Highway to Hell
or Stairway to Cloud?

Percona Live 2018,
Frankfurt

ALEXANDER KUKUSHKIN
06-11-2018
ABOUT ME

Alexander Kukushkin

Database Engineer @ZalandoTech

The Patroni guy

alexander.kukushkin@zalando.de

Twitter: @cyberdemn
WE BRING FASHION TO PEOPLE IN 17 COUNTRIES

17 markets
7 fulfillment centers
23 million active customers
4.5 billion € net sales 2017
200 million visits per month
15,000 employees in Europe
FACTS & FIGURES

> 300 databases on premise

> 650 clusters in the Cloud (AWS)
AGENDA

About the old setup
Choosing your cloud options
Retain access & make it secure
Data migration & switchover
Backup & recovery
Conclusions
The old setup

- Provisioned in 2015
- DELL PowerEdge R730xd
- 2 * Intel Xeon E5-2667v3 (16 cores)
- 256 GB RAM
- 14 * 1.5 TB SSD in raid10 (10.5 TB)
- Network: 2 * 10 GBit/s
- PostgreSQL 9.6
Under the hood

- 3000 tables
  - two tables per event
    - Hot data (last 10 days)
    - Archived data
  - No primary/unique keys!
- About 100 millions inserts/day
- Size (before the migration): 10 TB
- Avg growth 2 TB per year
Free space: 500 GB

Upgrade or migrate?
Migrate it!

● Minimize costs (cloud isn’t cheap)
● How to switch back to the data center if something goes wrong?
● How to retain access through the old connection url?
● Make it secure
● Minimal downtime
About the old setup

Choosing your cloud options

Retain access & make it secure

Data migration & switchover

Backup & recovery

Conclusions
Candidates

- Amazon Aurora
- DIY
  - i3 instances
  - EBS backed instances
    - gp2
    - io1
Amazon Aurora

**PROS**

- AWS promise decent performance
- Storage auto-scaling
  - All instances are sharing the same storage!
- Price for storage is the same as for gp2 EBS, **$0.119/GB-month**

**CONS**

- $0.22 per 1 million I/O requests.
- `plproxy` extension is not available
i3 instances

**PROS**

- Local NVMe volumes:
  - low latency
  - high bandwidth and throughput
- Low storage price
- 488 GB RAM

**CONS**

- Ephemeral volumes
  - Minimum 3 instances for HA
- The biggest instance has “only” 15TB
EBS backed instances (m4/r4)

**PROS**

- Data on EBS survives instance restart
- Easy to scale up or down
- Makes it possible to run only two instances

**CONS**

- I/O latencies
- Limited IOPS and bandwidth per volume:
  - **gp2**: 160 MB/s, 10000 IOPS
  - **io1**: 500 MB/s, 32000 IOPS
- Price per GB (comparing with i3)
gp2 vs io1

EBS, USD for 10 TB

10000 IOPS

- gp2: 1190 USD
- io1: 1490 USD

2270 USD

30000 IOPS

- gp2: 1190 USD
- io1: 1490 USD

3830 USD

- IOPS
- Storage

[zalando logo]
Do benchmarks

● Cloud makes it very easy to conduct experiments

● Apply the load similar to production
  ○ Ideally, replicate production workload

● Use **Spot** instances to make it cheaper
It’s all about the money (and risks)
The cloud setup

- r4.8xlarge
  - 32 vCPU cores
  - 244 GB RAM
  - 37500 IOPS
  - 875 MB/s

- 20 TB EBS gp2
  - 6 * 3333 GB, raid 0
Data migration & switchover

About the old setup

Choosing your cloud options

Retain access & make it secure

Data migration & switchover

Backup & recovery

Conclusions
How to retain access via old conn_url?

- Possible options:
  - DNS
  - “Proxy” (iptables/HAProxy/pgbouncer)

- Think about security:
  - Internet traffic **MUST** be encrypted!
  - Some of the legacy applications are not using **SSL**
    - Nobody wants to fix legacy code :(  
  - How to protect from Man-in-the-Middle attack?
Pgbouncer to the rescue

- primary pgbouncer
  - primary
  - data center
  - vip
- replica pgbouncer
  - replica
  - vip

SSL 5432

Primary

Replica

Cluster Security Group

AWS
Make it secure

- Setup CA
- Generate server and client keys
- Sign server and client certs with the CA private key
- Postgres must validate the client certificate from pgbouncer
- Pgbouncer must validate the Postgres server certificate
Postgres configuration

- **postgresql.conf**
  - `ssl_cert_file = 'server.crt'`
  - `ssl_key_file = 'server.key'`
  - `ssl_ca_file = 'ca.crt'`

- **pg_hba.conf**
  - `hostssl all all A.B.C.D/32 md5 clientcert=1`
  - `hostnossl all all A.B.C.D/32 reject`
Pgbouncer configuration

- Configure pgbouncer (in the data center)
  - pool_mode = session
  - auth_file = users.conf
  - auth_query = “SELECT * FROM pgbouncer.user_lookup($1)”
  - server_tls_sslmode = verify-ca
  - server_tls_ca_file = ca.crt
  - server_tls_cert_file = client.crt
  - server_tls_key_file = client.key
Data migration & switchover

Backup & recovery

Conclusions
Possible options

- pg_basebackup + physical replication
  - via VPN?
  - via SSH tunnel?

- S3 compatible backup tool
  - WAL-E
  - pgBackRest
  - WAL-G
Keep it Simple

Cluster Security Group

- Primary
- Replica
- Replica
- Replica

Data center

- app1
- app2
- app3

VIP

VIP

VIP

VIP

AWS

etcd

S3 bucket: Backup + WAL

Cluster Security Group

wal-e

wal-e

wal-e
Migration statistics

- “wal-e backup-push” in the DC: 12 hours
- “wal-e backup-fetch” on AWS: 9 hours
- Replay accumulated WAL: 4 hours

replication lag in such setup is usually about a few seconds and determined by amount of write activity on the primary.
Switchover plan

1. Shutdown the main application writing into DB
2. Move the replica virtual ip to the pgbouncer host
3. Shutdown the replica in the data center
4. Move the primary virtual IP to the pgbouncer host
5. Shutdown the primary in the data center
6. Promote replica in the Cloud
7. Start the main application
8. Start replicas in the data center with the new recovery.conf
Before the switchover

Cluster Security Group

Before the switchover to AWS, applications app1, app2, and app3 are running in a data center. The Primary data center includes a primary pgbouncer and a replica pgbouncer. The VIPs are connected to these components.

The S3 bucket is used for Backup + WAL, and wal-e is used to synchronize the data between the Primary and Replica data centers.

After the switchover, the Replica data center is connected to the etcd and AWS services. The VIPs are also connected to the Replica components, ensuring the smooth transition of the applications and data.
Move the replica VIP

- data center
- app1
- app2
- app3
- Primary
  - vip
  - primary pgbouncer
  - vip
  - replica pgbouncer
  - Replica
- etcd
- AWS
- Replica
  - S3 bucket: Backup + WAL
  - wal-e
- wal-e
- Cluster Security Group

Please write the title in all capital letters. Use bullet points to summarize information rather than writing long paragraphs in the text box.
Shut down the replica

- Primary
- Data center
- VIP
- Primary pgbouncer
- App 1
- App 2
- App 3
- VIP
- Replica
- Replica
- SSL
- Etcld
- AWS
- S3 bucket: Backup + WAL
- Replica
- Replica
- Cluster Security Group
Move the primary VIP

- data center
  - Primary
    - vip
    - primary pgbouncer
  - vip
    - replica pgbouncer
- app1
- app2
- app3

- Replica
  - etcd
  - S3 bucket: Backup + WAL
- Replica

Cluster Security Group
Shutdown the primary

- data center
- vip
  - primary pgbouncer
  - replica pgbouncer
- app1
- app2
- app3
- S3 bucket: Backup + WAL
- etcd
- SSL
- Replica
- Replica
- vip
  - SSL
  - wal-e
  - wal-e
- Cluster Security Group

- Cluster Security Group
Promote the replica on AWS

- **Cluster Security Group**
- **Promote the replica on AWS**
- **Primary**
- **Replica**
- **SSL**
- **wal-e**
- **S3 bucket: Backup + WAL**
- **etcd**
- **AWS**
- **data center**
- **app1**
- **app2**
- **app3**
- **etcd**
- **SSL**
- **VIP**
- **pgbouncer**
- **primary**
- **replica**
- **pgbouncer**
Start replicas in the data center

Cluster Security Group

- Start replicas in the data center

- Use bullet points to summarize information rather than writing long paragraphs.
About the old setup
Choosing your cloud options
Retain access & make it secure
Data migration & switchover
Backup & recovery
Conclusions
S3 compatible backup tools

- **WAL-E** is our primary backup tool in the cloud
  - is too slow on big volumes of data :(
  - can’t take basebackup from the replica :(

- **pgBackRest**
  - incremental & differential backups
  - can’t use AWS instance profile credentials :(

- **WAL-G**
  - delta backups
  - configurable compression methods: `lz4`, `lzma`, `zstd`, `brotli`
  - backward compatible with **WAL-E**
WAL-E vs WAL-G on r4.8xlarge

- WAL-G full, brotli: 2.4 TB, 0.075 Hr
- WAL-G delta, brotli: 2.4 TB, 0.075 Hr
- WAL-E: 17 TB, 3.2 Hr
- Baseline: 12 TB, 4 Hr
After the migration

Keep an eye on monitoring!!!
Switchover

synchronous_commit = ‘off’
Links

● Patroni: https://github.com/zalando/patroni
● WAL-E: https://github.com/wal-e/wal-e/
● WAL-G: https://github.com/wal-g/wal-g/
● pgBackRest: https://pgbackrest.org/
● pgbouncer: https://pgbouncer.github.io/
● Easy Amazon EC2 Instance Comparison: EC2instances.info
Thank you!