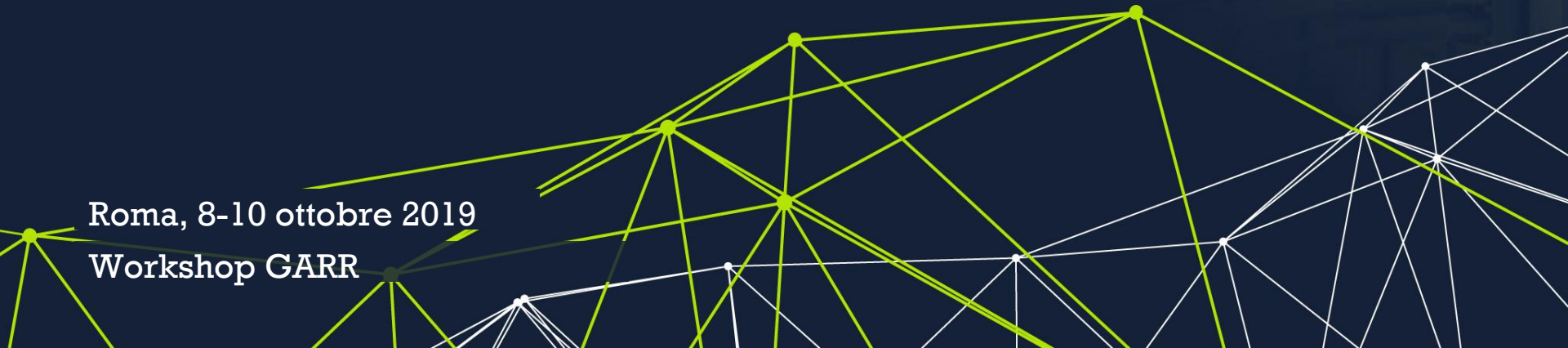
Consortium GARR | THE ITALIAN EDUCATION & RESEARCH NETWORK

Juju Charms per il deployment e la gestione di servizi (e come svilupparli)

Claudio Pisa – GARR – Dipartimento di Calcolo e Storage Distribuito



Roma, 8-10 ottobre 2019

Workshop GARR

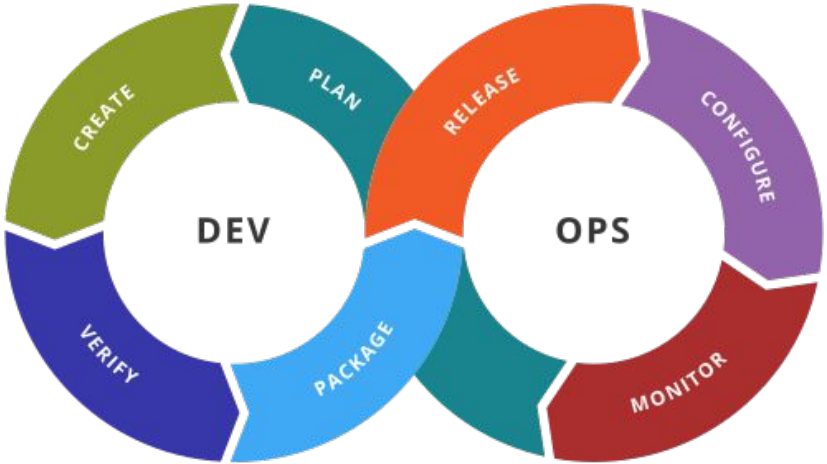
GARR Distributed Computing and Storage Department

- Role:
  - provider of resources (“long tail of science”)
  - resource aggregator (federation)
  - embody a replicable model for storage and computing provisioning

- Goals:
  - harmonize (SSO / federation of resources)
  - build secure and open infrastructures
  - enhance the user experience

GARR CSD - Guidelines

- Automation and Replicability
  - Infrastructure as Code
    - self-deployment
  - Open documentation
  - Continuous integration
  - Monitoring and self-healing
- Security and Privacy
  - GDPR
  - ISO 27001



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)

GARR Cloud Platform

- Based on **OpenStack**
  - cloud OS for data centers
- Features:
  - IDEM authentication
  - Ceph based storage
    - replicated and erasure-coded
- Users:
  - Single Users
  - Virtual Data Centers
    - administrative delegation


OpenStack Overview
Last 3 hours
CT1 →
Allocated VCPUs CT1
2252
Allocated RAM CT1
5.46 TB
Taken Public IPs CT1
594
Active instances CT1
684
Volume Usage CT1
342 TB
Active compute hosts CT1
24
PA1 →
Allocated VCPUs PA1
1819
Allocated RAM PA1
4.03 TB
Taken Public IPs PA1
506
Active instances PA1
561
Volume Usage PA1
191 TB
Active compute hosts PA1
16
NA2 →
Allocated VCPUs NA2
276
Allocated RAM NA2
546 GB
Taken Public IPs NA2
43
Active instances NA2
34
Volume Usage NA2
1.925 TB
Active compute hosts NA2
3
Active users
614
Active Projects
935
Demo projects
602
vDCs
47
Active images
495
Floating IPs
CT1 active floating IPs Current: 487
CT1 inactive floating IPs Current: 50
PA1 active floating IPs Current: 439
PA1 inactive floating IPs Current: 16
NA2 active floating IPs Current: 34
Active Instances
CT1 Current: 684
PA1 Current: 562
NA2 Current: 34
Active volumes
CT1 total volumes Current: 773
PA1 total volumes Current: 610
CT1 in use volumes Current: 558
PA1 in use volumes Current: 461
CT1 error volumes Current: 0
PA1 error volumes Current: 3
Total RAM Usage (GiB)
CT1 Current: 5.34 K
PA1 Current: 3.93 K
NA2 Current: 533
Total volume usage (GiB)
CT1 Current: 342 K
PA1 Current: 191 K
NA2 Current: 2 K
Flavor Distribution CT1
m1.small Current: 128
m1.large Current: 64
m1.xxl Current: 6
c2.large Current: 3
m1.xlarge Current: 18
unipa.largemem Current: 58
Flavor Distribution PA1
m1.small Current: 82
m1.large Current: 85
m1.xxl Current: 19
c2.large Current: 4
m1.xlarge Current: 28
idem-idpcloud Current: 33

GARR Container Platform

- Environment for automating deployment, scaling, and management of containerized applications
  - based on **Kubernetes**
- Cluster
  - Deployed on bare metal
  - Provides GPU resources
  - Ceph based storage
- Helm support
- Multitenant
  - Unified authentication with OpenStack
    - result of GN4-2
    - contributed upstream

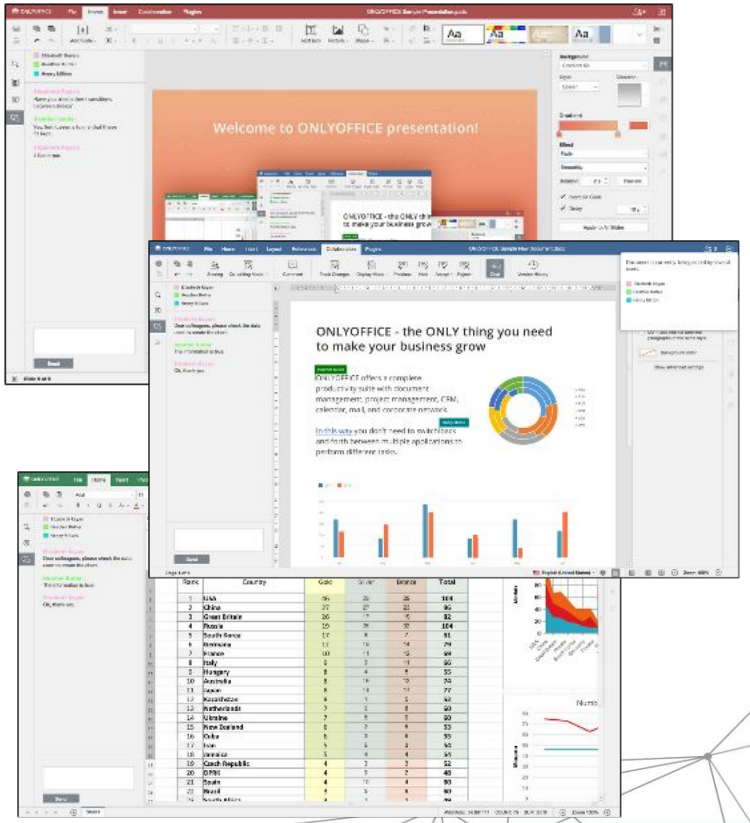
DaaS (Deployment as a Service)

- A graphical user interface to install applications on the GARR Cloud Platform
  - based on Juju
  - application catalogue at https://jaas.ai/
- https://cloud.garr.it/apps/

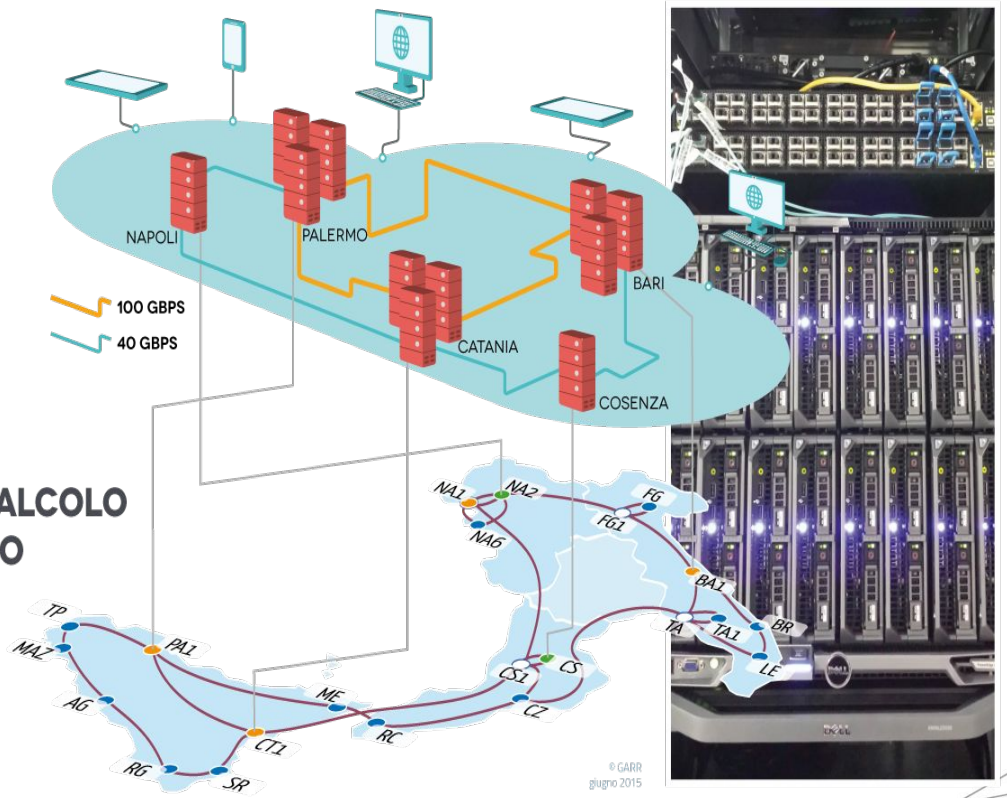
https://daas-pa.cloud.garr.it
https://daas-ct.cloud.garr.it

GARR Workplace

- A web-based office and collaborative suite hosted on the GARR cloud
  - based on OnlyOffice
    - community and commercial licenses
- Features:
  - Document editor
    - highly compatible with popular document formats
  - Collaborative tools
    - e-mail, calendar, wiki, chat, etc.
  - CRM
  - Project management
  - Pluggable cloud storage
    - e.g. GARRBox



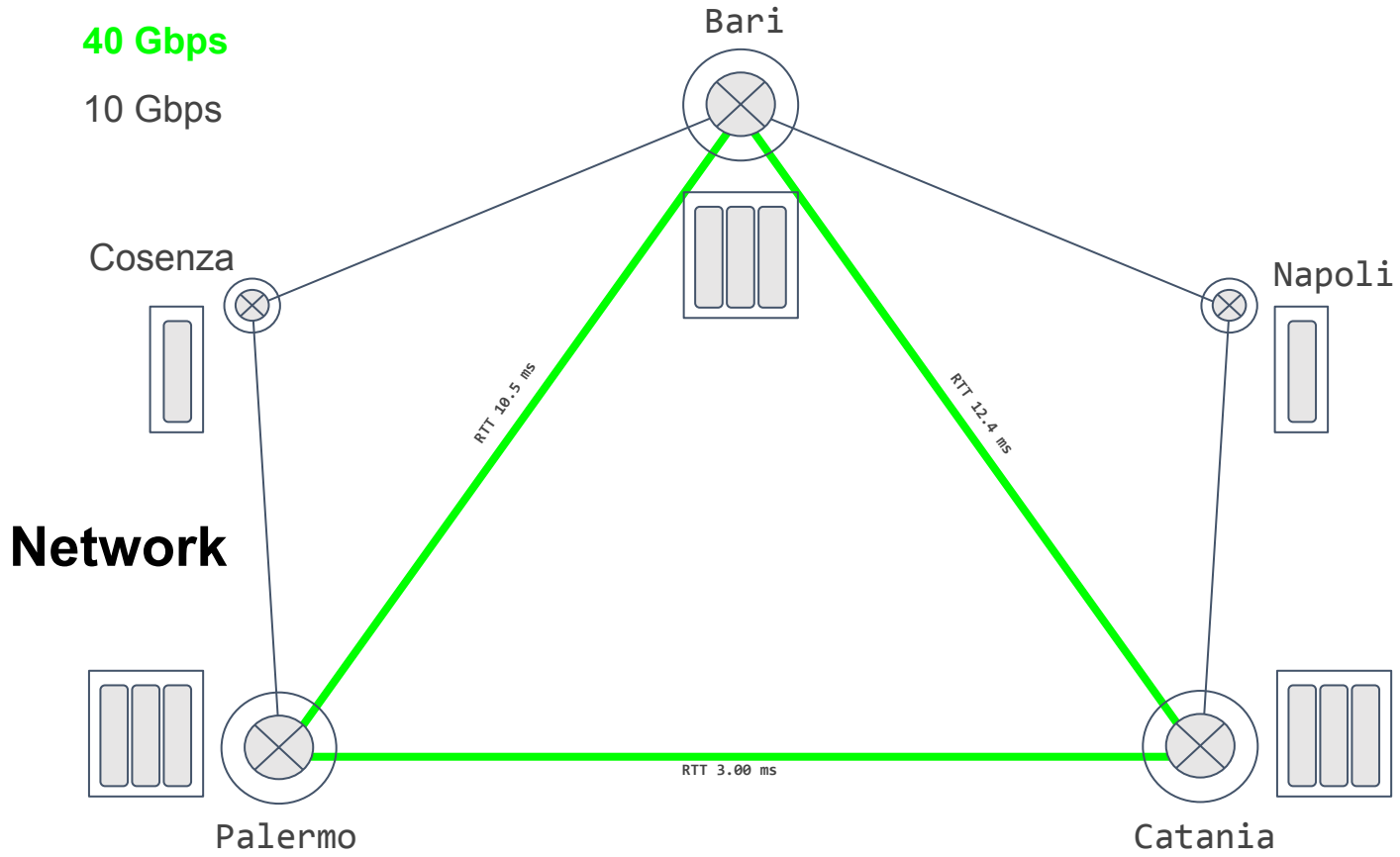
GARR Computing and Storage Infrastructure

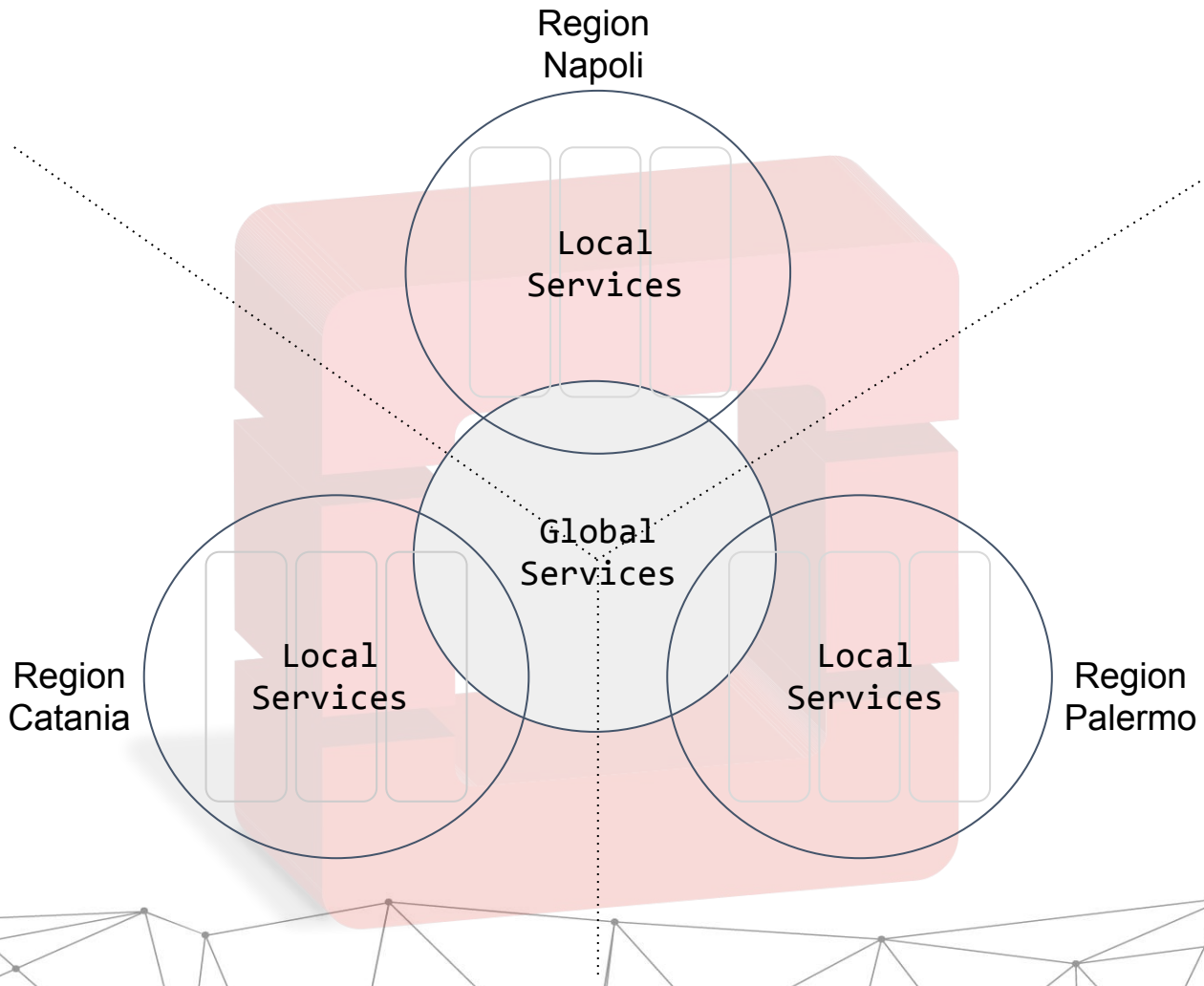


INFRASTRUTTURA DI CALCOLO E STORAGE DISTRIBUITO

- 5 siti distribuiti
- 8.448 virtual CPU
- 10 PB spazio storage

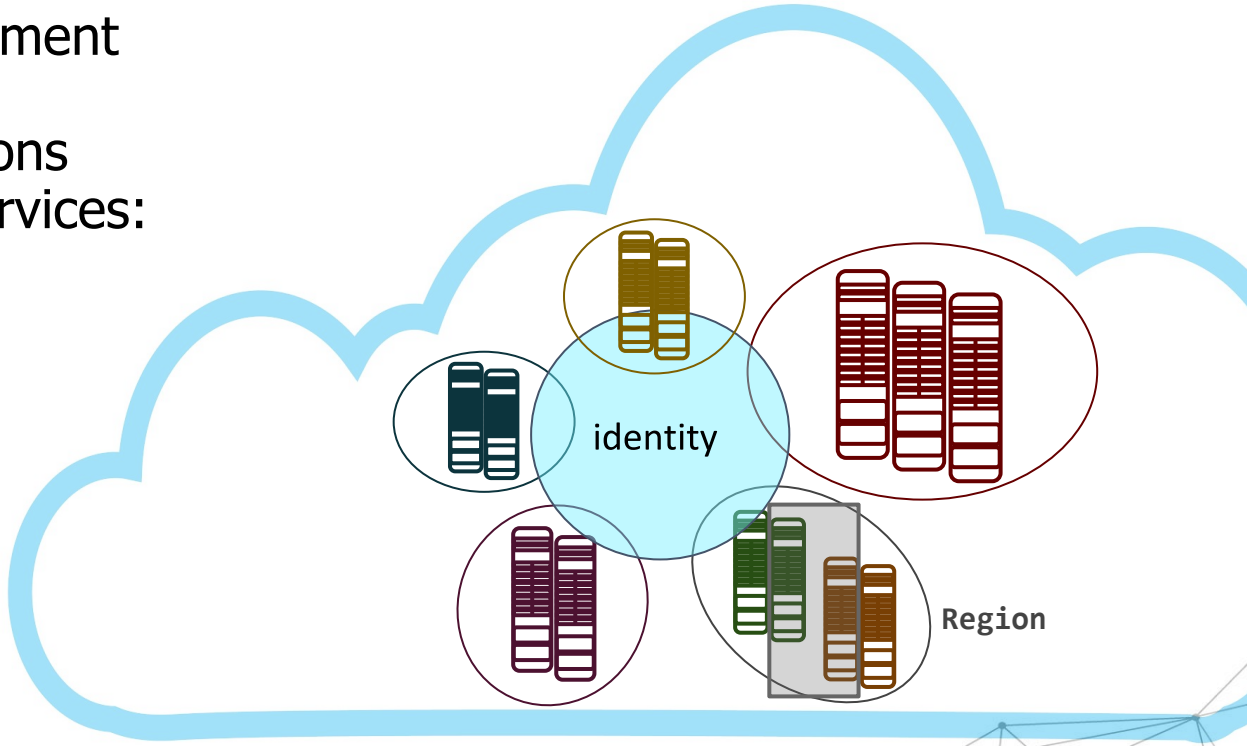
Network


40 Gbps
10 Gbps
Bari
Cosenza
Napoli
RTT 10.5 ms
RTT 12.4 ms
RTT 3.00 ms
Palermo
Catania


Region
Napoli
dns
Local
Services
Global
Services
Region
Catania
Local
Services
Local
Services
Region
Palermo

Multi-region/multi-domain Federation Model

- Region: separate deployment of OpenStack
  - Linked to other regions through common services:
    - identity service
    - dashboard
    - image service



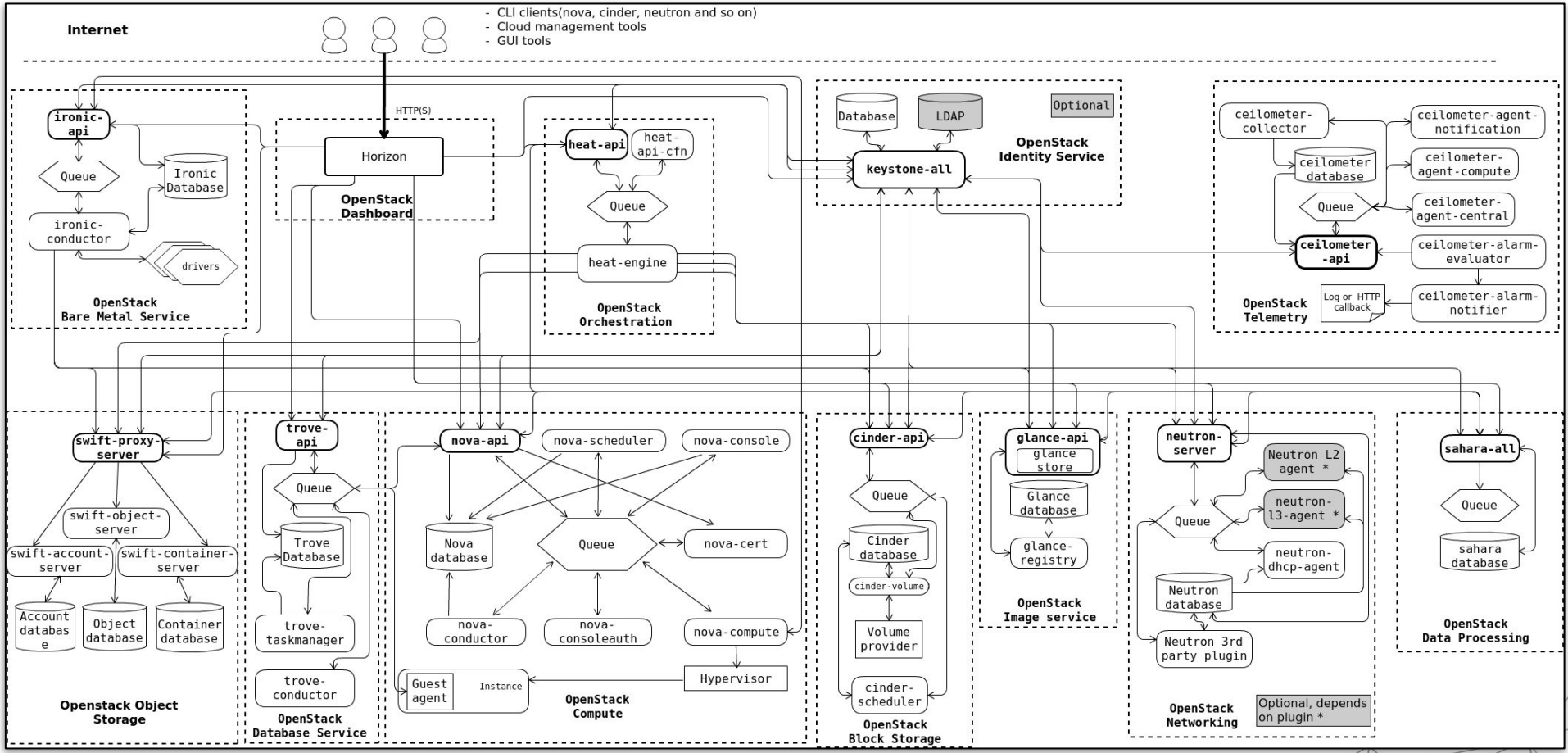
GARR Cloud Federation

- Simple Federation recipe
  - Reference architecture with use cases
  - git repository and knowledge base
    - https://git.garr.it/cloud/federation/
    - https://cloud.garr.it/doc/federation/
  - Federated authentication
  - Administrative delegation
    - work in (advanced) progress

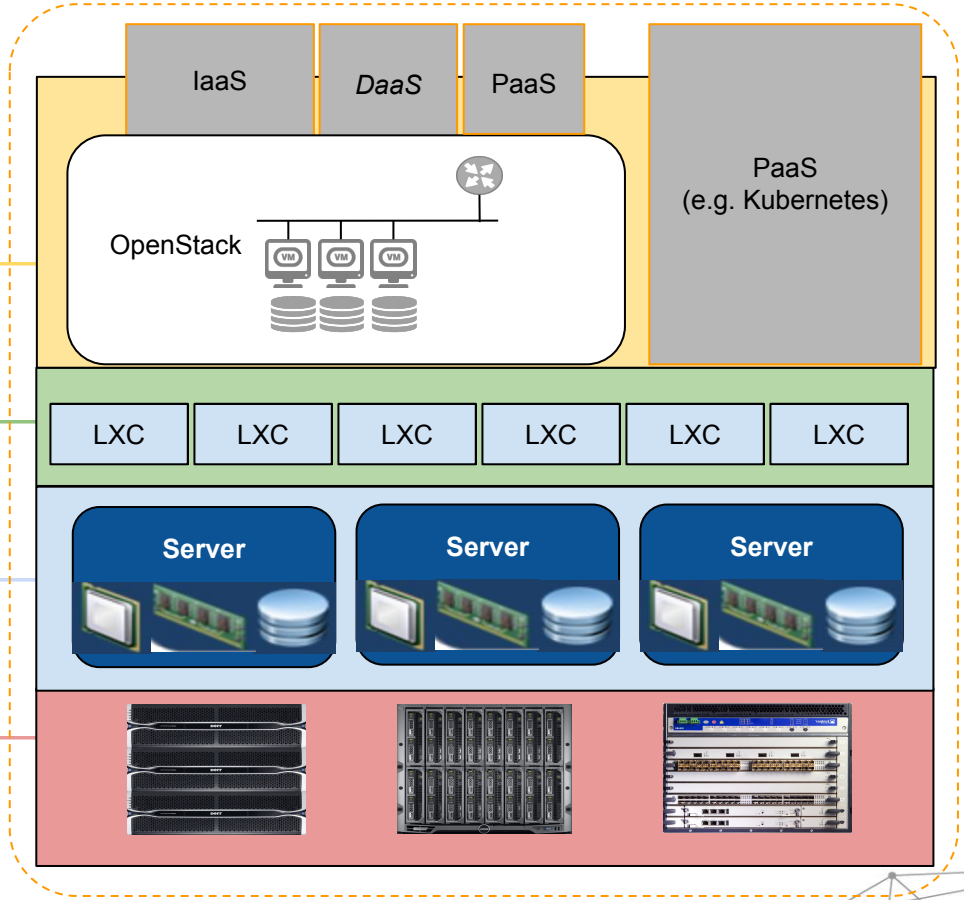
- Our recipe is working: join us or just copy us!
  - UniTo, PoliTo, UniPD (federating)
  - 3 INGV regions (external federation)
  - HPC4AI project
  - EAPConnect (East EU Nren, external federation)
  - Hungary
  - ...

Open Infrastructures

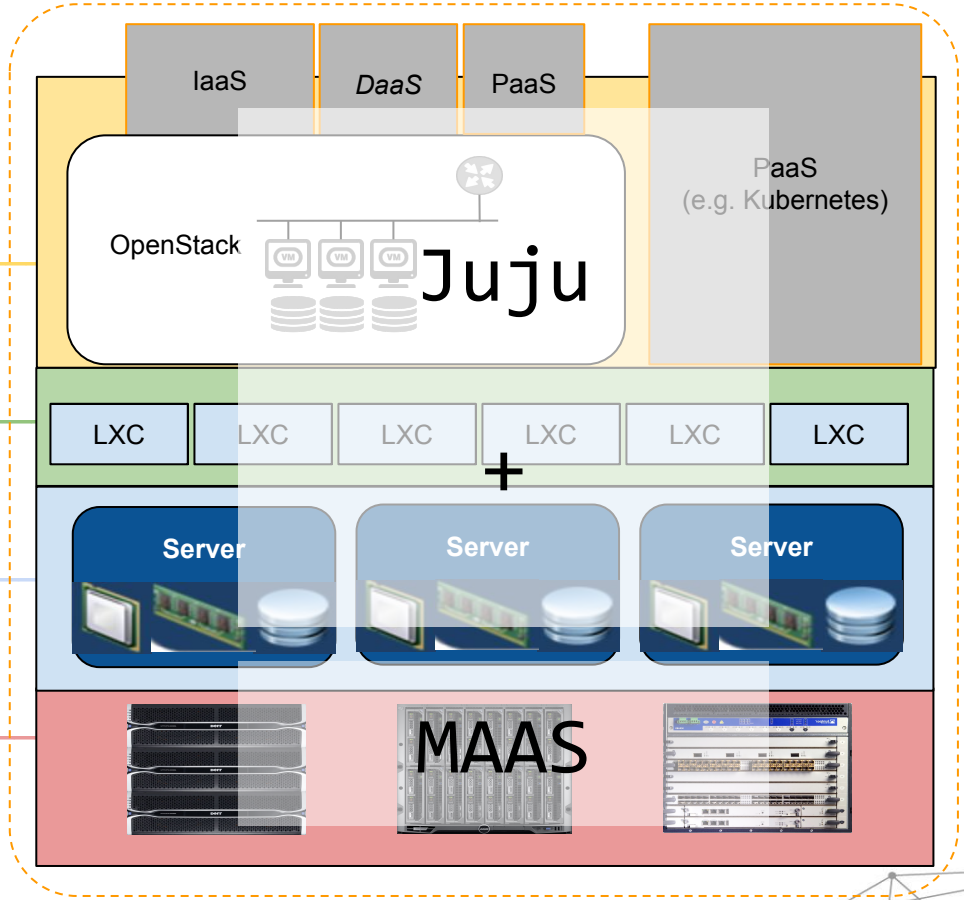
- Open Documentation
  - replicable model
  - Infrastructure as (open source) Code
- Open Federation
  - members of the GARR community can federate their computing resources
- Open Source / Free Software contributions:
  - Horizon
  - k8s-keystone-auth
  - Juju charms: ceph, keystone saml/sso, default gw, moodle…


Internet
- CLI clients(nova, cinder, neutron and so on)
- Cloud management tools
- GUI tools
HTTP(S)
ironic-api
Queue
Ironic Database
ironic-conductor
drivers
OpenStack Bare Metal Service
Horizon
OpenStack Dashboard
heat-api
heat-api-cfn
Queue
heat-engine
OpenStack Orchestration
Database
LDAP
Optional
keystone-all
OpenStack Identity Service
ceilometer-collector
ceilometer-agent-notification
ceilometer database
ceilometer-agent-compute
Queue
ceilometer-agent-central
ceilometer-api
ceilometer-alarm-evaluator
Log or HTTP callback
ceilometer-alarm-notifier
OpenStack Telemetry
swift-proxy-server
swift-object-server
swift-account-server
swift-container-server
Account database
Object database
Container database
Openstack Object Storage
trove-api
Queue
Trove Database
trove-taskmanager
trove-conductor
OpenStack Database Service
nova-api
nova-scheduler
nova-console
Nova database
Queue
nova-cert
nova-conductor
nova-consoleauth
nova-compute
Guest agent
Instance
Hypervisor
OpenStack Compute
cinder-api
Queue
Cinder database
cinder-volume
Volume provider
cinder-scheduler
OpenStack Block Storage
glance-api
glance store
Glance database
glance-registry
OpenStack Image service
neutron-server
Neutron L2 agent *
Queue
neutron-l3-agent *
neutron-dhcp-agent
Neutron database
Neutron 3rd party plugin
OpenStack Networking
Optional, depends on plugin *
sahara-all
Queue
sahara database
OpenStack Data Processing

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

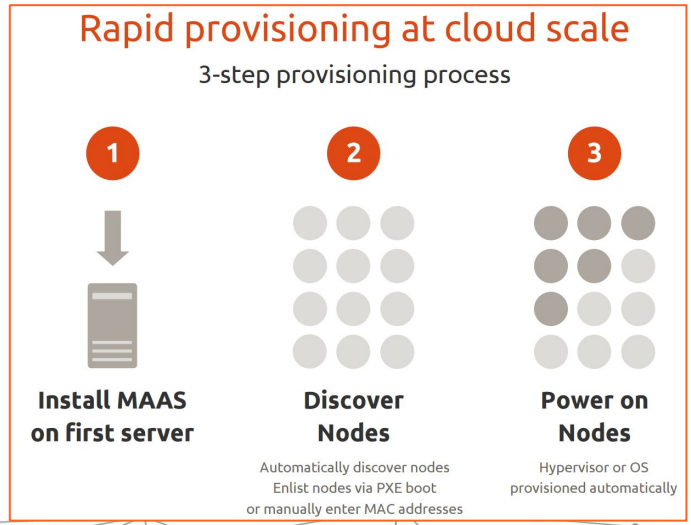


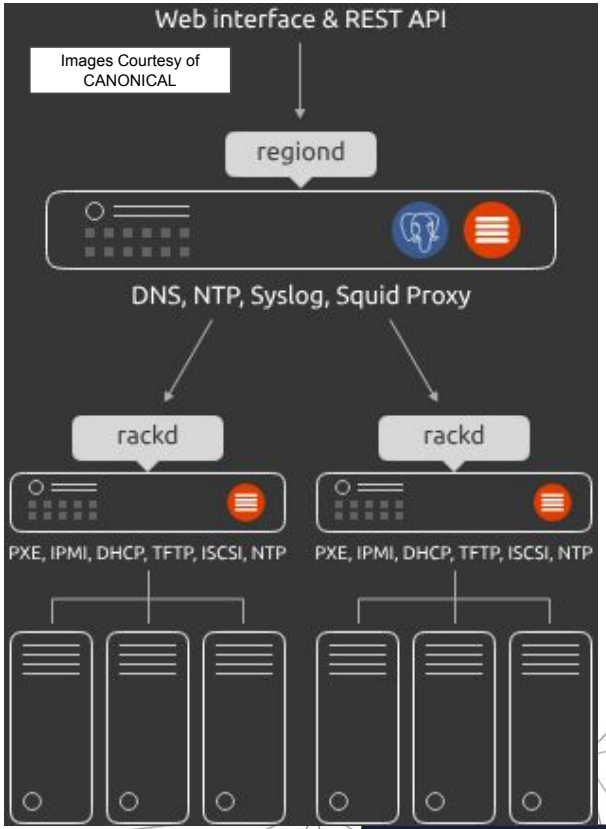
1. Application Services
2. Infrastructure *Virtualization*
3. Operating System
4. Physical resources



Metal As A Service (MAAS)

- Discovers and deploys physical servers
- Allocates resources to match requirements
- Undeploys servers when they are no longer needed
- Allows cross datacenter provisioning

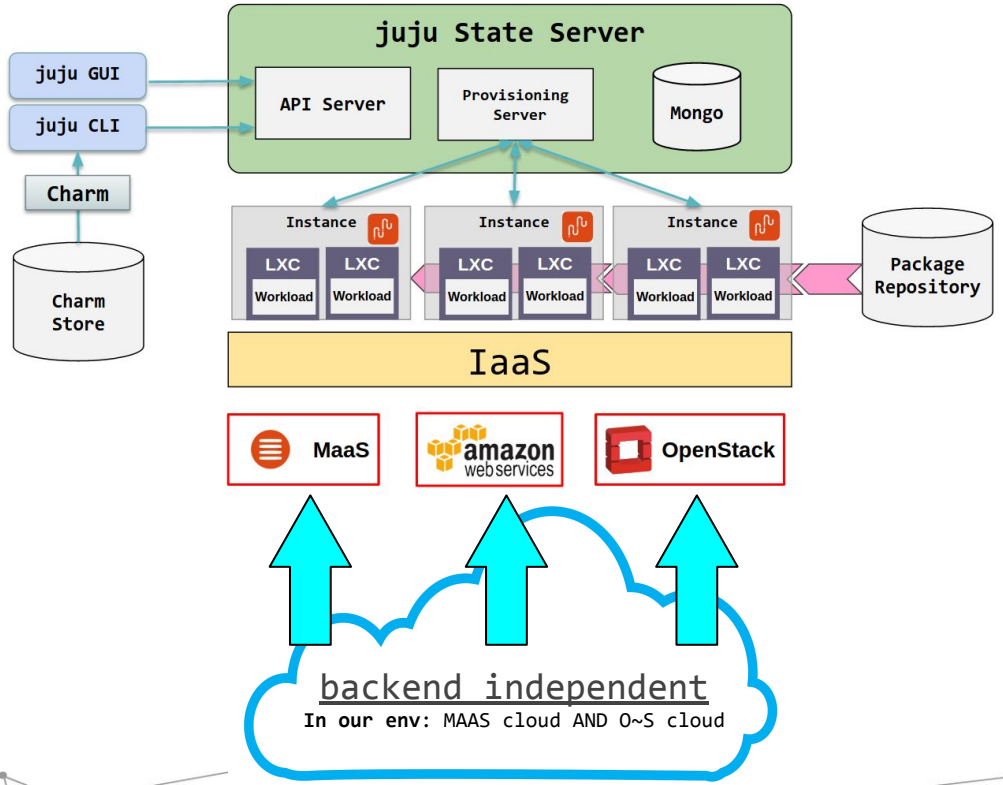




Juju

- Deployment, configuration and management cloud-native tool
- Free and open source
- Exposes a high-level declarative language
- Supports different backends:
  - OpenStack
  - Kubernetes
  - Amazon AWS
  - Microsoft Azure
  - Google CGE
  - VMWare
  - LXD
  - MAAS

Juju Architecture



Juju Charms

- Charm
  - deployment "brick" / microservice
  - a collection of scripts called *hooks* + metadata
  - can expose configurable parameters
  - several charms available online in the charm store

- Hook
  - executable file with a pre-defined name which defines how to handle an event
    - install, start, config-changed, ...
  - can be written in any scripting/programming language

Juju Charms

- Charm relations
  - establish a logical connection between services
  - in practice: configuration parameters are exchanged between charms

- Bundle
  - a YAML file which describes charms, their configurations and their relations

Charm simple example

- Writing a charm can be simple
- Example: wordpress
  - but could be your web application
- Example charm structure:
  - `metadata.yaml`
  - `hooks/install`
  - `hooks/start`
  - `hooks/database-relation-changed`
  - `hooks/website-relation-joined`

metadata.yaml 273 Bytes

Edit | Web IDE | Replace | Delete

```yaml
name: wordpress
summary: The wordpress CMS
maintainer: GARR CSD <cloud-support@garr.it>
description: The Wordpress Content Management System
tags: [cms]
subordinate: false
series: [bionic]
provides:
  website:
    interface: http
requires:
  database:
    interface: mysql
```

install 698 Bytes

Edit | Web IDE | Replace | Delete

```
#!/bin/bash

status-set maintenance "Installing wordpress"
apt-get install -y wordpress

status-set maintenance "Configuring wordpress"
echo "
        Alias /blog /usr/share/wordpress
        <Directory /usr/share/wordpress>
            Options FollowSymLinks
            AllowOverride Limit Options FileInfo
            DirectoryIndex index.php
            Order allow,deny
            Allow from all
        </Directory>
        <Directory /usr/share/wordpress/wp-content>
            Options FollowSymLinks
            Order allow,deny
            Allow from all
        </Directory>" > /etc/apache2/sites-available/wordpress.conf
a2ensite wordpress
status-set maintenance "Wordpress installed"
```

start 131 Bytes

Edit | Web IDE | Replace | Delete

```
#!/bin/bash
status-set maintenance "Starting wordpress"
service apache2 restart
status-set blocked "Waiting for database relation"
```

database-relation-changed 557 Bytes

Edit | Web IDE | Replace | Delete

```
#!/bin/bash
database=`relation-get database`

[ -z "$database" ] && exit 0

# retrieve the remaining values
# which have been set by the mysql charm
user=`relation-get user`
password=`relation-get password`
host=`relation-get private-address`

# write configuration file
echo "<?php
define('DB_NAME', '${database}');
define('DB_USER', '${user}');
define('DB_PASSWORD', '${password}');
define('DB_HOST', '${host}');
define('WP_CONTENT_DIR', '/usr/share/wordpress/wp-content');
?>" >/etc/wordpress/config-default.php

open-port 80
status-set active "Ready"

```

website-relation-joined 89 Bytes

Edit | Web IDE | Replace | Delete

```
#!/bin/bash

relation-set "hostname=$(unit-get private-address)"
relation-set "port=80"

```

Deploying a cloud application with Juju

```
# deploy a local web application and a database from the charm store
juju deploy ./wordpress
juju deploy mysql

# and connect them
juju add-relation wordpress mysql
```


/blog/
IL MIO BELLISSIMO BLOG
Just another WordPress site

Cloud applications and high availability

- Deploying applications on the cloud introduces a paradigm shift:
  - any component of the cloud infrastructure can fail at any moment
    - as opposed to expensive, redundant hardware
  - high availability has to be moved to higher layers of the stack

"no single component can guarantee 100% uptime (and even the most expensive hardware eventually fails)" - Netflix technology blog

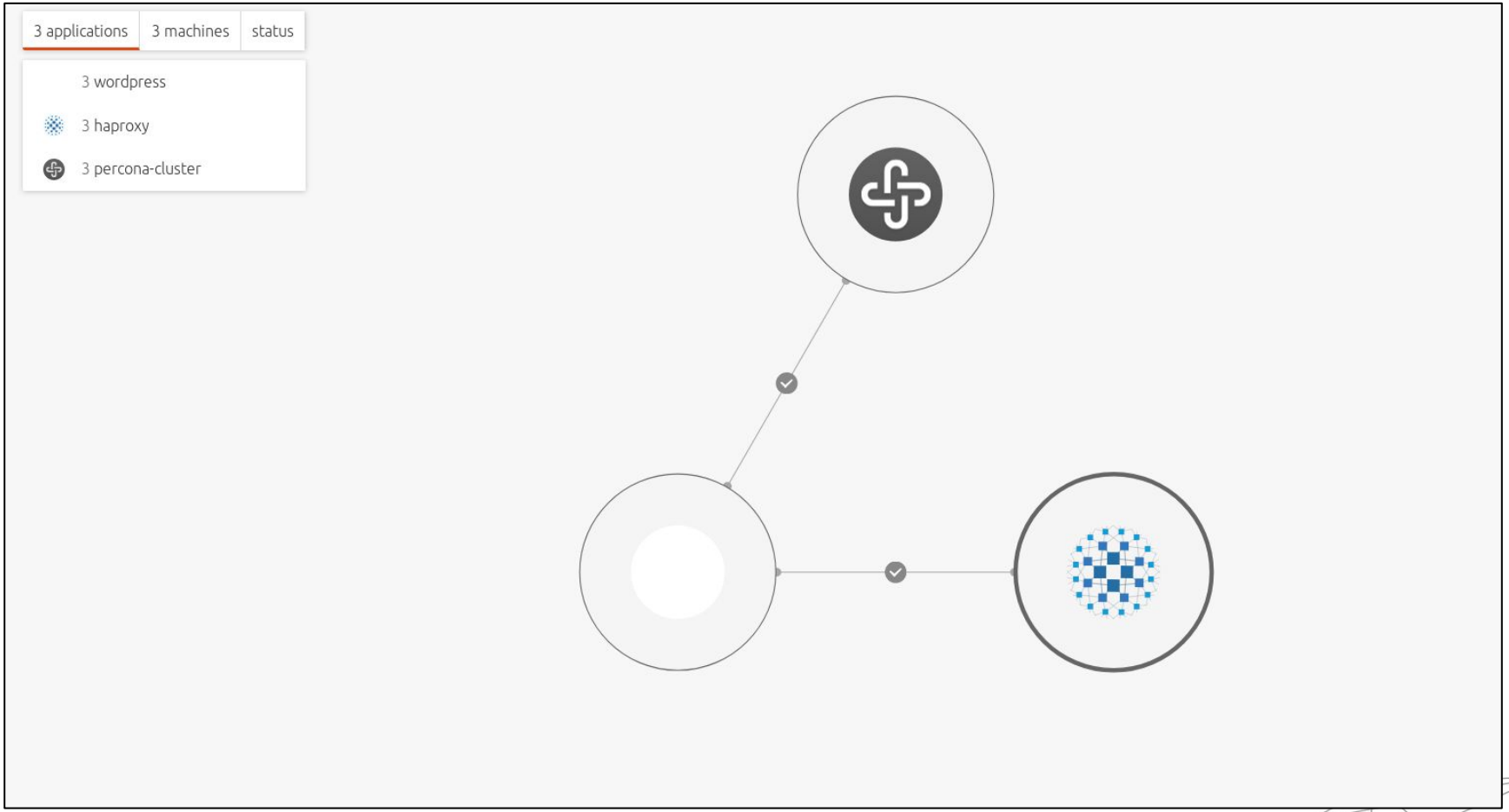
Chaos Engineering - Chaos Monkey

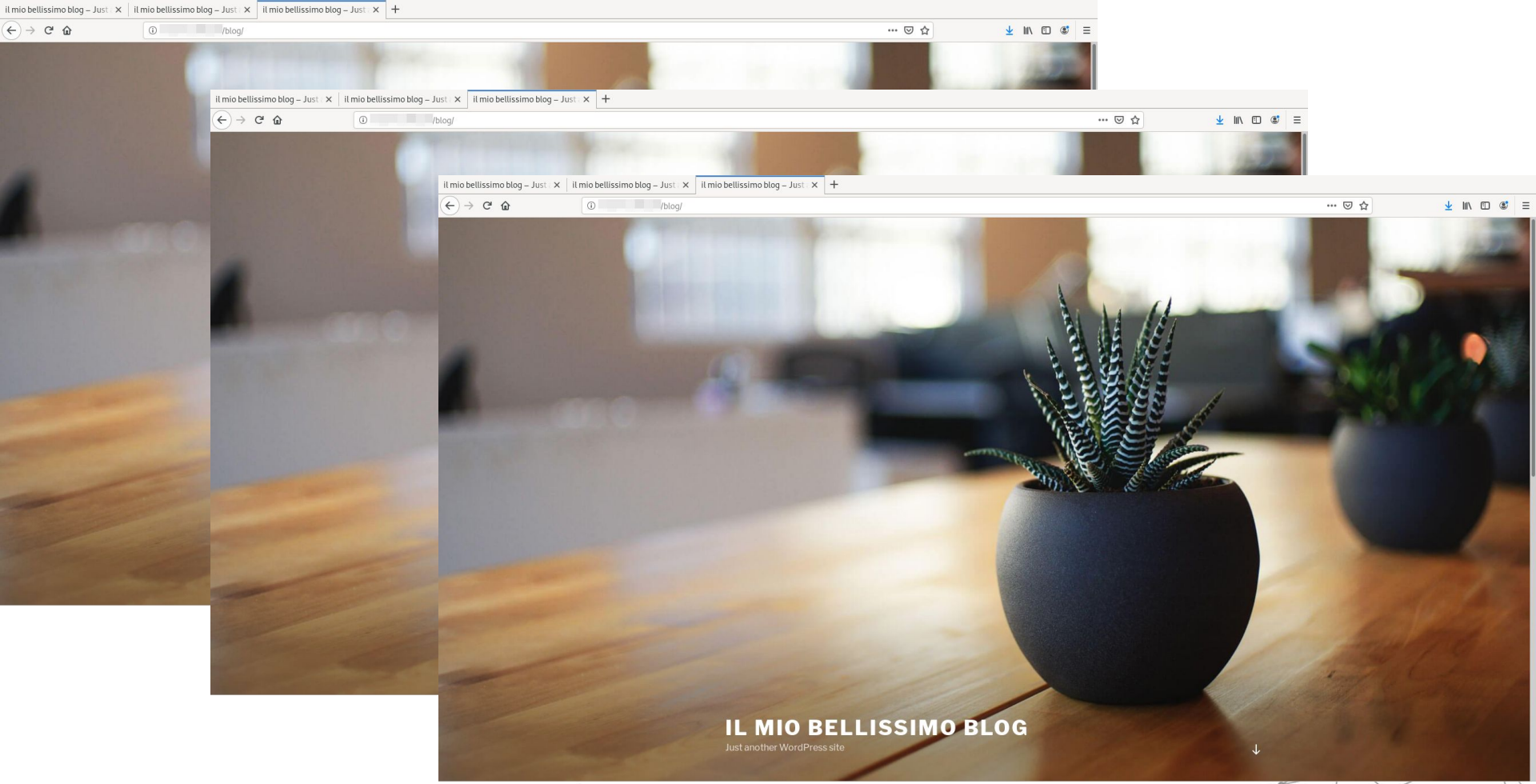
- Chaos Monkey randomly terminates virtual machine instances and containers that run inside of your production environment
- The name comes from the idea of unleashing a wild monkey with a weapon in your data center (or cloud region) to randomly shoot down instances and chew through cables — all the while we continue serving customers without interruption
- Exposing engineers to failures more frequently incentivizes them to build resilient services

hawp_bundle.yaml 529 Bytes

Edit | Web IDE | Replace | Delete

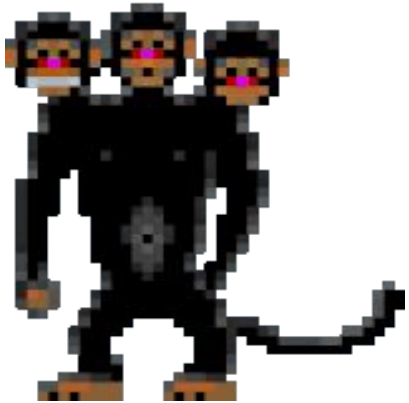
```yaml
series: bionic
applications:
  wordpress:
    charm: "/home/ubuntu/wordpress"
    num_units: 3
    to: [ "0", "1", "2" ]
  percona-cluster:
    charm: cs:percona-cluster-279
    num_units: 3
    to: [ "3", "4", "5" ]
  haproxy:
    charm: cs:haproxy
    num_units: 3
    to: [ "3", "4", "5" ]
    expose: true
    options:
      peering_mode: active-active
machines:
  "0": {}
  "1": {}
  "2": {}
  "3": {}
  "4": {}
  "5": {}
relations:
- [ wordpress:database, percona-cluster:db ]
- [ wordpress:website, haproxy:reverseproxy ]
```


3 applications
3 machines
status
3 wordpress
3 haproxy
3 percona-cluster


il mio bellissimo blog – Just
il mio bellissimo blog – Just
il mio bellissimo blog – Just
/blog/
IL MIO BELLISSIMO BLOG
Just another WordPress site

Wrap up

- we wrote a simple charm for our web application
- we deployed our application on a cloud infrastructure
- we used Juju to compose our charm with other charms to obtain a scalable, highly available application
  - monkey-proof



Takeaways

- We can use Juju in several ways:
  - to deploy and manage cloud infrastructures
  - to deploy and manage scalable, highly-available applications
  - to make our own applications scalable and highly-available
- Juju is cloud agnostic
  - the same tool for private, community and public clouds
  - the same tool for physical hardware up to pods
- Note that Juju can use other configuration management systems (e.g. Ansible) for its hooks
- Juju limitations:
  - needs an agent to be installed (i.e. it is not OS independent)
  - does not manage networking appliances
  - needs scripting/programming skills to implement complex charms

https://www.hackthecloud.it/

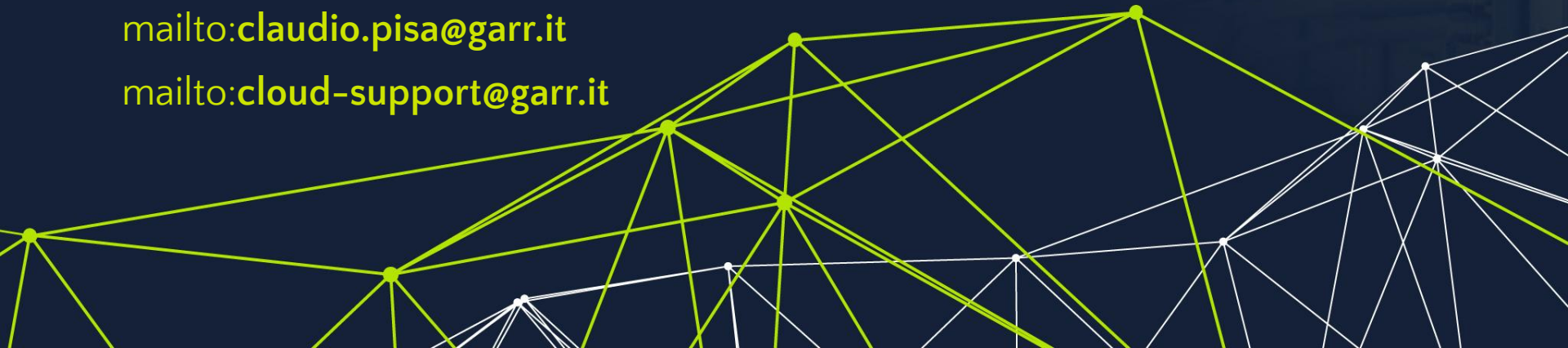
Consortium GARR
THE ITALIAN EDUCATION & RESEARCH NETWORK

Grazie

https://**cloud.garr.it**

mailto:**claudio.pisa@garr.it**

mailto:**cloud-support@garr.it**



Instance All | Namespace kube-system

all pods

Memory Working Set

Last 1 minute

Used	Total
49.17 GiB	1.60 TiB

Cpu Usage

Last 1 minute

Used	Total
3.29 cores	224.00 cores

Filesystem Usage

Last 1 minute

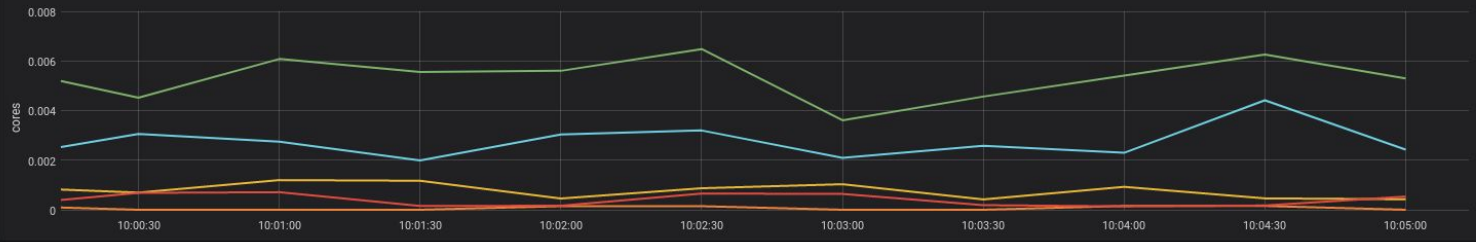
Used	Total
529.09 GiB	2.39 TiB

Network

	avg	c
receive	13.70 kBs	11.
transmit	-1.87 kBs	-1.

each pod

Cpu Usage



	avg	current
kube-dns-8f7866879-w8qzh	0.005	0.005
metrics-server-v0.3.1-67b497cfd8-8jpdg	0.003	0.002
tiller-deploy-85686555b8-62drk	0.000	0.001
kubernetes-dashboard-57df4db6b-2kcgq	0.001	0.000
nvidia-device-plugin-daemonset-xvt45	0.000	0.000

GARR Cloud: the crew

Alex Barchiesi

Alberto Colla

Fulvio Galeazzi

Claudio Pisa

GARR Cloud: the crew

Alex Barchiesi

Alberto Colla

Fulvio Galeazzi

Claudio Pisa

New!

New!

New!

Matteo Di Fazio

Marco Lorini

Delia Passalacqua

Juju vs the rest of the world

vs ansible, terraform, puppet

Juju Layers

docker layer

ansible layer

GARR Computing and Storage

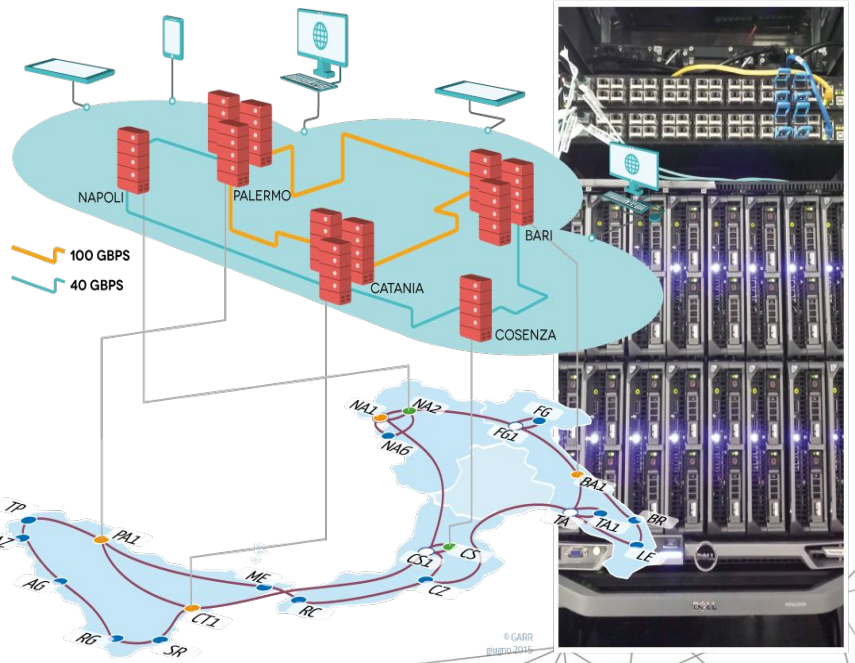
GARR cloud

The engine

MAAS JUJU ceph

600 Users
1100 VMs
3500 vCPUs 9 TB_RAM
1 PB_storage 900 IPv4

Container platform
(GPU inside)

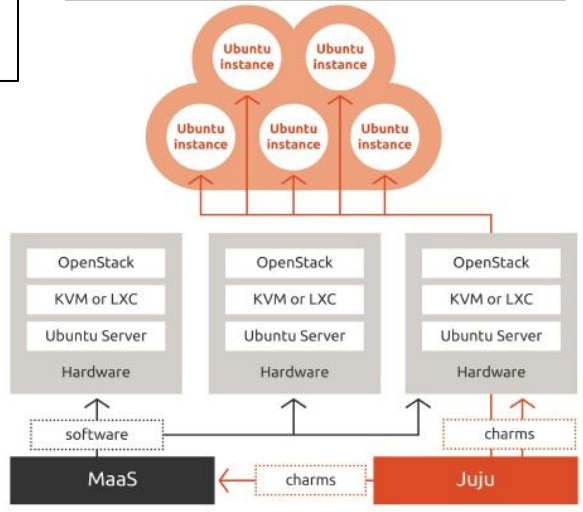


DaaS (deployment a.a s.)

more powerful than a PaaS, easier than a IaaS

Juju on OpenStack Cloud

backend



Frontend UI/CLI

=

DaaS

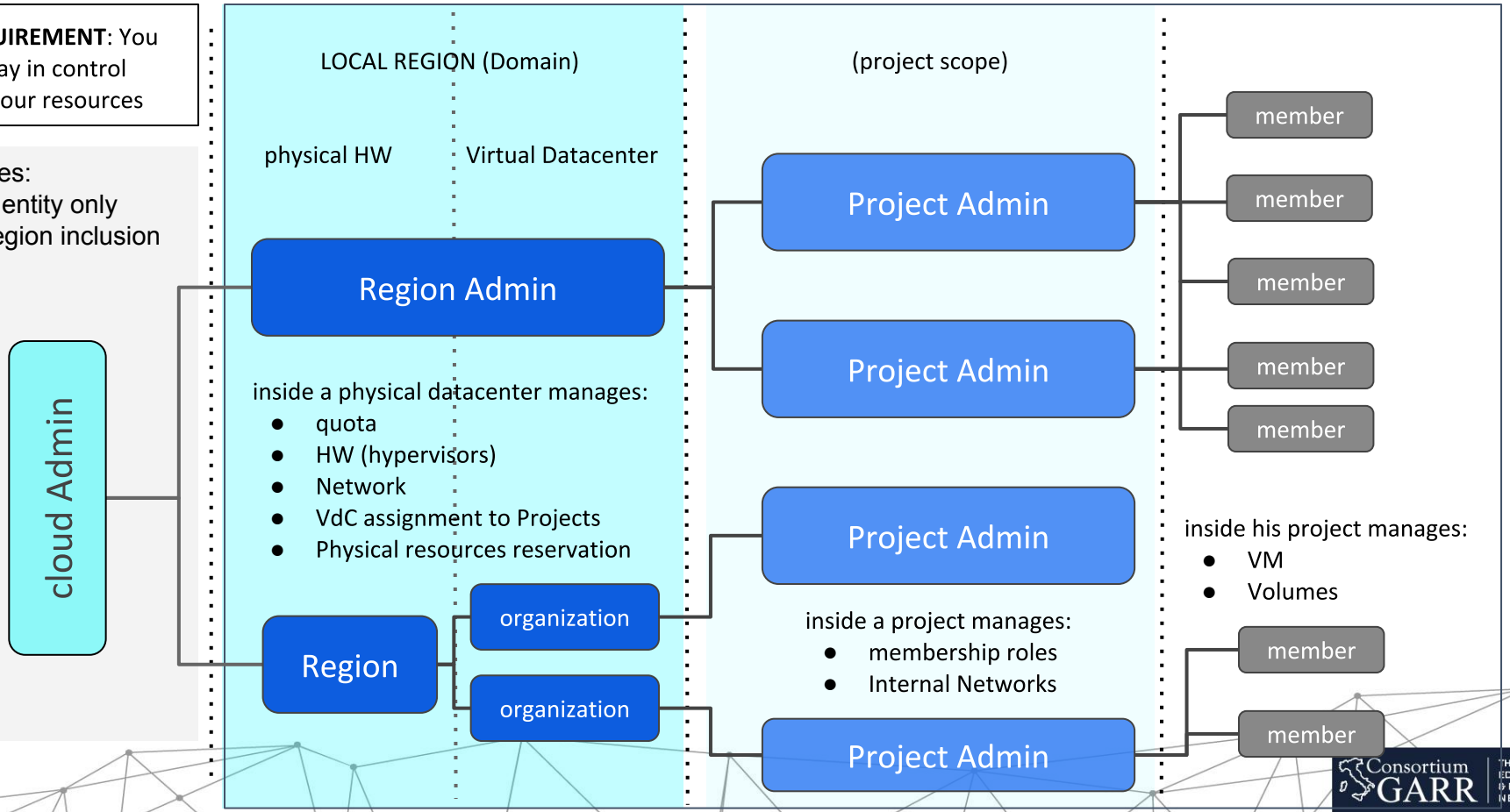
GARR Cloud

https://daas-pa.cloud.garr.it
https://daas-ct.cloud.garr.it

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

REQUIREMENT: You stay in control of your resources

manages:
- identity only
- region inclusion



(few) details about deployment *requirements*

networking prescriptions

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

2. public network
   - 2.1. Public ip (for infrastructure and Cloud service frontend)
     Needed minimum 40 IP (suggested to have a 3 members HA of each service: /25 subnet)
   - 2.2. 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)

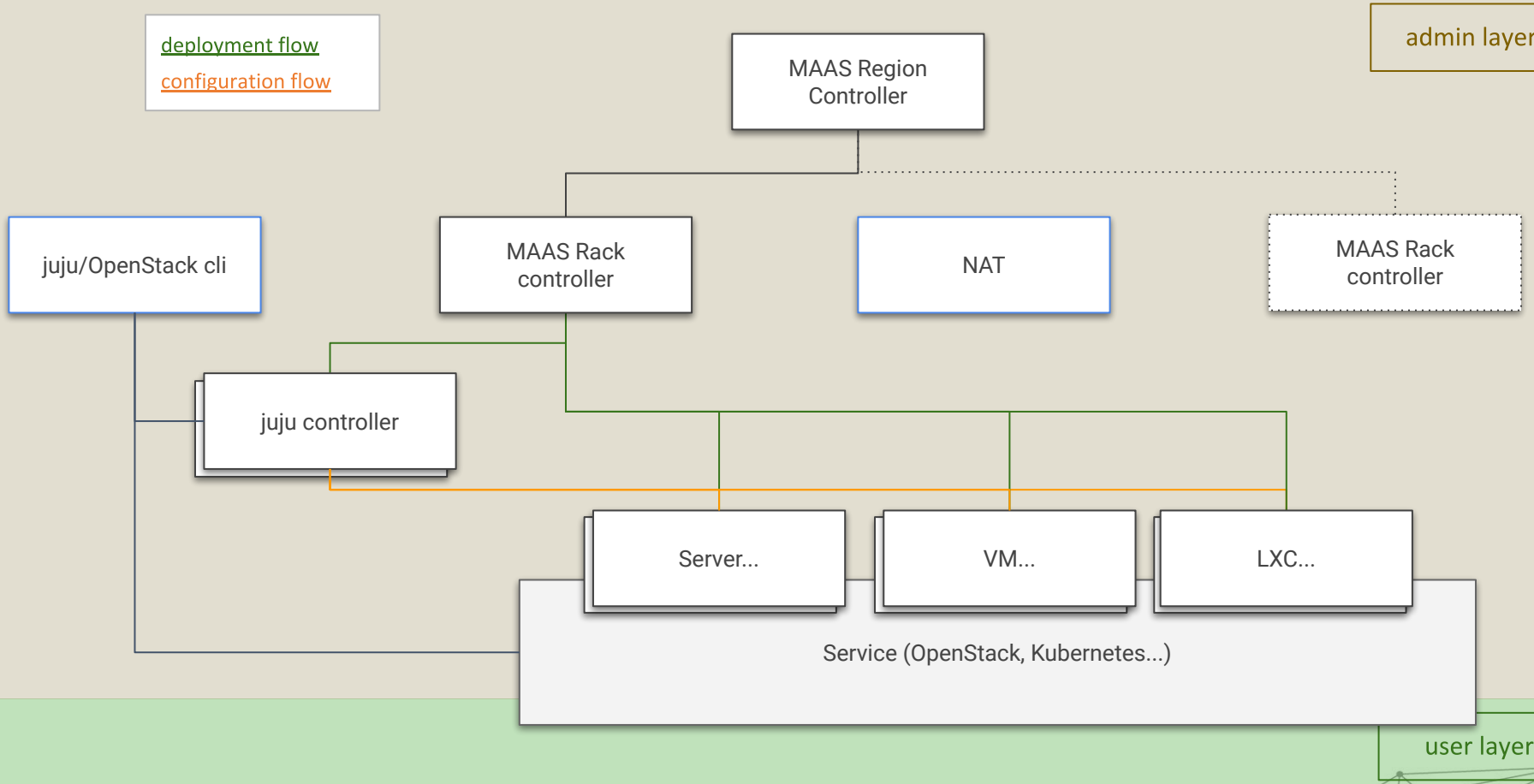
Best practice:

- 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)

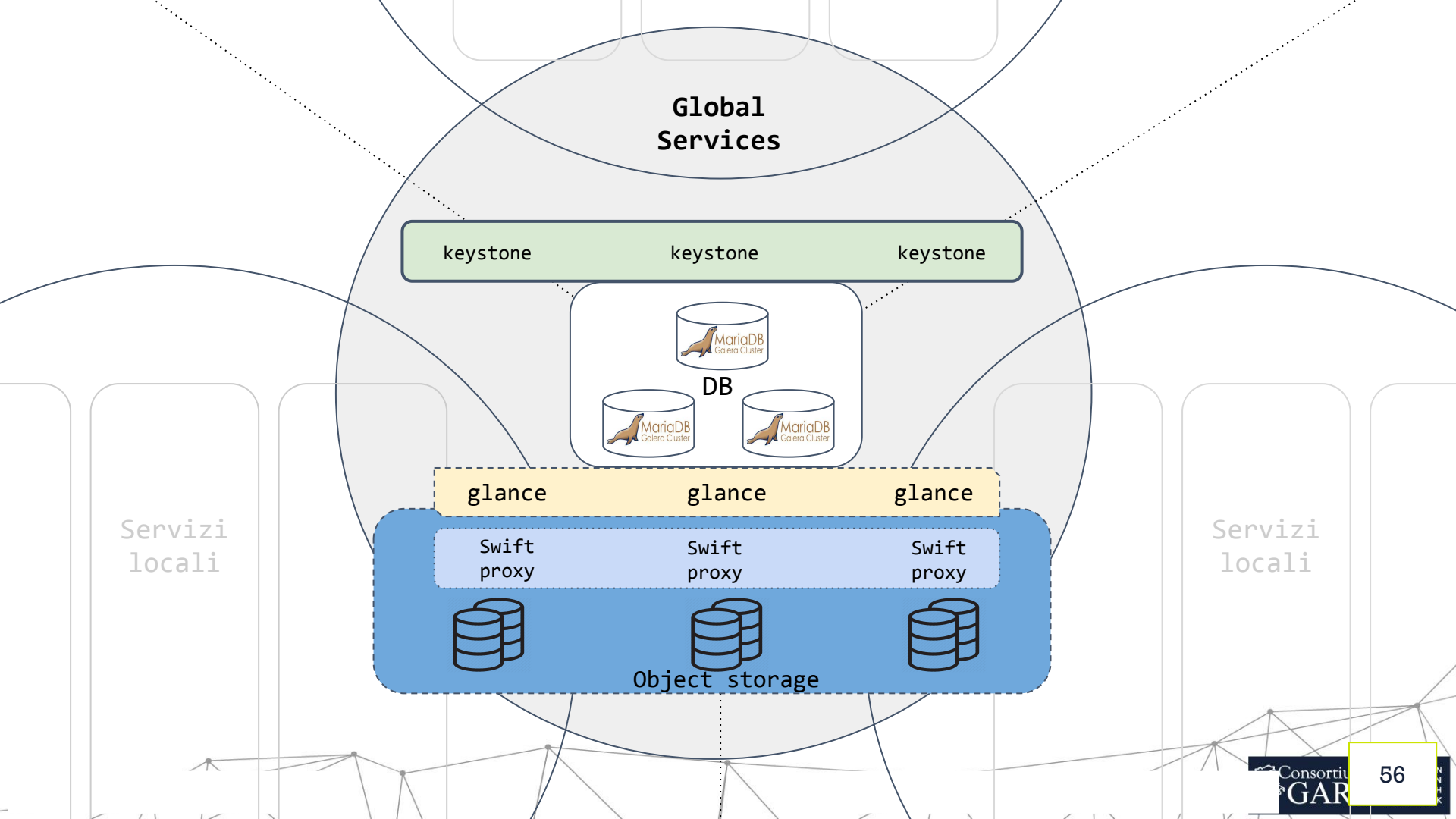


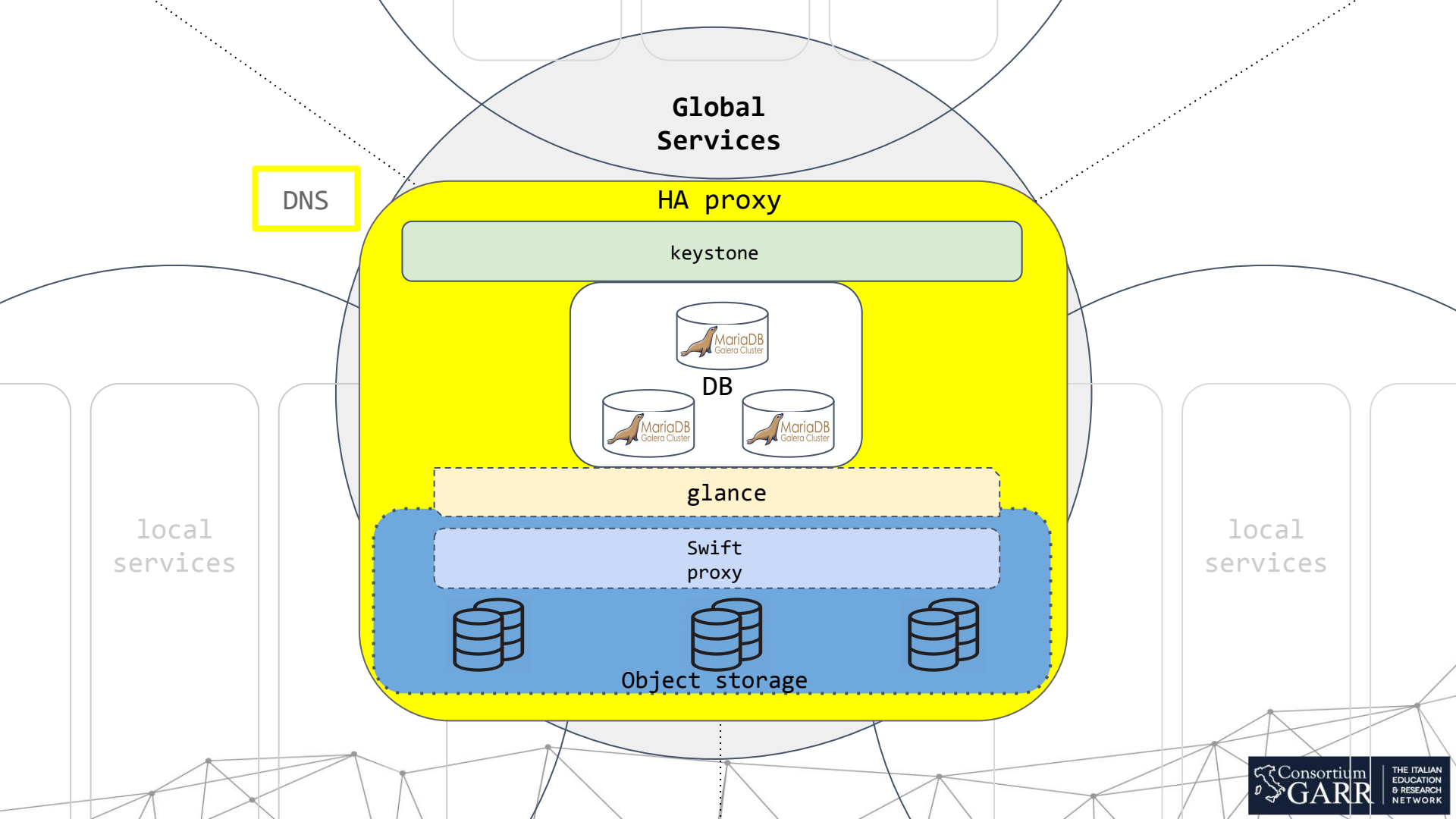
next steps

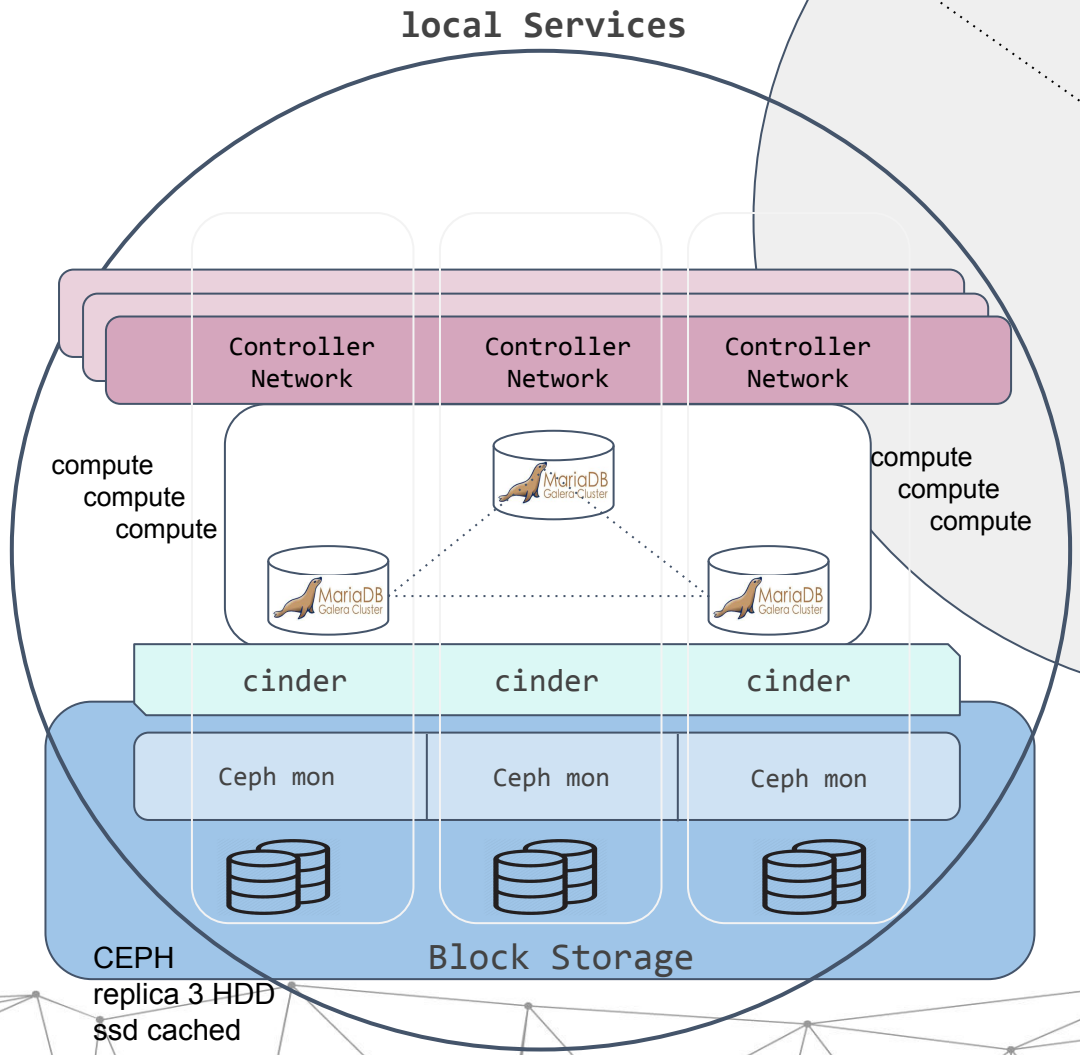
governance

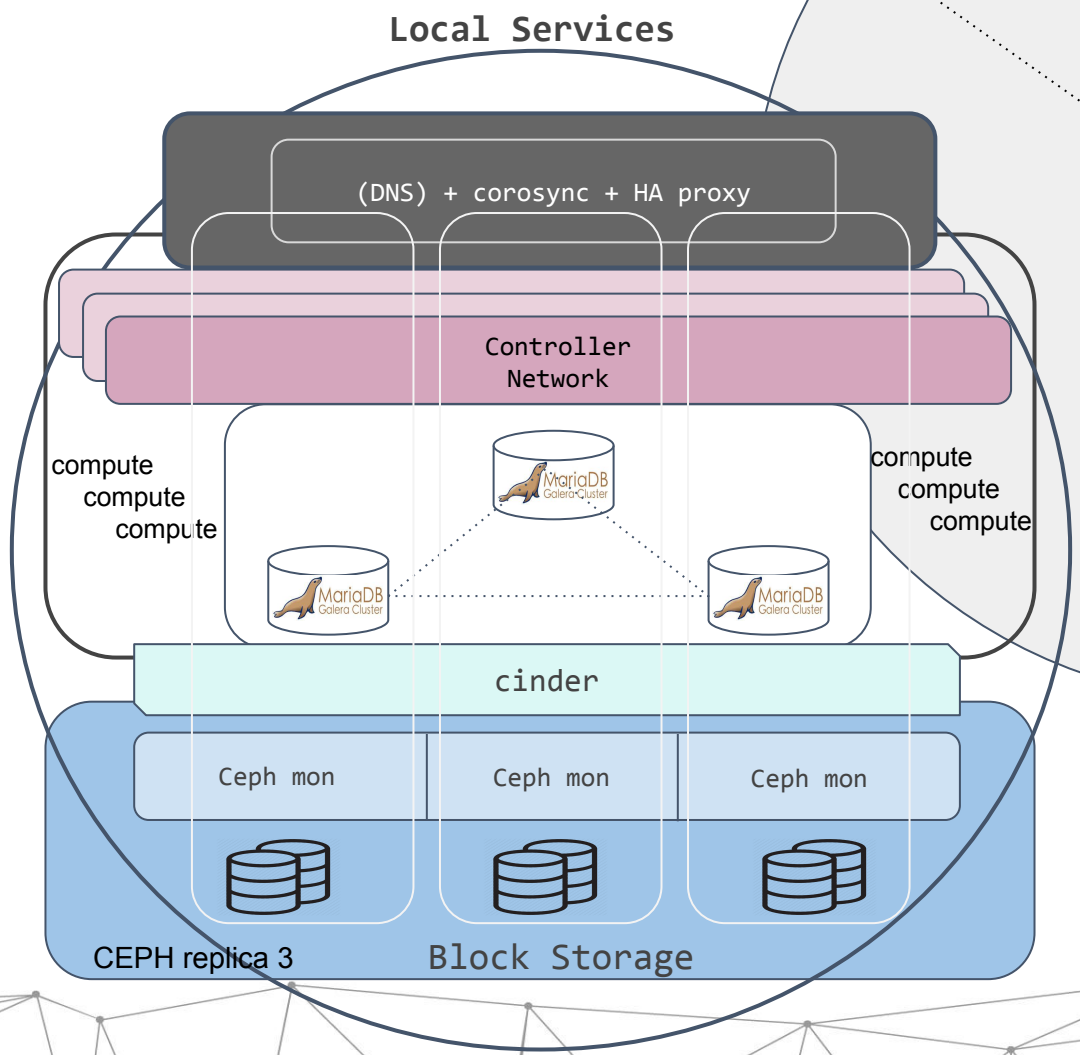
accounting (blockchain?)

federation of federations (keystone to keystone trust mesh)

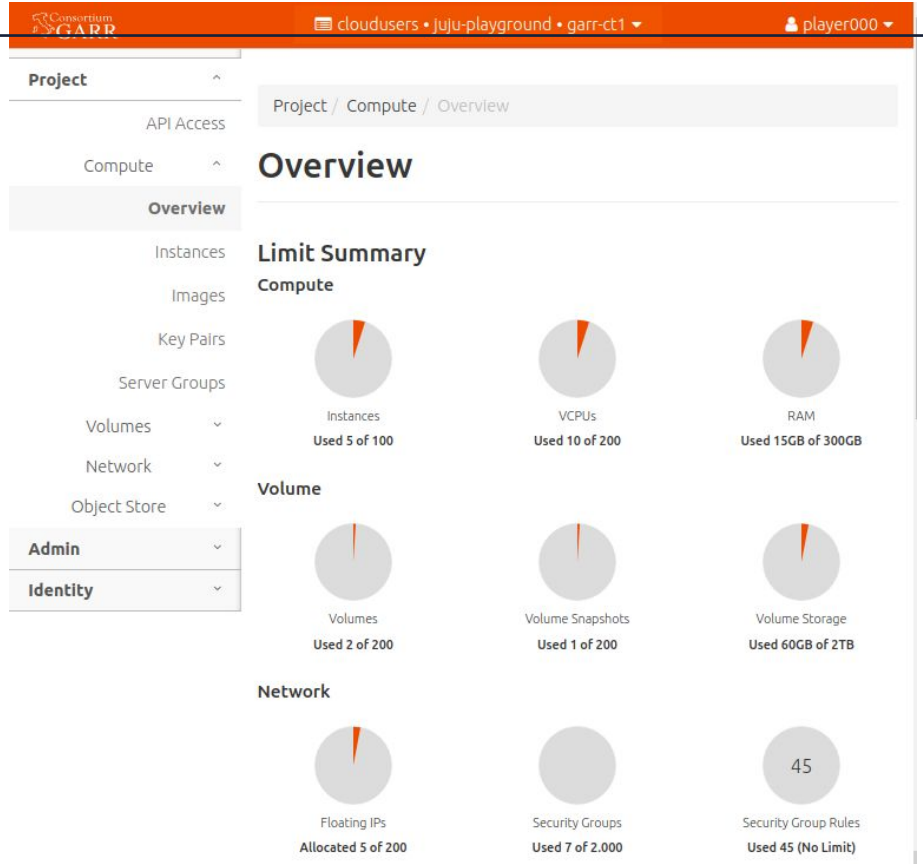

Global
Services
keystone
keystone
keystone
MariaDB
Galera Cluster
DB
MariaDB
Galera Cluster
MariaDB
Galera Cluster
glance
glance
glance
Swift
proxy
Swift
proxy
Swift
proxy
Object storage
Servizi
locali
Servizi
locali

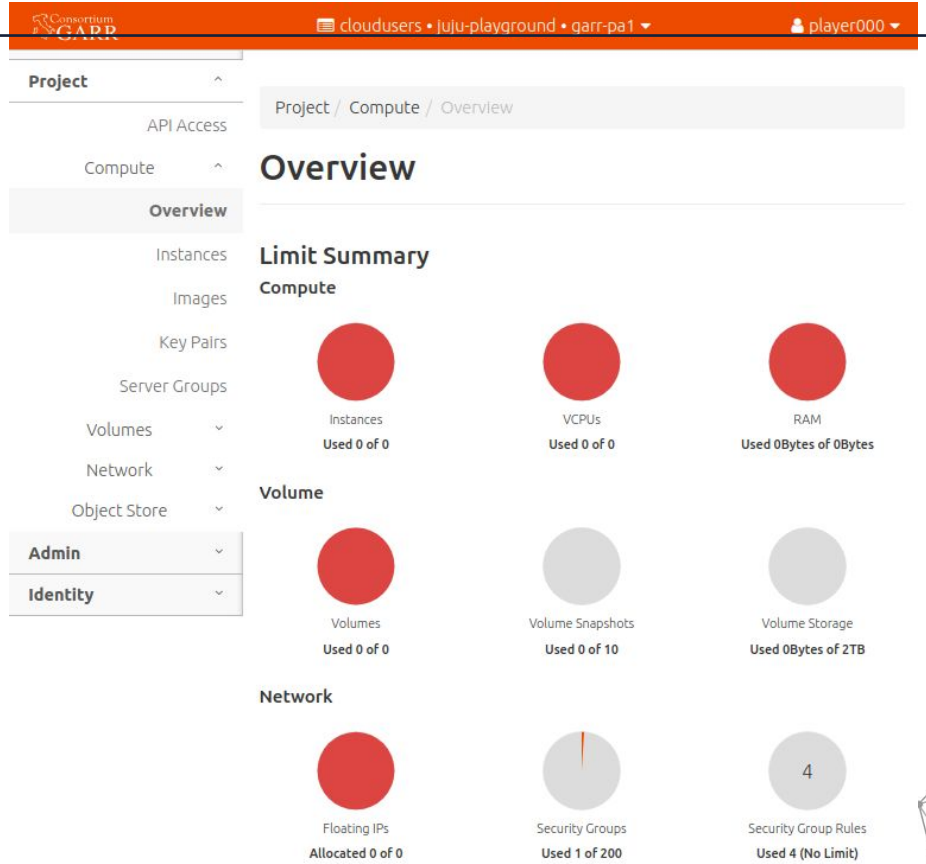

Global
Services
DNS
HA proxy
keystone
MariaDB
Galera Cluster
DB
MariaDB
Galera Cluster
MariaDB
Galera Cluster
glance
Swift
proxy
Object storage
local
services
local
services


local Services
Global Serivices
Controller Network
Controller Network
Controller Network
compute
compute
compute
compute
compute
compute
MariaDB Galera Cluster
MariaDB Galera Cluster
MariaDB Galera Cluster
cinder
cinder
cinder
Ceph mon
Ceph mon
Ceph mon
Block Storage
CEPH
replica 3 HDD
ssd cached

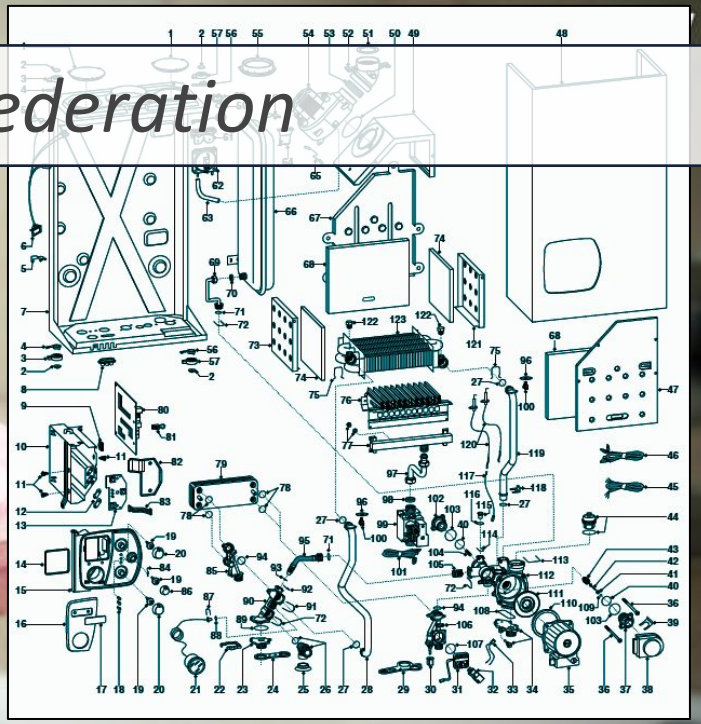


Regions, projects and quotas





our concept of *Federation*



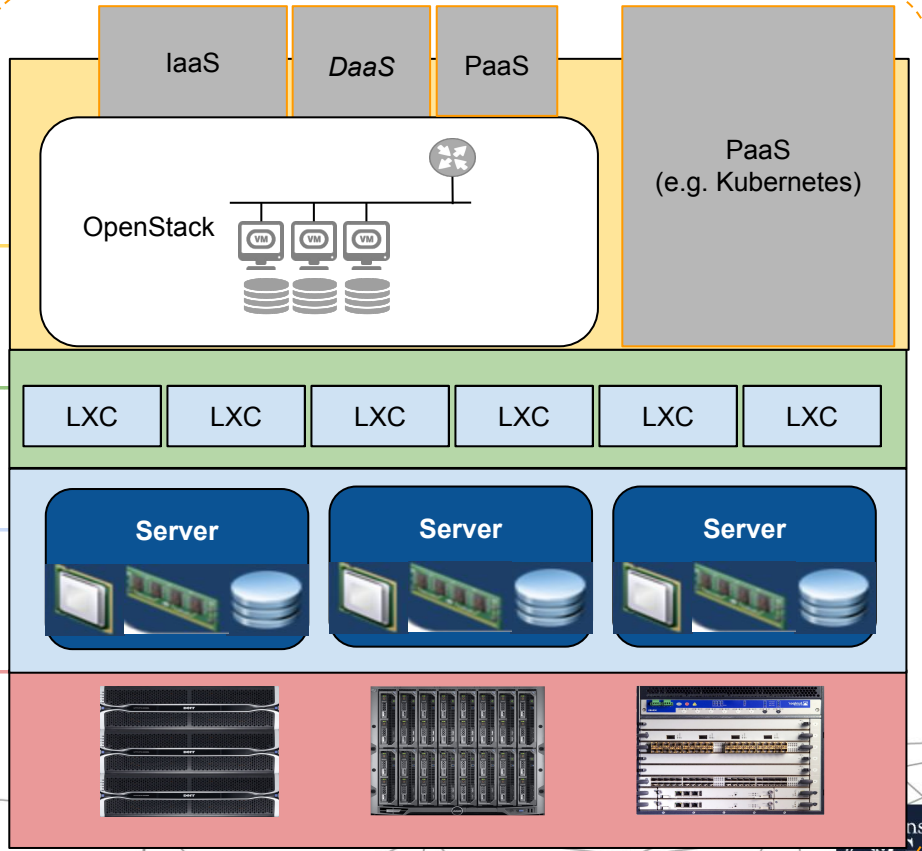
the simplest (for federated region admin) the better
the less requirements the more inclusive

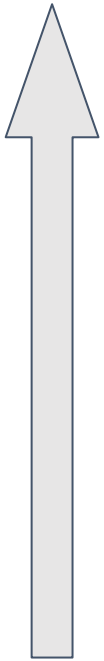
BETTY GILLIS

Ceph

4 Layers recipe:

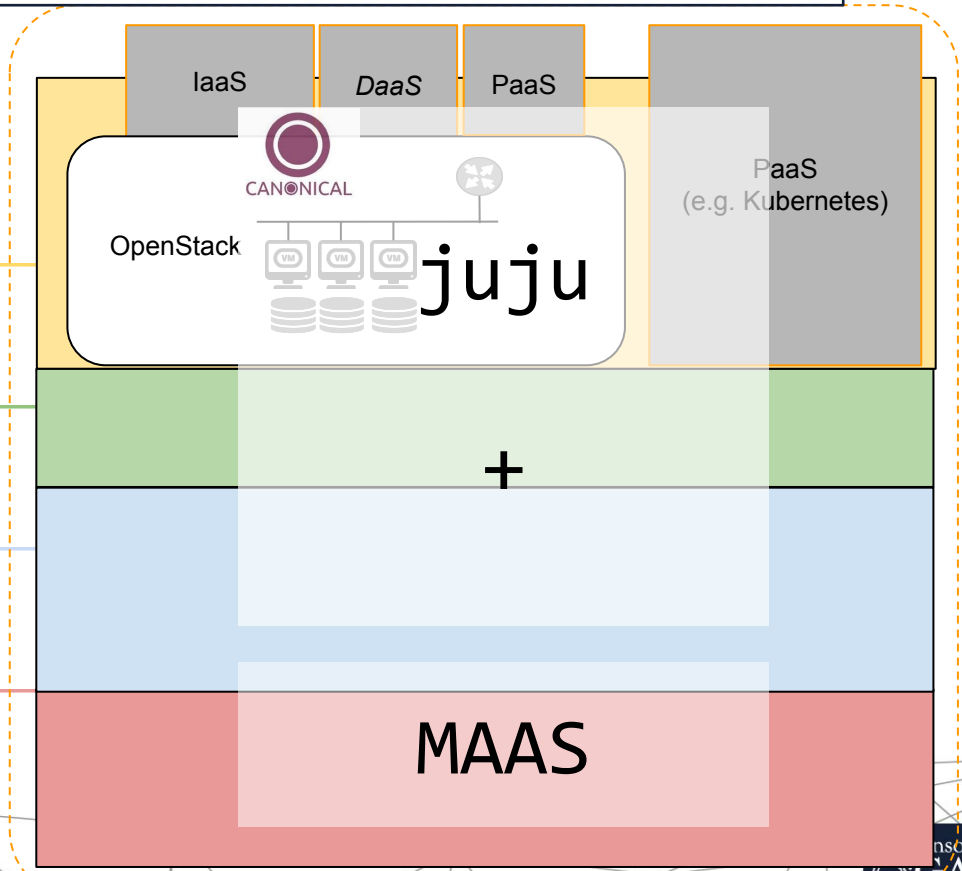
1. Application Services
2. Infrastructure *Virtualization*
3. Operating System
4. Physical resources





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

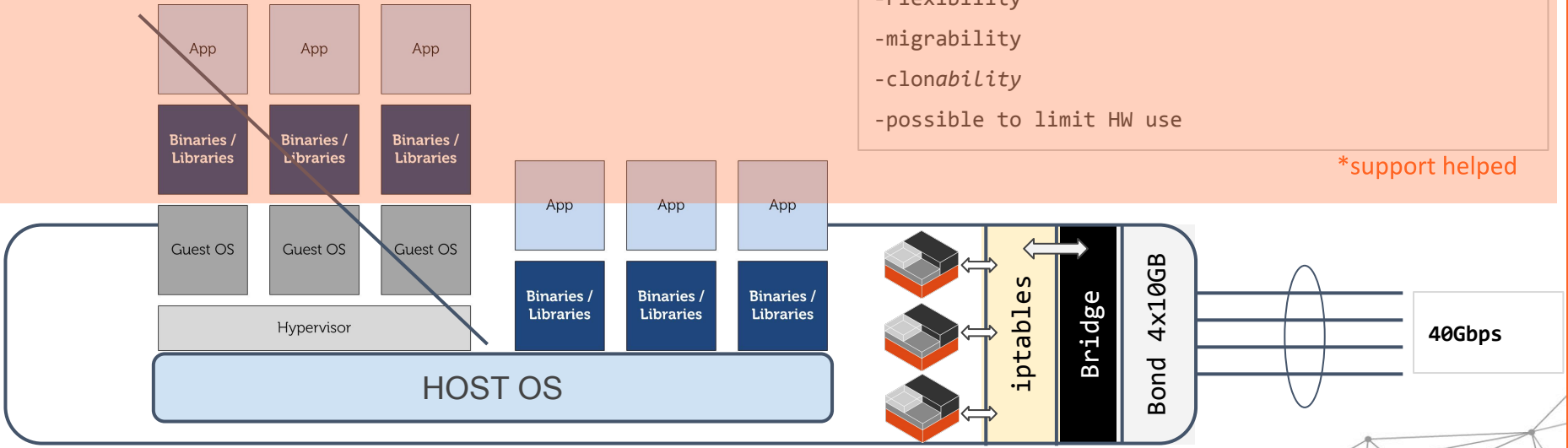
-Flexibility

-migrability

-clon*ability*

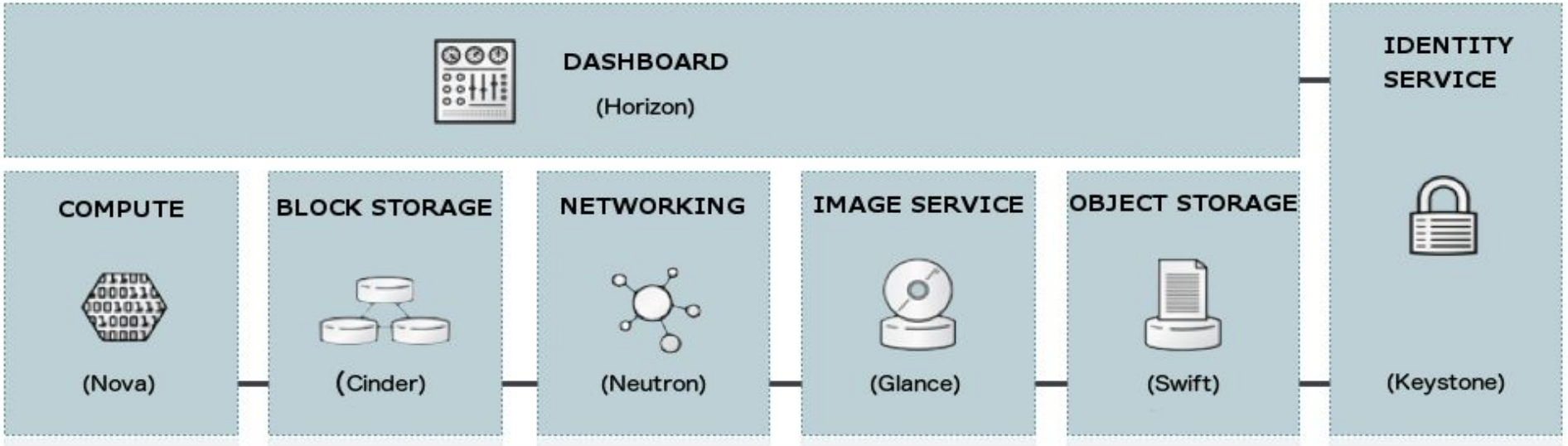
-possible to limit HW use

*support helped



OpenStack 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)



OpenStack through VM provisioning

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

Dashboard:
Horizon or CLI

Nova

Nova API
Scheduler
Conductor
DB
Queue

Compute node

Nova compute
VM
Hypervisor
Network

Keystone

Keystone API
DB

Cinder

Cinder API
Scheduler
Cinder Vol

Glance

Glance API
Glance Registry

radosGW (swift)

API
Ceph cli

Neutron

Neutron API
Scheduler
Plugin/agent

Block/Object Storage backend (Ceph)

Network Node

DHCP/IPAM
Router/GW

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):

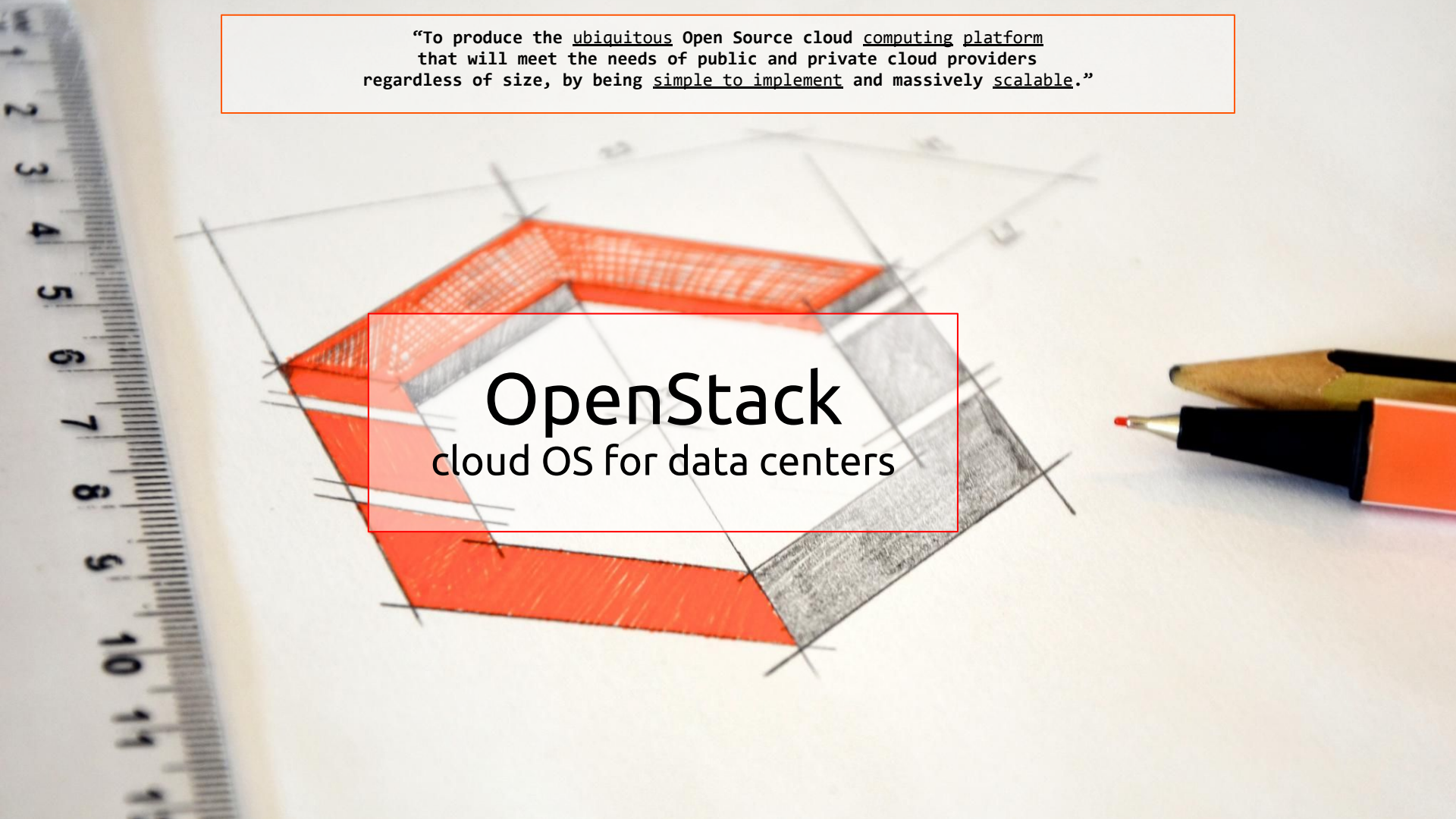
- HPC4AI project,
- Politecnico To,
- Uni Padova,
- 3 INGV region (external federation, see confederation)
- EAPConnect (East EU Nren, external federation)
- Hungary
- RECAS INFN Bari (maintenance)

federation recipe:

https://cloud.garr.it

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

“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.”

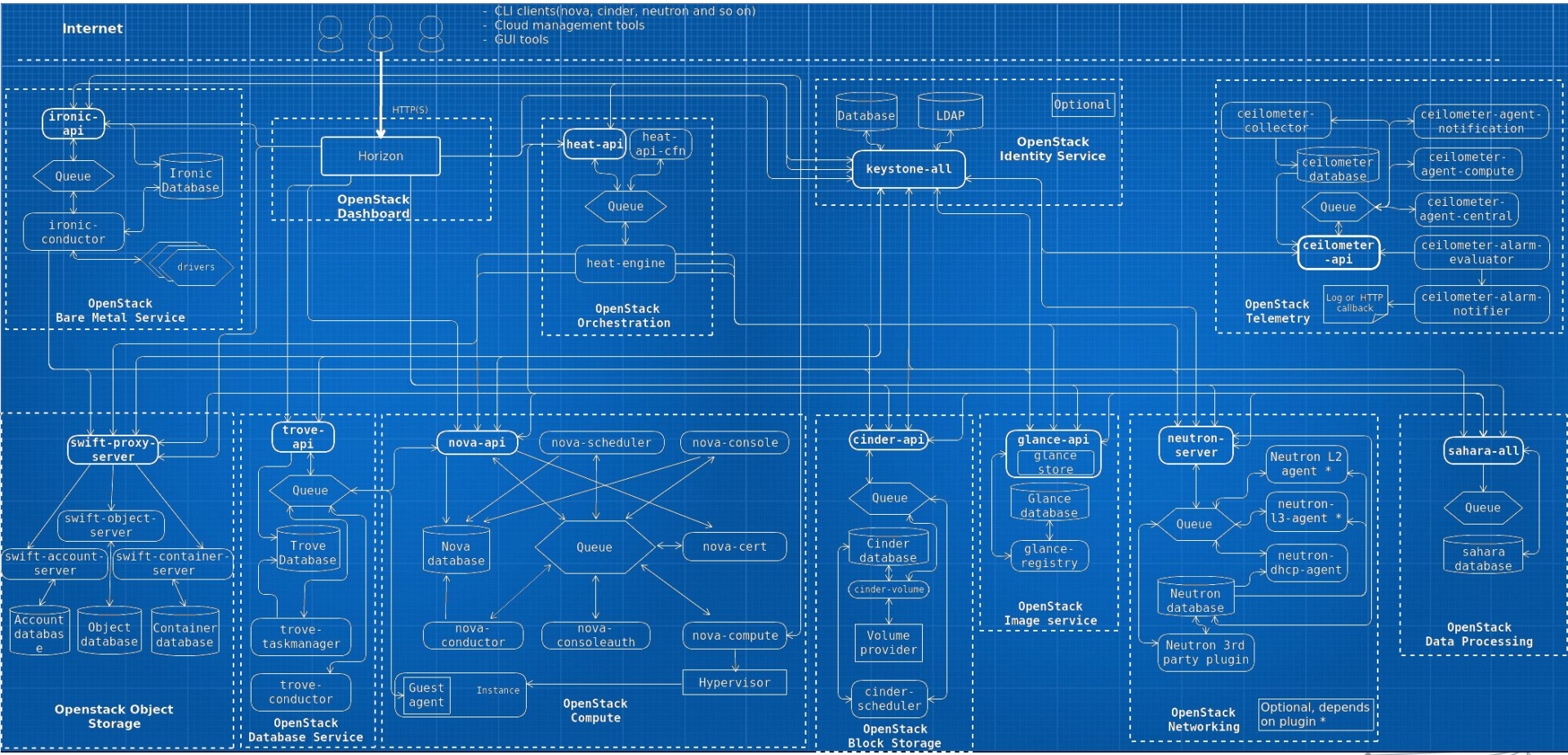


OpenStack

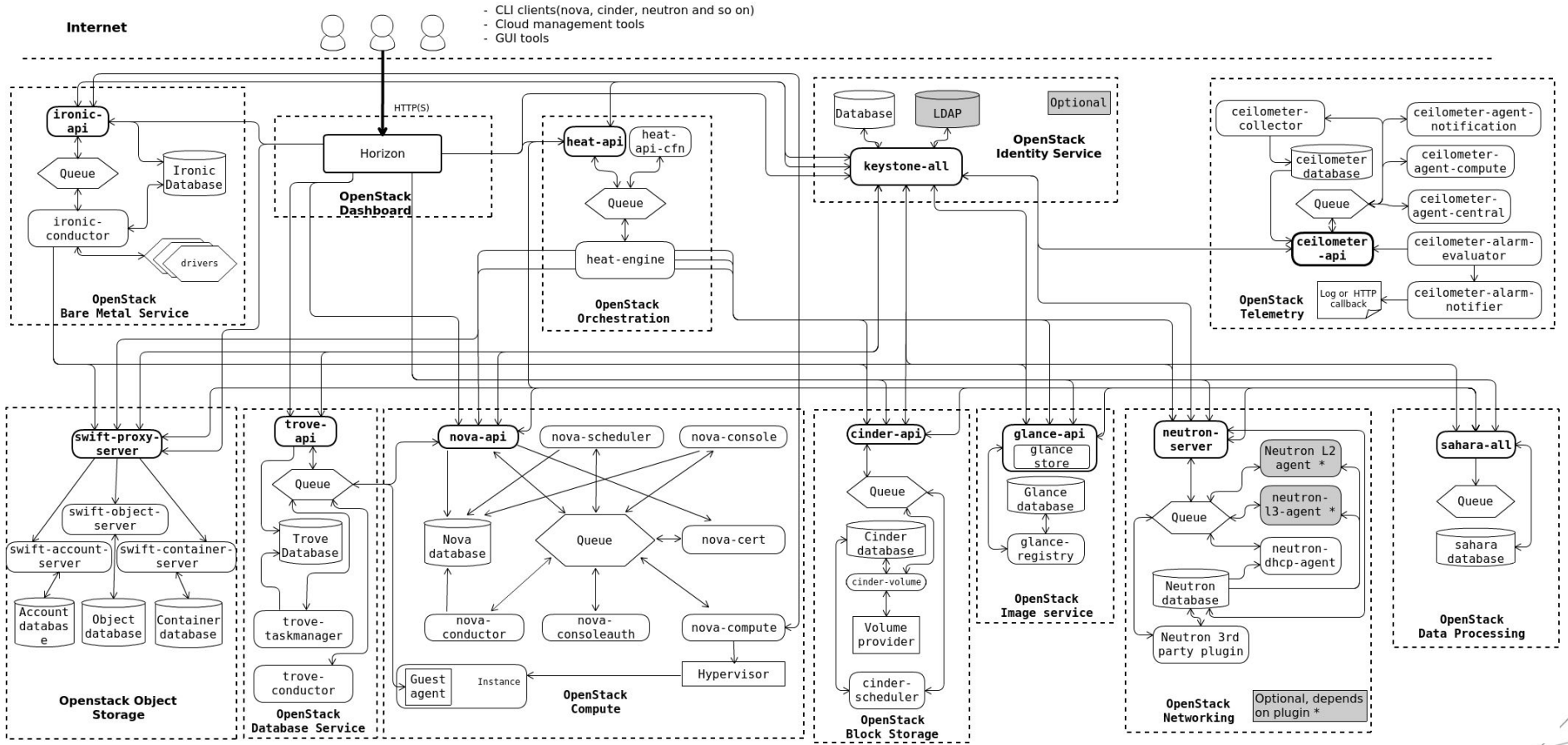
cloud OS for data centers

Juju

- Juju allows configuring, managing, maintaining, deploying and scaling cloud services quickly and efficiently on multiple providers:
  - private or public clouds
  - bare metal, leveraging MAAS to control the hardware
- Uses script collections called Charms which specify how to deploy a service
- Juju can manage and scale models consisting of many charms, creating complex architectures like OpenStack.
- Juju can be controlled via a web GUI, the command line, or API.



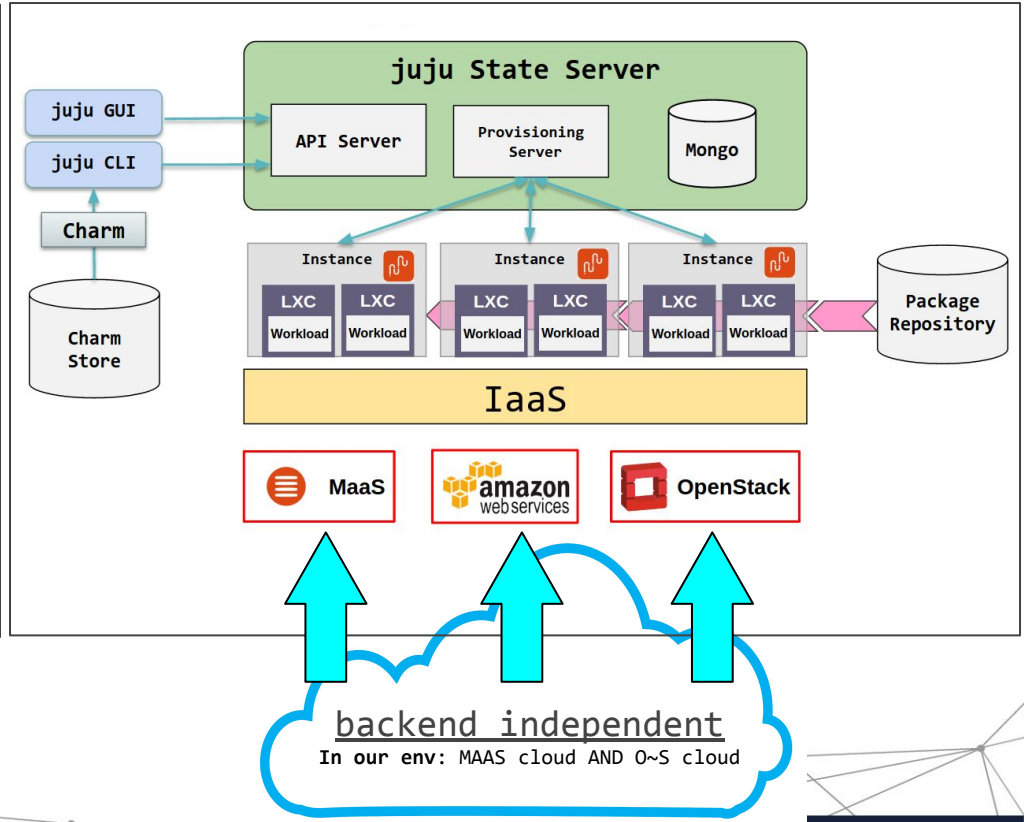
OpenStack Architecture



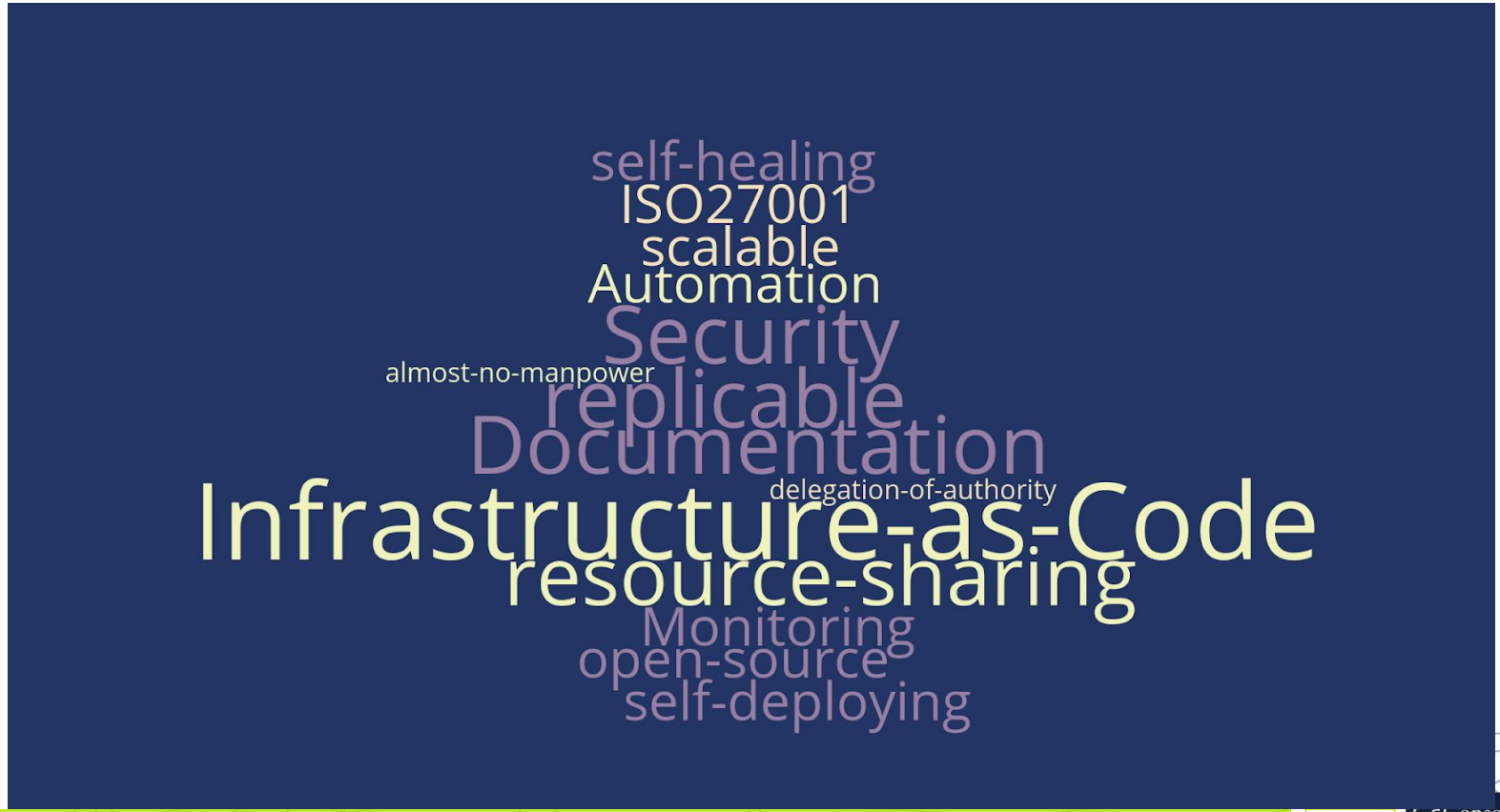
OpenStack Architecture

juju workflow

- engine follows a **reactive pattern**, triggered by events that cause hook handlers to run
- Multiple **handlers** may match for a given hook and **run** in a **non-determined order**
- The **state** engine is **evaluated** every event occurred
- The engine **runs until convergence** to a stable state



GARR Computing and Storage



Deploying an HA cloud application with Juju

```
# deploy a local web application and a database from the charm store
juju deploy ./wordpress
juju deploy mysql

# and connect them
juju add-relation wordpress mysql
```

Deploying an HA cloud application with Juju

```
# implement high-availability
juju remove-application mysql
juju deploy percona-cluster
juju add-relation wordpress percona-cluster
juju deploy haproxy
juju add-relation wordpress haproxy
juju config haproxy peering_mode=active-active
juju expose haproxy

# scale the application
juju add-unit -n 2 percona-cluster
juju add-unit -n 2 wordpress
juju add-unit -n 2 haproxy
```

Metal As A Service (MAAS)

- Discovers and deploys physical servers
- Allocates resources to match requirements
- Undeploys servers when they are no longer needed
- Allows cross datacenter provisioning