dbi services by Sequotech

PostgreSQL Security – The defense line for your data

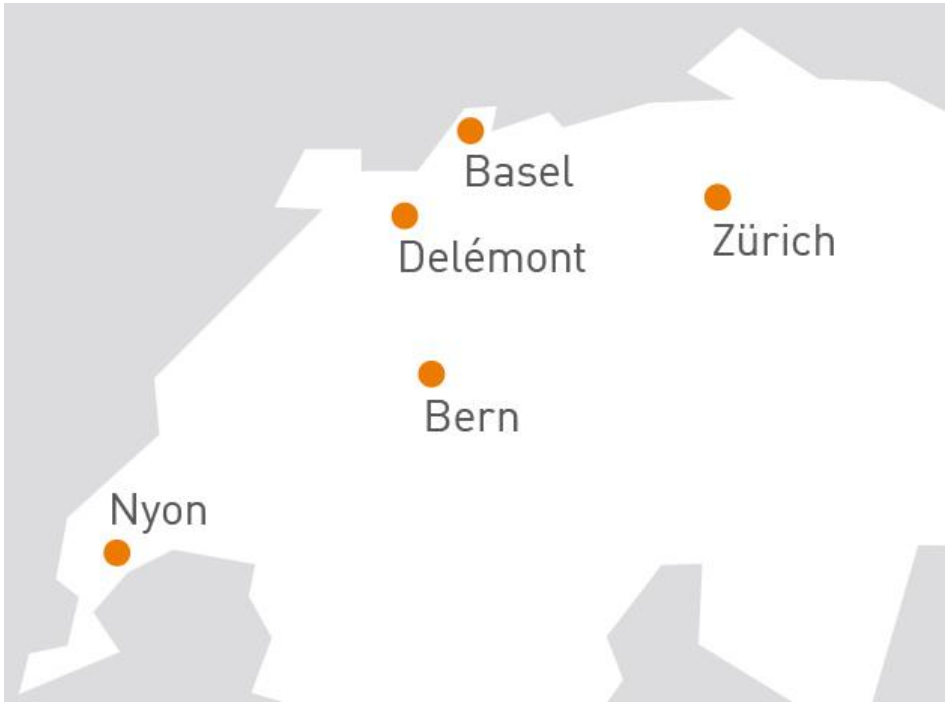
Who we are

The Company

> Founded in 2010
> More than 100 employees
> Specialized in the Middleware Infrastructure
> The invisible part of IT
> Customers in Switzerland and all over Europe

Our Offer

> Consulting
> Service Level Agreements (SLA)
> Trainings
> License Management



swiss safety center ISO 20000 certified system

swiss safety center ISO 27001 certified system

About me

Julia Gugel

Delivery Manager

Senior Consultant

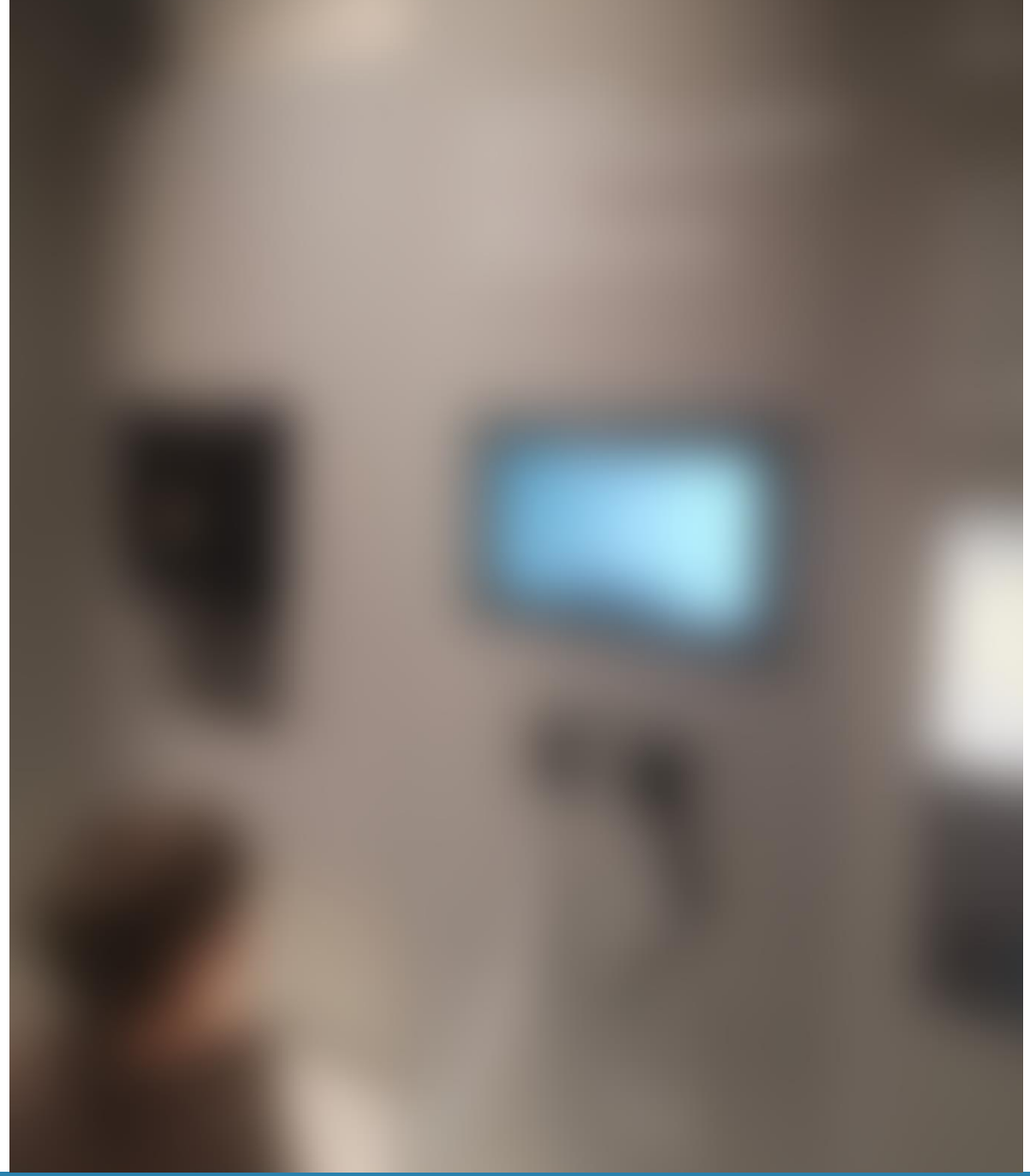
+41 78 320 43 07

julia.gugel[at]dbi-services.com

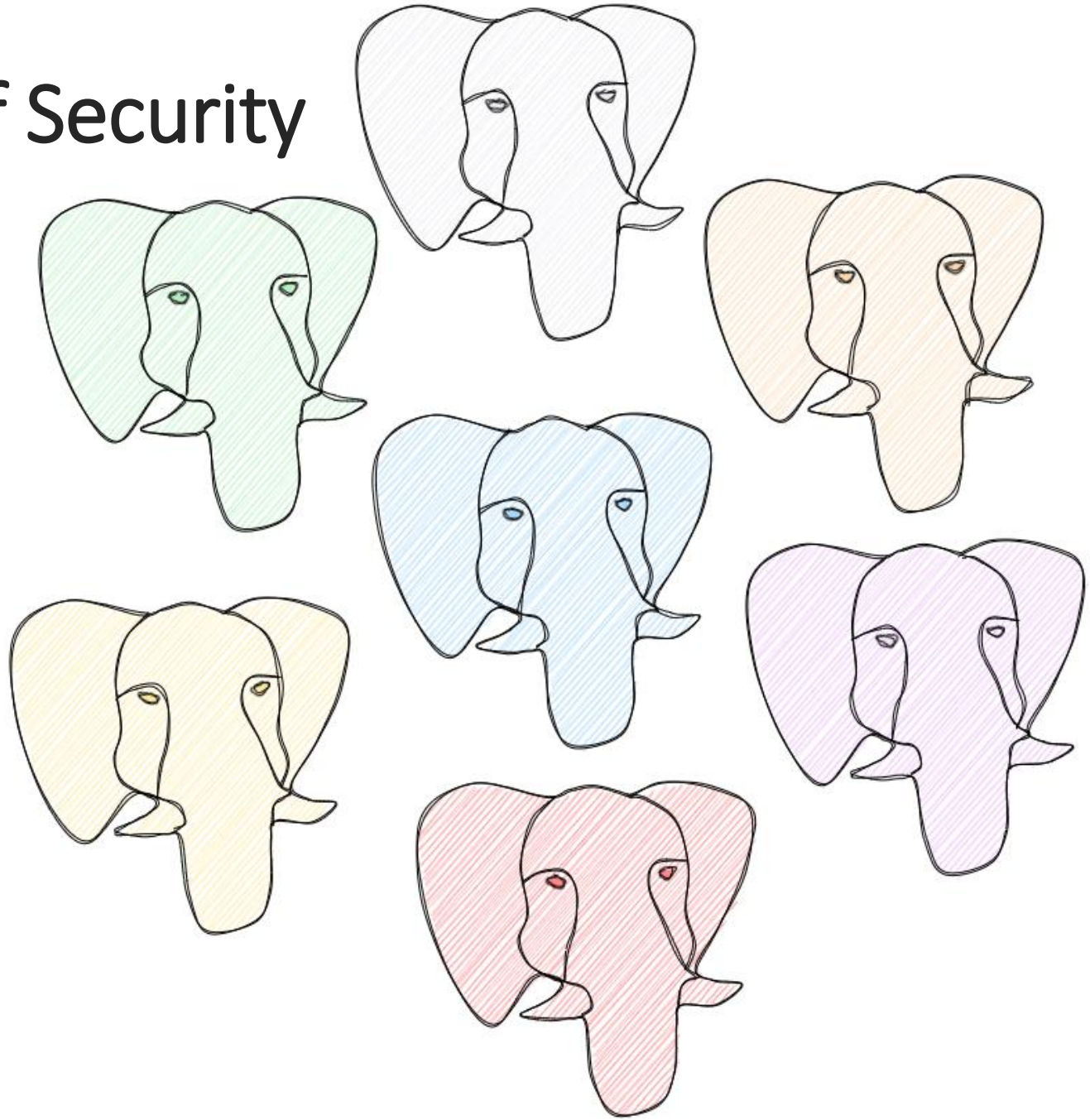
Agenda

1. The idea
2. Variations of Security
3. Let’s do it
4. Conclusion

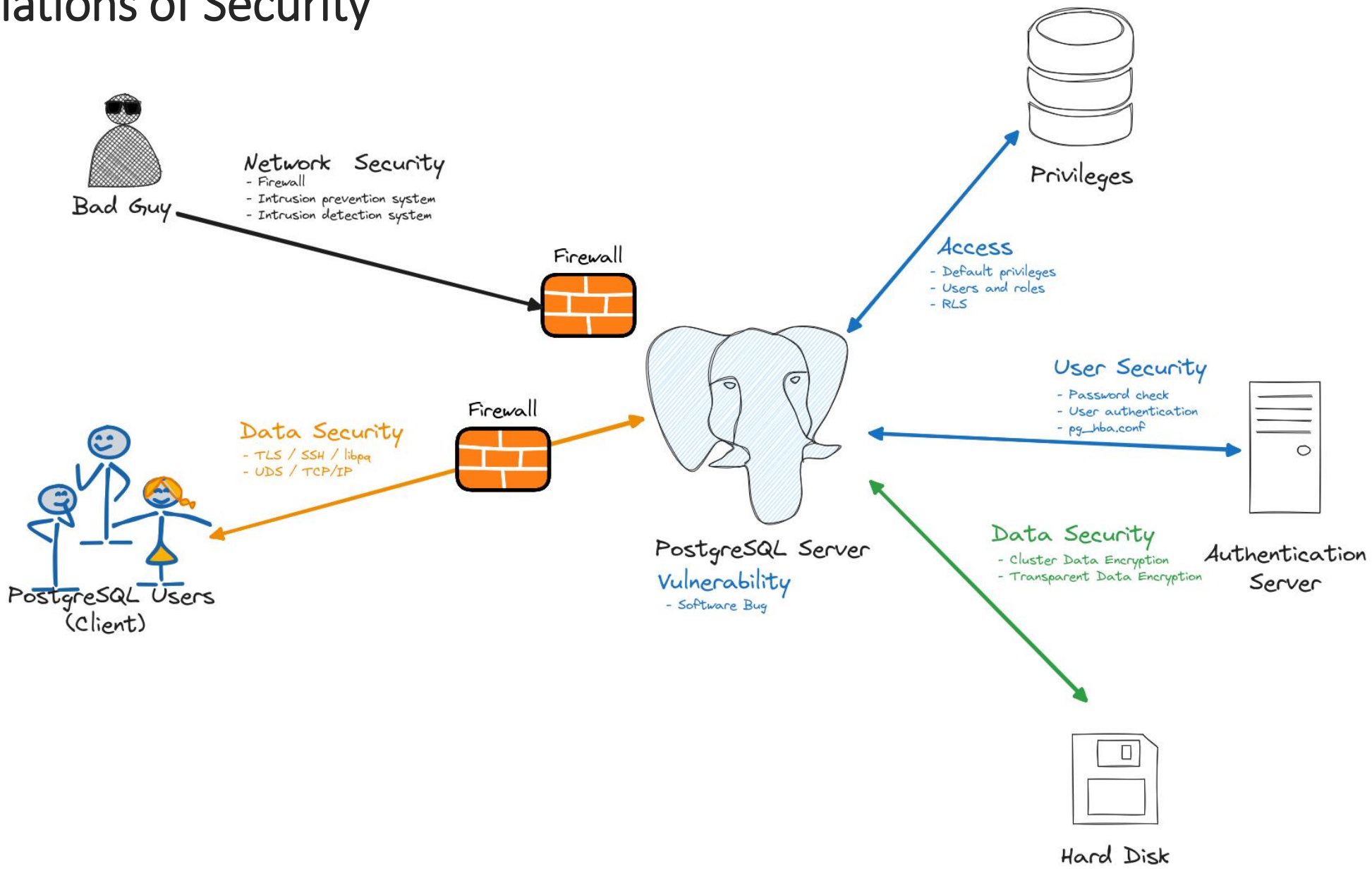
The idea



Variations of Security



Variations of Security



Variations of Security
Access Security - Authentication

pg_hba.conf

```
# "local" is for Unix domain socket connections only
local                   all             all                                     trust
# IPv4 local connections:
host                    all             all             127.0.0.1/32            trust
```

> Everyone that can login to the server can become superuser and login to your PostgeSQL cluster

> Only allow temporarily to reset the password

Variations of Security
Access Security - Authentication

pg_hba.conf

```
# "local" is for Unix domain socket connections only
local                   all             all                                     peer
# IPv4 local connections:
host                    all             all             127.0.0.1/32            ident
```

- peer
  - Users are authenticated by the underlying Operating System
  - Only for local connections
- ident
  - Network connections
  - Needs an ident server running on the client

Variations of Security
Access Security - Authentication

pg_hba.conf

```
# IPv4 connections:
host            all             all             10.159.41.0/32          md5
host            all             all             10.159.42.0/32          scram-sha-256
```

- md5
  - Prevents password sniffing
  - Password not stored in plain text on the server
  - No protection if attacker steals password hash
- scram-sha-256
  - Prevents password sniffing
  - Password is cryptographically hashed
  - Most secure method at the moment

Variations of Security
Access Security - Authentication

pg_hba.conf

```
# IPv4 connections:
host    all     all     10.159.41.0/24          gss include_realm=1 krb_realm=AD.kittyCAT.CH
host    all     all     10.159.41.0/32          gss include_realm=1 krb_realm=AD.kittyCAT.CH
```

- kerberos
  - Setup kerberos service on client and PostgreSQL Server
    - krb5-workstation or krb5-server
  - /etc/krb5.conf on client and server machines
  - Create a keytab file and verify it on the server

```
krb_server_keyfile=/home/…/postgres.PG1.kittycat.ch.keytab
```

  - Create a user

```
postgres=# create user "kitty@AD.kittycat.ch" superuser;
CREATE ROLE
```

Variations of Security
Access Security - Authentication

pg_hba.conf

```
# IPv4 connections:
host    all     all     10.159.41.0/24          ldap ldapserver=10.159.12.222
ldapbasedn="cn=Users,dc=kittycat,dc=local"
ldapbinddn="CN=ldap,CN=Users,dc=kittycat,dc=local" ldapbindpasswd="'cittiesPW!"
ldapsearchattribute="sAMAccountName"
```

- LDAP
  - ldapserver=192.168.55.200 : LDAP Server
  - ldapbasedn="cn=Users,dc=kittycat,dc=local" : starting point in the domain's hierarchy for your search
  - ldapbinddn="CN=ldap,CN=Users,dc=samplecompany,dc=local" : ldap user, which will authenticate to AD to perform the searches
  - ldapbindpasswd : Password
  - ldapsearchattribute="sAMAccountName" : the AD attribute that will be searched for

Variations of Security
Access Security - Authentication

pg_hba.conf - be precise

```
# IPv4 connections:
host            all             all             192.159.241.12/32       scram-sha-256
hostssl         db1             cat             192.159.242.23/32       scram-sha-256
hostssl         db3             kitty           10.15.2.0/24            scram-sha-256
hostssl         db3             all             10.15.2.123/32          scram-sha-256
```

- host:database:user:ip/range:authentication method
  - Define as specific as possible
- hostssl
  - Use SSL connections whenever possible
- https://www.postgresql.org/docs/current/auth-pg-hba-conf.html

Variations of Security
Access Security - Authentication

pg_hba.conf - new PG16

```
# IPv4 connections:
local       db1,"/^db\d{2,4}$",db2          all                 localhost         trust
```

- Allows connections to
  - Databases db1 and db2
  - Any databases with a name beginning with "db" and finishing with a number using 2 to 4 digits
- Pattern matching must start with a slash "/"
- Can be used for columns database, user and address
- In addition include files are also supported as of PostgreSQL 16
  - include / include_if_exists / include_dir

Variations of Security
Access Security – Connection Restriction

UDS – Unix domain socket

- Accessible from machines
- Can be controlled by permissions
- PostgreSQL can create multiple sockets
- Unix_socket_directory= /tmp, /home/postgres
  - Empty: only TCP possible

TCP/IP

- Access from remote system
- By default, only listening to localhost
  - listen_addresses = “REQUIRED_NETWORK_ADDRESSES”

Variations of Security
Database Security - Roles

Limit database access

- Roles can be granted to other roles
- Roles have specific permission
- Roles with fixed attributes
  - LOGIN
  - SUPERUSER
  - REPLICATION
  - CREATEDB
- SUPERUSER bypass all permission check
  - Don’t use for daily work
- Don’t grant permissions to users directly

```
postgres=# alter role cat with createdb;
ALTER ROLE
```

Variations of Security
Database Security - Roles

SET ROLES

> Change user identifier

```
postgres=# set role cat;
SET
```

> Comparable to «sudo su – user»

> Use NOINHERIT to prevent inheriting privileges

SET SESSION AUTHORIZATION

> Changes current_user and session_user

```
postgres=# set session authorization cat;
SET
```

> Allows superuser to act like another user

Variations of Security
Database Security – ACL

Attached to objects (tables/functions/views or columns)

> Select, insert, execute

> ACL shows an additional entry for each privilege, if it can be granted to others and who granted

```
kitty= arwdDxt/kitty   =r/kitty   animals_team=arw/kitty
```

kitty has all permissions on the table

Granted by kitty when created the table

Public/ everyone has read permission

Granted by kitty

Animals teams has INSERT, SELECT and UPDATE

Granted by kitty

a - insert r - select w - update d - delete D - truncate x - references t - trigger

Variations of Security
Database Security – ACL

GRANT / REVOKE ACLs

> Mostly: only the owner has privileges

> Always: drop and modify an object is reserved for the owner and superusers

```
postgres=# \dp cattab
                                 Access privileges
 Schema |  Name   | Type  |   Access privileges   |   Column privileges   | Policies
--------+---------+-------+-----------------------+-----------------------+----------
 public | cattab  | table | kitty =arwdDxt/kitty +| col1:                +|
        |         |       | =r/kitty             +|   kitty_rw=rw/kitty   |
        |         |       | animals_team=arw/kitty|                       |
(1 row)
```

Variations of Security
Database Security – Default Privileges

In other databases you would need to grant access to each new object

PostgreSQL comes with default privileges

- Granting access to new objects in advance, before they even exist
- No need to worry about grants in the future
- Certain privileges are automatically granted to roles
- Only new objects not for existing

```
postgres=# ALTER DEFAULT PRIVILEGES
                GRANT INSERT, SELECT, UPDATE
                on TABLES
                to animals_team;
```

Variations of Security

Database Security – Default Privileges

Because existing objects will not be affected when default privileges are modified

> Adjust the default privileges right from the beginning

> So you never need to worry about it in the future

Default privileges exist for

> select, insert, update, delete, truncate, references, trigger, all for tables

> usage, select, update, all for sequences

> execute, all for functions and routines

> usage, all for types

> usage, create, all for schemata

> revoke for all above

Variations of Security
Database Security – Row Security Policy

Per-User basic restriction

> Which row can be returned by normal queries

Not defined per default

```
postgres=# alter table cats_data enable row level security;
ALTER TABLE
```

Policies can be applied to roles or commands or both

Table owner only is allowed to enable/disable and add policies

Policies need a unique name

```
postgres=# CREATE POLICY data_managers ON cats_data TO kitty_managers
    USING (b = 'value123');
CREATE POLICY
```

Variations of Security
Database Security – Security functions

Permission to execute a function does not give permissions on the underlying objects

```
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
    | SET configuration_parameter { TO value | = value | FROM CURRENT }
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  } ...
```

Variations of Security
Database Security – Security functions

SECURITY INVOKER

- The function will execute with the privileges of the user who calls it
  - This is the default

SECURITY DEFINER

- The function will execute with the privileges of the user who owns it
- The owner has full access to the tables

Variations of Security
Database Security – Encryption

pgcrypto

- Extension – included as a contrib module
- Provides SQL functions for hashing and encryption

```
postgres=# CREATE EXTENSION pgcrypto;
CREATE EXTENSION
```

- Hashing
  - Hash sensitive data e.g. passwords
  - Check if the checksum is correct
  - One way – hashed data stays hashed

Variations of Security
Database Security – Encryption

pgcrypto

- Encryption
  - Store data in a secure way but retrieve it anyway
  - Offers symmetric and
  - PGP functions
    - Generate a key and export it
    - Use the public key to encrypt
    - Use the private key to decrypt
  - PGP encryption can be a risk when sharing the key with too much people
- PGP encryption for exchanging dates with others
- Symmetric for self-contained application

Variations of Security
Transport Encryption

Transport Layer Security (TLS)

- Encrypt the TCP traffic
  - PostgreSQL needs a server certificate and a key protected by a passphrase
- Either asked at startup or
  - ssl_passphrase_command
- Use server cert and key
  - ssl_cert_file / ssl_key_file
- Use CA files
  - ssl_ca_cile / ssl_crl_file

Variations of Security
Restrict Connections - Firewall

Define inbound and outbound rules

- Protocol
- Local port
- Source address

Define a program

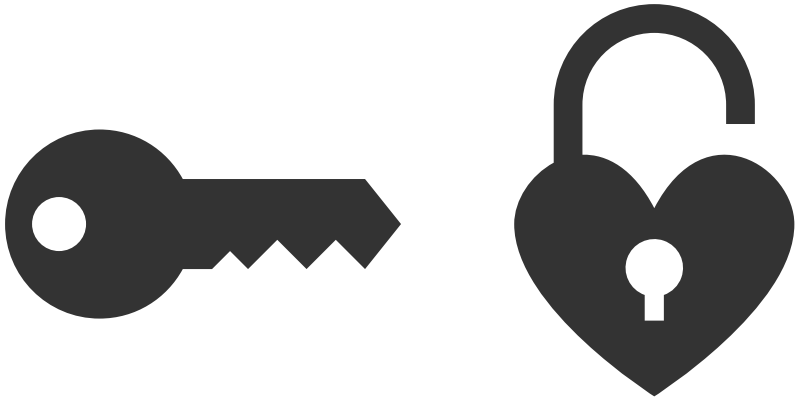
Nftables

Use tools to administrate easier

Variations of Security
Filesystem Security

Data Partition Encryption

- File system level (linux)
  - eCryptfs
  - EncFS
- Block level (linux)
  - dm-crypt
  - LUKS
- Many other operating systems support this
- As long as the file system is mounted, the data is unencrypted



Variations of Security
Access Rules – Server room

Variations of Security
Access Rules – Server room

Server in a secure facility

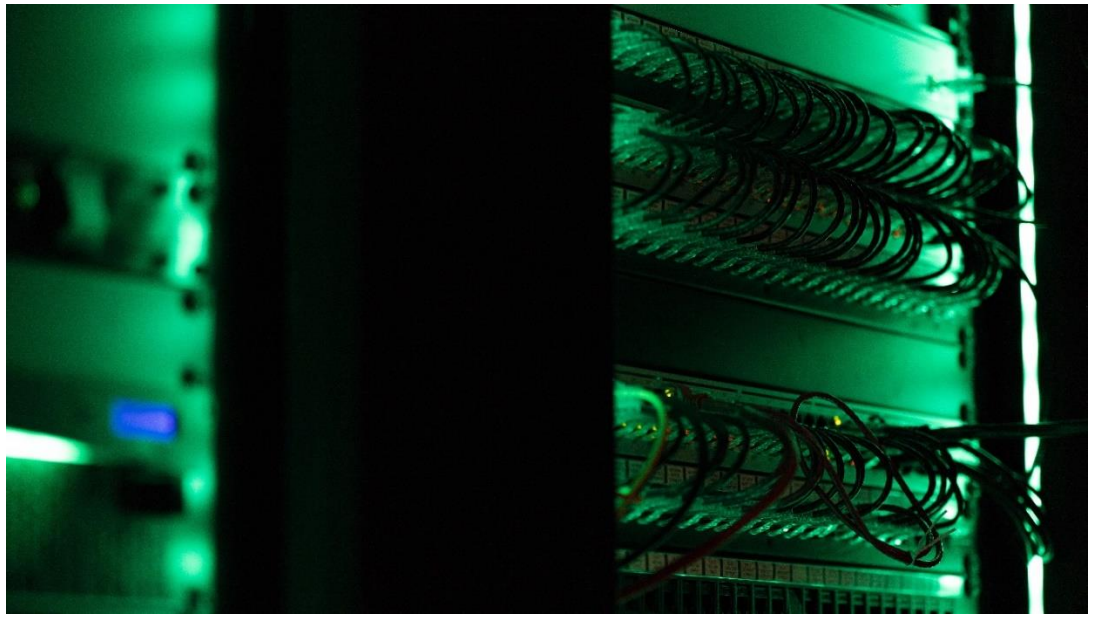
Strict security policy

> Who is allowed to enter the server room

Locked racks

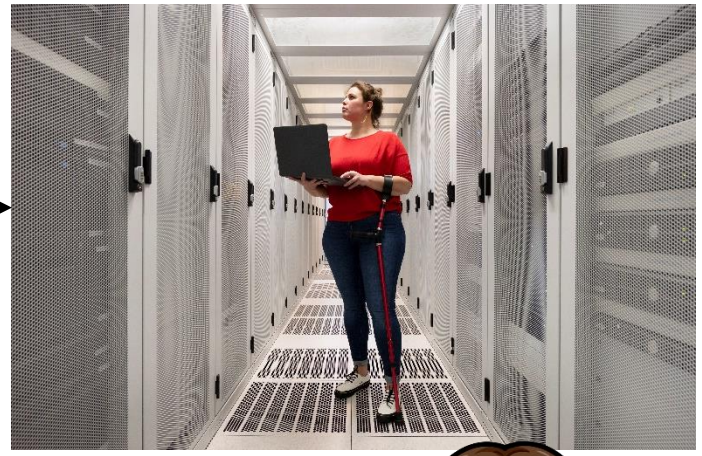
> Access to keys

> Locker



Variations of Security
Access Rules – Server room

ACCESS DENIED



Variations of Security
Vulnerabilities

Known PostgreSQL Security Vulnerabilities in Supported Versions

You can filter the view of patches to show just patches for version:
16 - 15 - 14 - 13 - 12 - 11 - all

Reference	Affected	Fixed	Component & CVSS v3 Base Score	Description
CVE-2023-39418 Announcement	15	15.4	core server 3.1 AV:N/AC:H/PR:L/UI:N/S:U/C:N/I:L/A:N	MERGE fails to enforce UPDATE or SELECT row security policies more details
CVE-2023-39417 Announcement	15, 14, 13, 12, 11	15.4, 14.9, 13.12, 12.16, 11.21	core server 7.5 AV:N/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H	Extension script @substitutions@ within quoting allow SQL injection more details
CVE-2023-2455 Announcement	15, 14, 13, 12, 11	15.3, 14.8, 13.11, 12.15, 11.20	core server 4.2 AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:L/A:N	Row security policies disregard user ID changes after inlining more details
CVE-2023-2454 Announcement	15, 14, 13, 12, 11	15.3, 14.8, 13.11, 12.15, 11.20	core server 7.2 AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H	CREATE SCHEMA ... schema_element defeats protective search_path changes more details
CVE-2022-41862 Announcement	15, 14, 13, 12	15.2, 14.7, 13.10, 12.14	client 3.7 AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	Client memory disclosure when connecting, with Kerberos, to modified server more details
CVE-2022-2625 Announcement	14, 13, 12, 11	14.5, 13.8, 12.12, 11.17	core server 7.1 AV:N/AC:H/PR:L/UI:R/S:U/C:H/I:H/A:H	Extension scripts replace objects not belonging to the extension more details
CVE-2022-1552 Announcement	14, 13, 12, 11	14.3, 13.7, 12.11, 11.16	core server 8.8 AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	Autovacuum, REINDEX, and others omit "security restricted operation" sandbox more details

Variations of Security
Vulnerabilities

The PostgreSQL Global Development Group (PGDG) takes security seriously.

Allows users to place their trust in PostgreSQL for protecting their mission-critical data.

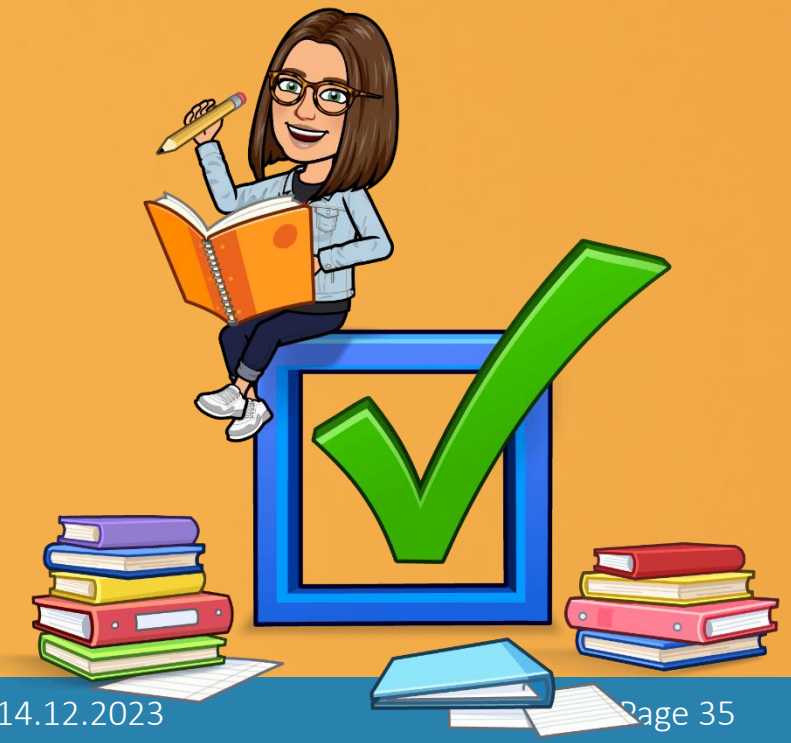
Shares responsibility between PostgreSQL itself and its deployment environment

> Including hardware, operating system and the application layer.

> In case you find a Security Vulnerability → security@postgresql.org

Vulnerabilities are fixed as part of minor version updates

Make sure your PostgreSQL software is up to date



Let’s do it

Demo
Commands

```
/*****************************************
* CREATE USERS
*****************************************/
CREATE USER ignazio with login password 'ignazio';
CREATE USER viola  with login password 'viola';
CREATE USER alain  with login password 'alain';
CREATE USER elisabeth with login password 'elisabeth';
CREATE USER heidi with login password 'heidi';
CREATE USER pierre with login password 'pierre';
CREATE USER didier with login password 'didier';
CREATE USER chantal with login password 'chantal';
CREATE USER verena with login password 'verena';
CREATE USER damien with login password 'damien';
CREATE USER alec with login password 'alec';
```

Demo
Commands

```
/*****************************************
* CREATE ROLES
*****************************************/
CREATE ROLE federal_council;
CREATE ROLE cantonal_council;
CREATE ROLE district_admin;
CREATE ROLE major;
CREATE ROLE connect_db;
```

Demo
Commands

```
/*****************************************
* GRANT ROLES TO USERS;
*****************************************/
GRANT connect_db to federal_council;
GRANT connect_db to cantonal_council;
GRANT connect_db to district_admin;
GRANT connect_db to major;
GRANT federal_council TO ignazio;
GRANT federal_council TO viola;
GRANT federal_council TO alain;
GRANT federal_council TO elisabeth;
GRANT cantonal_council TO heidi;
GRANT cantonal_council TO pierre;
GRANT district_admin TO didier;
GRANT district_admin TO chantal;
```

Demo
Commands

```
/*****************************************
* GRANT ROLES TO USERS
*****************************************/
GRANT major TO verena;
GRANT major TO damien;
GRANT major TO alec;

/*****************************************
* ALL NEEDED USERS CREATED?
*****************************************/
SELECT * FROM pg_catalog.pg_user;
```

Demo
Commands

```
/*****************************************
* CREATE DATABASE HABITANTS
*****************************************/
CREATE DATABASE habitants;

/*****************************************
* GRANT PERMISSIONS TO ROLES
*****************************************/
GRANT CONNECT ON DATABASE habitants TO connect_db;
GRANT pg_read_server_files TO federal_council;

/*****************************************
* CONNECT ignazio TO DATABASE habitants
*****************************************/
\c habitants ignazio
```

Demo
Commands

```
/*****************************************
* CREATE TABLE
*****************************************/
CREATE TABLE historical_data (
   year          int,
   town_no       int,
   town_name     text,
   district_no   int,
   district_name text,
   canton_no     int,
   canton        text,
   unit          text,
   number        int
);
```

Demo
Commands

```
/*****************************************
* OUPS.....we need to grant permission
* CONNECT AS POSTGRES again
*****************************************/
\c postgres postgres

/*****************************************
* CHANGE OWNER OF habitants
*****************************************/
ALTER DATABASE habitants OWNER TO federal_council;

/*****************************************
* CONNECT ignazio TO DATABASE habitants
* ***************************************/
\c habitants ignazio
```

Demo
Commands

```
/*****************************************
* RETRY TO CREATE TABLE
*****************************************/
CREATE TABLE historical_data (
   year          numeric,
   town_no       numeric,
   town_name     text,
   district_no   numeric,
   district_name text,
   canton_no     numeric,
   canton        text,
   canton_name   text,
   unit          text,
   number        numeric
);
```

Demo
Commands

```
/*****************************************
* INSERT DATA INTO TABLE
*****************************************/
COPY historical_data(year,town_no,town_name,district_no,district_name, canton_no, canton,
       canton_name, unit, number)
FROM '/home/postgres/historical_data.csv' DELIMITER ';' CSV HEADER;

/*****************************************
* CHECK THE ACLs
*****************************************/
 \dp historical_data
```

Demo
Commands

```
/*****************************************
* CHECK THE DATA
*****************************************/
SELECT * FROM historical_data LIMIT 5;

/*****************************************
* CONNECT elisabeth TO DATABASE habitants
* \c habitants elisabeth
*****************************************/
\c habitants elisabeth

/*****************************************
* CAN ELISABETH SELECT AS WELL?
*****************************************/
SELECT * FROM historical_data LIMIT 5;
```

Demo
Commands

```
/*****************************************
* CONNECT AS ignazio AGAIN AND GRANT PERMISSION
* EVEN IF IT'S POSSIBLE TO GRANT TO USER DIRECTLY,
* IT SHOULD BE AVOIDED
*****************************************/
\c habitants ignazio
GRANT SELECT ON historical_data TO elisabeth WITH GRANT OPTION;

/*****************************************
* CONNECT as POSTGRES
* GRANT ignazio TO elisabeth;
*****************************************/
\c habitants postgres
GRANT ignazio TO elisabeth;
```

Demo
Commands

```
/*****************************************
* CAN ELISABETH SELECT NOW?
* CONNECT elisabeth TO DATABASE habitants
*****************************************/
\c habitants elisabeth
SELECT * FROM historical_data LIMIT 5;



/*****************************************
* GRANT SELECT TO ALL OTHERS
*****************************************/
GRANT SELECT ON historical_data TO cantonal_council;
GRANT SELECT ON historical_data TO district_admin;
```

Demo
Commands

```
/*****************************************
* SET ROLE
*****************************************/
SET ROLE ignazio;
GRANT SELECT ON historical_data TO major;


/*****************************************
* CHECK THE ACLs
*****************************************/
\dp historical_data
```

Demo
Commands

```
/*****************************************
* ONE PERMISSION IS MISSING
*****************************************/
\c postgres postgres
GRANT SELECT ON historical_data TO federal_council;

/*****************************************
* SET SESSION AUTHORIZATION
*****************************************/
SET SESSION AUTHORIZATION ignazio;
GRANT ALL ON historical_data TO federal_council;
```

Demo
Commands

```
/*****************************************
* CHECK THE ACLs
*****************************************/
\dp historical_data

/*****************************************
* CREATE A DUPLICATE OF THE FIRST TABLE
*****************************************/
\c habitants ignazio
CREATE TABLE history_renewed AS TABLE historical_data;

/*****************************************
* CHECK THE ACLs
*****************************************/
\dp history_renewed
```

Demo
Commands

```
/*****************************************
* DEFINE DEFAULT PRIVILEGES
*****************************************/
ALTER DEFAULT PRIVILEGES GRANT INSERT, SELECT, UPDATE, DELETE ON TABLES TO federal_council;
ALTER DEFAULT PRIVILEGES GRANT INSERT, SELECT, UPDATE ON TABLES TO cantonal_council;
ALTER DEFAULT PRIVILEGES GRANT INSERT, SELECT ON TABLES TO district_admin;
ALTER DEFAULT PRIVILEGES GRANT SELECT ON TABLES TO major;

/*****************************************
* CREATE ANOTHER DUPLICATE OF THE FIRST TABLE
*****************************************/
DROP TABLE history_renewed;
CREATE TABLE history_renewed AS TABLE historical_data;
```

Demo
Commands

```
/*****************************************
* CHECK ACL and DEFAULT PRIVILEGES
*****************************************/
\dp history_renewed
\ddp

/*****************************************
* SELECT DATA AS A MAJOR
*****************************************/
\c habitants verena
SELECT * FROM history_renewed LIMIT 5;
```

Demo
Commands

```
/*****************************************
* DEFINE ROW SECURITY POLICY
*****************************************/
\c habitants Ignazio

ALTER TABLE history_renewed ENABLE ROW LEVEL SECURITY;

CREATE POLICY gipf_oberfrick ON history_renewed TO verena
       USING (town_name='Gipf-Oberfrick');
CREATE POLICY berne ON history_renewed TO alec USING (town_name='Bern');
CREATE POLICY delemont ON history_renewed TO damien USING (town_name='Delémont');
CREATE POLICY canton_ag ON history_renewed TO cantonal_council USING (canton='AG');
```

Demo
Commands

```
/*****************************************
* CHECK ROW LEVEL SECURITY
*****************************************/
SELECT relname, relrowsecurity, relforcerowsecurity
  FROM pg_class
  JOIN pg_catalog.pg_namespace n ON n.oid = pg_class.relnamespace
  WHERE n.nspname = 'public' AND relkind = 'r';



SELECT relname, relrowsecurity, relforcerowsecurity
FROM pg_class
WHERE oid = 'history_renewed'::regclass;
```

Demo
Commands

```
/*****************************************
* SELECT DATA AS A MAJOR
*****************************************/
\c habitants verena
SELECT * FROM history_renewed LIMIT 5;

/*****************************************
* SELECT DATA AS A MAJOR
*****************************************/
\c habitants alec
SELECT * FROM history_renewed LIMIT 5;
```

Demo
Commands

```
/*****************************************
* USE PGCRYPTO HASHING
*****************************************/
\c habitants ignazio
CREATE EXTENSION pgcrypto;
SELECT * FROM history_renewed where town_name='Hilfikon' limit 5;
UPDATE history_renewed SET town_name=crypt('Hilfikon',gen_salt('md5'))
       where town_name='Hilfikon';
SELECT * FROM history_renewed where town_name='Hilfikon' limit 5;
SELECT * FROM history_renewed where town_no='4070' limit 5;
```

Conclusion

Conclusion

Logging

> Database log

> OS log

> Network log

Password

> Only basic password check for password complexity

> No password profiles

Make sure your PostgreSQL software is up to date

There is already a lot that can be defined to secure your data

Conclusion

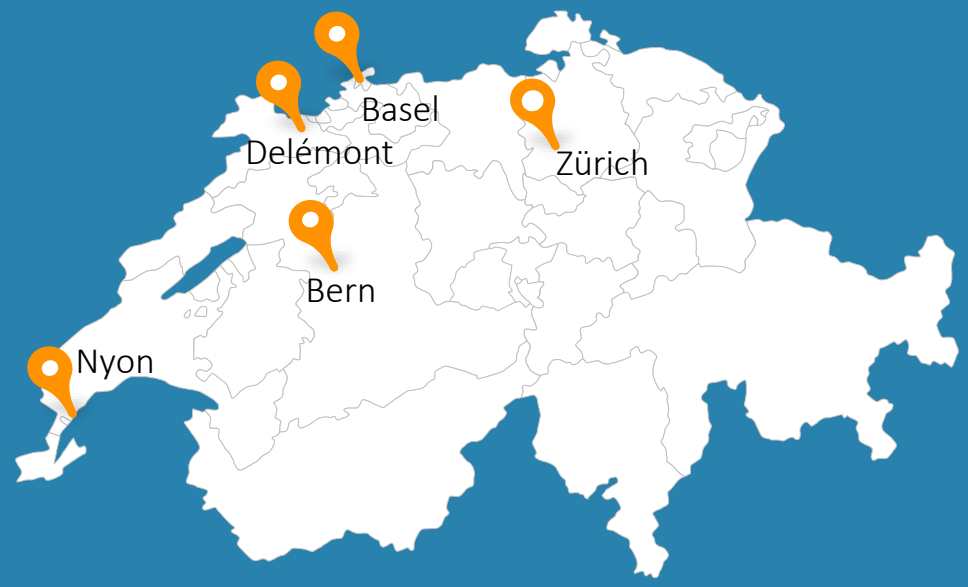
PostgreSQL offers a high level of data security per default

- Users cannot connect per default
- You cannot load data from a file even when you're an admin
- pg_hba.conf makes the entry hurdle higher
  - Cannot connect without an entry

Always take care of your server security as well

Any questions?

Please do ask!



We would love to boost your IT-Infrastructure

How about you?