Citus Con:
An Event for Postgres 2023

Hosted by
Microsoft

Azure AD authentication with PostgreSQL Flexible Servers

Andrey Chudnovskiy

Software Engineer - Azure Database for PostgreSQL

#CitusCon | April 18-19, 2023

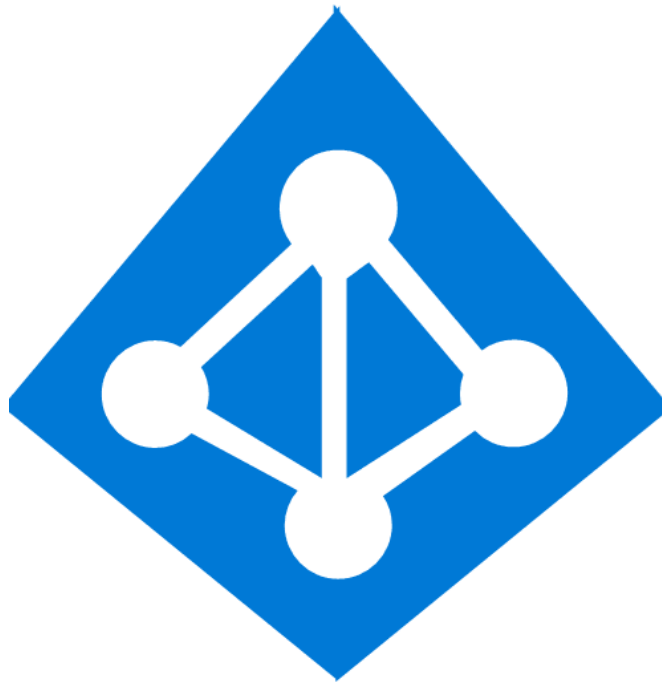
About Me

Microsoft

Google

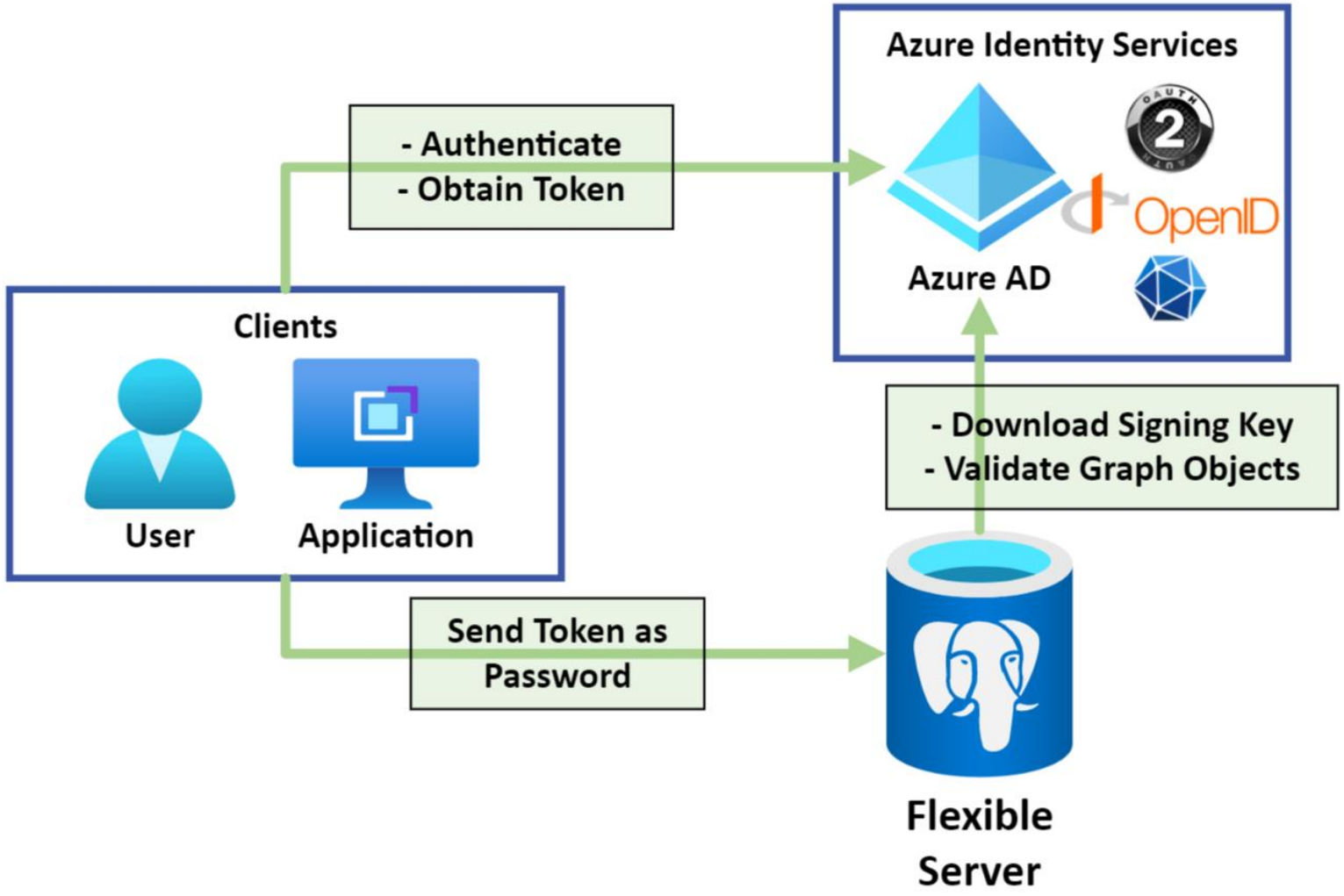
Azure Active Directory

- Cloud-based identity and access management (IAM)
- Implements industry standards for authentication: OAUTH2, OpenId Connect (OIDC)
- Multiple authentication factor options
- Advanced enterprise security and management features.
- Microsoft Graph – Users, Groups, Applications and Identities metadata API



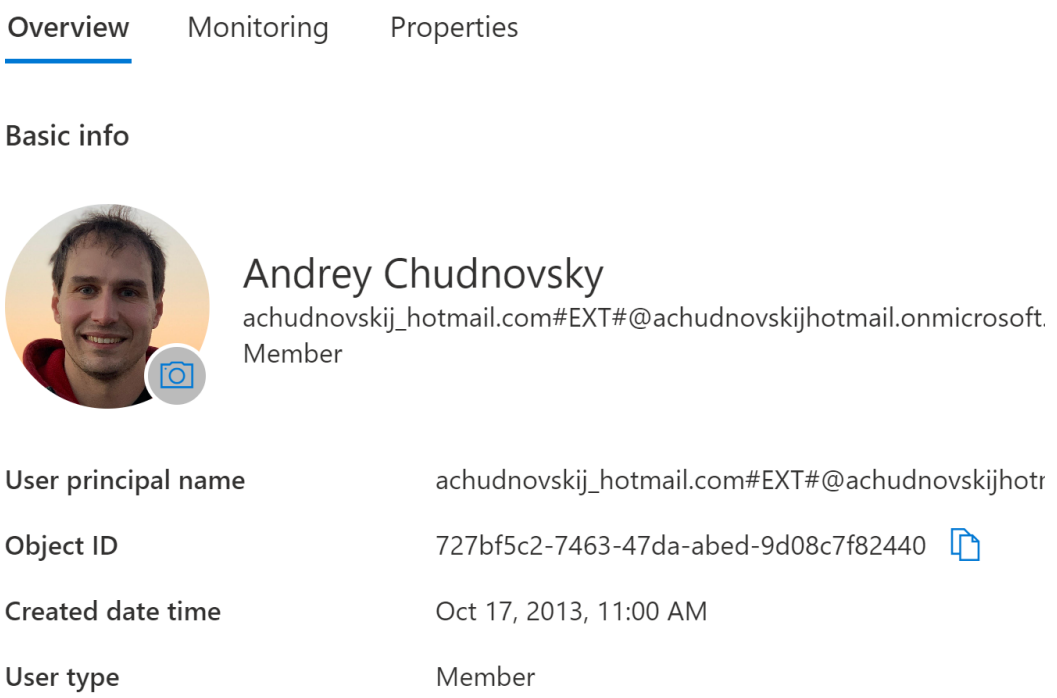
Azure AD Authentication in Flexible Servers

- Authentication exchange between Client and AAD
- Token as Password
- PostgreSQL validates token



Azure AD is integrated throughout Azure

- Every Azure user is Azure AD user.
- Find your Tenant with Sign In status
- Azure AD can be managed in Portal
- Azure AD authentication works with free Azure AD tier and on the minimal PostgreSQL Flexible SKUs



Quick Start -Setup

1. Create Server	`az postgres flexible-server create --name "expenses-db" --resource-group "expenses-app" --active-directory-auth "Enabled“`
2. Create Admin (via CLI)	`az postgres flexible-server ad-admin create --resource-group "expenses-app" --server-name "expenses-db" --display-name "admin@expenses.app" --object-id "<object_id>" --type "User“`

Quick Start - Connect

3. Get Token	`export PGPASSWORD=$(az account get-access-token --resource-type oss-rdbms | jq -r .accessToken)`
4. Connect	`psql "host=server-name.postgres.database.azure.com port=5432 dbname=postgres user=admin@expenses.app“`

Look Inside Azure AD Role in PG

```
select * from pgaadauth_list_principals(false);

rolname       | admin@expenses.app
principaltype | user
objectid      | f1d3f32b-8845-4b8d-94df-ff4e7a96ef05
tenantid      | <Tenant_id>
ismfa         | 0               // Is Multi factor enforced for role
isadmin       | 1               // See Azure Ad Admin below

select * from pg_shseclabel;    // You can look inside here to see how we store mapping
```

Azure AD Admin has privileges of regular Azure PG Admin role and can create other Azure AD enabled roles.

Security Principal Types

- “user” - Users
- “service” - Service Principals
  - Applications – "Clients" in OAUTH2. Can access with their own credentials or on user behalf
  - Managed Identities – Service Principal managed by Azure
- “group” - Groups – Can contains users or service principals

*Object Id – Guid of User, Group or Service Principal.
Use “Enterprise Applications” page in Azure Portal to find Service Principal Object Id

Note: Azure AD Authentication in PostgreSQL: 1 PG Role => 1 AAD Principal.

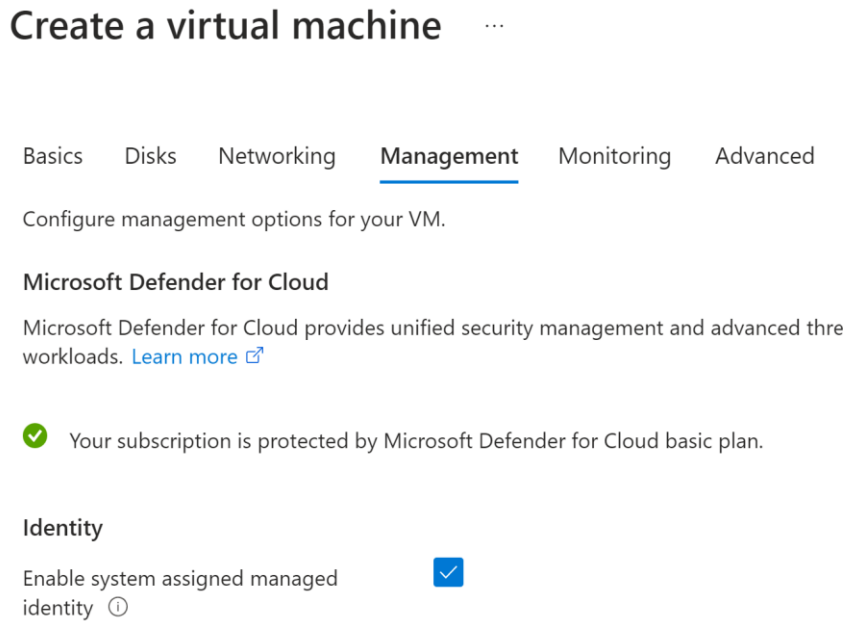
Setup a Managed Identity

Just connection from a VM as application would do using Managed Identity

1. Create an Azure VM with Managed Identity enabled
2. Install psql and jq
3. I'm using name "expenses-vm"
4. From your admin psql :

```
select * from pgaadauth_create_principal
('expenses-vm', false, false);

select * from pgaadauth_list_principals(false);
```



Connect as a Managed Identity

1. Allow VM connection	`az postgres flexible-server firewall-rule create --resource-group "expenses-app" --name "expenses-db" --rule-name "allow-app" --start-ip-address "<vm_ip>" --end-ip-address "<vm_ip>"`
2. Export managed Identity Url	`export url="http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://ossrdbms-aad.database.windows.net/"`

Connect as a Managed Identity

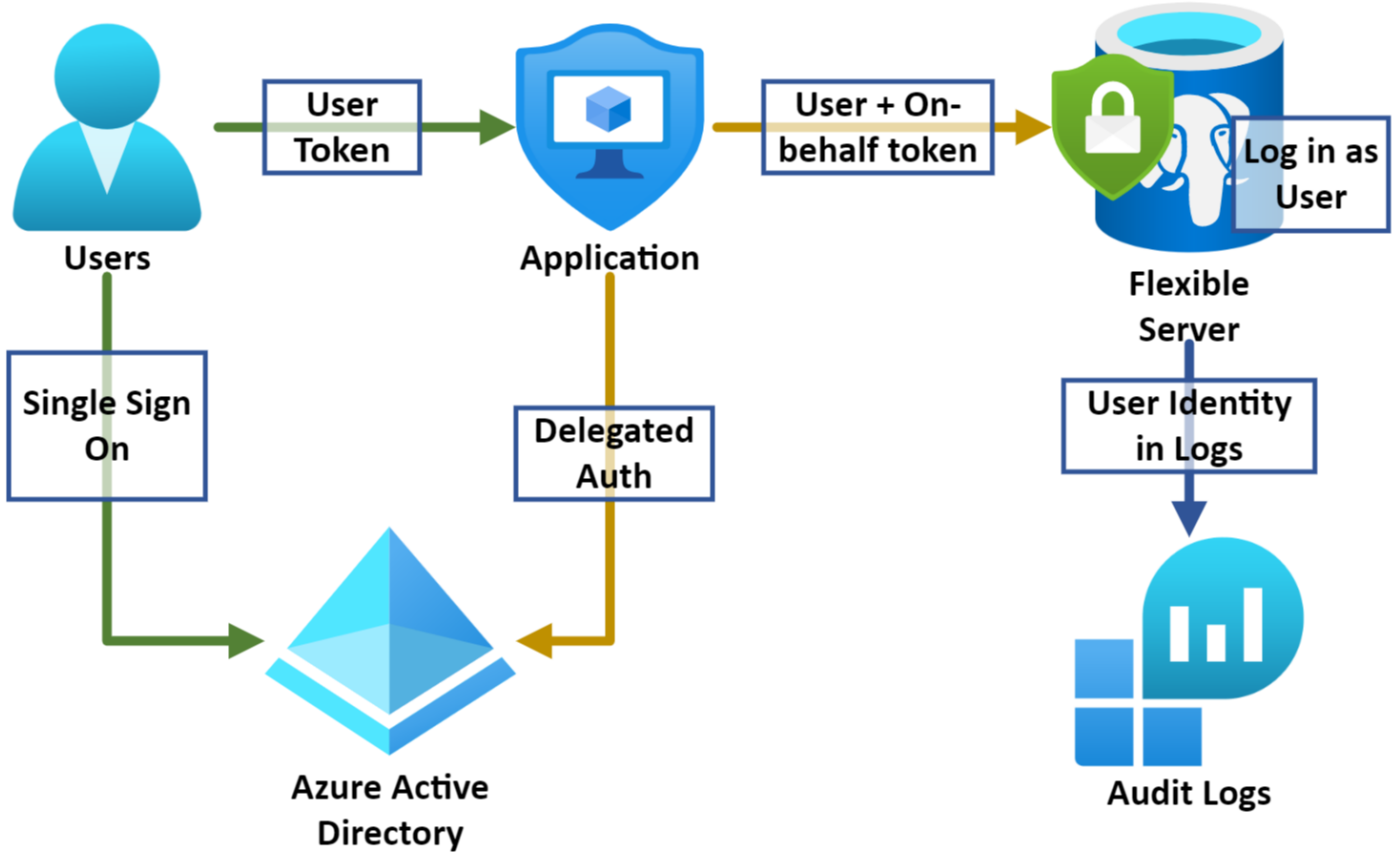
3. Get Token	`export PGPASSWORD=$(curl -H "Metadata: true" “$(url)" | jq -r '.access_token’)`
4. Connect	`psql "host=expenses-db.postgres.database.azure.com port=5432 dbname=postgres user=expenses-vm"`

Repeat with your App Platform

- **Java** – Spring Cloud Azure supports seamless Managed Identity integration with Passwordless Connections
- Use Azure SDKs to get Managed Identity Token:
  - ✓ .Net / .Net core
  - ✓ Node.js
  - ✓ Python
  - ✓ Go
  - ✓ Ruby
- Applications can authenticate with client secrets or certificates using MSAL libraries.
- See https://azure.com/sdk for more info

Impersonation / Delegated Auth

- Application data access controlled by PG
- Row level access to different slices of application data
- Audit of Data operations in database



Login as a Group Member

1. Create group role (as admin)	`select pgaadauth_create_principal("app-users", false, false);`
2. Get user token (as app user)	`export PGPASSWORD=$(az account get-access-token --resource-type oss-rdbms | jq -r .accessToken)`
3. Connect as group role	`psql "host=expenses-db.postgres.database.azure.com port=5432 dbname=postgres user=app-users"`

* Must be security group

Login as a Group Member

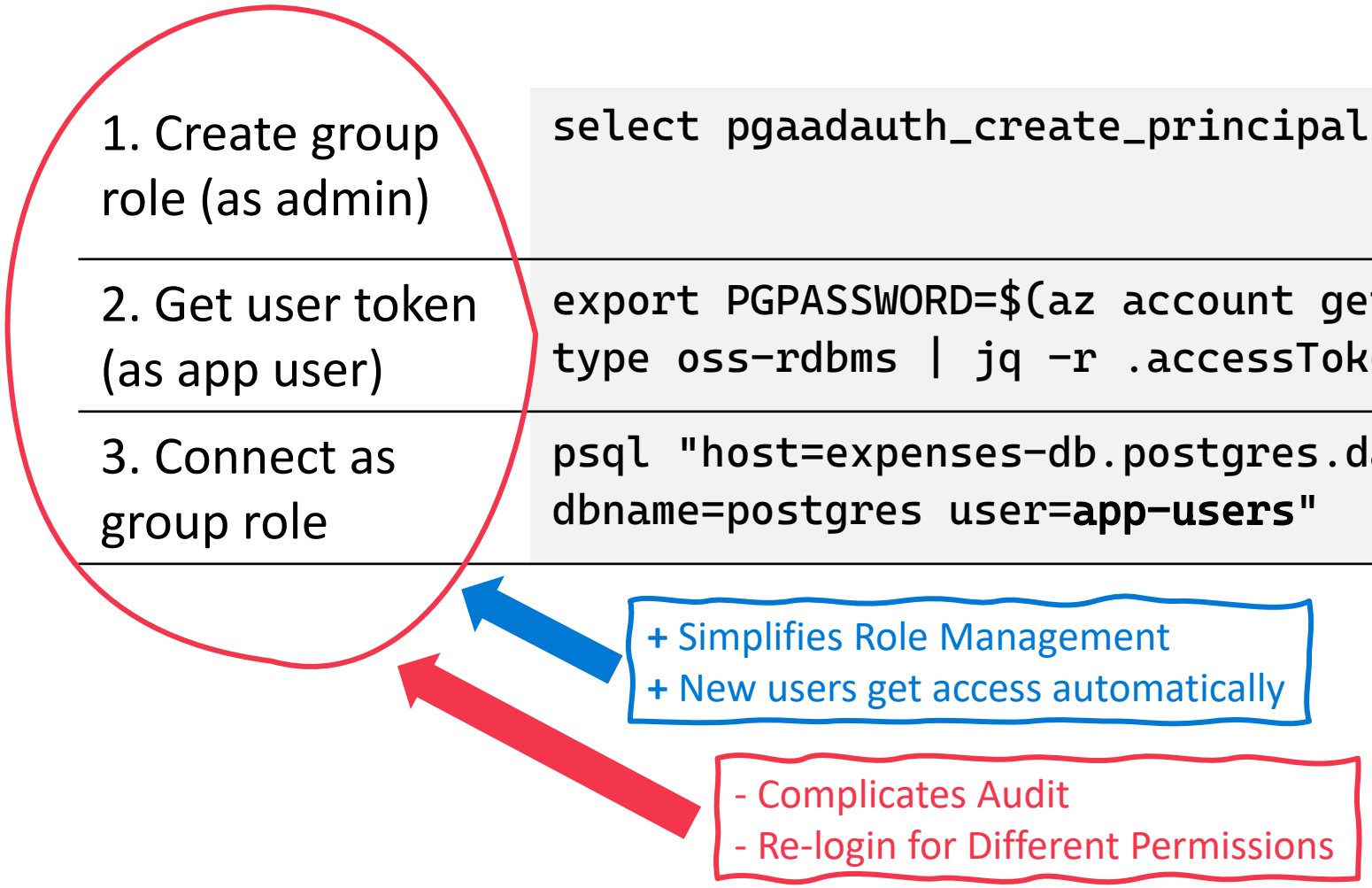
1. Create group role (as admin)	`select pgaadauth_create_principal("app-users*", false, false);`
2. Get user token (as app user)	`export PGPASSWORD=$(az account get-access-token --resource-type oss-rdbms | jq -r .accessToken)`
3. Connect as group role	`psql "host=expenses-db.postgres.database.azure.com port=5432 dbname=postgres user=app-users"`

+ Simplifies Role Management
+ New users get access automatically

* Must be security group

Login as a Group Member

1. Create group role (as admin)	`select pgaadauth_create_principal("app-users", false, false);`
2. Get user token (as app user)	`export PGPASSWORD=$(az account get-access-token --resource-type oss-rdbms | jq -r .accessToken)`
3. Connect as group role	`psql "host=expenses-db.postgres.database.azure.com port=5432 dbname=postgres user=app-users"`



+ Simplifies Role Management
+ New users get access automatically

- Complicates Audit
- Re-login for Different Permissions

* Must be security group

Advanced Role Management

- Disambiguate non-unique names
- Create readable role names
- Re-map existing role to a new AAD principal

Create role	`pgaadauth_create_principal_with_oid ('app-users', 'e2942019-441e-4cd9-8928-6cfebbaded5a', 'group', true, false);`
Update role	`pgaadauth_update_principal_with_oid ('app-users', 'e2942019-441e-4cd9-8928-6cfebbaded5a', 'group', true, false);`
Parameters	- Role Name - Object Id - Object Type - Is Admin - Is Mfa (only accept tokens with mfa claim)

Final Points

Throughput / Performance	• No slowdown for users / services • First time delay for user/group combo
PgBouncer	• Supported with Build-in PgBouncer
VNet Deployments	• Connection to Azure AD services must be open
Cross Tenant	• Objects must be in the Server subscription tenant

Thank you!

- Link to my slides @ aka.ms/XXX
- More Citus Con talks @ aka.ms/cituscon
- linkedin.com/in/achudnovskij/
- Microsoft Learn documentation @ aka.ms/docs-azuredbpostgres-authentication