SRE 4.1

Container Deployment Guide - Draft

Table of Contents

1 Hardware Requirements

1.1 CPU and Memory

The minimum requirements for an SRE server are:

- vCPU: 4
- Memory Size: 8192 MB
- Hard Disk Size: 250GB (SSD)

Note

A more tailored dimensioning is needed on a project basis.

1.2 Networking

SRE should be equipped with a minimum of 1 Gigabit Ethernet link.

Element Managers can be configured with 1 or 2 NIC interfaces

- 1 for Management (GUI/SSH access)
- 1 for DB replication between all components (EM's and CP's) and for CP modules control

Note

They can be combined in a single interface

Call Processors can be configured with multiple NIC Interfaces

- 1 for Management (SSH access)
- 1 for DB Replication and access to the element Managers
- 1 or more for the Call Processing in case SRE needs to communicate with multiple voice networks.

Note

They can be combined in a single interface

1.2.1 Communication Matrix

Source	Destination	Interface	Protocol/DestinationPort	Description
terminal	EM	management	TCP/22	ssh/sftp
EM	CP	internal	TCP/22	ssh
browser	EM	management	TCP/8080 (*)	http/https GUI access
external provisioners	EM	management	TCP/5000 (*) (**)	REST APIs
EM	DNS server	management	UDP/53 (**)	dns resolution

Source	Destination	Interface	Protocol/DestinationPort	Description
EM/CP	NTP server	management	UDP/123	time synchronization
EM	EM	internal	TCP/5432	DB traffic
CP	EM	internal	TCP/5432	DB traffic
CP	CP	internal	TCP/5555 (*) (**)	kamailio-broker traffic for hitless update
CP	EM	internal	TCP/5000 (*)	db updates from service logic
CP	EM	internal	TCP/10000	SRE log and stats
CP	EM	internal	TCP/10001	SRE internal requests
CP	EM	internal	TCP/10002	Accounting data
EM	EM	internal	TCP/10003	Accounting synchronization
EM	SNMP managers	management	UDP/162 (*) (**)	SNMP traps
EM	SMTP server	management	TCP/587 (**)	mail server
EM	syslog server	management	UDP/514 (*) (**)	syslog data
EM	LDAP server	management	TCP/389/636 (*) (**)	GUI authentication
SIP endpoint(s)	CP	SIP	UDP/TCP/5060 (*) (**)	SIP traffic interface
HTTP endpoints	CP	HTTP	TCP/6000 (*) (**)	http traffic interface
ENUM endpoints	CP	ENUM	UDP/TCP/53 (*) (**)	ENUM traffic interface

(*) port can be customized (**) optional

1.3 Disk partitioning

Create the necessary partitions according to the table below.

The example below is sized for a total disk space of 250GB.

Partition	Size	Type	Description
/var/lib/pgsql	70 GB	Ext4 or XFS on LVM	PostgreSQL database
/data/sre/db/backups	50 GB	Ext4 or XFS on LVM	workspace for backups
/data/sre/db/wals	20 GB	Ext4 or XFS on LVM	work-ahead logs
/data/sre/provisioning	10 GB	Ext4 or XFS on LVM	provisioning data (EM only)
/data/sre/mongo	10GB	Ext4 or XFS on LVM	Mongo database (only if CAC or global caching is configured)

Note

Partition dimensions should be tailored to the specific requirements of each project.

2 Software Requirements

2.1 Database

2.1.1 Postgres

Postgres 14 and repmgr package are required and must be installed on all nodes.

{#discover-data-dir}

Location of the data directory can be retrieved with this command:

```
sudo -u postgres psql -c "show data_directory;"
       data_directory
-----------------------------
 /var/lib/postgresql/14/main
```

2.1.1.1 Primary Element Manager Node Configuration

The procedure in this section should be executed on the Primary Element Manager only.

Start postgresql with:

```
sudo systemctl start postgresql-14 (or postgresql depending on your
    ↪ distribution)
sudo systemctl enable postgresql-14
```

2.1.1.1.1 Configure the database access

Locate *pg_hba.conf* location depending on your distribution and edit it:

```
sudo vi pg_hba.conf
```

This file manages the access rights to the database. It lists all possible origins from where connections can be made to the database and which authentication method should be used. We will always apply 'trust' as an authentication method which means that we unconditionally trust the specified sources and allow them to connect to the database without any specific verification or authentication. The first line which must be present allows the localhost to initiate connections over TCP/IP connections to the database. In the default configuration file this line is configured with authentication method = 'ident'. You must assure it is configured as 'trust' for local host connections to all databases:

IPv4 local connections:

```
host all all 127.0.0.1/32 trust
```

Also, all other possible 'trusted' sources for database queries must be granted access to the databases previously created. Later in this document we will spin up a standby element manager and call processing nodes. All these nodes must be granted access to the sre and custom databases. For the sake of simplicity access can be granted to an entire range of IP addresses but for security reasons it is sometimes recommended that single hosts should be accepted by using a subnetmask of /32. Nevertheless, we should only accept database connections when using the database user 'sre'.

IPv4 remote connections:

```
host sre sre <local_subnet>/<subnetmask> trust
host postgres postgres <local_subnet>/<subnetmask> trust
```

2.1.1.1.2 Postgresql.conf tuning

Locate *postgresql.conf* location depending on your distribution and edit it:

```
sudo vi postgresql.conf
```

uncomment and set the parameter listen_addresses to "*". Increase the maximum number of connections to 1000:

```
listen_addresses = '*'
max_connections = 1000
```

Make sure that the destination directory is writable by the user postgres, on all nodes. Unless different directory is used, the configuration parameters to use for WAL archiving in /data/sre/db/wals/ are:

```
archive_mode = on
archive_command = 'test ! -f /data/sre/db/wals/%f && cp %p /data/sre/db/wals/%f'
```

Uncomment and set the following parameters (in this example the system keeps 200 WALS files of 16 MB each):

```
wal_level = replica
max_wal_senders = 10
wal_keep_size = 3200
```

Enable hot_standby:

```
hot_standby = on
```

For automatic EM switchover add the following configuration settings:

```
wal_log_hints = on
shared_preload_libraries = 'repmgr'
```

Your configuration file should look like the following (in red the deviations from default values):

```
# -----------------------------
# PostgreSQL configuration file
# -----------------------------

............

#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -

listen_addresses = '*'          # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
                                        # defaults to 'localhost'; use '*' for all
                                        # (change requires restart)
#port = 5432                            # (change requires restart)
max_connections = 1000                  # (change requires restart)
#superuser_reserved_connections = 3     # (change requires restart)
#unix_socket_directories = '/var/run/postgresql, /tmp'  # comma-separated list of directories
                                        # (change requires restart)

...........
```

```


#------------------------------------------------------------------------------
# WRITE AHEAD LOG
#------------------------------------------------------------------------------

# - Settings -

wal_level = replica                 # minimal, archive, hot_standby, or logical
                                        # (change requires restart)

...................

# - Archiving -

archive_mode = on               # enables archiving; off, on, or always
                                # (change requires restart)
archive_command = 'test ! -f /data/sre/db/wals/%f && cp %p /data/sre/db/wals/%f
    ↪ '                 # command to use to archive a logfile segment
                                # placeholders: %p = path of file to archive
                                #               %f = file name only
                                # e.g. 'test ! -f /mnt/server/archivedir/%f &&
    ↪ cp %p /mnt/server/archivedir/%f'
#archive_timeout = 0            # force a logfile segment switch after this
                                # number of seconds; 0 disables
#------------------------------------------------------------------------------
# REPLICATION
#------------------------------------------------------------------------------

# - Sending Server(s) -

# Set these on the master and on any standby that will send replication data.

max_wal_senders = 10            # max number of walsender processes
                                # (change requires restart)
wal_keep_size = 3200            # in logfile segments, 16MB each; 0 disables
#wal_sender_timeout = 60s       # in milliseconds; 0 disables

..................

# - Standby Servers -

# These settings are ignored on a master server.

hot_standby = on                        # "on" allows queries during recovery
                                        # (change requires restart)
```

```
#max_standby_archive_delay = 30s        # max delay before canceling queries
                                        # when reading WAL from archive;
                                        # -1 allows indefinite delay
#max_standby_streaming_delay = 30s      # max delay before canceling queries
                                        # when reading streaming WAL;
                                        # -1 allows indefinite delay
#wal_receiver_status_interval = 10s     # send replies at least this often
                                        # 0 disables
#hot_standby_feedback = off             # send info from standby to prevent
                                        # query conflicts
#wal_receiver_timeout = 60s             # time that receiver waits for
                                        # communication from master
                                        # in milliseconds; 0 disables
#wal_retrieve_retry_interval = 5s       # time to wait before retrying to
                                        # retrieve WAL after a failed attempt
wal_log_hints = on
shared_preload_libraries = 'repmgr'
```

Restart the database after changing the file.

2.1.1.1.3 Postgresql replication

Create a user for repmgr to setup replication:

```
su - postgres
createuser -s repmgr
createdb repmgr -O repmgr
exit
```

Edit the configuration file *pg_hba.conf* to change access rights for the replication:

```
...
local   replication     repmgr          trust
host    replication     repmgr  127.0.0.1/32    trust
host    replication     repmgr  10.0.11.30/32   trust
local   repmgr  repmgr          trust
host    repmgr  repmgr  127.0.0.1/32    trust
host    repmgr  repmgr  10.0.11.30/32   trust
```

Restart the DB.

Edit the configuration file Postgresql cluster and set the parameters:

```
sudo vi repmgr.conf
```

```
node_id = <ID that should be unique in the SRE environment and greater than 0>
node_name=<name that should be unique in the SRE environment>
```

```
conninfo='host=<IP address of the server, this IP address should be accessible
    ↪ to all other nodes> dbname=repmgr user=repmgr'
data_directory='/var/lib/pgsql/14/data/'
ssh_options='-q -o ConnectTimeout=10'
failover='automatic'
reconnect_attempts=2
reconnect_interval=2
promote_command='/usr/pgsql-14/bin/repmgr standby promote -f /etc/repmgr/14/
    ↪ repmgr.conf --log-to-file; sudo docker restart sre'
 follow_command='/usr/pgsql-14/bin/repmgr standby follow -f /etc/repmgr/14/
    ↪ repmgr.conf --log-to-file --upstream-node-id=%n'
repmgrd_pid_file='/run/repmgr/repmgrd-14.pid'
always_promote=true
service_start_command = 'sudo systemctl start postgresql-14'
service_stop_command = 'sudo systemctl stop postgresql-14'
service_restart_command = 'sudo systemctl restart postgresql-14'
```

2.1.1.2 Secondary Element Manager and Call Processor Node Configuration

Cloning the databases In order to clone the database from the master database, ensure that all the tablespaces directories are exactly the same as the Primary Element Manager node and that the access rights are identical. Check that the PostgreSQL is not running:

```
sudo systemctl stop postgresql-14 (or postgresql depending on your distribution
    ↪ )
```

Ensure that the data directory is empty. See here for the procedure to discover this path.

```
sudo rm -rf <data_dir>/*
```

Edit the configuration file *repmgr.conf* set the parameters:

```
node_id = <ID that should be unique in the SRE environment and greater than 0>
node_name=<name that should be unique in the SRE environment>
conninfo='host=<IP address of the server, this IP address should be accessible
    ↪ to all other nodes> dbname=repmgr user=repmgr'
data_directory=<postgres data dir>
ssh_options='-q -o ConnectTimeout=10'
failover='automatic'
reconnect_attempts=2
reconnect_interval=2
promote_command='/<path to repmgr bin>/repmgr standby promote -f /<path to
    ↪ repmgr conf>/repmgr.conf --log-to-file; sudo docker restart sre'
follow_command='/<path to repmgr bin>/repmgr standby follow -f /<path to repmgr
    ↪  conf>/repmgr.conf --log-to-file --upstream-node-id=%n'
repmgrd_pid_file='<full path to repmgrd pid file>'
```

```
always_promote=true
service_start_command = 'sudo systemctl start postgresql-14' # or postgresql
service_stop_command = 'sudo systemctl stop postgresql-14' # or postgresql
service_restart_command = 'sudo systemctl restart postgresql-14' # or
    ↪ postgresql
```

Connect as user postgres and clone the PostgreSQL data directory files from the master. Be sure to copy also *postgresql.conf* and *pg_hba.conf* from master.

```
su - postgres
/<path to repmgr bin>/repmgr -h <IP Mater EM> -U repmgr -d repmgr -f /<path to
    ↪ repmgr conf>/repmgr.conf standby clone
exit
```

As root, start PostgreSQL and enable it at boot:

```
sudo systemctl start postgresql-14 (or postgresql)
sudo systemctl enable postgresql-14 (or postgresql)
```

As postgres user, register the standby server:

```
su - postgres
/<path to repmgr bin>/repmgr -f /<path to repmgr conf>/repmgr.conf standby
    ↪ register
exit
```

2.1.1.3 Repmgrd

On all nodes start and enable repmgrd daemon:

```
sudo systemctl start repmgrd
sudo systemctl enable repmgrd
```

2.1.2 Influxdb

Support is provided for InfluxDB OSS versions 2.4 through 2.7. Please refer to official site for installation instructions.

Note

Influxdb is needed only on EM nodes.

2.1.2.1 Configuration

As root, start Influxdb and enable it at boot:

```
[root@em ~]# systemctl start influxd
[root@em ~]# systemctl enable influxd
```

Configure it with:

```
[root@em ~]# influx setup -u influxuser -p influxuser -t my-super-secret-token
    ↪ -o influxorg -b bucket -r 1h (it will ask for confirmation)
[root@em ~]# influx bucket delete -n bucket
```

2.1.3 Mongodb (optional, required for CAC)

MongoDB 5.0.x is required on all nodes. Please refer to official site for installation instructions.

2.1.3.1 Configuration

Note

The number of nodes for MongoDB must be an odd number (N+1) to minimize the possibility for an outage of CAC. Indeed, if less than N/2 nodes (example 1 node out of 3) are unavailable, there is still a quorum (majority of nodes) therefore a PRIMARY node is available and the CAC feature is generally available.

```
sudo cat /etc/mongod.conf
# Where and how to store data.
storage:
    dbPath: /data/sre/mongodb
    journal:
        enabled: true

# network interfaces
net:
    port: 27017
    bindIp: 0.0.0.0

#security:

#operationProfiling:

replication:
    replSetName: sre_location
```

Start and enable mongodb at boot:

```
sudo chown -R mongod.mongod /data/sre/mongodb
sudo systemctl restart mongod
sudo systemctl enable mongod
```

2.1.3.1.1 Add arbiter (optional)

Note

It is recommended to configure one of EM nodes as mongodb arbiter if total number of SRE CP hosts is even.

```
sudo cat /etc/mongod.conf
# Where and how to store data.
storage:
    dbPath: /data/sre/mongodb/arb
    journal:
        enabled: true

# network interfaces
net:
    port: 27017
    bindIp: 0.0.0.0

#security:

#operationProfiling:

replication:
    replSetName: sre_location
```

Start and enable mongodb at boot:

```
sudo mkdir /data/sre/mongodb/arb
sudo chown -R mongod.mongod /data/sre/mongodb
sudo systemctl restart mongod
sudo systemctl enable mongod
```

2.1.3.1.2 Cluster initialization

Connect to one SRE host and start the mongodb shell with:

```
mongo (or mongosh)
```

Issue the following command:

```
> rs.initiate({_id : "sre_location", members: [{_id: 0, host: "<address1>" }]})
```

Add additional nodes to the cluster with (every node must have a unique id):

```
> rs.add({_id: 1, host: "<address2>" })
...
```

If an arbiter has been configured, add it with:

```
> rs.addArb("<address3>")
```

If all went well, issue the following command to check where the PRIMARY node is located:

```
> rs.status()
```

2.1.3.1.3 Setting write concern

Connect to primary node and on that node launch mongo cli and insert the following command:

```
> db.adminCommand({"setDefaultRWConcern" : 1,"defaultWriteConcern" : {"w" :
    ↪ 1}})
```

2.2 Container runtime

Docker 20.10.7 or higher is required on all nodes. Please refer to official site for installation instructions.

Note

Using Docker installations from alternative sources, like Snap, is discouraged.

3 SRE Deployment

3.1 Importing SRE images

On EMs and CPs:

```
sudo docker import sre-oci.tar.gz netaxis/sre
```

3.2 Extracting default configuration files

```
sudo mkdir -p /opt/sre/
sudo mkdir -p /var/log/sre/
sudo mkdir -p /date/sre/
```

3.3 SRE Configuration

3.4 Deploying SRE container

3.4.1 create directory structure for SRE data directory

On all nodes run:

```
sudo mkdir -p /data/sre/accounting/state
sudo mkdir -p /data/sre/db/backups
sudo mkdir -p /data/sre/db/wals
sudo mkdir -p /data/sre/provisioning
```

3.4.2 Start the container

3.4.3 Supported configuration options

Option name	Description	Default Value
DB_USER	user for connecting to postgres database	sre
DB_PASSWORD	password for connecting to postgres database	sre
DB_HOST	postgres host to connect (hostname or address)	initialize and use embedded database
DB_NAME	name of the database to create	sre
DB_PORT	postgres port to connect	5432
REPMGR_NODE_ID	repmgr node id to use	1
INFLUXDB_HOSTS	list of comma separated influxdb hostnames or addresses	initialize and use embedded database
INFLUXDB_TOKENS	list of comma separated influxdb tokens for connection	generate token for embedded database

Option name	Description	Default Value
INFLUXDB_ORG	influxdb organization to use for connection	influxorg
INFLUXDB_USER	influxdb user to use for connection	influxuser
INFLUXDB_PASSWORD	influxdb password to use for connection	influxpassword
ENABLE_MONGODB	enable(1) or disable(0) embedded mongodb	1 (enabled)
MONGODB_HOSTS	list of comma separated mongodb hostnames or addresses	use embedded mongodb
ENABLE_MANAGER	make this instance a manager(1) or a client(0)	1 (enabled)
ENABLE_KAMAILIO	enable(1) or disable(0) embedded kamailio	1 (enabled)
KAMAILIO_PORT	use this port for sip traffic	5060
KAMAILIO_EXTRA_DEFINES	list of comma separated defines for kamailio	no extra defines
NUM_KAMAILIO_WORKERS	number of kamailio workers to start	8
NUM_CALL_PROCESSORS	number of call processor instances to start	1
ADMIN_PASSWORD	gui admin password	admin
ENABLE_HTTPS	enable(1) or disable(0) https for gui	0 (disabled)
GUI_PORT	listen port for gui	8080
API_PORT	listen port for rest apis	5000
ENABLE_ENUM_PROCESSOR	enable(1) or disable(0) enum processor	0 (disabled)
ENUM_PROCESSOR_PORT	listen port for enum traffic	53
NUM_ENUM_PROCESSORS	number of enum processor instances to start	1
ENABLE_HTTP_PROCESSOR	enable(1) or disable(0) http processor	0 (disabled)
HTTP_PROCESSOR_PORT	listen port for http traffic	6000
NUM_HTTP_PROCESSORS	number of http processor instances to start	1
MANAGER_HOSTS	list of comma separated of manager node addresses	127.0.0.1 (single node deployment)
IMPORT_PATH	path for importing datamodels and service logics at start	/import

3.4.4 Sample deploy of 2 EMs and 2 CPs

3.4.4.1 EMs

To start container on first EM:

```
sudo docker run -d --name sre -v /var/log/sre:/var/log/sre -v /data/sre:/data/
    ↪ sre --network host \
    -e MANAGER_HOSTS=<em1 address>,<em2 address> \
    -e DB_HOST=<em1 address> \
    -e MONGODB_HOSTS=<em1 address>,<em2 address> \
    -e INFLUXDB_HOSTS=<em1 address>,<em2 address> \
    -e INFLUXDB_TOKENS=<influxdb token on em1>,<influxdb token on em2> \
    --restart=always netaxis/sre
```

To start container on second EM:

```
sudo docker run -d --name sre -v /var/log/sre:/var/log/sre -v /data/sre:/data/
    ↪ sre --network host \
    -e MANAGER_HOSTS=<em1 address>,<em2 address> \
    -e DB_HOST=<em2 address> \
    -e MONGODB_HOSTS=<em1 address>,<em2 address> \
    -e INFLUXDB_HOSTS=<em1 address>,<em2 address> \
    -e INFLUXDB_TOKENS=<influxdb token on em1>,<influxdb token on em2> \
    --restart=always netaxis/sre
```

3.4.4.2 CPs

Start the container with the following options:

```
sudo docker run -d --name sre -v /var/log/sre:/var/log/sre -v /data/sre:/data/
    ↪ sre --network host \
    -e ENABLE_MANAGER=0 \
    -e DB_HOST=<cp address> \
    -e MONGODB_HOSTS=<em1 address>,<em2 address> \
    --restart=always netaxis/sre
```

3.4.5 All-in-one deployment

For an all-in-one deployment only docker runtime is required on the host.

To start SRE run:

```
sudo docker run -d --name sre -v /var/log/sre:/var/log/sre -v /data/sre:/data/
    ↪ sre --network host --restart=always netaxis/sre
```