



# Guardians of the Database

*Best Practices for a Secure PostgreSQL Environment*

Guy Gyles  
Senior DB Architect - Technical Lead 🦒

**pgConf.BE @ UCLL**

# Agenda

## Guardians of the Database

🔗 Introduction

🔗 System

🔗 Database

🔗 Monitoring

🔗 Healthcheck



pgConf.BE @ UCLL

# Introduction

## Guardians of the Database

### The story

Have you ever noticed that **no two zebras have the same stripe pattern**? Just like those unique markings, every business has its own distinct database needs. That's why, at Zebanza, we don't believe in one-size-fits-all solutions. We take the time to understand your specific challenges and goals, tweaking our services to **fit your unique requirements**.

And just like a zebra's black and white stripes, we believe different technologies can coexist and work together harmoniously. Whether your environment is primarily open-source or a **blend of open-source and traditional solutions**, we'll help you find the right balance for your needs.



### The community

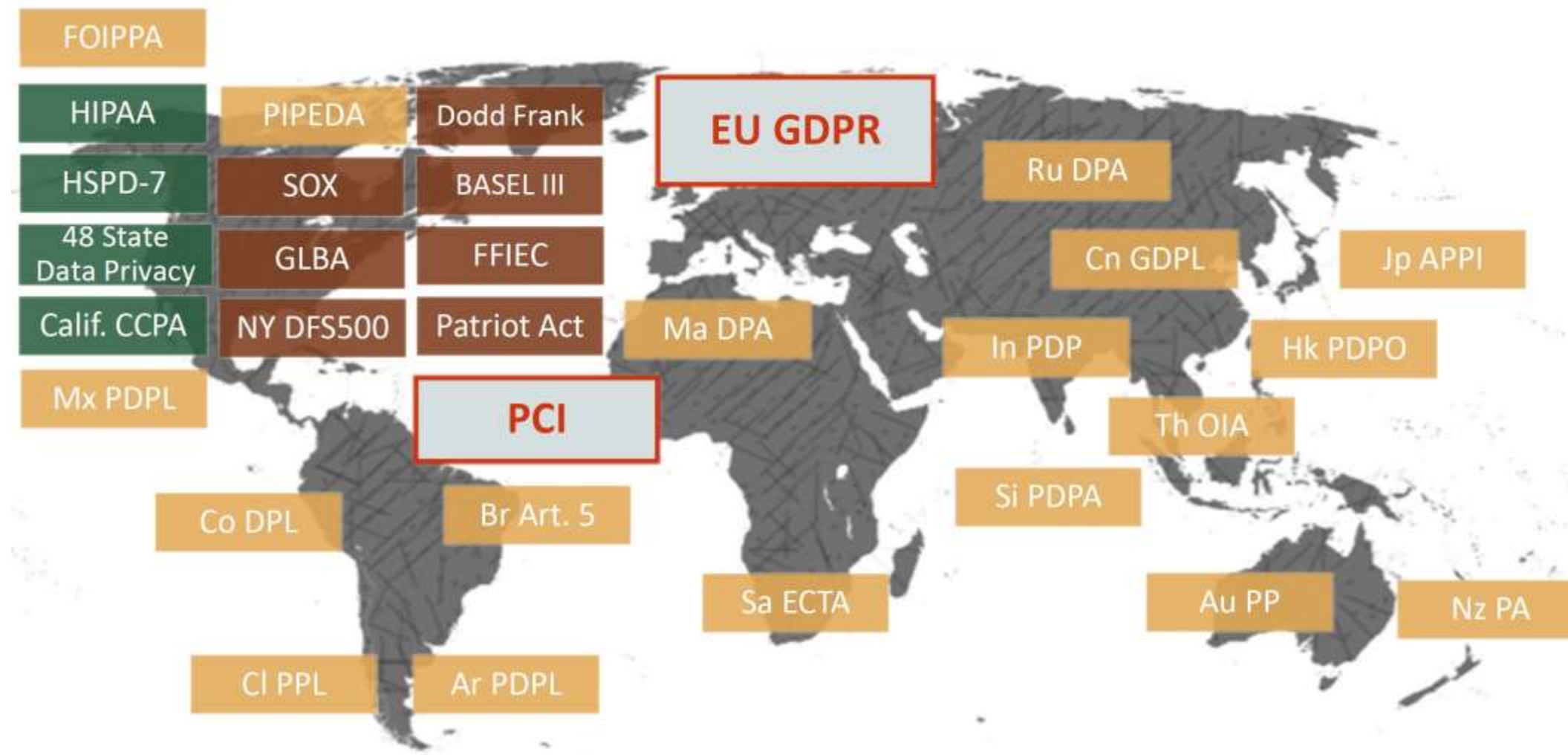
Our passion for open source extends beyond our work with clients. We're committed to building a stronger, more vibrant open-source community. We do this by **organizing meetups**, sharing our knowledge through **blog posts and presentations**, and actively contributing to open-source **projects**.

We're also passionate about connecting the talented open-source experts within the **Cronos Group** with the broader open-source world. This lets us tap into a **large network of expertise and resources**, benefiting both our clients and the open-source community.



# Data Security & Privacy

## Standards & Regulations





# Data Security & Privacy

## Standards & Regulations

---



### **PCI DSS**

-  *Payment Card Industry Data Security Standard*
-  Information security standard used to handle credit cards from major card brands

### **GDPR**

-  *General Data Protection Regulation*
-  European data privacy and security law

### **FIPS**

-  *Federal Information Processing Standards*
-  Standard requirements for ensuring computer security and interoperability

### **HIPAA**

-  *Health Insurance Portability and Accountability Act*
-  Information maintained by the healthcare and healthcare insurance industries

### **NIS**

-  *Network and Information Security Directive*

### ... ..

# Data Security & Privacy

## Standards & Regulations - NIS2

1 of 2



### 🔒 Perform a Risk Assessment

- 🔒 Identify potential vulnerabilities in your database.
- 🔒 Evaluating access controls, data encryption, and backup procedures.

### 🔒 Enhance Access Controls

- 🔒 User Authentication
- 🔒 Role-Based Access Control

### 🔒 Data Encryption

- 🔒 At Rest → Encrypt sensitive data stored in the database
- 🔒 In Transit → Use SSL/TLS to encrypt data transmitted between the database and clients.

### 🔒 Regular Updates and Patching

- 🔒 Keep the database and associated software up to date with the latest security patches.

### 🔒 Monitoring and Logging

- 🔒 Audit Logs

# Data Security & Privacy

## Standards & Regulations - NIS2

2 of 2



### Backup and Recovery

- Regular database backups and securely stored.
- Test the recovery procedures.

### Incident Response Plan

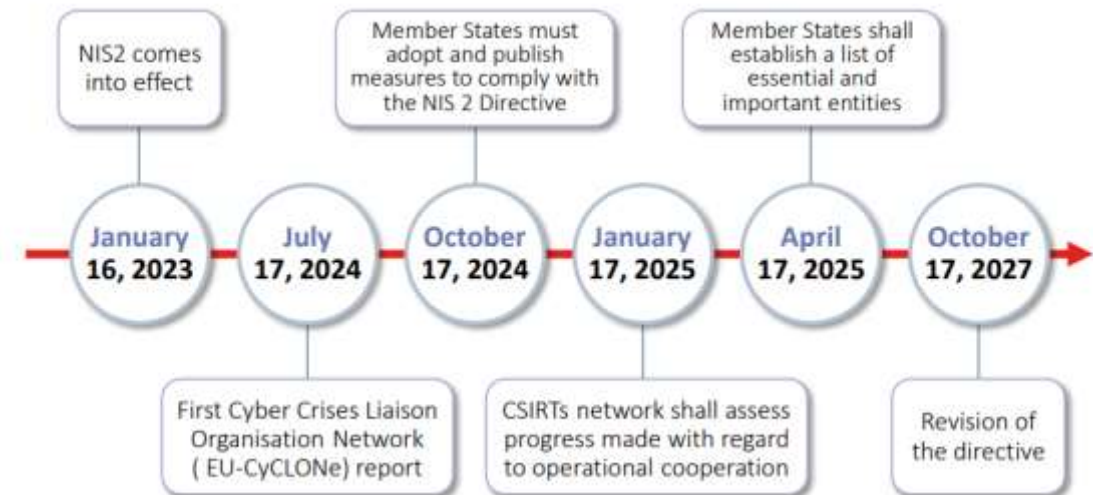
- ... outlines steps to take in the event of a security breach.

### Training and Awareness

- ... on cybersecurity best practices and NIS2 compliance requirements.

### Periodic Reviews and Audits

- ... of database security measures to ensure ongoing compliance with NIS2.





# Introduction

## General Security Recommendations

- 🔒 Avoid physical access to the (database) server
- 🔒 Limit access to the data network in general
- 🔒 Keep operating system and database software up-to-date



Windows Server Update Service (WSUS)  
System Center Configuration Manager (SCCM)

```
Install-Module -Name PSWindowsUpdate  
Get-WindowsUpdate  
Get-WindowsUpdate -Classification SecurityUpdates
```

# Introduction

## Linux Security Patching



```
apt update --only-upgrade
apt list --upgradable
apt upgrade --only-upgrade
```

```
apt install unattended-upgrades
dpkg-reconfigure --priority=low unattended-upgrades

apt-mark hold postgresql-* postgres* ... ..
apt-mark unhold postgresql-* postgres* ... ..
```



```
yum | dnf update-minimal --security --exclude=kernel*
yum | dnf update-minimal --security
yum | dnf update-minimal
yum | dnf update --security
```

```
dnf install dnf-automatic
systemctl enable --now dnf-automatic.timer

yum updateinfo list security | grep "ELSA"
yum updateinfo list security | grep "ELBA"
```



```
zypper list-patches
zypper list-patches --category security
zypper patch --category security
```

# Database Security

Defense line for your data



## Network

- Firewall
- Intrusion

## Vulnerability

- Software

## Connectivity

- TLS / SSH
- UDS / TCP

## Access

- Users / RBAC
- RLS

## Authentication

- Password checks
- HBA

## Data

- Encryption
- Views





# Database Security

## Security Overview

### 🔒 Network

- 🔒 Firewalls and Connectivity
- 🔒 Intrusion prevention / detection

```
iptables -A INPUT -p tcp --dport 5432 -s  
<allowed-ip-address>/32 -j ACCEPT
```

### 🔒 Access

- 🔒 Authentication and Authorization
- 🔒 Users / Roles → Role Based Access Control
- 🔒 Row Level Security

### 🔒 Data

- 🔒 Encryption → in Transit and at Rest
- 🔒 API's → Views / Stored Procedures / Prepared Statements

### 🔒 Auditing

- 🔒 Monitoring



# Database Security

## Security Levels

---

### 🔒 Server and Application

- 🔒 Known clients / application servers

### 🔒 Database

- 🔒 Users / Roles
- 🔒 Password
- 🔒 Permissions



Consider ***CredCheck*** extension in PostgreSQL

### 🔒 Objects

- 🔒 Table privileges
- 🔒 Grant / Revoke



# Database Security

Access to the servers and databases

---

## Restrict incoming connections to PostgreSQL

listen\_addresses → specifies IPs listening for incoming connections

## Restrict the ability to connect to a database

Host-Based Access Control File → pg\_hba.conf

## SSL can be forced for certain clients

## Authentication methods

trust, reject, **gss**, **sspi**, **krb5**, ident, peer, pam, lpad, radius, bsd, **cert**, **scram**, md5

Password Encryption

## Allow *superuser* access only from certain IP's



# Database Security

## Host Based Access

Before activating new ***hba-rule*** check contents of the ***pg\_hba.conf*** file

PostgreSQL v15 and before

```
SELECT error, r.* FROM pg_hba_file_rules r
WHERE error IS NOT NULL
ORDER BY Line_Number;
```

PostgreSQL v16 and newer

```
SELECT error, r.* FROM pg_hba_file_rules r
WHERE error IS NOT NULL
ORDER BY Rule_Number;
```

SELECT error, r.* FROM pg_hba_file_rules r ORDER BY Rule_Number							
	error	rule_number	file_name	line_number	type	database	user_name
1	[NULL]	1	/etc/postgresql/15/main/pg_hba.conf	118	local	all	postgres
2	[NULL]	2	/etc/postgresql/16/main/pg_hba.conf	123	local	all	all
3	[NULL]	3	/etc/postgresql/16/main/pg_hba.conf	125	host	all	all
4	[NULL]	4	/etc/postgresql/16/main/pg_hba.conf	126	host	all	g-gyles
5	[NULL]	5	/etc/postgresql/16/main/pg_hba.conf	130	host	all	all
6	[NULL]	6	/etc/postgresql/16/main/pg_hba.conf	133	local	replication	all
7	[NULL]	7	/etc/postgresql/16/main/pg_hba.conf	134	host	replication	all
8	[NULL]	8	/etc/postgresql/16/main/pg_hba.conf	135	host	replication	all
9	invalid connection type	[NULL]	/etc/postgresql/16/main/pg_hba.conf	127	[NULL]	[NULL]	[NULL]



# Database Security

## Host Based Access

### 🐘 peer

- 🐘 Authenticated by underlying OS
- 🐘 Use it **Local-Only** and on **Trusted** OS Environment

### 🐘 ident

- 🐘 Need an ident server to allow a network connection → **pg\_ident.conf**
- 🐘 IDENT is not encrypted and can be spoofed
- 🐘 Use it in trusted internal networks

```
# TYPE          DATABASE      USER          ADDRESS        METHOD
# "local" is for socket connections only
local           all           all
# IPv4 local connections:
host            all           all            127.0.0.1/32   ident
```



# Database Security

## Host Based Access

### 🐘 md5

- 🐘 Password is stored in encrypted format

### 🐘 scram-sha-256

- 🐘 Password is cryptographically hashed
- 🐘 Most secure method at the moment
- 🐘 Default and recommended since PostgreSQL v10

#	TYPE	DATABASE	USER	ADDRESS	METHOD
	host	all	all	10.3.21.0/32	<b>md5</b>
	host	all	all	10.3.22.0/32	<b>scram-sha-256</b>



# Database Security

## Host Based Access

### krb5

-  Kerberos service on client and PostgreSQL Server

### ldap

-  Authenticate to e.g. AD

```
# TYPE DATABASE USER ADDRESS METHOD
Host all all 10.3.21.0/32 gss include_realm=1 krb_realm=AD.zebanza.BE
Host all all 10.3.22.0/32 ldap ldapserver=10.3.12.222 ldapbaseddn="... .."
```

```
Postgres=# create role guy.gyles@AD.zebanza.BE superuser;
```



# Database Security

## Host Based Access

### 🐘 host

- 🐘 Allows TCP/IP connections, regardless of whether SSL is used.
- 🐘 SSL is optional and depends on server/client settings.

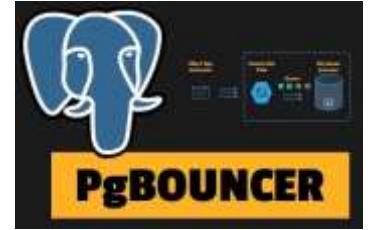
### 🐘 hostssl

- 🐘 Allows only SSL-encrypted TCP/IP connections.
- 🐘 If the client does not use SSL, the connection is rejected

#	TYPE	DATABASE	USER	ADDRESS	METHOD
	host	all	all	10.13.2.0/32	<b>scram-sha-256</b>
	<b>hostssl</b>	db1	all	192.168.24.196/32	<b>md5</b>
	hostssl	<b>db5</b>	<b>jan</b>	10.13.22.0/24	scram-sha-256
	hostssl	db5	all	10.13.22.123/32	scram-sha-256

### 🐘 Using regular expression patterns in PostgreSQL v16 and newer

#	TYPE	DATABASE	USER	ADDRESS	METHOD
	local	<b>db1</b> , <b>"/^db\d{2,4}\$",db7</b>	all	localhost	trust



# Connection Pooling

pgBouncer

1 of 4

## 🐘 Authentication and Encryption

- 🐘 Only authorized clients can access the database.
- 🐘 Extra layer of security to the database.

## 🐘 Connection Control

- 🐘 Limits direct access to the PostgreSQL server.
- 🐘 Prevent direct connections to the database.

## 🐘 TLS / SSL Support

- 🐘 Support for TLS/SSL client certificate authentication
- 🐘 Securing the data transmission between the client and the server.

## 🐘 Passthrough Authentication

- 🐘 Authenticate securely users without having access to passwords.
- 🐘 More secure and easier to maintain, no passwords updates needed.

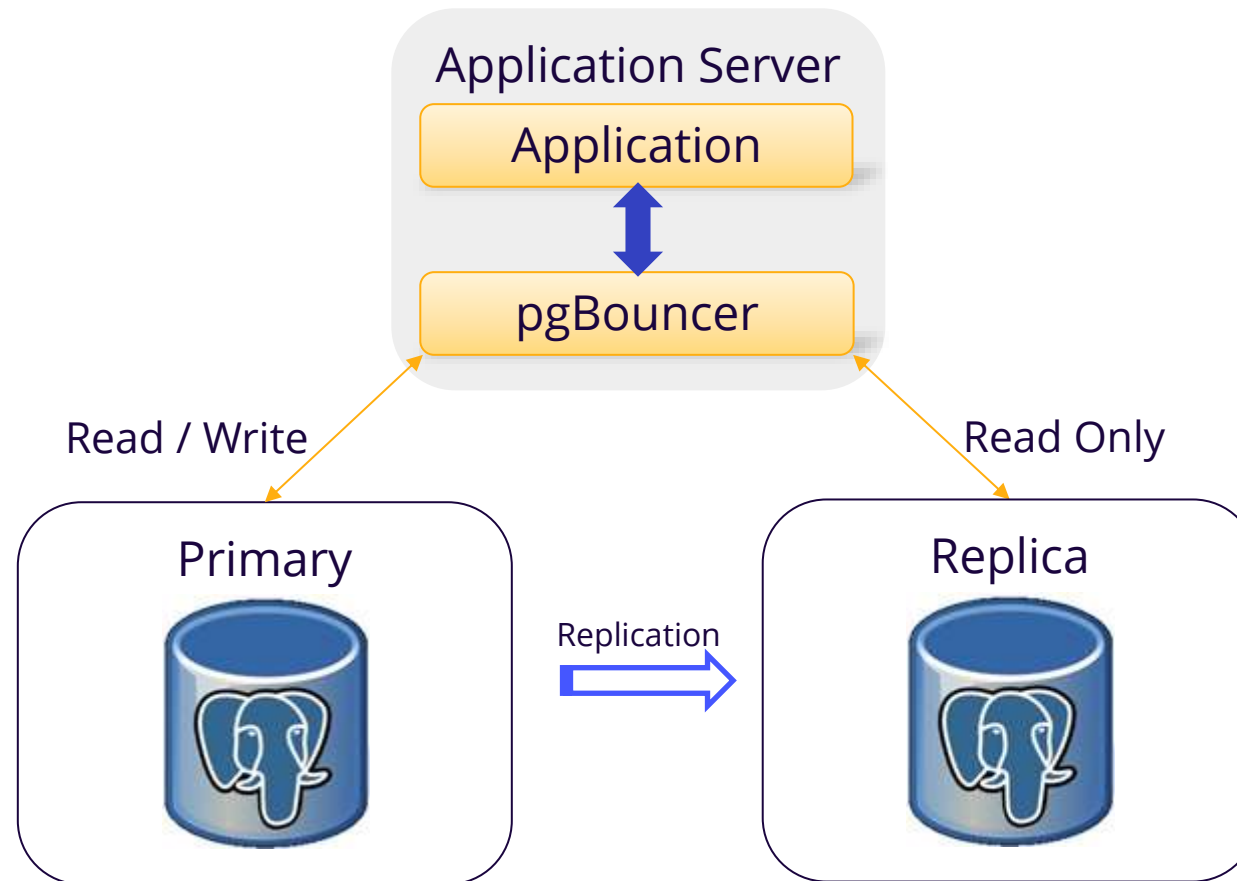
## 🐘 Local Connection Lockdown

- 🐘 Prevents unauthorized remote access.

# Connection Pooling

pgBouncer – Basic setup

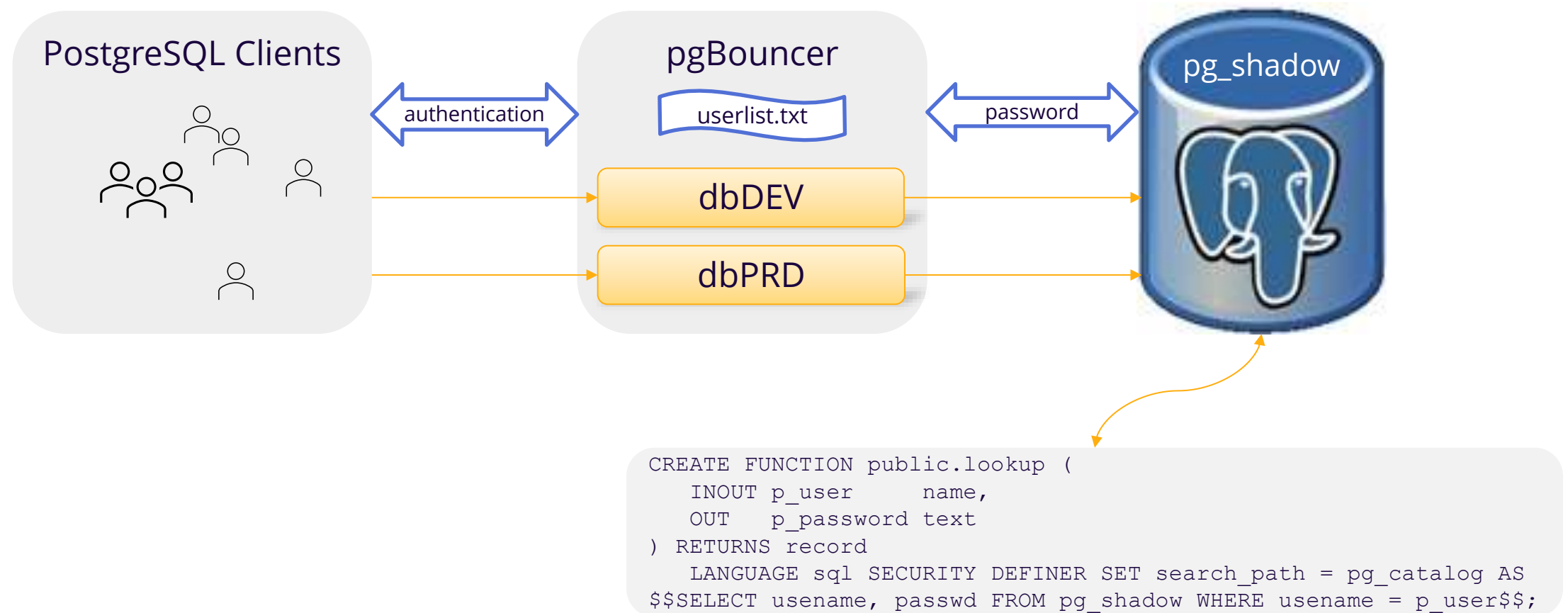
2 of 4



# Connection Pooling

pgBouncer – Basic setup

3 of 4

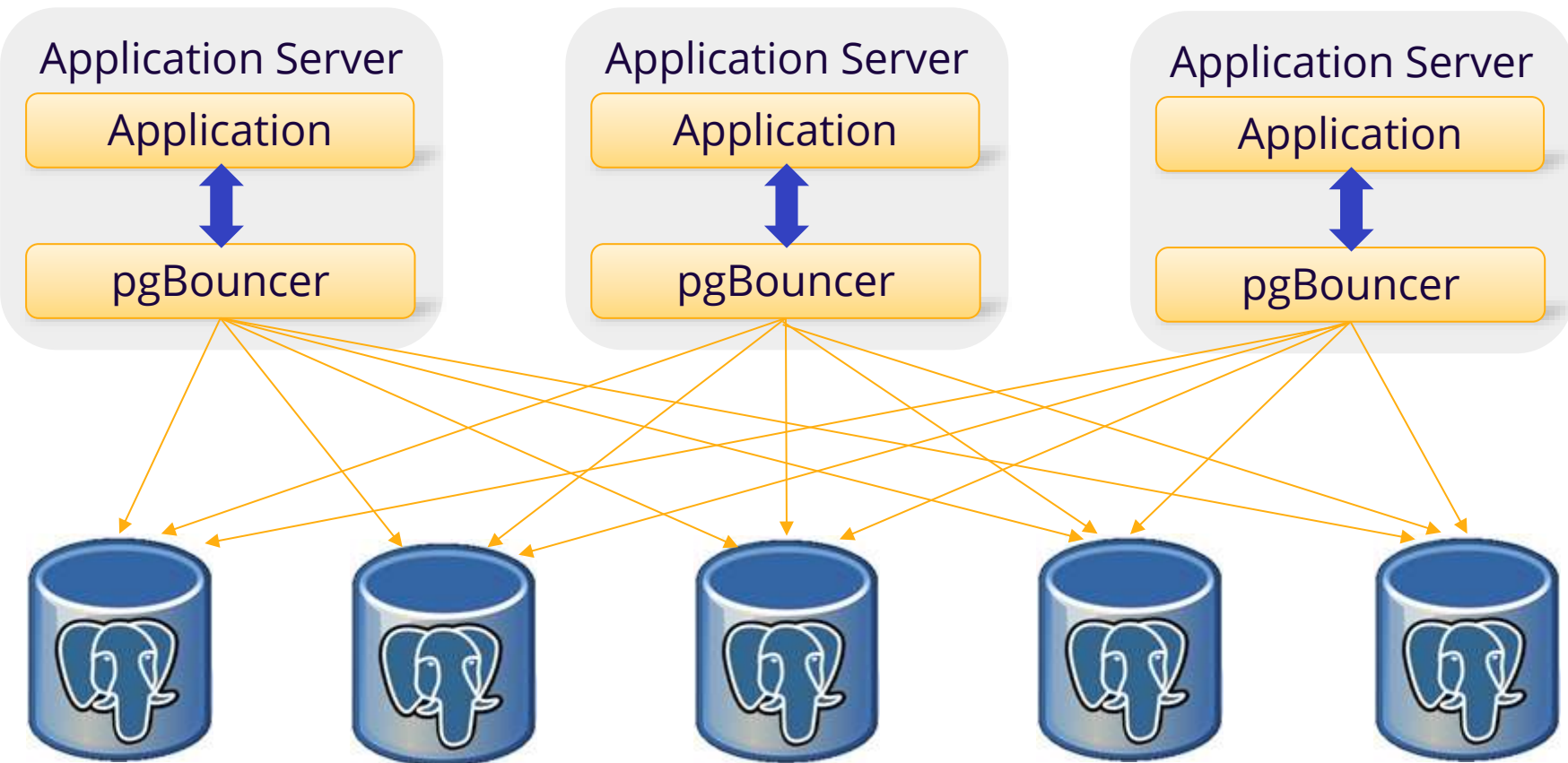




# Connection Pooling

pgBouncer - Failover setup

4 of 4





# Connection Pooling

pgPool

## 🔒 Authentication and Encryption

- 🔒 Ensuring that only authorized clients can access the database.
- 🔒 Extra layer of security between the application and the database.

## 🔒 High Availability (HA)

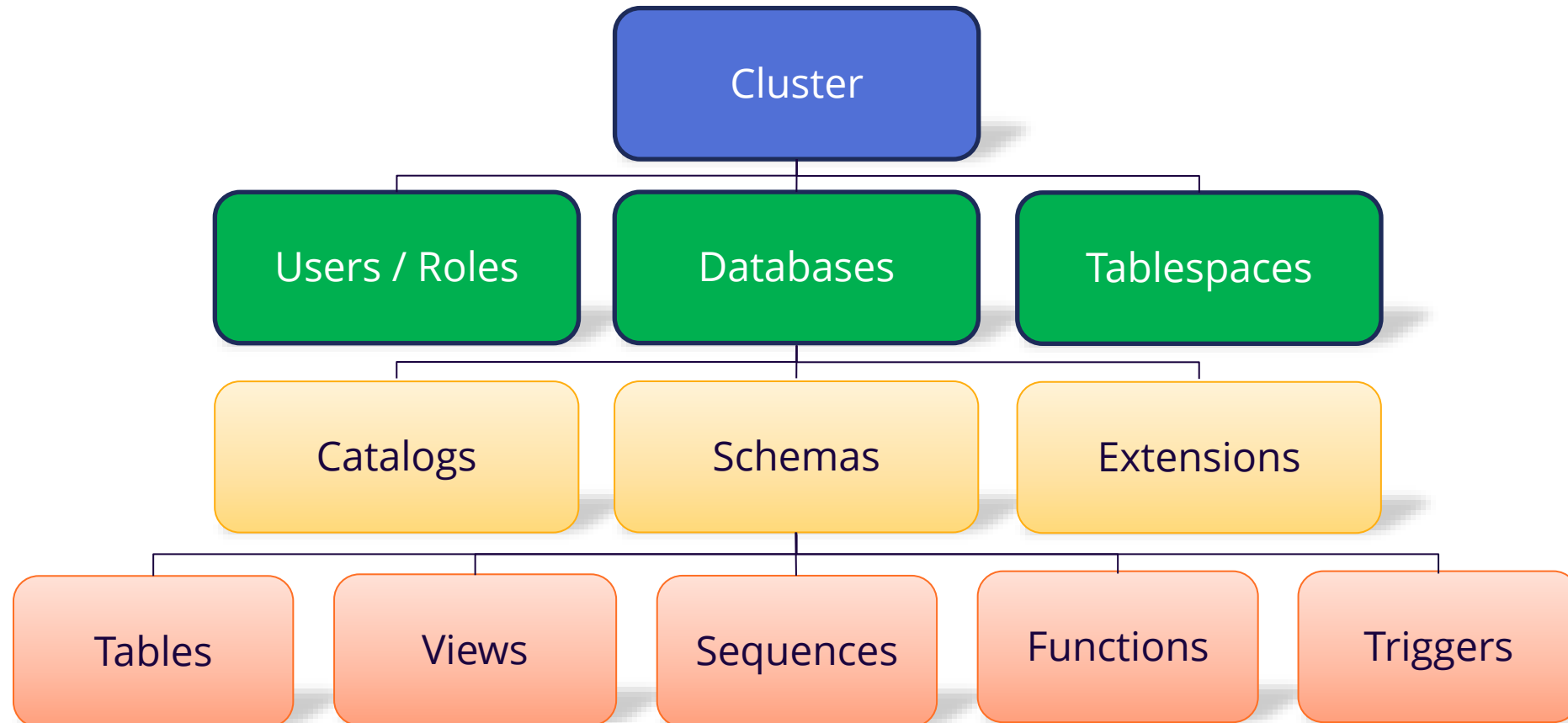
- 🔒 Includes features like automatic failover, online recovery, and watchdog, which help maintain database availability even if some servers go down.
- 🔒 Ensures that the applications can continue functioning without interruption.

## 🔒 Quorum Algorithm

- 🔒 Uses sophisticated quorum algorithm to avoid false positive errors and split-brain scenarios, making the HA system highly reliable

# PostgreSQL

## Structure



# PostgreSQL

## Database Security – Roles

1 of 3



### Limiting Databases Access

🔒 Principle of Least Privilege (PoLP)

🔒 Roles can be granted to other roles

🔒 Roles have specific permissions

🔒 Roles with fixed attributes – e.g.

🔒 LOGIN	↔	NOLOGIN
🔒 SUPERUSER	↔	NOSUPERUSER
🔒 INHERIT	↔	NOINHERIT
🔒 REPLICATION	↔	NOREPLICATION
🔒 CONNECTION LIMIT <amount>		
🔒 VALID UNTIL <time>		

```
CREATE ROLE role_name
[ WITH ]
[ SUPERUSER | NOSUPERUSER ]
[ CREATEDB | NOCREATEDB ]
[ CREATEROLE | NOCREATEROLE ]
[ INHERIT | NOINHERIT ]
[ LOGIN | NOLOGIN ]
[ REPLICATION | NOREPLICATION ]
[ BYPASSRLS | NOBYPASSRLS ]
[ CONNECTION LIMIT connlimit ]
[ PASSWORD 'password' | PASSWORD NULL ]
[ VALID UNTIL 'timestamp' ]
[ IN ROLE role_name [, ...] ]
[ IN GROUP role_name [, ...] ]
[ ROLE role_name [, ...] ]
[ ADMIN role_name [, ...] ]
[ USER role_name [, ...] ]
[ SYSID uid ]
```

```
ALTER ROLE "guy" SET statement_timeout=30min;
```



### Privilege Inquiry Functions

Function	Checks Privileges On
has_schema_privilege	Schemas
has_table_privilege	Tables, views, sequences
has_column_privilege	Specific columns in a table
has_function_privilege	Functions
has_language_privilege	Procedural languages
has_foreign_data_wrapper_privilege	Foreign data wrappers
has_server_privilege	Foreign servers
has_tablespace_privilege	Tablespaces
has_type_privilege	Data types → PostgreSQL 14+

```
SELECT DatName, UserName, has_database_privilege(UserName, DatName, 'CONNECT') AS conn_priv
FROM pg_database, pg_user
WHERE has_database_privilege(UserName, DatName, 'CONNECT') = 't'
  AND DatName = 'dwh'
ORDER BY DatName, UserName;
```

# PostgreSQL

## Database Security – Roles

3 of 3



🔧 Change user identifier

🔧 Use NOINHERIT

```
set role role_accounting;
```

🔧 Change `current_user` and `session_user`

🔧 Allows superuser to act like another user

```
set session authorization filip;
```

🔧 Modifying and removing objects is reserved for the owner and superusers

🔧 ALTER | DROP

# PostgreSQL

## Database Security – Default privileges



### ■ PostgreSQL can specify DEFAULT PRIVILEGES

```
ALTER DEFAULT PRIVILEGES
  [ FOR { ROLE | USER } target_role [, ...] ]
  [ IN SCHEMA schema_name [, ...] ]
  abbreviated_grant_or_revoke
```

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER | MAINTAIN }
  [, ...] | ALL [ PRIVILEGES ] }
ON TABLES
TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
REVOKE [ GRANT OPTION FOR ]
  { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER | MAINTAIN }
  [, ...] | ALL [ PRIVILEGES ] }
ON TABLES
FROM { [ GROUP ] role_name | PUBLIC } [, ...]
  [ CASCADE | RESTRICT ]
```

```
ALTER DEFAULT PRIVILEGES
  GRANT SELECT, INSERT, UPDATE
  on TABLES
  to accounting_team;
```



# PostgreSQL

## Row Level Security

---

- 🔒 Security policies are controlled at database level
- 🔒 Policies can be applied to roles or command or both
- 🔒 Only table owner is allowed to enable / disable and add policies
- 🔒 Not defined per default

```
alter table personnel enable row level security;
```

```
create policy only_manager personnel to  
role_manager using (Manager IS TRUE);
```



# PostgreSQL

## Column-Level Privileges

🔒 To avoid updating a specific column in a table

```
revoke update (salary) on table personnel from public;  
grant update (salary) on table personnel to role_hr;
```

🔒 To avoid SELECTing a certain column

```
CREATE TABLE employees (  
    id        SERIAL PRIMARY KEY,  
    name      TEXT,  
    email     TEXT,  
    salary    NUMERIC);
```

```
GRANT SELECT (name, email) ON employees TO analyst_usr;  
SELECT salary FROM employees; --> permission denied  
REVOKE SELECT (email) ON employees FROM analyst_usr;
```

# PostgreSQL



## Function Security

🔒 Permission to execute a function does not give permission on underlying objects

### 🔒 SECURITY INVOKER

🔒 Will execute with the privileges of the CALLER → is default

### 🔒 SECURITY DEFINER

🔒 Will execute with the privileges of the OWNER

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE lang_name
  | TRANSFORM { FOR TYPE type_name } [, ... ]
  | WINDOW
  | { IMMUTABLE | STABLE | VOLATILE }
  | [ NOT ] LEAKPROOF
  | { CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT }
  | { [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER }
  | PARALLEL { UNSAFE | RESTRICTED | SAFE }
  | COST execution_cost
  | ROWS result_rows
  | SUPPORT support_function
  | SET configuration_parameter { TO value | = value | FROM CURRENT }
  | AS 'definition'
  | AS 'obj_file', 'link_symbol'
  | sql_body
} ...
```

# PostgreSQL

## Data Encryption

1 of 2



Both **pgCrypto** and **pgSodium** are extensions that provide cryptographic functions.

pgCrypto	pgSodium
<ul style="list-style-type: none"><li>• Hashing Function</li><li>• Password Hashing</li><li>• PGP Encryption</li><li>• RAW Encryption</li><li>• Random Data</li></ul>	<ul style="list-style-type: none"><li>• High-Performance Cryptography operators</li><li>• Advanced Encryption</li><li>• Key Management</li><li>• SignCrypton</li><li>• Steaming Encryption</li></ul>
Dependencies: <i>OpenSSL</i>	Dependencies: libsodium

Integrating **pgCrypto** or **pgSodium** will require software changes.

# PostgreSQL

## Data Encryption

2 of 2



 Transparent data encryption (TDE)



 TDE is an optional feature in EDB Postgres Advanced and Extended Server



 TDE is part of CyberTec PostgreSQL Enterprise Edition (PGEE)



 RHEL supports Linux Unified Key Setup-on-disk-format, encrypts drive partition



 BitLocker for full disk encryption or Encrypting File System (EFS)

# PostgreSQL

## Auditing



Extension **pgAudit** provides detailed session and / or object audit logging.

### Detailed Logging

- Detailed information about database activities
- Including the exact SQL statements executed and their context

### Session and Object Auditing

- Session-level activities → user logins
- Object-level activities → table modifications

### Compliance Support

- Can help to be comply with regulatory requirements

### Integration with PostgreSQL

- Uses native logging facility
- Integrated with the standard logs → CSV, JSON, SYSLOG, PLAIN TEXT

# PostgreSQL

## Backups

1 of 3



### 🔧 Data Loss Prevention

- 🔧 Human errors, hardware failures, disasters.
- 🔧 Restore the database to a previous state.

### 🔧 Disaster Recovery

- 🔧 Outage, failed upgrade, corrupted hardware.
- 🔧 Ensures recovery and minimize downtime.

### 🔧 Data Integrity

- 🔧 Restore if needed to a consistent state.

### 🔧 Compliance and Auditing

- 🔧 According regulations

# PostgreSQL

## Backups

2 of 3



### Testing, Development, Training

-  Mirroring production



### *Anonymizer*

-  Static Masking → Permanent
-  Dynamic → Role based



### pg\_datamask

-  Generic Masking → Ready-Made functions
-  Custom-Built → Write own functions



```
CREATE TABLE persons (  
    id                SERIAL,  
    first_name        TEXT,  
    last_Name         TEXT,  
...);  
  
SECURITY LABEL FOR anon COLUMN  
    persons.Last_Name IS  
    'MASKED WITH FUNCTION anon.fake_last_name()';
```



# PostgreSQL

## Backups

3 of 3

Both	BaRMAN 	pgBackRest 
<ul style="list-style-type: none"><li>• PITR</li><li>• Compression</li><li>• Local or Remote</li><li>• Repository</li><li>• Checksums</li><li>• FULL, INCR, DIFF</li></ul>	<ul style="list-style-type: none"><li>• gzip, bzip2, and lzma</li><li>• Catalogue</li></ul>	<ul style="list-style-type: none"><li>• lz4, zstd</li><li>• Multiple Repositories</li><li>• Parallel</li><li>• <b>Encryption</b></li></ul>

### Incremental Backups

-  Using *pg\_basebackup* in PostgreSQL v17 by enabling WAL summarization.
-  Restore via *pg\_combinebackup* to merge previous base backups.



# PostgreSQL

## Monitoring & Health Check

---

- 🔧 Logging & Auditing & Activity
  - 🔧 e.g. *pg\_stat\_statements* & *pg\_stat\_replication*
- 🔧 Strong Authentication
  - 🔧 e.g. SCRAM-SHA-256 instead of MD5
- 🔧 Role-Based Access
  - 🔧 e.g. Least privilege
- 🔧 Encryption
  - 🔧 e.g. SSL / TLS & encryption of sensitive data
- 🔧 Updates / Patching
  - 🔧 also the extensions
- 🔧 Network Security
  - 🔧 Only trusted hosts
- 🔧 Backup & Recovery

# Questions

About Database Security

---



# Contact us:



[info@zebanza.be](mailto:info@zebanza.be)



+32 497 31 91 60



Veldkant 33a | 2550 Kontich

