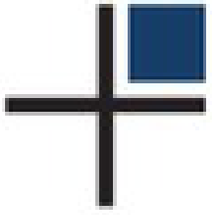
Alta disponibilità con repmgr 3.1

Gianni Ciolli

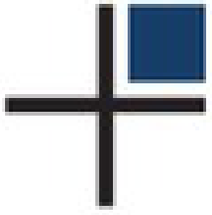
2ndQuadrant
Professional PostgreSQL

Milano
28-29 giugno 2016



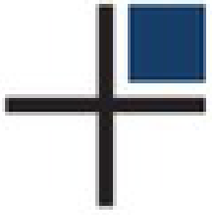
Argomenti

- Cluster PostgreSQL in Alta Disponibilità
- Integrazione
  - `repmgr` e PgBouncer
  - `repmgr` e Barman



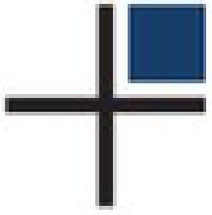
Programma

- Software
- Architettura
- Aspetti tecnici
- Affidabilità
- Manutenzione



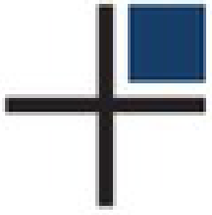
Panoramica su `repmgr`

- Clusterware per la replica con PostgreSQL
- Open source (GPL)
- Ultima versione: 3.1.3
  - Rilascio: 27 maggio 2016
- `http://www.repmgr.org/`



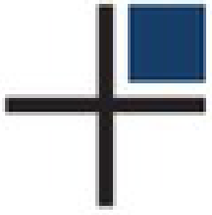
Caratteristiche di `repmgr`

- Monitoraggio
- Failover automatico
- Base Backup con rsync oppure `pg_basebackup`
- "Follow" senza restart
- Supporto per la replica in cascata
- Supporto per gli slot di replica
- Log degli eventi e comandi



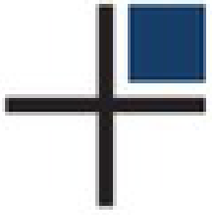
Panoramica su PgBouncer

- Pool di connessioni
- Open Source (BSD)
- Ultima versione: 1.7.2
  - Rilascio: 26 febbraio 2016
- `http://pgbouncer.github.io/`



Caratteristiche di PgBouncer

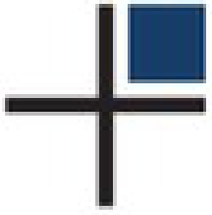
- Pool di connessioni
- Concentratore di connessioni
- Leggero
- Semplice
- Flessibile
- `PAUSE`, `RESUME`
  - riavvia ("bounce") il server senza interruzioni!



Sorpresa!

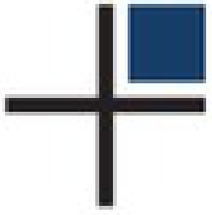
- Si parla anche di **Barman**
  - **Ba**ckup and **R**ecovery **Man**ager
- Perché?
- Niente Produzione Senza Backup!

Niente Produzione Senza Backup!



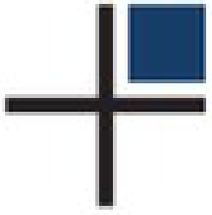
Più in generale...

- Il nodo primario cambierà regolarmente
  - Failover, Switchover, manutenzione...
- Anche il nodo primario richiederà manutenzione
  - Manutenzione automatica (cronjob)
  - Utile avere un alias che non cambia
- Dicendo "Barman" pensiamo a tutti questi processi



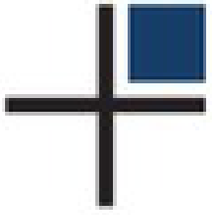
Barman

- Software per la Disaster Recovery
- Open source (GPL)
- Ultima versione: 1.6.1
  - Rilascio: 23 maggio 2016
- `http://www.pgbarman.org/`



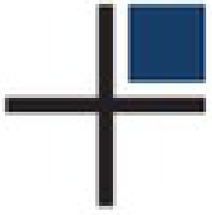
Caratteristiche di Barman

- Configurazione con file `.ini`
- Override
  - Per utente
  - Per server
- Policy di ritenzione
- Monitoraggio
- Backup incrementale
- Backup da Standby



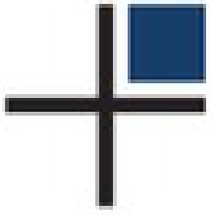
Il futuro di Barman

- Metodi di backup: `tar`, `pg_basebackup`
- Strategie di persistenza: `tar`, S3
- Compressione dei backup
- Backup criptati
- Geo-ridondanza
- Import/Export
- …



Architettura iniziale

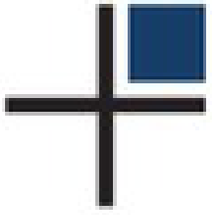
- Un server di database (PostgreSQL)
- Un server di backup (Barman)



Configurazione 1/3

`barman.conf` contiene:

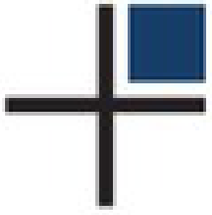
```
[vm95]
ssh_command = ssh vmp
conninfo = service=vmp
description = 9.5 cluster on VMs
```



Configurazione 2/3

`~barman/.pg_service.conf` contiene:

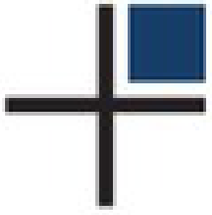
```
[vmp]
host=vm1
user=postgres
dbname=postgres
```



Configurazione 3/3

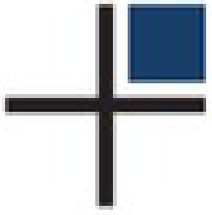
`~barman/.ssh/config` contiene:

```
Host vmp
     HostName vm1
     User postgres
     Port 22
```



Configurazione

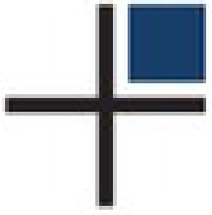
- Ciò che dipende dallo stato è collocato in userspace
- Decisione / buona prassi?



Arriva repmgr

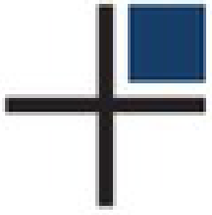
`repmgr.conf` contiene:

```
cluster=vm95
node=1
node_name=vm1
conninfo=host=vm1
```



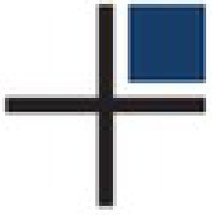
Come si usa `repmgr`

```
repmgr master register

repmgr standby clone ...
repmgr standby register
repmgr standby unregister
repmgr standby promote
repmgr standby follow
repmgr standby switchover

repmgr witness create

repmgr cluster show
```



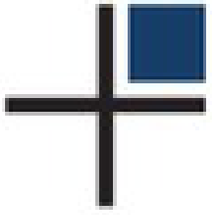
Il primo nodo

```
postgres@vm1:~$ repmgr master register

postgres@vm1:~$ repmgr cluster show
 Role      | Connection String
 * master  | host=vm1
```



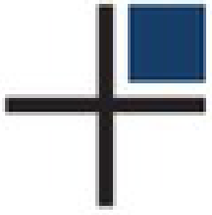
Un altro nodo

```
postgres@vm2:~$ repmgr standby clone -h vm1

postgres@vm2:~$ repmgr standby register

postgres@vm2:~$ repmgr cluster show
 Role      | Connection String
 * master  | host=vm1
   standby | host=vm2
```



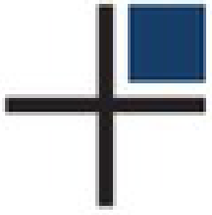
Arriva PgBouncer

- PgBouncer definisce dei *database*
- In PgBouncer
  - un database è una *stringa di connessione*
  - Sia locale che remota
- Il client si collega a PgBouncer
  - Connessione client
- PgBouncer inoltra la connessione al server
  - Connessione server
- Connessioni: client >> server



Configurazione: db PgBouncer

- Scelta: separare letture e scritture
  - Buona prassi



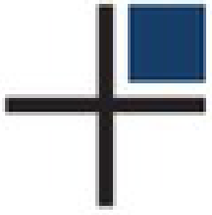
Configurazione: db PgBouncer

pgbouncer.ini su vm1:

```
[databases]
postgres_rw = host=vm1 dbname=postgres
postgres_ro = host=vm1 dbname=postgres
```

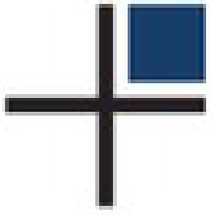
pgbouncer.ini su vm2:

```
[databases]
postgres_rw = host=vm1 dbname=postgres
postgres_ro = host=vm2 dbname=postgres
```



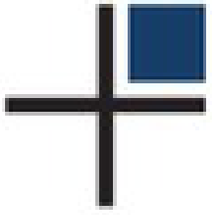
E Barman?

- I nodi standby sono cloni del primario
- Più copie di un server di database
- Barman ha bisogno di un nodo
- Barman può fare il backup da uno standby...
  - (usando `pgespresso`)...
  - ma noi useremo il nodo primario
    - Per semplicità
    - La simmetria è utile



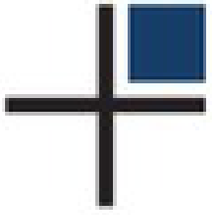
Automazione di `repmgr`

- Daemon `repmgrd`
  - Failover automatico
  - Monitoraggio
- Ulteriore automazione:
  - quando lo stato cambia:
    riconfigura ciò che va riconfigurato



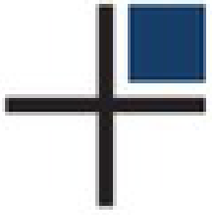
Lo stato del cluster?

- Uno *standby* può sostituire il *primario*
  - È il significato letterale di "stand by"
- Due diversi termini:
  - Switchover: previsto
  - Failover: imprevisto
- Differenza cruciale!
- Lo stato del cluster:
  - Lista dei nodi
  - Quale dei nodi è il primario



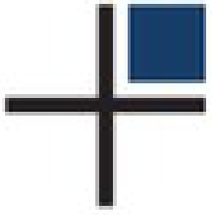
Nuovo primario (switchover)

```
postgres@vm1:~$ pg_ctl shutdown

postgres@vm2:~$ repmgr standby promote

postgres@vm3:~$ repmgr standby follow
postgres@vm4:~$ repmgr standby follow
...
postgres@vm100:~$ repmgr standby follow
```



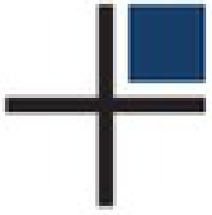
Nuova sintassi

- `repmgr standby switchover`
- Introdotto recentemente
- Ancora in via di perfezionamento



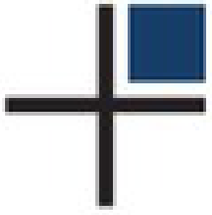
Failover automatico

```
failover=automatic
master_response_timeout=20
reconnect_attempts=3
reconnect_interval=5
promote_command=repmgr standby promote
follow_command=repmgr standby follow -W
```



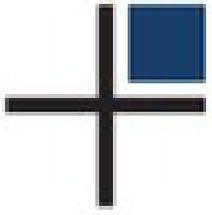
Scegliere il nuovo primario

- Si può definire la priorità di un nodo
  - Numero intero
- Criteri di scelta:
  - il nodo più avanti
  - a parità, quello con priorità maggiore
- I nodi con priorità zero o negativa sono ignorati
  - Non tutti gli standby sarebbero dei buoni master



Cambiamento di stato

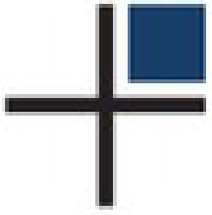
- Quando lo stato cambia
  - Occorre aggiornare la configurazione
- Tutto in *userspace*:
  - `~barman/.ssh/config`
  - `~barman/.pg_service.conf`
- Ehm, quasi...
- Non in userspace:
  - `/etc/pgbouncer/pgbouncer.ini`



Event Notification Command

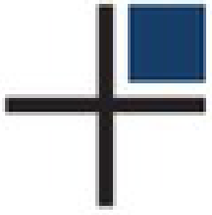
Aggiungere a `repmgr.conf` (solo due righe):

```
event_notification_command =
  repmgr-agent.sh repmgr.conf
  barman-server %n %e %s

event_notifications =
  master_register, standby_register,
  standby_promote
```



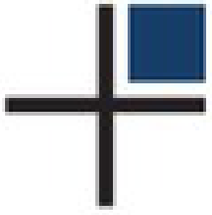
Event Notification Command

- Esegui uno script in occasione di eventi del cluster
  - Un po' come dei triggeri di tipo AFTER
- Solo gli eventi che *cambiano lo stato*



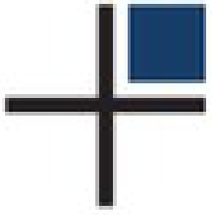
repmgr-agent.sh

- Script che aggiorna la configurazione
- Idempotente
- Prototipo, contributo a `repmgr`
- Legge lo stato del cluster
  - Da ogni nodo del cluster
- Riscrive:
  - `~barman/.ssh/config`
  - `~barman/.pg_service.conf`
  - `/etc/pgbouncer/pgbouncer.ini`



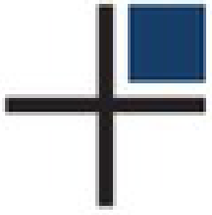
Il futuro di `repmgr`

- Integrazione with Barman
- Integrazione with PgBouncer
- Failover più robusto
  - Condivisione delle informazioni tra i vari nodi



E ora. . .

Domande?



E poi...

Grazie!

gianni@2ndquadrant.com

@GianniCiolli

Licenza