# Building a Trusted Language Extension for PostgreSQL

Jonathan Katz

Principal Product Manager Technical
Amazon Web Services

# Amazon Relational Database Service (Amazon RDS)

MANAGED RELATIONAL DATABASE SERVICE WITH YOUR CHOICE OF DATABASE ENGINE

Amazon Aurora

MySQL

PostgreSQL

MariaDB

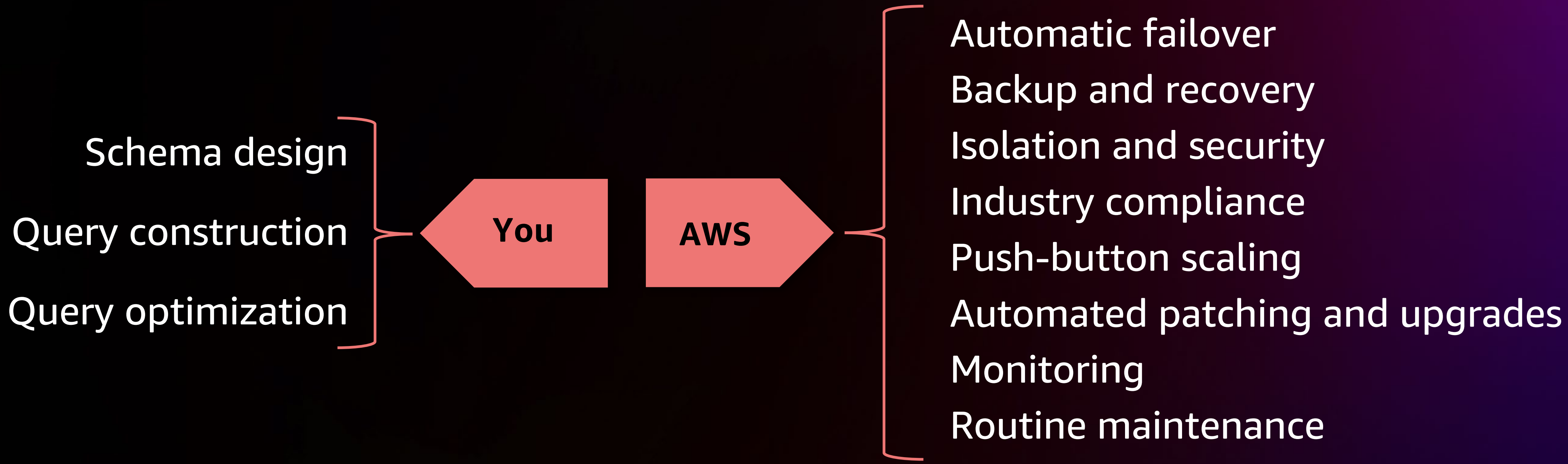
Microsoft SQL Server

Oracle

| Easy to administer | Available and durable | Performant and scalable | Designed to be secure and compliant |
|---|---|---|---|
| Easily deploy and maintain hardware, OS and DB software; built-in monitoring | Automatic Multi-AZ data replication; automated backup, snapshots, and failover | Scale compute and storage with a few clicks; minimal downtime for your application | Data encryption at rest and in transit; industry compliance and assurance programs |

# Amazon RDS – Fully managed

Spend time innovating and building new applications, not managing infrastructure



## Why PostgreSQL?

**Open source**

- Active development for more than 35 years
- Controlled by a community, not a single company

**Performance and scale**

- Robust data type implementations
- Extensive indexing support
- Parallel processing for complex queries
- Native partitioning for large tables

# What are PostgreSQL extensions?

Software packages containing new functionality for PostgreSQL

Large landscape of open-source and commercial extensions for PostgreSQL

Amazon Aurora and Amazon RDS support 85 PostgreSQL extensions

## PostgreSQL extensions

Syntax added in PostgreSQL 9.1 in 2011

Have been part of the PostgreSQL landscape much longer as contrib modules

Most databases are already using one or more extensions

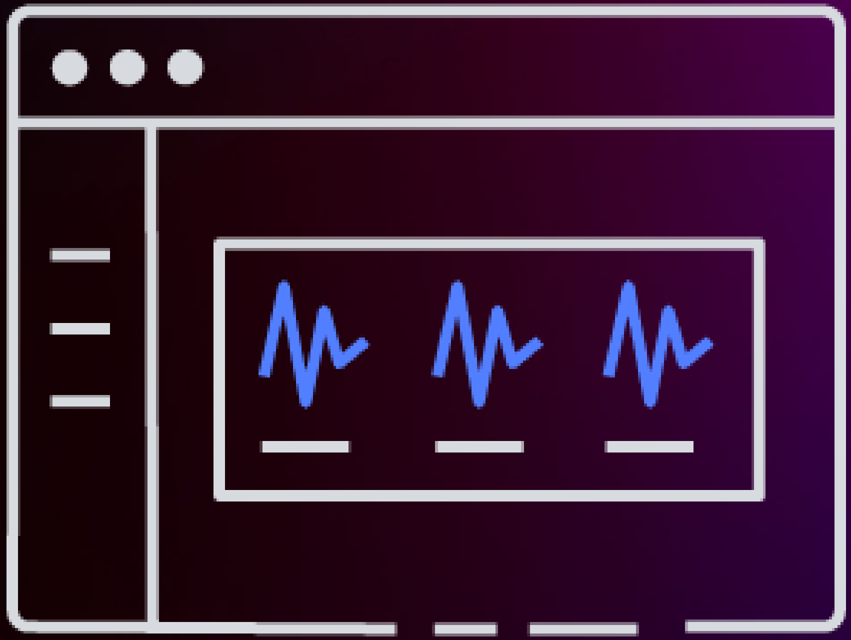
```
CREATE EXTENSION
  pg_stat_statements;

CREATE EXTENSION
  postgis;
```

# pg_stat_statements

Tracks execution statistics of all SQL statements executed by the database

Used extensively by monitoring tools

One of the first places to look when performance tuning



## pg_partman

Partition management helper on top of native PostgreSQL support

Automatically create/delete partitions

Used for optimizing data access/storage on large tables

# PostgreSQL foreign data wrapper (postgres_fdw)

Query/modify data from a remote PostgreSQL database

"Push down" work to other databases

Used for data federation, sharding, and extract, transform, and load (ETL)

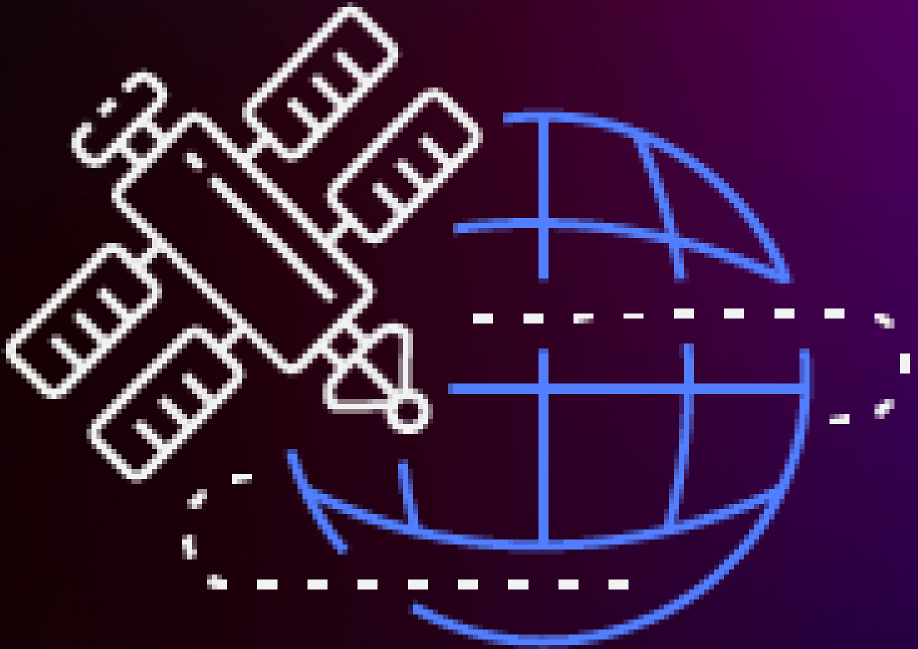
# PostGIS

Extends PostgreSQL to store and manage spatial information

Adds data types and thousands of functions

Used for mapping and GIS systems

Dependent extensions such as pgRouting

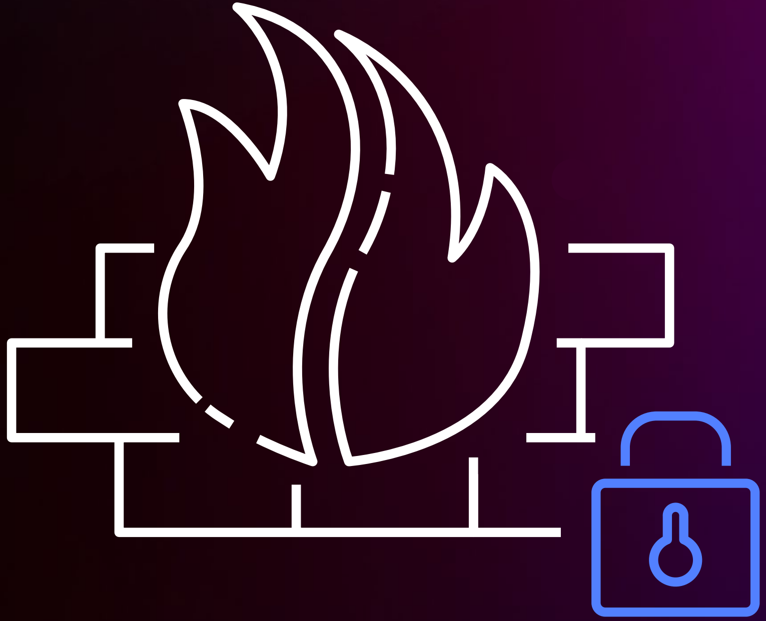


# Challenges with extensions

Installing extensions requires access to the file system

An extension may not be universally available

Upgrades across PostgreSQL major versions can be difficult



# Trusted Language Extensions (TLE)

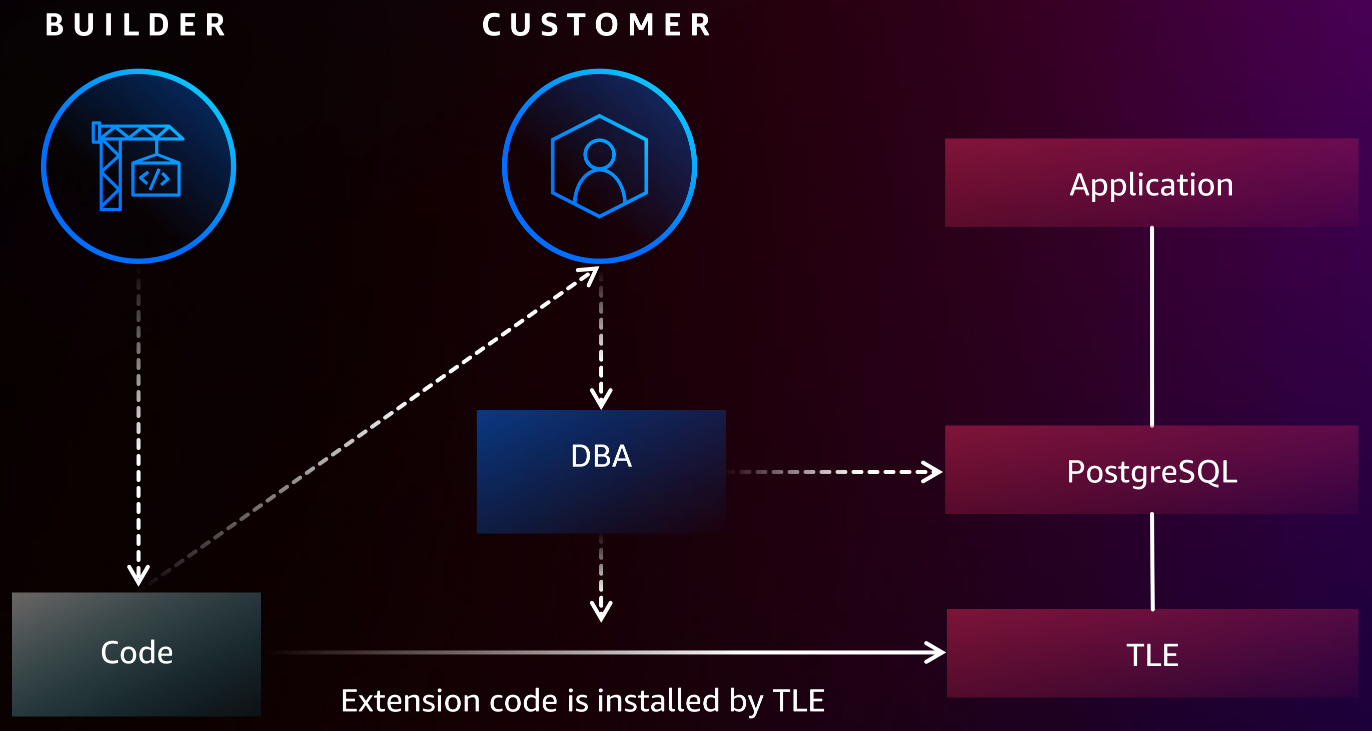
AMAZON AURORA POSTGRESQL-COMPATIBLE EDITION AND AMAZON RDS FOR POSTGRESQL

Available on Aurora PostgreSQL-Compatible and RDS for PostgreSQL 14.5 and higher

Builders create new libraries using TLE framework

Customers choose TLE extensions for their applications

DBAs control who can install and manage TLE extensions



# Safely extend PostgreSQL capabilities

TLE PROVIDES SAFE OPERATION OF POSTGRESQL EXTENSIONS IN PRODUCTION

## Improved safety

Enforces use of PostgreSQL trusted languages

TLE API provides safe access to PostgreSQL internals

## High performance

Supports languages that have C-like performance

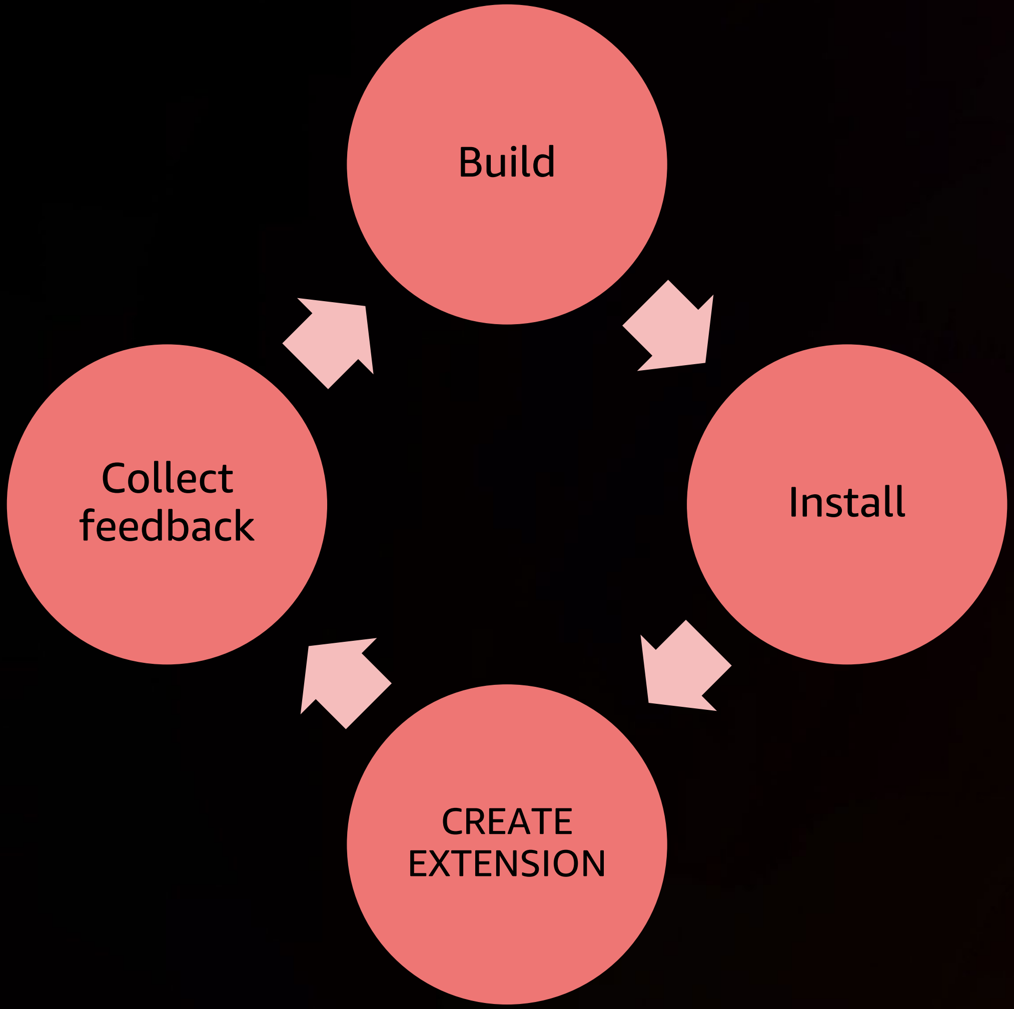
Removes need for C expertise to create a safe extension

## More flexibility

Build and use extensions on your timeline

Reduces DBA certification burden

# How it works



Any “trusted” PostgreSQL procedural language can be used in a TLE

- JavaScript
- Perl
- Tcl
- PL/pgSQL

# Enabling TLE

By design, requires configuration in a parameter group and inside the database

The extension pg_tle needs to be added to the parameter shared_preload_libraries

The extension needs to be created inside of the database

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name
      pgtle-params \
  --parameters
“ParameterName=shared_preload_libraries
,ParameterValue=pg_tle
,ApplyMethod=pending-reboot“
```

```
CREATE EXTENSION pg_tle;
```

# Managing a TLE

A builder can load a TLE through the install_extension function

Once installed, the TLE can be created like an ordinary extension

```
SELECT pgtle.install_extension(
  'tle_test',
  '0.1',
  'A very simple test extension',
  $pgtle$
    CREATE FUNCTION re()
    RETURNS int
    LANGUAGE SQL
    AS $$ SELECT 1 $$;

    CREATE FUNCTION Invent()
    RETURNS int
    LANGUAGE SQL
    AS $$ SELECT 2 $$;
  $pgtle$
);

CREATE EXTENSION tle_test;
```

# Managing a TLE

A builder can stage new versions of a TLE to be rolled out during the appropriate window

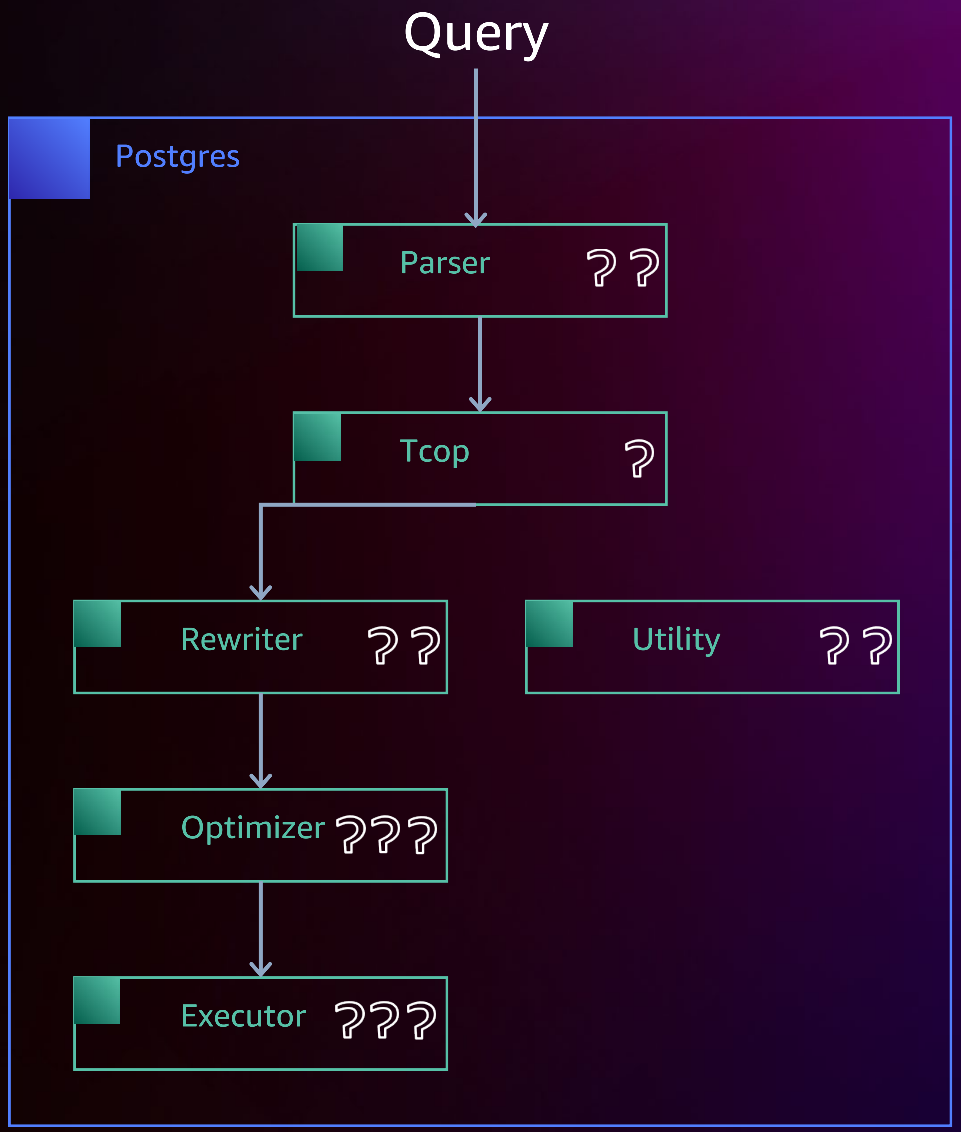
The code is the change needed to move from the old version to the new version

```
SELECT pgtle.install_update_path(
  'tle_test',
  '0.1',
  '0.2',
  $pgtle$
    CREATE OR REPLACE FUNCTION re()
    RETURNS int
    LANGUAGE SQL
    AS $$ SELECT 10 $$;

    CREATE OR REPLACE FUNCTION Invent()
    RETURNS int
    LANGUAGE SQL
    AS $$ SELECT 20 $$;
  $pgtle$
);

ALTER EXTENSION tle_test
  UPDATE TO '0.2';
```

## What is a hook?

Hooks are points along the execution path that allow for additional code to be plugged in to change or enhance the standard behavior

There are hooks in each PostgreSQL system with a total of 30 hooks in PostgreSQL 15



## Enabling a TLE hook

By design, requires configuration in a parameter group and inside the database

The specific hook needs to be enabled in the parameter group for it to be active

Once active, the hook needs to be registered as a TLE feature

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name pgtle-params \
  --parameters
"ParameterName=pgtle.enable_password_check
,ParameterValue=on
,ApplyMethod=immediate"
```

```
SELECT pgtle.register_feature(
   'test_password_hook',
   'passcheck');
```

# What are TLE features?

Create an association between a builder’s function and a TLE hook

Multiple functions can be associated with each feature

Features can be unregistered as needed

# Creating a hook function

Can be an ordinary function of any trusted language

Must have the proper function signature with all required parameters

Any actions in the function will affect the top command

```
CREATE FUNCTION test_password_hook(
  username text,
  password text,
  password_type pgtle.password_types,
  valid_until timestamptz,
  valid_null boolean)
RETURNS void AS
$$
BEGIN
  PERFORM pg_sleep(5);
END
$$
LANGUAGE plpgsql;
```

## Security around hooks

Hooks execute with elevated privileges

Only members of the pgtle_admin role can register functions with a feature

It is the responsibility of the builder to write functions that adhere to security policies

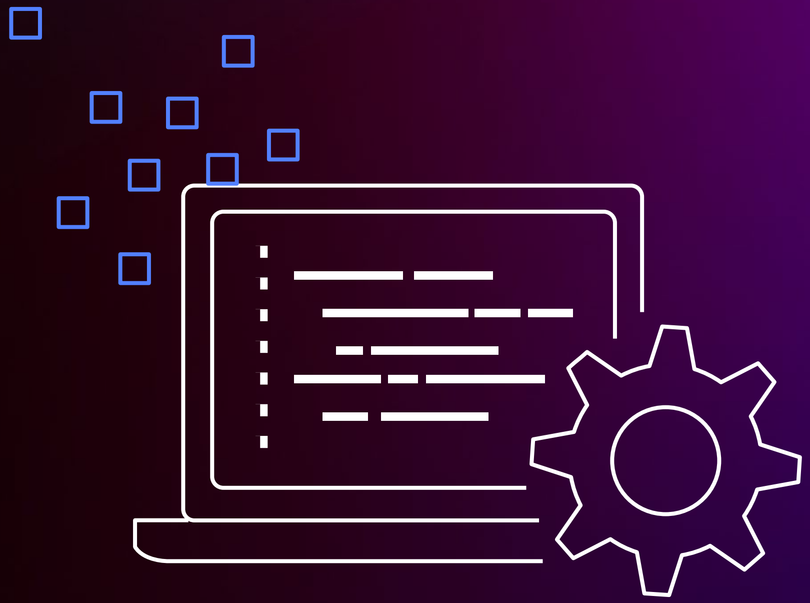
## Choosing a trusted language

Familiarity should be the leading factor

Each language has its own technical advantage

Amazon Aurora and Amazon RDS currently support PL/pgSQL, PLV8 (JavaScript), PL/Perl, and PL/Tcl

When in doubt, use PL/pgSQL



# PL/pgSQL

Installed by default on all PostgreSQL databases, except where explicitly removed

Simple syntax for database access

Frequently used for Oracle to PostgreSQL migrations

```
CREATE FUNCTION reinvent (a int)
  RETURNS varchar AS
$$
DECLARE
  r varchar;
BEGIN
  IF (a > 0) THEN
    r := 're';
  ELSE
    r := ‘Invent';
  END IF;

  RETURN r;
END
$$
STRICT
LANGUAGE plpgsql;
```

# PLV8 (JavaScript)

Strong JSON support

Can be used to create user-defined window functions

Can call other PLV8 functions without context switching back to the engine

```
CREATE FUNCTION reinvent (a int)
  RETURNS varchar AS
$$
  if (a > 0) {
    r = "re";
  } else  {
    r = "Invent";
  }
  return r;
$$
STRICT
LANGUAGE plv8;
```

## Demo

# Loading community extensions

Community extensions that use trusted languages can be loaded into Amazon Aurora and Amazon RDS

A client tool simplifies loading the extension

From GitHub to RDS in less than a minute

```
git clone \
  https://github.com/adjust/pg-telemetry.git

cd pg-telemetry

./pg_tle_manage_local_extension.sh \
  --action install \
  --pgconn
"postgresql://postgres@reinvent...rds.amazonaws
.com/postgres" \
  --extname pgtelemetry \
  --extpath ~/extensions/pg-telemetry
```

## Documentation

Amazon Aurora:
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/PostgreSQL_trusted_language_extension.html

Amazon RDS:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/PostgreSQL_trusted_language_extension.html

AWS News: https://aws.amazon.com/blogs/aws/new-trusted-language-extensions-for-postgresql-on-amazon-aurora-and-amazon-rds/

## Open-source project

Trusted Language Extensions for PostgreSQL is Apache 2.0 licensed

Source code: https://github.com/aws/pg_tle

We invite everyone to leave feedback and contribute to TLE

# Thank you!

Jonathan Katz
jkatz@amazon.com