HAUTE DISPONIBILITÉ AVEC POSTGRESQL ET PATRONI

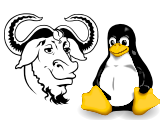
Jean-Christophe Arnu

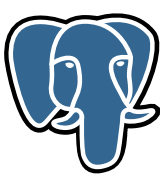
Capitôle Du Libre 2024

ENCHANTÉ, JE ME PRÉSENTE !

JEAN-CHRISTOPHE ARNU

- Création d'une association autour des logiciels libres à Toulouse 1996
- PostgreSQL depuis 1998
- Un des membres fondateurs de PostgreSQLFr
- Organisateur du premier PGDay.fr à Toulouse en 2008
- Consultant PostgreSQL chez LOXODATA





QUOI ?

- ~~Mais au fait c'est quoi la haute-dispo ?~~
- Quels problèmes la haute dispo cherche à résoudre ?

Spoiler-alert : c'est barbant, c'est de la théorie

- Ok PostgreSQL? Il le gère comment ?
- **PATRONI** et la haute disponibilité.

LA HAUTE DISPONIBILITÉ ? KEZAKO ?

Fonctionne tout le temps, de manière optimale:

- On redonde (Alim, réseaux, serveurs ...)
- On distribue (plusieurs zones (AZ), plusieurs DC)

Les indicateurs:

- On ne veut aucun « downtime » : RTO=0
- On ne veut perdre aucune donnée : RPO=0
- ... ?

Que veut-on au juste ?

LES MYTHES

« Je veux que ça fonctionne de manière performante, sans interruption de service, sans perte de donnée, quoi qu'il arrive »

*« Je veux que ça fonctionne 99.999% du temps »**

Nous sommes dans une conférence technique, nous allons voir pourquoi ce n'est pas possible.

VOYONS VOIR (SUR UN AN)...

99% :3 JOURS 14H 24 MINUTES

```
 select justify_interval('1 year'::interval*0.01)
 justify_interval
------------------
 3 days 14:24:00
```

99.9% : 8H 38 MINUTES

```
select justify_interval('1 year'::interval*0.001)
 justify_interval
------------------
 08:38:24
```

99.99%: ~ 52 MINUTES

```
select justify_interval('1 year'::interval*0.0001)
 justify_interval
------------------
 00:51:50.4
```

99.999% : 5 MINUTES

```
select justify_interval('1 year'::interval*0.00001)
 justify_interval
------------------
 00:05:11.04
```

PLEIN DE 9 ?

* MAIS NE FAIT-ON JAMAIS DE MAINTENANCE DE L'APPLI, DES BASES (UPGRADES), L'OS, ... ?

POURQUOI NE PAS APPLIQUER LES MÊMES RÈGLES À LA BASE ET AUX APPLIS ?

LE MYTHE DU RPO=0

« Je ne veux perdre aucune donnée »

LE MYTHE DU RPO=0

D'accord, mais en cas de désastre, on perd :

- Le cache système (fsync)
- Le cache disque (batteries ?)
- L'hyperviseur ou le CP
- Le cache réseau (pour le stockage réseau)

LE MYTHE DU RPO=0

QUE PERDONS-NOUS RÉELLEMENT ?

EN BASE DE DONNÉES : DES TRANSACTIONS

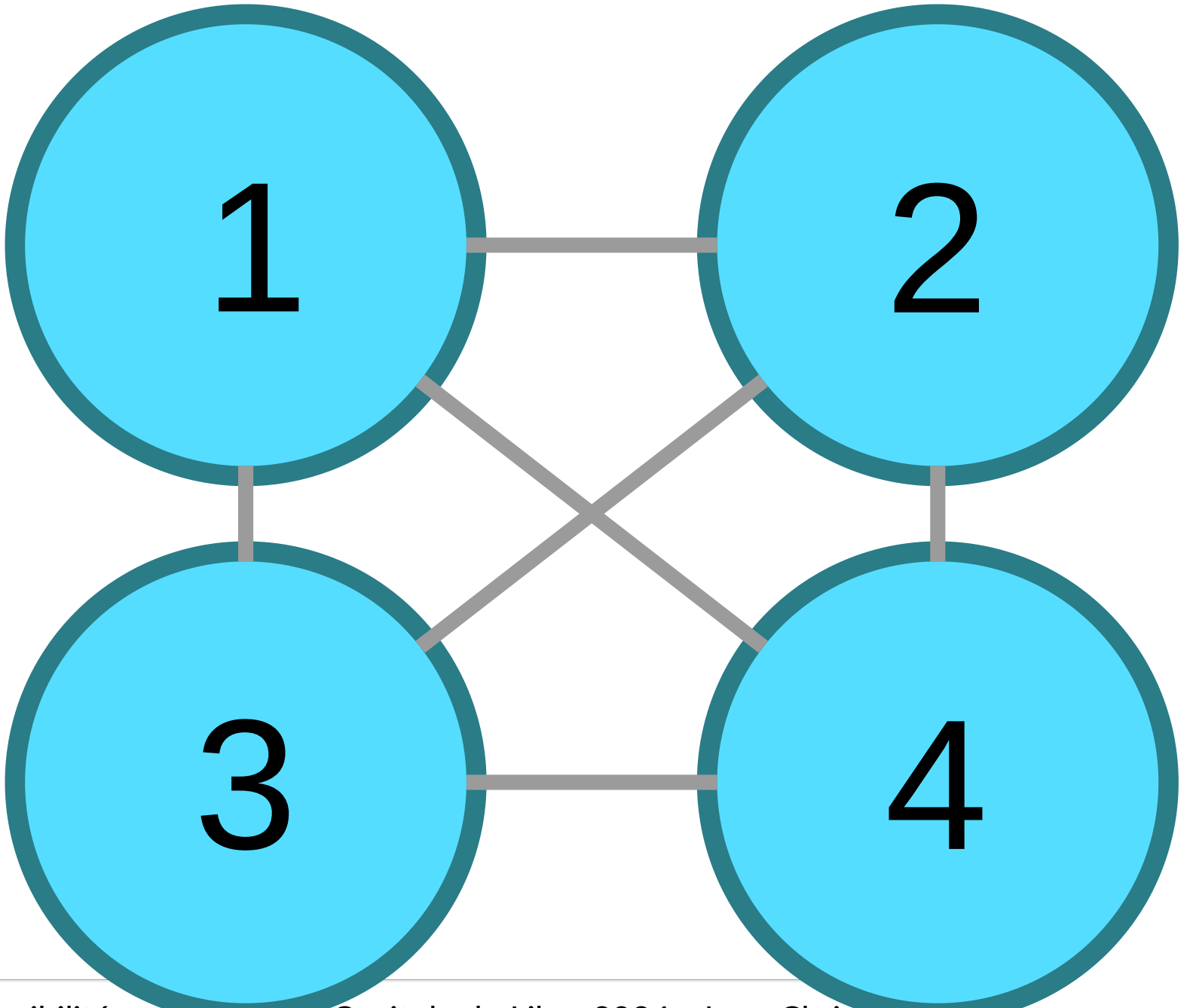
UN SYSTÈME DISTRIBUÉ

ON VA REDONDER LE SYSTÈME SUR LE RÉSEAU, MAIS

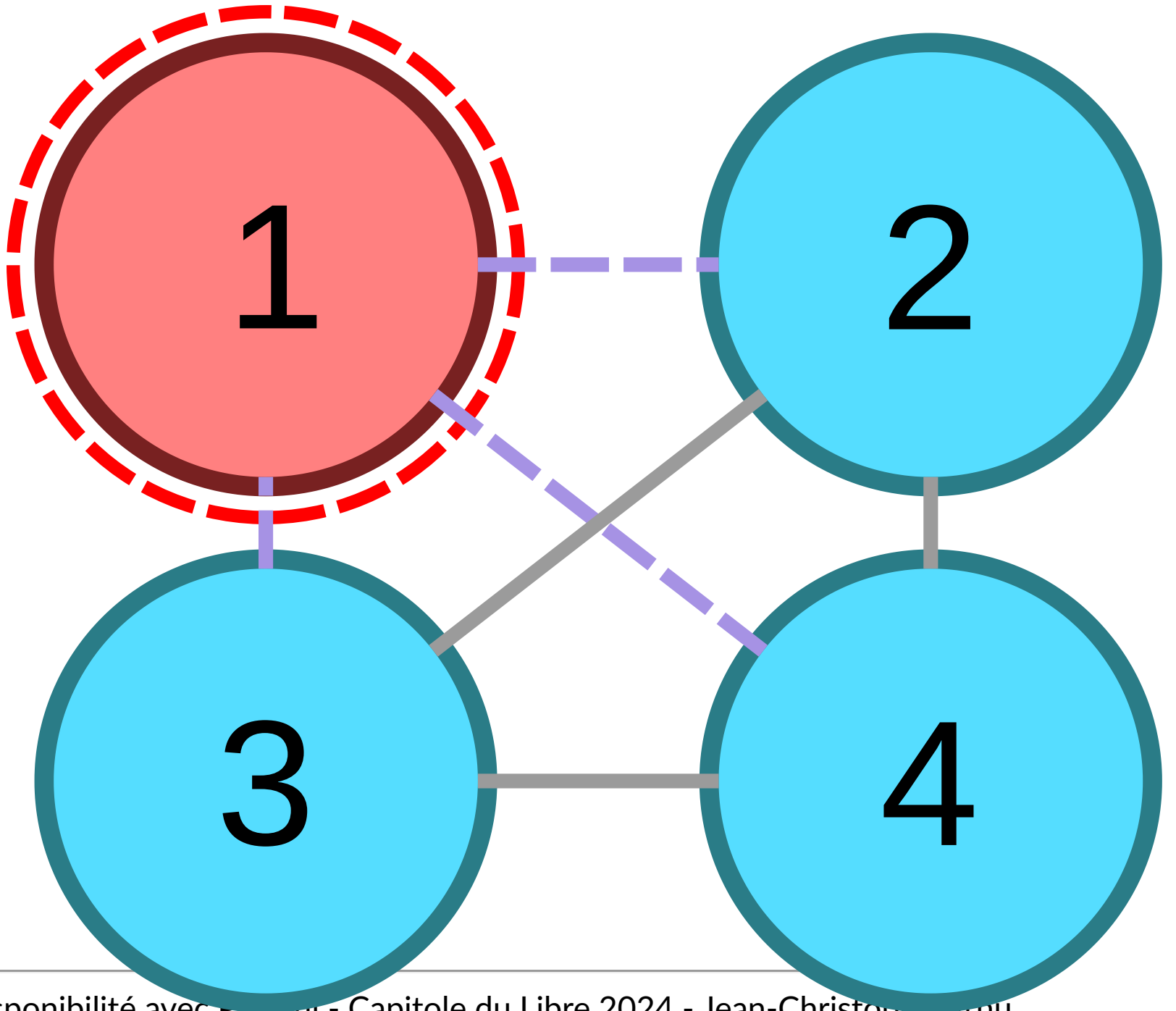
- On veut garantir la redondance, mais un nœud peut:
  - tomber en panne
  - se retrouver isolé
  - se retrouver indisponible

On peut se retrouver en situation de partitionnement réseau

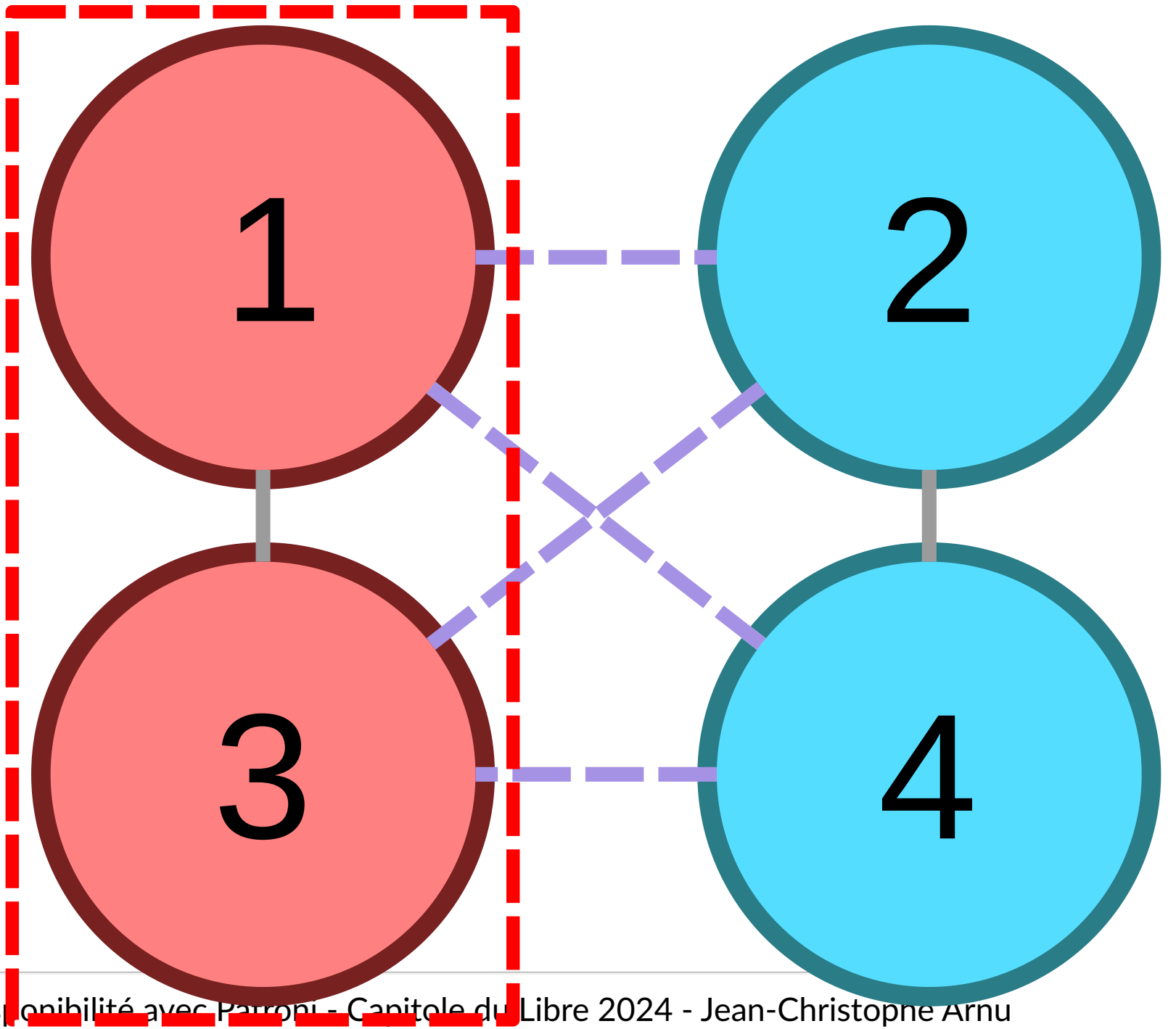
LE PARTITIONNEMENT RÉSEAU



LE PARTITIONNEMENT RÉSEAU



LE PARTITIONNEMENT RÉSEAU

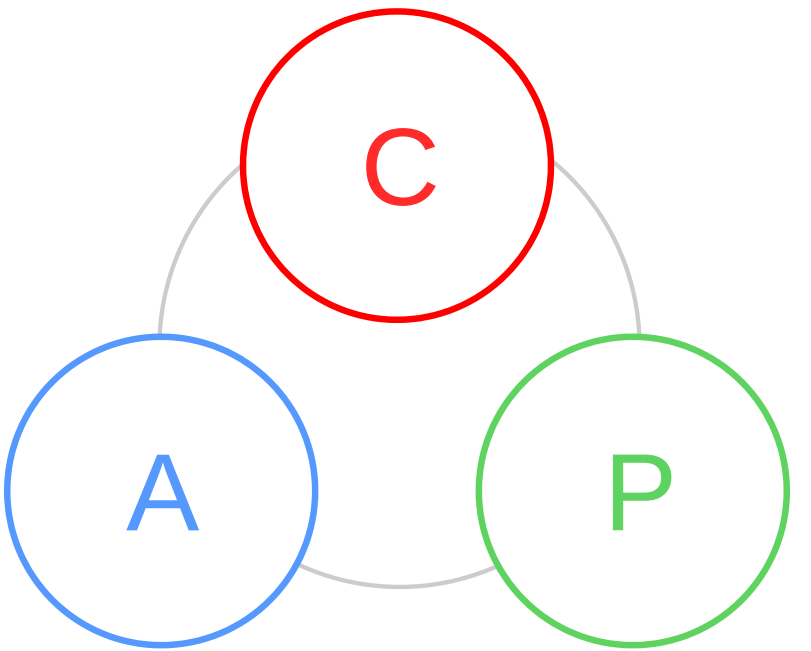


CREUSONS

THÉORÈME CAP

Eric Brewer

- C: Consistency : les données sont cohérentes entre tous les nœuds
- A: Availability : le service est disponible
- P: Partitionning : le service est résistant au partitionnement réseau



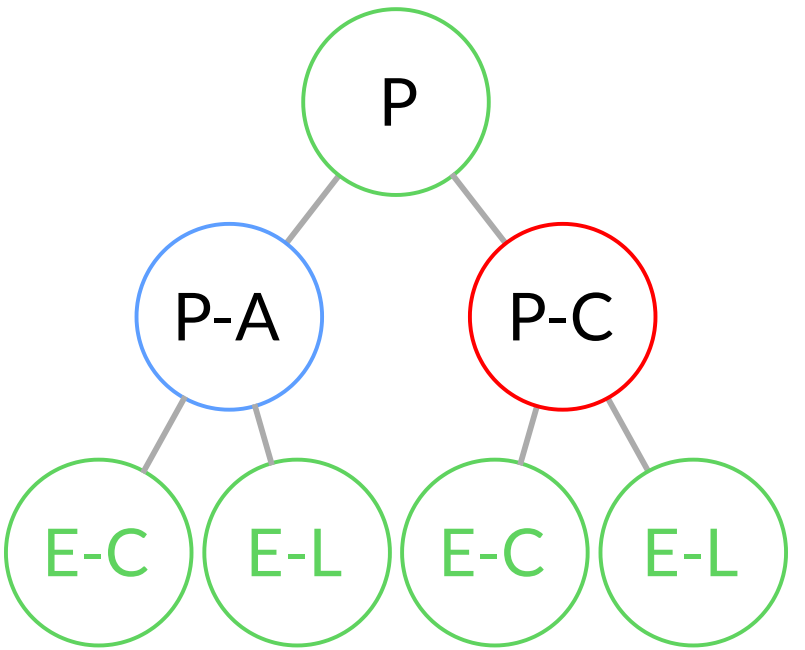
ON NE PEUT EN AVOIR QUE DEUX PROPRIÉTÉS EN MÊME TEMPS.

IL FAUT DONC FAIRE DES CHOIX!

THÉORÈME PACELC

Daniel Abadi

- Bascule:
  - Manuelle (P-C)
  - Automatique (P-A)
- Réplication:
  - Asynchrone (E-L)
  - Synchrone (E-C)



ET POSTGRESQL DANS TOUT ÇA ?

RÉPLICATION PAR FLUX NATIVE

- Un standby se connecte à un primaire
- Le standby reçoit le flux temps réel des modifications (WAL)
- Peut être asynchrone PC/EL ou synchrone PC/EC

QUELQUES MOTS SUR LA RÉPLICATION SYNCHRONE

- Permet d'implémenter RPO=0
- Mais tout `COMMIT` est bloqué tant qu'au moins un standby n'a pas acquitté
  - Latence
  - Blocage en cas d'indispo d'un standby

SAUVEGARDE

- Native (pg_basebackup + archive command) [NIH]
- Restauration sans impact sur le primaire:
  - Barman
  - PgBackrest
- Archivage des WALs

FONCTIONS ?

POURQUOI PAS DE HAUTE DISPONIBILITÉ DANS POSTGRESQL ?

- PostgreSQL fournit une **infrastructure** pour l'implanter
  - Gestion de la réplication (lecture, contrôle)
  - Gestion de la promotion/« demotion » en primaire
- Permet de résoudre une grande variété de cas d'usage
- Sans contraindre l'utilisateur à une seule vision

CONDITIONS DE LA HAUTE DISPONIBILITÉ

(AVEC POSTGRESQL)

LES DCS

Distributed Consensus Store

- Consensus entre les nœuds du DCS
  - Un leader, N followers
  - Si k nœuds dans le cluster ⇒ quorum = (k/2)+1
- Configuration du cluster PostgreSQL décentralisée
- Élection du primaire
  - Choix du primaire (le plus en avance)
  - Nouvelle topologie
  - Lock config
  - Primaire → Update config

EXEMPLE DE DCS

- zookeeper (praxos)
- etcd (raft)
- consul (raft)
- K8s API (raft)

ACCÉDER AUX NŒUDS

Entrypoints

- VIP (IP virtuelle)
- HAProxy
- PgBouncer
- N'oublions pas les drivers (libpq + java + DotNet)
- 12 factors directement auprès du DCS

COMMENT RECONSTRUIRE DES RÉPLICAS ?

Avec des outils de sauvegarde

- Basique et natif:
  - pg_basebackup
- Plus évolués:
  - Barman
  - PgBackRest

PATRONI

INITIÉ PAR ZALANDO (INTÉGRATION DOCKER : SPILO)

LICENCE MIT

PATRONI EST UN TEMPLATE

S'interface avec

- PostgreSQL
- Un DCS
- Un outil de backup
- Un seul fichier de configuration

FONCTIONNALITÉS

Pour le cluster

- Suivi de la réplication synchrone ou asynchrone (configuration)
- Réplication simple ou en cascade
- Failover
- Switchover
- Empêche l'état splitbrain ⇒ demote primary ∄ Quorum
- Sait prendre en compte l'extension Citus (sharding)

PRINCIPE

Système modulable

- Utilise un DCS pour
  - Propager la configuration du cluster
  - Stocker l'état du cluster
  - Propager certains éléments de configuration de PostgreSQL
- Utilise un système de backup pour construire des nœuds
- Configure PostgreSQL, pilote son fonctionnement et sa réplication

QUIZZ

Quelle politique de haute-disponibilité
PATRONI
met-il en œuvre vis à vis:

- Du théorème CAP ?
- Du théorème PACELC ?

OUTILS/INTERFACES

API REST

- Information sur la configuration
- État
- Santé
- ...

OUTILS/INTERFACES

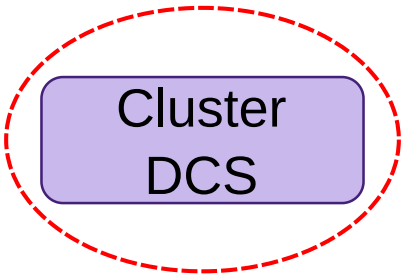
Outil CLI pour contrôler le cluster

- Éditer/consulter la configuration
- Voir l'état, le lag
- Piloter le cluster (switchover)

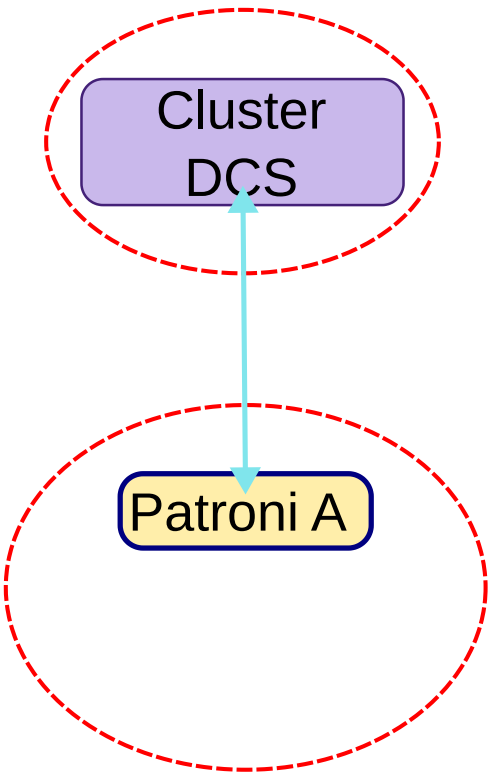
FONCTIONNEMENT

- **Patroni :**
  - Pilote PostgreSQL
  - Bootstrap du primaire
  - Création de réplicas
  - Gère les bascules
- **Mais on doit configurer indépendamment :**
  - Le cluster DCS
  - Le système de sauvegarde
- **Et les entrypoints ?**
  - Utiliser les données du DCS pour les autoconfigurer
  - Par exemple avec confd/remco

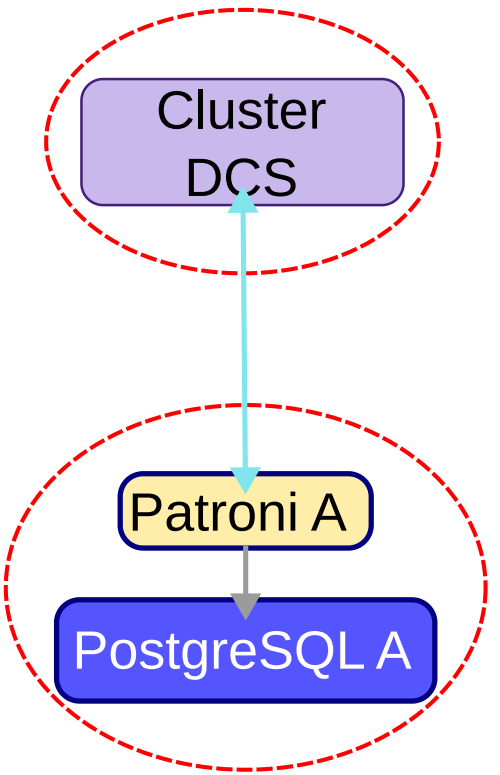
FONCTIONNEMENT



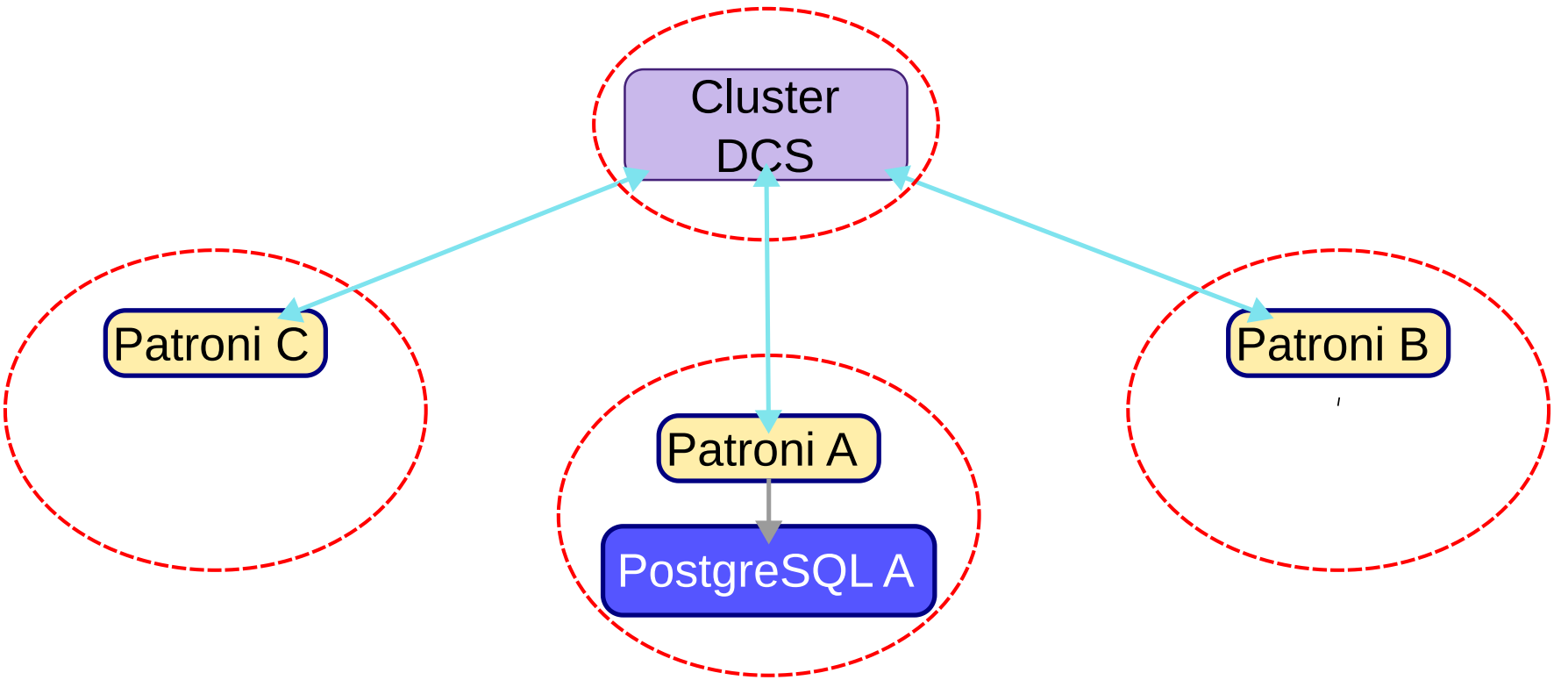
FONCTIONNEMENT



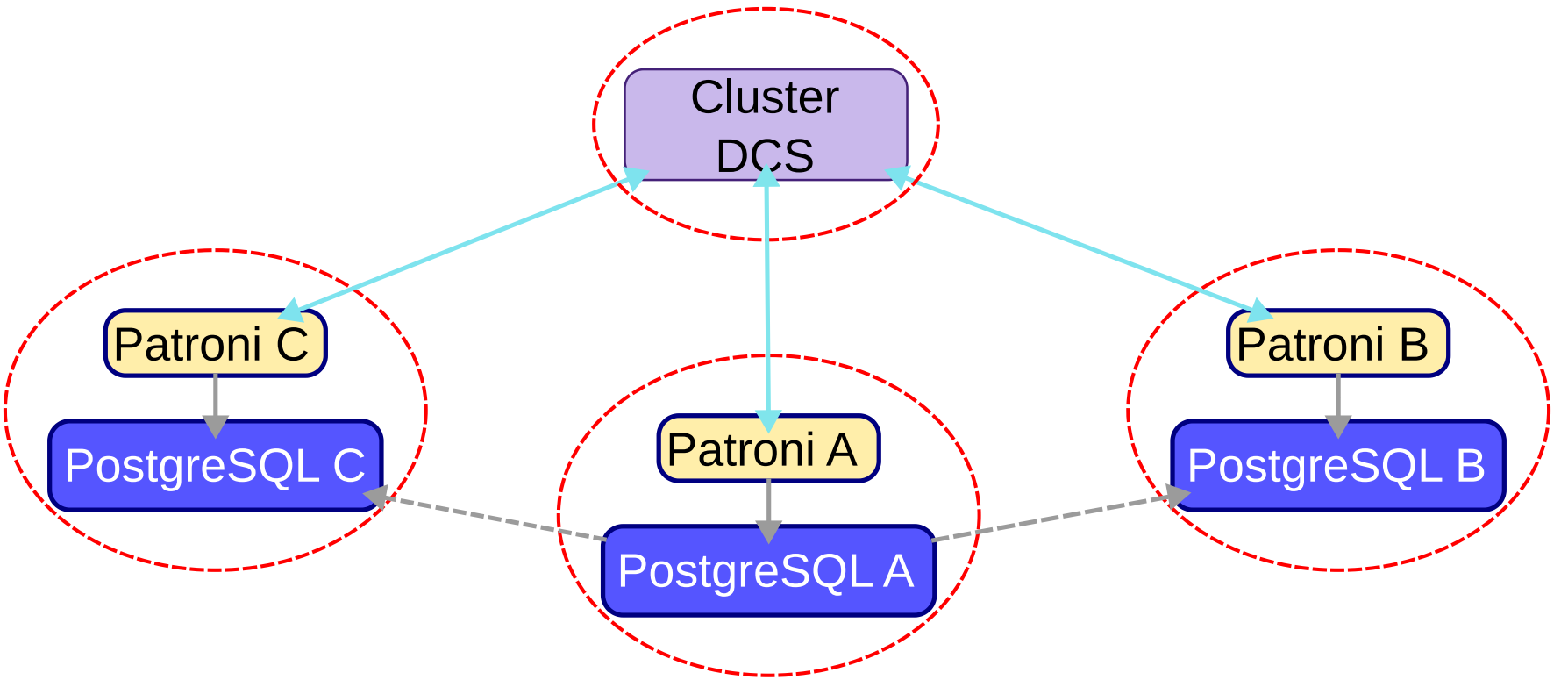
FONCTIONNEMENT



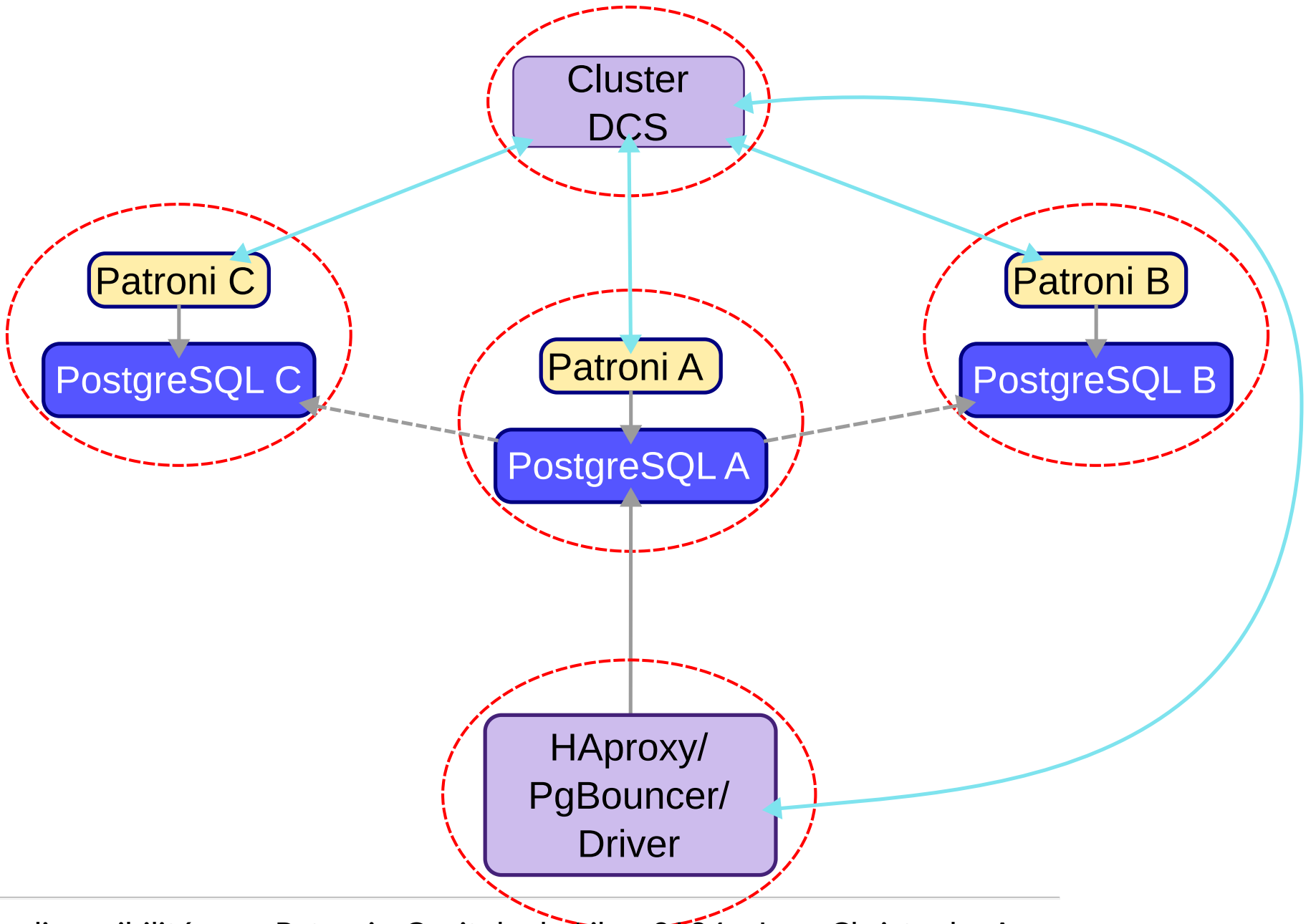
FONCTIONNEMENT



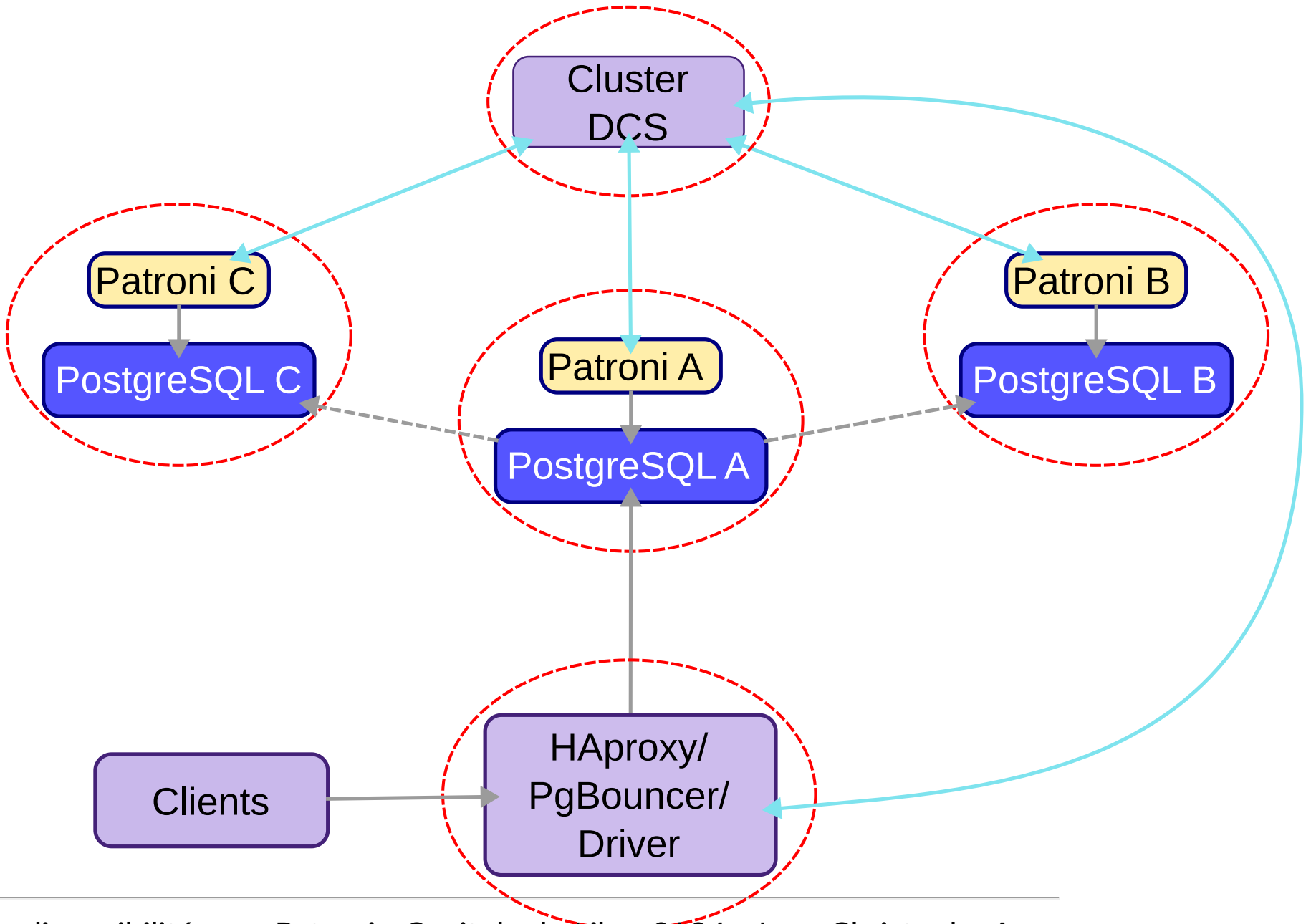
FONCTIONNEMENT



FONCTIONNEMENT



FONCTIONNEMENT



CONFIGURATION

```
scope: loxodemo
name: pgdeb01

restapi:
  listen: 10.200.0.11:8008
  connect_address: 10.200.0.11:8008

etcd3:
  hosts: # membres du cluster etcd.
    - 10.200.0.11:2379
    - 10.200.0.12:2379
    - 10.200.0.13:2379

log:
  level: INFO
  dir: /var/log/patroni
  file_size: 50000000
  file_num: 10
  format: '%(asctime)s %(levelname)s: %(message)s'
  dateformat: '%Y-%m-%d %H:%M:%S'
  loggers:
    etcd.client: INFO
```

CONFIGURATION

```
bootstrap: #DCS et initdb
  dcs:   # Utiliser edit-config pour modifier après init
    ttl: 30 #tps (s) validité d'une clé dans le DCS après mise à jour
    loop_wait: 10 # tps(s) entre 2 update DCS
    retry_timeout: 10 # tps (s) a attendre (2*) si DCS non joignable
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: false
      use_slots: false
      parameters:
        archive_mode: "on"
        archive_command: pgbackrest --stanza=loxodemo archive-push %p
      recovery_conf:
        restore_command: pgbackrest --stanza=loxodemo archive-get %f %p
  initdb:
  - encoding: UTF8
  - data-checksums

  pg_hba:
  - host all all 10.200.0.0/24 trust
  - host replication repli 10.200.0.0/24 trust
  users:
```

CONFIGURATION

```
postgresql:
  listen: 10.200.0.11:5432
  connect_address: 10.200.0.11:5432
  data_dir: /data/postgres/loxodemo
  bin_dir: /usr/lib/postgresql/16/bin/
  authentication:
    replication:
      username: repli
      password: azerty
    superuser:
      username: postgres
      password: azerty
  parameters:
    unix_socket_directories: '/tmp'
    logging_collector: on
  create_replica_methods:
    - pgbackrest
  pgbackrest:
    command: pgbackrest --stanza=loxodemo --delta restore
    keep_data: True
    no_params: True

```

CLI

```
patronictl topology loxodemo
+ Cluster: loxodemo (7436044389321027980) ------+----+-----------+
| Member    | Host         | Role    | State     | TL | Lag in MB |
+-----------+--------------+---------+-----------+----+-----------+
| pgdeb01   | 10.200.0.11 | Leader  | running   |  1 |           |
| + pgdeb02 | 10.200.0.12 | Replica | streaming |  1 |         0 |
| + pgdeb03 | 10.200.0.13 | Replica | streaming |  1 |         0 |
+-----------+--------------+---------+-----------+----+-----------+
```

```
 patronictl -c /etc/patroni/config.yml switchover --candidate=pgdeb02  loxodemo  --force
Current cluster topology
+ Cluster: loxodemo (7436044389321027980) ----+----+-----------+
| Member  | Host         | Role    | State     | TL | Lag in MB |
+---------+--------------+---------+-----------+----+-----------+
| pgdeb01 | 10.200.0.11 | Leader  | running   |  1 |           |
| pgdeb02 | 10.200.0.12 | Replica | streaming |  1 |         0 |
| pgdeb03 | 10.200.0.13 | Replica | streaming |  1 |         0 |
+---------+--------------+---------+-----------+----+-----------+
2024-11-11 15:35:05.12331 Successfully switched over to "pgdeb02"
+ Cluster: loxodemo (7436044389321027980) --------------+----+-----------+
| Member  | Host         | Role    | State               | TL | Lag in MB |
+---------+--------------+---------+---------------------+----+-----------+
| pgdeb01 | 10.200.0.11 | Replica | stopped             |    |   unknown |
| pgdeb02 | 10.200.0.12 | Leader  | running             |  1 |           |
| pgdeb03 | 10.200.0.13 | Replica | in archive recovery |  1 |         0 |
+---------+--------------+---------+---------------------+----+-----------+
```

```
patronictl -c /etc/patroni/config.yml topology loxodemo
+ Cluster: loxodemo (7436044389321027980) ------+----+-----------+
| Member    | Host         | Role    | State     | TL | Lag in MB |
+-----------+--------------+---------+-----------+----+-----------+
| pgdeb02   | 10.200.0.12 | Leader  | running   |  2 |           |
| + pgdeb01 | 10.200.0.11 | Replica | streaming |  2 |         0 |
| + pgdeb03 | 10.200.0.13 | Replica | streaming |  2 |         0 |
+-----------+--------------+---------+-----------+----+-----------+
```

CLI (JSON)

```
patronictl -c /etc/patroni/config.yml  list loxodemo  --format=json|jq .
[
  {
    "Cluster": "loxodemo",
    "Member": "pgdeb01",
    "Host": "10.200.0.11",
    "Role": "Replica",
    "State": "streaming",
    "TL": 2,
    "Lag in MB": 0
  },
  {
    "Cluster": "loxodemo",
    "Member": "pgdeb02",
    "Host": "10.200.0.12",
    "Role": "Leader",
    "State": "running",
    "TL": 2
  },
  {
    "Cluster": "loxodemo",
    "Member": "pgdeb03",
    "Host": "10.200.0.13",
    "Role": "Replica",
    "State": "streaming",
    "TL": 2,
    "Lag in MB": 0
  }
]
```

EXEMPLE DE CONTENU DU DCS

ICI AVEC ETCD

```
etcdctl get --from-key /service/loxodemo/
/service/loxodemo/config
{"ttl":30,"loop_wait":10,"retry_timeout":10,"maximum_lag_on_failover":1048576,"postgresql":{"use_pg
/service/loxodemo/failover
{}
/service/loxodemo/history
[[1,83886240,"no recovery target specified","2024-11-11T15:35:04.535410+00:00","pgdeb02"]]
/service/loxodemo/initialize
7436044389321027980
/service/loxodemo/leader
pgdeb02
/service/loxodemo/members/pgdeb01
{"conn_url":"postgres://10.200.0.11:5432/postgres","api_url":"http://10.200.0.11:8008/patroni","sta
/service/loxodemo/members/pgdeb02
{"conn_url":"postgres://10.200.0.12:5432/postgres","api_url":"http://10.200.0.12:8008/patroni","sta
/service/loxodemo/members/pgdeb03
{"conn_url":"postgres://10.200.0.13:5432/postgres","api_url":"http://10.200.0.13:8008/patroni","sta
/service/loxodemo/status
{"optime":83886696,"slots":{"pgdeb02":83886696},"retain_slots":["pgdeb01","pgdeb02"]}
```

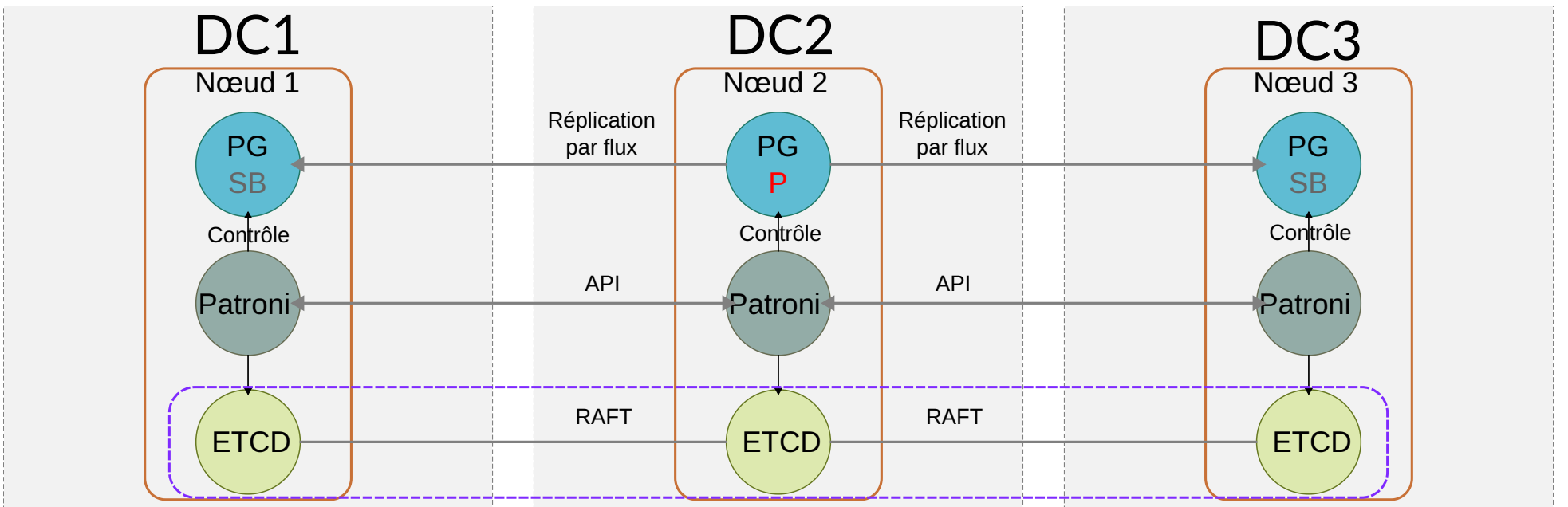
PENSEZ BIEN

- Quorum pas atteint ⇒ tous les nœuds passent read-only (demote du primaire)
- Mode DCS failsafe depuis patroni 3.0
- Outil de sauvegarde = offload du primaire lors de la création de standbys
- Architectures:
  - Simple
  - Cascade
  - Réplication asynchrone/synchrone (configuration spécifique!!!)

CAS D'USAGE

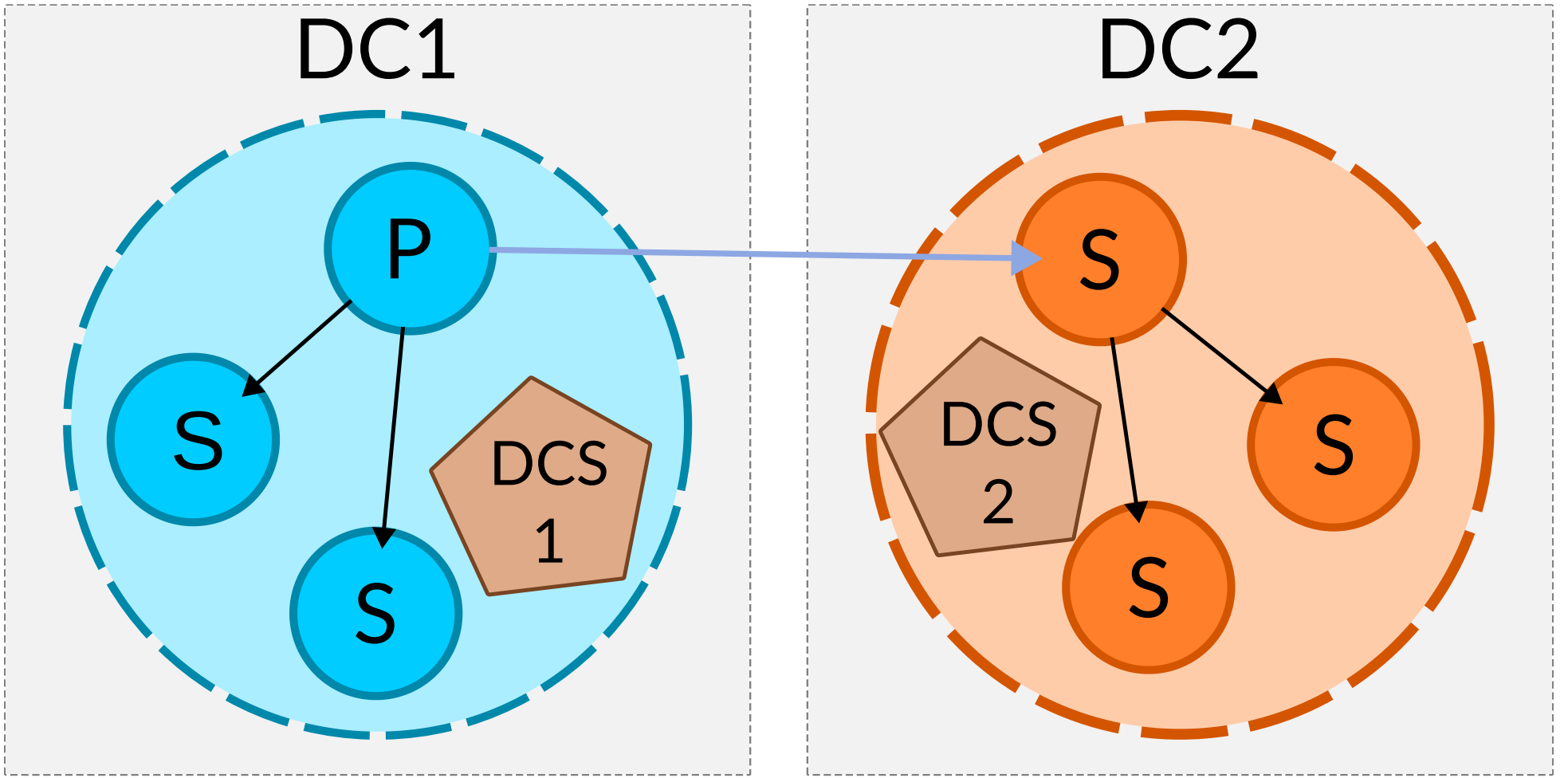
MULTI DC 1/2

1 DC est vu comme un standby de l'autre.


DC1
Nœud 1
PG
SB
Contrôle
Patroni
ETCD
Réplication
par flux
API
RAFT
DC2
Nœud 2
PG
P
Contrôle
Patroni
ETCD
Réplication
par flux
API
RAFT
DC3
Nœud 3
PG
SB
Contrôle
Patroni
ETCD

MULTI DC 2/2

1 DC est vu comme un standby de l'autre.


DC1
P
S
S
DCS
1
DC2
S
DCS
2
S
S

MERCI!