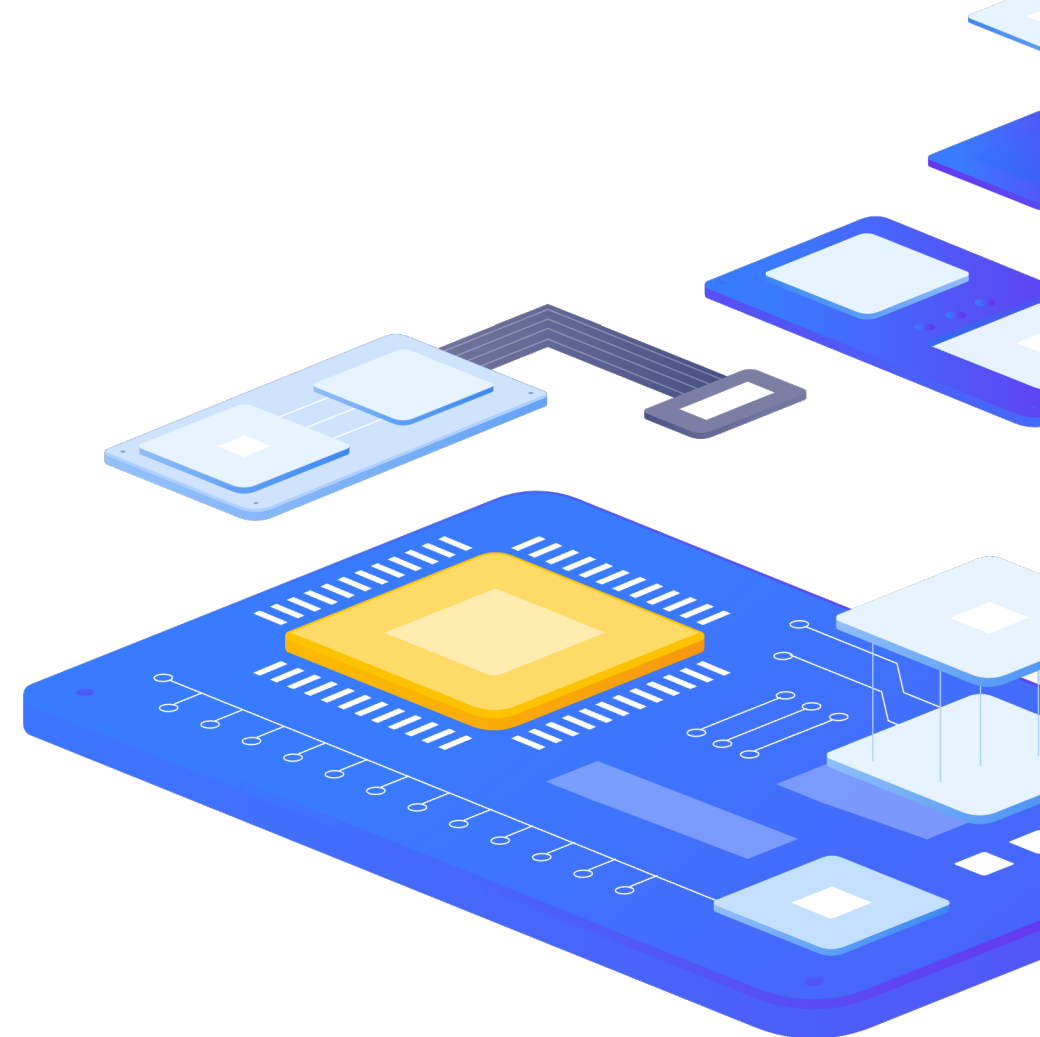




PostgreSQL access control using an external authentication provider

Introduction

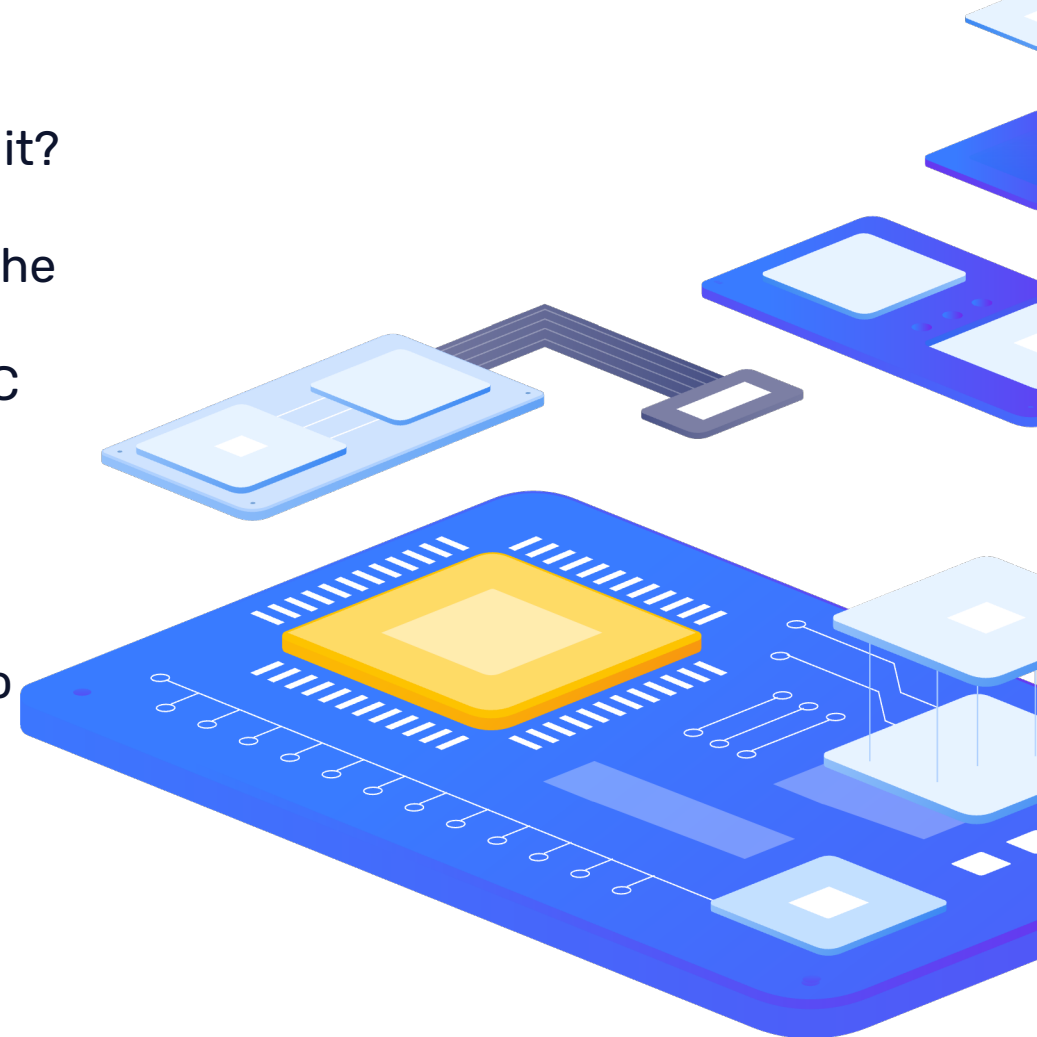
- › PostgreSQL supports 11 authentication methods, the basic ones include:
 - › **Trust authentication**, which simply trusts that users are who they say they are.
 - › **Password Authentication**, which requires users to authenticate with a password.
 - › **LDAP Authentication**, which relies on an 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. .



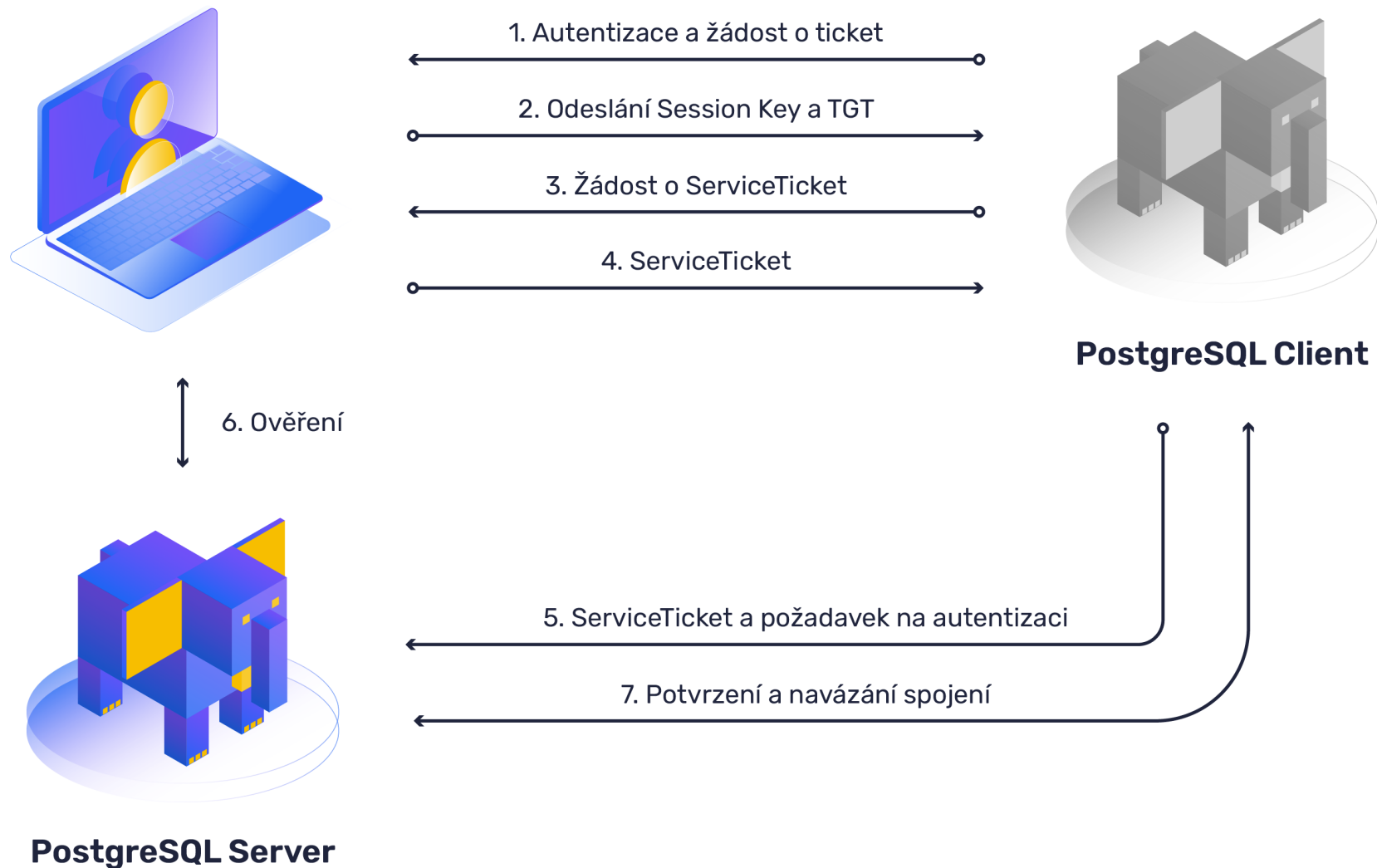
PostgreSQL access control using an external authentication provider

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**



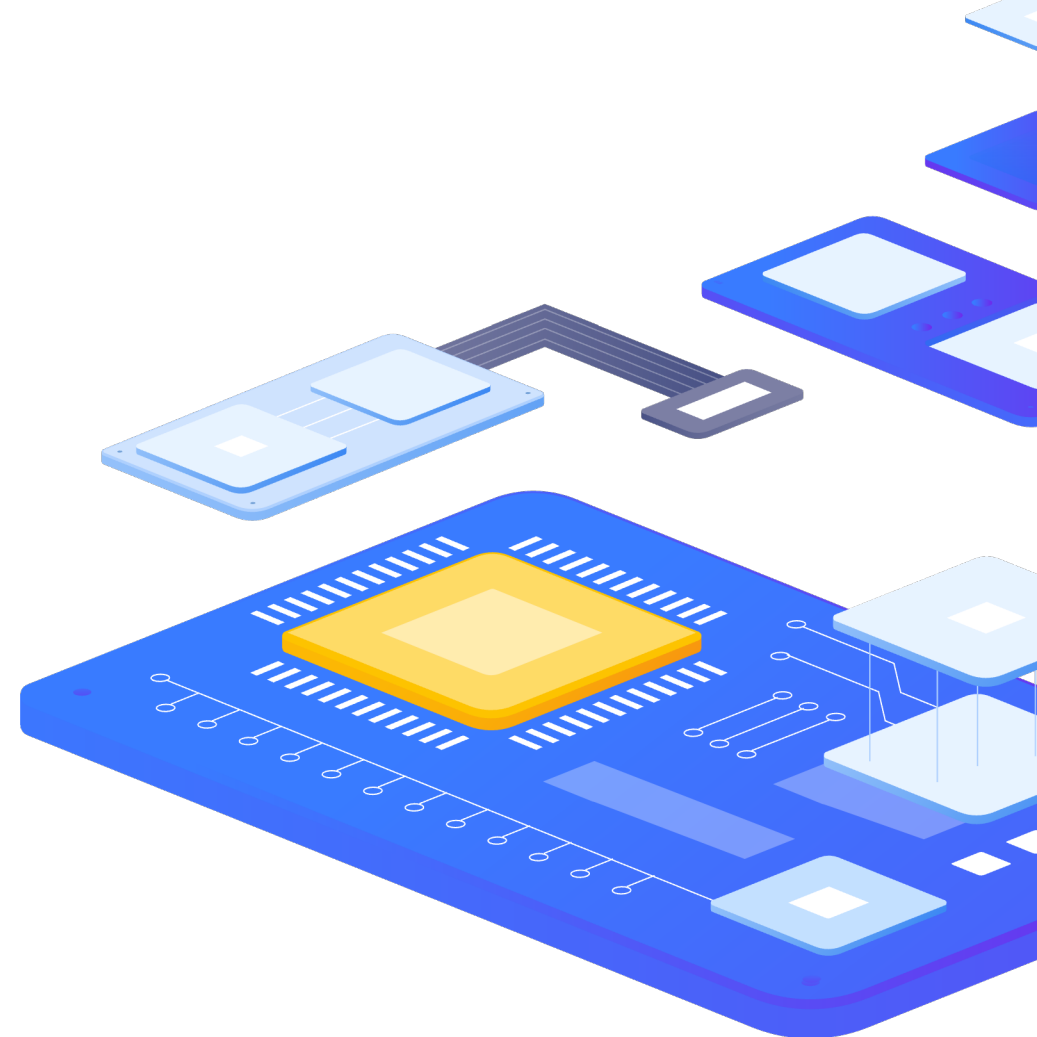
PostgreSQL access control using an external authentication provider



PostgreSQL access control using an external authentication provider

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 (Active Directory or kinit)



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 = DEMO.INITMAX.CZ
dns_lookup_realm = false
# ticket_lifetime = 24h
# renew_lifetime = 7d
forwardable = true
udp_preference_limit = 1
default_ccache_name = KEYRING:persistent:%{uid}

[realms]
DEMO.INITMAX.CZ = {
    kdc = demo.initmax.cz
    admin_server = demo.initmax.cz
}

[domain_realm]
.demo.initmax.cz = DEMO.INITMAX.CZ
demo.initmax.cz = DEMO.INITMAX.CZ
```

PostgreSQL access control using an external authentication provider

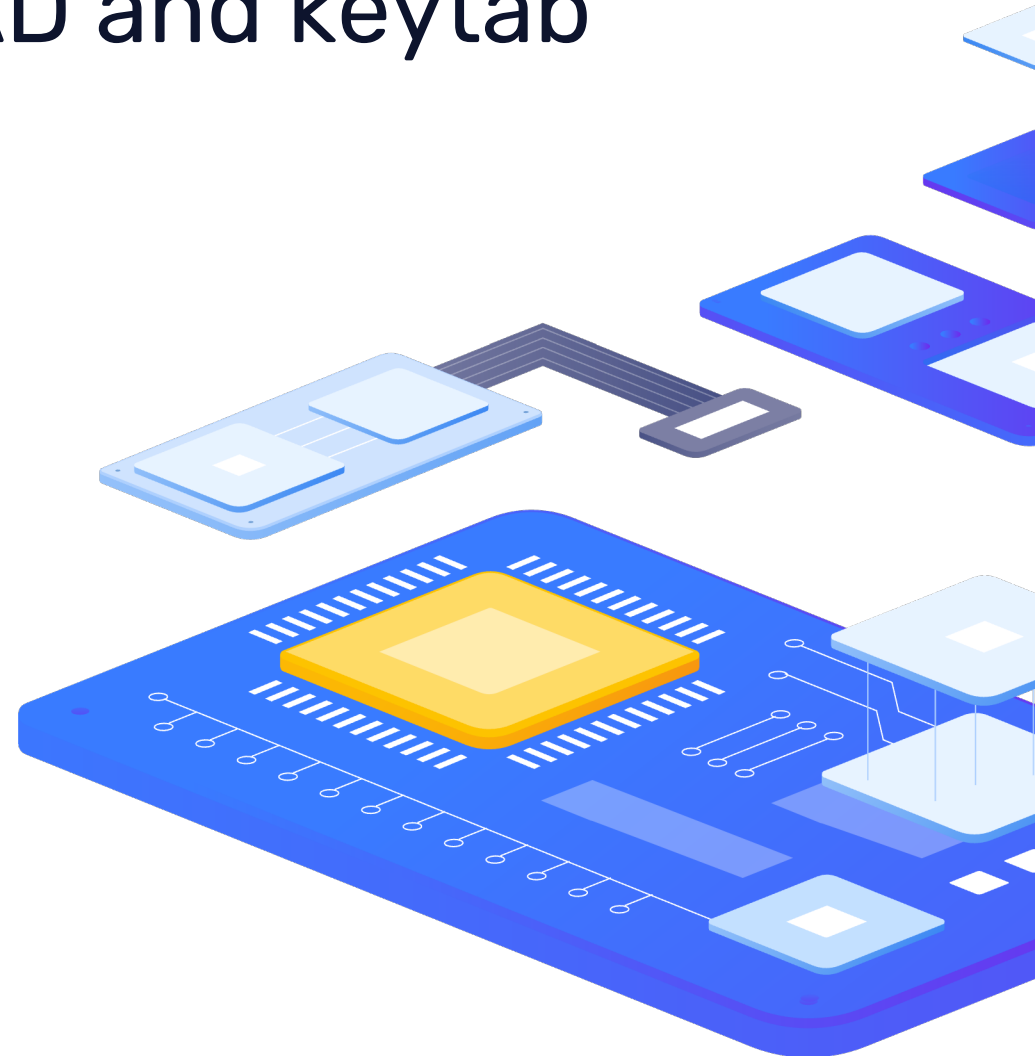
User account for DB server in AD and keytab

- › In Active Directory, create a service account for the database server - for example `pg_db_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/pgsql.demo.initmax.cz@DEMO.INITMAX.CZ -pass heslo -mapuser pg_db_srv01  
-crypto ALL -ptype KRB5_NT_Principal -out pgsql.demo.initmax.cz.keytab
```

- › We copy the keytab obtained in this way to the DB server, for example in the `/etc` folder
- › And we can verify its functionality on the PostgreSQL server

```
kinit -k -t /etc/pgsql.demo.initmax.cz.keytab POSTGRES/pgsql.demo.initmax.cz@DEMO.INITMAX.CZ -V  
Using existing cache: 0  
Using principal: POSTGRES/pgsql.demo.initmax.cz@DEMO.INITMAX.CZ  
Using keytab: /etc/pgsql.demo.initmax.cz.keytab  
Authenticated to Kerberos v5
```



PostgreSQL access control using an external authentication provider

PostgreSQL configuration

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

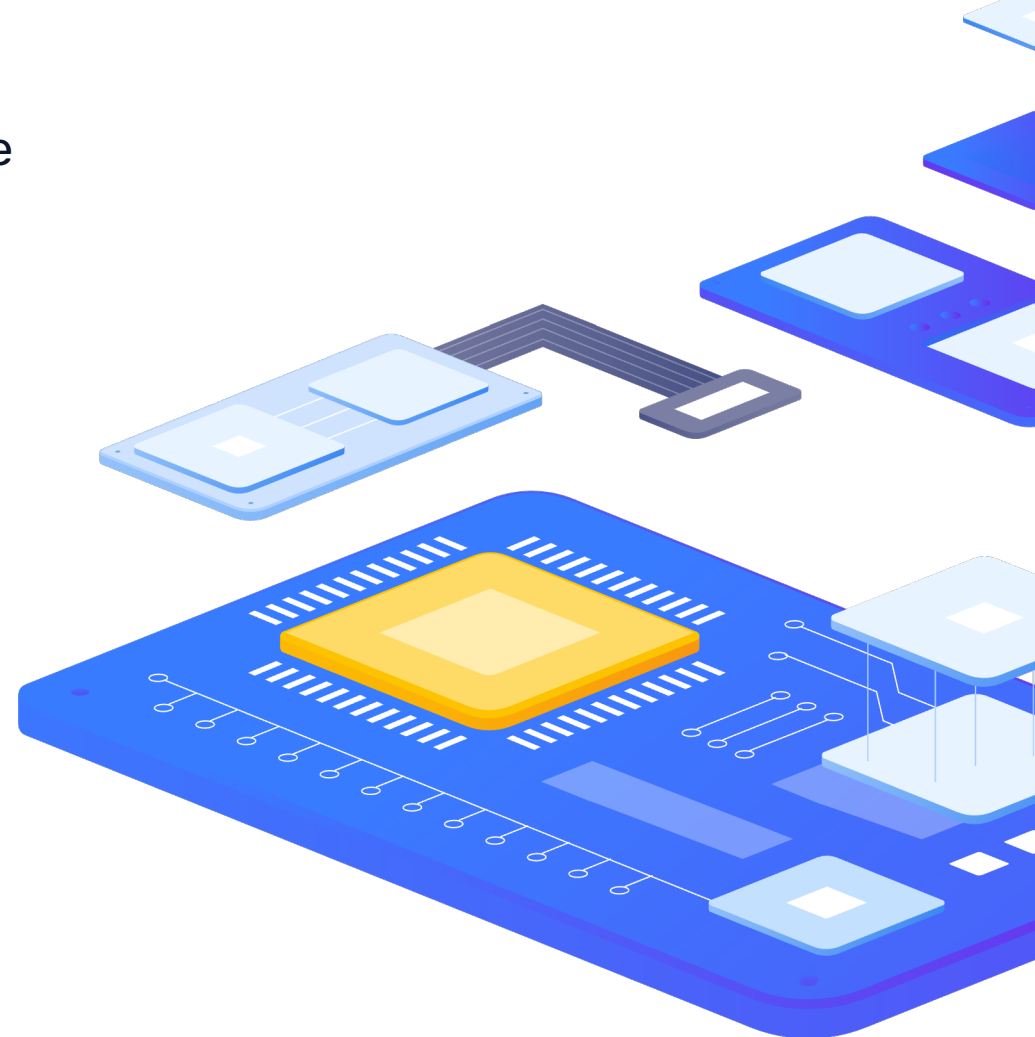
```
krb_server_keyfile=/etc/pgsql.demo.initmax.cz.keytab
```

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

```
# IPv4 local connections:  
#host all all 127.0.0.1/32 ident  
host all all 0.0.0.0/0 gss include_realm=0 krb_realm=DEMO.INITMAX.CZ
```

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

```
pgsqldemo=# create user "ad_user" superuser;
```



PostgreSQL access control using an external authentication provider

Login to PostgreSQL

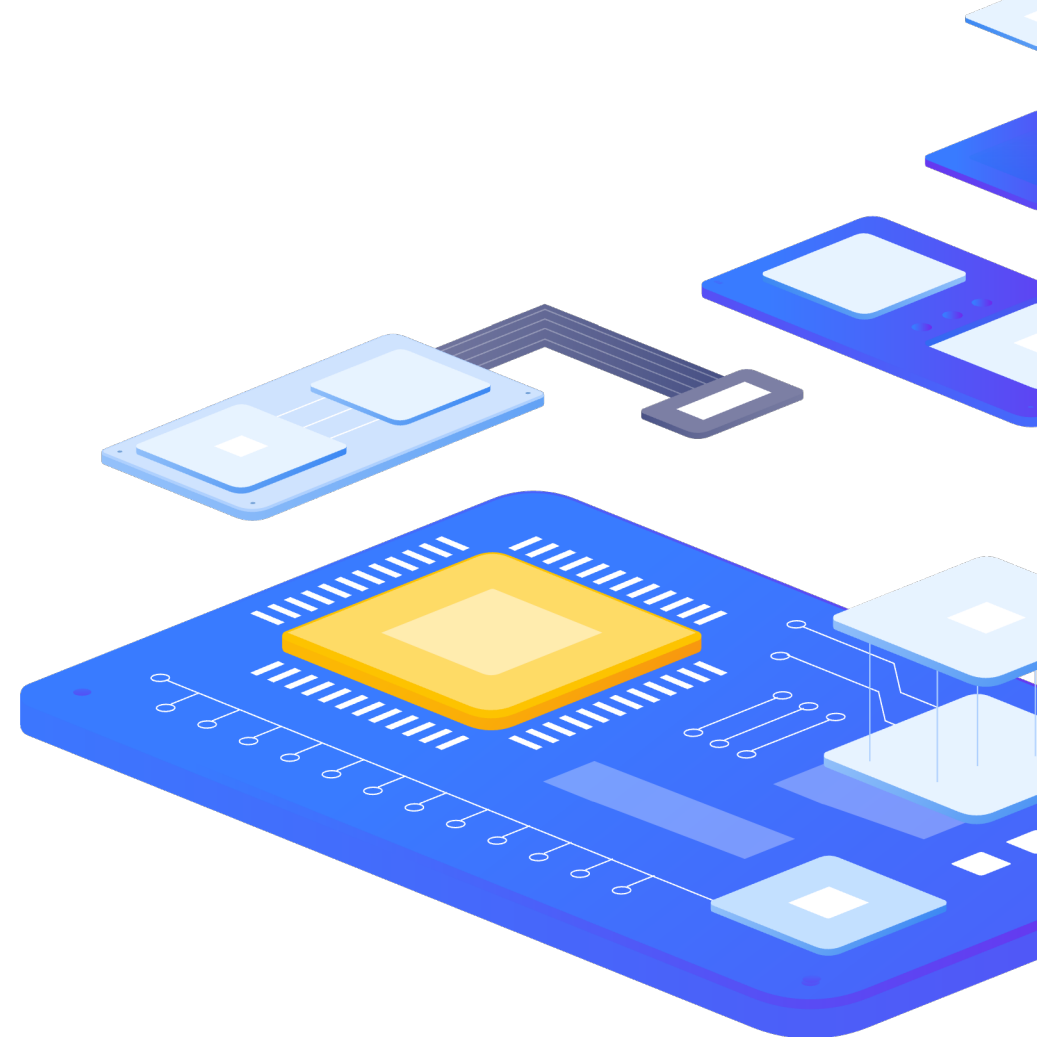
- › Getting a ticket from Active Directory

```
kinit ad_user
```

- › Login to PostgreSQL

```
psql -U "ad_user" -h csas-pgsql.win.initmax.cz postgres
```

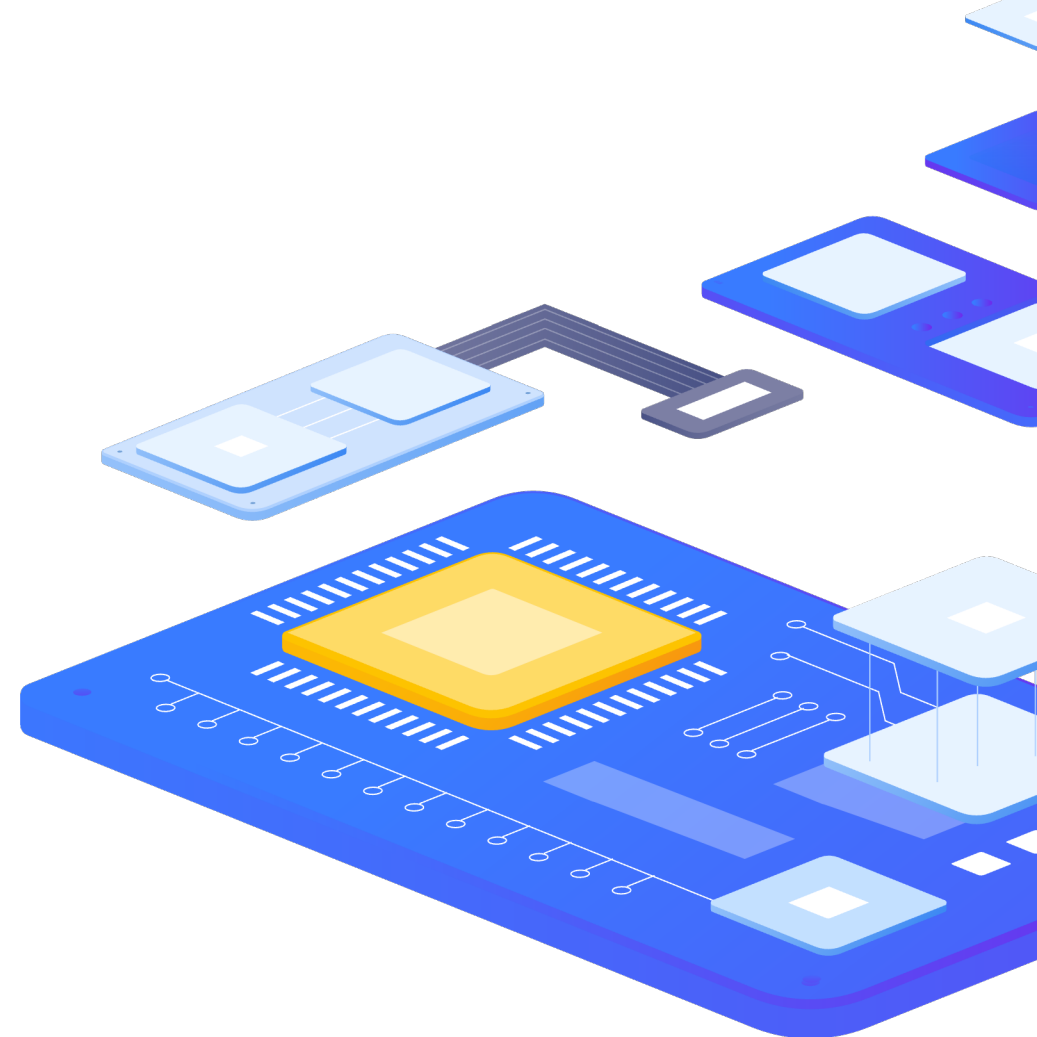
- › In larger environments, user creation can be automated
- › For example, a combination of the following can be used
 - › LDAP (Active Directory, FreeIPA, OpenLDAP,...)
 - and
 - › Idap2pg



PostgreSQL access control using an external authentication provider

Idap2pg

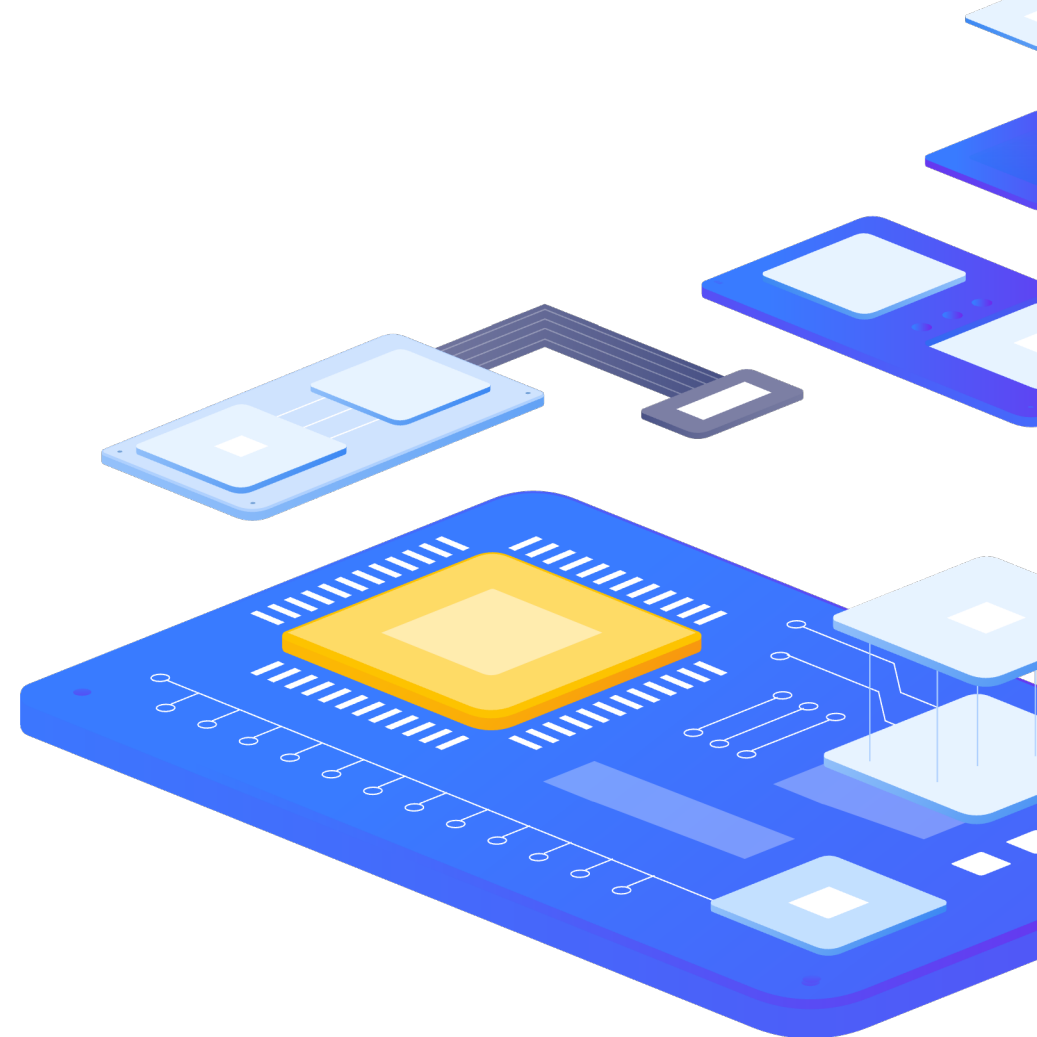
- › Idap2pg automates the creation, update and removal of PostgreSQL roles and users
- › A YAML file is used for configuration
- › Creates, changes and deletes roles in PostgreSQL according to settings in LDAP
- › Can set or remove permissions statically or according to LDAP settings
- › Can manage role membership
- › It can perform validation of settings before its deployment using parameters `--dry` and `--check`



PostgreSQL access control using an external authentication provider

Idap2pg - installation

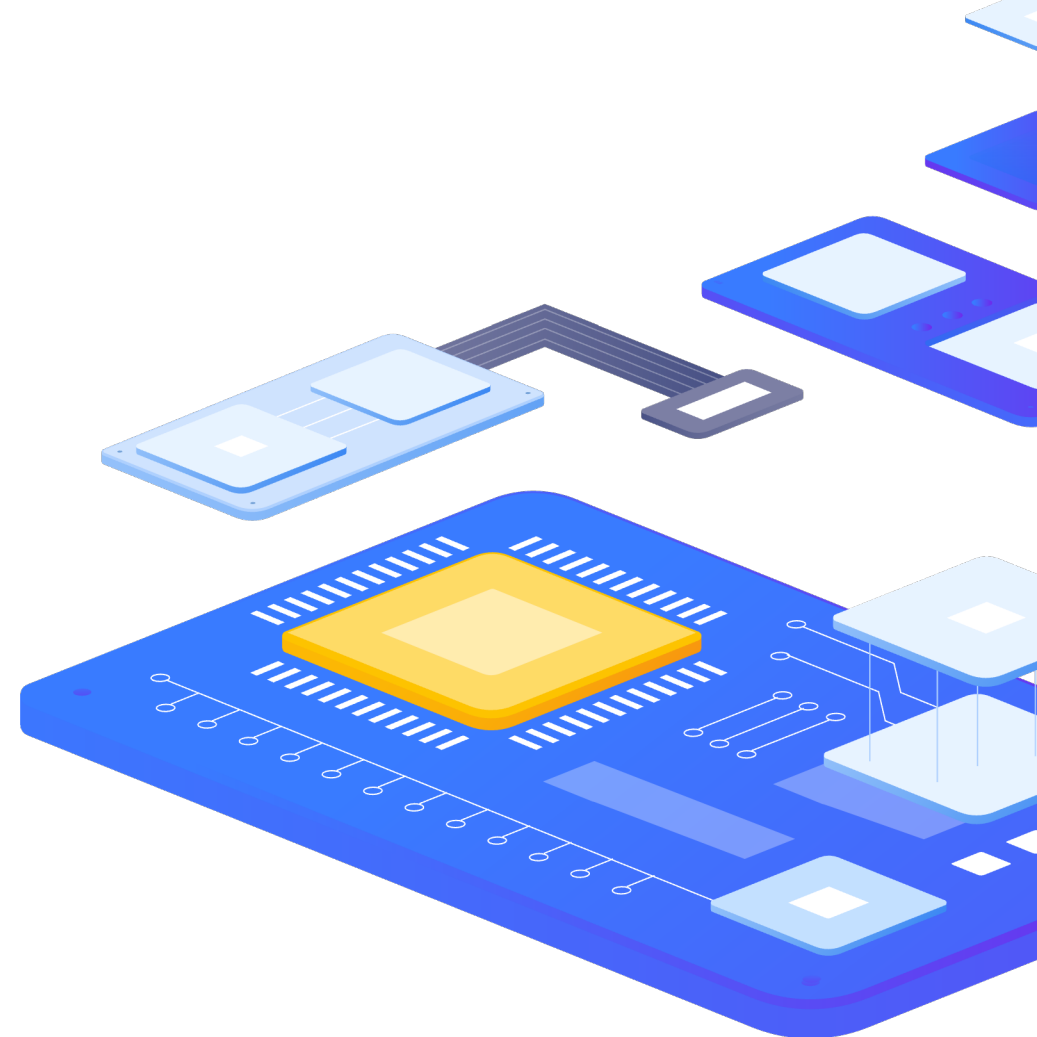
- › Idap2pg is available as a Python package
- › Idap2pg requires:
 - › Python 2.6+ or Python 3.4+
 - › Pyyaml
 - › python-ldap
 - › python-psycopg2
- › The authors recommend using distribution packages both for installing dependencies and for Idap2pg itself, if available.



PostgreSQL access control using an external authentication provider

Idap2pg - installation

- › Installation on RHEL 6/7/8/9 compatible OS
 - › Two repositories are available
 - › PGDG YUM repository
 - › the official PostgreSQL repository
 - › You may already have it available on a server with PostgreSQL
 - › Dalibo Labs YUM repository
 - › upstream
 - › Packages are more current
- › Debian 8/9/10/11
 - › It is necessary to install using pip



PostgreSQL access control using an external authentication provider

Idap2pg - installation

- ▶ 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

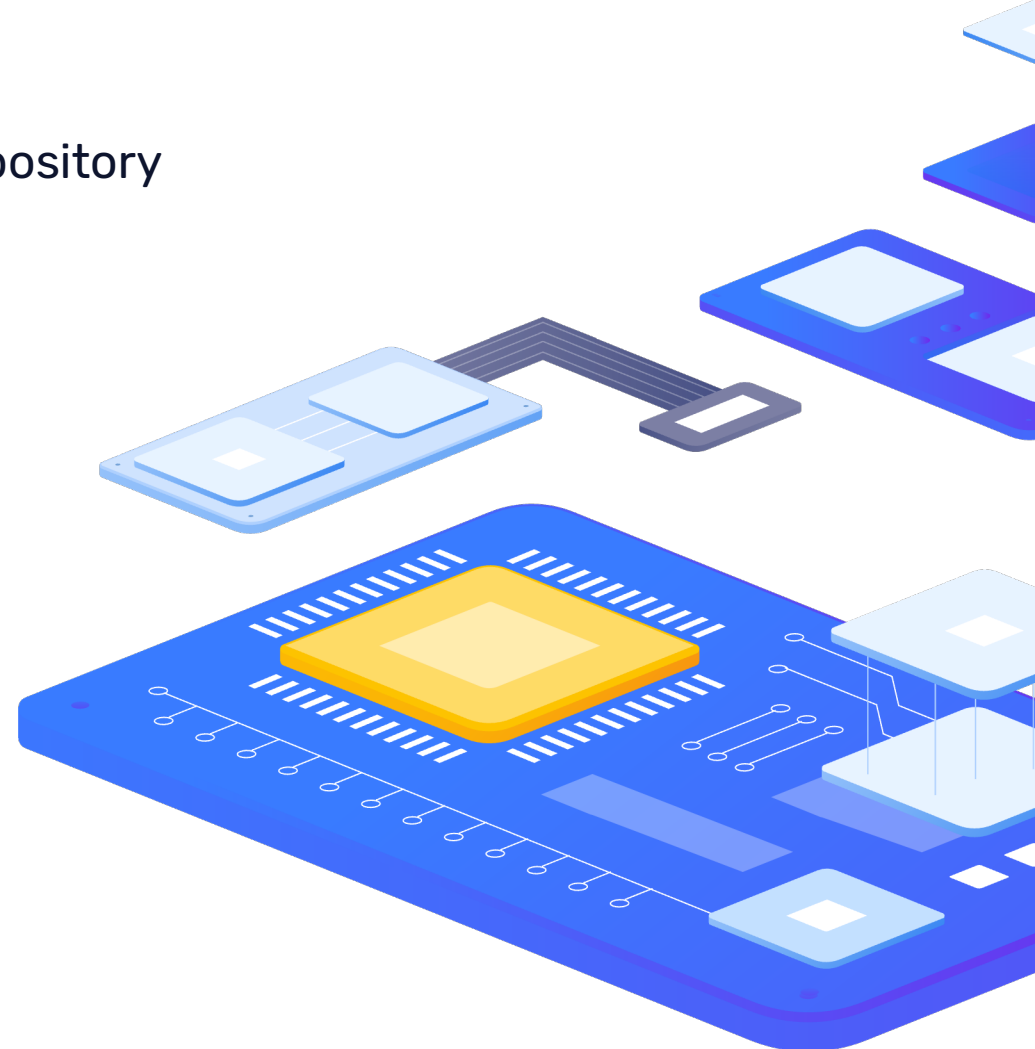
```
vi /etc/yum.repos.d/dalibolabs.repo

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

# uložit a obnovit dnf cache
dnf makecache fast
```

- ▶ Install Idap2pg itself

```
dnf install ldap2pg
```



PostgreSQL access control using an external authentication provider

ldap2pg – verifying the installation

```
ldap2pg -V
ldap2pg 5.8
psycopy2 2.8.6 (dt dec pq3 ext lo64) libpq 12.4
python-ldap 3.3.1
Python 3.6.8 (default, Nov  9 2021, 14:44:26)
[GCC 8.5.0 20210514 (Red Hat 8.5.0-3)]ld
```

ldap2pg --help

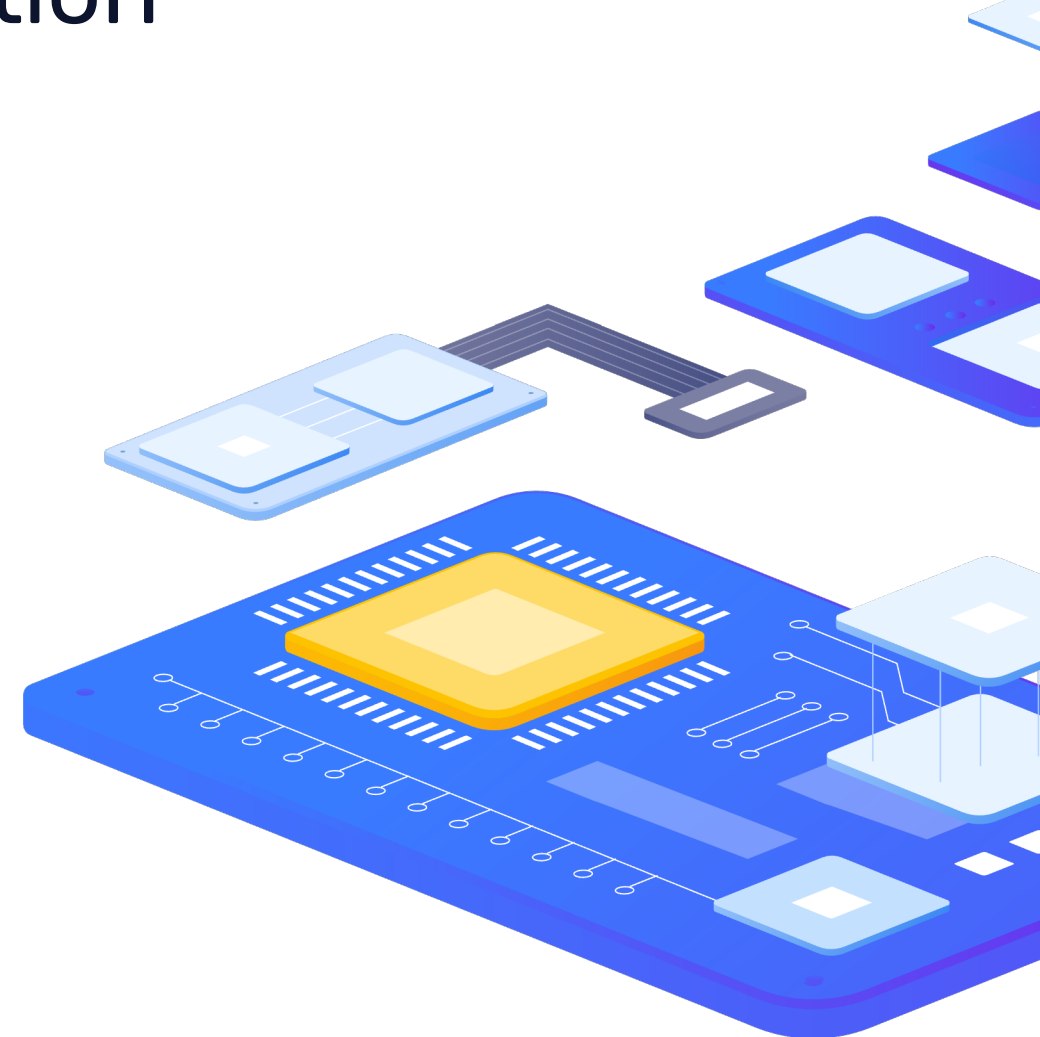
```
usage: ldap2pg [-c PATH] [-C] [-n] [-N] [-q] [-v] [--color] [--no-color] [-?]
              [-V]
```

PostgreSQL roles and privileges management.

optional arguments:

-c PATH, --config PATH	path to YAML configuration file (env: LDAP2PG_CONFIG). Use - for stdin.
-C, --check	check mode: exits with 1 on changes in cluster
-n, --dry	don't touch Postgres, just print what to do (env: DRY=1)
-N, --real	real mode, apply changes to Postgres (env: DRY='')
-q, --quiet	decrease log verbosity (env: VERBOSITY)
-v, --verbose	increase log verbosity (env: VERBOSITY)
--color	force color output (env: COLOR=1)
--no-color	force plain text output (env: COLOR='')
-, --help	show this help message and exit
-V, --version	show version and exit

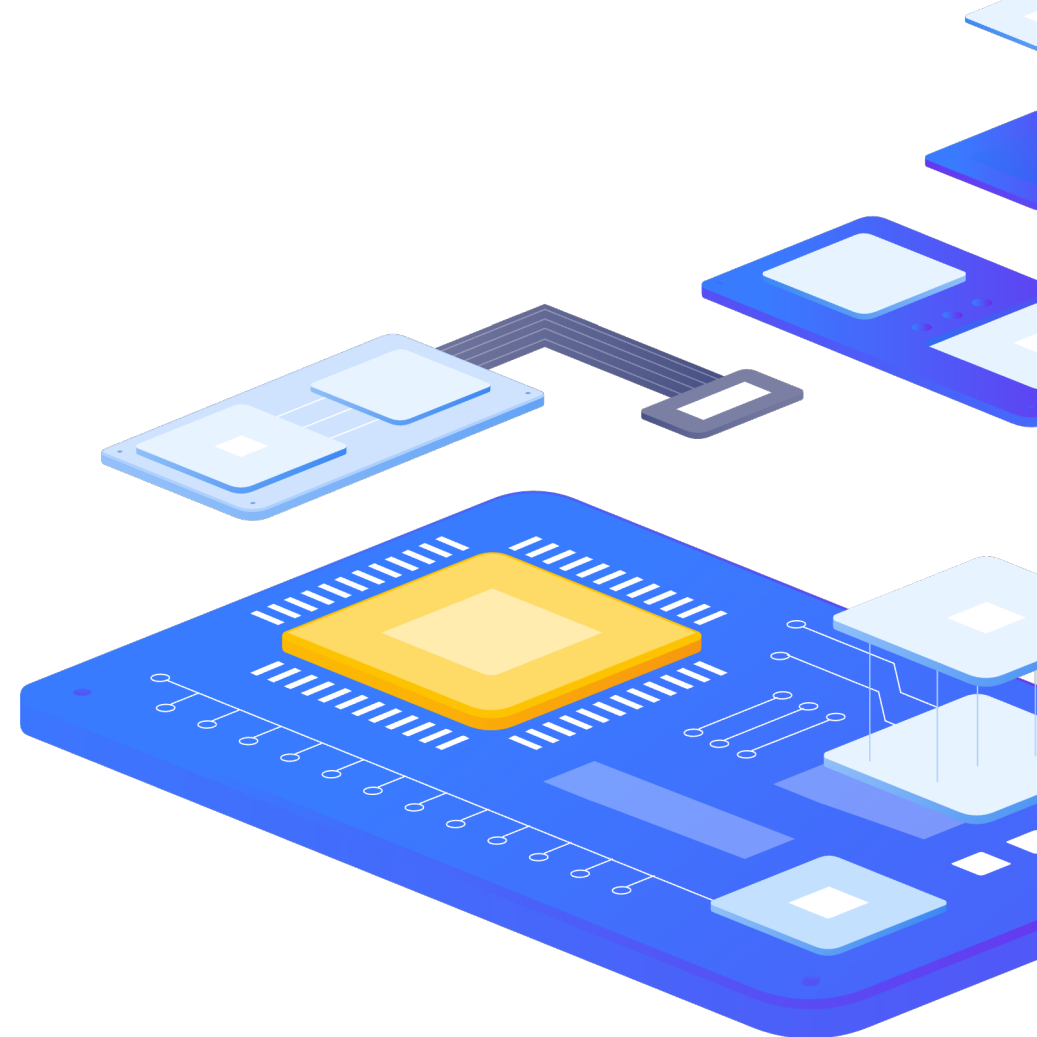
ldap2pg requires a configuration file to describe LDAP searches and role mappings. See <https://ldap2pg.readthedocs.io/en/latest/> for further details. By default, ldap2pg runs in dry mode.



PostgreSQL access control using an external authentication provider

Idap2pg - configuration

- › Configuration of Idap2pg is saved in the Idap2pg.yml file
- › Configuration is done in YAML format – watch out for syntax
- › It can contain everything needed to run Idap2pg
- › The configuration file is searched for in these standard locations:
 - › Idap2pg.yml in current working directory
 - › ~/.config/Idap2pg.yml
 - › /etc/Idap2pg.yml
- › If the LDAP2PG_CONFIG variable or the --config <path to configuration> parameter is set, Idap2pg will skip searching the default file locations
- › It is also possible to specify Idap2pg - (with a dash) to read the configuration from standard input



PostgreSQL access control using an external authentication provider

ldap2pg – example configuration

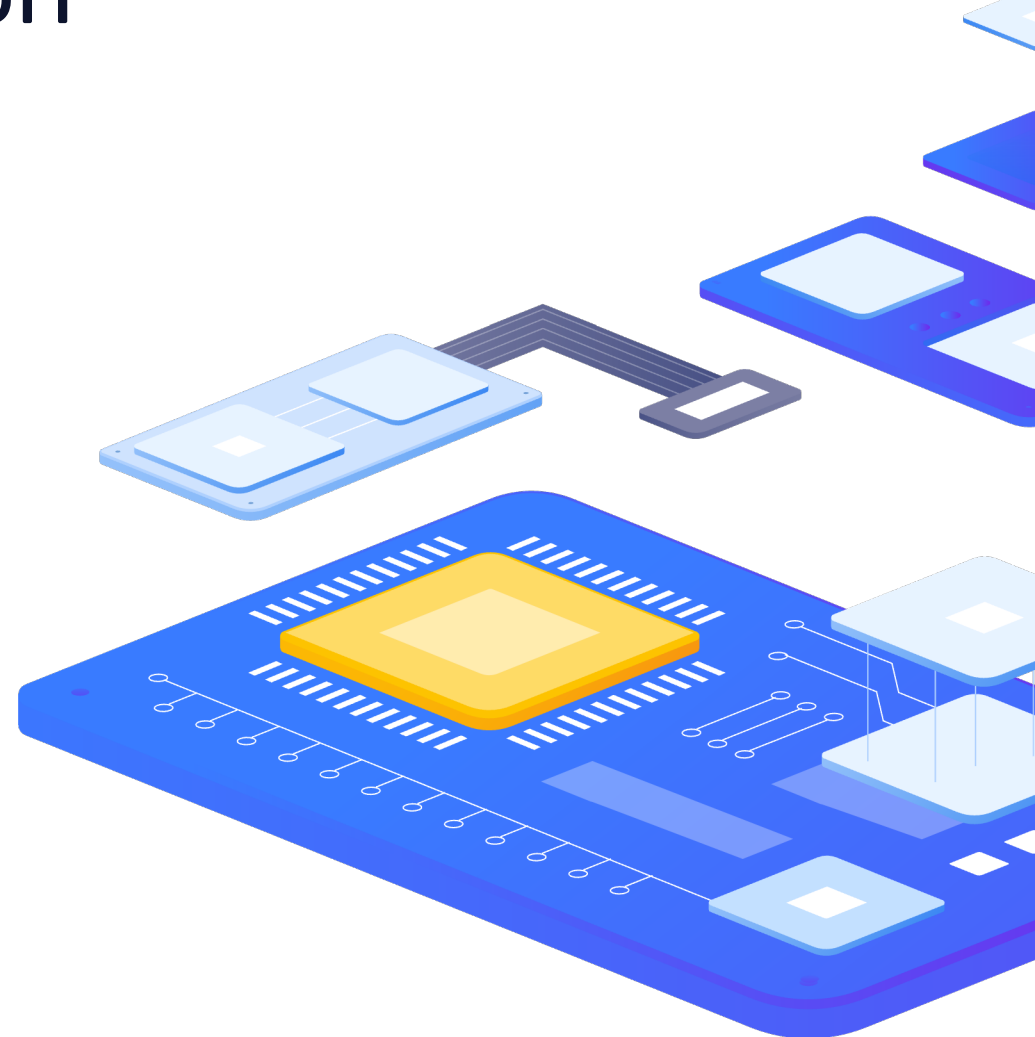
```
postgres:
  dsn: postgres://alfa@csas-pgsql.win.initmax.cz:5432/postgres
  roles_blacklist_query:
    - postgres
    - "pg_*"
    - "rds_*"

ldap:
  uri: ldap://dc1.win.initmax.cz
  binddn: CN=Test User Alfa,OU=Users,OU=testAccounts,DC=win,DC=initmax,DC=cz
  password: "heslo"

sync_map:
- role:
  name: alfa
  options: LOGIN SUPERUSER
  names:
    - ad_roles
  comment: "LDAP role managed by ldap2pg."

- ldapsearch:
  base: CN=pg_DBA_users,OU=Groups,OU=testAccounts,DC=win,DC=initmax,DC=cz
  role:
  name: 'dba_{member.samaccountname}'
  options: LOGIN SUPERUSER
  parent: ad_roles
  comment: "Synced from AD: {dn}"

- ldapsearch:
  base: CN=pg_RO_users,OU=Groups,OU=testAccounts,DC=win,DC=initmax,DC=cz
  role:
  name: '{member.samaccountname}'
  options: LOGIN
  parent: ad_roles
  comment: "Synced from AD: {dn}"
```

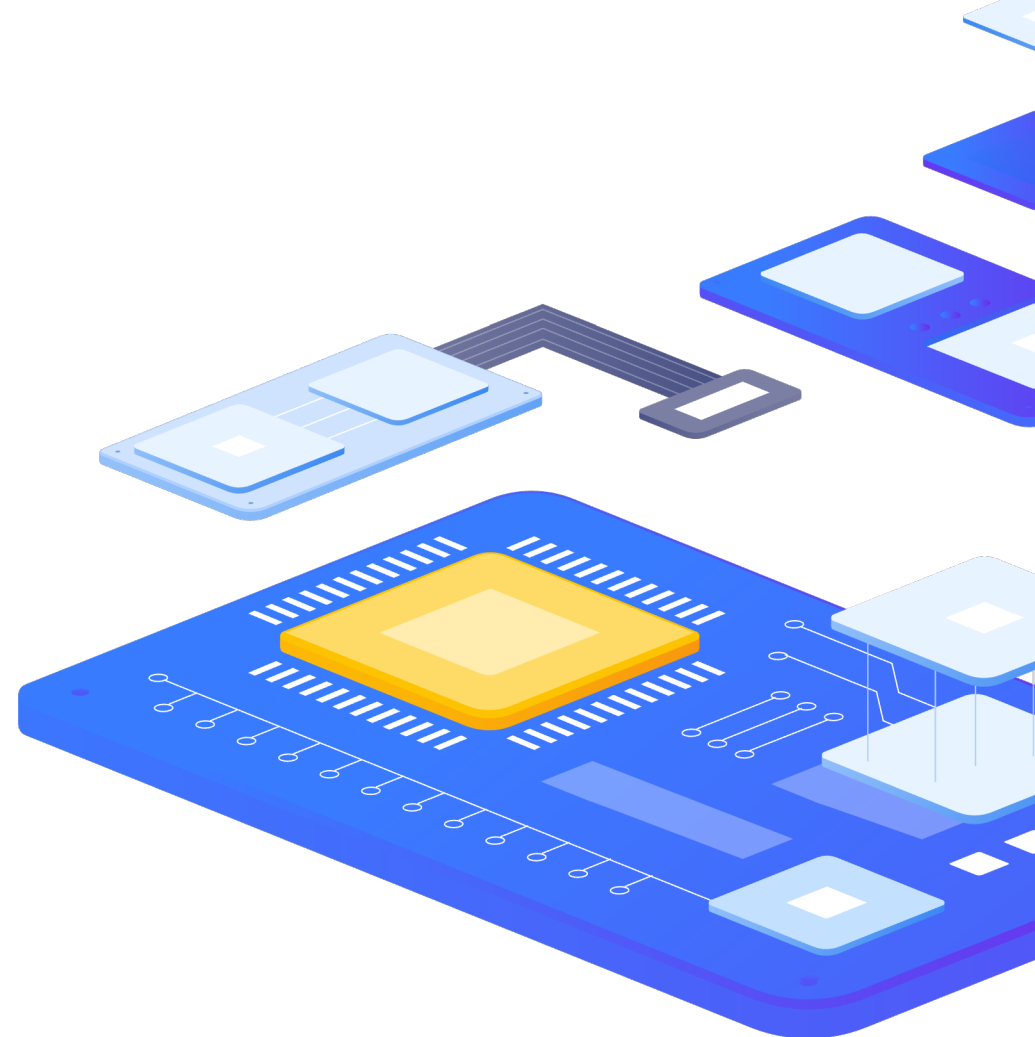


PostgreSQL access control using an external authentication provider

ldap2pg - usage

```
# ldap2pg --dry
Starting ldap2pg 5.8.
Using /root/ldap2pg.yml.
Connecting to LDAP server ldap://dc1.win.initmax.cz.
Trying simple bind.
Running in dry mode. Postgres will be untouched.
Inspecting roles in Postgres cluster...
Querying LDAP CN=pg_DBA_users,OU=Group... (objectClass...
Missing 'member' from CN=pg_DBA_users,OU=Groups,OU=testAccounts,DC=win,DC=initmax,DC=cz.
Considering it as an empty list.
Querying LDAP CN=pg_RO_users,OU=Groups... (objectClass...
Missing 'member' from CN=pg_RO_users,OU=Groups,OU=testAccounts,DC=win,DC=initmax,DC=cz. Considering
it as an empty list.
Nothing to do.
Comparison complete.
```

```
$ ldap2pg --real
Starting ldap2pg 5.8.
Using /root/ldap2pg.yml.
Connecting to LDAP server ldap://dc1.win.initmax.cz.
Trying simple bind.
Running in real mode.
Inspecting roles in Postgres cluster...
Querying LDAP CN=pg_DBA_users,OU=Group... (objectClass...
Missing 'member' from CN=pg_DBA_users,OU=Groups,OU=testAccounts,DC=win,DC=initmax,DC=cz. Considering
it as an empty list.
Querying LDAP CN=pg_RO_users,OU=Groups... (objectClass...
Missing 'member' from CN=pg_RO_users,OU=Groups,OU=testAccounts,DC=win,DC=initmax,DC=cz. Considering
it as an empty list.
Nothing to do.
Synchronization complete.
```





Demonstration



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