Maestro Server - Cloud Inventory Documentation

Versão 0.1

Felipe Signorini

18 jan, 2019

Contents:

The docs its be separated into 3 parts, the first is about installation and setup Maestro, the second is about User Guide how you create and manage Maestro in the business point of view, and the last we have a developer guide for people like to contribute for the project.

CAPÍTULO 1

Visão Geral

1.1 O que é o Maestro Server?

Maestro Server is an open source software platform for management and discovery servers, apps and system for Hybrid IT. Can manage small and large environments, be able to visualize the latest multi-cloud environment state.

You will be able to:

- Centralize and visualize the lastest state multi-cloud environment
- Continuously discover new servers and services of all environments
- Powerful reports, can create a relation with servers, services, apps and clients
- Automatically populate inventory with ansible, logging jobs, audit and cordenate multiple teams.
- Tracking all changes of your infrastructure

1.2 What problems does it solve?

Maestro had built to solve some problems founded in operating multi-cloud environments, multi shared devops culture and multi clients, where turns hard to keep track the lastest environment state, bottlenecks to apply a compliance in all teams, visualization gaps to understand your's infrastructure state, access security flaws for internals employees and out of date documentation.

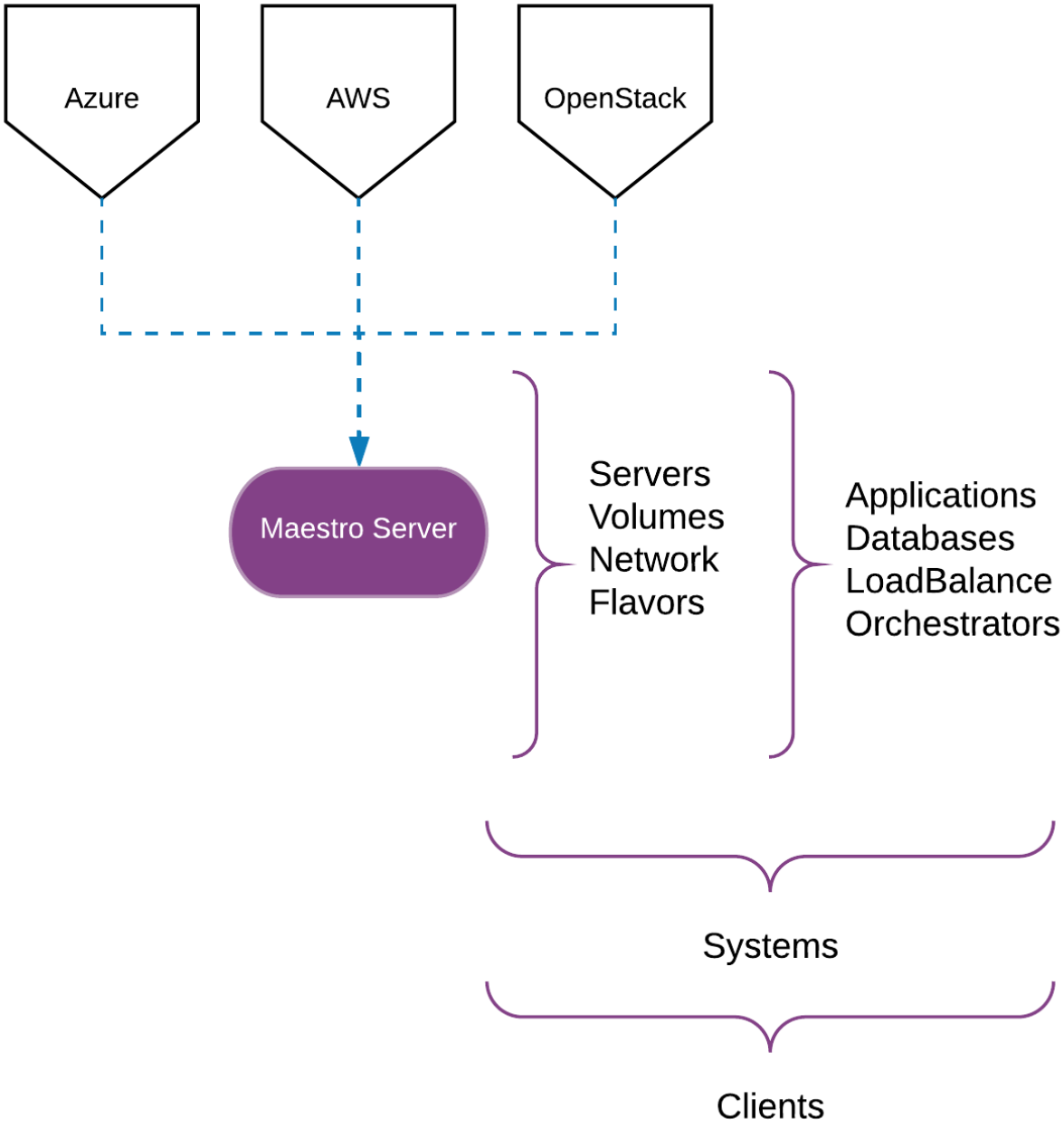
- How we can audit your's env?
- How control and keep track your's environment?
- How garantee if my documentation is updated?
- Witch servers belongs to this client?

Maestro comes to help IT operation teams to organize and audit multicloud infrastructure, come to substitute CMDB systems, auto-discovery servers, services and apps, be organizing in smart way, it's possible to classify each service, like database, message queues, vpns, api gateway, service mesh and etc, create a relation between servers and services,

docs clusters, and points target, relate services, system and clients. Maestro come for you, to be a complete and simple cloud inventory.

1.3 Como posso usá-lo?

Analysis your's full state environment of all providers do you have, centralize all information about datacenters, servers, loadbalance, orchestrations tools, volumes, vpns and etc, keep track their relations, can create complex and powerful reports, analysis costs, growing up velocity, standards services names, network configurations, available deploys for each server.



See demo cloud inventory here.

CAPÍTULO 2

Quick Start

Get Maestro up in just a few minutes, we recommend to use docker, but if you like to install direcly read the installation section.

2.1 Overview

List of micro service:

Client App	FrontEnd client	Vue2 + Bootstrap 3
Server App	Primary API, authetication, crud and manager	NodeJs 8.11 Kraken
Discovery App	Auto discovery and crawlers	Python 3.6, flask
Scheduler App	Jobs manager with celery beat	Python 3.6, celery
Reports App	Reports generetor	Python 3.6, flask
Analytics App	Analytics Maestro - Graphs Generator	Python 3.6, flask
Analytics Front	Analytics Front	NodeJs 8.11 Kraken
Data DB App	Data layer	Python 3.6, flask
Audit App	HIstory tracker service	NodeJs 8.11 Kraken
WebSocket APP	WebSocket - Events	Go, Centrifugo

2.2 Running locally

We recommend to use docker, if you like to see demo version, copy and execute docker-compose below, you need to change only two variable in client-app, url, and port.

Nota: PS: Docker will be created and manager all networks and communication between services.

PS: The containers its prepared for run in production ready, but its recommend to create a separate database environment and export the volume (remember all storage inside of docker its temporary)

Nota: Download docker-compose file.

Aviso: This is quickstart, it's a docker compose to setup fast in local machines, if you like to install in production env, go to installing guide.

```
version: '2'

services:
    client:
        image: maestroserver/client-maestro
        ports:
        - "80:80"
        environment:
        - "API_URL=http://localhost:8888"
        - "STATIC_URL=http://localhost:8888/static/"
        - "ANALYTICS_URL=http://localhost:9999"
        - "WEBSOCKET_URL=ws://localhost:8000"
        depends_on:
        - server

    server:
        image: maestroserver/server-maestro
        ports:
        - "8888:8888"
        environment:
        - "MAESTRO_MONGO_URI=mongodb"
        - "MAESTRO_MONGO_DATABASE=maestro-client"
        - "MAESTRO_DISCOVERY_URI=http://discovery:5000"
        - "MAESTRO_ANALYTICS_URI=http://analytics:5020"
        - "MAESTRO_ANALYTICS_FRONT_URI=http://analytics_front:9999"
        - "MAESTRO_REPORT_URI=http://reports:5005"
        - "SMTP_PORT=25"
        - "SMTP_HOST=maildev"
        - "SMTP_SENDER=myemail@gmail.com"
        - "SMTP_IGNORE=true"
        depends_on:
        - mongodb
        - discovery
        - reports

    discovery:
        image: maestroserver/discovery-maestro
        ports:
        - "5000:5000"
        environment:
        - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
        - "MAESTRO_DATA_URI=http://data:5010"
        depends_on:
        - rabbitmq
        - data

    discovery_worker:
        image: maestroserver/discovery-maestro-celery
        environment:
```

(continues on next page)

(continuação da página anterior)

```
        - "MAESTRO_DATA_URI=http://data:5010"
        - "MAESTRO_WEBSOCKET_URI=http://ws:8000"
        - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
        depends_on:
        - rabbitmq
        - data

    reports:
        image: maestroserver/reports-maestro
        environment:
        - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
        - "MAESTRO_MONGO_URI=mongodb"
        - "MAESTRO_MONGO_DATABASE=maestro-reports"
        depends_on:
        - rabbitmq
        - mongodb

    reports_worker:
        image: maestroserver/reports-maestro-celery
        environment:
        - "MAESTRO_REPORT_URI=http://reports:5005"
        - "MAESTRO_DATA_URI=http://data:5010"
        - "MAESTRO_WEBSOCKET_URI=http://ws:8000"
        - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
        depends_on:
        - rabbitmq
        - data

    scheduler:
        image: maestroserver/scheduler-maestro
        environment:
        - "MAESTRO_DATA_URI=http://data:5010"
        - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
        - "MAESTRO_MONGO_URI=mongodb"
        - "MAESTRO_MONGO_DATABASE=maestro-client"
        depends_on:
        - mongodb
        - rabbitmq

    scheduler_worker:
        image: maestroserver/scheduler-maestro-celery
        environment:
        - "MAESTRO_DATA_URI=http://data:5010"
        - "MAESTRO_DISCOVERY_URI=http://discovery:5000"
        - "MAESTRO_ANALYTICS_URI=http://analytics:5020"
        - "MAESTRO_REPORT_URI=http://reports:5005"
        - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
        depends_on:
        - rabbitmq
        - data

    analytics:
        image: maestroserver/analytics-maestro
        ports:
        - "5020:5020"
        environment:
        - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
```

(continues on next page)

(continuação da página anterior)

```
      - "MAESTRO_DATA_URI=http://data:5010"
    depends_on:
    - rabbitmq
    - data

analytics_worker:
    image: maestroserver/analytics-maestro-celery
    environment:
    - "MAESTRO_DATA_URI=http://data:5010"
    - "MAESTRO_ANALYTICS_FRONT_URI=http://analytics_front:9999"
    - "MAESTRO_WEBSOCKET_URI=http://ws:8000"
    - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
    - "CELERYD_MAX_TASKS_PER_CHILD=2"
    depends_on:
    - rabbitmq
    - data

analytics_front:
    image: maestroserver/analytics-front-maestro
    ports:
    - "9999:9999"
    environment:
    - "MAESTRO_MONGO_URI=mongodb"
    - "MAESTRO_MONGO_DATABASE=maestro-client"

data:
    image: maestroserver/data-maestro
    environment:
    - "MAESTRO_MONGO_URI=mongodb"
    - "MAESTRO_MONGO_DATABASE=maestro-client"
    depends_on:
    - mongodb

audit:
    image: maestroserver/audit-app-maestro
    environment:
    - "MAESTRO_MONGO_URI=mongodb"
    - "MAESTRO_MONGO_DATABASE=maestro-audit"
    - "MAESTRO_DATA_URI=http://data:5010"

ws:
    image: maestroserver/websocket-maestro
    ports:
    - "8000:8000"

rabbitmq:
    hostname: "discovery-rabbit"
    image: rabbitmq:3-management
    ports:
    - "15672:15672"
    - "5672:5672"

mongodb:
    image: mongo
    volumes:
    - mongodata:/data/db
    ports:
```

(continues on next page)

(continuação da página anterior)

```
        - "27017:27017"

    maildev:
        image: djfarrelly/maildev
        mem_limit: 80m
        ports:
        - "1025:25"
        - "1080:80"


volumes:
    mongodata: {}
```

Nota: Remember to config API_URL and STATIC_URL on client app with ip/dns of your server.

2.3 Vagrant

We have VagrantFile, its good for visualization (demo) or the best way to create a development environment.

Nota: Download vagrantFile.

Nota: HA - High availability and critical system

If your necessity is, HA, critical situation, go in Ha session.

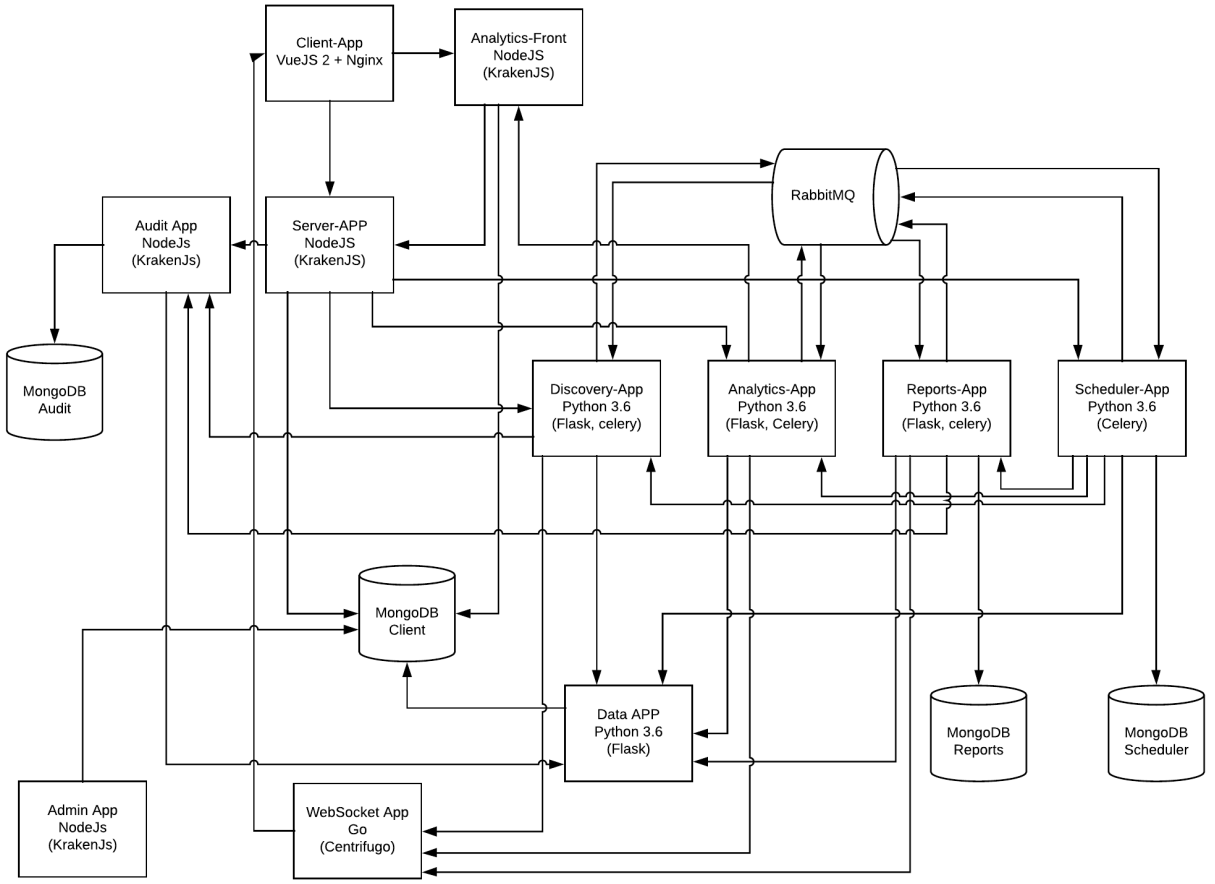
CAPÍTULO 3

Installing Maestro

We have docker compose file with all services (download here), this is the easy way to install Maestro, if you like can install in a pure way (we did a doc show each step to install without docker, see here Developer Guide).

This section will show installation briefing for each service.

3.1 High Architecture



First: A minimun installation can be done with:

- Client App
- Server App
- MongoDB

You can setup and use a minimun installation, you can create and delete servers, apps, datacenters, change acl and create new users, with these you have a simple inventory system.

If you like to use a synchronous features with AWS or other providers, then you need:

- Discovery App
- Data App
- RabbitMq

Or use auto-discovery feature, will polling and maintain or inventory synchronous, then:

- Scheduler App

If you like to create and export reports then:

- Reports App

- Data App
- RabbitMq

Create bussiness analytics graphs, public and shared these maps, need:

- Analytics App
- Analytics Front App
- Data App
- RabbitMq

And if you like to tracking history and smart update beetween entities, should install:

- Audit App

And if you need to manage all data, create and reset passwords, privilege admin stuffs, use:

- Admin App

Let´s start

3.2 Client App

Installation by docker-compose

```
client:
    image: maestroserver/client-maestro
    ports:
    - "80:80"
    environment:
    - "API_URL=http://server-app:8888"
    - "STATIC_URL=http://server-app:8888/static/"
    - "ANALYTICS_URL=http://localhost:9999"
```

```
docker run -p 80:80 -e 'API_URL=http://localhost:8888' -e 'STATIC_URL=http://
↪localhost:8888/static/' -e "ANALYTICS_URL=http://localhost:9999" maestroserver/
↪client-maestro
```

Aviso:

- API_URL it's rest endpoint provide by server-app.
- ANALYTICS_URL it's rest endpoint provide by analytics-front.
- STATIC_URL it's endpoint for static files, if you use local upload type need to be {server-app-url}/static - More details upload.

Env variables

Env Variables	Example	Description
API_URL	http://localhost:8888	Server App Url
STATIC_URL	/static	Relative path of static content
LOGO	/static/imgs/logo300.png	Logotype, (login page)
THEME	theme-lotus	Theme (gold|wine|blue|green|dark)

3.3 Server APP

Installation by docker

```
server:
    image: maestroserver/server-maestro
    ports:
    - "8888:8888"
    environment:
    - "MAESTRO_MONGO_URI=mongodb"
    - "MAESTRO_MONGO_DATABASE=maestro-client"
    - "MAESTRO_DISCOVERY_URI=http://discovery:5000"
    - "MAESTRO_ANALYTICS_URI=http://analytics:5020"
    - "MAESTRO_REPORT_URI=http://reports:5005"
    - "MAESTRO_AUDIT_URI=http://audit:10900"
```

```
docker run -p 8888:8888
    -e "MAESTRO_MONGO_URI=mongodb"
    -e "MAESTRO_MONGO_DATABASE=maestro-client"
    -e "MAESTRO_DISCOVERY_URI=http://localhost:5000"
    -e "MAESTRO_REPORT_URI=http://localhost:5005"
    -e "MAESTRO_ANALYTICS_URI=http://localhost:5020"
    -e "MAESTRO_AUDIT_URI=http://audit:10900"
    maestroserver/server-maestro
```

Aviso:

- MAESTRO_MONGO_URI - Must be uri, mongodb://{MAESTRO_MONGO_URI}/{MAESTRO_MONGO_DATABASE}
- MAESTRO_MONGO_DATABASE - Only mongodb database name (ex: maestro-client)
- SMTP_X - Used for reset emails and accounts, need to be valid SMTP server - More details smtp.
- MAESTRO_UPLOAD_TYPE - Can be local or S3 More details upload.
- MAESTRO_SECRETJWT - Hash to crypt JWT strings and connections between Discovery App (need to be the same)
- MAESTRO_SECRETJWT_PUBLIC - Hash used only do public shared resources, must be different as MAESTRO_SECRETJWT
- MAESTRO_SECRETJWT_PRIVATE - Hash used on private comunication (only beetween services)
- MAESTRO_NOAUTH - Handshake authentication (private request only)

Env variables

Env Variables	Example	Description
MAESTRO_PORT	8888	
NODE_ENV	development|production	
MAESTRO_MONGO_URI	localhost	DB string connection
MAESTRO_MONGO_DATABASE	maestro-client	Database name
MAESTRO_SECRETJWT	XXXX	Secret key - session
MAESTRO_SECRETJWT_FORGOT	XXXX	Secret key - forgot request
MAESTRO_SECRET_CRYPTO_FORGOT	XXXX	Secret key - forgot content
MAESTRO_SECRETJWT_PUBLIC	XXX	Secret key - public shared
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections
MAESTRO_DISCOVERY_URL	http://localhost:5000	Url discovery-app (flask)
MAESTRO_REPORT_URL	http://localhost:5005	Url reports-app (flask)
MAESTRO_ANALYTICS_URI	http://localhost:5020	Url Analytics-app (flask)
MAESTRO_AUDIT_URI	http://localhost:10900	Url Audit-app (krakenjs)
MAESTRO_TIMEOUT	1000	Timeout micro service request
SMTP_PORT	1025	
SMTP_HOST	localhost	
SMTP_SENDER	myemail@XXXX	
SMTP_IGNORE	true|false	
SMTP_USETSL	true|false	
SMTP_USERNAME		
SMTP_PASSWORD		
AWS_ACCESS_KEY_ID	XXXX	
AWS_SECRET_ACCESS_KEY	XXXX	
AWS_DEFAULT_REGION	us-east-1	
AWS_S3_BUCKET_NAME	maestroserver	Bucket name
MAESTRO_UPLOAD_TYPE	S3 or Local	Upload mode
LOCAL_DIR	/public/static/	Where files will be uploaded
PWD	$rootDirectory	PWD process

3.4 Discovery App

Installation by docker

```
discovery:
    image: maestroserver/discovery-maestro
    ports:
    - "5000:5000"
    environment:
    - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
    - "MAESTRO_DATA_URI=http://data:5010"

discovery_worker:
    image: maestroserver/discovery-maestro-celery
    environment:
    - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
    - "MAESTRO_DATA_URI=http://data:5010"
    - "MAESTRO_AUDIT_URI=http://audit:10900"
```

```
docker run -p 5000:5000  -e "MAESTRO_DATA_URI=http://localhost:5010" -e "CELERY_
↪BROKER_URL=amqp://rabbitmq:5672" maestroserver/discovery-maestro

docker run \
    -e "MAESTRO_DATA_URI=http://localhost:5010" \
    -e "CELERY_BROKER_URL=amqp://rabbitmq:5672" \
    -e "MAESTRO_AUDIT_URI=http://localhost:10900" \
    maestroserver/discovery-maestro-celery
```

Aviso:

- MAESTRO_REPORT_URI - Enpoint API of Discovery - default port is 5010
- MAESTRO_DATA_URI - Enpoint API of Data App - default port is 5000
- MAESTRO_AUDIT_URI - Endpoint API of Audit App - default port is 10900
- MAESTRO_SECRETJWT - Hash to crypt JWT strings and connections between Server App (need to be the same)

Env variables

Env Variables	Example	Description
MAESTRO_PORT	5000	Port used
MAESTRO_DATA_URI	http://localhost:5010	Data Layer API URL
MAESTRO_AUDIT_URI	http://localhost:10900	Audit App - API URL
MAESTRO_WEBSOCKET_URI	http://localhost:8000	Webosocket App - API URL
MAESTRO_SECRETJWT	XXX	Same that Server App
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections
MAESTRO_WEBSOCKET_SECRET	XXX	Secret Key - JWT Websocket connections
MAESTRO_TRANSLATE_QTD	200	Prefetch translation process
MAESTRO_GWORKERS	2	Gunicorn multi process
CELERY_BROKER_URL	amqp://rabbitmq:5672	RabbitMQ connection
CELERYD_TASK_TIME_LIMIT	10	Timeout workers

3.5 Reports App

Installation by docker

```
reports:
    image: maestroserver/reports-maestro
    ports:
    - "5005:5005"
    environment:
    - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
    - "MAESTRO_MONGO_URI=mongodb"
    - "MAESTRO_MONGO_DATABASE=maestro-reports"

reports_worker:
    image: maestroserver/reports-maestro-celery
    environment:
```

(continues on next page)

(continuação da página anterior)

```
    - "MAESTRO_REPORT_URI=http://reports:5005"
    - "MAESTRO_DATA_URI=http://data:5010"
    - "MAESTRO_AUDIT_URI=http://audit:10900"
    - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
```

Aviso:

- MAESTRO_REPORT_URI - Enpoint API of Reports - default port is 5005
- MAESTRO_DATA_URI - Enpoint API of Data App - default port is 5000
- MAESTRO_AUDIT_URI - Endpoint API of Audit App - default port is 10900

```
docker run -p 5005 -e "MAESTRO_DATA_URI=http://localhost:5010" -e "CELERY_BROKER_
↪URL=amqp://rabbitmq:5672" -e 'MAESTRO_MONGO_URI=localhost' maestroserver/reports-
↪maestro

docker run \
    -e "MAESTRO_DATA_URI=http://localhost:5010" \
    -e "MAESTRO_REPORT_URI=http://localhost:5005" \
    -e "CELERY_BROKER_URL=amqp://rabbitmq:5672" \
    -e "MAESTRO_AUDIT_URI=http://audit:10900" \
    maestroserver/reports-maestro-celery
```

Env variables

Env Variables	Example	Description
MAESTRO_PORT	5005	Port used
MAESTRO_MONGO_URI	localhost	Mongo Url conn
MAESTRO_MONGO_DATABASE	maestro-reports	Db name, its differente of servers-app
MAESTRO_DATA_URI	http://localhost:5010	Data layer api
MAESTRO_REPORT_URI	http://localhost:5005	Report api
MAESTRO_AUDIT_URI	http://localhost:10900	Audit App - API URL
MAESTRO_WEBSOCKET_URI	http://localhost:8000	Webosocket App - API URL
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections
MAESTRO_WEBSOCKET_SECRET	XXX	Secret Key - JWT Websocket connections
MAESTRO_REPORT_RESULT_QTD	1500	Limit default
MAESTRO_INSERT_QTD	20	Prefetch data insert
MAESTRO_GWORKERS	2	Gworkers thread pool
CELERY_BROKER_URL	amqp://rabbitmq:5672	RabbitMQ connection

3.6 Analytics App

Installation by docker

```
analytics:
    image: maestroserver/analytics-maestro
    ports:
    - "5020:5020"
```

(continues on next page)

(continuação da página anterior)

```
    environment:
    - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
    - "MAESTRO_DATA_URI=http://data:5010"

analytics_worker:
    image: maestroserver/analytics-maestro-celery
    environment:
    - "MAESTRO_DATA_URI=http://data:5010"
    - "MAESTRO_ANALYTICS_FRONT_URI=http://analytics_front:9999"
    - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
    - "CELERYD_MAX_TASKS_PER_CHILD=2"
```

Aviso:

- MAESTRO_ANALYTICS_FRONT_URI - Enpoint API of Analytics Front - default port is 9999
- MAESTRO_DATA_URI - Enpoint API of Data App - default port is 5000

```
docker run -p 5020
    -e "MAESTRO_DATA_URI=http://localhost:5010"
    -e "CELERY_BROKER_URL=amqp://rabbitmq:5672"
    -e 'MAESTRO_MONGO_URI=localhost'
    maestroserver/analytics-maestro

docker run
    -e "MAESTRO_DATA_URI=http://localhost:5010"
    -e "MAESTRO_ANALYTICS_FRONT_URI=http://localhost:9999"
    -e "CELERY_BROKER_URL=amqp://rabbitmq:5672"
    maestroserver/analytics-maestro-celery
```

Env variables

Env Variables	Example	Description
MAESTRO_PORT	5020	Port
MAESTRO_DATA_URI	http://localhost:5010	Data Layer API URL
MAESTRO_ANALYTICS_FRONT_URI	http://localhost:9999	Analytics Front URL
MAESTRO_WEBSOCKET_URI	http://localhost:8000	Webosocket App - API URL
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections
MAESTRO_WEBSOCKET_SECRET	XXX	Secret Key - JWT Websocket connections
MAESTRO_GWORKERS	2	Gunicorn multi process
CELERY_BROKER_URL	amqp://rabbitmq:5672	RabbitMQ connection
CELERYD_TASK_TIME_LIMIT	10	Timeout workers

3.7 Analytics Front

Installation by docker

```
reports:
    image: maestroserver/analytics-front-maestro
```

(continues on next page)

(continuação da página anterior)

```
    ports:
    - "9999:9999"
    environment:
    - "MAESTRO_MONGO_URI=mongodb"
    - "MAESTRO_MONGO_DATABASE=maestro-client"
```

Aviso:

- MAESTRO_REPORT_URI - Enpoint API of Reports - default port is 5005
- MAESTRO_DATA_URI - Enpoint API of Data App - default port is 5000

```
docker run -p 5005
    -e "MAESTRO_MONGO_URI=mongodb"
    -e "MAESTRO_MONGO_DATABASE=maestro-client"
    maestroserver/analytics-front-maestro
```

Env variables

Env Variables	Example	Description
MAESTRO_PORT	9999	
API_URL	http://localhost:8888	Server app Url
NODE_ENV	development|production	
MAESTRO_MONGO_URI	localhost	DB string connection
MAESTRO_MONGO_DATABASE	maestro-client	Database name
MAESTRO_SECRETJWT	XXXX	Secret key - server app
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections
MAESTRO_SECRETJWT_PUBLIC	XXXX	Secret key - same server app
AWS_ACCESS_KEY_ID	XXXX	
AWS_SECRET_ACCESS_KEY	XXXX	
AWS_DEFAULT_REGION	us-east-1	
AWS_S3_BUCKET_NAME	maestroserver	
MAESTRO_UPLOAD_TYPE	S3/Local	Upload mode
LOCAL_DIR	/public/static/	Where files will be uploaded
PWD	$rootDirectory	PWD process

3.8 Data App

Installation by docker

```
data:
    image: maestroserver/data-maestro
    ports:
    - "5010:5010"
    environment:
        - "MAESTRO_MONGO_URI=mongodb"
        - "MAESTRO_MONGO_DATABASE=maestro-client"
```

```
docker run -p 5010 -e "MAESTRO_MONGO_URI=mongodb" -e "MAESTRO_MONGO_DATABASE=maestro-
↪client" maestroserver/data-maestro
```

Env variables

Env Variables	Example	Description
MAESTRO_PORT	5010	Port used
MAESTRO_MONGO_URI	localhost	Mongo Url conn
MAESTRO_MONGO_DATABASE	maestro-client	Db name, its differente of servers-app
MAESTRO_GWORKERS	2	Gunicorn multi process
MAESTRO_INSERT_QTD	200	Throughput insert in reports collection
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections

3.9 Scheduler App

Installation by docker

```
scheduler:
    image: maestroserver/scheduler-maestro
    environment:
    - "MAESTRO_DATA_URI=http://data:5010"
    - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
    - "MAESTRO_MONGO_URI=mongodb"
    - "MAESTRO_MONGO_DATABASE=maestro-client"

scheduler_worker:
    image: maestroserver/scheduler-maestro-celery
    environment:
    - "MAESTRO_DATA_URI=http://data:5010"
    - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
    - "MAESTRO_DISCOVERY_URI=http://discovery:5000"
    - "MAESTRO_ANALYTICS_URI=http://analytics:5020"
    - "MAESTRO_REPORT_URI=http://reports:5005"
```

```
docker run
    -e "MAESTRO_DATA_URI=http://localhost:5010"
    -e "CELERY_BROKER_URL=amqp://rabbitmq:5672"
    maestroserver/scheduler-maestro

docker run
    -e "MAESTRO_DATA_URI=http://localhost:5010"
    -e "MAESTRO_DISCOVERY_URI=http://localhost:5000"
    -e "MAESTRO_ANALYTICS_URI=http://localhost:5020"
    -e "MAESTRO_REPORT_URI=http://localhost:5005"
    -e "CELERY_BROKER_URL=amqp://rabbitmq:5672"
    maestroserver/scheduler-maestro-celery
```

Aviso:

- MAESTRO_DATA_URI - Enpoint API of Data App - default port is 5000

Env variables

Env Variables	Example	Description
MAESTRO_DATA_URI	http://localhost:5010	Data Layer API URL
MAESTRO_DISCOVERY_URI	http://localhost:5000	Discovery App URL
MAESTRO_ANALYTICS_URI	http://localhost:5020	Analytics App URL
MAESTRO_REPORT_URI	http://localhost:5005	Reports App URL
MAESTRO_MONGO_URI	localhost	MongoDB URI
MAESTRO_MONGO_DATABASE	maestro-client	Mongo Database name
CELERY_BROKER_URL	amqp://rabbitmq:5672	RabbitMQ connection
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections

3.10 Audit App

Installation by docker

```
audit:
    image: maestroserver/audit-app-maestro
    ports:
    - "10900:10900"
    environment:
    - "MAESTRO_MONGO_URI=mongodb"
    - "MAESTRO_MONGO_DATABASE=maestro-audit"
    - "MAESTRO_DATA_URI=http://data:5010"
```

Aviso:

- MAESTRO_DATA_URI - Enpoint API of Data App - default port is 5000

```
docker run -p 10900
    -e "MAESTRO_MONGO_URI=mongodb"
    -e "MAESTRO_MONGO_DATABASE=maestro-audit"
    maestroserver/audit-app-maestro
```

Env variables

Env Variables	Example	Description
MAESTRO_PORT	10900	
NODE_ENV	development|production	
MAESTRO_MONGO_URI	localhost	DB string connection
MAESTRO_MONGO_DATABASE	maestro-audit	Database name
MAESTRO_TIMEOUT	1000	Timeout any http private request
MAESTRO_DATA_URI	http://localhost:5010	Data App - API URL
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections

3.11 WebSocket App

Installation by docker

```
data:
    image: maestroserver/websocket-maestro
    ports:
    - "8000:8000"
```

```
docker run -p 8000:800 maestroserver/websocket-maestro
```

Env variables

Env Variables	Example	Description
MAESTRO_WEBSOCKET_SECRET	backSecretToken	Token to authenticate backends apps
MAESTRO_SECRETJWT	frontSecretToken	Token to autheticate front end users
CENTRIFUGO_ADMIN	adminPassword	Admin password
CENTRIFUGO_ADMIN_SECRET	adminSecretToken	Token to autheticate administrator users

3.12 HA - Production Read

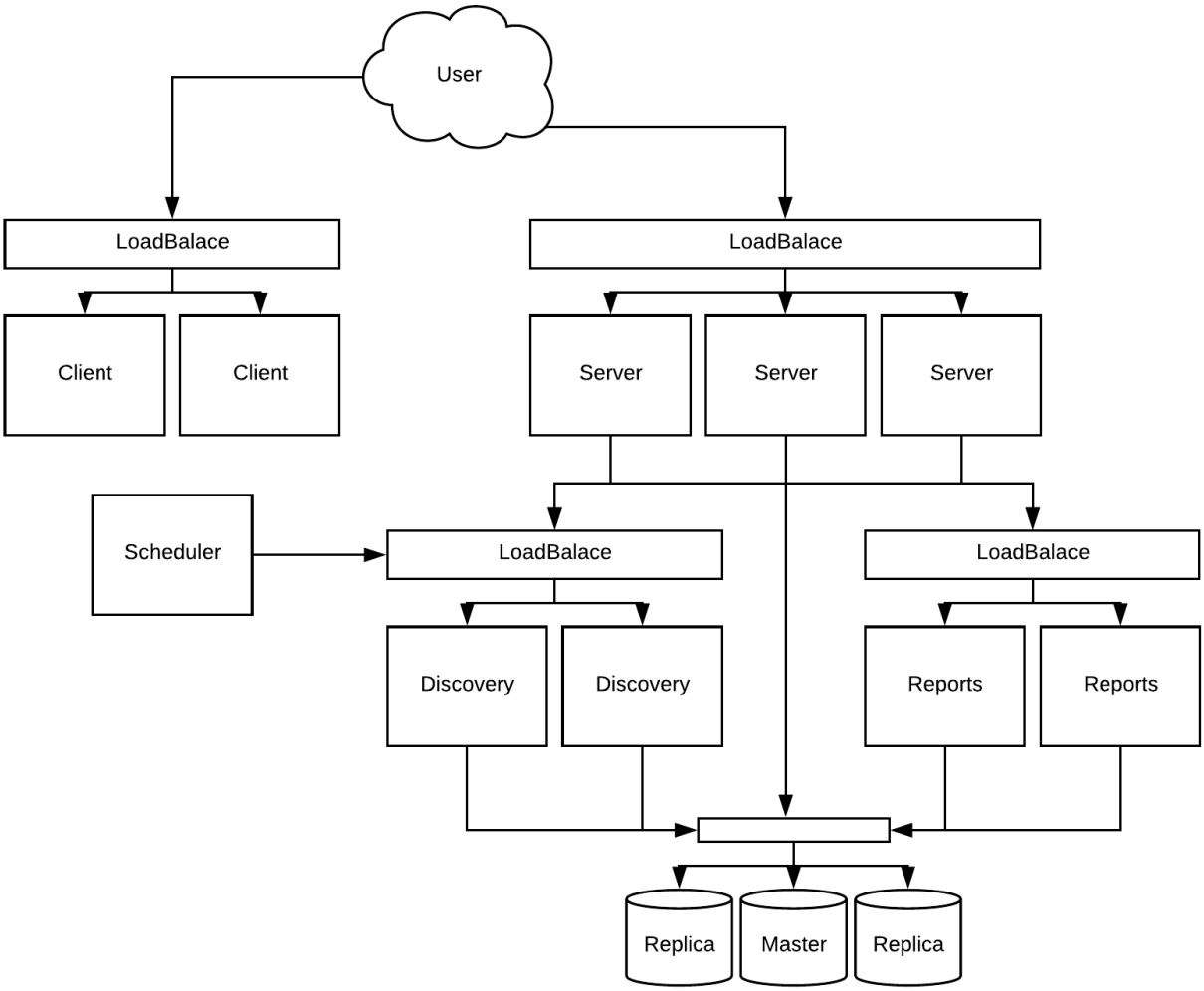
Not exist one way to do right, but we have some concerns, needed a backup system, HA avaliability, monitoring and etc, like a normal app system. Maestro built with 12 factory in mind, its able to hight availability and horizontal scaling.

3.12.1 12 Factory and Horizontal Scaling

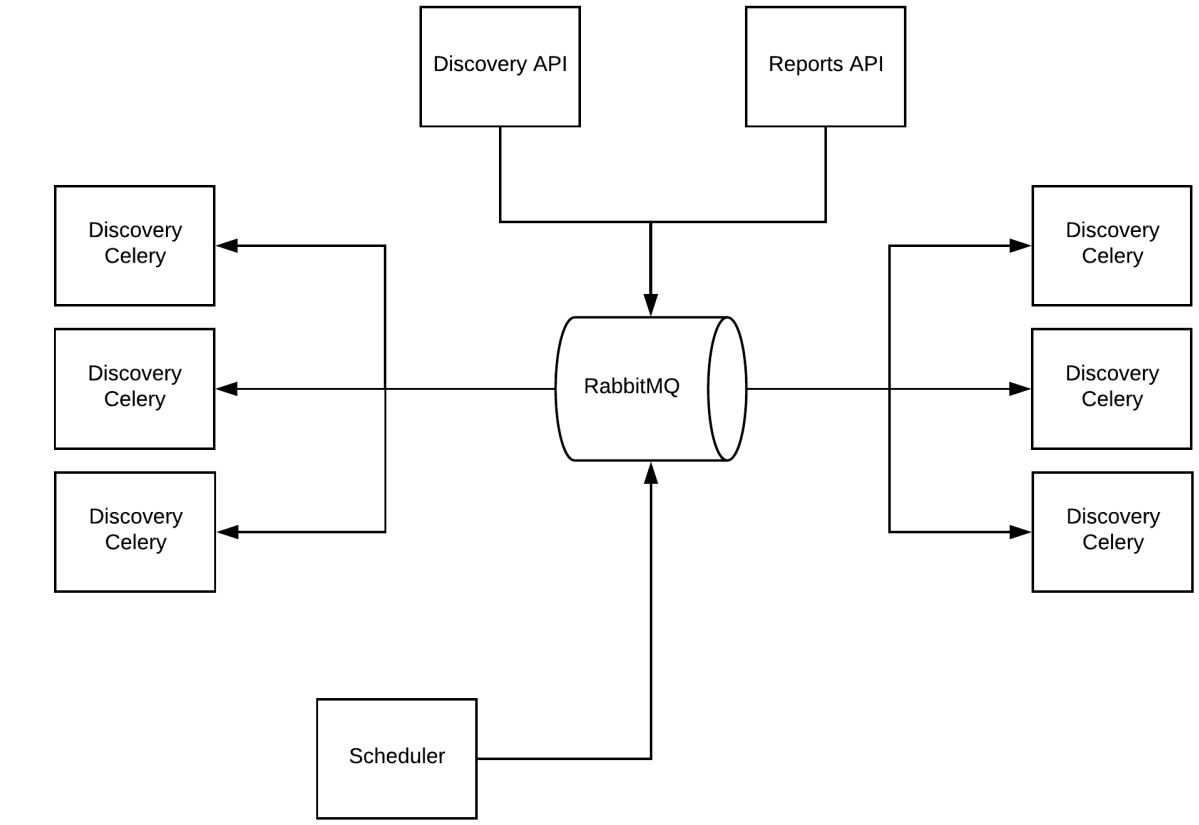
All services need work in stateless, like session, images or any process, we use JWT for authentication and isolated storage server.

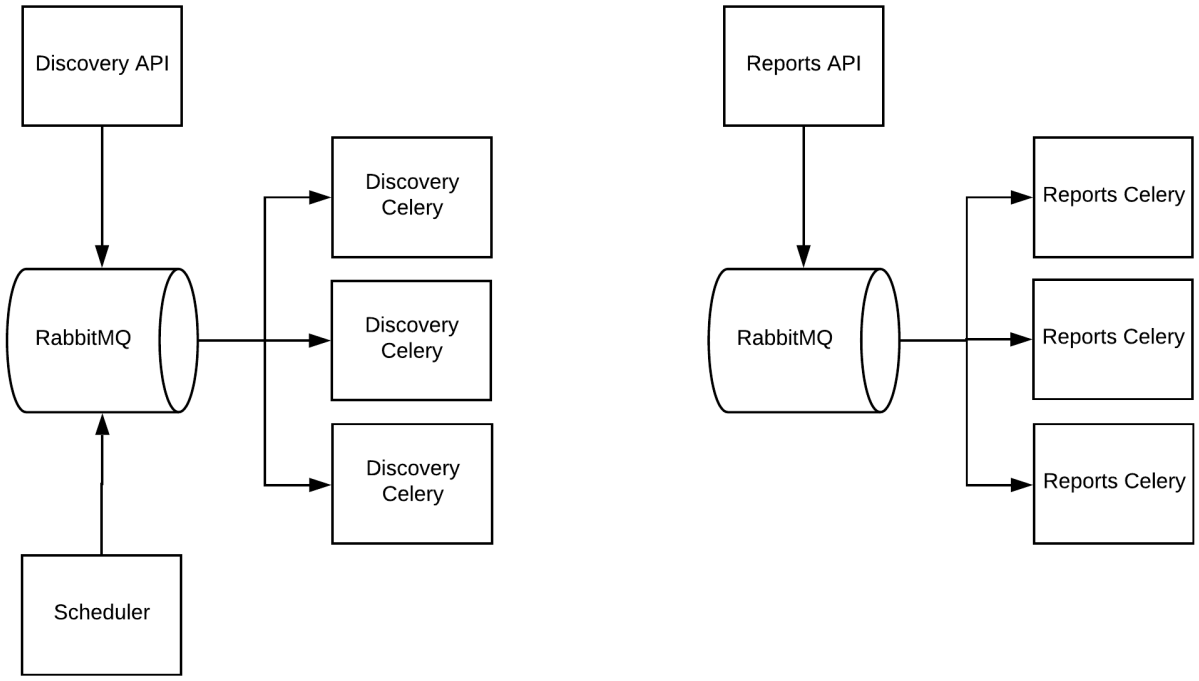
- Avoid to use any local configurate like local upload, maestro has supported to use s3 storage object.
- Put all connection config in env docker setup, would able to use kubernetes, rancher or any orchestration is good advice.
- Its possible to deploy discovery api in one server, and discovery celery another server.
- In front end, use nginx, or any other proxy.

One example setup, can be in each node,



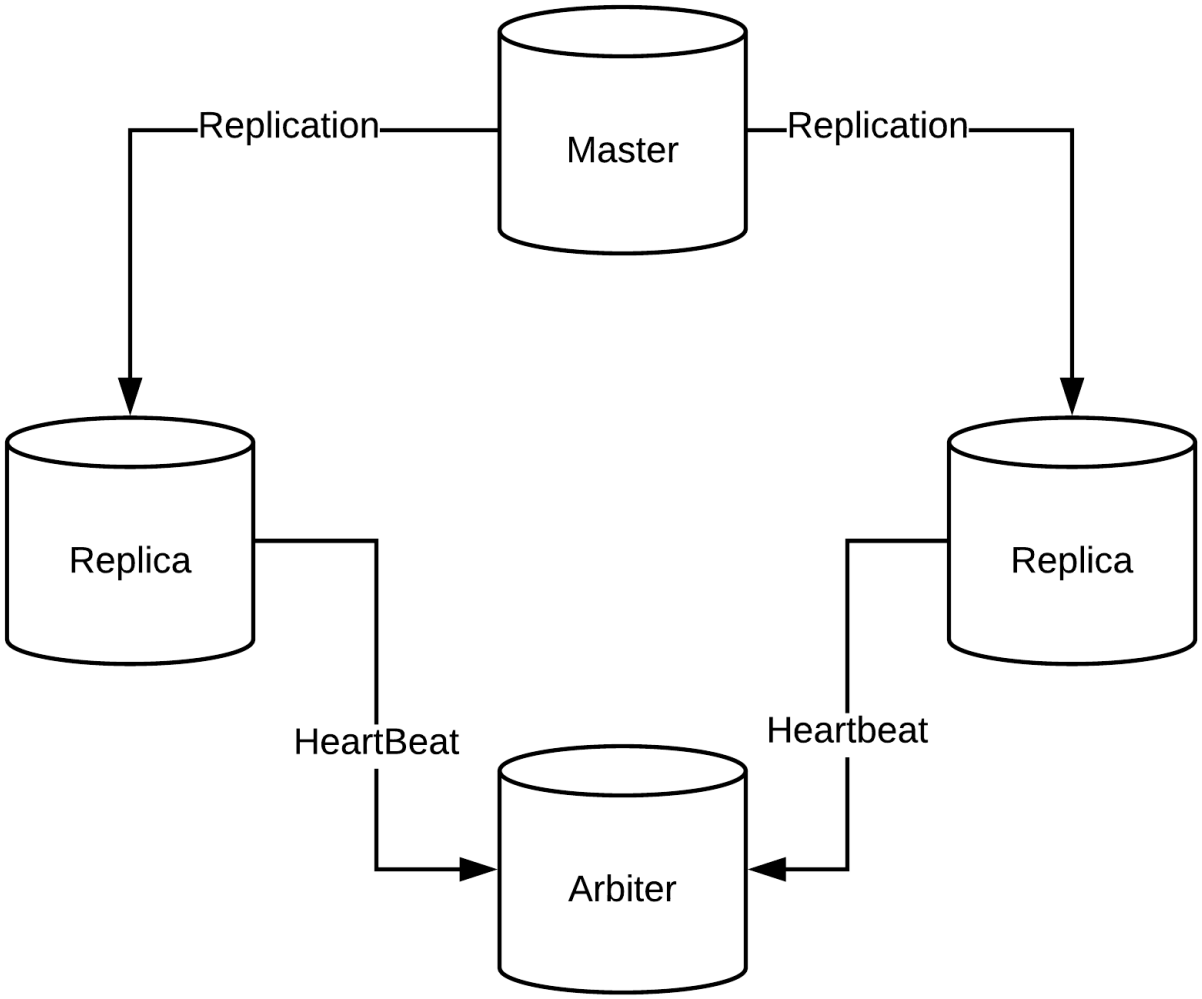
It's possible to improve discovery and reports app


Discovery API
Reports API
Discovery Celery
Discovery Celery
Discovery Celery
RabbitMQ
Discovery Celery
Discovery Celery
Discovery Celery
Scheduler


Discovery API
Discovery Celery
RabbitMQ
Discovery Celery
Discovery Celery
Scheduler
Reports API
Reports Celery
RabbitMQ
Reports Celery
Reports Celery

3.12.2 MongoDB

MongoDB is prepared to scale and high availability, the best setup creates a master and replica,



3.12.3 Scheduler Beat App

Perigo: Scheduler app have two parts, the producer called beat and workers, the beat its only service without prepare to setup in high availability, be carefull. It´s hard to put a beat service in HA system in a simple way, I prefer to go in simple way, to minimize, beat schedule is isolated and build in an immutable state (if fall, you call up in another server, and all schedules will be recovered), but must have only one beat instance per time.

3.12.4 Version

We controlled a version (semversion), any docker has this version, for example, maestro-server:0.3, if you need to rollback is easier to do. How find my version:

- Front end, show in right-footer.

- http://{server-api}:8888/
- http://{discovery-api}:5000/
- http://{reports-api}:5005/
- http://{analytics-maestro}:5020/
- http://{analytics-front}:9999/
- http://{audit}:10900/

3.13 Advanced configs

3.13.1 SMTP Config

To set up smtp, you need to declare some envrioment inside server-app

SMTP_PORT	465	
SMTP_HOST	smtp.gmail.com	
SMTP_SENDER	"maestrosmtp@gmail.com"	
SMTP_USERNAME	"maestrosmtp"	
SMTP_PASSWORD	"XXXX"	
SMTP_USETSL	truelfalse	Enable TLS connect
SMTP_IGNORE	truelfalse	During the connection, validate security connection?

Example

```
version: '2'

services:
server:
    image: maestroserver/server-maestro
    ports:
    - "8888:8888"
    environment:
    - SMTP_PORT=465
    - SMTP_HOST=smtp.gmail.com
    - SMTP_SENDER='mysender@gmail.com'
    - SMTP_USERNAME=myusername
    - SMTP_PASSWORD=mysecret
    - SMTP_USETSL=true
```

Services

- Server App

3.13.2 Upload Config

You can choose two mode to upload the files, a local file or using S3 to storage direcly.

Where need to configure an upload file manage:

server-app	Using in avatar users, teams and projects images.
discovery app	Using to store report artifact, like pdfs, csv and jsons.

Local

For a single node, the file will be stored in local disk.

Env variables

UPLOAD_TYPE	Local
LOCAL_DIR	/upload

```
services:
server:
    image: maestroserver/server-maestro
    environment:
    - UPLOAD_TYPE=Local
    - LOCAL_DIR=/upload

client:
    image: maestroserver/client-maestro
    environment:
    - STATIC_URL='http://server-app:8888/static'
```

AWS S3

You can use S3 Amazon storage object service, perfectly for HA environments

Env variables

UPLOAD_TYPE	S3
AWS_ACCESS_KEY_ID	XXXXXXXXXX
AWS_SECRET_ACCESS_KEY	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
AWS_DEFAULT_REGION	us-east-1
AWS_S3_BUCKET_NAME	maestroserver

```
services:
server:
    image: maestroserver/server-maestro
    environment:
    - AWS_ACCESS_KEY_ID='XXXXXXXXXX'
    - AWS_SECRET_ACCESS_KEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
    - AWS_DEFAULT_REGION='us-east-1'
    - AWS_S3_BUCKET_NAME='maestroserver'


client:
    image: maestroserver/client-maestro
    environment:
    - STATIC_URL='http://maestroserver.s3.aws.com.br'
```

Nota: Need to be adjusted client-app appoint new local file

Services

- Server App
- Analytics Front

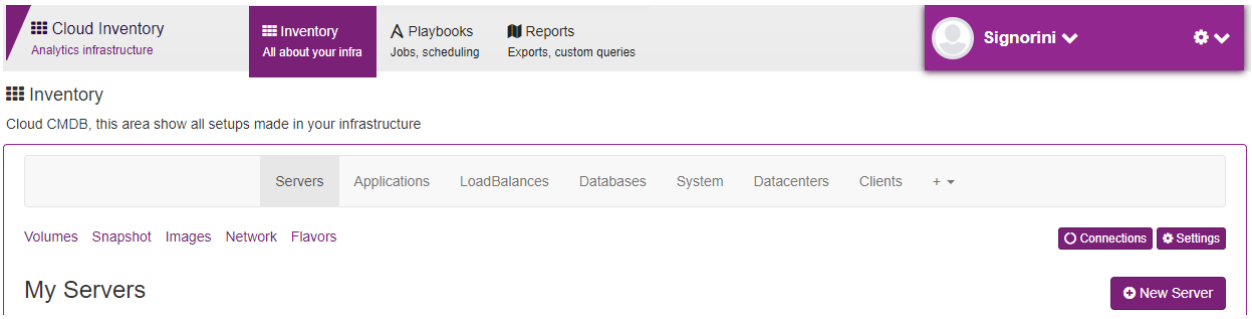
3.13.3 Themes

You can change the maestro theme, its a variable environment into client app

```
client:
    image: maestroserver/client-maestro
    ports:
    - "80:80"
    environment:
    - "API_URL=http://localhost:8888"
    - "THEME=gold"
```

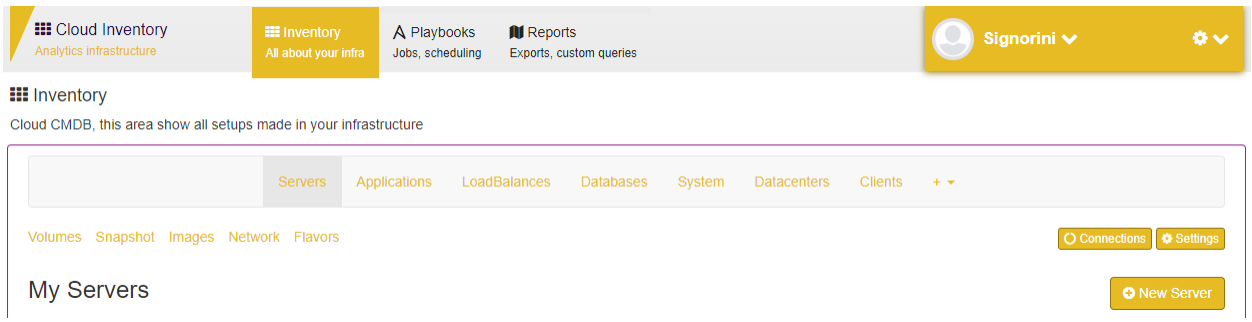
There are fill options to choose.

Default



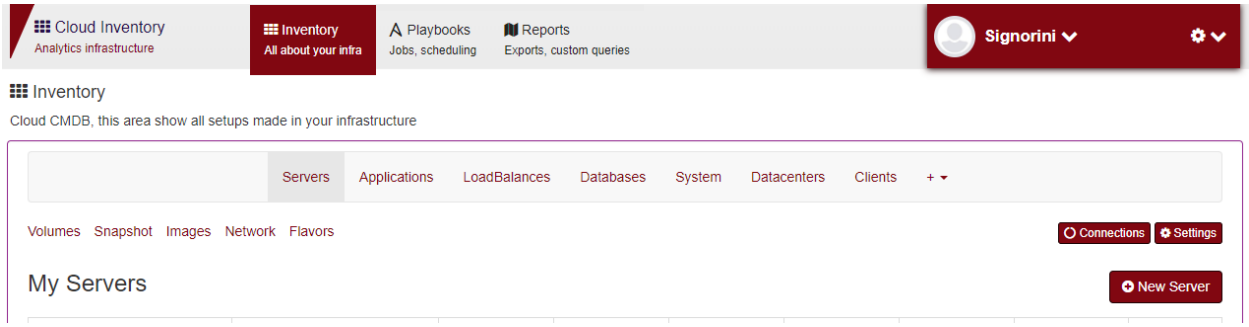
THEME=lotus

Gold



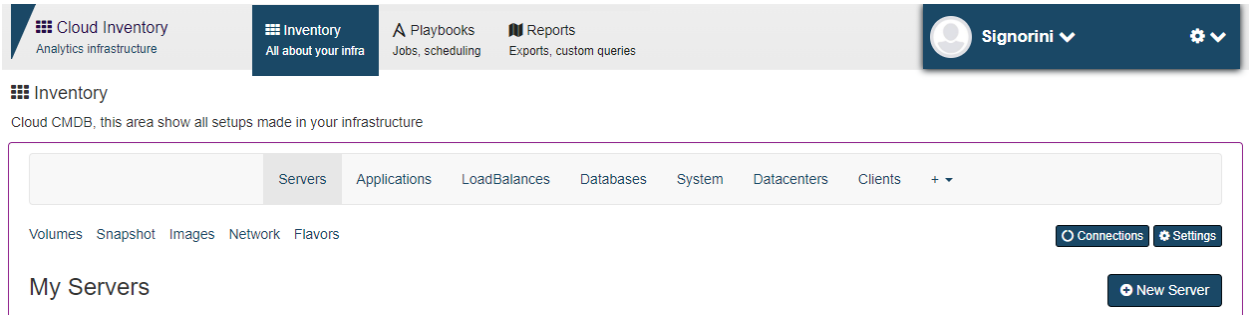
THEME=gold

Wine



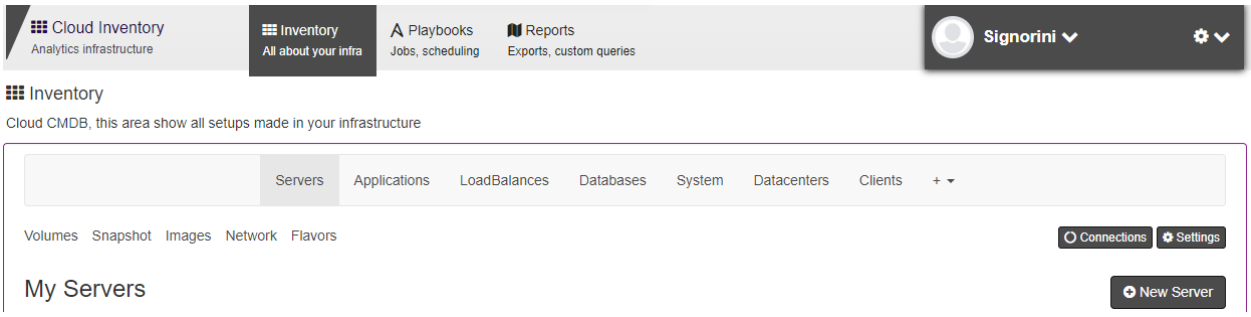
THEME=wine

Blue



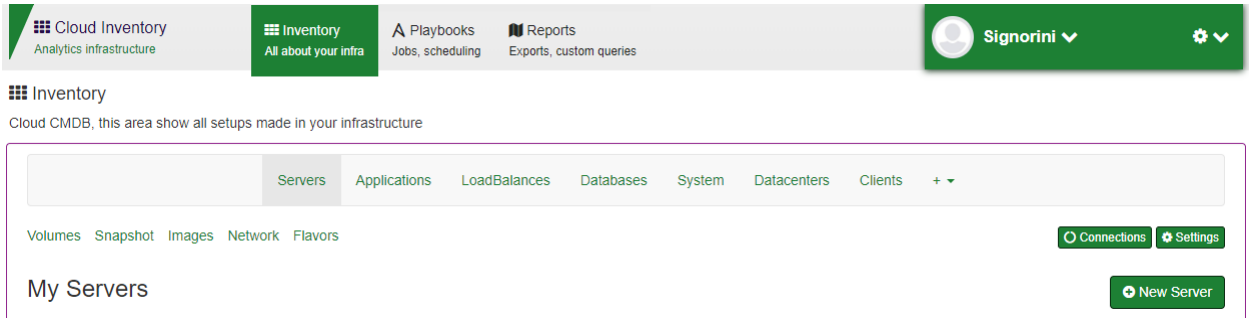
THEME=blue

Dark



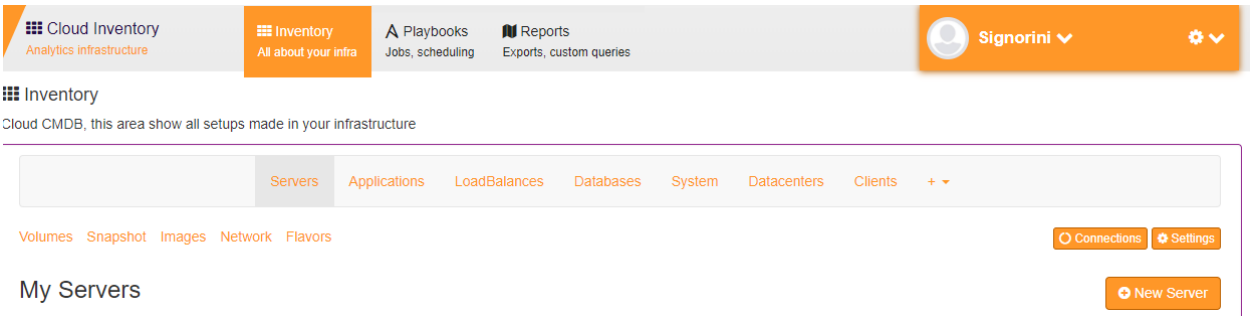
THEME=dark

Green



THEME=green

Orange



THEME=orange

CAPÍTULO 4

User Guide

The inventory is divided into three parts:

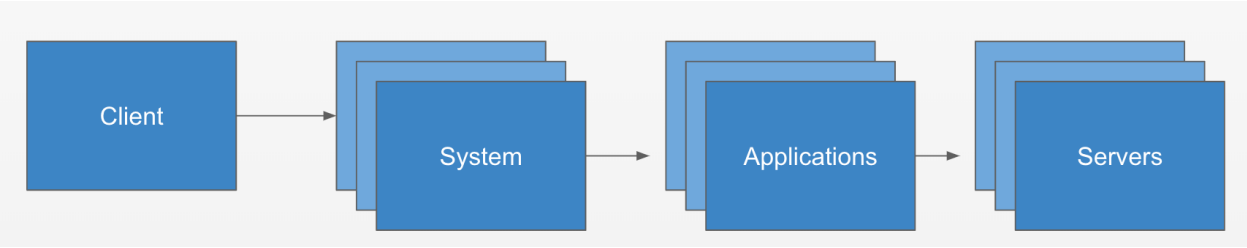
> **Cloud Inventory:** All about your infra, its an area with all information of register and visualization of cloud envriroment, like servers, applications, loadbalances, databases, datacenters, system, and clients.

> **Analytics:** Elegant graphic of dependencies. Through elegant view, it's easy to understand the architecture system. You can export to svg, png or embed the graph on any service like Confluence, GitHub,

> **Reports:** Exclusive to generate reports, with a possibility to apply advanced filters and export in different types of file.

4.1 Maestro Cloud Inventory

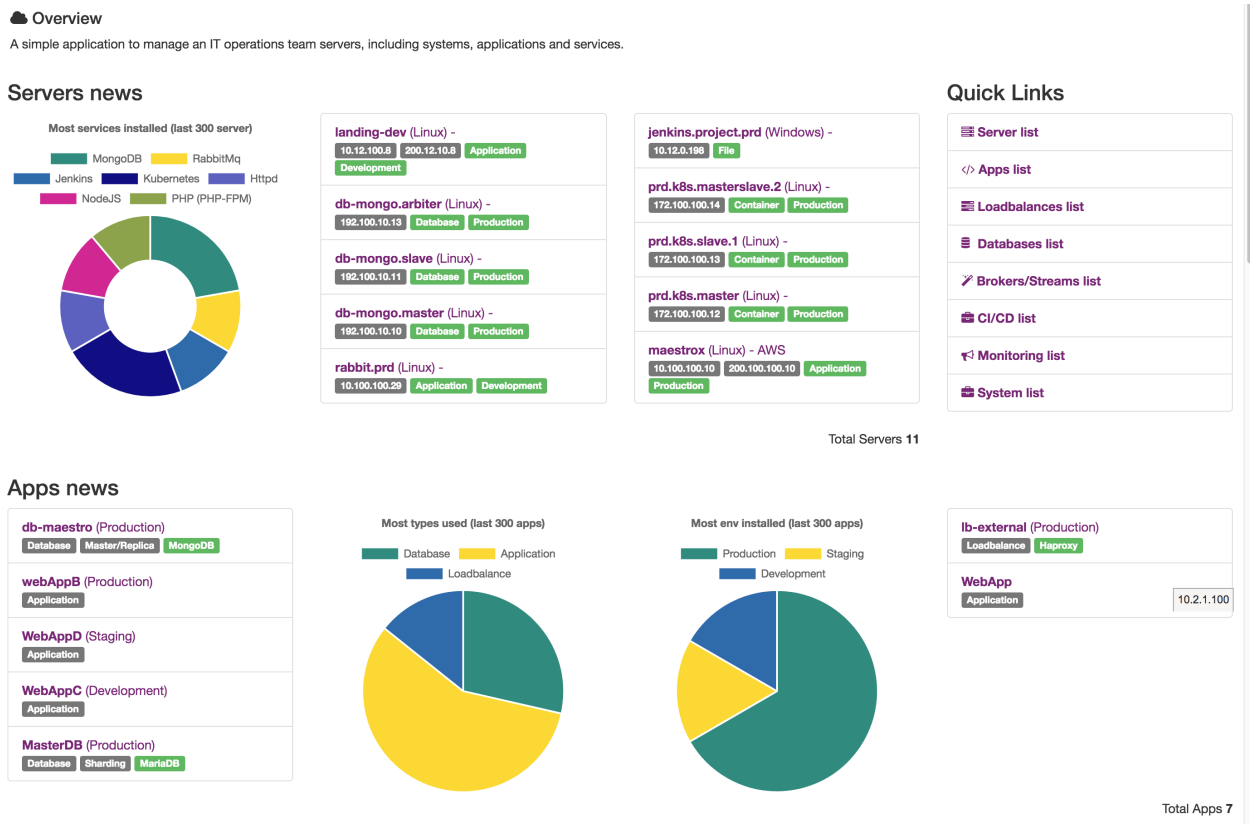
- Help IT operation teams to organize and audit multicloud infrastructure
- Substitute CMDB systems
- Enable continuous discovery of servers, services and apps
- Classify each service as a database, message queue, VPN, API gateway, service mesh, etc
- Create a relation between servers, services and clients.



4.2 Cloud Inventory

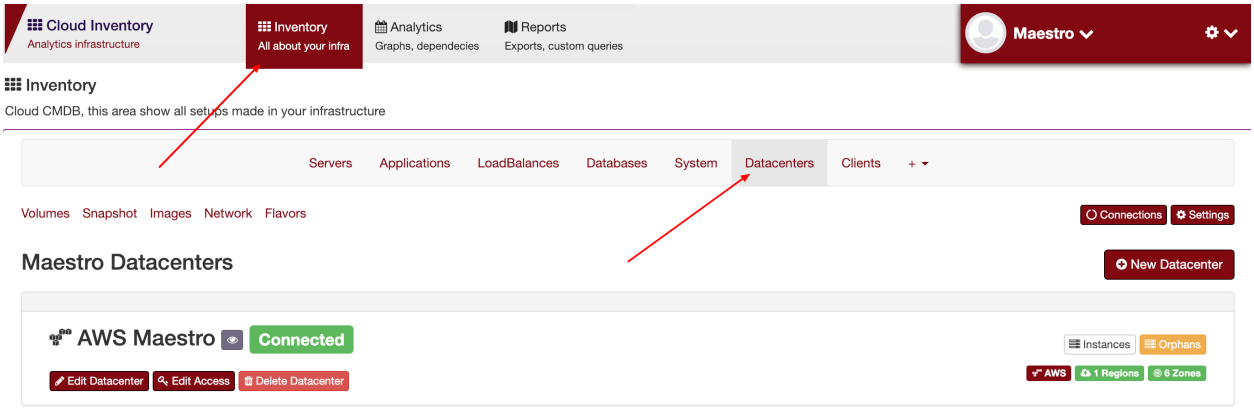
4.2.1 Inventory

Inside inventory there are a groups of entities organize by responsabilitie accordly architecture point of view.



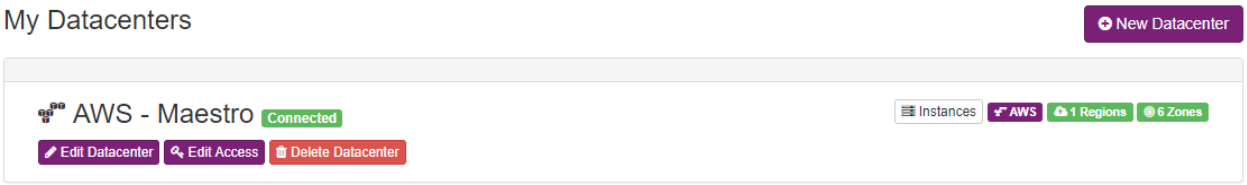
Datacenters

Inventory > Datacenter

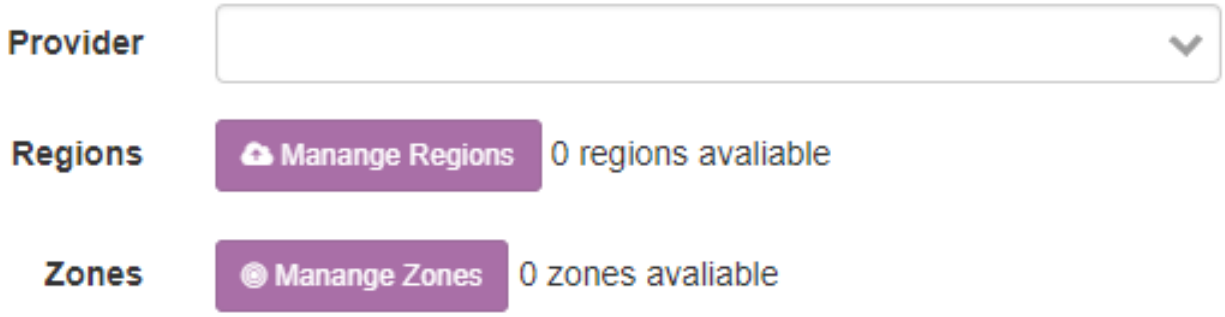


Register all clouds, bare metal, providers and etc.

Field	Functional
Name	Identity name
Provider	Choose a provider, or create a new one
Regions	Choose or create regions
Zones	Choose or create zones



List of datacenters, with instances, regiions and zones



You can choose the provider, regions and zones.

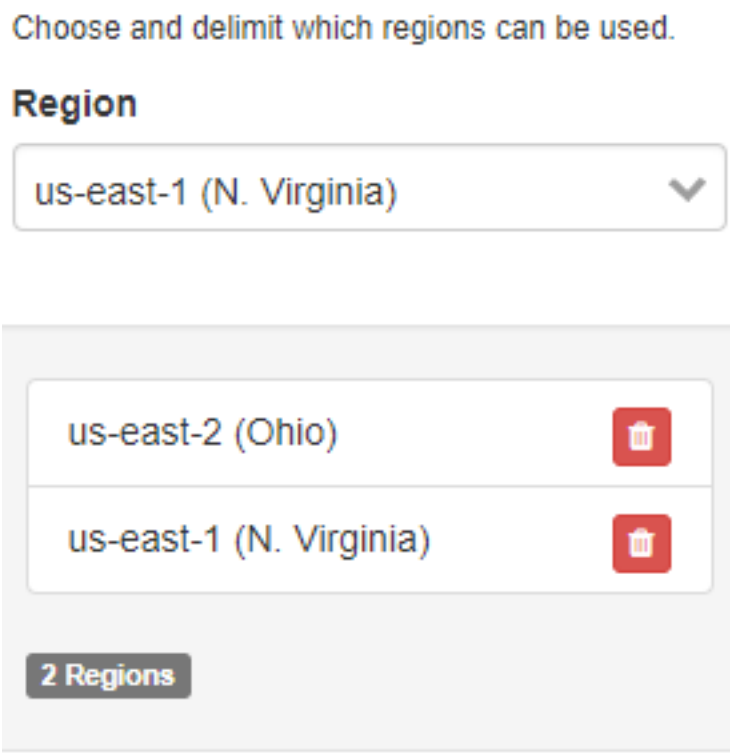
Choose regions or create it.

Servers

Inventory > Server

The most import fields in server register is:

Field	Functional
Hostname	Hostname, accept duplicate hostname per team, but the inveotry will warning about this.
Ipv4 Private	Ipv4 private, same situation of hostname (accept duplicate but will warning)
Ipv4 Public	Ipv4 public, only for external servers.
OS	Base is most basic form, like Linux adn Windows, Distro normally use for linux, like ubuntu, centos, and version its a number
CPU	CPU
Memory	Memory
Environment	Production | Development | Stage

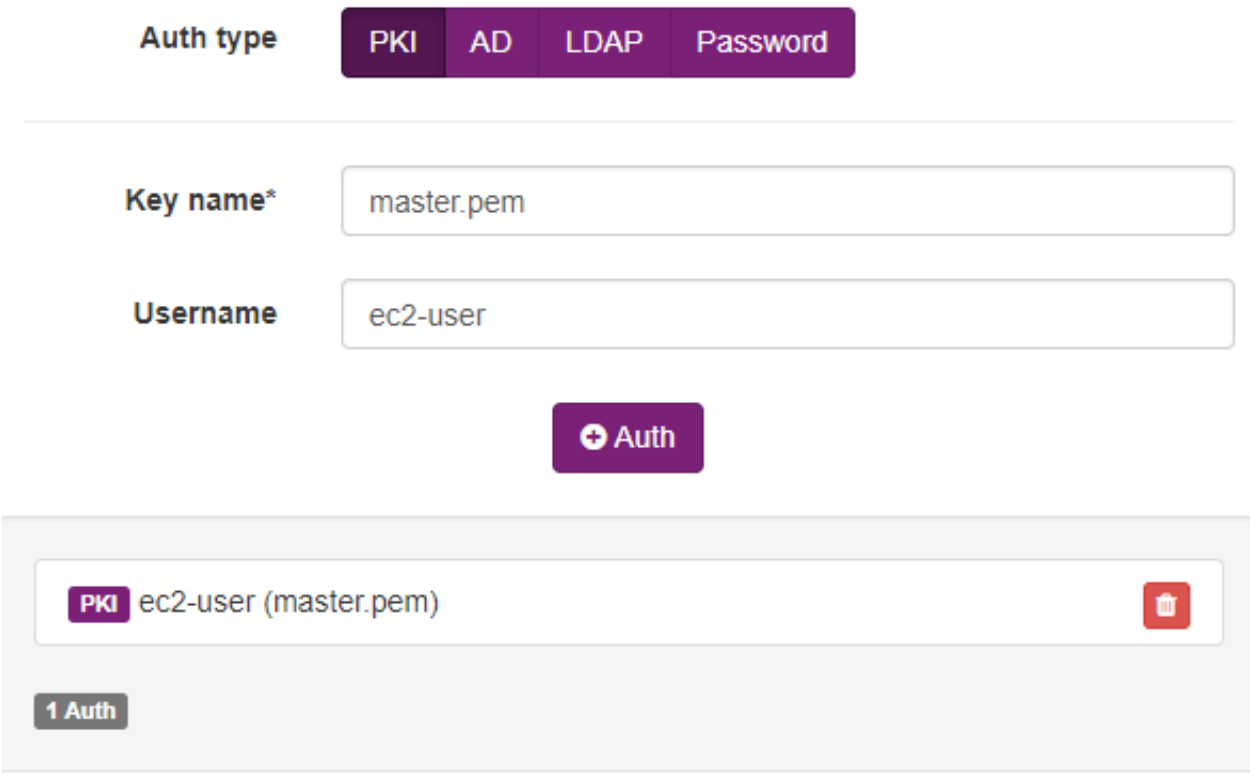


Setup OS server

You have some tabs,

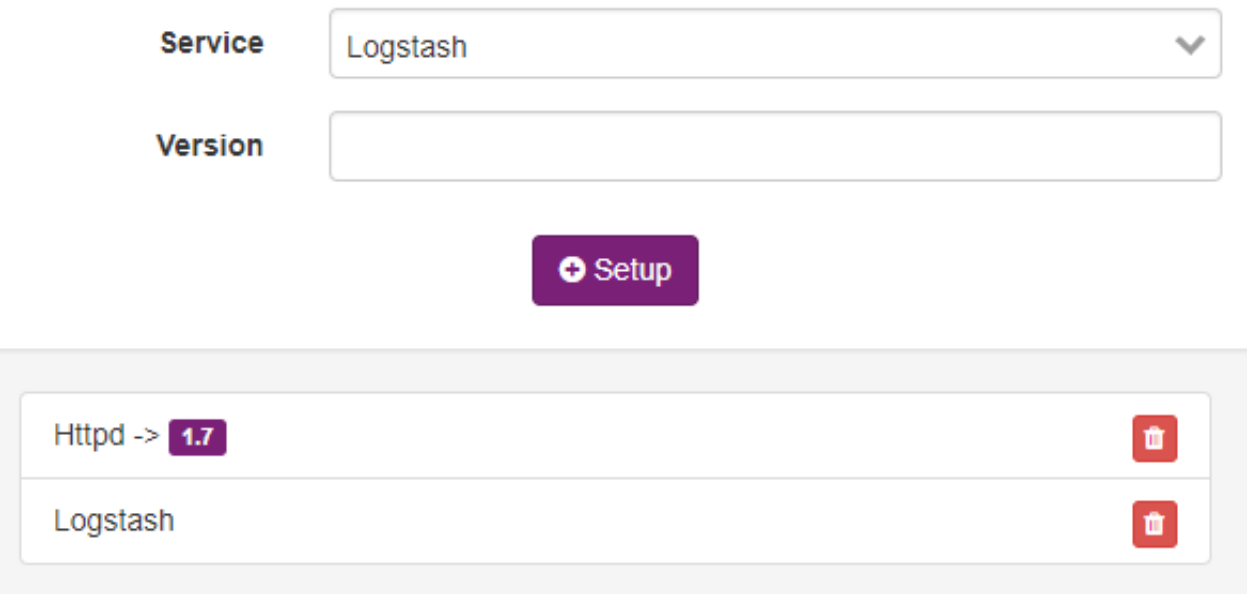
Field	Functional
Storage	Add your storage information, mount path, size in GB and if is a root device, some cloud maybe bring news informations.
Datacenter	Provider, region and zones, if this server still in cloud Environment you can put id in id_instance, will be create a link and Maestro will know if its duplicate when execute a auto discovery servers.
Auth	Used to register a methods to authenticate in servers.
Service	Register all services its run inside a servers, this information its used to create some links with applications inventory and used in `Application Manager` system.

Assing a dc name, region and zone.



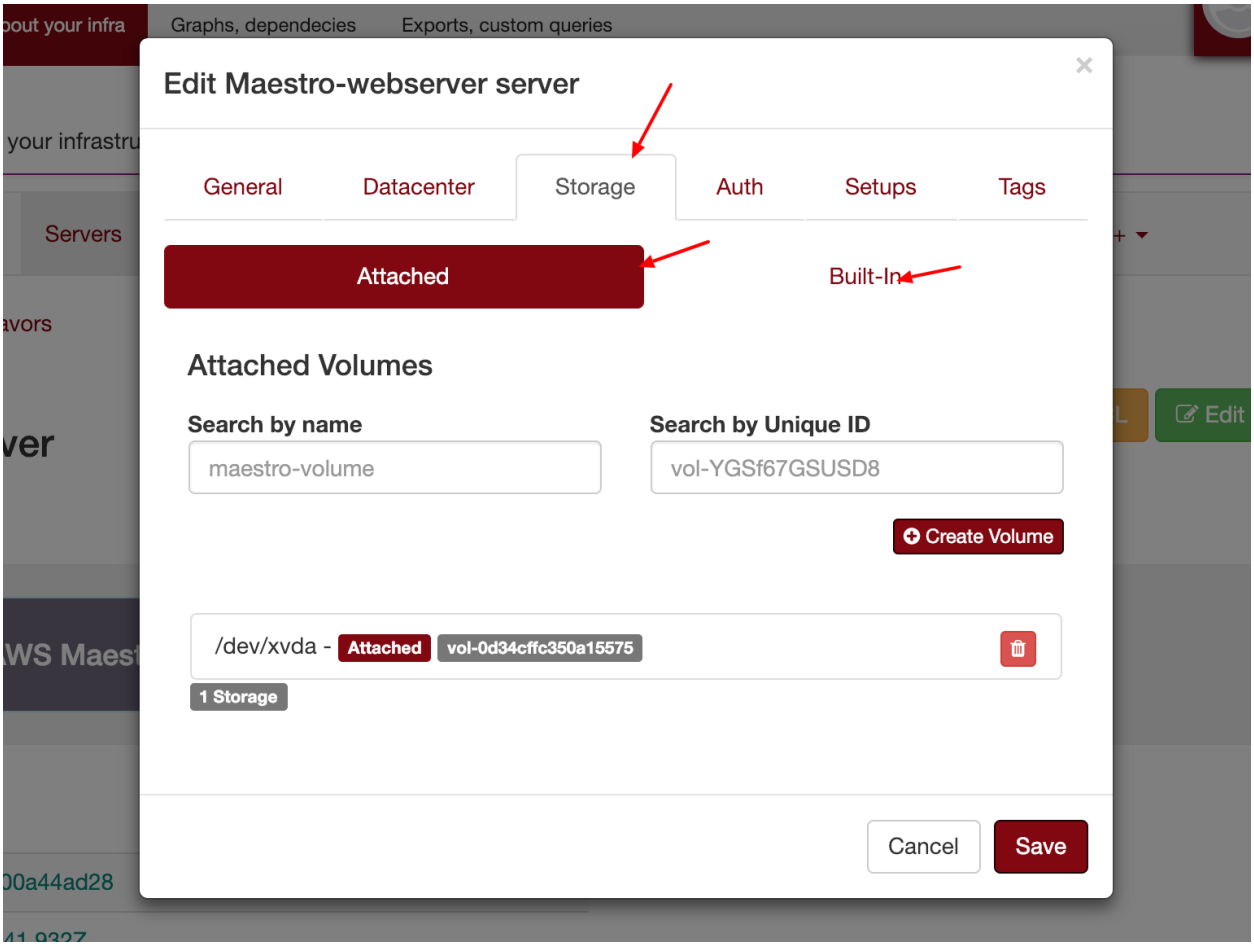
Register, which mode you can to access and authenticate on server.

Nota: Services is a very important field in servers, which this information the system will try to find some relation with applications, for example if you register Oracle Database, and create a database and register a relation between this servers, the system will automatically create a service relation.



Related services running.

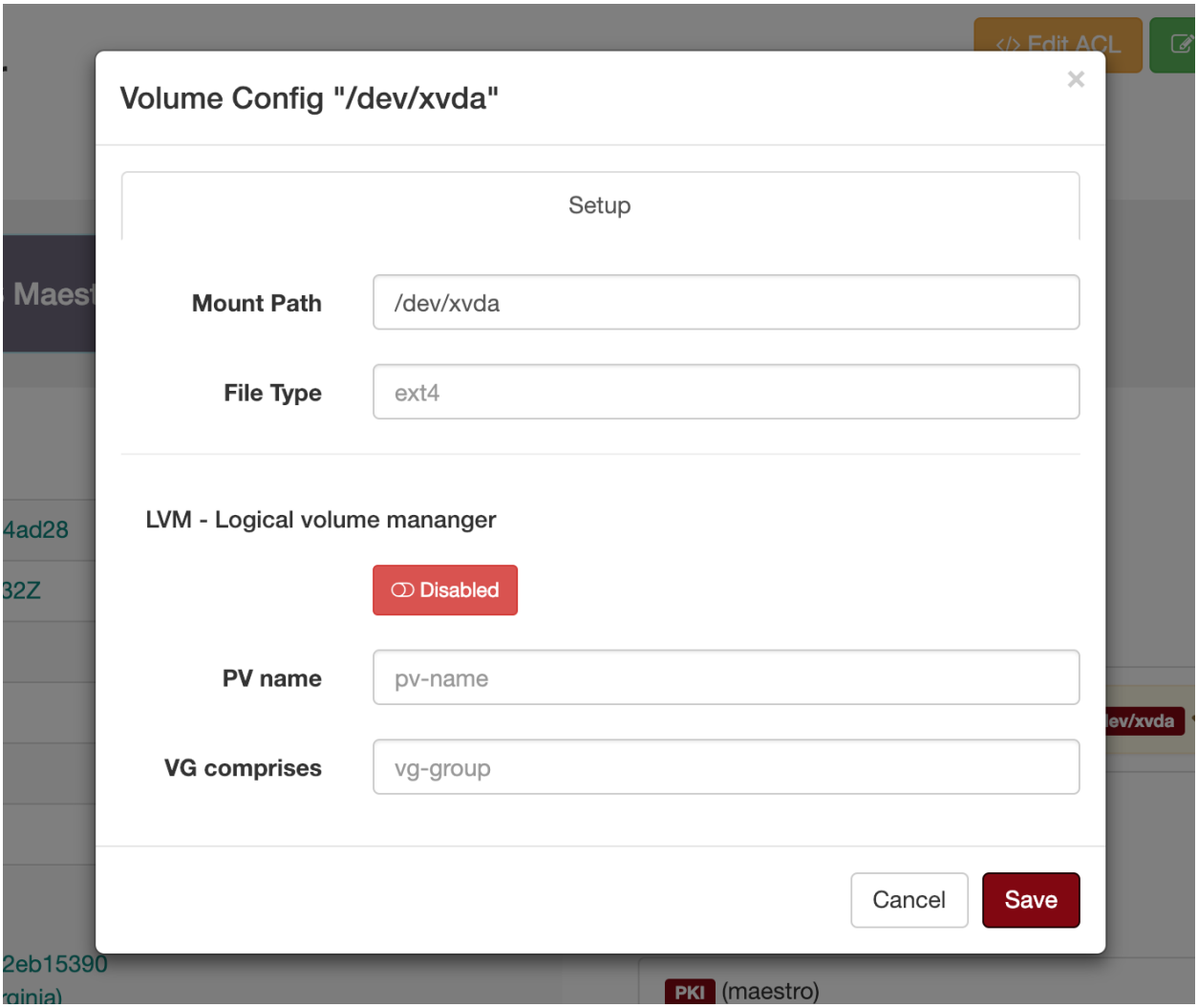
Volumes



Can be attached or built-in:

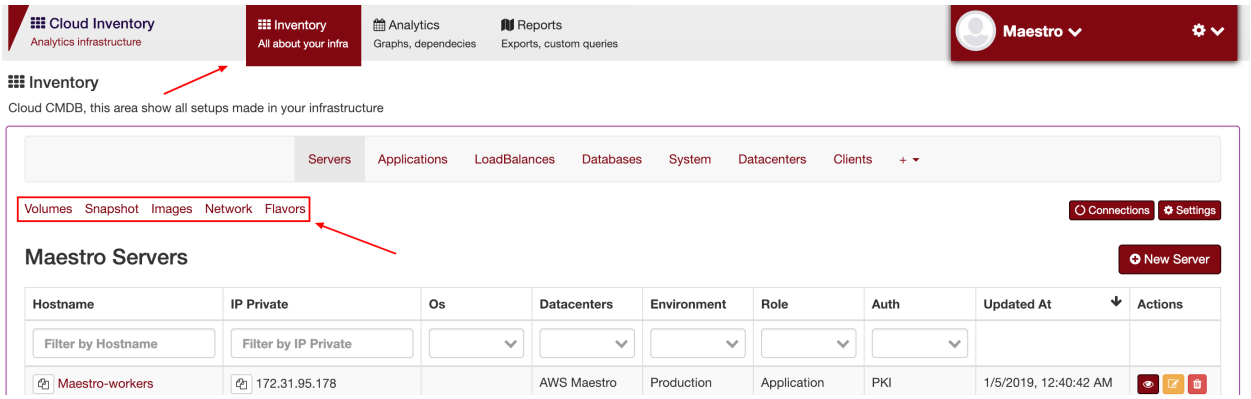
- **Attached** are network storage or distributed storage service (ex: NFS)
- **built-in** its hard drive directly on a server (usually premise datacenters).

After created you can describe a mount path, file type, size and virtual volumes configuration (if exist).



Server Resources

Datacenters and servers have some assumption resources, and there are volumes, flavors, images, and networks.



- **Volumes**: List of volumes (ex: EBS, HardDisk)
- **Flavors**: Datacenters information, can be public or private.
- **Images**: Images hosted on datacenter, used for virtualization services.
- **Network**: depends on providers architecture, for example aws have security groups, acls, vpcs and subnets and etc.

Apps

Inventory > Application

Apps are it all services with is a business responsibility, normally it is an application made by the developer and deploys.

Some fields:

Field	Functional
Name	Hostname, accept duplicate hostname per team, but the inveotry will warning about this.
Environment	Production | Development | Stage
Language	
Cluster mode	

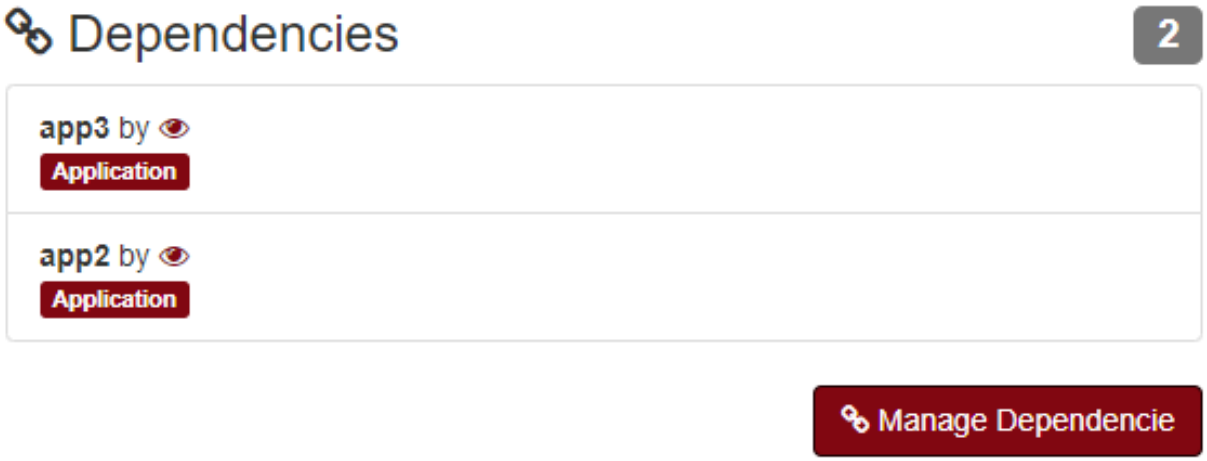
Specification

Field	Functional
Role	Point information like endpoint, commands, health check, its good for doc.
System	Systems related it.
Server	Servers deployed by app.
Deploy	List of ways to deploy this app.

Choose language like node or php.

Add dependency

Nota: Dependency is a relation between two or more applications, example its database its a dependency of app4, and app4 its dependency of loadbalance.

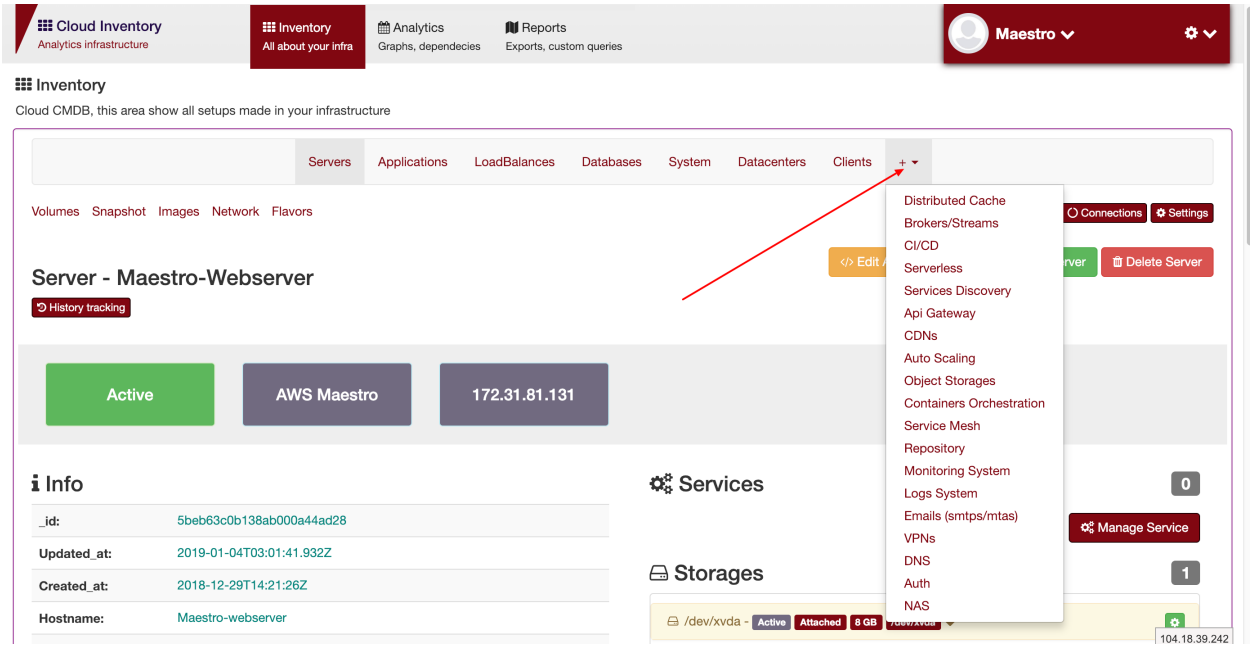


Add outers applications with dependency.

Resources

Inventory > ${Resource}

Resource usually its is no functional application, brokers, logs, dns are an example of these resources.

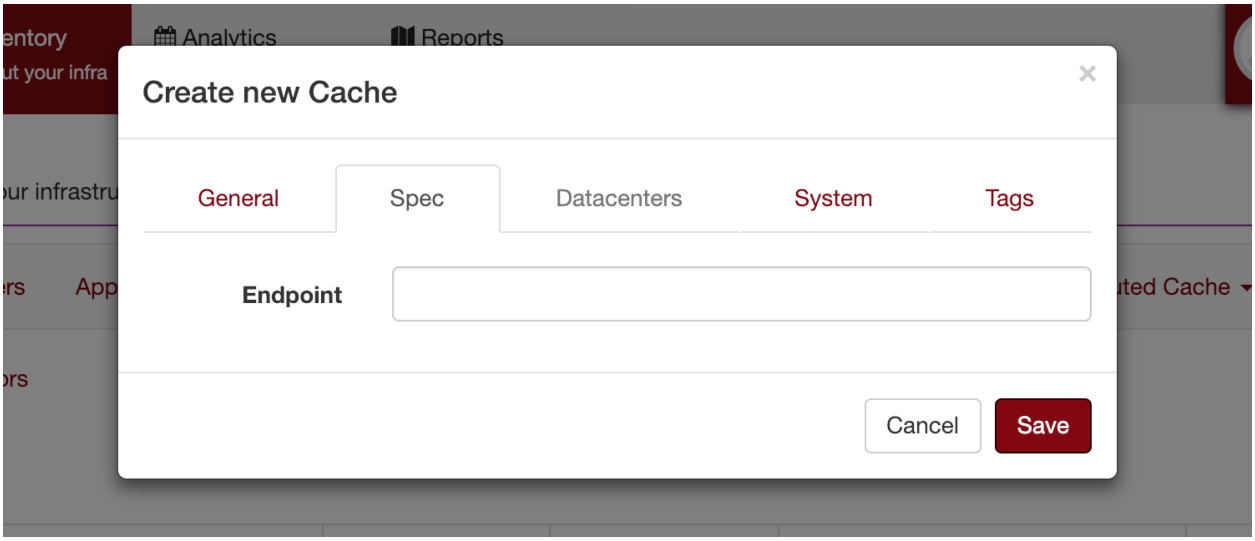


Resources types:

Family	Description
Distributed cache	Cache system, example is Redis, Memcache
Brokers/Streams	Message system or streams system, can be RabbitMQ, SQS, Kafka and Spark Streams
CI/CD	Ci Tools, like Jenkins, Atlassian Stack, AWS Pipeline
Serverless	Function list, AWS lambdas, step functions, google function or Kuberless
Services Discovery	We have consul, etcD, hystrix and etc.
Api Gateway	Api Gateway service, like Kong, AWS api gateway or tunned nginx.
CDNs	CDNs services, cloudflare, akamai, cloud front and etc.
Auto Scaling	Autoscaling setup
Objects Storages	Objects storages, S3, GlusterFS, Ceph, DO Storages.
Containers Orchestration	Main pieces of orchestration tools, like master of kubernetes, eks configs, docker swarm, mesos and etc
Service Mesh	Like Linkerd, IstIO, Consul or AWS x-ray
Repository	Nexus3, npm repository, docker repository, S3, private pip, nuget, gems, maven and more
Monitoring System	Prometheus, New Relic, Data dog, zabbix, nagios and etc
Logs System	ELK stack, data dog, graylog and etc
Emails	SMTP servers, postfix, or third service like sendgrid
VPNs	VPNs Gateways
DNS	Master and slaves, bind9, route 53
Auth	Authetication systems, AD, LDAP, IAMs and etc
NAS	NAS Gateway

Specification

Field	Functional
System	Systems related it.
Server	Servers deployed by app.
Cluster	If service runned on cluster mode
Spec	Specification about sercive, like endpoint, ports and etc.



Databases

Inventory > *Database*

Databases are it all with data persistence responsibility, can be relational, norelational, in memory, distribuited storage and etc.

We have some specific database, in this case, can have exclusive form

Field	Functional
Oracle	You can register ASM DB, CDBs, configurations like Rac, grid system and golden gate backups
MySQL	Register some features like Master/Slave, Cluster with Aurora, backups services and more.

Oracle

Support version 10g, 11g and 12g

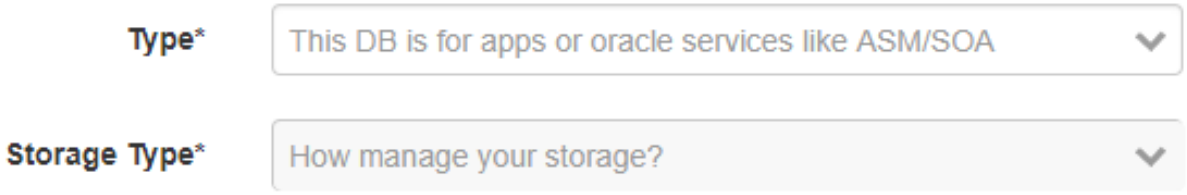
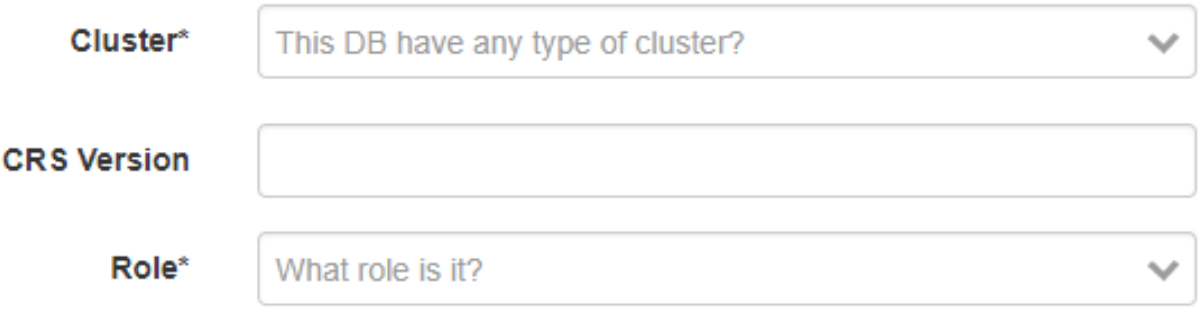
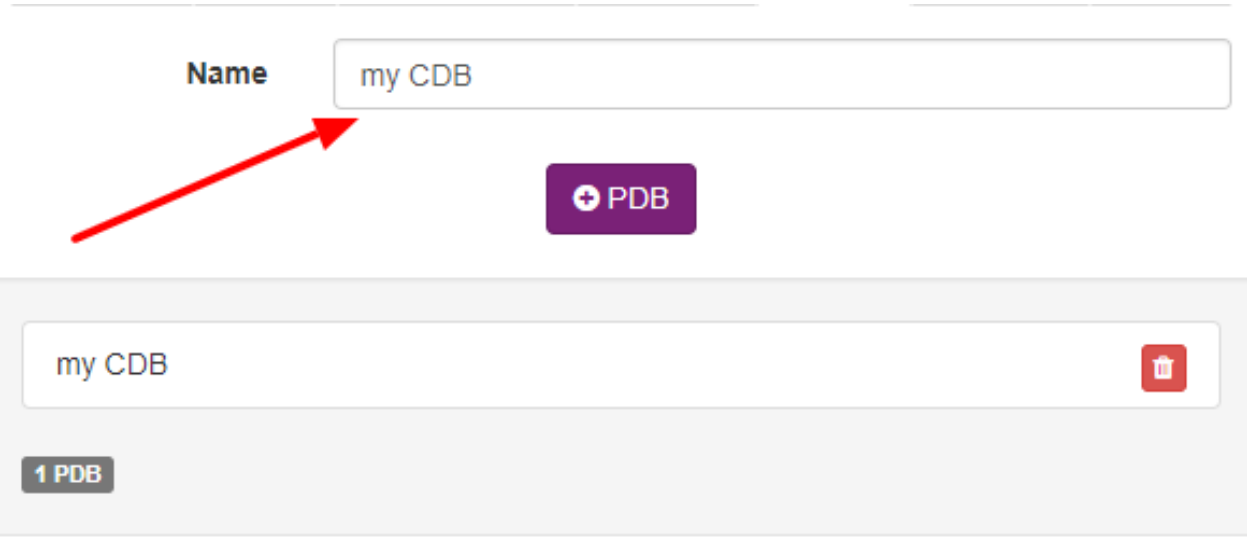


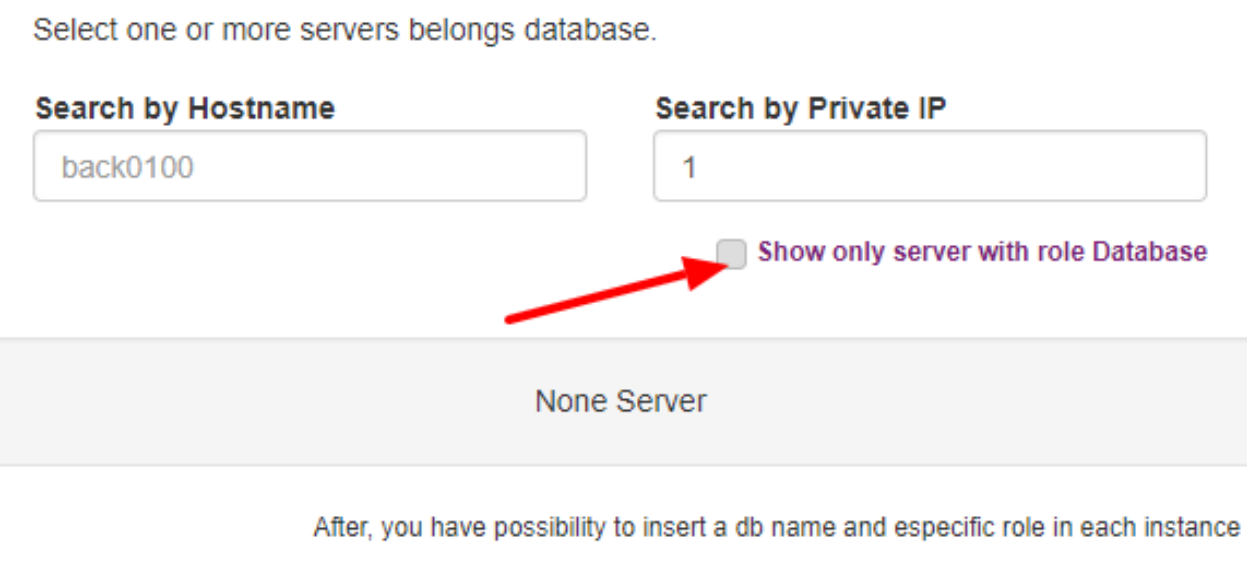
Fig. 1: Choose how Oracle will be storage data, can be local disk, networks disk or ASM.



Choose how Oracle will be run, single node, RAC/Grid mode.



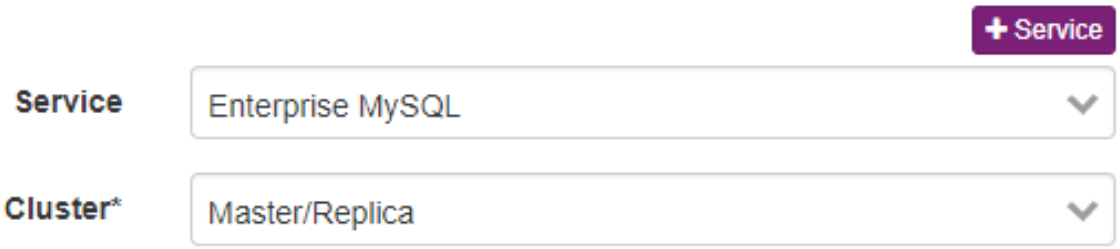
Which CDBS run in oracle database.



Which servers this db run, if is single node, its only one server, but if is rac setup, will be run in multiple servers.

MySQL

Support MySQL, AWS Aurora, MariaDB, Percona and etc



Version and mode to run.

Other databases

Partial support whitch all bases

Version and mode to run.

Field	Functional
Spec	Point information like endpoint, port, commands, health check, its good for doc.
Datacenters	Provider, (only by third party services)
Server	Servers deployed by db.
CDBS	Used only by Oracle DB
System	Systems related it.

LoadBalances

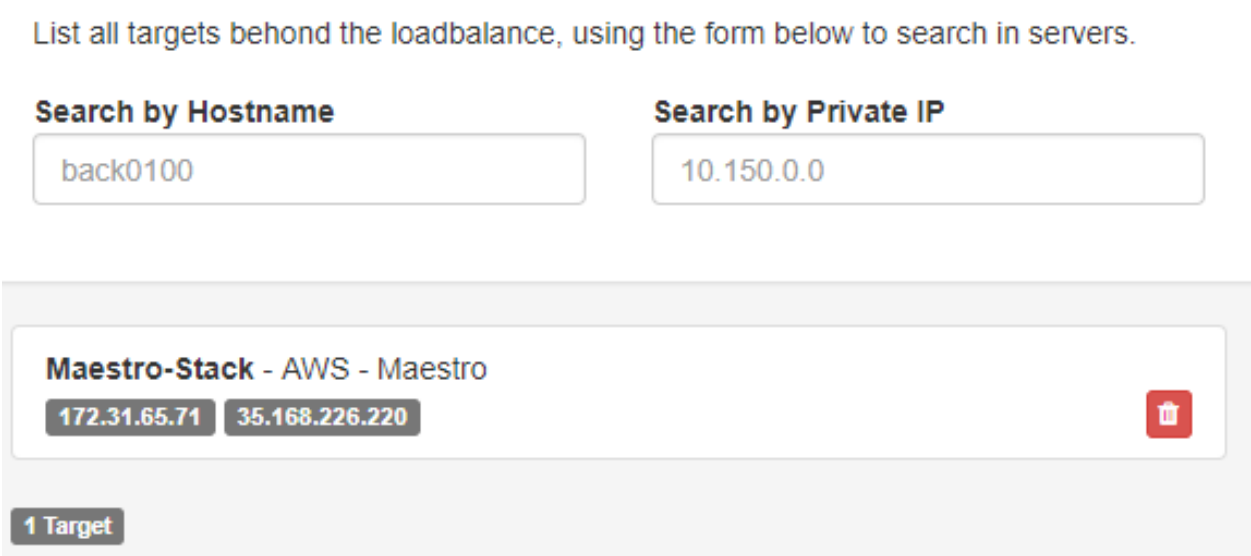
Inventory > *Loadbalance*

Service with responsibility of distributed request through other servers

Field	Functional
Service	Which is service?

Field	Functional
Targets	Which servers this lb send it
Servers	Which servers this lb still installed
Spec	Endpoint and healthcheck

Docs a endpoint and healthcheck used in app.



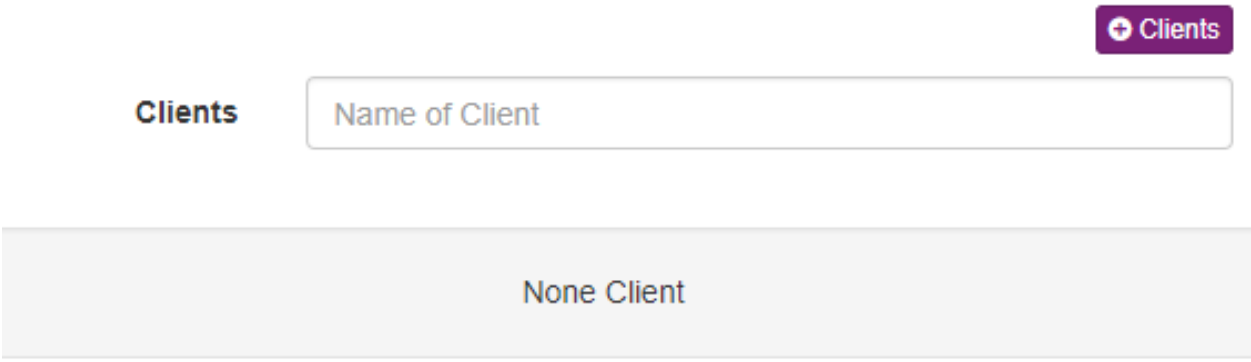
Select loadbalance targets.

System

Inventory > *System*

A group of application, databases, loadbalances and etc, compond a unique system.

Field	Functional
Links	Some useful links
Clients	Relation to this system and client



Select owner of system

Clients

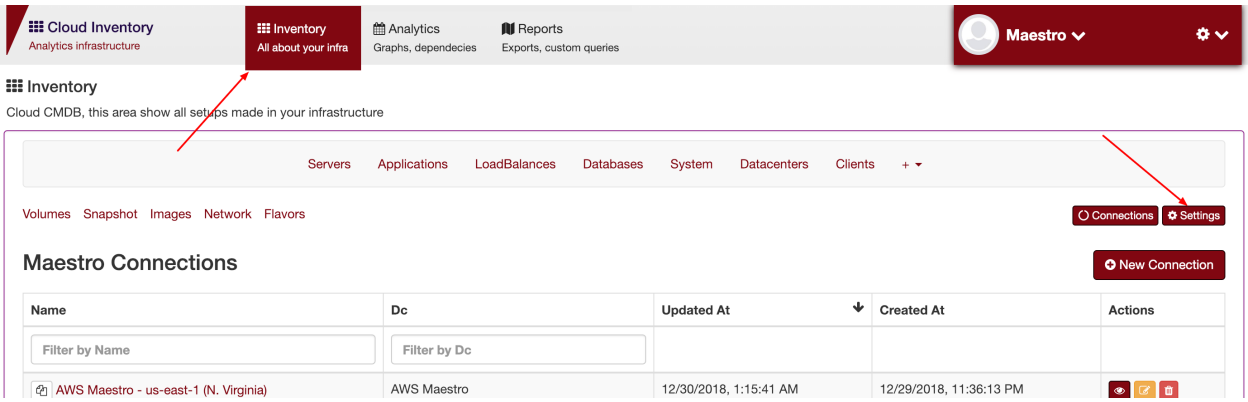
Inventory > *Clients*

SLA owner, clients

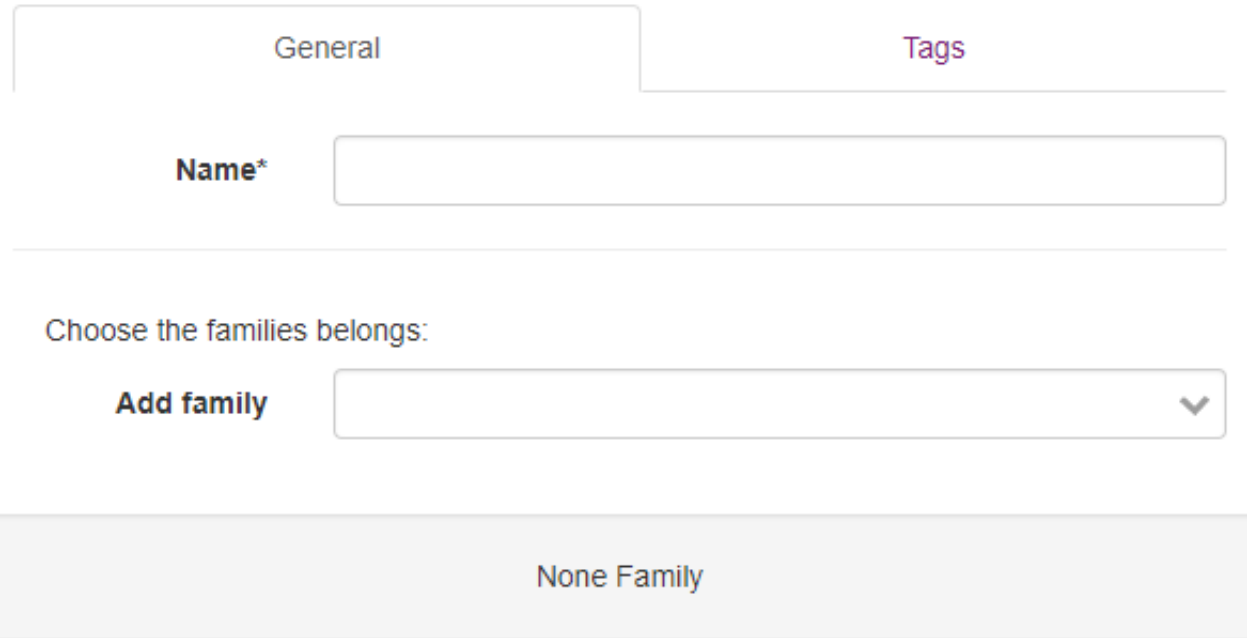
Field	Functional
Contacts/Channel	How contact the client

Services

Inventory > Settings > Services

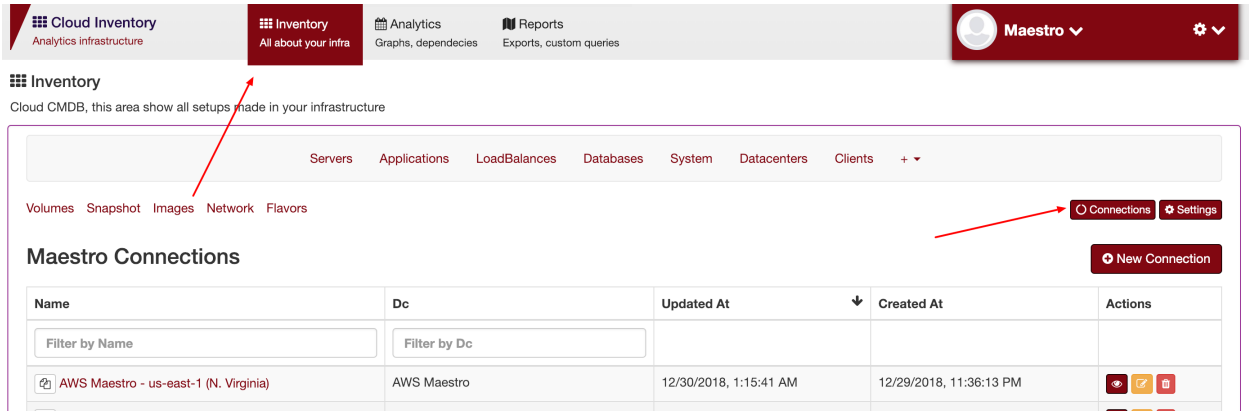


Common program running inside on server



Create a new service, to use in server and any app.

4.2.2 Auto Discovery



To setup new connections in auto-discovery:

- Create datacenters (select all regions with you like to discovery)
- Create a connection - Select previous datacenters created and regions
- Pass aws key and secret - Maestro need only a readyonly permission, the best pratice is, create a specific key for maestro

1 - Create datacenters

2 - Create connection - Go inventory > connections.

Create a iam key with read only permission.

Access connection

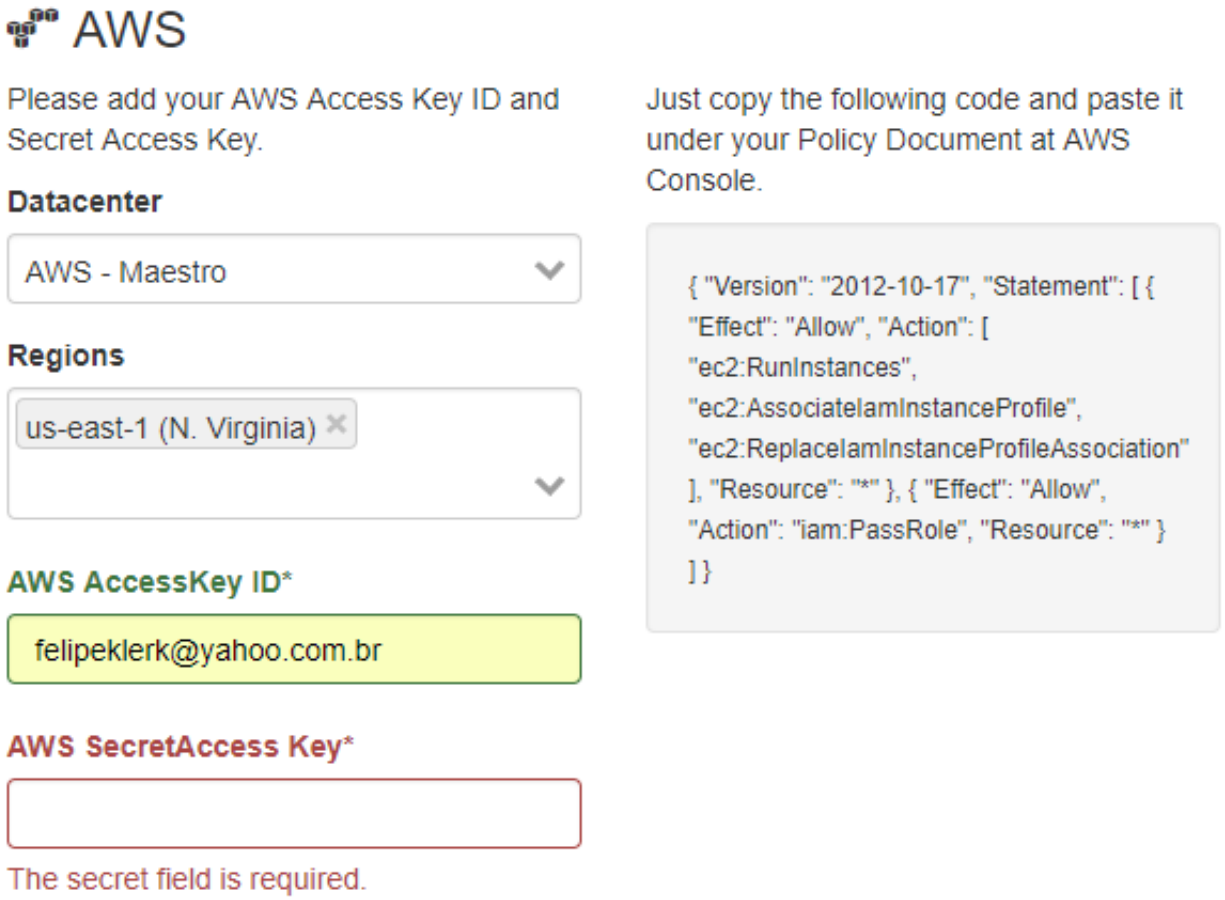
In version 0.1, we have two providers:

Connectiong with AWS

To register one aws account use access_key and secret_key

Synchronized and permissions to grant.

server-List	ec2 describe_instances
loadbalancer-list	describe_load_balancers and describe_load_balancers
dbs-list	rds describe_db_instances
storage-object-list	s3 list_buckets
volumes-list	ec2 describe_volumes
cdns-list	cloudfront list_distributions
snapshot-list	ec2 describe_snapshots
images-list	ec2 vdescribe_images
security-list	ec2 describe_security_groups
network-list	ec2 describe_vpcs, describe_subnets, describe_vpc_peering_connections, describe_vpn_gateways, describe_vpc_endpoints, describe_route_tables, describe_network_interfaces, describe_nat_gateways and describe_network_acls



Setup connection with AWS

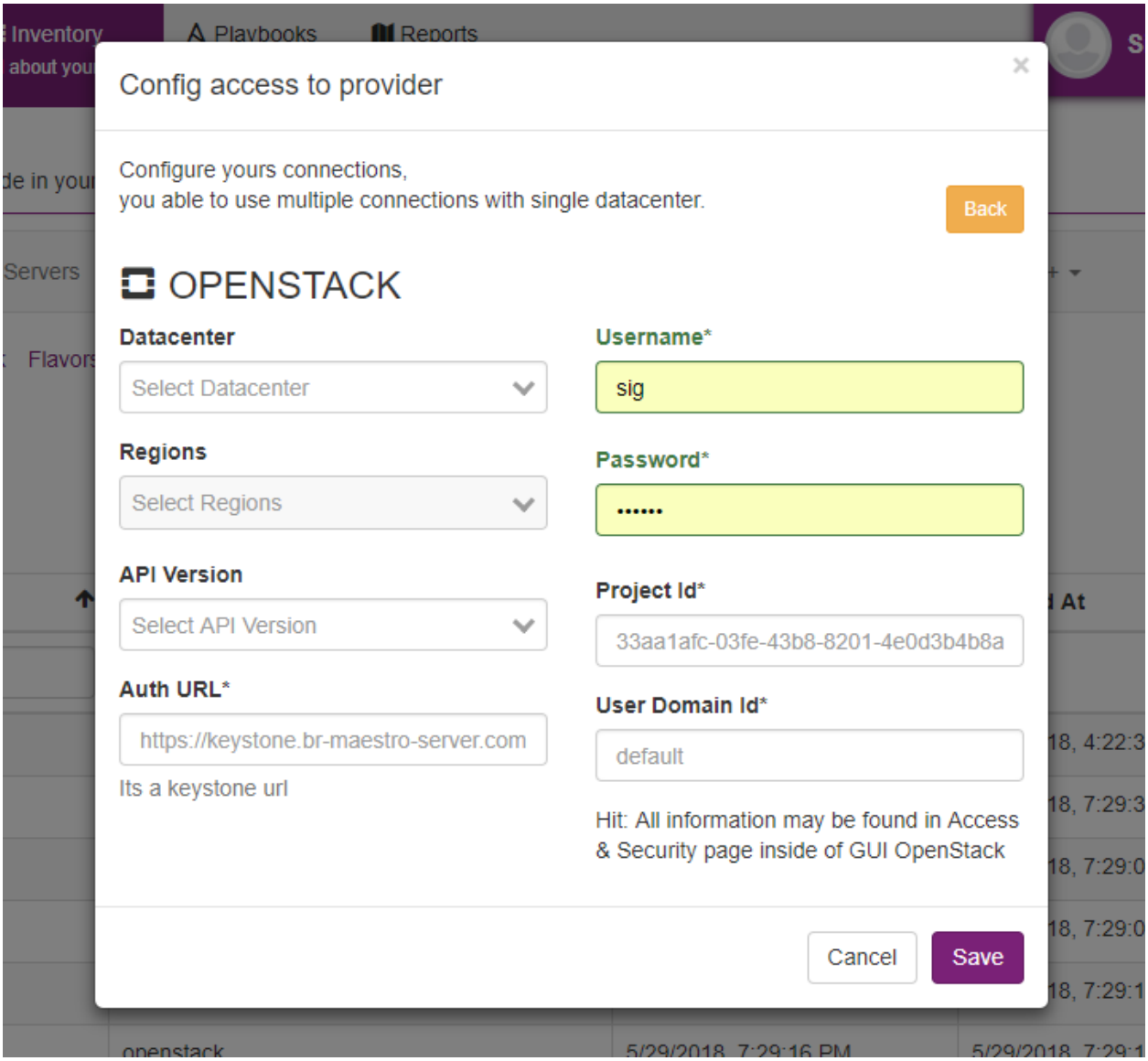
Connectiong with OpenStack

To register one openstack account, use project name, url api, user, and password.

Synchronized and permissions to grant.

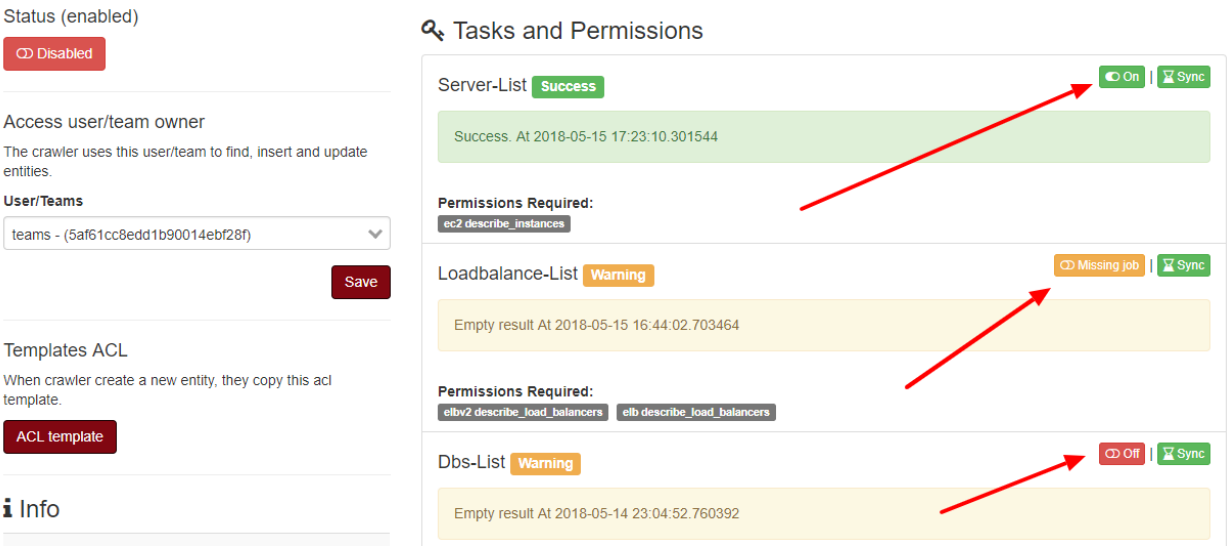
Server-List:	servers compute
Loadbalance-list:	load_balancers load_balancer
volumes-list:	volumes block_store
snapshot-list:	block_store snapshots
images-list:	compute images
security-list:	network security_groups
flavor-list:	compute flavors
network-list:	network networks, subnets, ports and routers

If you like, choose how the resource will be synchronized with an active and inactive button.



Setupconnection with OpenStack

Nota: PS: There is scheduler job, its automatize sync, this schedule will be activated by default, and each resource have our own time, in the example, server-list will be synchronized for every 5 minutes, networks stuffs normally happen for every 2 weeks. You can the time using in each resource, more details see schedulers.



Enable and disable the job

FAQ

- **Permission error**

  If through Unauthorized error, you need to grant ready only permission, in AWS you need to create IAM and grant these permissions.

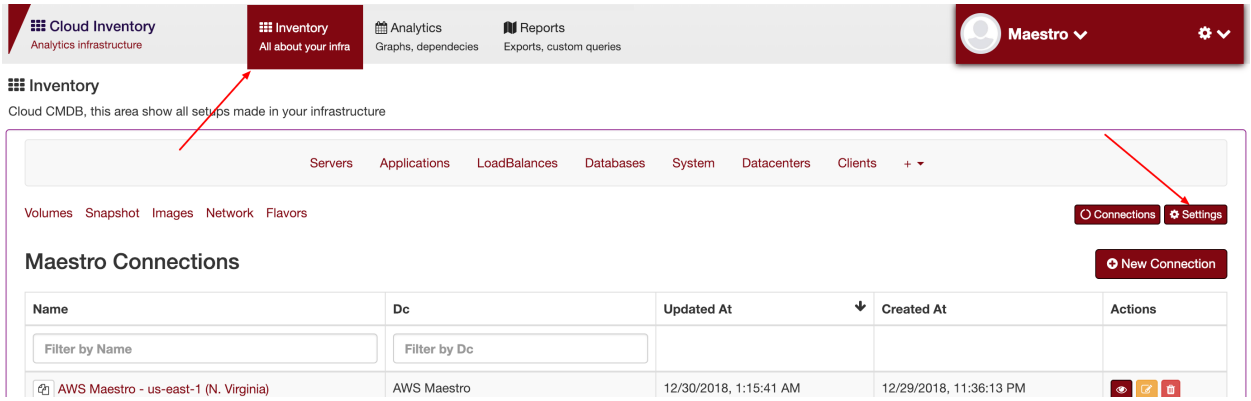
- **Infinitive process loading**

  Its common problem, Maestro needs two services to execute a successful synchronize, Discovery APP and RabbitMQ, normally when discovery app is down, we have infinite process message (because server app notify to start a process, and discovery app need to finish with Success process). Guarantees if discovery app up and running and if it's connected correctly with rabbitmq.

For debbug, use stdout docker, like

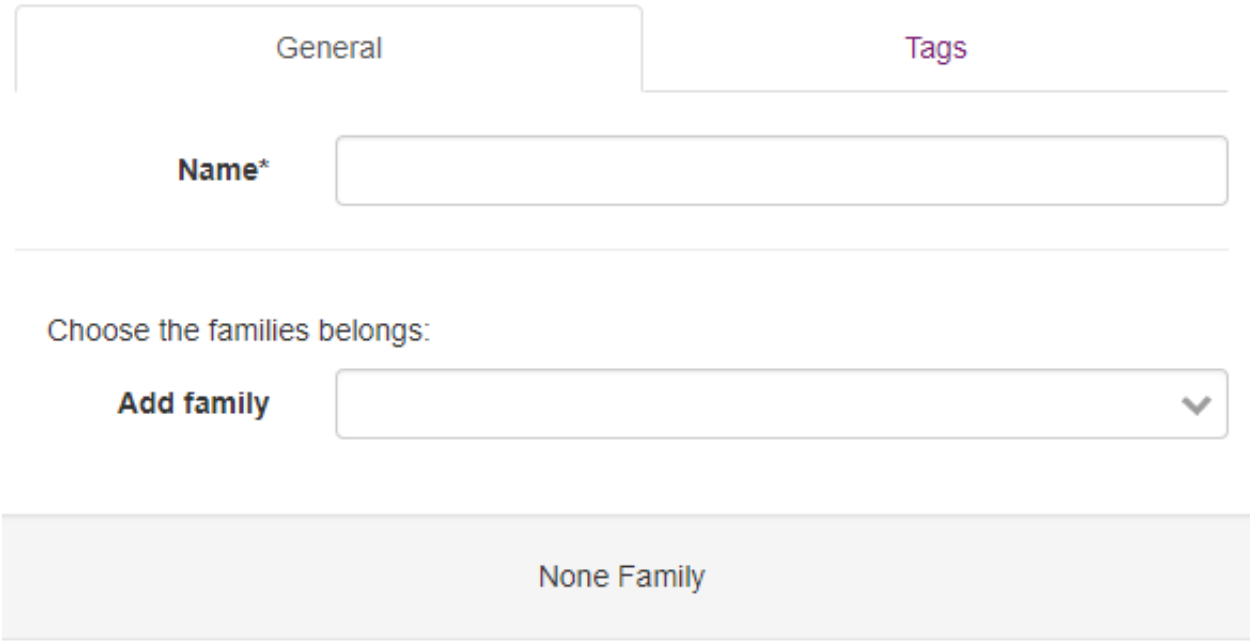
```
docker-compose logs discovery-maestro
# or
docker-compose logs discovery-celery
```

4.2.3 Config / Settings



Services

You can create and delete new services used in servers and apps services field, choose a name, family and some tags.

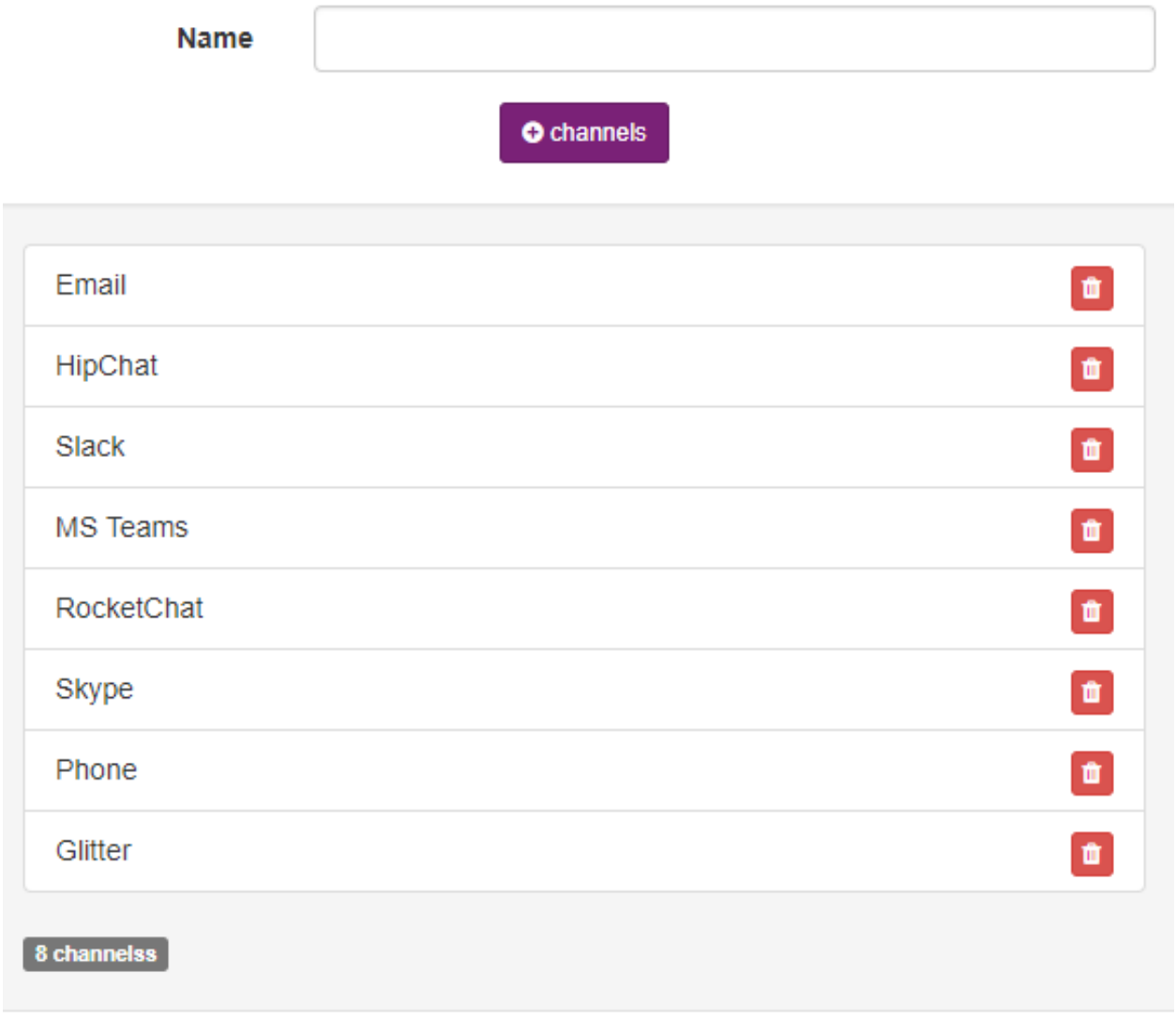


Register a new, or update service.

Config Options

Config options, its used in point part options, like environments, types of services, and time to schedule updates providers lists.

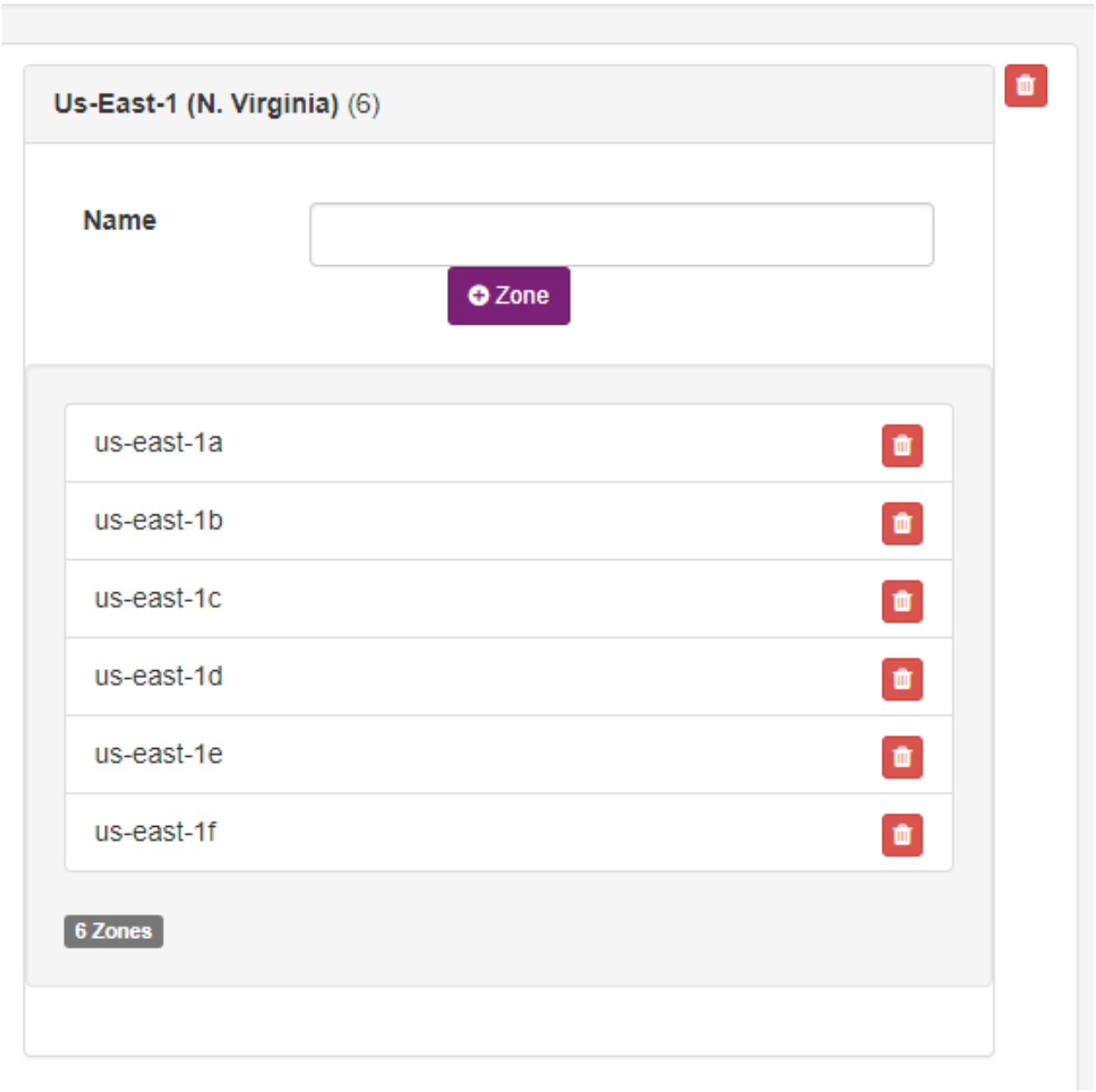
application_options	Options of apps
clients_options	
connections	Time scheduler and crawler connections
database_options	
datacenter_options	
env_options	
server_options	
services_options	Initial setup of services
system_options	



Maestro configs, created when run migration command.

Regions and zones

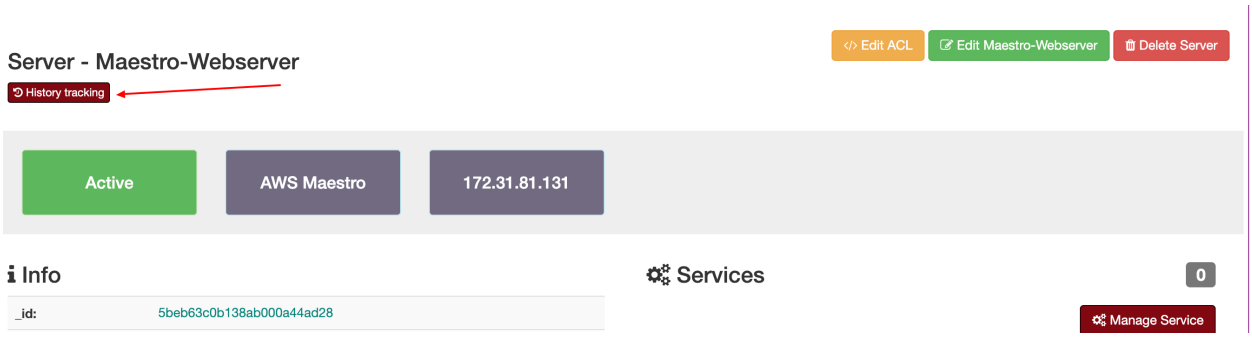
Regions and zones, if you like, its possible to configure and pre-define some regions and zones.



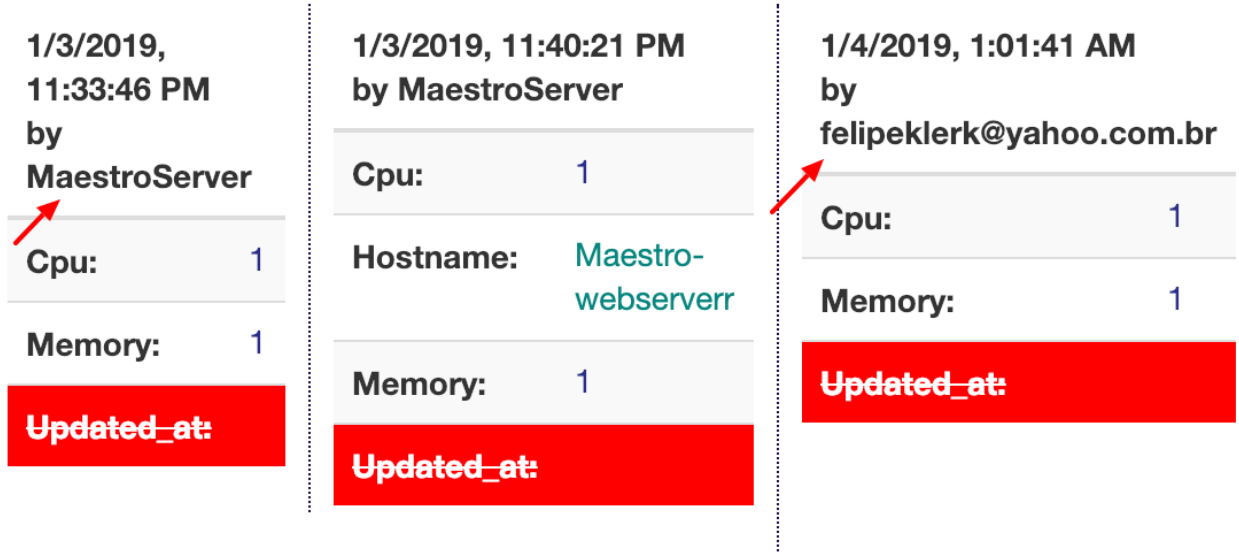
Pre-define regions and zones.

4.2.4 History Track

Inventory > *Single Application* > *History Track*



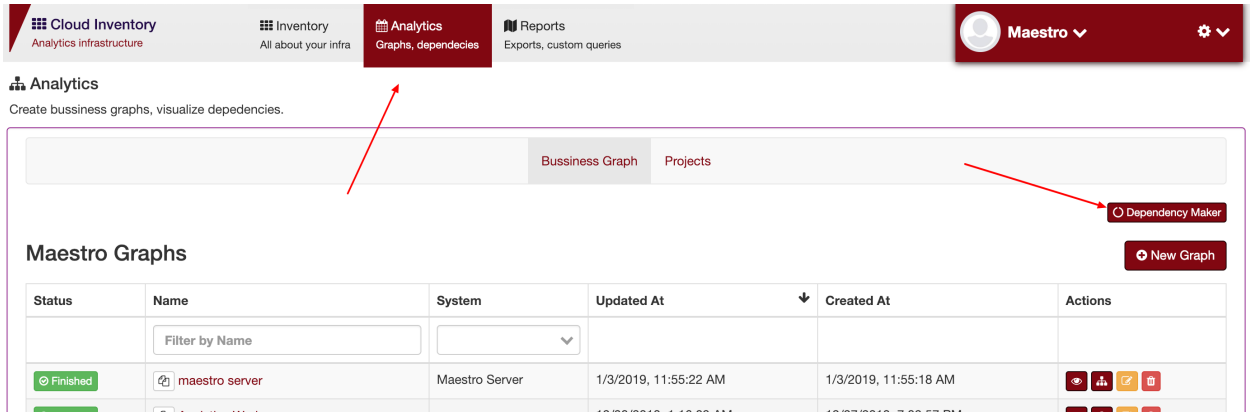
Every change made by user or discovery crawler its record on history track panel.



Tracking Page

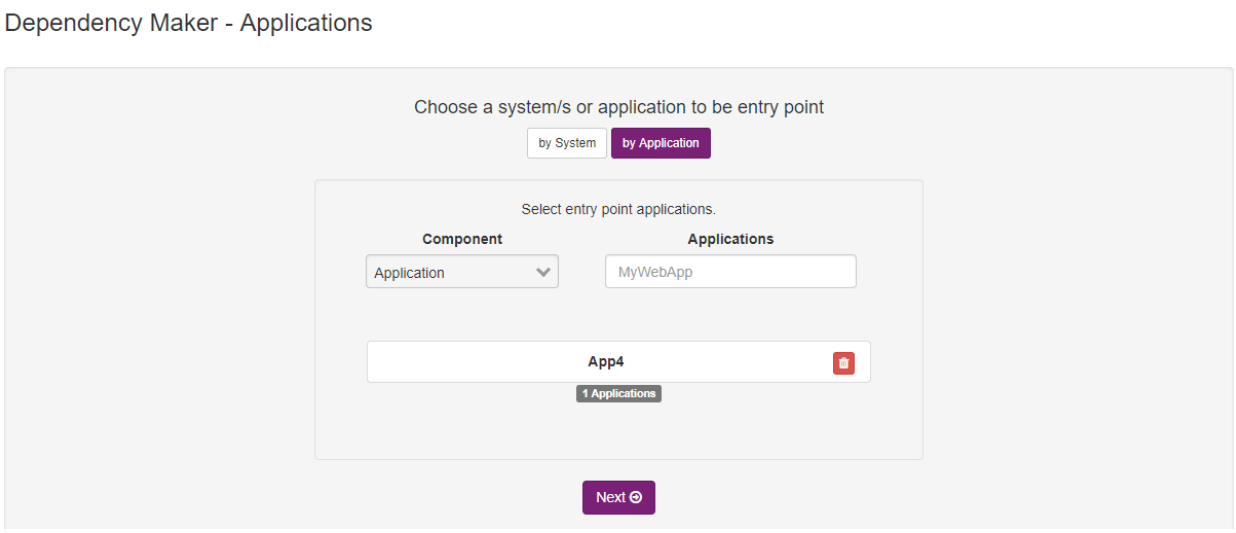
4.3 Graphs - Architecture maps

4.3.1 Create a dependecy tree

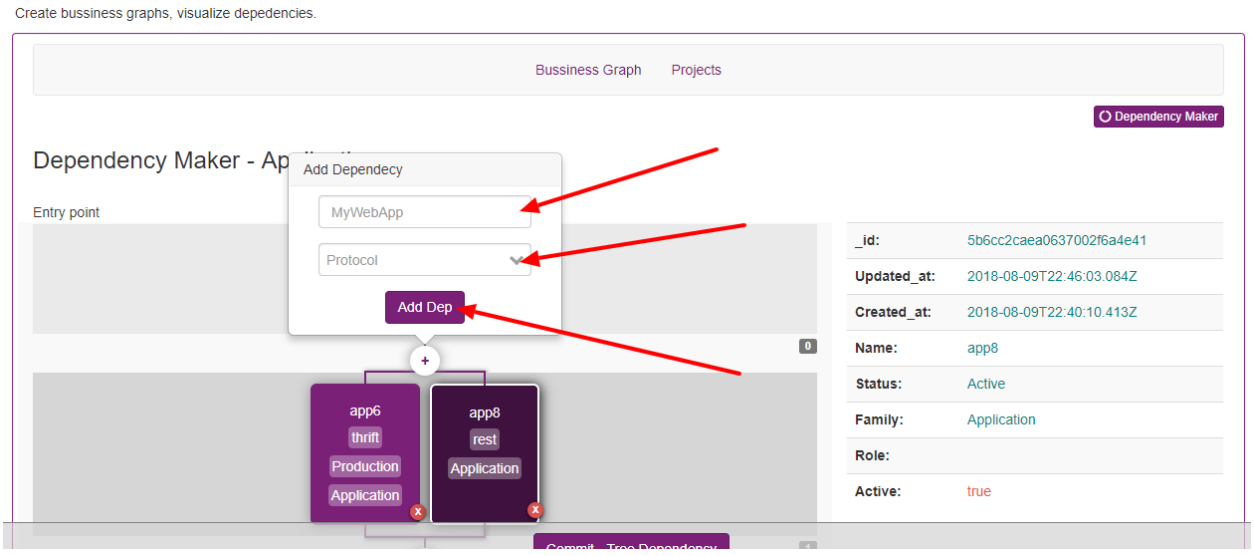


Dependency tree maker it's a feature which to turn easy to link dependency service, you can link an entire graph map in minutes, without access to each service and set all dependency.

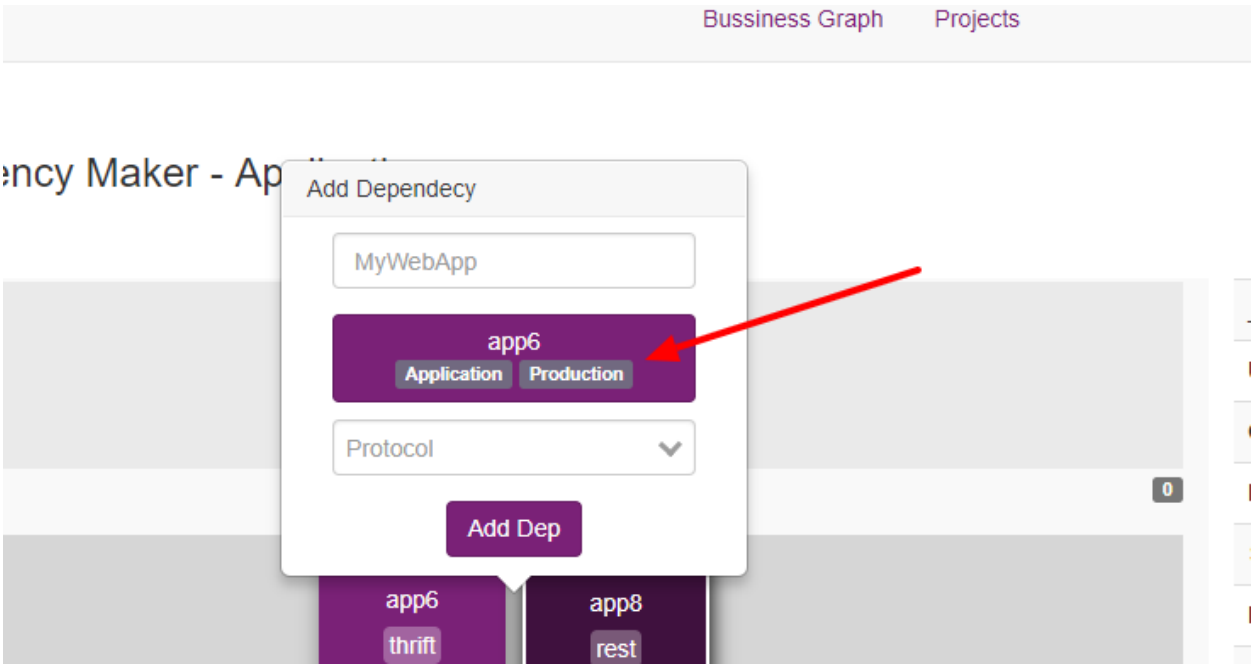
Starting select a system or a entry application, these application will be the first application.



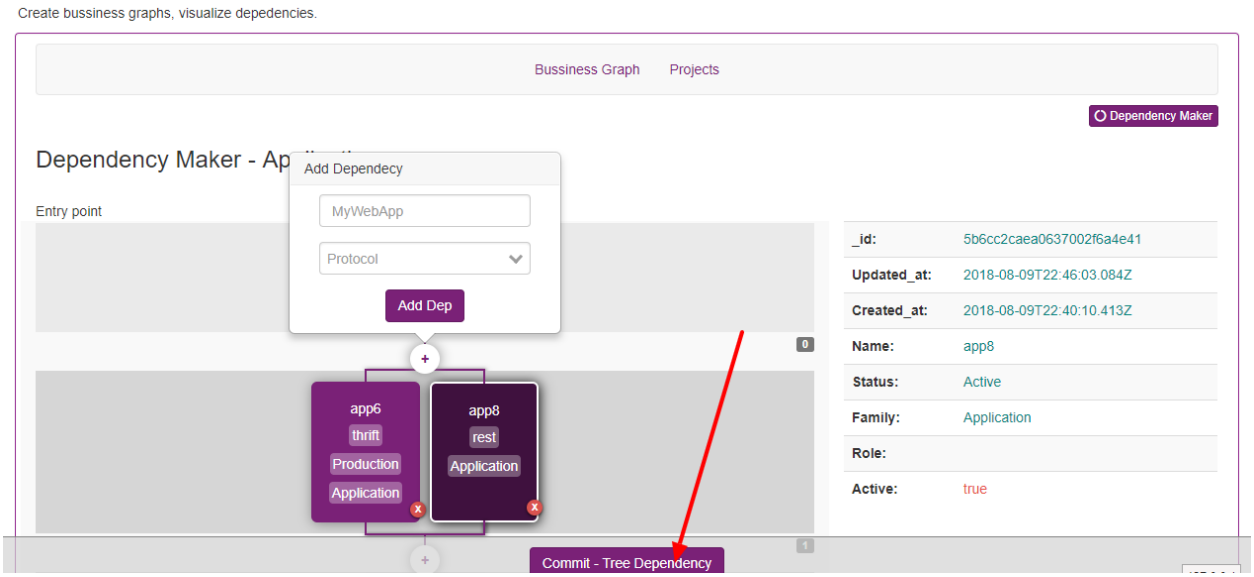
After selected, you can select any application (will be active the app), and use plus button to add new dependecies.



Click on app, and add to dependency



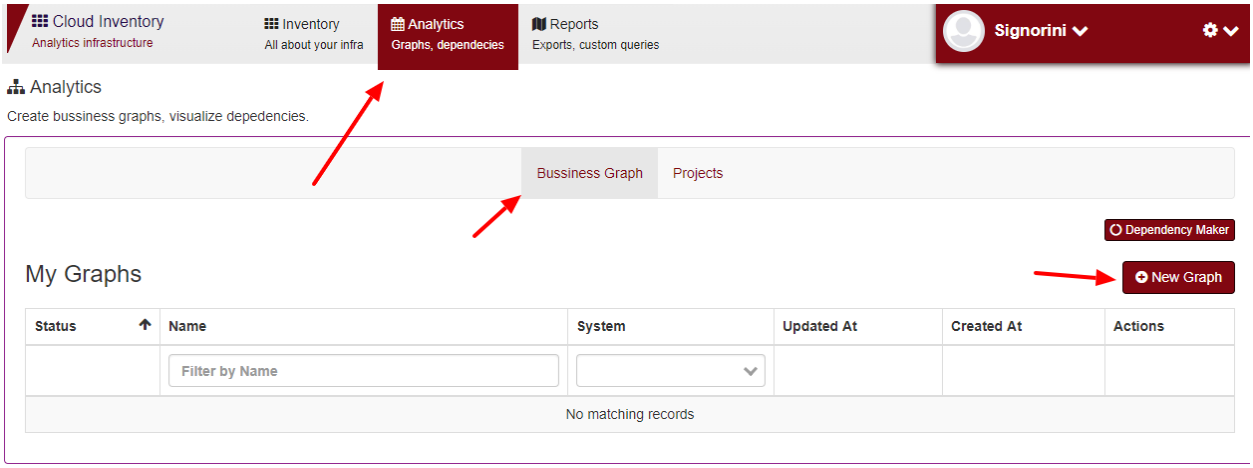
After create the tree, salve the tree, clicking on commit button.



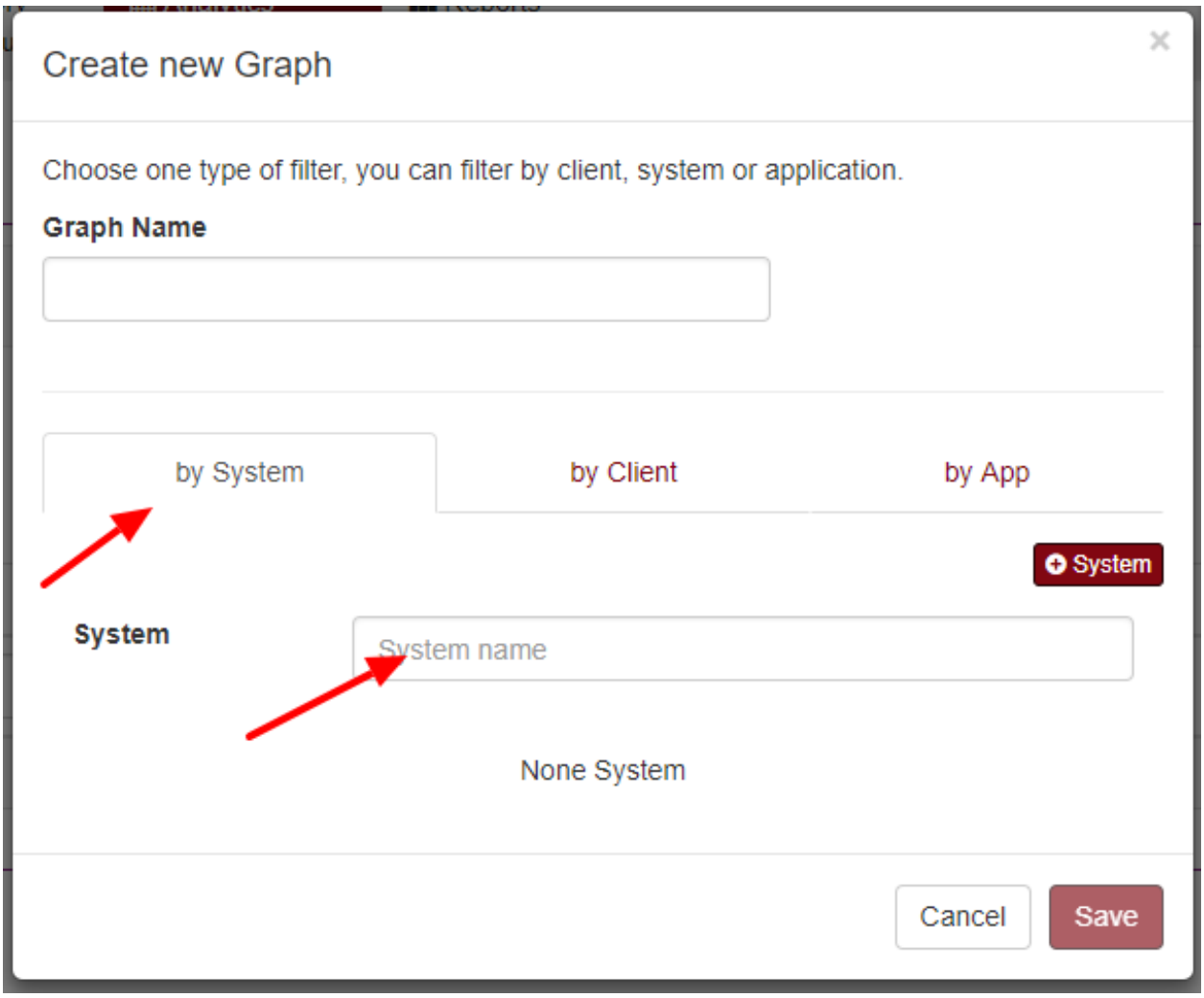
4.3.2 Bussiness Graphs

You can create an architecture of one or more systems, relation of each application can be made in application single page on dependency field, or fast way using dependency maker page.

To create a graph, go to Analytics > Bussiness Graph > New Graph



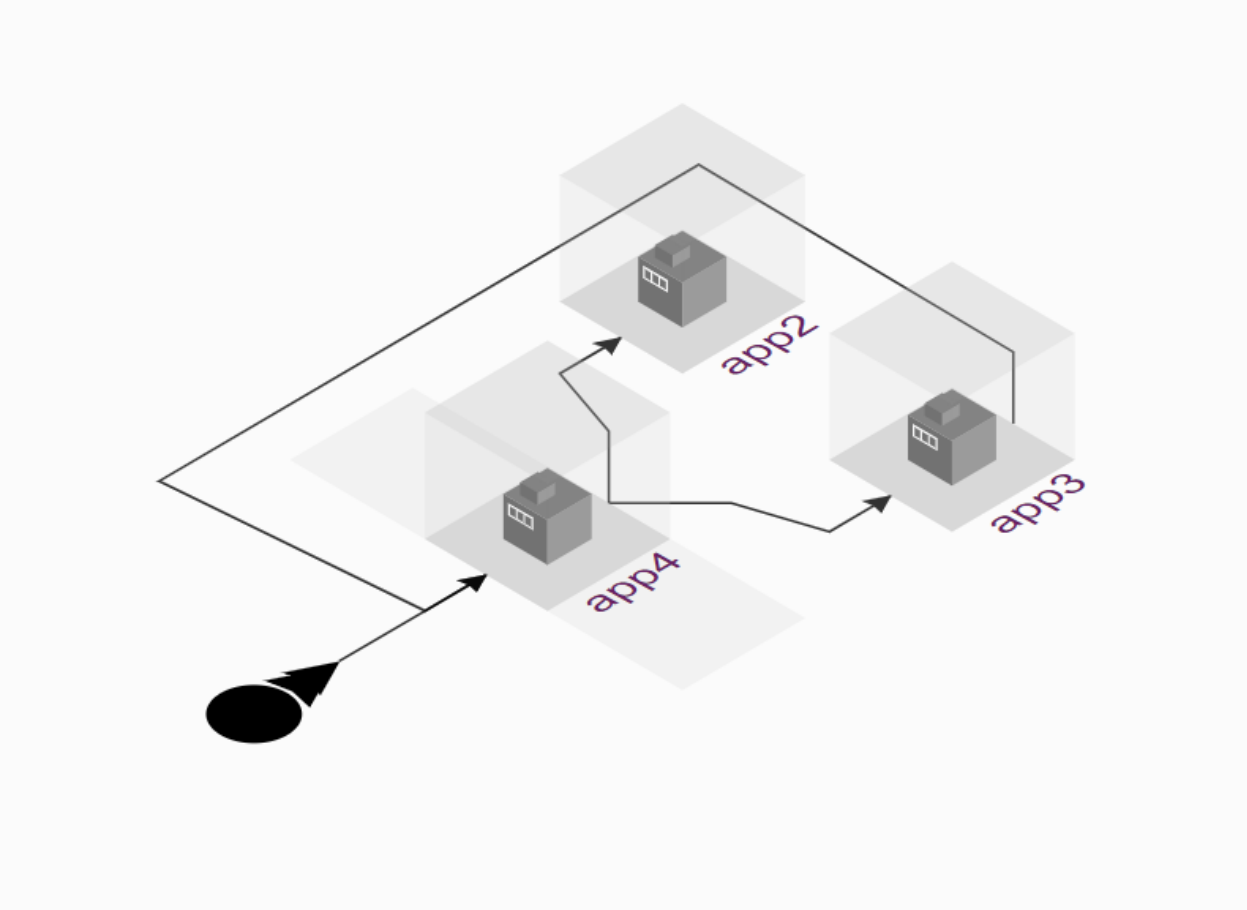
On create graph modal, have three options, create by System, by Client or by App



by System	Will use all entry apps registered on selected system.
by Client	Will use all entry apps registered in all client systems.
by App	Will use these applications as entry application.

Entries applications

Maestro use a entry applications to start create a graph tree based on deps links. Can be one and more.

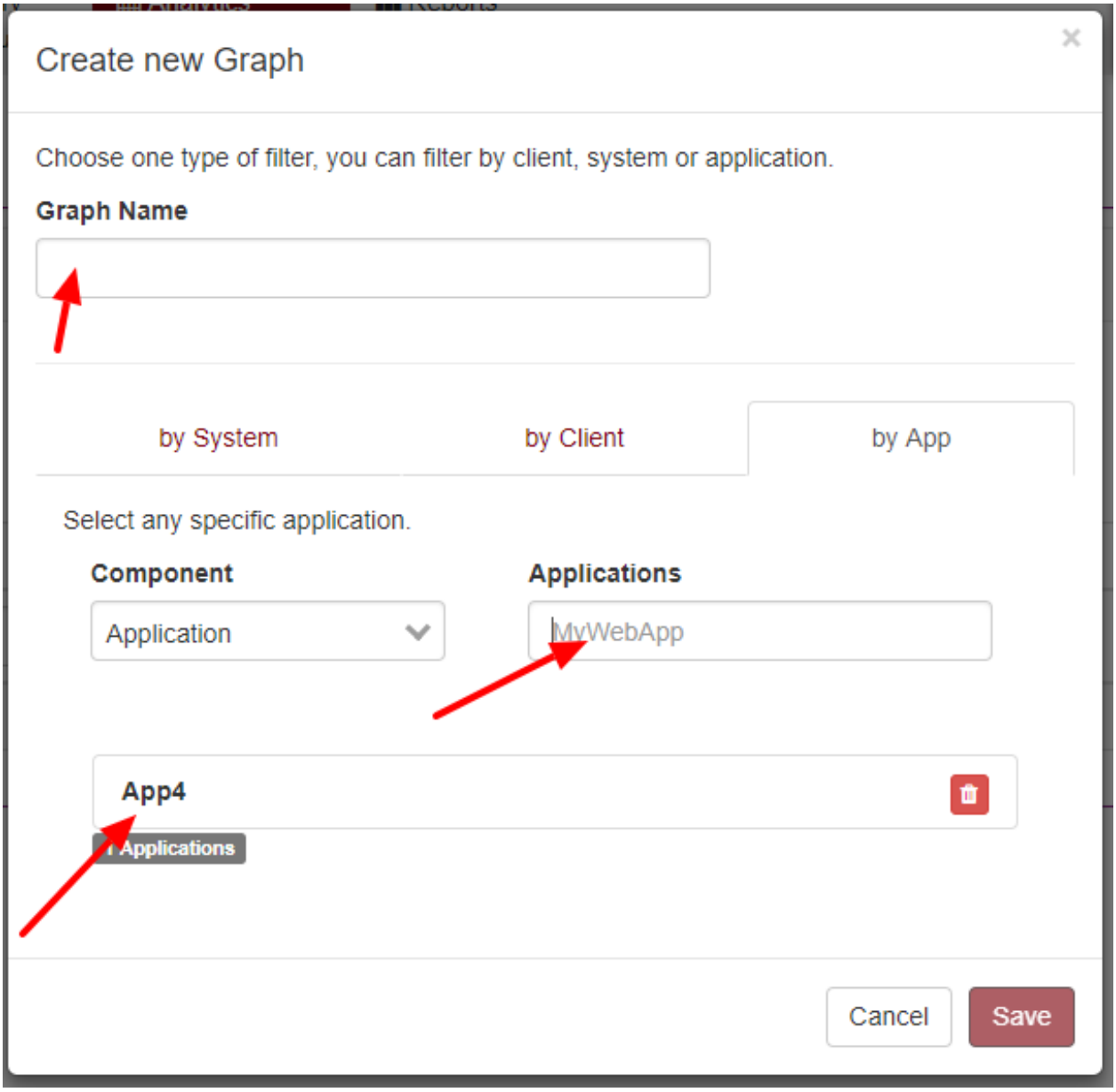


In this example, app4 its a entry applications.

Nota: Except a entry applications, only linked apps will be drawed.

Nota: You can choose with applications can be used as entry point on each system. (It's a endpoint tab).

Filled graph name field, and starting to figure out all applications you like to set as entry point application.



We choose to set only app4 as entry point.

After some seconds all map its create, you can see density, total conections, histograms, all clients, systems and applications linked and the graph inself.

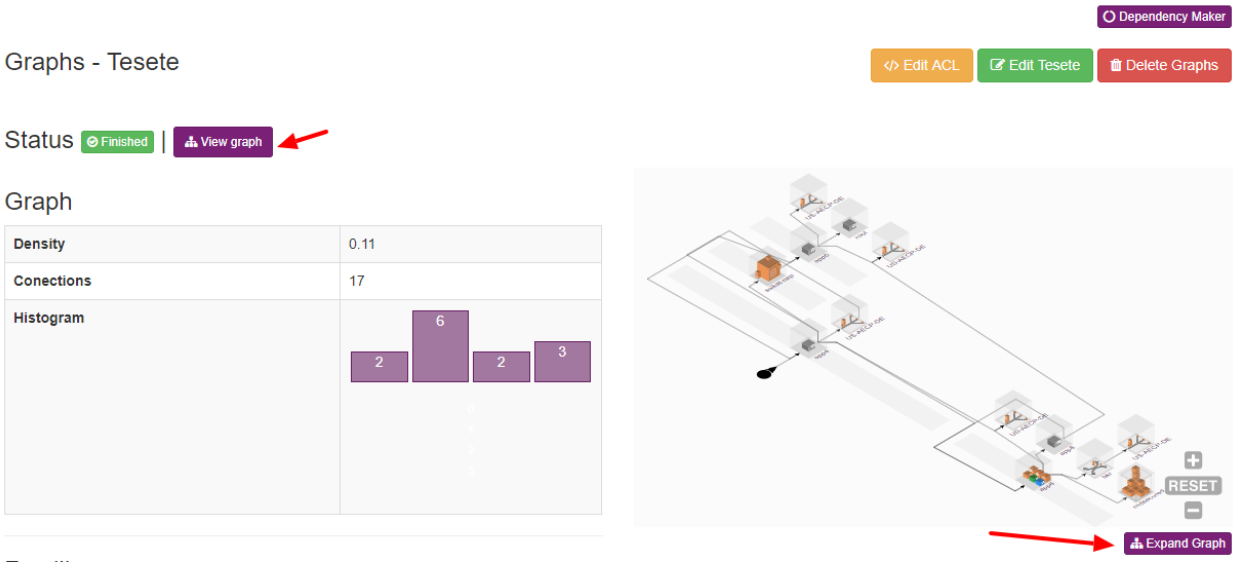
- **Density** - The density for undirected graphs is [d = frac{m}{n(n-1)},] where (n) is the number of nodes and (m) is the number of edges in (G).

The density is 0 for a graph without edges and 1 for a complete graph. The density of multigraphs can be higher than 1.

Self loops are counted in the total number of edges so graphs with self loops can have density higher than 1.

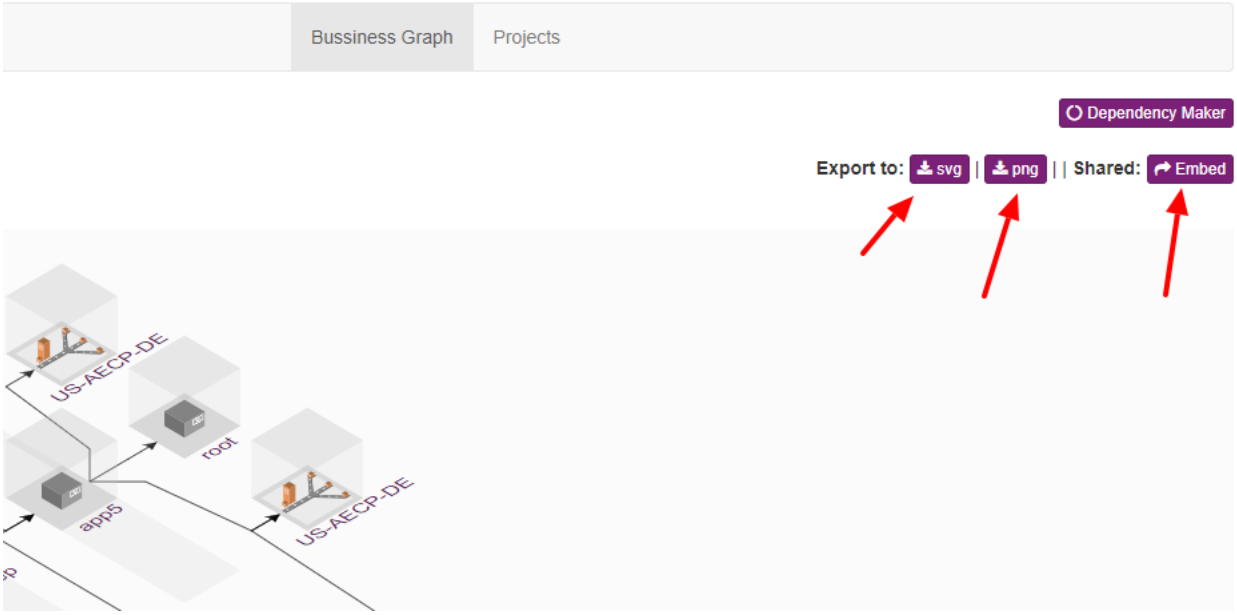
More details - NetworkX Graph - Density.

- **Histogram** - Total by deep dependency.

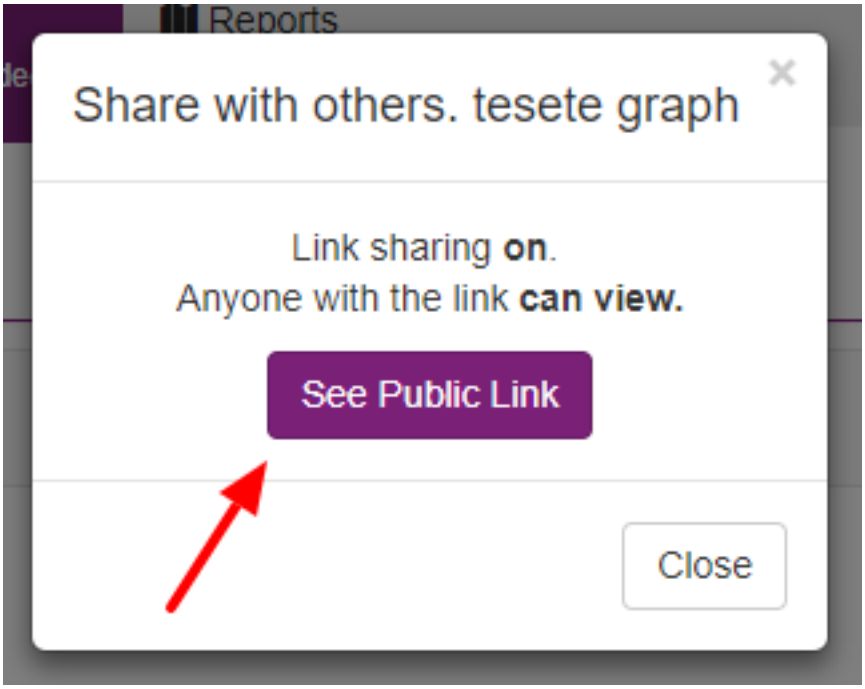


If you like, can expand the graph.

Expanded graph, you can export svg, png or shared this graph. Also, you can see each connection with mouse hover in each line.

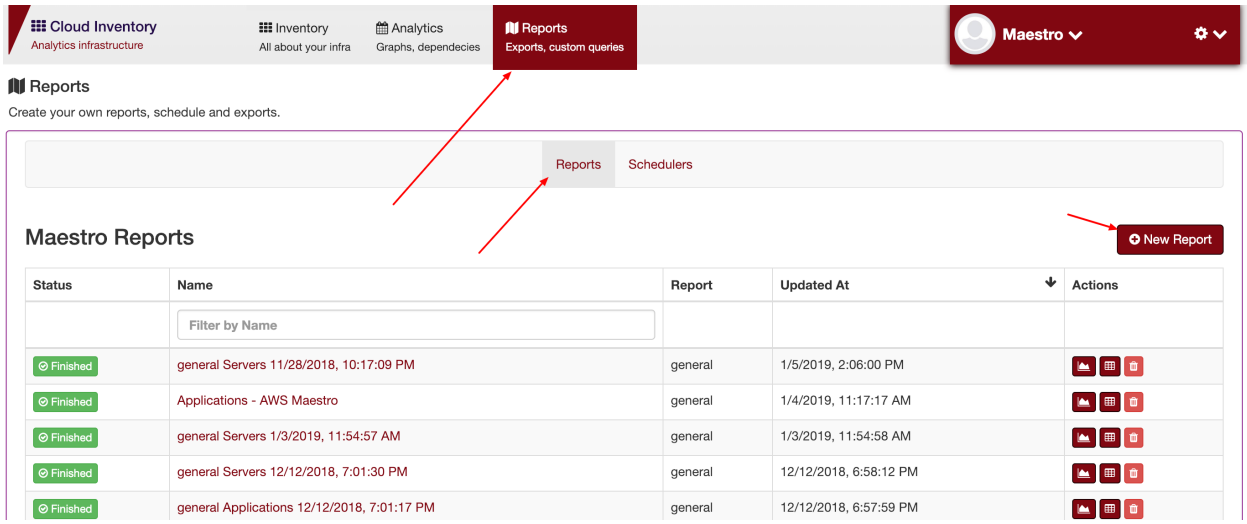


On shared modal, click in «see public link», this will be generate a shared link, it´s possible to embed on external tools, like Confluence.



4.4 Reports - Generate complete reports

4.4.1 Reports



Single table report

General table able to create single report, you can add specific filters.

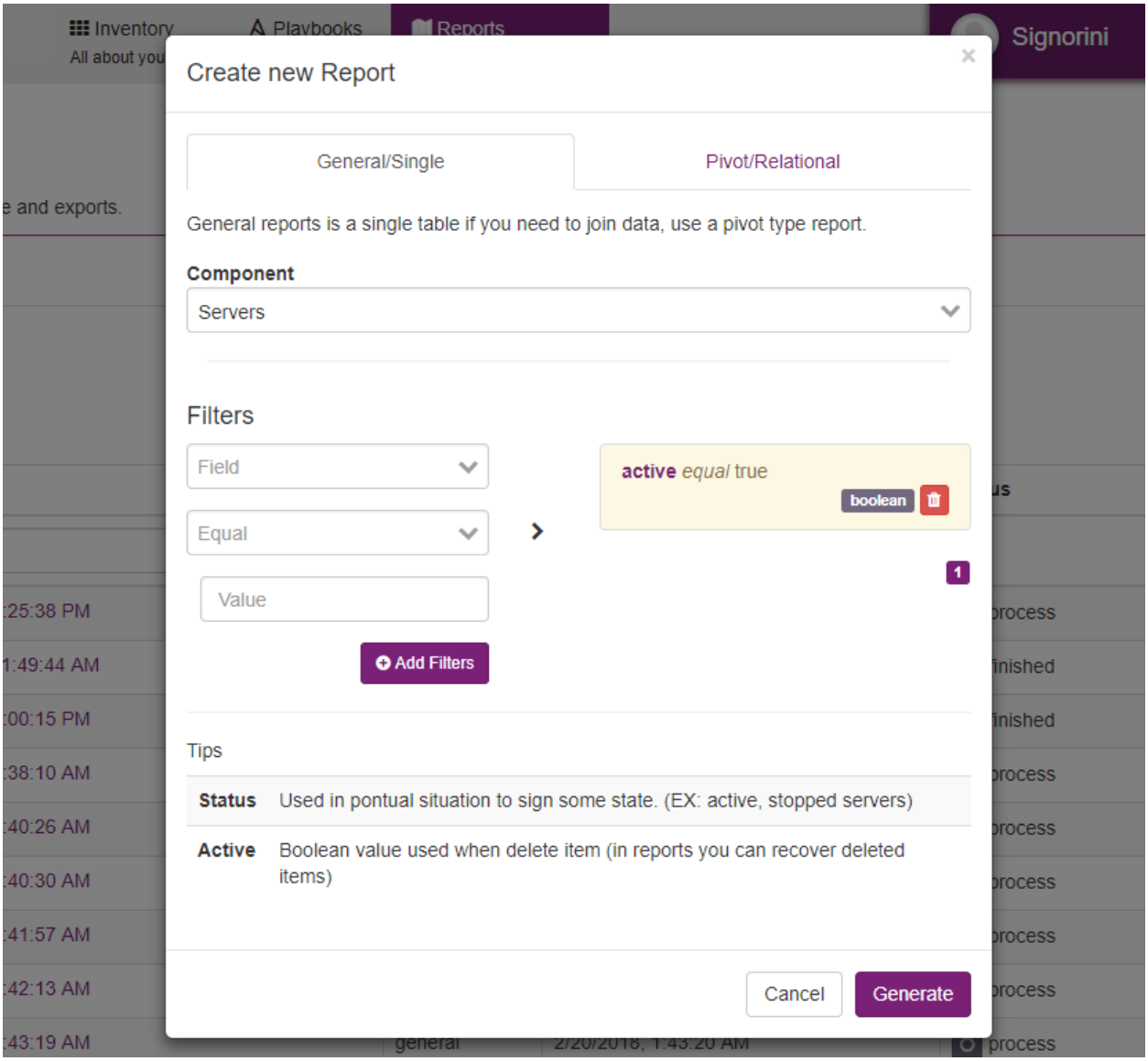
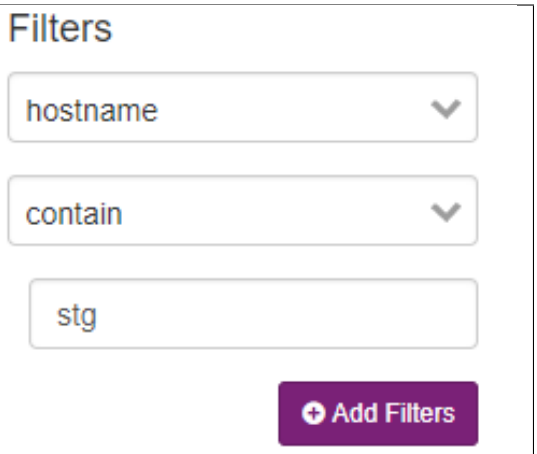
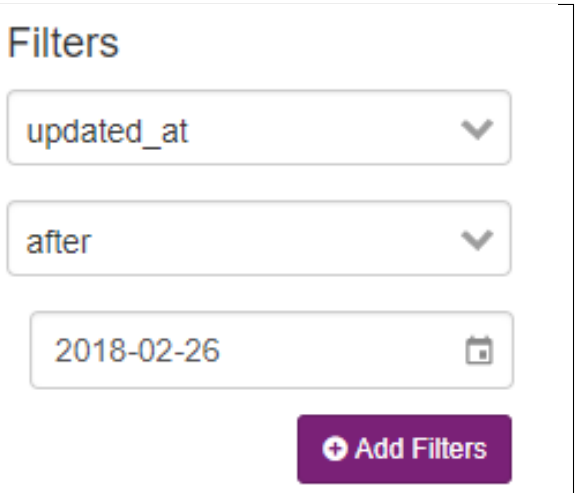


Fig. 2: Single report, use only one entity

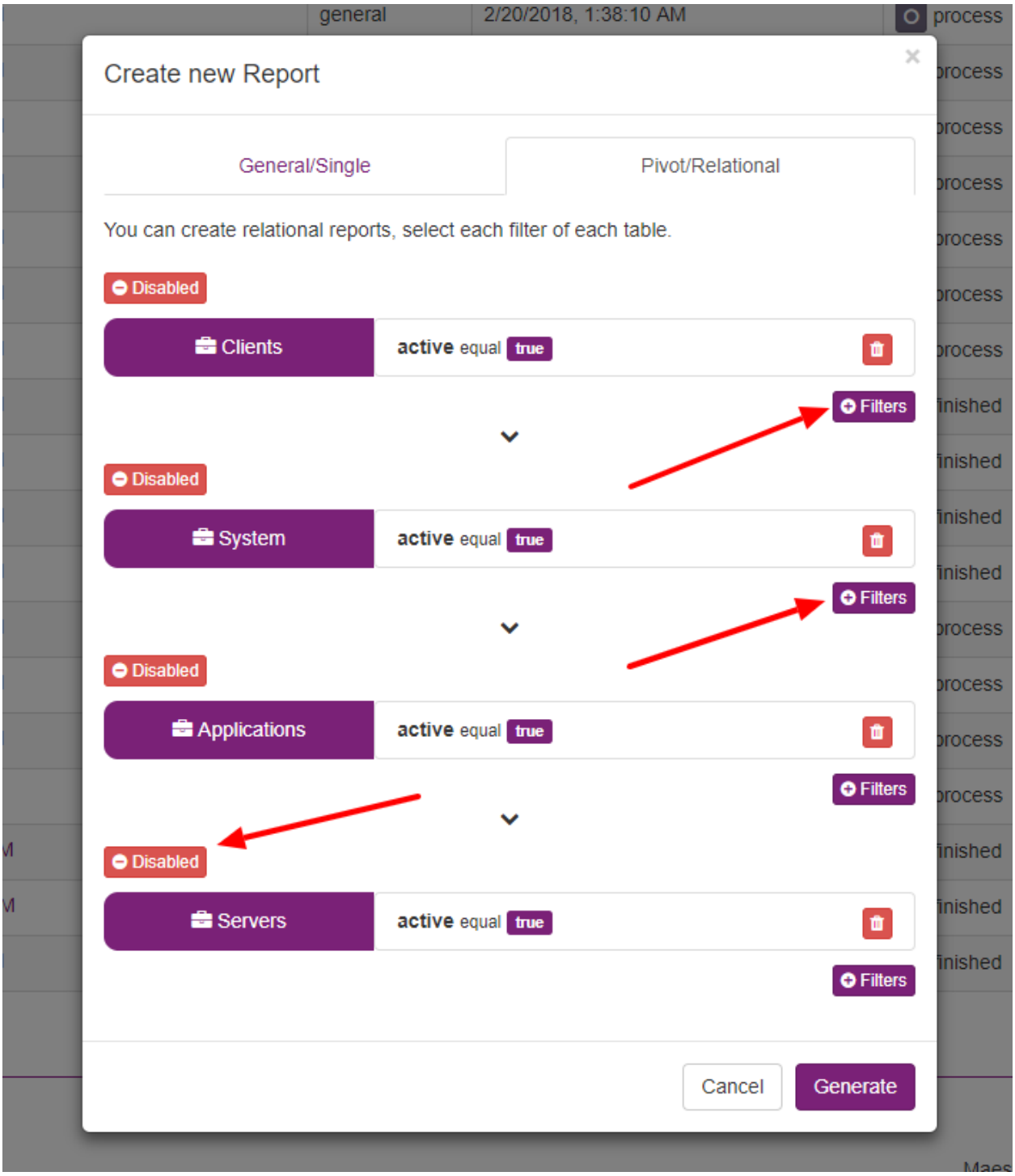
Host-name/name	string	equal/contains	Name or host-name	
Ex: Hostname contais stg, will find somethiing like webserver.stg or stg2.				

Upda-ted_at	date	af-ter/equal/before	Last data up-date	
Ex: Select only the new items udated in this month				

Active	Boolean	true/false	Flag, used to deleted items, possibility to retrived deleted items

Pivot table reports

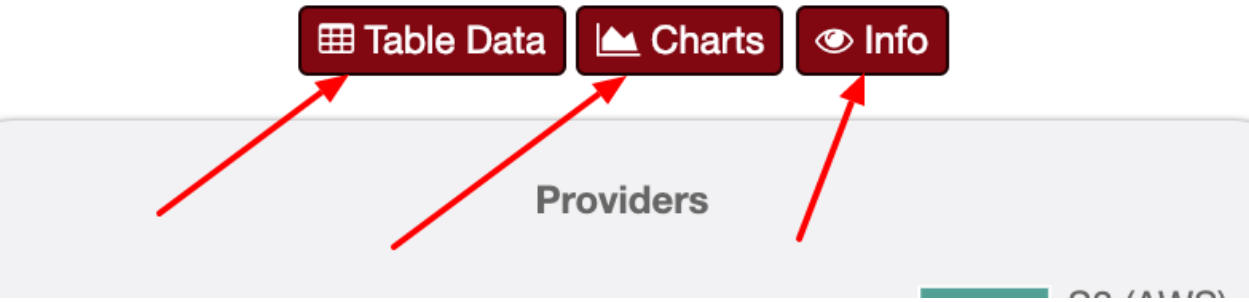
Pivot reports, can create a relational table, relational is, client -> system -> app -> servers, you can use any type of filters in any step.



Nesting relation entities.

Then each report have three pages

- **Charts:** Show a aggregation charts of all entities
- **Table:** List a table data with results
- **Info:** Show info about the report, good for debug propose



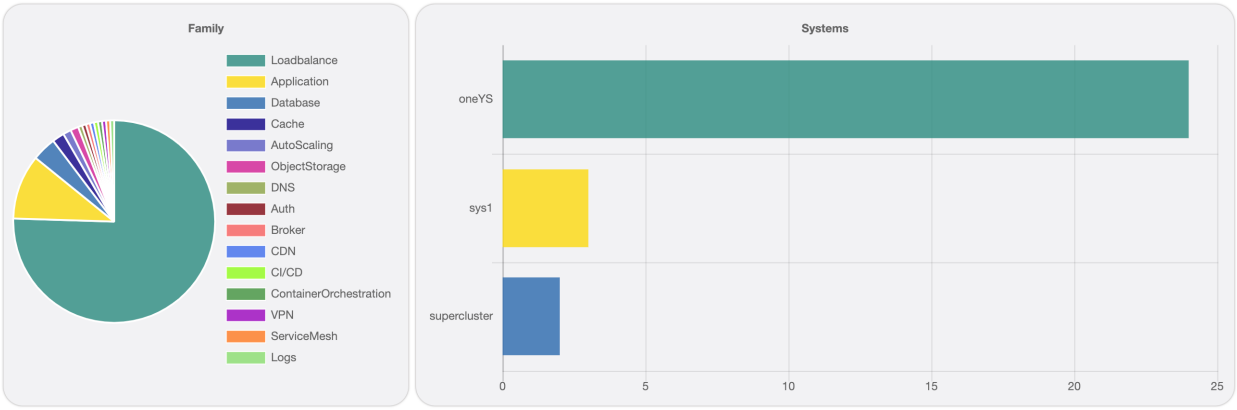
Result report

4.4.2 Report aggregation

Reports > *Single Report* > *Charts*

All reports have chars overview, Maestro is smart to identify with fields can be aggregate and what chart type should use.

Applications charts



A list o aggregate fields:

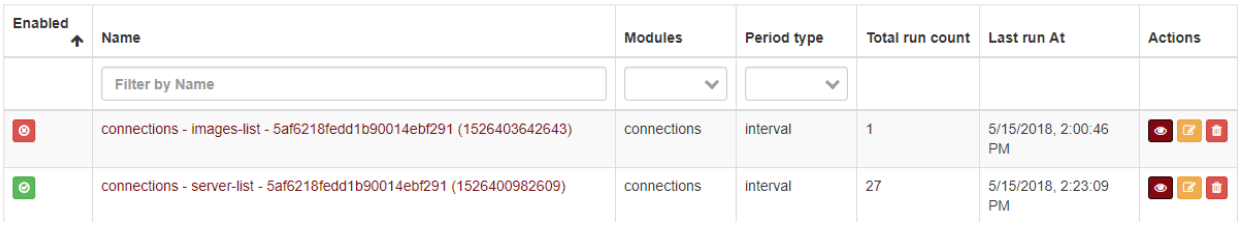
- Datacenter - Providers
- Datacenter - Resource
- Datacenter - Instance type
- Datacenter - Regions
- Datacenter - Zones
- Tags
- Sizes

- Application - Family
- Application - Dependencies
- Application - Deploys
- System by Application
- Clients by System
- System - Entry Applications

4.4.3 Scheduler

Scheduler section normally is used to automatize a polling synchronization in connections and playbooks, but you can create a custom schedule.

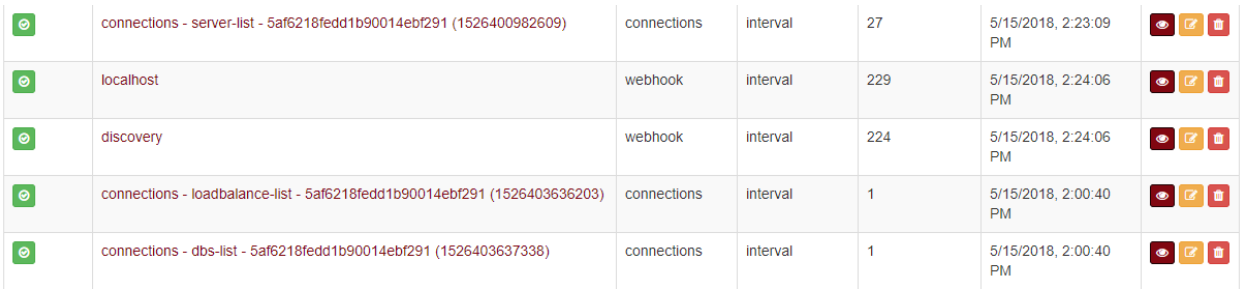
You can list all schedules, the first column show if that schedule is enabled or disabled.



Enabled	Name	Modules	Period type	Total run count	Last run At	Actions
	Filter by Name					
	connections - images-list - 5af6218fedd1b90014ebf291 (1526403642643)	connections	interval	1	5/15/2018, 2:00:46 PM	
	connections - server-list - 5af6218fedd1b90014ebf291 (1526400982609)	connections	interval	27	5/15/2018, 2:23:09 PM	

List Scheduler

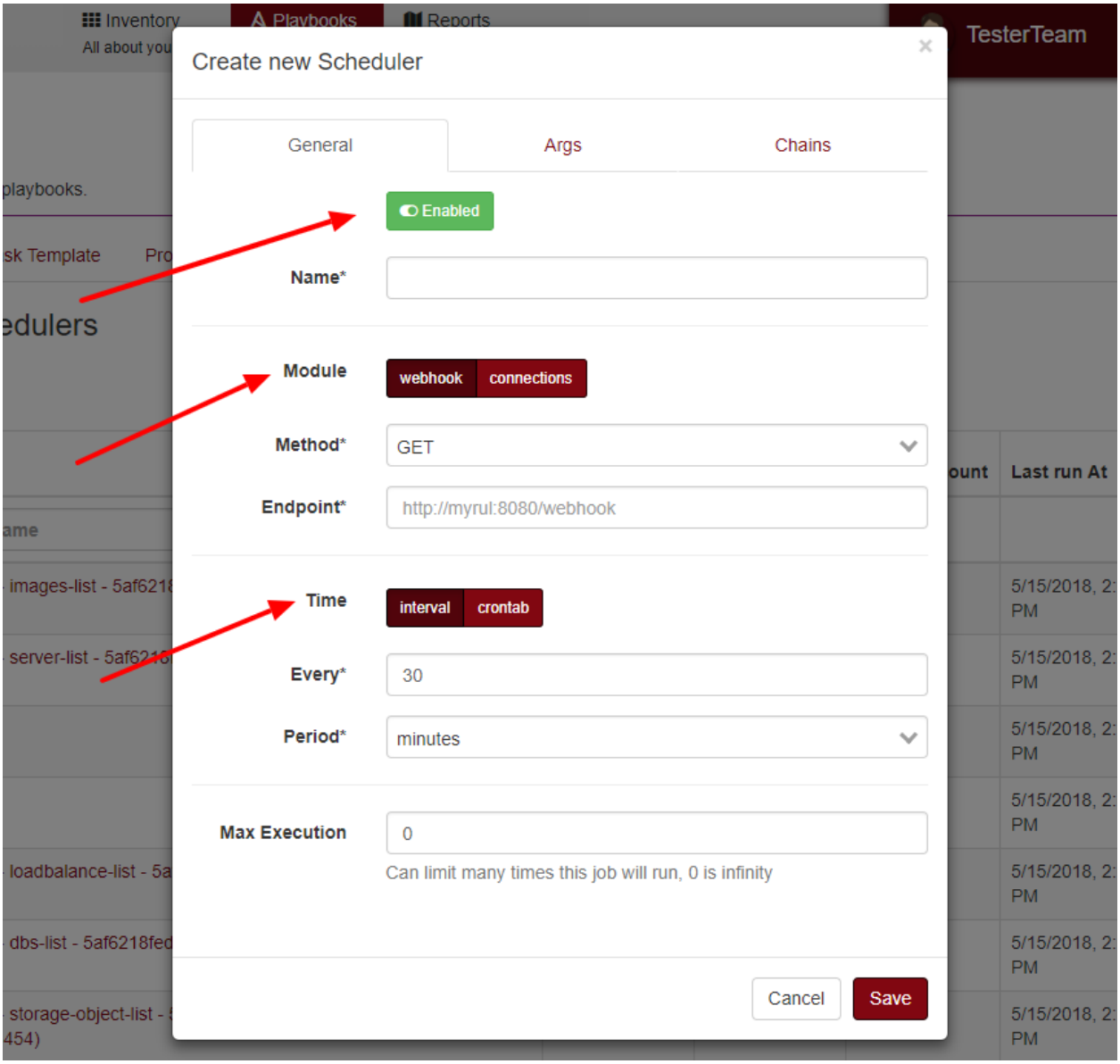
Details for connections schedulers, each time you create a new connection, automatically will create a lot of schedules, each schedule represents resources tracked if you like you can change the time processed of each schedule.



	connections - server-list - 5af6218fedd1b90014ebf291 (1526400982609)	connections	interval	27	5/15/2018, 2:23:09 PM	
	localhost	webhook	interval	229	5/15/2018, 2:24:06 PM	
	discovery	webhook	interval	224	5/15/2018, 2:24:06 PM	
	connections - loadbalance-list - 5af6218fedd1b90014ebf291 (1526403636203)	connections	interval	1	5/15/2018, 2:00:40 PM	
	connections - dbs-list - 5af6218fedd1b90014ebf291 (1526403637338)	connections	interval	1	5/15/2018, 2:00:40 PM	

Total counts

You can create a custom schedule, normally is rest calling.



Creating

4.5 Users and Teams

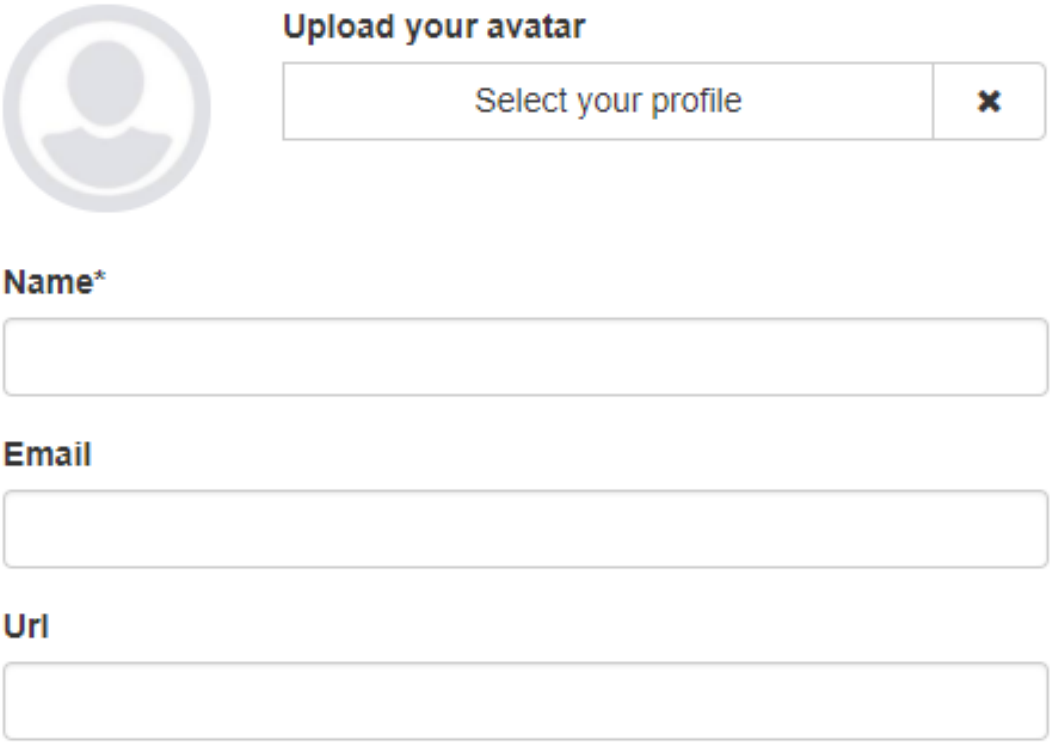
4.5.1 Teams

You can create teams, to do this, click in main menu right corner, and go teams page.

Each team has a name, email, avatar and more important than others is members and permission.

You can add new members and config each access role.

Form team

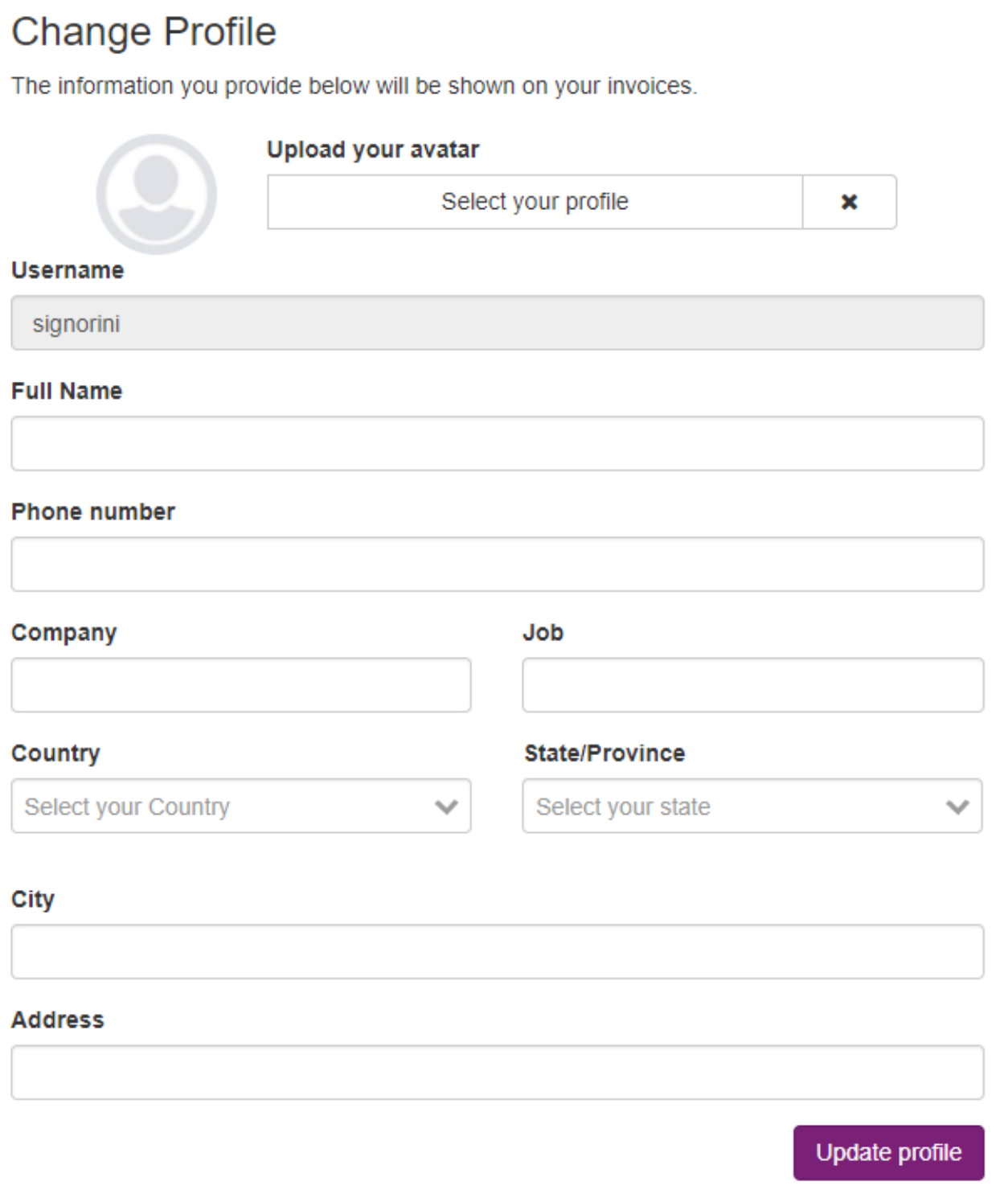


4.5.2 Access and Auth

All entitites in Maestro has access roles, the access role

Account / Profile

On profile you can update your profile.

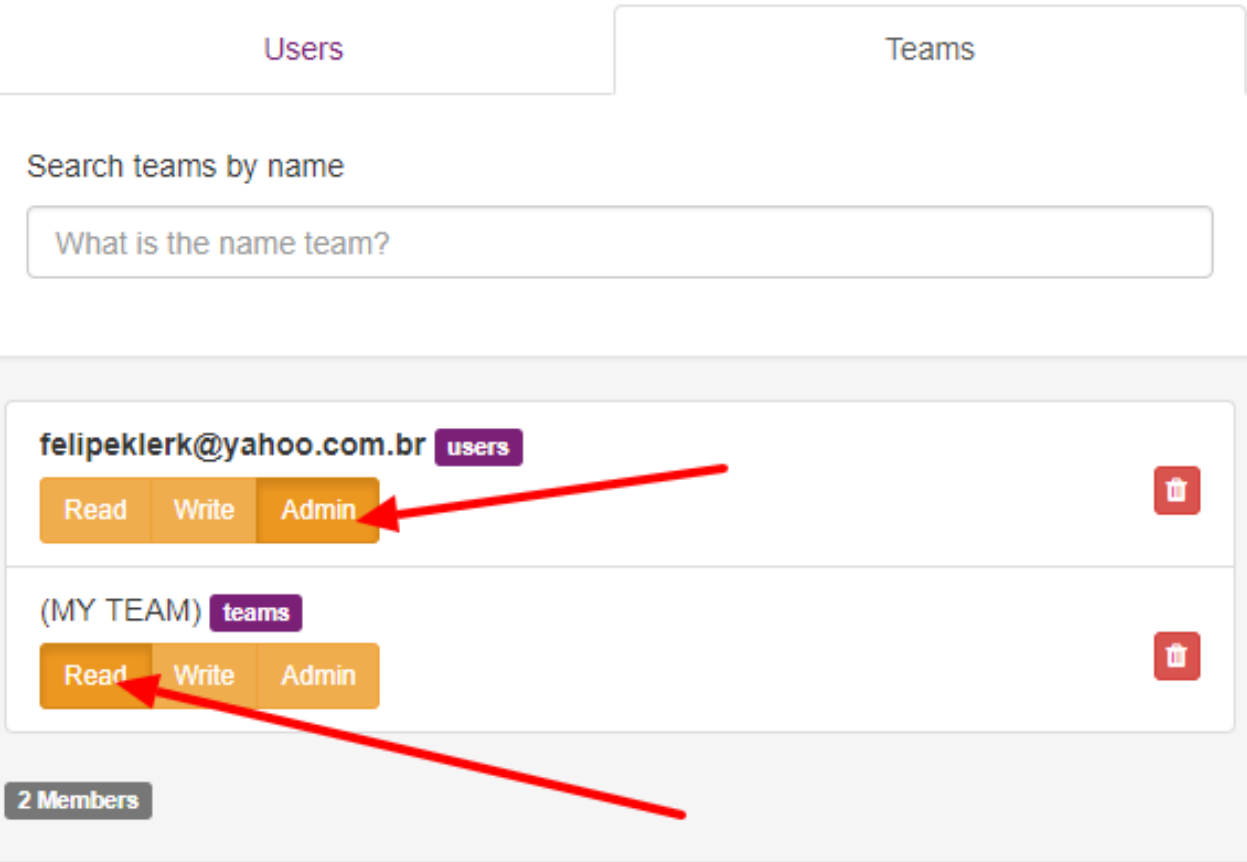


Profile fields

ACL

All entities in Maestro has access roles, the access role, we have, each access role maybe in a team or in user.

Read:	Have only read access
Write:	Have read and write access (update)
Admin:	Able to delete, grant and revoke access.

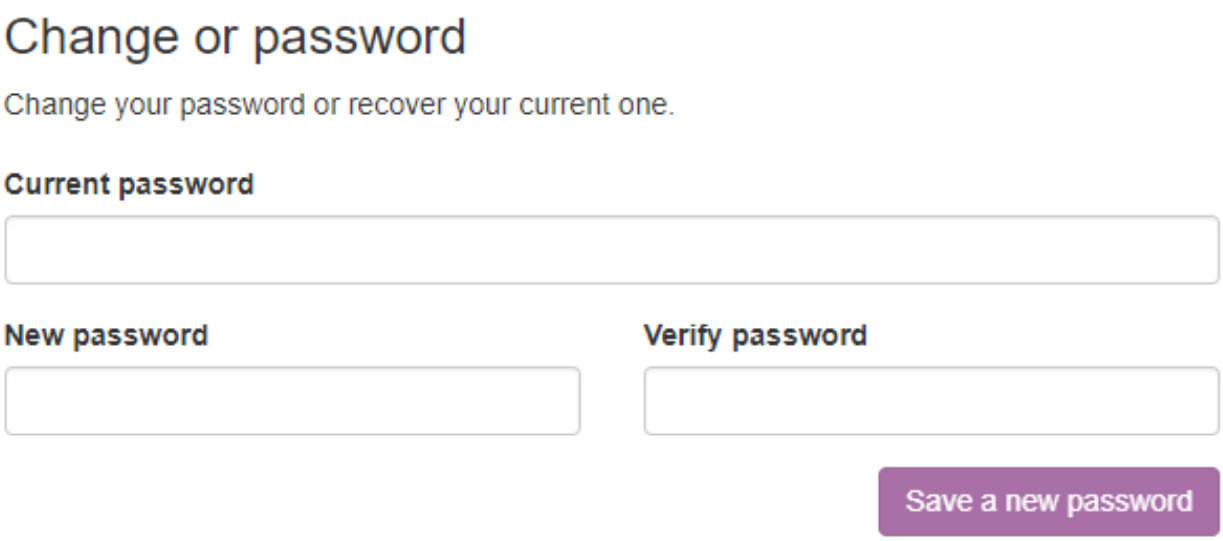


You can change any acl point for specify user or team.

Change password

If you like to change password, you need to go on profile > change password

---


Change or password
Change your password or recover your current one.
Current password
New password
Verify password
Save a new password

New pass

CAPÍTULO 5

Developer Guide

This chapter will explain internal concepts about Maestro, its only matter if you will contribute with code, customize and etc.

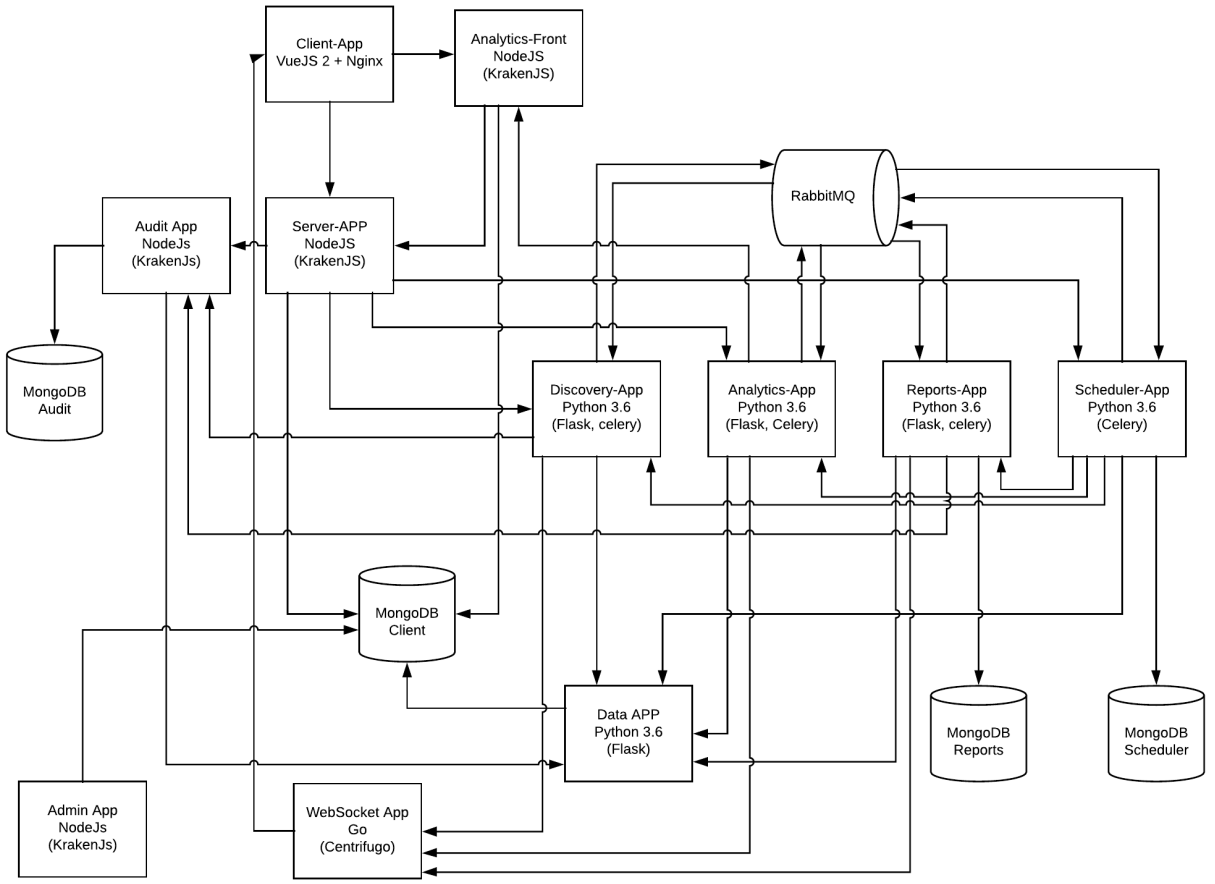
5.1 Architecture

This section show a advanced configurations for each micro service.

Constainerazed system: Made with containers in mind, Maestro Server is deployed with Docker.

Micro service arch: Created with micro services in mind, each service has your responsability.

Services relation, each service communicate by *rest* (http) calls.



5.1.1 FrontEnd - Client App

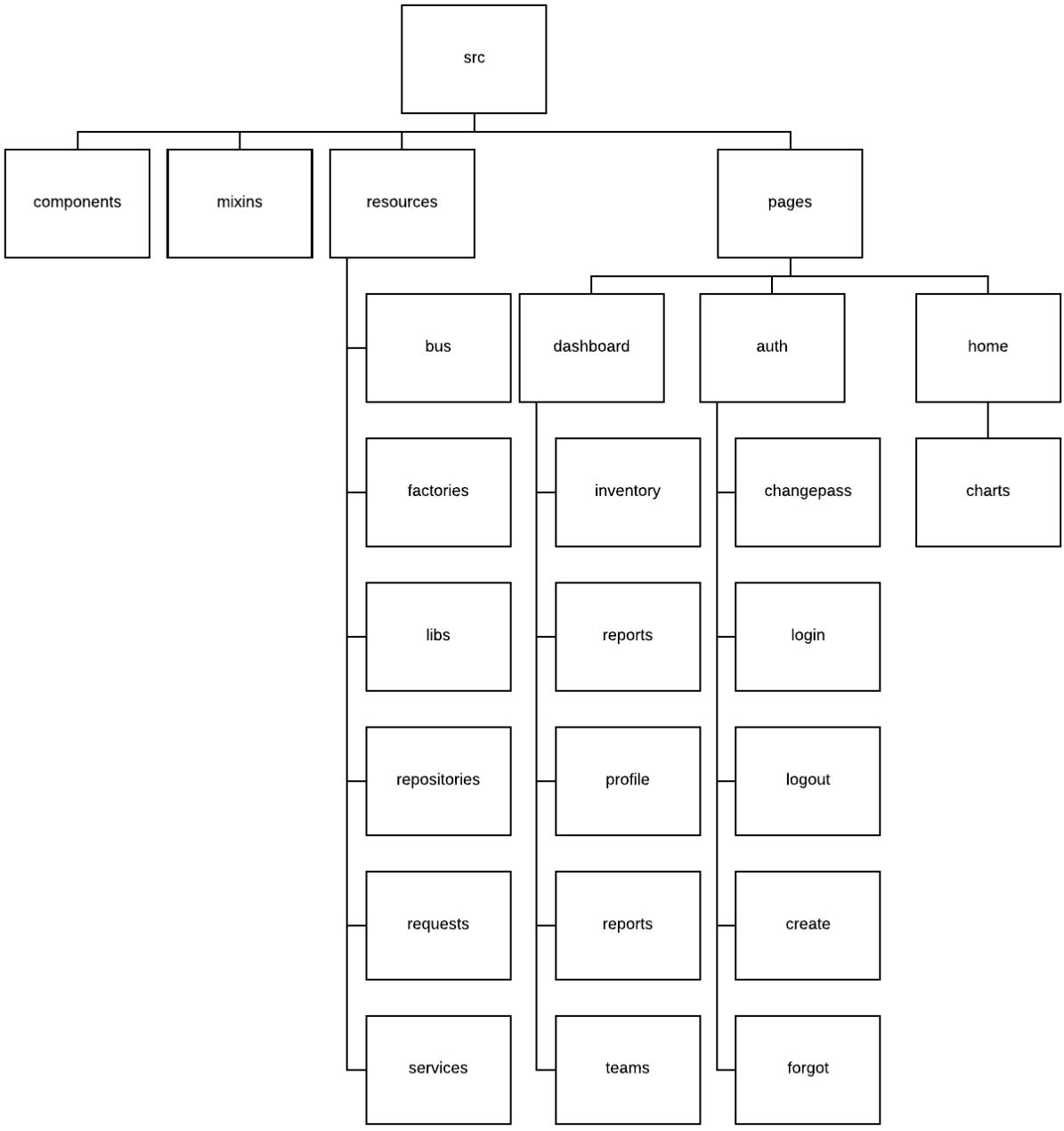
Client App front end application

- Html and Js client application
- Single page app (SPA)
- Cache layer

Aviso: This service needs a proxy reverse like nginx or haproxy.

Vue2	Main framework, using by react and manager views, routes and temaplates, use vue-loader with webpack
Webpack2	bundler generate
Bootue	All micro components, like buttons, tables, forms and etc, its 100% Bootstrap3 components built with Vue2, 100% standalone, no query.
Nginx	Using for proxy reverse
Mocha / Chai / Sinon	Test, asserts and mock library.

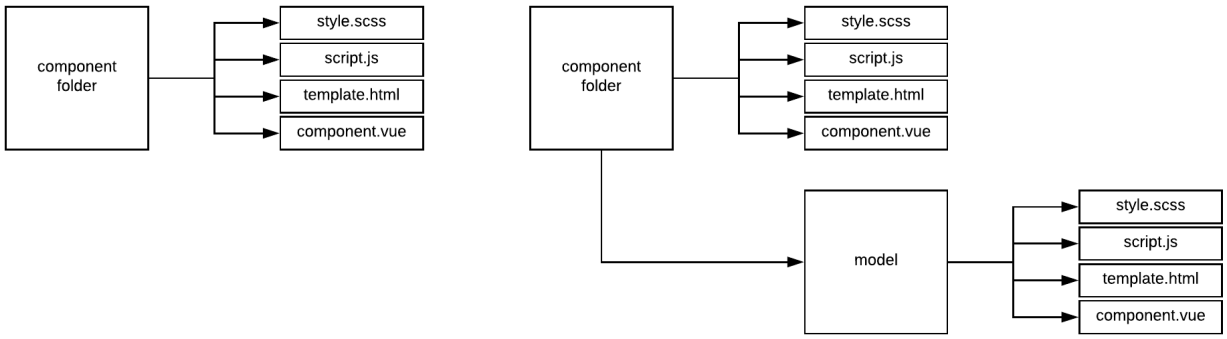
Vue2 Macro Architecture



Important topics

- Front end application is divided in:
  - **src/pages:** templates and bussiness rules (domain layer)
  - **resources:** factories, modals, and cache managers (infrastructure layer)

A single folder structure components normally use:



Installation with node

- Nodejs >= 7.4

Download de repository

```
git clone https://github.com/maestro-server/client-app.git
```

Install dependences

```
npm install
```

Production build

```
npm run build
```

Dev run

```
npm run dev
```

Env variables

Env Variables	Example	Description
API_URL	http://localhost:8888	Server App Url
STATIC_URL	/upload/	Relative path of static content
ANALYTICS_URL	http://localhost:9999	Analytics App Url
LOGO	/static/imgs/logo300.png	Logotype, (login page)
THEME	theme-lotus	Theme (gold|wine|blue|green|dark)

5.1.2 Server App

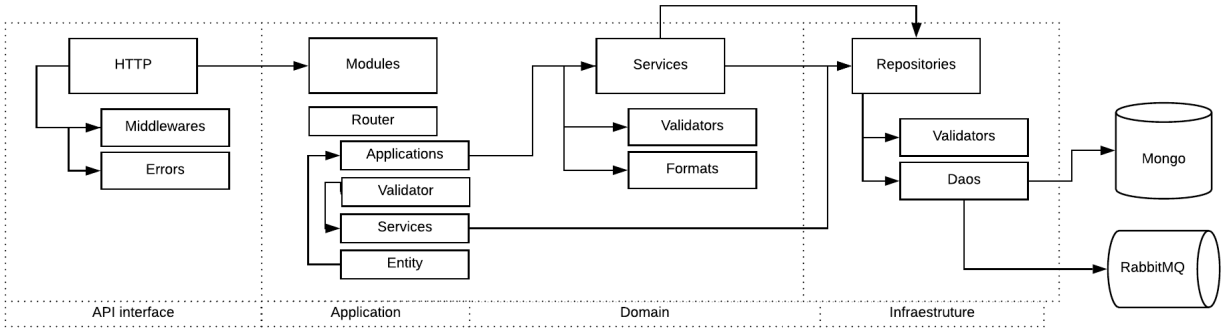
Server App main application, some responsibility is

- Authentication and authorization
- Validate and create entities (crud ops)

- Proxy to others services

Aviso: This service can be external access

We using DDD to organize the code, has infra, repositories, entities (values objects), interfaces, application, and domain, if like to learn read this article is very cool DDD in Node Apps



Server its have constructed with KrakenJs, we create a lot of middleware and organize by domain.

Setup dev env

```
cd devtool/

docker-compose up -d
```

Will be setup mongodb and fake smtp server

Installation with node

- Nodejs 8 or above
- Npm
- MongoDB
- Gcc + python (bcrypt package, if need be compilate)

Download de repository

```
git clone https://github.com/maestro-server/server-app.git
```

Install dependences

```
cd server-app
npm install
```

Configure some env variable

create .env file

```
SMTP_PORT=1025
SMTP_HOST=localhost
SMTP_SENDER='maestro@gmail.com'
SMTP_IGNORE=true

MAESTRO_PORT=8888
MAESTRO_MONGO_URI='localhost'
MAESTRO_MONGO_DATABASE='maestro-client'

MAESTRO_DISCOVERY_URI=http://localhost:5000 // list and get status connection
MAESTRO_REPORT_URI=http://localhost:5005 // create and get reports data
MAESTRO_ANALYTICS_URI=http://analytics:5020 // create analytics report
MAESTRO_ANALYTICS_FRONT_URI=http://analytics_front:9999 // get analytics html
MAESTRO_AUDIT_URI=http://audit:10900 // notify audit update event and get history␣
↪track
```

and

```
npm run server
```

Multiple env

Every config can be pass by env variables, but if you like, can be organize by .env files,

Name	Desc
.env	Default
.env.test	Used on run test
.env.development	node_env is setted development
.env.production	node_env is setted prodcution

Migrate setup data

create .env file

```
npm run migrate
```

For production environment, need to use pm2 or forever lib.

Like (PM2):

```
npm install -g pm2

# Create a file pm2.json

{
"apps": [{
    "name": "server-maestro",
    "script": "./server.js",
    "env": {
    "production": true,
    "PORT": 8888
    }
}]
}
```

```
pm2 start --json pm2.json
```

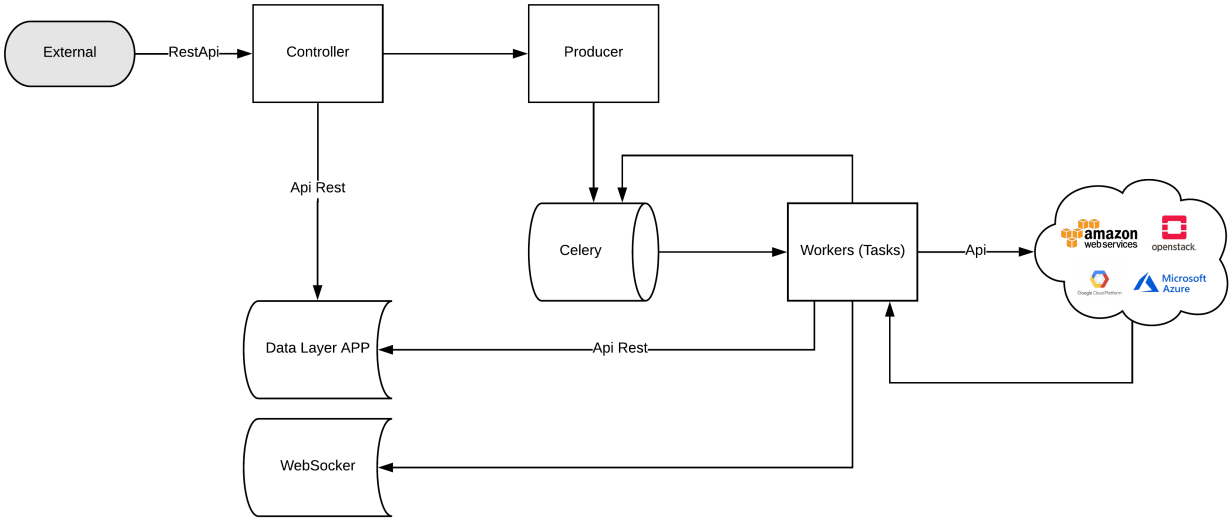
Env variables

Env Variables	Example	Description
MAESTRO_PORT	8888	
NODE_ENV	development|production	
MAESTRO_MONGO_URI	localhost	DB string connection
MAESTRO_MONGO_DATABASE	maestro-client	Database name
MAESTRO_SECRETJWT	XXXX	Secret key - session
MAESTRO_SECRETJWT_FORGOT	XXXX	Secret key - forgot request
MAESTRO_SECRET_CRYPTO_FORGOT	XXXX	Secret key - forgot content
MAESTRO_SECRETJWT_PUBLIC	XXX	Secret key - public shared
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections
MAESTRO_DISCOVERY_URL	http://localhost:5000	Url discovery-app (flask)
MAESTRO_REPORT_URL	http://localhost:5005	Url reports-app (flask)
MAESTRO_ANALYTICS_URI	http://localhost:5020	Url Analytics-app (flask)
MAESTRO_AUDIT_URI	http://localhost:10900	Url Audit-app (krakenjs)
MAESTRO_TIMEOUT	1000	Timeout micro service request
SMTP_PORT	1025	
SMTP_HOST	localhost	
SMTP_SENDER	myemail@XXXX	
SMTP_IGNORE	true|false	
SMTP_USETSL	true|false	
SMTP_USERNAME		
SMTP_PASSWORD		
AWS_ACCESS_KEY_ID	XXXX	
AWS_SECRET_ACCESS_KEY	XXXX	
AWS_DEFAULT_REGION	us-east-1	
AWS_S3_BUCKET_NAME	maestroserver	Bucket name
MAESTRO_UPLOAD_TYPE	S3 or Local	Upload mode
LOCAL_DIR	/public/static/	Where files will be uploaded
PWD	$rootDirectory	PWD process

5.1.3 Discovery App

Discovery App service to connect and crawler provider

- Encharge to manager and authenticate in each provider
- Crawler the data and record into db
- Consume batch insert data



Discovery using Flask, and python >3.5, has api rest, and tasks.

Setup dev env

```
cd devtool/

docker-compose up -d
```

Will be setup rabbitmq and redis

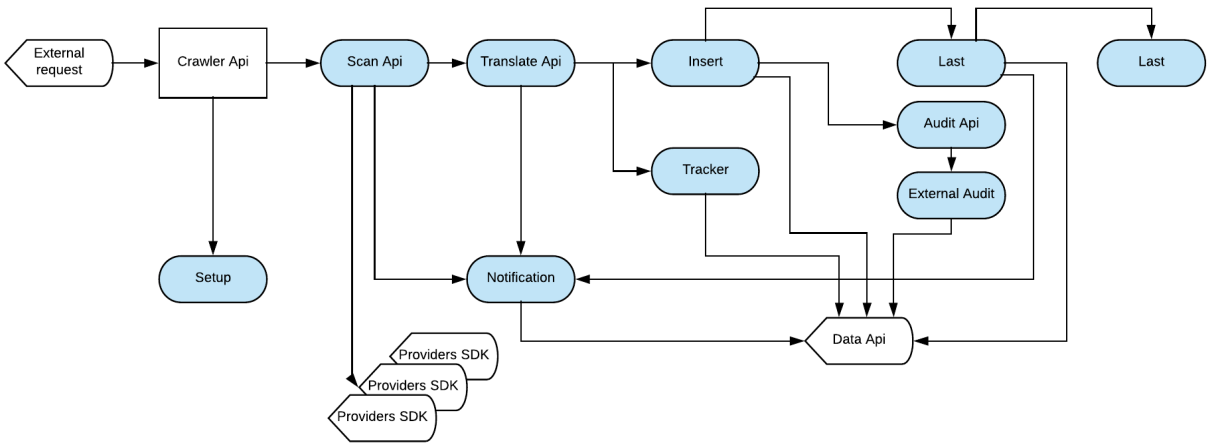
Windows Env

If you use windows, celery havent support for windows, the last version is 3.1.25.

```
pip3 install celery==3.1.25

npm run powershell
```

Important topics



- Controller used factory dc abstract to create easy way to make CRUD in mongodb
- The crawler is divided in:

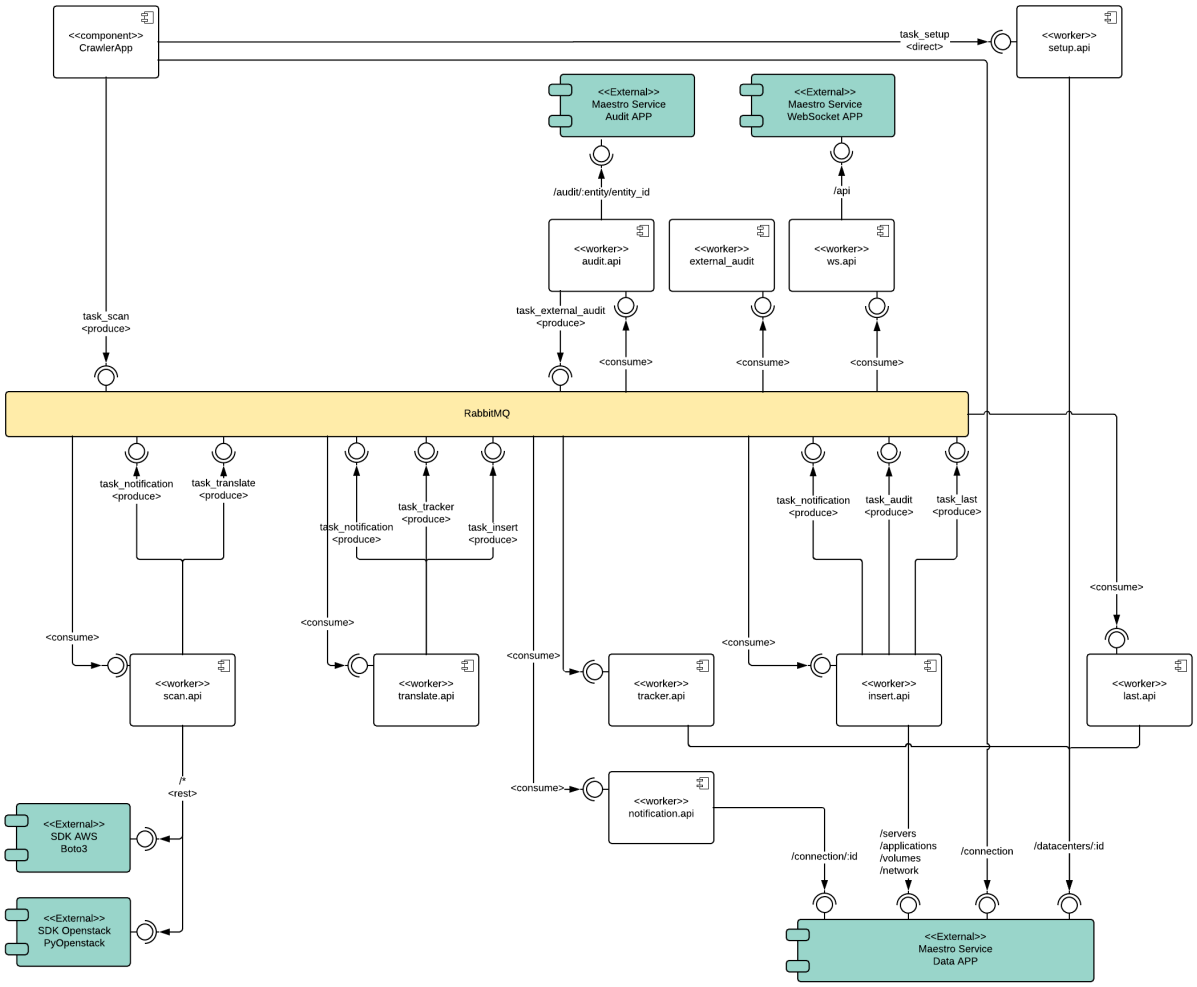
  - **api:** connect in api provider and get result
  - **translate:** normalize the data
  - **setup:** reset tracker stats (used in datacenters to ensure a sync resource)
  - **tracker:** add list entry into tracker stats
    - **insert:** insert/update data in mongodb
  - **audit:** prepare and transform data to be send record to external audit task
  - **external_audit:** Send http request to Audit app
  - **ws:** Send http notification to webscoket api

    Each step have unique task.

- Config is managed by env variables, need to be, because in production env like k8s is easier to manager the pods.
- Repository has pymongo objects.

Component Diagram

Follow the component diagram to show a relation of each worker and service.



Flower - Debbug Celery

You can install a flower, it's a control panel to centralize results throughout rabbitMQ, very useful to troubleshooting producer and consumers.

```
pip install flower

flower -A app.celery

npm run flower
```

Installation with python 3

- Python >3.4
- RabbitMQ

Download de repository

```
git clone https://github.com/maestro-server/discovery-api.git
```

Install dependences

```
pip install -r requeriments.txt
```

Install run api

```
python -m flask run.py

or

FLASK_APP=run.py FLASK_DEBUG=1 flask run

or

npm run server
```

Install run rabbit workers

```
celery -A app.celery worker -E -Q discovery --hostname=discovery@%h --loglevel=info

or

npm run celery
```

Aviso: For production environment, use something like gunicorn.

```
# gunicorn_config.py

import os

bind = "0.0.0.0:" + str(os.environ.get("MAESTRO_PORT", 5000))
workers = os.environ.get("MAESTRO_GWORKERS", 2)
```

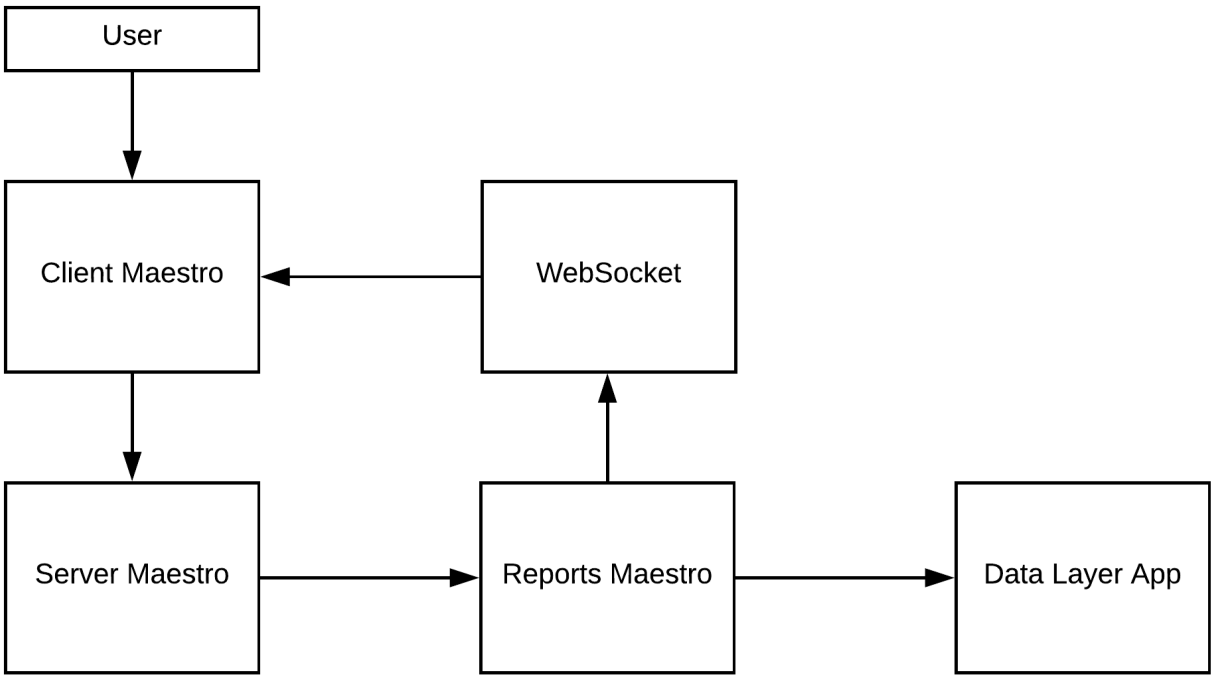
Env variables

Env Variables	Example	Description
MAESTRO_PORT	5000	Port used
MAESTRO_DATA_URI	http://localhost:5010	Data Layer API URL
MAESTRO_AUDIT_URI	http://localhost:10900	Audit App - API URL
MAESTRO_WEBSOCKET_URI	http://localhost:8000	Webosocket App - API URL
MAESTRO_SECRETJWT	XXX	Same that Server App
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections
MAESTRO_WEBSOCKET_SECRET	XXX	Secret Key - JWT Websocket connections
MAESTRO_TRANSLATE_QTD	200	Prefetch translation process
MAESTRO_GWORKERS	2	Gunicorn multi process
CELERY_BROKER_URL	amqp://rabbitmq:5672	RabbitMQ connection
CELERYD_TASK_TIME_LIMIT	10	Timeout workers

5.1.4 Reports App

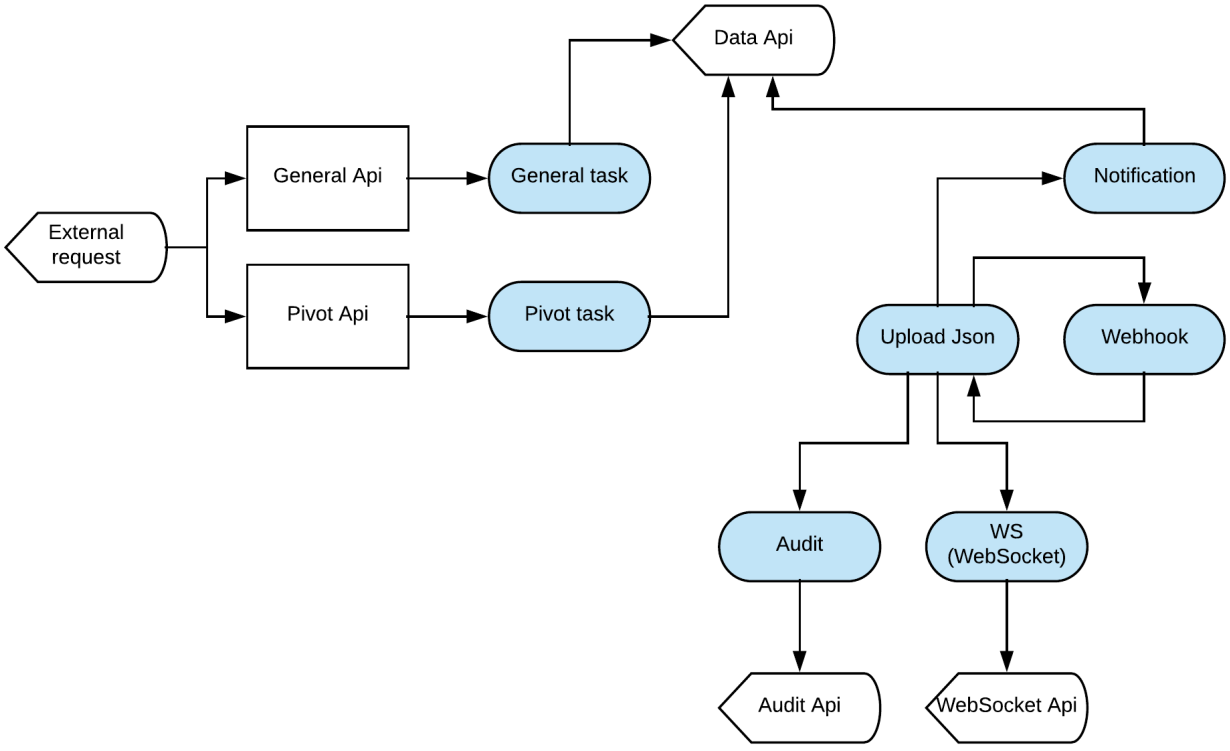
Reports app, generate reports

- Understand complex queries and generate reports
- Manage storage and control each technical flow
- Transform in artifact pdf, csv or json



Reports using Flask, and python >3.6, used Celery Beat feature to call tasks, have strong dependences with discovery app and server app, reports use a standalone MongoDB (only reports app see this db).

Important topics



- Controller used factory task to organize the workflow report generetaion.

- The process is divided
  - **general/pivot:** prepare and select result (communicate with discovery api)
  - **notification:** notificate any message (use discovery app to do)
  - **upload:** control flow data (throttle inserets)
  - **webhook:** insert/update data in mongodb or any endpoint
  - **aggregation** - Execute aggregation tasks and save in report collections
  - **notify** - Notify status task to websocket and audit services

---

Installation with python 3

- Python >3.4
- RabbitMQ
- MongoDB

Download de repository

```
git clone https://github.com/maestro-server/report-app.git
```

Install run api

```
python -m flask run.py --port 5005

or

FLASK_APP=run.py FLASK_DEBUG=1 flask run --port 5005

or

npm run server
```

Install run rabbit workers

```
celery -A app.celery worker -E -Q report --hostname=report@%h --loglevel=info

or

npm run celery
```

Aviso: For production environment, use something like gunicorn.

```
# gunicorn_config.py

import os

bind = "0.0.0.0:" + str(os.environ.get("MAESTRO_PORT", 5005))
workers = os.environ.get("MAESTRO_GWORKERS", 2)
```

Env variables

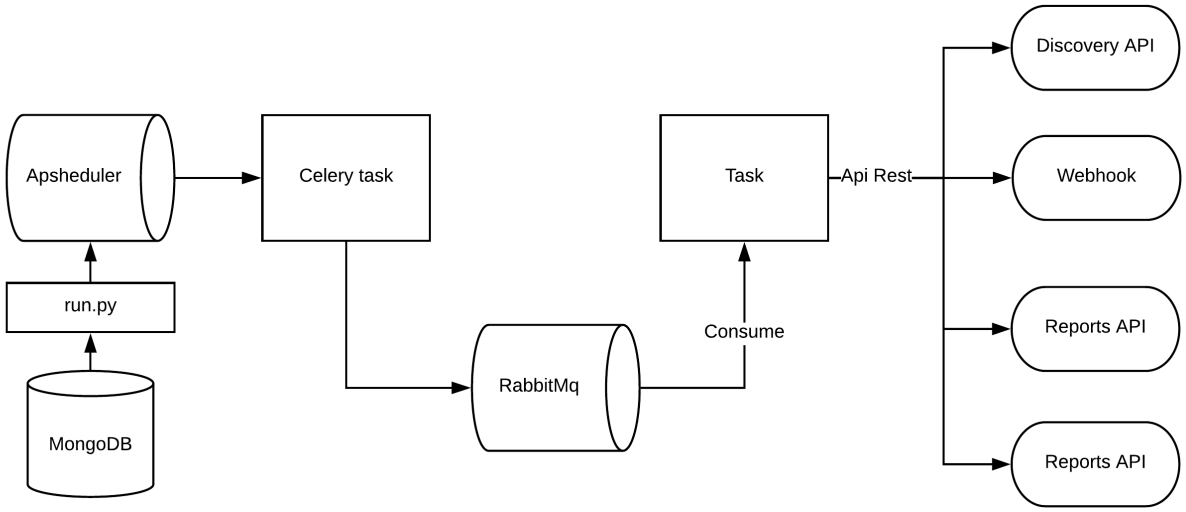
Env Variables	Example	Description
MAESTRO_PORT	5005	Port used
MAESTRO_MONGO_URI	localhost	Mongo Url conn
MAESTRO_MONGO_DATABASE	maestro-reports	Db name, its differente of servers-app
MAESTRO_DATA_URI	http://localhost:5010	Data layer api
MAESTRO_REPORT_URI	http://localhost:5005	Report api
MAESTRO_AUDIT_URI	http://localhost:10900	Audit App - API URL
MAESTRO_WEBSOCKET_URI	http://localhost:8000	Webosocket App - API URL
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections
MAESTRO_WEBSOCKET_SECRET	XXX	Secret Key - JWT Websocket connections
MAESTRO_REPORT_RESULT_QTD	1500	Limit default
MAESTRO_INSERT_QTD	20	Prefetch data insert
MAESTRO_GWORKERS	2	Gworkers thread pool
CELERY_BROKER_URL	amqp://rabbitmq:5672	RabbitMQ connection

5.1.5 Scheduler App

Scheduler App service to manage and execute jobs

- Schedule jobs, interval or crontab
- Requests chain jobs
- **Modules**
  - Webhook: Call URL request
  - Connections: Call Crawler task

Scheduler use apscheduler to control scheduler jobs, Apscheduler documentation



Installation with python 3

- Python >3.4
- RabbitMQ
- MongoDB

Download de repository

```
git clone https://github.com/maestro-server/scheduler-app.git
```

Important topics

- Celery Beat consult schedulers collection in mongodb every 5 seconds and updated time to call the tasks.
- Have 2 tasks called by beat
  - **webhook:** Call HTTP request accordly arguments.
  - **connection:** Consulting connection data, after call webhook.
- Have support tasks called by outhers tasks.
  - **chain and chain_exec:** Called by webhook, this create another job after the first finish.
  - **depleted_job:** If any job recevied something wrong, this taks is called e depleted that job.
  - **notify_event:** Send notification event.

Installation with python 3

- Python >3.4
- RabbitMQ
- MongoDB

Download de repository

```
git clone https://github.com/maestro-server/scheduler-app.git
```

Install run celery beat

```
celery -A app.celery beat -S app.schedulers.MongoScheduler --loglevel=info

or

npm run beat
```

Install run rabbit workers

```
celery -A app.celery worker -E --hostname=scheduler@%h --loglevel=info

or

npm run celery
```

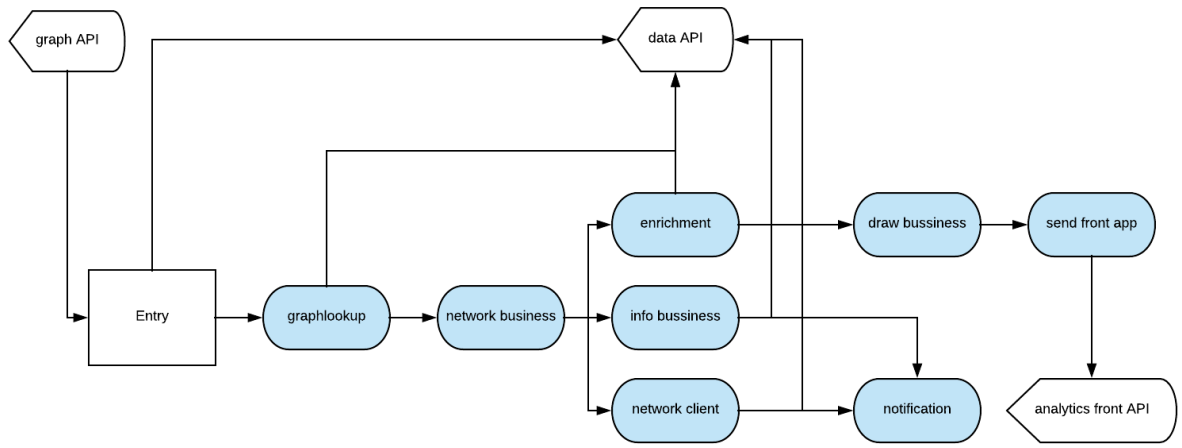
Env variables

Env Variables	Example	Description
MAESTRO_DATA_URI	http://localhost:5010	Data Layer API URL
MAESTRO_DISCOVERY_URI	http://localhost:5000	Discovery App URL
MAESTRO_ANALYTICS_URI	http://localhost:5020	Analytics App URL
MAESTRO_REPORT_URI	http://localhost:5005	Reports App URL
MAESTRO_MONGO_URI	localhost	MongoDB URI
MAESTRO_MONGO_DATABASE	maestro-client	Mongo Database name
CELERY_BROKER_URL	amqp://rabbitmq:5672	RabbitMQ connection
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections

5.1.6 Analytics Maestro

Analytics App is a module to analytics graph and create xml of Maestro Server, yours responsibility is:

- Create grids
- Create bussiness graph
- Create network graph
- Create infra graph
- Drawing
- SVGs



Analytics using Flask, and python >3.5, has api rest, and tasks.

Setup dev env

```
cd devtool/

docker-compose up -d
```

Will be setup rabbitmq and redis

Windows Env

If you use windows, celery havent support for windows, the last version is 3.1.25.

```
pip3 install celery==3.1.25

npm run powershell
```

Important topics

- Controller used only graph to start all tasks:
- The drawer process is compound by:
  - **entry:** First task, figure out all entry applications accordingly system endpoint parameters, our any direct application if avalaible.
  - **graphlookup:** Request for Data App a aggregate query using MongoDB $graphLookup.
  - **network bussiness:** Construct Grid Map, and send to enrichment and info bussines.
  - **enrichment:** Request for Data App all servers used on grid.
  - **info bussiness:** Calculate histogram, counts, density and connections.
  - **network client:** Request for Data App all clients used in grid.
  - **draw bussiness:** Create svgs based of grid.
  - **notification:** Send updates for Data App.
  - **send front app:** Send svgs to Analytics Front app.

    Each step have unique task.
- Config is managed by env variables, need to be, because in production env like k8s is easier to manager the pods.
- Repository has pymongo objects.

Flower - Debbug Celery

You can install a flower, it's a control panel to centralize results throughout rabbitMQ, very useful to troubleshooting producer and consumers.

```
pip install flower

flower -A app.celery

npm run flower
```

Installation with python 3

- Python >3.4
- RabbitMQ

Download de repository

```
git clone https://github.com/maestro-server/discovery-api.git
```

Install dependences

```
pip install -r requeriments.txt
```

Install run api

```
python -m flask run.py

or

FLASK_APP=run.py FLASK_DEBUG=1 flask run

or

npm run server
```

Install run rabbit workers

```
celery -A app.celery worker -E -Q analytics --loglevel=info

or

npm run celery
```

Aviso: For production environment, use something like gunicorn.

```
# gunicorn_config.py

import os

bind = "0.0.0.0:" + str(os.environ.get("MAESTRO_PORT", 5020))
workers = os.environ.get("MAESTRO_GWORKERS", 2)
```

Env variables

Env Variables	Example	Description
MAESTRO_PORT	5020	Port
MAESTRO_DATA_URI	http://localhost:5010	Data Layer API URL
MAESTRO_ANALYTICS_FRONT_URI	http://localhost:9999	Analytics Front URL
MAESTRO_WEBSOCKET_URI	http://localhost:8000	Webosocket App - API URL
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections
MAESTRO_WEBSOCKET_SECRET	XXX	Secret Key - JWT Websocket connections
MAESTRO_GWORKERS	2	Gunicorn multi process
CELERY_BROKER_URL	amqp://rabbitmq:5672	RabbitMQ connection
CELERYD_TASK_TIME_LIMIT	10	Timeout workers

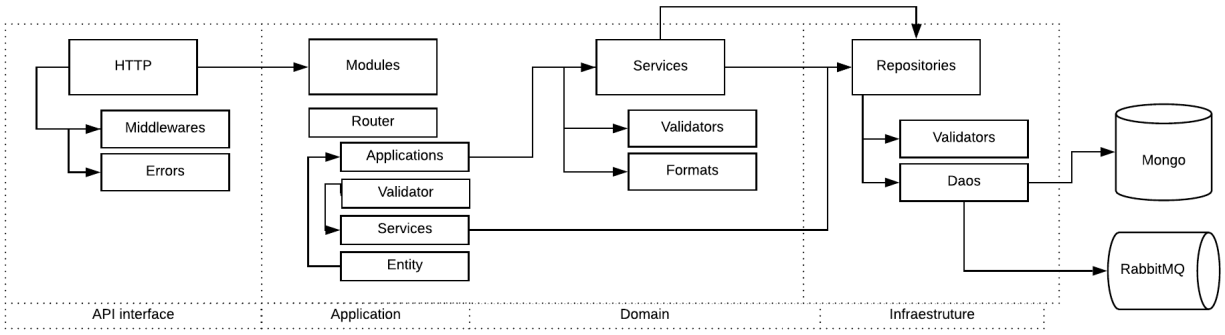
5.1.7 Analytics Front

Analytics Front App is front end of analytics graph of Maestro Server, yours responsibility is:

- Authentication
- Show graphs SVGs
- Upload internal SVGs of analytics

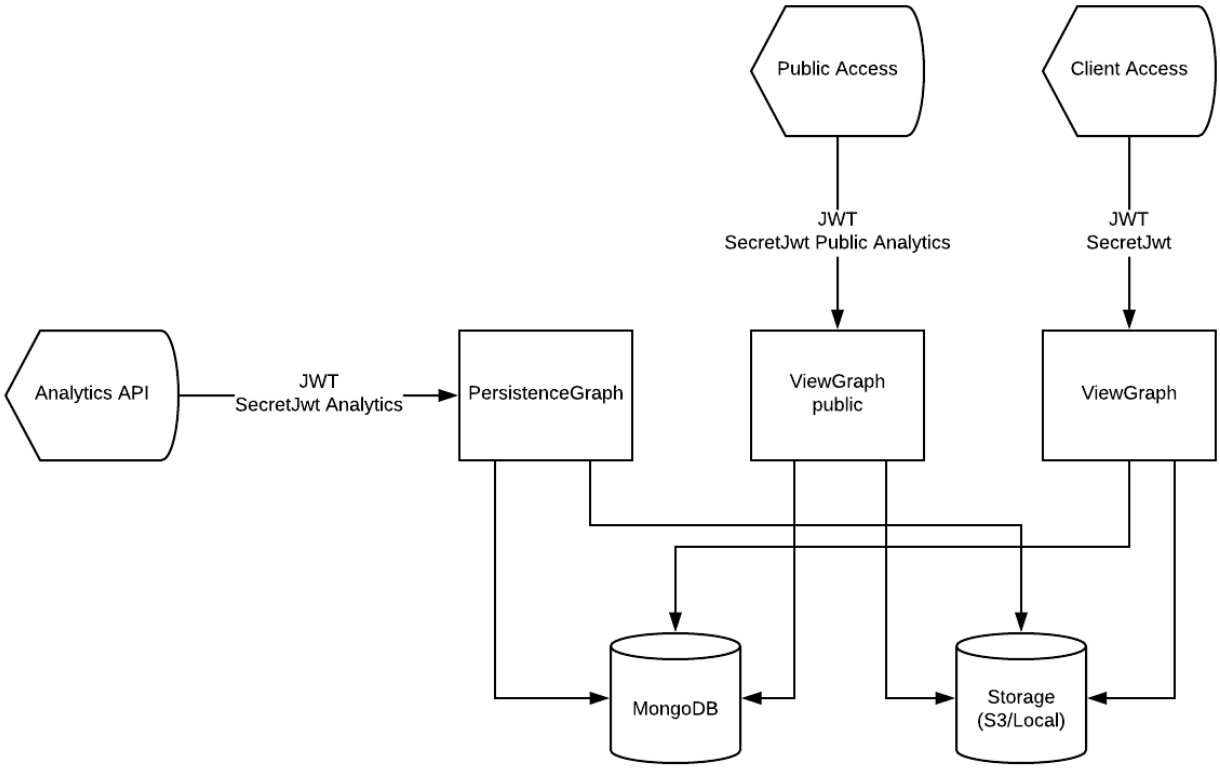
Aviso: This service can be external access

We using DDD to organize the code, has infra, repositories, entities (values objects), interfaces, application, and domain, if like to learn read this article is very cool DDD in Node Apps



Analytics its have constructed with KrakenJs, we create a lot of middleware and organize by domain.

Core API, organized by modules:



- Core
- Authetication
- Graph
- View

Installation with node

- Nodejs 8 or above
- MongoDB 3.4
- RabbitMQ
- AWS S3 (If using S3 upload)

Download de repository

```
git clone https://github.com/maestro-server/analytics-front.git
```

Install dependences

```
cd analytics-front
npm install
```

Configure some env variable

create .env file

```
MAESTRO_PORT=9999
MAESTRO_MONGO_URI='localhost'
MAESTRO_MONGO_DATABASE='maestro-client'
```

and

```
npm run server
```

Multiple env

Every config can be pass by env variables, but if you like, can be organize by .env files,

Name	Desc
.env	Default
.env.test	Used on run test
.env.development	node_env is setted development
.env.production	node_env is setted prodcution

Migrate setup data

create .env file

```
npm run migrate
```

For production environment, need to use pm2 or forever lib.

Like (PM2):

```
npm install -g pm2

# Create a file pm2.json

{
"apps": [{
    "name": "analytics-front",
    "script": "./server.js",
    "env": {
        "production": true,
        "NODE_ENV": "production",
        "PORT": 9999
    }
}]
}
```

```
pm2 start --json pm2.json
```

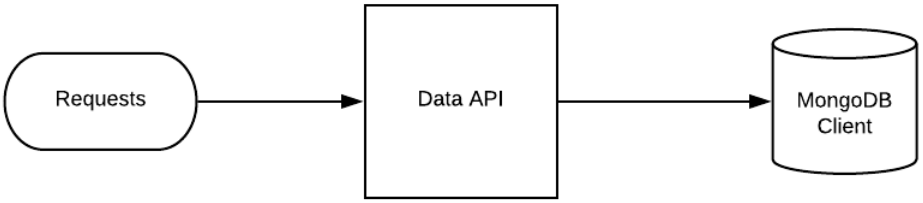
Env variables

Env Variables	Example	Description
MAESTRO_PORT	9999	
API_URL	http://localhost:8888	Server app Url
NODE_ENV	development|production	
MAESTRO_MONGO_URI	localhost	DB string connection
MAESTRO_MONGO_DATABASE	maestro-client	Database name
MAESTRO_SECRETJWT	XXXX	Secret key - server app
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections
MAESTRO_SECRETJWT_PUBLIC	XXXX	Secret key - same server app
AWS_ACCESS_KEY_ID	XXXX	
AWS_SECRET_ACCESS_KEY	XXXX	
AWS_DEFAULT_REGION	us-east-1	
AWS_S3_BUCKET_NAME	maestroserver	
MAESTRO_UPLOAD_TYPE	S3/Local	Upload mode
LOCAL_DIR	/public/static/	Where files will be uploaded
PWD	$rootDirectory	PWD process

5.1.8 Data APP

Data app, database gateway micro service - Request and response database operations

Simple Rest API using Flask (python) + pymongo.



Setup dev env

```
pip install

    FLASK_APP=run.py FLASK_DEBUG=1 flask run --port=5010

    or

    npm run server
```

Mongo service

```
cd devtool/

docker-compose up -d
```

Will be setup mongodb

Installation with python 3

- Python >3.4
- MongoDB

Download de repository

```
git clone https://github.com/maestro-server/data-app.git
```

Install run api

```
python -m flask run.py --port 5010

or

FLASK_APP=run.py FLASK_DEBUG=1 flask run --port 5010

or

npm run server
```

Aviso: For production environment, use something like gunicorn.

```
# gunicorn_config.py

import os

bind = "0.0.0.0:" + str(os.environ.get("MAESTRO_PORT", 5010))
workers = os.environ.get("MAESTRO_GWORKERS", 2)
```

Env variables

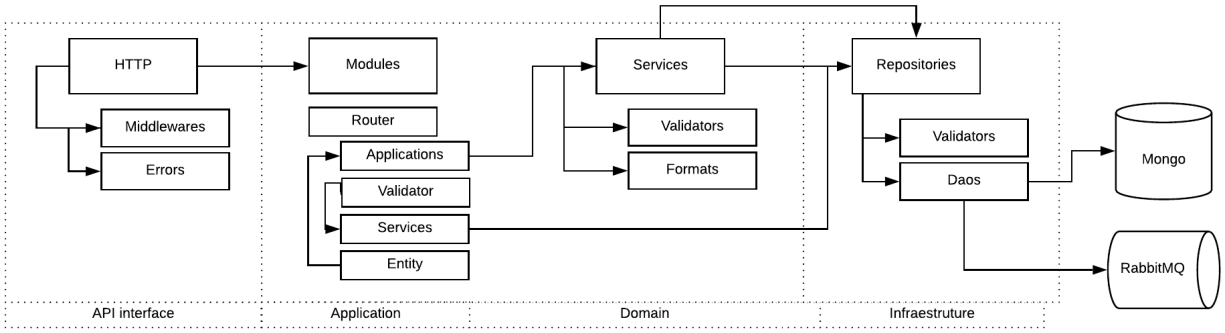
Env Variables	Example	Description
MAESTRO_PORT	5010	Port used
MAESTRO_MONGO_URI	localhost	Mongo Url conn
MAESTRO_MONGO_DATABASE	maestro-client	Db name, its differente of servers-app
MAESTRO_GWORKERS	2	Gunicorn multi process
MAESTRO_INSERT_QTD	200	Throughput insert in reports collection
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections

5.1.9 Audit App

Audit App is webapp application port of Maestro Server stack, yours responsibility is:

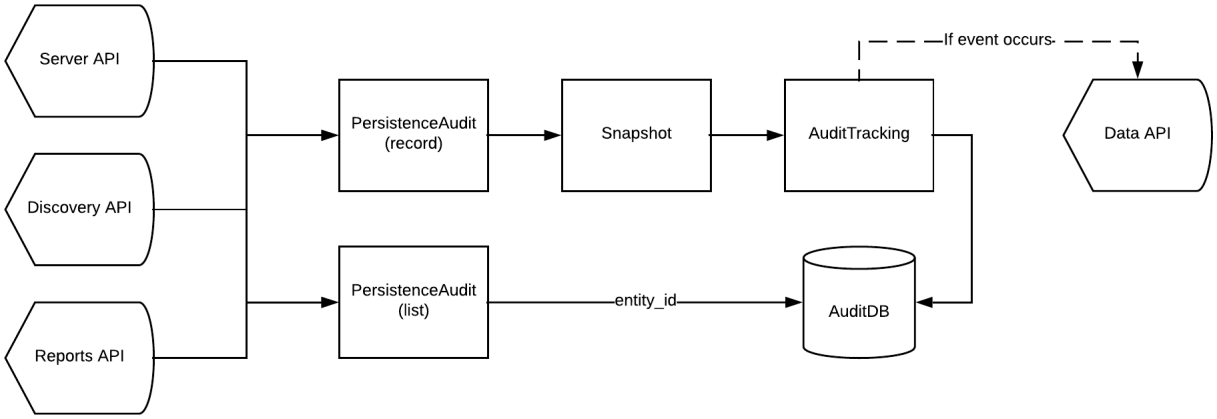
- Control and manage changed data on Maestro
- Show a tree of changed entities
- Store and control data changes
- Trigger hooks based in changed rules

We using DDD to organize the code, has infra, repositories, entities (values objects), interfaces, application, and domain, if like to learn read this article is very cool DDD in Node Apps



Audit its have constructed with KrakenJs, we create a lot of middleware and organize by domain.

Core API, organized by modules:



- Core
- Audit // make diff of each request
- Snapshot // hold last state of each entity

Installation with node

- Nodejs 8 or above
- MongoDB 3.x

Download de repository

```
git clone https://github.com/maestro-server/audit-app.git
```

Install dependences

```
cd audit-app
npm install
```

Configure some env variable

create .env file

```
MAESTRO_PORT=10900
MAESTRO_MONGO_URI='localhost'
MAESTRO_MONGO_DATABASE='maestro-audit'
MAESTRO_DATA_URI="localhost:5005"
```

and

```
npm run server
```

Multiple env

Every config can be pass by env variables, but if you like, can be organize by .env files,

Name	Desc
.env	Default
.env.test	Used on run test
.env.development	node_env is setted development
.env.production	node_env is setted prodcution

Migrate setup data

create .env file

```
npm run migrate
```

For production environment, need to use pm2 or forever lib.

Like (PM2):

```
npm install -g pm2

# Create a file pm2.json

{
"apps": [{
    "name": "audit-app",
    "script": "./server.js",
    "env": {
```

(continues on next page)

(continuação da página anterior)

```
        "production": true,
        "NODE_ENV": "production",
        "PORT": 10900
    }
}]
}
```

```
pm2 start --json pm2.json
```

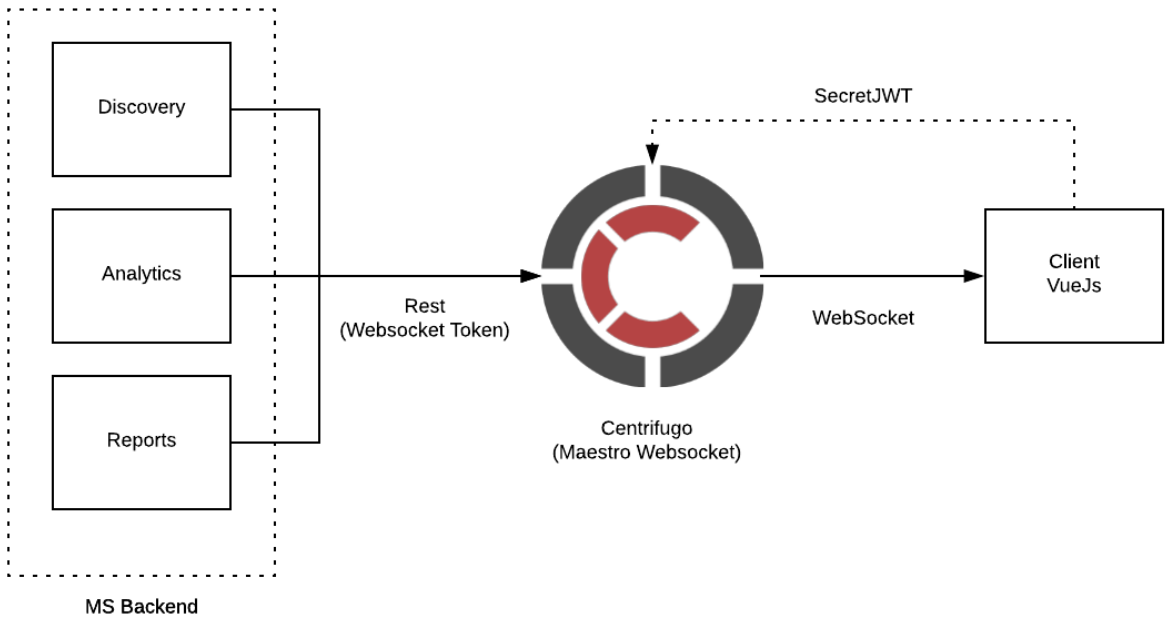
Env variables

Env Variables	Example	Description
MAESTRO_PORT	10900	
NODE_ENV	development|production	
MAESTRO_MONGO_URI	localhost	DB string connection
MAESTRO_MONGO_DATABASE	maestro-audit	Database name
MAESTRO_TIMEOUT	1000	Timeout any http private request
MAESTRO_DATA_URI	http://localhost:5010	Data App - API URL
MAESTRO_SECRETJWT_PRIVATE	XXX	Secret Key - JWT private connections
MAESTRO_NOAUTH	XXX	Secret Pass to validate private connections

5.1.10 WebSocket APP

It's websocket server with restfull hooks, maestro websocket use centrifugo project. - Client notification using websockets

Websocket system using Centrifugo OpenSource project (Centrifugo OpenSource project).



Setup dev env

```
# Generate config
docker run maestro-websocket centrifugo genconfig

# Run websocket
docker run -e MAESTRO_WEBSOCKET_SECRET='secret' -e MAESTRO_SECRETJWT='jwttoken'␣
↪maestroserver/websocket-maestro

# Run centrifugo with admin enabled
docker run -e CENTRIFUGO_ADMIN='pass' -e CENTRIFUGO_ADMIN_SECRET='jwttoken'␣
↪maestroserver/websocket-maestro
```

Download de repository (Centrifugal project)

```
git clone https://github.com/centrifugal/centrifugo
```

Endpoints

Client access

```
var centrifuge = new Centrifuge('ws://{server}/connection/websocket');

centrifuge.subscribe("news", function(message) {
    console.log(message);
});

centrifuge.connect();
```

Backend access

```
import json
import requests

command = {
    "method": "publish",
    "params": {
        "channel": "maestro#${ID-USER}",
        "data": {
            "notify": { // call notify
                "title": "<string>",
                "msg": "<string>",
                "type": "danger|warning|info|success"
            },
            "event": {
                "caller": "<string>" //custom event on client
            }
        }
    }
}
```

Env variables

Env Variables	Example	Description
MAESTRO_WEBSOCKET_SECRET	backSecretToken	Token to authenticate backends apps
MAESTRO_SECRETJWT	frontSecretToken	Token to autheticate front end users
CENTRIFUGO_ADMIN	adminPassword	Admin password
CENTRIFUGO_ADMIN_SECRET	adminSecretToken	Token to autheticate administrator users

5.2 APIs

All commands, jobs, tasks or action is made by rest api, you can use se same api to integrate with your own applications.

5.2.1 Server API

See docs server api.

5.2.2 Discovery API

See docs discovery api.

5.2.3 Report API

See docs report api.

5.2.4 Analytics API

See docs analytics api.

5.2.5 Data API

See docs data api.

5.2.6 Analytics Front API

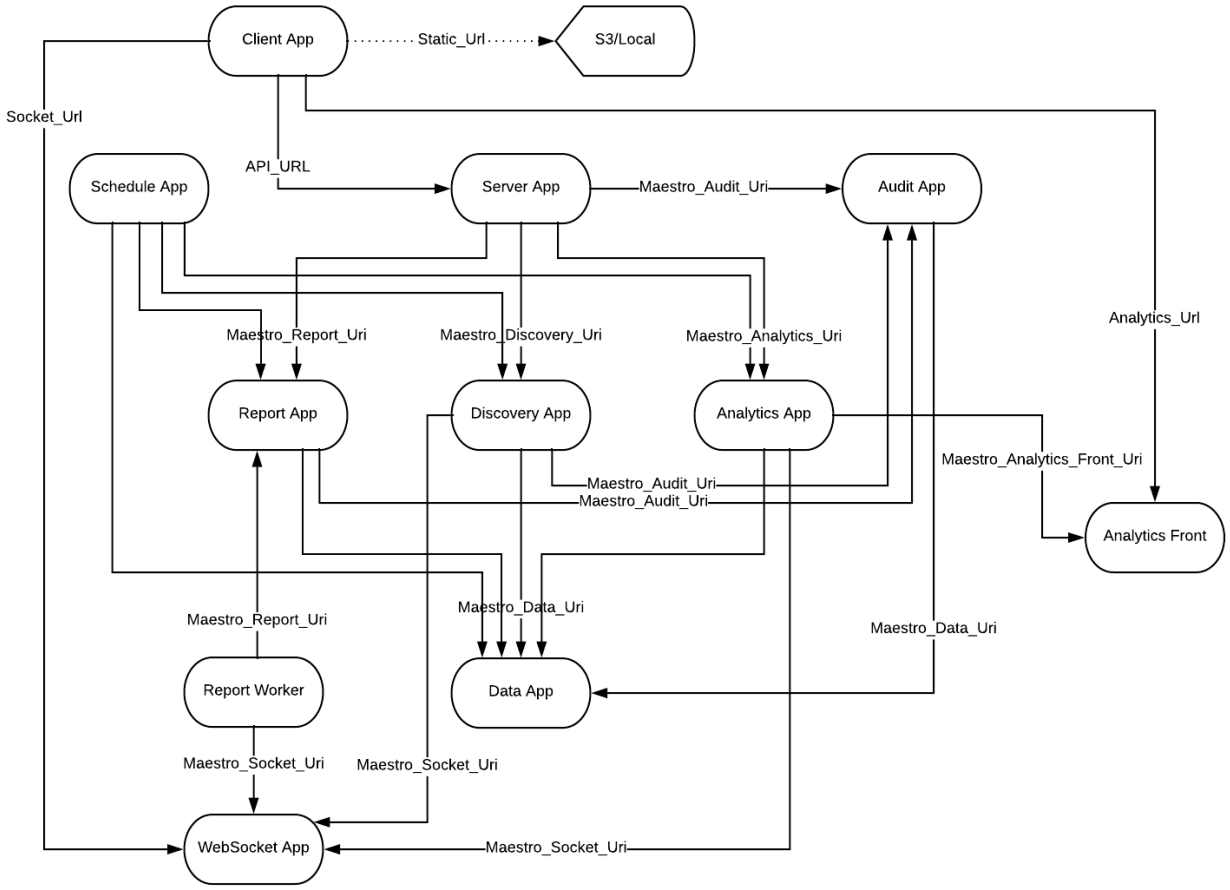
See docs analytics front api.

5.2.7 Audit API

See docs audit api.

5.3 Service Dependecy tree

Architecture map among service dependencies and environment variable used to config this discovery on each service.

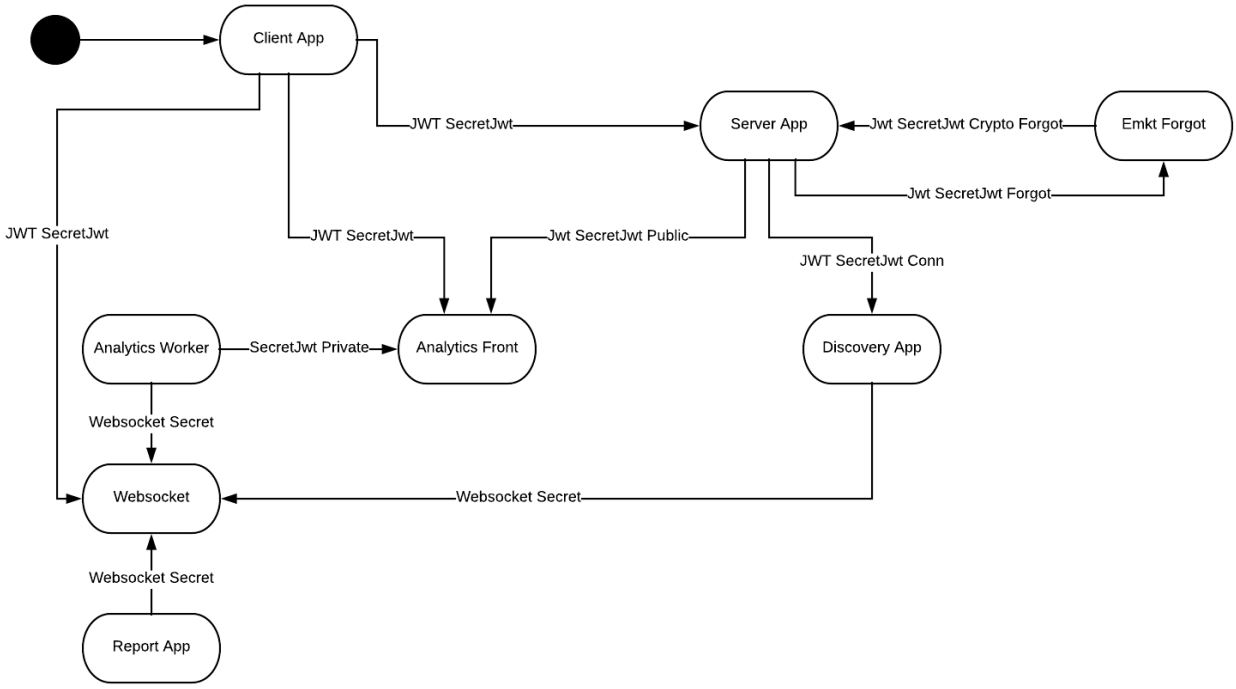


Service	Need to see	Context	Protocol
Client App	Server App	SPA application	Rest
	WebSocket App	Received status message (service bus)	WebSocket
	Analytics Front	Show graphs on bussiness analytics	Iframe HTTP
Server App	Report App	Create any reports	Rest
	Discovery App	Execute crawler actions	Rest
	Analytics App	Create bussiness graphs	Rest
	Audit App	Send any update to audit	Rest
Report App	Data App	Update report status	Rest
	Audit App	Send any update to audit	Rest
	WebSocket App	Send to client any status	WebSocket
Discovery App	Data App		Rest
	Audit App	Send any update to audit	Rest
	WebSocket App		WebSocket
Analytics App	Data App	Populate meta data in analytics entity	Rest
	Analytics Front	Post svgs	Rest
	WebSocket App	Send to client any status	Socket
Scheduler App	Report App	Automated and manage reports	Rest
	Discovery App	Automated and manage discovery	Rest
	Analytics App	Automated and manage analçytics	Rest
	Data App	Dump connections parameters.	Rest
Audit App	Data App	Update any sync rule	Rest

5.4 JWT Tokens

Some endpoint needs to be authenticated, Maestro use JWT token to authenticate a user and two or more systems, each system has own secret token shared between concerned services. Example, WebSocket only accept a request if another service uses a specific jwt (maestro_secretjwt_socket).

Follow an architecture of switch tokens



JWT Name	Context	Owned by	Used by	
SecreteJwt	Autheticate user	Server App	Client App	Jwt user auth
			Discovery App	Command to crawler 3 party provider
			Analytics Front	Jwt user auth
			WebSocket	Hashtable message bus received
SecretJwt Public	Auth shared links (public access)	Server App	Analytics Front	Used to create token to allowed public access on graphs
SecretJwt Analytics	Auth along analytics apps	Analytics App (Worker)	Analytics Front	Security key to allowed to post on analytics front
SecretJwt Crpto Forgot	First secret key, request forgot password	Server App	Client App	
SecretJwt Forgot	Second secret key, confirm forgot password	Server App	Server App	
SecretJwt Socket	Auth along websockets apps	Websocket App	Analytics App	Security key to allowed to post on websocket message bus
			Discovery App	

JWT Name	Context	Owned by	Used by	
SecretJwt Private	Private Authenticate	Server	Analytics App	Security key beetween services
			Discovery App	
			Report App	
		Discovery App	Data App	
			Audit App	
		Reports App	Data App	
			Audit App	
			Report App	Report Worker -> Report Api
		Analytics App	Data App	

- **Owned** - Responsible to create and maintain that token
- **Context** - High-level description
- **Used** - Consumed the token

5.5 Lints

Each project uses lint program to guarantee the pattern and quality.

5.5.1 JavaScript (Client App)

Use eslint, default vue-loader

```
npm run lint
```

5.5.2 NodeJs (Server App)

Eslint too,

Airbnb with some changes

```
npm run lint
```

```
"rules": {
    "linebreak-style": [
        0
    ],
    "semi": [
        2,
        "always"
    ],
    "semi-spacing": [2, {            // http://eslint.org/docs/rules/semi-spacing
        "before": false,
        "after": true
    }],
    "no-console": 0,
    "strict": ["error", "global"],
    "no-catch-shadow": 2, // disallow the catch clause parameter name being the same␣
↪as a variable in the outer scope (off by default in the node environment)
```

(continues on next page)

(continuação da página anterior)

```
    "no-delete-var": 2, // disallow deletion of variables
    "no-label-var": 2, // disallow labels that share a name with a variable
    "no-shadow": 2, // disallow declaration of variables already declared in the␣
↪outer scope
    "no-shadow-restricted-names": 2, // disallow shadowing of names such as arguments
    "no-undef": 0, // disallow use of undeclared variables unless mentioned in a /
↪*global */ block
    "no-undef-init": 2, // disallow use of undefined when initializing variables
    "no-undefined": 2, // disallow use of undefined variable (off by default)
    "no-unused-vars": 2, // disallow declaration of variables that are not used in␣
↪the code
    "no-use-before-define": 2, // disallow use of variables before they are defined
    "complexity": 0, // specify the maximum cyclomatic complexity allowed in a␣
↪program (off by default)
    "no-var": 2, // require let or const instead of var (off by default)
    "generator-star-spacing": [2, "before"] // enforce the spacing around the * in␣
↪generator functions (off by default)
}
```

5.5.3 Python 3 (Discovery, Scheduler and Reports)

pytlint, default config.

```
npm run lint
```

5.6 Tests

Each service need to be testing.

5.6.1 Server APP

Testing use Mocha + Chai and Sinon, test coverage with Istambul

```
npm run test

npm run e2e

npm run unit

#if you like to code and testing in the same time
npm run tdd
```

```
gulp test_e2e
```

Coverage

```
istanbul cover ./node_modules/mocha/bin/_mocha test/**/*js
```

Coveralls

5.6.2 Discovery APP

Testing use pytest

```
npm run test

python -m unittest discover
```

Coveralls

5.6.3 Reports APP

Testing use pytest

```
npm run test

python -m unittest discover
```

Coveralls

5.6.4 Data Layer APP

Testing use pytest

```
npm run test

python -m unittest discover
```

Coveralls

5.6.5 Analytics Apps

Testing use pytest

```
npm run test

python -m unittest discover
```

Coveralls

5.6.6 Analytics Front

Testing use pytest

```
npm run e2e
```

Coveralls	

5.6.7 Audit App

Testing use pytest

```
npm run e2e
```

Coveralls	

5.7 Quality Assurance

We use some tools to mensure quality.

Nota: Travis with all projects

5.7.1 Client Maestro

Codacy	
Travis	
CodeClimate	

5.7.2 Server App

CodeClimate	
Travis	
DavidDm	
Codacy	
Coveralls	

5.7.3 Discovery Maestro

Codacy	
Travis	
CodeClimate	

5.7.4 Report Maestro

Codacy	
Travis	
CodeClimate	

5.7.5 Scheduler Maestro

Codacy	
Travis	
CodeClimate	

5.7.6 Data Layer API

Codacy	
Travis	
CodeClimate	

5.7.7 Analytics App

Codacy	
Travis	
CodeClimate	

5.7.8 Analytics Front

Codacy	
Travis	
CodeClimate	

5.7.9 Audit App

Codacy	
Travis	
CodeClimate	

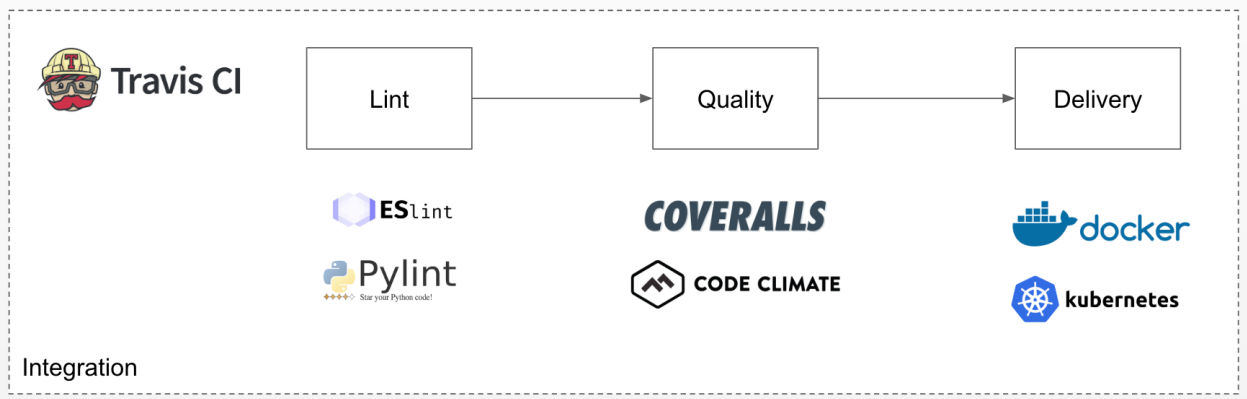
5.8 Third Party

Third Party Support

Provider	Library
AWS	Boto3
OpenStack	OpenStackSDK

5.9 CI and CD

Maestro Server using a lots of opensource project to run each pipeline.



5.10 Versions

Microservices compatible versions

5.10.1 v0.5x - Omega

Admin	0.5.x

5.10.2 v0.5x - Beta

Break changes - All services of version 0.5.x isnt compatible with early versions.

Client	0.14.x
Server	0.5.x
Discovery	0.5.x
Scheduler	0.5.x
Data	0.5.x
Reports	0.5.x
Analytics	0.5.x
Analytics Front	0.5.x
Audit	0.5.x

5.10.3 v0.4x - Beta

Break changes - All services of version 0.4.x isnt compatible with early versions.

Client	0.13.x
Server	0.4.x
Discovery	0.4.x
Scheduler	0.4.x
Data	0.4.x
Reports	0.4.x
Analytics	0.4.x
Analytics Front	0.4.x
WebSocket	0.4.x

5.10.4 v0.3x - Beta

Client	0.12.x
Server	0.3.x
Discovery	0.3.x
Scheduler	0.3.x
Data	0.3.x
Reports	0.2.x

5.10.5 v0.2x - Alpha

Client	0.11.x
Server	0.2.x
Discovery	0.2.x
Scheduler	0.2.x
Data	0.1.x
Reports	0.1.x

CAPÍTULO 6

Contrib

6.1 Reporting issues

- Describe what you expected to happen.
- If possible, include a minimal, complete, and verifiable example to help us identify the issue. This also helps check that the issue is not with your own code.
- Describe what actually happened. Include the full traceback if there was an exception.

6.2 Submitting patches

- All test need to be pass
- All lint need to be green
- Include tests if your patch is supposed to solve a bug, and explain clearly under which circumstances the bug happens. Make sure the test fails without your patch.

Nota: All contribuition will be accept by Pull Request

CAPÍTULO 7

Donate

I have made Maestro Server with my heart, think to solve a real operation IT problem. Its not easy, take time and resources.

The donation will be user to:

- All pages are hosted on AWS
- Demo service is hosted on AWS, and we would like to use kubernetes environment.
- Use telemetry and monitoring services to improve the system.
- Create new features, implement new providers.
- Maintenance libs, securities flaws, and technical points.

If you could, you can help me, buy me a coffee, together we can keep the project up and create excited new features.

CAPÍTULO 8

License

MIT License

Copyright (c) 2018 Maestro Server - Felipe Signorini and contributors

Some rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Names of the contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the «Software»), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED «AS IS», WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.