```
whoami
```

- 25+ years on IT
- PostgreSQL Developer at OnGres
- PostgreSQL Contributor
- Brazilian Community Leader

Fabrízio de Royes Mello

<fabrizio@ongres.com>

@fabriziomello
`whoami`

- Founder & CEO, OnGres
- 20+ years Postgres user and DBA
- Mostly doing R&D to create new, innovative software on Postgres
- Frequent speaker at Postgres, database conferences
- Founder and President of the NPO Fundación PostgreSQL
- AWS Data Hero

Álvaro Hernández
<aht@ongres.com>

aht.es
ENHANCING POSTGRES OBSERVABILITY
Postgres monitoring

- Not in-core integrated solution.
- Postgres provides catalog views with rich monitoring information.
- But that means **making queries** to gather monitoring data.
- Eg. Prometheus exporter: dozens/hundreds/... of queries per monitoring cycle.
- Postgres monitoring extensions:
  - may require restart -> downtime.
  - may require configuration / external binaries-> complexity

Can we do better?
Postgres wire protocol (FeBe)

- Custom (layer 7) TCP protocol. Well documented.
- Well structured and defined messages (no "generic message" for many things).
- Very stable (current v3 was introduced in 2003 with PG 7.4).
- Implemented by countless tools and drivers.
- Used also for non-Postgres databases (Yugabyte, CockroachDB, Crate.io, NoisePage, ...).
FeBe protocol architecture
Idea: proxy and decode the protocol to get metrics!
Advantages of metrics via proxying/decoding

- s/pull/push/
- **Zero impact on the database.** 100% transparent.
  - No performance impact.
  - No configuration required. No agents/tools to install.
- Can be deployed as a sidecar (eg. via injection in K8s).
- **May significantly increase the volume of metrics obtained.**
- Opens the door for other added functionality.
Envoy extensibility: simple connection model

Application (client)

tcp_proxy

POSTGRES NETWORK FILTER FOR ENVOY PROXY
Envoy extensibility: filter chain

Application (client) → tcp_proxy → PostgreSQL
PostgreSQL Filter Architecture: metadata

Application (client)
How it all started

[GitHub Issue #9107](https://github.com/envoyproxy/envoy/issues/9107)

**Background**

It is becoming more and more important to have detailed insights into the performance of PostgreSQL databases. In particular, query performance is a hot topic. But it is not easy or feasible on all situations to get metrics about query performance.

The usual way is to either:

- Log all queries by setting parameters like `log_min_duration_statement` to 0 or `log_statement` to `read` or `all`. This approach may create a logging storm, which can significantly affect the workload. Plus it still requires parsing the logs, which is a complicated task (as log entries are complex and can be multi-line).

- Use Postgres extensions like `pg_stat_statements`. They capture query performance, but only on an aggregated fashion, and reading them requires querying Postgres (pull mode), versus metrics being pushed to the monitoring system. Plus the performed aggregation makes it very difficult to compute relevant performance metrics such as query latency percentiles. And it relies on the availability of the extension, and the user has to explicitly install it – it is not enabled by default, and turning it on requires a database restart.

**Design Goals**

[Learn more on UNGRES](https://ungres.com/postgres-network-filter-for-envoy-proxy)
An effort developed by a great Community

- Contributions from Tetrade, Envoy Maintainers, OnGres and others.
- Merged and released with Envoy 1.15.
- Led to 10 issues and new functionality being implemented in several other areas.
- New features to come in new releases!

Help wanted!
### Metrics currently being exported

<table>
<thead>
<tr>
<th>Errors</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>(error, fatal, panic, ...)</td>
<td>(frontend, backend)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sessions</th>
<th>Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>(encrypted, unencrypted)</td>
<td>(insert, update, delete, ...)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transactions</th>
<th>Notices</th>
</tr>
</thead>
<tbody>
<tr>
<td>(commit, rollback)</td>
<td>(notice, log, warning, ...)</td>
</tr>
</tbody>
</table>

Counters (metric / second)
Expected result

https://github.com/ongres/envoy-postgres-stats-example
COMING TO ENVOY 1.18*: POSTGRES SSL OFFLOADING AND MONITORING
Postgres SSL

- Doesn’t operate at the TCP level 4, but rather application level (7).
- Initial connection is unencrypted, then a request to “upgrade the connection to SSL” is performed. Similar to STARTTLS in SMTPS.
- Database connection costs are high. SSL database connection costs are very high.
- Use a connection pooler! Like PgBouncer!
  - PgBouncer -> single threaded -> swamped under heavy SSL connection load.
- Turning on/off SSL or rotating certificates requires database restart -> downtime.
Offload Postgres SSL to Envoy!

- Avoids both Postgres and PgBouncer SSL performance problems.
- Allows monitoring of encrypted traffic!
- Turn on/off, rotate certificates without database impact.
- Programmatic management: use Envoy xDS APIs to manage certificates.

StartTls infrastructure already released on 1.17. Postgres specific filter implementation coming on 1.18*. 

*Postgres specific filter implementation is currently planned for version 1.18, subject to development milestones and release schedules.
Architecture
SSL DEMO
USE CASES
THE FUTURE OF
ENVOY’S POSTGRES PLUGIN
Envoy’s filter use-case: StackGres.io

StackGres Architecture
Future plans

- Better SQL parsing
- Producing per-database statistics
- Routing based on Query type
- Traffic drain
- Opent Telemetry Integration

Community:
- Envoy slack: envoyproxy.slack.com
- PostgreSQL specific channel: #envoy-postgres
- PostgreSQL related issues: [https://github.com/envoyproxy/envoy/labels/area%2Fpostgres](https://github.com/envoyproxy/envoy/labels/area%2Fpostgres)
References

Original Github Issue:
https://github.com/envoyproxy/envoy/issues/9107

First post about Envoy Postgres Filter:

Envoy Documentation:
https://www.envoyproxy.io/docs/envoy/latest/intro/arch_overview/other_protocols/postgres
