demo#1
RETHINKDB_ DEMONSTRATOR STARTUP & CLUSTER CREATION

0) creare 3 nodi (macchine virtuali) con una distro linux (es. Ubuntu) →
(es. su proxmox http://virt11:8006) ed **accenderli**

es: tre macchine virtuali:
amasdb4 192.156.213.164 (user: cedadm → pwd: XXXXXX)
amasdb5 192.156.213.165 (user: cedadm → pwd: XXXXXX)
amasdb6 192.156.213.166 (user: cedadm → pwd: XXXXXX)

1) Installazione da git Rethinkdb

git clone https://github.com/rethinkdb/rethinkdb.git

```
Cloning into 'rethinkdb'...
remote: Enumerating objects: 290857, done.
remote: Counting objects: 100% (138/138), done.
remote: Compressing objects: 100% (120/120), done.
remote: Total 290857 (delta 38), reused 18 (delta 18), pack-reused 290719
(from 2)
Receiving objects: 100% (290857/290857), 134.18 MiB | 38.77 MiB/s, done.
Resolving deltas: 100% (224139/224139), done.
root@grid005:~# git checkout v2.4.4
fatal: not a git repository (or any of the parent directories): .git
```

cd rethinkdb/
PYTHON=/usr/bin/python3 ./configure --allow-fetch --prefix=/usr

```
* Detecting system configuration
Bash:                          5.2.21(1)-release
Use ccache:                    no
C++ Compiler:                  GCC 13.3.0 (/usr/bin/c++)
Host System:                   x86_64-linux-gnu
Build System:                  no
Operating System:              Linux
OS Version:                    Linux 6.8.0-90-generic x86_64
Cross-compiling:               no
Host Operating System:         Linux
Build Architecture:            x86_64
C++11:                         ok
Protobuf compiler:             external/protobuf_3.19.4
python:                        python 3.12.3
wget:                          /usr/bin/wget
curl:                          /usr/bin/curl
Google Test:                   external/gtest_1.8.1
termcap:                       no
boost_system:                  external/boost_1.60.0
Protobuf library:              external/protobuf_3.19.4
RE2:                           external/re2_2015-11-01
z:                             -lz
crypto:                        -lcrypto
```

```
ssl:                              -lssl
curl:                             external/curl_7.82.0
quickjs:
external/quickjs_95d6dcf6358a74e9cbe04eade7c383f43ba306cb
malloc:                           jemalloc
jemalloc (static):                external/jemalloc_5.2.1
Test protobuf:                    external/protobuf_3.19.4
Test boost:                       external/boost_1.60.0
Installation prefix:              /usr/local
Configuration prefix:             /usr/local/etc
Runtime data prefix:              /usr/local/var
* Wrote configuration to config.mk
```

make -j4 (opzione –j e' il numero di cores qui abbiamo macchine virtuali con 6 cores)

```
    [1/447] FETCH protobuf_3.19.4
    [2/447] FETCH gtest_1.8.1
    [3/447] FETCH re2_2015-11-01
    [4/447] FETCH boost_1.60.0
      ……
      ……
    [446/447] CC build/release/obj/rpc/connectivity/server_id.o
-L/root/rethinkdb/build/external/libidn_1.38/lib -
L/root/rethinkdb/build/external/zlib_1.2.11/lib ||||| -lzstd
Warning: 'curl' links with '-lzstd'
    [447/447] LD build/release/rethinkdb
```

make install

```
    [1/9] BUILD quickjs_95d6dcf6358a74e9cbe04eade7c383f43ba306cb
    [2/9] INSTALL build/release/rethinkdb /usr/local/bin
    [3/9] M4 build/release/assets/rethinkdb.1
    [4/9] GZIP build/release/assets/rethinkdb.1.gz
    [5/9] INSTALL build/release/assets/rethinkdb.1.gz
/usr/local/share/man/man1
    [6/9] INSTALL ./packaging/assets/docs/LICENSE
/usr/local/share/doc/rethinkdb/copyright
    [7/9] INSTALL ./packaging/assets/init/rethinkdb /usr/local/etc/init.d
    [8/9] INSTALL /usr/local/var/lib/rethinkdb/instances.d
    [9/9] INSTALL /usr/local/etc/rethinkdb/default.conf.sample
```

2) Creazione utente di servizio

sudo useradd -r -s /usr/sbin/nologin rethinkdb

3) Creazione layout directory e Setup Permessi

```
sudo mkdir -p /var/lib/rethinkdb/instance1/data
sudo mkdir -p /etc/rethinkdb/instances.d
sudo mkdir -p /var/run/rethinkdb/instance1
sudo mkdir -p /var/log/rethinkdb

sudo chown -R rethinkdb:rethinkdb \
  /var/lib/rethinkdb \
```

```
  /var/run/rethinkdb \
  /var/log/rethinkdb

sudo chmod -R 750 /var/lib/rethinkdb
sudo chmod -R 755 /var/run/rethinkdb
```

4) Inizializzazione database

```
sudo -u rethinkdb rethinkdb create \
  -d /var/lib/rethinkdb/instance1/data
```

5) Creazione del file di configurazione istanza

sudo emacs /etc/rethinkdb/instances.d/instance1.conf

```
=========
      runuser=rethinkdb
      rungroup=rethinkdb

      directory=/var/lib/rethinkdb/instance1/data

      #pid-file=/var/run/rethinkdb/instance1/pid_file

      log-file=/var/log/rethinkdb/instance1.log

      bind=all
      driver-port=28015
      cluster-port=29015
      cores=4
      cache-size=4096
      io-threads=64
      http-port=8080
      ## SSL config ##
      #http-tls-key=/etc/rethinkdb/cert/newrethinkdb.key
      #http-tls-cert=/etc/rethinkdb/cert/newrethinkdb.crt
      #server-name=instance1

=========
```

6) Avvio daemone

```
/usr/bin/rethinkdb --daemon -config-file
/etc/rethinkdb/instances.d/instance1.conf
```

oppure
/usr/bin/rethinkdb --daemon --config-file /etc/rethinkdb/instances.d/instance1.conf --runuser rethinkdb --rungroup rethinkdb --directory /var/lib/rethinkdb/instance1/data

7) crea script di installazione:

emacs install_rethinkdb_layout.sh

```
=========
```

```
#!/bin/bash
#installa dipendenze
sudo apt-get install gcc g++ make bzip2 libssl-dev pkg-config m4

#installa rethinkd
git clone https://github.com/rethinkdb/rethinkdb.git
cd rethinkdb/
PYTHON=/usr/bin/python3 ./configure --allow-fetch -prefix=/usr
make -j4
make install
#aggiungi utente di sistema
useradd -r -s /usr/sbin/nologin rethinkdb 2>/dev/null

#paths & permissions
mkdir -p /var/lib/rethinkdb/instance1/data
mkdir -p /etc/rethinkdb/instances.d
mkdir -p /var/run/rethinkdb/instance1
mkdir -p /var/log/rethinkdb

chown -R rethinkdb:rethinkdb \
/var/lib/rethinkdb \
/var/run/rethinkdb \
/var/log/rethinkdb

#crea file instance1.conf
cat <<'EOF' > /etc/rethinkdb/instances.d/instance1.conf
runuser=rethinkdb
rungroup=rethinkdb

directory=/var/lib/rethinkdb/instance1/data
#pid-file=/var/run/rethinkdb/instance1/pid_file
log-file=/var/log/rethinkdb/instance1.log

bind=all
driver-port=28015
cluster-port=29015
#join=192.156.213.164:29015  #per gli altri nodi (non il master)
cores=4
cache-size=4096
io-threads=64
http-port=8080
## SSL config ##
#http-tls-key=/etc/rethinkdb/cert/newrethinkdb.key
#http-tls-cert=/etc/rethinkdb/cert/newrethinkdb.crt
#server-name=instance1
EOF

#inizializzione db
sudo -u rethinkdb rethinkdb create \
-d /var/lib/rethinkdb/instance1/data || true
```

#start the daemon

usr/bin/rethinkdb --daemon --config-file /etc/rethinkdb/instances.d/instance1.conf --runuser rethinkdb --rungroup rethinkdb --directory /var/lib/rethinkdb/instance1/data

==========

ps aux | grep rethinkdb

```
rethink+ 1655498  0.0  0.0 323488 26116 ?        Ssl  17:52   0:00
/usr/bin/rethinkdb --daemon --config-file
/etc/rethinkdb/instances.d/instance1.conf --runuser rethinkdb -rungroup
rethinkdb --pid-file /var/run/rethinkdb/instance1/pid_file --directory
/var/lib/rethinkdb/instance1/data
rethink+ 1655499  0.0  0.0  67428 16004 ?        S    17:52   0:00
/usr/bin/rethinkdb --daemon --config-file
/etc/rethinkdb/instances.d/instance1.conf --runuser rethinkdb --rungroup
rethinkdb --pid-file /var/run/rethinkdb/instance1/pid_file --directory
/var/lib/rethinkdb/instance1/data
```

8) Visualizzazione console admin (browser)

http://localhost:8080

Se esegui da remoto (immettere IP completo) http://192.156.213.164:8080

9) Creazione del Cluster

fatto questo:
sul nodo4 →

1) sudo /etc/init.d/rethinkdb stop
2) rethinkdb --bind all

sul nodo5 →

1) sudo /etc/init.d/rethinkdb stop
2) rethinkdb --join 192.156.213.164:29015 --bind all

sul nodo6 →

1) sudo /etc/init.d/rethinkdb stop
2) rethinkdb --join 192.156.213.164:29015 --bind all

→ i tre server si parlano in un cluster verificare sul browser: http://localhost:8080←

a quel punto→ si riavvia prima amasdb1 poi amasdb2 poi amasdb3

######SE FUNZIONA NEL TERINALE DEL NODO4
comparirà così:
cedadm@amasdb4:~$ rethinkdb --bind all
Running rethinkdb 2.4.3~0jammy (x86_64-linux-gnu) (GCC 11.3.0)...
Running on Linux 5.15.0-43-generic x86_64
Loading data from directory /home/cedadm/rethinkdb_data
Listening for intracluster connections on port 29015
Listening for client driver connections on port 28015
Listening for administrative HTTP connections on port 8080
Listening on cluster addresses: 127.0.0.1, 192.156.213.164, ::1, fe80::5054:ff:fe86:e8ad%2
Listening on driver addresses: 127.0.0.1, 192.156.213.164, ::1, fe80::5054:ff:fe86:e8ad%2
Listening on http addresses: 127.0.0.1, 192.156.213.164, ::1, fe80::5054:ff:fe86:e8ad%2

Server ready, "amasdb4_2gy" 856f2138-dbef-4dea-b9ac-120b842ddfba
Connected to server "amasdb5_btg" 7443ea6f-eae4-4b67-8e77-b006261768c3
Connected to server "amasdb6_ddx" 27030a42-6838-4680-9d14-bd3a99cca315

mentre sugli altri due nodi:
sul nodo5 →

cedadm@amasdb5:~$ rethinkdb --join 192.156.213.164:29015 --bind all
Recursively removing directory /home/cedadm/rethinkdb_data/tmp
Initializing directory /home/cedadm/rethinkdb_data
Running rethinkdb 2.4.3~0jammy (x86_64-linux-gnu) (GCC 11.3.0)...
Running on Linux 5.15.0-75-generic x86_64
Loading data from directory /home/cedadm/rethinkdb_data
Listening for intracluster connections on port 29015
Connected to server "amasdb4_2gy" 856f2138-dbef-4dea-b9ac-120b842ddfba
Listening for client driver connections on port 28015
Listening for administrative HTTP connections on port 8080
Listening on cluster addresses: 127.0.0.1, 192.156.213.165, ::1, fe80::5054:ff:fe01:83ee%2
Listening on driver addresses: 127.0.0.1, 192.156.213.165, ::1, fe80::5054:ff:fe01:83ee%2
Listening on http addresses: 127.0.0.1, 192.156.213.165, ::1, fe80::5054:ff:fe01:83ee%2
Server ready, "amasdb5_btg" 7443ea6f-eae4-4b67-8e77-b006261768c3
Connected to server "amasdb6_ddx" 27030a42-6838-4680-9d14-bd3a99cca315

e sul nodo6 →

cedadm@amasdb6:~$ rethinkdb --join 192.156.213.164:29015 --bind all
Recursively removing directory /home/cedadm/rethinkdb_data/tmp
Initializing directory /home/cedadm/rethinkdb_data
Running rethinkdb 2.4.3~0jammy (x86_64-linux-gnu) (GCC 11.3.0)...
Running on Linux 5.15.0-75-generic x86_64
Loading data from directory /home/cedadm/rethinkdb_data
Listening for intracluster connections on port 29015
Connected to server "amasdb4_2gy" 856f2138-dbef-4dea-b9ac-120b842ddfba
Listening for client driver connections on port 28015
Listening for administrative HTTP connections on port 8080
Listening on cluster addresses: 127.0.0.1, 192.156.213.166, ::1, fe80::5054:ff:fe14:a189%2
Listening on driver addresses: 127.0.0.1, 192.156.213.166, ::1, fe80::5054:ff:fe14:a189%2
Listening on http addresses: 127.0.0.1, 192.156.213.166, ::1, fe80::5054:ff:fe14:a189%2
Server ready, "amasdb6_ddx" 27030a42-6838-4680-9d14-bd3a99cca315
Connected to server "amasdb5_btg" 7443ea6f-eae4-4b67-8e77-b006261768c3

Dal browser console Admin:

→ http://amasdb4:8080
→ http://amasdb5:8080
→ http://amasdb6:8080
oppure
→ http://192.156.213.164:8080

→ http://192.156.213.165:8080
→ http://192.156.213.166:8080

10) Come automatizzare con Ansible (se ho molti nodi)?

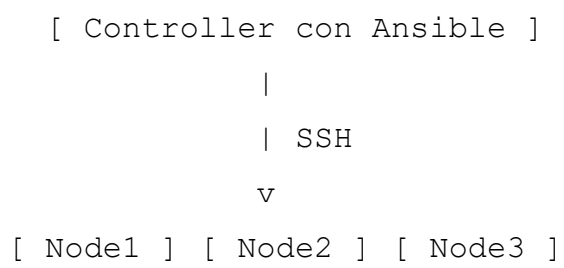
Creare una macchina virtuale/laptop dove installare Ansible:

apt update

apt install -y ansible

#verifica:

ansible –version



Per funzionare I nodi devono essere raggiungibili in SSH senza password:

Creare una directory con ils eguente contenuto:

rethinkdb-ansible/

├── inventory.ini

├── playbook.yml

└── vars.yml

I files sono: **inventory.ini** →

```
[rethinkdb]
node1 ansible_host=10.0.0.1
node2 ansible_host=10.0.0.2
node3 ansible_host=10.0.0.3
```

Parametri modificabili: **vars.yml** →

```
rethinkdb_prefix: /usr
```

```
rethinkdb_user: rethinkdb
instance_name: instance1
driver_port: 28015
cluster_port: 29015
http_port: 8080
cores: 4
cache_size: 4096
io_threads: 64
```

Infine creare il **playbook.yml** con contenuto →

```
- name: Install and configure RethinkDB
  hosts: rethinkdb
  become: true
  vars_files:
    - vars.yml

  tasks:

  - name: Install build dependencies
    apt:
      name:
        - git
        - build-essential
        - python3
        - libssl-dev
        - wget
      state: present
      update_cache: yes

  - name: Clone rethinkdb repo
    git:
      repo: https://github.com/rethinkdb/rethinkdb.git
      dest: /opt/rethinkdb
      version: master

  - name: Configure rethinkdb
    command: >
```

```
      PYTHON=/usr/bin/python3 ./configure --allow-fetch --
prefix={{ rethinkdb_prefix }}
    args:
      chdir: /opt/rethinkdb
      creates: /opt/rethinkdb/build


  - name: Build rethinkdb
    command: make -j{{ ansible_processor_vcpus }}
    args:
      chdir: /opt/rethinkdb


  - name: Install rethinkdb
    command: make install
    args:
      chdir: /opt/rethinkdb


  - name: Create rethinkdb user
    user:
      name: "{{ rethinkdb_user }}"
      system: yes
      shell: /usr/sbin/nologin


  - name: Create directories
    file:
      path: "{{ item }}"
      state: directory
      owner: "{{ rethinkdb_user }}"
      group: "{{ rethinkdb_user }}"
      mode: "0755"
    loop:
      - /var/lib/rethinkdb/{{ instance_name }}/data
      - /etc/rethinkdb/instances.d
      - /var/run/rethinkdb/{{ instance_name }}
      - /var/log/rethinkdb


  - name: Create instance config
    copy:
      dest: /etc/rethinkdb/instances.d/{{ instance_name }}.conf
```

```
    owner: root
    group: root
    mode: "0644"
    content: |
      runuser={{ rethinkdb_user }}
      rungroup={{ rethinkdb_user }}

      directory=/var/lib/rethinkdb/{{ instance_name }}/data
      #pid-file=/var/run/rethinkdb/{{ instance_name }}/pid_file
      log-file=/var/log/rethinkdb/{{ instance_name }}.log

      bind=all
      driver-port={{ driver_port }}
      cluster-port={{ cluster_port }}
      cores={{ cores }}
      cache-size={{ cache_size }}
      io-threads={{ io_threads }}
      http-port={{ http_port }}

      server-name={{ inventory_hostname }}

      {% if inventory_hostname != groups['rethinkdb'][0] %}
      join={{ groups['rethinkdb'][0] }}:{{ cluster_port }}
      {% endif %}

- name: Initialize database directory
  command: >
    {{ rethinkdb_prefix }}/bin/rethinkdb create
    -d /var/lib/rethinkdb/{{ instance_name }}/data
  become_user: "{{ rethinkdb_user }}"
  args:
    creates: /var/lib/rethinkdb/{{ instance_name }}/data/metadata
```

10bis) KUBERNETIZZAZIONE!

Per Kubernetes non si installa il database con uno script bash sul nodo come nell'esempio principale (esteso con Ansible).

In Kubernetes l'approccio corretto è:

1. mettere RethinkDB in un **container**
2. usare un **StatefulSet** (perché ogni nodo del cluster deve avere identità stabile)
3. usare un **Headless Service** per la discovery tra nodi
4. usare una **PersistentVolume** se vuoi dati persistenti

-> i nodi del cluster RethinkDB si collegano tra loro con --join

Nel nostro caso abbiamo 3 nodi fisici con IP:

- 192.156.213.164
- 192.156.213.165
- 192.156.213.166

Possiamo far girare 1 pod per nodo usando nodeSelector oppure nodeName.

Namespace

Creare un file YAML (N.B. ATTENZIONE ALL'INDENTAZIONE!):

namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: rethinkdb
```

Headless Service (per cluster discovery)

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: rethinkdb
  namespace: rethinkdb
spec:
  clusterIP: None
  selector:
```

```
    app: rethinkdb
  ports:
    - name: driver
      port: 28015
    - name: cluster
      port: 29015
    - name: http
      port: 8080
```

Questo crea DNS tipo:

- rethinkdb-0.rethinkdb
- rethinkdb-1.rethinkdb
- rethinkdb-2.rethinkdb

Prima etichetta i nodi Kubernetes.

Sul cluster:

kubectl label node 192.156.213.164 rethinkdb=node1
kubectl label node 192.156.213.165 rethinkdb=node2
kubectl label node 192.156.213.166 rethinkdb=node3

StatefulSet

statefulset.yaml

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: rethinkdb
  namespace: rethinkdb
spec:
  serviceName: rethinkdb
  replicas: 3
```

```
selector:
  matchLabels:
    app: rethinkdb

template:
  metadata:
    labels:
      app: rethinkdb

  spec:
    containers:
    - name: rethinkdb
      image: rethinkdb:2.4

      ports:
      - containerPort: 28015
      - containerPort: 29015
      - containerPort: 8080

      command:
      - rethinkdb
      - "--bind"
      - "all"
      - "--join"
      - "rethinkdb-0.rethinkdb:29015"

      volumeMounts:
      - name: data
        mountPath: /data

volumeClaimTemplates:
- metadata:
    name: data
  spec:
```

```
        accessModes: ["ReadWriteOnce"]
        resources:
          requests:
            storage: 20Gi
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
          - matchExpressions:
            - key: rethinkdb
              operator: In
              values:
              - node1
              - node2
              - node3
```

Deployment

kubectl apply -f namespace.yaml

kubectl apply -f service.yaml

kubectl apply -f statefulset.yaml

Accesso UI

Port-forward:

kubectl port-forward svc/rethinkdb 8080:8080 -n rethinkdb

poi andare sul browser:

http://localhost:8080

N.B. Sul fronte Kubernetes si puo' gestire sia un sistema di sviluppo sia un sistema test bed o di produzione. La ridondanza high-availability and no-spof si ottiene aggiungendo modulo per modulo dentro l'environment di Kubernetes.

Ci fermiamo qui perche' non e' lo scopo del corso.

Quindi ritorniamo alla configurazione del cluster da script per capire passo passo cosa succede nel DB Rethinkdb.

11) Come scegliere il nodo scarico ?

Basta creare una lista di nodi:

```
r.connect(hosts=[
  {'host':'node4','port':28015},
  {'host':'node5','port':28015},
  {'host':'node6','port':28015}
])
```

così nell'ordine fornito il client tenterà la connessione al primo nodo poi al secondo e poi al terzo in caso di successo gli altri vengono messi in stato fallback.

Si può randomizzare livello applicativo:

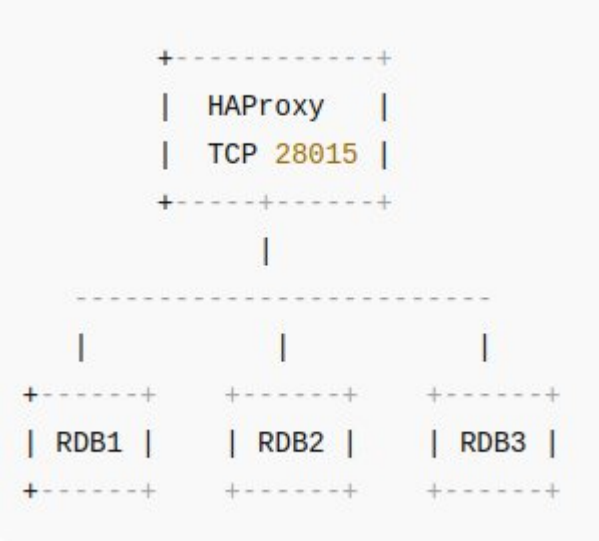
```
HOSTS = [
    {'host':'192.156.213.161','port':28015},
    {'host':'192.156.213.162','port':28015},
    {'host':'192.156.213.163','port':28015}
]
def connect_rethinkdb(hosts, db=None, user=None, password=None, timeout=5):
    hosts = hosts.copy()
    random.shuffle(hosts)
    last_error = None
    for h in hosts:
        try:
            conn = r.connect(
                host=h['host'],
                port=h.get('port', 28015),
                db=db,
                user=user,
                password=password,
                timeout=timeout
            )
            print(f"Connected to RethinkDB node {h['host']}")
            return conn
        except Exception as e:
            last_error = e
            print(f"Failed connecting to {h['host']}: {e}")

    raise Exception(f"No RethinkDB nodes reachable. Last error: {last_error}")
```

12) Alternativa: creare una entry point unica TCP per il cluster rethinkdb?

Si usa HAProxy in questo caso ci si connette al server HAProxy → haproxy_ip:28015

Immaginiamo di avere tre nodi rethinkdb con 3 IP separati.
Ogni nodo:

- parte con --join
- fa parte dello stesso cluster
- gestisce shard + repliche all'interno del cluster



installazione HAProxy

```
apt install haproxy
```

Modificare:

```
/etc/haproxy/haproxy.cfg
global
    log /dev/log local0
    maxconn 4096
    daemon

defaults
    mode tcp
    timeout connect 5s
    timeout client  1m
    timeout server  1m

frontend rethinkdb_front
    bind *:28015
    default_backend rethinkdb_back
```

```
backend rethinkdb_back
    mode tcp
    balance roundrobin
    option tcp-check

    option clitcpka #per keepalive client
    option srvtcpka #per keepalive HAProxy
    server rdb1 192.156.213.164:28015 check
    server rdb2 192.156.213.165:28015 check
    server rdb3 192.156.213.166:28015 check
```

#Riavvio Haproxy:

systemctl restart haproxy

systemctl status haproxy

#Ora i client possono collegarsi al solo nodo Haproxy:

haproxy_ip:28015

13) CLIENT configuration

su AMAS il client:

cedadm@amas:~$ sudo apt-get install python3-venv
cedadm@amas:~$ python3 -m venv ./venv
cedadm@amas:~$ source venv/bin/activate
(venv) cedadm@amas:~$ pip install rethinkdb
Collecting rethinkdb
Using cached rethinkdb-2.4.9-py2.py3-none-any.whl (160 kB)
Collecting six
Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, rethinkdb
Successfully installed rethinkdb-2.4.9 six-1.16.0
(venv) cedadm@amas:~$ python

```
Python 3.10.12 (main, Jan  8 2026, 06:52:19) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

>>> from rethinkdb import r
>>> r.connect('192.156.213.164', 28015).repl()

```
<rethinkdb.net.DefaultConnection object at 0x7fa8de241570>
```

>>>

>>> r.db('test').table_create('tv_shows').run()

```
{'config_changes': [{'new_val': {'db': 'test', 'durability': 'hard',
'id': '204e9d99-343a-4f05-9a0d-bb225027b121', 'indexes': [], 'name':
'tv_shows', 'primary_key': 'id', 'shards': [{'nonvoting_replicas':
[], 'primary_replica': 'amasdb1_2gy', 'replicas': ['amasdb1_2gy']}],
'write_acks': 'majority', 'write_hook': None}, 'old_val': None}],
'tables_created': 1}
```

>>> r.table('tv_shows').insert({ 'name': 'Star Trek TNG' }).run()

```
{'deleted': 0, 'errors': 0, 'generated_keys': ['e014156a-0195-4103-
abaa-4e043345bb79'], 'inserted': 1, 'replaced': 0, 'skipped': 0,
'unchanged': 0}
```

>>> CTRL+D exit

#VERIFICA SUSSISTENZA ENTRY CREATA SU NODO4:

(venv) cedadm@amas:~$ python

```
Python 3.10.6 (main, May 29 2023, 11:10:38) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

>>> from rethinkdb import r
>>> r.connect('192.156.213.164', 28015).repl()

```
<rethinkdb.net.DefaultConnection object at 0x7f74976380a0>
```

>>> cursor = r.db("test").table("tv_shows").run()
>>> for document in cursor:
... print(document)
...

```
{'id': 'e014156a-0195-4103-abaa-4e043345bb79', 'name': 'Star Trek TNG'}
```

OK!

ora verificare sul nodo5 e 6 che sia sincronizzato il cluster!

```
(venv) cedadm@amas:~$ python
Python 3.10.6 (main, May 29 2023, 11:10:38) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from rethinkdb import r
>>> r.connect('192.156.213.165', 28015).repl()
<rethinkdb.net.DefaultConnection object at 0x7f98b8634100>
>>> cursor = r.db("test").table("tv_shows").run()
>>> for document in cursor:
...     print(document)
...
{'id': 'e014156a-0195-4103-abaa-4e043345bb79', 'name': 'Star Trek TNG'}
>>>  CTRL+D exit!
```

OK!

<u>Ripetere lo stesso su node6</u>

```
(venv) cedadm@amas:~$ python
Python 3.10.6 (main, May 29 2023, 11:10:38) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from rethinkdb import r
>>> r.connect('192.156.213.166', 28015).repl()
<rethinkdb.net.DefaultConnection object at 0x7fa77da6c0a0>
>>> cursor = r.db("test").table("tv_shows").run()
>>> for document in cursor:
...     print(document)
...
{'id': 'e014156a-0195-4103-abaa-4e043345bb79', 'name': 'Star Trek TNG'}
>>> CTRL+D exit!
```

OK!

14) Backup e Dbdump

14a) Creare un backup

(venv) cedadm@amas**:~$** rethinkdb-dump -c 192.156.213.164 -f test.dump.tgz --clients 3 -e test
(venv) cedadm@amas**:~$** more test.dump.tgz
(venv) cedadm@amas:~$ **tar -xvf test.dump.tgz**
rethinkdb_dump_2023-06-16T12:26:45/test/tv_shows.info
rethinkdb_dump_2023-06-16T12:26:45/test/tv_shows.json
(venv) cedadm@amas:~$ cd rethinkdb_dump_2023-06-16T12\:26\:45/
(venv) cedadm@amas:~/rethinkdb_dump_2023-06-16T12:26:45$ **ls**
test
(venv) cedadm@amas:~/rethinkdb_dump_2023-06-16T12:26:45$ **cd test/**
(venv)cedadm@amas:~/rethinkdb_dump_2023-06-16T12:26:45/test$ **ls**

```
      tv_shows.info  tv_shows.json
(venv)cedadm@amas:~/rethinkdb_dump_2023-06-16T12:26:45/test$
more tv_shows.info
      {"db": {"id": "6b62684f-6bbe-4005-8fcb-5daa4471100f", "name":
      "test", "type": "DB"}, "doc_count_estimates": [2], "id": "204e9d99-
      343a-4f05-9a0d-bb225027b121", "indexes": [], "name": "tv_shows",
      "prim
      ary_key": "id", "type": "TABLE", "write_hook": null}
(venv)cedadm@amas:~/rethinkdb_dump_2023-06-16T12:26:45/test$  more tv_shows.json
      [
      {"id": "e014156a-0195-4103-abaa-4e043345bb79", "name": "Star Trek
      TNG"}
      ]
```

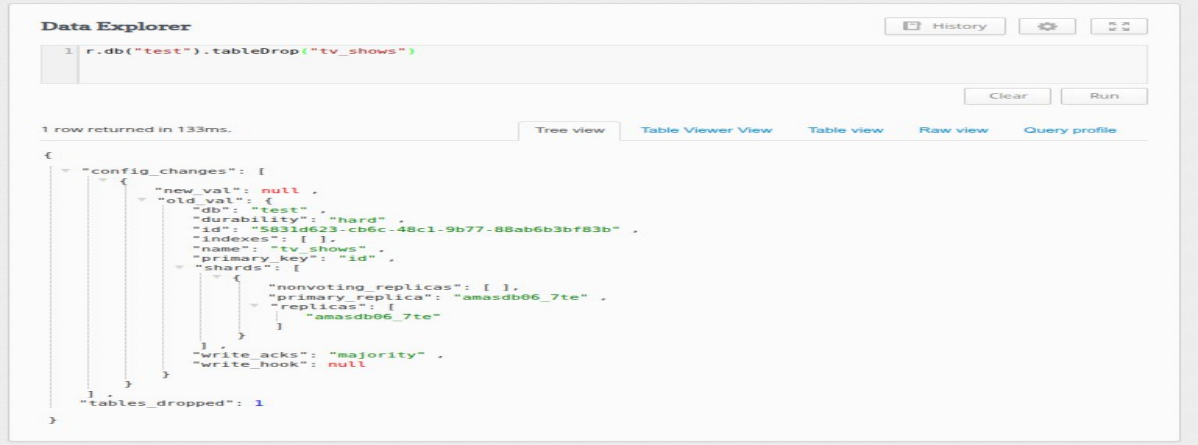
Ripristino di un DUMP

14b) Resumare un DB DUMP

```
(venv) cedadm@amas:~$  python
>>> from rethinkdb import r
>>> r.connect('192.156.213.164', 28015).repl()
<rethinkdb.net.DefaultConnection object at 0x7f00421cc0a0>
>>> r.table('tv_shows').insert({ 'name': 'Star Trek Next Generation' }).run()
{'deleted': 0, 'errors': 0, 'generated_keys': ['97de98ce-aae9-4c06-b354-c08dace0965a'],
'inserted': 1, 'replaced': 0, 'skipped': 0, 'unchanged': 0}
>>>
>>> cursor = r.db("test").table("tv_shows").run()
>>> for document in cursor:
...     print(document)
...
{'id': 'e014156a-0195-4103-abaa-4e043345bb79', 'name': 'Star Trek TNG'}
{'id': '97de98ce-aae9-4c06-b354-c08dace0965a', 'name': 'Star Trek Next Generation'}
>>>  CTRL+D   exit
```

Aggiunta una entry proviamo a ripristinare il DB con il file “test.dump.tgz”
in questo caso I record con ID uguali verranno saltati e gli altri aggiunti,mentre se si inserisce l’opzione –force allora gli ID uguali verranno sovrascritti dal dump importato.

Se si vuole cancellare la tabella bisogna eseguire il comando di drop a monte prima del restore: es. Dalla console Data Explorer: cancellando il DB o la tabella:



r.dbDrop("test")
#oppure

r.db("test").tableDrop("tv_shows")
#oppure ancora cancellando le entry nella tabella:

r.db("test").table("tv_shows").delete()

#e poi eseguire il restore:

(venv) cedadm@amas:~$ **rethinkdb-restore test.dump.tgz -c 192.156.213.164 --clients 8 -i test --force**
Extracting archive file...
Done (0 seconds)
Importing from directory...
[==] 100%
1 row imported to 1 table in 0.51 secs
(venv) cedadm@amas:~$

(l' opzione --force forza la sovrascrittura degli stessi ID ma non cancella quelli già salvati, per fare quello devi fare una cancellazione a monte)

Lo stesso si può fare chiaramente da riga di comando/ shell python:

(venv)cedadm@amas:~/$ python
>>> from rethinkdb import r
>>> conn=r.connect('192.156.213.163', 28015).repl()
>>> r.db_drop("test").run(conn)
{'config_changes': [{'new_val': None, 'old_val': {'id': '6b62684f-6bbe-4005-8fcb-5daa4471100f', 'name': 'test'}}], 'dbs_dropped': 1, 'tables_dropped': 1}
>>>
>>> cursor = r.db("test").table("tv_shows").run()
Traceback (most recent call last):
File "<stdin>", line 1, in <module> File "/home/cedadm/venv/lib/python3.10/site-packages/rethinkdb/ast.py", line 154, in run return c._start(self, **global_optargs)

```
  File "/home/cedadm/venv/lib/python3.10/site-packages/rethinkdb/net.py", line 749,
in _start    return self._instance.run_query(q, global_optargs.get("noreply", False))
  File "/home/cedadm/venv/lib/python3.10/site-packages/rethinkdb/net.py", line 578,
in run_query
    raise res.make_error(query)
rethinkdb.errors.ReqlOpFailedError: Database `test` does not exist in:
r.db('test').table('tv_shows')
>>> CTRL + D   exit

(venv)cedadm@amas:~/$  rethinkdb-restore test.dump.tgz -c 192.156.213.161 --clients 8 -i
test (non serve --force)
    Extracting archive file...
    Done (0 seconds)
    Importing from directory...
  [========================================] 100%
  1 row imported to 1 table in 0.83 secs
(venv) cedadm@amas:~$
(venv) cedadm@amas:~$ python
    >>> from rethinkdb import r
    >>> r.connect('192.156.213.163', 28015).repl()
    <rethinkdb.net.DefaultConnection object at 0x7f88c5f0c0d0>
    >>> cursor = r.db("test").table("tv_shows").run()
    >>> for document in cursor:
    ...     print(document)
    ...
    {'id': 'e014156a-0195-4103-abaa-4e043345bb79', 'name': 'Star Trek TNG'}
    >>> CTRL + D    exit
```

14c) monitoring ISSUES:

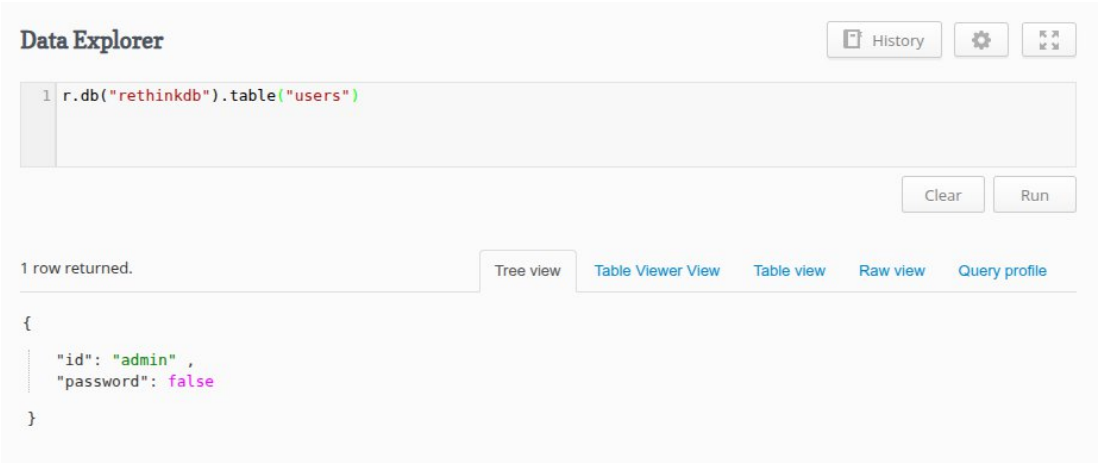
r.db("rethinkdb").table("current_issues")

14d) monitoring active JOBS:

r.db("rethinkdb").table("jobs")

15 Accounting e Sicurezza

15a) DALLA CONSOLE DATA EXPLORER



15b) setup password admin→

```
r.db('rethinkdb').table('users').get('admin').
update({password: 'admin1234'})
```

```
{
        "deleted": 0 ,
        "errors": 0 ,
        "inserted": 0 ,
        "replaced": 1 ,
        "skipped": 0 ,
        "unchanged": 0

}
```

15c) CREARE UTENTI:

```
#user redbteam → READ & WRITE ALL
r.db('rethinkdb').table('users').insert({
  id: 'redbteam',
  password: {
    password: 'password.01',
    iterations: 4096
  }
})
```

oppure da riga di comando:

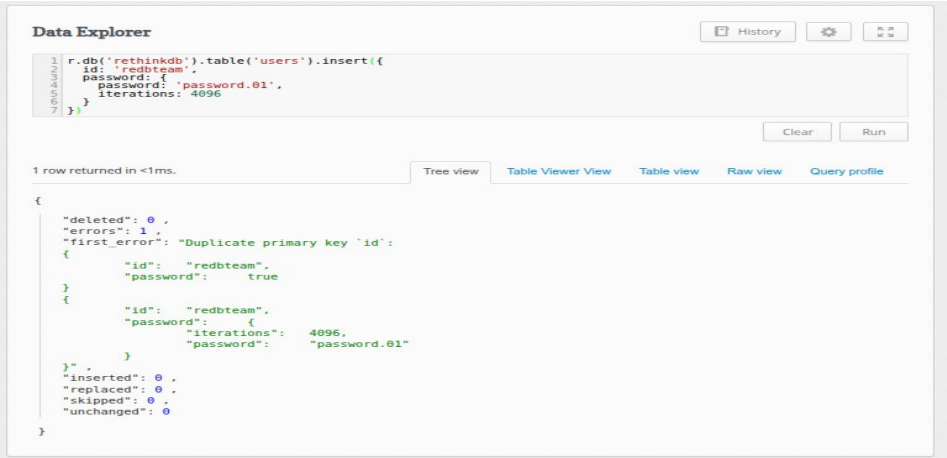
```
rethinkdb admin create user redbteam --password password.01
```

```
{
"deleted": 0 ,
```

"errors": 0 ,
"inserted": 1 ,
"replaced": 0 ,
"skipped": 0 ,
"unchanged": 0

}

se inserisco due volte lo stesso utente ovviamente da errore:



15d)GRANT UTENTI

#redbteam READ & WRITE SU TUTTO

```
r.db('test').grant('redbteam', {
  read: true,
  write: true,
  config: false
})
```

{
"granted": 1 ,
"permissions_changes": [
{
"new_val": {
"read": true ,
"write": true
} ,
"old_val": null
}
]}

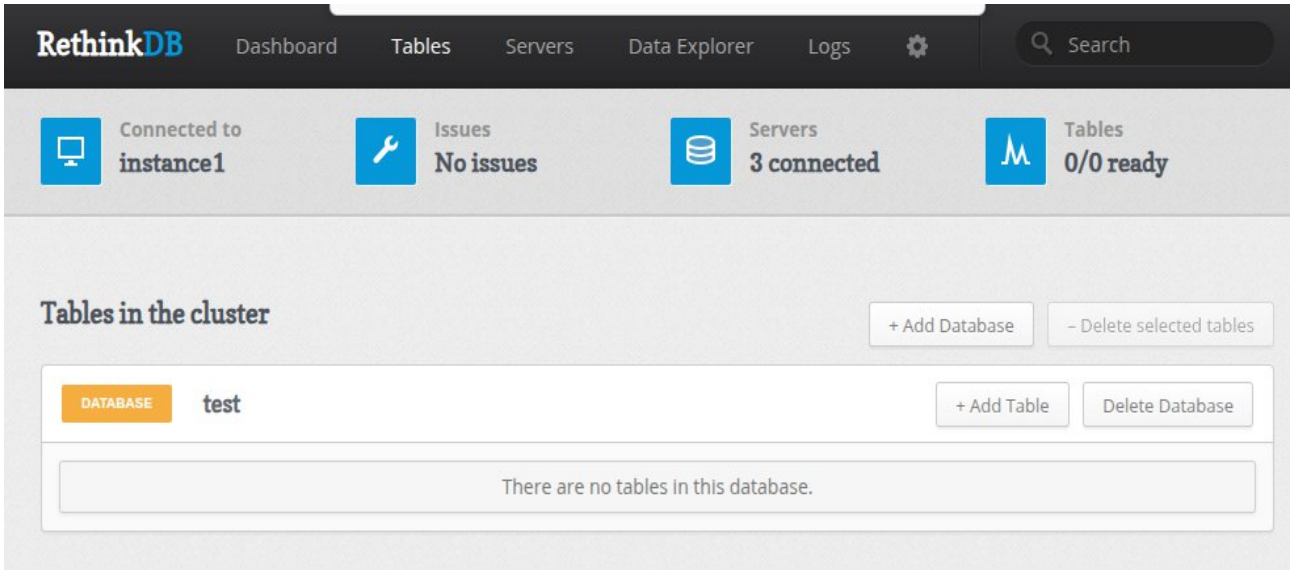
#GUEST utente in sola lettura

```
r.db('rethinkdb').table('users').insert({
  id: 'guest',
```

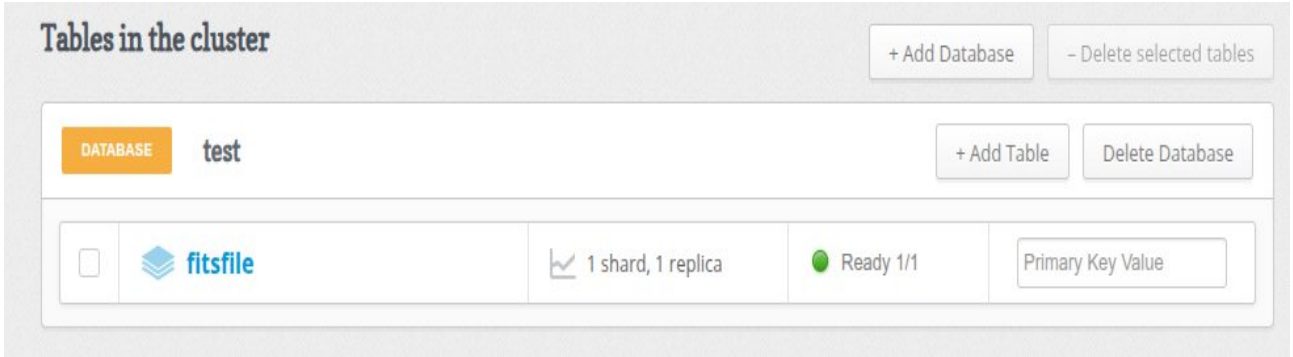
```
  password: {
    password: 'guest',
    iterations: 4096
  }
})

#GRANT DELLE TABELLE DATI IN SOLA LETTURA
r.db('test').grant('guest', {read:true,
write:null, config:null})
```

16) CREAZIONE TABELLE , REPLICHE E SHARDS

16a) creazione Tabelle

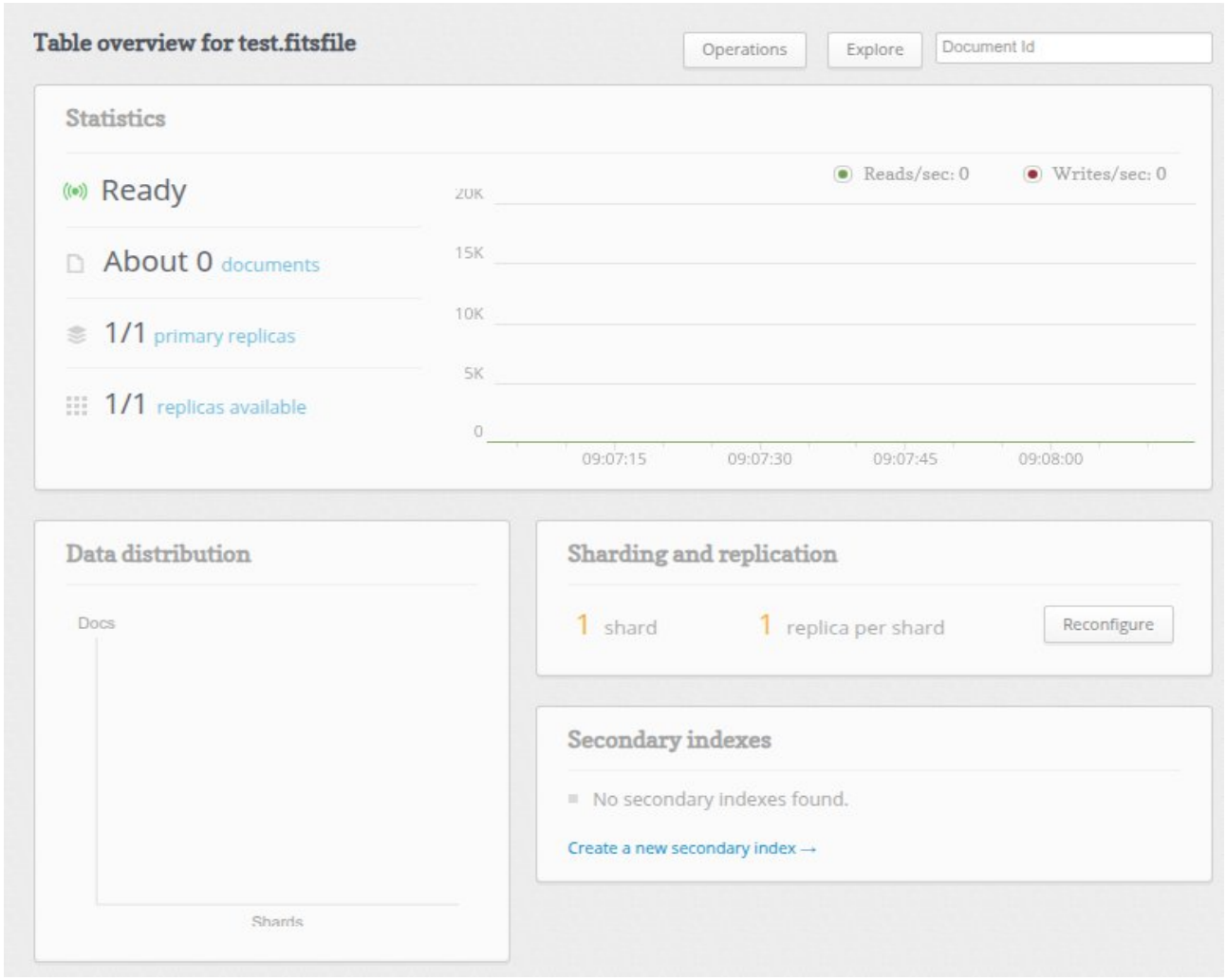


Nella dashboard andare su tabelle, “+Add Table” → inserire il nome della tabella (es “fitsfile”).

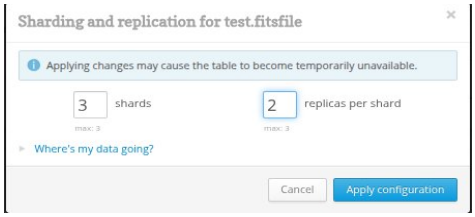


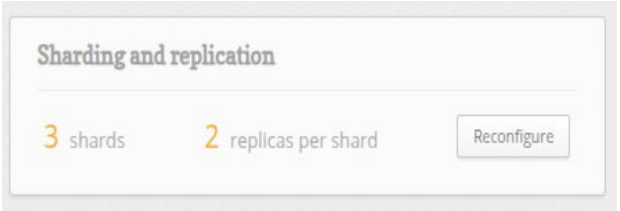
16b) Sharding e Replication

Cliccare sulla tabella e configurare il livello di ridondanza “Reconfigure” → sharding and Replication.



Ed impostare (ad esempio) 3 shards e 2 repliche per shard → “apply configuration”





Attualmente la tabella e’ vuota nel prossimo capitolo la andremo a popolare

16c) Definizione del datamodel

Il database RethinkDB e' un database schemaless! Quindi si possono utilizzare qualsivoglia schema di dati, anche non coerenti purche' vengano opportunamente mappati in uno schema per gli applicativi che dovranno utilizzare quel datamodel.

Definiamo uno schema delle entry della tabella (attualmente vuota).

AD ESEMPIO: Prendiamo ad esempio lo **schema pubblico dei dati FITS di livello zero di ASTRI-Miniarray:**

https://amas.oa-roma.inaf.it/static/aipMADLFITSschema

Che descrive il datamodel dei documenti JSON per i dati fits di livello zero.


```
{
    "aaid": "17702838561523838",
    "aipvers": "AMAS_1.0.0",
    "archive": {
        "PFN": "20260204_MA03_cal-phd_Fixed_00000001_I_000247_1001.lv0.fits.gz",
        "PFNP":
"/archive/MINIARRAY/PHYSICAL/pass_0.0.1/20260204/00000001/dl0/hk/v1",
        "archdate": "2026-02-05 09:30:56.152404",
        "archtime": 1770283856,
        "checksum": "b50fc30f",
        "container": "20260205",
        "dataset": "fits-data",
        "filesize": 445096,
        "paths": {
            "RSE": "astriFS",
            "replicaflag": 0,
            "type": "web-https",
            "uid": "17702838561523838",
            "uripath": "https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.1/20260204/00000001/dl0/hk/v1/202602
04_MA03_cal-phd_Fixed_00000001_I_000247_1001.lv0.fits.gz"
        },
        "replicas": {
            "replica": [
                {
                    "number": "0",
                    "rdata": "2026-02-05 09:30:56.152404",
                    "rid": "17702838561523838",
                    "rtype": "web-https",
                    "spool": "AMAS",
                    "uri": "https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.1/20260204/00000001/dl0/hk/v1/202602
04_MA03_cal-phd_Fixed_00000001_I_000247_1001.lv0.fits.gz"
                }
            ]
        },
        "scope": "MA01"
    },
    "author": "Stefano Gallozzi",
    "camera": "{'name': 'ASTRIMA', 'origin': 'ASTRIDPS', 'creator': 'adas
preprocessing v1.1', 'npdm': 37, 'modeid': 'I', 'datatype': 'fits-data'}",
    "daqmode": "I",
    "datadesc": "lv0_housekeeping",
```

```
    "datatype": "1001",
    "dateobs": "2026-02-04",
    "event": {
        "obsdate": "2026-02-04"
    },
    "file_version": 1,
    "filename": "20260204_MA03_cal-
phd_Fixed_00000001_I_000247_1001.lv0.fits.gz",
    "fsize": 445096,
    "header": {
        "Primary": {
            "ALT_PNT": -999,
            "AZ_PNT": -999,
            "BITPIX": 16,
            "CHECKSUM": "g5XEj3XBg3XBg3XB",
            "COMMENT": [
                "= 'FITS (Flexible Image Transport System) format is defined in
''Astron'"
            ],
            "CREATOR": "adas preprocessing v2.4.4",
            "DAQ_ID": "000247",
            "DAQ_MODE": "I",
            "DATAFORMAT": "v1.0",
            "DATALEVEL": "lv0",
            "DATAMODE": "10",
            "DATASUM": "0",
            "DATE": "2026-02-05T08:10:33",
            "DATE-END": "2026-02-04T21:31:36",
            "DATE-OBS": "2026-02-04T21:29:54",
            "DEC_OBJ": -999,
            "DEC_PNT": -999,
            "EQUINOX": "2000.0",
            "EXTEND": true,
            "FILENAME": "20260204_MA03_cal
phd_Fixed_00000001_I_000247_1001.lv0.fits",
            "FILEVERS": 1,
            "INSTRUME": "CAMERA",
            "MJDREFF": 0.00080074,
            "MJDREFI": 58849,
            "NAXIS": 0,
            "NTEL": 1,
            "OBJECT": "cal-phd",
            "OBS_DATE": "20260204",
            "OBS_MODE": "Fixed",
            "ORIGIN": "ASTRIDPS",
            "ORIG_ID": "00",
            "PROG_ID": "001",
            "RADECSYS": "FK5",
            "RA_OBJ": -999,
            "RA_PNT": -999,
            "RUN_ID": "00000001",
            "SBL_ID": "002",
            "SIMPLE": true,
            "SUBMODE": "01",
            "TELAPSE": "102",
            "TELESCOP": "ASTRI-MA",
            "TEL_ID": "03",
            "TIMEOFFS": 0,
            "TIMESYS": "TT",
            "TIMEUNIT": "s",
            "TSTART": "192403794",
            "TSTOP": "192403896"
```

```
    }
},
"id": "b0b32ecc-e2fd-44a0-bc2b-e1b9559375ca",
"infomail": "stefano.gallozzi@inaf.it",
"latest_version": 1,
"object": "cal-phd",
"obsid": 247,
"obsmode": "Fixed",
"packtype": "fits-data",
"programid": 111,
"proposal": {
    "carryover": "Y",
    "category": "EXT/RP",
    "cycle": {
        "name": "cycle2024/1",
        "period": "[ 2023-05-01,2023-05-05 ]",
        "type": "semester"
    },
    "obsprog": {
        "arrayconf": {
            "acq_mode": "wobble",
            "acq_submode": "2.5",
            "telmatrix": {
                "confname": "fulla_ma",
                "on_off": "[ 1,0,0,0,0,0,0,0,0 ]",
                "type": "array"
            },
            "trigger": "S2"
        },
        "constraints": {
            "maxMIF": "0.7",
            "maxZA": "60.0 [deg]",
            "minAT": "0.7",
            "minET": "100 [hrs]",
            "minMD": "100 [deg]",
            "minZA": "0.0 [deg]"
        },
        "progid": "1",
        "target": {
            "coord": "[ 1e-09, 1e-09 ]",
            "dec": 1e-09,
            "diameter": "0.0 [arsec]",
            "epoch": "J2000",
            "magnitude": "0.0 [ABmag]",
            "name": "cal-dark-ped-var_Fixed",
            "rad": 1e-09,
            "tooflag": "0",
            "type": "wcs"
        }
    },
    "piname": "Giovanni Pareschi",
    "propdate": "2023-11-21",
    "propid": "2",
    "proplink": "https://amas.oa-roma.inaf.it/proposals/2/",
    "proptype": "SCI",
    "reqtime": "300 [hrs]"
},
"runid": 1,
"schema": "https://amas.oa-roma.inaf.it/static/aipMADLFITSschema",
"schemavers": "FITS_v0.0.1",
"telescope": {
    "altitude": "2390 [m]",
```

```
        "diamS1": "4.6 [m]",
        "diamS2": "1.5 [m]",
        "geo": {
            "coord": [
                28.30015,
                -16.50965
            ],
            "type": "gps"
        },
        "optconf": "DM",
        "telid": "03",
        "telname": "MA03",
        "type": "AIC"
    },
    "timestamp": "2026-02-05 09:30:56+00:00"
}
```


Carichiamo alcune decine di entry già nel formato json.

Scarichiamo un piccolo dataset preparato per il corso:

```
wget: https://amas.oa-roma.inaf.it/static/download/json/AMAS_test_20260206T081824.json
```

17) PYTHON SCRIPTS INTERFACES

17a) CONNETTERSI A DB da utente:

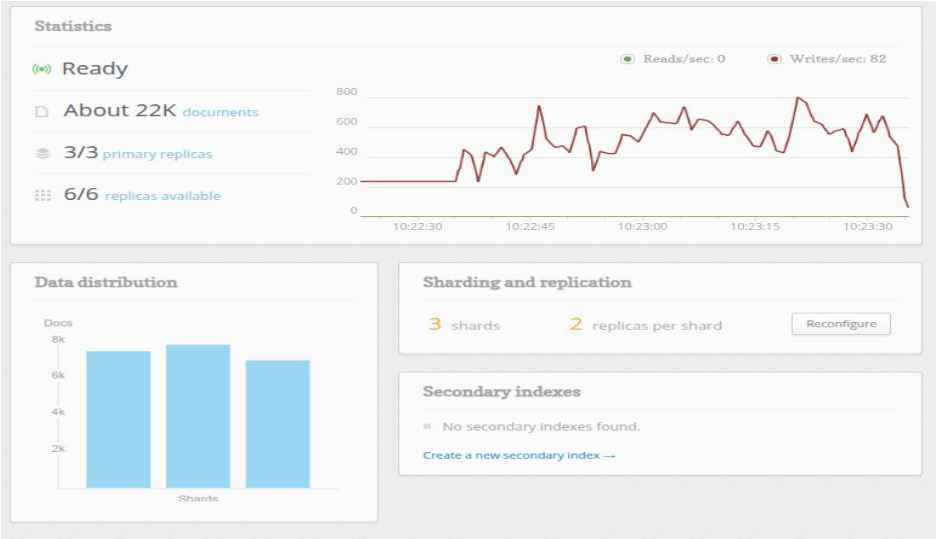
N.B. per semplicità nei successi test ci collegheremo ad un singolo host e non all Haproxy o usando la funzione di connect che esegue lo shuffle tra in nodi del cluster.

```
python3
>>> import json
>>> from rethinkdb import RethinkDB
>>> r = RethinkDB()
>>> conn = r.connect( host='192.156.213.164', port=28015, db='test', user='redbteam', password='password.01' )
>>> conn.repl()
<rethinkdb.net.DefaultConnection object at 0x7a869501b020>
```

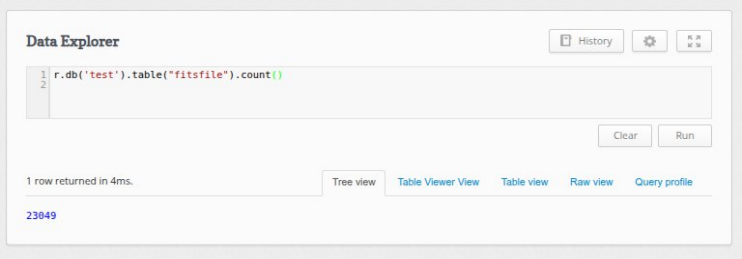
17b) CARICAMENTO DEL DATASET nella tabella:

```
>>> with open('AMAS_test_20260206T081824.json', 'r', encoding='utf-8') as f:
...     data = json.load(f)
...
>>> r.table('fitsfile').insert(data).run(conn)
{'deleted': 0, 'errors': 0, 'inserted': 23049, 'replaced': 0, 'skipped': 0, 'unchanged': 0}
```

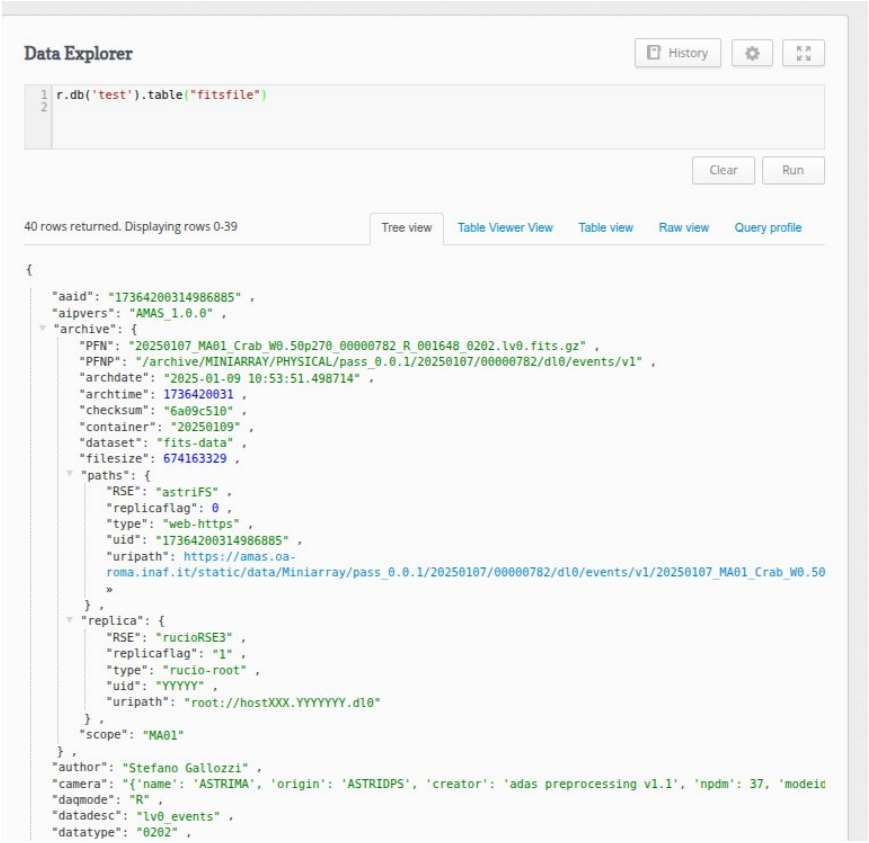
Sulla dashboard di RethinkdB si vedrà l'andamento del caricamento dei records.



#Sul Data Explorer possiamo fare un count :
r.db("test").table("fitsfile").count()



e vedere l'anteprima di alcuni records:



18) Effettuare query

18a) Funzione FILTER

python3 -m venv ./venv
source venv/bin/activate

(venv) :/home/# python
```
Python 3.12.3 (main, Jan 22 2026, 20:57:42) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```
>>> from rethinkdb import RethinkDB
>>> r = RethinkDB()
>>> conn = r.connect(host='192.156.213.164', port=28015, db='test', user='redbteam', password='password.01').repl()
>>> results = r.table("fitsfile").run()
```
<RqlQuery instance: r.table('fitsfile') >
```
>>> for entry in results:
... print (entry)
...
```
{'aaid': '17364200314986885', 'aipvers': 'AMAS_1.0.0', 'archive': {'PFN':
'20250107_MA01_Crab_W0.50p270_00000782_R_001648_0202.lv0.fits.gz', 'PFNP':
'/archive/MINIARRAY/PHYSICAL/pass_0.0.1/20250107/00000782/dl0/events/v1',
'archdate': '2025-01-09 10:53:51.498714', 'archtime': 1736420031, 'checksum':
'6a09c510', 'container': '20250109', 'dataset': 'fits-data', 'filesize':
674163329, 'paths': {'RSE': 'astriFS', 'replicaflag': 0, 'type': 'web-https',
'uid': '17364200314986885', 'uripath': 'https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.1/20250107/00000782/dl0/events/v1/20
250107_MA01_Crab_W0.50p270_00000782_R_001648_0202.lv0.fits.gz'}, 'replica':
{'RSE': 'rucioRSE3', 'replicaflag': '1', 'type': 'rucio-root', 'uid': 'YYYYY',
'uripath': 'root://hostXXX.YYYYYYY.dl0'}, 'scope': 'MA01'}, 'author': 'Stefano
Gallozzi', 'camera': "{'name': 'ASTRIMA', 'origin': 'ASTRIDPS', 'creator': 'adas
preprocessing v1.1', 'npdm': 37, 'modeid': 'R', 'datatype': 'fits-data'}",
'daqmode': 'R', 'datadesc': 'lv0_events', 'datatype': '0202', 'dateobs': '2025-
01-07',
….
….
….
>>>
```
→ dentro il cursore entry c'è tutto il documento JSON riferito alla posizione n-esima

Quindi per prendere / concatenare I path dei files basterà fare print dei soli campi interessati entro le strutture annidate:
>>> results = r.table("fitsfile").run()
>>> for entry in results:
... pfnp = entry["archive"].get("PFNP", "")
... pfn = entry["archive"].get("PFN", "")
... filefits = f"{pfnp}/{pfn}"
... print (filefits)
...
```
/archive/MINIARRAY/PHYSICAL/pass_0.0.1/20250107/00000782/dl0/events/v1/20250107_
MA01_Crab_W0.50p270_00000782_R_001648_0202.lv0.fits.gz
/archive/MINIARRAY/PHYSICAL/pass_0.0.2/20260114/00001505/dl2/events/v2/20260114_
MA_OffFixed-20-000_Fixed_00001505_R_003539_0201.lv2b.fits.gz
```

```
/archive/MINIARRAY/PHYSICAL/pass_0.0.1/20260115/00001525/dl0/scitech/v1/20260115
_MA01_Mrk421_W0.50p270_00001525_I_003560_1004.lv0.fits.gz
/archive/MINIARRAY/PHYSICAL/pass_0.0.1/20251115/00001381/dl0/hk/v2/20251115_MA01
_OffFixed-20-270_Fixed_00001381_R_003330_1001.lv0.fits.gz
/archive/MINIARRAY/PHYSICAL/pass_0.0.1/20250731/00001155/dl0/scitech/v1/20250731
_MA01_cal-phd_Fixed_00001155_I_002908_1004.lv0.fits.gz
…
```

>>>

Così facendo ci portiamo dietro tutti I metadati del JSON quindi possiamo ridurre il tempo di elaborazione selezionando solo I metadati che ci interessano con la funzione “pluck”

```
>>> results = r.table("fitsfile").pluck({'aaid': True, 'file_version': True, 'archive': {'PFN': True, 'paths':
{'uripath': True }}}).run()
>>> for entry in results:
...     print (entry)
...
{'aaid': '17536053947110617', 'archive': {'PFN': '20250725_MA01_cal-dark-
ped_Fixed_00001204_I_002863_1003.lv0.fits.gz', 'paths': {'uripath':
'https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.1/20250725/00001204/dl0/varlg/v1/202
50725_MA01_cal-dark-ped_Fixed_00001204_I_002863_1003.lv0.fits.gz'}},
'file_version': 1}
{'aaid': '17384851986276 97', 'archive': {'PFN':
'20250201_MA01_Crab_W0.50p090_00000967_R_001940_1001.lv0.fits.gz', 'paths':
{'uripath': 'https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.1/20250201/00000967/dl0/hk/v1/202502
01_MA01_Crab_W0.50p090_00000967_R_001940_1001.lv0.fits.gz'}}, 'file_version': 1}
{'aaid': '17488531087228715', 'archive': {'PFN': '20250529_MA01_EL73-AZ39-
scan_Fixed_00001074_R_002270_1004.lv0.fits.gz', 'paths': {'uripath':
'https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.1/20250529/00001074/dl0/scitech/v3/2
0250529_MA01_EL73-AZ39-scan_Fixed_00001074_R_002270_1004.lv0.fits.gz'}},
'file_version': 3}
{'aaid': '1762877567607774', 'archive': {'PFN':
'20250824_MA_Mrk501_W0.50p000_00001141_R_003041_0201.lv2b.fits.gz', 'paths':
{'uripath': 'https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.2/20250824/00001141/dl2/events/v1/20
250824_MA_Mrk501_W0.50p000_00001141_R_003041_0201.lv2b.fits.gz'}},
'file_version': 1}
{'aaid': '1753173351377538', 'archive': {'PFN':
'20250721_MA01_BLLac_W0.5p090_00001159_I_002794_1001.lv0.fits.gz', 'paths':
{'uripath': 'https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.1/20250721/00001159/dl0/hk/v1/202507
21_MA01_BLLac_W0.5p090_00001159_I_002794_1001.lv0.fits.gz'}}, 'file_version': 1}
{'aaid': '17510423418369935', 'archive': {'PFN':
'20250626_MA01_Mrk501_W0.50p180_00001131_I_002553_1002.lv0.fits.gz', 'paths':
{'uripath': 'https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.1/20250626/00001131/dl0/varhg/v1/202
50626_MA01_Mrk501_W0.50p180_00001131_I_002553_1002.lv0.fits.gz'}},
'file_version': 1}
…
>>>
```

Per filtrare ad esempio I dati per una qualsiasi keyword (es. RunID, dateObs ed obsmode) o qualsiasi altra keyword si può usare la funzione “filter” che deve essere concatenata, criterio di ricerca per criterio di ricerca. In quesot modo nel cursore ci saranno SOLO I record che rispondono ai criteri di ricerca e non tutto il dataset.
→ diminuisce il tempo di calcolo e di risposta della query

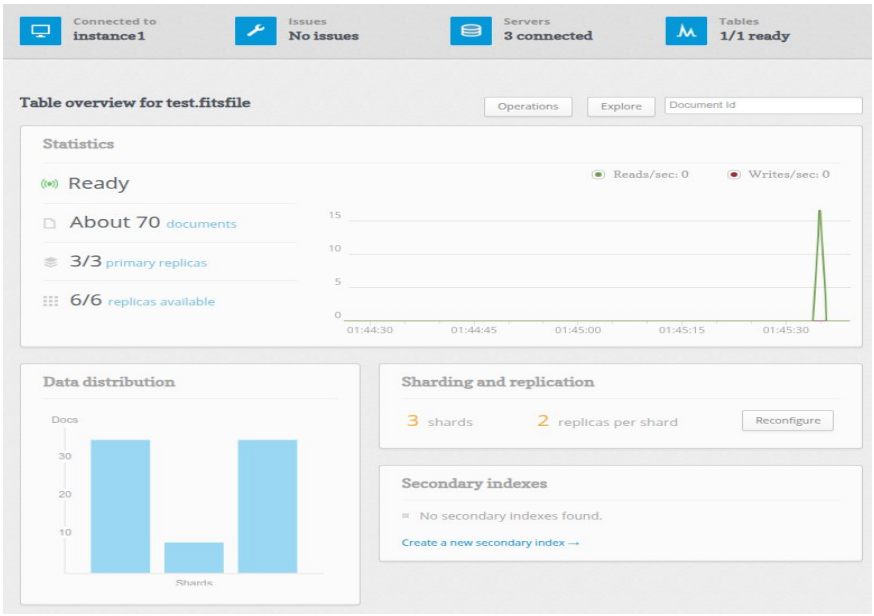
```
>>> cursor=r.table("fitsfile")
>>> cursor=cursor.filter(r.row['dateobs'].match('2025-08-22'))
```

```
>>> results = cursor.pluck({'file_version': True, 'archive': {'PFNP': True, 'PFN': True,
'paths': {'uripath': True }}}).run()
>>> for entry in results:
...     pfnp = entry["archive"].get("PFNP", "")
...     pfn = entry["archive"].get("PFN", "")
...     filefits = f"{pfnp}/{pfn}"
...     print (filefits)
...
/archive/MINIARRAY/PHYSICAL/pass_0.0.1/20250822/00001220/dl0/hk/v2/20250822_MA01
_Mrk501_W0.50p000_00001220_I_003007_1001.lv0.fits.gz
/archive/MINIARRAY/PHYSICAL/pass_0.0.1/20250822/00001127/dl0/scitech/v2/20250822
_MA01_BLLac_W0.50p000_00001127_I_003018_1004.lv0.fits.gz
```

E' possibile concatenare differenti filtri anche su keyword annidate:

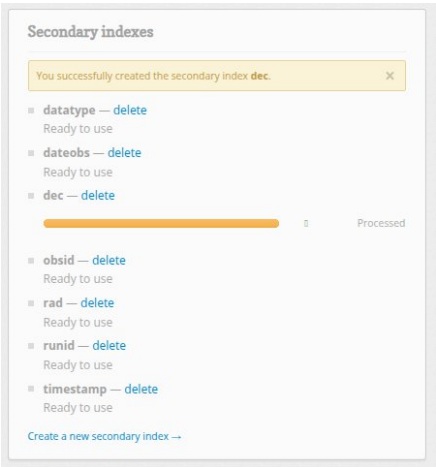
```
>>> cursor = r.table("fitsfile") \
...     .filter(lambda row: row['dateobs'].eq("2025-08-22")) \
...     .filter(lambda row: (row['runid'] >= 1219) & (row['runid'] <= 1221)) \
...     .filter(lambda row: row['daqmode'].eq("I"))
>>> results = cursor.pluck({
...     "file_version": True,
...     "archive": {
...         "PFNP": True,
...         "PFN": True,
...         "paths": {
...             "uripath": True
...         }
...     }
... }).run()
>>> for entry in results:
...     arch = entry.get("archive", {})
...     pfnp = arch.get("PFNP")
...     pfn  = arch.get("PFN")
...     uri  = arch.get("paths", {}).get("uripath")
...     print(pfnp, pfn, uri)
...
/archive/MINIARRAY/PHYSICAL/pass_0.0.1/20250822/00001220/dl0/hk/v2
20250822_MA01_Mrk501_W0.50p000_00001220_I_003007_1001.lv0.fits.gz
https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.1/20250822/00001220/dl0/hk/v2/202508
22_MA01_Mrk501_W0.50p000_00001220_I_003007_1001.lv0.fits.gz
>>>
```

18b) INDICIZZAZIONE



Quando la ricerca è lenta perché ci sono molti campi è possibile indicizzare alcuni campi dei metadati per permettere una ricerca più veloce.
Nella dashboard dentro la tabella I n questione si può cliccare su "create a new secondary index"

es. aggiungiamo come indicizzate "runid, dateobs, obsid, datatype, timestamp
ed eventualmente "rad" e "dec" per il cone search.



Per eseguire le query utilizzando gli indici si richiama index all'interno della query.

```
>>> cursor = r.table("fitsfile") \
... .between("2025-01-30", "2026-02-06", index='dateobs', left_bound='closed',right_bound='closed')
>>> results = cursor.pluck({"file_version": True,"archive": {   "PFNP": True,   "PFN": True,   "paths":
{       "uripath": True     } }}).run()
>>> for entry in results:
```

```
...     arch = entry.get("archive", {})
...     pfnp = arch.get("PFNP")
...     pfn  = arch.get("PFN")
...     uri  = arch.get("paths", {}).get("uripath")
...     print(pfnp, pfn, uri)
...
/archive/MINIARRAY/PHYSICAL/pass_0.0.1/20250220/00000993/dl0/scitech/v1
20250220_MA01_OffFixed-40-270_Fixed_00000993_I_001985_1004.lv0.fits.gz
https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.1/20250220/00000993/dl0/scitech/v1/2
0250220_MA01_OffFixed-40-270_Fixed_00000993_I_001985_1004.lv0.fits.gz
/archive/MINIARRAY/PHYSICAL/pass_0.0.1/20250130/00000946/dl0/hk/v1
20250130_MA01_cal-phd_Fixed_00000946_I_001904_1001.lv0.fits.gz https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.1/20250130/00000946/dl0/hk/v1/202501
30_MA01_cal-phd_Fixed_00000946_I_001904_1001.lv0.fits.gz
/archive/MINIARRAY/PHYSICAL/pass_0.0.2/20251023/00001344/dl1/events/v1
20251023_MA01_Crab_W0.50p180_00001344_R_003242_0201.lv1a.fits.gz
https://amas.oa-
roma.inaf.it/static/data/Miniarray/pass_0.0.2/20251023/00001344/dl1/events/v1/20
251023_MA01_Crab_W0.50p180_00001344_R_003242_0201.lv1a.fits.gz
```

19) CHANGE-Feed

Le funzionalità realtime di RethinkDB si testano con Change-Feed.

1) Creare una tabella di test per changefeed (es events).

PS: l'utente "redbteam" deve avere i permessi di "config" per poter creare tabelle:


```
r.db('test').grant('redbteam', { read: true, write: true, config: true })
```

====

```
from rethinkdb import RethinkDB

r = RethinkDB()

r.connect( host='192.156.213.164', port=28015,
db='test', user='redbteam',
password='password.01' ).repl()

DB = "test"
TABLE = "events"

if DB not in r.db_list().run():
    r.db_create(DB).run()

if TABLE not in r.db(DB).table_list().run():
    r.db(DB).table_create(TABLE).run()
      {'config_changes': [{'new_val': {'db': 'test', 'durability': 'hard', 'id': '6235be5a-
      e85f-41e0-9dde-5cb13014e43c', 'indexes': [], 'name': 'events', 'primary_key': 'id',
      'shards': [{'nonvoting_replicas': [], 'primary_replica': 'amasdb04_5os', 'replicas':
      ['amasdb04_5os']}], 'write_acks': 'majority', 'write_hook': None}, 'old_val': None}],
      'tables_created': 1}
# indice su type
if "type" not in r.db(DB).table(TABLE).index_list().run():
    r.db(DB).table(TABLE).index_create("type").run()
    r.db(DB).table(TABLE).index_wait("type").run()

print("Database and table ready.")
```

python initialize.py →

```
Database and table ready.
```

2) creare uno script **producer.py** che esegue una modifica ad una tabella es. "fitsfile"

```
from rethinkdb import RethinkDB
import time
import random
import os
import uuid
import string

def random_file_path_uuid(base_dir="/tmp", extension="json"):
    uid = uuid.uuid4()
    return os.path.join(base_dir, str(uid)[:2], str(uid)[2:4],
f"{uid}.{extension}")
```

```
r = RethinkDB()
r.connect( host='192.156.213.164', port=28015, db='test', user='redbteam',
password='password.01' ).repl()

while True:
    doc = {
        "type": "filepath",
        "value": random_file_path_uuid(),
        "size": random.randint(10,40),
        "ts": r.now()
    }

    time.sleep(2)
    res = r.table("events").insert(doc).run()
    print("Inserted:", res["generated_keys"][0])

    time.sleep(2)
    r.table("events").filter({'id':
res["generated_keys"[0]}).update({"value": random.randint(20,42)}).run()
    print("updated:", res["generated_keys"][0])

    time.sleep(2)
    r.table("events").filter({'id':
res["generated_keys"[0]}).delete().run()
    print("deleted:", res["generated_keys"][0])
```

Sul client creare lo script **listener.py**:

```
from rethinkdb import RethinkDB

r = RethinkDB()
r.connect( host='192.156.213.164', port=28015, db='test', user='redbteam',
password='passwor
d.01' ).repl()

feed = r.table("events").changes().run()
print("Listening for changes...")

for change in feed:
    print(change)

# solo nuove righe e non updates
# feed = r.table("events").changes(include_initial=False).run()
```

L'OUTPUT dei due script:

```
updated: 38410352-45a7-47ba-87ef-584683eaf6ce
deleted: 38410352-45a7-47ba-87ef-584683eaf6ce
Inserted: 9ff93853-52dd-4ea6-b557-14a792d8a108
updated: 9ff93853-52dd-4ea6-b557-14a792d8a108
deleted: 9ff93853-52dd-4ea6-b557-14a792d8a108
Inserted: d49ab5cd-ec93-408d-b4db-6ed008a2fcd6
updated: d49ab5cd-ec93-408d-b4db-6ed008a2fcd6
deleted: d49ab5cd-ec93-408d-b4db-6ed008a2fcd6
Inserted: 9125da34-a537-4f14-8575-66c1c28913d6
updated: 9125da34-a537-4f14-8575-66c1c28913d6
deleted: 9125da34-a537-4f14-8575-66c1c28913d6
Inserted: be8f65e3-4a8a-4d77-a75f-3f1f7a7f95ba
updated: be8f65e3-4a8a-4d77-a75f-3f1f7a7f95ba
deleted: be8f65e3-4a8a-4d77-a75f-3f1f7a7f95ba
Inserted: 84805181-f2d5-4bff-8c97-335929cdb511
updated: 84805181-f2d5-4bff-8c97-335929cdb511
deleted: 84805181-f2d5-4bff-8c97-335929cdb511
Inserted: 05335912-26db-4c44-90ce-aee6c916d25a
updated: 05335912-26db-4c44-90ce-aee6c916d25a
deleted: 05335912-26db-4c44-90ce-aee6c916d25a
Inserted: e61c6ff2-4f9e-4edf-a4dd-0b5769a045c2
updated: e61c6ff2-4f9e-4edf-a4dd-0b5769a045c2
```

```
atetime(2026, 2, 13, 14, 50, 17, 606000, tzinfo=<rethinkdb.ast.RqlTzinfo object at 0x74
89249bc9d0>), 'type': 'filepath', 'value': '/tmp/d4/37/d437df1c-a48c-410b-8292-2cb82057
ff2f.json'}, 'old_val': None}
{'new_val': {'id': '05335912-26db-4c44-90ce-aee6c916d25a', 'size': 28, 'ts': datetime.d
atetime(2026, 2, 13, 14, 50, 17, 606000, tzinfo=<rethinkdb.ast.RqlTzinfo object at 0x74
89249bc190>), 'type': 'filepath', 'value': 27}, 'old_val': {'id': '05335912-26db-4c44-9
0ce-aee6c916d25a', 'size': 28, 'ts': datetime.datetime(2026, 2, 13, 14, 50, 17, 606000,
 tzinfo=<rethinkdb.ast.RqlTzinfo object at 0x7489249bcd50>), 'type': 'filepath', 'value
': '/tmp/d4/37/d437df1c-a48c-410b-8292-2cb82057ff2f.json'}}
{'new_val': None, 'old_val': {'id': '05335912-26db-4c44-90ce-aee6c916d25a', 'size': 28,
 'ts': datetime.datetime(2026, 2, 13, 14, 50, 17, 606000, tzinfo=<rethinkdb.ast.RqlTzin
fo object at 0x7489249bc3d0>), 'type': 'filepath', 'value': 27}}
{'new_val': {'id': 'e61c6ff2-4f9e-4edf-a4dd-0b5769a045c2', 'size': 28, 'ts': datetime.d
atetime(2026, 2, 13, 14, 50, 23, 829000, tzinfo=<rethinkdb.ast.RqlTzinfo object at 0x74
89249bcc10>), 'type': 'filepath', 'value': '/tmp/92/ea/92ea7f54-0863-4e14-914d-30309c8d
c472.json'}, 'old_val': None}
{'new_val': {'id': 'e61c6ff2-4f9e-4edf-a4dd-0b5769a045c2', 'size': 28, 'ts': datetime.d
atetime(2026, 2, 13, 14, 50, 23, 829000, tzinfo=<rethinkdb.ast.RqlTzinfo object at 0x74
89249bcbd0>), 'type': 'filepath', 'value': 42}, 'old_val': {'id': 'e61c6ff2-4f9e-4edf-a
4dd-0b5769a045c2', 'size': 28, 'ts': datetime.datetime(2026, 2, 13, 14, 50, 23, 829000,
 tzinfo=<rethinkdb.ast.RqlTzinfo object at 0x7489249bd010>), 'type': 'filepath', 'value
': '/tmp/92/ea/92ea7f54-0863-4e14-914d-30309c8dc472.json'}}
```

Ovviamente invece di un print sul listener è pensabile avviare un qualsiasi script automatico che risponde alla modifica del DB, per esempio facendo partire una pipeline automatica nel momento in cui una nuova entry è presente nel DB. Questa funzionalità che tra I DB open source possiede SOLO RethinkDB, permette al DB di diventare il central core di ogni sistema (archivio ecc).

LO SCHEMA DA TNERE A MENTE COME EVENTI:

```
INSERT  -> old_val == None and new_val != None
UPDATE  -> old_val != None and new_val != None
DELETE  -> old_val != None and new_val == None
```

Quindi se voglio solo i nuovi record inseriti ad esempio un nuovo file di livello0 per poter eseguire le pipeline di calibrazione…

si può inserire un IF nel ciclo for:

```
for change in feed:
    if change["old_val"] is None and change["new_val"] is not None:
        print("NEW ENTRY:", change["new_val"])
      print ("...executing calibration pipeline in background on HPC...")
      print ("...")
      print ("...")
```

oppure mettere direttamente il filtro nella query:

```
feed = r.table("events") \
        .changes() \
        .filter(lambda c:
            (c["old_val"] == None) & (c["new_val"] != None)
        ) \
        .run()
```

se non vuoi I record già presenti all'avvio:

```
feed = r.table("events") \
        .changes(include_initial=False) \
```

```
        .filter(lambda c: c["old_val"] == None) \
        .run()
```

Lo script listener.py diventerà:

```
==========
from rethinkdb import RethinkDB

r = RethinkDB()
r.connect( host='192.156.213.164', port=28015, db='test', user='redbteam',
password='pa
ssword.01' ).repl()

#all changes
#feed = r.table("events").changes().run()
print("Listening for changes...")

#only new changes
feed = r.table("events") \
        .changes() \
        .filter(lambda c:
            (c["old_val"] == None) & (c["new_val"] != None)
        ) \
        .run()

#exclude old records
feed = r.table("events") \
        .changes(include_initial=False) \
        .filter(lambda c: c["old_val"] == None) \
        .run()

for change in feed:
    new = change.get("new_val")
    if new:  # insert o update
        print(
            "exec_calpipe -> "
            + str(new['type'])
            + "="
            + str(new['value'])
            + " [" + str(new['size']) + "MB] ..."
        )

# solo nuove righe e non updates
# feed = r.table("events").changes(include_initial=False).run()

==========
```

L'OUTPUT dei due script (con listener aggiornato):

```
(venv) root@amasdb06:/home/cedadm# python producer.py
Inserted: b91dd366-7b74-49c4-b40c-d32b7fca24b0
updated: b91dd366-7b74-49c4-b40c-d32b7fca24b0
deleted: b91dd366-7b74-49c4-b40c-d32b7fca24b0
Inserted: bfbaa066-ab89-44d8-901d-2ec4c478ae85
updated: bfbaa066-ab89-44d8-901d-2ec4c478ae85
deleted: bfbaa066-ab89-44d8-901d-2ec4c478ae85
Inserted: fed98689-0056-49bb-9155-49af3098d8b0
updated: fed98689-0056-49bb-9155-49af3098d8b0
deleted: fed98689-0056-49bb-9155-49af3098d8b0
Inserted: 01aaa11f-f9f4-4102-88df-aea217ade0ba
updated: 01aaa11f-f9f4-4102-88df-aea217ade0ba
deleted: 01aaa11f-f9f4-4102-88df-aea217ade0ba
Inserted: 65b8d159-b4f3-4041-9579-099fde9ab050
```

```
exec_calib_pipe -> filepath=/tmp/ea/d3/ead3e8c1-4d07-414b-8a83-c32a5c9583c9.json [23MB]
...
exec_calib_pipe -> filepath=/tmp/56/03/56037caf-a090-44c9-bfe6-0de34e18200f.json [11MB]
...
exec_calib_pipe -> filepath=/tmp/c4/11/c4114ae7-2e39-4876-b51e-440f302d5acb.json [27MB]
...
exec_calib_pipe -> filepath=/tmp/4a/50/4a502353-e954-4024-a959-c7b1a17aef86.json [39MB]
...
exec_calib_pipe -> filepath=/tmp/90/b1/90b15db5-bda1-4faf-8879-4a5f1342c9e8.json [25MB]
...
exec_calib_pipe -> filepath=/tmp/4c/56/4c56a1b4-10b8-4cde-98ed-d3432e0853d9.json [29MB]
...
exec_calib_pipe -> filepath=/tmp/80/fb/80fb629d-c6cb-43d5-9212-30a986a002c6.json [27MB]
...
```

TEST FINALE DI AUTOAPPRENDIMENTO:

https://forms.gle/dm6FtoFhAbnfMXTx5

Riceverete le risposte alla fine del test ed il certificato successivamente con mail personale.

N.B. Il corso e' gratuito ma la riproduzione di questo materiale e' sogetta a preventiva autorizzazione.
Inviare le eventuali richieste a
stefano.gallozzi@inaf.it