# Maestro Server - Cloud Inventory Documentation

*Release 0.1*

Felipe Signorini

Sep 25, 2018

# 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.

# CHAPTER 1

# Overview

## 1.1 What is 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.

## 1.2 What problems does it solve?

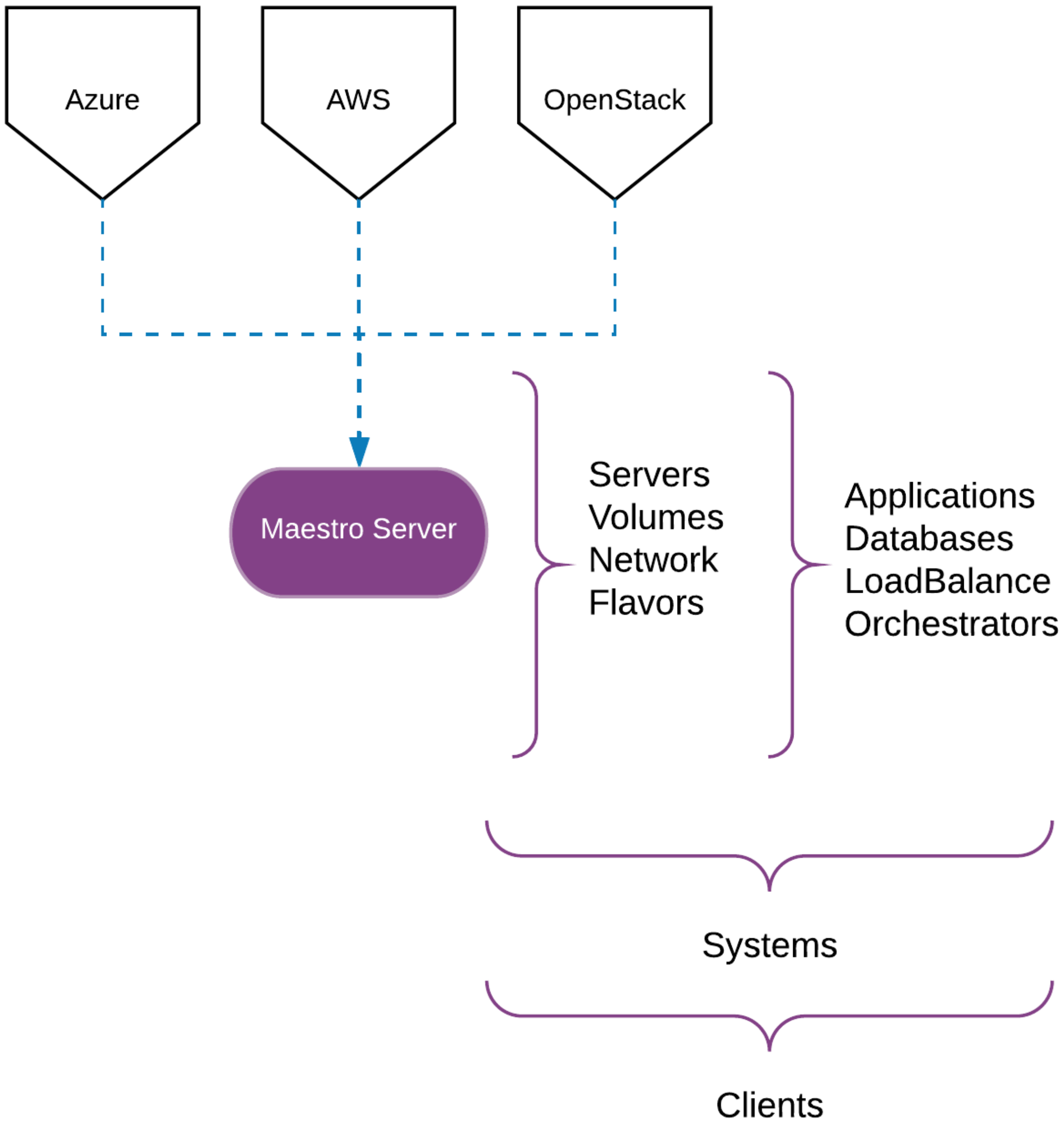
Maestro had built to solve some problems founded in operating multi-cloud environments, multi shared devops culture and multi clients, where turns hard to keep track the lastest environment state, bottlenecks to apply a compliance in all teams, visualization gaps to understand your's infrastructure state, access security flaws for internals employees and out of date documentation.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

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.

# CHAPTER 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 6.10 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 |
| Data DB App | Data layer | Python, flask |

## 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.

**Note:** 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)

**Note:** Download docker-compose file.

**Warning:** 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/"
        depends_on:
        - server

    server:
        image: maestroserver/server-maestro
        ports:
        - "8888:8888"
        environment:
        - "MAESTRO_MONGO_URI=mongodb/maestro-client"
        - "MAESTRO_DISCOVERY_URI=http://discovery:5000"
        - "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:
        - "MAESTRO_DATA_URI=http://data:5010"
        - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
        depends_on:
        - rabbitmq
        - data

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

(continues on next page)

(continued from previous page)

```
        - "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"
        - "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"
        - "CELERY_BROKER_URL=amqp://rabbitmq:5672"
        depends_on:
        - rabbitmq
        - data

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

    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)

(continued from previous page)

```
        - "27017:27017"

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


volumes:
    mongodata: {}
```

## 2.3 Vagrant

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

**Note:** Download vagrantFile.

**Note: HA - High availability and critical system**

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

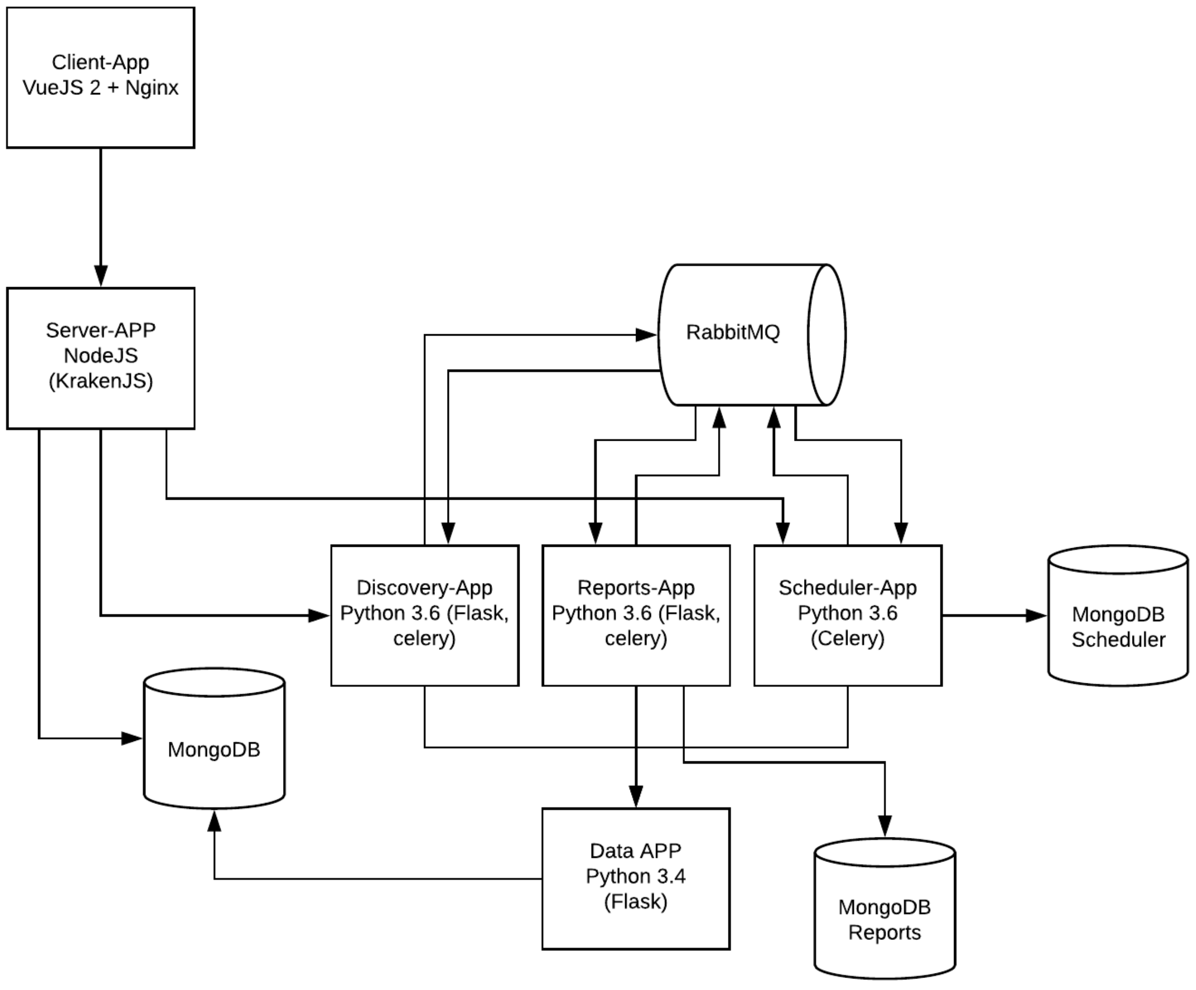
# CHAPTER 3

# 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

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/"
```

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

**Warning:**

- API_URL it's rest endpoint provide by server-app.
- 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"
```

(continues on next page)

(continued from previous page)

```
    environment:
    - "MAESTRO_MONGO_URI=mongodb/maestro-client"
    - "MAESTRO_DISCOVERY_URI=http://discovery:5000"
    - "MAESTRO_REPORT_URI=http://reports:5005"
```

```
docker run -p 8888:8888  -e "MAESTRO_MONGO_URI=mongodb/maestro-client" -e "MAESTRO_
↪DISCOVERY_URI=http://localhost:5000" -e "MAESTRO_REPORT_URI=http://localhost:5005"␣
↪maestroserver/server-maestro
```

**Warning:**

- MAESTRO_MONGO_URI - Must be mongodb, mongodb://{uri}/{db-name}
- 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)

**Env variables**

| Env Variables | Example | Description |
|---|---|---|
| MAESTRO_PORT | 8888 | |
| NODE_ENV | development\|production | |
| MAESTRO_MONGO_URI | localhost/maestro-client | DB string connection |
| MAESTRO_SECRETJWT | XXXX | Secret key - session |
| MAESTRO_SECRETJWT_FORGOT | XXXX | Secret key - forgot request |
| MAESTRO_SECRET_CRYPTO_FORGOT | XXXX | Secret key - forgot content |
| MAESTRO_DISCOVERY_URI | http://localhost:5000 | Url discovery-app (flask) |
| MAESTRO_REPORT_URI | http://localhost:5005 | Url reports-app (flask) |
| 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 | | |
| MAESTRO_UPLOAD_TYPE | S3 or Local | Upload mode |
| AWS_ACCESS_KEY_ID | XXXX | |
| AWS_SECRET_ACCESS_KEY | XXXX | |
| AWS_DEFAULT_REGION | us-east-1 | |
| AWS_S3_BUCKET_NAME | maestroserver | Buckete name |
| 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"
```

```
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" maestroserver/discovery-maestro-celery
```

**Warning:**

- MAESTRO_REPORT_URI - Enpoint API of Discovery - default port is 5010
- MAESTRO_DATA_URI - Enpoint API of Data App - default port is 5000
- 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_SECRETJWT | xxxx | Same that Server App |
| 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"
```

(continues on next page)

(continued from previous page)

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

**Warning:**

- 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_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" 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_REPORT_RESULT_QTD | 200 | Limit default |
| MAESTRO_TIMEOUT_DATA | 10 | Timeout for data retrived |
| MAESTRO_TIMEOUT_WEBHOOK | 5 | Timeout for notifications |
| MAESTRO_INSERT_QTD | 20 | Prefetch data insert |
| MAESTRO_GWORKERS | 2 | Gworkers thread pool |
| CELERY_BROKER_URL | amqp://rabbitmq:5672 | RabbitMQ connection |

## 3.6 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
```

(continues on next page)

(continued from previous page)

```
```

**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 |

## 3.7 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"
```

```
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 "CELERY_BROKER_URL=amqp://
↪rabbitmq:5672" maestroserver/scheduler-maestro-celery
```

**Warning:**

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

**Env variables**

| Env Variables | Example | Description |
|---|---|---|
| MAESTRO_DATA_URI | http://data:5010 | Data Layer API URL |
| MAESTRO_MONGO_URI | localhost | MongoDB URI |
| MAESTRO_MONGO_DATABASE | maestro-client | Mongo Database name |
| CELERY_BROKER_URL | amqp://rabbitmq:5672 | RabbitMQ connection |

## 3.8 HA - Production Read

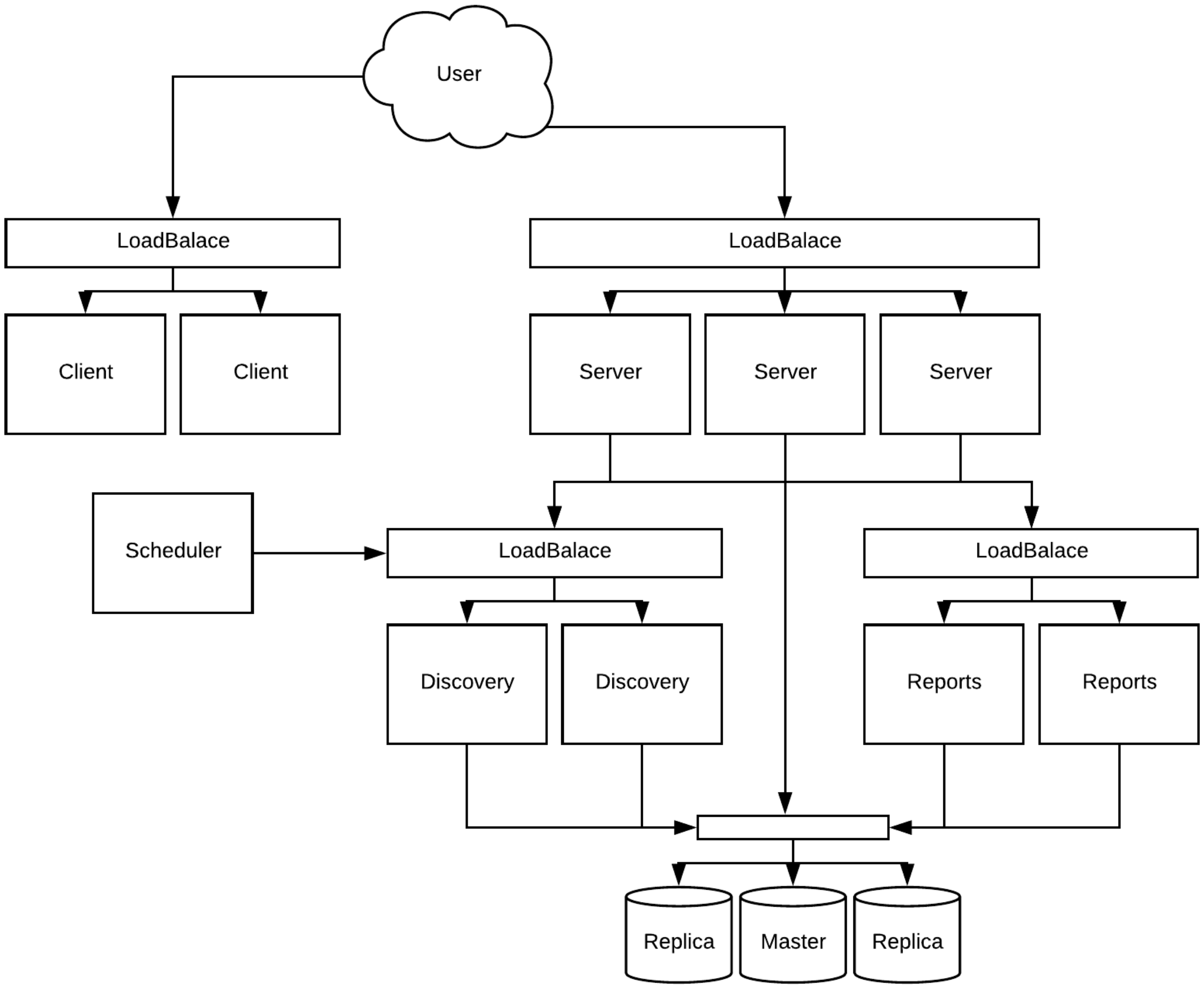
Not exist one way to do right, but we have some concerns, needed a backup system, HA avaliability, monitoring and etc, like a normal app system. Maestro built with 12 factory in mind, its able to hight availability and horizontal scaling.

### 3.8.1 12 Factory and Horizontal Scaling

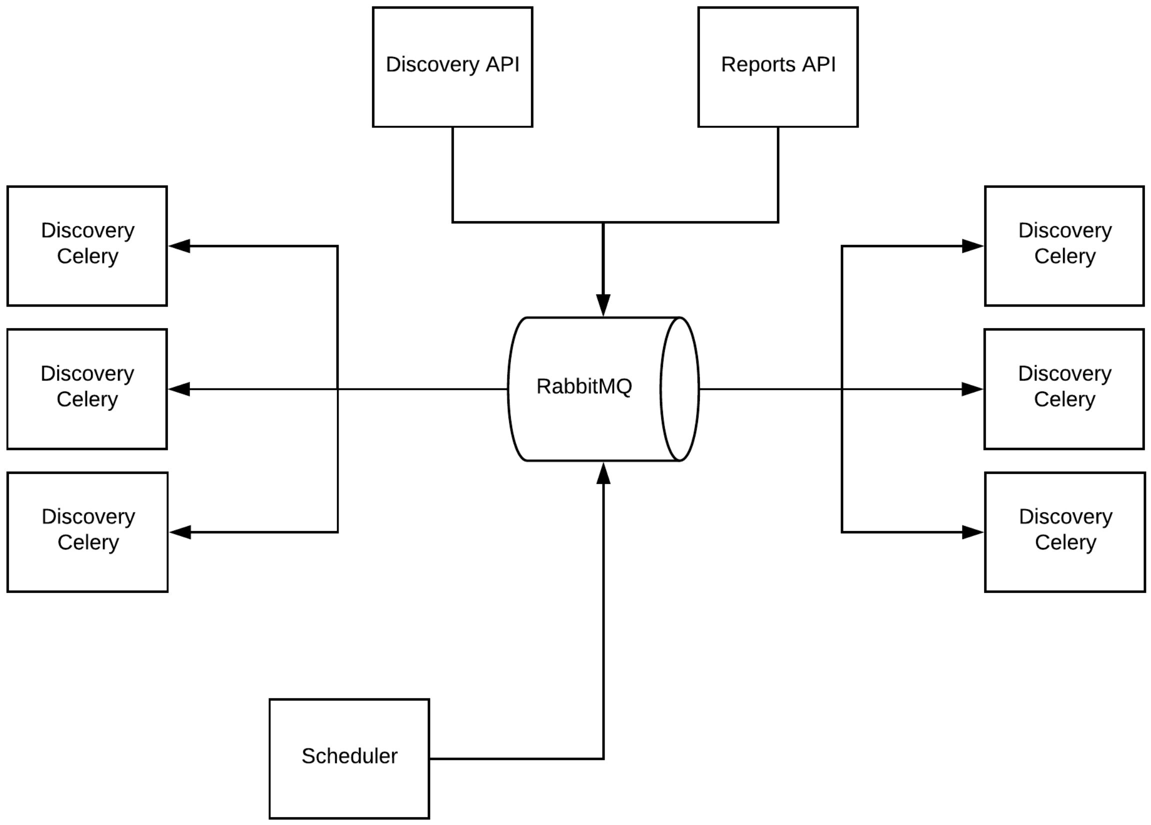
All services need work in stateless, like session, images or any process, we use JWT for authentication and isolated storage server.

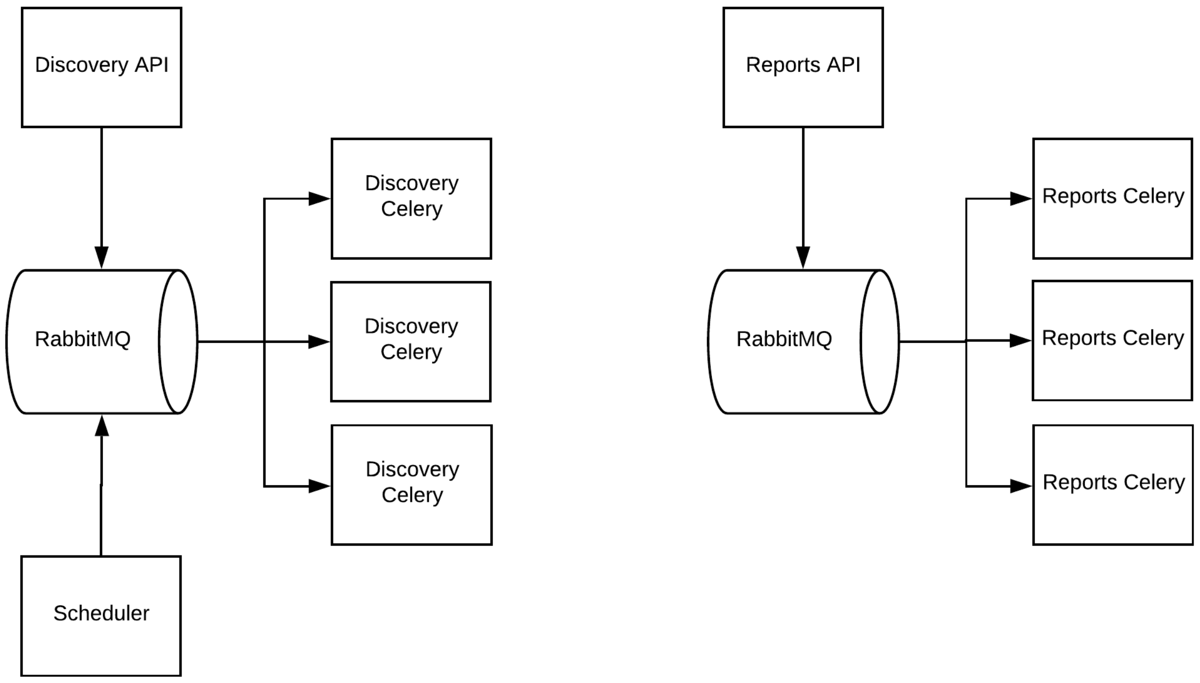
- Avoid to use any local configurate like local upload, maestro has supported to use s3 storage object.
- Put all connection config in env docker setup, would able to use kubernetes, rancher or any orchestration is good advice.
- Its possible to deploy discovery api in one server, and discovery celery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

One example setup, can be in each node,



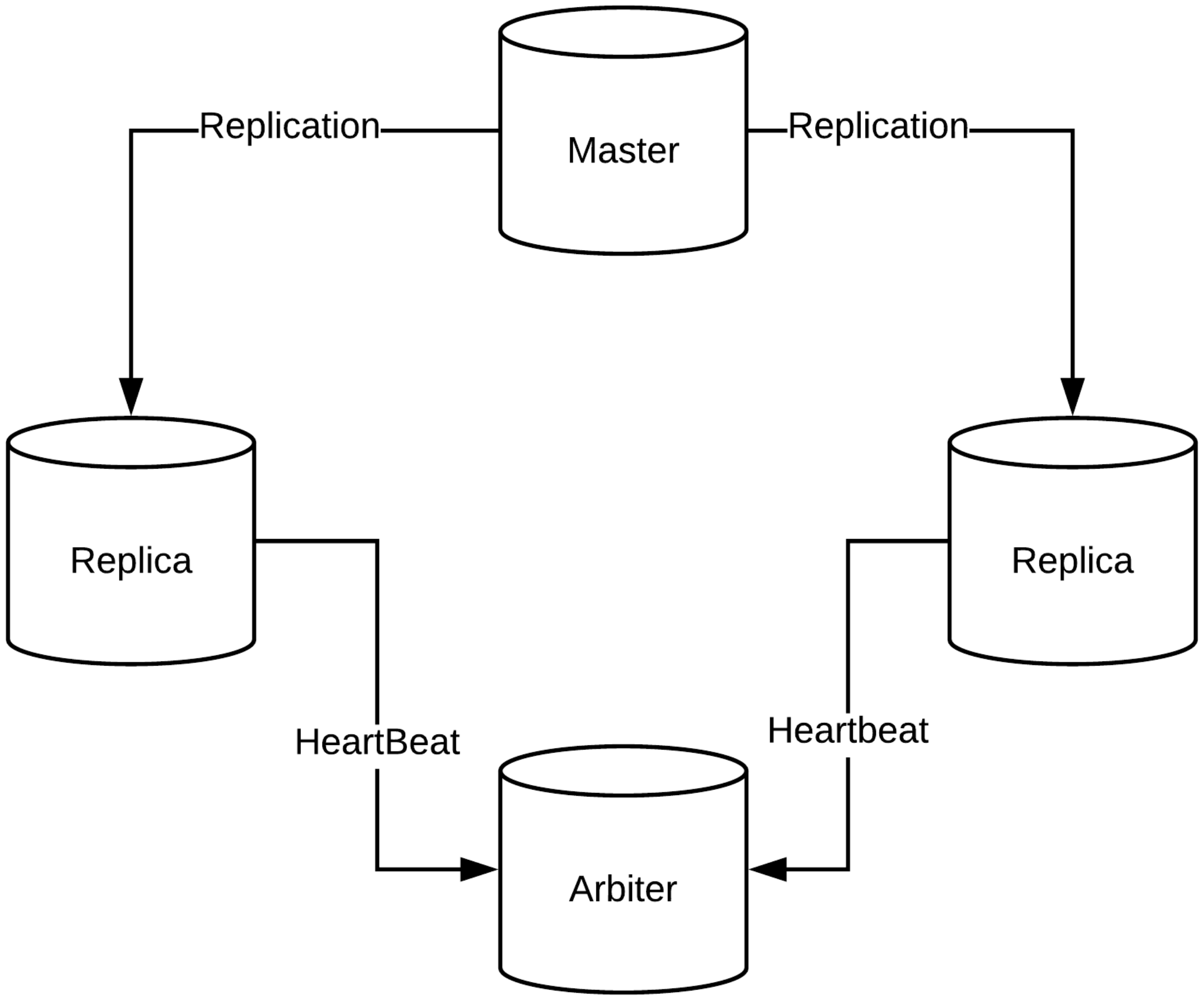
It’s possible to improve discovery and reports app


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


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

### 3.8.2 MongoDB

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



### 3.8.3 Scheduler Beat App

**Danger:** 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.

### 3.8.4 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/

## 3.9 Advanced configs

### 3.9.1 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 | true\|false | Enable TLS connect |
| SMTP_IGNORE | true\|false | 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
```

### 3.9.2 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'
```

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

## 3.9.3 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

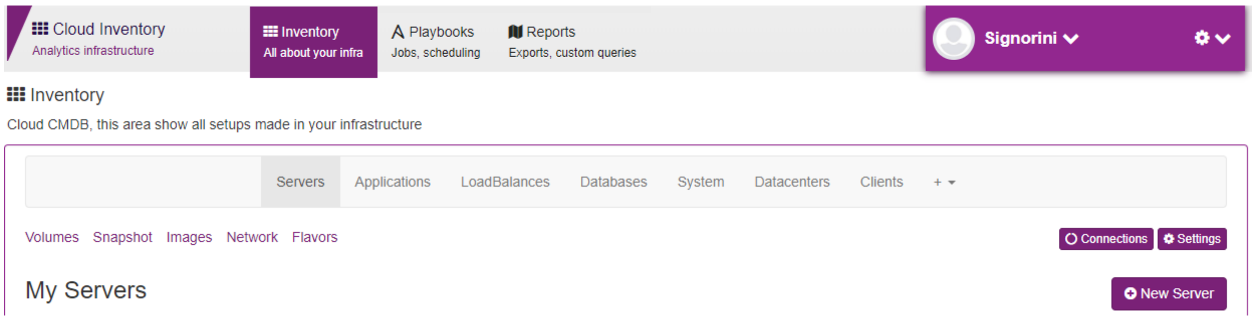


Fig. 1: THEME=lotus

### Gold

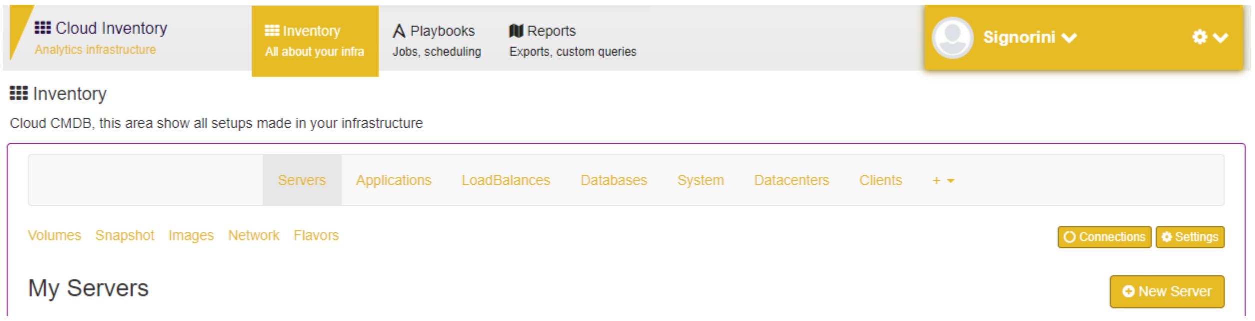


Fig. 2: THEME=gold

### Wine

### Blue

### Dark

### Green

### Orange

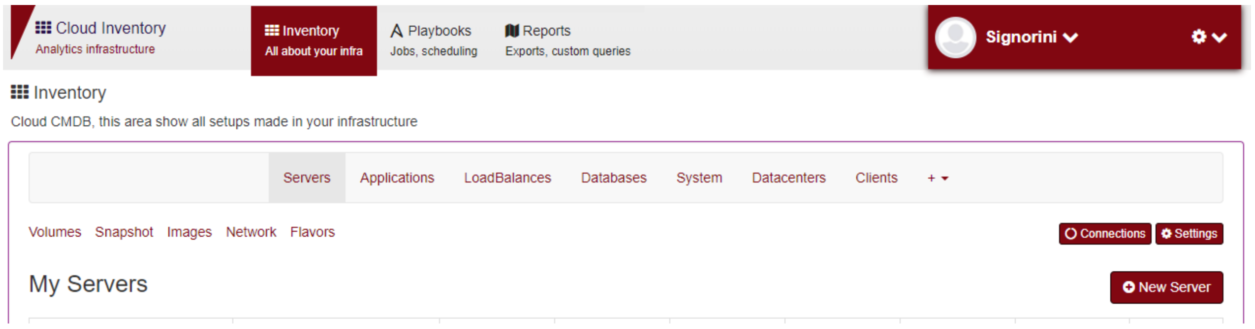


Fig. 3: THEME=wine

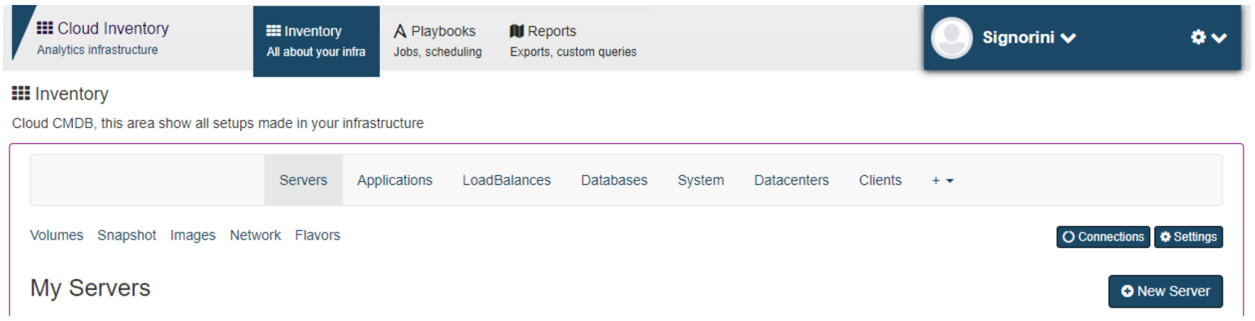


Fig. 4: THEME=blue

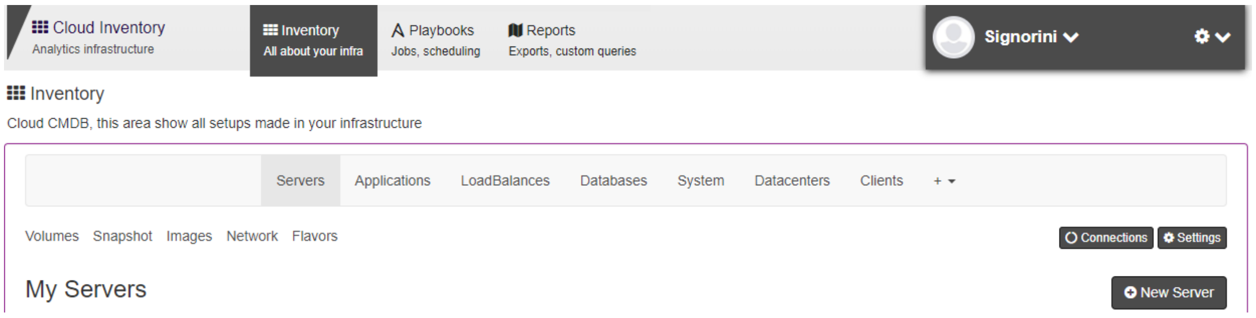


Fig. 5: THEME=dark

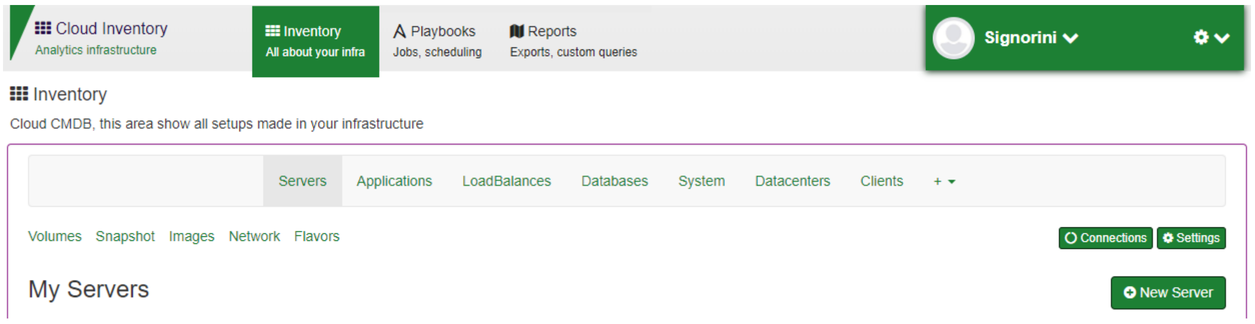


Fig. 6: THEME=green

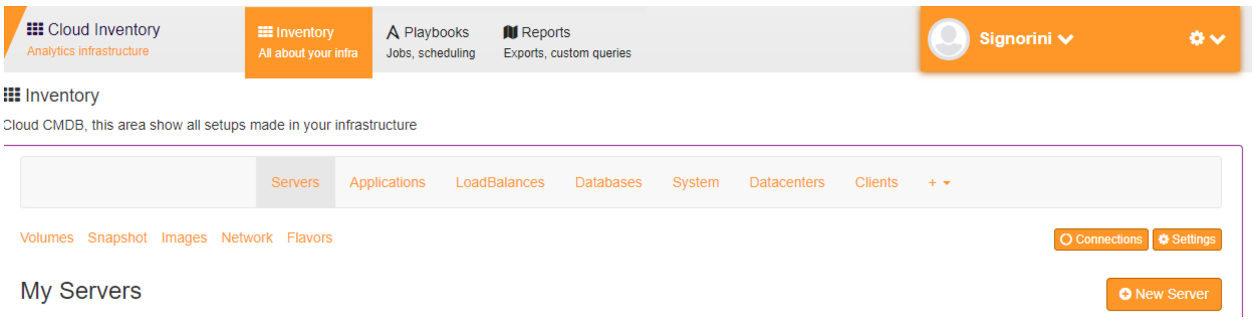


Fig. 7: THEME=orange

# CHAPTER 4

# User Guide

The inventory is divided into three parts:

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

> **Playbooks:** Location to register and manager playbooks, scheduler commands, tasks execution and audit actions.

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

## 4.1 Cloud Inventory

### 4.1.1 Inventory

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

#### 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 |

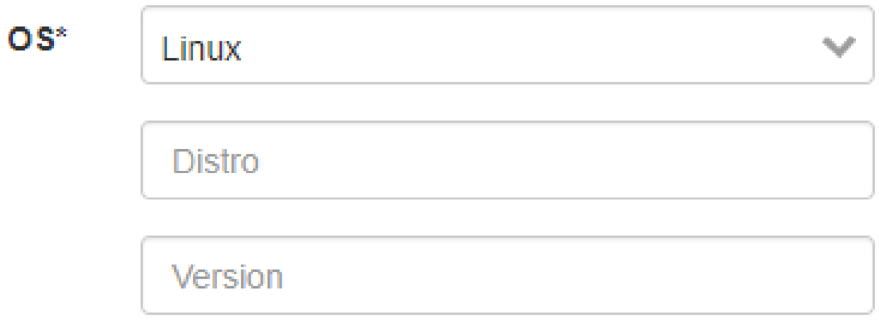
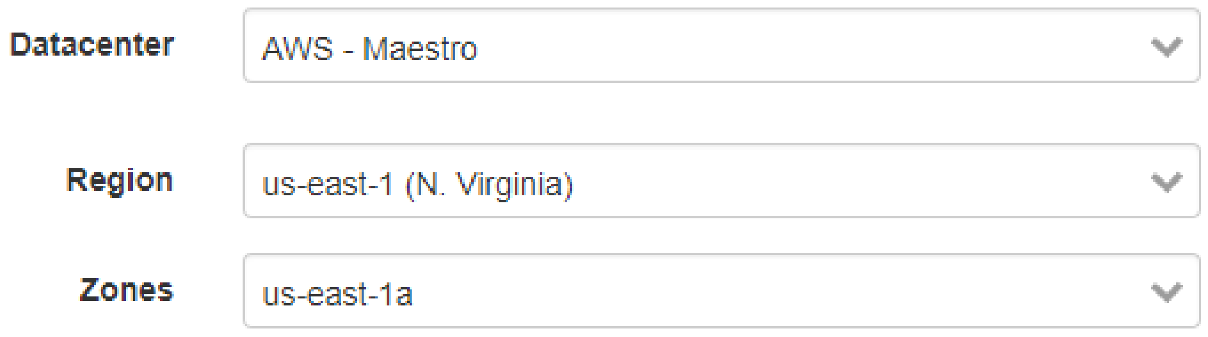


Fig. 1: Setup OS server

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. |

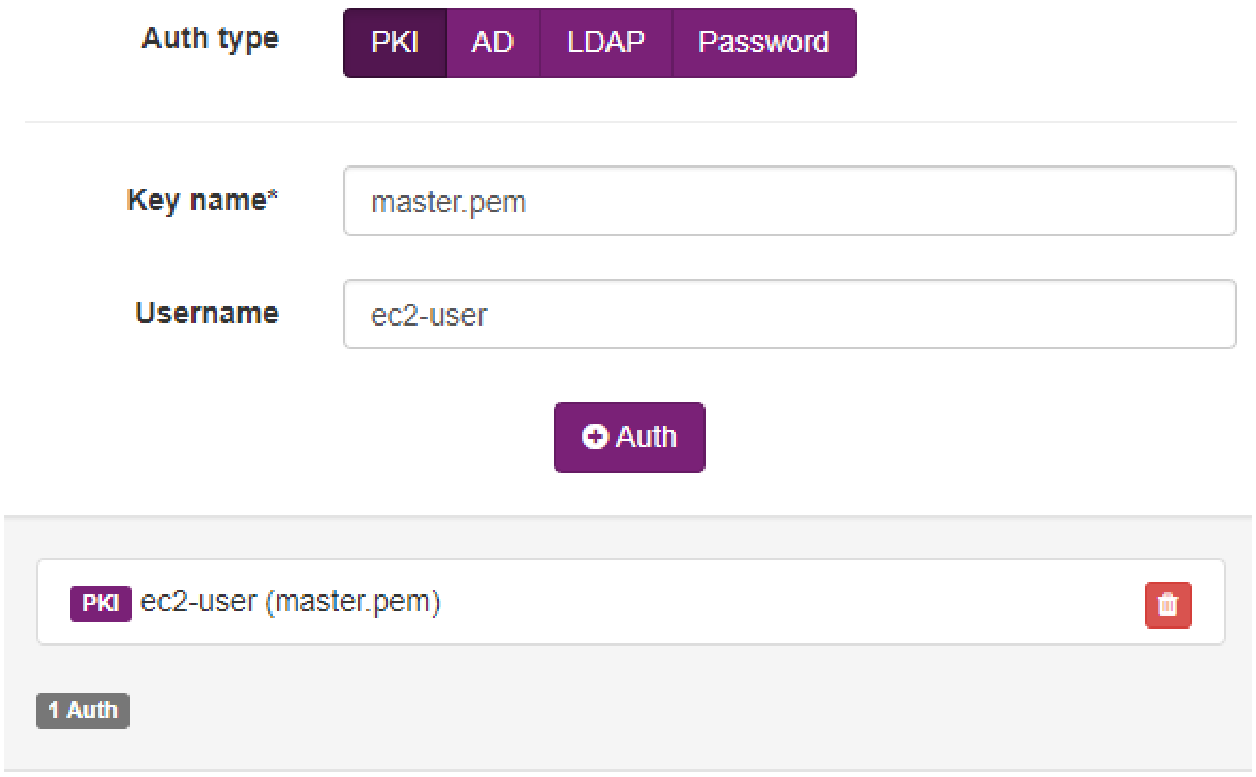
**Note:** Services is a very important field in servers, which this information the system will try to find some relation with applications, for example if you register Oracle Database, and create a database and register a relation between this servers, the system will automatically create a service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


Datacenter: AWS - Maestro

Region: us-east-1 (N. Virginia)

Zones: us-east-1a

Fig. 2: Assing a dc name, region and zone.



Key name*: master.pem

Username: ec2-user

PKI ec2-user (master.pem)

1 Auth

Fig. 3: Register, which mode you can to access and authenticate on server.

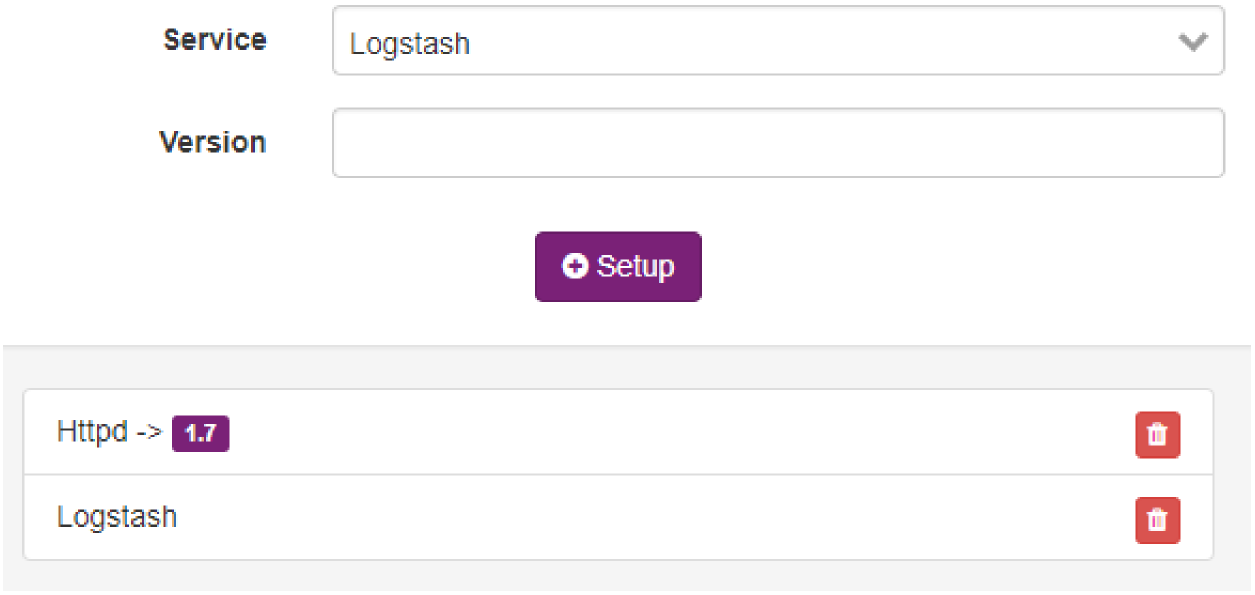


Fig. 4: Related services running.

### 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

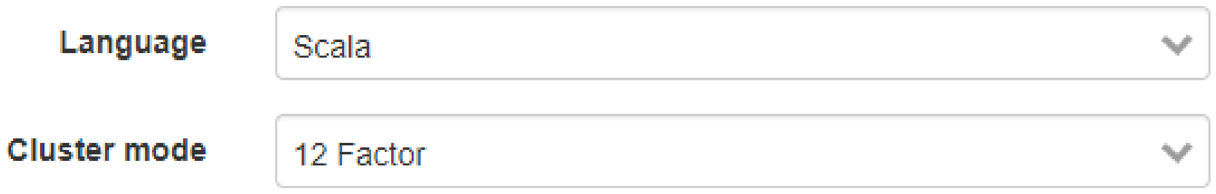
| Field | Functional |
|---|---|
| Role | Point information like endpoint, commands, health check, its good for doc. |
| System | Systems related it. |
| Server | Servers deployed by app. |
| Deploy | List of ways to deploy this app. |

---

---

### Databases

*Inventory > Database*



| | |
|---|---|
| Language | Scala |
| Cluster mode | 12 Factor |

Fig. 5: Choose language like node or php.

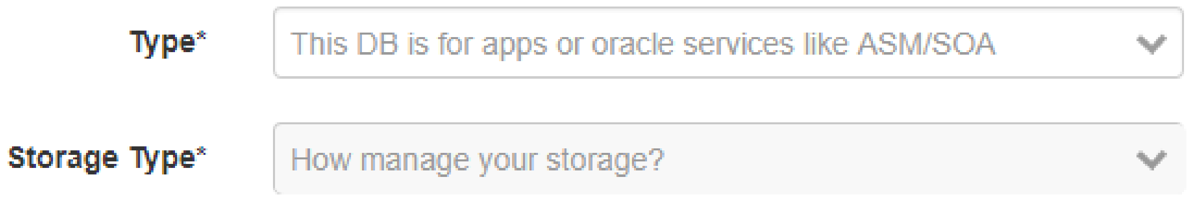
Databases are it all with data persistence responsibility, can be relational, norelational, in memory, distribuited storage and etc.

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**

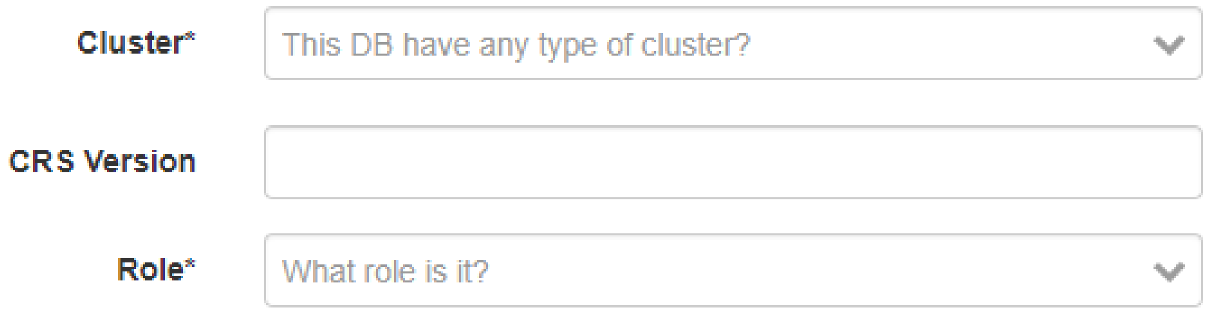
Support version 10g, 11g and 12g



| | |
|---|---|
| Type* | This DB is for apps or oracle services like ASM/SOA |
| Storage Type* | How manage your storage? |

Fig. 6: Choose how Oracle will be storage data, can be local disk, networks disk or ASM.

---



| | |
|---|---|
| Cluster* | This DB have any type of cluster? |
| CRS Version | |
| Role* | What role is it? |

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

---

---

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**

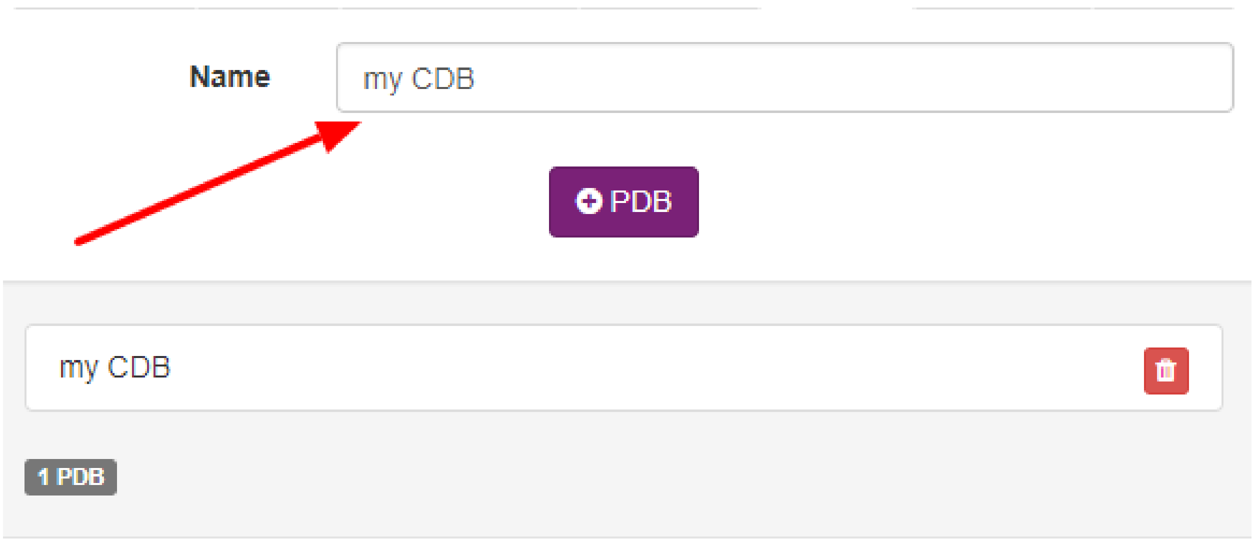
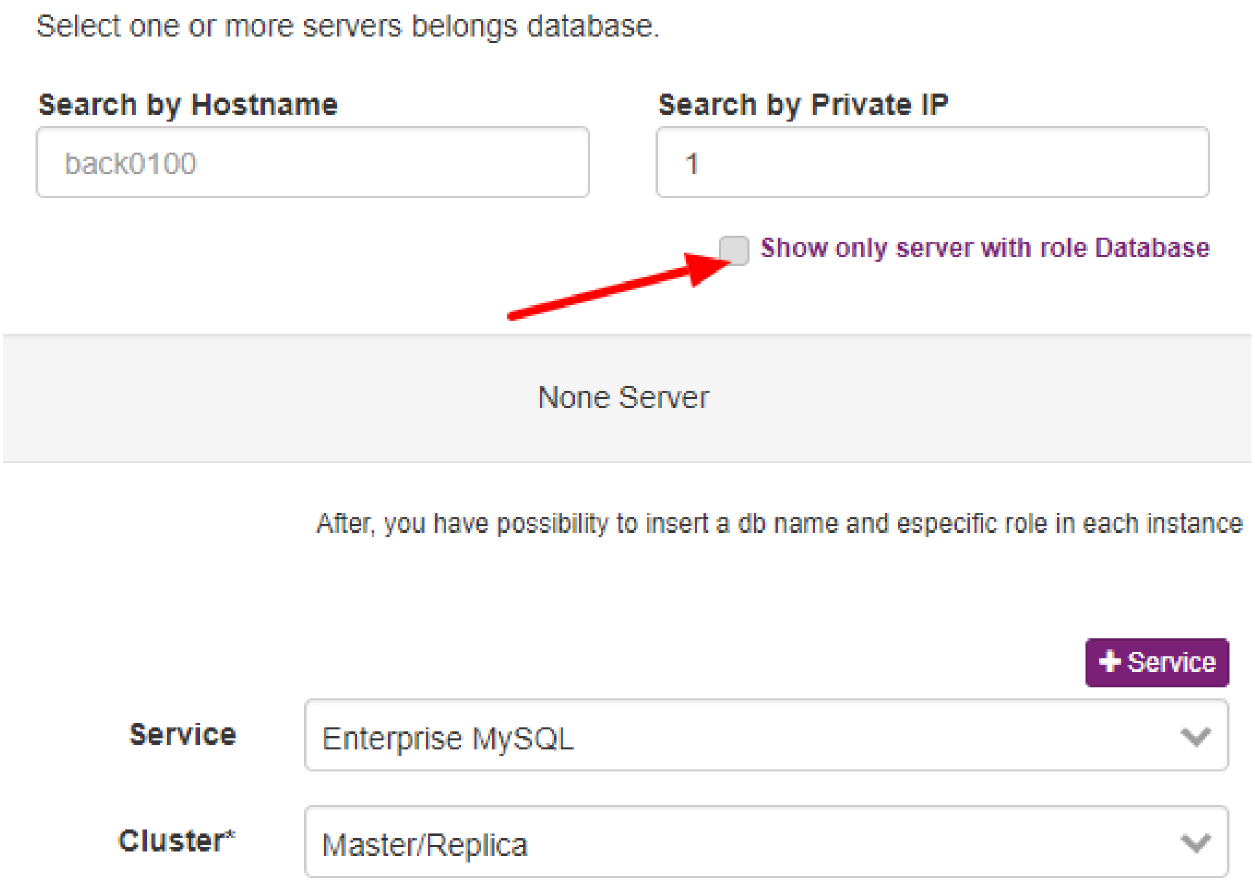


Fig. 7: Which CDBS run in oracle database.


Select one or more servers belongs database.

Search by Hostname: back0100

Search by Private IP: 1

Show only server with role Database

None Server

After, you have possibility to insert a db name and especific role in each instance

Service: Enterprise MySQL

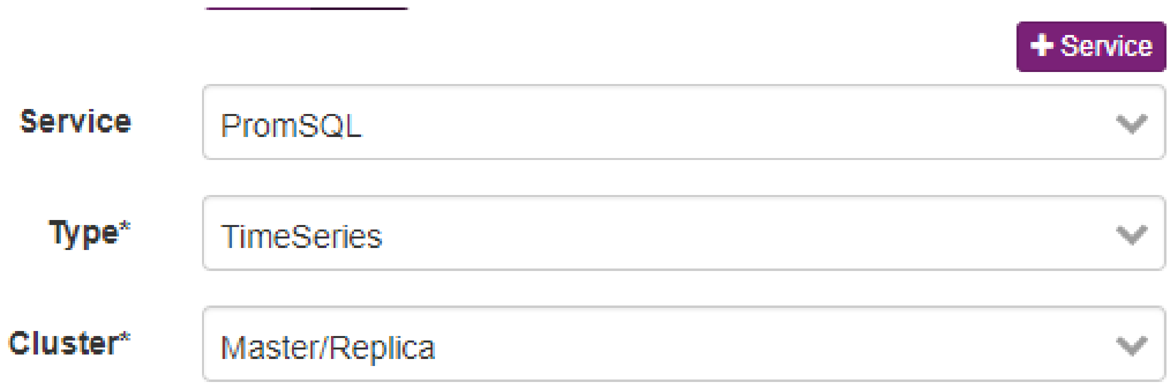
Cluster*: Master/Replica

Fig. 8: Version and mode to run.

Support MySQL, AWS Aurora, MariaDB, Percona and etc

**Other databases**

Partial support whitch all bases



| | |
|---|---|
| Service | PromSQL |
| Type* | TimeSeries |
| Cluster* | Master/Replica |

Fig. 9: 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. |

---

### 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 |

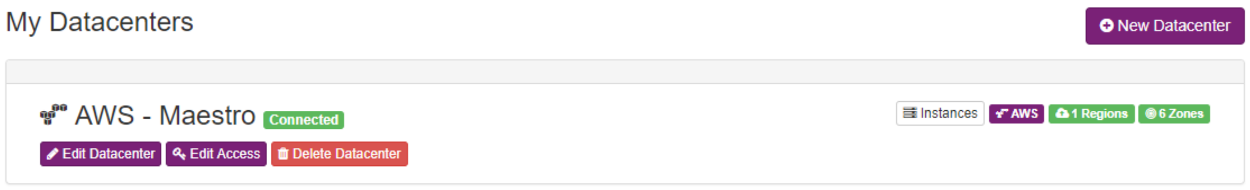


Fig. 10: List of datacenters, with instances, regiions and zones

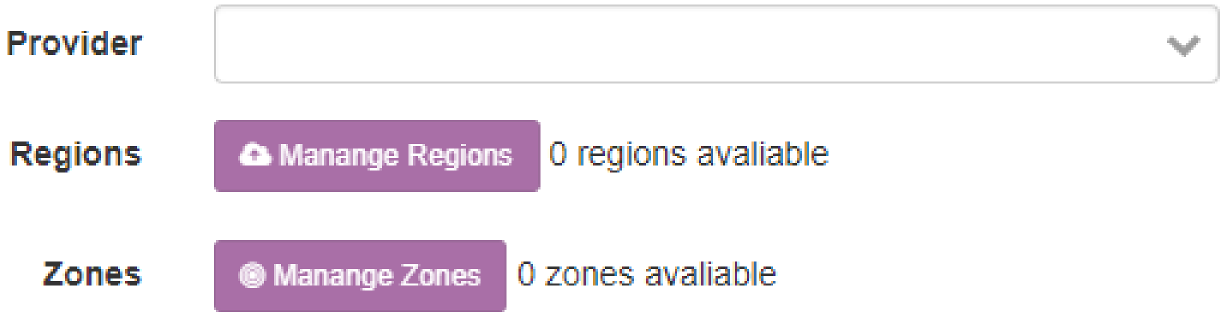


Fig. 11: You can choose the provider, regions and zones.

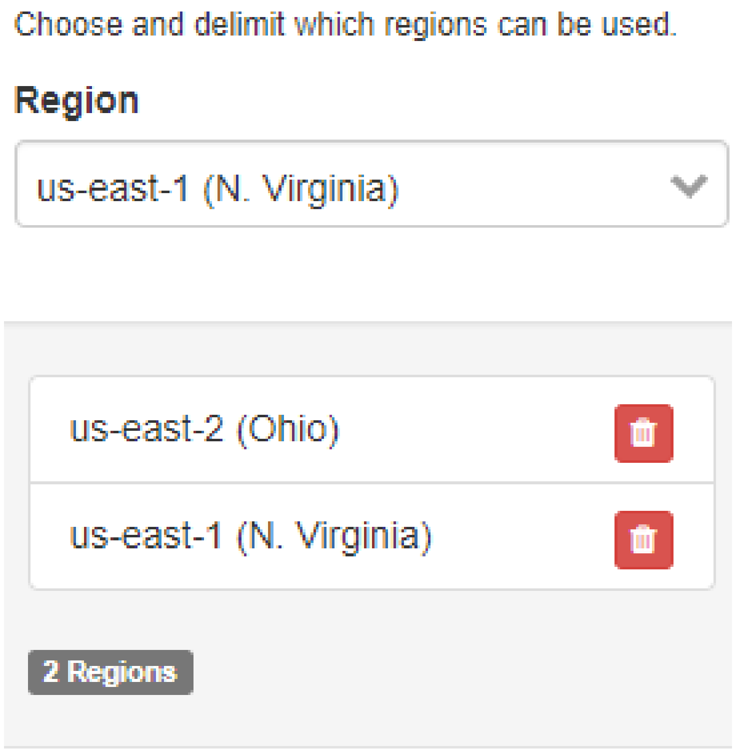


Fig. 12: Choose regions or create it.

### 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 |

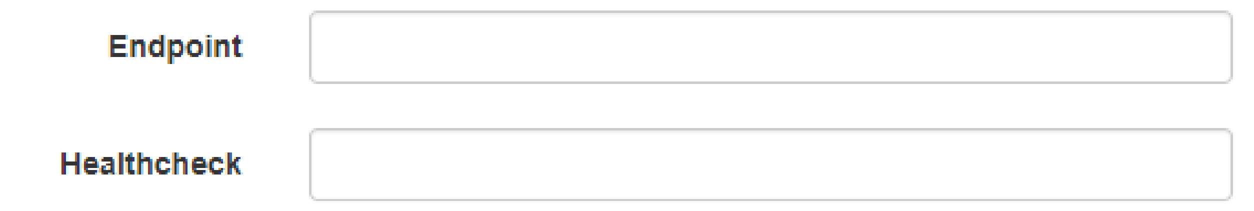
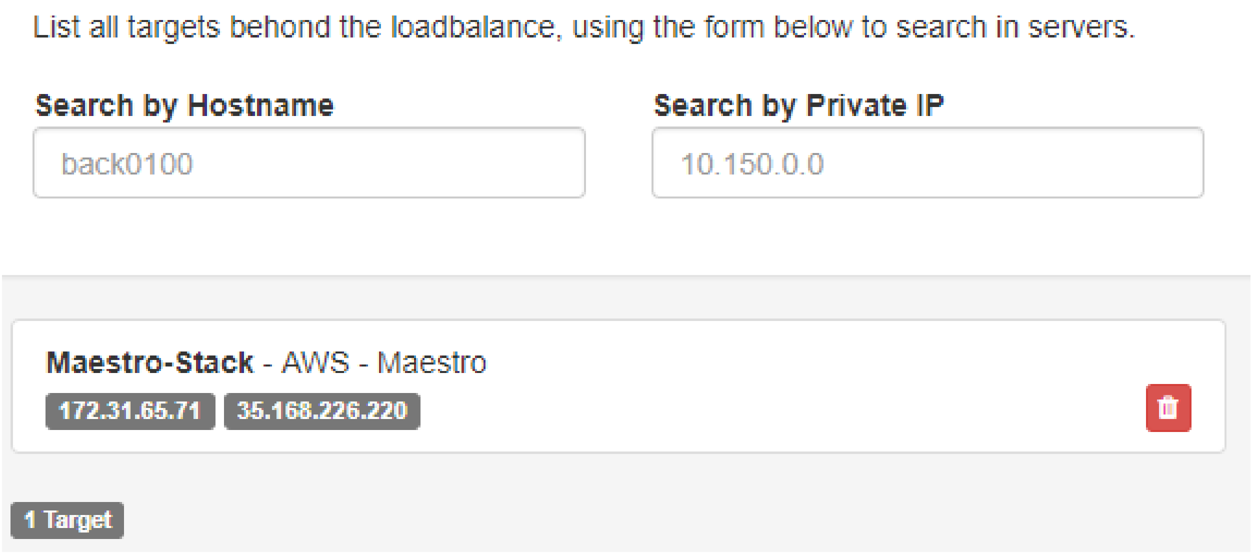


Fig. 13: Docs a endpoint and healthcheck used in app.


List all targets behond the loadbalance, using the form below to search in servers.

Search by Hostname: back0100

Search by Private IP: 10.150.0.0

Maestro-Stack - AWS - Maestro

172.31.65.71 35.168.226.220

1 Target

Fig. 14: 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 |

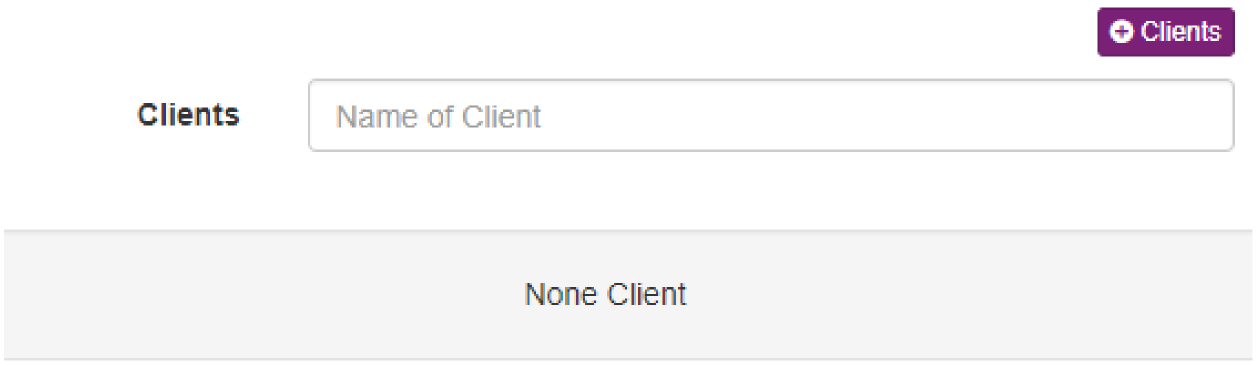


Fig. 15: Select owner of system

#### Clients

*Inventory > Clients*

SLA owner, clients

| Field | Functional |
|---|---|
| Contacts/Channel | How contact the client |

#### Services

*Inventory > Settings > Services*

Common program running inside on server

### 4.1.2 Auto Discovery

To setup connections in auto-discovery, go inventory > connections.

In version 0.1, we have two providers:

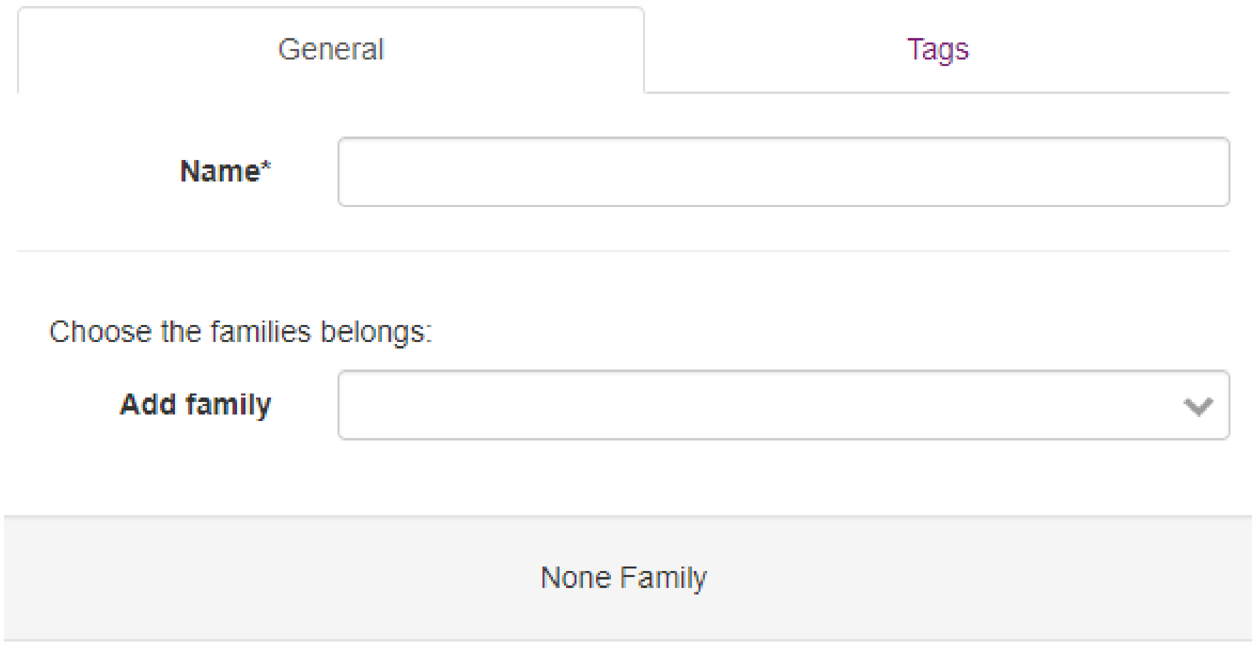
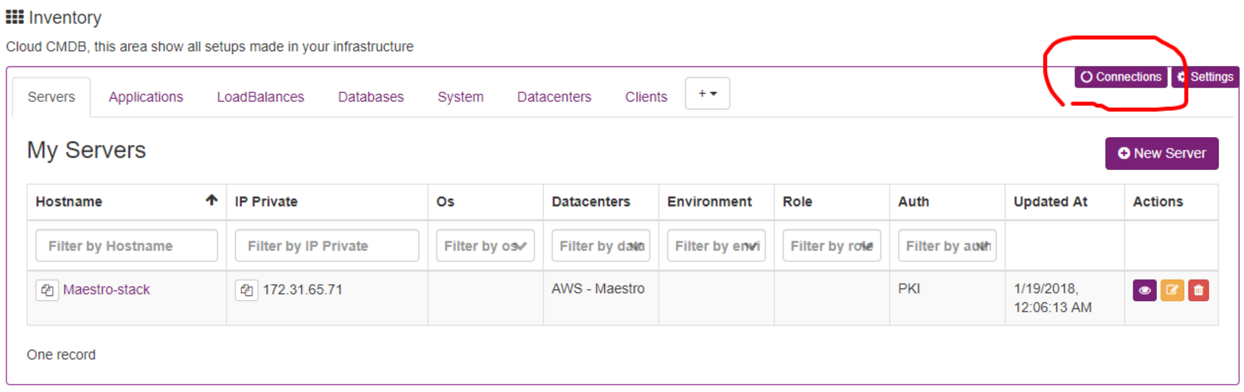


Fig. 16: Create a new service, to use in server and any app.



| Hostname | IP Private | Os | Datacenters | Environment | Role | Auth | Updated At | Actions |
|---|---|---|---|---|---|---|---|---|
| Maestro-stack | 172.31.65.71 | | AWS - Maestro | | | PKI | 1/19/2018, 12:06:13 AM | |

One record

Fig. 17: Access connection

### Connectiong with AWS

To register one aws account use access_key and secret_key

**Synchronized and permissions to grant.**

| | |
|---|---|
| server-List | ec2 describe_instances |
| loadbalance-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 |

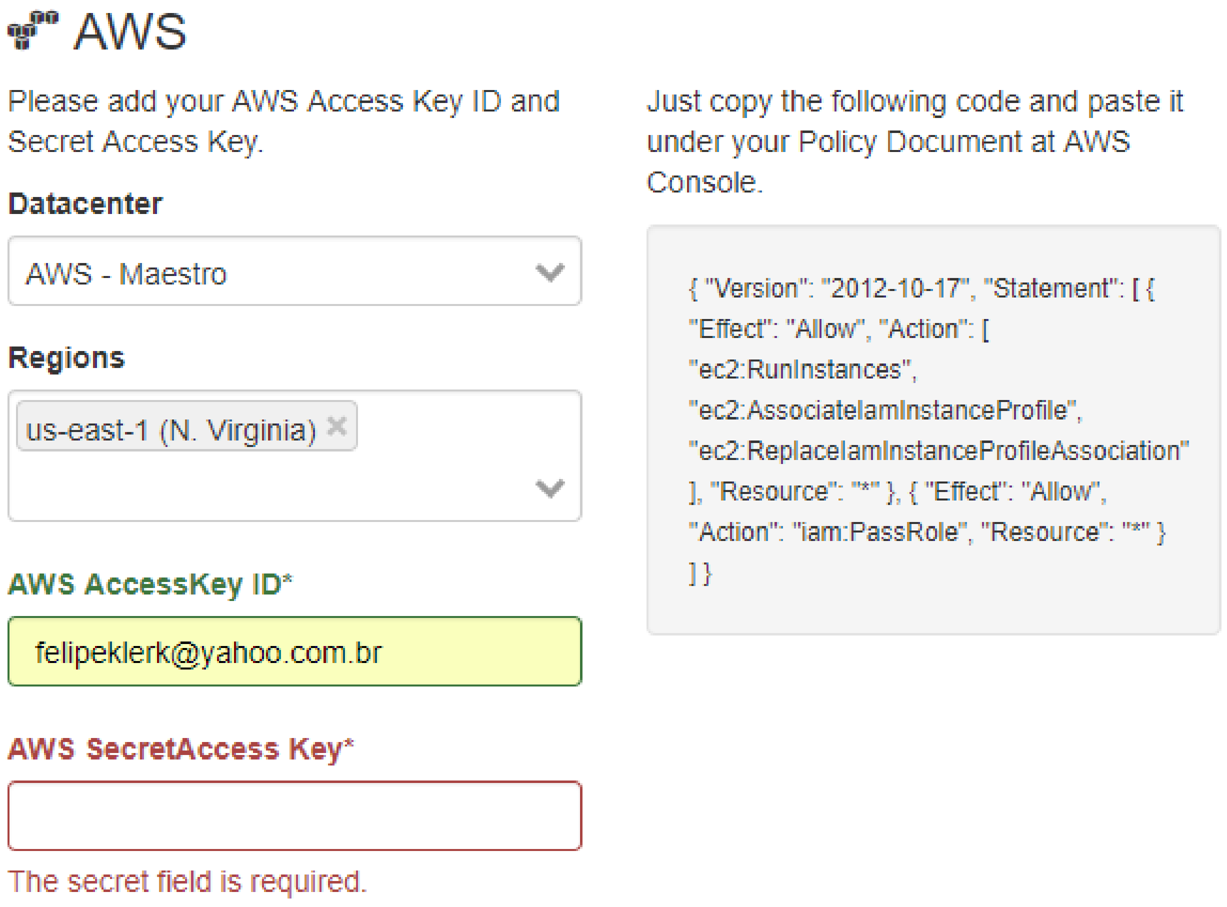
### Connectiong with OpenStack

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.


AWS

Please add your AWS Access Key ID and Secret Access Key.

Datacenter

AWS - Maestro

Regions

us-east-1 (N. Virginia)

AWS AccessKey ID*

felipeklerk@yahoo.com.br

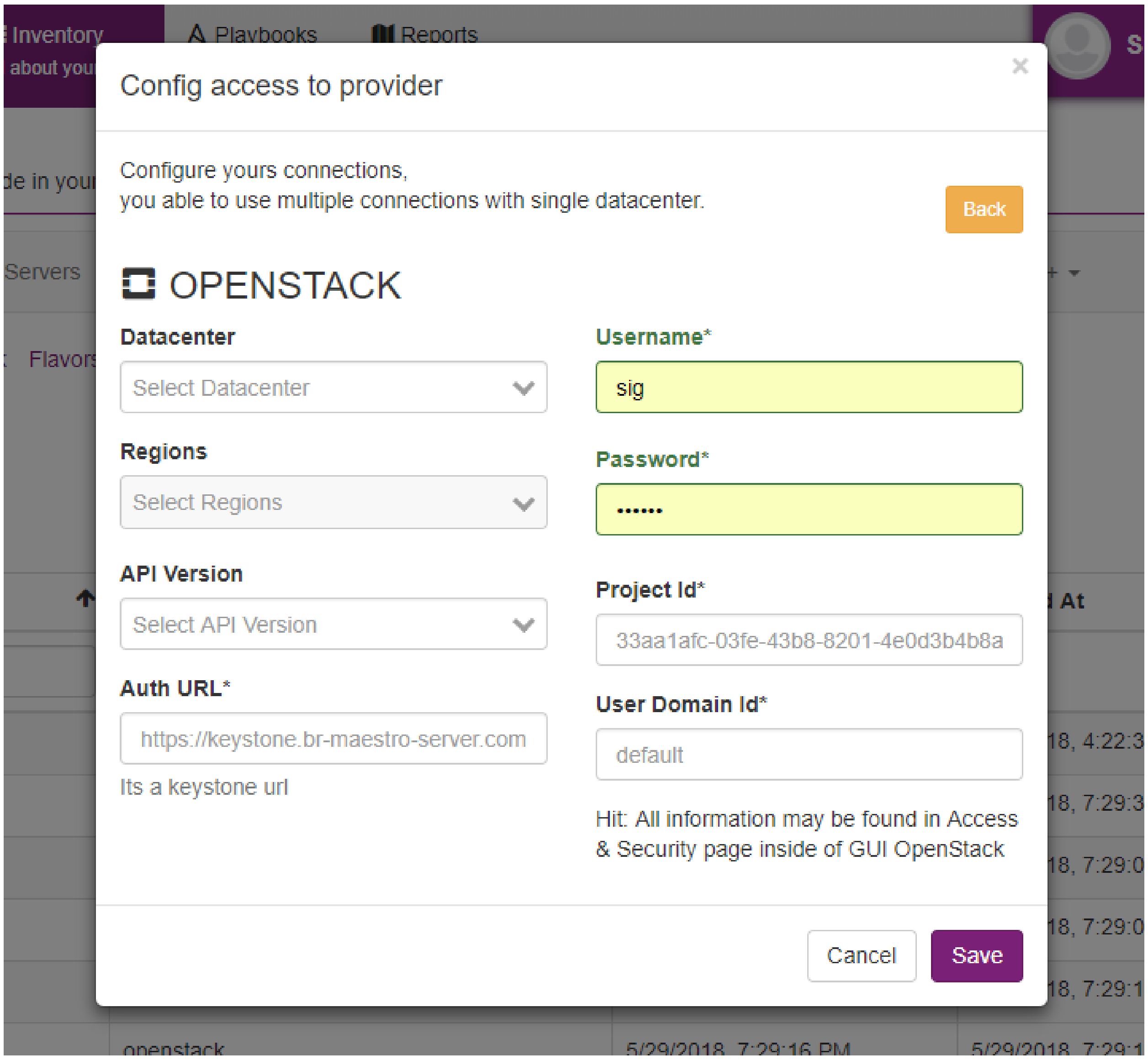
AWS SecretAccess Key*

The secret field is required.

Just copy the following code and paste it under your Policy Document at AWS Console.

```
{ "Version": "2012-10-17", "Statement": [ {
"Effect": "Allow", "Action": [
"ec2:RunInstances",
"ec2:AssociateIamInstanceProfile",
"ec2:ReplaceIamInstanceProfileAssociation"
], "Resource": "*" }, { "Effect": "Allow",
"Action": "iam:PassRole", "Resource": "*" }
] }
```

Fig. 18: Setup connection with AWS



Config access to provider

Configure yours connections,
you able to use multiple connections with single datacenter.

OPENSTACK

Datacenter: Select Datacenter

Regions: Select Regions

API Version: Select API Version

Auth URL*: https://keystone.br-maestro-server.com

Its a keystone url

Username*: sig

Password*: ••••••

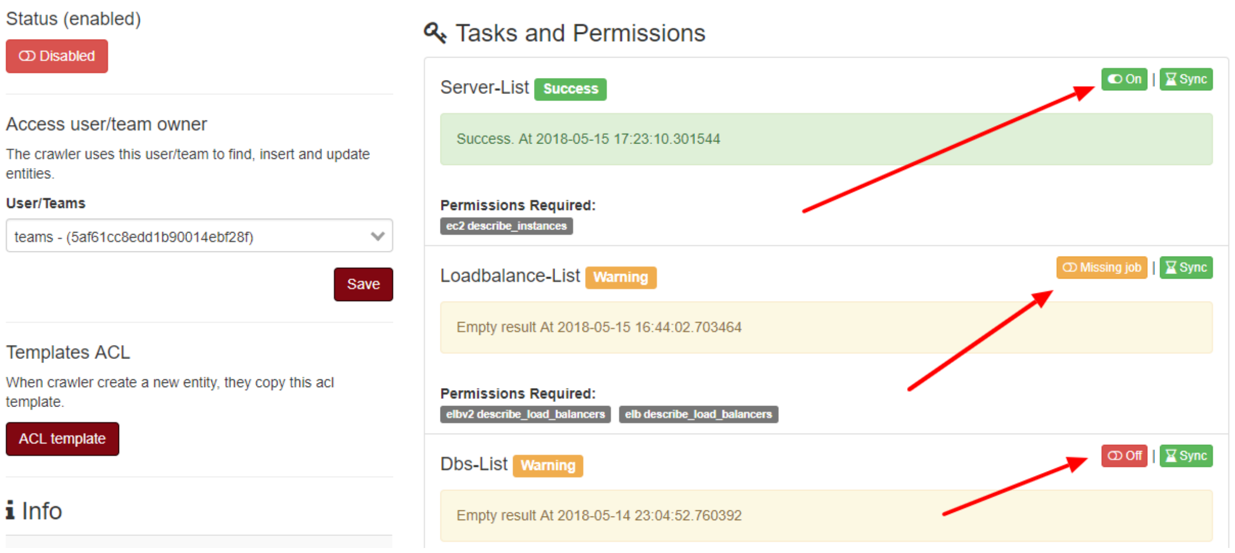
Project Id*: 33aa1afc-03fe-43b8-8201-4e0d3b4b8a

User Domain Id*: default

Hit: All information may be found in Access & Security page inside of GUI OpenStack

Fig. 19: Setupconnection with OpenStack

**Note:** 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.


Status (enabled)

Disabled

Access user/team owner

The crawler uses this user/team to find, insert and update entities.

User/Teams: teams - (5af61cc8edd1b90014ebf28f)

Save

Templates ACL

When crawler create a new entity, they copy this acl template.

ACL template

Info

Tasks and Permissions

Server-List Success — On | Sync

Success. At 2018-05-15 17:23:10.301544

Permissions Required: ec2 describe_instances

Loadbalance-List Warning — Missing job | Sync

Empty result At 2018-05-15 16:44:02.703464

Permissions Required: elbv2 describe_load_balancers, elb describe_load_balancers

Dbs-List Warning — Off | Sync

Empty result At 2018-05-14 23:04:52.760392

Fig. 20: 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.

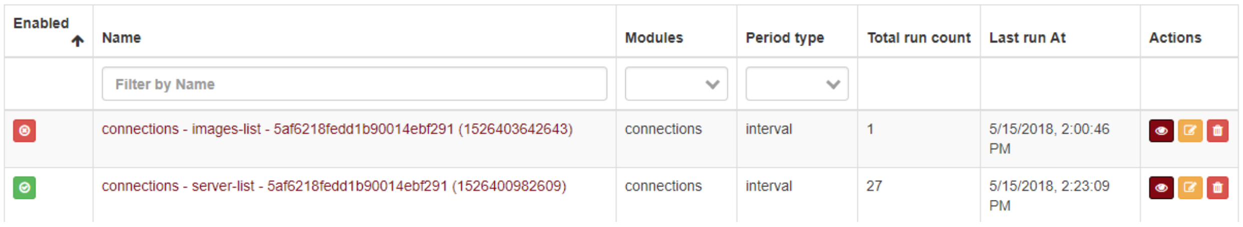
For debbug, use stdout docker, like

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

## 4.1.3 Scheduler

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

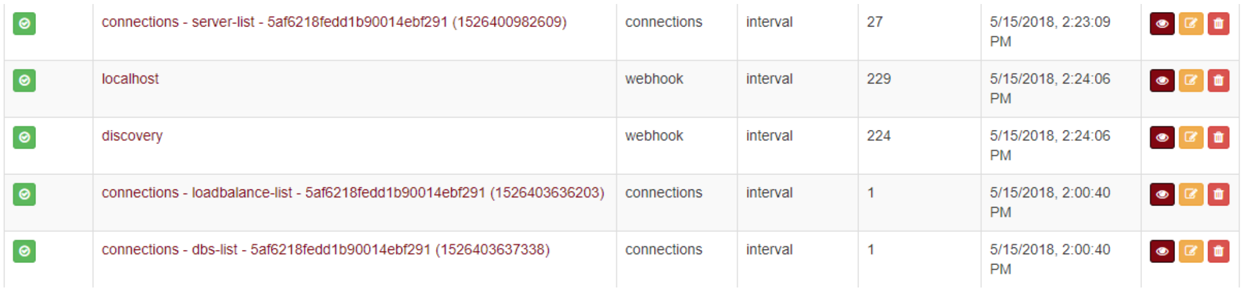
You can list all schedules, the first column show if that schedule is enabled or disabled.



| 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 | |

Fig. 21: 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 | |

Fig. 22: Total counts

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

### 4.1.4 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.

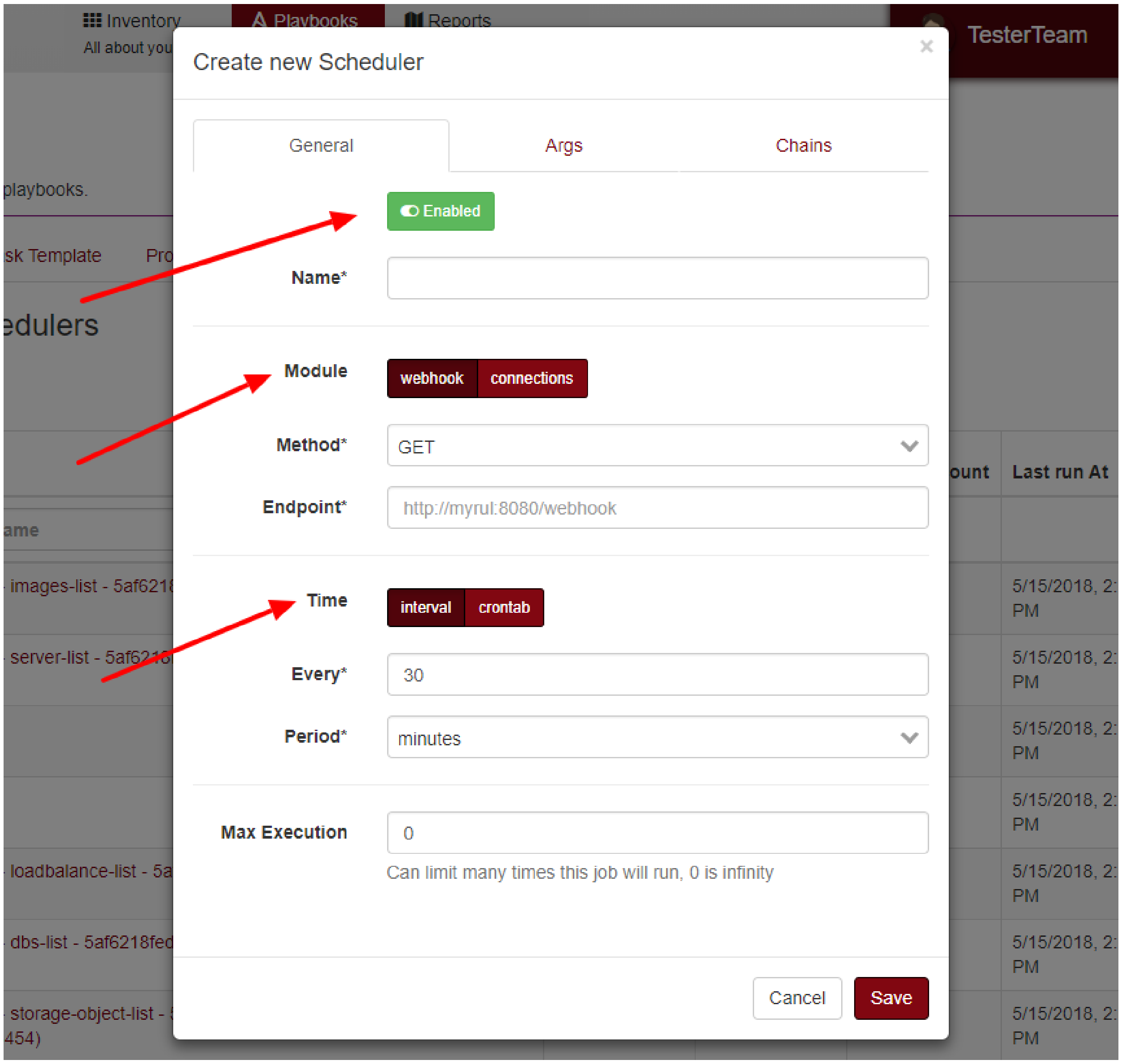
You can add new members and config each access role.

### 4.1.5 Access and Auth

All entitites in Maestro has access roles, the access role

#### Account / Profile

On profile you can update your profile.



Create new Scheduler

General | Args | Chains

Enabled

Name*

Module: webhook | connections

Method*: GET

Endpoint*: http://myrul:8080/webhook

Time: interval | crontab

Every*: 30

Period*: minutes

Max Execution: 0

Can limit many times this job will run, 0 is infinity

Cancel | Save

Fig. 23: Creating

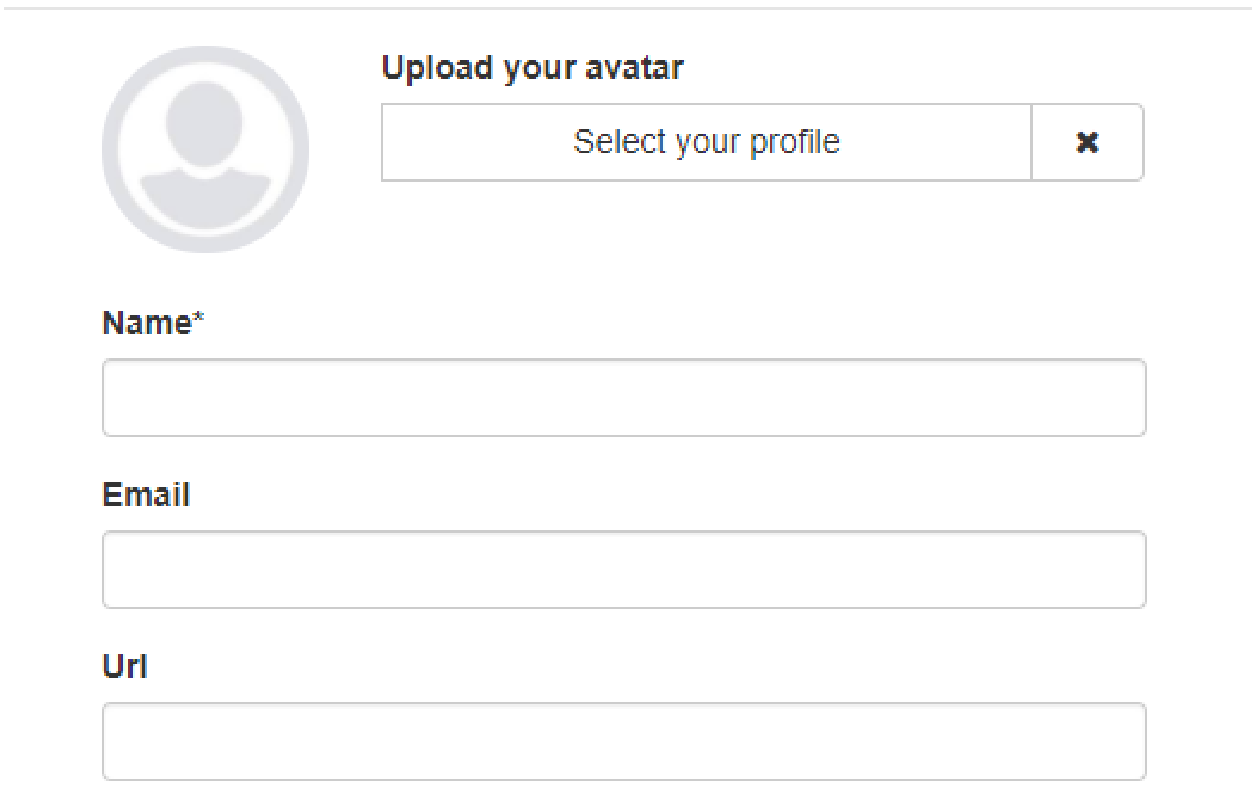
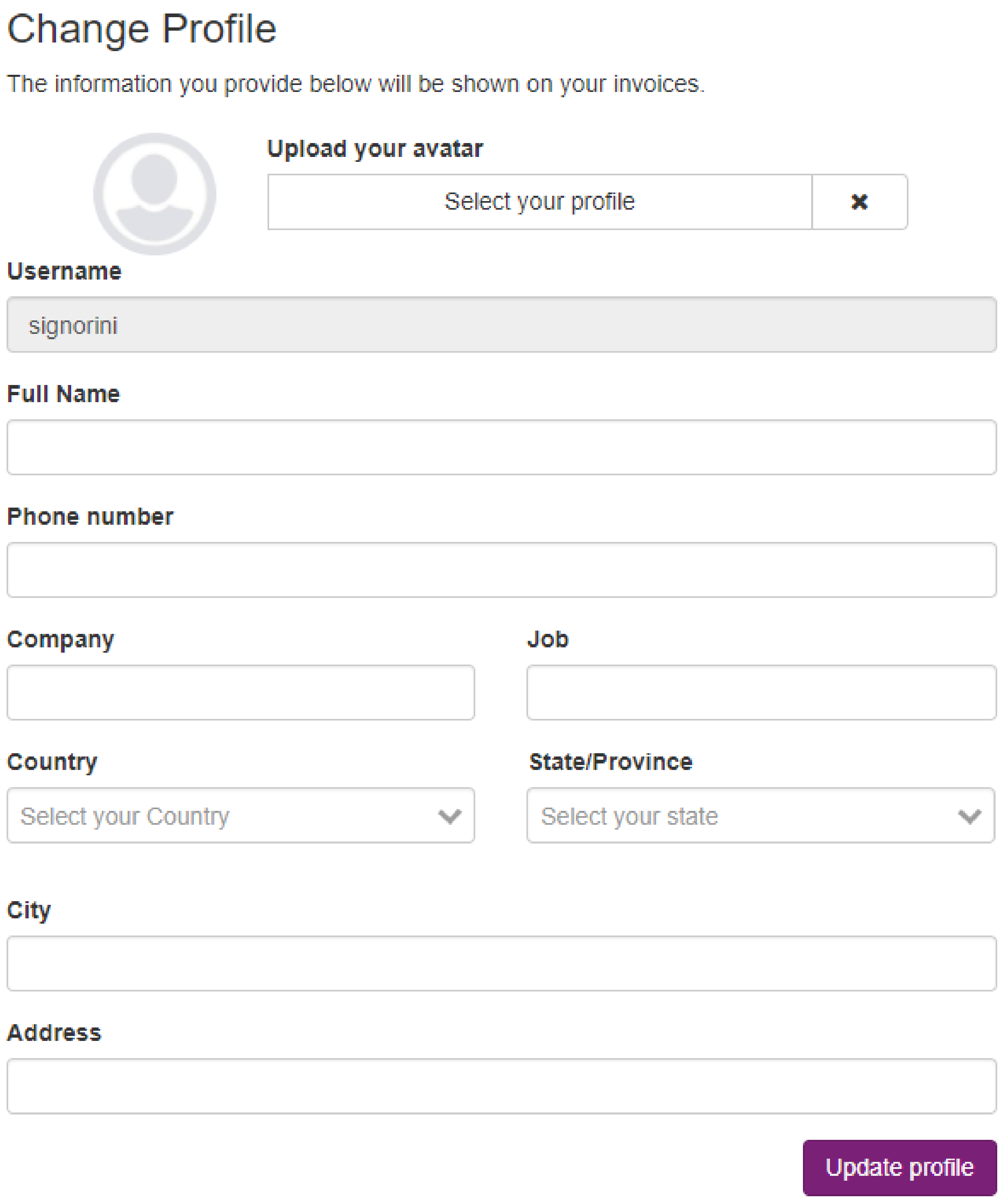


Fig. 24: Form team


Change Profile

The information you provide below will be shown on your invoices.

Username: signorini

Full Name:

Phone number:

Company:

Job:

Country: Select your Country

State/Province: Select your state

City:

Address:

Fig. 25: 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. |

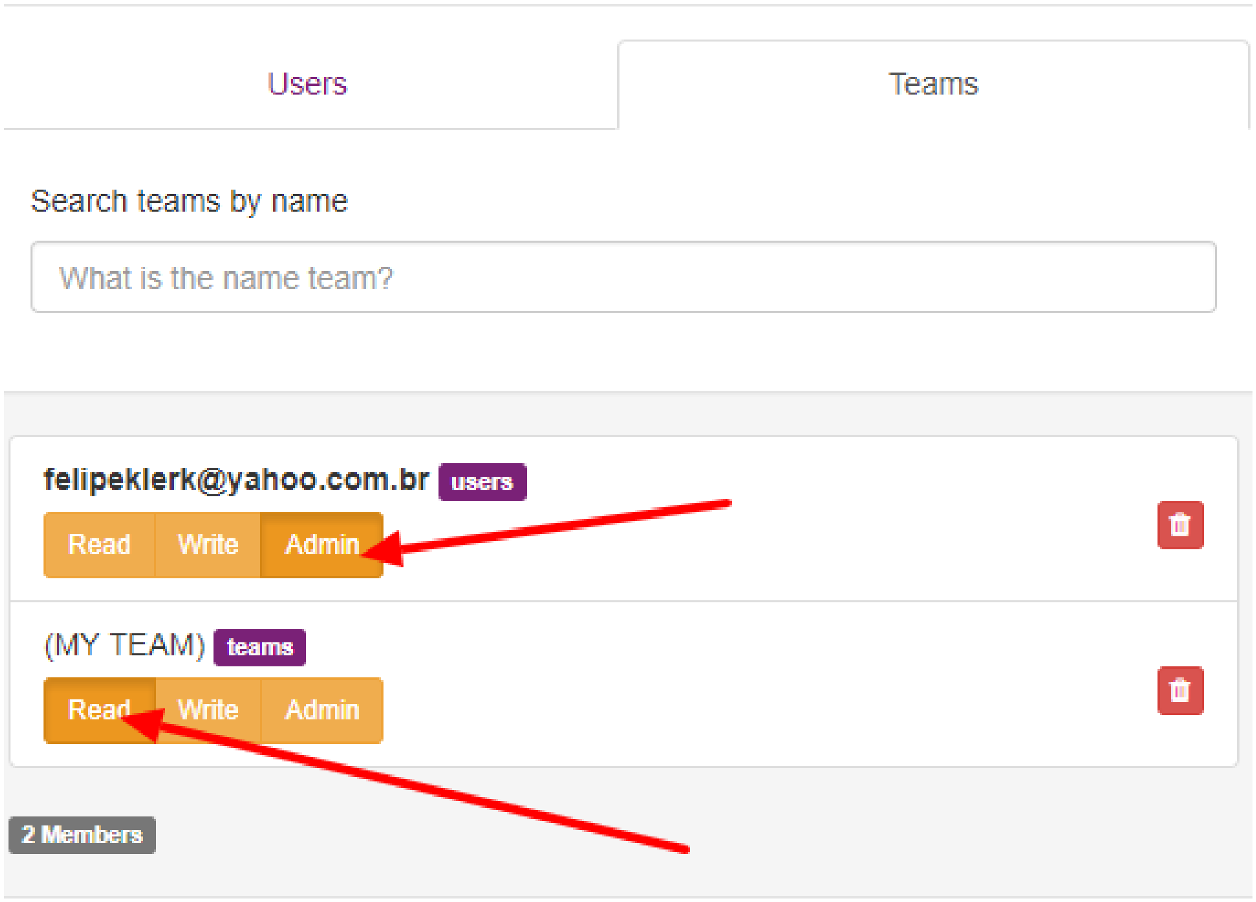
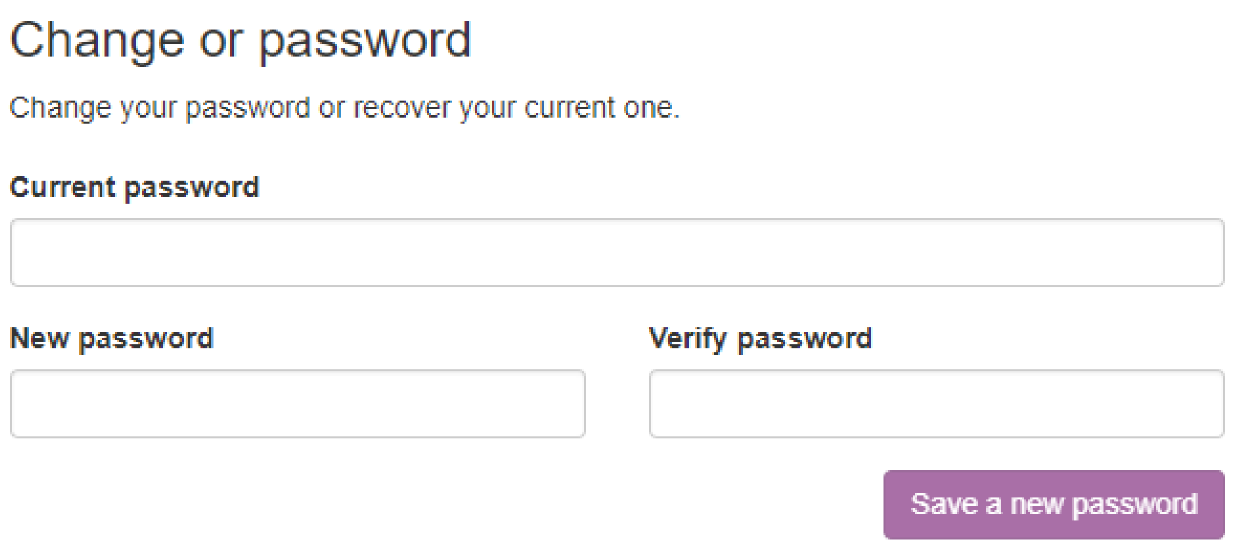


Fig. 26: You can change any acl point for specify user or team.

### 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

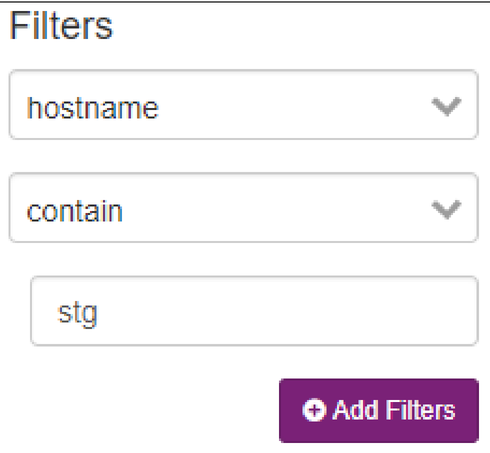
Verify password

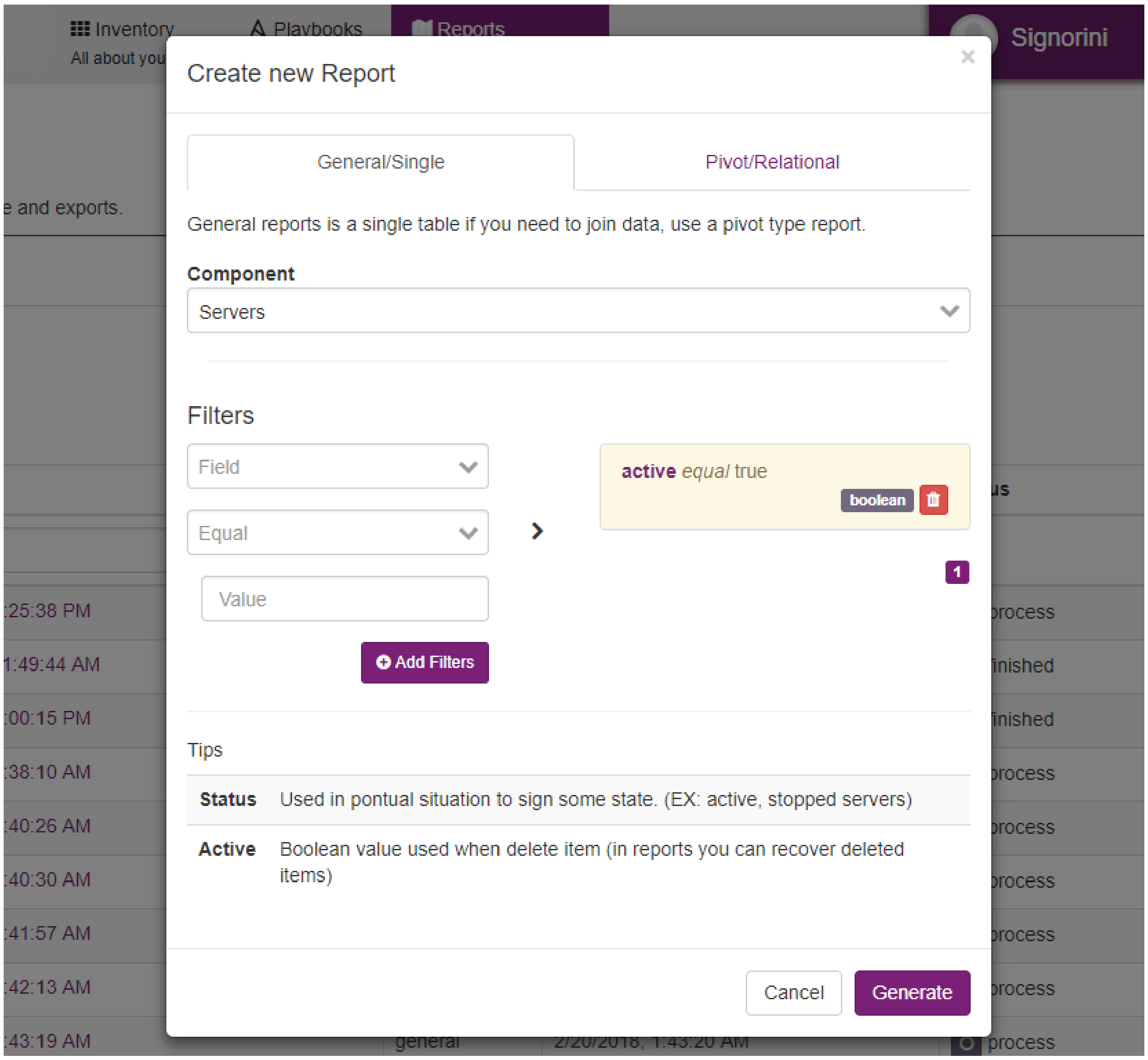
Fig. 27: New pass

### 4.1.6 Reports

#### Single table report

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

| | | | | |
|---|---|---|---|---|
| Host-name/name | string | equal/contains | Name or host-name |  Filters<br>hostname<br>contain<br>stg<br> |
| Ex: Hostname contais stg, will find somethiing like webserver.stg or stg2. | | | | |



General reports is a single table if you need to join data, use a pivot type report.

Component: Servers

Filters

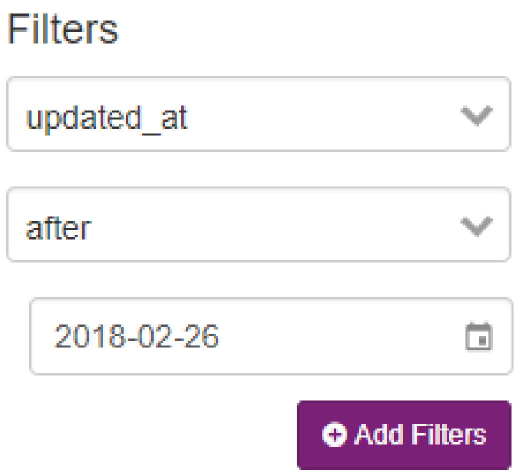
active *equal* true boolean

1

Tips

| | |
|---|---|
| **Status** | Used in pontual situation to sign some state. (EX: active, stopped servers) |
| **Active** | Boolean value used when delete item (in reports you can recover deleted items) |

Fig. 28: Single report, use only one entity

| | | | | |
|---|---|---|---|---|
| Updated_at | date | after/equal/before | Last data update |  |
| 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.

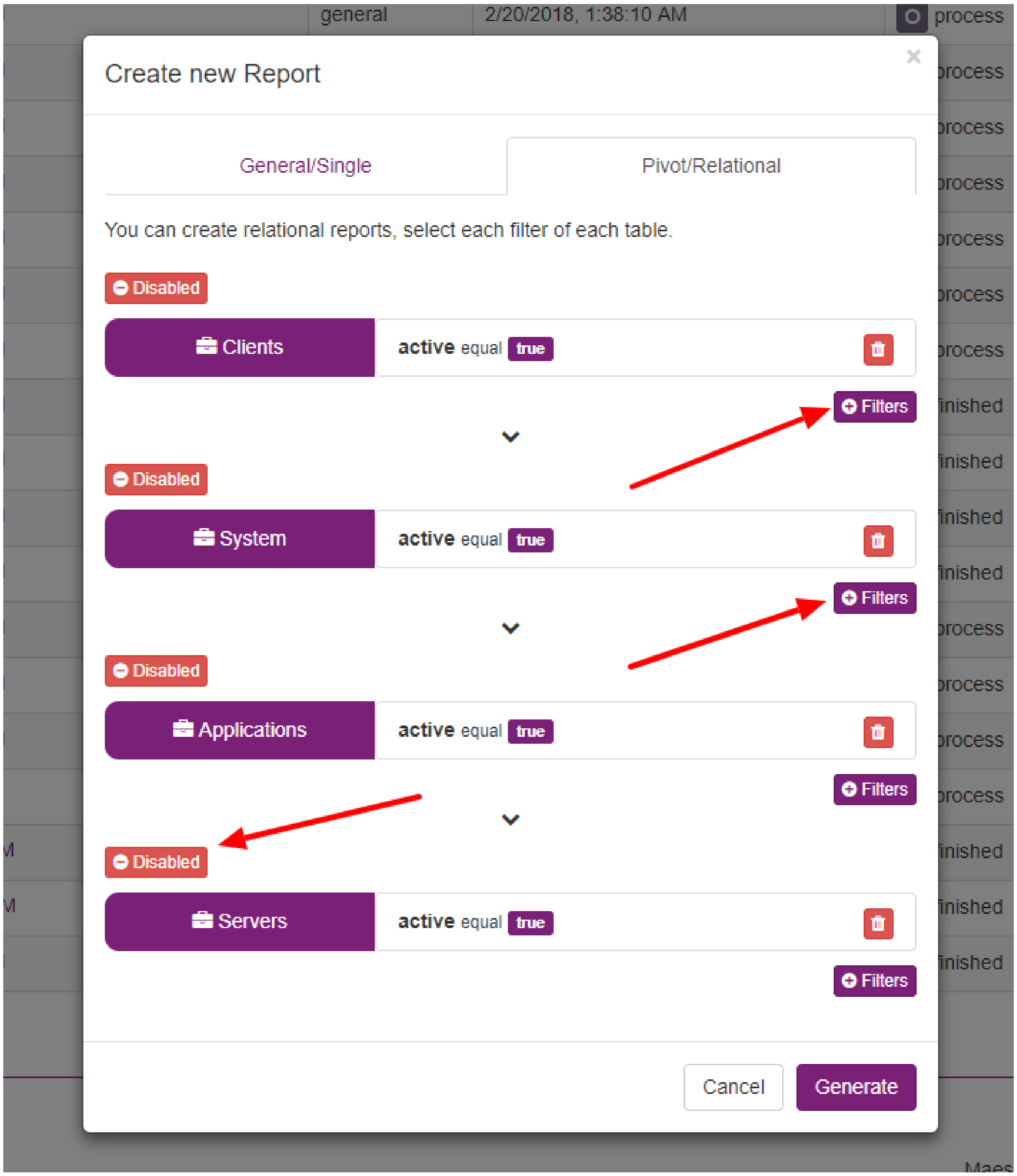
## 4.1.7 Config / Settings

### Services

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

### Config Options

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


Create new Report

General/Single | Pivot/Relational

You can create relational reports, select each filter of each table.

Fig. 29: Nesting relation entities.

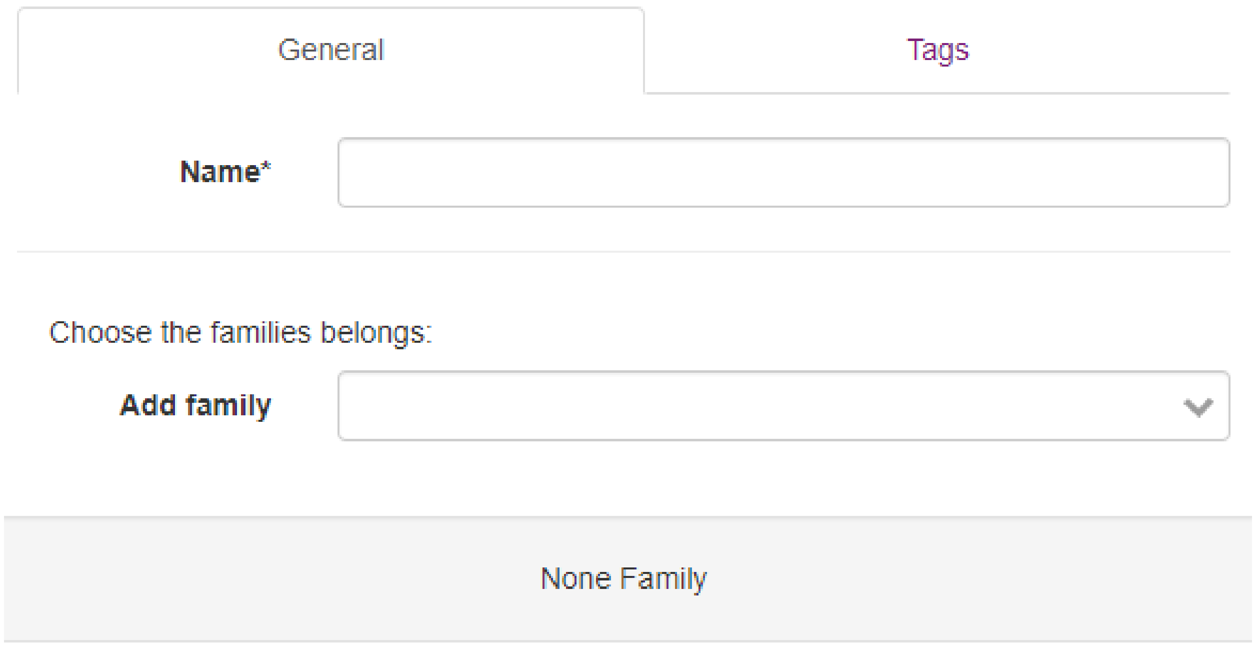


Fig. 30: Register a new, or update service.

| | |
|---|---|
| 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 | |

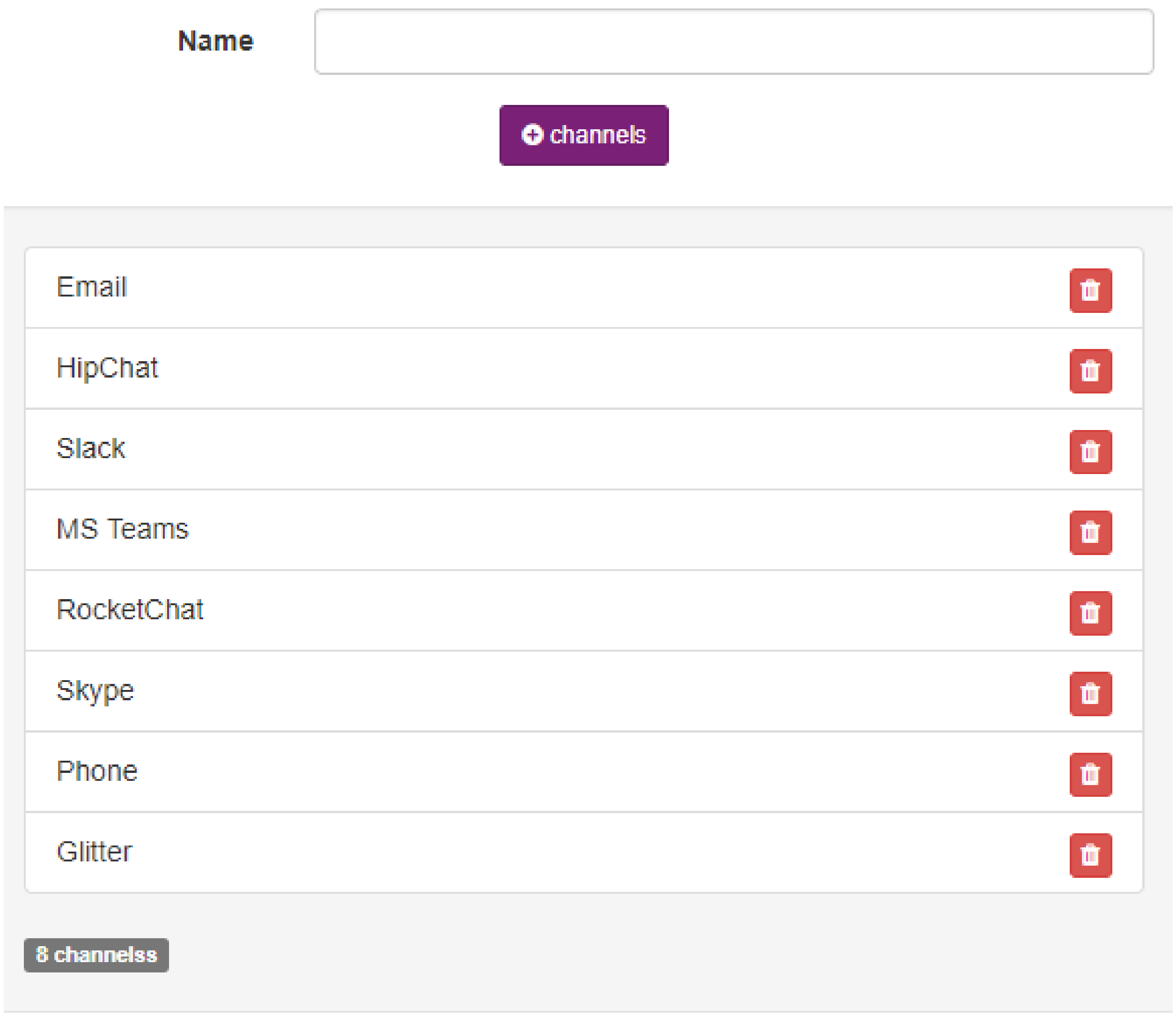
---

---

### Regions and zones

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

---



Email

HipChat

Slack

MS Teams

RocketChat

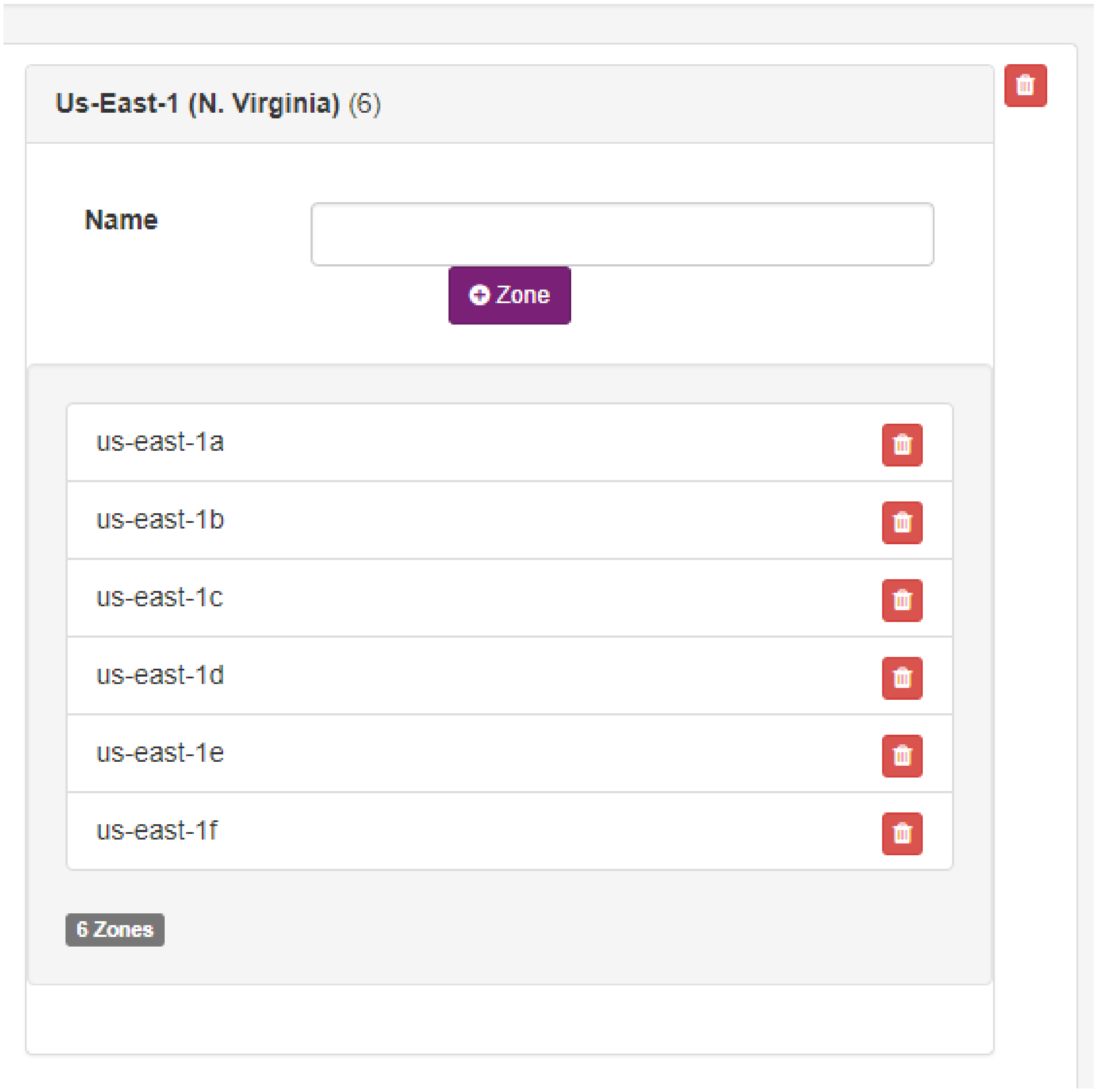
Skype

Phone

Glitter

8 channelss

Fig. 31: Maestro configs, created when run migration command.


Us-East-1 (N. Virginia) (6)

Name

us-east-1a
us-east-1b
us-east-1c
us-east-1d
us-east-1e
us-east-1f

Fig. 32: Pre-define regions and zones.

# CHAPTER 5

# Developer Guide

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

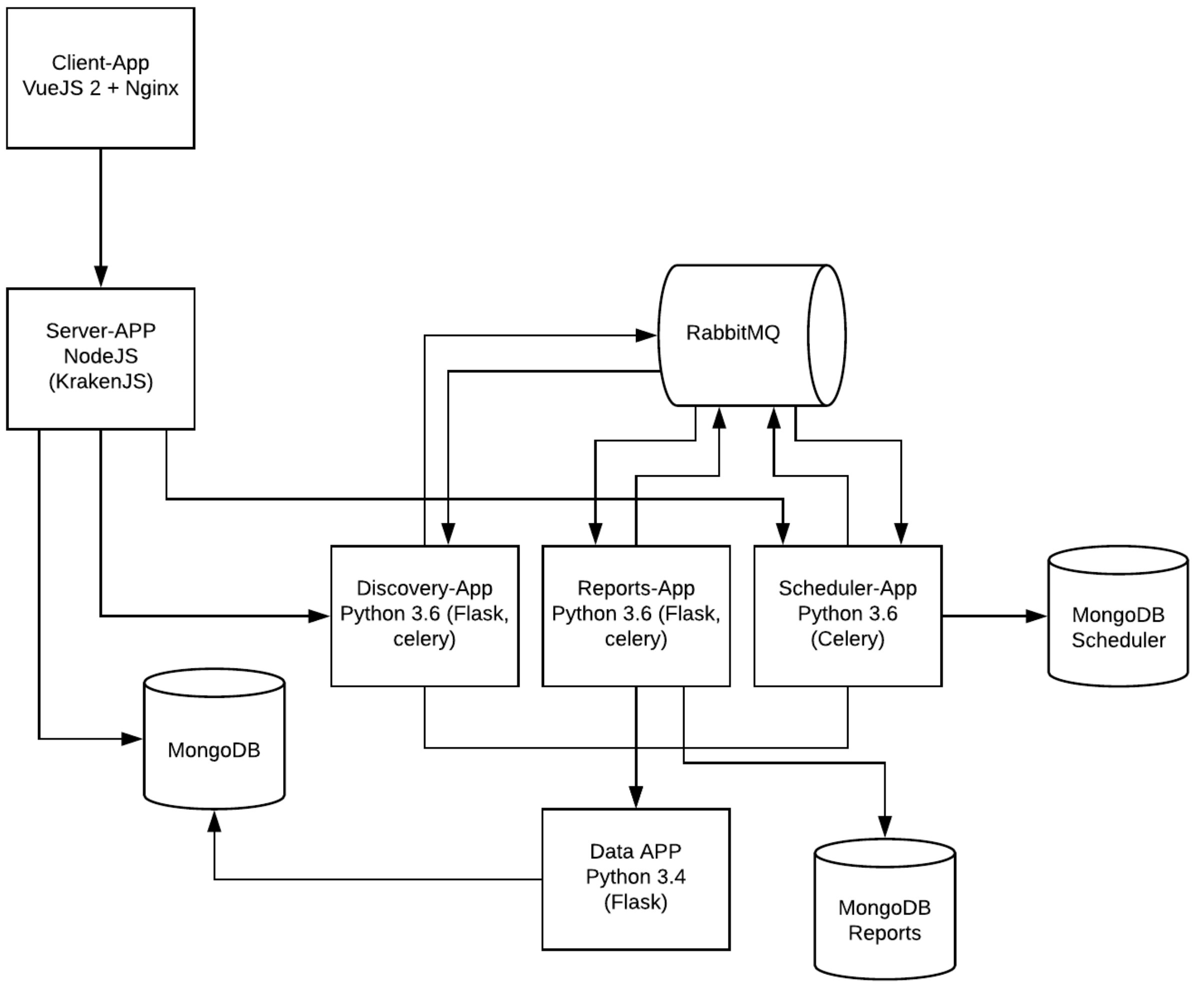
## 5.1 Architecture

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

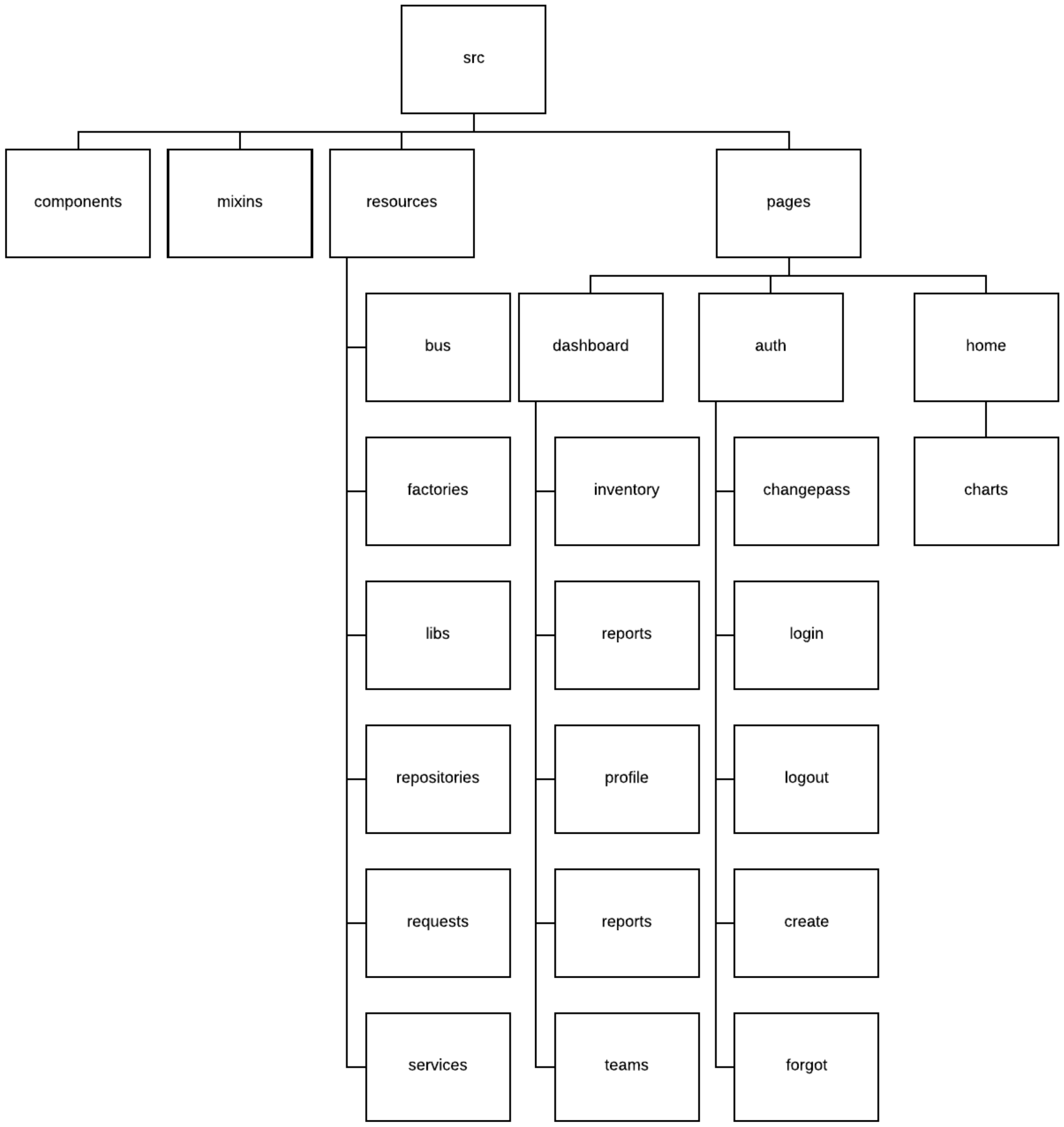
Client App front end application

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

**Warning:** 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**

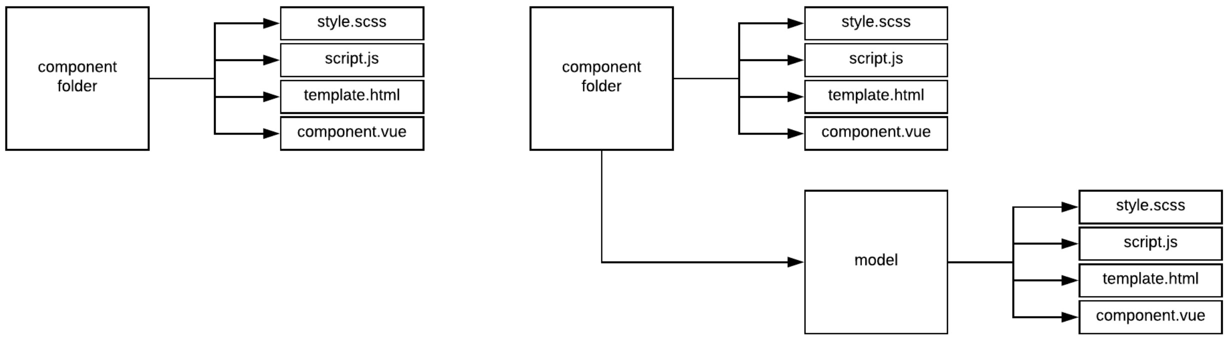


### Important topics

- 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 |
| LOGO | /static/imgs/logo300.png | Logotype, (login page) |
| THEME | theme-lotus | Theme (gold\|wine\|blue\|green\|dark) |

### 5.1.2 Server App

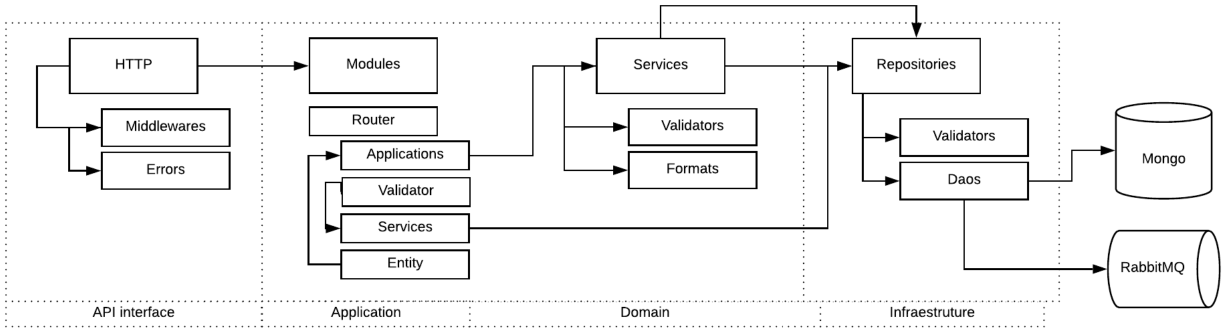
Server App main application, some responsibility is

- Authentication and authorization
- Validate and create entities (crud ops)
- Proxy to others services

**Warning:** 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 6 or 7
- 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-client'
```

and

```
npm run server
```

---

### 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
```

---

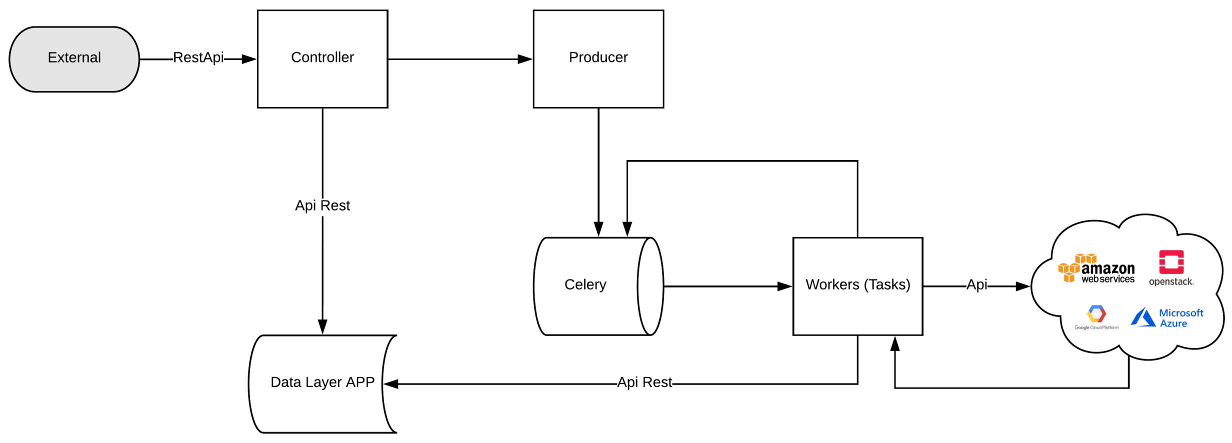
### Env variables

| Env Variables | Example | Description |
|---|---|---|
| MAESTRO_PORT | 8888 | |
| NODE_ENV | development\|production | |
| MAESTRO_MONGO_URI | localhost/maestro-client | DB string connection |
| MAESTRO_SECRETJWT | XXXX | Secret key - session |
| MAESTRO_SECRETJWT_FORGOT | XXXX | Secret key - forgot request |
| MAESTRO_SECRET_CRYPTO_FORGOT | XXXX | Secret key - forgot content |
| MAESTRO_DISCOVERY_URL | http://localhost:5000 | Url discovery-app (flask) |
| MAESTRO_REPORT_URL | http://localhost:5005 | Url reports-app (flask) |
| 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

    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 discovery --hostname=discovery@%h --loglevel=info

or

npm run celery
```

---

**Warning:** 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_DATA_URI | http://localhost:5010 | Data Layer API URL |
| MAESTRO_SECRETJWT | xxxx | Same that Server App |
| 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 |

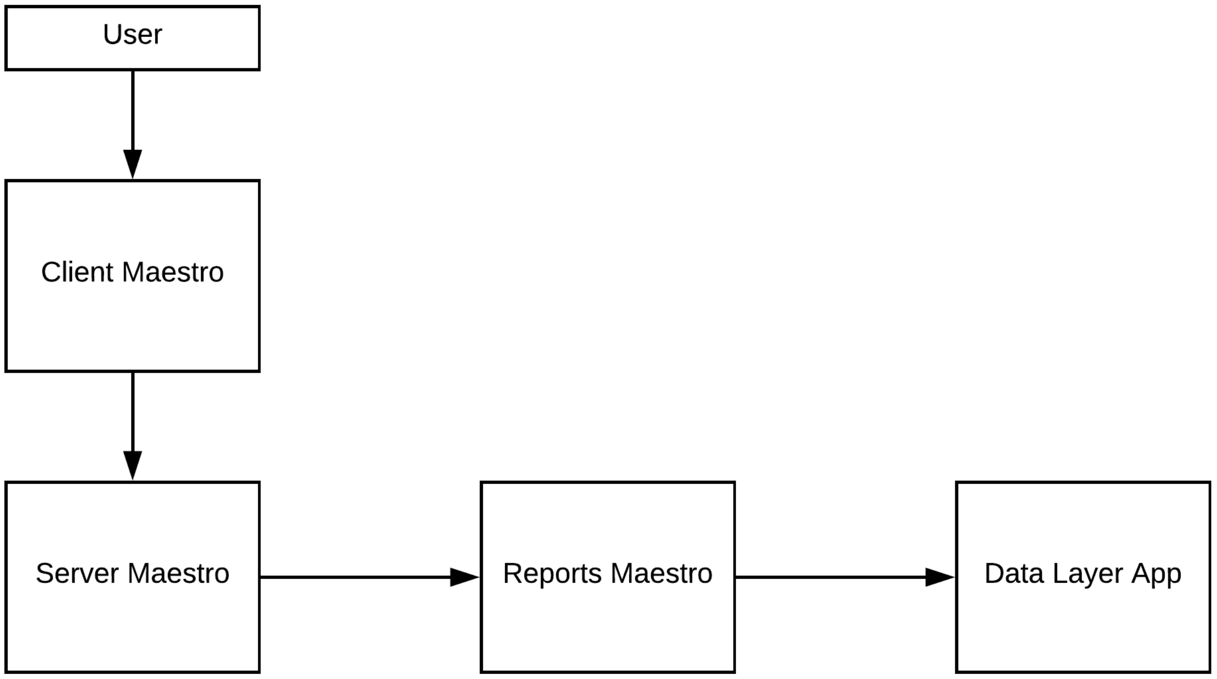
### 5.1.4 Reports App

Reports app, generate reports

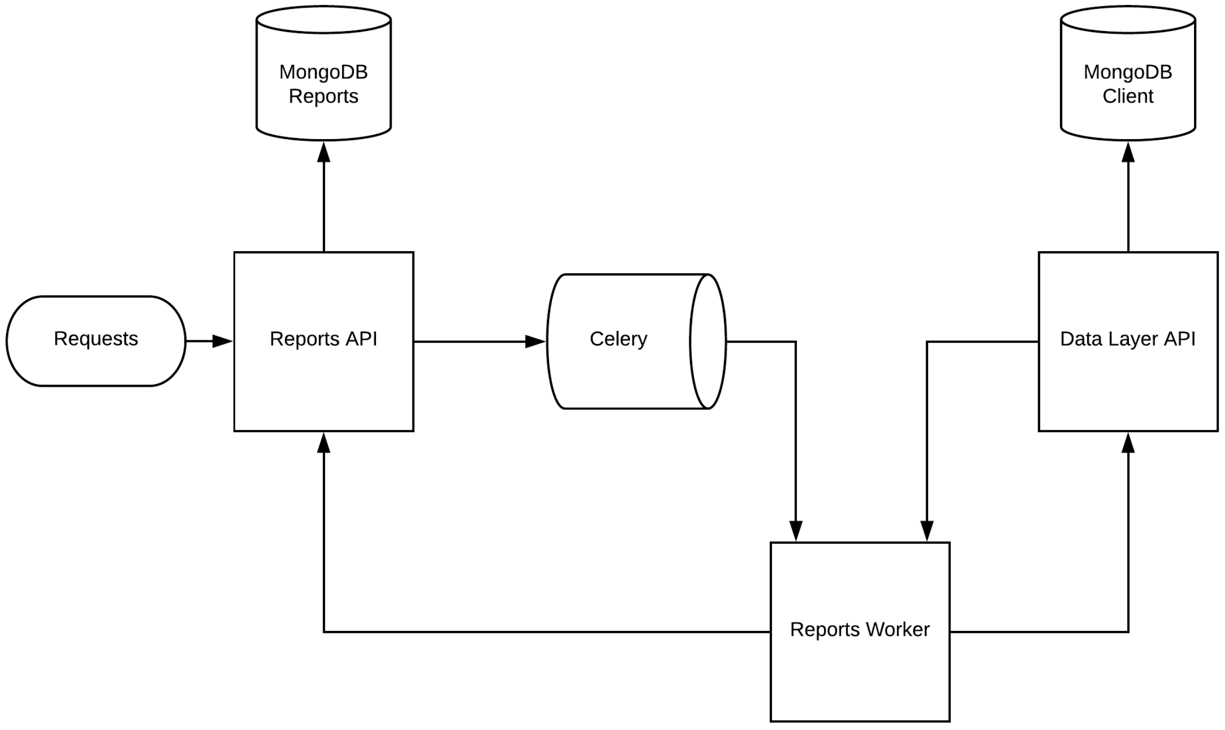
- Understand complex queries and generate reports

- 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 in 4 parts
  - **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 an y endpoint

---

#### 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
```

---

**Warning:** 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**

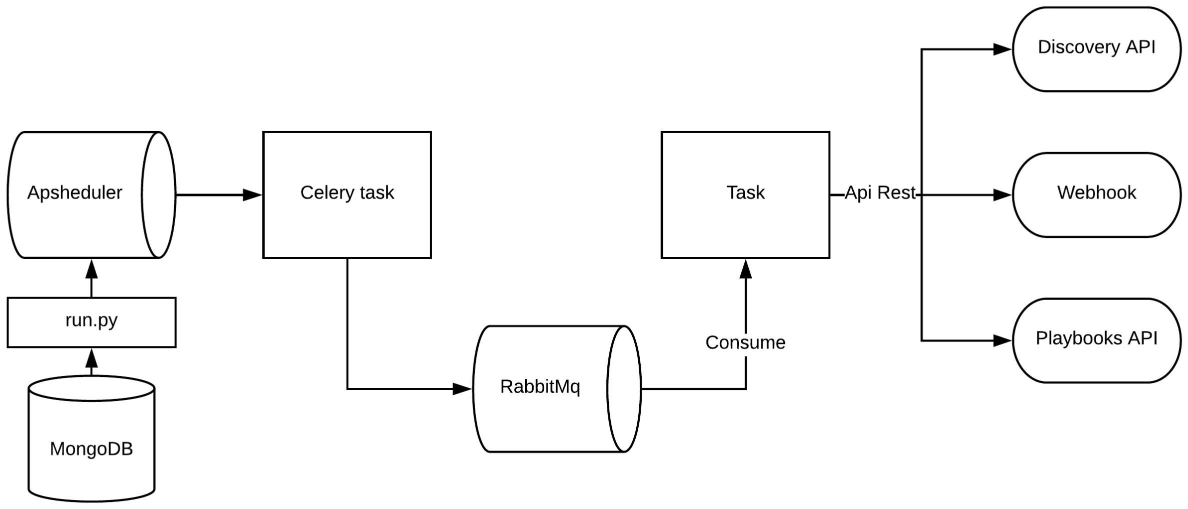
| Env Variables | Example | Description |
|---|---|---|
| MAESTRO_MONGO_URI | localhost | Mongo Url conn |
| MAESTRO_MONGO_DATABASE | maestro-reports | Db name, its differente of servers-app |
| MAESTRO_DATA_URI | http://localhost:5010 | Data layer api |
| MAESTRO_REPORT_URI | http://localhost:5005 | Report api |
| MAESTRO_REPORT_RESULT_QTD | 200 | Limit default |
| MAESTRO_TIMEOUT_DATA | 10 | Timeout for data retrived |
| MAESTRO_TIMEOUT_WEBHOOK | 5 | Timeout for notifications |
| 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
```

---

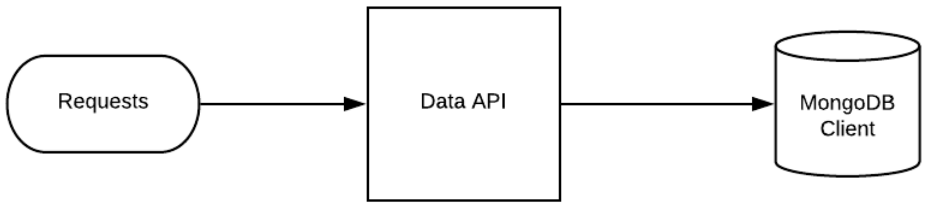
**Env variables**

| Env Variables | Example | Description |
|---|---|---|
| MAESTRO_DATA_URI | http://data:5000 | Data Layer API URL |
| MAESTRO_MONGO_URI | localhost | MongoDB URI |
| MAESTRO_MONGO_DATABASE | maestro-client | Mongo Database name |
| CELERY_BROKER_URL | amqp://rabbitmq:5672 | RabbitMQ connection |

### 5.1.6 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
```

---

**Warning:** 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 | 5000 | Port used |
| MAESTRO_MONGO_URI | localhost | Mongo Url conn |
| MAESTRO_MONGO_DATABASE | maestro-reports | Db name, its differente of servers-app |
| MAESTRO_GWORKERS | 2 | Gunicorn multi process |
| MAESTRO_INSERT_QTD | 200 | Throughput insert in reports collection |

## 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

Primary api, encharge by autentication, create and retrive data. See docs server api.

### 5.2.2 Discovery API

Crawler and connection with api providers. See docs discovery api.

### 5.2.3 Data Layer API

Data Layer API. See docs data layer api.

## 5.3 Lints

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

### 5.3.1 JavaScript (Client App)

Use eslint, default vue-loader

```
npm run lint
```

---

### 5.3.2 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)
    "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)
```

(continues on next page)

(continued from previous page)

```
    "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)
}
```

### 5.3.3 Python 3 (Discovery, Scheduler and Reports)

pytlint, default config.

```
npm run lint
```

## 5.4 Tests

Each service need to be testing.

### 5.4.1 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

### 5.4.2 Discovery APP

Testing use pytest

```
npm run test

python -m unittest discover
```

### 5.4.3 Reports APP

Testing use pytest

```
npm run test

python -m unittest discover
```

### 5.4.4 Data Layer APP

Testing use pytest

```
npm run test

python -m unittest discover
```

## 5.5 Quality Assurance

We use some tools to mensure quality.

**Note:** Travis with all projects

---

### 5.5.1 Client Maestro

| | |
|---|---|
| Codacy | |
| Travis | |
| CodeClimate | |

---

### 5.5.2 Server App

| | |
|---|---|
| CodeClimate | |
| Travis | |
| DavidDm | |
| Codacy | |
| Coveralls | |

---

### 5.5.3 Discovery Maestro

| | |
|---|---|
| Codacy | |
| Travis | |
| CodeClimate | |

---

### 5.5.4 Report Maestro

| | |
|---|---|
| Codacy | |
| Travis | |
| CodeClimate | |

---

### 5.5.5 Scheduler Maestro

| | |
|---|---|
| Codacy | |
| Travis | |
| CodeClimate | |

---

### 5.5.6 Data Layer API

| | |
|---|---|
| Codacy | |
| Travis | |
| CodeClimate | |

## 5.6 Versions

Microservices compatible versions

---

### 5.6.1 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 |

---

### 5.6.2 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 |

# CHAPTER 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.

**Note:** All contribuition will be accept by Pull Request

# CHAPTER 7

# 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.