

Versatile billing engine made for clouds and beyond..



Dr. Piyush Harsh  
<http://piyush-harsh.info/>

Senior Researcher, ICCLab  
Zurich University of Applied Sciences

# Talk Outline

What you will learn :)

★ Quick primer

★ Current Release Info

- Architecture
- Description of Micro services
- Key innovations
- Supported platforms

★ Specific use-cases

- EU FP7 Project: MCN
- EU FP7 Project: T-Nova
- Swiss National Project: SCALEUP
- Swiss Transfer Project: SSC

★ Short demo

★ Future Plans

★ Where to get this framework?



# ICCLab @ ZHAW

We are a young lab - 4 Yrs old, mostly funded by EU and Swiss projects.

Rapid growth: from 3 founding members to ~25 researchers and still growing.

Highly diverse and international team

4 professors, 4 senior researchers, 6 researchers, 3 MS students, 1 Bachelor student and 5 interns.

Find more info: [blog.zhaw.ch/icclab/](http://blog.zhaw.ch/icclab/)



We transformed from



to ...



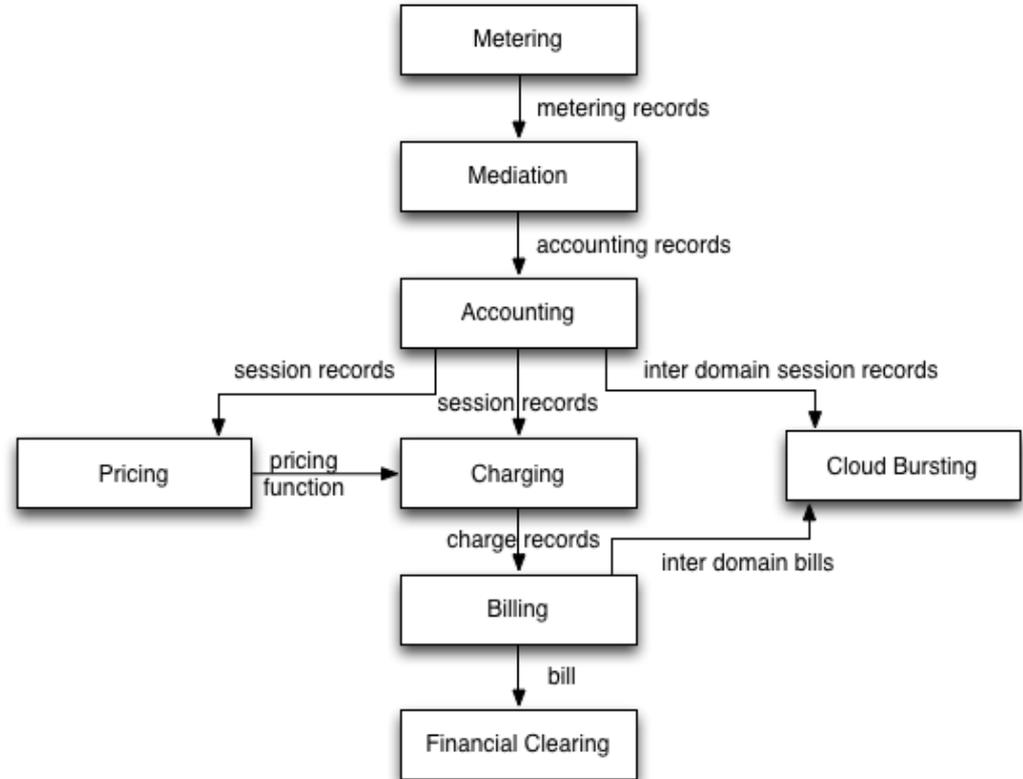
A quick primer ...



# How does a typical accounting process look like?

Key phases -

- Metering
- Mediation
- Accounting
- Pricing
- Charging
- Billing
- Financial Clearing



# Cyclops: present release

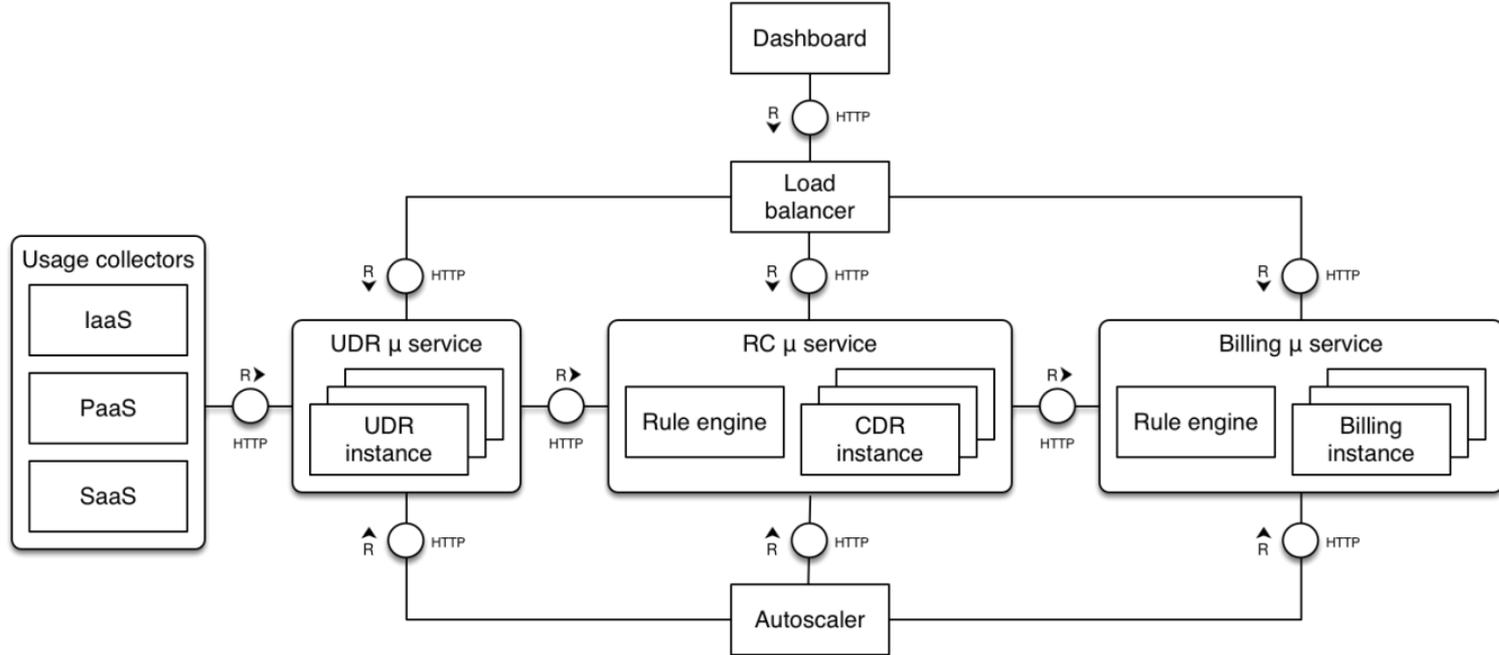


# What is Cyclops?

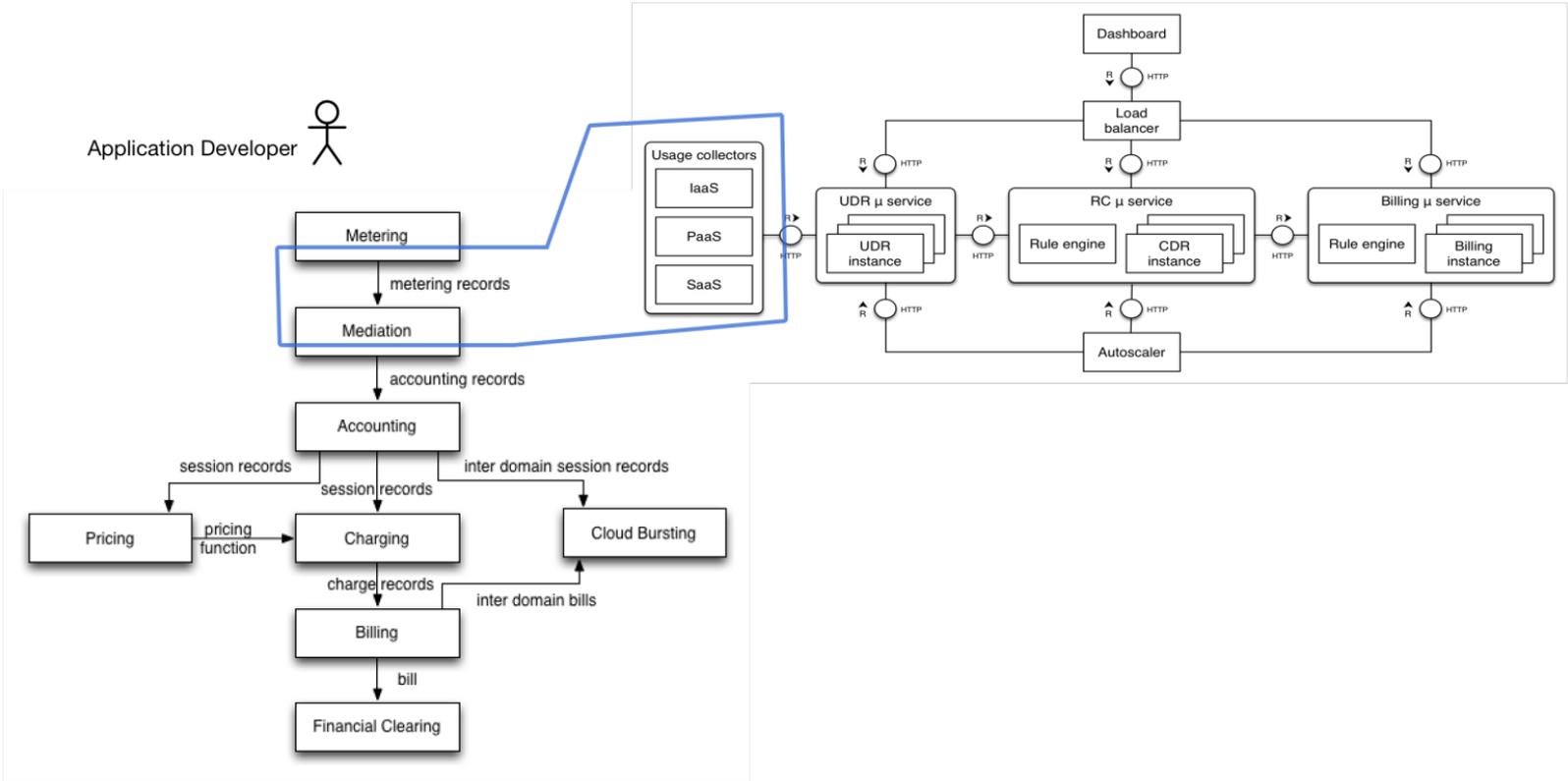
An open-source (ASL2.0), actively developed and maintained (by ICCLab), framework consisting of well defined micro-services that enables agile, model-based accounting, billing for cloud and cloud based services.



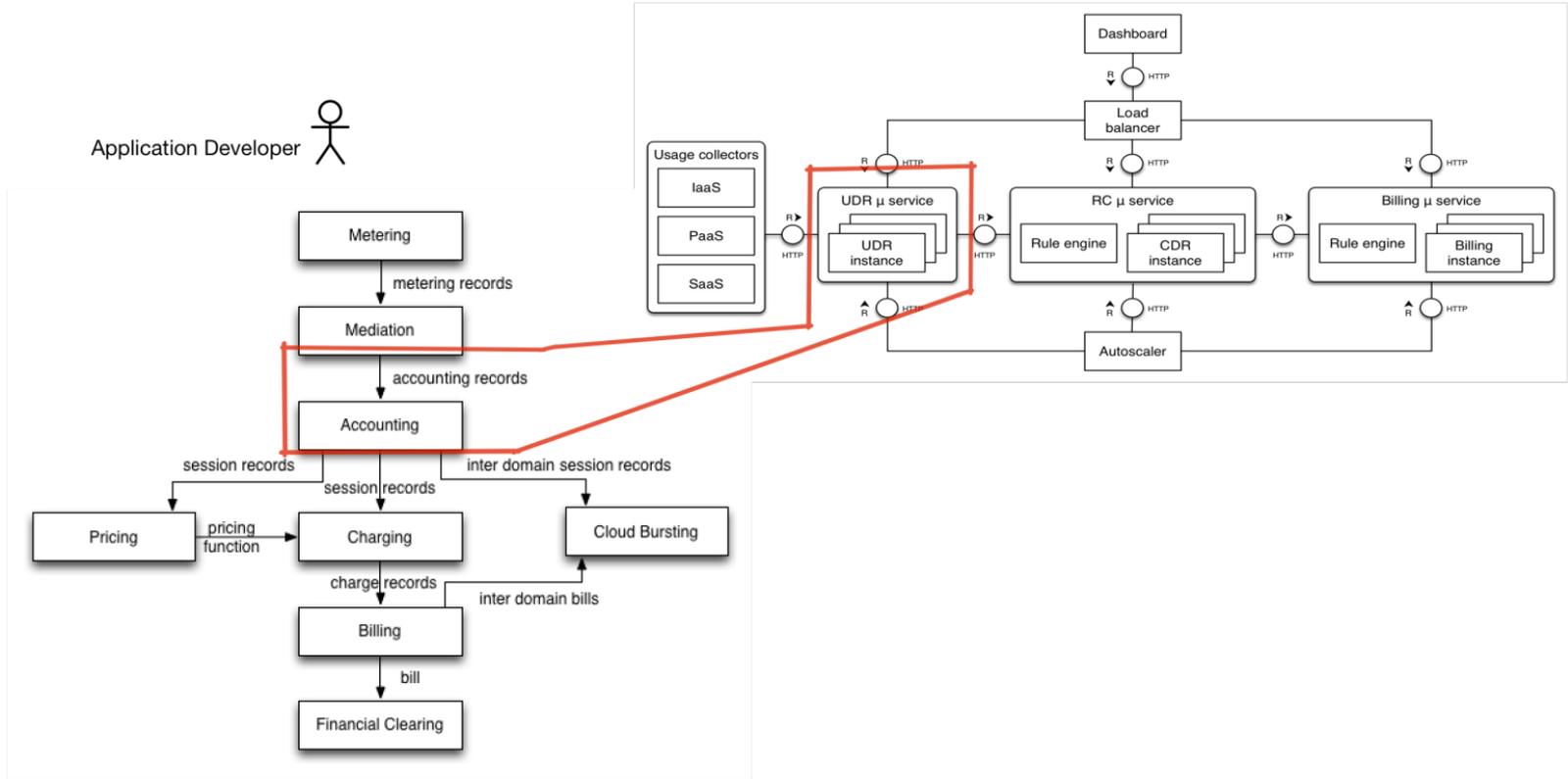
# Architecture



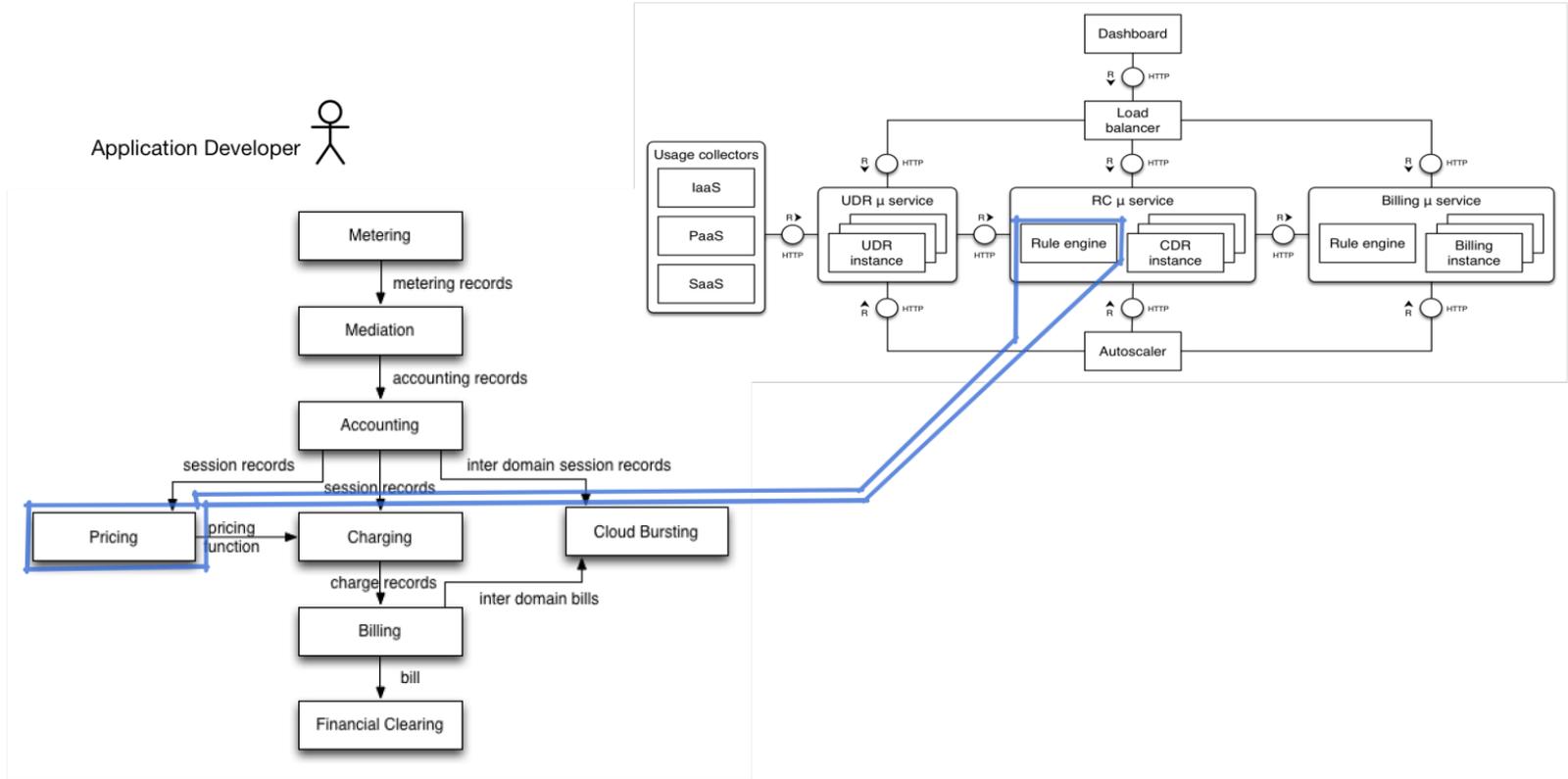
# Mapping Cyclops to generic accounting process



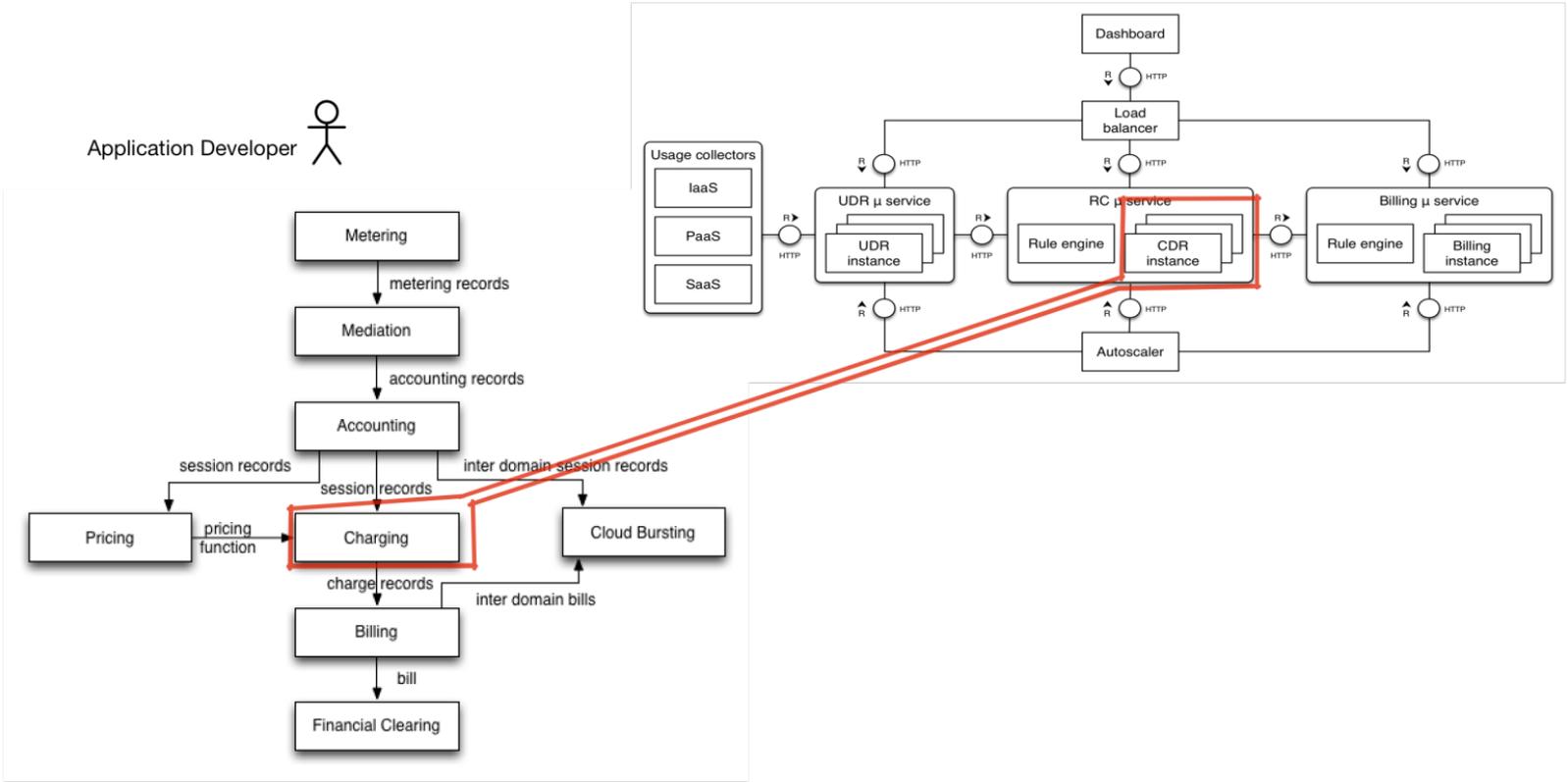
# Mapping Cyclops to generic accounting process



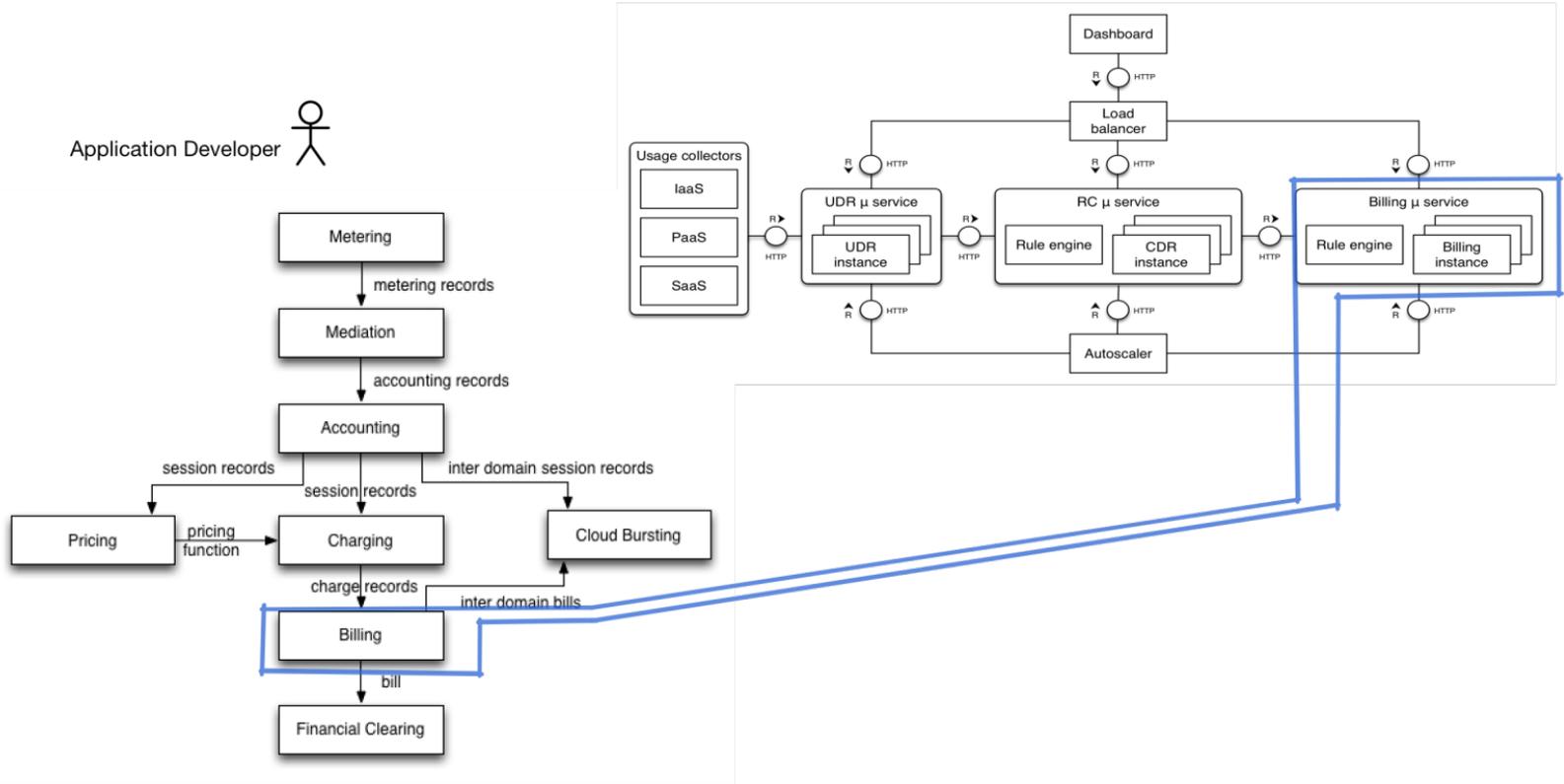
# Mapping Cyclops to generic accounting process



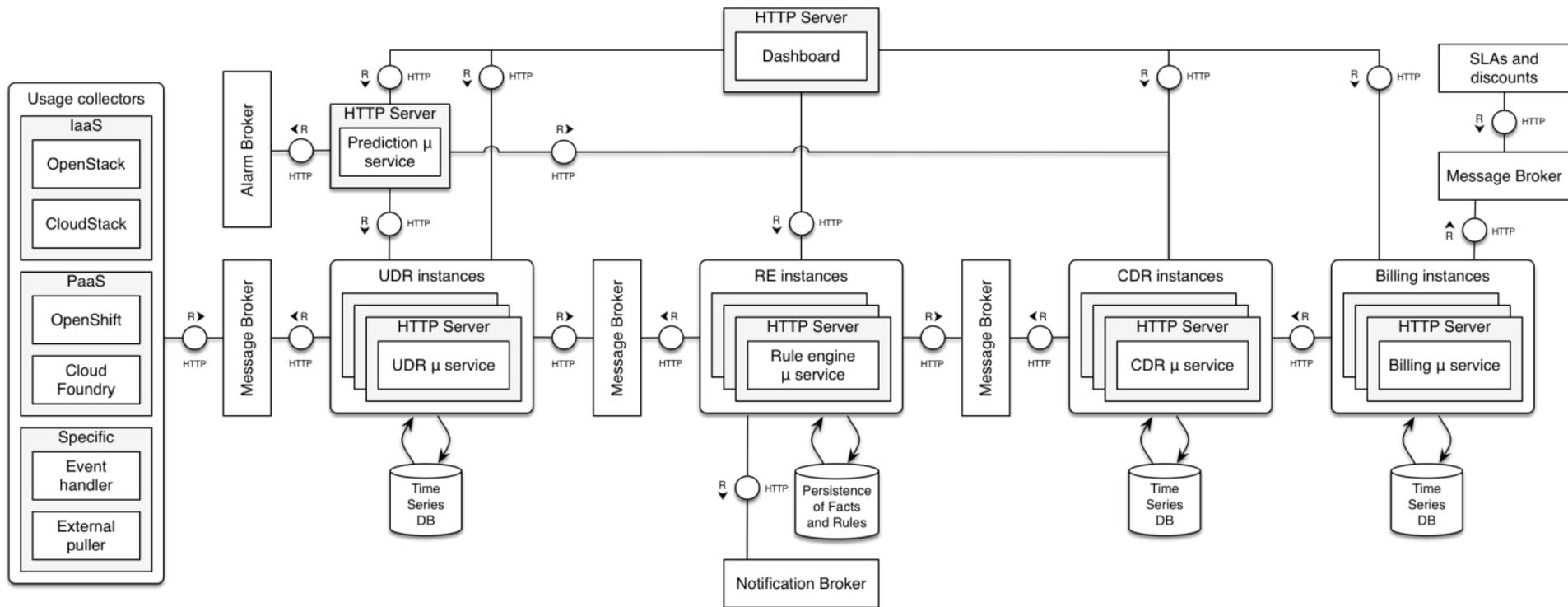
# Mapping Cyclops to generic accounting process



# Mapping Cyclops to generic accounting process



# Architecture (expanded) & workflows



# Cyclops: micro-services



# Microservices: Usage Data Records (UDR)

- Harmonizes the resource consumption data coming from the collectors
- Merges multiple data stream into a single output stream - udr-reports
- Data APIs
  - ◆ Usage data visualization in the dashboard
  - ◆ Data prediction based on historic trends



# Data transformation in UDR

```
{
  "metadata": {
    "project_id": "c49f7e6c177d44afb38fdaec62be8bc9",
    "resource_id": "935bec67-9f26-4be7-9a01-33c20e66bf00",
    "instance_name": "cyclops-demo"
  },
  "unit": "ns",
  "usage": 38190000000,
  "time": 1474488799,
  "_class": "OpenStackCeilometerCpu",
  "account": "8a3f30bb84a0495987393740264b8064"
}
```

```
{
  "metadata": {
    "project_id": "c49f7e6c177d44afb38fdaec62be8bc9",
    "resource_id": "935bec67-9f26-4be7-9a01-33c20e66bf00",
    "instance_name": "cyclops-demo"
  },
  "unit": "ns",
  "usage": 84980000000,
  "time": 1474537399,
  "_class": "OpenStackCeilometerCpu",
  "account": "8a3f30bb84a0495987393740264b8064"
}
```

Transforms into

```
{
  "measurement": "UDR",
  "data": [
    {
      "data": [
        {
          "metadata": {
            "instance_name": "cyclops-demo",
            "project_id": "c49f7e6c177d44afb38fdaec62be8bc9",
            "resource_id": "935bec67-9f26-4be7-9a01-33c20e66bf00"
          },
          "_class": "OpenStackCeilometerCpu",
          "usage": 123170000000,
          "account": "8a3f30bb84a0495987393740264b8064",
          "unit": "ns"
        }
      ],
      "time": 1474537464,
      "account": "8a3f30bb84a0495987393740264b8064",
      "from": 1474488799,
      "_class": "UDR",
      "to": 1474535599
    }
  ],
  "displayedRecords": 1,
  "totalRecords": 1,
  "pageNumber": 0,
  "pageSize": 500
}
```

# Microservices: Rating & Charging (RC)

Made of rule-engine + data persistence and query interface

→ Converts UDR into charge records

- ◆ Pricing derived from business model encoded as rules
- ◆ Rule Engine maintains real time facts in memory (or in db as per configuration)
- ◆ Easy to configure tiered rates, and customized pricing for “special” customers

→ Rating and charging is truly generic

- ◆ You configure the rules, as per your business model
- ◆ Completely RESTful interface
- ◆ Highly programmable

→ CDRs are stored back for visualization (time series)



# Data transformation in CDR

```
{
  "measurement": "UDR",
  "data": [
    {
      "data": [
        {
          "metadata": {
            "instance_name": "cyclops-demo",
            "project_id": "c49f7e6c177d44afb38fdaec62be8bc9",
            "resource_id": "935bec67-9f26-4be7-9a01-33c20e66bf00"
          },
          "_class": "OpenStackCeilometerCpu",
          "usage": 12317000000,
          "account": "8a3f30bb84a0495987393740264b8064",
          "unit": "ns"
        }
      ],
      "time": 1474537464,
      "account": "8a3f30bb84a0495987393740264b8064",
      "from": 1474488799,
      "_class": "UDR",
      "to": 1474535599
    }
  ],
  "displayedRecords": 1,
  "totalRecords": 1,
  "pageNumber": 0,
  "pageSize": 500
}
```



```
{
  "data": [
    {
      "metadata": {
        "instance_name": "cyclops-demo",
        "project_id": "c49f7e6c177d44afb38fdaec62be8bc9",
        "resource_id": "935bec67-9f26-4be7-9a01-33c20e66bf00"
      },
      "_class": "OpenStackCeilometerCpu",
      "usage": 12317000000,
      "account": "8a3f30bb84a0495987393740264b8064",
      "charge": 12
    }
  ],
  "time": 1474537469,
  "account": "018d1f191f6e4e27b726c84a51e0d248",
  "from": 1474488799,
  "_class": "CDR",
  "charge": 12,
  "stateful": false,
  "to": 1474535599
}
```

# Microservices: Billing

Made of rule-engine + data persistence and query interfaces

- Handles customer retention policies (discounts, rebates, promotions etc.)
- SLA violations and penalties
- Geographical rules (VATs, sales-tax, currencies, etc.)
- Federated billing (experimental)
  - ◆ Combines CDR records belonging to the list of users into same consolidated bill
  - ◆ Example: Input - List[ $\{\text{user-id}_{\text{provider-A}}\}$ ,  $\{\text{user-id}_{\text{provider-B}}\}$ , ...], Output: Federated Bill Record



# Microservices: Rule Engine

Allows template instantiation and truth maintenance over RESTful APIs & allows business developers to write their own business rules for data driven rule execution (advances cyclops generality).

The most prominent features are:

→ Supports rules with priorities

→ Stateless and stateful sessions

◆ Live counters allow tiered rates, volume discounts, etc.

→ Ability to unpack incoming data stream and repack transformed data into output stream



# Example rule insertion

```
curl -X "POST" "http://localhost:$1/ruleengine/rule" -H "Content-Type: text/plain" -d '$import
ch.icclab.cyclops.facts.model.rcb.usagedata.openstack.ceilometer.OpenStackCeilometerCpuUtil;
import ch.icclab.cyclops.facts.model.rcb.chargedata.ChargeData;

rule "Rate Ceilometer CPU Util"
saliency 10
when
  $usage: OpenStackCeilometerCpuUtil()
then
  ChargeData charge = new ChargeData($usage);
  charge.setCharge(20);
  insert(charge);
  retract($usage);
end'
```

- Give the rule a name!
- Set the weight, if multiple rules' entry conditions are satisfied then the one with higher "weight" is executed
- The micro-service unpacks the UDR, applies rules to each element and repacks and stores



# Microservices: Gatekeeper

<http://icclab.github.io/gatekeeper/>

## ❖ RBAC microservice written in golang!

- Supports 1000s of API calls / minute
- User+process authentication and authorization
- Token generation and management (validation, revocation, etc.)
- Support for arbitrary set of roles, and fine-grained authorization granularity (per API endpoint, service, object, etc.)
- Inter-services authorization support
- RESTful APIs for ease of integration and programmability



Gatekeeper also features in these projects

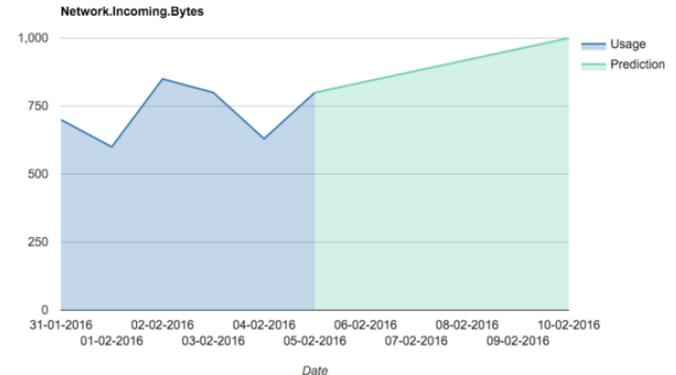
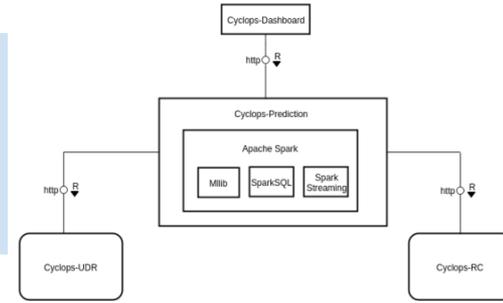


# Microservices: Prediction Engine

Designed to predict values based on historical data but also capable of generic factual forecasting by encapsulating mllib and openforecast into a singular prediction engine.

- Mllib - linear regression
- OpenForecast - enables selection of best forecasting algorithm based on input data samples
  - ◆ Linear regression, exponential smoothing, naive bayes, etc.
- Provides usage forecasting to service end users

Planned: Resource planning tool for administrators, providers!



# UI Tools: Dashboard

A fully function component allows all options of a billing process to be managed

- Differentiated views: admin view + user view
- Authenticated access (currently uses Keystone / Gatekeeper)
- For end users
  - ◆ Allows access to usage and charge graphs
  - ◆ Usage and cost prediction
  - ◆ Current and past bills
- For cloud providers (under development in dashboard 2.0 branch)
  - ◆ Selection and management of meters, both built-in ceilometer meters or external meters
  - ◆ Pricing control
  - ◆ Bill generation and release control (making a bill visible to the customer after validation)



# Cyclops Dashboard (then)

Cyclops Dashboard

Search...

Usage

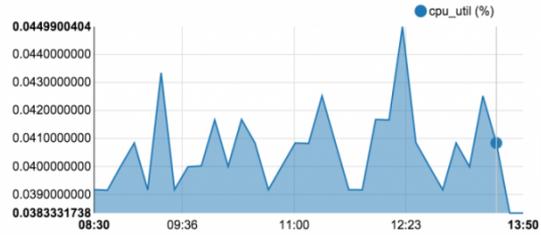
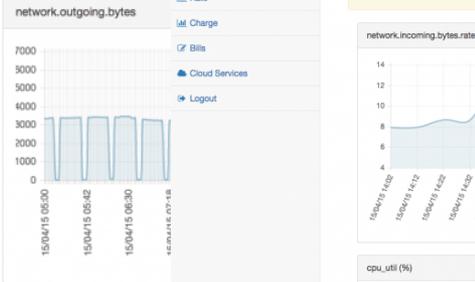
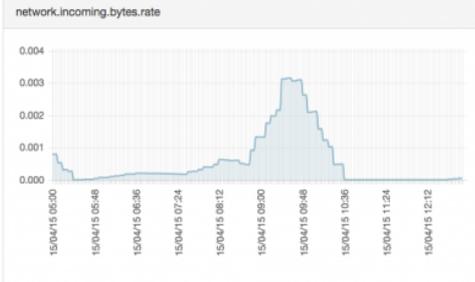
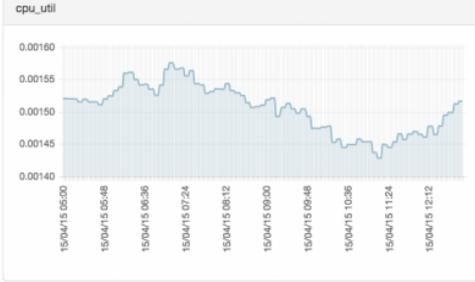
Rate

Charge

Bills

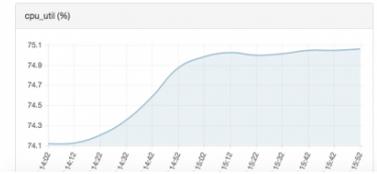
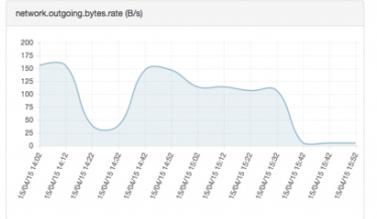
Cloud Services

Logout



13:30  
■ cpu\_util (%) 0.0408334211

Currently displaying:  
Last 6 Hours



# Cyclops Dashboard (then)

The screenshot displays the Cyclops Dashboard interface, which is divided into several sections:

- Administrators:** A table listing administrators with their usernames and a "Revoke Admin Rights" button for each. The listed administrators are piyush, trui, trui3, pata, and trui2.
- Users:** A table listing users with their usernames. The listed users are amAdmin, trui5, demo, and charts.
- Configuration:** A sidebar on the left contains various configuration options, including Usage, Rate, Charge, Bills, Cloud Services, UDR Configuration, User Management, Meter Configuration, RC Configuration, and Billing Configuration.
- Monitoring:** A central area displays a list of monitoring metrics, each with a checkbox. The metrics are organized into three columns. The first column includes metrics like cpu, disk, image, instance, network, port, router, storage, subnet, and volume. The second column includes metrics like cpu\_util, disk, image, instance, network, and volume. The third column includes metrics like disk, image, instance, ip, network, port, router, storage, subnet, and vcpus. A "Save" button is located at the bottom right of this section.

**Administrators Table:**

Username	Action
piyush	Revoke Admin Rights
trui	Revoke Admin Rights
trui3	Revoke Admin Rights
pata	Revoke Admin Rights
trui2	Revoke Admin Rights

**Users Table:**

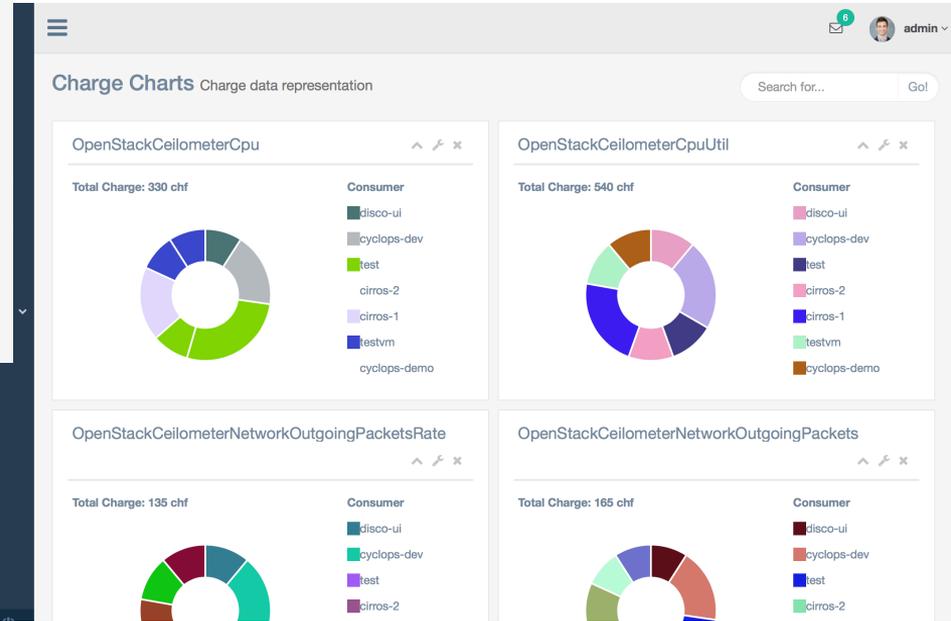
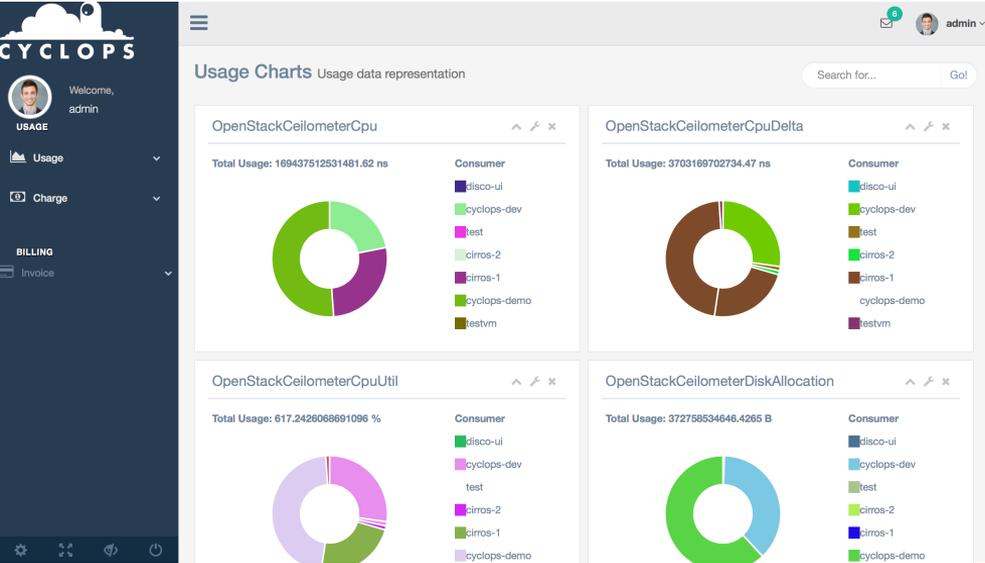
Username
amAdmin
trui5
demo
charts

**Monitoring Metrics:**

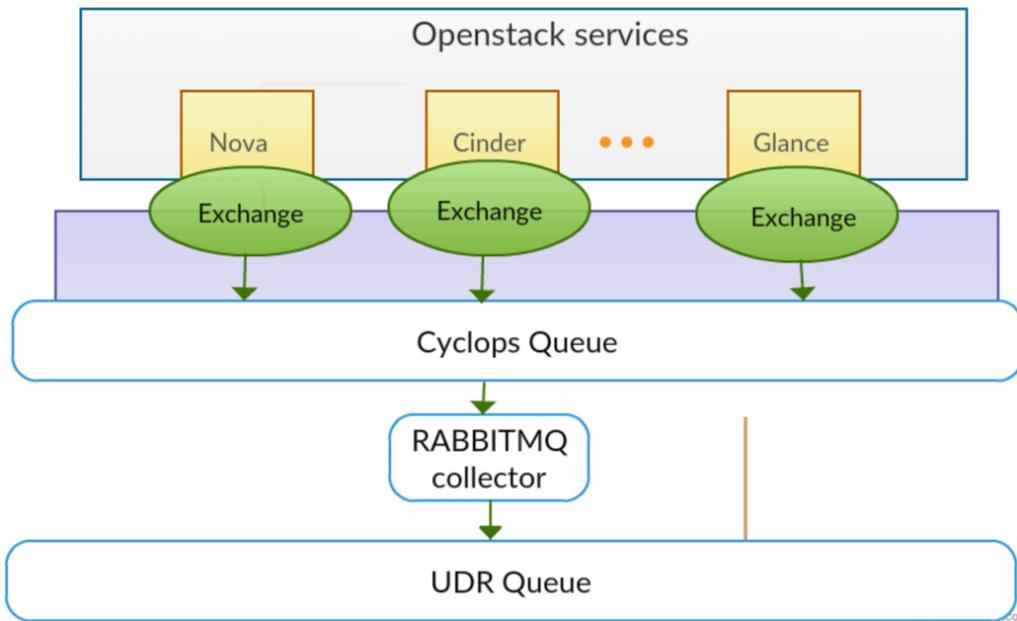
- cpu
- disk.device.read.requests
- disk.ephemeral.size
- disk.read.requests
- disk.write.bytes
- disk.write.requests.rate
- image.download
- image.update
- instance:m1.large
- instance:m1.tiny
- ip.floating.update
- network.create
- network.incoming.packets
- network.outgoing.bytes.rate
- network.services.lb.active.connections
- network.services.lb.member
- network.services.lb.total.connections
- port.create
- router.create
- storage.objects.containers
- subnet.create
- volume
- cpu\_util
- disk.device.write.bytes
- disk.read.bytes
- disk.read.requests.rate
- disk.write.bytes.rate
- image
- image.serve
- image.upload
- instance:m1.medium
- ip.floating
- memory
- network.incoming.bytes
- network.incoming.packets.rate
- network.outgoing.packets
- network.services.lb.health\_monitor
- network.services.lb.outgoing.bytes
- network.services.lb.vip
- port.update
- router.update
- storage.objects.size
- subnet.update
- volume.size
- disk.device.read.bytes
- disk.device.write.requests
- disk.read.bytes.rate
- disk.root.size
- disk.write.requests
- image.delete
- image.size
- instance
- instance:m1.small
- ip.floating.create
- network
- network.incoming.bytes.rate
- network.outgoing.bytes
- network.outgoing.packets.rate
- network.services.lb.incoming.bytes
- network.services.lb.pool
- port
- router
- storage.objects
- subnet
- vcpus



# Cyclops Dashboard (now, under development)



# Cyclops collector: OpenStack events



→ Billing now possible based on key Nova events

- ◆ VM start
- ◆ VM terminated
- ◆ VM suspended
- ◆ VM resumed

→ Key VM metadata is now processed

- ◆ VM flavor information

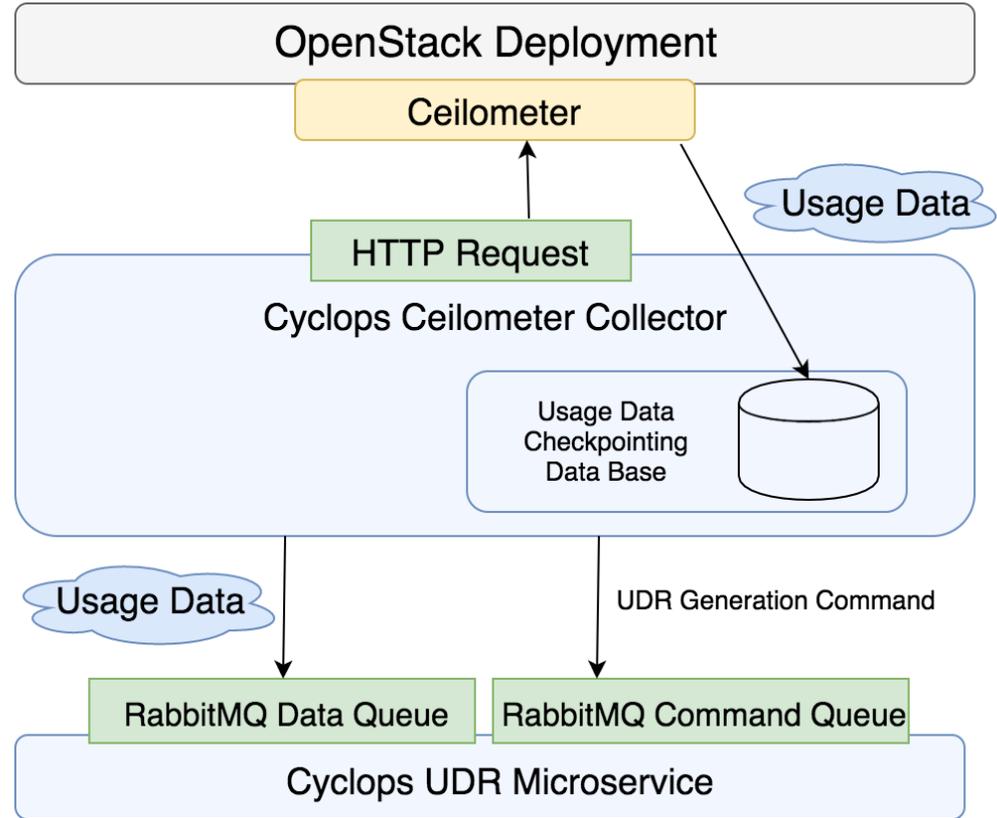
It is now possible define pricing rules like

- x.yy CHF / hr for m1.large VM
- ★ Extension of similar capabilities for events from Glance, Neutron are under development



# Cyclops collector: OpenStack Ceilometer

- Harmonizes all data sources
  - ◆ Proper handling of -
    - Cumulative
    - Delta
    - Gauge meters
- Computation of actual usage done within the collector itself
  - ◆ UDR micro-service treats all data sources as usage streams



# List of supported platforms

## ❖ OpenStack

- Ceilometer Collector
- Events collector (taps into OpenStack messaging bus)
- Tests with Mitaka release underway

## ❖ CloudStack

- Usage collector supports all built-in meters

## PS: External Meters

→ Any non-natively supported framework can be billed with Cyclops

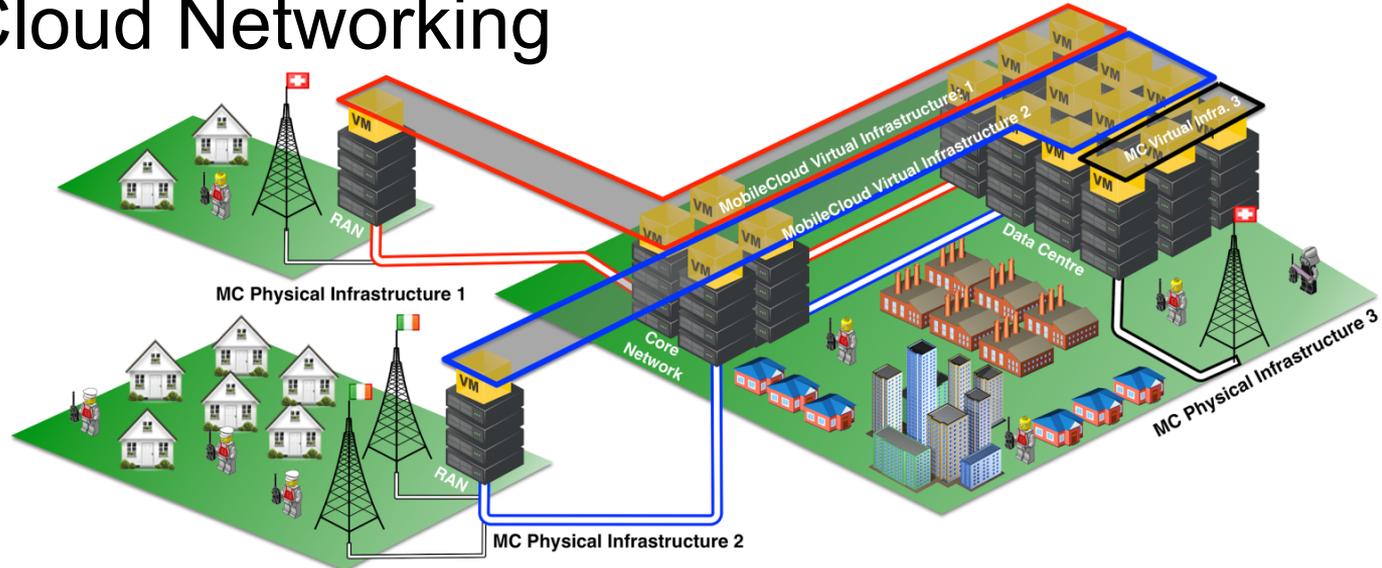
- ◆ Use external meters: define a unique meter-name, implement the collector for the external framework, and modify the rules in the rule-engine. THAT'S IT!



# Cyclops Demonstrable Use Cases



# Mobile Cloud Networking

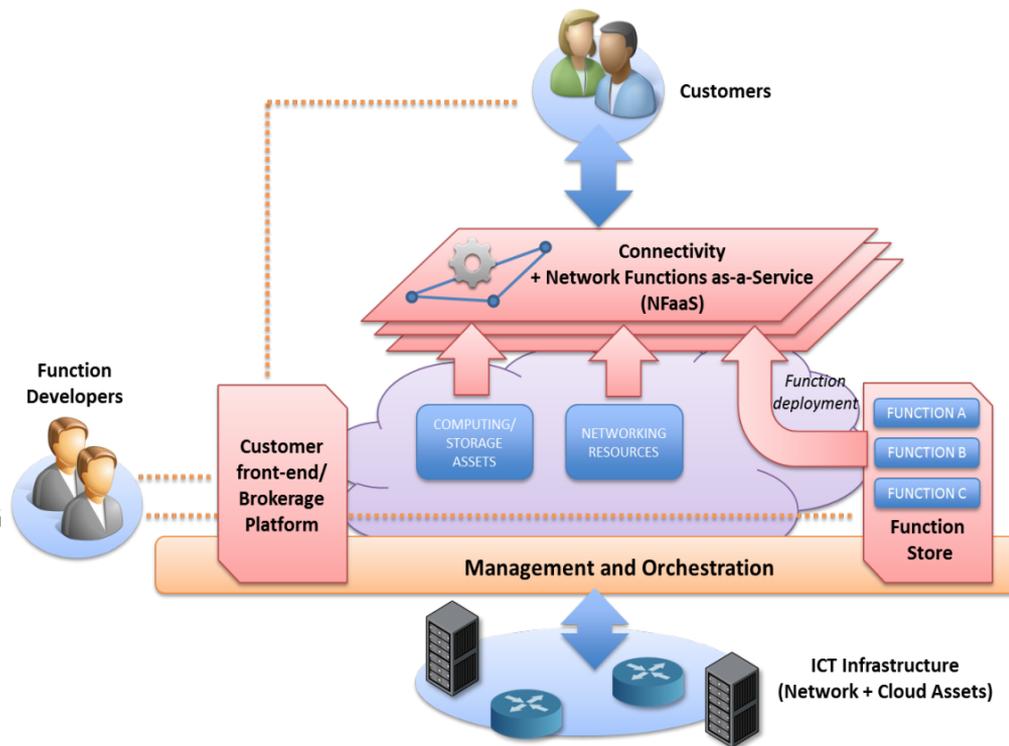


Rating-charging-billing for composed telco-functions deployed over OS

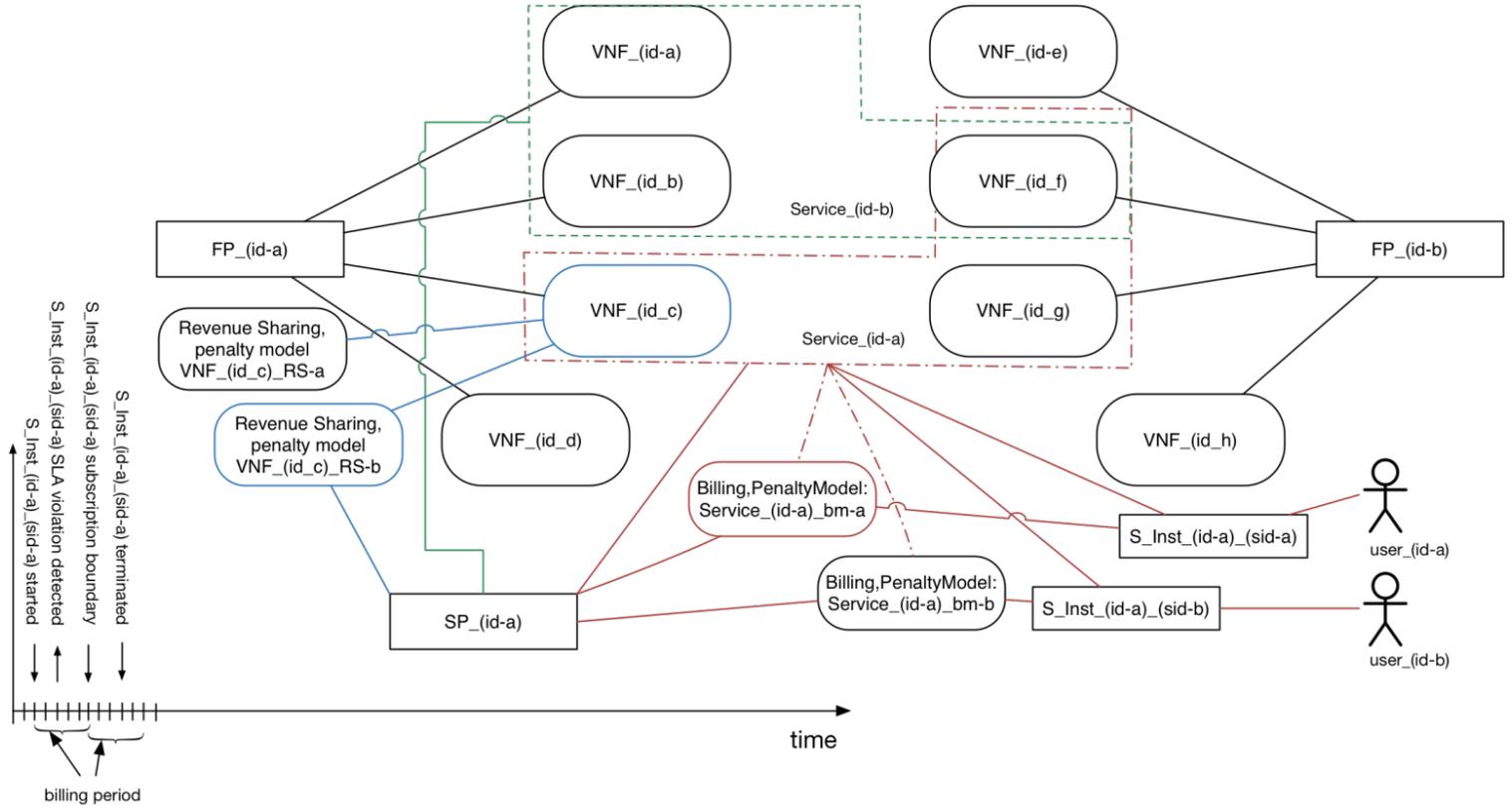
- {RAN + EPC + IMS + Support Services (mobility predictor, CDN, db, DNS, etc.)}aaS
- Rating, charging based on events (start, stop, resume, suspend) + OpenStack usage
  - ◆ Implemented using cyclops via external meters + built in Ceilometer collector!
  - ◆ Converged billing (OpenStack + on-demand provisioned telco services)

# T-NOVA

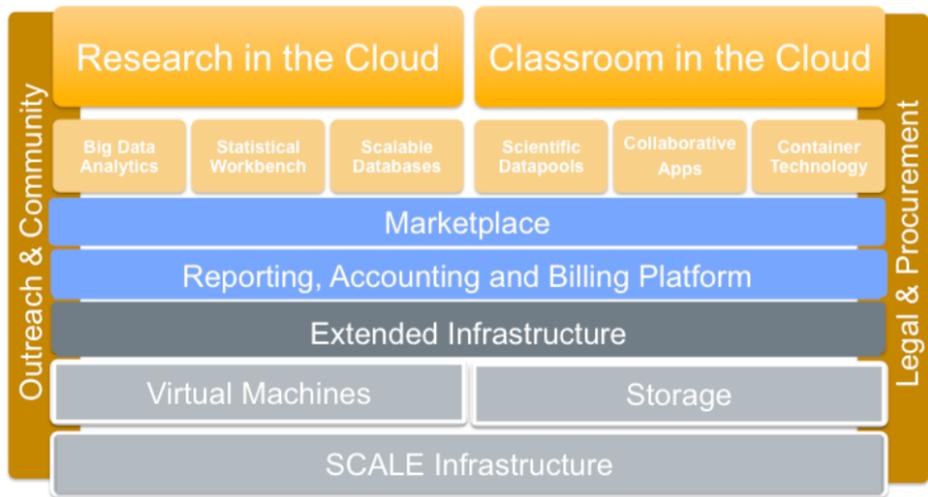
- ❖ Billing support for VNF marketplace
  - Multi-actor environment - service provider, enterprise customers, function developers
  - Negotiated revenue sharing models had to be supported
  - Penalty models based on SLA violations.
  - Billing models: subscription or PAYG
- ❖ Cyclops easily integrated with accounting & sla-m by ATOS
- ❖ Billing based on key lifecycle events



# T-NOVA - Billing Objects



# SCALE-UP Project - Organizational Hierarchies



Rule engine easily encodes logic for organization hierarchies and users costs are automatically aggregated in the bill!

## ❖ Billing in academia

- Organization level billing: university, research labs, academic units, users.
- Cost trackable at sub-organization level
- Tracking of resource usages per user
  - compute, storage, network, software licenses, support, etc.

## ❖ Supporting academic marketplace over OpenStack cloud

- Revenue sharing, virtual credits, penalties upon sla violations, etc.
- Developer friendly APIs

## ❖ Rating and charging of compute clusters - Hadoop, Spark, etc.

<https://projects.switch.ch/scale-up/>

# SSC: Marketplace, product bundling

→Ecosystem with ISVs, White label resellers and distributors

- ◆ Revenue sharing models are to be supported

→Marketplace

- ◆ Individualized billing models for marketplace offered services and appliances

- Custom meter creation, pricing rules, metrics collection and processing

- ◆ Product bundles

- New services and offerings to be created from existing appliances or products

- Billing at the level of product bundles

→Ability to track usage at elemental level of any given composition

**Cyclops rule engine handles all these specific needs, and UDR reports are generated as long as collectors are written correctly.**





Time for a quick demo

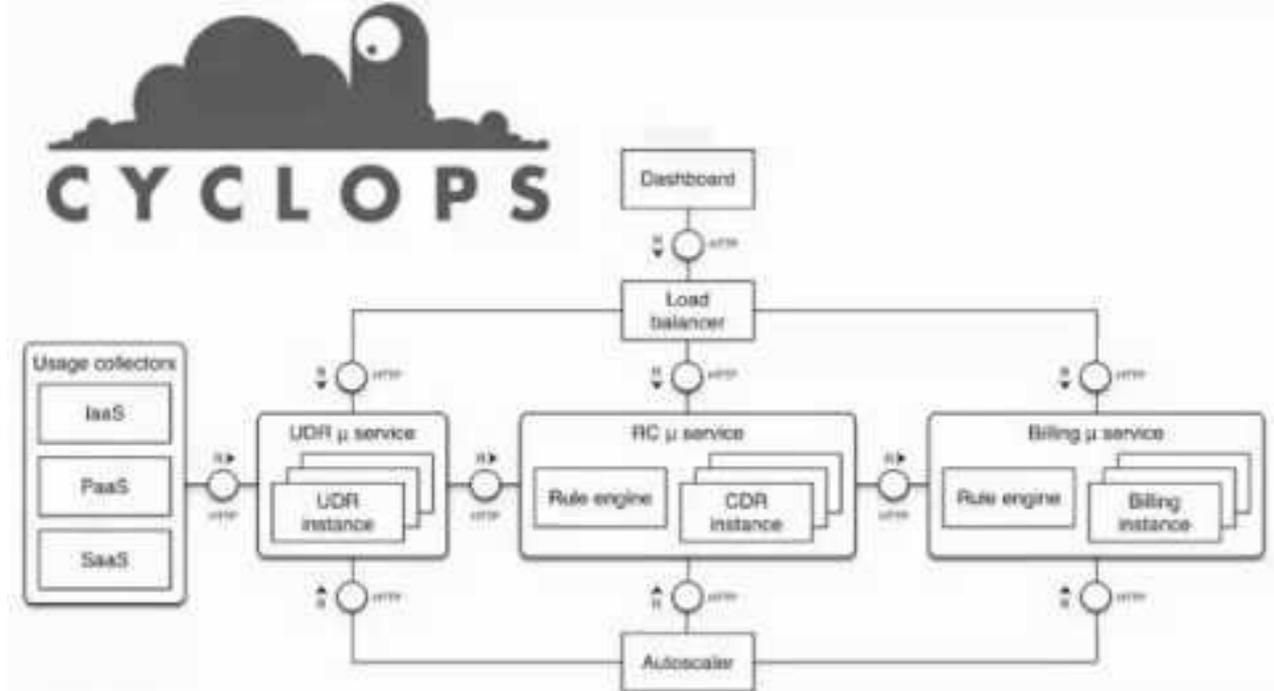
What you see in this video?

### Rule management

- Starting of services
- Rule engine initialization
- Insertion of usage data
- Generation of consolidated bill

### Dashboard

- Usage & Charge Views



# Cyclops: Dev & deployment support

Every microservice has been containerized.

→ Docker build images and docker-compose.yml files are already released as open source

→ Search for *cyclopsbilling* in hub.docker.com

→ Extensive developer's guide to extend the framework has been released

→ Automated build scripts for each microservice is available.



**CYCLOPS**

cyclopsbilling

© Joined April 2016

 <a href="#">cyclopsbilling/cyclops-postgres</a> public   automated build	0 STARS	22 PULLS	> DETAILS
 <a href="#">cyclopsbilling/cyclops-gatekeeper</a> public   automated build	0 STARS	20 PULLS	> DETAILS
 <a href="#">cyclopsbilling/cyclops-influxdb</a> public   automated build	0 STARS	20 PULLS	> DETAILS
 <a href="#">cyclopsbilling/cyclops-rabbitmq</a> public   automated build	0 STARS	20 PULLS	> DETAILS
 <a href="#">cyclopsbilling/cyclops-base</a> public   automated build	0 STARS	16 PULLS	> DETAILS



# What next with Cyclops?

Billing for smart IoT deployments

→ Managing data-gaps in streams

- ◆ Devices may not be connected always for data offloading

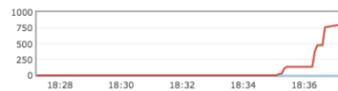
→ Corrupt data identification, tagging & management

We wrote our own IoT deployment simulation tool!

## Queue cyclops-test

### Overview

Queued messages (chart: last ten minutes) (?)



Message rates (chart: last ten minutes) (?)



### Details

Features	durable: true	State	Idle
Policy		Consumers	0
		Consumer utilisation (?)	0%
Messages (?)	Total: 806	Ready: 806	Unacked: 0
Message body bytes (?)	108kB	108kB	0B
		In memory	806

```
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ IoT-Simulator ---
[INFO] Deleting /Users/harh/Code/java/gitlab/iot-simulator/IoT-Simulator/target
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ IoT-Simulator ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 2 resources
[INFO] --- maven-compiler-plugin:2.5.1:compile (default-compile) @ IoT-Simulator ---
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 8 source files to /Users/harh/Code/java/gitlab/iot-simulator/IoT-Simulator/target/classes
[INFO] --- maven-assembly-plugin:2.2-beta-5:single (default-cli) @ IoT-Simulator ---
[INFO] Building jar: /Users/harh/Code/java/gitlab/iot-simulator/IoT-Simulator/target/IoT-Simulator-0.01-jar-with-dependencies.jar
[INFO] BUILD SUCCESS
[INFO] Total time: 5.010 s
[INFO] Finished at: 2016-03-16 18:36:09
```

<http://icclab.github.io/iot-simulator/>



# Further information ...

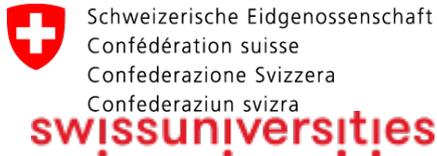
Cyclops Page: <http://icclab.github.io/cyclops/>

Gatekeeper: <http://icclab.github.io/gatekeeper/>

ICCLab RCB initiative page: <http://bit.ly/1WYC27I>



@rcb\_cyclops - follow us on twitter.



Cyclops has been proudly funded by these organizations / projects