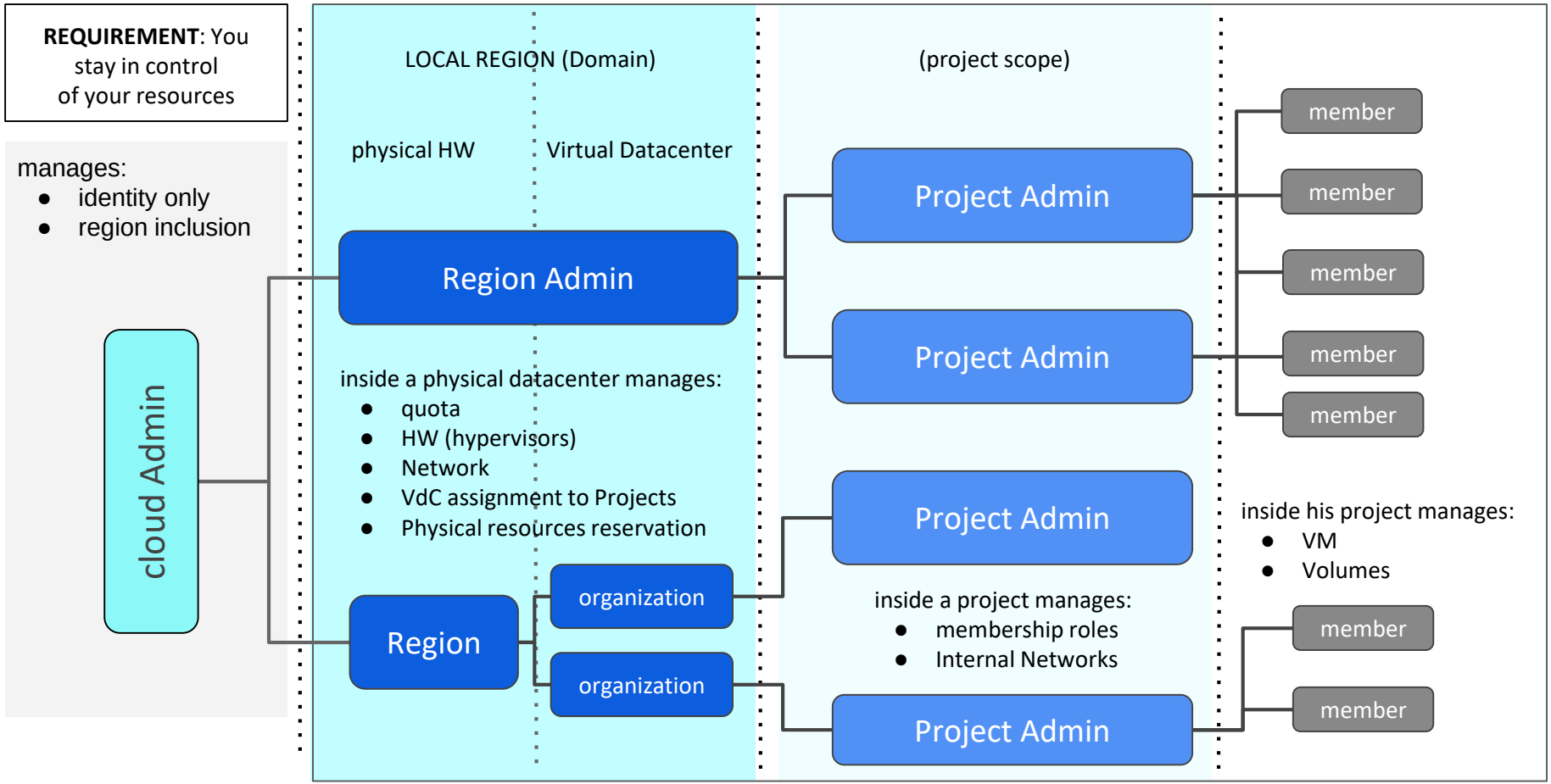
1. You own HW, but have no manpower/knowledge (yet)
2. You already have an OpenStack deployment (recent one)
3. None of the previous, but you have men-power

but for everyone...**everything is publicly available** (see next) and we contribute to community projects:

- Horizon
- K8s-keystone-auth
- juju charms: ceph, keystone saml/sso, default gw, moodle...



# Delegation of authority (via domains, policy and metadata filters - ongoing)



# (few) details about deployment *requirements*

# networking prescriptions

1. private network
    1. ipmi (usually untagged)
    2. pxe/boot/management (best practice untagged - simplifies the setup of pxe)
    3. Storage ceph “priv” (to be used by OSDs only)
    4. Storage ceph “pub” (used by MON and Ceph clients, it normally is a private network)
    5. OpenStack data+mgmt

1. public network
    1. Public ip (for infrastructure and Cloud service frontend)

        Needed minimum 40 IP (suggested to have a 3 members HA of each service: /25 subnet)
    1. OpenStack floating ip - (this is basically the number of VM that you foresee to have publicly exposed)

        To be evaluated according to computing resources/use case

**Best practice:**

- link aggregate (bond) all interfaces and set a virtual interface (vlan) for each of the previously mentioned networks, except PXE.
- keep IPMI network separated.
- NB configure ILO/IDRAC with IPMI over LAN

Note: The networking must be configured on the switches/routers while MAAS takes into account the server configuration (ref https://docs.maas.io/2.5/en/installconfig-networking)

# firewall: ports needed

egress: 80, 443, 5000, 35357, 8774, 8776, 8778, 9292, 9696, 6080

ingress (only needed on public network from subnets which need access to OpenStack and NB from GARR-keystone network ): 80, 443, 8774, 8776, 8778, 9292, 9696, 6080, 5000

N.B. open port 5000 to test “local” keystone functionality, after including the region in the federation it can be closed again.

# Automation and deployment: MAAS + juju

**MAAS** (responsible of physical machine deployment - ref: https://maas.io/)

Note MAAS is not a demanding service in terms of CPU, RAM, Disk, Bandwidth.

1 physical machine (2 if in HA):

- LXD host (MAAS + Juju + OpenStack clients)
- Hypervisor host (mainly for VM hosting Juju controller)

container LXC:

- Region controller (needs to reach ipmi network (ipmi network routed towards MAAS)
- Rack controller (may be co-located with Region)
- NAT (Note this is a Gateway for outgoing requests of services with only private networks, e.g. installation/upgrade of DB)

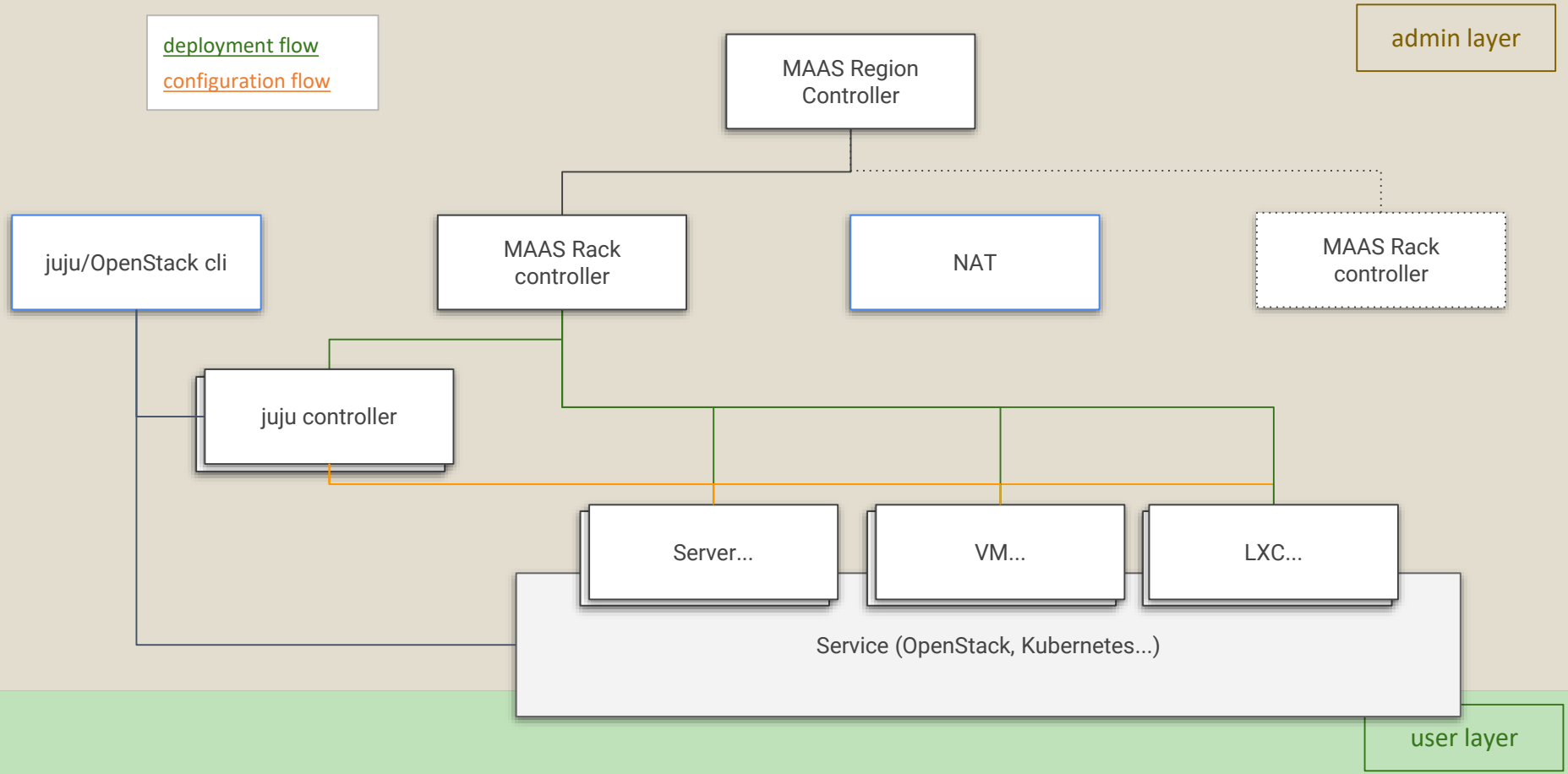
Best practice:

- configure HA on the hosted services
- Link aggregation of network interfaces (bond) + virtual interfaces (vlan)
- You can also install Region and Rack controller on the same LXC
- You can install juju and OpenStack client on the same LXC container (may be also deployed on separate containers)
- NAT configuration: (iptables -t nat -A POSTROUTING -o <NIC_NAME> -j MASQUERADE)

**juju controller** (ref: https://docs.jujucharms.com/2.5/en/)

- will be *bootstrapped* under MAAS supervision as a VM (Best-practice: several VMs on separate servers for HA)

Note juju controller is a key service to deploy and manage any service (i.e. OpenStack)


deployment flow
configuration flow
admin layer
MAAS Region Controller
juju/OpenStack cli
MAAS Rack controller
NAT
MAAS Rack controller
juju controller
Server...
VM...
LXC...
Service (OpenStack, Kubernetes...)
user layer

# federation recipe:

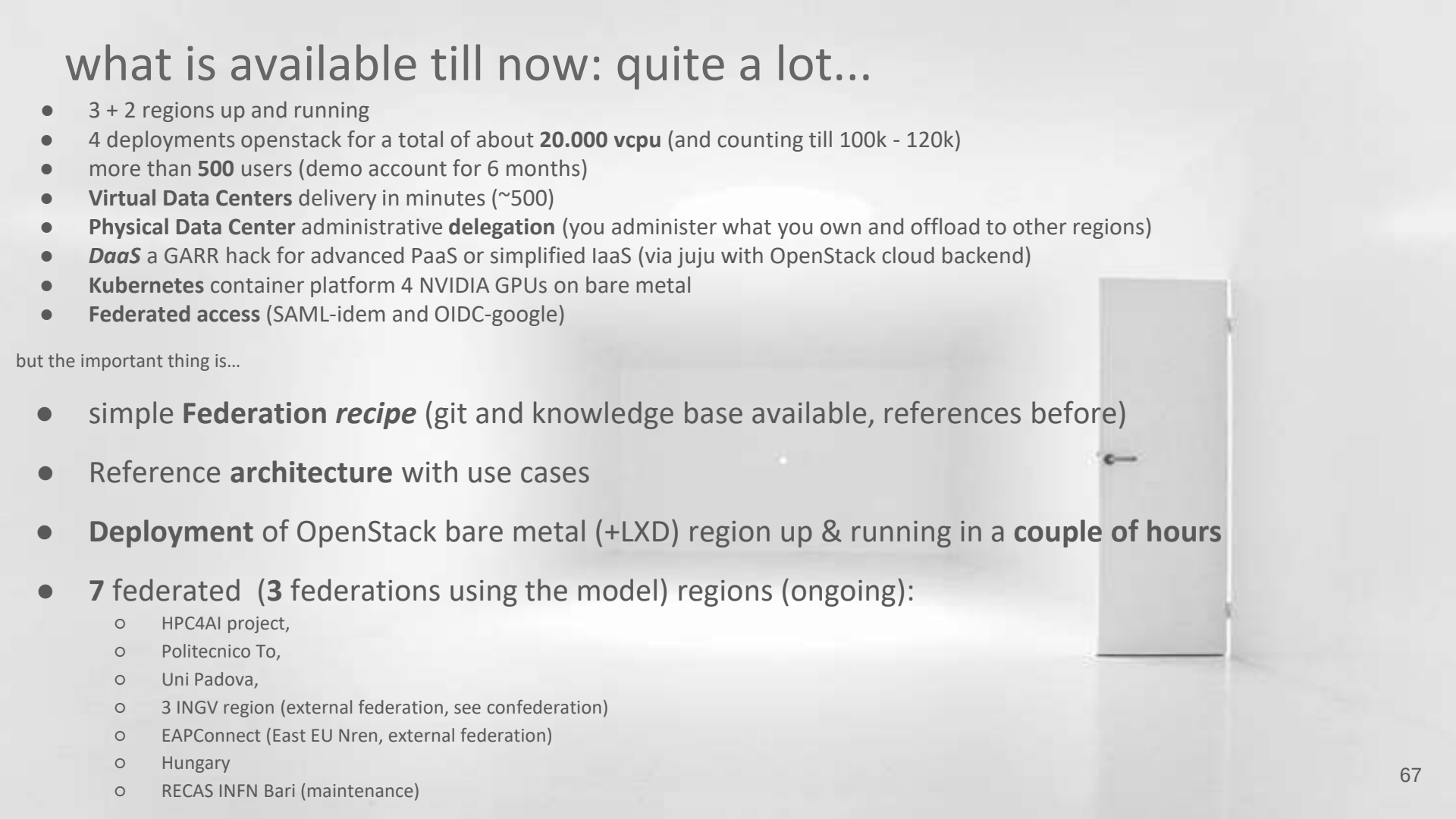
https://cloud.garr.it

https://git.garr.it/cloud/federation

# what is available till now: quite a lot...

- 3 + 2 regions up and running
- 4 deployments openstack for a total of about **20.000 vcpu** (and counting till 100k - 120k)
- more than **500** users (demo account for 6 months)
- **Virtual Data Centers** delivery in minutes (~500)
- **Physical Data Center** administrative **delegation** (you administer what you own and offload to other regions)
- ***DaaS*** a GARR hack for advanced PaaS or simplified IaaS (via juju with OpenStack cloud backend)
- **Kubernetes** container platform 4 NVIDIA GPUs on bare metal
- **Federated access** (SAML-idem and OIDC-google)

but the important thing is…



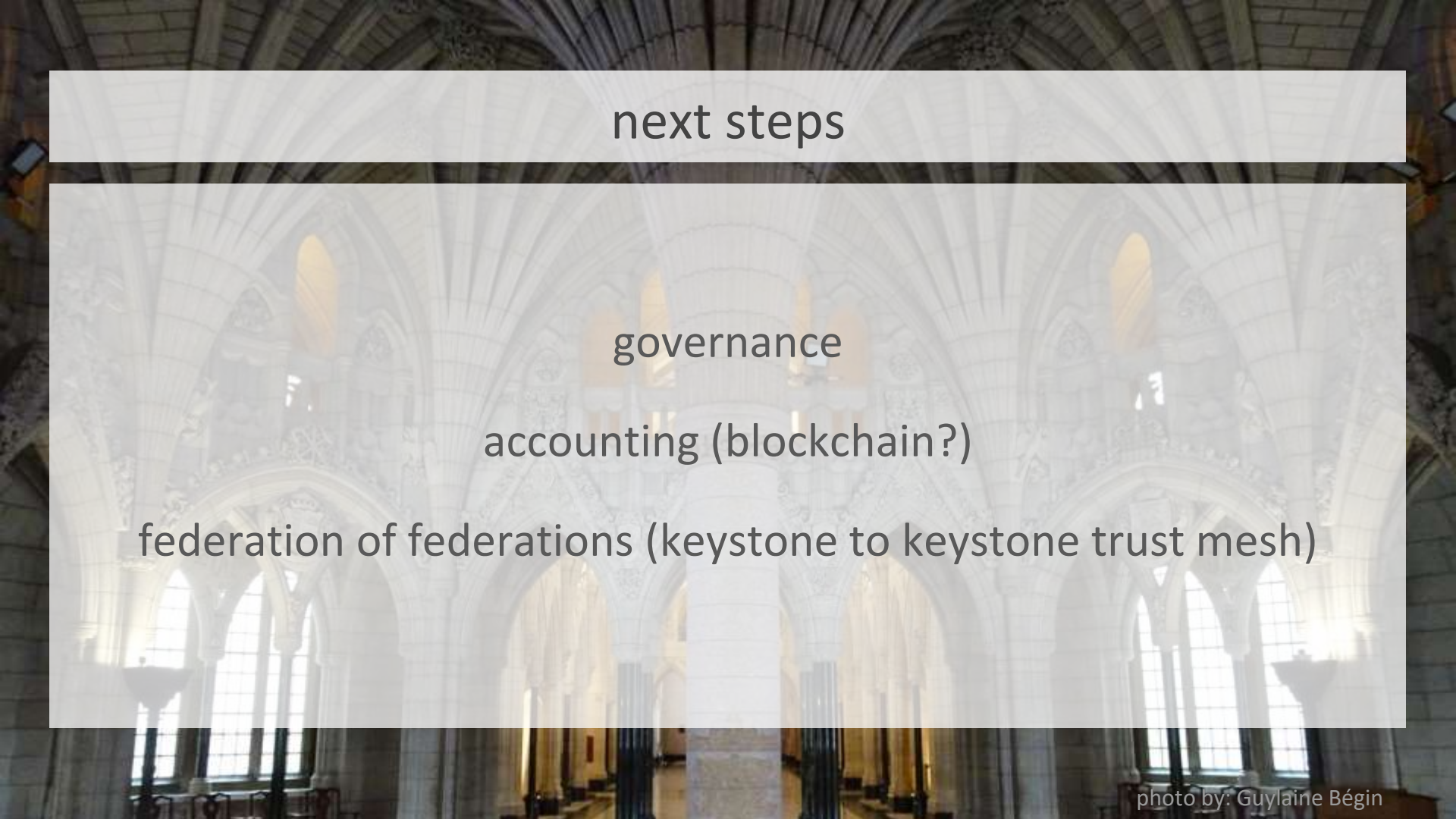
- simple **Federation *recipe*** (git and knowledge base available, references before)
- Reference **architecture** with use cases
- **Deployment** of OpenStack bare metal (+LXD) region up & running in a **couple of hours**
- **7** federated (**3** federations using the model) regions (ongoing):
  - HPC4AI project,
  - Politecnico To,
  - Uni Padova,
  - 3 INGV region (external federation, see confederation)
  - EAPConnect (East EU Nren, external federation)
  - Hungary
  - RECAS INFN Bari (maintenance)

# next steps

governance

accounting (blockchain?)

federation of federations (keystone to keystone trust mesh)



# thanks

interested in a fed-federation working group?
drop us a line: cloud-support@garr.it

photo by: Guylaine Bégin

Consortium GARR
THE ITALIAN EDUCATION & RESEARCH NETWORK

# K8S:under the hood

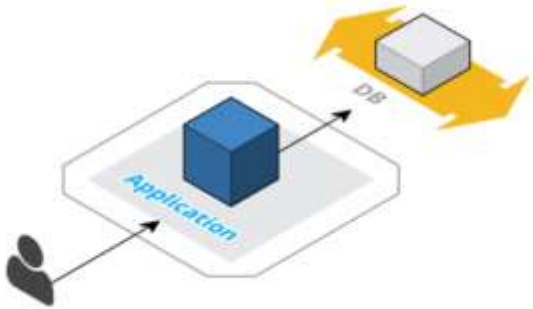
(aka all you wanted to know about k8s but you never dare to ask)

Alex Barchiesi - Cloud CSD department

# container orchestration

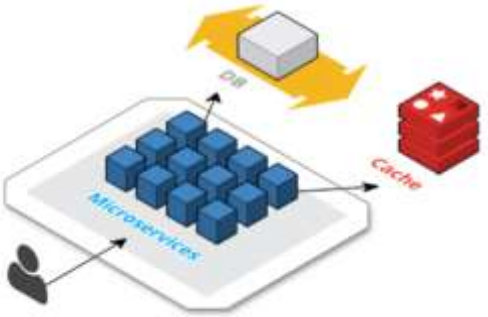
## monolith application:

- eats CPU, RAM
- does not scale
- difficult to maintain




## split into microservice:

- caching layer
- service discovery
- load balancing
- health checks
- storage managment
- auto scaling...



# What do we want from an orchestrator?

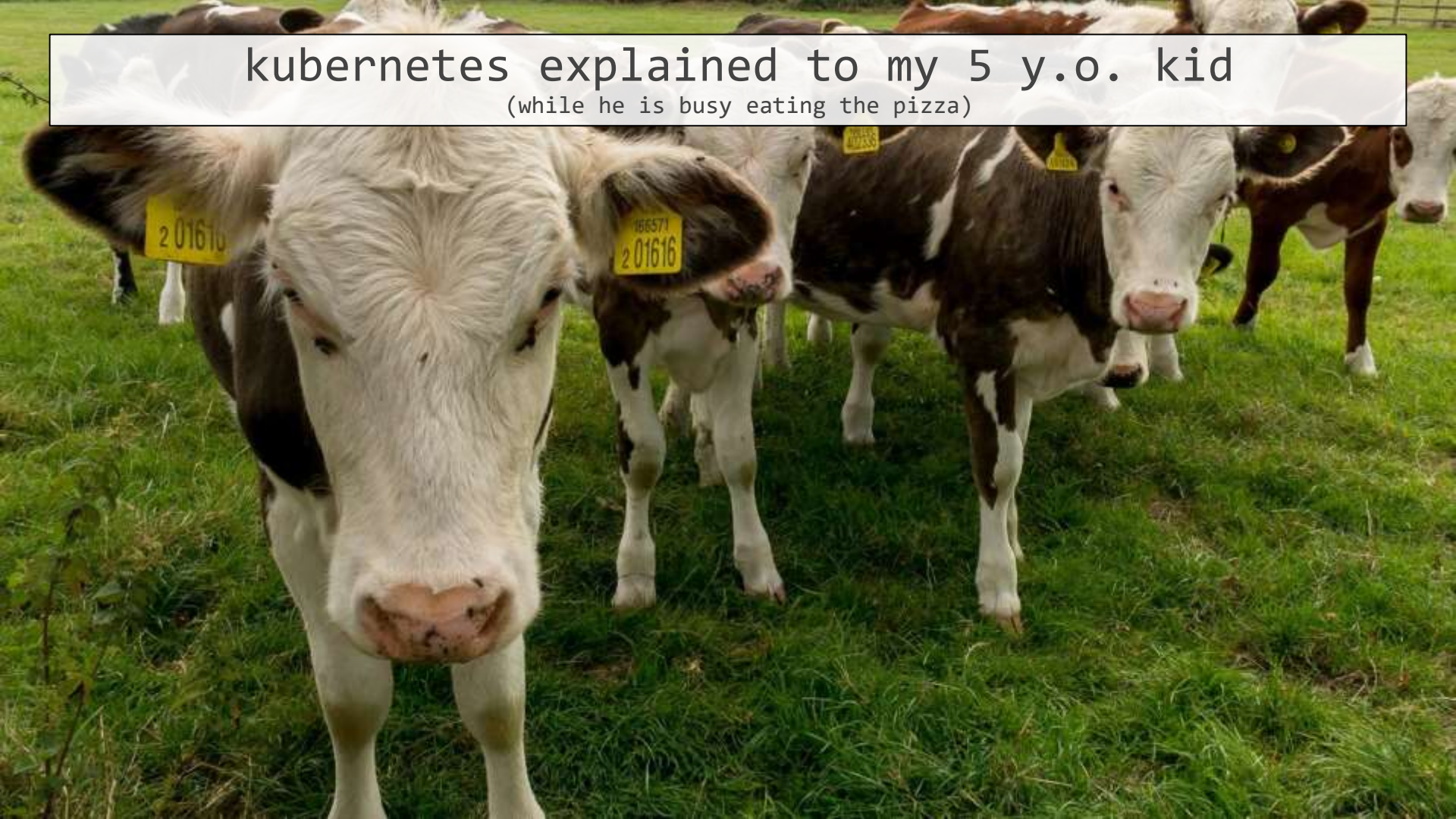
desiderata:

- Replication of components
- Auto-scaling
- Load balancing
- Rolling updates
- Logging across components
- Monitoring and health checking
- Service discovery
- Authentication

solution (most widespread):

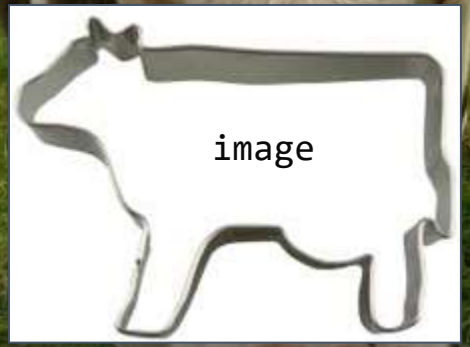
# kubernetes explained to my 5 y.o. kid

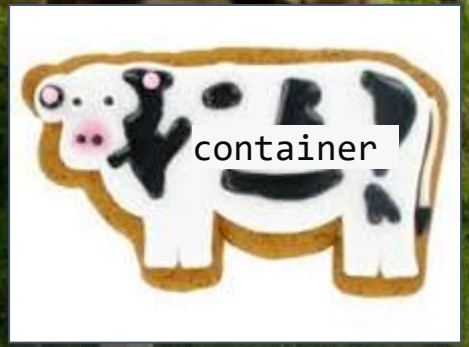
(while he is busy eating the pizza)



# kubernetes explained to my 5 y.o. kid

(while he is busy eating the pizza)



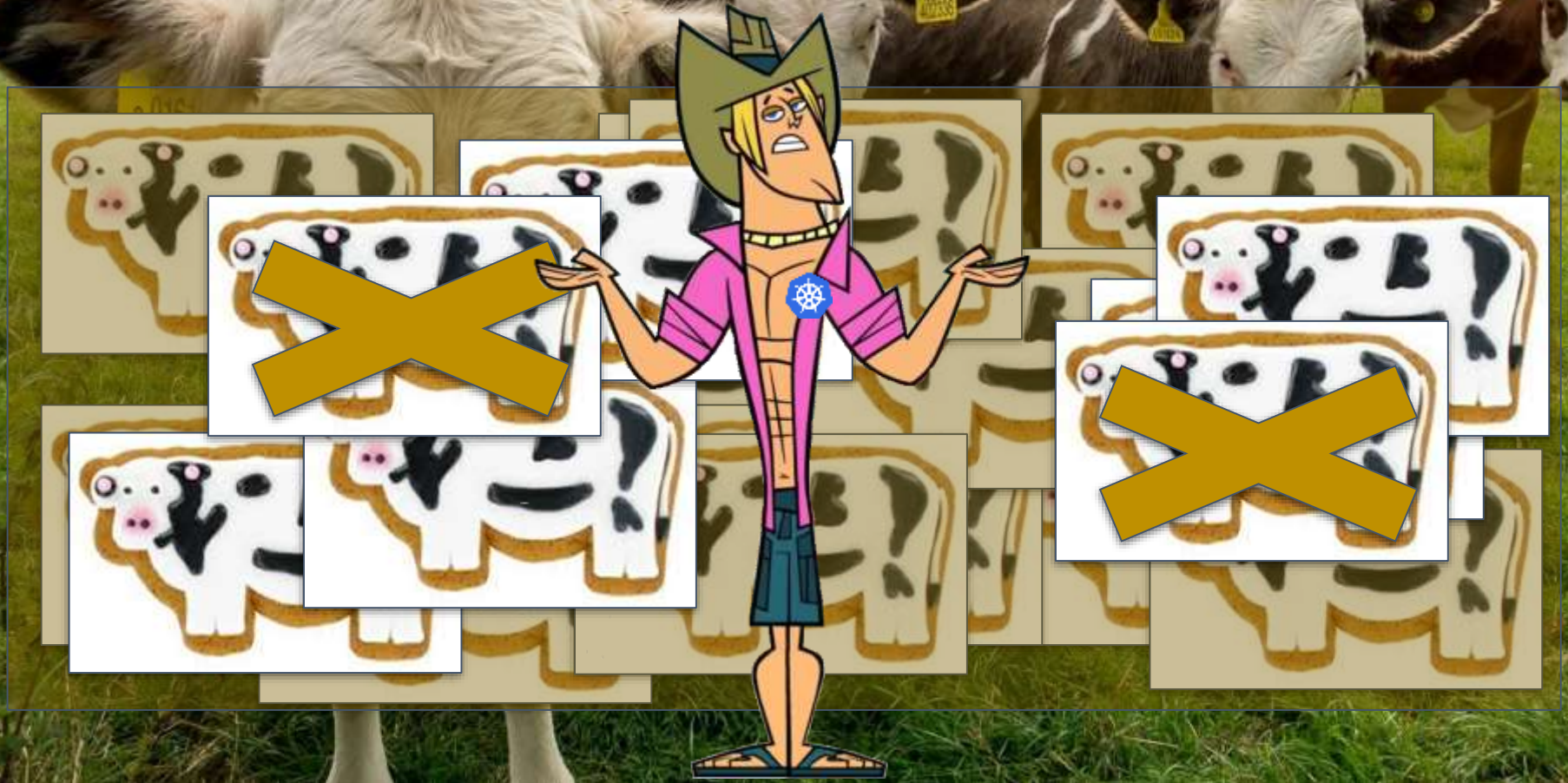



# kubernetes explained to my 5 y.o. kid

(while he is busy eating the pizza)

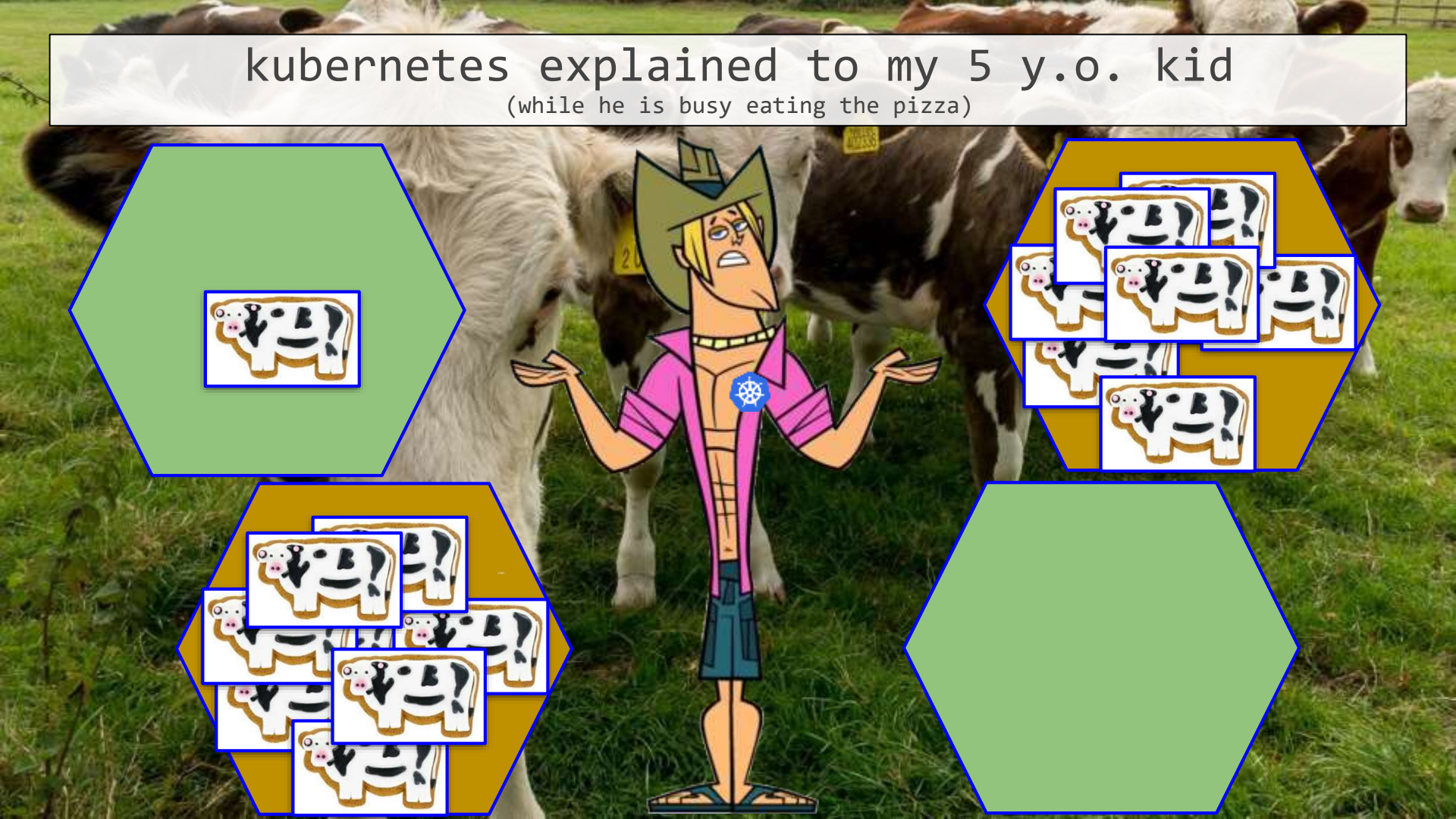
# kubernetes explained to my 5 y.o. kid

(while he is busy eating the pizza)



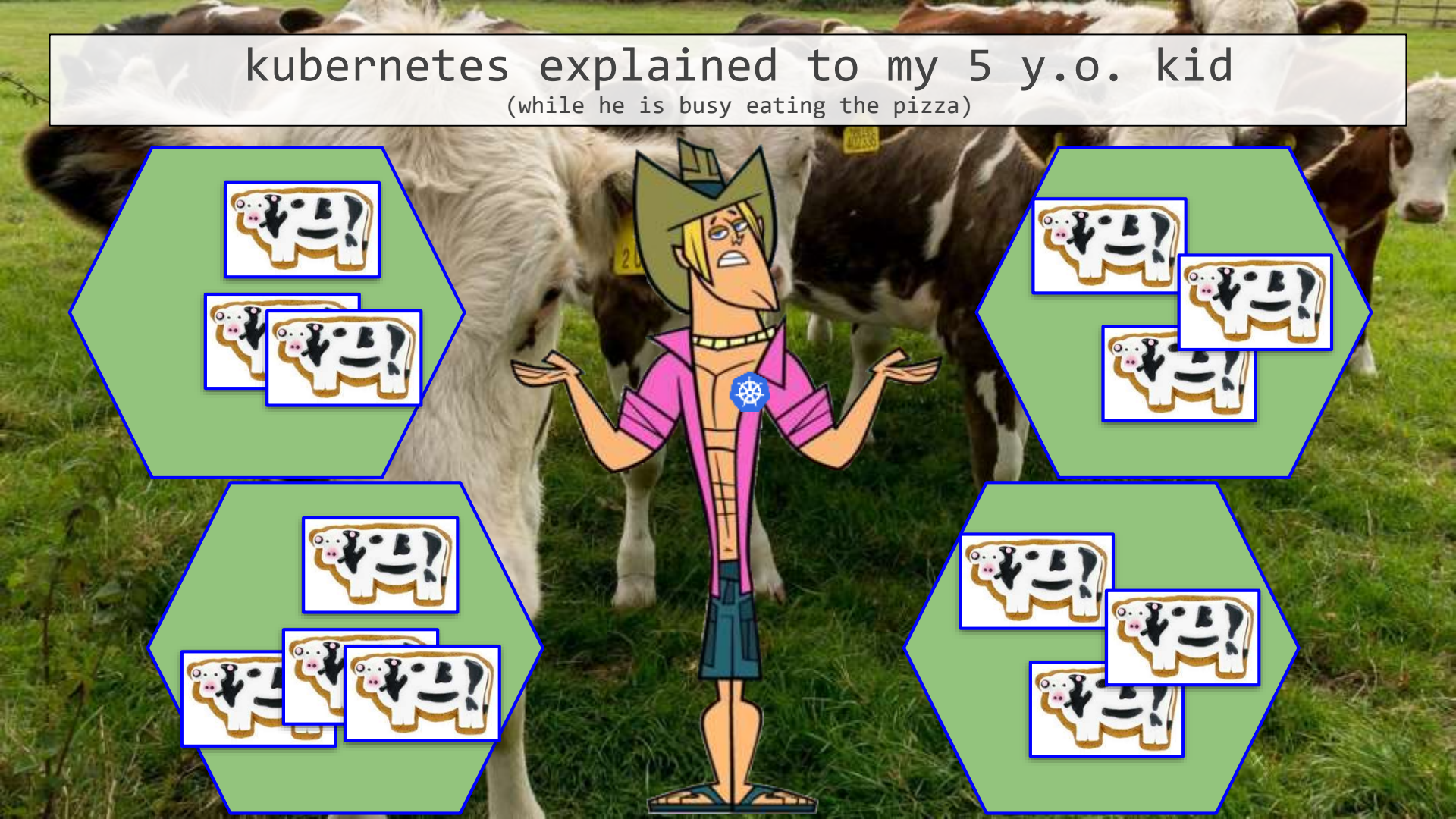
# kubernetes explained to my 5 y.o. kid

(while he is busy eating the pizza)



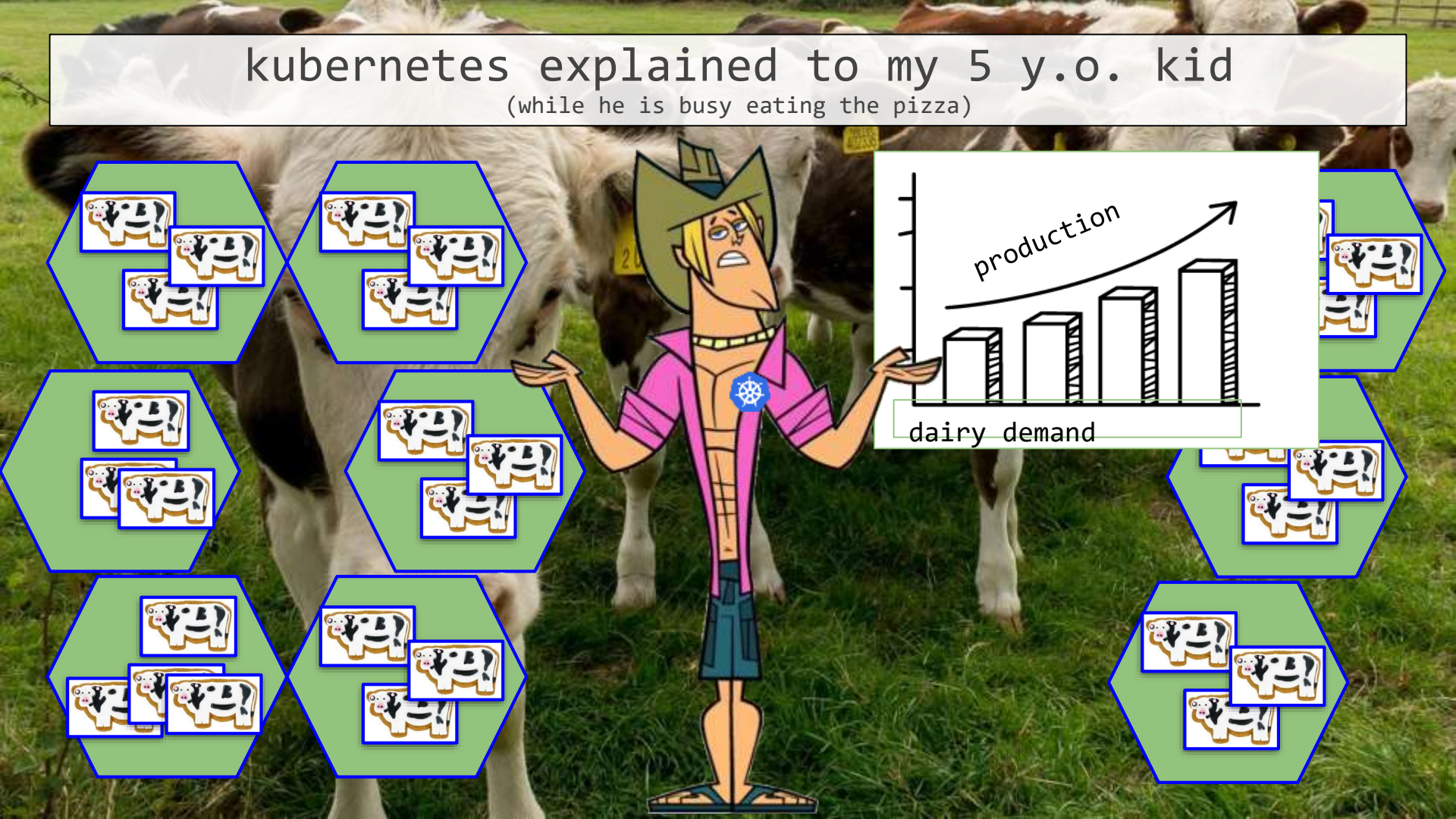
# kubernetes explained to my 5 y.o. kid

(while he is busy eating the pizza)



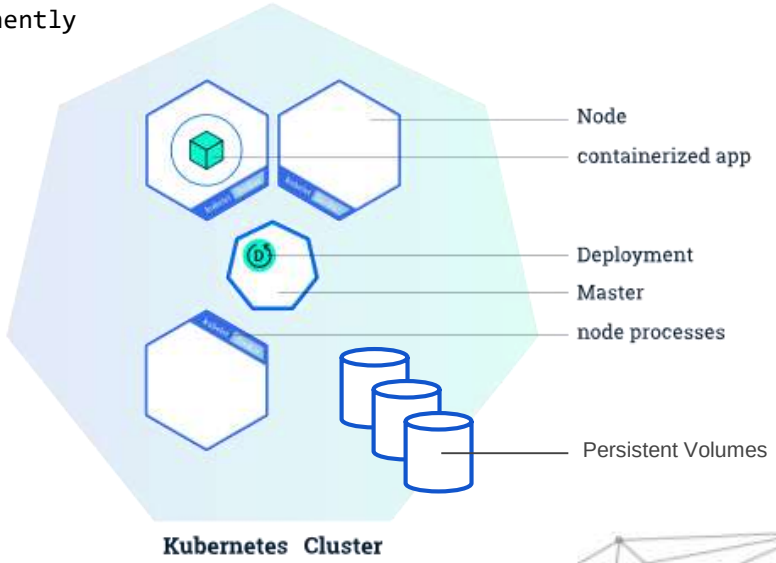
# kubernetes explained to my 5 y.o. kid

(while he is busy eating the pizza)



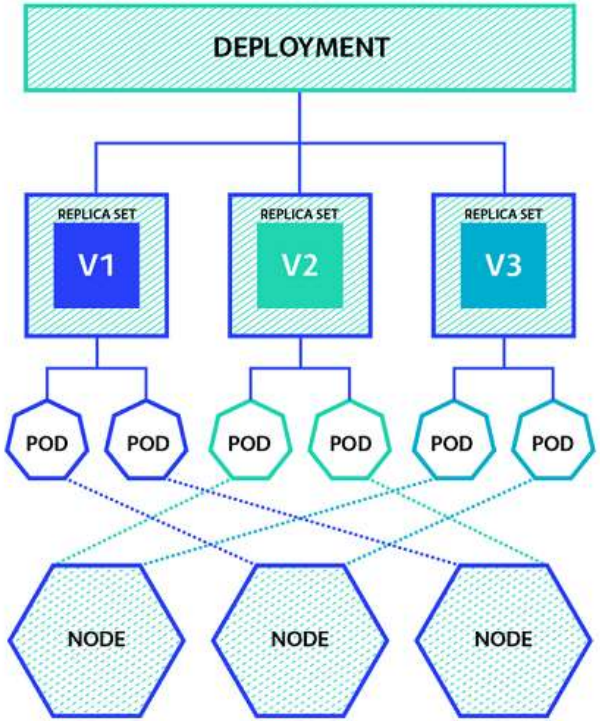
# kubernetes *hardware*

- **nodes**: smallest unit of computing. Single machine. Physical or virtual.
- **cluster:** pool of resources managed together. Handles distribution towards nodes.
- **persistent volumes:** shared storage space to save data permanently

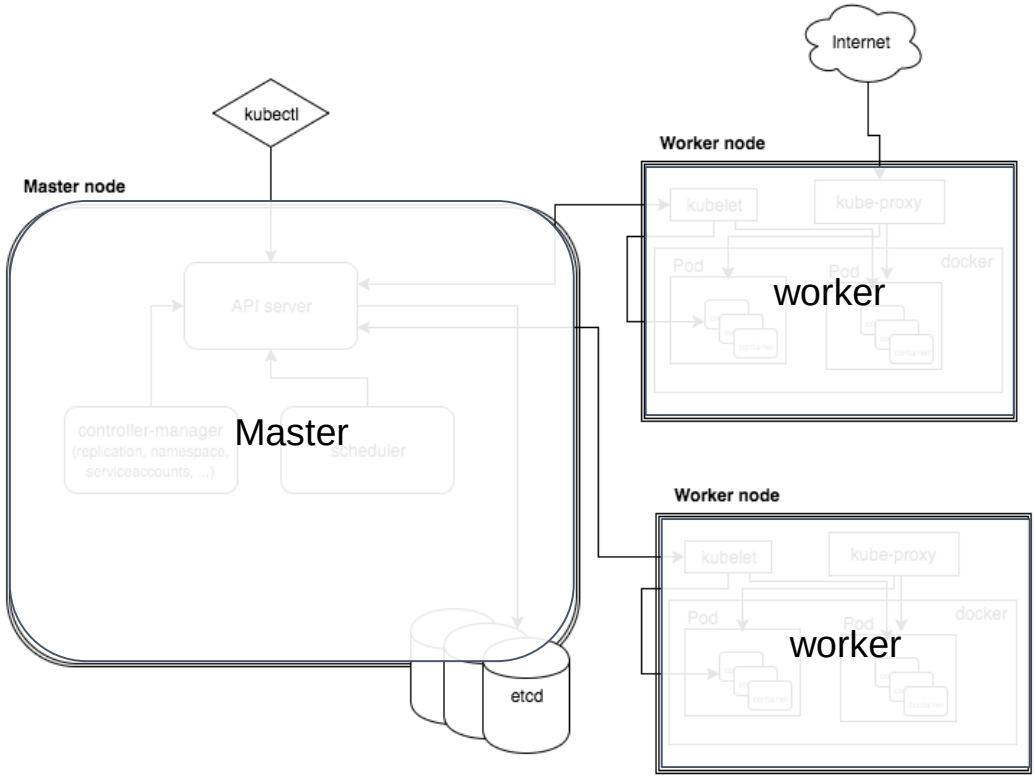


# kubernetes *software*

- **containers**: programs packed as single file and shared on internet.
- **pod:** basic units. Holds multiple container. Replication bb.
- **deployment:** abstraction layer for pods + state (i.e. number of replicas)
- **service**: abstraction for logical set of pods and a policy of access (way to expose an application)

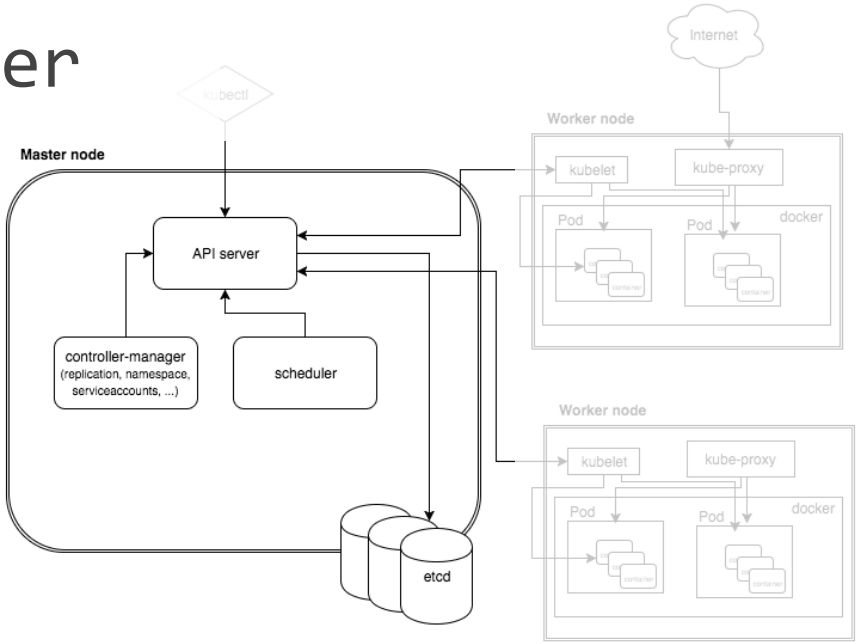


# high-level component architecture



# master

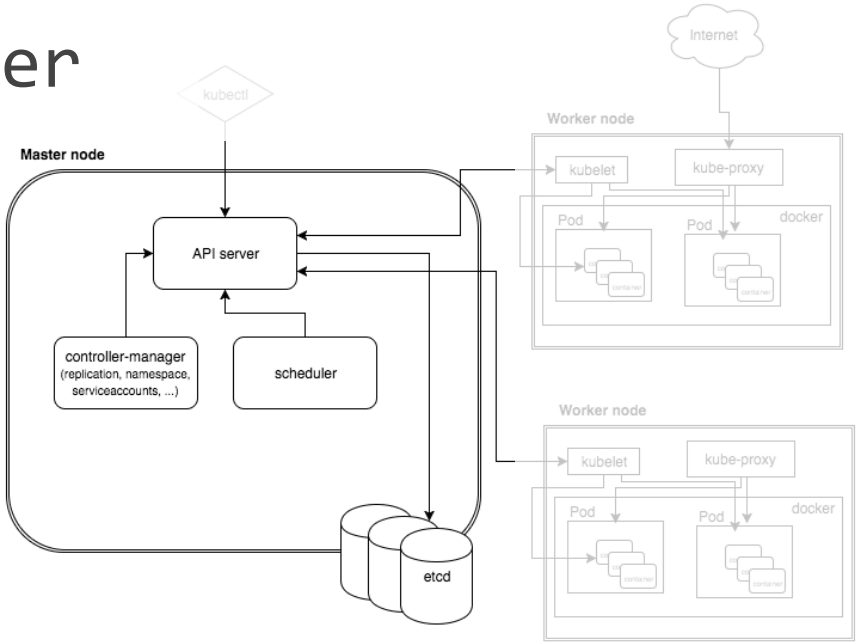
- api server
- etcd storage
- scheduler
- controller-manager




- entry point for all the REST commands used to control the cluster.
- processes the REST requests, validates them, and executes the bound business logic.
- The result state has to be persisted somewhere

# master

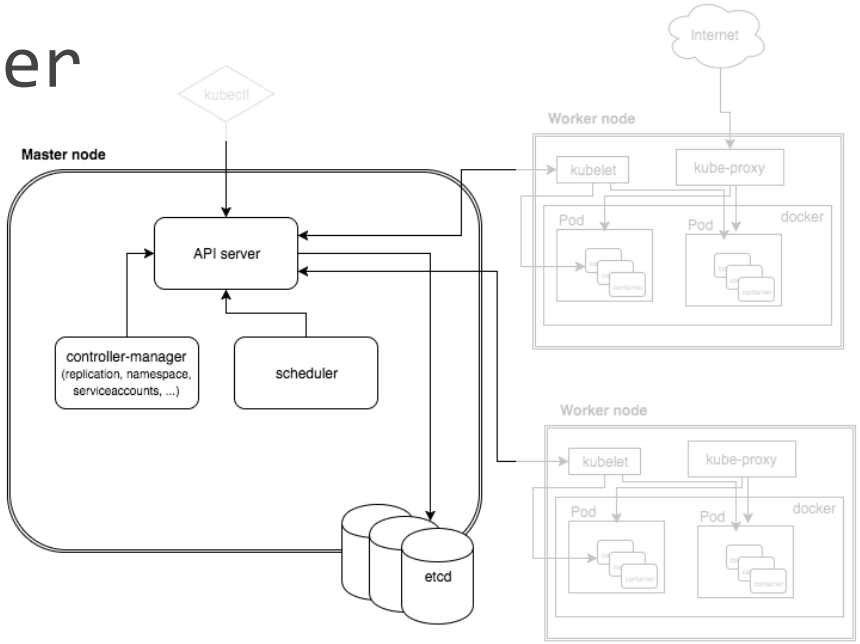
- api server
- etcd storage
- scheduler
- controller-manager




- simple, distributed, consistent key-value store. Used for shared configuration and service discovery.
- REST API for CRUD operations and an interface to register watchers on specific nodes (reliable way to notify the rest of the cluster about configuration changes)
- i.e. stored data: jobs being scheduled, created and deployed, pod/service details and state, namespaces, etc.

# master

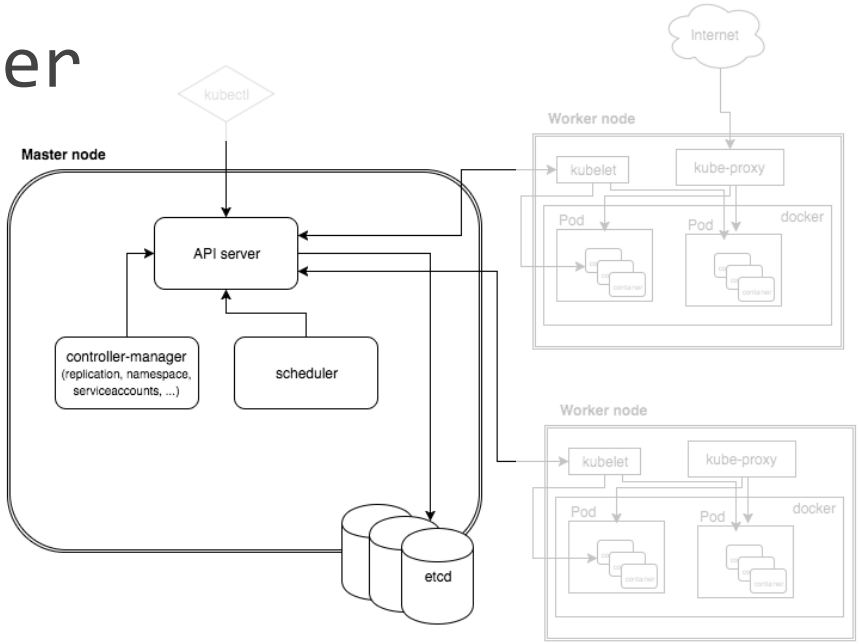
- api server
- etcd storage
- scheduler
- controller-manager




- manages the deployment of configured pods and services onto the nodes
- has the information regarding resources on the cluster, and required for the configured service to run
- decides where to deploy a specific service

# master

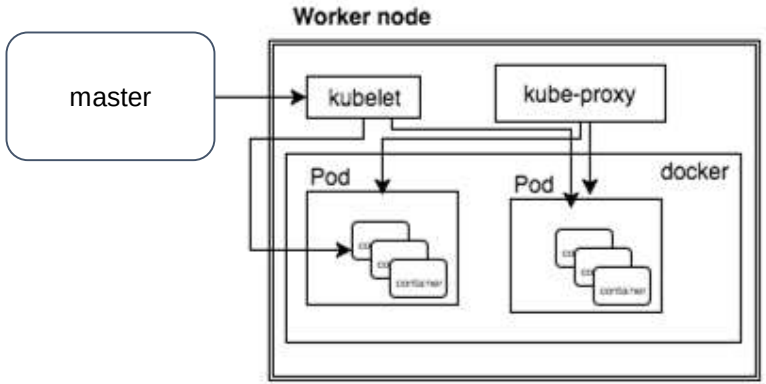
- api server
- etcd storage
- scheduler
- controller-manager




- controller-manager is a daemon embedding different kinds of controllers
- a controller uses apiserver to watch the shared state of the cluster and makes corrective changes to the desired one.
- i.e. Replication controller: takes care of the number of pods in the system
- Other are endpoints controller, namespace controller, and serviceaccounts controller (we will not dive into these)

# worker

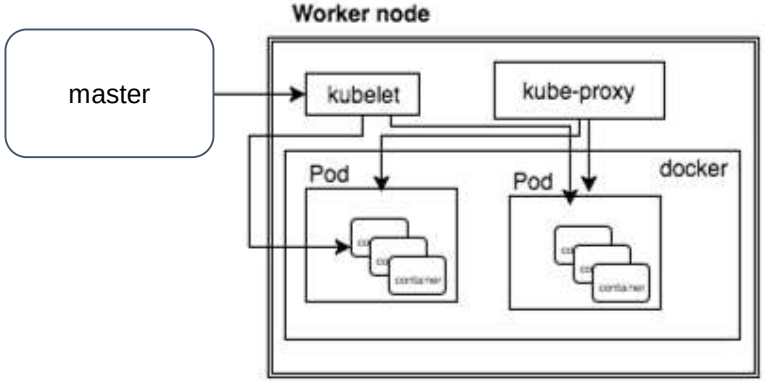
- docker
- kubelet
- kube-proxy




- on each of the worker nodes runs the configured pods.
- takes care of downloading the images and starting the containers.

# worker

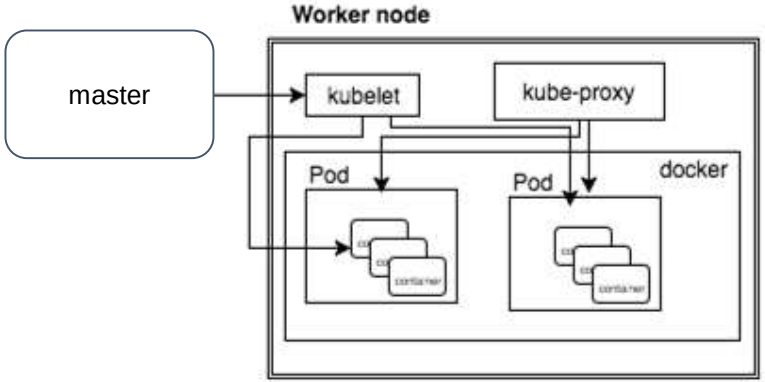
- docker
- kubelet
- kube-proxy




- responsible for communicating with the master node.
- gets the configuration of a pod from the apiserver and ensures that the described containers are up
- communicates with etcd, to get information about services and write the details about newly created

# worker

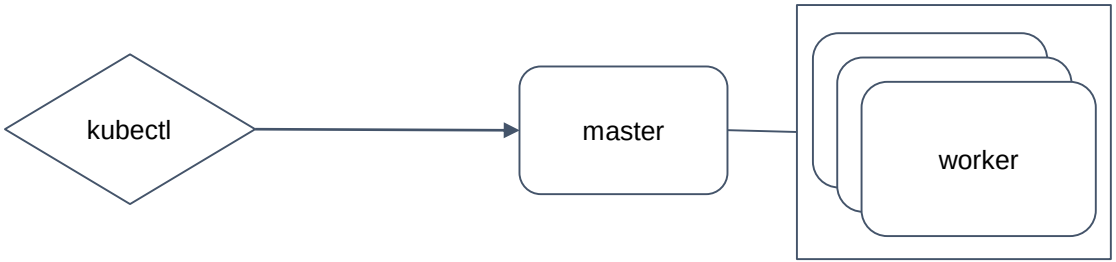
- docker
- kubelet
- kube-proxy




- acts as a network proxy and load balancer for a service on a single worker node.
- takes care of the network routing for TCP and UDP packets.

# kubectl

And the final bit – a command line tool to communicate with the API service and send commands to the master node.



# A cluster requires:

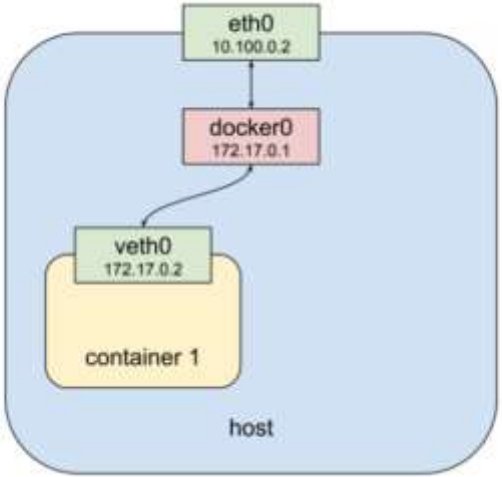
- a Container Runtime such as docker/containerd
- a Software Defined Network (SDN)
- a Transport Layer Security (TLS) to communicate securely

# Kubernetes networking

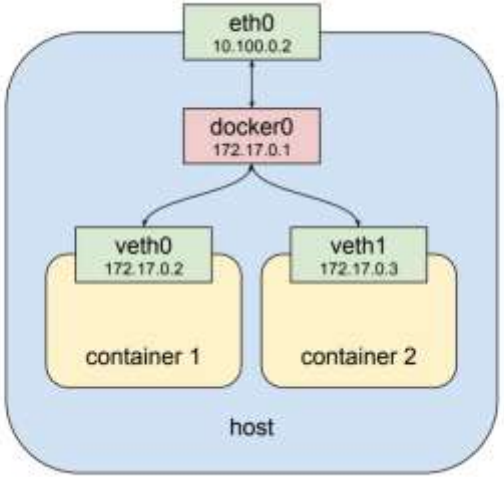
problems to address:

- **container-to-container communications:**
  - this is solved by pods and localhost communications
- **Pod-to-Pod communications:**
  - this is solved with overlay network (i.e. flannel)
- **Pod-to-Service communications:**
  - this is solved by kubeproxy+netfilter
- **External-to-Service communications:**
  - this is solved by nodeport (loadbalancers and ingress)

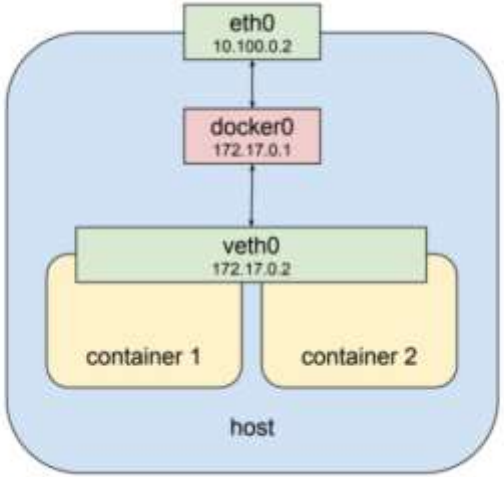
# container-to-container



# container-to-container

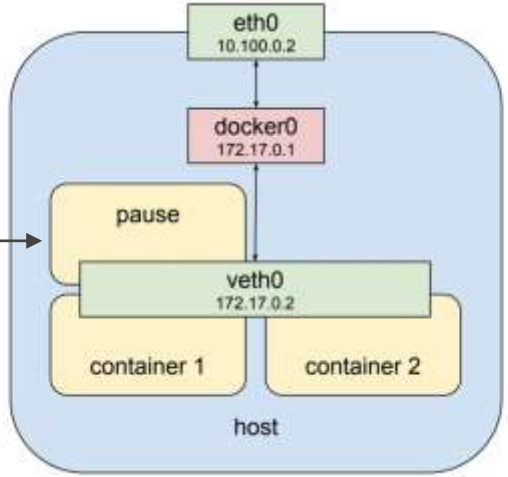


# Pod containers talking to each other

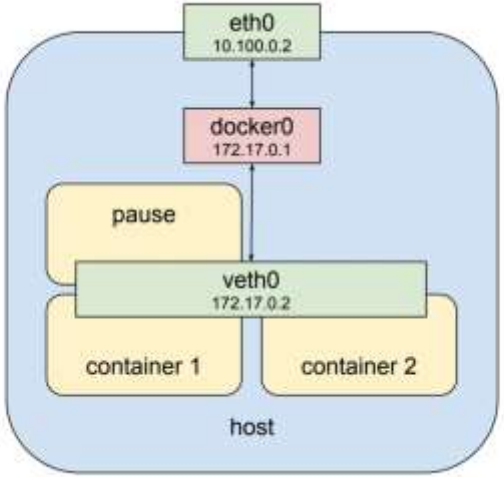


# Pod containers talking to each other

provides network i-face



# Pod containers talking to each other

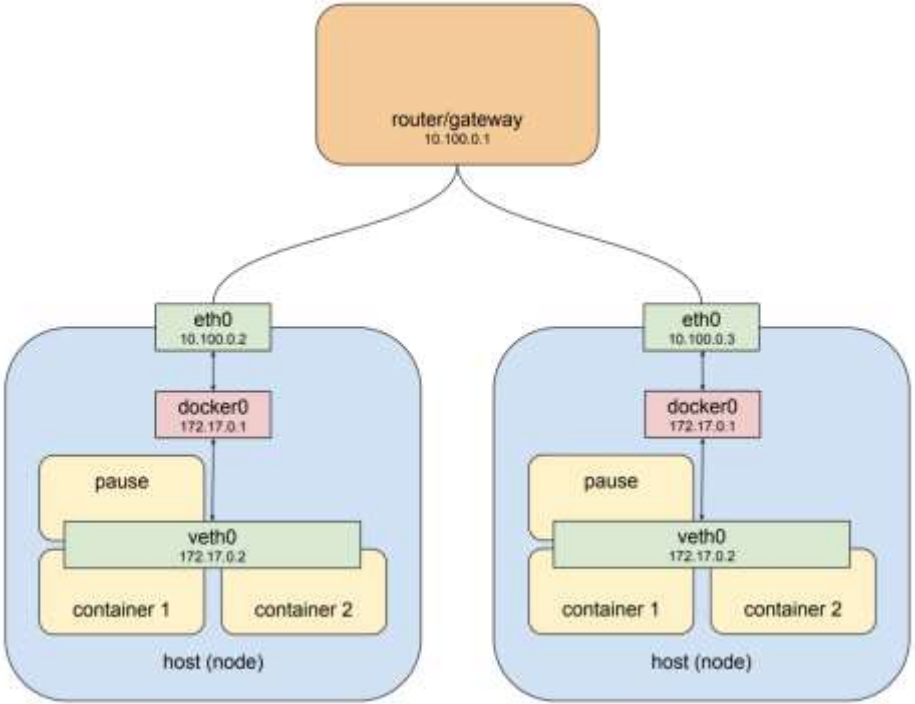



not enough… what’s next

(pods can be on different nodes)

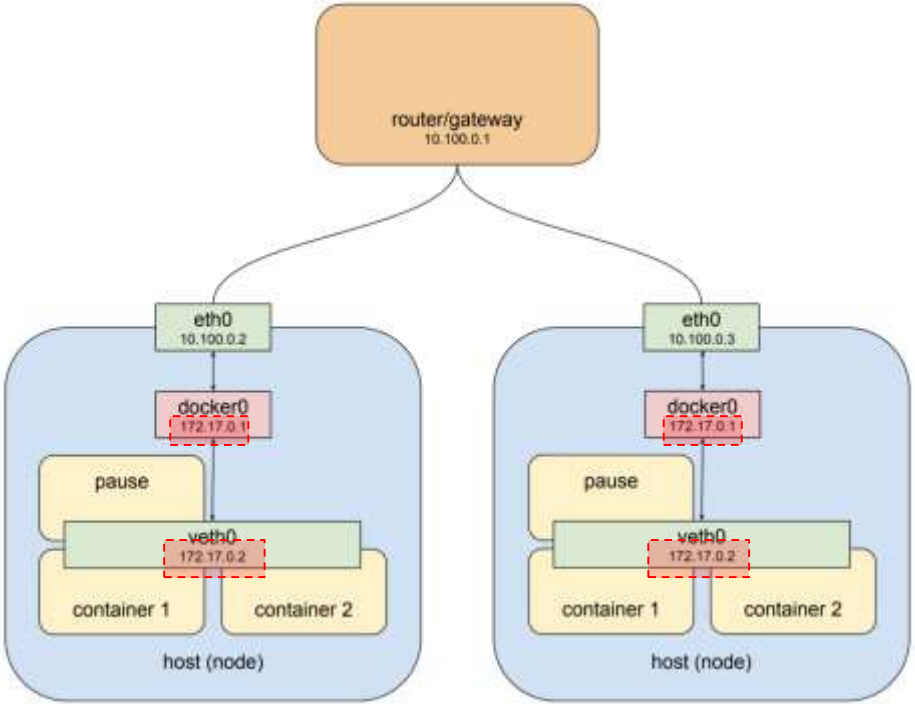
# pod-to-pod: overlay

- pods should be able to communicate with other pods irrespective of the location of the pods



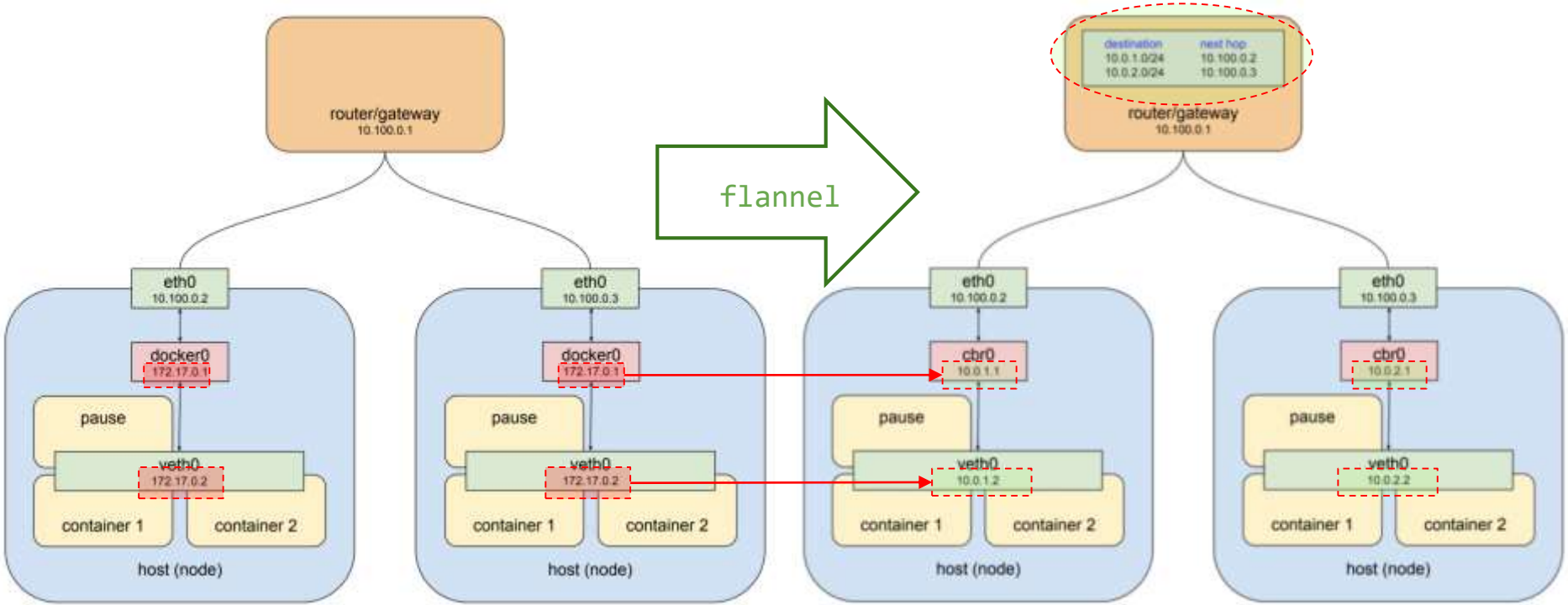
# pod-to-pod: overlay

- pods should be able to communicate with other pods irrespective of the location of the pods



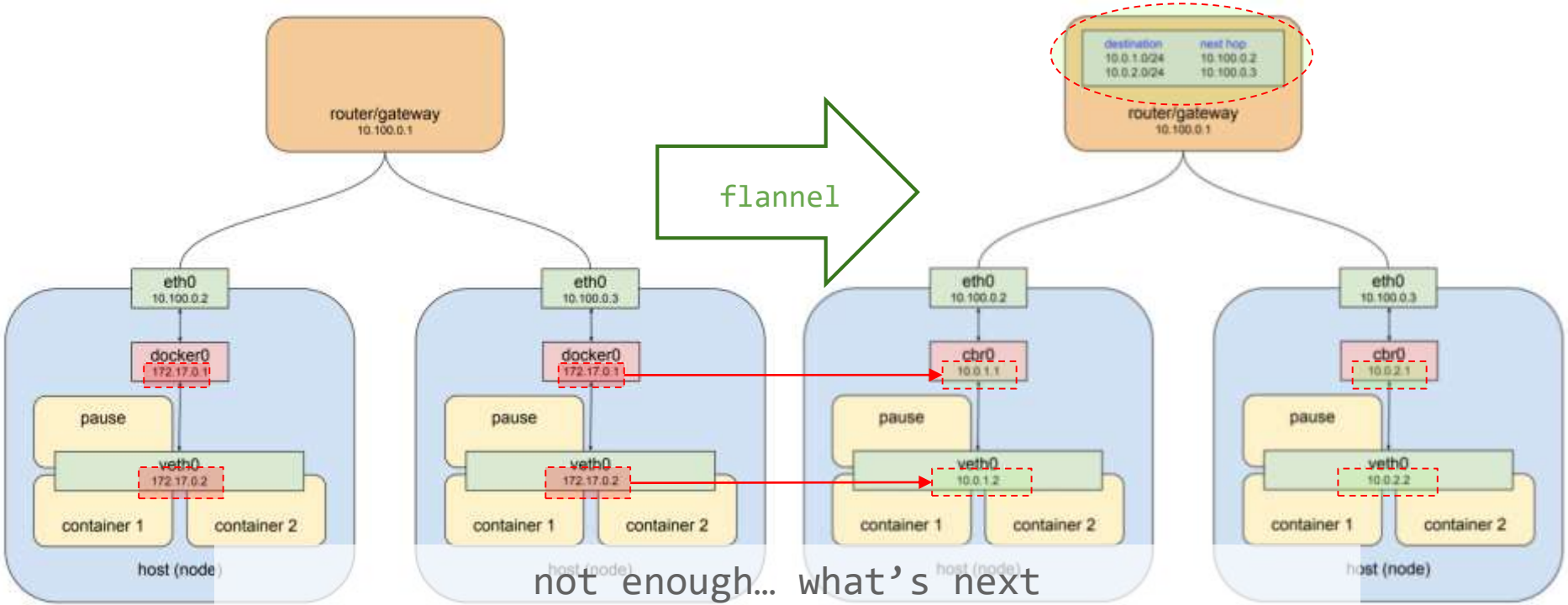
# pod-to-pod: overlay

- pods should be able to communicate with other pods irrespective of the location of the pods



# pod-to-pod: overlay

- pods should be able to communicate with other pods irrespective of the location of the pods

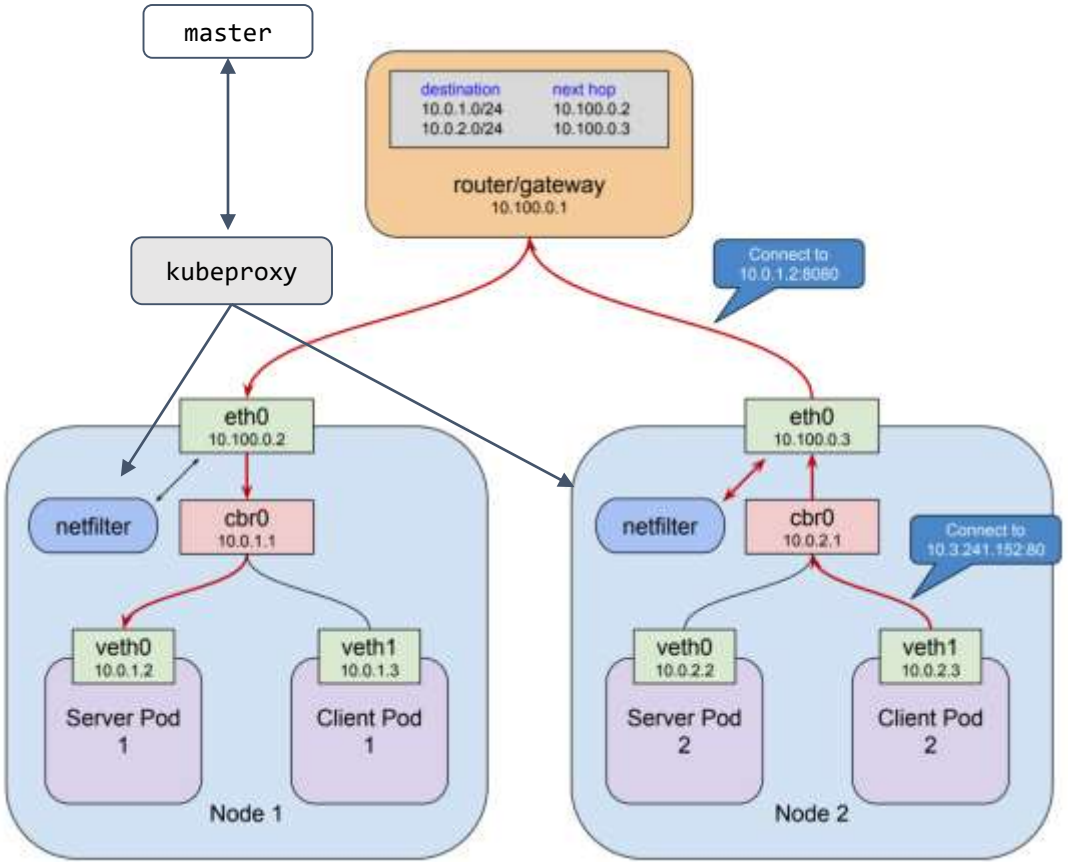



not enough… what's next

(pods do not last)

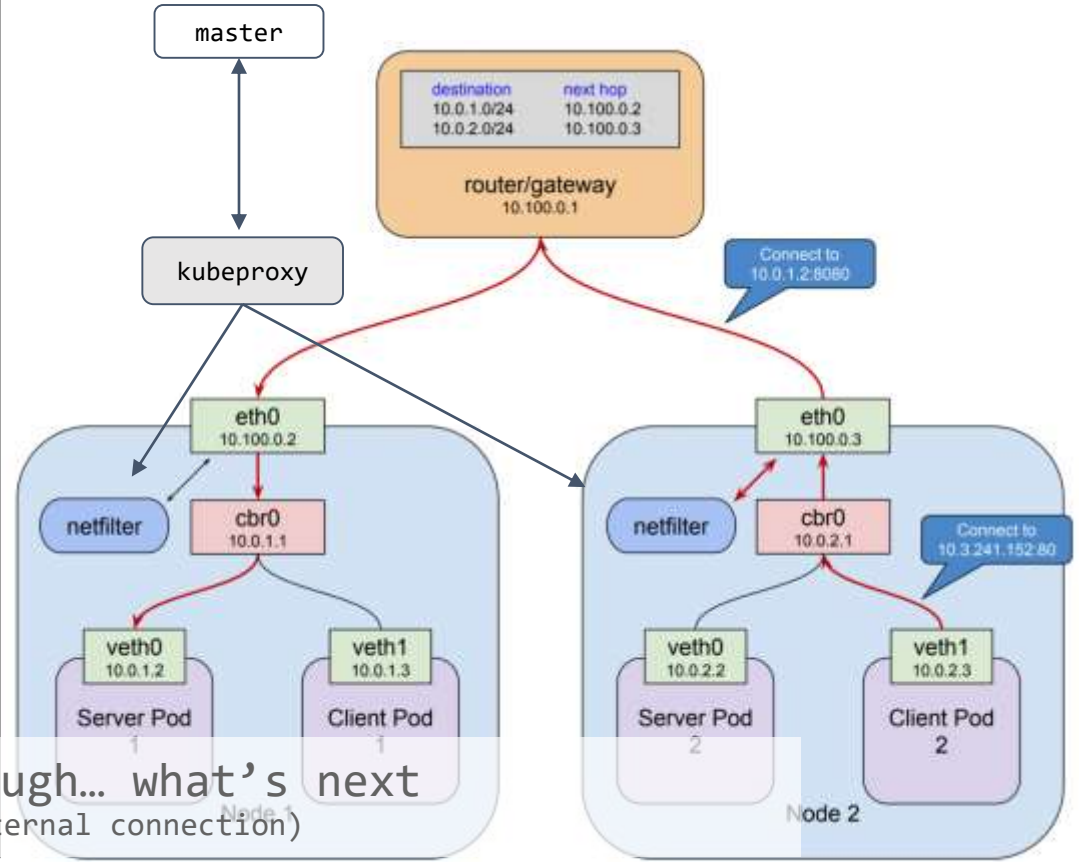
# pod-to-service: kubeproxy + netfilter
## (aka cluster-ip for loadbalancing)

- service: abstract way to expose an app running on a set of pods
- kube-proxy:
  - listens master api for changes (services and endpoints)
  - netfilter rules sync every update
- kube-proxy runs as a systemd unit (restarts when down)
- if the node is up so is netfilter
- cluster-ip (i.e. 10.3.241.152) to a service (separate range) and traffic is routed to a healthy pod



# pod-to-service: kubeproxy + netfilter
(aka cluster-ip for loadbalancing)

- service: abstract way to expose an app running on a set of pods
- kube-proxy:
  - listens master api for changes (services and endpoints)
  - netfilter rules sync every update
- kube-proxy runs as a systemd unit (restart when down)
- if the node is up so is netfilter
- cluster-ip (i.e. 10.3.241.152) to a service (separate range) and traffic is routed to a healthy pod




not enough… what's next
(external connection)

# external-to-service: publishing services

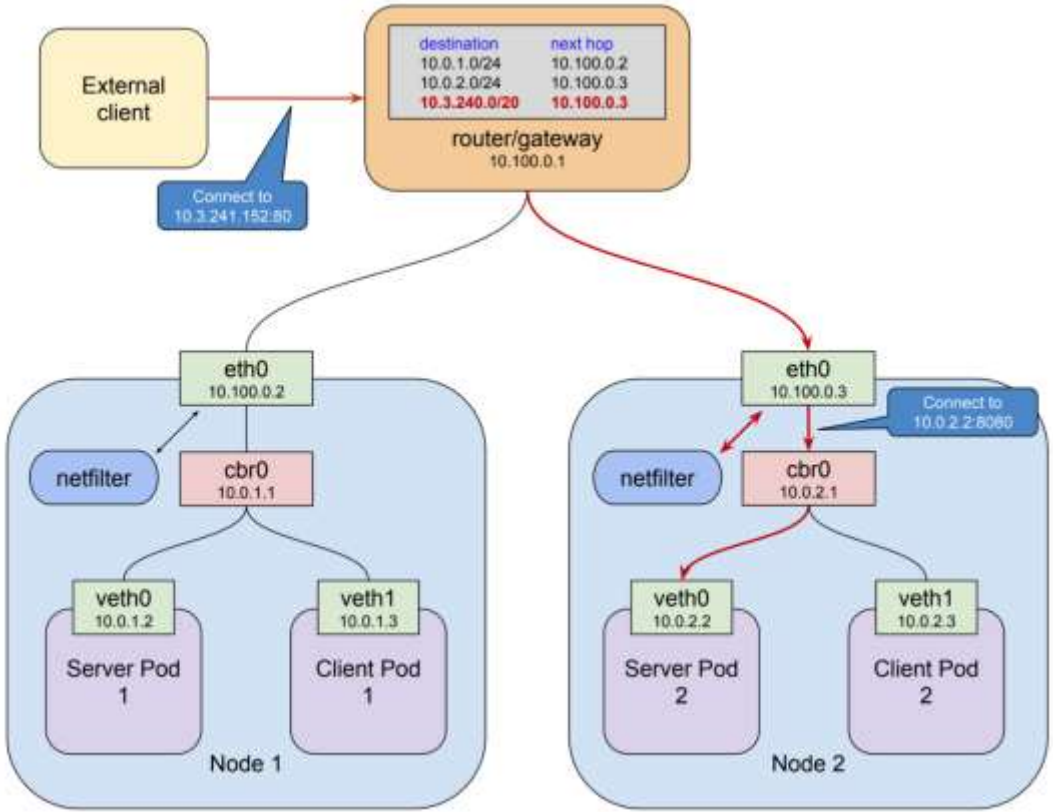
issues:

- forward traffic from a publicly visible IP endpoint to cluster IP that is only reachable once the packet is already on a node

# external-to-service: publishing services

issues:

- forward traffic from a publicly visible IP endpoint to cluster IP that is only reachable once the packet is already on a node
- nodes are ephemeral. If unreachable route should not break.
- if route durable, traffic balanced



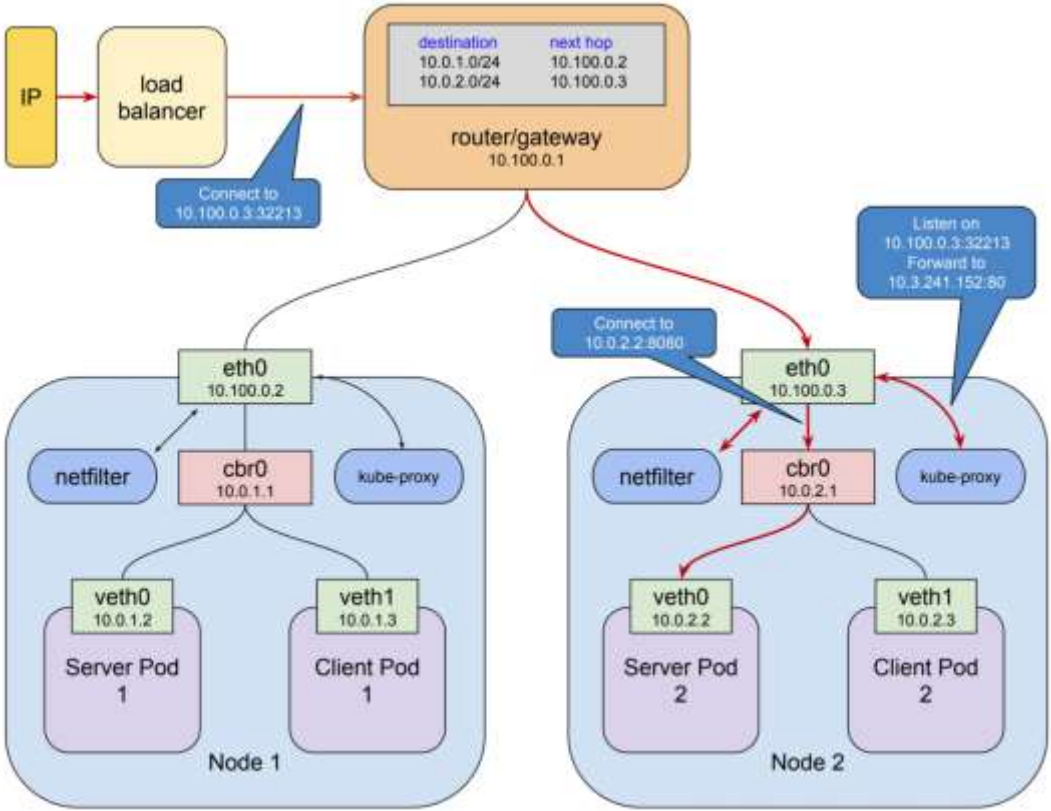
# external-to-service: publishing services

issues:

- forward traffic from a publicly visible IP endpoint to cluster IP that is only reachable once the packet is already on a node
- nodes are ephemeral. If unreachable route should not break.
- if route durable, traffic balanced

solution:

- nodeport
  - kube-proxy allocates a port 30000–32767
  - opens port on **eth0** interface of every node
  - connections are forwarded
- Loadbalancer service (nodeport+ingress path)
- ingress service (lb+https+virtual hosting...)



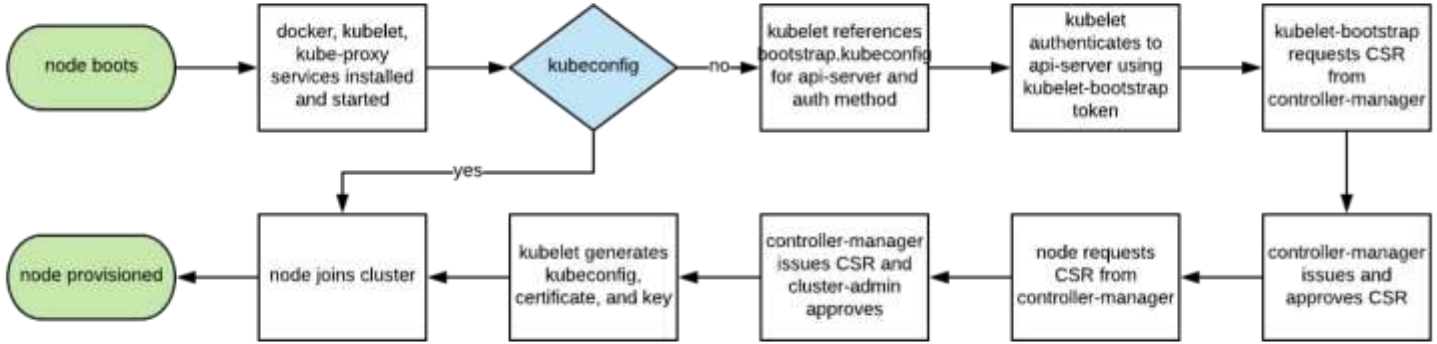
# flannel overlay network

- agent flanneld on each host
  - responsible for allocating a subnet to each host (etcd directly to store the network configuration)
- Kubernetes assumes that each container (pod) has a unique, routable IP inside the cluster.
  - this model removes the port mapping complexities that come from sharing a single host IP.
- Flannel provides a layer 3 IPv4 network between multiple nodes in a cluster.
  - does not control how containers are networked to the host, only
  - controls how the traffic is transported between hosts.

# kubelet and nodes: TLS communication

## security between layers: (i.e. easyRSA) provisioning of a node

TLS bootstrapping is designed to ease the ability of securely joining nodes to masters



# suggested readings and references

- A wonderful book about: **Metaphors We Live** by George Lakoff, Mark Johnson
- A Brief History of Cloud Application Architectures DOI: 10.3390/app8081368
- k8s bible for beginners and developers https://www.level-up.one/kubernetes-bible-beginners/
- kubernetes documentation at its website
- (find the veth) net debug https://community.pivotal.io/s/article/How-to-get-tcpdump-for-containers-inside-Kubernetes-pods
- About monopoly debate in cloud http://shirky.com/writings/powerlaw_weblog.html
- “The Big switch” N.Carr (yes it’s a paper book!)
- SDN and NFV Simplified: A Visual Guide to Understanding Software Defined Networks and Network Function Virtualization (Jim Doherty)
- About GARR cloud http://cloud.garr.it

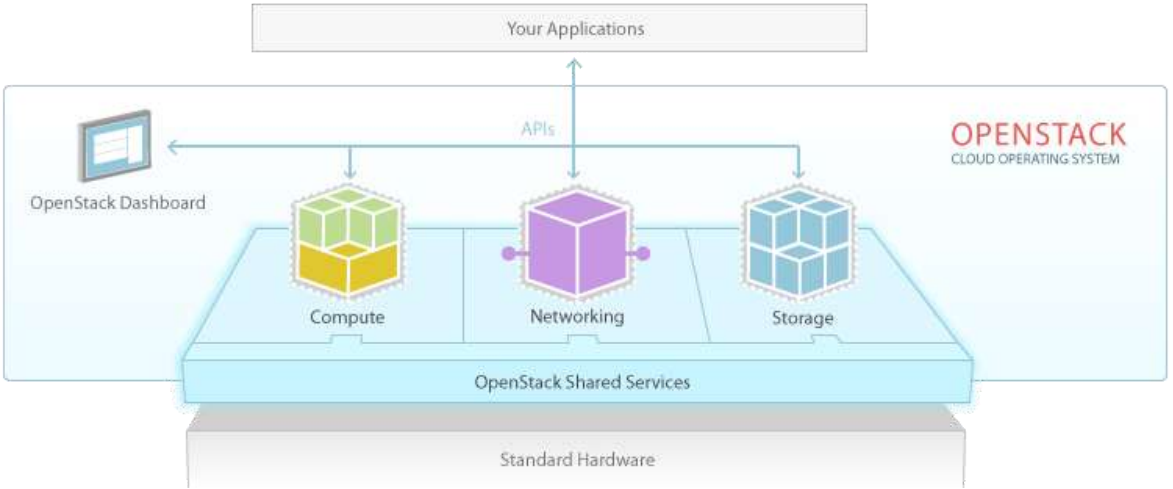
# Useful and Suggested readings

- About monopoly debate in cloud http://shirky.com/writings/powerlaw_weblog.html
- “The Big switch” N.Carr (yes it’s a paper book!)
- OpenStack Cloud Administrator Guide http://docs.openstack.org/admin-guide-cloud/content/index.html
- OpenStack keystone developer documentation http://docs.openstack.org/developer/keystone/
- OpenStack Identity Administration documentation http://docs.openstack.org/trunk/openstack-compute/install/content/ch_installingopenstack-identity-service.html
- Deploying openstack - Ken Pepple (O’Really)
- About GARR cloud http://cloud.garr.it
- OPENSTACK NETWORKING GUIDE (ask google for the latest)
- SDN and NFV Simplified: A Visual Guide to Understanding Software Defined Networks and Network Function Virtualization (Jim Doherty)
- OpenStack Networking Essentials (James Denton)
- Learning OpenStack Networking (Neutron) (James Denton)
- MAAS (CANONICAL) on www
- Juju (CANONICAL) on www
- Troubleshooting O~S (Alex Barchiesi) https://docs.google.com/document/d/1DcO7Cj_E_yJlj93g0ynkZ08YxsPJF1hQ23kS4F5awjA/edit?usp=sharing


let the dance
begin

# What is OpenStack ? (what is important for us)

- Apache 2.0 license (OSI)
- Open design process
- Publicly available code
- Open community processes documented and transparent
- Commitment to drive and adopt open standards
- Modular design for deployment flexibility via APIs

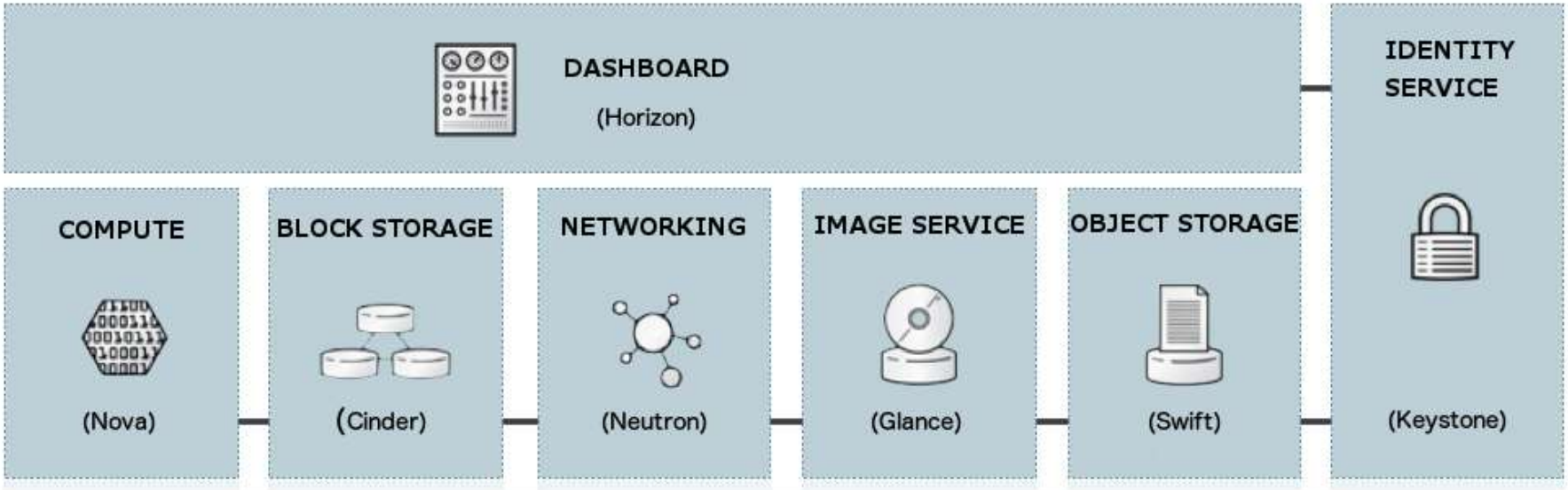



## Is a community

creating public software to build private and public clouds

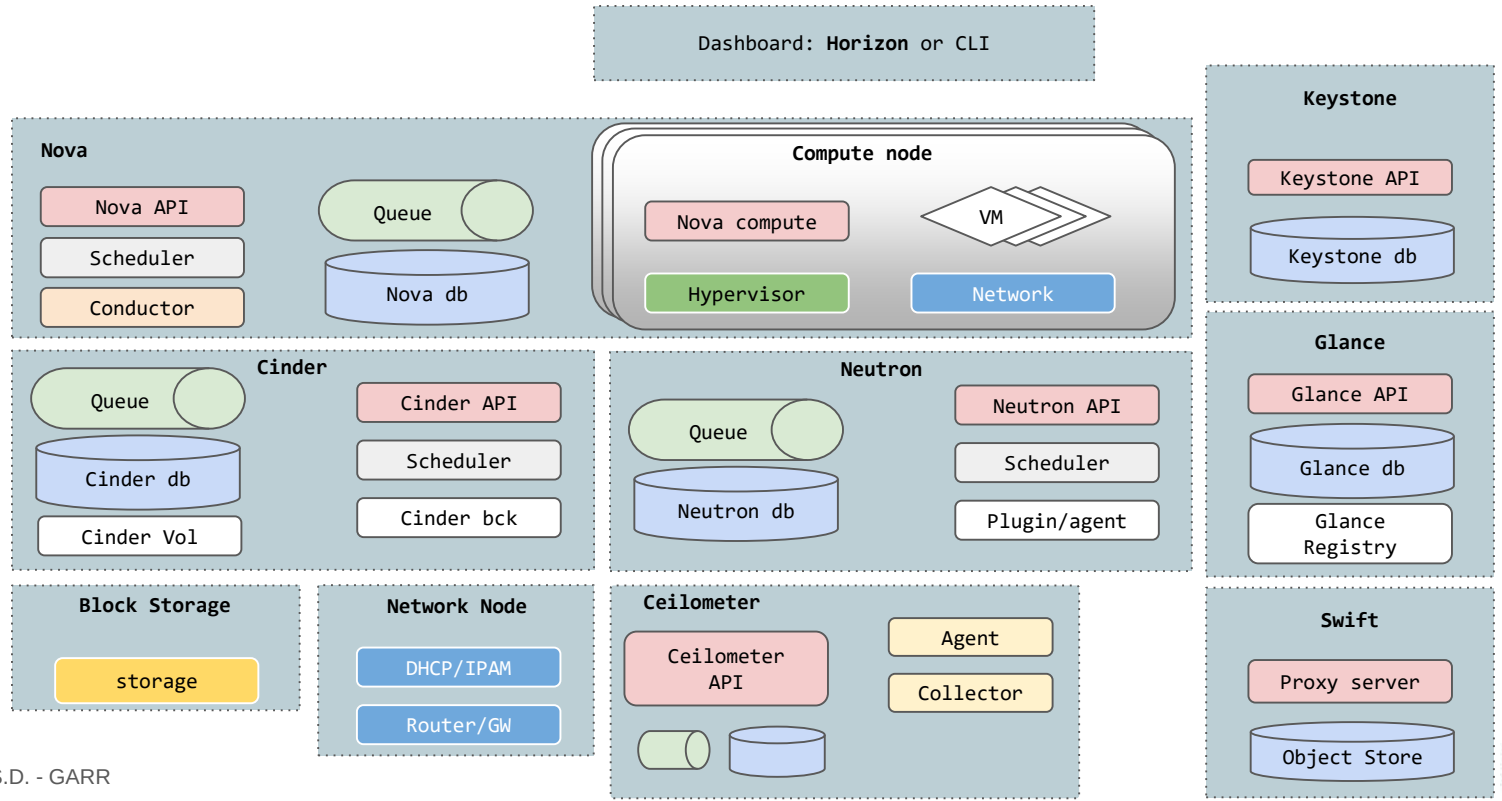
# Components (umbrella project for)

- **Horizon** (Dashboard)
- **Keystone** (Identity Management)
- **Nova** (Compute, where VMs are run)
- **Glance** (Image Service, where templates are)
- **Cinder** (Block Storage, persistent storage for VMs)
- **Swift** (Object Storage, snapshots and not frequently updated data)
- **Neutron** (Networking and SDN)
- **Ceilometer** (Telemetry)



# OpenStack through VM provisioning

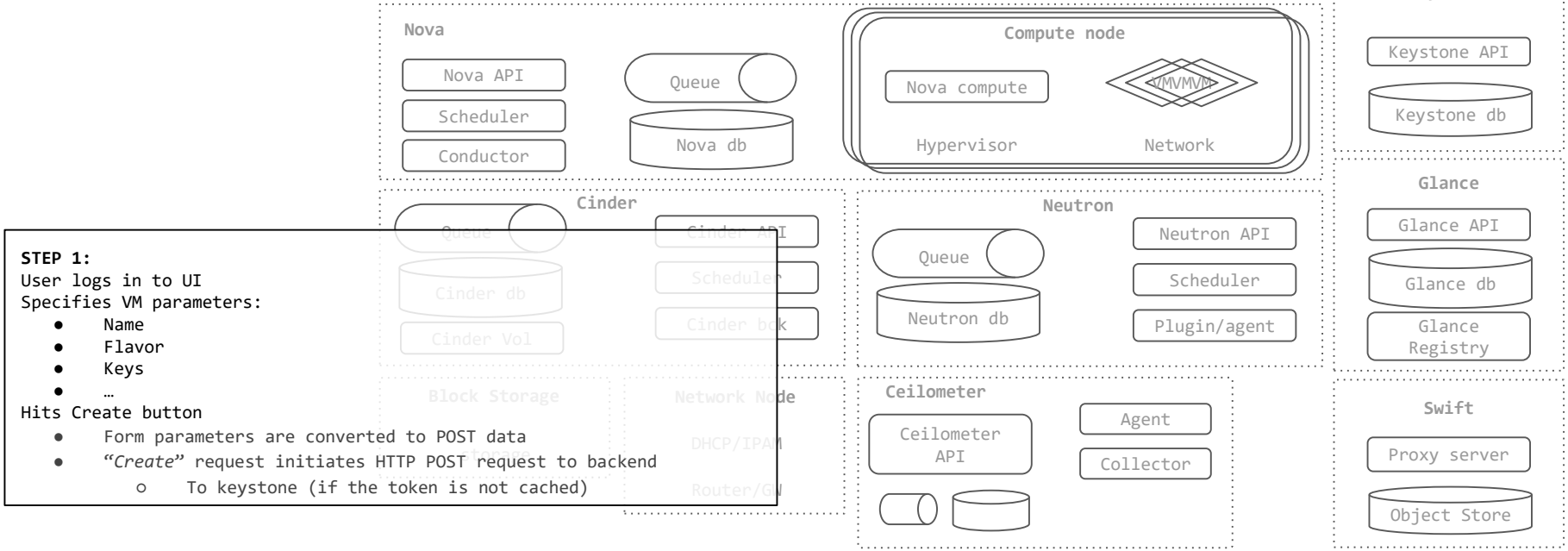
- Most common and complex process
- Involves interaction of most components



# Horizon: the OpenStack dashboard

Provides a baseline user interface for managing OpenStack services

- Is “stateless” - no database required
- Delegates error handling to the back-end
- Doesn’t support all the API functions
- Can use memcached or database to store sessions




**STEP 1:**
User logs in to UI
Specifies VM parameters:

- Name
- Flavor
- Keys
- …

Hits Create button

- Form parameters are converted to POST data
- *“Create”* request initiates HTTP POST request to backend
  - To keystone (if the token is not cached)

**STEP 2:**
Validate Auth Data:

- Horizon sends HTTP request to keystone. Auth info in HTTP headers
- Keystone sends temporary token back to Horizon via HTTP


K
33
8
m.m
Keystone
OLYMPIC
BOSTON 24, MASS U.S.A.

# Keystone: the OpenStack Identity Service

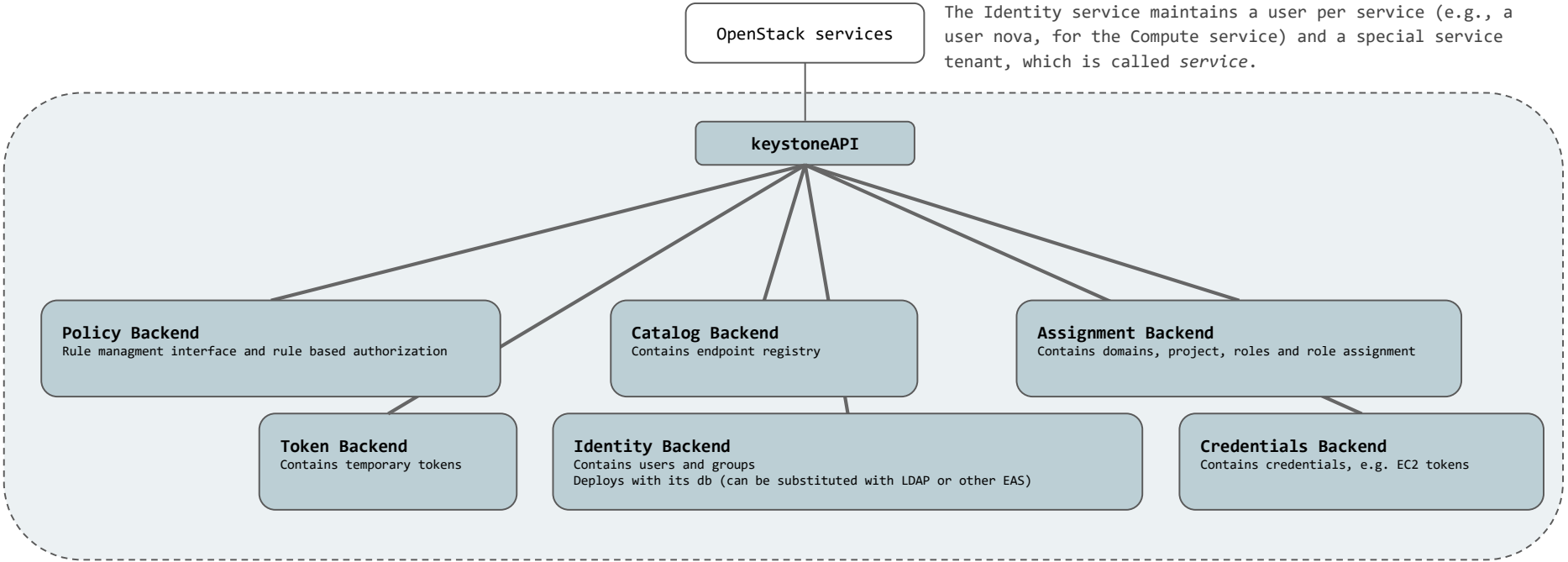
Provides **identity, token, catalog and policy services**

for use specifically by projects in the OpenStack family.

Provides **service catalog** to let other OpenStack systems know where relevant API endpoints for Services.
Two main concepts of Identity service management are:

- **Services**
- **Endpoints**

The Identity service maintains a user per service (e.g., a user nova, for the Compute service) and a special service tenant, which is called *service*.



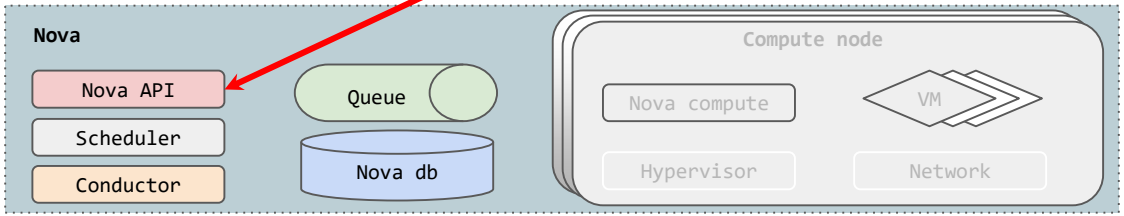

Nova
SS
RADIAL GT

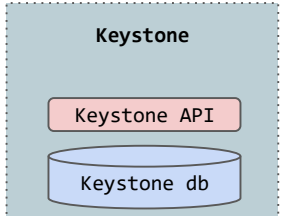
# OpenStack Compute API (Nova API)

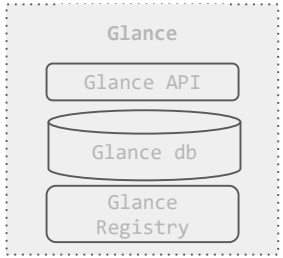
Nova API is a **RESTful** API web service used to interact with Nova

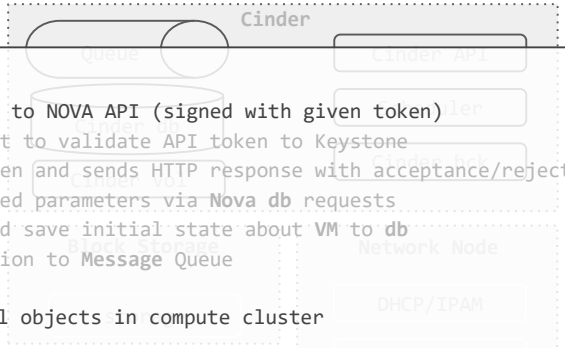
- Exposes **REST API** via HTTP
- Provides multiple APIs on different sub-domains:
  - **EC2-compatible** - starting to be deprecated
  - **Compute API** - all innovation happens here
- Is the only "allowed" way to interact with Nova
- Is "**stateless**"

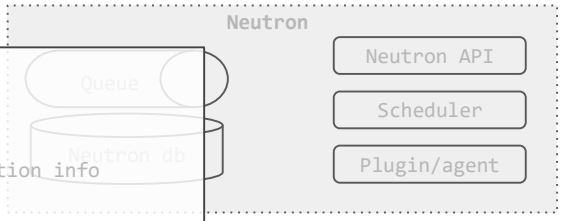
Dashboard: **Horizon** or CLI



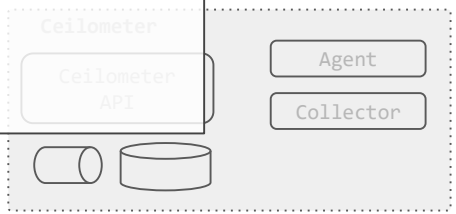



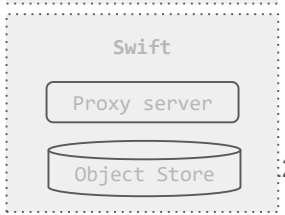










STEP3:

- **Horizon** sends POST request to NOVA API (signed with given token)
- **Nova API** sends HTTP request to validate API token to Keystone
- **Keystone** validates API token and sends HTTP response with acceptance/rejection info
- **Nova** validates cloud-related parameters via **Nova db** requests
- If request can be processed save initial state about **VM** to **db**
- **Send** message with next action to **Message** Queue

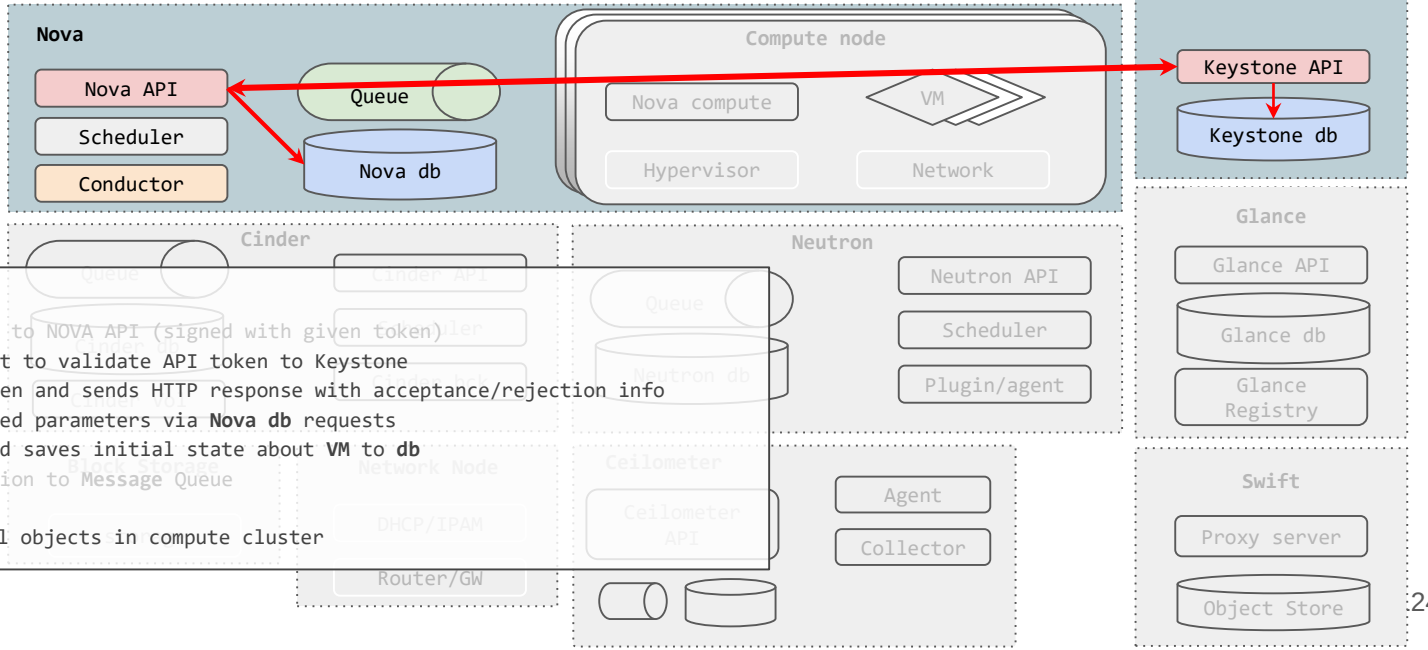
Nova db stores current state of all objects in compute cluster

# OpenStack Compute API (Nova API)

Nova API is a **RESTful** API web service used to interact with Nova

- Exposes **REST API** via HTTP
- Provides multiple APIs on different sub-domains:
  - **EC2-compatible** - starting to be deprecated
  - **Compute API** - all innovation happens here
- Is the only "allowed" way to interact with Nova
- Is "**stateless**"

Dashboard: **Horizon** or CLI




STEP4:

- **Horizon** sends POST request to NOVA API (signed with given token)
- **Nova API** sends HTTP request to validate API token to Keystone
- **Keystone** validates API token and sends HTTP response with acceptance/rejection info
- **Nova** validates cloud-related parameters via **Nova db** requests
- If request can be processed saves initial state about **VM** to **db**
- Send message with next action to **Message** Queue

Nova db stores current state of all objects in compute cluster

# OpenStack Nova Scheduler

Dashboard: **Horizon** or CLI

**Nova**

Nova API

Scheduler

Conductor

Queue

Nova db

Compute node

Nova compute

VM

Hypervisor

Network

**Keystone**

Keystone API

Keystone db

Neutron API

Scheduler

Plugin/agent

Glance

Glance API

Glance db

Glance Registry

Swift

Proxy server

Object Store

STEP5:

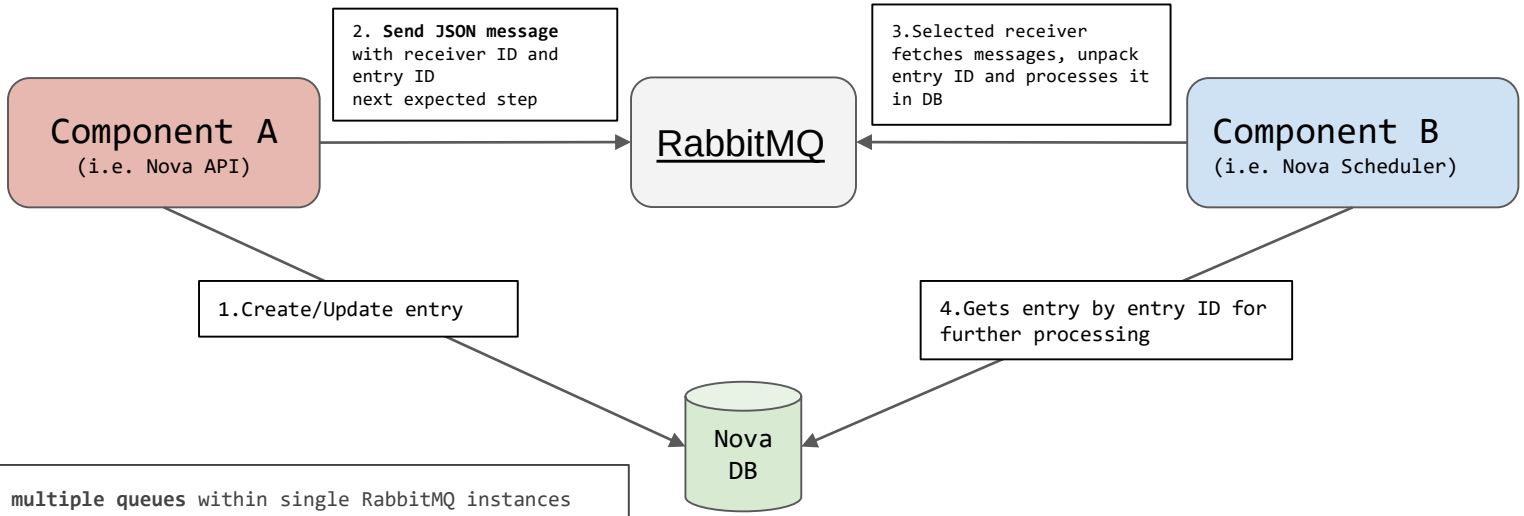
- **Horizon** sends POST request to NOVA API (signed with given token)
- **Nova API** sends HTTP request to validate API token to Keystone
- **Keystone** validates API token and sends HTTP response with acceptance/rejection info
- **Nova** validates cloud-related parameters via **Nova db** requests
- If request can be processed save initial state about **VM** to **db**
- **Sends** message with **VM info** to **Message** Queue and **scheduler** picks it up
- **Scheduler** fetches info about the whole cluster from **db, filters, selects compute** and updates db
- **Scheduler** publishes message to the compute queue to **trigger VM** provisioning

Nova db stores current state of all objects in compute cluster


RabbitMQ
© Michelle Middleton

# Message Queue

Is a unified way for collaboration between sub-components




- Uses **multiple queues** within single RabbitMQ instances
  - Used by services to build machine state
  - Each compute node has a queue
- Message traffic is **not intensive**
- **No broadcast** messages
  - Monitoring uses API polling
- **HA** should be be configured **separately**
  - Mirror queues not handled by OpenStack

# Nova Compute

Nova Compute is a worker daemon, which primarily **creates and terminates VMs** via Hypervisor API

Nova Conductor is the key to **no-db-compute**

- Eliminates remote db access
- Horizontal scalability
- Hides db implementations from Nova Compute (upgrades)
- Beneficial for operations that cross multiple compute (migration, resize)

Dashboard: **Horizon** or CLI

Nova

Nova API

Scheduler

Conductor

Queue

Nova db

Compute node

Nova compute

VM

Hypervisor

Network

Keystone

Keystone API

Keystone db

Glance

Glance API

Glance db

Glance Registry

Cinder

Queue

Cinder API

Scheduler

Cinder bck

Neutron

Queue

Neutron db

Neutron API

Scheduler

Plugin/agent

Ceilometer

Ceilometer API

Agent

Collector

Swift

Proxy server

Object Store

STEP6:

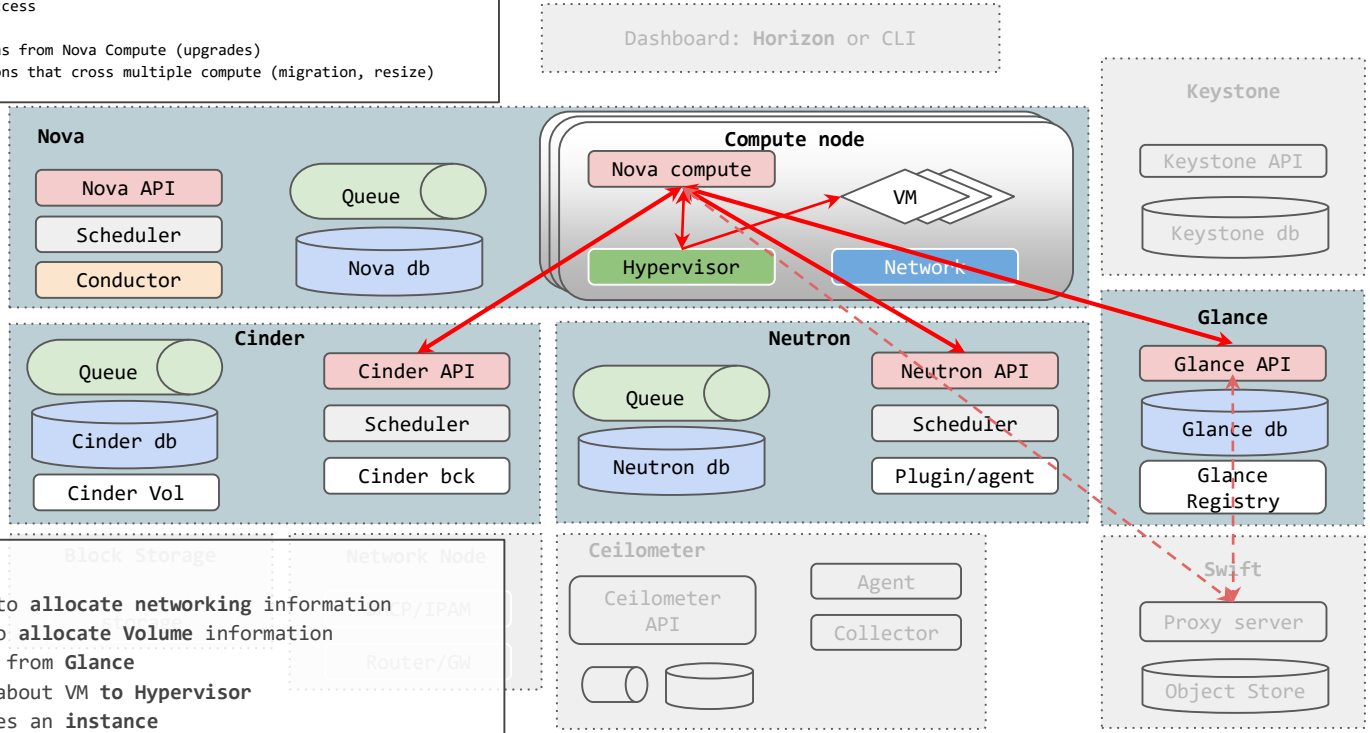
- **Nova Compute** gets message from MQ
- Asks **Nova Conductor** for **VM info** from DB
- Queries Neutron to allocate networking information
- Queries Cinder to allocate Volume information
- Fetches VM image from Glance
- Passes all info about VM to Hypervisor
- Hypervisor creates an instance

# Nova Compute

Nova Compute is a worker daemon, which primarily **creates and terminates VMs** via Hypervisor API

Nova Conductor is the key to **no-db-compute**

- Eliminates remote db access
- Horizontal scalability
- Hides db implementations from Nova Compute (upgrades)
- Beneficial for operations that cross multiple compute (migration, resize)




STEP7:

- **Queries** Neutron to **allocate networking** information
- **Queries** Cinder to **allocate Volume** information
- Fetches VM **image** from **Glance**
- Passes all info about VM **to Hypervisor**
- **Hypervisor** creates an **instance**

# Nova Compute drivers (for reference)

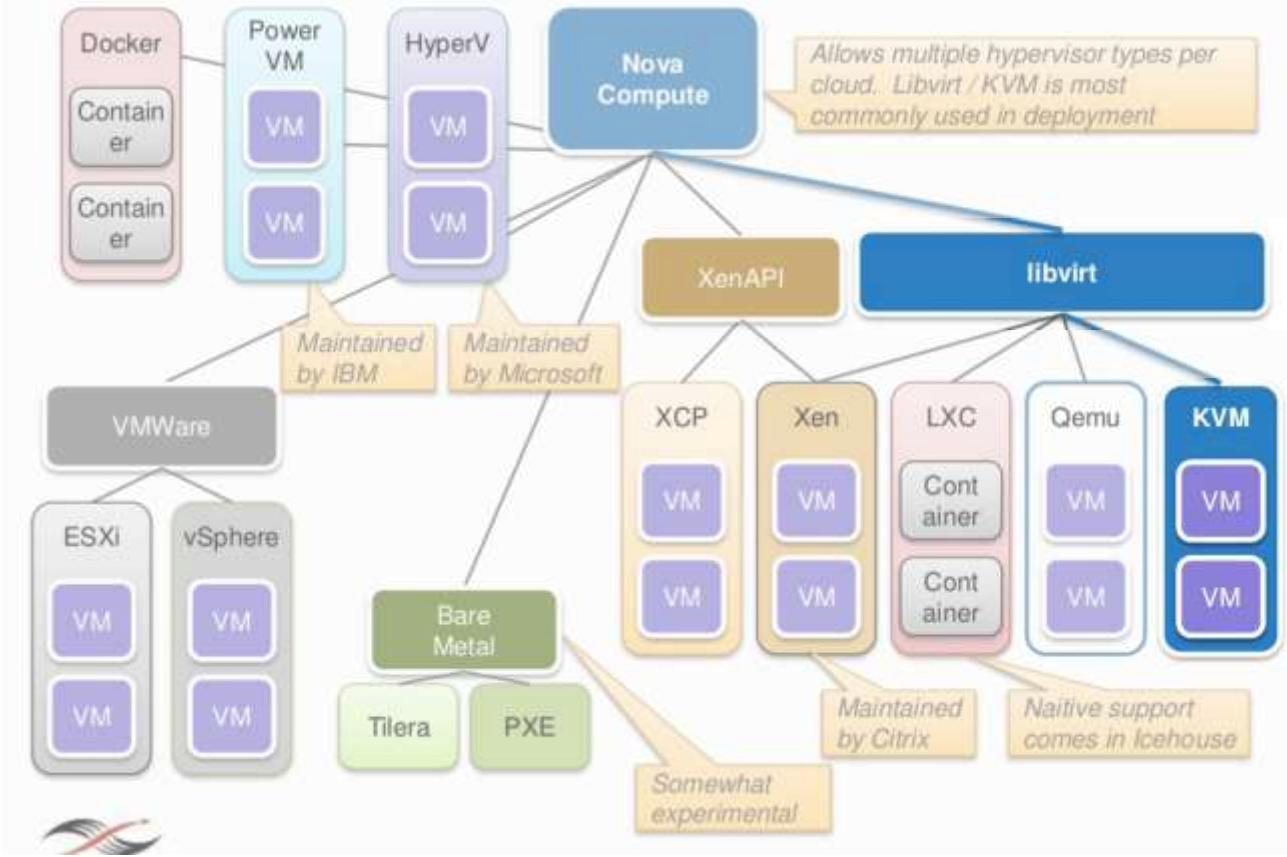



Image Courtesy of MIRANTIS


glance

# Glance: OpenStack image service

Provides services for:

- Discovering
- Registering
- Retrieving virtual machine images

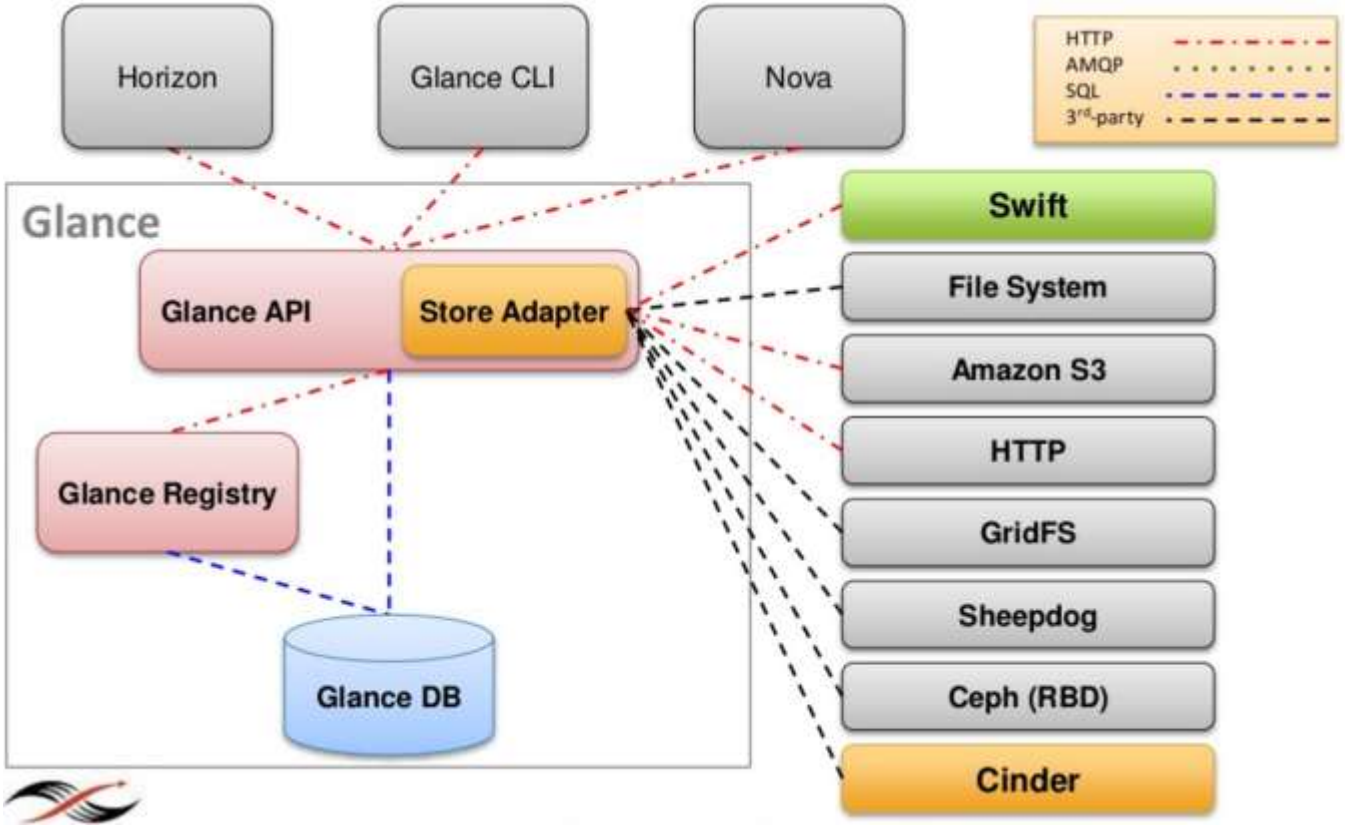
May use **multiple backends** for image storage

May store the same image in multiple locations

Supports **multiple** image **formats**

| Disk Format | Description |
|---|---|
| raw | an unstructured (unrestricted) disk image format |
| vhd | VHD disk format, a common disk format used by virtual machine monitors from VMWare, Xen, Microsoft, VirtualBox, and others |
| vmdk | Another common disk format supported by many common virtual machine monitors |
| vdi | disk format supported by VirtualBox virtual machine monitor and the QEMU emulator |
| iso | archive format for the data contents of an optical disc (e.g. CDROM) |
| qcow2 | disk format supported by the QEMU emulator that can expand dynamically and supports Copy on Write |
| aki | indicates what is stored in Glance is an Amazon kernel image |
| ari | indicates what is stored in Glance is an Amazon ramdisk image |
| ami | indicates what is stored in Glance is an Amazon machine image |

# Glance architecture



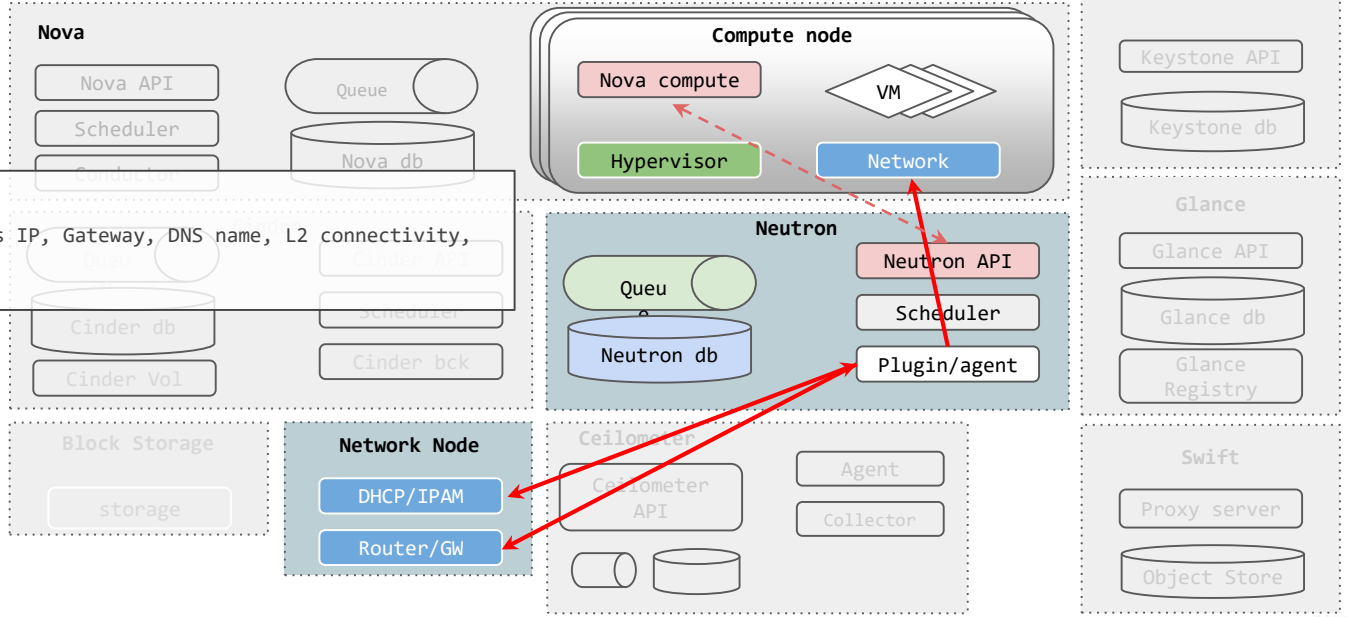

neutron
rail box
RB OX
41 291
next load
any road
NETWORK

# Openstack networking:Neutron (configure VM Network)

- Provides a flexible **API (POST/GET)** for service providers or their tenants **to manage OpenStack network topologies.**
  - **Create networks, associate VMs, set routers,** etc.
- Presents a **logical API** and a corresponding plug-in architecture that **separates** the **description** of network connectivity from its **implementation.**

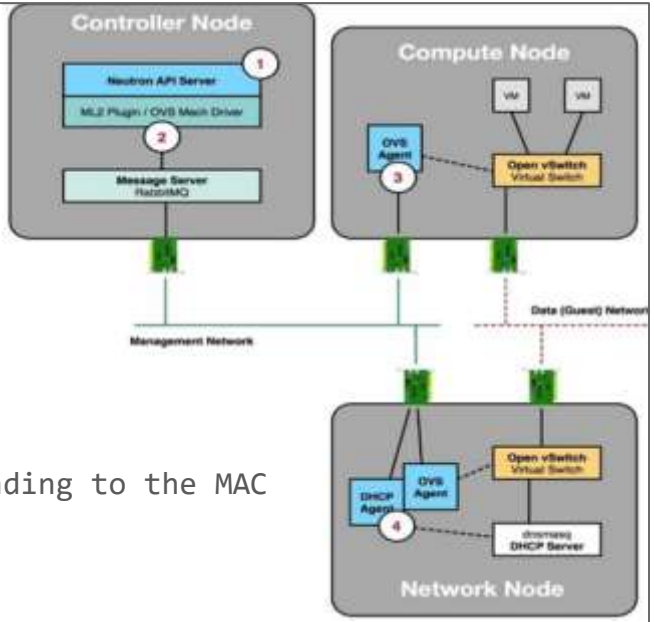
STEP9:

- Neutron configures IP, Gateway, DNS name, L2 connectivity, etc



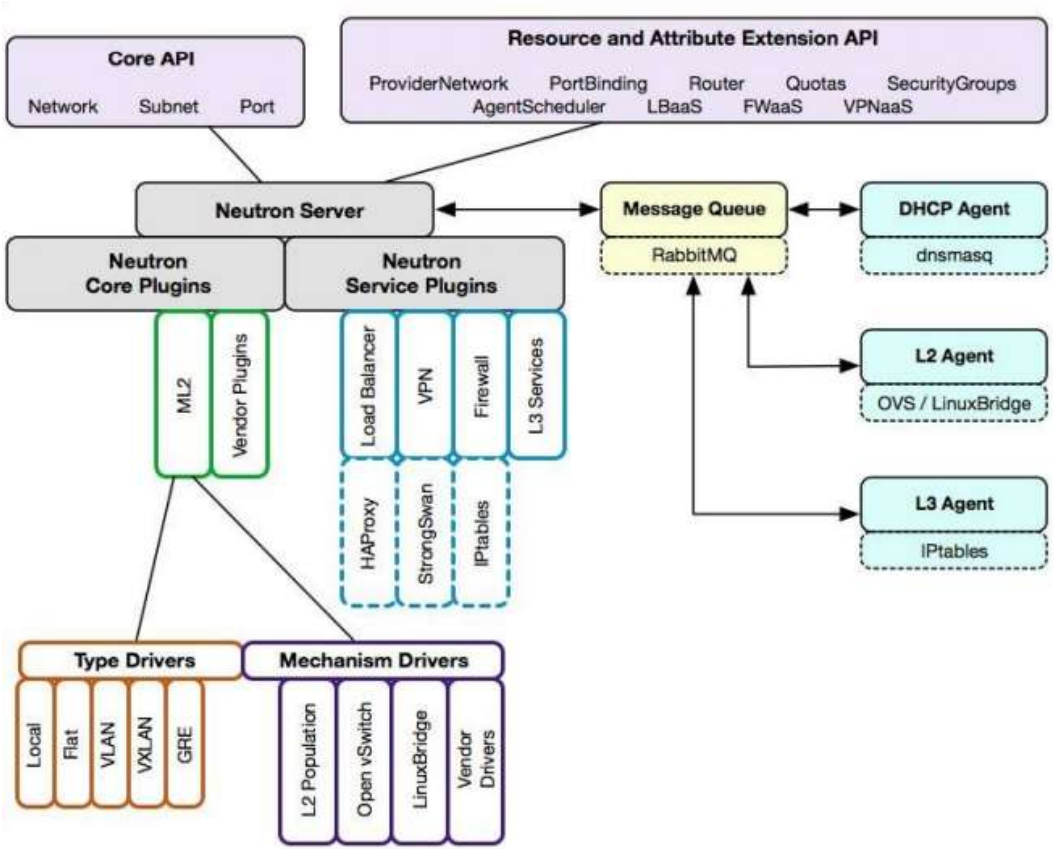
# Neutron WorkFlow: booting an instance

1. Create a network
2. Create a subnet and associate with the network
3. Boot the VM and specify the network
4. Nova interacts with Neutron to create a port on the network
5. Neutron assigns a MAC and IP to the newly created port
6. Nova starts the VM
7. VM sends a DHCP request during boot and gets the IP corresponding to the MAC

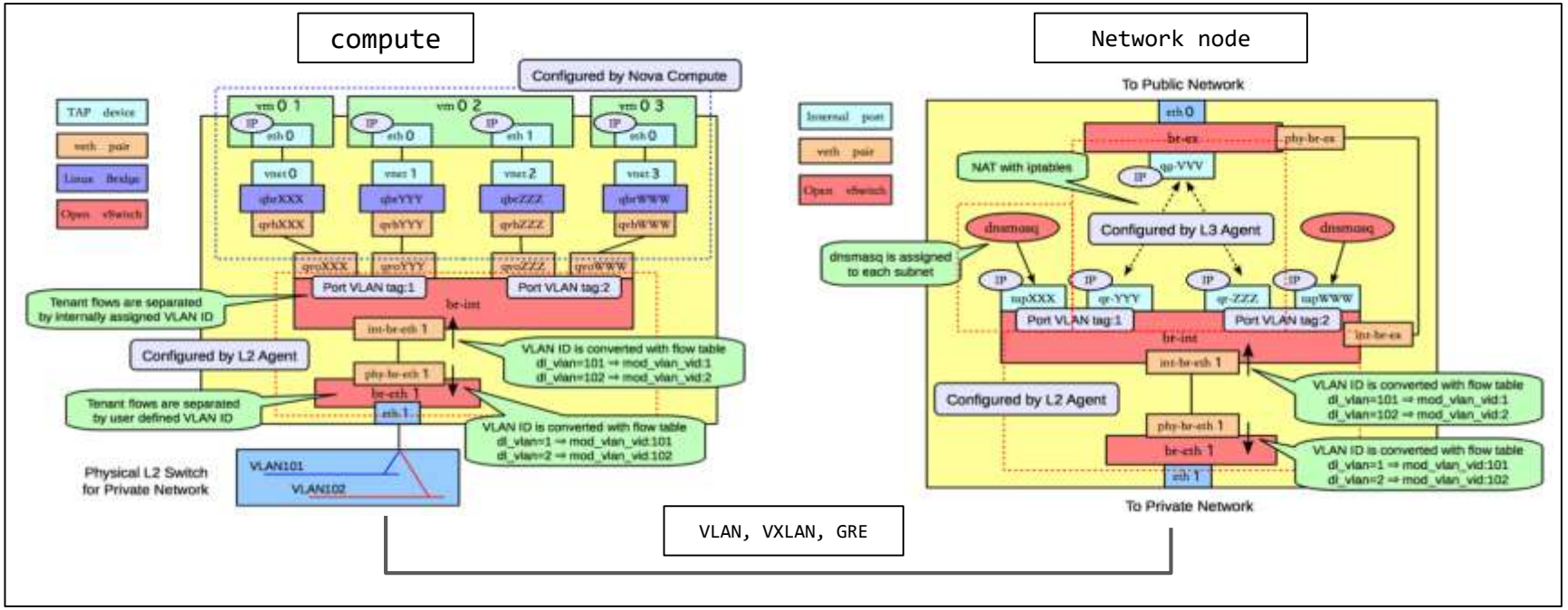


# Neutron main components

- Neutron Server
- Plugins
- Plugin Agents
- Message Queue
- Database
- DHCP Agent
- L3 Agent



# Networking in (too many) details




[O~S networking trobleshooting link]

# Storage Models

- Ephemeral
  - Persists until VM terminated
  - Accessible from within VM as local file system
  - Used to run operating system and or scratch space
  - Managed by Nova
- Block
  - Persists until specifically deleted by user
  - Accessible from within VM as a block dev
  - Used to add additional persistent storage to VM and/or run operating system
  - Managed by Cinder
- Object
  - Persists until specifically deleted by user
  - Accessible from anywhere
  - Used to add store files, including VM images
  - Managed by Swift

# Nova Compute (Requests Volume)

Dashboard: **Horizon** or CLI

STEP8:

- contacts **Cinder** to **get Volume data**
- **Instruct** the **Hypervisor** to use vol. As a new block dev

Nova

Nova API

Queue

Scheduler

Conductor

Nova db

**Compute node**

Nova compute

VM

Hypervisor

Network

Keystone

Keystone API

Keystone db

**Cinder**

Queu

Cinder API

Cinder db

Scheduler

Cinder Vol

Cinder bck

Neutron

Queu

Neutron API

Scheduler

Neutron db

Plugin/agent

Glance

Glance API

Glance db

Glance Registry

**Block Storage**

storage

Network Node

DHCP/IPAM

Router/GW

Ceilometer

Ceilometer API

Agent

Collector

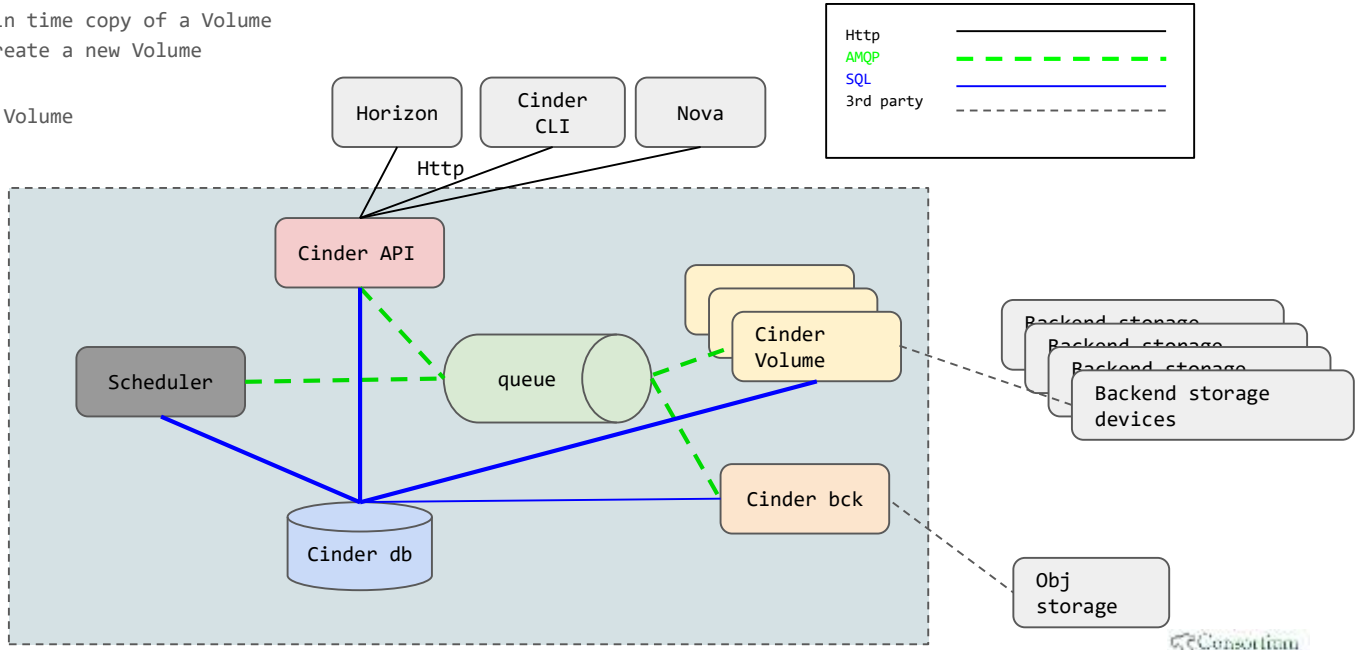
Swift

Proxy server

Object Store

# Cinder resources: OpenStack block storage

- Volume
  - Is a persistent R/W block storage device
  - Can be attached to VMs as a secondary storage
  - Can be root store to boot VMs
  - Can be attached only to one instance at a time
  - Keeps its state independent of an instance
- Snapshot
  - Is a read only point in time copy of a Volume
  - Can then be used to create a new Volume
- Backup
  - An archived copy of a Volume

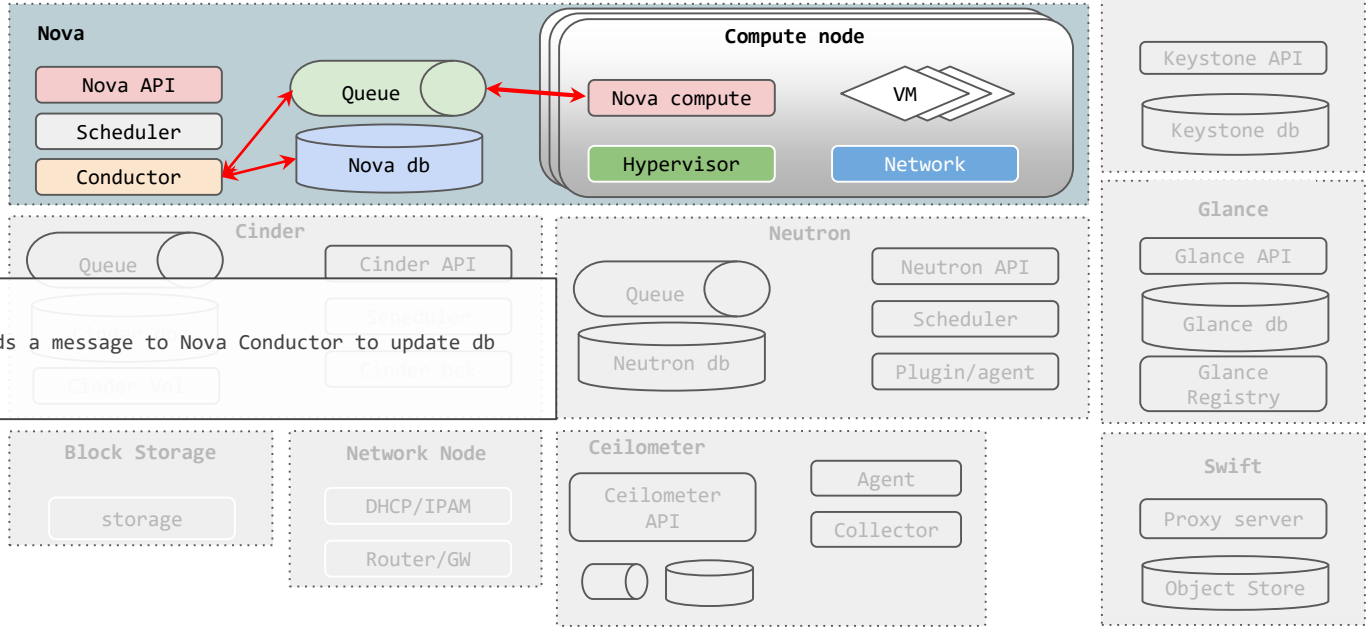


# Cinder Volume driver (for reference)

- iSCSI:
  - Dell EqualLogic
  - EMC VMAX/VNX
  - Hitach HDS
  - HP 3PAR (StoreServ)
  - HP / Lefthand SAN (StoreVirtual)
  - Huawei T/Dorado/HVS
  - IBM Storwize family/SVC/XIV
  - **LVM (Reference Implementation)**
  - Nexenta
  - NetApp
  - SolidFire
  - VMware VMDK
  - Windows Server 2012
  - Zadara
- GlusterFS NFS (volumes as sparse files)
- IBM General Parallel File System (GPFS) (volumes as sparse files):
  - GPFS NSD
- ATA over Ethernet (AoE):
  - Coraid
- Fibre Channel:
  - NetApp
  - HP 3PAR (StoreServ)
  - Huawei T/Dorad/HVS
  - IBM Storwize family/SVC/XIV
  - VMware VMDK
- NFS (volumes as sparse files):
  - NFS
  - Nexenta
  - NetApp
  - VMware VMDK
  - Zadara
  - XenAPI Storage Manager
- RADOS Block Devices (RBD):
  - Ceph
- Shared SAS:
  - VMware VMDK
- Scale Out File System (SOFS) (volumes as sparse files):
  - Scality
- VirtIO (Local raw storage) (volumes as sparse files)

More on this will follow

# VM is up




STEP10:

- **Nova Compute** sends a message to Nova Conductor to update db with VM state

# User is happy

STEP 11:

- Horizon polls Nova API for **VM status** and power state (taken from the db)