CRUNCHY DATA SOLUTIONS, INC.

Audit Logging with PostgreSQL

David Steele

PGCon 2016
May 20th, 2016

About the Speaker

- Senior Data Architect at Crunchy Data Solutions, the PostgreSQL company for secure enterprises.
- Actively developing with PostgreSQL since 1999.

2ndQuadrant’s Contribution

- The pgAudit extension presented here is based on work done by Simon Riggs, Ian Barwick, and Abhijit Menon-Sen at 2ndQuadrant sponsored by the AXLE project.
- It was forked and modified considerably in order to address concerns raised by the PostgreSQL core community during the 9.5 release cycle.
- Crunchy Data is working with 2ndQuadrant to move the plaudit project forward.

Agenda

- What is Audit Logging
- Why Audit Log
- How to Audit Log
- pgAudit Design
- Examples
- Demo

What is Audit Logging

- An audit is an official inspection of an individual's or organization's accounts, typically by an independent body.
- The information gathered by the PostgreSQL Audit extension (pgAudit) is properly called an audit trail or audit log.
- The pgAudit extension provides detailed session and/or object audit logging via the standard PostgreSQL logging facility.

Why Audit Log

- The goal of the PostgreSQL Audit extension (pgAudit) is to provide PostgreSQL users with capability to produce audit logs often required to comply with government, financial, or ISO certifications.
- Organizations may also have internal requirements that can be satisfied with pgAudit.
- Can also be used for detailed debugging, metrics, and monitoring.

How to Audit Log

- Triggers
  - Won’t do SELECTs
  - Event triggers can be used for most DDL (improved in 9.5) but not ROLE commands
- Functions
  - All inserts, selects, updates, etc. are done through functions
- log_statement = all
  - Catches all client statements
  - Is very hard to parse and can miss nuances that might not be obvious.
  - No way to filter - it’s the proverbial firehose

How to Audit Log (pgAudit)

- More granular logging
  - Multiple logging classes: READ, WRITE, FUNCTION, ROLE, DDL, MISC
- Object logging
  - Grants system can be used to give fine control over logging of SELECT, INSERT, UPDATE, and DELETE on relations
- More detail in audit logs
  - Log records contain the command, object type, fully-qualified object name, stack depth, statement, parameters, etc.

pgAudit Design (and Caveats)

- Implemented as a standard PostgreSQL extension.
- Uses various hooks to audit statements executed by users.
- May log statements that eventually raise an exception.
- Does not log statements that contain syntax errors (though these will be caught by log_statement = error). The same is true for statements attempted while a transaction is in aborted state.

Example (log_statement = all)

- User statement:

```
DO $$
BEGIN
    EXECUTE 'CREATE TABLE import' || 'ant_table (id INT)';
END $$;
```

- What gets logged:

```
LOG:  statement: DO $$
BEGIN
    EXECUTE 'CREATE TABLE import' || 'ant_table (id INT)';
END $$;
```

Example (pgAudit)

- User statement:

```
DO $$
BEGIN
    EXECUTE 'CREATE TABLE import' || 'ant_table (id INT)';
END $$;
```

- What gets logged:

```
AUDIT: SESSION,33,1,FUNCTION,DO,,,"DO $$
BEGIN
    EXECUTE 'CREATE TABLE import' || 'ant_table (id INT)';
END $$;"
AUDIT: SESSION,33,2,DDL,CREATE
TABLE,TABLE,public.important_table,CREATE TABLE important_table (id
INT)
```

Life at one of the big four audit firms.

Demo Time!

- Live Demo, this will be fun…

Thank You! Questions?

email: david@crunchydata.com

github page: https://github.com/pgaudit/pgaudit

slides & demo: https://github.com/dwsteele/conference/releases