Charging and Billing

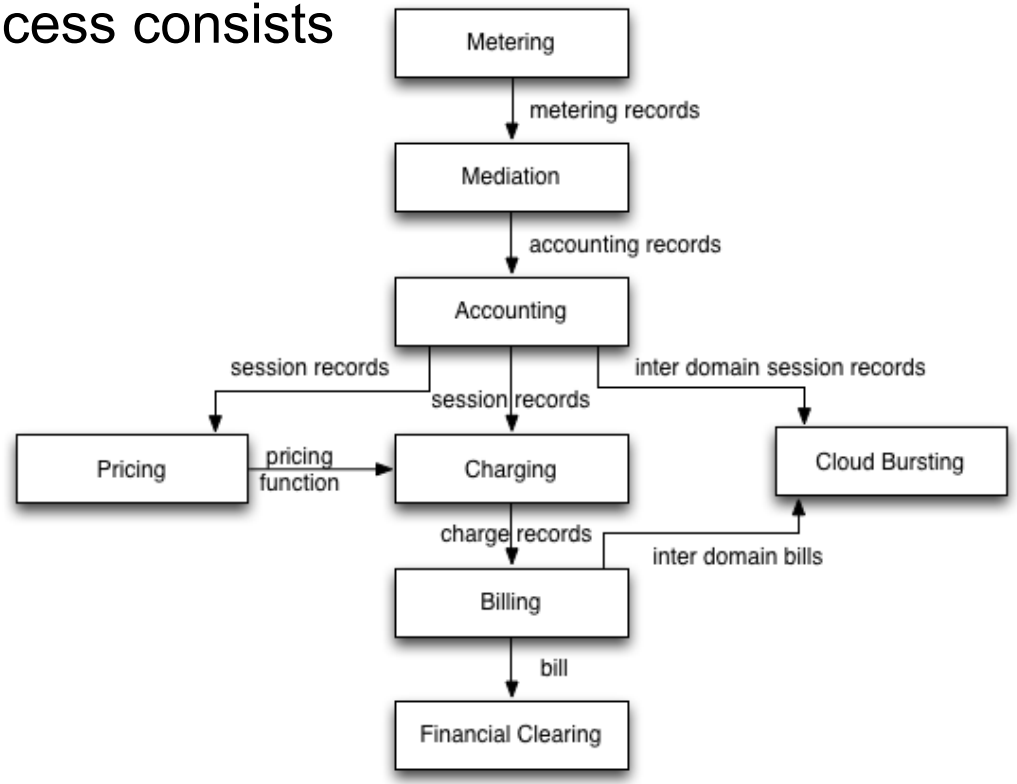
Giuseppe Attardi

Department CSD

Consortium GARR

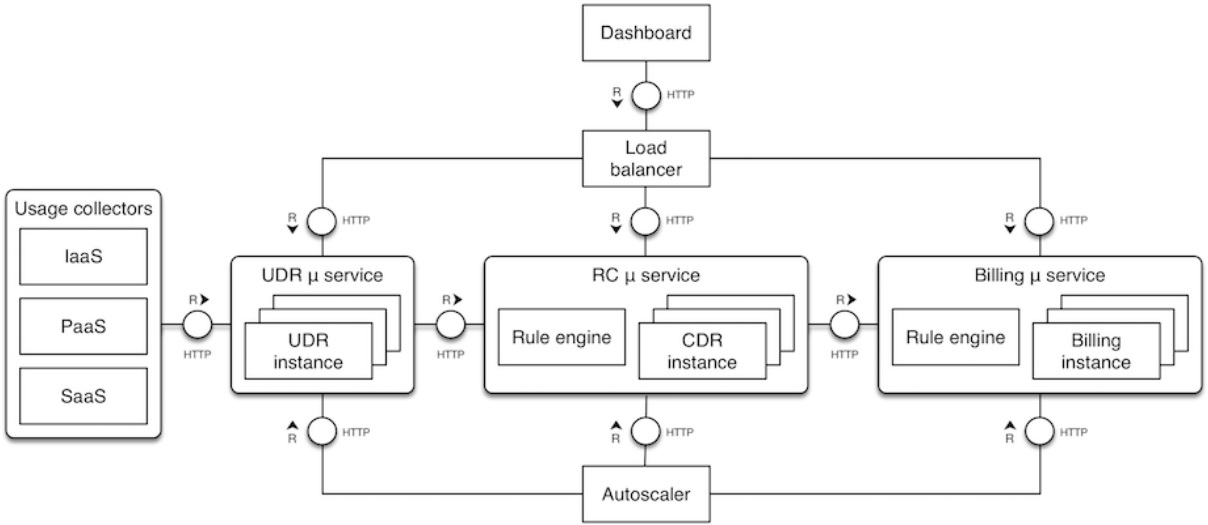
Architecture

- A typical accounting process consists in the following steps:
  - Metering
  - Mediation
  - Accounting
  - Pricing
  - Charging
  - Billing
  - Financial Clearing



Cyclops

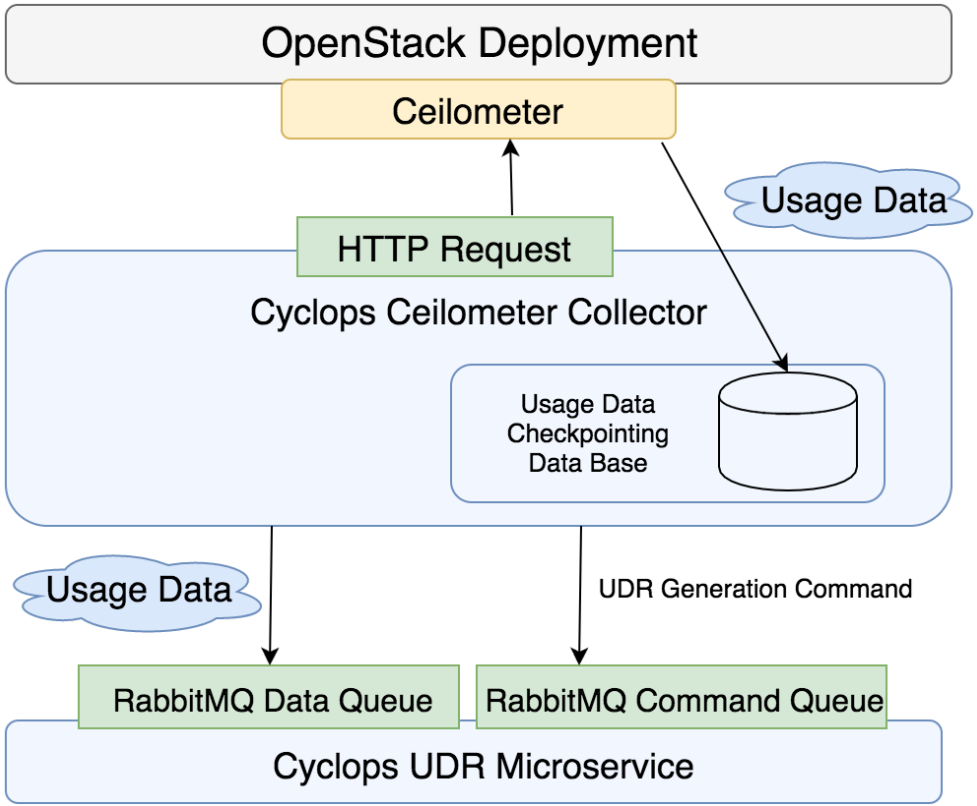
- Cyclops is an open-source accounting and billing framework, actively developed and maintained by ICCLab, consisting of micro-services, that enables agile, model-based accounting, billing for cloud and cloud based services.



- UDR micro service
  - stores the usage data records into an InfluxDB time series database and based on developer's preference, published over RabbitMQ.
- RC micro service:
  - Rating function
    - Once UDR publishes usage data records over RabbitMQ, the rating function takes care of calculating charge data records.
  - CDR micro service
    - The Rating function is also responsible for sending the created charge data records to the CDR micro service, which persists them into the time series database and also makes them available through RESTful API queries.

Ceilometer Collector

- operates on an internal scheduler, periodically asking OpenStack for new usage records.
- records are processed, harmonised and broadcasted to all RCB Cyclops.



Data

CumulativeMeterUsage	
id	Int autoincrement
usageCounter	double
usageKey	String

OpenStackMeter	
user_id	String
name	String
resource_id	String
source	String
meter_id	String
project_id	String
type	String
unit	String

AbstractOpenStackCeilometerUsage		
_class	String	Name of class
time	timestamp	Measurement time
account	String	User account name
usage	Double	Usage value
unit	String	Meter Unit
metadata	Map<String, Object>	Metadata

Collectors

CPU	DiskDeviceUsage	Instance
CPUdelta	DiskReadBytes	IpFloating
CPUutil	DiskReadBytesRate	MemoryResident
DiskAllocation	DiskReadRequests	MemoryUsage
DiskCapacity	DiskReadRequestRate	NetworkIncomingBytes
DiskDeviceAllocation	DiskUsage	NetworkIncomingBytesRate
DiskDeviceCapacity	DiskWriteBytes	NetworkIncomingPackets
DiskDeviceReadBytes	DiskWriteBytesRate	NetworkIncomingPacketsRate
DiskDeviceReadBytesRate	DiskWriteRequests	NetworkOutgoingBytes
DiskDeviceReadRequests	DiskWriteRequestRate	NetworkOutgoingBytesRate
DiskDeviceReadRequestRate	DiskUsage	NetworkOutgoingPackets
DiskDeviceUsage	Image	NetworkOutgoingPacketsRate
DiskDeviceWriteBytes	ImageDownload	StorageObjects
DiskDeviceWriteBytesRate	ImageServed	StorageObjectsContainers
DiskDeviceWriteRequests	ImageSize	StorageObjectsSize
DiskDeviceWriteRequestRate		Usage

UDR

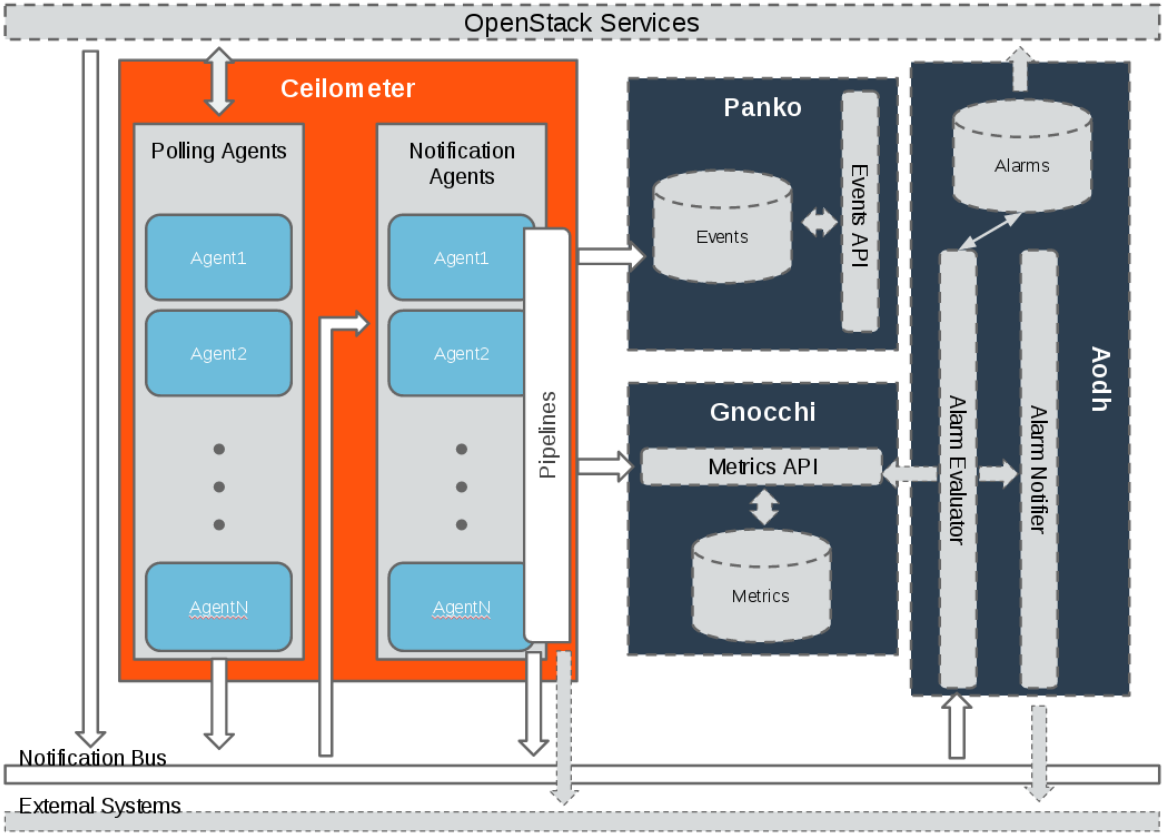
OpenStackUsageData		
count	int	Data output count
duration_start	timestamp	time where the measurement started
duration_end	timestamp	time where the measurement ended
min	double	Minimum value in the measurement
max	double	Maximum value in the measurement
sum	double	Sum of values in the measurement
avg	double	Average value in the measurement
period	double	Collection period
period_start	timestamp	time where the collection period started
period_end	timestamp	time where the collection period ended
duration	double	Duration of the measurement
unit	string	Measurement usage unit
groupby	Map	map containing project_id, user_id and resource_id

Ceilometer API and Collector

- Unfortunately the Ceilometer API and Collector on which Cyclops relies have been discontinued because of scalability limits
- They have been replaced by Gnocchi

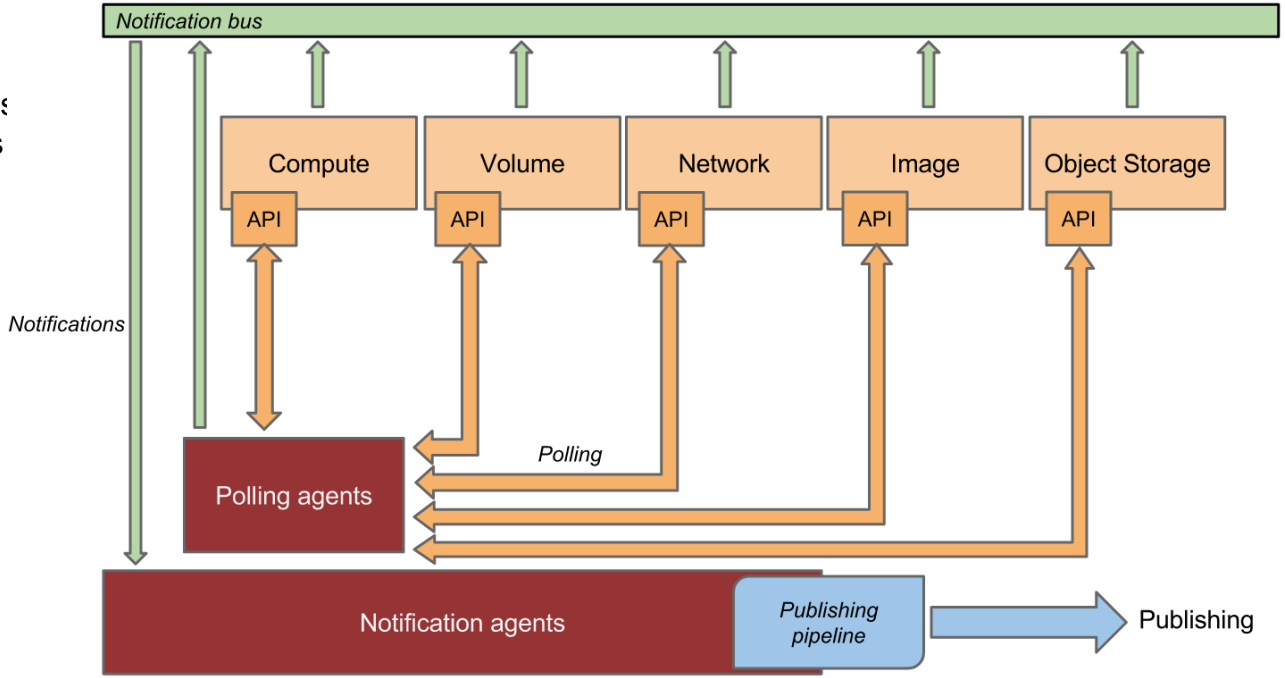
Ceilometer + Gnocchi + Aodh Architecture

- Ceilometer (data collection): to efficiently collect, normalise and transform data produced by OpenStack services
  - Polling agent: daemon designed to poll OpenStack services and build Meters
  - Notification agent: daemon designed to listen to notifications on message queue, convert them to Events and Samples, and apply pipeline actions
- Gnocchi (metric storage): to provide a time-series resource indexing, metric storage service
- Aodh (alarming): to enable the ability to trigger actions based on defined rules against sample or event data collected by Ceilometer
- See https://docs.openstack.org/developer/ceilometer/architecture.html



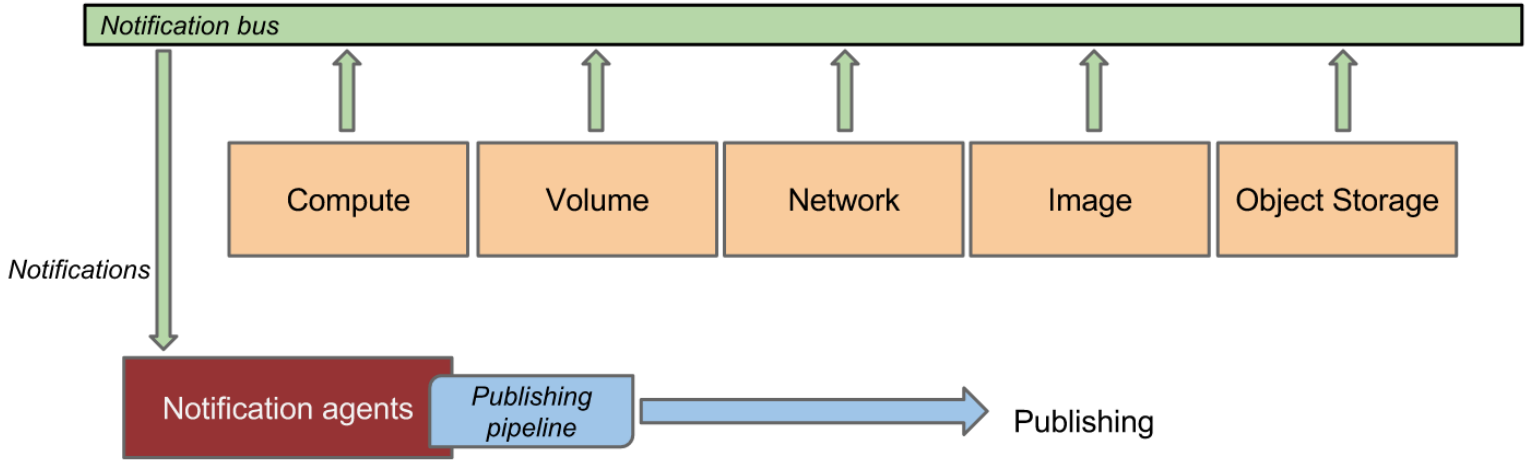
Gathering the data

- Notification agent: takes messages generated on the notification bus and transforms them into Ceilometer samples or events. This is **the preferred method** of data collection.
- Polling agent: polls some API or other tool to collect information at a regular interval

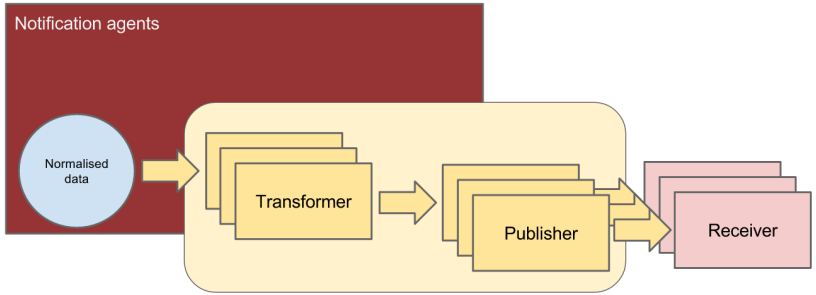


Notification agents: Listening for data

- Notification daemon (agent-notification) monitors the message queue for data sent by other OpenStack components such as Nova, Glance, Cinder, Neutron, Swift, Keystone, as well as Ceilometer internal communication.
  - Loads one or more *listener* plugins
  - Sample-oriented plugins provide a method to list the event types they're interested in and a callback for processing messages accordingly

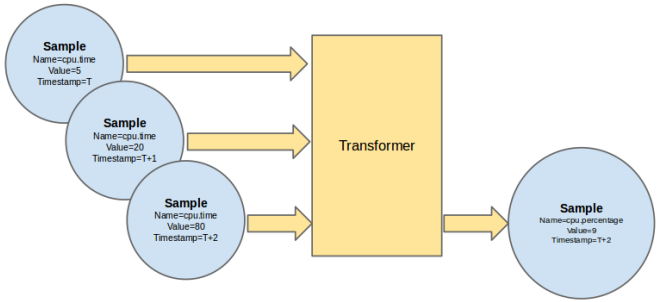


Processing the data

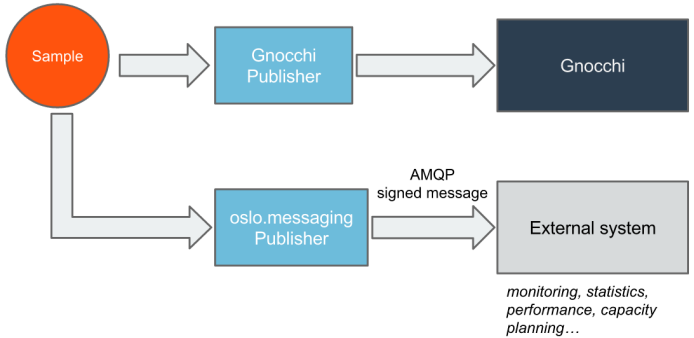


Pipeline: a set of transformers mutating data points into something that publishers know how to send to external systems.

Pipeline Manager



Transforming the data

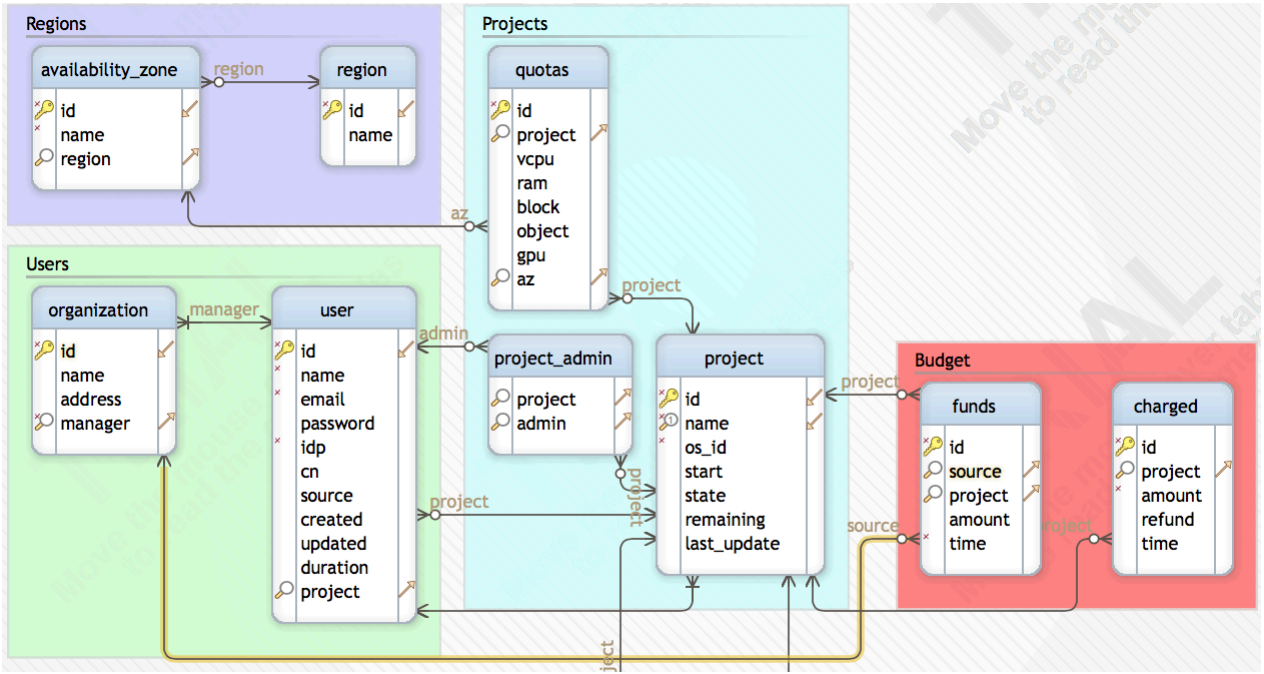


Publishing the data

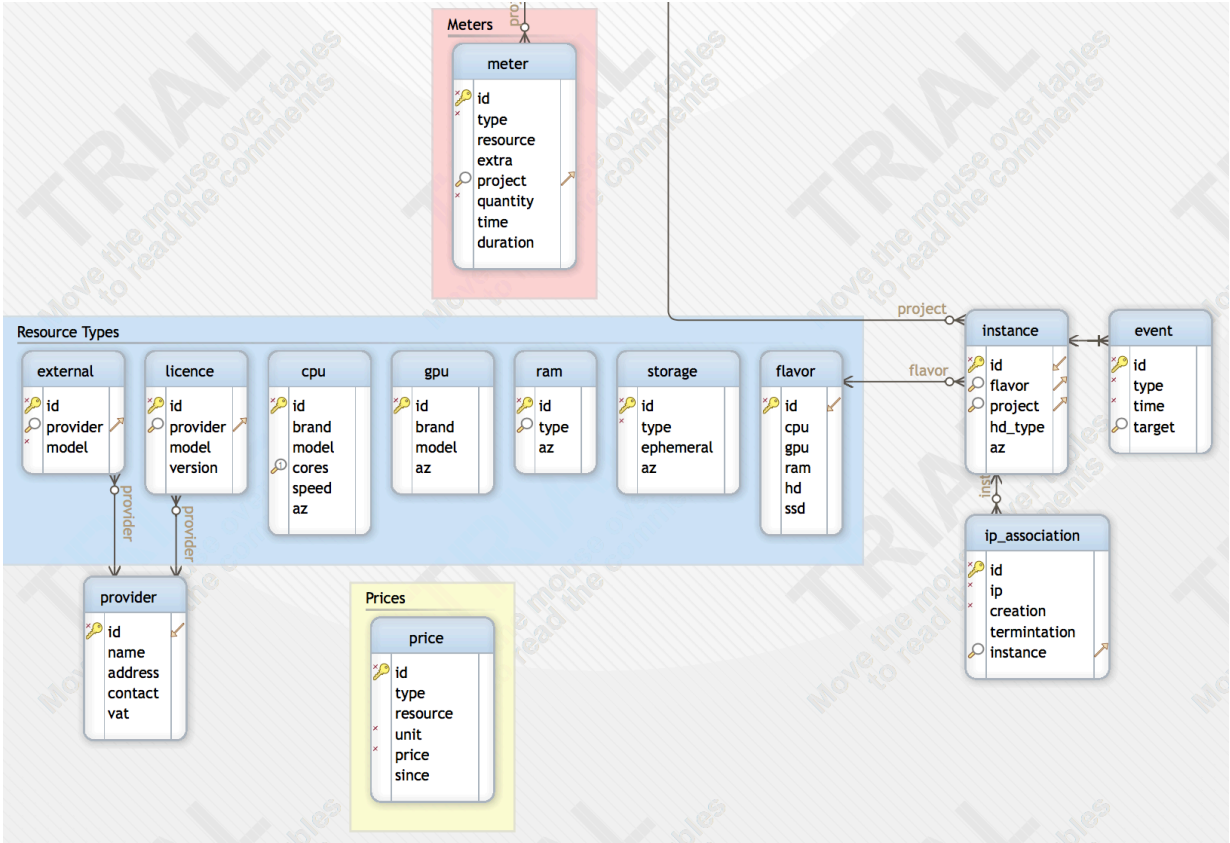
Proposed Solution

- Ceilometer (data collection)
- Gnocchi (metrics as a service)
- CloudKitty (rating and billing)

Database di Gestione (1)



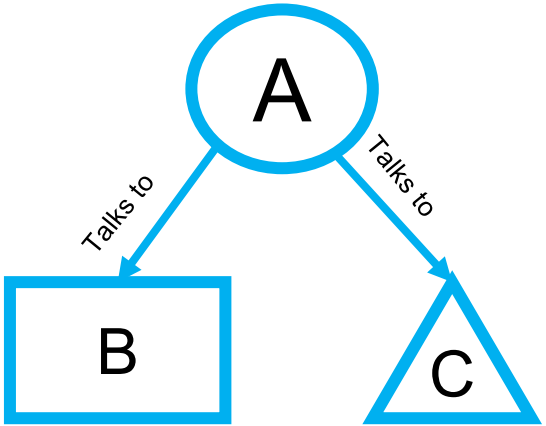
Database di Gestione (2)



Why Juju

Declarative Modeling

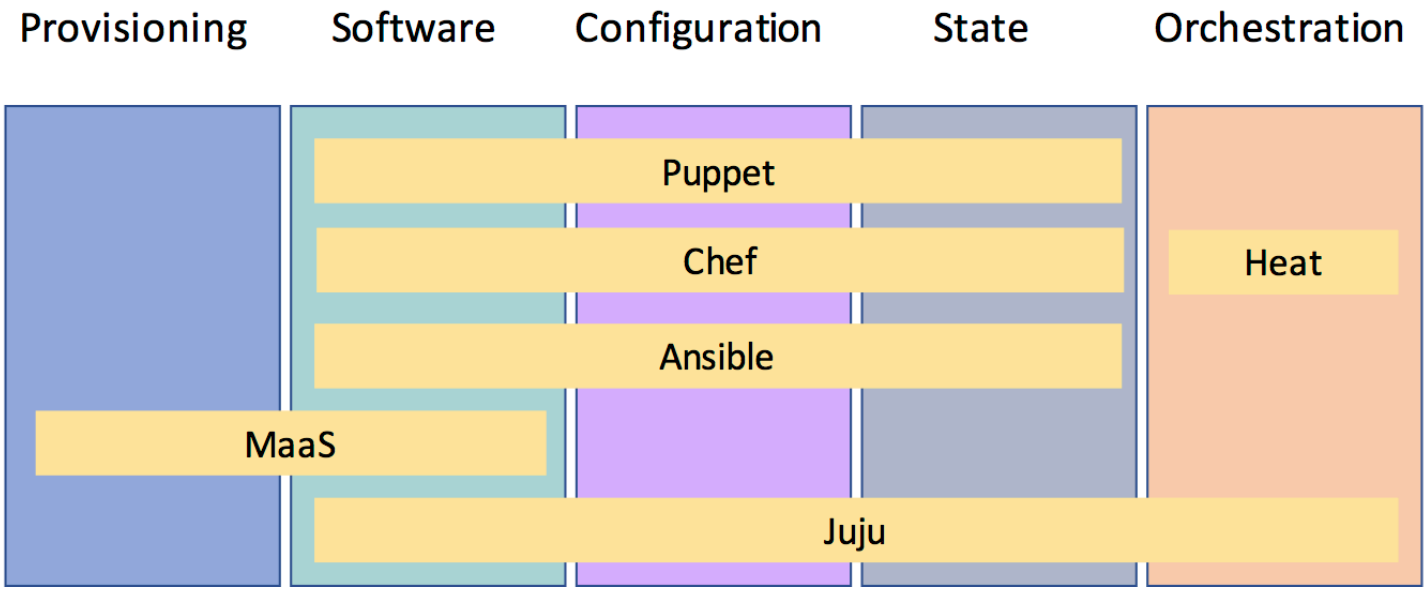
- App A requires:
  - X GB memory and Y CPU
  - N GB storage
  - To talk with B and C
  - URL contact
  - Run locally, close to B



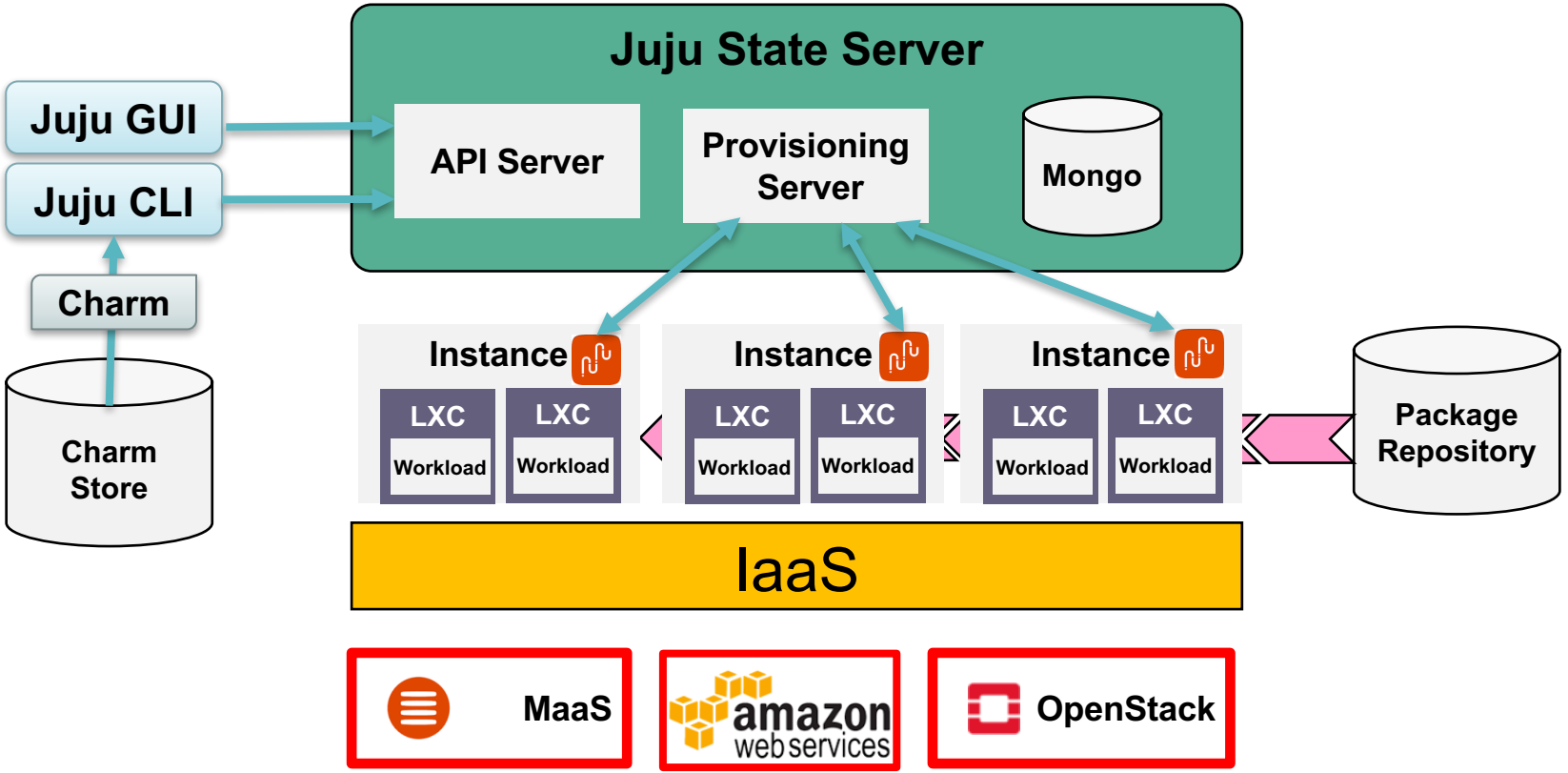
Declarative Modeling

- Bikash Koley, director of Google Architecture:
  - 70% of outages are due to interventions on the infrastructure
  - Human mind unable to keep track of the full state of a complex system
  - It must be done by software
- Focus on automation: code first
- Describe what, not how
- Workflow Engine generates execution plan from the desired architecture
- Asynchronous process that converges by computing the differences between the current and the desired state
- See:
  - https://www.thinkmind.org/download.php?articleid=patterns_2017_2_30_78006
  - https://arxiv.org/pdf/1706.05272

Automation Tools



Juju Architecture



Basic Series of Events

- install
  - Invoked just once when the charm is deployed
- config-changed
  - Invoked whenever a configuration parameter is changed (either grom the GUI, or from the CLI)
- relation-joined, relation-changed
  - When a relation is added to a charm relation-joined fires first, so that the two units can communicate with each other, and then relation-changed fires
- leader-elected
  - Occurs when many nodes require a “leader” node to coordinate among them
- pool-storage-attached, pool-storage-detached
  - Actions to take when a storage pool is attached/detached

Hooks

- Represent the handlers to be run when an event occurs
- Hooks must be *idempotent*
  - To avoid inconsistencies or divergence if run more than once

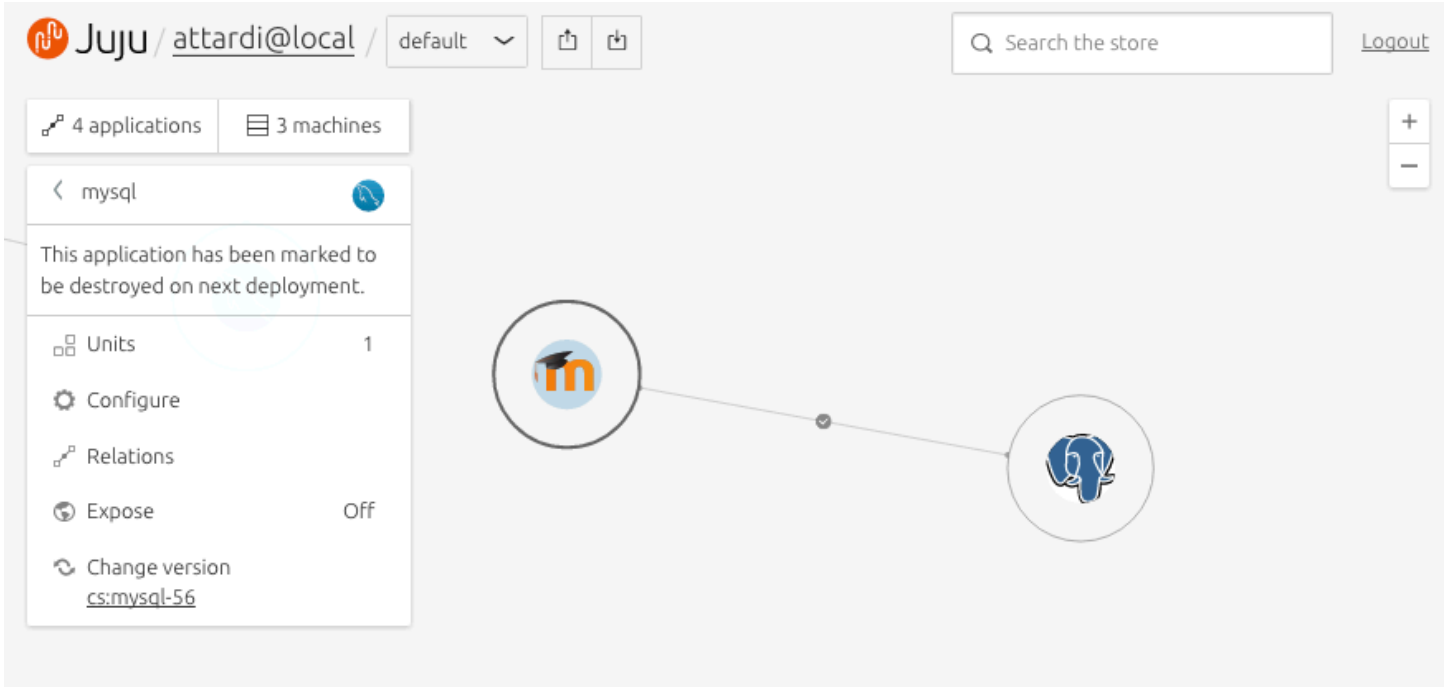
Bundles

- Bundles describe a service consisting of several charms
- They express constrains, configuration parameters and relationships between charms that provide/implement an interface
- Can be configured before/after deployment
- They provide scalability options

Juju Engine

- The Juju engine follows a reactive pattern, triggered by events that cause corresponding hook handlers to run
- Multiple handlers may match for a given hook and will be run in a non-determined order
- Running the handlers or issuing Juju commands may cause additional events
- The state engine is evaluated every time an event occurs
- The engine runs until convergence to a stable state

Deployment bundle Moodle



Federation with External Clouds: AWS

Access to Juju controller

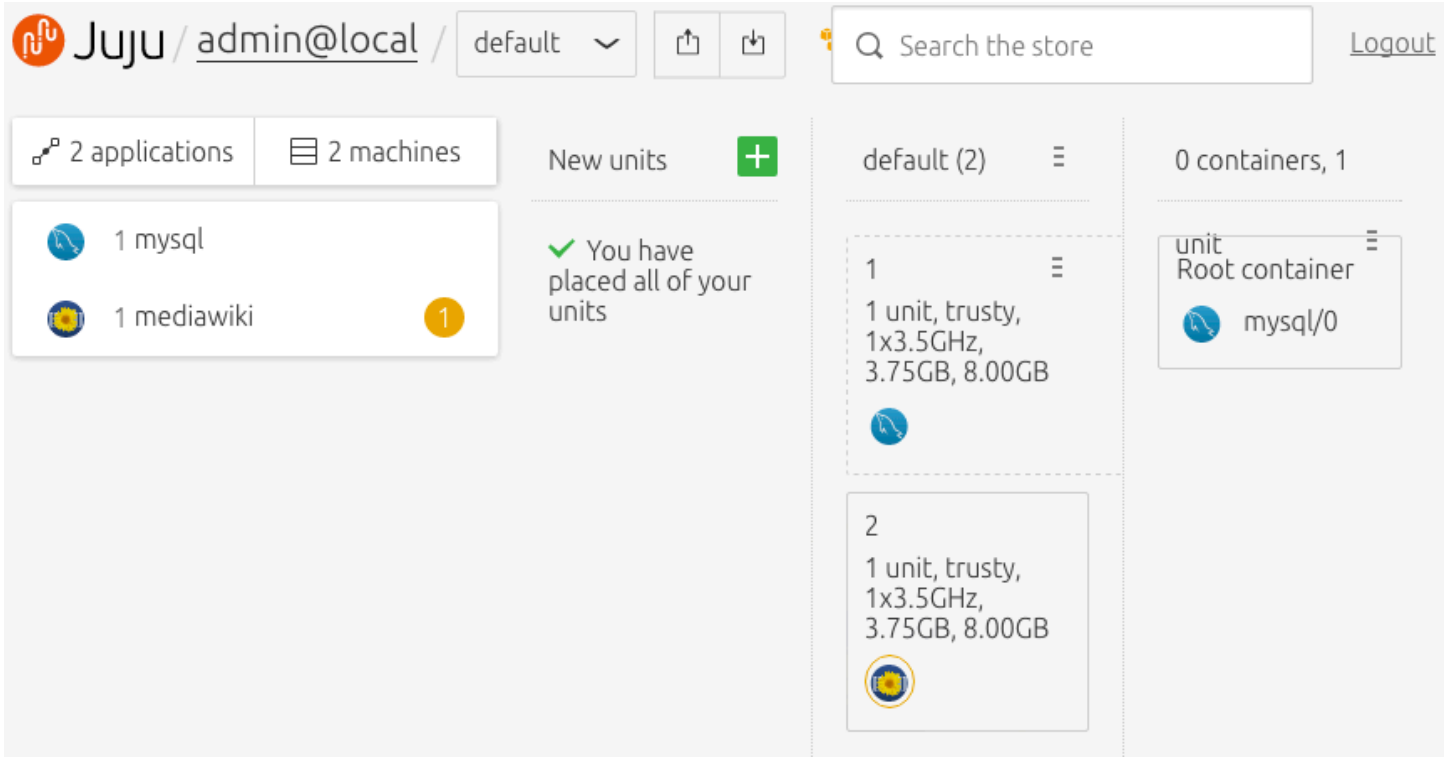
Available clouds

```
$ ssh -i aws.pem ubuntu@90.147.166.80

$ juju clouds
Cloud        Regions  Default        Type        Description
aws               12  us-east-1      ec2         Amazon Web Services
aws-china          1  cn-north-1     ec2         Amazon China
aws-gov            1  us-gov-west-1  ec2         Amazon (USA Government)
azure             18  centralus      azure       Microsoft Azure
azure-china        2  chinaeast      azure       Microsoft Azure China
cloudsigma         5  hnl            cloudsigma  CloudSigma Cloud
google             4  us-east1       gce         Google Cloud Platform

$ juju deploy ~/mediawiki-single
```

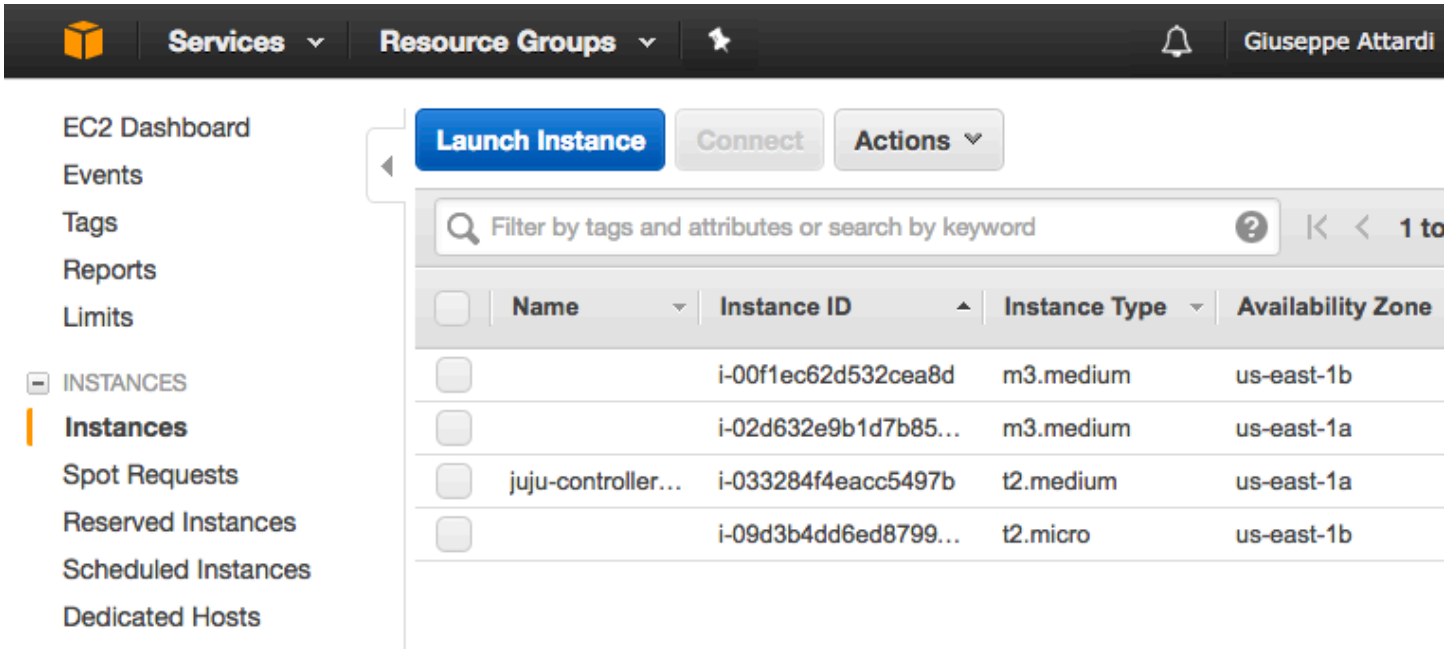
MediaWiki application

```
$ juju status
Model    Controller     Cloud/Region   Version
default  aws-us-east-1  aws/us-east-1  2.0.2

App        Version  Status   Scale  Charm      Store       Rev  OS      Notes
mediawiki           unknown      1  mediawiki  jujucharms    3  ubuntu  exposed
mysql               waiting      0  mysql      jujucharms   29  ubuntu

Unit          Workload  Agent  Machine  Public address  Ports  Message
mediawiki/1*  unknown   idle   2        54.161.6.44

Machine  State    DNS          Inst id              Series  AZ
2        started  54.161.6.44  i-02d632e9b1d7b8507  trusty  us-east-1a
```

MediaWiki Deployed on AWS: Juju view



MediaWiki Deployed on AWS: AWS View



Long Tail of Science

- Self-service App oriented to researchers from any domain
- Without expertise in system administration
- Catalogues of over200 applications at jujucharms.com