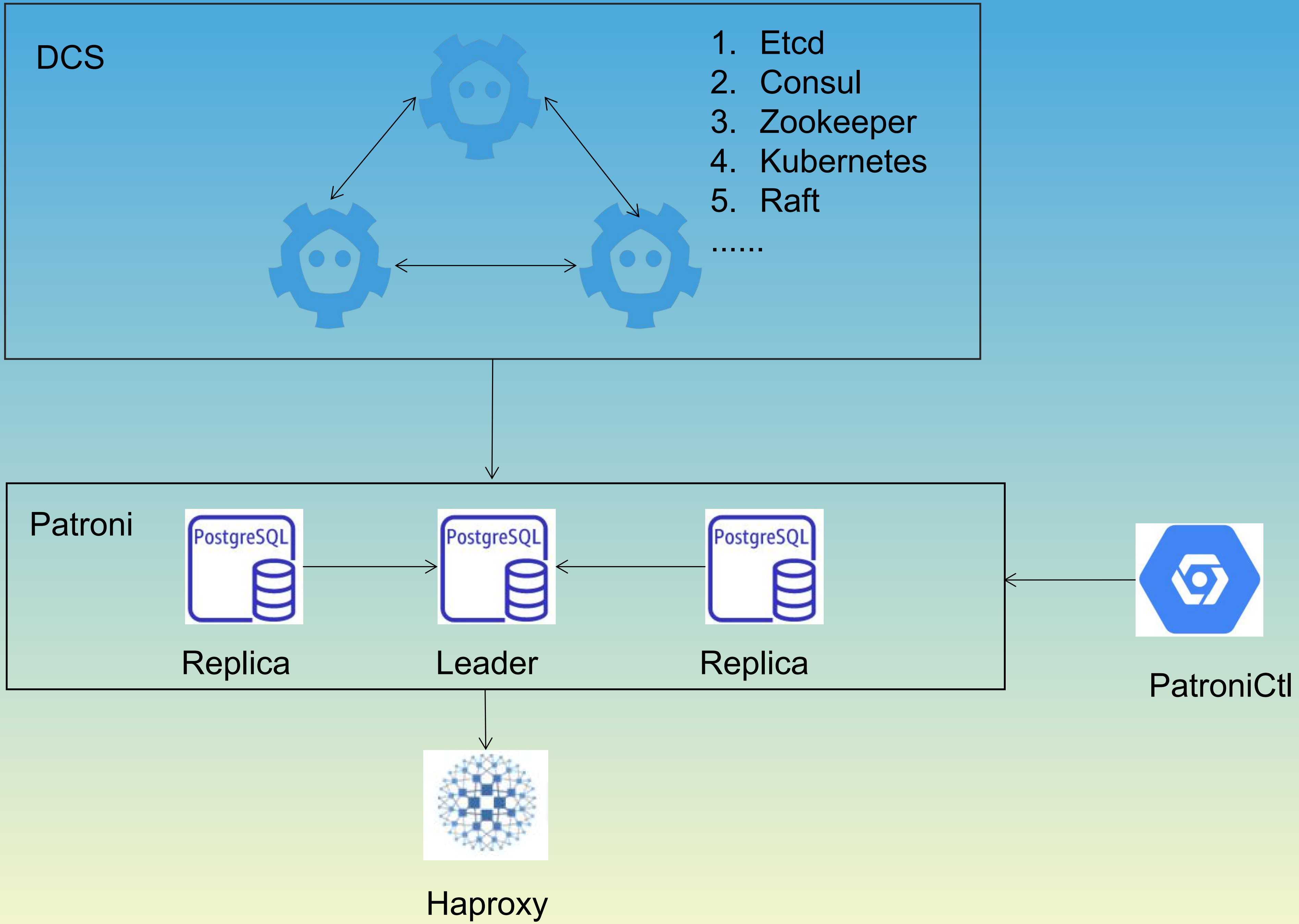
Patroni分享

- ✓ 简介
- ✓ 架构
- ✓ 功能

- 开箱即用高可用解决方案
- 降低运维成本，提升服务效率
  - 模板化快速部署
  - 避免PG集群脑裂发生
  - 提供备用集群功能
  - 一键故障切换
  - 故障自动转移
  - Watchdog机制

- 简介
- 架构
- 功能



DCS

1. Etcd
2. Consul
3. Zookeeper
4. Kubernetes
5. Raft

......

Patroni

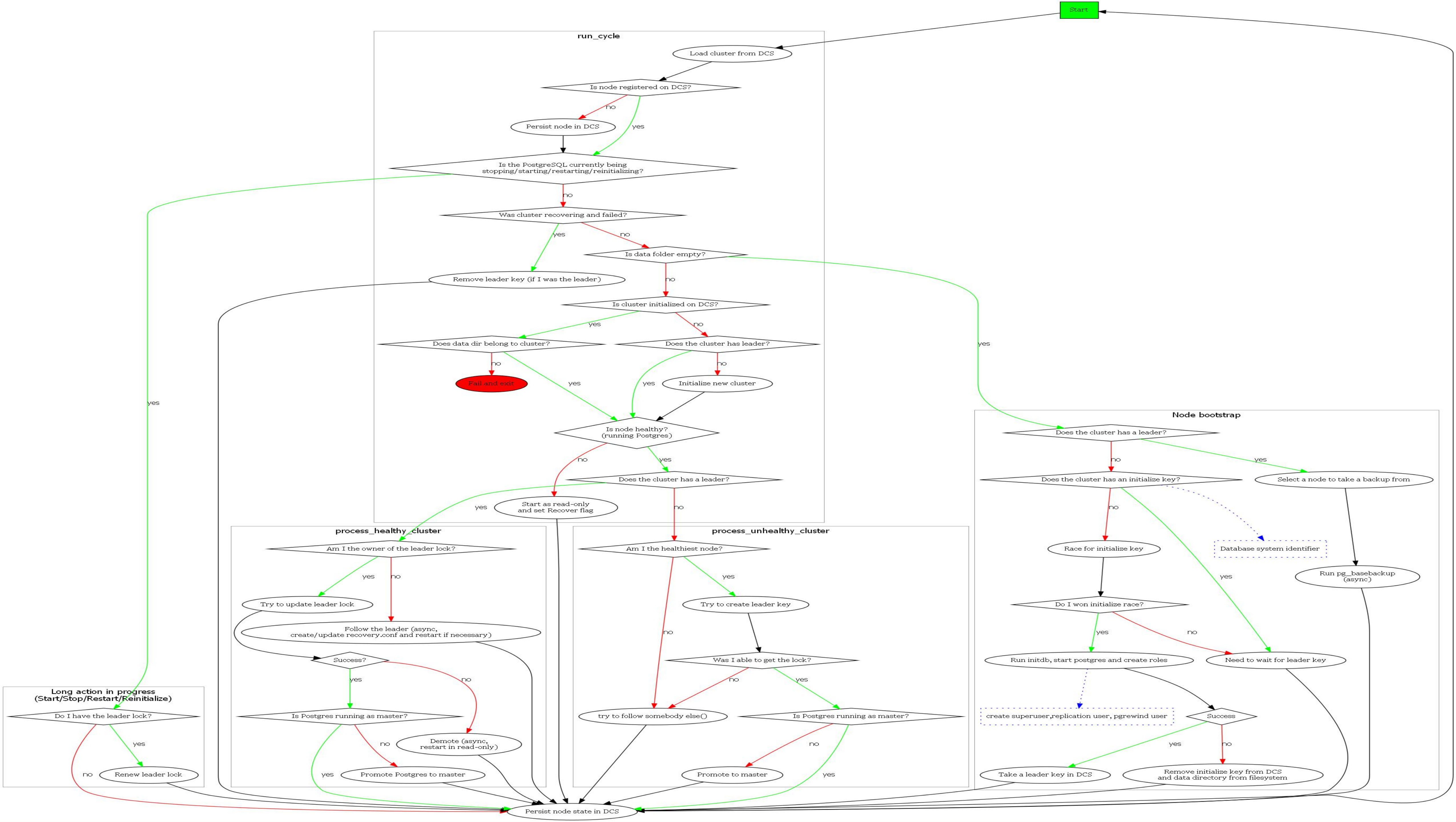
Replica

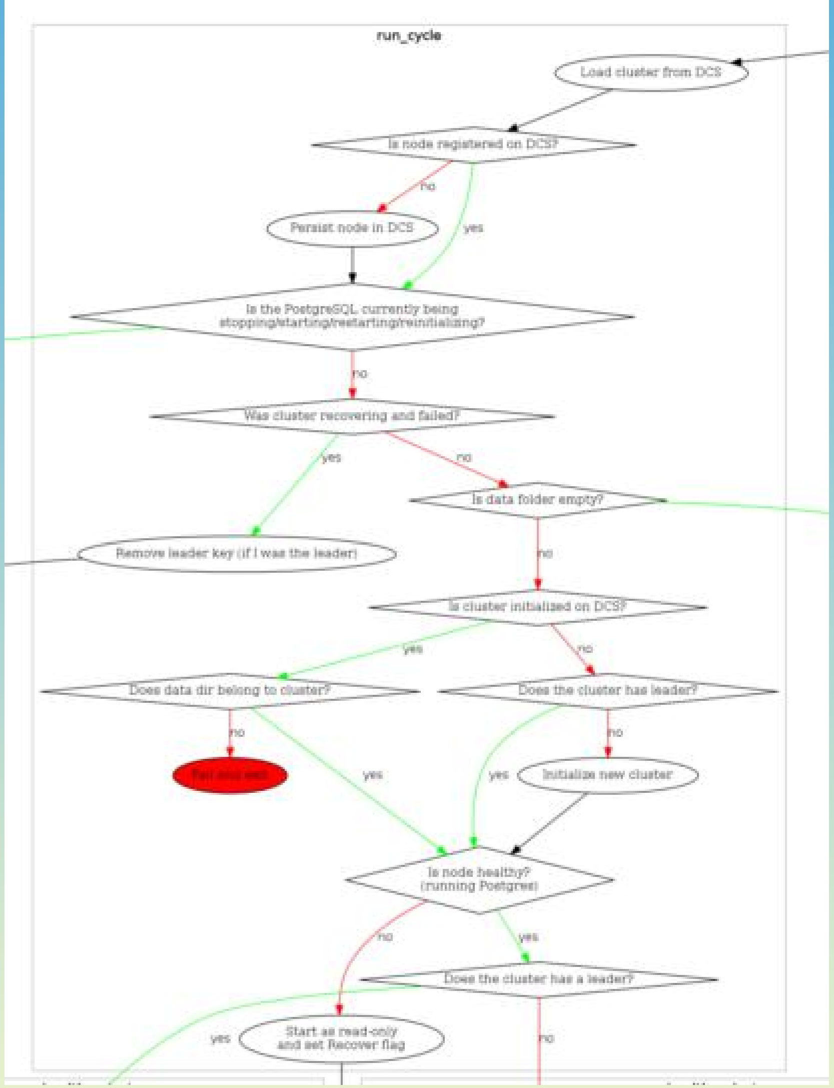
Leader

Replica

PatroniCtl

Haproxy


Start
run_cycle
Load cluster from DCS
Is node registered on DCS?
no
yes
Persist node in DCS
Is the PostgreSQL currently being stopping/starting/restarting/reinitializing?
no
yes
Was cluster recovering and failed?
yes
no
Is data folder empty?
Remove leader key (if I was the leader)
no
Is cluster initialized on DCS?
yes
no
Does data dir belong to cluster?
Does the cluster has leader?
no
yes
no
Fail and exit
yes
Initialize new cluster
Is node healthy? (running Postgres)
no
yes
Does the cluster has a leader?
yes
Start as read-only and set Recover flag
no
process_healthy_cluster
Am I the owner of the leader lock?
yes
no
Try to update leader lock
Follow the leader (async, create/update recovery.conf and restart if necessary)
Success?
yes
no
Is Postgres running as master?
Demote (async, restart in read-only)
no
yes
Promote Postgres to master
process_unhealthy_cluster
Am I the healthiest node?
yes
Try to create leader key
no
Was I able to get the lock?
no
yes
try to follow somebody else()
Is Postgres running as master?
no
Promote to master
yes
Long action in progress (Start/Stop/Restart/Reinitialize)
Do I have the leader lock?
yes
no
Renew leader lock
Node bootstrap
Does the cluster has a leader?
no
yes
Does the cluster has an initialize key?
Select a node to take a backup from
no
Database system identifier
Race for initialize key
yes
Run pg_basebackup (async)
Do I won initialize race?
yes
no
Run initdb, start postgres and create roles
Need to wait for leader key
create superuser,replication user, pgrewind user
Success
yes
no
Take a leader key in DCS
Remove initialize key from DCS and data directory from filesystem
Persist node state in DCS


run_cycle
Load cluster from DCS
Is node registered on DCS?
no
Persist node in DCS
yes
Is the PostgreSQL currently being stopping/starting/restarting/reinitializing?
no
Was cluster recovering and failed?
yes
no
Is data folder empty?
Remove leader key (if I was the leader)
no
Is cluster initialized on DCS?
yes
no
Does data dir belong to cluster?
Does the cluster has leader?
no
Fail and exit
yes
yes
no
Initialize new cluster
Is node healthy? (running Postgres)
no
yes
Does the cluster has a leader?
yes
Start as read-only and set Recover flag
no

①节点启动

步骤1.1：从DCS中加载集群信息

判断条件2.1：如果在DCS中已经注册了节点，那么执行判断条件2.2，如果没有注册，则执行步骤1.2

步骤1.2：在DCS中持久化节点信息

判读条件2.2：判读当前节点Postgresql的状态，如果是开始中、停止中、重新启动中以及重新初始化中，那么进入判断条件2.3，否则就进入判断条件2.4

判断条件2.3：判断该节点是否拥有领导者锁，如果拥有领导者锁，则执行步骤1.3,否则执行步骤1.4

步骤1.3：更新领导者锁

步骤1.4：持久化节点状态到DCS中

判断条件2.4：集群是否还原状态，并且失败了，如果是执行步骤1.5，否则执行判断条件2.5

步骤1.5: 如果当前节点是leader节点，则移除leader key。

判断条件2.5：判断数据目录是否为空，如果是执行步骤1.6，否则执行判断条件2.6

步骤1.6：执行②节点拉起流程

判断条件2.6：集群信息在DCS中初始化，如果是则执行判断条件2.7，否则执行判断条件2.8

判断条件2.7：数据目录是否属于集群，如果是执行判断条件2.9，否则执行步骤1.7

判断条件2.8：集群是否有领导者，如果是执行判断条件2.9，否则执行步骤1.8

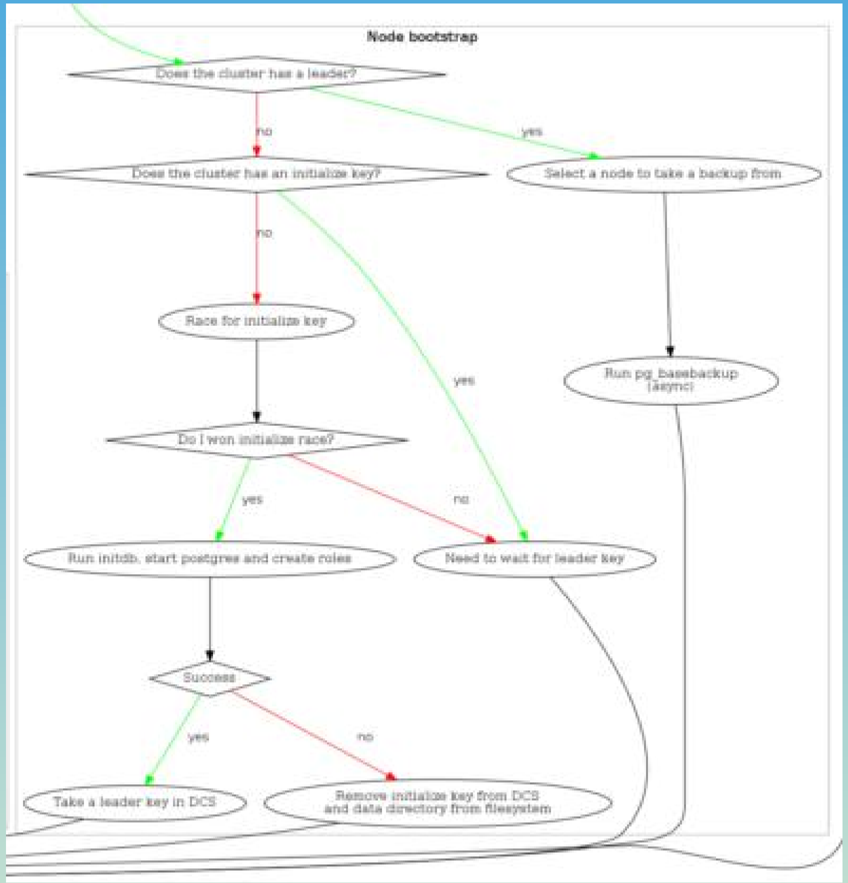
步骤1.7：节点启动失败并退出

判断条件2.9：节点是否健康状态（Postgresql运行中），如果是执行判断条件2.10，否则执行步骤1.9

步骤1.8：初始化一个新集群

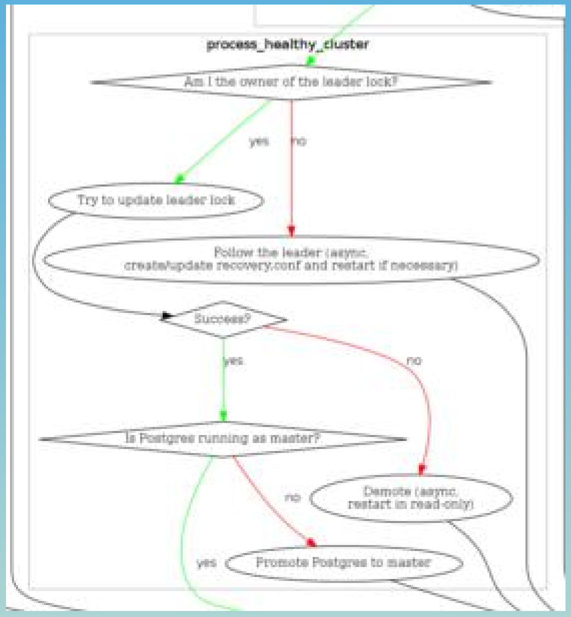
步骤1.9：设置成只读节点及还原标志

判断条件2.10：集群是否有一个领导者，如果是执行③处理健康集群流程，否则执行④处理不健康集群流程


Node bootstrap
Does the cluster has a leader?
no
yes
Does the cluster has an initialize key?
Select a node to take a backup from
no
Race for initialize key
yes
Run pg_basebackup
(async)
Do I won initialize race?
yes
no
Run initdb, start postgres and create roles
Need to wait for leader key
Success
yes
no
Take a leader key in DCS
Remove initialize key from DCS
and data directory from filesystem

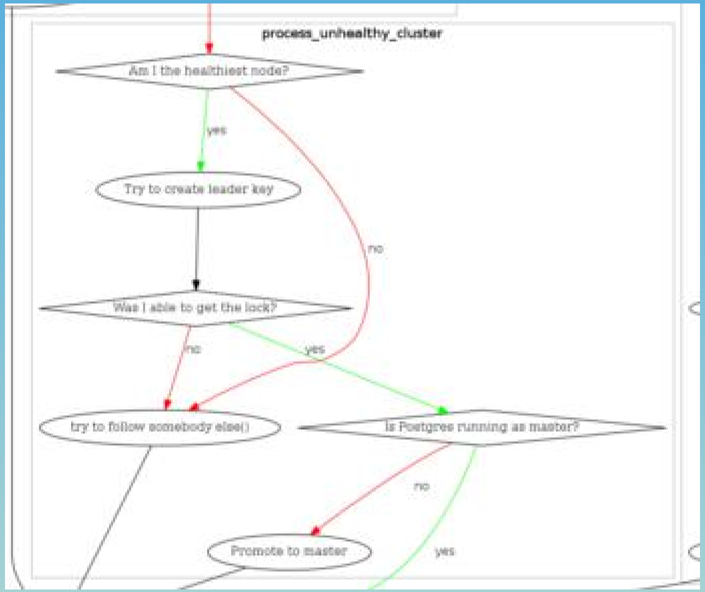
②节点拉起流程

判断条件2.1：集群是否有一个领导者，如果是执行步骤1.1，否则执行判断条件2.2
步骤1.1：选择一个节点，并且获得备份，执行步骤1.2
步骤1.2：执行pg_basebackup还原备份
判断条件2.2：集群是否有一个初始键，如果是执行步骤1.3，否则执行步骤1.4
步骤1.3：等待一个leader key
步骤1.4：竞争初始键
判断条件2.3：判断是否赢得了竞争初始键，如果是执行步骤1.5，否则执行步骤1.6
步骤1.5：初始化数据库、运行Postgresql并且创建对应角色，执行判断条件2.4
步骤1.6：需要等待leader key
判断条件2.4：操作成功，执行步骤1.7，否则执行步骤1.8
步骤1.7：将leader key存储到DCS中，执行步骤1.9
步骤1.8：从DCS中移除初始化键，并且删除数据目录，执行步骤1.9
步骤1.9：持久化节点状态到DCS中


process_healthy_cluster
Am I the owner of the leader lock?
yes
no
Try to update leader lock
Follow the leader (async, create/update recovery.conf and restart if necessary)
Success?
yes
no
Is Postgres running as master?
no
Demote (async, restart in read-only)
yes
Promote Postgres to master

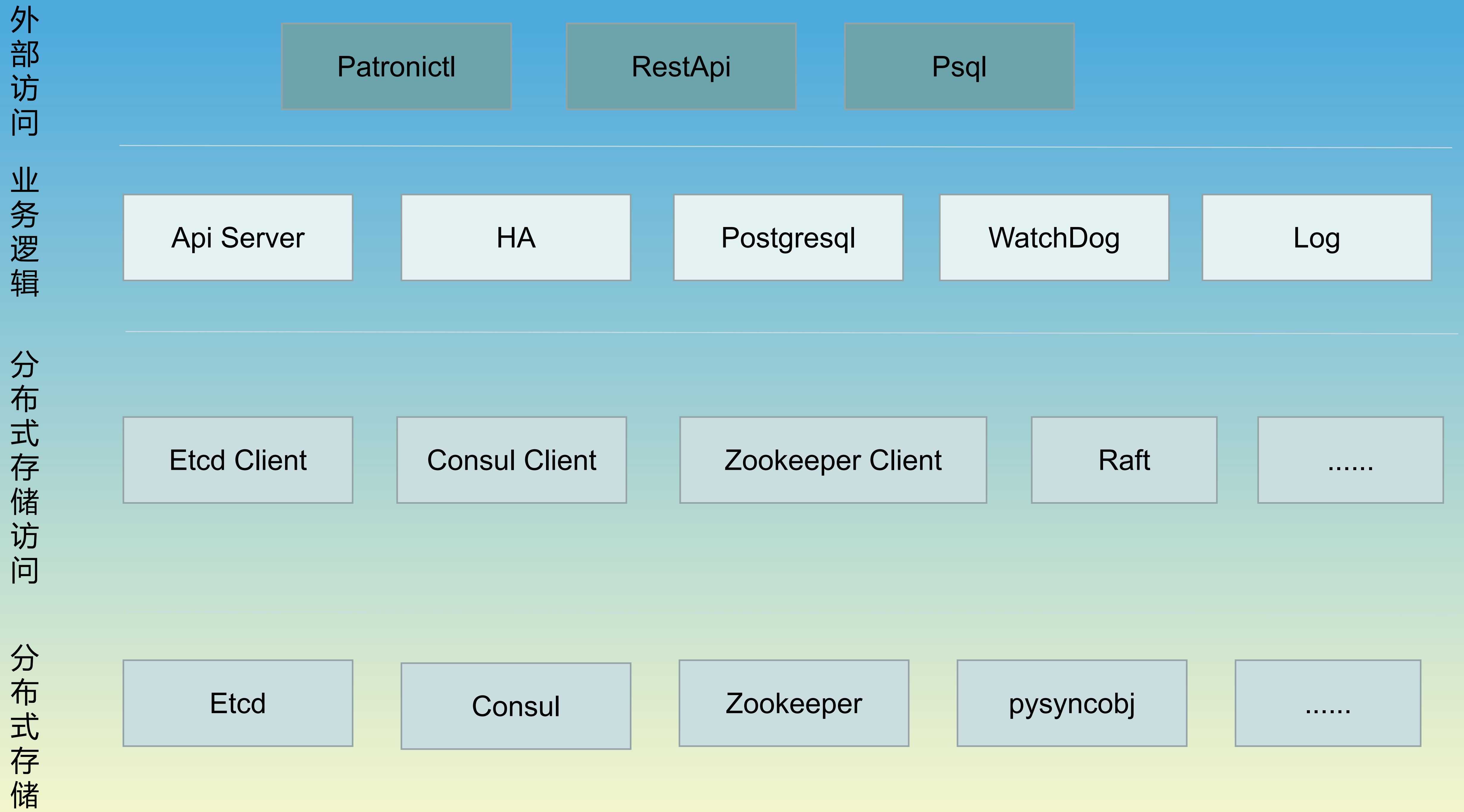
③处理健康集群

判断条件2.1：判断当前节点是否拥有领导者锁，如果是执行步骤1.1，否则执行步骤1.2
步骤1.1：尝试更新领导者锁，执行判断条件2.2
步骤1.2：跟随领导者
判断条件2.2：如果执行成功，则执行判断条件2.3，否则执行步骤1.3
判断条件2.3：当前节点是否作为主节点在运行，如果是执行步骤1.5，否则步骤1.4
步骤1.3：执行节点降级操作
步骤1.4：提升当前节点为主节点
步骤1.5：持久化节点状态到DCS中


process_unhealthy_cluster
Am I the healthiest node?
yes
no
Try to create leader key
Was I able to get the lock?
no
yes
try to follow somebody else()
Is Postgres running as master?
no
Promote to master
yes

④处理不健康集群

判断条件2.1：判断当前节点是否为健康状态，如果是执行步骤1.1，否则执行步骤1.2
步骤1.1：创建leader key
步骤1.2：尝试跟随其他节点
判断条件2.2：是否可以获得锁，如果可以执行判断条件2.3，否则执行步骤1.2
判断条件2.3：当前节点是否作为Postgresql主节点在运行，如果是执行步骤1.3，否则执行步骤1.4
步骤1.3：持久化节点状态到DCS中
步骤1.4：提升当前节点为主节点



外部访问

Patronictl　RestApi　Psql

业务逻辑

Api Server　HA　Postgresql　WatchDog　Log

分布式存储访问

Etcd Client　Consul Client　Zookeeper Client　Raft　......

分布式存储

Etcd　Consul　Zookeeper　pysyncobj　......

- 简介
- 架构
- 功能

Patroni

参数配置

REST API

安全

复制和引导

WatchDog

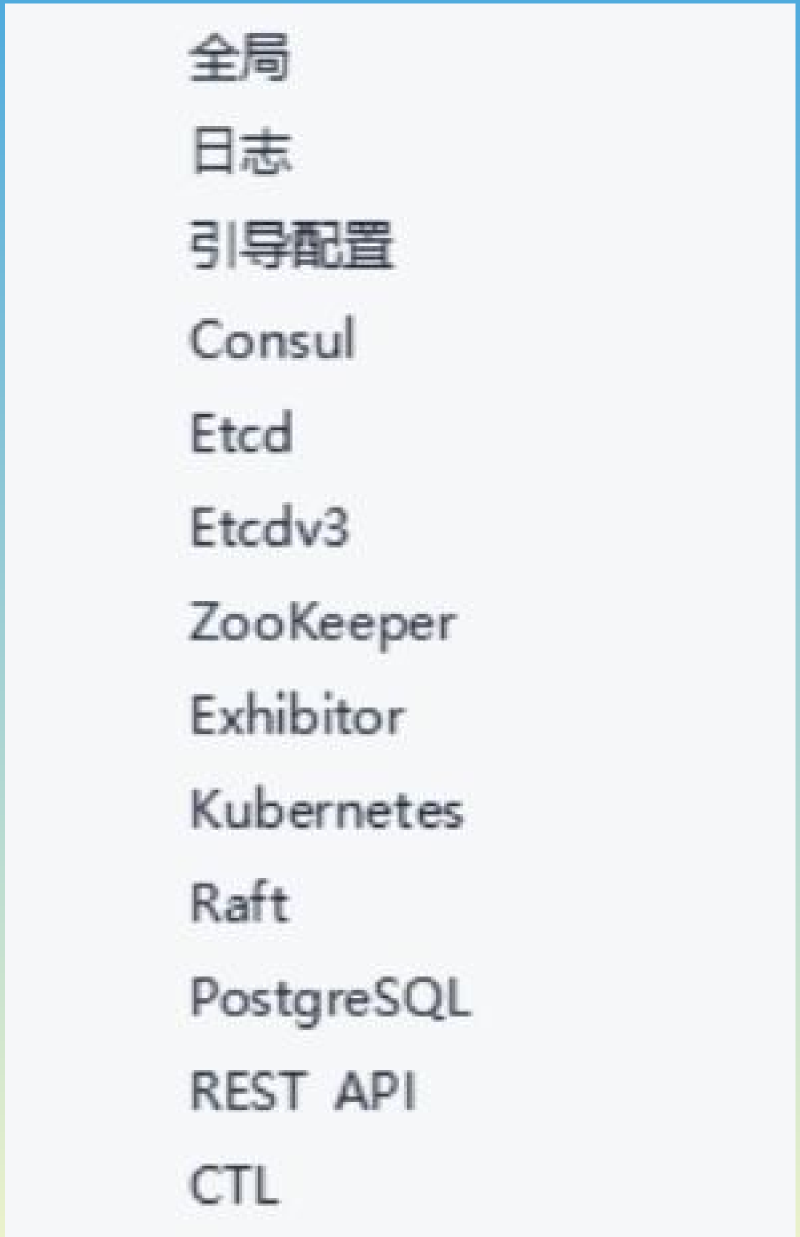
Kubernetes

暂停和恢复

复制模式

参数配置

- 动态配置
- 本地配置
- 环境配置



全局

日志

引导配置

Consul

Etcd

Etcdv3

ZooKeeper

Exhibitor

Kubernetes

Raft

PostgreSQL

REST API

CTL

参数配置

- 动态配置

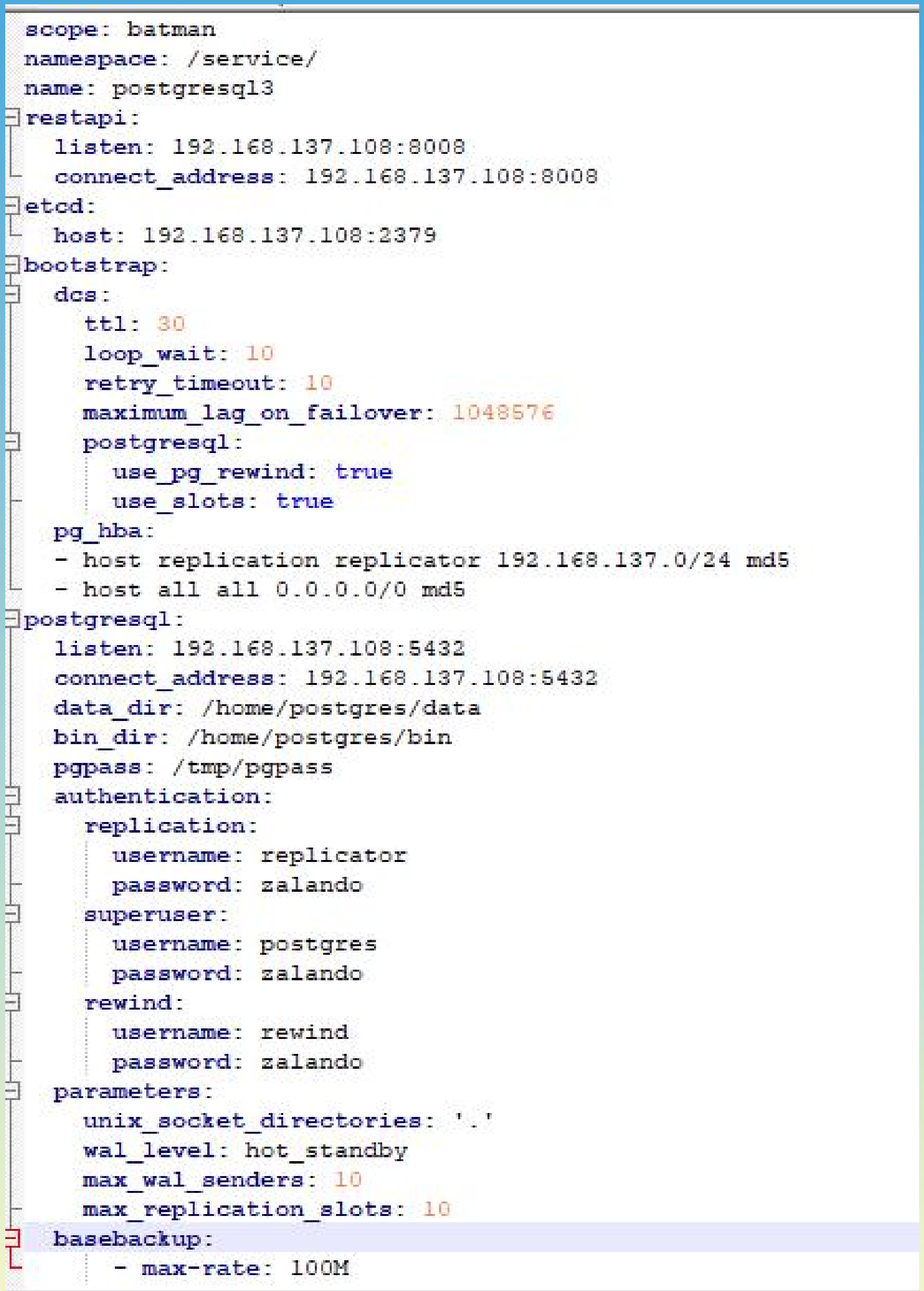
存储在DCS中配置信息

```
ttl: 30
loop_wait: 10
retry_timeouts: 10
maximum_lag_on_failover: 1048576
max_timelines_history: 0
check_timeline: false
postgresql.use_slots: true

max_connections: 100
max_locks_per_transaction: 64
max_worker_processes: 8
max_prepared_transactions: 0
wal_level: hot_standby
wal_log_hints: on
track_commit_timestamp: off
max_wal_senders: 5
max_replication_slots: 5
wal_keep_segments: 8
wal_keep_size: 128MB
```

参数配置

- 本地配置

postgresql.yml

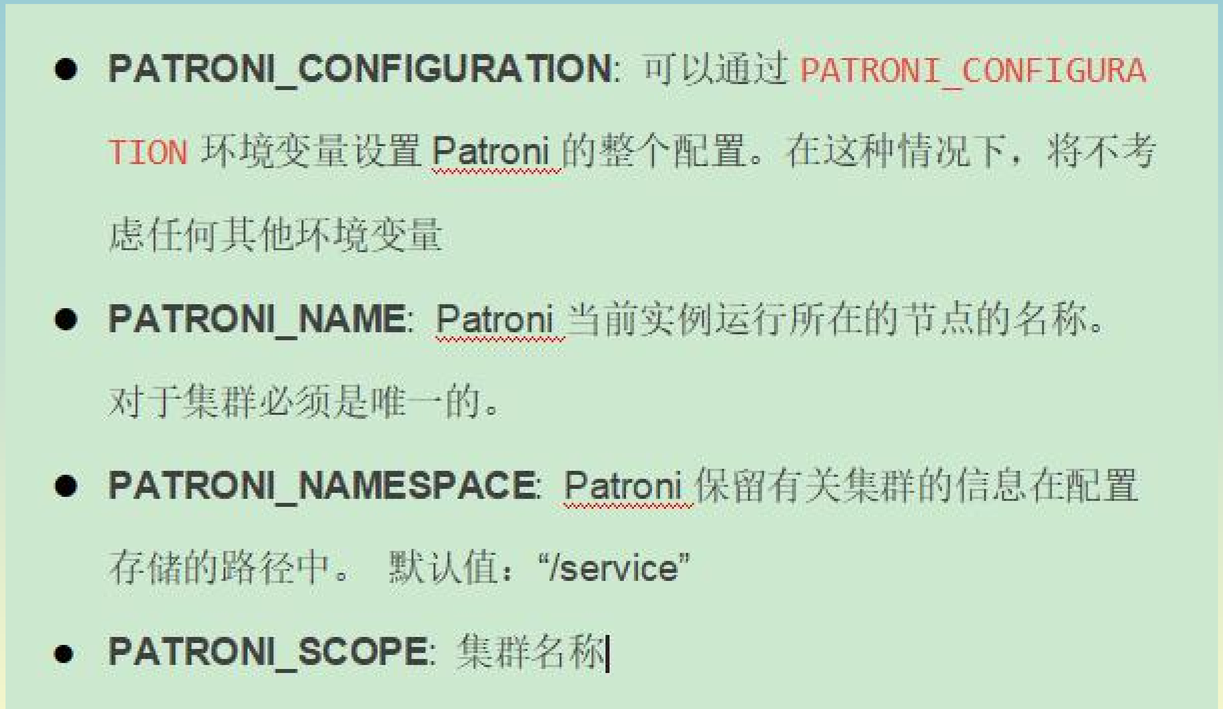


```
scope: batman
namespace: /service/
name: postgresql3
restapi:
  listen: 192.168.137.108:8008
  connect_address: 192.168.137.108:8008
etcd:
  host: 192.168.137.108:2379
bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: true
      use_slots: true
  pg_hba:
  - host replication replicator 192.168.137.0/24 md5
  - host all all 0.0.0.0/0 md5
postgresql:
  listen: 192.168.137.108:5432
  connect_address: 192.168.137.108:5432
  data_dir: /home/postgres/data
  bin_dir: /home/postgres/bin
  pgpass: /tmp/pgpass
  authentication:
    replication:
      username: replicator
      password: zalando
    superuser:
      username: postgres
      password: zalando
    rewind:
      username: rewind
      password: zalando
  parameters:
    unix_socket_directories: '.'
    wal_level: hot_standby
    max_wal_senders: 10
    max_replication_slots: 10
  basebackup:
      - max-rate: 100M
```

参数配置

- 环境配置

存储在本地环境变量中，常用于容器环境



- **PATRONI_CONFIGURATION**: 可以通过 PATRONI_CONFIGURATION 环境变量设置 Patroni 的整个配置。在这种情况下，将不考虑任何其他环境变量
- **PATRONI_NAME**: Patroni 当前实例运行所在的节点的名称。对于集群必须是唯一的。
- **PATRONI_NAMESPACE**: Patroni 保留有关集群的信息在配置存储的路径中。 默认值：“/service”
- **PATRONI_SCOPE**: 集群名称

RestApi

- 健康检查

GET /
GET /master
GET /leader
GET /primary
GET /read-write
GET /replica

GET /read-only
GET /standby-leader
GET /synchronous or GET /sync
GET /asynchronous or GET /async
GET /async?lag=1048576
GET /health
GET /liveness
GET /readiness

RestApi

- 监控

  GET /patroni

- 集群状态

  GET /cluster
  GET /history

RestApi

- 配置

  GET /config
  PATCH /config
  PUT /config

- 切换和故障转移

  POST /switchover
  POST /failover

RestApi

- 重启

  POST /restart
  DELETE /restart

- 重载

  POST /reload

- 重新初始化

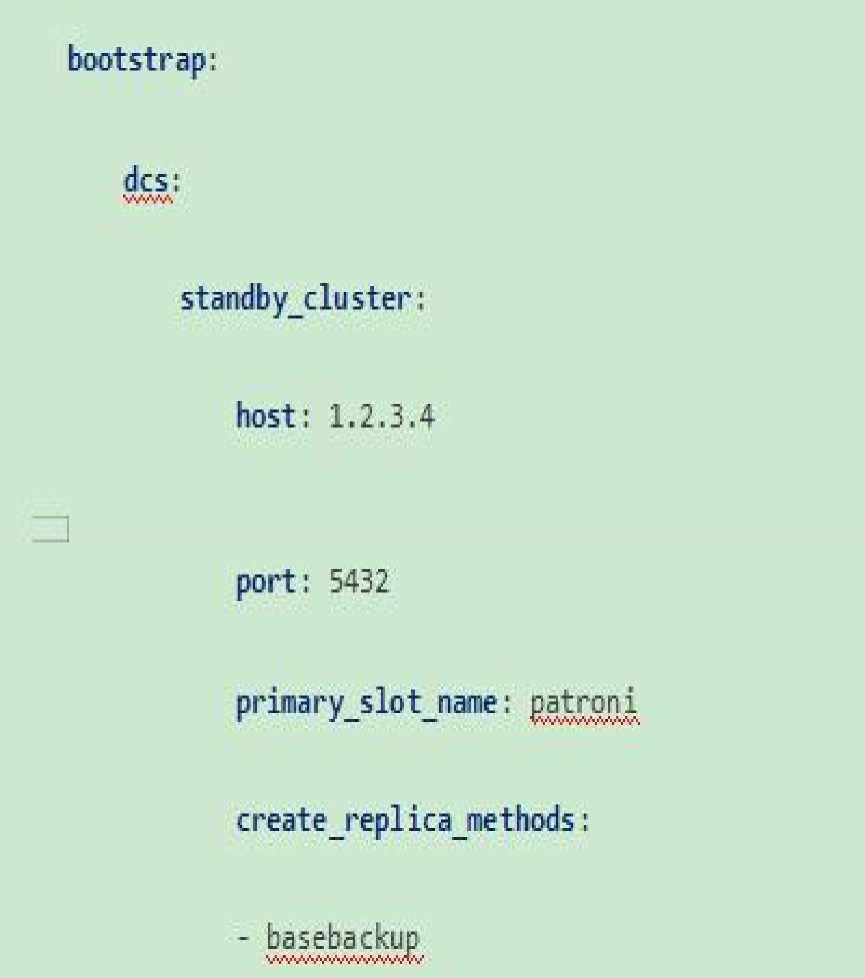
  POST /reinitialize

安全

- DCS
- RestApi

引导和复制

- 引导
- 复制
- 备用集群



```
bootstrap:
  dcs:
    standby_cluster:
      host: 1.2.3.4
      port: 5432
      primary_slot_name: patroni
      create_replica_methods:
      - basebackup
```

复制模式

- 异步模式
- 同步模式
  - synchronous_commit: "on"
  - synchronous_standby_names: "*"
  - synchronous_mode: "on"
  - synchronous_mode_strict： "on"
  - synchronous_node_count: 1

暂停与恢复

- Patroni进行数据库升级

原则：先备后主

1. 暂停Patroni故障转移，即执行pause命令。

2. 在每个备用Postgresql节点上为单个Postgresql节点执行升级步骤。

3. 恢复Patroni故障转移，即执行resume命令。

4. 手动将Postgresql主节点切换到升级后的备用节点。

5. 再次暂停Patroni故障转移,即执行pause命令。

6. 在以前的主节点上为单个Postgresql节点执行升级步骤。

7. 再次恢复Patroni故障转移，即执行resume命令。

8. 或者将Postgresql主节点切换回原始节点。

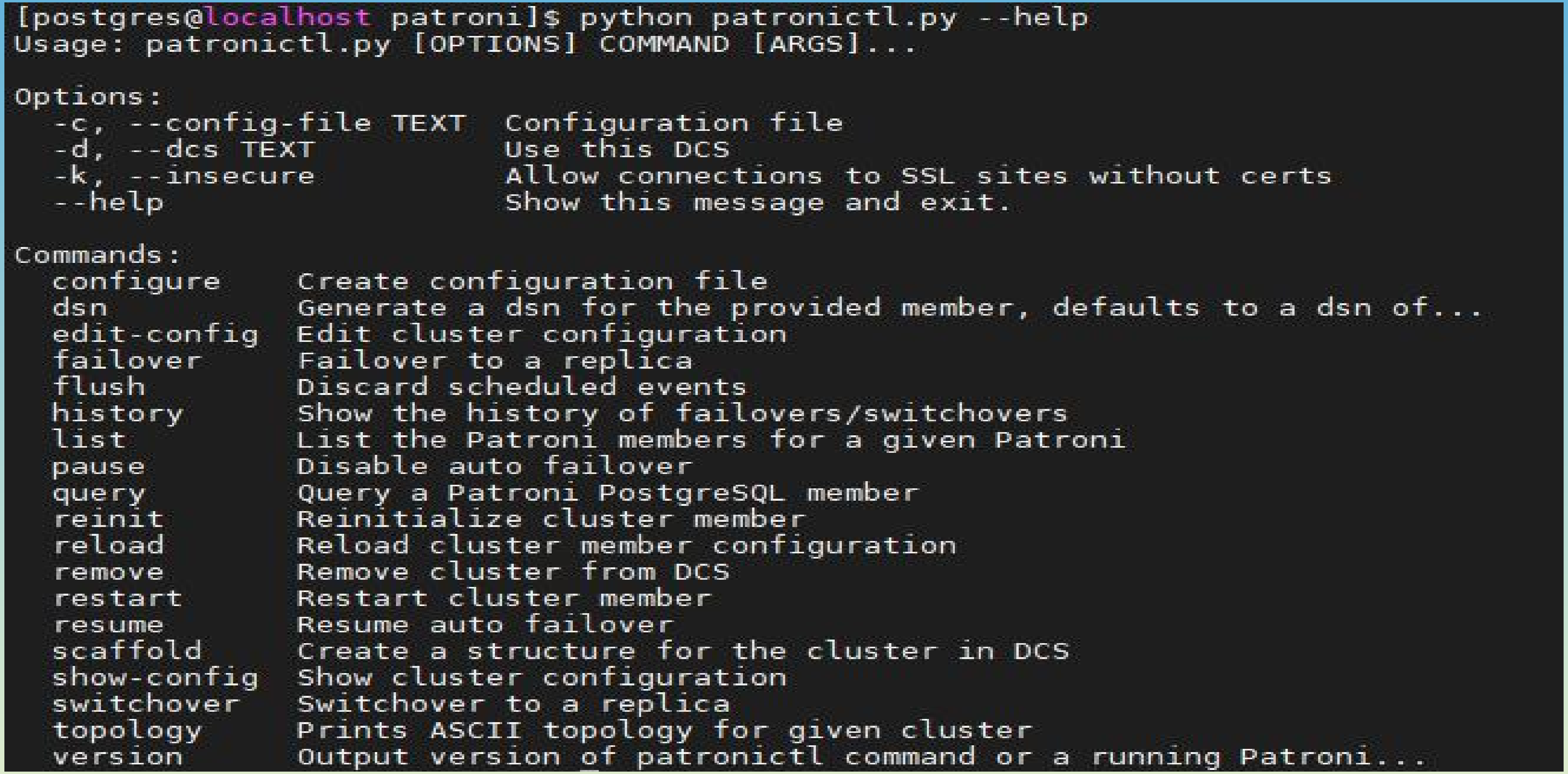
Kubernetes

- StatefulSet
- Endpoints
- Service
- Secret
- ServiceAccount
- Role
- ClusterRole
- ClusterRoleBinding

WatchDog

- mode: off, automatic 或者required.
- device: watchdog设备的路径。默认为 /dev/watchdog.
- safety_margin: 看门狗触发和领导者密钥到期之间的安全余量秒数。

一、Patronictl



```
[postgres@localhost patroni]$ python patronictl.py --help
Usage: patronictl.py [OPTIONS] COMMAND [ARGS]...

Options:
  -c, --config-file TEXT  Configuration file
  -d, --dcs TEXT          Use this DCS
  -k, --insecure          Allow connections to SSL sites without certs
  --help                  Show this message and exit.

Commands:
  configure    Create configuration file
  dsn          Generate a dsn for the provided member, defaults to a dsn of...
  edit-config  Edit cluster configuration
  failover     Failover to a replica
  flush        Discard scheduled events
  history      Show the history of failovers/switchovers
  list         List the Patroni members for a given Patroni
  pause        Disable auto failover
  query        Query a Patroni PostgreSQL member
  reinit       Reinitialize cluster member
  reload       Reload cluster member configuration
  remove       Remove cluster from DCS
  restart      Restart cluster member
  resume       Resume auto failover
  scaffold     Create a structure for the cluster in DCS
  show-config  Show cluster configuration
  switchover   Switchover to a replica
  topology     Prints ASCII topology for given cluster
  version      Output version of patronictl command or a running Patroni...
```

- 环境要求

  硬件环境:

  oraclevm 虚拟机

  软件环境:

  操作系统版本：CentOS Linux release 7.7.1908 (Core)

  python版本: 2.7.5

主机名	IP	安装软件	角色
centos1	192.168.137.101	Etcd、patroni、haproxy	Leader
Centos2	192.168.137.104	Etcd、patroni	Follower
Centos3	192.168.137.103	Etcd、patroni	Follower

系统依赖

- 安装系统依赖

```
yum -y install gcc etcd haproxy libyaml
yum -y install epel-release
yum -y install python-pip
yum -y install python-devel
```

防火墙

- 关闭防火墙

```
systemctl stop firewalld
systemctl disable firewalld
```

ETCD

- 配置文件

```
vim /etc/etcd/etcd.conf
ETCD_NAME=etcd_1
ETCD_DATA_DIR="/var/lib/etcd/default.etcd"
ETCD_LISTEN_PEER_URLS="http://192.168.137.101:2380"
ETCD_LISTEN_CLIENT_URLS="http://192.168.137.101:2379,http://127.0.0.1:2379"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.168.137.101:2380"
ETCD_INITIAL_CLUSTER="etcd_1=http://192.168.137.101:2380,etcd_2=http://192.168.137.104:2380,etcd_3=http://192.168.137.103:2380"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster"
ETCD_ADVERTISE_CLIENT_URLS="http://192.168.137.101:2379"
```

ETCD

- 启动etcd服务

```
systemctl start etcd
```

- 查看etcd状态

```
etcdctl --write-out="table" --endpoints=http://192.168.137.101:2379,http://192.168.137.104:2379,http://192.168.137.103:2379 endpoint status
```

- 安装软件依赖

```
pip install --upgrade pip
pip install psycopg2==2.5.4
pip install --upgrade setuptools
pip install -r requirements.txt
```

- 查看依赖项目

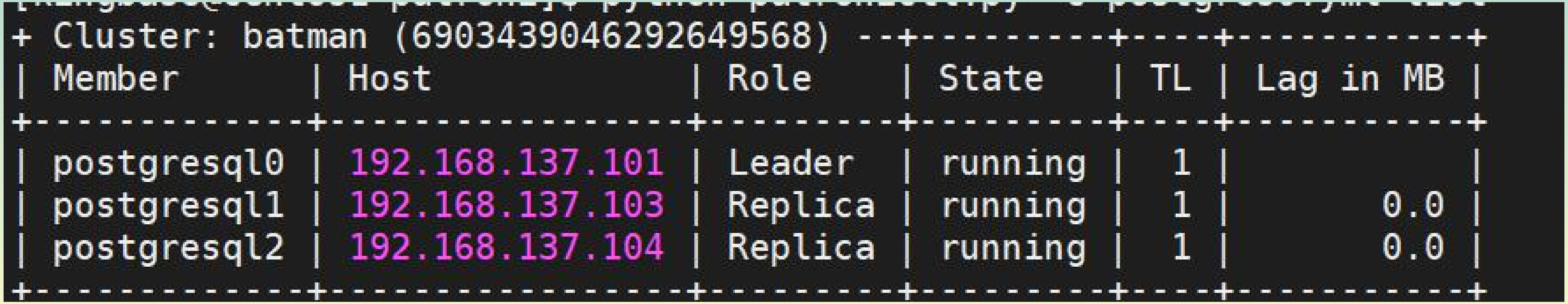
```
pip list
```

```
urllib3>=1.19.1,!=1.21
ipaddress; python_version=="2.7"
boto
PyYAML
six >= 1.7
kazoo>=1.3.1
python-etcd>=0.4.3,<0.5
python-consul>=0.7.1
click>=4.1
prettytable>=0.7
python-dateutil
pysyncobj>=0.3.7
psutil>=2.0.0
ydiff>=1.2.0
```

- 修改patroni的配置文件
  pg0.yml
  pg1.yml
  pg2.yml

- 启动patroni
  python patroni.py pg0.yml
  python patroni.py pg1.yml
  python patroni.py pg2.yml

- 查看patroni状态
  python patronictl.py -c pg0.yml list

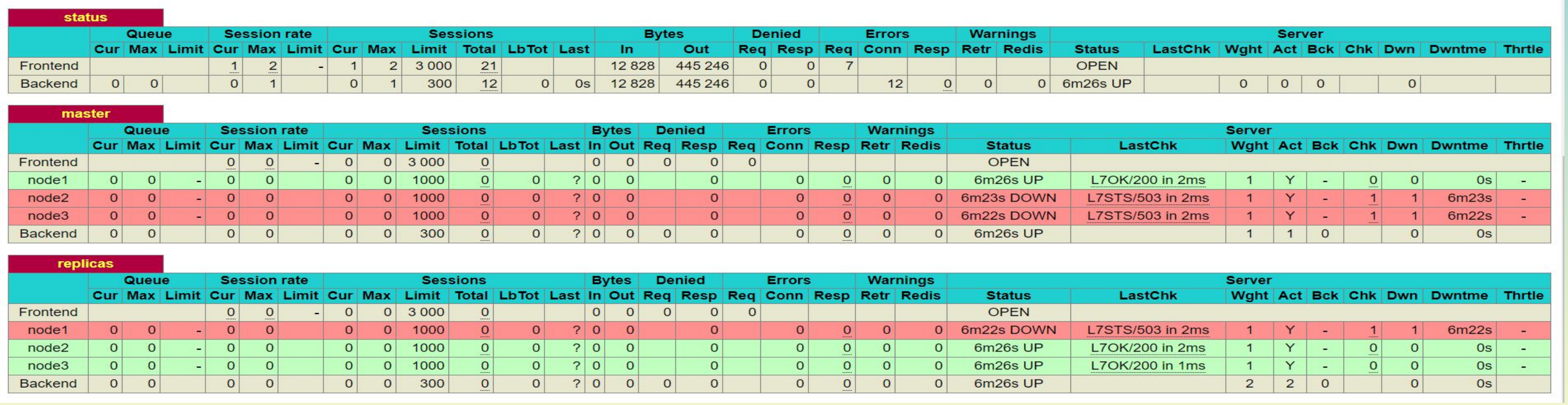


```
+ Cluster: batman (6903439046292649568) --+---------+----+-----------+
| Member      | Host            | Role    | State   | TL | Lag in MB |
+-------------+-----------------+---------+---------+----+-----------+
| postgresql0 | 192.168.137.101 | Leader  | running |  1 |           |
| postgresql1 | 192.168.137.103 | Replica | running |  1 |       0.0 |
| postgresql2 | 192.168.137.104 | Replica | running |  1 |       0.0 |
+-------------+-----------------+---------+---------+----+-----------+
```

HAPROXY

- 配置HAPROXY
  配置管理端口（1080）
  配置写端口（5000）
  配置读端口（5001）

- 启动Haproxy
  systemctl start haproxy
- 访问Haproxy
  访问：http://192.168.137.101:1080/haproxy-stats
  用户名：admin
  密码：passw0rd



status

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				1	2	-	1	2	3 000	21			12 828	445 246	0	0	7					OPEN								
Backend	0	0		0	1		0	1	300	12	0	0s	12 828	445 246	0	0		12	0	0	0	6m26s UP		0	0	0		0		

master

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	3 000	0			0	0	0	0	0					OPEN								
node1	0	0	-	0	0		0	0	1000	0	0	?	0	0		0		0	0	0	0	6m26s UP	L7OK/200 in 2ms	1	Y	-	0	0	0s	-
node2	0	0	-	0	0		0	0	1000	0	0	?	0	0		0		0	0	0	0	6m23s DOWN	L7STS/503 in 2ms	1	Y	-	1	1	6m23s	-
node3	0	0	-	0	0		0	0	1000	0	0	?	0	0		0		0	0	0	0	6m22s DOWN	L7STS/503 in 2ms	1	Y	-	1	1	6m22s	-
Backend	0	0		0	0		0	0	300	0	0	?	0	0	0	0		0	0	0	0	6m26s UP		1	1	0		0	0s	

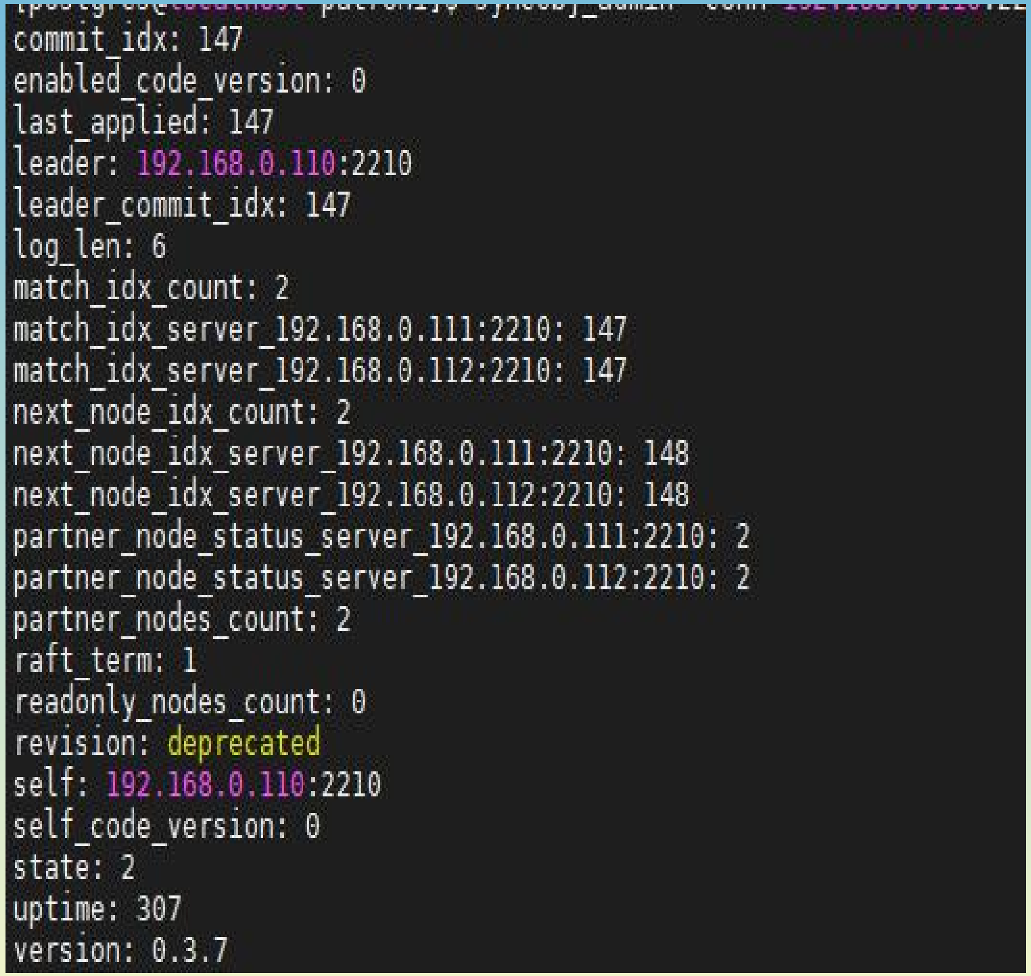
replicas

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	3 000	0			0	0	0	0	0					OPEN								
node1	0	0	-	0	0		0	0	1000	0	0	?	0	0		0		0	0	0	0	6m22s DOWN	L7STS/503 in 2ms	1	Y	-	1	1	6m22s	-
node2	0	0	-	0	0		0	0	1000	0	0	?	0	0		0		0	0	0	0	6m26s UP	L7OK/200 in 2ms	1	Y	-	0	0	0s	-
node3	0	0	-	0	0		0	0	1000	0	0	?	0	0		0		0	0	0	0	6m26s UP	L7OK/200 in 1ms	1	Y	-	0	0	0s	-
Backend	0	0		0	0		0	0	300	0	0	?	0	0	0	0		0	0	0	0	6m26s UP		2	2	0		0	0s	

其他扩展功能

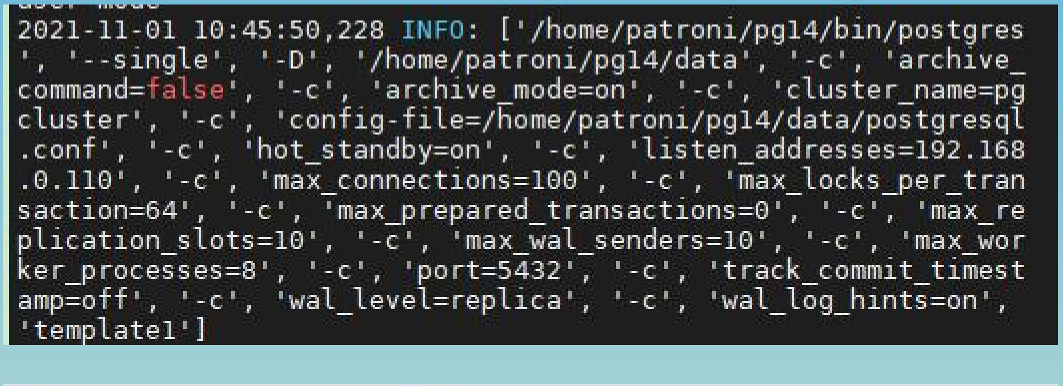
➢ 支持Raft

- state - it can be one of the folowing:
  0 - folower
  1 - candidate
  2 - leader
- leader - current cluster leader
- partner_nodes_count - number of partner nodes
- partner_node_status - statuses of connections to partner nodes:
  0 - disconnected
  1 - connecting
  2 - connected
- commit_idx - last commited transaction number
- last_applied - last applied transaction number
- version - version of a library
- revision - previous git commit hash
- uptime - number of seconds that node process is alive

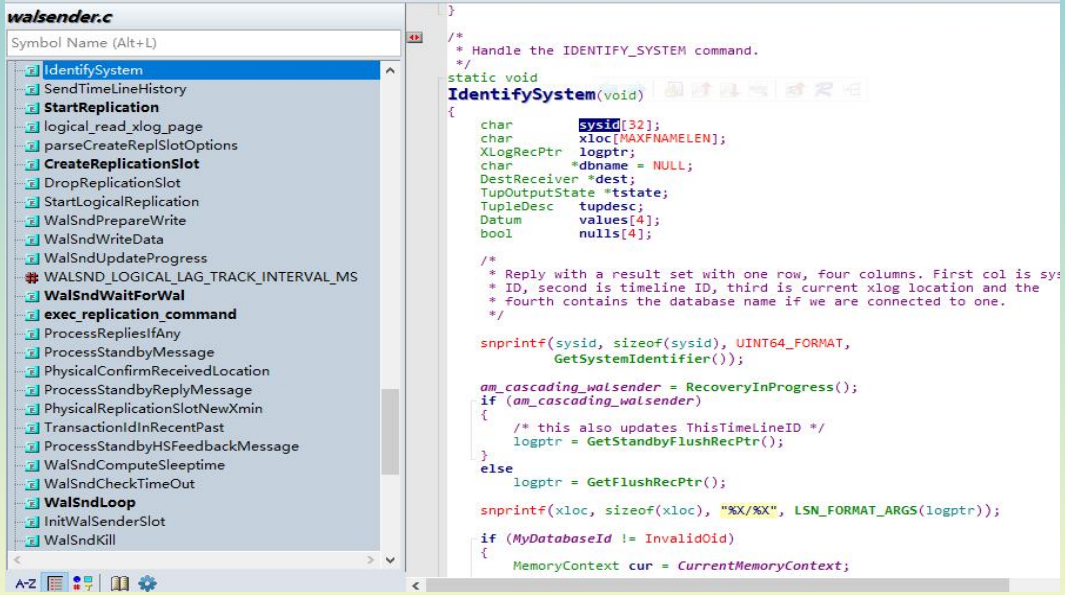


```
commit_idx: 147
enabled_code_version: 0
last_applied: 147
leader: 192.168.0.110:2210
leader_commit_idx: 147
log_len: 6
match_idx_count: 2
match_idx_server_192.168.0.111:2210: 147
match_idx_server_192.168.0.112:2210: 147
next_node_idx_count: 2
next_node_idx_server_192.168.0.111:2210: 148
next_node_idx_server_192.168.0.112:2210: 148
partner_node_status_server_192.168.0.111:2210: 2
partner_node_status_server_192.168.0.112:2210: 2
partner_nodes_count: 2
raft_term: 1
readonly_nodes_count: 0
revision: deprecated
self: 192.168.0.110:2210
self_code_version: 0
state: 2
uptime: 307
version: 0.3.7
```

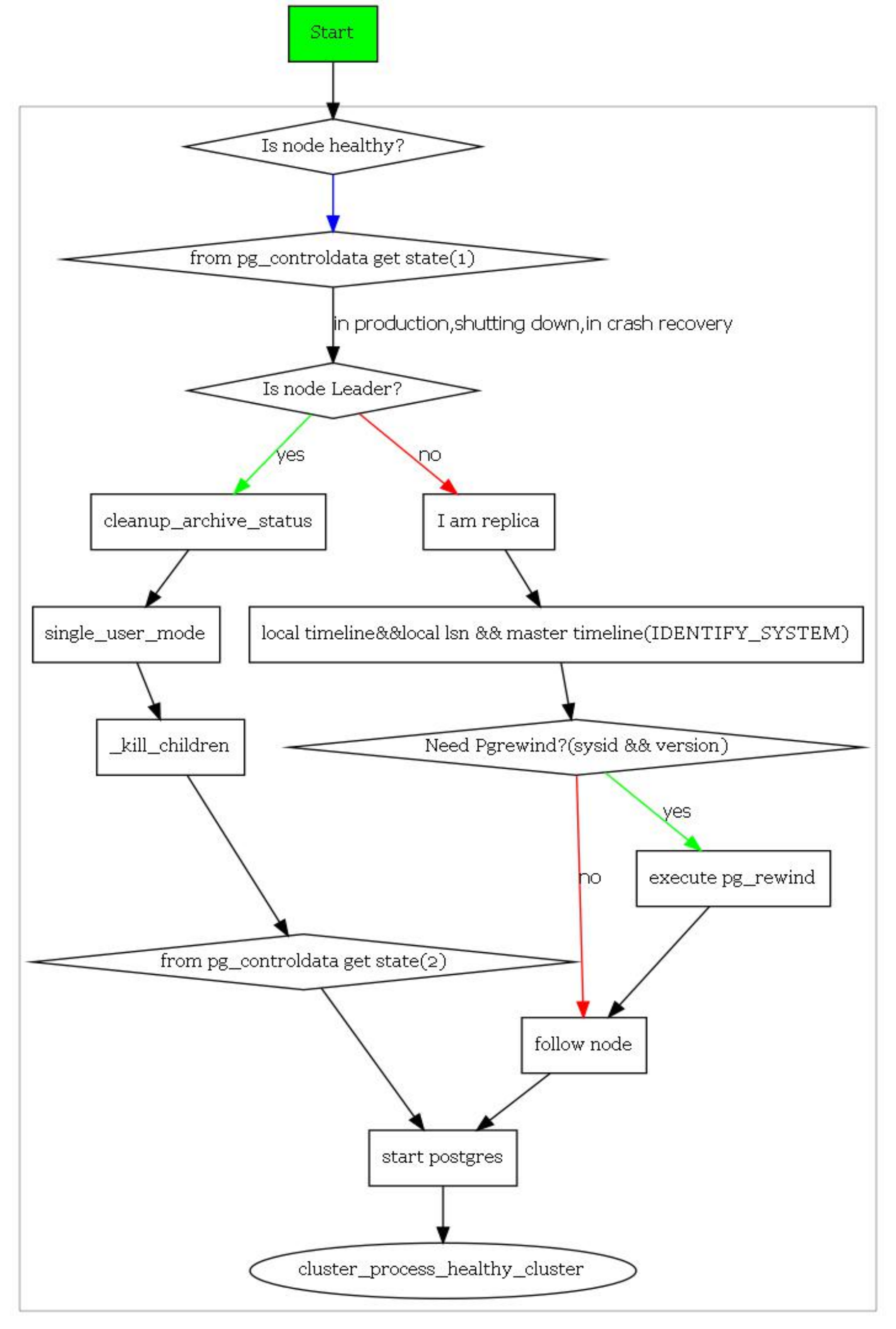
- 场景一
主机或备机数据库异常终止



```
2021-11-01 10:45:50,228 INFO: ['/home/patroni/pg14/bin/postgres
', '--single', '-D', '/home/patroni/pg14/data', '-c', 'archive_
command=false', '-c', 'archive_mode=on', '-c', 'cluster_name=pg
cluster', '-c', 'config-file=/home/patroni/pg14/data/postgresql
.conf', '-c', 'hot_standby=on', '-c', 'listen_addresses=192.168
.0.110', '-c', 'max_connections=100', '-c', 'max_locks_per_tran
saction=64', '-c', 'max_prepared_transactions=0', '-c', 'max_re
plication_slots=10', '-c', 'max_wal_senders=10', '-c', 'max_wor
ker_processes=8', '-c', 'track_commit_timest
amp=off', '-c', 'wal_level=replica', '-c', 'wal_log_hints=on',
'template1']
```



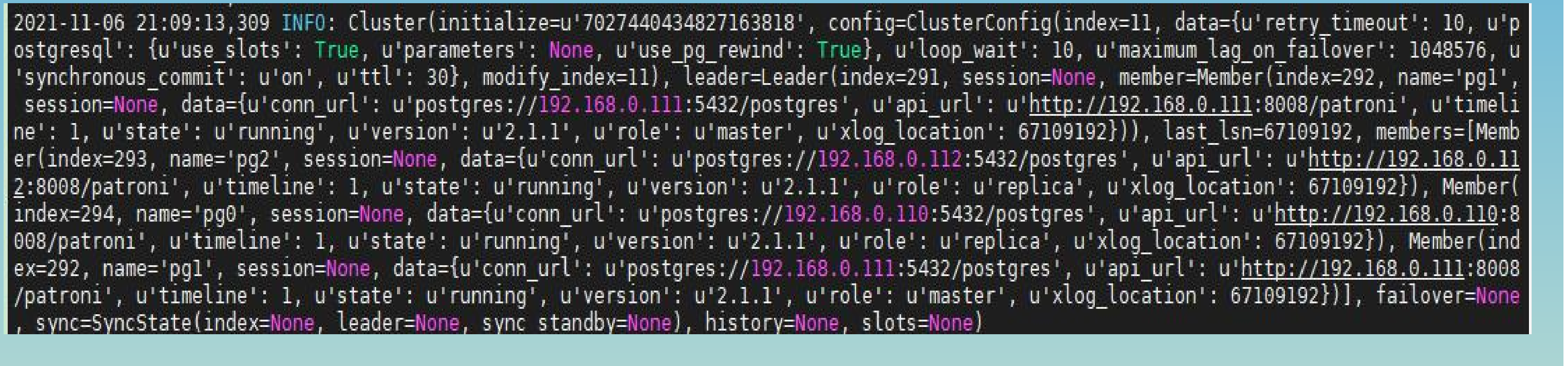
```c
}

/*
 * Handle the IDENTIFY_SYSTEM command.
 */
static void
IdentifySystem(void)
{
    char        sysid[32];
    char        xloc[MAXFNAMELEN];
    XLogRecPtr  logptr;
    char       *dbname = NULL;
    DestReceiver *dest;
    TupOutputState *tstate;
    TupleDesc   tupdesc;
    Datum       values[4];
    bool        nulls[4];

    /*
     * Reply with a result set with one row, four columns. First col is sys
     * ID, second is timeline ID, third is current xlog location and the
     * fourth contains the database name if we are connected to one.
     */

    snprintf(sysid, sizeof(sysid), UINT64_FORMAT,
             GetSystemIdentifier());

    am_cascading_walsender = RecoveryInProgress();
    if (am_cascading_walsender)
    {
        /* this also updates ThisTimeLineID */
        logptr = GetStandbyFlushRecPtr();
    }
    else
        logptr = GetFlushRecPtr();

    snprintf(xloc, sizeof(xloc), "%X/%X", LSN_FORMAT_ARGS(logptr));

    if (MyDatabaseId != InvalidOid)
    {
        MemoryContext cur = CurrentMemoryContext;
```



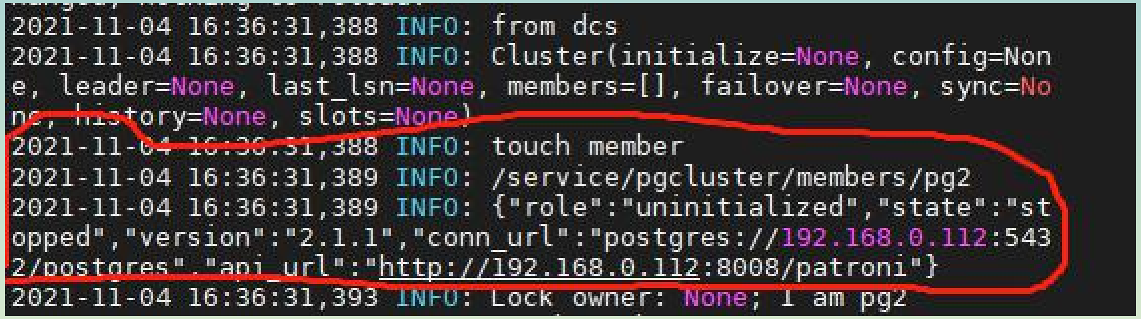
Start
Is node healthy?
from pg_controldata get state(1)
in production,shutting down,in crash recovery
Is node Leader?
yes
no
cleanup_archive_status
I am replica
single_user_mode
local timeline&&local lsn && master timeline(IDENTIFY_SYSTEM)
_kill_children
Need Pgrewind?(sysid && version)
yes
no
execute pg_rewind
from pg_controldata get state(2)
follow node
start postgres
cluster_process_healthy_cluster

场景二

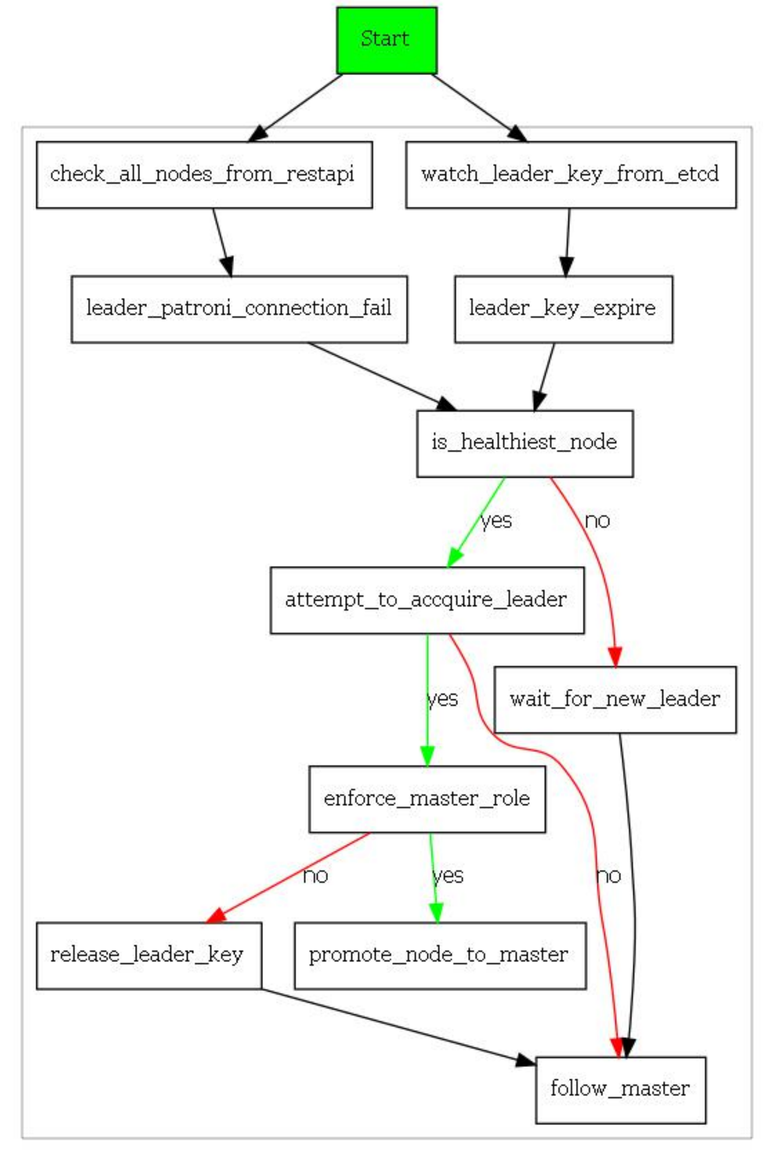
主机Patroni退出，自动切换



```
2021-11-06 21:09:13,309 INFO: Cluster(initialize=u'7027440434827163818', config=ClusterConfig(index=11, data={u'retry_timeout': 10, u'p
ostgresql': {u'use_slots': True, u'parameters': None, u'use_pg_rewind': True}, u'loop_wait': 10, u'maximum_lag_on_failover': 1048576, u
'synchronous_commit': u'on', u'ttl': 30}, modify_index=11), leader=Leader(index=291, session=None, member=Member(index=292, name='pg1',
 session=None, data={u'conn_url': u'postgres://192.168.0.111:5432/postgres', u'api_url': u'http://192.168.0.111:8008/patroni', u'timeli
ne': 1, u'state': u'running', u'version': u'2.1.1', u'role': u'master', u'xlog_location': 67109192})), last_lsn=67109192, members=[Memb
er(index=293, name='pg2', session=None, data={u'conn_url': u'postgres://192.168.0.112:5432/postgres', u'api_url': u'http://192.168.0.11
2:8008/patroni', u'timeline': 1, u'state': u'running', u'version': u'2.1.1', u'role': u'replica', u'xlog_location': 67109192}), Member(
index=294, name='pg0', session=None, data={u'conn_url': u'postgres://192.168.0.110:5432/postgres', u'api_url': u'http://192.168.0.110:8
008/patroni', u'timeline': 1, u'state': u'running', u'version': u'2.1.1', u'role': u'replica', u'xlog_location': 67109192}), Member(ind
ex=292, name='pg1', session=None, data={u'conn_url': u'postgres://192.168.0.111:5432/postgres', u'api_url': u'http://192.168.0.111:8008
/patroni', u'timeline': 1, u'state': u'running', u'version': u'2.1.1', u'role': u'master', u'xlog_location': 67109192})], failover=None
, sync=SyncState(index=None, leader=None, sync standby=None), history=None, slots=None)
```



```
2021-11-04 16:36:31,388 INFO: from dcs
2021-11-04 16:36:31,388 INFO: Cluster(initialize=None, config=Non
e, leader=None, last_lsn=None, members=[], failover=None, sync=No
ne, history=None, slots=None)
2021-11-04 16:36:31,388 INFO: touch member
2021-11-04 16:36:31,389 INFO: /service/pgcluster/members/pg2
2021-11-04 16:36:31,389 INFO: {"role":"uninitialized","state":"st
opped","version":"2.1.1","conn_url":"postgres://192.168.0.112:543
2/postgres","api_url":"http://192.168.0.112:8008/patroni"}
2021-11-04 16:36:31,393 INFO: Lock owner: None; I am pg2
```



谢谢