REDCURRANT 1.8.1 INSTALLATION GUIDE

AUTHOR(S)	:	Redcurrant development team
DOCUMENT NUMBER	:	RC-INST-1801
VERSION	:	1.1
STATUS	:	Final
SOURCE	:	Atos Worldline
DOCUMENT DATE	:	May 17th 2013April 4, 2013
NUMBER OF PAGES	:	33

version: 1.0 **document number:** RC-INST-1801

Contents

1 Introduction 4

1.1 Requirements 4
1.2 Convention 4
1.3 Installation set 4
1.4 Path convention 5
1.5 Note 5

2 Installation 6

2.1 Packages installation 6
2.2 Zookeeper 6
2.3 Gridinit and gridagent 6
2.4 Conscience 7
2.5 Meta0 9
2.6 Meta1 10
2.7 Meta2 11
2.8 Container purge crawler 12
2.9 Container deduplication crawler 13
2.10 RAWX 13
2.11 SQLX 15
2.12 Configure conscience 16

3 Command line 18

3.1 Content management 18
3.2 Service management 20

4 Troubleshoot 22

4.1 Score 22
4.2 Permissions 22
4.3 Log files 22

5 Multi node installation 23

5.1 Advised distribution 23

List of changes

Version	Date	Description	Author(s)
1.0	17/05/13	Initial version for Redcurrant 1.8.1	Redcurrant devel team

1 Introduction

The goal of this document is to describe a proper way to install Redcurrant product in a single node installation mode. The reader is expected to be familiar with Redcurrant concepts (namespace, container, content) and services (meta0,meta1,meta2,rawx, conscience …)

1.1 Requirements

1.1.1 Hardware

There is no hardware requirement except for a x86 CPU with 10GB disk space
This single node installation procedure does not expect any specific storage device nor capacity and can be executed inside a VM.

1.1.2 Software

Operating System: CentOS 5 or 6, 64b (advised)
Sun/Oracle JVM, version 1.6 +
Yum repositories with Redcurrant software as well as EPEL must be available from the server. Yum configuration is not part of the present document.

1.2 Convention

The following notation will be used across the document.

Command lines to be executed are in bordered paragraph, grey background, 'courrier new' font.

```
root@mysystem:> Run_this_command this_parameter
```

Custom parameters are always preceded by a $ char. Example: $tcp_port

Either on command line or configuration files, custom parameters are in **bold red**.

```
<xml>
   <namespace> $your value </namespace>
</xml>
```

1.3 Installation set

The minimum installation set for Redcurrant 1.8 is composed of:

- 1 Zookeeper
- 1 Meta0 service
- 3 Meta1 service
- 1 Conscience
- 1 Gridinit
- 1 Gridagent

Such a reduced installation only allows container management and score management.

The present document will provide a guide toward a more complete installation, with:

- 2 Meta2
- 2 Rawx

- 3 sqlx

This installation allows content management

1.4 Path convention

Redcurrant relies on two main rootDir. /GRID for configuration, /DATA for persistent data storage.

/GRID				
/GRID	/$hostname			
/GRID	/$hostname	/conf		Configuration file of server dependent process, like gridinit, gridagent
/GRID	/$namespace			
/GRID	/$namespace	/$storagedevice		
/GRID	/$namespace	/$storagedevice	/conf	Configuration file of namespace dependent process, like meta0, meta1 meta2 conscience
/DATA				
/DATA	/$namespace			
/DATA	/$namespace	/$storagedevice		
/DATA	/$namespace	/$storagedevice	/$service	Depending on the service, this mountpoint must be sized appropriately

1.5 Note

Please note that commands output in the following examples might slightly differ sometimes, mainly due to IP address, node name, namespace name.

2 Installation

In this tutorial, we will configure service to listen on TCP as follow

Process	Port	Later referenced as
Conscience	6000	$port_conscience
Meta0	6001	$port_meta0
Meta1	6002 6003 6004	$port_meta1
Meta2	6005 6006	$port_meta2
Rawx	6007 6008 6015	$port_rawx
Sqlx	6014 6016 6017	$port_sqlx

2.1 Packages installation

```
[root@singlenode ~]# yum install redcurrant-server redcurrant-common redcurrant-
client
[root@singlenode ~]# yum install zookeeper
[root@singlenode ~]# yum install grid-init.x86_64
[root@singlenode ~]# yum install redcurrant-mod-httpd.x86_64
[root@singlenode ~]# yum install redcurrant-rsyslog.noarch
[root@singlenode ~]# yum install redcurrant-python
[root@singlenode ~]# yum install redcurrant-java
[root@singlenode ~]# yum install JavaAsn1Compiler
```

BEWARE: Zookeeper requires Sun jdk, and does not work well with gcj / openjdk. Be sure to check that default java is the Sun edition, version >= 1.6

```
[root@singlenode ~]# java -version
java version "1.6.0_29"
Java(TM) SE Runtime Environment (build 1.6.0_29-b11)
Java HotSpot(TM) 64-Bit Server VM (build 20.4-b02, mixed mode)
```

2.2 Zookeeper

Create /etc/gridstorage.conf.d/$namespace

```
[$namespace]
conscience=$ip:$port_conscience
zookeeper=$ip:2181
```

Start zookeeper at boot

```
[root@singlenode ~]# chkconfig zookeeper on
```

2.3 Gridinit and gridagent

Prepare directory

```
[root@singlenode ~]# mkdir -p /GRID/$singlenode/{conf,core,logs,run,spool}
[root@singlenode ~]# chown -R admgrid:admgrid /GRID/$singlenode/
[root@singlenode ~]# mkdir /GRID/common/run
[root@singlenode ~]# chown admgrid:admgrid  /GRID/common/run
```

Prepare configuration

```
[root@singlenode conf]# cd /GRID/$singlenode/conf/
[root@singlenode conf]# cp /GRID/common/conf/gridinit.* .
```

Replace /GRID/common by /GRID/$singlenode in the copied configuration files
Prepare gridinit configuration directory. Gridinit will process each file found there:

```
[root@singlenode conf]# mkdir gridinit.conf.d
```

Create a service file for gridagent

```
[Service.gridagent]
command=/usr/local/bin/gridagent --supervisor /GRID/common/conf/gridagent.conf
/GRID/common/conf/gridagent.log4crc
enabled=true
start_at_boot=yes
on_die=respawn
group=common,common,$singlenode,gridagent
env.PATH=/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

Now start gridinit

```
[root@singlenode gridinit.conf.d]# /etc/init.d/gridinit start
```

And check status

```
[root@singlenode gridinit.conf.d]# gridinit_cmd status
KEY        STATUS      PID GROUP
gridagent UP          3282 common,common,$singlenode,gridagent
```

From now on, gridinit and gridagent are ready.

Start gridinit at boot

```
[root@singlenode ~]# chkconfig --add gridinit
[root@singlenode ~]# chkconfig gridinit on
```

Next service will be namespace bound. Prepare namespace level directories

```
[root@singlenode gridinit.conf.d]# mkdir -p
/GRID/$namespace/{core,run,logs,$singlenode}
[root@singlenode gridinit.conf.d]# mkdir -p
/GRID/$namespace/$singlenode/{core,conf,logs,run}
[root@singlenode gridinit.conf.d]# chown -R admgrid:admgrid /GRID/$namespace
```

2.4 Conscience

Create a service file for conscience (/GRID/$singlenode/conf/gridinit.conf.d/conscience-1)

```
[Service.$namespace-$singlenode-conscience-1]
command=/usr/local/bin/gridd /GRID/$namespace/$singlenode/conf/conscience-
1.conf /GRID/$namespace/$singlenode/conf/conscience-1.log4crc
enabled=true
start_at_boot=yes
on_die=respawn
group=$namespace,$singlenode,$singlenode,conscience
env.PATH=/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

The file reference namespace wide configuration. Prepare directories:

```
[root@singlenode gridinit.conf.d]# mkdir -p
/GRID/$namespace/$singlenode/{conf,run}
```

```
[root@singlenode gridinit.conf.d]# chown -R admgrid:admgrid
/GRID/TESTNS/singlenode/
```

Create conscience configuration file /GRID/$namespace/$singlenode/conf/conscience-1.conf

```
[General]
daemon=false
to_op=300000
to_cnx=300000
pidfile=/GRID/$namespace/$singlenode/run/conscience-1.pid

[Service]
namespace=$namespace
type=conscience
register=false
load_ns_info=false

[Server.conscience]
min_workers=30
min_spare_workers=30
max_spare_workers=100
max_workers=200
listen=$port_conscience

plugins=conscience,stats,fallback

[Plugin.stats]
path=/usr/local/lib64/grid/msg_stats.so

[Plugin.fallback]
path=/usr/local/lib64/grid/msg_fallback.so

[Plugin.conscience]
path=/usr/local/lib64/grid/msg_conscience.so
param_namespace=$namespace
param_chunk_size=10485760
param_score_timeout=300
param_meta0=$ip:$port_meta0

param_events=/GRID/$namespace/$singlenode/conf/conscience-1.events

param_service.default.score_timeout=300
param_service.default.score_variation_bound=5
param_service.default.score_expr=100
```

Create event configuration file /GRID/$namespace/$singlenode/conf/conscience-1.events

```
*=drop
```

Create log4c configuration file /GRID/$namespace/$singlenode/conf/conscience-1.log4crc

```
[root@singlenode conf]# cp gridinit.log4crc
/GRID/$namespace/$singlenode/conf/conscience-1.log4crc
```

Edit appender name to match RC,$namespace,$singlenode,conscience

Now reload gridinit.

```
[root@singlenode gridinit.conf.d]# /etc/init.d/gridinit reload
```

Check status

```
[root@singlenode gridinit.conf.d]# gridinit_cmd status
KEY                             STATUS      PID GROUP
gridagent                       UP         3282 common,common,singlenode,gridagent
```

```
TESTNS-singlenode-conscience-1 UP        10762
TESTNS,singlenode,singlenode,conscience
```

Request status from conscience

```
[root@singlenode gridinit.conf.d]# gridcluster $namespace

NAMESPACE INFORMATION

                Name : $namespace
          Chunk size : 10485760 bytes
              Option : writable_vns = $namespace


-- meta0 --
                Addr                    Score
$ip:$port_meta0                       100


-- meta2 --

-- meta1 --

-- rawx -
```

From now on, conscience is ready.

2.5 Meta0

Create a service file for meta0 (/GRID/$singlenode/conf/gridinit.conf.d/meta0-1)

```
[Service.$namespace-$singlenode-meta0-1]
command=/usr/local/bin/meta0_server -v -p /GRID/$namespace/$singlenode/run/meta0-
1.pid -s RC,$namespace,$singlenode,meta0-1 -O Endpoint=$ip:$meta0_port $namespace
/DATA/$namespace/$singlenode/meta0-1
enabled=true
start_at_boot=yes
on_die=respawn
group=$namespace,$singlenode,$singlenode,meta0
env.PATH=/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

Prepare data directory

```
[root@singlenode gridinit.conf.d]# mkdir -p /DATA/$namespace/$sinlenode/meta0-1
[root@singlenode gridinit.conf.d]# chown -R admgrid:admgrid /DATA
```

Reload gridinit

```
[root@singlenode gridinit.conf.d]# /etc/init.d/gridinit reload
```

Check status

```
[root@singlenode gridinit.conf.d]# gridinit_cmd status
KEY                           STATUS      PID GROUP
gridagent                     UP         3199 common,common,singlenode,gridagent
TESTNS-singlenode-conscience-1 UP        3198
TESTNS,singlenode,singlenode,conscience
TESTNS-singlenode-meta0-1     UP         3374 TESTNS,singlenode,singlenode,meta0
```

2.6 Meta1

Create a service file for each meta1 (/GRID/$singlenode/conf/gridinit.conf.d/meta1-{1,2,3})

```
[Service.$namespace-$singlenode-meta1-1]
command=/usr/local/bin/meta1_server -v -p /GRID/$namespace/$singlenode/run/meta1-
1.pid -s RC,$namespace,$singlenode,meta1-1 -O Endpoint=$ip:$meta1_port $namespace
/DATA/$namespace/$singlenode/meta1-1
enabled=true
start_at_boot=yes
on_die=respawn
group=$namespace,$singlenode,$singlenode,meta1
env.PATH=/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

Prepare data directory

```
[root@singlenode gridinit.conf.d]# mkdir /DATA/$namespace/$singlenode/meta1-
{1,2,3}
[root@singlenode gridinit.conf.d]# chown admgrid:admgrid
/DATA/$namespace/$singlenode/meta1*
```

Reload gridinit

```
[root@singlenode gridinit.conf.d]# /etc/init.d/gridinit reload
```

Check status

```
[root@singlenode gridinit.conf.d]# gridinit_cmd status
KEY                            STATUS      PID GROUP
gridagent                      UP         3199 common,common,singlenode,gridagent
TESTNS-singlenode-conscience-1 UP         3198
TESTNS,singlenode,singlenode,conscience
TESTNS-singlenode-meta0-1      UP         3374 TESTNS,singlenode,singlenode,meta0
TESTNS-singlenode-meta1-1      UP         3429 TESTNS,singlenode,singlenode,meta1
TESTNS-singlenode-meta1-2      UP         3431 TESTNS,singlenode,singlenode,meta1
TESTNS-singlenode-meta1-3      UP         3430 TESTNS,singlenode,singlenode,meta1
```

Check conscience. Meta1 services are now displayed

```
[root@singlenode gridinit.conf.d]# gridcluster TESTNS

NAMESPACE INFORMATION

               Name : TESTNS
         Chunk size : 10485760 bytes
             Option : writable_vns = TESTNS


-- meta0 --
               Addr                    Score
192.168.244.143:6001                     100


-- meta2 --

-- meta1 --
               Addr                    Score
192.168.244.143:6004                      98
192.168.244.143:6003                      98
192.168.244.143:6002                      98
```

```
-- rawx -
```

2.6.1 Initialize RC

Zookeeper

```
[root@singlenode conf]# zk-bootstrap.py $namespace
```

Meta0

```
[root@singlenode conf]# meta0_init -O NbReplicas=3  $namespace
1335779987.321 03547 5228 INF g.m.c   Ready to configure [3] META1
1335779989.114 03547 5228 INF g.m.c   META0 filled!
```

Restart RC

```
[root@singlenode conf]# /etc/init.d/gridinit restart
```

2.7 Meta2

Create a service file for each meta2 (/GRID/$singlenode/conf/gridinit.conf.d/meta2-{1,2})

```
[Service.$namespace-$singlenode-meta2-1]
command=/usr/local/bin/meta2_server -vv -p /GRID/$namespace/$singlenode/run/meta2-
1.pid -s RC,$namespace,$singlenode,meta2-1 -O Endpoint=$meta2_ip:$meta2_port -O
Tag=location=$datacenter.$room.$rack.$cluster.$server.$volume $namespace
/DATA/$namespace/$singlenode/meta2-1
enabled=true
start_at_boot=yes
on_die=respawn
group=$namespace,$singlenode,$singlenode,meta2
env.PATH=/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

Tag location is not mandatory for meta2_service. Nevertheless, it can help improving data availability in case of major disaster, by forcing replication between services at the desired distance

Prepare data directory

```
[root@singlenode conf]# mkdir /DATA/TESTNS/singlenode/meta2-{1,2}
[root@singlenode conf]# chown admgrid:admgrid /DATA/TESTNS/singlenode/meta2*
```

Reload gridinit

```
[root@singlenode conf]# /etc/init.d/gridinit reload
```

Check status

```
[root@singlenode conf]# gridinit_cmd status
KEY                           STATUS      PID GROUP
gridagent                     UP         3199 common,common,singlenode,gridagent
TESTNS-singlenode-conscience-1 UP         3198
TESTNS,singlenode,singlenode,conscience
TESTNS-singlenode-meta0-1     UP         3374 TESTNS,singlenode,singlenode,meta0
TESTNS-singlenode-meta1-1     UP         3429 TESTNS,singlenode,singlenode,meta1
TESTNS-singlenode-meta1-2     UP         3431 TESTNS,singlenode,singlenode,meta1
TESTNS-singlenode-meta1-3     UP         3430 TESTNS,singlenode,singlenode,meta1
TESTNS-singlenode-meta2-1     UP         3500 TESTNS,singlenode,singlenode,meta2
TESTNS-singlenode-meta2-2     UP         3501 TESTNS,singlenode,singlenode,meta2
```

Check conscience

```
[root@singlenode conf]# gridcluster TESTNS

NAMESPACE INFORMATION

                Name : TESTNS
          Chunk size : 10485760 bytes
              Option : writable_vns = TESTNS


-- meta0 --
                Addr                    Score
192.168.244.143:6001                      100


-- meta2 --
                Addr                    Score
192.168.244.143:6005                        0
192.168.244.143:6006                        0


-- meta1 --
                Addr                    Score
192.168.244.143:6004                       25
192.168.244.143:6003                       25
192.168.244.143:6002                       25
```

2.8 Container purge crawler

A container crawler is needed to handle deletion of chunks which were used in deleted, deduplicated or out of versions contents. This crawler is composed of two parts:

- A purge service which is unique per storage node and multi-namespace
- A container crawler which is dedicated to a meta2 data volume

To start the purge service, create the following gridinit service config file:

```
[Service.common-crawler-purge-1]
command=/usr/local/bin/action_purge_container_service -s
"RC,common,common,crawler_purge-1"
enabled=true
on_die=respawn
group=common,crawler-purge
env.PATH=/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

To start the container crawler, create a gridinit service config file like the following for each meta2 service:

```
[Service.$namespace-$singlenode-purge_crawler-1]
command=/usr/local/bin/crawler -Oinfinite=on -Otrip=trip_container -
Oaction=action_purge_container -- -
trip_container.s=/DATA/$namespace/$singlenode/meta2-1 -
action_purge_container.n=$namespace
enabled=true
on_die=respawn
group=common,crawler-purge
env.PATH=/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

2.9 Container deduplication crawler

A deduplication crawler is available to deduplicate identical contents at a container level. This crawler is composed of two parts:

- A deduplication service which is unique per storage node and multi-namespace
- A container crawler which is dedicated to a meta2 data volume

To start the dedup service, create the following gridinit service config file:

```
[Service.common-crawler-dedup-1]
command=/usr/local/bin/action_dedup_container_service -s
"RC,common,common,crawler_dedup-1"
enabled=true
on_die=respawn
group=common,crawler-dedup
env.PATH=/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

To start the deduplication crawler, create a gridinit service config file like the following for each meta2 service:

```
[Service.$namespace-$singlenode-dedup_crawler-1]
command=/usr/local/bin/crawler -Oinfinite=on -Otrip=trip_container -
Oaction=action_dedup_container -- -
trip_container.s=/DATA/$namespace/$singlenode/meta2-1 -
action_dedup_container.n=$namespace
enabled=true
on_die=respawn
group=common,crawler-dedup
env.PATH=/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

2.10 RAWX

Create a service file for each rawx. This example will iterate on vol01 and vol02.

```
[Service.$namespace-$singlenode-rawx-vol01]
command=rawx-monitor /GRID/$namespace/$singlenode/conf/rawx-vol01-monitor.conf
/GRID/$namespace/$singlenode/conf/rawx-vol01-monitor.log4crc
enabled=true
start_at_boot=yes
on_die=respawn
group=$namespace,$singlenode,$singlenode,rawx
env.PATH=/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

Create configuration for each rawx monitor (/GRID/$namespace/$singlenode/conf/rawx-vol01-monitor.conf

```
[Default]
daemon=false
pidfile=/GRID/$namespace/$singlenode/run/rawx-vol01-monitor.pid

[Child]
command=/usr/sbin/httpd.worker -D FOREGROUND -f
/GRID/$namespace/$singlenode/conf/rawx-vol01-httpd.conf
respawn=true
rlimit.stack_size=1048576
rlimit.core_size=-1
rlimit.max_files=32768

[Service]
ns=$namespace
type=rawx
```

```
addr=$ip:$rawx_port
location=$datacenter.$room.$rack.$cluster.$server.$volume

[Volume]
Docroot=/DATA/$namespace/$singlenode/vol01
```

Do not forget the directive location, mandatory in RC 1.8. Location specifies the physical location of the rawx service, used by storage policy engine. Please refer to the administration guide for details on URL syntax.

Create httpd config file for each rawx monitor (/GRID/$namespace/$singlenode/conf/rawx-vol01-httpd.conf)

```
LoadModule dav_module        /usr/lib64/httpd/modules/mod_dav.so
LoadModule dav_rawx_module   /usr/lib64/httpd/modules/mod_dav_rawx.so
LoadModule mime_module          /usr/lib64/httpd/modules/mod_mime.so
#LoadModule log_config_module /usr/lib64/httpd/modules/mod_log_config.so

Listen $ip:$raw_port
PidFile /GRID/$namespace/$singlenode/run/rawx-vol01-httpd.pid
ServerRoot /GRID/$namespace/$singlenode/core
ServerName localhost
ServerSignature Off
ServerTokens Prod
DocumentRoot /GRID/$namespace/$singlenode/run
TypesConfig /etc/mime.types

User  admgrid
Group admgrid

#LogFormat "%h %l %u %t \"%r\" %>s %b" log/common
ErrorLog /GRID/$namespace/$singlenode/logs/vol01-httpd-errors.log
#CustomLog /GRID/$namespace/$singlenode/logs/vol01-httpd-access.log log/common
LogLevel debug

<IfModule mod_env.c>
SetEnv nokeepalive 1
SetEnv downgrade-1.0 1
SetEnv force-response-1.0 1
</IfModule>

<IfModule prefork.c>
MaxClients       150
StartServers       5
MinSpareServers    5
MaxSpareServers   10
</IfModule>

<IfModule worker.c>
StartServers           5
MaxClients           100
MinSpareThreads        5
MaxSpareThreads       25
ThreadsPerChild       10
MaxRequestsPerChild    0
</IfModule>

DavDepthInfinity Off

grid_hash_width 2
```

```
grid_hash_depth 2
grid_docroot /DATA/$namespace/$singlenode/vol01
grid_namespace $namespace
grid_dir_run /GRID/$namespace/$singlenode/run

<Directory />
DAV rawx
AllowOverride None
</Directory>

<VirtualHost $ip:$rawx_port>
# DO NOT REMOVE (even if empty) !
</VirtualHost>
```

Create rawx directory. Chunk will be stored there

```
[root@singlenode gridinit.conf.d]# mkdir
/DATA/$namespace/$singlenode/{vol01,vol02}
[root@singlenode gridinit.conf.d]# chown admgrid:admgrid
/DATA/$namespace/$singlenode/vol*
```

Reload gridinit

```
[root@singlenode conf]# /etc/init.d/gridinit reload
```

Check status

```
[root@singlenode gridinit.conf.d]# gridinit_cmd status
KEY                           STATUS      PID GROUP
gridagent                     UP         3571 common,common,singlenode,gridagent
TESTNS-singlenode-conscience-1 UP        3596
TESTNS,singlenode,singlenode,conscience
TESTNS-singlenode-meta0-1     UP         3567 TESTNS,singlenode,singlenode,meta0
TESTNS-singlenode-meta1-1     UP         3628 TESTNS,singlenode,singlenode,meta1
TESTNS-singlenode-meta1-2     UP         3630 TESTNS,singlenode,singlenode,meta1
TESTNS-singlenode-meta1-3     UP         3629 TESTNS,singlenode,singlenode,meta1
TESTNS-singlenode-meta2-1     UP         3661 TESTNS,singlenode,singlenode,meta2
TESTNS-singlenode-meta2-2     UP         3662 TESTNS,singlenode,singlenode,meta2
TESTNS-singlenode-rawx-vol01  UP         5287 TESTNS,singlenode,singlenode,rawx
TESTNS-singlenode-rawx-vol02  UP         5363 TESTNS,singlenode,singlenode,rawx
```

2.11 SQLX

Create a service file for each sqlx. (/GRID/$singlenode/conf/gridinit.conf.d/sqlx-{1,2,3})

```
[Service.$namespace-$singlenode-sqlx-1]
command=/usr/local/bin/sqlx_server -s 'RC,$namespace,$singlenode,sqlx-1' -O
Endpoint=$ip:$port_sqlx $namespace /DATA/$namespace/$singlenode/sqlx-1/
enabled=true
start_at_boot=yes
on_die=respawn
group=$namespace,$singlenode,$singlenode,sqlx
env.PATH=/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

Create data directory

```
[root@singlenode ~]# mkdir -p /DATA/$namespace/$singlenode/sqlx-1/
[root@singlenode ~]# chown -R admgrid:admgrid /DATA/$namespace/$singlenode/sqlx-1/
```

Reload gridinit and check status

2.12 Configure conscience

Edit conscience configuration file (/GRID/$namespace/$singlenode/conf/conscience-1.conf), and add the following rules for score calculation in "[Plugin.conscience]" section.

```
param_service.meta1.score_timeout=300
param_service.meta1.score_variation_bound=5
param_service.meta1.score_expr=root(2,((num stat.cpu) * (num stat.req_idle)))

param_service.meta2.score_timeout=300
param_service.meta2.score_variation_bound=5
param_service.meta2.score_expr=((num stat.space)>=5) * root(3,((num stat.cpu)*(num
stat.req_idle)*(num stat.space)))

param_service.rawx.score_timeout=300
param_service.rawx.score_variation_bound=5
param_service.rawx.score_expr=((num stat.space)>=3) * root(2,((num stat.cpu)*(num
stat.space)))

param_service.sqlx.score_timeout=300
param_service.sqlx.score_variation_bound=5
param_service.sqlx.score_expr=((num stat.space)>=3) * root(2,((num stat.cpu)*(num
stat.space)))
```

Complete with the service allocation policy when a service is linked to a container. Please refer to the Administration Guide for customization.

```
# NONE|KEEP: no policy
# APPEND : add a reference
# # REPLACE: replace the reference if it exists
# (NONE|APPEND|REPLACE|KEEP)|REPLICA|DISTANCE|FILTER
param_option.service_update_policy.meta1=meta2=NONE|3|1;meta1=REPLACE;sqlx=APPEND|
3
```

Complete the section with the versioning policy directive.

```
# Default versioning policy (allow versioning and keep maximum 5 versions)
param_option.meta2_max_versions=5
```

Complete the section with the storage policy directive. This installation guide will implement simple duplication.

```
param_storage_conf=/usr/local/share/grid/storage_conf
# Default namespace policy
param_option.storage_policy=TWOCOPIES
```

Create the storage policy definition file

```
[STORAGE_POLICY]
# Format: NAMEOFPOLICY=STORAGECLASS:NAMEOFDATASECURITY:DATATREATMENT
# STORAGECLASS is not yet implemented
# DATATREATMENT is not yet implemented
# Default namespace policy is TWOCOPIES
# Another one is available, called NONE
```

```
TWOCOPIES=DUMMY:DUPONETWO:NONE
NONE=DUMMY:NONE:NONE

[DATA_SECURITY]
# Format: NAMEOFDATASECURITY=ALGORITHM:DISTANCE:NUMBEROFDUPLICATE
# ALGORITHM=DUP or NONE
# Datasecurity DUPONETWO requires each chunk to be stored twice, with a minimum
distance of 1
DUPONETWO=DUP:distance=1|nb_copy=2
# Datasecurity NONE requires each chunk to be stored only once
NONE=DUP:distance=0|nb_copy=1
```

Restart conscience. Find its name in gridinit

```
[root@singlenode gridinit.conf.d]# gridinit_cmd status
KEY                            STATUS      PID GROUP
gridagent                      UP         3571 common,common,singlenode,gridagent
TESTNS-singlenode-conscience-1 UP         3596
TESTNS,singlenode,singlenode,conscience
..
```

Then restart

```
[root@singlenode gridinit.conf.d]# gridinit_cmd stop TESTNS-singlenode-conscience-
1
DONE            TESTNS-singlenode-conscience-1  Success
[root@singlenode gridinit.conf.d]# gridinit_cmd start TESTNS-singlenode-
conscience-1
DONE            TESTNS-singlenode-conscience-1  Success
```

Display RC status and verify the default policy

```
[root@singlenode17 conf]# gridcluster $namespace

NAMESPACE INFORMATION

               Name : TESTNS
         Chunk size : 10485760 bytes
             Option : meta2_max_versions = 5
             Option : storage_policy = TWOCOPIES
             Option : container_max_size = 8192
             Option : writable_vns = TESTNS.Virtual2,TESTNS.Virtual1,TESTNS
             Option : vns_list = TESTNS.Virtual1,TESTNS.Virtual2
      Storage Policy : TWOCOPIES = DUMMY:DUPONETWO:NONE
       Data Security : DUPONETWO = DUP:1:2
(…)
```

3 Command line

3.1 Content management.

3.1.1 Create container

```
[root@localhost vagrant]# redc put $namespace/container-test
Container [container-test] created in namespace [TESTNS].
```

In order to force a versioning policy different from the namespace default (meta2_max_versions in conscience) use the -O ActivateVersioning option:

- -1 to create a non-versioned container
- 0 to create a versioned container with an unlimited number of versions
- N to create a versioned container with a limit of n versions

```
[root@localhost vagrant]# redc put  -O ActivateVersioning=3 $namespace/container-
test
Container [container-test] created in namespace [TESTNS].
```

In order to force a storage policy different from the namespace default , use the -O StoragePolicy option with a valid storage policy name

```
[root@localhost vagrant]# redc put -O StoragePolicy=TWOCOPIES
$namespace/container-test
Container [container-test] created in namespace [TESTNS].
```

3.1.2 Store content

```
[root@localhost ~]# redc put $namespace/container-test test.txt
Uploaded a new version of content [test.txt] in container [container-test]
```

Be sure that your rawx services are configured with the appropriate location variables. If not, this put operation will fail.

In a container without versioning, uploading the same content will generate an error

```
[root@localhost ~]# redc put $namespace/container-test test.txt

ERROR : Content [test.txt] alreay exists in container [container-test]
```

In a container with versioning, the new version of the content will be uploaded

```
[root@localhost ~]# redc put $namespace/container-test test.txt
Uploaded a new version of content [test.txt] in container [container-test]
```

3.1.3 List content

```
[root@localhost ~]# redc get $namespace/container-test
```

```
#Listing container=[container-test]
test.txt
#Total in [container-test] : 1 elements
```

In a container with versioning, use -O ShowInfo=on in order to display versions (and size)

```
[root@localhost ~]# redc get -O ShowInfo=on $namespace/container-test
#Listing container=[container-test]
9095 1 test.txt
9095 2 test.txt
#Total in [container-test] : 2 elements
```

First column is the content size. Second column is the version number.

3.1.4 Get content

```
[root@localhost ~]# redc get $namespace/container-test/test.txt /tmp/test.txt
```

By default, redc get will output content to stdout. Specify a filename (/tmp/test.txt) to store content in a file

In order to retrieve a specific version of a content in a versioning enabled container, specify the version in the request URL. By default, redc get will always retrieve the latest version.

```
[root@localhost ~]# redc get $namespace/container-test/test.txt?version=1
/tmp/test.txt
```

Please note that if the target file exists, redc get will not process the request

3.1.5 Delete content

```
[root@localhost ~]# redc delete $namespace/container-test/test.txt
```

In a versioning enabled container, redc delete will not physically delete content, rather mark them as DELETED for future deletion. Also note that redc delete will delete the latest version if not specified in the URL

If the last version of a content is (marked as) deleted, it will not appear in the basic list content command

```
[root@localhost ~]# redc get $namespace/container-test
#Listing container=[container-test]
#Total in [container-test] : 0 elements
```

Nevertheless, in a versioning enabled container, older version might still be available

```
[root@localhost ~]# redc get -O ShowInfo=on $namespace/container-test
#Listing container=[container-test]
9095 1 test.txt
9095 2 test.txt (deleted)
#Total in [container-test] : 2 elements
```

3.1.6 Destroy container

It is not possible to delete a non empty container. In a versioning enabled container, each version of each content must be (marked as) deleted, as described in the following sequence

```
[root@localhost ~]# redc get -O ShowInfo=on $namespace/container-test
#Listing container=[container-test]
9095 1 test.txt
9095 2 test.txt (deleted)
#Total in [container-test] : 2 elements

[root@localhost ~]# redc delete $namespace/container-test

ERROR : Container [container-test] not empty [TESTNS].

(Run action with -v option for technical details.)
[root@localhost ~]# redc get $namespace/container-test

[root@localhost ~]# redc delete $namespace/container-test/test.txt?version=1
[root@localhost ~]# redc get -O ShowInfo=on $namespace/container-test
#Listing container=[container-test]
9095 2 test.txt (deleted)
9095 1 test.txt (deleted)
#Total in [container-test] : 2 elements
[root@localhost ~]# redc delete $namespace/container-test
```

3.2 Service management

3.2.1 Display rawx location

```
[root@singlenode ~]# gridcluster -r $namespace | grep rawx
TESTNS|rawx|192.168.244.128:6008|score=91|tag.up=true|tag.loc=seclin.bureau119.pc1
|tag.vol=/DATA/TESTNS/singlenode17/vol02
TESTNS|rawx|192.168.244.128:6007|score=91|tag.up=true|tag.loc=seclin.bureau119.pc1
|tag.vol=/DATA/TESTNS/singlenode17/vol01
```

3.2.2 Assign a sqlx to a container

The database schema must be known before creating the database. In the following example, schema will be named test

```
 [root@singlenode17 conf]# redccdir link $namespace/container-test sqlx.test
Service [1|sqlx.test|192.168.244.128:6016|] linked to reference [container-test]
```

Verify that the container has been assigned the appropriate number of sqlx service, as defined in the conscience. Here we require 3 sqlx

```
[root@singlenode17 conf]# redcdir list $namespace/container-test sqlx.test
Reference [container-test], services [sqlx.test] linked:
   [1|sqlx.test|192.168.244.128:6016|]
   [1|sqlx.test|192.168.244.128:6014|]
   [1|sqlx.test|192.168.244.128:6017|]
```

3.2.3 Test a sqlx service

```
[root@singlenode17 conf]# redcsqlx $TESTNS/container-test test 'select * from
sqlite_master'
# query = "select * from sqlite_master"
# rows = 2
# status = 0
table|admin|admin|2|CREATE TABLE admin (k TEXT PRIMARY KEY NOT NULL, v BLOB
DEFAULT NULL)
index|sqlite_autoindex_admin_1|admin|3|(nil)
```

3.2.4 Unassign a SQLx from a container

```
[root@singlenode17 conf]# redcdir unlink $namespace/container-test sqlx.test
Services [sqlx.test] unlinked from reference [container-test]
```

4 Troubleshoot

4.1 Score

Service is considered unavailable if its score equals 0. It happens that score can be temporarily set to 0 in a local instance. To unlock score:

```
[root@singlenode gridinit.conf.d]# gridcluster --unlock-score -S
"$namespace|$service|$ip:$port"
```

Score can be forced to a specific value for debug purpose. Here score is forced to 100

```
[root@singlenode gridinit.conf.d]# gridcluster --set-score 100 -S
"$namespace|$service|$ip:$port"
```

4.2 Permissions

Be sure that each directory under /GRID and /DATA are set to admgrid:admgrid. If not, change permissions recursively.
Selinux can prevent rsyslog from working properly. Disable it.

4.3 Log files

Process	Log
Meta0	/GRID/$namespace/logs/meta0.{log,access}
Meta1	/GRID/$namespace/logs/meta1.{log,access}
Conscience	/GRID/$namespace/logs/conscience.{log,access}
Meta2	/GRID/$namespace/logs/meta2.{log,access}
Rawx	/GRID/$namespace/$singlenode/logs/volXX-httpd-errors.log
Sqlx	/GRID/$namespace/logs/sqlx.{log,access}

5 Multi node installation

This chapter will only provide advices on how to install Redcurrant on several nodes.

5.1 Advised distribution

Process	How many	Why	Comment
Conscience	2	Must be HA. Service is Stateless	Active/Passive failover with Pacemaker
Zookeeper	3 at least	Quorum is N/2+1	Zookeeper is self replicated
Gridinit	On every node		Except zookeeper only node
Gridagent	On every node		Except zookeeper only node
Meta0	3	Must be HA	Roundrubbin loadbalancing with LVS
Meta1	3	Must be HA	Meta1 is self replicated
Meta2	As much as possible	I/O intensive	Use high performance storage
Rawx	As much as possible	Space greedy	Use high capacity storage
Sqlx	As much as possible	I/O intensive	SQLx is self replicated