2ndQuadrant®
PostgreSQL

Construindo um Cluster de Alta Disponibilidade

William Ivanski
2ndQuadrant

Business Continuity

- **Atividade** realizada por uma organização para **garantir** que **funções críticas para o negócio** estejam **disponíveis** para clientes, fornecedores, reguladores e outras entidades que precisam ter acesso a essas funções.

- **Business Continuity**:
  - ***High Availability***
  - *Disaster Recovery*

Business Continuity

- **RPO (Recovery Point Objective)**
  - Período máximo em que pode haver **perda de dados** devido a um desastre
  - Basicamente é o **quanto de dados podem ser perdidos** (medido em tempo)

- **RTO (Recovery Time Objective)**
  - Período máximo em que o serviço pode ficar **indisponível** na ocorrência de um desastre

Business Continuity

- Basicamente queremos:
  - **RPO = 0**, ou seja, *zero data-loss*
  - **RTO = 0**, ou seja, *zero downtime* (utopia)

- **High Availability**:
  - Eventos planejados e não planejados
  - **RTO** e **RPO** o mais próximos possível de zero

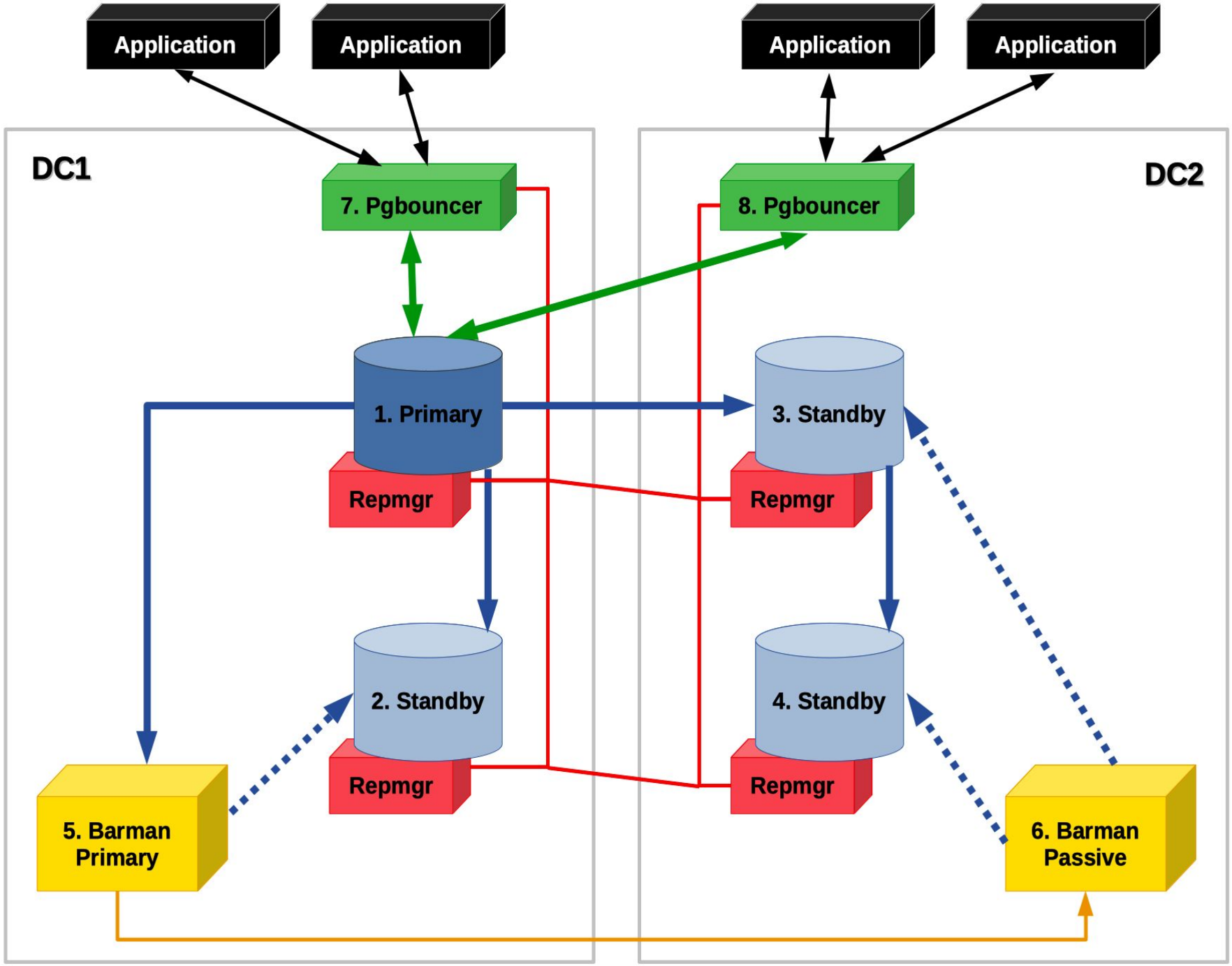
High Availability Cluster

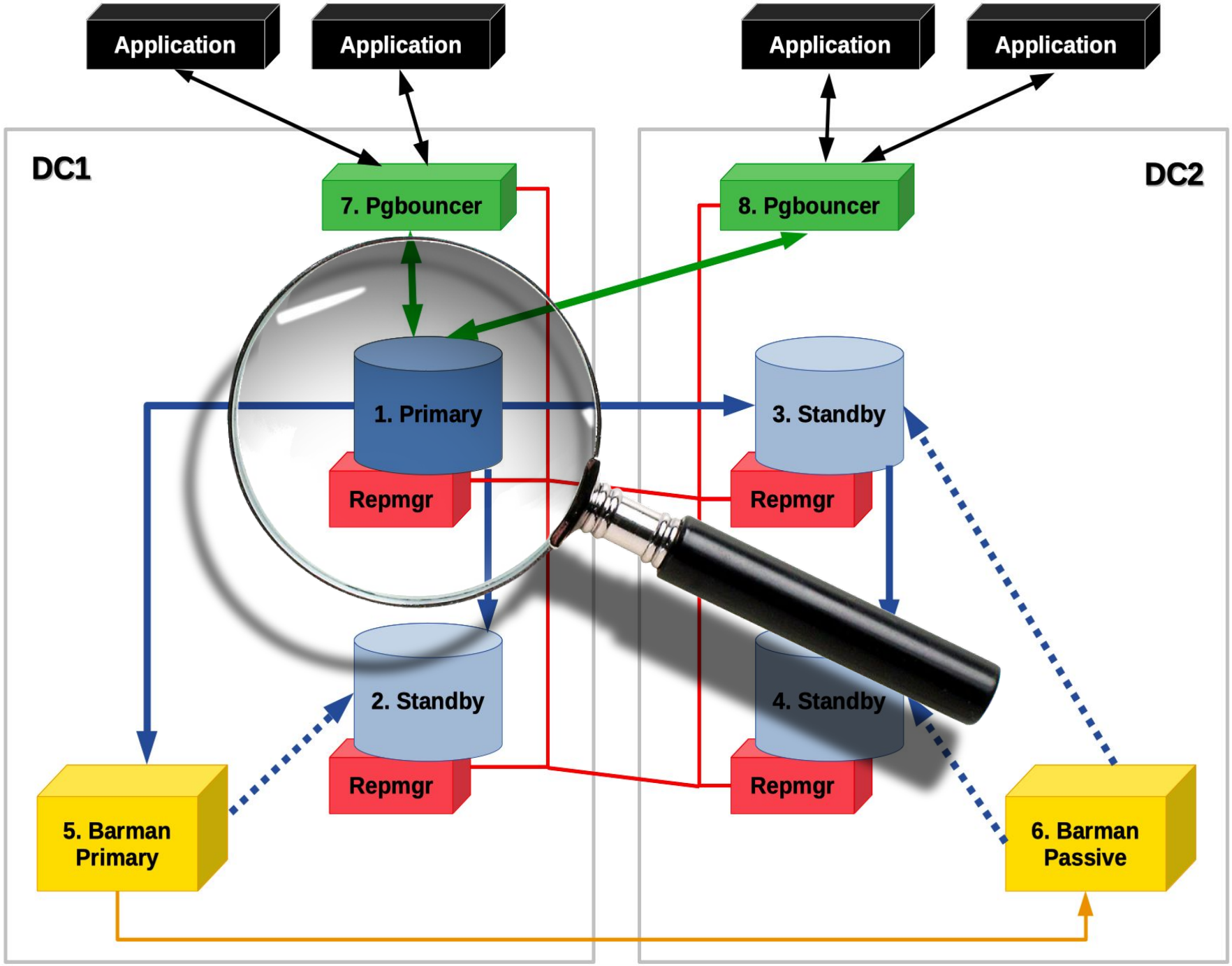
- Disponível em 2 *Data Centers*
- 8 máquinas Ubuntu 18.04

- 4 servidores de banco de dados
  - Software:
    - **PostgreSQL** 11.4
    - **Repmgr** 4.4
  - DC1: Primary e 1 standby
  - DC2: 2 standby

- 2 servidores de backup
  - **Barman** 2.8
  - DC1: Primary
  - DC2: Passive

- 2 connection poolers
  - **PgBouncer** 1.9
  - Um em cada DC

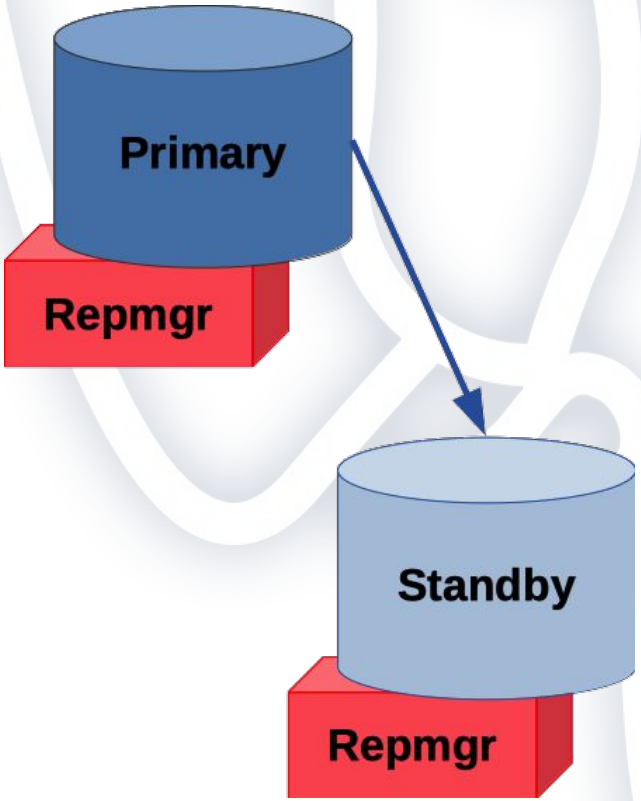

Application
Application
Application
Application
DC1
DC2
7. Pgbouncer
8. Pgbouncer
1. Primary
3. Standby
Repmgr
Repmgr
2. Standby
4. Standby
Repmgr
Repmgr
5. Barman Primary
6. Barman Passive


Application
Application
Application
Application
DC1
DC2
7. Pgbouncer
8. Pgbouncer
1. Primary
3. Standby
Repmgr
Repmgr
2. Standby
4. Standby
Repmgr
Repmgr
5. Barman Primary
6. Barman Passive

PostgreSQL + Repmgr

- Node1 é primário
  - `repmgr primary register`

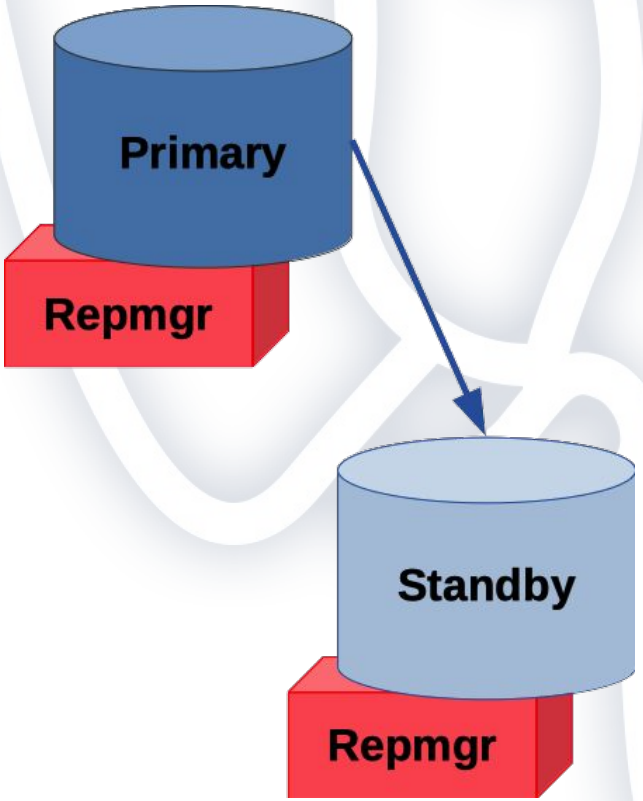
- Node2 e node3 replicam do node 1
- Node4 replica do node3 (cascading)
  - `repmgr standby clone \`
    `--upstream-node-id X`
  - `repmgr standby register`



PostgreSQL + Repmgr

- Arquivo **`postgresql.conf`**

```
hot_standby = on
wal_level = replica / logical
track_commit_timestamp = on
wal_log_hints = on
max_connections = 100
max_wal_senders = 10
max_worker_processes = 10
max_replication_slots = 10
shared_preload_libraries = 'repmgr'
```

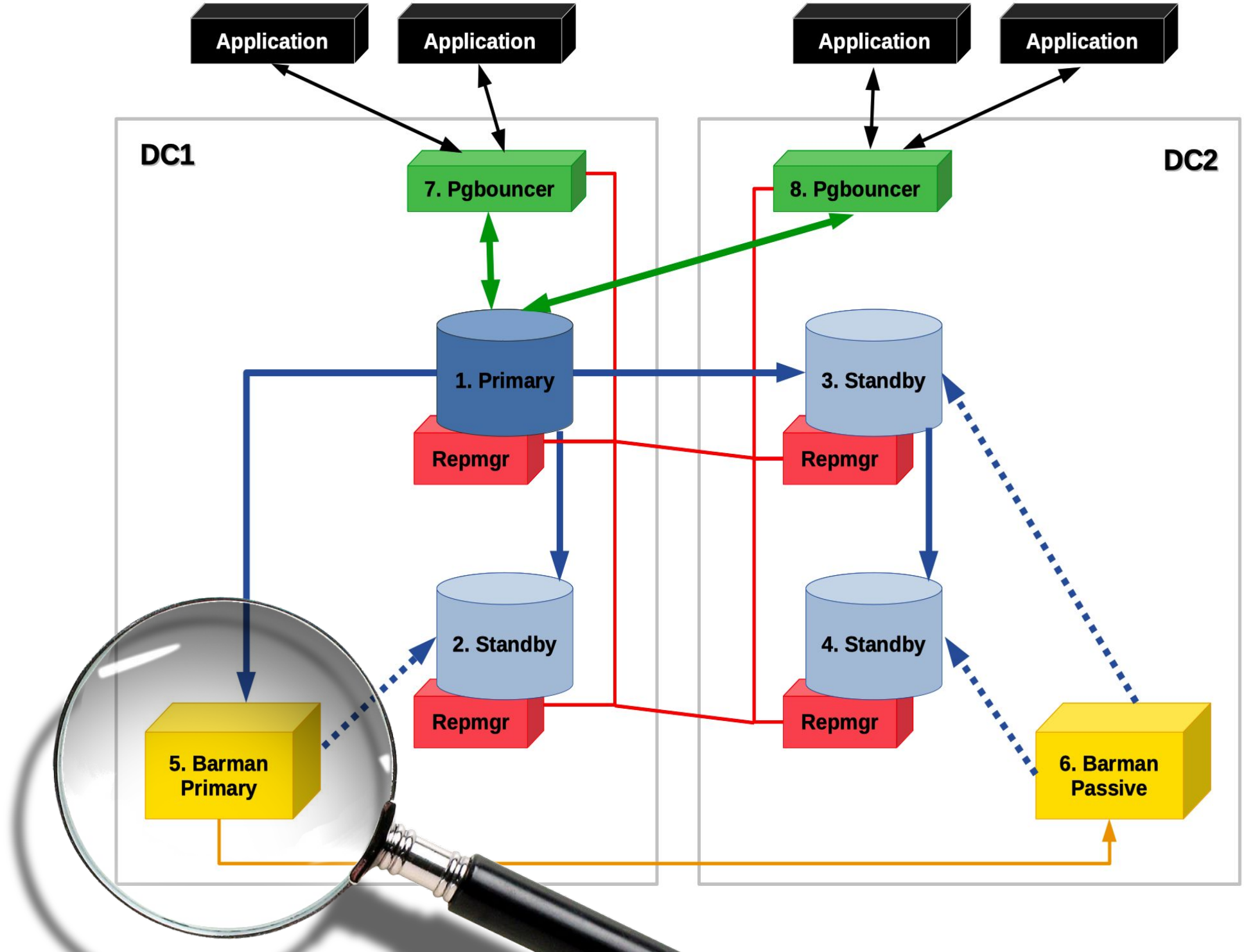


- Arquivo **`pg_hba.conf`**

PostgreSQL + Repmgr

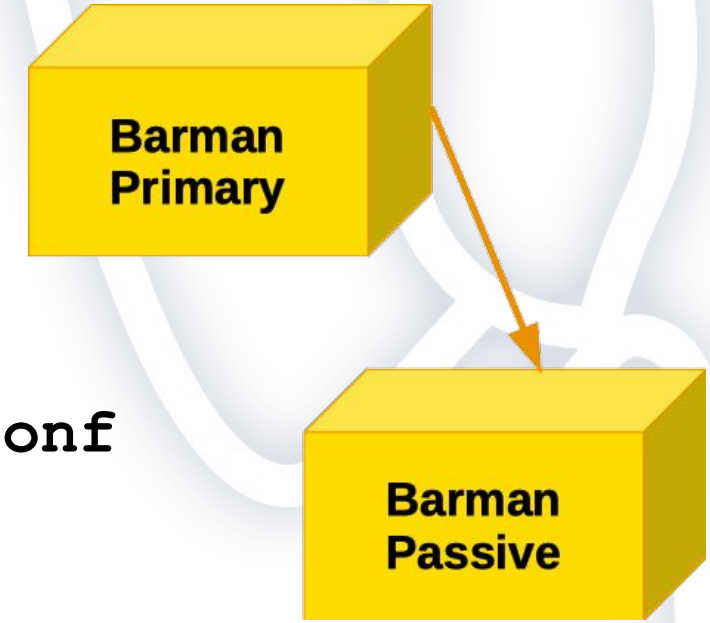
- Arquivo **`repmgr.conf`**

```
node_id=1
node_name=node1
conninfo='host=node1 application_name=node1 connect_timeout=2'
data_directory=/var/lib/postgresql/11/main/
location='dc1'
use_replication_slots=no
failover=automatic
promote_command='/etc/repmgr/11/promote_command.sh'
follow_command='/etc/repmgr/11/follow_command.sh'
restore_command='barman-wal-restore -U barman -z -p 8 node5 pg %f %p'
service_start_command= sudo systemctl start postgresql
service_stop_command= sudo systemctl stop postgresql
service_restart_command= sudo systemctl restart postgresql
service_reload_command= sudo systemctl reload postgresql
```


Application
Application
Application
Application
DC1
DC2
7. Pgbouncer
8. Pgbouncer
1. Primary
3. Standby
Repmgr
Repmgr
2. Standby
4. Standby
Repmgr
Repmgr
5. Barman Primary
6. Barman Passive

Barman

- Geo-redundancy (Barman 2.6+)
  - Node5 é primário (DC1)
  - Node6 é passivo (DC2)
- Standby servers podem buscar WAL do servidor Barman no mesmo DC
  - `restore_command` no `recovery.conf`
- Pode ser usado para reconstruir standby
- Disaster Recovery



Barman

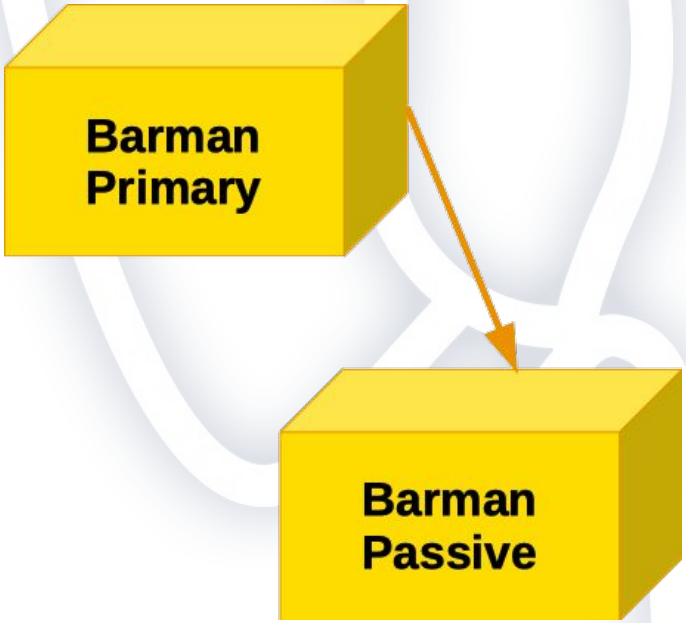
- Barman primário

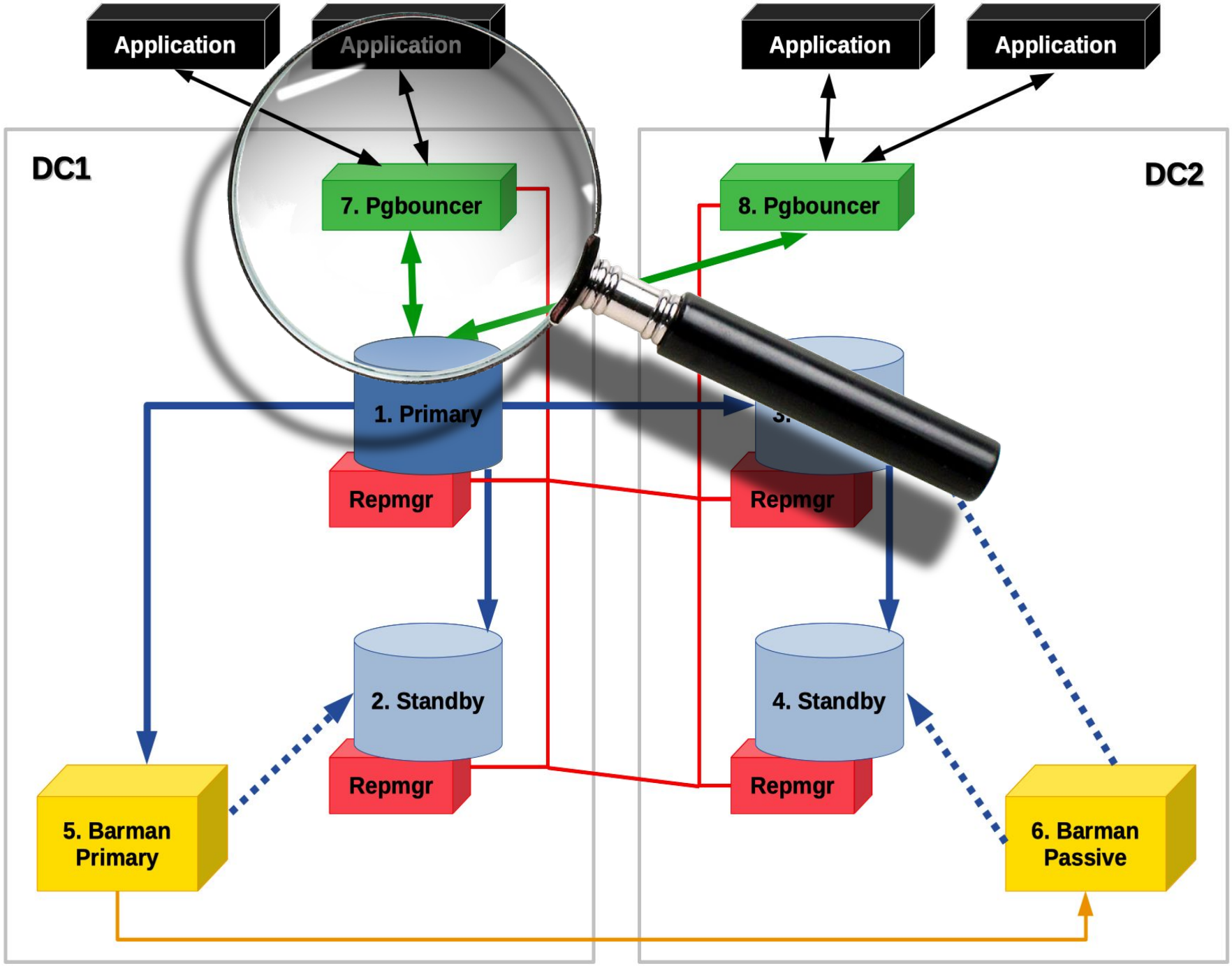
```
[pg]
description = "Backups from PostgreSQL database"
ssh_command = "ssh -q postgres@node1"
conninfo = "host=node1 dbname=postgres user=barman "
streaming_conninfo = "host=node1 dbname=postgres user=streaming_barman "
streaming_archiver = on
slot_name = "backup_node5"
active = true
backup_method = rsync
reuse_backup = link
parallel_jobs = 4
minimum_redundancy = 3
retention_policy = RECOVERY WINDOW OF 4 WEEKS
last_backup_maximum_age = 1 WEEK
recovery_options = 'get-wal'
```

Barman

- Barman passivo

```
[passive]
description = "Passive backups"
primary_ssh_command = ssh barman@node5
```



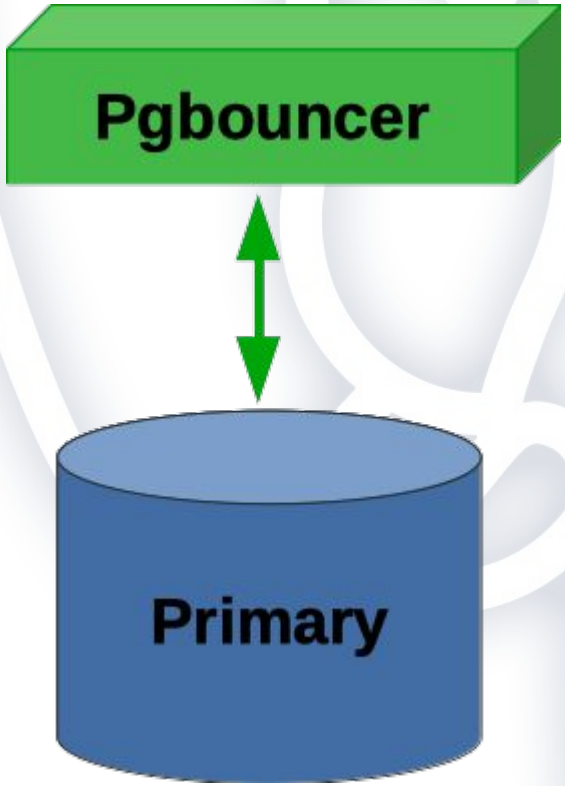

Application
Application
Application
Application
DC1
DC2
7. Pgbouncer
8. Pgbouncer
1. Primary
Repmgr
Repmgr
2. Standby
4. Standby
Repmgr
Repmgr
5. Barman Primary
6. Barman Passive

Pgbouncer

- Node7 e node8 são Pgbouncer
  - Um em cada DC
- Aplicação se conecta no Pgbouncer, usando multi-host connection string:

```
host=node7,node8 port=6432 dbname=mydb
```

- Ambos apontando pro primary **atual**

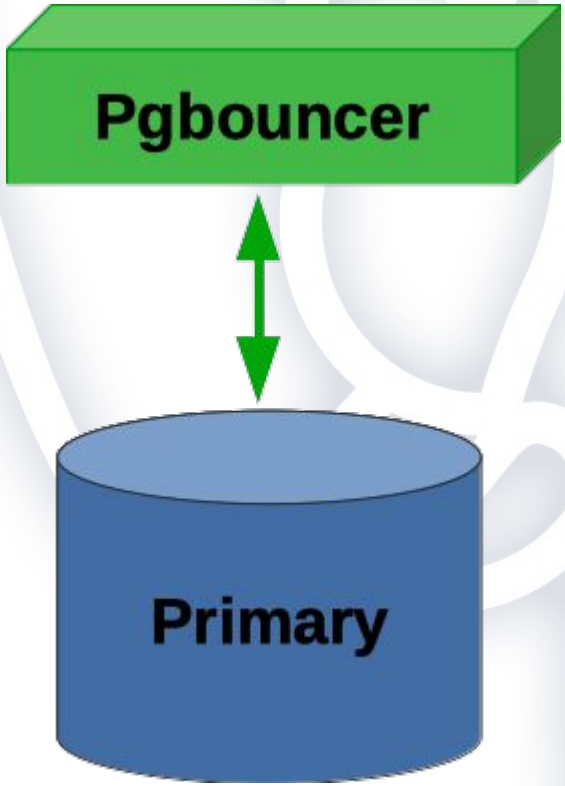


Pgbouncer

- Arquivo **`pgbouncer.ini`**

```
[databases]
* = host=node1 port=5432 auth_user=pgbouncer

[pgbouncer]
listen_addr = *
listen_port = 6432
pool_mode = session
auth_type = md5
auth_file = /etc/pgbouncer/userlist.txt
auth_user = pgbouncer
auth_query = SELECT * FROM pgbouncer.get_auth($1)
admin_users = pgbouncer
```

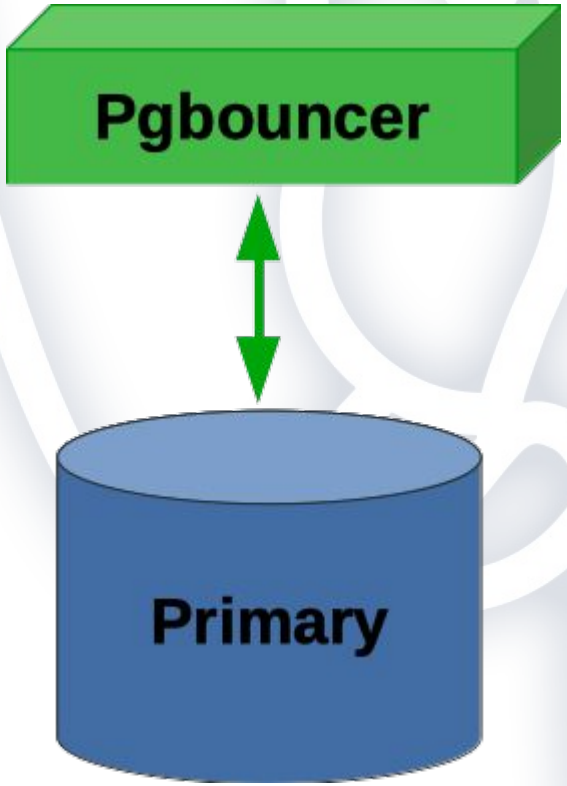


`https://www.2ndquadrant.com/en/blog/pg-phriday-securing-pgbouncer/`

Pgbouncer

- Arquivo **pgbouncer.ini.node1**
  - **host=node1**
- .....
- Arquivo **pgbouncer.ini.node4**
  - **host=node4**

- **ln -s pgbouncer.ini.node1 **
  pgbouncer.ini



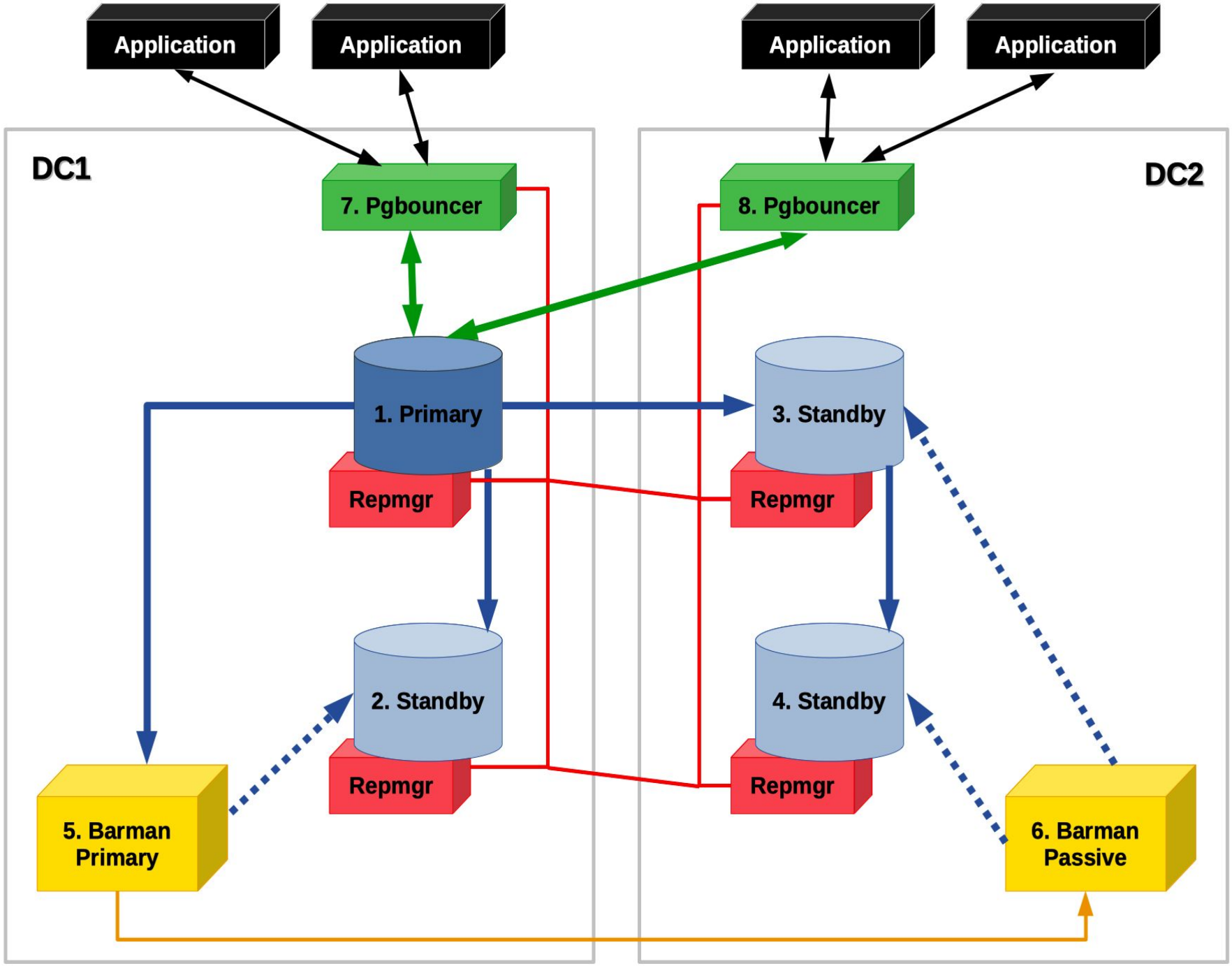
- Arquivo **follow_command.sh**
  - **repmgr standby follow --upstream-node-id=%n**

Pgbouncer

- Arquivo **`promote_command.sh`**

```
ssh node7 touch promotion.begin.$(hostname).$(date +%Y%m%d%H%M%S)
psql -h node7 -p 6432 -U pgbouncer -d pgbouncer -c 'PAUSE'
repmgr standby promote
ssh node7 rm /etc/pgbouncer/pgbouncer.ini
ssh node7 ln -s /etc/pgbouncer/pgbouncer.ini.$(hostname) /etc/pgbouncer.ini
psql -h node7 -p 6432 -U pgbouncer -d pgbouncer -c 'RELOAD'
psql -h node7 -p 6432 -U pgbouncer -d pgbouncer -c 'RESUME'
ssh node7 touch promotion.end.$(hostname).$(date +%Y%m%d%H%M%S)
```

Idem para node8


Application
Application
Application
Application
DC1
DC2
7. Pgbouncer
8. Pgbouncer
1. Primary
3. Standby
Repmgr
Repmgr
2. Standby
4. Standby
Repmgr
Repmgr
5. Barman Primary
6. Barman Passive

HA Procedures

- Planejados:
  - Switchover
  - Upgrade
- Não Planejados:
  - Falha no Primary
  - Falha no Standby
  - Falha no Barman
  - Falha no Pgbouncer
  - Falha no Datacenter

Switchover (manual)

1. **`CHECKPOINT`** no primary
2. Executar **`switchover_command.sh`** no standby

```
psql -h node7 -p 6432 -U pgbouncer -d pgbouncer -c 'PAUSE'
repmgr standby switchover --siblings-follow
ssh node7 rm /etc/pgbouncer/pgbouncer.ini
ssh node7 ln -s /etc/pgbouncer/pgbouncer.ini.$(hostname) /etc/pgbouncer.ini
psql -h node7 -p 6432 -U pgbouncer -d pgbouncer -c 'RELOAD'
psql -h node7 -p 6432 -U pgbouncer -d pgbouncer -c 'RESUME'
```

Idem para node8

3. Redirecionar Barman para o novo primary

Upgrade (manual)

- **Binary Compatible Upgrade**
  - Só pode ser feito num standby
  - Procedimento:
    - Parar o PostgreSQL
    - Fazer o upgrade
    - Iniciar o PostgreSQL
  - Cascading standby: node4 primeiro, depois o node3
- **Binary Incompatible Upgrade**
  - Necessário criar um cluster separado
  - Usar logical replication para transferir de um DC pro outro

Falha no Primary

- **Timeout: 60 segundos**
  - `reconnect_attempts=6 * reconnect_interval=10`
- **Automaticamente**
  - Pgbouncer é pausado
  - Standby local é promovido e se torna primary
  - Demais standbys "seguem" o novo primary
  - Pgbouncer é redirecionado para o novo primary
  - Pgbouncer é despausado
- **Manualmente**
  - Redirecionar barman para o novo primary
  - Avaliar primary que falhou
  - Re-incluir no cluster o primary que falhou como standby

Falha no Standby (manual)

- **Falha temporária: replication lag**
  - Arquivo WAL requisitado pelo standby não está mais disponível no primary
  - Standby se desconecta do primary
  - Standby passa a usar o **`restore_command`** para requisitar arquivos WAL que já foram arquivados pelo Barman
  - Eventualmente o standby se re-conecta ao primary
- **Falha permanente**
  - Construir um novo standby
  - Cascading standby

Falha no Barman (manual)

- **Falha temporária**
  - Replication slot garante que arquivos WAL serão mantidos
- **Falha permanente do Barman Primary**
  - Remover o replication slot
  - Configurar o Barman Passivo como Barman Primary
    - DC diferente do PostgreSQL primary
  - Configurar um novo Barman Passive no mesmo DC que o PostgreSQL primary
  - Esperar até que o novo Barman Passive sincronize
  - Inverter os papéis entre os dois Barman
- **Falha permanente do Barman Passive**

Falha no Pgbouncer (manual)

- Cada Pgbouncer node é uma entrada separada para o PostgreSQL primary
- Se um deles falhar, aplicação vai se conectar com o outro automaticamente
  - LibPQ multi-host connection string
  - DNS
  - Virtual IP

Falha no Datacenter (manual)

- **Falha no Datacenter onde não está o PostgreSQL primary**
  - Temporário: replicação vai continuar normalmente depois que o Datacenter estiver disponível
  - Permanente: necessário construir um novo Datacenter
- **Falha no Datacenter onde está o PostgreSQL primary**
  - Promover o node3 para se tornar o novo PostgreSQL primary
  - Configurar o Barman Passive para ser um Barman Primary, direcionado ao node3
  - Reconstruir o Datacenter que falhou

Perguntas?

2ndQuadrant PostgreSQL

Muito obrigado!

William Ivanski
william.ivanski@2ndquadrant.com