initMAX

Enterprise solution in PostgreSQL: efficient and flexible access management

initMAX

About initMAX s.r.o.

Your partner in Zabbix monitoring, **PostgreSQL solutions**, Wazuh and OpenSearch

ZABBIX
CERTIFIED TRAINER

ZABBIX
PREMIUM PARTNER

CYBERTEC
POSTGRESQL SERVICES & SUPPORT
OFFICIAL PARTNER

OpenSearch

wazuh.

Certified PostgreSQL training partner

SQL BASICS

INTRODUCTION TO POSTGRESQL

POSTGRESQL PROFESSIONAL

ADMINISTRATION AND PERFORMANCE TUNING

HIGH AVAILABILITY & PATRONI

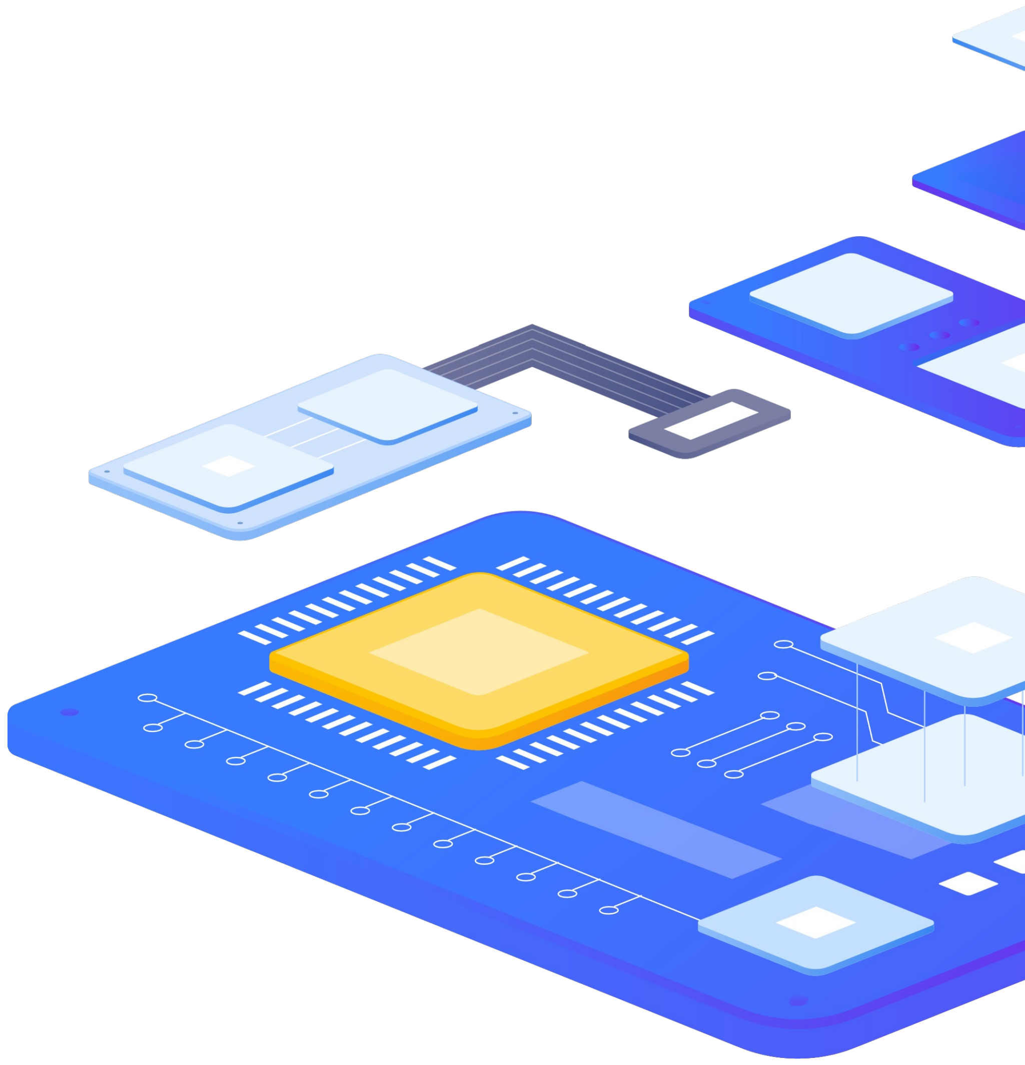
INTRODUCTION TO POSTGIS

MIGRATION TUTORIAL

initMAX

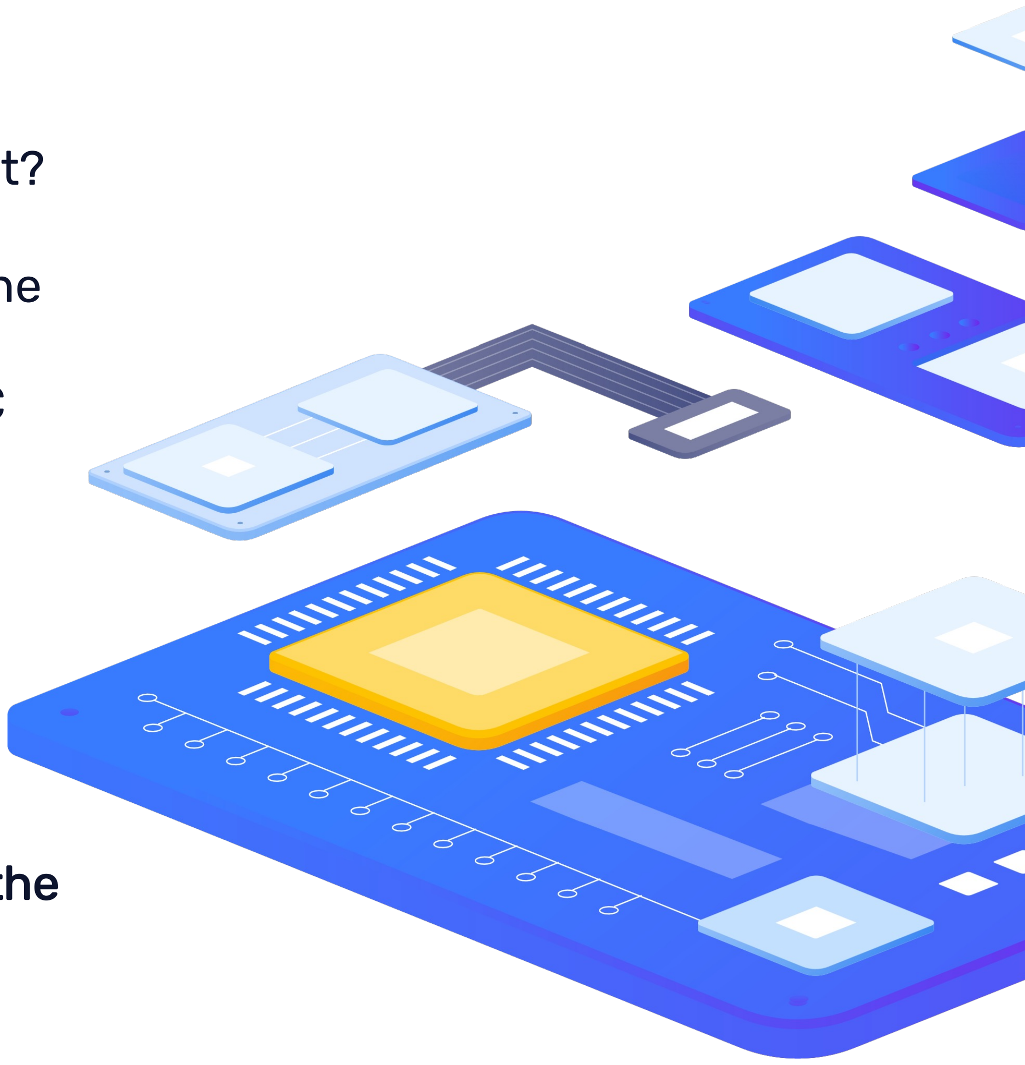
Introduction

- PostgreSQL supports 11 authentication methods; the basic ones are:
  - **Trust authentication**, which simply trusts that users are who they say they are.
  - **Password Authentication,** which requires users to authenticate with a password.
  - **LDAP Authentication**, which relies on an LDAP authentication server.
  - **PAM authentication,** which relies on PAM (Pluggable Authentication Modules) library.
  - **Certificate authentication**, which requires an SSL connection and authenticates the user by checking the received SSL certificate.
  - **GSSAPI authentication**, which relies on a GSSAPI-compatible library. It is typically used to access an authentication service such as FreeIPA or Microsoft Active Directory and uses the Kerberos protocol.



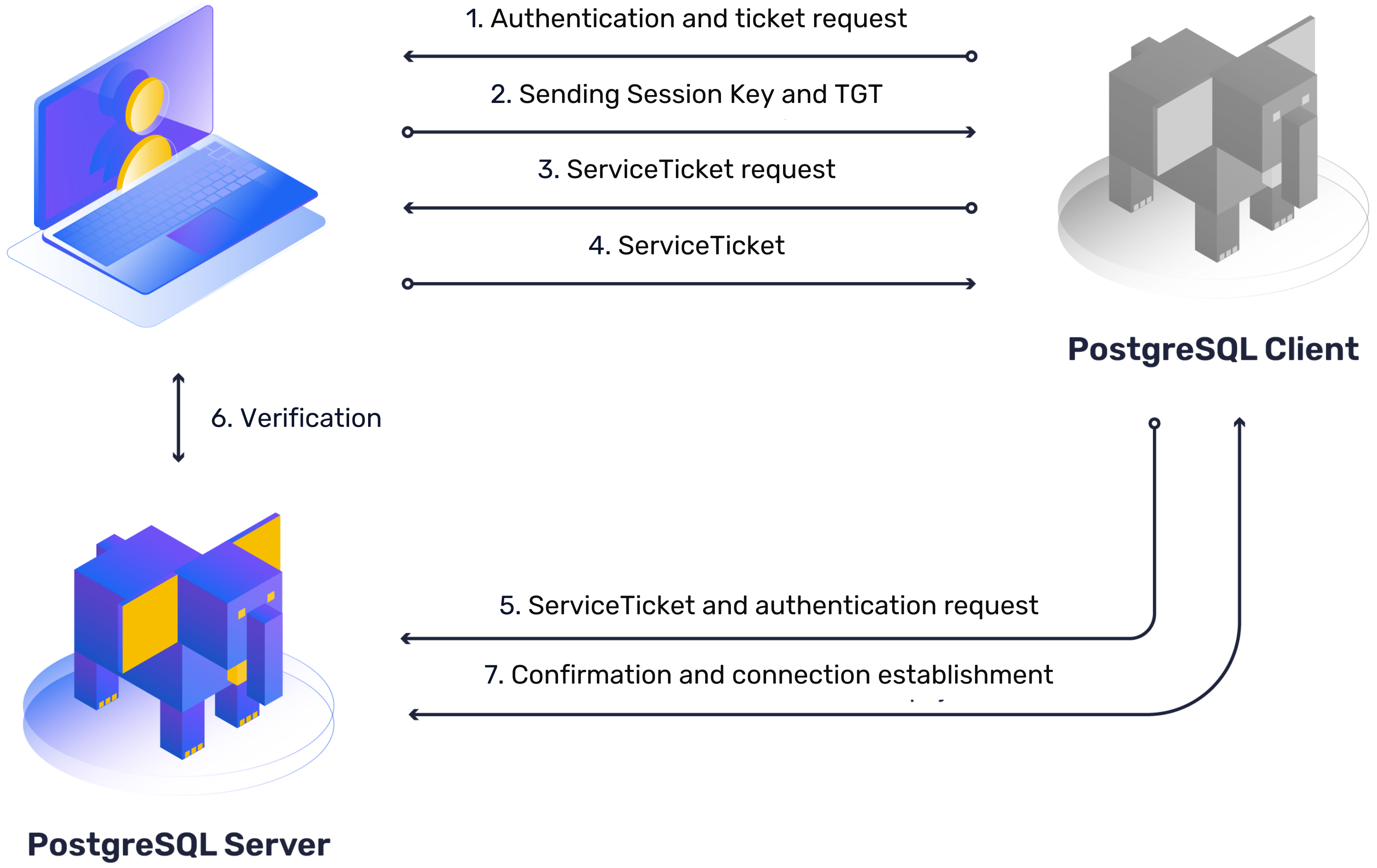
Introduction

- What is Kerberos, how does it work and why is it good to use it?
  - Kerberos is a network authentication protocol, which serves for secure authentication of both the client and the server
  - The client authenticates itself against a third party - KDC (Key Distribution Center)
  - No passwords are sent over the network, nor are they stored locally on the client
  - Strong encryption algorithms are used
  - The KDC is a central element and can provide services to many applications and clients
  - Access can be controlled from one place
  - **Failure of the central authentication service may affect the operation of multiple systems**



Enterprise solution in PostgreSQL: efficient and flexible access management

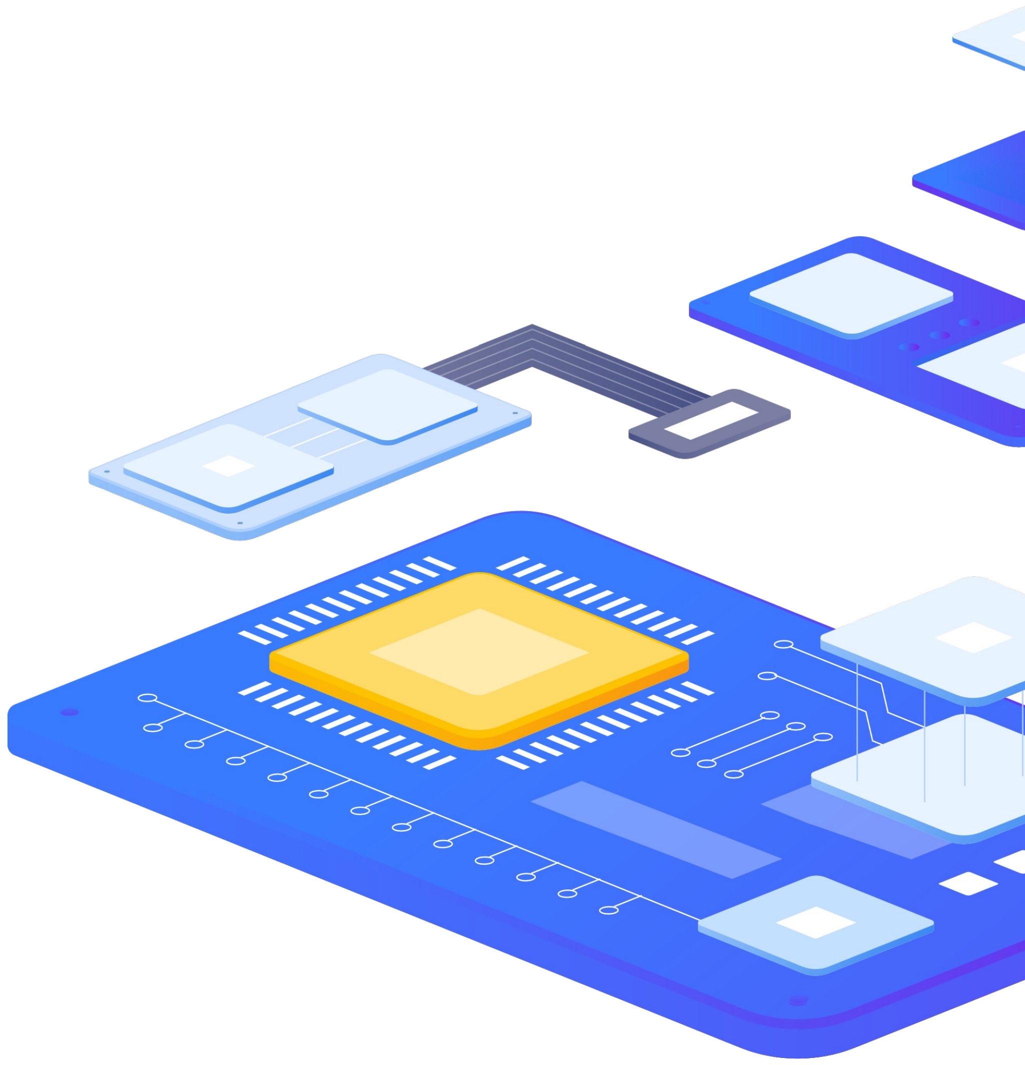
initMAX



1. Authentication and ticket request
2. Sending Session Key and TGT
3. ServiceTicket request
4. ServiceTicket
PostgreSQL Client
6. Verification
5. ServiceTicket and authentication request
7. Confirmation and connection establishment
PostgreSQL Server

Basic requirements

- Installed PostgreSQL server
- Kerberos support and configuration
  - krb5-workstation & krb5-server
  - /etc/krb5.conf
- User account for PostgreSQL in Active Directory
- Generated keytab for the DB server
- PostgreSQL configuration
  - pg_hba.conf
  - postgresql.conf
- User account in PostgreSQL with required privileges
- Kerberos ticket for DB user



Kerberos support and configuration

- The necessary libraries must be installed on the server and support for Kerberos must be set up
- Installation of required packages

```
dnf install krb5-server krb5-workstation
```

- Configuring Kerberos support for the client
  - Editing the file /etc/krb5.conf (see example)
  - Editing must be done by the root user

```
[logging]
    default = /var/log/krb5libs.log
    kdc = /var/log/krb5kdc.log
    admin_server = /var/log/kadmind.log

[libdefaults]
    default_realm = INITMAX.LOCAL
    dns_lookup_realm = false
#    ticket_lifetime = 24h
#    renew_lifetime = 7d
    forwardable = true
    udp_preference_limit = 1
    default_ccache_name = KEYRING:persistent:%{uid}

[realms]
INITMAX.LOCAL = {
     kdc = ad.initmax.local
     admin_server = ad.initmax.local
}

[domain_realm]
.initmax.local = INITMAX.LOCAL
initmax.local = INITMAX.LOCAL
```

User account for DB server in AD and keytab

- In Active Directory, create a service account for the database server - for example pg_gitlab_srv01
- Next, you need to generate a **Kerberos keytab** linked to the account from the previous step on the Active Directory server

```
ktpass -princ postgres/pg.initmax.local@INITMAX.LOCAL -pass heslo -
mapuser pg_gitlab_srv01 -crypto ALL -ptype KRB5_NT_Principal -out keytab
```

- We copy the keytab obtained in this way to the DB server, for example in the /etc/postgres directory
- And we can verify its functionality on the PotgreSQL server

```
klist -k /etc/postgres/keytab
kinit -k -t /etc/postgres/keytab postgres/pg.initmax.local@INITMAX.LOCAL -V
Using existing cache: 0
Using principal: postgres/pg.initmax.local@INITMAX.LOCAL
Using keytab: /etc/postgres/keytab
Authenticated to Kerberos v5
```

PostgreSQL configuration

- In the configuration file of the PostgreSQL server, modify the parameter **krb_server_keyfile**

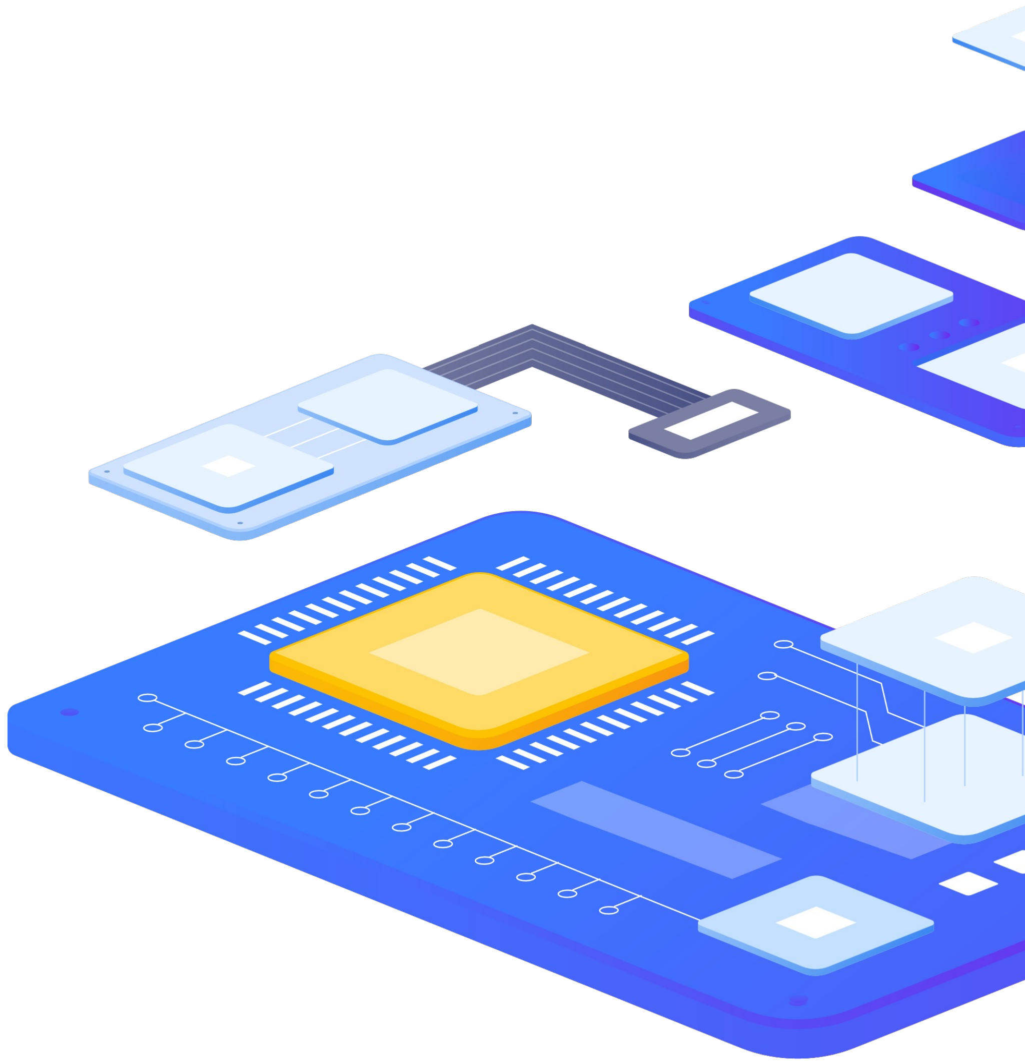
```
krb_server_keyfile=/etc/postgres/keytab
```

- In the **pg_hba.conf** file, enable login using the GSSAPI method

```
# IPv4 local connections:
host   all   all   0.0.0.0/0      gss include_realm=0
krb_realm=INITMAX.LOCAL
```

- And create a user in PostgreSQL
  - The user must match a real user in AD

```
pgdemo=# create user "pgusera" superuser;
```



Login to PostgreSQL

- Getting a ticket from Active Directory

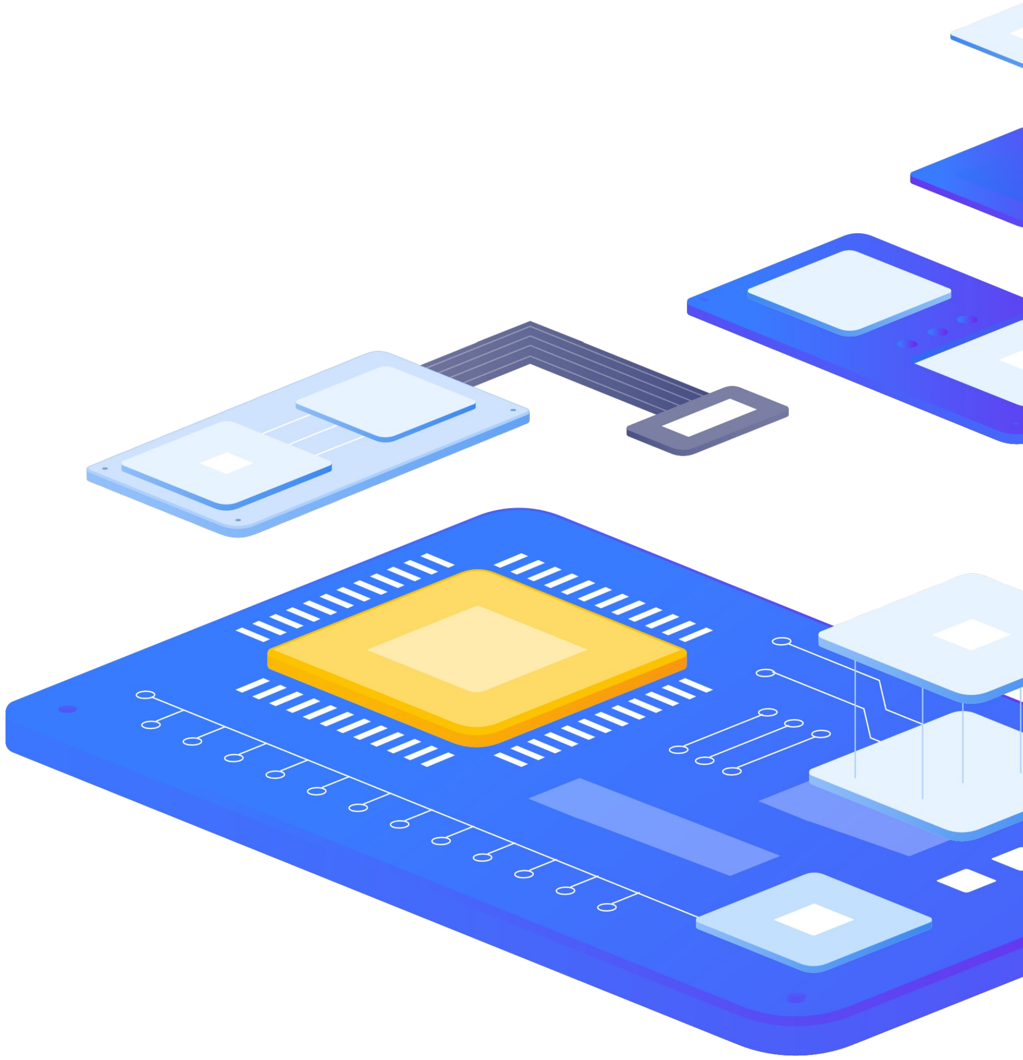
```
kinit pgusera
```

- Login to PostgreSQL

```
psql -U pgusera -h pg.initmax.local a
```

- In larger environments, user creation can be automated
- For example, a combination of the following can be used
  - LDAP (Active Directory, FreeIPA, OpenLDAP,…)

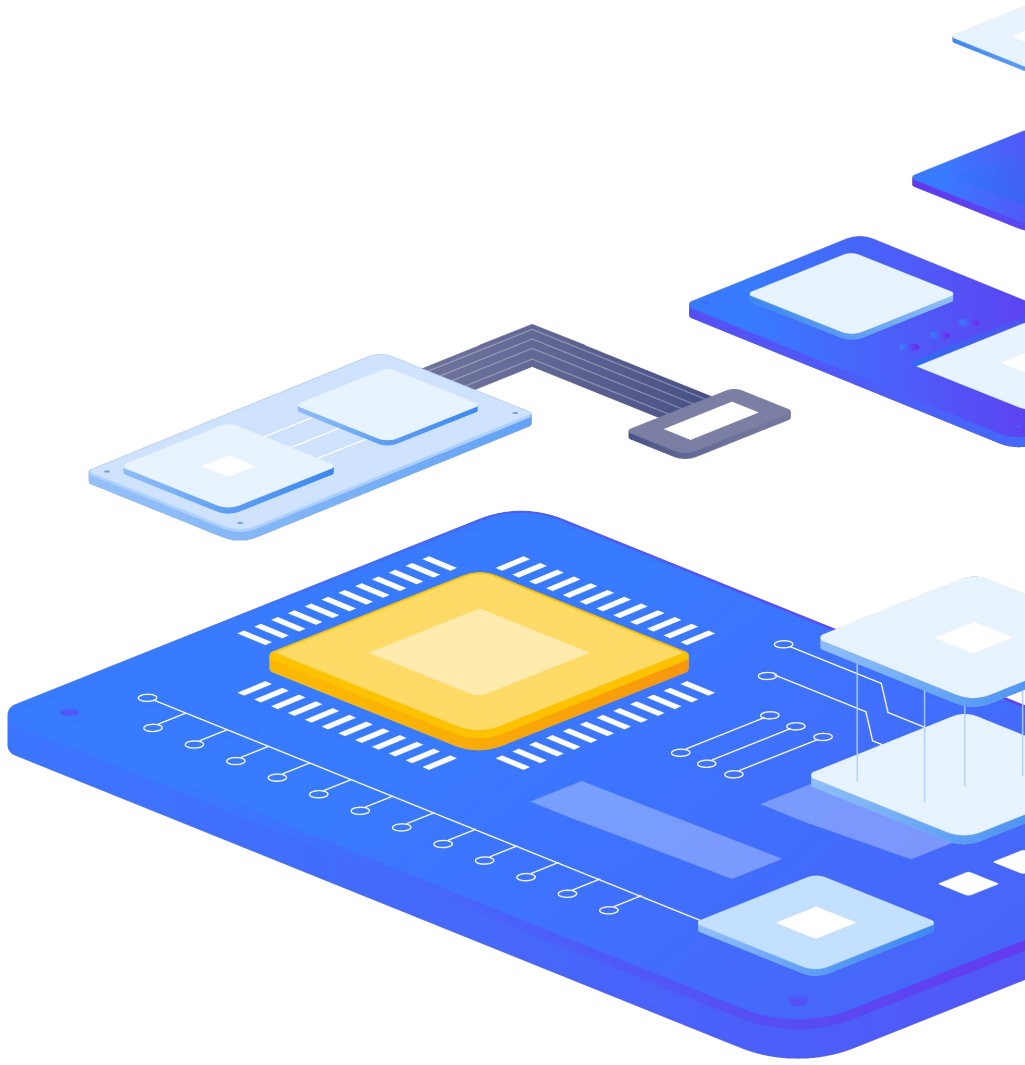
  and
  - ldap2pg



initMAX

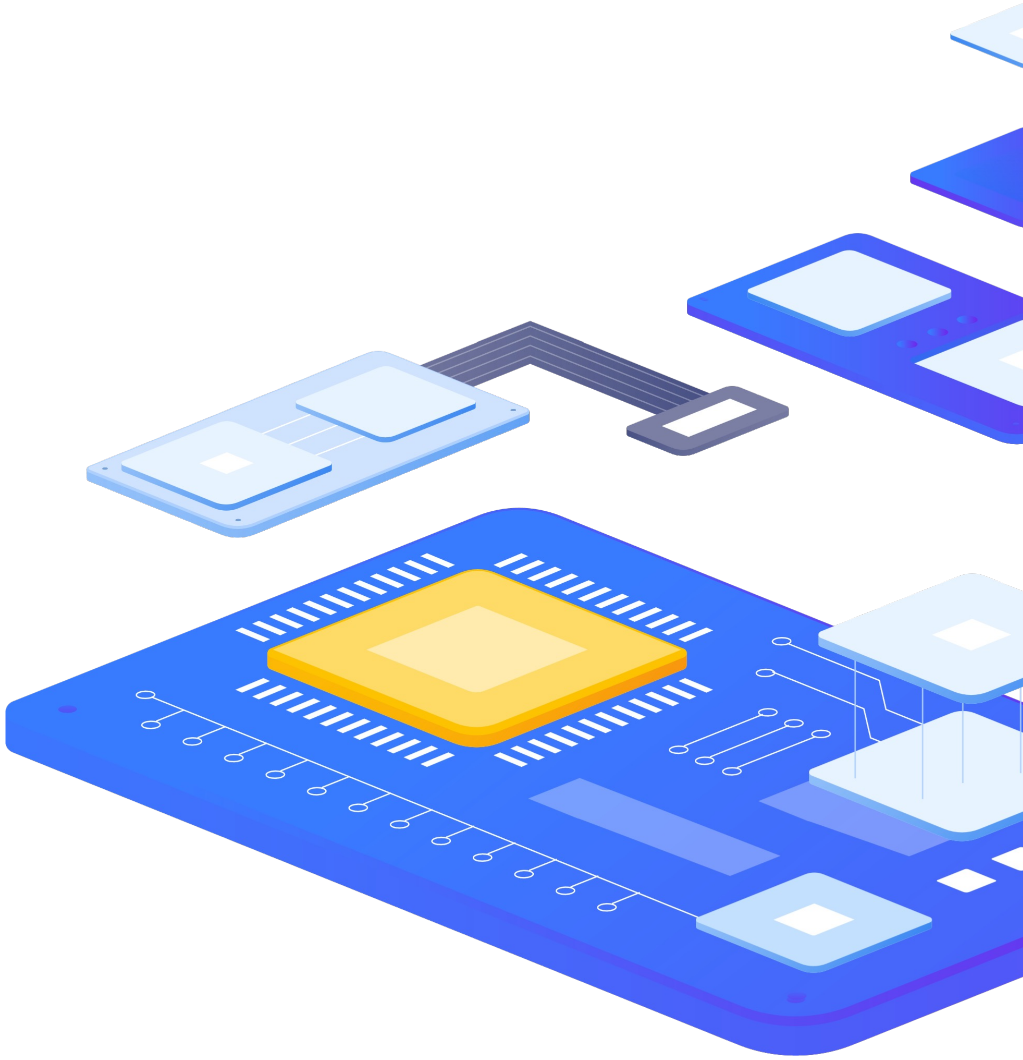
ldap2pg

- ldap2pg automates the creation, update and removal of PostgreSQL roles
- A YAML file is used for configuration
- Creates, changes and deletes roles in PostgreSQL according to settings in LDAP
- Can set or remove permissions statically or according to LDAP settings
- Can manage role membership
- Performs validation of the settings before its deployment use **--real** parameter to apply changes



ldap2pg – installation

- ldap2pg is available as a Python package starting from version 6 ldap2pg is rewriten in go with no dependencies
- ldap2pg python requires:
  - Python 2.6+ or Python 3.4+
  - Pyyaml
  - python-ldap
  - python-psycopg2
- The authors recommend using distribution packages both for installing dependencies and for ldap2pg itself, if available.

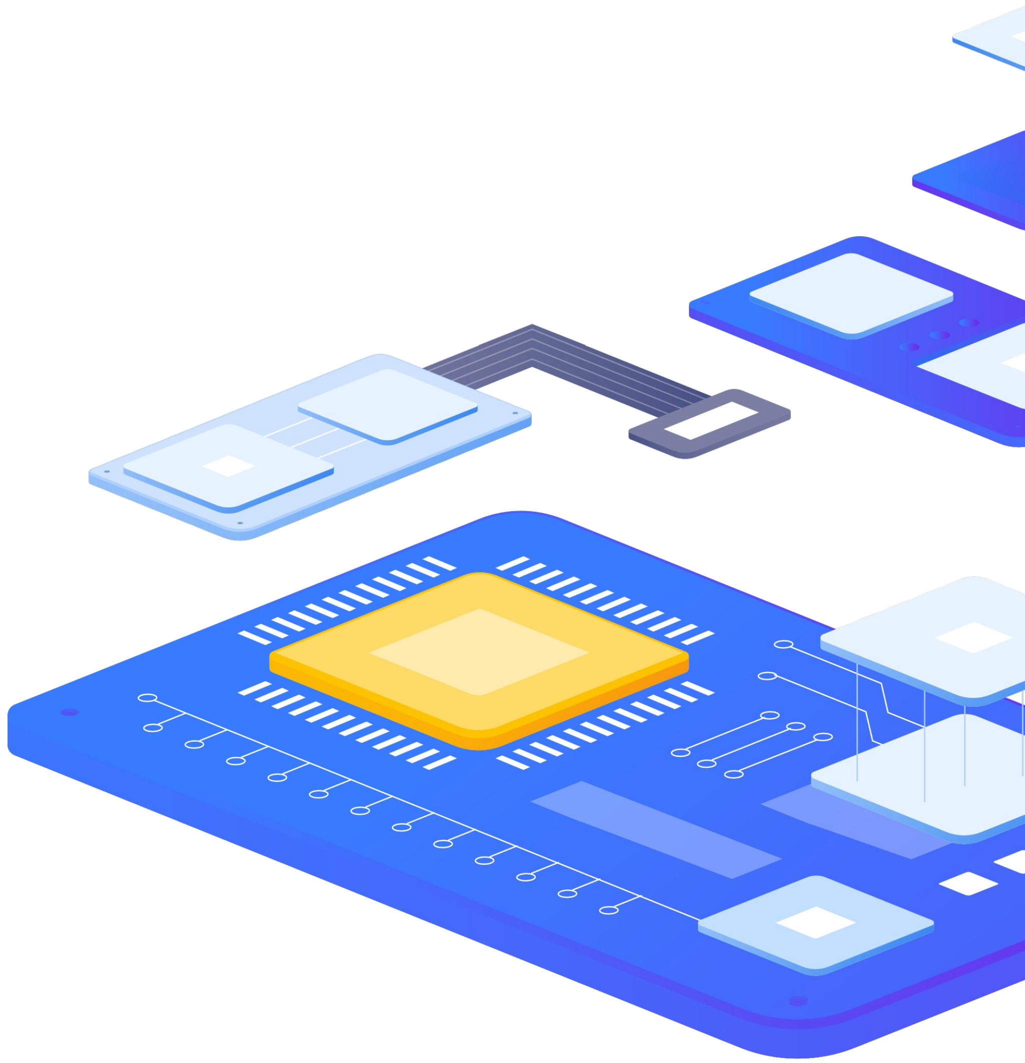


initMAX

ldap2pg – installation

- Download binary for your target system and architecture from release page
- Move the binary to /usr/local/bin.
- Ensure it's executable
- Test installation with ldap2pg --version

```
$ ldap2pg --version
ldap2pg 6.1
github.com/jackc/pgx/v5 v5.5.5
github.com/go-ldap/ldap/v3 v3.4.8
gopkg.in/yaml.v3 v3.0.1
go1.22.1 linux amd64
```



ldap2pg – installation from repository

- Guide for RHEL 6/7/8/9 compatible and Dalibo Labs YUM repository
  - Install the repository and refresh dnf cache

```
dnf install -y https://yum.dalibo.org/labs/dalibo-labs-4-1.noarch.rpm
dnf makecache fast
```

  - The repository can also be added manually

```
vi /etc/yum.repos.d/dalibolabs.repo

[dalibolabs]
name = Dalibo Labs - RHEL/CentOS/Rockylinux $releasever -
$basearch
baseurl = https://yum.dalibo.org/labs/RHEL$releasever-$basearch
gpgcheck = 1
enabled = 1

dnf makecache fast
```

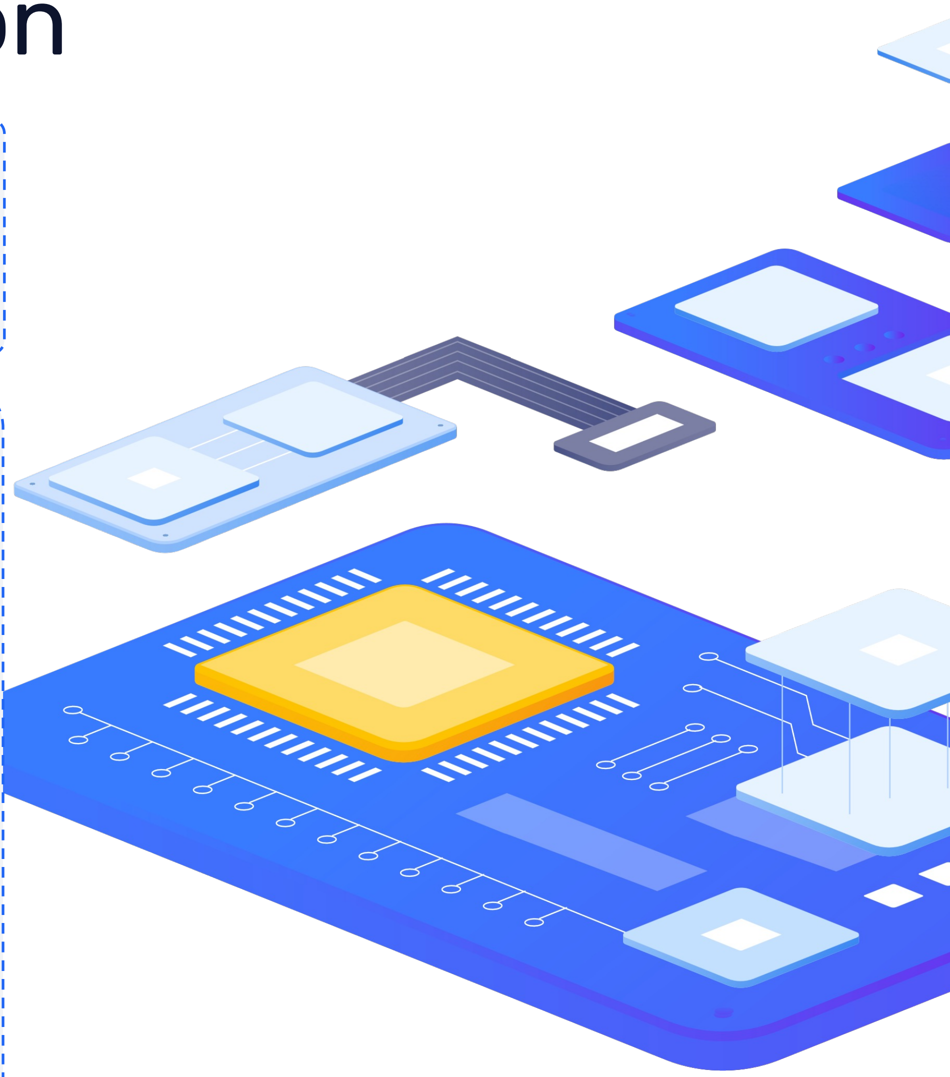
  - Install ldap2pg itself

```
dnf install ldap2pg
```

ldap2pg – verifying the installation

```
ldap2pg -V
ldap2pg 6.1
github.com/jackc/pgx/v5 v5.5.5
github.com/go-ldap/ldap/v3 v3.4.8
gopkg.in/yaml.v3 v3.0.1
go1.22.1 linux amd64
```

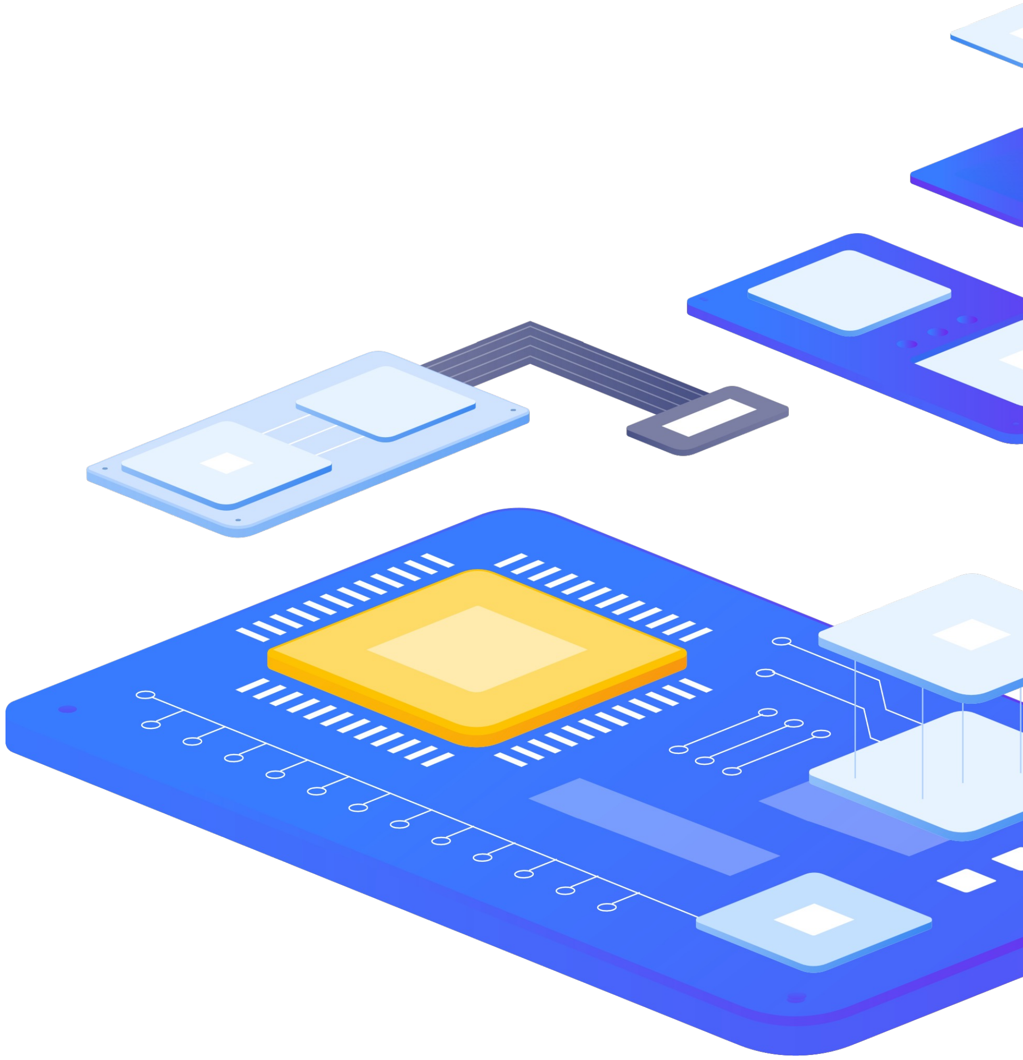
```
ldap2pg --help
usage: ldap2pg [OPTIONS] [dbname]

      --check               Check mode: exits with 1 if Postgres instance is unsynchronized.
      --color               Force color output. (default true)
  -c, --config string       Path to YAML configuration file. Use - for stdin.
  -C, --directory string    Path to directory containing configuration files.
  -?, --help                Show this help message and exit.
  -q, --quiet count         Decrease log verbosity.
  -R, --real                Real mode. Apply changes to Postgres instance.
  -P, --skip-privileges     Turn off privilege synchronisation.
  -v, --verbose count       Increase log verbosity.
  -V, --version             Show version and exit.


Optional argument dbname is alternatively the database name or a conninfo string or an URI.
See man psql(1) for more information.

By default, ldap2pg runs in dry mode.
ldap2pg requires a configuration file to describe LDAP searches and mappings.
```



initMAX

ldap2pg – configuration

- Configuration of ldap2pg is done via the ldap2pg.yml file
- Configuration is done in YAML format – watch out for syntax
- It can contain everything needed to run ldap2pg
- The configuration file is searched for in these standard locations:
  - ldap2pg.yml in current working directory
  - ~/.config/ldap2pg.yml
  - /etc/ldap2pg.yml
- If the LDAP2PG_CONFIG variable or the **--config** <path to configuration> parameter is set, ldap2pg will skip searching the default file locations
- It is also possible to specify ldap2pg - (with a dash) to read the configuration from standard input



ldap2pg – configuration file sections

- The postgres section defines custom SQL queries for PostgreSQL inspection.
- postgres:
  - databases_query
  - fallback_owner
  - managed_roles_query
  - roles_blacklist_query
  - schemas_query

ldap2pg – configuration file sections

- The privileges top level section is a mapping defining privilege profiles, referenced later in Synchronization maps.
- Using predefined privilege profiles (starts and ends with __)
- privileges:
    - default
        - Can be undefined or either global or schema
    - type
        - SELECT, REFERENCES, USAGE, etc.
    - on
        - Target ACL of privilege type. e.g. TABLES, SEQUENCES, SCHEMAS

ldap2pg – configuration file sections

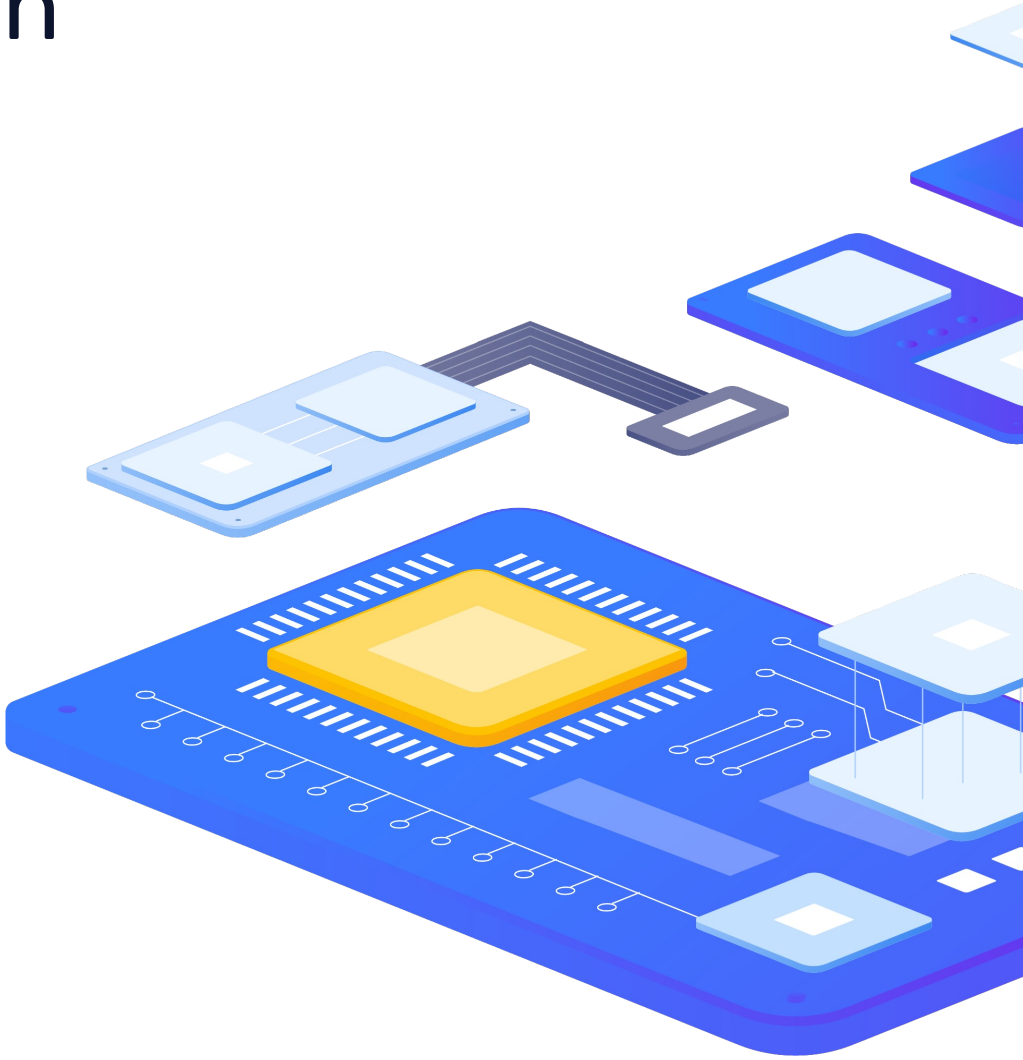
- The top-level rules section is a YAML list. This is the only mandatory parameter in `ldap2pg.yaml`.
- Each item of rules is called a mapping. A mapping is a YAML dict with any of role or grant subsection.
- rules:
    - description
    - ldapsearch
    - joins
    - role
        - comment
        - name
        - options

ldap2pg – configuration file sections

- rules: …
    - config
    - parent
    - before_create
    - after_create
  - grant
    - database
    - privilege
    - role
    - schema
    - owner

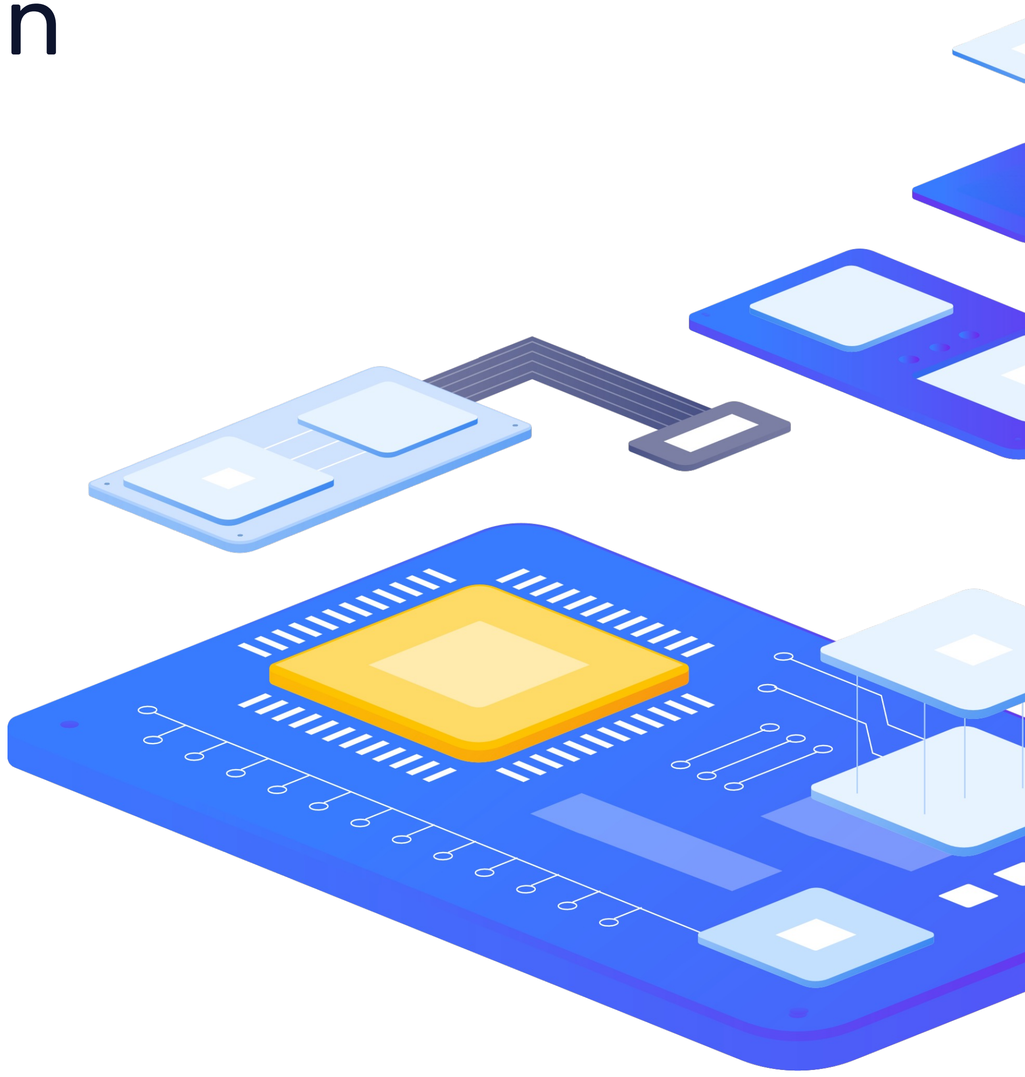
ldap2pg – example configuration

```
version: 6

postgres:
  roles_blacklist_query: [postgres, pg_*]
  # databases_query: "SELECT datname FROM pg_catalog.pg_databases;"
  databases_query: [postgres, a, b, gitlab]

privileges:
  ro:
    - __connect__
    - __select_on_tables__
    - __select_on_sequences__
    - __usage_on_schemas__
    - __usage_on_types__
  rw:
    - __temporary__
    - __all_on_tables__
    - __all_on_sequences__
  ddl:
    - __create_on_schemas__
...
```



ldap2pg – example configuration

```
...
rules:
  - description: "Setup static roles and grants."
    roles:
      - name: readers
        options: NOLOGIN
        comment: Managed by ldap2pg
      - name: writers
        parent: readers
        options: NOLOGIN INHERIT
        comment: Managed by ldap2pg
      - name: owners
        parent: writers
        options: NOLOGIN INHERIT
        comment: Managed by ldap2pg
    grant:
      - privilege: ro
        role: readers
      - privilege: rw
        role: writers
      - privilege: ddl
        role: owners
  - description: "Search LDAP to create roles from all groups found."
    ldapsearch:
      base: OU=p2d2,DC=initmax,DC=local
      filter: "(&(ObjectClass=Group)(cn=POSTGRES_gitlab_*))"
    role:
      name: "{member.sAMAccountName}"
      options: LOGIN INHERIT
      parent: "{description.lower()}"
      comment: "Generated from LDAP entry {member}"
      config:
        temp_file_limit: 100000
  - description: "Search LDAP to create DBA's roles."
    ldapsearch:
      base: CN=DBAs,OU=p2d2,DC=initmax,DC=local
    role:
      name: "{member.sAMAccountName}"
      options:
        SUPERUSER: yes
        LOGIN: yes
        CONNECTION LIMIT: 2
```

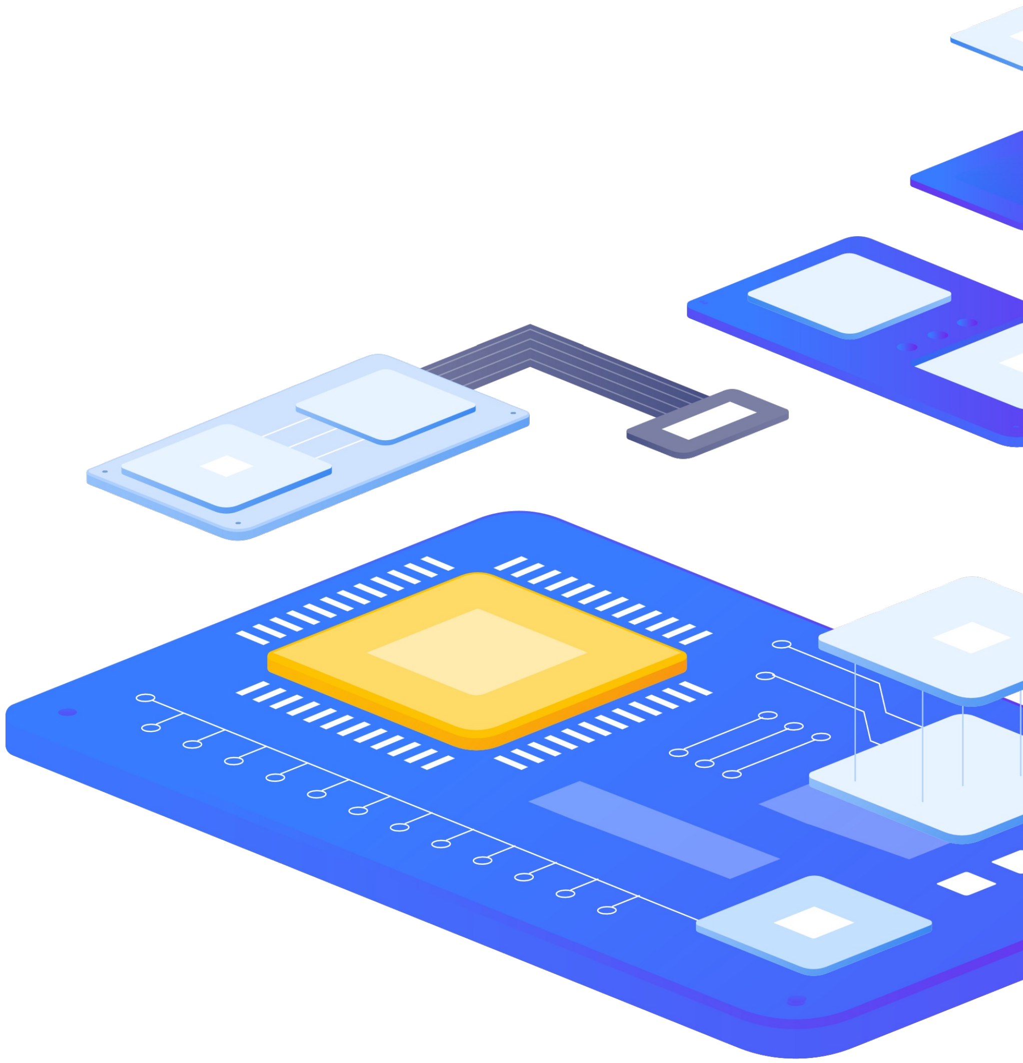


initMAX

ldap2pg – usage

```
$ /usr/bin/ldap2pg -c /tmp/ldap2pg_gitlab.yml
05:59:19 INFO   Starting ldap2pg                                  version=6.1 runtime=go1.22.1
commit=ac0bc021 pid=5994
05:59:19 INFO   Using YAML configuration file.                    path=/tmp/ldap2pg_gitlab.yml
05:59:19 WARN   Dry run. Postgres instance will be untouched.
05:59:19 INFO   Running as superuser.                             user=postgres super=true
server="PostgreSQL 16.3" cluster="" database=postgres
05:59:19 INFO   Connected to LDAP directory.                      uri=ldap://ad.initmax.local
05:59:19 INFO   Setup static roles and grants.
05:59:19 INFO   Search LDAP to create roles from all groups found.
05:59:19 INFO   Search LDAP to create DBA's roles.
05:59:19 INFO   All roles synchronized.
05:59:19 INFO   All privileges configured.                        database=postgres
…
05:59:19 INFO   Comparison complete.                         searches=6 roles=7 queries=48 grants=28
05:59:19 INFO   Use --real option to apply changes.
```

```
# /usr/bin/ldap2pg -c /tmp/ldap2pg_gitlab.yml --real
05:54:43 INFO   Starting ldap2pg                                  version=6.1 runtime=go1.22.1
commit=ac0bc021 pid=5958
05:54:43 INFO   Using YAML configuration file.                    path=/tmp/ldap2pg_gitlab.yml
05:54:43 INFO   Real mode. Postgres instance will be modified.
05:54:43 INFO   Running as superuser.                             user=postgres super=true
server="PostgreSQL 16.3" cluster="" database=postgres
05:54:43 INFO   Connected to LDAP directory.                      uri=ldap://ad.initmax.local
05:54:43 INFO   Setup static roles and grants.
05:54:43 INFO   Search LDAP to create roles from all groups found.
05:54:43 INFO   Search LDAP to create DBA's roles.
05:54:43 INFO   All roles synchronized.
05:54:43 INFO   All privileges configured.                        database=postgres
…
05:54:44 INFO   Comparison complete.                         searches=6 roles=7 queries=48 grants=28
```



initMAX

Demo

initMAX

Contact us:

Phone:	+420 800 244 442
Web:	https://www.initmax.cz
Email:	tomas.hermanek@initmax.cz
LinkedIn:	https://www.linkedin.com/company/initmax
Twitter:	https://twitter.com/initmax
Tomáš Heřmánek:	+420 732 447 184