PostgreSQLの認証方式入門

～ pg_hba.confからOAuth認証まで ～

株式会社SRA OSS
馬 雪テイ

自己紹介

名前: 馬 雪テイ

所属: 経営基盤部 システム・インフラ開発グループ

担当業務:

- PostgreSQLサポート
- PostgreSQL HAクラスタ構築・CDC支援・コンサルティング
- 社内システム開発

会社紹介

株式会社SRA OSS

所 在 地: 東京都豊島区南池袋2-32-8
設 立 日: 2022年6月17日
株　　主: 株式会社SRA
　　株式会社NTTデータ
資 本 金: 7,000万円
社　　長: 稲葉 香理

事業内容

- オープンソースソフトウェア (OSS) 関連のサポート、製品開発・販売、構築・コンサル
- OSSの教育、開発、コミュニティ運営支援
- ソフトウェアの研究開発

顧　　問: 石井 達夫

技術顧問: 増永 良文 (お茶の水女子大学名誉教授)

本日の内容

- PostgreSQLの認証の全体像
- pg_hba.confの基本
- 代表的な認証方式
- PostgreSQL 18のOAuth認証
- 認証方式の選び方

認証方式一覧

分類	認証方式	主な特徴
認証なし	trust	無条件で接続を許可
OS連携	peer	OSユーザ情報を利用
パスワード	password	パスワードを利用
パスワード	MD5	MD5ベースの認証
パスワード	SCRAM-SHA-256	より安全なパスワード認証
証明書	cert	クライアント証明書を利用
外部認証	LDAP	LDAPサーバーと連携
外部認証	OAuth	OAuth 2.0による認証

本講演のゴール

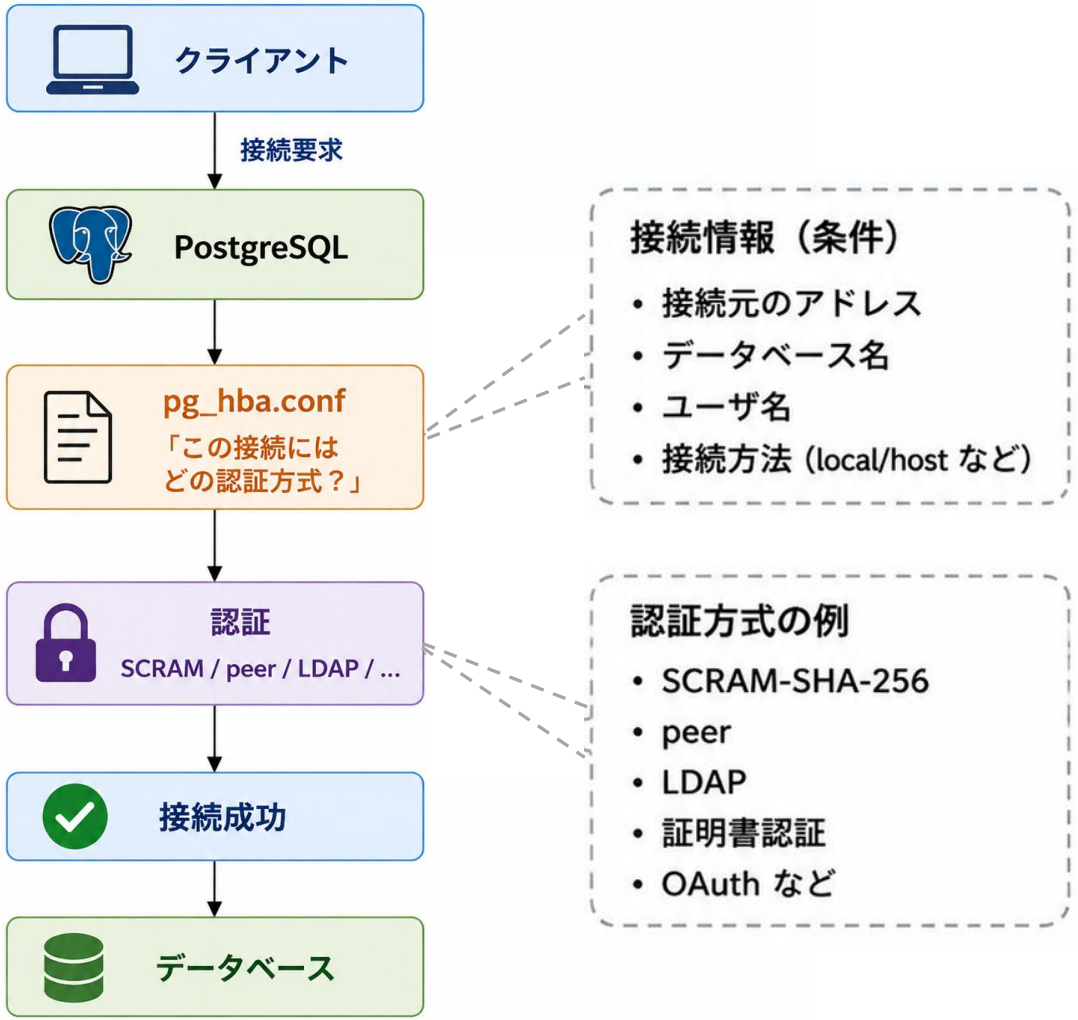
- PostgreSQLの認証の基本的な仕組みを理解する
- 代表的な認証方式の違いを理解する
- 環境に応じた認証方式を選べるようになる

最後に、

PostgreSQL 18のOAuth認証を知る

PostgreSQLの認証の全体像

- クライアントからPostgreSQLへ接続する流れ



PostgreSQLの認証の全体像

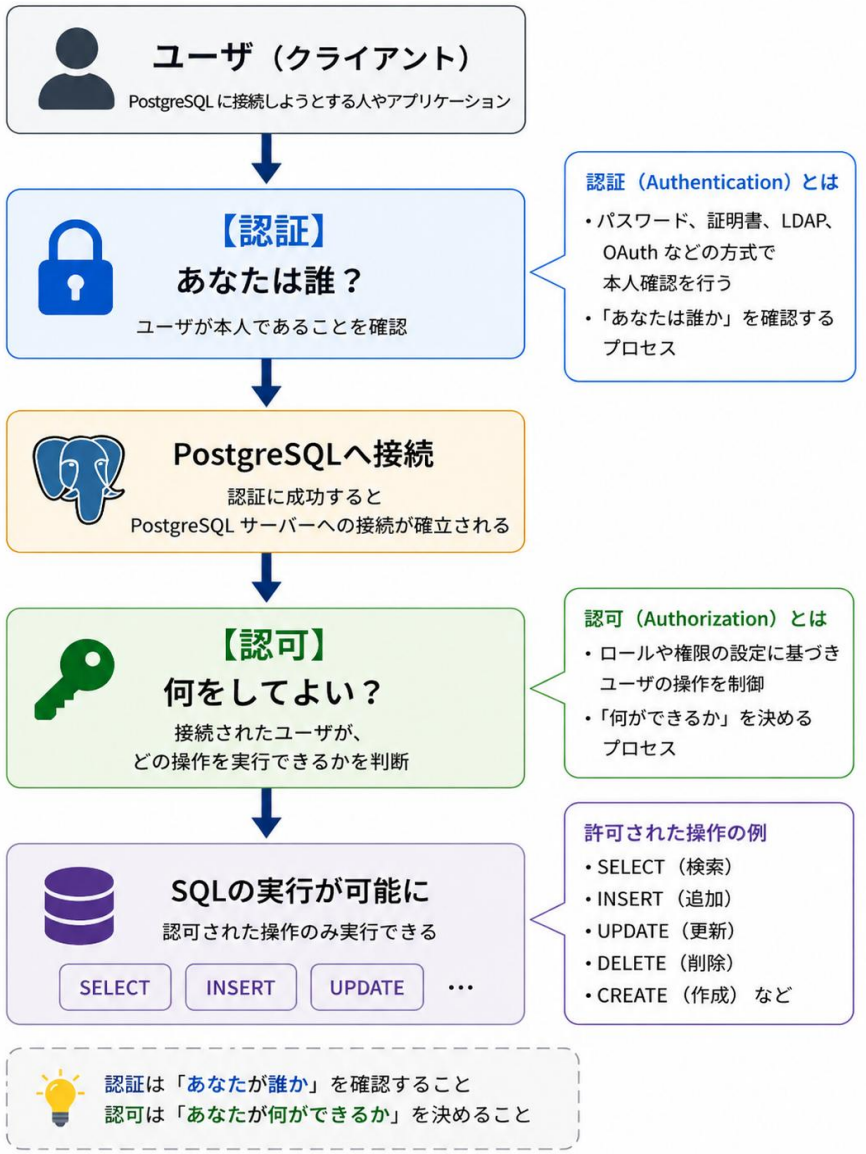
- PostgreSQLロールとOSユーザの違い

OSユーザ	PostgreSQLロール
Linux/Windowsが管理	PostgreSQLが管理
OSへのログイン	DBへの接続・権限管理

OSユーザ postgres　≠　PostgreSQLロール postgres

PostgreSQLの認証の全体像

- 認証と認可の違い



PostgreSQLの認証の全体像

- 認証方式を決定するpg_hba.confの役割

接続要求

↓

どこから？

どのDBへ？

どのユーザ？

➡ pg_hba: 接続条件に応じて使用する認証方式を決める

↓

SCRAM / peer / LDAP ...

pg_hba.confの基本

- HBAは「Host-Based Authentication」の略
- 接続元、DB、ユーザなどの条件から認証方式を決定する
- デフォルトのpg_hba.confファイルは、データディレクトリがinitdbで初期化される時にインストールされる
- 一般的な書式は、1行につき1つのレコードというレコードの集合

```
host    mydb    myuser    192.168.1.0/24    scram-sha-256
```

どこから？　→ 192.168.1.0/24

誰が？　→ myuser

どのDBへ？　→ mydb

どう認証する？　→ SCRAM-SHA-256

pg_hba.confの基本

- 各項目の意味

```
TYPE   DATABASE    USER      ADDRESS              METHOD
host   mydb        user1     192.168.1.0/24       scram-sha-256
```

項目	意味	例
TYPE	接続方法	local/host/hostssl
DATABASE	DB	mydb
USER	ロール	user1
ADDRESS	接続元	192.168.1.0/24
METHOD	認証方式	scram-sha-256

local → Unixドメインソケット
host → TCP/IP
hostssl → SSL/TLSを利用したTCP/IP

pg_hba.confの基本

- 上から順に評価される仕組み

```
# ①
host  all  all    192.168.1.0/24  trust

# ②
host  all  admin  192.168.1.0/24  scram-sha-256
```

①が一致 → trust（**危険!!**）

pg_hba.confの基本

- 設定変更時の注意点

変更の流れ

pg_hba.conf編集
↓
設定内容を確認
↓
reload
↓
接続確認

Reload方法

SELECT pg_reload_conf();

pg_ctl reload

kill -HUP

確認SQL

SELECT * FROM pg_hba_file_rules;

注意

- reloadで反映可能
- PostgreSQL再起動は基本的に不要
- 設定ミスで自分自身が接続できなくならないよう注意

pg_hba.confの基本

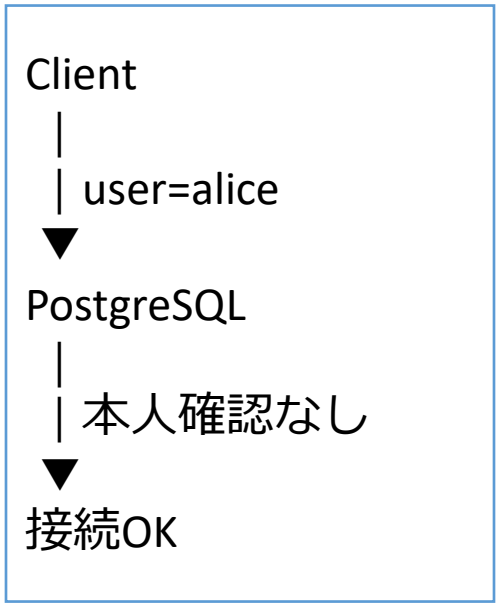
「接続できない！」ときのチェック

- 一致する行はある？
- 上の行に先に一致していない？
- IPアドレスは正しい？
- local / hostを間違えていない？
- reloadした？

代表的な認証方式

- trust

仕組み



特徴

- パスワード不要
- 本人確認を行わない
- 設定は簡単
- セキュリティ上の注意が必要

pg_hba.conf

```
host  all  all  127.0.0.1/32  trust
```

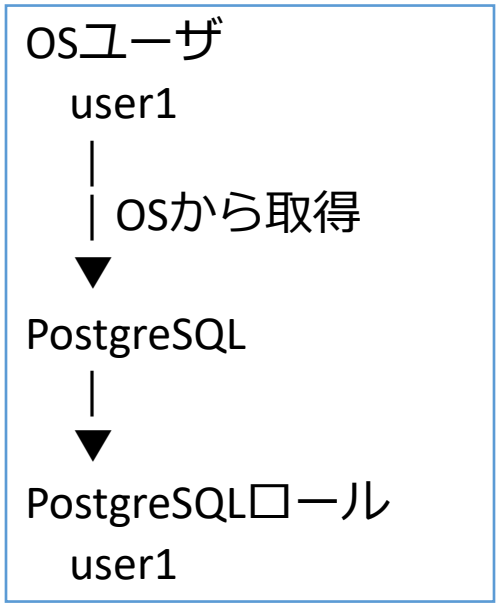
「便利」≠「安全」

サーバに接続する際に適切なオペレーティングシステムレベルの保護が掛けられている場合にのみ使用すべき

代表的な認証方式

- peer

仕組み



特徴

- OSユーザ情報を利用
- パスワード不要
- ローカル接続でのみ使用可能

pg_hba.conf

```
local  all  all  peer
```

代表的な認証方式

- password

仕組み

特徴

- パスワードを使用
- パスワード自体を暗号化せず送信
- SSL/TLSなどで通信経路を保護する必要がある

pg_hba.conf

```
hostssl  all  all  0.0.0.0/0  password
```

代表的な認証方式

- MD5

仕組み

password
↓
MD5形式で保存
↓
pg_authid
↓
MD5認証

特徴

- パスワードをそのまま送信しない
- MD5ベース
- 古いPostgreSQL環境で多く利用
- 現在はSCRAM-SHA-256への移行が必要

pg_hba.conf

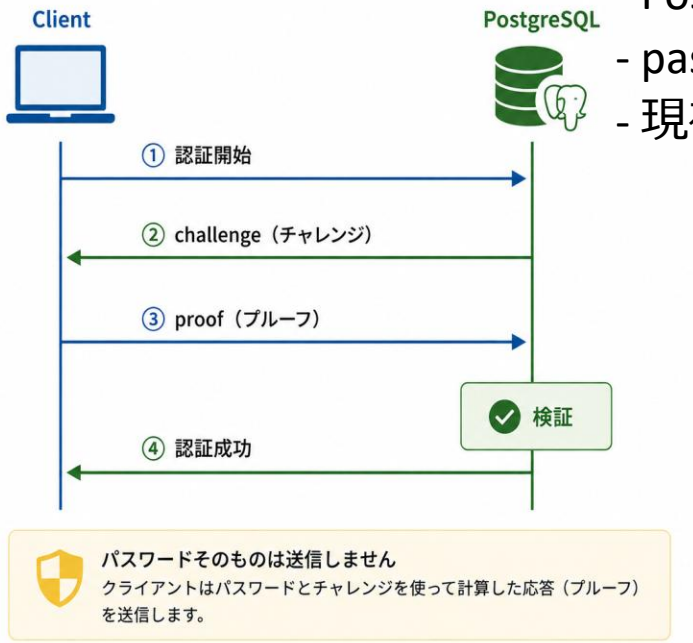
```
host  all  all  192.168.1.0/24  MD5
```

PostgreSQL 18ではMD5パスワード認証は非推奨。
MD5と書かれていても、SCRAM認証へ自動切り替える

代表的な認証方式

- SCRAM-SHA-256

認証イメージ



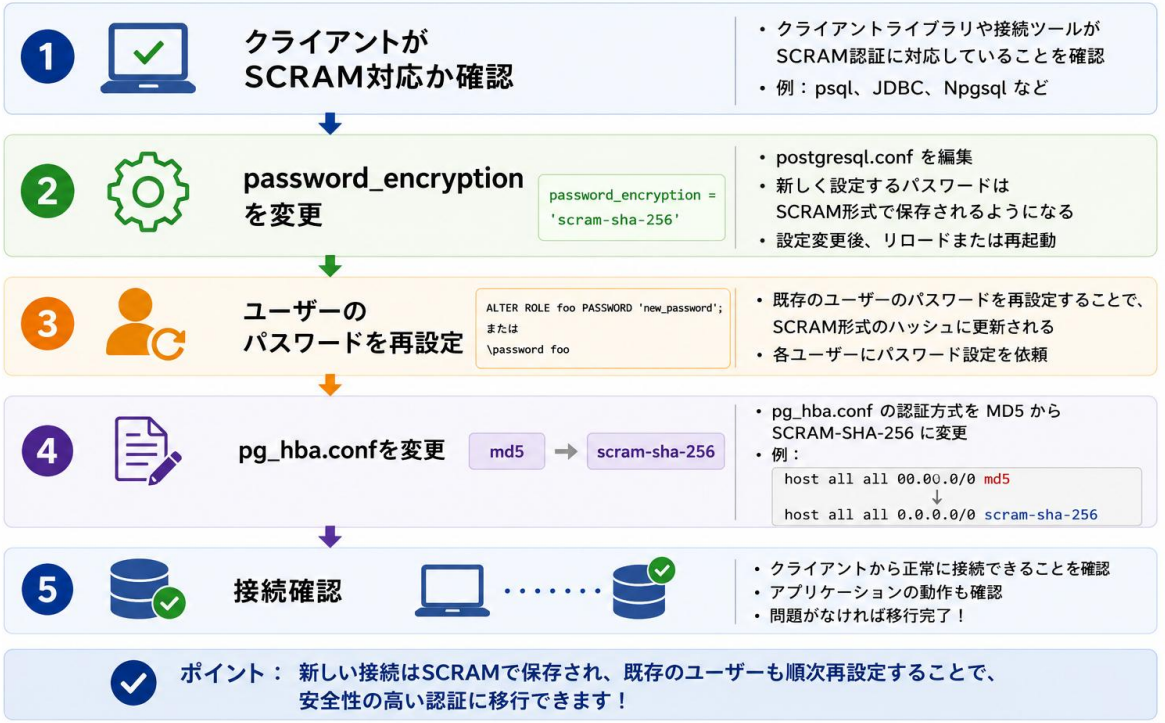
特徴

- PostgreSQL 10から利用可能
- passwordやMD5より安全な方式
- 現在の基本的な選択肢

pg_hba.conf

```
host  all  all  192.168.1.0/24  scram-sha-256
```

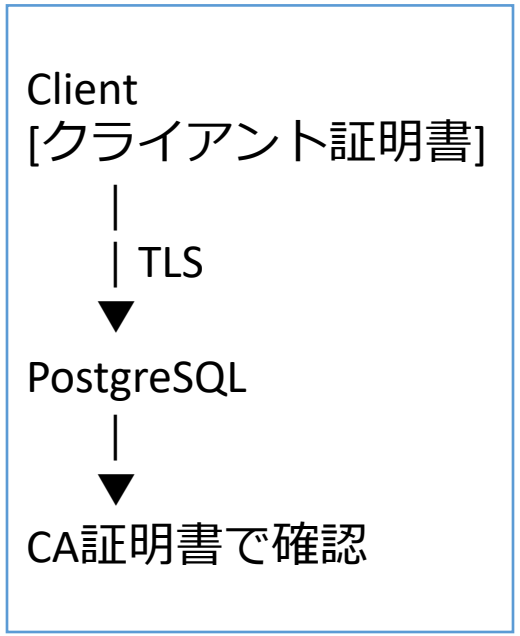
MD5認証からSCRAM-SHA-256認証への移行手順



代表的な認証方式

- 証明書認証

仕組み



特徴

- クライアント証明書を利用
- パスワード不要
- SSL/TLSが前提

pg_hba.conf

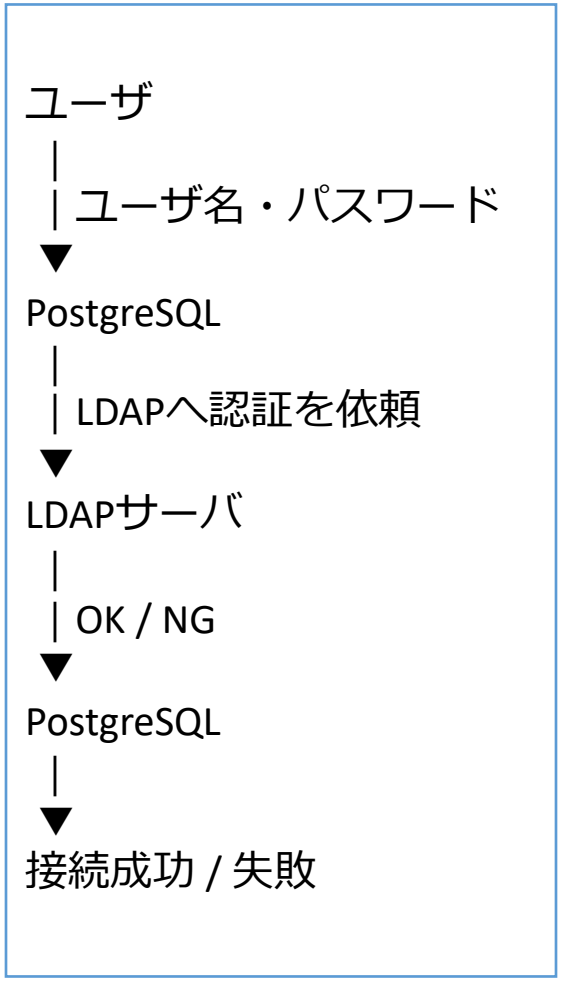
```
hostssl    mydb       myuser     192.168.1.0/24    cert
```

適した場面：システム間接続

代表的な認証方式

- LDAP

仕組み



特徴

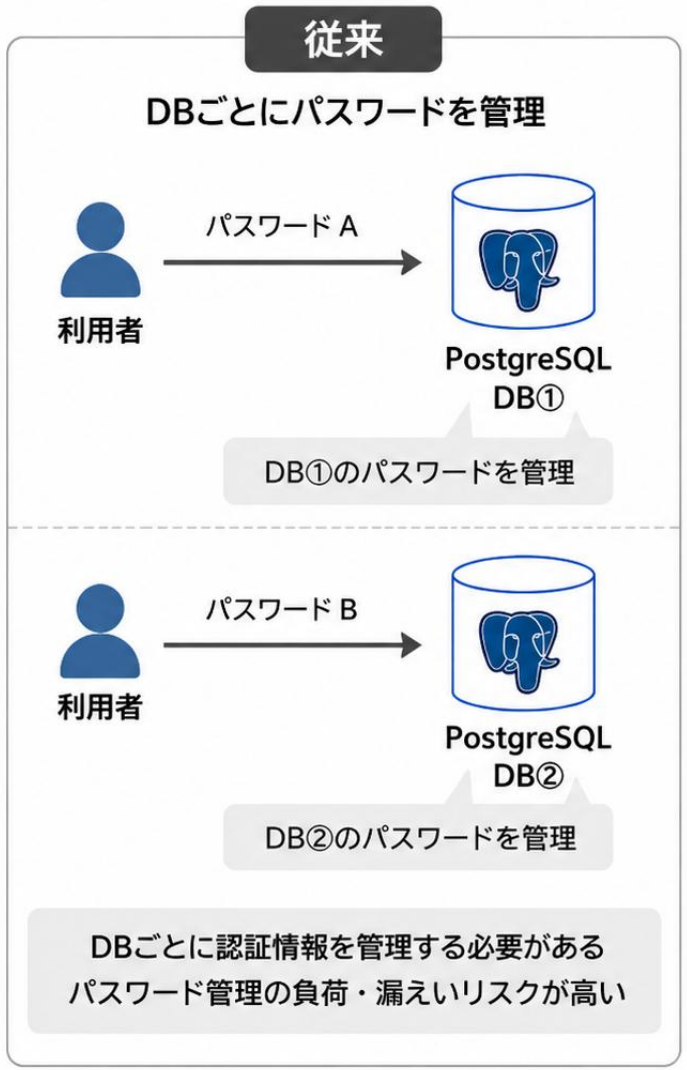
- LDAPでユーザ名・パスワードを検証
- Simple Bind / Search + Bindの2方式
- LDAPユーザとは別にPostgreSQLロールが必要
- LDAP/Active Directoryなど既存の認証基盤を利用できる
- LDAPとの通信をTLSで暗号化可能

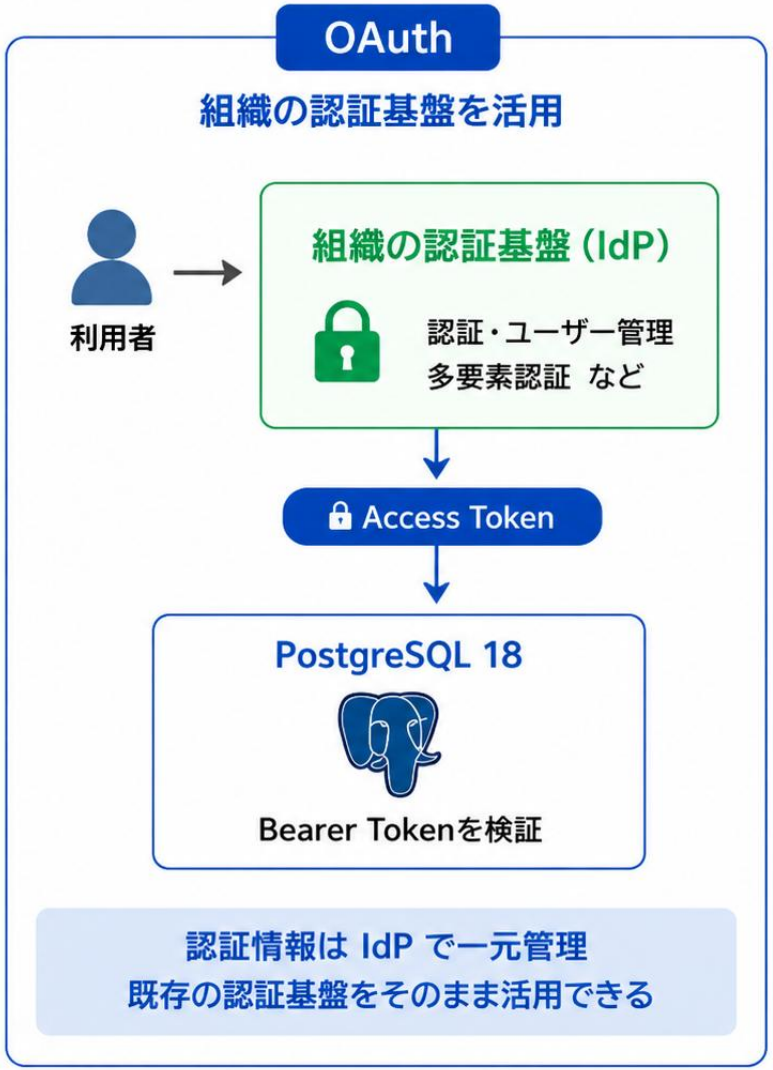
pg_hba.conf

```
host  all  all  192.168.1.0/24  ldap
¥  ldapserver=ldap.example.net
¥  ldapbasedn="dc=example,dc=net"
¥  ldapsearchattribute=uid
```

PostgreSQL18 OAuth

- 背景



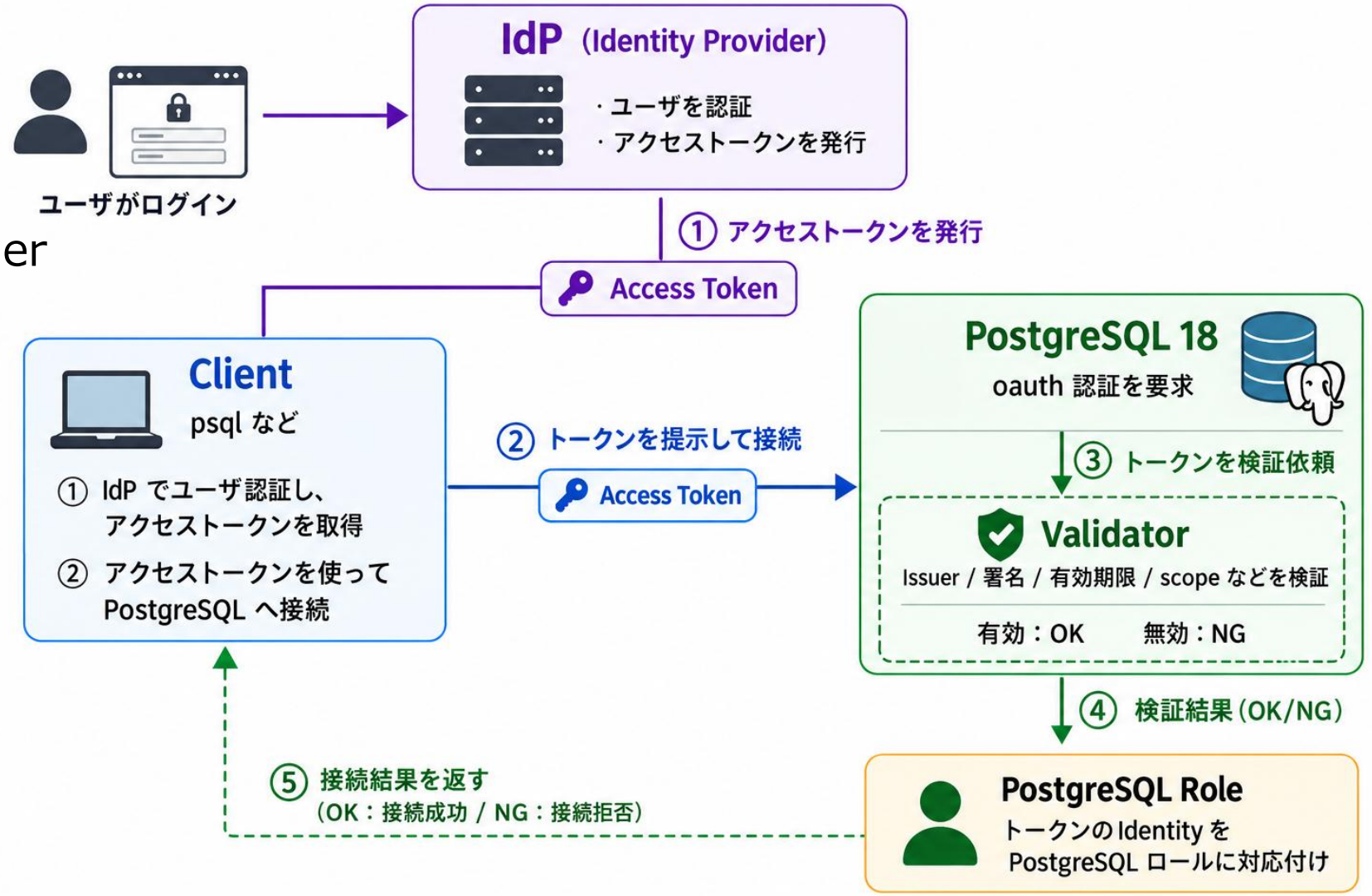


PostgreSQL 18

- OAuth 2.0による認証をサポート
- パスワードの代わりにBearer Tokenを利用
- 外部のIdPと連携
- 既存の認証基盤を活用可能

PostgreSQL18 OAuth

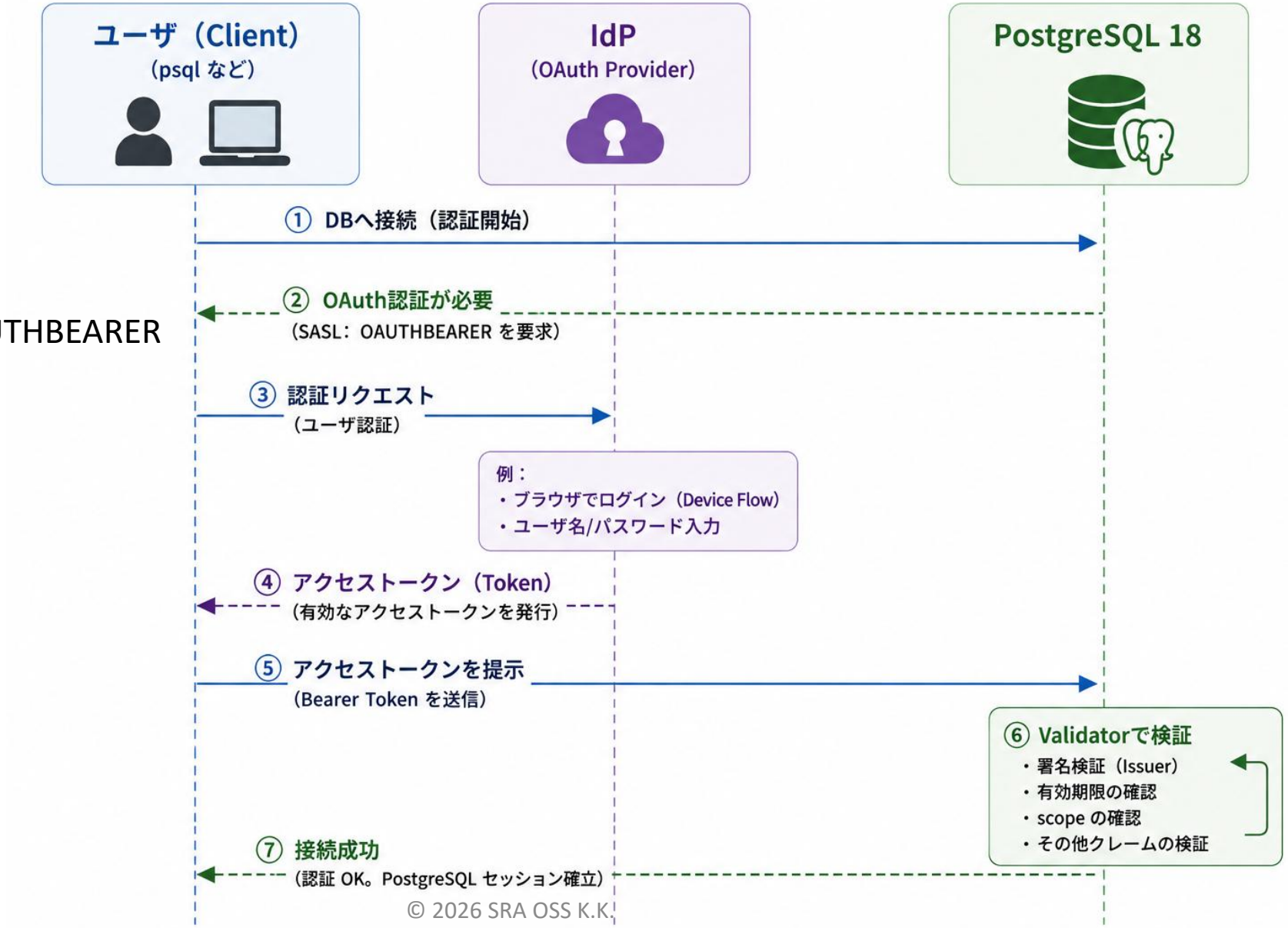
- 要素
- Client
- PostgreSQL Server
- OAuth 2.0 Identity Provider
- Validator



PostgreSQL18 OAuth

- 認証の流れ

PostgreSQLではSASLのOAUTHBEARERを利用。



PostgreSQL18 OAuth

- 設定

pg_hba.conf

```
host  all  tester  192.168.1.0/24  oauth
¥  issuer="https://idp.example.com"
¥  scope="openid profile"
¥  validator=my_validator
¥  map=my_oauth_map
```

postgresql.conf

```
oauth_validator_libraries = 'my_validator'など
```

項目	内容
oauth	OAuth認証を利用
issuer	OAuth Authorization Server
scope	必要なOAuth Scope
validator	Tokenを検証するモジュール
map	OAuth IdentityとDBロールの対応

PostgreSQL18 OAuth

- OAuth認証でもPostgreSQLロールは必要

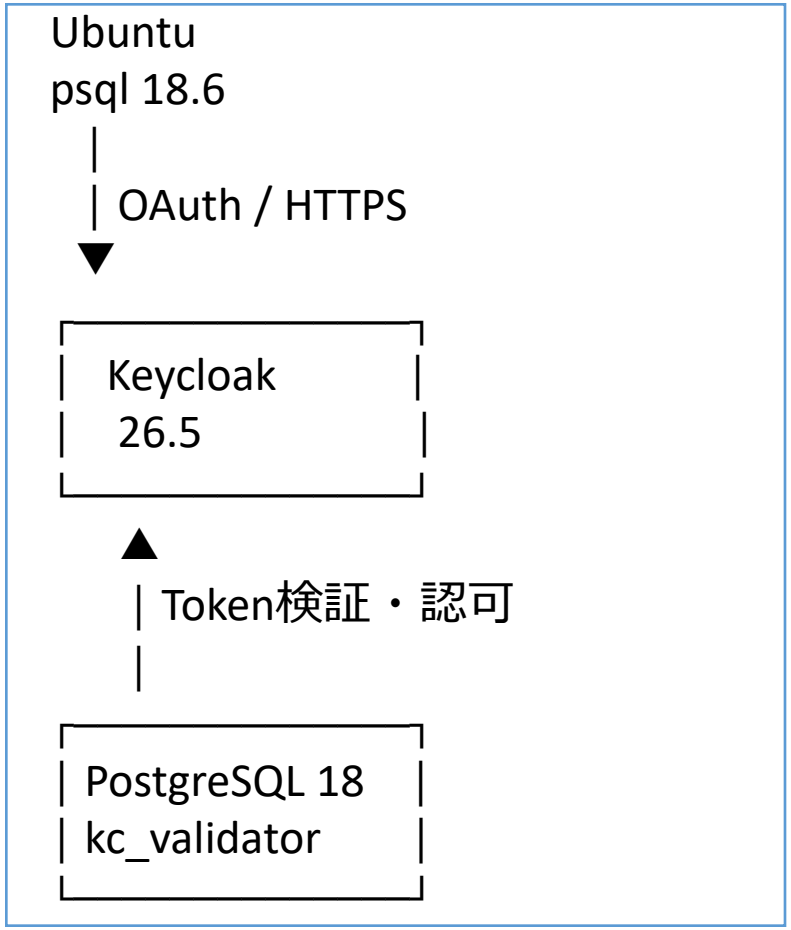
外部Identity
Peggie
↓
Identity Mapping
Peggie → app_user
↓
PostgreSQLロール
app_user

OAuthで本人確認できても、最終的にPostgreSQLへ接続するためのロールは必要。

PostgreSQL18 OAuth 検証

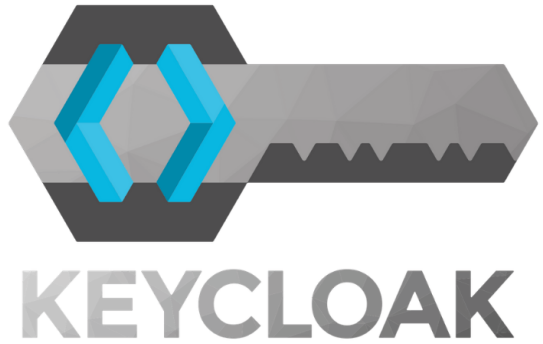
【検証環境】

- **Client**
  - Ubuntu
  - psql 18.6
  - libpq-oauth
- **PostgreSQLサーバ**
  - PostgreSQL 18.6
  - Docker
  - PostgreSQL OAuth Validator for Keycloak
- **IdP**
  - Keycloak 26.5
  - Docker
  - HTTPS



PostgreSQL18 Oauth 検証

- Keycloak（キークローク）は、アプリケーションやサービスの認証・認可を一元管理するオープンソースのIdentity and Access Management（IAM）製品
- 主な特徴：
  - シングルサインオン（SSO）
  - OpenID Connect / OAuth 2.0 / SAMLに対応
  - ユーザ・ロール・権限の管理
  - 外部LDAP / Active Directoryとの連携
  - アクセストークンの発行
  - Authorization Servicesによるアクセス制御



- 今回の検証では、Keycloakを**OAuthのIdentity Provider（IdP）／Authorization Server**として利用している

PostgreSQL18 Oauth 検証

【設定】

PostgreSQL側

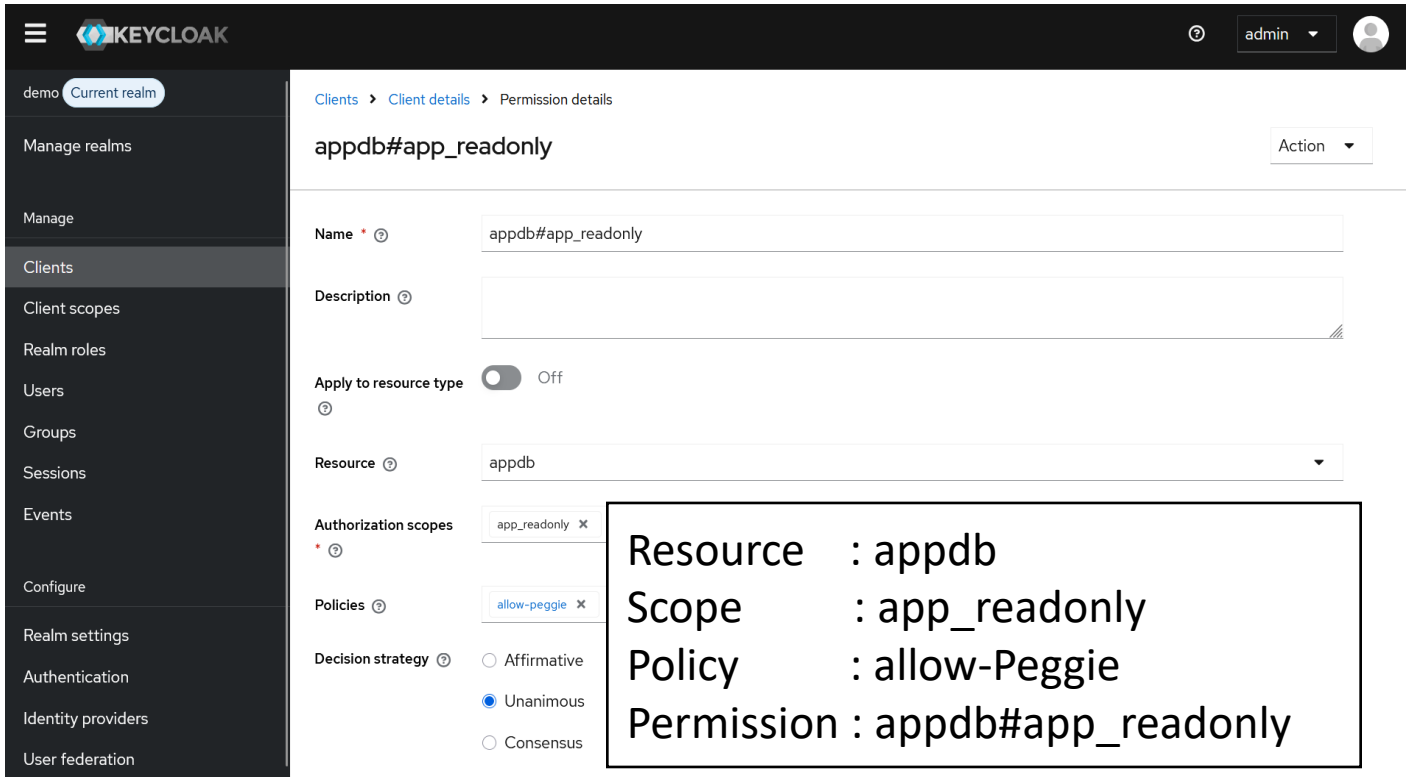
- pg_hba.conf

```
host all all 0.0.0.0/0 oauth
issuer="https://localhost:8443/realms/demo"
scope="db_access"
validator="kc_validator"
delegate_ident_mapping=1
```

- postgresql.conf

```
oauth_validator_libraries = 'kc_validator'
kc.audience = 'postgres-resource'
kc.resource_name = 'appdb'
kc.client_id = 'postgres-resource'
kc.token_endpoint =
'https://keycloak:8443/realms/demo/protocol/openid-connect/token'
```

Keycloak側



Resource : appdb
Scope : app_readonly
Policy : allow-Peggie
Permission : appdb#app_readonly

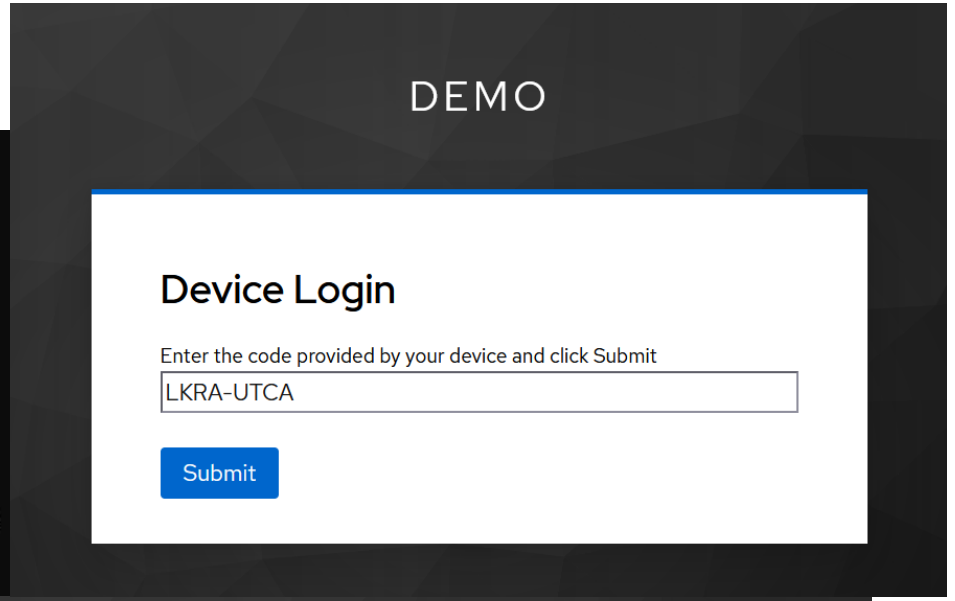
PostgreSQL18 Oauth 検証

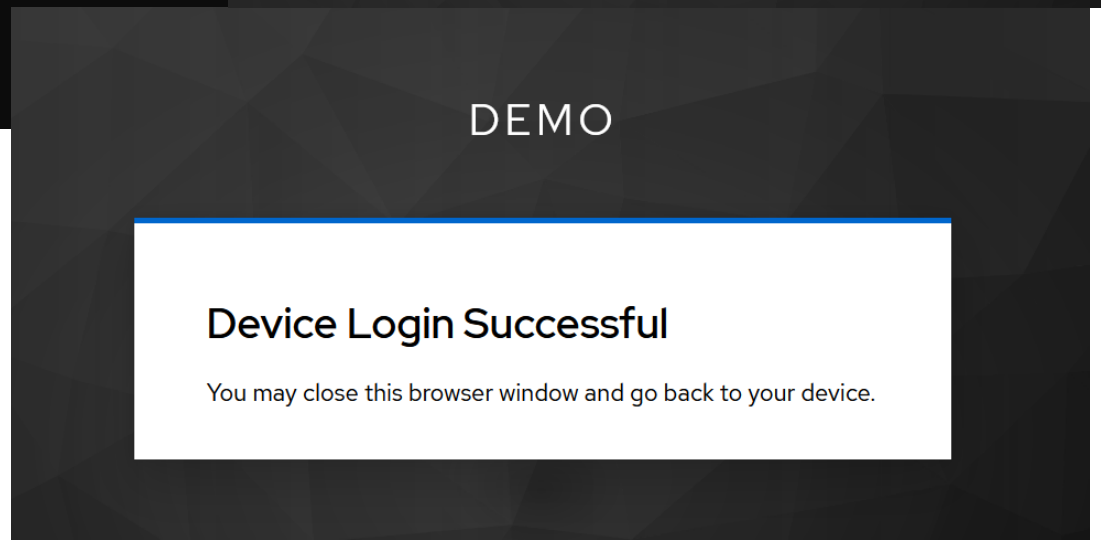
【psqlからOAuth認証で接続】

```
ma@[redacted]:~/pg18-oauth-demo$ psql "host=localhost \
port=5432 \
dbname=appdb \
user=app_readonly \
oauth_issuer=https://localhost:8443/realms/demo \
oauth_client_id=appA \
oauth_client_secret=[redacted]\
oauth_scope=db_access"
Visit https://localhost:8443/realms/demo/device and enter the code: LKRA-UTCA
psql (18.6 (Ubuntu 18.6-1.pgdg24.04+2))
Type "help" for help.

appdb=>
```

Client Secret key





その他

PostgreSQLには他にも……

方式	概要
PAM	OSの認証機構
GSSAPI	Kerberosなど
SSPI	Windows統合認証
RADIUS	外部RADIUSサーバ

認証方式の選び方

- 利用場面と主な選択肢

選ぶときは
→ 誰が？
→ どこから？
→ 何で本人確認する？
→ 誰が認証情報を管理する？

認証方式	向いているケース	特徴・選択ポイント
peer	ローカル接続	OSユーザを利用。 パスワード不要
SCRAM-SHA-256	一般的なDB接続	PostgreSQL内でパスワードを管理する場合の基本選択肢
証明書認証	高いセキュリティが必要	クライアント証明書で本人確認。証明書の発行・更新管理が必要
LDAP	組織でLDAPを利用済み	既存のユーザ・パスワード管理を活用できる
OAuth	IdP・SSOを利用する環境	外部IdPで認証。トークンを利用し、パスワードをDBへ直接渡さない

まとめ

- ① **pg_hba.conf**
  接続条件に応じて認証方式を決定
- ② **認証方式**
  peer / SCRAM / 証明書 / LDAPなど、それぞれ特徴が異なる
- ③ **PostgreSQL 18**
  OAuth認証が新たな選択肢に
- ④ **大切なのは**
  環境・利用者・既存の認証基盤に応じて選択すること

ご清聴ありがとうございました。

製品・サービスに関するお問い合わせ: sales@sraoss.co.jp 03-5979-2701

SRA OSS 技術情報

- SRA OSS Tech Blog
  - https://www.sraoss.co.jp/tech-blog/
  - PostgreSQLのリリースノートの日本語訳など様々なOSS技術情報を掲載中
- SRA OSS 公式 Youtube チャンネル
  - https://www.youtube.com/c/sraoss-official
  - 過去のセミナー動画を公開中