Рамблер/

RAMBLER&Co

PGConf.Russia 2018

Рамблер/

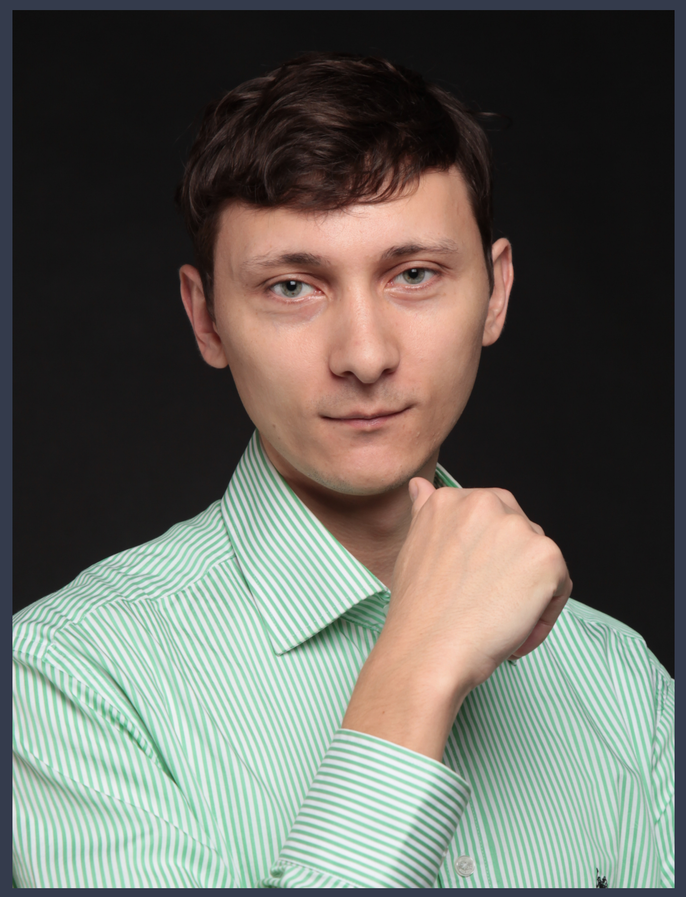
RAMBLER&Co

PostgreSQL SaaS в Rambler&Co

Отдел администрирования и развития инфраструктурных сервисов

Рамблер/

RAMBLER&Co



Игорь Успенский

Ведущий инженер отдела администрирования и развития инфраструктурных сервисов

Рамблер/

RAMBLER&Co

Рамблер/

RAMBLER&Co

Вступление

PostgreSQL SaaS в Rambler&Co

Рамблер/

RAMBLER&Co

Вступление

Rambler&Co - это множество изданий, сервисов и проектов. Появляются новые и растут существующие. Такой среде нужна надежная, отказоустойчивая, масштабируемая, автоматизированная система.

Взаимодействие и профиль нагрузки на PostgreSQL самые разные: web приложения, инфраструктурные элементы, мониторинг, роботы, аналитика.

В прежнем исполнении каждому кластеру были выделены и статично сконфигурированы собственные серверы, со слабой автоматизацией, и гетерогенной конфигурацией

Рамблер/

RAMBLER&Co

Вступление

Задача PostgreSQL as a Service

- Ускорение предоставления услуги базы данных
- Оптимизация утилизации ресурсов
- Унификация конфигураций
- Быстрый ввод функциональности
- Учет ресурсов

Рамблер/

RAMBLER&Co

Обзор

PostgreSQL SaaS в Rambler&Co

Рамблер/

RAMBLER&Co

PostgreSQL по модели SaaS состоит из следующих элементов:

- 3 Дата Центра
- Динамическая маршрутизация BGP
- Единая точка входа - один IP адрес

- Мониторинг и алертинг Prometheus
- Управление конфигурацией Salt Stack

- Кластер etcd
- Слой frontend
- Ноды баз данных с PostgreSQL
- Backup
- OS CentOS 7

Всего это около 100 машин и 300 баз данных

Рамблер/

RAMBLER&Co

Контуры

PostgreSQL SaaS в Rambler&Co

Рамблер/

RAMBLER&Co

Контуры

Каждый контур имеет логическое разделение между собой и отдельную инсталляцию

- Доступ к Production ограничен
- Синхронизация кластера между контурами

- Production
- Infrastructure
- Development
- Loadtesting

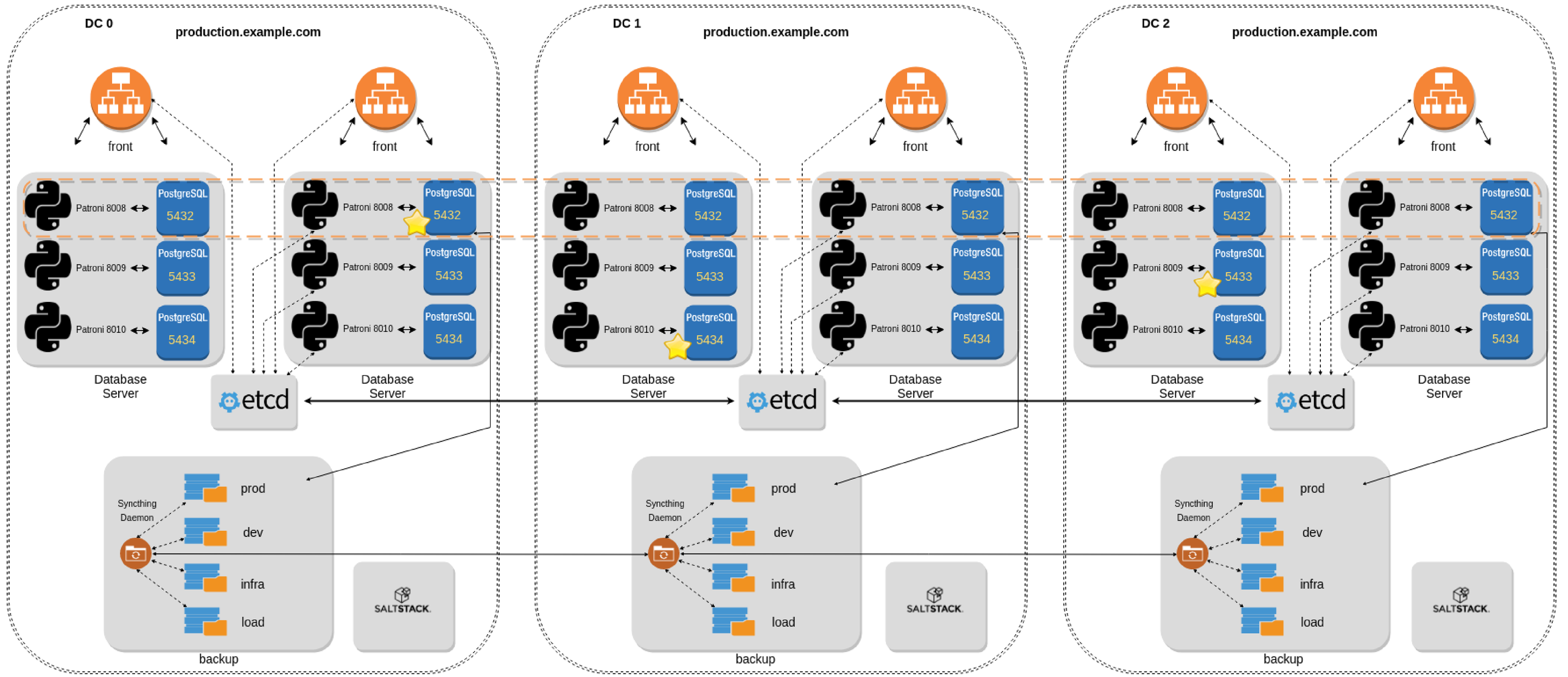


Схема контура

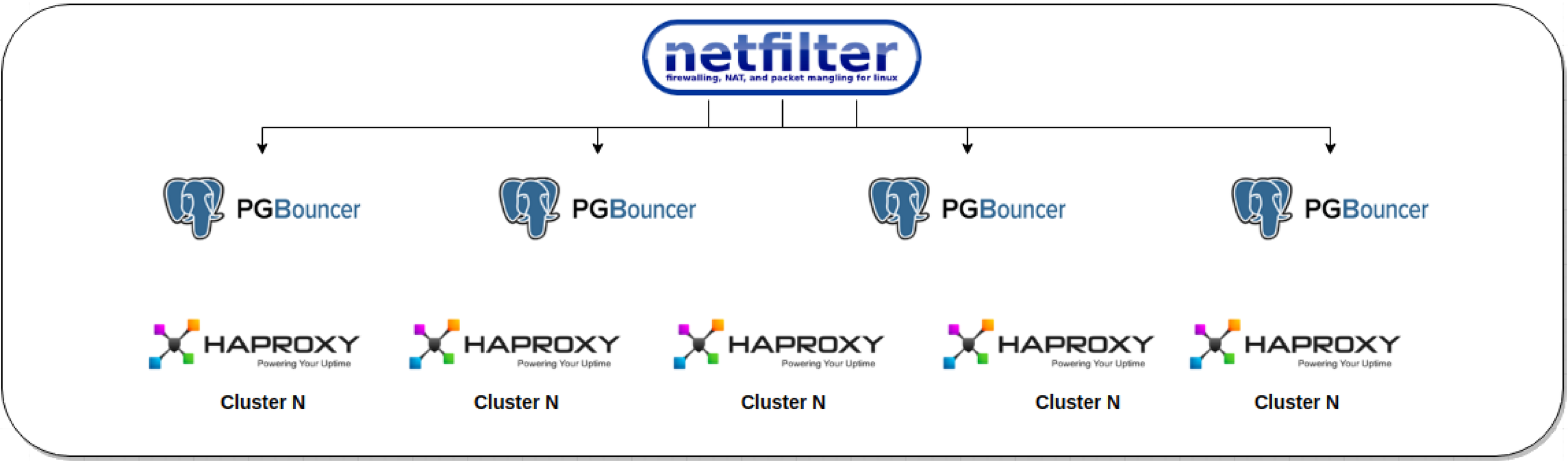


Схема frontend

Распределение соединений на pgBouncer

postgresql://production.example.com:6000/database_prod_**master**
postgresql://production.example.com:6000/database_prod_**slave**

Netfilter → pgBouncer → HAProxy

```
*NAT
-A PREROUTING -d 10.99.3.11/32 -p tcp --dport 6000 -m state --state NEW -j PGBOUNCER
-A PGBOUNCER -p tcp -m statistic --mode random --probability 0.25 -j DNAT --to-destination 127.0.0.1:6001
-A PGBOUNCER -p tcp -m statistic --mode random --probability 0.33 -j DNAT --to-destination 127.0.0.1:6002
-A PGBOUNCER -p tcp -m statistic --mode random --probability 0.50 -j DNAT --to-destination 127.0.0.1:6003
-A PGBOUNCER -p tcp -j DNAT --to-destination 127.0.0.1:6004
```

pgBouncer	pgBouncer	pgBouncer	pgBouncer
listen_addr = 127.0.0.1	listen_addr = 127.0.0.1	listen_addr = 127.0.0.1	listen_addr = 127.0.0.1
listen_port = 6001	listen_port = 6002	listen_port = 6003	listen_port = 6004

```
database_prod_master = host=127.0.0.2 port=5100 dbname=database
database_prod_slave =  host=127.0.0.2 port=5101 dbname=database
```

HAProxy instance per cluster

Systemd service haproxy@cluster_name

Netfilter → pgBouncer **➔ HAProxy**

Systemd unit haproxy@.service: ExecStart=/usr/sbin/haproxy-systemd-wrapper -f /etc/haproxy/conf.d/%i.cfg

cfg:

```
frontend master
    bind 127.0.0.2:5100

backend master
    option httpchk OPTIONS /master
    http-check expect status 200
    default-server on-marked-down shutdown-sessions
    server 2node0009 2node0009.db:5432 check port 8008 <--- DOWN
    server 1node0010 1node0010.db:5432 check port 8008 <--- UP
    server 0node0010 0node0010.db:5432 check port 8008 <--- DOWN
    server 0node0012 0node0012.db:5432 check port 8008 <--- DOWN
    server 1node0011 1node0011.db:5432 check port 8008 <--- DOWN
```

```
frontend replicas
    bind 127.0.0.2:5101

backend replicas
    option httpchk OPTIONS /replica
    http-check expect status 200
    default-server on-marked-down shutdown-sessions
    server 2node0009 2node0009.db:5432 check port 8008 <--- UP
    server 1node0010 1node0010.db:5432 check port 8008 <--- DOWN
    server 0node0010 0node0010.db:5432 check port 8008 <--- UP
    server 0node0012 0node0012.db:5432 check port 8008 <--- UP
    server 1node0011 1node0011.db:5432 check port 8008 <--- UP
```

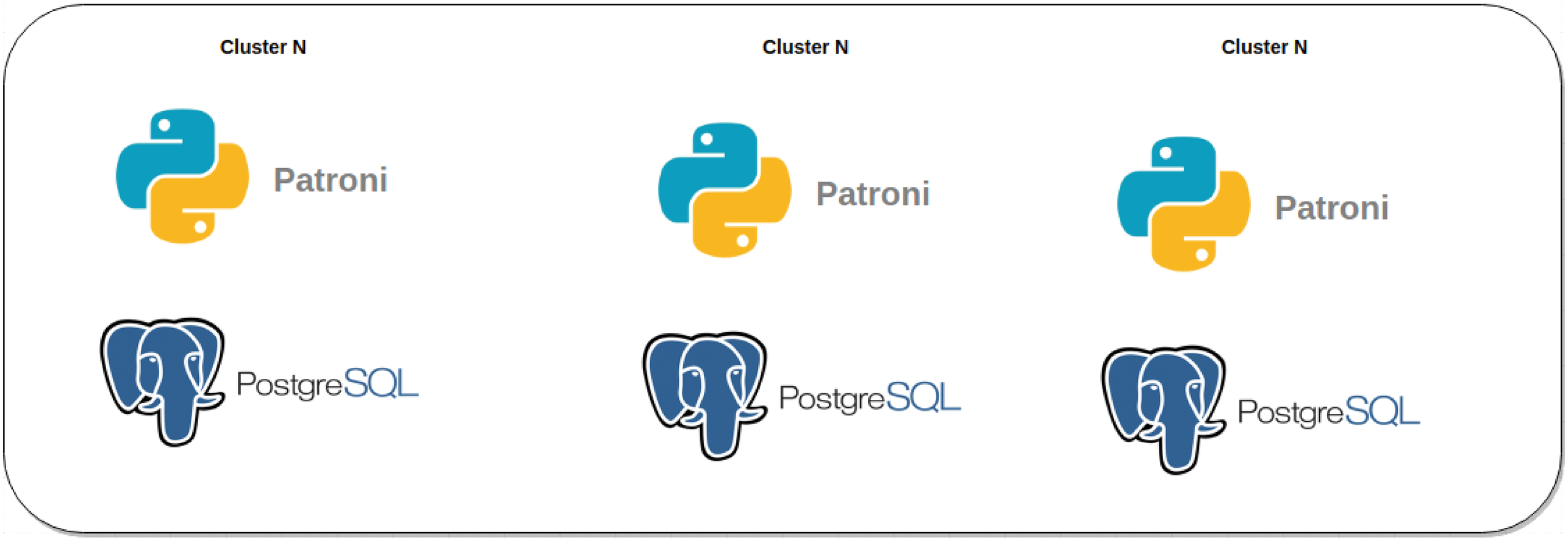


Схема database node

Patroni

Systemd service patroni@cluster_name

HAProxy → **PostgreSQL (Patroni)**

Systemd unit patroni@.service: ExecStart=/usr/bin/patroni /etc/patroni.d/%i.yml

yml:

```
scope: cluster_name
namespace: /postgresql/
name: 1node0010_5432
restapi:
  listen: 0.0.0.0:8008
  connect_address: 1node0010.db:8008
etcd:
  srv: production.example.com
```

```
postgresql:
  listen: 0.0.0.0:5432
  connect_address: 1node0010.db:5432
  data_dir: /data/pg/cluster_name
  bin_dir: /usr/pgsql-9.6/bin
  create_replica_method:
   - backup
   - basebackup
  backup:
   command: /usr/local/libexec/patroni_restore
```

Рамблер/

RAMBLER&Co

Контуры

Создание реплики без нагрузки на кластер

PostgreSQL SaaS в Rambler&Co

Рамблер/

RAMBLER&Co

Логика создания реплики из backup:

1. Подключиться к мастеру - pg_receivexlog
2. pg_control_checkpoint():
Когда checkpoint_time будет после начала pg_receivexlog
получить XID и WAL (oldest_active_xid - 1)
3. Восстановить backup
4. Запустить PostgreSQL с recovery_target_xid=XID при наличии необходимого WAL в архиве
5. Восстановить архив
6. Остановить PostgreSQL и выйти с кодом 0

Рамблер/

RAMBLER&Co

Контуры

Актуализация кластера между контурами

PostgreSQL SaaS в Rambler&Co

Рамблер/

RAMBLER&Co

Актуализация кластера для разработки:

- Восстановление одной (и более) бд из кластера
- Восстановление кластера большого объема
- Обезличивание данных

- pgbackrest --db-include (Выбирает по datid)
- pgbackrest --delta (SHA-1)
- триггеры в своем сценарии актуализации

Рамблер/

RAMBLER&Co

Контуры

Управление конфигурацией

PostgreSQL SaaS в Rambler&Co

Управление конфигурацией — SaltStack

```
patroni-prod:
  cluster:
    prod01:
      0node0004.db:
        port: 5433
        rest: 8009
      1node0004.db:
        port: 5433
        rest: 8009
      2node0004.db:
        port: 5433
        rest: 8009
```

```
patroni-dev:
  cluster:
    database_preprod:
      2node0006.db:
        port: 5434
        rest: 8010
        restore_from_env: prod
        restore_from_env_cluster: prod01
        restore_from_env_cluster_db: database
      1node0007.db:
        port: 5434
        rest: 8010
        restore_from_env: prod
        restore_from_env_cluster: prod01
        restore_from_env_cluster_db: database
```

Рамблер/

RAMBLER&Co

Контуры

Горизонтальное масштабирование и перенос кластера

PostgreSQL SaaS в Rambler&Co

Рамблер/

RAMBLER&Co

Горизонтальное масштабирование и перенос кластера

Масштабирование:

- Запустить instance на нодах
- pg_prewarm если нужно (patroni callback)
- confd конфигурирует HAProxy

Перенос:

- Выполнить масштабирование
- Переключить master

Рамблер/

RAMBLER&Co

Мониторинг

PostgreSQL SaaS в Rambler&Co

Рамблер/

RAMBLER&Co

Мониторинг состоит из следующих элементов:

- Prometheus
- Prometheus Alertmanager
- Grafana
- prometheus exporters
- pghero (pg_stat_statements)

RAMBLER&Co

Каждому проекту предоставляется Dashboard. В нем есть **панель статуса** для быстрого получения общей картины состояния кластера.

Мониторинг sql запросов с течением времени из pg_stat_statements.

И около **60 графиков**.


Average query time
1.3 min
1.2 min
1.0 min
50 s
40 s
30 s
20 s
10 s
0 ms
12:10 12:20 12:30 12:40 12:50 13:00 13:10 13:20 13:30 13:40 13:50 14:00 14:10 14:20 14:30 14:40 14:50 15:00 15:10 15:20 15:30 15:40 15:50 16:00 16:10 16:20 16:30 16:40 16:50 17:00 17:10 17:20 17:30 17:40 17:50 18:00
max current
SELECT 1.166 min 35.7 s
BEGIN; 43.0 s
INSERT 42.0 s
SELECT 10.5 s 4.9 s
UPDATE 9.5 s
SELECT 9.1 s 1.6 s
INSERT 8.8 s
UPDATE 7.6 s 4.7 s
SELECT 5.5 s 2.6 s
SELECT 5.0 s 1.9 s
SELECT 4.9 s 1.5 s
SELECT 4.8 s 833 ms
SELECT 3.9 s 1.9 s
UPDATE 3.4 s
INSERT 2.8 s 321 ms
SELECT 2.4 s 1.2 s
SELECT 1.8 s 903 ms
SELECT 1.8 s 160 ms
SELECT 1.6 s 512 ms
SELECT 1.5 s 975 ms
SELECT 1.3 s 223 ms
SELECT 1.3 s 313 ms

Queries

Reset

Total Time	Average Time	Calls
2,371 min 55%	1,562 ms	91,084

```
SELECT
        c.id, c.html, c.status, c."hasLink", c."hasOrngWord", c.text, c.moderation,
c.rating,
        c."textUpdatedAt", c."createdAt", c."deletedAt", c."ParentId" AS
"parentId", c."UrlId",
        c.attachments, c."sessionSourceIcon",
        u.xid, u.permaLink,
        pu.id "parentUserId", pu."DisplayName" "parentUsername"

        FROM "Comments" c
          JOIN "Urls" u ON u.id = c."UrlId" AND u."AppId" = c."AppId"
          LEFT JOIN "Comments" pc ON pc.id = c."ParentId"
```

1,008 min 23%	1 ms	55,672,497

```
SELECT "count", "xid" FROM "Urls" AS "Url" WHERE "Url"."AppId" = ? AND
"Url"."xid" IN (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);
```

299 min 7%	3 ms	5,566,637

```
SELECT
        c.id, c.html, c.status, c."hasLink", c."hasOrngWord", c.text,
c.moderation, c.rating,         c."textUpdatedAt", c."createdAt", c."deletedAt",
c."ParentId" AS "parentId",
        c.attachments, c."sessionSourceIcon",
        u.id AS "userId", u.userpic, u."DisplayName", u.username, u."xpUserId",
u.xid, u.url AS "userUrl"
        FROM "Comments" c
        LEFT OUTER JOIN "Users" u ON c."UserId"=u.id
        WHERE c."UrlId"=?
        ORDER BY c."createdAt" ASC LIMIT ?;
```

RAMBLER&Co

Мониторинг sql запросов

pg_stat_statements

Explain

```
SELECT
    c.id, c.html, c.states, c."hasLink", c."hasGreyWord", c.text, c.moderation, c.rating,    c."levtUpdatedAt",
c."createdAt", c."deletedAt", c."ParentId" AS "parentId",
    c.attachments, c."versionSaveAction",
    u.id AS "userId", u.userpic, u."displayName", u.username, u."ramblerId", u.sid, u.url AS "userUrl"
    FROM "Comments" c
    LEFT OUTER JOIN "Users" u ON c."UserId"=u.id
    WHERE c."UrlId"=342341
    ORDER BY c."createdAt" ASC LIMIT 1;
```

Explain | Analyze | Visualize

```
Limit  (cost=1016.25..1016.25 rows=1 width=1194) (actual time=92.865..92.865
rows=1 loops=1)
  ->  Sort  (cost=1016.25..1016.46 rows=85 width=1194) (actual
time=92.865..92.865 rows=1 loops=1)
        Sort Key: c."createdAt"
        Sort Method: top-N heapsort  Memory: 26kB
        ->  Nested Loop Left Join  (cost=0.99..1015.82 rows=85 width=1194)
(actual time=44.711..92.844 rows=3 loops=1)
              ->  Index Scan using comments_url_id on "Comments" c
(cost=0.56..297.15 rows=85 width=502) (actual time=44.679..92.762 rows=3 loops=1)
                    Index Cond: ("UrlId" = 342341)
              ->  Index Scan using "Users_pkey" on "Users" u  (cost=0.43..8.45
rows=1 width=696) (actual time=0.020..0.021 rows=1 loops=3)
                    Index Cond: (c."UserId" = id)
Planning time: 2.372 ms
Execution time: 92.928 ms
```

RAMBLER&Co

Также есть возможность произвести EXPLAIN

Рамблер/

RAMBLER&Co

Итог

PostgreSQL SaaS в Rambler&Co

Рамблер/

RAMBLER&Co

Итог

Клиент получает:

- Единую точку входа
- Актуальный dev, stage, preprod для разработки
- Мониторинг и оповещения по своему проекту

Система имеет:

- Отказоустойчивость
- Масштабирование на чтение
- Непрерывный backup

Также во всех ДЦ есть мощные hot swap ноды. Их задействуют при непредвиденных ситуациях, например, резкое увеличение нагрузки на отдельный кластер.

Рамблер/

RAMBLER&Co

Итог

К чему стремимся

- Автоматизация предоставления ресурсов
- Распределение по нодам с учетом утилизации их компонент
- Автоматический поиск bottleneck с рекомендациями по устранению
- Оптимизация конфигурации представления в SCM (SaltStack)
- Перевод хранения backup в собственный S3 storage

Рамблер/

RAMBLER&Co

Links

- https://www.postgresql.org
- https://pgbouncer.github.io
- http://www.haproxy.org
- https://coreos.com/etcd
- http://www.confd.io
- https://prometheus.io
- https://saltstack.com/community
- https://github.com/zalando/patroni
- https://github.com/pgbackrest/pgbackrest
- https://github.com/syncthing/syncthing
- https://github.com/ankane/pghero
- https://github.com/prometheus/haproxy_exporter
- https://github.com/wrouesnel/postgres_exporter
- http://git.cbaines.net/prometheus-pgbouncer-exporter

Рамблер/

RAMBLER&Co

Обратная связь

Если у вас есть предложение, отзыв или вы хотите поделиться собственными наработками по теме — пишите

saas-admin@rambler-co.ru