Operations and Monitoring Guide

SRE 3.3

Table of Contents

1 Introduction

This guide provides an overview of the different maintenance commands that can be performed on SRE to manage the platform. Procedures are also detailed to perform backup and restore operations.

2 Process Management

SRE software processes are controlled by the *supervisord* daemon. The systemctl service name to start, stop and request status of the SRE software is *sre*.

Besides these standard service operations, it is possible to connect directly to the running supervisord instance by launching directly **/opt/sre/bin/supervisorctl**. Once connected to supervisord, several commands are available, as shown with the help command.

```
supervisor> help
default commands (type help <topic>):
=====================================
add exit open reload restart start tail
avail fg pid remove shutdown status update
clear maintail quit reread signal stop version
```

The current status of the processes can be obtained by using the status command.

```
supervisor> status
sre-REST STOPPED Not started
sre-call-processor:0 RUNNING pid 6671, uptime 0:48:58
sre-gui STOPPED Not started
sre-manager STOPPED Not started
```

The current status can also be obtained by looking at the SRE GUI Dashboard, as shown in the figure below.

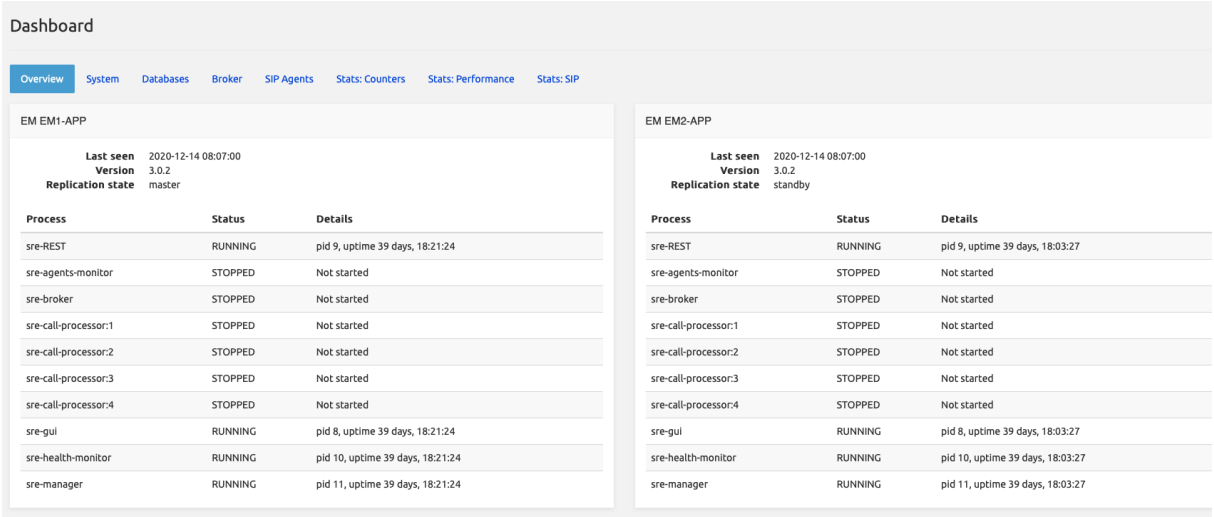


Figure 1: Graphical user interface, application Description automatically generated

On start, *supervisord* reads its configuration file (/opt/sre/etc/supervisord-program.conf) to select which programs must be started. It is possible to overrule this configuration by manually starting or stopping processes.

A single process can be restarted with the restart <program> command.

```
supervisor> restart sre-manager
sre-manager: stopped
sre-manager: started
```

A single process can be stopped with the stop <program> command.

```
supervisor> stop sre-manager
sre-manager: stopped
```

A single process can be started with the start <program> command.

```
supervisor> start sre-manager
sre-manager: started
```

The *supervisord* configuration can be reloaded with the reload command. This operation stops all the processes and they are restarted according to the supervisord configuration file. In particular, if a process has been manually started while it is not active in the configuration, this process will not start after the reload operation.

```
supervisor> reload
Really restart the remote supervisord process y/N? y
Restarted supervisord
```

It is possible to read what a process outputs on its standard output with the tail <program>command.

```
supervisor> tail sre-manager
```

3 Services Management

3.1 SRE

SRE service can be stopped/started/restarted using the following commands

```
[root@sre-em1 ~]# systemctl stop sre
[root@sre-em1 ~]# systemctl start sre
[root@sre-em1 ~]# systemctl restart sre
```

3.2 PostgreSQL

PostgreSQL service can be stopped/started/restarted using the following commands

```
[root@sre-em1 ~]# systemctl stop postgresql-14
[root@sre-em1 ~]# systemctl start postgresql-14
[root@sre-em1 ~]# systemctl restart postgresql-14
```

3.3 InfluxDB

InfluxDB service can be stopped/started/restarted using the following commands

```
[root@sre-em ~]# systemctl stop influxd
[root@sre-em ~]# systemctl start influxd
[root@sre-em ~]# systemctl restart influxd
```

3.4 Kamailio

Kamailio service can be stopped/started/restarted using the following commands

```
[root@sre-cp1 ~]# systemctl stop kamailio
[root@sre-cp1 ~]# systemctl start kamailio
[root@sre-cp1 ~]# systemctl restart kamailio
```

3.5 Mongo

Mongo service can be stopped/started/restarted using the following commands

```
[root@sre-cp1 ~]# systemctl stop mongod
[root@sre-cp1 ~]# systemctl start mongod
[root@sre-cp1 ~]# systemctl restart mongod
```

4 Monitoring

This section describes several key indicators of the system health. These indicators should be monitored by external monitoring systems to trigger alarms in case of issues.

Note

Some of these monitoring commands rely on queries run against the PostgreSQL database with the psql CLI tool. In case these tasks should be scripted, the output format can be adapted to ease parsing of the results. In particular, the option -t does not print the headers, the option -A does not align the table output and the option -R allows to define the separator. Other output format options can be obtained by running /usr/pgsql-<version>/bin/psql --help. The output samples in the following sections are provided with the full output, to better illustrate the output data. Alternatively, these queries can be run remotely on a PostgreSQL connection, provided that the access rights allow them.

4.1 Filesystems Monitoring

These filesystems should be monitored through SRE Alarming and optionally through external scripts:

- /: there must be enough space on the root filesystem to allow normal operations of system services, PostgreSQL, Kamailio and SRE software. Alarm threshold for disk usage should be set on maximum 75%.
- /var/lib/pgsql: (if existing) there must be enough space (< 75%) for PostgreSQL
- /var/log/: logs are rotated daily and size should remain stable under standard log levels. Alarm threshold for disk usage should be set on maximum 90%.
- /data/sre/db/backups: automated backups should not fill the filesystem. Alarm threshold for disk usage should be set on maximum 90%.

- /data/sre/db/wals: archived work-ahead-logs are essential for backup recovery. Alarm threshold for disk usage should be set on maximum 90%.
- /data/sre/db/provisioning: sufficient disk space must be retained to keep an history of provisioning and ensure that automatic NPACT synchronization does not block. Alarm threshold for disk usage should be set on maximum 80%.
- /data/sre/accounting: on EM nodes, sufficient disk space must be retained to be able to collect events from CP nodes and consequently produce CDRs. Alarm threshold for disk usage should be set on maximum 80%.
- /var/lib/mongo: on all nodes, sufficient disk space must be retained to be able to store call counters for CAC. Alarm threshold for disk usage should be set on maximum 80%.

The **"df -k"** command provide you file system usage information

```
[root@sre-em1 ~]# df -k
Filesystem           1K-blocks    Used Available Use% Mounted on
devtmpfs               1928240       0   1928240   0% /dev
tmpfs                  1940072     204   1939868   1% /dev/shm
tmpfs                  1940072  178504   1761568  10% /run
tmpfs                  1940072       0   1940072   0% /sys/fs/cgroup
/dev/mapper/cl-root   20503120 3866652  15571920  20% /
/dev/sda1               999320  151016    779492  17% /boot
/dev/mapper/data-sre  51470816  661184  48172016   2% /data/sre
/dev/mapper/cl-var    10190100 4282252   5367176  45% /var
tmpfs                   388016       0    388016   0% /run/user/0
```

4.2 Logs Cleaning

An alarm "full disk usage" may be related to the /var/log/sre directory (logs are filling in the file system).

You can run the following command to clean that directory:

```
[root@sre-em1 ~]# cd /var/log/sre
[root@sre-em1 sre]# rm -f *.1
[root@sre-em1 sre]# rm -f *.2
[root@sre-em1 sre]# rm -f *.3
[root@sre-em1 sre]# rm -f *.4
[root@sre-em1 sre]# rm -f *.5
[root@sre-em1 sre]# rm -f *.6
[root@sre-em1 sre]# rm -f *.7
[root@sre-em1 sre]# rm -f *.8
[root@sre-em1 sre]# rm -f *.9
```

```
[root@sre-em1 sre]# rm -f *.10
```

Be aware than doing that means that you will loose the logs history.

4.3 Memory and CPU Usage Monitoring

Memory and CPU usage consumption can be monitored using the **top** command.

```
Tasks: 247 total,   1 running, 246 sleeping,   0 stopped,   0 zombie
%Cpu(s): 50,0 us,  3,1 sy,  0,0 ni, 43,8 id,  0,0 wa,  0,0 hi,  0,0 si,  3,1 st
KiB Mem :  3880144 total,   253148 free,  1468636 used,  2158360 buff/cache
KiB Swap:  1048572 total,   760308 free,   288264 used.  1818724 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
26429 influxdb  20   0 1560028 239292  74696 S  80,0  6,2  20982:59 influxd
 3297 sre       20   0 1646556 156956   6652 S  13,3  4,0   4363:27 sre-health-
    ↪ moni
    9 root      20   0       0      0      0 S   6,7  0,0 673:47.40 rcu_sched
28612 root      20   0  162244   2368   1548 R   6,7  0,1   0:00.01 top
    1 root      20   0  125640   2940   1628 S   0,0  0,1  53:17.18 systemd
    2 root      20   0       0      0      0 S   0,0  0,0   0:04.12 kthreadd
    4 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kworker/0:0
    ↪ H
    6 root      20   0       0      0      0 S   0,0  0,0  25:55.80 ksoftirqd/0
    7 root      rt   0       0      0      0 S   0,0  0,0   0:01.36 migration/0
    8 root      20   0       0      0      0 S   0,0  0,0   0:00.00 rcu_bh
   10 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 lru-add-
    ↪ drain
   11 root      rt   0       0      0      0 S   0,0  0,0   2:24.84 watchdog/0
   12 root      rt   0       0      0      0 S   0,0  0,0   1:44.89 watchdog/1
   13 root      rt   0       0      0      0 S   0,0  0,0   0:06.07 migration/1
   14 root      20   0       0      0      0 S   0,0  0,0  13:13.88 ksoftirqd/1
   16 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kworker/1:0
    ↪ H
   18 root      20   0       0      0      0 S   0,0  0,0   0:00.00 kdevtmpfs
   19 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 netns
   20 root      20   0       0      0      0 S   0,0  0,0   0:19.20 khungtaskd
   21 root       0 -20       0      0      0 S   0,0  0,0   0:00.06 writeback
   22 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kintegrityd
   23 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 bioset
   24 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 bioset
   25 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 bioset
   26 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kblockd
   27 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 md
   28 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 edac-poller
   29 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 watchdogd
```

```
35    35 root      20   0       0      0      0 S   0,0  0,0   7:41.33 kswapd0
36    36 root      25   5       0      0      0 S   0,0  0,0   0:00.00 ksmd
37    37 root      39  19       0      0      0 S   0,0  0,0   1:15.34 khugepaged
38    38 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 crypto
39    46 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kthrotld
40    48 root       0 -20       0      0      0 S   0,0  0,0   0:00.00
      ↪ kmpath_rdacd
41    49 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kaluad
42    51 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kpsmoused
43    53 root       0 -20       0      0      0 S   0,0  0,0   0:00.00
      ↪ ipv6_addrconf
```

The output of the top command provide you valuable information like:

- CPU in %
- RAM memory usage
- The list of processes ordered by their consumption

4.4 SRE Software Monitoring

The SRE software can be monitored at several levels: processes and stats.

4.4.1 SRE Process Monitoring

The operational status of processes running on SRE can be monitored by using the ps command and “grepping” on the process name. All SRE processes start with the string /opt/sre/bin/python /opt/sre/bin/sre-.

```
1 [root@sre-em1 ~]# pgrep -a -f "/opt/sre/bin/python"
2 3283 /opt/sre/bin/python /opt/sre/bin/supervisord -n
3 3294 /opt/sre/bin/python /opt/sre/bin/sre-REST
4 3295 /opt/sre/bin/python /opt/sre/bin/sre-cdr-collector
5 3296 /opt/sre/bin/python /opt/sre/bin/sre-gui
6 3297 /opt/sre/bin/python /opt/sre/bin/sre-health-monitor
7 3298 /opt/sre/bin/python /opt/sre/bin/sre-http-processor
8 19487 /opt/sre/bin/python /opt/sre/bin/sre-manager
```

The administrative status of the processes can be monitored with the supervisorctl tool, as all SRE processes are managed by supervisord. You can also retrieve the status of the SRE processes by polling the sre service status:

```
1 [root@sre-em1 ~]# /opt/sre/bin/supervisorctl status
```

```
sre-REST                         RUNNING   pid 3294, uptime 46 days, 17:00:24
sre-agents-monitor               STOPPED   Not started
sre-broker                       STOPPED   Not started
sre-call-processor:0             STOPPED   Not started
sre-cdr-collector                RUNNING   pid 3295, uptime 46 days, 17:00:24
sre-cdr-postprocessor            STOPPED   Not started
sre-cdr-sender                   STOPPED   Not started
sre-dns-updater                  STOPPED   Not started
sre-enum-processor               STOPPED   Not started
sre-gui                          RUNNING   pid 3296, uptime 46 days, 17:00:24
sre-health-monitor               RUNNING   pid 3297, uptime 46 days, 17:00:24
sre-http-processor               RUNNING   pid 3298, uptime 46 days, 17:00:24
sre-manager                      RUNNING   pid 19487, uptime 3 days, 5:56:38
```

On a typical deployment, the processes to check are respectively:

- Element Manager:
  - sre-REST
  - sre-gui
  - sre-manager
  - sre-health-monitor
  - sre-cdr-collector
- Call Processor:
  - SIP interface: sre-agents-monitor
  - SIP interface: sre-broker
  - SIP interface: sre-call-processor: 0-N (the number of processes might be different depending on the supervisord configuration file)
  - HTTP interface: sre-http-processor
  - ENUM interface: sre-enum-processor
  - sre-health-monitor
  - sre-cdr-sender

4.5 Stats Monitoring

Near real-time stats are kept in InfluxDB.

Counters of occurrences of events are stored also in the *counter.csv* file, in /var/log/sre. Each record is composed of the fields:

- *hostname*: node which generated the event
- *stat name*: counter (event) identifier. It can represent system resources stats, or nodes in the Service Logic(s), or number of outcome form SRE (relay/redirect/serviceLogicError/sipResponse/…)
- *timestamp (60-sec aligned) in human format*: timestamp of the minute for which event occurred.
- *timestamp (60-sec aligned):* Unix timestamp (seconds since EPOCH) of the minute for which event occurred (this value is always a multiple of 60).
- *values*: this covers the following 15 fields. Each field contains the total number of occurrences of this counter type during this window of 1 minute, from the most recent one to the least recent. For instance, the first value contains the number of occurrences at 14:41, the second one the number of occurrences at 14:42, and so on.
  These values will “shift to the right” every minute, as the file is refreshed with new stats every minute.

To provide an example, here is a possible content of the counter.csv

```
1 [root@sre-em1 ~]# more /var/log/sre/counters.csv
2 sre32-cp2-testbed,custom.fleg_relay,2022-05-10T16
      ↪ :31:00,1652193060,,,,,,,,,,,,,,,
3 sre32-cp2-testbed,profiling.cp.CAC test.503,2022-05-10T16
      ↪ :31:00,1652193060,,,,,,,,,,,,,,,
4 sre32-cp2-testbed,profiling.cp.CAC test.Extract Contacts,2022-05-10T16
      ↪ :31:00,1652193060,12,11,13,12,12,12,12,12,12,12,12,12,12,12,12
5 sre32-cp2-testbed,profiling.cp.CAC test.Extract mContacts,2022-05-10T16
      ↪ :31:00,1652193060,,,,,,,,,,,,,,,
6 sre32-cp2-testbed,profiling.cp.CAC test.Start,2022-05-10T16
      ↪ :31:00,1652193060,12,11,13,12,12,12,12,12,12,12,12,12,12,12,12
7 sre32-cp2-testbed,profiling.cp.CAC test.add counter0,2022-05-10T16
      ↪ :31:00,1652193060,12,11,13,12,12,12,12,12,12,12,12,12,12,12,12
8 sre32-cp2-testbed,profiling.cp.CAC test.check CAC,2022-05-10T16
      ↪ :31:00,1652193060,12,11,13,12,12,12,12,12,12,12,12,12,12,12,12
9 sre32-cp2-testbed,profiling.cp.CAC test.register CAC,2022-05-10T16
      ↪ :31:00,1652193060,12,11,13,12,12,12,12,12,12,12,12,12,12,12,12
10 sre32-cp2-testbed,profiling.cp.CAC test.relay msg,2022-05-10T16
      ↪ :31:00,1652193060,12,11,13,12,12,12,12,12,12,12,12,12,12,12,12
11 sre32-cp2-testbed,profiling.cp.CAC test.remove t,2022-05-10T16
      ↪ :31:00,1652193060,12,11,13,12,12,12,12,12,12,12,12,12,12,12,12
12 sre32-cp2-testbed,profiling.cp.CAC test.replace To,2022-05-10T16
      ↪ :31:00,1652193060,12,11,13,12,12,12,12,12,12,12,12,12,12,12,12
13 sre32-cp2-testbed,profiling.cp.CAC test.set a and b,2022-05-10T16
      ↪ :31:00,1652193060,12,11,13,12,12,12,12,12,12,12,12,12,12,12,12
```

```
14 sre32-cp2-testbed,request.INVITE,2022-05-10T16
      ↪ :31:00,1652193060,12,11,13,12,12,12,12,12,12,12,12,12,12,12,12
15 sre32-cp2-testbed,response.503,2022-05-10T16:31:00,1652193060,,,,,,,,,,,,,
16 sre32-cp2-testbed,response.loop,2022-05-10T16:31:00,1652193060,,,,,,,,,,,,,
17 sre32-cp2-testbed,response.relay,2022-05-10T16
      ↪ :31:00,1652193060,12,11,13,12,12,12,12,12,12,12,12,12,12,12,12
18 sre32-cp2-testbed,response.serviceLogicError,2022-05-10T16
      ↪ :31:00,1652193060,,,,,,,,,,,,,,,
```

Counters of interest are described in the following table.

Counter name	Description
request.INVITE	INVITE requests
request.OPTIONS	OPTIONS requests
response.redirect	Redirect responses (301/302)
response.loop	Loop responses (482)
response.serviceLogicError	Service Logic Error responses (604)
response.serviceDown	Service Down responses (503)
response.genericError	Generic Error responses (500)
response.genericError	Generic Error responses (500)
request.http.<method>	HTTP requests
response.http.<code>	HTTP responses
request.dns.NAPTR	NAPTR requests
response.dns.NOERROR	Successful DNS responses
response.dns.SERVFAIL	Failed DNS responses

In addition, the file *samples.csv* will provide in the values fields the average processing time for each event (based on the formula: sum of duration of events occurred / number of events (e.g. processing time for INVITE's).

An example of samples.csv is provided here below:

```
1 [root@sre-em1 ~]# more /var/log/sre/samples.csv
2 sre32-cp2-testbed,accounting.openCalls,2022-05-10T16
      ↪ :32:01,1652193121,3800.000,3800.000,3800.000,3700.000,3566.667,3833.333,3833.333,3866.6
      ↪
```

```
sre32-cp2-testbed,profiling.cp.CAC test.503,2022-05-10T16:32:01,1652193121,,,,,,,,,,,,,,,
sre32-cp2-testbed,profiling.cp.CAC test.Extract Contacts,2022-05-10T16:32:01,1652193121,0.113,0.121,0.103,0.115,0.123,0.114,0.111,0.117,0.121,0.110,0.115,0.1
sre32-cp2-testbed,profiling.cp.CAC test.Extract mContacts,2022-05-10T16:32:01,1652193121,,,,,,,,,,,,,,,
sre32-cp2-testbed,profiling.cp.CAC test.Start,2022-05-10T16:32:01,1652193121,0.076,0.079,0.064,0.080,0.080,0.083,0.062,0.081,0.087,0.072,0.075,0.0
sre32-cp2-testbed,profiling.cp.CAC test.add counter0,2022-05-10T16:32:01,1652193121,0.064,0.063,0.066,0.059,0.057,0.067,0.052,0.060,0.070,0.064,0.076,0.0
sre32-cp2-testbed,profiling.cp.CAC test.check CAC,2022-05-10T16:32:01,1652193121,4.674,3.809,4.454,4.630,7.464,5.426,9.562,3.780,4.517,7.097,5.204,4.3
sre32-cp2-testbed,profiling.cp.CAC test.register CAC,2022-05-10T16:32:01,1652193121,0.050,0.049,0.053,0.049,0.050,0.060,0.048,0.050,0.049,0.049,0.052,0.0
sre32-cp2-testbed,profiling.cp.CAC test.relay msg,2022-05-10T16:32:01,1652193121,0.097,0.088,0.080,0.085,0.088,0.092,0.079,0.081,0.085,0.082,0.092,0.0
sre32-cp2-testbed,profiling.cp.CAC test.remove t,2022-05-10T16:32:01,1652193121,0.041,0.045,0.045,0.042,0.050,0.052,0.036,0.039,0.042,0.041,0.045,0.0
sre32-cp2-testbed,profiling.cp.CAC test.replace To,2022-05-10T16:32:01,1652193121,0.054,0.051,0.055,0.048,0.059,0.064,0.049,0.048,0.054,0.046,0.055,0.0
sre32-cp2-testbed,profiling.cp.CAC test.set a and b,2022-05-10T16:32:01,1652193121,0.081,0.085,0.071,0.087,0.088,0.080,0.078,0.084,0.081,0.064,0.083,0.0
sre32-cp2-testbed,profiling.cp.INVITE,2022-05-10T16:32:01,1652193121,13.214,10.280,13.943,13.042,19.519,13.832,19.339,10.739,15.154,16.718
sre32-cp2-testbed,profiling.cp.loop,2022-05-10T16:32:01,1652193121,0.836,0.078,0.722,0.668,2.033,1.148,0.823,0.086,1.377,1.157,0.881,1.3
sre32-cp1-testbed,system.cpu,2022-05-10T16:32:01,1652193121,11718.333,11660.000,11665.000,11911.667,12000.000,11545.000,11658.333
sre32-cp2-testbed,system.cpu,2022-05-10T16:32:01,1652193121,11463.333,11186.667,11150.000,11330.000,11451.667,11265.000,11330.000
sre32-em1-testbed,system.cpu,2022-05-10T16:32:01,1652193121,10006.122,9622.449,10550.000,9261.224,10508.163,9285.417,9935.417,951
```

```
sre32-em2-testbed,system.cpu,2022-05-10T16
    ↪ :32:01,1652193121,10224.074,10170.909,10077.778,10127.778,10260.000,10033.333,10049.091
    ↪
sre32-cp1-testbed,system.disk./,2022-05-10T16
    ↪ :32:01,1652193121,67900.000,67900.000,67900.000,67900.000,67900.000,67900.000,67900.000
    ↪
sre32-cp2-testbed,system.disk./,2022-05-10T16
    ↪ :32:01,1652193121,69600.000,69600.000,69600.000,69600.000,69600.000,69600.000,69600.000
    ↪
sre32-em1-testbed,system.disk./,2022-05-10T16
    ↪ :32:01,1652193121,76400.000,76400.000,76400.000,76400.000,76400.000,76400.000,76400.000
    ↪
sre32-em2-testbed,system.disk./,2022-05-10T16
    ↪ :32:01,1652193121,67500.000,67500.000,67500.000,67500.000,67500.000,67500.000,67500.000
    ↪
sre32-cp1-testbed,system.disk./boot,2022-05-10T16
    ↪ :32:01,1652193121,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000
    ↪
sre32-cp2-testbed,system.disk./boot,2022-05-10T16
    ↪ :32:01,1652193121,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000
    ↪
sre32-em1-testbed,system.disk./boot,2022-05-10T16
    ↪ :32:01,1652193121,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000
    ↪
sre32-em2-testbed,system.disk./boot,2022-05-10T16
    ↪ :32:01,1652193121,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000
    ↪
sre32-cp1-testbed,system.disk./boot/efi,2022-05-10T16
    ↪ :32:01,1652193121,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.0
    ↪
sre32-cp2-testbed,system.disk./boot/efi,2022-05-10T16
    ↪ :32:01,1652193121,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.0
    ↪
sre32-em1-testbed,system.disk./boot/efi,2022-05-10T16
    ↪ :32:01,1652193121,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.0
    ↪
sre32-em2-testbed,system.disk./boot/efi,2022-05-10T16
    ↪ :32:01,1652193121,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.0
    ↪
sre32-cp1-testbed,system.disk./data/sre/db/backups,2022-05-10T16
    ↪ :32:01,1652193121,17100.000,17100.000,17100.000,17100.000,17100.000,17100.000,17100.000
    ↪
sre32-cp2-testbed,system.disk./data/sre/db/backups,2022-05-10T16
    ↪ :32:01,1652193121,17100.000,17100.000,17100.000,17100.000,17100.000,17100.000,17100.000
    ↪
sre32-em1-testbed,system.disk./data/sre/db/backups,2022-05-10T16
```

```
      ↪ :32:01,1652193121,10400.000,10400.000,10400.000,10400.000,10400.000,10400.000,10400.000
      ↪
35 sre32-em2-testbed,system.disk./data/sre/db/backups,2022-05-10T16
      ↪ :32:01,1652193121,17300.000,17300.000,17300.000,17300.000,17300.000,17300.000,17300.000
      ↪
36 sre32-cp1-testbed,system.disk./data/sre/db/main,2022-05-10T16
      ↪ :32:01,1652193121,600.000,600.000,600.000,600.000,600.000,600.000,600.000,600.000,600.0
      ↪
37 sre32-cp2-testbed,system.disk./data/sre/db/main,2022-05-10T16
      ↪ :32:01,1652193121,600.000,600.000,600.000,600.000,600.000,600.000,600.000,600.000,600.0
      ↪
38 sre32-em1-testbed,system.disk./data/sre/db/main,2022-05-10T16
      ↪ :32:01,1652193121,600.000,600.000,600.000,600.000,600.000,600.000,600.000,600.000,600.0
      ↪
39 sre32-em2-testbed,system.disk./data/sre/db/main,2022-05-10T16
      ↪ :32:01,1652193121,600.000,600.000,600.000,600.000,600.000,600.000,600.000,600.000,600.0
      ↪
40 sre32-cp1-testbed,system.disk./data/sre/db/tablespace_a,2022-05-10T16
      ↪ :32:01,1652193121,900.000,900.000,900.000,900.000,900.000,900.000,900.000,900.000,900.0
      ↪
41 sre32-cp2-testbed,system.disk./data/sre/db/tablespace_a,2022-05-10T16
      ↪ :32:01,1652193121,900.000,900.000,900.000,900.000,900.000,900.000,900.000,900.000,900.0
      ↪
42 sre32-em1-testbed,system.disk./data/sre/db/tablespace_a,2022-05-10T16
      ↪ :32:01,1652193121,900.000,900.000,900.000,900.000,900.000,900.000,900.000,900.000,900.0
      ↪
43 sre32-em2-testbed,system.disk./data/sre/db/tablespace_a,2022-05-10T16
      ↪ :32:01,1652193121,900.000,900.000,900.000,900.000,900.000,900.000,900.000,900.000,900.0
      ↪
44 sre32-cp1-testbed,system.disk./data/sre/db/tablespace_b,2022-05-10T16
      ↪ :32:01,1652193121,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.0
      ↪
45 sre32-cp2-testbed,system.disk./data/sre/db/tablespace_b,2022-05-10T16
      ↪ :32:01,1652193121,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.0
      ↪
46 sre32-em1-testbed,system.disk./data/sre/db/tablespace_b,2022-05-10T16
      ↪ :32:01,1652193121,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.0
      ↪
47 sre32-em2-testbed,system.disk./data/sre/db/tablespace_b,2022-05-10T16
      ↪ :32:01,1652193121,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.0
      ↪
48 sre32-cp1-testbed,system.disk./data/sre/db/wals,2022-05-10T16
      ↪ :32:01,1652193121,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.0
      ↪
49 sre32-cp2-testbed,system.disk./data/sre/db/wals,2022-05-10T16
      ↪ :32:01,1652193121,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.0
```

```
      ↪
50 sre32-em1-testbed,system.disk./data/sre/db/wals,2022-05-10T16
      ↪ :32:01,1652193121,1900.000,1900.000,1900.000,1900.000,1900.000,1900.000,1900.000,1900.0
      ↪
51 sre32-em2-testbed,system.disk./data/sre/db/wals,2022-05-10T16
      ↪ :32:01,1652193121,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.0
      ↪
52 sre32-cp1-testbed,system.disk./data/sre/provisioning,2022-05-10T16
      ↪ :32:01,1652193121,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.0
      ↪
53 sre32-cp2-testbed,system.disk./data/sre/provisioning,2022-05-10T16
      ↪ :32:01,1652193121,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.0
      ↪
54 sre32-em1-testbed,system.disk./data/sre/provisioning,2022-05-10T16
      ↪ :32:01,1652193121,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.0
      ↪
55 sre32-em2-testbed,system.disk./data/sre/provisioning,2022-05-10T16
      ↪ :32:01,1652193121,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.0
      ↪
56 sre32-cp1-testbed,system.disk./opt,2022-05-10T16
      ↪ :32:01,1652193121,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000
      ↪
57 sre32-cp2-testbed,system.disk./opt,2022-05-10T16
      ↪ :32:01,1652193121,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000
      ↪
58 sre32-em1-testbed,system.disk./opt,2022-05-10T16
      ↪ :32:01,1652193121,35000.000,35000.000,35000.000,35000.000,35000.000,35000.000,35000.000
      ↪
59 sre32-em2-testbed,system.disk./opt,2022-05-10T16
      ↪ :32:01,1652193121,34900.000,34900.000,34900.000,34900.000,34900.000,34900.000,34900.000
      ↪
60 sre32-cp1-testbed,system.disk./var/log,2022-05-10T16
      ↪ :32:01,1652193121,44426.667,44500.000,44500.000,44496.667,44400.000,44400.000,44400.000
      ↪
61 sre32-cp2-testbed,system.disk./var/log,2022-05-10T16
      ↪ :32:01,1652193121,39906.667,39916.667,39916.667,39900.000,39910.000,39900.000,39905.000
      ↪
62 sre32-em1-testbed,system.disk./var/log,2022-05-10T16
      ↪ :32:01,1652193121,40967.347,40900.000,40900.000,40900.000,40900.000,40831.250,40800.000
      ↪
63 sre32-em2-testbed,system.disk./var/log,2022-05-10T16
      ↪ :32:01,1652193121,62600.000,62600.000,62600.000,62600.000,62600.000,62600.000,62600.000
      ↪
64 sre32-cp1-testbed,system.mem.ram,2022-05-10T16
      ↪ :32:01,1652193121,51891.667,52018.333,52100.000,52100.000,52098.333,52100.000,52100.000
      ↪
```

```
sre32-cp2-testbed,system.mem.ram,2022-05-10T16
    ↪ :32:01,1652193121,46798.333,46798.333,46733.333,46595.000,46593.333,46580.000,46588.333
    ↪
sre32-em1-testbed,system.mem.ram,2022-05-10T16
    ↪ :32:01,1652193121,59100.000,59100.000,59100.000,59100.000,59097.959,59100.000,59039.583
    ↪
sre32-em2-testbed,system.mem.ram,2022-05-10T16
    ↪ :32:01,1652193121,34618.519,34601.818,34609.259,34605.556,34605.455,34611.111,34607.273
    ↪
sre32-cp1-testbed,system.mem.swap,2022-05-10T16
    ↪ :32:01,1652193121,3200.000,3200.000,3200.000,3200.000,3200.000,3200.000,3200.000,3200.0
    ↪
sre32-cp2-testbed,system.mem.swap,2022-05-10T16
    ↪ :32:01,1652193121,2500.000,2500.000,2500.000,2500.000,2500.000,2500.000,2500.000,2500.0
    ↪
sre32-em1-testbed,system.mem.swap,2022-05-10T16
    ↪ :32:01,1652193121,5600.000,5600.000,5600.000,5600.000,5600.000,5600.000,5600.000,5600.0
    ↪
sre32-em2-testbed,system.mem.swap,2022-05-10T16
    ↪ :32:01,1652193121,1200.000,1200.000,1200.000,1200.000,1200.000,1200.000,1200.000,1200.0
    ↪
```

Samples of interest are described in the following table.

Sample name	Description
profiling.cp.INVITE	Duration to process INVITE requests
profiling.cp.OPTIONS	Duration to process OPTIONS requests
profiling.cp.loop	Duration to perform loop detection
profiling.enum.NAPTR	Duration to process NAPTR requests
profiling.http.<method>	Duration to process HTTP requests
accounting.openCalls	Calls opened in the last minute

4.5.1 InfluxDB query

Regularly, stats collected by the sre-manager are dumped to the internal Influx database.

Counters of occurrences of events of the last minute can be shown with this command:

```
[root@sre-em1 ~]# influx query 'from(bucket: "counters")|> range(start: -1m)|>
    ↪ drop(columns: ["_start", "_stop", "_field"])' | grep request.OPTIONS
```

```
2                     sip         request.OPTIONS              SRE-33-CP1
      ↪ 2023-10-18T12:18:44.000000000Z                             1
3                     sip         request.OPTIONS              SRE-33-CP2
      ↪ 2023-10-18T12:18:09.000000000Z                             1
```

Cumulative time of the duration of events and the number of such occurrences are stored in the *samples* bucket. Each record is composed of the fields:

- *hostname*: node which generated the event
- *_measurement*: event name.
- *_time*: event timestamp.
- *elapsed_time*: sum of the durations of the single events.
- *occurrences*: total number of occurrences of this event type during this window of 1 minute.

Dividing *elapsed_time* by *occurrences* for a record computes the average duration of such an event.

Samples of events of the last 10 seconds can be shown with this command:

```
1 [root@sre-em1 ~]# influx query 'from(bucket: "samples")|> range(start: -10s)|>
      ↪ drop(columns: ["_start", "_stop"])' | grep profiling.cp.INVITE
2             elapsed_time    profiling.cp.INVITE              SRE-33-CP1
      ↪ 2023-10-18T12:20:50.000000000Z                                0.212
3             occurences    profiling.cp.INVITE              SRE-33-CP1
      ↪ 2023-10-18T12:20:50.000000000Z                             2
```

4.6 PostgreSQL Monitoring

4.6.1 Service Monitoring

To verify the status of PostgreSQL, execute the command *systemctl status postgresql-14*.

```
1 [root@sre-em ~]# systemctl status postgresql-14
2 - postgresql-14.service - PostgreSQL 14 database server
3    Loaded: loaded (/usr/lib/systemd/system/postgresql-14.service; enabled;
      ↪ vendor preset: disabled)
4    Active: active (running) since ven 2023-04-28 10:11:14 CEST; 5 months 20
      ↪ days ago
5      Docs: https://www.postgresql.org/docs/14/static/
6  Main PID: 21254 (postmaster)
7    CGroup: /system.slice/postgresql-14.service
8            3353 postgres: sre sre 127.0.0.1(38618) idle
9            3501 postgres: postgres postgres 127.0.0.1(39082) idle
10           5775 postgres: sre sre 127.0.0.1(50674) idle
```

```
11            5778 postgres: sre sre 127.0.0.1(50688) idle
12            5781 postgres: sre stirshaken_a 127.0.0.1(50704) idle
13            5782 postgres: sre stirshaken_b 127.0.0.1(50706) idle
14            5783 postgres: sre regression_test_08_12_22_a 127.0.0.1(50708) idle
15            5784 postgres: sre regression_test_08_12_22_b 127.0.0.1(50716) idle
```

4.6.2 Process Monitoring

The master process, postmaster should be present on all nodes.

```
1 [root@sre-em ~]# ps -ef|grep postmaster|grep -v grep
2 postgres 21254     1  0 apr28 ?        04:53:27 /usr/pgsql-14/bin/postmaster -D
      ↪  /var/lib/pgsql/14/data/
```

On the **master** node, there should be a number of work-ahead logs senders equal to the number of nodes replicating from the master (e.g. standby EM and 4 CP nodes). Beware that the streaming ID (that is, the current "screenshot" of the DB should be the same on all nodes, unless the synchronization has been stopped on one or more nodes on purpose).

```
1 [root@sre-em ~]# ps -ef|grep "postgres: walsender"|grep -v grep
2 postgres 16401 21254  0 apr28 ?        00:19:45 postgres: walsender repmgr
      ↪ 10.0.161.181(44576) streaming 2/B8264AB0
3 postgres 21300 21254  0 apr28 ?        00:26:17 postgres: walsender repmgr
      ↪ 10.0.161.183(48710) streaming 2/B8264AB0
4 postgres 21306 21254  0 apr28 ?        00:20:02 postgres: walsender repmgr
      ↪ 10.0.161.182(57050) streaming 2/B8264AB0
```

On the **standby** nodes, there should be exactly one work-ahead log receiver.

```
1 [root@sre-cp ~]# ps -ef|grep "postgres: walreceiver"|grep -v grep
2 postgres  2444  2436  0 apr28 ?        03:32:36 postgres: walreceiver streaming
      ↪  2/B8264AB0
```

The number of open connections from the sre user should remain stable. If the number of connections increases over time, this might be an indication that the sessions are not correctly ended by the SRE software.

```
1 [root@sre-cp ~]# ps -ef|grep "postgres: sre"|grep -v grep
2 postgres 10216  2436  0 set01 ?        00:00:00 postgres: sre sre
      ↪ 127.0.0.1(37400) idle
3 postgres 10221  2436  0 set01 ?        00:00:00 postgres: sre sre
      ↪ 127.0.0.1(37426) idle
4 postgres 10224  2436  0 set01 ?        00:00:00 postgres: sre stirshaken_a
      ↪ 127.0.0.1(37432) idle
```

```
postgres 10225  2436  0 set01 ?        00:00:00 postgres: sre stirshaken_b
    ↪ 127.0.0.1(37434) idle
postgres 10226  2436  0 set01 ?        00:00:00 postgres: sre
    ↪ regression_test_08_12_22_a 127.0.0.1(37436) idle
postgres 10227  2436  0 set01 ?        00:00:00 postgres: sre
    ↪ regression_test_08_12_22_b 127.0.0.1(37438) idle
```

Processes handling active transactions can be counted by “grepping” on the string “in transaction”. This number should be stable over time.

```
[root@sre-cp ~]# ps -ef|grep "postgres: sre"|grep -c "in transaction"
1
```

Idle connections are the other ones.

```
[root@sre-cp ~]# ps -ef|grep "postgres: sre"|grep -c idle
59
```

4.6.3 Replication

Replication status is shown in the *GUI->Dashboard->Database*.

The status can be also checked with the following queries. The result f (false) indicates that the node is not replicating, so is master. The result t (true) indicates that the node is replicating from a master node.

On the master EM:

```
[root@sre-em ~]# /usr/pgsql-14/bin/psql -U postgres -h 127.0.0.1 -c "select *
    ↪ from pg_is_in_recovery()"
 pg_is_in_recovery
-------------------
 f
(1 row)
```

The number of clients connected to replicate the databases can be retrieved by querying the pg_stat_replication.

The state field can be used in the query to differentiate between streaming replication (normal mode of operation for all nodes) and backup replication (result of an ongoing backup activity). This table is only present on the master PostgreSQL instance.

On the master EM:

```
[root@sre.em ~]# /usr/pgsql-14/bin/psql -U postgres -h 127.0.0.1 -c "select *
    ↪ from pg_stat_replication"
```

```
 pid  | usesysid | usename | application_name | client_addr  | client_hostname | client_port |         backend_start         | backend_xmin |   state   |  sent_lsn  | write_lsn  | flush_lsn  | replay_lsn |    write_lag    |    flush_lag    |   replay_lag    | sync_priority | sync_state |          reply_time
-------+----------+---------+------------------+--------------+-----------------+-------------+-------------------------------+--------------+-----------+------------+------------+------------+------------+-----------------+-----------------+-----------------+---------------+------------+-------------------------------
 21300 |    16389 | repmgr  | cp2              | 10.0.161.183 |                 |       48710 | 2023-04-28 10:11:15.927884+02 |              | streaming | 2/B827AF38 | 2/B827AF38 | 2/B827AF38 | 2/B827AF38 | 00:00:00.001141 | 00:00:00.002427 | 00:00:00.002952 |             0 | async      | 2023-10-18 11:01:45.525076+02
 21306 |    16389 | repmgr  | cp1              | 10.0.161.182 |                 |       57050 | 2023-04-28 10:11:16.291374+02 |              | streaming | 2/B827AF38 | 2/B827AF38 | 2/B827AF38 | 2/B827AF38 | 00:00:00.001203 | 00:00:00.002264 | 00:00:00.002315 |             0 | async      | 2023-10-18 11:01:45.524272+02
 16401 |    16389 | repmgr  | em2              | 10.0.161.181 |                 |       44576 | 2023-04-28 11:59:13.512262+02 |              | streaming | 2/B827AF38 | 2/B827AF38 | 2/B827AF38 | 2/B827AF38 | 00:00:00.001277 | 00:00:00.002492 | 00:00:00.002493 |             0 | async      | 2023-10-18 11:01:45.524472+02
(3 rows)
```

4.6.4 WAL files

WAL (work-ahead log) files are transferred from the master to the standby nodes to replicate the data stored in each table.

Checking the WAL files on the master and standby nodes provide indication about the replication status. The content of the directory */var/lib/pgsql/14/data/pg_wal/* on the stanby nodes must be the same as on the active nodes. You should also see the latest WAL files being updated. This can be seen using the following command:

```
[root@SRE-33-EM1 ~]# ls -ltr /var/lib/pgsql/14/data/pg_wal/
totale 573468
-rw-------  1 postgres postgres      343 20 gen  2023 00000009.history
-rw-------  1 postgres postgres      386 20 gen  2023 0000000A.history
-rw-------  1 postgres postgres      430 20 gen  2023 0000000B.history
-rw-------  1 postgres postgres      474 20 gen  2023 0000000C.history
-rw-------  1 postgres postgres      518 20 gen  2023 0000000D.history
-rw-------  1 postgres postgres 16777216 20 ago 03.17 0000000D00000002000000B9
-rw-------  1 postgres postgres 16777216 22 ago 22.56 0000000D00000002000000BA
```

```
-rw-------  1 postgres postgres 16777216 25 ago 23.36 0000000D000000020000000BB
-rw-------  1 postgres postgres 16777216 27 ago 03.15 0000000D00000002000000099
-rw-------  1 postgres postgres 16777216 27 ago 03.16 0000000D00000002000000009A
-rw-------  1 postgres postgres 16777216 30 ago 10.41 0000000D00000002000000009B
-rw-------  1 postgres postgres 16777216  2 set 16.11 0000000D00000002000000009C
-rw-------  1 postgres postgres 16777216  3 set 03.48 0000000D00000002000000009D
-rw-------  1 postgres postgres 16777216  3 set 03.49 0000000D00000002000000009E
-rw-------  1 postgres postgres 16777216  6 set 16.08 0000000D00000002000000009F
-rw-------  1 postgres postgres 16777216 10 set 02.20 0000000D000000020000000A0
-rw-------  1 postgres postgres 16777216 10 set 03.15 0000000D000000020000000A1
-rw-------  1 postgres postgres 16777216 10 set 03.17 0000000D000000020000000A2
-rw-------  1 postgres postgres 16777216 13 set 03.06 0000000D000000020000000A3
-rw-------  1 postgres postgres 16777216 15 set 08.31 0000000D000000020000000A4
-rw-------  1 postgres postgres 16777216 17 set 03.15 0000000D000000020000000A5
-rw-------  1 postgres postgres 16777216 17 set 03.17 0000000D000000020000000A6
-rw-------  1 postgres postgres 16777216 20 set 04.21 0000000D000000020000000A7
-rw-------  1 postgres postgres 16777216 23 set 02.08 0000000D000000020000000A8
-rw-------  1 postgres postgres 16777216 24 set 03.15 0000000D000000020000000A9
-rw-------  1 postgres postgres 16777216 24 set 03.17 0000000D000000020000000AA
-rw-------  1 postgres postgres 16777216 27 set 14.57 0000000D000000020000000AB
-rw-------  1 postgres postgres 16777216 30 set 00.36 0000000D000000020000000AC
-rw-------  1 postgres postgres 16777216  1 ott 03.15 0000000D000000020000000AD
-rw-------  1 postgres postgres 16777216  1 ott 03.17 0000000D000000020000000AE
-rw-------  1 postgres postgres 16777216  3 ott 20.01 0000000D000000020000000AF
-rw-------  1 postgres postgres 16777216  6 ott 17.22 0000000D000000020000000B0
-rw-------  1 postgres postgres 16777216  8 ott 03.15 0000000D000000020000000B1
-rw-------  1 postgres postgres 16777216  8 ott 03.18 0000000D000000020000000B2
-rw-------  1 postgres postgres 16777216 10 ott 16.41 0000000D000000020000000B3
-rw-------  1 postgres postgres 16777216 13 ott 04.37 0000000D000000020000000B4
-rw-------  1 postgres postgres 16777216 15 ott 03.15 0000000D000000020000000B5
-rw-------  1 postgres postgres 16777216 15 ott 03.17 0000000D000000020000000B6
-rw-------  1 postgres postgres      345 15 ott 03.17 0000000D000000020000000B6
    ↪ .00000028.backup
-rw-------  1 postgres postgres 16777216 17 ott 23.44 0000000D000000020000000B7
drwx------. 2 postgres postgres     4096 17 ott 23.46 archive_status
-rw-------  1 postgres postgres 16777216 18 ott 11.01 0000000D000000020000000B8
```

4.6.5 DB Disk Usage

Databases sizes (in bytes) can be retrieved in the GUI *Dashboard-> Databases*, or alternatively with the following query:

```
[root@SRE-33-EM1 ~]# /usr/pgsql-14/bin/psql -U postgres -h 127.0.0.1 -c "select
    ↪  datname, pg_database_size(datname) from pg_database"
                datname                 | pg_database_size
```

```
----------------------------------------+------------------
 postgres                               |          8979235
 template1                              |          8823299
 template0                              |          8823299
 repmgr                                 |          9257763
 sre                                    |         48988963
 temp_test_delete_me_a                  |          9151267
 temp_test_delete_me_b                  |          9118499
 temp_test_delete_me_please_a           |          9102115
 temp_test_delete_me_please_b           |          9077539
 test_default_value_a                   |          9044771
 test_default_value_b                   |          9044771
 test_a                                 |          9044771
 test_b                                 |          9044771
 test2_a                                |          9044771
 test2_b                                |          9044771
 sss_a                                  |          9093923
 sss_b                                  |          9044771
 stirshaken_a                           |          9167651
 stirshaken_b                           |          9044771
 test_versioning_a                      |          9069347
 test_versioning_b                      |          9044771
 test_versioning_2_a                    |          9069347
 test_versioning_2_b                    |          9069347
 regression_test_08_12_22_a             |          9224995
 regression_test_08_12_22_b             |          9143075
 m247_lab_a                             |          9585443
 m247_lab_b                             |          9044771
 fuse2_voice_a                          |          9044771
 fuse2_voice_b                          |          9044771
 dm_validation_1_a                      |          9298723
 dm_validation_1_b                      |          9044771
 demo_a                                 |          9110307
 demo_b                                 |          9044771
 demo_doc_versioning_position_a         |          9093923
 demo_doc_versioning_position_b         |          9044771
 demo_doc_export_dm_diagram_a           |          9044771
 demo_doc_export_dm_diagram_b           |          9069347
 inventory_a                            |          9216803
 inventory_b                            |          9044771
...

(61 rows)
```

Tablespaces sizes (in bytes) can be retrieved with this query.

```
[root@SRE-33-EM1 ~]# /usr/pgsql-14/bin/psql -U postgres -h 127.0.0.1 -c "select
```

```
      ↪  spcname, pg_tablespace_size(spcname) from pg_tablespace"
  spcname    | pg_tablespace_size
------------+--------------------
 pg_default |          594550039
 pg_global  |             622368
 main       |                  0
 version_a  |                  0
 version_b  |                  0
(5 rows)
```

4.7 Kamailio Monitoring

4.7.1 Process Monitoring (on CP Nodes Only)

Kamailio processes can be listed with the ps command. Their number should remain stable and they should not be continuously restarted (check the PID's).

On each CP node:

```
[root@sre-cp ~]# ps -ef|grep kamailio|grep -v grep
kamailio  7992     1  0 lug20 ?        00:00:32 /usr/sbin/kamailio -DD -P /run/
      ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8006  7992  0 lug20 ?        00:00:00 /usr/sbin/kamailio -DD -P /run/
      ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8007  7992  0 lug20 ?        00:00:00 /usr/sbin/kamailio -DD -P /run/
      ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8008  7992  0 lug20 ?        00:00:00 /usr/sbin/kamailio -DD -P /run/
      ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8009  7992  0 lug20 ?        00:00:00 /usr/sbin/kamailio -DD -P /run/
      ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8010  7992  0 lug20 ?        00:00:00 /usr/sbin/kamailio -DD -P /run/
      ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8011  7992  0 lug20 ?        00:00:00 /usr/sbin/kamailio -DD -P /run/
      ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8012  7992  0 lug20 ?        00:00:00 /usr/sbin/kamailio -DD -P /run/
      ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8013  7992  0 lug20 ?        00:00:00 /usr/sbin/kamailio -DD -P /run/
      ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8014  7992  0 lug20 ?        00:05:55 /usr/sbin/kamailio -DD -P /run/
      ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8015  7992  0 lug20 ?        00:05:41 /usr/sbin/kamailio -DD -P /run/
      ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8016  7992  0 lug20 ?        00:05:39 /usr/sbin/kamailio -DD -P /run/
      ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
```

```
kamailio  8017  7992  0 lug20 ?        00:05:40 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8018  7992  0 lug20 ?        00:05:27 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8019  7992  0 lug20 ?        00:05:32 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8020  7992  0 lug20 ?        00:05:25 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8021  7992  0 lug20 ?        00:05:53 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8022  7992  0 lug20 ?        00:14:58 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8023  7992  0 lug20 ?        01:17:49 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8024  7992  0 lug20 ?        00:05:52 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8025  7992  0 lug20 ?        00:13:57 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8026  7992  0 lug20 ?        00:00:00 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8027  7992  0 lug20 ?        00:11:12 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8028  7992  0 lug20 ?        00:00:37 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8029  7992  0 lug20 ?        00:03:42 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8030  7992  0 lug20 ?        00:03:44 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8031  7992  0 lug20 ?        00:03:41 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8032  7992  0 lug20 ?        00:03:41 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8033  7992  0 lug20 ?        00:03:40 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8034  7992  0 lug20 ?        00:03:39 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8035  7992  0 lug20 ?        00:03:42 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8036  7992  0 lug20 ?        00:03:42 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
kamailio  8037  7992  0 lug20 ?        00:02:14 /usr/sbin/kamailio -DD -P /run/
    ↪ kamailio/kamailio.pid -f /etc/kamailio/kamailio.cfg -m 64 -M 8
```

Detailed information about the role of each processes can be obtained with the *kamctl ps* command.

```
[root@sre-cp ~]# kamctl ps
{
```

```
  "jsonrpc":  "2.0",
  "result": [
    {
      "IDX":  0,
      "PID":  7992,
      "DSC":  "main process - attendant"
    }, {
      "IDX":  1,
      "PID":  8006,
      "DSC":  "udp receiver child=0 sock=127.0.0.1:5060"
    }, {
      "IDX":  2,
      "PID":  8007,
      "DSC":  "udp receiver child=1 sock=127.0.0.1:5060"
    }, {
      "IDX":  3,
      "PID":  8008,
      "DSC":  "udp receiver child=2 sock=127.0.0.1:5060"
    }, {
      "IDX":  4,
      "PID":  8009,
      "DSC":  "udp receiver child=3 sock=127.0.0.1:5060"
    }, {
      "IDX":  5,
      "PID":  8010,
      "DSC":  "udp receiver child=4 sock=127.0.0.1:5060"
    }, {
      "IDX":  6,
      "PID":  8011,
      "DSC":  "udp receiver child=5 sock=127.0.0.1:5060"
    }, {
      "IDX":  7,
      "PID":  8012,
      "DSC":  "udp receiver child=6 sock=127.0.0.1:5060"
    }, {
      "IDX":  8,
      "PID":  8013,
      "DSC":  "udp receiver child=7 sock=127.0.0.1:5060"
    }, {
      "IDX":  9,
      "PID":  8014,
      "DSC":  "udp receiver child=0 sock=10.0.161.182:5060"
    }, {
      "IDX":  10,
      "PID":  8015,
      "DSC":  "udp receiver child=1 sock=10.0.161.182:5060"
```

```
    }, {
      "IDX":  11,
      "PID":  8016,
      "DSC":  "udp receiver child=2 sock=10.0.161.182:5060"
    }, {
      "IDX":  12,
      "PID":  8017,
      "DSC":  "udp receiver child=3 sock=10.0.161.182:5060"
    }, {
      "IDX":  13,
      "PID":  8018,
      "DSC":  "udp receiver child=4 sock=10.0.161.182:5060"
    }, {
      "IDX":  14,
      "PID":  8019,
      "DSC":  "udp receiver child=5 sock=10.0.161.182:5060"
    }, {
      "IDX":  15,
      "PID":  8020,
      "DSC":  "udp receiver child=6 sock=10.0.161.182:5060"
    }, {
      "IDX":  16,
      "PID":  8021,
      "DSC":  "udp receiver child=7 sock=10.0.161.182:5060"
    }, {
      "IDX":  17,
      "PID":  8022,
      "DSC":  "slow timer"
    }, {
      "IDX":  18,
      "PID":  8023,
      "DSC":  "timer"
    }, {
      "IDX":  19,
      "PID":  8024,
      "DSC":  "secondary timer"
    }, {
      "IDX":  20,
      "PID":  8025,
      "DSC":  "JSONRPCS FIFO"
    }, {
      "IDX":  21,
      "PID":  8026,
      "DSC":  "JSONRPCS DATAGRAM"
    }, {
      "IDX":  22,
```

```
      "PID":  8027,
      "DSC":  "ctl handler"
    }, {
      "IDX":  23,
      "PID":  8028,
      "DSC":  "Dialog Clean Timer"
    }, {
      "IDX":  24,
      "PID":  8029,
      "DSC":  "tcp receiver (generic) child=0"
    }, {
      "IDX":  25,
      "PID":  8030,
      "DSC":  "tcp receiver (generic) child=1"
    }, {
      "IDX":  26,
      "PID":  8031,
      "DSC":  "tcp receiver (generic) child=2"
    }, {
      "IDX":  27,
      "PID":  8032,
      "DSC":  "tcp receiver (generic) child=3"
    }, {
      "IDX":  28,
      "PID":  8033,
      "DSC":  "tcp receiver (generic) child=4"
    }, {
      "IDX":  29,
      "PID":  8034,
      "DSC":  "tcp receiver (generic) child=5"
    }, {
      "IDX":  30,
      "PID":  8035,
      "DSC":  "tcp receiver (generic) child=6"
    }, {
      "IDX":  31,
      "PID":  8036,
      "DSC":  "tcp receiver (generic) child=7"
    }, {
      "IDX":  32,
      "PID":  8037,
      "DSC":  "tcp main process"
    }
  ],
  "id": 7588
}
```

4.7.2 Kamailio Stats Monitoring

Stats about Kamailio internals can be displayed with the kamctl stats command. By default, it displays stats for all groups. Individual groups stats can be retrieved with the kamctl stats <group> command (e.g. kamctl stats sl). Under normal operation, these counters should be increasing proportionally.

```
[root@sre-cp ~]# kamctl stats
{
  "jsonrpc":  "2.0",
  "result": [
    "app_python3:active_dialogs = 0",
    "app_python3:early_dialogs = 0",
    "app_python3:expired_dialogs = 118",
    "app_python3:failed_dialogs = 6",
    "app_python3:processed_dialogs = 76014",
    "core:bad_URIs_rcvd = 0",
    "core:bad_msg_hdr = 0",
    "core:drop_replies = 0",
    "core:drop_requests = 3",
    "core:err_replies = 0",
    "core:err_requests = 0",
    "core:fwd_replies = 132759",
    "core:fwd_requests = 1711587",
    "core:rcv_replies = 419914",
    "core:rcv_replies_18x = 65248",
    "core:rcv_replies_1xx = 108873",
    "core:rcv_replies_1xx_bye = 0",
    "core:rcv_replies_1xx_cancel = 0",
    "core:rcv_replies_1xx_invite = 108873",
    "core:rcv_replies_1xx_message = 0",
    "core:rcv_replies_1xx_prack = 0",
    "core:rcv_replies_1xx_refer = 0",
    "core:rcv_replies_1xx_reg = 0",
    "core:rcv_replies_1xx_update = 0",
    "core:rcv_replies_2xx = 310992",
    "core:rcv_replies_2xx_bye = 106582",
    "core:rcv_replies_2xx_cancel = 0",
    "core:rcv_replies_2xx_invite = 83786",
    "core:rcv_replies_2xx_message = 0",
    "core:rcv_replies_2xx_prack = 0",
    "core:rcv_replies_2xx_refer = 0",
    "core:rcv_replies_2xx_reg = 0",
    "core:rcv_replies_2xx_update = 0",
    "core:rcv_replies_3xx = 0",
    "core:rcv_replies_3xx_bye = 0",
    "core:rcv_replies_3xx_cancel = 0",
```

```
    "core:rcv_replies_3xx_invite = 0",
    "core:rcv_replies_3xx_message = 0",
    "core:rcv_replies_3xx_prack = 0",
    "core:rcv_replies_3xx_refer = 0",
    "core:rcv_replies_3xx_reg = 0",
    "core:rcv_replies_3xx_update = 0",
    "core:rcv_replies_401 = 0",
    "core:rcv_replies_404 = 0",
    "core:rcv_replies_407 = 0",
    "core:rcv_replies_480 = 0",
    "core:rcv_replies_486 = 0",
    "core:rcv_replies_4xx = 48",
    "core:rcv_replies_4xx_bye = 48",
    "core:rcv_replies_4xx_cancel = 0",
    "core:rcv_replies_4xx_invite = 0",
    "core:rcv_replies_4xx_message = 0",
    "core:rcv_replies_4xx_prack = 0",
    "core:rcv_replies_4xx_refer = 0",
    "core:rcv_replies_4xx_reg = 0",
    "core:rcv_replies_4xx_update = 0",
    "core:rcv_replies_5xx = 1",
    "core:rcv_replies_5xx_bye = 0",
    "core:rcv_replies_5xx_cancel = 0",
    "core:rcv_replies_5xx_invite = 0",
    "core:rcv_replies_5xx_message = 0",
    "core:rcv_replies_5xx_prack = 0",
    "core:rcv_replies_5xx_refer = 0",
    "core:rcv_replies_5xx_reg = 0",
    "core:rcv_replies_5xx_update = 0",
    "core:rcv_replies_6xx = 0",
    "core:rcv_replies_6xx_bye = 0",
    "core:rcv_replies_6xx_cancel = 0",
    "core:rcv_replies_6xx_invite = 0",
    "core:rcv_replies_6xx_message = 0",
    "core:rcv_replies_6xx_prack = 0",
    "core:rcv_replies_6xx_refer = 0",
    "core:rcv_replies_6xx_reg = 0",
    "core:rcv_replies_6xx_update = 0",
    "core:rcv_requests = 2039960",
    "core:rcv_requests_ack = 83991",
    "core:rcv_requests_bye = 112644",
    "core:rcv_requests_cancel = 5",
    "core:rcv_requests_info = 0",
    "core:rcv_requests_invite = 76022",
    "core:rcv_requests_message = 0",
    "core:rcv_requests_notify = 0",
```

```
    "core:rcv_requests_options = 1767298",
    "core:rcv_requests_prack = 0",
    "core:rcv_requests_publish = 0",
    "core:rcv_requests_refer = 0",
    "core:rcv_requests_register = 0",
    "core:rcv_requests_subscribe = 0",
    "core:rcv_requests_update = 0",
    "core:unsupported_methods = 0",
    "dns:failed_dns_request = 0",
    "dns:slow_dns_request = 0",
    "registrar:accepted_regs = 0",
    "registrar:default_expire = 3600",
    "registrar:default_expires_range = 0",
    "registrar:expires_range = 0",
    "registrar:max_contacts = 1",
    "registrar:max_expires = 3600",
    "registrar:rejected_regs = 0",
    "shmem:fragments = 6",
    "shmem:free_size = 64144248",
    "shmem:max_used_size = 7896000",
    "shmem:real_used_size = 2964616",
    "shmem:total_size = 67108864",
    "shmem:used_size = 2718232",
    "sl:1xx_replies = 0",
    "sl:200_replies = 0",
    "sl:202_replies = 0",
    "sl:2xx_replies = 0",
    "sl:300_replies = 0",
    "sl:301_replies = 0",
    "sl:302_replies = 0",
    "sl:3xx_replies = 0",
    "sl:400_replies = 0",
    "sl:401_replies = 0",
    "sl:403_replies = 0",
    "sl:404_replies = 0",
    "sl:407_replies = 0",
    "sl:408_replies = 0",
    "sl:483_replies = 0",
    "sl:4xx_replies = 0",
    "sl:500_replies = 0",
    "sl:5xx_replies = 4",
    "sl:6xx_replies = 0",
    "sl:failures = 0",
    "sl:received_ACKs = 3",
    "sl:sent_err_replies = 0",
    "sl:sent_replies = 76018",
```

```
    "sl:xxx_replies = 76014",
    "tcp:con_reset = 0",
    "tcp:con_timeout = 0",
    "tcp:connect_failed = 0",
    "tcp:connect_success = 0",
    "tcp:current_opened_connections = 0",
    "tcp:current_write_queue_size = 0",
    "tcp:established = 0",
    "tcp:local_reject = 0",
    "tcp:passive_open = 0",
    "tcp:send_timeout = 0",
    "tcp:sendq_full = 0",
    "tmx:2xx_transactions = 307515",
    "tmx:3xx_transactions = 0",
    "tmx:4xx_transactions = 12551",
    "tmx:5xx_transactions = 0",
    "tmx:6xx_transactions = 386",
    "tmx:UAC_transactions = 0",
    "tmx:UAS_transactions = 314069",
    "tmx:active_transactions = 0",
    "tmx:inuse_transactions = 0",
    "tmx:rpl_absorbed = 43649",
    "tmx:rpl_generated = 142170",
    "tmx:rpl_received = 287155",
    "tmx:rpl_relayed = 243506",
    "tmx:rpl_sent = 385676",
    "usrloc:location_contacts = 0",
    "usrloc:location_expires = 0",
    "usrloc:location_users = 0",
    "usrloc:registered_users = 0"
  ],
  "id": 8599
}
```


4.8 Mongo Monitoring

4.8.1 Service monitoring

Mongo service status can be retrieved with the following command

```
[root@sre-cp ~]# systemctl status mongod
- mongod.service - MongoDB Database Server
   Loaded: loaded (/usr/lib/systemd/system/mongod.service; enabled; vendor
    ↪ preset: disabled)
```

```
   Active: active (running) since gio 2023-01-12 15:12:45 CET; 9 months 4 days
    ↪ ago
     Docs: https://docs.mongodb.org/manual
 Main PID: 1258 (mongod)
   CGroup: /system.slice/mongod.service
           1258 /usr/bin/mongod -f /etc/mongod.conf

Warning: Journal has been rotated since unit was started. Log output is
    ↪ incomplete or unavailable.
```

4.8.2 Replica Set Status

The following command provide the following information (amongst many others)

- If nodes belong to replica set
- The other nodes belonging to that replica set
- The node which acting as primary

First enter the mongo CLI interface with:

```
[root@sre-cp ~]# mongo
MongoDB shell version v5.0.13
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&
    ↪ gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("31a2a49c-e4ad-4a5d-9764-898450fec607")
    ↪  }
MongoDB server version: 5.0.13
sre_location:SECONDARY>
```

Then use the command *rs.status()* in the CLI:

```
sre_location:SECONDARY> rs.status()
{
    "set" : "sre_location",
    "date" : ISODate("2023-10-18T09:11:22.901Z"),
    "myState" : 2,
    "term" : NumberLong(9),
    "syncSourceHost" : "10.0.161.183:27017",
    "syncSourceId" : 2,
    "heartbeatIntervalMillis" : NumberLong(2000),
    "majorityVoteCount" : 3,
    "writeMajorityCount" : 3,
    "votingMembersCount" : 4,
    "writableVotingMembersCount" : 4,
```

```
    "optimes" : {
        "lastCommittedOpTime" : {
            "ts" : Timestamp(1697620274, 1),
            "t" : NumberLong(9)
        },
        "lastCommittedWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
        "readConcernMajorityOpTime" : {
            "ts" : Timestamp(1697620274, 1),
            "t" : NumberLong(9)
        },
        "appliedOpTime" : {
            "ts" : Timestamp(1697620274, 1),
            "t" : NumberLong(9)
        },
        "durableOpTime" : {
            "ts" : Timestamp(1697620274, 1),
            "t" : NumberLong(9)
        },
        "lastAppliedWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
        "lastDurableWallTime" : ISODate("2023-10-18T09:11:14.341Z")
    },
    "lastStableRecoveryTimestamp" : Timestamp(1697620254, 1),
    "electionParticipantMetrics" : {
        "votedForCandidate" : true,
        "electionTerm" : NumberLong(9),
        "lastVoteDate" : ISODate("2023-08-04T14:02:09.274Z"),
        "electionCandidateMemberId" : 2,
        "voteReason" : "",
        "lastAppliedOpTimeAtElection" : {
            "ts" : Timestamp(1691157640, 1),
            "t" : NumberLong(8)
        },
        "maxAppliedOpTimeInSet" : {
            "ts" : Timestamp(1691157640, 1),
            "t" : NumberLong(8)
        },
        "priorityAtElection" : 1,
        "newTermStartDate" : ISODate("2023-08-04T14:02:13.753Z"),
        "newTermAppliedDate" : ISODate("2023-08-04T14:02:25.595Z")
    },
    "members" : [
        {
            "_id" : 0,
            "name" : "10.0.161.180:27017",
            "health" : 1,
            "state" : 2,
```

```
            "stateStr" : "SECONDARY",
            "uptime" : 4606205,
            "optime" : {
                "ts" : Timestamp(1697620274, 1),
                "t" : NumberLong(9)
            },
            "optimeDurable" : {
                "ts" : Timestamp(1697620274, 1),
                "t" : NumberLong(9)
            },
            "optimeDate" : ISODate("2023-10-18T09:11:14Z"),
            "optimeDurableDate" : ISODate("2023-10-18T09:11:14Z"),
            "lastAppliedWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
            "lastDurableWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
            "lastHeartbeat" : ISODate("2023-10-18T09:11:22.273Z"),
            "lastHeartbeatRecv" : ISODate("2023-10-18T09:11:20.959Z"),
            "pingMs" : NumberLong(0),
            "lastHeartbeatMessage" : "",
            "syncSourceHost" : "10.0.161.183:27017",
            "syncSourceId" : 2,
            "infoMessage" : "",
            "configVersion" : 1,
            "configTerm" : 9
        },
        {
            "_id" : 1,
            "name" : "10.0.161.182:27017",
            "health" : 1,
            "state" : 2,
            "stateStr" : "SECONDARY",
            "uptime" : 24087522,
            "optime" : {
                "ts" : Timestamp(1697620274, 1),
                "t" : NumberLong(9)
            },
            "optimeDate" : ISODate("2023-10-18T09:11:14Z"),
            "lastAppliedWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
            "lastDurableWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
            "syncSourceHost" : "10.0.161.183:27017",
            "syncSourceId" : 2,
            "infoMessage" : "",
            "configVersion" : 1,
            "configTerm" : 9,
            "self" : true,
            "lastHeartbeatMessage" : ""
        },
```

```
        {
            "_id" : 2,
            "name" : "10.0.161.183:27017",
            "health" : 1,
            "state" : 1,
            "stateStr" : "PRIMARY",
            "uptime" : 1588046,
            "optime" : {
                "ts" : Timestamp(1697620274, 1),
                "t" : NumberLong(9)
            },
            "optimeDurable" : {
                "ts" : Timestamp(1697620274, 1),
                "t" : NumberLong(9)
            },
            "optimeDate" : ISODate("2023-10-18T09:11:14Z"),
            "optimeDurableDate" : ISODate("2023-10-18T09:11:14Z"),
            "lastAppliedWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
            "lastDurableWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
            "lastHeartbeat" : ISODate("2023-10-18T09:11:21.792Z"),
            "lastHeartbeatRecv" : ISODate("2023-10-18T09:11:21.017Z"),
            "pingMs" : NumberLong(0),
            "lastHeartbeatMessage" : "",
            "syncSourceHost" : "",
            "syncSourceId" : -1,
            "infoMessage" : "",
            "electionTime" : Timestamp(1691157730, 1),
            "electionDate" : ISODate("2023-08-04T14:02:10Z"),
            "configVersion" : 1,
            "configTerm" : 9
        }
    ],
    "ok" : 1,
    "$clusterTime" : {
        "clusterTime" : Timestamp(1697620274, 1),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    },
    "operationTime" : Timestamp(1697620274, 1)
}
```

One cluster member should be in the *PRIMARY* state, while all the others should be in the *SECONDARY* state. The field *syncSourceHost* indicates, for each secondary server, from which server data is replicated.

4.9 Pacemaker Monitoring (if Implemented)

An SRE implementation may include a Clustering layer of CP (Kamailio) resources, obtained by means of a Pacemaker configuration.

When the CP Cluster is used, kamailio instances are not started directly through the kamailio service, instead they are controlled by pcs. It is therefore important not to start kamailio instances by service commands, rather do it from pcs commands. The pcs configuration provides twin resources:

- VIP (Virtual IP used by one of the CP in the cluster)
- SIP resource associated to a VIP

A Cluster can host multiple VIP+SIP resources, as long as each VIP and its associated SIP resource runs on the same node.

The resource agent kamailio is using SIPSAK as a mechanism to send SIP OPTIONS messages to SRE Call Processing Instances, therefore internal SIP OPTIONS messages are expected in SRE to poll resources availability.

To check the pcs configuration, run the following command:

```
[root@sre-cp ~]# pcs config show
```

To check the status of the configuration, as well as latest failure/timeout actions, run either:

```
[root@sre-cp ~]# pcs status
[root@sre-cp ~]# pcs cluster status
[root@sre-cp ~]# pcs status resources
```

Sample output:

```
[root@sre-cp1 ~]# pcs status
Cluster name: hacluster
Stack: corosync
Current DC: sre-cp2 (version 1.1.23-1.el7_9.1-9acf116022) - partition with
    ↪ quorum
Last updated: Wed Oct 18 11:16:08 2023
Last change: Thu Jul 20 16:21:13 2023 by root via cibadmin on sre-cp1

2 nodes configured
0 resource instances configured

Online: [ sre-cp1 sre-cp2 ]

Full list of resources:
Resource Group: Group1
```

```
ClusterIP1 (ocf::heartbeat:IPaddr2): Started sre-cp2
Kamailio1 (ocf::heartbeat:kamailio): Started sre-cp2

Resource Group: Group2
ClusterIP2 (ocf::heartbeat:IPaddr2): Started sre-cp1
Kamailio2 (ocf::heartbeat:kamailio): Started sre-cp1

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

4.9.1 Put a CP Code in Standby

The following command puts the specified node into standby mode. The specified node is no longer able to host resources. Any resources currently active on the node will be moved to another node. The

```
[root@sre-cp1 ~]# pcs cluster standby <node>
```

The following command removes the specified node from the standby mode.

```
[root@sre-cp1 ~]# pcs cluster unstandby <node>
```

The following command removes all nodes from the standby mode.

```
[root@sre-cp1 ~]# pcs cluster unstandby --all
```

4.9.2 Disabling a resource

The following command disables a resource. This command may be useful if we want to disable a virtual IP address.

```
[root@sre-cp1 ~]# pcs resource disable <resource_id>
```

The following command enables a resource. This command may be useful to put back a virtual IP address.

```
[root@sre-cp1 ~]# pcs resource enable <resource_id>
```

4.9.3 Restarting a Resource

The following command restarts a resource. This command may be useful if we want to restart kamailio (after modifying kamailio.cfg for example).

```
1 [root@sre-cp1 ~]# pcs resource restart <resource_id>
```

4.9.4 Moving a Resource

The following command moves a resource. This command may be useful if we want to move a virtual IP address.

```
1 [root@sre-cp1 ~]# pcs resource move <resource_id> <node>
```

Moving a resource means adding a constraint in the config in the background. To remove this constraint, we need to specify the constraint ID (displayed through “pcs config show”).

```
1 [root@sre-cp1 ~]# pcs constraint location remove <constraint ID>
```

4.9.5 Cluster Resources Cleanup

If a resource has failed or a move action didn’t succeed, a failure message appears when you display the cluster status. You can then clear that failure status with the pcs resource cleanup command. This command resets the resource status and failcount, telling the cluster to forget the operation history of a resource and re-detect its current state. The following command cleans up the resource specified by resource_id.

```
1 [root@sre-cp1 ~]# pcs resource cleanup <resource_id>
```

Note

If you do not specify a resource_id, this command resets the resource status and failcount for all resources, which results in a restart of all kamailio instances in the Cluster.

4.9.6 Deleting a Resource

```
1 [root@sre-cp1 ~]# pcs resource delete <resource_id>
```

4.9.7 Removing a Node

```
[root@sre-cp1 ~]# pcs cluster node remove <node>
```

4.9.8 Adding a Node

```
[root@sre-cp1 ~]# pcs cluster node add <node>
```

4.9.9 PCS Backup and Restore

Useful if a node is completely lost.

From a running node:

```
[root@sre-cp1 ~]# pcs cluster stop --all
[root@sre-cp1 ~]# pcs config backup backupfile
[root@sre-cp1 ~]# pcs config restore backupfile.pc.tar.bz2
[root@sre-cp1 ~]# pcs cluster start --all
[root@sre-cp1 ~]# pcs cluster enable --all
```

4.9.10 Log Files

You can check log files at

/var/log/cluster/corosync.log

5 Troubleshooting

In case of service outage, go first to the SRE GUI on any of the Element Manager nodes and check the statistics for each Call Processor nodes in the tab “Stats: Counters” of the Dashboard.

If you notice that the counter for the INVITE is equal to 0.00/sec on a Call Processor node where you expect traffic, connect in SSH to the Call Processor node:

- check the status of SRE process as described in section SRE Process Monitoring
- check the status of the kamailio processes as described in section Kamailio Monitoring.

If you notice a high response.genericError counter on a Call Processor, check the status of the PostgreSQL process on the Call Processor node, as described in section PostgreSQL Monitoring.

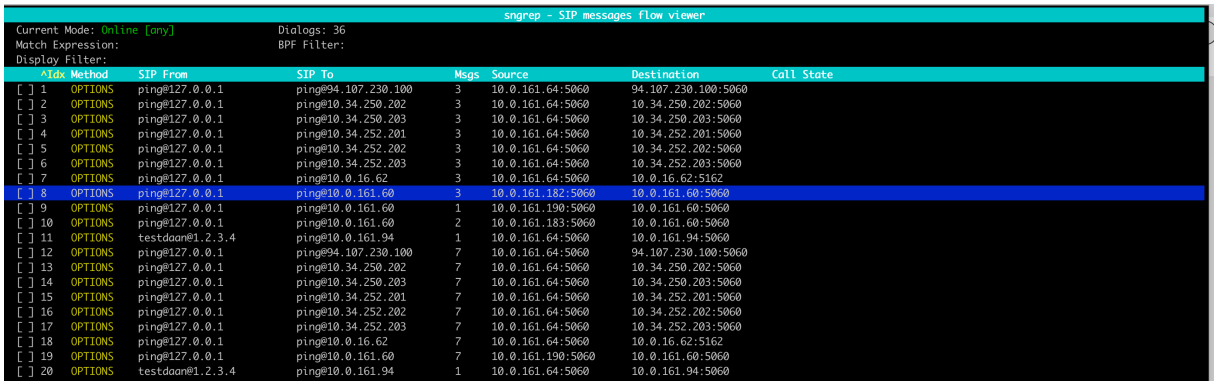
If all processes are running correctly, while the INVITE counter is null, there is possibly no SIP traffic arriving on the Call Processor node. If you expect traffic to hit the CP, check for any incoming SIP traffic: you can display all SIP messages arriving on the interface eth0 with the following command (the list of available interfaces can be retrieved from tshark -D).

```
[root@sre-cp ~]# tshark -i eth0 -R sip
Running as user \"root\" and group \"root\". This could be dangerous.
Capturing on eth0

0.769872938 10.211.1.1 -> 10.210.1.5 SIP 386 Request: OPTIONS
sip:10.210.1.5:5060
0.778437740 10.210.1.5 -> 10.211.1.1 SIP 430 Status: 200 OK
0.994916328 10.211.1.1 -> 10.210.1.3 SIP 386 Request: OPTIONS
sip:10.210.1.3:5060
1.000359472 10.210.1.3 -> 10.211.1.1 SIP 430 Status: 200 OK
...
```

If sngrep is installed on the CP nodes, this is a valid graphic alternative to tshark for sip traffic. Sngrep is a CLI based tool allowing to trace SIP messages. This can be very useful in order to troubleshoot issues. This can be used, for example, to troubleshoot agent monitoring. In case agent monitoring identifies an agent as down, you can check if SIP OPTION messages are sent and answer by this agent.

```
[root@sre-cp ~]# sngrep
```



If some processes are not running correctly on a Call Processor node, try to restart them.

- At first check the status of the PostgreSQL cluster and restart it if it is stopped, using the command service postgresql-14 {start|stop|status|restart}. Logging information can be found in the file /var/lib/pgsql/14/data/pg_log/postgresql-<Day>.log.
- Then check the status of the SRE using the command service sre status. On the Call Processor node, check that the *sre-call-processor* is RUNNING. Restart the SRE if needed, using the commands service sre {start|stop|status|restart}. Logging information can be found in /var/log/sre/.

- Then check the status of Kamailio. If it is stopped, restart it using the command service kamailio {start|stop|status|restart}. Logging information can be found in /var/log/messages.

Kamailio should listen on the UDP port **5060** (or different if configured in the kamailio.cfg files) for SIP requests, thus make sure that the CP are listening on the expected address:port

```
[root@sre-cp ~]# netstat -anu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 10.0.161.182:42235      0.0.0.0:*
udp        0      0 10.0.161.182:5405       0.0.0.0:*
udp        0      0 127.0.0.1:48531         0.0.0.0:*
udp        0      0 0.0.0.0:53              0.0.0.0:*
udp        0      0 10.0.161.182:55379      0.0.0.0:*
udp        0      0 127.0.0.1:323           0.0.0.0:*
udp        0      0 10.0.161.182:5060       0.0.0.0:*
udp        0      0 127.0.0.1:5060          0.0.0.0:*
udp6       0      0 ::1:323                 :::*
udp6       0      0 ::1:59940               ::1:59940               ESTABLISHED
```

For all the chain to be ready to host calls, the sre-broker should be listening on TCP port **5555** for requests originated by Kamailio that trigger the interface to SRE, also the PostgreSQL cluster should be listening on TCP port **5432**

```
[root@sre-cp ~]# netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:5432            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:8090          0.0.0.0:*               LISTEN
tcp        0      0 10.0.161.182:5060       0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:5060          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:9001          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:27017           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:6666          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:6000            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:5555          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:10004           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:53              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 10.0.161.182:48764      10.0.161.182:27017      ESTABLISHED
tcp        0      0 127.0.0.1:50198         127.0.0.1:9001          TIME_WAIT
tcp        0      0 127.0.0.1:49848         127.0.0.1:9001          TIME_WAIT
tcp        0      0 127.0.0.1:48462         127.0.0.1:6666          ESTABLISHED
tcp        0      0 10.0.161.182:39652      10.0.161.181:10000      ESTABLISHED
tcp        0      0 10.0.161.182:39386      10.0.161.183:27017      ESTABLISHED
```

```
tcp        0      0 127.0.0.1:48452         127.0.0.1:6666          ESTABLISHED
tcp        0      0 10.0.161.182:45534      10.0.161.180:10000      ESTABLISHED
tcp        0      0 127.0.0.1:5555          127.0.0.1:33690         ESTABLISHED
...
```

5.1 SRE Service Issues

At the application level, you might encounter issues related to the expected execution of a Service Logic. Such issues might be caused, for example, by misconfiguration of one or more nodes in a Service Logic, or by missing data in the user's data (Data Administration) used by the logic.

The Service Logic stats (Dashboard and Counters) will allow you to understand the size of the issue, namely how many times the response is an SRE-generated error or how often the logic traverses a node. At some point, you will need to either trace a call where the issue appears or reproduce it through the SIP Simulation. Both methods are suitable for understanding the exact part of the logic (node or group of nodes) that must be modified in order to obtain the desired behavior.

In order to activate tracing-flow traces, there are 2 conditions:

- the log-level of "Call tracing service logic flow" must be set to DEBUG
- the Tracing criteria (calling and called ranges) must match the ones of the call

Warning

While it is not a problem in lab environments, in production networks the tracing capability will reduce the CP performance, therefore it is recommended to not activate it in high-traffic conditions, and to limit the Tracing criteria to match exactly the calling/called ranges of interest.

When a Trace is produced, the tracing flow logs are available:

- in the GUI, on the active Service Logic (and its sub-service logics), under the Trace tab
- in the CLI, in the log file /var/log/sre/sre.log: you can grep on the string "tracing.flow"

To provide an example of an issue that can be noticed at the application level, 604 responses from the SRE would be the result of service logic exceptions in the SRE (e.g. a query node is failing due to unexpected inputs/outputs). In order to isolate those errors, please activate tracing for calls which end up in a 604 message, and check /var/log/sre/sre.log, which indicates which is the exception and which is the latest node traversed (where the exception is occurring). Further information might be obtained by the /var/log/sre/sre-call-processor.out.log in the CP.

5.2 SRE Logs

SRE provides application logs per-channel, that is, per functionality, which are available in the EM and CP nodes, at /var/log/sre:

Log type	File (in /var/log/sre)	Node type
Generic logs (including tracing logs collected from the CP)	sre.log	EM
Accounting	accounting.log	EM
CDR Sender	sre-cdr-sender.out.log	CP
CDR Collector	sre-cdr-collector.out.log	EM
CDR Post-processing	accounting-post-processing.log	EM
Audit	audit.log	EM
Service Logic Execution	service-logic-execution.log	EM
GUI logs	sre-gui.out.log	EM
Health monitor	sre-health-monitor.out.log	EM and CP
Manager	sre-manager.out.log	EM
REST API	sre-REST.out.log	EM
Supervisord	supervisord.log	EM and CP
ENUM processor	sre-enum-processor.out.log	CP
HTTP processor	sre-http-processor.out.log	CP
Interface (between Kamailio and SRE core)	interface.log	CP
SIP Agents monitor	sre-agents-monitor.out.log	CP
Broker	sre-broker.out.log	CP

Logs are rotated daily and kept on a 7-day circular buffer.

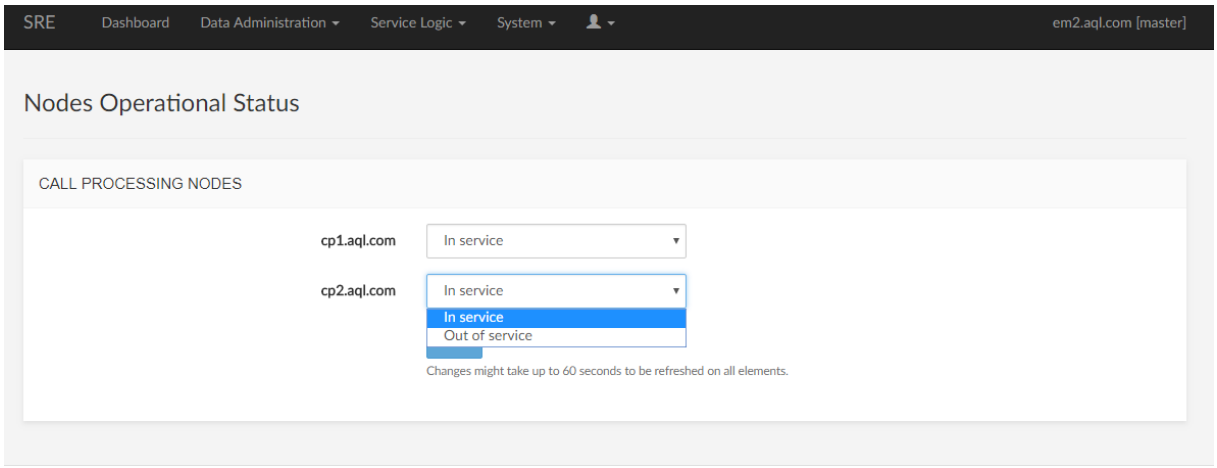
6 Nodes Operational Status

6.1 Putting a CP Out of Service

The page *System -> Nodes Operational Status* allows the operator to modify the operational state of a CP node. For each node, the operator can put the node in service (default) or out of service.

If, for any reason, the GUI is not available or the setting cannot be saved (e.g. no master PostgreSQL instance, ...), this setting can be overridden by creating an empty file named */tmp/cp.oos* on the CP node to disable (e.g. *touch /tmp/cp.oos*). The presence of such file has always priority on the configuration setting.

Once a node is put out of service, it will answer to both INVITE requests and OPTIONS requests with a SIP response *503 Service Unavailable*.



7 Backup & Restore Procedure

The purpose of this section is to describe the manual backup procedure. We also explain how to restore the master PostgreSQL cluster from the backup.

It is important to note that the backup and the restore procedures can only be applied on the master PostgreSQL cluster.

Several backup & restore procedures are available:

- Full database backup: This method creates a DB backup including all data and configuration (e.g. system configuration, privileges, …). Upon restore, the other servers must be re-synchronized from the master server.

- Database dumps: This method creates one SQL dump of the data per database (including the SRE database and the services databases). Upon restore, only the specified database is restored. There is no-need to resynchronize from the master server.

Node re-synchronization: In the event that a standby server must be restored and that the master server is available, data is re-synchronized from the master server.

7.1 Choosing the Best Backup Strategy

Each backup methods feature different advantages, as described in the table below.

Method	Granularity	Backup Speed	Restore Speed
Full database backup	(full system)	++	++
Database dumps	++ (database-based)	+	+
Node re-synchronisation	- (current re-synchronisation master)	N/A	++

In case of server issue, as long as the master DB server is available, the server DB can be restored from the master server by performing a node re-synchronisation.

In case of human error, where data has been affected on all servers through replication, the database dumps offer a way to restore the specific DB (be it system configuration or services data) where the error occurred. The time-accuracy of the restore depends on how often these dumps are performed.

In case of loss of all servers or if restore speed is a concern, then the full database backup may offer the best option to restore the DB. The time-accuracy of the restore depends on how often these backups are performed.

7.2 Full Database Backup

The master PostgreSQL cluster is periodically and automatically backed up.

Nevertheless, on some occasions, the Operator may want to execute a manual backup. This operation can be safely executed while the database is running.

7.2.1 Backup Procedure

Note

It is not required to backup the standby PostgreSQL nodes as they can be recovered at any time from a master PostgreSQL node, by cloning them with the repmgr tool.

For this, connect as a postgres user on the master PostgreSQL cluster, and use command pg_basebackup to create a backup. The backup is stored as tar gzip files (one file per tablespace, and one file for the content of the directory /var/lib/pgsql/14/data). The option -D specifies the directory receiving the base backup. In the example below, the tar gzip files are stored in the directory backup-20190212.

```
[root@sre-em ~]# su - postgres
-bash-4.1$ pg_basebackup -h <master EM ip address> -U repmgr -F t -z -X f -D
    ↪ backup-20230212/
-bash-4.1$ ls backup-20230212/
16662.tar.gz 16663.tar.gz 16664.tar.gz base.tar.gz
```

The gzip backup file is named after the oid of the tablespace. The file base.tar.gz contains the root PostgreSQL directory.

To find back the name of the tablespace based on the oid, you need to look at the table pg_tablespace:

```
[root@sre-em ~]# su - postgres
-bash-4.2$ psql
psql (14.5)
Type "help" for help.

postgres=# SELECT oid,spcname FROM pg_tablespace;
  oid  |  spcname
-------+------------
  1663 | pg_default
  1664 | pg_global
 16386 | main
 16387 | version_a
 16388 | version_b
(5 rows)
```

7.2.2 Restore Procedure

A backup can be used to restore the content of the postgreSQL database.

Warning

The restore procedure should only be used when the complete cluster must be recovered. If a single node must be recovered and a master PostgreSQL node is available, this node can be more

easily recovered by using the repmgr tool to clone its database content from the current master PostgreSQL instance.

Before proceeding with the restore operation, stop the PostgreSQL server.

```
[root@sre-em ~]# systemctl stop sre
[root@sre-em ~]# systemctl stop postgresql-14
```

As postgres user, delete all data in the location directories of each tablespace. Delete also the data into the PostgreSQL root directory /var/lib/pgsql/14/data/.

```
[root@sre-em ~]# rm -rf /var/lib/pgsql/14/data/*
[root@sre-em ~]# rm -rf /data/sre/db/tablespace_a/*
[root@sre-em ~]# rm -rf /data/sre/db/tablespace_b/*
[root@sre-em ~]# rm -rf /data/sre/db/main/*
```

Then in each of these directories, gunzip the corresponding file, as user postgres.

```
[root@sre-em ~]# su - postgres
-bash-4.1$ cd /var/lib/pgsql/14/data/
-bash-4.1$ tar -zxvf /var/lib/pgsql/backup-20230212/base.tar.gz
-bash-4.1$ cd /data/sre/db/main/
-bash-4.1$ tar -zxvf /var/lib/pgsql/backup-20230212/16662.tar.gz
-bash-4.1$ cd /data/sre/db/tablespace_a/
-bash-4.1$ tar -zxvf /var/lib/pgsql/backup-20230212/16663.tar.gz
-bash-4.1$ cd /data/sre/db/tablespace_b/
-bash-4.1$ tar -zxvf /var/lib/pgsql/backup-20230212/16664.tar.gz
```

Restart the PostgreSQL cluster and the sre software.

```
[root@sre-em ~]# systemctl start postgresql-14
[root@sre-em ~]# systemctl start sre
```

7.3 Database Dumps

Database dumps can be performed on a running system. They can also be restored on a live system and the replication will pick up the modifications and stream them to the standby servers.

These dumps are performed automatically but can also be performed manually.

7.3.1 Backup Procedure

The backup can be performed by executing the pg_dump command and indicating the DB name to dump. This DB can either be the system DB (sre) holding the system configuration or a service DB

(<service-name> suffixed with _a or _b, depending on the version).

In this example, a backup of the DB mix_a (i.e. service mix, version A) is performed and stored in the file /data/sre/backup/ em1/db/dump/manual_backup:

```
1 [root@sre-em ~]# pg_dump -h <master EM ip address> -U repmgr -c mix_a > /data/
      ↪ sre/backup/em1/db/dump/manual_backup
```

The produced file contains the list of SQL statements to remove the current schema, create a new one and insert the data.

7.3.2 Restore Procedure

The restore of a single DB can be performed by launching the psql command in such a way to execute the SQL statements from the backup file created. This procedure can be executed on a live system.

In this example, the manual file previously created is used to rebuild schema and data for the DB mix_a:

```
1 [root@sre-em ~]# psql -h <master EM ip address>  -U repmgr mix_a < /data/sre/
      ↪ backup/em1/db/dump/manual_backup
```

8 Full System Backup and Restore Procedure

A full database backup is only useful in specific circumstances, namely when you need to restore a VM from an empty system, when a snapshot of the virtual environment is not available.

8.1 Creating a Full Backup File

Note

Once the backup is created, download and store it somewhere external to the server!

If the tar command fails because a file/directory is not present, check which file is missing and adjust the command (or delete that part if not needed).

Check that you can create the backup file in a partition that has enough disk space (1-2 GB).

Before launching the command, replace the placeholders with the actual directories/files relevant to your case. It might be the case that a specific deployment does not have all the components referenced in this document (for instance Mongo or rsyslog etc.). In such a case, for the command to

work properly, it's most likely needed to remove the part of the backup commands related to the not existing components.

The command to create a backup on a EM node is:

```
[root@sre-em ~]# tar zcf <backup_path_filename>.tar.gz /opt/sre/ \
/data/sre/accounting /etc/mongod.conf \
/etc/cron.d/<crontabfile> /var/log/sre/ \
/etc/repmgr/14/repmgr.conf /var/lib/pgsql/14/data/pg_hba.conf \
/var/lib/pgsql/14/data/postgresql.conf \
/data/sre/db/backups/<node>*/ \
/etc/sysconfig/network-scripts/ifcfg-eth*
```

The command to create a backup on a CP node is:

```
[root@sre-cp ~]# tar zcf <backup_path_filename>.tar.gz /opt/sre/ \
/data/sre/accounting /etc/mongod.conf \
/etc/cron.d/<crontabfile> /var/log/sre/ \
/etc/repmgr/14/repmgr.conf /etc/sysconfig/network-scripts/ifcfg-eth* \
/etc/kamailio/kamailio.cfg
```

The backup content is:

- /data/sre/db/backups/<node>*/: backup folder (can be different in some deployments, check the crontab in /etc/cron.d/<sre crontab file>
- /opt/sre/: SRE sw and configuration files
- /data/sre/accounting: CDRs
- /etc/mongod.conf: MongoDB config (if applicable)
- /etc/kamailio/kamailio.cfg: Kamailio config
- /etc/cron.d/*: crontab files that were configured for this host
- /var/log/sre/: SRE logs
- /etc/repmgr/14/repmgr.conf: Replication Manager config
- /var/lib/pgsql/14/data/pg_hba.conf: PostgreSQL access config
- /etc/rsyslog.conf: Rsyslog config (if applicable)

8.2 Fully Restoring a Server

Note

The restore procedure from a full backup should only be used when the complete cluster must be recovered. If a single node must be recovered and a master PostgreSQL node is available, this node should be recovered by using the repmgr tool to clone its database content from the current master PostgreSQL instance.

If the master node is down and cannot be recovered in an acceptable timeframe, the suggestion is to proceed with Master Switchover (see here) and re-synchronize the failed node once again available.

Make sure the host has CentOS/RedHat running and meets all the requirement to run SRE.

Adjust IP, DNS, NTP (check that the date is the same as in the other nodes). Add the following directory needed by MongoDB (if applicable to your deployment):

```
# mkdir /data/sre/location
# chown mongod.mongod /data/sre/location
# systemctl restart mongod
```

You need to be root and positioned in / to launch the restore:

```
# cd /
```

This will restore all files in the original directories:

```
# tar -zxvf /data/<backup_filename>
```

Then either:

a. if you are restoring a PostgreSQL standby node, you need to resynch the node using Node Re-Synchronisation.

b. if it's the PostgreSQL master node that you are attempting to restore, then follow Restore Procedure.

For MongoDB (in case you use it on that node for CAC or Registrar), restoring the /etc/mongod.conf should be sufficient for the platform to re-synch the restored node from the primary node (assuming conditions are met to have a primary node elected).

At the end of the re-synch, the restored node's mongo instance will appear as SECONDARY as one of the other previously available nodes has been promoted to PRIMARY.

8.3 Node Re-Synchronisation

Node re-synchronization is the preferred way to recover a standby node. As the node will be cloned from a master node, it ensures that the data is up-to-date and that the standby node immediately starts replicating from the master.

8.3.1 Backup Procedure

As the failed server is recovered from the master server, there is no specific backup operation to perform in advance. The failed machine can be recovered or re-installed from zero using the SRE Installation

Guide, and then following Restore procedure.

8.3.2 Restore Procedure (through DB Clone)

Prerequisite: the OS system is installed, base packages needed by SRE are installed (e.g. postgres, repmgr, mongo, ...) and SRE sw is installed.

Warning

Performing a DB clone not only clone the data but also the PostgreSQL configuration. Therefore, the existing DB configuration, if any, will be overwritten.

Stop the SRE service, then PostgreSQL.

```
[root@sre-cp ~]# systemctl stop sre
[root@sre-cp ~]# systemctl stop postgresql-14
```

Delete all content of the main PostgreSQL directory and the different tablespaces (if present).

```
[root@sre-cp ~]# rm -rf /var/lib/pgsql/14/data/*
[root@sre-cp ~]# rm -rf /data/sre/db/main/* (may not exist if deploy does not
    ↪ user tablespaces)
[root@sre-cp ~]# rm -rf /data/sre/db/tablespace_a/* (may not exist if deploy
    ↪ does not user tablespaces)
[root@sre-cp ~]# rm -rf /data/sre/db/tablespace_b/* (may not exist if deploy
    ↪ does not user tablespaces)
```

Clone the data from the master node, as user *postgres*. The parameter -h indicates the IP address of the master node.

```
[root@sre-cp ~]# su - postgres
-bash-4.1$ /usr/pgsql-14/bin/repmgr -h 10.0.10.45 -F -U repmgr -d repmgr -f /
    ↪ etc/repmgr/14/repmgr.conf standby clone
[2023-04-16 17:43:56] [NOTICE] Redirecting logging output to '/var/log/repmgr/
    ↪ repmgr-14.log'
-bash-4.1$ cat /var/log/repmgr/repmgr-14.log
[2023-04-14 17:52:05] [NOTICE] setting data directory to: /var/lib/pgsql/14/
    ↪ data
[2023-04-14 17:52:05] [HINT] use -D/--data-dir to explicitly specify a data
    ↪ directory
[2023-04-14 17:52:05] [NOTICE] starting backup (using pg_basebackup)...
[2023-04-14 17:52:05] [HINT] this may take some time; consider using the -c/--
    ↪ fast-checkpoint option
[2023-04-14 17:52:22] [NOTICE] standby clone (using pg_basebackup) complete
[2023-04-14 17:52:22] [NOTICE] you can now start your PostgreSQL server
```

```
[2023-04-14 17:52:22] [HINT] for example : /etc/init.d/postgresql start
```

Restart PostgreSQL, then SRE.

```
[root@sre-cp ~]# systemctl start postgresql-14
[root@sre-cp ~]# systemctl start sre
```

Next you need to force node registration with the following command, the parameter -h indicates the IP address of the master node.

```
[root@sre-cp ~]# su - postgres
-bash-4.1$ /usr/pgsql-14/bin/repmgr -f /etc/repmgr/14/repmgr.conf -h 10.0.10.45
    ↪  -U repmgr -d repmgr standby register --force
```

9 PostgreSQL Cluster Switchover

If, for any reason, the current master PostgreSQL instance is not available anymore and cannot be restored in a sensible time, a standby PostgreSQL instance can be promoted as master, usually the standby EM. This does not affect the application service and will restore the possibility to provision the system.

Warning

In order to promote a standby node as a master, and instruct the other nodes to follow the new master, it is critical to ensure that the node previously master stays down until all operations have been carried out. Also, if the node is down, it is important that the master node is not restored while carrying out this procedure, since at no time there can be more than one master node to which the standby nodes replicate.

If the platform is prepared with SSH keys exchanged between EMs and CPs on the postgres user (so that an EM can connect to a CP using SSH keys), then the promote command can at the same time instruct all CP nodes to follow the new EM master. This is the suggested procedure detailed here.

Alternatively, without SSH keys it's still possible for CP nodes to follow the new master, although this requires an explicit action on each CP node (see here).

9.1 Procedure with CP Nodes Automatically Following new Master

Connect on the standby to-become-master node (usually the standby EM) as *postgres* user. The first time you can do a dry run to ensure all the prerequisites are in place.

```
[root@sre-em ~]# su - postgres
bash-4.2$ /usr/pgsql-14/bin/repmgr -f /etc/repmgr/14/repmgr.conf standby
    ↪ promote --siblings-follow --dry-run
INFO: node is a standby
INFO: no active primary server found in this replication cluster
INFO: all sibling nodes are reachable via SSH
INFO: 2 walsenders required, 10 available
INFO: node will be promoted using the "pg_promote()" function
INFO: prerequisites for executing STANDBY PROMOTE are met
If the test is ok, you can proceed with the real standby promotion
(removing the --dry-run)
```

If no error are shown you can run the same command without dry-run flag.

```
[root@sre-em ~]# su - postgres
bash-4.1$ /usr/pgsql-14/bin/repmgr -f /etc/repmgr/14/repmgr.conf standby
    ↪ promote ---siblings-follow
NOTICE: promoting standby to primary
DETAIL: promoting server "sre-em" (ID: 2) using pg_promote()
NOTICE: waiting up to 60 seconds (parameter "promote_check_timeout") for
    ↪ promotion to complete
NOTICE: STANDBY PROMOTE successful
DETAIL: server "sre-em" (ID: 2) was successfully promoted to primary
NOTICE: executing STANDBY FOLLOW on 2 of 2 siblings
INFO: STANDBY FOLLOW successfully executed on all reachable sibling nodes
```

After promoting the server to master role, you can check that all CP nodes are following the new master by performing:

```
[root@em2 ~]# su - postgres
-bash-4.2$ /usr/pgsql-14/bin/repmgr -f /etc/repmgr/14/repmgr.conf cluster show
 ID | Name | Role    | Status    | Upstream | Location | Priority | Timeline |
    ↪ Connection string
----+------+---------+-----------+----------+----------+----------+----------+---------------
    ↪
 1  | em1  | primary |    failed | ?        | default  | 100      |          |
    ↪ host=10.0.161.180 dbname=repmgr user=repmgr
 2  | em2  | standby | * running |          | default  | 100      | 13       |
    ↪ host=10.0.161.181 dbname=repmgr user=repmgr
 3  | cp1  | standby |   running | em2      | default  | 0        | 13       |
    ↪ host=10.0.161.182 dbname=repmgr user=repmgr
 4  | cp2  | standby |   running | em2      | default  | 0        | 13       |
    ↪ host=10.0.161.183 dbname=repmgr user=repmgr

WARNING: following issues were detected
- unable to connect to node "em1" (ID: 1)
```

```
12 HINT: execute with --verbose option to see connection error messages
```

After promoting the server to master role, we can observe that two masters are present in the repl_nodes table. The old master is marked as inactive (the active parameter is set to f for id 1, name em1).

It is recommended to also restart the SRE software on CPs so that the database connection pool is re-initialized.

Proceed with restart of SRE on the new master with:

```
1 [root@sre-em2 ~]# systemctl restart sre
```

When the inactive node em1 becomes available again, follow the node-resynchronization in order to restore it in the DB cluster.

9.1.1 Switching Back to a Previously Master Node

You can choose to switch back to a previously master node, by:

- isolating the current master (em2 in the example) with a stop of the sre and postgres services

```
1 [root@sre-em2 ~]# systemctl stop sre
2 [root@sre-em2 ~]# systemctl stop postgresql-14
```

- promoting em1 to master with the option –siblings-follow or with explicit follow
- re-cloning the em2 to em1 (see here)
- restarting postgres on em2

```
1 [root@sre-em2 ~]# systemctl start postgresql-14
```

- forcing db registration of em2 to em1

```
1 [root@sre-em2 ~]# su - postgres
2 -bash-4.1$ /usr/pgsql-14/bin/repmgr -f /etc/repmgr/14/repmgr.conf -h
3 10.0.10.45 -U repmgr -d repmgr standby register --force
```

- restarting sre on em2

```
1 [root@sre-em2 ~]# systemctl restart sre
```

9.2 Procedure with Explicit CP Follow

Connect on the standby node (usually the standby EM) as *postgres* user and promote it as master.

```
[root@sre-em2 ~]#  su - postgres
-bash-4.1$ /usr/pgsql-14/bin/repmgr -f /etc/repmgr/14/repmgr.conf standby
    ↪ promote
```

Proceed with restart of SRE on the new master with:

```
[root@sre-em2 ~]# service sre restart
```

At this point, all standby nodes must be instructed to follow the new master.

On the CP nodes:

```
[root@sre-cp ~]# su - postgres
-bash-4.1\$ /usr/pgsql-14/bin/repmgr -f /etc/repmgr/14/repmgr.conf standby
    ↪ follow
```

The command instructs the PostgreSQL to restart to follow the new master node. It is recommended to also restart the SRE software so that the database connection pool is re-initialized. Interruption of service can be minimized by isolating the CP nodes, one at a time.

When the inactive node em1 becomes available again, follow the node-resynchronization in order to restore it in the DB cluster.

10 Node and Site Isolation

To isolate a node, or an entire site, from database updates, the admin needs to reconfigure the permissions linked to the Postgres database replication, so that the affected node / site doesn't get any more updates.

Also, if needed, the services (SIP / ENUM / HTTP) can be stopped so that the CP doesn't reply to such requests.

For the database isolation, namely on the master EM node the admin must reconfigure the file /var/lib/pgsql/14/data/pg_hba.conf and set the lines applicable to replication and repmgr of the affected nodes to "reject".

For example, to isolate the CP node with IP = 10.0.161.64, the pg_hba.conf should contain these lines:

host all sre 10.0.161.64/32 reject

host replication repmgr 10.0.161.64/32 reject

host repmgr repmgr 10.0.161.64/32 reject

The same applies to entire subnets, in order to isolate a full site.

The operation requires a restart of postgres on the master EM node:

```
1 # systemctl restart postgresql-14
```

From the restart, any change done on the master EM node is not replicated on the isolated node / site.

To suppress node/site isolation and restore normal system operations, the admin must set back to “trust” the replication and repmgr lines on master EM and restart postgres.

11 Data Version Management

All SRE user data databases, created in the form of Data Models, are stored in two versions: A and B. This versioning system allows the operator to select the active version of the data in use for call processing or provision the data version not in service without affecting the service to subscribers. By default the A version is the Active version.

11.1 Version Selection

All data versioning is managed from the GUI in the *System -> Data Versioning* page.

In order to change the data version used for call processing, GUI or provisioning, the tab *Data Lock* must be used. Locking of data is required to be able to change the current active data version for call processing, GUI or provisioning.

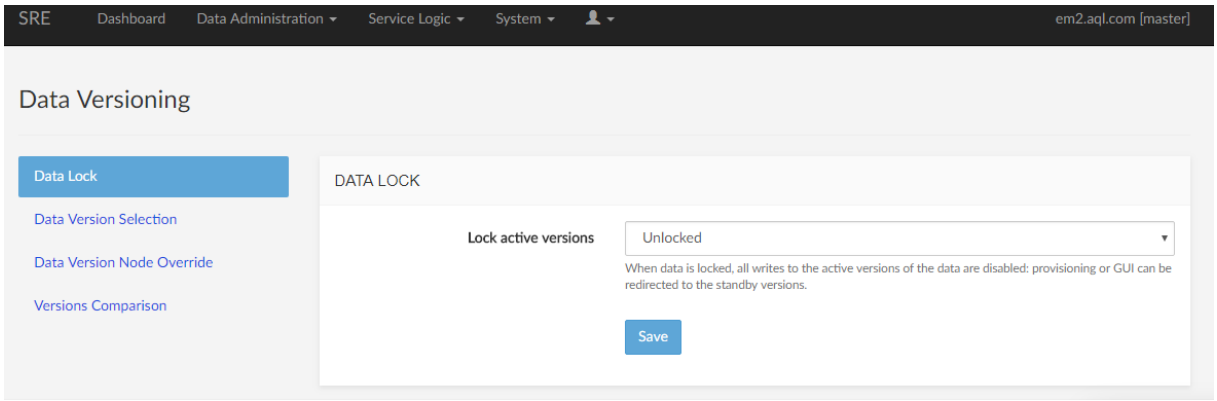


Figure 2: Graphical user interface, application, Teams Description automatically generated

If data is locked, the *Data Version Selection* page allows the operator to select the active data version, which is used for call processing. The GUI and provisioning can either be directed to the active version of data or to the standby version of data. This selection is performed by individual service.

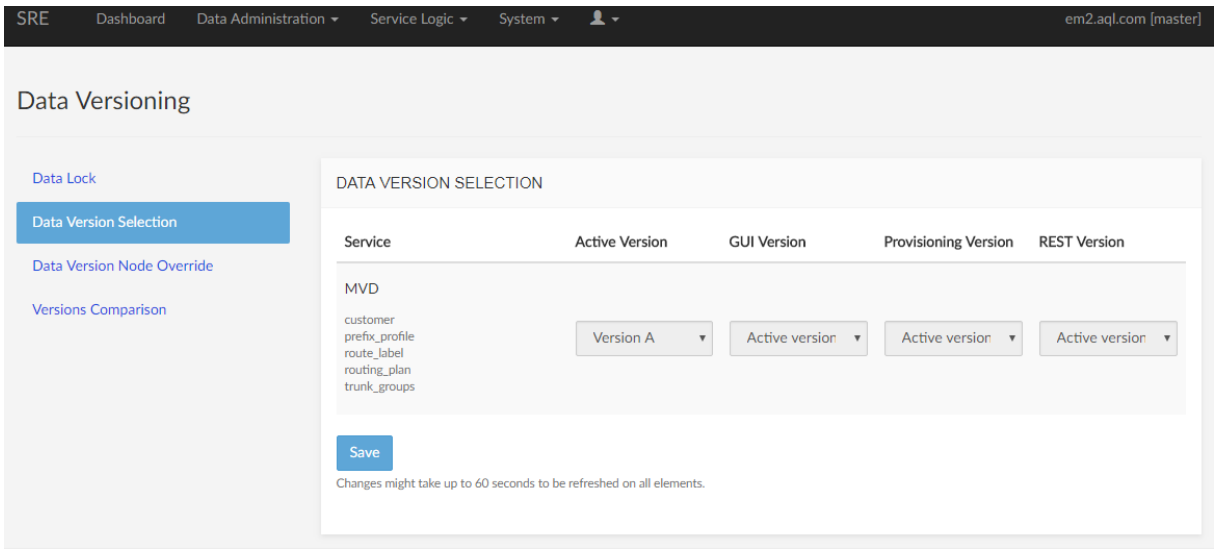


Figure 3: Graphical user interface, application, Teams Description automatically generated

The version selection logic (per service) is:

- If data is **unlocked**:
  - If active version is A:
    * Call processing uses version A of the data
    * If GUI version is Active:
      · GUI changes are directed to version A and data modification is allowed
    * If GUI version is Standby:
      · GUI changes are directed to version B and data modification is allowed
    * if provisioning version is Active:
      · batch provisioning is directed to version A and data modification is allowed
    * if provisioning version is Standby:
      · batch provisioning is directed to version B and data modification is allowed
    * if REST API version is Active
      · REST API to <DB>/active/<table> is directed to version A and data modification is allowed
    * if REST API version is Standby

        - REST API to <DB>/standby/<table> is directed to version B and data modification is allowed
    - If active version is B:
        * Call processing uses version B of the data
        * If GUI version is Active:
            - GUI changes are directed to version B and data modification is allowed
        * If GUI version is Standby:
            - GUI changes are directed to version A and data modification is allowed
        * if provisioning version is Active:
            - batch provisioning is directed to version B and data modification is allowed
        * if provisioning version is Standby:
            - batch provisioning is directed to version A and data modification is allowed
        * if REST API version is Active
            - REST API to <DB>/active/<table> is directed to version B and data modification is allowed
        * if REST API version is Standby
            - REST API to <DB>/standby/<table> is directed to version A and data modification is allowed

1. If data is **locked**:
    - If active version is A:
        - Call processing uses version A of the data
        - If GUI version is Active:
            * GUI changes are directed to version A and data modification is allowed
        - If GUI version is Standby:
            * GUI changes are directed to version B and data modification is allowed
        - if provisioning version is Active:
            * batch provisioning is **blocked**
        - if provisioning version is Standby:

        * batch provisioning is directed to version B and data modification is allowed
      - if REST API version is Active
        * REST API to <DB>/active/<table> directed to version A is **blocked**
      - if REST API version is Standby
        * REST API to <DB>/standby/<table> is directed to version B and data modification is allowed
  - If active version is B:
    - Call processing uses version B of the data
    - If GUI version is Active:
      * GUI changes are directed to version B and data modification is allowed
    - If GUI version is Standby:
      * GUI changes are directed to version A and data modification is allowed
    - if provisioning version is Active:
      * batch provisioning is **blocked**
    - if provisioning version is Standby:
      * batch provisioning is directed to version A and data modification is allowed
    - if REST API version is Active
      * REST API to <DB>/active/<table> directed to version B is blocked
    - if REST API version is Standby
      * REST API to <DB>/standby/<table> is directed to version A and data modification is allowed

The *Data Version Node Override* tab allows the operator to temporarily switch the version in use for a particular service on a particular call processor node to test it.

The version selection is presented as a matrix of CP nodes vs. service.

- The option *Default* instructs the CP node to use the global version selected under the tab *Data Version Selection*.
- The option *Override: version A* forces this CP node to use the version A of data, no matter the version selected under the tab *Data Version Selection*.

- The option *Override: version B* forces this CP node to use the version B of data, no matter the version selected under the tab *Data Version Selection*.

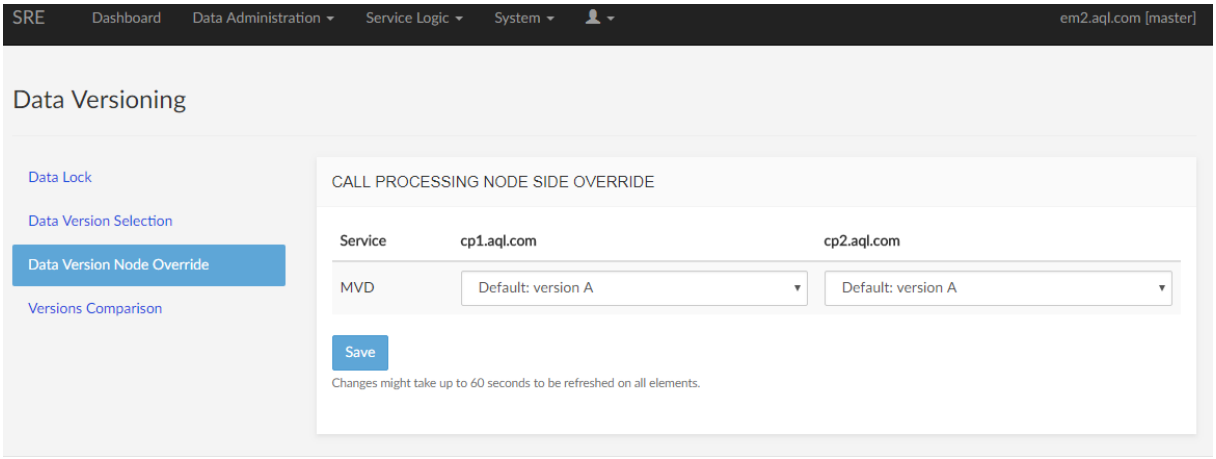


Figure 4: Graphical user interface, application Description automatically generated

Note

Once a CP node has been configured to use a version of the data different from what the other CP nodes use, the stats on the dashboard can be used to ensure that the CP node behaves correctly with this specific version of the data. Once this is confirmed, all the other CP nodes can be switched to the same version of the data under the tab *Data Version Selection*.

The tab *Versions Comparison* gives an overview of the records counts for the versions A and B, per service. The version highlighted in green is the version currently active for call processing.

11.2 Version Cloning

Under specific circumstances outside the normal maintenance operations, the operator might want to copy one version of a database on the other one. This can be achieved by using the pg_dump tool in the CLI to dump a particular service version and “pipe” it into psql connected to the other version of the database.

Warning

The option -a must be used to dump the data only. Without this option, the source database schema would be dumped too, with the tablespaces information specific to that version. Upon restoring on the destination database, this would cause the destination database tables to be

relocated on the same tablespaces as defined in the source database tables.

The tables in the destination database should be empty to allow the copy.

Warning

These commands operate at superuser level without any safeguards against human mistake such as mistyping or wrong versions. The operator must pay particular attention to the source version and destination version, to ensure that the destination version is not in use for call processing, GUI or provisioning. If this is the case, this could lead to catastrophic consequences as data is immediately replicated to all nodes.

On the master EM run:

```
[root@sre-em ~]# su - postgres
-bash-4.1$ pg_dump -a <service>_a\|psql <service>_b
```