

Private Preview: Migrate from Azure PostgreSQL – Single Server to Flexible Server

Contents

- Introduction..... 1**
- Pre-requisites 2**
 - Azure CLI setup..... 2**
 - Target server creation 2**
 - Source server pre-requisites 2**
 - AAD App setup 3**
- CLI Commands 7**
 - Create Migration 8**
 - List Migrations 13**
 - Show Details 13**
 - Update migration 15**
 - Delete migration..... 16**
- End-to-end flow with examples..... 16**
 - Source and Target servers – Public Access..... 16*
 - Source and Target servers – Private Access in the same Vnet 23*
 - Source and Target servers – Private Access in the different Vnets 28*
 - Public Source with Virtual Network Service EndPoints and Private target 34*
- Current Limitations 37**
- Downloads..... 37**
- Appendix..... 37**

Introduction

This document contains details of an automated solution to migrate schema and data from an Azure Database for PostgreSQL – Single Server instance to Azure Database for PostgreSQL – Flexible Server with minimal downtime to the application. It requires you to create a target flexible server instance and to take care of few pre-requisites before you try a migration. All the details pertaining to pre-requisites are covered in the later sections of this documents. This solution migrates only schema and data. Other server components such as server parameters, connection security details, users and roles, tags must be manually copied to the target flexible server.

How does it work?

The migration service is a hosted solution where we deploy a VM on Azure and automatically setup the all the infrastructure needed for doing an online migration. The solution leverages [Azure database migration service](#) (DMS) which internally uses the logical replication of the PostgreSQL engine to support online migration.

It automates all the steps needed to perform an online migration such as setting up of DMS, creating the database in the target server, migrating schema, handling of foreign keys and triggers, adding firewall rules at both source and target to allow DMS access them etc, thus making it super easy.

How can it be consumed?

The migration service is currently exposed through easy-to-use Azure CLI commands. You can create migrations, list migrations, display migration details, modify state of the migration, and delete the migration using these CLI commands. The details of these CLI commands are covered in detail in later section of this document.

Given that the solution uses DMS underneath, it is [free of cost for the first six months of usage](#).

Pre-requisites

Azure CLI setup

- Firstly, install the latest Azure CLI for your corresponding operating system from the [Azure CLI install page](#).
- In case Azure CLI is already installed, check the version by issuing **az version** command. The version should be at least **2.28.0 or above** to use the private preview version of the CLI commands. If not, please update your Azure CLI using the following [link](#).
- Once installed, run the **az login** command. This will open the default browser and load an Azure sign-in page to authenticate. Pass in your azure credentials to do a successful authentication. For other ways to sign with Azure CLI, visit this [link](#).

Target server creation

You need to create the target flexible PostgreSQL server prior to commencing the migration. Use the [creation QuickStart guide](#) to create one. The instance SKU and storage allocated should match the actual source server.

Source server pre-requisites

This automated migration solution uses Azure DMS to perform an online migration from the source to target. As a result, you must enable logical replication pre-requisites in the source DB server. Enabling logical replication will require a server reboot for the change to take effect. You have a couple of options to enable logical replication in the source.

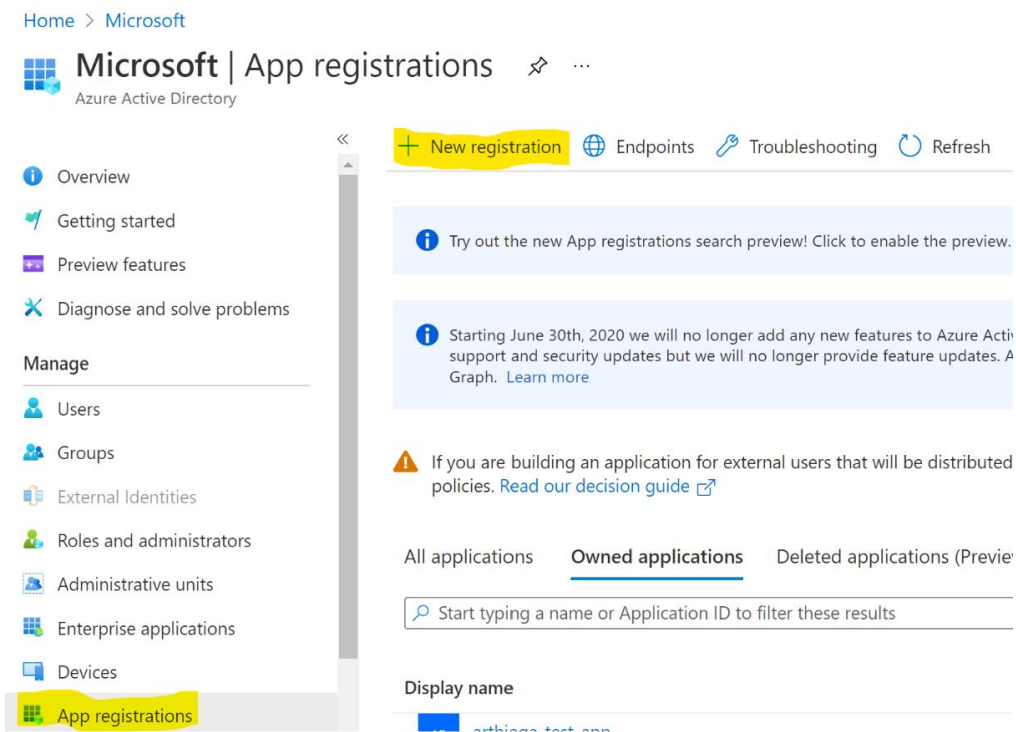
- a. You can [enable it yourself](#) and reboot the source server when possible.
- b. You can have this automated solution enable logical replication via Azure CLI command in the source server. We will cover this aspect in the set of CLI commands explained below.

AAD App setup

One of the most important components of this automated solution is the creation of [Azure Active Directory app \(AAD App\)](#) which helps in role-based access control. This automation service needs access to both the source and target servers. Access to these resources is restricted by the roles assigned to the AAD App.

To get started, create a new AAD Enterprise App by doing the following.


1. Search for Azure Active Directory in the search bar on the top in the portal.
2. Within the Azure Active Directory portal, under manage on the left, choose App Registrations.
3. Click on new registration.



4. Give the app registration a name, choose "Accounts in any organizational directory" and click register.

[Home](#) > [Microsoft](#) >

Register an application ...

 If you are building an application for external users that will be distributed by Microsoft, you must register as a first party application to meet all security, privacy, and compliance policies. [Read our decision guide](#)

* Name

The user-facing display name for this application (this can be changed later).

sample-aad-app 

Supported account types


Who can use this application or access this API?

- ☐ Accounts in this organizational directory only (Microsoft only - Single tenant)
- ☒ Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- ☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- ☐ Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

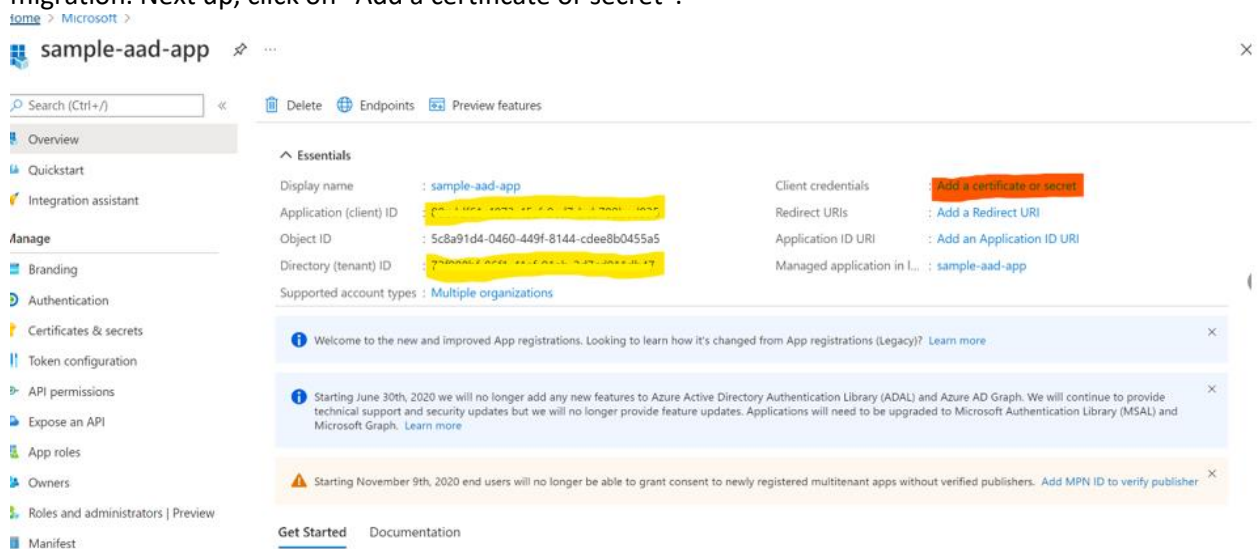
We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Web  e.g. https://example.com/auth

By proceeding, you agree to the [Microsoft Platform Policies](#)

[Register](#)

- Once the app is created, you can copy the client ID and tenant ID required for later steps in the migration. Next up, click on “Add a certificate or secret”.



- In the next screen, click on “New client secret”.

Home > Microsoft > sample-aad-app

sample-aad-app | Certificates & secrets

Search (Ctrl+/) « Got feedback?

- Overview
- Quickstart
- Integration assistant
- Manage
 - Branding
 - Authentication
 - Certificates & secrets**
 - Token configuration
 - API permissions
 - Expose an API
 - App roles
 - Owners
 - Roles and administrators | Preview
 - Manifest
- Support + Troubleshooting
 - Troubleshooting
 - New support request

Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

Certificates

Certificates can be used as secrets to prove the application's identity when requesting a token. Also can be referred to as public keys.

Upload certificate

Thumbprint	Start date	Expires	Certificate ID
No certificates have been added for this application.			

Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

Description	Expires	Value	Secret ID
No client secrets have been created for this application.			

- In the fan-out blade that opens, add a name, and click add –
Add a client secret

Add a client secret

Description

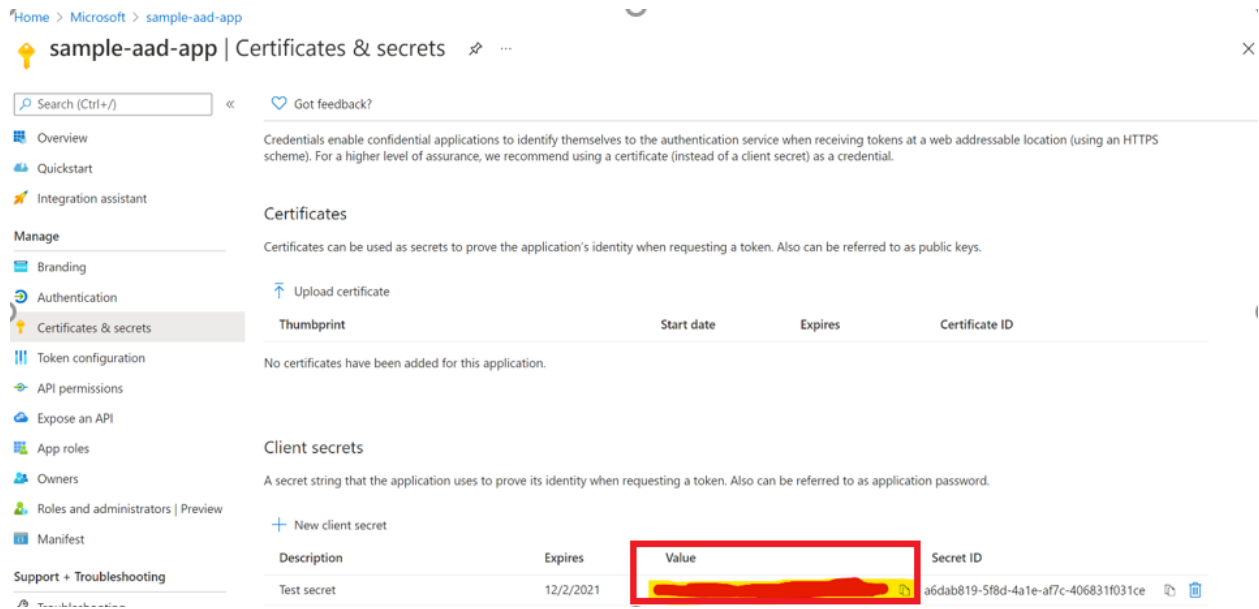
Test secret

Expires

Recommended: 6 months

Add
Cancel

- In the next screen, copy the **Value** column (highlighted in the below pic) which has the details of the AAD App secret. This can be copied only while creation. If you miss copying this secret, you will need to delete this secret and create another one for future tries.

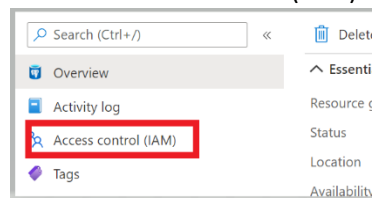


At the end of the setup, you should have an object ID, a tenant ID, and a secret to use for the migration. In the next step you will provide access to appropriate resources to automate the migration.

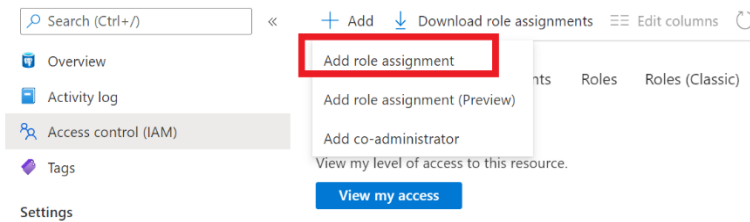
9. Once AAD App is created, you will need to add contributor privileges for this AAD app to the following resources:
 - a. **REQUIRED:** Source single server you are migrating from.
 - b. **REQUIRED:** Target flexible server you are migrating into.
 - c. **REQUIRED:** Resource group for the migration (By default this is the target flexible server resource group). (Or) If you are using a temporary resource group to create the migration infrastructure (we cover this later in the document), the AAD App will require contributor privileges to this resource group as well.
 - d. **OPTIONAL:**
 - i. If the source or the target happens to be inside a VNet then the AAD App will require contributor privileges to corresponding VNet.
 - ii. If the source and the target happen to be in different VNets, then the AAD app will require contributor privileges to both the source and target VNets.

Let's look at how to add contributor privileges to an Azure resource. For the target flexible server, do the following:

1. Select the target flexible server in the Azure portal.
2. Click on Access Control (IAM) on the top left.

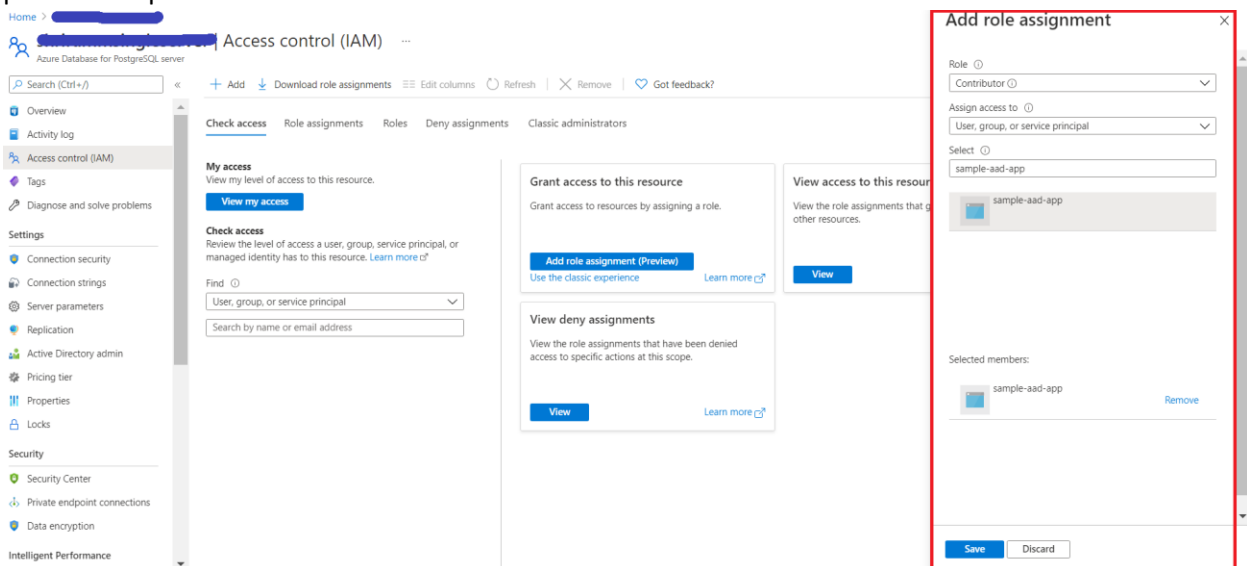


3. Click add and choose "Add role assignment".



Note: The Add role assignment capability is only enabled for users in the subscription with role type as “Owners”. Users with other roles do not have permission to add role assignments.

4. In the role assignment blade on the right, set role to “Contributor”, set Assign access to the default (User, group, or service principal), click select field and search for the AAD App we created in the previous step.



5. Click save.

The AAD App now has contributor privileges to the target flexible server instance. Repeat process for all resources given above.

Once all these pre-requisites are taken care of, you are now ready to start the migration process.

CLI Commands

The private preview comes with a list of easy-to-use CLI commands to perform migration related tasks. All the CLI commands starts with “**az postgres flexible-server migration**”. There are also **help** statements provided to assist you in understanding the various options and in framing the right syntax for the CLI commands.

az postgres flexible-server migration --help gives you the following output.

```

C:\Users\Administrator\Documents\single to flex migrations\docs>az postgres flexible-server migration --help

Group
  az postgres flexible-server migration : Manage migration workflows for PostgreSQL Flexible Servers.
  Command group 'postgres flexible-server' is in preview and under development. Reference and support levels: https://aka.ms/CLI_refstatus
  This command group is experimental and under development. Reference and support levels: https://aka.ms/CLI_refstatus

Commands:
  create : Create a new migration workflow for a flexible server.
  delete : Delete a specific migration.
  list   : List the migrations of a flexible server.
  show   : Get the details of a specific migration.
  update : Update a specific migration.

For more specific examples, use: az find "az postgres flexible-server migration"

Please let us know how we are doing: https://aka.ms/azureclihats

```

It clearly lists out the name and the corresponding verbs that are supported in private preview. Let us take a deep dive of these commands.

Create Migration

The create migration command helps in creating a migration workflow from the source to target using the database migration service. The CLI command to create a migration is given below

```

az postgres flexible-server migration create --subscription <sub_id> --resource-group <rg_name> --name <target_server_name> --migration-name <name_of_migration> --properties "<absolute_path_of_jsonfile ">

```

For e.g:

```

az postgres flexible-server migration create --subscription 5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30 --resource-group Shriramm-learning-rg --name flexserver12 --migration-name migration1 --properties "C:\Users\Administrator\Documents\migrationBody.json"

```

The parameters highlighted in yellow need to be filled in with appropriate values as given in the above example. You can assign a name to the migration workflow by passing it as a value to the **migration-name** argument. The **migration-name** argument is unique to the target server. No two migrations for a target server can have the same **migration-name**. The migration-name argument is also used in other CLI commands such as **update**, **delete**, **show details** as an identifier to the migration workflow to perform the corresponding actions.

Here is a snapshot of the response for the **create migration** command with the migration-name parameter highlighted


```
C:\Users\Administrator>az postgres flexible-server migration create --subscription 5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30
--name migration1 --properties @"C:\Users\Administrator\Documents\single to flex migrations\docs\migrationBodyDemo.json"
Command group 'postgres flexible-server' is in preview and under development. Reference and support levels: https://aka.ms/azcli-preview
Command group 'postgres flexible-server migration' is experimental and under development. Reference and support levels: https://aka.ms/azcli-experimental
{
  "id": "/subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/migrations/migration1",
  "location": "East US",
  "name": "migration1",
  "properties": {
    "currentStatus": {
      "state": "InProgress"
    },
    "dbsToMigrate": [
      "db1",
      "db2",
      "db3"
    ],
    "migrationDetailsLevel": "Default",
    "migrationId": "fe85fc1c-1fb2-4f65-9e9c-1f971de29e89",
    "migrationName": "migration1",
    "migrationWindowStartTimeInUtc": "2021-09-16T09:52:35.0170276Z",
    "overwriteDBsInTarget": true,
    "sourceDBServerResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/instance1",
    "targetDBServerResourceId": "/subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/instance2"
  },
  "resourceGroup": "Shriramm-learning-rg",
  "type": "Microsoft.DBforPostgreSQL/flexibleServers/migrations"
}
```

Also, interesting to note is that Azure CLI can configure default options for resource groups and subscriptions. This is beneficial in cases where you try multiple migration attempts between resources in the same subscription or if you want to use a default resource group to place the resources created by the automation service.

- Configure the **default subscription** by using the following [link](#).
- Configure the **default resource group** by using the following [link](#).

The following table depicts the easiness to prepare and invoke CLI commands with the help of default configurations.

Without default configuration	With default configuration
az postgres flexible-server migration create --subscription <sub_id> --resource-group <rg_name> --name <target_server_name> --migration-name <migration_name> --properties "<absolute_path_of_jsonfile>"	az postgres flexible-server migration create --name <target_server_name> --migration-name <migration_name> --properties "<absolute_path_of_jsonfile>"

With the default configuration set, the user need not specify the parameters which has been configured with default options again in the CLI command and thus make the commands shorter and easier to use.

Also note that there is a help command associated with each of the CLI commands.

az postgres flexible-server migration create -- help gives the following output

```

Command
az postgres flexible-server migration create : Create a new migration workflow for a flexible
server.
    Command group 'postgres flexible-server' is in preview and under development. Reference
    and support levels: https://aka.ms/CLI\_refstatus
    Command group 'postgres flexible-server migration' is experimental and under
    development. Reference and support levels: https://aka.ms/CLI\_refstatus

Arguments
--name -n [Required] : Migration target server name.
--properties -b [Required] : Request properties. Use @{} to load from a file. For quoting
issues in different terminals, see https://github.com/Azure/azure-
cli/blob/dev/doc/use\_cli\_effectively.md#quoting-issues.
--migration-name : Name of the migration.
--resource-group -g : Resource Group Name of the migration target server. Config:
Shriramm-learning-rg.

Global Arguments
--debug : Increase logging verbosity to show all debug logs.
--help -h : Show this help message and exit.
--only-show-errors : Only show errors, suppressing warnings.
--output -o : Output format. Allowed values: json, jsonc, none, table, tsv,
yaml, yamlc. Default: json.
--query : JMESPath query string. See http://jmespath.org/ for more
information and examples.
--subscription : Name or ID of subscription. You can configure the default
subscription using 'az account set -s NAME_OR_ID'.
--verbose : Increase logging verbosity. Use --debug for full debug logs.

Examples
Start a migration workflow on the target server identified by the parameters. The configurations
of the migration should be specified in the migrationConfig.json file.
az postgres flexible-server migration create --subscription xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx --resource-group testGroup --name testServer --properties
@migrationConfig.json

For more specific examples, use: az find "az postgres flexible-server migration create"

Please let us know how we are doing: https://aka.ms/azurecliats

```

It clearly calls out the expected arguments and has an example syntax that needs to be used to create a successful migration attempt from the source to target.

The **create migration** command needs a json file to be passed on as part of its **properties** argument. Below is a sample of **migrationBody.json**. Note that the parameters marked in yellow are optional.

Contents of migrationBody.json:

```
{
  "properties": {

    "SourceDBServerResourceId": "subscriptions/<subscriptionid>/resourceGroups/<src_rg_name>/providers/Microsoft.DBforPostgreSQL/servers/<source server name>",

    "SourceDBServerFullyQualifiedDomainName": "fqdn of the source server as per the custom DNS server",
    "TargetDBServerFullyQualifiedDomainName": "fqdn of the target server as per the custom DNS server"

    "SecretParameters": {
      "AdminCredentials": {
        "SourceServerPassword": "<password>",
        "TargetServerPassword": "<password>"
      },
      "AADApp": {
        "ClientId": "<client id>",
        "TenantId": "<tenant id>",
        "AadSecret": "<secret>"
      }
    },

    "MigrationResourceGroup": {
      "ResourceId": "subscriptions/<subscriptionid>/resourceGroups/<temp_rg_name>",
      "SubnetResourceId": "/subscriptions/<subscriptionid>/resourceGroups/<rg_name>/providers/Microsoft.Network/virtualNetworks/<Vnet_name>/subnets/<subnet_name>"
    },

    "DBsToMigrate": [
      "<db1>", "<db2>"
    ],

    "SetupLogicalReplicationOnSourceDBIfNeeded": "true",

    "OverwriteDBsInTarget": "true"

  }
}
```

Let us take a deep dive of the various parameters in the json file.

- **SourceDBServerResourceID** – This is the property for the resource ID of the single server and is mandatory
- **SourceDBServerFullyQualifiedDomainName** – This is an optional property and should only be used when a custom DNS server is used for name resolution inside a VNet. The FQDN of the single server as per the custom DNS server should be provided for this property.

How to figure out if a custom DNS server is being used for the name resolution?
 Navigate to your Vnet in the portal and click on DNS server. It should indicate if it is using a custom DNS server or default Azure provided DNS server.



- **TargetDBServerFullyQualifiedDomainName** - This is an optional property and should only be used when a custom DNS server is used for name resolution inside a VNet. The FQDN of the flexible server as per the custom DNS server should be provided for this property.
- **SecretParameters** – All the fields under the SecretParameters sections are mandatory. They help to authenticate against the source and target servers and help in checking proper authorization access to the resources.
- **MigrationResourceGroup** – This section consists of two properties
 - **ResourceID**- The automation service creates DMS and other network infrastructure components on the fly to move data from the source to target. By default, all the components created by this service will be under the resource group of the target server. If a customer wishes to put them under a different resource group, they can assign the ID of that resource group to this property.
 - **SubnetResourceID** – In case when a customer wants to create the DMS inside a particular VNet, then this property can be used to specify the subnet under the VNet in which the DMS should be created. One common scenario for deploying DMS under a Vnet would be when the flexible server is in private access mode.

- **SetupLogicalReplicationOnSourceDBIfNeeded** – Logical replication pre-requisite can be enabled on the source server automatically by setting this property to **true**.
- **OverwriteDBsinTarget** - If the target database is not empty, the migration will not begin until the user acknowledges that overwrites in the target DBs are allowed. This can be done by setting the value of this property to **true**.

Note that **SetupLogicalReplicationOnSourceDBIfNeeded**, **OverwriteDBsinTarget** are optional properties

You can download the json files using the following link.

- [Public Access \(both source and target\)](#)
- [Private Access \(either source using private end point or target is inside a VNet\)](#)

List Migrations

This command lists down the migration attempts that were done to a particular target flexible server. The CLI command to list migrations is given below

```
az postgres flexible-server migration list --subscription <sub_id> --resource-group <rg_name> --name <target_server_name> --filter Active
```

Please note that you can remove subscription and resource group parameters from the above command if you have already set the default configuration for the same. This applies to all the following CLI commands as well.

There is a parameter called **filter** and it can take **Active** and **All** as values.

- **Active** – Lists down the current active migration attempts for the target server. It does not include the migrations that have failed/cancelled/succeeded before.
- **All** – Lists down all the migration attempts to the target server which includes both the active and past migrations irrespective of the state.

Use the **az postgres flexible-server migration list -- help** for any additional information.

Show Details

This command gets the details of a specific migration. This includes information on the current state and substate of the migration workflow. The CLI command to show the details of a migration is given below:

```
az postgres flexible-server migration show --subscription <sub_id> --resource-group <rg_name> --name <target_server_name> --migration-name <migration_name>
```

The **migration_name** is the name assigned to the migration workflow during the **create migration** command.

Here is a snapshot of the sample response from the **Show Details** CLI command.

```

C:\Users\Administrator>az postgres flexible-server migration show --name flexserver12 --migration-name mymigration1
Command group 'postgres flexible-server' is in preview and under development. Reference and support levels: https://aka.ms/cli_refstatus
Command group 'postgres flexible-server migration' is experimental and under development. Reference and support levels: https://aka.ms/cli_refstatus
{
  "id": "/subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/ShriRam-Learning-RG/providers/Microsoft.DBforPostgreSQL/flexibleServers/flexserver12/migrations/mymigration1",
  "name": "mymigration1",
  "properties": {
    "currentStatus": {
      "currentSubStateDetails": {
        "currentSubState": "WaitingForCutoverTrigger",
        "dbDetails": {
          "db3": {
            "appliedChanges": 0,
            "cdcDeleteCounter": 0,
            "cdcInsertCounter": 0,
            "cdcUpdateCounter": 0,
            "databaseName": "db3",
            "fullLoadCompletedTables": 1,
            "fullLoadErroredTables": 0,
            "fullLoadLoadingTables": 0,
            "fullLoadQueuedTables": 0,
            "incomingChanges": 0,
            "initializationCompleted": true,
            "latency": 0,
            "migrationState": "READY_TO_COMPLETE",
            "startedOn": "2021-09-16T11:49:58.9499824+00:00"
          },
          "db4": {
            "appliedChanges": 0,
            "cdcDeleteCounter": 0,
            "cdcInsertCounter": 0,
            "cdcUpdateCounter": 0,
            "databaseName": "db4",
            "fullLoadCompletedTables": 1,
            "fullLoadErroredTables": 0,
            "fullLoadLoadingTables": 0,
            "fullLoadQueuedTables": 0,
            "incomingChanges": 0,
            "initializationCompleted": true,
            "latency": 0,
            "migrationState": "READY_TO_COMPLETE",
            "startedOn": "2021-09-16T11:49:59.0799608+00:00"
          }
        }
      }
    }
  }
}

```

Some important points to note on the command response –

- c. Each DB being migrated has its own section with all migration details such as table count, success rate, number, and type of changes, etc.
- d. Possible migration states include –
 - i. **InProgress**: The infrastructure setup or the actual data migration is in progress.
 - ii. **WaitingForUserAction**: Automated flow is waiting on a user action. This state has a corresponding substate below that also starts with waiting to identify the appropriate user action.
 - iii. **Canceled**: The migration has been cancelled or deleted.
 - iv. **Failed**: The migration has failed.
 - v. **Succeeded**: The migration has succeeded and is complete.
- e. Possible migration substates include –
 - i. **PerformingPreRequisiteSteps**: Infrastructure is being set up and is being prepped for data migration.
 - ii. **MigratingData**: Data is being migrated.
 - iii. **CompletingMigration**: Migration cutover in progress.
 - iv. **WaitingForLogicalReplicationSetupRequestOnSourceDB**: Waiting for logical replication enablement. You can manually enable this or enable via the **update migration** CLI command covered in the next section.
 - v. **WaitingForDBsToMigrateSpecification**: Waiting to know information on DBs to migrate. You can add DBs to migrate via the **update migration** CLI command covered in the next section.
 - vi. **WaitingForTargetDBOverwriteConfirmation**: Waiting for confirmation on target overwrite as data is present in the target database being migrated into. You can enable this via the **update migration** CLI command covered in the next section.

- vii. **WaitingForCutoverTrigger**: Migration is ready for cutover. You can start the cutover when ready.
- viii. **Completed**: Cutover was successful, and migration is complete.

Use the **az postgres flexible-server migration show -- help** for any additional information.

Update migration

As soon as the infrastructure setup is complete, the migration activity will pause with appropriate messages seen in the **show details** CLI command response if some pre-requisites are missing. At this point, the migration goes into a state called “**WaitingForUserAction**” and it can have the following sub states

- **WaitingForLogicalReplicationSetupRequestOnSourceDB**
- **WaitingForDBsToMigrateSpecification**
- **WaitingForTargetDBOverwriteConfirmation**
- **WaitingForCutoverTrigger**

The **update migration** command is used to set values for parameters which helps the migration to move to the next stage in the migration process. Let us look at this in detail for each of the substates.

- a. **WaitingForLogicalReplicationSetupRequestOnSourceDB** - The migration is waiting for logical replication to be enabled at the source. This can be enabled by the following CLI command

```
az postgres flexible-server migration update --subscription <sub_id> --resource-group <rg_name> --name <target_server_name> --migration-name <migration_name> --initiate-data-migration true
```

The **initiate-data-migration** flag is set to true which will turn on the logical replication flag at the source. This requires a reboot of the server and might take some time to move to the next stage of the migration. A user can also enable this manually by going to the source server in the Azure portal and changing the Azure Replication Support flag to **Logical**. This would restart the server. In case you have enabled it manually, you would still **need to issue the above update command** for the migration to move out of the “**WaitingForUserAction**” state. The server does not do a reboot again since it was already done via the portal action.

- b. **WaitingForCutoverTrigger** – The migration cutover can be done when the target is in full sync with the source. The CLI command for the cutover is given below.

```
az postgres flexible-server migration update --subscription <sub_id> --resource-group <rg_name> --name <target_server_name> --migration-name <migration_name> --cutover
```

Note that it is recommended to run the migration on test workload and doing a dummy cutover of the application with flexible server. Once all required tests pass and SLAs are met with a high degree of confidence, rerun the migration on your production workload and schedule a cutover. You will also need to update your application database connection string to point to the new target flexible server as part of the cutover.

The details on the substates **WaitingForDBsToMigrateSpecification**, **WaitingForTargetDBOverwriteConfirmation** are documented in the [appendix](#).

Use the **az postgres flexible-server migration update --help** for any additional information.

Delete migration

Any migration and all its underlying infrastructure can be deleted at once using the **delete migration** command. Below is the CLI command to delete a migration attempt:

```
az postgres flexible-server migration delete --subscription <sub_id> --resource-group <rg_name> --name <target_server_name> --migration-name <migration_name>
```

Use the **az postgres flexible-server migration delete --help** for any additional information.

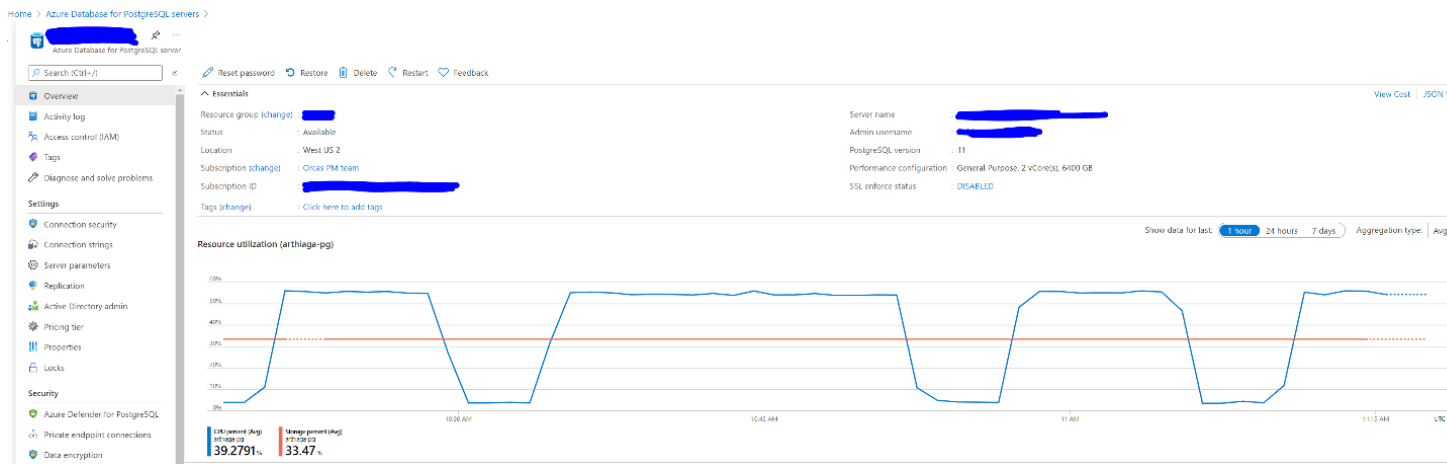
End-to-end flow with examples

Source and Target servers – Public Access

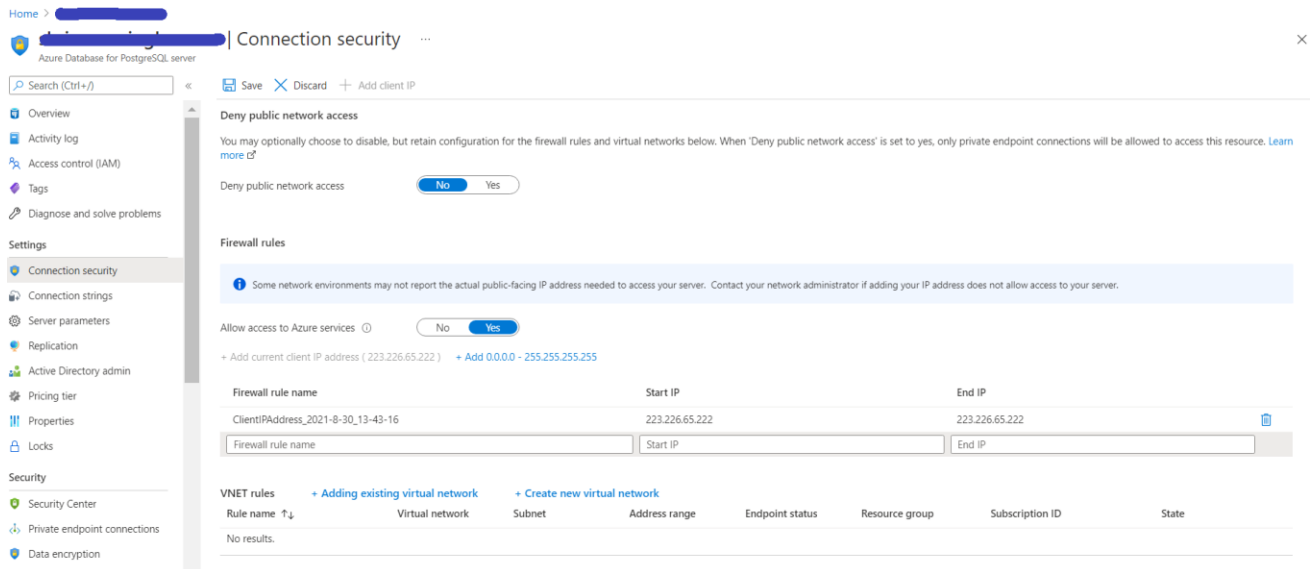
Download the [template for migrationBody.json for public access](#). There are placeholders that need to be filled with appropriate values pertaining to your subscription, resource groups, AAD privileges etc.

Let's assume that we need to migrate a sample **sportsdb_sample** database from a Postgres single server to a public access flexible server.

Below is the source server screenshot from the Azure Portal:



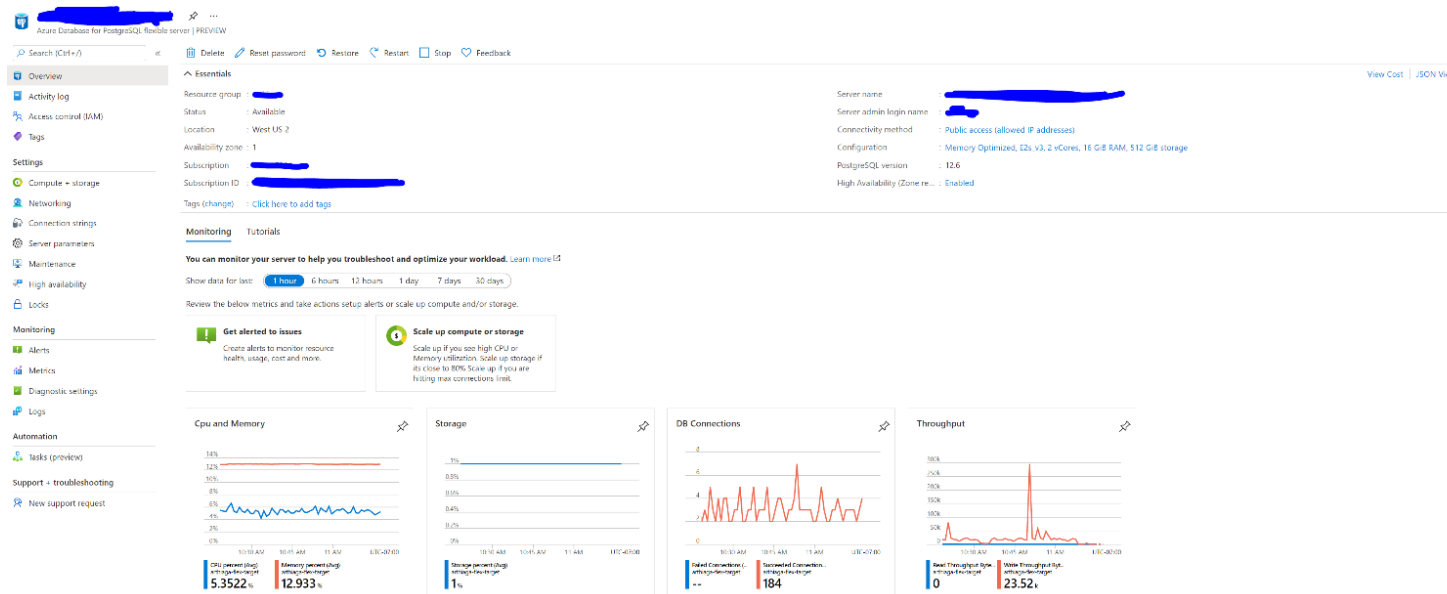
The Connection Security tab for the source server is shown below.



It allows public network access and clients IP addresses can be added to the list of firewall rules to access the source.

Step 1: Create target flexible server.

We used the [QuickStart guide](#) to create a corresponding PostgreSQL target flexible server. We kept the SKU same and given we are just migrating a small sample database; we are allocating 512 GB of storage. Below is the target server screenshot once created:



Please note that the target flexible server is also created with public access and is not under any virtual network.

Step 2: Setup pre-requisites.

1. The Azure CLI environment and all appropriate defaults are setup.

2. As stated, logical replication has been enabled in the source server.
3. Lastly, Active Directory App (AAD) credentials have been setup with appropriate permissions

Step 3: Create migration.

CREATE API CALL:

```
az postgres flexible-server migration create --name flexv12 --migration-name  
mymigration --properties "C:\Users\admin\Documents\migrationBodyPublicAccess.json"
```

Contents of migrationBody.json:

```
{  
  "properties": {  
    "SourceDBServerResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-  
learning-rg/providers/Microsoft.DBforPostgreSQL/servers/shrirammsingleserver",  
    "SecretParameters": {  
      "AdminCredentials": {  
        "SourceServerPassword": "SourceAdminPassword",  
        "TargetServerPassword": "TargetAdminPassword"  
      },  
      "AADApp": {  
        "ClientId": "ClientAAD",  
        "TenantId": "TenantAAD",  
        "AadSecret": "SecretAAD"  
      }  
    },  
    "DBsToMigrate": [  
      "sportsdb_sample"  
    ]  
  }  
}
```



Note: Portions highlighted are replaced for security reasons.
shrirammsingleserver = Source Single Server Postgres

API CALL RESPONSE:

```
{  
  "id": "/subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/flexv12/migrations/8f2cc81a-3875-469b-ba1f-341126b971f2",  
  "location": "East US",  
  "name": "mymigration",  
  "properties": {  
    "currentStatus": {  
      "state": "InProgress"  
    },  
    "dbsToMigrate": [  
      "sportsdb_sample"  
    ],  
    "migrationDetailsLevel": "Default",  
    "migrationId": "8f2cc81a-3875-469b-ba1f-341126b971f2",  
    "migrationName": "mymigration",  
    "migrationWindowStartTimeInUtc": "2021-08-13T13:53:15.3848516Z",  
    "overwriteDBsInTarget": true,  
    "sourceDBServerResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/servers/shrirammsingleserver",  
    "targetDBServerResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/flexv12"  
  },  
  "resourceGroup": "Shriramm-learning-rg",  
  "type": "Microsoft.DBforPostgreSQL/flexibleServers/migrations"  
}
```


This will automatically setup all infrastructure in the background and start the migration process. If the API call was successful, you should now see a DMS service being automatically created with the same name as the target flexible server instance.

Home >

Azure Database Migration Services  

Microsoft (microsoft.azuredatamigration.com)

Create

Manage views 

Refresh

Export to CSV


Open query


Assign tags


Feedback

Filter for any field...

Subscription == Orcas PM team

Resource group == all 


Location == West US 2 

Resource group == arthiaga 



Add filter


Showing 1 to 2 of 2 records.


☐


Name 


☐



 

Status 

Subscription 

Location 

Resource group 

arthiaga-flex-target

Deploying

Orcas PM team

West US 2

arthiaga

Step 3: Check migration status using migration-name.

CHECK MIGRATION STATUS API CALL:

```
az postgres flexible-server migration show --name flexv12 --migration-name mymigration
```

API CALL RESPONSE FROM FIRST API CALL:

```
{
  "id": "/subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/flexv12/migrations/8f2cc81a-3875-469b-ba1f-341126b971f2",
  "name": "mymigration",
  "properties": {
    "currentStatus": {
      "currentSubStateDetails": {
        "currentSubState": "WaitingForCutoverTrigger",
        "dbDetails": {
          "sportsdb_sample": {
            "appliedChanges": 0,
            "cdcDeleteCounter": 0,
            "cdcInsertCounter": 0,
            "cdcUpdateCounter": 0,
            "databaseName": "sportsdb_sample",
            "fullLoadCompletedTables": 1,
            "fullLoadErroredTables": 0,
            "fullLoadLoadingTables": 0,
            "fullLoadQueuedTables": 0,
            "incomingChanges": 0,
            "initializationCompleted": true,
            "latency": 0,
            "migrationState": "READY_TO_COMPLETE",
            "startedOn": "2021-08-13T14:02:40.3966618+00:00"
          }
        }
      },
      "state": "WaitingForUserAction"
    },
    "dbsToMigrate": [
      "sportsdb_sample"
    ],
    "migrationDetailsLevel": "Default",
    "migrationId": "8f2cc81a-3875-469b-ba1f-341126b971f2",
    "migrationName": "mymigration",
    "migrationWindowStartTimeInUtc": "2021-08-13T13:53:15.3848516Z",
    "overwriteDBsInTarget": true,
    "sourceDBServerMetadata": {
      "location": "eastus",
      "sku": {
        "name": "GP_Gen5_4",
        "tier": "GeneralPurpose"
      },
      "storageMB": 102400,
      "version": "11"
    },
    "sourceDBServerResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-rg/providers/Microsoft.DBforPostgreSQL/servers/shriramssingleserver",
    "targetDBServerMetadata": {
      "location": "East US",
      "sku": {
        "name": "Standard_B1ms",
        "tier": "Burstable"
      },
      "storageMB": 32768,

```

We can see that the automatic infrastructure setup completed, migrated schema and data, and we are now ready to cutover.

Step 4: Cutover.

CUTOVER API CALL:

```
az postgres flexible-server migration update --name flexv12 --migration-name  
mymigration --cutover
```

RESPONSE

```
{  
  "id": "/subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/flexv12/migrations/8f2cc81a-3875-469b-ba1f-341126b971f2",  
  "location": "East US",  
  "name": "mymigration",  
  "properties": {  
    "currentStatus": {  
      "currentSubStateDetails": {  
        "currentSubState": "WaitingForCutoverTrigger",  
        "dbDetails": {  
          "sportsdb_sample": {  
            "appliedChanges": 0,  
            "cdcDeleteCounter": 0,  
            "cdcInsertCounter": 0,  
            "cdcUpdateCounter": 0,  
            "databaseName": "db1",  
            "fullLoadCompletedTables": 1,  
            "fullLoadErroredTables": 0,  
            "fullLoadLoadingTables": 0,  
            "fullLoadQueuedTables": 0,  
            "incomingChanges": 0,  
            "initializationCompleted": true,  
            "latency": 0,  
            "migrationState": "READY_TO_COMPLETE",  
            "startedOn": "2021-08-13T14:02:40.3966618+00:00"  
          }  
        }  
      },  
      "state": "WaitingForUserAction"  
    },  
    "dbsToMigrate": [  
      {  
        "name": "db1",  
        "type": "FullLoad",  
        "status": "Completed",  
        "details": {  
          "fullLoadCompletedTables": 1,  
          "fullLoadErroredTables": 0,  
          "fullLoadLoadingTables": 0,  
          "fullLoadQueuedTables": 0,  
          "incomingChanges": 0,  
          "initializationCompleted": true,  
          "latency": 0,  
          "migrationState": "READY_TO_COMPLETE",  
          "startedOn": "2021-08-13T14:02:40.3966618+00:00"  
        }  
      }  
    ]  
  }  
}
```

Step 5: Point application to new target and clean up old resources.

Once the application is running successfully in the target flexible server platform, you can delete the AAD app and source single server Postgres.

Note: Transient resources like DMS, Migration VNet etc. will automatically be cleaned up when the migration completes (in all successful/error/cancel scenarios).

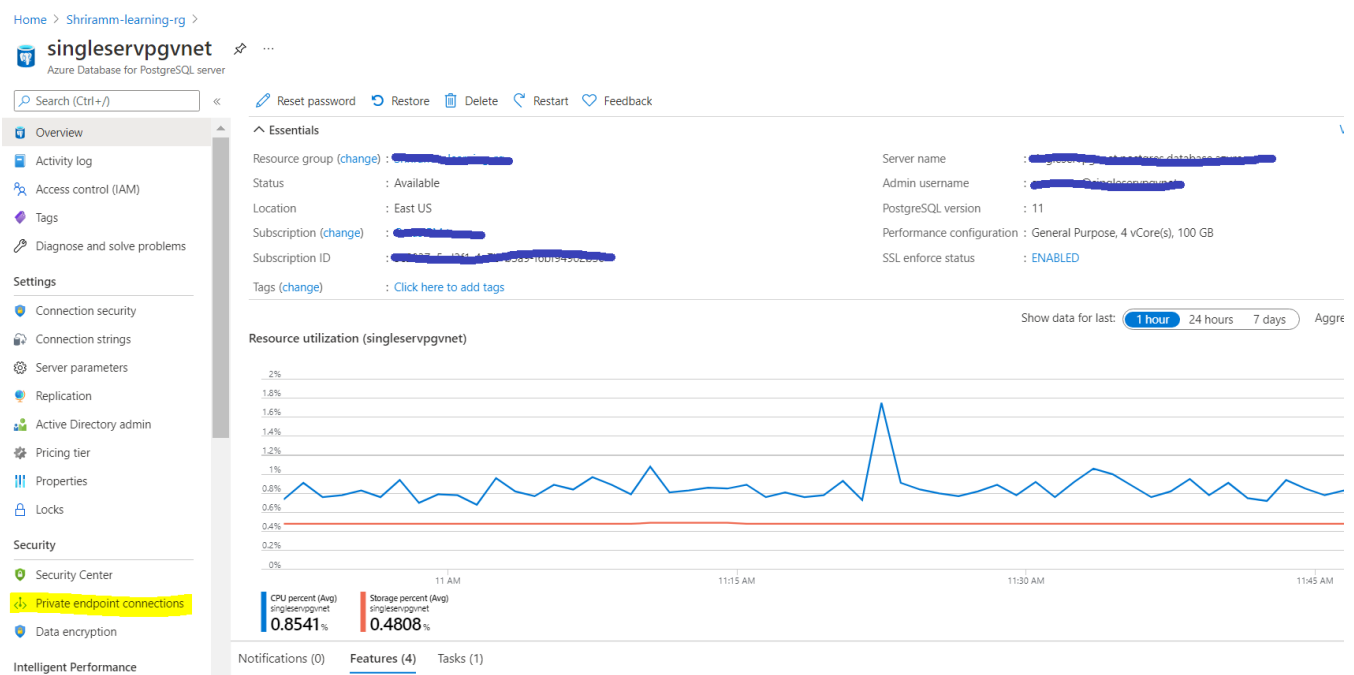
Source and Target servers – Private Access in the same Vnet

Download the [template for migrationBody.json for private access](#). There are placeholders that need to be filled with appropriate values pertaining to your subscription, resource groups, AAD privileges etc.

Let's assume that we need to migrate a sample **sportsdb_sample** database from a private access single server to private access flexible server.

The only way to make a PG single server to private access is to create a private end point connection.

Below is the source server screenshot from the Azure Portal:



Click on the Private endpoint connections (marked in yellow) in the above pic to see the list of private end point connections created to your single server. To learn more about private endpoint connections and the process to configure it, use the following [link](#).

Microsoft Azure (Preview) Report a bug Search resources, services, and docs (G+)

Home > singleservpgvnet

singleservpgvnet | Private endpoint connections

Azure Database for PostgreSQL server

Search (Ctrl+/) « + Private endpoint ✓ Approve ✕ Reject 🗑 Remove ↻ Refresh

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems

Settings

- Connection security
- Connection strings
- Server parameters
- Replication
- Active Directory admin
- Pricing tier
- Properties
- Locks

Security

- Security Center
- Private endpoint connections
- Data encryption

Intelligent Performance

Private Endpoint Connection

Private endpoint connections allow connections from within a Virtual Network to a private IP using the private endpoint feature. Connections using these private endpoints specified below provide access to singleservpgvnet

Search... 3 selected

<input type="checkbox"/> Connection name	State	Private endpoint name	Request/Response Message
<input checked="" type="checkbox"/> singleservpgvnet_endpoint-f4dd15ee-cd02-429e-bea3-d9014603666d	Approved	singleservpgvnet_endpoint	Auto-approved

So, the source's private end point has been created in Vnet called "**vnet_test**" under the subnet called "**source_subnet**". These details will be listed in the link which opens when user clicks on the private end point link (highlighted in yellow in the above pic)

Screenshot of the Connection Security tab of the single server looks like below.

Home > Shriramm-learning-rg > singlesevpvnet

singlesevpvnet | Connection security

Azure Database for PostgreSQL server

Search (Ctrl+/) Save Discard + Add client IP

Deny public network access

You may optionally choose to disable, but retain configuration for the firewall rules and virtual networks below. When 'Deny public network access' is set to yes, only private endpoint connections will be allowed to access this resource. [Learn more](#)

Deny public network access ☐ No ☒ Yes

Firewall rules

Connections from the IPs specified below provides access to all the databases in singlesevpvnet.

Allow access to Azure services ☐ No ☒ Yes

+ Add current client IP address (122.172.194.123) + Add 0.0.0.0 - 255.255.255.255

Firewall rule name	Start IP	End IP
Firewall rule name	Start IP	End IP

VNET rules + Adding existing virtual network + Create new virtual network

Rule name	Virtual network	Subnet	Address range	Endpoint status	Resource group	Subscription ID	State
No results.							

SSL settings

An important thing to note is that the **Deny public network access** is set to **Yes**. This in combination with the private endpoint connection puts your single server in private access.

As in the previous public access example, similar steps should be followed to create a target flexible server but with private access instead of public access. Here is a snapshot of the networking tab of the target flex server

Home > Shriramm-learning-rg > flexservervnet

flexservervnet | Networking

Azure Database for PostgreSQL flexible server

Search (Ctrl+/) Save Discard Download SSL Certificate Feedback

Encrypted connections

This server supports encrypted connections using Transport Layer Security (TLS). You can download the public SSL certificate from the above menu. For information on TLS version and certificates, refer to connecting with TLS/SSL. [Learn more](#)

Network connectivity

You can connect to your server by specifying a public IP address specified below or from within a selected virtual network.

Connectivity method ☐ Public access (allowed IP addresses) ☒ Private access (VNet Integration)

Connections from within the virtual network configured below will have access to this server. [Learn more](#)

Virtual network

Virtual networks are logically isolated from each other in Azure. Virtual network gives you a highly secure environment to run your PostgreSQL Flexible Server and other types of Azure resources

Subscription

Virtual network

Subnet

This subnet is delegated for use only with PostgreSQL Flexible Server (Microsoft.DBforPostgreSQL/flexibleServers).

Private DNS integration

Private DNS zone integration is required to connect to your Flexible Server in virtual network using server name (fully qualified domain name). A new private DNS zone will be created or you can optionally choose an existing one linked to the selected virtual network. With private DNS zone integration, the DNS records for the server name will be created.

As you can see the target is also in the vnet called **"vnet_test"** which is the same vnet as the single server but in a different subnet called **flex_subnet**.

Make sure the pre-requisites are taken care of like the previous public access example. **Since both source and target are in private access, you need to add the AAD privileges to the VNet as well**

CREATE API CALL:

```
az postgres flexible-server migration create --name --flexservervnet migration-name  
mymigration --properties "C:\Users\Desktop\migrationBody_privateAccess.json"
```

Contents of migrationBody_privateAccess.json:

```
{  
  "properties": {  
    "SourceDBServerResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-  
learning-rg/providers/Microsoft.DBforPostgreSQL/servers/singleservpgvnet",  
    "SecretParameters": {  
      "AdminCredentials": {  
        "SourceServerPassword": "<password>",  
        "TargetServerPassword": "<password>"  
      },  
      "AADApp": {  
        "ClientId": <ClientID>,  
        "TenantId": <TenantID>,  
        "AadSecret": <ClientSecret>  
      }  
    },  
    "MigrationResourceGroup": {  
      "ResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-rg",  
      "SubnetResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-  
learning-rg/providers/Microsoft.Network/virtualNetworks/vnet_test/subnets/source_subnet"  
    },  
    "DBsToMigrate": [  
      "sportsdb_sample"  
    ],  
    "OverwriteDBsInTarget": "true"  
  }  
}
```

Note: Portions highlighted are replaced for security reasons.
singleservpgvnet = Source Single Server Postgres

API CALL RESPONSE:

```
{  
  "id": "/subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/flexservervnet/migrations/70300fc3-88f0-411f-9113-573ac5d1f811",  
  "location": "East US",  
  "name": "mymigration",  
  "properties": {  
    "currentStatus": {  
      "state": "InProgress"  
    },  
    "dbsToMigrate": [  
      "sportdb_sample"  
    ],  
    "migrationDetailsLevel": "Default",  
    "migrationId": "70300fc3-88f0-411f-9113-573ac5d1f811",  
    "migrationName": "mymigration",  
    "migrationResourceGroup": {  
      "resourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-rg"  
    },  
    "migrationWindowStartTimeInUtc": "2021-08-13T15:02:09.2035307Z",  
    "overwriteDBsInTarget": true,  
    "sourceDBServerResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/servers/singleservpgvnet",  
    "targetDBServerResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/flexservervnet"  
  },  
  "resourceGroup": "Shriramm-learning-rg",  
  "type": "Microsoft.DBforPostgreSQL/flexibleServers/migrations"  
}
```

There is a new field in the json file namely **SubnetResourceId**. This should be the subnet where DMS needs to be created by the automation service. **This could be the subnet of the single server or different subnet under the same Vnet.**

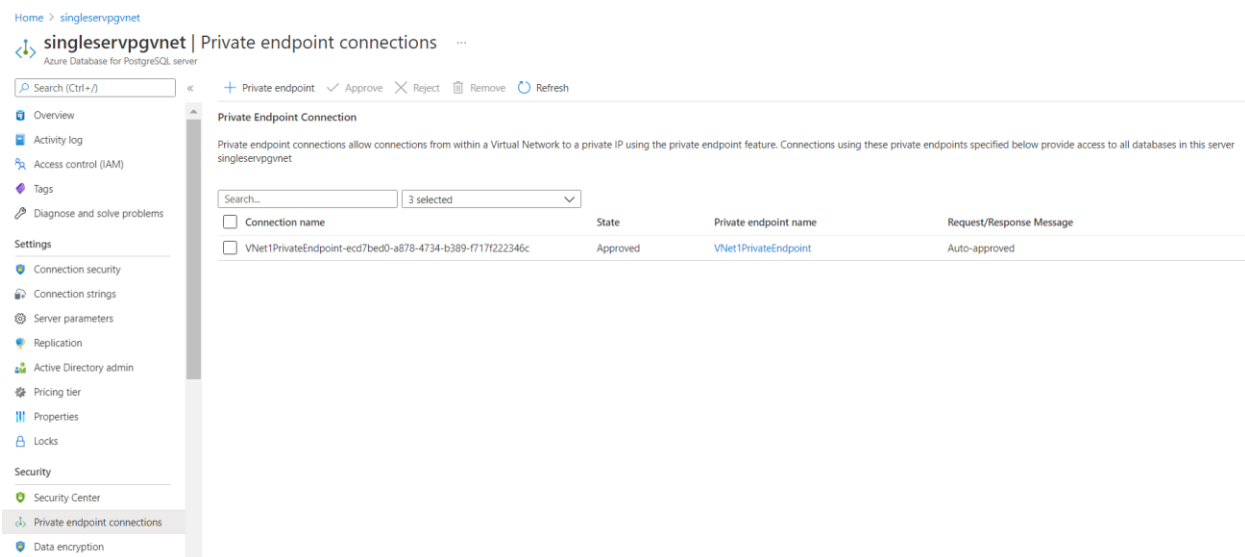
Follow similar steps as in public access to follow the migration status and to do the cut over.

Source and Target servers – Private Access in the different Vnets

Download the [template for migrationBody.json for private access](#). There are placeholders that needs to be filled with appropriate values pertaining to your subscription, resource groups, AAD privileges etc.

Let's assume that we need to migrate a sample **sportsdb sample** database from a private access single server to private access flexible server deployed in another VNet.

Below is snapshot of the private end point on the source.



The single server has a private end point connection to “vnet1” and is under the “sourcesubnet” subnet.

As in the previous example, similar steps should be followed to create target flexible server with private access. Here is a snapshot of the networking tab of the target flexible server.

Home > flexserver1

flexserver1 | Networking

Azure Database for PostgreSQL flexible server

Search (Ctrl+/) << Save Discard Download SSL Certificate Feedback

- Overview
- Activity log
- Access control (IAM)
- Tags

Settings

- Compute + storage
- Networking**
- Connection strings
- Server parameters
- Maintenance
- High availability
- Locks

Monitoring

- Alerts
- Metrics
- Diagnostic settings
- Logs

Automation

Encrypted connections

This server supports encrypted connections using Transport Layer Security (TLS). You can download the public SSL certificate from the above menu. For information on TLS version and certificates, refer to connecting with TLS/SSL. [Learn more](#)

Network connectivity

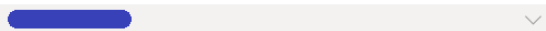
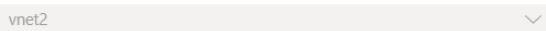
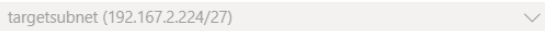
You can connect to your server by specifying a public IP address specified below or from within a selected virtual network.

- Connectivity method ⓘ
- ☐ Public access (allowed IP addresses)
 - ☒ Private access (VNet Integration)

Connections from within the virtual network configured below will have access to this server. [Learn more](#)

Virtual network

Virtual networks are logically isolated from each other in Azure. Virtual network gives you a highly secure environment to run your PostgreSQL Flexible Server.

- Subscription ⓘ 
- Virtual network ⓘ  vnet2
- Subnet ⓘ  targetsubnet (192.167.2.224/27)

This subnet is delegated for use only with PostgreSQL Flexible Server (Microsoft.DBforPostgreSQL/flexibleServers).

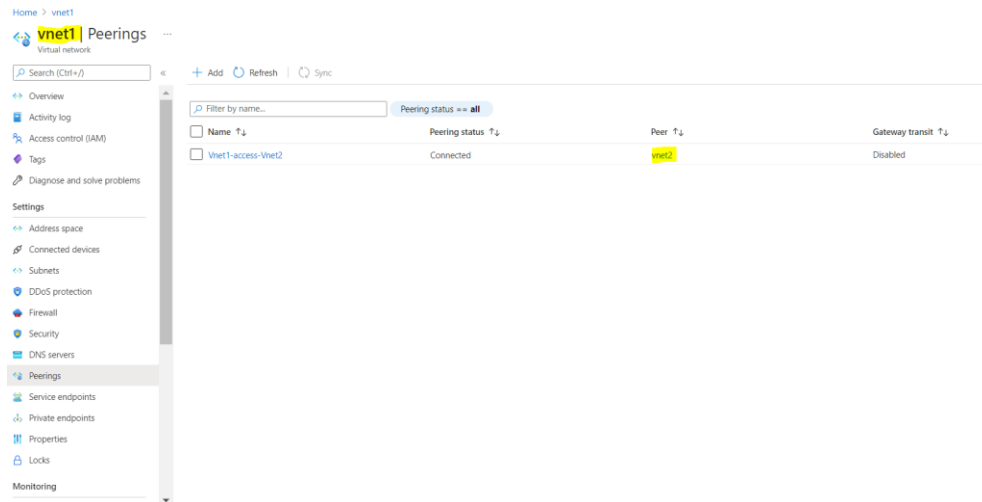
Private DNS integration

The target flex server is in “**vnet2**” under the “**targetsubnet**” subnet.

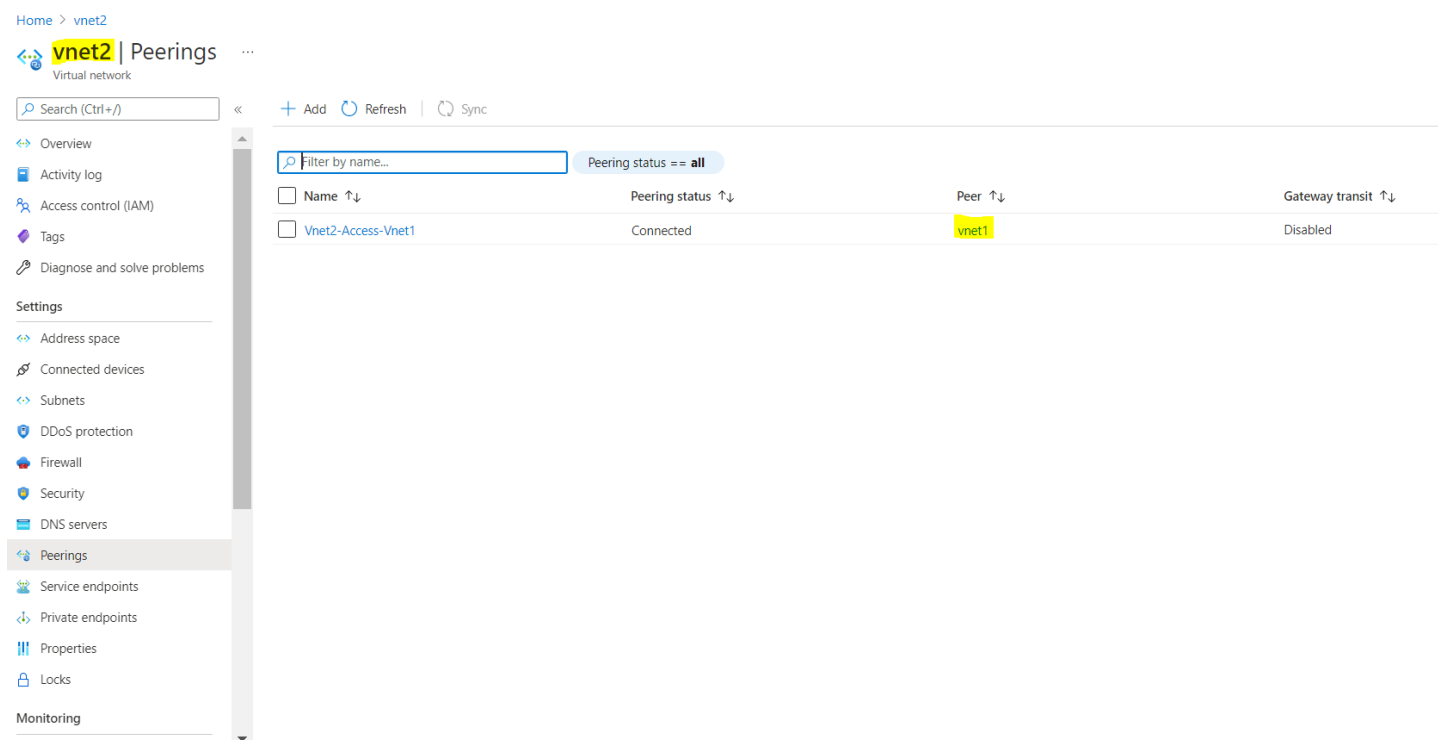
In this scenario, DMS can be created in either **vnet1 (source vnet)** or **vnet2 (target vnet)**. If created in **vnet1**, DMS can be in the same subnet as the single server(**sourcesubnet**) or in a different subnet of **vnet1**. If DMS is created in **vnet2**, then it needs to be in a different subnet than the subnet of the flex server (**targetsubnet**). This is because flexible server needs a delegated subnet and other resources cannot be added to the subnet.

In order to establish communication using private IP addresses between two different VNets, we need to establish VNet Peering. To learn more about VNet peering and the steps needed to establish peering between vnets, use the following [link](#).

After successful establishment of vnet peering, snapshot of **vnet1** should look like below



Snapshot of **vnet2** should look like below



When a flexible server is deployed in private access, there is a private DNS zone that gets automatically created. The private DNS zone integration is required to connect to your flexible server in a virtual network using the fully qualified domain name of the flexible server.

Home > flexserver1

flexserver1 | Networking

Azure Database for PostgreSQL flexible server

Search (Ctrl+/) Save Discard Download SSL Certificate Feedback

- Overview
- Activity log
- Access control (IAM)
- Tags
- Settings
 - Compute + storage
 - Networking**
 - Connection strings
 - Server parameters
 - Maintenance
 - High availability
 - Locks
- Monitoring
 - Alerts
 - Metrics
 - Diagnostic settings
 - Logs
- Automation

Connectivity method

☐ Public access (allowed IP addresses)

☒ Private access (VNet Integration)

Connections from within the virtual network configured below will have access to this server. [Learn more](#)

Virtual network

Virtual networks are logically isolated from each other in Azure. Virtual network gives you a highly secure environment to run your PostgreSQL Flexible Server and other types of Azure resources

Subscription vnet2

Virtual network vnet2

Subnet targetsubnet (192.167.2.224/27)

This subnet is delegated for use only with PostgreSQL Flexible Server (Microsoft.DBforPostgreSQL/flexibleServers).

Private DNS integration

Private DNS zone integration is required to connect to your Flexible Server in virtual network using server name (fully qualified domain name). A new private DNS zone will be created or you can optionally choose an existing one linked to the selected virtual network. With private DNS zone integration, the DNS records for the server name will be updated automatically in case the IP address of your Flexible Server changes. [Learn more](#)

Subscription

Private DNS zone flexserver1.private.postgres.database.azure.com

In order to establish connection between peered virtual networks, we need to link the private DNS zone of the flexible server to the VNet of the single server. In order to achieve this, navigate to the resource group in which your flexible server is deployed and search for the Private DNS zone that got auto provisioned during the creation of flexible server.

Home > Shriramm-learning-rg >

flexserver1.private.postgres.database.azure.com

Private DNS zone

Search (Ctrl+/) Record set Move Delete zone Refresh

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Settings
 - Virtual network links**
 - Properties
 - Locks
- Monitoring
 - Alerts
 - Metrics
- Automation
 - Tasks (preview)
 - Export template
- Support + troubleshooting
 - New Support Request

Essentials

Resource group (change) : shriram-learning-rg

Subscription (change) : Orcas PM team

Subscription ID : 5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30

Tags (change) : [Click here to add tags](#)

You can search for record sets that have been loaded on this page. If you don't see what you're looking for, you can try scrolling to allow more record sets to load.

Search record sets

Name	Type	TTL	Value	Auto registered
@	SOA	3600	Email: azureprivatedns-host.microsoft.com Host: azureprivatedns.net Refresh: 3600 Retry: 300 Expire: 2419200 Minimum TTL: 10 Serial number: 1	False
d67a343a0ea5	A	30	192.167.2.228	False
vm-acvee4pjiemr	A	10	192.167.1.229	True

Click on the virtual network links to link the single server's virtual network (**vnet1**) to the private DNS Zone.

Home > [Subscription](#) > [flexserver1.private.postgres.database.azure.com](#) >

Add virtual network link ...

flexserver1.private.postgres.database.azure.com

Link name *

AddingVNet1



Virtual network details



Only virtual networks with Resource Manager deployment model are supported for linking with Private DNS zones. Virtual networks with Classic deployment model are not supported.

☐ I know the resource ID of virtual network ⓘ

Subscription * ⓘ

[Subscription](#)



Virtual network *

vnet1 (flexserver1.private.postgres.database.azure.com)



Configuration

☒ Enable auto registration ⓘ

OK

Now vnet1 has been linked to the private DNS zone of the flex server and communication can now be established between the source and target vnets (vnet1 and vnet2). Also make sure AAD app privileges are added to both source and target VNets.

CREATE API CALL:

```
az postgres flexible-server migration create --name flexserver1 --migration-name  
mymigration --properties "C:\Users\Desktop\migrationBodyprivateAccess.json"
```

Contents of migrationBodyprivateAccess.json:

```
{  
  "properties": {  
    "SourceDBServerResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/servers/singleservpgvnet",  
    "SecretParameters": {  
      "AdminCredentials": {  
        "SourceServerPassword": "SourceAdminPassword",  
        "TargetServerPassword": "TargetAdminPassword"  
      },  
      "AADApp": {  
        "ClientId": "ClientAAD",  
        "TenantId": "TenantAAD",  
        "AadSecret": "SecretAAD"  
      }  
    },  
    "DBsToMigrate": [  
      "sportsdb_sample"  
    ],  
    "MigrationResourceGroup": {  
      "SubnetResourceID": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.Network/virtualnetworks/vnet1/subnets/sourcesubnet"  
    }  
  }  
}
```

Note: Portions highlighted are replaced for security reasons.

singleservpgvnet = Source Single Server

API CALL RESPONSE:

```
{  
  "id": "/subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/flexserver1/migrations/44fecf94-65a6-46f8-813a-44756ce0e089",  
  "location": "East US",  
  "name": "mymigration",  
  "properties": {  
    "currentStatus": {  
      "state": "InProgress"  
    },  
    "dbsToMigrate": [  
      "sportsdb_sample"  
    ],  
    "migrationDetailsLevel": "Default",  
    "migrationId": "44fecf94-65a6-46f8-813a-44756ce0e089",  
    "migrationName": "mymigration",  
    "migrationWindowStartTimeInUtc": "2021-08-13T16:45:49.5364934Z",  
    "overwriteDBsInTarget": true,  
    "sourceDBServerResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/servers/singleservpgvnet",  
    "targetDBServerResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/flexserver1",  
    "targetDBServerSubnetResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-  
learning-rg/providers/Microsoft.Network/virtualnetworks/vnet01/subnets/singleserversubnet"  
  },  
  "resourceGroup": "Shriramm-learning-rg",  
}
```

In the above command, the DMS was created in the Vnet of the source and in the same subnet as the single server. This is denoted by the following parameter.

```
SubnetResourceId": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-rg/providers/Microsoft.Network/virtualnetworks/vnet1/subnets/sourcesubnet"
```

Similarly, DMS can be created in the Vnet of flexible server in a different subnet and can migrate the data from the source to target. The same concept can be extended to put the DMS in a VNet that is different from the source and target VNets. If peering is established between Source and DMS VNet as well as target and DMS VNet and their private DNS zones are linked, migration can be established between the source and target.

Follow the similar steps as in previous sections to follow the migration status and to do the cut over.

[*Public Source with Virtual Network Service EndPoints and Private target*](#)

Download the [template for migrationBody.json for private access](#). There are placeholders that need to be filled with appropriate values pertaining to your subscription, resource groups, AAD privileges etc.

Let's assume that we need to migrate a sample **sportsdb_sample** database from a public access PG single server to private access flexible server deployed in a VNet. In this case there is a virtual network service endpoint rule created on the single server which allows access to a particular subnet mentioned in the rule. You can add multiple such rules on the single server to allow access to many subnets.

The Connection Security tab for the source server is shown below

Home > Shriramm-learning-rg > [Server Name] > Connection security

Search (Ctrl+F) Save Discard Add client IP

You may optionally choose to disable, but retain configuration for the firewall rules and virtual networks below. When 'Deny public network access' is set to yes, only private endpoint connections will be allowed to access this resource. [Learn more](#)

Deny public network access: ☒ No ☐ Yes

Firewall rules

Connections from the IPs specified below provides access to all the databases in shrirammserver.

Allow access to Azure services: ☒ No ☒ Yes

+ Add current client IP address (122.172.194.123) + Add 0.0.0.0 - 255.255.255.255

Firewall rule name	Start IP	End IP
Firewall rule name	Start IP	End IP

VNET rules + Adding existing virtual network + Create new virtual network

Rule name	Virtual network	Subnet	Address range	Endpoint status	Resource group	Subscription ID	State
pvtnetworkLink	pvtnetwork	sourcesubnet	172.16.1.0/24	Enabled	Shriramm-learning-rg	Orcas PM team	Ready

SSL settings

The source server allows public network access and there is also a VNet service endpoint Rule enabled which gives access to the "sourcesubnet" subnet of the "pvtnetwork" vnet. The target flexible server can be in the same pvtnetwork vnet in a delegated subnet and DMS can be deployed in the subnet described in the service endpoint rule(sourcesubnet). Snapshot of the target server is given below.



Search (Ctrl+/)



Save



Discard



Download SSL Certificate



Feedback

Overview

Activity log

Access control (IAM)

Tags

Settings

Compute + storage

Networking

Connection strings

Server parameters

Maintenance

High availability

Locks

Monitoring

Alerts

Metrics

Diagnostic settings

Logs

Automation

Tasks (preview)

Encrypted connections

This server supports encrypted connections using Transport Layer Security (TLS). You can download the public SSL certificate from the above menu. For information on TLS version and certificates, refer to connecting with TLS/SSL. [Learn more](#)

Network connectivity

You can connect to your server by specifying a public IP address specified below or from within a selected virtual network.

Connectivity method ⓘ



Public access (allowed IP addresses)



Private access (VNet Integration)



Connections from within the virtual network configured below will have access to this server. [Learn more](#)

Virtual network

Virtual networks are logically isolated from each other in Azure. Virtual network gives you a highly secure environment to run your PostgreSQL Flexible Server and other types of Azure resources

Subscription ⓘ

Orcas PM team



Virtual network ⓘ

pvtnetwork



Subnet ⓘ

flexsubnet (172.16.2.0/24)



This subnet is delegated for use only with PostgreSQL Flexible Server (Microsoft.DBforPostgreSQL/flexibleServers).

Private DNS integration

Private DNS zone integration is required to connect to your Flexible Server in virtual network using server name (fully qualified domain name).

A new private DNS zone will be created or you can optionally choose an existing one linked to the selected virtual network. With private DNS zone integration, the DNS records for the server name

With this network topology, the DMS deployed on the subnet in the service endpoint rule can now access the public single server and can also access the target flex server deployed inside another subnet in the same vnet and can migrate data from source to target.

CREATE API CALL:

```
az postgres flexible-server migration create --name privateflex --migration-name  
mymigration --properties "C:\Users\Desktop\migrationBody.json"
```

Contents of migrationBody.json:

```
{  
  "properties": {  
    "SourceDBServerResourceID": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-  
learning-rg/providers/Microsoft.DBforPostgreSQL/servers/shriramssingleserver",  
    "SecretParameters": {  
      "AdminCredentials": {  
        "SourceServerPassword": "SourceAdminPassword",  
        "TargetServerPassword": "TargetAdminPassword"  
      },  
      "AADApp": {  
        "ClientId": "ClientAAD",  
        "TenantId": "TenantAAD",  
        "AadSecret": "SecretAAD"  
      }  
    },  
    "DBsToMigrate": [  
      "sportsdb_sample"  
    ],  
    "MigrationResourceGroup": {  
      "SubnetResourceID": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.Network/virtualnetworks/pvtnetwork/subnets/sourcesubnet"  
    }  
  }  
}
```

Note: Portions highlighted are replaced for security reasons.

shriramssingleserver = Source Single Server Postgres

API CALL RESPONSE:

```
{  
  "id": "/subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/privateflex/migrations/f5bd091e-f9f4-4bdc-b73d-6d3c220ac014",  
  "location": "East US",  
  "name": "mymigration",  
  "properties": {  
    "currentStatus": {  
      "state": "InProgress"  
    },  
    "DBsToMigrate": [  
      "sportsdb_sample"  
    ],  
    "migrationDetailsLevel": "Default",  
    "migrationId": "f5bd091e-f9f4-4bdc-b73d-6d3c220ac014",  
    "migrationName": "mymigration",  
    "migrationWindowStartTimeInUtc": "2021-08-13T17:23:11.1605524Z",  
    "overwriteDBsInTarget": true,  
    "sourceDBServerResourceID": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/servers/shriramssingleserver",  
    "targetDBServerResourceID": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-learning-  
rg/providers/Microsoft.DBforPostgreSQL/flexibleServers/privateflex",  
    "targetDBServerSubnetResourceID": "subscriptions/5c5037e5-d3f1-4e7b-b3a9-f6bf94902b30/resourceGroups/Shriramm-  
learning-rg/providers/Microsoft.Network/virtualnetworks/pvtnetwork/subnets/thirdsubnet"  
  },  
  "resourceGroup": "Shriramm-learning-rg",  
}
```

Follow the similar steps as in previous sections to follow the migration status and to do the cut over.

Current Limitations

- All [DMS-based limitations](#) apply to this automated solution as well.
- This is a logical replication solution that will use resources in the source single server instance. Plan to scale resources (CPU, memory, and storage IOPS) accordingly.
- All [limitations](#) applicable to logical replication are applicable to the solution as well.
- You can migrate up to 8 databases per server in a single migration attempt. If you have more than 8 databases to migrate, you can create multiple migration attempts. Migrating more than 8 databases in parallel from a source single server instance may put extra load on the source server.
- An AAD App with appropriate privileges is required for this automated solution to work. The solution cannot be used without an AAD App.
- You can migrate databases up to a size of 1 TB with this automated solution.
- There will be a short downtime to the application during cutover. The amount of downtime depends on other post-migration tasks that may need to be performed after attempting the cutover.
- This solution migrates data and schema for the database. It does not validate the data being moved. We recommend going through test runs of the migration and perform an end-to-end test of the application in the target flexible server platform before cutting over workloads in production.
- This solution requires the target database server with flexible server deployment option be pre-created before the migration commences.
- This solution does not migrate other managed service features such as server parameters, connection security details, azure_sys DB, etc. In other words, everything except data and schema must be manually copied over.
- This solution only migrates user databases and not system databases like “**template_0**,” “**template_1**”.

Downloads

Download the json files used in the **create migration** command suitable to your network topology using the following link.

- [Public Access \(both source and target\)](#)
- [Private Access \(either source using private end point or target is inside a VNet\)](#)

Appendix

When the migration moves into “**WaitingforUserAction**” state, the **update** command along with its options can be used to move the migration to the next state.

- a. **WaitingForDBsToMigrateSpecification** - The migration is waiting to know information on DBs to migrate. This happens in cases where DBsToMigrate parameter was not included in migrationBody.json for the **create migration** command. This can be enabled by the following CLI command

```
az postgres flexible-server migration update --subscription <sub_id> --resource-group <rg_name> --name <target_server_name> --migration-name <migration_name> --db-names db1 db2 db3
```

Please note that you can migrate up to 8 databases per server in a single migration attempt. If you have more than 8 databases to migrate, you can create multiple migration attempts

- b. **WaitingForTargetDBOverwriteConfirmation** - The migration is waiting for confirmation on target overwrite as data is present in the target database being migrated into. This can be enabled by the following CLI command.

```
az postgres flexible-server migration update --subscription <sub_id> --resource-group <rg_group> --name <target_server_name> --migration-name <migration_name> --overwrite-dbs true
```