



Operations and Monitoring Guide

SRE 4.0

Table of Contents

1	Process Management	3
2	Services management	5
2.1	SRE	5
2.2	PostgreSQL	5
2.3	InfluxDB	5
2.4	Kamailio	5
2.5	Mongo	6
3	Monitoring	6
3.1	Filesystems Monitoring	6
3.2	/var/log/sre cleaning	7
3.3	Memory and CPU usage monitoring	8
3.4	SRE Software Monitoring	9
3.4.1	SRE Process Monitoring	9
3.5	Stats Monitoring	10
3.5.1	InfluxDB query	16
3.6	PostgreSQL Monitoring	17
3.6.1	Service monitoring	17
3.6.2	Process Monitoring	18
3.6.3	Replication	19
3.6.4	WALS files	20
3.6.5	DB Disk Usage	21
3.6.6	Database automatic switchover monitoring	23
3.7	Kamailio Monitoring	24
3.7.1	Process Monitoring (on CP nodes only)	24
3.7.2	Kamailio Stats Monitoring	29
3.8	Mongo Monitoring	33
3.8.1	Service monitoring	33
3.8.2	Replica set status	33
3.9	Pacemaker Monitoring (if implemented)	38
3.9.1	Put a CP node in standby	39
3.9.2	Disabling a resource	39
3.9.3	Restarting a resource	40
3.9.4	Moving a resource	40
3.9.5	Cluster resources cleanup	40

3.9.6	Deleting a resource	40
3.9.7	Removing a node	41
3.9.8	Adding a node	41
3.9.9	Pcs backup and restore	41
3.9.10	Log files	41
4	Troubleshooting	41
4.1	SRE service issues	44
4.2	SRE Logs	45
5	Nodes Operational Status	46
5.1	Putting a CP in out of service	46
6	Backup & Restore Procedure	46
6.1	Choosing the Best Backup Strategy	47
6.2	Full Database Backup	47
6.2.1	Backup Procedure	47
6.2.2	Restore Procedure	48
6.3	Database Dumps	49
6.3.1	Backup Procedure	49
6.3.2	Restore Procedure	49
7	Full system backup and restore procedure	49
7.1	Creating a full backup file	50
7.2	Fully restoring a server	51
7.3	Node Re-Synchronisation	52
7.3.1	Backup Procedure	52
7.3.2	Restore Procedure (via DB Clone)	53
8	PostgreSQL Cluster Switchover	54
8.1	Automatic switchover	54
8.1.1	Disable automatic switchover	54
8.1.2	Enable automatic switchover	54
8.1.3	Doing manual switchover	54
8.2	Rebooting the master postgres node	55
8.3	Rebooting the standby postgres node	55
8.4	Failure of the master postgres node	55
8.4.1	With automatic switchover	55
8.4.2	Without automatic switchover	56

9 Node and Site isolation	58
10 Data Version Management	59
10.1 Version Selection	59
10.1.1 Version Cloning	64

1 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.

```
1 supervisor> help
2 default commands (type help <topic>):
3 =====
4 add exit open reload restart start tail
5 avail fg pid remove shutdown status update
6 clear maintail quit reread signal stop version
```

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

```
1 supervisor> status
2 sre-REST STOPPED Not started
3 sre-call-processor:0 RUNNING pid 6671, uptime 0:48:58
4 sre-gui STOPPED Not started
5 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.

Dashboard

Overview System Databases Broker SIP Agents Stats: Counters Stats: Performance Stats: SIP

EM EM1-APP

Last seen 2020-12-14 08:07:00
Version 3.0.2
Replication state master

Process	Status	Details
sre-REST	RUNNING	pid 9, uptime 39 days, 18:21:24
sre-agents-monitor	STOPPED	Not started
sre-broker	STOPPED	Not started
sre-call-processor:1	STOPPED	Not started
sre-call-processor:2	STOPPED	Not started
sre-call-processor:3	STOPPED	Not started
sre-call-processor:4	STOPPED	Not started
sre-gui	RUNNING	pid 8, uptime 39 days, 18:21:24
sre-health-monitor	RUNNING	pid 10, uptime 39 days, 18:21:24
sre-manager	RUNNING	pid 11, uptime 39 days, 18:21:24

EM EM2-APP

Last seen 2020-12-14 08:07:00
Version 3.0.2
Replication state standby

Process	Status	Details
sre-REST	RUNNING	pid 9, uptime 39 days, 18:03:27
sre-agents-monitor	STOPPED	Not started
sre-broker	STOPPED	Not started
sre-call-processor:1	STOPPED	Not started
sre-call-processor:2	STOPPED	Not started
sre-call-processor:3	STOPPED	Not started
sre-call-processor:4	STOPPED	Not started
sre-gui	RUNNING	pid 8, uptime 39 days, 18:03:27
sre-health-monitor	RUNNING	pid 10, uptime 39 days, 18:03:27
sre-manager	RUNNING	pid 11, uptime 39 days, 18:03:27

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.

```
1 supervisor> restart sre-manager
2 sre-manager: stopped
3 sre-manager: started
```

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

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

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

```
1 supervisor> start sre-manager
2 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.

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

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

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

2 Services management

2.1 SRE

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

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

2.2 PostgreSQL

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

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

2.3 InfluxDB

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

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

2.4 Kamailio

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

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

2.5 Mongo

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

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

3 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.

3.1 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, Kamilio 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

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

3.2 /var/log/sre cleaning

Very often, a full disk usage is related to the /var/log/sre directory (logs are filling in the fie system).

You can run the following command to clean that directory:

```
1 [root@sre-em1 ~]# cd /var/log/sre
2 [root@sre-em1 sre]# rm -f *.1
3 [root@sre-em1 sre]# rm -f *.2
4 [root@sre-em1 sre]# rm -f *.3
5 [root@sre-em1 sre]# rm -f *.4
6 [root@sre-em1 sre]# rm -f *.5
7 [root@sre-em1 sre]# rm -f *.6
8 [root@sre-em1 sre]# rm -f *.7
9 [root@sre-em1 sre]# rm -f *.8
10 [root@sre-em1 sre]# rm -f *.9
11 [root@sre-em1 sre]# rm -f *.10
```


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

3.3 Memory and CPU usage monitoring

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

```

1 Tasks: 247 total,  1 running, 246 sleeping,  0 stopped,  0 zombie
2 %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
3 KiB Mem : 3880144 total, 253148 free, 1468636 used, 2158360 buff/cache
4 KiB Swap: 1048572 total, 760308 free, 288264 used. 1818724 avail Mem
5
6  PID USER      PR  NI   VIRT   RES    SHR S  %CPU  %MEM     TIME+ COMMAND
7 26429 influxdb 20   0 1560028 239292 74696 S   80,0   6,2 20982:59 influxd
8 3297 sre       20   0 1646556 156956 6652 S   13,3   4,0 4363:27 sre-health-
   ↪ moni
9   9 root      20   0     0     0     0 S    6,7   0,0 673:47.40 rcu_sched
10 28612 root     20   0 162244   2368 1548 R    6,7   0,1  0:00.01 top
11   1 root     20   0 125640   2940 1628 S    0,0   0,1 53:17.18 systemd
12   2 root     20   0     0     0     0 S    0,0   0,0  0:04.12 kthreadd
13   4 root      0 -20     0     0     0 S    0,0   0,0  0:00.00 kworker/0:0
   ↪ H
14   6 root     20   0     0     0     0 S    0,0   0,0 25:55.80 ksoftirqd/0
15   7 root      rt   0     0     0     0 S    0,0   0,0  0:01.36 migration/0
16   8 root     20   0     0     0     0 S    0,0   0,0  0:00.00 rcu_bh
17  10 root      0 -20     0     0     0 S    0,0   0,0  0:00.00 lru-add-
   ↪ drain
18  11 root      rt   0     0     0     0 S    0,0   0,0  2:24.84 watchdog/0
19  12 root      rt   0     0     0     0 S    0,0   0,0  1:44.89 watchdog/1
20  13 root      rt   0     0     0     0 S    0,0   0,0  0:06.07 migration/1
21  14 root     20   0     0     0     0 S    0,0   0,0 13:13.88 ksoftirqd/1
22  16 root      0 -20     0     0     0 S    0,0   0,0  0:00.00 kworker/1:0
   ↪ H
23  18 root     20   0     0     0     0 S    0,0   0,0  0:00.00 kdevtmpfs
24  19 root      0 -20     0     0     0 S    0,0   0,0  0:00.00 netns
25  20 root     20   0     0     0     0 S    0,0   0,0  0:19.20 khungtaskd
26  21 root      0 -20     0     0     0 S    0,0   0,0  0:00.06 writeback
27  22 root      0 -20     0     0     0 S    0,0   0,0  0:00.00 kintegrityd
28  23 root      0 -20     0     0     0 S    0,0   0,0  0:00.00 bioset
29  24 root      0 -20     0     0     0 S    0,0   0,0  0:00.00 bioset
30  25 root      0 -20     0     0     0 S    0,0   0,0  0:00.00 bioset
31  26 root      0 -20     0     0     0 S    0,0   0,0  0:00.00 kblockd
32  27 root      0 -20     0     0     0 S    0,0   0,0  0:00.00 md
33  28 root      0 -20     0     0     0 S    0,0   0,0  0:00.00 edac-poller
34  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 ksm
  
```

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_rdadcd
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

3.4 SRE Software Monitoring

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

3.4.1 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
2 sre-REST                RUNNING    pid 3294, uptime 46 days, 17:00:24
3 sre-agents-monitor      STOPPED   Not started
  
```

4	sre-broker	STOPPED	Not started
5	sre-call-processor:0	STOPPED	Not started
6	sre-cdr-collector	RUNNING	pid 3295, uptime 46 days, 17:00:24
7	sre-cdr-postprocessor	STOPPED	Not started
8	sre-cdr-sender	STOPPED	Not started
9	sre-dns-updater	STOPPED	Not started
10	sre-enum-processor	STOPPED	Not started
11	sre-gui	RUNNING	pid 3296, uptime 46 days, 17:00:24
12	sre-health-monitor	RUNNING	pid 3297, uptime 46 days, 17:00:24
13	sre-http-processor	RUNNING	pid 3298, uptime 46 days, 17:00:24
14	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:
 - sre-agents-monitor
 - sre-broker
 - sre-call-processor:[0-N] (the number of processes might be different depending on the supervisord configuration file)
 - sre-health-monitor
 - sre-cdr-sender

3.5 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
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
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
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
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
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
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 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
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
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
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
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
    ↪ :32:01,1652193121,3800.000,3800.000,3800.000,3700.000,3566.667,3833.333,3833.333,3866.667
    ↪
3 sre32-cp2-testbed,profiling.cp.CAC test.503,2022-05-10T16
    ↪ :32:01,1652193121,,,,,,,,,,,,,
4 sre32-cp2-testbed,profiling.cp.CAC test.Extract Contacts,2022-05-10T16
    ↪ :32:01,1652193121,0.113,0.121,0.103,0.115,0.123,0.114,0.111,0.117,0.121,0.110,0.115,0.113
    ↪
  
```

```

5 sre32-cp2-testbed,profiling.cp.CAC test.Extract mContacts,2022-05-10T16
  ↳ :32:01,1652193121,,,,,,,,,,,,,
6 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
  ↳
7 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
  ↳
8 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
  ↳
9 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
  ↳
10 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
  ↳
11 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
  ↳
12 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
  ↳
13 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
  ↳
14 sre32-cp2-testbed,profiling.cp.INVITE,2022-05-10T16
  ↳ :32:01,1652193121,13.214,10.280,13.943,13.042,19.519,13.832,19.339,10.739,15.154,16.71
  ↳
15 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
  ↳
16 sre32-cp1-testbed,system.cpu,2022-05-10T16
  ↳ :32:01,1652193121,11718.333,11660.000,11665.000,11911.667,12000.000,11545.000,11658.33
  ↳
17 sre32-cp2-testbed,system.cpu,2022-05-10T16
  ↳ :32:01,1652193121,11463.333,11186.667,11150.000,11330.000,11451.667,11265.000,11330.00
  ↳
18 sre32-em1-testbed,system.cpu,2022-05-10T16
  ↳ :32:01,1652193121,10006.122,9622.449,10550.000,9261.224,10508.163,9285.417,9935.417,95
  ↳
19 sre32-em2-testbed,system.cpu,2022-05-10T16
  ↳ :32:01,1652193121,10224.074,10170.909,10077.778,10127.778,10260.000,10033.333,10049.09
  ↳
20 sre32-cp1-testbed,system.disk./,2022-05-10T16
  ↳ :32:01,1652193121,67900.000,67900.000,67900.000,67900.000,67900.000,67900.000,67900.00

```

```
↪
21 sre32-cp2-testbed,system.disk./,2022-05-10T16
↪ :32:01,1652193121,69600.000,69600.000,69600.000,69600.000,69600.000,69600.000,69600.000,69600.000
↪
22 sre32-em1-testbed,system.disk./,2022-05-10T16
↪ :32:01,1652193121,76400.000,76400.000,76400.000,76400.000,76400.000,76400.000,76400.000,76400.000
↪
23 sre32-em2-testbed,system.disk./,2022-05-10T16
↪ :32:01,1652193121,67500.000,67500.000,67500.000,67500.000,67500.000,67500.000,67500.000,67500.000
↪
24 sre32-cp1-testbed,system.disk./boot,2022-05-10T16
↪ :32:01,1652193121,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000
↪
25 sre32-cp2-testbed,system.disk./boot,2022-05-10T16
↪ :32:01,1652193121,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000
↪
26 sre32-em1-testbed,system.disk./boot,2022-05-10T16
↪ :32:01,1652193121,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000
↪
27 sre32-em2-testbed,system.disk./boot,2022-05-10T16
↪ :32:01,1652193121,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000,66700.000
↪
28 sre32-cp1-testbed,system.disk./boot/efi,2022-05-10T16
↪ :32:01,1652193121,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000
↪
29 sre32-cp2-testbed,system.disk./boot/efi,2022-05-10T16
↪ :32:01,1652193121,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000
↪
30 sre32-em1-testbed,system.disk./boot/efi,2022-05-10T16
↪ :32:01,1652193121,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000
↪
31 sre32-em2-testbed,system.disk./boot/efi,2022-05-10T16
↪ :32:01,1652193121,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000
↪
32 sre32-cp1-testbed,system.disk./data/sre/db/backups,2022-05-10T16
↪ :32:01,1652193121,17100.000,17100.000,17100.000,17100.000,17100.000,17100.000,17100.000,17100.000
↪
33 sre32-cp2-testbed,system.disk./data/sre/db/backups,2022-05-10T16
↪ :32:01,1652193121,17100.000,17100.000,17100.000,17100.000,17100.000,17100.000,17100.000,17100.000
↪
34 sre32-em1-testbed,system.disk./data/sre/db/backups,2022-05-10T16
↪ :32:01,1652193121,10400.000,10400.000,10400.000,10400.000,10400.000,10400.000,10400.000,10400.000
↪
35 sre32-em2-testbed,system.disk./data/sre/db/backups,2022-05-10T16
↪ :32:01,1652193121,17300.000,17300.000,17300.000,17300.000,17300.000,17300.000,17300.000,17300.000
↪
```

```
36 sre32-cp1-testbed,system.disk./data/sre/db/wals,2022-05-10T16
    ↪ :32:01,1652193121,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000
    ↪
37 sre32-cp2-testbed,system.disk./data/sre/db/wals,2022-05-10T16
    ↪ :32:01,1652193121,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000
    ↪
38 sre32-em1-testbed,system.disk./data/sre/db/wals,2022-05-10T16
    ↪ :32:01,1652193121,1900.000,1900.000,1900.000,1900.000,1900.000,1900.000,1900.000,1900.000,1900.000,1900.000
    ↪
39 sre32-em2-testbed,system.disk./data/sre/db/wals,2022-05-10T16
    ↪ :32:01,1652193121,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000,200.000
    ↪
40 sre32-cp1-testbed,system.disk./data/sre/provisioning,2022-05-10T16
    ↪ :32:01,1652193121,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000
    ↪
41 sre32-cp2-testbed,system.disk./data/sre/provisioning,2022-05-10T16
    ↪ :32:01,1652193121,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000
    ↪
42 sre32-em1-testbed,system.disk./data/sre/provisioning,2022-05-10T16
    ↪ :32:01,1652193121,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000
    ↪
43 sre32-em2-testbed,system.disk./data/sre/provisioning,2022-05-10T16
    ↪ :32:01,1652193121,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000,400.000
    ↪
44 sre32-cp1-testbed,system.disk./opt,2022-05-10T16
    ↪ :32:01,1652193121,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000
    ↪
45 sre32-cp2-testbed,system.disk./opt,2022-05-10T16
    ↪ :32:01,1652193121,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000,35300.000
    ↪
46 sre32-em1-testbed,system.disk./opt,2022-05-10T16
    ↪ :32:01,1652193121,35000.000,35000.000,35000.000,35000.000,35000.000,35000.000,35000.000,35000.000,35000.000,35000.000
    ↪
47 sre32-em2-testbed,system.disk./opt,2022-05-10T16
    ↪ :32:01,1652193121,34900.000,34900.000,34900.000,34900.000,34900.000,34900.000,34900.000,34900.000,34900.000,34900.000
    ↪
48 sre32-cp1-testbed,system.disk./var/log,2022-05-10T16
    ↪ :32:01,1652193121,44426.667,44500.000,44500.000,44496.667,44400.000,44400.000,44400.000,44400.000,44400.000,44400.000
    ↪
49 sre32-cp2-testbed,system.disk./var/log,2022-05-10T16
    ↪ :32:01,1652193121,39906.667,39916.667,39916.667,39900.000,39910.000,39900.000,39905.000,39905.000,39905.000,39905.000
    ↪
50 sre32-em1-testbed,system.disk./var/log,2022-05-10T16
    ↪ :32:01,1652193121,40967.347,40900.000,40900.000,40900.000,40900.000,40900.000,40831.250,40800.000,40800.000,40800.000
    ↪
51 sre32-em2-testbed,system.disk./var/log,2022-05-10T16
```



```

↪ :32:01,1652193121,62600.000,62600.000,62600.000,62600.000,62600.000,62600.000,62600.000,62600.000
↪
52 sre32-cp1-testbed,system.mem.ram,2022-05-10T16
↪ :32:01,1652193121,51891.667,52018.333,52100.000,52100.000,52098.333,52100.000,52100.000,52100.000
↪
53 sre32-cp2-testbed,system.mem.ram,2022-05-10T16
↪ :32:01,1652193121,46798.333,46798.333,46733.333,46595.000,46593.333,46580.000,46588.333,46588.333
↪
54 sre32-em1-testbed,system.mem.ram,2022-05-10T16
↪ :32:01,1652193121,59100.000,59100.000,59100.000,59100.000,59097.959,59100.000,59039.583,59039.583
↪
55 sre32-em2-testbed,system.mem.ram,2022-05-10T16
↪ :32:01,1652193121,34618.519,34601.818,34609.259,34605.556,34605.455,34611.111,34607.273,34607.273
↪
56 sre32-cp1-testbed,system.mem.swap,2022-05-10T16
↪ :32:01,1652193121,3200.000,3200.000,3200.000,3200.000,3200.000,3200.000,3200.000,3200.000
↪
57 sre32-cp2-testbed,system.mem.swap,2022-05-10T16
↪ :32:01,1652193121,2500.000,2500.000,2500.000,2500.000,2500.000,2500.000,2500.000,2500.000
↪
58 sre32-em1-testbed,system.mem.swap,2022-05-10T16
↪ :32:01,1652193121,5600.000,5600.000,5600.000,5600.000,5600.000,5600.000,5600.000,5600.000
↪
59 sre32-em2-testbed,system.mem.swap,2022-05-10T16
↪ :32:01,1652193121,1200.000,1200.000,1200.000,1200.000,1200.000,1200.000,1200.000,1200.000
↪
    
```

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

3.5.1 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:

```

1 [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           occurrences   profiling.cp.INVITE           SRE-33-CP1
  ↳ 2023-10-18T12:20:50.000000000Z           2
  
```

3.6 PostgreSQL Monitoring

3.6.1 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
  
```

3.6.2 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
5 postgres 10225 2436 0 set01 ?      00:00:00 postgres: sre stirshaken_b
   ↪ 127.0.0.1(37434) idle
6 postgres 10226 2436 0 set01 ?      00:00:00 postgres: sre
   ↪ regression_test_08_12_22_a 127.0.0.1(37436) idle
7 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.

```

1 [root@sre-cp ~]# ps -ef|grep "postgres: sre"|grep -c "in transaction"
2 1
  
```

Idle connections are the other ones.

```

1 [root@sre-cp ~]# ps -ef|grep "postgres: sre"|grep -c idle
2 59
  
```

3.6.3 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:

```

1 [root@sre-em ~]# /usr/pgsql-14/bin/psql -U postgres -h 127.0.0.1 -c "select *
   ↪ from pg_is_in_recovery()"
2 pg_is_in_recovery
3 -----
4 f
5 (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:

```
1 [root@sre.em ~]# /usr/pgsql-14/bin/psql -U postgres -h 127.0.0.1 -c "select *
   ↳ from pg_stat_replication"
2 pid | usesysid | username | application_name | client_addr | client_hostname
   ↳ | client_port | backend_start | backend_xmin | state
   ↳ | sent_lsn | write_lsn | flush_lsn | replay_lsn
3 | write_lag | flush_lag | replay_lag | sync_priority |
   ↳ sync_state | reply_time
4 -----+-----+-----+-----+-----+-----+-----
   ↳
5 +-----+-----+-----+-----+-----+-----+-----
   ↳
6 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
7 | 00:00:00.001141 | 00:00:00.002427 | 00:00:00.002952 | 0 | async
   ↳ | 2023-10-18 11:01:45.525076+02
8 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
9 | 00:00:00.001203 | 00:00:00.002264 | 00:00:00.002315 | 0 | async
   ↳ | 2023-10-18 11:01:45.524272+02
10 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
11 | 00:00:00.001277 | 00:00:00.002492 | 00:00:00.002493 | 0 | async
   ↳ | 2023-10-18 11:01:45.524472+02
12 (3 rows)
```

3.6.4 WAL files

WALS 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 direction `/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 file being updated. This can be seen using the following command:

```
1 [root@SRE-33-EM1 ~]# ls -ltr /var/lib/pgsql/14/data/pg_wal/
2 totale 573468
3 -rw----- 1 postgres postgres 343 20 gen 2023 00000009.history
4 -rw----- 1 postgres postgres 386 20 gen 2023 0000000A.history
5 -rw----- 1 postgres postgres 430 20 gen 2023 0000000B.history
```

```

6 -rw----- 1 postgres postgres      474 20 gen  2023 00000000C.history
7 -rw----- 1 postgres postgres      518 20 gen  2023 00000000D.history
8 -rw----- 1 postgres postgres 16777216 20 ago  03.17 00000000D000000020000000B9
9 -rw----- 1 postgres postgres 16777216 22 ago  22.56 00000000D000000020000000BA
10 -rw----- 1 postgres postgres 16777216 25 ago  23.36 00000000D000000020000000BB
11 -rw----- 1 postgres postgres 16777216 27 ago  03.15 00000000D00000002000000099
12 -rw----- 1 postgres postgres 16777216 27 ago  03.16 00000000D0000000200000009A
13 -rw----- 1 postgres postgres 16777216 30 ago  10.41 00000000D0000000200000009B
14 -rw----- 1 postgres postgres 16777216  2 set  16.11 00000000D0000000200000009C
15 -rw----- 1 postgres postgres 16777216  3 set  03.48 00000000D0000000200000009D
16 -rw----- 1 postgres postgres 16777216  3 set  03.49 00000000D0000000200000009E
17 -rw----- 1 postgres postgres 16777216  6 set  16.08 00000000D0000000200000009F
18 -rw----- 1 postgres postgres 16777216 10 set  02.20 00000000D000000020000000A0
19 -rw----- 1 postgres postgres 16777216 10 set  03.15 00000000D000000020000000A1
20 -rw----- 1 postgres postgres 16777216 10 set  03.17 00000000D000000020000000A2
21 -rw----- 1 postgres postgres 16777216 13 set  03.06 00000000D000000020000000A3
22 -rw----- 1 postgres postgres 16777216 15 set  08.31 00000000D000000020000000A4
23 -rw----- 1 postgres postgres 16777216 17 set  03.15 00000000D000000020000000A5
24 -rw----- 1 postgres postgres 16777216 17 set  03.17 00000000D000000020000000A6
25 -rw----- 1 postgres postgres 16777216 20 set  04.21 00000000D000000020000000A7
26 -rw----- 1 postgres postgres 16777216 23 set  02.08 00000000D000000020000000A8
27 -rw----- 1 postgres postgres 16777216 24 set  03.15 00000000D000000020000000A9
28 -rw----- 1 postgres postgres 16777216 24 set  03.17 00000000D000000020000000AA
29 -rw----- 1 postgres postgres 16777216 27 set  14.57 00000000D000000020000000AB
30 -rw----- 1 postgres postgres 16777216 30 set  00.36 00000000D000000020000000AC
31 -rw----- 1 postgres postgres 16777216  1 ott  03.15 00000000D000000020000000AD
32 -rw----- 1 postgres postgres 16777216  1 ott  03.17 00000000D000000020000000AE
33 -rw----- 1 postgres postgres 16777216  3 ott  20.01 00000000D000000020000000AF
34 -rw----- 1 postgres postgres 16777216  6 ott  17.22 00000000D000000020000000B0
35 -rw----- 1 postgres postgres 16777216  8 ott  03.15 00000000D000000020000000B1
36 -rw----- 1 postgres postgres 16777216  8 ott  03.18 00000000D000000020000000B2
37 -rw----- 1 postgres postgres 16777216 10 ott  16.41 00000000D000000020000000B3
38 -rw----- 1 postgres postgres 16777216 13 ott  04.37 00000000D000000020000000B4
39 -rw----- 1 postgres postgres 16777216 15 ott  03.15 00000000D000000020000000B5
40 -rw----- 1 postgres postgres 16777216 15 ott  03.17 00000000D000000020000000B6
41 -rw----- 1 postgres postgres      345 15 ott  03.17 00000000D000000020000000B6
    ↔ .00000028.backup
42 -rw----- 1 postgres postgres 16777216 17 ott  23.44 00000000D000000020000000B7
43 drwx----- . 2 postgres postgres      4096 17 ott  23.46 archive_status
44 -rw----- 1 postgres postgres 16777216 18 ott  11.01 00000000D000000020000000B8
    
```

3.6.5 DB Disk Usage

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

```

1 [root@localhost ~]# /usr/pgsql-14/bin/psql -U postgres -h 127.0.0.1 -c "select
  ↵ datname, pg_database_size(datname) from pg_database"
2           datname                | pg_database_size
3 -----+-----
4 postgres                        |          8979235
5 template1                       |          8823299
6 template0                       |          8823299
7 repmgr                           |          9257763
8 sre                              |         48988963
9 temp_test_delete_me_a           |          9151267
10 temp_test_delete_me_b          |          9118499
11 temp_test_delete_me_please_a   |          9102115
12 temp_test_delete_me_please_b   |          9077539
13 test_default_value_a           |          9044771
14 test_default_value_b           |          9044771
15 test_a                          |          9044771
16 test_b                          |          9044771
17 test2_a                         |          9044771
18 test2_b                         |          9044771
19 sss_a                           |          9093923
20 sss_b                           |          9044771
21 stirshaken_a                   |          9167651
22 stirshaken_b                   |          9044771
23 test_versioning_a              |          9069347
24 test_versioning_b              |          9044771
25 test_versioning_2_a            |          9069347
26 test_versioning_2_b            |          9069347
27 regression_test_08_12_22_a     |          9224995
28 regression_test_08_12_22_b     |          9143075
29 m247_lab_a                      |          9585443
30 m247_lab_b                      |          9044771
31 fuse2_voice_a                  |          9044771
32 fuse2_voice_b                  |          9044771
33 dm_validation_1_a              |          9298723
34 dm_validation_1_b              |          9044771
35 demo_a                         |          9110307
36 demo_b                         |          9044771
37 demo_doc_versioning_position_a |          9093923
38 demo_doc_versioning_position_b |          9044771
39 demo_doc_export_dm_diagram_a   |          9044771
40 demo_doc_export_dm_diagram_b   |          9069347
41 inventory_a                    |          9216803
42 inventory_b                    |          9044771
43 ...
44
45 (61 rows)

```

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

```

1 [root@localhost ~]# /usr/pgsql-14/bin/psql -U postgres -h 127.0.0.1 -c "select
  ↳ spcname, pg_tablespace_size(spcname) from pg_tablespace"
2 spcname | pg_tablespace_size
3 -----+-----
4 pg_default | 594550039
5 pg_global | 622368
6 (2 rows)
  
```

3.6.6 Database automatic switchover monitoring

SSH access is essential for performing a manual or automated cluster switchover. To check that all nodes are reachable via ssh in both directions run the following command as user postgres:

```

1 -bash-4.2$ /usr/pgsql-14/bin/repmgr cluster crosscheck
2 Name | ID | 1 | 2 | 3 | 4
3 -----+-----+-----+-----+-----+-----
4 sre-em1 | 1 | * | * | * | *
5 sre-em2 | 2 | * | * | * | *
6 sre-cp1 | 3 | * | * | * | *
7 sre-cp2 | 4 | * | * | * | *
  
```

All cells should contain a ***** meaning that a succesful connection is working between the servers.

To show the status of *repmgrd* daemons and if the automatic switchover is disabled (paused) run the following command as user postgres:

```

1 -bash-4.2$ /usr/pgsql-14/bin/repmgr service status
2 ID | Name | Role | Status | Upstream | repmgrd | PID | Paused
  ↳ ? | Upstream last seen
3 -----+-----+-----+-----+-----+-----+-----+-----
  ↳
4 1 | sre-em1 | primary | * running | | running | 107904 | no
  ↳ | n/a
5 2 | sre-em2 | standby | running | sre-em1 | running | 1343 | no
  ↳ | 0 second(s) ago
6 3 | sre-cp1 | standby | running | sre-em1 | running | 6087 | no
  ↳ | 1 second(s) ago
7 4 | sre-cp2 | standby | running | sre-em1 | running | 3826938 | no
  ↳ | 1 second(s) ago
  
```

The identical information is displayed within the dashboard’s Databases tab in the GUI.

SRE Dashboard Data Administration Datamodel Service Logic System Super Administrator 4.0 Testbed - sre-40-em1 [master] 4.0.alpha

Dashboard

Service Overview System Overview **Databases** Broker SIP Agents Stats: Counters Stats: Performance Stats: SIP

SRE-40-EM1

PostgreSQL Status

Replication state	master
Repmgr state	running
Automatic switchover	enabled
Current location	1/4D1507E8

Database	Size
postgres	8.8 MB

If repmgrd daemon is not currently running on a node, establish a connection to that node and execute the command:

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

3.7 Kamailio Monitoring

3.7.1 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:

```
1 [root@sre-cp ~]# ps -ef|grep kamailio|grep -v grep
2 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
3 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
4 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
5 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
6 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
7 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
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
9 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
```

```
10 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
11 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
12 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
13 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
14 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
15 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
16 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
17 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
18 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
19 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
20 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
21 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
22 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
23 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
24 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
25 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
26 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
27 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
28 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
29 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
30 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
31 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
32 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
```

```

33 kamilio  8036  7992  0 lug20 ?          00:03:42 /usr/sbin/kamilio -DD -P /run/
    ↪ kamilio/kamilio.pid -f /etc/kamilio/kamilio.cfg -m 64 -M 8
34 kamilio  8037  7992  0 lug20 ?          00:02:14 /usr/sbin/kamilio -DD -P /run/
    ↪ kamilio/kamilio.pid -f /etc/kamilio/kamilio.cfg -m 64 -M 8
  
```

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

```

1 [root@sre-cp ~]# kamctl ps
2 {
3   "jsonrpc": "2.0",
4   "result": [
5     {
6       "IDX": 0,
7       "PID": 7992,
8       "DSC": "main process - attendant"
9     }, {
10      "IDX": 1,
11      "PID": 8006,
12      "DSC": "udp receiver child=0 sock=127.0.0.1:5060"
13    }, {
14      "IDX": 2,
15      "PID": 8007,
16      "DSC": "udp receiver child=1 sock=127.0.0.1:5060"
17    }, {
18      "IDX": 3,
19      "PID": 8008,
20      "DSC": "udp receiver child=2 sock=127.0.0.1:5060"
21    }, {
22      "IDX": 4,
23      "PID": 8009,
24      "DSC": "udp receiver child=3 sock=127.0.0.1:5060"
25    }, {
26      "IDX": 5,
27      "PID": 8010,
28      "DSC": "udp receiver child=4 sock=127.0.0.1:5060"
29    }, {
30      "IDX": 6,
31      "PID": 8011,
32      "DSC": "udp receiver child=5 sock=127.0.0.1:5060"
33    }, {
34      "IDX": 7,
35      "PID": 8012,
36      "DSC": "udp receiver child=6 sock=127.0.0.1:5060"
37    }, {
38      "IDX": 8,
39      "PID": 8013,
40      "DSC": "udp receiver child=7 sock=127.0.0.1:5060"
  
```

```
41     }, {
42         "IDX": 9,
43         "PID": 8014,
44         "DSC": "udp receiver child=0 sock=10.0.161.182:5060"
45     }, {
46         "IDX": 10,
47         "PID": 8015,
48         "DSC": "udp receiver child=1 sock=10.0.161.182:5060"
49     }, {
50         "IDX": 11,
51         "PID": 8016,
52         "DSC": "udp receiver child=2 sock=10.0.161.182:5060"
53     }, {
54         "IDX": 12,
55         "PID": 8017,
56         "DSC": "udp receiver child=3 sock=10.0.161.182:5060"
57     }, {
58         "IDX": 13,
59         "PID": 8018,
60         "DSC": "udp receiver child=4 sock=10.0.161.182:5060"
61     }, {
62         "IDX": 14,
63         "PID": 8019,
64         "DSC": "udp receiver child=5 sock=10.0.161.182:5060"
65     }, {
66         "IDX": 15,
67         "PID": 8020,
68         "DSC": "udp receiver child=6 sock=10.0.161.182:5060"
69     }, {
70         "IDX": 16,
71         "PID": 8021,
72         "DSC": "udp receiver child=7 sock=10.0.161.182:5060"
73     }, {
74         "IDX": 17,
75         "PID": 8022,
76         "DSC": "slow timer"
77     }, {
78         "IDX": 18,
79         "PID": 8023,
80         "DSC": "timer"
81     }, {
82         "IDX": 19,
83         "PID": 8024,
84         "DSC": "secondary timer"
85     }, {
86         "IDX": 20,
```

```
87     "PID": 8025,  
88     "DSC": "JSONRPCS FIFO"  
89   }, {  
90     "IDX": 21,  
91     "PID": 8026,  
92     "DSC": "JSONRPCS DATAGRAM"  
93   }, {  
94     "IDX": 22,  
95     "PID": 8027,  
96     "DSC": "ctl handler"  
97   }, {  
98     "IDX": 23,  
99     "PID": 8028,  
100    "DSC": "Dialog Clean Timer"  
101  }, {  
102    "IDX": 24,  
103    "PID": 8029,  
104    "DSC": "tcp receiver (generic) child=0"  
105  }, {  
106    "IDX": 25,  
107    "PID": 8030,  
108    "DSC": "tcp receiver (generic) child=1"  
109  }, {  
110    "IDX": 26,  
111    "PID": 8031,  
112    "DSC": "tcp receiver (generic) child=2"  
113  }, {  
114    "IDX": 27,  
115    "PID": 8032,  
116    "DSC": "tcp receiver (generic) child=3"  
117  }, {  
118    "IDX": 28,  
119    "PID": 8033,  
120    "DSC": "tcp receiver (generic) child=4"  
121  }, {  
122    "IDX": 29,  
123    "PID": 8034,  
124    "DSC": "tcp receiver (generic) child=5"  
125  }, {  
126    "IDX": 30,  
127    "PID": 8035,  
128    "DSC": "tcp receiver (generic) child=6"  
129  }, {  
130    "IDX": 31,  
131    "PID": 8036,  
132    "DSC": "tcp receiver (generic) child=7"
```

```
133     }, {
134         "IDX": 32,
135         "PID": 8037,
136         "DSC": "tcp main process"
137     }
138 ],
139 "id": 7588
140 }
```

3.7.2 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.

```
1 [root@sre-cp ~]# kamctl stats
2 {
3   "jsonrpc": "2.0",
4   "result": [
5     "app_python3:active_dialogs = 0",
6     "app_python3:early_dialogs = 0",
7     "app_python3:expired_dialogs = 118",
8     "app_python3:failed_dialogs = 6",
9     "app_python3:processed_dialogs = 76014",
10    "core:bad_URIs_rcvd = 0",
11    "core:bad_msg_hdr = 0",
12    "core:drop_replies = 0",
13    "core:drop_requests = 3",
14    "core:err_replies = 0",
15    "core:err_requests = 0",
16    "core:fwd_replies = 132759",
17    "core:fwd_requests = 1711587",
18    "core:rcv_replies = 419914",
19    "core:rcv_replies_18x = 65248",
20    "core:rcv_replies_1xx = 108873",
21    "core:rcv_replies_1xx_bye = 0",
22    "core:rcv_replies_1xx_cancel = 0",
23    "core:rcv_replies_1xx_invite = 108873",
24    "core:rcv_replies_1xx_message = 0",
25    "core:rcv_replies_1xx_prack = 0",
26    "core:rcv_replies_1xx_refer = 0",
27    "core:rcv_replies_1xx_reg = 0",
28    "core:rcv_replies_1xx_update = 0",
29    "core:rcv_replies_2xx = 310992",
30    "core:rcv_replies_2xx_bye = 106582",
```

```
31 "core:rcv_replies_2xx_cancel = 0",
32 "core:rcv_replies_2xx_invite = 83786",
33 "core:rcv_replies_2xx_message = 0",
34 "core:rcv_replies_2xx_prack = 0",
35 "core:rcv_replies_2xx_refer = 0",
36 "core:rcv_replies_2xx_reg = 0",
37 "core:rcv_replies_2xx_update = 0",
38 "core:rcv_replies_3xx = 0",
39 "core:rcv_replies_3xx_bye = 0",
40 "core:rcv_replies_3xx_cancel = 0",
41 "core:rcv_replies_3xx_invite = 0",
42 "core:rcv_replies_3xx_message = 0",
43 "core:rcv_replies_3xx_prack = 0",
44 "core:rcv_replies_3xx_refer = 0",
45 "core:rcv_replies_3xx_reg = 0",
46 "core:rcv_replies_3xx_update = 0",
47 "core:rcv_replies_401 = 0",
48 "core:rcv_replies_404 = 0",
49 "core:rcv_replies_407 = 0",
50 "core:rcv_replies_480 = 0",
51 "core:rcv_replies_486 = 0",
52 "core:rcv_replies_4xx = 48",
53 "core:rcv_replies_4xx_bye = 48",
54 "core:rcv_replies_4xx_cancel = 0",
55 "core:rcv_replies_4xx_invite = 0",
56 "core:rcv_replies_4xx_message = 0",
57 "core:rcv_replies_4xx_prack = 0",
58 "core:rcv_replies_4xx_refer = 0",
59 "core:rcv_replies_4xx_reg = 0",
60 "core:rcv_replies_4xx_update = 0",
61 "core:rcv_replies_5xx = 1",
62 "core:rcv_replies_5xx_bye = 0",
63 "core:rcv_replies_5xx_cancel = 0",
64 "core:rcv_replies_5xx_invite = 0",
65 "core:rcv_replies_5xx_message = 0",
66 "core:rcv_replies_5xx_prack = 0",
67 "core:rcv_replies_5xx_refer = 0",
68 "core:rcv_replies_5xx_reg = 0",
69 "core:rcv_replies_5xx_update = 0",
70 "core:rcv_replies_6xx = 0",
71 "core:rcv_replies_6xx_bye = 0",
72 "core:rcv_replies_6xx_cancel = 0",
73 "core:rcv_replies_6xx_invite = 0",
74 "core:rcv_replies_6xx_message = 0",
75 "core:rcv_replies_6xx_prack = 0",
76 "core:rcv_replies_6xx_refer = 0",
```

```
77 "core:rcv_replies_6xx_reg = 0",
78 "core:rcv_replies_6xx_update = 0",
79 "core:rcv_requests = 2039960",
80 "core:rcv_requests_ack = 83991",
81 "core:rcv_requests_bye = 112644",
82 "core:rcv_requests_cancel = 5",
83 "core:rcv_requests_info = 0",
84 "core:rcv_requests_invite = 76022",
85 "core:rcv_requests_message = 0",
86 "core:rcv_requests_notify = 0",
87 "core:rcv_requests_options = 1767298",
88 "core:rcv_requests_prack = 0",
89 "core:rcv_requests_publish = 0",
90 "core:rcv_requests_refer = 0",
91 "core:rcv_requests_register = 0",
92 "core:rcv_requests_subscribe = 0",
93 "core:rcv_requests_update = 0",
94 "core:unsupported_methods = 0",
95 "dns:failed_dns_request = 0",
96 "dns:slow_dns_request = 0",
97 "registrar:accepted_regs = 0",
98 "registrar:default_expire = 3600",
99 "registrar:default_expires_range = 0",
100 "registrar:expires_range = 0",
101 "registrar:max_contacts = 1",
102 "registrar:max_expires = 3600",
103 "registrar:rejected_regs = 0",
104 "shmem:fragments = 6",
105 "shmem:free_size = 64144248",
106 "shmem:max_used_size = 7896000",
107 "shmem:real_used_size = 2964616",
108 "shmem:total_size = 67108864",
109 "shmem:used_size = 2718232",
110 "sl:1xx_replies = 0",
111 "sl:200_replies = 0",
112 "sl:202_replies = 0",
113 "sl:2xx_replies = 0",
114 "sl:300_replies = 0",
115 "sl:301_replies = 0",
116 "sl:302_replies = 0",
117 "sl:3xx_replies = 0",
118 "sl:400_replies = 0",
119 "sl:401_replies = 0",
120 "sl:403_replies = 0",
121 "sl:404_replies = 0",
122 "sl:407_replies = 0",
```



```
123     "sl:408_replies = 0",
124     "sl:483_replies = 0",
125     "sl:4xx_replies = 0",
126     "sl:500_replies = 0",
127     "sl:5xx_replies = 4",
128     "sl:6xx_replies = 0",
129     "sl:failures = 0",
130     "sl:received_ACKs = 3",
131     "sl:sent_err_replies = 0",
132     "sl:sent_replies = 76018",
133     "sl:xxx_replies = 76014",
134     "tcp:con_reset = 0",
135     "tcp:con_timeout = 0",
136     "tcp:connect_failed = 0",
137     "tcp:connect_success = 0",
138     "tcp:current_opened_connections = 0",
139     "tcp:current_write_queue_size = 0",
140     "tcp:established = 0",
141     "tcp:local_reject = 0",
142     "tcp:passive_open = 0",
143     "tcp:send_timeout = 0",
144     "tcp:sendq_full = 0",
145     "tmx:2xx_transactions = 307515",
146     "tmx:3xx_transactions = 0",
147     "tmx:4xx_transactions = 12551",
148     "tmx:5xx_transactions = 0",
149     "tmx:6xx_transactions = 386",
150     "tmx:UAC_transactions = 0",
151     "tmx:UAS_transactions = 314069",
152     "tmx:active_transactions = 0",
153     "tmx:inuse_transactions = 0",
154     "tmx:rpl_absorbed = 43649",
155     "tmx:rpl_generated = 142170",
156     "tmx:rpl_received = 287155",
157     "tmx:rpl_relayed = 243506",
158     "tmx:rpl_sent = 385676",
159     "usrloc:location_contacts = 0",
160     "usrloc:location_expires = 0",
161     "usrloc:location_users = 0",
162     "usrloc:registered_users = 0"
163 ],
164 "id": 8599
165 }
```

3.8 Mongo Monitoring

3.8.1 Service monitoring

Mongo service status can be retrieved with the following command

```
1 [root@sre-cp ~]# systemctl status mongod
2 - mongod.service - MongoDB Database Server
3   Loaded: loaded (/usr/lib/systemd/system/mongod.service; enabled; vendor
4     ↪ preset: disabled)
5   Active: active (running) since gio 2023-01-12 15:12:45 CET; 9 months 4 days
6     ↪ ago
7     Docs: https://docs.mongodb.org/manual
8   Main PID: 1258 (mongod)
9   CGroup: /system.slice/mongod.service
10          1258 /usr/bin/mongod -f /etc/mongod.conf
11
12 Warning: Journal has been rotated since unit was started. Log output is
13     ↪ incomplete or unavailable.
```

3.8.2 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:

```
1 [root@sre-cp ~]# mongo
2 MongoDB shell version v5.0.13
3 connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&
4     ↪ gssapiServiceName=mongodb
5 Implicit session: session { "id" : UUID("31a2a49c-e4ad-4a5d-9764-898450fec607")
6     ↪ }
7 MongoDB server version: 5.0.13
8 =====
9 Warning: the "mongo" shell has been superseded by "mongosh",
10 which delivers improved usability and compatibility.The "mongo" shell has been
11     ↪ deprecated and will be removed in
12 an upcoming release.
13 For installation instructions, see
14 https://docs.mongodb.com/mongodb-shell/install/
```

```

12 =====
13 ---
14 The server generated these startup warnings when booting:
15     2023-01-12T15:12:41.813+01:00: Using the XFS filesystem is strongly
16     ↪ recommended with the WiredTiger storage engine. See http://dochub.mongodb
17     ↪ .org/core/prodnotes-filesystem
18     2023-01-12T15:12:44.786+01:00: Access control is not enabled for the
19     ↪ database. Read and write access to data and configuration is unrestricted
20     2023-01-12T15:12:44.786+01:00: /sys/kernel/mm/transparent_hugepage/
21     ↪ enabled is 'always'. We suggest setting it to 'never'
22     2023-01-12T15:12:44.786+01:00: /sys/kernel/mm/transparent_hugepage/
23     ↪ defrag is 'always'. We suggest setting it to 'never'
24 ---
25 ---
26     Enable MongoDB's free cloud-based monitoring service, which will then
27     ↪ receive and display
28     ↪ metrics about your deployment (disk utilization, CPU, operation
29     ↪ statistics, etc).
30
31     The monitoring data will be available on a MongoDB website with a
32     ↪ unique URL accessible to you
33     ↪ and anyone you share the URL with. MongoDB may use this information to
34     ↪ make product
35     ↪ improvements and to suggest MongoDB products and deployment options to
36     ↪ you.
37
38     To enable free monitoring, run the following command: db.
39     ↪ enableFreeMonitoring()
40     To permanently disable this reminder, run the following command: db.
41     ↪ disableFreeMonitoring()
42 ---
43 sre_location:SECONDARY>
  
```

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

```

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

```

13   "writableVotingMembersCount" : 4,
14   "optimes" : {
15     "lastCommittedOpTime" : {
16       "ts" : Timestamp(1697620274, 1),
17       "t" : NumberLong(9)
18     },
19     "lastCommittedWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
20     "readConcernMajorityOpTime" : {
21       "ts" : Timestamp(1697620274, 1),
22       "t" : NumberLong(9)
23     },
24     "appliedOpTime" : {
25       "ts" : Timestamp(1697620274, 1),
26       "t" : NumberLong(9)
27     },
28     "durableOpTime" : {
29       "ts" : Timestamp(1697620274, 1),
30       "t" : NumberLong(9)
31     },
32     "lastAppliedWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
33     "lastDurableWallTime" : ISODate("2023-10-18T09:11:14.341Z")
34   },
35   "lastStableRecoveryTimestamp" : Timestamp(1697620254, 1),
36   "electionParticipantMetrics" : {
37     "votedForCandidate" : true,
38     "electionTerm" : NumberLong(9),
39     "lastVoteDate" : ISODate("2023-08-04T14:02:09.274Z"),
40     "electionCandidateMemberId" : 2,
41     "voteReason" : "",
42     "lastAppliedOpTimeAtElection" : {
43       "ts" : Timestamp(1691157640, 1),
44       "t" : NumberLong(8)
45     },
46     "maxAppliedOpTimeInSet" : {
47       "ts" : Timestamp(1691157640, 1),
48       "t" : NumberLong(8)
49     },
50     "priorityAtElection" : 1,
51     "newTermStartDate" : ISODate("2023-08-04T14:02:13.753Z"),
52     "newTermAppliedDate" : ISODate("2023-08-04T14:02:25.595Z")
53   },
54   "members" : [
55     {
56       "_id" : 0,
57       "name" : "10.0.161.180:27017",
58       "health" : 1,

```

```

59     "state" : 2,
60     "stateStr" : "SECONDARY",
61     "uptime" : 4606205,
62     "optime" : {
63         "ts" : Timestamp(1697620274, 1),
64         "t" : NumberLong(9)
65     },
66     "optimeDurable" : {
67         "ts" : Timestamp(1697620274, 1),
68         "t" : NumberLong(9)
69     },
70     "optimeDate" : ISODate("2023-10-18T09:11:14Z"),
71     "optimeDurableDate" : ISODate("2023-10-18T09:11:14Z"),
72     "lastAppliedWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
73     "lastDurableWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
74     "lastHeartbeat" : ISODate("2023-10-18T09:11:22.273Z"),
75     "lastHeartbeatRecv" : ISODate("2023-10-18T09:11:20.959Z"),
76     "pingMs" : NumberLong(0),
77     "lastHeartbeatMessage" : "",
78     "syncSourceHost" : "10.0.161.183:27017",
79     "syncSourceId" : 2,
80     "infoMessage" : "",
81     "configVersion" : 1,
82     "configTerm" : 9
83   },
84   {
85     "_id" : 1,
86     "name" : "10.0.161.182:27017",
87     "health" : 1,
88     "state" : 2,
89     "stateStr" : "SECONDARY",
90     "uptime" : 24087522,
91     "optime" : {
92         "ts" : Timestamp(1697620274, 1),
93         "t" : NumberLong(9)
94     },
95     "optimeDate" : ISODate("2023-10-18T09:11:14Z"),
96     "lastAppliedWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
97     "lastDurableWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
98     "syncSourceHost" : "10.0.161.183:27017",
99     "syncSourceId" : 2,
100    "infoMessage" : "",
101    "configVersion" : 1,
102    "configTerm" : 9,
103    "self" : true,
104    "lastHeartbeatMessage" : ""

```

```

105     },
106     {
107         "_id" : 2,
108         "name" : "10.0.161.183:27017",
109         "health" : 1,
110         "state" : 1,
111         "stateStr" : "PRIMARY",
112         "uptime" : 1588046,
113         "optime" : {
114             "ts" : Timestamp(1697620274, 1),
115             "t" : NumberLong(9)
116         },
117         "optimeDurable" : {
118             "ts" : Timestamp(1697620274, 1),
119             "t" : NumberLong(9)
120         },
121         "optimeDate" : ISODate("2023-10-18T09:11:14Z"),
122         "optimeDurableDate" : ISODate("2023-10-18T09:11:14Z"),
123         "lastAppliedWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
124         "lastDurableWallTime" : ISODate("2023-10-18T09:11:14.341Z"),
125         "lastHeartbeat" : ISODate("2023-10-18T09:11:21.792Z"),
126         "lastHeartbeatRecv" : ISODate("2023-10-18T09:11:21.017Z"),
127         "pingMs" : NumberLong(0),
128         "lastHeartbeatMessage" : "",
129         "syncSourceHost" : "",
130         "syncSourceId" : -1,
131         "infoMessage" : "",
132         "electionTime" : Timestamp(1691157730, 1),
133         "electionDate" : ISODate("2023-08-04T14:02:10Z"),
134         "configVersion" : 1,
135         "configTerm" : 9
136     }
137 ],
138 "ok" : 1,
139 "$clusterTime" : {
140     "clusterTime" : Timestamp(1697620274, 1),
141     "signature" : {
142         "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
143         "keyId" : NumberLong(0)
144     }
145 },
146 "operationTime" : Timestamp(1697620274, 1)
147 }

```

One cluster member should be in the *PRIMARY* state, while all the others should be in the *SECONDARY* state.

3.9 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 SIPSACK 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:

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

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

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

Sample output:

```
1 [root@sre-cp1 ~]# pcs status
2 Cluster name: hacluster
3 Stack: corosync
4 Current DC: sre-cp2 (version 1.1.23-1.el7_9.1-9acf116022) - partition with
   ↔ quorum
5 Last updated: Wed Oct 18 11:16:08 2023
6 Last change: Thu Jul 20 16:21:13 2023 by root via cibadmin on sre-cp1
7
8 2 nodes configured
9 0 resource instances configured
10
11 Online: [ sre-cp1 sre-cp2 ]
12
13 Full list of resources:
14 Resource Group: Group1
```

```
15 ClusterIP1 (ocf::heartbeat:IPaddr2): Started sre-cp2
16 Kamailio1 (ocf::heartbeat:kamailio): Started sre-cp2
17
18 Resource Group: Group2
19 ClusterIP2 (ocf::heartbeat:IPaddr2): Started sre-cp1
20 Kamailio2 (ocf::heartbeat:kamailio): Started sre-cp1
21
22 Daemon Status:
23   corosync: active/enabled
24   pacemaker: active/enabled
25   pcsd: active/enabled
```

3.9.1 Put a CP node 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

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

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

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

The following command removes all nodes from the standby mode.

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

3.9.2 Disabling a resource

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

```
1 [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.

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


3.9.3 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>
```

3.9.4 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>
```

3.9.5 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.

3.9.6 Deleting a resource

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

3.9.7 Removing a node

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

3.9.8 Adding a node

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

3.9.9 Pcs backup and restore

Useful if a node is completely lost.

From a running node:

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

3.9.10 Log files

You can check log files at

`/var/log/cluster/corosync.log`

4 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`).

```

1 [root@sre-cp ~]# tshark -i eth0 -R sip
2 Running as user \"root\" and group \"root\". This could be dangerous.
3 Capturing on eth0
4
5 0.769872938 10.211.1.1 -> 10.210.1.5 SIP 386 Request: OPTIONS
6 sip:10.210.1.5:5060
7 0.778437740 10.210.1.5 -> 10.211.1.1 SIP 430 Status: 200 OK
8 0.994916328 10.211.1.1 -> 10.210.1.3 SIP 386 Request: OPTIONS
9 sip:10.210.1.3:5060
10 1.000359472 10.210.1.3 -> 10.211.1.1 SIP 430 Status: 200 OK
11 ...
  
```

If `sngrep` is installed on the SIP 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.

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

sngrep - SIP messages flow viewer							
Current Mode: Online [any]		Dialogs: 36					
Match Expression:		BPF Filter:					
Display Filter:							
#Idx	Method	SIP From	SIP To	Msgs	Source	Destination	Call State
[] 1	OPTIONS	ping@127.0.0.1	ping@94.107.230.100	3	10.0.161.64:5060	94.107.230.100:5060	
[] 2	OPTIONS	ping@127.0.0.1	ping@10.34.250.202	3	10.0.161.64:5060	10.34.250.202:5060	
[] 3	OPTIONS	ping@127.0.0.1	ping@10.34.250.203	3	10.0.161.64:5060	10.34.250.203:5060	
[] 4	OPTIONS	ping@127.0.0.1	ping@10.34.252.201	3	10.0.161.64:5060	10.34.252.201:5060	
[] 5	OPTIONS	ping@127.0.0.1	ping@10.34.252.202	3	10.0.161.64:5060	10.34.252.202:5060	
[] 6	OPTIONS	ping@127.0.0.1	ping@10.34.252.203	3	10.0.161.64:5060	10.34.252.203:5060	
[] 7	OPTIONS	ping@127.0.0.1	ping@10.0.16.62	3	10.0.161.64:5060	10.0.16.62:5162	
[] 8	OPTIONS	ping@127.0.0.1	ping@10.0.161.60	3	10.0.161.182:5060	10.0.161.60:5060	
[] 9	OPTIONS	ping@127.0.0.1	ping@10.0.161.60	1	10.0.161.190:5060	10.0.161.60:5060	
[] 10	OPTIONS	ping@127.0.0.1	ping@10.0.161.60	2	10.0.161.183:5060	10.0.161.60:5060	
[] 11	OPTIONS	testdaan@1.2.3.4	ping@10.0.161.94	1	10.0.161.64:5060	10.0.161.94:5060	
[] 12	OPTIONS	ping@127.0.0.1	ping@94.107.230.100	7	10.0.161.64:5060	94.107.230.100:5060	
[] 13	OPTIONS	ping@127.0.0.1	ping@10.34.250.202	7	10.0.161.64:5060	10.34.250.202:5060	
[] 14	OPTIONS	ping@127.0.0.1	ping@10.34.250.203	7	10.0.161.64:5060	10.34.250.203:5060	
[] 15	OPTIONS	ping@127.0.0.1	ping@10.34.252.201	7	10.0.161.64:5060	10.34.252.201:5060	
[] 16	OPTIONS	ping@127.0.0.1	ping@10.34.252.202	7	10.0.161.64:5060	10.34.252.202:5060	
[] 17	OPTIONS	ping@127.0.0.1	ping@10.34.252.203	7	10.0.161.64:5060	10.34.252.203:5060	
[] 18	OPTIONS	ping@127.0.0.1	ping@10.0.16.62	7	10.0.161.64:5060	10.0.16.62:5162	
[] 19	OPTIONS	ping@127.0.0.1	ping@10.0.161.60	7	10.0.161.190:5060	10.0.161.60:5060	
[] 20	OPTIONS	testdaan@1.2.3.4	ping@10.0.161.94	1	10.0.161.64:5060	10.0.161.94:5060	

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

```

1 [root@sre-cp ~]# netstat -anu
2 Active Internet connections (servers and established)
3 Proto Recv-Q Send-Q Local Address           Foreign Address         State
4 udp      0      0 10.0.161.182:42235     0.0.0.0:*
5 udp      0      0 10.0.161.182:5405      0.0.0.0:*
6 udp      0      0 127.0.0.1:48531       0.0.0.0:*
7 udp      0      0 0.0.0.0:53            0.0.0.0:*
8 udp      0      0 10.0.161.182:55379    0.0.0.0:*
9 udp      0      0 127.0.0.1:323         0.0.0.0:*
10 udp     0      0 10.0.161.182:5060     0.0.0.0:*
11 udp     0      0 127.0.0.1:5060        0.0.0.0:*
12 udp6    0      0 :::1:323              :::*
13 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**

```

1 [root@sre-cp ~]# netstat -ant
2 Active Internet connections (servers and established)
3 Proto Recv-Q Send-Q Local Address           Foreign Address         State
4 tcp      0      0 0.0.0.0:5432           0.0.0.0:*              LISTEN
5 tcp      0      0 127.0.0.1:25           0.0.0.0:*              LISTEN
6 tcp      0      0 127.0.0.1:8090         0.0.0.0:*              LISTEN
7 tcp      0      0 10.0.161.182:5060     0.0.0.0:*              LISTEN
8 tcp      0      0 127.0.0.1:5060        0.0.0.0:*              LISTEN
9 tcp      0      0 127.0.0.1:9001        0.0.0.0:*              LISTEN
10 tcp     0      0 0.0.0.0:27017         0.0.0.0:*              LISTEN
11 tcp     0      0 127.0.0.1:6666        0.0.0.0:*              LISTEN
12 tcp     0      0 0.0.0.0:6000          0.0.0.0:*              LISTEN
13 tcp     0      0 127.0.0.1:5555        0.0.0.0:*              LISTEN
14 tcp     0      0 0.0.0.0:10004         0.0.0.0:*              LISTEN
15 tcp     0      0 0.0.0.0:53            0.0.0.0:*              LISTEN
16 tcp     0      0 0.0.0.0:22            0.0.0.0:*              LISTEN
17 tcp     0      0 10.0.161.182:48764    10.0.161.182:27017     ESTABLISHED
18 tcp     0      0 127.0.0.1:50198       127.0.0.1:9001        TIME_WAIT
19 tcp     0      0 127.0.0.1:49848       127.0.0.1:9001        TIME_WAIT
20 tcp     0      0 127.0.0.1:48462       127.0.0.1:6666        ESTABLISHED
21 tcp     0      0 10.0.161.182:39652    10.0.161.181:10000    ESTABLISHED
22 tcp     0      0 10.0.161.182:39386    10.0.161.183:27017    ESTABLISHED
  
```

23	tcp	0	0	127.0.0.1:48452	127.0.0.1:6666	ESTABLISHED
24	tcp	0	0	10.0.161.182:45534	10.0.161.180:10000	ESTABLISHED
25	tcp	0	0	127.0.0.1:5555	127.0.0.1:33690	ESTABLISHED
26	...					

4.1 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

Note

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.

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

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

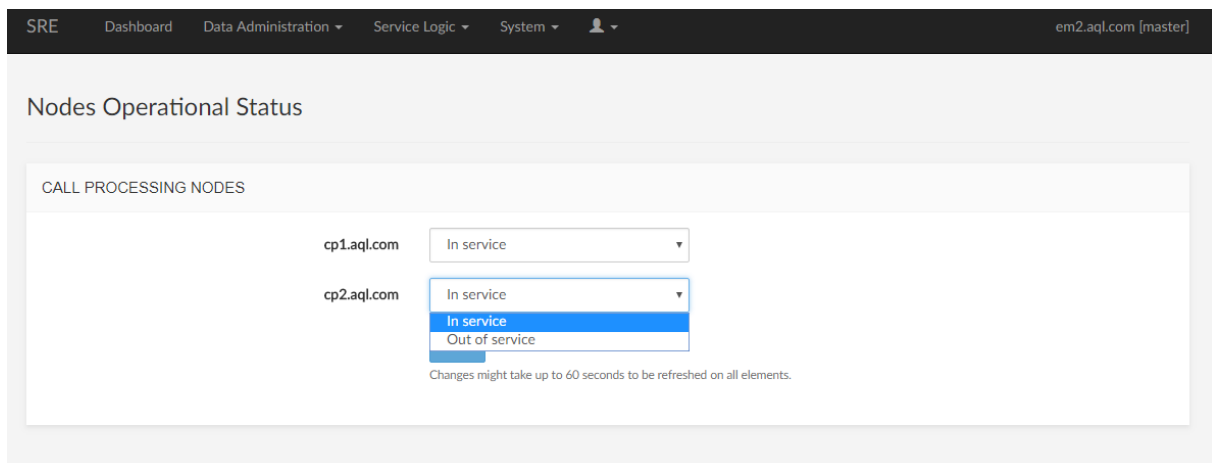
5 Nodes Operational Status

5.1 Putting a CP in 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*.



CALL PROCESSING NODES	
cp1.aql.com	In service
cp2.aql.com	In service

Changes might take up to 60 seconds to be refreshed on all elements.

6 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.

6.1 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.

6.2 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.

6.2.1 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 saved as a tar.gz file, containing the contents of the `/var/lib/pgsql/14/data` directory. 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`.

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

6.2.2 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.

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

As postgres user, delete the data into the PostgreSQL root directory `/var/lib/pgsql/14/data/`.

```
1 [root@sre-em ~]# rm -rf /var/lib/pgsql/14/data/*
```

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

```
1 [root@sre-em ~]# su - postgres
2 -bash-4.1$ cd /var/lib/pgsql/14/data/
3 -bash-4.1$ tar -zxvf /var/lib/pgsql/backup-20230212/base.tar.gz
```

Restart the PostgreSQL cluster and the sre software.

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

6.3 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.

6.3.1 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.

6.3.2 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
```

7 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.

7.1 Creating a full backup file

Note 1: Once the backup is created, download and store it somewhere external to the server!

Note 2: 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).

Note 3: 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 parts in yellow 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:

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

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

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

Comments:

- /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)

7.2 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 requirements 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):

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

Verify that the postgres db versions a and b for the customer are there, if not create them (e.g. `service_a` and `service_b`):

```
1 # su - postgres
2 > psql
3 postgres=# create database sre owner sre;
4 postgres=# create database <service>_a owner sre;
5 postgres=# create database <service>_b owner sre;
6 postgres=# \q
```

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

```
1 # cd /
```

This will restore all files in the original directories:

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

Then either:

- a. if you are restoring a postgres standby node, you need to resynch the node using [Node Re-Synchronisation](#).

- b. if it's the postgres master node that you are attempting to restore, then follow [Restore Procedure](#) from the Monitor SRE.

As postgres user, delete also the data into the PostgreSQL root directory `/var/lib/pgsql/14/data/`.

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

Restart the PostgreSQL cluster and the sre software.

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

Delete the content of the following directories before re-synching:

```
1 [root@sre-em ~]# rm -rf /var/lib/pgsql/14/data/*
```

```
1 [root@sre-em ~]# su - postgres
2 -bash-4.1$ cd /var/lib/pgsql/14/data/
3 -bash-4.1$ tar -zxvf /var/lib/pgsql/backup-20230415/base.tar.gz \...
```

Start postgresql and SRE:

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

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 there are half+1 nodes available after the restore.

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.

7.3 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.

7.3.1 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](#).

7.3.2 Restore Procedure (via 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.

Stop the SRE service, then PostgreSQL.

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

Delete all content of the main PostgreSQL directory.

```
1 [root@sre-cp ~]# rm -rf /var/lib/pgsql/14/data/*
```

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

```
1 [root@sre-cp ~]# su - postgres
2 -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
3 [2023-04-16 17:43:56] [NOTICE] Redirecting logging output to '/var/log/repmgr/
  ↪ repmgr-14.log'
4 -bash-4.1$ cat /var/log/repmgr/repmgr-14.log
5 [2023-04-14 17:52:05] [NOTICE] setting data directory to: /var/lib/pgsql/14/
  ↪ data
6 [2023-04-14 17:52:05] [HINT] use -D/--data-dir to explicitly specify a data
  ↪ directory
7 [2023-04-14 17:52:05] [NOTICE] starting backup (using pg_basebackup)...
8 [2023-04-14 17:52:05] [HINT] this may take some time; consider using the -c/--
  ↪ fast-checkpoint option
9 [2023-04-14 17:52:22] [NOTICE] standby clone (using pg_basebackup) complete
10 [2023-04-14 17:52:22] [NOTICE] you can now start your PostgreSQL server
11 [2023-04-14 17:52:22] [HINT] for example : /etc/init.d/postgresql start
```

Restart PostgreSQL, then SRE.

```
1 [root@sre-cp ~]# systemctl start postgresql-14
2 [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.

```
1 [root@sre-cp ~]# su - postgres
2 -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
```

8 PostgreSQL Cluster Switchover

8.1 Automatic switchover

Automatic switchover is enabled by default. To check the enablement status of automatic switchover run the following command as user postgres:

```

1 [postgres@sre4-em1 ~]$ /usr/pgsql-14/bin/repmgr service status
2 ID | Name | Role      | Status      | Upstream | repmgrd | PID   | Paused? |
   ↪ Upstream last seen
3 ----+-----+-----+-----+-----+-----+-----+-----+
   ↪
4 1  | em1  | primary  | * running  |          | running | 24710 | no       | n/a
5 2  | em2  | standby  | running    | em1      | running | 2365  | no       | 0
   ↪ second(s) ago
6 3  | cp1  | standby  | running    | em1      | running | 1936  | no       | 1
   ↪ second(s) ago
7 4  | cp2  | standby  | running    | em1      | running | 2373  | no       | 0
   ↪ second(s) ago
  
```

The *paused* column set to no denotes the activation of automatic switchover.

8.1.1 Disable automatic switchover

in order to disable automatic switchover run the following command as user postgres:

```
1 /usr/pgsql-14/bin/repmgr service pause
```

8.1.2 Enable automatic switchover

in order to enable again automatic switchover run the following command as user postgres:

```
1 /usr/pgsql-14/bin/repmgr service unpause
```

8.1.3 Doing manual switchover

Connect to the standby EM node and run the following command to check all the preconditions are met:

```
1 /usr/pgsql-14/bin/repmgr standby switchover --siblings-follow --dry-run
```

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

```
1 /usr/pgsql-14/bin/repmgr standby switchover --siblings-follow
```

8.2 Rebooting the master postgres node

If reboot of the master postgres node is needed for maintenance and automatic switchover is enabled, first pause it by following this [procedure](#) and then reboot the server.

If automatic switchover is disabled, a reboot can be performed without requiring any additional commands.

8.3 Rebooting the standby postgres node

A reboot can be performed without requiring any additional commands in any case.

8.4 Failure of the master postgres node

8.4.1 With automatic switchover

If automatic switchover is enabled nothing needs to be done on the new master.

When the old master is recovered its database is no more synchronized with the new master.

The `repmgr cluster show` will show an output similar to this one (in this case em1 is the failed master and em2 is the new master):

```
1 -bash-4.2$ /usr/pgsql-14/bin/repmgr cluster show
2 ID | Name | Role | Status | Upstream | Location | Priority |
   ↳ Timeline | Connection string
3 ----+-----+-----+-----+-----+-----+-----+-----+
   ↳
4 1 | em1 | primary | * running | | default | 100 |
   ↳ 11 | host=10.0.161.180 dbname=repmgr user=repmgr
5 2 | em2 | standby | ! running as primary | | default | 100 |
   ↳ 12 | host=10.0.161.181 dbname=repmgr user=repmgr
6 3 | cp1 | standby | running | ! em2 | default | 0 |
   ↳ 12 | host=10.0.161.182 dbname=repmgr user=repmgr
7 4 | cp2 | standby | running | ! em2 | default | 0 |
   ↳ 12 | host=10.0.161.183 dbname=repmgr user=repmgr
```

To rejoin the failed master (em1) run the following commands on that node:


```
1 [root@sre-em1 ~]$ systemctl stop postgresql-14
2 [root@sre-em1 ~]$ su - postgres
3 -bash-4.2$ /usr/pgsql-14/bin/repmgr node rejoin -d "host=<address of master>
    ↪ dbname=repmgr user=repmgr" --config-files=postgresql.local.conf,
    ↪ postgresql.conf --verbose --force-rewind
4 -bash-4.2$ exit
5 [root@sre-em1 ~]$ systemctl start postgresql-14
```

8.4.2 Without automatic switchover

If automatic switchover is disabled and 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 and referred to [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](#)).

8.4.2.1 Procedure with CP nodes automatically following

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.

```
1 [root@sre-em ~]# su - postgres
2 bash-4.2$ /usr/pgsql-14/bin/repmgr -f /etc/repmgr/14/repmgr.conf standby
    ↪ promote --siblings-follow --dry-run
3 INFO: node is a standby
4 INFO: no active primary server found in this replication cluster
5 INFO: all sibling nodes are reachable via SSH
6 INFO: 2 walsenders required, 10 available
```

```

7 INFO: node will be promoted using the "pg_promote()" function
8 INFO: prerequisites for executing STANDBY PROMOTE are met
9 If the test is ok, you can proceed with the real standby promotion
10 (removing the --dry-run)

```

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

```

1 [root@sre-em ~]# su - postgres
2 bash-4.1$ /usr/pgsql-14/bin/repmgr -f /etc/repmgr/14/repmgr.conf standby
   ↪ promote ---siblings-follow
3 NOTICE: promoting standby to primary
4 DETAIL: promoting server "sre-em" (ID: 2) using pg_promote()
5 NOTICE: waiting up to 60 seconds (parameter "promote_check_timeout") for
   ↪ promotion to complete
6 NOTICE: STANDBY PROMOTE successful
7 DETAIL: server "sre-em" (ID: 2) was successfully promoted to primary
8 NOTICE: executing STANDBY FOLLOW on 2 of 2 siblings
9 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:

```

1 [root@em2 ~]# su - postgres
2 -bash-4.2$ /usr/pgsql-14/bin/repmgr -f /etc/repmgr/14/repmgr.conf cluster show
3 ID | Name | Role   | Status   | Upstream | Location | Priority | Timeline |
   ↪ Connection string
4 -----+-----+-----+-----+-----+-----+-----+-----+
   ↪
5 1  | em1  | primary | failed  | ?        | default  | 100     |          |
   ↪ host=10.0.161.180 dbname=repmgr user=repmgr
6 2  | em2  | standby | * running |          | default  | 100     | 13      |
   ↪ host=10.0.161.181 dbname=repmgr user=repmgr
7 3  | cp1  | standby | running | em2      | default  | 0       | 13      |
   ↪ host=10.0.161.182 dbname=repmgr user=repmgr
8 4  | cp2  | standby | running | em2      | default  | 0       | 13      |
   ↪ host=10.0.161.183 dbname=repmgr user=repmgr
9
10 WARNING: following issues were detected
11 - 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.

8.4.2.2 Procedure with manual CP follow

Note

This procedure is not recommended for bigger deployments since the follow command must be launched on every standby node

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

```
1 [root@sre-em2 ~]# su - postgres
2 -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:

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

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

On the CP nodes:

```
1 [root@sre-cp ~]# su - postgres
2 -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.

9 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:

```
1 host all sre 10.0.161.64/32 reject
2 host replication repmgr 10.0.161.64/32 reject
3 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 [root@localhost ~]# 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.

10 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.

10.1 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.

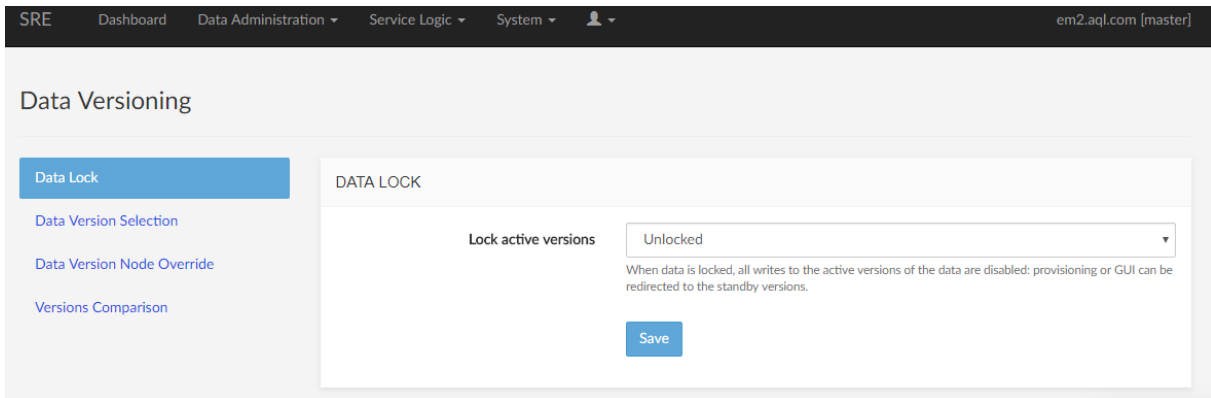


Figure 2: Graphical user interface, application, Teams Description automatically generated

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.

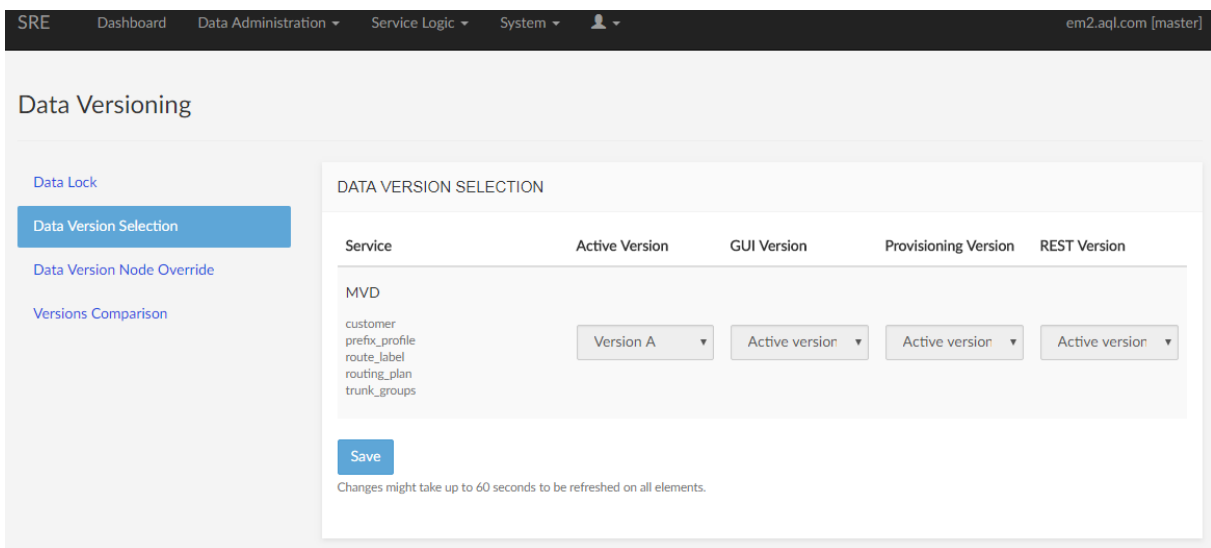


Figure 3: Graphical user interface, application, Teams Description automatically generated

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*.

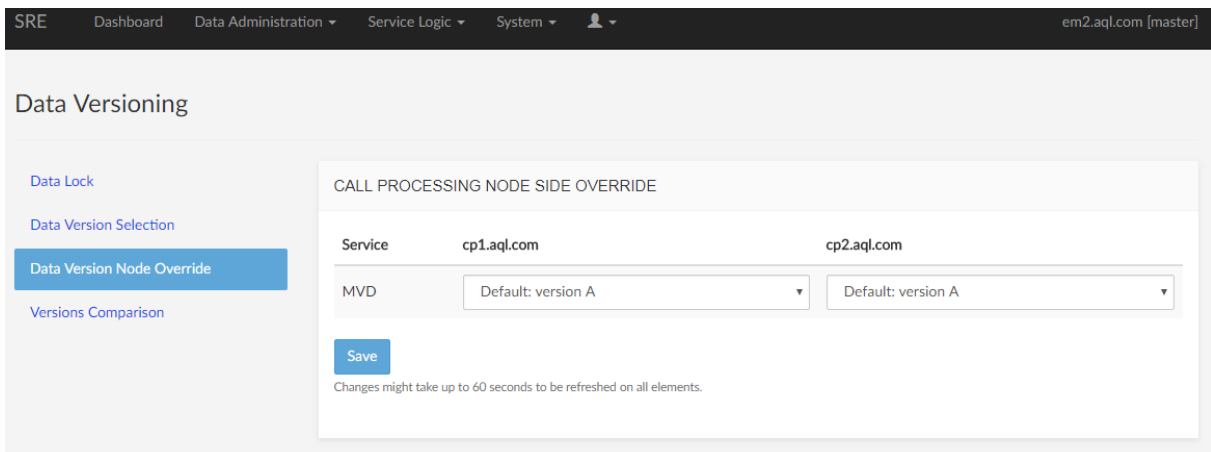


Figure 4: Graphical user interface, application Description automatically generated

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.

10.1.1 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.

Note

The option `-a` must be used to dump the data only. Without this option, the source database schema would be dumped too.

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:

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