Multi-Active with Spock

April 2023

Jan Wieck

- Contributor since 1996 (fixing the rewrite rule system)
- Guilty of implementing TOAST
- Guilty of creating loadable, procedural languages
- Guilty of PL/Tcl and PL/pgSQL
- Guilty of PG's implementation of foreign keys
- Guilty of the NUMERIC data type
- Guilty of Slony-I
- Guilty of the background writer, the statistics collector, lazy vacuum
- Core team member from 2000 to 2010

Why pgEdge

- Distributed systems need distributed databases
  - Latency between distributed client and central application

    or
  - Latency between distributed application and central database

Why Logical Replication

- Cross PG Version
- Table by Table
- Row Level Filtering
- More flexible standby with read/write tables
- Uses logical decoding
- Just row values, no vacuum, indexes, etc
- Allows for near zero time upgrades between major versions

pgLogical / BDR

- Does not support multi-active, but has some of the plumbing
- Has been in maintenance mode for the last 4+ years, no major new features
- 2ndQuadrant and now EnterpriseDB have a proprietary multi-active solution
- pgLogical will not compete with their proprietary solution
- Features introduced in pgLogical have been incrementally added to core postgres
- pg9.6 took a stab at bidirectional replication, BDR1 never maintained

Spock

Spock introduces multi-active replication with conflict resolution and avoidance

- Support for Asynchronous Multi-Active Replication
- Conflict-free Delta-Apply Columns
- Conflict Resolution with better error handling
- Better management & monitoring stats and integration
- Performance, stability and networking stress testing
- Replication of Partitioned Tables (to help robustly support Geo-Sharding)
- Link database to a country of residence, configurable rules for keeping PII in country

Conflict Resolution

In case the node is subscribed to multiple providers, or when local writes happen on a subscriber, conflicts can arise for the incoming changes. These are automatically detected and can be acted on depending on the configuration.

- last_update_wins - alway keep the changes with the newest commit timestamp
- first_update_wins - alway keep the changes with the oldest commit timestamp
- apply_remote - ignore local and always keep remote
- keep_local - ignore remote and always keep local

Spock is optimized for a mostly non conflicting workload, if a conflict does arise monitoring and diagnostic are provided in a resolutions table

Conflict Avoidance

Conflict-Free Delta-Apply

- Logical Multi-Active replication can get itself into trouble on running sums
- Unlike other solutions, we do NOT have special data types for this
- Any numeric data type can be used (including numeric, float, double precision, int4, int8, etc).

Conflict Avoidance

Conflict-Free Delta-Apply

- Small diff patch to postgreSQL Core
- Adds functionality to `ALTER TABLE t COLUMN c SET (log_old_value = True)`
- Patch will be submitted to Community and discussed at Ottowa Conference

Conflict Avoidance

Conflict-Free Delta-Apply

- The old and new values for that column is captured to the WAL
- During apply the new value for the column is calculated as

```
new-local = old-local + (new-remote - old-remote)
```

Conflict Avoidance - Example

Initially, bal is equal to 300 for id = 1

Node 1	Node 2

`BEGIN;`
`UPDATE t1 SET bal=310`
`  WHERE id=1;`	

	`BEGIN;`
	`UPDATE t1 SET bal=305`
	`  WHERE id=1;`
	`COMMIT;`

`COMMIT;`	

At this point, bal is equal to 315 for id = 1
Even if the application does the math, Conflict-Free Delta-Apply still will produce the correct results

Spock - What's Critical

- Better node failure resilience without down time (3+ node clusters)
  - Currently intermittent outages are handled nicely
  - Recovering from catastrophic node loss without missing or duplicate transactions
  - Spinning up full replacement nodes

pgEdge Platform

- All features of pgEdge Distributed PostgreSQL
- Self-host on-premises or in cloud accounts
- For developer evaluations or production usage
- Enterprise support available

pgEdge nodectl spock

Logical and Multi-Active PostgreSQL node configuration

Synopsis

```
./nodectl spock <command> [parameters] [options]
```

install - Install PG and configure with SPOCK extension
validate - Check pre-req's for advanced commands
tune - Tune for this configuration
node-create - Name this spock node
repset-create - Define a replication set
sub-create - Create a subscription
repset-add-table - Add table to replication set
sub-add-respset - Add replication set to subscription
sub-show-status - Display status of subcription
sub-show-table - Display subscription table(s)
sub-wait-for-sync - Pause until subscription synched
health-check - Check if PG is accepting connections
metrics-check - Retrieve advanced DB & OS metrics

PGEDGE NODECTL CLUSTER CONTROLLER

Installation and configuration of a pgEdge SPOCK cluster

Synopsis

```
./nodectl spock <command> [parameters] [options]
```

create-local - Create local cluster of N pgEdge nodes on different ports
destroy - Stop and then nuke a cluster
validate - Validate a remote cluster configuration
init - Initialize a remote cluster for SPOCK
command - Run `nodectl` command on one or all nodes of a cluster
diff-tables - Compare table on different cluster nodes

Key Value Propositions

Low Latency

Ultra-high Availability

Data Residency

pgEdge Community License

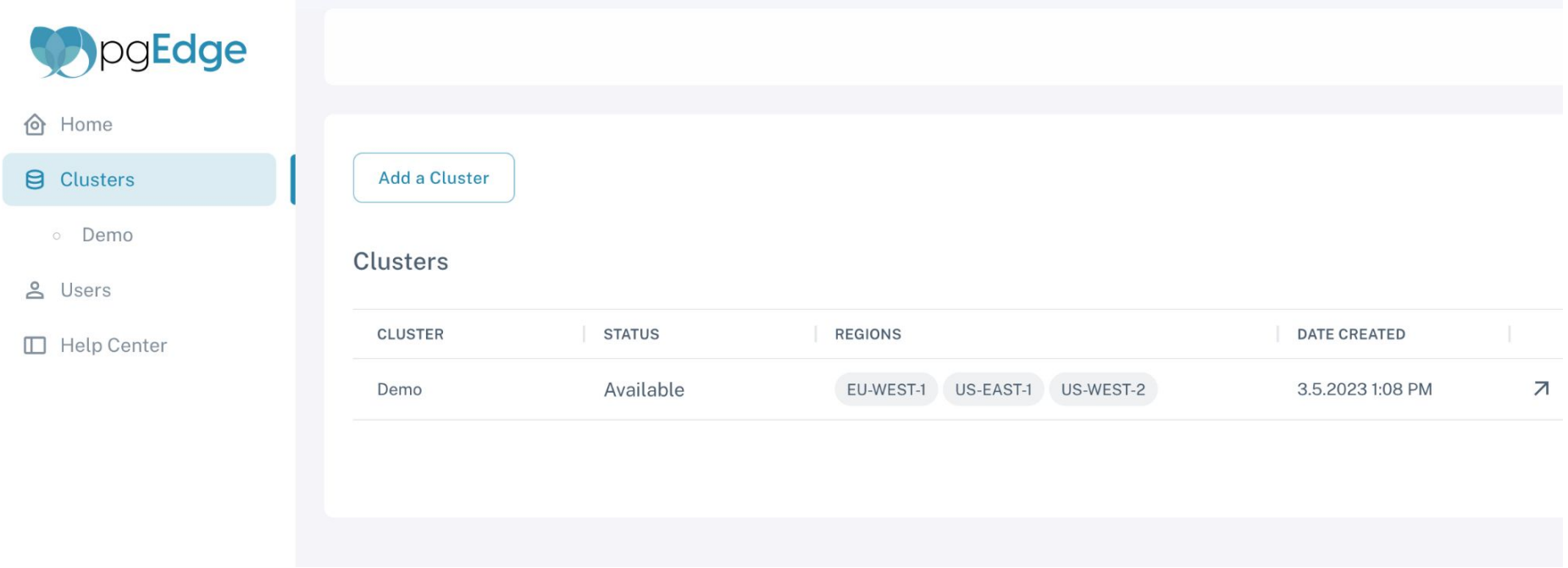
- Spock is open and pgEdge Community Licensed
- Our license allows unlimited end user usage (including in production)
- Essentially the same as Confluent Community License
- Prohibits packaging and selling a competitive product

pgEdge Cloud

- Fully managed Database-as-a-Service (DBaaS)
- Handles provisioning, security and monitoring
- Access via web dashboard, CLI and API
- AWS and Azure (GCP coming soon)

pgEdge Create Cluster

pgEdge List Clusters



pgEdge Monitor Cluster

pgEdge Security

- Bring your own account
- VPC Peering- Automatically creating global network private to your account
- Built with Infrastructure as Code following cloud provider best practices
- Secure and auditable

pgEdge in Summary

- Low Latency
- Ultra-high Availability
- Data Residency
- Optimized for the network edge
- Fully Managed Database-as-a-service
- Access via Web Dashboard, API, and CLI

Thank You!