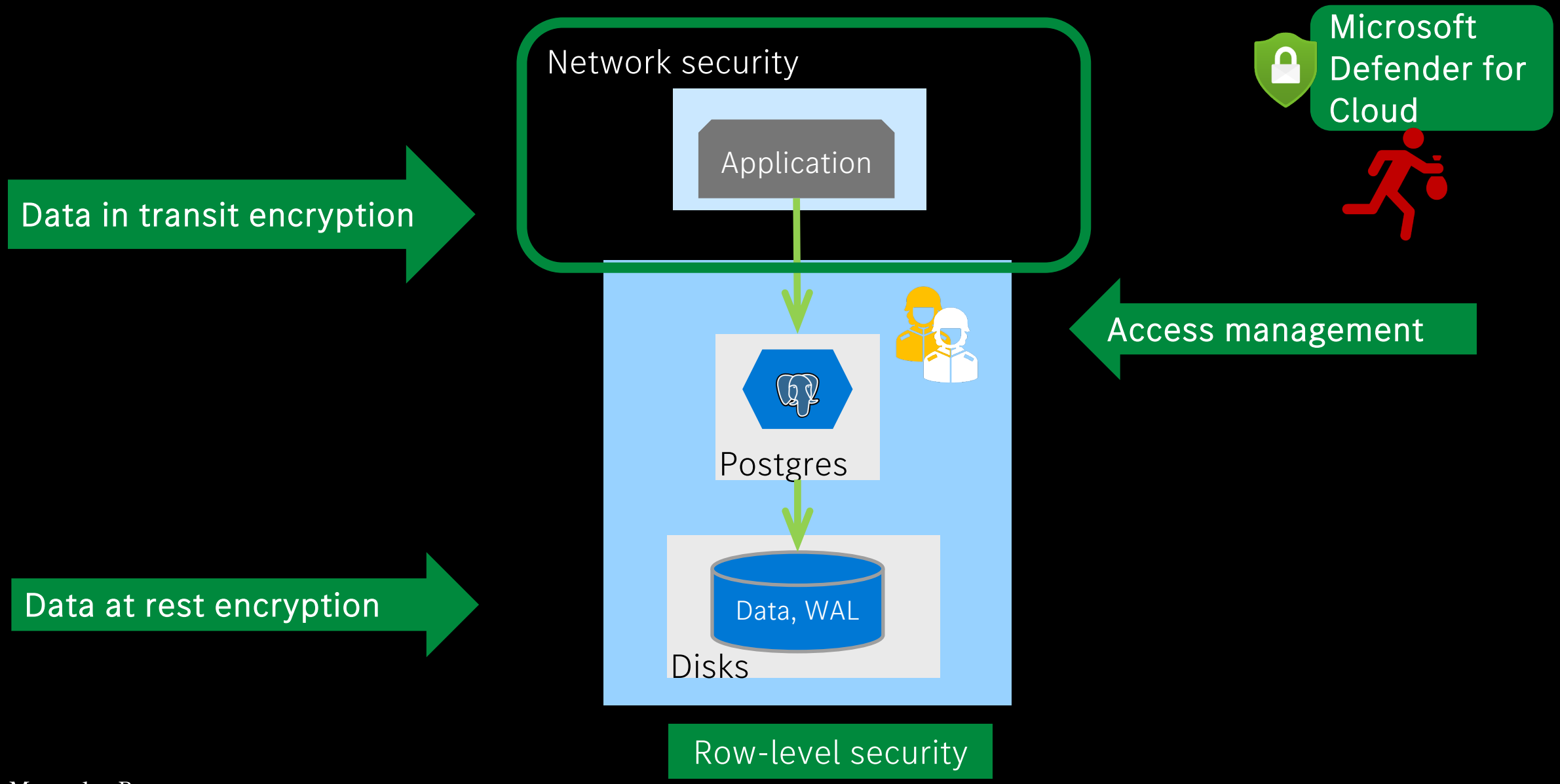
Fortifying Azure Database for PostgreSQL: Stop Intrusions in Their Tracks

Johannes Schuetzner
Mercedes-Benz R&D

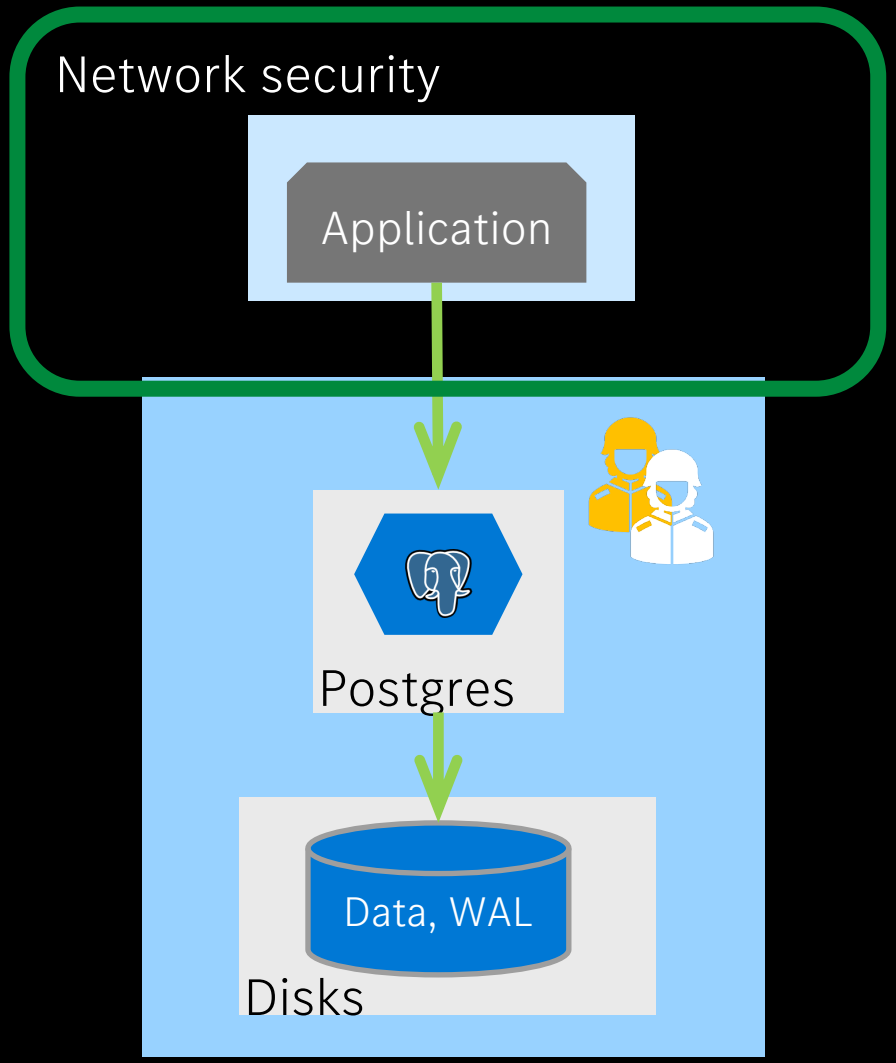
POSETTE, June 2025

Mercedes-Benz

Azure Database for PostgreSQL - Security



Azure Database for PostgreSQL - Security



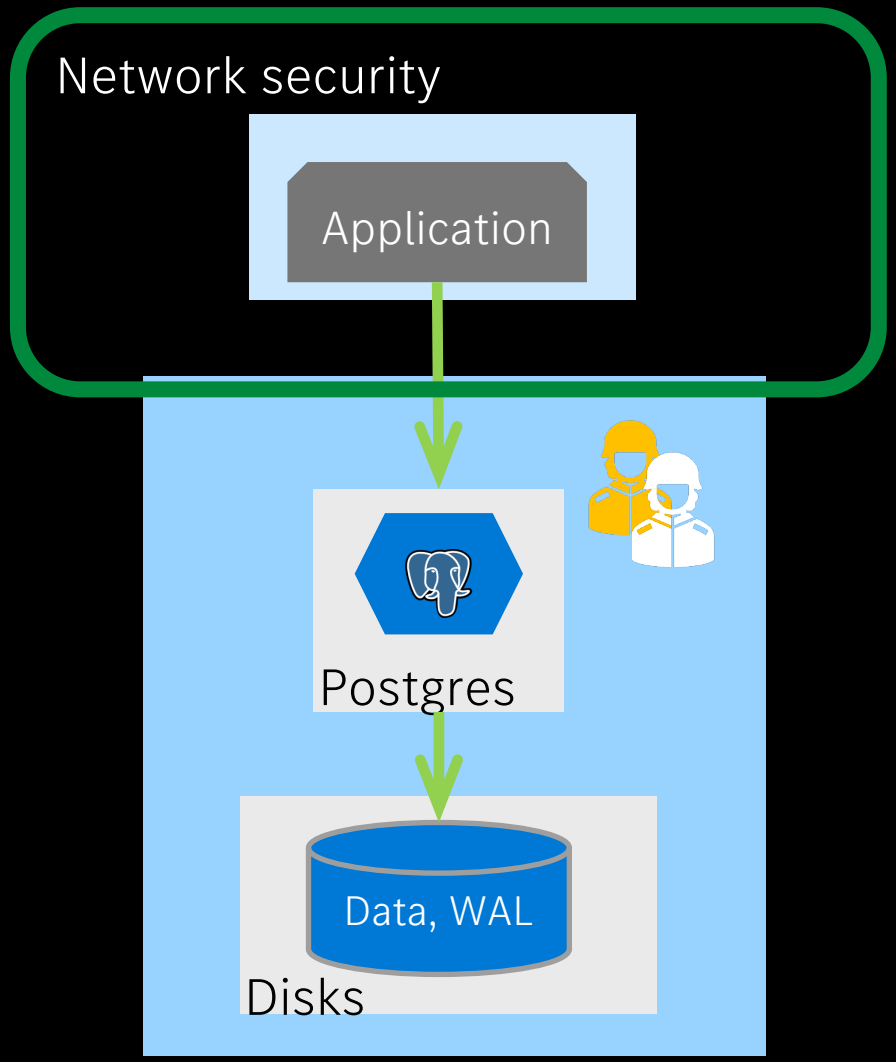
Azure Database for PostgreSQL
Networking Options

Private access: integrated in virtual network

Public access: accessible from public internet

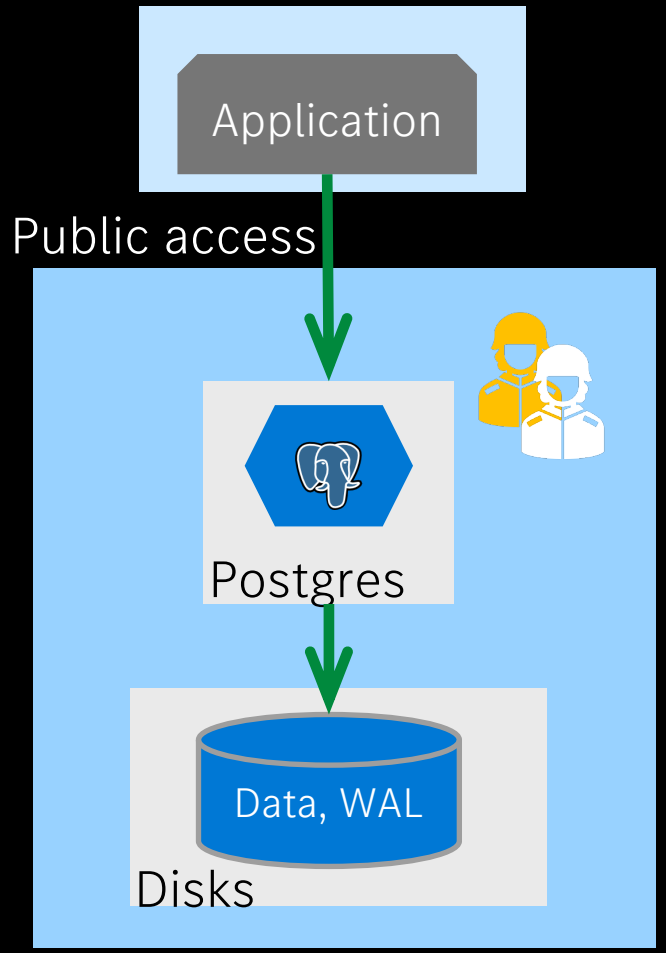
In both cases:

- Authentication needed when connecting
- Network traffic is encrypted using TLS
- PostgreSQL server has domain name (FQDN)
- Authorization rules control access at database and table level

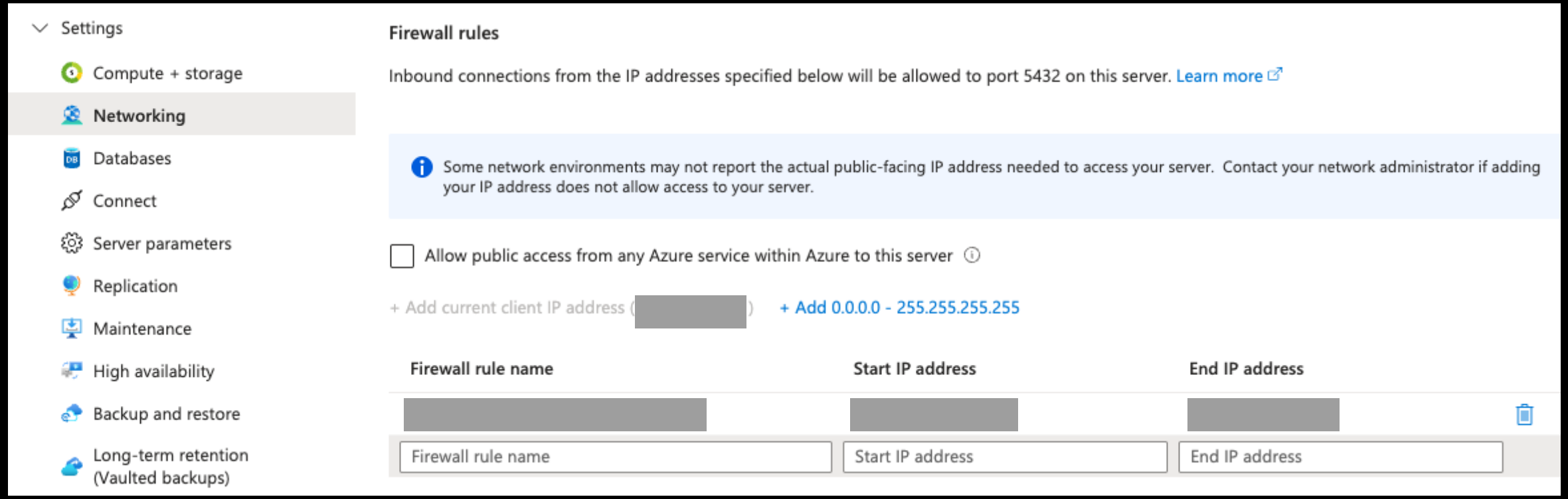


Azure Database for PostgreSQL
Networking with Public Access

- PostgreSQL server has public endpoint
  - Publicly resolvable DNS address
- Firewall rules control range of IP addresses that are allowed to access the server
- PostgreSQL server not part of any VNET
- Traffic goes over public internet pathways



Azure Database for PostgreSQL
Public Access Firewall Rules



Azure Database for PostgreSQL
Public Access Firewall Rules

Provision firewall rules via Terraform (or similar):

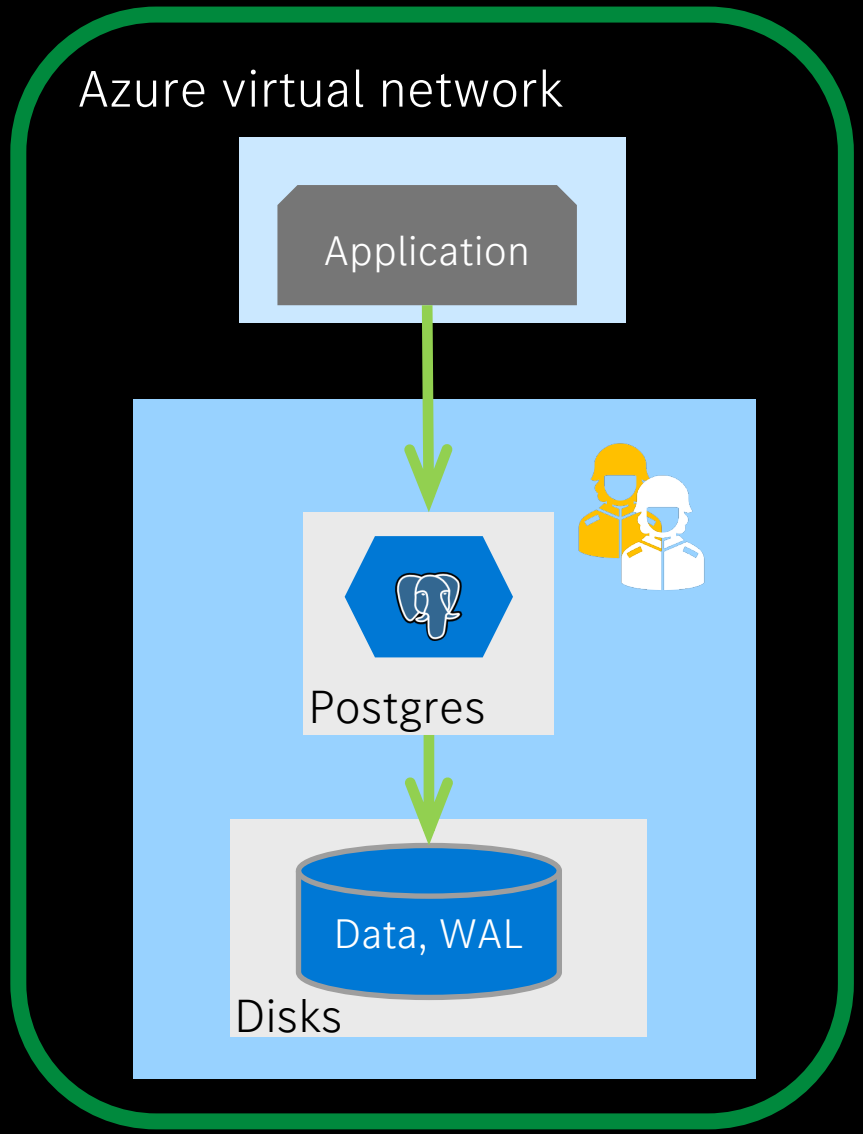
```
resource "azurerm_postgresql_flexible_server_firewall_rule" "ranges" {
 for_each = {   for k, v in var.aks_ip_ranges: k => v  }

 name             = each.value.name
 server_id        = azurerm_postgresql_flexible_server.app[0].id
 start_ip_address = each.value.start_address
 end_ip_address   = each.value.end_address
 ...
```

Azure Database for PostgreSQL
Public Access Drawbacks

- PostgreSQL server vulnerable to external attacks
- Traffic travels over public internet

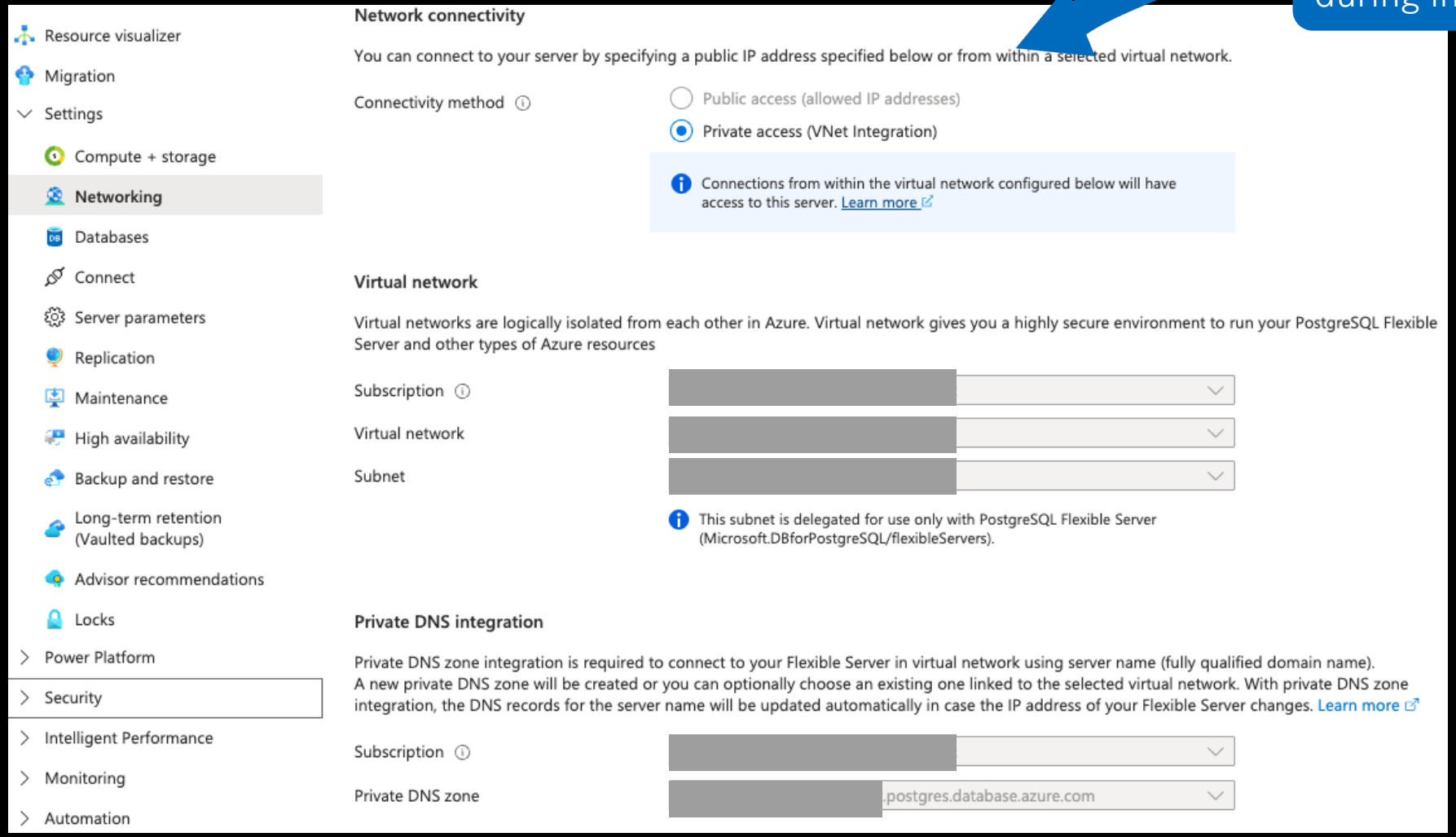
Azure Database for PostgreSQL
Networking with Private Access

- Inject PostgreSQL server into Azure VNET
- Secure network traffic on Azure backbone within VNET
- VNET needs to be in same Azure region
- Delegated subnet needed for PostgreSQL
- If application is in different VNET
  → VNET peering is needed
- Use Azure Private DNS to resolve domain names



Azure Database for PostgreSQL
Private Access VNET Integration

You can configure it only during initial provisioning



Azure Database for PostgreSQL

Private Access VNET Provisioning

```
resource "azurerm_virtual_network" ”usecase" {
 name          = "${var.app_stage}-${var.app_key}-usecase-vnet"
 address_space      = ["10.1.0.0/24", "10.1.1.0/24"]
 …
```

```
resource "azurerm_subnet" "vnet_subnet_for_postgres" {
 name          = "${azurerm_virtual_network.usecase.name}-postgres"
 virtual_network_name = azurerm_virtual_network.usecase.name
 address_prefixes          = ["10.1.0.128/27"]
 …
```

```
resource "azurerm_private_dns_zone" "postgres_zone" {
 name                = "${postgres.name}.postgres.database.azure.com”
 …
```

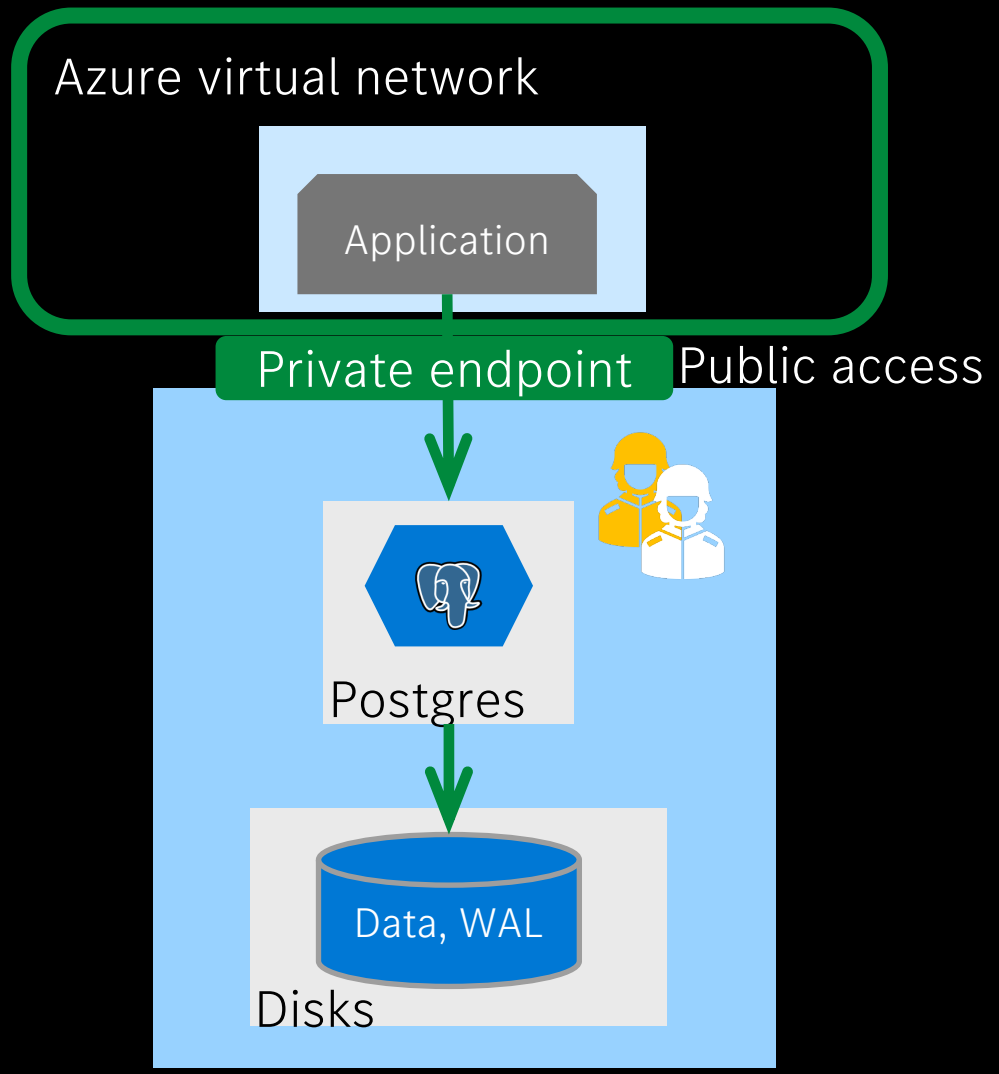
```
resource "azurerm_private_dns_zone_virtual_network_link" "link" {
 private_dns_zone_name = azurerm_private_dns_zone.postgres_zone.name
 virtual_network_id          = azurerm_virtual_network.pipeline.id
```

Azure Database for PostgreSQL
Private Access Considerations

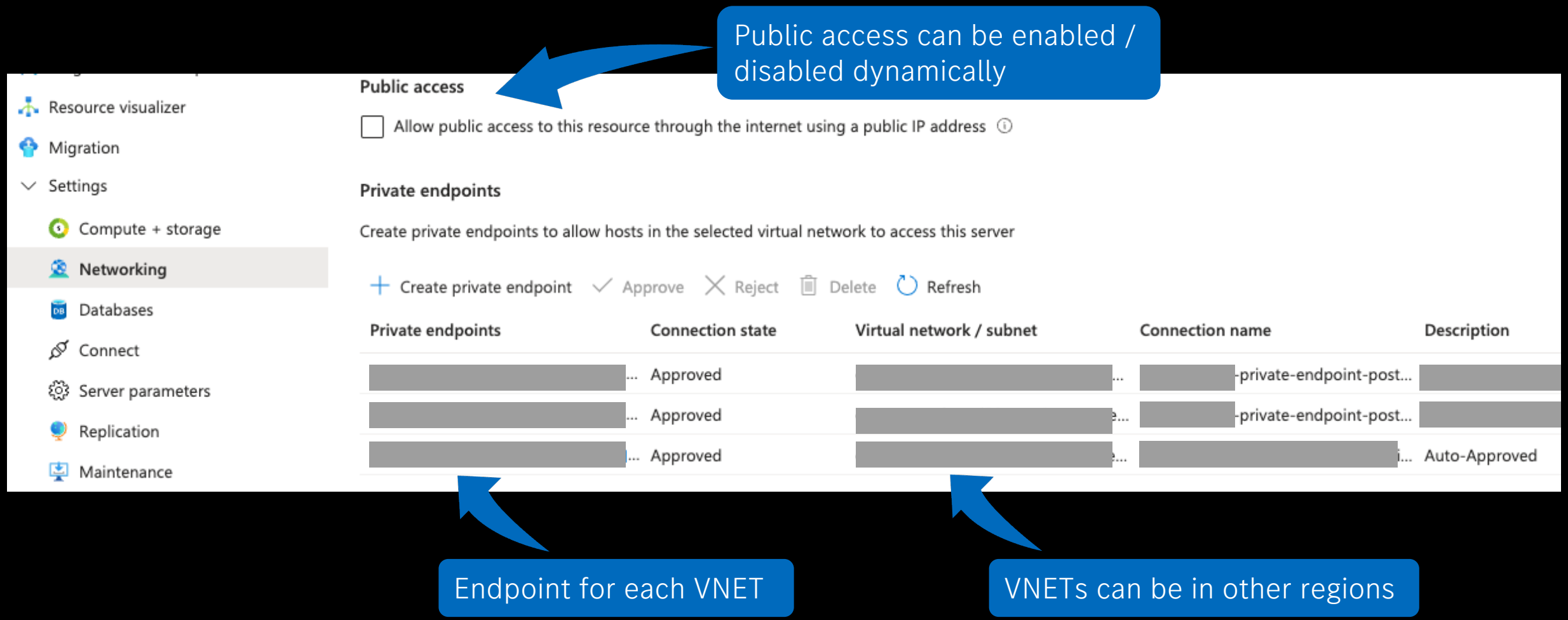
- A VNET is restricted to one Azure region
- Network architecture with multiple VNETs may be complex
- With VNET peering you need to trust the resources in the other VNETs - NSGs work at level of protocol / IP / port
- Data exfiltration may be hard to prevent
- PostgreSQL server cannot be reassigned to another VNET

Azure Database for PostgreSQL
Networking with Private Endpoint

- Based on public access
- Supported for PostgreSQL servers created after private endpoint was introduced (Feb 2024)
- Private endpoint adds network interface to a resource
  - Providing private IP address from a VNET
- Secure network traffic on Azure backbone from VNET to private endpoint
- Additional costs



Azure Database for PostgreSQL
Supports Multiple Private Endpoints



Azure Database for PostgreSQL

Provision Private Endpoint

```
resource "azurerm_private_endpoint" "postgres_private_endpoint" {
 name                          = "${azurerm_postgresql_flexible_server.app.name}-usecase-endpoint"
 resource_group_name = var.vnet_resource_group
 location                      = var.vnet_region
 subnet_id                     = data.azurerm_subnet. vnet_subnet_for_postgres.id
 private_service_connection {
  name                                =  "${azurerm_postgresql_flexible_server.app.name}-connection"
  private_connection_resource_id = azurerm_postgresql_flexible_server.app.id
  is_manual_connection                = false
  subresource_names                   = ["postgresqlServer"]
 }
 private_dns_zone_group {
  name                          = "${azurerm_postgresql_flexible_server.app.name}-dns-zone-group"
  private_dns_zone_ids  = [data.azurerm_private_dns_zone.postgres_zone.id]
 }
...
```

Endpoint is deployed in VNET region

Is approval needed?

Azure Database for PostgreSQL
Networking: Operational aspects

If you cannot access PostgreSQL from your local machine:

- Run PgAdmin (or other tool) within VNET
- For example, in AKS
- www.enterprisedb.com/blog/how-deploy-pgadmin-kubernetes

Alternatively, use Azure Bastion:

```
sudo apt install postgresql-client
export PGHOST=<yourhost>.postgres.database.azure.com
...
psql
```

Azure Database for PostgreSQL
Networking: Operational aspects

To check if private endpoint is enabled for Azure Postgres server:

```
nslookup samplepostgresserver.postgres.database.azure.com
Server:         54.60.5.254
Address:        54.60.5.254#53

Non-authoritative answer:
samplepostgresserver.postgres.database.azure.com
canonical name = samplepostgresserver.privatelink.postgres.database.azure.com.
Name: samplepostgresserver.privatelink.postgres.database.azure.com
Address: 40.113.111.187
```

Azure Database for PostgreSQL
Networking: Operational aspects

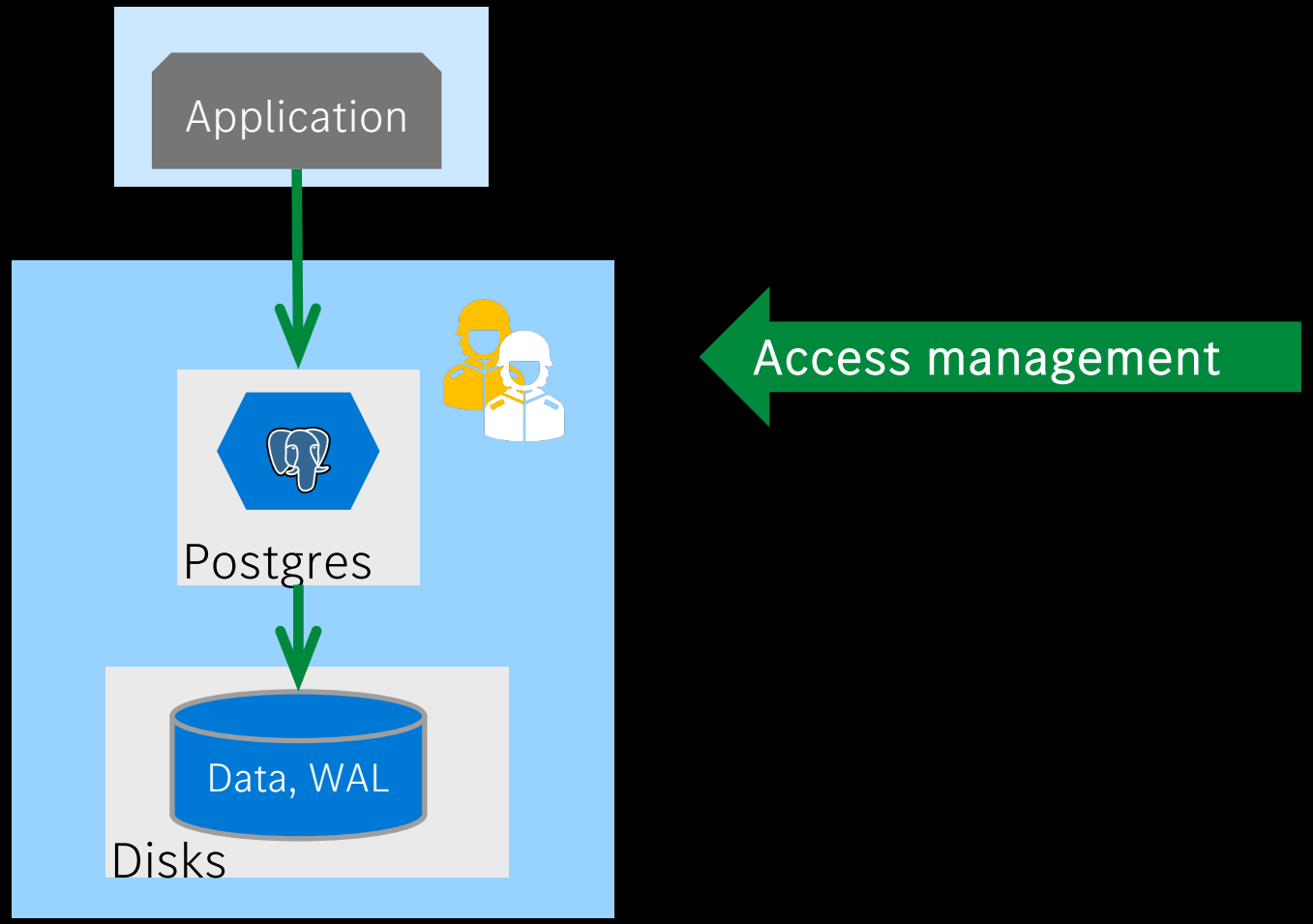
If private endpoint is not operational:

- As long as private endpoint exists, connections from within the VNET use it
- If there is connection issue, testing the access from outside the VNET will not be useful
- If resource can also be accessed publicly, you may drop private endpoint and re-create it

Azure Database for PostgreSQL
Private Endpoint Considerations

- Works at granularity of individual instances of Azure PostgreSQL service
- Explicit control over who has access at source of resource
- Traffic travels over Microsoft backbone network
- In-built data exfiltration protection
- Flexible provisioning and de-provisioning
- Additional costs
- Cannot create private endpoints for Azure Postgres servers with private access

Azure Database for PostgreSQL - Security

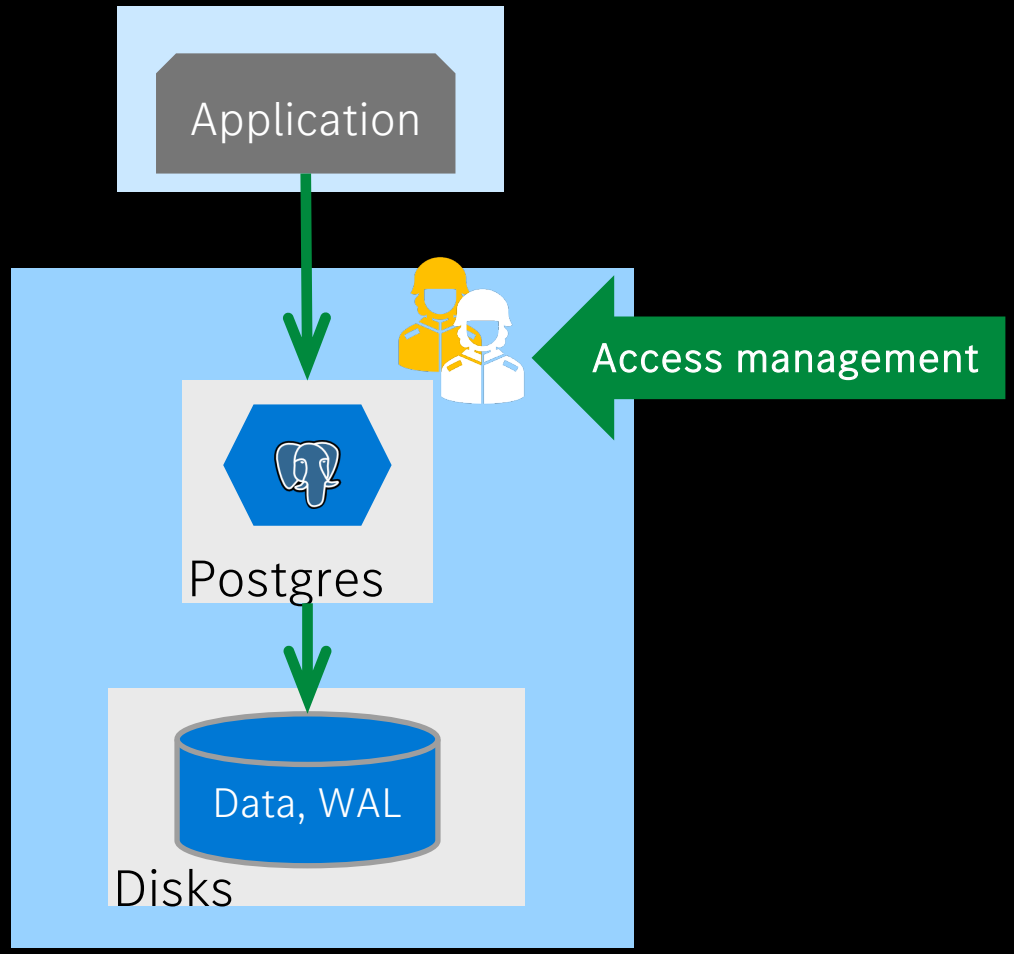


Azure Database for PostgreSQL

Access Management: Authentication and Authorization

Authentication: verify identity

Authorization: check permission for actions



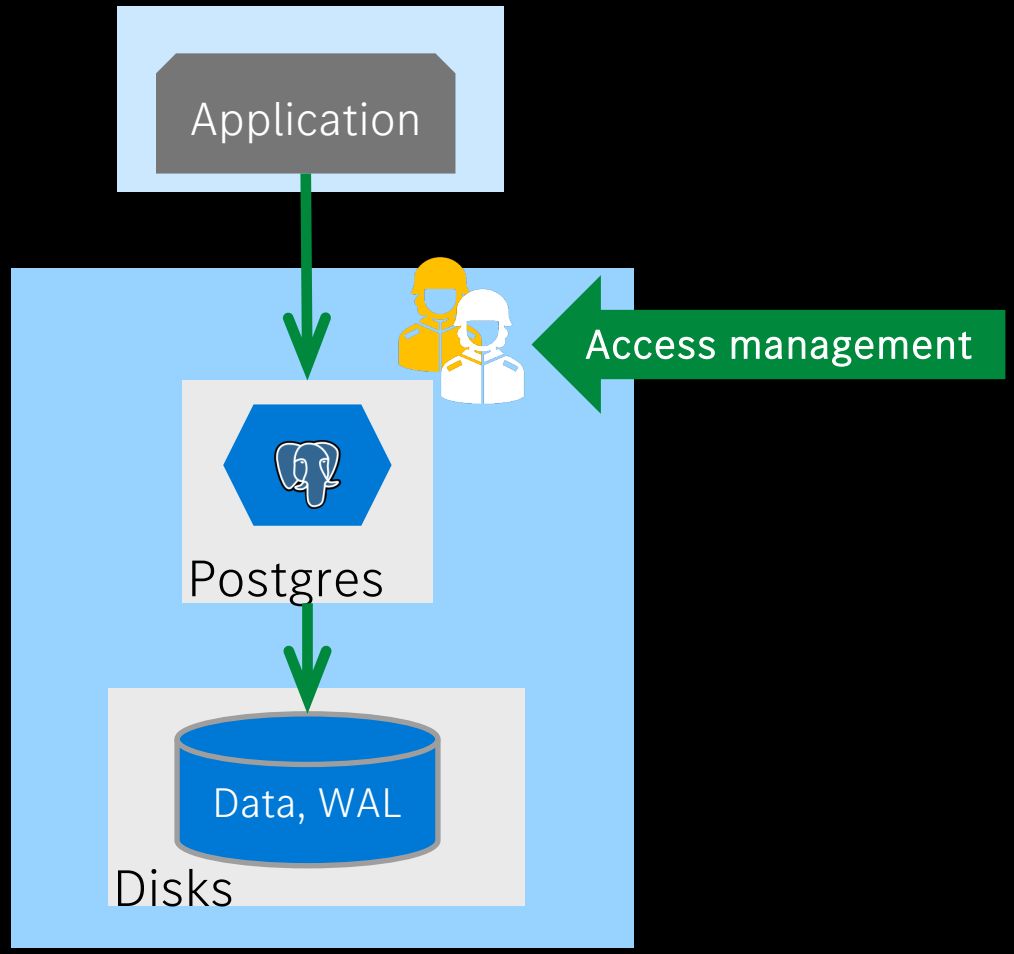
Azure Database for PostgreSQL
Authentication

PostgreSQL authentication:

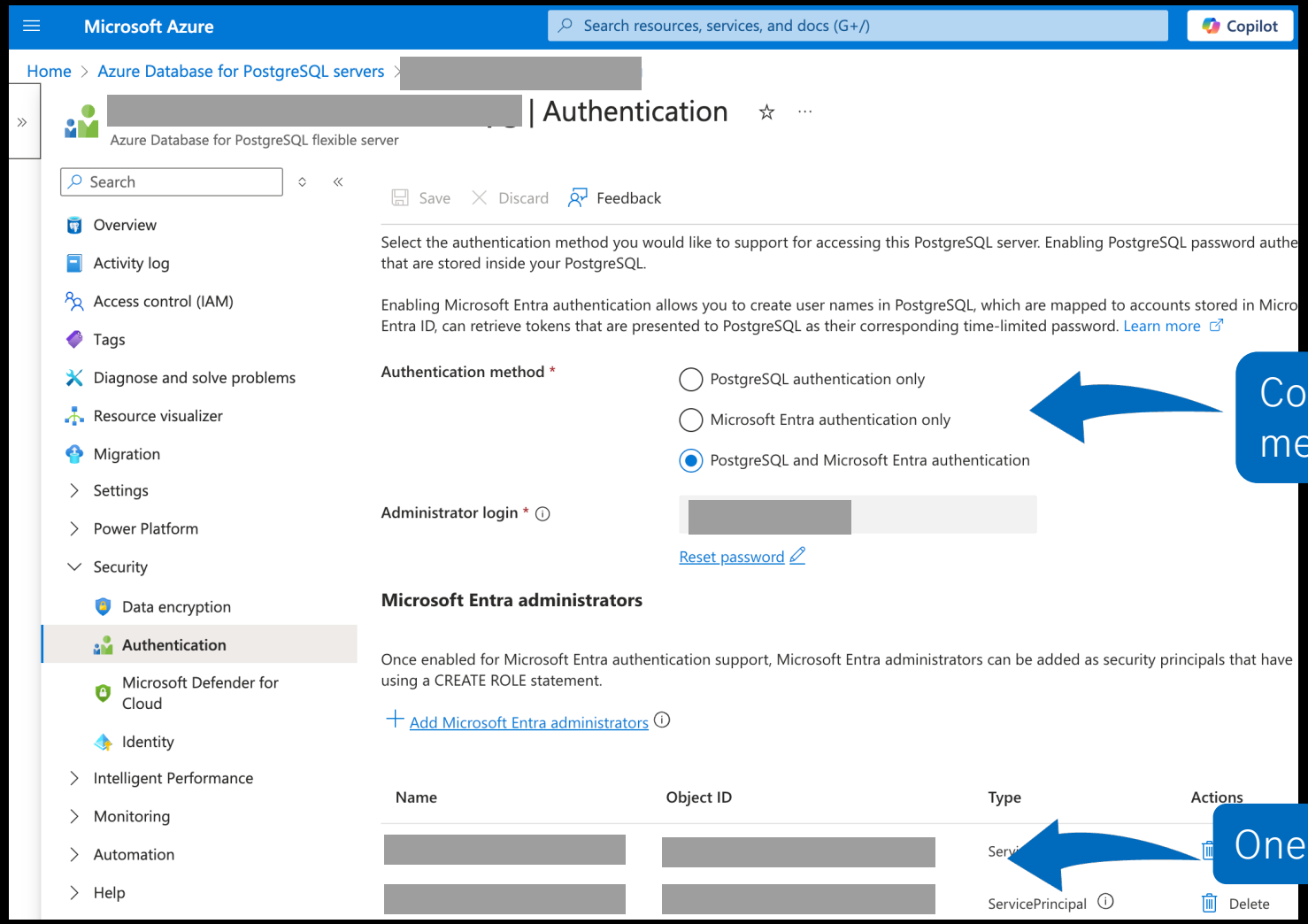
- Using user and password stored in Postgres
- You need to manage it yourself, e.g. password rotation

Entra authentication:

- Uniform management of users
- Option to use Entra groups
- Using password or passwordless

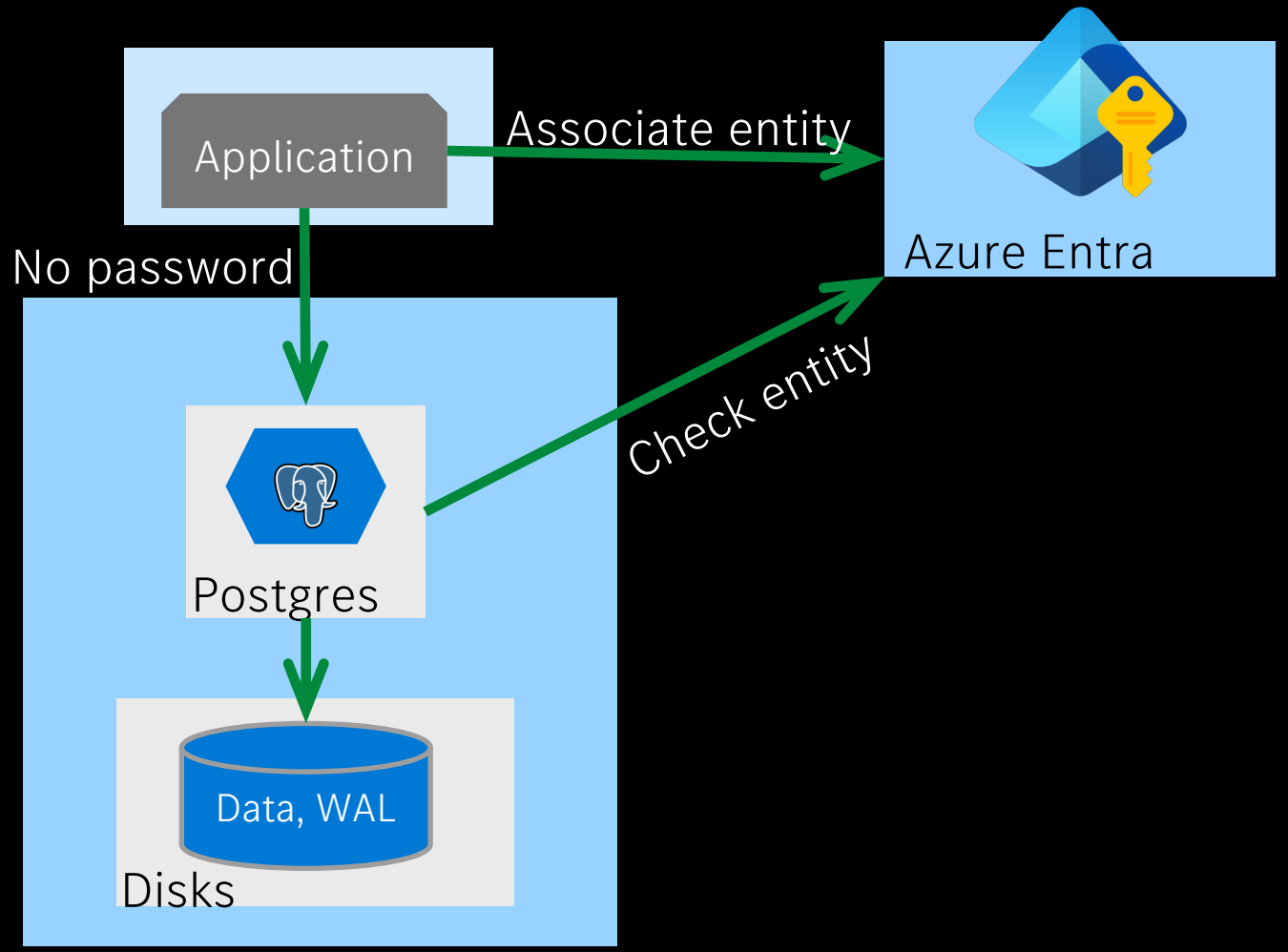


Azure Database for PostgreSQL Authentication



Azure Database for PostgreSQL

Authentication: Passwordless connections



Azure Database for PostgreSQL
Passwordless connections: Enablement

1. Provision managed identity
2. Configure AKS Cluster
3. Provision federated identity credential
4. Configure Azure PostgreSQL
5. Configure service in AKS
6. Configure Spring Boot application

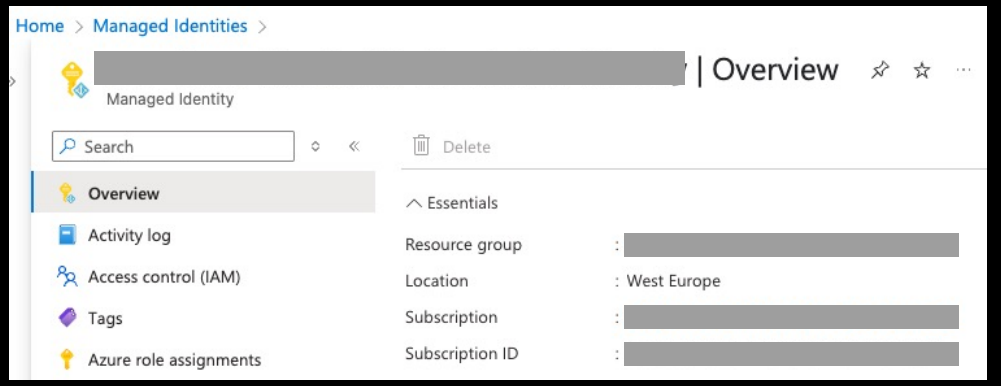
Other app platforms like Azure Functions or Azure App Service are also supported

Azure Database for PostgreSQL
Passwordless connections: Enablement

1. Provision managed identity

```
resource "azurerm_user_assigned_identity" ”example" {
 name          = "${var.app_stage}-${var.app_key}-identity"
 ...
```

2. Configure AKS Cluster
3. Provision federated identity credential
4. Configure Azure PostgreSQL
5. Configure service in AKS
6. Configure Spring Boot application



Azure Database for PostgreSQL
Passwordless connections: Enablement

1. Provision managed identity
2. Configure AKS Cluster

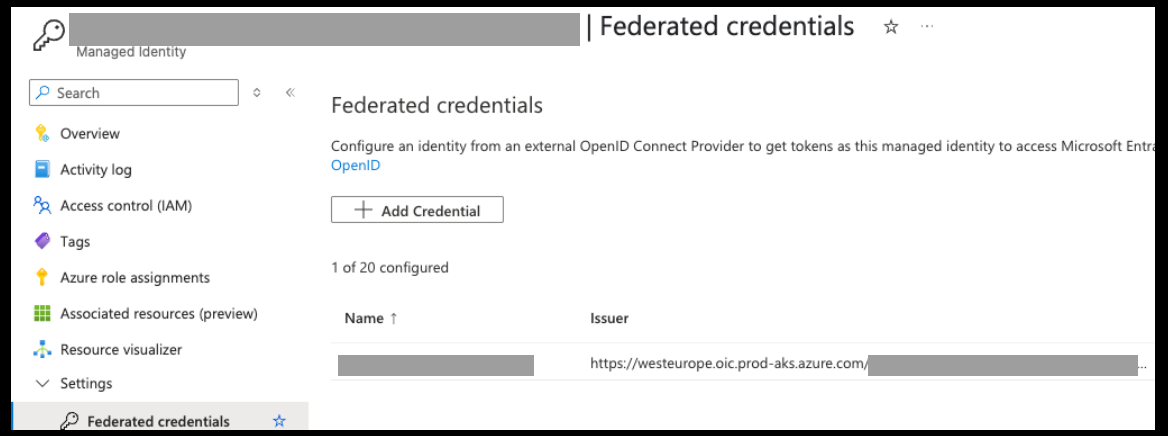
```
resource "azurerm_kubernetes_cluster" ”example” {
  oidc_issuer_enabled             = true
  workload_identity_enabled   = true
  ...
```

OpenID Connect provider that issues tokens

3. Provision federated identity credential
4. Configure Azure PostgreSQL
5. Configure service in AKS
6. Configure Spring Boot application

Azure Database for PostgreSQL
Passwordless connections: Enablemennt

1. Provision managed identity
2. Configure AKS Cluster
3. Provision federated identity credential




```
resource "azurerm_federated_identity_credential" ”federation” {
  parent_id          = azurerm_user_assigned_identity.example.id
  issuer             = azurerm_kubernetes_cluster.example.oidc_issuer_url
  audience           = [“api://AzureADTokenExchange”]
  subject            = “system:serviceaccount:<aks_namespace>:<aks_serviceaccount>”
  …
```

Associates managed identity to AKS account in this namespace

Azure Database for PostgreSQL
Passwordless connections: Enablement

4. Configure Azure PostgreSQL

```
resource "azurerm_postgresql_flexible_server" ”server” {
  authentication {
    active_directory_auth_enabled = true
    password_auth_enabled         = false
    tenant_id                     = data.azurerm_client_config.app.tenant_id
  }
  ...
```

```
resource "azurerm_postgresql_flexible_server_active_directory_administrator” ”a” {
    server_name         = azurerm_postgresql_flexible_server.server.name
    object_id           = var.entry_identity_client_id
    principal_name      = var.entry_identity_client_name
    principal_type      = "ServicePrincipal"
  ...
```

Azure Database for PostgreSQL
Passwordless connections: Enablement

5. Configure service in AKS
   - Configure AKS service account

```
labels:
  azure.workload.identity/use:              "true"
annotations:
  azure.workload.identity/client-id:    "${USER_ASSIGNED_IDENTITY_CLIENT_ID}"
```

   - Configure pod spec

```
labels:
  azure.workload.identity/use:              "true"
```

Azure Database for PostgreSQL
Passwordless connections: Enablement

6. Configure Spring Boot application
   - In Java add a dependency to Spring Cloud Azure Starter JDBC PostgreSQL
   - Configure Spring data source in application properties

```
spring.datasource.username=${USER_ASSIGNED_IDENTITY_NAME}
spring.datasource.azure.passwordless-enabled=true
```

Azure Database for PostgreSQL
Passwordless connections

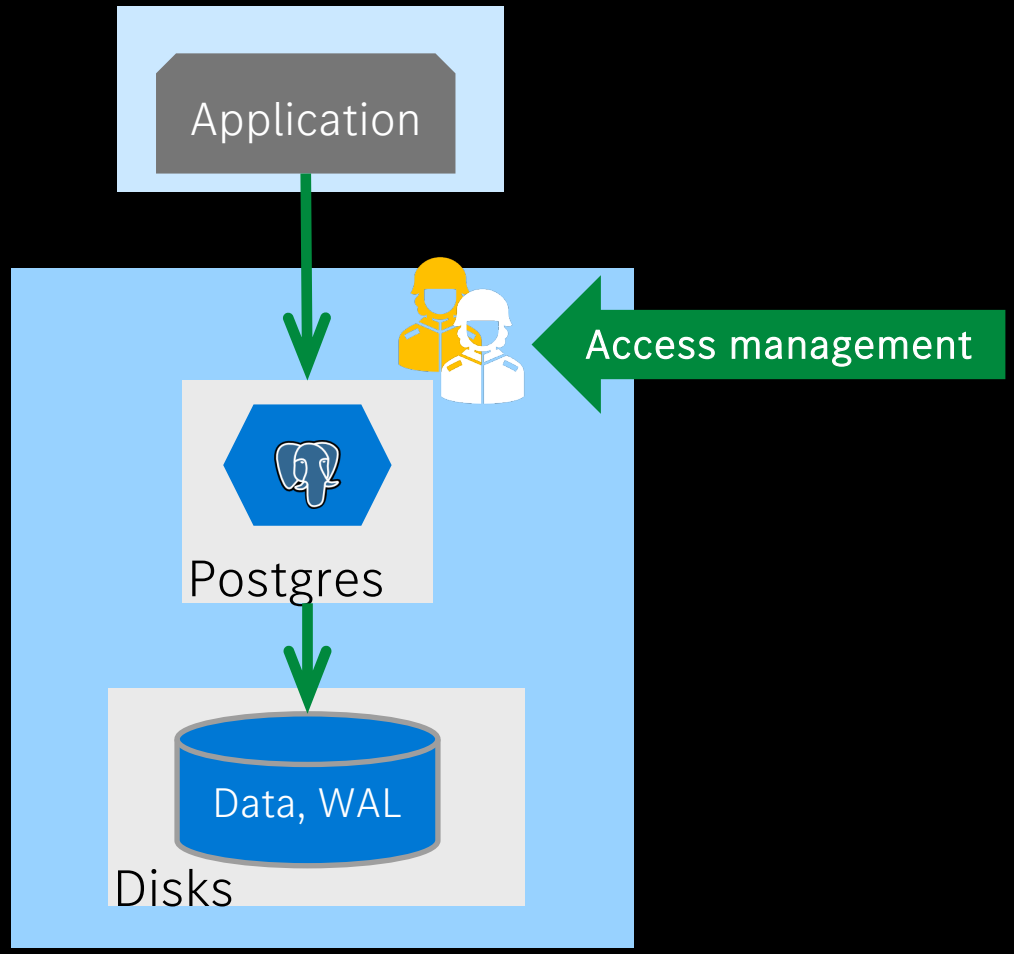
Does it work?

Deployment log:

```
[main] com.zaxxer.hikari.HikariDataSource.getConnection - HikariPool-1 - Starting...
[azure-sdk-global-thread-0] c.a.i.ManagedIdentityCredential.performLogging - Azure Identity => Managed Identity environment: AZURE AKS TOKEN EXCHANGE
```

Azure Database for PostgreSQL
Access Management: Authorization

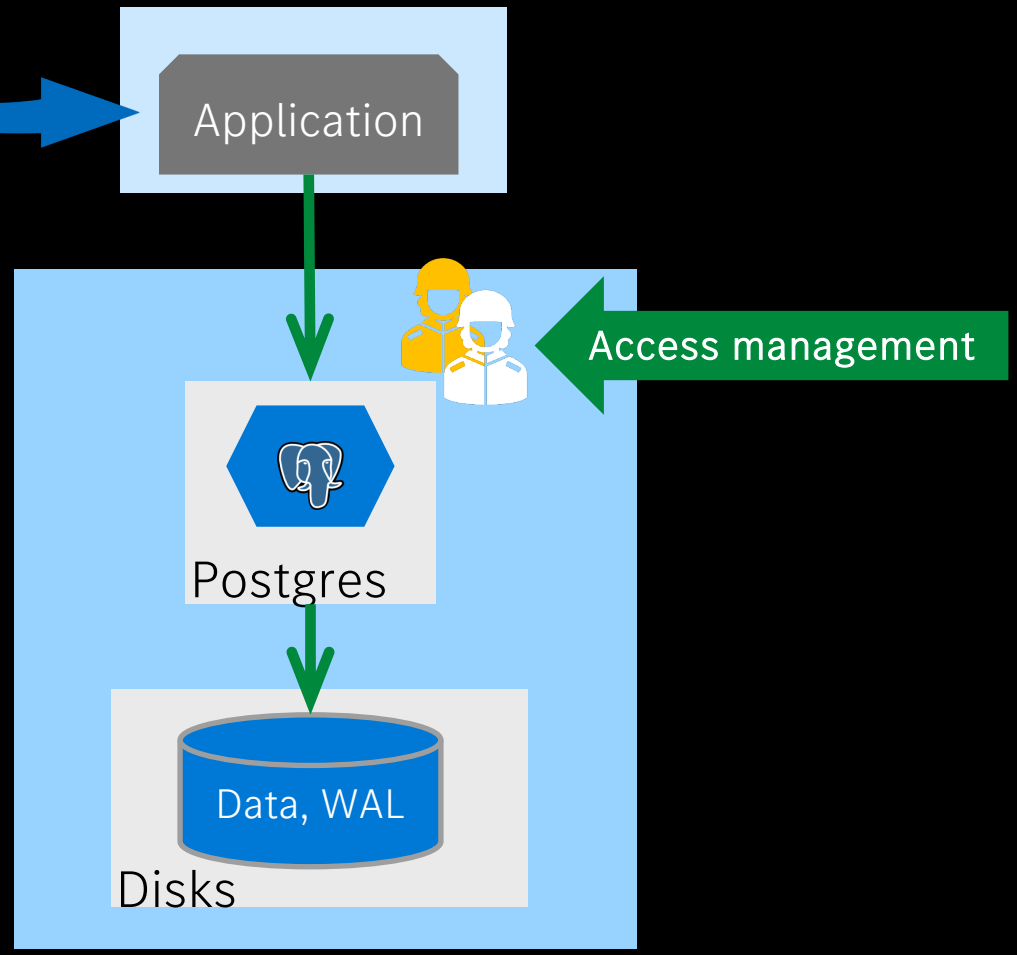
Authorization: check permission for actions



Azure Database for PostgreSQL

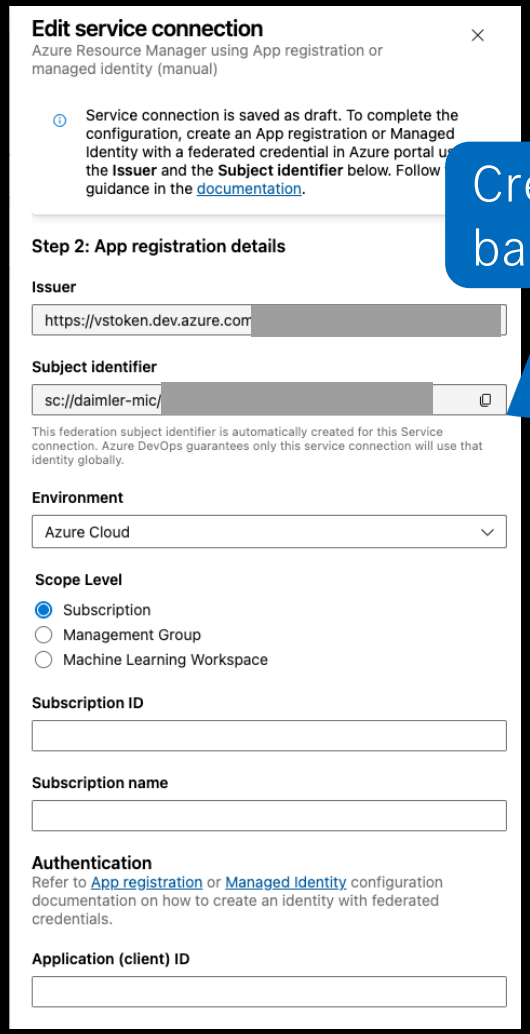
Access Management: Principle of Least Privilege

We want the application to have only minimum permissions in PostgreSQL



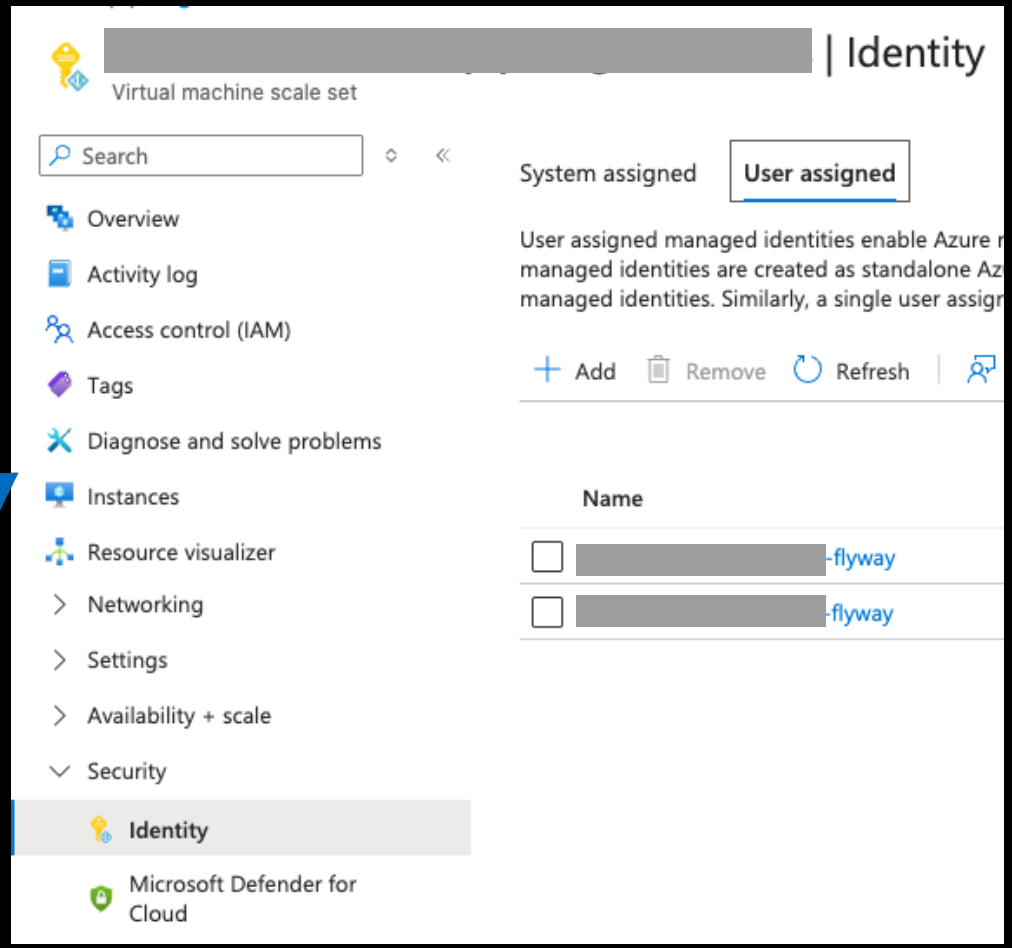
Azure Database for PostgreSQL

Access Management: Associate managed identity with CI/CD pipeline



Create service connection based on managed identity

Associate managed identity with VM scale set



Azure Database for PostgreSQL

Access Management: Assume identity in CI/CD pipeline

Example: Run Flyway script

```
az login --identity --client-id "${POSTGRESQL_ENTRA_ID}” --allow-no-subscriptions

ACCESS_TOKEN=$(az account get-access-token \
                    --resource https://ossrdbms-aad.database.windows.net \
                    --query accessToken --output tsv)

FLYWAY_PARAMS="-url=${DATASOURCE_URL} \
                    -user=${POSTGRESQL_ENTRA_NAME}   -password=${ACCESS_TOKEN} \
                    -configFiles=${CONFIG_FILE_NAME}”

flyway -connectRetries=3 -cleanDisabled="false" clean ${FLYWAY_PARAMS}
```

Azure Database for PostgreSQL

Access Management: Create role in Postgres to be used by application

Create role in Postgres:

```
export PGPASSWORD=$ACCESS_TOKEN

psql "host=${PGHOST} port=5432 dbname=postgres \
      user=${POSTGRESQL_ENTRA_NAME}  sslmode=require" \
  -c "select * from \
        pg_catalog.pgaadauth_create_principal_with_oid('${POSTGRESQL_ENTRA_SERVICE_NAME}', \
              '<object_id_of_managed_identity>', 'service', false, false)"
```

Execute in „postgres“ database

Role to be created by function

Create this managed identity upfront

Regular user or admin user?

FYI – to drop a role use pg_catalog.pgaadauth_drop_principal_if_exists

Azure Database for PostgreSQL

Access Management: Create role in Postgres to be used by application

Grant permissions to role:

Database of application

```
QUOTE="\""

psql "host=${PGHOST} port=5432 dbname=${PGDATABASE} \
      user=${POSTGRESQL_ENTRA_NAME} sslmode=require" \
  -c "GRANT USAGE ON SCHEMA public TO \
            ${QUOTE}${POSTGRESQL_ENTRA_SERVICE_NAME}${QUOTE};\
      GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO \
            ${QUOTE}${POSTGRESQL_ENTRA_SERVICE_NAME}${QUOTE};"
```

For example, grant permissions to process data in tables – but not to perform DDL

Azure Database for PostgreSQL

Access Management: Alternative approach

Enable Entra auth for existing Postgres role:

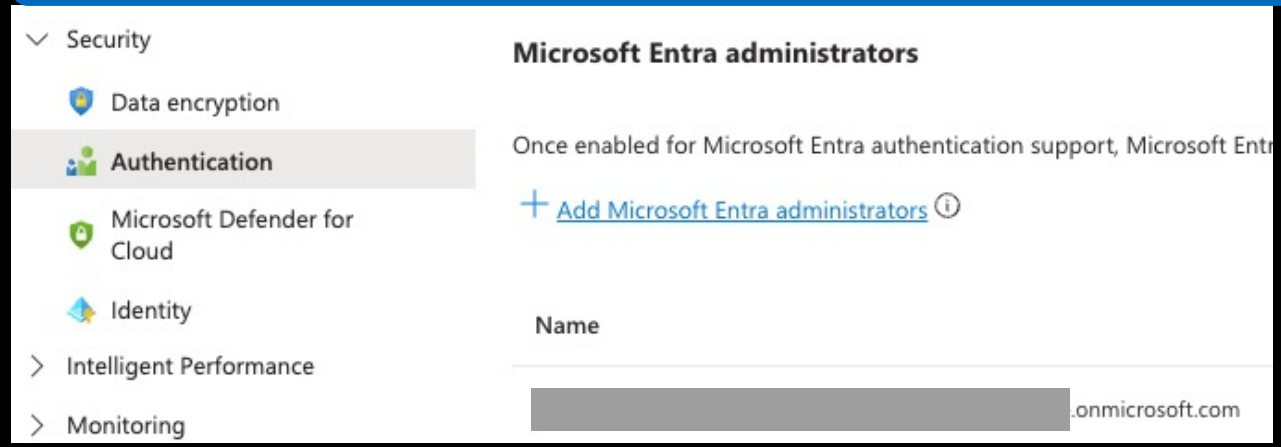
```
psql "host=${PGHOST} port=5432 dbname= postgres \
      user=${POSTGRESQL_ENTRA_NAME} sslmode=require" \
  -c ”SECURITY LABEL for ${QUOTE}pgaadauth${QUOTE} on role \
      ${QUOTE} ${POSTGRESQL_ENTRA_SERVICE_NAME}${QUOTE} is \
      'aadauth,oid= <object_id_of_managed_identity>,type=service,admin';"
```

Azure Database for PostgreSQL
Access Management: Operational aspects

To authenticate in DBA tool using your personal Entra ID:

```
az login
az account get-access-token --resource-type oss-rdbms --output tsv --query accessToken
```

To login, use your Entra name exactly as shown and the token as password:



Azure Database for PostgreSQL
Access Management: Operational aspects

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO ”managed_identity";
```

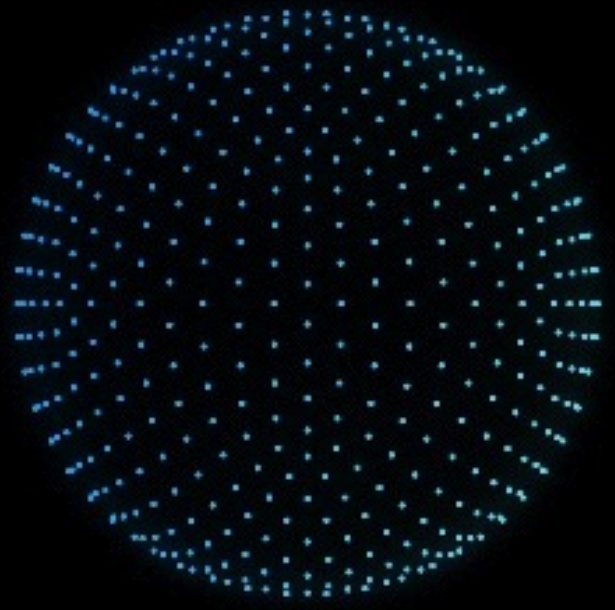
- Make sure to run GRANT statements as user or managed identity with suitable permissions for relevant objects
- GRANT statement may appear to have assigned permissions even though no permissions were assigned
- If in doubt, check `information_schema.role_table_grants`

Summary

source: media.mercedes-benz.com

- Private endpoints
  - Provide new networking opportunities with Azure PostgreSQL
  - **Additional costs**
  - **More flexible, more dynamic**
- Passwordless connections
  - Once it is set up, Azure takes good care
  - Passwords cannot be leaked anymore
  - Ultimate solution to password management

Thank you



Johannes Schuetzner

Mercedes-Benz Research & Development

POSETTE – An Event for Postgres 2025

Background information

- Private endpoint: learn.microsoft.com/en-us/azure/private-link/private-endpoint-overview
- Managed identity: learn.microsoft.com/en-us/entra/identity/managed-identities-azure-resources/overview
- Microsoft Entra ID for authentication with Azure Database for PostgreSQL: learn.microsoft.com/en-us/azure/postgresql/flexible-server/how-to-configure-sign-in-azure-ad-authentication
- Access a database without passwords in Spring: learn.microsoft.com/en-us/azure/developer/java/spring-framework/deploy-passwordless-spring-database-app?tabs=postgresql
- Migrate an application to use passwordless connections with Azure Database for PostgreSQL: learn.microsoft.com/en-us/azure/developer/java/spring-framework/migrate-postgresql-to-passwordless-connection