



Trusted Language Extensions for Amazon RDS for PostgreSQL

Peter Celentano

Senior Specialist Solutions Architect
Amazon Web Services

Sukhpreet Kaur Bedi

Specialist Solutions Architect
Amazon Web Services

What are PostgreSQL extensions?

Software packages containing new functionality for PostgreSQL

Large landscape of open source and commercial extensions for PostgreSQL



PostgreSQL extensions

Syntax added in PostgreSQL 9.1 in 2011

```
CREATE EXTENSION  
    pg_stat_statements;
```

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

```
CREATE EXTENSION  
    postgis;
```

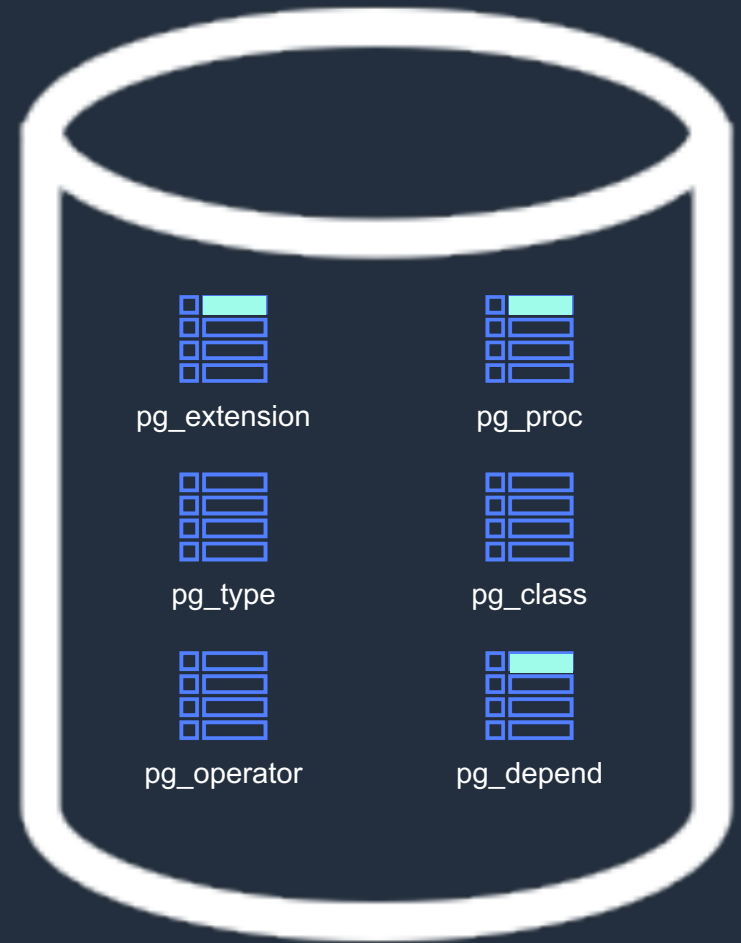
Most databases are already using one or more extensions



Creating an extension

CREATE EXTENSION earthdistance
CASCADE;

```
# earthdistance extension  
comment = 'calculate great circle distance'  
CREATE FUNCTION earth(  
  default_version = '1.0',  
  LANGUAGE SQL IMMUTABLE,  
  module_pathname = '$libdir',  
  AS 'SELECT '6378168'::float  
  relocatable = true  
  requires = 'cube'
```



Challenges with extensions

Installing extensions requires access to the filesystem

An extension may not be universally available

Upgrades across PostgreSQL major versions can be difficult



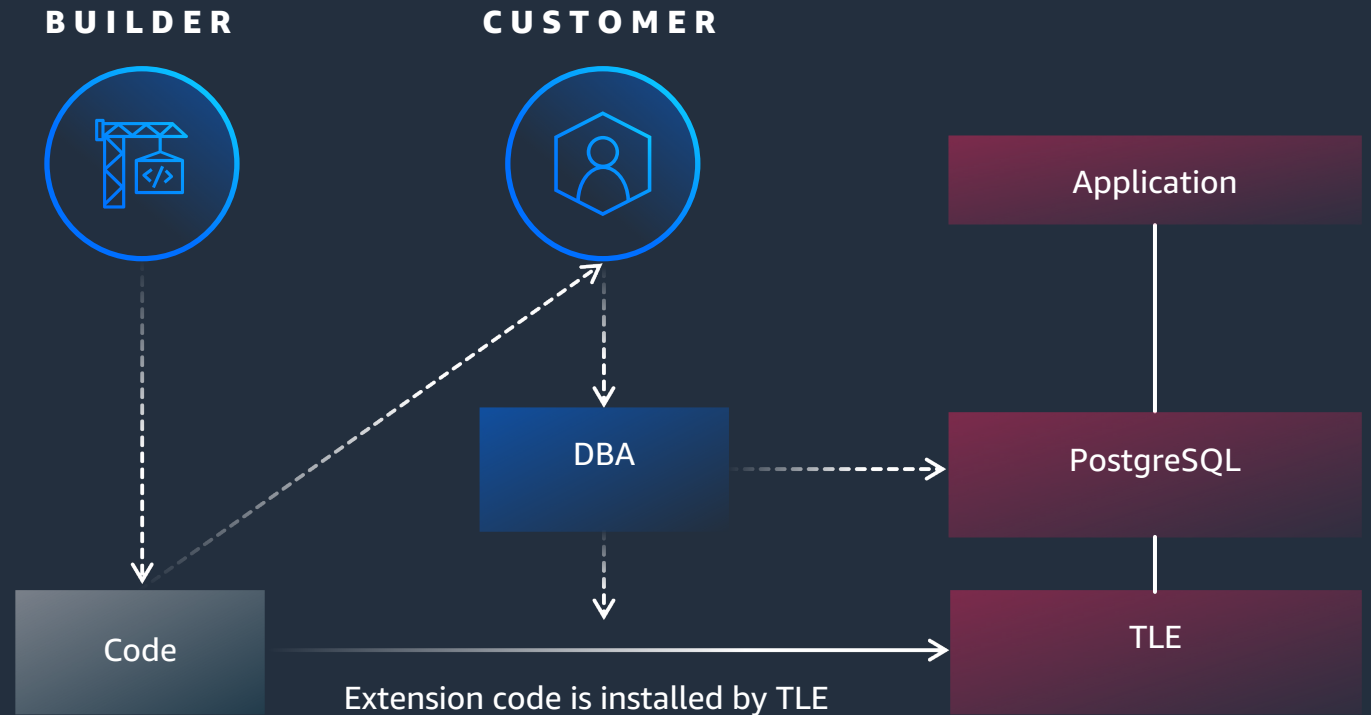
Trusted Language Extensions (TLE)

Builders create new libraries using TLE framework

Customers choose TLE extensions for their apps

DBAs control who can install and manage TLE extensions

Apache 2.0 Licenced



Safely extend PostgreSQL capabilities

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

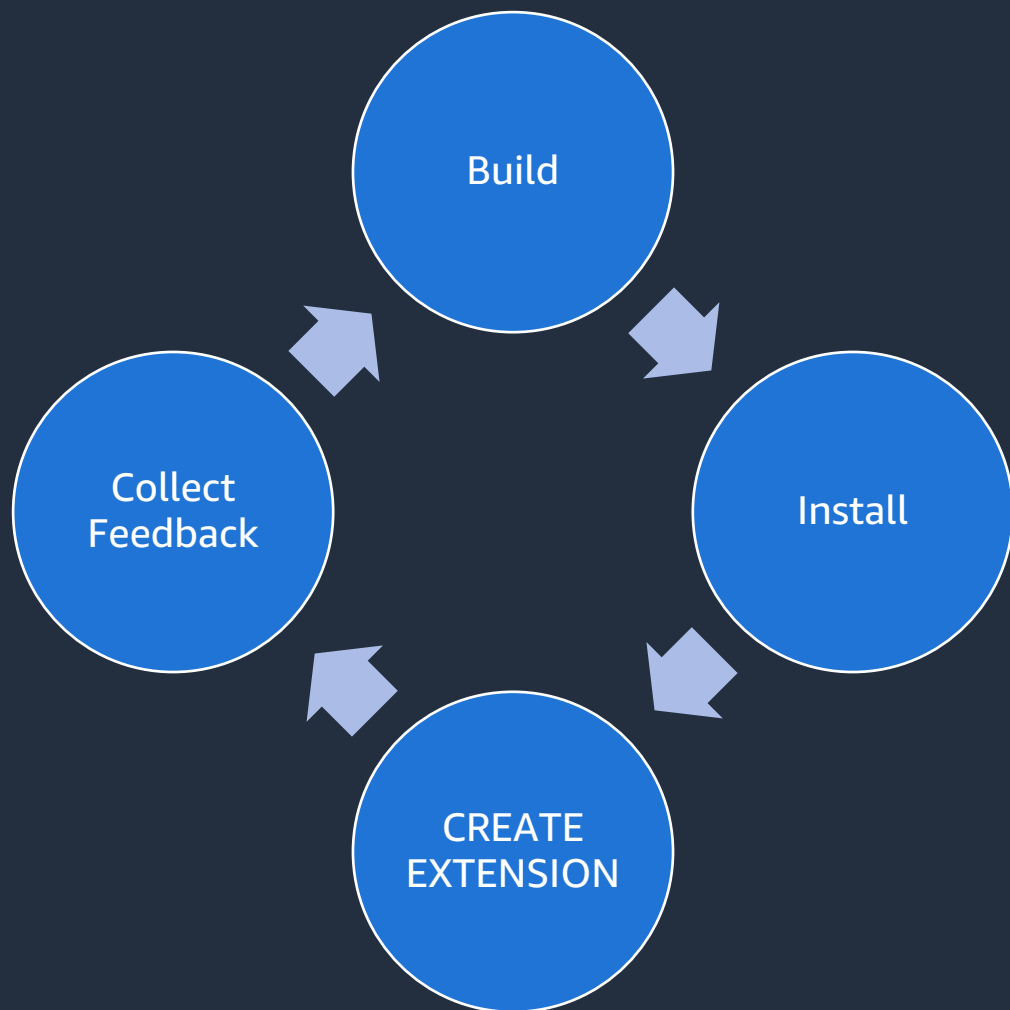
Removes need for C expertise to create a safe extension

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

Rust

Enabling TLE

When deployed in community PostgreSQL - pull down from GIT, make, make install, configure, and enable

The extension `pg_tle` needs to be added to `shared_preload_libraries` in `postgresql.conf`

The extension needs to be created inside of the database

```
shared_preload_libraries = "pg_tle"
```

```
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 Pgconf()  
    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 changes 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 Pgconf()  
        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 16



Enabling a TLE hook

By design, requires configuration within the database config file and inside the database it's used within

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

```
SELECT * FROM pgtle.feature_info;
```

The specific hook needs to be enabled in the postgresql.conf for it to be active

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



What are TLE features

Creates 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



Custom Data Types in PostgreSQL

Core PostgreSQL contains 55+ standard data types

While flexible, they don't always suit the needs of all workloads

Custom data types can streamline your application logic and make codebase more comprehensible



Custom Data Types in TLE

TLE supports custom PostgreSQL data types as extensions for better version control and deployment

Supports writing custom data types with all supported Trusted Languages used by TLE

Available from TLE v1.1.1 on Github



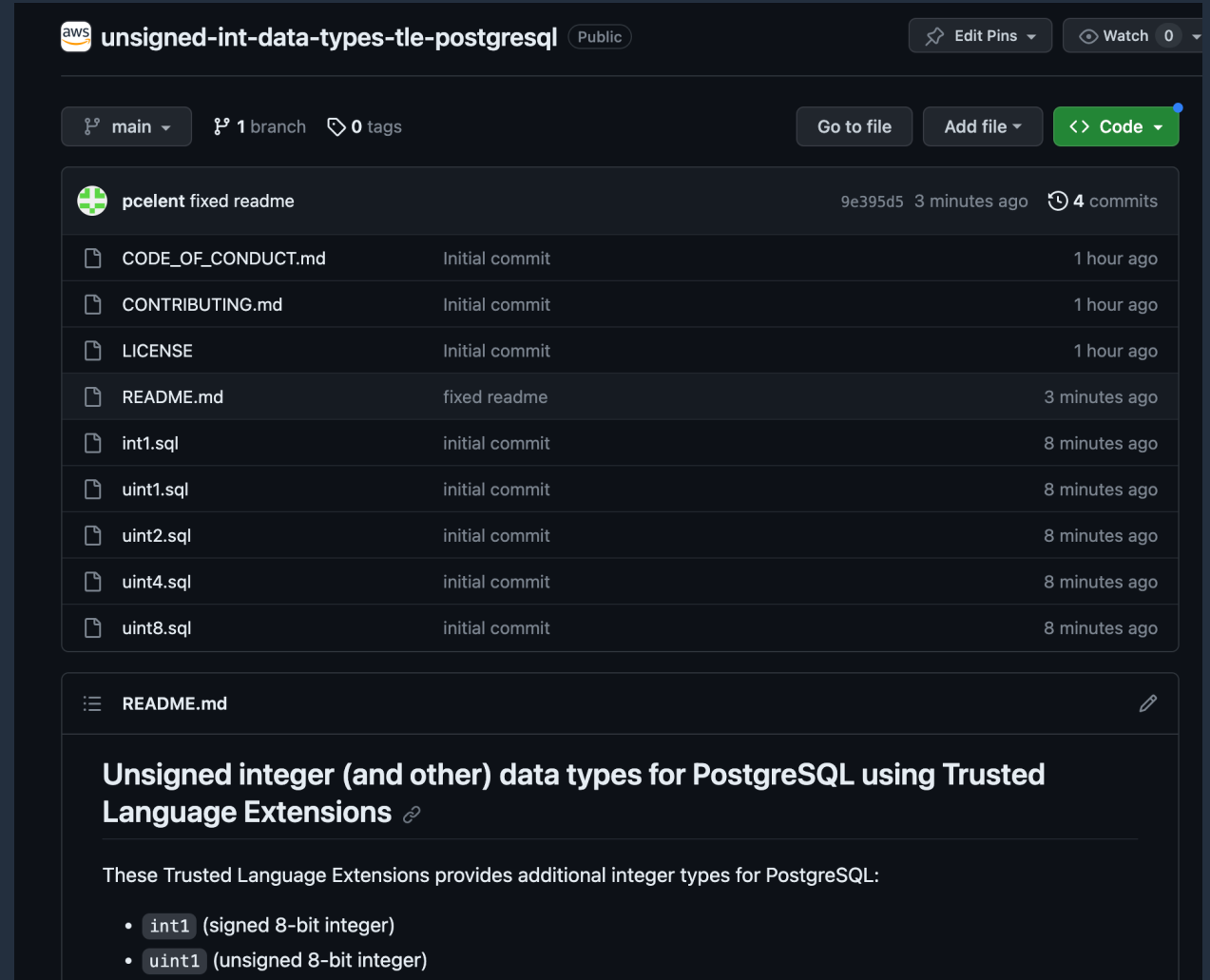
Custom Data Types in TLE – Example: unsigned int

Unsigned integer data types can be useful in vector workloads

-PostgreSQL core doesn't yet support them

A git repo containing a number of uint data types for PostgreSQL (written as TLEs) is available here:

<https://github.com/aws-samples/unsigned-int-data-types-tle-postgresql>



The screenshot shows a GitHub repository page for 'unsigned-int-data-types-tle-postgresql'. The repository is public and has 1 branch (main) and 0 tags. The commit history shows a 'fixed readme' commit by pcelent 3 minutes ago, with 4 total commits. The file list includes CODE_OF_CONDUCT.md, CONTRIBUTING.md, LICENSE, README.md, int1.sql, uint1.sql, uint2.sql, uint4.sql, and uint8.sql. The README.md file is open, showing the title 'Unsigned integer (and other) data types for PostgreSQL using Trusted Language Extensions' and a list of data types: int1 (signed 8-bit integer) and uint1 (unsigned 8-bit integer).

File	Commit	Time
CODE_OF_CONDUCT.md	Initial commit	1 hour ago
CONTRIBUTING.md	Initial commit	1 hour ago
LICENSE	Initial commit	1 hour ago
README.md	fixed readme	3 minutes ago
int1.sql	initial commit	8 minutes ago
uint1.sql	initial commit	8 minutes ago
uint2.sql	initial commit	8 minutes ago
uint4.sql	initial commit	8 minutes ago
uint8.sql	initial commit	8 minutes ago

Unsigned integer (and other) data types for PostgreSQL using Trusted Language Extensions

These Trusted Language Extensions provides additional integer types for PostgreSQL:

- `int1` (signed 8-bit integer)
- `uint1` (unsigned 8-bit integer)



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



Demo



Demo

Resource Groups & Tag Editor

Amazon RDS ×

- Dashboard
- Databases**
- Query Editor
- Performance insights
- Snapshots
- Exports in Amazon S3
- Automated backups
- Reserved instances
- Proxies

Subnet groups

Parameter groups

Option groups

Custom engine versions

Zero-ETL integrations [New](#)

Events

Event subscriptions

CloudShell [Feedback](#)

RDS > Databases > pgtle-demo

pgtle-demo

[Modify](#) [Actions](#) ▼

Summary

DB identifier pgtle-demo	CPU <div><div style="width: 2.93%;"></div></div> 2.93%	Status ✔ Available	Class db.r5.large
Role Instance	Current activity <div><div style="width: 0.00 sessions;"></div></div> 0.00 sessions	Engine PostgreSQL	Region & AZ us-east-1c

[Connectivity & security](#) | [Monitoring](#) | [Logs & events](#) | [Configuration](#) | [Maintenance & backups](#) | [Tags](#)

Connectivity & security

Endpoint & port	Networking	Security
Endpoint pgtle-demo.ciculd4q7utg.us-east-1.rds.amazonaws.com	Availability Zone us-east-1c	VPC security groups rds-ec2-1 (sg-0ad0139b776fb343e) ✔ Active MyFirstSG (sg-0be1633180e162f74) ✔ Active
Port	VPC vpc-b2cd16cf	

© 2023, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)



Future direction

Additional trusted languages (Go, Java)

Support more hooks

Foreign data wrappers

Background workers

AWS SDK



Q/A





Thank you!

Peter Celentano

pcelent@amazon.com

Sukhpreet Kaur Bedi

bedsukhp@amazon.com

