Percona Replication Manager

http://bit.ly/prm-mysql-slides

Sheeri Cabral

@sheeri

Mozilla Database Engineering

What is PRM?

- High availability
- Automatic failover
- Set of tools
- Created by Baron Schwartz and Yves Trudeau
    - End of 2011

Uses “standard” tools

- All open source
- Uses existing MySQL
- Pacemaker
  - Cluster management tool

Uses “standard” tools

- Corosync
  - Group communication system
  - Can also use heartbeat
- MySQL resource agent
  - For pacemaker
  - The only original component

Why these tools?

- Already standard
- Mature technologies
- OpenStack uses pacemaker + corosync
- Using many tools means lots to learn

Getting the Tools

- Corosync
  - corosync.github.io/corosync
- Pacemaker
  - clusterlabs.org
- Percona's Pacemaker MySQL agent
  - https://github.com/percona/percona-pacemaker-agents
  - No “releases”, just checkout from github

Pacemaker

- Lots of configuration options
  - This talk covers only some
- The heavy-lifter of the PRM solution
- Maintains a **c**luster **i**nformation **b**ase
  - CIB
  - Shared among nodes by corosync

Corosync

- Coordinates communications among nodes
- Uses an event queue
- Uses Totem protocol over UDP
- Default for communication in PRM
  - Can use others, like Heartbeat

MySQL Resource Agent

- bash script
- supports stop, start, monitor, promote, etc
- Provided by Percona
  - The only original part of PRM

Setting it up

- Many parts
- Not hard, but detailed
- https://github.com/percona/percona-pacemaker-agents/blob/master/doc/PRM-setup-guide.rst
  - Not updated since June 2012

Install Core Packages

- Use a package manager to install:
  - corosync
  - pacemaker
  - e.g. apt-get, yum install, etc.

Corosync Messaging

- Messaging requires authentication
- Via keys
- corosync-keygen
    - Make sure there's entropy on the machine!

Corosync Keys

- Like ssh keys

- Copy “authkey” to all the nodes
    - chown root:root
    - chmod 0400

Corosync Initial Configuration

- Usually /etc/corosync/corosync.conf
- Different sections { … }
- `compatibility`
  - `none` – compatible with corosync v1 only
  - `whitetank` – compatible with openais
  - Corosync was derived from ais

Corosync Initial Configuration

```
logging {
        fileline: off
        to_stderr: no
        to_logfile: yes
        to_syslog: yes
        logfile:
  /var/log/cluster/corosync.log
        debug: off
        timestamp: on
        logger_subsys {
                subsys: AMF
                debug: off
        }
}
```

Corosync Initial Configuration

```
totem {
        version: 2
        secauth: on
        threads: 0
        interface {
                ringnumber: 0
```

- ringnumber starts at 0
- different value for each interface
- for redundant ring protocol
- to determine which interface to use for which ring

Corosync Initial Configuration

```
totem {
        version: 2
        secauth: on
        threads: 0
        interface {
                ringnumber: 0
                bindnetaddr: 172.30.222.0
```

- Corosync uses network interface on **bindnetaddr** subnet
  - IPv4 only
    - easier configuration on many machines
  - For IPv6, specify full address
    - Different config on each machine

Corosync Initial Configuration

```
totem {
        version: 2
        secauth: on
        threads: 0
        interface {
                ringnumber: 0
                bindnetaddr: 172.30.222.0
                mcastaddr: 226.94.1.1
                mcastport: 5405
```

- multicast seems to be preferred vs. broadcast
- Avoid 224.x.x.x
    - Config multicast addr

Corosync Initial Configuration

```
totem {
        version: 2
        secauth: on
        threads: 0
        interface {
                ringnumber: 0
                bindnetaddr: 172.30.222.0
                mcastaddr: 226.94.1.1
                mcastport: 5405
                ttl: 1
        }
}
```

- TTL is # seconds; only valid when using multicast
- Can increase up to 255
- Usually only need to increase TTL in a routed network

Congratulations!

Now start corosync:

sudo service corosync start
verify with
corosync-objctl

Verify Corosync Works

```
[root@host-01 corosync]# corosync-objctl |
  grep members | grep ip

runtime.totem.pg.mrp.srp.members.-
  723640660.ip=r(0) ip(172.30.222.212)

runtime.totem.pg.mrp.srp.members.-
  1042407764.ip=r(0) ip(172.30.222.193)
```

Now, on to initializing Pacemaker!

Initialize Pacemaker

- Much more simple
- /etc/corosync/service.d/pacemaker
  - Name
  - Version

```
cat /etc/corosync/service.d/pacemaker:
service {
        name: pacemaker
        ver: 1
}
```

Congratulations!

Now start pacemaker:

sudo service pacemaker start
verify with
crm status

Verify Pacemaker Works

```
[root@host-02 corosync]# crm status
============
Last updated: Thu May 24 17:06:57 2012
Last change: Thu May 24 17:05:32 2012 via crmd
  on host-01
Stack: openais
Current DC: host-01 - partition with quorum
Version: 1.1.6-3.el6-
  a02c0f19a00c1eb2527ad38f146ebc0834814558
2 Nodes configured, 2 expected votes
0 Resources configured.
============

Online: [ host-01 host-02 ]
```

MySQL Configuration

- Configured for replication:
  - server-id
  - log-bin
  - log-slave-updates
- Do not start mysql automatically on server restart
  - Pacemaker will control mysql stop/start

MySQL Configuration

- For auto-failover, replication user should have:
  - PROCESS
  - RELOAD
  - SUPER
- MySQL should be running at this point

At this point, 3 of 4 tools are up:
MySQL
Corosync
Pacemaker

Remaining:
MySQL agent for Pacemaker

MySQL Agent for Pacemaker

- Code on Percona github
- https://github.com/percona/percona-pacemaker-agents
- No github releases
  - In resource-agents v3.9.3 or higher
- Or get from Github – wget or checkout
  - Install to different dir than resource-agents

Tell Pacemaker to Use Agent

- Set a primitive; includes
  - MySQL config file location
  - mysqld binary
  - MySQL process id
  - Socket
  - Replication and test user authentication
    - Test user needs SELECT on test table
  - And more...
- Use `crm configure edit` to load primitive

```
primitive p_mysql ocf:percona:mysql
 params config="/etc/my.cnf"
 pid="/var/lib/mysql/mysqld.pid"
 socket="/var/run/mysqld/mysqld.sock"
 replication_user="repl_user"
 replication_passwd="ola5P1ZMU"
 max_slave_lag="60"
```

- What an “out of date” slave is (max_slave_lag)
- What to do with an out of date slave
  - If evict_outdated_slaves true, stop slave

```
primitive p_mysql ocf:percona:mysql
 params config="/etc/my.cnf"
 pid="/var/lib/mysql/mysqld.pid"
 socket="/var/run/mysqld/mysqld.sock"
 replication_user="repl_user"
 replication_passwd="ola5P1ZMU"
 max_slave_lag="60"
 evict_outdated_slaves="false"
```

- What to do with an out of date slave
  - If evict_outdated_slaves true, stop slave

```
primitive p_mysql ocf:percona:mysql
 params config="/etc/my.cnf"
 pid="/var/lib/mysql/mysqld.pid"
 socket="/var/run/mysqld/mysqld.sock"
 replication_user="repl_user"
 replication_passwd="ola5P1ZMU"
 max_slave_lag="60"
 evict_outdated_slaves="false"
 binary="/usr/libexec/mysqld"
 test_user="test_user" test_passwd="2JcXCxKF”
 op monitor interval="5s" role="Master"
   OCF_CHECK_LEVEL="1"
 op monitor interval="2s" role="Slave"
   OCF_CHECK_LEVEL="1"
```

- Monitor intervals MUST be different
- Intervals define roles, not role name

```
primitive p_mysql ocf:percona:mysql
 params config="/etc/my.cnf"
 pid="/var/lib/mysql/mysqld.pid"
 socket="/var/run/mysqld/mysqld.sock"
 replication_user="repl_user"
 replication_passwd="ola5P1ZMU"
 max_slave_lag="60"
 evict_outdated_slaves="false"
 binary="/usr/libexec/mysqld"
 test_user="test_user" test_passwd="2JcXCxKF”
 op monitor interval="5s" role="Master"
   OCF_CHECK_LEVEL="1"
 op monitor interval="2s" role="Slave"
   OCF_CHECK_LEVEL="1"
 op start interval="0" timeout="60s"
 op stop interval="0" timeout="60s"
```

Pacemaker ms

- ms is for master/slave
- Start the p_mysql resources:

```
ms ms_MySQL p_mysql
 meta master-max="1" master-node-max="1"
 clone-max="2" clone-node-max="1" notify="true"
 globally-unique="false" target-role="Master"
 is-managed="true"
```

Pacemaker ms

- ms is for master/slave
- Start the p_mysql resources:

```
ms ms_MySQL p_mysql
  meta master-max="1" master-node-max="1"
  clone-max="2" clone-node-max="1" notify="true"
```

- clone-max is # of nodes (currently 2)
- When adding a node, increase clone-max
- notify=true must be set

Setup Virtual IPs for Failover

- Test user is used with rules
  - e.g. to be a reader, read from test table must succeed
- Use `crm configure edit` to add Virtual IPs

Setup Virtual IPs for Failover

- Write VIP 172.30.222.100

```
primitive writer_vip ocf:heartbeat:IPaddr2
        params ip="172.30.222.100" nic="eth1"
        op monitor interval="10s"
```

Setup Virtual IPs for Failover

- Read VIPs 172.30.222.101 and 172.30.222.102

```
primitive reader_vip_1 ocf:heartbeat:IPaddr2
        params ip="172.30.222.101" nic="eth1"
        op monitor interval="10s"
primitive reader_vip_2 ocf:heartbeat:IPaddr2
        params ip="172.30.222.102" nic="eth1"
        op monitor interval="10s"
```

Congratulations!

Now you have a MySQL cluster
managed with Percona Replication Manager
(Corosync, Pacemaker and a good agent)

Verify with `crm configure show`

```
[root@host-01 ~]# crm configure show
node host-01
 attributes p_mysql_mysql_master_IP="172.30.222.193"
node host-02
 attributes p_mysql_mysql_master_IP="172.30.222.212"
primitive p_mysql ocf:percona:mysql
       params config="/etc/my.cnf" pid="/var/lib/mysql/mysqld.pid"
  socket="/var/run/mysqld/mysqld.sock"
  replication_user="repl_user" replication_passwd="ola5P1ZMU"
  max_slave_lag="60" evict_outdated_slaves="false"
  binary="/usr/libexec/mysqld" test_user="test_user"
  test_passwd="2JcXCxKF" \
       op monitor interval="5s" role="Master"
  OCF_CHECK_LEVEL="1" \
       op monitor interval="2s" role="Slave" OCF_CHECK_LEVEL="1"
      op start interval="0" timeout="60s"
       op stop interval="0" timeout="60s"
```

```
primitive reader_vip_1 ocf:heartbeat:IPaddr2
        params ip="172.30.222.101" nic="eth1"
        op monitor interval="10s"
primitive reader_vip_2 ocf:heartbeat:IPaddr2
        params ip="172.30.222.102" nic="eth1"
        op monitor interval="10s"
primitive writer_vip ocf:heartbeat:IPaddr2
        params ip="172.30.222.100" nic="eth1"
        op monitor interval="10s"
```

```
ms ms_MySQL p_mysql
        meta master-max="1" master-node-max="1" clone-max="2"
    clone-node-max="1" notify="true" globally-unique="false"
    target-role="Master" is-managed="true"
location loc-No-reader-vip-2 reader_vip_2
        rule $id="rule-no-reader-vip-2" -inf: readable gt 0
location loc-no-reader-vip-1 reader_vip_1
        rule $id="rule-no-reader-vip-1" -inf: readable gt 0
colocation writer_vip_on_master inf: writer_vip ms_MySQL:Master
order ms_MySQL_promote_before_vip inf: ms_MySQL:promote
    writer_vip:start
property $id="cib-bootstrap-options"
        dc-version="1.1.6-3.el6-
    a02c0f19a00c1eb2527ad38f146ebc0834814558"
        cluster-infrastructure="openais"
        expected-quorum-votes="2"
        no-quorum-policy="ignore"
        stonith-enabled="false"
        last-lrm-refresh="1338928815"
property $id="mysql_replication"
        p_mysql_REPL_INFO="172.30.222.193|mysqld-bin.000002|106"
```

That's a crazy way to monitor, though....

crm status

crm_mon

(same as crm status, but refreshes like top)

```
root@host-01 ~]# crm status
============
Last updated: Tue Jun  5 17:09:01 2012
Last change: Tue Jun  5 16:43:08 2012 via cibadmin on host-01
Stack: openais
Current DC: host-01 - partition with quorum
Version: 1.1.6-3.el6-a02c0f19a00c1eb2527ad38f146ebc0834814558
2 Nodes configured, 2 expected votes
5 Resources configured.
============

Online: [ host-01 host-02 ]

Master/Slave Set: ms_MySQL [p_mysql]
    Masters: [ host-01 ]
    Slaves: [ host-02 ]
reader_vip_1   (ocf::heartbeat:IPaddr2):  Started host-01
reader_vip_2   (ocf::heartbeat:IPaddr2):  Started host-02
writer_vip     (ocf::heartbeat:IPaddr2):  Started host-01
```

Maintenance

- `crm node standby <hostname>`
  - Put a node into standby mode
- `crm node online <hostname>`
  - Put the node back online

PRM With Only Two Hosts

- Pacemaker quorum when one host behaves badly
- If only 2 hosts, no quorum

```
crm_attribute --attr-name no-quorum-policy
  --attr-value ignore
```

STONITH

- Shoot The Other Node In The Head
- If not using STONITH, tell Pacemaker:

```
crm_attribute --attr-name stonith-enabled
  --attr-value false
```

Questions/Comments/Feedback?

Slides:
http://bit.ly/prm-mysql-slides